

UNIT – I

Data Foundation – SCSA1309

UNIT 1 INTRODUCTION

Overview of Data: Definition - Types of data – Quantitative and Qualitative (Nominal, Ordinal, Discrete and Continuous) Big Data: Structured, Unstructured and semi-structured - Metadata: Concepts of metadata – Types of metadata – Uses Data Source: Enterprise Data Source, Social Media Data Source, Public Data Source – Web Scrapping- Basic Concepts of Data Warehouse and Data Mining – Distributed File System.

I. Introduction

DATA : Data is defined as facts or figures, or information that's stored in or used by a computer. An example of data is information collected for a research paper. An example of data is an email.

Types of Data

Understanding the different types of data (in statistics, marketing research, or data science) allows you to pick the data type that most closely matches your needs and goals. Whether you are a businessman, marketer, data scientist, or another professional who works with some kinds of data, you should be familiar with the key list of data types. Why? Because the various data classifications allow you to correctly use measurements and thus to correctly make decisions.

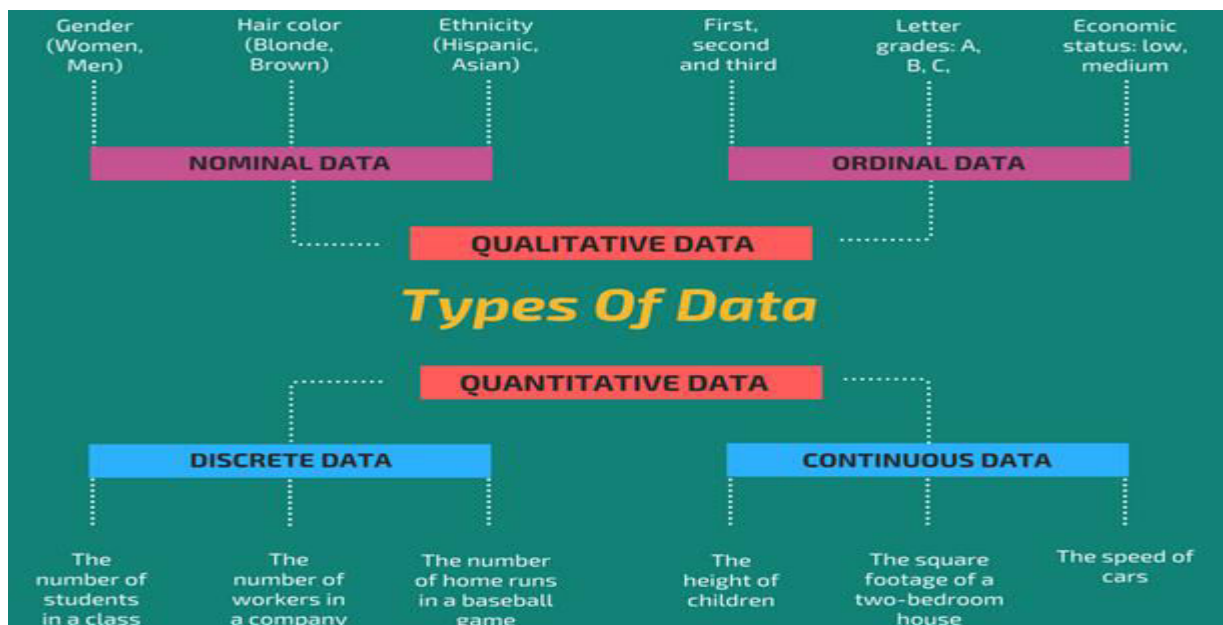


Fig 1.1 Types of Data

Qualitative vs Quantitative Data

1. Quantitative data

Quantitative data seems to be the easiest to explain. It answers key questions such as “how many,” “how much” and “how often”.

Quantitative data can be expressed as a number or can be quantified. Simply put, it can be measured by numerical variables.

Quantitative data are easily amenable to statistical manipulation and can be represented by a wide variety of statistical types of graphs and charts such as line, bar graph, scatter plot, and etc.

Examples of quantitative data:

- Scores on tests and exams e.g. 85, 67, 90 and etc.
- The weight of a person or a subject.
- Your shoe size.
- The temperature in a room.

There are 2 general types of quantitative data: discrete data and continuous data. We will explain them later in this article.

2. Qualitative data

Qualitative data can't be expressed as a number and can't be measured. Qualitative data consist of words, pictures, and symbols, not numbers.

Qualitative data is also called categorical data because the information can be sorted by category, not by number.

Qualitative data can answer questions such as “how this has happened” or and “why this has happened”.

Examples of qualitative data:

- Colors e.g. the color of the sea
- Your favorite holiday destination such as Hawaii, New Zealand and etc.
- Names as John, Patricia,.....
- Ethnicity such as American Indian, Asian, etc.

More you can see on our post qualitative vs quantitative data.

There are 2 general types of qualitative data: nominal data and ordinal data. We will explain them after a while.



Fig 1.2 Quantitative Vs Qualitative Data

Nominal vs Ordinal Data

3. Nominal data

Nominal data is used just for labeling variables, without any type of quantitative value. The name 'nominal' comes from the Latin word "nomen" which means 'name'.

The nominal data just name a thing without applying it to order. Actually, the nominal data could just be called "labels."

Examples of Nominal Data:

- Gender (Women, Men)
- Hair color (Blonde, Brown, Brunette, Red, etc.)
- Marital status (Married, Single, Widowed)
- Ethnicity (Hispanic, Asian)

As you see from the examples there is no intrinsic ordering to the variables.

Eye color is a nominal variable having a few categories (Blue, Green, Brown) and there is no way to order these categories from highest to lowest.

4. Ordinal data

Ordinal data shows where a number is in order. This is the crucial difference from nominal types of data.

Ordinal data is data which is placed into some kind of order by their position on a scale. Ordinal data may indicate superiority.

However, you cannot do arithmetic with ordinal numbers because they only show sequence. Ordinal variables are considered as “in between” qualitative and quantitative variables.

In other words, the ordinal data is qualitative data for which the values are ordered.

In comparison with nominal data, the second one is qualitative data for which the values cannot be placed in an ordered.

We can also assign numbers to ordinal data to show their relative position. But we cannot do math with those numbers. For example: “first, second, third...etc.”

Examples of Ordinal Data:

- The first, second and third person in a competition.
- Letter grades: A, B, C, and etc.
- When a company asks a customer to rate the sales experience on a scale of 1-10.
- Economic status: low, medium and high.

Much more on the topic plus a quiz, you can learn in our post: nominal vs ordinal data.

Download the following infographic in PDF



Fig 1.3 Nominal Vs Ordinal Data

Discrete vs Continuous Data

As we mentioned above discrete and continuous data are the two key types of quantitative data. In statistics, marketing research, and data science, many decisions depend on whether the basic data is discrete or continuous.

5. Discrete data

Discrete data is a count that involves only integers. The discrete values cannot be subdivided into parts. For example, the number of children in a class is discrete data. You can count whole individuals. You can't count 1.5 kids. To put in other words, discrete data can take only certain values. The data variables cannot be divided into smaller parts.

It has a limited number of possible values e.g. days of the month.

Examples of discrete data:

- The number of students in a class.
- The number of workers in a company.
- The number of home runs in a baseball game.
- The number of test questions you answered correctly

6. Continuous data

Continuous data is information that could be meaningfully divided into finer levels. It can be measured on a scale or continuum and can have almost any numeric value. For example, you can measure your height at very precise scales — meters, centimeters, millimeters and etc.

You can record continuous data at so many different measurements – width, temperature, time, and etc. This is where the key difference from discrete types of data lies. The continuous variables can take any value between two numbers. For example, between 50 and 72 inches, there are literally millions of possible heights: 52.04762 inches, 69.948376 inches and etc. A good great rule for defining if a data is continuous or discrete is that if the point of measurement can be reduced in half and still make sense, the data is continuous.

Examples of continuous data:

- The amount of time required to complete a project.
- The height of children.
- The square footage of a two-bedroom house.
- The speed of cars.

Much more on the topic you can see in our detailed post discrete vs continuous data: with a comparison chart.

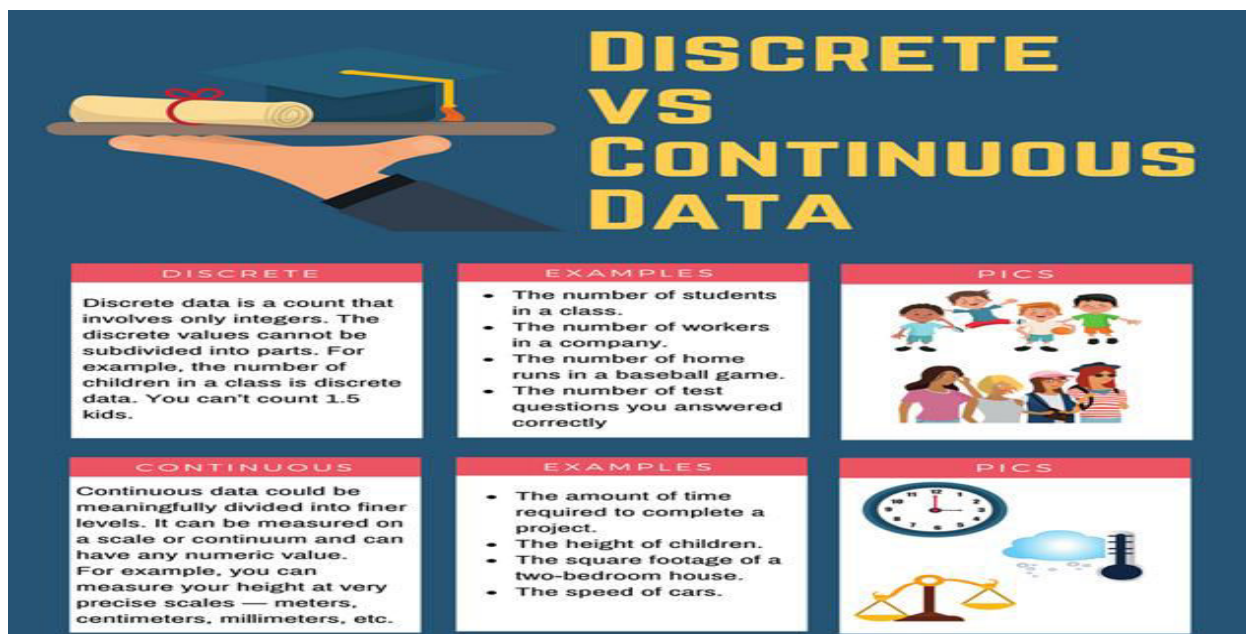


Fig 1.4 Discrete Vs Continuous Data

Conclusion

All of the different types of data have a critical place in statistics, research, and data science. Data types work great together to help organizations and businesses from all industries build successful data-driven decision-making process. Working in the data management area and having a

good range of data science skills involves a deep understanding of various types of data and when to apply them.

I. Big Data

Big data is a combination of structured, semistructured and unstructured data collected by organizations that can be mined for information and used in machine learning projects, predictive modeling and other advanced analytics applications. Systems that process and store big data have become a common component of data management architectures in organizations, combined with tools that support big data analytics uses. Big data is often characterized by the three V's:

- The large *volume* of data in many environments;
- the wide *variety* of data types frequently stored in big data systems; and
- the *velocity* at which much of the data is generated, collected and processed.

These characteristics were first identified in 2001 by Doug Laney, then an analyst at consulting firm Meta Group Inc.; Gartner further popularized them after it acquired Meta Group in 2005. More recently, several other V's have been added to different descriptions of big data, including *veracity*, *value* and *variability*. Although big data doesn't equate to any specific volume of data, big data deployments often involve terabytes, petabytes and even exabytes of data created and collected over time.

Importance of big data

Companies use big data in their systems to improve operations, provide better customer service, create personalized marketing campaigns and take other actions that, ultimately, can increase revenue and profits. Businesses that use it effectively hold a potential competitive advantage over those that don't because they're able to make faster and more informed business decisions.

For example, big data provides valuable insights into customers that companies can use to refine their marketing, advertising and promotions in order to increase customer engagement and conversion rates. Both historical and real-time data can be analyzed to assess the evolving

preferences of consumers or corporate buyers, enabling businesses to become more responsive to customer wants and needs.

Big data is also used by medical researchers to identify disease signs and risk factors and by doctors to help diagnose illnesses and medical conditions in patients. In addition, a combination of data from electronic health records, social media sites, the web and other sources gives healthcare organizations and government agencies up-to-date information on infectious disease threats or outbreaks.

Here are some more examples of how big data is used by organizations:

- In the energy industry, big data helps oil and gas companies identify potential drilling locations and monitor pipeline operations; likewise, utilities use it to track electrical grids.
- Financial services firms use big data systems for risk management and real-time analysis of market data.
- Manufacturers and transportation companies rely on big data to manage their supply chains and optimize delivery routes.
- Other government uses include emergency response, crime prevention and smart city initiatives.



Fig 1.5 Benefits of Big Data

These are some of the business benefits organizations can get by using big data. Examples of big data.

Big data comes from myriad sources -- some examples are transaction processing systems, customer databases, documents, emails, medical records, internet clickstream logs, mobile apps and social networks. It also includes machine-generated data, such as network and server log files and data from sensors on manufacturing machines, industrial equipment and internet of things devices.

In addition to data from internal systems, big data environments often incorporate external data on consumers, financial markets, weather and traffic conditions, geographic information, scientific research and more. Images, videos and audio files are forms of big data, too, and many big data applications involve streaming data that is processed and collected on a continual basis.

Breaking down the V's of big data

Volume is the most commonly cited characteristic of big data. A big data environment doesn't have to contain a large amount of data, but most do because of the nature of the data being collected and stored in them. Clickstreams, system logs and stream processing systems are among the sources that typically produce massive volumes of data on an ongoing basis.

Big data also encompasses a wide variety of data types, including the following:

- structured data, such as transactions and financial records;
- unstructured data, such as text, documents and multimedia files; and
- semistructured data, such as web server logs and streaming data from sensors.

Various data types may need to be stored and managed together in big data systems. In addition, big data applications often include multiple data sets that may not be integrated upfront. For example, a big data analytics project may attempt to forecast sales of a product by correlating data on past sales, returns, online reviews and customer service calls.

Velocity refers to the speed at which data is generated and must be processed and analyzed. In many cases, sets of big data are updated on a real- or near-real-time basis, instead of the daily, weekly or

monthly updates made in many traditional data warehouses. Managing data velocity is also important as big data analysis further expands into machine learning and artificial intelligence (AI), where analytical processes automatically find patterns in data and use them to generate insights.

More characteristics of big data

Looking beyond the original three V's, here are details on some of the other ones that are now often associated with big data:

- Veracity refers to the degree of accuracy in data sets and how trustworthy they are. Raw data collected from various sources can cause data quality issues that may be difficult to pinpoint. If they aren't fixed through data cleansing processes, bad data leads to analysis errors that can undermine the value of business analytics initiatives. Data management and analytics teams also need to ensure that they have enough accurate data available to produce valid results.
- Some data scientists and consultants also add value to the list of big data's characteristics. Not all the data that's collected has real business value or benefits. As a result, organizations need to confirm that data relates to relevant business issues before it's used in big data analytics projects.
- Variability also often applies to sets of big data, which may have multiple meanings or be formatted differently in separate data sources -- factors that further complicate big data management and analytics.

Some people ascribe even more V's to big data; various lists have been created with between seven and 10.

The six Vs of big data

Big data is a collection of data from various sources, often characterized by what's become known as the 3Vs: *volume*, *variety* and *velocity*. Over time, other Vs have been added to descriptions of big data:







VOLUME	VARIETY	VELOCITY	VERACITY	VALUE	VARIABILITY
The amount of data from myriad sources.	The types of data: structured, semi-structured, unstructured.	The speed at which big data is generated.	The degree to which big data can be trusted.	The business value of the data collected.	The ways in which the big data can be used and formatted.
					

Fig 1.6 Vs of Big Data

The characteristics of big data are commonly described by using words that begin with 'v,' including these six.

II. Meta Data

Metadata is simply **data about data**. It means it is a description and context of the data. It helps to organize, find and understand data.

Some typical metadata elements:

1. Title and description,
2. Tags and categories,
3. Who created and when,
4. Who last modified and when,
5. Who can access or update.

Types of metadata:

Actually, there are only three main types, but it's important to understand each type and how they function to make your assets more easily discoverable. So, if you're not sure what the difference is between structural metadata, administrative metadata, and descriptive metadata (spoiler alert: those are the three main types of metadata), let's clear up the confusion.

Structural Metadata

Let's start with the basics. Structural metadata is data that indicates how a digital asset is organized, such as how pages in a book are organized to form chapters, or the notes that make up a notebook in Evernote or OneNote. Structural metadata also indicates whether a particular asset is part of a single collection or multiple collections and facilitates the navigation and presentation of information in an electronic resource. Examples include:

- Page numbers
- Sections
- Chapters
- Indexes
- Table of contents

Beyond basic organization, structural metadata is the key to documenting the relationship between two assets. For example, it's used to indicate that a specific stock photo was used in a particular sales brochure, or that one asset is a raw, unedited version of another.

Administrative Metadata

Administrative metadata relates to the technical source of a digital asset. It includes data such as the file type, as well as when and how the asset was created. This is also the type of metadata that relates to usage rights and intellectual property, providing information such as the owner of an asset, where and how it can be used, and the duration a digital asset can be used for those allowable purposes under the current license.

The National Information Standards Organization (NISO) actually breaks administrative metadata down into three sub-types:

- **Technical Metadata** – Information necessary for decoding and rendering files

- **Preservation Metadata** – Information necessary for the long-term management and archiving of digital assets
- **Rights Metadata** – Information pertaining to intellectual property and usage rights

A Creative Commons license, for instance, is administrative metadata. Other examples include the date a digital asset was created, and for photos, administrative data might include the camera model used to take the photo, light source, and resolution. In addition, administrative metadata is used to indicate who can access a digital asset, the key to effective permissions management in a DAM system.

Descriptive Metadata

Descriptive metadata is essential for discovering and identifying assets. Why? It's information that describes the asset, such as the asset's title, author, and relevant keywords. Descriptive metadata is what allows you to locate a book in a particular genre published after 2016, for instance, as a book's metadata would include both genre and publication date. In fact, the ISBN system is a good example of an early effort to use metadata to centralize information and make it easier to locate resources (in this case, books in a traditional library).

Essentially, descriptive metadata includes any information describing the asset that can be used for later identification and discovery. According to Cornell University, this includes:

- Unique identifiers (such as an ISBN)
- Physical attributes (such as file dimensions or Pantone colors)
- Bibliographic attributes (such as the author or creator, title, and keywords)

Descriptive metadata can be the most robust of all the types of metadata, simply because there are many ways to describe an asset. When implementing a DAM solution, standardizing the specific attributes used to describe your assets and how they're documented is the key to streamlined discoverability.

What's the Point of All This Metadata?

By now, it should be clear that metadata is imperative for ensuring that your organization's vast collection of digital assets is easy to navigate and that your team members can quickly

find the assets they're looking for. Without metadata, your valuable digital assets could get buried in the sea of a complex, disorganized folder hierarchy somewhere, on someone's computer (but you're not sure whose). In short, it makes your digital assets searchable.

Metadata is what allows you to determine that you're working with the most recent version of a digital asset, and it can save you from legal hassles by preventing unauthorized users from accessing rights-restricted assets and using them for purposes not included in the usage license.

A data source is the location where data that is being used originates from.

A data source may be the initial location where data is born or where physical information is first digitized, however even the most refined data may serve as a source, as long as another process accesses and utilizes it. Concretely, a data source may be a database, a flat file, live measurements from physical devices, scraped web data, or any of the myriad static and streaming data services which abound across the internet.

III. Web Scraping

What is web scraping?

Web scraping is the process of collecting structured web data in an automated fashion. It's also called web data extraction. Some of the main use cases of web scraping include price monitoring, price intelligence, news monitoring, lead generation, and market research among many others.

In general, web data extraction is used by people and businesses who want to make use of the vast amount of publicly available web data to make smarter decisions. If you've ever copy and pasted information from a website, you've performed the same function as any web scraper, only on a microscopic, manual scale. Unlike the mundane, mind-numbing process of manually extracting data, web scraping uses intelligent automation to retrieve hundreds, millions, or even billions of data points from the internet's seemingly endless frontier.

The basics of web scraping

It's extremely simple, in truth, and works by way of two parts: a web crawler and a web scraper. The web crawler is the horse, and the scraper is the chariot. The crawler leads the scraper, as if by hand, through the internet, where it extracts the data requested. Learn the difference between web crawling & web scraping and how they work.

The crawler

A web crawler, which we generally call a “spider,” is an artificial intelligence that browses the internet to index and searches for content by following links and exploring, like a person with too much time on their hands. In many projects, you first “crawl” the web or one specific website to discover URLs which then you pass on to your scraper.

The scraper

A web scraper is a specialized tool designed to accurately and quickly extract data from a web page. Web scrapers vary widely in design and complexity, depending on the project. An important part of every scraper is the data locators (or selectors) that are used to find the data that you want to extract from the HTML file - usually, XPath, CSS selectors, regex, or a combination of them is applied.

The web scraping process

If you do it yourself

This is what a general DIY web scraping process looks like:

1. Identify the target website
2. Collect URLs of the pages where you want to extract data from
3. Make a request to these URLs to get the HTML of the page
4. Use locators to find the data in the HTML
5. Save the data in a JSON or CSV file or some other structured format

Simple enough, right? It is! If you just have a small project. But unfortunately, there are quite a few challenges you need to tackle if you need data at scale. For example, maintaining the scraper if the website layout changes, managing proxies, executing javascript, or working around antibots. These are all deeply technical problems that can eat up a lot of resources. That's part of the reason many businesses choose to outsource their web data projects.

If you outsource it

1. Our team gathers your requirements regarding your project.
2. Our veteran team of web scraping experts write the scraper(s) and set up the infrastructure to collect your data and structure it based on your requirements.
3. Finally, we deliver the data in your desired format and desired frequency.

Ultimately, the flexibility and scalability of web scraping ensure your project parameters, no matter how specific, can be met with ease. Fashion retailers inform their designers with upcoming trends based on web scraped insights, investors time their stock positions, and marketing teams overwhelm the competition with deep insights, all thanks to the burgeoning adoption of web scraping as an intrinsic part of everyday business.

What is web scraping used for?

Price intelligence

In our experience, price intelligence is the biggest use case for web scraping. Extracting product and pricing information from e-commerce websites, then turning it into intelligence is an important part of modern e-commerce companies that want to make better pricing/marketing decisions based on data.

How web pricing data and price intelligence can be useful:

- Dynamic pricing
- Revenue optimization
- Competitor monitoring

- Product trend monitoring
- Brand and MAP compliance

Market research

Market research is critical – and should be driven by the most accurate information available. High quality, high volume, and highly insightful web scraped data of every shape and size is fueling market analysis and business intelligence across the globe.

- Market trend analysis
- Market pricing
- Optimizing point of entry
- Research & development
- Competitor monitoring

Lead generation is a crucial marketing/sales activity for all businesses. In the 2020 Hubspot report, 61% of inbound marketers said generating traffic and leads was their number 1 challenge. Fortunately, web data extraction can be used to get access to structured lead lists from the web.

Brand monitoring

In today's highly competitive market, it's a top priority to protect your online reputation. Whether you sell your products online and have a strict pricing policy that you need to enforce or just want to know how people perceive your products online, brand monitoring with web scraping can give you this kind of information.

Business automation

In some situations, it can be cumbersome to get access to your data. Maybe you have some data on your own website or on your partner's website that you need in a structured way. But there's no easy internal way to do it and it makes sense to create a scraper and simply grab that data. As opposed to trying to work your way through complicated internal systems.

IV. Data mining and Data Warehousing

What is Data Mining?

Data Mining is defined as extracting information from huge sets of data. In other words, we can say that data mining is the procedure of mining knowledge from data. The information or knowledge extracted so can be used for any of the following applications –

- Market Analysis
- Fraud Detection
- Customer Retention
- Production Control
- Science Exploration

Data Mining Applications

Data mining is highly useful in the following domains –

- Market Analysis and Management
- Corporate Analysis & Risk Management
- Fraud Detection

Apart from these, data mining can also be used in the areas of production control, customer retention, science exploration, sports, astrology, and Internet Web Surf-Aid

Market Analysis and Management

Listed below are the various fields of market where data mining is used –

- Customer Profiling – Data mining helps determine what kind of people buy what kind of products.
- Identifying Customer Requirements – Data mining helps in identifying the best products for different customers. It uses prediction to find the factors that may attract new customers.
- Cross Market Analysis – Data mining performs Association/correlations between product sales.

- Target Marketing – Data mining helps to find clusters of model customers who share the same characteristics such as interests, spending habits, income, etc.
- Determining Customer purchasing pattern – Data mining helps in determining customer purchasing pattern.
- Providing Summary Information – Data mining provides us various multidimensional summary reports.

Corporate Analysis and Risk Management

Data mining is used in the following fields of the Corporate Sector –

- Finance Planning and Asset Evaluation – It involves cash flow analysis and prediction, contingent claim analysis to evaluate assets.
- Resource Planning – It involves summarizing and comparing the resources and spending.
- Competition – It involves monitoring competitors and market directions.

Fraud Detection

Data mining is also used in the fields of credit card services and telecommunication to detect frauds. In fraud telephone calls, it helps to find the destination of the call, duration of the call, time of the day or week, etc. It also analyzes the patterns that deviate from expected norms.

Data mining deals with the kind of patterns that can be mined. On the basis of the kind of data to be mined, there are two categories of functions involved in Data Mining –

- Descriptive
- Classification and Prediction

Descriptive Function

The descriptive function deals with the general properties of data in the database. Here is the list of descriptive functions –

- Class/Concept Description
- Mining of Frequent Patterns

- Mining of Associations
- Mining of Correlations
- Mining of Clusters

Class/Concept Description

Class/Concept refers to the data to be associated with the classes or concepts. For example, in a company, the classes of items for sales include computer and printers, and concepts of customers include big spenders and budget spenders. Such descriptions of a class or a concept are called class/concept descriptions. These descriptions can be derived by the following two ways –

- Data Characterization – This refers to summarizing data of class under study. This class under study is called as Target Class.
- Data Discrimination – It refers to the mapping or classification of a class with some predefined group or class.

Mining of Frequent Patterns

Frequent patterns are those patterns that occur frequently in transactional data. Here is the list of kind of frequent patterns –

- Frequent Item Set – It refers to a set of items that frequently appear together, for example, milk and bread.
- Frequent Subsequence – A sequence of patterns that occur frequently such as purchasing a camera is followed by memory card.
- Frequent Sub Structure – Substructure refers to different structural forms, such as graphs, trees, or lattices, which may be combined with item-sets or subsequences.

Mining of Association

Associations are used in retail sales to identify patterns that are frequently purchased together. This process refers to the process of uncovering the relationship among data and determining association rules.

For example, a retailer generates an association rule that shows that 70% of time milk is sold with bread and only 30% of times biscuits are sold with bread.

Mining of Correlations

It is a kind of additional analysis performed to uncover interesting statistical correlations between associated-attribute-value pairs or between two item sets to analyze that if they have positive, negative or no effect on each other.

Mining of Clusters

Cluster refers to a group of similar kind of objects. Cluster analysis refers to forming group of objects that are very similar to each other but are highly different from the objects in other clusters.

Classification and Prediction

Classification is the process of finding a model that describes the data classes or concepts. The purpose is to be able to use this model to predict the class of objects whose class label is unknown. This derived model is based on the analysis of sets of training data. The derived model can be presented in the following forms –

- Classification (IF-THEN) Rules
- Decision Trees
- Mathematical Formulae
- Neural Networks

The list of functions involved in these processes are as follows –

- Classification – It predicts the class of objects whose class label is unknown. Its objective is to find a derived model that describes and distinguishes data classes or concepts. The Derived Model is based on the analysis set of training data i.e. the data object whose class label is well known.
- Prediction – It is used to predict missing or unavailable numerical data values rather than class labels. Regression Analysis is generally used for prediction. Prediction can also be used for identification of distribution trends based on available data.
- Outlier Analysis – Outliers may be defined as the data objects that do not comply with the general behavior or model of the data available.

- Evolution Analysis – Evolution analysis refers to the description and model regularities or trends for objects whose behavior changes over time.

Data Mining Task Primitives

- We can specify a data mining task in the form of a data mining query.
- This query is input to the system.
- A data mining query is defined in terms of data mining task primitives.

Note – These primitives allow us to communicate in an interactive manner with the data mining system. Here is the list of Data Mining Task Primitives –

- Set of task relevant data to be mined.
- Kind of knowledge to be mined.
- Background knowledge to be used in discovery process.
- Interestingness measures and thresholds for pattern evaluation.
- Representation for visualizing the discovered patterns.

Set of task relevant data to be mined

This is the portion of database in which the user is interested. This portion includes the following –

- Database Attributes
- Data Warehouse dimensions of interest

Kind of knowledge to be mined

It refers to the kind of functions to be performed. These functions are –

- Characterization
- Discrimination
- Association and Correlation Analysis
- Classification
- Prediction

- Clustering
- Outlier Analysis
- Evolution Analysis

Background knowledge

The background knowledge allows data to be mined at multiple levels of abstraction. For example, the Concept hierarchies are one of the background knowledge that allows data to be mined at multiple levels of abstraction.

Interestingness measures and thresholds for pattern evaluation

This is used to evaluate the patterns that are discovered by the process of knowledge discovery. There are different interesting measures for different kind of knowledge.

Representation for visualizing the discovered patterns

This refers to the form in which discovered patterns are to be displayed. These representations may include the following. –

- Rules
- Tables
- Charts
- Graphs
- Decision Trees
- Cubes

Data Warehouse

A data warehouse exhibits the following characteristics to support the management's decision-making process –

- Subject Oriented – Data warehouse is subject oriented because it provides us the information around a subject rather than the organization's ongoing operations. These subjects can be product, customers, suppliers, sales, revenue, etc. The data warehouse does

not focus on the ongoing operations, rather it focuses on modelling and analysis of data for decision-making.

- Integrated – Data warehouse is constructed by integration of data from heterogeneous sources such as relational databases, flat files etc. This integration enhances the effective analysis of data.
- Time Variant – The data collected in a data warehouse is identified with a particular time period. The data in a data warehouse provides information from a historical point of view.
- Non-volatile – Nonvolatile means the previous data is not removed when new data is added to it. The data warehouse is kept separate from the operational database therefore frequent changes in operational database is not reflected in the data warehouse.

Data Warehousing

Data warehousing is the process of constructing and using the data warehouse. A data warehouse is constructed by integrating the data from multiple heterogeneous sources. It supports analytical reporting, structured and/or ad hoc queries, and decision making.

Data warehousing involves data cleaning, data integration, and data consolidations. To integrate heterogeneous databases, we have the following two approaches –

- Query Driven Approach
- Update Driven Approach

Query-Driven Approach

This is the traditional approach to integrate heterogeneous databases. This approach is used to build wrappers and integrators on top of multiple heterogeneous databases. These integrators are also known as mediators.

Process of Query Driven Approach

- When a query is issued to a client side, a metadata dictionary translates the query into the queries, appropriate for the individual heterogeneous site involved.
- Now these queries are mapped and sent to the local query processor.

- The results from heterogeneous sites are integrated into a global answer set.

Disadvantages

This approach has the following disadvantages –

- The Query Driven Approach needs complex integration and filtering processes.
- It is very inefficient and very expensive for frequent queries.
- This approach is expensive for queries that require aggregations.

Update-Driven Approach

Today's data warehouse systems follow update-driven approach rather than the traditional approach discussed earlier. In the update-driven approach, the information from multiple heterogeneous sources is integrated in advance and stored in a warehouse. This information is available for direct querying and analysis.

Advantages

This approach has the following advantages –

- This approach provides high performance.
- The data can be copied, processed, integrated, annotated, summarized and restructured in the semantic data store in advance.

Query processing does not require interface with the processing at local sources.

From Data Warehousing (OLAP) to Data Mining (OLAM)

Online Analytical Mining integrates with Online Analytical Processing with data mining and mining knowledge in multidimensional databases. Here is the diagram that shows the integration of both OLAP and OLAM –

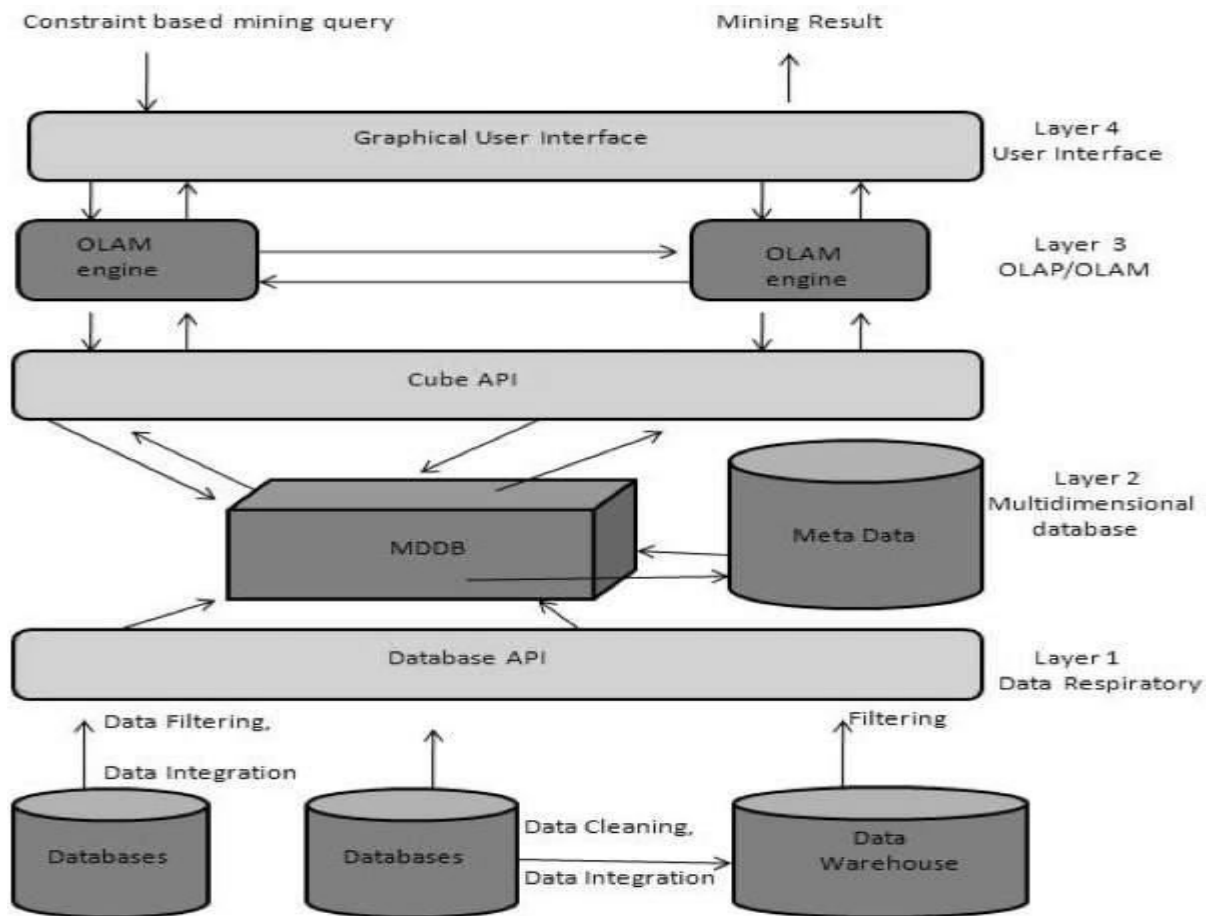


Fig 1.7 Datawarehousing to Data Mining

Importance of OLAM

OLAM is important for the following reasons –

- High quality of data in data warehouses – The data mining tools are required to work on integrated, consistent, and cleaned data. These steps are very costly in the preprocessing of data. The data warehouses constructed by such preprocessing are valuable sources of high quality data for OLAP and data mining as well.
- Available information processing infrastructure surrounding data warehouses – Information processing infrastructure refers to accessing, integration, consolidation, and transformation of multiple heterogeneous databases, web-accessing and service facilities, reporting and OLAP analysis tools.

- OLAP–based exploratory data analysis – Exploratory data analysis is required for effective data mining. OLAM provides facility for data mining on various subset of data and at different levels of abstraction.
- Online selection of data mining functions – Integrating OLAP with multiple data mining functions and online analytical mining provide users with the flexibility to select desired data mining functions and swap data mining tasks dynamically.

VI. Distributed File System

What is a Distributed File System?

A distributed file system (DFS) differs from typical file systems (i.e., NTFS and HFS) in that it allows direct host access to the same file data from multiple locations. Indeed, the data behind a DFS can reside in a different location from all of the hosts that access it.

Features of Distributed File System

- Transparent local access – From a host perspective, the data is accessed as if it's local to the host that is accessing it.
- Location independence – Hosts may have no idea where file data physically resides. Data location is managed by the DFS and not by the host accessing it.
- Coherent access – DFS file data is managed so that it appears to the host(s) as if it's all within a single file system, even though its data could be distributed across many storage devices/servers and locations.
- Great large-file streaming throughput – DFS systems emerged to supply high-streaming performance for HPC workloads, and most continue to do so.
- File locking – DFSs usually support file locking across or within locations, which ensures that no two hosts can modify the same file at the same time.
- Data-in-flight encryption – Most DFS systems support encrypting data and metadata while it is in transit.
- Diverse storage media/systems – Most DFS systems can make use of spinning disk, SAS SSDs, NVMe SSDs, and S3 object storage, as well as private, on-premises object storage to hold file data. While most DFS systems have very specific requirements for metadata servers, their data or file storage can often reside on just about any storage available, including the public cloud.

- Multi-protocol access – Hosts can access DFS data using standard NFS, SMB, or a POSIX client supplied by the solution provider. Occasionally one can also see NVMe-oF for files and (NVIDIA) GPU Direct Storage access. This can also mean that the same file can be accessed with all protocols the DFS solution supports.
- Multi-networking access – While all DFSs provide Ethernet access to file system data, some also provide InfiniBand and other high performance networking access.
- Local gateways – DFS systems may require some server and storage resources at each location that has access to its file data. Local gateways often cache metadata and data referenced by hosts. Gateways like this can typically be scaled up or down to sustain performance requirements. In some cases where access and data reside together, gateways are not needed.
- Software-defined solutions – Given all the above, most DFS systems are software defined solutions. Some DFS systems are also available in appliance solutions, but that is more for purchasing/deployment convenience than a requirement of the DFS solution.
- Scale-out storage solution – Most DFS systems support scale-out file systems in which file data and metadata service performance and capacity can be increased by adding more metadata or file data server resources—which includes gateways.

Characteristics of a Modern Distributed File System

- High IOPS/great small-file performance – Some DFS systems support very high IOPS for improved small file performance.
- Cross-protocol locking – Some DFS systems allow for one protocol to lock a file while being modified by another protocol. This feature prohibits a file from being corrupted by multi-host access even when accessing the file with different protocols.
- Cloud resident services – Some DFS solutions can run in a public cloud environment. That is, their file data storage, metadata services, and any monitoring/management services all run in a public cloud provider. File data access can then take place all within the same cloud AZ or across cloud regions or even on premises with access to that cloud data.
- High availability support – Some DFS systems also support very high availability by splitting and replicating their control, metadata, and file data storage systems across multiple sites, AZs or servers.
- (File) data reduction – Some DFS solutions support data compression or deduplication designed to reduce the physical data storage space required to store file data.

- Data-at-rest encryption – Some DFS systems offer encryption of file data and metadata at rest.
Single name space – Some DFS systems provide the ability to stitch multiple file systems/shares into a single name space, which can be used to access any file directory being served.
- Geo-fencing – Some DFS systems can limit or restrict the physical locations in which data can reside and from which it can be accessed. This capability can be required to support GDPR and other legal restrictions on data movement.

Advantages of Distributed File System

There are several reasons why one might consider a DFS solution for their environment, but they all boil down to a need to access the same data from multiple locations. This need could be due to requiring support for multiple sites processing the same data. An example of a team with such a need would be a multi-site engineering team that has compute resources local to their environment and that has a group of engineers who all participate in different phases of engineering a system while using the same data. It could also be due to cloud bursting, where workloads move from on-premises to in-cloud or even to other AZs within a cloud to gain access to more processing power. Finally, another example might include any situation in which one wants to utilize a hybrid cloud solution that requires access to the same data. All these can benefit from the use of a DFS.

Yes, these capabilities could all be accomplished in some other way, such as copying/moving data from one site/cloud to another. But that takes planning, time, and manual, error-prone procedures to make it happen. A DFS can do all of this for you—automatically—without any of the pain.

Disadvantages of Distributed File System

If data resides on cloud storage and is consumed elsewhere, cloud egress charges may be significant. This cost can be mitigated somewhat by data reduction techniques and the fact that DFS will move only data that is being accessed, but that may still leave a significant egress expense.

While DFS systems can cache data to improve local access, none can defeat the speed of light. That is, the access to the first non-cached byte of data may take an inordinate amount of time depending on how far it is from where it is being accessed. DFS solutions can minimize this overhead by overlapping cached data access while fetching the next portion of data, but doing this will not always mask the network latency required to access a non-cached portion of the data.

DFS systems can sometimes be complex to deploy. With data that can reside just about anywhere with hosts that can access all of it from anywhere else, getting all of this to work together properly and tuning it for high performance can be a significant challenge. Vendor support can help considerably. DFS vendors will have professional services that can be used to help deploy and configure their systems to get them up and running in a timely fashion. Also, they will have sophisticated modeling that can tell them how much gateway, metadata, and storage resources will be required to support your performance.

Conclusion

It all boils down to this: DFS systems offer global access to the same data that is very difficult to accomplish effectively in any other way, especially when you have multiple sites, all computing with and consuming the same data. DFS systems can make all of this look easy and seamless, without breaking a sweat. So, if you have multiple sites with a need to access the same data, a DFS system can be a solution.

UNIT – II

Data Foundation – SCSA1309

UNIT 2 DATA PROCESS OVERVIEW

Defining Goals- Data Acquisition – Sources of acquiring the data - Data preprocessing- Imputation of Missing values - Data cleaning - Data Reduction, Data Transformation and Data Discretization. Exploratory Data Analysis (EDA) – Philosophy of EDA - The Data Science Process. Significance of EDA in data science - Basic tools (plots, graphs and summary statistics) of EDA.

V.

Introduction

What is Data Science?

Data Science is the area of study which involves extracting insights from vast amounts of data by the use of various scientific methods, algorithms, and processes. It helps you to discover hidden patterns from the raw data. The term Data Science has emerged because of the evolution of mathematical statistics, data analysis, and big data.

Data Science is an interdisciplinary field that allows you to extract knowledge from structured or unstructured data. Data science enables you to translate a business problem into a research project and then translate it back into a practical solution.

- Data is the oil for today's world. With the right tools, technologies, algorithms, we can use data and convert it into a distinctive business advantage
- Data Science can help you to detect fraud using advanced machine learning algorithms
- It helps you to prevent any significant monetary losses
- Allows to build intelligence ability in machines
- You can perform sentiment analysis to gauge customer brand loyalty
- It enables you to take better and faster decisions
- Helps you to recommend the right product to the right customer to enhance your business

Data Science Components

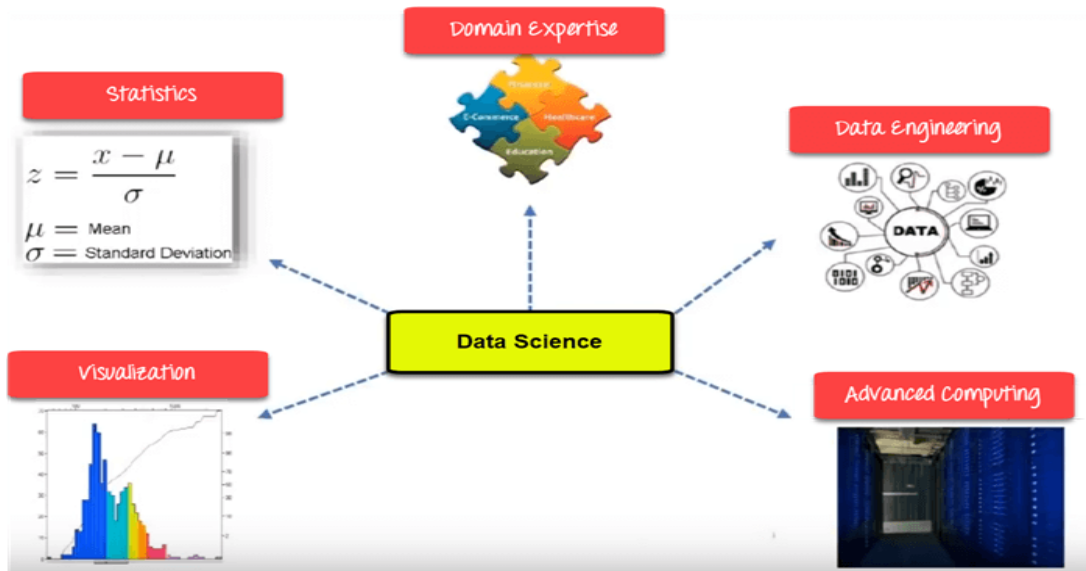


Fig. 2.1 Data Science Components

Statistics:

Statistics is the most critical unit of Data Science basics. It is the method or science of collecting and analyzing numerical data in large quantities to get useful insights.

Visualization:

Visualization technique helps you to access huge amounts of data in easy to understand and digestible visuals.

Machine Learning:

Machine Learning explores the building and study of algorithms which learn to make predictions about unforeseen/future data.

Deep Learning:

Deep Learning method is new machine learning research where the algorithm selects the analysis model to follow.

Data Science Process

Now in this Data Science Tutorial, we will learn the Data Science Process:



Fig. 2.2 Data Science Process

1. Discovery:

Discovery step involves acquiring data from all the identified internal & external sources which helps you to answer the business question.

The data can be:

- Logs from webserver
- Data gathered from social media
- Census datasets
- Data streamed from online sources using APIs

2. Preparation:

Data can have lots of inconsistencies like missing value, blank columns, incorrect data format which needs to be cleaned. You need to process, explore, and condition data before modeling. The cleaner your data, the better are your predictions.

3. Model Planning:

In this stage, you need to determine the method and technique to draw the relation between input variables. Planning for a model is performed by using different statistical formulas and visualization tools. SQL analysis services, R, and SAS/access are some of the tools used for this purpose.

4. Model Building:

In this step, the actual model building process starts. Here, Data scientist distributes datasets for training and testing. Techniques like association, classification, and clustering are applied to the training data set. The model once prepared is tested against the "testing" dataset.

5. Operationalize:

In this stage, you deliver the final baselined model with reports, code, and technical documents. Model is deployed into a real-time production environment after thorough testing.

6. Communicate Results

In this stage, the key findings are communicated to all stakeholders. This helps you to decide if the results of the project are a success or a failure based on the inputs from the model.

Data Science Jobs Roles

Most prominent Data Scientist job titles are:

- Data Scientist
- Data Engineer
- Data Analyst
- Statistician

- Data Architect
- Data Admin
- Business Analyst
- Data/Analytics Manager

Now in this Data Science Tutorial, let's learn what each role entails in detail:

Data Scientist:

Role:

A Data Scientist is a professional who manages enormous amounts of data to come up with compelling business visions by using various tools, techniques, methodologies, algorithms, etc.

Languages:

R, SAS, Python, SQL, Hive, Matlab, Pig, Spark

Data Engineer:

Role:

The role of data engineer is of working with large amounts of data. He develops, constructs, tests, and maintains architectures like large scale processing system and databases.

Languages:

SQL, Hive, R, SAS, Matlab, Python, Java, Ruby, C + +, and Perl

Data Analyst:

Role:

A data analyst is responsible for mining vast amounts of data. He or she will look for relationships, patterns, trends in data. Later he or she will deliver compelling reporting and visualization for analyzing the data to take the most viable business decisions.

Languages:

R, Python, HTML, JS, C, C++ , SQL

Statistician:

Role:

The statistician collects, analyses, understand qualitative and quantitative data by using statistical theories and methods.

Languages:

SQL, R, Matlab, Tableau, Python, Perl, Spark, and Hive

Data Administrator:

Role:

Data admin should ensure that the database is accessible to all relevant users. He also makes sure that it is performing correctly and is being kept safe from hacking.

Languages:

Ruby on Rails, SQL, Java, C#, and Python

Business Analyst:

Role:

This professional need to improves business processes. He/she as an intermediary between the business executive team and IT department.

Languages:

SQL, Tableau, Power BI and, Python



Fig. 2.3 Tools for Data Science

Data Analysis	Data warehousing	Data Visualization	Machine Learning
<u>R</u> , Spark, <u>Python</u> and <u>SAS</u>	<u>Hadoop</u> , SQL, <u>Hive</u>	R, <u>Tableau</u> , Raw	<u>Spark</u> , Azure ML studio, Mahout

Difference Between Data Science with BI (Business Intelligence)

Parameters	Business Intelligence	Data Science
Perception	Looking Backward	Looking Forward
Data Sources	Structured Data. Mostly SQL, but some time Data Warehouse)	Structured and Unstructured data. Like logs, SQL, NoSQL, or text
Approach	Statistics & Visualization	Statistics, Machine Learning, and Graph
Emphasis	Past & Present	Analysis & Neuro-linguistic Programming
Tools	Pentaho. Microsoft BI, QlikView,	R, <u>TensorFlow</u>

Applications of Data Science

Now in this Data Science Tutorial, we will learn about Applications of Data Science:

Internet Search:

Google search use Data science technology to search a specific result within a fraction of a second

Recommendation Systems:

To create a recommendation system. Example, "suggested friends" on Facebook or suggested videos" on YouTube, everything is done with the help of Data Science.

Image & Speech Recognition:

Speech recognizes system like Siri, Google assistant, Alexa runs on the technique of Data science. Moreover, Facebook recognizes your friend when you upload a photo with them, with the help of Data Science.

Gaming world:

EA Sports, Sony, Nintendo, are using Data science technology. This enhances your gaming experience. Games are now developed using Machine Learning technique. It can update itself when you move to higher levels.

Online Price Comparison:

PriceRunner, Junglee, Shopzilla work on the Data science mechanism. Here, data is fetched from the relevant websites using APIs.

Challenges of Data science Technology

- High variety of information & data is required for accurate analysis
- Not adequate data science talent pool available
- Management does not provide financial support for a data science team
- Unavailability of/difficult access to data

- Data Science results not effectively used by business decision makers
- Explaining data science to others is difficult
- Privacy issues
- Lack of significant domain expert
- If an organization is very small, they can't have a Data Science team

Summary

- Data Science is the area of study which involves extracting insights from vast amounts of data by the use of various scientific methods, algorithms, and processes.
- Statistics, Visualization, Deep Learning, Machine Learning, are important Data Science concepts.
- Data Science Process goes through Discovery, Data Preparation, Model Planning, Model Building, Operationalize, Communicate Results.
- Important Data Scientist job roles are: 1) Data Scientist 2) Data Engineer 3) Data Analyst 4) Statistician 5) Data Architect 6) Data Admin 7) Business Analyst 8) Data/Analytics Manager
- R, SQL, Python, SaS, are essential Data science tools
- The predictions of Business Intelligence is looking backward while for Data Science it is looking forward.
- Important applications of Data science are 1) Internet Search 2) Recommendation Systems 3) Image & Speech Recognition 4) Gaming world 5) Online Price Comparison.
- High variety of information & data is the biggest challenge of Data Science technology.

VI. DATA ACQUISITION

Data acquisition has been understood as the process of gathering, filtering, and cleaning data before the data is put in a data warehouse or any other storage solution.

Data acquisition and understanding stage of the Team Data Science Process

Here are the goals, tasks, and deliverables associated with the data acquisition and understanding stage of the Team Data Science Process (TDSP). This process provides a recommended lifecycle

that you can use to structure your data-science projects. The lifecycle outlines the major stages that projects typically execute, often iteratively:

1. Business understanding
2. Data acquisition and understanding
3. Modeling
4. Deployment
5. Customer acceptance

Here is a visual representation of the TDSP lifecycle:

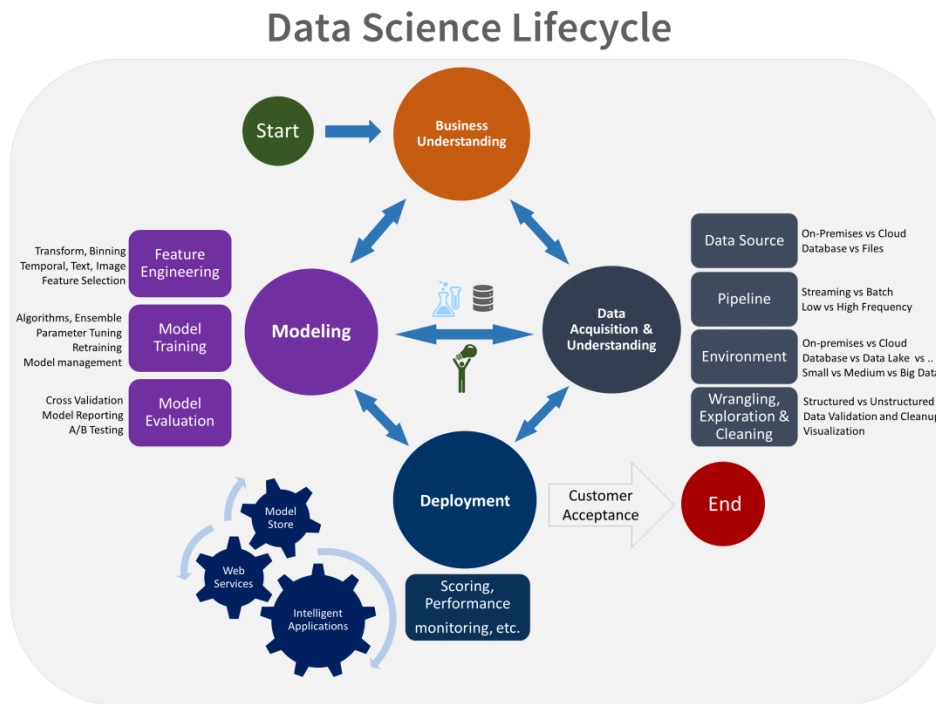


Fig. 2.4 Data Science Life Cycle

Goals

- Produce a clean, high-quality data set whose relationship to the target variables is understood. Locate the data set in the appropriate analytics environment so you are ready to model.
- Develop a solution architecture of the data pipeline that refreshes and scores the data regularly.

How to do it

There are three main tasks addressed in this stage:

- Ingest the data into the target analytic environment.
- Explore the data to determine if the data quality is adequate to answer the question.
- Set up a data pipeline to score new or regularly refreshed data.

Ingest the data

Set up the process to move the data from the source locations to the target locations where you run analytics operations, like training and predictions. For technical details and options on how to move the data with various Azure data services, see [Load data into storage environments for analytics](#).

Explore the data

Before you train your models, you need to develop a sound understanding of the data. Real-world data sets are often noisy, are missing values, or have a host of other discrepancies. You can use data summarization and visualization to audit the quality of your data and provide the information you need to process the data before it's ready for modeling. This process is often iterative. For guidance on cleaning the data, see [Tasks to prepare data for enhanced machine learning](#).

After you're satisfied with the quality of the cleansed data, the next step is to better understand the patterns that are inherent in the data. This data analysis helps you choose and develop an appropriate predictive model for your target. Look for evidence for how well connected the data is to the target. Then determine whether there is sufficient data to move forward with the next modeling steps. Again, this process is often iterative. You might need to find new data sources with more accurate or more relevant data to augment the data set initially identified in the previous stage.

Set up a data pipeline

In addition to the initial ingestion and cleaning of the data, you typically need to set up a process to score new data or refresh the data regularly as part of an ongoing learning process. Scoring may be completed with a data pipeline or workflow. The [Move data from a SQL Server instance to Azure SQL Database with Azure Data Factory](#) article gives an example of how to set up a pipeline with Azure Data Factory.

In this stage, you develop a solution architecture of the data pipeline. You develop the pipeline in parallel with the next stage of the data science project. Depending on your business needs and the constraints of your existing systems into which this solution is being integrated, the pipeline can be one of the following options:

- Batch-based
- Streaming or real time
- A hybrid

Artifacts

The following are the deliverables in this stage:

- Data quality report: This report includes data summaries, the relationships between each attribute and target, variable ranking, and more.
- Solution architecture: The solution architecture can be a diagram or description of your data pipeline that you use to run scoring or predictions on new data after you have built a model. It also contains the pipeline to retrain your model based on new data. Store the document in the Project directory when you use the TDSP directory structure template.
- Checkpoint decision: Before you begin full-feature engineering and model building, you can reevaluate the project to determine whether the value expected is sufficient to continue pursuing it. You might, for example, be ready to proceed, need to collect more data, or abandon the project as the data does not exist to answer the question.

VII. Data Preprocessing

Data is truly considered a resource in today's world. As per the World Economic Forum, by 2025 we will be generating about 463 exabytes of data globally per day! But is all this data fit enough to be used by machine learning algorithms? How do we decide that? In this article we will explore the topic of data preprocessing — transforming the data such that it becomes machine-readable...

The aim of this article is to introduce the concepts that are used in data preprocessing, a major step in the Machine Learning Process.

What is Data Preprocessing?

When we talk about data, we usually think of some large datasets with huge number of rows and columns. While that is a likely scenario, it is not always the case — data could be in so many different forms: Structured Tables, Images, Audio files, Videos etc..

Machines don't understand free text, image or video data as it is, they understand 1s and 0s. So it probably won't be good enough if we put on a slideshow of all our images and expect our machine learning model to get trained just by that!

In any Machine Learning process, Data Preprocessing is that step in which the data gets transformed, or *Encoded*, to bring it to such a state that now the machine can easily parse it. In other words, the *features* of the data can now be easily interpreted by the algorithm.

Features in Machine Learning

A dataset can be viewed as a collection of *data objects*, which are often also called as a records, points, vectors, patterns, events, cases, samples, observations, or entities. Data objects are described by a number of *features*, that capture the basic characteristics of an object, such as the mass of a physical object or the time at which an event occurred, etc.. Features are often called as variables, characteristics, fields, attributes, or dimensions.

A feature is an individual measurable property or characteristic of a phenomenon being observed

For instance, color, mileage and power can be considered as features of a car. There are different types of features that we can come across when we deal with data.

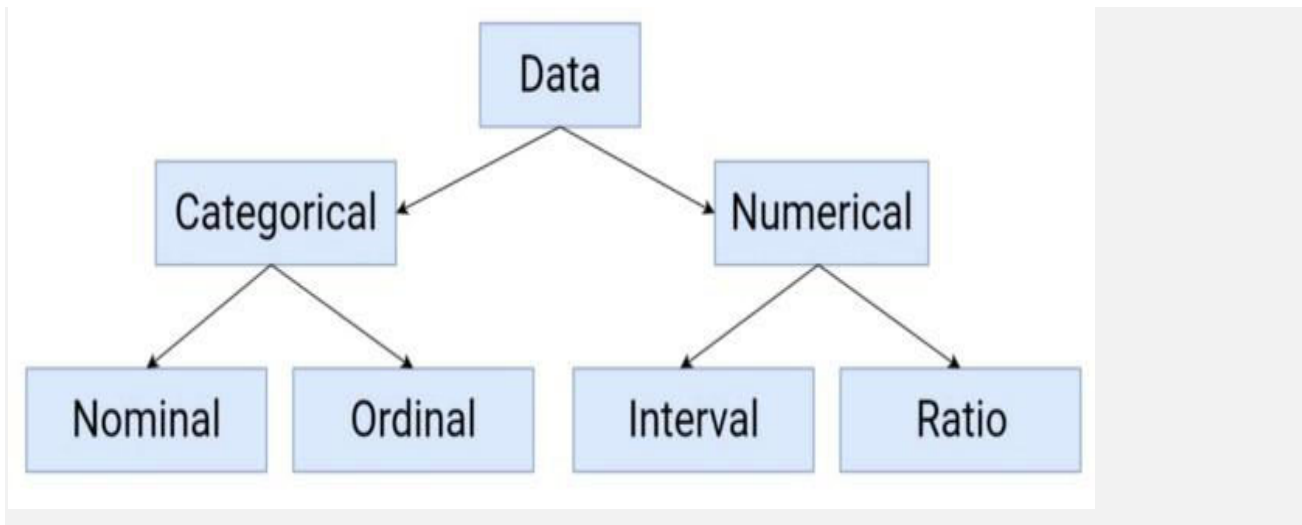


Fig 2.5 Statistical Data Types

Features can be:

- **Categorical** : Features whose values are taken from a defined set of values. For instance, days in a week : {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday} is a category because its value is always taken from this set. Another example could be the Boolean set : {True, False}
- **Numerical** : Features whose values are continuous or integer-valued. They are represented by numbers and possess most of the properties of numbers. For instance, number of steps you walk in a day, or the speed at which you are driving your car at.

Feature Types

Now that we have gone over the basics, let us begin with the steps of Data Preprocessing. Remember, not all the steps are applicable for each problem, it is highly dependent on the data we are working with, so maybe only a few steps might be required with your dataset. Generally they are :

- Data Quality Assessment
- Feature Aggregation
- Feature Sampling

- Dimensionality Reduction
- Feature Encoding

Data Quality Assessment

Because data is often taken from multiple sources which are normally not too reliable and that too in different formats, more than half our time is consumed in dealing with data quality issues when working on a machine learning problem. It is simply unrealistic to expect that the data will be perfect. There may be problems due to human error, limitations of measuring devices, or flaws in the data collection process. Let's go over a few of them and methods to deal with them :

1. Missing values :

It is very much usual to have missing values in your dataset. It may have happened during data collection, or maybe due to some data validation rule, but regardless missing values must be taken into consideration.

- Eliminate rows with missing data :
Simple and sometimes effective strategy. Fails if many objects have missing values. If a feature has mostly missing values, then that feature itself can also be eliminated.
- Estimate missing values :
If only a reasonable percentage of values are missing, then we can also run simple interpolation methods to fill in those values. However, most common method of dealing with missing values is by filling them in with the mean, median or mode value of the respective feature.

2. Inconsistent values :

We know that data can contain inconsistent values. Most probably we have already faced this issue at some point. For instance, the 'Address' field contains the 'Phone number'. It may be due to human error or maybe the information was misread while being scanned from a handwritten form.

- It is therefore always advised to perform data assessment like knowing what the data type of the features should be and whether it is the same for all the data objects.

3. Duplicate values :

A dataset may include data objects which are duplicates of one another. It may happen when say the same person submits a form more than once. The term deduplication is often used to refer to the process of dealing with duplicates.

- In most cases, the duplicates are removed so as to not give that particular data object an advantage or *bias*, when running machine learning algorithms.

Feature Aggregation

Feature Aggregations are performed so as to take the aggregated values in order to put the data in a better perspective. Think of transactional data, suppose we have day-to-day transactions of a product from recording the daily sales of that product in various store locations over the year. Aggregating the transactions to single store-wide monthly or yearly transactions will help us reducing hundreds or potentially thousands of transactions that occur daily at a specific store, thereby reducing the number of data objects.

- This results in reduction of memory consumption and processing time
- Aggregations provide us with a high-level view of the data as the behaviour of groups or aggregates is more stable than individual data objects

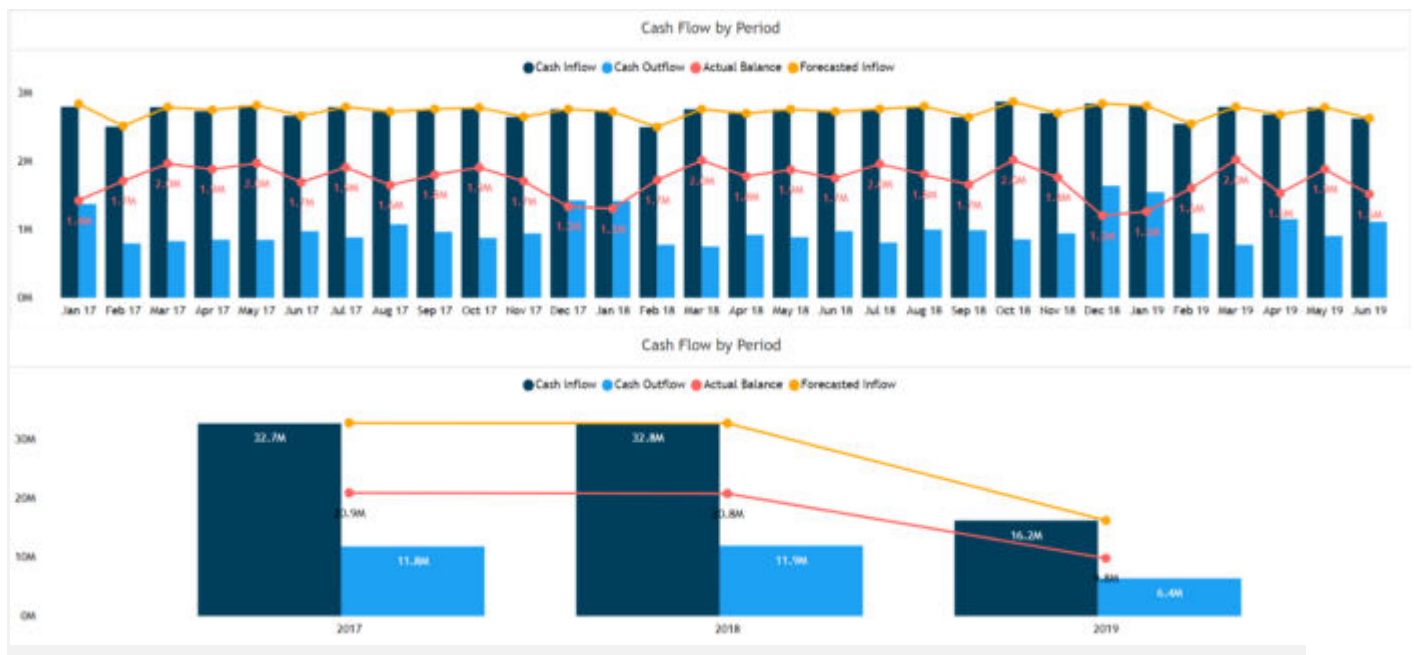


Fig. 2.6 Aggregation from Monthly to Yearly

Feature Sampling

Sampling is a very common method for selecting a subset of the dataset that we are analyzing. In most cases, working with the complete dataset can turn out to be too expensive considering the memory and time constraints. Using a sampling algorithm can help us reduce the size of the dataset to a point where we can use a better, but more *expensive*, machine learning algorithm.

The key principle here is that the sampling should be done in such a manner that the sample generated should have approximately the same properties as the original dataset, meaning that the sample is *representative*. This involves choosing the correct sample size and sampling strategy.

Simple Random Sampling dictates that there is an equal probability of selecting any particular entity. It has two main variations as well :

- Sampling without Replacement : As each item is selected, it is removed from the set of all the objects that form the total dataset.

- Sampling with Replacement : Items are not removed from the total dataset after getting selected. This means they can get selected more than once.



Fig. 2.7 Data Sampling

Although Simple Random Sampling provides two great sampling techniques, it can fail to output a representative sample when the dataset includes object types which vary drastically in ratio. This can cause problems when the sample needs to have a proper representation of all object types, for example, when we have an *imbalanced* dataset.

An Imbalanced dataset is one where the number of instances of a class(es) are significantly higher than another class(es), thus leading to an imbalance and creating rarer class(es). It is critical that the rarer classes be adequately represented in the sample. In these cases, there is another sampling technique which we can use, called *Stratified Sampling*, which begins with predefined groups of objects. There are different versions of Stratified Sampling too, with the simplest version suggesting equal number of objects be drawn from all the groups even though the groups are of different sizes. .

Dimensionality Reduction

Most real world datasets have a large number of features. For example, consider an image processing problem, we might have to deal with thousands of features, also called as *dimensions*. As the name suggests, dimensionality reduction aims to reduce the number of features - but not simply by selecting a sample of features from the *feature-set*, which is something else — Feature Subset Selection or simply Feature Selection.

Conceptually, *dimension* refers to the number of geometric planes the dataset lies in, which could be high so much so that it cannot be visualized with pen and paper. More the number of such planes, more is the complexity of the dataset.

The Curse of Dimensionality

This refers to the phenomena that generally data analysis tasks become significantly harder as the dimensionality of the data increases. As the dimensionality increases, the number planes occupied by the data increases thus adding more and more sparsity to the data which is difficult to model and visualize.

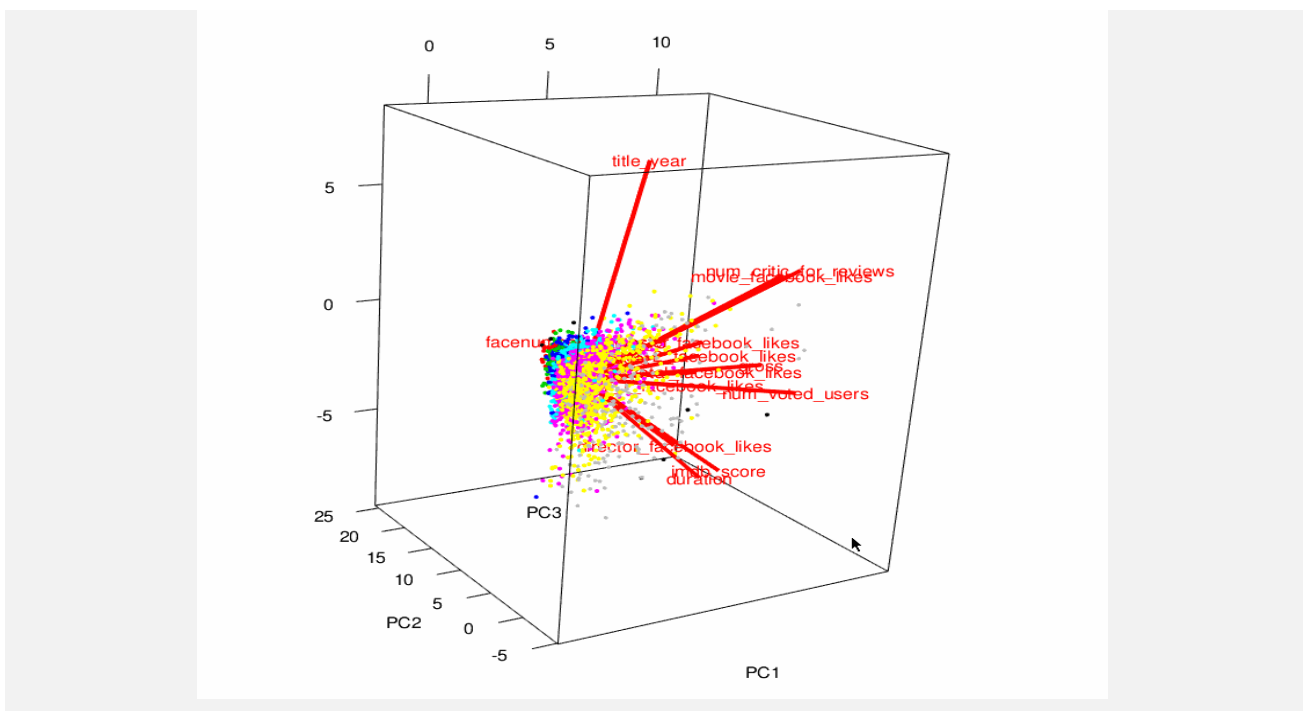


Fig. 2.8 Representation of components in different spaces

What dimension reduction essentially does is that it maps the dataset to a lower-dimensional space, which may very well be to a number of planes which can now be visualized, say 2D. The basic objective of techniques which are used for this purpose is to reduce the dimensionality of a dataset by creating new features which are a combination of the old features. In other words, the higher-dimensional feature-space is mapped to a lower-dimensional feature-space. Principal Component Analysis and Singular Value Decomposition are two widely accepted techniques.

A few major benefits of dimensionality reduction are :

- Data Analysis algorithms work better if the dimensionality of the dataset is lower. This is mainly because irrelevant features and noise have now been eliminated.
- The models which are built on top of lower dimensional data are more understandable and explainable.
- The data may now also get easier to visualize!
Features can always be taken in pairs or triplets for visualization purposes, which makes more sense if the feature set is not that big.

Feature Encoding

As mentioned before, the whole purpose of data preprocessing is to *encode* the data in order to bring it to such a state that the machine now understands it.

Feature encoding is basically performing transformations on the data such that it can be easily accepted as input for machine learning algorithms while still retaining its original meaning.

There are some general norms or rules which are followed when performing feature encoding. For Continuous variables :

- Nominal : Any one-to-one mapping can be done which retains the meaning. For instance, a permutation of values like in One-Hot Encoding.
- Ordinal : An order-preserving change of values. The notion of small, medium and large can be represented equally well with the help of a new function, that is, $\langle \text{new_value} = f(\text{old_value}) \rangle$
- For example, {0, 1, 2} or maybe {1, 2, 3}.

	Name	Generation	Gen 1	Gen 2	Gen 3	Gen 4	Gen 5
4	Octillery	Gen 2	0	1	0	0	0
5	Helioptile	Gen 6	0	0	0	0	0
6	Dialga	Gen 4	0	0	0	1	0
7	DeoxysDefense Forme	Gen 3	0	0	1	0	0
8	Rapidash	Gen 1	1	0	0	0	0
9	Swanna	Gen 5	0	0	0	0	1

Fig. 2.9 One-hot encoding of the data

For Numeric variables:

- Interval : Simple mathematical transformation like using the equation $\langle \text{new_value} = a * \text{old_value} + b \rangle$, a and b being constants. For example, Fahrenheit and Celsius scales, which differ in their Zero values size of a unit, can be encoded in this manner.
- Ratio : These variables can be scaled to any particular measures, of course while still maintaining the meaning and ratio of their values. Simple mathematical transformations work in this case as well, like the transformation $\langle \text{new_value} = a * \text{old_value} \rangle$. For, length can be measured in meters or feet, money can be taken in different currencies.

VIII. Exploratory Data Analysis

EDA:

There are several models that data can be fit into for a thorough analysis. But before you do so, you have to determine which model is an ideal fit for the data at hand. For this reason, you end up exploring the data, its shape, characteristics and come up with a summary that describes the current state of data at hand and whether the data needs further processing before it can be modeled with statistical and scientific techniques. This exploration of data, usually with the help of descriptive statistics, visualization tools, and presentation techniques, make up for what we call Exploratory Data Analysis or EDA in data science.

EDA in data science is quite like the service advisor doing a rough inspection of your car, asking a few preliminary questions, setting expectations and then taking the car in for service. It is one of the first things done with the data, so it is a critical phase, as many inferences and consequent actions depend on this exploration.

1. DEFINITION

The initial analysis of data supplied or extracted, to understand the trends, underlying limitations, quality, patterns, and relationships between various entities within the data set, using descriptive statistics and visualization tools is called Exploratory Data Analysis (EDA). EDA will give you a fair idea of what model better fits the data and whether any data cleansing and massaging might be required before taking the data through advanced modeling techniques or even put through Machine Learning and Artificial Intelligence algorithms.

2. WHAT IS EDA

EDA can be quite expensive and time-consuming depending on what and how much data you have. Unfortunately, there is no structured way to perform EDA, although there are a few techniques that will give you the best results out of EDA. Of the many outcomes of EDA, the important ones that one should try to get from the data are,

- Detect outliers and anomalies
- Determine the quality of data
- Determine what statistical models can fit the data
- Find out if the assumptions about the data, that you or your team started out with is correct or way off.
- Extract variables or dimensions on which the data can be pivoted.
- Determine whether to apply univariate or multivariate analytical techniques.

3. EXAMPLES OF EDA IN DATA SCIENCE

Ok, time for some examples, that might give you an idea about what EDA really entails and what are you looking for, what questions are you trying to answer.

Example 1: Missing data

With data comes a lot of anomalies. One of them is missing data. Although the overall data might be good, there are columns within the data set that might be missing values. This can skew your results and not provide an accurate model for further use.

One great way to identify missing values visually is the use of the missing package in Python. This is obviously for a large data set. This gives you a graphic story of how much data is missing and on which variables.

Without going into the details of the coding, the above graphic was achieved with a single line of code. The white lines in the above graph indicate missing values.

Example 2: Summary statistics

Another example here is of summary statistics that give you a fair idea of your numeric data.

As shown above, the columns are features of the data set, and the statistics on the left column describe each.

Example 3: Outliers

Outliers are data that lie on the extreme or even outside the spectrum of values that a variable should normally hold, thereby giving you a hint or an opportunity to explore.

You can immediately notice an outlier, where it shows a good percentage of customers are buying more than 50 products. An investigation can be initiated and, in many cases, they turn out to be resellers. This can be seen as an opportunity to develop a B2B relationship with the resellers and grow it as a separate vertical in the business.

4. TECHNIQUES OF EDA IN DATA SCIENCE

There are broadly two categories of EDA, graphical and non-graphical. These two are further divided into univariate and multivariate EDA, based on interdependency of variables in your data.

Univariate non-graphical: Here, the data features a single variable, and the EDA is done in mostly tabular form, for example, summary statistics. These non-graphical analyses give you a statistic that indicates how skewed your data might be or which is the dominant value for your variable if any.

Univariate graphical: The EDA here, involves graphic tools like bar charts and histograms to get a quick view of how variable properties are stacked against each other, whether there is a relationship between these properties and whether there is any interdependency among these properties.

Multivariate non-graphical: Non-graphical methods like crosstabs are used to depict the relationship between two or more variables. Statistical values like correlation coefficient indicate if there are a possible relationship and the measure of correlation.

Multivariate graphical: A graphical representation always gives you a better understanding of the relationship, especially among multiple variables.

5. *TOOLS*

The most commonly used software tools to perform EDA are Python and R. Both enjoy massive community support and frequent updates on packages that can be used to EDA. Let's look at the various graphical instruments that can be used to execute an EDA.

- **Box plots**

Box plots are used where there is a need to summarize data on an interval scale like the ones on the stock market, where ticks observed in one whole day may be represented in a single box, highlighting the lowest, highest, median and outliers.

- **Heatmap**

Heatmaps are most often used for the representation of the correlation between variables. Here is an example of a heatmap. As you can see from the chart, there is a strong correlation between density and residual sugar and absolutely no correlation between alcohol and residual sugar.

- **Histograms**

The histogram is the graphical representation of numerical data that splits the data into ranges. The taller the bar, the greater the number of data points falling in that range. A good example here is the height data of a class of students. You would notice that the height data looks like a bell curves for a particular class with most the data lying within a certain range and a few of outside these ranges. There will be outliers too, either very short or very small.

CONCLUSION

Exploratory Data Analysis is essential in the analysis of massive data sets, to be able to ensure that you have the right data for the chosen statistical model. You certainly would not want to figure out at a later stage that the data is not a good fit for the statistical model you are trying to build. A sound EDA must be performed before any data mining, data analysis, or data modeling occurs.

UNIT – III

Data Foundation – SCSA1309

UNIT III DATA ORGANIZATION

Data Structures: Basics – Stack, Queue, Linked List, Tree, Graph - Data Organizational Models- Centralized Model-Embedded Model- Hybrid Model-The Three-Layered structure-Centre of Excellence Model – Roles and Responsibilities- Data GovernanceData Privacy-Data Quality- Data Extraction-Extraction and ETL(Extract,Load,Transform)-Types- Physical -Logical-Data extraction with SQL.

1. Stack

Stack is a linear data structure which follows a particular order in which the operations are performed. In stack, insertion and deletion of elements happen only at one end, i.e., the most recently inserted element is the one deleted first from the set.

This could be explained using a simple analogy, a pile of plates, where one plate is placed on the top of another. Now, when a plate is to be removed, the topmost one is removed. Hence insertion/removal of a plate is done at the topmost position. Few real world examples are given next:

- * Stack of plates in a buffet table. The plate inserted at last will be the first one to be removed out of stack.



- * Stack of Compact Discs



- * Stack of Moulded chairs



* Bangles on Women's Hand



* Books piled on top of each other



These above examples show, stack as a linear list where all insertion and deletion are done only at one end of the list called Top. i.e., Stack implements Last-in, First-out (**LIFO**) policy. Stack can be implemented using an Array or Linked list.

3.1 STACK:

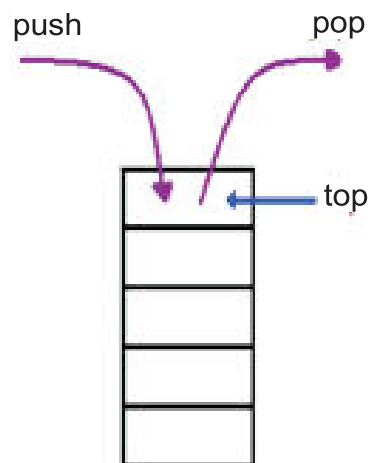


Fig. 3.1: Representation of stack as Array

Where stack concepts can be used in computers? Though there are various applications, simple answer is in function calls. Consider the example:

main ():	def funA():	def funB():	def funC():
_____	_____	_____	_____
funA()	funB()	funC()	_____
_____	_____	_____	_____

In order to keep track of returning point of each active function, a special stack named System stack is used.

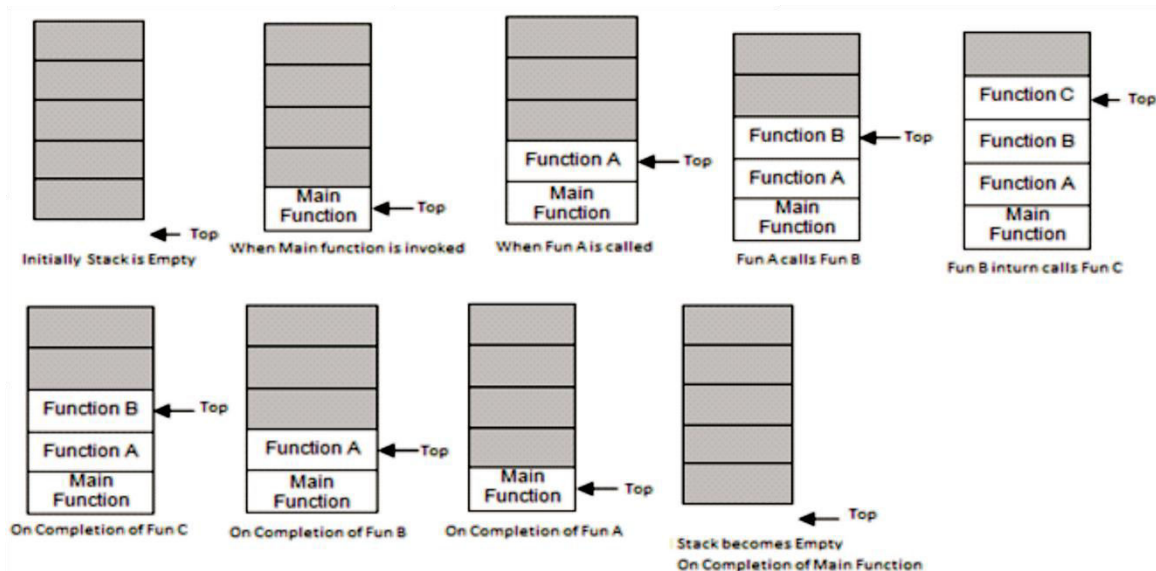


Fig. 3.2: Function calls with Stack

when A calls function B, A is pushed onto system stack. Similarly when B calls C, B is pushed onto stack and so on. Once the execution of function C is complete, the control will remove C from the stack. Thus system stack ensures proper execution order of functions.

The below diagram (Figure 3.3) depicts expansion and shrinking of a stack, initially stack is empty

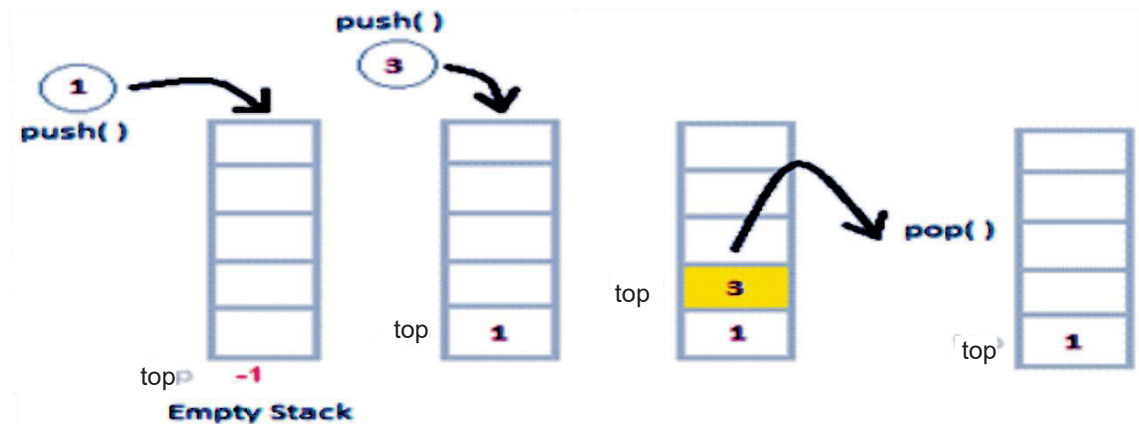


Fig. 3.3: Operations on Stack

In Stack, all operations take place at the “top”. Push(x) operation adds an item x at the top of stack and the pop() operation removes an item from the top of stack.

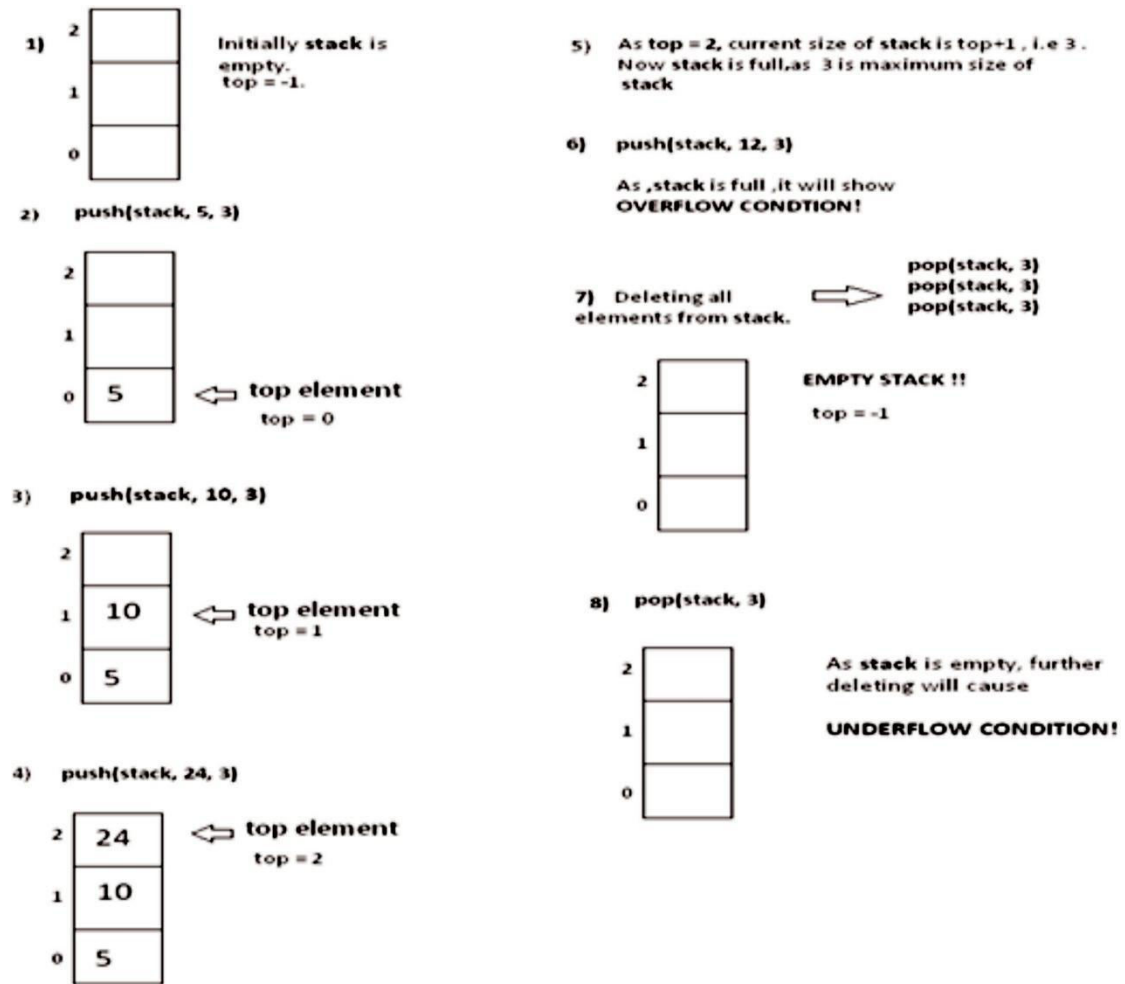
3.1.1 Stack as an ADT (Abstract Data Type):

Stack can be represented using an array. A one dimensional array is used to hold the elements of the stack and a variable “top” is used for representing the index of the top most element. Formally defined as:

```

type def struct stack
{
    int data [size];
    int top;
} s;
//size is constant, represents maximum
//number of elements that can be stored.
    
```

3.1.2 Basic Stack Operations



* **isEmpty:** Returns true if stack is empty, else false.

* **Push:** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.

* **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.

The fundamental operations in stack are given as follows:

STACK-EMPTY(S)

```

if s.top == 0
    return TRUE
else
    return FALSE

```

PUSH(S.x)

```

S.top = S.top+1
S[S.top] = x

```

POP(S)

```

if STACK-EMPTY(S)
    error "underflow"
else
    x = S[S.top]
    S.top = S.top-1
    return x

```

When $S.top = 0$, the stack contains no elements and is empty. If we attempt to pop an element from empty stack, it's called stack underflows, which is normally reported as error. If $S.top$ exceeds size, the stack overflows. In pseudo code implementation we don't represent about stack overflow. Each of the above three stack operations takes $O(1)$ time.

3.1.3 ARRAY REPRESENTATION OF STACKS

Stack is represented as a linear array. In an array-based implementation we maintain the following fields: an array S of a default Size (≥ 1), the variable top that refers to the top element in the stack and the size that refers to the array size. The variable top changes from ≥ 1 to $Size-1$. Stack is said to be empty when $top = -1$, and the stack is full when $top = Size-1$. Consider an example stack with size=10.

10	20	30	40						
0	1	2	Top=3	4	5	6	7	8	9

The variable Top represents the topmost element of the stack. In the above example six more elements could be stored.

OPERATIONS ON STACK

The two basic operations of stack are push and pop. Let's first discuss about array representation of these operations.

Push Operation

The push operation is used to insert an element into the stack. The element is added always at the topmost position of stack. Since the size of array is fixed at the time of declaration, before inserting the value, check the availability of empty location. If $\text{top} = \text{Size} - 1$, stack is full and no more insertion is possible. If an attempt to insert a value in a full stack, an OVERFLOW message gets displayed.

Consider the below example: Size=10

A	B	C	D	E					
0	1	2	3	Top=4	5	6	7	8	9

To insert a new element F, first check if $\text{Top} \neq \text{Size} - 1$. If the condition fails, then increment the Top and store the value. Thus the updated stack is:

Consider the below example: Size=10

A	B	C	D	E	F				
0	1	2	3	4	Top=5	6	7	8	9

Algorithm:

```

Push(X)
if Top == Size-1
    Write "Stack Overflow"
return
else
    Top = Top + 1
    St[Top] = X
End

```

// St represents an Array with
// maximum limit as Size
// X element to be inserted

Pop Operation

The pop operation is used to remove the topmost element from the stack. In this case, first check the presence of element, if $\text{top} \leq -1$ (indicates no array elements), then it indicates empty stack and thereby deletion not possible. If an attempt to delete a value in an empty stack, an UNDERFLOW message gets displayed.

Consider the below example: Size=10

A	B	C	D	E					
0	1	2	3	Top=4	5	6	7	8	9

To delete the topmost element, first check if $\text{Top} \leq -1$. If the condition fails, then decrement the Top. Thus the updated stack is:

A	B	C	D						
0	1	2	Top=3	4	5	6	7	8	9

Algorithm:

```
Pop()
if Top == -1
    Write "Stack Underflow"

else return
    X = St[Top]
    Top = Top-1
```

// St represents an Array
with maximum limit as
Size
// X represents element
removed

Peek Operation

The peek operation returns the topmost element of the stack. Here if stack is not empty, the top element is displayed.

Consider the below example: Size=10

A	B	C	D	E					
0	1	2	3	Top=4	5	6	7	8	9

Here, the peek operation will return E, as it's the topmost element.

Algorithm:

```

Peek()
if Top == -1
    Write "Stack Underflow"
else
    return St[Top]
End

```

// St represents an Array with maximum limit as Size

Implementation of stack operations using arrays in Python

```

class Stack:
    # Constructor

    def __init__(self):
        self.stack = list()
        self.maxSize = 5
        self.top = 0

    # Add element to the Stack
    def push(self, data):
        if self.top >= self.maxSize:
            return ("Stack Full!")
        self.stack.append(data)
        self.top += 1
        return "element inserted"

```

```

# Remove element from the stack
def pop(self):
    if self.top <= 0:
        return ("Stack Empty!")
    item = self.stack.pop()
    self.top -= 1
    return item

# Size of the stack
def size(self):
    return self.top

s = Stack()

print("Push : ", s.push(1))
print("Push : ", s.push(2))
print("Push : ", s.push(3))
print("Push : ", s.push(4))
print("Push : ", s.push(5))
print("Push : ", s.push(6))
print("Size of Array :", s.size())

```


Output:

```
Push : element inserted
Push : element inserted
Push : element inserted
Push : element inserted
Push : element inserted
Push : Stack Full!
Size of Array : 5
Element Popped : 5
Element Popped : 4
Element Popped : 3
Element Popped : 2
Element Popped : 1
Element Popped : Stack Empty!
```

```
print("Element Popped :",s.pop())
```

```
print("Element Popped :",s.pop())
```

```
print("Element Popped :",s.pop())
```

```
print("Element Popped :",s.pop())
```

```
print("Element Popped :",s.pop())
```

```
print("Element Popped :",s.pop())
```

Pros:

- * Easier to use. Array elements can be accessed randomly using the array index.
- * Less memory allocation, no need to track the next node. Data elements are stored in contiguous locations in memory.

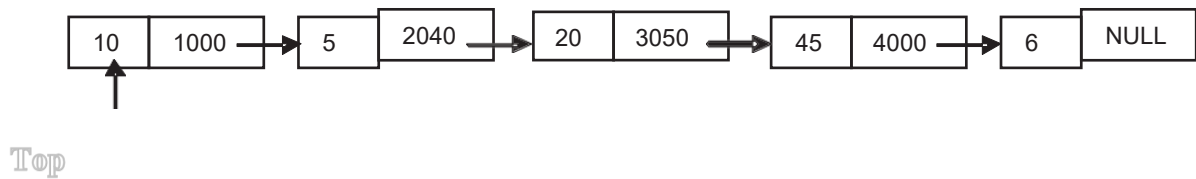
Cons:

- * Fixed size □ cannot increase or decrease the total number of elements.

3.1.4 LINKED REPRESENTATION OF STACK

Stack using arrays was discussed in the previous section, the drawback of the above concept is that the array must be pre-declared i.e., size of array is fixed. If array size cannot be determined in advance, i.e., in case of dynamic storage, Linked List representation could be implemented.

In a linked list, every node has two parts, one that stores data and the other represents address of next node. The head pointer is given as Top, all insertions and deletions occur at the node pointed by Top.



Linked list representation of stack

OPERATIONS ON STACK

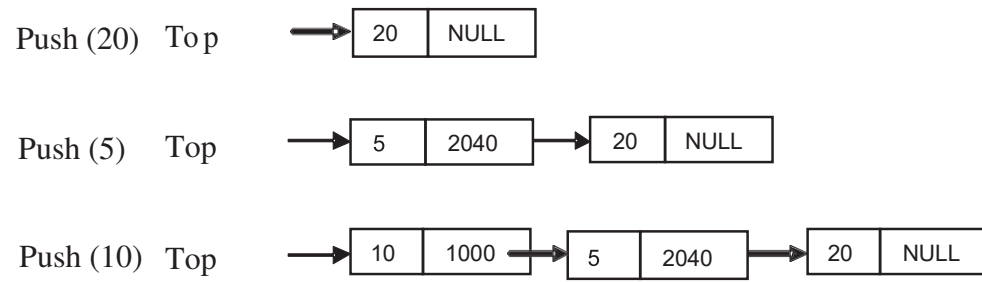
Linked List representation supports the two basic operations of stack, push and pop.

Push operation

Initially, when the stack is empty the pointer top points NULL. When an element is added using the push operation, top is made to point to the latest element whichever is added.

Stack Empty Top →

NULL



Algorithm:

Push(x)Begin

Temp->data = x

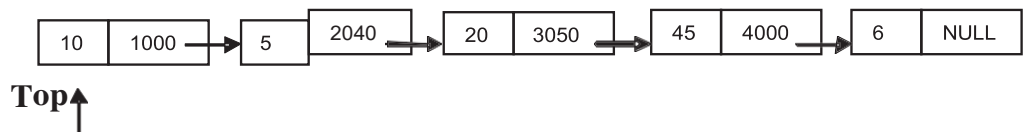
Temp->link = Top

Top = Temp

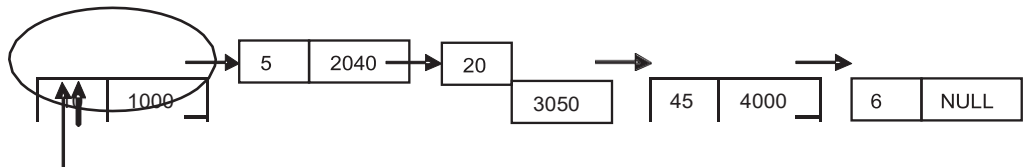
End

Pop operation

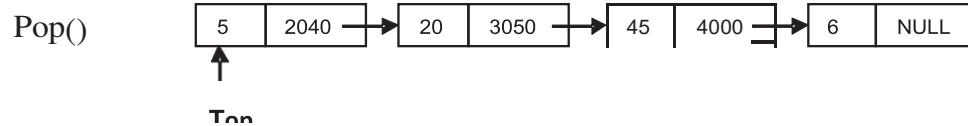
The data in the topmost node of the stack, which is to be removed, is stored in a variable called item. Then a temporary pointer is created to point to top. The top is now safely moved to the next node below it in the stack. Temp node is then deleted and the item is returned.



Element to be removed



Temp



Algorithm:

POP()

```
if Top == NULL
    print "Underflow"
    return

else
    Temp = Top
    Item = Temp->data
    Top = Top->link
    delete Temp
    return Item
```

Implementation of stack operations using linked lists in Python

```
class Node:
    def __init__(self,data):
        self.data = data
        self.next = None
class Stack:
    # default value of head is NULL
    def __init__(self):
        self.head = None
    def isempty(self):
        # Checks if stack is empty
        if self.head == None:
            return True
        else:
            return False

    # Method to add data to the stack
    def push(self,data):
        if self.head == None:
            self.head=Node(data)
        else:
            newnode = Node(data)
            newnode.next = self.head
            self.head = newnode
```

```

# Remove element that is the current head (start of the stack)
def pop(self):
    if self.isempty():
        return None
    else:
        temp = self.head
        self.head = self.head.next
        temp.next = None
        return temp.data
# Returns the head node data
def peek(self):
    if self.isempty():
        return None
    else:
        return self.head.data
# Prints out the stack
def display(self):
    temp = self.head
    if self.isempty():
        print("Stack Underflow")
    else:
        while(temp != None):
            print(temp.data,"->",end = " ")
            temp = temp.next
        print("NULL")
    return

```

```
# Driver code
St = Stack()
St.push(11)
St.push(22)
St.push(33)
St.push(44)

# Display stack elements
print("\nElements in Stack : ")
St.display()

# Print top element of stack
print("\nTop element is ",St.peak())

# Delete top elements of stack
St.pop()
St.pop()

# Display stack elements
print("\nElements in Stack : ")
St.display()

# Print top element of stack
print("\nTop element is ",St.peak())
```

Pros:

- * No fixed size declaration.
- * New elements can be stored anywhere and a reference is created for the new element using pointers.

Cons:

- * Requires extra memory to keep details about next node.
- * Random accessing is not possible in linked lists. The elements will have to be accessed sequentially.

Difference between an Array and Stack ADT**Stack Data Structure:**

- * Size of the stack keeps on changing with insertion/deletion operation.
- * Stack can store elements of different data types [Heterogeneous].

Array Data Structure:

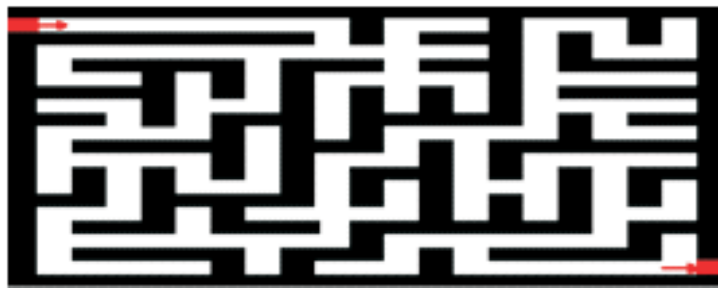
- * Size of the array is fixed at the time of declaration itself
- * Array stores elements of similar data type [Homogeneous].

3.2 APPLICATIONS OF STACK:

In this section, simple applications of stack that are extensively used in computer applications are discussed. Few applications of Stack are listed as below:

- * Stack - undo\redo operation in word processors.
- * Expression evaluation and syntax parsing.
- * Many virtual machines like JVM are stack oriented.
- * DFS Algorithm - Depth First Search
- * Graph Connectivity.
- * LIFO scheduling policy of CPU.
- * When a processor receives an interrupt, it completes its execution of the current instruction, then pushes the process's PCB to the stack and then perform the ISR (Interrupt Service Routine)

- * When a process calls a function inside a function - like recursion, it pushes the current data like local variables onto the stack to be retrieved once the control is returned.
- * Reverse polish AKA postfix notations.
- * Used in IDEs to check for proper parenthesis matching.
- * Browser back-forth button.
- * Another application is an “undo” mechanism in text editors; this operation is accomplished by keeping all text changes in a stack
- * Backtracking. This is a process when you need to access the most recent data element in a series of elements. Think of a labyrinth or maze - how do you find a way from an entrance to an exit?



Once you reach a dead end, you must backtrack. But backtrack to where?, the previous choice point. Therefore, at each choice point you store on a stack all possible choices. Then backtracking simply means popping a next choice from the stack.

Points to Remember

- * A stack is a data structure in which addition of new element or deletion of an existing element always takes place at the same end. This end is often known as top of stack. When an item is added to a stack, the operation is called push, and when an item is removed from the stack the operation is called pop. Stack is also called as Last-In-First-Out (LIFO) list.
- * Stacks are implemented using Arrays or Linked Lists.
- * The storage requirements of linked representation of stack with n elements is $O(n)$ and the time required for all stack operations is given as $O(1)$.

- * All Recursive function calls are implemented using System Stack.

2. Queue

Queue is an abstract data structure, somewhat similar to Stacks. Unlike stacks, a queue is open at both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first. Real world examples are given below.

- * People moving on an escalator. The people who got on the escalator first will be the first one to step out of it.

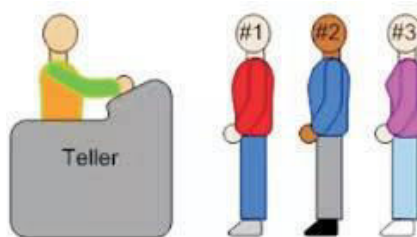


Fig 3.4 Queue

- * Cars lined at a toll bridge. The first car to reach the toll bridge will be the first one to leave.



- * Customers in a ticket counter where the customer came first are served first.



- * Luggage kept on conveyor belts. The bag which was placed first will be the first to come out at the other end.



In all these examples, the element at the first position is served first. Same is the case with queue data structure.

Queue is a linear data structure that permits insertion of new element at one end and deletion of an element at the other end. The end at which the deletion of an element takes place is called Front, and the end at which insertion of a new element can take place is called Rear. The deletion and insertion of elements can take place only at the Front and Rear end of the queue.

Queue can be implemented as an array, or a linked list. The simplest representation is the array representation which follows First in First out (FIFO) behaviour. (Fig 4.1)

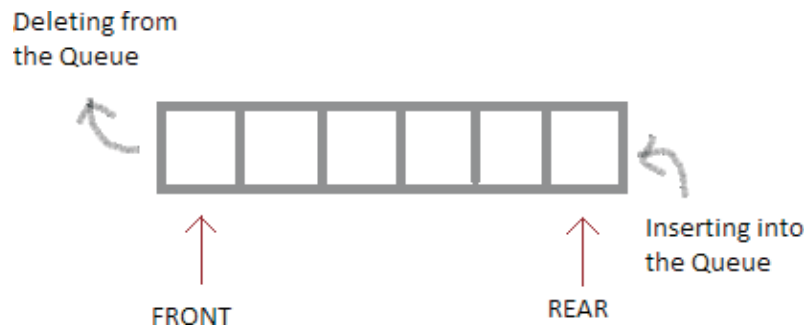


Fig. 3.5 Representation of Queue

3.2.1. Queue as ADT

Abstract data types, commonly abbreviated ADTs, are a way of classifying data structures based on how they are used and the behaviors they provide. They do not specify how the data structure must be implemented but simply provide a minimal expected interface and set of behaviors.

Queue abstract data type holds a collection of elements where they are added at one end called 'Rear' and removed from other end called 'front'. The principle by which a queue is ordered is called **FIFO (First-In First-Out)**. Formally defined as:

```
typedef struct queue
{
    int data[size];
    int front;
    int rear;
}Q;
```

3.2.2. Queue Operation Using Array Representation

Queues can be easily represented using linear arrays. Every queue has two variables i.e. Front and Rear that point to the position from where deletions and insertions can be done respectively. Initially, the value of Front and Rear of the queue is '-1' which represents an empty queue.

An array representation of queue requires three entities:

- (a) An array to hold queue elements.
- (b) A variable to hold the index of the front element.
- (c) A variable to hold the index of the rear element.

Basic Queue Operations

Queue operations may involve initializing or defining the queue, utilizing it, and then deleting it from the memory. The operations that can be performed in queues are listed below:

- * Enqueue() □ Insertion of elements to the end of the queue.
- * Dequeue() □ Deletion of elements from the beginning of the Queue.

Few more functions are required to make the above-mentioned queue operation efficient. These are

- * isfull() □ Checks whether the queue is full.
- * isempty() □ Checks whether the queue is empty

Enqueue Operation

An element can be added to the queue only at the rear end of the queue. Queues maintain two data pointers, Front and Rear. Check if the queue is already full by comparing rear to max - 1. If so, then return an overflow error. The following steps should be taken to enqueue data into a queue (Fig 4.2).

- * **Step 1** □ Check if the queue is full.
- * **Step 2** □ If the queue is full, produce overflow error and exit.

- * **Step 3** □ If the queue is not full, increment rear pointer to point the next empty space.
- * **Step 4** □ Add data element to the location, where the rear is pointing.
- * **Step 5** □ Return success.

Initially Front and Rear points to -1 and the size of the queue is 4.

[0] [1] [2] [3]



Front = Rear = -1

Enqueue 27: Front and Rear is incremented by one and it points to position '0'. Element '27' is stored in location '0'.

[0] [1] [2] [3]

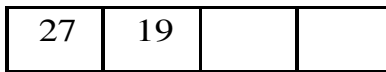


□ □

Front Rear

Enqueue 19: Rear is incremented by 1 and it points to position '1'. Element '19' is stored in location '1'.

[0] [1] [2] [3]

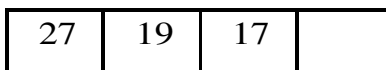


□ □

Front Rear

Enqueue 17: Rear is incremented by 1 and it points to position '2'. Element 17 is stored in location '2'.

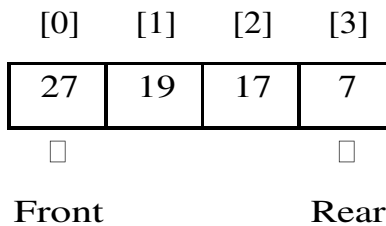
[0] [1] [2] [3]



□ □

Front Rear

Enqueue 7: Rear is incremented by 1 and it points to position '3'. Element '7' is stored in location '3'.



Enqueue 10: Rear=3 i.e) max-1, Queue is Full, so return Overflow. Newelement will not be inserted, because queue is overflow.

Fig. 3.6 Enqueue Operation

Algorithm for Enqueue Operation

```

Enqueue(data)
Begin
    If rear=max-1
        Write "OVERFLOW"
    Endif
    If front=-1 and rear=-1 Set
        front=rear=0
    Else
        Set rear=rear+1
    Endif Queue[rear]=data
End
    
```

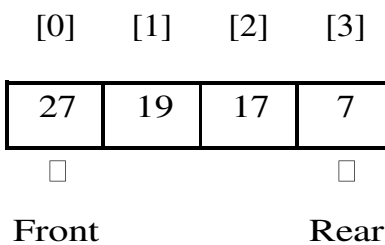
Dequeue Operation

The Dequeue operation deletes the element from the queue pointed by the Front pointer. Before deleting an element from a queue, check for underflow conditions. An underflow condition occurs when trying to delete an element from a queue that is already empty. If FRONT = □1 and REAR = □1, it means there is no element in the queue (Fig 4.3).

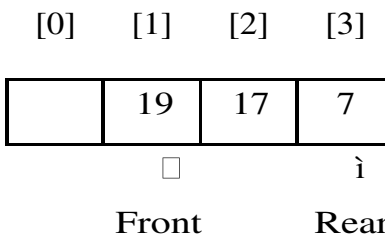
The following steps are taken to perform dequeue operation

- * **Step 1** □ Check if the queue is empty.
- * **Step 2** □ If the queue is empty, produce underflow error and exit.
- * **Step 3** □ If the queue is not empty, access the data where Front is pointing.
- * **Step 4** □ Increment Front pointer to point to the next available data element.
- * **Step 5** □ Return success.

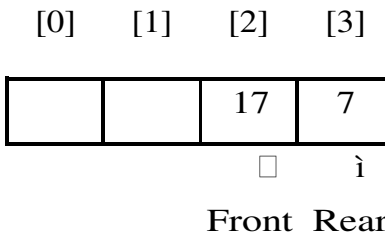
Example: consider the queue given below:



Dequeue() □ Front points to position '0'. So 27 is deleted from the queue and the Front pointer is incremented by 1.



Dequeue() – Front points to position '1'. So 19 is deleted from the queue and the Front pointer is incremented by 1.



After two successive dequeue operation, the queue becomes empty and Front and Rear points to '□1'.

Fig. 3.7 Dequeue Operation

Algorithm for Dequeue Opeartion

```
Dequeue()
Begin
    if queue is empty return
        underflow
    endif
    data= queue[front]
    front= front +1
    return true
End
```

Isfull()

This function helps to check whether the queue is full or not. If aqueue is full, this function returns 'true' else it returns false.

Algorithm

```
Isfull()
Begin
    if rear = MAXSIZE
        return true
    else
        return false
    endif
End
```

IsEmpty()

This function helps to check whether the queue is empty or not. If aqueue is empty, this function returns 'true' else it returns false.

```
IsEmpty()
Begin
    if Front is less than MIN or FRONT is greater than Rearreturn
        true;
    else
        return False
    endif
End
```

Implementation of Queue in Python

```

def enqueue():
    global front, rear, queue, n
    if rear == n-1:
        print("Queue is full")
    else:
        x = int(input("Enter the element :").strip())
        if front == -1:
            front=0
        rear+=1
        queue[rear] = x

        print("Enqueued successfully")
    return

def dequeue():
    global front, rear, queue, n
    if front == -1:
        print("Queue is empty")
    else:
        print("Deleted:",queue[front])
        queue[front]=None

        if front == rear:
            front=-1
            rear=-1

        else:
            front+=1
    return

def display():
    global front, rear, queue, n
    if rear == -1:

```

```

        print("Queue is empty")
    else:
        print("Elements of Queue are :",queue)
    return

n=int(input("Enter the size of Queue:").strip())
front=rear=-1

queue=list(None for i in range(n))
print("Menu:")

print("    1.Enqueue","2.Dequeue","3.Display","4.Exit",sep="\n    ")
while(1):

    choice = int(input("Enter your choice: ").strip())
    if choice==1:
        enqueue()

    elif choice==2:
        dequeue()
    elif choice==3:
        display()

    elif choice==4:
        quit(0)

    else:
        print("Enter valid number")

```

Output:

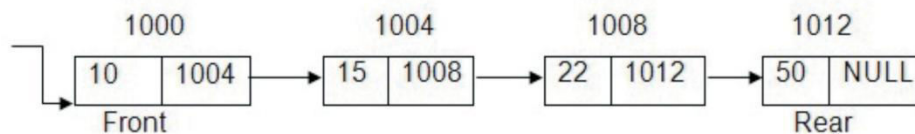
```
Enter the size of Queue:3
Menu:
    1.Enqueue
    2.Dequeue
    3.Display
    4.Exit
Enter your choice: 1
Enter the element :9
Enqueued successfully
Enter your choice: 1
Enter the element :5
Enqueued successfully
Enter your choice: 3
Elements of Queue are : [9, 5, None]
Enter your choice: 1
Enter the element :1
Enqueued successfully
Enter your choice: 1
Queue is full
Enter your choice: 3
Elements of Queue are : [9, 5, 1]
Enter your choice: 2
Deleted: 9
Enter your choice: 3
Elements of Queue are : [None, 5, 1]
Enter your choice: 1
Queue is full
Enter your choice: 2
Deleted: 5
Enter your choice: 2
Deleted: 1
Enter your choice: 2
Queue is empty
Enter your choice: 1
Enter the element :6
Enqueued successfully
Enter your choice: 3
Elements of Queue are : [6, None, None]
Enter your choice:
```

4.1.2 Queue Operation Using Linked List Representation

The major problem with the queue implemented using an array is, it will work for only fixed number of data values. Queue using an array is not suitable when the size of the queue is not known. If a queue is allocated for 50 elements and it hardly uses 20-25 locations, then half of the space will be wasted. A queue data structure can be implemented using a linked list data structure. The queue which is implemented using a linked list can work for an unlimited number of values. That means, queue using linked list can work for the variable size of data (No need to fix the size at the beginning of the implementation).

In a linked queue, every element has two parts, one that stores the data and other stores the address of the next element. The START pointer of the linked list is used as Front and the address of the last element in the queue is denoted as Rear. All insertions will be done at the Rear end and all the deletions will be done at the Front end. If Front = Rear = NULL, then it indicates that the queue is empty. The linked representation of a queue is shown in (Fig 4.4)

Example



Operations

Fig. 3.8 Queue Using Linked List

A queue has two basic operations: insert and delete. The insert operation adds an element to the end of the queue, and the delete operation removes an element from the front or the start of the queue.

To implement queue using linked list, the following steps to be done before implementing actual operations.

Step 1 - Define a 'Node' structure with two members data and next. Step

2 - Define two Node pointers 'Front' and 'Rear' and set both to NULL.

EnQueue(data) - Inserting an element into the Queue

The enqueue operation is used to insert an element into a queue. The new element is added as the last element of the queue. Consider the linked queue shown in Fig 4.4.

To insert an element with value 25, first check if Front=NULL. If the condition holds, then the queue is empty. Create a new node; store the value in its data part and NULL in its next part. The new node will then be called both Front and Rear. However, if Front!= NULL, then insert the new node at the Rear end of the linked queue and name this new node as Rear. The updated queue is shown in Fig. 4.5.

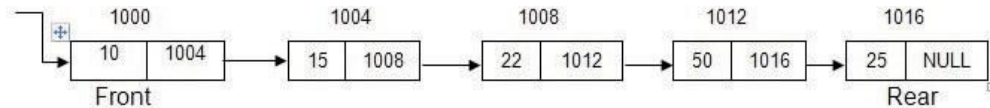


Fig. 3.9 Linked Queue after inserting new node

The following steps are taken to perform enqueue operation:

- * Step 1 - Create a newNode with given data and set 'newNode □ next' to NULL.
- * Step 2 - Check whether queue is Empty (rear □ □ NULL)
- * Step 3 - If it is Empty then, set front = newNode and rear = newNode.
- * Step 4 - If it is Not Empty then, set rear □ next = newNode and rear = newNode.

Algorithm:

```

Enqueue(self):
    read x

    newnode=Node(x)

    if self.front==None:

        self.front=newnode

    else:

        self.rear.next=newnode
        self.rear=newnode
        print("Enqueued Successfully")

End enqueue

```


Queue() - Deleting an Element from Queue

The dequeue operation is used to delete the element that is first inserted in a queue, i.e., the element whose address is stored in Front. However, before deleting the value, first check if Front=NULL because if this is the case, then the queue is empty and no more deletions can be done. If an attempt is made to delete a value from a queue that is already empty, an underflow message is printed. Consider the queue shown in Fig. 4.6.

To delete an element, check if Front=NULL. If the condition is false, then delete the node pointed by Front. The Front will now point to the second element of the linked queue. The linked queue after deleting the element is shown in Fig. 4.7.

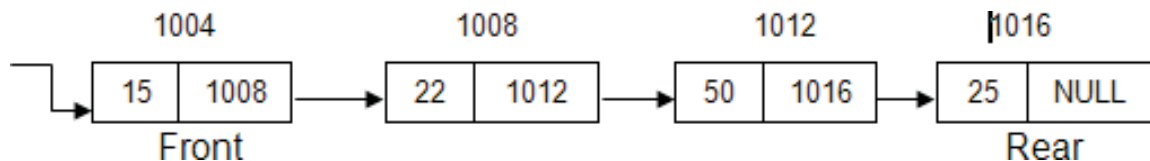


Fig. 3.10 Linked Queue after deletion of an element

The following steps are taken to perform dequeue operation:

- * Step 1 - Check whether queue is Empty (Front \square NULL).
- * Step 2 - If it is Empty, then display "Queue is Empty!!! Deletion is not possible!!! and terminate from the function
- * Step 3 - If it is Not Empty then, define a Node pointer 'temp' and set it to 'Front'.
- * Step 4 - Then set 'Front = Front \square next' and delete 'temp' (free(temp)).

Algorithm:

```
Dequeue(self):  
    read x  
  
    newnode=Node(x)  
  
    if self.front==None:  
        self.front=newnode  
  
    else:  
        self.rear.next=newnode  
        self.rear=newnode  
        print("Enqueued Successfully")  
  
End dequeue
```

Display the elements of Queue

The following steps are taken to perform display operation:

- * Step 1 - Check whether queue is Empty (Front \square \square NULL).
- * Step 2 - If it is Empty then, display 'Queue is Empty!!!' and terminate the function.
- * Step 3 - If it is Not Empty then, define a Node pointer 'temp' and initialize with Front.
- * Step 4 - Display 'temp \square data' and move it to the next node. Repeat the same until 'temp' reaches to 'rear' (temp \square next \square NULL).
- * Step 5 - Finally! Display 'temp \square data \square NULL'.

Implementation of Queue Data structure using Linked List in Python

```
class Node:
    def __init__(self, initval=None):
        self.value=initval
        self.next=None

class Queue:
    def __init__(self):
        self.front=None
        self.rear=None

    def enqueue(self):
        x=int(input("Enter the element:").strip())
        newnode=Node(x)

        if self.front==None:
            self.front=newnode

        else:
            self.rear.next=newnode
            self.rear=newnode
            print("Enqueued Successfully")
            return

    def dequeue(self):
        if self.front==None:
            print("Queue is empty")

        else:
            print("Deleted:",self.front.value)
            if self.front.next==None:
```

```

        self.front=self.rear=None
    else:
        self.front=self.front.next
    return

def display(self):
    if self.front==None:
        print("Queue is empty")
    else:
        print("Elements of Queue are : ")
        temp=Node()
        self.temp=self.front

        while self.temp!=None:
            print(self.temp.value,end="-->")
            self.temp=self.temp.next

        print("None")

print(" 1.Enqueue","2.Dequeue","3.Display","4.Exit",sep="\n  ")
while(1):
    choice = int(input("Enter your choice:").strip())
    if choice==1:
        q.enqueue()

    elif choice==2:
        q.dequeue()
    elif choice==3:
        q.display()
    elif choice==4:
        quit(0)

```

Output:

```
Menu:
  1.Enqueue
  2.Dequeue
  3.Display
  4.Exit
Enter your choice:1
Enter the element:9
Enqueued Successfully
Enter your choice:1
Enter the element:7
Enqueued Successfully
Enter your choice:3
Elements of Queue are : 9-->7-->None
Enter your choice:1
Enter the element:8
Enqueued Successfully
Enter your choice:3
Elements of Queue are : 9-->7-->8-->None
Enter your choice:2
Deleted: 9
Enter your choice:3
Elements of Queue are : 7-->8-->None
Enter your choice:1
Enter the element:2
Enqueued Successfully
Enter your choice:3
Elements of Queue are : 7-->8-->2-->None
Enter your choice:2
Deleted: 7
Enter your choice:2
Deleted: 8
Enter your choice:2
Deleted: 2
Enter your choice:3
Queue is empty
Enter your choice:|
```

4.2 CIRCULAR QUEUES

In a linear Queue, elements are inserted until queue becomes full. But once the queue becomes full, the next element cannot be inserted until all the elements are deleted from the queue. Look at the queue shown in Fig. 4.8.

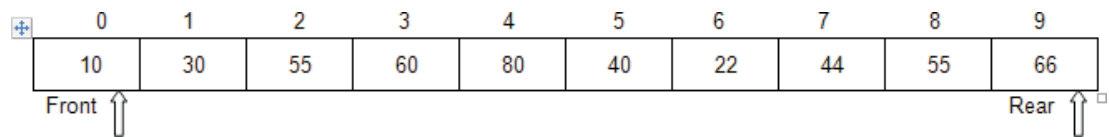


Fig. 3.11 Linear Queue

Now, if you want to insert another value, it will not be possible because the queue is completely full. There is no empty space where the value can be inserted. Consider a scenario in which three successive deletions are made (Fig 4.9.)

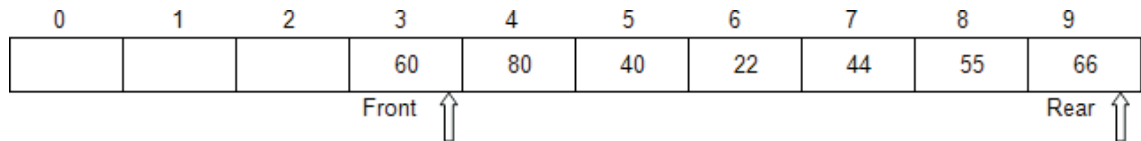


Fig. 3.12 Linear Queue after three successive deletions

This situation also says that Queue is full and also new element cannot be inserted because 'rear' is still at the last position. In the above situation, even though there is empty positions in the queue new element cannot be inserted. This is the major problem in a linear queue data structure. This is overcome by using circular queue data structure.

A circular queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. Elements are added at the Rear end and deleted at Front end of the Queue. (Fig 4.10)

Graphical representation of a circular queue is as follows.

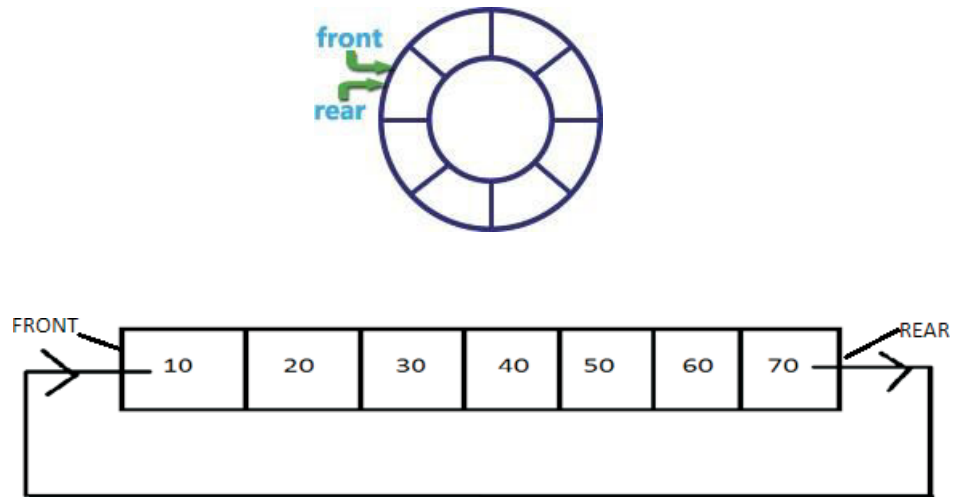


Fig. 3.13 Circular Queue

All operations of a simple queue can be applied to a circular queue also. The Front and Rear pointers have to be updated to the first location after the maximum size is reached.

Operations on Circular Queue
Enqueue operation:

In a circular queue, enqueue() is a function which is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at rear position.

Consider the Fig. 4.9. Front and Rear points to the positions 3 and 9 respectively, now enqueue the element 75 in to the circular queue. The new element is inserted in the position '0' and rear points to '0'. (Fig 4.11)

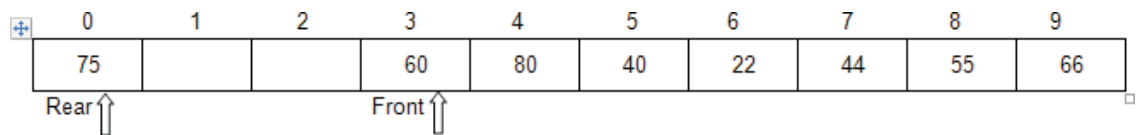


Fig. 3.14 Circular Queue

Following steps are to be followed to insert an element into the circular queue.

- * Step 1 - Check whether queue is FULL. (Rear \neq SIZE-1 && Front \neq 0)
- * Step 2 - If it is FULL, then display "Queue is FULL!!! Insertion is not possible!!!" and terminate the function.
- * Step 3 - If it is NOT FULL, then check Front \neq Rear \neq 1. If it is TRUE, then set Rear = Front = 0.
- * Step 4 - Increment Rear value by (rear+1)%MAX and set queue[rear] = value.

Algorithm:

```
CircularEnqueue()  
if(front==(rear+1)%MAX )  
    Write "queue is full"  
else  
    Read x  
    if front == -1  
        front=rear=0  
        rear=(rear+1)%n  
    else  
  
    End If  
End if
```

Dequeue operation:

In a circular queue, deQueue() is a function used to delete an element from the circular queue. In a circular queue, the element is always deleted from Front position. The deQueue() function doesn't take any value as a parameter. Consider the Fig.4.11, dequeue an element from the circular queue. After deleting the element, the Front pointer points to the position '4'. (Fig. 4.12).

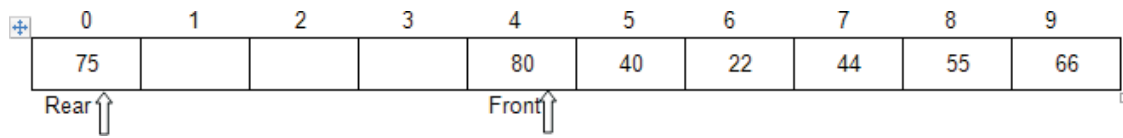


Fig. 3.15 Circular Queue after deletion of an element

Following steps are to be followed to delete an element from the circular queue.

- * Step 1 - Check whether queue is EMPTY. (Front =1 && Rear=1)
- * Step 2 - If it is EMPTY, then display “Queue is EMPTY!!! Deletion is not possible!!!” and terminate the function.
- * Step 3 - If it is NOT EMPTY, then display queue[Front] as deleted element then check Front and Rear are same, if they are same then assign both Front and Rear with 0 else increment the front value by $(front+1)\%MAX$.

Algorithm:

```

Circulardequeue()
  If front == -1
    Write " Queue is empty"
  Else
    Writre(queue[front])
    Queue[front]=0
    if front == rear
      front = rear = -1
    else
      front=(front+1)%n
  
```

Implementation of circular queue - Python

```
def enqueue():
    global front, rear, queue, n
    if front == (rear+1)%n:
        print("Queue is full")
    else:
        x= int(input("Enter the element : ").strip())
        if front == -1:
            front=rear=0
        else:
            rear=(rear+1)%n
        queue[rear]=x

    print("Enqueued successfully")
    return

def dequeue():
    global front, rear, queue, n
    if rear== -1:
        print("Queue is empty")
    else:
        print("Deleted : ",queue[front])
        queue[front]=None

        if front == rear:
            front = rear = -1
        else:
            front=(front+1)%n

    return

def display():
```

```

def enqueue():
    global front, rear, queue, n
    if front == (rear+1)%n:
        print("Queue is full")
    else:
        x= int(input("Enter the element : ").strip())
        if front == -1:
            front=rear=0
        else:
            rear=(rear+1)%n
        queue[rear]=x
    print("Enqueued successfully")
    return

def dequeue():
    global front, rear, queue, n
    if rear== -1:
        print("Queue is empty")
    else:
        print("Deleted : ",queue[front])
        queue[front]=None
        if front == rear:
            front = rear = -1
        else:
            front=(front+1)%n
    return

def display():
    global front, rear, queue, n
    if rear == -1:
        print("Queue is empty")
    else:
        print("Elements of Queue are : ",queue)

```

Output

```
Enter the size of circular queue : 3
Menu:
    1.Enqueue
    2.Dequeue
    3.Display
    4.Exit
Enter your choice : 1
Enter the element : 3
Enqueued successfully
Enter your choice : 3
Elements of Queue are : [3, None, None]
Enter your choice : 1
Enter the element : 10
Enqueued successfully
Enter your choice : 1
Enter the element : 15
Enqueued successfully
Enter your choice : 3
Elements of Queue are : [3, 10, 15]
Enter your choice : 2
Deleted : 3
Enter your choice : 3
Elements of Queue are : [None, 10, 15]
Enter your choice : 1
Enter the element : 25
Enqueued successfully
Enter your choice : 3
Elements of Queue are : [25, 10, 15]
Enter your choice : 2
Deleted : 10
Enter your choice : 3
Elements of Queue are : [25, None, 15]
Enter your choice : |
```

4.3 PRIORITY QUEUES

Priority Queue is more specialized data structure than Queue. A priority queue is a data structure in which each element is assigned a priority. The priority of the element will be used to determine the order in which the elements to be processed. The general rules of processing the elements of a priority queue are

- * An element with higher priority is processed before an element with a lower priority.
- * Two elements with the same priority are processed on a First-come-First-served (FCFS) basis.

In Priority queue elements are ordered by a key value, so that elements with the lowest value of key are at front and elements with the highest value of key is at rear or vice versa. Priority is assigned to the elements based on its key value. The priority of the element can be set based on various factors. Priority queues are widely used in operating systems to execute the highest priority to process first. The priority of the process may be set based on the CPU time it requires to get executed completely. For example, if there are three processes, where the first process needs 5 ns to complete, the second process needs 4 ns, and the third process needs 7 ns, then the second process will have the highest priority and will thus be the first to be executed. However, CPU time is not the only factor that determines the priority; rather it is just one among several factors. Another factor is the importance of one process over another.

Array Representation of a Priority Queue

Priority queue can be implemented in several ways. The simplest way is to add the elements in to queue according to the order (ascending or descending). Elements with the highest value of key are at front and elements with the lowest value of key are at rear or vice versa. Priority is assigned to the elements based on its key value.

In the second form, a separate queue for each priority number is maintained. Every individual queue will have its own FRONT and REAR pointers. A two-dimensional array is used for this purpose where each queue will be allocated the same amount of space.

Applications of priority Queue:

1. CPU Scheduling.
2. Graph algorithms like:
 - Dijkstra's shortest path algorithm
 - Prim's minimum spanning tree
3. All priority based application implemented in/with queue.

Implementation of priority queue in python:

```
def enqueue():
    global front, rear, queue, n
    if rear == n-1:
        print("Queue is full")
    else:
        x = int(input("Enter the Element : "))
        if front == -1:
            front=0
            rear+=1
            c=1

        # Code Below finds the correct position of the Element to be enqueued
        for i in range(front,n):
            if queue[i]==None:
                queue[i]=c=0

            if queue[i]<=x:
                queue.insert(i,x)
                if c==0:
                    queue[i+1]=None
                queue=queue[:n]
                break

        print("Enqueued successfully")
def dequeue():
    global front, rear, queue, n
    if front == -1:
        print("Queue is empty")
    else:
```

```

        rear=-1
    else:
        front+=1
    return

def display():
    global front, rear, queue, n
    if rear == -1:
        print("Queue is empty")
    else:
        print("Elements of Queue are : ",queue)
    return

n=int(input("Enter the size of the Queue : ").strip())
front=rear=-1
print("    1.Enqueue","2.Dequeue","3.Display","4.Exit",sep="\n    ")
while(1):
    choice = int(input("Enter your choice:").strip())
    if choice==1:
        enqueue()

    elif choice==2:
        dequeue()
    elif choice==3:
        display()

    elif choice==4:
        quit()

```

Output:

```
Enter the size of the Queue : 3
Menu:
    1.Enqueue
    2.Dequeue
    3.Display
    4.Exit
Enter your choice : 1
Enter the Element : 9
Enqueued successfully
Enter your choice : 1
Enter the Element : 8
Enqueued successfully
Enter your choice : 3
Elements of Queue are : [9, 8, None]
Enter your choice : 1
Enter the Element : 20
Enqueued successfully
Enter your choice : 3
Elements of Queue are : [20, 9, 8]
Enter your choice : 2
Deleted : 20
Enter your choice : 3
Elements of Queue are : [None, 9, 8]
Enter your choice : 1
Queue is full
Enter your choice : 2
Deleted : 9
Enter your choice : 2
Deleted : 8
Enter your choice : 3
Queue is empty
Enter your choice : 1
Enter the Element : 5
Enqueued successfully
Enter your choice : 3
Elements of Queue are : [5, None, None]
Enter your choice : 1
Enter the Element : 50
Enqueued successfully
Enter your choice : 3
Elements of Queue are : [50, 5, None]
Enter your choice : |
```


4.4 APPLICATIONS OF QUEUE

- * Queue is useful in CPU scheduling and Disk Scheduling. When multiple processes require CPU at the same time, various CPU scheduling algorithms are used which are implemented using Queuedata structure.
- * When data is transferred asynchronously between two processes, Queue is used for synchronization. Examples: IO Buffers, pipes, fileIO, etc.
- * In print spooling, documents are loaded into a buffer and then the printer pulls them off the buffer at its own rate. Spooling also lets you place a number of print jobs on a queue instead of waiting for each one to finish before specifying the next one.
- * Breadth First search in a Graph .It is an algorithm for traversing or searching graph data structures. It starts at some arbitrary node of a graph and explores the neighbor nodes first, before moving to the next level neighbors.This Algorithm uses Queue data structure.
- * Handling of interrupts in real-time systems. The interrupts are handled in the same order as they arrive, First come first served.
- * In real life, Call Center phone systems will use Queues, to hold people calling them in an order, until a service representative is free.
- * Queues are widely used as waiting lists for a single shared resource like printer, disk, CPU.

Linked List

A linked list is a sequence of data structures, which are connected together via links.

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following are the important terms to understand the concept of Linked List.

Link – Each link of a linked list can store a data called an element.

Next – Each link of a linked list contains a link to the next link called Next.

LinkedList – A Linked List contains the connection link to the first link called First.

Linked List Representation

Linked list can be visualized as a chain of nodes, where every node points to the next node.



As per the above illustration, following are the important points to be considered.

Linked List contains a link element called first.

Each link carries a data field(s) and a link field called next.

Each link is linked with its next link using its next link.

Last link carries a link as null to mark the end of the list.

Types of Linked List

Following are the various types of linked list.

Singly Linked List – Item navigation is forward only.

Doubly Linked List – Items can be navigated forward and backward.

Circular Linked List – Last item contains link of the first element as next and the first element has a link to the last element as previous.

Basic Operations

Following are the basic operations supported by a list.

Insertion – Adds an element at the beginning of the list.

Deletion – Deletes an element at the beginning of the list.

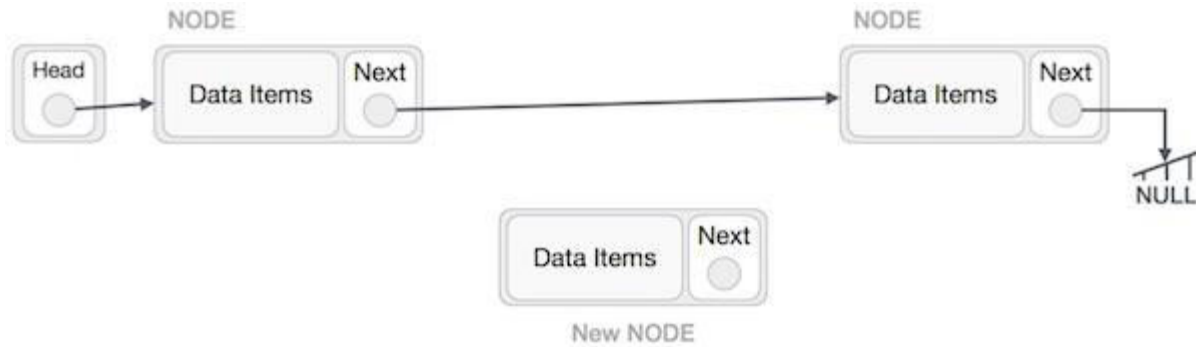
Display – Displays the complete list.

Search – Searches an element using the given key.

Delete – Deletes an element using the given key.

Insertion Operation

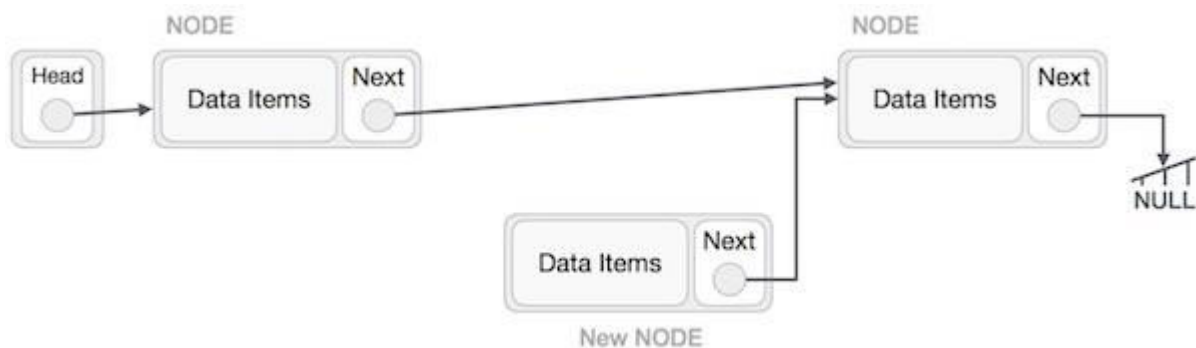
Adding a new node in linked list is a more than one step activity. We shall learn this with diagrams here. First, create a node using the same structure and find the location where it has to be inserted.



Imagine that we are inserting a node **B** (NewNode), between **A** (LeftNode) and **C** (RightNode). Then point B.next to C –

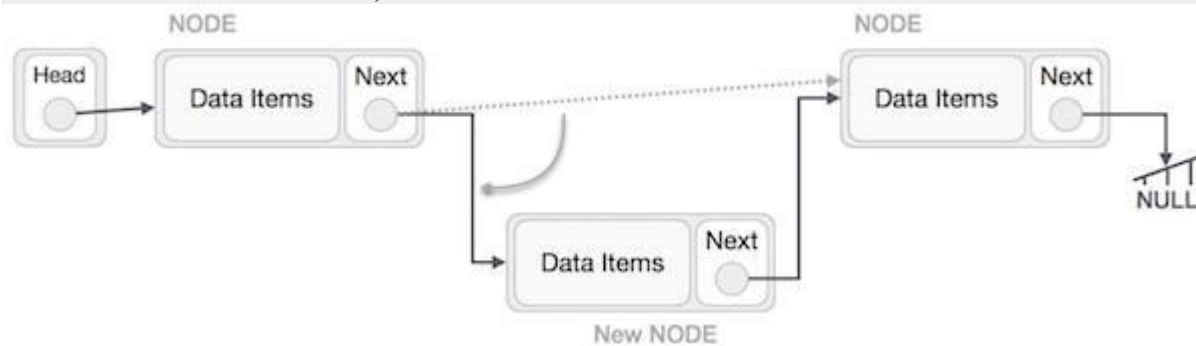
`NewNode.next -> RightNode;`

It should look like this –



Now, the next node at the left should point to the new node.

`LeftNode.next -> NewNode;`



This will put the new node in the middle of the two. The new list should look like this –



Similar steps should be taken if the node is being inserted at the beginning of the list. While inserting it at the end, the second last node of the list should point to the new node and the new node will point to NULL.

Deletion Operation

Deletion is also a more than one step process. We shall learn with pictorial representation. First, locate the target node to be removed, by using searching algorithms.



The left (previous) node of the target node now should point to the next node of the target node –

`LeftNode.next -> TargetNode.next;`

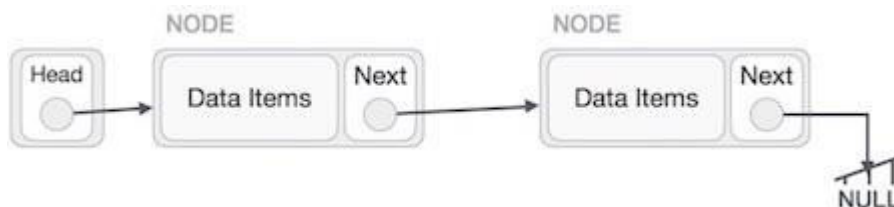


This will remove the link that was pointing to the target node. Now, using the following code, we will remove what the target node is pointing at.

`TargetNode.next -> NULL;`

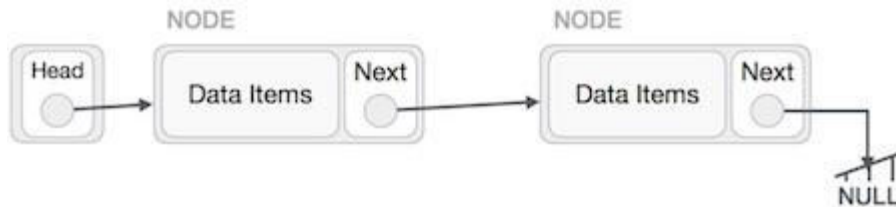


We need to use the deleted node. We can keep that in memory otherwise we can simply deallocate memory and wipe off the target node completely.

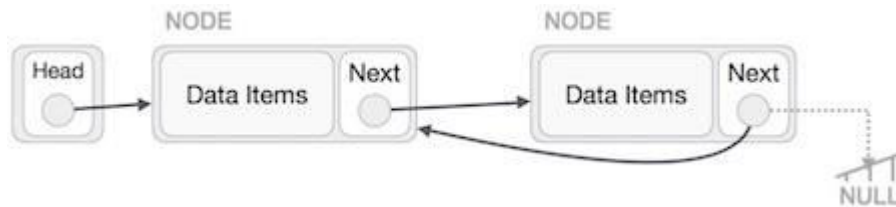


Reverse Operation

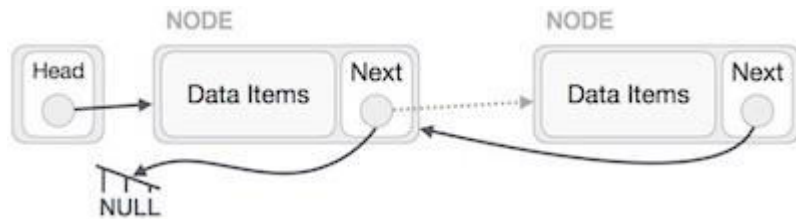
This operation is a thorough one. We need to make the last node to be pointed by the head node and reverse the whole linked list.



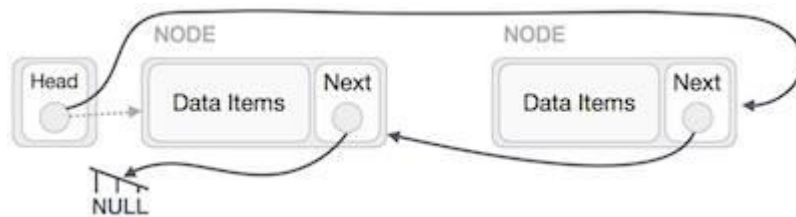
First, we traverse to the end of the list. It should be pointing to NULL. Now, we shall make it point to its previous node –



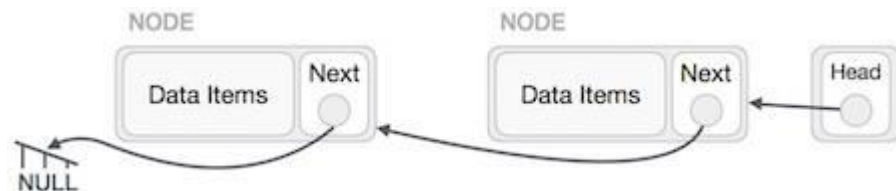
We have to make sure that the last node is not the last node. So we'll have some temp node, which looks like the head node pointing to the last node. Now, we shall make all left side nodes point to their previous nodes one by one.



Except the node (first node) pointed by the head node, all nodes should point to their predecessor, making them their new successor. The first node will point to NULL.



We'll make the head node point to the new first node by using the temp node.



Tree Data Structure

We read the linear data structures like an array, linked list, stack and queue in which all the elements are arranged in a sequential manner. The different data structures are used for different kinds of data.

Some factors are considered for choosing the data structure:

- **What type of data needs to be stored?:** It might be a possibility that a certain data structure can be the best fit for some kind of data.
- **Cost of operations:** If we want to minimize the cost for the operations for the most frequently performed operations. For example, we have a simple list on which we have to perform the search operation; then, we can create an array in which elements are stored in sorted order to perform the **binary search**. The binary search works very fast for the simple list as it divides the search space into half.
- **Memory usage:** Sometimes, we want a data structure that utilizes less memory.

A **tree** is also one of the data structures that represent hierarchical data. Suppose we want to show the employees and their positions in the hierarchical form then it can be represented as shown below:

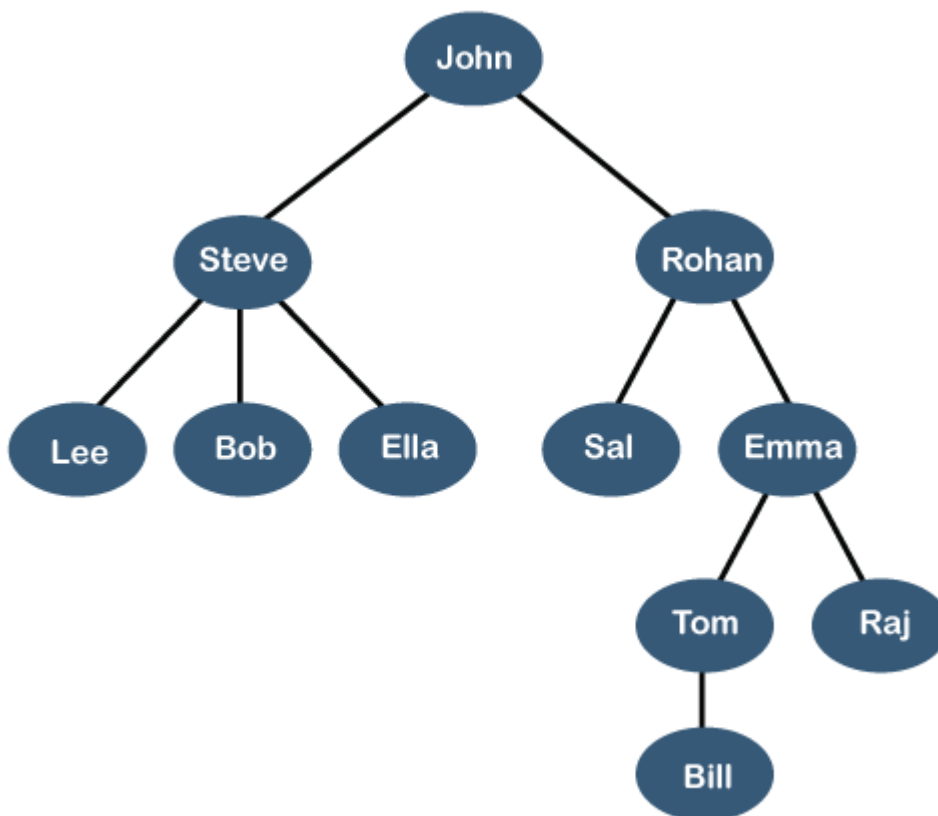


Fig. 3.16 Tree

The above tree shows the **organization hierarchy** of some company. In the above structure, **john** is the **CEO** of the company, and John has two direct reports named as **Steve** and **Rohan**. Steve has three direct reports named **Lee**, **Bob**, **Ella** where **Steve** is a manager. Bob has two direct reports named **Sal** and **Emma**. **Emma** has two direct reports named **Tom** and **Raj**. Tom has one direct report named **Bill**. This particular logical structure is known as a **Tree**. Its structure is similar to the real tree, so it is named a **Tree**. In this structure, the **root** is at the top, and its branches are moving in a downward direction. Therefore, we can say that the Tree data structure is an efficient way of storing the data in a hierarchical way.

Let's understand some key points of the Tree data structure.

- A tree data structure is defined as a collection of objects or entities known as nodes that are linked together to represent or simulate hierarchy.
- A tree data structure is a non-linear data structure because it does not store in a sequential manner. It is a hierarchical structure as elements in a Tree are arranged in multiple levels.
- In the Tree data structure, the topmost node is known as a root node. Each node contains some data, and data can be of any type. In the above tree structure, the node contains the name of the employee, so the type of data would be a string.
- Each node contains some data and the link or reference of other nodes that can be called children.

Some basic terms used in Tree data structure.

Let's consider the tree structure, which is shown below:

Introduction to Trees

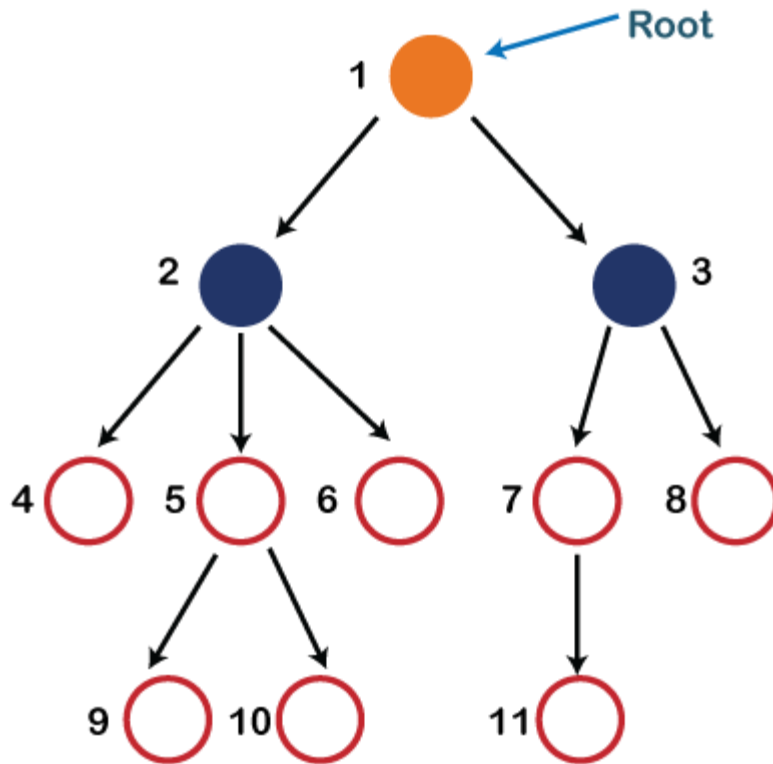


Fig. 3.17 General Tree

In the above structure, each node is labeled with some number. Each arrow shown in the above figure is known as a **link** between the two nodes.

- **Root:** The root node is the topmost node in the tree hierarchy. In other words, the root node is the one that doesn't have any parent. In the above structure, node numbered 1 is **the root node of the tree**. If a node is directly linked to some other node, it would be called a parent-child relationship.
- **Child node:** If the node is a descendant of any node, then the node is known as a child node.
- **Parent:** If the node contains any sub-node, then that node is said to be the parent of that sub-node.
- **Sibling:** The nodes that have the same parent are known as siblings.
- **Leaf Node:-** The node of the tree, which doesn't have any child node, is called a leaf node. A leaf node is the bottom-most node of the tree. There can be any number of leaf nodes present in a general tree. Leaf nodes can also be called external nodes.
- **Internal nodes:** A node has atleast one child node known as an **internal**

- **Ancestor node:-** An ancestor of a node is any predecessor node on a path from the root to that node. The root node doesn't have any ancestors. In the tree shown in the above image, nodes 1, 2, and 5 are the ancestors of node 10.
- **Descendant:** The immediate successor of the given node is known as a descendant of a node. In the above figure, 10 is the descendant of node 5.

Properties of Tree data structure

- **Recursive data structure:** The tree is also known as a **recursive data structure**. A tree can be defined as recursively because the distinguished node in a tree data structure is known as a **root node**. The root node of the tree contains a link to all the roots of its subtrees. The left subtree is shown in the yellow color in the below figure, and the right subtree is shown in the red color. The left subtree can be further split into subtrees shown in three different colors. Recursion means reducing something in a self-similar manner. So, this recursive property of the tree data structure is implemented in various applications.

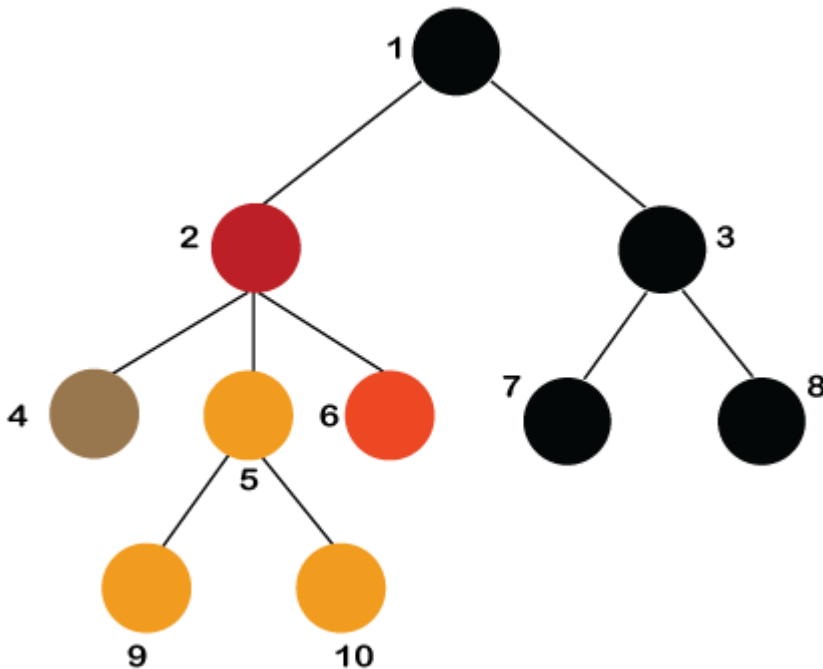


Fig. 3.18 Tree Structure

- **Number of edges:** If there are n nodes, then there would be $n-1$ edges. Each arrow in the structure represents the link or path. Each node, except the root node, will have at least one incoming link known as an edge. There would be one link for the parent-child relationship.
- **Depth of node x :** The depth of node x can be defined as the length of the path from the root to the node x . One edge contributes one-unit length in the path. So, the depth of node x can

also be defined as the number of edges between the root node and the node x. The root node has 0 depth.

- **Height of node x:** The height of node x can be defined as the longest path from the node x to the leaf node.

Based on the properties of the Tree data structure, trees are classified into various categories.

Implementation of Tree

The tree data structure can be created by creating the nodes dynamically with the help of the pointers. The tree in the memory can be represented as shown below:

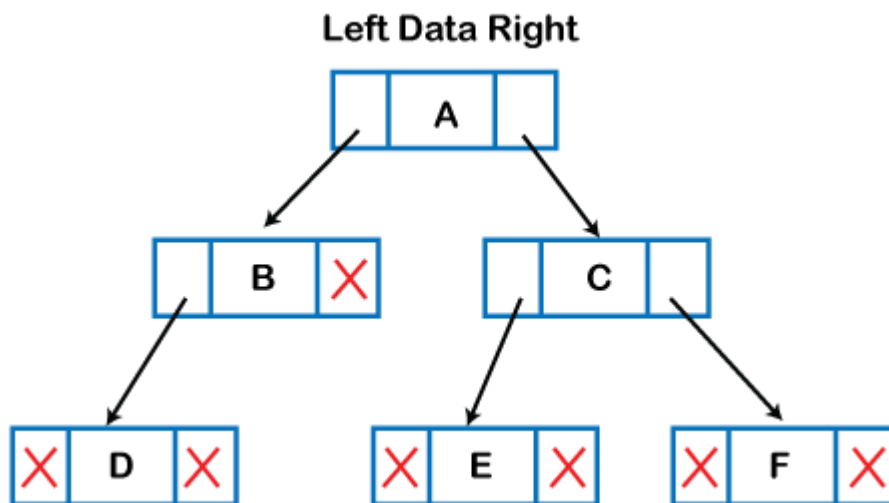


Fig. 3.19 Tree Implementation

The above figure shows the representation of the tree data structure in the memory. In the above structure, the node contains three fields. The second field stores the data; the first field stores the address of the left child, and the third field stores the address of the right child.

In programming, the structure of a node can be defined as:

```
struct node
{
    int data;
    struct node *left;
    struct node *right;
```

}

The above structure can only be defined for the binary trees because the binary tree can have utmost two children, and generic trees can have more than two children. The structure of the node for generic trees would be different as compared to the binary tree.

Applications of trees

The following are the applications of trees:

- **Storing naturally hierarchical data:** Trees are used to store the data in the hierarchical structure. For example, the file system. The file system stored on the disc drive, the file and folder are in the form of the naturally hierarchical data and stored in the form of trees.
- **Organize data:** It is used to organize data for efficient insertion, deletion and searching. For example, a binary tree has a $\log N$ time for searching an element.
- **Trie:** It is a special kind of tree that is used to store the dictionary. It is a fast and efficient way for dynamic spell checking.
- **Heap:** It is also a tree data structure implemented using arrays. It is used to implement priority queues.
- **B-Tree and B+Tree:** B-Tree and B+Tree are the tree data structures used to implement indexing in databases.
- **Routing table:** The tree data structure is also used to store the data in routing tables in the routers.

Types of Tree data structure

The following are the types of a tree data structure:

- **General tree:** The general tree is one of the types of tree data structure. In the general tree, a node can have either 0 or maximum n number of nodes. There is no restriction imposed on the degree of the node (the number of nodes that a node can contain). The topmost node in a

general tree is known as a root node. The children of the parent node are known as **subtrees**.

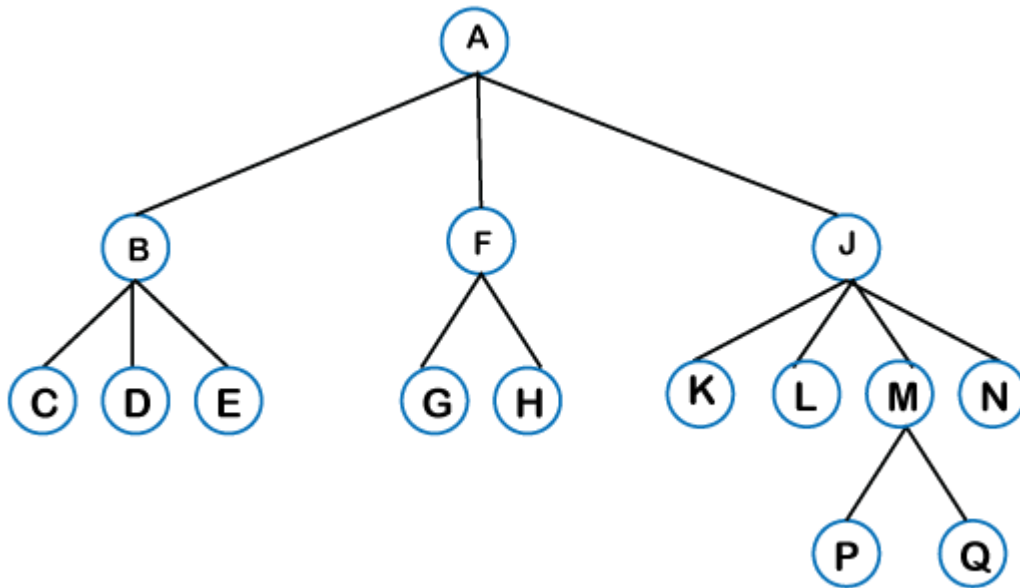


Fig 3.20 General Trees

There can be **n** number of subtrees in a general tree. In the general tree, the subtrees are unordered as the nodes in the subtree cannot be ordered.

Every non-empty tree has a downward edge, and these edges are connected to the nodes known as **child nodes**. The root node is labeled with level 0. The nodes that have the same parent are known as **siblings**.

- **Binary tree:** Here, binary name itself suggests two numbers, i.e., 0 and 1. In a binary tree, each node in a tree can have utmost two child nodes. Here, utmost means whether the node

has 0 nodes, 1 node or 2 nodes.

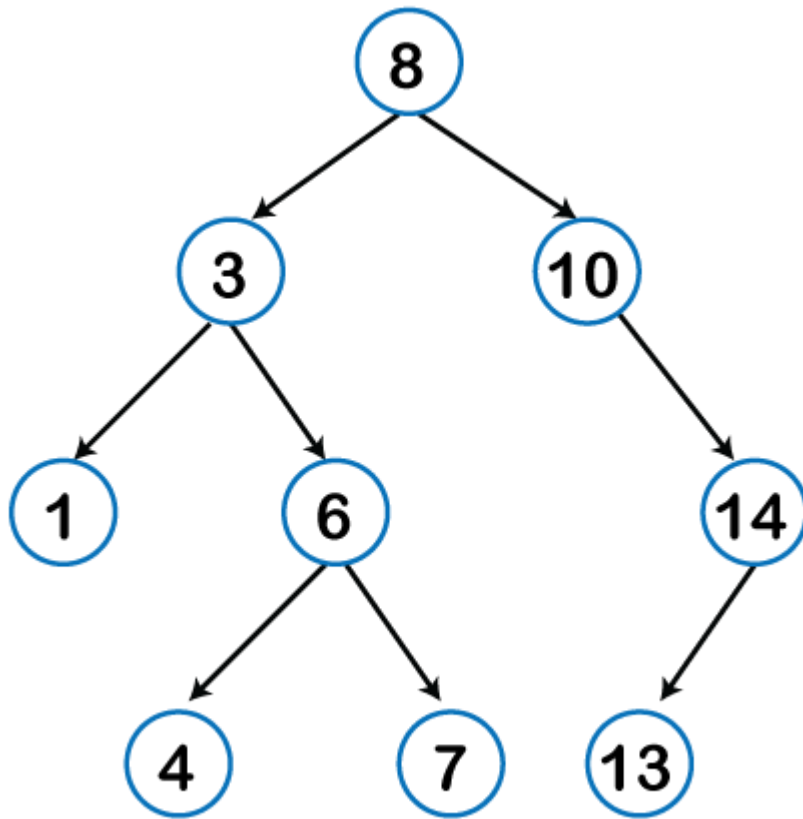


Fig 3.21 Binary Trees

Binary Search tree: Binary search tree is a non-linear data structure in which one node is connected to **n** number of nodes. It is a node-based data structure. A node can be represented in a binary search tree with three fields, i.e., data part, left-child, and right-child. A node can be connected to the utmost two child nodes in a binary search tree, so the node contains two pointers (left child and right child pointer).

Every node in the left subtree must contain a value less than the value of the root node, and the value of each node in the right subtree must be bigger than the value of the root node.

A node can be created with the help of a user-defined data type known as **struct**, as shown below:

struct node

```

{
    int data;

    struct node *left;
    struct node *right;
}

```

The above is the node structure with three fields: data field, the second field is the left pointer of the node type, and the third field is the right pointer of the node type.

○ AVL tree

It is one of the types of the binary tree, or we can say that it is a variant of the binary search tree. AVL tree satisfies the property of the **binary tree** as well as of the **binary search tree**. It is a self-balancing binary search tree that was invented by **Adelson Velsky Lindas**. Here, self-balancing means that balancing the heights of left subtree and right subtree. This balancing is measured in terms of the **balancing factor**.

We can consider a tree as an AVL tree if the tree obeys the binary search tree as well as a balancing factor. The balancing factor can be defined as the **difference between the height of the left subtree and the height of the right subtree**. The balancing factor's value must be either 0, -1, or 1; therefore, each node in the AVL tree should have the value of the balancing factor either as 0, -1, or 1.

To know more about the AVL tree, click on the link given below:

<https://www.javatpoint.com/avl-tree>

○ Red-Black Tree

The red-Black tree is the binary search tree. The prerequisite of the Red-Black tree is that we should know about the binary search tree. In a binary search tree, the value of the left-subtree should be less than the value of that node, and the value of the right-subtree should be greater than the value of that node. As we know that the time complexity of binary search in the average case is $\log_2 n$, the best case is $O(1)$, and the worst case is $O(n)$.

When any operation is performed on the tree, we want our tree to be balanced so that all the operations like searching, insertion, deletion, etc., take less time, and all these operations will have the time complexity of **$\log_2 n$** .

The red-black tree is a self-balancing binary search tree. AVL tree is also a height balancing binary search tree then **why do we require a Red-Black tree**. In the AVL tree, we do not know how many rotations would be required to balance the tree, but in the Red-black tree, a maximum of 2 rotations are required to balance the tree. It contains one extra bit that represents either the red or black color of a node to ensure the balancing of the tree.

- **Splay tree**

The splay tree data structure is also binary search tree in which recently accessed element is placed at the root position of tree by performing some rotation operations. Here, **splaying** means the recently accessed node. It is a **self-balancing** binary search tree having no explicit balance condition like **AVL** tree.

It might be a possibility that height of the splay tree is not balanced, i.e., height of both left and right subtrees may differ, but the operations in splay tree takes order of **logN** time where **n** is the number of nodes.

Splay tree is a balanced tree but it cannot be considered as a height balanced tree because after each operation, rotation is performed which leads to a balanced tree.

- **Treap**

Treap data structure came from the Tree and Heap data structure. So, it comprises the properties of both Tree and Heap data structures. In Binary search tree, each node on the left subtree must be equal or less than the value of the root node and each node on the right subtree must be equal or greater than the value of the root node. In heap data structure, both right and left subtrees contain larger keys than the root; therefore, we can say that the root node contains the lowest value.

In treap data structure, each node has both **key** and **priority** where key is derived from the Binary search tree and priority is derived from the heap data structure.

The **Treap** data structure follows two properties which are given below:

- Right child of a node \geq current node and left child of a node \leq current node (binary tree)
- Children of any subtree must be greater than the node (heap)

- **B-tree**

B-tree is a balanced **m-way** tree where **m** defines the order of the tree. Till now, we read that the node contains only one key but b-tree can have more than one key, and more than 2 children. It always maintains the sorted data. In binary tree, it is possible that leaf nodes can be at different levels, but in b-tree, all the leaf nodes must be at the same level.

If order is m then node has the following properties:

- Each node in a b-tree can have maximum **m** children
- For minimum children, a leaf node has 0 children, root node has minimum 2 children and internal node has minimum ceiling of $m/2$ children. For example, the value of **m** is 5 which means that a node can have 5 children and internal nodes can contain maximum 3 children.

- Each node has maximum $(m-1)$ keys.

The root node must contain minimum 1 key and all other nodes must contain atleast **ceiling of $m/2$ minus 1** keys.

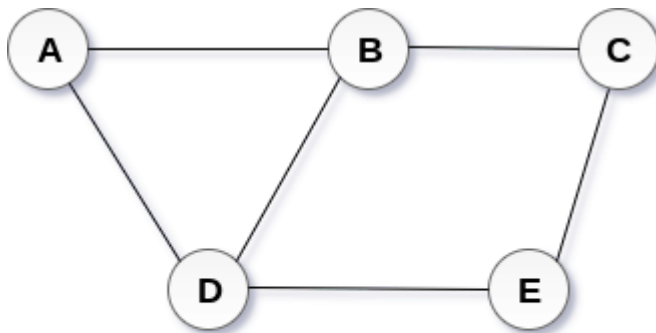
Graph

A graph can be defined as group of vertices and edges that are used to connect these vertices. A graph can be seen as a cyclic tree, where the vertices (Nodes) maintain any complex relationship among them instead of having parent child relationship.

Definition

A graph G can be defined as an ordered set $G(V, E)$ where $V(G)$ represents the set of vertices and $E(G)$ represents the set of edges which are used to connect these vertices.

A Graph $G(V, E)$ with 5 vertices (A, B, C, D, E) and six edges ((A,B), (B,C), (C,E), (E,D), (D,B), (D,A)) is shown in the following figure.



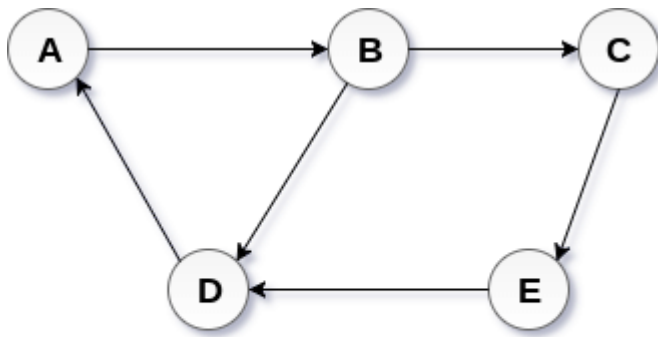
Undirected Graph

Directed and Undirected Graph

A graph can be directed or undirected. However, in an undirected graph, edges are not associated with the directions with them. An undirected graph is shown in the above figure since its edges are not attached with any of the directions. If an edge exists between vertex A and B then the vertices can be traversed from B to A as well as A to B

In a directed graph, edges form an ordered pair. Edges represent a specific path from some vertex A to another vertex B. Node A is called initial node while node B is called terminal node.

A directed graph is shown in the following figure.



Directed Graph

Graph Terminology

Path

A path can be defined as the sequence of nodes that are followed in order to reach some terminal node V from the initial node U .

Closed Path

A path will be called as closed path if the initial node is same as terminal node. A path will be closed path if $V_0 = V_N$.

Simple Path

If all the nodes of the graph are distinct with an exception $V_0 = V_N$, then such path P is called as closed simple path.

Cycle

A cycle can be defined as the path which has no repeated edges or vertices except the first and last vertices.

Connected Graph

A connected graph is the one in which some path exists between every two vertices (u, v) in V . There are no isolated nodes in connected graph.

Complete Graph

A complete graph is the one in which every node is connected with all other nodes. A complete graph contain $n(n-1)/2$ edges where n is the number of nodes in the graph.

Weighted Graph

In a weighted graph, each edge is assigned with some data such as length or weight. The weight of an edge e can be given as $w(e)$ which must be a positive (+) value indicating the cost of traversing the edge.

Digraph

A digraph is a directed graph in which each edge of the graph is associated with some direction and the traversing can be done only in the specified direction.

Loop

An edge that is associated with the similar end points can be called as Loop.

Adjacent Nodes

If two nodes u and v are connected via an edge e , then the nodes u and v are called as neighbours or adjacent nodes.

Degree of the Node

A degree of a node is the number of edges that are connected with that node. A node with degree 0 is called as isolated node.

Graph Representation

By Graph representation, we simply mean the technique which is to be used in order to store some graph into the computer's memory.

There are two ways to store Graph into the computer's memory. In this part of this tutorial, we discuss each one of them in detail.

1. Sequential Representation

In sequential representation, we use adjacency matrix to store the mapping represented by vertices and edges. In adjacency matrix, the rows and columns are represented by the graph vertices. A graph having n vertices, will have a dimension $n \times n$.

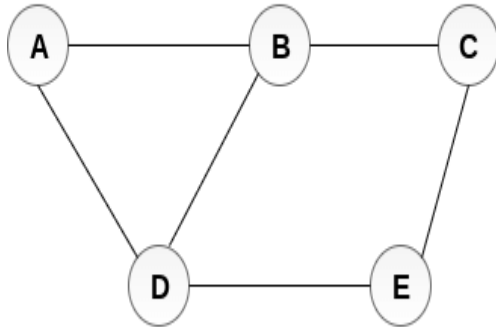
An entry M_{ij} in the adjacency matrix representation of an undirected graph G will be 1 if there exists an edge between V_i and V_j .

10.3M

226

Exception Handling in Java - Javatpoint

An undirected graph and its adjacency matrix representation is shown in the following figure.



Undirected Graph

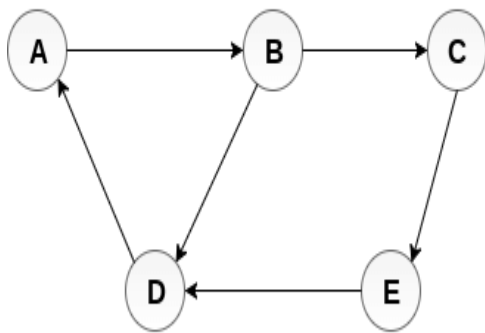
	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	1	0
C	0	1	0	0	1
D	1	1	0	0	1
E	0	0	1	1	0

Adjacency Matrix

in the above figure, we can see the mapping among the vertices (A, B, C, D, E) is represented by using the adjacency matrix which is also shown in the figure.

There exists different adjacency matrices for the directed and undirected graph. In directed graph, an entry A_{ij} will be 1 only when there is an edge directed from V_i to V_j .

A directed graph and its adjacency matrix representation is shown in the following figure.



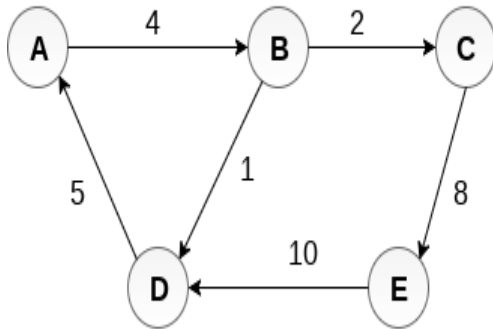
Directed Graph

	A	B	C	D	E
A	0	1	0	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	1	0	0	0	0
E	0	0	0	1	0

Adjacency Matrix

Representation of weighted directed graph is different. Instead of filling the entry by 1, the Non-zero entries of the adjacency matrix are represented by the weight of respective edges.

The weighted directed graph along with the adjacency matrix representation is shown in the following figure.



Weighted Directed Graph

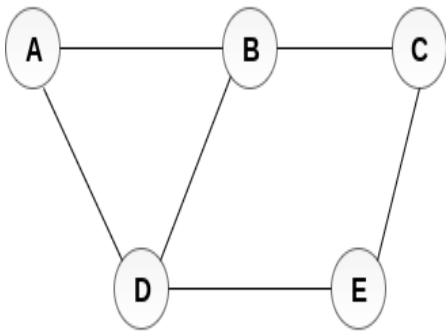
	A	B	C	D	E
A	0	4	0	0	0
B	0	0	2	1	0
C	0	0	0	0	8
D	5	0	0	0	0
E	0	0	0	10	0

Adjacency Matrix

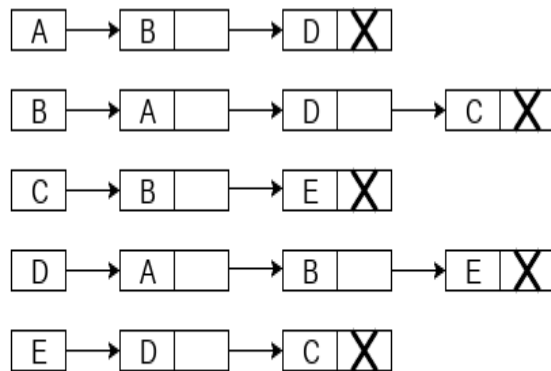
Linked Representation

In the linked representation, an adjacency list is used to store the Graph into the computer's memory.

Consider the undirected graph shown in the following figure and check the adjacency list representation.



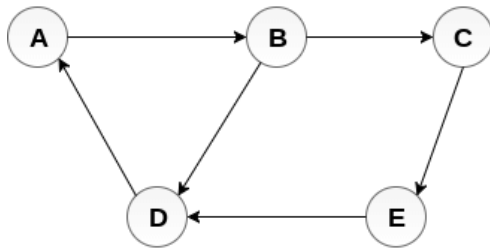
Undirected Graph



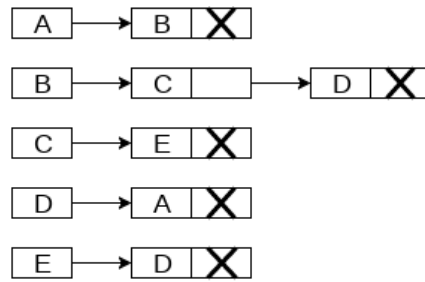
Adjacency List

An adjacency list is maintained for each node present in the graph which stores the node value and a pointer to the next adjacent node to the respective node. If all the adjacent nodes are traversed then store the NULL in the pointer field of last node of the list. The sum of the lengths of adjacency lists is equal to the twice of the number of edges present in an undirected graph.

Consider the directed graph shown in the following figure and check the adjacency list representation of the graph.



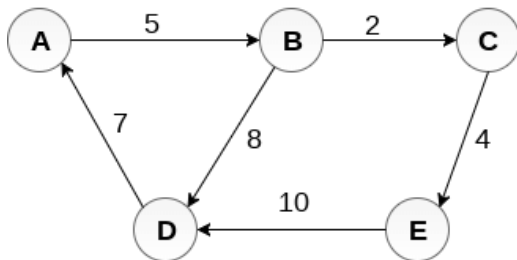
Directed Graph



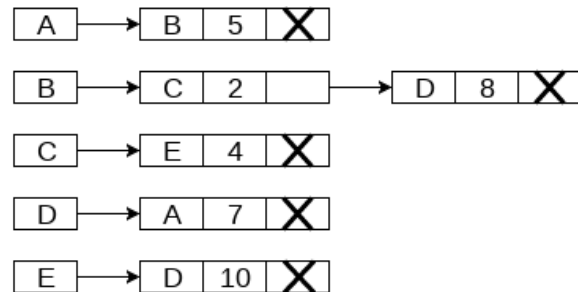
Adjacency List

In a directed graph, the sum of lengths of all the adjacency lists is equal to the number of edges present in the graph.

In the case of weighted directed graph, each node contains an extra field that is called the weight of the node. The adjacency list representation of a directed graph is shown in the following figure.



Weighted Directed Graph



Adjacency List

Data Organizational Models

Optimizing Data Fluency Across a Company

Structuring data science and analytics within a company is really about optimization of talent. Within departments, acquiring data skills directly drives insights and business decisions. But executives who oversee those functions also have the power to organize them for greater efficiency and impact. Let's take a look at the different ways companies can organize their data function.

Organizational Models for Data Science and Analytics

Among our customers and more broadly, we see companies using three main models to organize their data teams: centralized, embedded, and hybrid.

The Centralized Model

The first model is a centralized data science and analytics team that fields requests from other departments, commonly set up as a Center of Excellence. In this model, the Chief Data Scientist or Chief Data Officer oversees the entire data function and is able to prioritize data needs independently of other functions. The centralized data team is essentially the gatekeeper of data.

This model is widely used but can be problematic because it creates a silo for data tools, skills, and responsibility. Employees outside of the data team are not encouraged to develop data fluency for themselves because they view it as someone else's responsibility. It may also mean that the data team is fully at the mercy of other teams' requests, and unable to have full ownership to establish its own strategic vision. Likewise, requestors are at the mercy of resources available on the data team. Either specialists are available or they aren't, and requestors typically do not have visibility into the progress being made. There may also be a lack of communication and mutual understanding along the way, as the data team and the requestors may have different ideas on prioritization and delivery.

The Embedded Model

The second is the embedded, or decentralized, model, where data professionals are embedded in functional teams. You might have a data scientist or analyst on the marketing team in charge of marketing analytics, another on the sales team supporting sales targets, another on the engineering team supporting data infrastructure, and another on the finance team in charge of financial modeling. The data professionals report directly to the heads of each department function and are in the loop with team needs, so they typically have a better understanding of the needs at hand than in the centralized model. But the embedded model can be problematic. Department heads may not be data fluent, meaning they may not be able to provide proper guidance and support for their data scientists. There may also be a lack of analytics standardization across the company resulting in decentralized reporting.

The Hybrid Model

The third is the hybrid model, where there is a central data team and data professionals are also embedded in functional teams. As in the embedded model, they report to department heads, but they also have a dotted line to the Chief Data Scientist at the company. This allows for the Chief Data Scientist to bridge the gap between data needs and functional expertise, acting as both a visionary and a technical lead. The hybrid model encourages more cross-functional collaboration and can enable a strong sense of purpose for each data professional. While it's very powerful, it requires careful organizational

structuring and is generally favored by companies whose data fluency is more mature.

Adapting to Evolving Data Needs

Business leaders should consider employee needs when choosing their data science and analytics organizational model. It's no secret that skilled data experts are in short supply, which means they have a lot of options. For employers, this means employee retention can be a problem and they must ensure their data fluent employees are sufficiently engaged.

Business leaders should also consider their company's current and future needs when choosing their organizational model. An incremental approach would be wise. Early-stage businesses can take a lean approach with a centralized data function supporting the whole company to start. Data scientists and analysts command high salaries (commonly in the six figures) and many early-stage businesses may not be able to afford to staff one on each team. Then, as analytics needs and capabilities scale, the company may pivot to an embedded model. As the company matures, it may choose to adopt a hybrid model to boost operational speed and extend capabilities.

Three Layers

Basically, the main concept is that Data Science should always be seen and faced as a work flow covering three distinct and hierarchically connected layers, which we could define as follows:

1. **Data lake:** the basis of all layers, this is where data is collected, validated and prepared for analysis
2. **Science:** the core of the matter, this is where data scientists do their magic with mathematics, statistics and AI
3. **Usability:** the real value, this is where the business comes into play and things are made usable and sellable

Given these three layers, it is evidently clear that we cannot run our analysis without data, as data is the main topic of data science. So, it is definitely useless thinking about science if a good, solid and well designed data lake is not available.

When we have the data lake we need, we can do our magic with science and creativity as well, testing models and providing data insights. But again, this will be useless by itself, as everything we do as data scientists must be business oriented and business usable.

Then, here we are with the third layer, which is where we will turn every useful scientific results into usable business knowledge. This means using a proper human language, creating dashboards, or clear business speaking reports, or even creating a full data product, that is some kind of software dedicated to business people.

In practice

The law of the three layers brings to practical strategies. These strategies are about how to build up a data science team and how to plan data science activities.

- A data science team should be made up of people covering all of the three layers. This means having onboard developers, data architects and data engineers for the data lake layer, physicists and statisticians for the science layer, business experts and solution architects for the usability layer
- The step zero in any data science project is having data. No data, no science. Moreover data must be healthy and usable
- Always keep in mind the final user. If we are doing data science, then everything we do must be useful to reach a sellable and usable solution
- The Data Science team must be involved in the data design and data manipulation processes. The ICT department should seriously consider the collaboration with Data Science team
- Always remember the existence of the three layers. Missing out the split will always bring to serious problems with the whole project. The three layers require very different, yet complimentary, skills. Trying to mix the three or forgetting one of them is definitely a dangerous mistake
- In project management, define three macro areas according to the three layers. Skills, tools, times, efforts and issues are really different from layer to layer. Then keep in mind the essence of the three layers when managing the project

Centre of Excellence

Data science is transforming businesses and generating new value across industries. IDC predicts worldwide AI investment of \$77.6B in 2022. Gartner predicts AI systems will create \$3.9T in business value in 2022. In recent years, this value has been recognized by leaders of old and new industries alike. As machine learning technologies have matured, they have become a new tool for telecoms, retailers, banks, healthcare providers, insurers, utilities, and many more. Having the ability to employ machine learning is no longer a cutting-edge advantage, but a competitive expectation.

Despite maturing tools, data science still often requires dedicated technology and sophisticated teams to execute successfully. There are applications of data science to improve marketing, sales, operations, customer services, and other departments, but investing in the teams and technology to realize it in each department is costly and duplicative. A center of excellence addresses this problem by creating a dedicated data science capability that delivers value across departments.

What is a Data Science Center of Excellence?

A center of excellence (CoE) is a single team that focuses the vision, strategy, and infrastructure for a discipline. This is particularly useful for data science because of the niche skills and technology the discipline requires. The CoE team typically acts as an internal consultant, working with multiple divisions in the organization to identify and exploit opportunities in data science. A center of excellence may not be the only team wielding data science in an organization, but acts as a leader, innovator, and standards setter. Additionally, a successful CoE team will often serve as a learning resource for individuals practicing data science outside of the team.

What are the benefits of Data Science Centers of Excellence?

- Consolidate expenses to reduce total cost
- Standardize process, tools, and approach to common machine learning problems
- Reduce time to delivery
- Make data science accessible and affordable, even for niche use cases
- Centralize a strategy for talent development and innovation

How do you build a Data Science Center of Excellence?

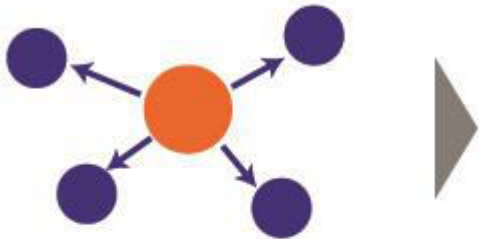
Determining where to start when building a machine learning and data science capability can be daunting. Naturally, one would like to see results as quickly as possible, without committing to major investments in staff and technology all at once. There is a common maturity model that can be used for building data science centers of excellence iteratively, while realizing gains across the organization. In this model, talent is pooled, a centralized consultancy is established, then that team develops standards and practices that spur innovation across the organization. This process often takes years to fully mature, but value is delivered quickly and with the flexibility to address changing opportunities efficiently.

The Data Science Center of Excellence Maturity Model



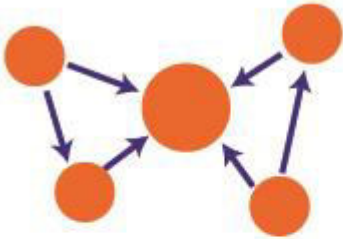
Pockets of Data Science

Knowledge, skills, and technology are scattered across the organization. Collaboration is limited and training is ad hoc.



Centralized Consulting

Infrastructure and skills are centralized, and benefits are rolled out to all areas of the business. Collaboration and training begin.



Integrated Innovation

Capacity development is standardized and distributed. A central team drives success through collaboration, training, and best practices.

Foremost, an executive strategy should be set to formalize where data science can be applied and how much investment should be made. In this strategy there should be the top opportunities (operational gains, improved conversion rates, customer experience enhancement, etc) and an estimate of how valuable they are. This roadmap may change in time, but provides a guide for how much to invest in team and infrastructure.

Next, talent needs to be assembled. Typically, there are already talented, knowledgeable analysts, developers, or data scientists in the organization. This is a huge asset. A great way to start a data science CoE is by seeding it with existing employees who bring to the table a depth of knowledge about the business. This helps keep the data science team connected to business stakeholders and can accelerate development by avoiding ramp-up time. Joining the CoE team can also be a big development opportunity and morale booster for employees who get a chance to build their skills and operate across the business. It is important to invest in these employees and help them sharpen their skills. Additionally, outside talent can be helpful for bringing in expertise in data science tools and practices that will become standards for the CoE. Many successful teams start with a core group of existing employees, assess their skills for development gaps, and then use training and strategic hires to level them up.

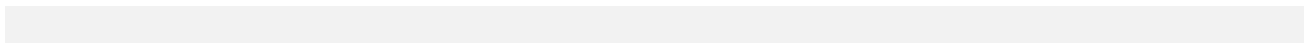
Lastly, a fully functional data science team requires infrastructure to access data, train models, and deploy solutions. There is an ever-expanding list of options. The best choices will depend on the existing infrastructure in the organization, the characteristics of the data, the data science use cases, and other operational factors. Different team members will bring in different experiences and opinions on what is best. As much as possible, it is helpful to establish some key tools and patterns that the team can adopt, while allowing flexibility to test new tools as needed. It is often best to start with the data infrastructure. Accessing and transforming data is one of the most time-consuming and repetitive aspects of data science. A flexible, scalable data store is crucial in the long run.

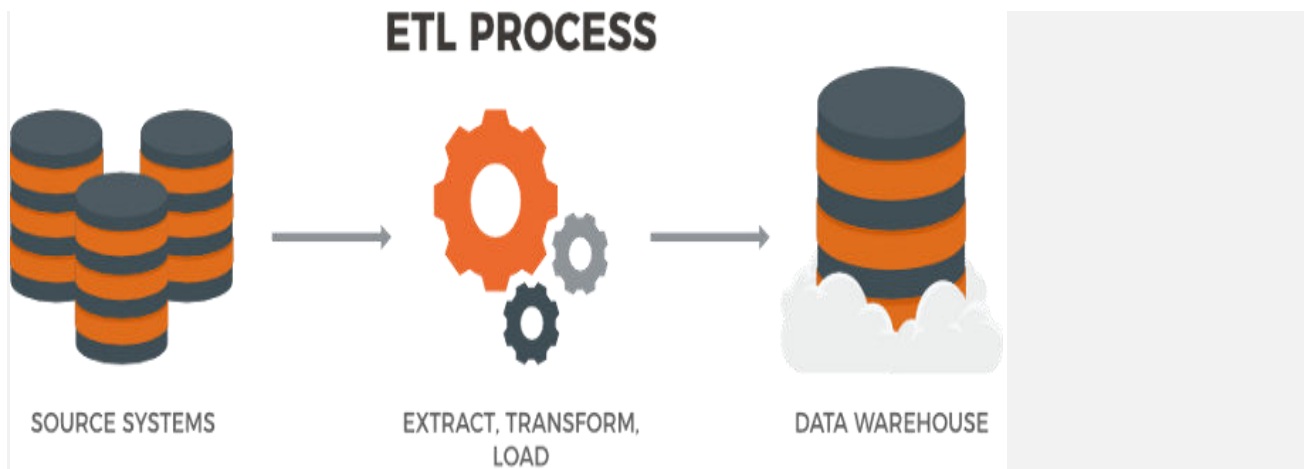
How do you measure success?

A key part of the CoE strategy is maximizing value delivered while increasing efficiency in costs. Data Science is a revenue stream, not a cost center. Data science has the ability to increase revenue, reduce costs, capture market share, reduce churn, and improve experiences.

It is important to track value from the beginning and tune growth accordingly. Although the CoE may exist outside of the departments it serves, it is critical to understand the impact it has on them and prioritize opportunities and investment accordingly. By recording the value of what is delivered, a strong case can be made to continue investment in a CoE that would be difficult to realize in scattered data science teams.

What is ETL ???





ETL Process

ETL stands for Extract, Transform and Load. It's a generic process in which data is firstly acquired, then changed or processed and is finally loaded into data warehouse or databases or other files such as PDF, Excel.

You can extract data from any data sources such as Files, any RDBMS/NoSql Database, Websites or real-time user activity, transform the acquired data and then load the transformed data into a data warehouse for business uses such as reporting or analytics.

Don't confuse ETL with a tool, it's a concept of data movement which can be established with the help of various tools such as Informatica, Tableau, Programming languages, etc.

ETL is a 3 steps process:

- Extracting Data from single or multiple Data Sources
- Transforming Data as per business logic. Transformation is in itself a two steps process- data cleansing and data manipulation.

- Loading transformed data into the target data source or data warehouse.

Why ETL ???

Data Analytics are mostly revolving around ETL. Many of you might already be doing it one way or another by writing different functions/scripts to perform tasks on data and get some useful information out of data.

Key reasons for using ETL are:

- Visualizing your entire data flow pipeline which helps business taking critical business decisions.
- Transactional databases cannot answer complex business questions that can be answered by ETL.
- ETL provides a method of moving the data from various sources into a data warehouse.
- As data sources change, the Data Warehouse will automatically update.
- ETL process can perform complex transformations and requires the extra area to store the data.
- ETL helps to Migrate data into a Data Warehouse. Convert to the various formats and types to adhere to one consistent system.
- ETL is a predefined process for accessing and manipulating source data into the target database.
- ETL offers deep historical context for the business.

In this blog, we will establish our ETL pipeline by using Python programming language, cause thankfully Python comes with lots of different libraries that help to establish tons of Data Analytics, Data Visualization or Data Science solutions.

Let's start with building our own ETL pipeline in python.

For the sake of simplicity our 3 steps are as follow:

- * Extract data from CSV file
- * Transform/Manipulate Data
- * Load Data into SQL Database (Python has inbuilt SQL module — 'sqlite3')

CSV

Data

Source

Link: https://raw.githubusercontent.com/diljeet1994/Python_Tutorials/master/Projects/Advanced%20ETL/crypto-markets.csv

1	slug	TGT	name	date	rankno	open	high	low	close	volume	market	close_r	spread
880	bitcoin	BTC	Bitcoin	28-04-2013	1	135.3	135.98	132.1	134.21	0	1.5E+09	0.5438	3.88
881	bitcoin	BTC	Bitcoin	29-04-2013	1	134.44	147.49	134	144.54	0	1.49E+09	0.7813	13.49
882	bitcoin	BTC	Bitcoin	30-04-2013	1	144	146.93	134.05	139	0	1.6E+09	0.3843	12.88
883	bitcoin	BTC	Bitcoin	01-05-2013	1	139	139.89	107.72	116.99	0	1.54E+09	0.2882	32.17
884	bitcoin	BTC	Bitcoin	02-05-2013	1	116.38	125.6	92.28	105.21	0	1.29E+09	0.3881	33.32
885	bitcoin	BTC	Bitcoin	03-05-2013	1	106.25	108.13	79.1	97.75	0	1.18E+09	0.6424	29.03
886	bitcoin	BTC	Bitcoin	04-05-2013	1	98.1	115	92.5	112.5	0	1.09E+09	0.8889	22.5
887	bitcoin	BTC	Bitcoin	05-05-2013	1	112.9	118.8	107.14	115.91	0	1.25E+09	0.7521	11.66
888	bitcoin	BTC	Bitcoin	06-05-2013	1	115.98	124.66	106.64	112.3	0	1.29E+09	0.3141	18.02
889	bitcoin	BTC	Bitcoin	07-05-2013	1	112.25	113.44	97.7	111.5	0	1.25E+09	0.8767	15.74
890	bitcoin	BTC	Bitcoin	08-05-2013	1	109.6	115.78	109.6	113.57	0	1.22E+09	0.6424	6.18
891	bitcoin	BTC	Bitcoin	09-05-2013	1	113.2	113.46	109.26	112.67	0	1.26E+09	0.8119	4.2
892	bitcoin	BTC	Bitcoin	10-05-2013	1	112.8	122	111.55	117.2	0	1.26E+09	0.5407	10.45
893	bitcoin	BTC	Bitcoin	11-05-2013	1	117.7	118.68	113.01	115.24	0	1.31E+09	0.3933	5.67
894	bitcoin	BTC	Bitcoin	12-05-2013	1	115.64	117.45	113.44	115	0	1.29E+09	0.389	4.01
895	bitcoin	BTC	Bitcoin	13-05-2013	1	114.82	118.7	114.5	117.98	0	1.28E+09	0.8286	4.2

Sample Data

Let's start with python code. I am using Pandas here to read CSV file. Pandas is a great python library that is used a lot for Data Analysis purpose. It's really easy to read data from CSV in pandas.

Code snippet to read CSV file:

```
import pandas as pd

crypto_df = pd.read_csv('crypto-markets.csv')
crypto_df.head()
```

After running code you could see following dataframe as output:

	slug	asset	name	date	ranknow	open	high	low	close	volume	market	close_ratio	spread
0	target-coin	TGT	Target Coin	29-09-2017	607	0.028961	0.054766	0.028961	0.041777	69996	0.0	0.4966	0.03
1	target-coin	TGT	Target Coin	30-09-2017	607	0.041783	0.046196	0.031435	0.031744	5725	0.0	0.0209	0.01
2	target-coin	TGT	Target Coin	01-10-2017	607	0.031761	0.035957	0.021040	0.028385	5012	0.0	0.4924	0.01
3	target-coin	TGT	Target Coin	02-10-2017	607	0.028375	0.054595	0.020417	0.022525	8010	0.0	0.0617	0.03
4	target-coin	TGT	Target Coin	03-10-2017	607	0.022527	0.032225	0.020211	0.020359	1787	0.0	0.0123	0.01

So we have cryptocurrencies data with us, containing crypto token name, its open, close, high and low price on certain dates and some other columns as well. These prices are in USD and we want to save this price into GBP currency (Great Britain Pound). Also, let's assume that some columns are irrelevant for us, so we will drop those columns in the end. Since our data does not contain any Null or blank values and its kind of structure as well so we can skip data cleansing part.

Let's start with transforming the data. Transformation logic is to convert the price of BTC, ETH , XRP and LTC cryptocurrency only into GBP from USD. Let's just assume that we don't care about other currencies.

So in Python, we have to build logic to iterate through all the rows of excel file and check if the current row consists of desired cryptocurrency (BTC, ETH , XRP and LTC). If so, then we will transform its price into GBP by simply multiplying price columns with 0.80, which is the conversion rate ((1 USD = 0.80 GBP)) of USD to GBP at the time of writing this blog.

Code Snippet for Transformation part:

```
import numpy as np
import pandas as pd

assetsCode = ['BTC','ETH','XRP','LTC']

# coverting open, close, high and low price of crypto currencies into GBP values since current price
is in Dollars
# if currency belong to this list ['BTC','ETH','XRP','LTC']
crypto_df['open'] = crypto_df[['open', 'asset']].apply(lambda x: (float(x[0]) * 0.80) if x[1] in
assetsCode else np.nan, axis=1)
crypto_df['close'] = crypto_df[['close', 'asset']].apply(lambda x: (float(x[0]) * 0.80) if x[1] in
assetsCode else np.nan, axis=1)
crypto_df['high'] = crypto_df[['high', 'asset']].apply(lambda x: (float(x[0]) * 0.80) if x[1] in
assetsCode else np.nan, axis=1)
crypto_df['low'] = crypto_df[['low', 'asset']].apply(lambda x: (float(x[0]) * 0.80) if x[1] in
assetsCode else np.nan, axis=1)

# dropping rows with null values by asset column
crypto_df.dropna(inplace=True)

# reset the data frame index
crypto_df.reset_index(drop=True,inplace=True)
crypto_df.head()
```

In code you can see following dataframe as output:

	slug	asset	name	date	ranknow	open	high	low	close	volume	market	close_ratio	spread
0	bitcoin	BTC	Bitcoin	28-04-2013	1	101.475	101.9850	99.0750	100.6575	0	1.500520e+09	0.5438	3.88
1	bitcoin	BTC	Bitcoin	29-04-2013	1	100.830	110.6175	100.5000	108.4050	0	1.491160e+09	0.7813	13.49
2	bitcoin	BTC	Bitcoin	30-04-2013	1	108.000	110.1975	100.5375	104.2500	0	1.597780e+09	0.3843	12.88
3	bitcoin	BTC	Bitcoin	01-05-2013	1	104.250	104.9175	80.7900	87.7425	0	1.542820e+09	0.2882	32.17
4	bitcoin	BTC	Bitcoin	02-05-2013	1	87.285	94.2000	69.2100	78.9075	0	1.292190e+09	0.3881	33.32

There are a lot of columns, let's assume that for us, relevant columns are only asset, name, date, open, high, low and close. So let's drop other irrelevant columns.

```
crypto_df.drop(labels=['slug', 'ranknow', 'volume', 'market', 'close_ratio', 'spread'], inplace=True, axis=1)
crypto_df.head()
```

Final Data set is as below:

	asset	name	date	open	high	low	close
0	BTC	Bitcoin	28-04-2013	101.475	101.9850	99.0750	100.6575
1	BTC	Bitcoin	29-04-2013	100.830	110.6175	100.5000	108.4050
2	BTC	Bitcoin	30-04-2013	108.000	110.1975	100.5375	104.2500
3	BTC	Bitcoin	01-05-2013	104.250	104.9175	80.7900	87.7425
4	BTC	Bitcoin	02-05-2013	87.285	94.2000	69.2100	78.9075

The final stage is to Load the transformed Data in SQL database.

One thing to keep in mind is that loading part could also be in the form of Data Visualizations (Graphs), PDF or Excel report, or database as in our case. It's just retrieval of the final data set for a further business case study.

Python does come along with an in-built SQL module 'sqlite3' for Python3, so we don't need to download any external library.

Code snippet to load our data into SQL:

```

import sqlite3

# connect function opens a connection to the SQLite database file,
conn = sqlite3.connect('session.db')
print(conn)
# Output: <sqlite3.Connection object at 0x0000015A87671730>

# Drop a table name Crypto if it exists already
try:
    conn.execute('DROP TABLE IF EXISTS `Crypto` ')
except Exception as e:
    raise(e)
finally:
    print('Table dropped')

# Create a new Table named as Crypto
try:
    conn.execute("""
        CREATE TABLE Crypto
        (ID      INTEGER PRIMARY KEY,
        NAME     TEXT  NOT NULL,
        Date     datetime,
        Open     Float DEFAULT 0,
        High     Float DEFAULT 0,
        Low      Float DEFAULT 0,
        Close    Float DEFAULT 0);""")
    print ("Table created successfully");
except Exception as e:
    print(str(e))
    print('Table Creation Failed!!!!')
finally:
    conn.close() # this closes the database connection

```

First, I created a sql connection, then check if any database with ‘Crypto’ name exists or not. If it existed then I dropped it (Just for FUN :P). And then created table Crypto with sql query inside conn.execute() method.

For data insertion, we again need to change our data from Pandas Dataframe to Python List of Lists or List of Tuples, because that's the format sqlite module understand for data insertion.

Code:

```
# this will convert pandas dataframe to list of list
crypto_list = crypto_df.values.tolist()

# lets make new connection to Insert crypto data in SQL DB
conn = sqlite3.connect('session.db')

# make a cursor - it will help with querying SQL DB
cur = conn.cursor()

try:
    # will use ? sign to represent each column names inside VALUE().
    cur.executemany("INSERT INTO Crypto(ASSET, NAME, Date, Open, High, Low, Close)
VALUES (?, ?, ?, ?, ?, ?, ?)", crypto_list)
    conn.commit()
    print('Data Inserted Successfully')
except Exception as e:
    print(str(e))
    print('Data Insertion Failed')
finally:
    # finally block will help with always closing the connection to DB even in case of error.
    conn.close()

# Output: Data Inserted Successfully
```

UNIT – IV

Data Foundation – SCSA1309

UNIT 4 DATA ANALYSIS & VISUALIZATION

Spreadsheets: Data Manipulations- Sort, filter, remove duplicates-text and math functions-pivot table-lookup functions-Data visualizations for quantitative and qualitative data- charts-Excel Modelling- forecast models using advanced lookup and data validation tools.

Tableau: Creating Visualizations in Tableau-Data hierarchies, filters, groups, sets, calculated fields-Map based visualizations-Build interactive dashboards-Data Stories.

I. SPREADSHEETS - DATA MANIPULATIONS

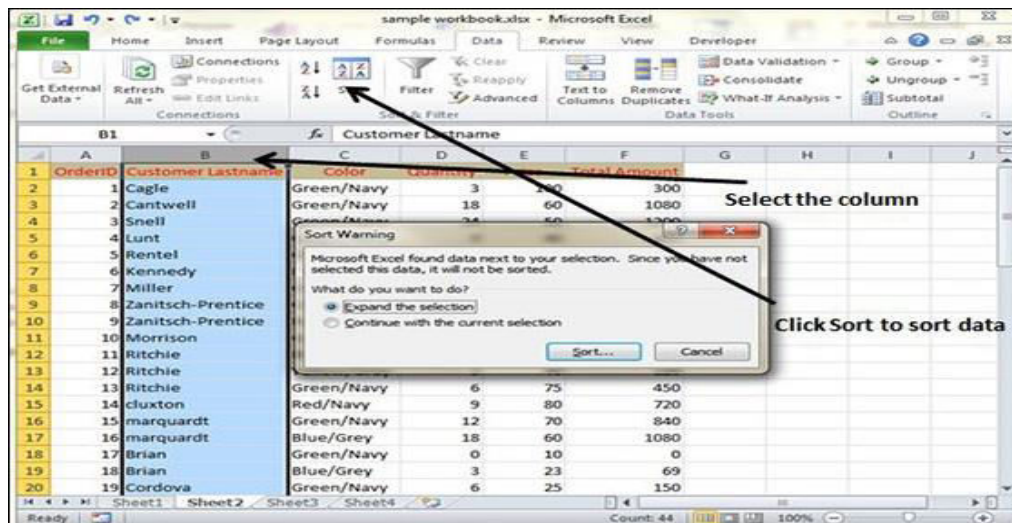
Microsoft Excel is a spreadsheet tool capable of performing calculations, analyzing data and integrating information from different programs.

Sorting in MS Excel

Sorting data in MS Excel rearranges the rows based on the contents of a particular column. You may want to sort a table to put names in alphabetical order. Or, maybe you want to sort data by Amount from smallest to largest or largest to smallest.

To Sort the data follow the steps mentioned below.

- Select the Column by which you want to sort data.
- Choose Data Tab » Sort Below dialog appears.



If you want to sort data based on a selected column, Choose **Continue with the selection** or if you want sorting based on other columns, choose **Expand Selection**.

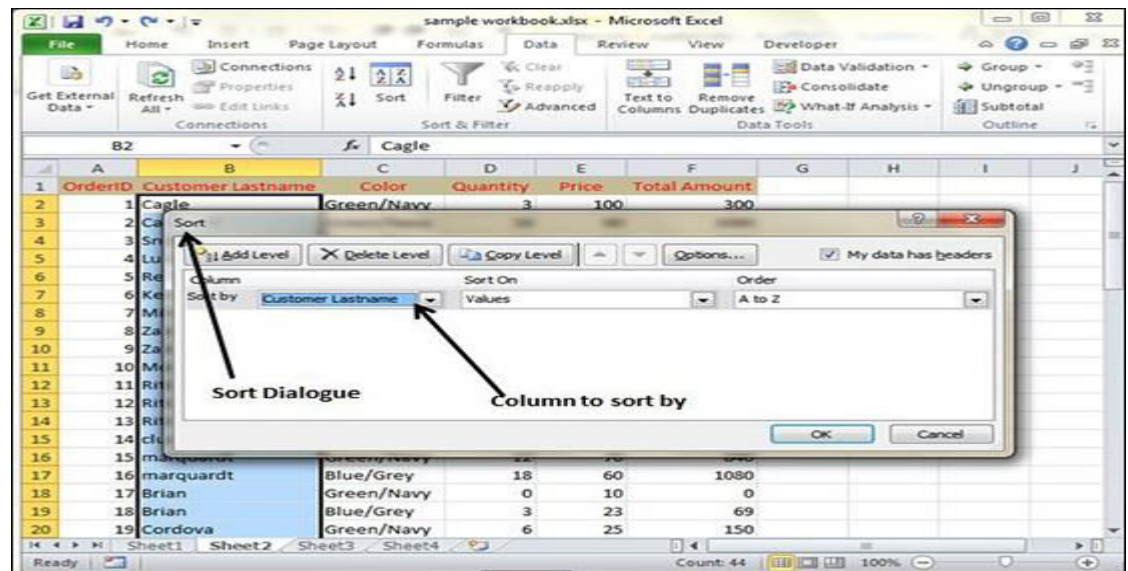
You can Sort based on the below Conditions.

Values – Alphabetically or numerically.

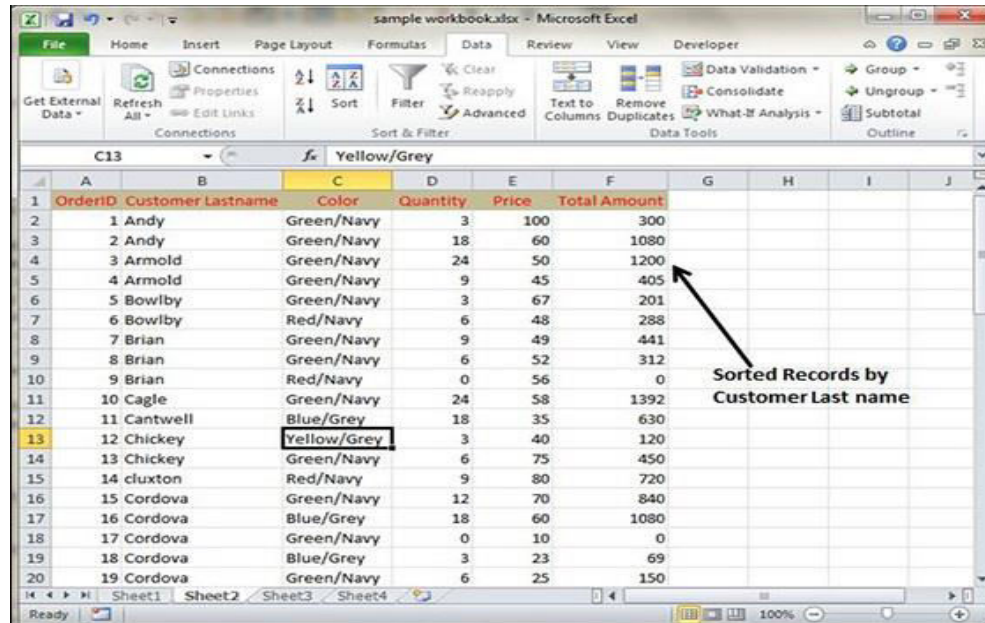
Cell Color – Based on Color of Cell.

Font Color – Based on Font color.

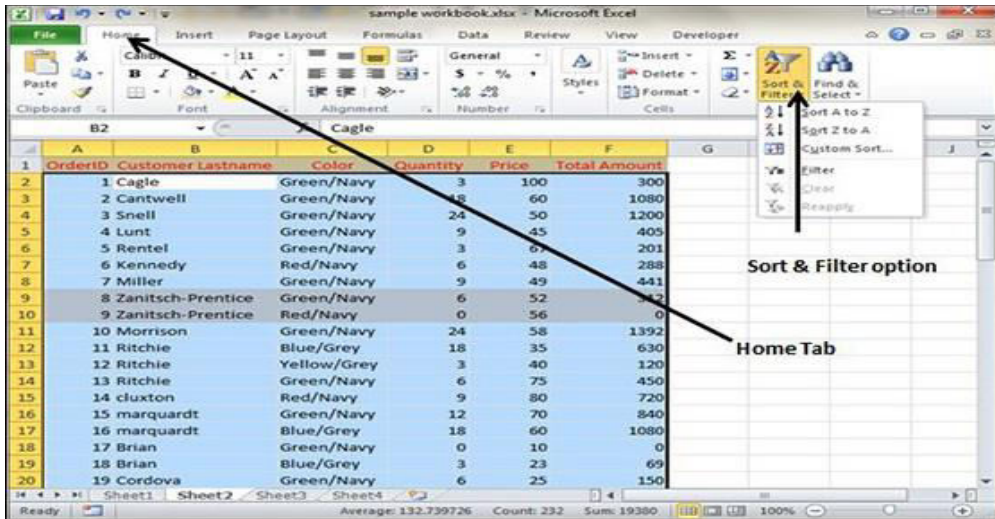
Cell Icon – Based on Cell Icon.



- Clicking Ok will sort the data.



Sorting option is also available from the Home Tab. Choose Home Tab » Sort & Filter. You can see the same dialog to sort records.



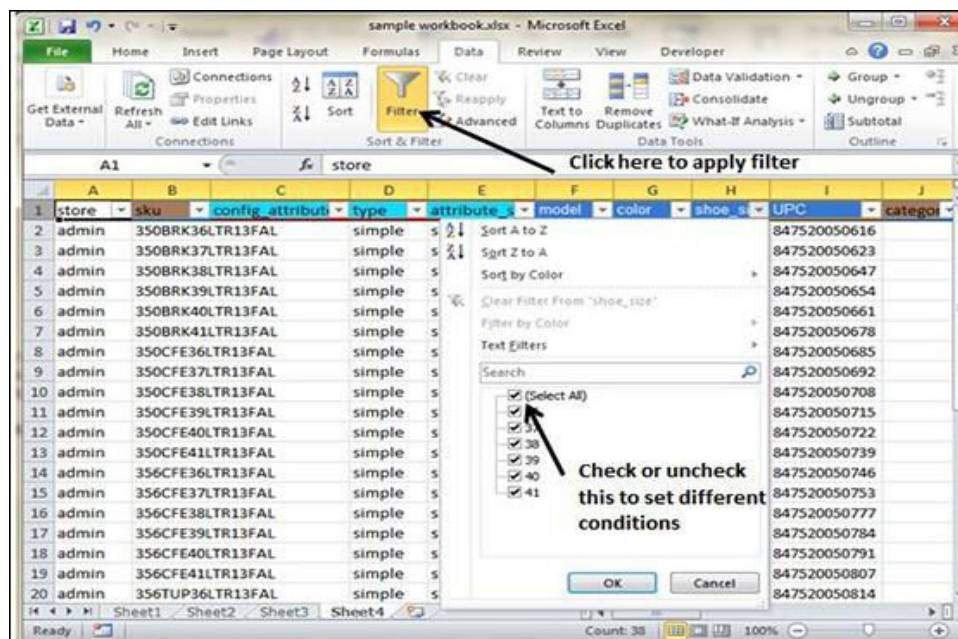
Filters in MS Excel

Filtering data in MS Excel refers to displaying only the rows that meet certain conditions. (The other rows get hidden.)

Using the store data, if you are interested in seeing data where Shoe Size is 36, then you can set filter to do this. Follow the below mentioned steps to do this.

Place a cursor on the Header Row.

Choose **Data Tab » Filter** to set filter.

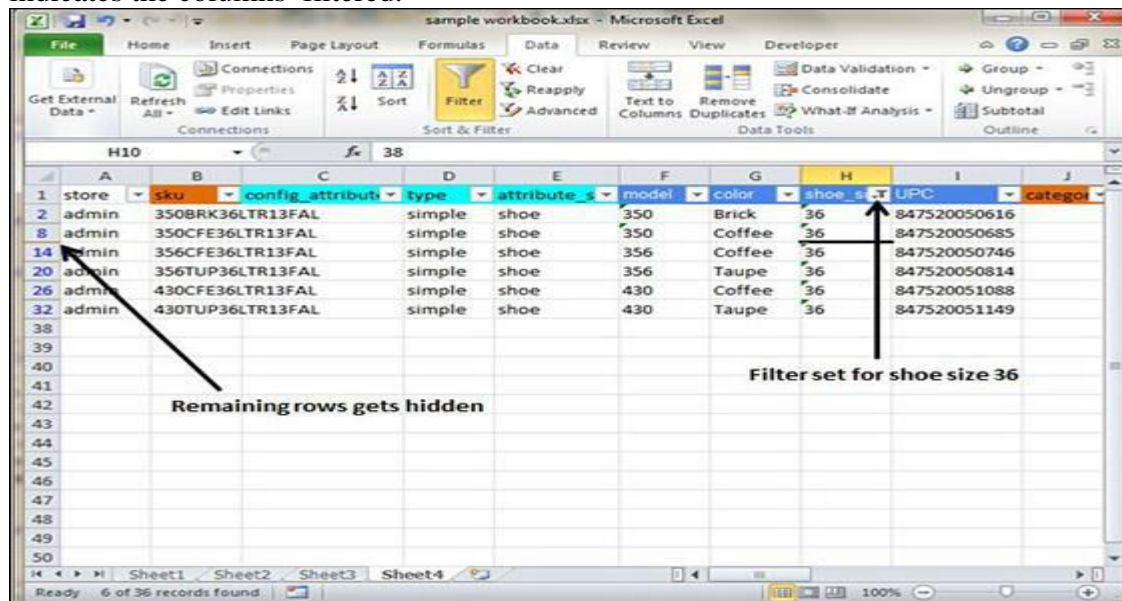


Click the drop-down arrow in the Area Row Header and remove the check mark from Select All, which unselects everything.

Then select the check mark for Size 36 which will filter the data and displays data of Shoe Size 36.

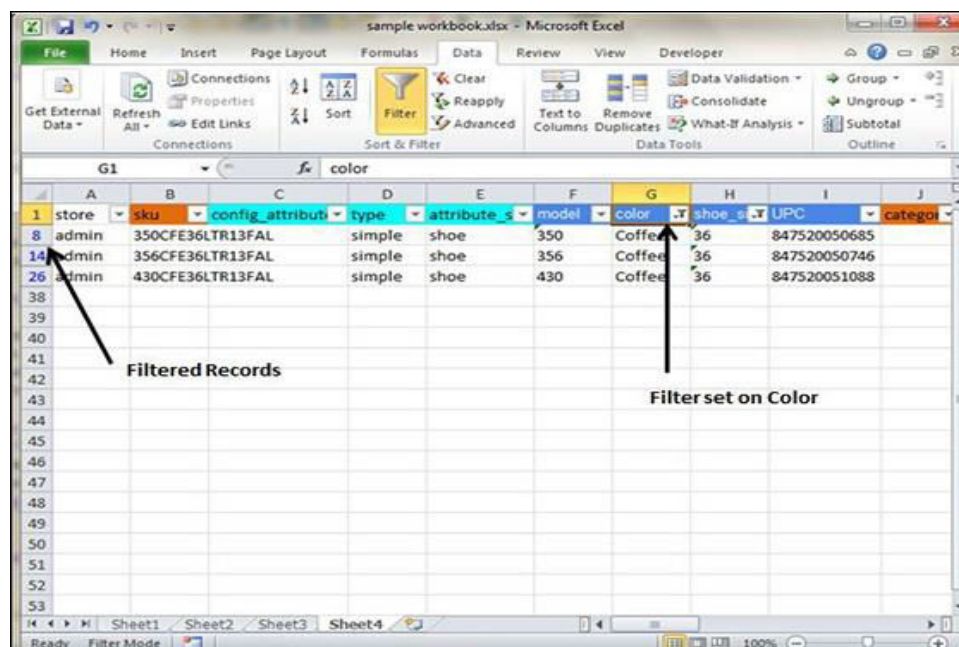
Some of the row numbers are missing; these rows contain the filtered (hidden) data.

There is drop-down arrow in the Area column now shows a different graphic — an icon that indicates the columns filtered.



Using Multiple Filters

You can filter the records by multiple conditions i.e. by multiple column values. Suppose after size 36 is filtered, you need to have the filter where color is equal to Coffee. After setting filter for Shoe Size, choose Color column and then set filter for color.



Remove duplicate values

- Select the range of cells that has duplicate values you want to remove. Tip: Remove any outlines or subtotals from your data before trying to remove duplicates.
- Click Data > Remove Duplicates, and then Under Columns, check or uncheck the columns where you want to remove the duplicates. ...
- Click OK.

Math Functions:

A very important feature in Excel is the formula. It is used to calculate values based on what is in cells, perform operations on a cell content, fetch values after an operation based on your search criteria and much more.

Mathematical Formulas in Excel are used to perform various arithmetic operations like sum, average, count, max, min etc. Here is a list of most frequently used mathematical formulas in excel.

SUM():

This function is used to adds all the values within a cell range.

Syntax:

```
sum(cell address : cell address)
```

Example: sum(C1:C3)=15

Here in the example below, we will create a basic function to calculate the sum of working hours generates in a day.

D13							
	A	B	C	D	E	F	G
1	Name	Sex	Production Quantity	Working Hours	Wages / Hr. (\$)	Total Salary (\$)	
2	Tom Davis	M	10	5	5.80	29	
3	Alex Hunk	M	9	3	6.30	18.9	
4	Peter Duke	M	11	4	4.90	19.6	
5	Jemes Lang	F	9	4	5.00	20	
6	Tim Burner	M	10	4	8.00	32	
7	Ladia Casina	F	10	5	6.50	32.5	
8	Billi Mouth	M	12	4	4.50	18	
9	Milli Rex	F	11	4	7.00	28	
10	Denes Kit	M	14	3	6.00	18	
11	Mario Silli	F	12	5	6.75	33.75	
12							
13							
14							

Select the cell where you want to put the formula, type the equals sign (=) and write the desired function name or choose the function from the suggested function list. Here in the example below we write the SUM function.

CEILING....									
	A	B	C	D	E	F	G	H	I
1	Name	Sex	Production Quantity	Working Hours	Wages / Hr. (\$)	Total Salary (\$)			
2	Tom Davis	M	10	5	5.80	29			
3	Alex Hunk	M	9	3	6.30	18.9			
4	Peter Duke	M	11	4	4.90	19.6			
5	Jemes Lang	F	9	4	5.00	20			
6	Tim Burner	M	10	4	8.00	32			
7	Ladia Casina	F	10	5	6.50	32.5			
8	Billi Mouth	M	12	4	4.50	18			
9	Milli Rex	F	11	4	7.00	28			
10	Denes Kit	M	14	3	6.00	18			
11	Mario Silli	F	12	5	6.75	33.75			
12									
13									
14									
15									
16									
17									
18									
19									
20									

Now write the range of sum or you can select the range by using the mouse to drag.

D13		:			=SUM(D2:D11	
	A	B	C	D	E	F
	Name	Sex	Production Quantity	Working Hours	Wages / Hr. (\$)	Total Salary (\$)
1						
2	Tom Davis	M	10	5	5.80	29
3	Alex Hunk	M	9	3	6.30	18.9
4	Peter Duke	M	11	4	4.90	19.6
5	Jemes Lang	F	9	4	5.00	20
6	Tim Burner	M	10	4	8.00	32
7	Ladia Casina	F	10	5	6.50	32.5
8	Billi Mouth	M	12	4	4.50	18
9	Milli Rex	F	11	4	7.00	28
10	Denes Kit	M	14	3	6.00	18
11	Mario Silli	F	12	5	6.75	33.75
12						
13	Total Workir			=SUM(D2:D11		
14				SUM(number1, [number2], ...)		
15						

Now press Enter key to see the result or press Ctrl+Enter key to stay in the formula cell. Here is the picture below.

D13							=SUM(D2:D11)
	A	B	C	D	E	F	
1	Name	Sex	Production Quantity	Working Hours	Wages / Hr. (\$)	Total Salary (\$)	
2	Tom Davis	M	10	5	5.80	29	
3	Alex Hunk	M	9	3	6.30	18.9	
4	Peter Duke	M	11	4	4.90	19.6	
5	Jemes Lang	F	9	4	5.00	20	
6	Tim Burner	M	10	4	8.00	32	
7	Ladia Casina	F	10	5	6.50	32.5	
8	Billi Mouth	M	12	4	4.50	18	
9	Milli Rex	F	11	4	7.00	28	
10	Denes Kit	M	14	3	6.00	18	
11	Mario Silli	F	12	5	6.75	33.75	
12							
13		Total Working Hours:		41			
14							

You can use the sum() function in other ways. Here is the syntax.

sum(number1,number2,number3....)

Example: sum(4,5,6)=15

AVERAGE							=SUM(4,5,6)
	A	B	C	D			
1		4					
2		5					
3		6					
4							
5		M(4,5,6)					
6							
7							
8							
9							

SUMIF():

Here in the example below, we will create a basic function to calculate the sum of working hours generates in a day only for female employees.

E14						
	A	B	C	D	E	F
	Name	Sex	Production Quantity	Working Hours	Wages / Hr. (\$)	Total Salary (\$)
1						
2	Tom Davis	M	10	5	5.80	29
3	Alex Hunk	M	9	3	6.30	18.9
4	Peter Duke	M	11	4	4.90	19.6
5	Jemes Lang	F	9	4	5.00	20
6	Tim Burner	M	10	4	8.00	32
7	Ladia Casina	F	10	5	6.50	32.5
8	Billi Mouth	M	12	4	4.50	18
9	Milli Rex	F	11	4	7.00	28
10	Denes Kit	M	14	3	6.00	18
11	Mario Silli	F	12	5	6.75	33.75
12						
13	Total Working Hours:			41		
14	Total working hours of Female Employees:					
15						

Syntax:

SUMIF(range,criteria)

Type the equals sign and write the desired function in the cell E14. Here is the picture below.

D2						
	A	B	C	D	E	F
	Name	Sex	Production Quantity	Working Hours	Wages / Hr. (\$)	Total Salary (\$)
1						
2	Tom Davis	M	10	5	5.80	29
3	Alex Hunk	M	9	3	6.30	18.9
4	Peter Duke	M	11	4	4.90	19.6
5	Jemes Lang	F	9	4	5.00	20
6	Tim Burner	M	10	4	8.00	32
7	Ladia Casina	F	10	5	6.50	32.5
8	Billi Mouth	M	12	4	4.50	18
9	Milli Rex	F	11	4	7.00	28
10	Denes Kit	M	14	3	6.00	18
11	Mario Silli	F	12	5	6.75	33.75
12						
13	Total Working Hours:			41		
14	Total working hours of			=SUMIF(B2:B11,"=F",D2:D11)		
15						
16						

Press Enter to see the result and move the cell pointer to below cell or press Ctrl+Enter to stay on the cell.

E14				=SUMIF(B2:B11,"=F",D2:D11)		
	A	B	C	D	E	F
1	Name	Sex	Production Quantity	Working Hours	Wages / Hr. (\$)	Total Salary (\$)
2	Tom Davis	M	10	5	5.80	29
3	Alex Hunk	M	9	3	6.30	18.9
4	Peter Duke	M	11	4	4.90	19.6
5	Jemes Lang	F	9	4	5.00	20
6	Tim Burner	M	10	4	8.00	32
7	Ladia Casina	F	10	5	6.50	32.5
8	Billi Mouth	M	12	4	4.50	18
9	Milli Rex	F	11	4	7.00	28
10	Denes Kit	M	14	3	6.00	18
11	Mario Silli	F	12	5	6.75	33.75
12						
13	Total Working Hours:			41		
14	Total working hours of Female Employees:				18	
15						

AVERAGE():

Here in the example below, we will create a basic function to calculate the average working hours of each employee.

CEILING....						=AVERAGE(D2:D11)	
	A	B	C	D	E	F	
	Name	Sex	Production Quantity	Working Hours	Wages / Hr. (\$)	Total Salary (\$)	
1							
2	Tom Davis	M	10	5	5.80	29	
3	Alex Hunk	M	9	3	6.30	18.9	
4	Peter Duke	M	11	4	4.90	19.6	
5	Jemes Lang	F	9	4	5.00	20	
6	Tim Burner	M	10	4	8.00	32	
7	Ladia Casina	F	10	5	6.50	32.5	
8	Billi Mouth	M	12	4	4.50	18	
9	Milli Rex	F	11	4	7.00	28	
10	Denes Kit	M	14	3	6.00	18	
11	Mario Silli	F	12	5	6.75	33.75	
12							
13	Total Working Hours:			41			
14	Total working hours of Female Employees:				18		
15	Average workii			=AVERAGE(D2:D11)			
16				AVERAGE(number1, [number2], ...)			
17							

Press Enter key and see the result.

D15							=AVERAGE(D2:D11)
	A	B	C	D	E	F	
1	Name	Sex	Production Quantity	Working Hours	Wages / Hr. (\$)	Total Salary (\$)	
2	Tom Davis	M	10	5	5.80	29	
3	Alex Hunk	M	9	3	6.30	18.9	
4	Peter Duke	M	11	4	4.90	19.6	
5	Jemes Lang	F	9	4	5.00	20	
6	Tim Burner	M	10	4	8.00	32	
7	Ladia Casina	F	10	5	6.50	32.5	
8	Billi Mouth	M	12	4	4.50	18	
9	Milli Rex	F	11	4	7.00	28	
10	Denes Kit	M	14	3	6.00	18	
11	Mario Silli	F	12	5	6.75	33.75	
12							
13	Total Working Hours:			41			
14	Total working hours of Female Employees:				18		
15	Average working hours:			4.1			

You can use the AVERAGEIF() and AVERAGEIFS() function in a similar way as SUMIF() function, to average cells based on one or multiple criteria.

COUNT()

Here in the example below, we will create a basic function to calculate the number of employees.

COUNTIF							=COUNT(C2:C11)
	A	B	C	D	E	F	
1	Name	Sex	Production Quantity	Working Hours	Wages / Hr. (\$)	Total Salary (\$)	
2	Tom Davis	M	10	5	5.80	29	
3	Alex Hunk	M	9	3	6.30	18.9	
4	Peter Duke	M	11	4	4.90	19.6	
5	Jemes Lang	F	9	4	5.00	20	
6	Tim Burner	M	10	4	8.00	32	
7	Ladia Casina	F	10	5	6.50	32.5	
8	Billi Mouth	M	12	4	4.50	18	
9	Milli Rex	F	11	4	7.00	28	
10	Denes Kit	M	14	3	6.00	18	
11	Mario Silli	F	12	5	6.75	33.75	
12							
13	Total Working Hours:			41			
14	Total working hours of Female Employees:				18		
15	Average working hours:			4.1			
16	Total numbers of Em			=COUNT(C2:C11)			
17				COUNT(value1, [value2], ...)			

ROUND():

The round function is used to round a number to a specified number of digits.

Syntax:

ROUND(number, number_of_digits)

B7		✕ ✓ f_x		=ROUND(B6,0)		
	A	B	C	D	E	F
1		15.25				
2		10.1				
3		6.3				
4		12.25				
5						
6	Total	43.9				
7	Round	44				
8						
9						

B7		✕ ✓ f_x		=ROUND(B6,2)		
	A	B	C	D		
1		15.2512				
2		10.1234				
3		6.3233				
4		12.2515				
5						
6	Total	43.9494				
7	Rounded upto two decimal	43.95				
8						
9						

RAND():

This function is used to returns a random number greater than or equal to 0 and less than 1.

Syntax:

RAND()

B2		✕ ✓ f_x		=RAND()		
	A	B	C	D	E	F
1						
2		0.368569				
3						
4						
5						
6						
7						
8						
9						

MOD()

This function is used to find the remainder after dividing a number by another number.

Syntax:

MOD(number,divisor)

B3	:	X	✓	<i>f_x</i>	=MOD(B1,B2)	
	A	B	C	D	E	F
1	Number	8				
2	Divisor	3				
3	Mod	2				
4						
5						
6						
7						
8						
9						

INT():

This function is used to convert a decimal number to integer lower than it.

Syntax:

INT(decimal number)

B4	:	X	✓	<i>f_x</i>	=INT(5.7589)	
	A	B	C	D	E	F
1	Decimal	5.256				
2	Integer	5				
3	Decimal	5.7589				
4	Integer	5				
5						
6						
7						
8						
9						

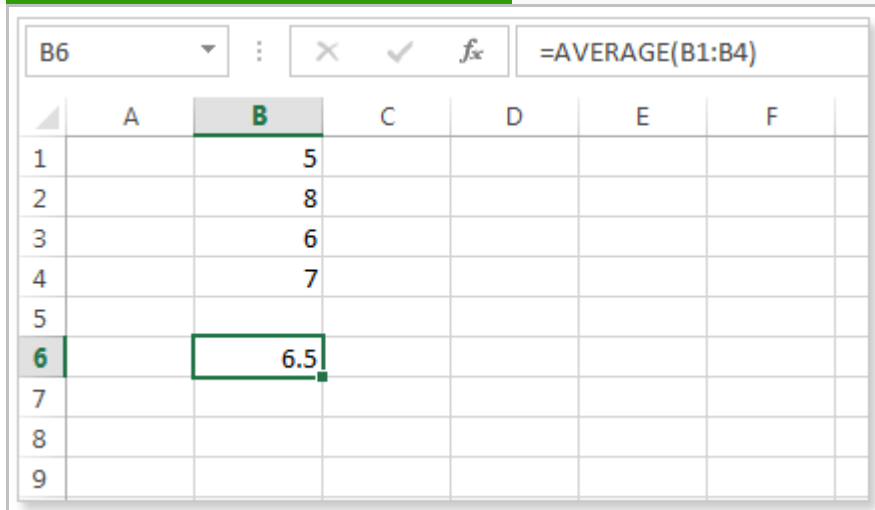
AVERAGE():

Formula:

This function is used to calculate the average of a range of cells.

Syntax:

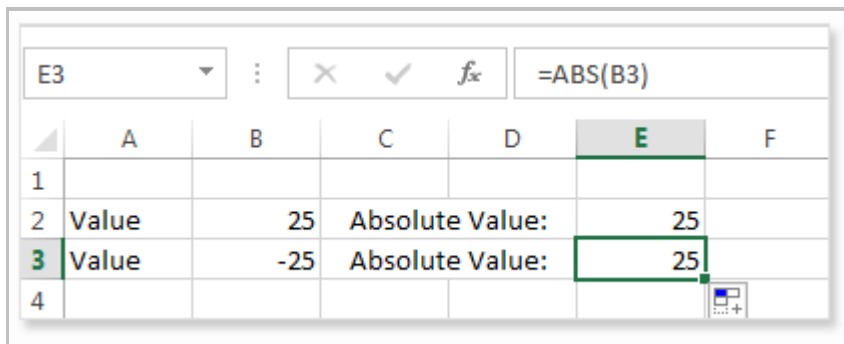
AVERAGE(number1,number2,.....)



	A	B	C	D	E	F
1		5				
2		8				
3		6				
4		7				
5						
6		6.5				
7						
8						
9						

ABS():

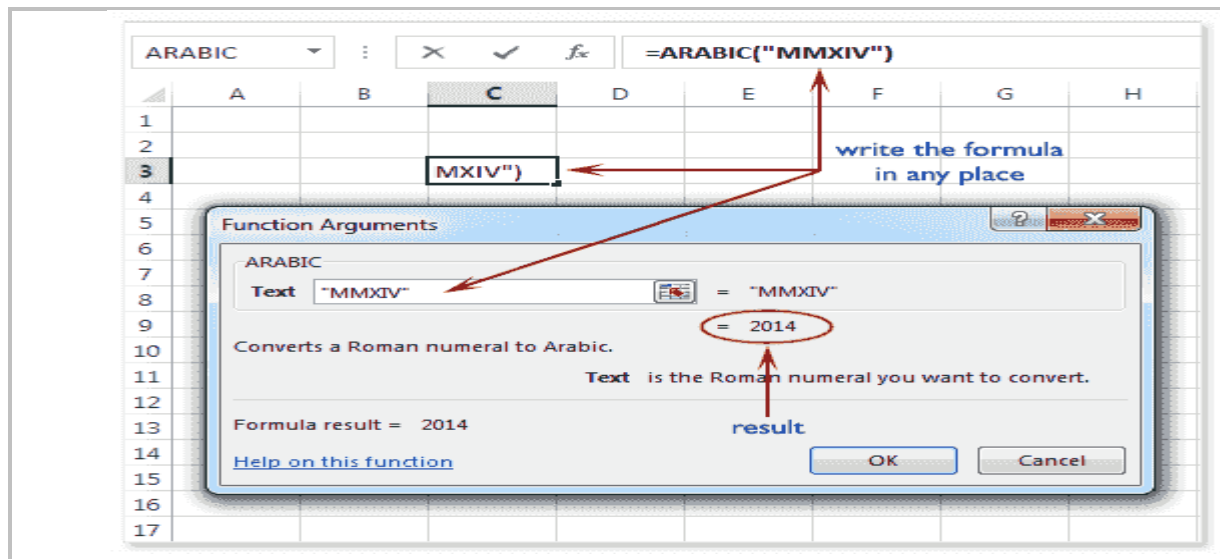
The abs() function is used to return the absolute value of a given number. The number may be positive or negative. Here is the example below.



	A	B	C	D	E	F
1						
2	Value	25	Absolute Value:		25	
3	Value	-25	Absolute Value:		25	
4						

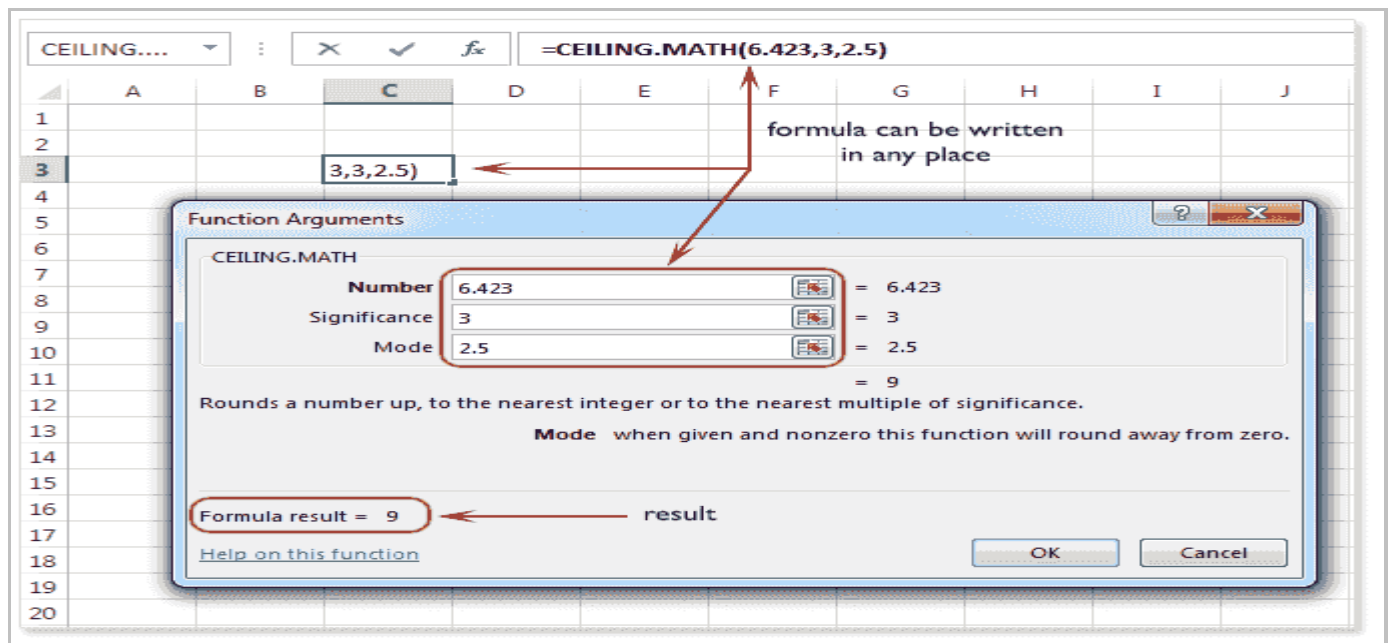
ARABIC():

This function is used to convert roman numeral to arabic. This function accepts roman numeral as an argument. The picture below shows that you can write the formula in any cell or you can use the function wizard or you can select any cell and write the formula in the formula bar and press Ctrl+enter to stay the cell or press enter see the result.



CEILING.MATH():

This function is used to rounds a number upto the nearest integer or to the nearest multiple significance. This function accepts three arguments, these are number, significance and mode. Number is a number, significance is the multiple to which you want to round and mode is also a number. Here in the example below the number is 6.423 and the significance is 3 and the nearest multiple of 3 of the given number is 9 and the mode is a nonzero, so this function starts rounding away from zero.



Text Functions:

- ✓ Len () Len function in Excel helps you to know the length of a string that is number of characters in a string. ...
- ✓ Mid () Mid function in Excel is used to extract the characters from the middle of a string. ...
- ✓ Find () ...
- ✓ Proper () ...
- ✓ Rept () ...
- ✓ Trim() ...
- ✓ Upper() ...
- ✓ Substitute ()

Create a Simple Calculated Field

Version: 2021.3

Applies to: Tableau Desktop, Tableau Online, Tableau Public, Tableau Server

Sometimes your data source does not contain a field (or column) that you need for your analysis. For example, your data source might contain fields with values for Sales and Profit, but not for Profit Ratio. If this is the case, you can create a calculated field for Profit Ratio using data from the Sales and Profit fields.

This topic demonstrates how to create a simple calculated field using an example.

Step 1: Create the calculated field

In a worksheet in Tableau, select **Analysis > Create Calculated Field**.

In the Calculation Editor that opens, give the calculated field a name.

In this example, the calculated field is called Profit Ratio.

Step 2: Enter a formula

In the Calculation Editor, enter a formula.

This example uses the following formula:

```
SUM([Profit])/SUM([Sales])
```

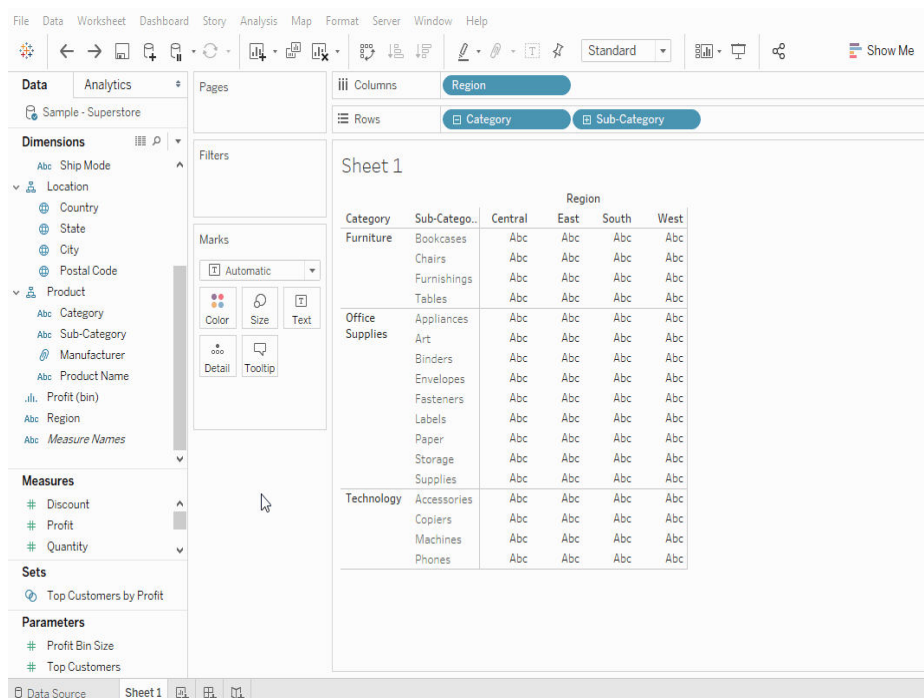
Formulas use a combination of functions, fields, and operators. To learn more about creating formulas in Tableau, see [Formatting Calculations in Tableau](#)(Link opens in a new window) and [Functions in Tableau](#)(Link opens in a new window).

When finished, click **OK**.

The new calculated field is added to the Data pane. If the new field computes quantitative data, it is added to Measures. If it computes qualitative data, it is added to Dimensions.

You are now ready to use the calculated field in the view.

Check your work! Watch how to create a simple calculated field in action:



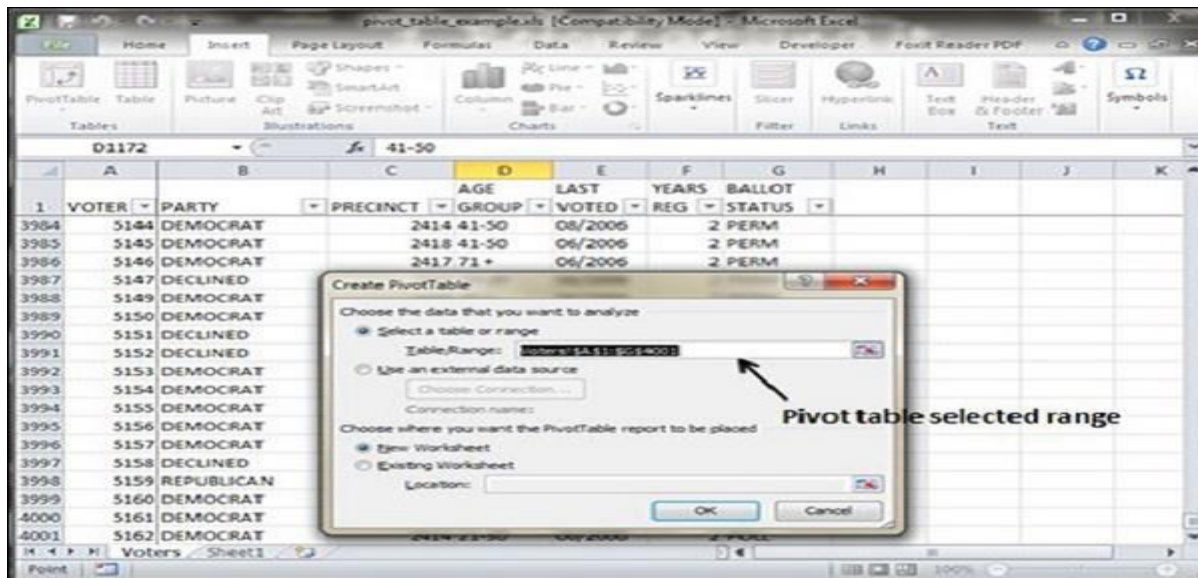
Pivot Tables

A pivot table is essentially a dynamic summary report generated from a database. The database can reside in a worksheet (in the form of a table) or in an external data file. A pivot table can help transform endless rows and columns of numbers into a meaningful presentation of the data. Pivot tables are very powerful tool for summarized analysis of the data.

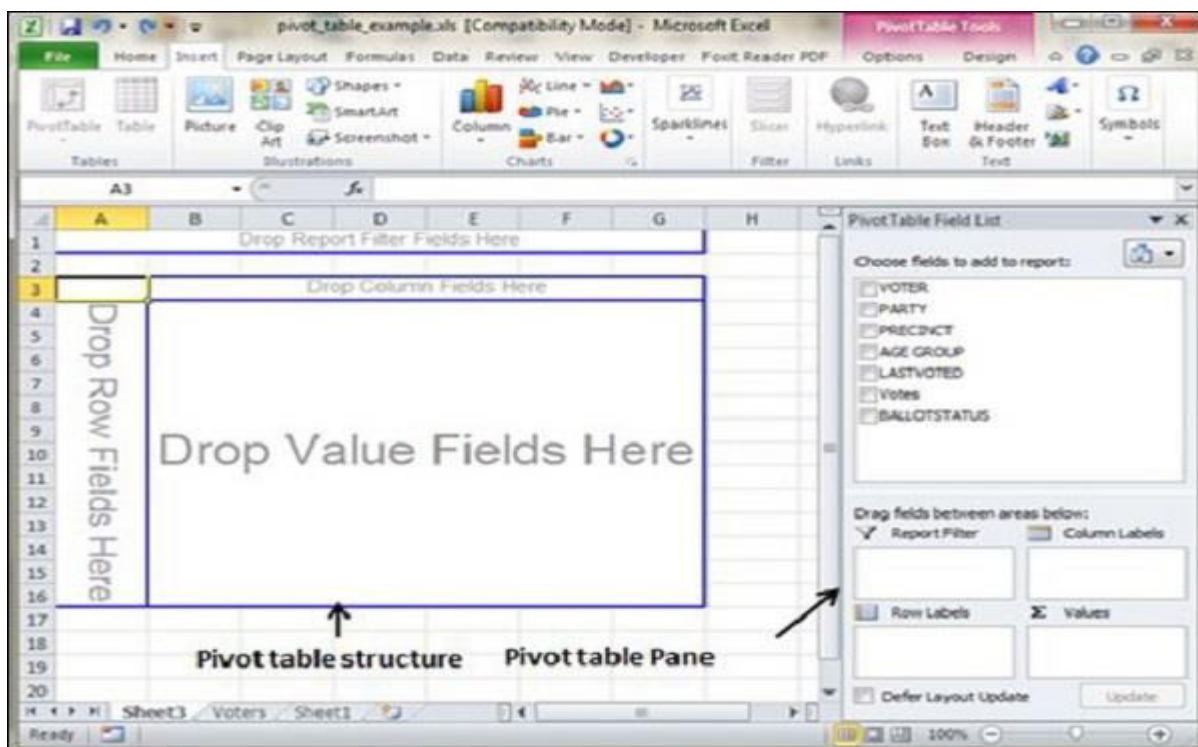
Pivot tables are available under **Insert tab » PivotTable dropdown » PivotTable**.

Pivot Table Example

Now, let us see Pivot table with the help of example. Suppose you have huge data of voters and you want to see the summarized data of voter Information per party, then you can use the Pivot table for it. Choose **Insert tab » Pivot Table** to insert pivot table. MS Excel selects the data of the table. You can select the pivot table location as existing sheet or new sheet.



This will generate the Pivot table pane as shown below. You have various options available in the Pivot table pane. You can select fields for the generated pivot table.



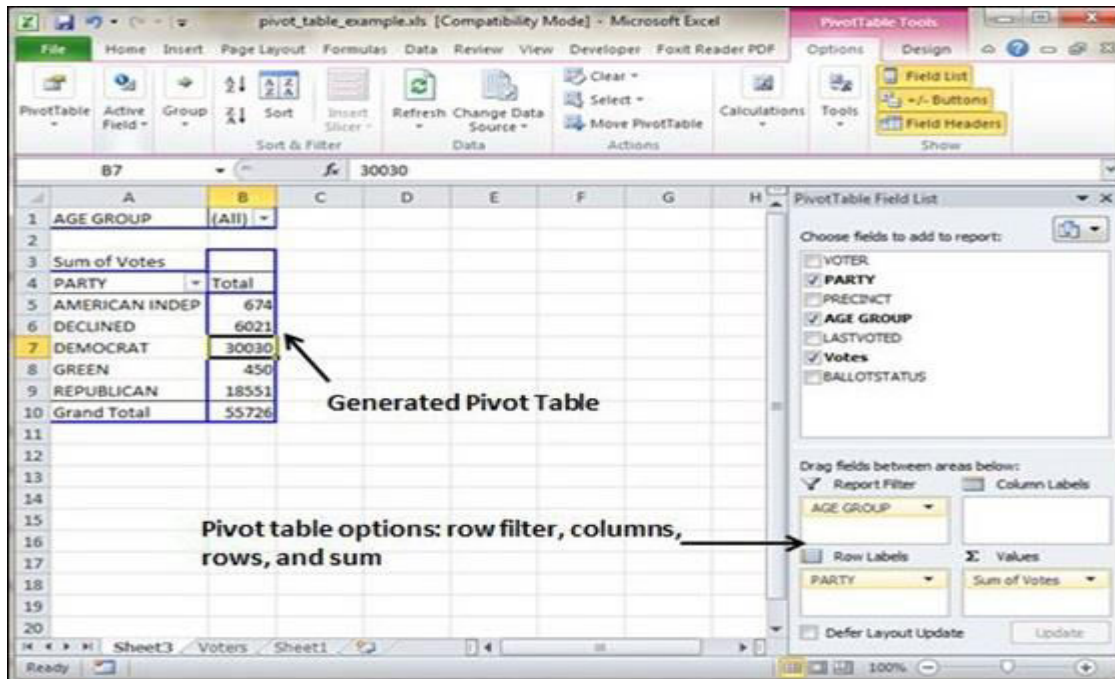
Column labels – A field that has a column orientation in the pivot table. Each item in the field occupies a column.

Report Filter – You can set the filter for the report as year, then data gets filtered as per the year.

Row labels – A field that has a row orientation in the pivot table. Each item in the field occupies a row.

Values area – The cells in a pivot table that contain the summary data. Excel offers several ways to summarize the data (sum, average, count, and so on).

After giving input fields to the pivot table, it generates the pivot table with the data as shown below.

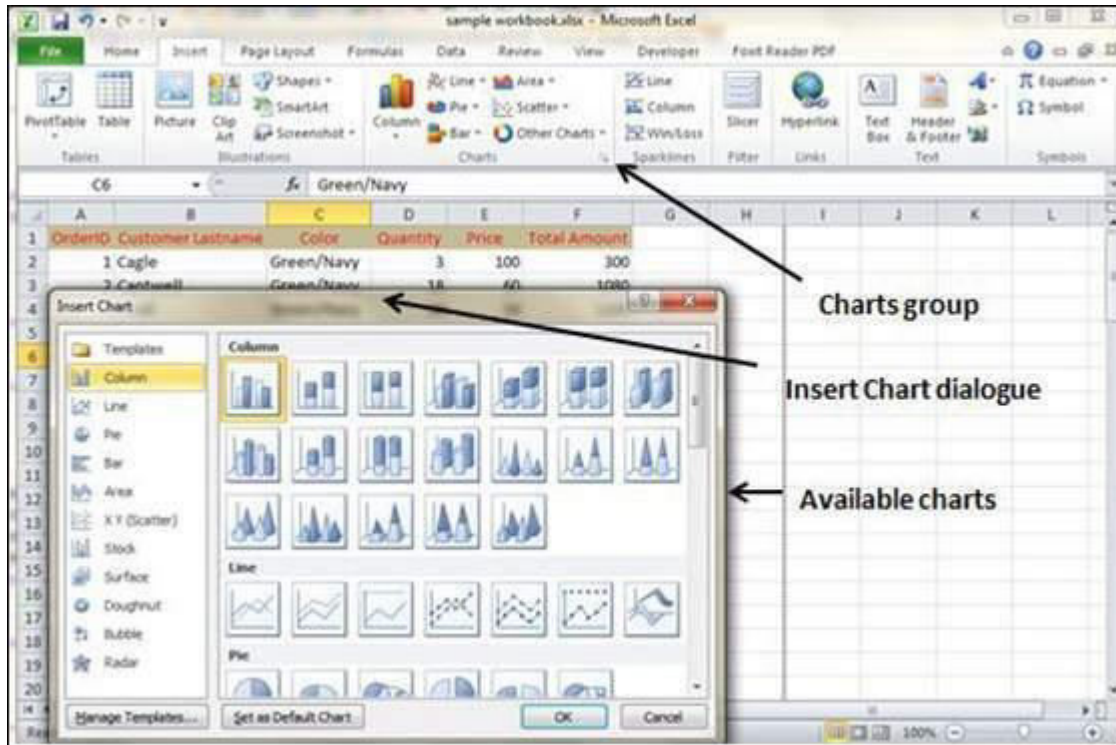


Charts

A chart is a visual representation of numeric values. Charts (also known as graphs) have been an integral part of spreadsheets. Charts generated by early spreadsheet products were quite crude, but they have improved significantly over the years. Excel provides you with the tools to create a wide variety of highly customizable charts. Displaying data in a well-conceived chart can make your numbers more understandable. Because a chart presents a picture, charts are particularly useful for summarizing a series of numbers and their interrelationships.

Types of Charts

There are various chart types available in MS Excel as shown in the below screen-shot.



Column – Column chart shows data changes over a period of time or illustrates comparisons among items.

Bar – A bar chart illustrates comparisons among individual items.

Pie – A pie chart shows the size of items that make up a data series, proportional to the sum of the items. It always shows only one data series and is useful when you want to emphasize a significant element in the data.

Line – A line chart shows trends in data at equal intervals.

Area – An area chart emphasizes the magnitude of change over time.

X Y Scatter – An xy (scatter) chart shows the relationships among the numeric values in several data series, or plots two groups of numbers as one series of xy coordinates.

Stock – This chart type is most often used for stock price data, but can also be used for scientific data (for example, to indicate temperature changes).

Surface – A surface chart is useful when you want to find the optimum combinations between two sets of data. As in a topographic map, colors and patterns indicate areas that are in the same range of values.

Doughnut – Like a pie chart, a doughnut chart shows the relationship of parts to a whole; however, it can contain more than one data series.

Bubble – Data that is arranged in columns on a worksheet, so that x values are listed in the first column and corresponding y values and bubble size values are listed in adjacent columns, can be plotted in a bubble chart.

Radar – A radar chart compares the aggregate values of a number of data series.

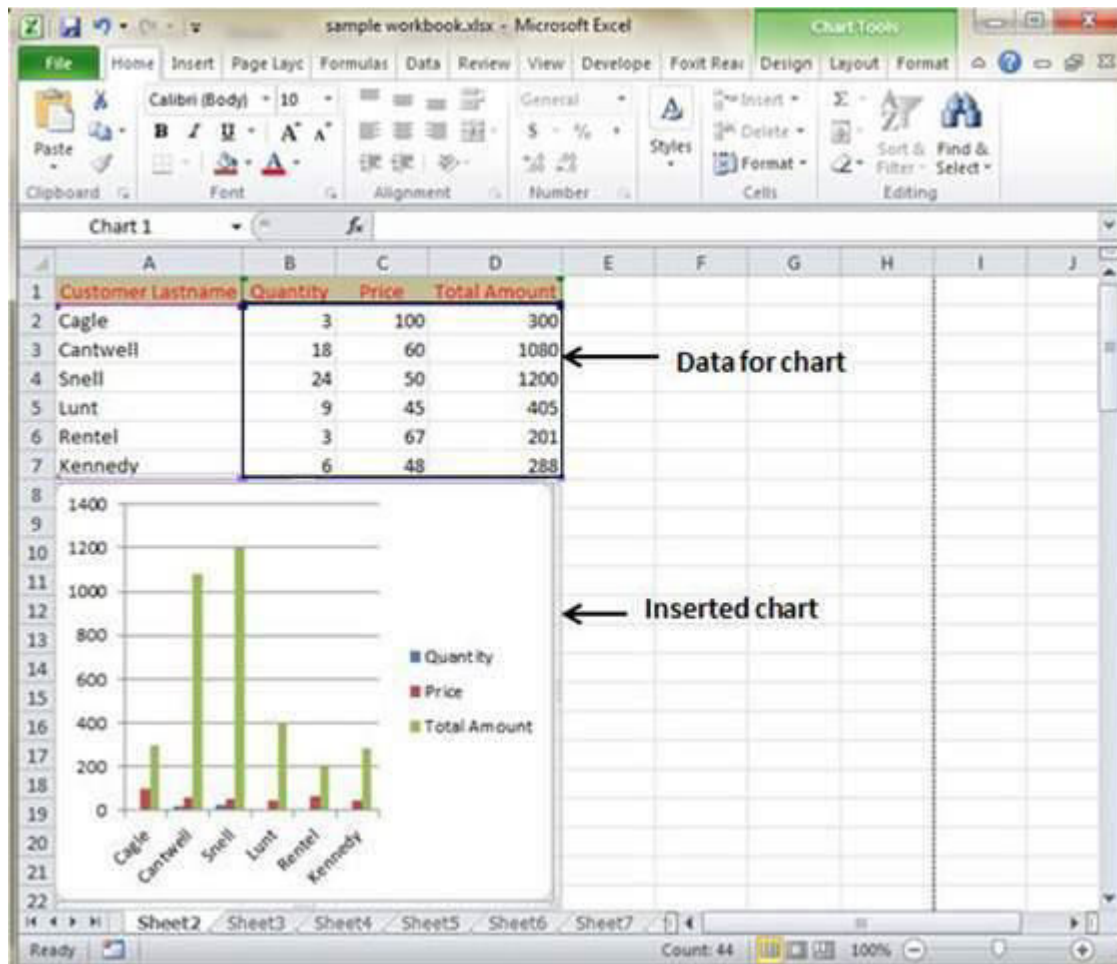
Creating Chart

To create charts for the data by below mentioned steps.

Select the data for which you want to create the chart.

Choose **Insert Tab » Select the chart or click on the Chart group** to see various chart types.

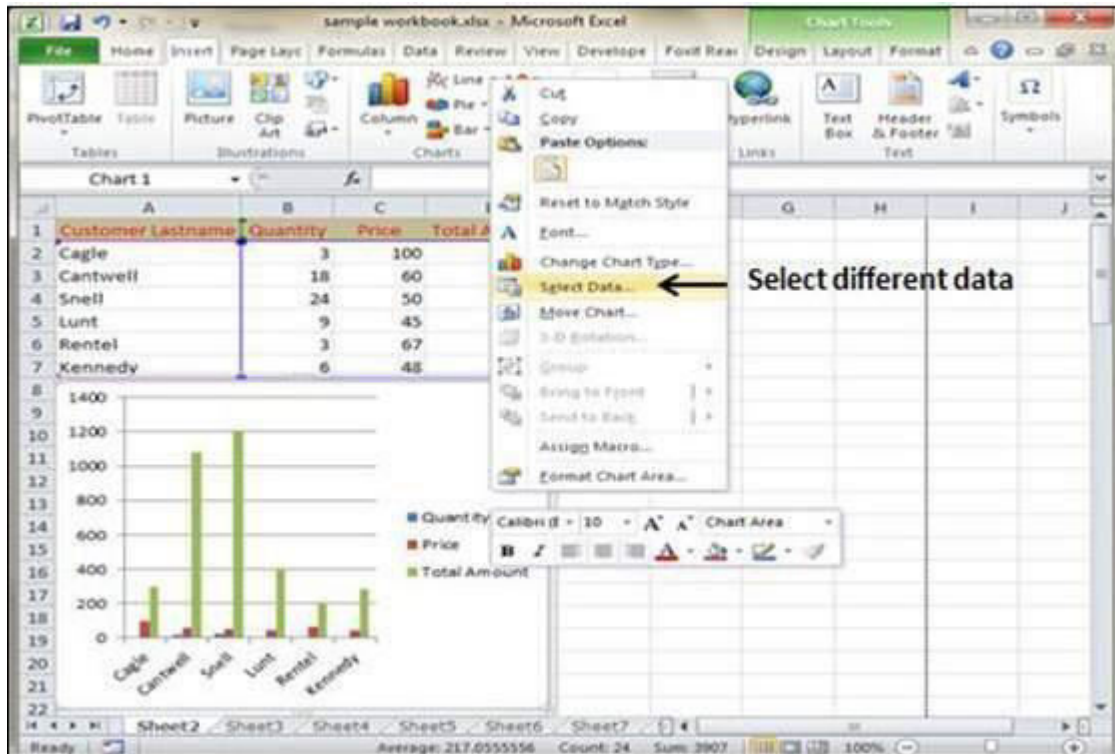
Select the chart of your choice and click OK to generate the chart.



Editing Chart

You can edit the chart at any time after you have created it.

You can select the different data for chart input with **Right click on chart » Select data**. Selecting new data will generate the chart as per the new data, as shown in the below screen-shot.



You can change the X axis of the chart by giving different inputs to X-axis of chart.

You can change the Y axis of chart by giving different inputs to Y-axis of chart.

Loop Functions:

Excel functions to –

- Find values in a range of data - VLOOKUP and HLOOKUP
- Obtain a value or the reference to a value from within a table or range - INDEX
- Obtain the relative position of a specified item in a range of cells - MATCH

You can also combine these functions to get the required results based on the inputs you have.

Using VLOOKUP Function

The syntax of the VLOOKUP function is

VLOOKUP (lookup_value, table_array, col_index_num, [range_lookup])

Where

lookup_value – is the value you want to look up. Lookup_value can be a value or a reference to a cell. Lookup_value must be in the first column of the range of cells you specify in table_array

table_array – is the range of cells in which the VLOOKUP will search for the lookup_value and the return value. table_array must contain

the lookup_value in the first column, and

the return value you want to find

Note – The first column containing the lookup_value can either be sorted in ascending order or not. However, the result will be based on the order of this column.

col_index_num – is the column number in the table_array that contains the return value. The numbers start with 1 for the left-most column of table_array

range_lookup – is an optional logical value that specifies whether you want VLOOKUP to find an exact match or an approximate match. range_lookup can be

omitted, in which case it is assumed to be TRUE and VLOOKUP tries to find an approximate match

TRUE, in which case VLOOKUP tries to find an approximate match. In other words, if an exact match is not found, the next largest value that is less than lookup_value is returned

FALSE, in which case VLOOKUP tries to find an exact match

1, in which case it is assumed to be TRUE and VLOOKUP tries to find an approximate match

0, in which case it is assumed to be FALSE and VLOOKUP tries to find an exact match

Note – If range_lookup is omitted or TRUE or 1, VLOOKUP works correctly only when the first column in table_array is sorted in ascending order. Otherwise, it may result in incorrect values. In such a case, use FALSE for range_lookup.

Using VLOOKUP Function with range_lookup TRUE

Consider a list of student marks. You can obtain the corresponding grades with VLOOKUP from an array containing the marks intervals and pass category.

table_array –

Student Grades	
Marks	Pass Category
0	Fail
35	Third Class
50	Second Class
60	First Class
75	First Class with Distinction

Note that the first column marks based on which the grades are obtained is sorted in ascending order. Hence, using TRUE for range_lookup argument you can get approximate match that is what is required.

Name this array as **Grades**.

It is a good practice to name arrays in this way so that you need not remember the cell ranges. Now, you are ready to look up the grade for the list of marks you have as follows –

	A	B	C	D	E	F
1						
2		Exam Results			Student Grades	
3		Marks	Pass Category		Marks	Pass Category
4		85	=VLOOKUP(B4,Grades,2,TRUE)		0	Fail
5		75	=VLOOKUP(B5,Grades,2,TRUE)		35	Third Class
6		72	=VLOOKUP(B6,Grades,2,TRUE)		50	Second Class
7		55	=VLOOKUP(B7,Grades,2,TRUE)		60	First Class
8		68	=VLOOKUP(B8,Grades,2,TRUE)		75	First Class with Distinction
9		34	=VLOOKUP(B9,Grades,2,TRUE)			
10		60	=VLOOKUP(B10,Grades,2,TRUE)			
11		50	=VLOOKUP(B11,Grades,2,TRUE)			
12		98	=VLOOKUP(B12,Grades,2,TRUE)			
13		59	=VLOOKUP(B13,Grades,2,TRUE)			
14		74	=VLOOKUP(B14,Grades,2,TRUE)			
15		99	=VLOOKUP(B15,Grades,2,TRUE)			
16		40	=VLOOKUP(B16,Grades,2,TRUE)			
17		35	=VLOOKUP(B17,Grades,2,TRUE)			

table_array

← VLOOKUP Function

As you can observe,

col_index_num – indicates the column of the return value in table_array is 2

the **range_lookup** is TRUE

The first column containing the lookup value in the table_array grades is in ascending order. Hence, the results will be correct.

You can get the return value for approximate matches also. i.e. VLOOKUP computes as follows –

Marks	Pass Category
< 35	Fail
>= 35 and < 50	Third Class
>= 50 and < 60	Second Class
>=60 and < 75	First Class
>= 75	First Class with Distinction

You will get the following results –

	A	B	C	D	E	F
1						
2		Exam Results			Student Grades	
3		Marks	Pass Category		Marks	Pass Category
4		85	First Class with Distinction		0	Fail
5		75	First Class with Distinction		35	Third Class
6		72	First Class		50	Second Class
7		55	Second Class		60	First Class
8		68	First Class		75	First Class with Distinction
9		34	Fail			
10		60	First Class			
11		50	Second Class			
12		98	First Class with Distinction			
13		59	Second Class			
14		74	First Class			
15		99	First Class with Distinction			
16		40	Third Class			
17		35	Third Class			

Using VLOOKUP Function with range_lookup FALSE

Consider a list of products containing the Product ID and price for each of the products. The product ID and price will be added to the end of the list whenever a new product is launched. This would mean that the product IDs need not be in ascending order. The product list might be as shown below –

table_array –

Product ID	Product	Price
FC0002	Floor Cleaner	191.90
HW0007	Hand Wash	179.65
AP0024	Air Purifier	254.28
DP0026	Detergent Powder	182.63
ISO0073	Soap	85.85

Name this array as ProductInfo.

You can obtain the price of a product given the product ID with the VLOOKUP function as the product ID is in the first column. The price is in column 3 and hence col_index_num should be 3.

- Use VLOOKUP Function with range_lookup as TRUE

- Use VLOOKUP Function with range_lookup as FALSE

	A	B	C	D
1				
2		Product ID	Product	Price
3		FC0002	Floor Cleaner	191.9
4		HW0007	Hand Wash	179.65
5		AP0024	Air Purifier	254.28
6		DP0026	Detergent Powder	182.63
7		ISO0073	Soap	85.85
8				
9		Product ID	Price	
10		HW0007	=VLOOKUP(B10,ProductInfo,3,TRUE) ← VLOOKUP with TRUE	
11			=VLOOKUP(B10,ProductInfo,3,FALSE) ← VLOOKUP with FALSE	

The correct answer is from the ProductInfo array is 171.65. You can check the results.

	A	B	C	D
1				
2		Product ID	Product	Price
3		FC0002	Floor Cleaner	191.90
4		HW0007	Hand Wash	179.65
5		AP0024	Air Purifier	254.28
6		DP0026	Detergent Powder	182.63
7		ISO0073	Soap	85.85
8				
9		Product ID	Price	
10		HW0007	182.63	← Wrong Result
11			179.65	← Correct Result

Look up Value ↑

You observe that you got –

- The correct result when range_lookup is FALSE, and
- A wrong result when range_lookup is TRUE.

This is because, the first column in the ProductInfo array is not sorted in ascending order. Hence, remember to use FALSE whenever the data is not sorted.

Using HLOOKUP Function

You can use **HLOOKUP** function if the data is in rows rather than columns.

Example

Let us take the example of product information. Suppose the array looks as follows –

	A	B	C	D	E	F	G
1							
2		Product ID	FC0002	HW0007	AP0024	DP0026	ISO0073
3		Product	Floor Cleaner	Hand Wash	Air Purifier	Detergent Powder	Soap
4		Price	191.9	179.65	254.28	182.63	85.85

Name this Array ProductRange. You can find the price of a product given the product ID with HLOOKUP function.

The Syntax of HLOOKUP function is

HLOOKUP (lookup_value, table_array, row_index_num, [range_lookup])

Where

lookup_value – is the value to be found in the first row of the table

table_array – is a table of information in which data is looked up

row_index_num – is the row number in table_array from which the matching value will be returned

range_lookup – is a logical value that specifies whether you want HLOOKUP to find an exact match or an approximate match

range_lookup can be

omitted, in which case it is assumed to be TRUE and HLOOKUP tries to find an approximate match

TRUE, in which case HLOOKUP tries to find an approximate match. In other words, if an exact match is not found, the next largest value that is less than lookup_value is returned

FALSE, in which case HLOOKUP tries to find an exact match

1, in which case it is assumed to be TRUE and HLOOKUP tries to find an approximate match

0, in which case it is assumed to be FALSE and HLOOKUP tries to find an exact match

Note – If range_lookup is Omitted or TRUE or 1, HLOOKUP works correctly only when the first column in table_array is sorted in ascending order. Otherwise, it may result in incorrect values. In such a case, use FALSE for range_lookup.

Using HLOOKUP Function with range_lookup FALSE

You can obtain the price of a product given the product ID with the HLOOKUP function as the product ID is in the first row. The price is in row 3 and hence row_index_num should be 3.

- Use HLOOKUP Function with range_lookup as TRUE.
- Use HLOOKUP Function with range_lookup as FALSE.

	A	B	C	D	E	F	G
1							
2		Product ID	FC0002	HW0007	AP0024	DP0026	ISO0073
3		Product	Floor Cleaner	Hand Wash	Air Purifier	Detergent Powder	Soap
4		Price	191.9	179.65	254.28	182.63	85.85
5							
6							
7		Product ID	HW0007				
8		Price	=HLOOKUP(C7,ProductRange,3,TRUE)			←	HLOOKUP with TRUE
9			=HLOOKUP(C7,ProductRange,3,FALSE)			←	HLOOKUP with FALSE

The correct answer from the ProductRange array is 171.65. You can check the results.

	A	B	C	D	E	F	G
1							
2		Product ID	FC0002	HW0007	AP0024	DP0026	ISO0073
3		Product	Floor Cleaner	Hand Wash	Air Purifier	Detergent Powder	Soap
4		Price	191.90	179.65	254.28	182.63	85.85
5							
6							
7		Product ID	HW0007				
8		Price	182.63				
9			179.65				

You observe that as in the case of VLOOKUP, you got

The correct result when range_lookup is FALSE, and

A wrong result when range_lookup is TRUE.

This is because the first row in the ProductRange array is not sorted in ascending order. Hence, remember to use FALSE whenever the data is not sorted.

Using HLOOKUP Function with range_lookup TRUE

Consider the example of student marks used in VLOOKUP. Suppose you have the data in rows instead of columns as shown in the table given below –

table_array –

Student Grades					
Marks	0	35	50	60	75
Pass Category	Fail	Third Class	Second Class	First Class	First Class with Distinction

Name this array as GradesRange.

Note that the first row marks based on which the grades are obtained is sorted in ascending order. Hence, using HLOOKUP with TRUE for range_lookup argument, you can get the Grades with approximate match and that is what is required.

	A	B	C	D	E	F	G
1							
2							
3		Student Grades					
4		Marks	0	35	50	60	75
5		Pass Category	Fail	Third Class	Second Class	First Class	First Class with Distinction
6							
7		Exam Results					
8		Marks	Pass Category				
9		85	=HLOOKUP(B9,GradesRange,2,TRUE)				
10		75	=HLOOKUP(B10,GradesRange,2,TRUE)				
11		72	=HLOOKUP(B11,GradesRange,2,TRUE)				
12		55	=HLOOKUP(B12,GradesRange,2,TRUE)				
13		68	=HLOOKUP(B13,GradesRange,2,TRUE)				
14		34	=HLOOKUP(B14,GradesRange,2,TRUE)				
15		60	=HLOOKUP(B15,GradesRange,2,TRUE)				
16		50	=HLOOKUP(B16,GradesRange,2,TRUE)				
17		98	=HLOOKUP(B17,GradesRange,2,TRUE)				
18		59	=HLOOKUP(B18,GradesRange,2,TRUE)				
19		74	=HLOOKUP(B19,GradesRange,2,TRUE)				
20		99	=HLOOKUP(B20,GradesRange,2,TRUE)				
21		40	=HLOOKUP(B21,GradesRange,2,TRUE)				
22		35	=HLOOKUP(B22,GradesRange,2,TRUE)				

As you can observe,

row_index_num – indicates the column of the return value in table_array is 2

the **range_lookup** is TRUE

The first column containing the lookup value in the table_array Grades is in ascending order. Hence, the results will be correct

You can get the return value for approximate matches also. i.e. HLOOKUP computes as follows –

Marks	< 35	>= 35 and < 50	>= 50 and < 60	>=60 and < 75	>= 75
Pass Category	Fail	Third Class	Second Class	First Class	First Class with Distinction

You will get the following results –

	A	B	C	D	E	F	G
1							
2		Student Grades					
3		Marks	0	35	50	60	75
4		Pass Category	Fail	Third Class	Second Class	First Class	First Class with Distinction
5							
6		Exam Results					
7		Marks	Pass Category				
8		85	First Class with Distinction				
9		75	First Class with Distinction				
10		72	First Class				
11		55	Second Class				
12		68	First Class				
13		34	Fail				
14		60	First Class				
15		50	Second Class				
16		98	First Class with Distinction				
17		59	Second Class				
18		74	First Class				
19		99	First Class with Distinction				
20		40	Third Class				
21		35	Third Class				

Using INDEX Function

When you have an array of data, you can retrieve a value in the array by specifying the row number and column number of that value in the array.

Consider the following sales data, wherein you find the sales in each of the North, South, East and West regions by the salespersons who are listed.

	A	B	C	D	E	F
1						
2			North	South	East	West
3		Vicky	406	107	251	562
4		Mathew	433	192	464	536
5		Ritchie	330	433	597	577
6		Jane	435	103	549	600
7		Sara	446	126	344	109
8		Andy	289	515	379	529
9		Bob	424	349	345	147
10		James	175	194	581	141
11		Katherine	179	372	431	215
12		Hardley	279	464	120	442

- Name the array as SalesData.

Using INDEX Function, you can find –

- The Sales of any of the Salespersons in a certain Region.
- Total Sales in a Region by all the Salespersons.
- Total Sales by a Salesperson in all the Regions.

	A	B	C	D	E	F
1						
2			North	South	East	West
3		Vicky	406	107	251	562
4		Mathew	433	192	464	536
5		Ritchie	330	433	597	577
6		Jane	435	103	549	600
7		Sara	446	126	344	109
8		Andy	289	515	379	529
9		Bob	424	349	345	147
10		James	175	194	581	141
11		Katherine	179	372	431	215
12		Hardley	279	464	120	442
13						
14		Vicky - South Sales	=INDEX(SalesData,1,2)			
15		Andy - West Sales	=INDEX(SalesData,6,4)			
16		Total North Sales	=SUM(INDEX(SalesData,0,1))			
17		James Total Sales	=SUM(INDEX(SalesData,8,0))			

You will get the following results –

	A	B	C	D	E	F
1						
2			North	South	East	West
3		Vicky	406	107	251	562
4		Mathew	433	192	464	536
5		Ritchie	330	433	597	577
6		Jane	435	103	549	600
7		Sara	446	126	344	109
8		Andy	289	515	379	529
9		Bob	424	349	345	147
10		James	175	194	581	141
11		Katherine	179	372	431	215
12		Hardley	279	464	120	442
13						
14		Vicky - South Sales	107			
15		Andy - West Sales	529			
16		Total North Sales	3396			
17		James Total Sales	1091			

Suppose you do not know the row numbers for the salespersons and column numbers for the regions. Then, you need to find the row number and column number first before you retrieve the value with the index function.

You can do it with the MATCH function as explained in the next section.

Using MATCH Function

If you need the position of an item in a range, you can use the MATCH function. You can combine MATCH and INDEX functions as follows –

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2			North	South	East	West		Name	Region	Row-Num of Name	Col_Num of Region	Result
3		Vicky	406	107	251	562		Vicky	South	=MATCH(I3,B3:B12,0)	=MATCH(I3,C2:F2,0)	=INDEX(C3:F12,J3,K3)
4		Mathew	433	192	464	536		Andy	West	=MATCH(I4,B3:B12,0)	=MATCH(I4,C2:F2,0)	=INDEX(C3:F12,J4,K4)
5		Ritchie	330	433	597	577		All	North	0		=SUM(IINDEX(C3:F12,J5,K5))
6		Jane	435	103	549	600		James	All	=MATCH(I6,B3:B12,0)	0	=SUM(IINDEX(C3:F12,J6,K6))
7		Sara	446	126	344	109						
8		Andy	289	515	379	529						
9		Bob	424	349	345	147						
10		James	175	194	581	141						
11		Katherine	179	372	431	215						
12		Hardley	279	464	120	442						

You will get the following results –

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2			North	South	East	West		Name	Region	Row-Num of Name	Col_Num of Region	Result
3		Vicky	406	107	251	562		Vicky	South	1	2	107
4		Mathew	433	192	464	536		Andy	West	6	4	529
5		Ritchie	330	433	597	577		All	North	0	1	3396
6		Jane	435	103	549	600		James	All	8	0	1091
7		Sara	446	126	344	109						
8		Andy	289	515	379	529						
9		Bob	424	349	345	147						
10		James	175	194	581	141						
11		Katherine	179	372	431	215						
12		Hardley	279	464	120	442						

II. TABLEAU

Tableau is a Business Intelligence tool for visually analyzing the data. Users can create and distribute an interactive and shareable dashboard, which depict the trends, variations, and density of the data in the form of graphs and charts. Tableau can connect to files, relational and Big Data sources to acquire and process data. The software allows data blending and real-time collaboration, which makes it very unique. It is used by businesses, academic researchers, and many government organizations for visual data analysis. It is also positioned as a leader Business Intelligence and Analytics Platform in Gartner Magic Quadrant.

Tableau Features

Tableau provides solutions for all kinds of industries, departments, and data environments. Following are some unique features which enable Tableau to handle diverse scenarios.

Speed of Analysis – As it does not require high level of programming expertise, any user with access to data can start using it to derive value from the data.

Self-Reliant – Tableau does not need a complex software setup. The desktop version which is used by most users is easily installed and contains all the features needed to start and complete data analysis.

Visual Discovery – The user explores and analyzes the data by using visual tools like colors, trend lines, charts, and graphs. There is very little script to be written as nearly everything is done by drag and drop.

Blend Diverse Data Sets – Tableau allows you to blend different relational, semistructured and raw data sources in real time, without expensive up-front integration costs. The users don't need to know the details of how data is stored.

Architecture Agnostic – Tableau works in all kinds of devices where data flows. Hence, the user need not worry about specific hardware or software requirements to use Tableau.

Real-Time Collaboration – Tableau can filter, sort, and discuss data on the fly and embed a live dashboard in portals like SharePoint site or Salesforce. You can save your view of data and allow colleagues to subscribe to your interactive dashboards so they see the very latest data just by refreshing their web browser.

Centralized Data – Tableau server provides a centralized location to manage all of the organization's published data sources. You can delete, change permissions, add tags, and manage schedules in one convenient location. It's easy to schedule extract refreshes and manage them in the data server. Administrators can centrally define a schedule for extracts on the server for both incremental and full refreshes.

These three steps are –

Connect to a data source – It involves locating the data and using an appropriate type of connection to read the data.

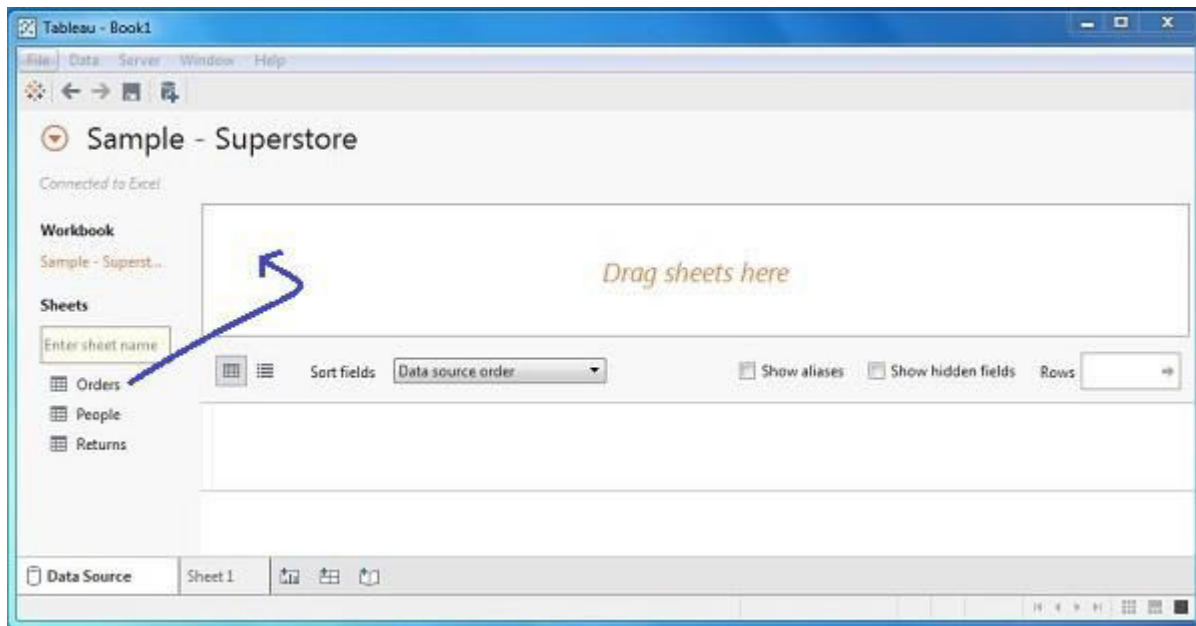
Choose dimensions and measures – This involves selecting the required columns from the source data for analysis.

Apply visualization technique – This involves applying required visualization methods, such as a specific chart or graph type to the data being analyzed.

For convenience, let's use the sample data set that comes with Tableau installation named sample – superstore.xls. Locate the installation folder of Tableau and go to **My Tableau Repository**. Under it, you will find the above file at **Datasources\9.2\en_US-US**.

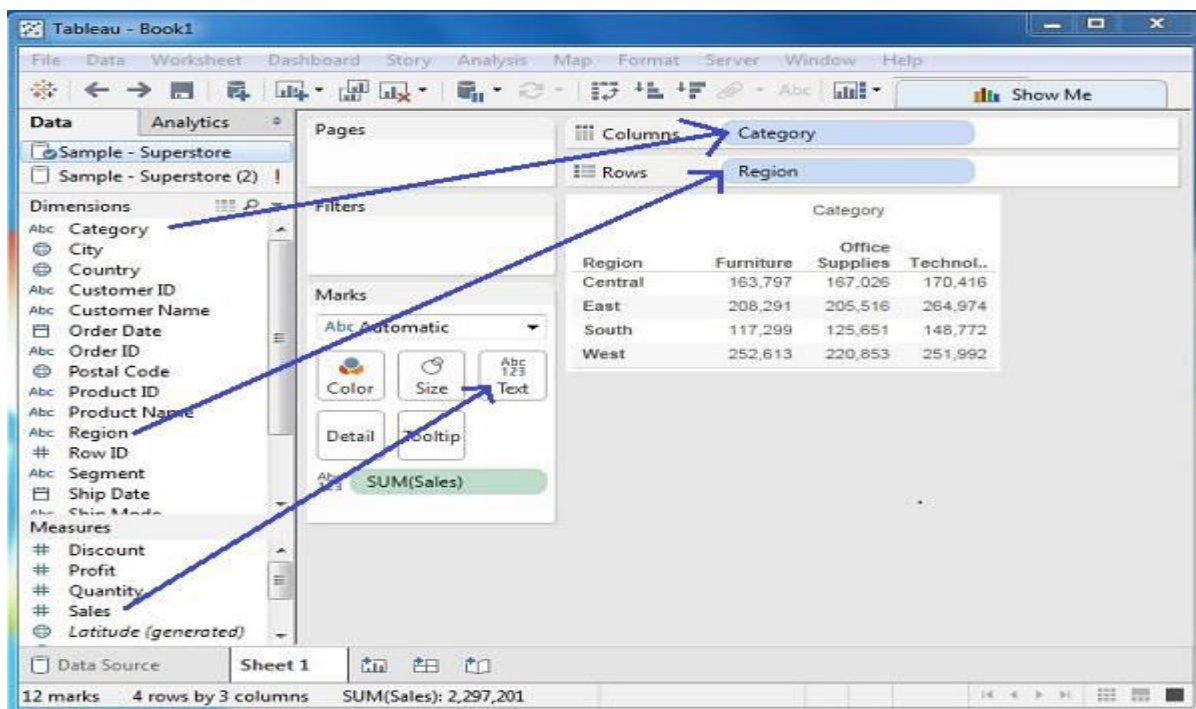
Connect to a Data Source

On opening Tableau, you will get the start page showing various data sources. Under the header “**Connect**”, you have options to choose a file or server or saved data source. Under Files, choose excel. Then navigate to the file “**Sample – Superstore.xls**” as mentioned above. The excel file has three sheets named Orders, People and Returns. Choose **Orders**.



Choose the Dimensions and Measures

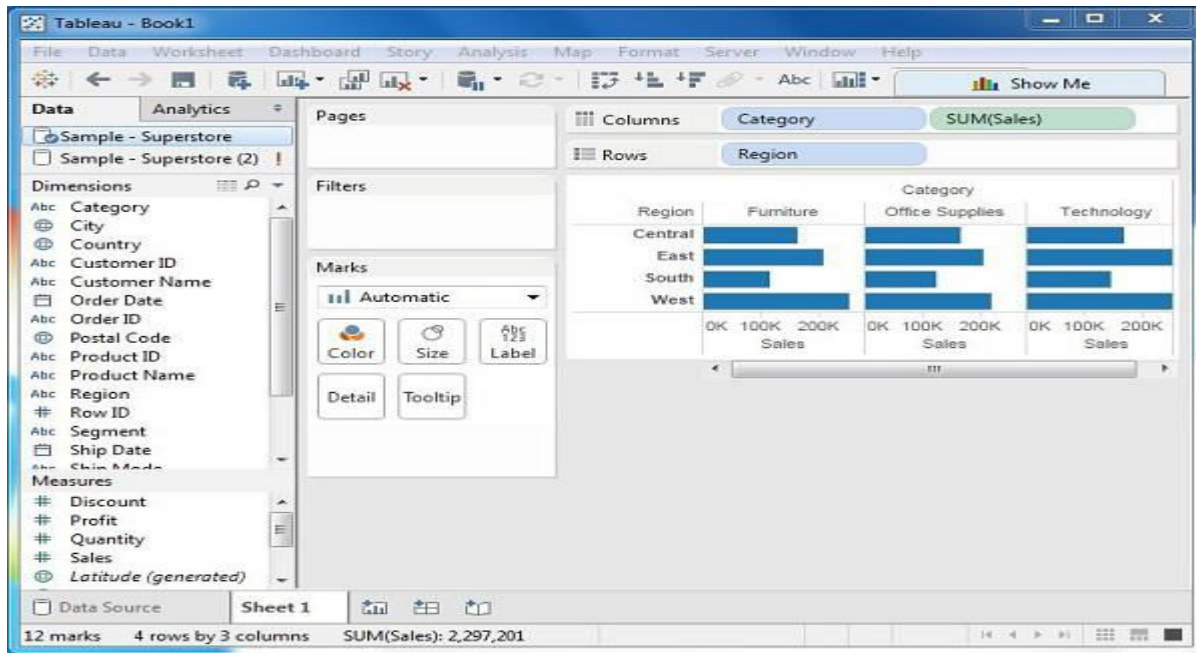
Next, choose the data to be analyzed by deciding on the dimensions and measures. Dimensions are the descriptive data while measures are numeric data. When put together, they help visualize the performance of the dimensional data with respect to the data which are measures. Choose **Category** and **Region** as the dimensions and **Sales** as the measure. Drag and drop them as shown in the following screenshot. The result shows the total sales in each category for each region.



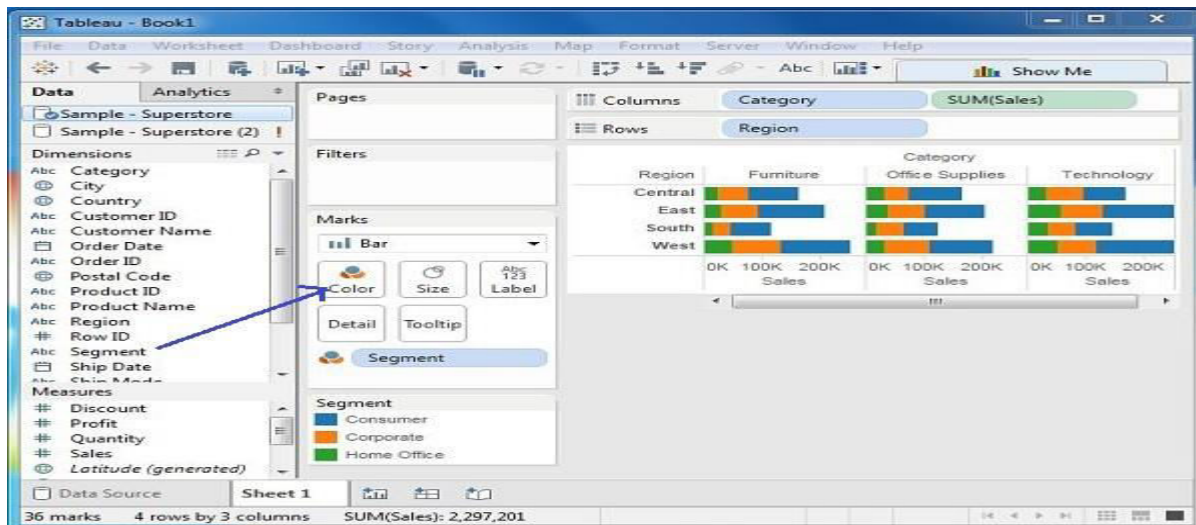
Apply Visualization Technique

In the previous step, you can see that the data is available only as numbers. You have to read and calculate each of the values to judge the performance. However, you can see them as graphs or charts with different colors to make a quicker judgment.

We drag and drop the sum (sales) column from the Marks tab to the Columns shelf. The table showing the numeric values of sales now turns into a bar chart automatically.

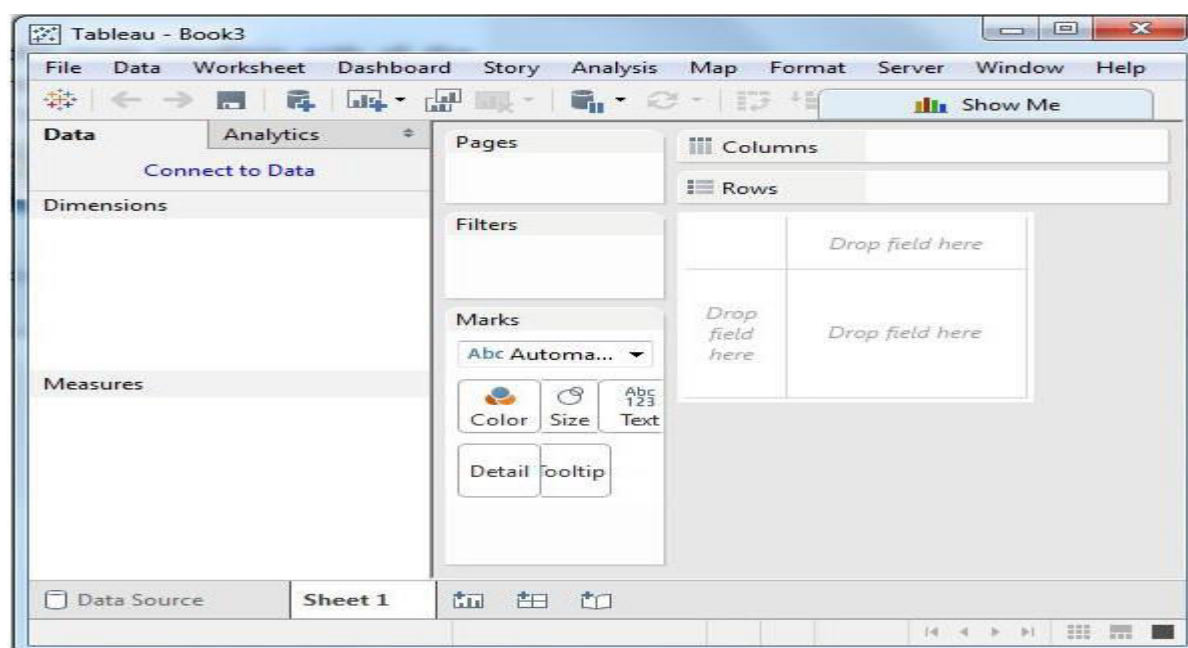


You can apply a technique of adding another dimension to the existing data. This will add more colors to the existing bar chart as shown in the following screenshot.



Menu Commands

On closing the getting started window, you get the main interface with all the available Menu commands. They represent the entire set of features available in Tableau. Various sections of the menu are shown in the following diagram. Next, you can see some details of each menu.



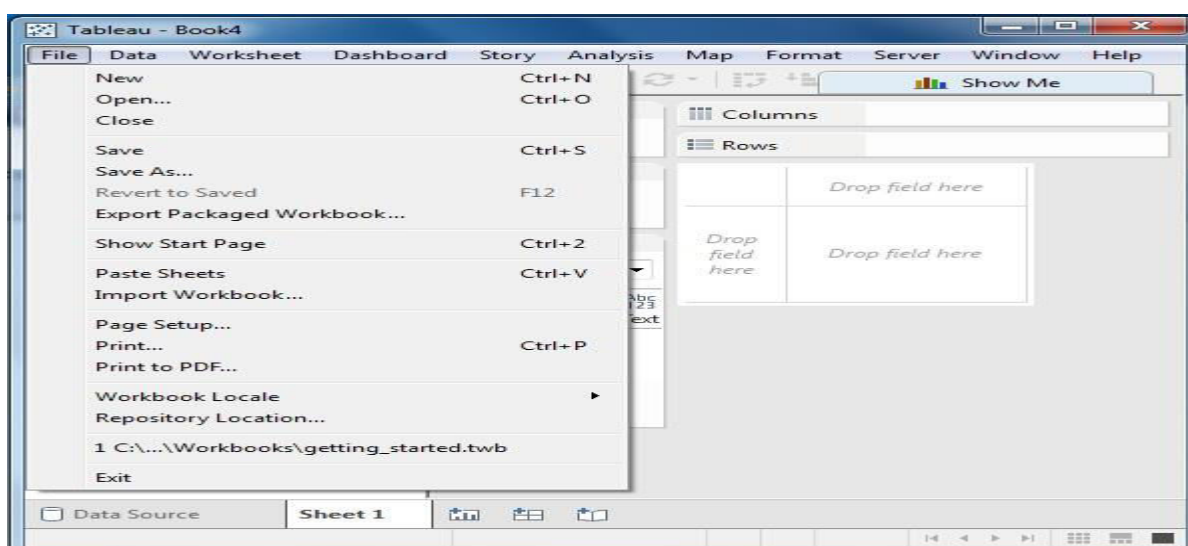
File Menu

This menu is used to create a new Tableau workbook and open existing workbooks from both the local system and Tableau server. The important features in this menu are –

Workbook Locale sets the language to be used in the report.

Paste Sheets pastes a sheet into the current workbook, which is copied from another workbook.

Export Packaged Workbook option is used to create a packaged workbook, which will be shared with other users.



Data Menu

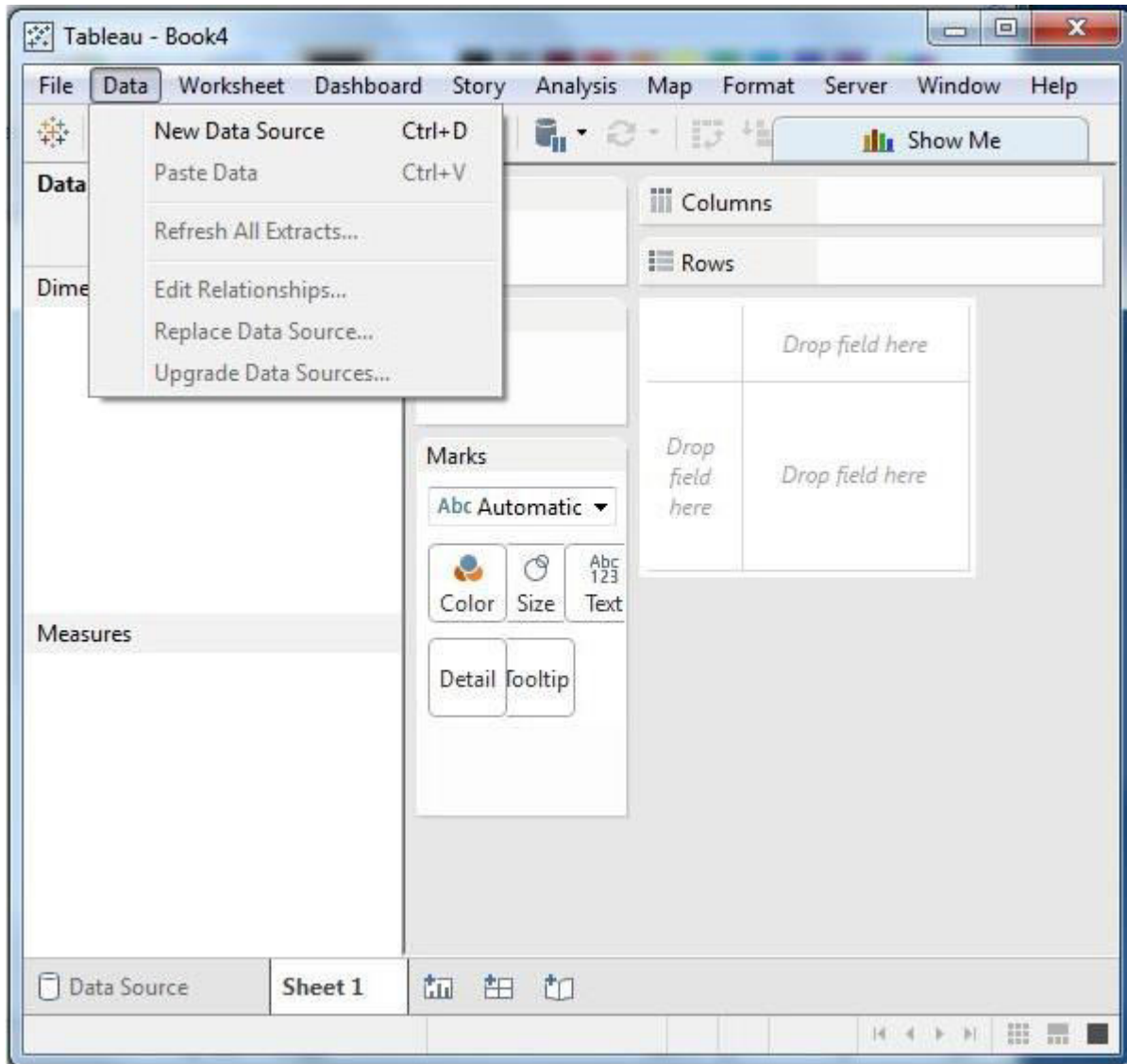
This menu is used to create new data source to fetch the data for analysis and visualization. It also allows you to replace or upgrade the existing data source.

The important features in this menu are as follows –

New Data Source allows to view all the types of connections available and choose from it.

Refresh All Extracts refreshes the data from the source.

Edit Relationships option defines the fields in more than one data source for linking.



Worksheet Menu

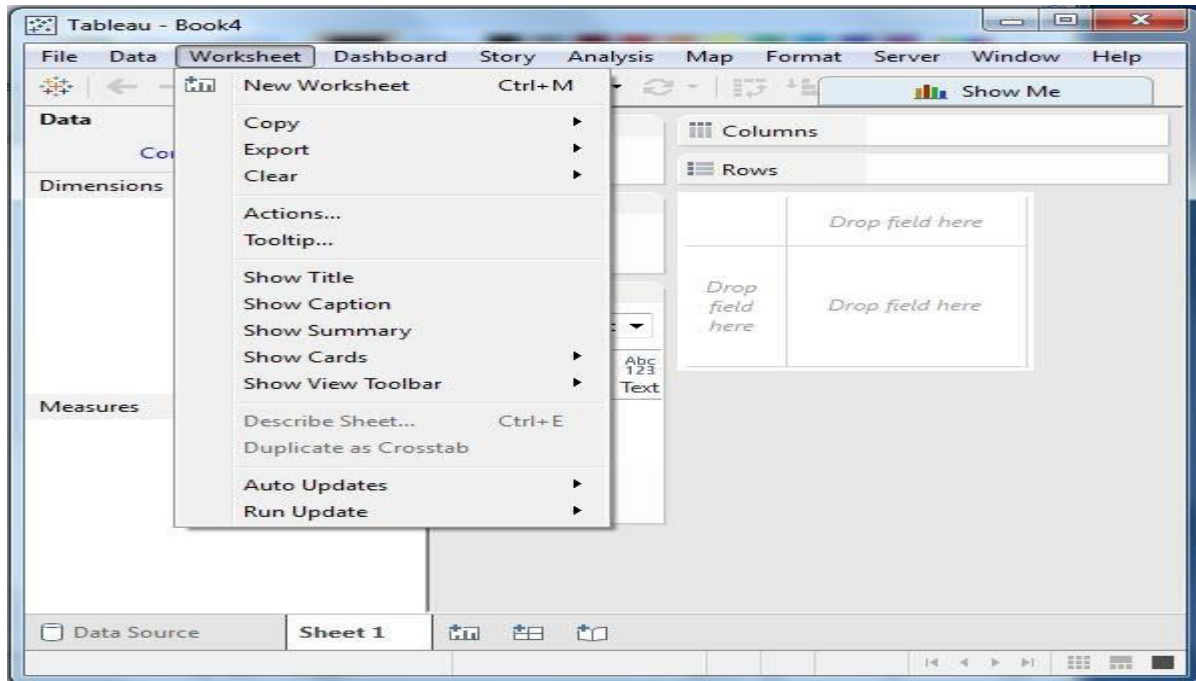
This menu is used to create a new worksheet along with various display features such as showing the title and captions, etc.

The important features in this menu are as follows.

Show Summary allows to view the summary of the data used in the worksheet such as, count, etc.

Tooltip shows the tooltip when hovering above various data fields.

Run Update option updates the worksheet data or filters used.



Dashboard Menu

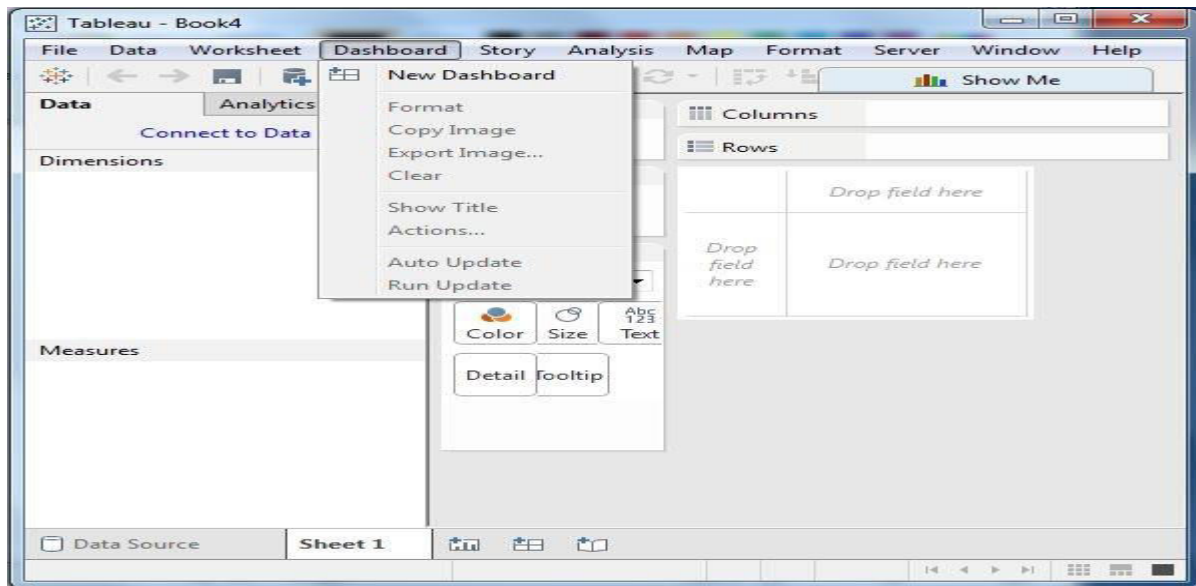
This menu is used to create a new dashboard along with various display features, such as showing the title and exporting the image, etc.

The important features in this menu are as follows –

Format sets the layout in terms of colors and sections of the dashboard.

Actions link the dashboard sheets to external URLs or other sheets.

Export Image option exports an image of the Dashboard.



Story Menu

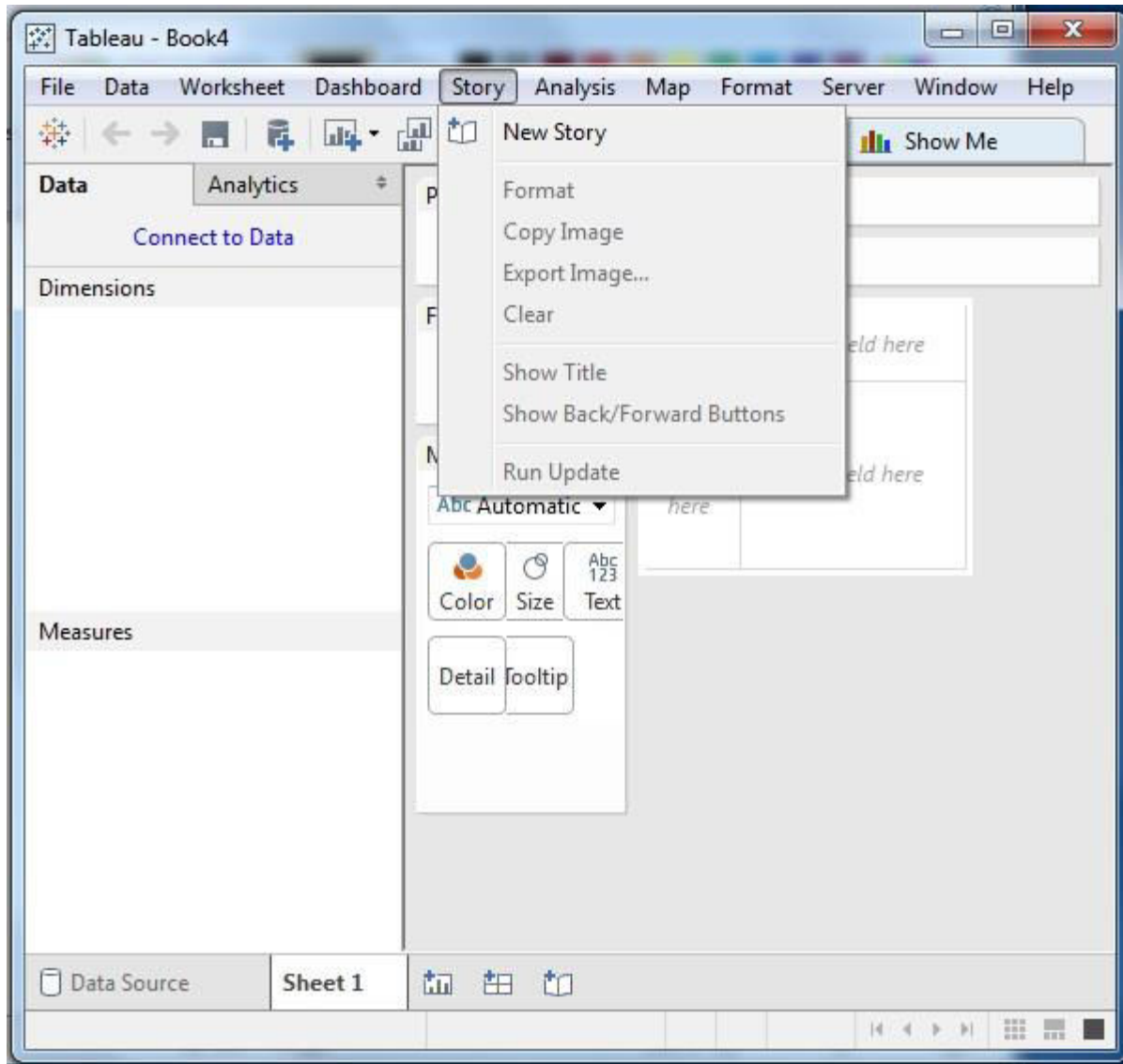
This menu is used to create a new story which has many sheets or dashboards with related data.

The important features in this menu are as follows –

Format sets the layout in terms of colors and sections of the story.

Run Update updates the story with the latest data from the source.

Export Image option exports an image of the story.



Analysis Menu

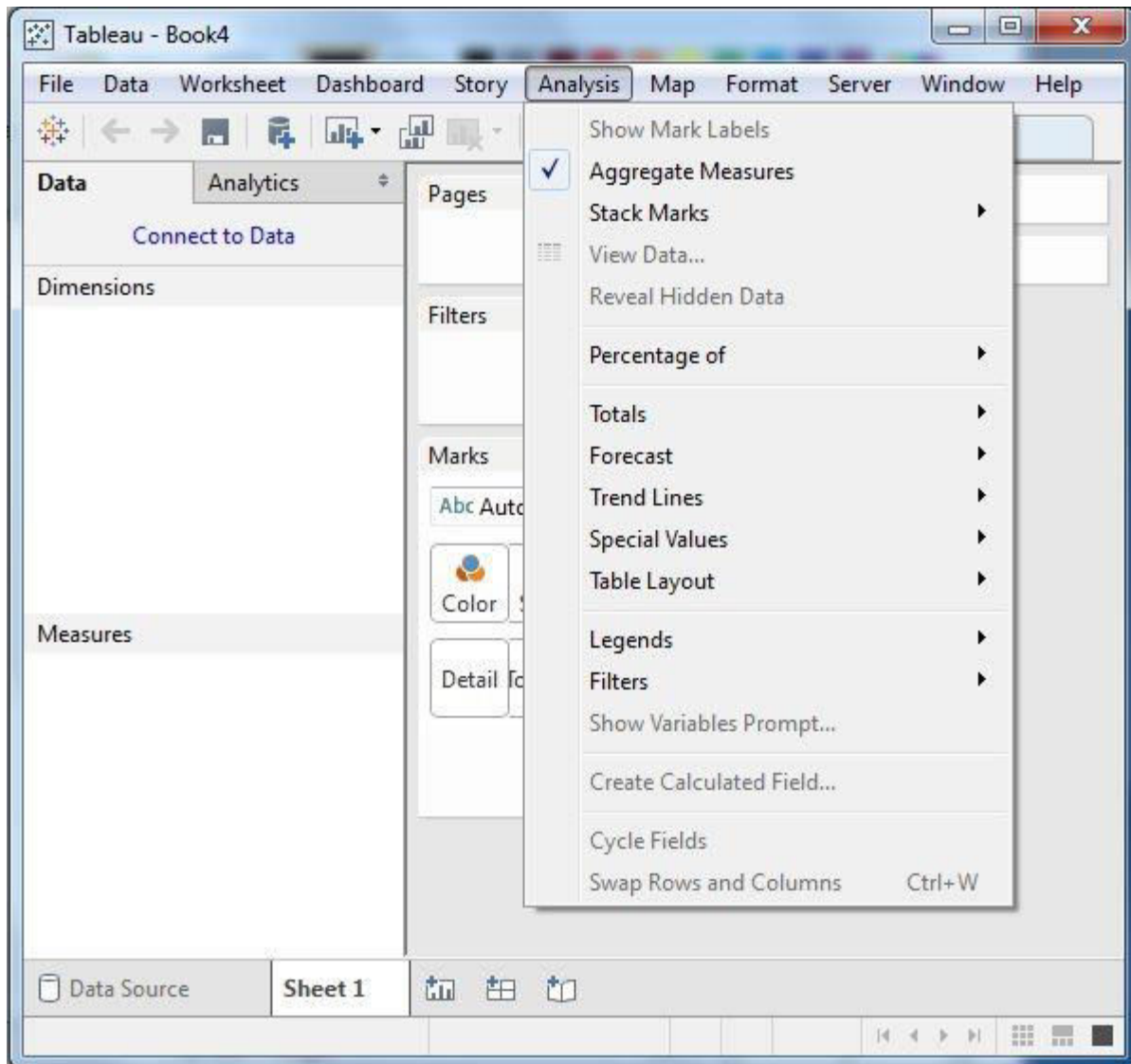
This menu is used for analyzing the data present in the sheet. Tableau provides many outof-the-box features, such as calculating the percentage and performing a forecast, etc.

The important features in this menu are as follows –

Forecast shows a forecast based on available data.

Trend Lines shows the trend line for a series of data.

Create Calculated Field option creates additional fields based on certain calculation on the existing fields.



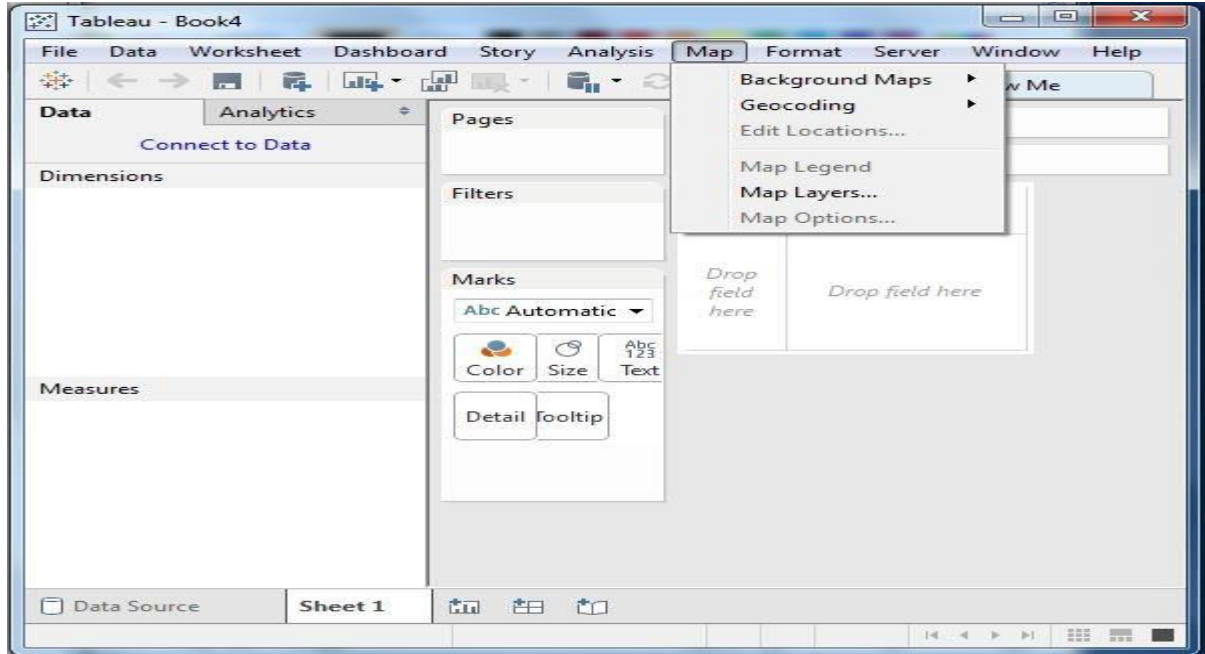
Map Menu

This menu is used for building map views in Tableau. You can assign geographic roles to fields in your data.

The important features in this menu are as follows –

Map Layers hides and shows map layers, such as street names, country borders, and adds data layers.

Geocoding creates new geographic roles and assigns them to the geographic fields in your data.



Format Menu

This menu is used for applying the various formatting options to enhance the look and feel of the dashboards created. It provides features such as borders, colors, alignment of text, etc.

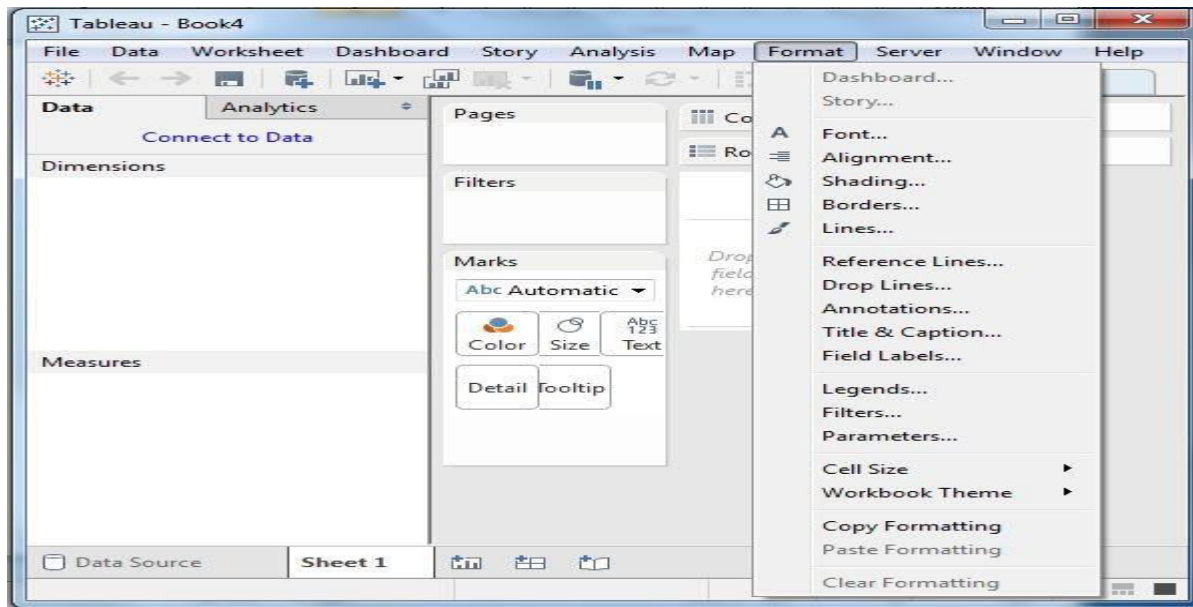
The important features in this menu are as follows –

Borders applies borders to the fields displayed in the report.

Title & Caption assigns a title and caption to the reports.

Cell Size customizes the size of the cells displaying the data.

Workbook Theme applies a theme to the entire workbook.



Server Menu

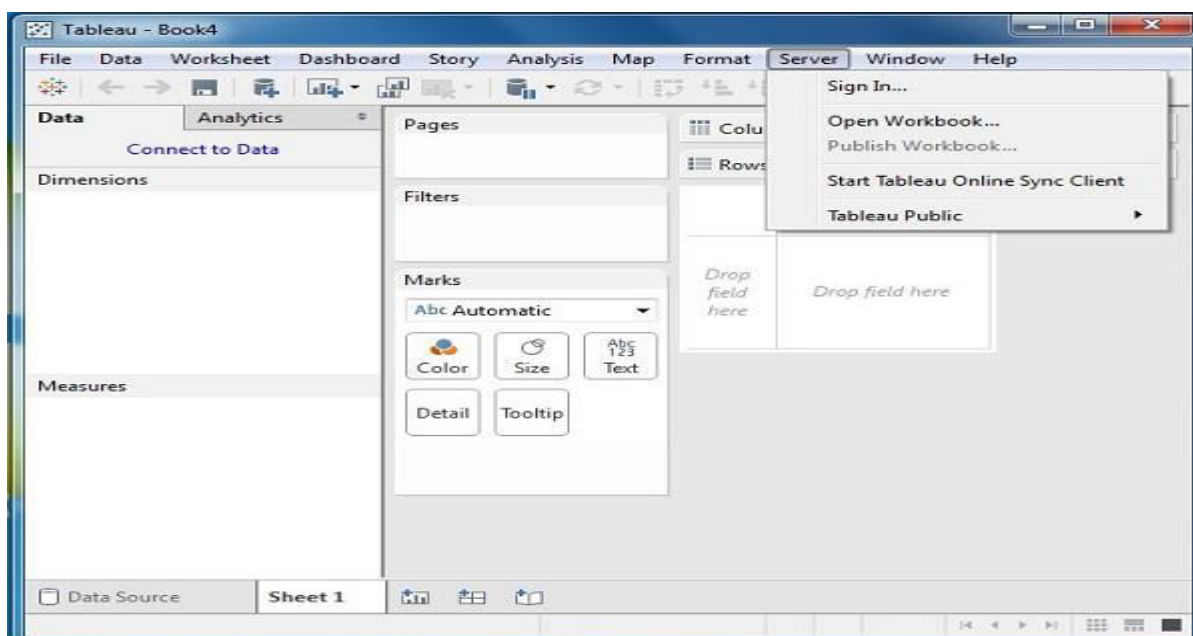
Server Menu is used to login to the Tableau server if you have access, and publish your results to be used by others. It is also used to access the workbooks published by others.

The important features in this menu are as follows –

Publish Workbook publishes the workbook in the server to be used by others.

Publish Data Source publishes the source data used in the workbook.

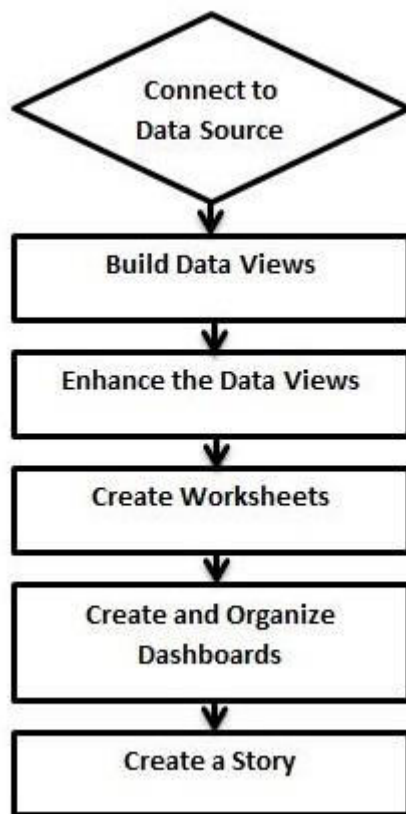
Create User Filters creates filters on the worksheet to be applied by various users while accessing the report.



Workflow:

As Tableau helps in analyzing lots of data over diverse time periods, dimensions, and measures, it needs a very meticulous planning to create a good dashboard or story. Hence, it is important to know the approach to design a good dashboard. Like any other field of human endeavor, there are many best practices to be followed to create good worksheets and dashboards.

Though the final outcome expected from a Tableau project is ideally a dashboard with story, there are many intermediate steps which needs to be completed to reach this goal. Following is a flow diagram of design steps that should be ideally followed to create effective dashboards.



Connect to Data Source

Tableau connects to all popular data sources. It has inbuilt connectors which take care of establishing the connection, once the connection parameters are supplied. Be it simple text files, relational sources, SQL sources or cloud data bases, Tableau connects to nearly every data source.

Build Data Views

After connecting to a data source, you get all the column and data available in the Tableau environment. You classify them as dimensions and measures, and create any hierarchy required.

Using these you build views, which are traditionally known as Reports. Tableau provides easy drag and drop feature to build views.

Enhance the Views

The views created above needs to be enhanced further by the use of filters, aggregations, labeling of axes, formatting of colors and borders, etc.

Create Worksheets

Create different worksheets to create different views on the same or different data.

Create and Organize Dashboards

Dashboards contain multiple worksheets which are linked. Hence, the action in any of the worksheet can change the result in the dashboard accordingly.

Create a Story

A story is a sheet that contains a sequence of worksheets or dashboards that work together to convey information. You can create stories to show how facts are connected, provide context, demonstrate how decisions relate to outcomes, or simply make a compelling case.

Sorting:

Sorting of data is a very important feature of data analysis. Tableau allows the sorting of data of the fields, which are called dimensions. There are two ways in which Tableau carries out the sorting.

Computed Sorting is the sort directly applied on an axis using the sort dialog button.

Manual Sorting is used to rearrange the order of dimension fields by dragging them next to each other in an ad hoc fashion.

Computed Sorting

This type of sorting involves choosing a field to be sorted and directly applying the sort using the sort dialog box. You have the option to choose the sort order as ascending or descending and choose the field on which to apply the sort.

Example

Choose Sample-Superstore to apply sorting on the field named **discount** by using the dimensions order date and Subcategory as shown below. The result shows the name of the sub-categories in a descending order arranged for each year.

Manual Sorting

This is basically changing the order in which the visualization elements appear in the screen. For example, you want to show the sales volume of different product segment in a descending order, however you have your own choice of order. This sort is not as per the exact values of number or text, rather they represent the user's choice of ordering. Hence, they are called as manual sorting.

In the following example, you move the segment named Home Office, below the segment named Consumer, even though the sales volume of Home Office is the lowest.

Filtering:

Filtering is the process of removing certain values or range of values from a result set. Tableau filtering feature allows both simple scenarios using field values as well as advanced calculation or context-based filters. In this chapter, you will learn about the basic filters available in Tableau.

There are three types of basic filters available in Tableau. They are as follows –

Filter Dimensions are the filters applied on the dimension fields.

Filter Measures are the filters applied on the measure fields.

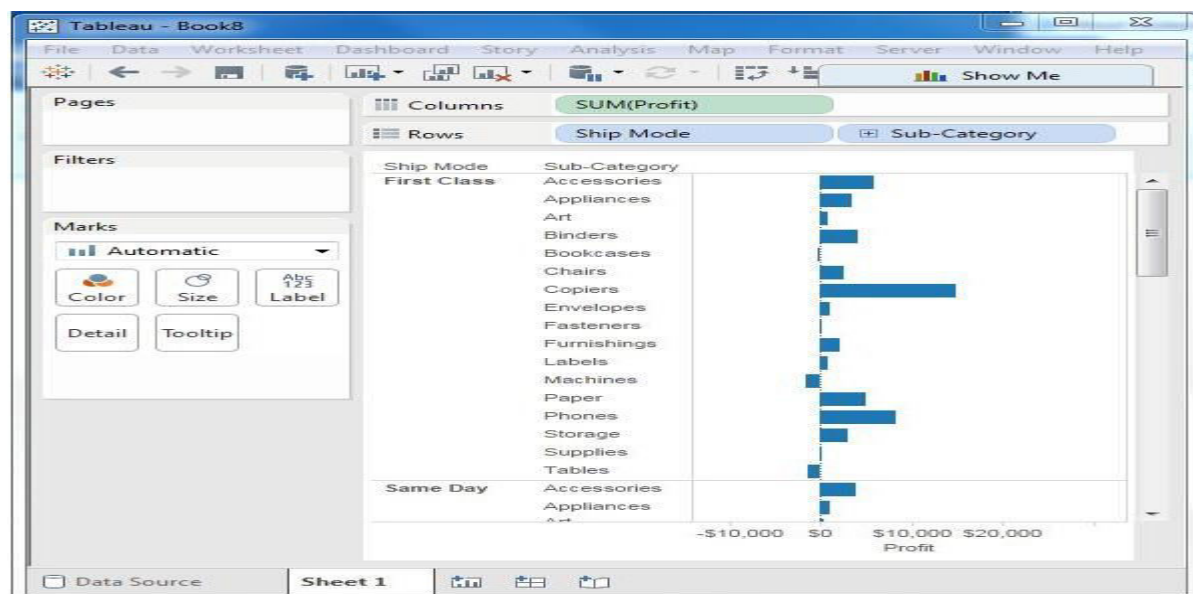
Filter Dates are the filters applied on the date fields.

Filter Dimensions

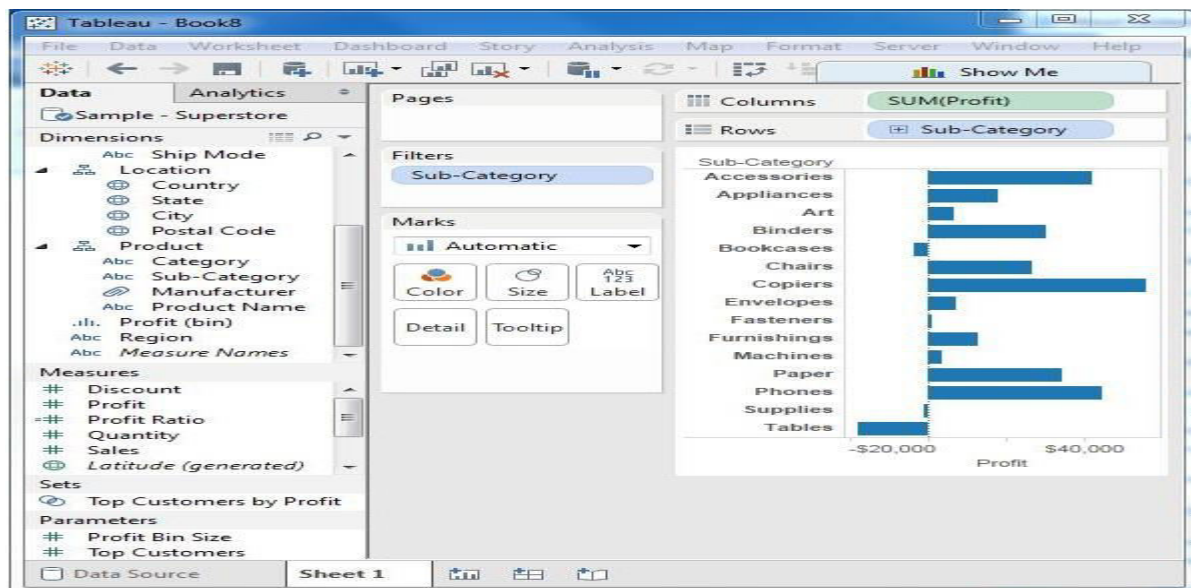
These filters are applied on the dimension fields. Typical examples include filtering based on categories of text or numeric values with logical expressions greater than or less than conditions.

Example

We use the Sample - Superstore data source to apply dimension filters on the sub-category of products. We create a view for showing profit for each sub-category of products according to their shipping mode. For it, drag the dimension field “Sub-Category” to the Rows shelf and the measure field “profit” to the Columns shelf.



Next, drag the Sub-Category dimension to the Filters shelf to open the Filter dialog box. Click the None button at the bottom of the list to deselect all segments. Then, select the Exclude option in the lower right corner of the dialog box. Finally, select Labels and Storage and then click OK. The following screenshot shows the result with the above two categories excluded.

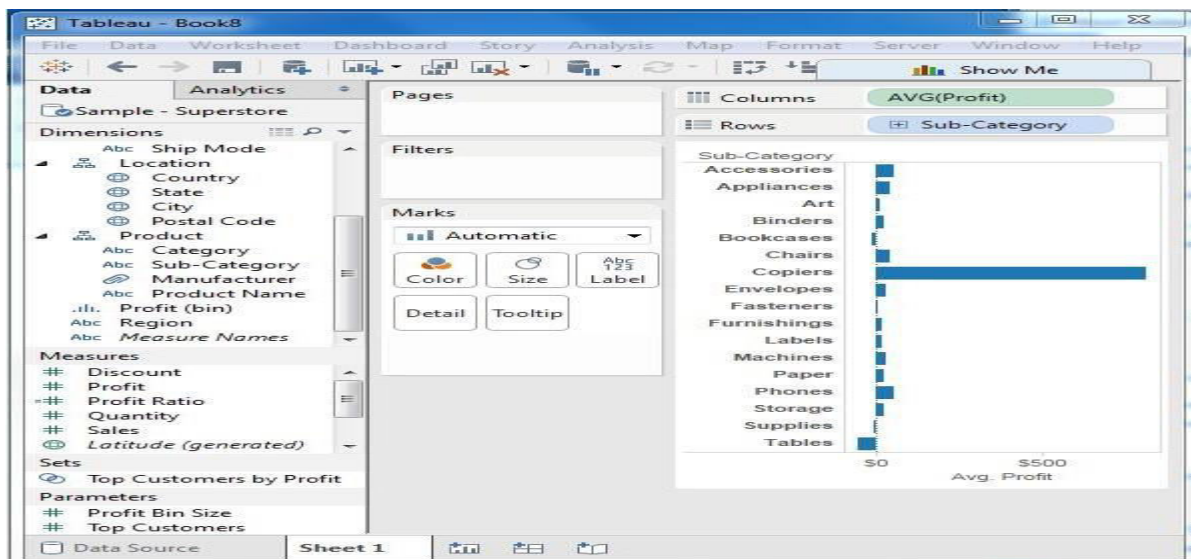


Filter Measures

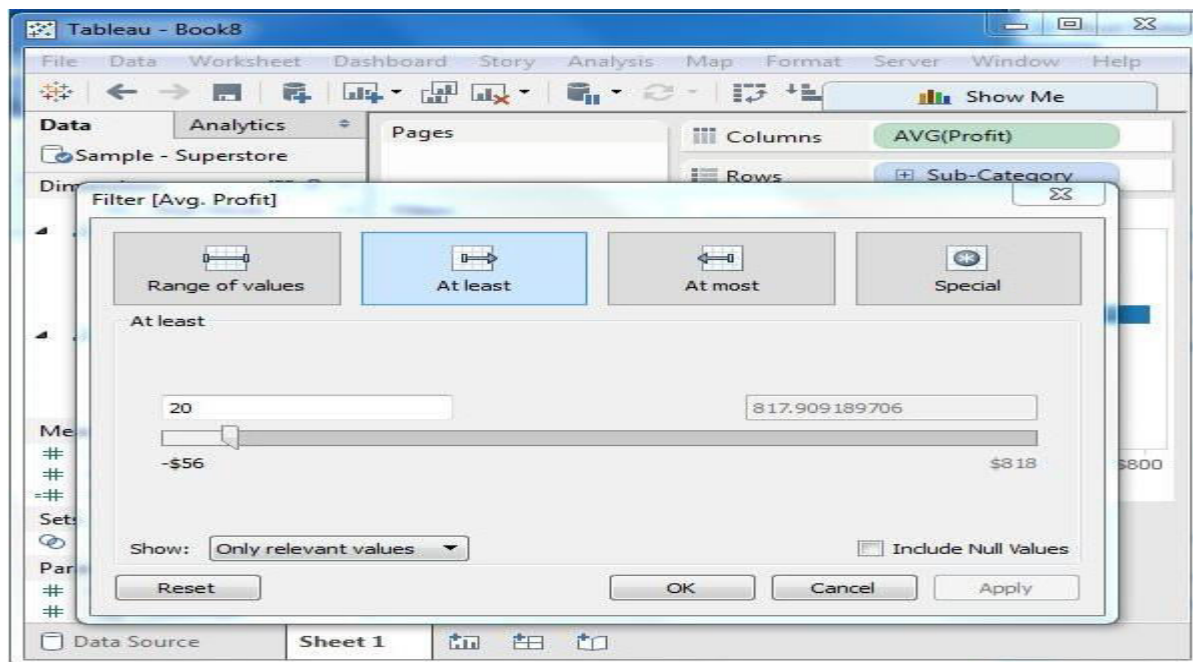
These filters are applied on the measure fields. Filtering is based on the calculations applied to the measure fields. Hence, while in dimension filters you use only values to filter, in measures filter you use calculations based on fields.

Example

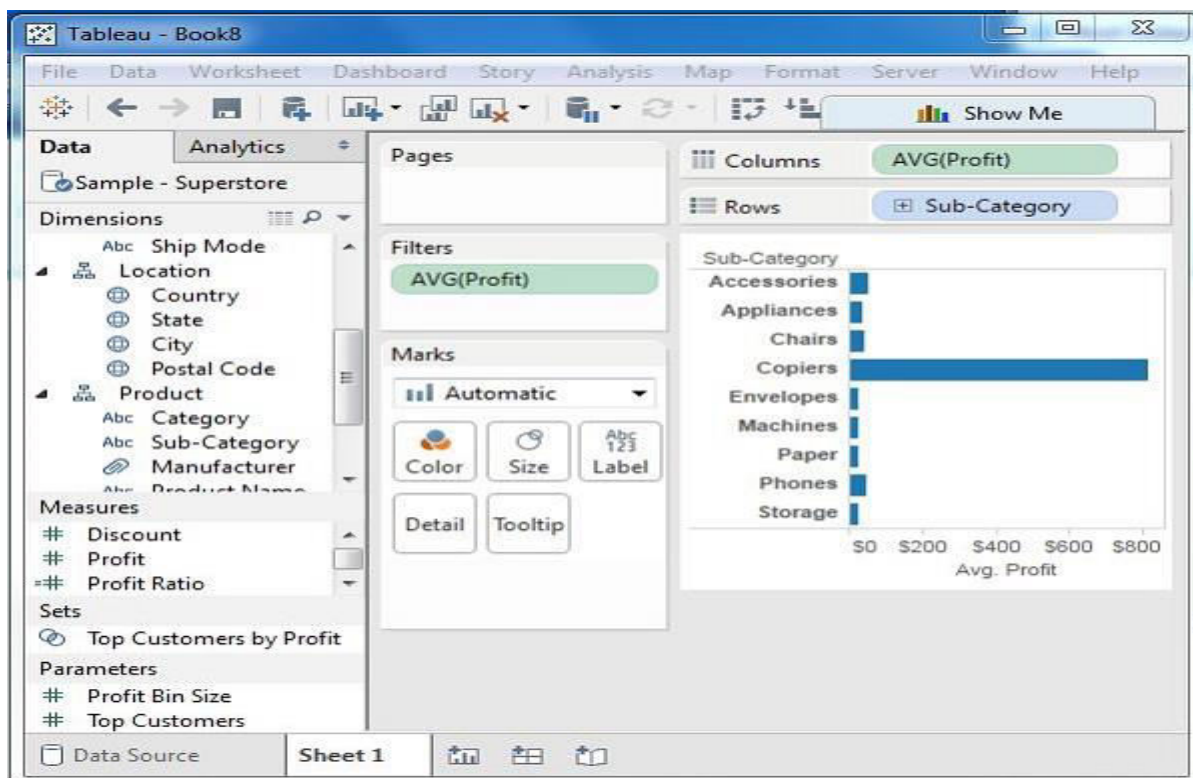
You can use the Sample - Superstore data source to apply dimension filters on the average value of the profits. First, create a view with ship mode and subcategory as dimensions and Average of profit as shown in the following screenshot.



Next, drag the AVG (profit) value to the filter pane. Choose Average as the filter mode. Next, choose "At least" and give a value to filter the rows, which meet these criteria.



After completion of the above steps, we get the final view below showing only the subcategories whose average profit is greater than 20.

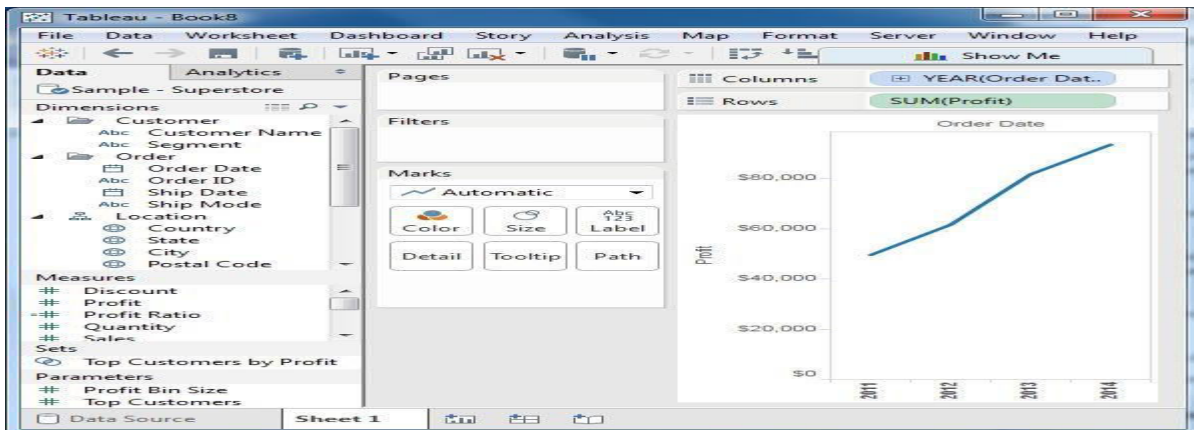


Filter Dates

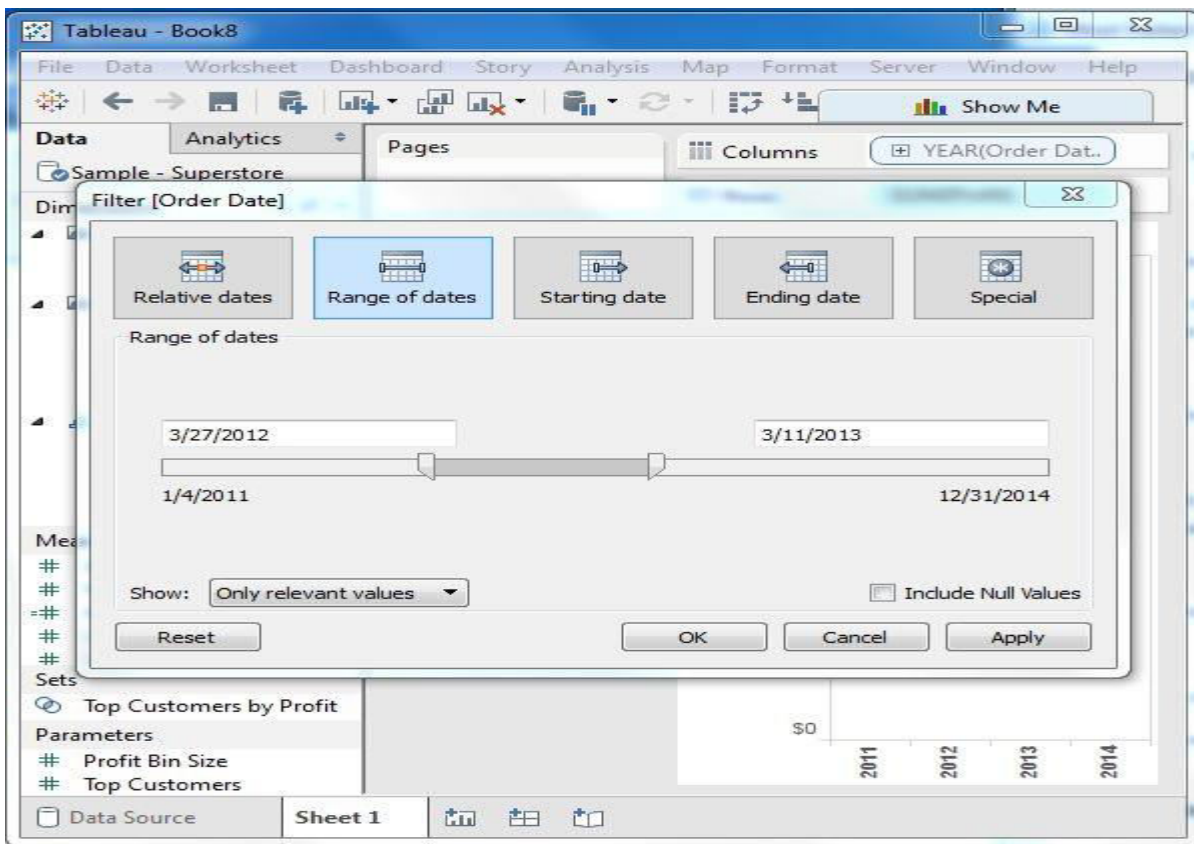
Tableau treats the date field in three different ways while applying the date field. It can apply filter by taking a relative date as compared to today, an absolute date, or range of dates. Each of this option is presented when a date field is dragged out of the filter pane.

Example

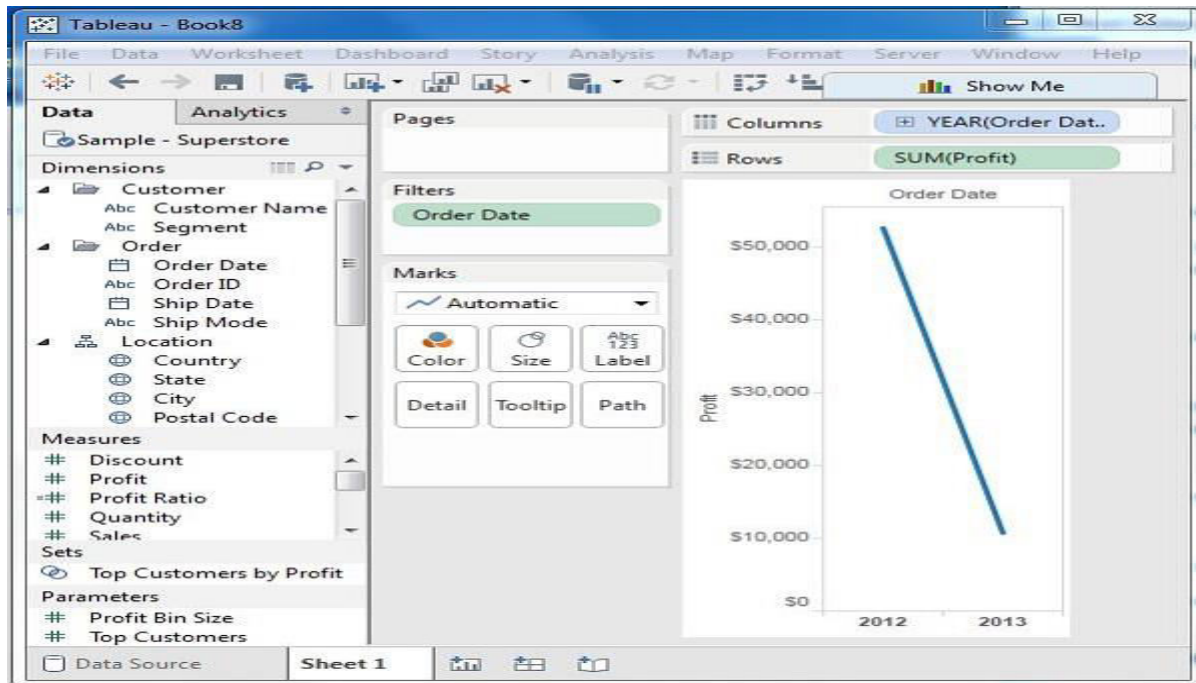
We choose the sample - Superstore data source and create a view with order date in the column shelf and profit in the rows shelf as shown in the following screenshot.



Next, drag the "order date" field to the filter shelf and choose Range of dates in the filter dialog box. Choose the dates as shown in the following screenshot.



On clicking OK, the final view appears showing the result for the chosen range of dates as seen in the following screenshot.



Creating Dashboard:

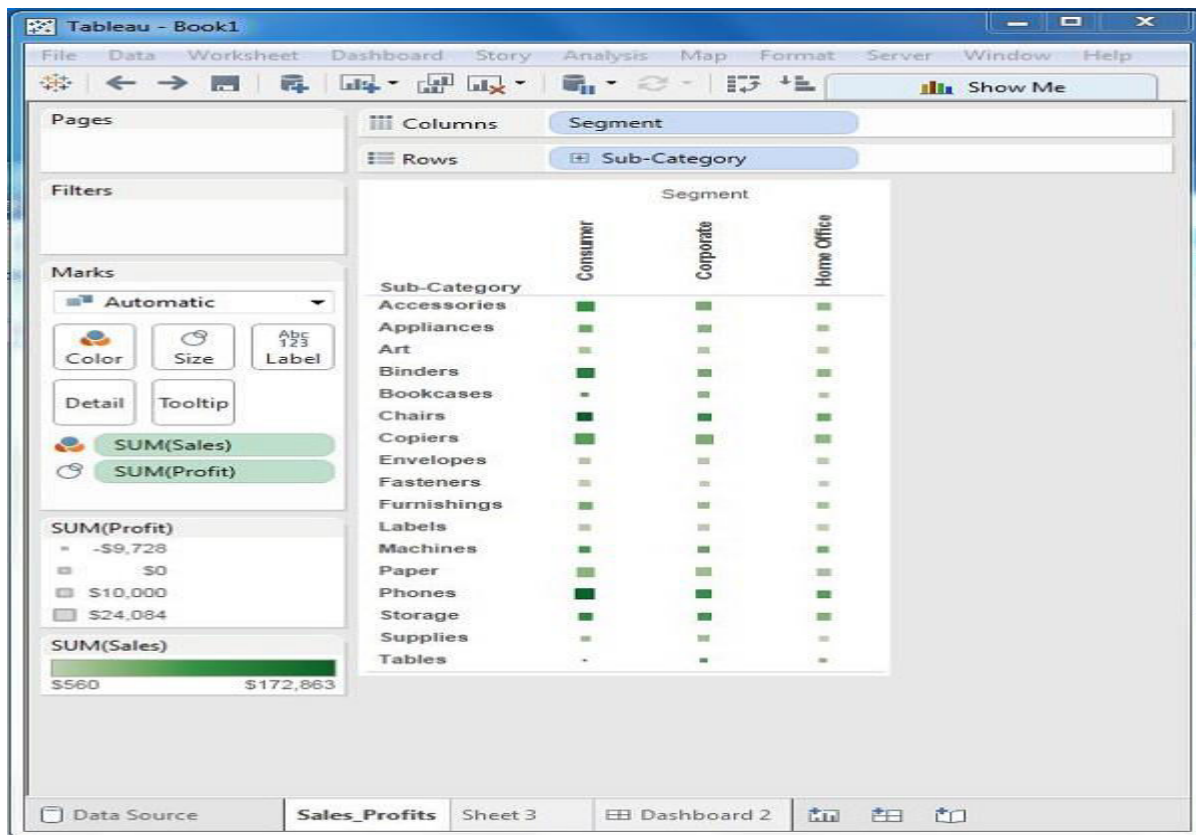
A dashboard is a consolidated display of many worksheets and related information in a single place. It is used to compare and monitor a variety of data simultaneously. The different data views are displayed all at once. Dashboards are shown as tabs at the bottom of the workbook and they usually get updated with the most recent data from the data source. While creating a dashboard, you can add views from any worksheet in the workbook along with many supporting objects such as text areas, web pages, and images.

Each view you add to the dashboard is connected to its corresponding worksheet. So when you modify the worksheet, the dashboard is updated and when you modify the view in the dashboard, the worksheet is updated.

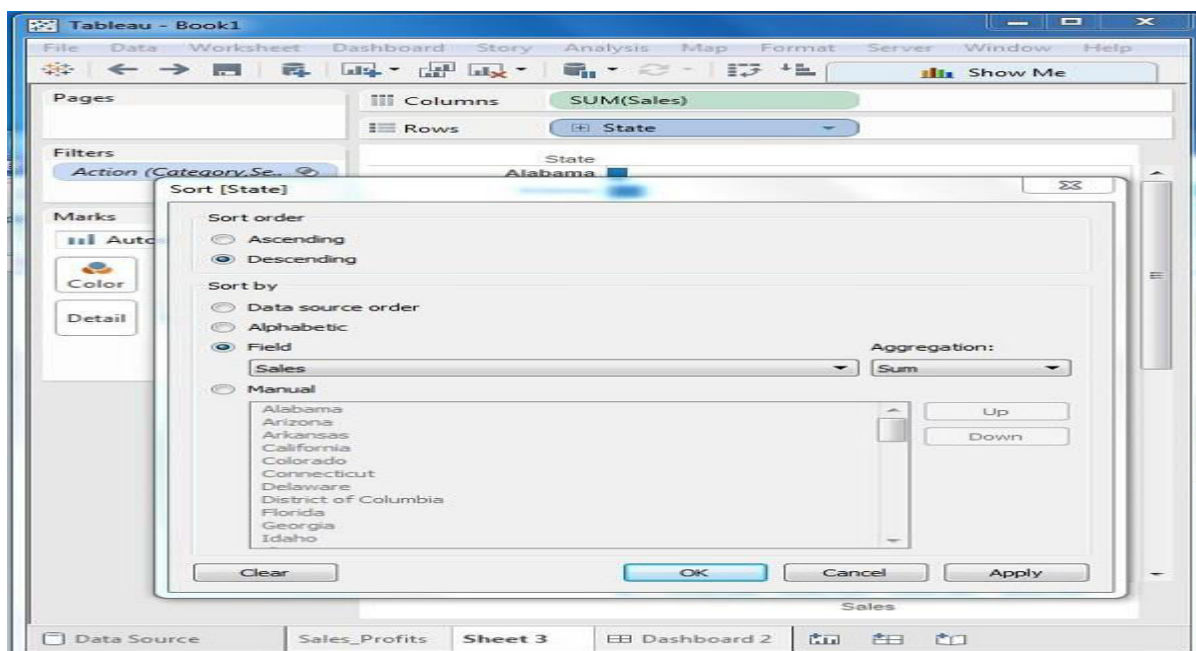
Creating a Dashboard

Using the Sample-superstore, plan to create a dashboard showing the sales and profits for different segments and Sub-Category of products across all the states. To achieve this objective, following are the steps.

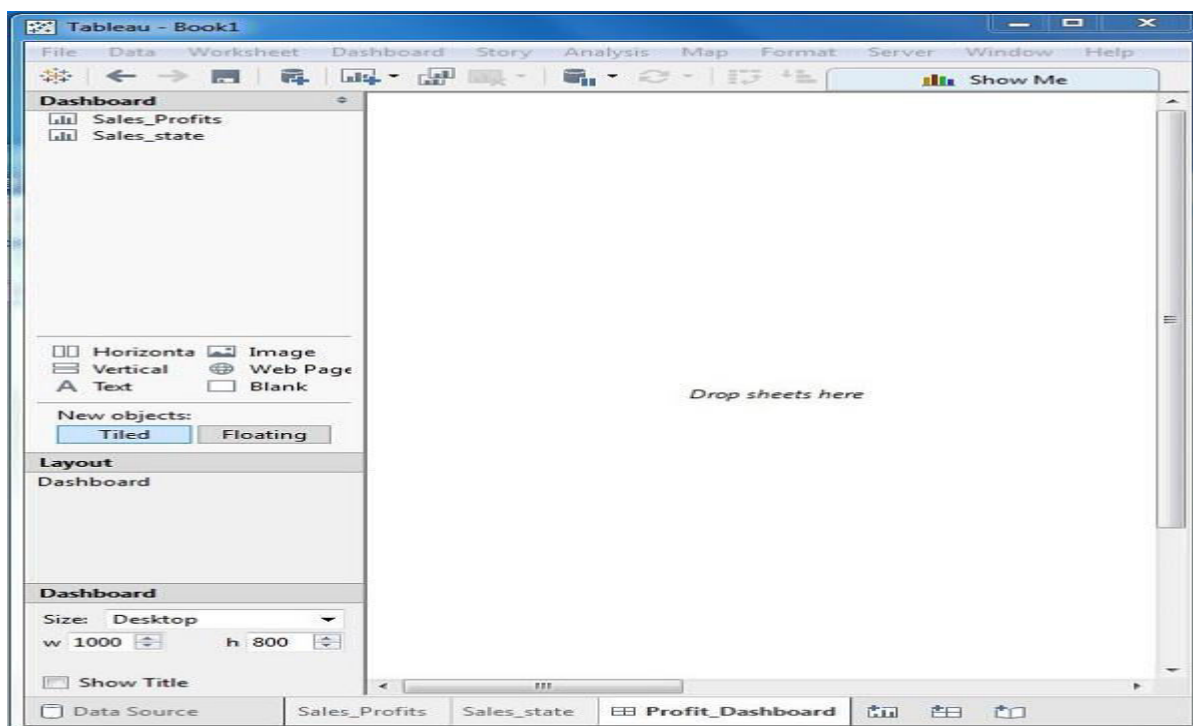
Step 1 – Create a blank worksheet by using the add worksheet icon located at the bottom of the workbook. Drag the dimension Segment to the columns shelf and the dimension Sub-Category to the Rows Shelf. Drag and drop the measure Sales to the Color shelf and the measure Profit to the Size shelf. This worksheet is referred as the Master worksheet. Right-click and rename this worksheet as **Sales_Profits**. The following chart appears.



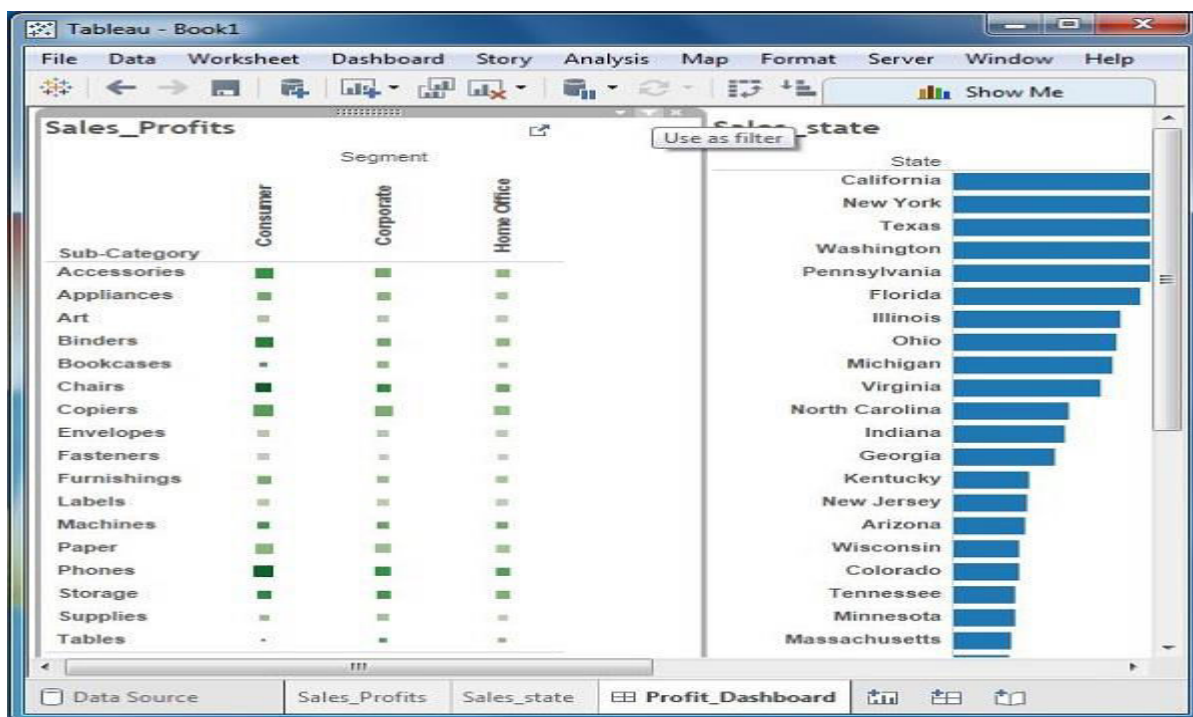
Step 2 – Create another sheet to hold the details of the Sales across the States. For this, drag the dimension State to the Rows shelf and the measure Sales to the Columns shelf as shown in the following screenshot. Next, apply a filter to the State field to arrange the Sales in a descending order. Right-click and rename this worksheet as **Sales_state**.



Step 3 – Next, create a blank dashboard by clicking the Create New Dashboard link at the bottom of the workbook. Right-click and rename the dashboard as Profit_Dashboard.

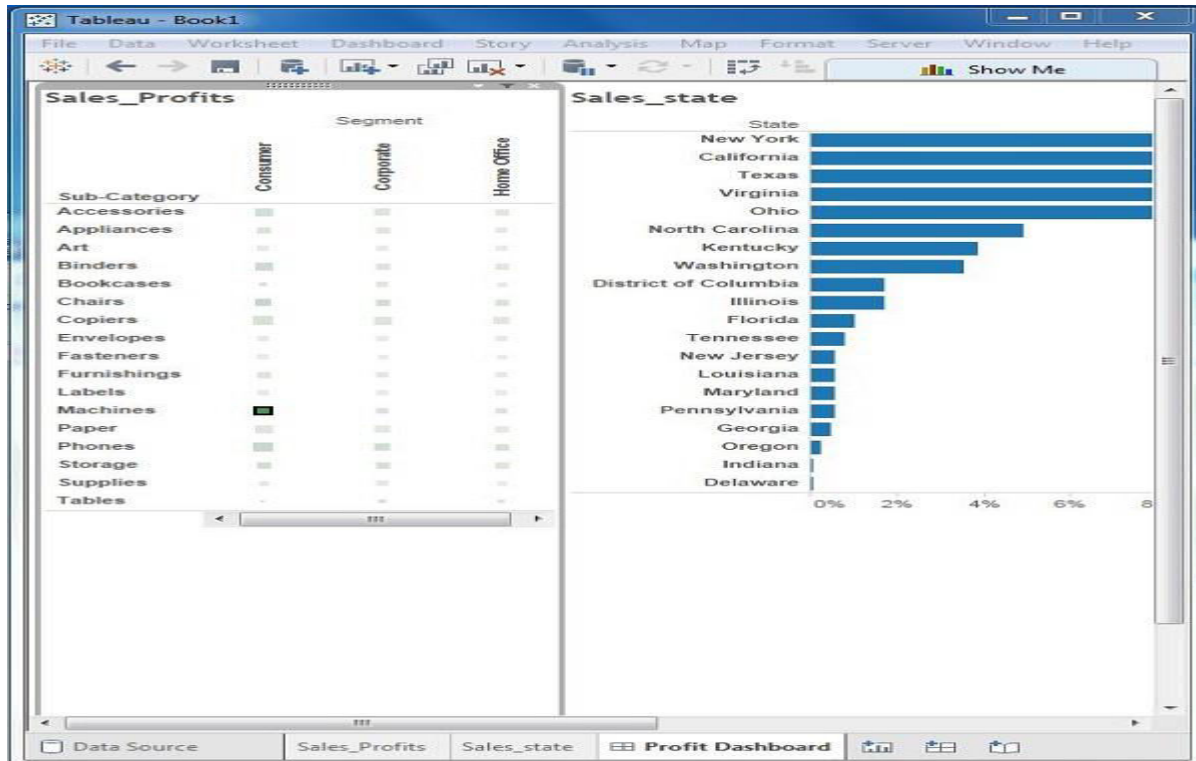


Step 4 – Drag the two worksheets to the dashboard. Near the top border line of Sales Profit worksheet, you can see three small icons. Click the middle one, which shows the prompt Use as Filter on hovering the mouse over it.



Step 5 – Now in the dashboard, click the box representing Sub-Category named Machines and segment named Consumer.

You can notice that only the states where the sales happened for this amount of profit are filtered out in the right pane named **Sales_state**. This illustrates how the sheets are linked in a dashboard.



Data Story:

Create a Story

Version: 2021.3

Applies to: Tableau Desktop, Tableau Online, Tableau Server

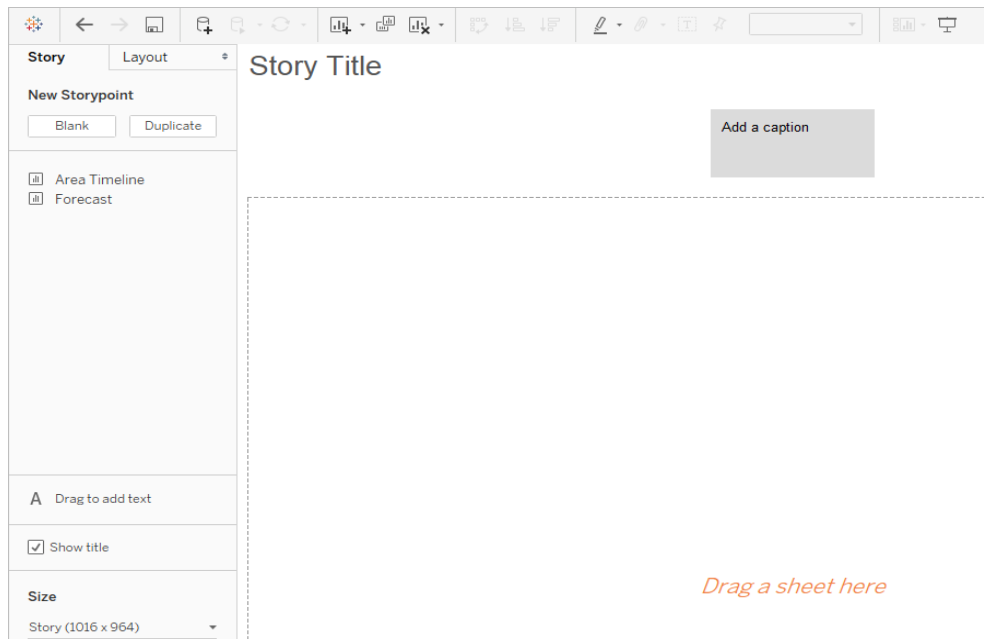
Use stories to make your case more compelling by showing how facts are connected, and how decisions relate to outcomes. You can then publish your story to the web, or present it to an audience.

Each story point can be based on a different view or dashboard, or the entire story can be based on the same visualization seen at different stages, with different filters and annotations.

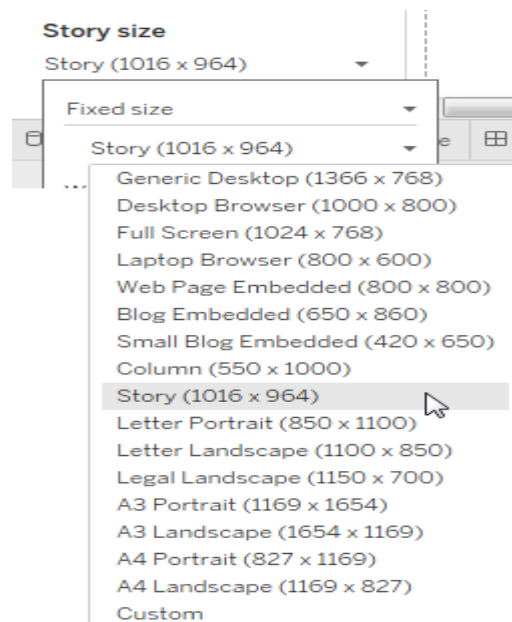
Create a story point

Click the **New Story** tab.

Tableau opens a new story as your starting point:



In the lower-left corner of the screen, choose a size for your story. Choose from one of the predefined sizes, or set a custom size, in pixels:



By default, your story gets its title from the sheet name. To edit it, right-click the sheet tab, and choose **Rename Sheet**.

If you're using Tableau Desktop, you can also rename a story by double-clicking the title.

To start building your story, double-click a sheet on the left to add it to a story point.

In Tableau Desktop, you can also drag sheets into your story point.



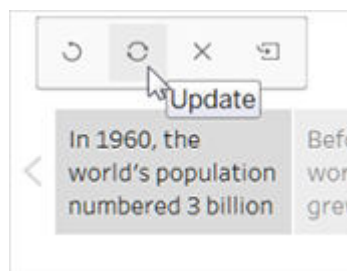
When you add a sheet to a story point, that sheet remains connected to the original sheet. If you modify the original sheet, your changes will automatically be reflected on the story points that use it.

If you are using Tableau Server or Tableau Online to author on the web and the original sheet has **Pause Auto Updates** enabled, the story sheet will be blank until auto-updates are resumed.

Click **Add a caption** to summarize the story point.

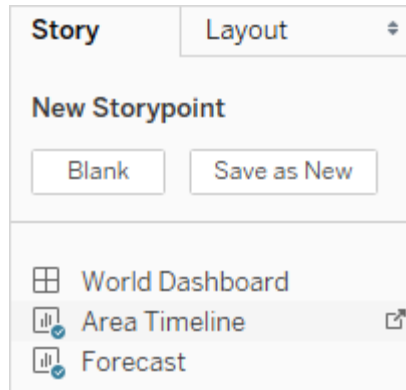
In Tableau Desktop, you can highlight a key takeaway for your viewers by dragging a text object to the story worksheet and typing a comment.

To further highlight the main idea of this story point, you can change a filter or sort on a field in the view. Then save your changes by clicking **Update** on the story toolbar above the navigator box:

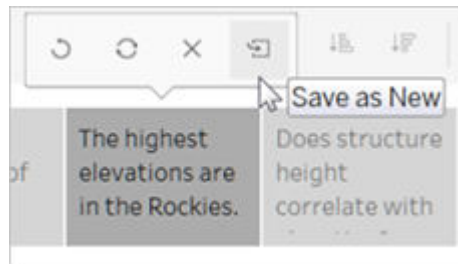


1. Add another story point by doing one of the following:

Click **Blank** to use a fresh sheet for the next story point.



Start customizing a story point and click **Save as New** on the toolbar above the navigator box



Click **Duplicate** to use the current story point as the basis for a new one.

Explore layout options

You can refine the look of your story using the options on the **Layout** tab.

Click the **Layout** tab.

Choose a navigator style that best suits your story, and show or hide the next and previous arrows.



Format a story

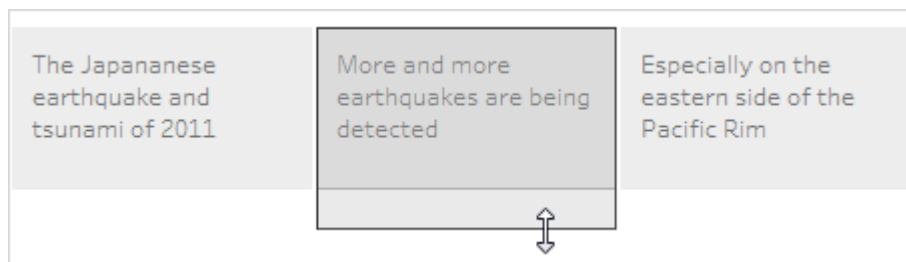
Resize captions (Tableau Desktop only)

Sometimes the text in one or more of your captions is too long to fit inside the height of the navigator. In this case, you can re-size the captions vertically and horizontally.

In the navigator, select a caption.

Drag the border left or right to resize the caption horizontally, down to resize vertically, or select a corner and drag diagonally to resize the caption both horizontally and vertically.

All captions in the navigator update to the new size.

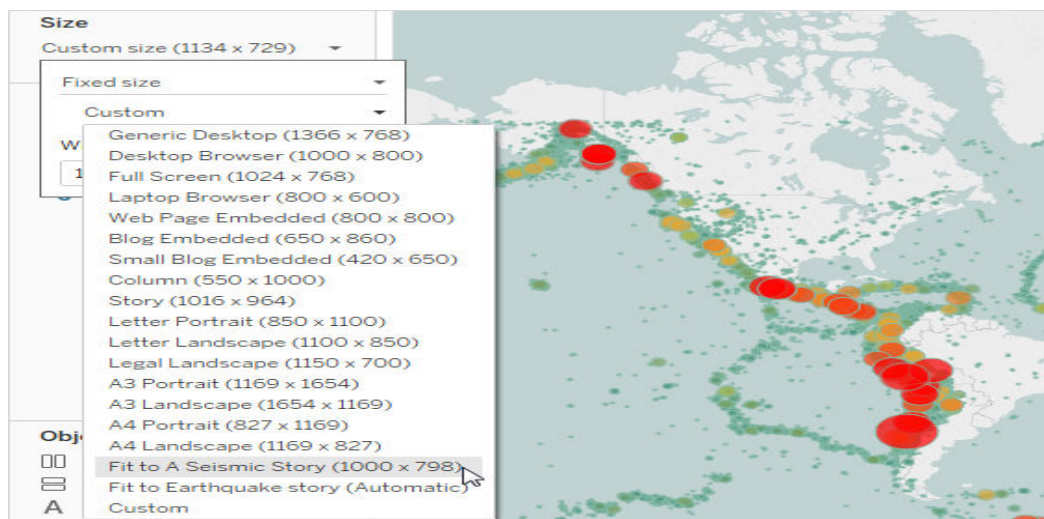


When you resize a caption, you can only select the left, right, or bottom border of the caption.

Fit a dashboard to a story

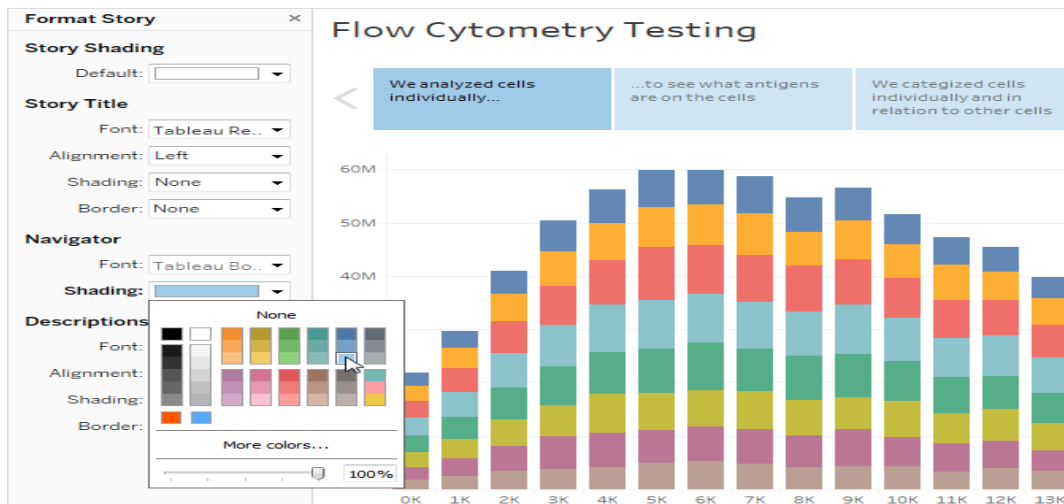
You can fit a dashboard to the exact size of a story. For example, if your story is exactly 800 by 600 pixels, you can shrink or expand a dashboard to fit inside that space.

Click the **Size** drop-down menu and select the story you want the dashboard to fit inside.



Format a story's shading, title, and text objects (Tableau Desktop only)

To open the **Format Story** pane, select **Format > Story**.

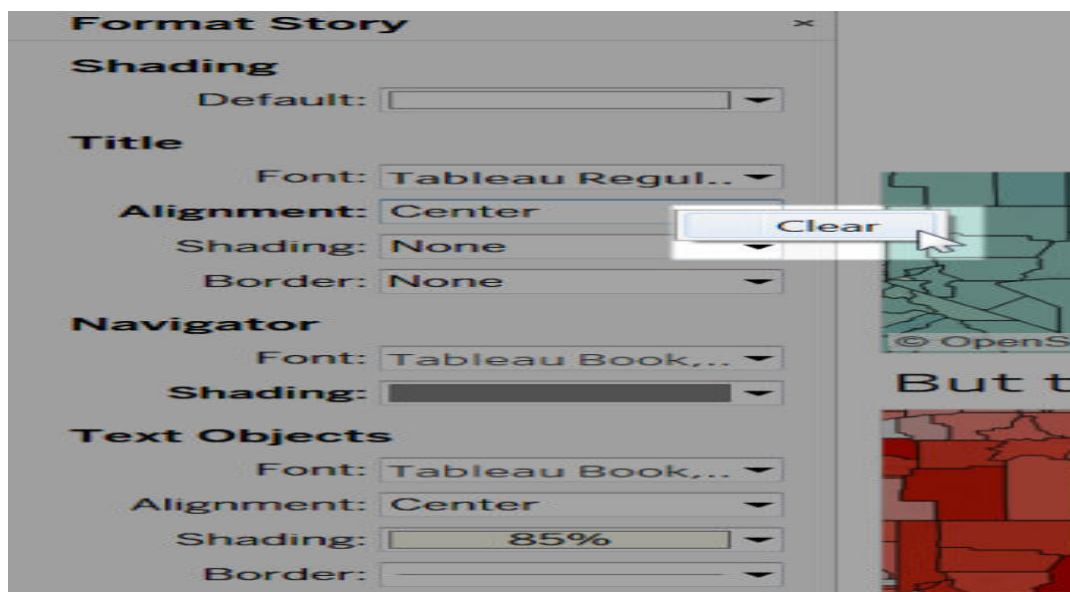


Clear all formatting (Tableau Desktop only)

To reset a story to its default format settings, click the **Clear** button at the bottom of the **Format Story** pane.

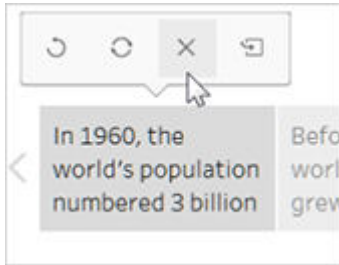
To clear a single format setting, right-click (Windows) or control-click (macOS) the format setting you want to undo in the **Format Story** pane. Then select **Clear**.

For example, if you want to clear the alignment of the story title, right-click (control-click on Mac) **Alignment** in the **Title** section, and then select **Clear**.



Delete a story point

Click the X in the toolbar above the point's caption:



Present your story

In Tableau Desktop, click the **Presentation Mode** button on the toolbar. Or, publish the story to Tableau Online or Tableau Server, and click the **Full Screen** button in the upper-right corner of the browser.

To step through your story, click the arrow to the right of the story points. Or, in Tableau Desktop, use the arrow keys on your keyboard.

To exit Presentation or Full Screen mode, press **Esc**.

Map Based Visualization:

Building a Simple Tableau Custom Map

To build a Simple Tableau Custom Map, you need a data source, like Sample-Superstore data, consisting of location data, location names, latitude, longitude coordinates. Without geographical coordinates, you can't prepare map visualization in Tableau. A simple Tableau Custom Map can be created by implementing the following steps:

- **Step 1:** Open Tableau Desktop, connect to the Sample-Superstore data source, and the worksheet will open in front of your screen.

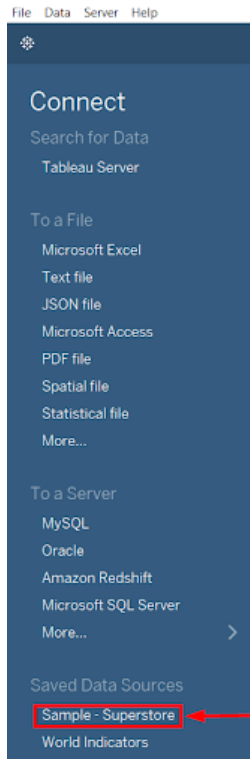
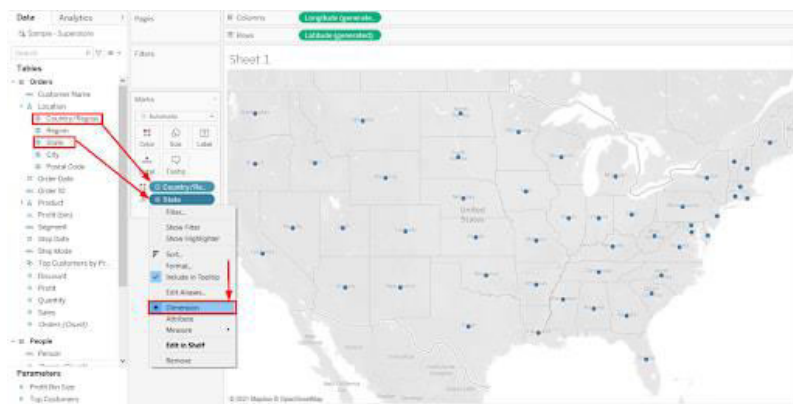


Image Source: <https://www.tableau.com/>

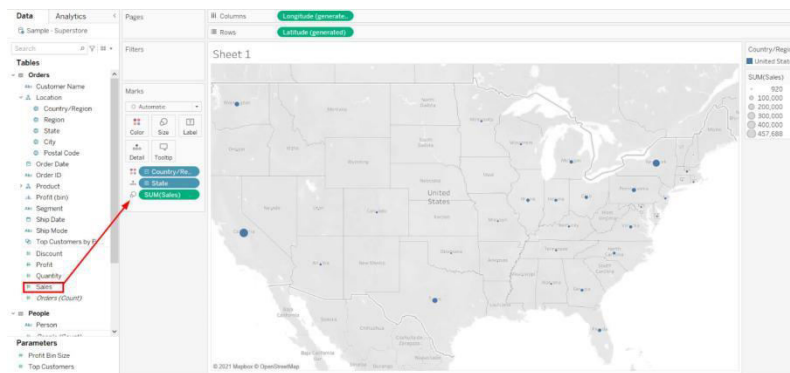
- **Step 2:** Move your cursor to the **Location** option, select the **Country & State** parameter under it, drag both of them to the **worksheet**.
- **Step 3:** Right-click over the **State** and select the **Dimension** from the drop-down list.



Image

Source: <https://www.tableau.com/>

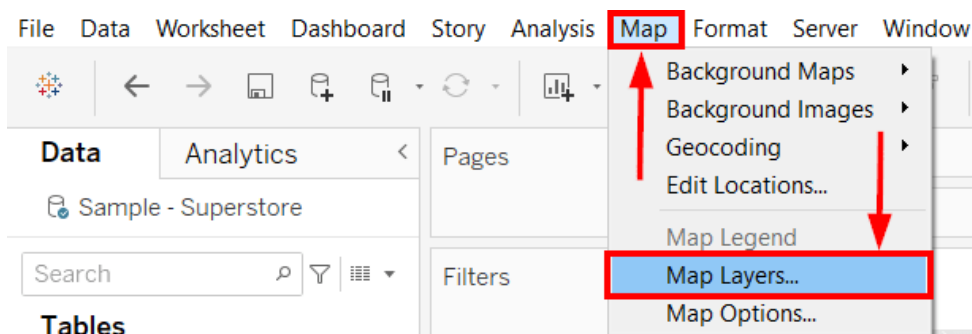
- **Step 4:** Drag the **Sales** table to the **Mark** sheet as shown below.



Image

Source: <https://www.tableau.com/>

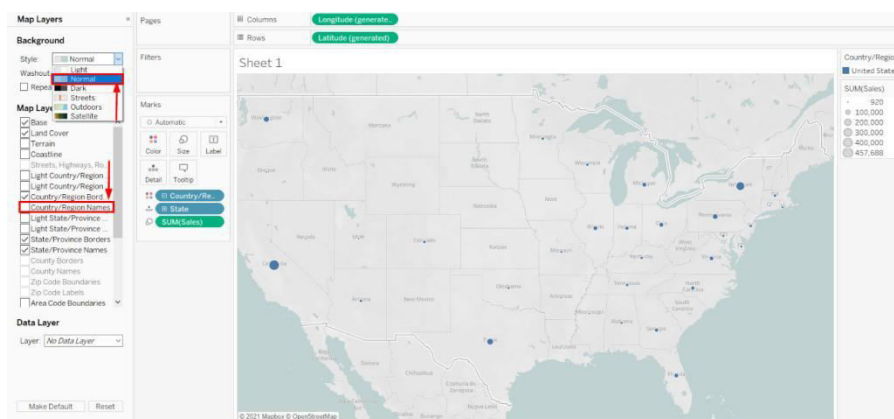
- **Step 5:** Move your cursor to the **Maps** tab and choose **Map Layers** from the drop-down list.



Image

Source: <https://www.tableau.com/>

- **Step 6:** On the Map Layers pane, unselect the **Country/Region Names** and select **Normal** from the drop-down list of style.



Image

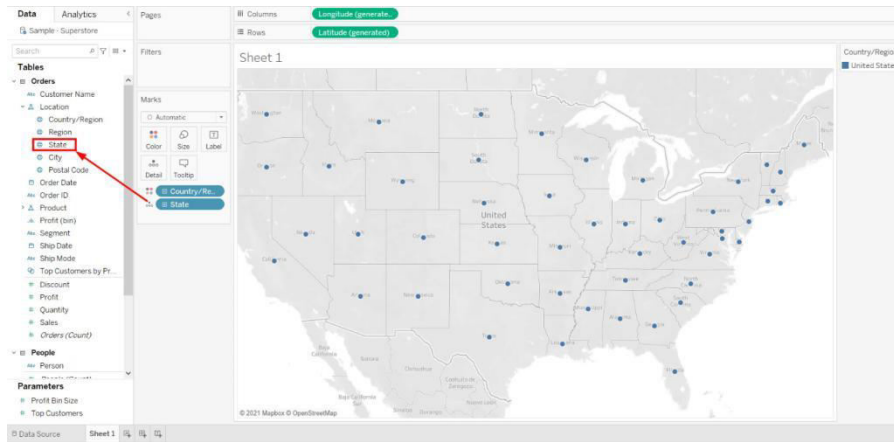
Source: <https://www.tableau.com/>

This is how you can design simple Tableau Custom Maps to perform a basic geographical analysis of your data.

Creating a Polygon Tableau Custom Map

A simple Polygon or Filled Tableau Custom Map helps you understand the basic mapping concepts in Tableau. The following steps can be implemented to create a Polygon Tableau Custom Map:

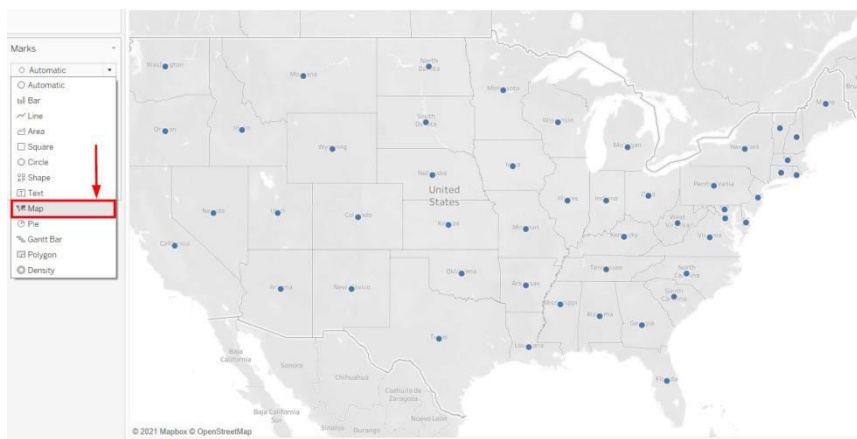
- **Step 1:** Drag the **State** parameter from the **Data** pane to the worksheet.



Image

Source: <https://www.tableau.com/>

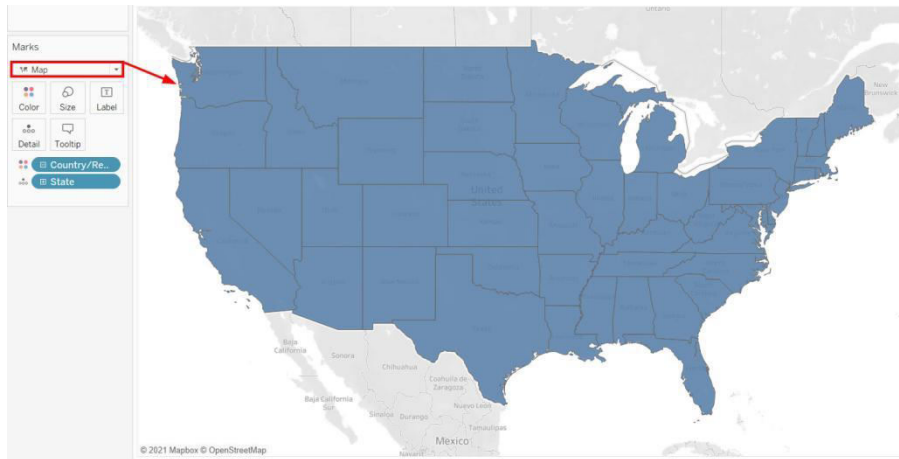
- **Step 2:** Move your cursor to the **Marks** pane, choose the **Map** option from the drop-down list.



Image

Source: <https://www.tableau.com/>

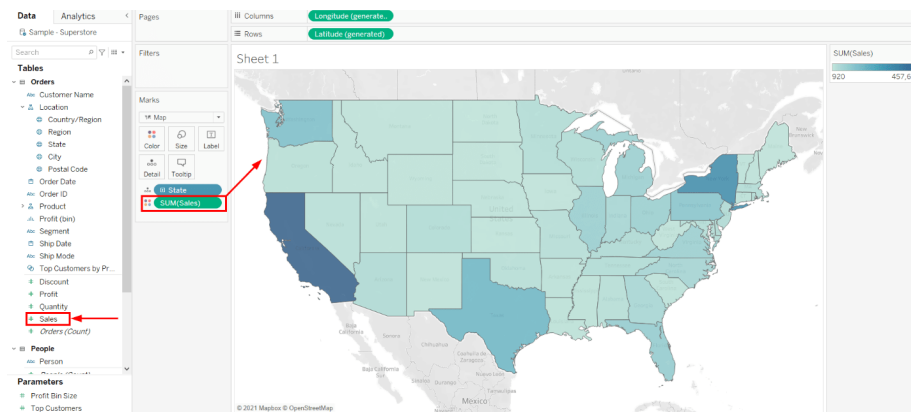
- **Step 3:** The worksheet will show the following visualization of the map.



Image

Source: <https://www.tableau.com/>

- **Step 4:** Select the **Sales** parameter from the **Data** pane and place it over the map.



Image

Source: <https://www.tableau.com/>

You can see the **Number of Sales** from the polygons in the updated map.

Building Different Types of Tableau Custom Maps

Tableau Desktop enables you to design the following types of maps depending upon your business needs and visualization requirements. The different kind of custom maps that can be created on Tableau are as follows:

- Choropleth or Filled Tableau Custom Maps
- Flow or Path Tableau Custom Maps
- Proportional Symbol Tableau Custom Maps
- Point Distribution Tableau Custom Maps
- Heat or Density Tableau Custom Maps
- Spider or Origin-Destination Tableau Custom Maps

1) Choropleth or Filled Tableau Custom Maps

Choropleth Maps, also known as Filled Maps, are perfect to represent ratio and aggregated data. Users can use ratio or aggregated data for polygons that can be related to locations, such as countries, regions, states, or any area.

For simplicity, download Tableau [Example Workbook](#) and implement the following steps to show ratio or combined data in Tableau Desktop:

- **Step 1:** Go to **Worksheet Tab** and select the **New Worksheet** from the drop-down list or press CTRL+M. A New Sheet will open.

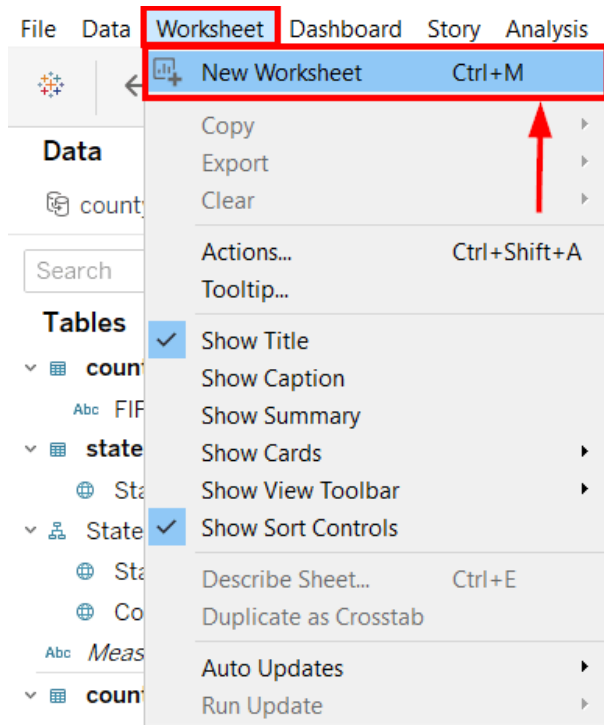
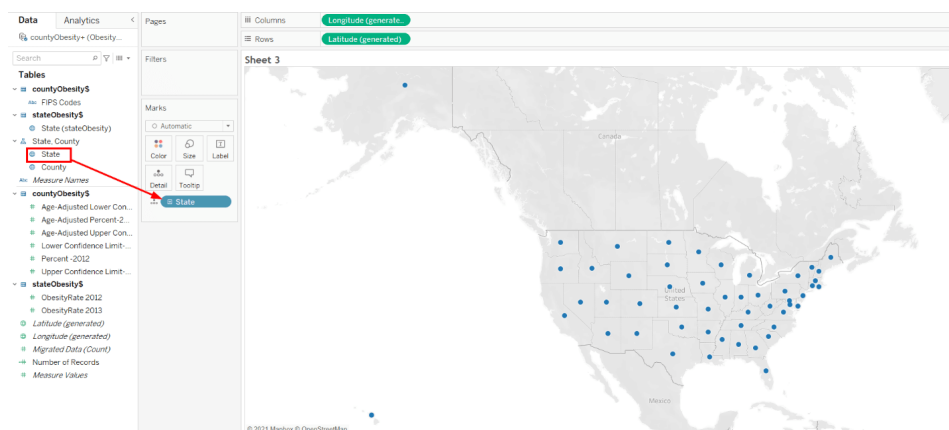


Image Source: <https://www.tableau.com/>

- **Step 2:** Under the State, **Country** dimension, select the **State** option and move to the sheet.



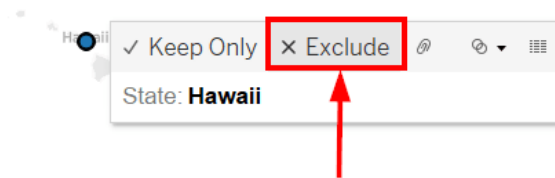
Image

Source: <https://www.tableau.com/>

- **Step 3:** Now select the **Datapoint** of **Alaska, United States**.
- **Step 4:** Click over it and choose the **Exclude** option.
- **Step 5:** Repeat the same steps for **Hawaii**.



Image

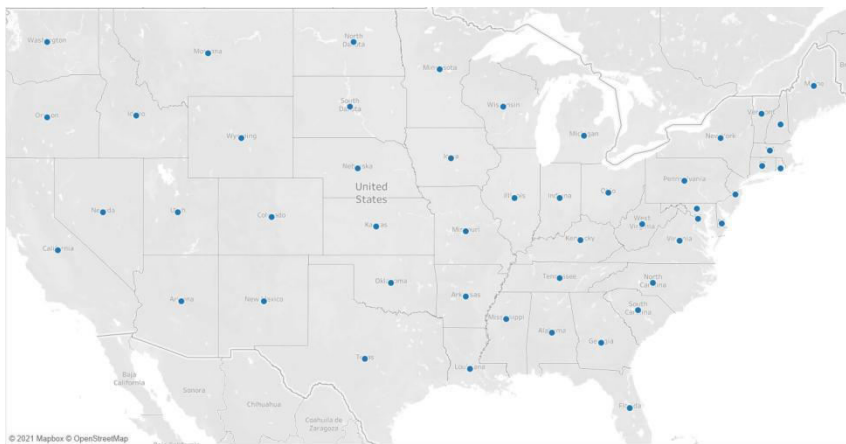


Image

Source: <https://www.tableau.com/>

Source: <https://www.tableau.com/>

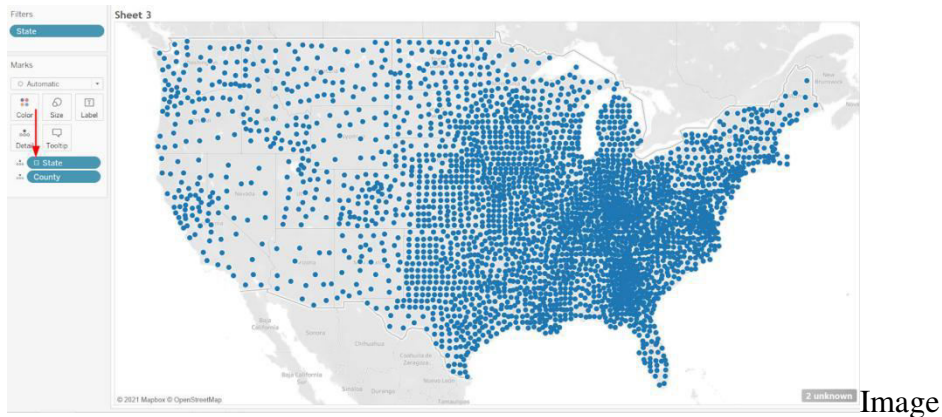
You'll see a better version of the map.



Image

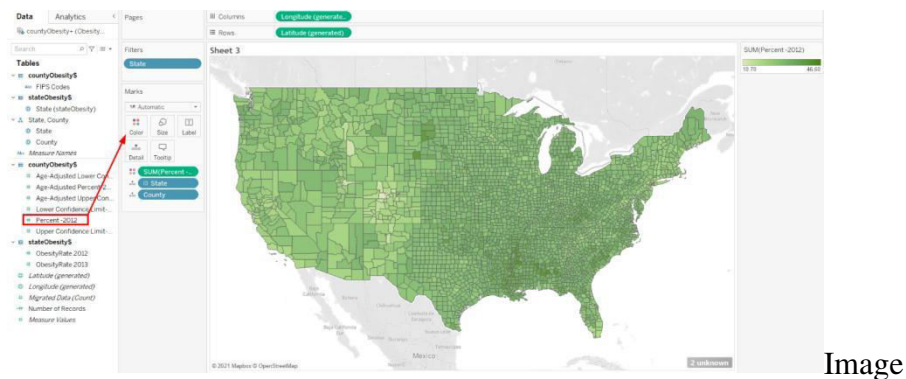
Source: <https://www.tableau.com/>

- **Step 6:** Click on the plus symbol of **State** to get details of all countries, as shown below.



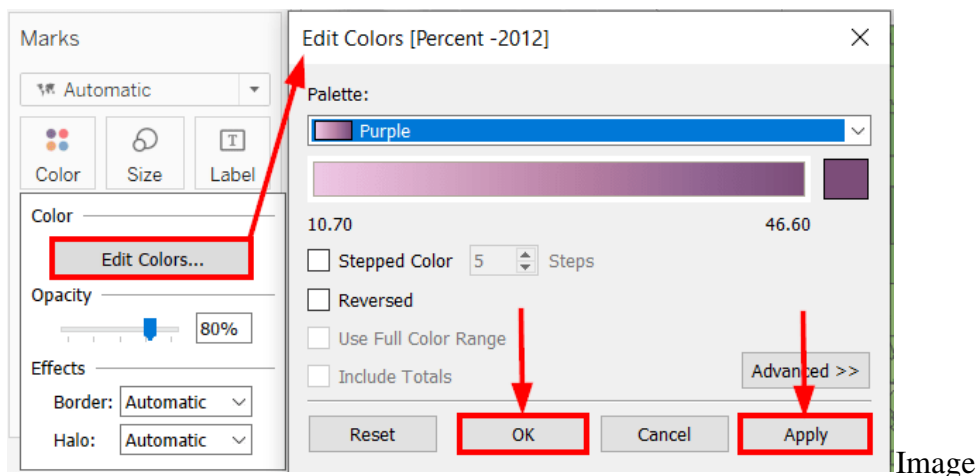
Source: <https://www.tableau.com/>

- **Step 7:** Place your cursor over **Percent- 2012** and drop it over the color option.



Source: <https://www.tableau.com/>

- **Step 8:** To get the obesity percentage range, select the **Edit Colors** option from the **Color** icon.
- **Step 9:** Select the **Purple** color and click **Apply**.



Source: <https://www.tableau.com/>

- **Step 10:** Open the **Effect** option from the Color icon.

- **Step 11:** Select the None option from the **Border** list.

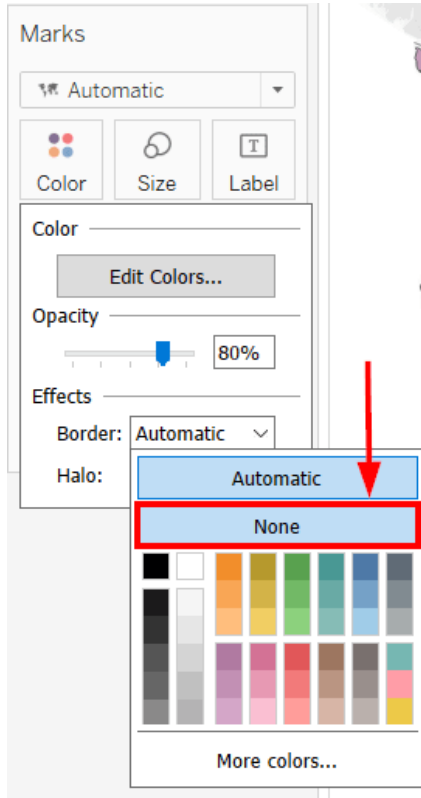
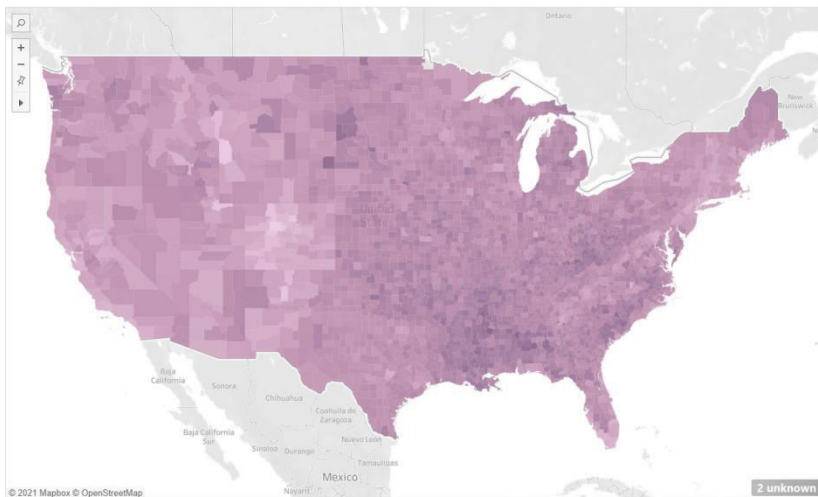


Image Source: <https://www.tableau.com/>

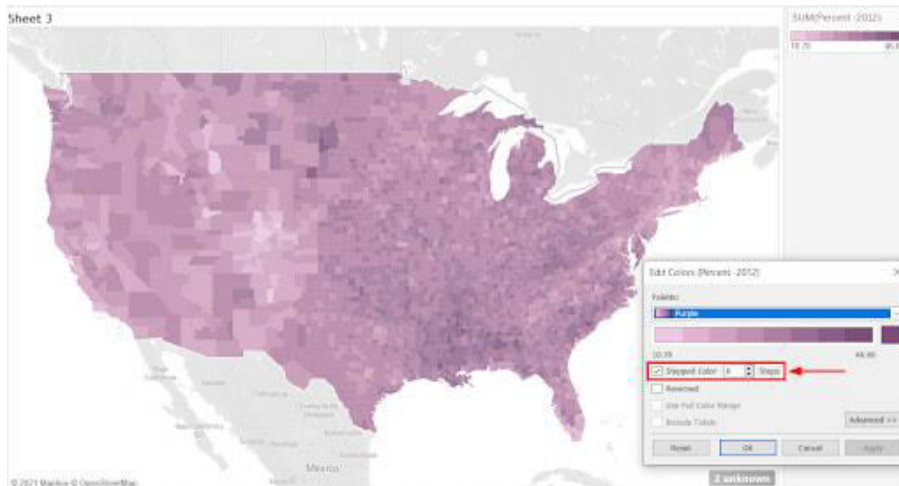
The output of the Choropleth will appear in front of your screen.



Image

Source: <https://www.tableau.com/>

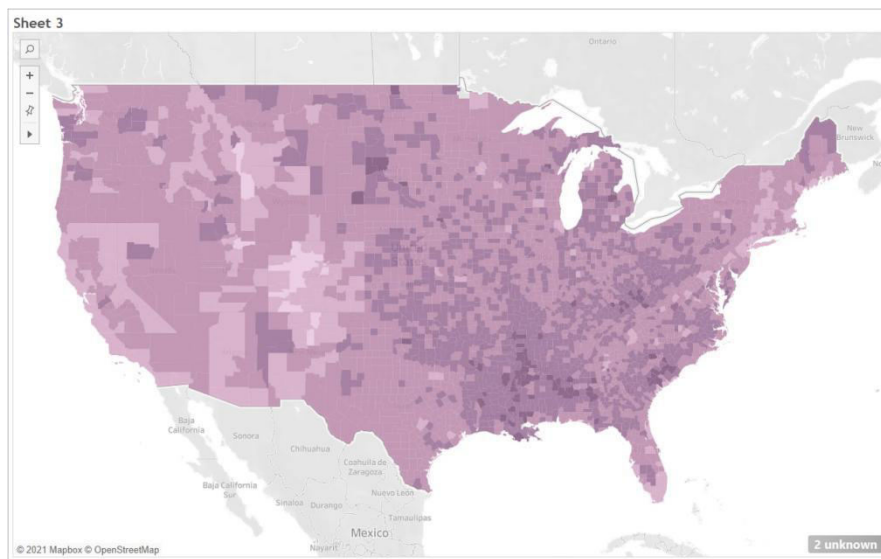
- **Step 12:** In the above map, Tableau selected the default color distribution. However, if you want to change it, go to the **Edit Color** menu again. Mark the **Stepped** color and enter 8. Click on the **Apply** button and close it.



Image

Source: <https://www.tableau.com/>

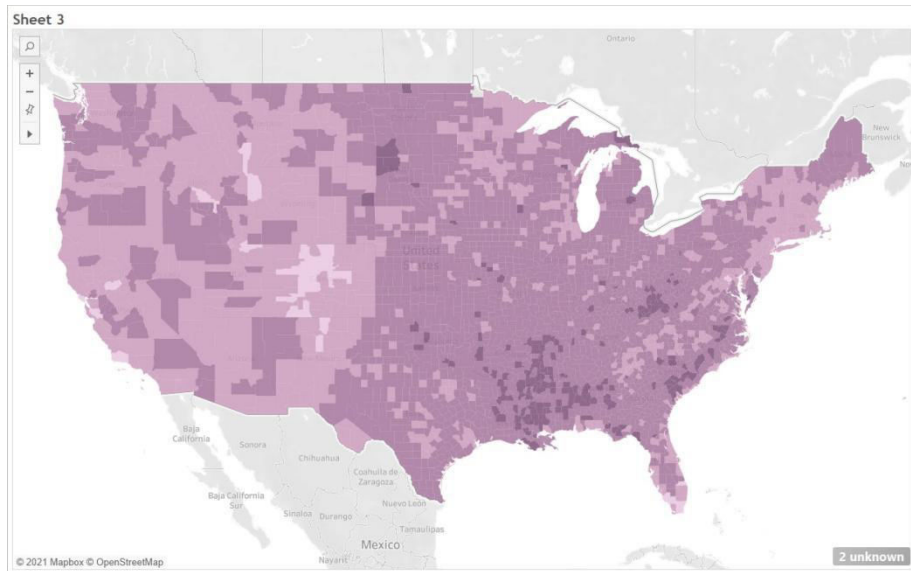
- **Step 13:** Repeat the same steps, but this time select **5**.



Image

Source: <https://www.tableau.com/>

- **Step 14:** The map view for **4 Stepped Color** might look like this.

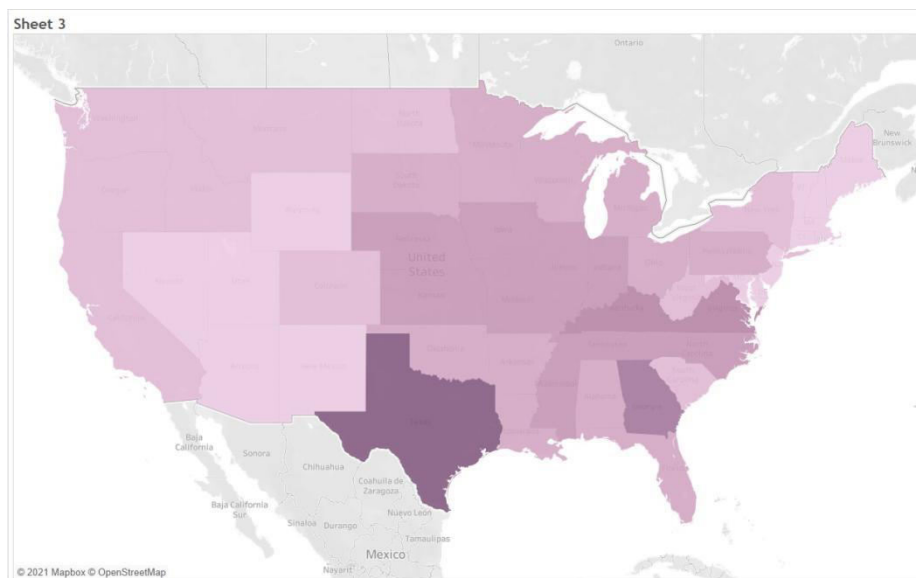


Image

Source: <https://www.tableau.com/>

By selecting 4 instead of 5, you will understand that obesity rates are higher in the South. However, prior maps were showing even distribution of colors or obesity rates in the United States (US). All the maps are great to show aggregate data, yet each is giving you prominent information on obesity rates in different countries.

Similarly, if you want to plot the map for a **State Level** detail, remove the County from the **Marks** chart. You can analyze that Texas has higher obesity rates, followed by Georgia.



Image

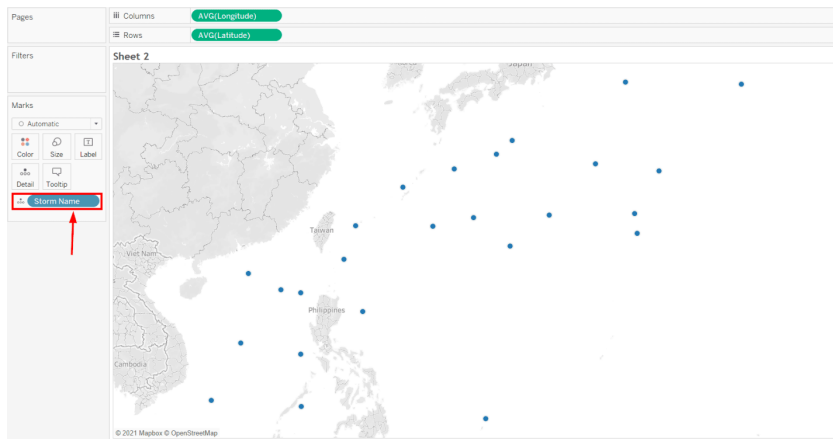
Source: <https://www.tableau.com/>

For more information, see [Choropleth Maps](#) or [Filled Maps](#) in detail.

2) Flow or Path Tableau Custom Maps

Tableau Business Intelligence tool allows users to determine the flow of path over time, e.g., the path of the storm, through Flow or Path Maps. To create Flow Maps in Tableau, download **Example Workbook** and implement the following steps:

- **Step 1:** Open the downloaded example in **Tableau Desktop**.
- **Step 2:** Open a **New Worksheet**.
- **Step 3:** Move **Latitude** and **Longitude** coordinates using the drag-and-drop method.
- **Step 4:** Choose the **Storm Name** and place it over the **Details** icon in the Marks pane.



Image

Source: <https://www.tableau.com/>

- **Step 5:** To apply filters, select the **Date** from the **Data** pane and place it over the **Filters** area.
- **Step 6:** Choose Year, such as 2012.
- **Step 7:** Click on **Apply** and close the screen.

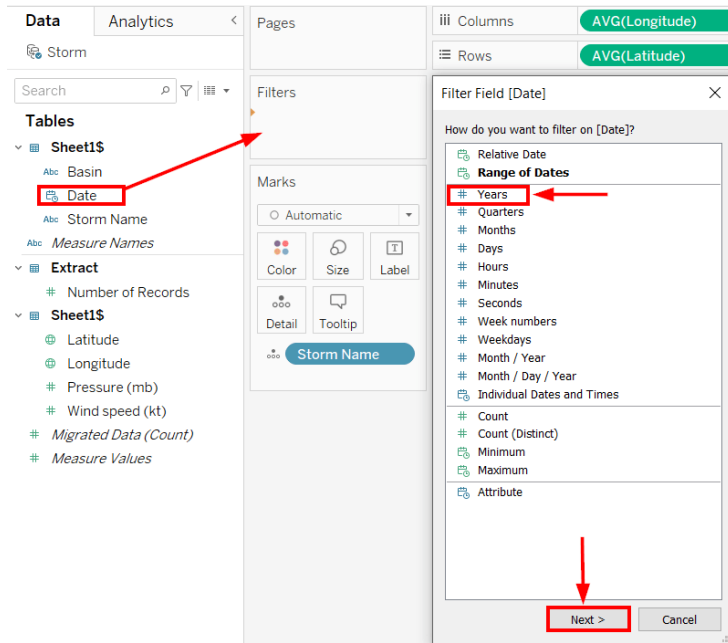


Image Source: <https://www.tableau.com/>

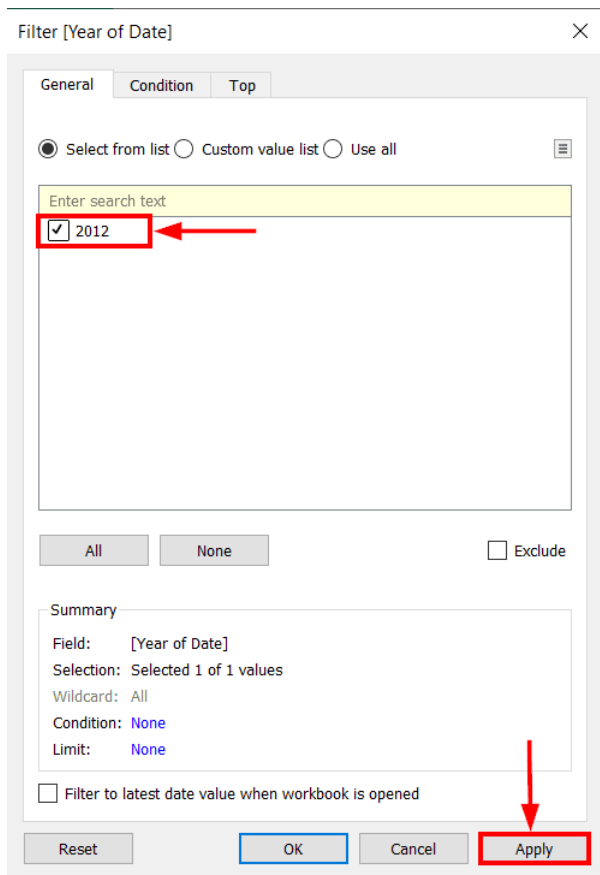
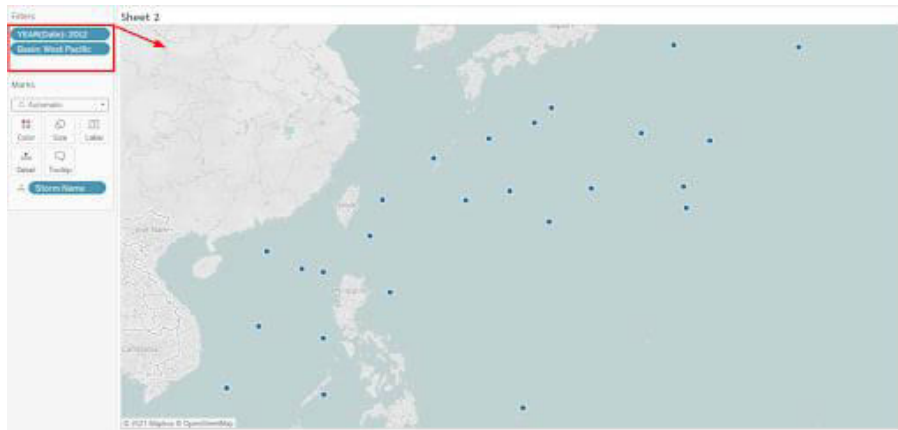


Image Source: <https://www.tableau.com/>

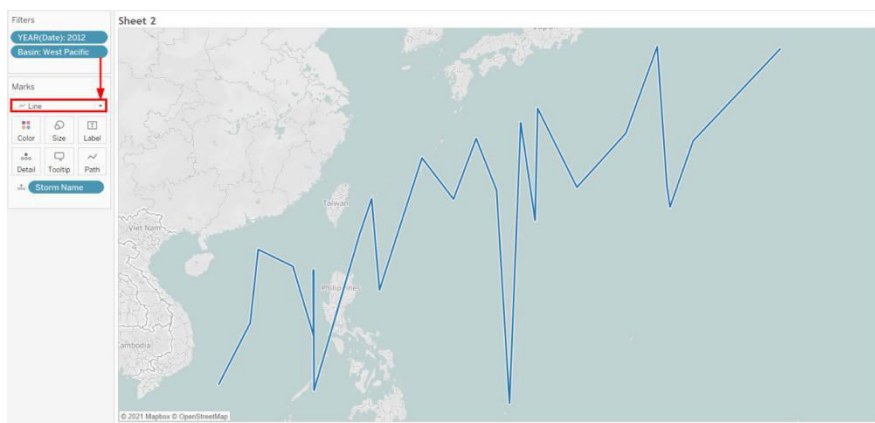
- **Step 8:** Repeat the same steps for **Bhasin** and select the **West Pacific** region.
- **Step 9:** Select **Normal Background** from the **Map** tab. The final map might look like this.



Image

Source: <https://www.tableau.com/>

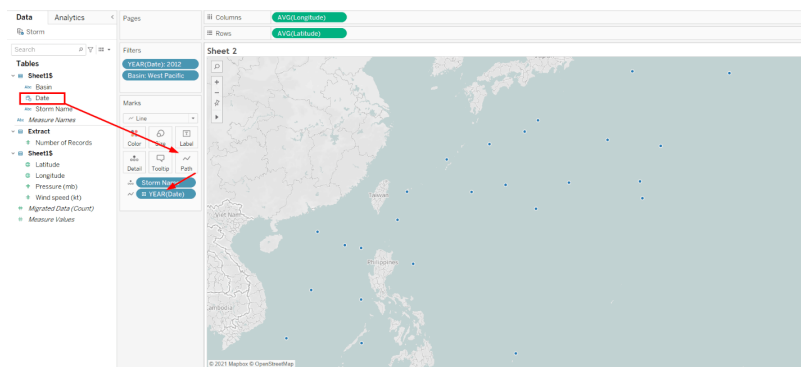
- **Step 10:** Select a **Line** from the **Marks** pane.



Image

Source: <https://www.tableau.com/>

- **Step 11:** Drag the **Date** from the **Data** pane and place it over the **Path** icon. This will remove the line due to the absence of an exact date.



Image

Source: <https://www.tableau.com/>

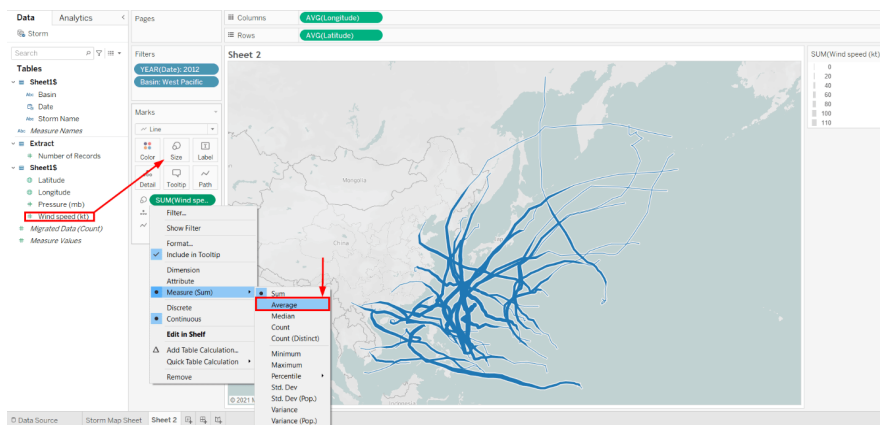
- **Step 12:** Click on the **Date** parameter, select an **Exact Date** option.



Image

Source: <https://www.tableau.com/>

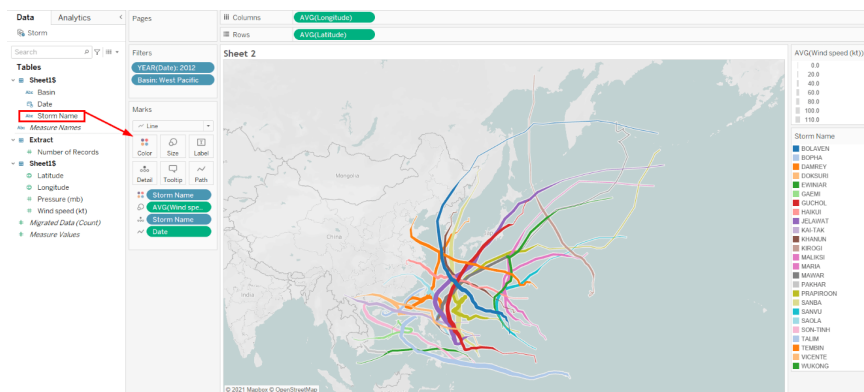
- **Step 13:** Drag the **Wind Speed (kt)** from the Data pane and place it over the **Size** icon.
- **Step 14:** Right-click over the **Wind Speed (kt)** and choose the **Average** parameter.



Image

Source: <https://www.tableau.com/>

- **Step 15:** Place **Storm Name** over the **Colors** icon to complete the **Flow Map**.



Image

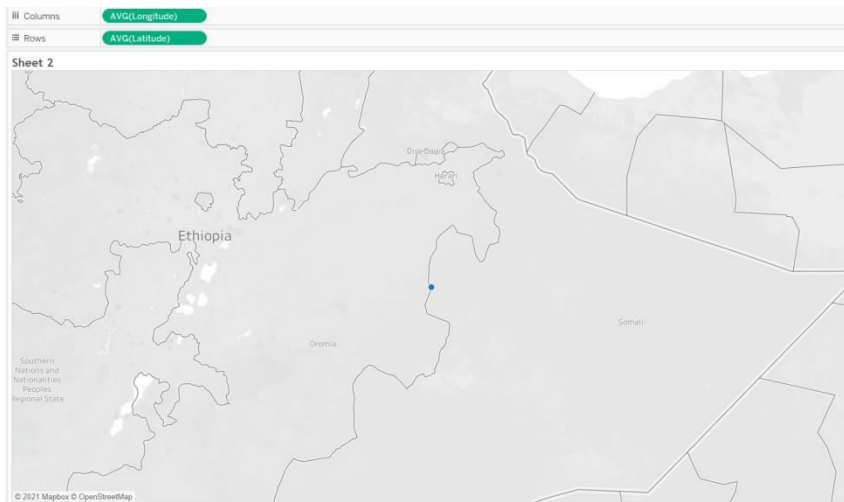
Source: <https://www.tableau.com/>

For more information, see [Flow Maps](#) or [Path Maps](#) in detail.

3) Proportional Symbol Tableau Custom Maps

A Proportional Symbol Map is a great start to acquire quantitative values for individual locations. Nevertheless, your data source should contain quantitative values, longitude and latitude coordinates to create Proportional Symbol Maps. Download [Example Workbook](#) to learn more in detail.

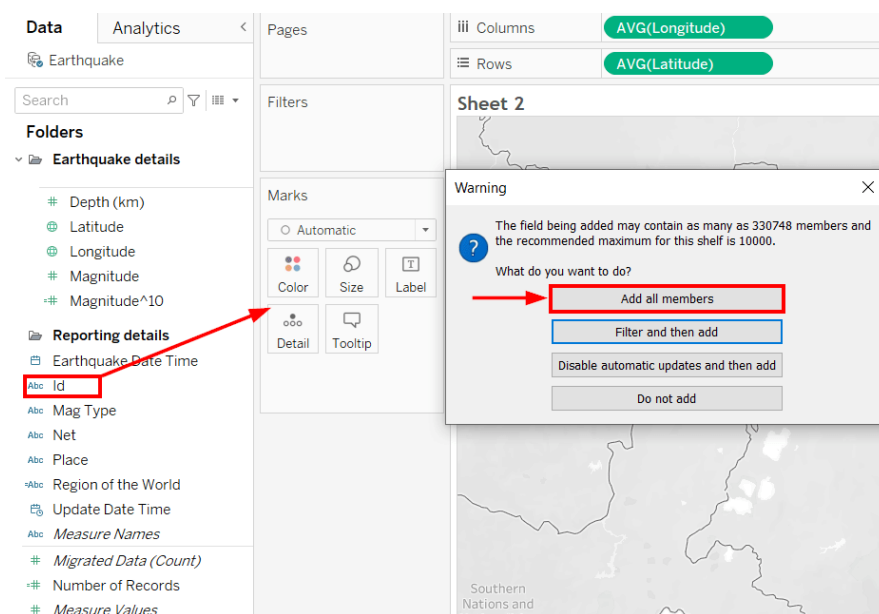
- **Step 1:** Download **Example Workbook**.
- **Step 2:** Open the downloaded file on **Tableau Desktop**.
- **Step 3:** Open a **New Worksheet**.
- **Step 4:** Drag **Latitude** and **Longitude** to the sheet.



Image

Source: <https://www.tableau.com/>

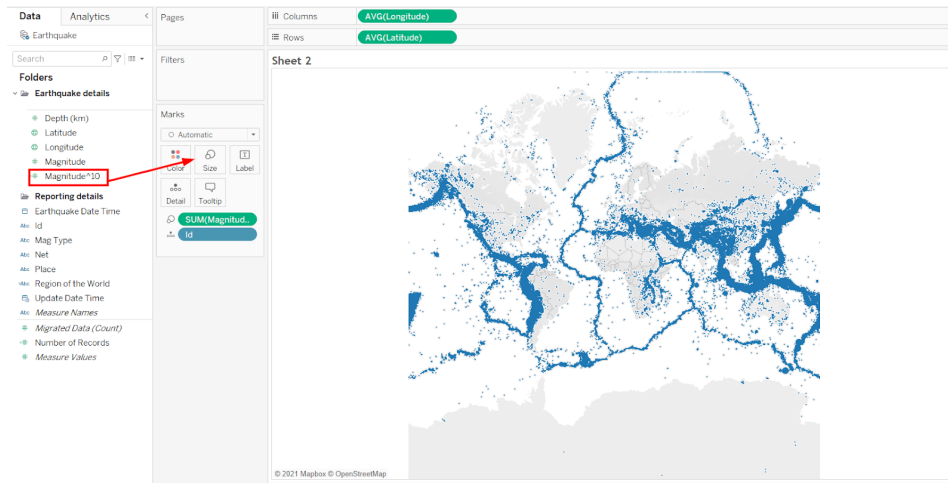
- **Step 5:** Select the **ID** from the **Data** pane and place it over the **Details** icon.
- **Step 6:** A dialog box will appear, select **Add All Members**.



Image

Source: <https://www.tableau.com/>

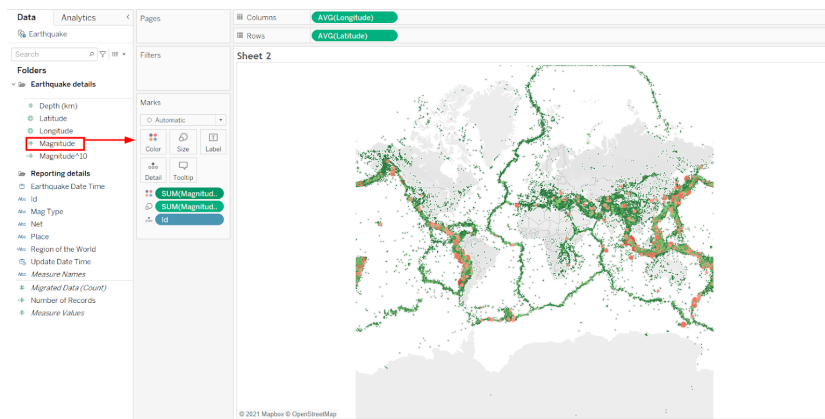
- **Step 7:** Select **Magnitude¹⁰** and place it over **Size** in Marks.



Image

Source: <https://www.tableau.com/>

- **Step 8:** Grab the **Magnitude to Color** option.



Image

Source: <https://www.tableau.com/>

- **Step 9:** Click on **Edit Colors** from the **Color** icon.
- **Step 10:** Select your preferred color, opacity (70%), and Border.
- **Step 11:** Click on **Stepped Color** and choose 8.
- **Step 12:** Click the **Reversed** checkbox.
- **Step 13:** Click on **Advanced**, click **Center**, and enter 7.

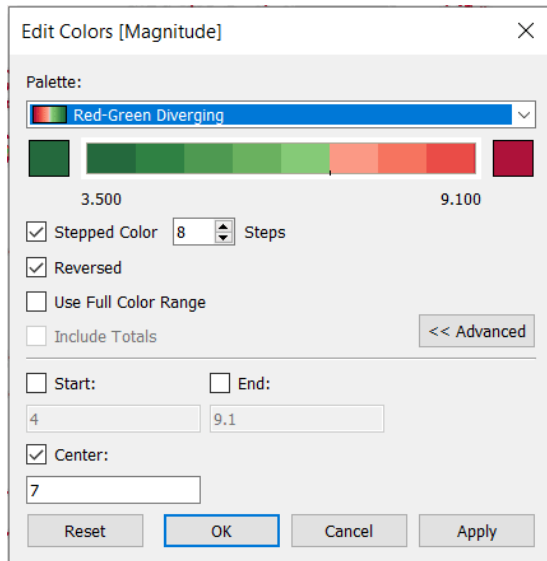


Image Source: <https://www.tableau.com/>

- **Step 14:** Right-click over the **ID** and select the **Sort** option.
- **Step 15:** Choose **Descending** for **Sort Order** option.

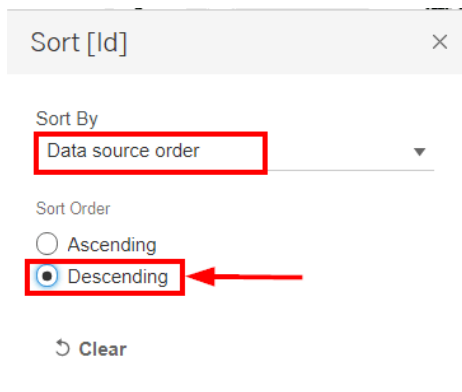


Image Source: <https://www.tableau.com/>

- **Step 16:** Choose **Fields** for **Sort By** option and then choose **Magnitude**.
- **Step 17:** Click **OK**.

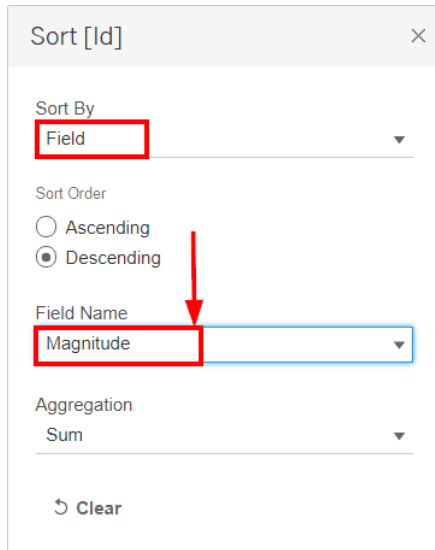
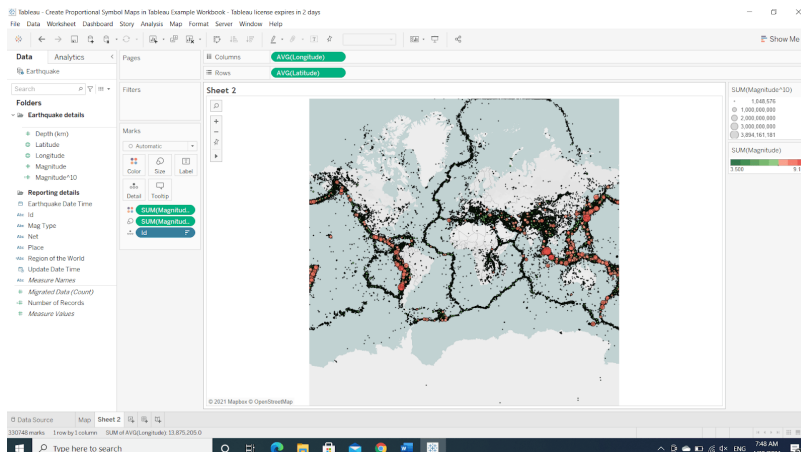


Image Source: <https://www.tableau.com/>

- **Step 18:** Select any appropriate background for your map. The final image of the map might look like this.



Image

Source: <https://www.tableau.com/>

To explore more, see [Proportional Symbol Maps](#) in detail.

4) Point Distribution Tableau Custom Maps

Everyone loves to spot visual clusters, and Tableau Desktop makes this process super easy through Point Distribution Maps. But your data source should have longitude and latitude coordinates to map visual clusters. Download the [Example Workbook](#) right now and implement the following steps:

- **Step 1:** Open Tableau Desktop and select a **New Worksheet**.
- **Step 2:** Click on the **Latitude** dimension.
- **Step 3:** Select the **Geographic Role**.
- **Step 4:** Choose the **Latitude** parameter.

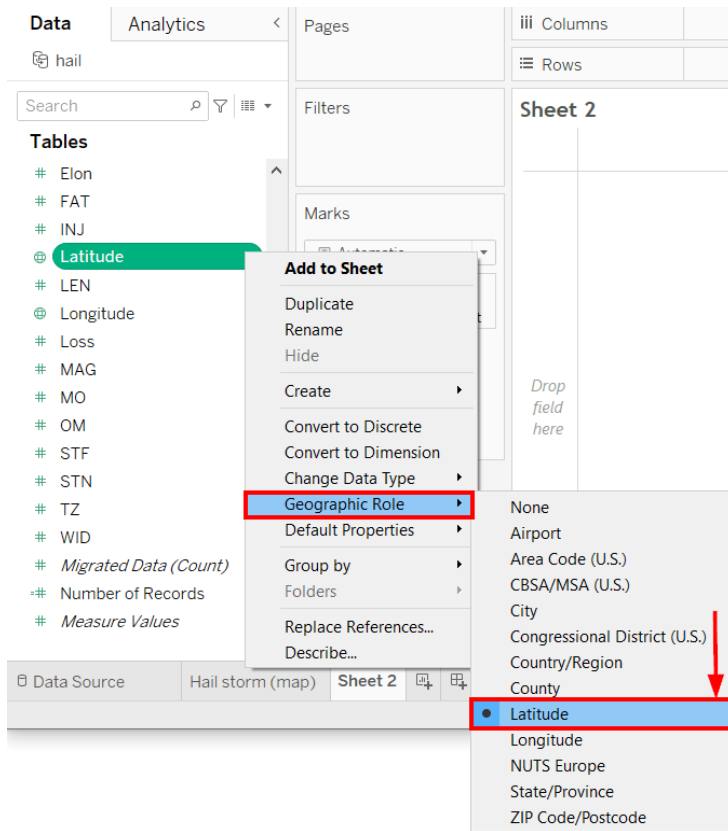
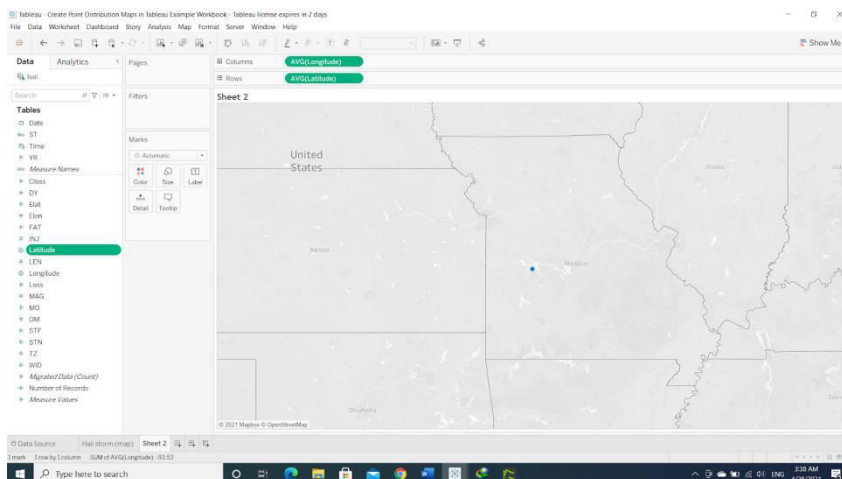


Image Source: <https://www.tableau.com/>

- **Step 5:** Click on the **Longitude** dimension.
- **Step 6:** Select the **Geographic role**.
- **Step 7:** Choose the **Longitude** parameter.
- **Step 8:** Double-click over both **Longitude** and **Latitude** dimensions.



Image

Source: <https://www.tableau.com/>

- **Step 9:** Click on the **AVG(Longitude)** and select **Dimension**.
- **Step 10:** Click on the **AVG(Latitude)** and select **Dimension**.

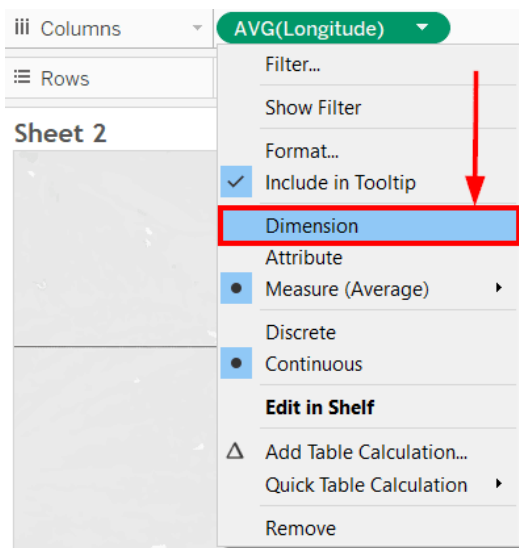


Image Source: <https://www.tableau.com/>

The map might look like something as follows:

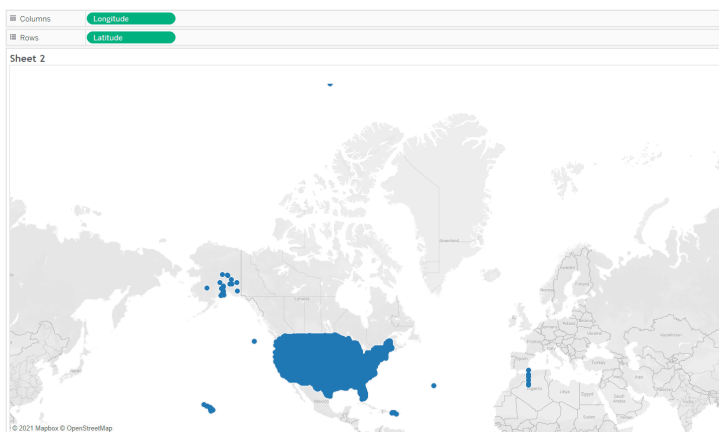


Image Source: <https://www.tableau.com/>

To make it clearer, move your cursor to the **Marks** card.

- **Step 11:** Click on the **Size**.
- **Step 12:** Move the bar to the left edge.

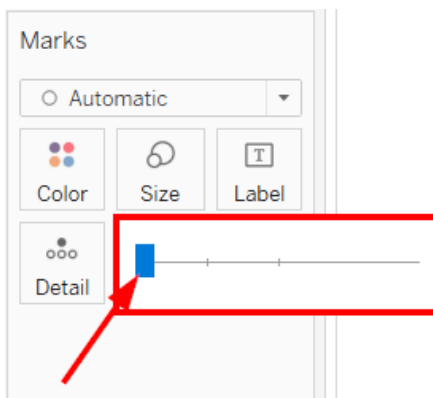


Image Source: <https://www.tableau.com/>

- **Step 13:** Zoom the graph to get the entire picture.

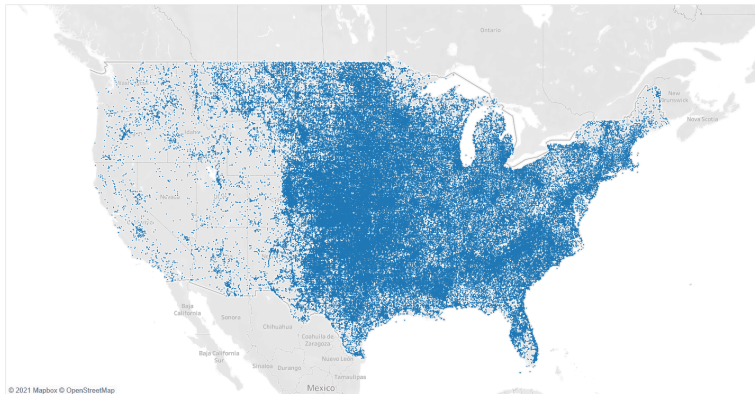


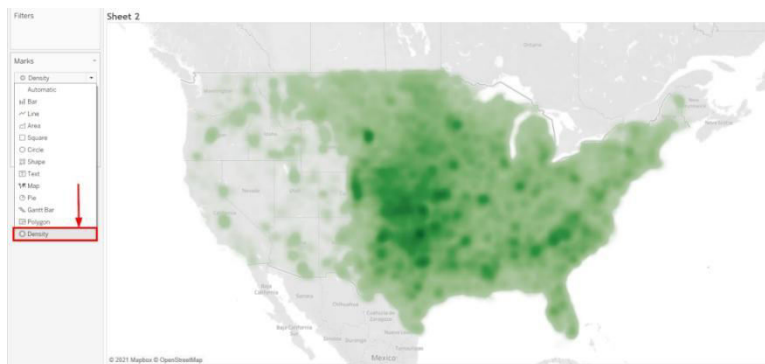
Image Source: <https://www.tableau.com/>

To learn more about point distribution maps, see [Visual Clusters of Data](#).

5) Heat or Density Tableau Custom Maps

Heat Maps, also known as Density Maps, show you trends, uncover patterns and unveil relative concentrations. Take the above Point Distribution Map, for instance and implement the following steps:

- **Step 1:** Go to **Marks** pane.
- **Step 2:** Click on the **Automatic** option.
- **Step 3:** Select **Density** from the drop-down menu.

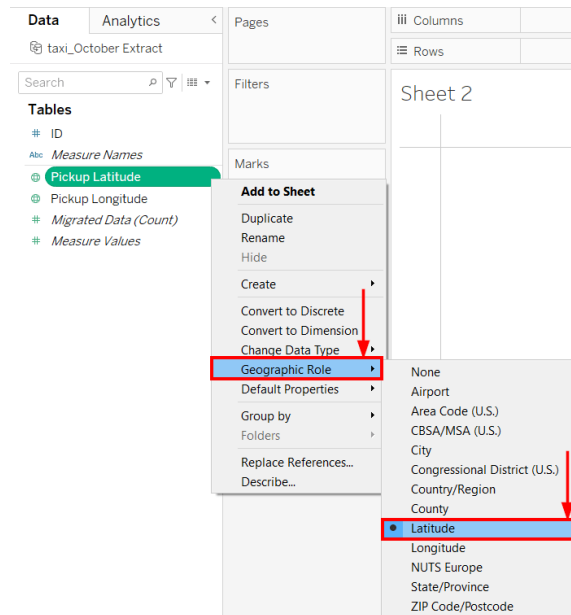


Image

Source: <https://www.tableau.com/>

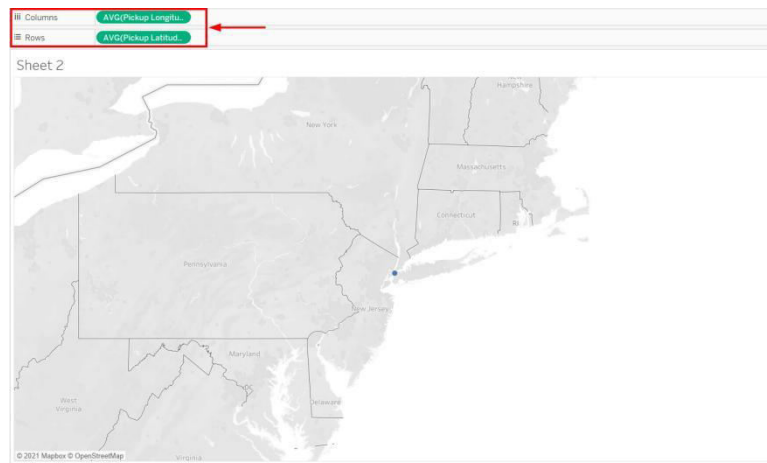
However, Tableau Desktop enables you to create more effective heat maps when the data points of a small geographic region are greater. But to plot heat maps, your data source must contain Longitude and Latitude coordinates. Download the [Example Workbook](#), helping you to explore more about Density Maps.

- **Step 4:** Open in **Tableau Desktop**.
- **Step 5:** Create a **New Sheet**.
- **Step 6:** Assign **Latitude Geographical Role** to PickupLatitude.
- **Step 7:** Assign **Longitude Geographical Role** to PickupLongitude.

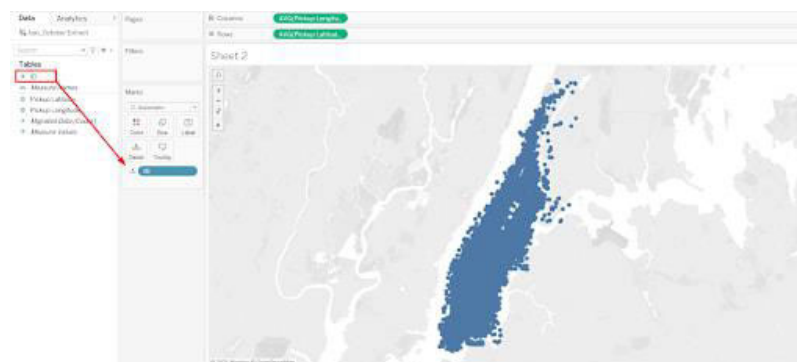


Step 8: Double-click over Pickuplongitude.

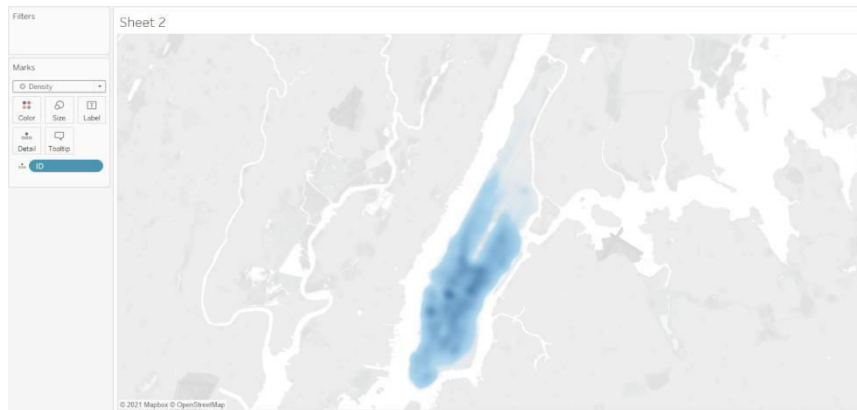
- **Step 9:** Double-click over Pickuplatitude.



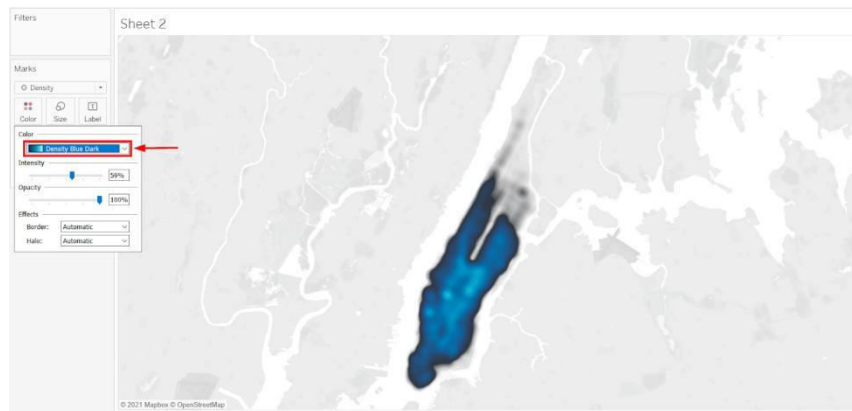
- **Step 10:** Drop the **ID** dimension from the **Data** pane to Sheet.



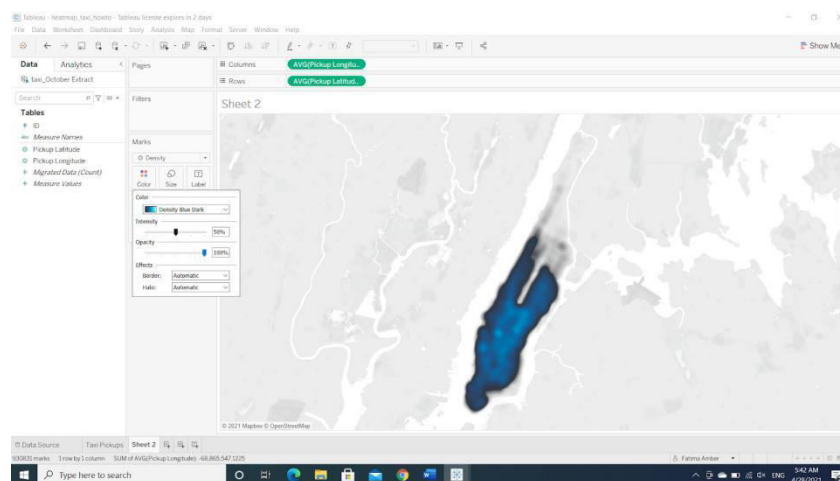
- **Step 11: Simple Heat** can be created by selecting the **Density** option from the **Marks** pane.



- **Step 12:** If you want to change the appearance of the map, select color options from the **Color** parameter.

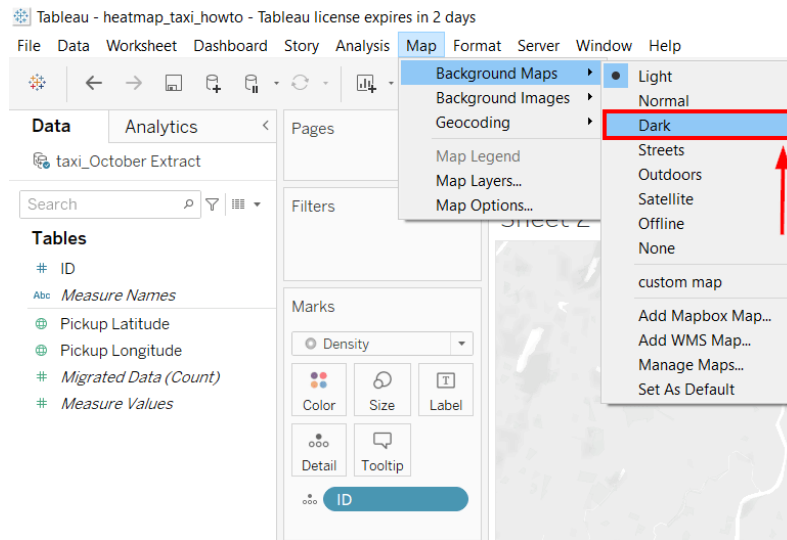


- **Step 13:** Change the intensity level to 50% and analyze the map.

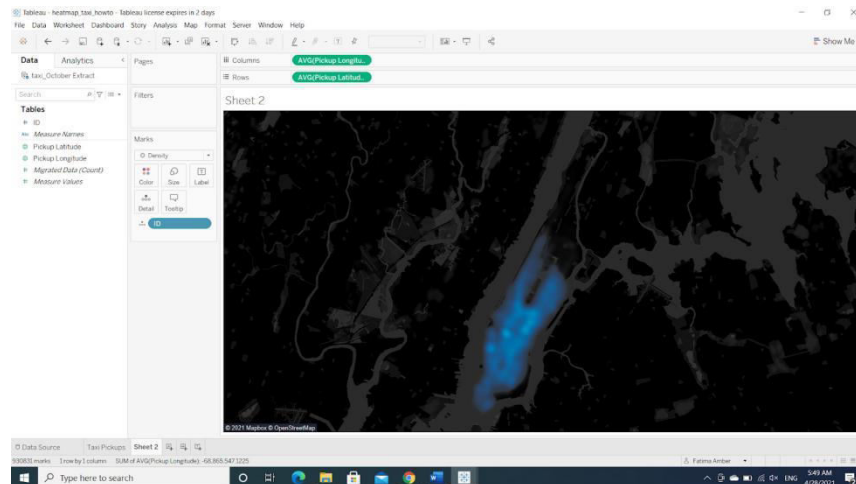


- **Step 14:** Go to the **Maps** tab.
- **Step 15:** Select the **Background Maps**.

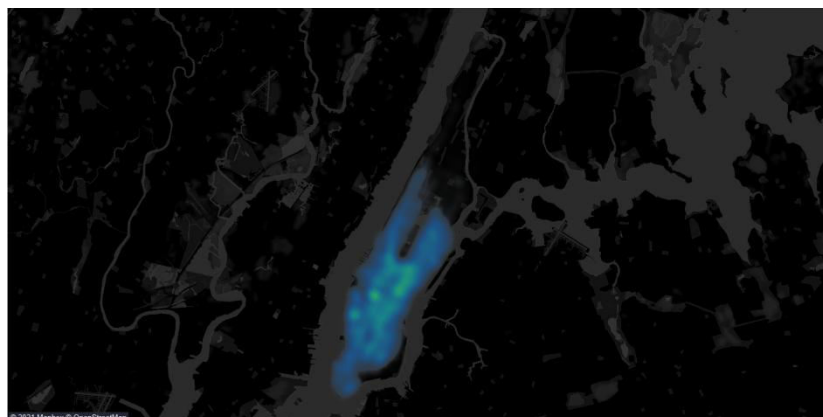
- **Step 16:** Select **Dark** instead of Light.



The Heat map might look like this.



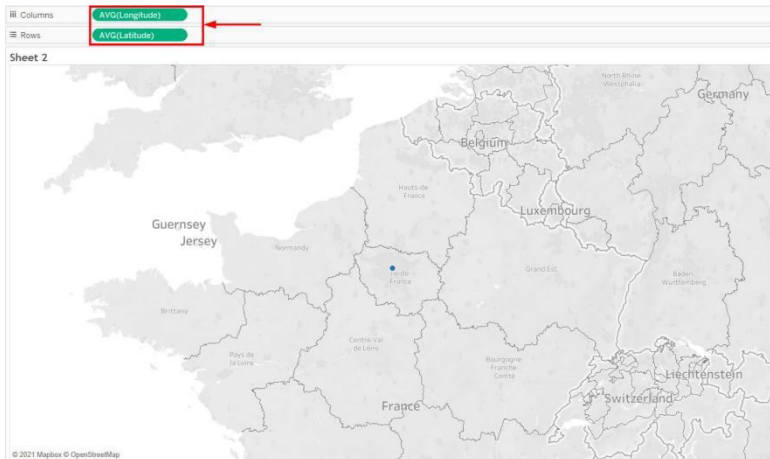
If you change color to Density Multi-Color Dark, the density becomes more visible.



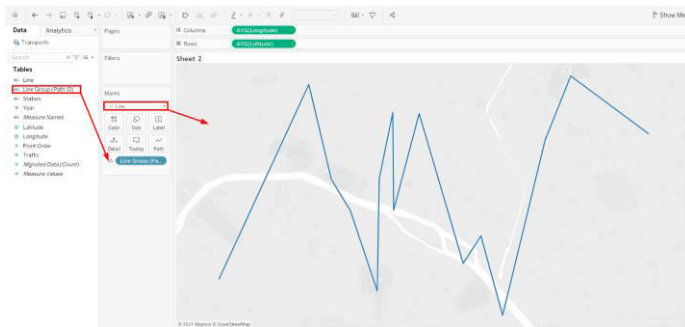
6) Spider or Origin-Destination Tableau Custom Maps

In Tableau Desktop, you can present paths between origins and destinations through Spider or Origin-Destination Maps. Let's determine the procedure to design a Spider Map in Tableau Desktop.

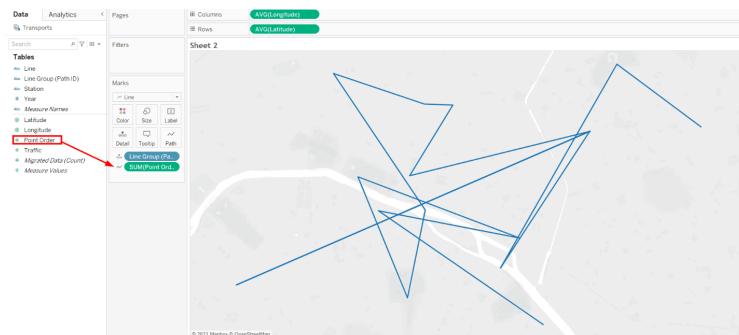
- **Step 1:** Download **Example Workbook**.
- **Step 2:** Open a **New Worksheet**.
- **Step 3:** Drag **Longitude** and **Latitude** from Measures to Sheet.



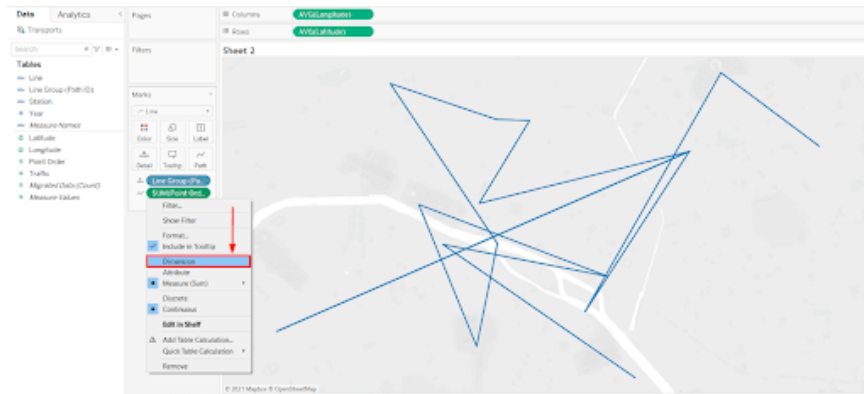
- **Step 4:** Select the **Line Group (Path ID)** and place it over the **Details** area.
- **Step 5:** Select **Line** instead of **Automatic** from Marks pane.



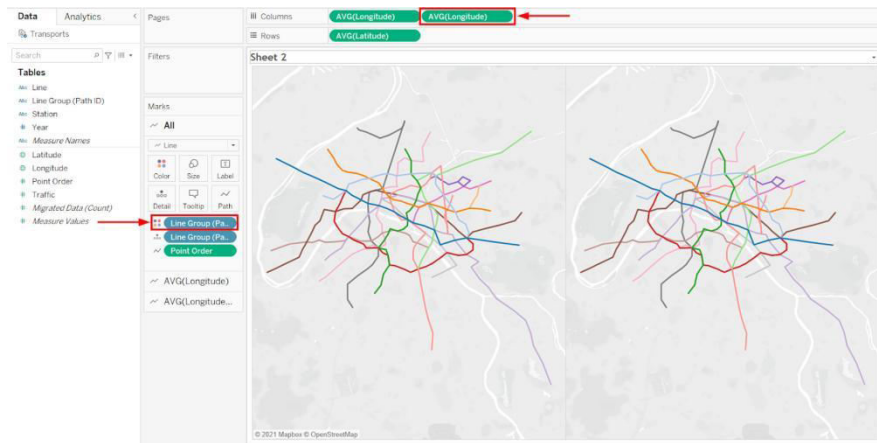
- **Step 6:** Select **Point Order** from Measures and place it over the **Path** area.



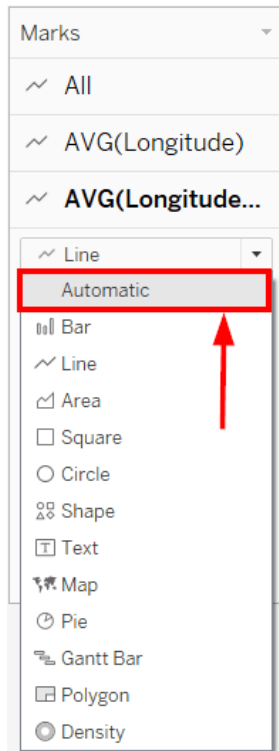
- **Step 7:** Right-click over **Point Order** and choose the **Dimension** from the drop-down menu.



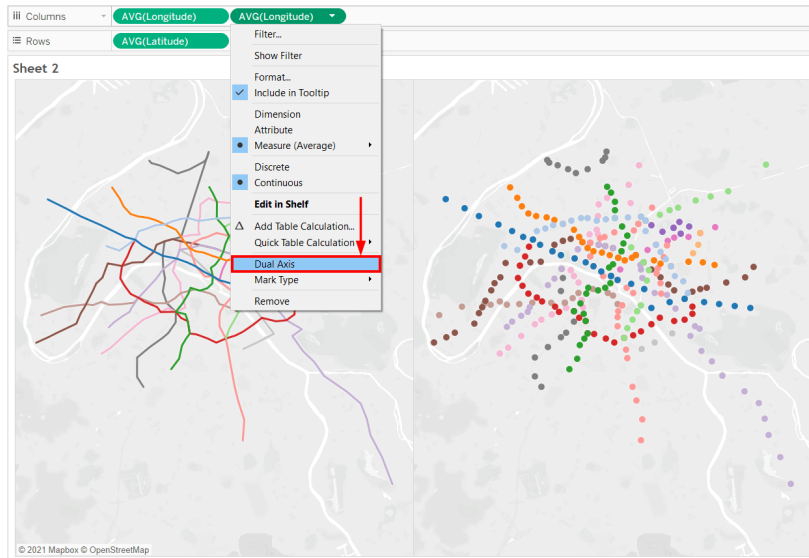
- **Step 8:** Place **Line Group (Path ID)** over the Color area.
- **Step 9:** Select **Longitude** and place it at the Column shelf.



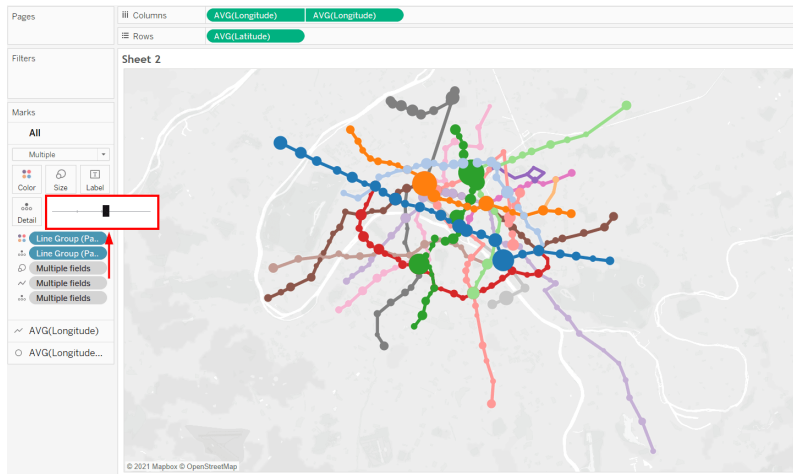
- **Step 10:** Click over the **AVG(Longitude)** and select **Automatic**.



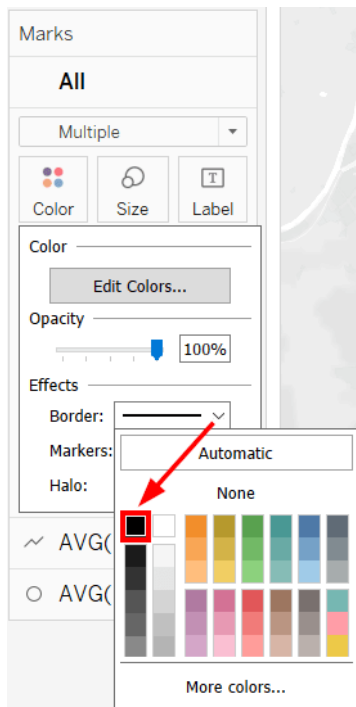
- **Step 11:** Select the **AVG(Longitude)** from the column shelf and choose **Dual Axis**.



- **Step 12:** Select the **Traffic** and place it over the **Size** of AVG(Longitude).
- **Step 13:** Choose **Size** and move the slider to the right edge.



- **Step 14:** Select the **Border** color from the **Edit Color** option of the **Mark** sheet.



The full map might look like the below picture.



For more information, visit [Origin Destination Map](#).

Conclusion

Map visualizations are significant to interpret spatial details. In Tableau Desktop, you can build both simple and spatial files, as discussed in this blog. In this comprehensive guide, you got an understanding of different ways to build Tableau Custom Maps, including Choropleth Map, Proportional Symbol Map, Heat Map, Flow Map, Point Distribution Map, and Spider Map. Although this article only covers the simple steps, you can learn more at [Tableau Custom Maps](#).

Choosing a Business Intelligence and Data Analysis tool for your business can be a tough decision primarily because almost all departments in a business such as Finance, Marketing, etc. now make use of multiple platforms to run their day-to-day operations and there is no single tool that can integrate with all these sources easily. Hence, businesses can consider using automated Data Integration platforms like Hevo.

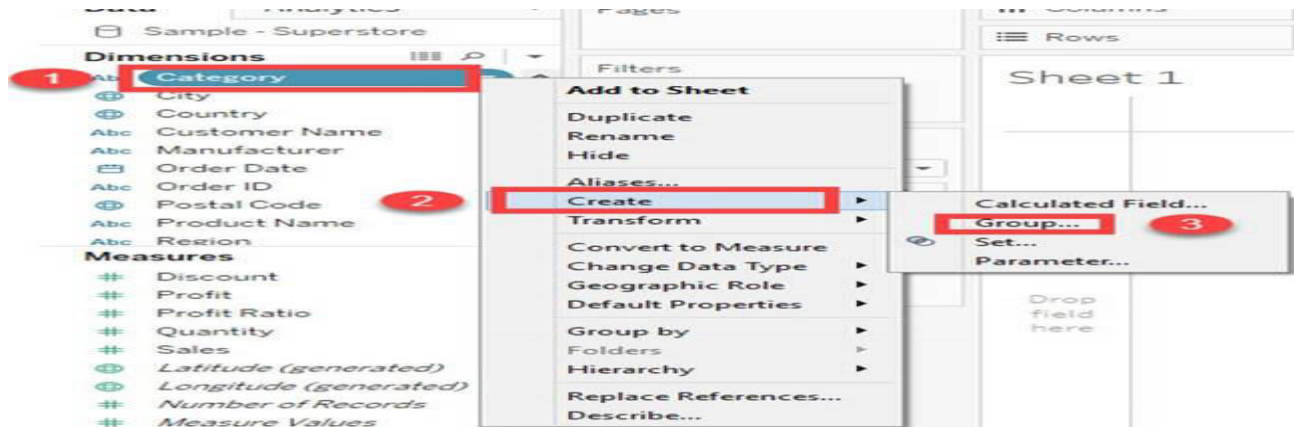
Group:

Create Groups:

Group is used to combine members present in a field. For example, aggregated values of 'Furniture' and 'Office Supplies' can be obtained by using group. Once the grouping data in Tableau is done, aggregated value of 'Furniture' and 'Office Supplies' can be shown in the visuals. The procedure to Group Data in Tableau is given as follows.

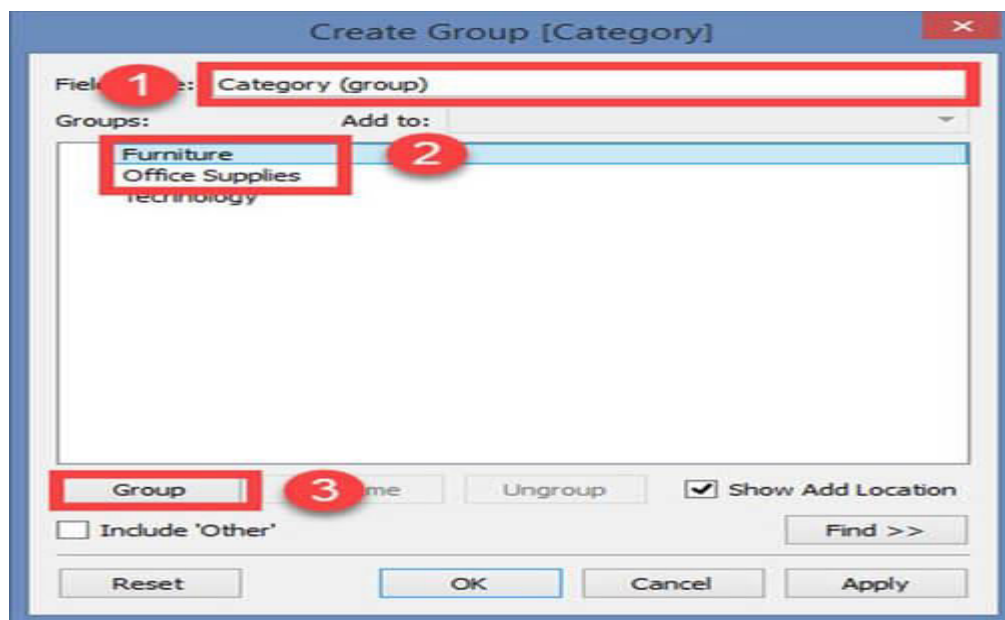
Step 1)

1. Right-click on the dimension 'Category'.
2. Click on 'Create' option.
3. Select 'Group' option.



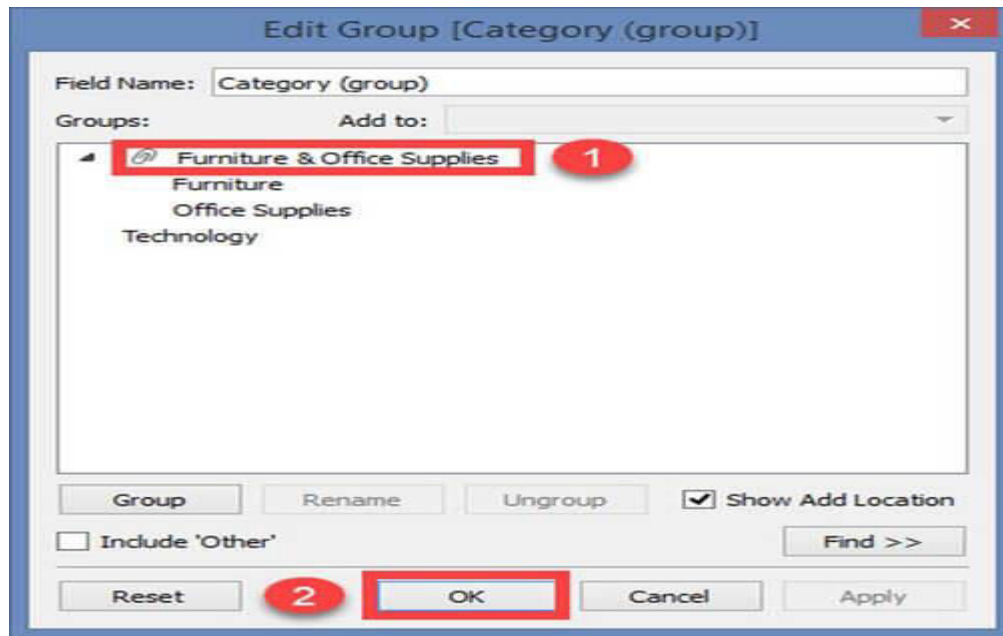
Step 2) It opens the 'Create group' window.

1. Type the name of the group data in Tableau.
2. Select the members to be grouped.
3. Click on 'Group' button.



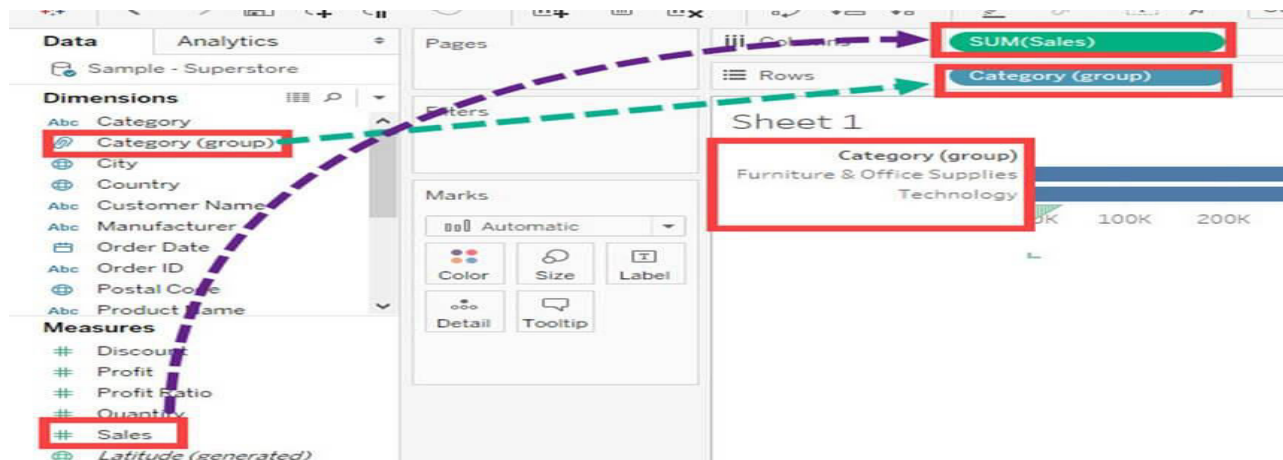
Step 3) In Edit Group Window,

1. It creates groups in Tableau of 'Furniture' and 'Office supplies'.
2. Click on Ok to create the group.



It created a group in Tableau with the name of Category (Group) and added in the dimension list. This can be used for visualizing the group by in Tableau method for members present in a field.

The following image explains the functionality of Tableau create group. The sum of sales is visualized for both furniture and office supplies for grouping in Tableau.



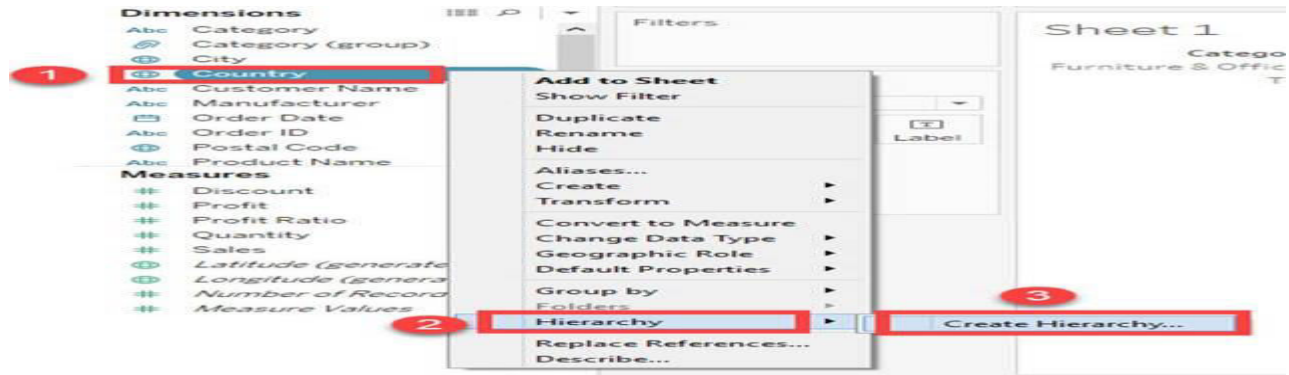
Create Hierarchy:

Hierarchies can be building in Tableau to visualize the data in granular level. Tableau hierarchies can be created by following the given steps.

Step 1) Go to a worksheet.

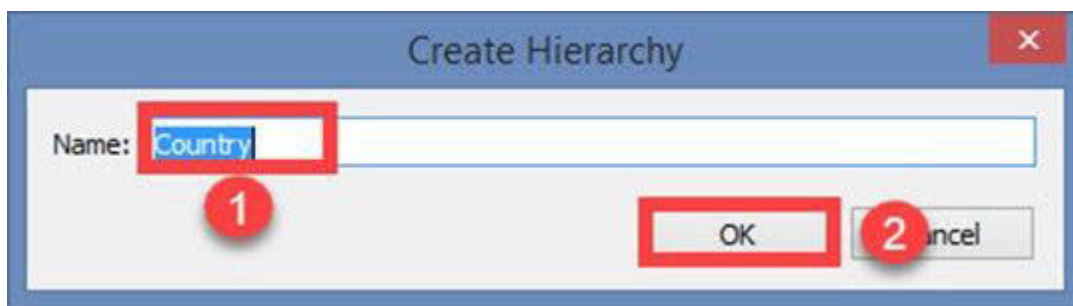
1. Select a dimension to create a hierarchy. Right-click on the dimension.
2. Select 'Hierarchy' option.

3. Click on 'Create hierarchy' option.



Step 2) It opens the 'Create Hierarchy' Window.

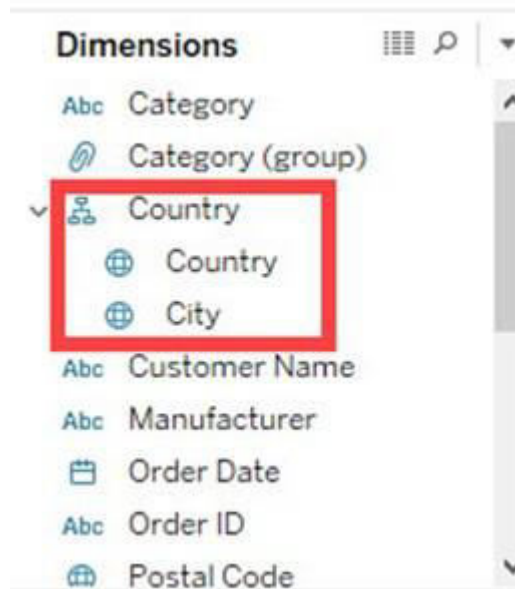
1. Enter a name for hierarchy.
2. Click on OK.



It creates a Hierarchy as shown in the image.



You can add another field to the box and create the hierarchy. In this example, the city is added into a country hierarchy.

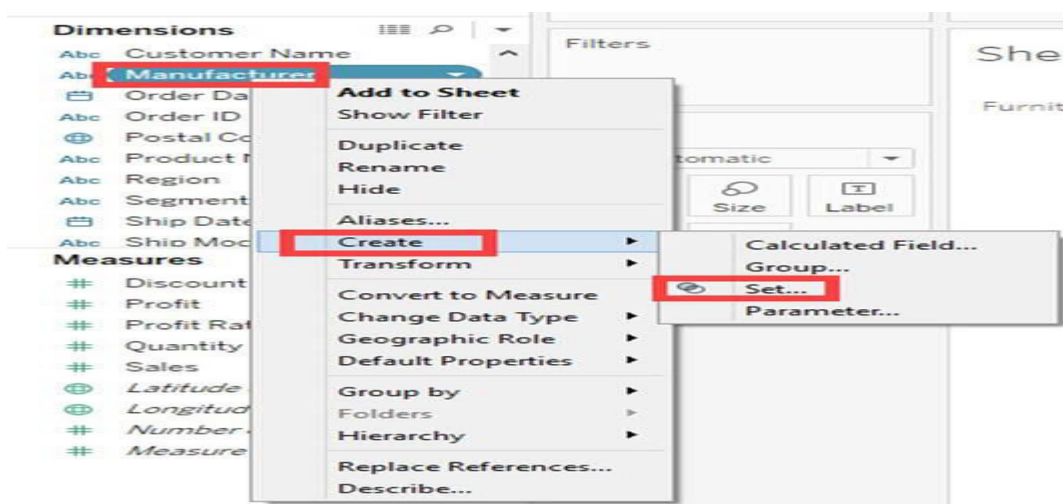


Build Sets:

Sets create a set of members out of the field present in a data set. It acts as a separated field or dimension. The procedure to build sets is given as follows.

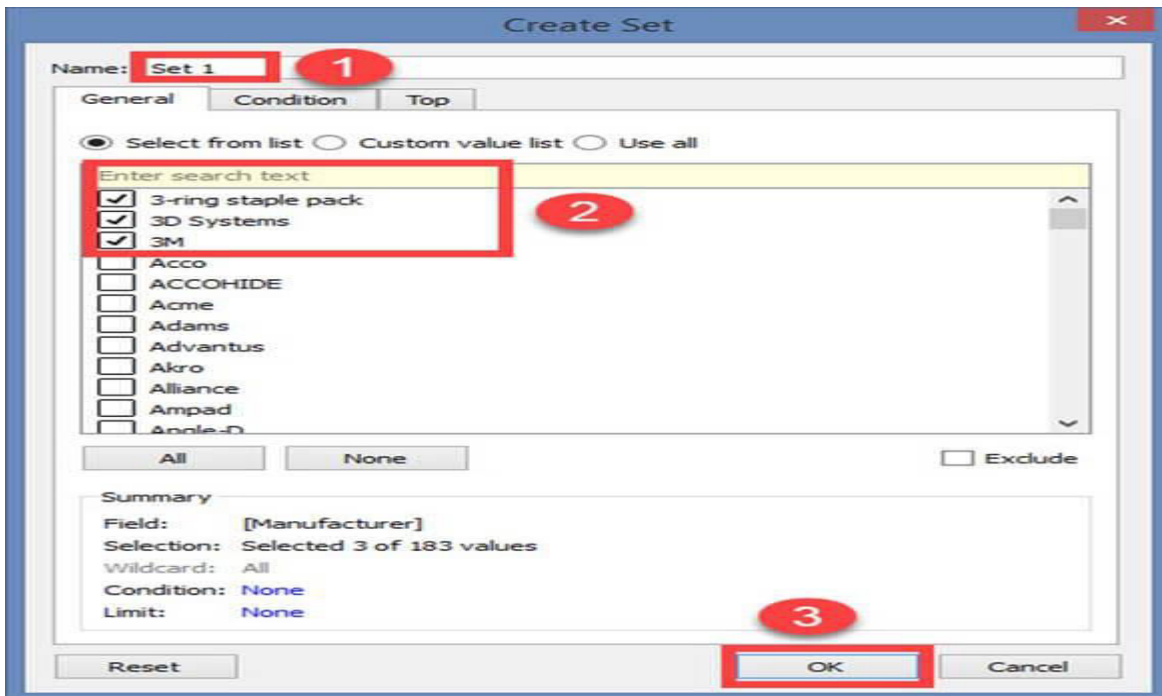
Step 1) Go to a Worksheet.

1. Right-click on a dimension.
2. Select 'Create' option.
3. Click on 'Set' option.



Step 2) It opens 'Create Set' Window.

1. Name the set to be created.
2. Select the members needs to be added in the set.
3. Click on OK.



This creates a set of the given name.

Summary:

- Users can sort the fields present in the data set.
- Tableau groups can be built to group the members present in a dimension.
- Users can build hierarchy to show the granularity level present in the dataset.
- Sets can be created to select or exclude one or more members from a field. A set can be added as a separate dimension in Tableau.

UNIT – V

Data Foundation – SCSA1309

UNIT 5 ETHICS AND RECENT TRENDS

Data and Business Insights- Data Science Engineering: - Need of Data Science - Ethics – Doing good data science – Natural Language Processing – Machine Learning Model- Valuing Data privacy – Getting informed consent - The Five Cs – Diversity – Inclusion – Future Trends.

I. Data and Business Insights

Data insights refers to the deep understanding an individual or organization gains from analyzing information on a particular issue. This deep understanding helps organizations make better decisions than by relying on gut instinct. Data insights are of utmost importance, since they drive greater understanding of a company's operations and markets while uncovering opportunities for growth. Yet, this can be achieved only if data is actually used and effectively employed within the company overall structure. The tips below explain how managers can make the most of the information they already possess.

How to define data for business insight?

Prior to any analysis, a company should determine the goals of an upcoming data analysis project. What information and knowledge does management wish to obtain from existing info sources? Doing so entails consideration of the three key factors.

Context

The first step is to look at a larger picture. What will these desired goals look like once a data analysis is on its way? Management should define goals proactively, as well as set project deadlines and milestones to couch analysis in a larger context.

Need

To make the most of available resources, management should tailor an analysis project in line with the organization's biggest needs. Whether those are improving operational efficiency, cutting a specific expense, or increasing general revenue, establishing a need helps direct analysis in a productive way.

Vision

Vision should be a part of the project. Management should know possible logical solutions to business problems and the ways to solve them by means of data and information.

Outcome

During the analysis stage, the desired outcome of the project should always be kept in mind. Data mining and analysis are iterative, but keeping a continuous outcome in mind places parameters and focus on the project at hand. Things to remember include: how will these results be used? Who will be using these results? What are the measurable objectives at play?

Insights are a big part of what makes businesses grow. It is very important for any organization to have a guide that helps them collect insights from the get-go, streamline and focus their approach on the right kind of methodology to get the best results. Most importantly, never ignore the valuable information that might help drive business objectives forward.

The four simple steps to guide this process are listed below:

Collect: A large, high quality database translates into good business insight for any organization. Data on sales prospects, for example, can be gathered and collected from network engagements, forums, blogs, reviews and website click streams, and ad engagements.

Connect: Data can be large and unwieldy, providing different sources of information in different places. It is not always structured to help analysts make connections, but it can be the other way around. Joining and connecting different data sources and sets can create more knowledge than relying on fragmented data only.

Manage: The world brands live and breathe in a social media focused world. There is a lot of data that can be collected and leveraged from these fast-paced interactions with the right techniques and tools. Some information can be accessed in real-time only, while the other data points can be stored conveniently.

Discover and analyze: Finally, collaboration is required for using the data to discover important business information. This process should include data scientists, marketing experts, and other subject matter gurus across the organization. The goal is to generate key information that can be applied to various business goals.

If you want to be successful in the modern-day highly competitive digital business landscape, you need to learn how to move from data to business insights and how to treat information as an asset.

What is data engineering?

Data engineering is a set of operations aimed at creating interfaces and mechanisms for the flow and access of information. It takes dedicated specialists – data engineers – to maintain data so that it remains available and usable by others. In short, data engineers set up and operate the organization’s data infrastructure preparing it for further analysis by data analysts and scientists.

To understand data engineering in simple terms, let’s start with data sources. Within a large organization, there are usually many different types of operations management software (e.g., ERP, CRM, production systems, etc.), all of which contain different databases with varied information. Besides, data can be stored as separate files or even pulled from external sources in real time (such as various IoT devices). As the number of data sources multiplies, having data scattered all over in various formats prevents the organization from seeing the full and clear picture of their business state.

For example, it’s necessary to make sales data from its dedicated database “talk” to inventory records kept in a SQL server. This task requires extracting data out of those systems and integrating it in a unified storage where it’s collected, reformatted, and kept ready for use. Such storages are called data warehouses. Now, end-users (which include employees from different departments, managers, data scientists, business intelligence (BI) engineers, etc.) can connect to the warehouse, access the needed data in the convenient format, and start getting valuable insights from it.

The process of moving data from one system to another, be it a SaaS application, a data warehouse (DW), or just another database, is maintained by data engineers (read on to learn more about the role and skillset of this specialist).

Now, let’s look deeper into the main concepts of the data engineering domain, step by step.

ETL data pipeline

A **Data pipeline** is basically a set of tools and processes for moving data from one system to another for storage and further handling. It captures datasets from multiple sources and inserts them into some form of database, another tool, or app, providing quick and reliable access to this combined data for the teams of data scientists, BI engineers, data analysts, etc.

Constructing data pipelines is the core responsibility of data engineering. It requires advanced programming skills to design a program for continuous and automated data exchange. A data pipeline is commonly used for

- moving data to the cloud or to a data warehouse,
- wrangling the data into a single location for convenience in machine learning projects,
- integrating data from various connected devices and systems in IoT,
- copying databases into a cloud data warehouse, and
- bringing data to one place in BI for informed business decisions.

You can read our detailed explanatory post to learn more about data pipelines, their components, and types. Now, let's explore what ETL stands for.

What is ETL?

Pipeline infrastructure varies depending on the use case and scale. However, data engineering usually starts with ETL operations:

1. **Extracting** data from source databases,
2. **Transforming** data to match a unified format for specific business purposes, and
3. **Loading** reformatted data to the storage (mainly, data warehouses).

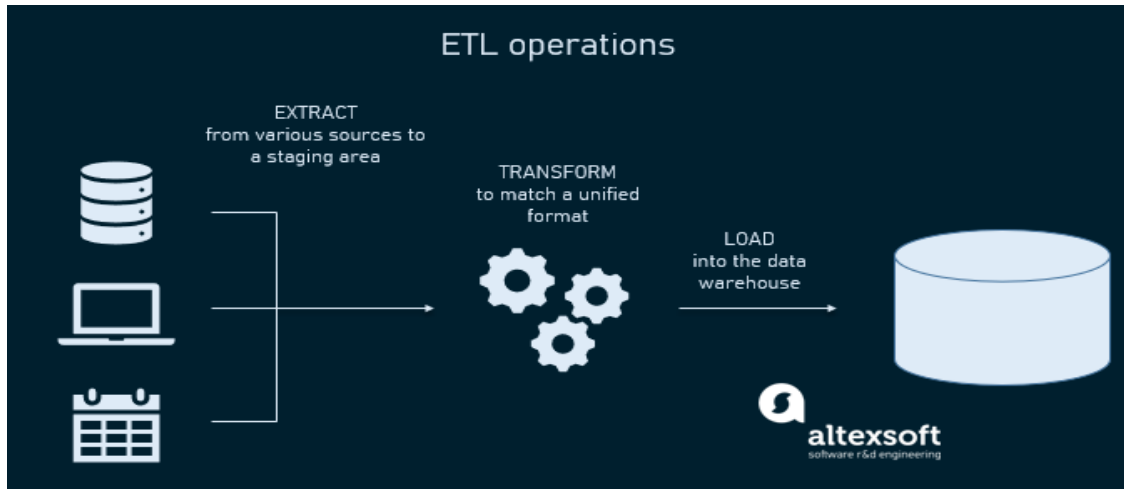


Fig 5.1. ETL operations

1. Extract — retrieving incoming data. At the start of the pipeline, we're dealing with raw data from numerous separate sources. Data engineers write pieces of code – jobs – that run on a schedule extracting all the data gathered during a certain period.

2. Transform — standardizing data. Data from disparate sources is often inconsistent. So, for efficient querying and analysis, it must be modified. Having data extracted, engineers execute another set of jobs that transforms it to meet the format requirements (e.g., units of measure, dates, attributes like color or size.) Data transformation is a critical function, as it significantly improves data discoverability and usability.

3. Load — saving data to a new destination. After bringing data into a usable state, engineers can load it to the destination that typically is a relational database management system (RDBMS), a data warehouse, or Hadoop. Each destination has its specific practices to follow for performance and reliability.

Once the data is transformed and loaded into a single storage, it can be used for further analysis and business intelligence operations, i.e., generating reports, creating visualizations, etc.

NB: Despite being automated, a data pipeline must be constantly maintained by data engineers. They repair failures, update the system by adding/deleting fields, or adjust the schema to the changing needs of the business.

Data pipeline challenges

Setting up secure and reliable data flow is a challenging task. There are so many things that can go wrong during data transportation: Data can be corrupted, hit bottlenecks causing latency, or data sources may conflict, generating duplicate or incorrect data. Getting data into one place requires careful planning and testing to filter out junk data, eliminating duplicates and incompatible data types to obfuscate sensitive information while not missing critical data.

Juan De Dios Santos, an experienced practitioner in the data industry, outlines two major pitfalls in building data pipelines:

- lacking relevant metrics and
- underestimating data load.

“The importance of a healthy and relevant metrics system is that it can inform us of the status and performance of each pipeline stage, while with underestimating the data load, I am referring to building the system in such a way that it won’t face any overload if the product experiences an unexpected surge of users,” elaborates Juan.

Having an ETL pipeline that is only connected to a standard transactional database is ineffective as the volume and variety of data requires a dedicated storage design. So, besides an ETL pipeline there’s a need to build a data warehouse that supports and facilitates BI activities. Let’s see how it works.

Data warehouse

A **data warehouse** (DW) is a central repository where data is stored in query-able forms. From a technical standpoint, a data warehouse is a relational database optimized for reading, aggregating, and querying large volumes of data. Traditionally, DWs only contained structured data, or data that can be arranged in tables. However, modern DWs can combine both structured and unstructured

data where unstructured refers to a wide variety of forms (such as images, pdf files, audio formats, etc.) that are harder to categorize and process.

Without DWs, data scientists would have to pull data straight from the production database and may wind up reporting different results to the same question or cause delays and even outages. Serving as an enterprise's single source of truth, the data warehouse simplifies the organization's reporting and analysis, decision making, and metrics forecasting.

Surprisingly, DW isn't a regular database. How so?

First of all, they differ in terms of data structure. A regular database normalizes data excluding any data redundancies and separating related data into tables. This takes up a lot of computing resources, as a single query combines data from many tables. Contrarily, a DW uses simple queries with few tables to improve performance and analytics.

Second, aimed at day-to-day transactions, standard transactional databases don't usually store historic data, while for warehouses, it's their main purpose, as they collect data from multiple periods. DW simplifies a data analyst's job, allowing for manipulating all data from a single interface and deriving analytics, visualizations, and statistics.

Typically, a data warehouse doesn't support as many concurrent users as a database, being designed for a small group of analysts and business users.

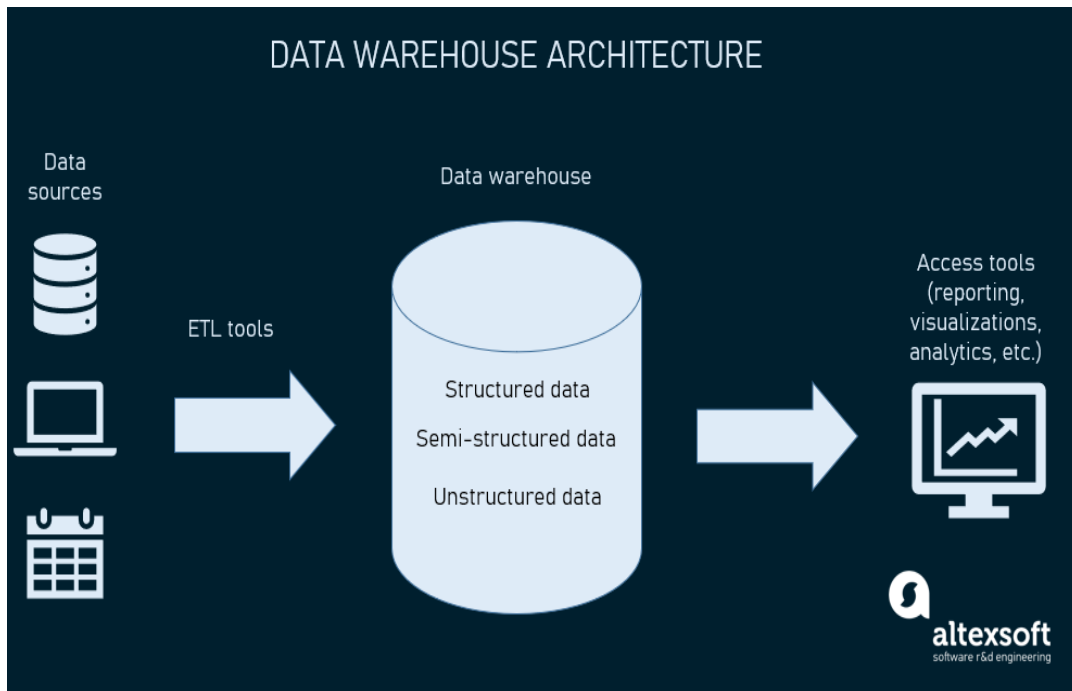


Fig 5.2. Datawarehouse Architecture

To construct a data warehouse, four basic components are combined.

Data warehouse storage. The foundation of data warehouse architecture is a database that stores all enterprise data allowing business users to access it for drawing valuable insights.

Data architects usually decide between on-premises and cloud-hosted DWs noting how the business can benefit from this or that solution. Although the cloud environment is more cost-efficient, easier to scale up or down, and isn't limited to a prescribed structure, it may lose to on-prem solutions in terms of querying speed and security. We're going to list the most popular tools further on.

A data architect can also design collective storage for your data warehouse – multiple databases running in parallel. This will improve the warehouse's scalability.

Metadata. Adding business context to data, metadata helps transform it into comprehensible knowledge. Metadata defines how data can be changed and processed. It contains information about any transformations or operations applied to source data while loading it into the data warehouse.

Data warehouse access tools. Designed to facilitate interactions with DW databases for business users, access tools need to be integrated with the warehouse. They have different functions. For example, query and reporting tools are used for generating business analysis reports. Another type of access tools – data mining tools – automate the process of finding patterns and correlations in large amounts of data based on advanced statistical modeling techniques.

Data warehouse management tools. Spanning the enterprise, data warehouse deals with a number of management and administrative operations. That's why managing a DW requires a solution that can facilitate all these operations. Dedicated data warehouse management tools exist to accomplish this.

For a more detailed description of different data warehouse architectures, types, and components, visit our dedicated post.

Data warehouses are a great step forward in enhancing your data architecture. However, if you have hundreds of users from different departments, DWs can be too bulky and slow to operate. In this case, data marts can be built and implemented to increase speed and efficiency.

Data marts

Simply speaking, a **data mart** is a smaller data warehouse (their size is usually less than 100Gb.). They become necessary when the company (and the amount of its data) grows and it becomes too long and ineffective to search for information in an enterprise DW. Instead, data marts are built to allow different departments (e.g., sales, marketing, C-suite) to access relevant information quickly and easily.

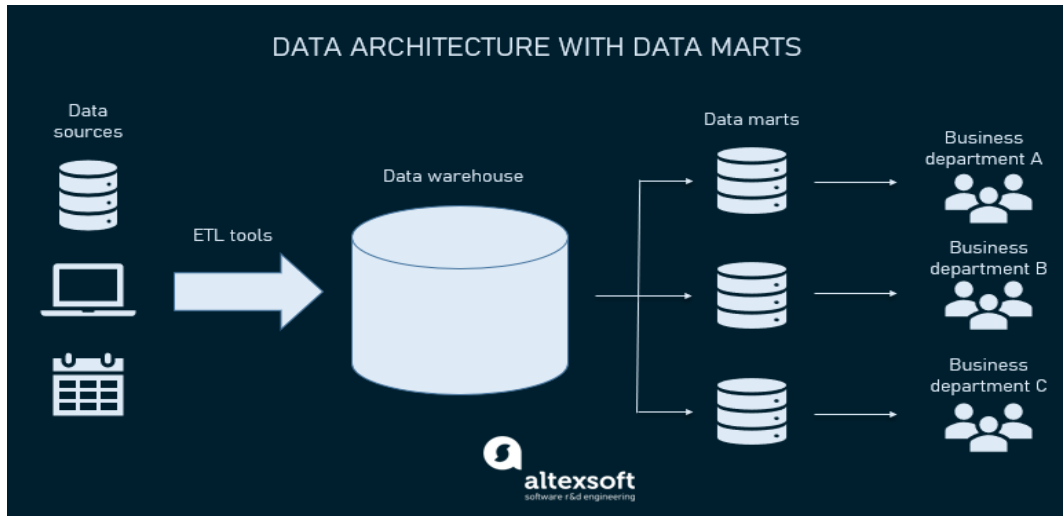


Fig 5.3. Data Marts

There are three main types of data marts.

Dependent data marts are created from an enterprise DW and use it as a main source of information (it's also known as a top-down approach).

Independent data marts are standalone systems that function without DWs extracting information from various external and internal sources (it's also known as a bottom-up approach).

Hybrid data marts combine information from both DW and other operational systems.

So, the main difference between data warehouses and data marts is that a DW is a large repository that holds all company data extracted from multiple sources, making it difficult to process and manage queries. Meanwhile, a data mart is a smaller storage that contains a limited amount of data for the usage of a particular business group or department.

Here is a comprehensive overview of the concept of data marts, their types, and structure if you want to learn more.

While data marts allow business users to quickly access the queried data, often just getting the information is not enough. It has to be efficiently processed and analyzed to get those actionable

insights that support decision-making. Looking at your data from different perspectives is possible thanks to OLAP cubes. Let's see what they are.

OLAP and OLAP cubes

OLAP or Online Analytical Processing refers to the computing approach that allows users to perform multidimensional data analysis. It's contrasted with OLTP or Online Transactional Processing, which is a simpler method of interacting with databases that isn't designed for analyzing massive amounts of data from different perspectives.

Traditional databases look like spreadsheets, using the two-dimensional structure of rows and columns. However, in OLAP, datasets are presented in multidimensional structures — **OLAP cubes**. Such structures enable efficient processing and advanced analysis of huge amounts of varied data. For example, a sales department report would include such dimensions as product, region, sales representative, sales amount, month, and so on.

Here's where OLAP cubes are in the company's data architecture. Information from DWs is aggregated and loaded into the OLAP cube where it gets precalculated and is readily available for users requests.

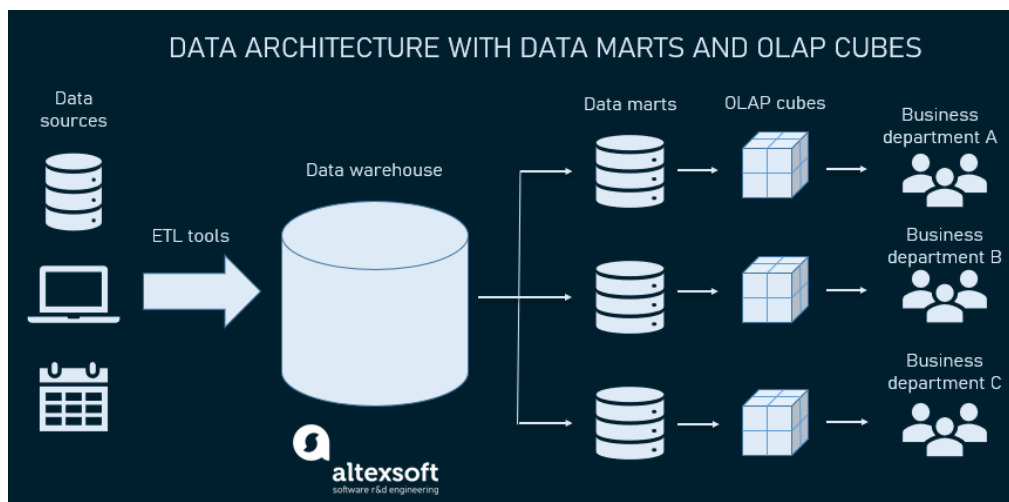


Fig 5.4. Data Cubes

Within OLAP, data can be analyzed from multiple perspectives. For example, it can be drilled down/rolled up if you need to change the hierarchy level of data representation and get a more/less detailed picture. You can also slice information to segment out a particular dataset as a separate spreadsheet or dice it to create a different cube. Using these and other techniques enables finding patterns in varied data and creating a wide range of reports.

It's important to note that OLAP cubes have to be custom built for every particular report or analytical query. However, their usage is justified since, as we said, they enable advanced, multidimensional analysis that was previously too complicated to perform.

For a more detailed explanation of OLAP concepts, architecture, and main operations, consider reading our dedicated post.

ELT data pipeline and big data engineering

Speaking about data engineering, we can't ignore the big data concept. Grounded in the four Vs – volume, velocity, variety, and veracity – big data usually floods large technology companies like YouTube, Amazon, or Instagram. Big data engineering is about building massive reservoirs and highly scalable and fault-tolerant distributed systems able to inherently store and process data.

Big data architecture differs from conventional data handling, as here we're talking about such massive volumes of rapidly changing information streams that a data warehouse isn't able to accommodate. The architecture that can handle such an amount of data is a data lake.

Data lake

A **Data lake** is a vast pool for saving data in its native, unprocessed form. A data lake stands out for its high agility as it isn't limited to a warehouse's fixed configuration.

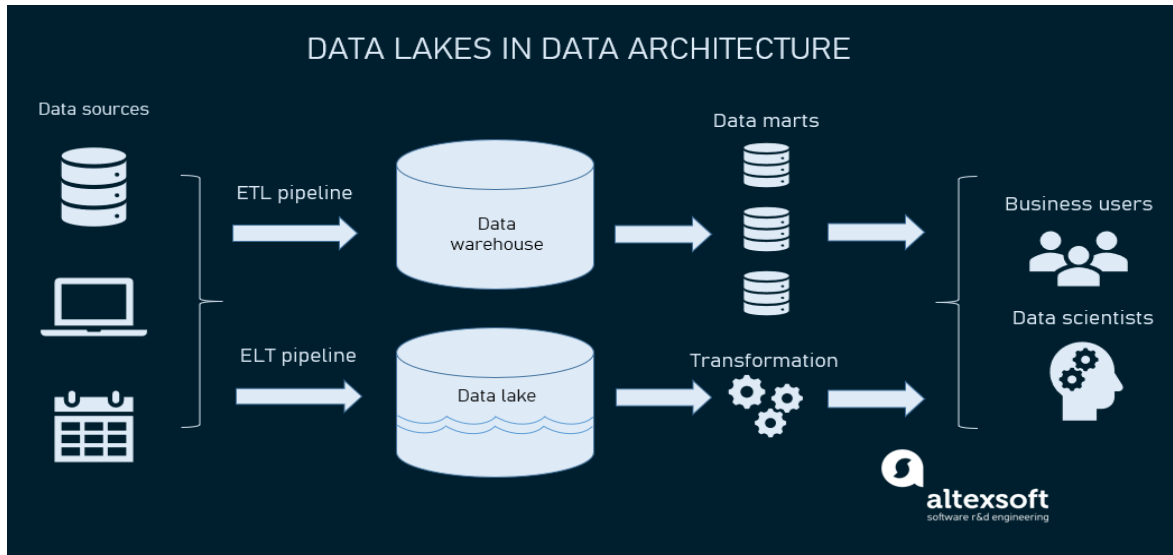


Fig 5.5. Data Lakes

Big data architecture with a data lake

In contrast to the ETL architecture we described above, a data lake uses the **ELT** approach swapping transform and load operations. Supporting large storage and scalable computing, a data lake starts data loading immediately after extracting it, handling raw — often unstructured — data.

You can check our detailed comparison of ETL and ELT approaches , but in a nutshell, ELT is a more advanced method as it allows for significantly increasing volumes of data to be processed. It also expedites information processing (since transformation happens only on-demand) and requires less maintenance.

A data lake is worth building in those projects that are going to scale and would need a more advanced architecture. Besides, they are very convenient, for instance, when the purpose of data hasn't been determined yet since you can load data quickly, store it, and then modify it as necessary. Once you need data, you can apply such data processing tools as Apache or MapReduce to transform it during retrieval and analysis.

Data lakes are also a powerful tool for data scientists and ML engineers, who would use raw data to prepare it for predictive analytics and machine learning. Read more about data preparation in our separate article or watch this 14-minute explainer.

Lakes are built on large, distributed clusters that would be able to store and process those masses of data. A popular example of such data lake platforms is Hadoop. Let's look closer at what that is.

Hadoop

So, Hadoop is a large-scale data processing framework based on Java. This software project is capable of structuring various big data types for further analysis. The platform allows for splitting data analysis jobs across various computers and processing them in parallel.

The Hadoop ecosystem consists of the following set of tools.

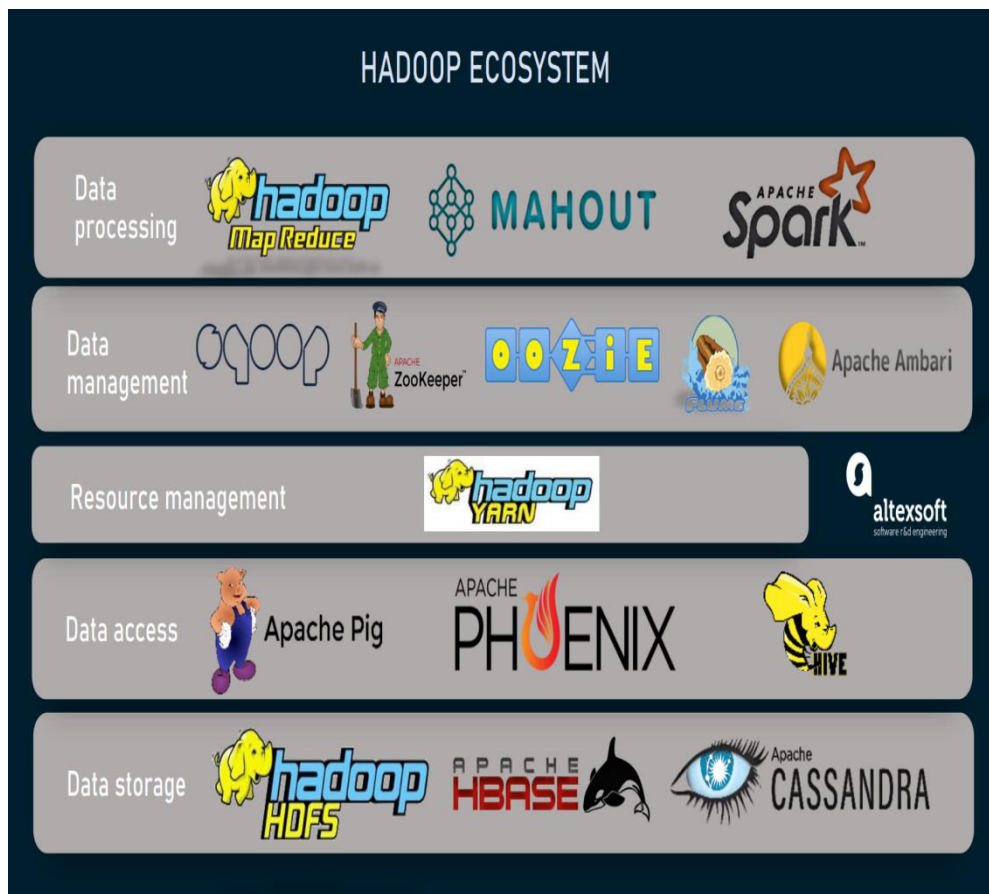


Fig 5.6. Hadoop Eco System

Hadoop ecosystem

Hadoop Distributed File System (HDFS). Java-based big data storage system, HDFS includes two components: NameNode stores metadata while DataNode is responsible for actual data and performs operations according to NameNode.

MapReduce. It's a framework for writing applications that process the data stored in HDFS. Parallel in nature, MapReduce programs are effective for performing big data analysis using multiple machines in the cluster.

YARN. As the operating system of Hadoop, YARN helps manage and monitor workloads.

Hive. A system for summarizing, querying and analyzing large datasets, Hive uses its own language – HQL- which is similar to SQL. HiveQL automatically translates SQL-like queries into MapReduce jobs for execution on Hadoop.

Pig. Having similar goals with Hive, Pig also has its own language – PigLatin. When to use Pig and when to use Hive is the question. Pig is a better option for programming purposes, while Hive is mainly used by data analysts for creating reports.

HBase. A NoSQL database built on top of HDFS that provides real-time access to read or write data.

There are many other components that empower Hadoop functionality: HCatalog, Avro, Thrift, Apache Drill, Mahout, Sqoop, Flume, Ambari, Zookeeper, Oozie. You can read about their specifications in Hadoop documentation. You can also check our post dedicated to Hadoop and Spark as the main big data tools for a more detailed description.

Enterprise data hub

Besides big data capabilities, data lakes also brought new challenges for governance and security, and the risk of turning into a data swamp – a collection of all kinds of data that is neither governable nor usable. To tackle these problems, a new data integration approach emerged – data hub – where data is physically moved and re-indexed into a new system.

Enterprise data hubs (EDHs) are the next generation of data management that evolved from DWs and data lakes, enabling easier data consolidation, harmonization, processing, governance, and exploration.

In the data hub architecture, data from many operational and analytic sources is acquired through replication and/or publish-and-subscribe interfaces (read about pub/sub messaging pattern in our explainer.). As data changes occur, replication uses changed data capture (CDC) to continuously populate the hub, while publish-and-subscribe allows the hub to subscribe to messages published by data sources. EDH data-centric storage architecture enables executing applications where the data resides.

So what are the benefits of EDH over traditional data consolidation? Let's have a closer look.

Easy connection of new data sources. Contrary to DW that lacks flexibility, modern EDH can connect multiple systems on the fly, integrating diverse data types.

Up-to-date data. Outdated data can be an issue with a DW, but EDH overcomes it, presenting fresh data ready for analysis right after capturing it.

Tools. EDH offers powerful tools for processing and analyzing data.

Rapid deployment. While a DW system deploy can last months and even years, EDH deployment is a matter of days or weeks.

To sum it all up, a data warehouse is constructed to deal mainly with structured data for the purpose of self-service analytics and BI; a data lake is built to deal with sizable aggregates of both structured and unstructured data to support deep learning, machine learning, and AI in general; and a data hub is created for multi-structured data portability, easier exchange, and efficient processing.

An EDH can be integrated with a DW and/or a data lake to streamline data processing and deal with the common challenges these architectures face.

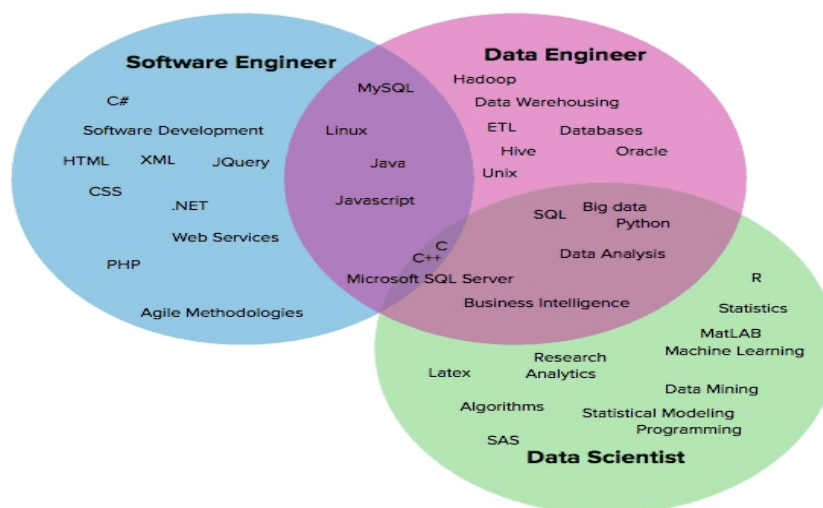
Role of data engineer

Now that we know what data engineering is concerned with, let's delve into the role that specializes in creating software solutions around big data – a data engineer.

Juan De Dios Santos, a data engineer himself, defines this role in the following way: “In a multidisciplinary team that includes data scientists, BI engineers, and data engineers, the role of the data engineer is mostly to ensure the quality and availability of the data.” He also adds that a data engineer might collaborate with the others at the time of implementing or designing a data-related feature (or product) such as an A/B test, the deployment of a machine learning model, and the refinement of an existing data source.

We have a separate article explaining the role, responsibilities, and skills of a data engineer, so here we'll only briefly recap.

Skills and qualifications



Software engineering background. Data engineers use programming languages to enable clean, reliable, and performative access to data and databases. Juan points out their ability to work with the complete cycle of software development including ideation, architecture design, deployment and DevOps, prototyping, testing, defining metrics, alerts, and monitoring systems. Data engineers are experienced programmers in at least Python or Scala/Java.

Data-related skills. “A data engineer should have knowledge of multiple kinds of databases (SQL and NoSQL), data platforms, concepts such as MapReduce, batch and stream processing, and even some basic theory of data itself, e.g., data types, and descriptive statistics,” underlines Juan.

Systems creation skills. Data engineers need to have experience with various data storage technologies and frameworks they can combine to build data pipelines.

Toolkit

Data engineering process involves using different data storage and manipulation tools together. So a data engineer should have a deep understanding of many data technologies to be able to choose the right ones for a certain job.

Tools for writing ETL/ELT pipelines:

Airflow. This Python-based workflow management system was initially developed by Airbnb to rearchitect their data pipelines. Migrating to Airflow, the company reduced their experimentation reporting framework (ERF) run-time from 24+ hours to about 45 minutes. Airflow’s key feature is automating scripts to perform tasks. Among the Airflow’s pros, Juan highlights its operators: “They allow us to execute bash commands, run a SQL query or even send an email”. Juan also stresses Airflow’s ability to send Slack notifications, complete and rich UI, and the overall maturity of the project. On the contrary, Juan dislikes that Airflow only allows for writing jobs in Python.

Cloud Dataflow. A cloud-based data processing service, Dataflow is aimed at large-scale data ingestion and low-latency processing through fast parallel execution of the analytics pipelines. Dataflow has a benefit over Airflow as it supports multiple languages like Java, Python, SQL, and

engines like Flink and Spark. It is also well maintained by Google Cloud. However, Juan warns that Dataflow's high cost might be a disadvantage for some.

Kafka. From a messaging queue to a full-fledged event streaming platform, Apache Kafka distributes data across multiple nodes for a highly available deployment within a single data center or across multiple availability zones. As an abstraction of a distributed commit log, it provides durable storage.

Other popular ETL and data solutions are the Stitch platform for rapidly moving data and Blendo, a tool for syncing data from various sources to a data warehouse.

Warehouse solutions. Widely used on-premise data warehouse tools include Teradata Data Warehouse, SAP Data Warehouse, IBM db2, and Oracle Exadata. Most popular cloud-based data warehouse solutions are Amazon Redshift and Google BigQuery. Be sure to check our detailed comparison of the top cloud warehouse software.

Big data tools. Big data technologies that a data engineer should be able to utilize (or at least know of) are Hadoop, distributed file systems such as HDFS, search engines like Elasticsearch, ETL and data platforms: Apache Spark analytics engine for large-scale data processing, Apache Drill SQL query engine with big data execution capabilities, Apache Beam model and software development kit for constructing and running pipelines on distributed processing backends in parallel.

Closing: Data engineer vs data scientist

It's not rare that a data engineer is confused with a data scientist. We asked Alexander Konduforov, a data scientist at AltexSoft, with over 10 years of experience, to comment on the difference between these two roles:

“Both data scientists and data engineers work with data but solve quite different tasks, have different skills, and use different tools. Data engineers build and maintain massive data storage and apply engineering skills: programming languages, ETL techniques, knowledge of different data warehouses and database languages. Whereas data scientists clean and analyze this data, get

valuable insights from it, implement models for forecasting and predictive analytics, and mostly apply their math and algorithmic skills, machine learning algorithms and tools.”

Alexander stresses that accessing data can be a difficult task for data scientists for several reasons.

- Vast data volumes require additional effort and specific engineering solutions to access and process it in a reasonable amount of time.
- Data is usually stored in lots of different storages and formats. In this case, it makes sense first to clean it up by taking dataset preparation measures, transform, merge, and move to a more structured storage, like a data warehouse. This is typically a task for data architects and engineers.
- Data storages have different APIs for accessing them. In this case, data scientists need data engineers to implement the most efficient and reliable pipeline of getting data for their purpose.

As we can see, working with data storages built by data engineers, data scientists become their “internal clients.” That’s where their collaboration takes place.

Natural Language Processing

Natural Language Processing or NLP is a field of Artificial Intelligence that gives the machines the ability to read, understand and derive meaning from human languages.

It is a discipline that focuses on the interaction between data science and human language, and is scaling to lots of industries. Today NLP is booming thanks to the huge improvements in the access to data and the increase in computational power, which are allowing practitioners to achieve meaningful results in areas like healthcare, media, finance and human resources, among others.

Use Cases of NLP

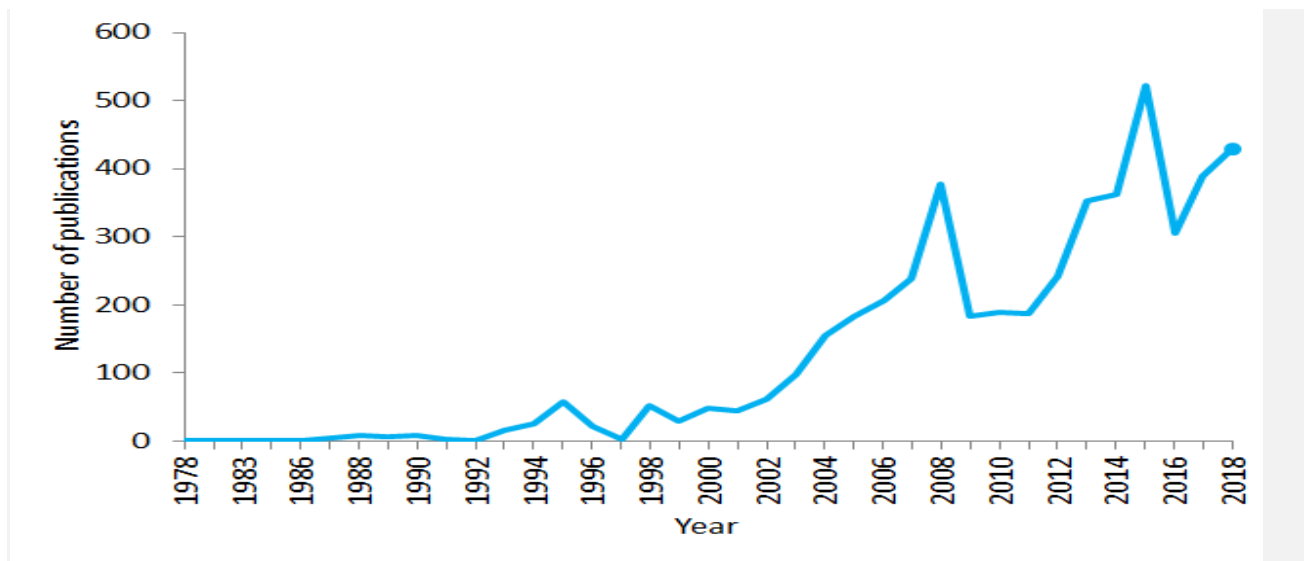
In simple terms, NLP represents the automatic handling of natural human language like speech or text, and although the concept itself is fascinating, the real value behind this technology comes from the use cases.

NLP can help you with lots of tasks and the fields of application just seem to increase on a daily basis. Let's mention some examples:

- NLP enables the recognition and **prediction of diseases** based on electronic health records and patient's own speech. This capability is being explored in health conditions that go from cardiovascular diseases to depression and even schizophrenia. For example, Amazon Comprehend Medical is a service that uses NLP to extract disease conditions, medications and treatment outcomes from patient notes, clinical trial reports and other electronic health records.
- Organizations can determine what customers are saying about a service or product by identifying and extracting information in sources like social media. This **sentiment analysis** can provide a lot of information about customers choices and their decision drivers.
- An inventor at IBM developed a **cognitive assistant** that works like a personalized search engine by learning all about you and then remind you of a name, a song, or anything you can't remember the moment you need it to.
- Companies like Yahoo and Google filter and classify your emails with NLP by analyzing text in emails that flow through their servers and **stopping spam** before they even enter your inbox.
- To help **identifying fake news**, the NLP Group at MIT developed a new system to determine if a source is accurate or politically biased, detecting if a news source can be trusted or not.

- Amazon's Alexa and Apple's Siri are examples of intelligent **voice driven interfaces** that use NLP to respond to vocal prompts and do everything like find a particular shop, tell us the weather forecast, suggest the best route to the office or turn on the lights at home.
- Having an insight into what is happening and what people are talking about can be very valuable to **financial traders**. NLP is being used to track news, reports, comments about possible mergers between companies, everything can be then incorporated into a trading algorithm to generate massive profits. Remember: buy the rumor, sell the news.
- NLP is also being used in both the search and selection phases of **talent recruitment**, identifying the skills of potential hires and also spotting prospects before they become active on the job market.
- Powered by IBM Watson NLP technology, LegalMation developed a platform to automate routine **litigation tasks** and help legal teams save time, drive down costs and shift strategic focus.

NLP is particularly booming in the **healthcare industry**. This technology is improving care delivery, disease diagnosis and bringing costs down while healthcare organizations are going through a growing adoption of electronic health records. The fact that clinical documentation can be improved means that patients can be better understood and benefited through better healthcare. The goal should be to optimize their experience, and several organizations are already working on this.



Number of publications containing the sentence “natural language processing” in PubMed in the period 1978–2018. As of 2018, PubMed comprised more than 29 million citations for biomedical literature

Companies like Winterlight Labs are making huge improvements in the treatment of Alzheimer’s disease by monitoring cognitive impairment through speech and they can also support clinical trials and studies for a wide range of central nervous system disorders. Following a similar approach, Stanford University developed Woebot, a **chatbot therapist** with the aim of helping people with anxiety and other disorders.

But serious controversy is around the subject. A couple of years ago Microsoft demonstrated that by analyzing large samples of search engine queries, they could identify internet users who were suffering from pancreatic cancer even before they have received a diagnosis of the disease. How would users react to such diagnosis? And what would happen if you were tested as a false positive? (meaning that you can be diagnosed with the disease even though you don’t have it). This recalls the case of Google Flu Trends which in 2009 was announced as being able to predict influenza but later on vanished due to its low accuracy and inability to meet its projected rates.

NLP may be the key to an effective clinical support in the future, but there are still many challenges to face in the short term.

Basic NLP to impress your non-NLP friends

The main drawbacks we face these days with NLP relate to the fact that language is very tricky. The process of understanding and manipulating language is extremely complex, and for this reason it is common to use different techniques to handle different challenges before binding everything together. Programming languages like Python or R are highly used to perform these techniques, but before diving into code lines (that will be the topic of a different article), it's important to understand the concepts beneath them. Let's summarize and explain some of the most frequently used algorithms in NLP when defining the vocabulary of terms:

Bag of Words

Is a commonly used model that allows you to count all words in a piece of text. Basically it creates an occurrence matrix for the sentence or document, disregarding grammar and word order. These word frequencies or occurrences are then used as features for training a classifier.

To bring a short example I took the first sentence of the song “Across the Universe” from The Beatles:

Words are flowing out like endless rain into a paper cup,

They slither while they pass, they slip away across the universe

Now let's count the words:

	words	rain	a	paper	they	slip	the	universe	...
<i>Words are flowing out like endless rain into a paper cup,</i>	1	1	1	1	0	0	0	0	...
<i>They slither while they pass, they slip away across the universe</i>	0	0	0	0	3	1	1	1	...

This approach may reflect several downsides like the absence of semantic meaning and context, and the facts that stop words (like “the” or “a”) add noise to the analysis and some words are not weighted accordingly (“universe” weights less than the word “they”).

To solve this problem, one approach is to rescale the frequency of words by how often they appear in all texts (not just the one we are analyzing) so that the scores for frequent words like “the”, that are also frequent across other texts, get penalized. This approach to scoring is called **“Term Frequency — Inverse Document Frequency” (TFIDF)**, and improves the bag of words by weights. Through TFIDF frequent terms in the text are “rewarded” (like the word “they” in our example), but they also get “punished” if those terms are frequent in other texts we include in the algorithm too. On the contrary, this method highlights and “rewards” unique or rare terms considering all texts. Nevertheless, this approach still has no context nor semantics.

Tokenization

Is the process of segmenting running text into sentences and words. In essence, it’s the task of cutting a text into pieces called *tokens*, and at the same time throwing away certain characters, such as punctuation. Following our example, the result of tokenization would be:

Words	are	flowing	out	like	endless	rain	into	a	paper	cup
They	slither	while	they	pass	they	slip	away	across	the	universe

Pretty simple, right? Well, although it may seem quite basic in this case and also in languages like English that separate words by a blank space (called segmented languages) not all languages behave the same, and if you think about it, blank spaces alone are not sufficient enough even for English to perform proper tokenizations. Splitting on blank spaces may break up what should be considered as one token, as in the case of certain names (e.g. San Francisco or New York) or borrowed foreign phrases (e.g. *laissez faire*).

Tokenization can remove punctuation too, easing the path to a proper word segmentation but also triggering possible complications. In the case of periods that follow abbreviation (e.g. dr.), the period following that abbreviation should be considered as part of the same token and not be removed.

The tokenization process can be particularly problematic when dealing with biomedical text domains which contain lots of hyphens, parentheses, and other punctuation marks.

For deeper details on tokenization, you can find a great explanation in this [article](#).

Stop Words Removal

Includes getting rid of common language articles, pronouns and prepositions such as “and”, “the” or “to” in English. In this process some very common words that appear to provide little or no value to the NLP objective are filtered and excluded from the text to be processed, hence removing widespread and frequent terms that are not informative about the corresponding text.

Stop words can be safely ignored by carrying out a lookup in a pre-defined list of keywords, freeing up database space and improving processing time.

There is no universal list of stop words. These can be pre-selected or built from scratch. A potential approach is to begin by adopting pre-defined stop words and add words to the list later on.

Nevertheless it seems that the general trend over the past time has been to go from the use of large standard stop word lists to the use of no lists at all.

The thing is stop words removal can wipe out relevant information and modify the context in a given sentence. For example, if we are performing a sentiment analysis we might throw our algorithm off track if we remove a stop word like “not”. Under these conditions, you might select a minimal stop word list and add additional terms depending on your specific objective.

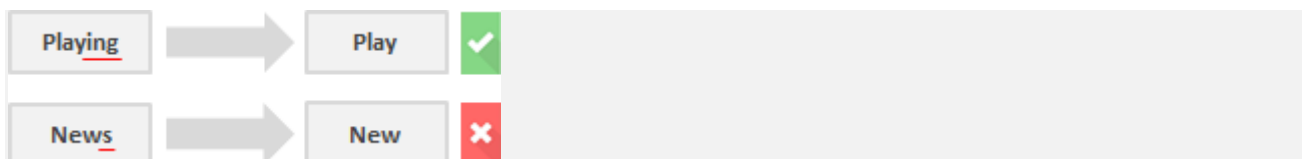
Stemming

Refers to the process of slicing the end or the beginning of words with the intention of removing affixes (lexical additions to the root of the word).

Affixes that are attached at the beginning of the word are called prefixes (e.g. “astro” in the word “astrobiology”) and the ones attached at the end of the word are called suffixes (e.g. “ful” in the word “helpful”).

The problem is that affixes can create or expand new forms of the same word (called *inflectional* affixes), or even create new words themselves (called *derivational* affixes). In English, prefixes are always derivational (the affix creates a new word as in the example of the prefix “eco” in the word “ecosystem”), but suffixes can be derivational (the affix creates a new word as in the example of the suffix “ist” in the word “guitarist”) or inflectional (the affix creates a new form of word as in the example of the suffix “er” in the word “faster”).

Ok, so how can we tell the difference and chop the right bit?



A possible approach is to consider a list of common affixes and rules (Python and R languages have different libraries containing affixes and methods) and perform stemming based on them, but of course this approach presents limitations. Since stemmers use algorithmic approaches, the result of the stemming process may not be an actual word or even change the word (and sentence) meaning. To offset this effect you can edit those predefined methods by adding or removing affixes and rules, but you must consider that you might be improving the performance in one area while producing a degradation in another one. Always look at the whole picture and test your model's performance.

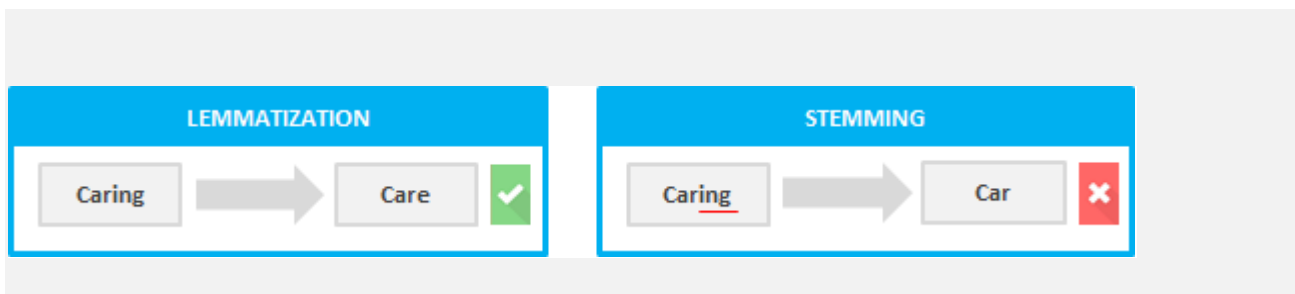
So if stemming has serious limitations, why do we use it? First of all, it can be used to correct spelling errors from the tokens. **Stemmers are simple to use and run very fast** (they perform simple operations on a string), and if speed and performance are important in the NLP model, then stemming is certainly the way to go. Remember, we use it with the objective of improving our performance, not as a grammar exercise.

Lemmatization

Has the objective of reducing a word to its base form and grouping together different forms of the same word. For example, verbs in past tense are changed into present (e.g. “went” is changed to “go”) and synonyms are unified (e.g. “best” is changed to “good”), hence standardizing words with similar meaning to their root. Although it seems closely related to the stemming process, lemmatization uses a different approach to reach the root forms of words.

Lemmatization resolves words to their dictionary form (known as lemma) for which it requires detailed dictionaries in which the algorithm can look into and link words to their corresponding lemmas.

For example, the words “*running*”, “*runs*” and “*ran*” are all forms of the word “*run*”, so “*run*” is the lemma of all the previous words.



Lemmatization also takes into consideration the context of the word in order to **solve other problems like disambiguation**, which means it can discriminate between identical words that have different meanings depending on the specific context. Think about words like “bat” (which can correspond to the animal or to the metal/wooden club used in baseball) or “bank” (corresponding to the financial institution or to the land alongside a body of water). By providing a part-of-speech parameter to a word (whether it is a noun, a verb, and so on) it’s possible to define a role for that word in the sentence and remove disambiguation.

As you might already pictured, lemmatization is a much more resource-intensive task than performing a stemming process. At the same time, since it requires more knowledge about the language structure than a stemming approach, it **demand more computational power** than setting up or adapting a stemming algorithm.

Topic Modeling

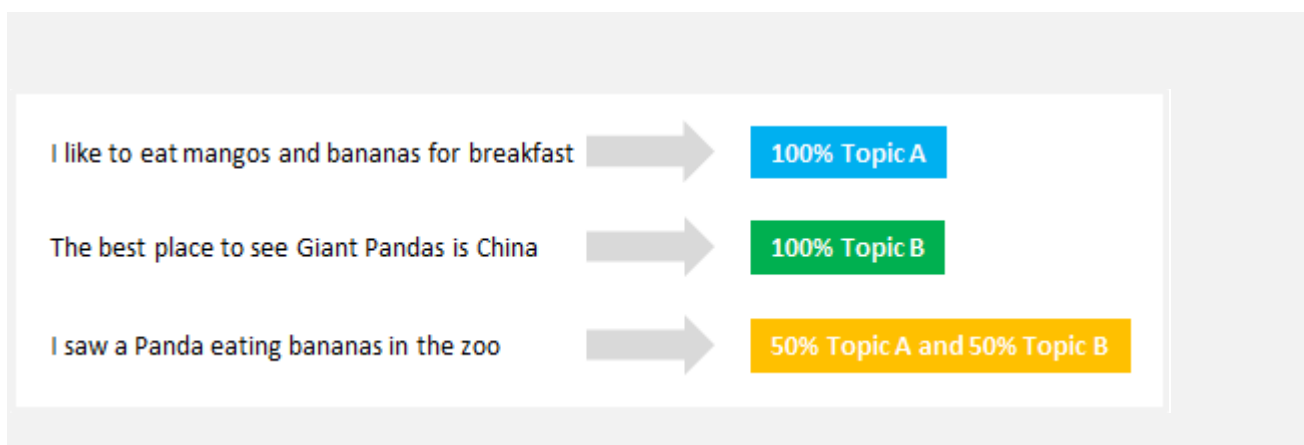
Is as a method for uncovering hidden structures in sets of texts or documents. In essence it clusters texts to discover latent topics based on their contents, processing individual words and assigning them values based on their distribution. This technique is based on the assumptions that each document consists of a mixture of topics and that each topic consists of a set of words, which means that if we can spot these hidden topics we can unlock the meaning of our texts.

From the universe of topic modelling techniques, **Latent Dirichlet Allocation (LDA)** is probably the most commonly used. This relatively new algorithm (invented less than 20 years ago) works as an unsupervised learning method that discovers different topics underlying a collection of documents.

In **unsupervised learning** methods like this one, there is no output variable to guide the learning process and data is explored by algorithms to find patterns. To be more specific, LDA finds groups of related words by:

1. Assigning each word to a random topic, where the user defines the number of topics it wishes to uncover. You don't define the topics themselves (you define just the number of topics) and the algorithm will map all documents to the topics in a way that words in each document are mostly captured by those imaginary topics.
2. The algorithm goes through each word iteratively and reassigns the word to a topic taking into considerations the probability that the word belongs to a topic, and the probability that the document will be generated by a topic. These probabilities are calculated multiple times, until the convergence of the algorithm.

Unlike other clustering algorithms like *K-means* that perform hard clustering (where topics are disjointed), LDA assigns each document to a mixture of topics, which means that each document can be described by one or more topics (e.g. Document 1 is described by 70% of topic A, 20% of topic B and 10% of topic C) and reflect more realistic results.



Topic modeling is extremely useful for classifying texts, building recommender systems (e.g. to recommend you books based on your past readings) or even detecting trends in online publications.

Machine Learning Model

Organizations are implementing AI projects for numerous applications in a wide range of industries. These applications include predictive analytics, pattern recognition systems, autonomous systems, conversational systems, hyper-personalization activities and goal-driven systems. Each of these projects has something in common: They're all predicated on an understanding of the business problem and that data and machine learning algorithms must be applied to the problem, resulting in a machine learning model that addresses the project's needs.

Deploying and managing machine learning projects typically follow the same pattern. However, existing app development methodologies don't apply because AI projects are driven by data, not programming code. The learning is derived from data. The right machine learning approach and methodologies stem from data-centric needs and result in projects that focus on working through the stages of data discovery, cleansing, training, model building and iteration.

7 steps to building a machine learning model

For many organizations, machine learning model development is a new activity and can seem intimidating. Even for those with experience in machine learning, building an AI model requires diligence, experimentation and creativity. The methodology for building data-centric projects, however, is somewhat established. The following steps will help guide your project.

Step 1. Understand the business problem (and define success)

The first phase of any machine learning project is developing an understanding of the business requirements. You need to know what problem you're trying to solve before attempting to solve it.

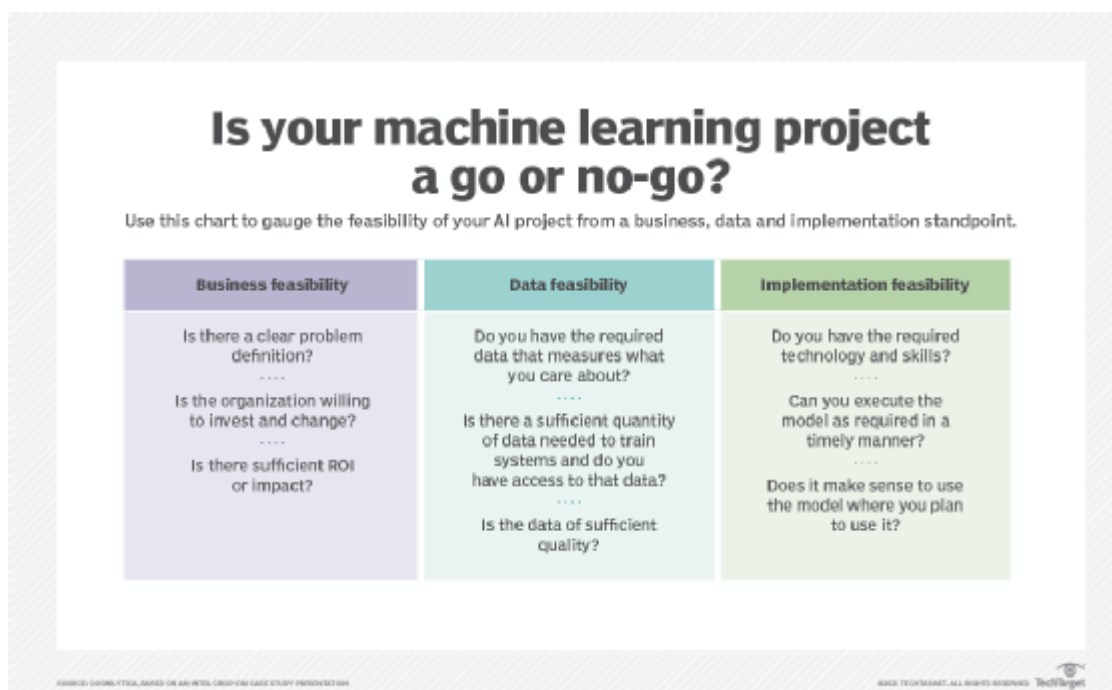
To start, work with the owner of the project and make sure you understand the project's objectives and requirements. The goal is to convert this knowledge into a suitable problem definition for the machine learning project and devise a preliminary plan for achieving the project's objectives. Key questions to answer include the following:

- What's the business objective that requires a cognitive solution?
- What parts of the solution are cognitive, and what aren't?
- Have all the necessary technical, business and deployment issues been addressed?
- What are the defined "success" criteria for the project?
- How can the project be staged in iterative sprints?

- Are there any special requirements for transparency, explainability or bias reduction?
- What are the ethical considerations?
- What are the acceptable parameters for accuracy, precision and confusion matrix values?
- What are the expected inputs to the model and the expected outputs?
- What are the characteristics of the problem being solved? Is this a classification, regression or clustering problem?
- What is the "heuristic" -- the quick-and-dirty approach to solving the problem that doesn't require machine learning? How much better than the heuristic does the model need to be?
- How will the benefits of the model be measured?

Although there are a lot of questions to be answered during the first step, answering or even attempting to answer them will greatly increase the chances of overall project success.

Setting specific, quantifiable goals will help realize measurable ROI from the machine learning project instead of simply implementing it as a proof of concept that'll be tossed aside later. The goals should be related to the business objectives and not just to machine learning. While machine learning-specific measures -- such as precision, accuracy, recall and mean squared error -- can be included in the metrics, more specific, business-relevant key performance indicators (KPIs) are better.



In order for a machine learning project to go forward, you need to determine the feasibility of the effort from a business, data and implementation standpoint.

Step 2. Understand and identify data

Once you have a firm understanding of the business requirements and receive approval for the plan, you can start to build a machine learning model, right? Wrong. Establishing the business case doesn't mean you have the data needed to create the machine learning model.

A machine learning model is built by learning and generalizing from training data, then applying that acquired knowledge to new data it has never seen before to make predictions and fulfill its purpose. Lack of data will prevent you from building the model, and access to data isn't enough. Useful data needs to be clean and in a good shape.

Identify your data needs and determine whether the data is in proper shape for the machine learning project. The focus should be on data identification, initial collection, requirements, quality identification, insights and potentially interesting aspects that are worth further investigation. Here are some key questions to consider:

- Where are the sources of the data that's needed for training the model?
- What quantity of data is needed for the machine learning project?
- What is the current quantity and quality of training data?
- How are the test set data and training set data being split?
- For supervised learning tasks, is there a way to label that data?
- Can pre-trained models be used?
- Where is the operational and training data located?
- Are there special needs for accessing real-time data on edge devices or in more difficult-to-reach places?

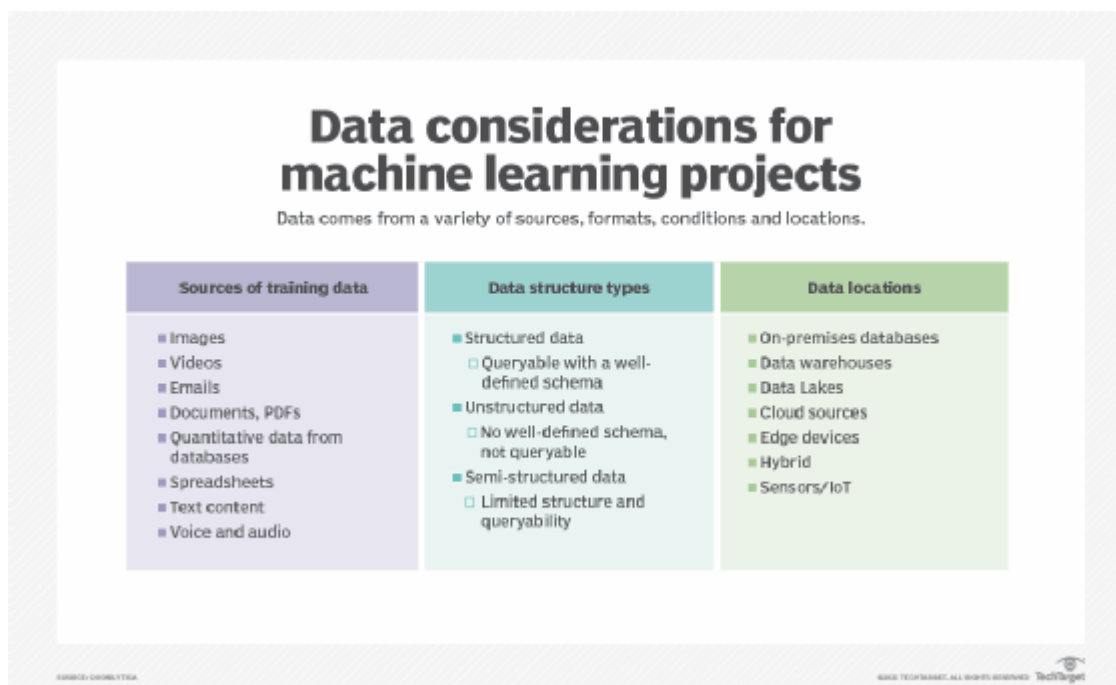
Answering these important questions helps you get a handle on the quantity and quality of data as well as understand the type of data that's needed to make the model work.

In addition, you need to know how the model will operate on real-world data. For example, will the model be used offline, operate in batch mode on data that's fed in and processed asynchronously, or

be used in real time, operating with high-performance requirements to provide instant results? This information will also determine the sort of data needed and data access requirements.

Determine also whether the model will be trained once, in iterations with versions of it deployed periodically or in real time. Real-time training imposes many requirements on data that might not be feasible for some setups.

During this phase of the AI project, it's also important to know if any differences exist between real-world data and training data as well as test data and training data, and what approach you will take to validate and evaluate the model for performance.



The above chart outlines different kinds of data and sources needed for machine learning projects.

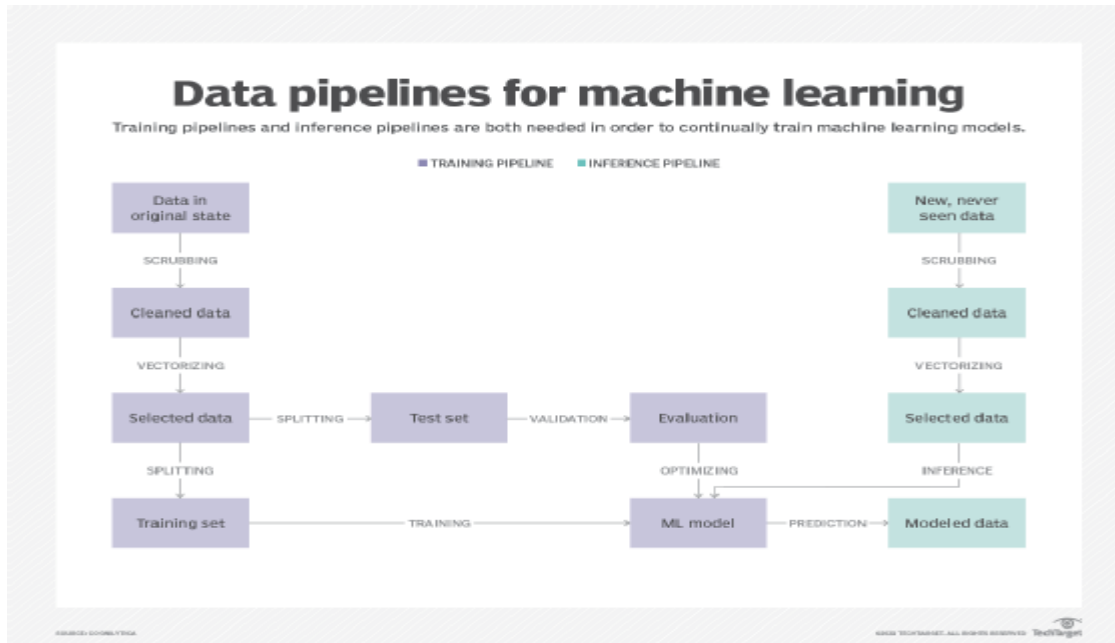
Step 3. Collect and prepare data

Once you've appropriately identified your data, you need to shape that data so it can be used to train your model. The focus is on data-centric activities necessary to construct the data set to be used for modeling operations. Data preparation tasks include data collection, cleansing, aggregation, augmentation, labeling, normalization and transformation as well as any other activities for structured, unstructured and semi-structured data.

Procedures during the data preparation, collection and cleansing process include the following:

- Collect data from the various sources.
- Standardize formats across different data sources.
- Replace incorrect data.
- Enhance and augment data.
- Add more dimensions with pre-calculated amounts and aggregate information as needed.
- Enhance data with third-party data.
- "Multiply" image-based data sets if they aren't sufficient enough for training.
- Remove extraneous information and deduplication.
- Remove irrelevant data from training to improve results.
- Reduce noise reduction and remove ambiguity.
- Consider anonymizing data.
- Normalize or standardize data to get it into formatted ranges.
- Sample data from large data sets.
- Select features that identify the most important dimensions and, if necessary, reduce dimensions using a variety of techniques.
- Split data into training, test and validation sets.

Data preparation and cleansing tasks can take a substantial amount of time. Surveys of machine learning developers and data scientists show that the data collection and preparation steps can take up to 80% of a machine learning project's time. As the saying goes, "garbage in, garbage out." Since machine learning models need to learn from data, the amount of time spent on prepping and cleansing is well worth it.



The above chart is an overview of the training and inference pipelines used in developing and updating machine learning models.

Step 4. Determine the model's features and train it

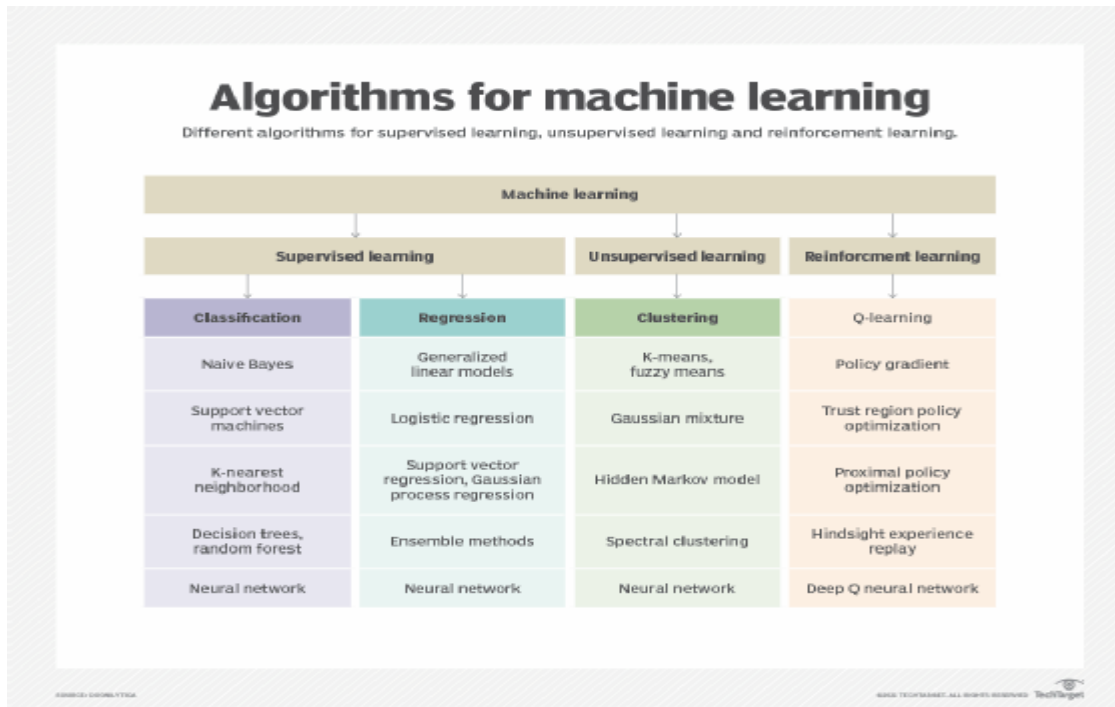
Once the data is in usable shape and you know the problem you're trying to solve, it's finally time to move to the step you long to do: Train the model to learn from the good quality data you've prepared by applying a range of techniques and algorithms.

This phase requires model technique selection and application, model training, model hyperparameter setting and adjustment, model validation, ensemble model development and testing, algorithm selection, and model optimization. To accomplish all that, the following actions are required:

- Select the right algorithm based on the learning objective and data requirements.
- Configure and tune hyperparameters for optimal performance and determine a method of iteration to attain the best hyperparameters.
- Identify the features that provide the best results.
- Determine whether model explainability or interpretability is required.

- Develop ensemble models for improved performance.
- Test different model versions for performance.
- Identify requirements for the model's operation and deployment.

The resulting model can then be evaluated to determine whether it meets the business and operational requirements.



In machine learning, an algorithm is the formula or set of instructions to follow to record experience and improve learning over time. Depending on what type of machine learning approach you are doing, different algorithms perform better than others.

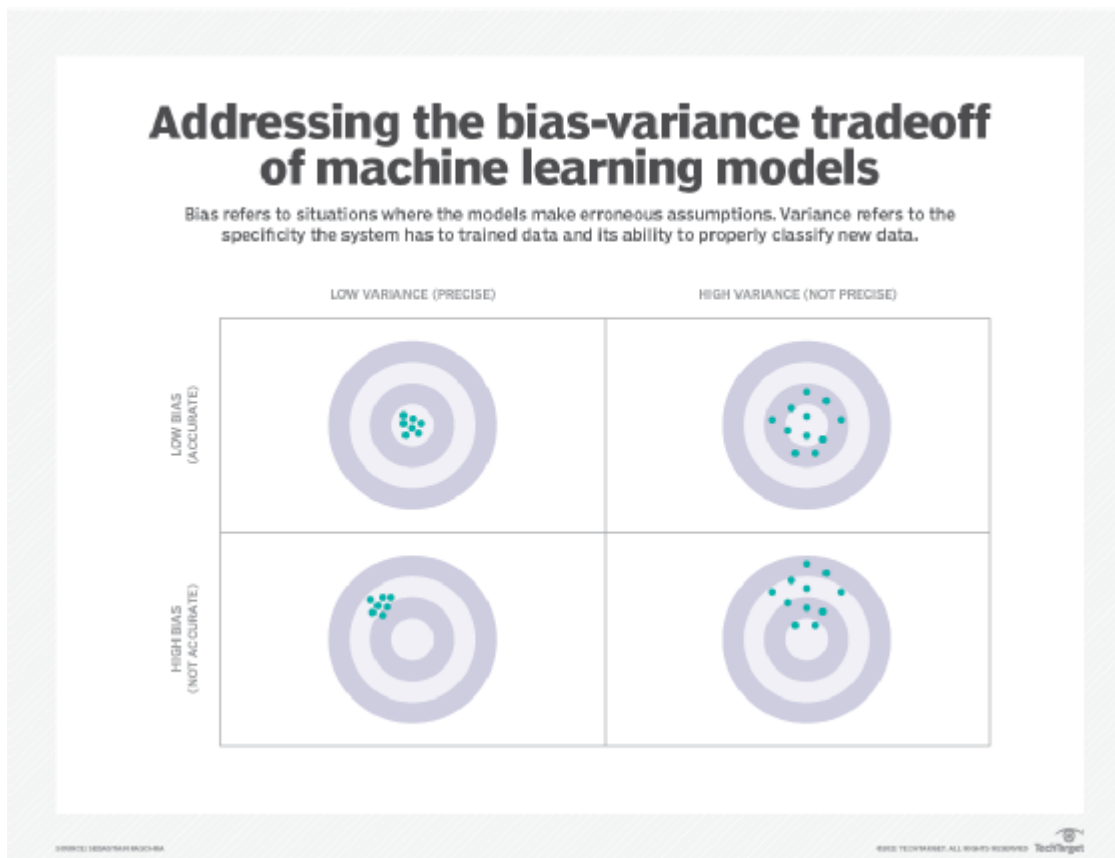
Step 5. Evaluate the model's performance and establish benchmarks

From an AI perspective, evaluation includes model metric evaluation, confusion matrix calculations, KPIs, model performance metrics, model quality measurements and a *final* determination of whether the model can meet the established business goals. During the model evaluation process, you should do the following:

- Evaluate the models using a validation data set.

- Determine confusion matrix values for classification problems.
- Identify methods for k-fold cross-validation if that approach is used.
- Further tune hyperparameters for optimal performance.
- Compare the machine learning model to the baseline model or heuristic.

Model evaluation can be considered the quality assurance of machine learning. Adequately evaluating model performance against metrics and requirements determines how the model will work in the real world.



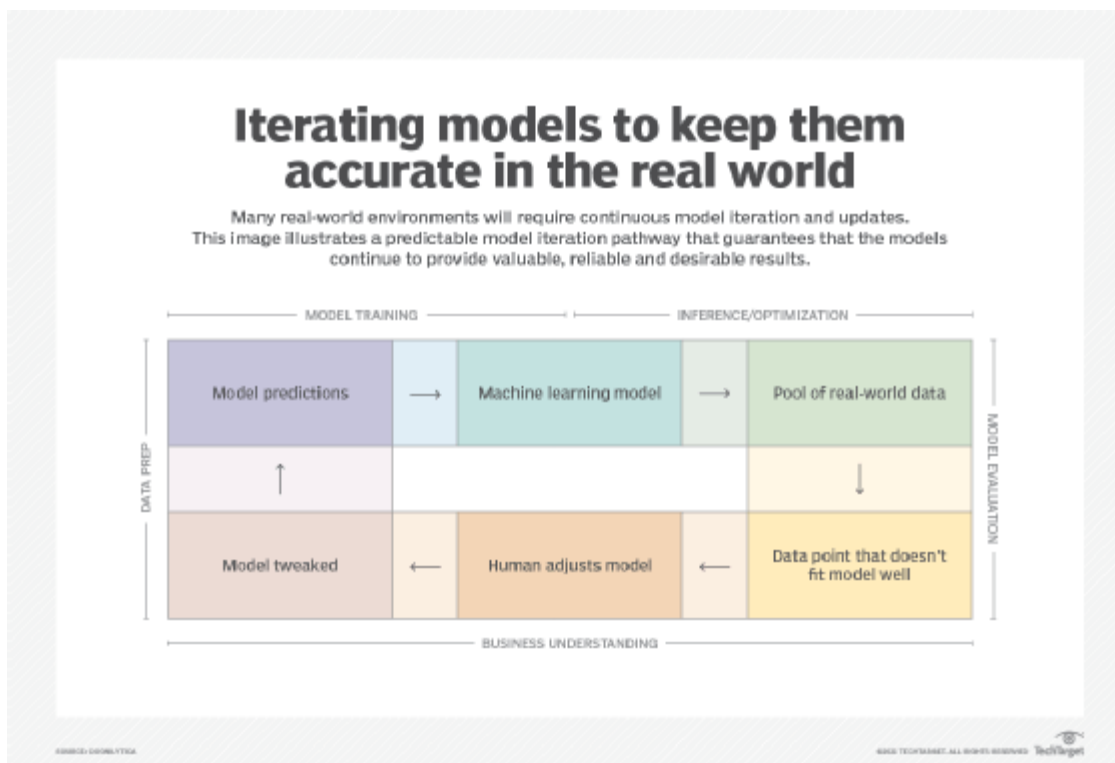
Understanding the concepts of bias and variance helps you find the sweet spot for optimizing the performance of your machine learning models.

Step 6. Put the model in operation and make sure it works well

When you're confident that the machine learning model can work in the real world, it's time to see how it actually operates in the real world -- also known as "operationalizing" the model:

- Deploy the model with a means to continually measure and monitor its performance.
- Develop a baseline or benchmark against which future iterations of the model can be measured.
- Continuously iterate on different aspects of the model to improve overall performance.

Model operationalization might include deployment scenarios in a cloud environment, at the edge, in an on-premises or closed environment, or within a closed, controlled group. Among operationalization considerations are model versioning and iteration, model deployment, model monitoring and model staging in development and production environments. Depending on the requirements, model operationalization can range from simply generating a report to a more complex, multi-endpoint deployment.



Successful AI projects iterate models to ensure the models continue to provide valuable, reliable and desirable results in the real world.

Step 7. Iterate and adjust the model

Even though the model is operational and you're continuously monitoring its performance, you're not done. When it comes to implementing technologies, it's often said that the formula for success is to start small, think big and iterate often.

Always repeat the process and make improvements in time for the next iteration. Business requirements change. Technology capabilities change. Real-world data changes in unexpected ways. All of which might create new requirements for deploying the model onto different endpoints or in new systems. The end may just be a new beginning, so it's best to determine the following:

- the next requirements for the model's functionality;
- expansion of model training to encompass greater capabilities;
- improvements in model performance and accuracy;
- improvements in model operational performance;
- operational requirements for different deployments; and
- solutions to "model drift" or "data drift," which can cause changes in performance due to changes in real-world data.

Reflect on what has worked in your model, what needs work and what's a work in progress. The surefire way to achieve success in machine learning model building is to continuously look for improvements and better ways to meet evolving business requirements.

Need of Data Science:

The reason why we need data science is the ability to process and interpret data. This enables companies to make informed decisions around growth, optimization, and performance. Demand for skilled data scientists is on the rise now and in the next decade. For example, machine learning is now being used to make sense of every kind of data – big or small. Data metrics are driving every business decision. The job market scenario for data scientists will grow to almost 11.5M by 2026 [U.S Bureau of Labor Statistics]. Companies are busy ramping up their data science workforce to enable higher efficiency and planning.

Ethics of Data Science:

The twin motors of data and information technology are driving innovation forward in most every aspect of human enterprise. In a similar fashion, Data Science today profoundly influences how

business is done in fields as diverse as the life sciences, smart cities, and transportation. As cogent as these directions have become, the dangers of data science without ethical considerations is as equally apparent — whether it be the protection of personally identifiable data, implicit bias in automated decision-making, the illusion of free choice in psychographics, the social impacts of automation, or the apparent divorce of truth and trust in virtual communication. Justifying the need for focus on the ethics of data science goes beyond a balance sheet of these opportunities and challenges, for the practice of data science challenges our perceptions of what it means to be human.

If ethics is defined as shared values that help humanity differentiate right from wrong, the increasing digitalization of human activity shapes the very definitions of how we evaluate the world around us.

Margo Boenig-Liptsin points out that our ever-increasing reliance on information technology has fundamentally transformed traditional concepts of “privacy”, “fairness” and “representation”, not to mention “free choice”, “truth” and “trust”. These mutations underline the increasing footprint and responsibilities of data science — beyond the bytes and bits, data science shakes the perceptual foundations of value, community, and equity. If academia has been quick to establish Data Science programs around statistics, computation and software engineering, few programs address the larger societal concerns of data science, and fewer still analyse how responsible data practices can be conditioned and even encouraged. Let’s sketch out the contours of this challenge.

Data citizens

Perhaps no area of ethics in data science has received more attention today than the protection of personal data. *The digital transformation of our interactions with social and economics communities reveals who we are, what we think, and what we do.* The recent introduction of legislation in Europe (GDPR), India (the Personal Data Protection Act, 2018), and California (the California Consumer Privacy Act of 2018) specifically recognize the rights of digital citizens, and implicitly address the dangers of the commercial use of personal and personally identifiable data. These legal frameworks

try to rebalance the inequitable relationships of power and influence between organizations and individuals in codifying ethical benchmarks including the right to be informed, the right to object, the right of access, the right to rectification, and the right to be forgotten.

The attention given to this legislation extends far beyond concerns for data protection. These attempts to define proper and illicit data practices respond to a number of ethical questions. As data become the new currency of the world economy, the lines between public and private, between individuals and society, and between the resource rich and the resource poor are being redrawn. Who owns personal data, and which rights can be assigned with explicit or implicit consent? To what extent should the public and private organizations be able to collect and control the vast records of our human interaction? To what extent should these data controllers and data processors be held accountable for the loss or misuse of our data?

Automated decision-making

The ability to conscientiously take decisions among alternative possibilities has long been viewed as a condition that separates man (or at least the living) from machines. As innovations in data science progress in algorithmic trading, self-driving cars, and robotics, the distinction between human and artificial intelligence is becoming increasingly difficult to distinguish. Current applications of machine learning cross the threshold of decision support systems and enter the realm of artificial intelligence where sophisticated algorithms are designed to replace human decision-making.

Crossing this threshold introduces several ethical considerations. Can economic and/or social organizations rely on increasingly complex methodologies in which many understand neither the assumptions nor the limits of the underlying models? Are we willing to accept that these applications, which by their very nature, learn from our experience — making us prisoners of our past and limiting our potential for growth and diversity? Do we understand that the inherent logic of these platforms

can be gamed — which creates opportunities to “cheat” the system? Last and but not least, who is legally responsible for the implicit bias inherent in automated decision -making?

Micro-targeting

John Battelle suggested years ago that *our digital footprints provide indelible roadmaps through the database of our intentions*. The leitmotif of Data Science has been to help organizations understand the objectives, motivations and actions of both individuals and communities. The work of Michal Kosinski and David Stillwell has promised even more further in suggesting that the pertinence of prescriptive analytics can be greatly enhanced in focusing on patterns of behavior (personality traits, beliefs, values, attitudes, interests, or lifestyles) rather than clusters of demographic data.

Applications of micro-targeting has since been pitched as powerful tools of influence in the fields of marketing, politics and economics. Even if an individual’s ability to exercise “free choice” has long been a subject of debate, the practice of feeding consumers only information that they will agree with binds rationality even further. Moreover, micro-targeting techniques allow researchers extrapolate sensitive information and personal preferences of individuals even when such data is not specifically captured. Finally, as the “client becomes the product”, there is a real danger that data science is used less to improve an organization’s product or service offering than to turn consumers into objects of manipulation.

Distributed ledgers

The goal of Information technology has long been to provide a single version of the truth to facilitate exchanges of products, services and ideas. For a number of reasons tied to the evolution of both the global economy and national markets, *a perceptual gap has grown between this “ground truth” and the trust consumers have in the intermediaries (like the State, the banks, and the corporations) that capture, collect, and monetize this data*. The social mechanics of the World Wide Web distort the

relationship even further, putting on equal footing fact and fiction, favoring the extremes to the banality of normality.

Distributed ledger technologies in general, and blockchain technologies in particular, offer their share of hope both for a more transparent and traceable source of information. Yet this vision of an Internet of Value is partially clouded by the potential societal challenges of relatively untested technologies. Can technology in and of itself be the standard for both truth and trust? To what degree will people and organizations accept the primacy of transparency? On what basis can social values like *the right to be forgotten* be reconciled with the technical requirements of public ledgers? Freed from the conventions and the logic of financial capitalism, can human nature accept a radically different basis for the distribution of wealth?

Human and machine intelligence

Although the impact of information technology on the organization of private and public enterprise has been largely debated over the last four decades, the impact of data science on the function of management has received considerably less attention. In the trade press, technology is often seen as ethically neutral, proving a digital mirror of the dominant managerial paradigms at any point in time. In academia the relationship is subject to closer scrutiny, authors like Latour, Callon and Law have demonstrated how different forms of technology influence the way that managers, employees and customers perceive the reality of social and economic exchanges, markets and industries.

Doing Good Data Science:

The hard thing about being an ethical data scientist isn't understanding ethics. It's the junction between ethical ideas and practice. It's doing good data science.

There has been a lot of healthy discussion about data ethics lately. We want to be clear: that discussion is good, and necessary. But it's also not the biggest problem we face. We already have good standards for data ethics. The ACM's code of ethics, which dates back to 1993, and is

currently being updated, is clear, concise, and surprisingly forward-thinking; 25 years later, it's a great start for anyone thinking about ethics. The American Statistical Association has a good set of ethical guidelines for working with data. So, we're not working in a vacuum.

And we believe that most people want to be fair. Data scientists and software developers don't want to harm the people using their products. There are exceptions, of course; we call them criminals and con artists. Defining "fairness" is difficult, and perhaps impossible, given the many crosscutting layers of "fairness" that we might be concerned with. But we don't have to solve that problem in advance, and it's not going to be solved in a simple statement of ethical principles, anyway.

The problem we face is different: how do we put ethical principles into practice? We're not talking about an abstract commitment to being fair. Ethical principles are worse than useless if we don't allow them to change our practice, if they don't have any effect on what we do day-to-day. For data scientists, whether you're doing classical data analysis or leading-edge AI, that's a big challenge. We need to understand how to build the software systems that implement fairness. That's what we mean by doing good data science.

Any code of data ethics will tell you that you shouldn't collect data from experimental subjects without informed consent. But that code won't tell you how to implement "informed consent." Informed consent is easy when you're interviewing a few dozen people in person for a psychology experiment. Informed consent means something different when someone clicks an item in an online catalog (hello, Amazon), and ads for that item start following them around *ad infinitum*. Do you use a pop-up to ask for permission to use their choice in targeted advertising? How many customers would you lose if you did so? Informed consent means something yet again when you're asking someone to fill out a profile for a social site, and you might (or might not) use that data for any number of experimental purposes. Do you pop up a consent form in impenetrable legalese that basically says "we will use your data, but we don't know for what"? Do you phrase this agreement as an opt-out, and hide it somewhere on the site where nobody will find it?

That's the sort of question we need to answer. And we need to find ways to share best practices. After the ethical principle, we have to think about the implementation of the ethical principle. That isn't easy; it encompasses everything from user experience design to data management. How do we design the user experience so that our concern for fairness and ethics doesn't make an application unuseable? Bad as it might be to show users a pop-up with thousands of words of legalese, laboriously guiding users through careful and lengthy explanations isn't likely to meet with approval, either. How do we manage any sensitive data that we acquire? It's easy to say that applications shouldn't collect data about race, gender, disabilities, or other protected classes. But if you don't gather that data, you will have trouble testing whether your applications are fair to minorities. Machine learning has proven to be very good at figuring its own proxies for race and

other classes. Your application wouldn't be the first system that was unfair despite the best intentions of its developers. Do you keep the data you need to test for fairness in a separate database, with separate access controls?

To put ethical principles into practice, we need space to be ethical. We need the ability to have conversations about what ethics means, what it will cost, and what solutions to implement. As technologists, we frequently share best practices at conferences, write blog posts, and develop open source technologies—but we rarely discuss problems such as how to obtain informed consent.

There are several facets to this space that we need to think about.

Foremost, we need corporate cultures in which discussions about fairness, about the proper use of data, and about the harm that can be done by inappropriate use of data can be considered. In turn, this means that we can't rush products out the door without thinking about how they're used. We can't allow "internet time" to mean ignoring the consequences. Computer security has shown us the consequences of ignoring the consequences: many companies that have never taken the time to implement good security practices and safeguards are now paying with damage to their reputations and their finances. We need to do the same when thinking about issues like fairness, accountability, and unintended consequences.

We particularly need to think about the unintended consequences of our use of data. It will never be possible to predict all the unintended consequences; we're only human, and our ability to foresee the future is limited. But plenty of unintended consequences could easily have been foreseen: for example, Facebook's "Year in Review" that reminded people of deaths and other painful events. Moving fast and breaking things is unacceptable if we don't think about the things we are likely to break. And we need the space to do that thinking: space in project schedules, and space to tell management that a product needs to be rethought.

We also need space to stop the production line when something goes wrong. This idea goes back to Toyota's **Kanban**: any assembly line worker can stop the line if they see something going wrong. The line doesn't restart until the problem is fixed. Workers don't have to fear consequences from management for stopping the line; they are trusted, and expected to behave responsibly. What would it mean if we could do this with product features? If anyone at Facebook could have said "wait, we're getting complaints about Year in Review" and pulled it out of production until someone could investigate what was happening?

It's easy to imagine the screams from management. But it's not hard to imagine a Toyota-style "stop button" working. After all, Facebook is the poster child for continuous deployment, and they've often talked about how new employees push changes to production on their first day. Why not let

employees pull features out of production? Where are the tools for instantaneous undeployment? They certainly exist; continuous deployment doesn't make sense if you can't roll back changes that didn't work. Yes, Facebook is a big, complicated company, with a big complicated product. So is Toyota. It worked for them.

The issue lurking behind all of these concerns is, of course, corporate culture. Corporate environments can be hostile to anything other than short-term profitability. That's a consequence of poor court decisions and economic doctrine, particularly in the US. But that inevitably leads us to the biggest issue: how to move the needle on corporate culture. Susan Etlinger has suggested that, in a time when public distrust and disenchantment is running high, ethics is a good investment. Upper-level management is only starting to see this; changes to corporate culture won't happen quickly.

Users want to engage with companies and organizations they can trust not to take unfair advantage of them. Users want to deal with companies that will treat them and their data responsibly, not just as potential profit or engagement to be maximized. Those companies will be the ones that create space for ethics within their organizations. We, the data scientists, data engineers, AI and ML developers, and other data professionals, have to demand change. We can't leave it to people that "do" ethics. We can't expect management to hire trained ethicists and assign them to our teams. We need to live ethical values, not just talk about them. We need to think carefully about the consequences of our work. We must create space for ethics within our organizations. Cultural change may take time, but it will happen—if we are that change. That's what it means to do good data science.

Five Cs of Data Science:

Five framing guidelines help us think about building data products. We call them the five Cs: consent, clarity, consistency, control (and transparency), and consequences (and harm). They're a framework for implementing the golden rule for data. Let's look at them one at a time.

Consent

You can't establish trust between the people who are providing data and the people who are using it without agreement about what data is being collected and how that data will be used. Agreement starts with obtaining consent to collect and use data. Unfortunately, the agreements between a service's users (people whose data is collected) and the service itself (which uses the data in many ways) are binary (meaning that you either accept or decline) and lack clarity. In business, when contracts are being negotiated between two parties, there are multiple iterations (redlines) before the

contract is settled. But when a user is agreeing to a contract with a data service, you either accept the terms or you don't get access. It's non-negotiable.

For example, when you check in to a hospital you are required to sign a form that gives them the right to use your data. Generally, there's no way to say that your data can be used for some purposes but not others. When you sign up for a loyalty card at your local pharmacy, you're agreeing that they can use your data in unspecified ways. Those ways certainly include targeted advertising (often phrased as "special offers"), but may also include selling your data (with or without anonymization) to other parties. And what happens to your data when one company buys another and uses data in ways that you didn't expect?

Data is frequently collected, used, and sold without consent. This includes organizations like Acxiom, Equifax, Experian, and Transunion, who collect data to assess financial risk, but many common brands also connect data without consent. In Europe, Google collected data from cameras mounted on cars to develop new mapping products. AT&T and Comcast both used cable set top boxes to collect data about their users, and Samsung collected voice recordings from TVs that respond to voice commands. There are many, many more examples of non-consensual data collection. At every step of building a data product, it is essential to ask whether appropriate and necessary consent has been provided.

Clarity

Clarity is closely related to consent. You can't really consent to anything unless you're told clearly what you're consenting to. Users must have clarity about what data they are providing, what is going to be done with the data, and any downstream consequences of how their data is used. All too often, explanations of what data is collected or being sold are buried in lengthy legal documents that are rarely read carefully, if at all. Observant readers of Eventbrite's user agreement recently discovered that listing an event gave the company the right to send a video team, and exclusive copyright to the recordings. And the only way to opt out was by writing to the company. The backlash was swift once people realized the potential impact, and Eventbrite removed the language.

Facebook users who played Cambridge Analytica's "This Is Your Digital Life" game may have understood that they were giving up their data; after all, they were answering questions, and those

answers certainly went somewhere. But did they understand how that data might be used? Or that they were giving access to their friends' data behind the scenes? That's buried deep in Facebook's privacy settings.

Even when it seems obvious that their data is in a public forum, users frequently don't understand how that data could be used. Most Twitter users know that their public tweets are, in fact, public; but many don't understand that their tweets can be collected and used for research, or even that they are for sale. This isn't to say that such usage is unethical; but as Casey Fiesler points out, the need isn't just to get consent, but to inform users what they're consenting to. That's clarity.

It really doesn't matter which service you use; you rarely get a simple explanation of what the service is doing with your data, and what consequences their actions might have. Unfortunately, the process of consent is often used to obfuscate the details and implications of what users may be agreeing to. And once data has escaped, there is no recourse. You can't take it back. Even if an organization is willing to delete the data, it's very difficult to prove that it has been deleted.

There are some notable exceptions: people like John Wilbanks are working to develop models that help users to understand the implications of their choices. Wilbanks' work helps people understand what happens when they provide sensitive medical and health data to a service.

Consistency and trust

Trust requires consistency over time. You can't trust someone who is unpredictable. They may have the best intentions, but they may not honor those intentions when you need them to. Or they may interpret their intentions in a strange and unpredictable way. And once broken, rebuilding trust may take a long time. Restoring trust requires a prolonged period of consistent behavior.

Consistency, and therefore trust, can be broken either explicitly or implicitly. An organization that exposes user data can do so intentionally or unintentionally. In the past years, we've seen many security incidents in which customer data was stolen: Yahoo!, Target, Anthem, local hospitals, government data, and data brokers like Experian, the list grows longer each day. Failing to safeguard customer data breaks trust—and safeguarding data means nothing if not consistency over time.

We've also seen frustration, anger, and surprise when users don't realize what they've agreed to. When Cambridge Analytica used Facebook's data to target vulnerable customers with highly specific advertisements, Facebook initially claimed that this was not a data breach. And while Facebook was technically correct, in that data was not stolen by an intruder, the public's perception was clearly different. This was a breach of trust, if not a breach of Facebook's perimeter. Facebook didn't consistently enforce its agreement with its customers. When the news broke, Facebook became unpredictable because most of its users had no idea what it would or wouldn't do. They didn't understand their user agreements, they didn't understand their complex privacy settings, and they didn't understand how Facebook would interpret those settings.

Control and transparency

Once you have given your data to a service, you must be able to understand what is happening to your data. Can you control how the service uses your data? For example, Facebook asks for (but doesn't require) your political views, religious views, and gender preference. What happens if you change your mind about the data you've provided? If you decide you're rather keep your political affiliation quiet, do you know whether Facebook actually deletes that information? Do you know whether Facebook continues to use that information in ad placement?

All too often, users have no effective control over how their data is used. They are given all-or-nothing choices, or a convoluted set of options that make controlling access overwhelming and confusing. It's often impossible to reduce the amount of data collected, or to have data deleted later.

A major part of the shift in data privacy rights is moving to give users greater control of their data. For example, Europe's General Data Protection Regulation (GDPR) requires a user's data to be provided to them at their request and removed from the system if they so desire.

Consequences

Data products are designed to add value for a particular user or system. As these products increase in sophistication, and have broader societal implications, it is essential to ask whether the data that is being collected could cause harm to an individual or a group. We continue to hear about unforeseen consequences and the "unknown unknowns" about using data and combining data sets. Risks can

never be eliminated completely. However, many unforeseen consequences and unknown unknowns could be foreseen and known, if only people had tried. All too often, unknown unknowns are unknown because we don't want to know.

Due to potential issues around the use of data, laws and policies have been put in place to protect specific groups: for example, the Children's Online Privacy Protection Act (COPPA) protects children and their data. Likewise, there are laws to protect specific sensitive data sets: for example, the Genetic Information Nondiscrimination Act (GINA) was established in 2008 in response to rising fears that genetic testing could be used against a person or their families. Unfortunately, policy doesn't keep up with technology advances; neither of these laws have been updated. Given how rapidly technology is being adopted by society, the Obama Administration realized that the pace of the regulatory process couldn't keep up. As a result, it created the roles of the U.S. chief technology officer and chief data scientist. The Obama administration also established more than 40 chief data officers and scientists across the federal government. The result has been to make sure the regulatory process fosters innovation while ensuring the question of potential of harm is asked regularly and often.

Even philanthropic approaches can have unintended and harmful consequences. When, in 2006, AOL released anonymized search data to researchers, it proved possible to "de-anonymize" the data and identify specific users. In 2018, Strava opened up their data to allow users to discover new places to run or bike. Strava didn't realize that members of the U.S. military were using GPS-enabled wearables, and their activity exposed the locations of bases and patrol routes in Iraq and Afghanistan. Exposure became apparent after the product was released to the public, and people exploring the data started talking about their concerns.

While Strava and AOL triggered a chain of unforeseen consequences by releasing their data, it's important to understand that their data had the potential to be dangerous even if it wasn't released publicly. Collecting data that may seem innocuous and combining it with other data sets has real-world implications. Combining data sets frequently gives results that are much more powerful and dangerous than anything you might get from either data set on its own. For example, data about running routes could be combined with data from smart locks, telling thieves when a house or apartment was unoccupied, and for how long. The data could be stolen by an attacker, and the company wouldn't even recognize the damage.

It's easy to argue that Strava shouldn't have produced this product, or that AOL shouldn't have released their search data, but that ignores the data's potential for good. In both cases, well-intentioned data scientists were looking to help others. The problem is that they didn't think through the consequences and the potential risks.

It is possible to provide data for research without unintended side-effects. For example, the U.S. Internal Revenue Service (IRS), in collaboration with researchers, opened a similar data set in a tightly controlled manner to help understand economic inequality. There were no negative repercussions or major policy implications. Similarly the Department of Transportation releases data about traffic fatalities. The U.K. Biobank (one of the largest collections of genomic data) has a sophisticated approach to opening up different levels of data. Other companies have successfully opened up data for the public benefit, including LinkedIn's Economic Graph project and Google Books' ngram viewer.

Many data sets that could provide tremendous benefits remain locked up on servers. Medical data that is fragmented across multiple institutions limits the pace of research. And the data held on traffic from ride-sharing and gps/mapping companies could transform approaches for traffic safety and congestion. But opening up that data to researchers requires careful planning.

Implementing the 5 Cs

Data can improve our lives in many ways, from the mundane to the amazing. Good movie recommendations aren't a bad thing; if we could consolidate medical data from patients around the world, we could make some significant progress on treating diseases like cancer. But we won't get either better movie recommendations or better cancer treatments if we can't ensure that the five Cs are implemented effectively. We won't get either if we can't treat others' data as carefully as we'd treat our own.

Over the past decade, the software industry has put significant effort into improving user experience (UX). Much of this investment has been in user-centric approaches to building products and services that depend on the data the collective user base provides. All this work has produced results: using software is, on the whole, easier and more enjoyable. Unfortunately, these teams have either intentionally or unintentionally limited their efforts to providing users with immediate gratification

or the ability to accomplish near-term goals. “Growth hacking” focuses on getting people to sign up for services through viral mechanisms. We’ve seen few product teams that try to develop a user experience that balances immediate experience with long-term values.

In short, product teams haven’t considered the impacts of the five Cs. For example, how should an application inform users about how their data will be used, and get their consent? That part of user experience can’t be swept under the rug. And it can’t mean making it easy for users to give consent, and difficult to say “no.” It’s all part of the total user experience. Users need to understand what they are consenting to and what effects that consent might have; if they don’t, the designer’s job isn’t done.

Responsibility for the 5 Cs can’t be limited to the designers. It’s the responsibility of the entire team. The data scientists need to approach the problem asking “what if” scenarios that get to all of the five C’s. The same is true for the product managers, business leaders, sales, marketing, and also executives.

The five C’s need to be part of every organization’s culture. Product and design reviews should go over the five Cs regularly. They should consider developing a checklist before releasing a project to the public. All too often, we think of data products as minimal viable products (MVPs: prototypes to test whether the product has value to users). While that’s a constructive approach for developing and testing new ideas, even MVPs must address the five Cs. The same is true for well-established products. New techniques may have been developed that could result in harm in unforeseen ways. In short, it’s about taking responsibility for the products that are built. The five C’s are a mechanism to foster dialogue to ensure the products “do no harm.”

Diversity:

Diversity Stats on Data Science

I will layout the foundations of our discussion here based on a study made by the General Assembly (GA) three years ago. GA is an education company that is specialized in training *students* in Data

Science and other technical fields. They put together their student’s data and what we will see here are some of the outputs of that. You can check their report here.

It is important to understand the characteristics of who is studying Data Science now. This might help us to *predict* the future because it can be an indicator of how the field is evolving — concerning diversity or not.

When comes down to **ethnicity**, we see an unbalancing. The numbers for Blacks and Latinos are very tiny. Looking at the **gender** distribution, we see that we have way more males than females studying Data Science, according to GA.

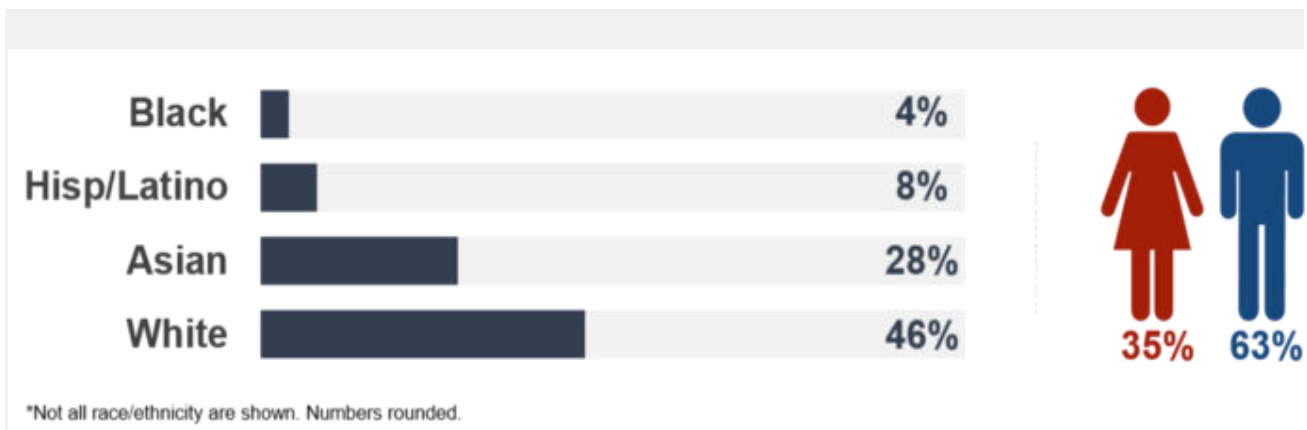


Fig. 5.7. Race/Ethnicity and Gender Distribution of Data Science Classes. (Source: General Assembly)

A further investigation shows that, at that point in time, the percentage of Blacks and Latinos taking their Data Science course was very low indeed. Only 12% of students enrolled in the program were Black and Latino.

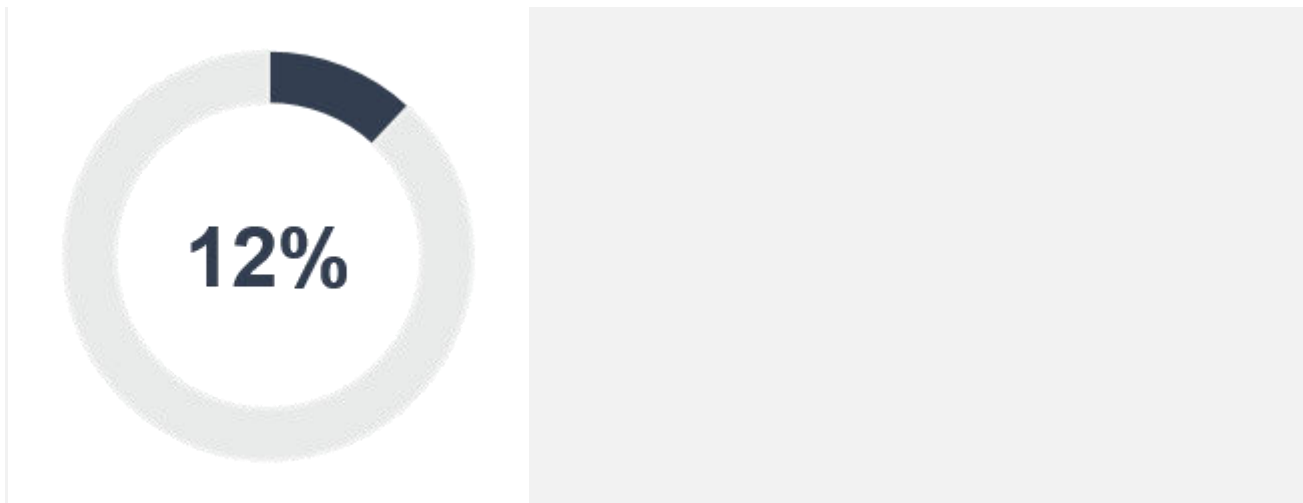


Fig. 5.8. Percent of Black and Hispanic/Latino taking Data Science Class at GA. (Source: General Assembly)

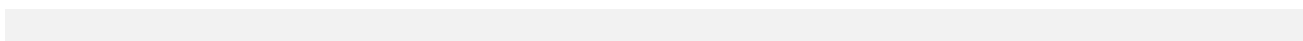
To give some perspective on that, we can take a look at the breakdown of race/ethnicity from the Census Bureau as below:

Race and Hispanic Origin	
White alone, percent	76.3%
Black or African American alone, percent (a)	13.4%
American Indian and Alaska Native alone, percent (a)	1.3%
Asian alone, percent (a)	5.9%
Native Hawaiian and Other Pacific Islander alone, percent (a)	0.2%
Two or More Races, percent	2.8%
Hispanic or Latino, percent (b)	18.5%
White alone, not Hispanic or Latino, percent	60.1%

Table 1. Population Estimates by Race, July 1, 2019. (Source: United States Census Bureau)

We can say that the population of Black and Latino sums up to around 32% of the total population in America. However, around 12% of them were enrolled in the Data Science program.

The educational background of people studying Data Science was also measured as follows:



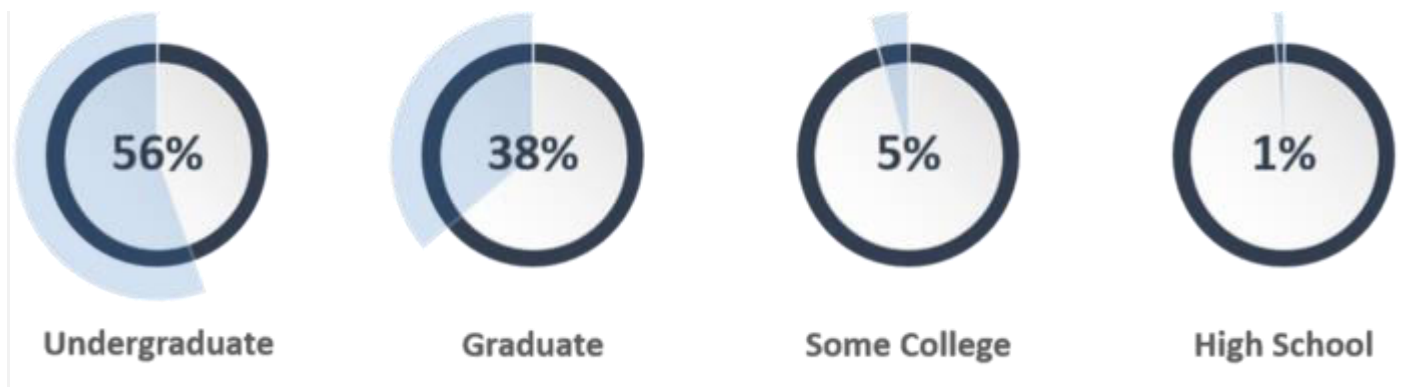
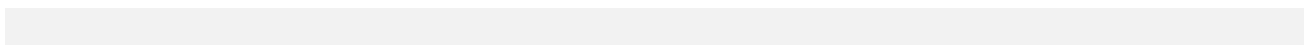


Fig. 5.9 . Educational Background of Students Taking Data Science Class at GA. (Source: General Assembly)

As we could see, Data Science students come from a highly educated background. “Graduate” means students with Master's or Doctoral degrees. General Assembly concludes that “Data Science seems to draw from a smaller, more specialized pool, which could, in part, perpetuate diversity issues”.

In fact, a study from Education Week shows that “Black, Latino Students Lack Access to High-Level Science and Math”. Certainly, this reduces the access and participation of these minorities in *STEM* (Science, Technology, Engineering, and Math) programs and careers.

Research from NCSES (National Center for Science and Engineering Statistics) shows that these minorities, including Native Americans, really have a low share in the S&E (Science and Engineering) occupations. See below:



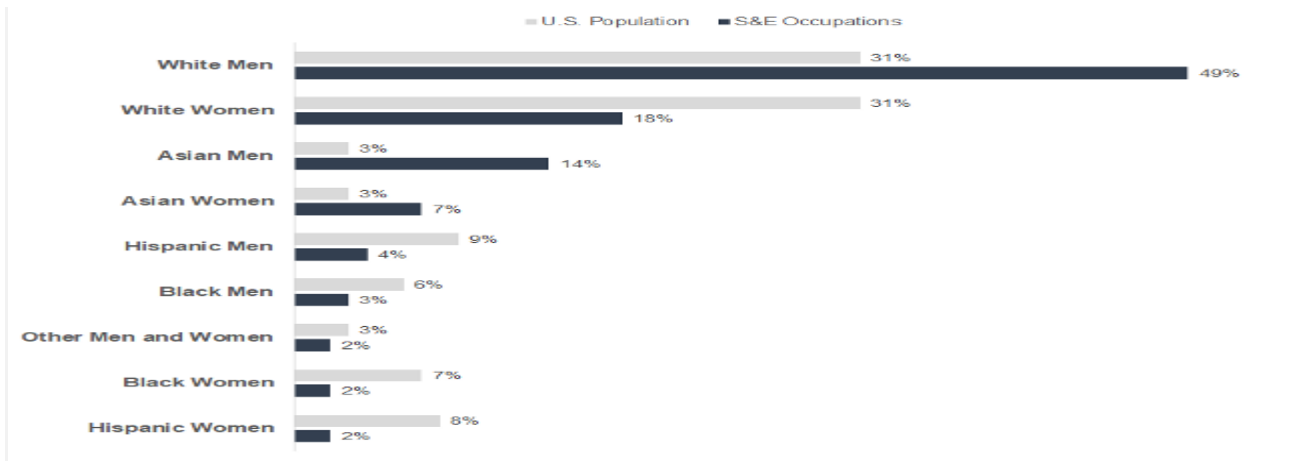


Fig. 5.10 S&E Occupations by Race/Ethnicity and Gender. (Source: NCSES)

We see a low contribution of the mentioned minorities in the S&E field. Moreover, we can further investigate women's participation — in 2014 — according to some degree fields as following:

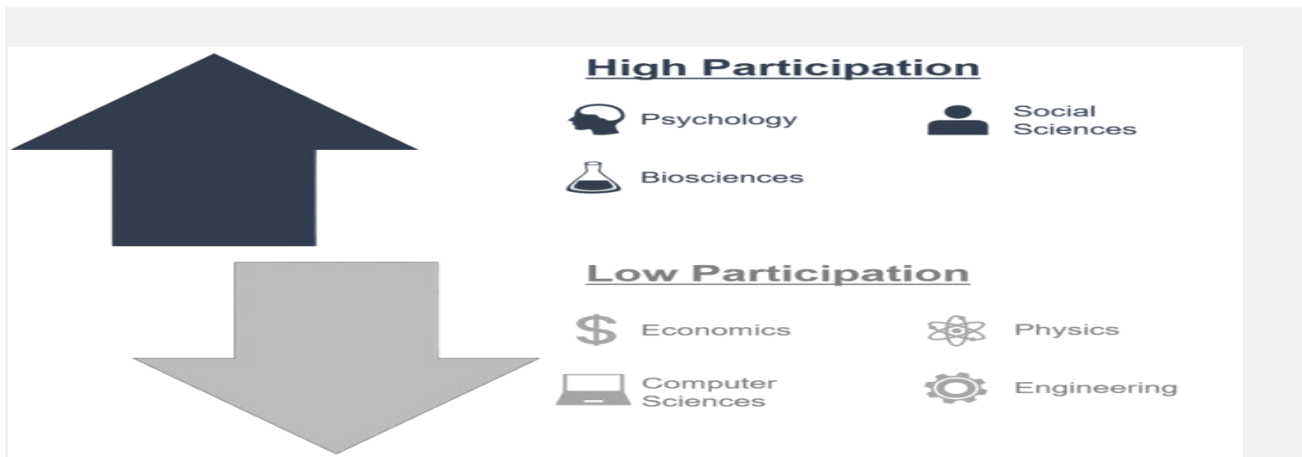


Fig. 5.11 Field of Degrees for Women. (Source: NCSES)

So, if you happen to be a woman and on top of that Black, Hispanic/Latino, or Native American, good luck to you. In fact, according to the National Science Board, only 28% of women with college degrees work on S&E. STEM careers and programs is where we find the highest gaps in race/ethnicity and gender. On top of that, these minorities tend to face a lot of biases while in college. This alone can make things worse and prevent minorities to thrive at the college level.

The lack of diversity in Data Science appears to be related to the lack of access to these minorities to STEM. While this might be a naive statement, due to its simplicity, it might be an assumption that we could take. So, nurturing access can be beneficial in the long run. I will explore more about this soon.

Who's Fault?

Instead of focusing on finding a scapegoat for this complex problem, we should drive our energy on what we can do today. Structural inequality like this one is an issue that has been rooted in society for centuries. Working on solutions makes more sense than trying to find who to blame.

This issue affects the Tech industry. It hits it hard. Some businesses say that they are investing X amount of money on diversity and inclusion, but how does that translate into shaping the reality?

Moving next in our analysis, we will see some of the *diversity reports* released by FAANG (Facebook, Apple, Amazon, Netflix, and Google) companies. As you can imagine, the inequality issue we saw previously, extent throughout these companies. Meaning, the lack of access and participation of minorities in Tech is strong.



Fig. 5.12 11 Estimated Number of Data Scientists by Company. (Source: Diffbot)

The biggest question one may ask is how many of these Data Scientists we see above belong to a minority group. Well, we will probably never know for sure, but Diffbot provided a breakdown of these numbers by gender. According to them, 23% are women and 77% are men.

This share looks familiar, huh? It's basically what we saw before on FAANG companies. Surprised? You shouldn't!

If you look again at "*Image 1*" from the General Assembly, you'll see that the percentage of women taking their Data Science course was higher than what we see in the market. Why is that?

To get an estimation about race/ethnicity we can use Facebook as an example. Facebook has 723 Data Scientists (according to our source, and at that point in time). If we sum up the percents for *Black*, *Hispanic*, and what they call *Additional Groups*, we get a total of 6.2% (this count is for 2020 alone) of participation of these minorities in ***Technical*** roles.

That is to say that we would have around 45 Data Scientists belonging to these minority groups if we are lucky. Again, this number might be off but based on the ***pattern*** that we are seeing across the board, it might make sense.

Kaggle's "State of Data Science and Machine Learning 2020" Report

Kaggle released their most recent report on Data Science and Machine Learning demographics. Around 82% of the people who participated in this Kaggle's survey were men. About 68% have a Master's or Doctoral degree. Although the report does not show any data regarding race/ethnicity, it has a lot of other interesting findings. I'd strongly recommend you to check it out [here](#).

As we could see, once again, the patterns that we are spotting throughout our analysis appears to be validated from different sources.

Issues Caused by Lack of Diversity in Data Science

The lack of diversity in Data Science has an impact on how algorithms are built. That also contributes to the average people distrust algorithms. This article from Vox talks more about this issue.

One of the most compelling pieces of evidence of this problem is the issue of *racial bias*. Now and then we hear some news on how, particularly, Black people are target on that. Women can also be forgotten and overlooked when designing algorithms.

We are all influenced by some *cognitive biases*. This is something we need to fight against daily. It's a tough fight. But it can get worse. How? Well, when you have a room full of people that look the same, think the same way, and came from the same schools. It's hard to create an environment that considers the different points of view. That could also limit the scope of creative solutions — innovation and inclusion should go hand in hand.

Future Trends:

The presence of data in every field that you can think of is what turns out to be a reason why organizations are showing interest in data science. Also, the fact that data will continue to be an integral part of our lives till eternity serves to be yet another driver of data science. That said, it's really important to stay updated with the hottest data science trends that could serve to be a blessing to grow your business. Here are the top 10 data science trends for this decade.

Predictive analysis

For a business to prosper, it is critical to know what the future might look like. This is exactly where predictive analysis comes into play. Organizations rely on their customers to a large extent. Hence, being able to understand their behaviors helps in making better decisions ahead. This technique is one of the smartest to come up with the best strategies to target the customers that'd aid in retaining the older ones and also get newer customers.

Machine learning

Over the years, we have seen how much automation has transformed the world. This is why machine learning has gained importance like never before. The coming years will see more automation and hence the rise in the number of organizations adopting machine learning will surpass one's imagination for sure.

IoT

Gone are the days when IoT was considered to be something that would have limited applications. Today, we are living in a world where our smartphones have the ability to control appliances like TV, AC, etc. All of this is possible because of IoT. Google Assistant is yet another remarkable innovation in the area of IoT. Thus, companies looking for ways to invest in this technology come as no big surprise. This simply throws light on how rapidly the IoT industry would grow in the days ahead.

Blockchain

Needless to say, cryptocurrencies like Bitcoin, Litecoin, etc. have become the talk of the world. All of these currencies employ blockchain technology. With the world showing keen interest in this field, it surely stands a far-reaching implementation in the coming time

Edge computing

Edge computing is known for faster processing of information and it also boasts of reducing latency, cost and traffic. It is solely because of these features that the organizations are not willing to sideline this option. With this computing in place, dealing with real-time applications couldn't have got any better. The coming years could see more of a considerable shift from traditional methods to that of edge computing.

DataOps

Lets' face the reality – the data pipeline has become more complex and thus requires even more integration and governance tools. DataOps to our rescue it is! Tasks right from collection to preparation to analysis, testing automation, implementing automated testing, delivery for providing enhanced data quality and analysis are all covered. This trend will continue for the years to come.

Artificial Intelligence

Be it a small enterprise or a tech giant, all of them have relied on AI in one way or the other. All those complex tasks are no longer a concern for we now can rely on AI for the same. Also, the

reduction in errors is yet another strong reason to why AI stands apart. Now that we've relied on AI so much, there's no coming back!

Data visualization

This is one of those prominent trends that we can trust with. This is because the organizations are moving their conventional data warehouses to the cloud.

Better user experience

The extent to which user experience is given importance to talks volume about the success of the company. This is why companies are leaving no stone unturned in providing the best possible user experience – be it in the form of chatbots, personal assistance, or AI-driven tools for that matter.

Data governance

This is yet another area that's gaining a lot of importance. Numerous companies out there are still struggling to comply with the rules and regulations. It is critical to not just comply with these but also to understand the impact of the same on the present and future operations. Data scientists who have sound knowledge about all of this is the need of the hour.

These trends show a clearer picture of what data science strategies need to be implemented to retain your customers and also take your business to new heights.

References:

1. www.analyticsvidhya.com
2. <https://towardsdatascience.com/>
3. <https://www.analyticsinsight.net>
4. www.oreilly.com

