

Subject

Data Analyst

Vol.01

 **Empowering Youth!**



Skill & Assessment Partner

NASSCOM®

stl.tech | stlacad.tech

Data Analyst

Submitted to :- Bihar Skill Development Mission, Labour Resources Department, GoB	Submitted By :- Sterlite Technologies Ltd
	Session : 2022-23

Course name:

- Course Id-
- Candidate Eligibility : Diploma/ Graduate
- Course Duration: (In hours) 650

CONTACT DETAILS OF THE BODY SUBMITTING THE QUALIFICATION FILE

Name and address of submitting body:

Sterlite Technologies Ltd

Name and contact details of individual dealing with the submission

Name : Mrs./Mr. Srikant Pattnaik

Position in the organization : Manager

Tel number (s) (Mobile no.) : 9702048264

Website : www.stlacad.tech

E-mail address : srikant.pattnaik@stl.tech

DATA ANALYST
STUDENT GUIDE



About the Student Guide

The student guide contains modules which will help you to acquire relevant knowledge and skills (generic and domain-specific skills) related to the 'Data Analyst' job role. Knowledge in each module is easily understood and grasped by you before you move on to the next module. Comprehensible diagrams & images from world of work have been included to bring about visual appeal and to make the text lively and interactive for you. You can also try to create your own illustrations using your imagination or taking the help of your trainer.

Let us now see what the sections in the modules have for you.

Section 1: Learning Outcome

This section introduces you to the learning objectives and knowledge criteria covered in the module. It also tells you what you will learn through the various topics covered in the module.

Section 2: Relevant Knowledge

This section provides you with the knowledge to achieve relevant skill and proficiency to perform tasks of the Data Analyst. The knowledge developed through the module will enable you to perform certain activities related to the job market. You should read through the textual information to develop an understanding on the various aspects of the module before you complete the exercise(s).

Section 3: Exercises

Each module has exercises, which you should practice on completion of the learning sessions of the module. You will perform the activities in the classroom, at home or at the workplace. The activities included in this section will help you to develop necessary knowledge, skills and attitude that you need for becoming competent in performing the tasks at workplace. The activities should be done under the supervision of your trainer who will guide you in completing the tasks and also provide feedback to you for improving your performance.

Section 4: Assessment Questionnaire

The review questions included in this section will help you to check your progress. You must be able to answer all the questions before you proceed to the next module.

CONTENTS

Module 1	Introduction to Data Analytics	1
1.1	Introduction to Data Analytics	1
1.2	What are the Tools Used in Data Analytics?	4
1.3	Dealing with Different Types of Data	6
1.4	Data Visualization for Decision Making	8
1.5	Overview of Data Science and Machine Learning	9
1.6	Application of Data Analytics in different sectors	11
1.7	Data Mining and Data Profiling	13
1.8	Data Wrangling	19
1.9	Sampling Techniques in Data Analytics	20
1.10	Analytics Framework in and Latest Trends	22
Exercises		24
Assessment Questionnaire		25
Module 2	Business Analytics with Excel	27
2.1	Introduction to Business Analytics	27
2.2	Conditional Formatting and Important Functions	30
2.3	Analysing Data with Pivot Tables	50
2.4	Dashboarding	64
2.5	Data Analysis Using Statistics	71
Exercises		75
Assessment Questionnaire		78
Module 3	Programming Basics and Data Analytics with Python	79
3.1	Introduction to Python	79
3.2	Python Environment Setup and Essentials	86
3.3	Python Programming Fundamentals	97
3.4	Data Analytics Overview	140
3.5	Statistical Computing	143
3.6	Mathematical Computing using NumPy	147
3.7	Data Manipulation with Pandas	152
3.8	Data Visualization with Python	163
3.9	Introduction to Model Building	187
Exercises		195
Assessment Questionnaire		196

Module 4	Tableau Training	197
4.1	Introduction to Tableau	197
4.2	Tools of Tableau	199
4.3	Tableau Architecture	201
4.4	Download and Installation of Tableau	204
4.5	Using the Tableau Workspace Control Effectively	208
4.6	Data Aggregation in Tableau	218
4.7	Tableau File Types	224
4.8	Data Connection with Data Sources	225
4.9	Tableau Editing Metadata	234
4.10	Tableau Operators and Calculation	249
4.11	Functions and Calculations in Tableau	253
4.12	Tableau LOD Expressions	266
4.13	Tableau Filter Operations	269
4.14	Tableau Sort Data	283
4.15	Tableau Groups, Hierarchy and Sets	286
4.16	Tableau Charts	293
4.17	Creating a Dashboard in Tableau	299
	Exercises	311
	Assessment Questionnaire	311
Module 5	SQL Training	312
5.1	Introduction to SQL	312
5.2	SQL Commands	315
5.3	SQL Syntax and Statements	316
5.4	SQL Data Types	323
5.5	SQL Operators	328
5.6	SQL Create and Select Database	349
5.7	SQL TABLE Variable and Statements	352
5.8	SQL Joins and Injection	357
5.9	Selection Commands, Alias and Subqueries	360
5.10	MySQL User Access Control Functions, Backup & Recovery	364
	Exercises	375
	Assessment Questionnaire	375

Module 6	Power BI Training	377
6.1	Introduction to Power BI	377
6.2	Components of Power BI	380
6.3	Architecture of Power BI	381
6.4	Power BI Tools	382
6.5	Power BI Advantages and Disadvantages	383
6.6	Download and Install Power BI Desktop	383
6.7	Power BI Dashboard	386
6.8	Power BI Reports	388
6.9	Difference between Dashboards and Reports	389
6.10	Power BI Data Sources	390
6.11	Power BI Embedded	394
6.12	Power BI Gateway	395
6.13	Building Blocks of Power BI	397
6.14	Power BI Report Server	398
6.15	Power BI DAX	398
6.16	Who Uses Power BI?	402
	Exercises	402
	Assessment Questionnaire	404

MODULE 1

INTRODUCTION TO DATA ANALYTICS

Section 1: Learning Outcomes

After completing this module, you will be able to:

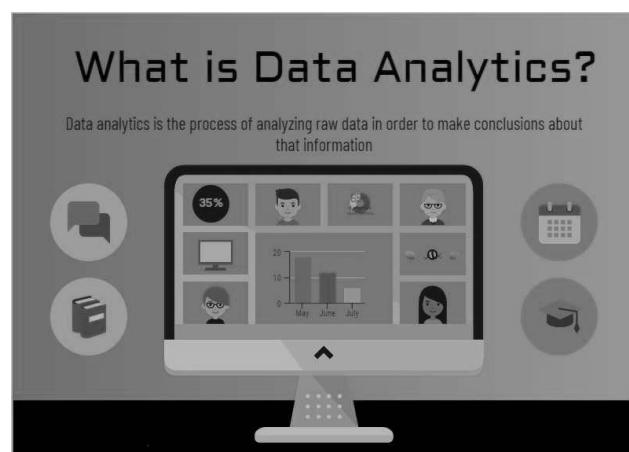
- Introduce Data Analytics
- Tell the functions of various tools used in Data Analytics
- Deal with Different Types of Data
- Describe the overview of Data Science and Machine Learning
- Explain application of Data Analytics in different sectors
- Tell the process of Data Mining and Data Profiling and Data Wrangling
- Differentiate between various Sampling Techniques
- Describe Analytics Framework in and Latest Trends

Section 2: Relevant Knowledge

1.1 Introduction to Data Analytics

What is Data Analytics?

- Data Analytics refers to the techniques used to analyze data to enhance productivity and business gain.
- A Data Analyst is a professional who can analyze data by applying various tools and techniques and gathering the required insights.
- The techniques and the tools used vary according to the organization or individual.
- In brief, if you understand your business administration and have the capability to perform exploratory Data Analysis, to gather the required information, then you are good to go with a career in Data Analytics.



Why Do We Need Data Analytics?

For ages, Data has always been the buzzword. As there is a humongous amount of data available in the market but it is of no use because it does not benefit the businesses. Whether the data is generated from an individual or generated from large-scale enterprises, in each aspect data needs to be analyzed to benefit yourself or businesses from it.

Then the next prime question that arises in our mind is how do analyze data for our greater good? Well, that is where the term ‘Data Analytics’ comes into the picture. In this course, you will get thorough insights into ‘What is Data Analytics?’, “How can You Be a Data Analyst?”, “How Much You Can Earn as Data Analyst?” and much more.

- Helps businesses monitor, manage, and collect performance measures to improve decision-making across the organization.
- Improves business operations.
- Improves consumer engagement, corporate performance, and boost revenue.
- Helps make decisions based on verifiable, data-driven proofs.

Data Analytics benefits the enterprises to:

- **Gather Hidden Insights:** Get all the hidden insights from the data, that are gathered and then analyzed according to the business requirements.
- **Generate Reports:** Generated Reports from the data are passed on to the respective teams and individuals to deal with further actions for a high rise in business & have a competitive edge.
- **Perform Market Analysis:** To understand the market sentiments, strengths and weaknesses of competitors etc. Market Analysis must need to be performed.

S	W	O	T
STRENGTHS	WEAKNESSES	OPPORTUNITIES	THREATS
<ul style="list-style-type: none"> • Things your company does well • Qualities that separate you from your competitors • Internal resources such as skilled, knowledgeable staff • Tangible assets such as intellectual property, capital, proprietary technologies etc. 	<ul style="list-style-type: none"> • Things your company lacks • Things your competitors do better than you • Resource limitations • Unclear unique selling proposition 	<ul style="list-style-type: none"> • Underserved markets for specific products • Few competitors in your area • Emerging need for your products or services • Press/media coverage of your company 	<ul style="list-style-type: none"> • Emerging competitors • Changing regulatory environment • Negative press/media coverage • Changing customer attitudes toward your company

- **Improve Business Requirement:** Data analysis allows for improving Business to the consumer’s expectations, requirements and experience.

Now that you know the need for Data Analytics, let us have a quick look at what is Data Analytics.

Who is a Data Analyst?

- A data analyst is a person who can do basic descriptive statistics, visualize data, and communicate data points for conclusions.
- They must have a basic understanding of statistics, a perfect sense of databases, the ability to create new views, and the perception to visualize the data. Data analytics can be referred to as the necessary level of data science.
- Data analysts translate numbers into plain English.
- A Data Analyst delivers value to their companies by taking information about specific topics and then interpreting, analyzing, and presenting findings in comprehensive reports. So, if you have the capabilities like collecting data from various sources, analyzing the data, gathering hidden insights, and generating reports, then you can become a Data Analyst.
- A Data Analyst should also possess skills such as Statistics, Data Cleaning, Exploratory Data Analysis, and Data Visualization. Also, if you know about Machine Learning, then that would make you stand out from the crowd.
- On average, a Data Analyst can expect a salary of ₹404,660 (IND) or \$83,878 (US). As experts, data analysts are often called on to use their skills and tools to provide competitive analysis and identify trends within industries.

Essential Skills to Become a Data Analyst

A data analyst should be able to take a specific question or topic, discuss what the data looks like, and represent that data to relevant stakeholders in the company. If you're looking to step into the role of a data analyst, you must gain these four key skills:

- Knowledge of mathematical statistics
- Fluent understanding of R and Python
- Data wrangling
- Understand PIG/ HIVE
- Should possess skills such as Statistics, Data Cleaning, Exploratory Data Analysis, and Data Visualization.
- Also know about Machine Learning to stand out from the crowd.
- Should understand Structured Query Language (SQL).
- Statistical visualization & critical thinking.
- Extensive knowledge in Microsoft Excel.



How Does a Data Analyst Work?

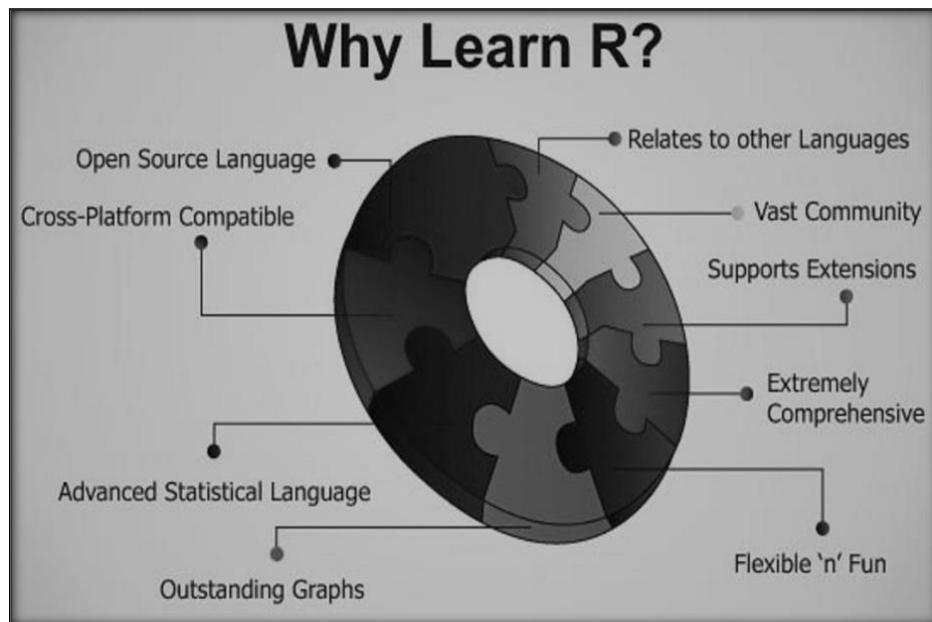
- Translate numbers into plain English.
- Collects, cleans, and interprets data sets in order to answer a question or solve a problem.
- Presents findings in comprehensive reports.

1.2 What are the Tools Used in Data Analytics?

With the increasing demand for Data Analytics in the market, many tools have emerged with various functionalities for this purpose. Either open-source or user-friendly, the top tools in the data analytics market are as follows.

- **R programming:**

- Commonly used in statistical computing, data analytics and scientific research.
- One of the most popular languages used by statisticians, data analysts, researchers and marketers to retrieve, clean, analyze, visualize and present data.

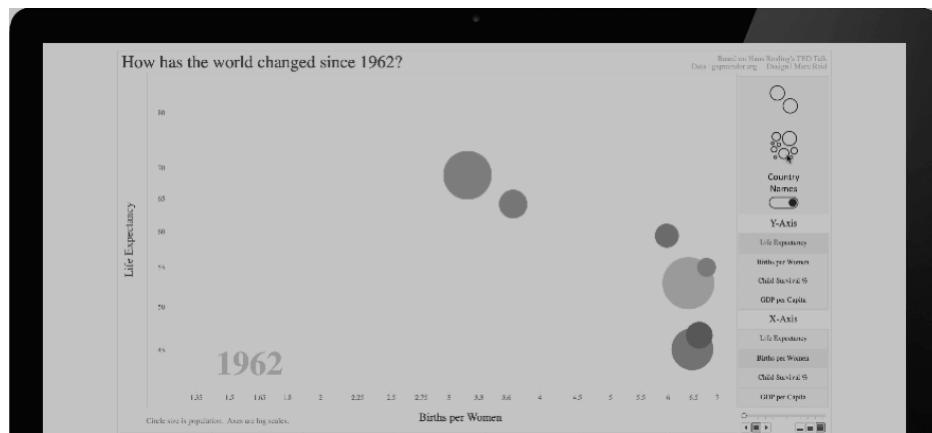


- **Python:**

- Python is an open-source, object-oriented programming language that is easy to read, write, and maintain.
- It provides various machine learning and visualization libraries such as Scikit-learn, TensorFlow, Matplotlib, Pandas, Keras, etc.

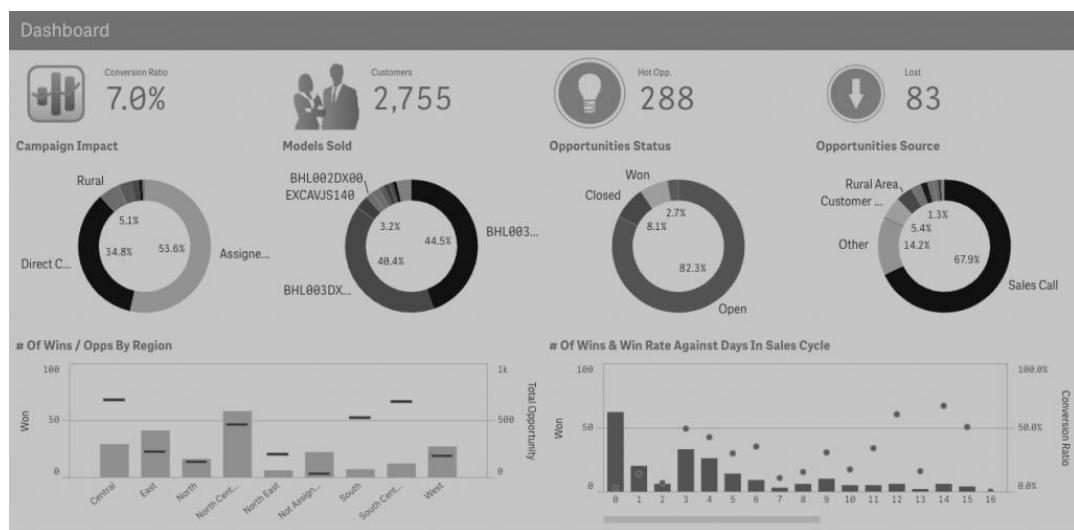


- **Tableau Public:** This is free software that connects to any data source such as Excel, corporate Data Warehouse, etc. It then creates visualizations, maps, dashboards etc with real-time updates on the web.



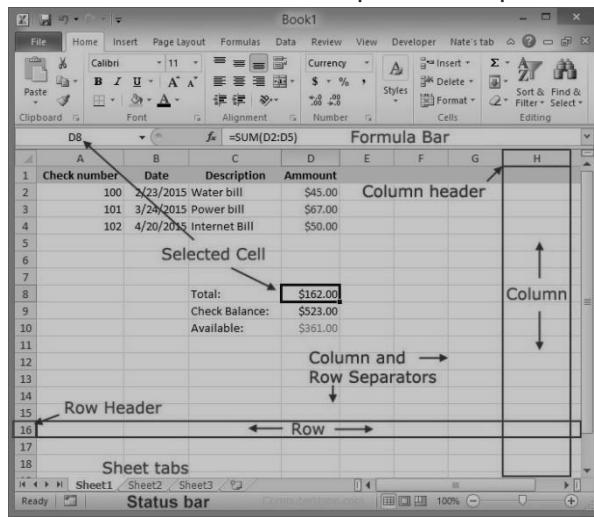
- **QlikView:**

- This tool offers in-memory data processing with the results delivered to the end-users quickly.
- A BI data discovery product.
- Rapidly develop and deliver interactive guided analytics applications and dashboards.
- Ask and answer your own questions and follow your own paths to insight.
- Offers in-memory data processing with the results delivered to the end-users quickly.
- Also offers data association and data visualization with data being compressed to almost 10% of its original size.



- **SAS:** A programming language and environment for data manipulation and analytics.

- **Microsoft Excel:** This tool is one of the most widely used tools for data analytics. This tool analyzes the tasks that summarize the data with a preview of pivot tables.



The screenshot shows a Microsoft Excel window titled "Book1". The formula bar at the top displays the formula $=SUM(D2:D5)$. The main area contains a table with columns for Check number, Date, Description, and Amount. Row 8 contains summary data: Total \$162.00, Check Balance \$523.00, and Available \$361.00. The table has a column header "Column header" and a row header "Row Header". A selected cell is highlighted with a red border. Labels with arrows point to several parts of the interface: "Selected Cell" points to the highlighted cell; "Column" points to the vertical scroll bar on the right; "Row" points to the horizontal scroll bar at the bottom; "Column and Row Separators" points to the grid lines between columns and rows; "Sheet tabs" points to the tabs at the bottom; and "Status bar" points to the status bar at the bottom right. The status bar shows "ComputerHope.com", "100%", and other status indicators.

- **RapidMiner:** A powerful, integrated platform that can integrate with any data source type such as Access, Excel, Microsoft SQL, Oracle, etc. Mostly used for predictive analytics, such as data mining, text analytics, and machine learning.
- **KNIME:** Konstanz Information Miner (KNIME) is an open-source data analytics platform to analyze and model data for reporting and integration through its modular data pipeline concept.
- **OpenRefine:** GoogleRefine is a data cleaning software that will help you clean up messy and parsing data from websites for analysis.
- **Apache Spark:** One of the largest large-scale data processing engines used for data pipelines and machine learning model development.

1.3 Dealing with Different Types of Data

- The different types of data analytics for a company depend on its stage of development. Most companies are likely already using some sort of analytics, but it typically only affords insights to make reactive, not proactive, business decisions.
- More and more, businesses are adopting sophisticated data analytics solutions with machine learning capabilities to make better business decisions and help determine market trends and opportunities.
- Organizations that do not start to use data analytics with proactive, future-casting capabilities may find business performance lacking because they cannot uncover hidden patterns and gain other insights.
- Typically, there are four main types of data used in Data Analytics. These are:

1. Predictive Data Analytics

- Predictive analytics may be the most commonly used category of data analytics.
- Businesses use predictive analytics to identify trends, correlations, and causation.
- The category can be further broken down into predictive modeling and statistical modeling; however, it is important to know that the two go hand in hand.
- For example, an advertising campaign for t-shirts on Facebook could apply predictive analytics to determine how closely the conversion rate correlates with a target audience's geographic area, income bracket, and interests. From there, predictive modeling could be used to analyze the statistics for two (or more) target audiences and provide possible revenue values for each demographic.

2. Prescriptive Data Analytics

- Prescriptive analytics is where AI and big data combine to help predict outcomes and identify what actions to take. This category of analytics can be further broken down into optimization and random testing.
- Using advancements in ML, prescriptive analytics can help answer questions such as “What if we try this?” and “What is the best action?” You can test the correct variables and even suggest new variables that offer a higher chance of generating a positive outcome.

3. Diagnostic Data Analytics

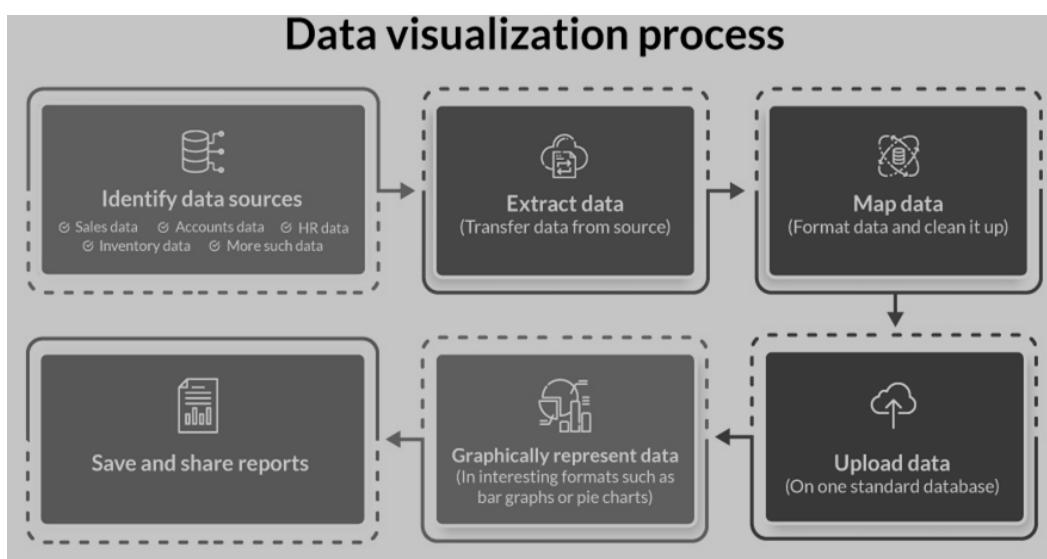
- While not as exciting as predicting the future, analyzing data from the past can serve an important purpose in guiding businesses.
- Diagnostic data analytics is the process of examining data to understand the cause and event or why something happened.
- Techniques such as drill-down, data discovery, data mining, and correlations are often employed.
- Diagnostic data analytics help answer why something occurred. Like the other categories, it too is broken down into two more specific categories:
 - Discover and Alerts
 - Query and Drill-downs
- Query and drill-downs are used to get more detail from a report. For example, a sales rep that closed significantly fewer deals one month. A drill-down could show fewer workdays, due to a two-week vacation.
- Discover and alerts notify of a potential issue before it occurs, for example, an alert about a lower amount of staff hours, which could result in a decrease in closed deals. You could also use diagnostic data analytics to “discover” information such as the most qualified candidate for a new position at your company.

4. Descriptive Data Analytics

- Descriptive analytics is the backbone of reporting— it is impossible to have business intelligence (BI) tools and dashboards without it.
- It addresses basic questions of “how many, when, where, and what.”
- Descriptive analytics can be further separated into two categories: ad hoc reporting and canned reports.
 - **Canned Reports:** A canned report has been designed previously and contains information around a given subject. An example of this is a monthly report sent by your ad agency or ad team that details performance metrics on your latest ad efforts.
 - **Ad Hoc Reports:** These are designed by you and usually are not scheduled. They are generated when there is a need to answer a specific business question. These reports are useful for obtaining more in-depth information about a specific query.
- It focuses on corporate social media profiles, examining the types of people who have liked your page and other industry pages, as well as other engagement and demographic information.
- Hyperspecificity helps give a more complete picture of your social media audience. Chances are you would not need to view this type of report a second time (unless there's a major change to your audience).

1.4 Data Visualization for Decision Making

- Data visualization is the graphical representation of information and data in a pictorial or graphical format (For example charts, graphs, and maps) that is vastly used in the space of data analytics.
- Data visualization tools provide an accessible way to see and understand trends, patterns in data and outliers.
- Data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.
- The concept of using pictures to understand data has been used for centuries. General types of data visualizations are Charts, Tables, Graphs, Maps, and Dashboards.
- Together Data visualization and analytics will conclude the datasets.
- In a few scenarios, it might act as a source for visualization. Plotly, DataHero, Tableau, Dygraphs, QlikView, ZingCHhart etc tools are used for creating data visualization.
- Data Analytics Visualization plays a crucial role in different sectors for big data processing, service management dashboards, analysis and design.

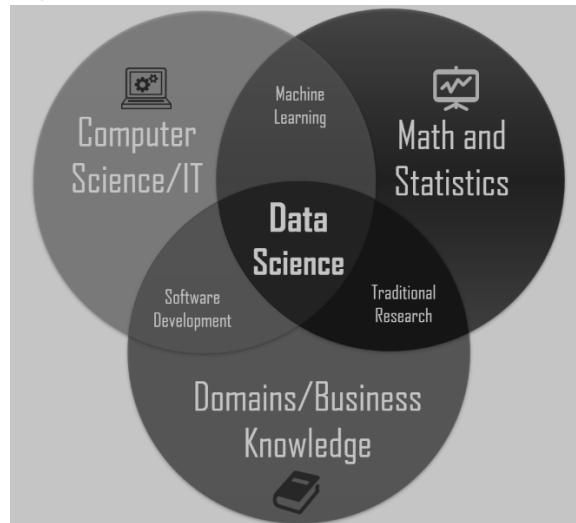


1.5 Overview of Data Science and Machine Learning

Data Science, Data Analytics, and Machine Learning are growing at an astronomical rate and companies are now looking for professionals who can sift through the goldmine of data and help them drive swift business decisions efficiently.

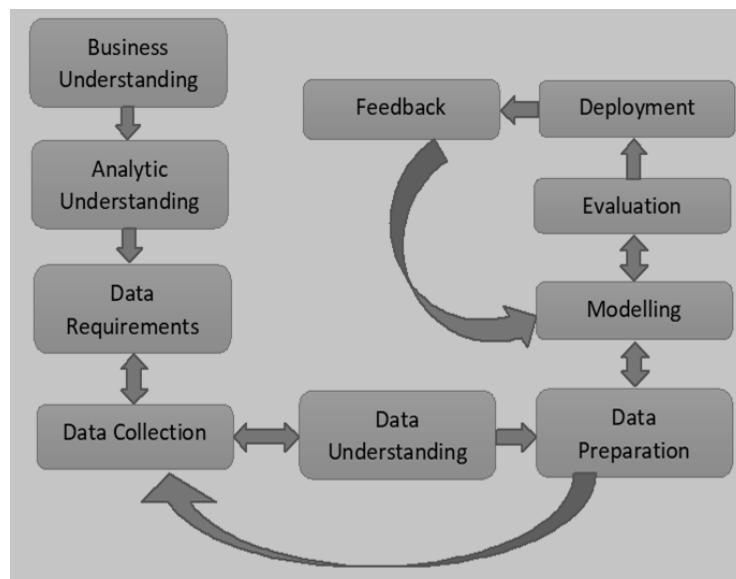
Data Science

- Data science is a concept used to tackle big data and includes data cleansing, preparation, and analysis.
- A data scientist gathers data from multiple sources and applies machine learning, predictive analytics, and sentiment analysis to extract critical information from the collected data sets.
- They understand data from a business point of view and can provide accurate predictions and insights that can be used to power critical business decisions.



Data Science Methodology

- The people who work in Data Science and are busy finding the answers to different questions every day come across the Data Science Methodology.
- Data Science Methodology indicates the routine for finding solutions to a specific problem. This is a cyclic process that undergoes critical behaviour guiding business analysts and data scientists to act accordingly.



Business Understanding:

Before solving any problem in the Business domain, it needs to be understood properly. Business understanding forms a concrete base, which further leads to easy resolution of queries. We should have the clarity of what is the exact problem we are going to solve.

Analytic Understanding:

Based on the above business understanding one should decide the analytical approach to follow. The approaches can be of 4 types: Descriptive approach (current status and information provided), Diagnostic approach (a.k.a statistical analysis, what is happening and why it is happening), Predictive approach (it forecasts the trends or future events probability) and Prescriptive approach (how the problem should be solved actually).

Data Requirements:

The above chosen analytical method indicates the necessary data content, formats and sources to be gathered. During the process of data requirements, one should find the answers to questions like 'what', 'where', 'when', 'why', 'how' & 'who'.

Data Collection:

Data collected can be obtained in any random format. So, according to the approach chosen and the output to be obtained, the data collected should be validated. Thus, if required one can gather more data or discard the irrelevant data.

Data Understanding:

Data understanding answers the question "Is the data collected representative of the problem to be solved?". Descriptive statistics calculates the measures applied over data to access the content and quality of matter. This step may lead to reverting to the previous step for correction.

Data Preparation:

Let us understand this by connecting this concept with two analogies. One is to wash freshly picked vegetables and the second is only taking the wanted items to eat on the plate during the buffet. Washing of vegetables indicates the removal of dirt i.e. unwanted materials from the data. Here noise removal is done. Taking only eatable items on the plate is, if we don't need specific data then we should not consider it for further process. This whole process includes transformation, normalization etc.

Modelling:

Modelling decides whether the data prepared for processing is appropriate or requires more finishing and seasoning. This phase focuses on the building of predictive/descriptive models.

Evaluation:

Model evaluation is done during model development. It checks for the quality of the model to be assessed and also if it meets the business requirements. It undergoes the diagnostic measure phase (the model works as intended and where are modifications required) and the statistical significance testing phase (ensures proper data handling and interpretation).

Deployment:

As the model is effectively evaluated it is made ready for deployment in the business market. The deployment phase checks how much the model can withstand the external environment and perform superiorly as compared to others.

Feedback:

Feedback is the necessary purpose which helps in refining the model and accessing its performance and impact. Steps involved in feedback define the review process, track the record, measure effectiveness and review with refining.

Essential Skills to Become a Data Scientist

Anyone interested in building a strong career in this domain should gain critical skills in three departments:

- Analytics
- Programming
- Domain knowledge

Going one level deeper, the following skills will help you carve out a niche as a data scientist:

- Strong knowledge of Python, SAS, R, Scala
- Hands-on experience in SQL database coding
- Ability to work with unstructured data from various sources like video and social media
- Understand multiple analytical functions
- Knowledge of machine learning

Machine Learning

- Machine learning can be defined as the practice of using algorithms to extract data, learn from it, and then forecast future trends for that topic.
- Traditional machine learning software is statistical analysis and predictive analysis that is used to spot patterns and catch hidden insights based on perceived data.
- A good example of machine learning implementation is Facebook. Facebook's machine learning algorithms gather behavioural information for every user on the social platform. Based on one's past behaviour, the algorithm predicts interests and recommends articles and notifications on the news feed.
- Similarly, when Amazon recommends products, or when Netflix recommends movies based on past behaviours, machine learning is at work.

Essential Skills to Become a Machine Learning Engineer

Machine learning is just a different perspective on statistics. The following are critical skills that can help you jumpstart your career in this fast-growing domain:

- Expertise in computer fundamentals
- In-depth knowledge of programming skills
- Knowledge of probability and statistics
- Data modelling and evaluation skills

1.6 Application of Data Analytics in different sectors

- It has been a widely acknowledged fact that big data has become a big game-changer in most modern industries over the last few years.
- As big data continues to permeate our day-to-day lives the number of different industries that are adopting big data continues to increase.
- It is well said that when new technologies become cheaper and easier to use, they have the potential to transform industries.
- That is exactly what is happening with big data right now. Here are 10 Industries redefined the most by big data analytics-

Government and Public Sector Services

- Analytics, data science, and big data have helped several cities to pilot the smart cities initiative where data collection, analytics and the IoT combine to create joined-up public services and utilities spanning the entire city.
- For example, a sensor network has been rolled out across all 80 of the council's neighbourhood recycling centres to help streamline collection services, so wagons can prioritize the fullest recycling centres and skip those with almost nothing in them.

Agriculture and Farming

- The power of AI has embraced even traditional industries like Agriculture and Farming.
- Big data practices have been adopted by the US agricultural manufacturer John Deere which has launched several data-enabled services that have led farmers to benefit from the real-time monitoring of data collected from its thousands of users worldwide.

Sports

- Most elite sports have now embraced data analytics. In Premier League football games, cameras installed around the stadiums track the movement of every player with the help of pattern recognition software generating over 25 data points per player every second.
- What more? NFL players have installed sensors on their shoulder pads to gather intelligent insights on their performance using data mining. It was analytics which helped British rowers row their way to the Olympic gold.

Hospitality

- Hotels and the luxury industry have turned to advanced analytics solutions to understand the secret behind customer satisfaction initiatives.
- Yield management in the hotel industry is one common use of analytics which is an important means to tackle the recurring peaks in demand throughout the year in consideration of other factors which include weather and local events, that can influence the number and nationalities of guests checking in.

Energy

- The costs of extracting oil and gas are rising, and the turbulent state of international politics adds to the difficulties of exploration and drilling for new reserves.
- In the energy industry Royal Dutch Shell, for example, has been developing the "data-driven oilfield" in an attempt to bring down the cost of drilling for oil.
- And on a smaller but no less important scale, data and the Internet of Things (IoT) are disrupting the way we use energy in our homes.
- The rise of "smart homes" includes technology like Google's Nest thermostat, which helps make homes more comfortable and cut down on energy wastage.

Education

- The education sector generates massive data through courseware and learning methodologies. Important insights can identify better teaching strategies, highlight areas where students may not be learning efficiently, and transform how the education is delivered.
- Increasingly educational establishments have been putting data into use for everything from planning school bus routes to improving classroom cleanliness.

Banking and Securities

- Securities Exchange Commission (SEC) has deployed big data to track and monitor the movements in the financial market.
- Big data and analytics with network analytics and natural language processors are used by the stock exchanges to catch illegal trade practices in the stock market.
- Retail traders, Big banks, hedge funds and other so-called ‘big boys’ in the financial markets use big data for trade analytics used in high-frequency trading, pre-trade decision-support analytics, sentiment measurement, predictive analytics, etc.
- This industry also heavily relies on big data for risk analytics including; anti-money laundering, demand enterprise risk management, “Know Your Customer”, and fraud mitigation.

Entertainment, Communications and the Media

- The on-demand music service, Spotify uses Hadoop big data analytics to collate data from its millions of users across the world.
- This data is used and analyzed to give customized music recommendations to its users. Over-the-top media, services have relied big on big data to offer customized content offerings to their users. An important move in the growing competitive market.

Retail and Wholesale Trade

- Big data has in a big way impacted the traditional brick-and-mortar retailers and wholesalers to current-day e-commerce traders.
- The retail and wholesale industry has gathered a lot of data over time which is derived from POS scanners, RFID, customer loyalty cards, store inventory, local demographics, etc.
- Big data applies to the retail and wholesale industry to mitigate fraud and offer customized products to the end-user thereby improving the user experience.

Transportation

- Big data analytics finds huge applications in the transportation industry.
- Governments of different countries use big data to control the traffic, optimize route planning and intelligent transport systems and congestion management.
- Moreover, the private sector uses big data in revenue management, technological enhancements, logistics and to gain a competitive advantage. Big data is improving user experiences, and the massive adoption change has just begun.

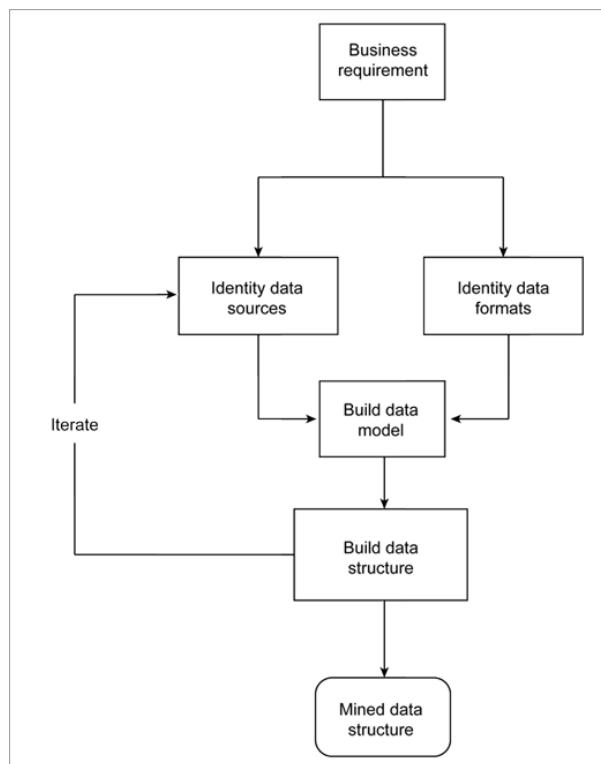
1.7 Data Mining and Data Profiling

Data Mining

- Data mining is the process of sorting through large data sets to identify patterns and relationships that can help solve business problems through data analysis.
- Data mining techniques and tools enable enterprises to predict future trends and make more-informed business decisions.

Data Mining Process

- Any business problem will examine the raw data to build a model that will describe the information and bring out the reports to be used by the business.
- Building a model from data sources and data formats is an iterative process as the raw data is available in many different sources and many forms.
- Data is increasing day by day, hence when a new data source is found, it can change the results.



Data Cleaning

- Data cleaning is the first step in data mining.
- It holds importance as dirty data if used directly in mining can cause confusion in procedures and produce inaccurate results.
- Basically, this step involves the removal of noisy or incomplete data from the collection.
- Many methods that generally clean data by itself are available but they are not robust.

This step carries out the routine cleaning work by:

- (i) Fill The Missing Data:
 - Missing data can be filled by methods such as:
 - Ignoring the tuple.
 - Filling the missing value manually.
 - Use the measure of central tendency, median or
 - Filling in the most probable value.

ii) Remove The Noisy Data:

Random error is called noisy data.

Methods to remove noise are:

Binning: Binning methods are applied by sorting values into buckets or bins. Smoothening is performed by consulting the neighboring values.

- Binning is done by smoothing by bin i.e. each bin is replaced by the mean of the bin. Smoothing by a median, where each bin value is replaced by a bin median.
- Smoothing by bin boundaries i.e. The minimum and maximum values in the bin are bin boundaries and each bin value is replaced by the closest boundary value.
 - Identifying the Outliers
 - Resolving Inconsistencies

Data Integration

- When multiple heterogeneous data sources such as databases, data cubes or files are combined for analysis, this process is called data integration.
- This can help in improving the accuracy and speed of the data mining process.
- Different databases have different naming conventions of variables, by causing redundancies in the databases.
- Additional Data Cleaning can be performed to remove the redundancies and inconsistencies from the data integration without affecting the reliability of data.
- Data Integration can be performed using Data Migration Tools such as Oracle Data Service Integrator and Microsoft SQL etc.

Data Reduction

This technique is applied to obtain relevant data for analysis from the collection of data. The size of the representation is much smaller in volume while maintaining integrity. Data Reduction is performed using methods such as Naive Bayes, Decision Trees, Neural network, etc.

Some strategies of data reduction are:

- **Dimensionality Reduction:** Reducing the number of attributes in the dataset.
- **Numerosity Reduction:** Replacing the original data volume by smaller forms of data representation.
- **Data Compression:** Compressed representation of the original data.

Data Transformation

In this process, data is transformed into a form suitable for the data mining process. Data is consolidated so that the mining process is more efficient and the patterns are easier to understand. Data Transformation involves Data Mapping and code generation process.

Strategies for data transformation are:

- **Smoothing:** Removing noise from data using clustering, regression techniques, etc.
- **Aggregation:** Summary operations are applied to data.
- **Normalization:** Scaling of data to fall within a smaller range.
- **Discretization:** Raw values of numeric data are replaced by intervals. For Example, Age.

Data Mining

- Data Mining is a process to identify interesting patterns and knowledge from a large amount of data.
- In these steps, intelligent patterns are applied to extract the data patterns.
- The data is represented in the form of patterns and models are structured using classification and clustering techniques.

Pattern Evaluation

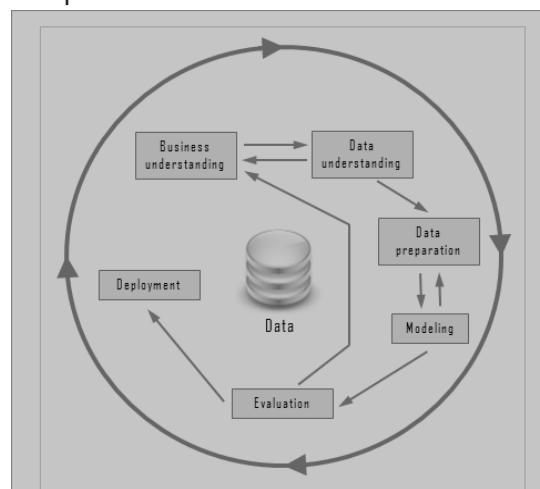
- This step involves identifying interesting patterns representing the knowledge based on interestingness measures.
- Data summarization and visualization methods are used to make the data understandable by the user.

Knowledge Representation

- Knowledge representation is a step where data visualization and knowledge representation tools are used to represent the mined data.
- Data is visualized in the form of reports, tables, etc.

Cross-Industry Standard Process for Data Mining

- CRISP-DM is a reliable data mining model consisting of six phases.
- It is a cyclical process that provides a structured approach to the data mining process.
- The six phases can be implemented in any order but it would sometimes require backtracking to the previous steps and repetition of actions.



Business Understanding:

- In this step, the goals of the businesses are set and the important factors that will help in achieving the goal are discovered.

Data Understanding:

- This step will collect the whole data and populate the data in the tool (if using any tool).
- The data is listed with its data source, location, how it is acquired and if any issue encountered. Data is visualized and queried to check its completeness.

Data Preparation:

- This step involves selecting the appropriate data, cleaning, constructing attributes from data, integrating data from multiple databases.

Modeling:

- Selection of the data mining technique such as decision-tree, generate test design for evaluating the selected model, building models from the dataset and assessing the built model with experts to discuss the result is done in this step.

Evaluation:

- This step will determine the degree to which the resulting model meets the business requirements.
- Evaluation can be done by testing the model on real applications.
- The model is reviewed for any mistakes or steps that should be repeated.

Deployment:

- In this step a deployment plan is made, strategy to monitor and maintain the data mining model results to check for its usefulness is formed, final reports are made and review of the whole process is done to check any mistake and see if any step is repeated.

Data Mining Challenges

Enlisted below are the various challenges involved in Data Mining:

- Data Mining needs large databases and data collection that are difficult to manage.
- The data mining process requires domain experts that are again difficult to find.
- Integration from heterogeneous databases is a complex process.
- The organizational level practices need to be modified to use the data mining results. Restructuring the process requires effort and cost.

Data Profiling

- Data profiling refers to the process of examining, analyzing, reviewing and summarizing data sets to gain insight into the quality of data.
- Data quality is a measure of the condition of data based on factors such as its accuracy, completeness, consistency, timeliness and accessibility.
- Data profiling may also be known as data archeology, data assessment, data discovery or data quality analysis.
- Organizations use data profiling at the beginning of a project to determine if enough data has been gathered, if any data can be reused or if the project is worth pursuing.
- The process of data profiling itself can be based on specific business rules that will uncover how the data set aligns with business standards and goals.

Types of Data Profiling

There are three types of data profiling.

- **Structure discovery:** This focuses on the formatting of the data, making sure everything is uniform and consistent. It uses basic statistical analysis to return information about the validity of the data.
- **Content discovery:** This process assesses the quality of individual pieces of data. For example, ambiguous, incomplete and null values are identified.
- **Relationship discovery:** This detects connections, similarities, differences and associations among data sources.

Steps of Data Profiling

A high-level breakdown of the process is as follows:

1. The first step of data profiling is gathering one or multiple data sources and the associated metadata for analysis.
2. The data is then cleaned to unify structure, eliminate duplications, identify interrelationships and find anomalies.
3. Once the data is cleaned, data profiling tools will return various statistics to describe the data set. This could include the mean, minimum/maximum value, frequency, recurring patterns, dependencies or data quality risks.

For example, by examining the frequency distribution of different values for each column in a table, a data analyst could gain insight into the type and use of each column. Cross-column analysis can be used to expose embedded value dependencies; inter-table analysis allows the analyst to discover overlapping value sets that represent foreign key relationships between entities.

Benefits of Data Profiling

Data profiling returns a high-level overview of data that can result in the following benefits:

- Leads to higher-quality, more credible data
- Helps with more accurate predictive analytics and decision-making
- Makes better sense of the relationships between different data sets and sources
- Keeps company information centralized and organized
- Eliminates errors, such as missing values or outliers, that add costs to data-driven projects
- Highlights areas within a system that experience the most data quality issues, such as data corruption or user input errors
- Produces insights surrounding risks, opportunities and trends.

Example of Data Profiling

- Projects that involve data warehousing or business intelligence may require gathering data from multiple disparate systems or databases for one report or analysis.
- Applying data profiling to these projects can help identify potential issues and corrections that need to be made in extract, transform and load (ETL) jobs and other data integration processes before moving forward.
- Additionally, data profiling is crucial in data conversion or data migration initiatives that involve moving data from one system to another.
- Data profiling can help identify data quality issues that may get lost in translation or adaptions that must be made to the new system prior to migration.

Techniques used in Data Profiling

The following four methods, or techniques, are used in data profiling:

- Column profiling, which assesses tables and quantifies entries in each column
- Cross-column profiling, which features both key analysis and dependency analysis
- Cross-table profiling, which uses key analysis to identify stray data as well as semantic and syntactic discrepancies
- Data rule validation, which assesses data sets against established rules and standards to validate that they're being followed

1.8 Data Wrangling

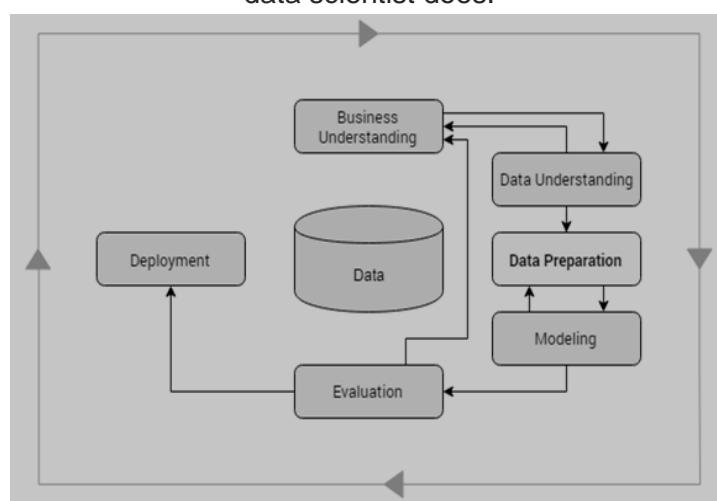
- Data wrangling is the process of removing errors and combining complex data sets to make them more accessible and easier to analyze.
- Due to the rapid expansion of the amount of data and data sources available today, storing and organizing large quantities of data for analysis is becoming increasingly necessary.
- What is “raw data”? It is any repository data (texts, images, database records) that is documented but yet to be processed and fully integrated into the system.
- The process of wrangling can be described as “digesting” data (often referred to as “munging” thus the alternative term “data wrangling techniques”) and making it useful (aka usable) for the system.
- It can be described as a preparation stage for every other data-related operation.
- Wrangling the data is usually accompanied by Mapping.
- The term “Data Mapping” refers to the element of the wrangling process that involves identifying source data fields to their respective target data fields.
- While Wrangling is dedicated to transforming data, Mapping is about connecting the dots between different elements.

Data Wrangling Process

Data wrangling covers the following processes:

- Getting data from the various source into one place
- Piecing the data together according to the determined setting
- Cleaning the data from the noise or erroneous, missing elements

It should be noted that Data Wrangling is a somewhat demanding and time-consuming operation both from computational capacities and human resources. Data wrangling takes over half of what data scientist does.

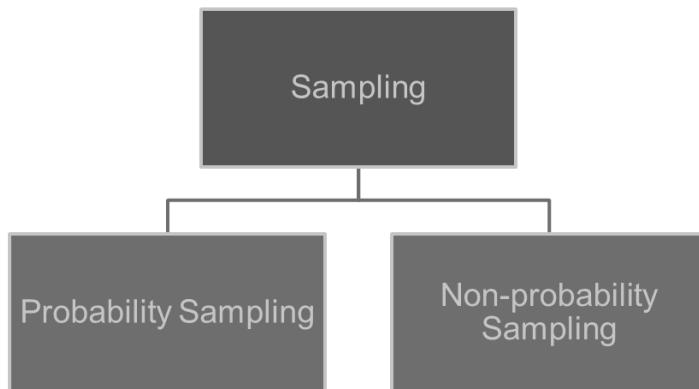


- The term “wrangling” refers to rounding up information in a certain way.
- This operation includes a sequence of the following processes:
 1. **Preprocessing** — the initial state that occurs right after the acquiring of data.
 2. **Standardizing data** into an understandable format. For example, you have a user profile events record, and you need to sort it by types of events and time stamps.
 3. **Cleaning data** from noise, missing, or erroneous elements.
 4. **Consolidating data** from various sources or data sets into a coherent whole. For example, you have an affiliate advertising network, and you need to gather performance statistics for the current stage of the marketing campaign.
 5. **Matching data** with the existing data sets. For example, you already have user data for a certain period and unite these sets into a more expansive one
 6. **Filtering data** through determined settings for the processing.

1.9 Sampling Techniques in Data Analytics

- Sampling is the practice of selecting an individual group from a population to study the whole population.
- Let's say we want to know the percentage of people who use iPhones in a city, for example. One way to do this is to call up everyone in the city and ask them what type of phone they use. The other way would be to get a smaller subgroup of individuals and ask them the same question, and then use this information as an approximation of the total population.

Types of Sampling Techniques in Data Analytics



- Sampling type comes under two broad categories:
 - **Probability sampling** - Random selection techniques are used to select the sample.
 - **Non-probability sampling** - Non-random selection techniques based on certain criteria are used to select the sample.

1. Simple Random Sampling

- In simple random sampling, the researcher selects the participants randomly. There are a number of data analytics tools like random number generators and random number tables used that are based entirely on chance.
- *Example:* The researcher assigns every member in a company database a number from 1 to 1000 (depending on the size of your company) and then use a random number generator to select 100 members.

2. Systematic Sampling

- In systematic sampling, every population is given a number as well like in simple random sampling. However, instead of randomly generating numbers, the samples are chosen at regular intervals.
- *Example:* The researcher assigns every member in the company database a number. Instead of randomly generating numbers, a random starting point (say 5) is selected. From that number onwards, the researcher selects every, say, 10th person on the list (5, 15, 25, and so on) until the sample is obtained.

3. Stratified Sampling

- In stratified sampling, the population is subdivided into subgroups, called strata, based on some characteristics (age, gender, income, etc.).
- After forming a subgroup, you can then use random or systematic sampling to select a sample for each subgroup.
- This method allows you to draw more precise conclusions because it ensures that every subgroup is properly represented.
- *Example:* If a company has 500 male employees and 100 female employees, the researcher wants to ensure that the sample reflects the gender as well. So, the population is divided into two subgroups based on gender.

4. Cluster Sampling

- In cluster sampling, the population is divided into subgroups, but each subgroup has similar characteristics to the whole sample.
- Instead of selecting a sample from each subgroup, you randomly select an entire subgroup.
- This method is helpful when dealing with large and diverse populations.
- *Example:* A company has over a hundred offices in ten cities across the world which has roughly the same number of employees in similar job roles. The researcher randomly selects 2 to 3 offices and uses them as the sample.

1. Convenience Sampling

- In this sampling method, the researcher simply selects the individuals which are most easily accessible to them.
- This is an easy way to gather data, but there is no way to tell if the sample is representative of the entire population.
- The only criteria involved is that people are available and willing to participate.
- *Example:* The researcher stands outside a company and asks the employees coming in to answer questions or complete a survey.

2. Voluntary Response Sampling

- Voluntary response sampling is similar to convenience sampling, in the sense that the only criterion is people are willing to participate.
- Instead of the researcher choosing the participants, the participants volunteer themselves.
- *Example:* The researcher sends out a survey to every employee in a company and gives them the option to take part in it.

3. Purposive Sampling

- In purposive sampling, the researcher uses their expertise and judgment to select a sample that they think is the best fit.
- It is often used when the population is very small and the researcher only wants to gain knowledge about a specific phenomenon rather than make statistical inferences.
- *Example:* The researcher wants to know about the experiences of disabled employees at a company. So, the sample is purposefully selected from this population.

4. Snowball Sampling

- In snowball sampling, the research participants recruit other participants for the study.
- It is used when participants required for the research are hard to find.
- It is called snowball sampling because like a snowball, it picks up more participants along the way and gets larger and larger.
- *Example:* The researcher wants to know about the experiences of homeless people in a city. Since there is no detailed list of homeless people, a probability sample is not possible. The only way to get the sample is to get in touch with one homeless person who will then put you in touch with other homeless people in a particular area.

1.10 Analytics Framework in and Latest Trends

You will be surprised by the fact that each day we are producing more data in 2 days than in decades of history. Yes, that's true, and most of us do not even realize this thing that we produce so much data just by browsing the Internet. If you don't want the future technologies to catch you off guard, pay attention to these current trends in big data analytics and succeed!

Data As Service

- Traditionally the Data is stored in data stores, developed to obtain by particular applications. When the SaaS (software as a service) was popular, Daas was just the beginning.
- As with Software-as-a-Service applications, Data as a service uses cloud technology to give users and applications with on-demand access to information without depending on where the users or applications may be.
- Data as a Service is one of the current trends in big data analytics and will deliver it simpler for analysts to obtain data for business review tasks and easier for areas throughout a business or industry to share data.

Responsible and Smarter Artificial Intelligence

- Responsible and Scalable AI will enable better learning algorithms with a shorter time to market. Businesses will achieve a lot more from AI systems like formulating processes that can function efficiently.
- Businesses will find a way to take AI to scale, which has been a great challenge till now.

Predictive Analytics

- Big data analytics has always been a fundamental approach for companies to become a competing edge and accomplish their aims.
- They apply basic analytics tools to prepare big data and discover the causes of why specific issues arise.
- Predictive methods are implemented to examine modern data and historical events to know customers and recognize possible hazards and events for a corporation.
- Predictive analysis in big data can predict what may occur in the future.
- This strategy is extremely efficient in correcting analyzed assembled data to predict customer response. This enables organizations to define the steps they have to practice by identifying a customer's next move before they even do it.

Quantum Computing

- Using current time technology can take a lot of time to process a huge amount of data. Whereas, Quantum computers, calculate the probability of an object's state or an event before it is measured, which indicates that they can process more data than classical computers.
- If only we compress billions of data at once in only a few minutes, we can reduce processing duration immensely, providing organizations with the possibility to gain timely decisions to attain more aspired outcomes.
- This process can be possible using Quantum computing. The experiment of quantum computers to correct functional and analytical research over several enterprises can make the industry more precise.

Edge Computing

- Running processes and moving those processes to a local system such as any user's system or IoT device or a server defines Edge Processing.
- Edge computing brings computation to a network's edge and reduces the amount of long-distance connection that has to happen between a customer and a server, which is making it the latest trend in big data analytics.
- It provides a boost to Data Streaming, including real-time data Streaming and processing without containing latency.
- It enables the devices to respond immediately. Edge computing is an efficient way to process massive data by consuming less bandwidth usage. It can reduce the development cost for an organization and help the software run in remote locations.

Natural Language Processing

- Natural Language Processing (NLP) lies inside artificial intelligence and works to develop communication between computers and humans.
- The objective of NLP is to read and decode the meaning of the human language. Natural language processing is mostly based on machine learning, and it is used to develop word processor applications or translating software.
- Natural Language Processing Techniques need algorithms to recognize and obtain the required data from each sentence by applying grammar rules.
- Mostly syntactic analysis and semantic analysis are the techniques that are used in natural language processing. Syntactic analysis is the one that handles sentences and grammatical issues, whereas semantic analysis handles the meaning of the data/text.

Hybrid Clouds

- A cloud computing system utilizes an on-premises private cloud and a third-party public cloud with orchestration between two interfaces.
- The hybrid cloud provides excellent flexibility and more data deployment options by moving the processes between private and public clouds. An organization must have a private cloud to gain adaptability with the aspired public cloud.
- For that, it has to develop a data center, including servers, storage, LAN, and load balancer. The organization has to deploy a virtualization layer/hypervisor to support the VMs and containers. And, install a private cloud software layer.
- The implementation of software allows instances to transfer data between the private and public clouds.

Dark Data

- Dark data is the data that a company does not use in any analytical system. The data is gathered from several network operations that are not used to determine insights or for prediction.
- The organizations might think that this is not the correct data because they are not getting any outcome from that. But they know that this will be the most valuable thing.
- As the data is growing day by day, the industry should understand that any unexplored data can be a security risk. The expansion in the amount of Dark Data can be seen as another Trend.

Data Fabric

- Data fabric is an architecture and collection of data networks. That provides consistent functionality across a variety of endpoints, both on-premises and cloud environments.
- To drive digital transformation, Data Fabric simplifies and incorporates data storage across cloud and on-premises environments.
- It enables access and sharing of data in a distributed data environment. Additionally provides a consistent data management framework across un-siloed storage.

XOps

- The aim of XOps (data, ML, model, platform) is to achieve efficiencies and economies of scale. XOps is achieved by implementing DevOps best practices. Thus, ensuring efficiency, reusability, and repeatability while reducing technology, process replication and allowing automation.
- These innovations would enable prototypes to be scaled, with flexible design and agile orchestration of governed systems.

Section 3: Exercises

Exercise 1: Sample Data

Husband_Age	Wife_Age	Husband_Income	Wife_Income	Number_Of_Bedrooms	Electricity_Units	Gas	Number_Of_Children	Internet_Connection	Mode	House_Owned/Rented	Speaking_Language	Decade_Of_House_Built
64	62	110000	29100	3	230	90	0	Yes	Followup	Rented	English only	1940
63	64	100000	31000	4	230	30	0	Yes	mail	Rented	English only	1950
56	51	31000	NA	2	200	40	0	No	Followup	Rented	English only	1960
72	68	31000	38000	2	210	3	0	Yes	Internet	Owned free and clear	English only	1950
37	33	16400	24000	3	260	9	2	Yes	Internet	Rented	English only	1990
86	91	77600	30000	4	20	30	0	Yes	Internet	Owned free and clear	English only	1980
67	67	8400	4800	4	70	150	0	Yes	Internet	Owned free and clear	Other	1960
70	74	73870	21000	0	180	30	0	Yes	Internet	Owned free and clear	English only	2000
33	31	545050	6000	2	20	30	0	Yes	Internet	Rented	English only	1930
41	47	42000	36000	3	80	200	2	Yes	Followup	Rented	English only	2000
37	36	50000	49000	4	60	200	2	Yes	mail	Owned free and clear	English only	1950
82	77	48800	21000	3	100	8	0	Yes	mail	Owned free and clear	English only	2000
56	50	109000	50000	2	160	3	2	Yes	Internet	Rented	English only	1930
64	63	95000	NA	4	80	90	0	Yes	Internet	Rented	Other	1990
74	75	39400	9700	2	70	130	0	No	mail	Rented	English only	1960
63	60	12000	12000	4	160	3	0	Yes	mail	Rented	English only	1930
59	54	210000	3900	4	230	140	2	Yes	Followup	Rented	English only	1950
46	45	40000	33000	3	60	60	2	Yes	Internet	Rented	English only	1980
63	58	22400	17660	3	110	3	0	Yes	mail	Rented	English only	1990
39	40	76000	10700	3	110	3	2	Yes	Internet	Rented	English only	1960
63	61	24000	24000	2	50	15	0	Yes	Followup	Rented	English only	1950
50	53	345000	64000	3	40	20	0	Yes	Internet	Rented	English only	1980
46	36	32000	35000	3	100	30	0	Yes	Followup	Rented	English only	2000
30	21	23000	18000	2	130	3	0	Yes	Internet	Rented	English only	1940

- House_number
- Husband_Age
- Wife_Age
- Husband_Income
- Wife_Income
- Number_Of_Bedrooms
- Electricity_Units
- Gas
- Number_Of_Children, Internet_Connection
- Mode
- House_Owned/Rented, Speaking_Language
- Decade_Of_House_Built

Problem Statement:

To find out the following:

- Know the minimum, maximum and average Age of the Wife
- Know the median, quantile, variance and standard deviation of Husband Income
- Find the frequency of the Number of Children and Number of Bedrooms

Exercise 2: Draw a diagram of Data Mining process.

Exercise 3: Draw flow chart of Data Wrangling process.

Exercise 4: Participate in a group discussion on following topics:

- a) Need and Importance of Data Analysis
- b) Types of Data
- c) Data Analytics in Different Sectors
- d) Sampling Techniques
- e) Data Wrangling
- f) Data Mining
- g) Data Profiling
- h) Analytics Framework in and Latest Trends

Section 4: Assessment Questionnaire

1. Mention the differences between Data Mining and Data Profiling?
2. Define the term 'Data Wrangling' in Data Analytics.
3. What are the various steps involved in any analytics project?
4. What are the common problems that data analysts encounter during analysis?
5. Which are the technical tools that you can use for analysis and presentation purposes?
6. What are the best methods for data cleaning?
7. What is the significance of Exploratory Data Analysis (EDA)?
8. Explain descriptive, predictive, and prescriptive analytics.
9. What are the different types of sampling techniques used by data analysts?
10. Describe univariate, bivariate, and multivariate analysis.
11. What are the essential skills to become data analyst?
12. What is Data Science?
13. What is Machine Learning?
14. What are Binning Methods?
15. What are the six phases of CRISP-DM?
16. _____ refers to the process of examining, analyzing, reviewing and summarizing data sets to gain insight into the quality of data.
17. Organizations use data profiling at the beginning of a project. (True/False)
18. What are the types of Data Profiling?
19. What are the Techniques used in Data Profiling?
20. The term _____ refers to the element of the wrangling process that involves identifying source data fields to their respective target data fields.
21. What is data wrangling process?

-----End of the Module-----

MODULE 2

BUSINESS ANALYTICS WITH EXCEL

Section 1: Learning Outcomes

After completing this module, you will be able to:

- Introduce the term ‘Business Analytics’
- Perform Conditional Formatting in Excel
- Analyse Data with Pivot Tables
- Build the Dashboard
- Analyse Data Using Statistics

Section 2: Relevant Knowledge

2.1 Introduction to Business Analytics

What is Business Analytics?

- As per a popular definition from authors Michael J Beller and Alan Barnett, “Business analytics refers to the skills, technologies, and practices for continuous iterative exploration and investigation of past business performance to gain insight and drive business planning”.
- If we study this definition and relate it to the present practices, the only things that might have been added are, one, the real-time analysis available on leading analytical software and two, the predictive analytics that is a standard feature in present-day analytics software.



- Business analytics is measuring the performance to make improvements to the bottom line of the business.

Types of Business Analytics

Let's delve into the types of Business Analytics. Primarily there are 4 types.

Descriptive Analytics

- The first generation of business analytics was based on studying historic data and drawing inferences about the performance of the business.
- This is exactly what descriptive analytics does with the available data.
- Summarizing data into a few key metrics give a reasonable understanding of how well or not well a business is doing.

Diagnostic Analytics

- This type of analytics is the next organic step after descriptive Analytics.
- It focuses on the how and why of past business performances and aims to take learning from this historic data to correct course.
- A root cause analysis might also be initiated using various industry-standard practices.

Predictive Analytics

- An advanced technique that combines historic data, statistical models, and machine learning, to arrive at a most probabilistic future performance given that the business ecosystem stays pretty much the same.
- Predictive analytics is a great way to understand how the business performs if certain business inputs are tweaked and then work towards optimizing those business input parameters.

Prescriptive Analytics

- A step ahead of predictive analytics is the ability of the analytical system to be able to suggest the optimal solution, in other words, a recommendation system that puts forth the best course of action after optimizing the chances of a beneficial outcome.

Scope of Business Analytics

- Quality consumer experience by studying their purchase habits and behaviours.
- Businesses can streamline supply chain processes and reduce overhead costs.
- Companies can study customers' reactions towards their marketing campaigns and product offerings to create targeted campaigns and identify the most effective cross-sell and up-sell opportunities.
- Big data and business analytics allow companies to handle their finances more effectively.
- Business analytics is a much-evolved domain today with it attaining the status of science. Also, a term used interchangeably with Data Science, business analytics has scope in every field that aims to improve in every aspect as long as there is data to measure.
- With the proliferation of IoT and the data explosion that it will bring about, the standard methods applied in business analytics to sift through tons of data to see what's happening behind this curtain of numbers and to make sense of all of it, to make better decisions will be the game changer for many businesses in the future.
- So, in today's world, a team that studies the data that a business churns out day in and day out and to step back and look at the larger picture is very crucial for the long-term survival of businesses.

As per McKinsey, business analytical capabilities are projected to create \$9.5 trillion to \$15.4 trillion annual economic impacts across nineteen industries worldwide, of which 40% can be credited to AI implementation.

Use Cases of Business Analytics

Finance

Business Analytics can help financial organizations to optimize budgeting, determine creditworthiness in case of a loan, and also suggest the chances of a customer defaulting on a loan. In a business that grapples with fraud, Business Analytics plays an important role in raising red flags in time.



Credit Card Business

- Business Analytics helps in extracting crucial information hidden behind the credit and debit transactions and lets the business know, the spending habits, lifestyle preferences, and financial standing, raising red flags wherever there is a probability of loss of business.
- Credit card companies can decide on the fly which customers they can extend a line of credit to and by how much.

**Customer Relationship Management (CRM)**

Business analytics built into today's CRM systems, enable businesses to gain deep insights into demographics, socio-economic information and lifestyle of their customer groups and what would be the best fit strategy to retain and increase the customer base.

Marketing

Business analytics plays an important role in determining the effectiveness of marketing campaigns by generating insights on which kind of campaign is most effective and which one is most penetrative in the market. How much each type of campaign should be invested in to gain maximum benefits and cut losses.

E Retailing

- Today the e-retailing business is expanding like never before with more and more people preferring to order online than visit brick-and-mortar stores with covid pandemic attenuating it further.
- There are many players in the market and it becomes necessary for the e-retailer to keep a hawk's eye on inventories to maintain with suppliers and keep the pricing competitive while cutting losses.

Business Analytics Software

There are business analytics software solutions customized to markets it serves. In many cases business analytics software comes bundled with whole systems like CRM and Financial management systems.

SAP Business Intelligence

- A software suite that offers predictive analytics via machine learning models, bundled with reporting and data analysis tools.
- Being mobile has become a necessity and software in this domain are no exceptions. There is a mobile-friendly version of the software.

Microstrategy

- A business intelligence tool that offers high speed and dynamic dashboard and data analysis helping the business catch trends, identify new opportunities, and more.
- The ability to read any type of data source, be it structured or semi-structured, locally stored or cloud-based is an important characteristic of today's analytical software programs.

Wrike

- Wrike is a project management tool that runs in the cloud.
- It aids in the establishing of deadlines, scheduling, and resource allocation.
- Business analysts may update and provide tasks from anywhere using Android and iOS apps.

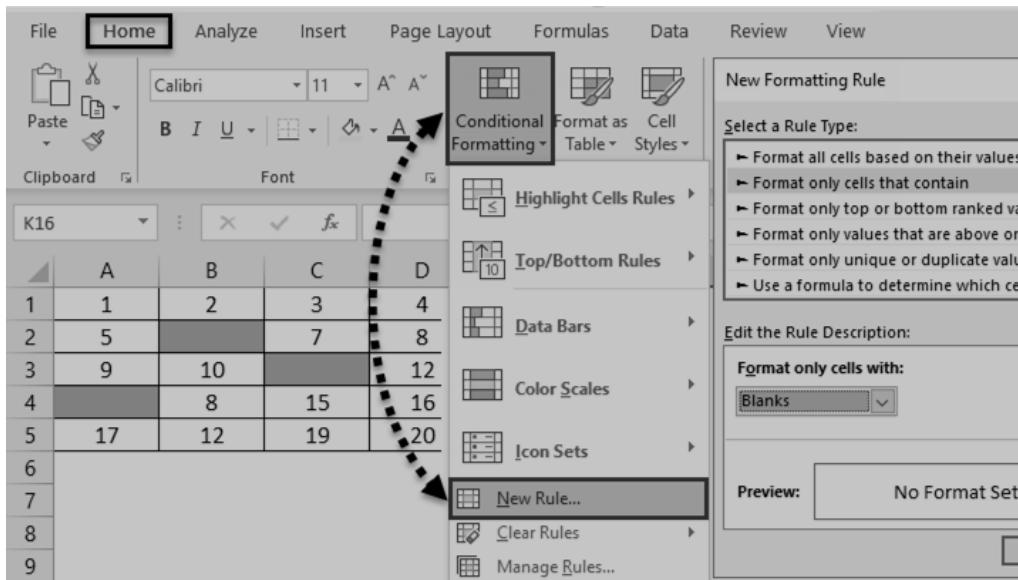


Oribi

- Custom reports, automated event collecting, visitor journey, and email capturing are just a few of Oribi's features.
- It is appropriate for all types of businesses.
- With only a few clicks, a business analyst can simply design marketing funnels and track where visitors are leaving.
- It has event monitoring capabilities and allows you to define conversion targets without writing any code.

2.2 Conditional Formatting and Important Functions

- Conditional formatting is a feature in Microsoft Excel that allows you to apply specific formatting to your cells according to certain criteria.
- It enables you to make sense of your data and spot significant trends.



- Conditional formatting in Excel enables you to highlight cells with a certain color, depending on the cell's value.
- We can perform following operations:
 - Highlight Cells Rules
 - Clear Rules
 - Top/Bottom
 - Conditional Formatting with Formulas
 - Finding Duplicate and Triplicate Values
 - Finding Duplicate Rows
 - Data Bars and Colour Scale
 - Icon Sets
 - Manage and Conflicting Rules

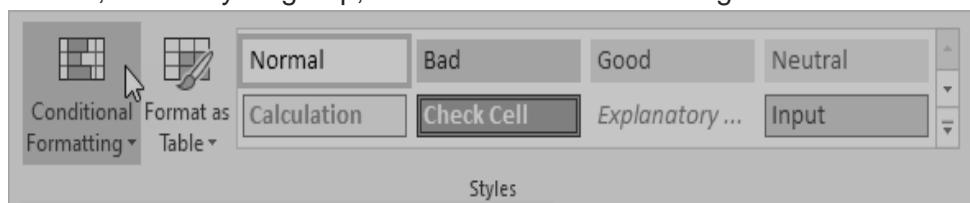
Highlight Cells Using Conditional Formatting

To highlight cells that are greater than a value, execute the following steps.

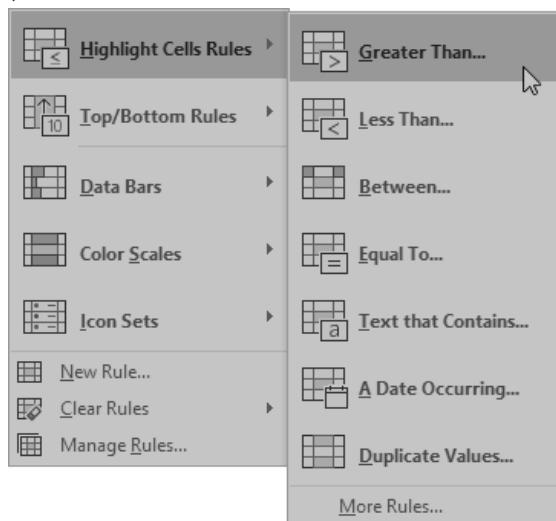
1. Select the range A1:A10.

	A	B
1	14	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

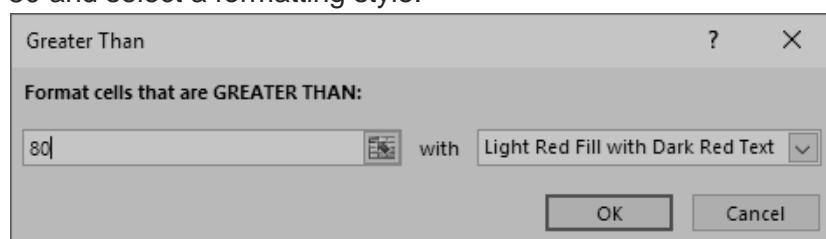
2. On the Home tab, in the Styles group, click Conditional Formatting.



3. Click Highlight Cells Rules, Greater Than.



4. Enter the value 80 and select a formatting style.



5. Click OK.

Result: Excel highlights the cells that are greater than 80.

	A	B
1	14	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

6. Change the value of cell A1 to 81.

Result: Excel changes the format of cell A1 automatically.

	A	B
1	81	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

You can also use this category (see step 3) to highlight cells that are less than a value, between two values, equal to a value, cells that contain specific text, dates (today, last week, next month, etc.), duplicates or unique values.

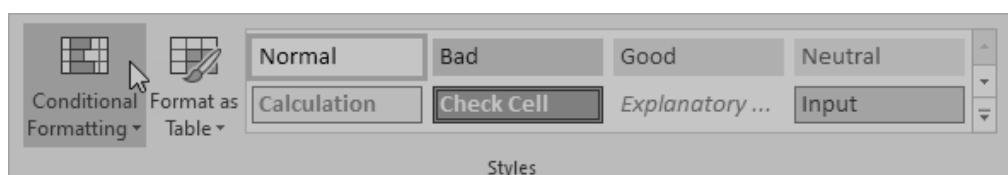
Clear Rules

To clear a conditional formatting rule, execute the following steps:

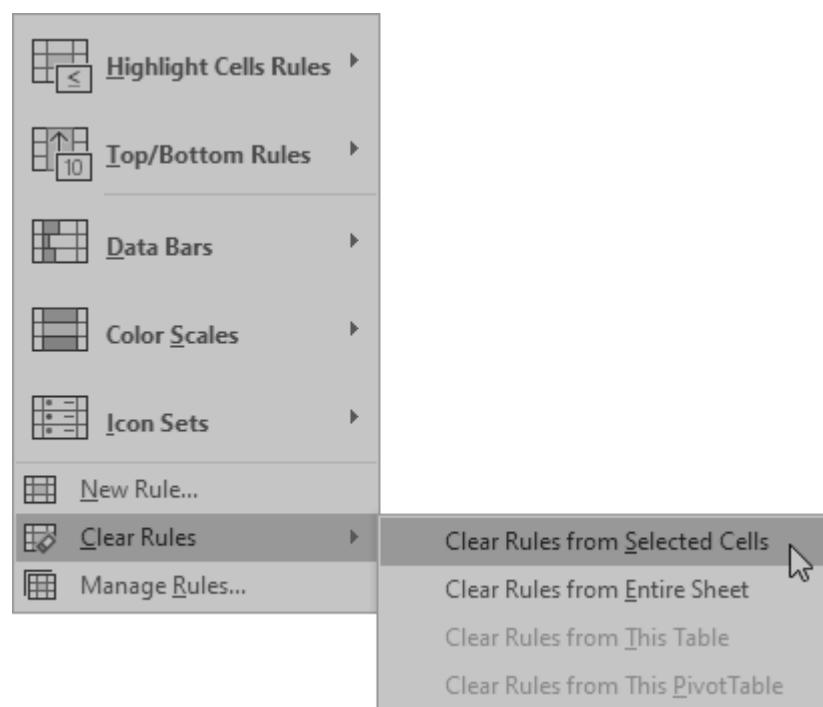
1. Select the range A1:A10.

	A	B
1	81	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

2. On the Home tab, in the Styles group, click Conditional Formatting.



3. Click Clear Rules, Clear Rules from Selected Cells.



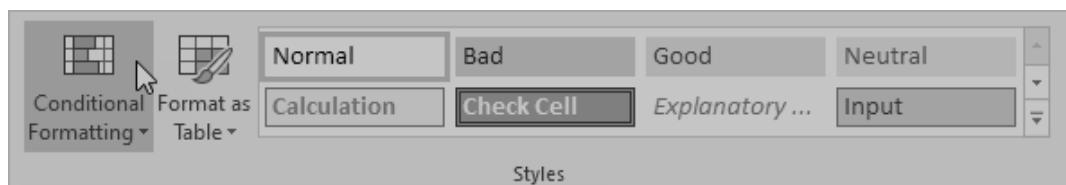
Top/Bottom

To highlight cells that are above average, execute the following steps.

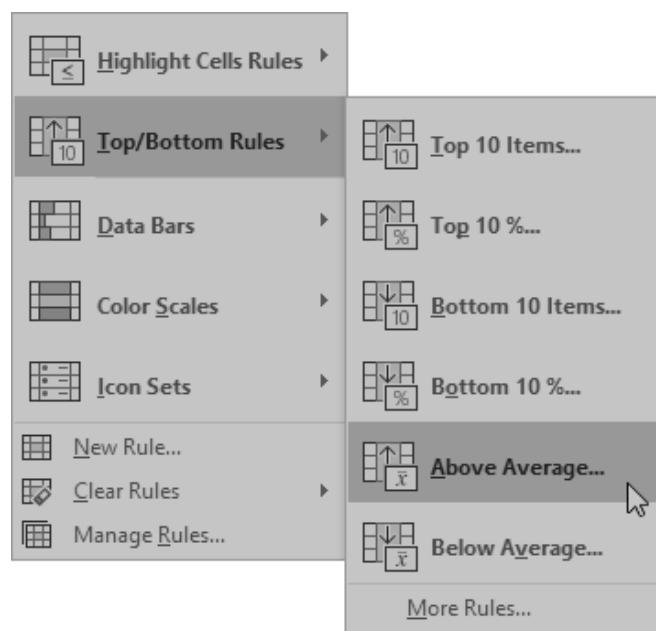
1. Select the range A1:A10.

	A	B
1	81	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

2. On the Home tab, in the Styles group, click Conditional Formatting.



3. Click Top/Bottom Rules, Above Average.



4. Select a formatting style.



5. Click OK.

Result: Excel calculates the average (42.5) and formats the cells that are above this average.

	A	B
1	81	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

You can also use this category (see step 3) to highlight the top n items, the top n percent, the bottom n items, the bottom n percent or cells that are below average.

Conditional Formatting With Formulas

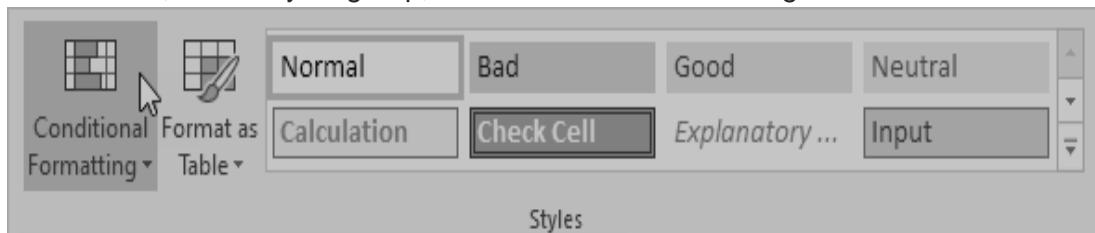
- Take your Excel skills to the next level and use a formula to determine which cells to format.
- Formulas that apply conditional formatting must evaluate to TRUE or FALSE.

Example 1:

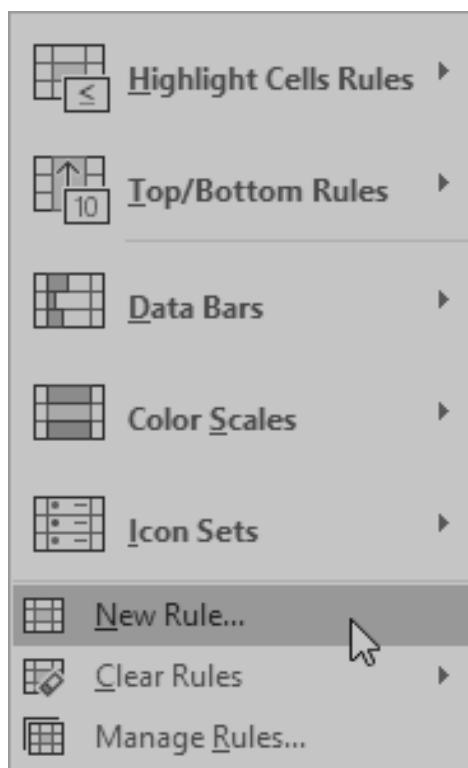
1. Select the range A1:E5.

	A	B	C	D	E	F
1	90	77	33	20	96	
2	59	66	20	61	44	
3	94	99	97	41	52	
4	36	43	70	13	54	
5	15	6	28	28	15	
6						

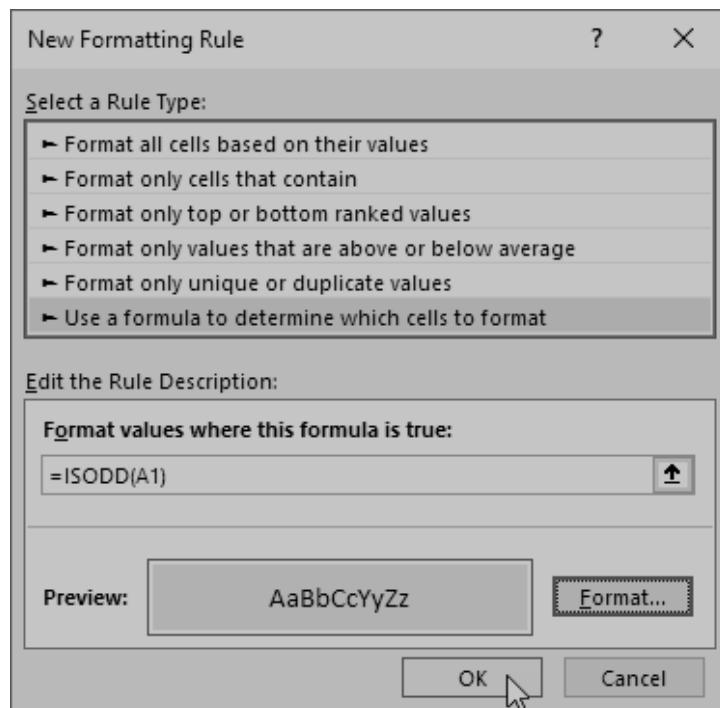
2. On the Home tab, in the Styles group, click Conditional Formatting.



3. Click New Rule.



4. Select 'Use a formula to determine which cells to format'.
5. Enter the formula =ISODD(A1)
6. Select a formatting style and click OK.



Result: Excel highlights all odd numbers.

	A	B	C	D	E	F
1	90	77	33	20	96	
2	59	66	20	61	44	
3	94	99	97	41	52	
4	36	43	70	13	54	
5	15	6	28	28	15	
6						

Explanation: Always write the formula for the upper-left cell in the selected range. Excel automatically copies the formula to the other cells. Thus, cell A2 contains the formula =ISODD(A2), cell A3 contains the formula =ISODD(A3), etc.

Example 2:

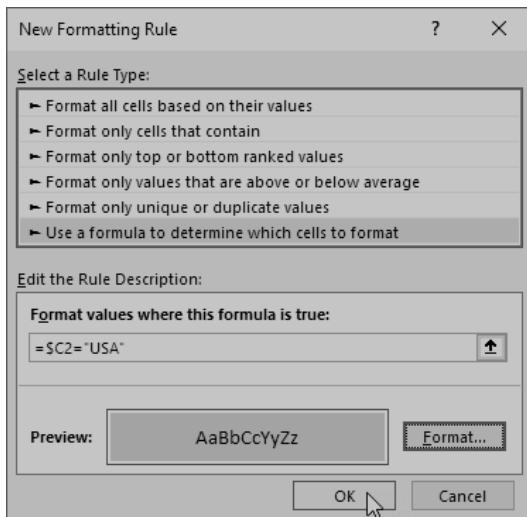
7. Select the range A2:D7.

	A	B	C	D	E
1	Last Name	Sales	Country	Quarter	
2	Smith	\$16,753.00	UK	Qtr 3	
3	Johnson	\$14,808.00	USA	Qtr 4	
4	Williams	\$10,644.00	UK	Qtr 2	
5	Jones	\$1,390.00	USA	Qtr 3	
6	Brown	\$4,865.00	USA	Qtr 4	
7	Williams	\$12,438.00	UK	Qtr 1	
8					

8. Repeat steps 2-4 above.

9. Enter the formula `=$C2="USA"`

10. Select a formatting style and click OK.



Result: Excel highlights all USA orders.

	A	B	C	D	E
1	Last Name	Sales	Country	Quarter	
2	Smith	\$16,753.00	UK	Qtr 3	
3	Johnson	\$14,808.00	USA	Qtr 4	
4	Williams	\$10,644.00	UK	Qtr 2	
5	Jones	\$1,390.00	USA	Qtr 3	
6	Brown	\$4,865.00	USA	Qtr 4	
7	Williams	\$12,438.00	UK	Qtr 1	
8					

Explanation: We fixed the reference to column C by placing a \$ symbol in front of the column letter (\$C2). As a result, cell B2, C2 and cell D2 also contain the formula `=$C2="USA"`, cell A3, B3, C3 and D3 contain the formula `=$C3="USA"`, etc.

Finding Duplicate Values

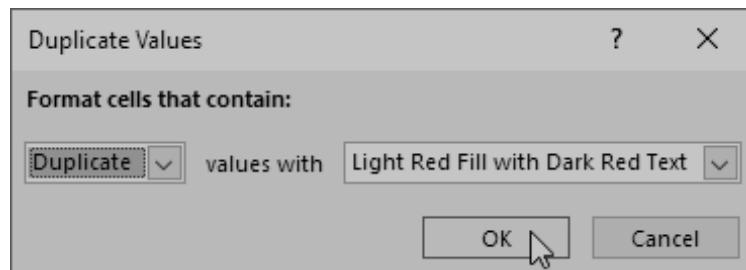
You can highlight the duplicate values in a range of cells using conditional formatting. To implement that, follow these steps:

1. Select the range of cells.

1	4	5	7	99	42	61
16	27	91	87	23	45	87
5	74	2	5	9	10	75
11	62	55	99	23	45	80

2. On the Home tab, go to Styles Group > Conditional Formatting.

3. Select Highlight Cells Rules > Duplicate values.



Result

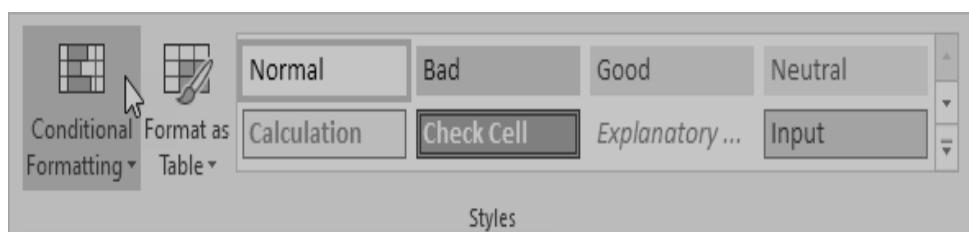
1	4	5	7	99	42	61
16	27	91	87	23	45	87
5	74	2	5	9	10	75
11	62	55	99	23	45	80

Finding Triplicates in Range of Cells

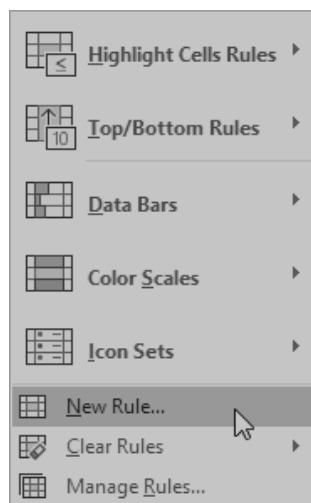
1. Select the range A1:C10.

	A	B	C	D
1	Sierra	Tango	Charlie	
2	Kilo	Bravo	Yankee	
3	Golf	Mike	Delta	
4	Juliet	Alpha	Foxtrot	
5	Papa	X-ray	November	
6	Zulu	Sierra	Whiskey	
7	Romeo	Echo	Quebec	
8	India	Oscar	Delta	
9	Sierra	Lima	Uniform	
10	Hotel	Juliet	Victor	
11				

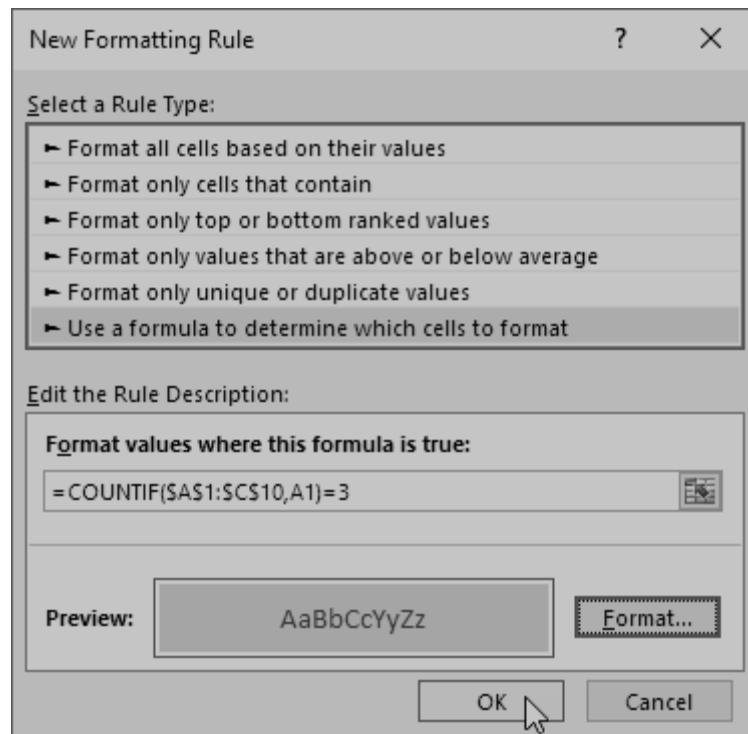
2. On the Home tab, in the Styles group, click Conditional Formatting.



3. Click New Rule.



4. Select 'Use a formula to determine which cells to format'.
5. Enter the formula =COUNTIF(\$A\$1:\$C\$10,A1)=3
6. Select a formatting style and click OK.



Result. Excel highlights the triplicate names.

	A	B	C	D
1	Sierra	Tango	Charlie	
2	Kilo	Bravo	Yankee	
3	Golf	Mike	Delta	
4	Juliet	Alpha	Foxtrot	
5	Papa	X-ray	November	
6	Zulu	Sierra	Whiskey	
7	Romeo	Echo	Quebec	
8	India	Oscar	Delta	
9	Sierra	Lima	Uniform	
10	Hotel	Juliet	Victor	
11				

Explanation:

- =COUNTIF(\$A\$1:\$C\$10,A1) counts the number of names in the range A1:C10 that are equal to the name in cell A1.
- If COUNTIF(\$A\$1:\$C\$10,A1) = 3, Excel formats cell A1. Always write the formula for the upper-left cell in the selected range (A1:C10).
- Excel automatically copies the formula to the other cells. Thus, cell A2 contains the formula =COUNTIF(\$A\$1:\$C\$10,A2)=3, cell A3 =COUNTIF(\$A\$1:\$C\$10,A3)=3, etc.
- Notice how we created an absolute reference (\$A\$1:\$C\$10) to fix this reference.

Note: you can use any formula you like. For example, use this formula =COUNTIF(\$A\$1:\$C\$10,A1)>3 to highlight names that occur more than 3 times.

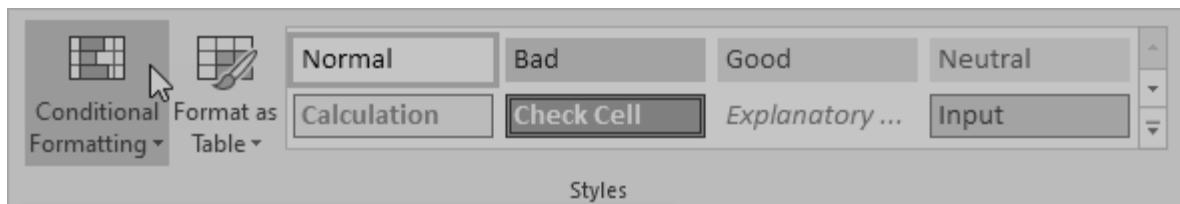
Finding Duplicate Rows

To find and highlight duplicate rows in Excel, use COUNTIFS (with the letter S at the end) instead of COUNTIF.

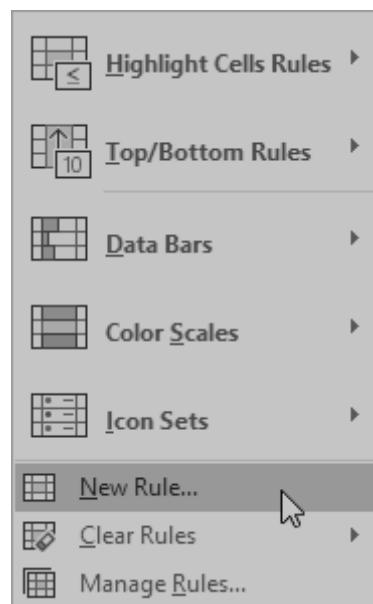
1. Select the range A1:C10.

	A	B	C	D
1	Leopard	Africa	Zambia	
2	Lion	Africa	South Africa	
3	Elephant	Asia	Thailand	
4	Leopard	Asia	India	
5	Rhino	Africa	South Africa	
6	Buffalo	Asia	Cambodia	
7	Lion	Africa	South Africa	
8	Rhino	Asia	Nepal	
9	Buffalo	Africa	South Africa	
10	Elephant	Africa	Botswana	
11				

2. On the Home tab, in the Styles group, click Conditional Formatting.



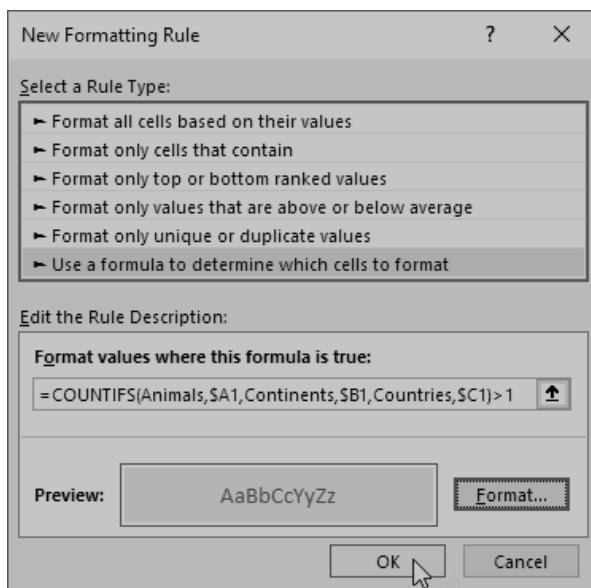
3. Click New Rule.



4. Select 'Use a formula to determine which cells to format'.

5. Enter the formula =COUNTIFS(Animals,\$A1,Continents,\$B1,Countries,\$C1)>1

6. Select a formatting style and click OK.



The named range Animals refers to the range A1:A10, the named range Continents refers to the range B1:B10 and the named range Countries refers to the range C1:C10. $=COUNTIFS(Animals,$A1,Continents,$B1,Countries,$C1)$ counts the number of rows based on multiple criteria (Leopard, Africa, Zambia).

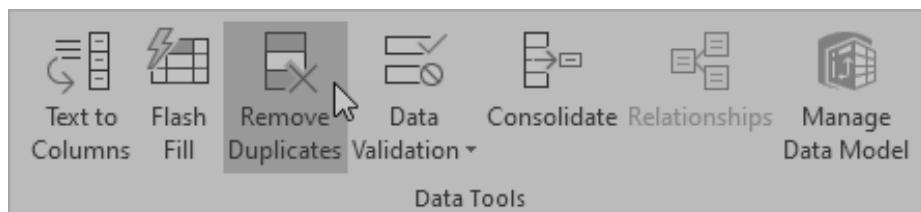
Result. Excel highlights the duplicate rows.

	A	B	C	D
1	Leopard	Africa	Zambia	
2	Lion	Africa	South Africa	
3	Elephant	Asia	Thailand	
4	Leopard	Asia	India	
5	Rhino	Africa	South Africa	
6	Buffalo	Asia	Cambodia	
7	Lion	Africa	South Africa	
8	Rhino	Asia	Nepal	
9	Buffalo	Africa	South Africa	
10	Elephant	Africa	Botswana	
11				

Explanation:

- If $COUNTIFS(Animals,$A1,Continents,$B1,Countries,$C1) > 1$, in other words, if there are multiple (Leopard, Africa, Zambia) rows, Excel formats cell A1.
- Always write the formula for the upper-left cell in the selected range (A1:C10). Excel automatically copies the formula to the other cells.
- We fixed the reference to each column by placing a \$ symbol in front of the column letter (\$A1, \$B1 and \$C1). As a result, cell A1, B1 and C1 contain the same formula, cell A2, B2 and C2 contain the formula $=COUNTIFS(Animals,$A2,Continents,$B2,Countries,$C2)>1$, etc.

7. Finally, you can use the Remove Duplicates tool in Excel to quickly remove duplicate rows. On the Data tab, in the Data Tools group, click Remove Duplicates.



In the example below, Excel removes all identical rows (blue) except for the first identical row found (yellow).

	A	B	C	D
1	Last Name	Sales	Country	Quarter
2	Smith	\$16,753.00	UK	Qtr 3
3	Johnson	\$14,808.00	USA	Qtr 4
4	Williams	\$10,644.00	UK	Qtr 2
5	Jones	\$1,390.00	USA	Qtr 3
6	Brown	\$4,865.00	USA	Qtr 4
7	Smith	\$16,753.00	UK	Qtr 3
8	Williams	\$12,438.00	UK	Qtr 1
9	Johnson	\$9,339.00	UK	Qtr 2
10	Smith	\$18,919.00	USA	Qtr 3
11	Jones	\$9,213.00	USA	Qtr 4
12	Jones	\$7,433.00	UK	Qtr 1
13	Smith	\$16,753.00	UK	Qtr 3
14	Brown	\$3,255.00	USA	Qtr 2
15	Williams	\$14,867.00	USA	Qtr 3
16	Williams	\$19,302.00	UK	Qtr 4
17	Smith	\$9,698.00	USA	Qtr 1
18				

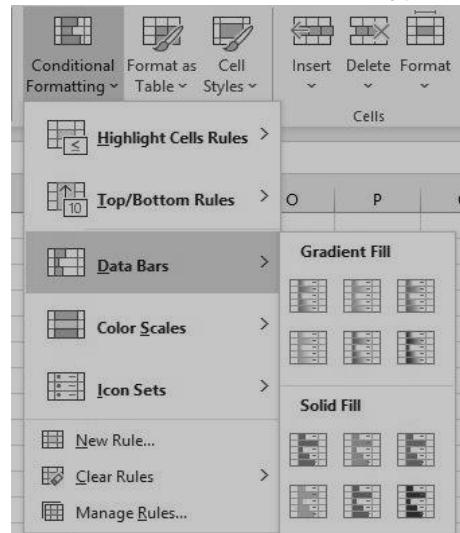
	A	B	C	D
1	Last Name	Sales	Country	Quarter
2	Smith	\$16,753.00	UK	Qtr 3
3	Johnson	\$14,808.00	USA	Qtr 4
4	Williams	\$10,644.00	UK	Qtr 2
5	Jones	\$1,390.00	USA	Qtr 3
6	Brown	\$4,865.00	USA	Qtr 4
7	Williams	\$12,438.00	UK	Qtr 1
8	Johnson	\$9,339.00	UK	Qtr 2
9	Smith	\$18,919.00	USA	Qtr 3
10	Jones	\$9,213.00	USA	Qtr 4
11	Jones	\$7,433.00	UK	Qtr 1
12	Brown	\$3,255.00	USA	Qtr 2
13	Williams	\$14,867.00	USA	Qtr 3
14	Williams	\$19,302.00	UK	Qtr 4
15	Smith	\$9,698.00	USA	Qtr 1
16				
17				
18				

Data Bars

Data bars in Excel are used to visualize the range of cells. The longer bar represents a higher value.

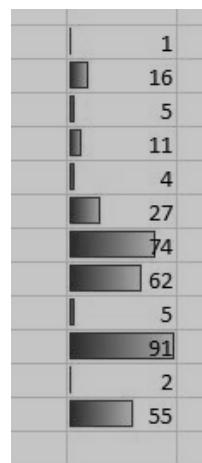
To add the data bars, follow these steps:

- Select the range of cells.
- On the Home tab, go to Conditional Formatting > Data Bars and select a subtype.



The screenshot shows the Microsoft Excel ribbon with the 'Conditional Formatting' tab selected. A dropdown menu is open, showing various rules: 'Highlight Cells Rules', 'Top/Bottom Rules', 'Data Bars', 'Color Scales', 'Icon Sets', 'New Rule...', 'Clear Rules', and 'Manage Rules...'. The 'Data Bars' option is highlighted. To the right of the dropdown, there are sections for 'Gradient Fill' and 'Solid Fill' with their respective color swatches.

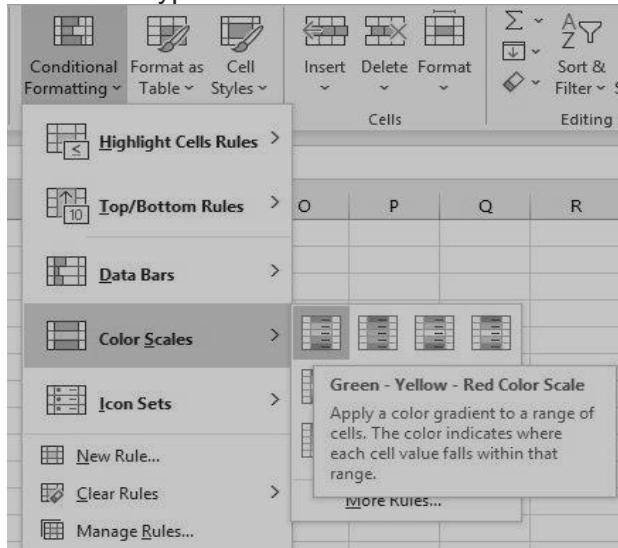
Result:



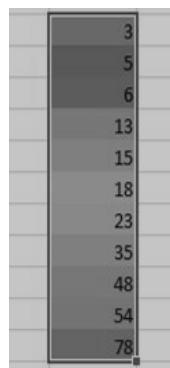
Color Scales

Color Scales in Excel make the visualization of values in a range of cells very easy. To add a color scale, follow these steps:

- Select the range of cells.
- On the Home tab, go to Styles Group > Conditional Formatting.
- Click Color Scales and select a subtype.



Result:

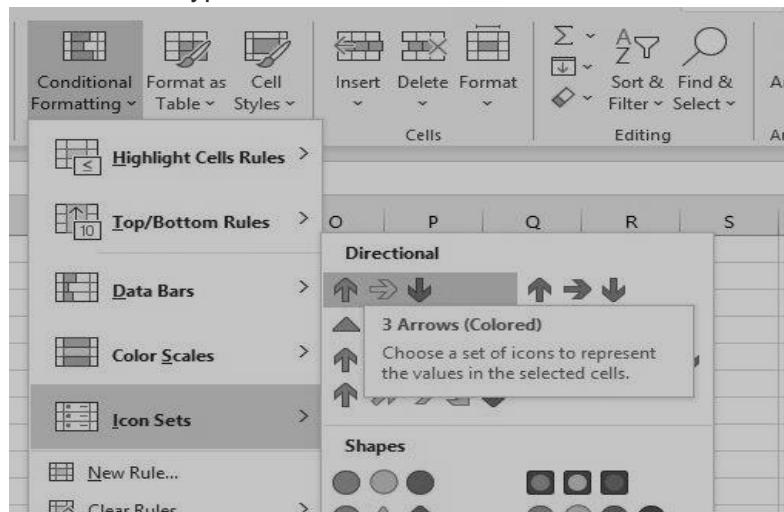


- The red color represents the minimum value in the range
- The yellow color represents the median value
- The green color represents the maximum value
- All the other values are colored proportionally

Icon Sets

Excel Conditional Formatting icon sets are used to visualize the data with the help of shapes, arrows, check marks, and other objects. To add an icon sets, follow these steps:

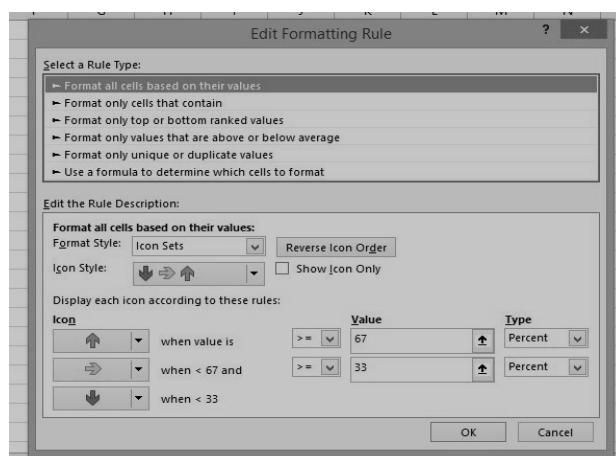
- Select the range of cells.
- On the Home tab, go to Styles Group > Conditional Formatting.
- Click Icon Sets and select a subtype.



Result:

		3
		6
		13
		15
		18
		23
	→	35
	→	48
	↑	54
	↑	78

- To update the rules, go to Conditional Formatting > Manage Rules > Edit rules. You can change the rules according to your preferences.



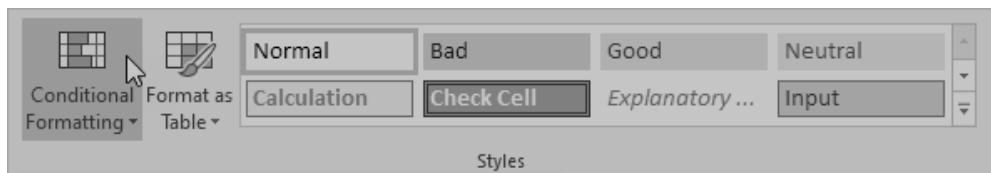
Manage Rules

To view all conditional formatting rules in a workbook, use the Conditional Formatting Rules Manager. You can also use this screen to create, edit and delete rules.

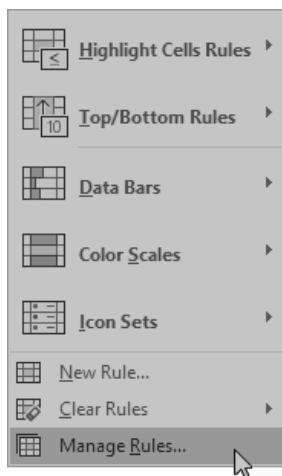
1. Select cell A1.

	A	B
1	81	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

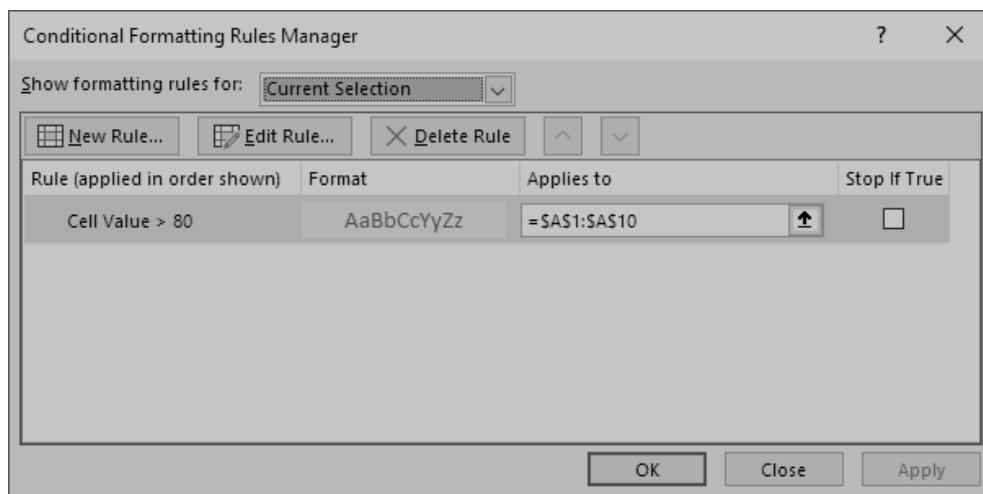
2. On the Home tab, in the Styles group, click Conditional Formatting.



3. Click Manage Rules.

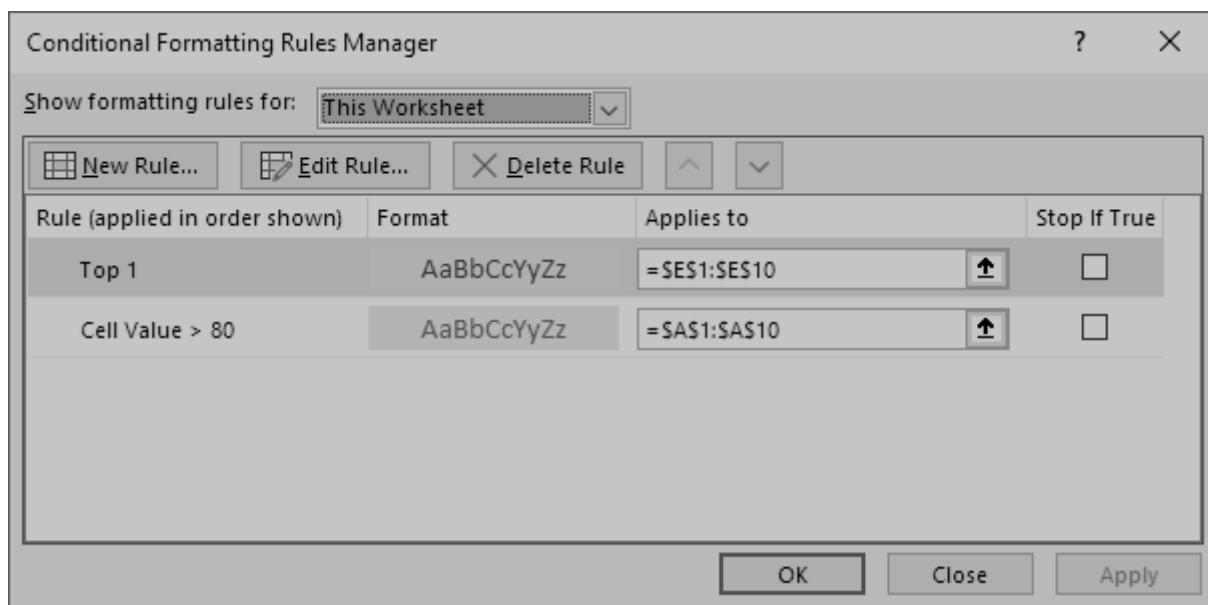


The Conditional Formatting Rules Manager appears.



Note: because we selected cell A1, Excel shows the rule applied to the range A1:A10.

- From the drop-down list, change Current Selection to This Worksheet, to view all conditional formatting rules in this worksheet.

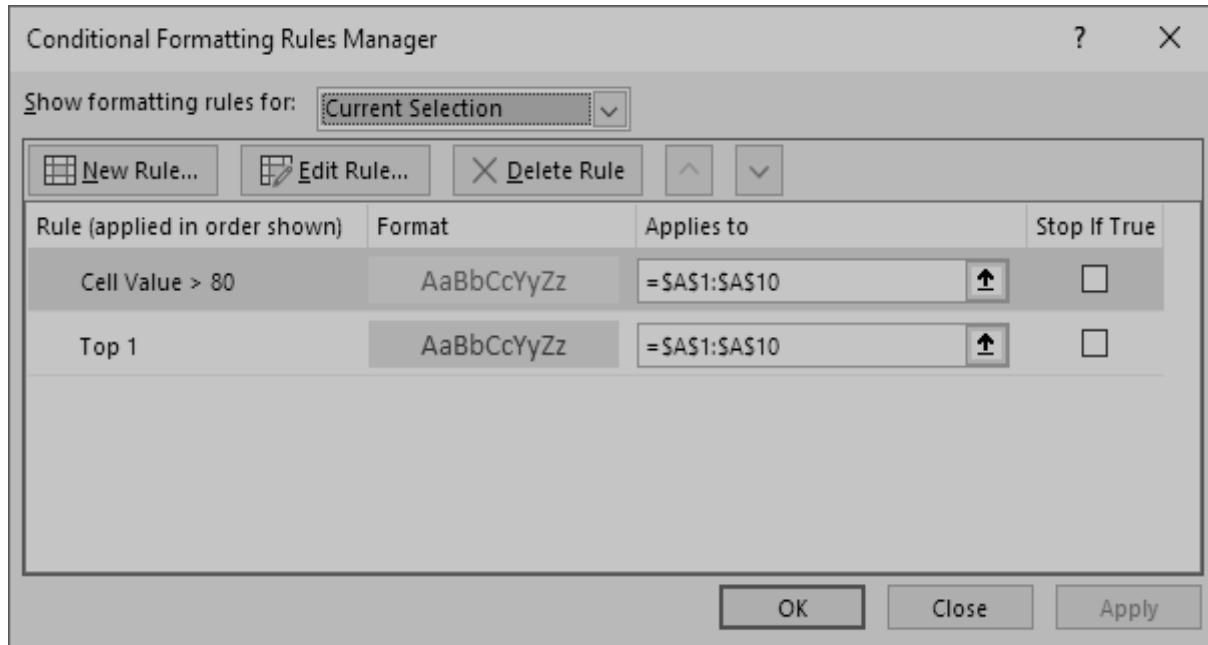


Note: click New Rule, Edit Rule and Delete Rule to create, edit and delete rules.

Conflicting Rule

Sometimes multiple conditional formatting rules in Excel conflict. A higher rule always wins. This example illustrates two different results.

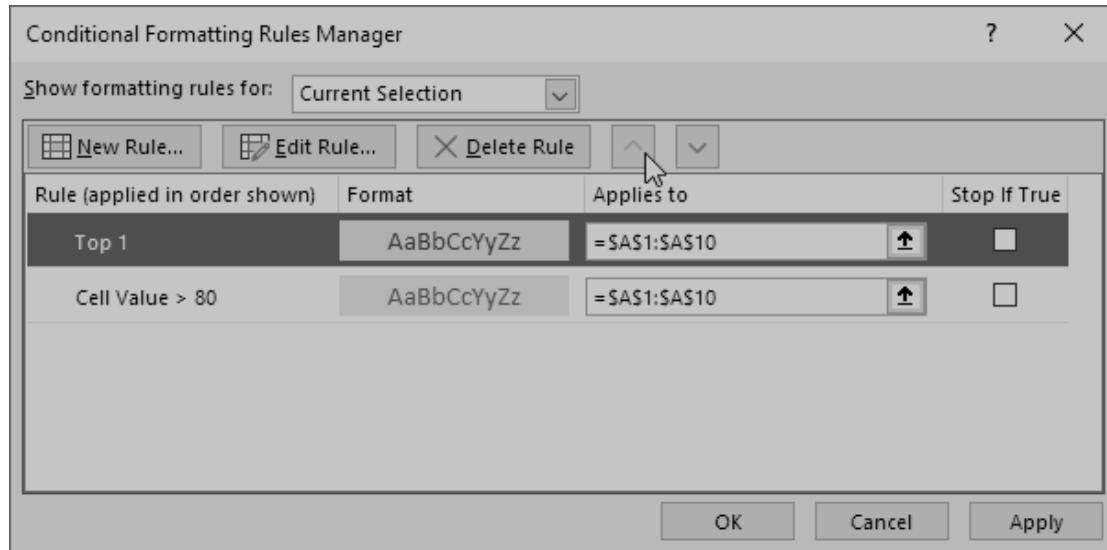
1. The value 95 is higher than 80 but is also the highest value (Top 1). The formats (yellow fill vs green fill and yellow text color vs green text color) conflict. A higher rule always wins. As a result, the value 95 is colored yellow.



Result:

	A	B
1	81	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

2. Move the second rule up. The value 95 is the highest value (Top 1) but is also higher than 80. The formats (green fill vs yellow fill and green text color vs yellow text color) conflict. A higher rule always wins. As a result, the value 95 is colored green.



Result:

	A	B
1	81	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

Note: only use the Stop If True check boxes for backwards compatibility with earlier versions of Microsoft Excel.

2.3 Analysing Data with Pivot Tables

What is Pivot Table?

- Pivot tables are one of Excel's most powerful features. A pivot table allows you to extract the significance from a large, detailed data set.
- It helps prepare reports and get insights from huge data sets with just a few clicks.
- It can help summarize data, reorganize data, group, sort and filter data quickly.

Pivot Table Source Data Format

- Before inserting a Pivot table, you need to ensure that the source data is in a proper format.
- The data set should not have any empty rows/ columns.
- All the columns in the data set should have headers.
- There are no totals/ subtotals in between.

What Should Pivot Table Source Data Should Look Like?

✓ Correct					✗ Incorrect						
Company	Region	Month	Product	Sales \$	ABC Co.	EMEA Region	Product	Jan	Feb	Mar	Apr
ABC	EMEA	Jan	Product 1	1,000			Product 1	1,000	2,000	3,000	4,000
ABC	EMEA	Jan	Product 2	1,010			Product 2	1,010	2,010	3,010	4,010
ABC	EMEA	Jan	Product 3	1,020			Product 3	1,020	2,020	3,020	4,020
ABC	EMEA	Jan	Product 4	1,030			Product 4	1,030	2,030	3,030	4,030
ABC	EMEA	Feb	Product 1	2,000			Total	4,060	8,060	12,060	16,060
ABC	EMEA	Feb	Product 2	2,010							
ABC	EMEA	Feb	Product 3	2,020							
ABC	EMEA	Feb	Product 4	2,030							
ABC	EMEA	Mar	Product 1	3,000							
ABC	EMEA	Mar	Product 2	3,010							
ABC	EMEA	Mar	Product 3	3,020							
ABC	EMEA	Mar	Product 4	3,030							
ABC	EMEA	Apr	Product 1	4,000							
ABC	EMEA	Apr	Product 2	4,010							
ABC	EMEA	Apr	Product 3	4,020							
ABC	EMEA	Apr	Product 4	4,030							

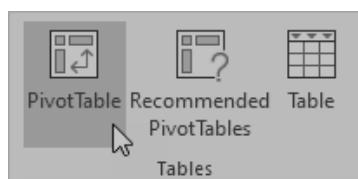
How to insert a Pivot Table?

Our data set consists of 213 records and 6 fields. Order ID, Product, Category, Amount, Date and Country.

	A	B	C	D	E	F	G	H
1	Order ID	Product	Category	Amount	Date	Country		
2	1	Carrots	Vegetables	\$4,270	1/6/2016	United States		
3	2	Broccoli	Vegetables	\$8,239	1/7/2016	United Kingdom		
4	3	Banana	Fruit	\$617	1/8/2016	United States		
5	4	Banana	Fruit	\$8,384	1/10/2016	Canada		
6	5	Beans	Vegetables	\$2,626	1/10/2016	Germany		
7	6	Orange	Fruit	\$3,610	1/11/2016	United States		
8	7	Broccoli	Vegetables	\$9,062	1/11/2016	Australia		
9	8	Banana	Fruit	\$6,906	1/16/2016	New Zealand		
10	9	Apple	Fruit	\$2,417	1/16/2016	France		
11	10	Apple	Fruit	\$7,421	1/16/2016	Canada		

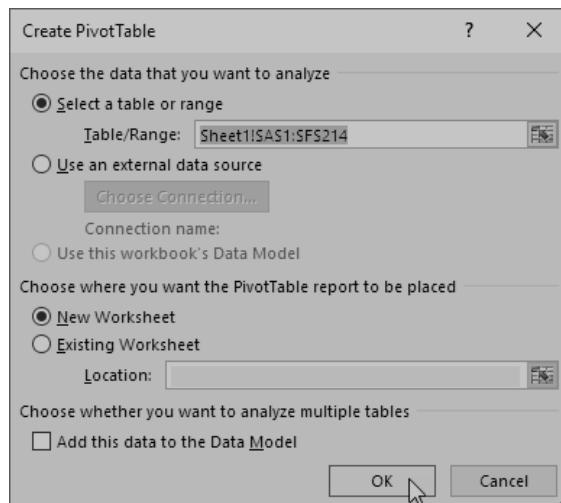
To insert a pivot table, execute the following steps.

1. Click any single cell inside the data set.
2. On the Insert tab, in the Tables group, click PivotTable.



- The following dialog box appears.
- Excel automatically selects the data for you.
- The default location for a new pivot table is New Worksheet.

3. Click OK.



- Once you click on OK. You get a Pivot table set up in another sheet and a Pivot Table Fields.

Drag Fields

The PivotTable Fields pane appears. To get the total amount exported of each product, drag the following fields to the different areas.

1. Product field to the Rows area.
2. Amount field to the Values area.
3. Country field to the Filters area.

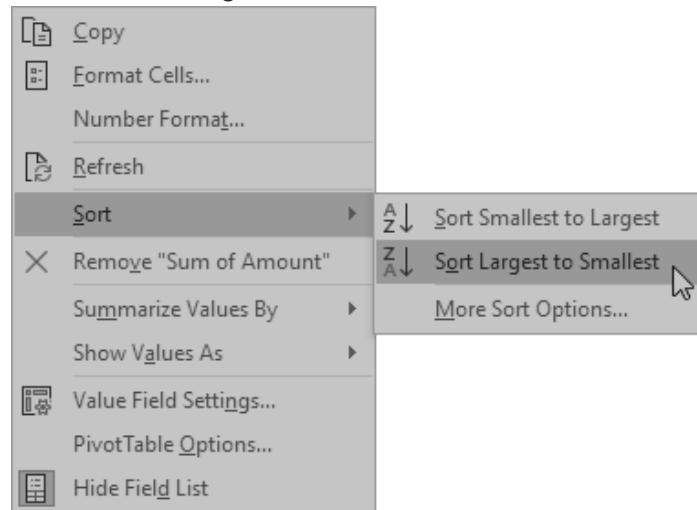
Below you can find the pivot table. Bananas are our main export product. That's how easy pivot tables can be!

	A	B	C
1	Country	(All)	
2			
3	Row Labels	Sum of Amount	
4	Apple	191257	
5	Banana	340295	
6	Beans	57281	
7	Broccoli	142439	
8	Carrots	136945	
9	Mango	57079	
10	Orange	104438	
11	Grand Total	1029734	
12			

Sort

To get Banana at the top of the list, sort the pivot table.

1. Click any cell inside the Sum of Amount column.
2. Right click and click on Sort, Sort Largest to Smallest.

**Result:**

	A	B	C
1	Country	(All)	
2			
3	Row Labels	Sum of Amount	
4	Banana	340295	
5	Apple	191257	
6	Broccoli	142439	
7	Carrots	136945	
8	Orange	104438	
9	Beans	57281	
10	Mango	57079	
11	Grand Total	1029734	
12			

Filter

Because we added the Country field to the Filters area, we can filter this pivot table by Country. For example, which products do we export the most to France?

1. Click the filter drop-down and select France.

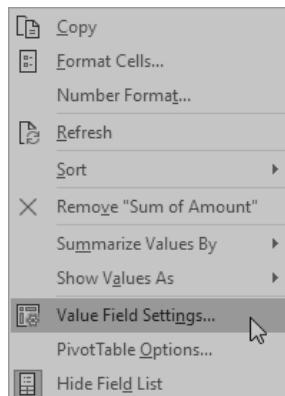
Result: Apples are our main export product to France.

	A	B	C
1	Country	France	
2			
3	Row Labels		Sum of Amount
4	Apple	80193	
5	Banana	36094	
6	Carrots	9104	
7	Mango	7388	
8	Broccoli	5341	
9	Orange	2256	
10	Beans	680	
11	Grand Total		141056
12			

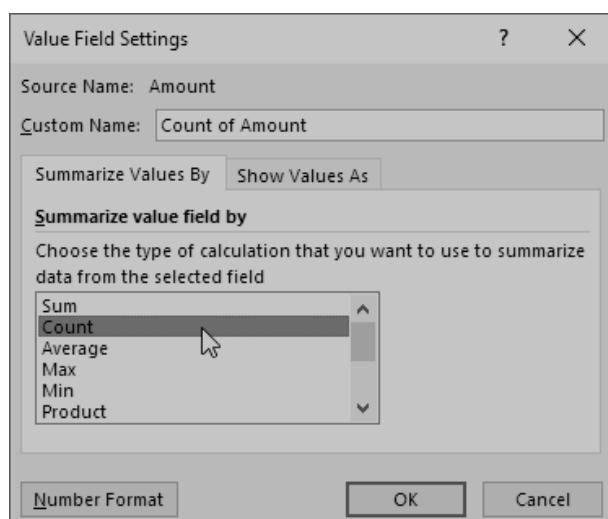
Note: you can use the standard filter (triangle next to Row Labels) to only show the amounts of specific products.

Change Summary Calculation

- By default, Excel summarizes your data by either summing or counting the items.
 - To change the type of calculation that you want to use, execute the following steps.
1. Click any cell inside the Sum of Amount column.
 2. Right click and click on Value Field Settings.



3. Choose the type of calculation you want to use. For example, click Count.



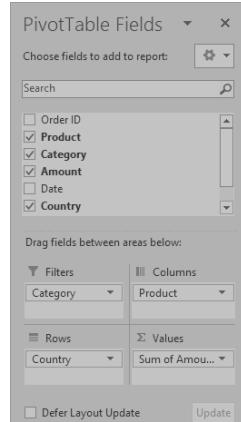
4. Click OK.

Result: 16 out of the 28 orders to France were 'Apple' orders.

A	B	C
1 Country	France	
2		
3 Row Labels	Count of Amount	
4 Apple	16	
5 Banana	7	
6 Carrots	1	
7 Mango	1	
8 Orange	1	
9 Beans	1	
10 Broccoli	1	
11 Grand Total	28	
12		

Two-dimensional Pivot Table

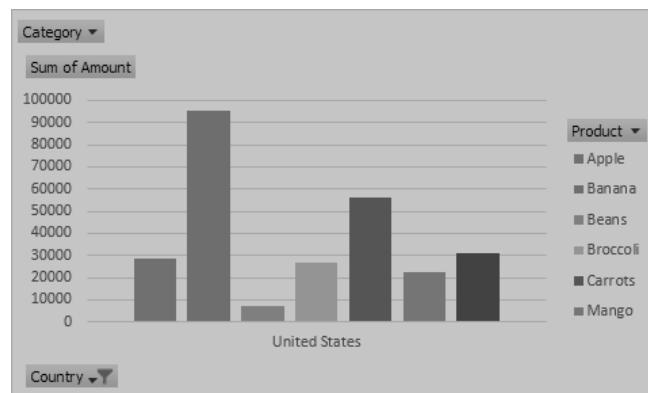
- If you drag a field to the Rows area and Columns area, you can create a two-dimensional pivot table.
- First, insert a pivot table.
- Next, to get the total amount exported to each country, of each product, drag the following fields to the different areas.
 - Country field to the Rows area.
 - Product field to the Columns area.
 - Amount field to the Values area.
 - Category field to the Filters area.



Below you can find the two-dimensional pivot table.

	A	B	C	D	E	F	G	H	I	J
1	Category	(All)								
2										
3	Sum of Amount	Column								
4	Row Labels	Apple	Banana	Beans	Broccoli	Carrots	Mango	Orange	Grand Total	
5	Australia	20634	52721	14433	17953	8106	9186	8680	131713	
6	Canada	24867	33775		12407		3767	19929	94745	
7	France	80193	36094	680	5341	9104	7388	2256	141056	
8	Germany	9082	39686	29905	37197	21636	8775	8887	155168	
9	New Zealand	10332	40050		4390			12010	66782	
10	United Kingdom	17534	42908	5100	38436	41815	5600	21744	173137	
11	United States	28615	95061	7163	26715	56284	22363	30932	267133	
12	Grand Total	191257	340295	57281	142439	136945	57079	104438	1029734	
13										

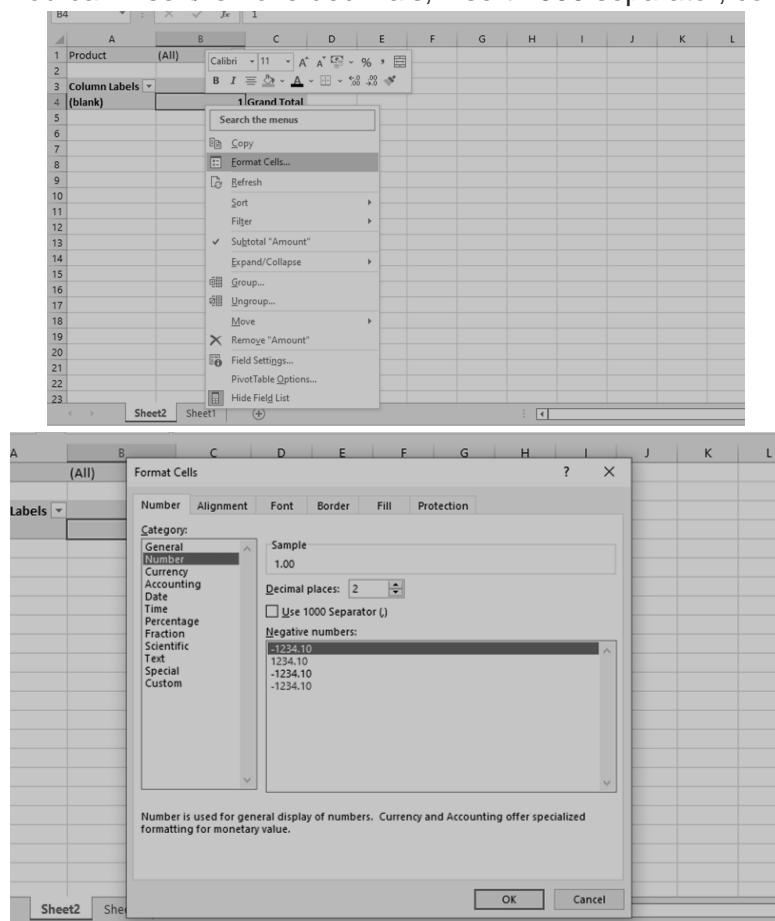
To easily compare these numbers, create a pivot chart and apply a filter. Maybe this is one step too far for you at this stage, but it shows you one of the many other powerful pivot table features Excel has to offer.



Pivot Table Report Formatting

Number Formatting

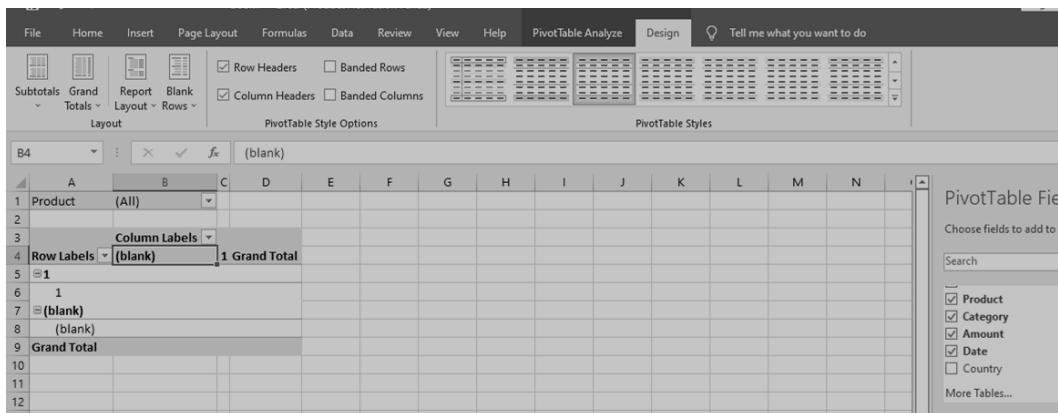
- To change the number format, Right-click on the pivot table Select Number Format and choose the format you want. You can insert/remove decimals, insert 1000 separator, convert to percent etc.



Pivot Table Design

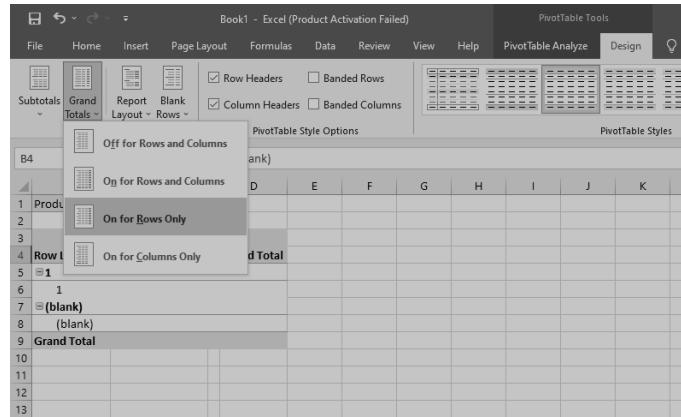
To change the design of the table,

- Click on the Pivot table
- Go to the design tab
- Select the design from multiple options available



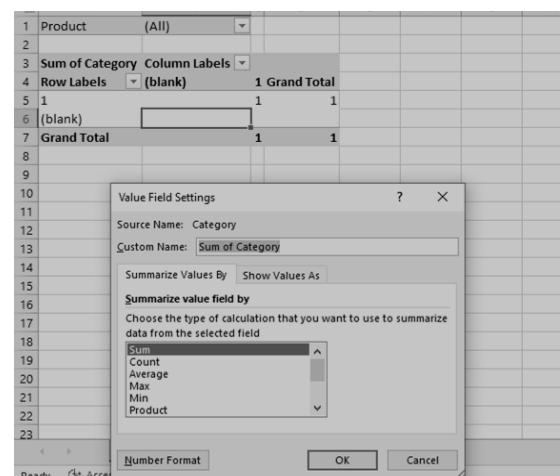
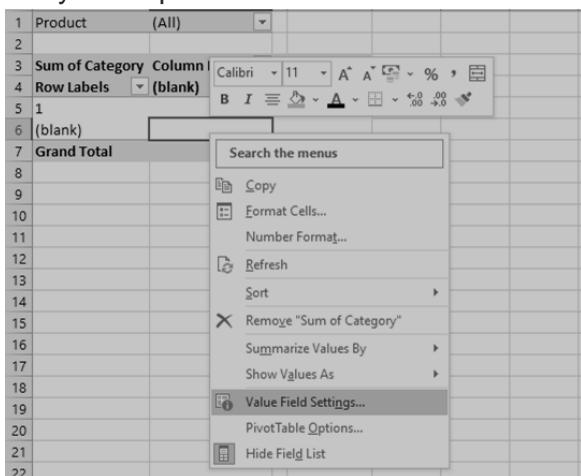
Grand Totals

- The default Pivot table has Grand Totals both in rows and in columns.
- We have the option to remove either or both.
- To add or remove Grand Totals Click on the Pivot Table so that the **Design** Tab gets activated.
- Click on the Design tab, then Click on **Grand Totals** and select the relevant option for your report.



Value Field Settings

- The default value in the Pivot table is Sum of total sales made by each salesperson by region.
- In case we want to change it to average sales, just right click on the Pivot Table and select **Summarize Values** by option.
- You get a range of options- **Sum, Count, Min, Max, Average** etc. You can select the option based on your requirement.



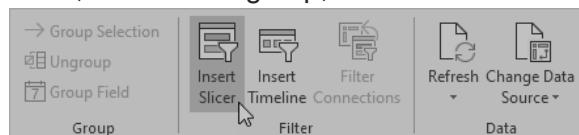
Slicer

- Use slicers in Excel to quickly and easily filter pivot tables.
- Connect multiple slicers to multiple pivot tables to create awesome reports.
- Below you can find a pivot table. Go back to Pivot Tables to learn how to create this pivot table.

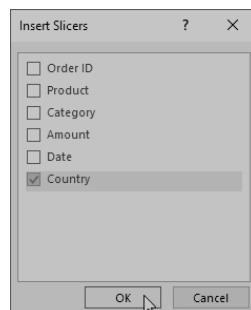
	A	B	C
1	Country	(All)	▼
2			
3	Row Labels	Sum of Amount	
4	Apple	191257	
5	Banana	340295	
6	Beans	57281	
7	Broccoli	142439	
8	Carrots	136945	
9	Mango	57079	
10	Orange	104438	
11	Grand Total	1029734	
12			

- To insert a slicer, execute the following steps.

1. Click any cell inside the pivot table.
2. On the PivotTable Analyze tab, in the Filter group, click Insert Slicer.



3. Check Country and click OK.



4. Click United States to find out which products we export the most to the United States.

	A	B	C	D	E	F
1	Country	United States	▼			
2						
3	Row Labels	Sum of Amount				
4	Apple	28615				
5	Banana	95061				
6	Beans	7163				
7	Broccoli	26715				
8	Carrots	56284				
9	Mango	22363				
10	Orange	30932				
11	Grand Total	267133				
12						
13						
14						
15						

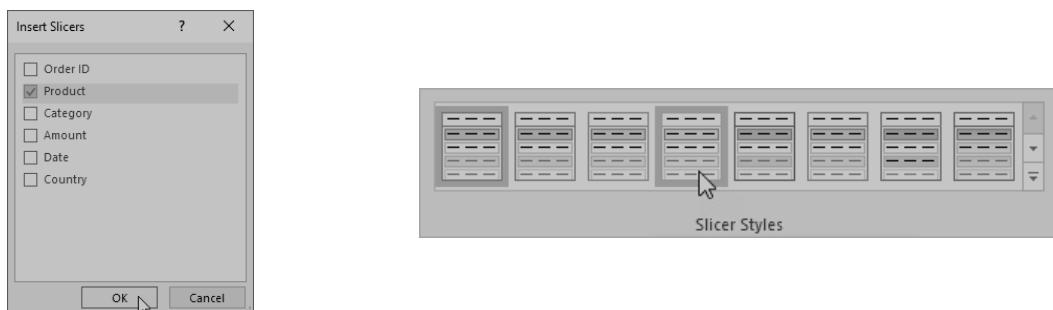
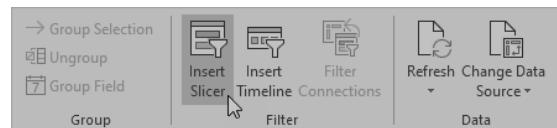
Country

- Australia
- Canada
- France
- Germany
- New Zealand
- United Kingdom
- United States**

Conclusion: Bananas are our main export product to the United States. The report filter (cell B1) changes to United States.

Let's insert a second slicer.

5. Click any cell inside the pivot table.
6. On the PivotTable Analyze tab, in the Filter group, click Insert Slicer.
7. Check Product and click OK.
8. Select the slicer.
9. On the Slicer tab, in the Slicer Styles group, click a slicer style.



10. Use the second slicer. Click the Multi-Select button to select multiple products.

	A	B	C	D	E	F
1	Country	United States				
2						
3	Row Labels	Sum of Amount				
4	Banana	95061				
5	Beans	7163				
6	Broccoli	26715				
7	Grand Total	128939				
8						
9	Country	☰				
10	Australia					
11	Canada					
12	France					
13	Germany					
14	New Zealand					
15	United Kingdom					
16	United States					
17						
18						
19						
20						
21						
22						
23						

Note: instead of using the Multi-Select button, hold down CTRL to select multiple items.

To really impress your client, execute the following steps.

11. Insert a second pivot table.

To connect both slicers to this pivot table, execute the following steps:

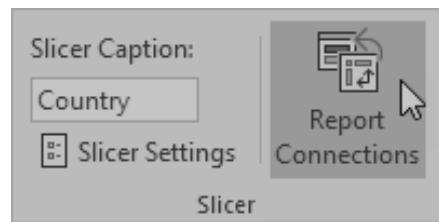
12. Select the first slicer.

13. On the Slicer tab, in the Slicer group, click Report Connections.

14. Select the second pivot table and click OK.

15. Repeat steps 12-14 for the second slicer.

16. Use both slicers.



Report Connections (Country)

Name	Sheet
PivotTable1	Sheet2
PivotTable2	Sheet2

Data Table:

	A	B	C	D	E	F
1	Country	Canada	Country	Canada		
2						
3	Row Labels	Sum of Amount	Row Labels	Count of Amount		
4	Apple	24867	Apple	6		
5	Orange	19929	Orange	3		
6	Grand Total	44796	Grand Total	9		
7						
8	Country		Product			
9	Australia		Apple			
10	Canada		Banana			
11	France		Broccoli			
12	Germany		Mango			
13	New Zealand		Orange			
14	United Kingdom		Beans			
15	United States		Carrots			
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						

Conclusion: The total amount of apples exported to Canada equals \$24,867 (6 orders) and the total amount of oranges exported to Canada equals \$19,929 (3 orders).

17. Click the icon in the upper-right corner of a slicer to clear the filter.

	A	B	C	D	E	F
1	Country	Canada	Country	Canada		
2						
3	Row Labels	Sum of Amount	Row Labels	Count of Amount		
4	Apple	24867	Apple	6		
5	Banana	33775	Banana	7		
6	Broccoli	12407	Broccoli	3		
7	Mango	3767	Mango	1		
8	Orange	19929	Orange	3		
9	Grand Total	94745	Grand Total	20		
10						
11	Country		Product			
12	Australia		Apple			
13	Canada		Banana			
14	France		Broccoli			
15	Germany		Mango			
16	New Zealand		Orange			
17	United Kingdom		Beans			
18	United States		Carrots			
19						
20						
21						
22						
23						
24						
25						

Note: We didn't export any beans or carrots to Canada.

Data Refresh

- In case there is any change in the Pivot Table source data, the Pivot table does not get updated automatically.
- Right-click on the table and then click on **Refresh** so that the changes in the data get reflected in the table.

A screenshot of a Microsoft Excel spreadsheet showing a PivotTable. The PivotTable has 'Product' in the Row Labels and '(All)' in the Column Labels. The values are 'Sum of Category'. A context menu is open over the PivotTable area, with 'Refresh' highlighted. Other options in the menu include Copy, Format Cells..., Number Format..., Sort, Remove 'Sum of Category', Summarize Values By, Show Values As, Value Field Settings..., PivotTable Options..., and Hide Field List.

Pivot Table Charts

- A pivot chart is the visual representation of a pivot table in Excel. Pivot charts and pivot tables are connected with each other.
- Below you can find a two-dimensional pivot table. Go back to Pivot Tables to learn how to create this pivot table.

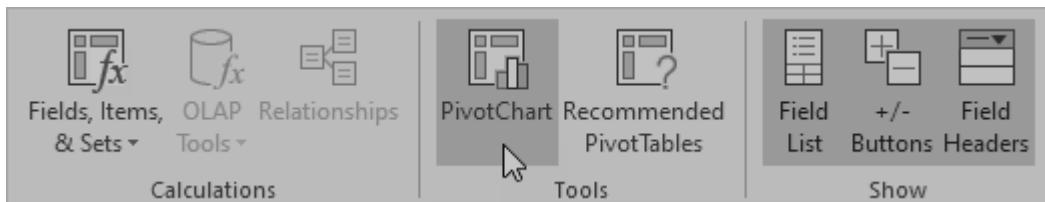
A screenshot of a Microsoft Excel PivotTable. The columns are labeled 'Category' (All), 'Column', 'Apple', 'Banana', 'Beans', 'Broccoli', 'Carrots', 'Mango', 'Orange', and 'Grand Total'. The rows are labeled 'Row Labels' and 'Country'. The data shows the following values:

Country	Apple	Banana	Beans	Broccoli	Carrots	Mango	Orange	Grand Total
Australia	20634	52721	14433	17953	8106	9186	8680	131713
Canada	24867	33775		12407		3767	19929	94745
France	80193	36094	680	5341	9104	7388	2256	141056
Germany	9082	39686	29905	37197	21636	8775	8887	155168
New Zealand	10332	40050		4390			12010	66782
United Kingdom	17534	42908	5100	38436	41815	5600	21744	173137
United States	28615	95061	7163	26715	56284	22363	30932	267133
Grand Total	191257	340295	57281	142439	136945	57079	104438	1029734

Insert Pivot Chart

To insert a pivot chart, execute the following steps.

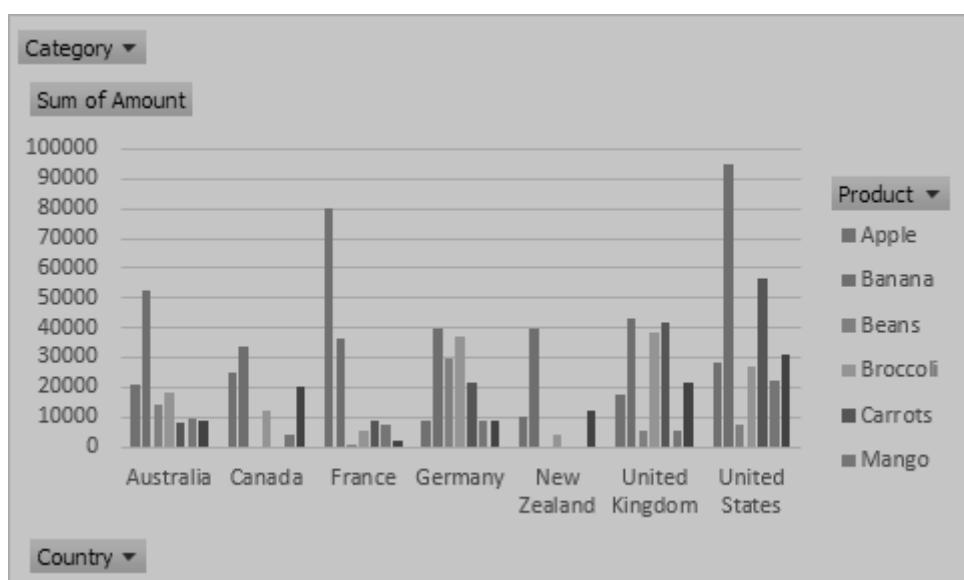
1. Click any cell inside the pivot table.
2. On the PivotTable Analyze tab, in the Tools group, click PivotChart.



The Insert Chart dialog box appears.

3. Click OK.

Below you can find the pivot chart.

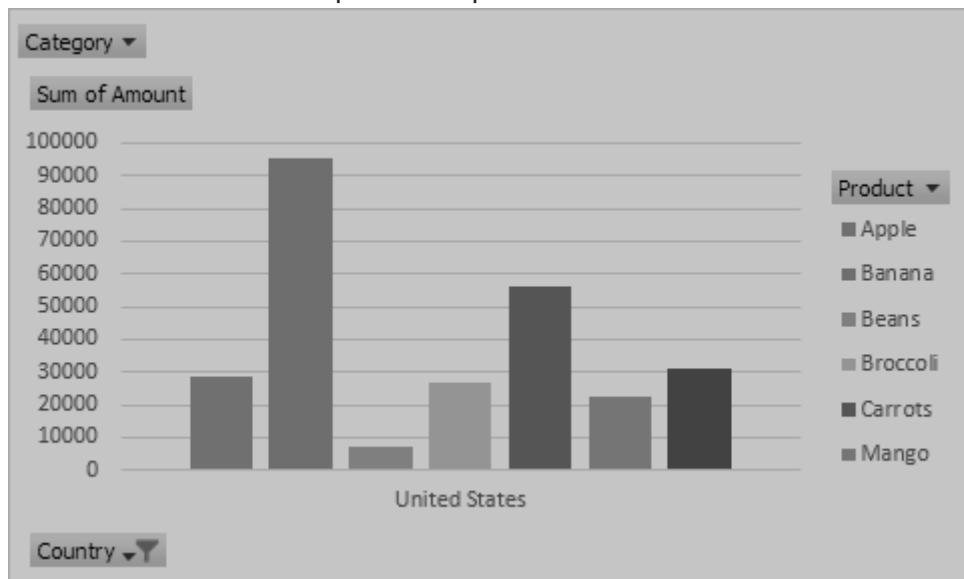


Note: Any changes you make to the pivot chart are immediately reflected in the pivot table and vice versa.

Filter Pivot Chart

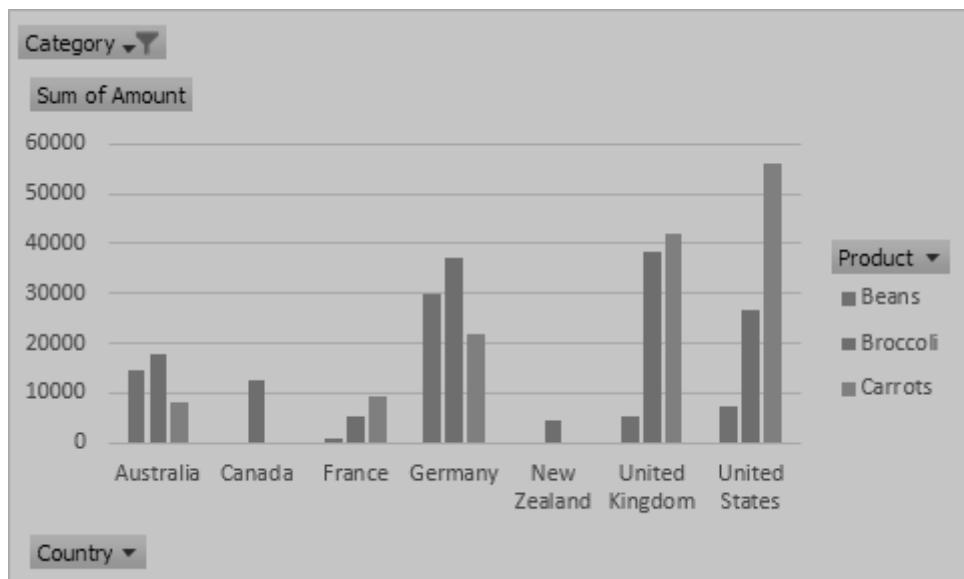
To filter this pivot chart, execute the following steps.

1. Use the standard filters (triangles next to Product and Country). For example, use the Country filter to only show the total amount of each product exported to the United States.



2. Remove the Country filter.

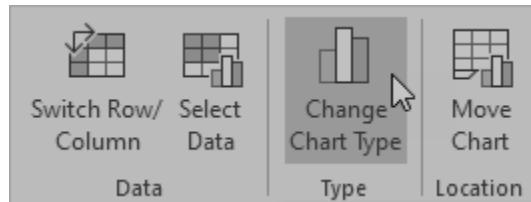
3. Because we added the Category field to the Filters area, we can filter this pivot chart (and pivot table) by Category. For example, use the Category filter to only show the vegetables exported to each country.



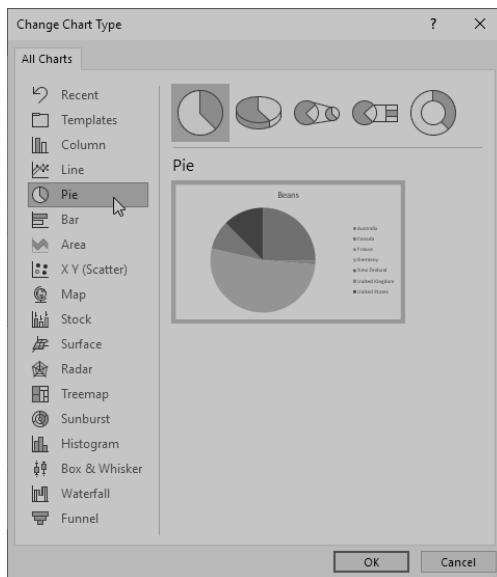
Change Pivot Chart Type

You can change to a different type of pivot chart at any time.

1. Select the chart.
2. On the Design tab, in the Type group, click Change Chart Type.

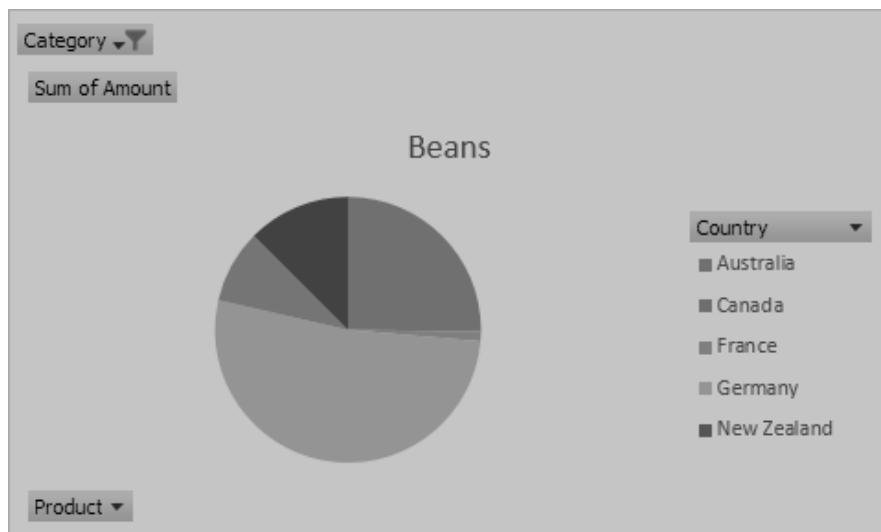


3. Choose Pie.



4. Click OK.

Result:

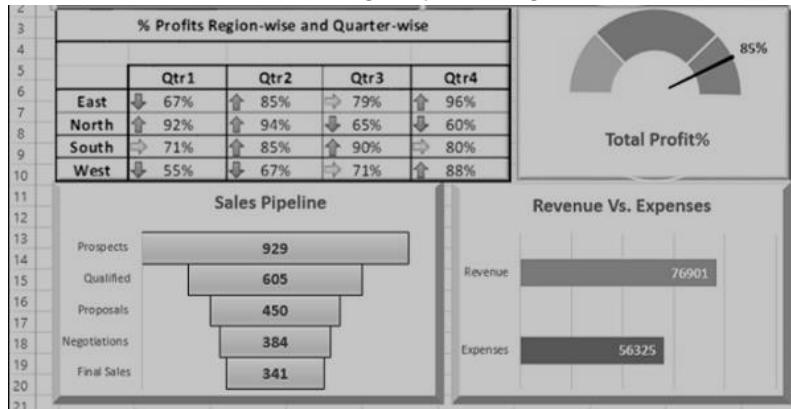


Note: pie charts always use one data series (in this case, Beans). To get a pivot chart of a country, swap the data over the axis. First, select the chart. Next, on the Design tab, in the Data group, click Switch Row/Column.

2.4 Dashboarding

What is an Excel Dashboard?

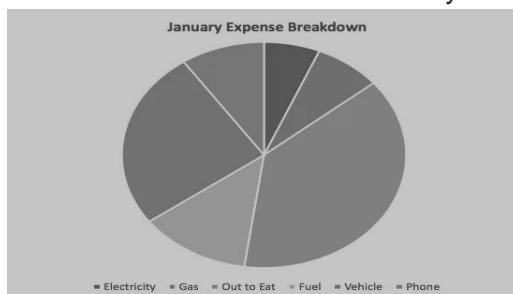
- The simplest way to think of a dashboard is as a visual representation of data.
- Raw data can be hard to look at. Sure, the need-to-know digits are there. But, all of those rows and columns are often impossible to process and understand.
- This is where dashboards come into play. They turn data into information by creating different charts, tables, and other visual elements that give you a high-level overview of that data.



- A dashboard simplifies that otherwise complex data you have in your spreadsheet and transforms it into something visual that's far easier for you to grasp and, thus, utilize.
- Needless to say, dashboards have a wide array of uses—from budgeting or project management to marketing or sales reporting.
- For a simple example, we have used a dashboard to transform this spreadsheet of first quarter expenses:

Item	Cost	Budgeted	Month	Quarter	Year	Expense Type	Credit Card used?	ETF?	Cash?
Electricity	50	55	January	Q1	2018	Variable	N	Y	N
Gas	60	75	January	Q1	2018	Variable	N	Y	N
Out to Eat	300	250	January	Q1	2018	Variable	Y	N	N
Fuel	100	100	January	Q1	2018	Variable	Y	N	N
Vehicle	200	200	January	Q1	2018	Fixed	N	N	Y
Phone	75	75	January	Q1	2018	Fixed	N	Y	N
Electricity	60	55	February	Q1	2018	Variable	N	Y	N
Gas	70	75	February	Q1	2018	Variable	N	Y	N
Out to Eat	310	250	February	Q1	2018	Variable	Y	N	N
Fuel	90	100	February	Q1	2018	Variable	Y	N	N
Vehicle	200	200	February	Q1	2018	Fixed	N	N	Y
Phone	75	75	February	Q1	2018	Fixed	N	Y	N
Electricity	55	55	March	Q1	2018	Variable	N	Y	N
Gas	75	75	March	Q1	2018	Variable	N	Y	N
Out to Eat	320	250	March	Q1	2018	Variable	Y	N	N
Fuel	110	100	March	Q1	2018	Variable	Y	N	N
Vehicle	200	200	March	Q1	2018	Fixed	N	N	Y
Phone	75	75	March	Q1	2018	Fixed	N	Y	N

Into this quick pie chart that shares a breakdown of where money was spent during January:



- This example is relatively straightforward. But Excel has tons of capabilities to create as complex of a dashboard as you require.

Before building the Dashboard: what you should know

1. Import your data into Excel

- In order to create a dashboard, your data first needs to exist in Excel. If it's already there? Great—there's nothing more you need to do with this step. But, if not? You'll need to import it into an Excel workbook.
- There are numerous ways to do this with ranging complexities—depending on where your data exists currently. So, your best bet is to research how to import your specific data format.

2. Clean your data

- When working with data within Excel, it's important that each piece of information lives within its own cell.
- If your existing spreadsheet is a bit of a jumbled mess, take some time to clean it up and ensure that things are organized into their appropriate rows and columns. It's also wise to briefly analyze your data and make sure that no glaring typos or errors jump out at you.
- Now is also an excellent time to search for any duplicate information that needs to be deleted, because each row of data needs to be unique in order to utilize the dashboard feature, otherwise you'll be double counting.
- Highlight your entire dataset and then click the “Remove Duplicates” button.

	A	B	C	D	E	F	G	H	I	J
1	Item	Cost	Budgeted	Month	Quarter	Year	Expense Type	Credit Card used?	ETF?	Cash?
2	Electricity	50	55	January	Q1	2018	Variable	N	Y	N
3	Gas	60	75	January	Q1	2018	Variable	N	Y	N
4	Out to Eat	300	250	January	Q1	2018	Variable	Y	N	N
5	Fuel	100	100	January	Q1	2018	Variable	Y	N	N
6	Vehicle	200	200	January	Q1	2018	Fixed	N	N	Y
7	Phone	75	75	January	Q1	2018	Fixed	N	Y	N
8	Electricity	60	55	February	Q1	2018	Variable	N	Y	N
9	Gas	70	75	February	Q1	2018	Variable	N	Y	N
10	Out to Eat	310	250	February	Q1	2018	Variable	Y	N	N
11	Fuel	90	100	February	Q1	2018	Variable	Y	N	N
12	Vehicle	200	200	February	Q1	2018	Fixed	N	N	Y
13	Phone	75	75	February	Q1	2018	Fixed	N	Y	N
14	Electricity	55	55	March	Q1	2018	Variable	N	Y	N
15	Gas	75	75	March	Q1	2018	Variable	N	Y	N
16	Out to Eat	320	250	March	Q1	2018	Variable	Y	N	N
17	Fuel	110	100	March	Q1	2018	Variable	Y	N	N
18	Vehicle	200	200	March	Q1	2018	Fixed	N	N	Y
19	Phone	75	75	March	Q1	2018	Fixed	N	Y	N

TIP: It's best to keep your original dataset somewhere else. That way, if you make an error, you'll be able to retrieve the data that you started with.

3. Set up your workbook

- To create a dashboard, you're going to need three separate sheets (or tabs) within your Excel workbook.
- Name your first tab (the one that has all of your raw data on it) with something you'll readily recognize—such as "Data" or "Raw Data."
- Then, create a second tab labeled "Chart Data." That tab is where you'll store only the data that needs to be fed into different charts for the dashboard. Finally, create a tab labeled "Dashboard" where your various charts will appear.
- You can leave those last two tabs totally blank for now! The important part is just to get your workbook set up and ready to work with.

	A	B	C	D	E	F	G	H	I	J
1	Item	Cost	Budgeted	Month	Quarter	Year	Expense Type	Credit Card used?	ETF?	Cash?
2	Electricity	50	55	January	Q1	2018	Variable	N	Y	N
3	Gas	60	75	January	Q1	2018	Variable	N	Y	N
4	Out to Eat	300	250	January	Q1	2018	Variable	Y	N	N
5	Fuel	100	100	January	Q1	2018	Variable	Y	N	N
6	Vehicle	200	200	January	Q1	2018	Fixed	N	N	Y
7	Phone	75	75	January	Q1	2018	Fixed	N	Y	N
8	Electricity	60	55	February	Q1	2018	Variable	N	Y	N
9	Gas	70	75	February	Q1	2018	Variable	N	Y	N
10	Out to Eat	310	250	February	Q1	2018	Variable	Y	N	N
11	Fuel	90	100	February	Q1	2018	Variable	Y	N	N
12	Vehicle	200	200	February	Q1	2018	Fixed	N	N	Y
13	Phone	75	75	February	Q1	2018	Fixed	N	Y	N
14	Electricity	55	55	March	Q1	2018	Variable	N	Y	N
15	Gas	75	75	March	Q1	2018	Variable	N	Y	N
16	Out to Eat	320	250	March	Q1	2018	Variable	Y	N	N
17	Fuel	110	100	March	Q1	2018	Variable	Y	N	N
18	Vehicle	200	200	March	Q1	2018	Fixed	N	N	Y
19	Phone	75	75	March	Q1	2018	Fixed	N	Y	N
20										

◀ ▶ Data Chart Data Dashboard +

4. Understand your requirements

- When you start familiarizing yourself with dashboards in Excel, you'll quickly realize that there are tons of options. Admittedly, that can be overwhelming—which is why it's important to get clear on the why of your dashboard first.
- What's your goal with this dashboard? Do you need to track progress? Analyze a budget? Identify trends?
- By considering that first and foremost—as well as things like who you'll need to share this with and what format it will need to be in—you'll be empowered to design a dashboard that fits your needs.

How to create an Excel Dashboard

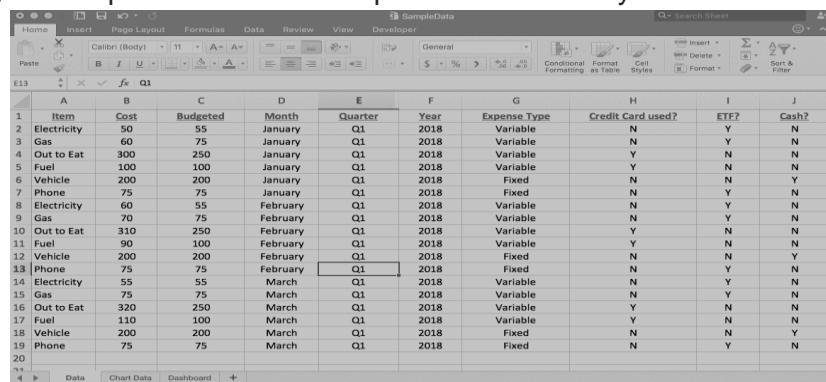
1. Figure out which charts best represent your data

- You know how we just said you'll be faced with tons of options to represent your data?
- There are bar charts, column charts, pie charts, line charts, scatter plots, waterfall charts, and so many more.
- Not all of them will be the best fit for the data that you want to represent. For example, a line chart is excellent for analyzing trends while a pie chart is effective for looking at a snapshot in time.
- Create a few different charts, look at the results, and see which ones make the most sense for displaying your data in an easily digestible manner.
- With so many options, we couldn't possibly dive into every chart type in detail here. So, instead, we're going to dip our toes into the dashboard waters by focusing on the step-by-step process for creating one specific type of chart: a column chart.
- We'll stick with the same budget data set that we used above. Our goal is to create column charts that will display how much we spend on each individual budget line item per month in the first quarter. This means that we'll end up with separate charts for electricity, gas, phone, etc.

Let's get started.

2. Filter your data

- When creating a chart, you're not going to need to use all of your data at once. You'll need to filter through it to focus on only the pieces you need at that given time.
- The easiest way to do this is by using the “Filter” option within Excel. For example, we want to filter by item type and only see numbers related to our electricity expenses.
- To do that, we'll highlight the entire data set, click the “Data” ribbon in the toolbar, and then click the “Filter” button.
- When doing so, you'll see that little arrows appear next to your column headers. If you click one of those arrows, you'll be presented with a drop-down menu that you can use to filter your data.

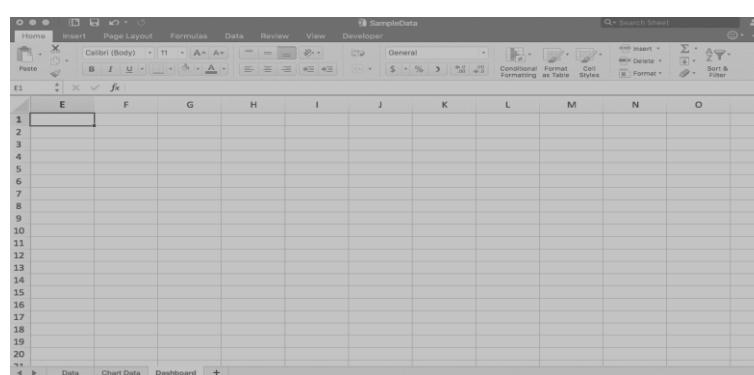


	A	B	C	D	E	F	G	H	I	J
1	Item	Cost	Budgeted	Month	Quarter	Year	Expense Type	Credit Card used?	ETF?	Cash?
2	Electricity	50	55	January	Q1	2018	Variable	N	Y	N
3	Gas	60	75	January	Q1	2018	Variable	N	Y	N
4	Out to Eat	300	250	January	Q1	2018	Variable	Y	N	N
5	Fuel	100	100	January	Q1	2018	Variable	Y	N	N
6	Vehicle	200	200	January	Q1	2018	Fixed	N	N	Y
7	Phone	75	75	January	Q1	2018	Fixed	N	Y	N
8	Electricity	60	55	February	Q1	2018	Variable	N	Y	N
9	Gas	70	75	February	Q1	2018	Variable	N	Y	N
10	Out to Eat	310	250	February	Q1	2018	Variable	Y	N	N
11	Fuel	90	100	February	Q1	2018	Variable	Y	N	N
12	Vehicle	200	200	February	Q1	2018	Fixed	N	N	Y
13	Phone	75	75	February	Q1	2018	Fixed	N	Y	N
14	Electricity	55	55	March	Q1	2018	Variable	N	Y	N
15	Gas	75	75	March	Q1	2018	Variable	N	Y	N
16	Out to Eat	320	250	March	Q1	2018	Variable	Y	N	N
17	Fuel	110	100	March	Q1	2018	Variable	Y	N	N
18	Vehicle	200	200	March	Q1	2018	Fixed	N	N	Y
19	Phone	75	75	March	Q1	2018	Fixed	N	Y	N
20										

- When you've filtered down to only the data that you want, highlight all of the cells of data, hit “copy,” and then paste only those rows into your “Chart Data” tab of your workbook. That's the tab that you'll pull data from when building your charts.
- Why can't you just select data from your regular “Data” tab? Put simply, because even though you've filtered the data, those other irrelevant rows are still included there (albeit hidden), meaning they'll throw things off in your chart.

3. Build your chart

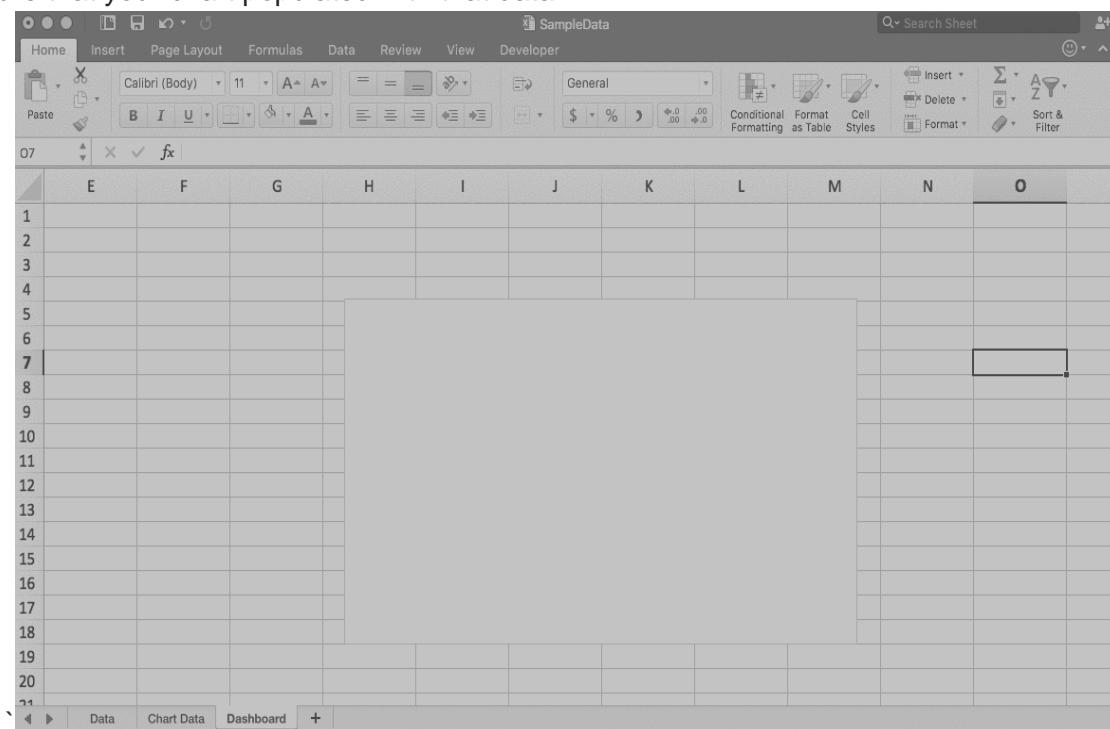
- Now that you have only the data that you need, you're ready to begin building your chart.
- Click on the “Dashboard” tab of your worksheet, click the “Insert” button in the toolbar, and then select the type of chart you want from the menu. In this case, we're going to use a clustered column chart.
- When you insert the chart, you'll see a blank box. We'll cover how to get your data to appear there in the next step.



TIP: Still aren't sure which chart option is the best fit for your data? Highlight all of your rows of data in your “Chart Data” tab and then click “Recommended Charts” within the “Insert” ribbon. Excel will suggest some charts for you to use.

4. Select your data

- Now that you have that blank box inserted in your “Dashboard” tab of your workbook, it’s time to pull some data in.
- To do so, right-click on that box and then choose “Select Data.” After that, navigate over to your “Chart Data” tab (where your filtered data is living) and highlight all of the data that you want to display, minus the column headers. In this case, we’re selecting the “Item” and “Cost” data, since the purpose of my chart is to show how much we spend on electricity each month.
- That data that you just selected is for the vertical axis. But you still need to select your data for the horizontal axis of your chart.
- To do that, click the button within the “Horizontal Axis” field of the “Select Data” popup and then highlight the information you need for the horizontal axis—in this case, the months.
- After doing so, hit your “Enter” key, click “OK,” and then head over to your “Dashboard” tab to ensure that your chart populated with that data.

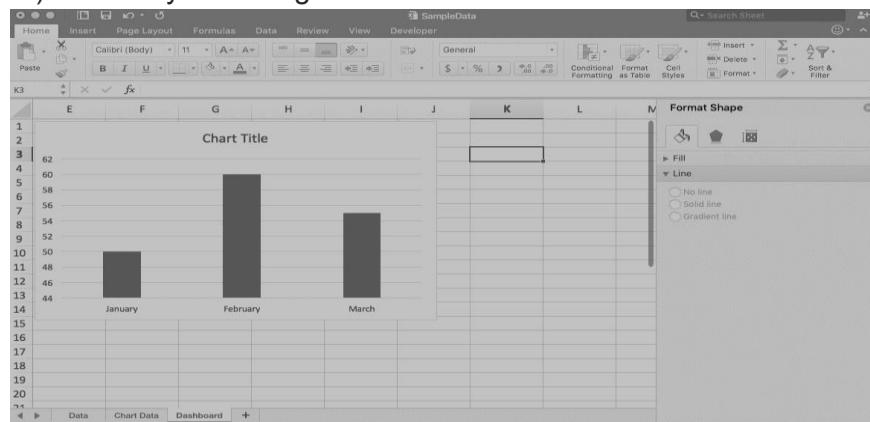


5. Double-check your data

- At this point, it’s wise to take a quick look at your chart and make sure that nothing looks off. Mistakes can happen. So, it’s worth it to take the time to ensure that your chart is pulling in your data correctly.
- Does something look off?
- Do some troubleshooting to figure out where you went wrong.

6. Polish your chart

- Put the finishing touches on your chart.
- From changing the colors to match your brand to adding labels, titles, or any other information that is required, you can polish up your chart by double-clicking on the chart area and then using the options in the toolbar (there are buttons up there for everything from “Add Chart Element” to “Change Colors) to make your changes.

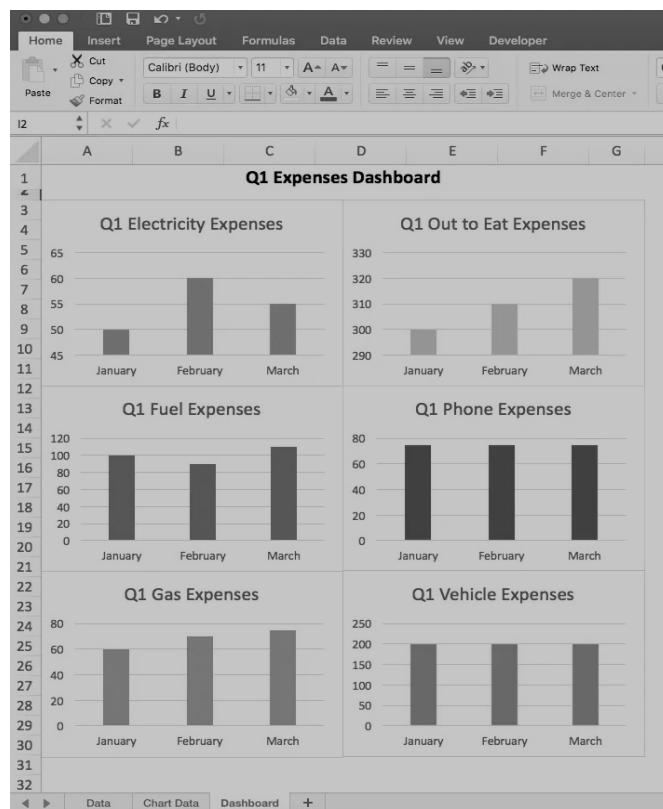


7. Repeat the process for other data

- A true Excel dashboard includes several different charts and gauges to display data. So, to complete our dashboard in this example scenario, we'll go back and repeat all of those steps for our other budget line items such as phone, vehicle, gas, etc.
- One important thing to note: You don't want to delete any data in your "Chart Data" tab, since the data there, is what's feeding your charts.
- So, when going through the process of making your other charts, make sure to paste the new data sets below each other, rather than deleting and replacing them within that tab:

Item	Cost	Budgeted	Month	Quarter	Year	Expense Type	Credit Card used?	ETF?	Cash?
Electricity	50	55	January	Q1	2018	Variable	N	Y	N
Gas	60	75	January	Q1	2018	Variable	N	Y	N
Out to Eat	300	250	January	Q1	2018	Variable	Y	N	N
Fuel	100	100	January	Q1	2018	Variable	Y	N	N
Vehicle	200	200	January	Q1	2018	Fixed	N	N	Y
Phone	75	75	January	Q1	2018	Fixed	N	Y	N
Electricity	60	55	February	Q1	2018	Variable	N	Y	N
Gas	70	75	February	Q1	2018	Variable	N	Y	N
Out to Eat	310	250	February	Q1	2018	Variable	Y	N	N
Fuel	90	100	February	Q1	2018	Variable	Y	N	N
Vehicle	200	200	February	Q1	2018	Fixed	N	N	Y
Phone	75	75	February	Q1	2018	Fixed	N	Y	N
Electricity	55	55	March	Q1	2018	Variable	N	Y	N
Gas	75	75	March	Q1	2018	Variable	N	Y	N
Out to Eat	320	250	March	Q1	2018	Variable	Y	N	N
Fuel	110	100	March	Q1	2018	Variable	Y	N	N
Vehicle	200	200	March	Q1	2018	Fixed	N	N	Y
Phone	75	75	March	Q1	2018	Fixed	N	Y	N

- After we've done that? We'll end up with a tab that shows how much we spend on each item each month.



2.5 Data Analysis Using Statistics

What is Statistical Analysis?

- Statistical analysis is the process of collecting and analyzing data in order to discern patterns and trends.
- It is a method for removing bias from evaluating data by employing numerical analysis.
- This technique is useful for collecting the interpretations of research, developing statistical models, and planning surveys and studies.
- Statistical analysis is a scientific tool that helps collect and analyze large amounts of data to identify common patterns and trends to convert them into meaningful information.
- In simple words, statistical analysis is a data analysis tool that helps draw meaningful conclusions from raw and unstructured data.

Types of Statistical Analysis

Given below are the 6 types of statistical analysis:

Descriptive Analysis

- Descriptive statistical analysis involves collecting, interpreting, analyzing, and summarizing data to present them in the form of charts, graphs, and tables.
- Rather than drawing conclusions, it simply makes the complex data easy to read and understand.

Inferential Analysis

- The inferential statistical analysis focuses on drawing meaningful conclusions on the basis of the data analyzed.
- It studies the relationship between different variables or makes predictions for the whole population.

Predictive Analysis

- Predictive statistical analysis is a type of statistical analysis that analyzes data to derive past trends and predict future events on the basis of them.
- It uses machine learning algorithms, data mining, data modelling, and artificial intelligence to conduct the statistical analysis of data.

Prescriptive Analysis

- The prescriptive analysis conducts the analysis of data and prescribes the best course of action based on the results.
- It is a type of statistical analysis that helps you make an informed decision.

Exploratory Data Analysis

- Exploratory analysis is similar to inferential analysis, but the difference is that it involves exploring the unknown data associations.
- It analyzes the potential relationships within the data.

Causal Analysis

- The causal statistical analysis focuses on determining the cause-and-effect relationship between different variables within the raw data.
- In simple words, it determines why something happens and its effect on other variables.
- This methodology can be used by businesses to determine the reason for failure.

Statistical Analysis Process

Given below are the 5 steps to conduct a statistical analysis that you should follow:

Step 1: Identify and describe the nature of the data that you are supposed to analyze.

Step 2: The next step is to establish a relation between the data analyzed and the sample population to which the data belongs.

Step 3: The third step is to create a model that clearly presents and summarizes the relationship between the population and the data.

Step 4: Prove if the model is valid or not.

Step 5: Use predictive analysis to predict future trends and events likely to happen.

Benefits of Statistical Analysis

- Statistical analysis can be called a boon to mankind and has many benefits for both individuals and organizations.
- Given below are some of the reasons why you should consider investing in statistical analysis:
 - It can help you determine the monthly, quarterly, yearly figures of sales profits, and costs making it easier to make your decisions.
 - It can help you make informed and correct decisions.
 - It can help you identify the problem or cause of the failure and make corrections. For example, it can identify the reason for an increase in total costs and help you cut the wasteful expenses.
 - It can help you conduct market analysis and make an effective marketing and sales strategy.
 - It helps improve the efficiency of different processes.

Statistical Analysis Methods

Mean

- Mean or average mean is one of the most popular methods of statistical analysis.
- Mean determines the overall trend of the data and is very simple to calculate.
- Mean is calculated by summing the numbers in the data set together and then dividing it by the number of data points.
- Despite the ease of calculation and its benefits, it is not advisable to resort to mean as the only statistical indicator as it can result in inaccurate decision making.

$$\text{Mean} = \frac{\text{Sum of All Data Points}}{\text{Number of Data Points}}$$

$$\text{Mean} = \text{Assumed Mean} + \frac{\text{Sum of All Deviations}}{\text{Number of Data Points}}$$

Marks Obtained	Number of student	Class Mark	$f_i x_i$
10-20	5	15	75
20-30	5	25	125
30-40	8	35	280
40-50	12	45	540
Total	$\sum f_i = 30$		$\sum f_i x_i = 1020$

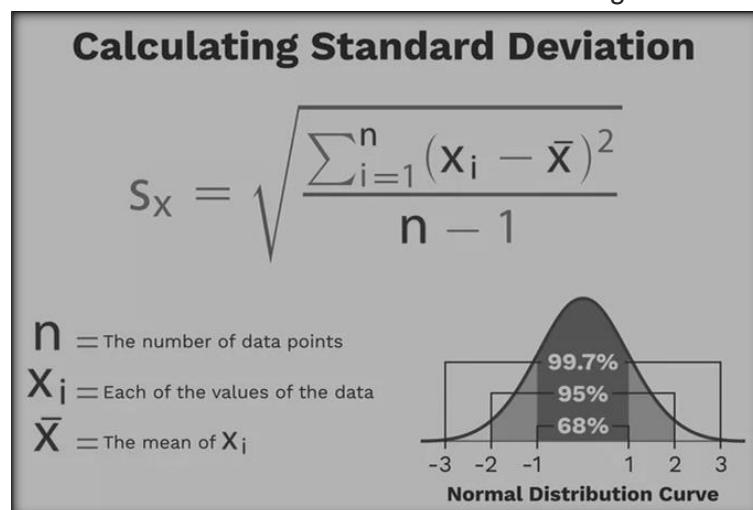
The mean of the data given above is:

$$\bar{x} = \frac{\sum f_i x_i}{\sum f_i} = \frac{1020}{30} = 34$$

Thus, the mean of the given data is 34.

Standard Deviation

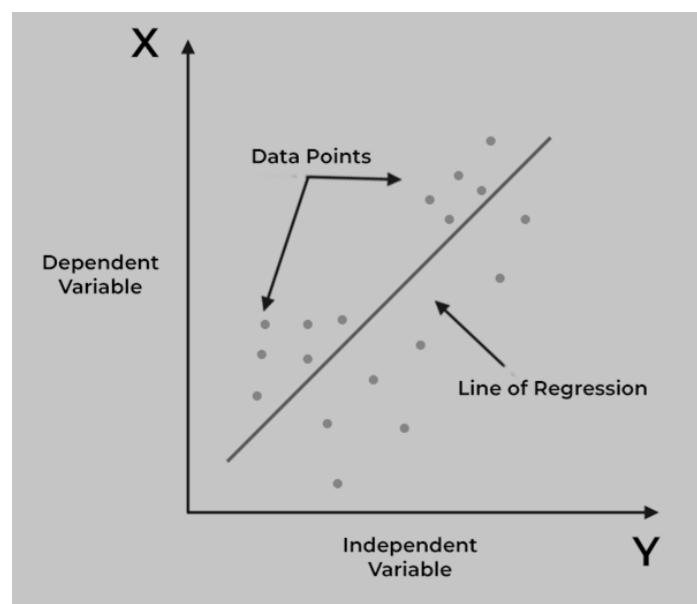
- Standard deviation is another very widely used statistical tool or method.
- It analyzes the deviation of different data points from the mean of the entire data set.
- It determines how data of the data set is spread around the mean.
- You can use it to decide whether the research outcomes can be generalized or not.



Regression

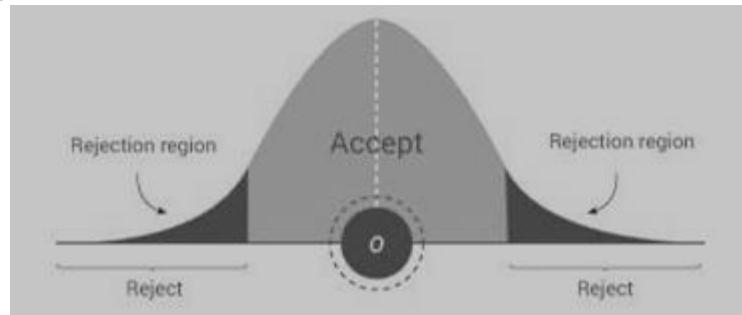
- Regression is a statistical tool that helps determine the cause-and-effect relationship between the variables.
- It determines the relationship between a dependent and an independent variable.
- It is generally used to predict future trends and events.
- Simple linear regression is commonly used in forecasting and financial analysis for a company to tell how a change in the GDP could affect sales.

For Example: Microsoft Excel and other software can do all the calculations, but it's good to know how the mechanics of simple linear regression work.



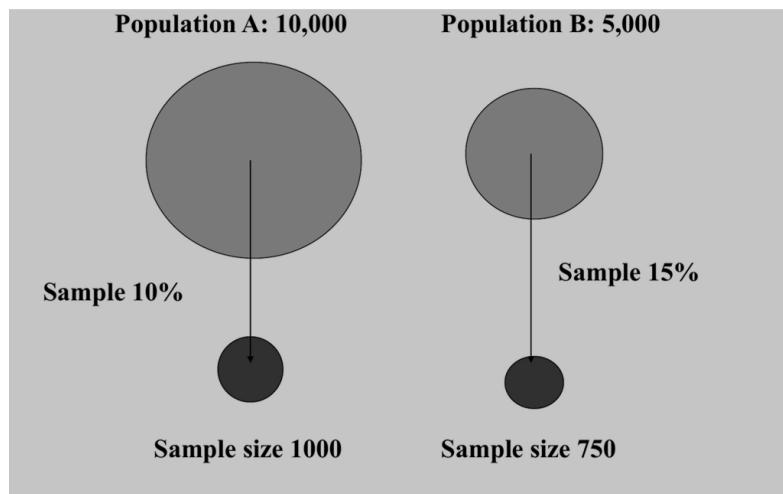
Hypothesis Testing

- Hypothesis testing can be used to test the validity or trueness of a conclusion or argument against a data set.
- The hypothesis is an assumption made at the beginning of the research and can hold or be false based on the analysis results.



Sample Size Determination

- Sample size determination or data sampling is a technique used to derive a sample from the entire population, which is representative of the population.
- This method is used when the size of the population is very large.
- You can choose from among the various data sampling techniques such as snowball sampling, convenience sampling, and random sampling.



Statistical Analysis Example

- Look at the standard deviation sample calculation given below to understand more about statistical analysis.
- The weights of 5 pizza bases in cms are as follows:

Particulars (Weight in cms)	Mean Deviation	Square of Mean Deviation
9	9-6.4 = 2.6	(2.6)2 = 6.76
2	2-6.4 = - 4.4	(-4.4)2 = 19.36
5	5-6.4 = - 1.4	(-1.4)2 = 1.96
4	4-6.4 = - 2.4	(-2.4)2 = 5.76
12	12-6.4 = 5.6	(5.6)2 = 31.36

Calculation of Mean = $(9+2+5+4+12)/5 = 32/5 = 6.4$

Calculation of mean of squared mean deviation = $(6.76+19.36+1.96+5.76+31.36)/5 = 13.04$

Sample Variance = 13.04

Standard Deviation = $\sqrt{13.04} = 3.611$

Section 3: Exercises

Exercise 1: Create a 3-dimensional column chart for following data.

a.

Car Sales By Gender			Charity Donations Forecast		
	Men	Women		Jan	Feb
Fiat	3%	32%	Income from Libel cases	150.00	175.00
Ford	39%	12%	Income from Appearances	750.00	250.00
BMW	21%	8%	Charity Donations	225.00	106.25
Mercedes	6%	17%	Tax Rebate	13.50	6.38
Vauxhall	19%	10%			
Nissan	12%	21%	Figures are in \$ 1000's		

b.

c.

<u>Superhero Laundry</u>				
Items	Spiderman	Superman	Batman	The Hulk
Capes	0	2	1	0
Spandex Suits	4	1	3	0
Masks	5	0	2	0
Underpants	0	6	2	8

Exercise 2: Create a pivot table from this data, then use the filters within to view the average prices of holidays that have a **Travel Method** of **Plane** and a **Resort Name** that begins with the letter **S**.

Country	Resort Name	No of Days	Travel Method	Price	Holiday ID
Australia	Great Barrier Reef	32	Plane	£750	I990AUS
Australia	Perth	28	Plane	£985	AUS112J
Chile	Santiago	21	Plane	£1,259	CH266H
England	London	3	Train	£69	I456UK
England	Bognor	1	Coach	£12	BG726H
France	Lyon	14	Plane	£399	A7995FR
France	Paris - Euro Disney	5	Train	£269	TH789FR
France	Paris - Euro Disney	3	Train	£125	TH788FR
France	Nice	7	Plane	£289	I7897FR
France	Toulouse	7	Train	£256	SG7637L
France	Nimes	7	Plane	£287	FR5625J
Germany	Black Forest	4	Coach	£69	A111G
Germany	Berlin	7	Coach	£289	BR6736G
Peru	Lima	21	Plane	£975	PG7836G
Saudi Arabia	Riyadh	14	Plane	£995	KSA8987
Spain	Barcelona	4	Train	£219	I6675SP
Spain	Nerja	6	Plane	£198	TH990ESP
Spain	Malaga	16	Plane	£234	A776ESP
Spain	Seville	14	Plane	£288	NM9876Y
Spain	Seville	10	Plane	£199	TH8956SP
Spain	Barcelona	8	Plane	£177	AJ9836L
Spain	Barcelona	7	Coach	£199	GG9836P
Spain	Malaga	14	Plane	£301	PL8726P
Spain	Barcelona	4	Train	£219	I6675SP
Spain	Seville	14	Train	£299	SV767HH
Spain	Madrid	8	Plane	£277	WE6735L
Spain	Granada	10	Plane	£345	GR7878G
Trinidad	Port of Spain	14	Plane	£885	TT67624G

Exercise 3: Apply conditional formatting rules, so that:

- Anyone achieving more than 100% of their target is shown in green
- Anyone achieving less than 100% of their target is shown in red
- Anyone achieving exactly 100% of their target is shown in yellow

Name	Target	Achieved	Percentage
Jerry Mouse	£85,000	£94,500	111.18%
Ivor Cricket	£1,20,000	£1,18,000	98.33%
Curly Sue	£1,00,000	£1,07,000	107.00%
Randy Cunningham	£65,000	£29,000	44.62%
Arthur Shilling	£50,000	£67,000	134.00%
Hugh Jass	£1,50,000	£1,98,000	132.00%
Wendy House	£95,000	£1,24,750	131.32%
Carol Service	£95,000	£94,750	99.74%
Christmas T. Rees	£75,000	£75,000	100.00%

Exercise 4: Create a Heat Map of following data using Conditional Formatting Colour Scales.

Average Monthly Temperatures at Central Park, New York													
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	
2009	27.9	36.7	42.4	54.5	62.5	67.5	72.7	75.7	66.3	55.0	51.2	35.9	
2010	32.5	33.1	48.2	57.9	65.3	74.7	81.3	77.4	71.1	58.1	47.9	32.8	
2011	29.7	36.0	42.3	54.3	64.5	72.3	80.2	75.3	70.0	57.1	51.9	43.3	
2012	37.3	40.9	50.9	54.8	65.1	71.0	78.8	76.7	68.8	58.0	43.9	41.5	
2013	35.1	33.9	40.1	53.0	62.8	72.7	79.8	74.6	67.9	60.2	45.3	38.5	
2014	28.6	31.6	37.7	52.3	64.0	72.5	76.1	74.5	69.7	59.6	45.3	40.5	
2015	29.9	23.9	38.1	54.3	68.5	71.2	78.8	79.0	74.5	58.0	52.8	50.8	
2016	34.5	37.7	48.9	53.3	62.8	72.3	78.7	79.2	71.8	58.8	49.8	38.3	
2017	38.0	41.6	39.2	57.2	61.1	72.0	76.8	74.0	70.5	64.1	46.6	33.4	

Exercise 5: Participate in a group discussion on following topics:

1. Types of Business Analytics
2. Use case of Business Analytics
3. Dashboarding

Section 4: Assessment Questionnaire

1. What are the common data formats in Microsoft Excel?
2. How are these data formats used in Microsoft Excel?
3. What are the cell references?
4. What are the functions of different cell references?
5. Which key or combination of keys allow you to toggle between the absolute, relative and mixed cell references?
6. What is the function of the dollar sign (\$) in Microsoft Excel?
7. Slicers in Excel is used to quickly and easily _____ pivot tables.
8. By Right-clicking on the table and then click on _____ so that the changes in the data get reflected in the table.
9. Dashboard is as a _____ of data.
10. What are the types of Statistical Analysis?
11. What are the methods of Statistical Analysis?
12. What is the formula to calculate Mean?
13. What are the types of Business Analytics?
14. What are the use cases of Business Analytics?
15. _____ determines how data of the data set is spread around the mean.

-----End of the Module-----

MODULE 3

PROGRAMMING BASICS AND DATA ANALYTICS WITH PYTHON

Section 1: Learning Outcomes

After completing this module, you will be able to:

- Introduce Python
- Use Python in Data Analytics
- Setup Python Environment
- Perform basic coding in Python
- Use Python Operators
- Perform Statistical Computing
- Perform Mathematical Computing Using NumPy
- Use Pandas
- Perform Data Visualization using Python
- Develop and Evaluate a Model

Section 2: Relevant Knowledge

3.1 Introduction to Python

- Python is a simple, general purpose, high level, and object-oriented programming language.
- Python is an interpreted scripting language, also *Guido Van Rossum* is known as the founder of Python programming.

What is Python

- Python is a general purpose, dynamic, high-level, and interpreted programming language.
- It supports Object Oriented programming approach to develop applications.
- It is simple and easy to learn and provides lots of high-level data structures.
- Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development.
- Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development.
- Python supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles.
- Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc.
- We don't need to use data types to declare variable because it is dynamically typed so we can write `a=10` to assign an integer value in an integer variable.
- Python makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

Python Features

Python provides many useful features which make it popular and valuable from the other programming languages. It supports object-oriented programming, procedural programming approaches and provides dynamic memory allocation. We have listed below a few essential features.

Easy to Learn and Use

- Python is easy to learn as compared to other programming languages.
- Its syntax is straightforward and much the same as the English language.
- There is no use of the semicolon or curly-bracket, the indentation defines the code block.
- It is the recommended programming language for beginners.

Expressive Language:

- Python can perform complex tasks using a few lines of code.
- A simple example, the hello world program you simply type `print("Hello World")`. It will take only one line to execute, while Java or C takes multiple lines.

Interpreted Language

- Python is an interpreted language; it means the Python program is executed one line at a time.
- The advantage of being interpreted language, it makes debugging easy and portable.

Cross-platform Language

- Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, etc. So, we can say that Python is a portable language.
- It enables programmers to develop the software for several competing platforms by writing a program only once.

Free and Open Source

- Python is freely available for everyone.
- It is freely available on its official website www.python.org.
- It has a large community across the world that is dedicatedly working towards make new python modules and functions.
- Anyone can contribute to the Python community.
- The open-source means, "Anyone can download its source code without paying any penny."

Object-Oriented Language

- Python supports object-oriented language and concepts of classes and objects come into existence.
- It supports inheritance, polymorphism, and encapsulation, etc.
- The object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

Extensible

- It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code.
- It converts the program into byte code, and any platform can use that byte code.

Large Standard Library

- It provides a vast range of libraries for the various fields such as machine learning, web developer, and also for the scripting.
- There are various machine learning libraries, such as Tensor flow, Pandas, Numpy, Keras, and Pytorch, etc.
- Django, flask, pyramids are the popular framework for Python web development.

GUI Programming Support

- Graphical User Interface is used for the developing Desktop application.
- PyQt5, Tkinter, Kivy are the libraries which are used for developing the web application.

Integrated

- It can be easily integrated with languages like C, C++, and JAVA, etc.
- Python runs code line by line like C,C++ Java. It makes easy to debug the code.

Embeddable

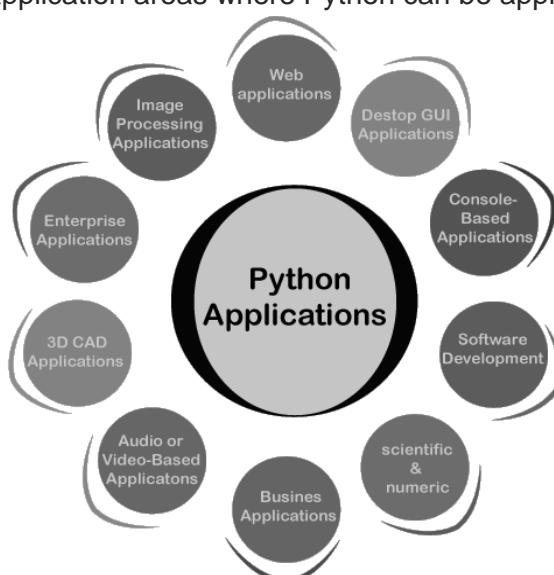
- The code of the other programming language can use in the Python source code.
- We can use Python source code in another programming language as well.
- It can embed other language into our code.

Dynamic Memory Allocation

- In Python, we don't need to specify the data-type of the variable.
- When we assign some value to the variable, it automatically allocates the memory to the variable at run time.
- Suppose we are assigned integer value 15 to x, then we don't need to write int x = 15. Just write x = 15.

Python Applications

- Python is known for its general-purpose nature that makes it applicable in almost every domain of software development.
- Python makes its presence in every emerging field. It is the fastest-growing programming language and can develop any application.
- Here, we are specifying application areas where Python can be applied.



Web Applications

- We can use Python to develop web applications.
- It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser, etc. One of Python web-framework named Django is used on Instagram.
- Python provides many useful frameworks, and these are given below:
 - Django and Pyramid framework(Use for heavy applications)
 - Flask and Bottle (Micro-framework)
 - Plone and Django CMS (Advance Content management)

Desktop GUI Applications

- The GUI stands for the Graphical User Interface, which provides a smooth interaction to any application. Python provides a Tk GUI library to develop a user interface.
- Some popular GUI libraries are given below.
 - Tkinter or Tk
 - wxWidgetM
 - Kivy (used for writing multitouch applications)
 - PyQt or Pyside

Console-based Application

- Console-based applications run from the command-line or shell.
- These applications are computer program which are used commands to execute.
- This kind of application was more popular in the old generation of computers. Python can develop this kind of application very effectively.
- It is famous for having REPL, which means the Read-Eval-Print Loop that makes it the most suitable language for the command-line applications.

Software Development

- Python is useful for the software development process.
- It works as a support language and can be used to build control and management, testing, etc.
 - SCons is used to build control.
 - Buildbot and Apache Gumps are used for automated continuous compilation and testing.
 - Round or Trac for bug tracking and project management.

Scientific and Numeric

- This is the era of Artificial intelligence where the machine can perform the task the same as the human. Implementing machine learning algorithms require complex mathematical calculation.
- Python has many libraries for scientific and numeric such as Numpy, Pandas, Scipy, Scikit-learn, etc.
- If you have some basic knowledge of Python, you need to import libraries on the top of the code.
- Few popular frameworks of machine libraries are given below.
 - SciPy
 - Scikit-learn
 - NumPy
 - Pandas
 - Matplotlib

Business Applications

- Business Applications differ from standard applications.
- E-commerce and ERP are an example of a business application.
- This kind of application requires extensively, scalability and readability, and Python provides all these features.
- Oddo is an example of the all-in-one Python-based application which offers a range of business applications.
- Python provides a Tryton platform which is used to develop the business application.

Audio or Video-based Applications

- Python is flexible to perform multiple tasks and can be used to create multimedia applications.
- Some multimedia applications which are made by using Python are TimPlayer, cplay, etc.
- The few multimedia libraries are given below.
 - Gstreamer
 - Pyglet
 - QT Phonon

3D CAD Applications

- The CAD (Computer-aided design) is used to design engineering related architecture.
- It is used to develop the 3D representation of a part of a system.
- Python can create a 3D CAD application by using the following functionalities.
- Fandango (Popular)
 - CAMVOX
 - HeeksCNC
 - AnyCAD
 - RCAM

Enterprise Applications

- Python can be used to create applications that can be used within an Enterprise or an Organization.
- Some real-time applications are OpenERP, Tryton, Picalo, etc.

Image Processing Application

- Python contains many libraries that are used to work with the image.
- The image can be manipulated according to our requirements.
- Some libraries of image processing are given below.
 - OpenCV
 - Pillow
 - SimpleITK

Python 2 vs. Python 3

- In most of the programming languages, whenever a new version releases, it supports the features and syntax of the existing version of the language, therefore, it is easier for the projects to switch in the newer version.
- However, in the case of Python, the two versions Python 2 and Python 3 are very much different from each other.

	Python 2	Python 3
String type	ASCII	Unicode
Error handling	except NameError, err	except NameError as err
Print	print function is treated more as a statement	print() function, treated as a function
Integer division	numbers without any digits are integers (3/2 = 1)	intuitive way of treating numbers: 3/2 = 1.5
Integer size	limited to 32 bits	Unlimited
Range	xrange method	range method
Rules of ordering comparisons	complex	Simplified
Leak of variables	never changes	changes while using it inside for loop

A list of differences between Python 2 and Python 3 are given below:

1. Python 2 uses **print** as a statement and used as print "something" to print some string on the console.
2. On the other hand, Python 3 uses **print** as a function and used as print("something") to print something on the console.
3. Python 2 uses the function raw_input() to accept the user's input. It returns the string representing the value, which is typed by the user. To convert it into the integer, we need to use the int() function in Python.
4. On the other hand, Python 3 uses input() function which automatically interpreted the type of input entered by the user. However, we can cast this value to any type by using primitive functions (int(), str(), etc.).
5. In Python 2, the implicit string type is ASCII, whereas, in Python 3, the implicit string type is Unicode.
6. Python 3 doesn't contain the xrange() function of Python 2. The xrange() is the variant of range() function which returns a xrange object that works similar to Java iterator. The range() returns a list for example the function range(0,3) contains 0, 1, 2.
7. There is also a small change made in Exception handling in Python 3. It defines a keyword **as** which is necessary to be used. We will discuss it in Exception handling section of Python programming tutorial.

Java vs Python Program

- Unlike the other programming languages, Python provides the facility to execute the code using few lines. **For example** – Suppose we want to print the “Hello World” program in Java; it will take three lines to print it.

Java Program

```
public class HelloWorld {  
    public static void main(String[] args){  
        // Prints "Hello, World" to the terminal window.  
        System.out.println("Hello World");  
    }  
}
```

Python Program

On the other hand, we can do this using one statement in Python.

```
print("Hello World")
```

Both programs will print the same result, but it takes only one statement without using a semicolon or curly braces in Python.

Python Basic Syntax

- There is no use of curly braces or semicolon in Python programming language.
- It is English-like language.
- But Python uses the indentation to define a block of code.
- Indentation is nothing but adding whitespace before the statement when it is needed.

For example -

```
def func():  
    statement 1  
    statement 2  
    .....  
    .....  
    statement N
```

In the above example, the statements that are same level to right belong to the function. Generally, we can use four whitespaces to define indentation.

Why learn Python?

Python provides many useful features to the programmer. These features make it most popular and widely used language. We have listed below few-essential feature of Python.

- Easy to use and Learn
- Expressive Language
- Interpreted Language
- Object-Oriented Language
- Open Source Language
- Extensible
- Learn Standard Library
- GUI Programming Support
- Integrated
- Embeddable
- Dynamic Memory Allocation
- Wide Range of Libraries and Frameworks

Where is Python used?

Python is a general-purpose, popular programming language and it is used in almost every technical field. The various areas of Python use are given below.

- Data Science
- Data Mining
- Desktop Applications
- Console-based Applications
- Mobile Applications
- Software Development
- Artificial Intelligence
- Web Applications
- Enterprise Applications
- 3D CAD Applications
- Machine Learning
- Computer Vision or Image Processing Applications.
- Speech Recognitions

3.2 Python Environment Setup and Essentials

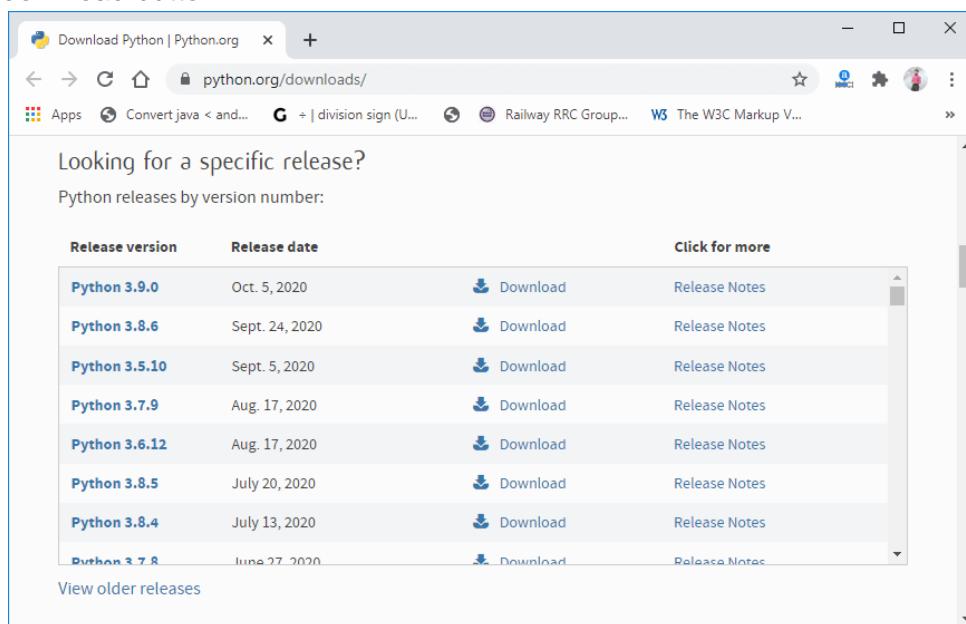
In order to become Python developer, the first step is to learn how to install or update Python on a local machine or computer.

Installation on Windows

Visit the link <https://www.python.org/downloads/> to download the latest release of Python. In this process, we will install Python 3.8.6 on our Windows operating system. When we click on the above link, it will bring us the following page.

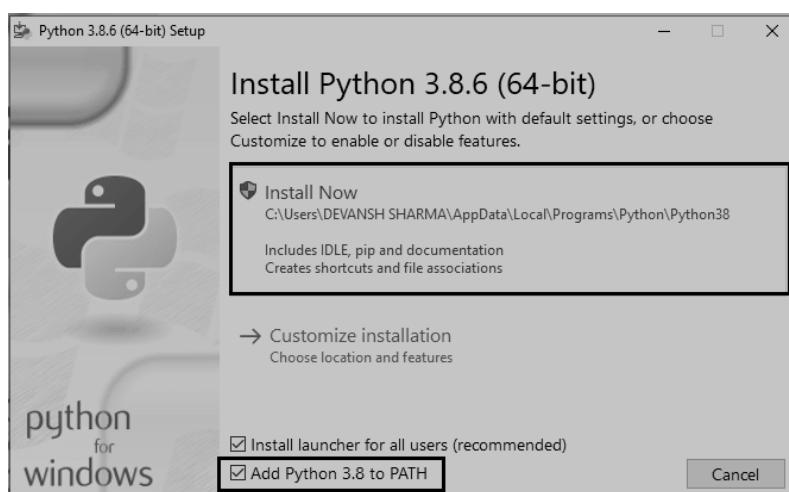
Step - 1: Select the Python's version to download.

Click on the download button.



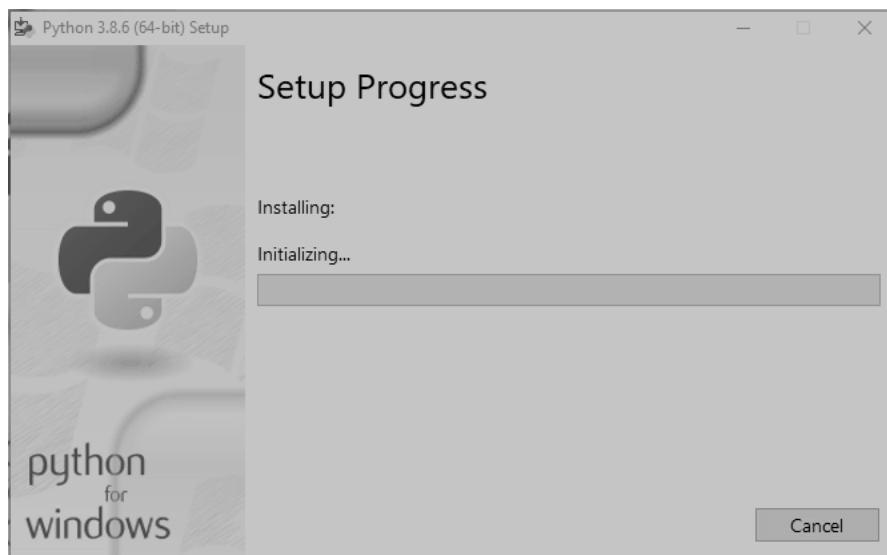
Step - 2: Click on the Install Now

Double-click the executable file, which is downloaded; the following window will open. Select Customize installation and proceed. Click on the Add Path check box, it will set the Python path automatically.

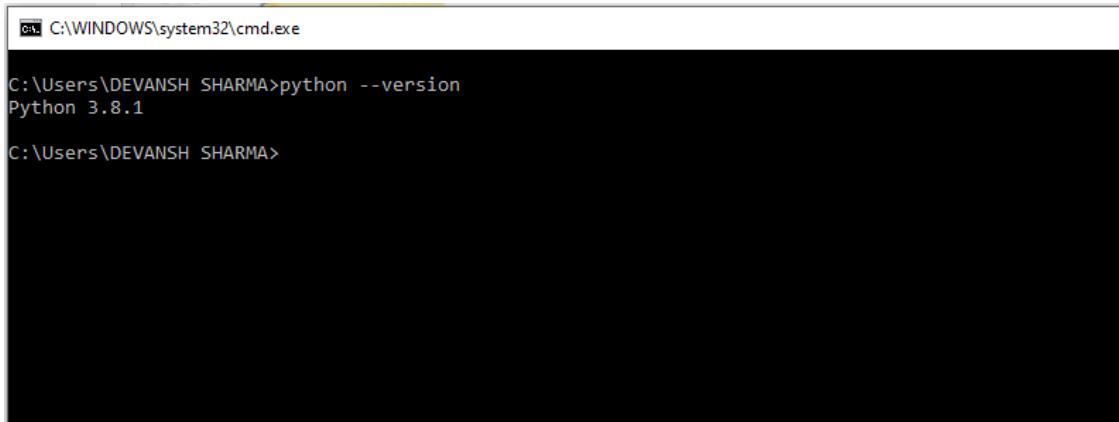


We can also click on the customize installation to choose desired location and features. Other important thing is install launcher for the all user must be checked.

Step - 3 Installation in Process



- Now, try to run python on the command prompt.
- Type the command `python -version` in case of python3.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\DEVANSH SHARMA>python --version
Python 3.8.1
C:\Users\DEVANSH SHARMA>
```

Program to print Hello World on the Console

In this Section, we will discuss the basic syntax of Python, we will run a simple program to print **Hello World** on the console.

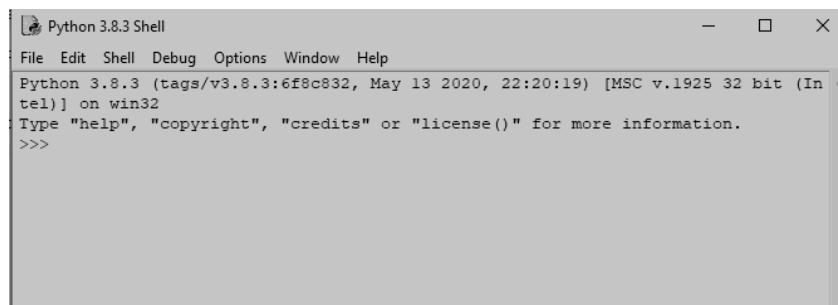
Python provides us the two ways to run a program:

- Using Interactive interpreter prompt
- Using a script file

Let's discuss each one of them in detail.

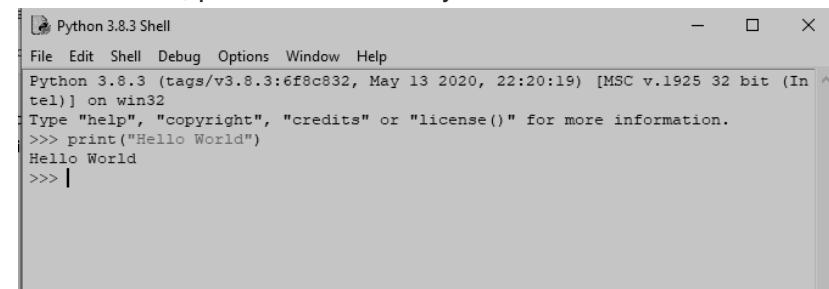
Interactive Interpreter Prompt

- Python provides us the feature to execute the Python statement one by one at the interactive prompt.
- It is preferable in the case where we are concerned about the output of each line of our Python program.
- To open the interactive mode, open the terminal (or command prompt) and type python (python3 in case if you have Python2 and Python3 both installed on your system).
- It will open the following prompt where we can execute the Python statement and check their impact on the console.



The screenshot shows the Python 3.8.3 Shell window. The title bar says "Python 3.8.3 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the Python version information: "Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In tel)] on win32". Below this, it says "Type 'help', 'copyright', 'credits' or 'license()' for more information." and shows the prompt ">>>".

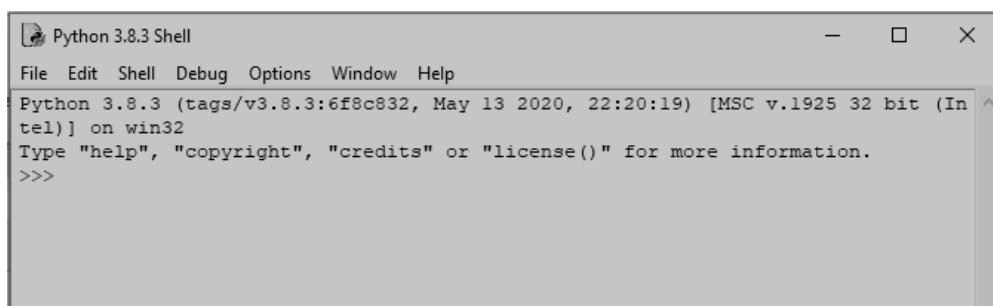
After writing the print statement, press the **Enter** key.



The screenshot shows the Python 3.8.3 Shell window after a print statement has been entered. The main window now displays the message "Hello World" followed by a new line, indicating the execution of the code. The prompt ">>>" is still visible at the bottom.

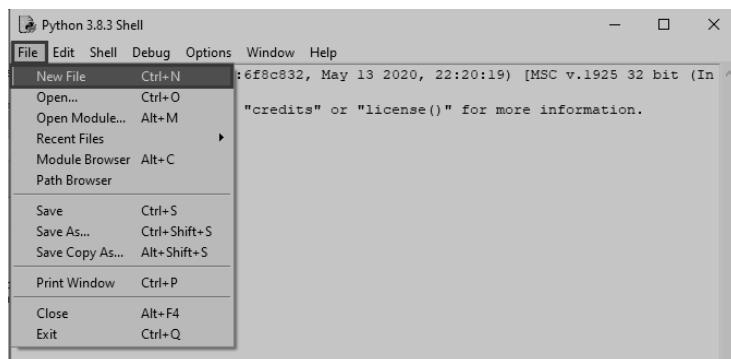
- Here, we get the message "**Hello World !**" printed on the console.
- Using a script file (Script Mode Programming)
- The interpreter prompt is best to run the single-line statements of the code. However, we cannot write the code every-time on the terminal. It is not suitable to write multiple lines of code.
- Using the script mode, we can write multiple lines code into a file which can be executed later.
- For this purpose, we need to open an editor like notepad, create a file named and save it with **.py** extension, which stands for "**Python**". Now, we will implement the above example using the script mode.

print ("hello world"); #here, we have used print() function to print the message on the console. To run this file named as first.py, we need to run the following command on the terminal.

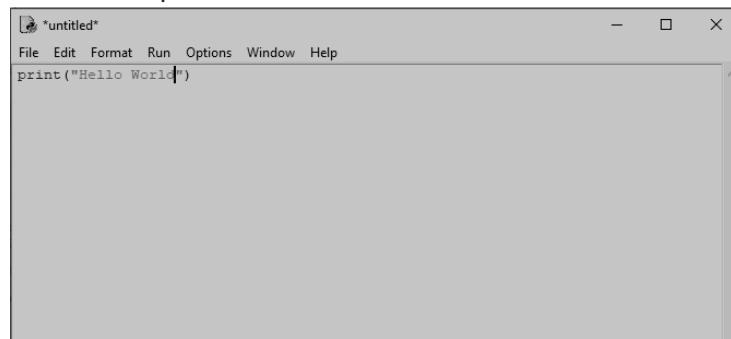


The screenshot shows the Python 3.8.3 Shell window again, but this time it is shown as a command-line interface. The title bar says "Python 3.8.3 Shell". The main window displays the Python version information and the prompt ">>>".

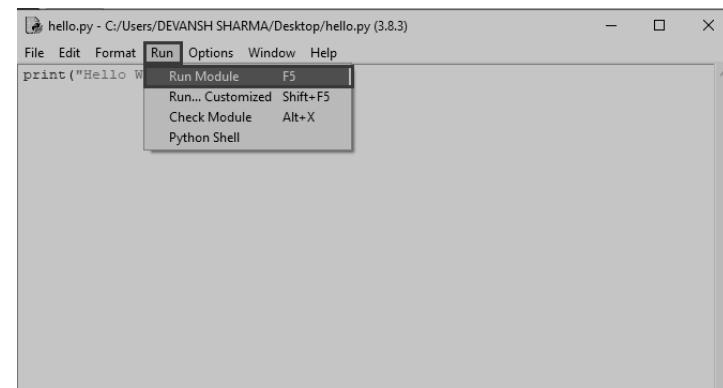
Step - 1: Open the Python interactive shell, and click "**File**" then choose "**New**", it will open a new blank script in which we can write our code.



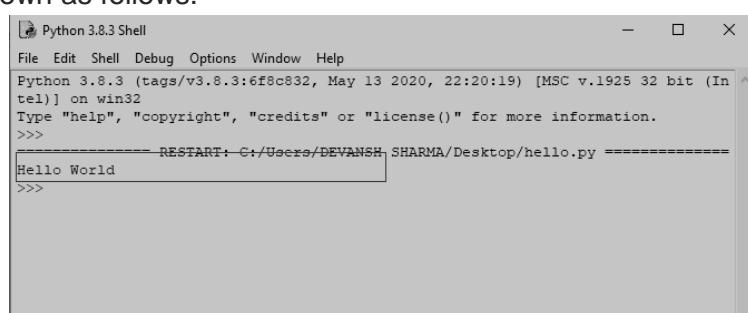
Step - 2: Now, write the code and press "**Ctrl+S**" to save the file.



Step - 3: After saving the code, we can run it by clicking "Run" or "Run Module". It will display the output to the shell.

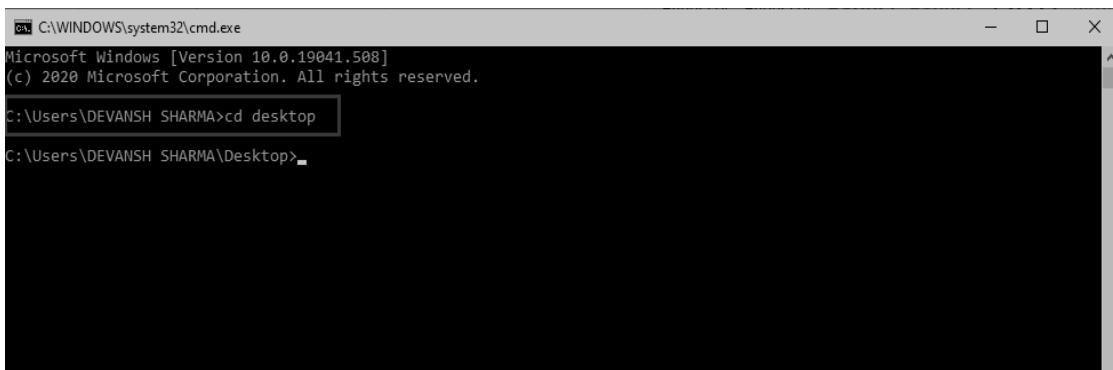


The output will be shown as follows.



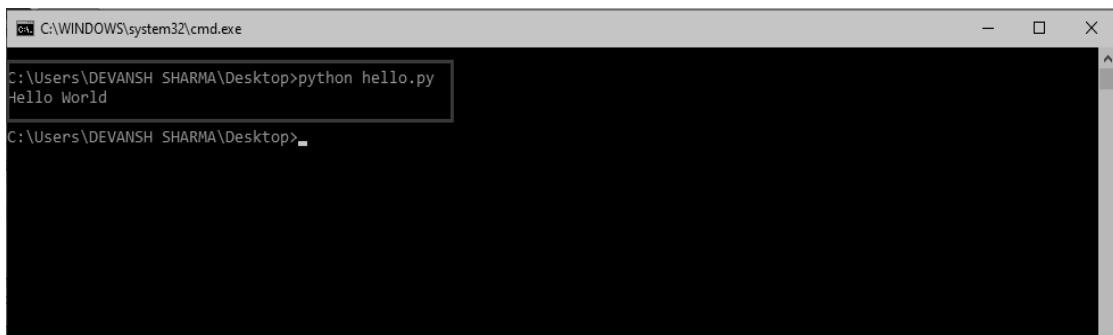
Step - 4: Apart from that, we can also run the file using the operating system terminal. But, we should be aware of the path of the directory where we have saved our file.

- Open the command line prompt and navigate to the directory.



A screenshot of a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The window shows the following text:
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.
C:\Users\DEVANSH SHARMA>cd desktop
C:\Users\DEVANSH SHARMA\Desktop>

- We need to type the **python** keyword, followed by the file name and hit enter to run the Python file.



A screenshot of a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The window shows the following text:
C:\Users\DEVANSH SHARMA\Desktop>python hello.py
Hello World
C:\Users\DEVANSH SHARMA\Desktop>

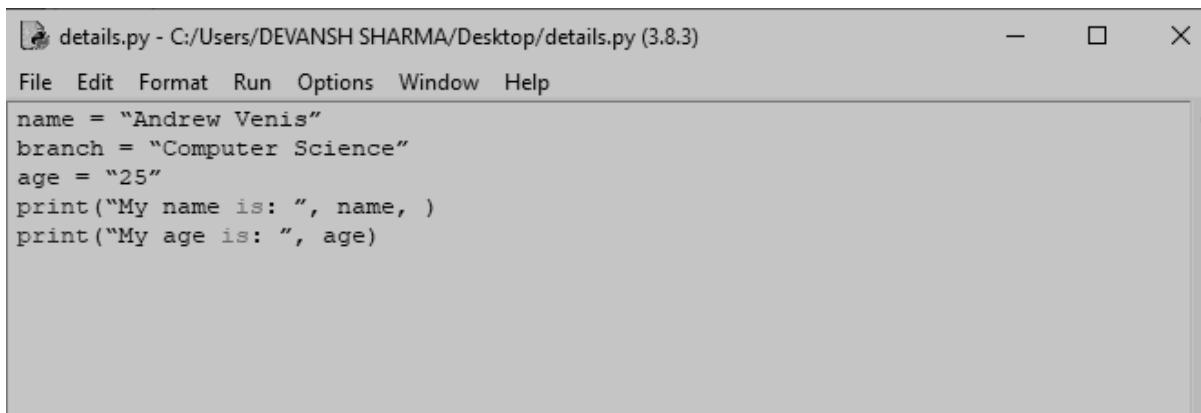
Multi-line Statements

Multi-line statements are written into the notepad like an editor and saved it with .py extension. In the following example, we have defined the execution of the multiple code lines using the Python script.

Code:

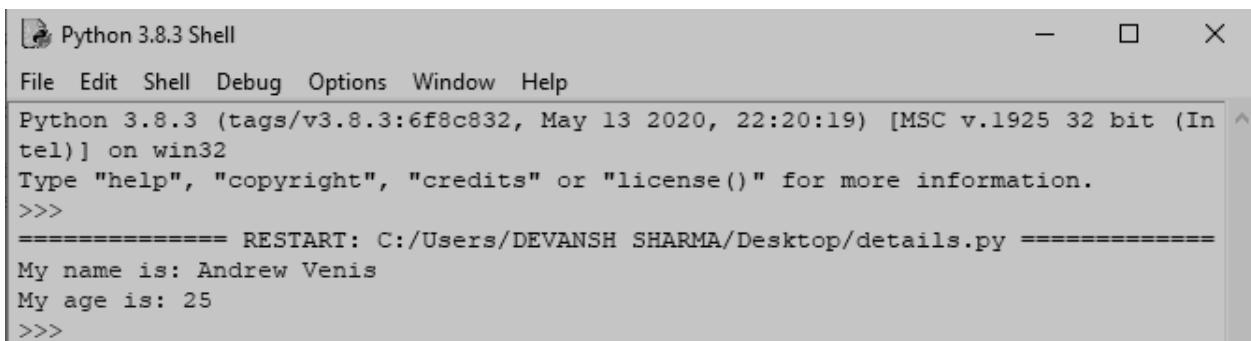
```
1. name = "Andrew Venis"
2. branch = "Computer Science"
3. age = "25"
4. print("My name is: ", name, )
5. print("My age is: ", age)
```

Script File



The screenshot shows a Windows Notepad window with the title bar 'details.py - C:/Users/DEVANSH SHARMA/Desktop/details.py (3.8.3)'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The main content area contains the following Python code:

```
name = "Andrew Venis"
branch = "Computer Science"
age = "25"
print("My name is: ", name, )
print("My age is: ", age)
```



The screenshot shows a Python 3.8.3 Shell window with the title bar 'Python 3.8.3 Shell'. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell output shows the Python version and build information, followed by a restart message and the execution of the script:

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/DEVANSH SHARMA/Desktop/details.py =====
My name is: Andrew Venis
My age is: 25
>>>
```

Pros and Cons of Script Mode

The script mode has few advantages and disadvantages as well. Let's understand the following advantages of running code in script mode.

- We can run multiple lines of code.
- Debugging is easy in script mode.
- It is appropriate for beginners and also for experts.

Let's see the disadvantages of the script mode.

- We have to save the code every time if we make any change in the code.
- It can be tedious when we run a single or a few lines of code.

How to Install PyCharm

In our first program, we have used gedit on our CentOS as an editor. On Windows, we have an alternative like notepad or notepad++ to edit the code. However, these editors are not used as IDE for python since they are unable to show the syntax related suggestions.

JetBrains provides the most popular and a widely used cross-platform IDE **PyCharm** to run the python programs.

As we have already stated, PyCharm is a cross-platform IDE, and hence it can be installed on a variety of the operating systems. In this section of the tutorial, we will cover the installation process of PyCharm on Windows.

Windows

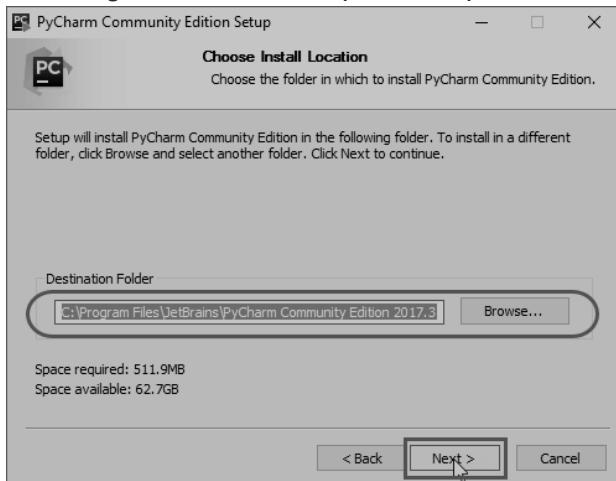
Here is a step-by-step process on how to download and install Pycharm IDE on Windows:

Step 1) To download PyCharm visit the website <https://www.jetbrains.com/pycharm/download/> and Click the “DOWNLOAD” link under the Community Section.

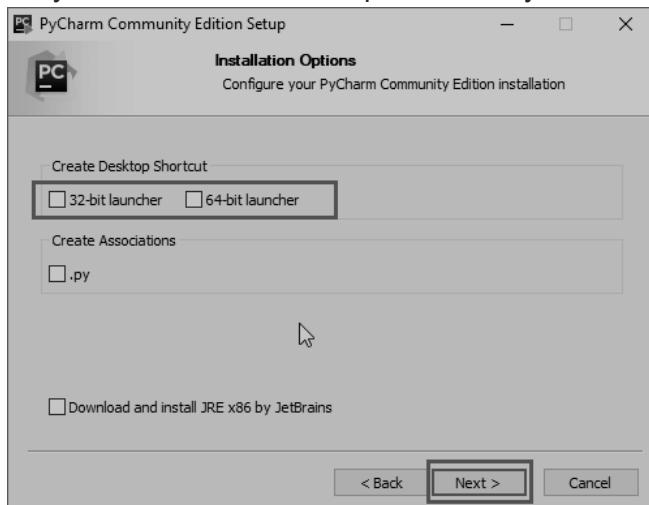
Step 2) Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.



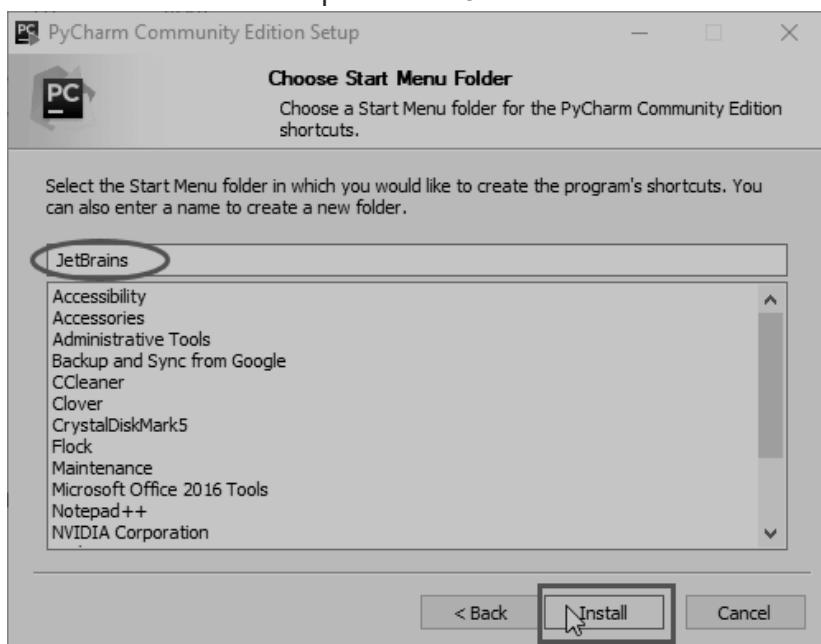
Step 3) On the next screen, Change the installation path if required. Click “Next”.



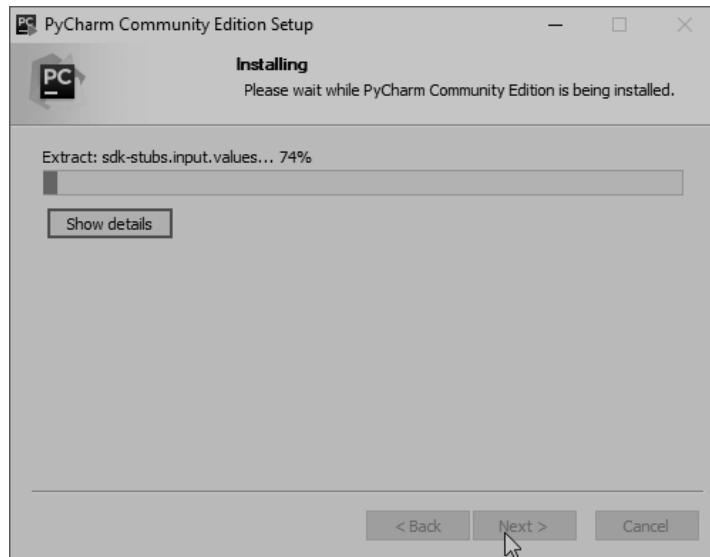
Step 4) On the next screen, you can create a desktop shortcut if you want and click on “Next”.



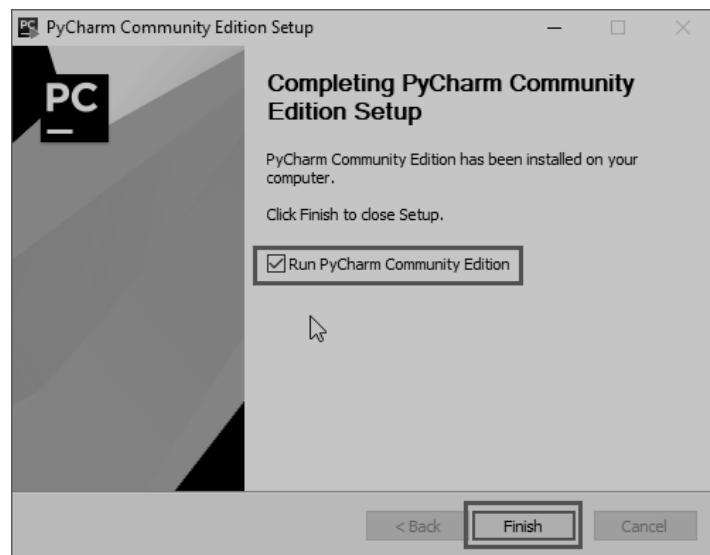
Step 5) Choose the start menu folder. Keep selected JetBrains and click on “Install”.



Step 6) Wait for the installation to finish.



Step 7) Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.



Step 8) After you click on “Finish,” the Following screen will appear.



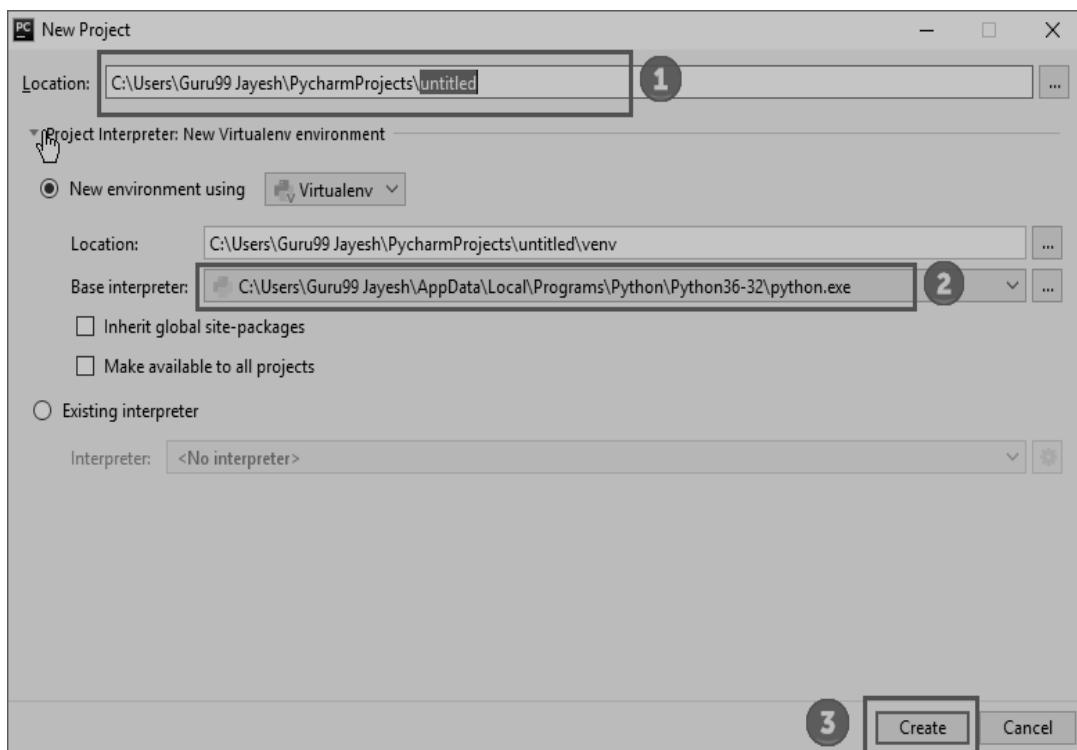
Create a program on PyCharm

Step 1) Open PyCharm Editor. You can see the introductory screen for PyCharm. To create a new project, click on “Create New Project”.

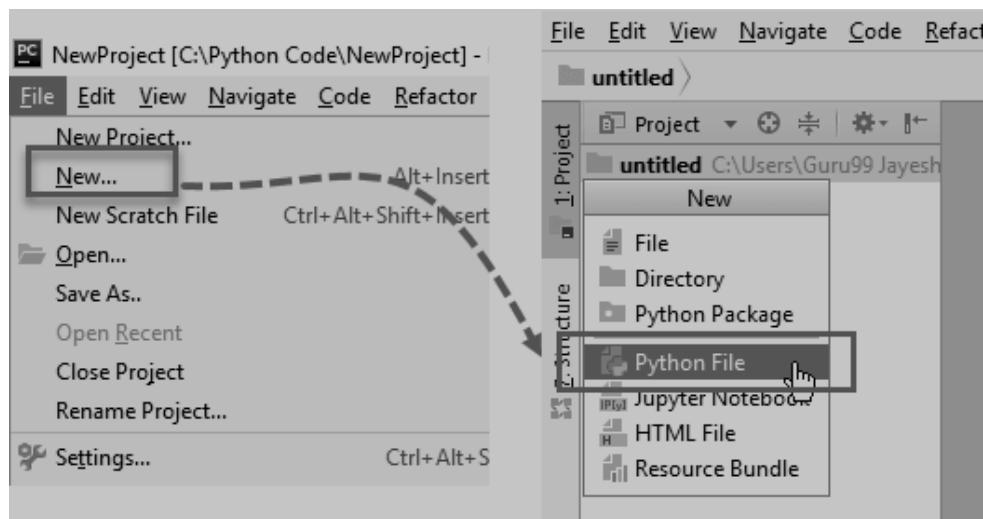


Step 2) You will need to select a location.

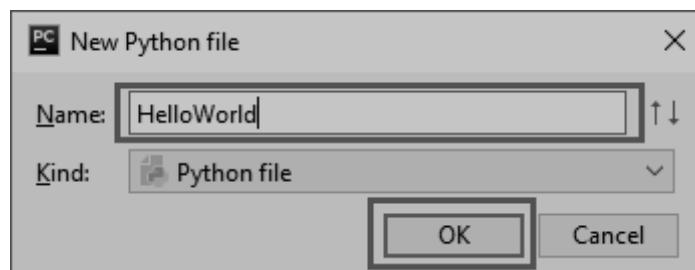
1. You can select the location where you want the project to be created. If you don't want to change location than keep it as it is but at least change the name from “untitled” to something more meaningful, like “FirstProject”.
2. PyCharm should have found the Python interpreter you installed earlier.
3. Next Click the “Create” Button.



Step 3) Now Go up to the “File” menu and select “New”. Next, select “Python File”.



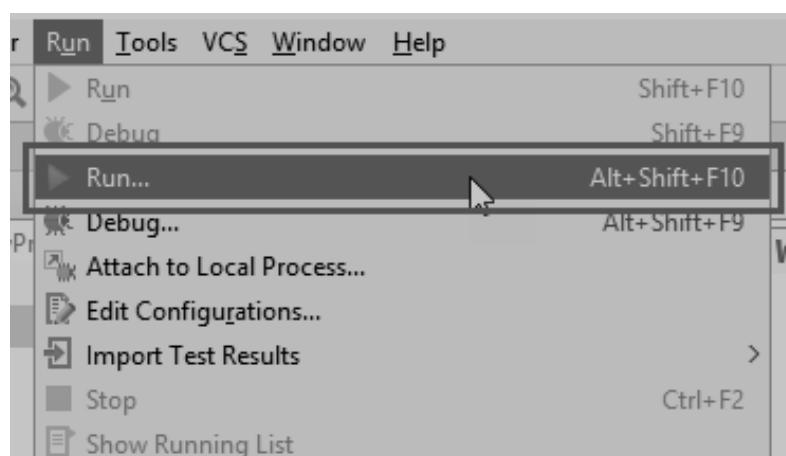
Step 4) A new pop up will appear. Now type the name of the file you want (Here we give “HelloWorld”) and hit “OK”.



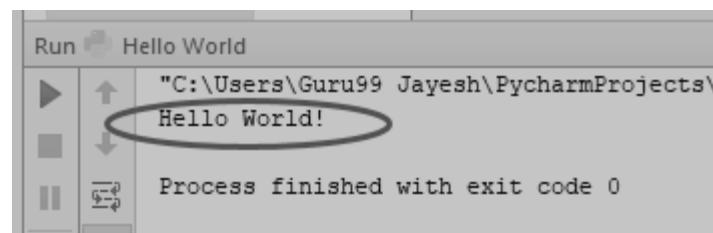
Step 5) Now type a simple program – print ('Hello World!').

```
1 print("Hello World!")
```

Step 6) Now Go up to the “Run” menu and select “Run” to run your program.



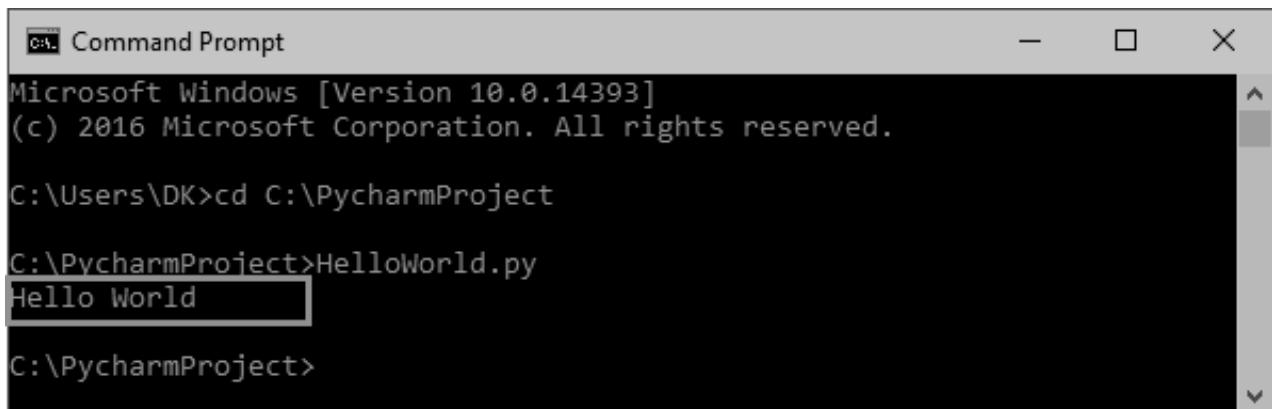
Step 7) You can see the output of your program at the bottom of the screen.



Step 8) Don't worry if you don't have Pycharm Editor installed, you can still run the code from the command prompt. Enter the correct path of a file in command prompt to run the program.



The output of the code would be:



3.3 Python Programming Fundamentals

Python Popular Frameworks and Libraries

Python has wide range of libraries and frameworks widely used in various fields such as machine learning, artificial intelligence, web applications, etc. We define some popular frameworks and libraries of Python as follows.

- **Web development (Server-side)** - Django, Flask, Pyramid, CherryPy
- **GUIs based applications** - Tk, PyGTK, PyQt, PyJs, etc.
- **Machine Learning** - TensorFlow, PyTorch, Scikit-learn, Matplotlib, Scipy, etc.
- **Mathematics** - Numpy, Pandas, etc.

Python Operators

- Operators are the symbols which perform various operations on Python objects. Python operators are the most essential to work with the Python data types.
- The operator can be defined as a symbol which is responsible for a particular operation between two operands.
- Operators are the pillars of a program on which the logic is built in a specific programming language.
- Python provides a variety of operators, which are described as follows.
 - Arithmetic operators
 - Comparison operators
 - Assignment Operators
 - Logical Operators
 - Bitwise Operators
 - Membership Operators
 - Identity Operators

Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations between two operands. It includes:

- + (addition)
- - (subtraction)
- *(multiplication)
- /(divide)
- %(reminder)
- //(floor division)
- and exponent (***) operators

Consider the following table for a detailed explanation of arithmetic operators.

Operator	Description
+ (Addition)	It is used to add two operands. For example, if $a = 20$, $b = 10 \Rightarrow a+b = 30$
- (Subtraction)	It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value results negative. For example, if $a = 20$, $b = 10 \Rightarrow a - b = 10$
/ (divide)	It returns the quotient after dividing the first operand by the second operand. For example, if $a = 20$, $b = 10 \Rightarrow a/b = 2.0$
* (Multiplication)	It is used to multiply one operand with the other. For example, if $a = 20$, $b = 10 \Rightarrow a * b = 200$
% (reminder)	It returns the remainder after dividing the first operand by the second operand. For example, if $a = 20$, $b = 10 \Rightarrow a \% b = 0$
** (Exponent)	It is an exponent operator represented as it calculates the first operand power to the second operand.
// (Floor division)	It gives the floor value of the quotient produced by dividing the two operands.

Comparison Operator

Comparison operators are used to comparing the value of the two operands and returns Boolean true or false accordingly. The comparison operators are described in the following table.

Operator	Description
==	If the value of two operands is equal, then the condition becomes true.
!=	If the value of two operands is not equal, then the condition becomes true.
<=	If the first operand is less than or equal to the second operand, then the condition becomes true.
>=	If the first operand is greater than or equal to the second operand, then the condition becomes true.
>	If the first operand is greater than the second operand, then the condition becomes true.
<	If the first operand is less than the second operand, then the condition becomes true.

Assignment Operators

The assignment operators are used to assign the value of the right expression to the left operand. The assignment operators are described in the following table.

Operator	Description
=	It assigns the value of the right expression to the left operand.
+=	It increases the value of the left operand by the value of the right operand and assigns the modified value back to left operand. For example, if $a = 10, b = 20 \Rightarrow a+ = b$ will be equal to $a = a + b$ and therefore, $a = 30$.
-=	It decreases the value of the left operand by the value of the right operand and assigns the modified value back to left operand. For example, if $a = 20, b = 10 \Rightarrow a- = b$ will be equal to $a = a - b$ and therefore, $a = 10$.
=	It multiplies the value of the left operand by the value of the right operand and assigns the modified value back to then the left operand. For example, if $a = 10, b = 20 \Rightarrow a = b$ will be equal to $a = a * b$ and therefore, $a = 200$.
%=	It divides the value of the left operand by the value of the right operand and assigns the remainder back to the left operand. For example, if $a = 20, b = 10 \Rightarrow a \% = b$ will be equal to $a = a \% b$ and therefore, $a = 0$.
=	$a^{}=b$ will be equal to $a=a^{**}b$, for example, if $a = 4, b = 2, a^{**}=b$ will assign $4^{**}2 = 16$ to a .
//=	$A//=b$ will be equal to $a = a// b$, for example, if $a = 4, b = 3, a//=b$ will assign $4//3 = 1$ to a .

Bitwise Operators

The bitwise operators perform bit by bit operation on the values of the two operands. Consider the following example.

For example,

```
if a = 7
b = 6
then, binary (a) = 0111
binary (b) = 0110
hence, a
& b = 0011
a | b = 0111
a ^ b = 0100
~ a = 1000
```

Operator	Description
& (binary and)	If both the bits at the same place in two operands are 1, then 1 is copied to the result. Otherwise, 0 is copied.
 (binary or)	The resulting bit will be 0 if both the bits are zero; otherwise, the resulting bit will be 1.
^ (binary xor)	The resulting bit will be 1 if both the bits are different; otherwise, the resulting bit will be 0.
~ (negation)	It calculates the negation of each bit of the operand, i.e., if the bit is 0, the resulting bit will be 1 and vice versa.
<< (left shift)	The left operand value is moved left by the number of bits present in the right operand.
>> (right shift)	The left operand is moved right by the number of bits present in the right operand.

Logical Operators

The logical operators are used primarily in the expression evaluation to make a decision. Python supports the following logical operators.

Operator	Description
and	If both the expression are true, then the condition will be true. If a and b are the two expressions, $a \rightarrow \text{true}$, $b \rightarrow \text{true} \Rightarrow a \text{ and } b \rightarrow \text{true}$.
or	If one of the expressions is true, then the condition will be true. If a and b are the two expressions, $a \rightarrow \text{true}$, $b \rightarrow \text{false} \Rightarrow a \text{ or } b \rightarrow \text{true}$.
not	If an expression a is true, then not (a) will be false and vice versa.

Membership Operators

Python membership operators are used to check the membership of value inside a Python data structure. If the value is present in the data structure, then the resulting value is true otherwise it returns false.

Operator	Description
in	It is evaluated to be true if the first operand is found in the second operand (list, tuple, or dictionary).
not in	It is evaluated to be true if the first operand is not found in the second operand (list, tuple, or dictionary).

Identity Operators

The identity operators are used to decide whether an element certain class or type.

Operator	Description
is	It is evaluated to be true if the reference present at both sides point to the same object.
is not	It is evaluated to be true if the reference present at both sides do not point to the same object.

Operator Precedence

The precedence of the operators is essential to find out since it enables us to know which operator should be evaluated first. The precedence table of the operators in Python is given below.

Operator	Description
**	The exponent operator is given priority over all the others used in the expression.
~ + -	The negation, unary plus, and minus.
* / % //	The multiplication, divide, modules, reminder, and floor division.
+ -	Binary plus, and minus
>> <<	Left shift. and right shift
&	Binary and.
^ 	Binary xor, and or
<= < > >=	Comparison operators (less than, less than equal to, greater than, greater than equal to).
<> == !=	Equality operators.
= %= /= //=-+=	Assignment operators
*= **=	
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Python print() Function

- The print() function displays the given object to the standard output device (screen) or to the text stream file.
- Unlike the other programming languages, Python print() function is most unique and versatile function.
- The syntax of print() function is given below.
- `print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)`
- Let's explain its parameters one by one.
 - **objects** - An object is nothing but a statement that to be printed. The * sign represents that there can be multiple statements.
 - **sep** - The **sep** parameter separates the print values. Default values is ''.
 - **end** - The **end** is printed at last in the statement.
 - **file** - It must be an object with a write(string) method.
 - **flush** - The stream or file is forcibly flushed if it is true. By default, its value is false.

Let's understand the following example.

Example - 1: Return a value

```
print("Welcome to lotus.")  
a = 10  
# Two objects are passed in print() function  
print("a =", a)  
b = a  
# Three objects are passed in print function  
print('a =', a, '= b')
```

Output:

```
Welcome to lotus.
```

```
a = 10
```

```
a = 10 = b
```

As we can see in the above output, the multiple objects can be printed in the single **print()** statement. We just need to use comma (,) to separate with each other.

Example - 2: Using sep and end argument

```
a = 10
print("a =", a, sep='dddd', end='\n\n\n')
print("a =", a, sep='0', end='$$$$$')
```

Output:

```
a =dddd10
a =010$$$$$
```

- In the first **print()** statement, we use the **sep** and **end** arguments.
- The given object is printed just after the **sep** values.
- The value of end parameter printed at the last of given object.
- As we can see that, the second **print()** function printed the result after the three black lines.

Taking Input to the User

Python provides the **input()** function which is used to take input from the user. Let's understand the following example.

Example 1-

```
name = input("Enter a name of student:")
print("The student name is: ", name)
```

Output:

```
Enter a name of student: Devansh
The student name is: Devansh
```

- By default, the **input()** function takes the string input but what if we want to take other data types as an input.
- If we want to take input as an integer number, we need to typecast the **input()** function into an integer.

Example 2-

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
print(a+b)
```

Output:

```
Enter first number: 50
Enter second number: 100
150
```

We can take any type of values using **input()** function.

Python Conditional Statements

- Decision-making in a programming language is automated using conditional statements, in which Python evaluates the code to see if it meets the specified conditions.
- The conditions are evaluated and processed as true or false. If this is found to be true, the program is run as needed.
- Python provides if and else keywords to set up logical conditions. The elif keyword is also used as conditional statement.
- Python supports the usual logical conditions from mathematics:
 - **Equals:** $a == b$
 - **Not Equals:** $a != b$
 - **Less than:** $a < b$
 - **Less than or equal to:** $a \leq b$
 - **Greater than:** $a > b$
 - **Greater than or equal to:** $a \geq b$
- These conditions can be used in several ways, most commonly in "**if statements**" and loops.
- An "if statement" is written by using the if keyword.

Example

In this example we use two variables, a and b, which are used as part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

- The **elif** keyword is python's way of saying "if the previous conditions were not true, then try this condition".

Example

```
a = 33
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
```

In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that "a and b are equal".

- The **else** keyword catches anything which isn't caught by the preceding conditions.

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

In this example a is greater than b, so the first condition is not true, also the **elif** condition is not true, so we go to the **else** condition and print to screen that "a is greater than b".

Python Loops

- Sometimes we may need to alter the flow of the program.
- The execution of a specific code may need to be repeated several numbers of times.
- For this purpose, the programming languages provide various types of loops capable of repeating some specific code several times. Consider the following tutorial to understand the statements in detail.
 - Python For Loop
 - Python While Loop

Python For Loop

- A **for** loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- This is less like the **for** keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.
- With the **for** loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Example:

Print each fruit in a fruit list:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

- The **for** loop does not require an indexing variable to set beforehand.

Looping Through a String

- Even strings are iterable objects, they contain a sequence of characters:

Example:

Loop through the letters in the word "banana":

```
for x in "banana":
    print(x)
```

The break Statement

- With the **break** statement we can stop the loop before it has looped through all the items:

Example:

Exit the loop when x is "banana":

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
    if x == "banana":
        break
```

Exit the loop when x is "banana", but this time the break comes before the print:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        break
    print(x)
```

The Continue Statement

- With the **continue** statement we can stop the current iteration of the loop, and continue with the next:

Example

Do not print banana:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        continue
    print(x)
```

Python While Loop

- With the **while** loop we can execute a set of statements as long as a condition is true.

Example

```
Print i as long as i is less than 6:
i = 1
while i < 6:
    print(i)
    i += 1
```

Note: remember to increment i, or else the loop will continue forever.

- The **while** loop requires relevant variables to be ready, in this example we need to define an indexing variable, i, which we set to 1.

The break Statement

- With the **break** statement we can stop the loop even if the while condition is true:

Example

Exit the loop when i is 3:

```
i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```

The continue Statement

- With the **continue** statement we can stop the current iteration, and continue with the next:

Example

Continue to the next iteration if I is 3:

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```

The else Statement

- With the **else** statement we can run a block of code once when the condition no longer is true:

Example

Print a message once the condition is false:

```
i = 1
while i < 6:
    print(i)
    i += 1
else:
    print("i is no longer less than 6")
```

Python Data Structures

- Data structures are referred which can hold some data together or we say that they are used to store the data in organized way.
- Python provides built-in data structures such as **list, tuple, string, dictionary, and set**. We can perform complex tasks using data structures.

Python List

- Python list holds the ordered collection of items. We can store a sequence of items in a list.
- Python list is mutable which means it can be modified after its creation.
- The items of lists are enclosed within the square bracket [] and separated by the comma. Let's see the example of list.

```
L1 = ["John", 102, "USA"]
```

```
L2 = [1, 2, 3, 4, 5, 6]
```

If we try to print the type of L1, L2, and L3 using **type()** function then it will come out to be a list.

```
print(type(L1))
```

```
print(type(L2))
```

Output:

```
<class 'list'>
```

```
<class 'list'>
```

Python Tuple

- Python Tuple is used to store the sequence of immutable Python objects.
- The tuple is similar to lists since the value of the items stored in the list can be changed, whereas the tuple is immutable, and the value of the items stored in the tuple cannot be changed.
- Tuple can be defined as follows:

Example -

```
tup = ("Apple", "Mango" , "Orange" , "Banana")
```

```
print(type(tup))
```

```
print(tup)
```

Output:

```
<class 'tuple'>
```

```
('Apple', 'Mango', 'Orange', 'Banana')
```

If we try to add new to the tuple, it will throw an error.

Example -

```
tup = ("Apple", "Mango" , "Orange" , "Banana")
```

```
tup[2] = "Papaya"
```

```
print(tup)
```

Output:

```
Traceback (most recent call last):
```

```
File "C:/Users/DEVANSH SHARMA/PycharmProjects/Hello/gamewithturtle.py", line 3, in
    tup[2] = "Papaya"
```

```
TypeError: 'tuple' object does not support item assignment
```

The above program throws an error because tuples are immutable type.

Python String

- Python string is a sequence of characters.
- It is a collection of the characters surrounded by single quotes, double quotes, or triple quotes.
- It can also define as collection of the Unicode characters. We can create a string as follows.

Example

```
# Creating string using double quotes
str1 = "Hi Python"
print(str1)

# Creating string using single quotes
str1 = 'Hi Python'
print(str1)

# Creating string using triple quotes
str1 = """Hi Python"""
print(str1)
```

Output:

```
Hi Python
Hi Python
Hi Python
```

- Python doesn't support the character data-type. A single character written as 'p' is treated as a string of length 1.
- Strings are also immutable. We can't change after it is declared.

Dictionaries

- Python Dictionary is a most efficient data structure and used to store the large amount of data.
- It stores the data in the key-value pair format. Each value is stored corresponding to its key.
- Keys must be a unique and value can be any type such as integer, list, tuple, etc.
- It is a mutable type; we can reassign after its creation. Below is the example of creating dictionary in Python.

Example –

```
employee = {"Name": "John", "Age": 29, "salary":250000,"Company":"GOOGLE"}
print(type(employee))
print("printing Employee data .... ")
print(employee)
```

Output:

```
<class 'dict'>
Printing Employee data ....
{'Name': 'John', 'Age': 29, 'salary': 250000, 'Company': 'GOOGLE'}
```

The empty curly braces {} are used to create empty dictionary.

Python Sets

- A Python set is a collection of unordered elements.
- Each element in set must be unique and immutable.
- Sets are mutable which means we can modify anytime throughout the program.
- Let's understand the example of creating set in Python.

Example -

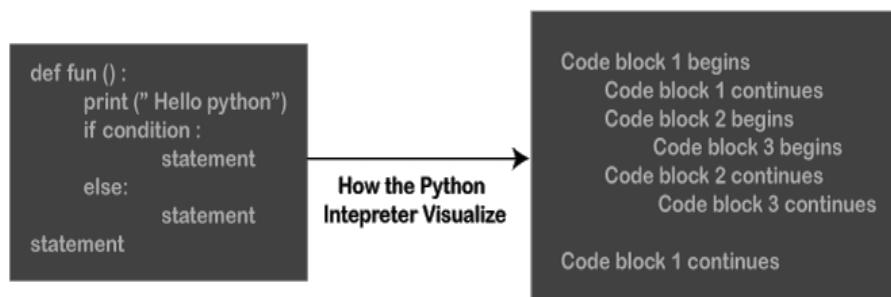
```
# Creating Set
Month = {"January", "February", "March", "April", "May", "June", "July"}
print(Month)
print(type(Month))
```

Output:

```
{'March', 'July', 'April', 'May', 'June', 'February', 'January'}
<class 'set'>
```

Indentation in Python

- Indentation is the most significant concept of the Python programming language.
- Improper use of indentation will end up "**IndentationError**" in our code.
- Indentation is adding whitespaces before the statement when it is needed.
- Without indentation Python doesn't know which statement to be executed to next.
- Indentation also defines which statements belong to which block.
- If there is no indentation or improper indentation, it will display "**IndentationError**" and interrupt our code.



- Python indentation defines the particular group of statements belongs to the particular block.
- The programming languages such as C, C++, java use the curly braces {} to define code blocks.
- In Python, statements that are the same level to the right belong to the same block.
- We can use four whitespaces to define indentation. Let's see the following lines of code.

Example

```
list1 = [1, 2, 3, 4, 5]
for i in list1:
    print(i)
    if i==4:
        break
print("End of for loop")
```

Output:

```
1
2
3
4
End of for loop
```

Explanation:

In the above code, for loop has a code blocks and if the statement has its code block inside for loop. Both indented with four whitespaces. The last **print()** statement is not indented; that's means it doesn't belong to for loop.

Comment in Python

- Comments are essential for defining the code and help us and other to understand the code.
- By looking the comment, we can easily understand the intention of every line that we have written in code.
- We can also find the error very easily, fix them, and use in other applications.
- In Python, we can apply comments using the # hash character.
- The Python interpreter entirely ignores the lines followed by a hash character.
- A good programmer always uses the comments to make code under stable.
- Let's see the following example of a comment.

```
name = "Thomas" # Assigning string value to the name variable
```

We can add comment in each line of the Python code.

```
Fees = 10000 # defining course fees is 10000
```

```
Fees = 20000 # defining course fees is 20000
```

It is good idea to add code in any line of the code section of code whose purpose is not obvious. This is a best practice to learn while doing the coding.

Types of Comment

Python provides the facility to write comments in two ways- single line comment and multi-line comment.

Single-Line Comment - Single-Line comment starts with the hash # character followed by text for further explanation.

```
# defining the marks of a student
```

```
Marks = 90
```

We can also write a comment next to a code statement. Consider the following example.

```
Name = "James" # the name of a student is James
```

```
Marks = 90      # defining student's marks
```

```
Branch = "Computer Science" # defining student branch
```

Multi-Line Comments - Python doesn't have explicit support for multi-line comments but we can use hash # character to the multiple lines. **For example -**

```
# we are defining for loop
```

```
# To iterate the given list.
```

```
# run this code.
```

We can also use another way.

```
'''
```

This **is** an example

Of multi-line comment

Using triple-quotes

```
'''
```

Python Identifiers

- Python identifiers refer to a name used to identify a variable, function, module, class, module or other objects. There are few rules to follow while naming the Python Variable.
 - A variable name must start with either an English letter or underscore (_).
 - A variable name cannot start with the number.
 - Special characters are not allowed in the variable name.
 - The variable's name is case sensitive.

Let's understand the following example.

```
number = 10
print(num)
_a = 100
print(_a)
x_y = 1000
print(x_y)
```

Output:

```
10
100
1000
```

Python Variables

- Variable is a name that is used to refer to memory location.
- Python variable is also known as an identifier and used to hold value.
- In Python, we don't need to specify the type of variable because Python is a infer language and smart enough to get variable type.
- Variable names can be a group of both the letters and digits, but they have to begin with a letter or an underscore.
- It is recommended to use lowercase letters for the variable name. Rahul and rahul both are two different variables.

Identifier Naming

Variables are the example of identifiers. An Identifier is used to identify the literals used in the program. The rules to name an identifier are given below.

- The first character of the variable must be an alphabet or underscore (_).
- All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore, or digit (0-9).
- Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, *).
- Identifier name must not be similar to any keyword defined in the language.
- Identifier names are case sensitive; for example, my name, and MyName is not the same.
- Examples of valid identifiers: a123, _n, n_9, etc.
- Examples of invalid identifiers: 1a, n%4, n 9, etc.

Declaring Variable and Assigning Values

- Python does not bind us to declare a variable before using it in the application.
- It allows us to create a variable at the required time.
- We don't need to declare explicitly variable in Python.
- When we assign any value to the variable, that variable is declared automatically.
- The equal (=) operator is used to assign value to a variable.

Object References

- It is necessary to understand how the Python interpreter works when we declare a variable.
- The process of treating variables is somewhat different from many other programming languages.
- Python is the highly object-oriented programming language; that's why every data item belongs to a specific type of class. Consider the following example.

```
print("John")
```

Output:

John

The Python object creates an integer object and displays it to the console. In the above print statement, we have created a string object. Let's check the type of it using the Python built-in **type()** function.

```
Type("John")
```

Output:

<class 'str'>

- In Python, variables are a symbolic name that is a reference or pointer to an object. The variables are used to denote objects by that name.

Let's understand the following example:

a = 50



- In the above image, the variable a refers to an integer object.
- Suppose we assign the integer value 50 to a new variable b.

a = 50

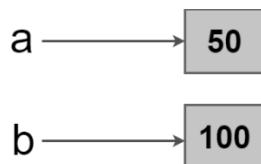
b = a



- The variable b refers to the same object that a points to because Python does not create another object.
- Let's assign the new value to b. Now both variables will refer to the different objects.

a = 50

b = 100



Python manages memory efficiently if we assign the same variable to two different values.

Object Identity

- In Python, every created object identifies uniquely in Python.
- Python provides the guarantee that no two objects will have the same identifier.
- The built-in **id()** function, is used to identify the object identifier. Consider the following example.

```

a = 50
b = a
print(id(a))
print(id(b))
# Reassigned variable a
a = 500
print(id(a))
Output:
140734982691168
140734982691168
2822056960944
  
```

We assigned the **b = a**, **a** and **b** both point to the same object. When we checked by the **id()** function it returned the same number. We reassign **a** to 500; then it referred to the new object identifier.

Variable Names

- We have already discussed how to declare the valid variable.
- Variable names can be any length can have uppercase, lowercase (A to Z, a to z), the digit (0-9), and underscore character(_).
- Consider the following example of valid variables names.

```
name = "Devansh"  
age = 20  
marks = 80.50  
print(name)  
print(age)  
print(marks)
```

Output:

```
Devansh  
20  
80.5
```

Consider the following valid variables name.

```
name = "A"  
Name = "B"  
naMe = "C"  
NAME = "D"  
n_a_m_e = "E"  
_name = "F"  
name_ = "G"  
_name_ = "H"  
na56me = "I"  
print(name,Name,naMe,NAME,n_a_m_e, NAME, n_a_m_e, _name, name_,_name, na56me)
```

Output:

```
A B C D E D E F G F I
```

The multi-word keywords can be created by the following method.

- **Camel Case** - In the camel case, each word or abbreviation in the middle of begins with a capital letter. There is no intervention of whitespace. For example - nameOfStudent, valueOfVaraible, etc.
- **Pascal Case** - It is the same as the Camel Case, but here the first word is also capital. For example - NameOfStudent, etc.
- **Snake Case** - In the snake case, Words are separated by the underscore. For example - name_of_student, etc.

Multiple Assignment

- Python allows us to assign a value to multiple variables in a single statement, which is also known as multiple assignments.
- We can apply multiple assignments in two ways, either by assigning a single value to multiple variables or assigning multiple values to multiple variables. Consider the following example.

1. Assigning single value to multiple variables**Eg:**

```
x=y=z=50  
print(x)  
print(y)  
print(z)
```

Output:

```
50  
50  
50
```

2. Assigning multiple values to multiple variables:**Eg:**

```
a,b,c=5,10,15  
print a  
print b  
print c
```

Output:

```
5  
10  
15
```

The values will be assigned in the order in which variables appear.

Python Variable Types

There are two types of variables in Python - Local variable and Global variable. Let's understand the following variables.

Local Variable

Local variables are the variables that declared inside the function and have scope within the function. Let's understand the following example.

Example -

```
# Declaring a function
def add():
    # Defining local variables. They has scope only within a function
    a = 20
    b = 30
    c = a + b
    print("The sum is:", c)

# Calling a function
add()
```

Output:

The sum is: 50

Explanation:

In the above code, we declared a function named **add()** and assigned a few variables within the function. These variables will be referred to as the **local variables** which have scope only inside the function. If we try to use them outside the function, we get a following error.

```
add()
```

```
# Accessing local variable outside the function
print(a)
```

Output:

The sum is: 50

```
    print(a)
```

NameError: name 'a' is not defined

We tried to use local variable outside their scope; it threw the **NameError**.

Global Variables

- Global variables can be used throughout the program, and its scope is in the entire program.
- We can use global variables inside or outside the function.
- A variable declared outside the function is the global variable by default.
- Python provides the **global** keyword to use global variable inside the function.
- If we don't use the **global** keyword, the function treats it as a local variable.
- Let's understand the following example.

Example -

```
# Declare a variable and initialize it
```

```
x = 101
```

```
# Global variable in function
```

```
def mainFunction():
```

```
    # printing a global variable
```

```
    global x
```

```
    print(x)
```

```
    # modifying a global variable
```

```
    x = 'Welcome To lotus'
```

```
    print(x)
```

```
mainFunction()
```

```
print(x)
```

Output:

```
101
```

```
Welcome To lotus
```

```
Welcome To lotus
```

Explanation:

In the above code, we declare a global variable **x** and assign a value to it. Next, we defined a function and accessed the declared variable using the **global** keyword inside the function. Now we can modify its value. Then, we assigned a new string value to the variable **x**.

Now, we called the function and proceeded to print **x**. It printed the as newly assigned value of **x**.

Delete a variable

We can delete the variable using the **del** keyword. The syntax is given below.

Syntax -

```
del <variable_name>
```

In the following example, we create a variable **x** and assign value to it. We deleted variable **x**, and print it, we get the error "**variable x is not defined**". The variable **x** will no longer use in future.

Example -

```
# Assigning a value to x
x = 6
print(x)
# deleting a variable.
del x
print(x)
```

Output:

6

Maximum Possible Value of an Integer in Python

Unlike the other programming languages, Python doesn't have long int or float data types. It treats all integer values as an **int** data type. Here, the question arises. What is the maximum possible value can hold by the variable in Python? Consider the following example.

Example -

```
# A Python program to display that we can store
# large numbers in Python
a = 100000000000000000000000000000000000000000000000000000000000000
a = a + 1
print(type(a))
print (a)
```

Output:

```
<class 'int'>
100000000000000000000000000000000000000000000000000000000000001
```

- As we can see in the above example, we assigned a large integer value to variable x and checked its type.
- It printed class `<int>` not long int. Hence, there is no limitation number by bits and we can expand to the limit of our memory.
- Python doesn't have any special data type to store larger numbers.

Print Single and Multiple Variables in Python

We can print multiple variables within the single print statement. Below are the example of single and multiple printing values.

Example - 1 (Printing Single Variable)

```
# printing single value  
a = 5  
print(a)  
print((a))
```

Output:

```
5  
5
```

Example - 2 (Printing Multiple Variables)

```
a = 5  
b = 6  
# printing multiple variables  
print(a,b)  
# separate the variables by the comma  
Print(1, 2, 3, 4, 5, 6, 7, 8)
```

Output:

```
5 6  
1 2 3 4 5 6 7 8
```

Python Data Types

- Variables can hold values, and every value has a data-type.
- Python is a dynamically typed language; hence we do not need to define the type of the variable while declaring it.
- The interpreter implicitly binds the value with its type.

a = 5

- The variable **a** holds integer value five and we did not define its type.
- Python interpreter will automatically interpret variables **a** as an integer type.
- Python enables us to check the type of the variable used in the program.
- Python provides us the **type()** function, which returns the type of the variable passed.

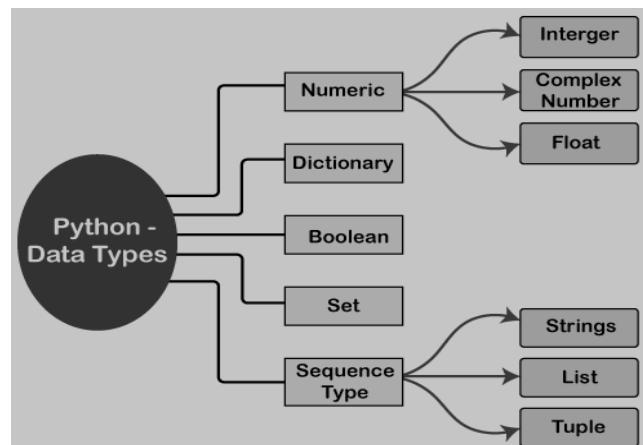
Consider the following example to define the values of different data types and checking its type.

```
a=10  
b="Hi Python"  
c = 10.5  
print(type(a))  
print(type(b))  
print(type(c))  
Output:  
<type 'int'>  
<type 'str'>  
<type 'float'>
```

Standard Data Types

- A variable can hold different types of values. For example, a person's name must be stored as a string whereas its id must be stored as an integer.
- Python provides various standard data types that define the storage method on each of them.
- The data types defined in Python are given below.

1. Numbers
2. Sequence Type
3. Boolean
4. Set
5. Dictionary



Numbers

- Number stores numeric values.
- The integer, float, and complex values belong to a Python Numbers data-type.
- Python provides the **type()** function to know the data-type of the variable.
- Similarly, the **isinstance()** function is used to check an object belongs to a particular class.
- Python creates Number objects when a number is assigned to a variable. For example;

```

a = 5
print("The type of a", type(a))
      b = 40.5
print("The type of b", type(b))
c = 1+3j
print("The type of c", type(c))
print(" c is a complex number", isinstance(1+3j,complex))
  
```

Output:

The type of a <class 'int'>
 The type of b <class 'float'>
 The type of c <class 'complex'>
 c is complex number: True

Python supports three types of numeric data.

1. **Int** - Integer value can be any length such as integers 10, 2, 29, -20, -150 etc. Python has no restriction on the length of an integer. Its value belongs to **int**
2. **Float** - Float is used to store floating-point numbers like 1.9, 9.902, 15.2, etc. It is accurate upto 15 decimal points.
3. **Complex** - A complex number contains an ordered pair, i.e., $x + iy$ where x and y denote the real and imaginary parts, respectively. The complex numbers like $2.14j$, $2.0 + 2.3j$, etc.

Sequence Type

String

- The string can be defined as the sequence of characters represented in the quotation marks.
- In Python, we can use single, double, or triple quotes to define a string.
- String handling in Python is a straightforward task since Python provides built-in functions and operators to perform operations in the string.
- In the case of string handling, the operator `+` is used to concatenate two strings as the operation `"hello"+ "python"` returns `"hello python"`.
- The operator `*` is known as a repetition operator as the operation `"Python" *2` returns 'Python Python'.

The following example illustrates the string in Python.

Example – 1

```
str = "string using double quotes"  
print(str)  
s = """A multiline  
string""  
print(s)
```

Output:

```
string using double quotes  
A multiline  
string
```

Consider the following example of string handling.

Example - 2

```
str1 = 'hello Lotus' #string str1
str2 = ' how are you' #string str2
print (str1[0:2]) #printing first two character using slice operator
print (str1[4]) #printing 4th character of the string
print (str1*2) #printing the string twice
print (str1 + str2) #printing the concatenation of str1 and str2
```

Output:

```
he
o
hello lotushello lotus
hello lotus how are you
```

Boolean

- Boolean type provides two built-in values, True and False.
- These values are used to determine the given statement true or false.
- It denotes by the class bool. True can be represented by any non-zero value or 'T' whereas false can be represented by the 0 or 'F'.
- Consider the following example.

```
# Python program to check the Boolean type
print(type(True))
print(type(False))
print(false)
```

Output:

```
<class 'bool'>
<class 'bool'>
NameError: name 'false' is not defined
```

Set

- Python Set is the unordered collection of the data type.
- It is iterable, mutable(can modify after creation), and has unique elements.
- In set, the order of the elements is undefined; it may return the changed sequence of the element.
- The set is created by using a built-in function **set()**, or a sequence of elements is passed in the curly braces and separated by the comma.
- It can contain various types of values. Consider the following example.

```
# Creating Empty set
set1 = set()
set2 = {'James', 2, 3,'Python'}
#printing Set value
print(set2)
# Adding element to the set
set2.add(10)
```

```
print(set2)
#Removing element from the set
set2.remove(2)
print(set2)
```

Output:

```
{3, 'Python', 'James', 2}
{'Python', 'James', 3, 2, 10}
{'Python', 'James', 3, 10}
```

Dictionary

- Dictionary is an unordered set of a key-value pair of items.
- It is like an associative array or a hash table where each key stores a specific value.
- Key can hold any primitive data type, whereas value is an arbitrary Python object.
- The items in the dictionary are separated with the comma (,) and enclosed in the curly braces {}.

Consider the following example.

```
d = {1:'Jimmy', 2:'Alex', 3:'john', 4:'mike'}
# Printing dictionary
print (d)

# Accesing value using keys
print("1st name is "+d[1])
print("2nd name is "+ d[4])

print (d.keys())
print (d.values())
```

Output:

```
1st name is Jimmy
2nd name is mike
{1: 'Jimmy', 2: 'Alex', 3: 'john', 4: 'mike'}
dict_keys([1, 2, 3, 4])
dict_values(['Jimmy', 'Alex', 'john', 'mike'])
```

List

- Python Lists are similar to arrays in C.
- However, the list can contain data of different types.
- The items stored in the list are separated with a comma (,) and enclosed within square brackets [].
- We can use slice [:] operators to access the data of the list.
- The concatenation operator (+) and repetition operator (*) works with the list in the same way as they were working with the strings.

Consider the following example.

```
list1 = [1, "hi", "Python", 2]
#Checking type of given list

print(type(list1))

#Printing the list1
print (list1)

# List slicing
print (list1[3:])

# List slicing
print (list1[0:2])

# List Concatenation using + operator
print (list1 + list1)

# List repetition using * operator
print (list1 * 3)
```

Output:

```
[1, 'hi', 'Python', 2]
[2]
[1, 'hi']
[1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2]
[1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2]
```

Tuple

- A tuple is similar to the list in many ways.
- Like lists, tuples also contain the collection of the items of different data types.
- The items of the tuple are separated with a comma (,) and enclosed in parentheses () .
- A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.

Let's see a simple example of the tuple.

```
tup = ("hi", "Python", 2)
# Checking type of tup
print(type(tup))

#printing the tuple
print(tup)

# Tuple slicing
print(tup[1:])
print(tup[0:1])
```

```
# Tuple repetition using * operator
print(tup * 3)

# Adding value to tup. It will throw an error.
t[2] = "hi"
```

Output:

```
<class 'tuple'>
('hi', 'Python', 2)
('Python', 2)
('hi',)
('hi', 'Python', 2, 'hi', 'Python', 2)
('hi', 'Python', 2, 'hi', 'Python', 2, 'hi', 'Python', 2)
```

Traceback (most recent call last):

```
File "main.py", line 14, in <module>
    t[2] = "hi";
TypeError: 'tuple' object does not support item assignment
```

Python Keywords

- Python Keywords are special reserved words that convey a special meaning to the compiler/interpreter.
- Each keyword has a special meaning and a specific operation.
- These keywords can't be used as a variable. Following is the List of Python Keywords.

None	break	except	in	raise
False	await	else	import	pass
and	continue	for	lambda	try
True	class	finally	is	return
as	def	from	nonlocal	while
async	elif	if	not	with
assert	del	global	or	yield

Consider the following explanation of keywords.

1. **True** - It represents the Boolean true, if the given condition is true, then it returns "True". Non-zero values are treated as true.
2. **False** - It represents the Boolean false; if the given condition is false, then it returns "False". Zero value is treated as false
3. **None** - It denotes the null value or void. An empty list or Zero can't be treated as **None**.
4. **and** - It is a logical operator. It is used to check the multiple conditions. It returns true if both conditions are true. Consider the following truth table.

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

5. **or** - It is a logical operator in Python. It returns true if one of the conditions is true. Consider the following truth table.

A	B	A and B
True	True	True
True	False	True
False	True	True
False	False	False

6. **not** - It is a logical operator and inverts the truth value. Consider the following truth table.

A	Not A
True	False
False	True

7. assert - This keyword is used as the debugging tool in Python. It checks the correctness of the code. It raises an **AssertionError** if found any error in the code and also prints the message with an error.

Example:

```
a = 10  
b = 0  
print('a is dividing by Zero')  
assert b != 0  
print(a / b)
```

Output:

```
a is dividing by Zero  
Runtime Exception:
```

Traceback (most recent call last):
File "/home/40545678b342ce3b70beb1224bed345f.py", line 4, in
 assert b != 0, "Divide by 0 error"

8. def - This keyword is used to declare the function in Python. If followed by the function name.

```
def my_func(a,b):  
    c = a+b  
    print(c)  
my_func(10,20)
```

Output:

```
30
```

9. class - It is used to represents the class in Python. The class is the blueprint of the objects. It is the collection of the variable and methods. Consider the following class.

```
class Myclass:  
    #Variables.....  
    def function_name(self):  
        #statements.....
```

10. continue - It is used to stop the execution of the current iteration. Consider the following example.

```
a = 0  
while a < 4:  
    a += 1  
    if a == 2:  
        continue  
    print(a)  
AssertionError  
divide by 0 error  
Output:  
1  
3  
4
```

11. break - It is used to terminate the loop execution and control transfer to the end of the loop. Consider the following example.

Example

```
for i in range(5):
    if(i==3):
        break
    print(i)
print("End of execution")
```

Output:

```
0
1
2
```

End of execution

12. If - It is used to represent the conditional statement. The execution of a particular block is decided by if statement. Consider the following example.

Example

```
i = 18
if (1 < 12):
    print("I am less than 18")
```

Output:

I am less than 18

13. else - The else statement is used with the if statement. When if statement returns false, then else block is executed. Consider the following example.

Example:

```
n = 11
if(n%2 == 0):
    print("Even")
else:
    print("odd")
```

Output:

Odd

14. elif - This Keyword is used to check the multiple conditions. It is short for else-if. If the previous condition is false, then check until the true condition is found. Consider the following example.

Example:

```
marks = int(input("Enter the marks:"))
if(marks>=90):
    print("Excellent")
elif(marks<90 and marks>=75):
    print("Very Good")
elif(marks<75 and marks>=60):
    print("Good")
else:
    print("Average")
```

Output:

Enter the marks:85

Very Good

15. del - It is used to delete the reference of the object. Consider the following example.

Example:

```
a=10  
b=12  
del a  
print(b)  
# a is no longer exist  
print(a)
```

Output:

```
12  
NameError: name 'a' is not defined
```

16. try, except - The try-except is used to handle the exceptions. The exceptions are run-time errors.

Consider the following example.

Example:

```
a = 0  
try:  
    b = 1/a  
except Exception as e:  
    print(e)
```

Output:

```
division by zero
```

17. raise - The raise keyword is used to through the exception forcefully. Consider the following example.

Example

```
a = 5  
if (a>2):  
    raise Exception('a should not exceed 2 ')
```

Output:

```
Exception: a should not exceed 2
```

18. finally - The finally keyword is used to create a block of code that will always be executed no matter the else block raises an error or not. Consider the following example.

Example:

```
a=0  
b=5  
try:  
    c = b/a  
    print(c)  
except Exception as e:  
    print(e)  
finally:  
    print('Finally always executed')
```

Output:

division by zero
Finally always executed

19. for, while - Both keywords are used for iteration. The for keyword is used to iterate over the sequences (list, tuple, dictionary, string). A while loop is executed until the condition returns false. Consider the following example.

Example: For loop

```
list = [1,2,3,4,5]  
for i in list:  
    print(i)
```

Output:

1
2
3
4
5

Example: While loop

1. a = 0
2. while(a<5):
3. print(a)
4. a = a+1

Output:

0
1
2
3
4

20. import - The import keyword is used to import modules in the current Python script. The module contains a runnable Python code.

Example:

```
1. import math  
2. print(math.sqrt(25))
```

Output:

5

21. from - This keyword is used to import the specific function or attributes in the current Python script.

Example:

```
1. from math import sqrt  
2. print(sqrt(25))
```

Output:

5

22. as - It is used to create a name alias. It provides the user-defined name while importing a module.

Example:

```
1. import calendar as cal  
2. print(cal.month_name[5])
```

Output:

May

23. pass - The pass keyword is used to execute nothing or create a placeholder for future code. If we declare an empty class or function, it will throw an error, so we use the pass keyword to declare an empty class or function.

Example:

```
class my_class:  
    pass
```

```
def my_func():  
    pass
```

24. return - The return keyword is used to return the result value or none to called function.

Example:

```
def sum(a,b):  
    c = a+b  
    return c  
  
print("The sum is:",sum(25,15))
```

Output:

The sum is: 40

25. is - This keyword is used to check if the two-variable refers to the same object. It returns the true if they refer to the same object otherwise false. Consider the following example.

Example

```
x = 5
```

```
y = 5
```

```
a = []
```

```
b = []
```

```
print(x is y)
```

```
print(a is b)
```

Output:

True

False

Note: A mutable data-types do not refer to the same object.

26. global - The global keyword is used to create a global variable inside the function. Any function can access the global. Consider the following example.

Example

```
def my_func():
```

```
    global a
```

```
    a = 10
```

```
    b = 20
```

```
    c = a+b
```

```
    print(c)
```

```
my_func()
```

```
def func():
```

```
    print(a)
```

```
func()
```

Output:

30

10

27. nonlocal - The nonlocal is similar to the global and used to work with a variable inside the nested function(function inside a function). Consider the following example.

Example

```
def outside_function():
    a = 20
    def inside_function():
        nonlocal a
        a = 30
        print("Inner function: ",a)
    inside_function()
    print("Outer function: ",a)
outside_function()
```

Output:

```
Inner function: 30
Outer function: 30
```

28. lambda - The lambda keyword is used to create the anonymous function in Python. It is an inline function without a name. Consider the following example.

Example

```
a = lambda x: x**2
for i in range(1,6):
    print(a(i))
```

Output:

```
1
4
9
16
25
```

29. yield - The yield keyword is used with the Python generator. It stops the function's execution and returns value to the caller. Consider the following example.

Example

```
def fun_Generator():
    yield 1
    yield 2
    yield 3
# Driver code to check above generator function
for value in fun_Generator():
    print(value)
```

Output:

```
1
2
3
```

30. with - The with keyword is used in the exception handling. It makes code cleaner and more readable. The advantage of using with, we don't need to call close(). Consider the following example.

Example

```
with open('file_path', 'w') as file:  
    file.write('hello world !')
```

31. None - The None keyword is used to define the null value. It is remembered that None does not indicate 0, false, or any empty data-types. It is an object of its data type, which is Consider the following example.

Example:

```
def return_none():  
    a = 10  
    b = 20  
    c = a + b  
x = return_none()  
print(x)
```

Output:

None

Python Literals

Python Literals can be defined as data that is given in a variable or constant.

Python supports the following literals:

1. String literals:

String literals can be formed by enclosing a text in the quotes. We can use both single as well as double quotes to create a string.

Example:

"Aman" , '12345'

Types of Strings:

There are two types of Strings supported in Python:

a) Single-line String- Strings that are terminated within a single-line are known as Single line Strings.

Example:

```
text1='hello'
```

b) Multi-line String - A piece of text that is written in multiple lines is known as multiple lines string.

There are two ways to create multiline strings:

1) Adding back slash at the end of each line.

Example:

```
text1='hello\
```

```
user'
```

```
print(text1)
```

```
'hellouser'
```

2) Using triple quotation marks:-

Example:

```
str2="""welcome
```

```
to
```

```
SSSIT""
```

```
print str2
```

Output:

```
welcome
```

```
to
```

```
SSSIT
```

2. Numeric literals

Example - Numeric Literals

```
x = 0b10100 #Binary Literals
```

```
y = 100 #Decimal Literal
```

```
z = 0o215 #Octal Literal
```

```
u = 0x12d #Hexadecimal Literal
```

```
#Float Literal
```

```
float_1 = 100.5
```

```
float_2 = 1.5e2
```

```
#Complex Literal
```

```
a = 5+3.14j
```

```
print(x, y, z, u)
```

```
print(float_1, float_2)
```

```
print(a, a.imag, a.real)
```

Output:

```
20 100 141 301
```

```
100.5 150.0
```

```
(5+3.14j) 3.14 5.0
```

3. Boolean literals:

A Boolean literal can have any of the two values: True or False.

Example - Boolean Literals

```
x = (1 == True)
y = (2 == False)
z = (3 == True)
a = True + 10
b = False + 10
```

```
print("x is", x)
print("y is", y)
print("z is", z)
print("a:", a)
print("b:", b)
```

Output:

```
x is True
y is False
z is False
a: 11
b: 10
```

4. Special literals:

- Python contains one special literal i.e., **None**.
- None is used to specify to that field that is not created. It is also used for the end of lists in Python.

Example - Special Literals

```
val1=10
val2=None
print(val1)
print(val2)
```

Output:

```
10
None
```

5. Literal Collections

Python provides the four types of literal collection such as List literals, Tuple literals, Dict literals, and Set literals.

List:

- List contains items of different data types. Lists are mutable i.e., modifiable.
- The values stored in List are separated by comma(,) and enclosed within square brackets([]). We can store different types of data in a List.

Example - List literals

```
list=['John',678,20.4,'Peter']
```

```
list1=[456,'Andrew']
```

```
print(list)
```

```
print(list + list1)
```

Output:

```
['John', 678, 20.4, 'Peter']
```

```
['John', 678, 20.4, 'Peter', 456, 'Andrew']
```

Dictionary:

- Python dictionary stores the data in the key-value pair.
- It is enclosed by curly-braces {} and each pair is separated by the commas(,).

Example

```
dict = {'name': 'Pater', 'Age':18,'Roll_nu':101}
```

```
print(dict)
```

Output:

```
{'name': 'Pater', 'Age': 18, 'Roll_nu': 101}
```

Tuple:

- Python tuple is a collection of different data-type. It is immutable which means it cannot be modified after creation.
- It is enclosed by the parentheses () and each element is separated by the comma(,).

Example

```
tup = (10,20,"Dev",[2,3,4])
```

```
print(tup)
```

Output:

```
(10, 20, 'Dev', [2, 3, 4])
```

Set:

- Python set is the collection of the unordered dataset.
- It is enclosed by the {} and each element is separated by the comma(,).

Example: - Set Literals

```
set = {'apple','grapes','guava','papaya'}
```

```
print(set)
```

Output:

```
{'guava', 'apple', 'papaya', 'grapes'}
```

Python Functional Programming

- Python provides the functional tools module that includes various functional programming tools.
- Functional programming is a programming paradigm in which the primary method of computation is evaluation of functions.
- Pure function is a function whose output value follows solely from its input values, without any observable side effects.
- In functional programming, a program consists entirely of evaluation of pure functions.
- Computation proceeds by nested or composed function calls, without changes to state or mutable data.
- To support functional programming, it's useful if a function in a given programming language has two abilities:
 - To take another function as an argument
 - To return another function to its caller
- Python plays nicely in both these respects. All objects in Python have more or less equal stature, and functions are no exception.
- In Python, functions are **first-class citizens**. That means functions have the same characteristics as values like strings and numbers. Anything you would expect to be able to do with a string or number you can do with a function as well.
- For example, you can assign a function to a variable. You can then use that variable the same as you would use the function itself:

```
>>> def func():
...     print("I am function func()!")
...
>>> func()
I am function func()

>>> another_name = func
>>> another_name()
I am function func()
```

- The assignment `another_name = func` on line 8 creates a new reference to `func()` named `another_name`.
- You can then call the function by either name, `func` or `another_name`, as shown on lines 5 and 9.
- You can display a function to the console with `print()`, include it as an element in a composite data object like a list, or even use it as a dictionary key:

```
>>> def func():
...     print("I am function func()!")
...
...
>>> print("cat", func, 42)
cat <function func at 0x7f81b4d29bf8> 42

>>> objects = ["cat", func, 42]
>>> objects[1]
<function func at 0x7f81b4d29bf8>
>>> objects[1]()
I am function func()

>>> d = {"cat": 1, func: 2, 42: 3}
>>> d[func]
```

- In this example, `func()` appears in all the same contexts as the values "cat" and 42, and the interpreter handles it just fine.

Note: What you can or can't do with any object in Python depends to some extent on context. There are some operations, for example, that work for certain object types but not for others. You can add two integer objects or concatenate two string objects with the plus operator (+). But the plus operator isn't defined for function objects.

- For present purposes, what matters is that functions in Python satisfy the two criteria beneficial for functional programming listed above. You can pass a function to another function as an argument:

```
>>> def inner():
...     print("I am function inner()!")
...
...
>>> def outer(function):
...     function()
...
...
>>> outer(inner)
I am function inner()
```

- Here's what's happening in the above example:
 - The call on line 9 passes `inner()` as an argument to `outer()`.
 - Within `outer()`, Python binds `inner()` to the function parameter `function`.
 - `outer()` can then call `inner()` directly via `function`.
- This is known as function composition.

Python File I/O

- Files are used to store data in a computer disk.
- We can open a file using Python script and perform various operations such as writing, reading, and appending.
- There are various ways of opening a file.

Python Modules

- Python modules are the program files that contain a Python code or functions.
- There are two types of module in the Python - User-defined modules and built-in modules.
- A module that the user defines, or we can say that our Python code saved with .py extension, is treated as a user-defined module.
- Built-in modules are predefined modules of Python. To use the functionality of the modules, we need to import them into our current working program.

Python Exceptions

- An exception can be defined as an unusual condition in a program resulting in the interruption in the flow of the program.
- Whenever an exception occurs, the program stops the execution, and thus the further code is not executed.
- Therefore, an exception is the run-time errors that are unable to handle by Python script. An exception is a Python object that represents an error.

Python CSV

- A csv stands for "comma separated values", which is defined as a simple file format that uses specific structuring to arrange tabular data.
- It stores tabular data such as spreadsheet or database in plain text and has a common format for data interchange.
- A csv file opens into the excel sheet, and the rows and columns data define the standard format.

Python Sending Mail

- We can send or read a mail using the Python script.
- Python's standard library modules are useful for handling various protocols such as POP3 and IMAP.

Python Magic Methods

- Python magic method is defined as the special method which adds "magic" to a class.
- It starts and ends with double underscores, for example, `_init_` or `_str_`.
- The built-in classes define many magic methods. The `dir()` function can be used to see the number of magic methods inherited by a class.
- It has two prefixes and suffix underscores in the method name.

Python OOPS Concepts

- Everything in Python is treated as an object including integer values, floats, functions, classes, and none.
- Apart from that, Python supports all oriented concepts. Below is the brief introduction of oops concepts of Python.

- **Classes and Objects** - Python classes are the blueprint of the object. An object is a collection of data and method that act on the data.
- **Inheritance** - An inheritance is a technique where one class inherits the properties of other classes.
- **Constructor** - Python provides a special method `__init__()` which is known as a constructor. This method is automatically called when an object is instantiated.
- **Data Member** - A variable that holds data associated with a class and its objects.

Python Advance Topics

Python includes many advance and useful concepts that help the programmer to solve the complex tasks. These concepts are given below.

Python Iterator

- An iterator is simply an object that can be iterated upon.
- It returns one object at a time.
- It can be implemented using the two special methods, `__iter__()` and `__next__()`.

Python Generators

The Generators are an easiest way of creating Iterators.

Python Decorators

- These are used to modify the behavior of the function.
- Decorators provide the flexibility to wrap another function to expand the working of wrapped function, without permanently modifying it.

Python Database Connections

We can use various databases along with Python. Python DBI-API acclaims standard sets of functionalities to be included in the database connectivity modules for respective RDBMS products.

Python CGI

- Python CGI stands for "Common Gateway Interface", which is used to define how to exchange information between the webserver and a custom Python script.
- The Common Gateway Interface is a standard for external gateway programs to interface with the server, such as HTTP Servers.

3.4 Data Analytics Overview

Introduction

- Python is one of the most important programming languages which is used in the data analytics nowadays.
- Many data scientists prefer it for its simplicity and it's helpful libraries like pandas, Matplotlib, SciKit-Learn, BeautifulSoup, and PyTorch.
- It helps in processing complex data in this era where the data is in petabyte.
- The kind of data on which we work during the analysis is mostly of the csv (comma separated values) format. Usually, the first row in the csv files represents as headers.

Essential Python libraries introductory:

- Scientific computing libraries such as NumPy, Pandas & SciPy.
- Visualization libraries such as Matplotlib and seaborn.
- Algorithm libraries such as scikit-learn and stats models.

NumPy:

- It is the short for Numerical Python it is important in the scientific computing analysis, provides narray objects efficiently, and linear algebra operations.

Pandas:

- Pandas' name came from panel data.
- It helps in making structuring data easier by providing data structure and functions made especially for it.

Matplotlib:

- Is the best-known Python library for providing interactive plots as well as many 2D data Visualizations.

IPython:

- It provides a robust and productive environment for interactive and exploratory computing, it provides a mathematical to connect IPython through a web browser, and an infrastructure for interactive parallel and distributed computing.

SciPy:

- A package addressing different scientific computing issues\

Importing & Exporting Datasets

The two essential things that we must take care of while importing the datasets are-

- Format- It refers to the way a file is encoded. The examples of prominent formats are .csv,.xlsx,.json, etc.
- Path of File- The path of the file refers to the location of the file where it is stored. It can be available either in any of the drives or some online source.
- It can be done in the following way-

Example:

```
import pandas as pd
path= " "
df = pd.read_csv(path)
If the dataset doesn't contain a header, we can specify it in the following way-
df = pd.read_csv(path,header=None)
```

To look at the first five and last five rows of the dataset, we can make use of df.head() and df.tail() respectively.

Let's have a look at how we can export the data, if we have a file present in the .csv format then,

```
path = " "
```

```
df.to_excel(path)
```

Python in Data Analytics Applications

RFM:

- Recency, frequency, and mandatory is important in the business analytics industry.
 - Recency helps in determining the last time the costumer purchased.
 - Frequency helps in determining how often the costumer purchases.
 - Mandatory helps in determining how much the costumer spends when he is purchasing from us.
- The RFM helps the business analytics to know their costumers more thus increase the profits, The customers are then ranked according to their RFM values.
- Python helps in calculating the RFM with its libraries (pandas, DateTime, and NumPy)
- Pandas helps in reading the data, DateTime helps in calculating the difference between the dates to determine the recency and the frequency, and NumPy helps in ranking the customers in order to determine the most and least loyal ones and after determining them the business analytics can decide how to gain more customers.

```
In [64]: rfm = fr_data.groupby('ID').agg({'ENDDATE':lambda
                                         date:(PRESENT - date.max()).days, 'STARTDATE':lambda
                                         num:len(num), 'TotalPrice':lambda price :price.sum()})
print(rfm.head())

```

ID	ENDDATE	STARTDATE	TotalPrice
71.0	453	1	1.2000
519.0	453	1	1.8000
688.0	453	1	6.4608
690.0	453	1	20.6360
693.0	453	1	24.4570



- It is a process of collecting raw data from the Web using automated method, but some webs forbid scrapping and they have their good reasons to protect their data.
- Python provide easy ways to make the web scrapping more powerful.
- Urllib is python standard library which helps in dealing with links to help accessing the web we want to scrap, BeautifulSoup helps in scraping the information from the web.

```
In [7]: page_soup = soup(page_html, "html.parser")
In [8]: page_soup.h1
Out[8]: <h1 class="header2021-logo"><a class="header2021-logo-img" href="https://www.newegg.com/" title="Newegg.com - Computer Parts, Laptops, Electronics, HDTVs, Digital Cameras and More!"></a></h1>
In [9]: page_soup.p
Out[9]: <p class="item-promo"><i class="item-promo-icon"></i>$150 promotional gift card w/ purchase, limited offer</p>
```

Market Basket Analysis

- It is one of the best applications in retail industry, industries need to mine and analyse their database to understand the data's pattern, Correlation Relationships among the data is very helpful in transactions, decision making and recognizing the customer's behavior in a large data set.
- To determine the history of:
 - Products that are likely purchased together
 - Products that are likely sequentially purchased
 - Products that are purchased seasonally
- It helps in choosing the best promotion, increase revenue and decrease the expenses.
- First, we need to calculate the support by using the sum of the two items together then dividing them by the total number of all the items.
- The confidence of item one to item two by taking the sum of the two items showing together and dividing them by the total of item 1 showing.
- After that we will be able to calculate the lift by dividing the confidence of item 1 to item 2 by item 1 divided by item 2.

Limitations:

- It takes long time to be implemented and may require regression and decision tree analysis skills and other more.
- Sometimes hard to determine the product groupings
- Complexity grows exponentially with size

Association Rule for Market Basket Analysis

- The market places use the association rule to know which application is more likely to be purchased after the other (antecedent, consequent)
- Association rule have associated population which consists of instances.
- MBA by python: We use the pandas, NumPy and apyori (used as API).

```
In [4]: import pandas as pd
import numpy as np
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

3.5 Statistical Computing

- Python statistics module provides the functions to mathematical statistics of numeric data.

Functions

mean() function

- The mean() function is used to calculate the arithmetic mean of the numbers in the list.

Example

```
import statistics
# list of positive integer numbers
datasets = [5, 2, 7, 4, 2, 6, 8]
x = statistics.mean(datasets)
# Printing the mean
print("Mean is :", x)
```

Output:

Mean is : 4.857142857142857

median() function

- The median() function is used to return the middle value of the numeric data in the list.

Example

```
import statistics
datasets = [4, -5, 6, 6, 9, 4, 5, -2]
# Printing median of the
# random data-set
print("Median of data-set is : % s "
      % (statistics.median(datasets)))
```

Output:

Median of data-set is : 4.5

mode() function

- The mode() function returns the most common data that occurs in the list.

Example

```
import statistics  
# declaring a simple data-set consisting of real valued positive integers.  
dataset =[2, 4, 7, 7, 2, 2, 3, 6, 6, 8]  
# Printing out the mode of given data-set  
print("Calculated Mode % s" % (statistics.mode(dataset)))
```

Output:

Calculated Mode 2

stdev() function

- The stdev() function is used to calculate the standard deviation on a given sample which is available in the form of the list.

Example

```
import statistics  
# creating a simple data - set  
sample = [7, 8, 9, 10, 11]  
# Prints standard deviation  
print("Standard Deviation of sample is % s "  
     % (statistics.stdev(sample)))
```

Output:

Standard Deviation of sample is 1.5811388300841898

median_low()

- The median_low function is used to return the low median of numeric data in the list.

Example

```
import statistics  
# simple list of a set of integers  
set1 = [4, 6, 2, 5, 7, 7]  
# Note: low median will always be a member of the data-set.  
# Print low median of the data-set  
print("Low median of data-set is % s "  
     % (statistics.median_low(set1)))
```

Output:

Low median of the data-set is 5

median_high()

- The median_high function is used to return the high median of numeric data in the list.

Example

```
import statistics  
# list of set of the integers  
dataset = [2, 1, 7, 6, 1, 9]  
print("High median of data-set is %s "  
     % (statistics.median_high(dataset)))
```

Output:

High median of the data-set is 6

Exploratory Data Analysis

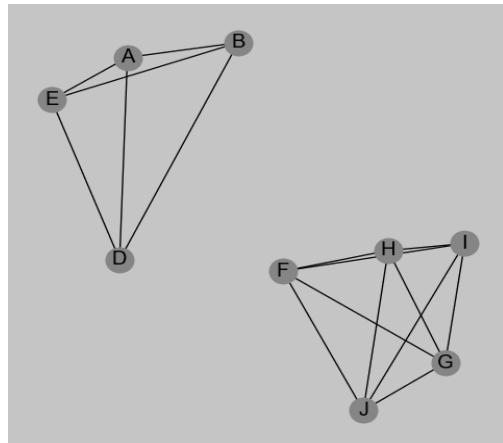
- We can find out the statistical summary of our dataset using describe() method.
- It can be used as df.describe().
- The categorical variables can be summarized using value_counts() method.

Using GroupBy

- The groupby() method of pandas can be applied to categorical variables.
- It groups the subsets based on different categories. It can involve single or multiple variables.
- Let us have a look at an example that would help us to understand how it can be used in Python.
 1. df_att=df[['attribute1', 'attribute2', 'attribute3']]
 2. df_g=df_att.groupby(['attribute1', 'attribute2'], as_index=False).mean()
 3. df_g

Correlation

- Correlation measures the scope to which two variables are interdependent.
- A visual idea of checking what kind of a correlation exists between the two variables.
- We can plot a graph and interpret how does a rise in the value of one attribute affects the other attribute.
- Concerning statistics, we can obtain the correlation using Pearson Correlation.
- It gives us the correlation coefficient and the P-value.



We can use it in our piece of code using scipy stat package.

CORRELATION COEFFICIENT	RELATIONSHIP
1. Close to +1	Large Positive
2. Close to -1	Large Negative
3. Close to 0	No relationship exists

P-VALUE	CERTAINTY
P-value<0.001	Strong
P-value<0.05	Moderate
P-value<0.1	Weak
P-value>0.1	No

3.6 Mathematical Computing using NumPy

Basics of Numpy

- Numpy is a Python library that is written in Python, but the parts that require fast computation are written in C or C++.
- For this reason, working with Numpy array is much faster than working with Python lists.
- Numpy being an open-source project has thousands of contributors working to keep NumPy fast, friendly, and bug-free.
- Numpy is hugely popular these days due to its use in various fields and tasks.
- Numpy has numerous uses:
 - Normal arithmetic and statistical operations are simple to implement Numpy.
 - Various trigonometric calculations can also be done.
 - Other uses are:
 - Broadcasting
 - Linear algebra
 - Matrix operations
 - Stacking
 - Copying and manipulating arrays
- NumPy contains a multi-dimensional array and matrix data structures, making large scale calculations simple and easy.
- Numpy has random number generators which can be used for various tasks like noise generation for signals for just random probability generations and other mathematical tasks.

```
import numpy as np
#numpy array
a= np.array([34,67,8,5,33,90,23])
print(a)
Output: [34 67 8 5 33 90 23]
```

```
#type
type(a)
Output: numpy.ndarray

print(a[2])

Output: 8
```

```
b=np.array([[77, 38, 9], [10, 11, 12]])
print(b)

Output: [[77 38 9] [10 11 12]]
```

Functions

Now, we look at some interesting functions in NumPy.

Ones: Creates a NumPy array according to the parameters given, with all elements being 1.

```
np.ones(5)
array([1., 1., 1., 1., 1.])
```

```
np.ones([6,7])
```

Output:

```
array([[1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.]])
```

Zeros: Creates a NumPy array according to the parameters given, with all elements being 0.

```
np.zeros(7)
Output: array([0., 0., 0., 0., 0., 0., 0.])
```

These functions are simple, they can be used to create sample arrays which are often needed for various computational purposes.

Eye: Let us now look at the eye function. This function returns a 2-D array with ones on the diagonal and zeros elsewhere.

```
np.eye(5)
Output:
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

Diag: Similarly, the diag() function creates a 2D array with the elements passed as diagonal elements and other elements being zero.

```
y=np.array([6,78,3,56,89])
np.diag(y)
```

Output:

```
array([[ 6,  0,  0,  0,  0],
       [ 0, 78,  0,  0,  0],
       [ 0,  0,  3,  0,  0],
       [ 0,  0,  0, 56,  0],
       [ 0,  0,  0,  0, 89]])
```

Let us try some more interesting functions. These are self-explanatory.

```
np.array([1, 2, 3,7] * 3)
```

Output : array([1, 2, 3, 7, 1, 2, 3, 7, 1, 2, 3, 7])

```
np.repeat([1, 4, 2, 3], 5)
```

Output : array([1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3])

```
p = np.ones([2, 3], int)
```

```
print(p)
```

Output :

```
[[1 1 1] [1 1 1]]
```

With vstack(), we can vertically append data, and with hstack(), we can horizontally stack data. Let us try out some examples.

```
np.vstack([p, 2*p])
```

Output :

```
array([[1, 1, 1],
       [1, 1, 1],
       [2, 2, 2],
       [2, 2, 2]])
```

Numpy Mathematical Computation

Let us get onto working with NumPy in various mathematical computations.

Now, we go for an element-by-element multiplication.

```
import numpy as np  
a=np.array([4,6,8])  
b=np.array([8,9,7])  
  
c=a+b  
  
print(c)
```

Output: [12 15 15]

```
np.vstack([3*p, 2*p, 6.5*p, 4.6*p])
```

Output :

```
array([[3. , 3. , 3. ],  
       [3. , 3. , 3. ],  
       [2. , 2. , 2. ],  
       [2. , 2. , 2. ],  
       [6.5, 6.5, 6.5],  
       [6.5, 6.5, 6.5],  
       [4.6, 4.6, 4.6],  
       [4.6, 4.6, 4.6]])
```

```
np.hstack([2*p, 5.5*p, 9*p])
```

Output:

```
array([[2. , 2. , 2. , 5.5, 5.5, 5.5, 9. , 9. , 9. ],  
       [2. , 2. , 2. , 5.5, 5.5, 5.5, 9. , 9. , 9. ]])
```

```
a=np.array([4,6,8])  
b=np.array([8,9,7])
```

```
c=a*b
```

```
print(c)
```

Output: [32 54 56]

Element by element division.

```
a=np.array([4,6,8])  
b=np.array([8,9,7])  
  
c=a/b  
  
print(c)
```

Output: [0.5 0.66666667 1.14285714]

We calculate the dot product now.

```
c=np.array([6,7,5,8])  
d=np.array([4,3,7,8])  
  
ans=c.dot(d) #dot product  
  
print(ans)
```

Output: 144

Let us now, look at how to create multi-dimensional numpy arrays.

```
z=np.array([[5,7,5,4,5],[6,2,3,4,6]])  
  
print(z)
```

Output: [[5 7 5 4 5] [6 2 3 4 6]]

Getting the transpose.

```
print(z.T)
```

Output: [[5 6] [7 2] [5 3] [4 4] [5 6]]

Let us create a list by giving a few parameters. The first and second parameter will be for determining the range and the third will be for the interval.

```
h=np.arange(4,90,5)  
print(h)
```

Output: [4 9 14 19 24 29 34 39 44 49 54 59 64 69 74 79 84 89]

Now, let us look at some standard functions. Let us carry some computations over multi-dimensional arrays.

```
test = np.random.randint(0,10,(4,3))
print(test)
```

Output: [[5 0 2] [3 6 0] [5 7 7] [0 5 9]]

Here 0 and 10 indicate the range, and 4,3 is the shape of the matrix/2D array.

```
test2 = np.random.randint(90,120,(8,3))
test2
```

Output: array([[106, 103, 104],
 [96, 93, 106],
 [110, 108, 115],
 [117, 106, 114],
 [91, 102, 103],
 [98, 104, 92],
 [112, 99, 105],
 [115, 111, 118]])

for row in test2:

```
    print(row)
```

Output:

```
[106 103 104] [ 96 93 106] [110 108 115] [117 106 114] [ 91 102 103] [ 98 104 92] [112 99 105]
[115 111 118]
```

```
test3 = test2**2
test3
```

Output:

```
array([[11236, 10609, 10816],
 [ 9216, 8649, 11236],
 [12100, 11664, 13225],
 [13689, 11236, 12996],
 [ 8281, 10404, 10609],
 [ 9604, 10816, 8464],
 [12544, 9801, 11025],
 [13225, 12321, 13924]], dtype=int32)
```

```
for i, j in zip(test2, test3):
    print(i,'+',j,'=',i+j)
```

Output:

```
[106 103 104] + [11236 10609 10816] = [11342 10712 10920] [ 96 93 106] + [ 9216 8649 11236]
= [ 9312 8742 11342] [110 108 115] + [12100 11664 13225] = [12210 11772 13340] [117 106 114]
+ [13689 11236 12996] = [13806 11342 13110] [ 91 102 103] + [ 8281 10404 10609] = [ 8372
10506 10712] [ 98 104 92] + [ 9604 10816 8464] = [ 9702 10920 8556] [112 99 105] + [12544 9801
11025] = [12656 9900 11130] [115 111 118] + [13225 12321 13924] = [13340 12432 14042]
```

Conclusion

- Thus, we can see that NumPy can be used for various types of mathematical calculations and the important thing is that the computation time is much less than python lists.
- This helps while working in real-life cases with millions of data points.
- Numpy array is also very convenient to use with a lot of data manipulation tricks and methods.

3.7 Data Manipulation with Pandas

- Pandas is an open-source data analysis and data manipulation library written in python.
- Pandas provide you with data structures and functions to work on structured data seamlessly.
- The name Pandas refer to “Panel Data”, which means a structured dataset. Pandas have two main classes to work on, DataFrame and Series.

Key Features of Pandas

- Perform Group by operation seamlessly
- Datasets are mutable using pandas which means we can add new rows and columns to them.
- Easy to handle missing data
- Merge and join datasets
- Indexing and subsetting data

Installation

Install via pip using the following command,

```
pip install pandas
```

Install via anaconda using the following command,

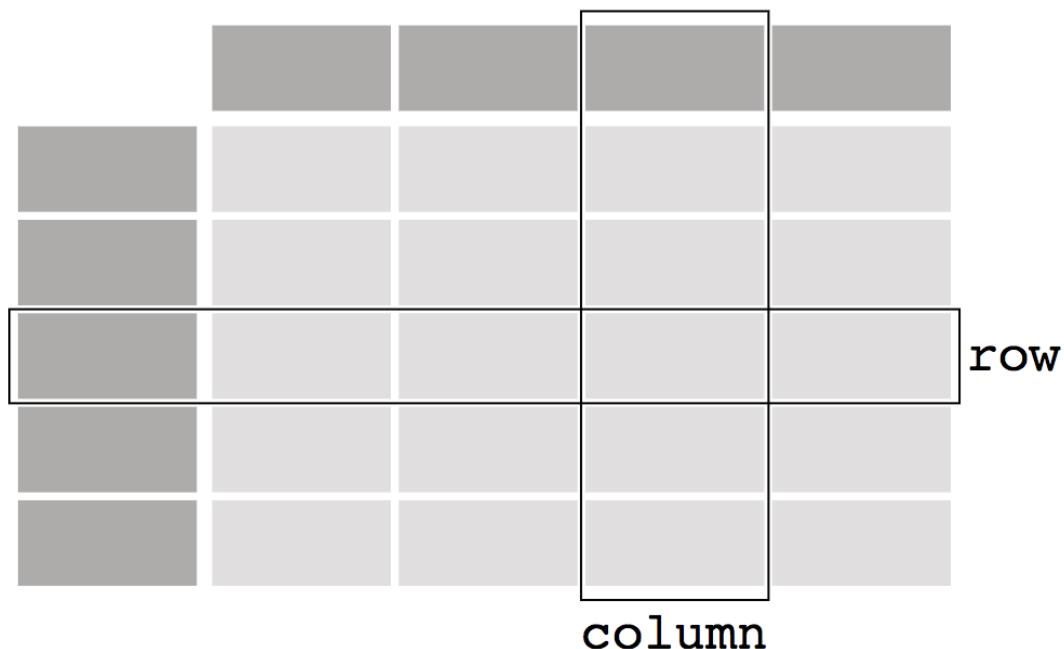
```
conda install pandas
```

DataFrame in Pandas

- A DataFrame is a two-dimensional table in pandas.
- Each column can have different data types like int, float, or string. Each column is of class Series in pandas.

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1		female	35.0	1	0	113803	53.1000	C123	S
4	5	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

DataFrame



Creating a DataFrame in Pandas

```
# import the library as pd
import pandas as pd
df = pd.DataFrame(
{
    'Name': ['Srividhayan', 'Hari'],
    'Age': [22, 11],
    'Country': ['India', 'India']
})
print(df)
# output
#      Name  Age Country
# 0  Srividhayan   22     India
# 1          Hari   11     India
```

- pd.DataFrame is a class available in pandas.
- Here we provide a dictionary whose keys are the column names ('Name', 'Age', 'Country') and the values are the values in those columns.
- Here each column is of class pandas.Series. Series is a one-dimensional data used in pandas.

```
# accessing the column 'Name' in df
print(df['Name'])
# Output
# 0  Srividhayan
# 1          Hari
# Name: Name, dtype: object
print(type(df['Name']))
# Output
# <class 'pandas.core.series.Series'>
```

Data Manipulation using Pandas

- For this purpose, we are going to use Titanic Dataset which is available on Kaggle.

```
import pandas as pd
path_to_data = 'path/to/titanic_dataset'
# read the csv data using pd.read_csv function
data = pd.read_csv(path_to_data)
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Allen, Mr. William Henry	male	35.0	1	0	113803	53.1000	C123	S
4	5	0	3				0	0	373450	8.0500	NaN	S

Dropping Columns in the Data

```
df_dropped = data.drop('Survived', axis=1)
df_dropped.head()
```

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0 26.0	1 0	0	PC 17599 STON/O2. 3101282	71.2833 7.9250	C85 NaN	C S
2	3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
3	4	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
4	5									

- The ‘Survived’ column is dropped in the data. The axis=1 denotes that it ‘Survived’ is a column, so it searches ‘Survived’ column-wise to drop.
- Drop multiple columns using the following code:

```
df_dropped_multiple = data.drop(['Survived', 'Name'], axis=1)
df_dropped_multiple.head()
```

PassengerId	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	female	35.0	1	0	113803	53.1000	C123	S
4	5	male	35.0	0	0	373450	8.0500	NaN	S

- The columns ‘Survived’ and ‘Name’ are dropped in the data.

Dropping rows in the data

```
df_row_dropped = data.drop(2, axis=0)
df_row_dropped.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q

- The row with index 2 is dropped in the data. The axis=0 denotes that index 2 is a row, so it searches the index 2 column-wise.
- Drop multiple rows using the following code:

```
df_row_dropped_multiple = data.drop([2, 3], axis=0)
df_row_dropped_multiple.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S

- The rows with indexes 2 and 3 are dropped in the data.

Renaming a column in the dataset

```
data.columns
# Output
# Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
#        'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
#       dtype='object')
df_renamed = data.rename(columns={'PassengerId': 'Id'})
df_renamed.head()
```

	Id	Survived	Pclass		Name	Sex	Age	SibSp	Parch		Ticket	Fare	Cabin	Embarked	
0	1	0	3		Braund, Mr. Owen Harris	male	22.0	1	0		A/5 21171	7.2500	NaN	S	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0		PC 17599	71.2833	C85		C	
2	3	1	3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0	0	0		STON/O2. 3101282	7.9250	NaN		S	
3	4	1	1		Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0		113803	53.1000	C123		S
4	5	0	3		Allen, Mr. William Henry	male	35.0	0	0		373450	8.0500	NaN		S

- The column ‘PassengerId’ is renamed to ‘Id’ in the data. Do not forget to mention the dictionary inside the columns parameter.
- Rename multiple columns using the following code:

```
df_renamed_multiple = data.rename(
    columns={
        'PassengerId': 'Id',
        'Sex': 'Gender',
    }
)
df_renamed_multiple.head()
```

	Id	Survived	Pclass		Name	Gender	Age	SibSp	Parch		Ticket	Fare	Cabin	Embarked	
0	1	0	3		Braund, Mr. Owen Harris	male	22.0	1	0		A/5 21171	7.2500	NaN	S	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0		PC 17599	71.2833	C85		C	
2	3	1	3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0	0	0		STON/O2. 3101282	7.9250	NaN		S	
3	4	1	1		Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0		113803	53.1000	C123		S
4	5	0	3		Allen, Mr. William Henry	male	35.0	0	0		373450	8.0500	NaN		S

- The columns ‘PassengerId’ and ‘Sex’ are renamed to ‘Id’ and ‘Gender’ respectively.

Select columns with specific data types

```
integer_data = data.select_dtypes('int')
integer_data.head()
```

	PassengerId	Survived	Pclass	SibSp	Parch
0	1	0	3	1	0
1	2	1	1	1	0
2	3	1	3	0	0
3	4	1	1	1	0
4	5	0	3	0	0

- The above code selects all columns with integer data types.

```
float_data = data.select_dtypes('float')
float_data.head()
```

	Age	Fare
0	22.0	7.2500
1	38.0	71.2833
2	26.0	7.9250
3	35.0	53.1000
4	35.0	8.0500

- The above code selects all columns with float data types.

Slicing the dataset

```
data.iloc[:5, 0]
```

```
0    1
1    2
2    3
3    4
4    5
Name: PassengerId, dtype: int64
```

- The above code returns the first five rows of the first column.
- The ‘:5’ in the iloc denotes the first five rows and the number 0 after the comma denotes the first column, iloc is used to locate the data using numbers or integers.

```
data.loc[:5, 'PassengerId']
```

```
0    1
1    2
2    3
3    4
4    5
5    6
Name: PassengerId, dtype: int64
```

- The above code does the same but we can use the column names directly using loc in pandas. Here the index 5 is inclusive.

Handle Duplicates in Dataset

- Since there are no duplicate data in the titanic dataset, let us first add a duplicated row into the data and handle it.

```
df_dup = data.copy()
# duplicate the first row and append it to the data
row = df_dup.iloc[:1]
df_dup = df_dup.append(row, ignore_index=True)
df_dup
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0	PC 17599 STON/O2. 3101282	71.2833 7.9250	C85	C
2	3	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0	0	0	113803	53.1000	C123	S
3	4	1	Allen, Mr. William Henry	male	35.0	1	0	373450	8.0500	NaN	S
4	5	0
...
887	888	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W.C. 6607	23.4500	NaN	S
889	890	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q
891	1	0	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S

```
df_dup[df_dup.duplicated()]
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
891	1	0	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S

- The above code returns the duplicated rows in the data.

```
df_dup.drop_duplicates()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0	PC 17599 STON/O2. 3101282	71.2833 7.9250	C85	C
2	3	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0	0	0	113803	53.1000	C123	S
3	4	1	Allen, Mr. William Henry	male	35.0	1	0	373450	8.0500	NaN	S
4	5	0
...
886	887	0	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W.C. 6607	23.4500	NaN	S
889	890	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

- The above code drops the duplicated rows in the data.

Select specific values in the column

```
data[data['Pclass'] == 1]
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th... female	38.0	1	0	PC 17599	71.2833		C85	C
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel) female	35.0	1	0	113803	53.1000		C123	S
6	7	0	McCarthy, Mr. Timothy J male	54.0	0	0	17463	51.8625		E46	S
11	12	1	Bonnell, Miss. Elizabeth female	58.0	0	0	113783	26.5500		C103	S
23	24	1	Sloper, Mr. William Thompson male	28.0	0	0	113788	35.5000		A6	S
...
871	872	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny) female	47.0	1	1	11751	52.5542		D35	S
872	873	0	Carlsson, Mr. Frans Olof male	33.0	0	0	695	5.0000	B51 B53 B55		S
879	880	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson) female	56.0	0	1	11767	83.1583		C50	C
887	888	1	Graham, Miss. Margaret Edith female	19.0	0	0	112053	30.0000		B42	S
889	890	1	Behr, Mr. Karl Howell male	26.0	0	0	111369	30.0000		C148	C

- The above code returns the values which are equal to one in the column ‘Pclass’ in the data.
- Select multiple values in the column using the following code:

```
data[data['Pclass'].isin([1, 0])]
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th... female	38.0	1	0	PC 17599	71.2833		C85	C
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel) female	35.0	1	0	113803	53.1000		C123	S
6	7	0	McCarthy, Mr. Timothy J male	54.0	0	0	17463	51.8625		E46	S
11	12	1	Bonnell, Miss. Elizabeth female	58.0	0	0	113783	26.5500		C103	S
23	24	1	Sloper, Mr. William Thompson male	28.0	0	0	113788	35.5000		A6	S
...
871	872	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny) female	47.0	1	1	11751	52.5542		D35	S
872	873	0	Carlsson, Mr. Frans Olof male	33.0	0	0	695	5.0000	B51 B53 B55		S
879	880	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson) female	56.0	0	1	11767	83.1583		C50	C
887	888	1	Graham, Miss. Margaret Edith female	19.0	0	0	112053	30.0000		B42	S
889	890	1	Behr, Mr. Karl Howell male	26.0	0	0	111369	30.0000		C148	C

- The above code returns the values which are equal to one and zero in the column ‘Pclass’ in the data.

Group by in DataFrame

```
data.groupby('Sex').agg({'PassengerId': 'count'})
```

PassengerId	
Sex	
female	314
male	577

- The above code groups the values of the column 'Sex' and aggregates the column 'PassengerId' by the count of that column.

```
data.groupby('Sex').agg({'Age':'mean'})
```

Age	
Sex	
female	27.915709
male	30.726645

- Group multiple columns using the following code:

```
data.groupby(['Pclass', 'Sex']).agg({'PassengerId': 'count'})
```

PassengerId		
Pclass	Sex	
1	female	94
	male	122
2	female	76
	male	108
3	female	144
	male	347

Map in Pandas

```
data['Survived'].map(lambda x: 'Survived' if x==1 else 'Not-Survived')
```

```
0      Not-Survived
1      Survived
2      Survived
3      Survived
4      Not-Survived
...
886    Not-Survived
887    Survived
888    Not-Survived
889    Survived
890    Not-Survived
Name: Survived, Length: 891, dtype: object
```

- The above code maps the values 0 to 'Not-Survived' and 1 to 'Survived'. You can alternatively use the following code to obtain the same results.

```
data['Survived'].map({1: 'Survived', 0: 'Not-Survived'})
```

Replacing values in a DataFrame

```
data['Sex'].replace(['male', 'female'], ["M", "F"])
```

```
0      M
1      F
2      F
3      F
4      M
...
886    M
887    F
888    F
889    M
890    M
Name: Sex, Length: 891, dtype: object
```

- The above code replaces 'male' as 'M' and 'female' as 'F'.

Save the DataFrame as a CSV file

```
data.to_csv('/path/to/save/the/data.csv', index=False)
```

- The index=False argument does not save the index as a separate column in the CSV.

Boolean Indexing in Pandas

- For instance, we want a list of all females who are not graduates and got a loan. Boolean indexing can help here. You can use the following code:

```
data.loc[(data["Gender"]=="Female")      &      (data["Education"]=="Not Graduate")      &
         (data["Loan_Status"]=="Y"), ["Gender", "Education", "Loan_Status"]]
```

	Gender	Education	Loan_Status
Loan_ID			
LP001155	Female	Not Graduate	Y
LP001669	Female	Not Graduate	Y
LP001692	Female	Not Graduate	Y
LP001908	Female	Not Graduate	Y
LP002300	Female	Not Graduate	Y
LP002314	Female	Not Graduate	Y
LP002407	Female	Not Graduate	Y
LP002489	Female	Not Graduate	Y
LP002502	Female	Not Graduate	Y
LP002534	Female	Not Graduate	Y
LP002582	Female	Not Graduate	Y
LP002731	Female	Not Graduate	Y
LP002757	Female	Not Graduate	Y
LP002917	Female	Not Graduate	Y

Apply Function in Pandas

- It is one of the commonly used Pandas functions for manipulating a pandas dataframe and creating new variables.
- Pandas Apply function returns some value after passing each row/column of a data frame with some function.
- The function can be both default or user-defined. For instance, here it can be used to find the #missing values in each row and column.

```
#Create a new function:
```

```
def num_missing(x):
    return sum(x.isnull())
```

```
#Applying per column:
```

```
print "Missing values per column:"
print data.apply(num_missing, axis=0) #axis=0 defines that function is to be applied on each column
```

```
#Applying per row:
```

```
print "\nMissing values per row:"
print data.apply(num_missing, axis=1).head() #axis=1 defines that function is to be applied on each row
```

```
Missing values per column:
Gender           13
Married          3
Dependents       15
Education         0
Self_Employed    32
ApplicantIncome   0
CoapplicantIncome 0
LoanAmount        22
Loan_Amount_Term 14
Credit_History    50
Property_Area     0
Loan_Status        0
dtype: int64
```

```
Missing values per row:
Loan_ID
LP001002    1
LP001003    0
LP001005    0
LP001006    0
LP001008    0
dtype: int64
```

- Thus, we get the desired result.
- Note: Pandas head() function is used in second output because it contains many rows.

Pivot Table in Pandas

- Pandas can be used to create MS Excel style pivot tables.
- For instance, in this case, a key column is “LoanAmount” which has missing values.
- We can impute it using mean amount of each ‘Gender’, ‘Married’ and ‘Self_Employed’ group.
- The mean ‘LoanAmount’ of each group in Pandas dataframe can be determined as:

```
#Determine pivot table
impute_grps = data.pivot_table(values=["LoanAmount"],
index=["Gender","Married","Self_Employed"], aggfunc=np.mean)
print impute_grps
```

		L
Married	Self_Employed	
No	No	1
Yes	No	1
Yes	Yes	1
No	No	2
No	Yes	1
Yes	No	1
Yes	Yes	1

Sorting Pandas DataFrames

- Pandas allow easy sorting based on multiple columns. This can be done as:

```
data_sorted = data.sort_values(['ApplicantIncome','CoapplicantIncome'], ascending=False)
data_sorted[['ApplicantIncome','CoapplicantIncome']].head(10)
```

	ApplicantIncome	CoapplicantIncome
Loan_ID		
LP002317	81000	0
LP002101	63337	0
LP001585	51763	0
LP001536	39999	0
LP001640	39147	4750
LP002422	37719	0
LP001637	33846	0
LP001448	23803	0
LP002624	20833	6667
LP001922	20667	0

3.8 Data Visualization with Python

- Python provides various libraries that come with different features for visualizing data.
- All these libraries come with different features and can support various types of graphs.
- We will be discussing four such libraries.
 - Matplotlib
 - Seaborn
 - Bokeh
 - Plotly

Tips Database

- Tips database is the record of the tip given by the customers in a restaurant for two and a half months in the early 1990s. It contains 6 columns such as total bill, tip, sex, smoker, day, time, size.

Example:

```
import pandas as pd
# reading the database
data = pd.read_csv("tips.csv")
# printing the top 10 rows

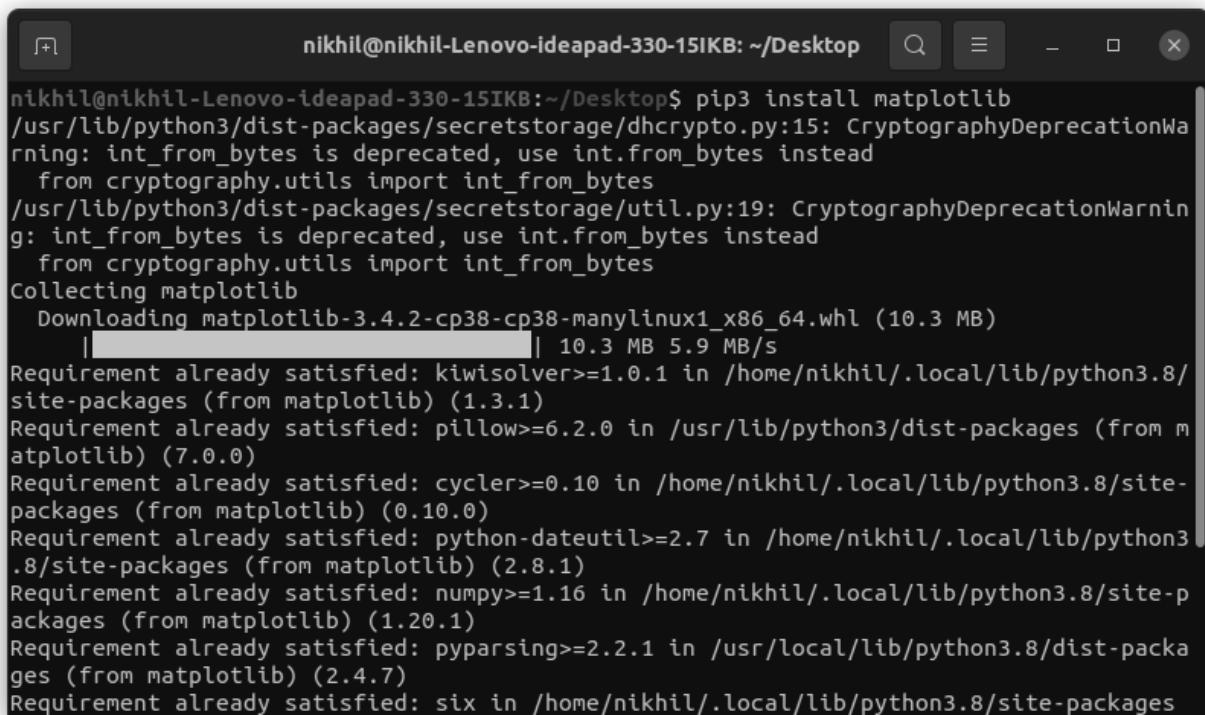
display(data.head(10))
```

Output:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2

Matplotlib

- Matplotlib is an easy-to-use, low-level data visualization library that is built on NumPy arrays.
- It consists of various plots like scatter plot, line plot, histogram, etc. Matplotlib provides a lot of flexibility.
- To install this type the below command in the terminal.
`pip install matplotlib`



```
nikhil@nikhil-Lenovo-ideapad-330-15IKB: ~/Desktop$ pip3 install matplotlib
/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes instead
    from cryptography.utils import int_from_bytes
/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes instead
    from cryptography.utils import int_from_bytes
Collecting matplotlib
  Downloading matplotlib-3.4.2-cp38-cp38-manylinux1_x86_64.whl (10.3 MB)
    |██████████| 10.3 MB 5.9 MB/s
Requirement already satisfied: kiwisolver>=1.0.1 in /home/nikhil/.local/lib/python3.8/site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/lib/python3/dist-packages (from matplotlib) (7.0.0)
Requirement already satisfied: cycler>=0.10 in /home/nikhil/.local/lib/python3.8/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>=2.7 in /home/nikhil/.local/lib/python3.8/site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: numpy>=1.16 in /home/nikhil/.local/lib/python3.8/site-packages (from matplotlib) (1.20.1)
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (2.4.7)
Requirement already satisfied: six in /home/nikhil/.local/lib/python3.8/site-packages
```

After installing Matplotlib, let's see the most commonly used plots using this library.

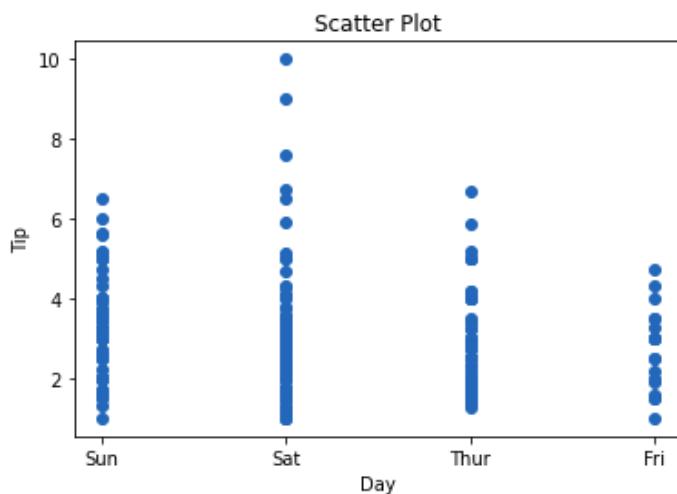
Scatter Plot

Scatter plots are used to observe relationships between variables and uses dots to represent the relationship between them. The [scatter\(\)](#) method in the matplotlib library is used to draw a scatter plot.

Example:

```
import pandas as pd
import matplotlib.pyplot as plt
# reading the database
data = pd.read_csv("tips.csv")
# Scatter plot with day against tip
plt.scatter(data['day'], data['tip'])
# Adding Title to the Plot
plt.title("Scatter Plot")
# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')
plt.show()
```

Output:



This graph can be more meaningful if we can add colors and also change the size of the points. We can do this by using the **c** and **s** parameter respectively of the scatter function. We can also show the color bar using the `colorbar()` method.

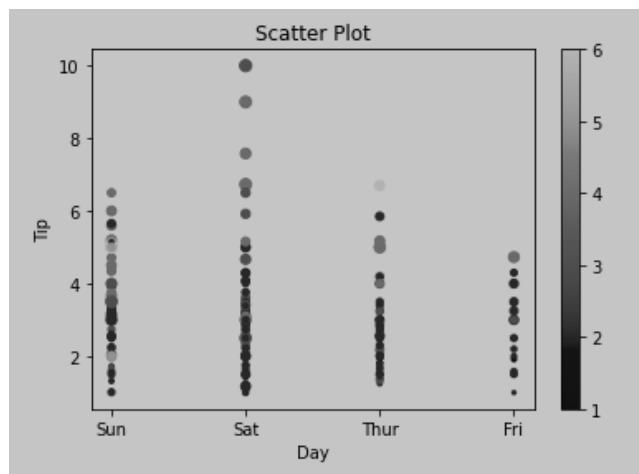
Example:

```
import pandas as pd
import matplotlib.pyplot as plt

# reading the database
data = pd.read_csv("tips.csv")
# Scatter plot with day against tip
plt.scatter(data['day'], data['tip'], c=data['size'],
            s=data['total_bill'])

# Adding Title to the Plot
plt.title("Scatter Plot")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')
plt.colorbar()
plt.show()
```

Output:

Line Chart

- Line Chart is used to represent a relationship between two data X and Y on a different axis.
- It is plotted using the **plot()** function. Let's see the below example.

Example:

```
import pandas as pd
import matplotlib.pyplot as plt

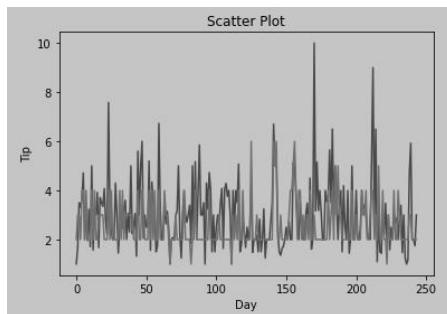
# reading the database
data = pd.read_csv("tips.csv")

# Scatter plot with day against tip
plt.plot(data['tip'])
plt.plot(data['size'])

# Adding Title to the Plot
plt.title("Scatter Plot")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

plt.show()
```

Output:

Bar Chart

- A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent.
- It can be created using the **bar()** method.

Example:

```
import pandas as pd
import matplotlib.pyplot as plt

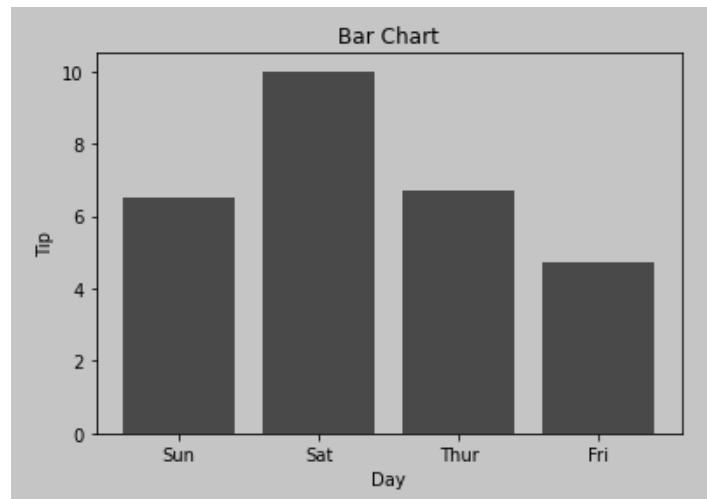
# reading the database
data = pd.read_csv("tips.csv")

# Bar chart with day against tip
plt.bar(data['day'], data['tip'])

plt.title("Bar Chart")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

# Adding the legends
plt.show()
```

Output:

Histogram

- A histogram is basically used to represent data in the form of some groups.
- It is a type of bar plot where the X-axis represents the bin ranges while the Y-axis gives information about frequency.
- The **hist()** function is used to compute and create a histogram.
- In histogram, if we pass categorical data then it will automatically compute the frequency of that data i.e. how often each value occurred.

Example:

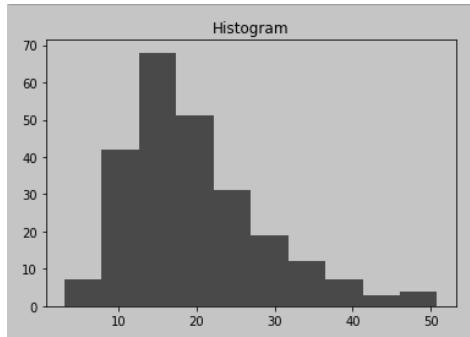
```
import pandas as pd
import matplotlib.pyplot as plt

# reading the database
data = pd.read_csv("tips.csv")

# histogram of total_bills
plt.hist(data['total_bill'])

plt.title("Histogram")

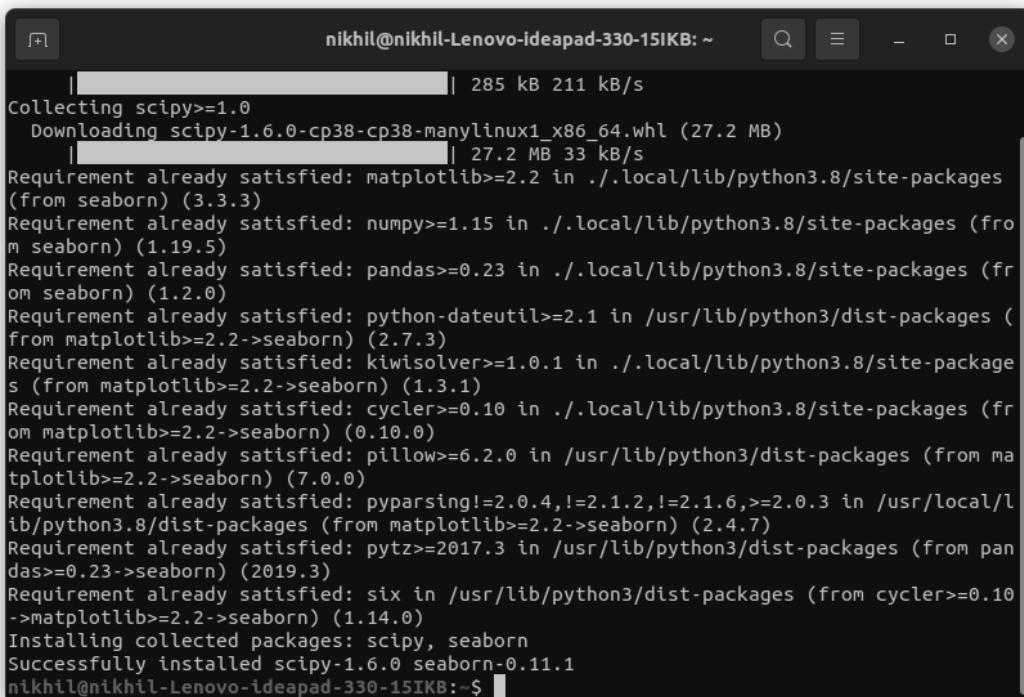
# Adding the legends
plt.show()
```

Output:

Seaborn

- **Seaborn** is a high-level interface built on top of the Matplotlib.
- It provides beautiful design styles and color palettes to make more attractive graphs.
- To install seaborn type the below command in the terminal.

```
pip install seaborn
```



A screenshot of a terminal window titled "nikhil@nikhil-Lenovo-ideapad-330-15IKB: ~". The window shows the output of the command "pip install seaborn". The output includes progress bars for downloading packages like "scipy-1.6.0-cp38-cp38-manylinux1_x86_64.whl" and "seaborn-0.11.1-py3-none-any.whl". It also lists requirements already satisfied for matplotlib, numpy, pandas, python-dateutil, kiwisolver, cycler, pillow, pyparsing, pytz, six, and seaborn. Finally, it shows the successful installation of "scipy-1.6.0" and "seaborn-0.11.1".

```
Collecting scipy>=1.0
  Downloading scipy-1.6.0-cp38-cp38-manylinux1_x86_64.whl (27.2 MB)
|██████████| 285 kB 211 kB/s
Requirement already satisfied: matplotlib>=2.2 in ./local/lib/python3.8/site-packages (from seaborn) (3.3.3)
Requirement already satisfied: numpy>=1.15 in ./local/lib/python3.8/site-packages (from seaborn) (1.19.5)
Requirement already satisfied: pandas>=0.23 in ./local/lib/python3.8/site-packages (from seaborn) (1.2.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/lib/python3/dist-packages (from matplotlib>=2.2->seaborn) (2.7.3)
Requirement already satisfied: kiwisolver>=1.0.1 in ./local/lib/python3.8/site-packages (from matplotlib>=2.2->seaborn) (1.3.1)
Requirement already satisfied: cycler>=0.10 in ./local/lib/python3.8/site-packages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/lib/python3/dist-packages (from matplotlib>=2.2->seaborn) (7.0.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.2->seaborn) (2.4.7)
Requirement already satisfied: pytz>=2017.3 in /usr/lib/python3/dist-packages (from pandas>=0.23->seaborn) (2019.3)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from cycler>=0.10->matplotlib>=2.2->seaborn) (1.14.0)
Installing collected packages: scipy, seaborn
Successfully installed scipy-1.6.0 seaborn-0.11.1
nikhil@nikhil-Lenovo-ideapad-330-15IKB:~$
```

- Seaborn is built on the top of Matplotlib; therefore, it can be used with the Matplotlib as well.
- Using both Matplotlib and Seaborn together is a very simple process.
- We just have to invoke the Seaborn Plotting function as normal, and then we can use Matplotlib's customization function.

Note: Seaborn comes loaded with dataset such as tips, iris, etc. but for the sake of this tutorial we will use Pandas for loading these datasets.

Example:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

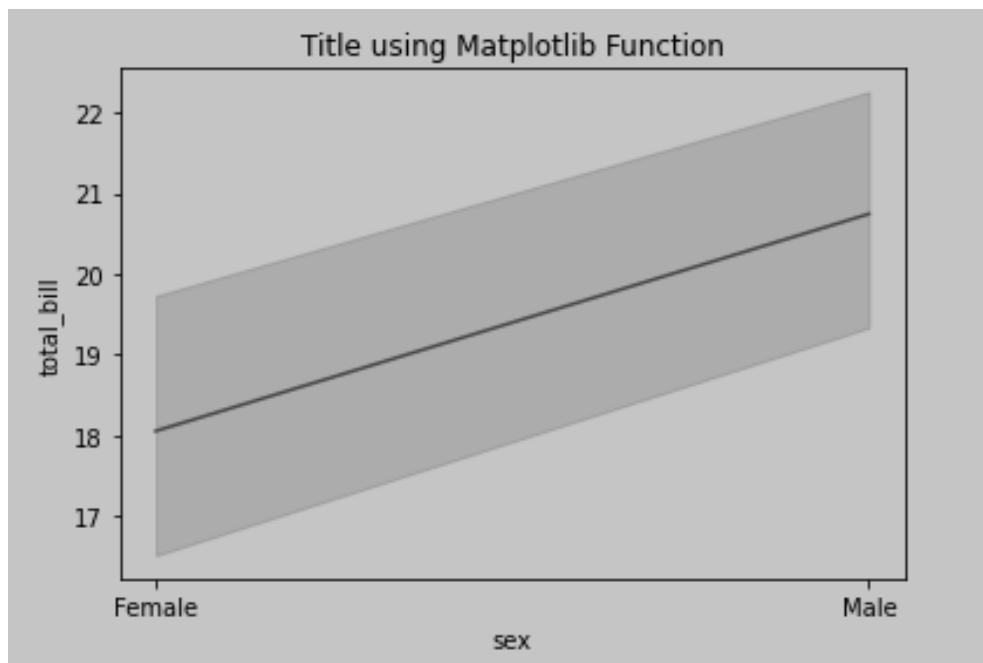
# reading the database
data = pd.read_csv("tips.csv")

# draw lineplot
sns.lineplot(x="sex", y="total_bill", data=data)

# setting the title using Matplotlib
plt.title('Title using Matplotlib Function')

plt.show()
```

Output:



Scatter Plot

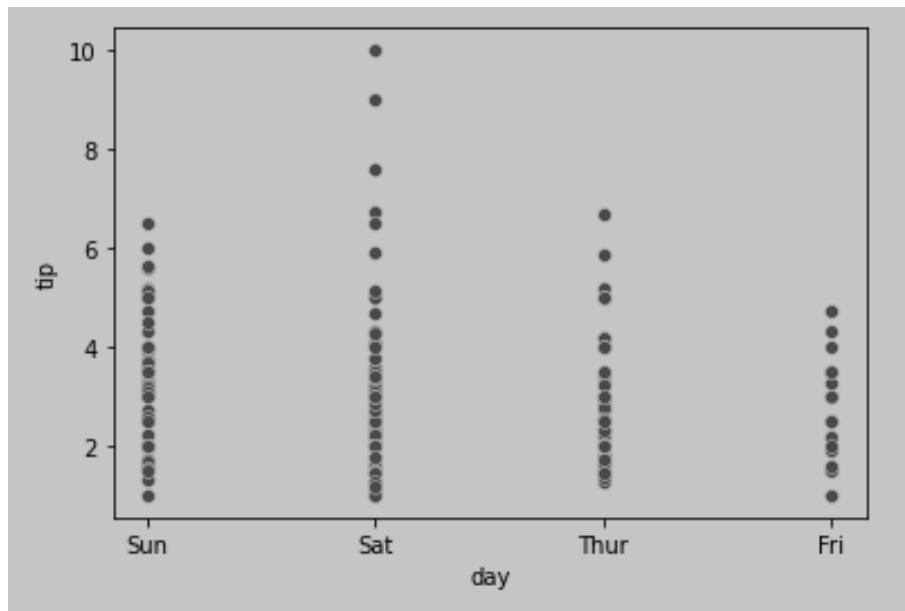
- Scatter plot is plotted using the **scatterplot()** method. This is similar to Matplotlib, but additional argument data is required.

Example:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

sns.scatterplot(x='day', y='tip', data=data,)
plt.show()
```

Output:

You will find that while using Matplotlib it will a lot difficult if you want to color each point of this plot according to the sex. But in scatter plot it can be done with the help of hue argument.

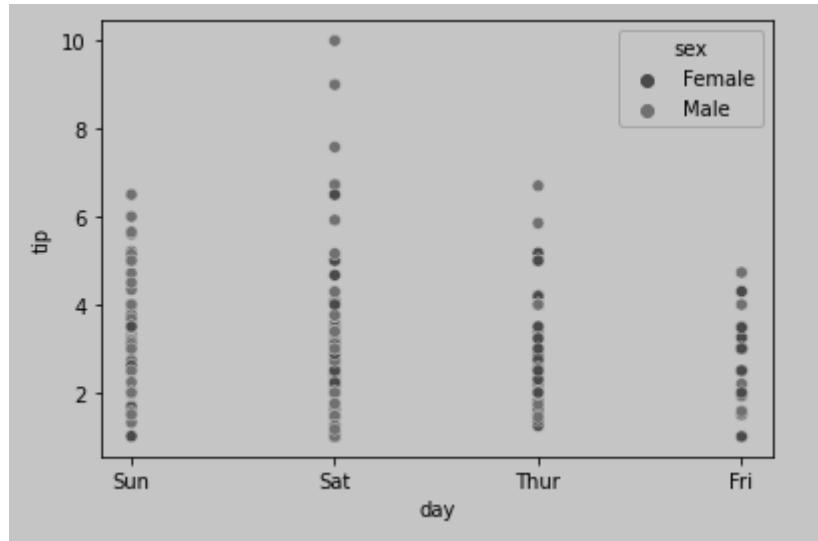
Example:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

sns.scatterplot(x='day', y='tip', data=data,
                 hue='sex')
plt.show()
```

Output:



Line Plot

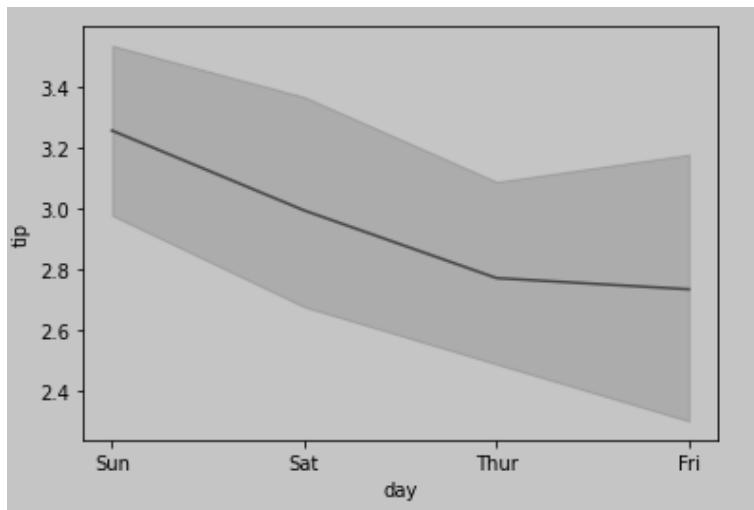
Line Plot in Seaborn plotted using the [lineplot\(\)](#) method. In this, we can pass only the data argument also.

Example 1:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

sns.lineplot(x='day', y='tip', data=data)
plt.show()
```

Output:

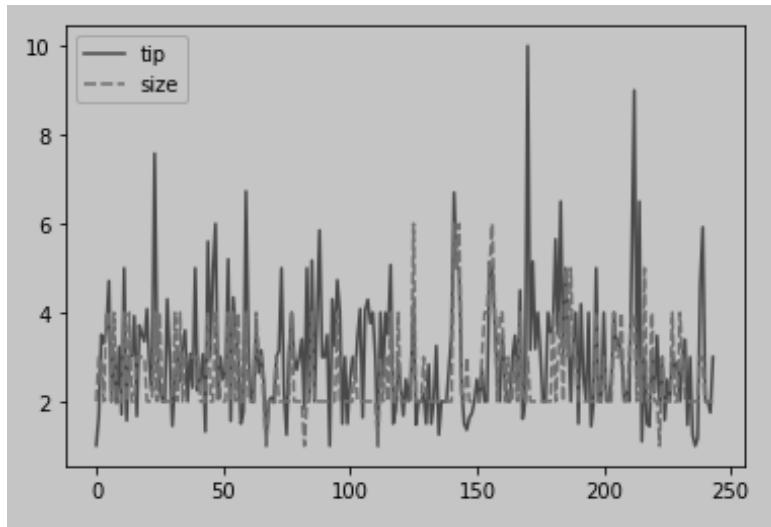
Example 2:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

# using only data attribute
sns.lineplot(data=data.drop(['total_bill'], axis=1))
plt.show()
```

Output:



Bar Plot

Bar Plot in Seaborn can be created using the **barplot()** method.

Example:

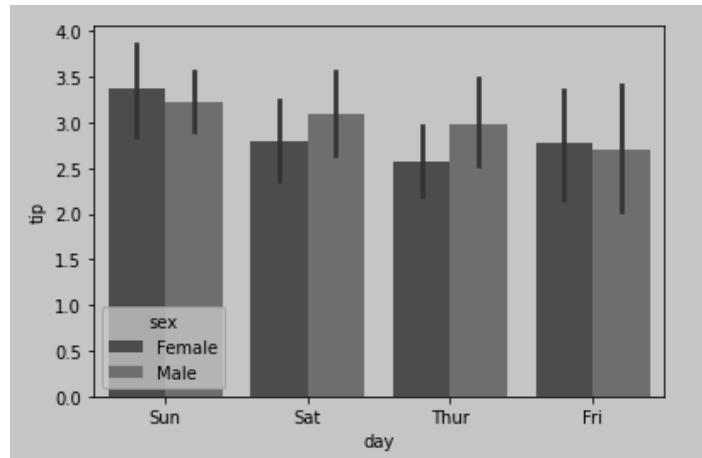
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

sns.barplot(x='day',y='tip', data=data,
            hue='sex')

plt.show()
```

Output:



Histogram

The histogram in Seaborn can be plotted using the **histplot()** function.

Example:

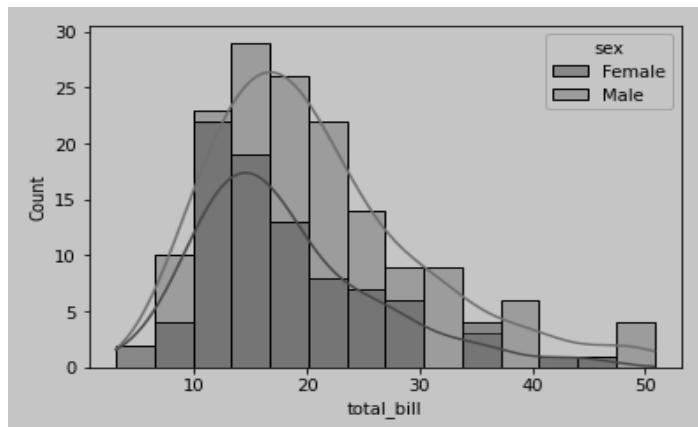
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

sns.histplot(x='total_bill', data=data, kde=True, hue='sex')

plt.show()
```

Output:

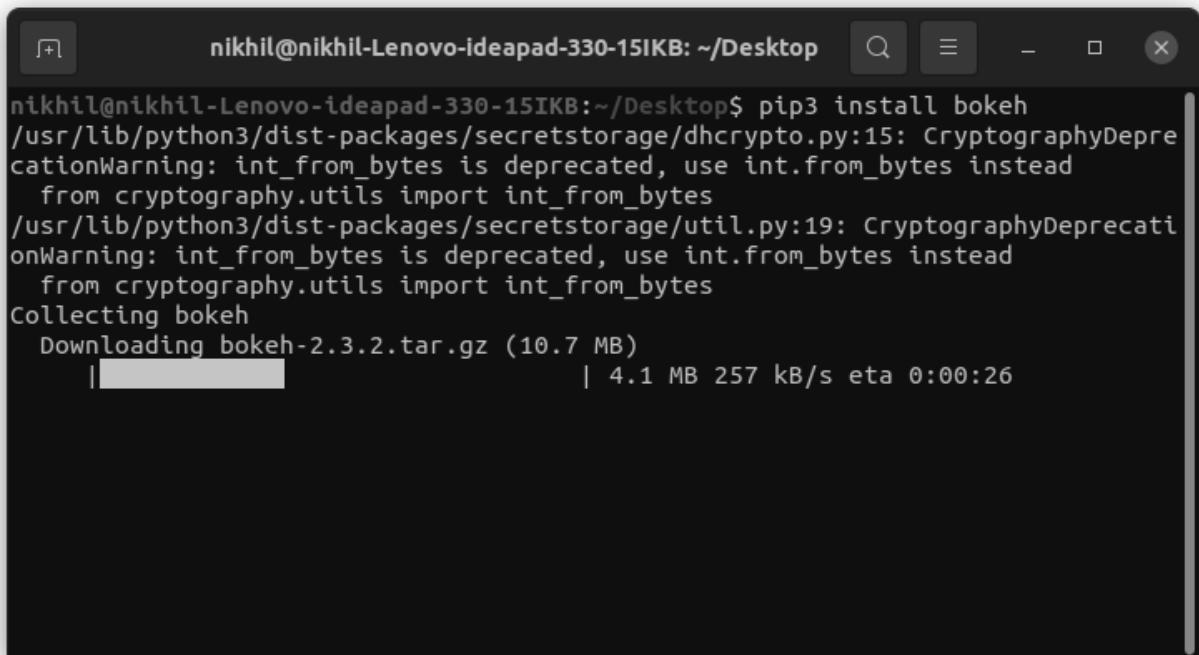


- After going through all these plots you must have noticed that customizing plots using Seaborn is a lot more easier than using Matplotlib.
- It is also built over matplotlib then we can also use matplotlib functions while using Seaborn.

Bokeh

- Let's move on to the third library of our list. Bokeh is mainly famous for its interactive charts visualization.
- Bokeh renders its plots using HTML and JavaScript that uses modern web browsers for presenting elegant, concise construction of novel graphics with high-level interactivity.
- To install this type the below command in the terminal.

```
pip install bokeh
```

A screenshot of a terminal window titled "nikhil@nikhil-Lenovo-ideapad-330-15IKB: ~/Desktop". The window shows the command "pip3 install bokeh" being run. The output includes several deprecation warnings from the cryptography library, followed by the download and collection of the bokeh package. A progress bar indicates the download speed is 4.1 MB/s and the estimated time remaining is 0:00:26.

```
nikhil@nikhil-Lenovo-ideapad-330-15IKB:~/Desktop$ pip3 install bokeh
/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes instead
    from cryptography.utils import int_from_bytes
/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes instead
    from cryptography.utils import int_from_bytes
Collecting bokeh
  Downloading bokeh-2.3.2.tar.gz (10.7 MB)
     |██████████| 4.1 MB 257 kB/s eta 0:00:26
```

Scatter Plot

Scatter Plot in Bokeh can be plotted using the scatter() method of the plotting module. Here pass the x and y coordinates respectively.

Example:

```
#importing the modules
from bokeh.plotting import figure, output_file, show
from bokeh.palettes import magma
import pandas as pd

# instantiating the figure object
graph = figure(title = "Bokeh Scatter Graph")

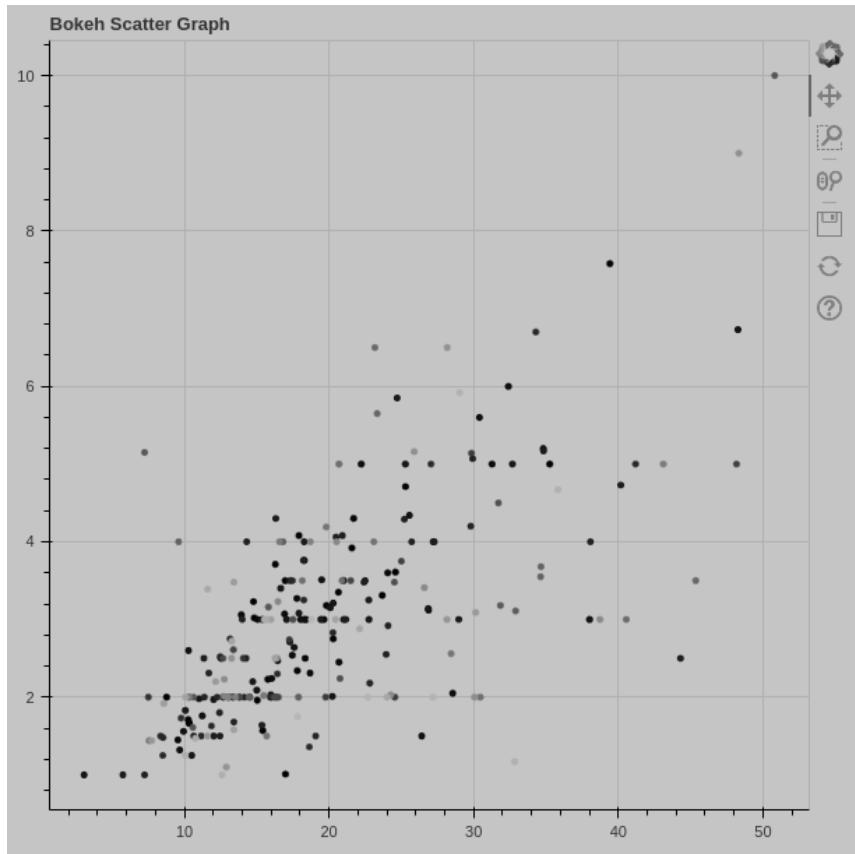
# reading the database
data = pd.read_csv("tips.csv")

color = magma(256)

# plotting the graph
graph.scatter(data['total_bill'], data['tip'], color=color)

# displaying the model
show(graph)
```

Output:



Line Chart

A line plot can be created using the line() method of the plotting module.

Example:

```
# importing the modules
from bokeh.plotting import figure, output_file, show
import pandas as pd

# instantiating the figure object
graph = figure(title = "Bokeh Bar Chart")

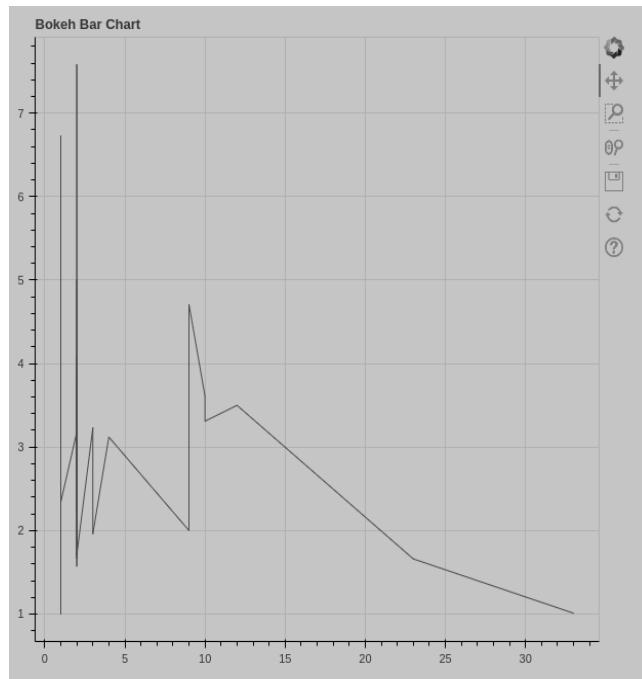
# reading the database
data = pd.read_csv("tips.csv")

# Count of each unique value of
# tip column
df = data['tip'].value_counts()

# plotting the graph
graph.line(df, data['tip'])

# displaying the model
show(graph)
```

Output:



Bar Chart

- Bar Chart can be of two types horizontal bars and vertical bars.
- Each can be created using the hbar() and vbar() functions of the plotting interface respectively.

Example:

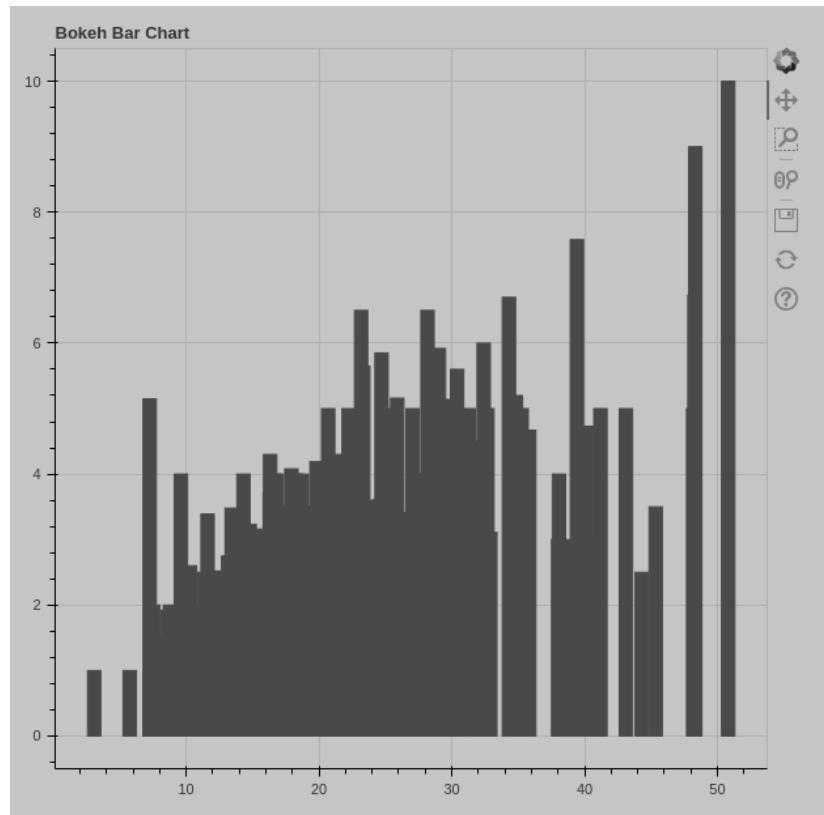
```
# importing the modules
from bokeh.plotting import figure, output_file, show
import pandas as pd

# instantiating the figure object
graph = figure(title = "Bokeh Bar Chart")

# reading the database
data = pd.read_csv("tips.csv")

# plotting the graph
graph.vbar(data['total_bill'], top=data['tip'])

# displaying the model
show(graph)
```

Output:

Interactive Data Visualization

One of the key features of Bokeh is to add interaction to the plots. Let's see various interactions that can be added.

Interactive Legends

`click_policy` property makes the legend interactive. There are two types of interactivities:

- **Hiding:** Hides the Glyphs.
- **Muting:** Hiding the glyph makes it vanish completely, on the other hand, muting the glyph just de-emphasizes the glyph based on the parameters.

Example:

```
# importing the modules
from bokeh.plotting import figure, output_file, show
import pandas as pd

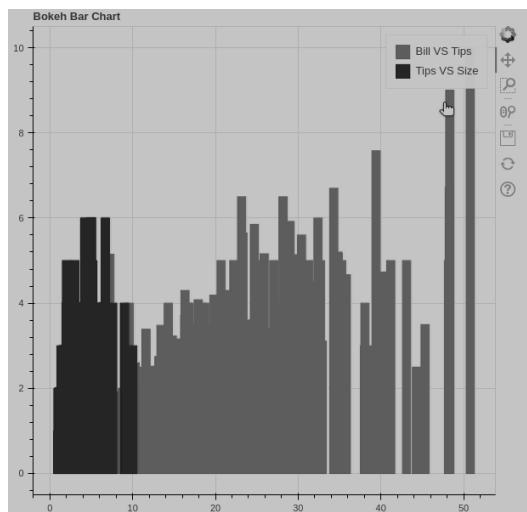
# instantiating the figure object
graph = figure(title = "Bokeh Bar Chart")

# reading the database
data = pd.read_csv("tips.csv")

# plotting the graph
graph.vbar(data['total_bill'], top=data['tip'],
            legend_label = "Bill VS Tips", color='green')
graph.vbar(data['tip'], top=data['size'],
            legend_label = "Tips VS Size", color='red')
graph.legend.click_policy = "hide"

# displaying the model
show(graph)
```

Output:



Adding Widgets

- Bokeh provides GUI features similar to HTML forms like buttons, sliders, checkboxes, etc.
- These provide an interactive interface to the plot that allows changing the parameters of the plot, modifying plot data, etc.
- Let's see how to use and add some commonly used widgets.
 - **Buttons:** This widget adds a simple button widget to the plot. We have to pass a custom JavaScript function to the CustomJS() method of the models class.
 - **CheckboxGroup:** Adds a standard check box to the plot. Similarly to buttons we have to pass the custom JavaScript function to the CustomJS() method of the models class.
 - **RadioGroup:** Adds a simple radio button and accepts a custom JavaScript function.

Example:

```
from bokeh.io import show
from bokeh.models import Button, CheckboxGroup, RadioGroup, CustomJS
button = Button(label="GFG")
button.js_on_click(CustomJS(
    code="console.log('button: click!', this.toString())"))
# Labels for checkbox and radio
# buttons
L = ["First", "Second", "Third"]
# the active parameter sets checks the selected value
# by default
checkbox_group = CheckboxGroup(labels=L, active=[0, 2])
checkbox_group.js_on_click(CustomJS(code="""
    console.log('checkbox_group: active=' + this.active, this.toString())
"""))
# the active parameter sets checks the selected value
# by default
radio_group = RadioGroup(labels=L, active=1)

radio_group.js_on_click(CustomJS(code="""
    console.log('radio_group: active=' + this.active, this.toString())
"""))
show(button)
show(checkbox_group)
show(radio_group)
```

Output:

GFG	<input checked="" type="checkbox"/> First <input type="checkbox"/> Second <input checked="" type="checkbox"/> Third	<input type="radio"/> First <input checked="" type="radio"/> Second <input type="radio"/> Third
-----	---	---

Note: All these buttons will be opened on a new tab.

Sliders: Adds a slider to the plot. It also needs a custom JavaScript function.

Example:

```
from bokeh.io import show
from bokeh.models import CustomJS, Slider
slider = Slider(start=1, end=20, value=1, step=2, title="Slider")
slider.js_on_change("value", CustomJS(code="""
    console.log('slider: value=' + this.value, this.toString())
"""))
show(slider)
```

Output:



Plotly

This is the last library of our list and you might be wondering why plotly. Here's why –

- Plotly has hover tool capabilities that allow us to detect any outliers or anomalies in numerous data points.
- It allows more customization.
- It makes the graph visually more attractive.

To install it type the below command in the terminal.

```
pip install plotly
```

A screenshot of a terminal window on a Linux system. The terminal shows the command "pip3 install plotly" being run. The output indicates that the package is already installed and up-to-date. The terminal window has a dark background with light-colored text and standard Linux window controls at the top.

Scatter Plot

Scatter plot in Plotly can be created using the [scatter\(\)](#) method of `plotly.express`. Like Seaborn, an extra data argument is also required here.

Example:

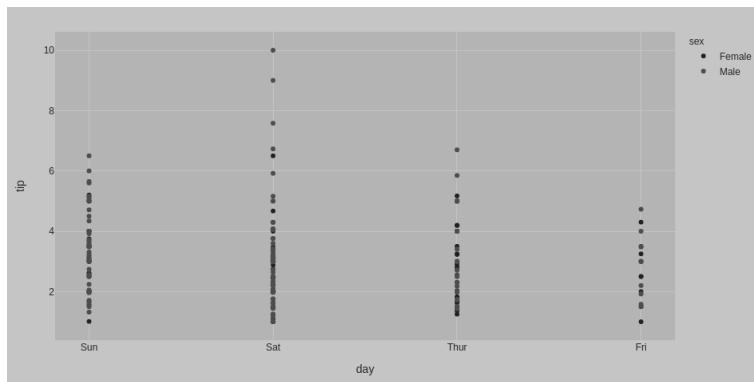
```
import plotly.express as px
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

# plotting the scatter chart
fig = px.scatter(data, x="day", y="tip", color='sex')

# showing the plot
fig.show()
```

Output:



Line Chart

- Line plot in Plotly is much accessible and illustrious annexation to plotly which manage a variety of types of data and assemble easy-to-style statistic.
- With `px.line` each data position is represented as a vertex

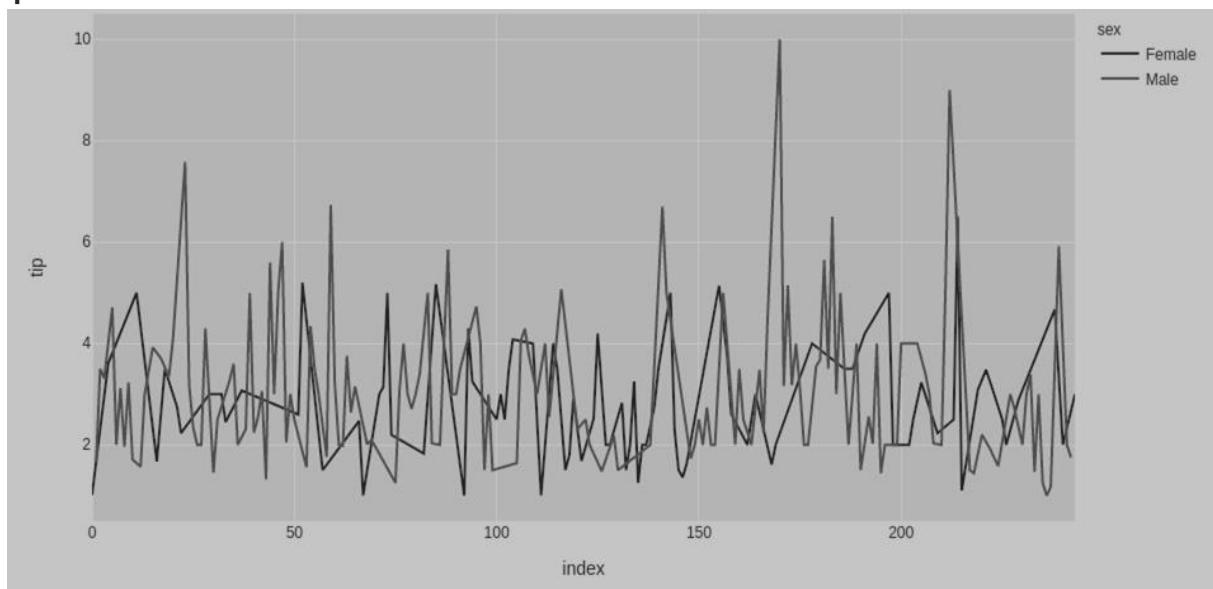
Example:

```
import plotly.express as px
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

# plotting the scatter chart
fig = px.line(data, y='tip', color='sex')

# showing the plot
fig.show()
```

Output:

Bar Chart

Bar Chart in Plotly can be created using the bar() method of plotly.express class.

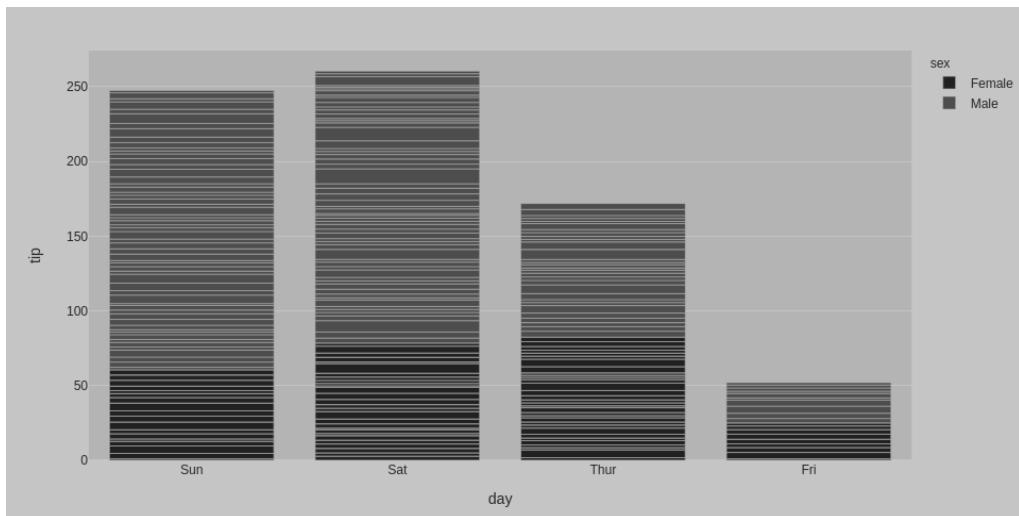
Example:

```
import plotly.express as px
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

# plotting the scatter chart
fig = px.bar(data, x='day', y='tip', color='sex')

# showing the plot
fig.show()
```

Output:

Histogram

In plotly, histograms can be created using the histogram() function of the plotly.express class.

Example:

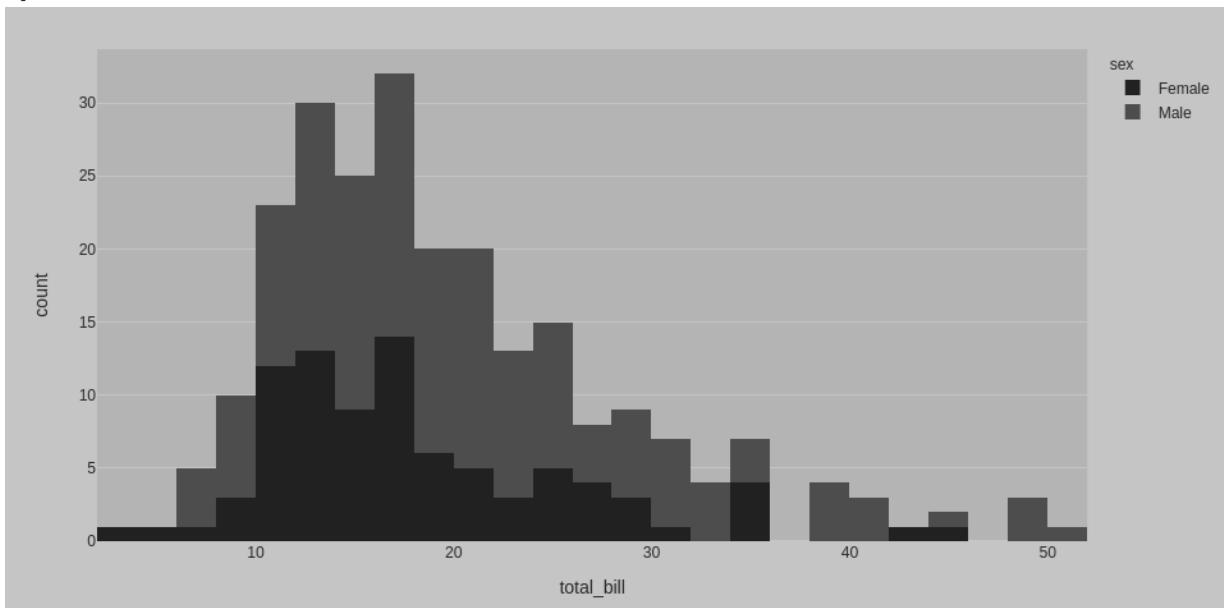
```
import plotly.express as px
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

# plotting the scatter chart
fig = px.histogram(data, x='total_bill', color='sex')

# showing the plot
fig.show()
```

Output:



Adding Interaction

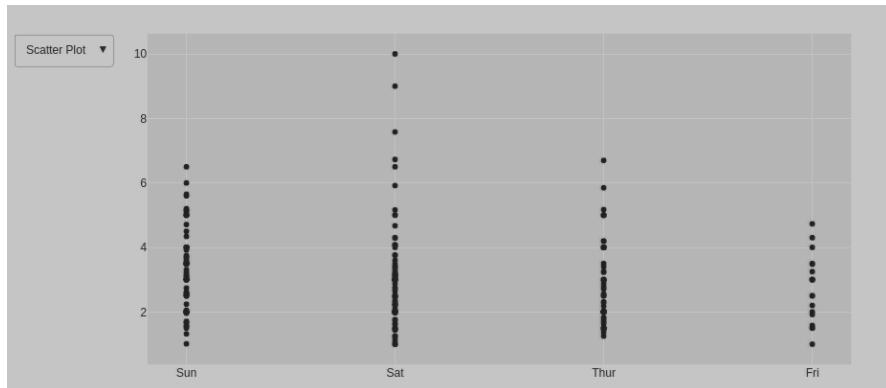
Just like Bokeh, plotly also provides various interactions. Let's discuss a few of them.

Creating Dropdown Menu

- A drop-down menu is a part of the menu-button which is displayed on a screen all the time.
- Every menu button is associated with a Menu widget that can display the choices for that menu button when clicked on it.
- In plotly, there are 4 possible methods to modify the charts by using update menu method.
 - **restyle**: modify data or data attributes
 - **relayout**: modify layout attributes
 - **update**: modify data and layout attributes
 - **animate**: start or pause an animation

Example:

```
import plotly.graph_objects as px
import pandas as pd
# reading the database
data = pd.read_csv("tips.csv")
plot = px.Figure(data=[px.Scatter(
    x=data['day'],
    y=data['tip'],
    mode='markers',)
])
# Add dropdown
plot.update_layout(
    updatemenus=[
        dict(
            buttons=list([
                dict(
                    args=["type", "scatter"],
                    label="Scatter Plot",
                    method="restyle"
                ),
                dict(
                    args=["type", "bar"],
                    label="Bar Chart",
                    method="restyle"
                )
            ]),
            direction="down",
        ),
    ]
)
plot.show()
```

Output:

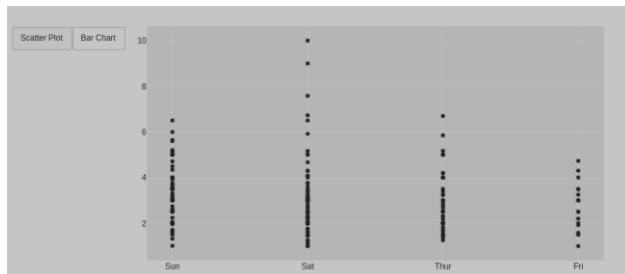
Adding Buttons

- In plotly, actions custom Buttons are used to quickly make actions directly from a record.
- Custom Buttons can be added to page layouts in CRM, Marketing, and Custom Apps.
- There are also 4 possible methods that can be applied in custom buttons:
 - **restyle**: modify data or data attributes
 - **relayout**: modify layout attributes
 - **update**: modify data and layout attributes
 - **animate**: start or pause an animation

Example:

```
import plotly.graph_objects as px
import pandas as pd
# reading the database
data = pd.read_csv("tips.csv")
plot = px.Figure(data=[px.Scatter(
    x=data['day'],
    y=data['tip'],
    mode='markers',
)])
# Add dropdown
plot.update_layout(
    updatemenus=[
        dict(
            type="buttons",
            direction="left",
            buttons=list([
                dict(
                    args=["type", "scatter"],
                    label="Scatter Plot",
                    method="restyle"
                ),
                dict(
                    args=["type", "bar"],
                    label="Bar Chart",
                    method="restyle"
                )
            ]),
        )
    ]
)
plot.show()
```

Output:



Creating Sliders and Selectors

- In plotly, the range slider is a custom range-type input control.
- It allows selecting a value or a range of values between a specified minimum and maximum range.
- The range selector is a tool for selecting ranges to display within the chart.
- It provides buttons to select pre-configured ranges in the chart.
- It also provides input boxes where the minimum and maximum dates can be manually input.

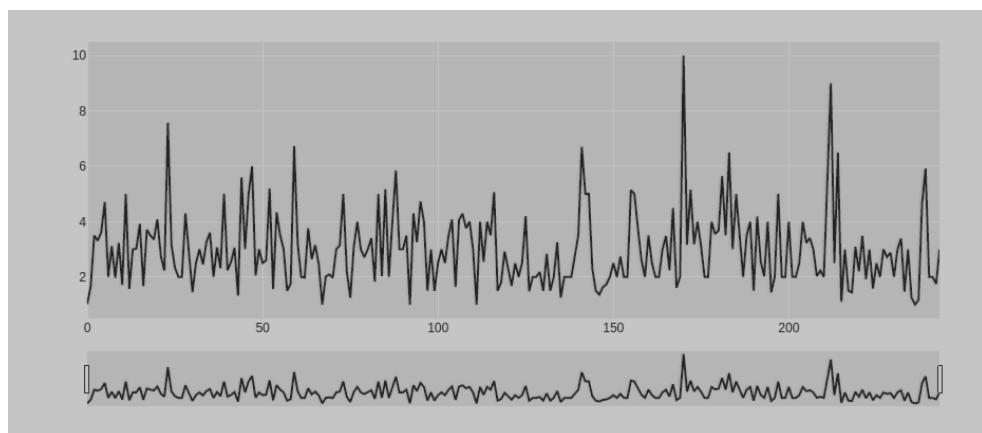
Example:

```
import plotly.graph_objects as px
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

plot = px.Figure(data=[px.Scatter(
    y=data['tip'],
    mode='lines',)
])
plot.update_layout(
    xaxis=dict(
        rangeselector=dict(
            buttons=list([
                dict(count=1,
                    step="day",
                    stepmode="backward"),
            ])
        ),
        rangeslider=dict(
            visible=True
        )
    )
)
plot.show()
```

Output:



3.9 Introduction to Model Building

- A model can refer to an equation that helps us to predict the outcomes.

Linear Regression & Multiple Linear Regression

- **Linear Regression** - As the name suggests, it involves only a single independent variable to make a prediction.
- **Multiple Regression** - It involves multiple independent variables to make a prediction.

The equation for a simple linear regression can be represented as-

$$y = b_0x + b_1$$

Here,

y-dependent variable

x - independent variable

b_0 -slope

b_1 -intercept

To implement Linear Regression in Python-

```
from sklearn.linear_model import Linear Regression  
lm=LinearRegression()  
X=df["attribute-name1"]  
Y=df["attribute-name1"]  
lm.fit(X,Y)  
yp=lm.predict(X)
```

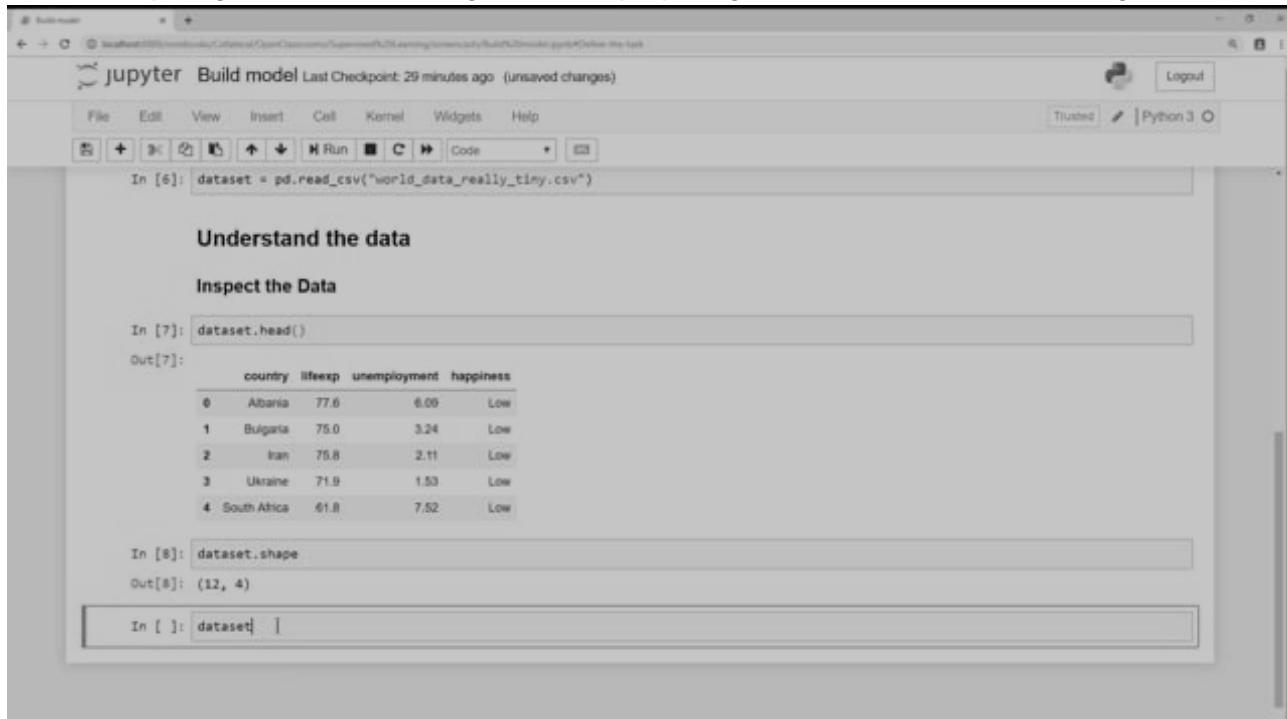
Import Libraries

- Start by importing the necessary libraries, including pandas, NumPy, and Matplotlib, to give you data manipulation and visualization capabilities.
- Then you'll import a few capabilities from **scikit**-learn (also called `sklearn`), which is the Python machine learning library we will be using.
- Finally, you'll import a library called *functions.py*.

```
# Import Python libraries for data manipulation and visualization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Import the Python machine learning libraries we need
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
# Import some convenience functions.
```

- This can be found on the course github from `functions import*`
- The screencast below explains the first few stages of the supervised learning process: defining the task, acquiring and understanding data, and preparing the data for machine learning.

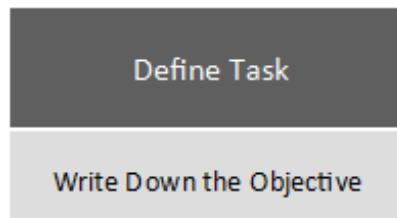


The screenshot shows a Jupyter Notebook interface with the following content:

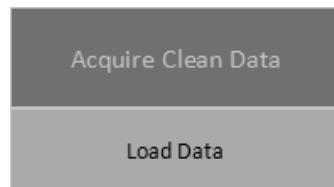
- Header: "jupyter Build model Last Checkpoint: 29 minutes ago (unsaved changes)"
- Toolbar: File, Edit, View, Insert, Cell, Kernel, Widgets, Help
- Cell 6: In [6]: `dataset = pd.read_csv("world_data_really_tiny.csv")`
- Section: Understand the data
- Section: Inspect the Data
- Cell 7: In [7]: `dataset.head()`
Out[7]:

	country	lifeexp	unemployment	happiness
0	Albania	77.6	6.09	Low
1	Bulgaria	75.0	3.24	Low
2	Iran	75.8	2.11	Low
3	Ukraine	71.9	1.53	Low
4	South Africa	61.8	7.52	Low

- Cell 8: In [8]: `dataset.shape`
Out[8]: (12, 4)
- Cell 9: In [9]: `dataset`

1. Define the Task**Remember the task:**

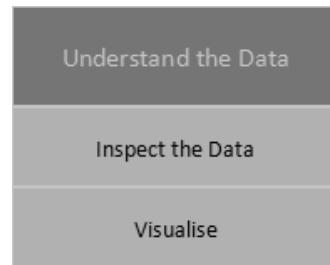
Use life expectancy and long-term unemployment rate to predict the perceived happiness (low or high) of inhabitants of a country.

2. Acquire Clean Data

Let's load the data from the CSV file using pandas:

```
# Load the data set  
dataset = pd.read_csv("world_data_really_tiny.csv")
```

3. Understand the Data



1. Inspect the Data

Use the `head()` function to show the first 12 rows (which is the entire dataset). Again, this is far too small for any real machine learning activity, but it serves our purpose in this learning exercise.

```
# Inspect first few rows  
dataset.head(12)  
Look at the Data Shape
```

Confirm the number of rows and columns in the data:

```
# Inspect data shape  
dataset.shape  
Compute Descriptive Statistics
```

Computing the descriptive stats can be done in one function call:

```
# Inspect descriptive stats  
dataset.describe()
```

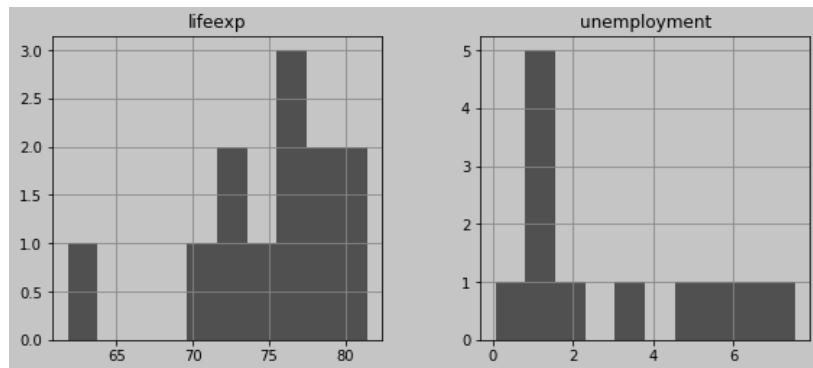
	lifeexp	unemployment
count	12.000000	12.000000
mean	74.833333	3.051667
std	5.213328	2.377664
min	61.800000	0.060000
25%	71.900000	1.412500
50%	75.750000	1.820000
75%	77.525000	5.102500
max	81.400000	7.520000

2. Visualize the Data

Use the `histPlotAll()` function from `functions.py` to plot a histogram for each numeric feature:

```
# View univariate histogram plots
```

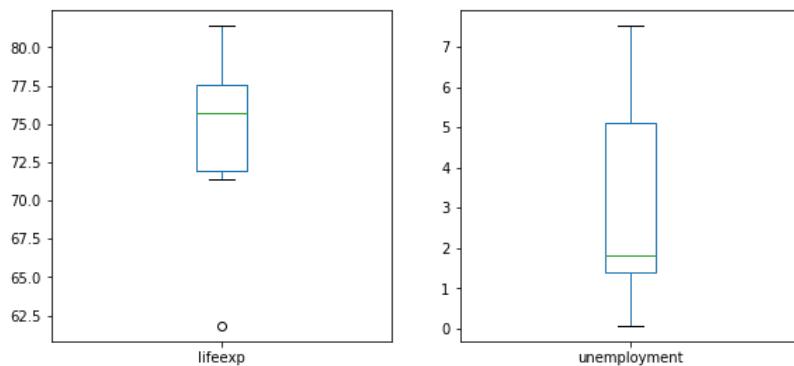
```
histPlotAll(dataset)
```



Use the `boxPlotAll()` function from `functions.py` to plot a box plot for each numeric feature:

```
# View univariate box plots
```

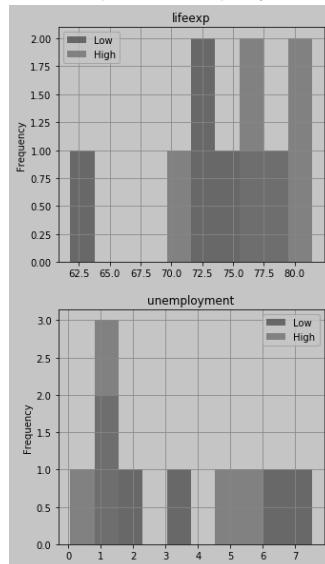
```
boxPlotAll(dataset)
```



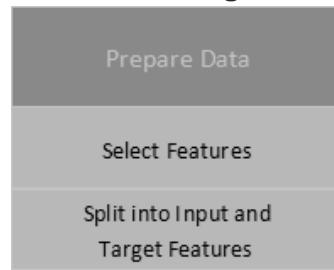
And the `classComparePlot()` function from `functions.py` to plot a comparative histogram for the two classes:

```
# View class split
```

```
classComparePlot(dataset[["happiness", "lifeexp", "unemployment"]], 'happiness', plotType='hist')
```



3. Prepare the Data for Supervised Machine Learning



Select Features and Split Into Input and Target Features

We can select happiness as the feature to predict (yy) and *lifeexp* and *unemployment* as the features to make the prediction (XX):

```
# Split into input and output features
```

```
y = dataset["happiness"]
```

```
X = dataset[["lifeexp", "unemployment"]]
```

```
X.head()
```

	lifeexp	unemployment
0	77.6	6.09
1	75.0	3.24
2	75.8	2.11
3	71.9	1.53
4	61.8	7.52

```
y.head()
```

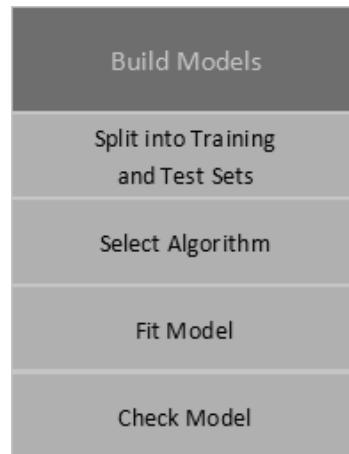
0	Low
1	Low
2	Low
3	Low
4	Low

Name: happiness, dtype: object

The screencast below explains the next stages: building and interpreting the model. Watch first, then read the notes below.

The screenshot shows a Jupyter Notebook interface with several code cells:

- Cell 24:** Prints accuracy score of 1.0.
- Cell 25:** Prints array(['Low', 'Low', 'High', 'Low'], dtype=object).
- Cell 26:** Prints accuracy score of 0.25.
- Cell 27:** Prints a copy of X_test with columns 'Actual' and 'Predictions' added.



i. Split Into Training and Test Sets

Use the `train_test_split()` function in `sklearn` to split the sample set into a training set, which we will use to train the model, and a test set, to evaluate the model:

```
# Split into test and training sets
test_size = 0.33
seed = 7
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=seed)
```

Let's look at the four samples produced and confirm the randomness of the selection:

`X_train`

	lifeexp	unemployment
0	77.6	6.09
1	75.0	3.24
11	77.5	0.06
8	80.7	1.36
3	71.9	1.53
6	81.4	1.43
9	75.7	4.96
4	61.8	7.52

`X_test`

	lifeexp	unemployment
7	77.3	5.53
10	71.4	1.26
2	75.8	2.11
5	71.9	1.53

y_train

```
0      Low
1      Low
11     High
8      High
3      Low
6      High
9      High
4      Low
Name: happiness, dtype: object
```

y_test

```
7      High
10     High
2      Low
5      Low
Name: happiness, dtype: object
```

ii. Select an Algorithm

Create a model using the sklearn decision tree algorithm:

```
# Select algorithm
model = DecisionTreeClassifier()
```

iii. Fit the Model to the Data

Now take the *training set* and use it to fit the model (i.e., train the model):

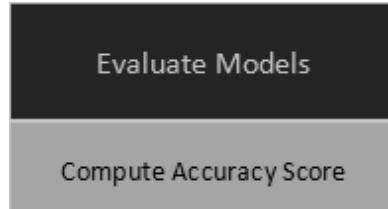
```
# Fit model to the data
model.fit(X_train, y_train)
```

iv. Check the Model

Next, assess how well the model predicts happiness using the *training data*, by “pouring” training set X into the decision tree:

```
# Check model performance on training data
predictions = model.predict(X_train)
print(accuracy_score(y_train, predictions))
```

Evaluate the Model



Compute Accuracy Score

Let's pour test set XX into the decision tree and see what it predicts:

```
# Evaluate the model on the test data
```

```
predictions = model.predict(X_test)
```

Look at the predictions it has made:

```
predictions
```

```
array(['Low', 'High', 'High', 'Low'], dtype=object)
```

And compute the accuracy score:

```
print(accuracy_score(y_test, predictions))
```

We can show the model predictions with the original data, with the actual happiness value:

```
df = X_test.copy()
```

```
df['Actual'] = y_test
```

```
df['Prediction'] = predictions
```

```
df
```

	lifeexp	unemployment	Actual	Prediction
7	77.3	5.53	High	Low
10	71.4	1.26	High	High
2	75.8	2.11	Low	High
5	71.9	1.53	Low	Low

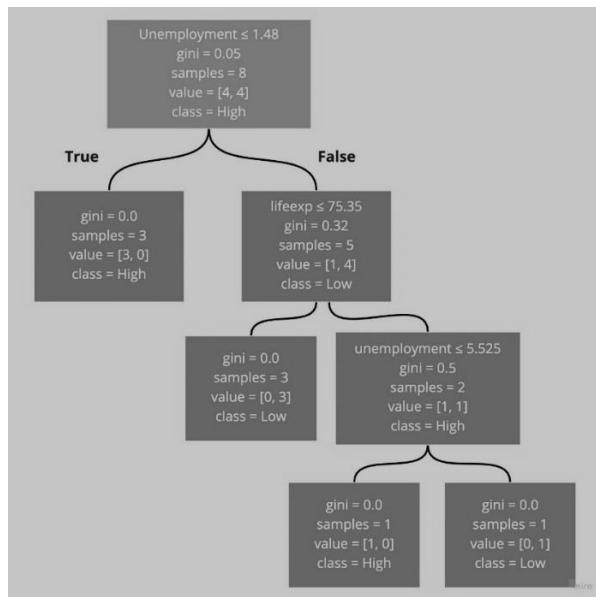
As you can see, the model hasn't performed too well on the test data. It's often the case that they perform worse, and it shouldn't be surprising.

What Rules Did Sklearn Come Up With?

- At this point, the model produced by sklearn is a bit of a black box.
- There is no set of rules to examine, but it is possible to inspect the model and visualize the rules.
- For this code to work, you need to first install *Graphviz* by running the following from a terminal session:
 conda install python-graphviz
- You can then run the following code, which uses a function from functions.py:

```
viewDecisionTree(model, X.columns)
```

And see the decision tree rules:



Section 3: Exercises

Exercise 1: Fill the following table.

Operator	Description
% (reminder)	
** (Exponent)	
!=	
+=	
//=	
^ (binary xor)	
not	
* / % //	

Exercise 2: Participate in a group discussion on following topics:

- a) Python 2 vs Python 3
- b) Java vs Python Program
- c) Python Popular Frameworks and Libraries
- d) Python Operators and Loops
- e) Python Applications
- f) Statistical Computing

Section 4: Assessment Questionnaire

1. What are the features of Python?
2. What are Python applications?
3. Is Python a dynamically-typed language or statically-typed?
4. What are the data types of Python?
5. How do you reverse a list in Python? When is it useful?
6. What does break and continue do in Python?
7. Can break and continue be used together?
8. What will be the output of the code below?

```
list = ['alfa', 'bravo', 'charlie', 'delta', 'echo']
print list[10:]
```
9. Explain generators vs iterators.
10. Does Python need to be compiled before it is run?
11. Is Python Call by Value or Call by Reference? How are arguments passed by value or by reference?
12. What is the lambda function?
13. Enlist Python operators?
14. What are the two types of variables in Python?
15. Python _____ can be defined as data that is given in a variable or constant.
16. What is web scrapping?
17. The _____ function is used to return the middle value of the numeric data in the list.
18. _____ measures the scope to which two variables are interdependent.
19. _____ is a Python library that is written in Python, but the parts that require fast computation are written in C or C++.
20. What are the common data visualization libraries in Python?

-----End of the Module-----

MODULE 4

TABLEAU TRAINING

Section 1: Learning Outcomes

After completing this module, you will be able to:

- Explain Features and Applications of Tableau
- Describe Tableau Architecture
- Download and Install Tableau
- Use various Functions of Tableau
- Create Charts in Tableau
- Use Tableau File Types and Operators
- Apply Filters and Analytics in Tableau
- Create Dashboard in Tableau
- Connect Data with Data Sources
- Use LOD (Level of Details) Expression

Section 2: Relevant Knowledge

4.1 Introduction to Tableau

- Tableau is the fastly growing and powerful data visualization tool.
- Tableau is a business intelligence tool which helps us to analyze the raw data in the form of the visual manner; it may be a graph, report, etc.
- If you have any data like Big Data, Hadoop, SQL, or any cloud data and if you want to analyze that given data in the form of pictorial representation of data, you can use Tableau.
- Data analysis is very fast with Tableau, and the visualizations created are in the form of worksheets and dashboards.
- Any professional can understand the data created using Tableau.
- Tableau software doesn't require any technical or any programming skills to operate. Tableau is easy and fast for creating visual dashboards.

Why use Tableau?

Here are some reasons to use Tableau:

- Ultimate skill for Data Science
- User-Friendly
- Apply to any Business
- Fast and Easy
- You don't need to do any Coding
- Community is Huge
- Hold the power of data
- It makes it easier to understand and explain the Data Reports

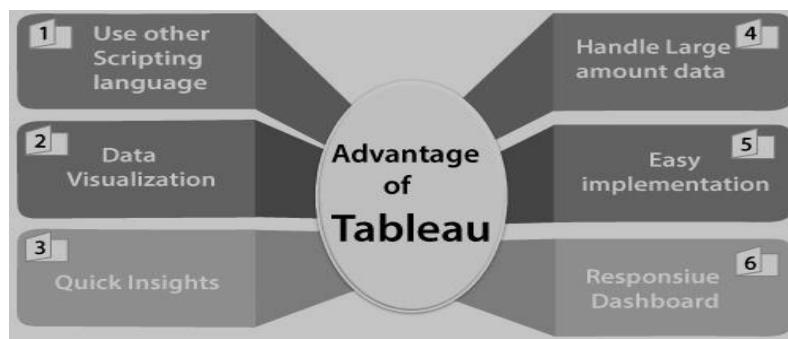
Features of Tableau

- **Data Blending:** Data blending is the most important feature in Tableau. It is used when we combine related data from multiple data sources, which you want to analyze together in a single view, and represent in the form of a graph.

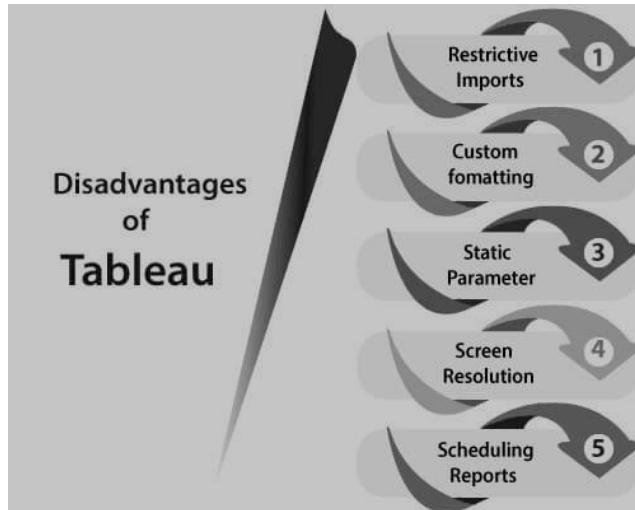
Example: Assume, we have Sales data in relational database and Sales Target data in an Excel sheet. Now, we have to compare actual sales with target sales, and blend the data based on common dimensions to get access. The two sources which are involved in data blending referred to as primary data and secondary data sources. A left join will be created between the primary data source and the secondary data source with all the data rows from primary and matching data rows from secondary data source to blend the data.

- **Real-time analysis:** Real-Time Analysis makes users able to quickly understand and analyze dynamic data, when the Velocity is high, and real-time analysis of data is complicated. Tableau can help extract valuable information from fast moving data with interactive analytics.
- **The Collaboration of data:** Data analysis is not isolating task. That's why Tableau is built for collaboration. Team members can share data, make follow up queries, and forward easy-to-digest visualizations to others who could gain value from the data. Making sure everyone understands the data and can make informed decisions is critical to success.

Advantages of Tableau



- **Data Visualization:** Tableau is a data visualization tool, and provides complex computation, data blending, and dashboarding for creating beautiful data visualizations.
- **Quickly Create Interactive Visualization:** Users can create a very interactive visual by using drag and drop functionalities of Tableau.
- **Comfortable in Implementation:** Many types of visualization options are available in Tableau, which enhances the user experience. Tableau is very easy to learn in comparison to Python. Who don't have any idea about coding, they also can quickly learn Tableau.
- **Tableau can Handle Large Amounts of Data:** Tableau can easily handle millions of rows of data. A large amount of data can create different types of visualization without disturbing the performance of the dashboards. As well as, there is an option in Tableau where the user can make 'live' to connect different data sources like SQL, etc.
- **Use of other Scripting Language in Tableau:** To avoid the performance issues and to do complex table calculations in Tableau, users can include Python or R. Using Python Script, user can remove the load of the software by performing data cleansing tasks with packages. However, Python is not a native scripting language accepted by Tableau. So, you can import some of the packages or visuals.
- **Mobile Support and Responsive Dashboard:** Tableau Dashboard has an excellent reporting feature that allows you to customize dashboard specifically for devices like a mobile or laptops. Tableau automatically understands which device is viewing the report by the user and make adjustments to ensure that accurate report is delivered to the right device.



- **Scheduling of Reports:** Tableau does not provide the automatic schedule of reports. That's why there is always some manual effort required when the user needs to update the data in the back end.
- **No Custom Visual Imports:** Other tools like Power BI, a developer can create custom visual that can be easily imported in Tableau, so any new visuals can recreate before imported, but Tableau is not a complete open tool.
- **Custom Formatting in Tableau:** Tableau's conditional formatting, and limited 16 column table that is very inconvenient for users. Also, to implement the same format in multiple fields, there is no way for the user that they can do it for all fields directly. Users have to do that manually for each, so it is a very time-consuming.
- **Static and Single Value Parameter:** Tableau parameters are static, and it always select a single value as a parameter. Whenever the data gets changed, these parameters also have to be updated manually every time. There is no other option for users that can automate the updating of parameters.
- **Screen Resolution on Tableau Dashboards:** The layout of the dashboards is distributed if the Tableau developer screen resolution is different from users screen resolution.

Example: If the dashboard is created on the screen resolution of 1920 X 1080 and it viewed on 2560 X 1440, then the layout of the dashboard will be destroyed a little bit, their dashboard is not responsive. So, you will need to create a dashboard for desktop and mobile differently.

4.2 Tools of Tableau

A list of Tableau tools:

- Tableau Desktop
- Tableau Public
- Tableau Online
- Tableau Server
- Tableau Reader



Data analytics in Tableau is classified into two parts:

1. **Developer Tools:** The Tableau tools which are used for development such as the creation of charts, dashboards, report generation and visualization are known as developer's tools. Tableau Desktop and the Tableau Public, are the example of this type.
2. **Sharing Tools:** The role of these tools is sharing the reports, visualizations, and dashboards that were created using the developer tools. The Tableau tools that fall into this category are Tableau Server, Tableau Online, and Tableau Reader.

Let's see all the Tools one by one:

Tableau Desktop

- Tableau Desktop has a rich feature set and allows us to code and customize reports.
- Right from creating the reports, charts to blending them all to form a dashboard, all the necessary work is created in Tableau Desktop.
- For live data analysis, Tableau Desktop establish connectivity between the Data Warehouse and other various types of files.
- The dashboards and the workbooks created here can be either shared locally or publicly.

Based on the connectivity to the publishing option and data sources, Tableau Desktop is also classified into two parts-

- **Tableau Desktop Personal:** The personal version of the Tableau desktop keeps the workbook private, and the access is limited. The workbooks can't be published online. So, it should be distributed either offline or in Tableau public.
- **Tableau Desktop Professional:** It is similar to Tableau desktop. The main difference is that the workbooks created in the Tableau desktop can be published online or in Tableau server. In the professional version, there is full access to all sorts datatypes. It is best for those who want to publish their workbook in Tableau server.

Tableau Public

- This Tableau version is specially built for cost-effective users.
- The word '**Public**' means that the created workbooks cannot be saved locally.
- They should be kept on the Tableau's public cloud, which can be accessed and viewed by anyone.
- There is no privacy of the files saved on the cloud, so anyone can access and download the same data.
- This version is the best for them who want to share their data with the general public and for the individuals who want to learn Tableau.

Tableau Online

- Its functionality is similar to the tableau server, but data is stored on the servers that hosted on the cloud, which is maintained by the Tableau group.
- There is no storage limit on the data which is published in the Tableau Online.
- Tableau Online creates a direct link over 40 data sources who are hosted in the cloud such as the Hive, MySQL, Spark SQL, Amazon Aurora, and many more.
- To be published, both Tableau Server and Tableau online require the workbooks that are created by Tableau Desktop.
- Data that flow from the web applications say Tableau Server and Tableau Online also support Google Analytics and Salesforce.com.

Tableau Server

- The software is correctly used to share the workbooks, visualizations, which is created in the Tableau Desktop application over the organization.
- To share dashboards in the Tableau Server, you should first publish your workbook in the Tableau Desktop. Once the workbook has been uploaded to the server, it will be accessible only to the authorized users.
- It's not necessary that the authorized users have the Tableau Server installed on their machine. They only require the login credentials by which they can check reports by the web browser.
- The security is very high in Tableau server, and it is beneficial for quick and effective sharing of data.
- The admin of the organization has full control over the server.
- The organization maintains the hardware and the software.

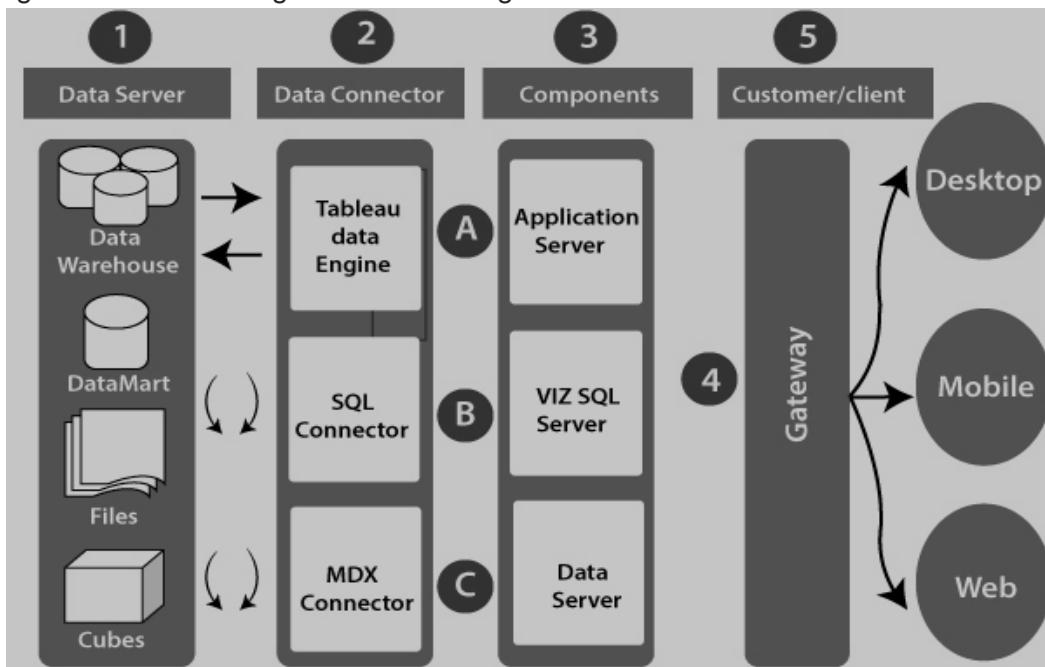
Tableau Reader

- Tableau Reader is a free tool which allows us to view the visualizations and workbooks, which is created using Tableau Desktop or Tableau Public.
- The data can be filtered, but modifications and editing are restricted.
- There is no security in Tableau Reader as anyone can view workbook using Tableau Reader.
- If you want to share the dashboards which are created by you, the receiver should have Tableau Reader to view the document.

4.3 Tableau Architecture

- Tableau Server is designed to connect many data tiers.
- It can connect clients from Mobile, Web, and Desktop.
- Tableau Desktop is a powerful data visualization tool.
- It is very secure and highly available.
- It can run on both the physical machines and virtual machines.
- It is a multi-process, multi-user, and multi-threaded system.

Providing such powerful features requires unique architecture. The different layers used in Tableau server are given in the following architecture diagram:



Let's study about the different component of the Tableau architecture:

1. Data Server

- The primary component of Tableau Architecture is the Data sources which can connect to it.
- Tableau can connect with multiple data sources. It can blend the data from various data sources.
- It can connect to an excel file, database, and a web application at the same time.
- It can also make the relationship between different types of data sources.

2. Data Connector

- The Data Connectors provide an interface to connect external data sources with the Tableau Data Server.
- Tableau has in-built SQL/ODBC connector.
- This ODBC Connector can be connected with any databases without using their native connector.
- Tableau desktop has an option to select both extract and live data.
- On the uses basis, one can be easily switched between live and extracted data.

a. Real-time data or live connection

- Tableau can be connected with real data by linking to the external database directly.
- It uses the infrastructure existing database by sending dynamic multidimensional expressions (MDX) and SQL statements.
- This feature can be used as a linking between the live data and Tableau rather than importing the data.
- It makes optimized and a fast database system.
- Mostly in other enterprises, the size of the database is large, and it is updated periodically.
- In these cases, Tableau works as a front-end visualization tool by connecting with the live data.

b. Extracted or in-memory data:

- Tableau is an option to extract the data from external data sources.
- We make a local copy in the form of Tableau extract file.
- It can remove millions of records in the Tableau data engine with a single click.
- Tableau's data engine uses storage such as ROM, RAM, and cache memory to process and store data.
- Using filters, Tableau can extract a few records from a large dataset.
- This improves performance, especially when we are working on massive datasets.
- Extracted data allows the users to visualize the data offline, without connecting to the data source.

3. Components of Tableau Server:

Different types of components of the Tableau server are:

- Application server
- VizQL server
- Data server

a. Application Server

- The application server is used to provide the authorizations and authentications.
- It handles the permission and administration for mobile and web interfaces.
- It gives a guarantee of security by recording each session id on Tableau Server.
- The administrator is configuring the default timeout of the session in the server.

b. VizQL server

- VizQL server is used to convert the queries from the data source into visualizations.
- Once the client request is forwarded to the VizQL process, it sends the query directly to the data source retrieves information in the form of images.
- This visualization or image is presented for the users.
- Tableau server creates a cache of visualization to reduce the load time.
- The cache can be shared between many users who have permission to view the visualization.

c. Data server:

Data server is used to store and manage the data from external data sources. It is a central data management system. It provides data security, metadata management, data connection, driver requirements, and data storage. It stores the related details of data set like calculated fields, metadata, groups, sets, and parameters. The data source can extract the data as well as make live connections with external data sources.

4. Gateway

- The gateway directed the requests from users to Tableau components.
- When the client sends a request, it is forwarded to the external load balancer for processing.
- The gateway works as a distributor of processes to different components. In case of absence of external load balancer, the gateway also works as a load balancer.
- For single server configuration, one gateway or primary server manages all the processes.
- For multiple server configurations, one physical system works as a primary server, and others are used as worker servers.
- Only one machine is used as a primary server in Tableau Server environment.

5. Clients

The visualizations and dashboards in Tableau server can be edited and viewed using different clients. Clients are a web browser, mobile applications, and Tableau Desktop.

- **Web Browser:** Web browsers like Google Chrome, Safari, and Firefox support the Tableau server. The visualization and contents in the dashboard can be edited by using these web browsers.
- **Mobile Application:** The dashboard from the server can be interactively visualized using mobile application and browser. It is used to edit and view the contents in the workbook.
- **Tableau Desktop:** Tableau desktop is a business analytics tool. It is used to view, create, and publish the dashboard in Tableau server. Users can access the various data source and build visualization in Tableau desktop.

4.4 Download and Installation of Tableau

Tableau is available in two ways:

- Tableau Public (Free)
- Tableau Desktop (Commercial)

Here is a comparison between the Tableau Public and Tableau Desktop

Tableau Public

- Tableau Public is a free and open-source.
- Tableau public data source can connect to Excel and Text files.
- Tableau public can be installed on Window and Mac operating system.
- Data and Visualizations are not secured in the Tableau public because it is available in public.
- In Tableau public, data cannot be obtained from different data sources as it is limited to connect only Excel and Text files.
- Tableau public uses the details at Personal level.

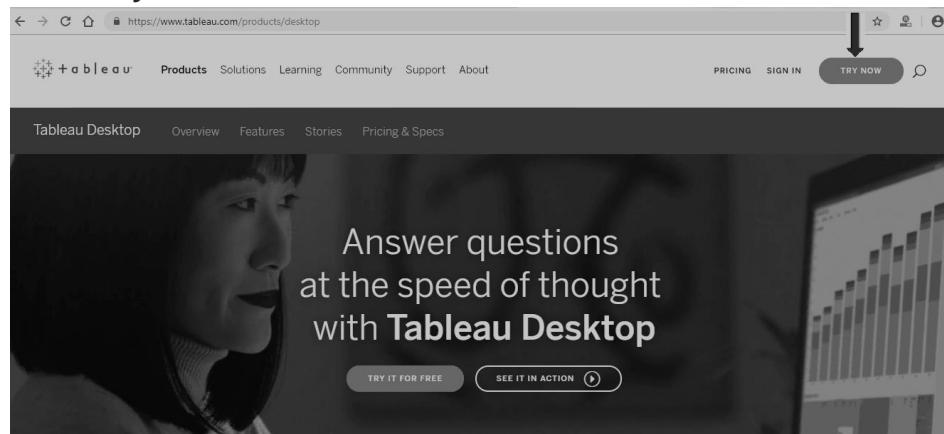
Tableau Desktop

- Tableau Desktop is a paid source, personal edition- \$35 per month and professional edition- \$70 per month.
- Tableau desktop data source can connect to any data source file, including databases, web applications, and more.
- Tableau desktop can also install on Window and Mac operating system.
- Data and Visualization are secured in Tableau desktop.
- In Tableau desktop, data can extract from various data sources and stored as Tableau extract file.
- Tableau desktop uses the details at Professional and Enterprise level.

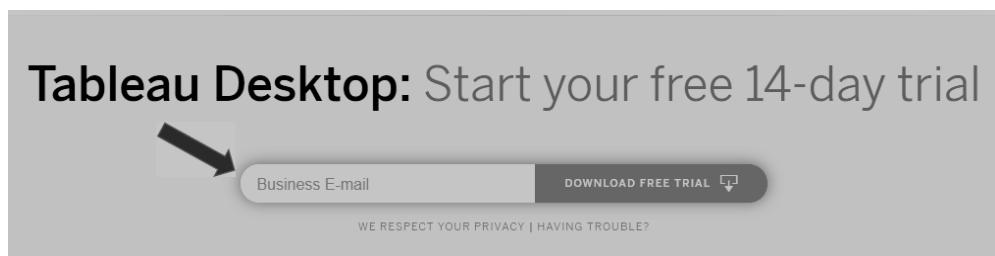
Let's install the Tableau Desktop on Window machine and go through step by step:

Step 1: Go to <https://www.tableau.com/products/desktop> on your Web browser.

Step 2: Click on the 'Try Now' button.

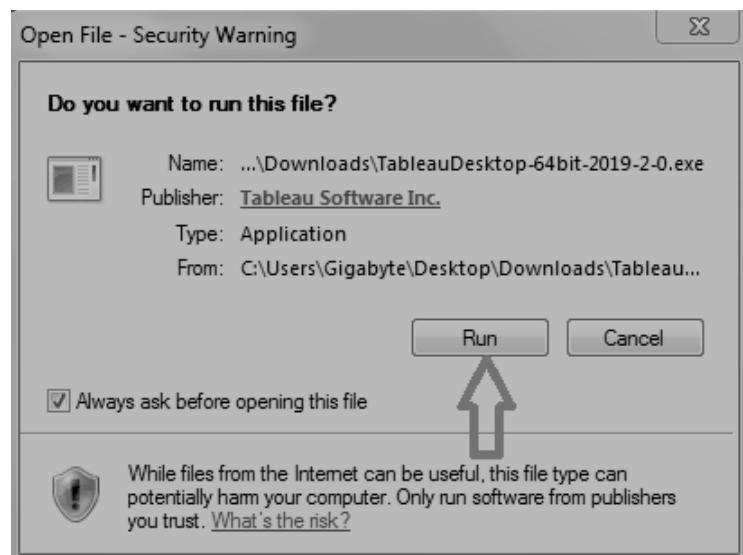


Step 3: Now, enter your **Email id** and click on the '**Download Free Trial**' button.

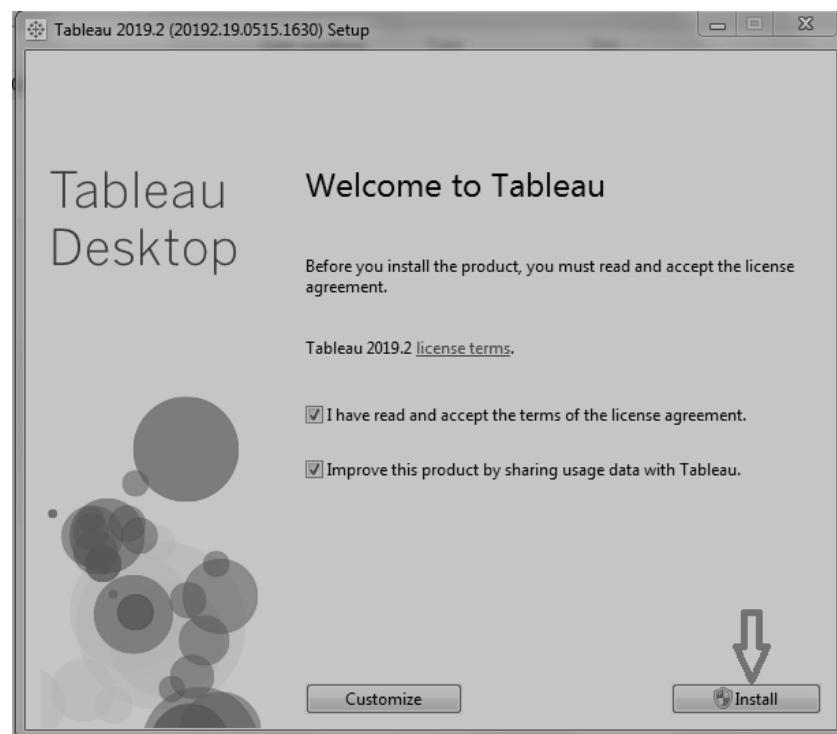


Step 4: This will start downloading the .exe File for window machine by default.

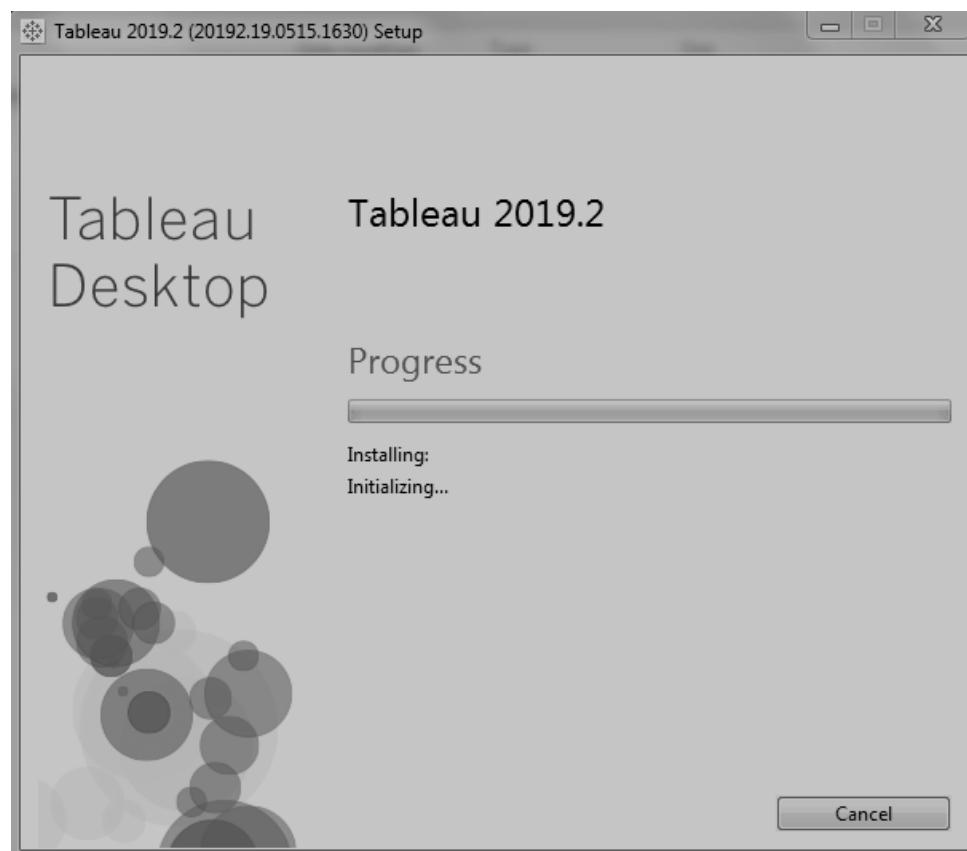
Step 5: Open the download file, and click on the '**Run**' button.



Step 6: Accept the terms and condition and click on '**Install**' button.



Step 7: A pop message will be shown on the screen to get the approval of the administrator to install the Tableau software. Click on '**yes**' to approve it than installation will be started.

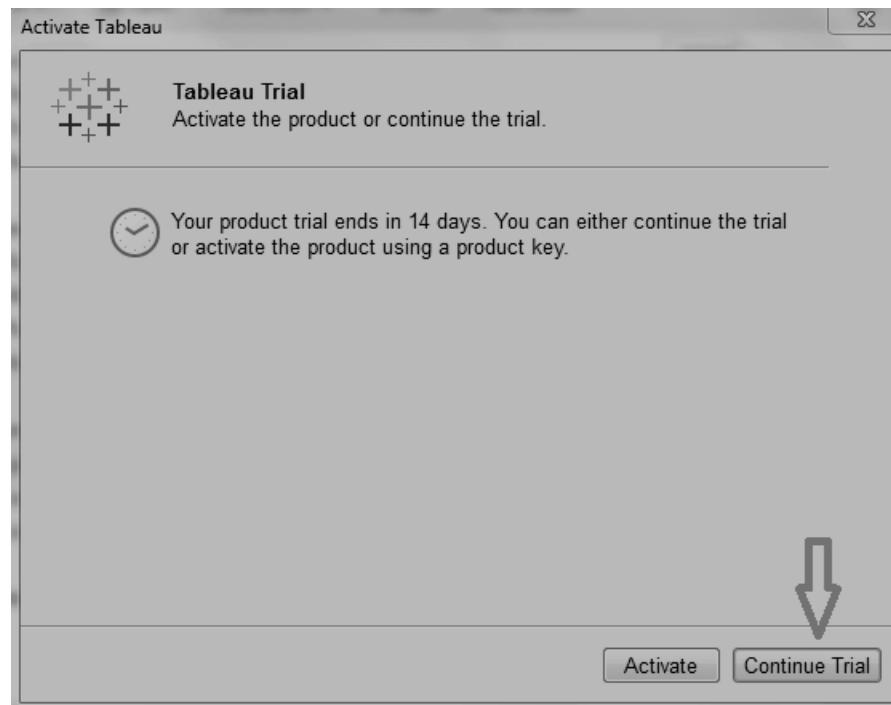


Step 8: Once the installation is completed, then open the Tableau desktop software.

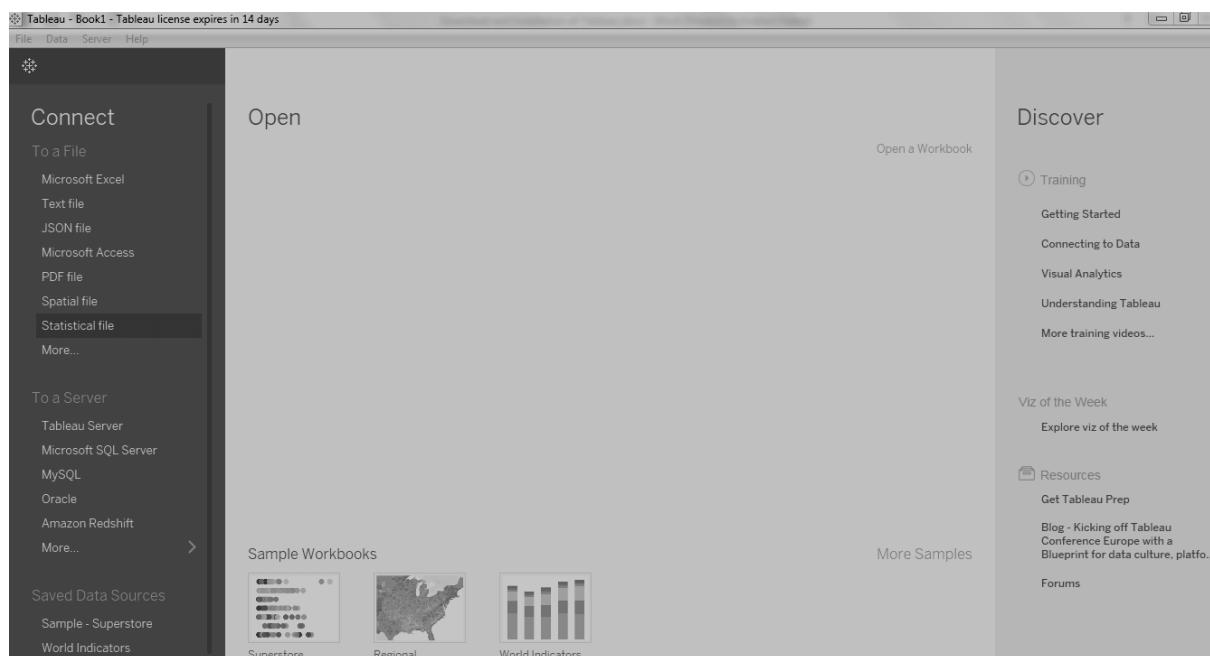
Step 9: In the registration window

1. Click on Activate Tableau and fill your complete details.
2. Click on start trial now.

Step 10: Wait for complete registration.



Step 11: Start screen of the Tableau Desktop.



Now, you are all set to use your Tableau desktop on your window machine.

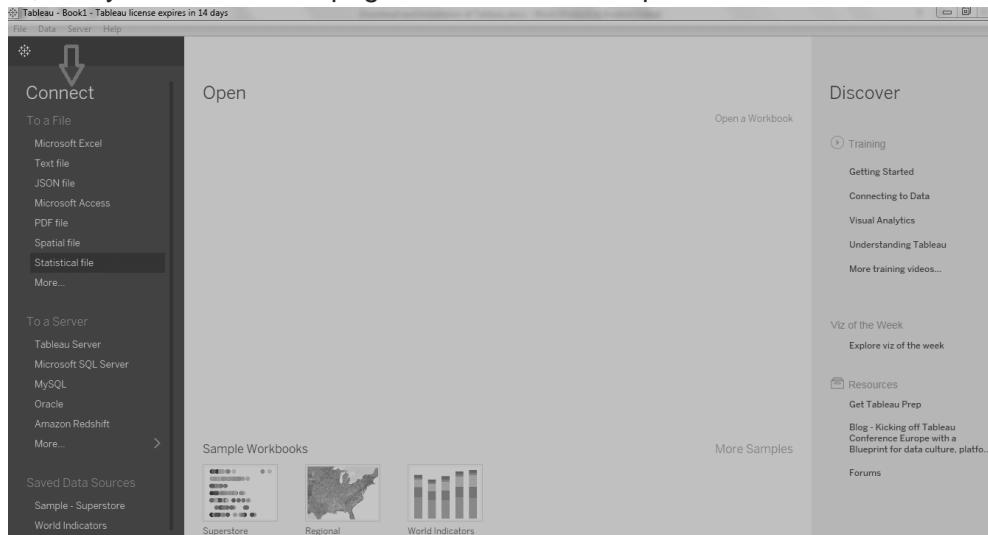
Copy rights reserved for STL Academy

4.5 Using the Tableau Workspace Control Effectively

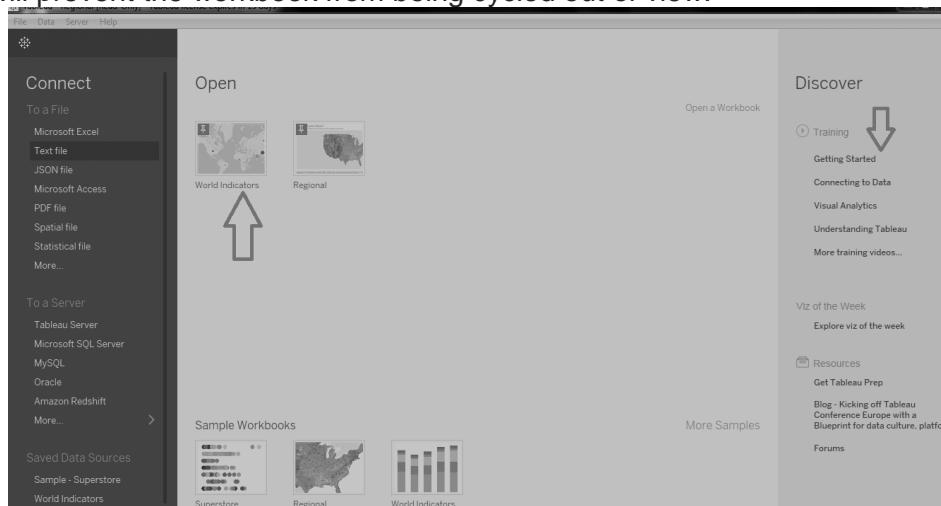
If you are addicted to working with spreadsheets or other analysis tools, learning Tableau's desktop environment will be helpful. If you have no familiarity with spreadsheets or database terminology, you can still be effectively using Tableau within a few days.

The Data Connection Page and Start Page

Open Tableau, and you see the start page of Tableau Desktop.



- On the left side, the data window gives connection options.
- If you click on that to connect to the Data, you are taken to the data connection workspace.
- You can also access this page by clicking on the hard disk tab which is next to the Start button.
- If you want to connect to one of the data sources listed On a Server section, you must go to Tableau's website and download a connector for the required database. Here is no limit on the number of data connection drivers you can install, but some dealer requires that you validate a valid license to their software before downloading their connector.
- On the right side of the Connect to the Data page, you will see saved data connections.
- Tableau provides four as sample data for learning.
- Any other links you have collected (.tds files) are displayed there as well. Return to the Home button and look at the Workbooks area in the start page.
- The Workbooks area saves the last nine workbooks you've opened.
- If you want to keep a workbook there that you frequently use, go over the workbook image and click on the push pin.
- That will prevent the workbook from being cycled out of view.



- To remove saved workbooks from the start page click on the red X that appears when you float over the workbook's image.
- At the bottom of this start page, the Getting Started area provides links to training videos and promotional materials.
- The sample workbook area provides links to sample workbooks containing excellent example material.
- Clicking on More Samples takes you to Tableau's visual gallery on the web with even more example workbooks.

Tableau Desktop Workspace

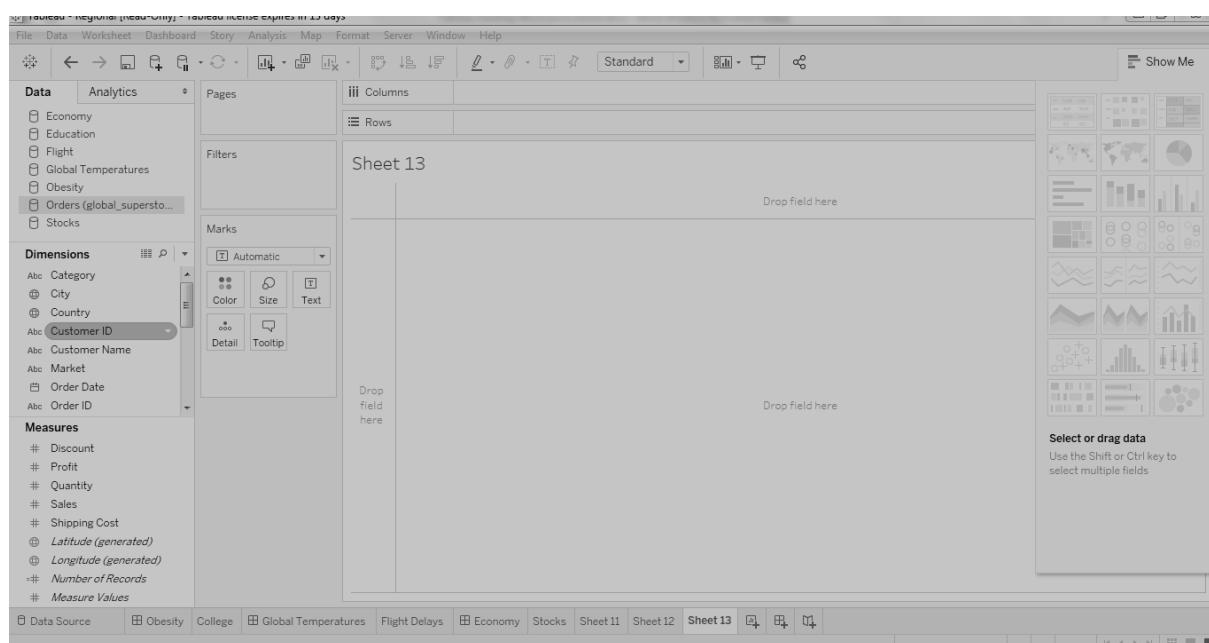
- Click on the Tableau icon displayed in the left-hand side of the Tableau worksheet page and expose the contents of the worksheet tab selected at the bottom of the screen.
- When you connect with a new data source, this is the default workspace view.



Go to the home page and select the global superstore sales-Excel sheet.

#	Abc Orders	Orders	Abc Orders	Abc Orders	Abc Orders	Abc Orders	Abc Orders
Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment
40098	CA-2014-AB1001514...	11-Nov-14	13-Nov-14	First Class	AB-100151402	Aaron Bergman	Consumer
26341	IN-2014-JR162107-41...	05-Feb-14	07-Feb-14	Second Class	JR-162107	Justin Ritter	Corporate
25330	IN-2014-CR127307-4...	17-Oct-14	18-Oct-14	First Class	CR-127307	Craig Reiter	Consumer
13524	ES-2014-KM1637548-...	28-Jan-14	30-Jan-14	First Class	KM-1637548	Katherine Murray	Home Office
47221	SG-2014-RH9495111-...	05-Nov-14	06-Nov-14	Same Day	RH-9495111	Rick Hansen	Consumer
22732	IN-2014-JM1565574...	28-Jun-14	01-Jul-14	Second Class	JM-156557	Jim Mitchum	Corporate
30570	IN-2012-TS2134092-4...	06-Nov-12	08-Nov-12	First Class	TS-2134092	Toby Swindell	Consumer
31192	IN-2013-MB1808592-...	14-Apr-13	18-Apr-13	Standard Class	MB-1808592	Mick Brown	Consumer

Open a connection to a saved data source, you also should have an open blank worksheet.

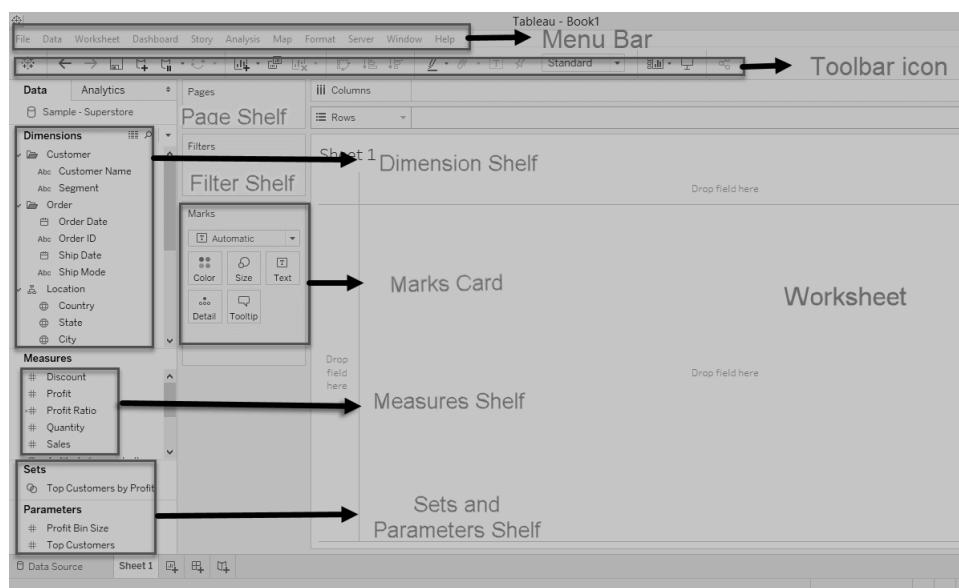


- In many ways, you can open a workspace page; for example, go to the display Tableau's icon on your desktop and you have a data source shown on your desktop.
- Dragging any data source icon and dropping it on the Tableau icon opens Tableau's worksheet page for the selected data source.
- Also, you can open as many connections as you need in Tableau by going to the data connection page or start page and select a new connection.

Now, the worksheet is connected to the global Superstore Sales-Excel dataset.

Tableau Desktop Workspace Menu

The Tableau desktop workspace consists of various elements as given below:



Menu Bar

- It consists of menu options like **File, Data, Worksheet, Dashboard, Story, Analysis, Map, Format, Server, Window, and Help**.
- The options in the menu bar, including features like data source connection, file saving, design, table calculation options, and file export features for creating a dashboard, worksheet, and storyboard.

File Menu

- For any Windows program the file menu contains New, Open, Close, Save, Save As, and Print, functions.
- The most frequently used feature found in this menu is the Print to pdf option.
- This allows us to export our dashboard or worksheet in pdf form.
- If you don't remember where Tableau places files, or you want to change the default file-save location, use the repository location option for review the file and change it.
- We can create a packaged workbook from the export packaged workbook option in a fast manner.

Data Menu

- You can use a data menu if you find some interesting tabular data on a website that you want to analyze with Tableau.
- Highlight and copy the data from the site, then use the Paste Data option to input it into Tableau.
- Once pasted, then Tableau will copy the data from the Windows clipboard and add a data source in the data window.
- The Edit Relationships menu option is used in data blending.
- This menu option is needed if the field names are not identical in two different data sources.
- It allows you to define the related fields correctly.

Worksheet Menu

- The Export option allows you to export the worksheet as an Excel crosstab, an image, or in Access database file format.
- The Duplicate as Crosstab option creates a crosstab version of the worksheet and places it in a new worksheet.

Dashboard Menu

- The Action Menu is a useful feature that is reachable from both the Worksheet Menu and the Dashboard Menu.

Analysis Menu

- In this menu, you can access the stack marks and aggregate measures options.
- These switches allow you to adjust default Tableau behaviours that are useful if you required to build non-standard chart types.
- The Create Edit Calculated Field and Calculated Field options are used to make measures and new dimensions that don't exist in your data source.

Map Menu

- The Map Menu bar is used to alter the base map color schemes.
- The other menu bar is related in the way of replacing Tableau's standard maps with other map sources.
- You can also import the geocoding for the custom locations using the geocoding menu.

Format Menu

- This menu is not used very commonly because pointing at anything, and right-clicking gets you to a context-specific formatting menu more quickly.
- You may need to alter the cell size in a worksheet rarely. If you don't like the default workbook theme, use the Workbook Theme menu to select one of the other two options.

Toolbar Icon

- Toolbar icon below the menu bar can be used to edit the workbook using different features like redo, undo, new data source, save, slideshow, and so on.

Dimension Shelf

- The dimension presents in the data source for example- customer (customer name, segment), order (order date, order id, ship date, and ship mode), and location (country, state, and city) these all type of data source can be viewed in the dimension shelf.

Measure Shelf

- The measures present in the data source, for example- Discount, Profit, Profit ratio, Quantity, and Sales- These all types of data source can be viewed in the measure shelf.

Sets and Parameters Shelf

- The user-defined sets and parameters can view in the sets and parameters. It is also used to edit the existing sets and parameters.

Page Shelf

Page shelf is used to view the visualization in video format by keeping the related filter on the page shelf.

Filter Shelf

Filter Shelf is used to filter the graphical view by the help of the measures and dimensions.

Marks Cards

Marks card is used to design the visualization. The data components of the visualization like size, color, path, shape, label, and tooltip are used in the visualizations. It can be modified in the marks card.

Worksheet

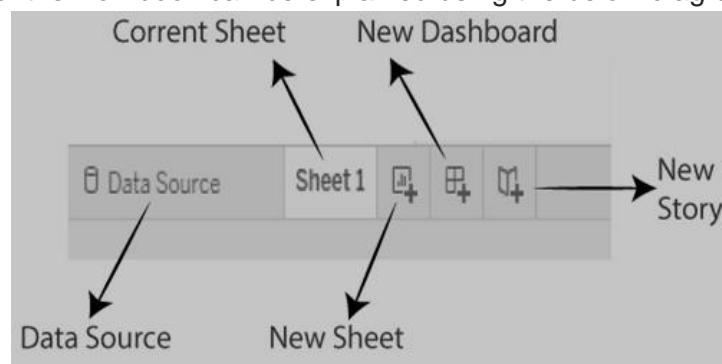
The worksheet is the space where the actual visualization, design, and functionalities are viewed in the workbook.

Tableau Repository

- Tableau repository is used to store all the files related to the Tableau desktop.
- It includes various folders like Connectors, Bookmarks, Data sources, Logs, Extensions, Map sources, Shapes, Services, Tab Online Sync Client, and Workbooks.
- My Tableau repository is located in the file path **C:\Users\User\Documents\My Tableau Repository**.

Tableau Navigation

Tableau Navigations of the workbook can be explained using the below diagram:



Data Source

We can modify existing data source, and create or add the new data source using the 'Data source' tab, which is present at the bottom of the Tableau desktop window.

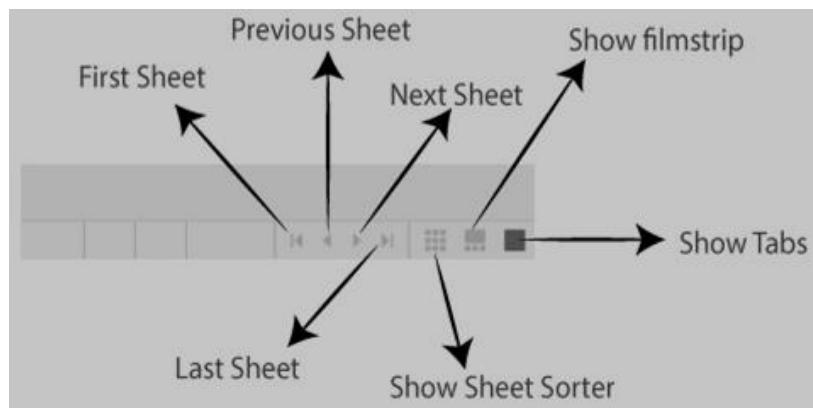
- **Current Sheet:** Current Sheet is a sheet of workbook in which we are currently working. All the dashboards, worksheets, and storyboard present in the workbook, are available in this tab.
- **New Sheet:** The **new sheet** icon presents in the tab is used to create a new worksheet in the Tableau workbook.

New Dashboard

The **new dashboard** icon presents in the tab is used to create a new dashboard in the Tableau workbook.

New Storyboard

The **new storyboard** icon presents in the tab is used to create a new storyboard in the Tableau Workbook.



First Sheet

This **first sheet** icon presents in the tab at the bottom of the right-hand side of Tableau desktop window is used for visiting the first sheet directly.

Previous Sheet

The **previous sheet** icon is used to return back to the last worksheet from the new sheet.

Next Sheet

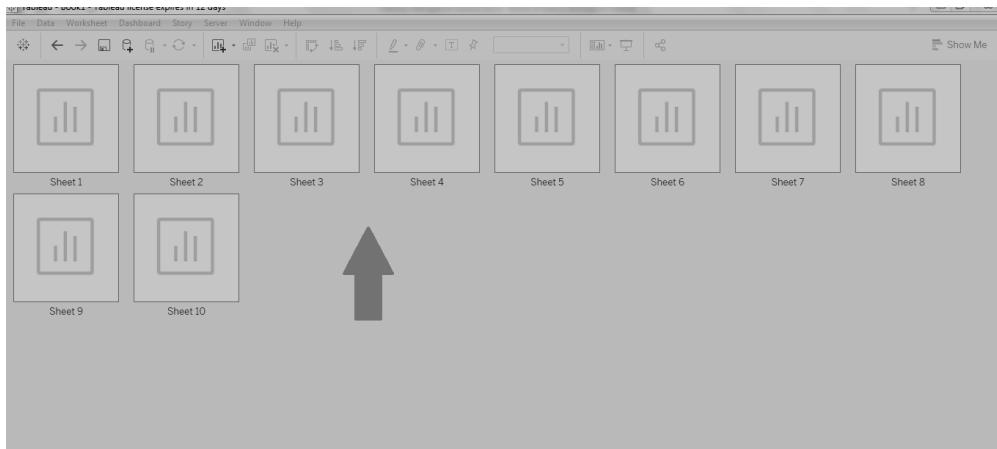
The **next sheet** icon is used to jump to the next worksheet of Tableau desktop.

Last Sheet

The **Last sheet** icon is used to visit the final sheet of tableau workbook.

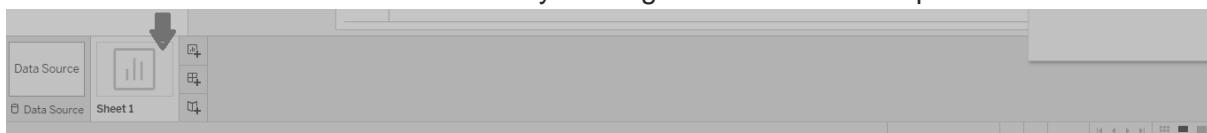
Show Sheet Sorter

You can view all the created worksheet in tableau desktop by clicking on the show sheet sorter icon.



Show Filmstrip

All the tabs are shown here with their icons by clicking on the show filmstrip.

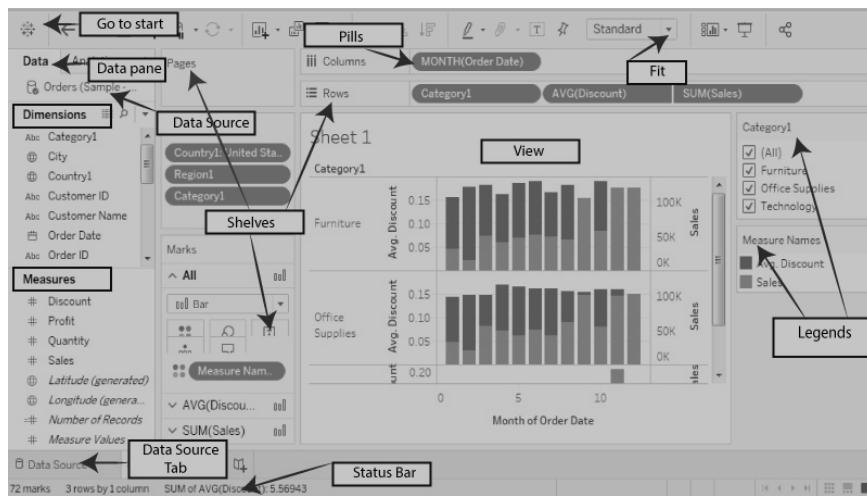


Show Tabs

This tab concludes all tabs such as worksheets, data sources, dashboards, and storyboard.

Tableau Data Terminology

Tableau is a powerful data visualization tool; that's why Tableau has many unique terminologies and definitions. You should know their meaning before you start using these features in Tableau.



The most commonly used Tableau terminologies are listed below:

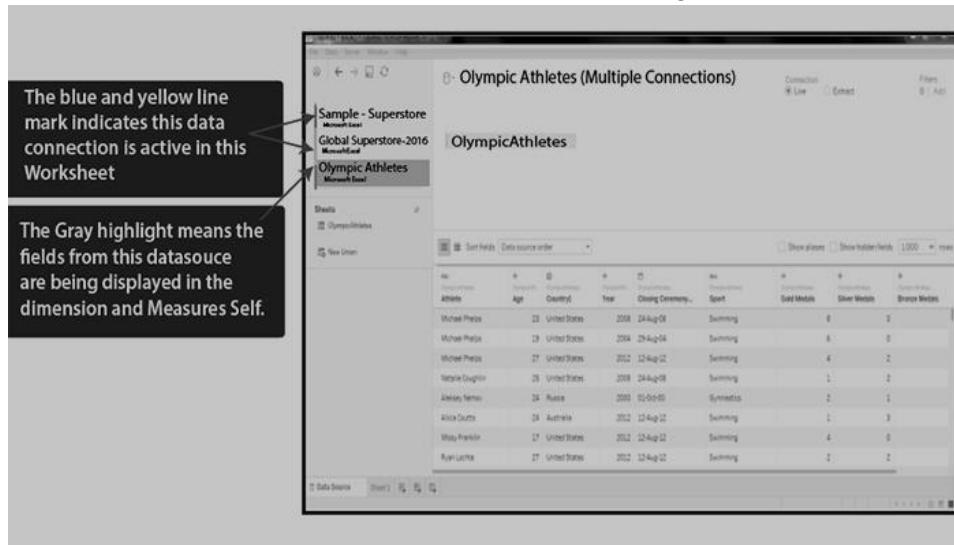
- Alias:** Alias is an alternative that you can assign to a dimension member, to a measurement part or a field.
- Bin:** Bin is a user-defined group of measures in the data source.
- Bookmark:** A .tmb document in the bookmarks folder in the Tableau repository that contains a single worksheet. It helps in improving data analysis. Unlike, web browser bookmarks, .tmb files are a compatible way to display various studies quickly.
- Calculated field:** Calculated field is a new field that the user creates derived files by using a formula to modify the existing fields in your data source. It is used to make your work simple and easy.
- Crosstab:** Crosstab is used for a text table view. It uses various text tables to display the numbers associated with dimension members.
- Dashboard:** The dashboard is a combination of several views that are arranged on a single page. In Tableau, dashboards are used to observe and compare a variety of data together, and also it allows interacting with other worksheets.
- Data Pane:** The data pane is on the left side of the workbook displays the fields of the data sources to which Tableau is connected. The fields are further divided into measures and dimensions. The data pane also reflects custom fields such as groups, binned fields, calculations, and many more. You can build views of your data by dragging fields from the data pane onto the various shelves, which is a part of every worksheet.
- Data Source Page:** Data Source is a page where you can set up your data source. This data source page generally consists of four main areas ? join area, left pane, a preview area, and metadata area.
- Dimension:** Dimension is commonly known as a field of categorical data. Dimensions hold discrete data such as members and hierarchies that cannot be aggregated. It also contains characteristic values such as dates, names, and geographical data. The dimensions used to reveal details of your information.
- Extract:** An extract is a saved subset of a data source which is used to improve performance and study offline. The users can create an extract by defining limits and filters that contain the data which you want in the extract.

11. **Filters Shelf:** Filter shelf is located on the left side of the workbook. Filters shelf is used to exclude the data from a view by filtering it using both dimensions and measures.
12. **Format Pane:** The Format pane is on the left side of the workbook, and it contains various formatting settings. It controls the entire view of the worksheet, as well as the individual fields in the view.
13. **Level of Detail expression (LOD):** The level of detail Expression is a syntax that supports the combination of various dimensions other than the view level. With the help of detail expressions, one can attach multiple dimensions with an aggregate expression.
14. **Marks:** Marks is a part of the view that visually represents one or more rows in a data source. It can be a line, square, or bar. You can control and alter the size, type, and color of marks.
15. **Marks Card:** Marks card is on the left side of the worksheet. The user can drag fields to the control mark properties such as color, type, shape, size, label, detail, and tooltip.
16. **Pages Shelf:** Page shelf is on the left side of the view. With the help of the page shelf, you can split a view into a sequence of pages based on the values and members in a continuous or discrete field. Adding a field with the pages shelf is similar to adding a field in rows shelf. For each new row, a new page is created.
17. **Rows shelf:** Row shelf is on the top of the workbook. It is used to create the rows of a data table. The Row shelf provides any numbers of measures and dimensions. When you placed a dimension on the Rows shelf, then Tableau creates headers for the members of that dimension. And when you place a measure on the Rows shelf, Tableau creates quantitative axes for that particular measure.
18. **Shelves:** The shelves are named areas that are located on the top and left of the view. You can build views by placing fields onto the shelves. Some shelves are only available when you select a particular mark type. For example, The Shape shelf is only open when you choose the specific Shape mark type.
19. **Workbook:** A workbook is a file with .twb extension that holds one or more worksheets as well as dashboards and stories.
20. **Worksheet:** The worksheet is a collection of sheets. It's a place where you build views of your data by dragging various fields onto the shelves.

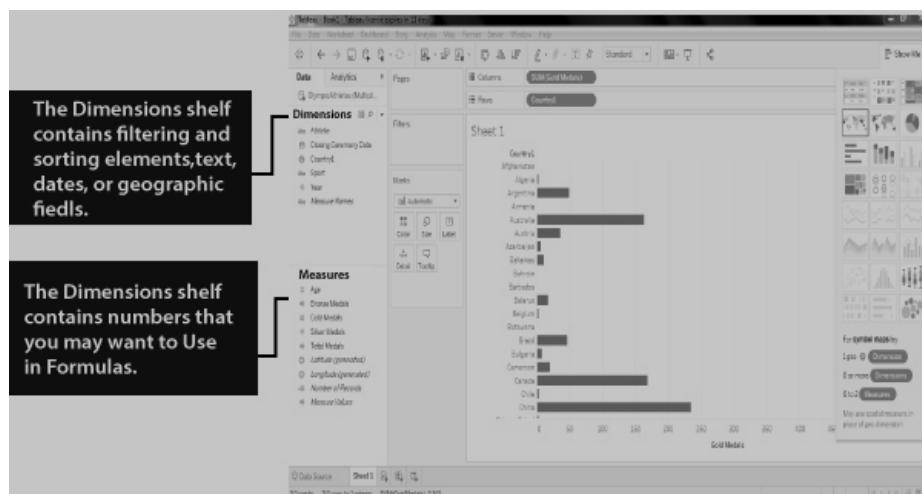
Data Window in Tableau

- Data window is a way to show the connection between Tableau and data source.
- You can connect to as multiple different data sources in a single workbook.
- The small icons associated with data connections provide additional details about the nature of the connection.

Here, a workbook that shows the three different data connection given below:



- The green line next to the global superstore data connection indicates that it is the active connection in the worksheet. So, the bar chart in the spreadsheet was created using '**dimensions and measures**' from that data source. Thus, the bar chart is created using the dimensions and measures from the data source.
- The **Olympic Athletes** data connection is a direct connection that is also indicated by the grey highlights. Those data source fields are currently displayed on the measures and dimensions shelves. The clipboard data source at the top of the data window was dragged and dropped into Tableau.
- When you create data connections, Tableau will automatically evaluate the fields and place them on the measures and dimensions shelves.



Usually, Tableau placed most of the fields correctly. If something is incorrectly placed, drag the field to the correct location. Errors sometimes occur when numbers are used to illustrate dimensions.

For example, if you want to connect a spreadsheet that contains Olympic Athletes details and you want to know how many gold medals were won by different countries in last years, that field is placed into the measures shelf. Dragging gold medal field from the measures shelf and dropped into the worksheet would result in the field being summed. Properly placed on the dimension shelf, the athletes country would behave like a dimension and be expressed in a column or row. In the same way, the gold medal and country are represented in the above Figure.

Data Types in Tableau

- Tableau expresses fields and assigns data types automatically.
- If the data source appoints the data type, Tableau will use that data type. If the data source doesn't individually assign a data type, Tableau will assign one.
- Tableau consist of the following data types:
 - Date values
 - Text values
 - Numerical values
 - Date and time values
 - Boolean values (True or False conditions)
 - Geographic values (longitude and latitude used for maps)



- In the above figure, focus on the icons next to the fields in the measures and dimension shelves. These icons denote specific data types.
- A calendar with a clock is a date or time field. Numeric values have pound signs, and "abc" icons indicate text fields. Boolean fields have "True or False" values.

4.6 Data Aggregation in Tableau

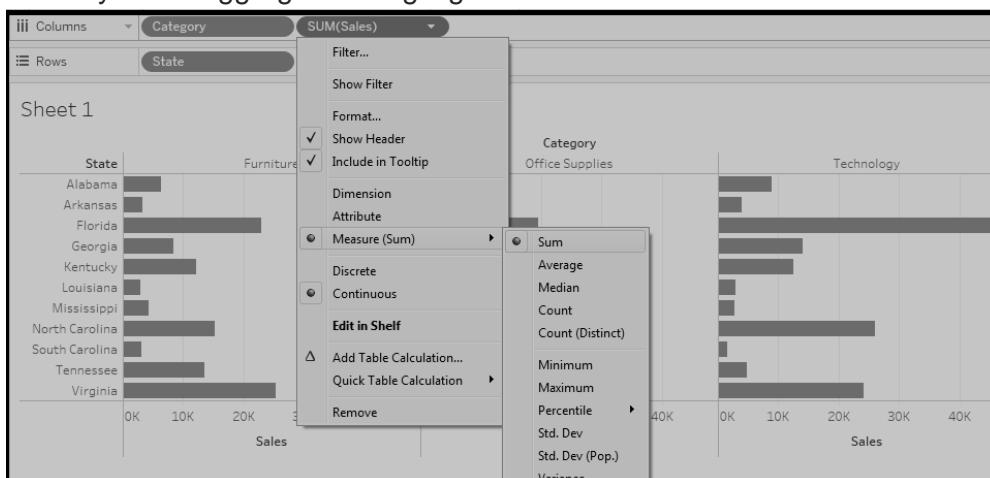
It is useful to look at numeric values using different aggregations function. Tableau supports many different aggregation types, such as:

- Sum
- Average
- Count
- Count Distinct
- Median
- Minimum
- Maximum
- Variance
- Variance of Population
- Standard Deviation
- Standard Deviation of Population
- Attribute
- Dimension

- In Tableau, you can create aggregation dimensions and measures. Whenever you add measures to your view, an aggregation is applied to those measures by default.
- The type of Aggregation used depends on the context of the view.
- If you are not familiar with the database, then refer to Tableau manual for detailed definition of these aggregate types. You are adding fields into the visualization by default then it will be displayed.
- Tableau allows you to change or alter the aggregation level for a specific view.
- To change the default aggregation, do right click on that field inside the data shelf and change its default by selecting the menu options (default properties or Aggregation).

You can also change the Aggregation of a field for specific use in a worksheet.

For example: By right-clicking on the SUM (Sales) pill and selecting the Measure (SUM) menu option, you can choose any of the aggregations highlighted.



- The data source used in the above figure is a data extract of an Excel spreadsheet.
- It is important to understand that if you depend on a direct connection to Excel, the median and count (distinct) aggregations would not be available.
- Access, Excel, and text files do not support these aggregate types. Tableau's extract engine do this task.

Aggregating Measures

- When you add a measure to the view, Tableau automatically aggregates its value. Average, sum and median are the common aggregation functions.
- The current Aggregation looks like part of the measure's name in the view.

For example: Sales becomes **SUM (Sales)**, and every measure has a default aggregation, which is set by Tableau when you connect to a data source. You can change or view the default aggregation for measures.

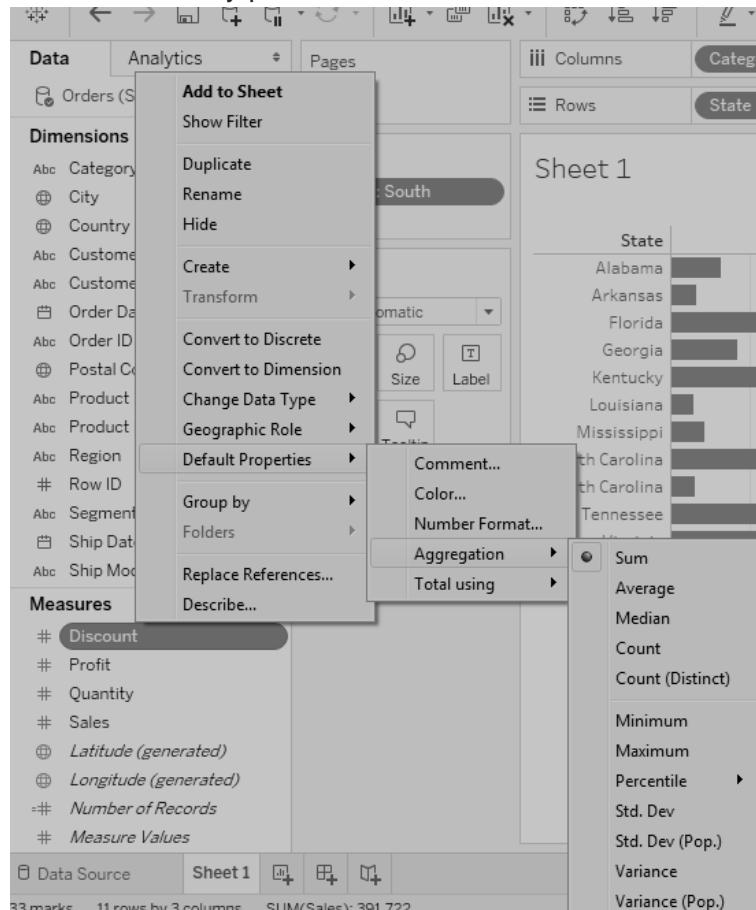
- You can aggregate a measure using Tableau only for relational data sources.
- Multidimensional data sources contain data sources which are already aggregated.
- In Tableau, the multidimensional data source is supported only in windows.

Set the default Aggregation for Measures

- You can set the default aggregations for any measures.
- It is not a calculated field that itself contains an aggregate, such as AVG ([Discount]).
- A default aggregation is the preferred calculation for summarizing a discrete or continuous field.
- The default aggregation is used when you drag a measure to a view automatically.

To change the default Aggregation

- Right-click on a measure menu option in the Data field and select **Default Properties** then select **Aggregation**, and then select one of the aggregation options.
- You cannot set default aggregation for the published data source. The default aggregation is set only when the data source is initially published.

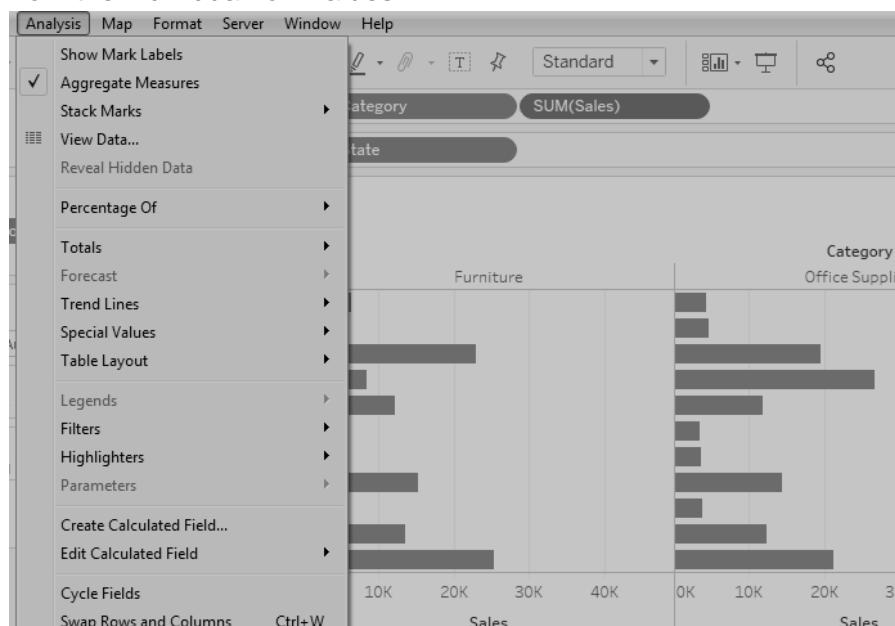


How to Disaggregate the Data

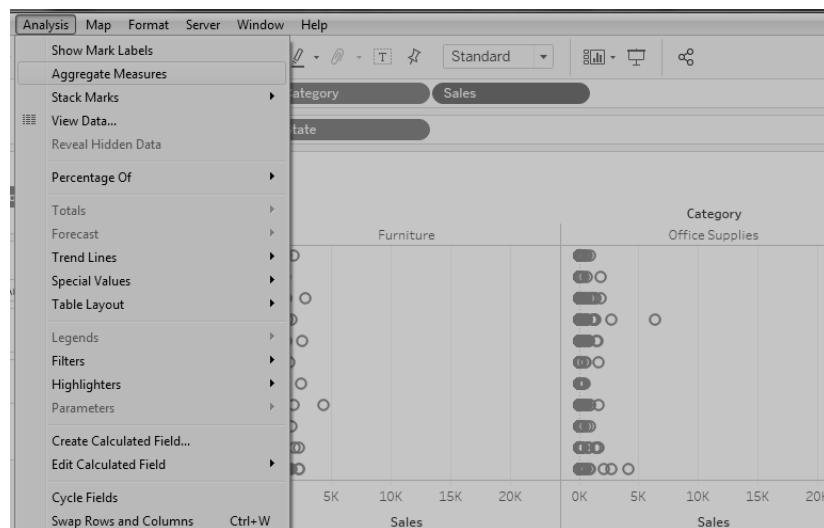
- When you add a measure to your view, then Aggregation is applied to that measure automatically. This default is controlled by the **Aggregate Measures** setting in the **Analysis** menu.
- If you want to see all of the marks in the view at the most detailed level of the model, you can disaggregate the view.
- Disaggregating your data means that the Tableau will display a separate mark for every data value in every row of your data source.

Disaggregation in all Measures in the view

- Click on the **analysis** then go to **aggregation measures** option.
- When **Aggregate Measures** is selected, then automatically Tableau will attempt to aggregate measures in the view.
- Means that it collects individual row values from your data source into a single value that is adjusted to the level of detail in your view.
- The different aggregations available for measures determine how the individual values are collected: they can be averaged (AVG), added (SUM), or set to the minimum (MIN) or maximum (MAX) value from the individual row values.



If it is already selected, click aggregation measures once for deselecting it. Then, you can see the changes.

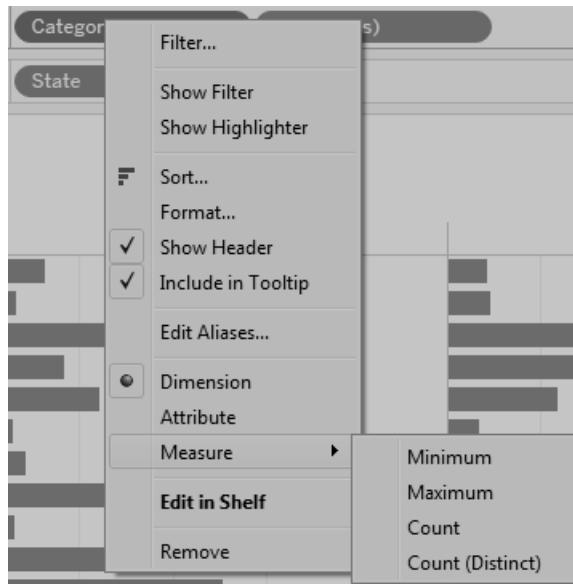


- Disaggregating data can be useful for analyzing measures which you want to use both dependently and independently in the view.

Note: If your data source is very large, then, as a result, disaggregating the data can degrade in significant performance.

Aggregating Dimensions

- You can aggregate dimension in the view as **Maximum**, **Minimum**, **Count**, and **Count Distinct**.
- When you aggregate a dimension, you have to create a new temporary measure column, so the dimension takes on the characteristics of a measure.



Note: The **Count Distinct** aggregation does not support the Text File and Microsoft Excel data sources using the inheritance connection. If you are connected to one of these types of data sources, then the **Count Distinct** aggregation is unavailable, and it shows the remark "Requires extract." If you save the data sources as an extract, you will be able to use the **Count Distinct** aggregation. Another way to view a dimension as an attribute. You can change it by choosing the Attribute from the context menu for the dimension.

The attribute aggregation has several uses:

- It ensures a consistent level of detail when blending multiple data sources.
- It provides a way to aggregate the dimension when computing table calculations, which require an aggregate expression.
- It improves query performance due to locally computed.

Tableau calculates the Attribute using the below given formula:

```
If MIN (dimension) = MAX (dimension) then MIN (dimension) else "*" end
```

- This given formula is calculated in Tableau after the data is retrieved from the initial query.
- The asterisk (*) is a visual indicator of a special type of Null value it occurs when there are multiple values.

Below is an example of using Attribute in a table calculation. This table shows the market, market size, state, and sales by the market that is SUM (sales). Suppose, you want to compute the percent of the total sales according to each state contribution for the market. When you add some Percent of Total in table calculation that calculates along State, the calculation computes within the black area shown above figure just because the Market Size of dimension is partitioning the data.

When you aggregate the Market Size as an Attribute, the calculation is computed within the Market (East), and the Market Size information is used as a label in the display.

iii Columns		Measure Names		
Rows		Market	Market Size	State
				SUM(Sales)
Sheet 1				
Market Central	Major Market	Colorado	48,179	No Measure..
		Illinois	69,883	Abc
		Ohio	34,517	Abc
	Small Market	Iowa	54,750	Abc
		Missouri	24,647	Abc
		Wisconsin	33,069	Abc
East	Major Market	Florida	37,443	Abc
		Massachusetts	29,965	Abc
		New York	70,852	Abc
	Small Market	Connecticut	25,429	Abc
		New Hampshire	14,887	Abc
		Texas	37,410	Abc
South	Major Market	Louisiana	23,161	Abc
		New Mexico	15,892	Abc
		Oklahoma	27,463	Abc

iii Columns		Measure Names		
Rows		Market	Market Size	State
				SUM(Sales)
Sheet 1				
Market Central	Major Market	Colorado	48,179	No Measure..
		Illinois	69,883	Abc
		Ohio	34,517	Abc
	Small Market	Iowa	54,750	Abc
		Missouri	24,647	Abc
		Wisconsin	33,069	Abc
East	Major Market	Florida	37,443	Abc
		Massachusetts	29,965	Abc
		New York	70,852	Abc
	Small Market	Connecticut	25,429	Abc
		New Hampshire	14,887	Abc
		Texas	37,410	Abc
South	Major Market	Louisiana	23,161	Abc
		New Mexico	15,892	Abc
		Oklahoma	27,463	Abc

4.7 Tableau File Types

Tableau File Types

- Tableau's output after data analysis can be saved into different formats, which further can be distributed into different platforms.
- There are various forms of different file categories, and the multiple different extensions identify them.
- Their extension depends on how it produces and for what purposes they are used in which format.
- These all are generally stored as **xml** file format, and it can be easily open and edited.
- You can save your work using several different Tableau specific file types such as bookmarks, workbooks, data extracts, packaged data files, and data connection files.

Each of these files is described below in detail:

Type	File Extension	Purpose
Tableau workbook	(.twb)	Tableau workbook can hold one or more worksheets, and also hold zero or more stories and dashboards.
Tableau Bookmarks	(.tmb)	Tableau bookmarks can hold a single worksheet that can be easily shared, and pasted into other workbooks.
Tableau Packaged workbook	(.twbx)	Tableau packaged workbook is a single zip file which contains a workbook along with any supporting local file data and background images. This is the best way to package your work for sharing with others who don't have access to the original data.
Tableau Data Extract	(.hyper or .tde)	Tableau data extract is a local copy of the entire data set. It is used to share the data with others when you worked offline, and want to improve the performance.
Tableau Data Source	(.tds)	Tableau data source file is a shortcut for quickly connecting to the original data that you use regularly. Data source file does not contain the actual data, and they only contain the necessary information to connect with the actual data. You can modify the top of the actual data such as creating calculated fields, changing default properties, adding groups, and so on.
Tableau Packaged Data Source	(.tdsx)	Tableau packaged data source is very similar to the tableau data source, but it has an addition of data along with the connection details.
Tableau Preferences	(.tps)	This file stores the color preferences, which is used among all the datasheets. It is also used to generate a customized look for the users.

- These files are saved in the associated folders in the **My Tableau Repository** directory, which is created in your **My Documents** folder by default when you install Tableau.
- Also, your work files can be saved in other locations, such as a network directory or your desktop.

How to Change the Tableau Repository Location

- You can be specified a new location for the Tableau repository if you are not using the default location in your Document folders.

For example: If you want to have your data on a network server instead of your local machine, then you can see the remote repository.

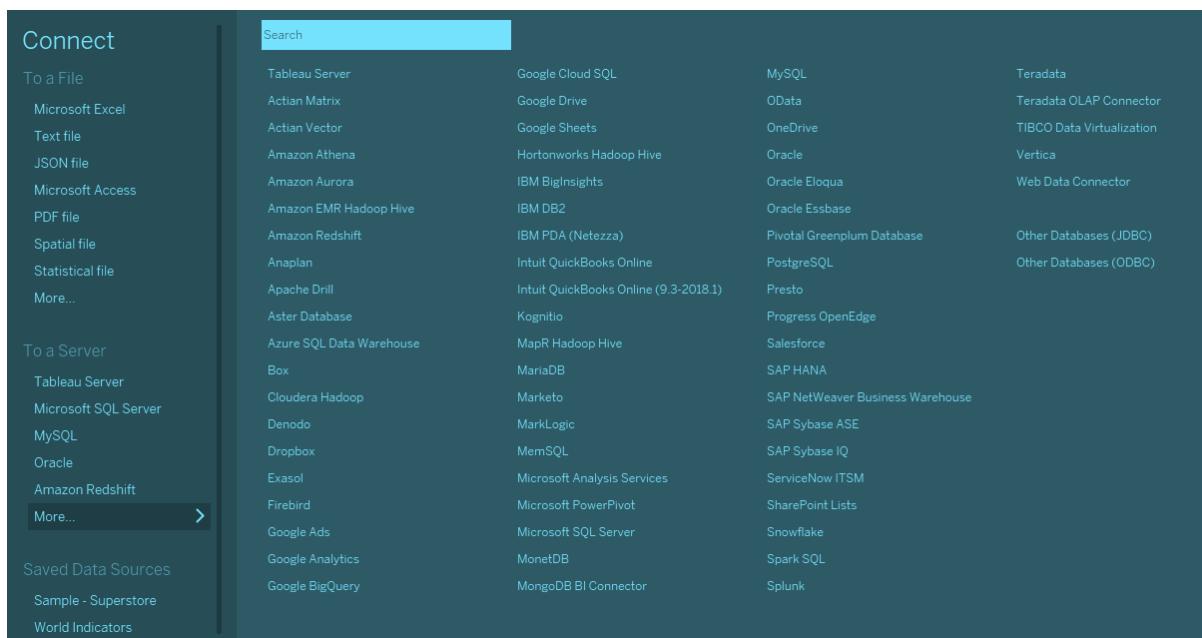
- Select File then go to Repository Location.
- Select a new folder that will be the new repository location in the select a repository dialog box.
- Restart Tableau then it uses the new repository.

Changing the repository location does not include the original repository. Alternatively, Tableau creates a new repository where you can store your files.

4.8 Data Connection with Data Sources

- Tableau can connect with all the accessible data sources which are broadly used.
- It can link to Excel files, PDF files, text files, etc. It can also connect to various databases using its ODBC connector. Tableau can connect to web connectors and servers.
- Tableau's native connectors can connect to the following types of data sources:
 - File Systems:** Such as Microsoft Excel, CSV, etc.
 - Cloud Systems:** Such as Google bigQuery, Windows Azure, etc.
 - Relational System:** Such as Microsoft SQL Server, Oracle, DB2, etc.
 - Other Sources:** It uses ODBC.

The given below picture shows all of the data sources available through Tableau's native data connectors.



Connect Live

- The Connect Live feature is used in real-time data analysis. In connect live case, Tableau connects with the real-time data source, and it keeps read the data.
- Thus, the result of the data analysis is up to the second, and the latest changes are reflected in this result. However, on the drawback, it's the source system as it has to keep send the data to Tableau.

In-Memory

- Tableau can also process the data in-memory by caching them in memory, and it not being connected to the source anymore while analyzing the data.
- Of course, there will be a limit on the amount of data cached depending on the availability of the memory.

Combine Data Sources

- Tableau can connect with different data sources at the same time.

For example: In a single workbook, you can connect to a relational source and a flat file by defined the multiple connections. This is also used in data blending, which is a unique feature in Tableau.

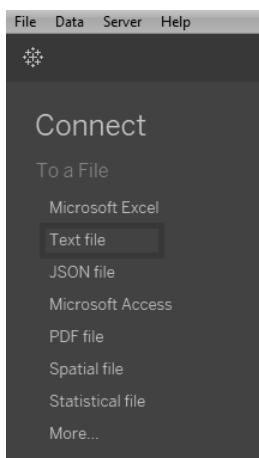
Data Connection with Text File

- Tableau can connect to the text file data and set up the data sources. Tableau connects to following text files (*.csv, *.txt, *.tsv, *.tab).

How to Make the Connection and Set up the Data Sources

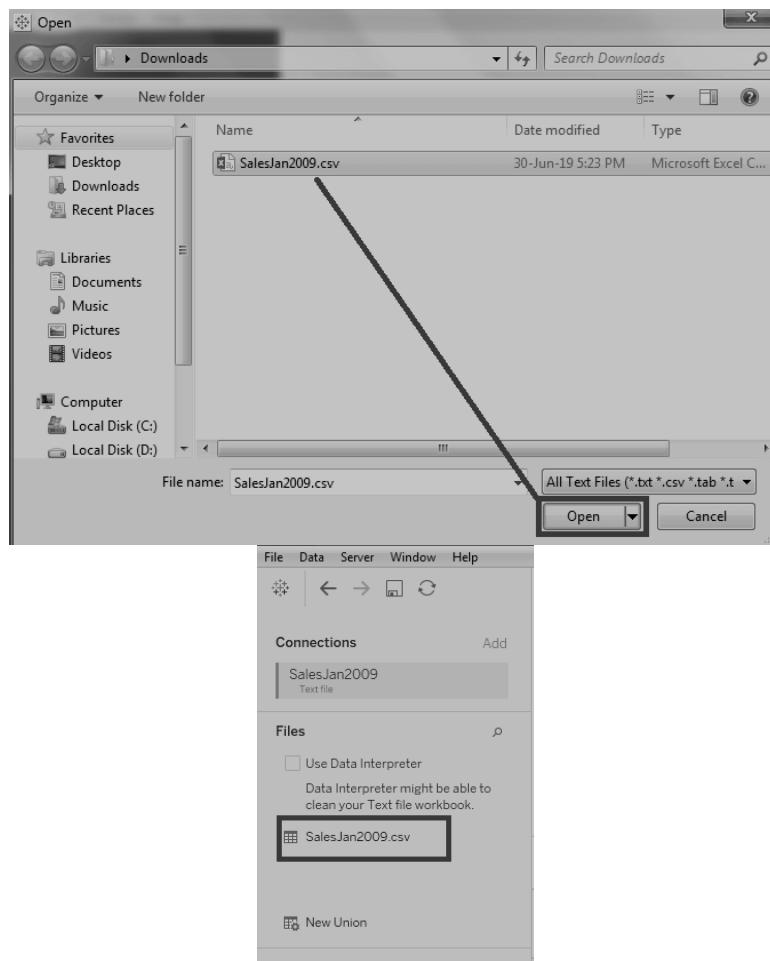
Step 1: Open Tableau.

Step 2: Below Connect, click on **Text File**.



Step 3: Go to the next screen,

- Select the file you want to connect such as SalesJan2009.CSV
- Click on Open option.
- On the left-hand side of the data source, you will see the **CSV file**.



Example:

Here is an example which shows the data connection with the text file.

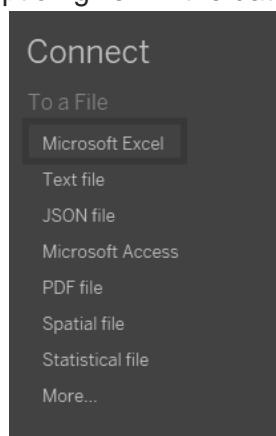
The screenshot shows the Tableau Data Source interface. In the top left, there's a 'Connections' section with a single entry: 'SalesJan2009' (Text file). Below it is a 'Files' section containing 'SalesJan2009.csv'. A note says 'Use Data Interpreter' with a sub-note: 'Data Interpreter might be able to clean your Text file workbook.' On the right, there are tabs for 'Connection' (set to 'Live') and 'Extract', and a 'Filters' section with '0 | Add'. The main area displays the contents of 'SalesJan2009.csv' with columns: Transaction date, Product, Price, Payment Type, Name, City, and State. The data shows several transactions from January 2009, such as '02-Jan-09 6:17:00 AM' for 'Product1' at '\$1,200' via 'Mastercard' to 'carolina' in 'Basildon, Eng'. Another transaction on '03-Jan-09 2:44:00 PM' for 'Product2' at '\$3,600' via 'Visa' to 'Gouda' in 'Cahaba Heights, AL'.

And the worksheet looks like

The screenshot shows the Tableau Worksheet interface. On the left, the 'Data' pane lists dimensions like 'Account Created', 'Country, State, City', 'Last Login', and measures like 'Price', 'Latitude', 'Longitude', and 'Number of Records'. The 'Analytics' tab is selected. The main workspace is titled 'Sheet 1' and contains three empty drop zones labeled 'Drop field here'. To the right, there's a large library of visualization icons, and at the bottom right, a note says 'Select or drag data' and 'Use the Shift or Ctrl key to select multiple fields'.

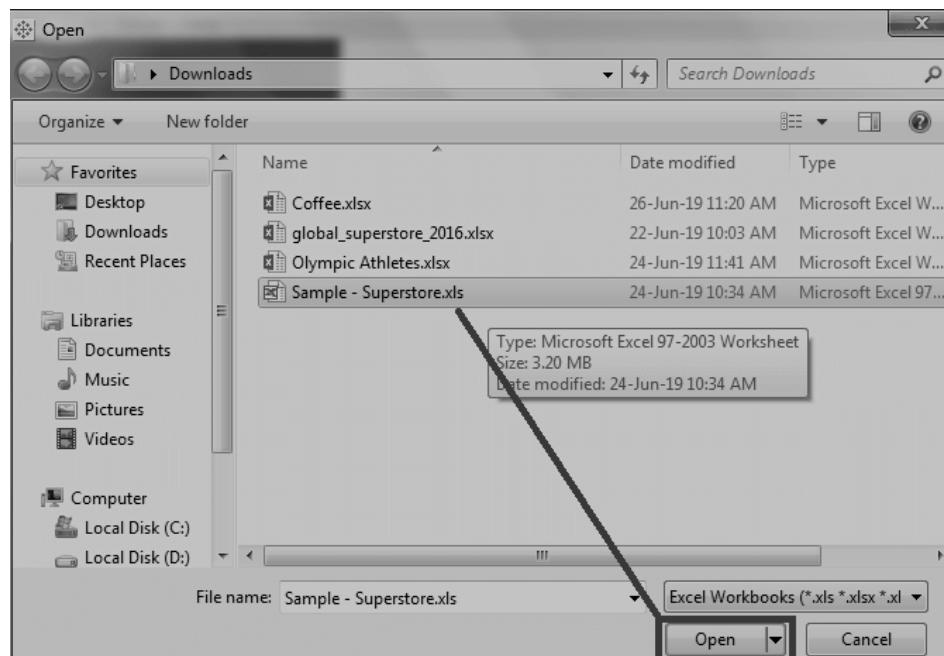
Data Connection with Microsoft Excel

Step 1: Click on the **Microsoft Excel** option given in the data tab.

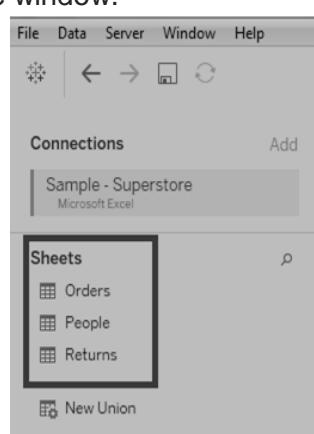


Step 2: In the next screen,

1. Select the Microsoft Excel file you want to connect such as **sample-superstore.xls**.
2. Click on **open** option.



Step 3: It connects the Microsoft Excel file to Tableau. The sheets present in the Microsoft Excel file are shown on the left-hand side of the window.



Step 4: You can drag one or more sheets from the sheets data tab such as Orders.

Then the data source looks like the below image:

And the worksheet looks like:

Tableau Extracting Data

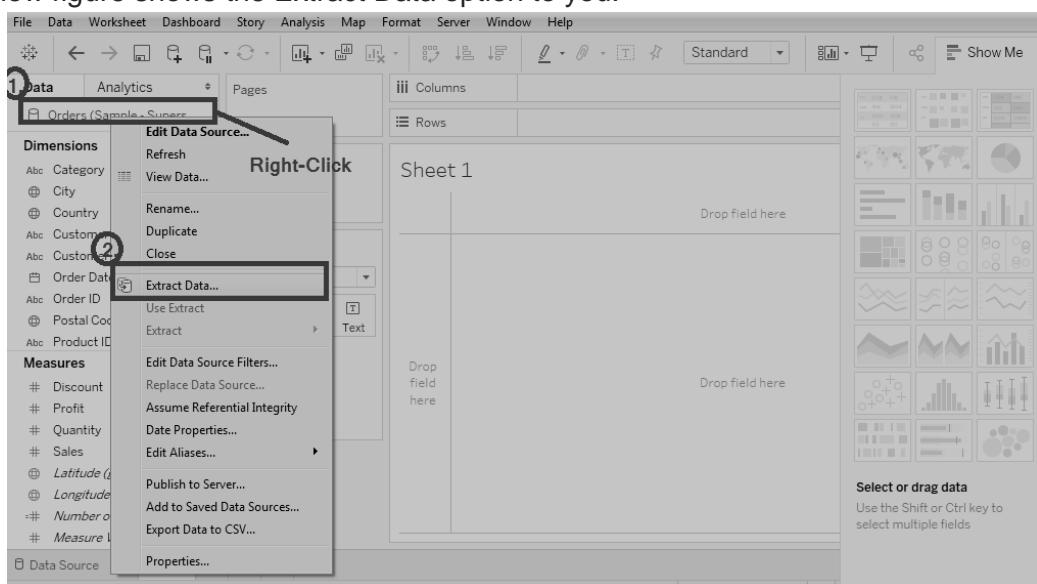
- In Tableau, Data extraction creates a subset of data from the data source.
- Data extraction is useful for increasing the performance by applying filters.
- It also helps in using some features of Tableau. Probably, which is not available in the data source like finding the distinct values in the data.
- However, the data extract feature is the most commonly used to create a local drive for offline access by Tableau.

Creating an Extract

- Extraction of the data is done by following the menu:

Data → Extract Data

- It creates multiple options such as applying limits to how many rows to extract and whether to aggregate data for dimensions.
- The below figure shows the Extract Data option to you.

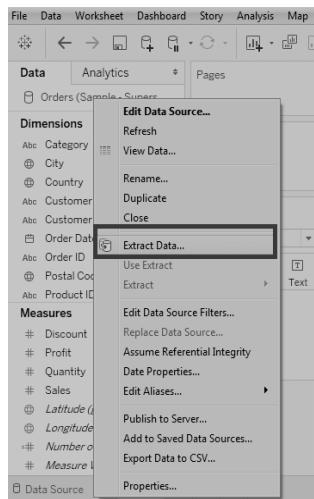


Applying Extract Filters

For extract a subset of data from the data source, you can create, filters which only return the relevant rows.

For example: The **Sample Superstore** data set.

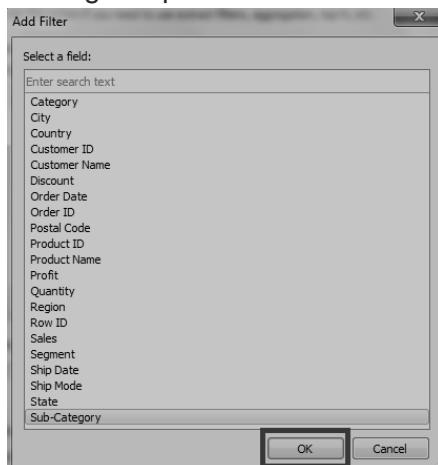
- Click on an **extract data**



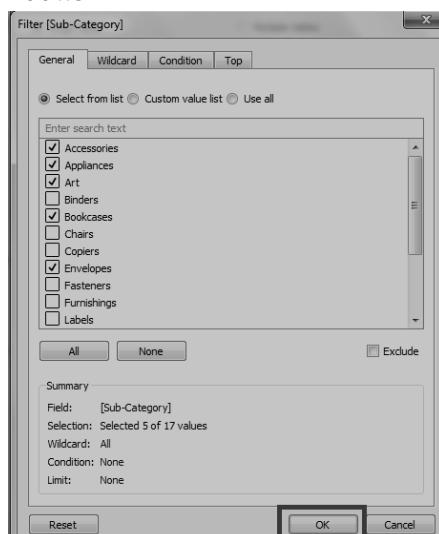
- Click on the **Add** button.



- Add any **filter** or select a **field** among all options such as **sub-category** and click **OK** button.

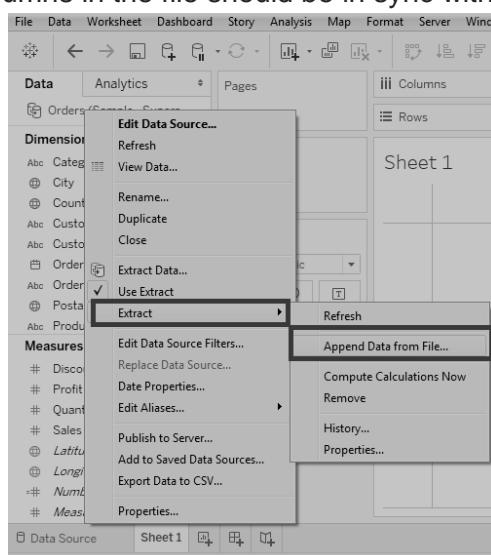


- Choose from the list and **tick mark** the checkbox value for which you need to pull the data from the source and click on the **OK** button.



Adding New Data to Extract

- Add more data for an already created extract, and you have to choose the **option Data → Extract → Append Data from File**.
- In this case, browse the file containing the data and click on the **OK** button to finish. Of course, the number and data type of columns in the file should be in sync with the existing data.

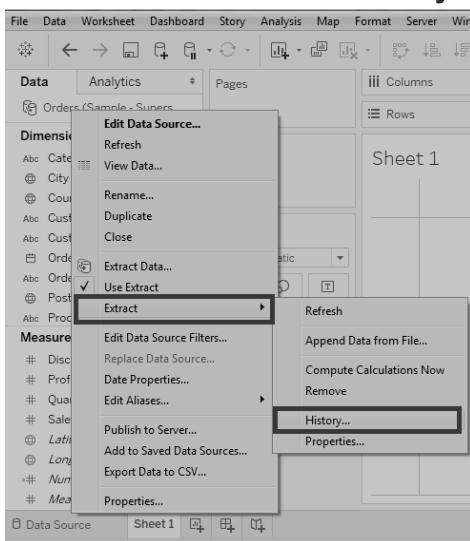


Extract History

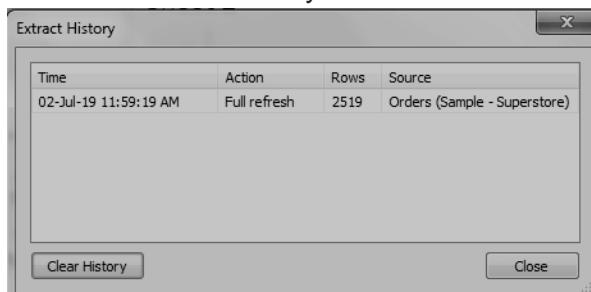
You can also verify the history of data extraction to know about how many times the data extraction has happened and at what times.

For this, you have to use the

menu - Data → Extract History



And then it shows you all the data extraction history.



4.9 Tableau Editing Metadata

- After connecting with the data source, Tableau captures the metadata details of the source, such as the columns and their data types.
- This is used to create the measures, dimensions, and calculated fields used in **views**.
- You can browse the metadata and change their properties for some specific requirements.

Checking the Metadata

After connecting with a data source all possible tables and columns will be displayed in the data source.

Example: The source '**Sample Coffee Chain**' for checking the metadata.

- Click the Data menu and select to connect with a data source. Browse for the **MS access** file named as '**Sample Coffee chain.**'
- Drag the table which is named **Product**, to the data canvas.
- After choosing the file, you will get the below-given screen that shows the column names, and their data types. In Tableau, the string data types are shown as "**Abc,**" and Numeric data types are shown as "**#.**"

Abc	Abc	#	Abc
Product	Product	Product	Product
Product Type	Product	Product Id	Type
Coffee	Amaretto	1	Regular
Coffee	Columbian	2	Regular
Coffee	Decaf Irish Cream	3	Decaf
Espresso	Caffe Latte	4	Regular
Espresso	Caffe Mocha	5	Regular
Espresso	Decaf Espresso	6	Decaf

Changing the Data Type

- You can change the data type for some of the fields (if required).
- Depending on the nature of the source data, sometimes Tableau may fail in recognizing the data type from the data source.
- In this structure, you can manually edit the data type. The below screenshot shows the options.

Abc	Abc	#	Abc
Product	Product	Product	Product
Product Type	Product	Product Id	Type
Coffee	Amaretto	1	Regular
Coffee	Columbian	2	Regular
Coffee	Decaf Irish Cream	3	Decaf
Espresso	Caffe Latte	4	Regular
Espresso	Caffe Mocha	5	Regular
Espresso	Decaf Espresso	6	Decaf

Renaming and Hiding

- You can change the column names by using the renaming option.
- You can also hide a column, after that it will not appear in the data view.
- All these options are available after clicking on the **data type** icon in the **metadata grid**, you can see in the below screenshot.

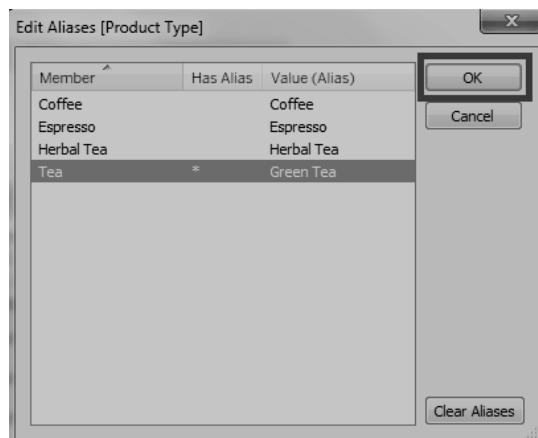
The screenshot shows the STL Metadata Grid application. In the center, there is a data grid titled "Product (Sample ...)" with columns "Product Type" and "Product". A context menu is open over the first row of the "Product Type" column, with "Rename" and "Hide" highlighted. Other options in the menu include "Copy Values", "Aliases...", "Create Calculated Field...", "Create Group...", and "Describe...". The application interface includes a toolbar at the top with File, Data, Server, Window, and Help menus, and a sidebar on the left showing connections and tables.

Column Alias

Each column of the data source is assigned as aliases, which helps in better understanding the nature of the column.

This screenshot is similar to the previous one, showing the STL Metadata Grid. The context menu over the "Product Type" column header now has "Aliases..." selected. The other options in the menu are "Rename", "Copy Values", "Hide", "Create Calculated Field...", "Create Group...", and "Describe...". The data grid and application interface are identical to the first screenshot.

- Choose the aliases option from the above figure, and a screen comes up, which is used to **Edit** or **Create** the aliases.



- Click on the **OK** button, and after that, you can see the changes in the column of the data sources.

Abc Product	Abc Product	# Product	Abc Product Type
Product Type	Product	Product Id	Type
Herbal Tea	Chamomile	8	Decaf
Herbal Tea	Lemon	9	Decaf
Herbal Tea	Mint	10	Decaf
Green Tea	Darjeeling	11	Regular
Green Tea	Earl Grey	12	Regular
Green Tea	Green Tea	13	Regular

Tableau Data Joining

- Data joining is a common requirement in any data analysis.
- You may need to join data from different tables in a single source or join data from multiple sources.
- Tableau provides the feature to join the tables by using the data pane that is available in the Data menu.
- A join means combining columns from one or more tables in a relational database.
- It also creates a set that can be saved as a table, or it can be used as it is.

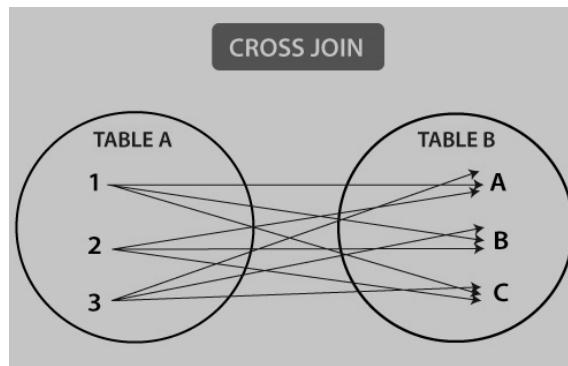
Joins are specifying into five types:

1. Cross Join
2. Inner Join
3. Natural Join
4. Outer Join
 - Left Outer Join
 - Right Outer Join
 - Full Outer Join
5. Self-Join

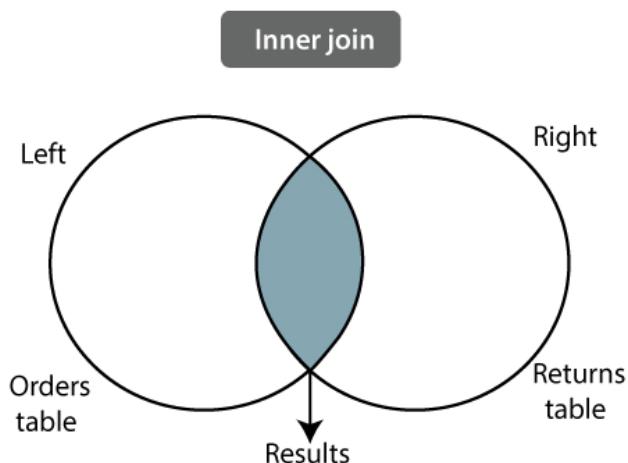
Types of Joins

A join section is used to combine rows from two or more tables, based on a related column between them.

1. Cross Join: Cross join produces rows which combine each row from the first table with each row from the second table.



2. Inner Join: An inner join returns the matching rows from the tables that are being joined.



3. Natural Join:

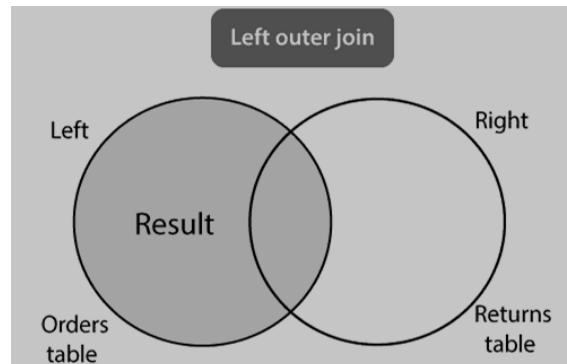
- Natural join is not used any comparison operator. It does not concatenate the way.
- Only we can perform a Natural Join if there is at least one common attribute that exists between two relations. Also, the attributes must have the same **name** and **domain**.
- Natural join works on those matching attributes where the values of attributes in both the relation are same.

4. Outer Join:

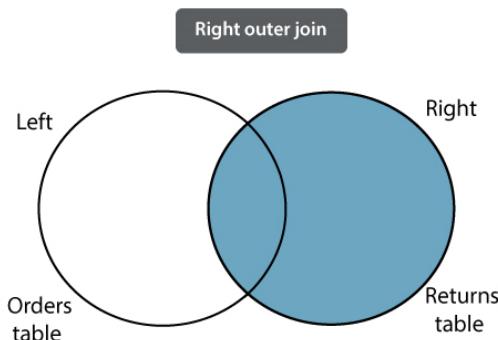
- An outer join is an extended form of the inner join.
- It returns both matching and non-matching rows for the tables that are being joined.

Types of outer joins are as follows:

i) **Left Outer Join:** The left outer join returns matching rows from the tables being joined, and also non-matching rows from the left table in the result and places **NULL** values in the attributes that come from the right table.

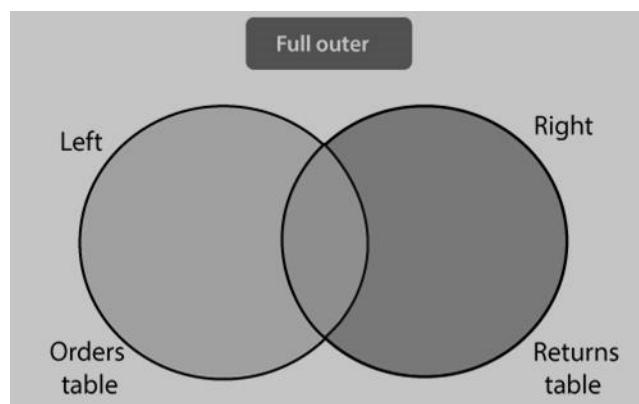


ii. **Right Outer Join:** The right outer join operation returns matching rows from the tables being joined, and also non-matching rows from the right table in the result and places **NULL** values in the attributes that come from the left table.



iii. Full Outer Join:

- The full outer join is used to combine tables. As a result, it contains all values from both tables.
- When a value from a table doesn't have a match with the other table, then it returns a **NULL** value in the data grid.

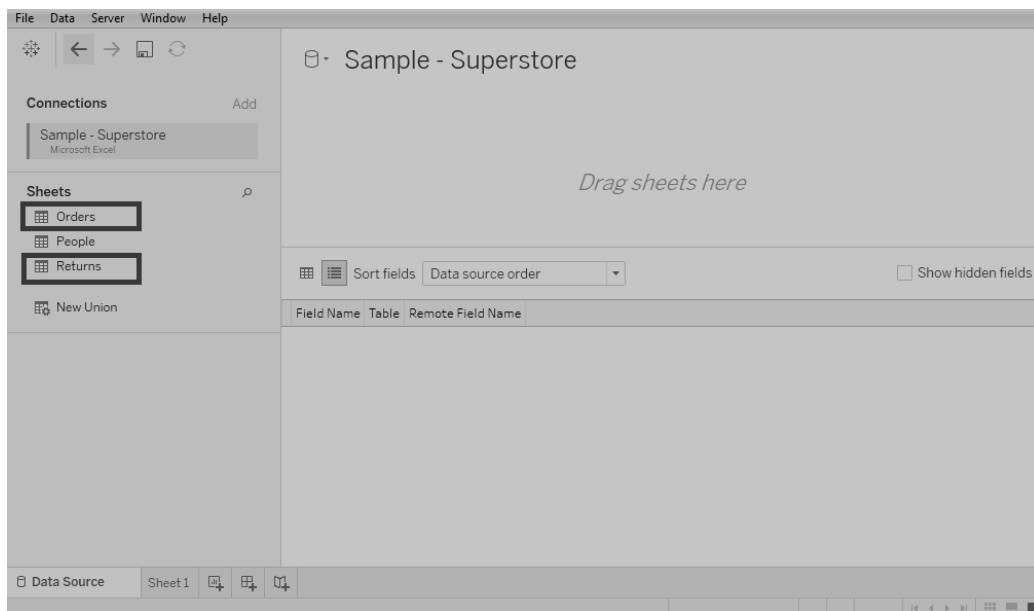


5. Self-Join: The self-join is used to join a table with itself. It means that each row of the table is combined with itself as well as with every other row of the table.

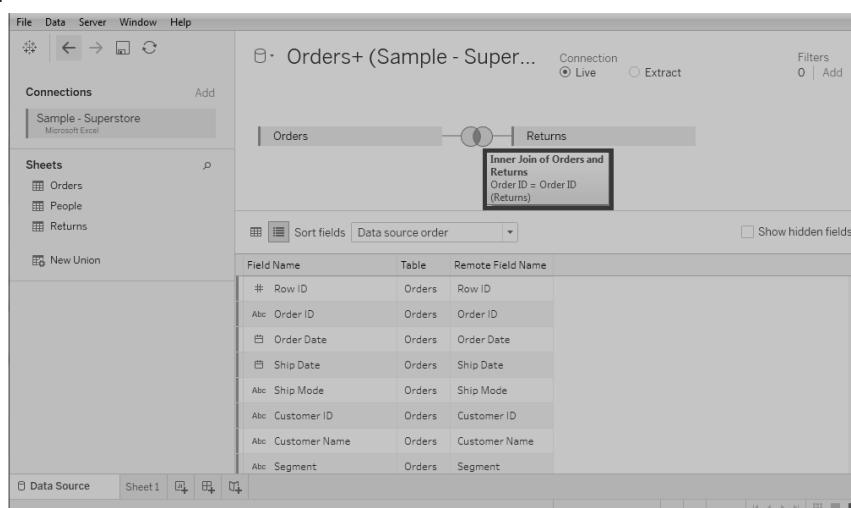
Creating a Join in Tableau

Let's assume a data source **Sample-superstore** to create a join between two tables such as **Orders** and **Returns**.

- Go to the Data menu and choose Microsoft Excel option below **connect**.
- Then select **sample-superstore** as a data source and click the **Open** button.
- Drag **Orders** and **Returns** tables from **sheets** of the data source to the **data pane**. After that Tableau will automatically create a join between **Orders** and **Returns** tables which can be changed later as per required joins.



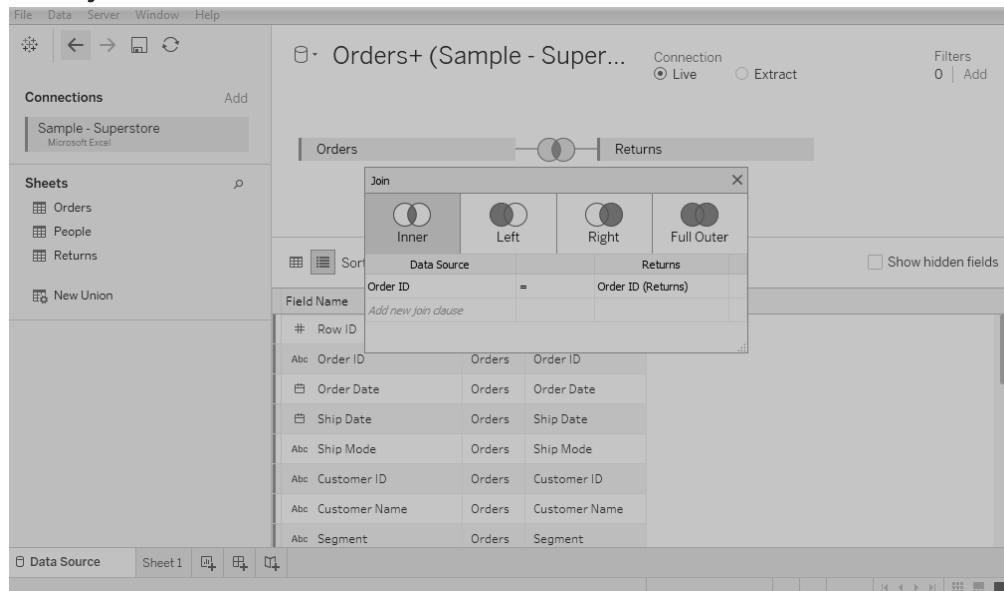
- Below screenshot shows the building **inner join** between Orders and Returns tables by using the **Order id** field.



Edit a Join Type in Tableau

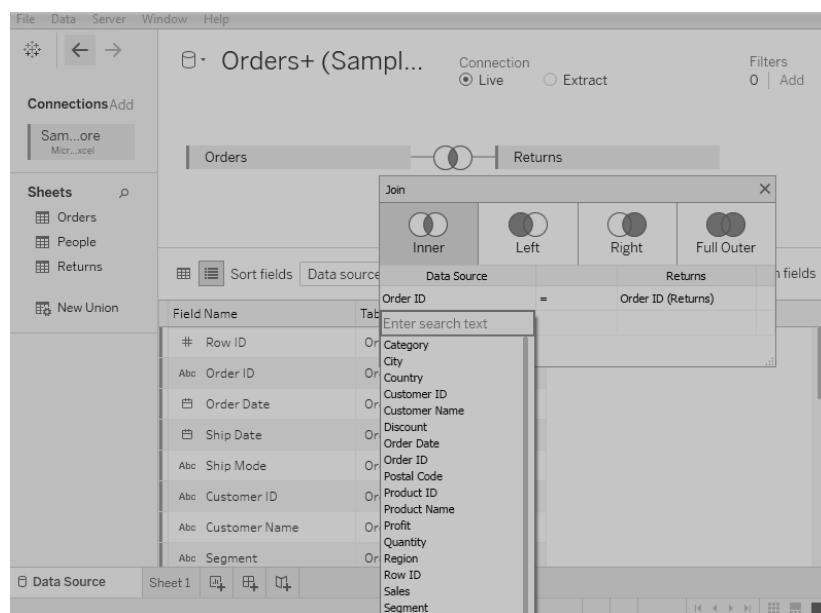
Tableau automatically creates a type of join between two tables, but it can be changed as per need.

- Click on the middle of two circles that showing the auto-created join.
- After clicking, a popup window appears which shows all the four types of the joins.
- In below screenshot, you can see all the joins such as **inner join**, **left outer join**, **right outer join**, and **full outer join**.



How to Edit Join Fields in Tableau

- Also, you can change the fields by clicking the Data Sources option to add a new join clause that is available in the join popup window.
- While selecting the field, you can search for the field using a search text box.



Data Blending in Tableau

- Data Blending is a powerful feature of Tableau. It is used to analyze the data in a single view from a related data in multiple data source.

For example: Suppose a Sales data is present in a relational database and Sales Target data in an Excel sheet.

- Now, for comparing the actual sales with the target sales, you have blended the data based on common dimensions to get access into the Sales Target measure.
- The two data sources are involved in data blending are referred as the primary data source and the secondary data source.
- A left join is built between the primary and the secondary data source with all the data rows from primary and only matching data rows from the secondary data source.

How to do Data Blending

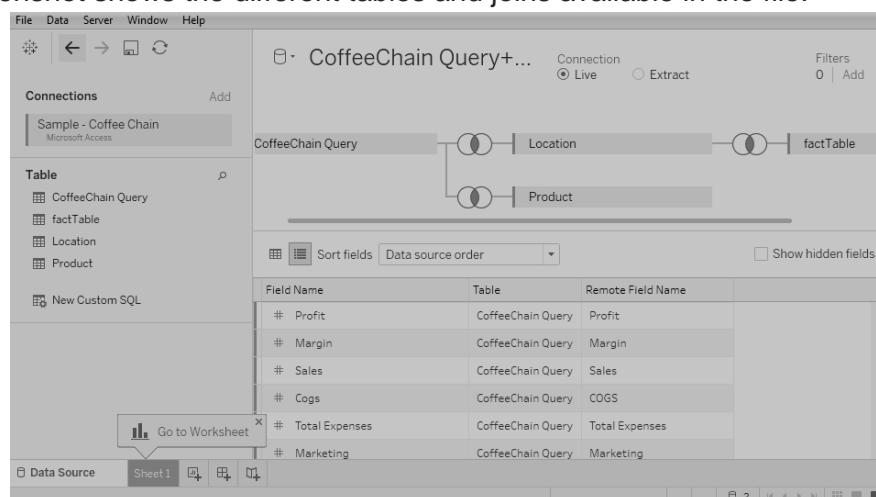
Tableau has two inbuilt data sources that are **Sample coffee chain.mdb** and **Sample-superstore**, which can be used to illustrate data blending.

- First, load the sample coffee chain into Tableau and visualize its metadata.

The screenshot shows the Tableau Data Source interface. At the top, it says "File Data Server Window Help". Below that is a toolbar with icons for file operations. The main area shows a connection named "CoffeeChain Query (...)" with "Live" selected. On the left, under "Connections", there is a list for "Sample - Coffee Chain Microsoft Access". Under "Tables", there are four items: "CoffeeChain Query", "factTable", "Location", and "Product". At the bottom, there is a "Go to Worksheet" button.

- Go to the **data source** below **connect** → click on **MS Access** database file and browse for the **sample coffee chain** file.

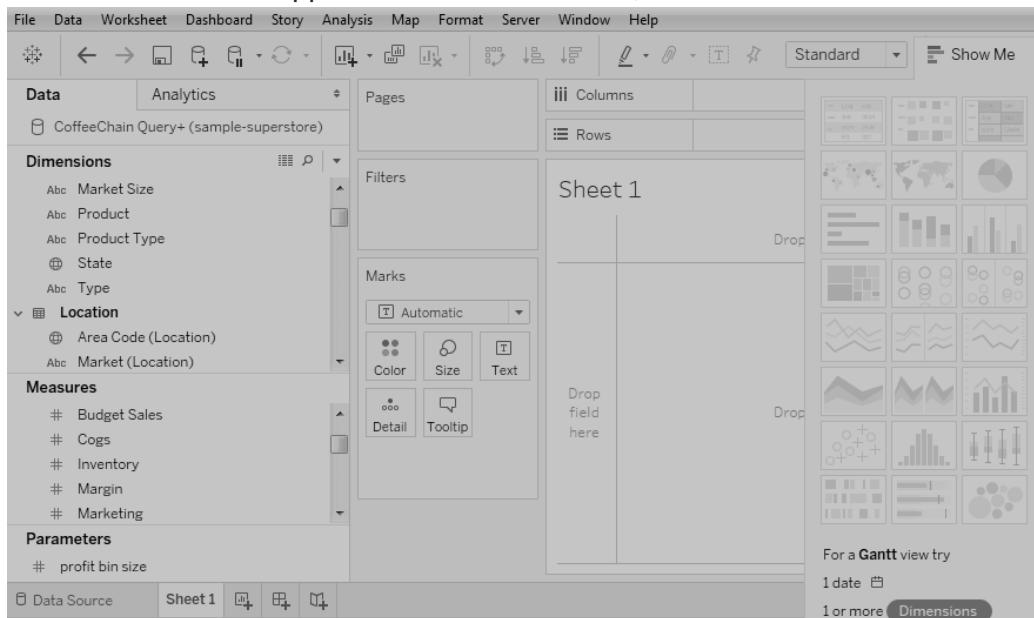
The below screenshot shows the different tables and joins available in the file:



How to Add Secondary Data Source

Add the secondary data source which name is **Sample-superstore.xls** with the following steps:

- Click on Add button of the data source.
- Add a new connection to use cross-database joins to a file and choose the data source such as Microsoft Excel.
- Now, both the data sources appear on the Data window, as shown in the below screenshot.



Blending the Data

You can integrate the data from **sample-superstore** and **sample coffee chain** sources based on a common dimension.

- A small chain image appears in the dimension field that is **State**. It indicates the common dimension between the **sample coffee chain** and **sample-superstore** data sources.
- Drag the field **State** from the primary data source into the **rows shelf** and also drag the field **Profit** from the secondary data source into the **Columns shelf**.
- Then, select the **horizontal bar** option from **Show Me** to get the graphical visualization.
- The chart shows how the **profit** varies for each **State** in both the **sample coffee chain** and **sample-superstore** data sources. Shown in the below screenshot:

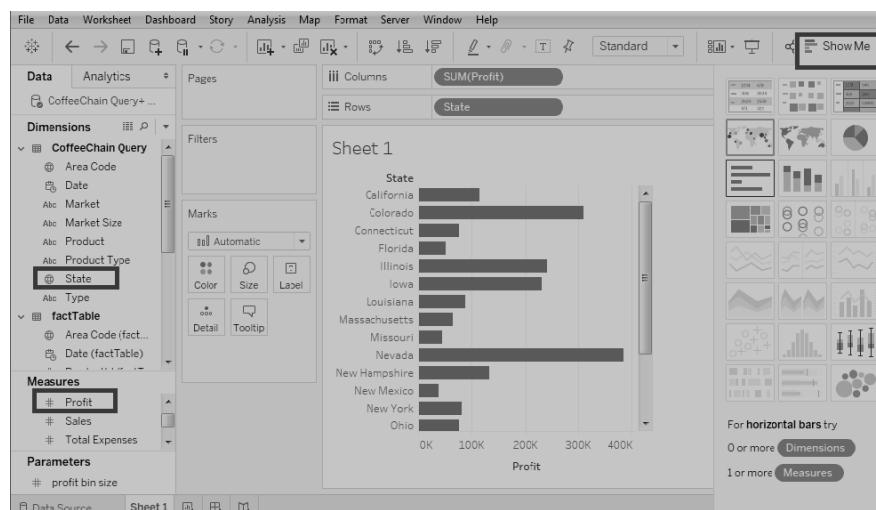
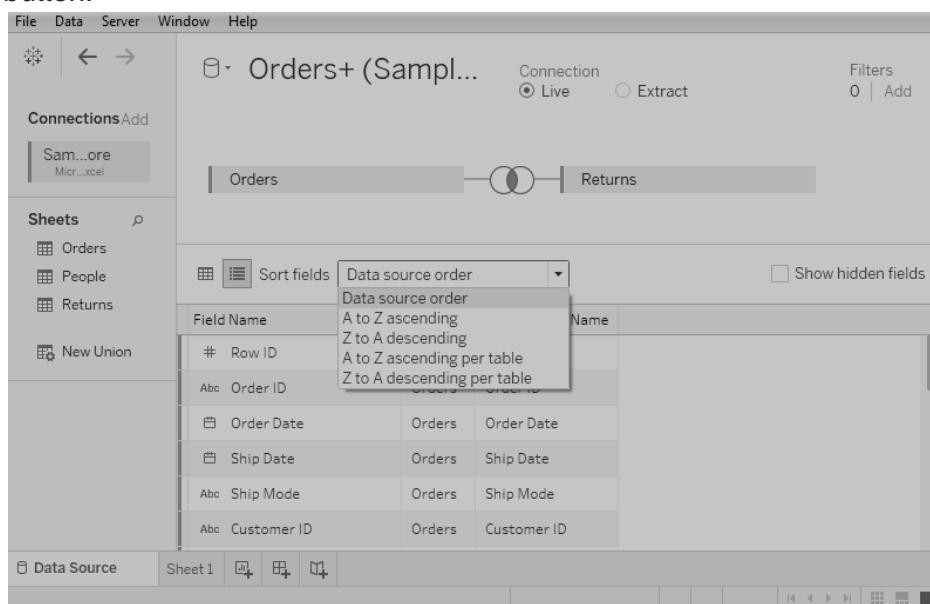


Tableau Data Sorting

- In the data source, data can be stored based on the user requirement. It can be sorted using data source order such as **A to Z ascending**, **Z to A descending**, **A to Z ascending per table** and **Z to A descending per table**.
- Once the data is connected with Tableau, data sorting is done using the **Sort Fields** option. The **Sort Fields** option is present in the **Data Source** tab.

There are two ways to sort the data in Tableau:

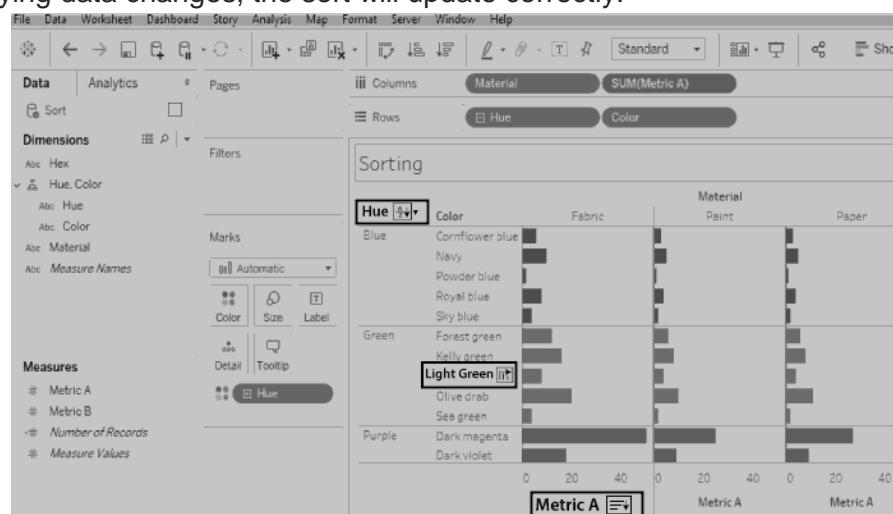
- Manual sorting:** Manual sorting is a sort that rearranges the order of dimension fields by dragging them next to each other in **ad hoc** fashion.
- Computed sorting:** The computed sorting is a sort which is directly applied on the axis using the sort dialog button.



When viewing a visualization, data can be sorted using the single-click option from a header, an axis or field label.

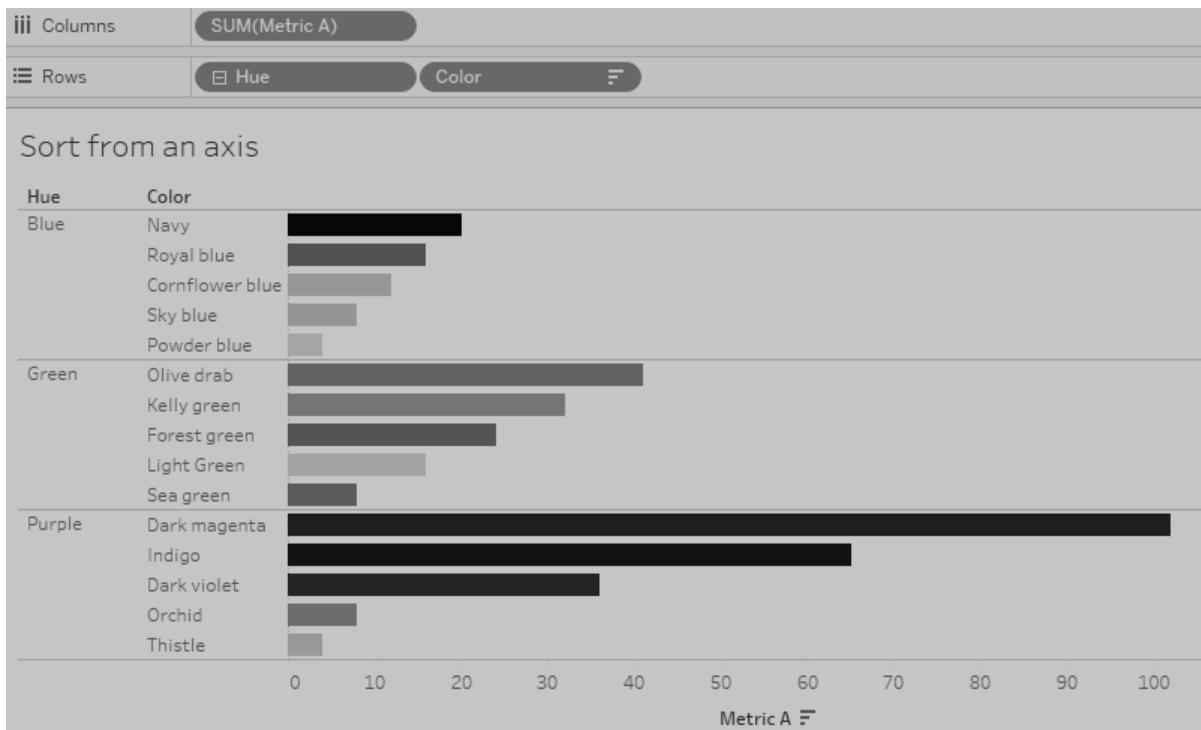
There are many ways to sort a visualization with single click sort buttons:

- In all cases, one-click means sorts the data in ascending order, and two-click means it sorts the data in descending order, and three-click means clear the sorts.
- If the underlying data changes, the sort will update correctly.



Sort from an Axis

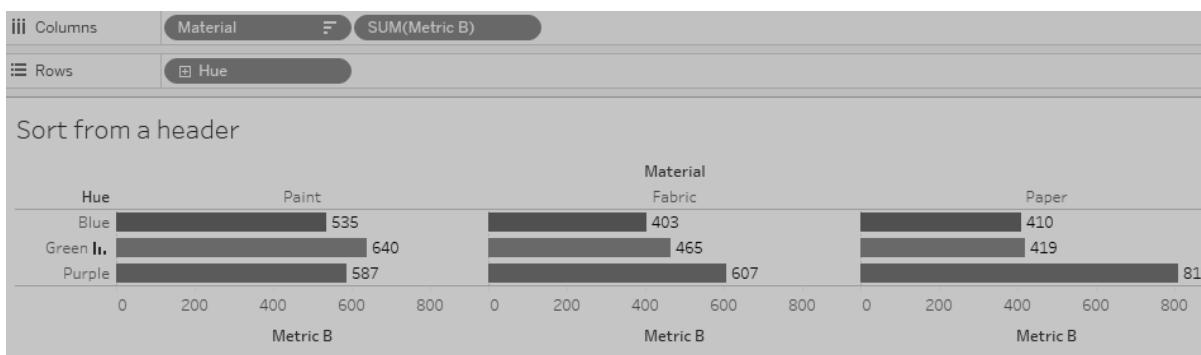
- Float over a numerical axis to get the sort icon.
- Click that icon to sort.



- In the above example, the sort is applied on **Color** rows based on the values of **Metric A**.
- If there are hierarchical dimensions shown in above example, that type of sort is used on the inner dimension.
- Here, it means that **Color** rows will sort inside **Hue**. Dark magenta cannot be sorted at the top of the viz because it should stay inside the Purple Hue.

Sort from a Header

- Float over a header to get the sort icon.
- Click that icon to sort.

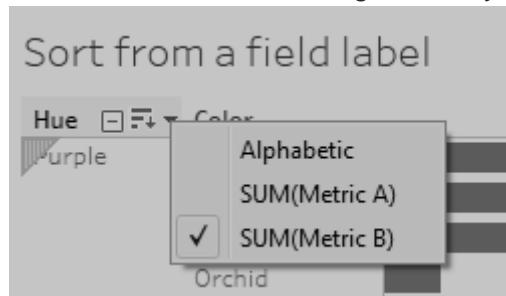


In the above example, the sort is applied to a **Material** column such as Paint, Paper and Fabric based on the values of Green since the header is used for the sort.

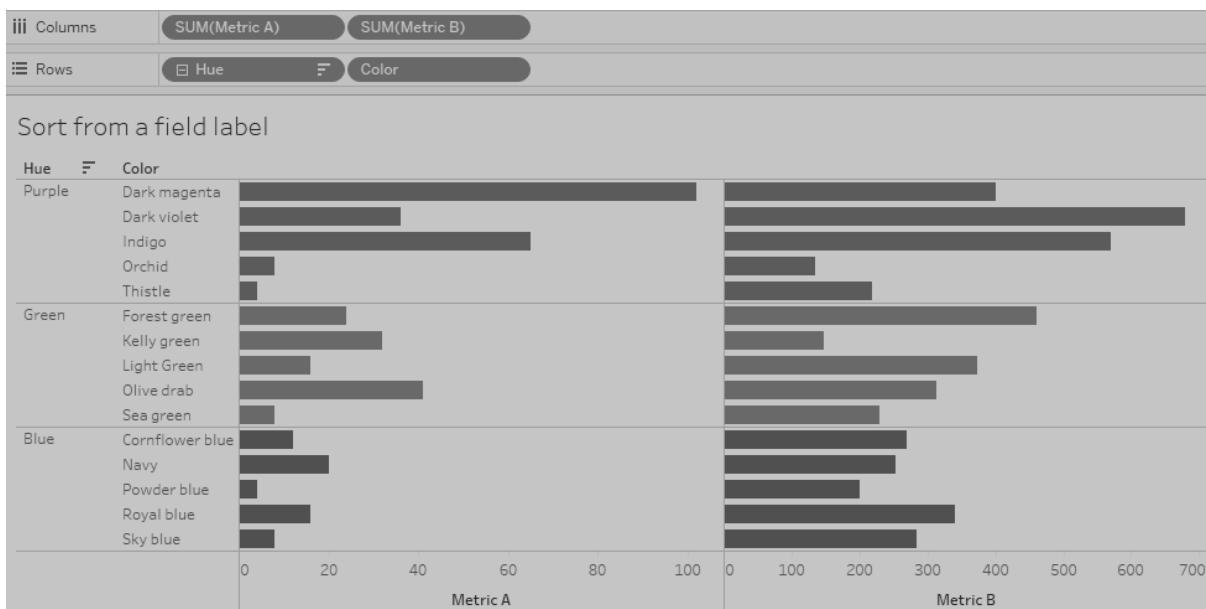
Sort from a Field Label

1. Float over a field label to get the sort icon.

- For a field label, the sort icon is slightly different from an axis or a header. Alphabetical sorting is the default option, but there is also a menu for choosing to sort by a field in the view.



2. Click on the A-Z icon to sort alphabetically, or open the menu to see a list of fields which is possible to sort according to the field. Then, click on sort after the icon switches to the bar icon.



In the above example, the sort is applied to the outermost dimension such as **Hue** is based on **Metric B**. (Metric B is aggregated for all the colors inside each Hue, and Hue is sorted as first is Purple, then Green, then Blue.)

Missing Sort Icons in Tableau

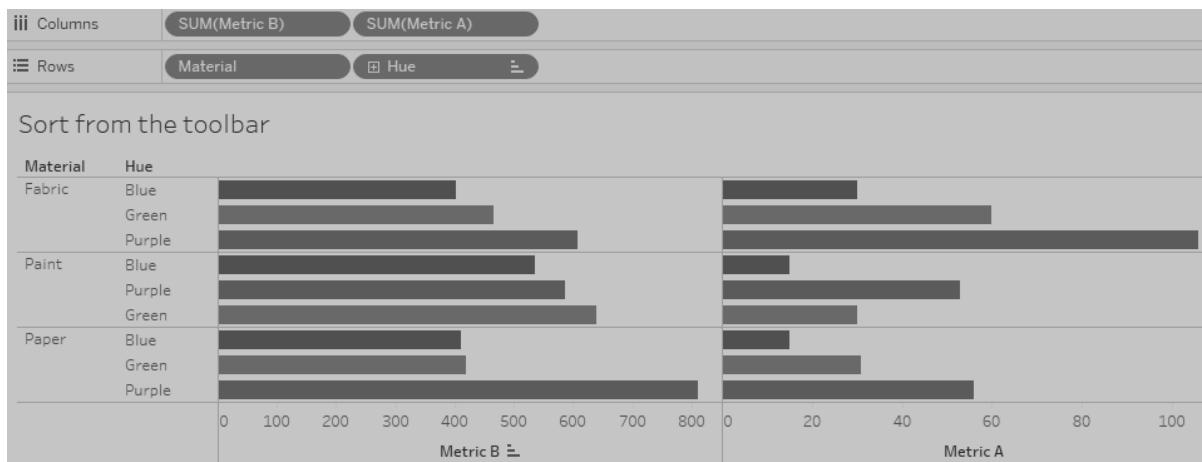
- If the sort icons do not appear, then this functionality may be turned off, or it cannot be possible to sort the view.
- For example**, Scatter plots cannot be sorted by a numerical axis because the data entirely determine the position of the marks. No sort icon will appear on the axis in scatter plots.

Sort Options While Authoring in Tableau

- In an authoring environment, there are some additional sorting options, such as:

Sort from the Toolbar

- Select the dimension which you want to sort.
- The default behavior has to sort the deepest dimension If you do not select a field before sorting.
- Choose the appropriate sort button such as ascending or descending order in the toolbar.



- In the above example, the sort is applied on **Hue** unless the **Material** field is selected before sorting. In the case of **Metric B**, the toolbar sort applies to the leftmost measure.
- And to sort by **Metric A**, it would be necessary to use another method of sorting or reverse their order on the Columns shelf. (To see the effect of sorting by Material, Hue is removed from the view. this makes it easy to see how the sort is computed.)

Sort by Drag and Drop

To sort manually, select a header in **Viz** or on a **legend** and drag it to the current location shown below:

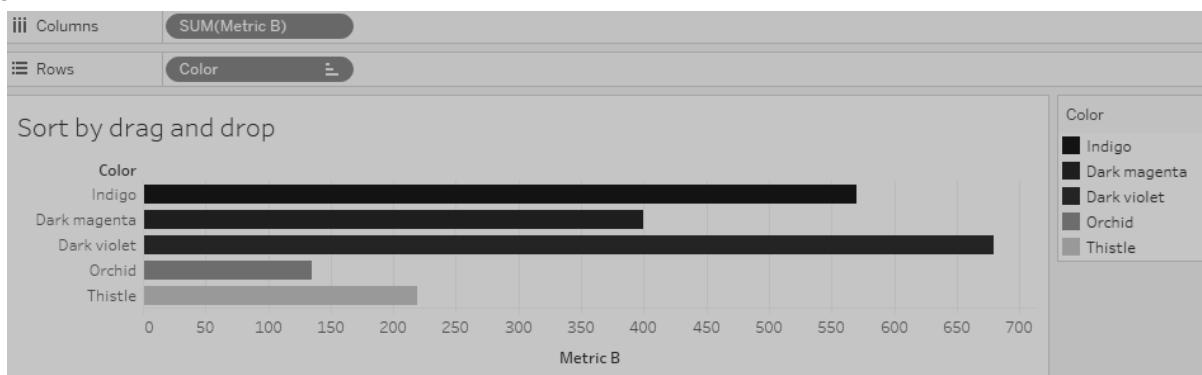
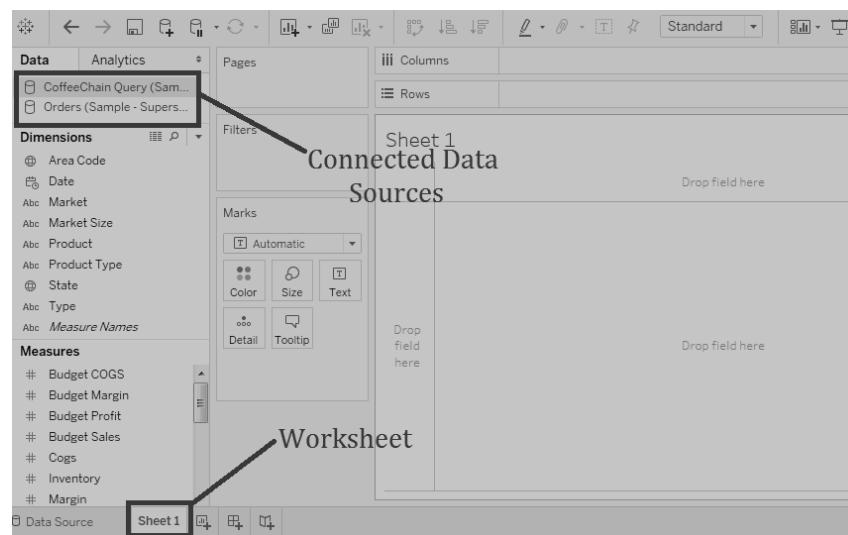


Tableau Replacing Data Source

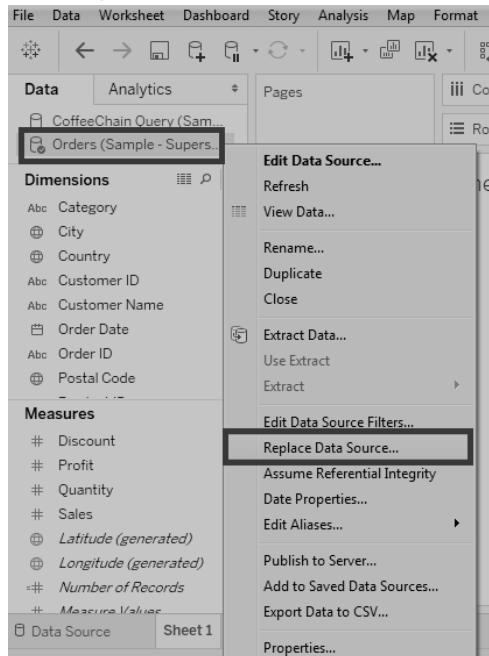
- Tableau can connect multiple data sources within a single workbook.
- The different data sources can be used to create various dashboards and sheets in Tableau.
- In some cases, the data source is needed to replace with the updated file.
- Tableau has the data source replacing feature which can replace the data source.
- This feature does not affect the already built visualizations using the old data source.
- It is important to keep or replace all the used dimensions and measures while replacing the data source.
- The data source connected in Tableau can be replaced with another data source. The procedure for replacing data source is shown in the below screenshot:



Step 1: Go to the connected data source or multi connection in Tableau.

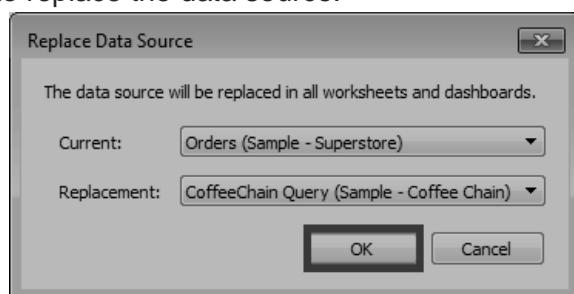
Step 2: Then,

- Select the data source which you want to replace.
- Right click on the data source.
- Select the "Replace Data Source" option.



Step 3: It opens the "Data Source Replacement" window.

- Fill the **Current** option.
- Then fill the **Data Source Replacement** option.
- Click on **OK** button to replace the data source.



4.10 Tableau Operators and Calculation

Tableau Calculation

There are four necessary components to the calculation in Tableau:

1. **Function:** Function statements are used to transform the values or members in a field.
For Example: The format of all functions in Tableau such as SUM (expression).
2. **Fields:** Field is dimensions and measures from your data source.
For Example: A field in a calculation is often surrounded by brackets [] such as [Sales].
3. **Operators:** Operator is a symbol that denotes an operation between the operands.
For Example: The types of operators you can use in Tableau calculations, as well as the order they are performed in a formula such as +, -, *, /, %, ==, =, >, <, >=, <=, !=, <>, ^, AND, OR, NOT, ()
4. **Literal Expression:** Literals expression are represent the constant values "as is" such as "profitable" and "unprofitable".

For Example: See the below calculation

1. IF [Profit per Day] > 5000 THEN "Highly Profitable"
2. ELSEIF [Profit per Day] <= 0 THEN "Unprofitable"
3. ELSE "Profitable"
4. END

The component of the above calculation can be further divided into the following:

1. **Functions:** IF, THEN, ELSEIF, ELSE, and END.
2. **Field:** Profit per Day.
3. **Operators:** > and <=.
4. **Literal Expression**
 - **String Literals:** "Highly Profitable", "Unprofitable", and "Profitable".
 - **Numeric Literals:** 5000, and 0.

Note: Not all calculation needs to contain all the four components.

Here is some important point for literal expression syntax:

- Numeric literals are written as numbers.

Example: 27 or 1.3567

- String literals are written with quotation marks.

Example: "profitable."

- Date literals are written with the # symbol.

Example: # June 8, 2018 #

- Boolean literals are written as either true or false.

Example: "True" or "False"

- Null literals are written as null.

Example: "NULL"

Two more calculations contain by Tableau

1. **Parameters:** Parameter is a placeholder variable that can be inserted into calculations to replace the constant values.

A parameter in a calculation is surrounded by brackets [].

For Example: [Profit Bin Size]

2. **Comments:** Comment is defined as the notes about a calculation or its parts, but comments not included in the computation of the calculation.

To enter a comment in a calculation, use two forward slashes //.

For Example

- SUM ([Sales]) / SUM ([Profit]) // Nick's calculation
- // to be used for profit ratio
- // Do not edit

Tableau Operators

- An operator is a symbol for performing specific mathematical and logical operations through the compiler.
- Tableau has several numbers of operators which are used to create calculated fields and formulas.
- Here are the types of operators with their order of precedence of operation:

Types of Operators

1. General operators
2. Arithmetic operators
3. Relational operators
4. Logical operators

1. General Operators

Here are some general operators supported by Tableau. These operators act on the character, numeric, and date data type.

- **Addition (+):** By the help of the addition operator, we can add the two numbers, concatenate two strings and also add days to dates.

Example: $10+15=25$

```
Sales+ profit  
'XYZ'+ 'PQR'= XYZPQR  
# June 8, 2018 # + 7= # June 15, 2018 #
```

- **Subtraction (-):** By the help of the subtraction operators, we can subtract two numbers and subtract days from dates.

Example: $- (10+15) = -25$

```
# June 8, 2018 # - 7= # June 1, 2018 #
```

2. Arithmetic Operators

Here are some arithmetic operators supported by Tableau. All these operators act only on the numeric data type.

- **Multiplication (*):** we can multiply two numbers by the help of multiplication operator.

Example: $5 * 2 = 10$

- **Division (/):** we can divide two numbers by the help of the division operator.

Example: $15 / 5 = 3$

- **Modulo (%):** modulo operator gives you the remainder of the numeric division.

Example: $17 \% 2 = 1$

- **Power (^):** raised to the power.

Example: $2 ^ 2 = 4$

3. Relational Operators

- Here are the relational operators supported by Tableau.
- These operators are used in the expressions.
- Each relational operator compares two numbers, strings, or dates and returns a Boolean value (True or False).
- However, Boolean operators themselves cannot be compared using these operators.

Equal to (= or ==): It compares two numbers, strings or two dates to be similar and returns the Boolean values, true if they are equal else returns False.

Example: $'hello' = 'hello'$, returns **True**

$'2' = '10/5'$, returns **True**

$'Hello' = 'hey'$, returns **False**

Not equal to (! = or <>): It compares two numbers, two strings, or dates to be unequal. And returns the Boolean values, true if they are equal else returns False.

Example: 'cold' <> 'hot'

'13' != '24/2'

Greater than (>): It compares two numbers, two strings or two dates where the first argument is greater than second, it Returns the Boolean value True else returns False.

Example: [Profit] > 10000

[Category] > 'Q'

[Ship date] > #April 1, 2018#

Less than (<): It compares two numbers, two strings or two dates, where the first argument is smaller than the second. It returns the Boolean value True, else returns false.

Example: [Profit] < 10000

[Category] < 'Q'

[Ship date] < #April 1, 2018#

4. Logical Operators

Here are the logical operators supported by Tableau. These operators are used in an expression whose result is a Boolean value (True or False).

- **AND:** If the Boolean values present on both sides of AND operator is evaluated to be TRUE, then the result is TRUE. Else the result is FALSE.

Example: [Ship Date] > #April 1, 2018# AND [Profit] > 20000

- **OR:** If anyone or both of the Boolean values present on both sides of the OR operator analyses to be TRUE, then the result is TRUE. Else the result is FALSE.

Example: [Ship Date] > #April 1, 2018# OR [Profit] > 20000

- **NOT:** This operator reverses the Boolean value of the expression.

Example: NOT [Ship Date] > #April 1, 2018#

Precedence of Operator

- The below table is describing the order of precedence of the operator.
- The top row of below table has the highest precedence. Some operators in the same row have the same precedence.
- If two operators have the same precedence, they are analyzed from left to the right in the formula.
- Parentheses can also be used in the same order, and the inner parentheses are evaluated before the outer parentheses.

Order of Precedence	Operators
1	-(negate)
2	^(power)
3	*, /, %
4	+, -
5	==, >, <, >=, <=, !=
6	NOT
7	AND
8	OR

4.11 Functions and Calculations in Tableau

Tableau Functions

- Data analysis involves a lot of calculations.
- In Tableau, the calculation editor has applied calculations to the fields being analyzed.
- Tableau has multiple inbuilt functions which help in creating expressions for complex calculations.

There is a list of Tableau functions that are categorized into five parts:

1. Number functions
2. String functions
3. Date functions
4. Logical functions
5. Aggregate functions

1. Number Functions

Number function is a function that uses for the numeric calculations. They take only numbers as inputs.

Let's see some essential examples of number functions:

- **Ceiling (Number):** It rounds a number to the nearest integer of equal or greater values.
Example: CEILING (4.155) = 5
- **Power (Number, Power):** It raises the number to the specified power.
Example: POWER (2^3) = 8
- **Round (Number, Decimals):** It rounds the number to a specified number of digits.
Example: ROUND (5.14522) = 5.14

2. String Functions

String functions are used for the manipulation of the string.

Let's see some essential examples of string functions:

- **LEN (String):** LEN string returns the length of the string.
Example: LEN ("Tableau") = 7
- **LTrim (String):** It returns a string that contains a copy of the specified string with no leading (LTrim) or trailing (RTrim) spaces.
Example: LTrim (" Tableau ") = "Tableau"
- **REPLACE (String, Substring Replacement):** It searches the string for substring and replaces it. If the substring is not found, that string is not changed.
Example: REPLACE ("Green yellow Green", "yellow", "Red") = "Green Red Green"
- **UPPER (String):** It returns the string with all uppercase characters.
Example: UPPER ("Tableau") = "TABLEAU"

3. Date Functions

Tableau has many date functions, and all the date functions use the **date part**, this is the string indicating part of the date such as day, month, or year.

Let's see some essential examples of date functions:

- **DATEADD (date_part, increment, date):** It's added an increment to the date. The type of increment is specified in the date_part.

Example: DATEADD ('month', 5, #2018-06-15#) = 2018-11-15 01:00:00 AM

- **DATENAME (date_part, date, start_of_week):** It returns **date_part** of date as a string. And the **start_of_week** parameter is optional.

Example: DATENAME ('month', #2018-03-15#) = "March"

- **DAY (date):** It returns the day of the given date in integer form.

Example: DAY (#2018-04-12#) = 12

- **NOW ():** It returns the current time and date.

Example: NOW () = 2018-04-15 1:08:21 PM

4. Logical Functions

These functions evaluate some single values and produce a Boolean output.

See some essential examples of logical function:

- **IFNULL (expression1, expression2):** If the result is not null, then **IFNULL** function returns the first expression, and if it is null, then it returns the second expression.

Example: IFNULL ([Sales], 0) = [Sales]

- **ISDATE (string):** If the string argument can be converted to a date, the **ISDATE** function returns TRUE, and if it cannot, it returns FALSE.

Example: ISDATE ("12/06/99") = "TRUE"

ISDATE ("14/06/99") = "FALSE"

- **MIN (expression):** The **MIN** function returns the minimum result for each record.

5. Aggregate Functions

Let's see some essential examples of aggregate functions:

- **AVG (expression):** It returns the average of all the values in the expression. **AVG** is used only with numeric fields. And the Null values are ignored.

- **COUNT (expression):** It returns the number of items in a group and the Null values are not counted.

- **MEDIAN (expression):** It returns the median of an expression over all records. Median can only be used with numeric fields, and Null values are ignored.

- **STDEV (expression):** It returns the statistical standard deviation of all values in the given expression based on a sample of the population.

Tableau Numeric Calculations

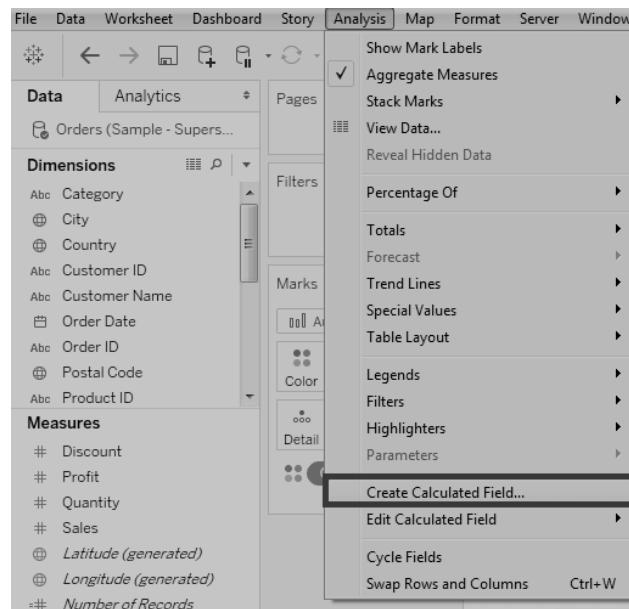
- In Tableau, numeric calculations are done using a wide range of inbuilt functions available in the formula editor.

Let's see how to apply calculations to the fields.

- The calculations are simple as subtracting the values of two fields or using an aggregate function to a single field.
- Here are the steps to create a calculation field and use numeric functions in it.

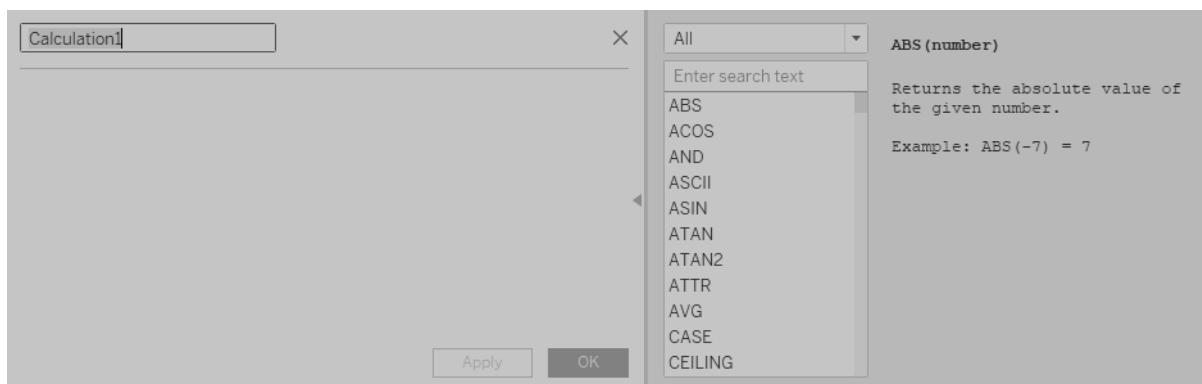
How to Create a Calculated Field

- After connecting to a data source such as Sample-Superstore.
- Go to Analysis menu.
- Click on Create Calculated Field as shown in the below image.

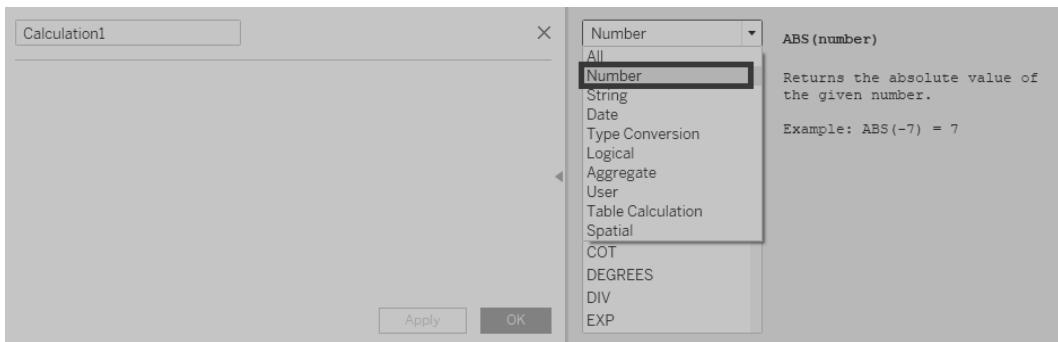


Calculation Editor in Tableau

The above process opens a calculation editor which lists all the functions available in Tableau.

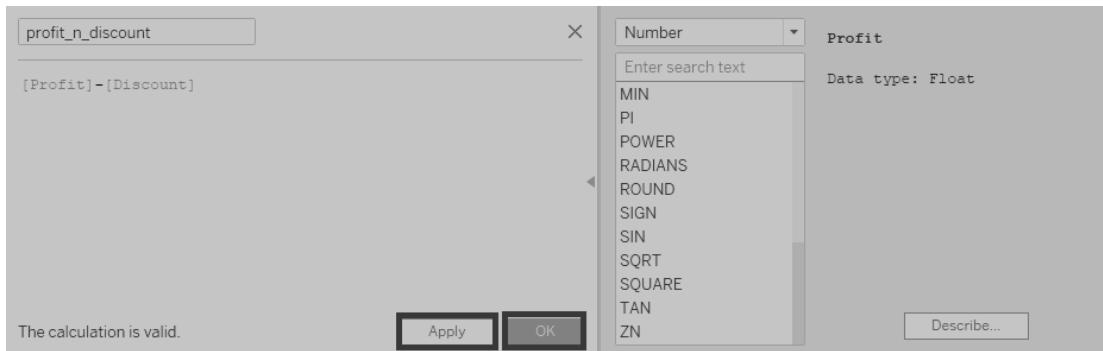


You can change the dropdown value and only see the related functions to numbers.



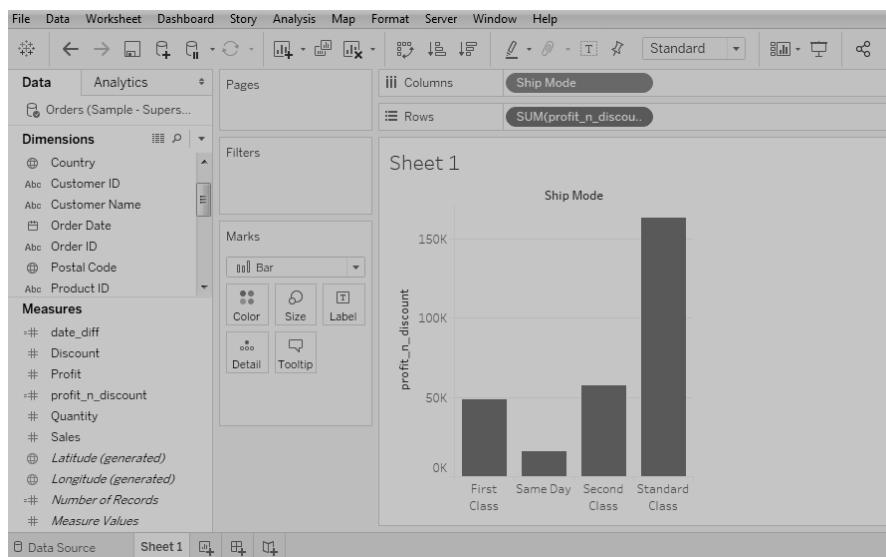
Create a Formula

To visualize the difference between **Profit** and **Discount** for different shipping mode of the products, create a formula that subtracts the **Discount** from the **Profit**, as shown in the below image, and the name of this field is **profit_n_discount**.



Using the Calculated Field

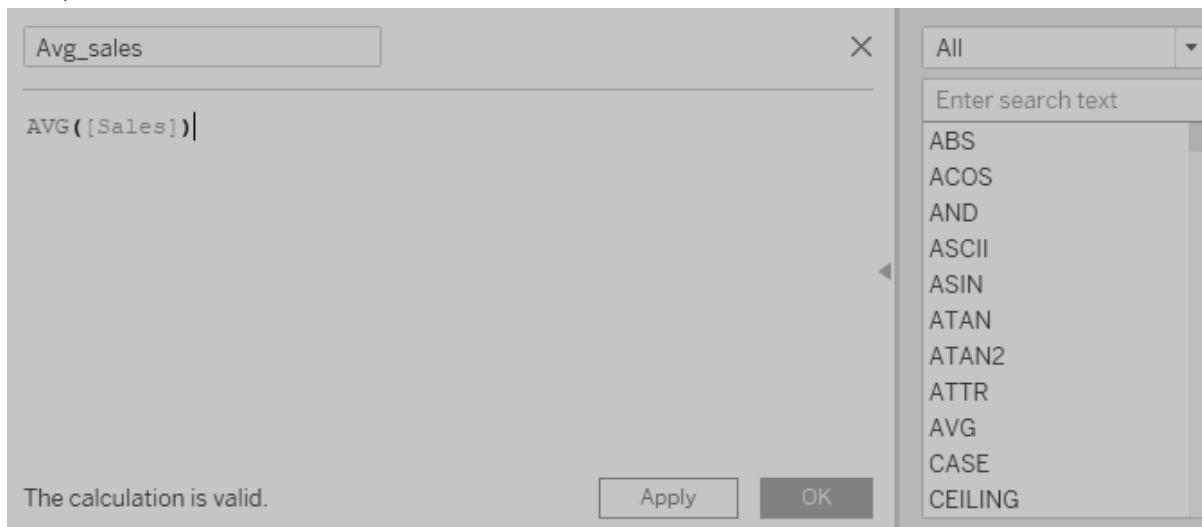
- The above-calculated field can be used in the view by dragging it to the Rows shelf as shown in the below screenshot.
- It produces a bar chart that shows the difference between profit and discount for different shipping modes.



Applying the Aggregate Calculations

You also can create a calculated field using an aggregate function.

- First, create AVG (sales) values for different ship mode.
- Then, Write the formula in the calculation editor as shown in the below screenshot.



- Click **OK** and dragging the **Avg_Sales** field to the Rows shelf, then you get the following view.

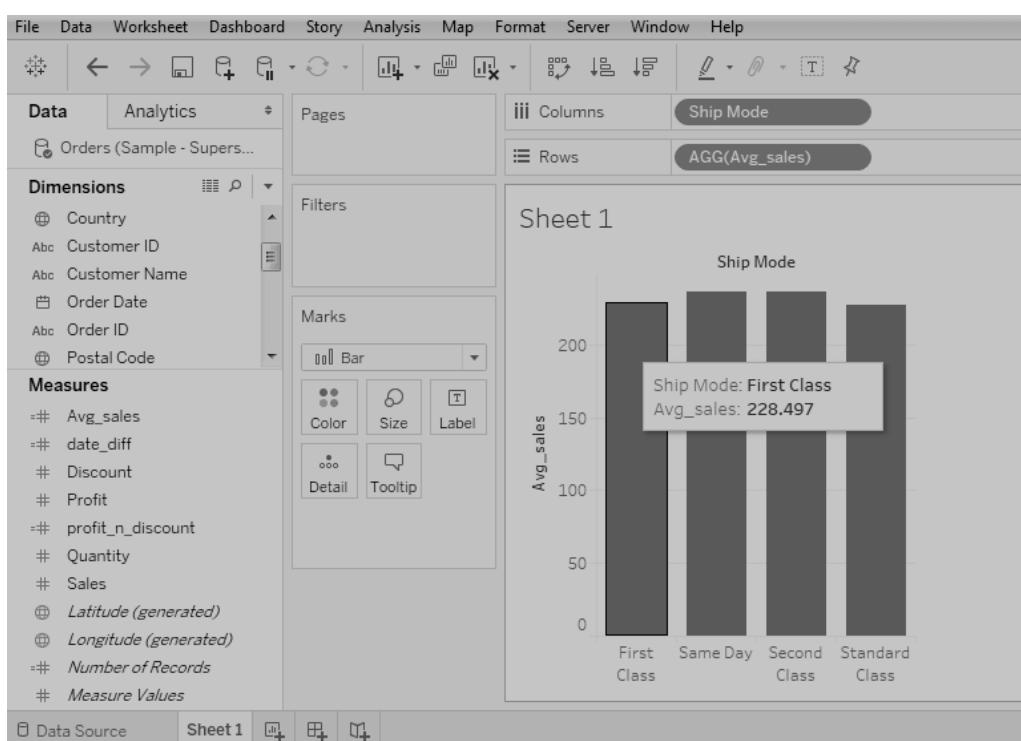
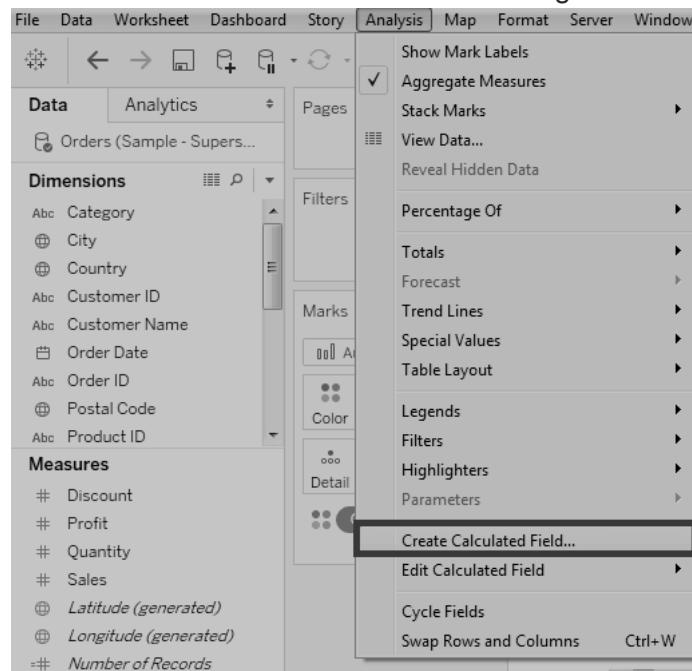


Tableau String Calculations

- Tableau has many inbuilt string functions used for string manipulation such as concatenating, comparing, and replacing few characters from a string, etc.
- Here are some steps to create a calculation field and use string function in it:

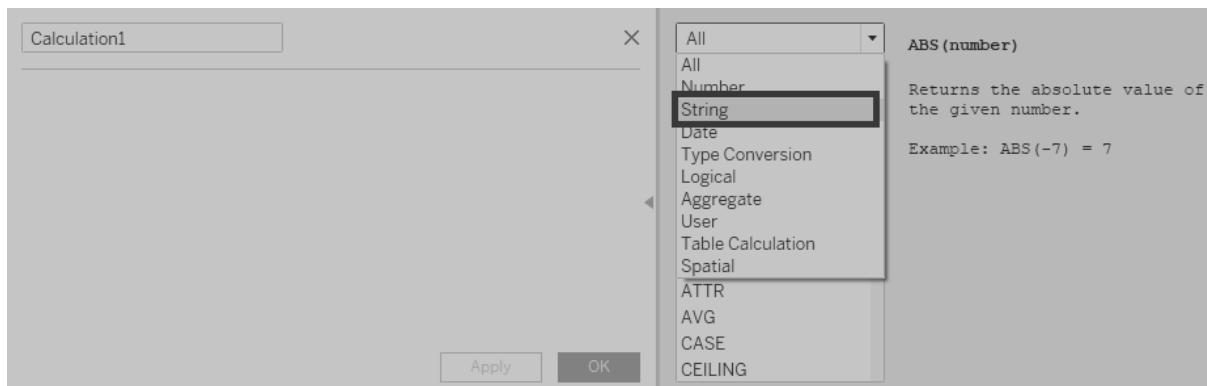
How to Create Calculated Field

- After connecting to a data source such as **Sample superstore**.
- Then, go to the **Analysis** menu.
- And click '**Create Calculated Field**' as shown in the below image.

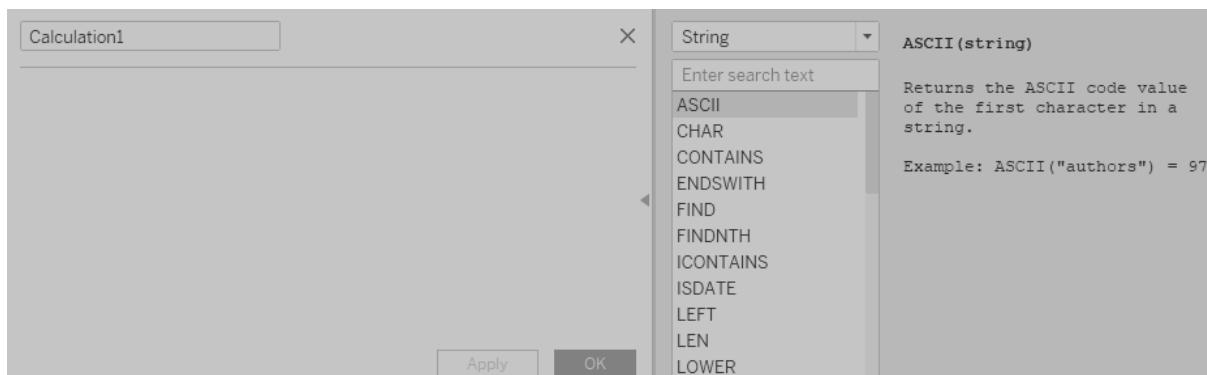


Calculation Editor in Tableau

The above process opens a calculation editor that contain all the functions available in Tableau.

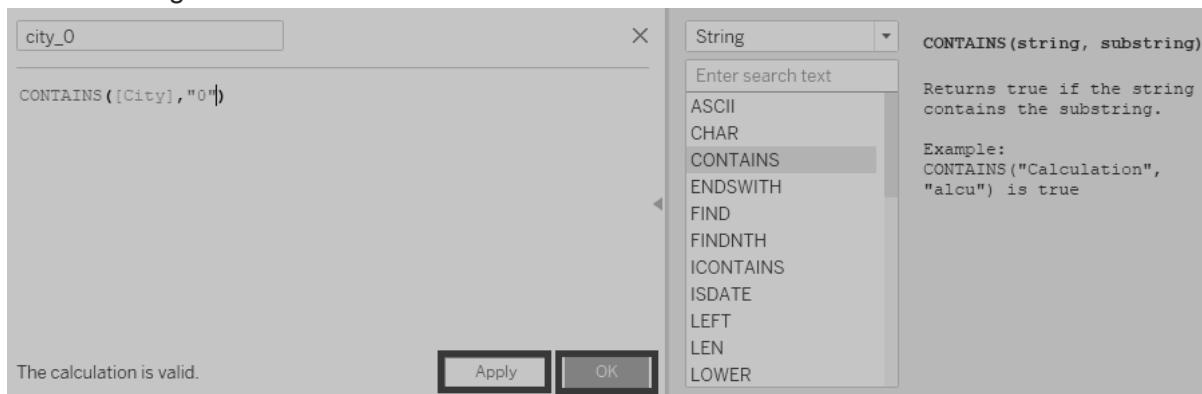


You can change the dropdown value and only see the related functions to strings.



Create a Formula

If you want to find out the **Sales** in the **Cities**, that contain the letter "A", create the formula as shown in the below image.



How to Use the Calculated Field

To see the created field into a graphical representation, you can drag **City** field into the Rows shelf and drag the **Sales** field into the Columns shelf.

The below image shows the Sales values for Cities:

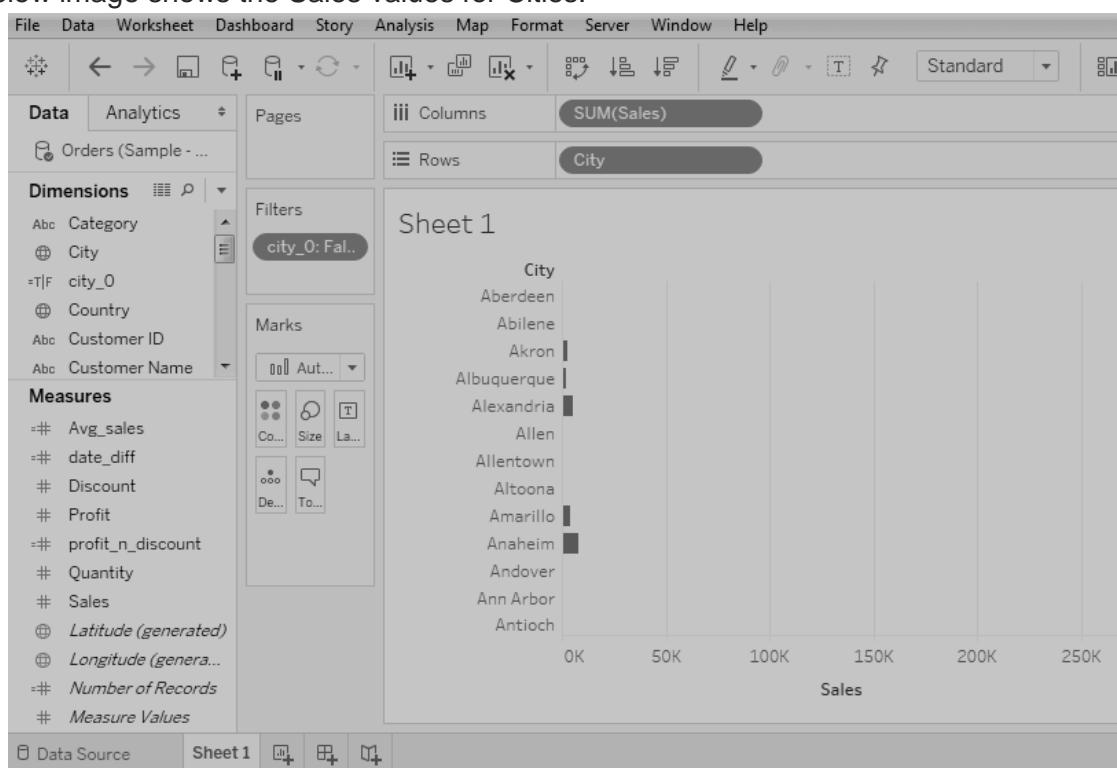


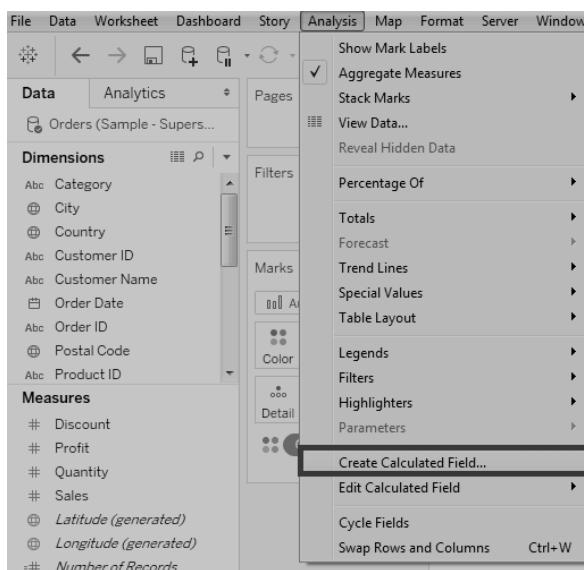
Tableau Date Calculations

- Tableau can provide a large number of inbuilt functions such as dates.
- Dates are one of the critical fields which are extensively used in most of the data analysis.
- You can manipulate the simple date such as adding or subtracting days from a date.
- Also, you can create complex expressions that include dates.

Here are the steps to create a calculation field and use date functions in it.

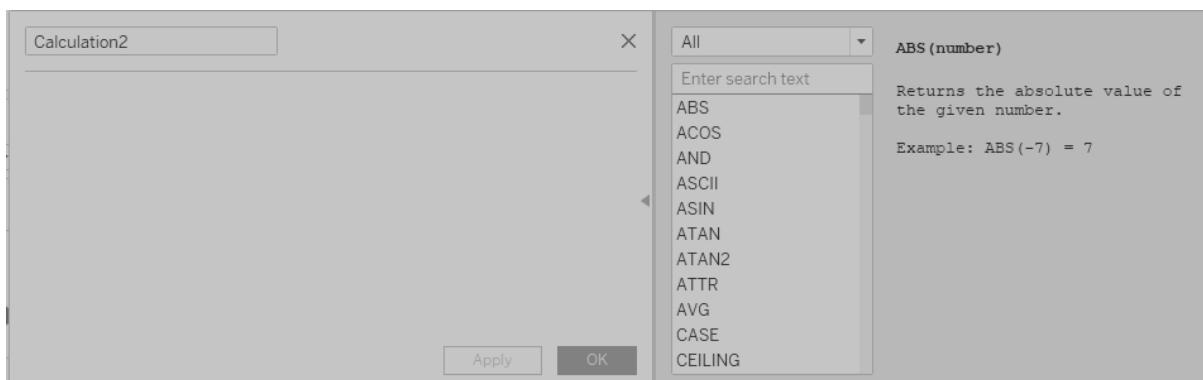
How to create a calculated field

- After connecting to a data source such as **sample superstore**.
- Then go to the **Analysis** menu.
- Click on the '**Create Calculated Field**' as shown in the below image.

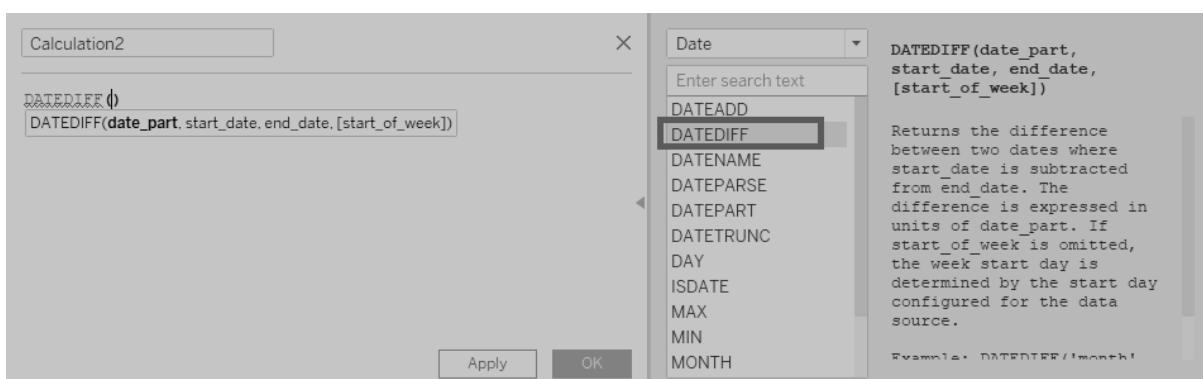


Calculation Editor in Tableau

The above process opens a calculation editor that lists all the functions available in Tableau.

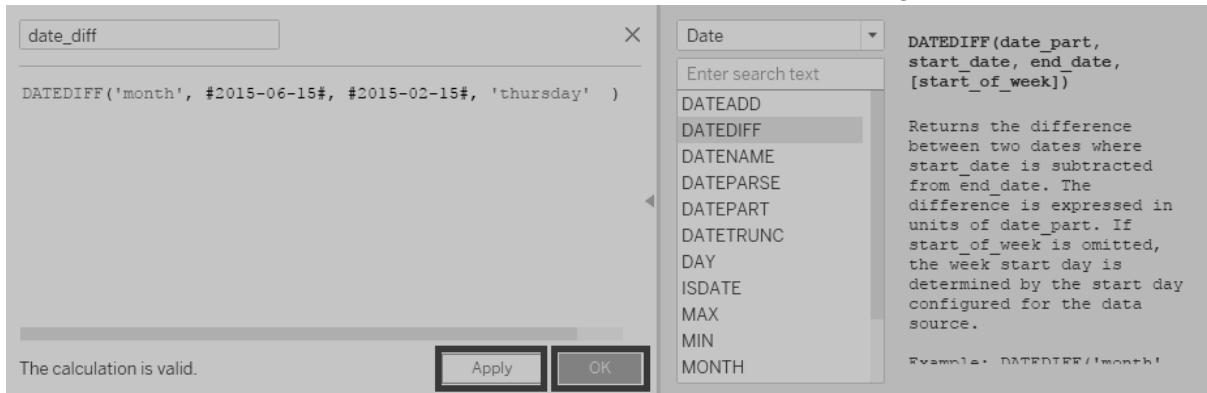


You can change the dropdown value and only see the related functions to date, shown in the below image:



Create a Formula

If you want to find out the **Sales** volume along with the difference in the date of sales in months from 15/06/2015 to 15/02/2015, create the formula as shown in the below image.



Using the Calculated Field

- To see the created field in graphical representation, you can drag **Month** and **date_diff** field into the Rows shelf and drag the Sales field to the Columns shelf.
- Also, drag the **ship Date** with months.

The below screenshot shows the Sales volume along with the difference in the date of sales:

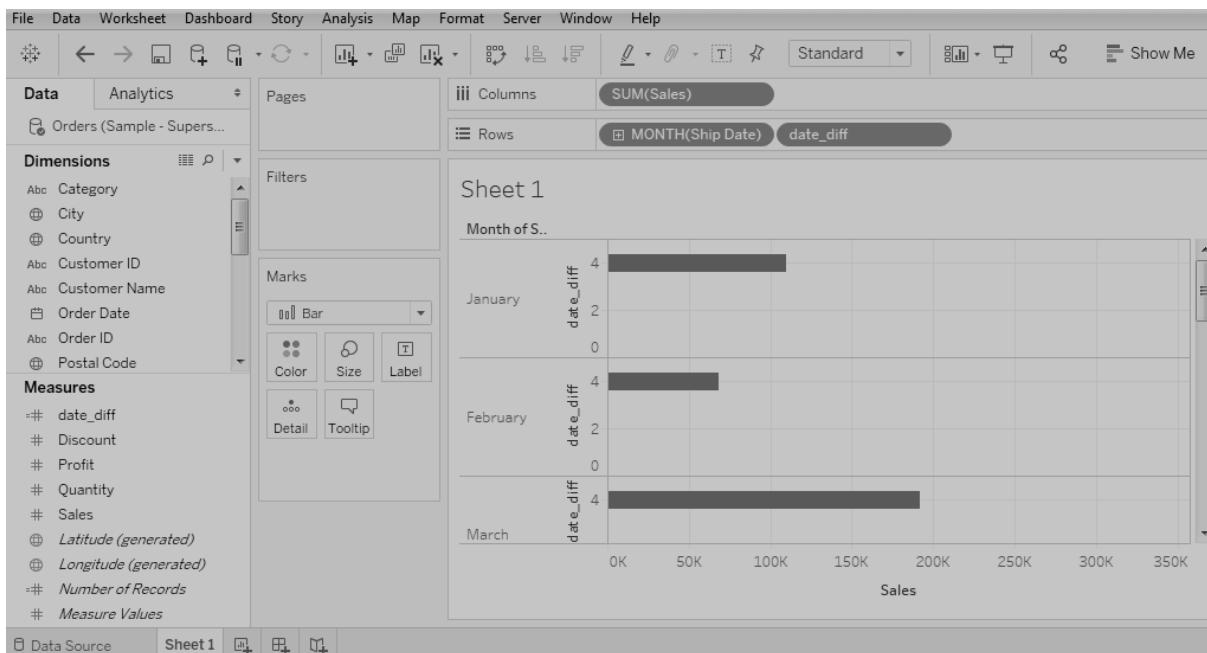
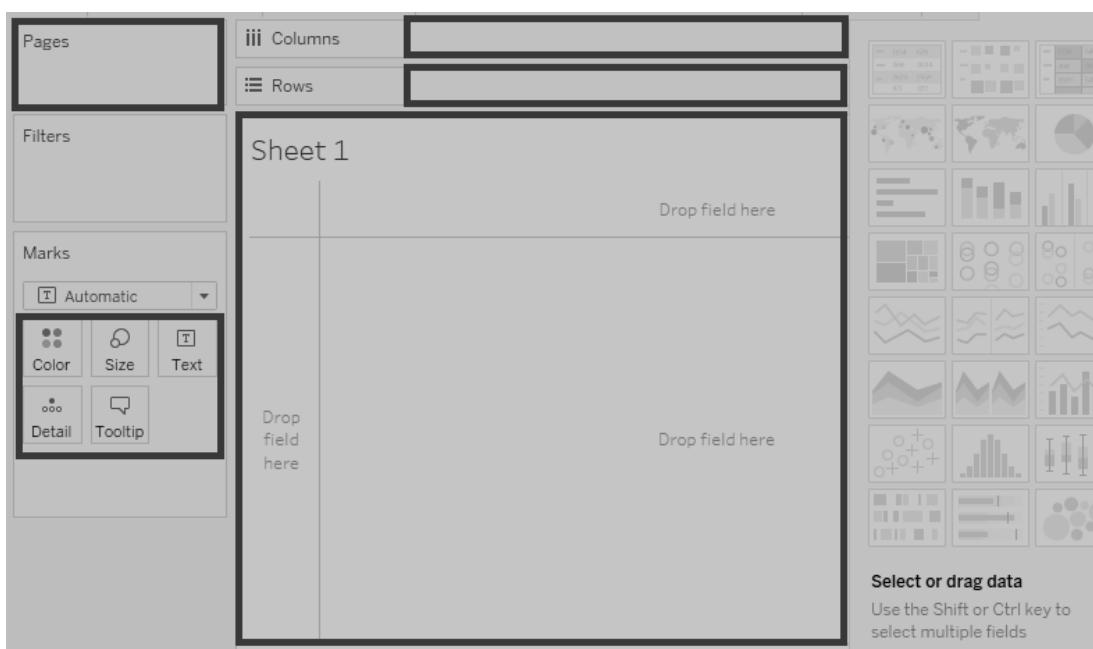


Tableau Table Calculations

- A table calculation is a transformation that applies to the values in a visualization.
- Table calculation is a special type of calculated field that computes on the local data in Tableau.
- They are calculated based on current visualization and do not consider any dimensions or measures that are filtered out of the visualization.
- These calculations are applied to the values of the entire table, not on some selected rows or columns.
- Table calculations are used for a variety of purposes, such as: Transforming values to rankings.
 - Transforming values to show running totals.
 - Transforming values to show the percent of the total.
- For any Tableau visualization, there is a virtual table which is determined by the dimensions in the view.
- This table is not the same as the tables in your data source.
- Mainly, the virtual table is determined by the dimensions within the "**level of detail**" means the dimensions on any of the following shelves in a Tableau worksheet:



- **For example**, for calculating an average, we need to apply a single method of calculations on an entire column. These calculations cannot be performed on some selected rows.
- The table has a feature known as "**Quick Table Calculations**", which is used to create such calculations.

Following are the steps applied in quick table calculations as:

Step 1: Select the **Measure** on which the table calculation has to be used and drag it to the column shelf.

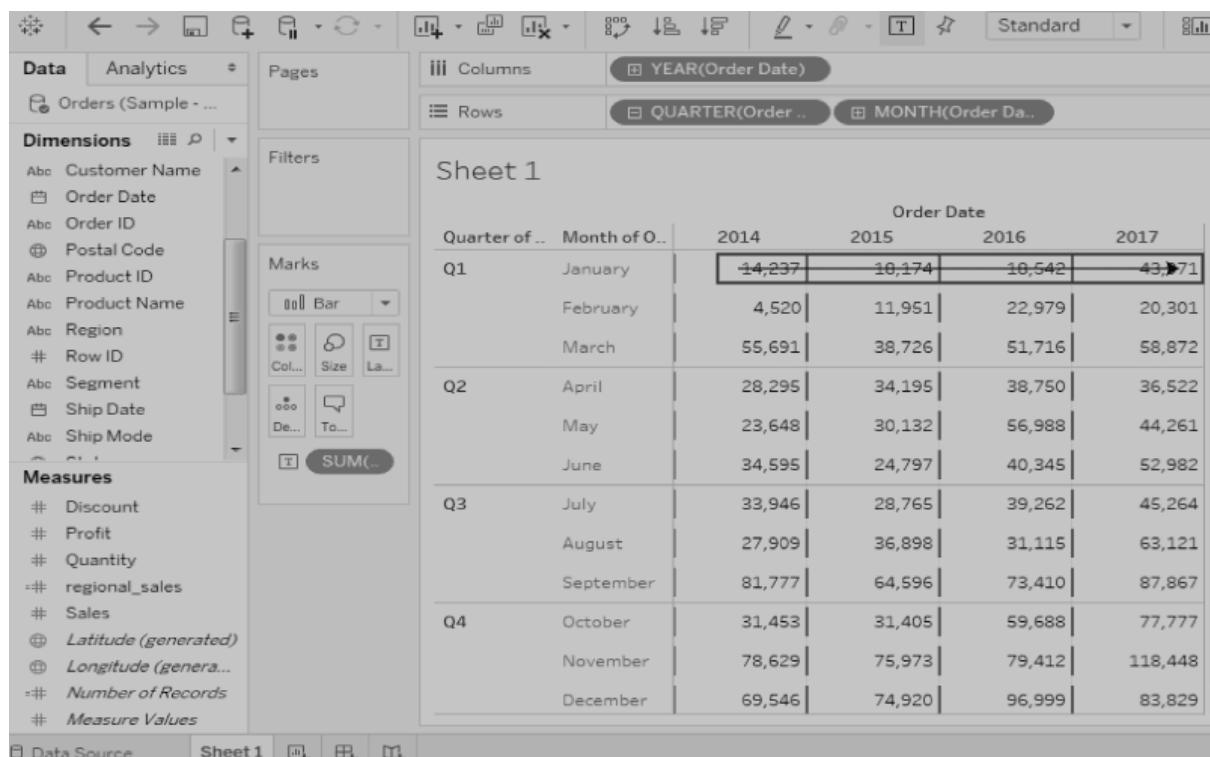
Step 2: Right-click on the **Measure** and choose the option **Quick Table Calculation**.

Step 3: Choose one option among the following options to be applied to the **Measure**.

- Running Total
- Difference
- Percent Difference
- Percent of Total
- Rank
- Percentile
- Moving Average
- Year to Date (YTD) Total
- Compound Growth Rate
- Year over Year Growth
- Year to Date (YTD) Growth

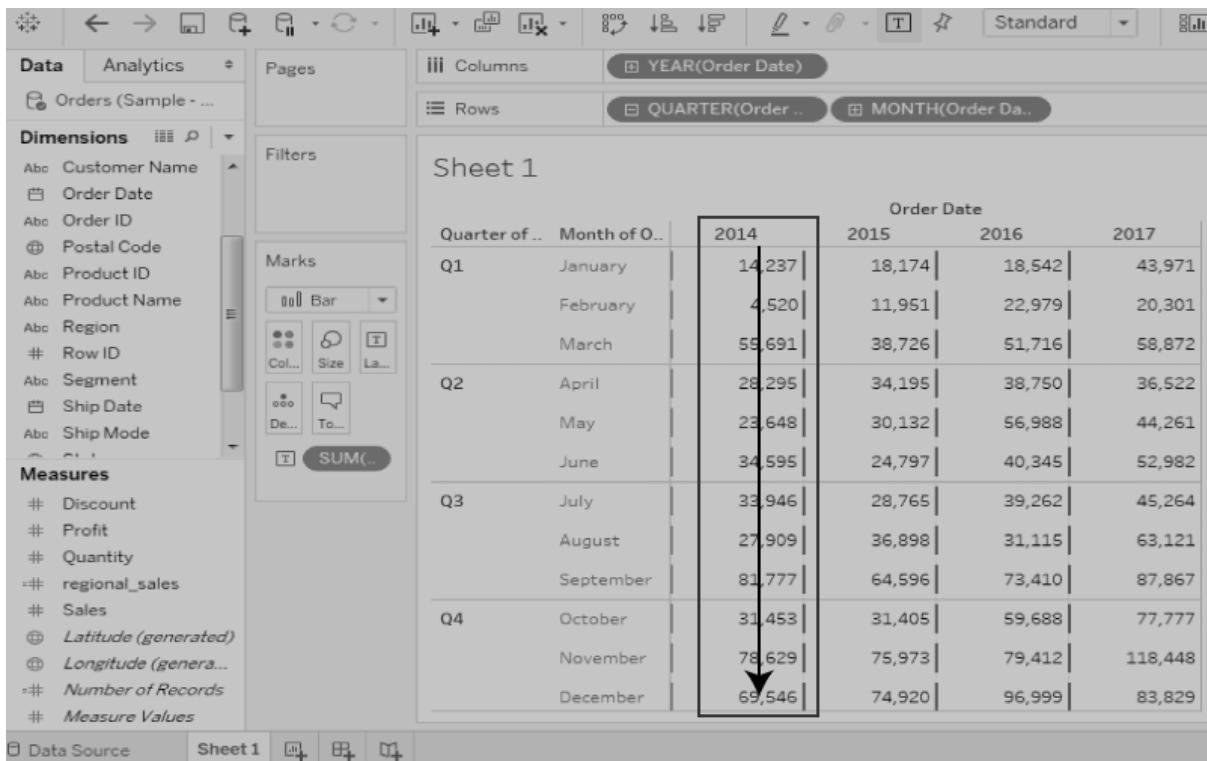
1. Table (Across): It computes across the length of the table and restarts after every partition.

For example, in the below screenshot, the calculation is computed across columns such as "Year (Order Date)" for every row such as "Month (Order Date)".



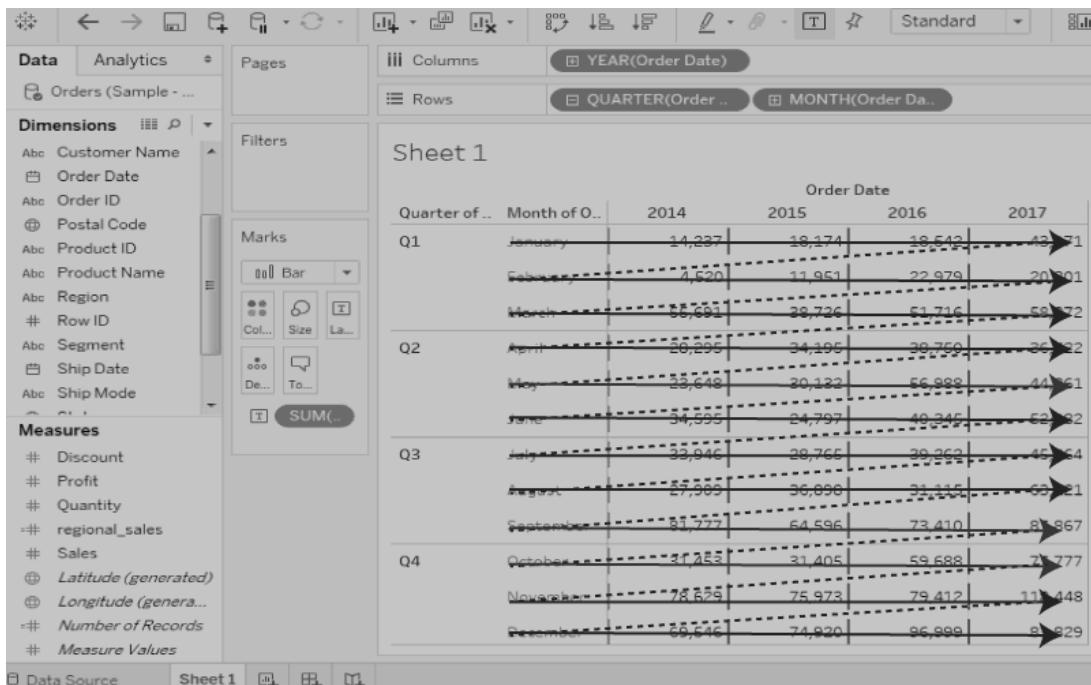
2. Table (Down): It computes down the length of the table and restarts after every partition.

For example, in the below screenshot, the calculation is computed down rows such as "Month (Order Date)" for every column such as "Year (Order Date)".



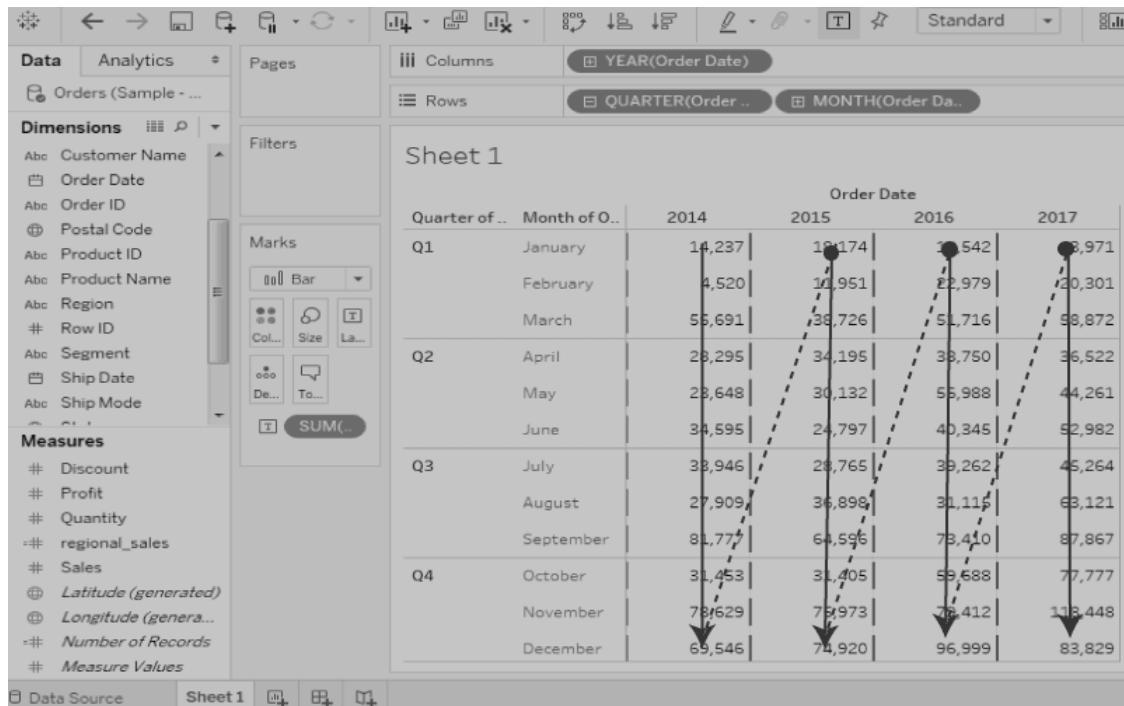
3. Table (Across then Down): It computes across the length of the table, and then down the length of the table.

For example, in the below screenshot, the calculation is computed across columns such as "Year (Order Date)", down a row such as "Month (Order Date)", and then across columns again for the entire table.



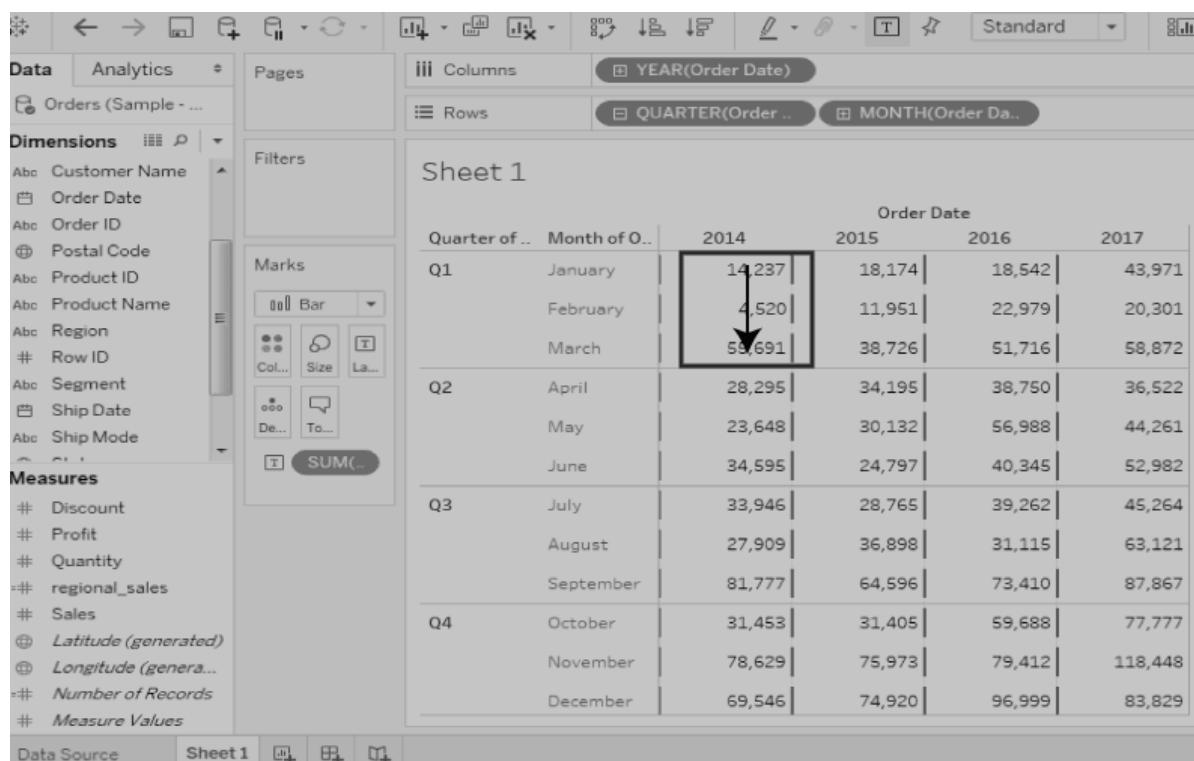
4. Table (Down then Across): It computes down the length of the table, and then across the length of the table.

For example, in the below screenshot, the calculation is computed down rows such as "Month (Order Date)", across a column such as "Year (Order Date)", and then down rows again.



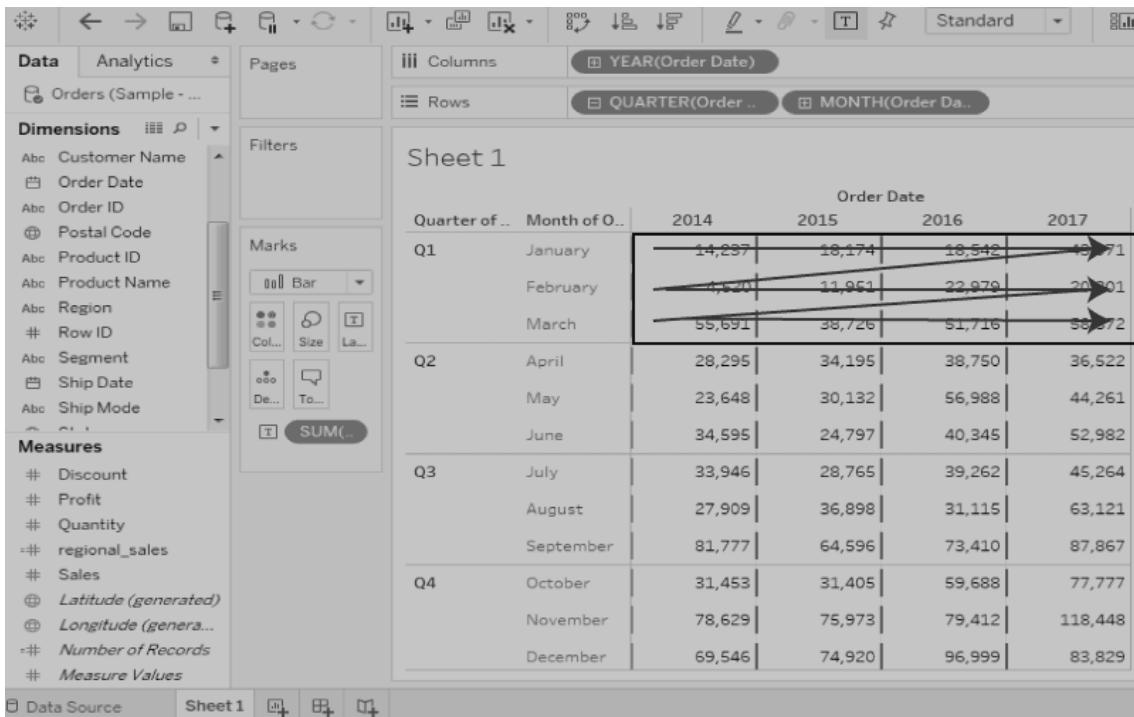
5. Pane (Down): It computes down an entire pane.

For example, in the below screenshot, the calculation is computed down rows such as "Month (Order Date)" for a single pane.



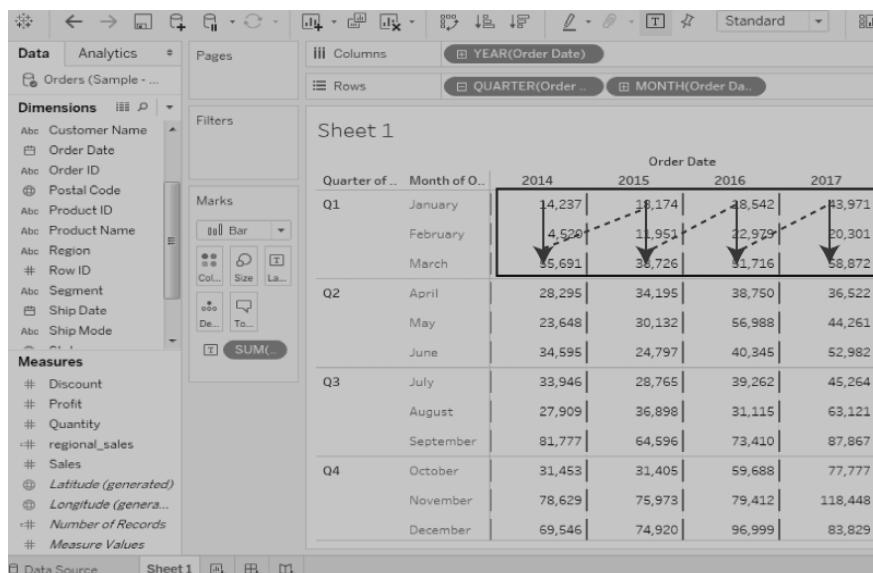
6. Pane (Across then Down): It computes across an entire pane and then down the pane.

For example: In the below screenshot, the calculation is computed across columns such as "Year (Order Date)" for the length of the pane, down a row such as "Month (Order Date)", and then across columns for the length of the pane again.



7. Pane (Down then Across): It computes down an entire pane and then across the pane.

For example, in the below screenshot, the calculation is computed down rows such as "Month (Order Date)" for the length of the pane, across a column such as "Year (Order Date)", and then down the length of the pane again.



4.11 Tableau LOD Expressions

LOD (level of Details) expression is used to run complex queries involving many dimensions at the data sources instead of bringing all the data to the Tableau interface.

Types of LOD expression

There are three types of LOD expressions in the Tableau:

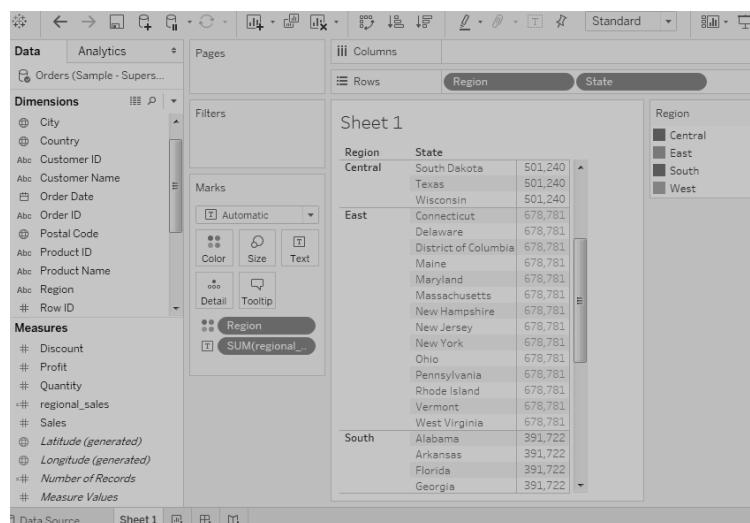
- **Fixed lod:** this lod expression computes the values using the specified dimensions without reference to any other dimensions in the view.
- **Include lod:** this lod expression computes the values using the specified dimensions with any other dimensions in the view.
- **Exclude lod:** these lod expressions subtract dimensions from the view level of detail.

For example, if you want to calculate the number of Sales for each state in each region. Then,

First, create the formula field named **regional_sales** using the formula as shown in the below screenshot.



- Drag the Region and State field to the Rows shelf and the calculated field (regional_sales) to the Text shelf under the Marks card.
- Also, drag the Region field to the Color shelf.
- This creates the below view, that shows a fixed value for different states because we fixed the dimension as a region for the calculation of Sales value.

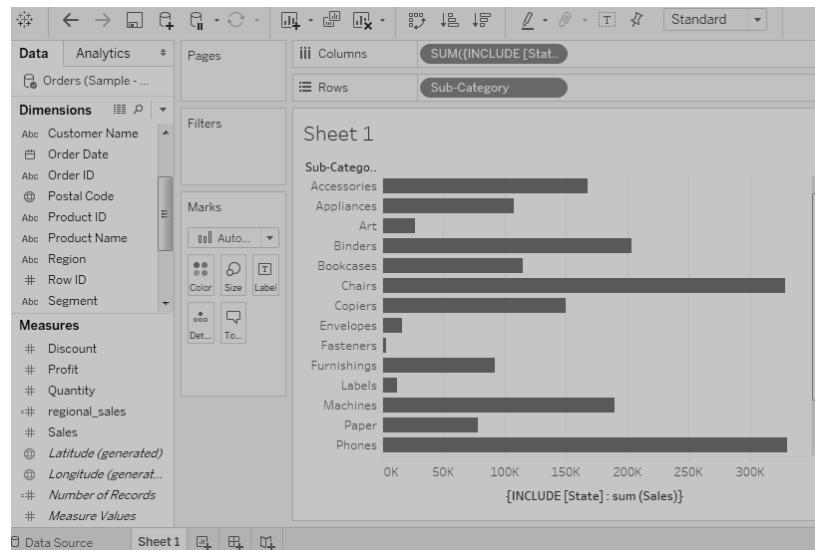


INCLUDE Level of Detail Expressions

INCLUDE level of detail expressions compute values using the specified dimensions whatever dimensions are in the view.

For example, if you want to calculate the sum of sales per state for each sub-category of products. Then,

- Drag the **Sub-Category** field to the Rows shelf.
- And, write the expression " **{INCLUDE [State] : SUM(Sales)}** " in the Columns shelf.
- It creates the view that includes both the dimensions in the calculations as shown in the below screenshot.

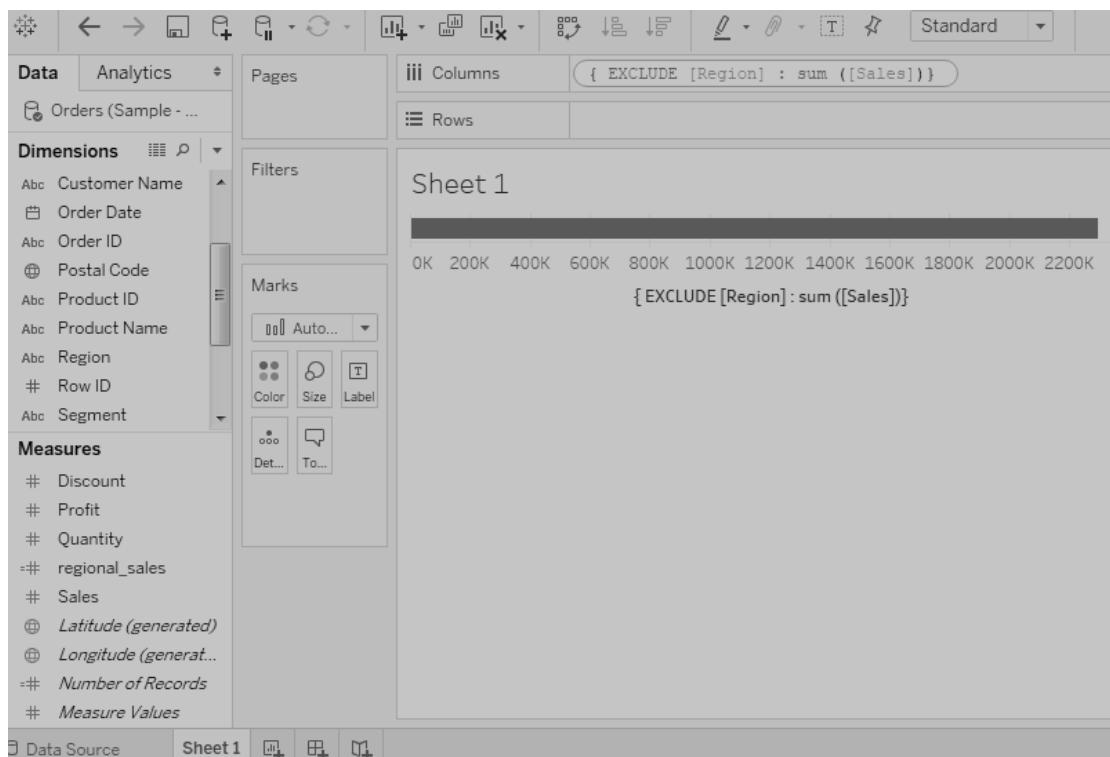


EXCLUDE Level of Detail Expressions

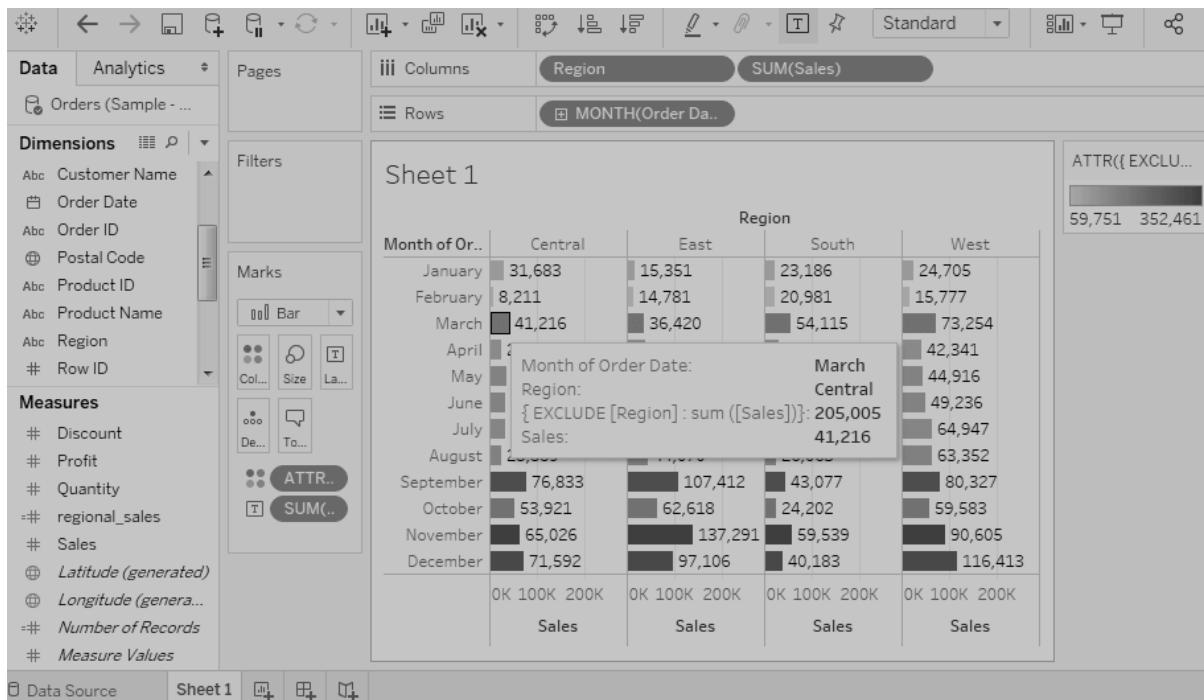
EXCLUDE level of detail expressions describe the dimensions to exclude from the view level of detail.

For example, Exclude **Region** from the **Sales** figure calculated for every month. First,

- Create the formula " **{EXCLUDE [Region] : sum([Sales])}** " as shown in the below screenshot.



- On dragging the relevant fields to the respective shelves, you get the final view for the EXCLUDE level of detail expressions as shown in the below screenshot.

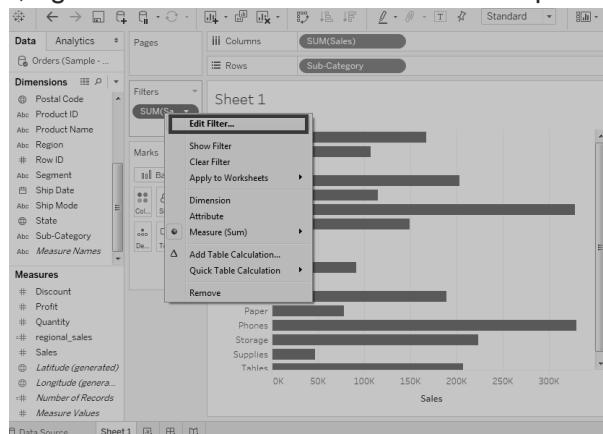


4.12 Tableau Filter Operations

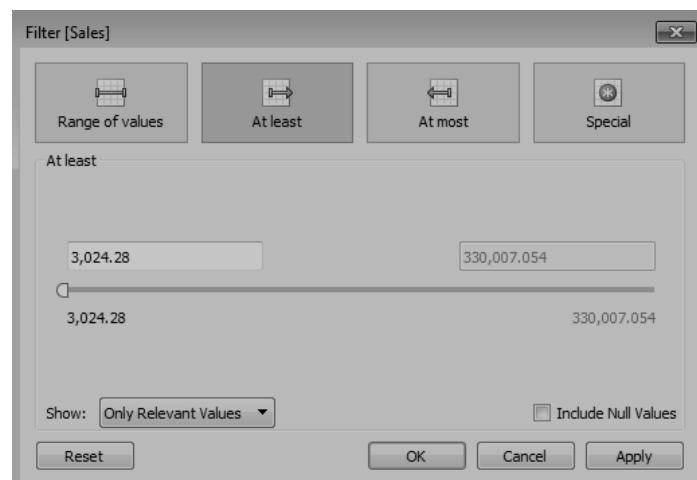
- Any data analysis and visualization work involve the use of extensive filtering of data.
- Tableau has a variety of filtrations to address these needs.
- Tableau has many inbuilt functions for applying filters on the data using both measures and dimensions.
- For the measures, the filter option offers numeric calculations.
- The filter option for dimension offers using a custom list of values or choosing string values from a menu.

Creating Filters

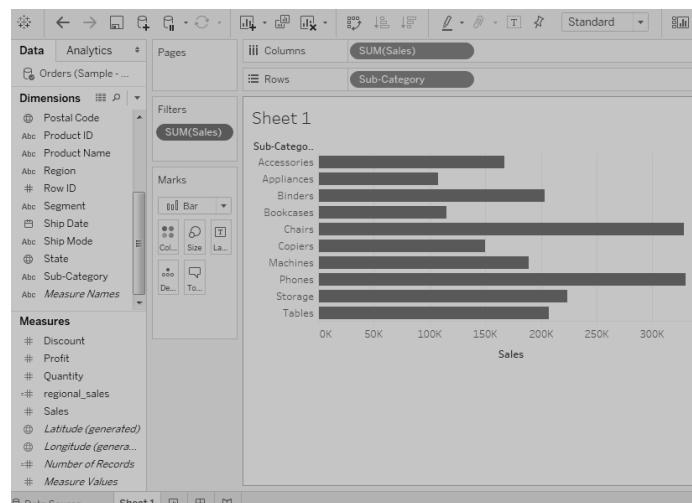
- Filters are designed by dragging the required field to the Filters shelf.
- Then, create a horizontal bar chart by dragging the dimension (**Sub-Category**) to the Rows shelf and the measure (**Sales**) to the Columns shelf.
- Again, drag the **Sales** into the Filters shelf, select **sum** option among all options, and click on the **Next** button.
- Once this filter is created, right-click and choose the **Edit Filter** option from the pop-up menu.



- Select one option among these options and click on **OK** button to apply the filter as shown in below screenshot.



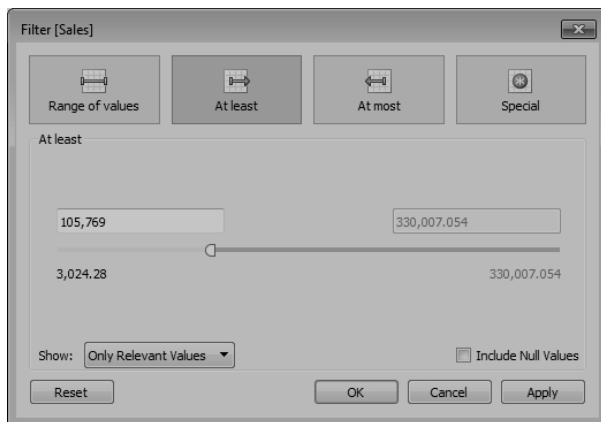
- The final view after applying filter looks like below screenshot:



Create Filters for Measures

- Measures are numeric fields. So, the filter options for such fields involve choosing values.
- There are following types of filters for measures in Tableau:
 - **Range of values:** It specifies the minimum and maximum values of the range to include in the view.
 - **At Least:** It includes all values that are greater than or equal to a specified minimum value.
 - **At Most:** It includes all values that are less than or equal to a specified maximum value.
 - **Special:** It helps you filter on Null values. It includes Null values, Non-null values, or All Values.

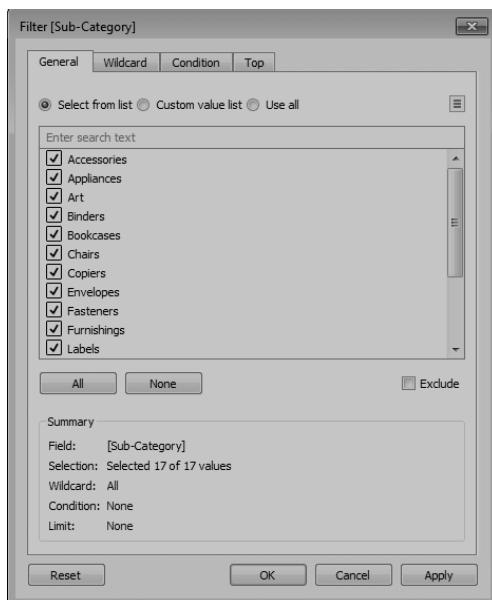
Below screenshot shows all these filters for Measures:



Create Filters for Dimensions

- Dimensions are descriptive fields having string values. There are following types of filters for dimensions in Tableau:
 - **General Filter:** It allows to select specific values from a list.
 - **Wildcard Filter:** It allows to mention wildcards like **cha*** to filter all string values starting with **cha**.
 - **Condition Filter:** It applies conditions such as sum of sales.
 - **Top Filter:** It chooses the records representing a range of high values.

Below screenshot shows all these filters for Dimensions:



How to Clear Filters

Filters can be easily removed after selecting the filter **Remove** options as shown in the below screenshot.

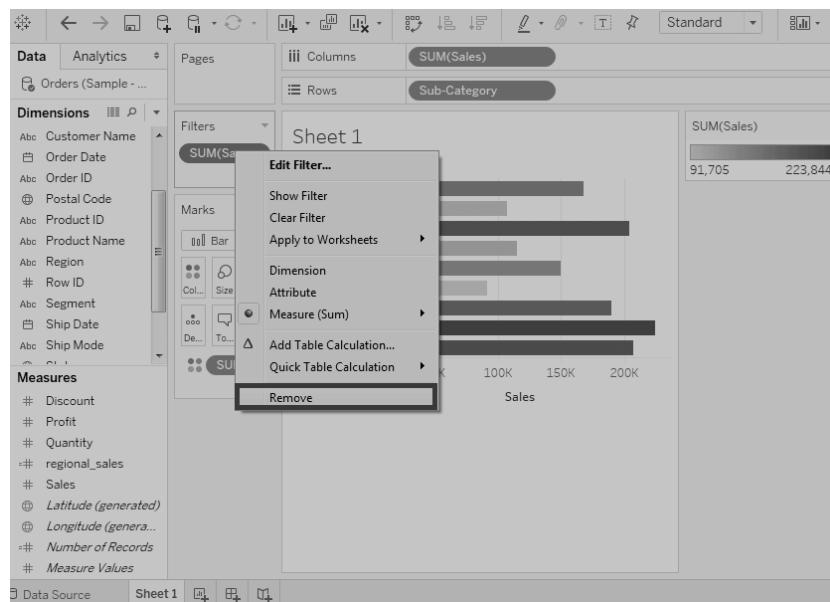
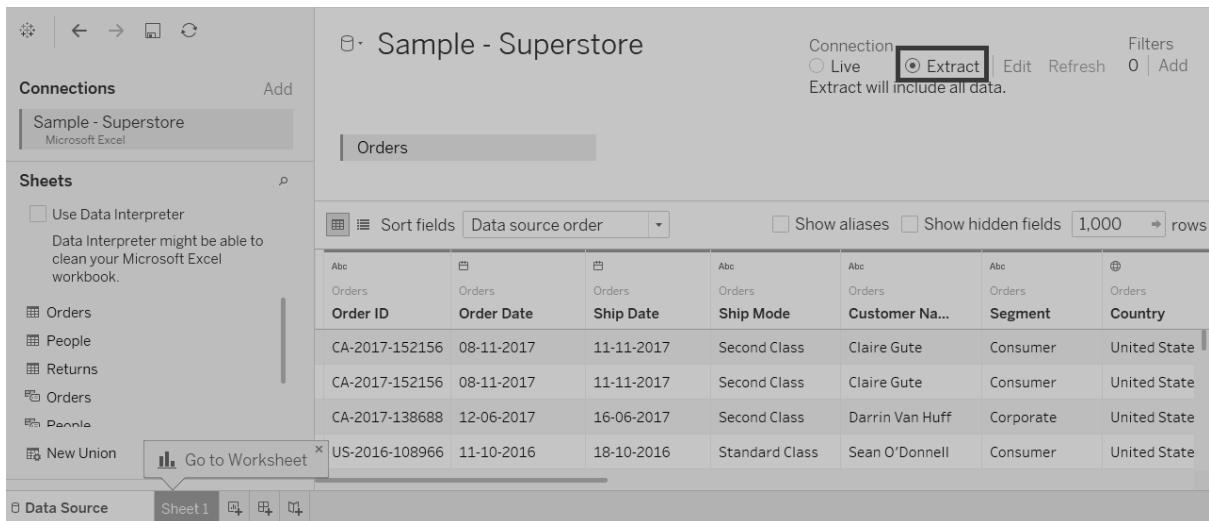


Tableau Extract Filters

- Extract filter is used to filter the extracted data from the data source.
- This filter is utilized if the user extracts the data from the data source.
- After connecting the text file to Tableau, you can see the two options, **Live** and **Extract** in the top right corner of the data source tab.
- A live connection is directly connected to a data source. And extract connection extracts the data from the data source and creates a local copy in Tableau repository. The procedure for creating an extracting filter is given below step by step as follows.

Step 1: Connect a text file with Tableau.

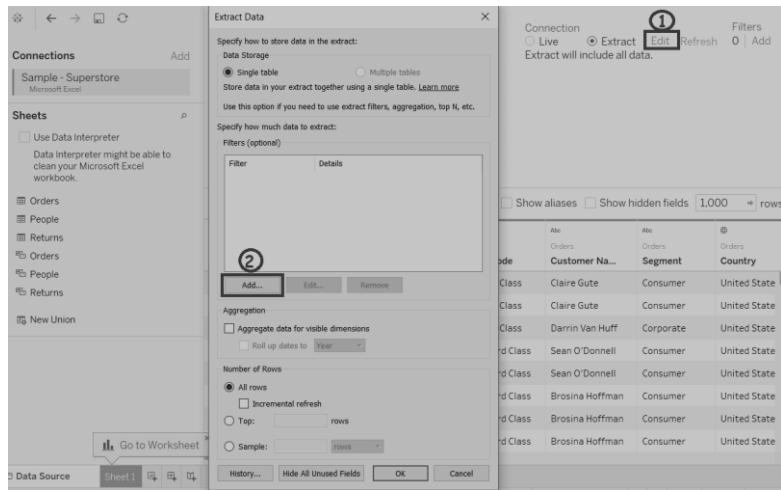
- Click on the "**Extract**" radio button as shown in below screenshot.



- It creates a local copy in Tableau repository.

Step 2: Then,

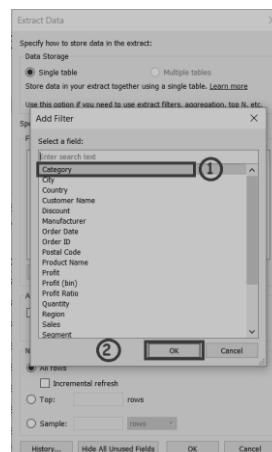
- Click on the "Edit" option that placed on the top right corner near to Extract button.
- It opens the "Extract data" window. Click on the "Add" option present in the Window.



Step 3: "Add Filter" Window is opened to select a filter condition.

You can choose any of the fields and add as **Extract** filter. In this example, we have selected "**Category**" as extract filter.

- Select the **Category** field from the list.
- Click on **OK** button.



After clicking on the **OK** button, it opens a filter window shown in the below screenshot.

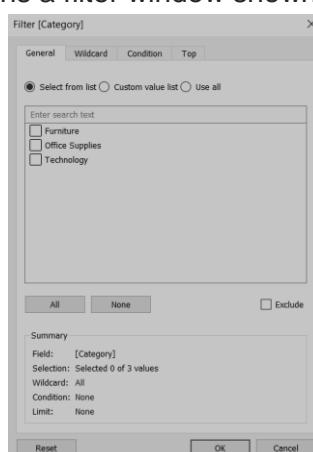
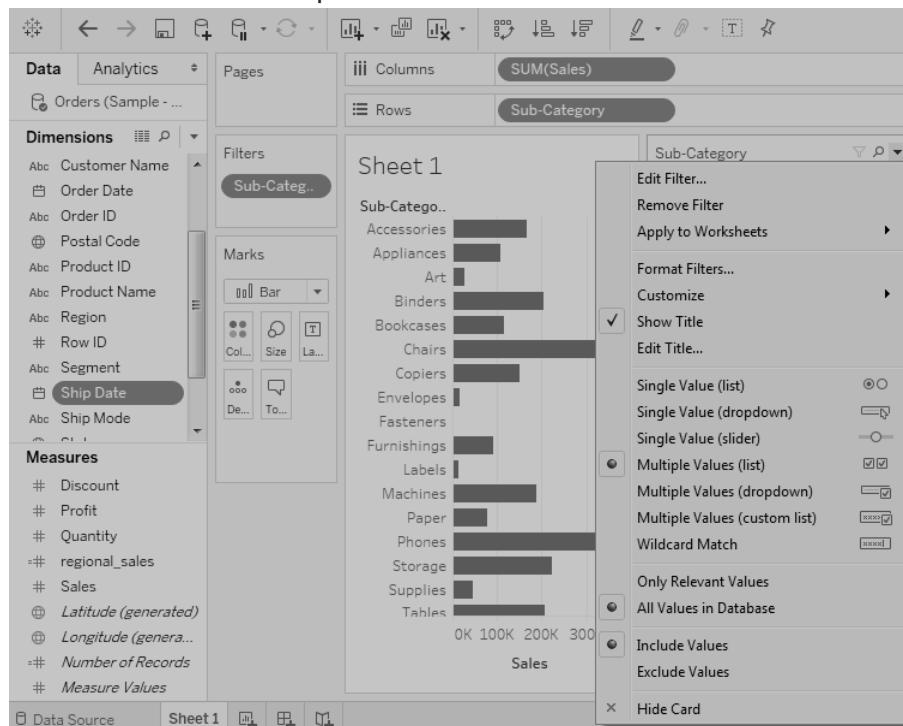


Tableau Quick Filters

- In Tableau, many filter types are quickly available using the right-click option on the measure and dimension.
- These filters have enough functionality to solve most of the everyday filtering needs. These filters are known as Quick filters.

The below screenshot shows how the quick filters are accessed:

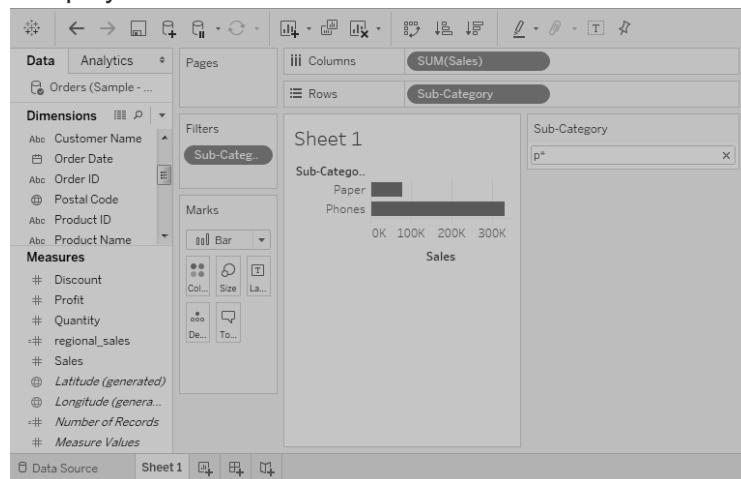


The given below table lists the various quick filters and their uses in Tableau.

Filter Name	Purpose
Single Value (List)	It selects only one value at a time in the list.
Single Value (Dropdown)	It selects a single value in a drop-down list.
Multiple Values (List)	It can select one or more values in a list.
Multiple Values (Dropdown)	It selects one or more values in the drop-down list.
Multiple Values (Custom List)	It selects and searches for one or more values.
Single Value (Slider)	It drags a horizontal slider for selecting a single value.
Wildcard Match	It selects values containing the specified characters.

For example, consider a data source such as **Sample-Superstore**, to apply some quick filters. First,

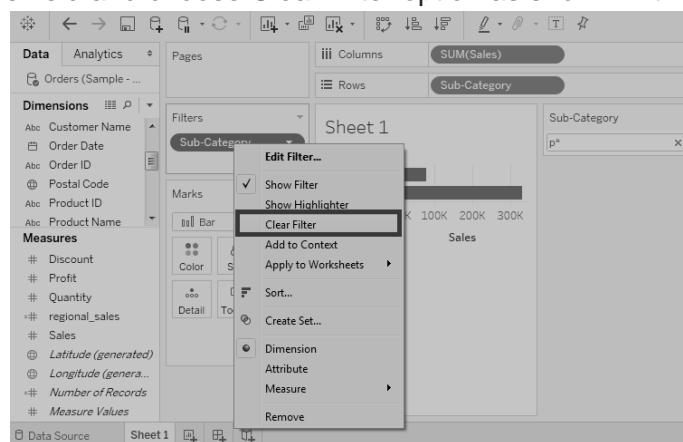
- Choose the sub-category field as the row shelf and sales as the column shelf that produces a horizontal bar chart.
- Drag the sub-category field to the filters pane. Apply wildcard filtering using the expression **p*** to select all subcategory names starting with "p".
- The below screenshot shows the result after applying this filter where only the sub-categories starting with "p" are displayed:



How to Clear the Filter

After the analysis is completed by applying the filter, you can remove it by using the clear filter option.

- First, go to the Filter Pane.
- And, right-click on the field and choose Clear Filter option as shown in the below screenshot.



After clearing filter from the filter pane, the worksheet looks like the below screenshot:

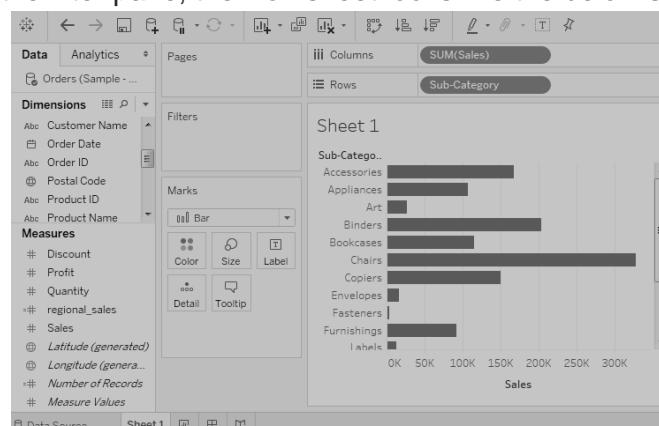


Tableau Context Filters

- All filters that you arrange in Tableau are computed independently.
- Each filter accesses all rows in your data source without view to other filters.
- You can arrange one or more categorical filters as context filters for the view.
- Context filter can work as an independent filter.
- Any other filters that you arrange are defined as dependent filters because they process only the data that passes through the context filter.

Context filter is created because of the following reasons:

- **Improve Performance:** If you want to set a lot of filters or have a significant data source, then queries start running slowly. In such case, you can set one or more context filters to improve performance.
- **Create a Dependent Numerical or Top N Filter:** You can set a context filter to include the data of interest only, and arrange a numerical or a top N filter.

Create a Context Filter

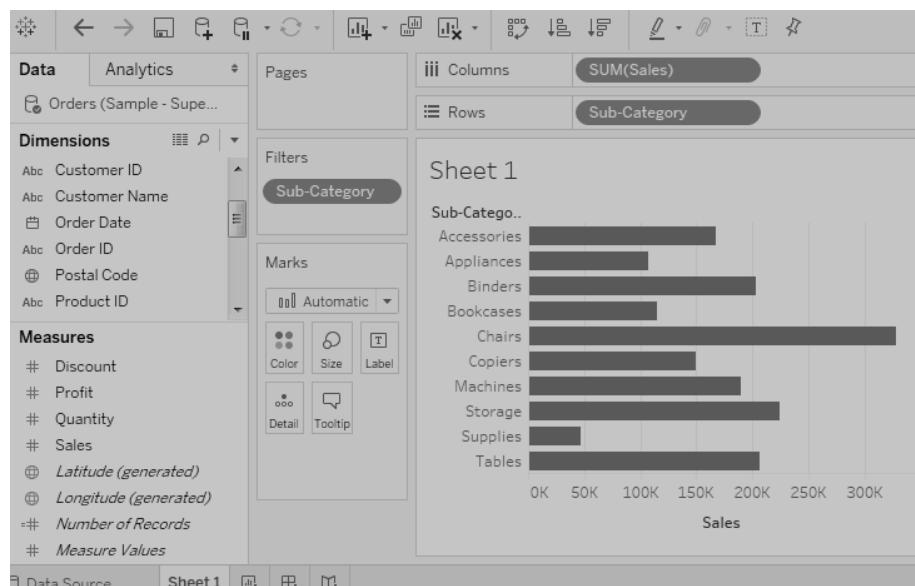
- To create a context filter, first select **Add to Context** from the context menu of an existing categorical filter.
- The context is computed once the view is generated.
- All other filters are then calculated relative to the context. Context filters are:
 - Appeared on the top of the filters pane.
 - Identified by the gray color on the filters pane.
 - Not rearranged on the filters pane.

For example, consider the data source such as **Sample-superstore**, find the top 10 Subcategory of products for the category called **Furniture**. There are the following steps:

Step 1: Drag the **Sub-Category** field to the Rows shelf and Sales field to the Columns Shelf.

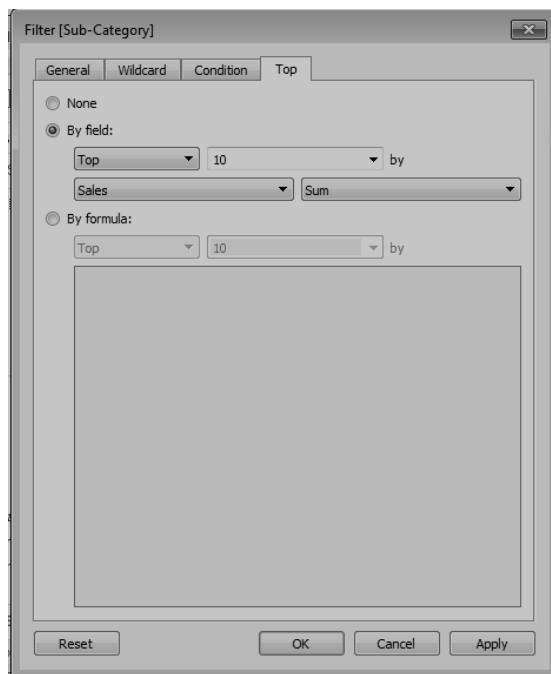
Step 2: And, choose the horizontal bar chart from the "Show Me" tab.

Step 3: Again, drag the **Sub-Category** to the Filters shelf. You get the chart shown in the below screenshot.

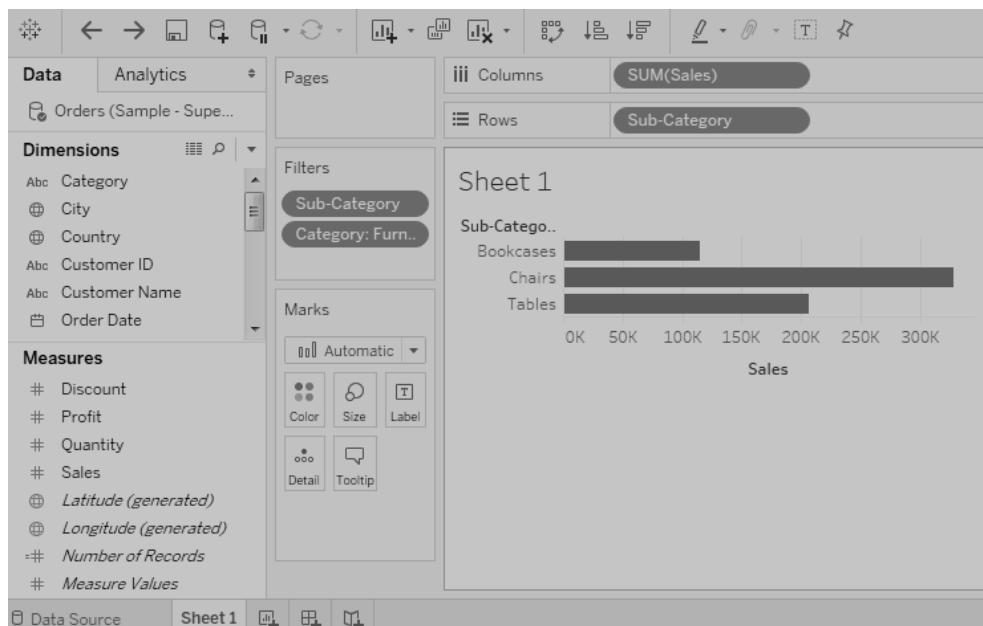


Step 4: Right-click on the **Sub-Category** field in the filter shelf and click on "Edit Filter" option then go the "Top" tab in the pop-up window.

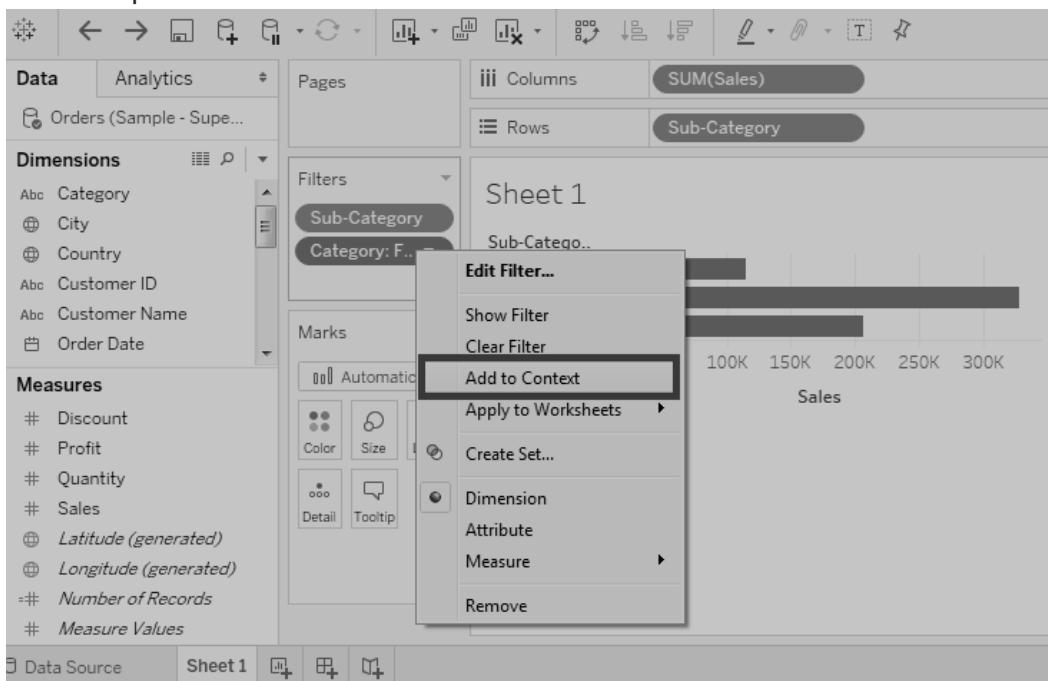
Step 5: And, choose the "By field" option. From the next drop-down, choose the option Top 10 by Sales Sum as shown in the below screenshot.



Step 6: Drag the **Category** field to the filter shelf. Right-click on the Category field to edit and choose **Furniture** from the list. It shows three subcategories of products as a result shown in below screenshot.



Step 7: Now, adding the context filter, Right-click on the **Category: Furniture** filter and select the "Add to Context" option.



Step 8: Above all steps produce the final result that shows the subcategory of products from the category Furniture.

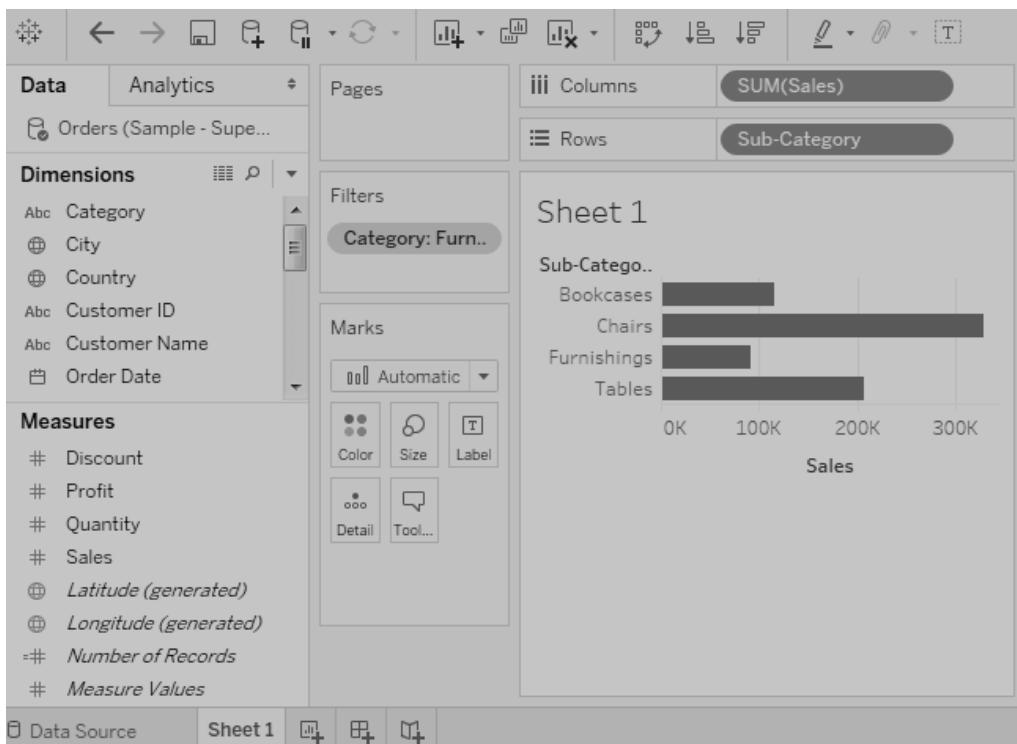


Tableau Condition Filters

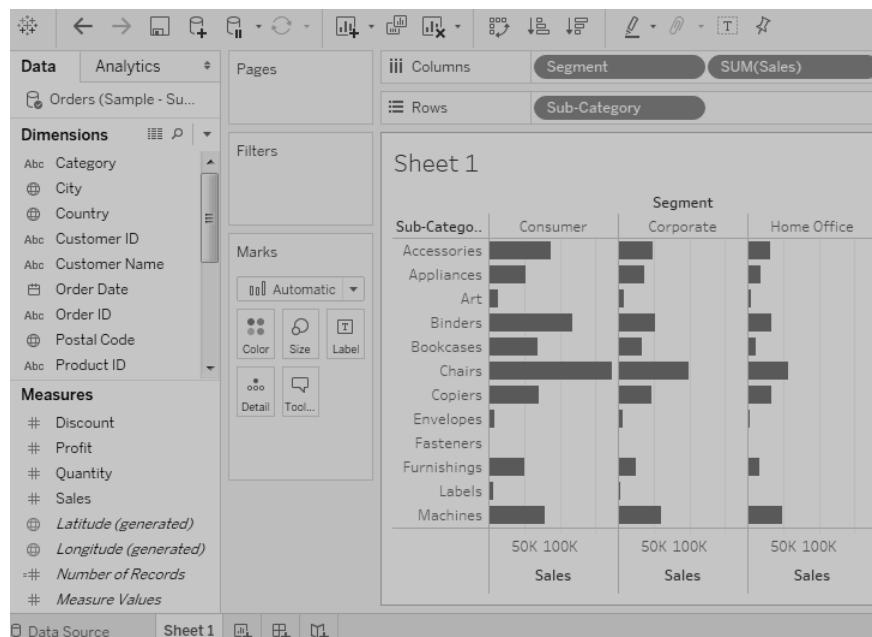
- In Tableau, condition filter is used to apply some conditions to already existing filters.
- These conditions are very simple, for example, finding only those sales which are higher than a certain amount. Also, these conditions can be applied to create a range filter.

Create a Condition Filter

For example, consider the data source such as **Sample-superstore**, let's find the **sub-category** of products across all **Segments** whose sales exceed two million. There are some steps for creating condition filter in Tableau.

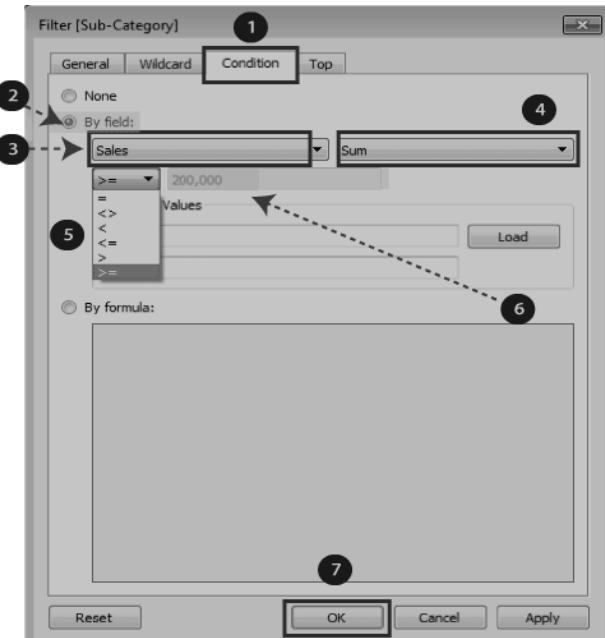
Step 1: Drag the **Segment** field and the **Sales** field to the Column shelf.

Step 2: Next, drag the **Sub-Category** field to the Rows shelf. Choose the horizontal bar chart option. And you get view shown in below screenshot.



Step 3: Again, drag the **Sub-Category** field to the Filters Shelf.

Step 4: Right-click on Sub-Category field to edit and then go to the "Condition" tab. And, choose the radio "By field" option. From the drop-down, select Sales, Sum, and greater than equal to symbol specifying the value 200000.



- After completing the above steps, you get a view which shows only those subcategories of products that have the required amount of sale.
- Also, this shows all the available **Segments** where the condition is **True** that shown in below screenshot.

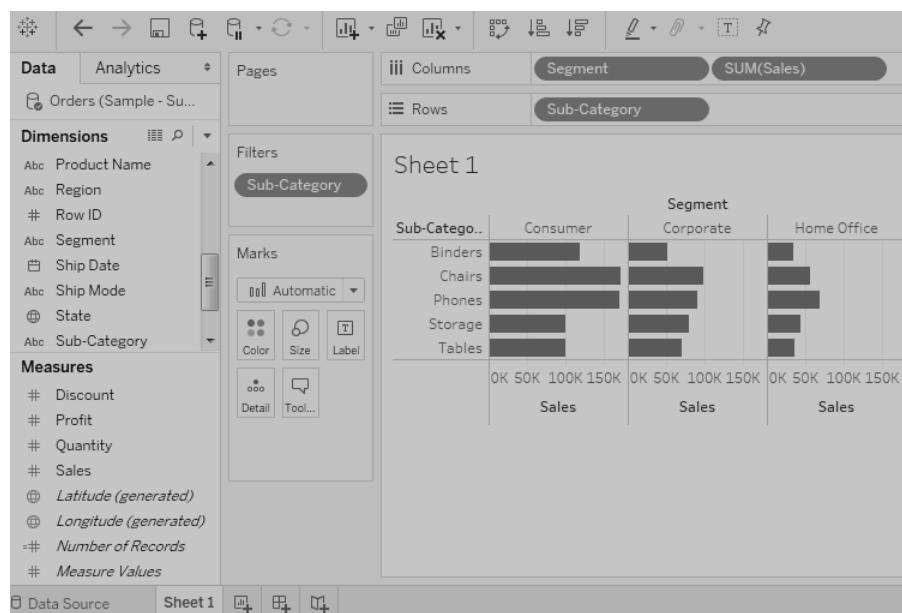


Tableau Data Source Filters

- The data source filter is used to filter the data in data source proportion.
- It restricts the files present in the data set. This filter is similar to the extract filter in securing the data. But data source filter and extract filter both are different, and they are not linked to each other.
- A data source filter works on both **Live** connection and **Extract** connection.

The procedure to select a data source filter is given as step by step below.

Step 1: Click on the "Add" button placed at the top right corner of the data source tab, shown in the following screenshot.

This screenshot shows the Tableau Data Source tab. On the left, there's a sidebar with 'Connections' (Sample - Superstore) and 'Sheets' (Orders, People, Returns, New Union). Below the sidebar is a 'Go to Worksheet' button. The main area displays the 'Orders' sheet from the 'Sample - Superstore' connection. At the top right of the main area, there are buttons for 'Connection' (radio buttons for 'Live' and 'Extract', with 'Extract' selected), 'Edit', 'Refresh', 'Filters' (set to 0), and an 'Add' button which is highlighted with a red box. Below these buttons, it says 'Extract will include all data.' The data table shows columns: Order ID, Order Date, Ship Date, Ship Mode, Customer Name, Segment, and Country. The data includes rows like CA-2017-152156, 08-11-2017, 11-11-2017, Second Class, Claire Gute, Consumer, United States.

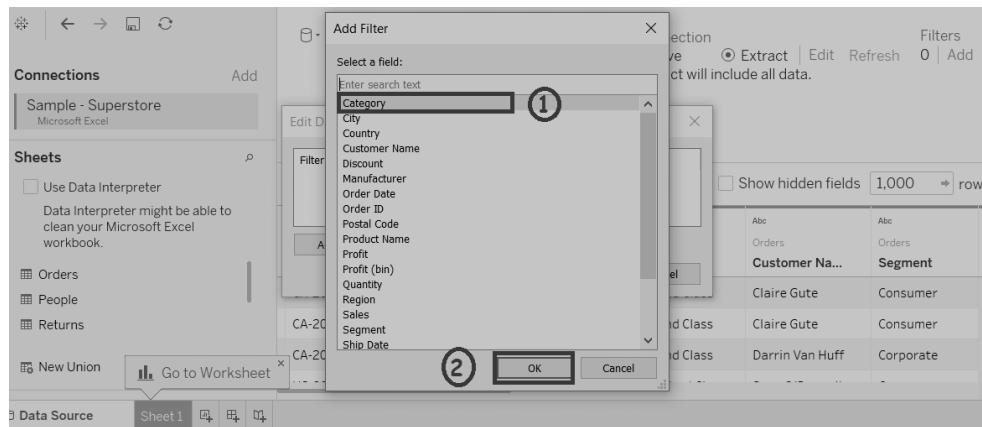
Step 2: It opens the "Edit Data Source Filters" Window. Then, click on "Add" Option of the window that shown in below screenshot.

This screenshot shows the 'Edit Data Source Filters' dialog box overlaid on the Tableau interface. The dialog has tabs for 'Filter' and 'Details'. At the bottom of the dialog, there are 'Add...', 'Edit...', and 'Remove' buttons, with the 'Add...' button highlighted with a red box. Below the dialog, the 'Orders' sheet is visible, showing the same data as the previous screenshot. The 'Edit Data Source Filters' dialog also has 'OK' and 'Cancel' buttons.

Step 3: "Add Filter" Window is opened to select the filter conditions.

Select any of the fields and add as extract filter. For example, you want to select the **Category** field as an extract filter. Then,

1. Select **Category** from the list.
2. Click on **OK** button, shows in below screenshot.



After clicking on **OK** button, it opens a filter window shown in below screenshot.

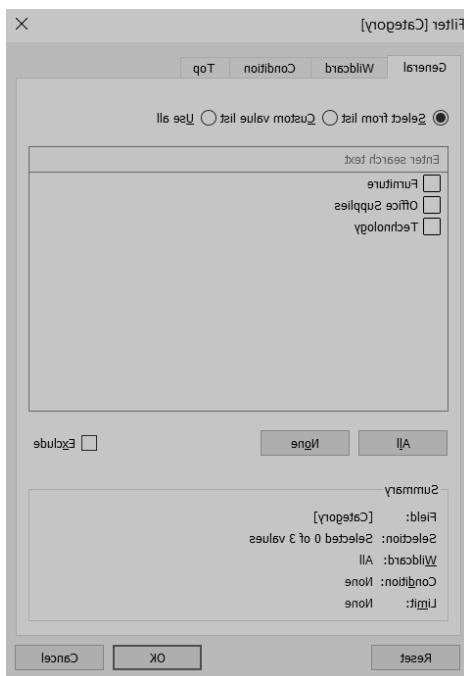


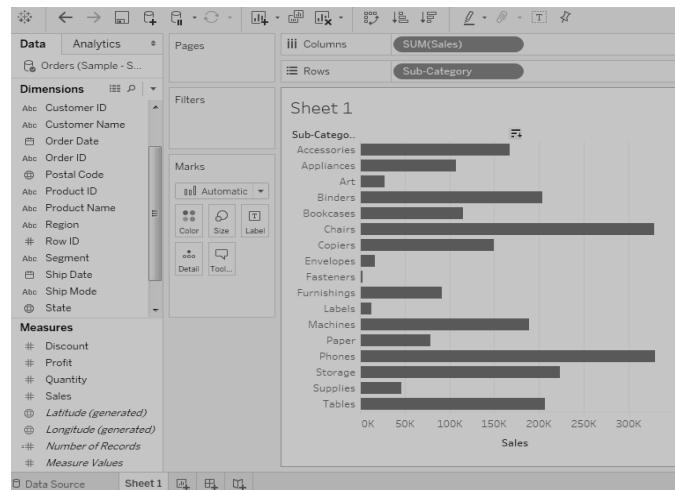
Tableau Top Filters

- In Tableau, Top filter is used to set the limit of result from a screen. For example, if you want to get only the top 10 values from a large set of records.
- Then, you can apply this filter using the inbuilt options for limiting the files in many ways or by creating a formula.

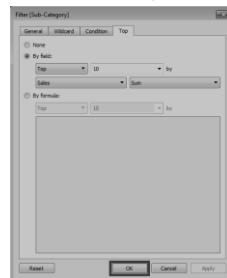
Create a Top Filter

For example, consider the data source such as **Sample-superstore**, and you want to find the sub-category of products that represents the top 10 sales amount. There are the following steps, such as:

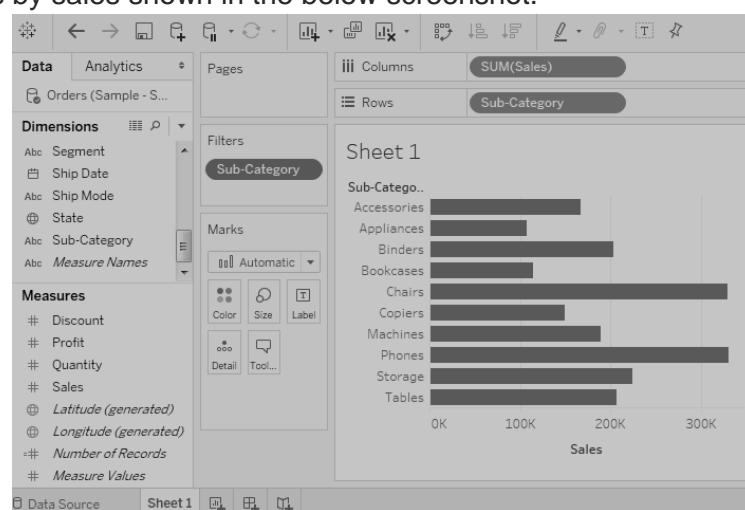
Step 1: Drag the **Sub-Category** field to the Rows shelf and the **Sales** field to the Columns shelf. Choose the horizontal bar from the "Show Me" tab. Tableau shows the following view:



Step 2: Right-click on the **Sub-Category** field and go to the "Top" tab. And, choose the second radio "By field" option. From the drop-down, select the Top 10 options by Sum of Sales.



After completing the above all steps, you will get the following view, which shows the top 10 Sub-Category of products by sales shown in the below screenshot.



4.13 Tableau Sort Data

- Data present in the worksheet can be sorted based on the requirement.
- It can sort the data based on the data source such as ascending, descending order, or depend on any measured value.
- The procedure for sorting the data is given below, step by step:

For example, consider a data source such as **sample-superstore**, and you want to sort the dimensions and the measures fields as follows.

Step 1: Add the **sample-superstore** data source with Tableau and drag the **Order** table to the pane shown in the below screenshot.

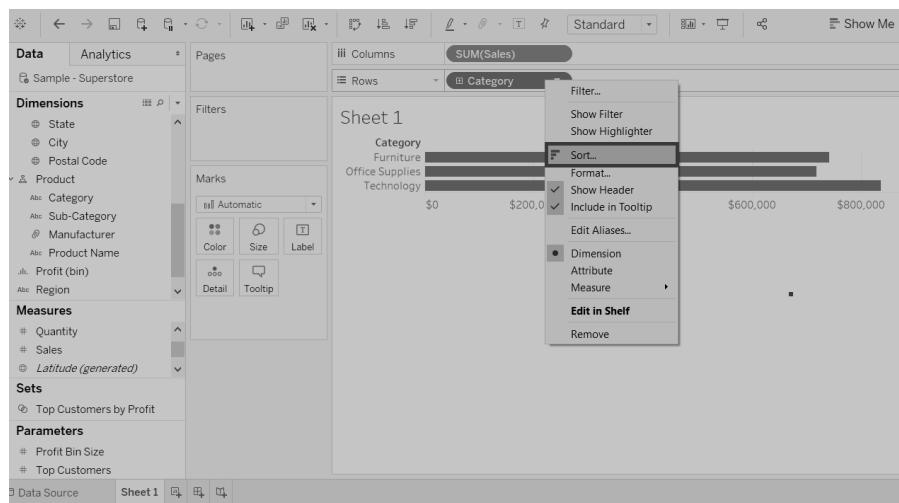
The screenshot shows the Tableau Data Source pane. On the left, under 'Connections', there is a single connection named 'Sample - Superstore (Microsoft Excel)'. Below it, the 'Sheets' section lists several sheets: 'Orders' (which is selected and highlighted in blue), 'People', 'Returns', 'Orders', 'People', 'Returns', and 'New Union'. A large arrow points from the 'Orders' sheet name towards the main data preview area. The main area displays the 'Orders' table with columns: Order ID, Order Date, Ship Date, Ship Mode, Customer Name, Segment, and Country. The data shows several rows of order information. At the bottom of the pane, there are buttons for 'Data Source' and 'Sheet 1'.

Step 2: Go to the worksheet and drag the dimension **Category** to the row shelf and the measure **Sales** to the column shelf.

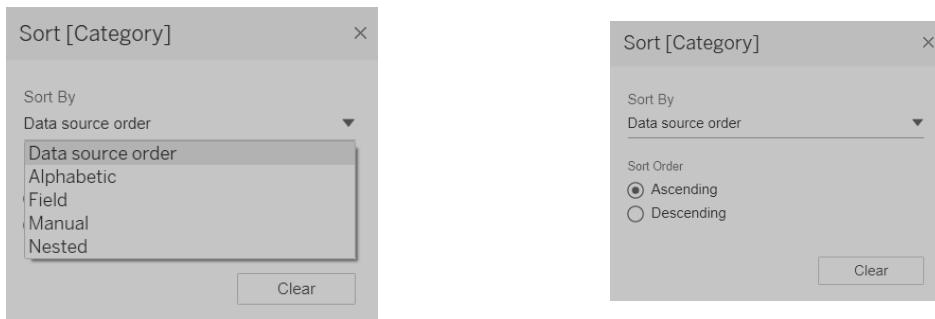
The screenshot shows a Tableau worksheet titled 'Sheet 1'. In the top-left corner, the 'Data' shelf has 'Sample - Superstore' selected. The 'Dimensions' shelf includes 'Category' (which is selected and highlighted in blue) and 'Region'. The 'Measures' shelf includes 'Sales' (which is selected and highlighted in blue). The 'Filters' shelf is empty. The 'Marks' shelf shows 'Automatic' selected. The main area displays a horizontal bar chart with three bars representing different categories: Furniture, Office Supplies, and Technology. The x-axis is labeled 'Sales' and ranges from \$0 to \$800,000. The chart title is 'Sheet 1'.

It creates a horizontal bar chart. **Category** field present in the visual order, and it is sorted based on data source by default. We can change the order of sorting by following the below procedure.

Step 3: Right-click on the **Category** field and select **Sort** option.



After that, it opens the Sort window. All options present inside the sort window is shown below as follows:



Sort Order

- **Ascending:** It sorts the order of selected dimensions and measures in ascending order.
- **Descending:** It sorts the order of selected dimensions and measures in descending order.

Sort By

The field can be sorted in different types of methods that are explained below as follows.

- **Data source order:** It sorts the field based on data source order.
- **Alphabetic:** It sorts the dimensions and measures in alphabetical order.
- **Field:** It sorts the field based on the other measure or dimension values.
- **Manual:** It can manually sort the data.

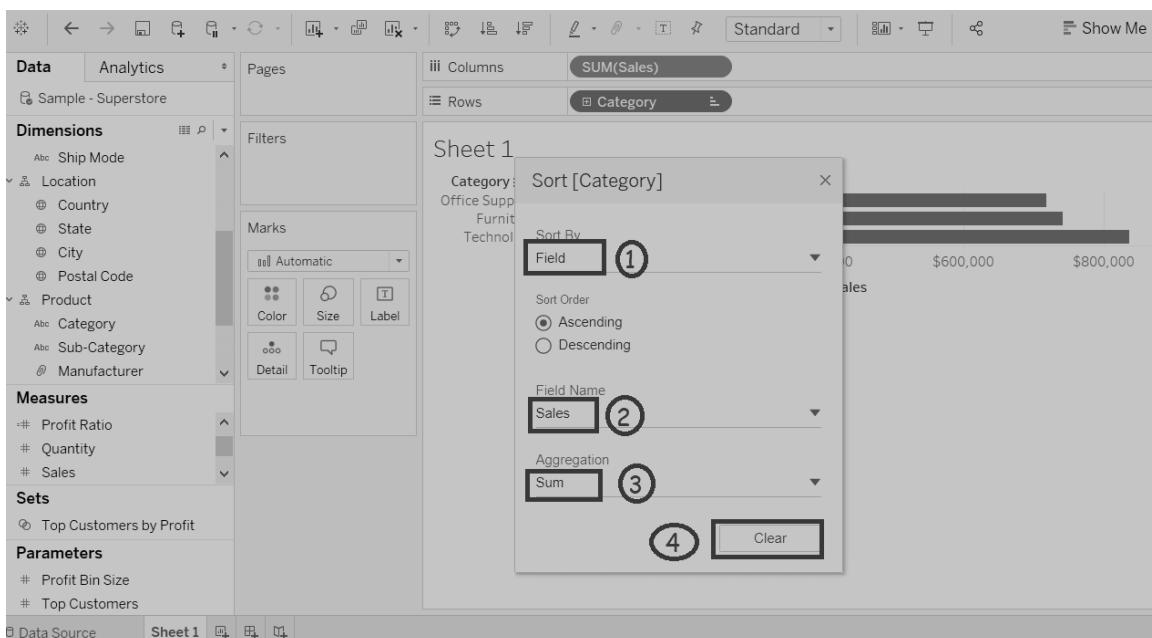
For example, suppose the **Category** field is sorted based on another field such as '**Sales**'.

Step 1: Click on '**Field**' radio button.

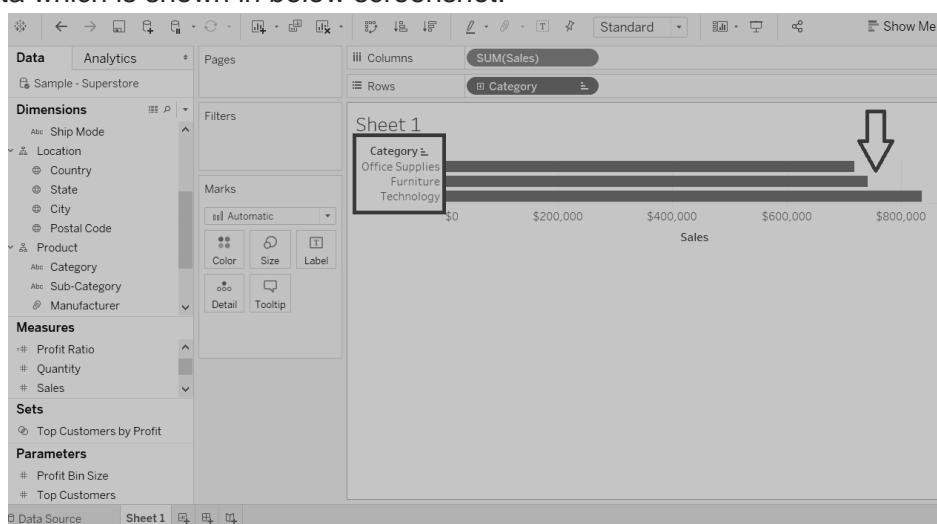
Step 2: Select the field on which the **Category** is to be filtered.

Step 3: Select the aggregation type.

Step 4: Click on **Clear** button.



In the above example, it filters the **Category** field based on the sum of sales in ascending order. And it sorts the data which is shown in below screenshot.



4.14 Tableau Groups, Hierarchy and Sets

Tableau Groups

- It creates a group to combine related members in the field.
- If you are working with a view and you want to group specific fields to create significant categories.

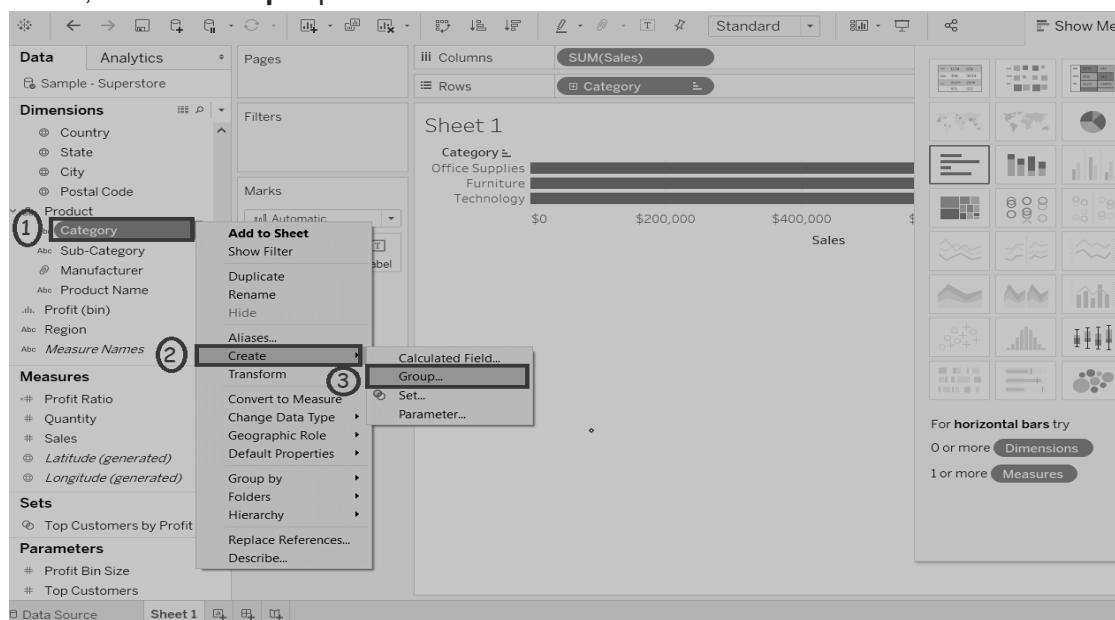
For example, consider the data source such as **sample-superstore**, then drag the **Sales** field in column shelf and **Category** field in row shelf and then sort them in ascending order (discussed in Tableau sort data).

- The aggregated values of Furniture and Office Supplies can be obtained by using the group.
- Once the group is built, the aggregated value of Furniture and Office Supplies can be shown in the visuals. The procedure to create a group is given below step by steps as follows.

Step 1: Right click on the **Category** field.

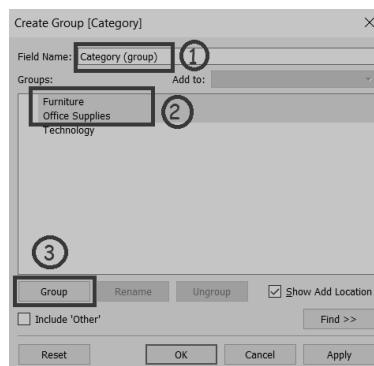
Step 2: Click on the "Create" option.

Step 3: Then, select "Group" option shown in below screenshot.



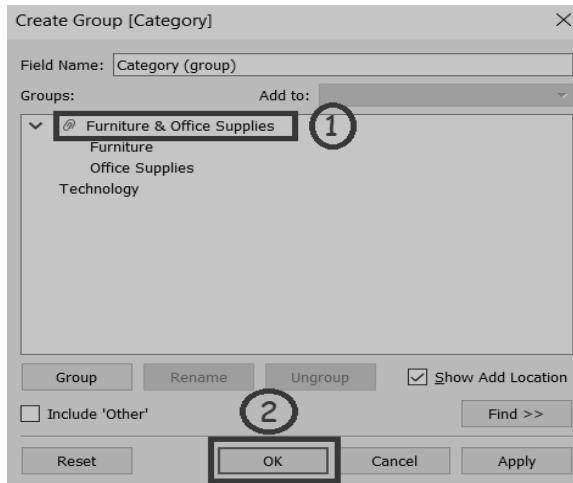
Step 4: It opens the "Create Group" window. Then,

1. Write the name of the group.
2. Select the members which you want to be grouped.
3. Click on the "Group" button.



Step 5: In "Edit Group" window,

1. It creates a group of **Furniture and Office Supplies**.
2. Then, click on the **OK** button to create the group.



- It created a group whose field name is **Category (Group)** and added in the dimension list. This is used for visualizing the group of members present in a field.
- The below screenshot explains the functionality. The sum of sales is visualized for both **Furniture** and **Office Supplies**.

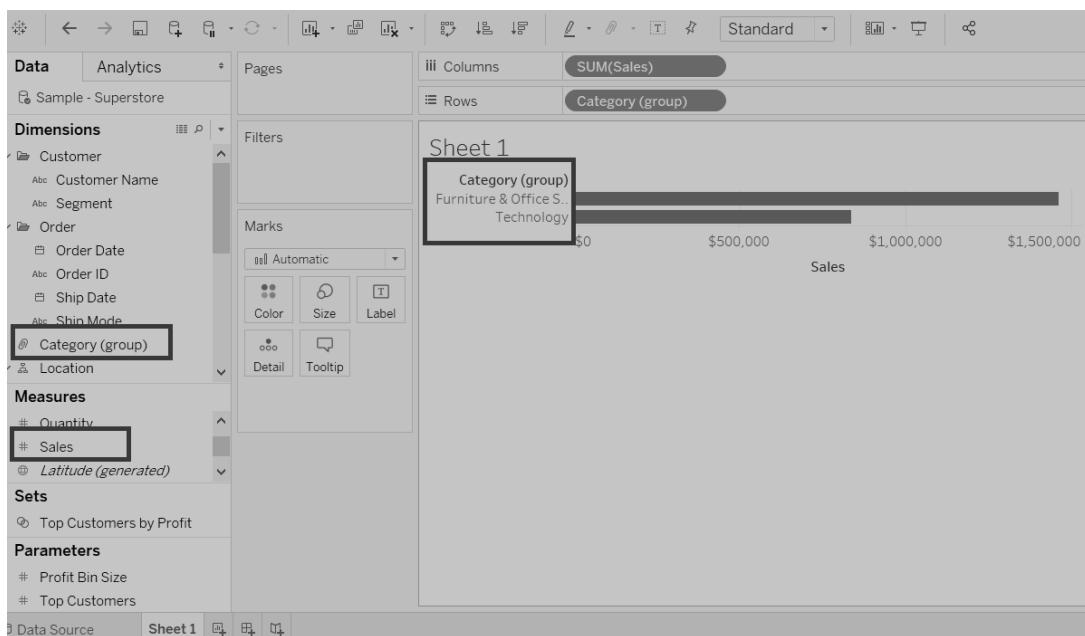


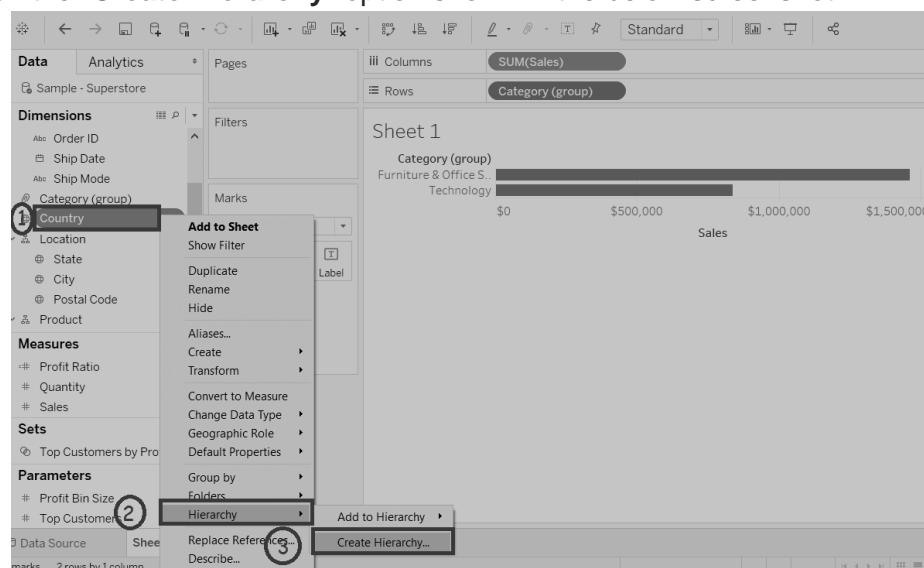
Tableau Hierarchy

In Tableau, Hierarchies can be built to visualize the data. It can be created in the Tableau by following the below steps:

For example, consider the data source such as **Sample-Superstore** and its dimensions and measures.

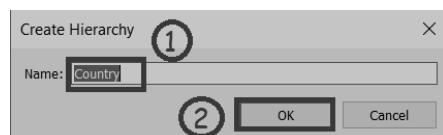
Step 1: First go to the worksheet. Then,

1. Select a dimension and right-click on that dimension to create a hierarchy.
2. Go to "Hierarchy" option.
3. And, click on the "**Create Hierarchy**" option shown in the below screenshot.

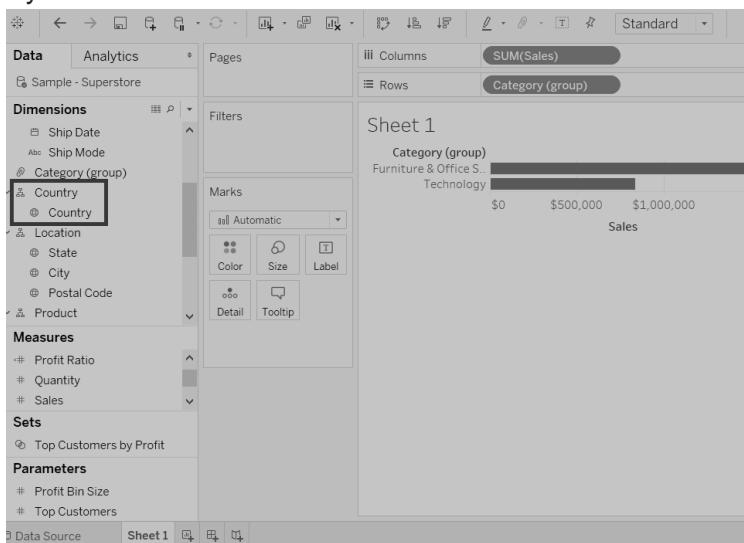


Step 2: It opens the "**Create Hierarchy?**" window. Then,

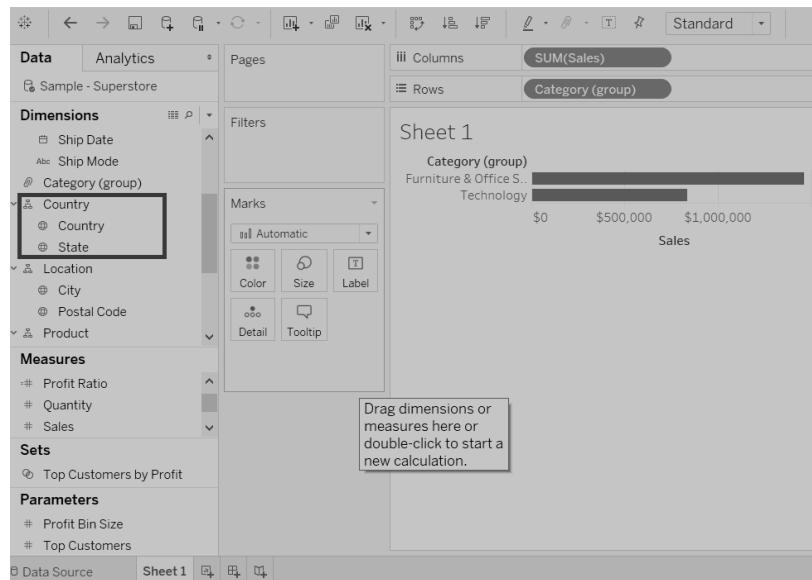
1. Enter a name of hierarchy.
2. And click on the **OK** button.



- It creates a hierarchy shown in below screenshot.



- Also, you can add another field in the hierarchy. For example, the **State** is inserted into the **Country** hierarchy. Then:
- Drag a field and drop it directly on top of another field in the hierarchy.
- It insert the **State** field into the **Country** hierarchy shown in the below screenshot.



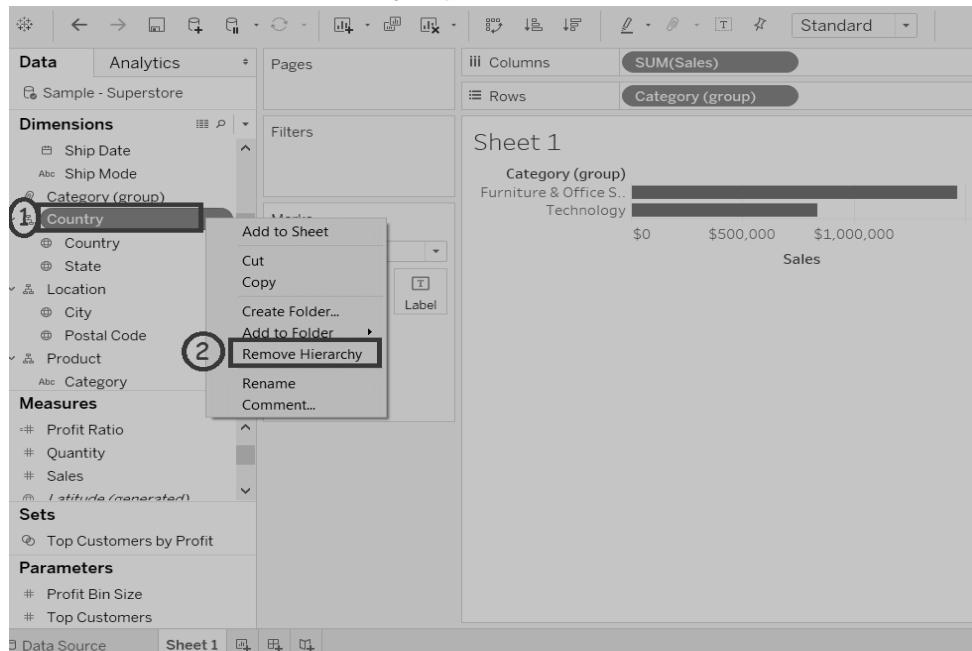
How to Remove a Hierarchy

From the data pane, you can remove the inbuilt hierarchy as well. Here are the following steps to remove the hierarchy.

Step 1: Select the hierarchy which you want to remove.

Step 2: Right-click on that hierarchy.

Step 3: And select the "Remove Hierarchy" option shown in below screenshot.



The fields in the hierarchy are also removed from the hierarchy, and the hierarchy disappears from the Data pane.

Tableau Sets

- Sets are custom fields and it defines a subset of data based on some conditions.
- Sets create a set of members out of the field present in a data set.
- It acts as a separate field or dimension.

The procedure to build sets is given step by step as follows.

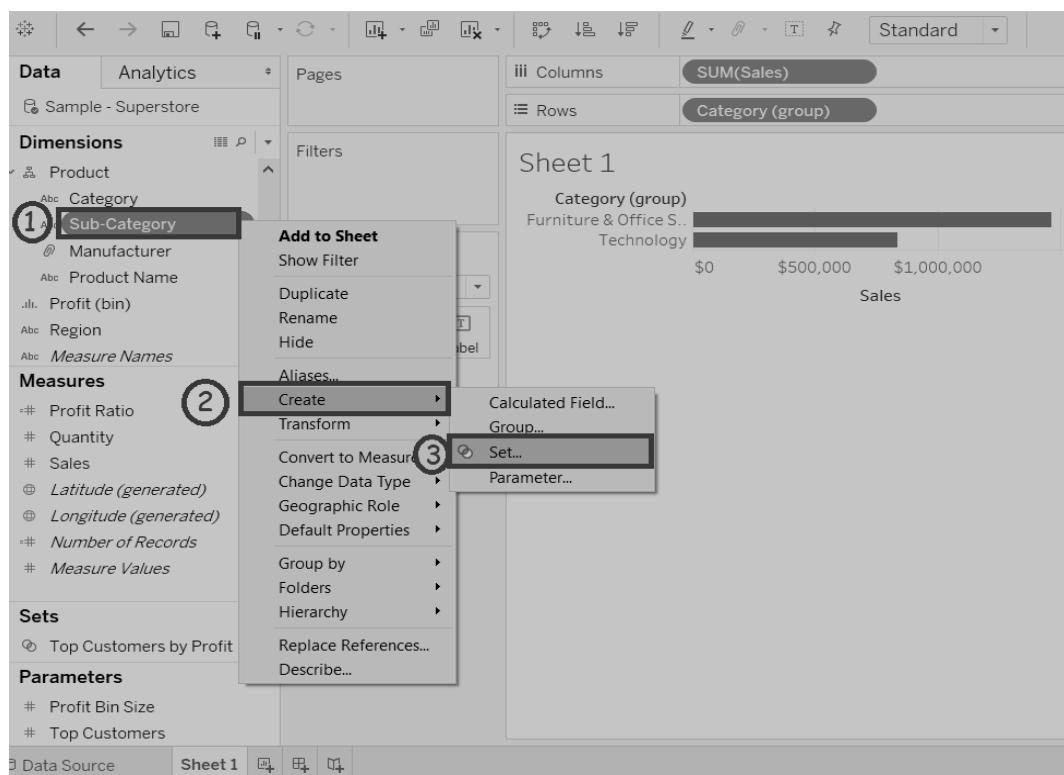
For examples, consider the data source such as **Sample-Superstore** and use its dimensions and measures to build the **Sets**.

Step 1: Go to the worksheet. And,

1. Right-click on a dimension **Sub-Category**.

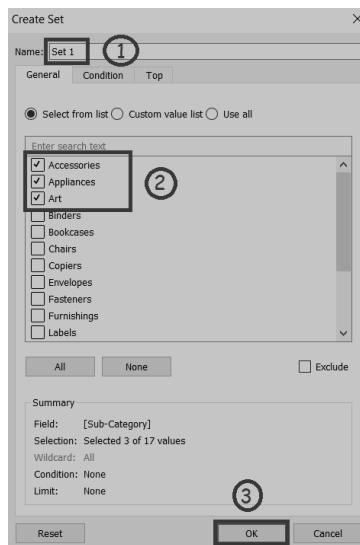
2. Select "**Create**" option.

3. Then click "**Set**" option shown in the following image.

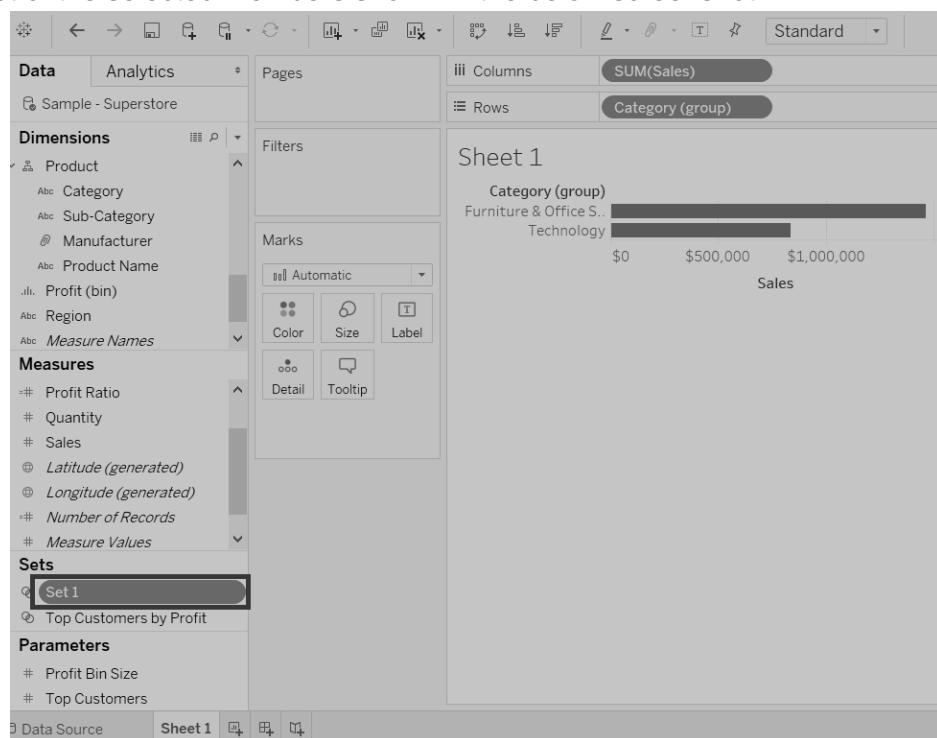


Step 2: It opens the **Create Set** window.

1. Enter the set name to be created.
2. Select the members which you want to add in the set.
3. Click on the **OK** button.



It creates a set of the selected members shown in the below screenshot.



Show Members in Set

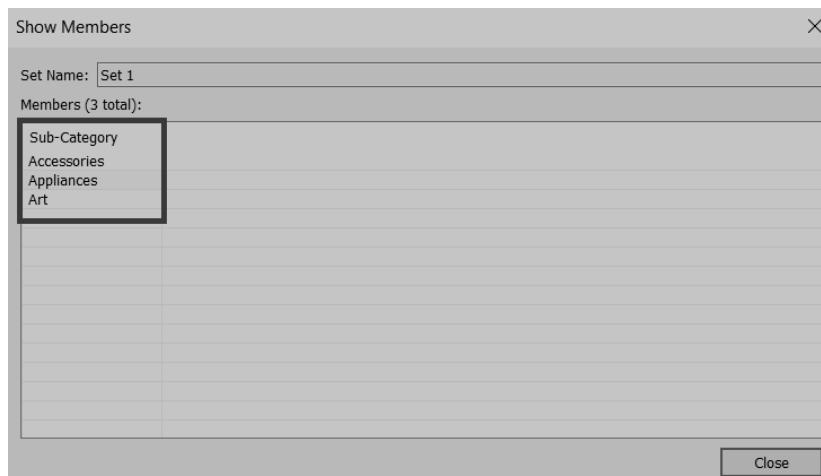
You can also see the selected members after a created or inbuilt set from the following steps:

Step 1: Right-click on the Set.

Step 2: Click on the "Show Members" option.

The screenshot shows the Tableau desktop interface. On the left, the Data pane displays various dimensions and measures. A context menu is open over a set named 'Set 1' in the 'Sets' section. The menu items include 'Add to Sheet', 'Show Filter', 'Cut', 'Copy', 'Create Folder...', 'Edit Set...', 'Duplicate', 'Rename', 'Hide', 'Delete', 'Create Calculated Field...', 'Create Parameter...', 'Default Properties', 'Hierarchy', 'Show members...', and 'Describe...'. The 'Show members...' option is highlighted with a circled number '2'. In the top right, a bar chart titled 'Sheet 1' shows sales for categories like Furniture & Office S., Technology, etc. The bottom right shows a sales distribution histogram.

After clicking the "Show Members" option, it will show all the members present in the set shown in below screenshot.



Edit the set

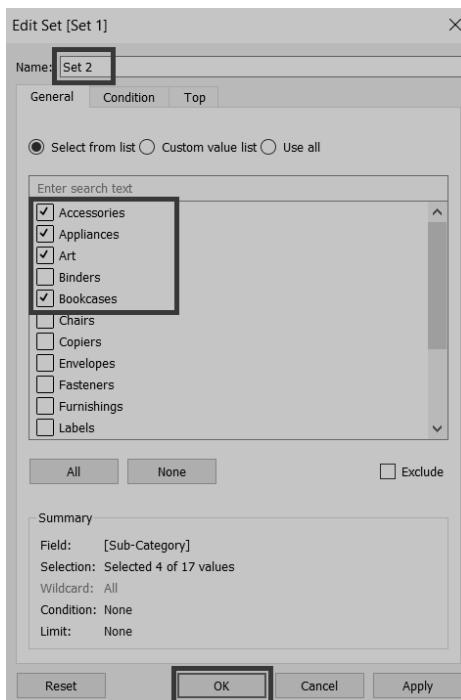
You also edit the set after created or inbuilt from the following steps:

Step 1: Right-click on the set.

Step 2: Click on the "Edit Set" option.

The screenshot shows the Tableau desktop interface. On the left, the data source pane lists 'Analytics' as the active data source, 'Sample - Superstore' as the active sheet, and various dimensions like Product, Region, and Measures like Sales and Profit Ratio. A context menu is open over 'Set 1' in the 'Sets' section of the sidebar. The menu items include 'Add to Sheet', 'Show Filter', 'Cut', 'Copy', 'Create Folder...', 'Edit Set...', 'Duplicate', 'Rename', 'Hide', 'Delete', 'Create Calculated Field...', 'Create Parameter...', 'Default Properties', 'Hierarchy', 'Show members...', and 'Describe...'. The 'Edit Set...' option is highlighted with a circled number 2. The main workspace shows a bar chart titled 'Sheet 1' with 'Category (group)' on the x-axis and 'Sales' on the y-axis, comparing Furniture & Office S. and Technology categories.

After clicking on the "Edit Set" option, **Edit Set** window will be opened with the set name. Now, you can edit the set shown in below screenshot.



4.15 Tableau Charts

Bar Chart

- In Tableau, there are various types of bar chart that can be created by using the dimensions and measures.
- A bar chart represents the data in rectangular bars.
- Tableau automatically produces a bar chart when you drag a dimension to the Row shelf and measure to the Column shelf.
- The bar chart option present in the "Show Me" button. If the data is not appropriate for the bar chart, then this option will be automatically blocked out.
- A bar chart can compare the data in different categories.
- The height of the bar represents the measured value of the category.
- It can be described as vertical and horizontal type bar charts.

The procedure to create a bar chart is given below through an example.

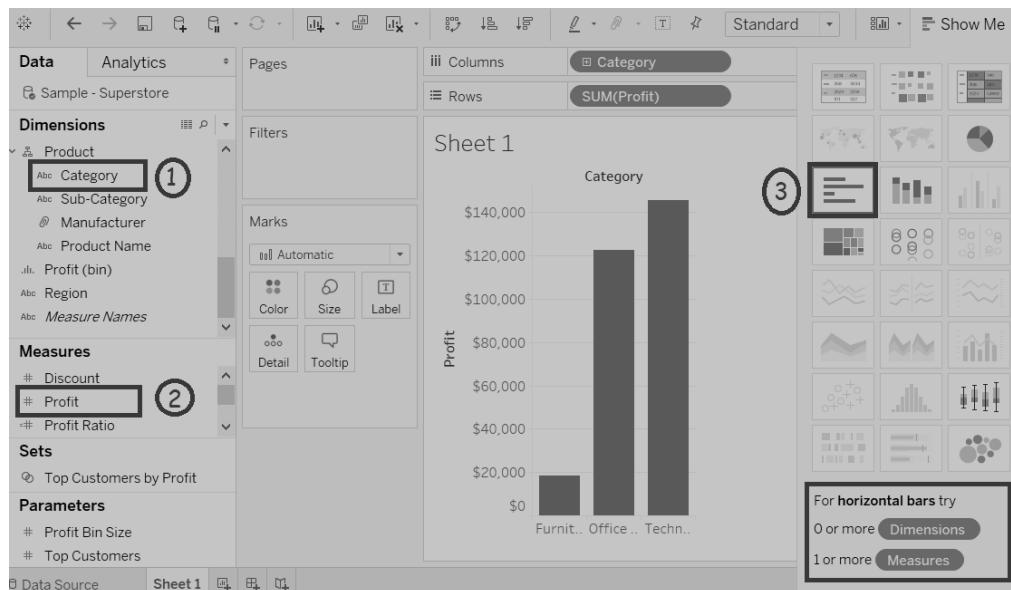
For example, consider a data source such as **Sample-Superstore** and its dimensions and measures.

Step 1: First, go to the worksheet

Step 2: Drag the **Category** field into the column shelf.

Step 3: Drag **Profit** field into the row shelf.

Step 4: By default, it creates the **bar chart** shown in the below screenshot.



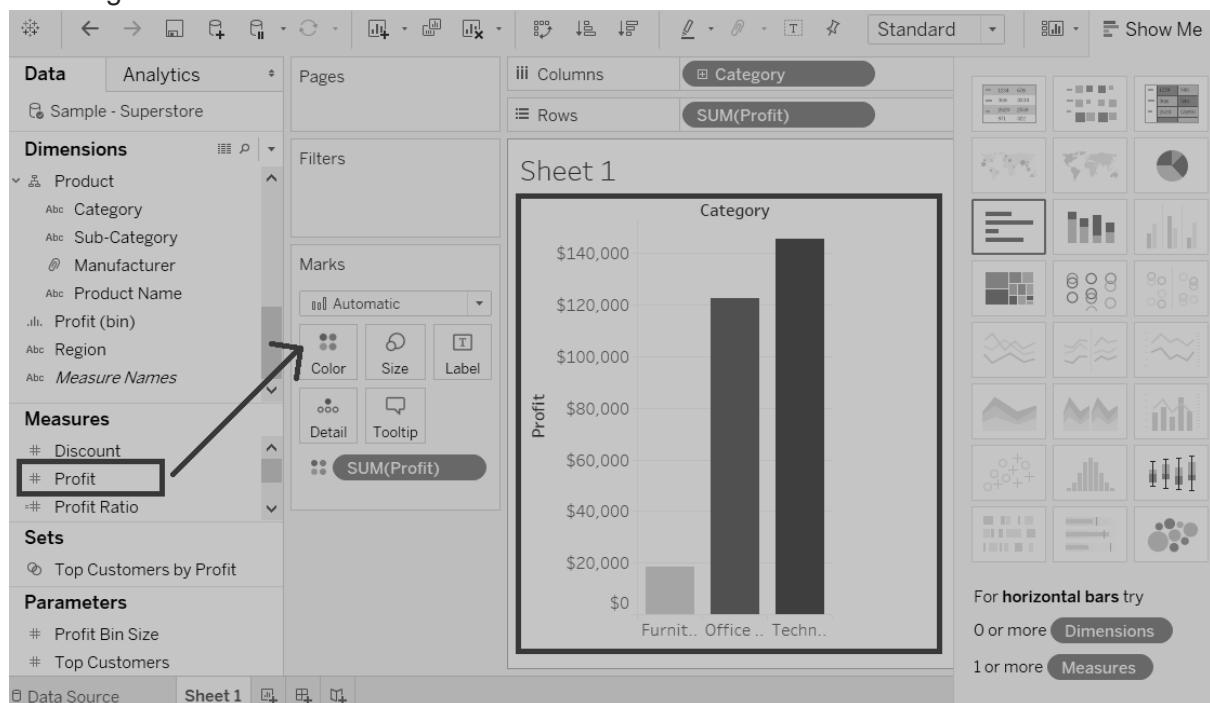
Bar Chart with Color Range

You can apply colors to the bars based on their ranges. The longer bars get darker shades, and the smaller bars get the lighter shades. Let's see step by steps,

Step 1: Drag the **Category** field into the column shelf.

Step 2: Drag **Profit** field into the row shelf.

Step 3: Also, drag the **Profit** field to the **Color** pane under the **Marks** Pane and, it produces a different color for negative bars.



Stacked Bar Chart

- You can also add one more dimension to the above bar chart to produce a stacked bar chart that shows different colors in each bar.

Step 1: Drag the **Segment** field.

Step 2: And drop the **Segment** field into **Color** pane.

The below-stacked chart appears that shows the distribution of each segment in each bar.

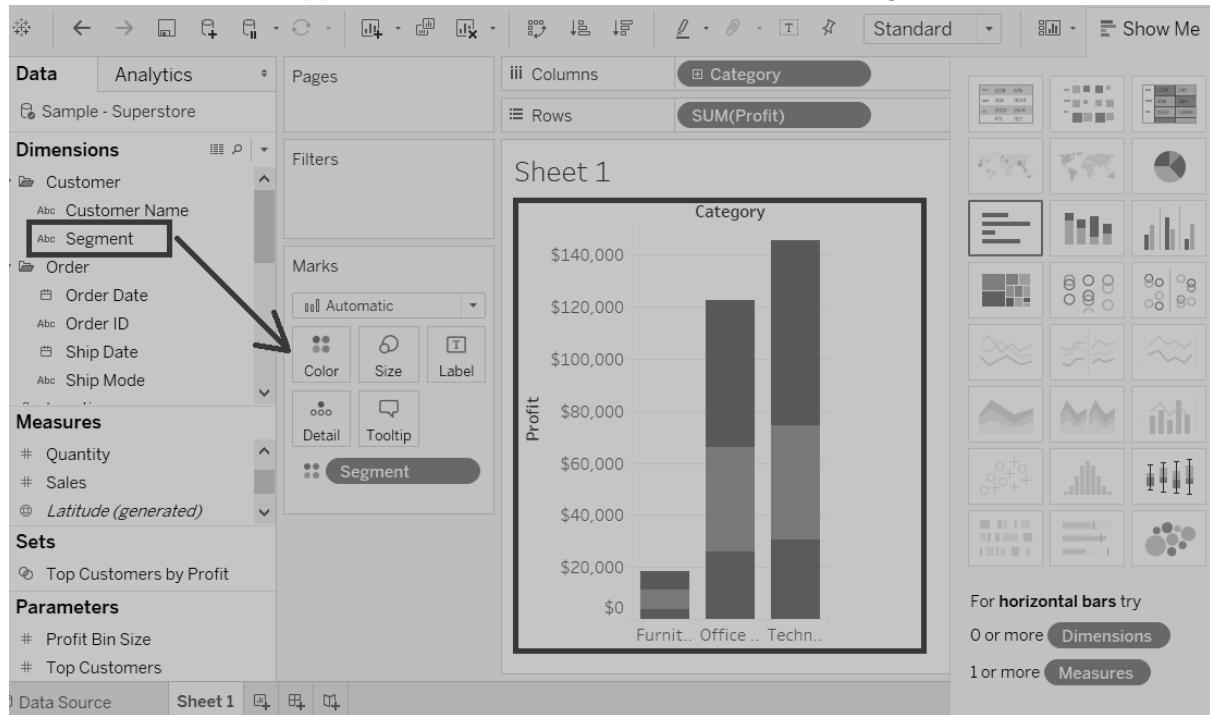


Tableau Line Chart

- A line Chart can compare the data over different periods.
- A series of dots create a line chart.
- These dots represent the measured values in each period.
- Measure and a dimension are taken two axes of the chart area in the line chart.
- The pair of values for each observation becomes a point.
- After joining all these points would become a line that shows the variation between the dimensions and measures.

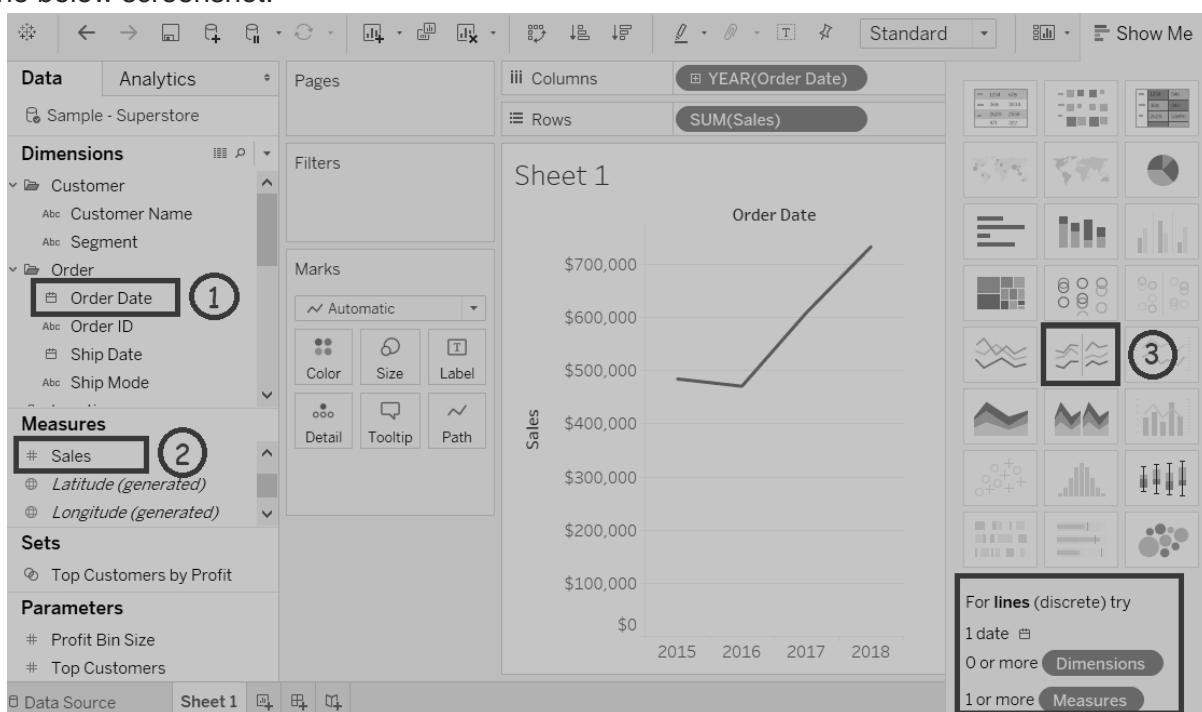
The procedure to create a line graph is shown step by step below.

For example, consider a data source such as **Sample-Superstore** and its dimensions and measures.

Step 1: Select one dimension and one measure to create a simple line chart.

1. Drag the dimension **Order Date** into Columns Shelf.
2. And **Sales** into the Rows shelf.
3. It creates the line chart by default or Chooses the Line chart from the "Show Me" button.

You will view the following line chart that shows the variation of **Sales** for different **Order Date** showing in the below screenshot.

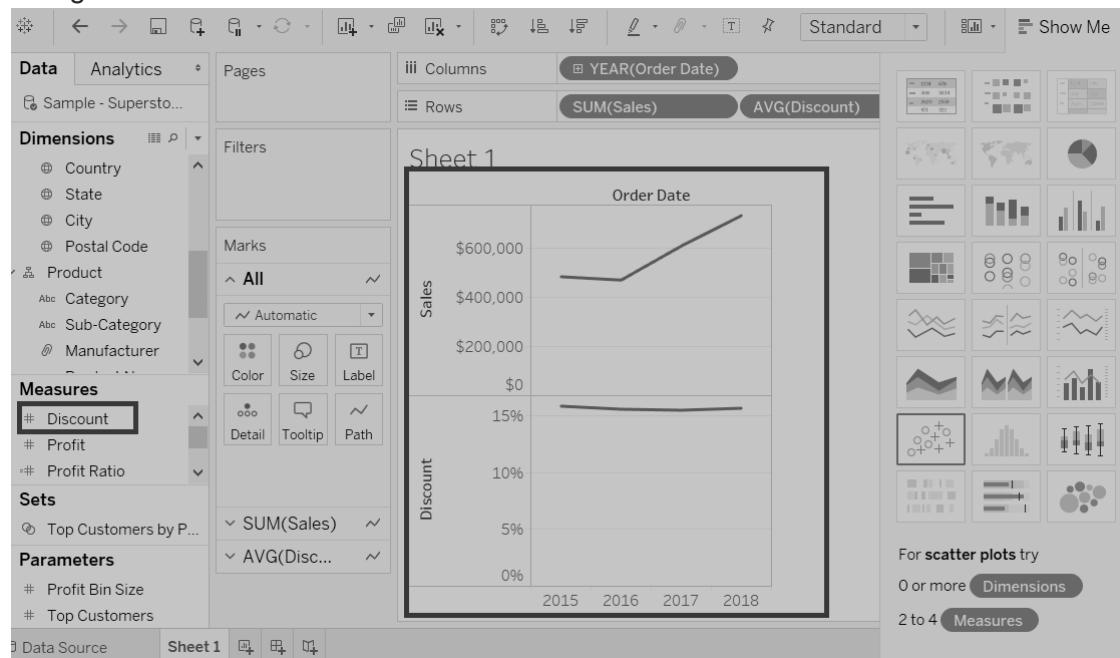


Multiple Measure Line Chart

- You can use one dimension with two or more measures in a single line chart.
- It produces various line charts in one pane.
- Each pane represents the variation between a dimension and the measures.

Step 1: Drag the dimension **Order Date** into Columns Shelf.

Step 2: Drag measures **Sales** and **Discount** into the Rows shelf.



Line Chart with Label

Each of the points that creates the line chart are labeled to make the values of the measure visible.

Step 1: Drop another measure **Profit** ratio into the "Labels" pane in the "Marks" card.

Step 2: Choose average as the aggregation, and you will view the below chart showing the labels.

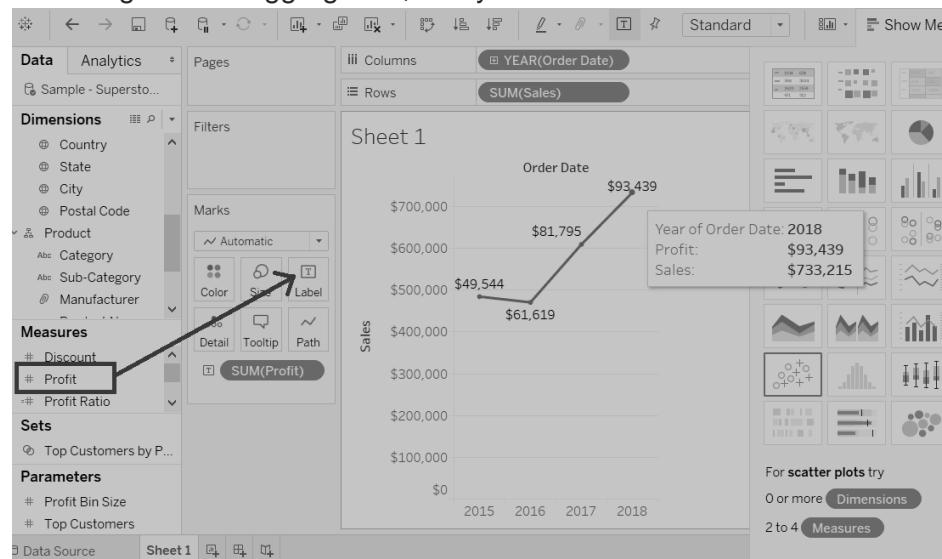


Tableau Pie Chart

- The pie chart shows the segment-wise data.
- It can show the contribution of measures over different members in a dimension.
- The angle of pie determines the measured value.
- Different colors can be assigned to pie to represent the members in a dimension.
- A pie chart represents the data in the form of the circle slice with different size and colors.
- These slices are labelled, and the numbers corresponding to each slice is also represented in the chart.

You select the pie chart option from the "Show Me" pane to create a pie chart.

For example, consider a data source such as **sample-superstore** and choose one dimension and one measure to create a simple pie chart.

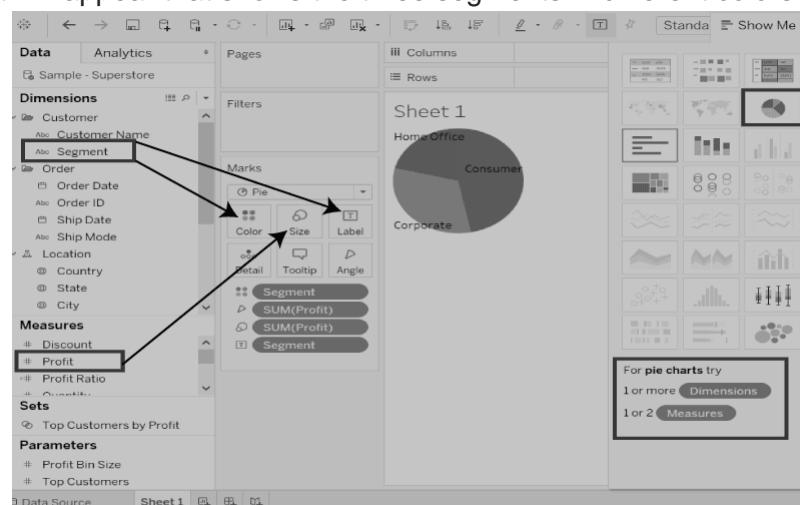
Step 1: Go to the worksheet.

Step 2: Drag the dimension **Segment** and drop into the **Color** and **Label** pane.

Step 3: Drag the measured **Profit** and drop into the **Size** pane.

Step 4: Choose the chart type from "Show Me" pane.

The following chart will appear that shows the three segments in different colors with labels.

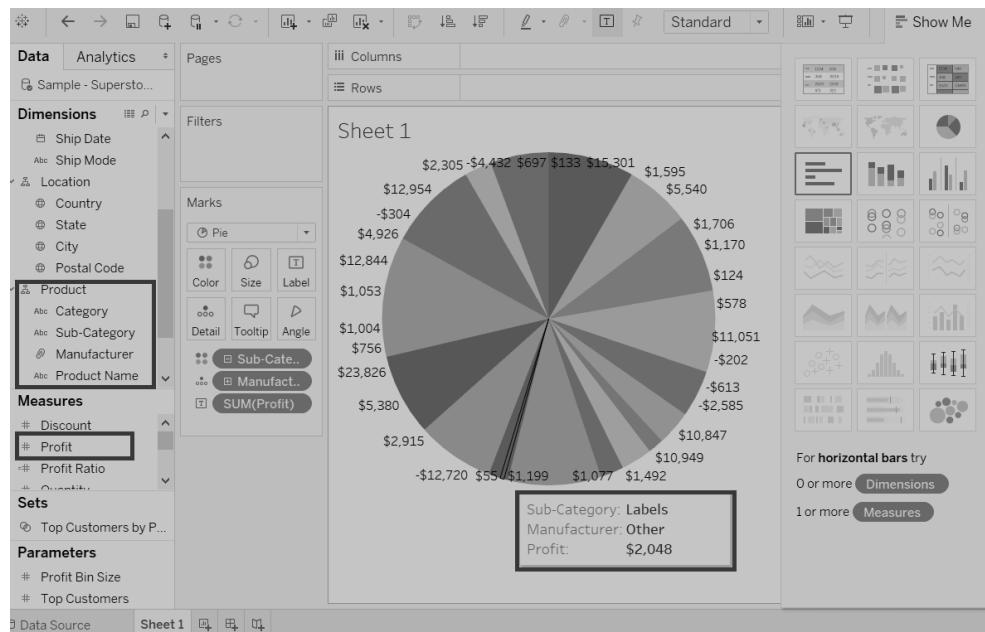


Drill Down Pie Chart

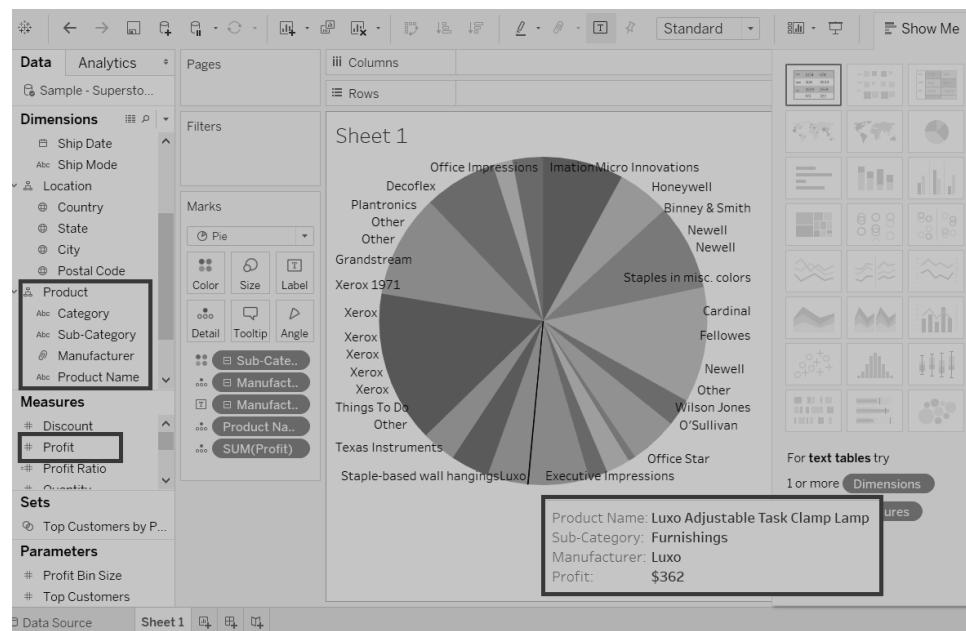
- You can choose a dimension with the hierarchy or go deeper into the hierarchy.
- The chart changes reflect the level of the selected dimension.

For example, consider a data source such as **sample-superstore**, then take the dimension **Product**, which has four more levels such as **Category**, **Sub-Category**, **Manufacturer**, and **Product Name**.

Drag the measured **Profit** and drop it to the **Labels** pane. The following pie chart appears that shows the values for each slice.



Here is one more level into this hierarchy, we get the manufacturer as the label shown in the below screenshot.

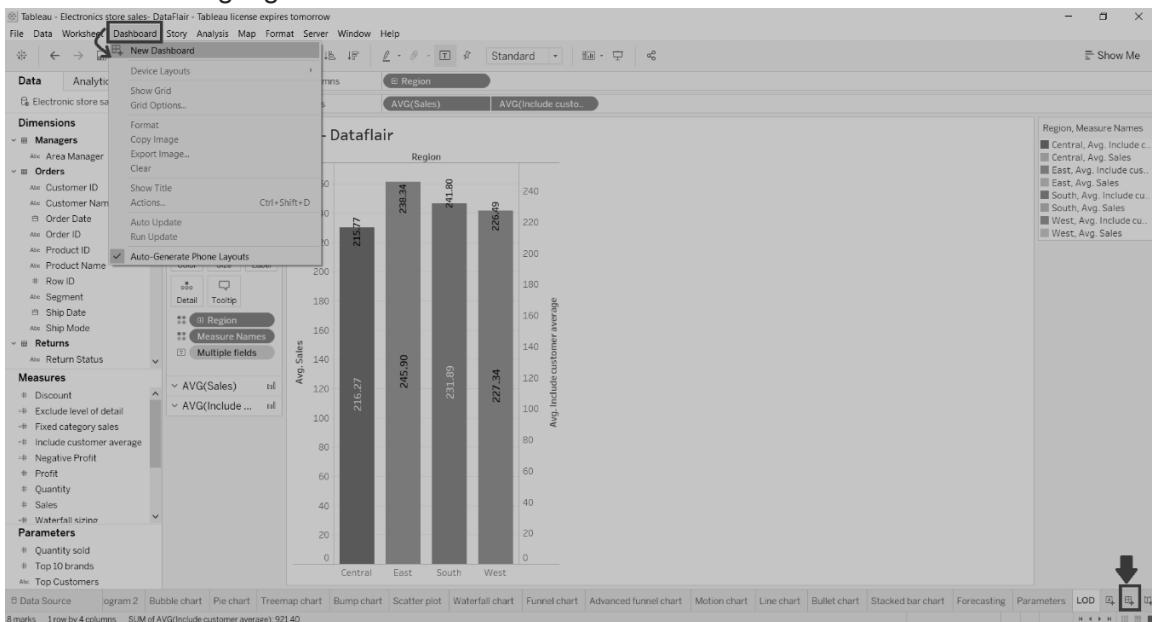


4.16 Creating a Dashboard in Tableau

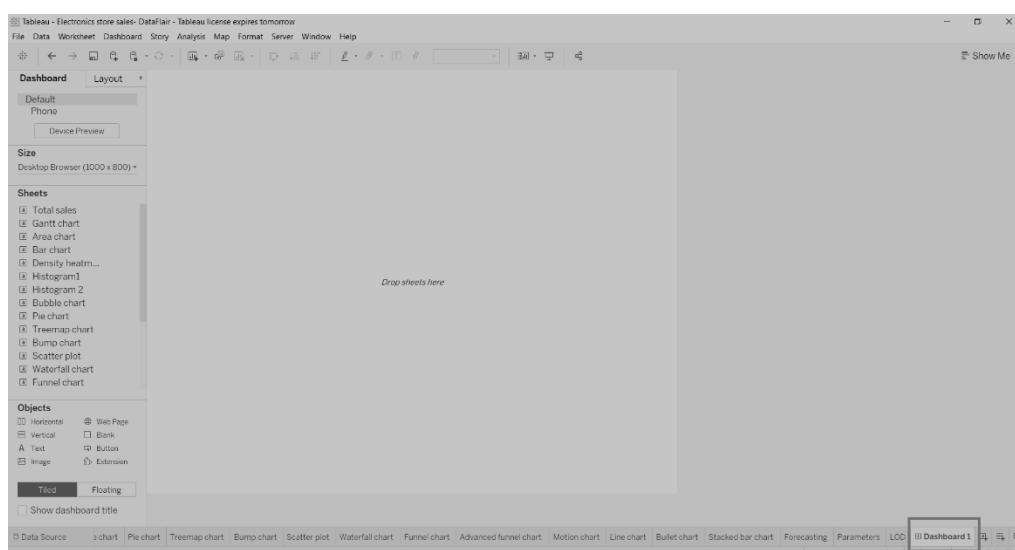
- A dashboard is a collection of different kinds of visualizations or views that we create on Tableau.
- We can bring together different elements of multiple worksheets and put them on a single dashboard.
- The dashboard option enables us to import and add charts and graphs from worksheets to create a dashboard.
- On a dashboard, we can place relevant charts and graphs in one view and analyze them for better insights.

1. Open a New Dashboard

You can open a dashboard window either from the **Dashboard** option given on the menu bar or from the Dashboard icon highlighted in red on the bottom bar.



Selecting the **New Dashboard** option or clicking on the Dashboard icon will open a new window named **Dashboard 1**. You change the name of the dashboard as per your liking.



2. Dashboard Pane

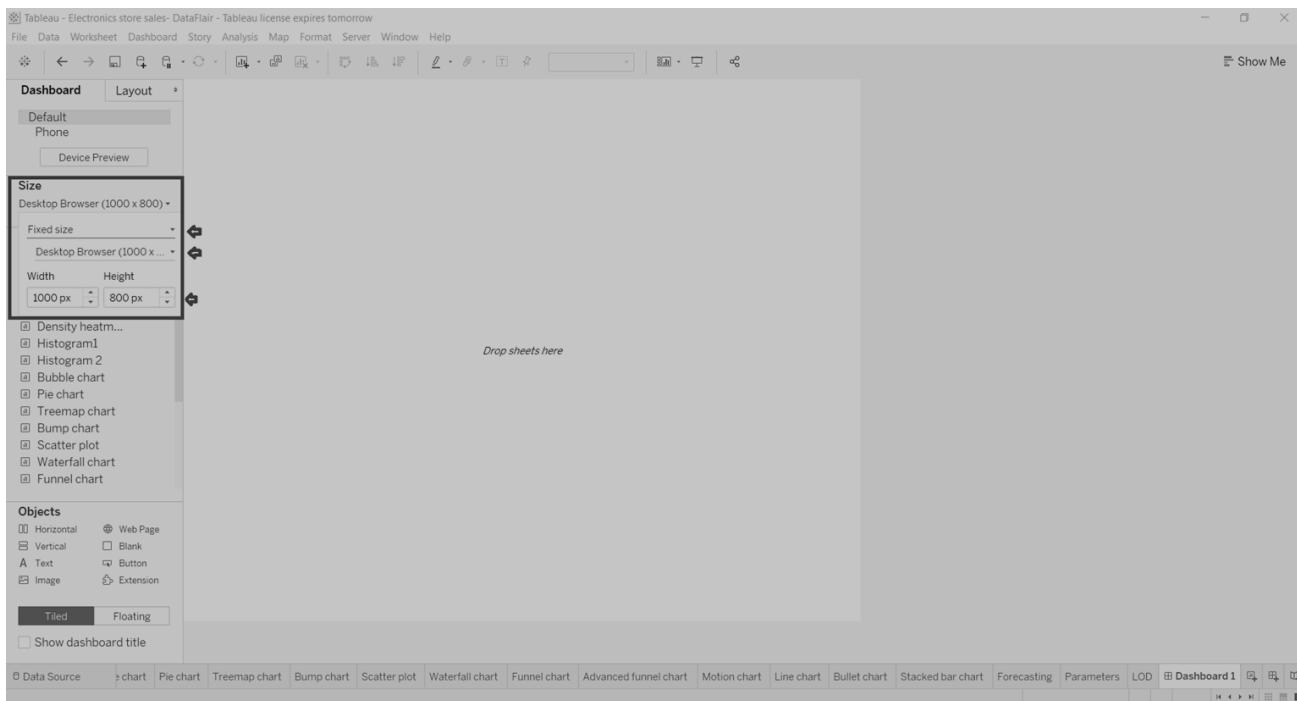
- In the window where we can create our dashboard, we get a lot of tabs and options related to dashboarding.
- On the left, we have a *Dashboard* pane which shows the dashboard size, list of available sheets in a workbook, objects, etc.

The screenshot shows the Tableau interface with the 'Dashboard' tab selected in the top navigation bar. The main workspace is a large area labeled 'Drop sheets here'. On the left, the 'Dashboard' pane is open, displaying the following sections:

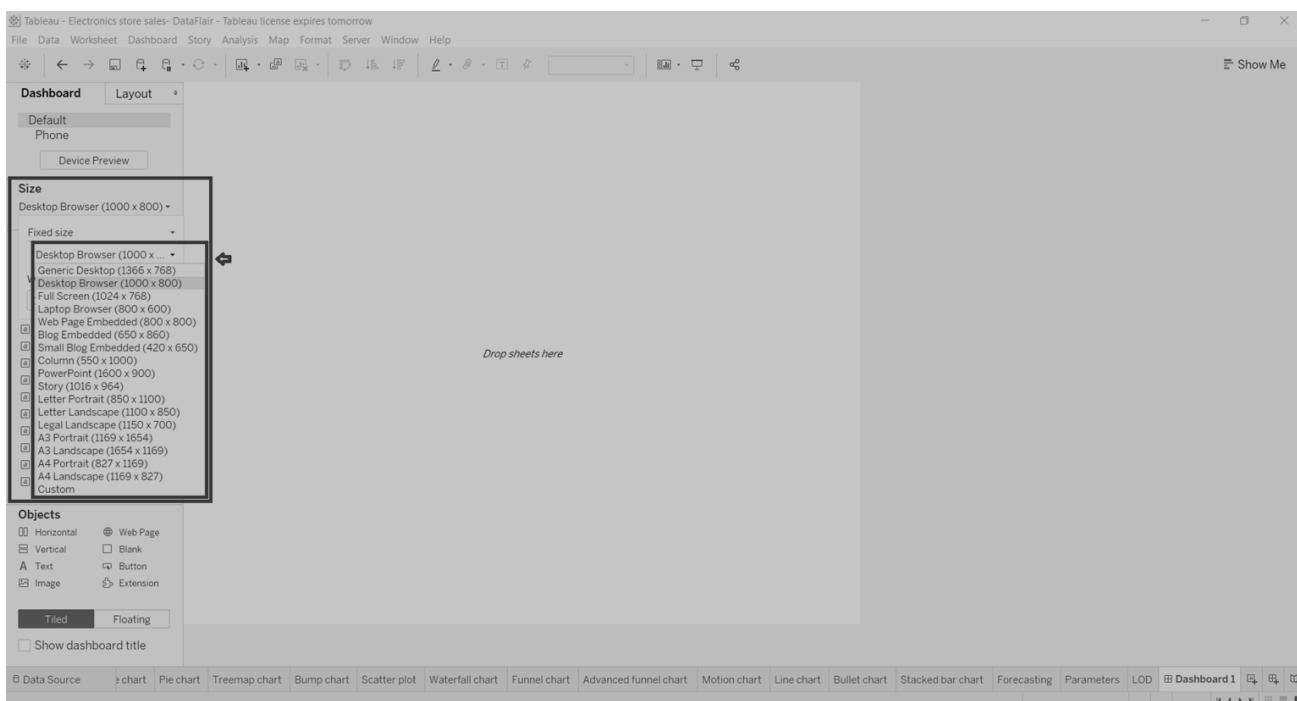
- Size:** Desktop Browser (1000 x 800)
- Sheets:** A list of available sheets including Total sales, Gantt chart, Area chart, Bar chart, Density heatmap, Histogram1, Histogram 2, Bubble chart, Pie chart, Treemap chart, Bump chart, Scatter plot, Waterfall chart, and Funnel chart. The 'Gantt chart' sheet is currently selected and previewed in the workspace.
- Objects:** Options for Horizontal, Vertical, Text, and Image objects, along with Tiled and Floating layout choices.

At the bottom of the dashboard pane, there are buttons for Data Source, >chart, Pie chart, Treemap chart, Bump chart, Scatter plot, Waterfall chart, Funnel chart, Advanced funnel chart, Motion chart, Line chart, Bullet chart, Stacked bar chart, Forecasting, Parameters, LOD, and Dashboard 1. The 'Dashboard 1' button is highlighted.

From the Dashboard tab, we can set the size of our dashboard. We can enter custom dimensions like the width and height of the dashboard as per our requirements.

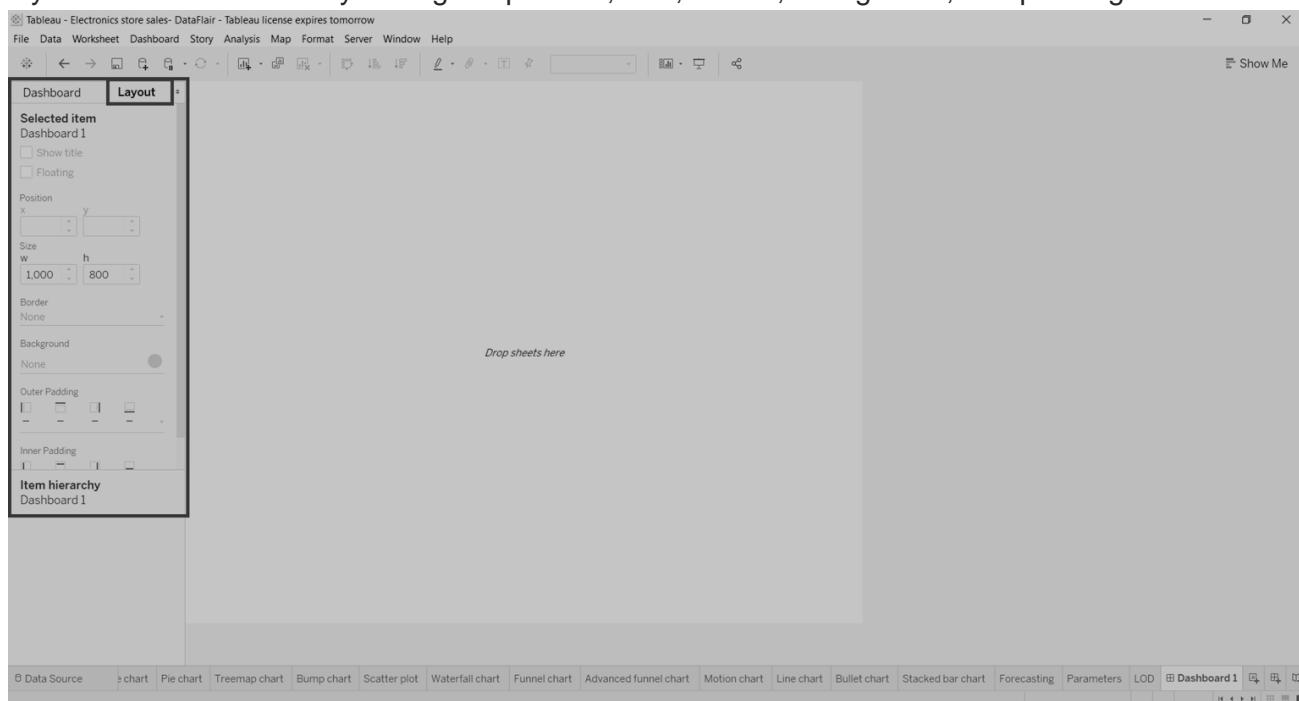


Or, you can select from a list of available fixed dashboard sizes as shown in the screenshot below.



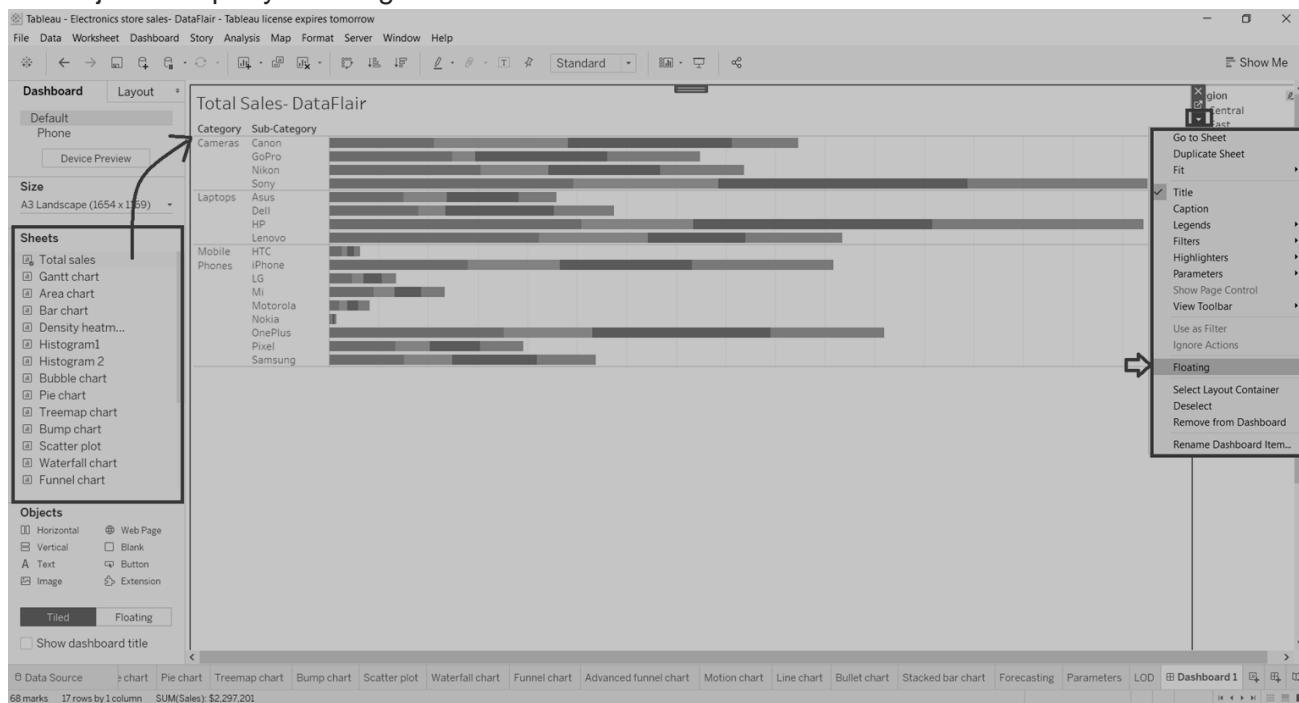
3. Layout Pane

Right next to the **Dashboard** pane is the **Layout** pane where we can enhance the appearance and layout of the dashboard by setting the position, size, border, background, and paddings.

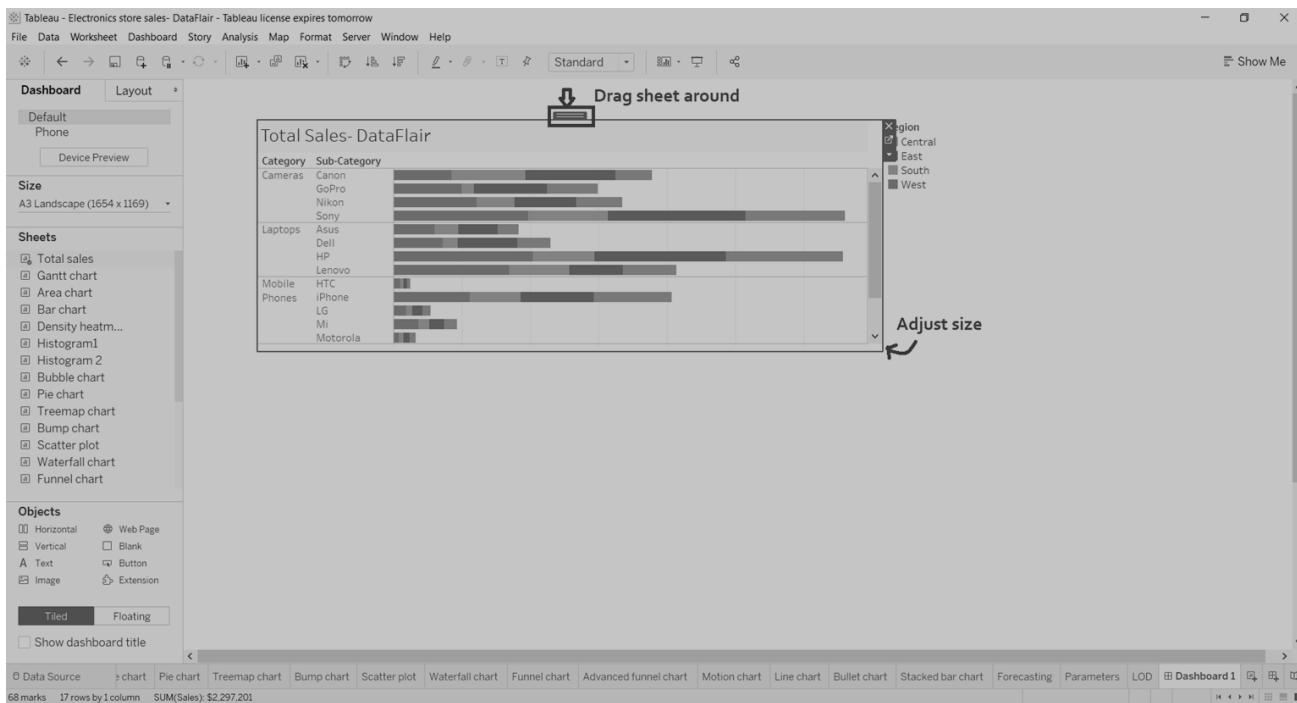


4. Adding a Sheet

Now, we'll add a sheet onto our empty dashboard. To add a sheet, drag and drop a sheet from the **Sheets** column present in the **Dashboard** tab. It will display all the visualizations we have on that sheet on our dashboard. If you wish to change or adjust the size and place of the visual/chart/graph, click on the **graph** then click on the small **downward arrow** given at the right. A drop-down list appears having the option **Floating**, select it. This will unfix your chart from one position so that you can adjust it as per your liking.

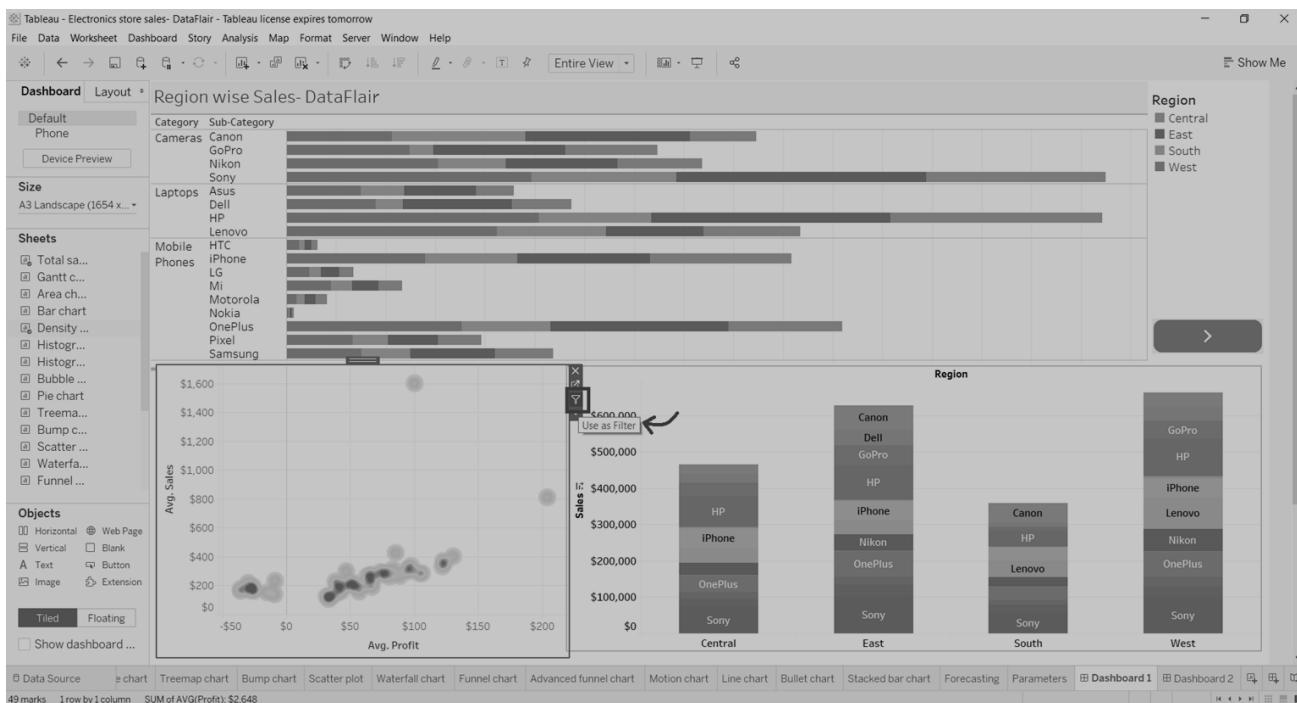


Have a look at the picture below to see how you can drag a sheet or visual around on the dashboard and adjust its size.

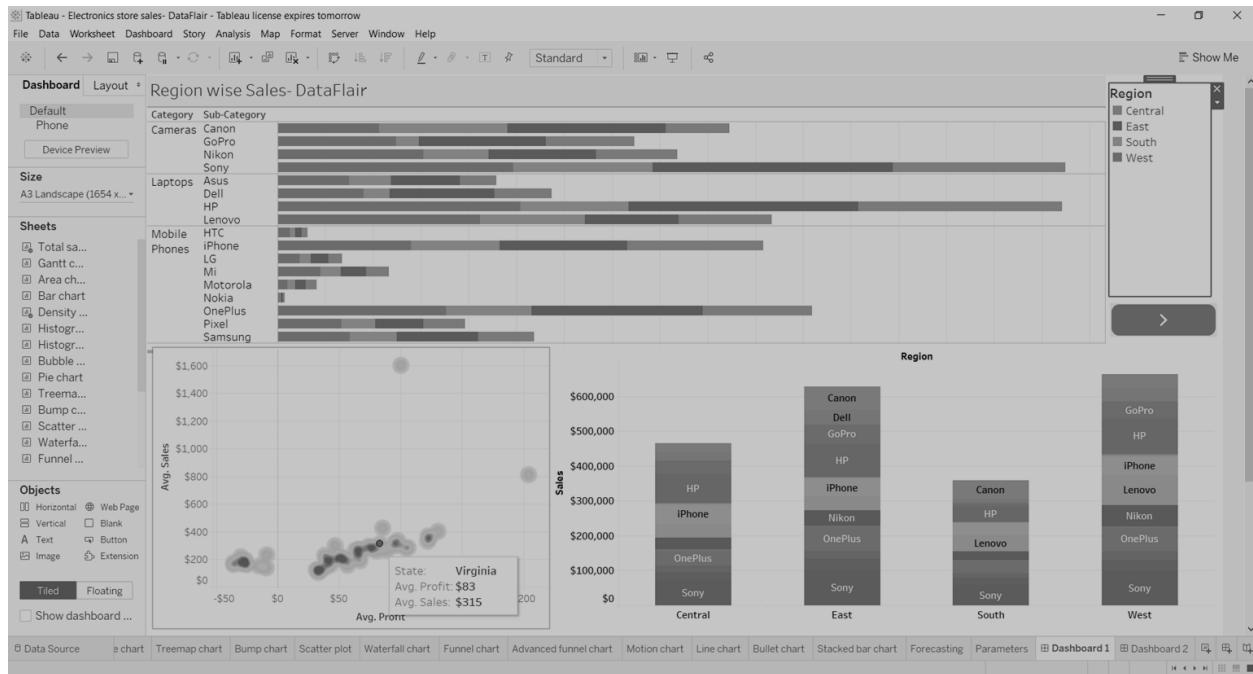


5. Adding more Sheets

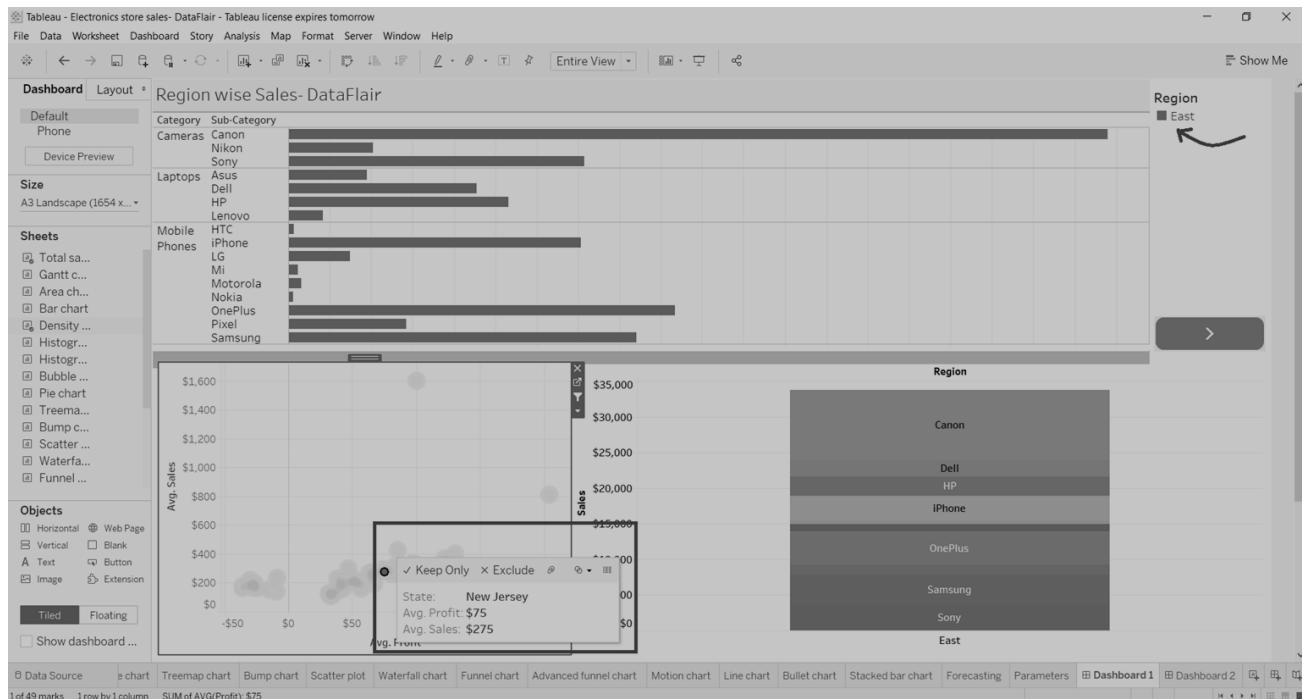
In a similar way, we can add as many sheets as we require and arrange them on the dashboard properly.



Also, you can apply the filter or selections on one graph and treat it like a filter for all the other visuals on the dashboard. To add a filter to a dashboard in Tableau, select **Use as Filter** option given on the right of every visual.

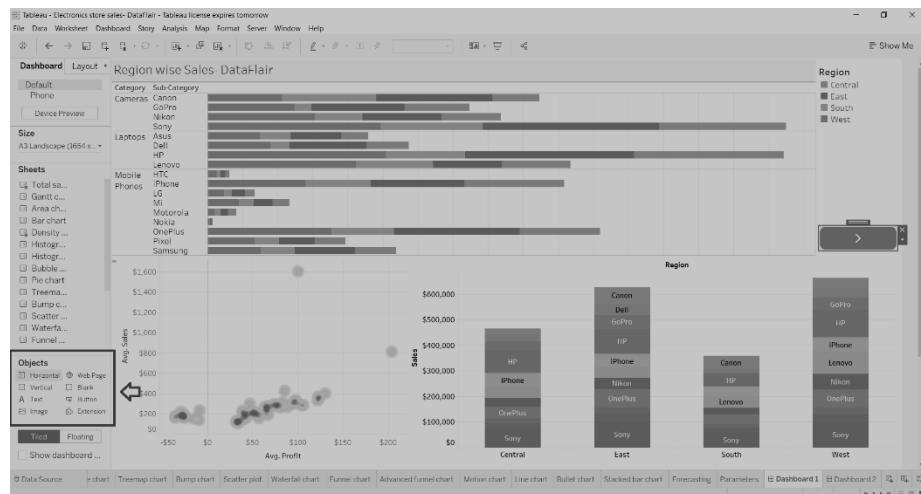


Then on the selected visual, we make selections. For instance, we select the data point corresponding to *New Jersey* in the heat map shown below. As soon as we select it, all the rest of the graphs and charts change their information and make it relevant to *New Jersey*. Notice in the **Region** section, the only region left is *East* which is where *New Jersey* is located.

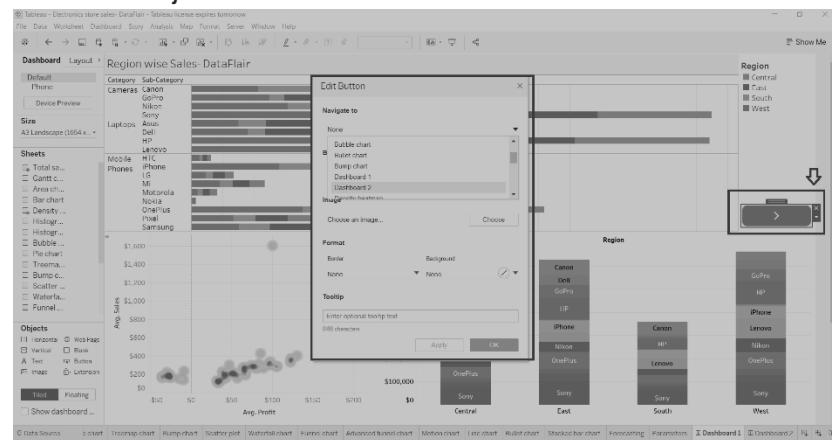


6. Adding Objects

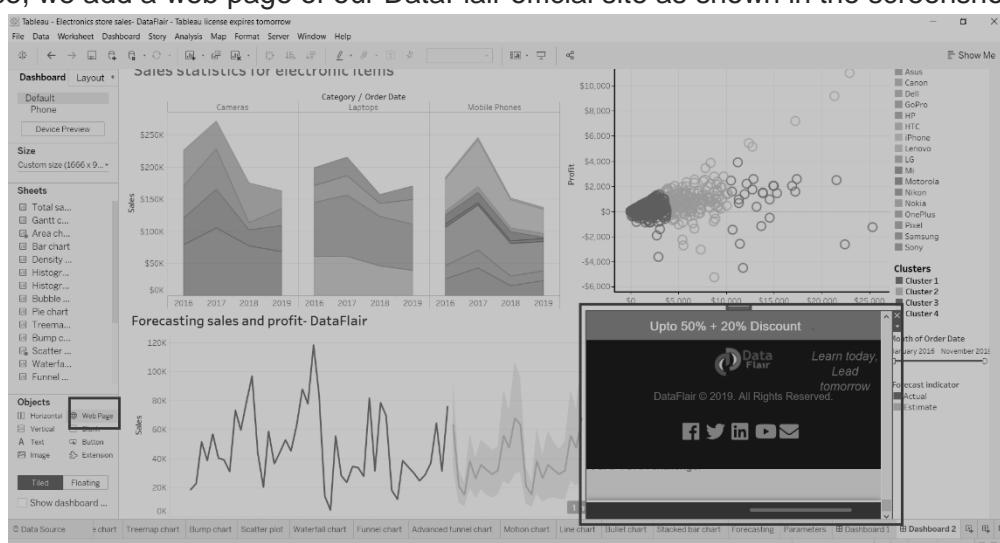
Another set of tools that we get to make our dashboard more interactive and dynamic is in the **Objects** section. We can add a wide variety of objects such as a web page, button, text box, extension, etc.



From the objects pane, we can add a button and also select the action of that button, that is, what that button should do when you click on it. Select the **Edit Button** option to explore the options you can select from for a button object.

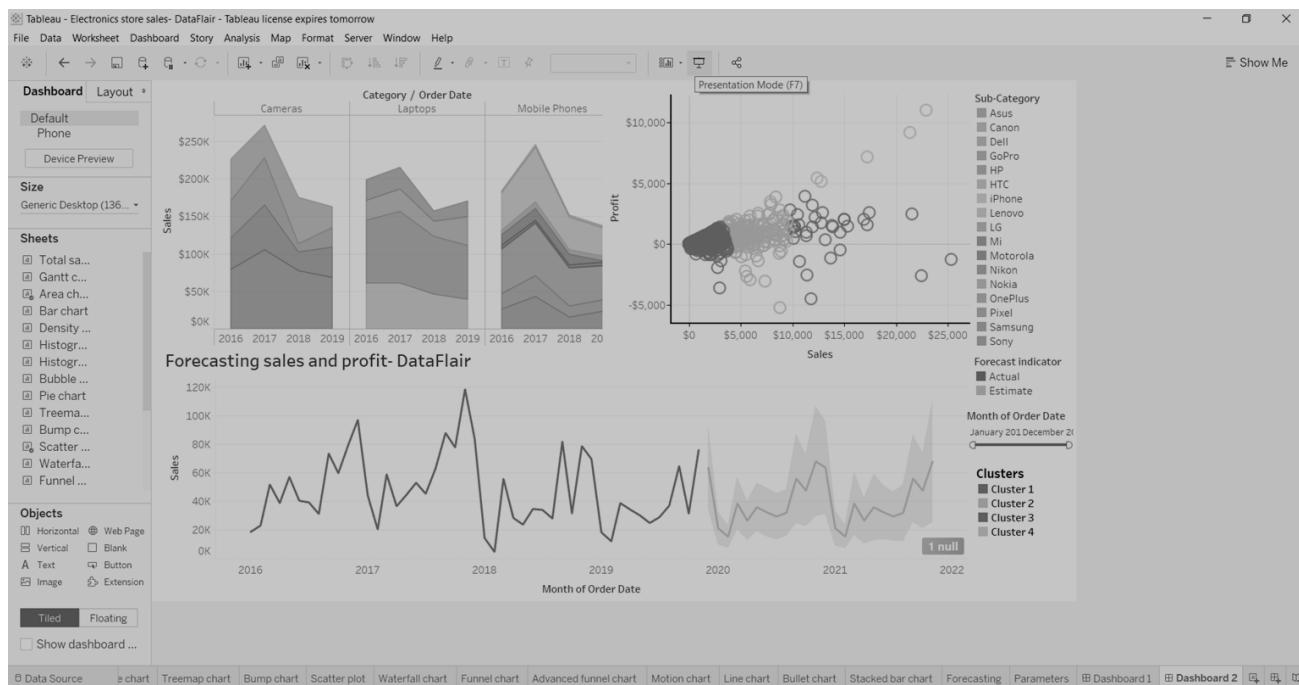


For instance, we add a web page of our DataFlair official site as shown in the screenshot below.



7. Final Dashboard

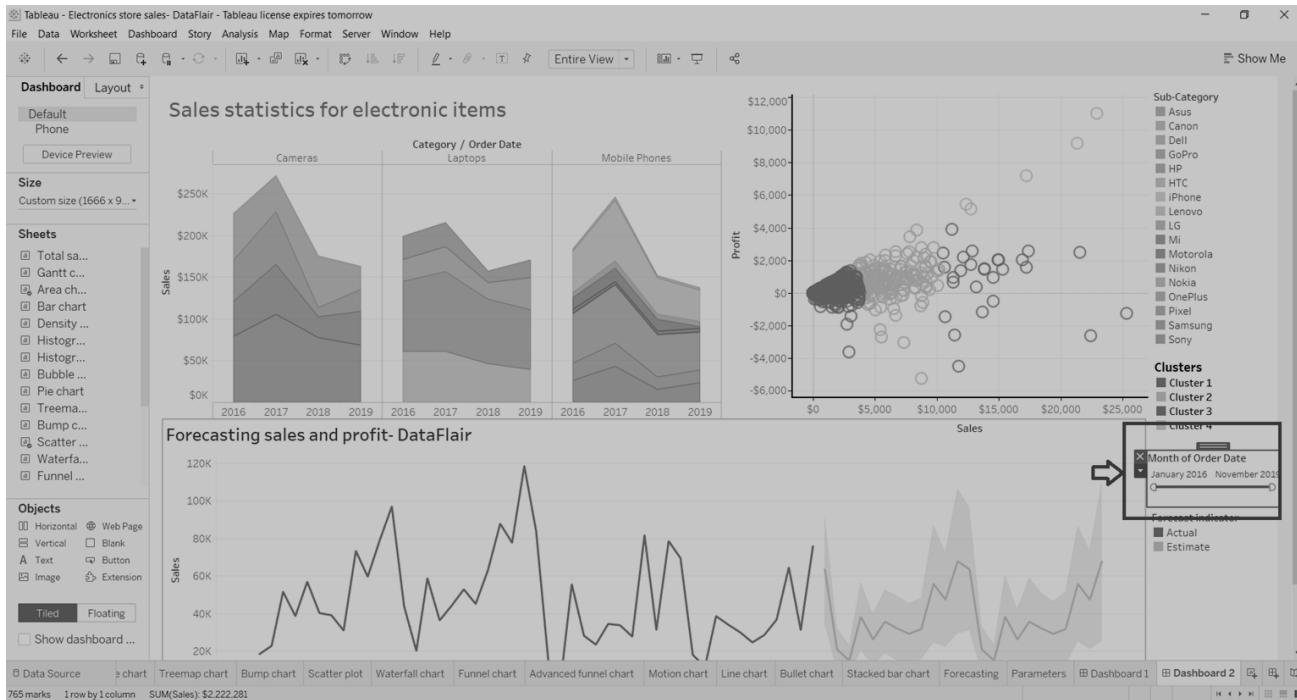
Now, we move towards making a final dashboard in Tableau with all its elements in place. As you can see in the screenshot below, we have three main visualizations on our current dashboard i.e. a **segmented area chart**, **scatter chart** and a **line chart** showing the sales and profits forecast. On the right pane, we have the list of legends showing Sub-category names, a forecast indicator and a list of clusters.



We can add filters on this dashboard by clicking on a visual. For instance, we want to add a filter based on months on the scatter plot showing sales values for different clusters. To add a months filter, we click on the small **downward arrow** and then select **Filters** option. Then we select **Months of Order Date** option. You can select any field based on which you wish to create a new filter.



This will give us a slider filter to select a range of months for which we want to see our data. You can adjust the position of the filter box and drag and drop it at whichever place you want.



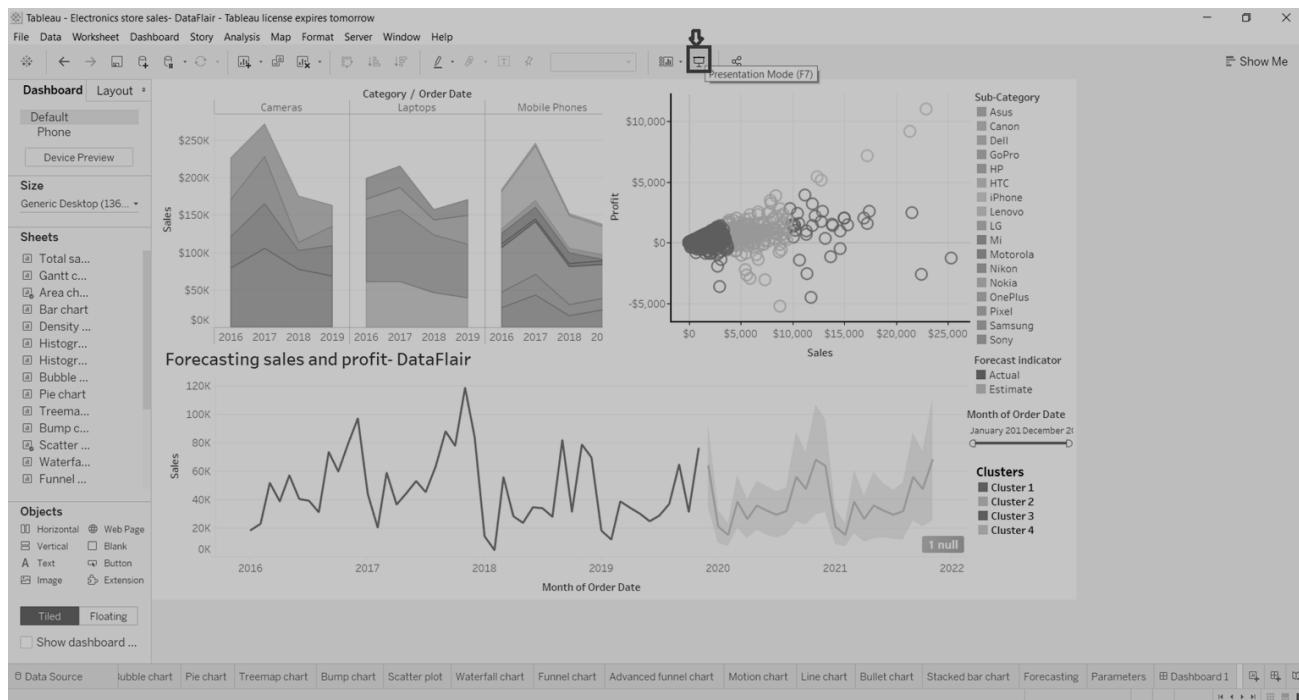
You can make more changes into the filter by right-clicking on it. Also, you can change the type of filter from the drop-down menu such as Relative Date, Range of Date, Start Date, End Date, Browse Periods, etc.



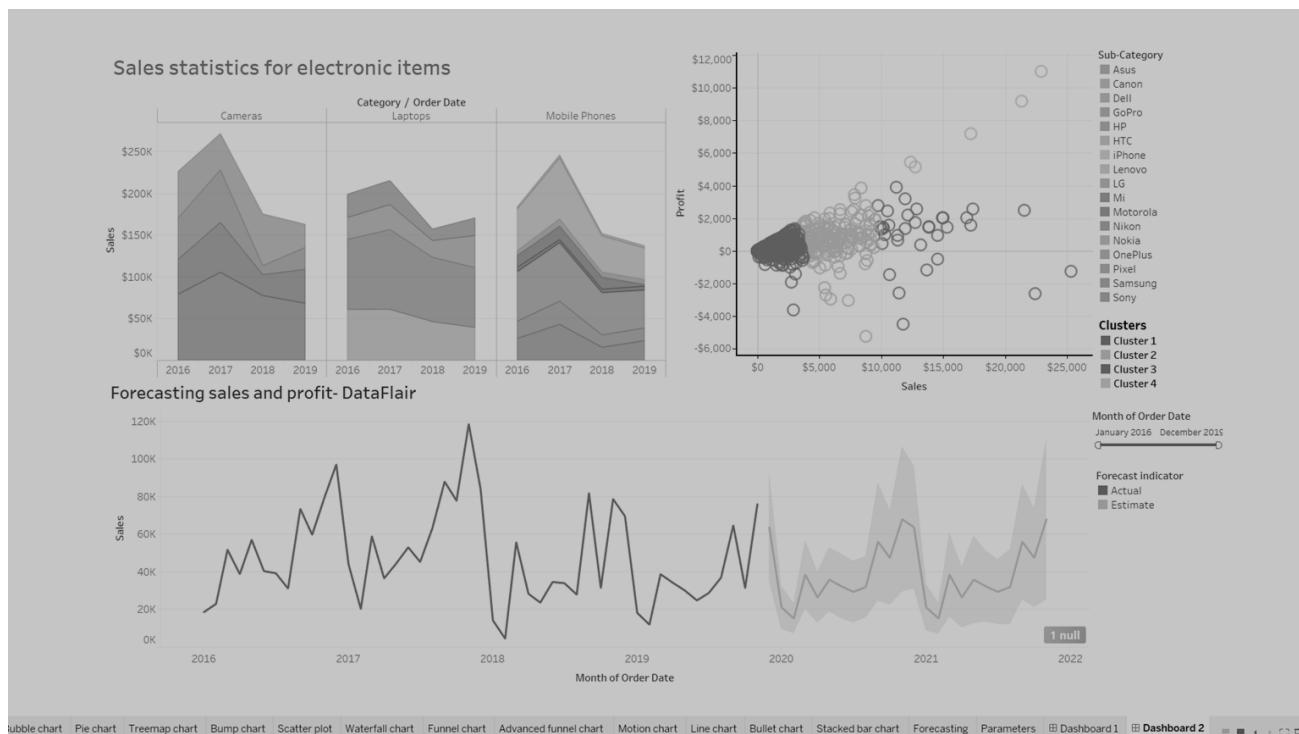
Similarly, you can add and edit more filters on the dashboard.

8. Presentation Mode

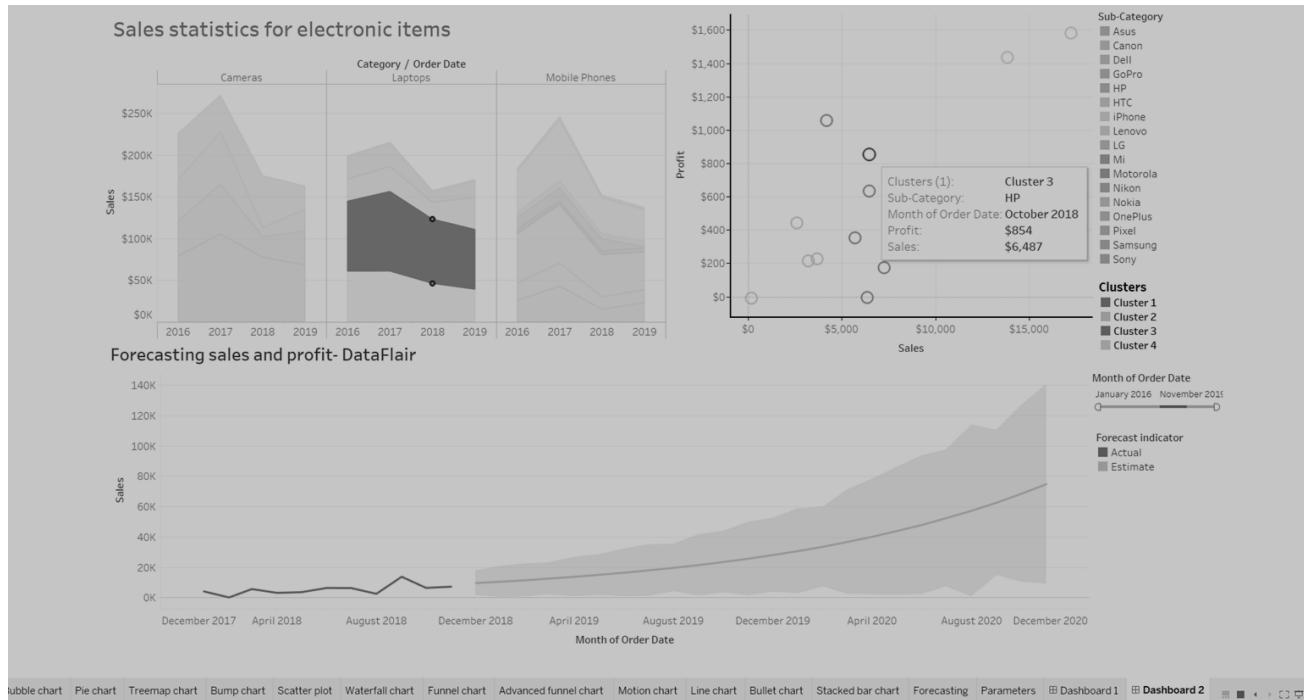
Once our dashboard is ready, we can view it in the **Presentation Mode**. To enable the presentation mode, click on the icon present on the bar at the top as shown in the screenshot below or press F7.



This opens our dashboard in the presentation mode. So far we were working in the Edit Mode. In the presentation mode, it neatly shows all the visuals and objects that we have added on the dashboard. We can see how the dashboard will look when we finally present it to others or share it with other people for analysis.



Here, we can also apply the filter range to our data. The dashboard is interactive and will change the data according to the filters we apply or selections we make.

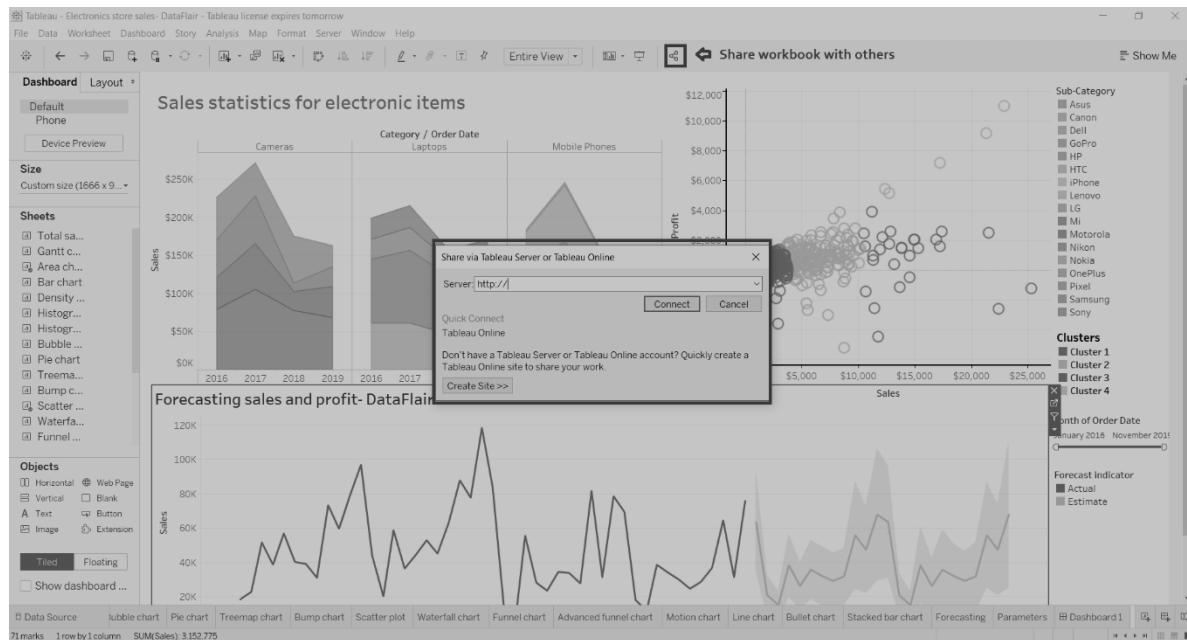


For instance, we selected the brand *Pixel* from our list of items from the sub-category field. This instantly changes the information on the visuals and makes it relevant to only Pixel.

9. Share Workbook with Others

We can also share all the worksheets and dashboard that we create together as a workbook with other users. To share the workbook with others, click on the **share** icon (highlighted in red). Next, you need to enter the server address of a Tableau server.

Note – You must have a Tableau Online or Tableau Server account in order to do this.



Section 3: Exercises

Exercise 1: Draw Tableau Architecture.

Exercise 2: Fill the data in following table.

Type	File Extension	Purpose
Tableau workbook		
Tableau Bookmarks		
Tableau Packaged workbook		
Tableau Data Extract		
Tableau Data Source		
Tableau Packaged Data Source		
Tableau Preferences		

Exercise 3: Tabulate the Precedence of Operator in Tableau.

Exercise 4: Participate in group discussion on following topics:

- a) Tools of Tableau
- b) Tableau Architecture
- c) Data Aggregation in Tableau
- d) Tableau File Types
- e) Tableau Operators and Calculation
- f) Functions and Calculations in Tableau
- g) Tableau LOD Expressions
- h) Tableau Filter Operations

Section 4: Assessment Questionnaire

1. What are the features of Tableau?
2. What are the advantages of Tableau?
3. What are the disadvantages of Tableau?
4. Name the tools of Tableau?
5. Enlist different aggregation types in Tableau.
6. What are different types of Joins?
7. Enlist Tableau functions.
8. What are the types of LOD expressions?
9. How is the Context Filter different from the other filters?
10. Can you use non-used columns, which are columns that are not used in reports but used in data sources, in Tableau filters?

-----End of the Module-----

MODULE 5

SQL TRAINING

Section 1: Learning Outcomes

After completing this module, you will be able to:

- Use SQL Statements, Data Types and Operators
- Perform Recovery and Back-up
- Execute Selection Commands
- Explain Types of Subqueries
- Describe Tables
- Use various Functions of SQL
- Perform Pattern (String) Matching
- Use Access Control Function

Section 2: Relevant Knowledge

5.1 Introduction to SQL

- SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.
- SQL is not a database system, but it is a query language.
- Suppose you want to perform the queries of SQL language on the stored data in the database. You are required to install any database management system in your systems, for example, Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, DB2, etc.

What is SQL?

- SQL is a short-form of the Structured Query Language, and it is pronounced as S-Q-L or sometimes as See-Quell.
- This database language is mainly designed for maintaining the data in relational database management systems.
- It is a special tool used by data professionals for handling structured data (data which is stored in the form of tables). It is also designed for stream processing in RDSMS.
- You can easily create and manipulate the database, access and modify the table rows and columns, etc.
- This query language became the standard of ANSI in the year of 1986 and ISO in the year of 1987.
- If you want to get a job in the field of data science, then it is the most important query language to learn.
- Big enterprises like Facebook, Instagram, and LinkedIn, use SQL for storing the data in the back-end.

Why SQL?

Nowadays, SQL is widely used in data science and analytics. Following are the reasons which explain why it is widely used:

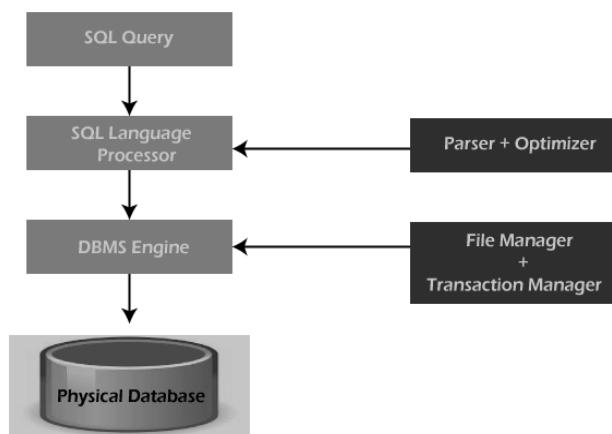
- The basic use of SQL for data professionals and SQL users is to insert, update, and delete the data from the relational database.
- SQL allows the data professionals and users to retrieve the data from the relational database management systems.
- It also helps them to describe the structured data.
- It allows SQL users to create, drop, and manipulate the database and its tables.
- It also helps in creating the view, stored procedure, and functions in the relational database.
- It allows you to define the data and modify that stored data in the relational database.
- It also allows SQL users to set the permissions or constraints on table columns, views, and stored procedures.

History of SQL

- "A Relational Model of Data for Large Shared Data Banks" was a paper which was published by the great computer scientist "E.F. Codd" in 1970.
- The IBM researchers Raymond Boyce and Donald Chamberlin originally developed the SEQUEL (Structured English Query Language) after learning from the paper given by E.F. Codd.
- They both developed the SQL at the San Jose Research laboratory of IBM Corporation in 1970.
- At the end of the 1970s, relational software Inc. developed their own first SQL using the concepts of E.F. Codd, Raymond Boyce, and Donald Chamberlin. This SQL was totally based on RDBMS.
- Relational Software Inc., which is now known as Oracle Corporation, introduced the Oracle V2 in June 1979, which is the first implementation of SQL language.
- This Oracle V2 version operates on VAX computers.

Process of SQL

- When we are executing the command of SQL on any Relational database management system, then the system automatically finds the best routine to carry out our request, and the SQL engine determines how to interpret that particular command.
- Structured Query Language contains the following four components in its process:
 - Query Dispatcher
 - Optimization Engines
 - Classic Query Engine
 - SQL Query Engine, etc.
- A classic query engine allows data professionals and users to maintain non-SQL queries. The architecture of SQL is shown in the following diagram:



Advantages of SQL

- SQL provides various advantages which make it more popular in the field of data science.
- It is a perfect query language which allows data professionals and users to communicate with the database.

Following are the best advantages or benefits of Structured Query Language:

1. No programming needed

- SQL does not require a large number of coding lines for managing the database systems.
- We can easily access and maintain the database by using simple SQL syntactical rules.
- These simple rules make the SQL user-friendly.

2. High-Speed Query Processing

- A large amount of data is accessed quickly and efficiently from the database by using SQL queries.
- Insertion, deletion, and updation operations on data are also performed in less time.

3. Standardized Language

- SQL follows the long-established standards of ISO and ANSI, which offer a uniform platform across the globe to all its users.

4. Portability

- The structured query language can be easily used in desktop computers, laptops, tablets, and even smartphones.
- It can also be used with other applications according to the user's requirements.

5. Interactive language

- We can easily learn and understand the SQL language.
- We can also use this language for communicating with the database because it is a simple query language.
- This language is also used for receiving the answers to complex queries in a few seconds.

6. More than one Data View

- The SQL language also helps in making the multiple views of the database structure for the different database users.

Disadvantages of SQL

With the advantages of SQL, it also has some disadvantages, which are as follows:

1. Cost

- The operation cost of some SQL versions is high. That's why some programmers cannot use the Structured Query Language.

SQL	No-SQL
1. SQL is a relational database management system.	1. While No-SQL is a non-relational or distributed database management system.
2. The query language used in this database system is a structured query language.	2. The query language used in the No-SQL database systems is a non-declarative query language.
3. The schema of SQL databases is predefined, fixed, and static.	3. The schema of No-SQL databases is a dynamic schema for unstructured data.
4. These databases are vertically scalable.	4. These databases are horizontally scalable.
5. The database type of SQL is in the form of tables, i.e., in the form of rows and columns.	5. The database type of No-SQL is in the form of documents, key-value, and graphs.
6. It follows the ACID model.	6. It follows the BASE model.
7. Complex queries are easily managed in the SQL database.	7. NoSQL databases cannot handle complex queries.
8. This database is not the best choice for storing hierarchical data.	8. While No-SQL database is a perfect option for storing hierarchical data.
9. All SQL databases require object-relational mapping.	9. Many No-SQL databases do not require object-relational mapping.
10. Gauges, CircleCI, Hootsuite, etc., are the top enterprises that are using this query language.	10. Airbnb, Uber, and Kickstarter are the top enterprises that are using this query language.
11. SQLite, Ms-SQL, Oracle, PostgreSQL, and MySQL are examples of SQL database systems.	11. Redis, MongoDB, Hbase, BigTable, CouchDB, and Cassandra are examples of NoSQL database systems.

2. Interface is Complex

- Another big disadvantage is that the interface of Structured query language is difficult, which makes it difficult for SQL users to use and manage it.

3. Partial Database control

- The business rules are hidden. So, the data professionals and users who are using this query language cannot have full database control.

SQL vs No-SQL

The following table describes the differences between the SQL and NoSQL, which are necessary to understand:

5.2 SQL Commands

The SQL commands help in creating and managing the database. The most common SQL commands which are highly used are mentioned below:

1. CREATE command
2. UPDATE command
3. DELETE command
4. SELECT command
5. DROP command
6. INSERT command

CREATE Command

This command helps in creating the new database, new table, table view, and other objects of the database.

UPDATE Command

This command helps in updating or changing the stored data in the database.

DELETE Command

- This command helps in removing or erasing the saved records from the database tables.
- It erases single or multiple tuples from the tables of the database.

SELECT Command

- This command helps in accessing the single or multiple rows from one or multiple tables of the database.
- We can also use this command with the WHERE clause.

DROP Command

This command helps in deleting the entire table, table view, and other objects from the database.

INSERT Command

- This command helps in inserting the data or records into the database tables.
- We can easily insert the records in single as well as multiple rows of the table.

5.3 SQL Syntax and Statements

SQL Syntax

- When you want to do some operations on the data in the database, then you must have to write the query in the predefined syntax of SQL.
- The syntax of the structured query language is a unique set of rules and guidelines, which is not case-sensitive. Its Syntax is defined and maintained by the ISO and ANSI standards.

Following are some most important points about the SQL syntax which are to remember:

1. You can write the keywords of SQL in both uppercase and lowercase, but writing the SQL keywords in uppercase improves the readability of the SQL query.
2. SQL statements or syntax are dependent on text lines. We can place a single SQL statement on one or multiple text lines.
3. You can perform most of the action in a database with SQL statements.
4. SQL syntax depends on relational algebra and tuple relational calculus.

SQL Statements

- SQL statements tell the database what operation you want to perform on the structured data and what information you would like to access from the database.
- The statements of SQL are very simple and easy to use and understand. They are like plain English but with a particular syntax.

Simple Example of SQL statement:

```
SELECT "column_name" FROM "table_name";
```

- Each SQL statement begins with any of the SQL keywords and ends with the semicolon (;).
- The semicolon is used in the SQL for separating the multiple SQL statements which are going to execute in the same call.
- In this SQL tutorial, we will use the semicolon (;) at the end of each SQL query or statement.

Few commonly used SQL Statements are:

1. Select Statement
2. Update Statement
3. Delete Statement
4. Create Table Statement
5. Alter Table Statement
6. Drop Table Statement
7. Create Database Statement
8. Drop Database Statement
9. Insert Into Statement
10. Truncate Table Statement
11. Describe Statement
12. Distinct Clause
13. Commit Statement
14. Rollback Statement
15. Create Index Statement
16. Drop Index Statement
17. Use Statement

Let's discuss each statement in short one by one with syntax and one example:

1. SELECT Statement

This SQL statement reads the data from the SQL database and shows it as the output to the database user.

Syntax of SELECT Statement:

```
SELECT column_name1, column_name2, ...., column_nameN  
[ FROM table_name ]  
[ WHERE condition ]  
[ ORDER BY order_column_name1 [ ASC | DESC ], .... ];
```

Example of SELECT Statement:

```
SELECT Emp_ID, First_Name, Last_Name, Salary, City  
FROM Employee_details  
WHERE Salary = 100000  
ORDER BY Last_Name
```

This example shows the Emp_ID, First_Name, Last_Name, Salary, and City of those employees from the Employee_details table whose Salary is 100000. The output shows all the specified details according to the ascending alphabetical order of Last_Name.

2. UPDATE Statement

This SQL statement changes or modifies the stored data in the SQL database.

Syntax of UPDATE Statement:

```
UPDATE table_name  
SET column_name1 = new_value_1, column_name2 = new_value_2, ...., column_nameN =  
new_value_N  
[ WHERE CONDITION ];
```

Example of UPDATE Statement:

```
UPDATE Employee_details  
SET Salary = 100000  
WHERE Emp_ID = 10;
```

This example changes the Salary of those employees of the Employee_details table whose Emp_ID is 10 in the table.

3. DELETE Statement

This SQL statement deletes the stored data from the SQL database.

Syntax of DELETE Statement:

```
DELETE FROM table_name  
[ WHERE CONDITION ];
```

Example of DELETE Statement:

```
DELETE FROM Employee_details  
WHERE First_Name = 'Sumit';
```

This example deletes the record of those employees from the Employee_details table whose First_Name is Sumit in the table.

4. CREATE TABLE Statement

This SQL statement creates the new table in the SQL database.

Syntax of CREATE TABLE Statement:

```
CREATE TABLE table_name
(
column_name1 data_type [column1 constraint(s)],
column_name2 data_type [column2 constraint(s)],
....,
.....,
column_nameN data_type [columnN constraint(s)],
PRIMARY KEY(one or more col)
);
```

Example of CREATE TABLE Statement:

```
CREATE TABLE Employee_details(
    Emp_Id NUMBER(4) NOT NULL,
    First_name VARCHAR(30),
    Last_name VARCHAR(30),
    Salary Money,
    City VARCHAR(30),
    PRIMARY KEY (Emp_Id)
);
```

- This example creates the table Employee_details with five columns or fields in the SQL database.
- The fields in the table are Emp_Id, First_Name, Last_Name, Salary, and City.
- The Emp_Id column in the table acts as a primary key, which means that the Emp_Id column cannot contain duplicate values and null values.

5. ALTER TABLE Statement

This SQL statement adds, deletes, and modifies the columns of the table in the SQL database.

Syntax of ALTER TABLE Statement:

```
ALTER TABLE table_name ADD column_name datatype[(size)];
```

The above SQL alter statement adds the column with its datatype in the existing database table.

```
ALTER TABLE table_name MODIFY column_name column_datatype[(size)];
```

The above 'SQL alter statement' renames the old column name to the new column name of the existing database table.

```
ALTER TABLE table_name DROP COLUMN column_name;
```

The above SQL alter statement deletes the column of the existing database table.

Example of ALTER TABLE Statement:

```
ALTER TABLE Employee_details
```

```
ADD Designation VARCHAR(18);
```

This example adds the new field whose name is Designation with size 18 in the Employee_details table of the SQL database.

6. DROP TABLE Statement

This SQL statement deletes or removes the table and the structure, views, permissions, and triggers associated with that table.

Syntax of DROP TABLE Statement:

```
DROP TABLE [ IF EXISTS ]  
table_name1, table_name2, ...., table_nameN;
```

The above syntax of the drop statement deletes specified tables completely if they exist in the database.

Example of DROP TABLE Statement:

```
DROP TABLE Employee_details;
```

This example drops the Employee_details table if it exists in the SQL database. This removes the complete information if available in the table.

7. CREATE DATABASE Statement

This SQL statement creates the new database in the database management system.

Syntax of CREATE DATABASE Statement:

```
CREATE DATABASE database_name;
```

Example of CREATE DATABASE Statement:

```
CREATE DATABASE Company;
```

The above example creates the company database in the system.

8. DROP DATABASE Statement

This SQL statement deletes the existing database with all the data tables and views from the database management system.

Syntax of DROP DATABASE Statement:

```
DROP DATABASE database_name;
```

Example of DROP DATABASE Statement:

```
DROP DATABASE Company;
```

The above example deletes the company database from the system.

9. INSERT INTO Statement

This SQL statement inserts the data or records in the existing table of the SQL database. This statement can easily insert single and multiple records in a single query statement.

Syntax of insert a single record:

```
INSERT INTO table_name
```

```
(  
column_name1,  
column_name2, ....,  
column_nameN  
)  
VALUES  
(value_1,  
value_2, .....,  
value_N  
);
```

Example of insert a single record:

```
INSERT INTO Employee_details
(
Emp_ID,
First_name,
Last_name,
Salary,
City
)
VALUES
(101,
Akhil,
Sharma,
40000,
Bangalore
);
```

This example inserts 101 in the first column, Akhil in the second column, Sharma in the third column, 40000 in the fourth column, and Bangalore in the last column of the table Employee_details.

Syntax of inserting a multiple records in a single query:

```
INSERT INTO table_name
( column_name1, column_name2, ...., column_nameN)
VALUES (value_1, value_2, ..... , value_N), (value_1, value_2, ..... , value_N),....;
```

Example of inserting multiple records in a single query:

```
INSERT INTO Employee_details
( Emp_ID, First_name, Last_name, Salary, City )
VALUES (101, Amit, Gupta, 50000, Mumbai), (101, John, Aggarwal, 45000, Calcutta), (101, Sidhu, Arora, 55000, Mumbai);
```

This example inserts the records of three employees in the Employee_details table in the single query statement.

10. TRUNCATE TABLE Statement

This SQL statement deletes all the stored records from the table of the SQL database.

Syntax of TRUNCATE TABLE Statement:

```
TRUNCATE TABLE table_name;
```

Example of TRUNCATE TABLE Statement:

```
TRUNCATE TABLE Employee_details;
```

This example deletes the record of all employees from the Employee_details table of the database.

11. DESCRIBE Statement

This SQL statement tells something about the specified table or view in the query.

Syntax of DESCRIBE Statement:

```
DESCRIBE table_name | view_name;
```

Example of DESCRIBE Statement:

```
DESCRIBE Employee_details;
```

This example explains the structure and other details about the Employee_details table.

12. DISTINCT Clause

This SQL statement shows the distinct values from the specified columns of the database table. This statement is used with the SELECT keyword.

Syntax of DISTINCT Clause:

```
SELECT DISTINCT column_name1, column_name2, ...
FROM table_name;
```

Example of DISTINCT Clause:

```
SELECT DISTINCT City, Salary
FROM Employee_details;
```

This example shows the distinct values of the City and Salary column from the Employee_details table.

13. COMMIT Statement

This SQL statement saves the changes permanently, which are done in the transaction of the SQL database.

Syntax of COMMIT Statement:

```
COMMIT
```

Example of COMMIT Statement:

```
DELETE FROM Employee_details
WHERE salary = 30000;
COMMIT;
```

This example deletes the records of those employees whose Salary is 30000 and then saves the changes permanently in the database.

14. ROLLBACK Statement

This SQL statement undo the transactions and operations which are not yet saved to the SQL database.

Syntax of ROLLBACK Statement:

```
ROLLBACK
```

Example of ROLLBACK Statement:

```
DELETE FROM Employee_details
WHERE City = Mumbai;
ROLLBACK;
```

This example deletes the records of those employees whose City is Mumbai and then undo the changes in the database.

15. CREATE INDEX Statement

This SQL statement creates the new index in the SQL database table.

Syntax of CREATE INDEX Statement:

```
CREATE INDEX index_name  
ON table_name ( column_name1, column_name2, ..., column_nameN );
```

Example of CREATE INDEX Statement:

```
CREATE INDEX idx_First_Name  
ON employee_details (First_Name);
```

This example creates an index idx_First_Name on the First_Name column of the Employee_Details table.

16. DROP INDEX Statement

This SQL statement deletes the existing index of the SQL database table.

Syntax of DROP INDEX Statement:

```
DROP INDEX index_name;
```

Example of DROP INDEX Statement:

```
DROP INDEX idx_First_Name;
```

This example deletes the index idx_First_Name from the SQL database.

17. USE Statement

This SQL statement selects the existing SQL database. Before performing the operations on the database table, you have to select the database from the multiple existing databases.

Syntax of USE Statement:

```
USE database_name;
```

Example of USE DATABASE Statement:

```
USE Company;
```

5.4 SQL Data Types

- Data types are used to represent the nature of the data that can be stored in the database table.
- For example, in a particular column of a table, if we want to store a string type of data then we will have to declare a string data type of this column.
- Data types mainly classified into three categories for every database.
 - String Data types
 - Numeric Data types
 - Date and time Data types

MySQL Data Types

A list of data types used in MySQL database. This is based on MySQL 8.0.

MySQL String Data Types

CHAR(Size)	It is used to specify a fixed length string that can contain numbers, letters, and special characters. Its size can be 0 to 255 characters. Default is 1.
VARCHAR(Size)	It is used to specify a variable length string that can contain numbers, letters, and special characters. Its size can be from 0 to 65535 characters.
BINARY(Size)	It is equal to CHAR() but stores binary byte strings. Its size parameter specifies the column length in the bytes. Default is 1.
VARBINARY(Size)	It is equal to VARCHAR() but stores binary byte strings. Its size parameter specifies the maximum column length in bytes.
TEXT(Size)	It holds a string that can contain a maximum length of 255 characters.
TINYTEXT	It holds a string with a maximum length of 255 characters.
MEDIUMTEXT	It holds a string with a maximum length of 16,777,215.
LONGTEXT	It holds a string with a maximum length of 4,294,967,295 characters.
ENUM(val1, val2, val3,...)	It is used when a string object having only one value, chosen from a list of possible values. It contains 65535 values in an ENUM list. If you insert a value that is not in the list, a blank value will be inserted.
SET(val1,val2,val3,...)	It is used to specify a string that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values at one time in a SET list.
BLOB(size)	It is used for BLOBs (Binary Large Objects). It can hold up to 65,535 bytes.

MySQL Numeric Data Types

BIT(Size)	It is used for a bit-value type. The number of bits per value is specified in size. Its size can be 1 to 64. The default value is 1.
INT(size)	It is used for the integer value. Its signed range varies from -2147483648 to 2147483647 and unsigned range varies from 0 to 4294967295. The size parameter specifies the max display width that is 255.
INTEGER(size)	It is equal to INT(size).
FLOAT(size, d)	It is used to specify a floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal point is specified by d parameter.
FLOAT(p)	It is used to specify a floating point number. MySQL used p parameter to determine whether to use FLOAT or DOUBLE. If p is between 0 to 24, the data type becomes FLOAT(). If p is from 25 to 53, the data type becomes DOUBLE().
DOUBLE(size, d)	It is a normal size floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal is specified by d parameter.
DECIMAL(size, d)	It is used to specify a fixed point number. Its size parameter specifies the total number of digits. The number of digits after the decimal parameter is specified by d parameter. The maximum value for the size is 65, and the default value is 10. The maximum value for d is 30, and the default value is 0.
DEC(size, d)	It is equal to DECIMAL(size, d).
BOOL	It is used to specify Boolean values true and false. Zero is considered as false, and nonzero values are considered as true.

MySQL Date and Time Data Types

DATE	It is used to specify date format YYYY-MM-DD. Its supported range is from '1000-01-01' to '9999-12-31'.
DATETIME(fsp)	It is used to specify date and time combination. Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1000-01-01 00:00:00' to 9999-12-31 23:59:59'.
TIMESTAMP(fsp)	It is used to specify the timestamp. Its value is stored as the number of seconds since the Unix epoch('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	It is used to specify the time format. Its format is hh:mm:ss. Its supported range is from '-838:59:59' to '838:59:59'
YEAR	It is used to specify a year in four-digit format. Values allowed in four digit format from 1901 to 2155, and 0000.

SQL Server Data Types

SQL Server String Data Type

char(n)	It is a fixed width character string data type. Its size can be up to 8000 characters.
varchar(n)	It is a variable width character string data type. Its size can be up to 8000 characters.
varchar(max)	It is a variable width character string data types. Its size can be up to 1,073,741,824 characters.
text	It is a variable width character string data type. Its size can be up to 2GB of text data.
nchar	It is a fixed width Unicode string data type. Its size can be up to 4000 characters.
nvarchar	It is a variable width Unicode string data type. Its size can be up to 4000 characters.
ntext	It is a variable width Unicode string data type. Its size can be up to 2GB of text data.
binary(n)	It is a fixed width Binary string data type. Its size can be up to 8000 bytes.
varbinary	It is a variable width Binary string data type. Its size can be up to 8000 bytes.
image	It is also a variable width Binary string data type. Its size can be up to 2GB.

SQL Server Numeric Data Types

bit	It is an integer that can be 0, 1 or null.
tinyint	It allows whole numbers from 0 to 255.
Smallint	It allows whole numbers between -32,768 and 32,767.
Int	It allows whole numbers between -2,147,483,648 and 2,147,483,647.
bigint	It allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807.
float(n)	It is used to specify floating precision number data from -1.79E+308 to 1.79E+308. The n parameter indicates whether the field should hold the 4 or 8 bytes. Default value of n is 53.
real	It is a floating precision number data from -3.40E+38 to 3.40E+38.
money	It is used to specify monetary data from -922,337,233,685,477.5808 to 922,337,203,685,477.5807.

SQL Server Date and Time Data Type

datetime	It is used to specify date and time combination. It supports range from January 1, 1753, to December 31, 9999 with an accuracy of 3.33 milliseconds.
datetime2	It is used to specify date and time combination. It supports range from January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds
date	It is used to store date only. It supports range from January 1, 0001 to December 31, 9999
time	It stores time only to an accuracy of 100 nanoseconds
timestamp	It stores a unique number when a new row gets created or modified. The time stamp value is based upon an internal clock and does not correspond to real time. Each table may contain only one-time stamp variable.

SQL Server Other Data Types

Sql_variant	It is used for various data types except for text, timestamp, and ntext. It stores up to 8000 bytes of data.
XML	It stores XML formatted data. Maximum 2GB.
cursor	It stores a reference to a cursor used for database operations.
table	It stores result set for later processing.
uniqueidentifier	It stores GUID (Globally unique identifier).

Oracle Data Types

Oracle String data types

CHAR(size)	It is used to store character data within the predefined length. It can be stored up to 2000 bytes.
NCHAR(size)	It is used to store national character data within the predefined length. It can be stored up to 2000 bytes.
VARCHAR2(size)	It is used to store variable string data within the predefined length. It can be stored up to 4000 byte.
VARCHAR(SIZE)	It is the same as VARCHAR2(size). You can also use VARCHAR(size), but it is suggested to use VARCHAR2(size)
NVARCHAR2(size)	It is used to store Unicode string data within the predefined length. We have to must specify the size of NVARCHAR2 data type. It can be stored up to 4000 bytes.

Oracle Numeric Data Types

NUMBER(p, s)	It contains precision p and scale s. The precision p can range from 1 to 38, and the scale s can range from -84 to 127.
FLOAT(p)	It is a subtype of the NUMBER data type. The precision p can range from 1 to 126.
BINARY_FLOAT	It is used for binary precision(32-bit). It requires 5 bytes, including length byte.
BINARY_DOUBLE	It is used for double binary precision (64-bit). It requires 9 bytes, including length byte.

Oracle Date and Time Data Types

DATE	It is used to store a valid date-time format with a fixed length. Its range varies from January 1, 4712 BC to December 31, 9999 AD.
TIMESTAMP	It is used to store the valid date in YYYY-MM-DD with time hh:mm:ss format.

Oracle Large Object Data Types (LOB Types)

BLOB	It is used to specify unstructured binary data. Its range goes up to $2^{32}-1$ bytes or 4 GB.
BFILE	It is used to store binary data in an external file. Its range goes up to $2^{32}-1$ bytes or 4 GB.
CLOB	It is used for single-byte character data. Its range goes up to $2^{32}-1$ bytes or 4 GB.
NCLOB	It is used to specify single byte or fixed length multibyte national character set (NCHAR) data. Its range is up to $2^{32}-1$ bytes or 4 GB.
RAW(size)	It is used to specify variable length raw binary data. Its range is up to 2000 bytes per row. Its maximum size must be specified.
LONG RAW	It is used to specify variable length raw binary data. Its range up to $2^{31}-1$ bytes or 2 GB, per row.

5.5 SQL Operators

- Every database administrator and user uses SQL queries for manipulating and accessing the data of database tables and views.
- The manipulation and retrieving of the data are performed with the help of reserved words and characters, which are used to perform arithmetic operations, logical operations, comparison operations, compound operations, etc.

What is SQL Operator?

- The SQL reserved words and characters are called operators, which are used with a WHERE clause in a SQL query.
- In SQL, an operator can either be a unary or binary operator.
- The unary operator uses only one operand for performing the unary operation, whereas the binary operator uses two operands for performing the binary operation.

Syntax of Unary SQL Operator

Operator SQL_Operand

Syntax of Unary SQL Operator

Operand1 SQL_Operator Operand2

What is the Precedence of SQL Operator?

- The precedence of SQL operators is the sequence in which the SQL evaluates the different operators in the same expression.
- Structured Query Language evaluates those operators first, which have high precedence.
- In the following table, the operators at the top have high precedence, and the operators that appear at the bottom have low precedence.

SQL Operator Symbols	Operators
**	Exponentiation operator
+, -	Identity operator, Negation operator
*, /	Multiplication operator, Division operator
+, -,	Addition (plus) operator, subtraction (minus) operator, String Concatenation operator
=, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN	Comparison Operators
NOT	Logical negation operator
&& or AND	Conjunction operator
OR	Inclusion operator

For Example,

1. UPDATE employee
2. SET salary = 20 - 3 * 5 WHERE Emp_Id = 5;

In the above SQL example, salary is assigned 5, not 85, because the * (Multiplication) Operator has higher precedence than the - (subtraction) operator, so it first gets multiplied with $3*5$ and then subtracts from 20.

Types of Operators

SQL operators are categorized in the following categories:

1. SQL Arithmetic Operators
2. SQL Comparison Operators
3. SQL Logical Operators
4. SQL Set Operators
5. SQL Bit-wise Operators
6. SQL Unary Operators

Let's discuss each operator with their types.

SQL Arithmetic Operators

- The **Arithmetic Operators** perform the mathematical operation on the numerical data of the SQL tables.
- These operators perform addition, subtraction, multiplication, and division operations on the numerical operands.

Following are the various arithmetic operators performed on the SQL data:

1. SQL Addition Operator
2. SQL Subtraction Operator
3. SQL Multiplication Operator
4. SQL Division Operator
5. SQL Modulus Operator

SQL Addition Operator (+)

- The **Addition Operator** in SQL performs the addition on the numerical data of the database table.
- In SQL, we can easily add the numerical values of two columns of the same table by specifying both the column names as the first and second operand.
- We can also add the numbers to the existing numbers of the specific column.

Syntax of SQL Addition Operator:

`SELECT operand1 + operand2;`

Let's understand the below example which explains how to execute Addition Operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_Monthlybonus**.

Emp Id	Emp Name	Emp Salary	Emp Monthlybonus
101	Tushar	25000	4000
102	Anuj	30000	200

- Suppose, we want to add **20,000** to the salary of each employee specified in the table. Then, we have to write the following query in the SQL:

`SELECT Emp_Salary + 20000 as Emp_New_Salary FROM Employee_details;`

In this query, we have performed the SQL addition operation on the single column of the given table.

- Suppose, we want to add the Salary and monthly bonus columns of the above table, then we have to write the following query in SQL:

`SELECT Emp_Salary + Emp_Monthlybonus as Emp_Total_Salary FROM Employee_details;`

In this query, we have added two columns with each other of the above table.

SQL Subtraction Operator (-)

- The Subtraction Operator in SQL performs the subtraction on the numerical data of the database table.
- In SQL, we can easily subtract the numerical values of two columns of the same table by specifying both the column names as the first and second operand.
- We can also subtract the number from the existing number of the specific table column.

Syntax of SQL Subtraction Operator:

```
SELECT operand1 – operand2;
```

Let's understand the below example which explains how to execute Subtraction Operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_Penalty**.

Emp Id	Emp Name	Emp Salary	Penalty
201	Abhay	25000	200
202	Sumit	30000	500

- Suppose we want to subtract 5,000 from the salary of each employee given in the **Employee_details** table. Then, we have to write the following query in the SQL:

```
SELECT Emp_Salary - 5000 as Emp_New_Salary FROM Employee_details;
```

In this query, we have performed the SQL subtraction operation on the single column of the given table.

- If we want to subtract the penalty from the salary of each employee, then we have to write the following query in SQL:

```
SELECT Emp_Salary - Penalty as Emp_Total_Salary FROM Employee_details;
```

SQL Multiplication Operator (*)

- The Multiplication Operator in SQL performs the Multiplication on the numerical data of the database table.
- In SQL, we can easily multiply the numerical values of two columns of the same table by specifying both the column names as the first and second operand.

Syntax of SQL Multiplication Operator:

```
SELECT operand1 * operand2;
```

Let's understand the below example which explains how to execute Multiplication Operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_Penalty**.

Emp Id	Emp Name	Emp Salary	Penalty
201	Abhay	25000	200
202	Sumit	30000	500

- Suppose, we want to double the salary of each employee given in the **Employee_details** table. Then, we have to write the following query in the SQL:

```
SELECT Emp_Salary * 2 as Emp_New_Salary FROM Employee_details;
```

In this query, we have performed the SQL multiplication operation on the single column of the given table.

- If we want to multiply the **Emp_Id** column to **Emp_Salary** column of that employee whose **Emp_Id** is **202**, then we have to write the following query in SQL:

```
SELECT Emp_Id * Emp_Salary as Emp_Id * Emp_Salary FROM Employee_details WHERE Emp_Id = 202;
```

In this query, we have multiplied the values of two columns by using the WHERE clause.

SQL Division Operator (/)

The Division Operator in SQL divides the operand on the left side by the operand on the right side.

Syntax of SQL Division Operator:

```
SELECT operand1 / operand2;
```

- In SQL, we can also divide the numerical values of one column by another column of the same table by specifying both column names as the first and second operand.
 - We can also perform the division operation on the stored numbers in the column of the SQL table.
- Let's understand the below example which explains how to execute Division Operator in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id**, **Emp_Name**, and **Emp_Salary**.

Emp Id	Emp Name	Emp Salary
201	Abhay	25000
202	Sumit	30000

- Suppose, we want to half the salary of each employee given in the **Employee_details** table. For this operation, we have to write the following query in the SQL:

```
SELECT Emp_Salary / 2 as Emp_New_Salary FROM Employee_details;
```

In this query, we have performed the SQL division operation on the single column of the given table.

SQL Modulus Operator (%)

The Modulus Operator in SQL provides the remainder when the operand on the left side is divided by the operand on the right side.

Syntax of SQL Modulus Operator:

```
SELECT operand1 % operand2;
```

Let's understand the below example which explains how to execute Modulus Operator in SQL query:

This example consists of a **Division** table, which has three columns Number, First_operand, and Second_operand.

Number	First operand	Second operand
1	56	4
2	32	8
3	89	9
4	18	10
5	10	5

- If we want to get the remainder by dividing the numbers of First_operand column by the numbers of Second_operand column, then we have to write the following query in SQL:

```
SELECT First_operand % Second_operand as Remainder FROM Employee_details;
```

SQL Comparison Operators

The **Comparison Operators** in SQL compare two different data of SQL table and check whether they are the same, greater, and lesser. The SQL comparison operators are used with the WHERE clause in the SQL queries.

Following are the various comparison operators which are performed on the data stored in the SQL database tables:

1. SQL Equal Operator (=)
2. SQL Not Equal Operator (!=)
3. SQL Greater Than Operator (>)
4. SQL Greater Than Equals to Operator (>=)
5. SQL Less Than Operator (<)
6. SQL Less Than Equals to Operator (<=)

SQL Equal Operator (=)

- This operator is highly used in SQL queries. The **Equal Operator** in SQL shows only data that matches the specified value in the query.
- This operator returns TRUE records from the database table if the value of both operands specified in the query is matched.

Let's understand the below example which explains how to execute Equal Operator in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id**, **Emp_Name**, and **Emp_Salary**.

Emp Id	Emp Name	Emp Salary
201	Abhay	30000
202	Ankit	40000
203	Bheem	30000
204	Ram	29000
205	Sumit	30000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose salary is 30000. Then, we have to write the following query in the SQL database:

```
SELECT * FROM Employee_details WHERE Emp_Salary = 30000;
```

In this example, we used the SQL equal operator with WHERE clause for getting the records of those employees whose salary is 30000.

SQL Equal Not Operator (!=)

- The **Equal Not Operator** in SQL shows only those data that do not match the query's specified value.
- This operator returns those records or rows from the database views and tables if the value of both operands specified in the query is not matched with each other.

Let's understand the below example which explains how to execute Equal Not Operator in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id**, **Emp_Name**, and **Emp_Salary**.

Emp Id	Emp Name	Emp Salary
201	Abhay	45000
202	Ankit	45000
203	Bheem	30000
204	Ram	29000
205	Sumit	29000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose salary is not 45000. Then, we have to write the following query in the SQL database:

```
SELECT * FROM Employee_details WHERE Emp_Salary != 45000;
```

In this example, we used the SQL equal not operator with WHERE clause for getting the records of those employees whose salary is not 45000.

SQL Greater Than Operator (>)

- The **Greater Than Operator** in SQL shows only those data which are greater than the value of the right-hand operand.

Let's understand the below example which explains how to execute Greater Than Operator (>) in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id**, **Emp_Name**, and **Emp_Salary**.

Emp Id	Emp Name	Emp Salary
201	Abhay	45000
202	Ankit	45000
203	Bheem	30000
204	Ram	29000
205	Sumit	29000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose employee id is greater than 202. Then, we have to write the following query in the SQL database:

```
SELECT * FROM Employee_details WHERE Emp_Id > 202;
```

Here, SQL greater than operator displays the records of those employees from the above table whose Employee Id is greater than 202.

SQL Greater Than Equals to Operator (>=)

The **Greater Than Equals to Operator** in SQL shows those data from the table which are greater than and equal to the value of the right-hand operand.

Let's understand the below example which explains how to execute greater than equals to the operator (>=) in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id**, **Emp_Name**, and **Emp_Salary**.

Emp Id	Emp Name	Emp Salary
201	Abhay	45000
202	Ankit	45000
203	Bheem	30000
204	Ram	29000
205	Sumit	29000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose employee id is greater than and equals to 202. For this, we have to write the following query in the SQL database:

```
SELECT * FROM Employee_details WHERE Emp_Id >= 202;
```

Here, 'SQL greater than equals to operator' with WHERE clause displays the rows of those employees from the table whose Employee Id is greater than and equals to 202.

SQL Less Than Operator (<)

- The **Less Than Operator** in SQL shows only those data from the database tables which are less than the value of the right-side operand.
- This comparison operator checks that the left side operand is lesser than the right-side operand.
- If the condition becomes true, then this operator in SQL displays the data which is less than the value of the right-side operand.

Let's understand the below example which explains how to execute less than operator (<) in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id**, **Emp_Name**, and **Emp_Salary**.

Emp Id	Emp Name	Emp Salary
201	Abhay	45000
202	Ankit	45000
203	Bheem	30000
204	Ram	29000
205	Sumit	29000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose employee id is less than 204. For this, we have to write the following query in the SQL database:

```
SELECT * FROM Employee_details WHERE Emp_Id < 204;
```

Here, **SQL less than operator** with WHERE clause displays the records of those employees from the above table whose Employee Id is less than 204.

SQL Less Than Equals to Operator (\leq)

- The **Less Than Equals to Operator** in SQL shows those data from the table which are lesser and equal to the value of the right-side operand.
- This comparison operator checks that the left side operand is lesser and equal to the right-side operand.

Let's understand the below example which explains how to execute less than equals to the operator (\leq) in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id**, **Emp_Name**, and **Emp_Salary**.

Emp Id	Emp Name	Emp Salary
201	Abhay	45000
202	Ankit	45000
203	Bheem	30000
204	Ram	29000
205	Sumit	29000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose employee id is less and equals **203**. For this, we have to write the following query in the SQL database:

```
SELECT * FROM Employee_details WHERE Emp_Id <= 203;
```

Here, **SQL less than equals to the operator** with WHERE clause displays the rows of those employees from the table whose Employee Id is less than and equals 202.

SQL Logical Operators

- The **Logical Operators** in SQL perform the Boolean operations, which give two results **True and False**. These operators provide **True** value if both operands match the logical condition.

Following are the various logical operators which are performed on the data stored in the SQL database tables:

1. ALL operator
2. AND operator
3. OR operator
4. BETWEEN operator
5. IN operator
6. NOT operator
7. ANY operator
8. LIKE operator

ALL Operator

- The ALL operator in SQL compares the specified value to all the values of a column from the subquery in the SQL database.
- This operator is always used with the following statement:

1. SELECT
2. HAVING
3. WHERE

Syntax of ALL operator:

`SELECT column_Name1, ..., column_NameN FROM table_Name WHERE column Comparison_operator ALL (SELECT column FROM tablename2)`

Let's understand the below example which explains how to execute ALL logical operators in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Gurgaon
202	Ankit	45000	Delhi
203	Bheem	30000	Jaipur
204	Ram	29000	Mumbai
205	Sumit	40000	Kolkata

- If we want to access the employee id and employee names of those employees from the table whose salaries are greater than the salary of employees who lives in Jaipur city, then we have to type the following query in SQL.

`SELECT Emp_Id, Emp_Name FROM Employee_details WHERE Emp_Salary > ALL (SELECT Emp_Salary FROM Employee_details WHERE Emp_City = Jaipur)`

Here, we used the **SQL ALL operator** with greater than the operator.

AND Operator

- The **AND operator** in SQL would show the record from the database table if all the conditions separated by the AND operator evaluated to True.
- It is also known as the conjunctive operator and is used with the WHERE clause.

Syntax of AND operator:

`SELECT column1, ..., columnN FROM table_Name WHERE condition1 AND condition2 AND condition3 AND AND conditionN;`

Let's understand the below example which explains how to execute AND logical operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Delhi
202	Ankit	45000	Chandigarh
203	Bheem	30000	Delhi
204	Ram	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose salary is 25000 and the city is Delhi. For this, we have to write the following query in SQL:

```
SELECT * FROM Employee_details WHERE Emp_Salary = 25000 OR Emp_City = 'Delhi';
```

Here, **SQL AND operator** with WHERE clause shows the record of employees whose salary is 25000 and the city is Delhi.

OR Operator

- The OR operator in SQL shows the record from the table if any of the conditions separated by the OR operator evaluates to True.
- It is also known as the conjunctive operator and is used with the WHERE clause.

Syntax of OR operator:

SELECT column1,, columnN FROM table_Name WHERE condition1 OR condition2 OR condition3 OR OR conditionN;

Let's understand the below example which explains how to execute OR logical operator in SQL query: This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Delhi
202	Ankit	45000	Chandigarh
203	Bheem	30000	Delhi
204	Ram	25000	Delhi
205	Sumit	40000	Kolkata

- If we want to access all the records of those employees from the **Employee_details** table whose salary is 25000 or the city is Delhi. For this, we have to write the following query in SQL:

SELECT * FROM Employee_details WHERE Emp_Salary = 25000 OR Emp_City = 'Delhi';
 Here, **SQL OR operator** with WHERE clause shows the record of employees whose salary is 25000 or the city is Delhi.

BETWEEN Operator

- The **BETWEEN operator** in SQL shows the record within the range mentioned in the SQL query. This operator operates on the numbers, characters, and date/time operands.
- If there is no value in the given range, then this operator shows NULL value.

Syntax of BETWEEN operator:

```
SELECT column_Name1, column_Name2 ...., column_NameN FROM table_Name WHERE column_name BETWEEN value1 and value2;
```

Let's understand the below example which explains how to execute BETWEEN logical operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Delhi
202	Ankit	45000	Chandigarh
203	Bheem	30000	Delhi
204	Ram	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to access all the information of those employees from the **Employee_details** table who is having salaries between 20000 and 40000. For this, we have to write the following query in SQL:

```
SELECT * FROM Employee_details WHERE Emp_Salary BETWEEN 30000 AND 45000;
```

Here, we used the **SQL BETWEEN operator** with the Emp_Salary field.

IN Operator

- The **IN operator** in SQL allows database users to specify two or more values in a WHERE clause. This logical operator minimizes the requirement of multiple OR conditions.
- This operator makes the query easier to learn and understand. This operator returns those rows whose values match with any value of the given list.

Syntax of IN operator:

```
SELECT column_Name1, column_Name2 ...., column_NameN FROM table_Name WHERE column_name IN (list_of_values);
```

Let's understand the below example which explains how to execute IN logical operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Delhi
202	Ankit	45000	Chandigarh
203	Bheem	30000	Delhi
204	Ram	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to show all the information of those employees from the **Employee_details** table whose **Employee Id** is 202, 204, and 205. For this, we have to write the following query in SQL:

```
SELECT * FROM Employee_details WHERE Emp_Id IN (202, 204, 205);
```

Here, we used the **SQL IN operator** with the Emp_Id column.

- Suppose, we want to show all the information of those employees from the **Employee_details** table whose **Employee Id** is not equal to 202 and 205. For this, we have to write the following query in SQL:

```
SELECT * FROM Employee_details WHERE Emp_Id NOT IN (202,205);
```

Here, we used the **SQL NOT IN operator** with the Emp_Id column.

NOT Operator

- The **NOT operator** in SQL shows the record from the table if the condition evaluates to false. It is always used with the WHERE clause.

Syntax of NOT operator:

```
SELECT column1, column2 ...., columnN FROM table_Name WHERE NOT condition;
```

Let's understand the below example which explains how to execute NOT logical operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Delhi
202	Ankit	45000	Chandigarh
203	Bheem	30000	Delhi
204	Ram	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to show all the information of those employees from the **Employee_details** table whose City is not Delhi. For this, we have to write the following query in SQL:

```
SELECT * FROM Employee_details WHERE NOT Emp_City = 'Delhi' ;
```

In this example, we used the **SQL NOT operator** with the Emp_City column.

- Suppose, we want to show all the information of those employees from the **Employee_details** table whose City is not Delhi and Chandigarh. For this, we have to write the following query in SQL:

```
SELECT * FROM Employee_details WHERE NOT Emp_City = 'Delhi' AND NOT Emp_City = 'Chandigarh';
```

In this example, we used the **SQL NOT operator** with the Emp_City column.

ANY Operator

- The **ANY operator** in SQL shows the records when any of the values returned by the sub-query meet the condition.
- The ANY logical operator must match at least one record in the inner query and must be preceded by any SQL comparison operator.

Syntax of ANY operator:

```
SELECT column1, column2 ...., columnN FROM table_Name WHERE column_name comparison_operator ANY ( SELECT column_name FROM table_name WHERE condition(s) ) ;
```

LIKE Operator

- The **LIKE operator** in SQL shows those records from the table which match with the given pattern specified in the sub-query.
- The percentage (%) sign is a wildcard which is used in conjunction with this logical operator.
- This operator is used in the WHERE clause with the following three statements:
 1. SELECT statement
 2. UPDATE statement
 3. DELETE statement

Syntax of LIKE operator:

```
SELECT column_Name1, column_Name2 ...., column_NameN FROM table_Name WHERE column_name LIKE pattern;
```

Let's understand the below example which explains how to execute LIKE logical operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Chandigarh
203	Saket	30000	Delhi
204	Abhay	25000	Delhi
205	Sumit	40000	Kolkata

- If we want to show all the information of those employees from the **Employee_details** whose name starts with "s". For this, we have to write the following query in SQL:

```
SELECT * FROM Employee_details WHERE Emp_Name LIKE 's%';
```

In this example, we used the SQL LIKE operator with **Emp_Name** column because we want to access the record of those employees whose name starts with s.

- If we want to show all the information of those employees from the **Employee_details** whose name ends with "y". For this, we have to write the following query in SQL:

```
SELECT * FROM Employee_details WHERE Emp_Name LIKE '%y';
```

- If we want to show all the information of those employees from the **Employee_details** whose name starts with "S" and ends with "y". For this, we have to write the following query in SQL:

```
SELECT * FROM Employee_details WHERE Emp_Name LIKE 'S%y';
```

SQL Set Operators

- The **Set Operators** in SQL combine a similar type of data from two or more SQL database tables. It mixes the result, which is extracted from two or more SQL queries, into a single result.
- Set operators combine more than one SELECT statement in a single query and return a specific result set.

Following are the various set operators which are performed on the similar data stored in the two SQL database tables:

1. Union Operator
2. Union ALL Operator
3. Intersect Operator
4. Minus Operator

Union Operator

- The SQL Union Operator combines the result of two or more SELECT statements and provides the single output.
- The data type and the number of columns must be the same for each SELECT statement used with the UNION operator. This operator does not show the duplicate records in the output table.

Syntax of UNION Set operator:

```
SELECT column1, column2 ...., columnN FROM table_Name1 [WHERE conditions]
```

UNION

```
SELECT column1, column2 ...., columnN FROM table_Name2 [WHERE conditions];
```

Let's understand the below example which explains how to execute Union operator in Structured Query Language:

In this example, **we used two tables**. Both tables have four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Delhi
203	Saket	30000	Aligarh

Table: Employee_details1

Emp Id	Emp Name	Emp Salary	Emp City
203	Saket	30000	Aligarh
204	Saurabh	40000	Delhi
205	Ram	30000	Kerala
201	Sanjay	25000	Delhi

Table: Employee_details2

- Suppose, we want to see the employee name and employee id of each employee from both tables in a single output. For this, we have to write the following query in SQL:

```
SELECT Emp_ID, Emp_Name FROM Employee_details1
```

UNION

```
SELECT Emp_ID, Emp_Name FROM Employee_details2 ;
```

Union ALL Operator

- The SQL Union Operator is the same as the UNION operator, but the only difference is that it also shows the same record.

Syntax of UNION ALL Set operator:

```
SELECT column1, column2 ...., columnN FROM table_Name1 [WHERE conditions]
```

UNION ALL

```
SELECT column1, column2 ...., columnN FROM table_Name2 [WHERE conditions];
```

Let's understand the below example which explains how to execute Union ALL operator in Structured Query Language:

In this example, we used two tables. Both tables have four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Delhi
203	Saket	30000	Aligarh

Table: Employee_details1

Emp Id	Emp Name	Emp Salary	Emp City
203	Saket	30000	Aligarh
204	Saurabh	40000	Delhi
205	Ram	30000	Kerala
201	Sanjay	25000	Delhi

Table: Employee_details2

- If we want to see the employee name of each employee of both tables in a single output. For this, we have to write the following query in SQL:

```
SELECT Emp_Name FROM Employee_details1
```

UNION ALL

```
SELECT Emp_Name FROM Employee_details2 ;
```

Intersect Operator

- The SQL Intersect Operator shows the common record from two or more SELECT statements.
- The data type and the number of columns must be the same for each SELECT statement used with the INTERSECT operator.

Syntax of INTERSECT Set operator:

SELECT column1, column2, columnN FROM table_Name1 [WHERE conditions]

INTERSECT

SELECT column1, column2, columnN FROM table_Name2 [WHERE conditions];

Let's understand the below example which explains how to execute INTERSECT operator in Structured Query Language:

In this example, **we used two tables**. Both tables have four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Delhi
203	Saket	30000	Aligarh

Table: Employee_details1

Emp Id	Emp Name	Emp Salary	Emp City
203	Saket	30000	Aligarh
204	Saurabh	40000	Delhi
205	Ram	30000	Kerala
201	Sanjay	25000	Delhi

Table: Employee_details2

Suppose, we want to see a common record of the employee from both the tables in a single output.

For this, we have to write the following query in SQL:

SELECT Emp_Name FROM Employee_details1

INTERSECT

SELECT Emp_Name FROM Employee_details2 ;

Minus Operator

- The SQL Minus Operator combines the result of two or more SELECT statements and shows only the results from the first data set.

Syntax of MINUS operator:

```
SELECT column1, column2 ...., columnN FROM First_tablename [WHERE conditions]
```

MINUS

```
SELECT column1, column2 ...., columnN FROM Second_tablename [WHERE conditions];
```

Let's understand the below example which explains how to execute INTERSECT operator in Structured Query Language:

In this example, **we used two tables**. Both tables have four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Delhi
203	Saket	30000	Aligarh

Table: Employee_details1

Emp Id	Emp Name	Emp Salary	Emp City
203	Saket	30000	Aligarh
204	Saurabh	40000	Delhi
205	Ram	30000	Kerala
201	Sanjay	25000	Delhi

Table: Employee_details2

- Suppose, we want to see the name of employees from the first result set after the combination of both tables. For this, we have to write the following query in SQL:

```
SELECT Emp_Name FROM Employee_details1
```

MINUS

```
SELECT Emp_Name FROM Employee_details2 ;
```

SQL Unary Operators

- The Unary Operators in SQL perform the unary operations on the single data of the SQL table, i.e., these operators operate only on one operand.
- These types of operators can be easily operated on the numeric data value of the SQL table.
- Following are the various unary operators which are performed on the numeric data stored in the SQL table:

 1. Unary Positive Operator
 2. Unary Negative Operator

Unary Positive Operator

- The SQL Positive (+) operator makes the numeric value of the SQL table positive.

Syntax of Unary Positive Operator

```
SELECT +(column1), +(column2) ...., +(columnN) FROM table_Name [WHERE conditions] ;
```

Let's understand the below example which explains how to execute a Positive unary operator on the data of SQL table:

This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Chandigarh
203	Saket	30000	Delhi
204	Abhay	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to see the salary of each employee as positive from the Employee details table.
- For this, we have to write the following query in SQL:

```
SELECT +Emp_Salary Employee_details ;
```

Unary Negative Operator

- The SQL Negative (-) operator makes the numeric value of the SQL table negative.

Syntax of Unary Negative Operator

```
SELECT -(column_Name1), -(column_Name2) ...., -
(column_NameN) FROM table_Name [WHERE conditions] ;
```

Let's understand the below example which explains how to execute Negative unary operator on the data of SQL table:

This example consists of an **Employee_details** table, which has four columns **Emp_Id**, **Emp_Name**, **Emp_Salary**, and **Emp_City**.

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Chandigarh
203	Saket	30000	Delhi
204	Abhay	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to see the salary of each employee as negative from the Employee_details table. For this, we have to write the following query in SQL:

```
SELECT -Emp_Salary Employee_details ;
```

- Suppose, we want to see the salary of those employees as negative whose city is Kolkata in the Employee_details table. For this, we have to write the following query in SQL:

```
SELECT -Emp_Salary Employee_details WHERE Emp_City = 'Kolkata';
```

SQL Bitwise Operators

- The Bitwise Operators in SQL perform the bit operations on the Integer values.
- To understand the performance of Bitwise operators, you just knew the basics of Boolean algebra.
- Following are the two important logical operators which are performed on the data stored in the SQL database tables:
 1. Bitwise AND (&)
 2. Bitwise OR(||)

Bitwise AND (&)

- The Bitwise AND operator performs the logical AND operation on the given Integer values.
- This operator checks each bit of a value with the corresponding bit of another value.

Syntax of Bitwise AND Operator

`SELECT column1 & column2 & & columnN FROM table_Name [WHERE conditions] ;`

Let's understand the below example which explains how to execute Bitwise AND operator on the data of SQL table:

- This example consists of the following table, which has two columns. Each column holds numerical values.
- When we use the Bitwise AND operator in SQL, then SQL converts the values of both columns in binary format, and the AND operation is performed on the converted bits.
- After that, SQL converts the resultant bits into user understandable format, i.e., decimal format.

Column1	Column2
1	1
2	5
3	4
4	2
5	3

- Suppose, we want to perform the Bitwise AND operator between both the columns of the above table. For this, we have to write the following query in SQL:

`SELECT Column1 & Column2 From TABLE_AND ;`

Bitwise OR (|)

- The Bitwise OR operator performs the logical OR operation on the given Integer values. This operator checks each bit of a value with the corresponding bit of another value.

Syntax of Bitwise OR Operator

```
SELECT column1 | column2 | .... | columnN FROM table_Name [WHERE conditions] ;
```

Let's understand the below example which explains how to execute Bitwise OR operator on the data of SQL table:

- This example consists of a table that has two columns. Each column holds numerical values.
- When we used the Bitwise OR operator in SQL, then SQL converts the values of both columns in binary format, and the OR operation is performed on the binary bits. After that, SQL converts the resultant binary bits into user understandable format, i.e., decimal format.

Column1	Column2
1	1
2	5
3	4
4	2
5	3

- Suppose, we want to perform the Bitwise OR operator between both the columns of the above table. For this, we have to write the following query in SQL:

```
SELECT Column1 | Column2 From TABLE_OR ;
```

5.6 SQL Create and Select Database

CREATE Database

- In SQL, the 'Create Database' statement is a first step for storing the structured data in the database.
- The database developers and the users use this statement in SQL for creating the new database in the database systems.
- It creates the database with the name which has been specified in the Create Database statement.

Syntax of Create Database statement in SQL

CREATE DATABASE Database_Name;

- In this syntax, **Database_Name** specifies the name of the database which we want to create in the system. We have to type the database name in query just after the 'Create Database' keyword.
- Following are the most important points which are required to learn while creating a database:
 - The database we want to create should be a simple and unique name, which can be easily identified.
 - Database name should be no more than 128 characters.

Syntax of Create Database statement in MySQL

- The same command is used in MySQL to create the new database for storing the structured data.

CREATE DATABASE Database_Name;

Syntax of Create Database in Oracle

- There is no need to create the database in Oracle systems. In the Oracle database, we can directly create the database tables.

Examples of Create Database statement in SQL

Following two examples which will help how to run and perform the Create Database query in SQL:

Example 1:

This example creates the **Student** database. To create the Student database, you have to type the following command in Structured Query Language:

CREATE DATABASE Student ;

When this query is executed successfully, then it will show the following output:

Database created successfully

You can also verify that your database is created in SQL or not by using the following query:

SHOW DATABASE ;

SQL does not allow developers to create the database with the existing database name.

Suppose if you want to create another Student database in the same database system, then the Create Database statement will show the following error in the output:

Can't create database 'Student'; database exists

- So, firstly you have to delete the existing database by using the Drop Statement.
- You can also replace the existing database with the help of Replace keyword.

If you want to replace the existing Student database, then you have to type the following SQL query:

CREATE OR REPLACE DATABASE Student ;

Example 2:

Suppose, we want to create the Employee database in the system.

Firstly, we have to type the following command in Structured Query Language

```
CREATE DATABASE Employee ;
```

When this query is executed successfully, then it will show the following output:

```
Database created successfully
```

You can also check that your database is created in SQL by typing the following query:

```
SHOW DATABASE ;
```

We know that SQL does not allow developers to create the database with the existing database name.

- Suppose, we want to create another Employee database in the same database system, firstly, we have to delete the existing database using a drop statement, or we have to replace the existing Employee database with the help of the 'replace' keyword.

To replace the existing Employee database with a new Employee database, we have to type the following query in SQL:

```
CREATE OR REPLACE DATABASE Employee;
```

SELECT Database

- Suppose database users and administrators want to perform some operations on tables, views, and indexes on the specific existing database in SQL.
- Firstly, they have to select the database on which they want to run the database queries.
- Any database user and administrator can easily select the particular database from the current database server using the **USE** statement in SQL.

Syntax of USE statement in SQL

```
USE database_name;
```

In this syntax, we have to define the name of the database after the **USE** keyword and the name of the database must be unique.

Syntax of USE statement in MySQL

```
USE database_name;
```

Syntax of USE statement in Oracle

There is no need to select the database in Oracle.

Examples of USE statement in SQL

In this section, we have taken the following three examples which will help you how to run and perform USE statement in SQL:

Example 1

- Suppose, you want to work with the **Hospital** database. For this firstly, you have to check that if the Hospital database exists on the current database server or not by using the following query:

```
SHOW DATABASES;
```

If the Hospital database is shown in the output, then you have to execute the following query to select the Hospital database:

```
USE Hospital;
```

Example 2

Suppose, you want to work with another College database in SQL. For this firstly, you have to check that the College database exists on the current database server or not by using the following query:

```
SHOW DATABASES;
```

If the College database is shown in the result, then you have to execute the following query to select the College database:

```
USE College;
```

Example 3

Suppose you want to work with another School database in SQL. For this firstly, you have to check that the School database exists on the current database server or not by using the following query:

```
SHOW DATABASES;
```

If the School database is shown in the result, then you have to execute the following query to select the School database:

```
USE School;
```

5.7 SQL TABLE Variable and Statements

SQL Table

- Table is a collection of data, organized in terms of rows and columns. In DBMS term, table is known as relation and row as tuple.

Note: A table has a specified number of columns, but can have any number of rows.

- Table is the simple form of data storage. A table is also considered as a convenient representation of relations.

Let's see an example of an employee table:

Employee		
EMP_NAME	ADDRESS	SALARY
Ankit	Lucknow	15000
Raman	Allahabad	18000
Mike	New York	20000

- In the above table, "Employee" is the table name, "EMP_NAME", "ADDRESS" and "SALARY" are the column names. The combination of data of multiple columns forms a row e.g. "Ankit", "Lucknow" and 15000 are the data of one row.

SQL TABLE Variable

- The **SQL Table variable** is used to create, modify, rename, copy and delete tables. Table variable was introduced by Microsoft.
- It was introduced with SQL server 2000 to be an alternative of temporary tables.
- It is a variable where we temporary store records and results. This is same like temp table but in the case of temp table we need to explicitly drop it.
- Table variables are used to store a set of records. So declaration syntax generally looks like CREATE TABLE syntax.

```
create table "tablename"
("column1" "data type",
"column2" "data type",
...
"columnN" "data type");
```

- When a transaction rolled back the data associated with table variable is not rolled back.
- A table variable generally uses lesser resources than a temporary variable.
- Table variable cannot be used as an input or an output parameter.

SQL CREATE TABLE

- SQL CREATE TABLE statement is used to create table in a database.
- If you want to create a table, you should name the table and define its column and each column's data type.

Let's see the simple syntax to create the table.

```
create table "tablename"
("column1" "data type",
"column2" "data type",
"column3" "data type",
...
"columnN" "data type");
```

- The data type of the columns may vary from one database to another.
- For example, NUMBER is supported in Oracle database for integer value whereas INT is supported in MySQL. Let us take an example to create a STUDENTS table with ID as primary key and NOT NULL are the constraint showing that these fields cannot be NULL while creating records in the table.

```
SQL> CREATE TABLE STUDENTS (
ID INT NOT NULL,
NAME VARCHAR (20) NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR (25),
PRIMARY KEY (ID)
);
```

- You can verify it, if you have created the table successfully by looking at the message displayed by the SQL Server, else you can use DESC command as follows:

```
SQL> DESC STUDENTS;
```

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
ID	Int(11)	NO	PRI		
NAME	Varchar(20)	NO			
AGE	Int(11)	NO			
ADDRESS	Varchar(25)	YES		NULL	

- 4 rows in set (0.00 sec)
- Now you have the STUDENTS table available in your database and you can use to store required information related to students.

SQL CREATE TABLE Example in MySQL

Let's see the command to create a table in MySQL database.

```
CREATE TABLE Employee
(
EmployeeID int,
FirstName varchar(255),
LastName varchar(255),
Email varchar(255),
AddressLine varchar(255),
City varchar(255)
);
```

SQL CREATE TABLE Example in Oracle

Let's see the command to create a table in Oracle database.

```
CREATE TABLE Employee
(
EmployeeID number(10),
FirstName varchar2(255),
LastName varchar2(255),
Email varchar2(255),
AddressLine varchar2(255),
City varchar2(255)
);
```

SQL CREATE TABLE Example in Microsoft SQLServer

Let's see the command to create a table in SQLServer database. It is same as MySQL and Oracle.

```
CREATE TABLE Employee
(
EmployeeID int,
FirstName varchar(255),
LastName varchar(255),
Email varchar(255),
AddressLine varchar(255),
City varchar(255)
);
```

Create a Table using another table

- We can create a copy of an existing table using the create table command.
- The new table gets the same column signature as the old table. We can select all columns or some specific columns.
- If we create a new table using an old table, the new table will be filled with the existing value from the old table.
- The basic syntax for creating a table with the other table is:

```
CREATE TABLE table_name AS
SELECT column1, column2, ...
FROM old_table_name WHERE ..... ;
```

The following SQL creates a copy of the employee table.

```
CREATE TABLE EmployeeCopy AS
SELECT EmployeeID, FirstName, Email
FROM Employee;
```

SQL Primary Key with CREATE TABLE Statement

The following query creates a PRIMARY KEY on the "D" column when the "Employee" table is created.

MySQL

```
CREATE TABLE Employee(
EmployeeID NOT NULL,
FirstName varchar(255) NOT NULL,
LastName varchar(255),
City varchar(255),
PRIMARY KEY (EmployeeID)
);
```

SQL Server / Oracle / MS Access

```
CREATE TABLE Employee(
EmployeeID NOT NULL PRIMARY KEY,
FirstName varchar(255) NOT NULL,
LastName varchar(255),
City varchar(255)
);
```

Use the following query to define a PRIMARY KEY constraints on multiple columns, and to allow naming of a PRIMARY KEY constraints.

```
CREATE TABLE Employee(  
EmployeeID NOT NULL,  
FirstName varchar(255) NOT NULL,  
LastName varchar(255),  
City varchar(255),  
CONSTRAINT PK_Employee PRIMARY KEY (EmployeeID, FirstName)  
);
```

SQL DELETE TABLE

The DELETE statement is used to delete rows from a table. If you want to remove a specific row from a table you should use WHERE condition.

DELETE FROM table_name [WHERE condition];

But if you do not specify the WHERE condition it will remove all the rows from the table.

DELETE FROM table_name;

There are some more terms similar to DELETE statement like as DROP statement and TRUNCATE statement but they are not exactly same there are some differences between them.

Difference between DELETE and TRUNCATE Statements

- There is a slight difference b/w delete and truncate statement.
- The **DELETE statement** only Deletes the rows from the table based on the condition defined by WHERE clause or delete all the rows from the table when condition is not specified. It does not free the space containing by the table.
- The **TRUNCATE statement** is used to delete all the rows from the table **and free the containing space**.

Let's see an "employee" table.

Emp_id	Name	Address	Salary
1	Aryan	Allahabad	22000
2	Shurabhi	Varanasi	13000
3	Pappu	Delhi	24000

Execute the following query to truncate the table:

TRUNCATE TABLE employee;

Difference b/w DROP and TRUNCATE Statements

- When you use the drop statement it deletes the table's row together with the table's definition so all the relationships of that table with other tables will no longer be valid.

When you drop a table:

- Table structure will be dropped
- Relationship will be dropped
- Integrity constraints will be dropped
- Access privileges will also be dropped

On the other hand when we **TRUNCATE** a table, the table structure remains the same, so you will not face any of the above problems.

5.8 SQL Joins and Injection

SQL JOIN

- As the name shows, JOIN means to combine something. In case of SQL, JOIN means "to combine two or more tables".
- The SQL JOIN clause takes records from two or more tables in a database and combines it together.
- ANSI standard SQL defines five types of JOIN:
 1. Inner join
 2. Left outer join
 3. Right outer join
 4. Full outer join
 5. Cross join

In the process of joining, rows of both tables are combined in a single table.

Why SQL JOIN is used?

- If you want to access more than one table through a select statement.
- If you want to combine two or more table then SQL JOIN statement is used. It combines rows of that tables in one table and one can retrieve the information by a SELECT statement.
- The joining of two or more tables is based on common field between them.
- SQL INNER JOIN also known as simple join is the most common type of join.

How to use SQL JOIN?

Let an example to deploy SQL JOIN process:

1. Staff Table

ID	Staff_NAME	Staff_AGE	STAFF_ADDRESS	Monthly_Package
1	ARYAN	22	MUMBAI	18000
2	SUSHIL	32	DELHI	20000
3	MONTY	25	MOHALI	22000
4	AMIT	20	ALLAHABAD	12000

2. Payment Table

Payment_ID	DATE	Staff_ID	AMOUNT
101	30/12/2009	1	3000.00
102	22/02/2010	3	2500.00
103	23/02/2010	4	3500.00

So if you follow this JOIN statement to join these two tables:

```
SELECT Staff_ID, Staff_NAME, Staff_AGE, AMOUNT
FROM STAFF s, PAYMENT p
WHERE s.ID =p.STAFF_ID;
```

This will produce the result like this:

STAFF_ID	NAME	Staff_AGE	AMOUNT
3	MONTY	25	2500
1	ARYAN	22	3000
4	AMIT	25	3500
1	ARYAN	22	3000

SQL Injection

- The SQL Injection is a code penetration technique that might cause loss to our database.
- It is one of the most practiced web hacking techniques to place malicious code in SQL statements, via webpage input. SQL injection can be used to manipulate the application's web server by malicious users.
- SQL injection generally occurs when we ask a user to input their username/userID. Instead of a name or ID, the user gives us an SQL statement that we will unknowingly run on our database.

For Example - we create a SELECT statement by adding a variable "demoUserId" to select a string.

- The variable will be fetched from user input (getRequestParam).

```
demoUserId = getRequestString("UserId");
demoSQL = "SELECT * FROM users WHERE UserId =" +demoUserId;
```

Types of SQL injection attacks

- SQL injections can do more harm other than passing the login algorithms. Some of the SQL injection attacks include:
- Updating, deleting, and inserting the data: An attack can modify the cookies to poison a web application's database query.
- It is executing commands on the server that can download and install malicious programs such as Trojans.
- We are exporting valuable data such as credit card details, email, and passwords to the attacker's remote server.
- Getting user login details: It is the simplest form of SQL injection. Web application typically accepts user input through a form, and the front end passes the user input to the back end database for processing.

Example of SQL Injection

- We have an application based on employee records.
- Any employee can view only their own records by entering a unique and private employee ID.
- We have a field like an Employee ID. And the employee enters the following in the input field:

236893238 or 1=1

It will translate to:

SELECT * from EMPLOYEE where EMPLOYEE_ID == 236893238 or 1=1

- The SQL code above is valid and will return EMPLOYEE_ID row from the EMPLOYEE table. The 1=1 will return all records for which this holds true.
- All the employee data is compromised; now, the malicious user can also similarly delete the employee records.

Example:

SELECT * from Employee where (Username == "" or 1=1) AND (Password="" or 1=1).

Now the malicious user can use the '=' operator sensibly to retrieve private and secure user information. So instead of the query mentioned above, the following query, when exhausted, retrieve protected data, not intended to be shown to users.

SELECT * from EMPLOYEE where (Employee_name = " " or 1=1) AND (Password=" " or 1=1)

SQL injection based on Batched SQL Statements

- Several databases support batched SQL statements. It is a group of two or more SQL statements separated by semicolons.
- The SQL statement given below will return all rows from the Employee table, then delete the Employee_Add table.

SELECT * From Employee; DROP Table Employee_Add

How to detect SQL Injection attacks

- Creating a SQL Injection attack is not difficult, but even the best and good-intentioned developers make mistakes.
- The detection of SQL Injection is, therefore, an essential component of creating the risk of an SQL injection attack.
- Web Application Firewall can detect and block basic SQL injection attacks, but we should depend on it as the sole preventive measure.
- Intrusion Detection System (IDS) is both network-based and host-based.
- It can be tuned to detect SQL injection attacks. Network-based IDSec can monitor all connections to our database server, and flags suspicious activities.
- The host-based IDS can monitor web server logs and alert when something strange happens.

Impact of SQL Injection

- The intruder can retrieve all the user-data present in the database, such as user details, credit card information, and social security numbers, and can also gain access to protected areas like the administrator portal.
- It is also possible to delete the user data from the tables.
- These days all the online shopping applications, bank transactions use back-end database servers. If the intruder can exploit SQL injection, the entire server is compromised.

How to prevent SQL Injection attack

- We should use user authentication to validate input from the user by pre-defining length, input type, and the input field.
- Restricting the access privileges of users and defining the amount of data any outsider can access from the database.
- Generally, the user cannot be granted permission to access everything in the database.
- We should not use system administrator accounts.

5.9 Selection Commands, Alias and Subqueries

FILTER Command

- If you wanted to refine your query even more by running your aggregations against a limited set of the values in a column you could use the FILTER keyword.
- For example, if you wanted to know both the number of deals won by a sales agent and the number of those deals that had a value greater than 1000, you could use the query:

```
SELECT sales_agent,
       COUNT(sales_pipeline.close_value) AS total,
       COUNT(sales_pipeline.close_value)
  FILTER(WHERE sales_pipeline.close_value > 1000) AS `over 1000`
   FROM sales_pipeline
 WHERE sales_pipeline.deal_stage = "Won"
 GROUP BY sales_pipeline.sales_agent
```

- The first several rows of the resulting table would look like this:

sales_agent	total	over 1000
Boris Faz	101	70
Maureen Marcano	149	96
Vicki Laflamme	221	111
Donn Cantrell	158	106
Jonathan Berthelot	171	74
Wilburn Farren	55	38
Elease Gluck	80	32
Cassey Cress	163	112
James Ascencio	135	88
Kami Bicknell	174	78
Anna Snelling	208	68
Violet Mclelland	122	33

Group By Command

- Returning to a previous section, when we were working with aggregations, we used the aggregate function AVG to find out the average deal size.
- If we wanted to know the average value of the deals won by each sales person from highest average to lowest, the query would look like:

```
SELECT sales_agent,
       AVG(close_value)
    FROM sales_pipeline
   WHERE sales_pipeline.deal_stage = "Won"
  GROUP BY sales_agent
 ORDER BY AVG(close_value) DESC
```

sales_agent	avg
Elease Gluck	3614.9375
Darcel Schlecht	3304.3381088825213
Rosalina Dieter	3269.4861111111113
Daniell Hammack	3194.9912280701756
James Ascencio	3063.2074074074076
Rosie Papadopoulos	2950.8846153846152
Wilburn Farren	2866.181818181818
Reed Clapper	2827.974193548387
Donn Cantrell	2821.8987341772154

Order By Command

- The ORDER BY statement in SQL is used to sort the fetched data in either ascending or descending according to one or more columns.
- By default ORDER BY sorts the data in ascending order.
- We can use the keyword DESC to sort the data in descending order and the keyword ASC to sort in ascending order.

ORDER BY Syntax

```
SELECT column1, column2, ...
   FROM table_name
 ORDER BY column1, column2, ... ASC|DESC;
```

SQL Aliases

- SQL aliases are used to give a table, or a column in a table, a temporary name.
- Aliases are often used to make column names more readable.
- An alias only exists for the duration of that query.
- An alias is created with the AS keyword.

Alias Column Syntax

```
SELECT column_name AS alias_name  
FROM table_name;
```

Alias Table Syntax

```
SELECT column_name(s)  
FROM table_name AS alias_name;
```

SQL Subqueries

- A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.
- A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.
- Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.
- There are a few rules that subqueries must follow –
 - Subqueries must be enclosed within parentheses.
 - A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
 - An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.
 - Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.
 - The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
 - A subquery cannot be immediately enclosed in a set function.
 - The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.

Subqueries with the SELECT Statement

Subqueries are most frequently used with the SELECT statement. The basic syntax is as follows:

```
SELECT column_name [, column_name ]
FROM table1 [, table2 ]
WHERE column_name OPERATOR
  (SELECT column_name [, column_name ]
   FROM table1 [, table2 ]
   [WHERE])
```

Example

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Now, let us check the following subquery with a SELECT statement.

```
SQL> SELECT *
  FROM CUSTOMERS
 WHERE ID IN (SELECT ID
    FROM CUSTOMERS
 WHERE SALARY > 4500) ;
```

This would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

Subqueries with the INSERT Statement

- Subqueries also can be used with INSERT statements.
- The INSERT statement uses the data returned from the subquery to insert into another table.
- The selected data in the subquery can be modified with any of the character, date or number functions.
- The basic syntax is as follows.

```
INSERT INTO table_name [ (column1 [, column2]) ]  
    SELECT [ *|column1 [, column2 ]  
        FROM table1 [, table2 ]  
        [ WHERE VALUE OPERATOR ]
```

Subqueries with the UPDATE Statement

- The subquery can be used in conjunction with the UPDATE statement.
- Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.
- The basic syntax is as follows.

```
UPDATE table  
SET column_name = new_value  
[ WHERE OPERATOR [ VALUE ]  
    (SELECT COLUMN_NAME  
        FROM TABLE_NAME)  
    [ WHERE) ]
```

Subqueries with the DELETE Statement

- The subquery can be used in conjunction with the DELETE statement like with any other statements mentioned above.
- The basic syntax is as follows.

```
DELETE FROM TABLE_NAME  
[ WHERE OPERATOR [ VALUE ]  
    (SELECT COLUMN_NAME  
        FROM TABLE_NAME)  
    [ WHERE) ]
```

5.10 MySQL User Access Control Functions, Backup & Recovery

Access Control and Account Management

- MySQL enables the creation of accounts that permit client users to connect to the server and access data managed by the server.
- The primary function of the MySQL privilege system is to authenticate a user who connects from a given host and to associate that user with privileges on a database such as SELECT, INSERT, UPDATE, and DELETE.
- Additional functionality includes the ability to grant privileges for administrative operations.
- To control which users can connect, each account can be assigned authentication credentials such as a password.
- The user interface to MySQL accounts consists of SQL statements such as CREATE USER, GRANT, and REVOKE.
- The MySQL privilege system ensures that all users may perform only the operations permitted to them.
- As a user, when you connect to a MySQL server, your identity is determined by the host from which you connect and the username you specify.
- When you issue requests after connecting, the system grants privileges according to your identity and what you want to do.
- MySQL considers both your host name and user name in identifying you because there is no reason to assume that a given user name belongs to the same person on all hosts.

For example, the user joe who connects from office.example.com need not be the same person as the user joe who connects from home.example.com. MySQL handles this by enabling you to distinguish users on different hosts that happen to have the same name: You can grant one set of privileges for connections by joe from office.example.com, and a different set of privileges for connections by joe from home.example.com. To see what privileges a given account has, use the SHOW GRANTS statement.

For example:

```
SHOW GRANTS FOR 'joe'@'office.example.com';
SHOW GRANTS FOR 'joe'@'home.example.com';
```

- Internally, the server stores privilege information in the grant tables of the MySQL system database.
- The MySQL server reads the contents of these tables into memory when it starts and bases access-control decisions on the in-memory copies of the grant tables.
- MySQL access control involves two stages when you run a client program that connects to the server:
 - **Stage 1:** The server accepts or rejects the connection based on your identity and whether you can verify your identity by supplying the correct password.
 - **Stage 2:** Assuming that you can connect, the server checks each statement you issue to determine whether you have sufficient privileges to perform it. For example, if you try to select rows from a table in a database or drop a table from the database, the server verifies that you have the SELECT privilege for the table or the DROP privilege for the database.
- There are some things that you cannot do with the MySQL privilege system:
 - You cannot explicitly specify that a given user should be denied access. That is, you cannot explicitly match a user and then refuse the connection.
 - You cannot specify that a user has privileges to create or drop tables in a database but not to create or drop the database itself.
 - A password applies globally to an account. You cannot associate a password with a specific object such as a database, table, or routine.

Account User Names and Passwords

- MySQL stores accounts in the user table of the mysql system database.
- An account is defined in terms of a user name and the client host or hosts from which the user can connect to the server.
- MySQL user names are up to 32 characters long. Operating system user names may have a different maximum length.
- An account may also have authentication credentials such as a password.
- The credentials are handled by the account authentication plugin.
- MySQL supports multiple authentication plugins.
- Some of them use built-in authentication methods, whereas others enable authentication using external authentication methods.
- Passwords stored in the user table are encrypted using plugin-specific algorithms.
- To connect to a MySQL server with a command-line client, specify user name and password options as necessary for the account that you want to use:

```
$> mysql --user=finley --password db_name
```

- If you prefer short options, the command looks like this:

```
$> mysql -u finley -p db_name
```

- If you omit the password value following the --password or -p option on the command line (as just shown), the client prompts for one. Alternatively, the password can be specified on the command line:

```
$> mysql --user=finley --password=password db_name
```

```
$> mysql -u finley -ppassword db_name
```

- If you use the -p option, there must be no space between -p and the following password value.
- Specifying a password on the command line should be considered insecure.

Specifying Account Names

- MySQL account names consist of a user name and a host name, which enables creation of distinct accounts for users with the same username who connect from different hosts.
- Account name syntax is 'user_name'@'host_name'.
- The '@'host_name' part is optional. An account name consisting only of a user name is equivalent to 'user_name'@'%''. For example, 'me' is equivalent to 'me'@'%''.
- The username and host name need not be quoted if they are legal as unquoted identifiers.
- Quotes must be used if a user_name string contains special characters (such as space or -), or a host_name string contains special characters or wildcard characters (such as . or %).
- For example, in the account name 'test-user'@'%com', both the username and host name parts require quotes.
- Quote user names and host names as identifiers or as strings, using either backticks (`), single quotation marks ('), or double quotation marks (").
- The username and host name parts, if quoted, must be quoted separately. That is, write 'me'@'localhost', not 'me@localhost'. The latter is actually equivalent to 'me@localhost'@'%''.
- A reference to the CURRENT_USER or CURRENT_USER() function is equivalent to specifying the current client's user name and host name literally.
- For a host value specified as an IPv4 address, a netmask can be given to indicate how many address bits to use for the network number. Netmask notation cannot be used for IPv6 addresses.
- The syntax is host_ip/netmask. For example:

CREATE USER 'david'@'198.51.100.0/255.255.255.0';

- This enables david to connect from any client host having an IP address client_ip for which the following condition is true:

$$\text{client_ip} \& \text{netmask} = \text{host_ip}$$

- That is, for the CREATE USER statement just shown:

$$\text{client_ip} \& 255.255.255.0 = 198.51.100.0$$

- IP addresses that satisfy this condition range from 198.51.100.0 to 198.51.100.255.

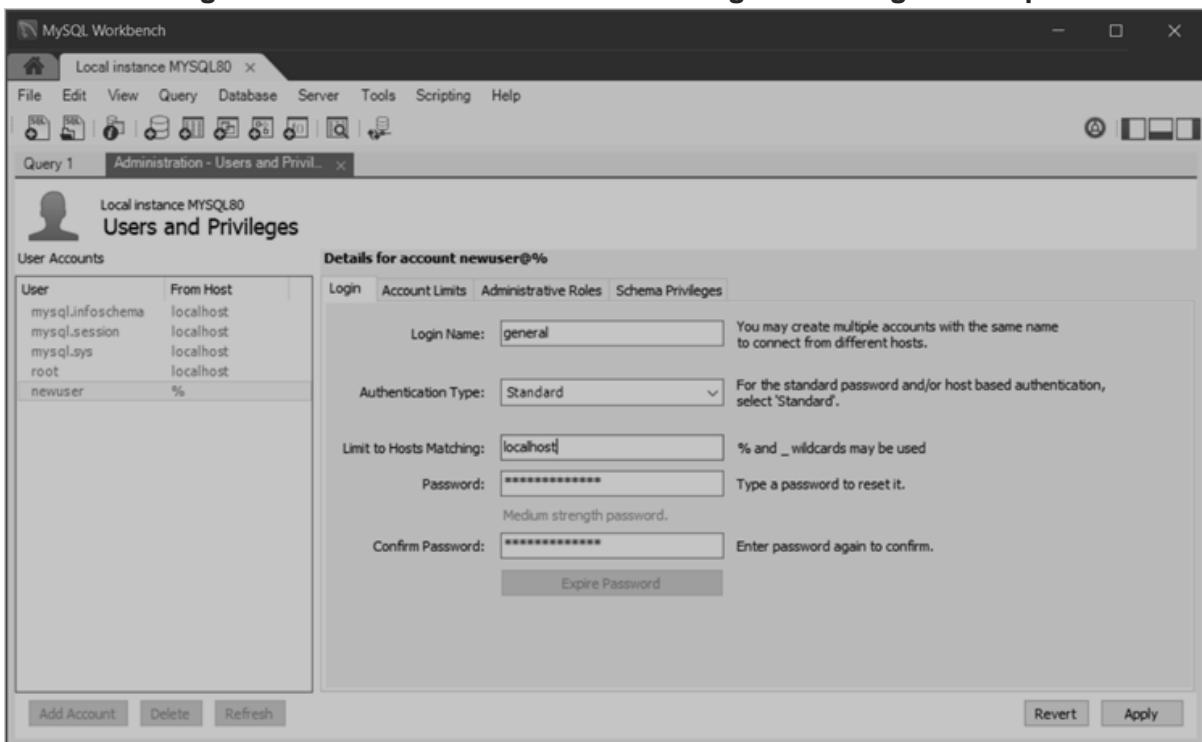
Users and Privileges

- The Administration - Users and Privileges tab provides a list of all users and privileges that relate to an active MySQL server instance.
- From this tab, you can add and manage user accounts, adjust privileges, and expire passwords.
- To open the Administration - Users and Privileges tab:
 - Establish a connection to an active MySQL server instance.
 - Within the connection tab, do one of the following:
 - Click Users and Privileges from the Management list within the Navigator area.
 - Click Server and then Users and Privileges from the menu.
- The Administration - Users and Privileges tab has several task areas, which are described in the following sections:
 - User Accounts
 - Login Tab
 - Account Limits Tab
 - Administrative Roles Tab
 - Schema Privileges Tab

User Accounts

- User Accounts consists of a vertical box that lists each user account associated with the active MySQL connection.
- The list contains each user name and the host name where the account resides.
- Use the Add Account, Delete, and Refresh buttons to manage the list of user accounts.
- Selecting an account from the list focuses the account details, which appear in set of tabs, onto the selected user account.
- The figure that follows shows the layout of the Administration - Users and Privileges tab with the Login detail tab open to show an example of general account information.

Navigator Administration: User and Privileges with Login Tab Open



Login Tab

- The Login tab provides the following information related to the selected user account:
 - **Login Name:** You may create multiple accounts with the same name to connect from different hosts.
 - **Authentication Type:** For standard password or host-based authentication, select Standard. The caching_sha2_password and SHA256_Password authentication types provide more secure password encryption than the Standard authentication type.
 - Starting with MySQL 8.0.4, the caching_sha2_password plugin is the default authentication plugin for the server. An account that authenticates with caching_sha2_password must use either a secure connection or an unencrypted connection that supports password exchange using an RSA key pair.
 - **Limit to Hosts Matching:** The % and _ characters may be used as wildcards. The percent sign (%) matches zero or more characters and the underscore (_) matches a single character.
 - **Password and Confirm Password:** To reset a password, type in the new password and then confirm it. Consider using a password of eight or more characters with mixed-case letters, numbers, and punctuation marks.
 - Use Expire Password to require a change of password to use the account.

Account Limits Tab

- The Account Limits tab defines the following limits on the selected user account:
 - **Max. Queries:** The number of queries the account can execute within one hour.
 - **Max. Updates:** The number of updates the account can execute within one hour.
 - **Max. Connections:** The number of times the account can connect to the server within an hour.
 - **Concurrent Connections:** The number of simultaneous connections to the server the account can have.

Administrative Roles Tab

- Roles are a quick way of granting a set of privileges to a user, based on the work the user must carry out on the server.
- It is also possible to assign multiple roles to a user account or to assign privileges directly to an account without first assigning roles.
- After you select a role for a user account, you will see the accumulated privileges in the Global Privileges panel.
- For example, if you select the role BackupAdmin, the privileges granted include EVENT, LOCK TABLES, SELECT, SHOW DATABASES.
- The Administrative Roles tab includes the following roles:
 - **DBA:** Grants the rights to perform all tasks.
 - **MaintenanceAdmin:** Grants rights to maintain the server.
 - **ProcessAdmin:** Grants rights to assess, monitor, and kill user processes.
 - **UserAdmin:** Grants rights to create user logins and reset passwords.
 - **SecurityAdmin:** Grants rights to manage logins and grant and revoke server privileges.
 - **MonitorAdmin:** Grants the minimum rights to monitor the server.
 - **DBManager:** Grants full rights on all databases.
 - **DBDesigner:** Grants rights to create and reverse-engineer any database schema.
 - **ReplicationAdmin:** Grants rights needed to set up and manage replication.
 - **BackupAdmin:** Grants minimum rights required to back up any database.
 - **Custom:** Lists other (custom) privileges that are assigned to the user account. This role is not available for all default accounts, such as root. If you select a user account and then select one or more privileges directly that are outside of any selected roles, the Custom role is added (and selected) to the list of roles.
- To remove all of the rights assigned to the selected user account, click Revoke All Privileges.

Schema Privileges Tab

- The Schema Privileges tab refines the way you assign access rights to one or more schemas by user account.
- To assign privileges to the selected account by schema, do the following:
 1. Add a schema entry (or rule) that specifies which schema or schemas apply the selected user account. Click Add Entry to open the New Schema Privilege Definition dialog. The dialog provides the following independent options to select:
 - **All Schema (%)** - This rule applies to any schema name.
 - **Schemas matching pattern: *pattern*** - Apply this rule to schemas that match the given name or pattern. You may use _ and % as wildcards in the pattern; however, to use the literal value you must escape each wildcard character with a backslash (\).
 - **Selected schema: *schema name*** - Apply the rule to the specific schema name selected from the list.

Use Delete Entry to remove an entry and the privileges associated with it from the list. When you click Revoke All Privileges, you are prompted remove all privileges assigned to the selected user account.

2. With an entry selected, mark the individual access rights that apply only to the schema or schemas defined in the entry. The access rights are categorized as Object Rights, DDL Rights, and Other Rights. Each right that you select appears in the Privileges column of the schema entry.

Access Control

Stage 1: Connection Verification

- When you attempt to connect to a MySQL server, the server accepts or rejects the connection based on these conditions:
 - Your identity and whether you can verify it by supplying the proper credentials.
 - Whether your account is locked or unlocked.
- The server checks credentials first, then account locking state.
- A failure at either step causes the server to deny access to you completely. Otherwise, the server accepts the connection, and then enters Stage 2 and waits for requests.
- The server performs identity and credentials checking using columns in the user table, accepting the connection only if these conditions are satisfied:
 - The client host name and user name match the Host and User columns in some user table row. For the rules governing permissible Host and User values.
 - The client supplies the credentials specified in the row (for example, a password), as indicated by the authentication_string column. Credentials are interpreted using the authentication plugin named in the plugin column.
 - The row indicates that the account is unlocked. Locking state is recorded in the account_locked column, which must have a value of 'N'. Account locking can be set or changed with the CREATE USER or ALTER USER statement.
- Your identity is based on two pieces of information:
 - Your MySQL user name.
 - The client host from which you connect.

- The following table shows how various combinations of User and Host values in the user table apply to incoming connections.

User Value	Host Value	Permissible Connections
'fred'	'h1.example.net'	fred, connecting from h1.example.net
"	'h1.example.net'	Any user, connecting from h1.example.net
'fred'	'%'	fred, connecting from any host
"	'%'	Any user, connecting from any host
'fred'	'%.example.net'	fred, connecting from any host in the example.net domain
'fred'	'x.example.%'	fred, connecting from x.example.net, x.example.com, x.example.edu, and so on; this is probably not useful
'fred'	'198.51.100.177'	fred, connecting from the host with IP address 198.51.100.177
'fred'	'198.51.100.%'	fred, connecting from any host in the 198.51.100 class C subnet
'fred'	'198.51.100.0/255.255.255.0'	Same as previous example

- It is possible for the client host name and user name of an incoming connection to match more than one row in the user table. The preceding set of examples demonstrates this: Several of the entries shown match a connection from h1.example.net by fred.
- As of MySQL 8.0.23, accounts with an IP address in the host part have this order of specificity:
 - Accounts that have the host part given as an IP address:

```
CREATE USER 'user_name'@'127.0.0.1';
CREATE USER 'user_name'@'198.51.100.44';
```

- Accounts that have the host part given as an IP address using CIDR notation:

```
CREATE USER 'user_name'@'192.0.2.21/8';
CREATE USER 'user_name'@'198.51.100.44/16';
```

- Accounts that have the host part given as an IP address with a subnet mask:

```
CREATE USER 'user_name'@'192.0.2.0/255.255.255.0';
CREATE USER 'user_name'@'198.51.0.0/255.255.0.0';
```

- The pattern '%' means "any host" and is least specific.
- The empty string " also means "any host" but sorts after '%'.

Stage 2: Request Verification

- After the server accepts a connection, it enters Stage 2 of access control.
- For each request that you issue through the connection, the server determines what operation you want to perform, then checks whether your privileges are sufficient.
- This is where the privilege columns in the grant tables come into play.
- These privileges can come from any of the user, global_grants, db, tables_priv, columns_priv, or procs_priv tables.
- The user and global_grants tables grant global privileges.
- The rows in these tables for a given account indicate the account privileges that apply on a global basis no matter what the default database is.
- For example, if the user table grants you the DELETE privilege, you can delete rows from any table in any database on the server host.
- It is wise to grant privileges in the user table only to people who need them, such as database administrators.
- For other users, leave all privileges in the user table set to 'N' and grant privileges at more specific levels only (for particular databases, tables, columns, or routines).
- It is also possible to grant database privileges globally but use partial revokes to restrict them from being exercised on specific databases.
- The db table grants database-specific privileges. Values in the scope columns of this table can take the following forms:
 - A blank User value matches the anonymous user. A nonblank value matches literally; there are no wildcards in user names.
 - The wildcard characters % and _ can be used in the Host and Db columns. These have the same meaning as for pattern-matching operations performed with the LIKE operator. If you want to use either character literally when granting privileges, you must escape it with a backslash. For example, to include the underscore character (_) as part of a database name, specify it as _ in the GRANT statement.
 - A '%' or blank Host value means "any host."
 - A '%' or blank Db value means "any database."
- The server reads the db table into memory and sorts it at the same time that it reads the user table.
- The server sorts the db table based on the Host, Db, and User scope columns.
- As with the user table, sorting puts the most-specific values first and least-specific values last, and when the server looks for matching rows, it uses the first match that it finds.
- The tables_priv, columns_priv, and procs_priv tables grant table-specific, column-specific, and routine-specific privileges.

- Values in the scope columns of these tables can take the following forms:
 - The wildcard characters % and _ can be used in the Host column. These have the same meaning as for pattern-matching operations performed with the LIKE operator.
 - A '%' or blank Host value means "any host."
 - The Db, Table_name, Column_name, and Routine_name columns cannot contain wildcards or be blank.
- The server sorts the tables_priv, columns_priv, and procs_priv tables based on the Host, Db, and User columns. This is similar to db table sorting, but simpler because only the Host column can contain wildcards.
- The server uses the sorted tables to verify each request that it receives.
- For requests that require administrative privileges such as SHUTDOWN or RELOAD, the server checks only the user and global_privilege tables because those are the only tables that specify administrative privileges.
- The server grants access if a row for the account in those tables permits the requested operation and denies access otherwise.
- Expressed in boolean terms, the preceding description of how a user's privileges are calculated may be summarized like this:

global privileges
OR database privileges
OR table privileges
OR column privileges
OR routine privileges

Restore and Back-up

Backup

- It is important to back up your databases so that you can recover your data and be up and running again in case problems occur, such as system crashes, hardware failures, or users deleting data by mistake.
- Backups are also essential as a safeguard before upgrading a MySQL installation, and they can be used to transfer a MySQL installation to another system or to set up replica servers.
- MySQL offers a variety of backup strategies from which you can choose the methods that best suit the requirements for your installation.

Physical (Raw) Versus Logical Backups

- **Physical backups** consist of raw copies of the directories and files that store database contents. This type of backup is suitable for large, important databases that need to be recovered quickly when problems occur.
- **Logical backups** save information represented as logical database structure (CREATE DATABASE, CREATE TABLE statements) and content (INSERT statements or delimited-text files). This type of backup is suitable for smaller amounts of data where you might edit the data values or table structure, or recreate the data on a different machine architecture.

Online Versus Offline Backups

- **Online backups** take place while the MySQL server is running so that the database information can be obtained from the server.
- **Offline backups** take place while the server is stopped. This distinction can also be described as “hot” versus “cold” backups; a “warm” backup is one where the server remains running but locked against modifying data while you access database files externally.

Local Versus Remote Backups

- A local backup is performed on the same host where the MySQL server runs, whereas a remote backup is done from a different host.
- For some types of backups, the backup can be initiated from a remote host even if the output is written locally on the server host.

Snapshot Backups

- Some file system implementations enable “snapshots” to be taken.
- These provide logical copies of the file system at a given point in time, without requiring a physical copy of the entire file system.
- MySQL itself does not provide the capability for taking file system snapshots.
- It is available through third-party solutions such as Veritas, LVM, or ZFS.

Full Versus Incremental Backups

- A full backup includes all data managed by a MySQL server at a given point in time.
- An incremental backup consists of the changes made to the data during a given time span (from one point in time to another).
- MySQL has different ways to perform full backups.
- Incremental backups are made possible by enabling the server's binary log, which the server uses to record data changes.

Making Backups with mysqldump

- The **mysqldump** program can make backups.
- It can back up all kinds of tables.
- For InnoDB tables, it is possible to perform an online backup that takes no locks on tables using the **--single-transaction** option to **mysqldump**.

Making Backups by Copying Table Files

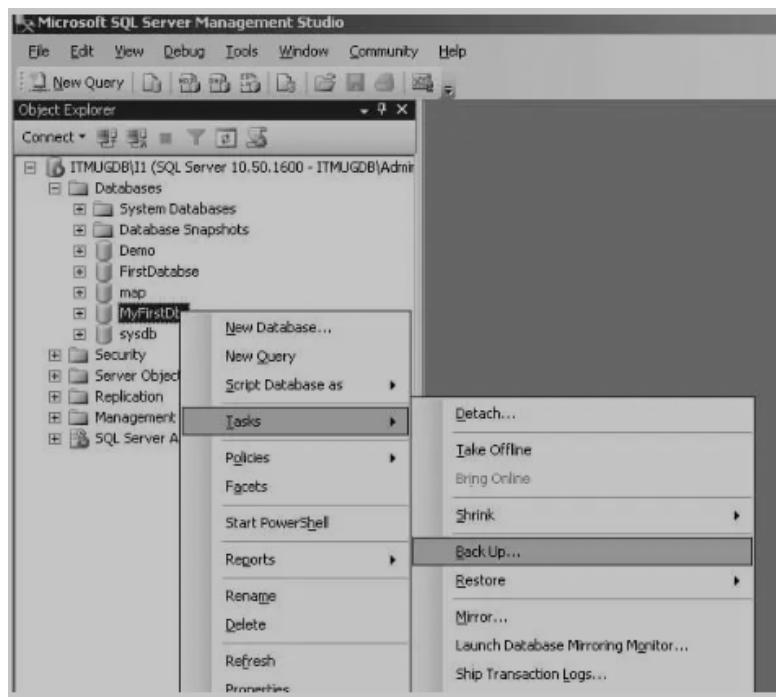
- MyISAM tables can be backed up by copying table files (*.MYD, *.MYI files, and associated *.sdi files).
- To get a consistent backup, stop the server or lock and flush the relevant tables:
`FLUSH TABLES tbl_list WITH READ LOCK;`
- You need only a read lock; this enables other clients to continue to query the tables while you are making a copy of the files in the database directory.
- The flush is needed to ensure that the all-active index pages are written to disk before you start the backup.

Backup SQL Database with SQL Server Management Studio

Step 1. Open SSMS and connect to the SQL Server.

Step 2. Expand **Databases** and select the required database.

Step 3. Right click on the database >> **Tasks** >> **Backup**



Step 4. In **Back Up Database** window, select the **Backup Type** as **Full** and under **Destination**, select **Back up to: Disk**.

Step 5. Select the **Remove** button.

Step 6. Click on **Add** button to select the destination and name for the database backup file.

Step 7. Select the required folder for the backup file and enter the file name with a **.bak** extension.

Step 8. Click **OK** to end the backup process.

Backup Scheduling, Compression, and Encryption

- Backup scheduling is valuable for automating backup procedures.
- Compression of backup output reduces space requirements, and encryption of the output provides better security against unauthorized access of backed-up data.
- MySQL itself does not provide these capabilities.
- The MySQL Enterprise Backup product can compress InnoDB backups, and compression or encryption of backup output can be achieved using file system utilities.
- Other third-party solutions may be available.

Full Versus Point-in-Time (Incremental) Recovery

- A **full recovery** restores all data from a full backup. This restores the server instance to the state that it had when the backup was made. If that state is not sufficiently current, a full recovery can be followed by recovery of incremental backups made since the full backup, to bring the server to a more up-to-date state.
- **Incremental recovery** is recovery of changes made during a given time span. This is also called point-in-time recovery because it makes a server's state current up to a given time. Point-in-time recovery is based on the binary log and typically follows a full recovery from the backup files that restores the server to its state when the backup was made. Then the data changes written in the binary log files are applied as incremental recovery to redo data modifications and bring the server up to the desired point in time.

Using Backups for Recovery

- Now, suppose that we have a catastrophic unexpected exit on Wednesday at 8 a.m. that requires recovery from backups. To recover, first we restore the last full backup we have (the one from Sunday 1 p.m.). The full backup file is just a set of SQL statements, so restoring it is very easy:

```
$> mysql < backup_sunday_1_PM.sql
```

- At this point, the data is restored to its state as of Sunday 1 p.m..
- To restore the changes made since then, we must use the incremental backups; that is, the gbichot2-bin.000007 and gbichot2-bin.000008 binary log files.
- Fetch the files if necessary, from where they were backed up, and then process their contents like this:

```
$> mysqlbinlog gbichot2-bin.000007 gbichot2-bin.000008 | mysql
```

- We now have recovered the data to its state as of Tuesday 1 p.m., but still are missing the changes from that date to the date of the crash.
- To not lose them, we would have needed to have the MySQL server store its MySQL binary logs into a safe location (RAID disks, SAN, ...) different from the place where it stores its data files, so that these logs were not on the destroyed disk. That is, we can start the server with a log-bin option that specifies a location on a different physical device from the one on which the data directory resides. That way, the logs are safe even if the device containing the directory is lost.
- If we had done this, we would have the gbichot2-bin.000009 file (and any subsequent files) at hand, and we could apply them using mysqlbinlog and mysql to restore the most recent data changes with no loss up to the moment of the crash:

```
$> mysqlbinlog gbichot2-bin.000009 ... | mysql
```

Section 3: Exercises

Exercise 1: Draw architecture of SQL?

Exercise 2: Make the tables of users and rides, write a query to report the distance travelled by each user in descending order.

Exercise 3: Select the largest three departments with ten or more employees and rank them according to the percentage of employees making over \$100,000.

In this problem, you are given two tables: An employees table and a departments table.

Exercise 4: Participate in group discussion on following topics:

- a) SQL Statements
- b) SQL Data Types
- c) SQL Operators
- d) Types of Recovery and Back-ups
- e) Selection Commands
- f) User Access Control Function

Section 4: Assessment Questionnaire

1. What are the four components in SQL process?
2. What are the advantages of SQL?
3. What are the disadvantages of SQL?
4. Name some commonly used SQL Commands.
5. The _____ of the structured query language is a unique set of rules and guidelines, which is not case-sensitive.
6. What are three categories of SQL data types?
7. What are types of operators in SQL?
8. The _____ is used to create, modify, rename, copy and delete tables.
9. What are types of Joins in SQL?
10. Why SQL JOIN is used?
11. SQL _____ are used to give a table, or a column in a table, a temporary name.
12. A _____ is a query within another SQL query and embedded within the WHERE clause.
13. What are different types of Backups?
14. When would you use the GROUP BY statement?
15. What are the most common aggregate functions in SQL? What do they do?
16. What is a unique key in SQL?
17. What is the difference between UNION and UNION ALL?
18. What is the difference between a RIGHT JOIN and a LEFT JOIN?
19. What is the difference between a table and a view?
20. What SQL operator is used for pattern matching?
21. What command would you use to update data in a table?
22. Which operator is used to select values within a range? What types of values can be selected?

-----End of the Module-----

MODULE 6

POWER BI TRAINING

Section 1: Learning Outcomes

After completing this module, you will be able to:

- Introduce ‘Power BI’
- Define Components, Features and Architecture of Power BI
- Describe the tools mostly used in Power BI
- Explain Advantages and Disadvantages of actual Power BI
- Download and install Power BI Desktop
- Create Power BI Dashboards and Reports
- Explain Power BI Data Sources
- Describe Power BI Embedded and Gateway
- Explore Building Blocks of Power BI
- State the functions of Power BI Report Server
- Describe about Power BI DAX

Section 2: Relevant Knowledge

6.1 Introduction to Power BI

What is BI?

- The BI term refers to ‘Business Intelligence’.
- It is a data-driven decision support system (DSS), which helps you to analyze the data and provide actionable information.
- It helps the business manager, corporate executives, and other users in making their decisions easily.
- Business intelligence refers to the applications, technologies, and practices for the collection, analysis, integration, and presents the business information.
- The purpose of business intelligence is to support better decision making.
- Sometimes the business intelligence is used interchangeably with briefing books, reports, query tools, and executive information systems.

Benefits of BI

- It allows real-time dashboard updates.
- It provides secure and reliable connections to the data sources in the cloud.
- It allows data exploration using a natural language query.
- Power BI provides a hybrid configuration, quick deployment, and secure environment.
- It provides features for dashboard visualization regularly updated with the community.
- It provides pre-built dashboards and reports for SaaS solutions.
- Some more potential benefits of business intelligence tools include:
 - Driving new revenues.
 - It increases operational efficiency.
 - It optimizes internal business processes.
 - It improves decision making.
 - It is gaining a competitive advantage over business rivals.
 - It is used in spotting business problems that need to be addressed.
 - It can be used in assisting companies in the identification of market trends.



Importance of BI

- Business intelligence is used to improve all parts of a company by improving access to the firm's data and then using that data to increase profitability.
- Companies that practice BI can translate their collected data into insights their business processors.
- Then the insights can be used to create strategic business decisions that improve productivity and accelerate the growth.

Types of BI Tools

BI combines a broad set of data analysis applications that includes:

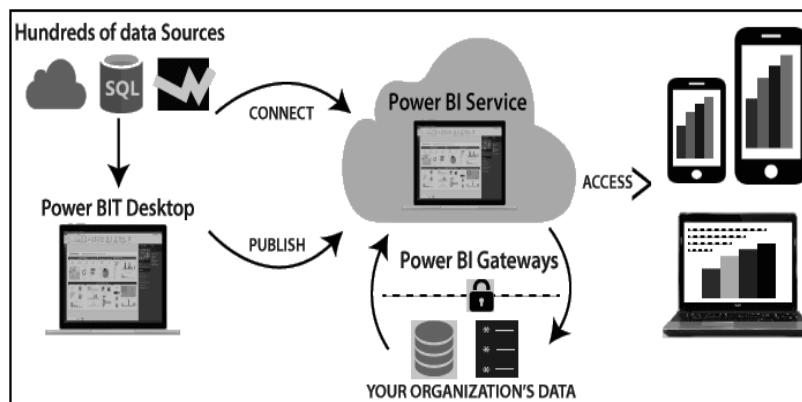
- Mobile BI
- Real-time BI
- Operational BI
- Open-source BI (OSBI)
- Collaborative BI
- Location intelligence (LI)
- Software-as-a-service BI (SaaS BI)
- Online analytical processing (OLAP)
- Ad hoc analytics

What is Power BI?

- Power BI is a Data Visualization, and Business Intelligence tool which helps to convert data from different data sources into interactive dashboards and BI reports.
- The Power BI tool is the collection of apps, data connectors, and software services which are used to get the data from different data sources, transforms data, and produces useful reports.
- It provides interactive visualizations with self-service business intelligence capabilities where end users can create reports and dashboards by themselves, without having to depend on information technology staff or database administrators.
- Power BI services are based on SaaS and mobile Power BI apps that are available for different platforms.
- These set of services are used by business users to consume data and to build BI reports.

Power BI Versions

- Different Power BI version like Desktop, Service-based (SaaS), and mobile Power BI apps are used in different platforms.
 - Power BI desktop app is used to create reports
 - Power BI Service (Software as a Service - SaaS) is used to publish those reports
 - Power BI mobile app is used to view the reports and dashboards.



Key Features of Power BI

Microsoft has added a number of data analytics features to Power BI since its inception, and continues to do so. Some of the most important features include:

Artificial Intelligence

Users can access image recognition and text analytics in Power BI, create machine learning models using automated machine learning capabilities and integrate with Azure Machine Learning.

Hybrid deployment support

This feature provides built-in connectors that allow Power BI tools to connect with a number of different data sources from Microsoft, Salesforce and other vendors.

Quick Insights

This feature allows users to create subsets of data and automatically apply analytics to that information.

Common data model support

Power BI's support for the common data model allows the use of a standardized and extensible collection of data schemas (entities, attributes and relationships).

Cortana integration

This feature, which is especially popular on mobile devices, allows users to verbally query data using natural language and access results, using Cortana, Microsoft's digital assistant.

Customization

This feature allows developers to change the appearance of default visualization and reporting tools and import new tools into the platform.

APIs for integration

This feature provides developers with sample code and application performance interfaces (APIs) for embedding the Power BI dashboard in other software products.

Self-service data prep

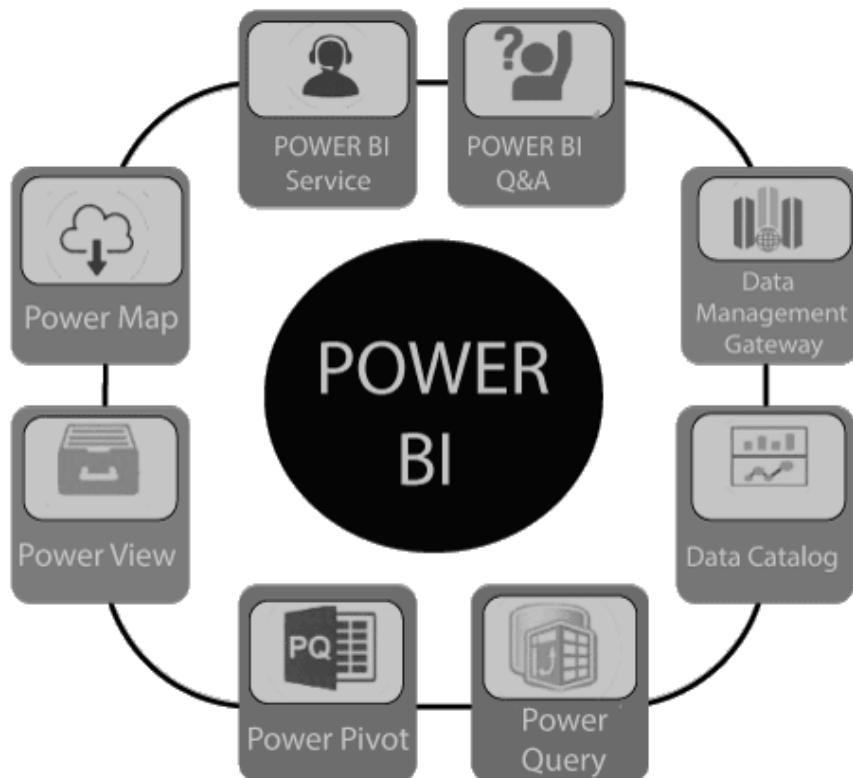
Using Power Query, business analysts can ingest, transform, integrate and enrich big data into the Power BI web service. Ingested data can be shared across multiple Power BI models, reports and dashboards.

Modelling view

This allows users to divide complex data models by subject area into separate diagrams, multiselect objects and set common properties, view and modify properties in the properties pane, and set display folders for simpler consumption of complex data models.

6.2 Components of Power BI

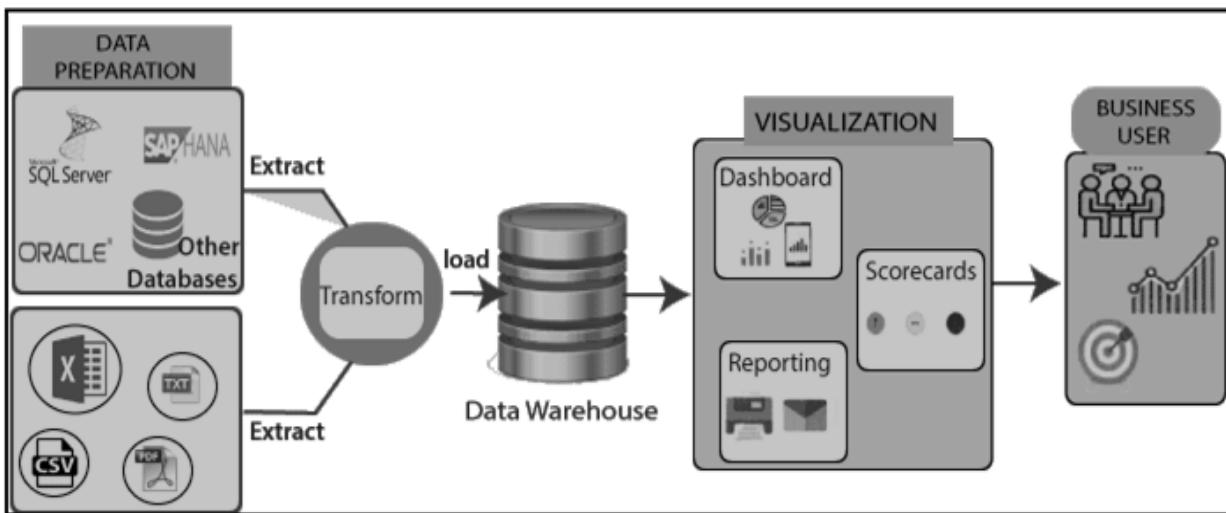
The components of Power BI are shown as below:



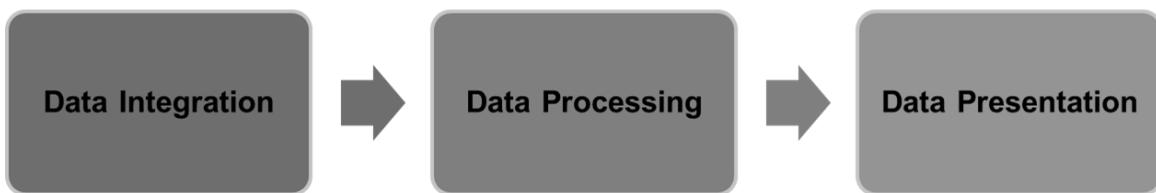
- 1. Power Query:** It is used to access, search, and transform public and internal data sources.
- 2. Power Pivot:** Power pivot is used in data modeling for in-memory analytics.
- 3. Power View:** By using the power view, you can analyze, visualize, and display the data as an interactive data visualization.
- 4. Power Map:** It brings the data to life with interactive geographical visualization.
- 5. Power BI Service:** You can share workbooks and data views which are restored from on-premises and cloud-based data sources.
- 6. Power BI Q&A:** You can ask any questions and get an immediate response with the natural language query.
- 7. Data Management Gateway:** You get periodic data refreshers, expose tables, and view data feeds.
- 8. Data Catalogue:** By using the data catalogue, you can quickly discover and reuse the queries.

6.3 Architecture of Power BI

The architecture of Power BI is shown as below:



Power BI architecture has three phases. The first two phases use ETL (extract, transform, and load) process to handle the data.



1. Data Integration

- An organization needs to deal with the data that comes from different sources.
- First, extract the data from different sources which can be your separate database, servers, etc.
- Then the data is integrated into a standard format and stored at a common area that's called staging area.

2. Data Processing

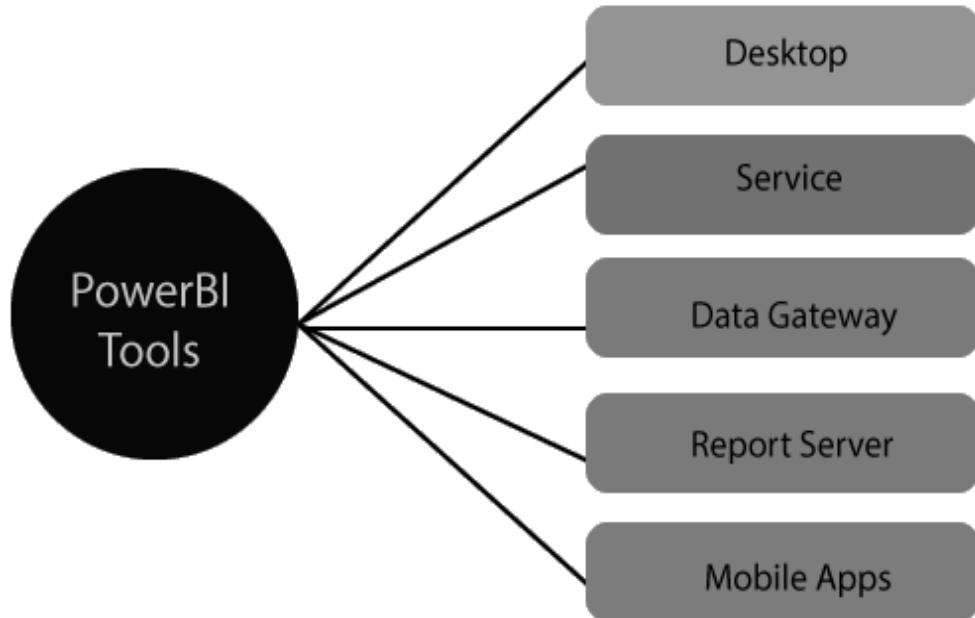
- Still, the integrated data is not ready for visualization because the data needs processing before it can be presented.
- This data is pre-processed. For example, the missing values or redundant values will be removed from the data sets.
- After that, the business rules will be applied to the data, and it transforms into presentable data.
- Then this data will be loaded into the data warehouse.

3. Data presentation

- Once the data is loaded and processed, then it can be visualized much better with the use of various visualization that Power BI offers.
- By using of dashboard and reports, we represent the data more intuitively.
- These visual reports help business end-users to take business decision based on the insights.

6.4 Power BI Tools

Here are some essential tools of Power BI, as shown below:



Power BI Desktop

- It is a primary authoring and publishing tool.
- Power BI users and developers use it to create brand new models and reports.
- Power BI Desktop tool is available at free of cost.

Power BI Service

- The Power BI data modules, dashboards, and reports are hosted in the online software as a service (SaaS).
- Sharing, administration, and collaboration happen in the cloud.
- Power BI Service tool is available at the pro license, and the user has to pay \$10 per month in 2022.

Power BI Data Gateway

- It works as the bridge between the Power BI service on-premises data sources such as Import, Direct Query, and Live Query.
- BI Admin installs Power BI.

Power BI Report Server

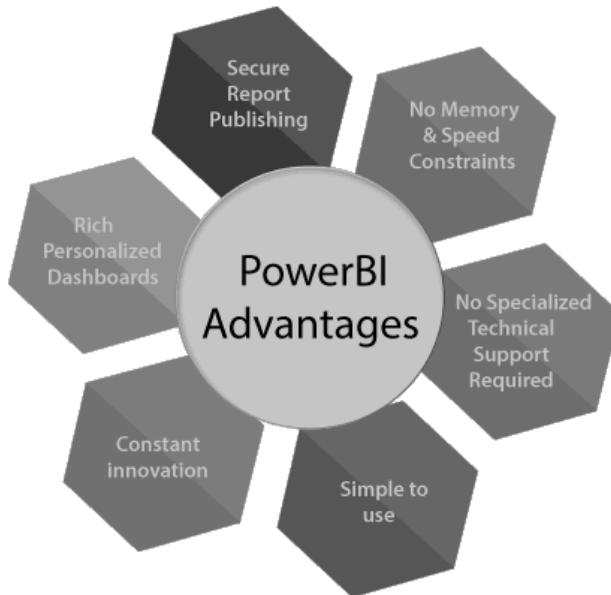
- It hosts paging reports, mobile reports, KPIs, and Power BI desktop reports.
- It requires updates in every four months and managed by the IT team.

Power BI Mobile Apps

- It is available for Android, iOS, and Windows.
- Microsoft Intune manages it by using this tool.
- You can view reports and dashboards on the Power BI Service Report Server.

6.5 Power BI Advantages and Disadvantages

Advantages



- 1. Secure Report Publishing:** You can automate setup data refresh and publish reports that allowing all the users to avail the latest information.
- 2. No Memory and Speed Constraints:** To Shift an existing BI system into a powerful cloud environment with Power BI embedded eliminates memory. Speed constraints ensure that data is quickly retrievable and analyzed.
- 3. No Specialized Technical Support required:** The Power BI provides quick inquiry and analysis without the need for specialized technical support. It also supports a powerful natural language interface and the use of intuitive graphical designer tools.
- 4. Simple to Use:** Power BI is simple to use. Users can easily find it only on behalf of a short learning curve.
- 5. Constant innovation:** The Power BI product is updated in every month with new functions and features.
- 6. Rich, personalized dashboard:** The crowning feature of Power BI is the information dashboards that can be customized to meet the exact need of any enterprise. You can easily embed the dashboards, and BI reports in the applications to provide a unified user experience.

Disadvantages

Here are some disadvantages of Power BI, as shown below:

1. Dashboards and reports are only shared with the users who are having the same email domains.
2. Power BI will not merge imported data that is accessed from real-time connections.
3. Power BI only accepts the file size maximum 250 Mb and the zip file which is compressed by the data of the x-velocity in-memory database.
4. Dashboard never accepts or pass user, account, or any other entity parameters.
5. Very few data sources permit real-time connections to Power BI reports and dashboards.

6.6 Download and Install Power BI Desktop

Here are some requirements of the system to download the Power BI Desktop:

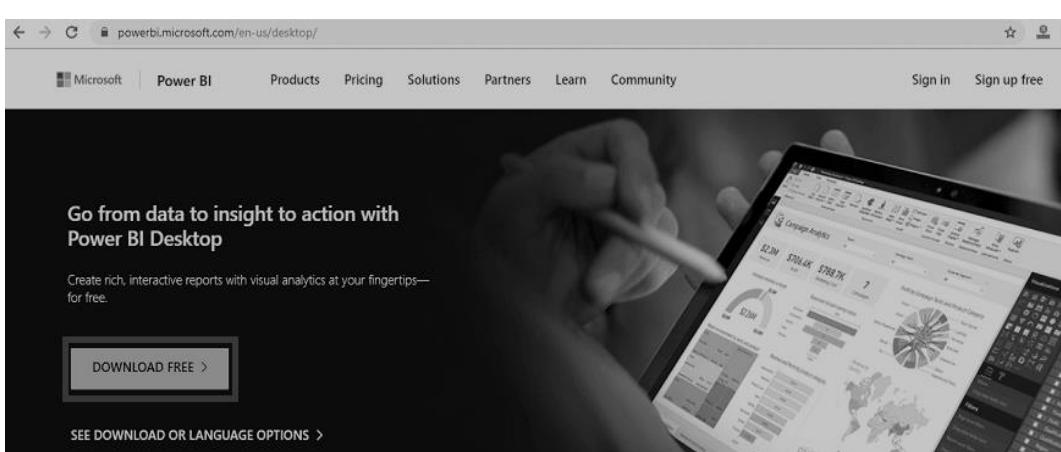
- Window 7, window 8, window 8.1, window 10, and windows server 2008 R2, windows server 2012, windows server 2012 R2.
- It requires internet explorer 9 or higher.
- Power BI Desktop is available for both 32 bit and 64-bit platforms.

Let's see the downloading process of the Power BI Desktop step by step:

Step 1: Click on the below link to directly download Power BI Desktop.

<https://powerbi.microsoft.com/en-us/desktop/>

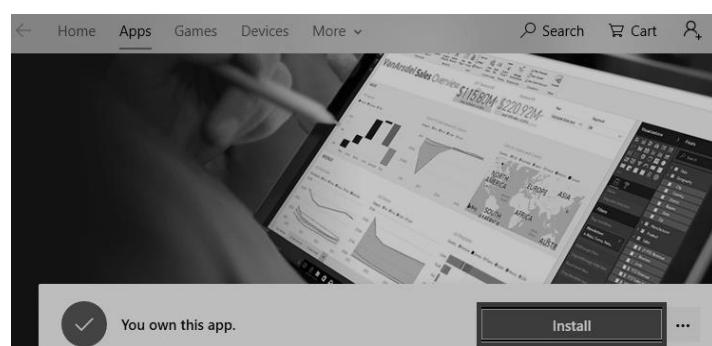
Step 2: Then click on the **Download Free** button.



Step 3: Now, you will redirect to a **Microsoft Store** and then select the **Get** button.



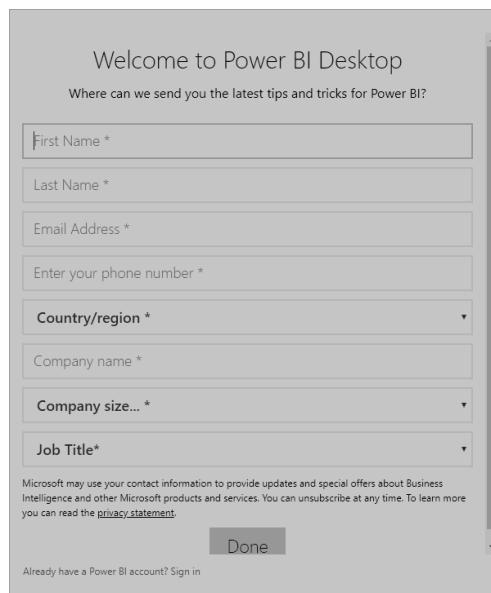
Step 4: Click on the **Install** button.



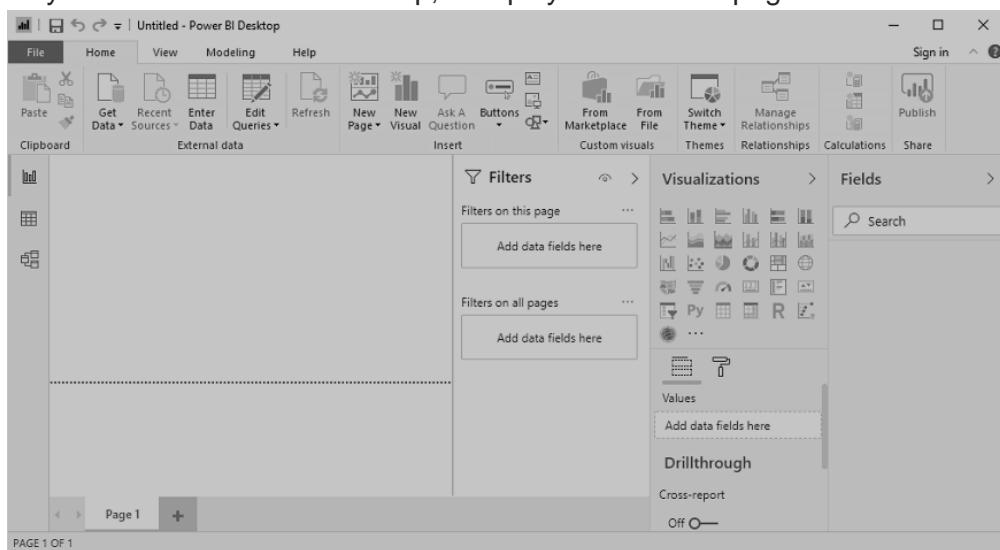
You can see the progress status of the Power BI Desktop on the screen.



Step 5: You can see "welcome to Power BI Desktop" screen and then register yourself on the desktop.

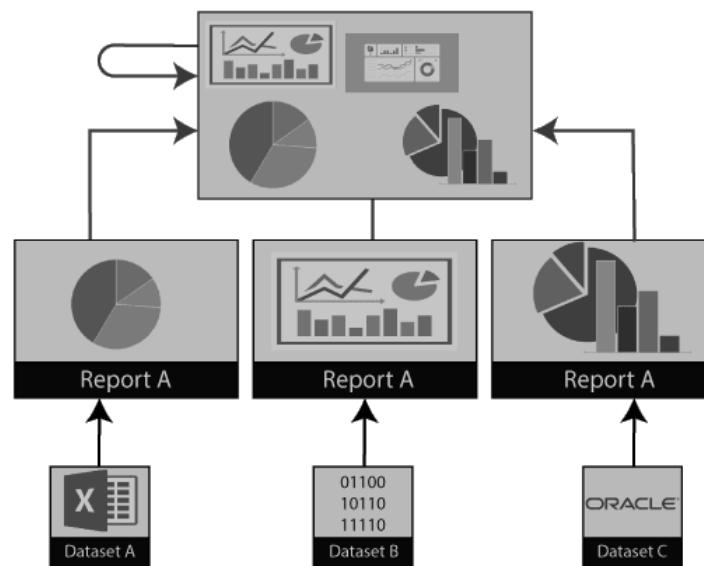


Step 6: When you run the Power BI desktop, it displayed the home page or welcome screen.



6.7 Power BI Dashboard

- Power BI dashboard is a single page, also called a canvas that uses visualization to tell the story.
- It is limited to one page; therefore, a well-designed dashboard contains only the most essential elements of that story.
- The visualizations visible on the dashboard are known as Tiles.
- These tiles are pinned to the dashboard from reports.
- The visualizations on a dashboard come from reports, and each report is based on one data set.
- A dashboard can combine on-premises and cloud-born data and they provide a consolidated view regardless of where the data lies.



Creating Dashboard in Power BI

We need to import one sample datasets of the Power BI and use it to create a new dashboard.

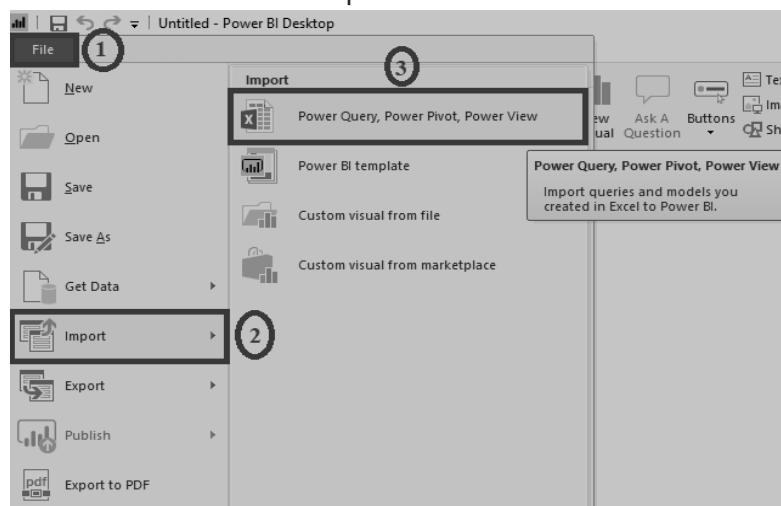
For example, suppose a sample such as **Procurement Analysis**. This sample is an excel workbook with two PowerView sheets.

When Power BI imports the workbook, it adds a dataset and a report to the workspace. Let's see step by step.

Step 1: Open the Power BI Desktop and click on the **File** pane.

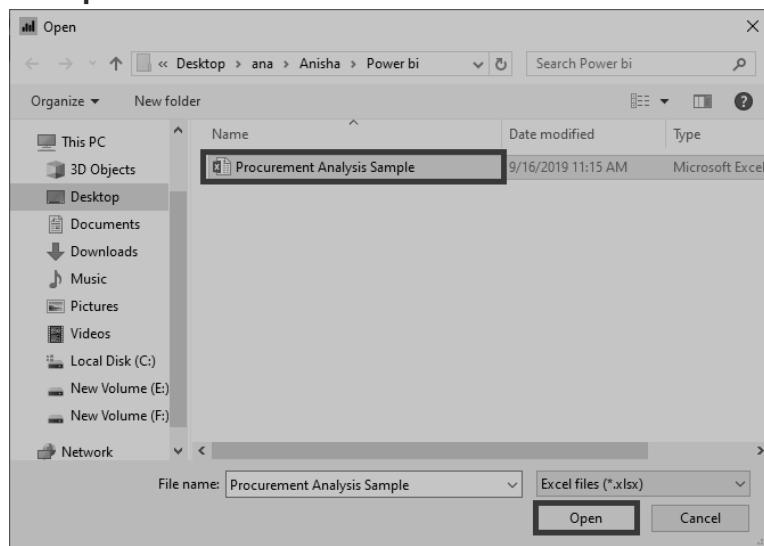
Step 2: Go to the **Import** option.

Step 3: And select the Excel dataset **file** to import the file.

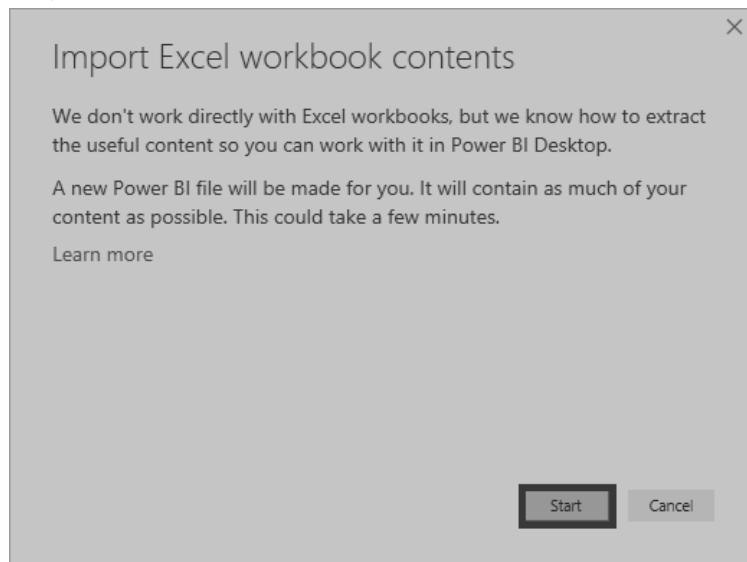


Step 4: Select the **procurement analysis** sample file.

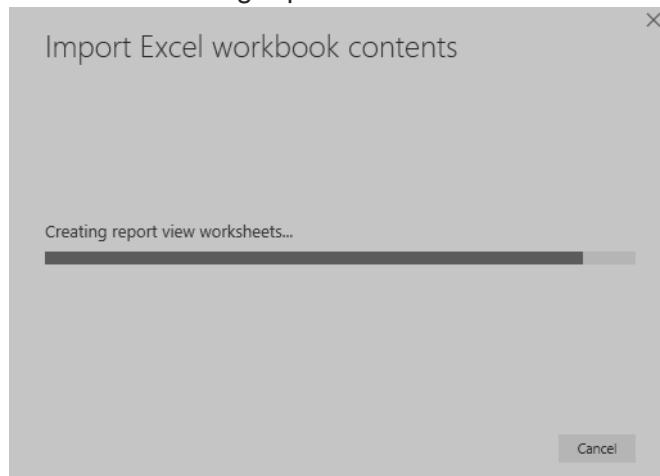
Step 5: And click on the **Open** button.



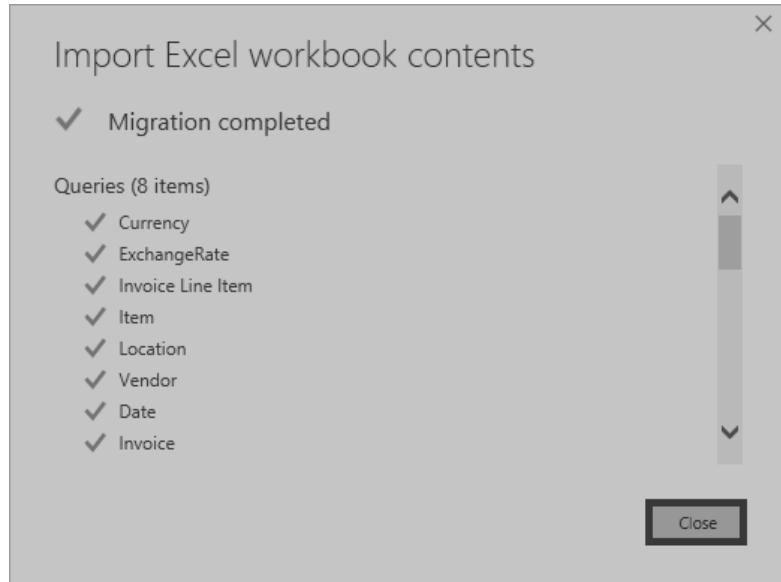
Step 6: For the exercise, select the **Start** button.



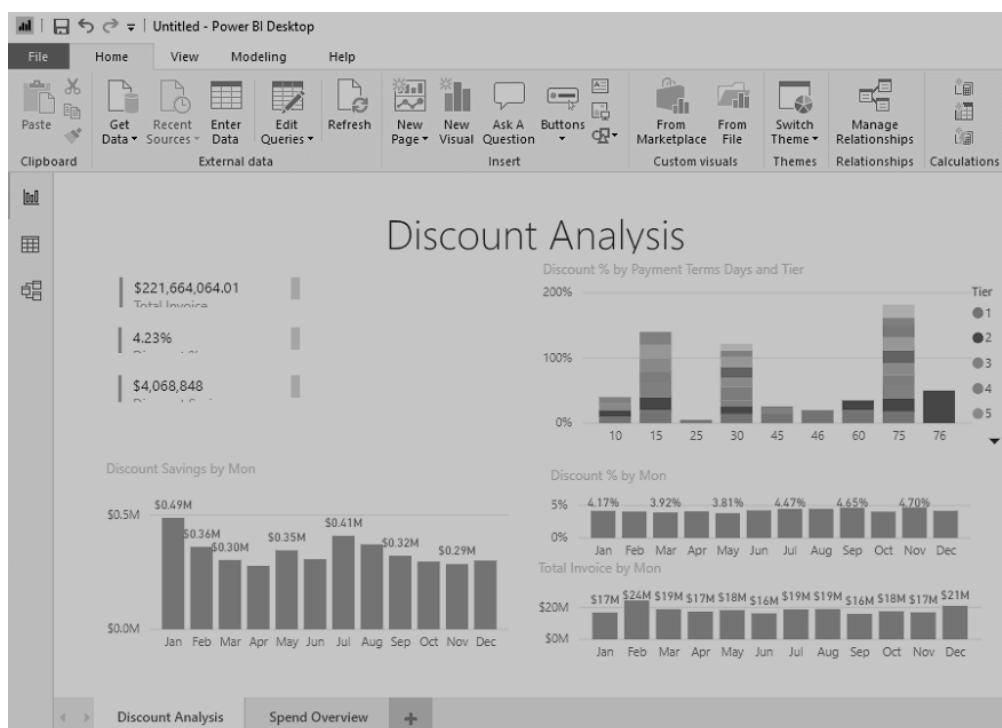
It starts import excel workbook and creating report view worksheets shown in the below screenshot.



Step 7: When the **completed** message appears, then select the **Close** button to dismiss it.



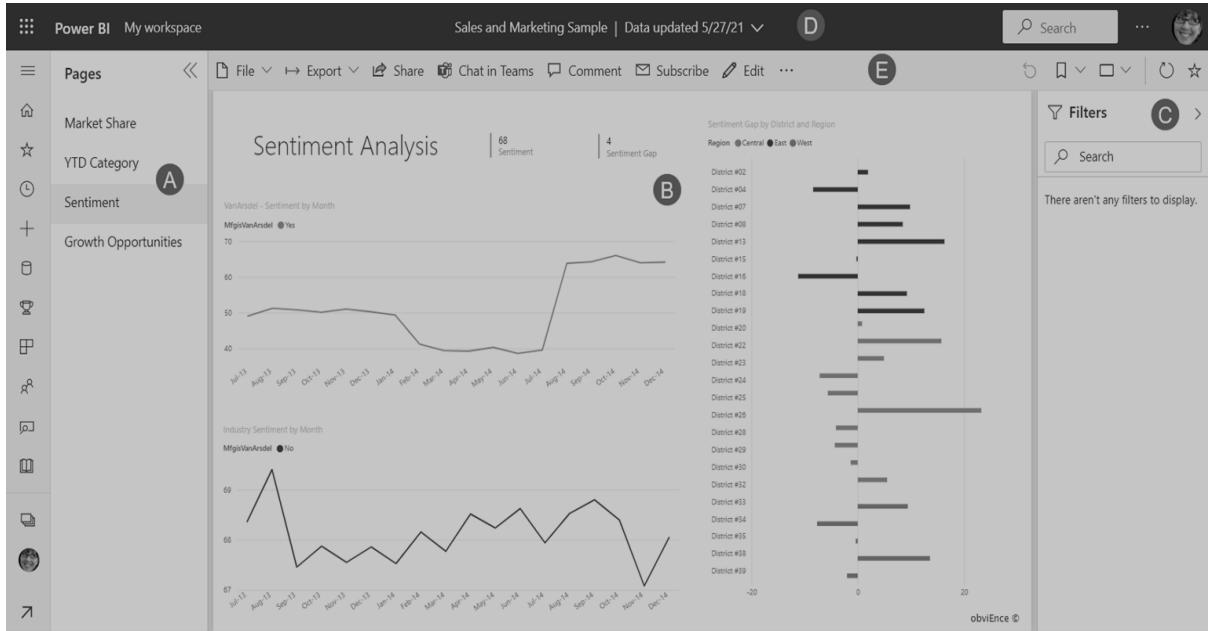
In the below screenshot, you can see the discount analysis of the imported dataset in the form of tiles.



6.8 Power BI Reports

- A Power BI report is a multi-perspective view into the dataset, with visualizations which represent different findings and insights from that dataset.
- A report can have a single visualization or multiple visualizations. The visualizations in a report represent something like a dashboard does but serve a different purpose.
- These visualizations are not static. These are highly interactive & highly customizable visualizations which update, as the underlying data changes. You can add and remove the data, change visualization types, and apply filters in your model to discover insights.

Parts of a Report



- This report has four pages (or tabs) and you're currently viewing the **Sentiment** page.
- On this page are five different visuals and a page title.
- The *Filters* pane shows us one filter applied to all report pages. To collapse the Filters pane, select the arrow (>).
- The Power BI banner displays the name of the report and the last updated date. Select the arrow to open a menu that also shows the name of the report owner.
- The action bar contains actions you can take on this report. For example, you can add a comment, view a bookmark, or export data from the report. Select **More options (...)** to reveal a list of additional report functionality.

6.9 Difference between Dashboards and Reports

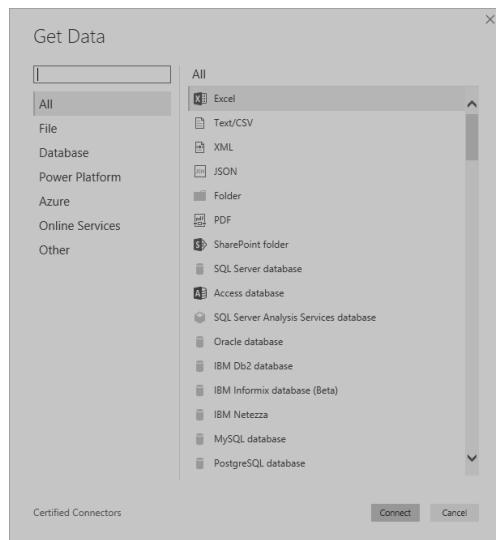
Dashboard and reports both terms are used interchangeably, but they are not synonymous. The below table compares the dashboard with the reports, such as:

Capabilities	Dashboards	Reports
Pages	It has only one page.	It can have one or more pages.
Data Sources	It has one or more reports and datasets per dashboard.	It has only a single dataset per report.
Pinning	It can pin existing visualizations only from the current dashboard to your other dashboards.	It can pin visualizations to any of the dashboards. And also, can pin entire report pages to any of the dashboards.
Filtering	It can't filter or slice.	It has many different ways to filter, highlight, and slice.
Feature	It can set one dashboard as the featured dashboard.	It cannot create a feature report.
Set alerts	No	Yes, it can set alerts.
Subscribe	We can't subscribe to a dashboard.	We can subscribe to report pages.
Available in Power BI desktop	No	Yes, it can create and view reports in desktop.
Change visualization type	No, if a report owner changes the visualization type in the report, the pinned visualization on the dashboard does not update.	Yes, it can change the visualization type.

6.10 Power BI Data Sources

- Power BI Desktop and Power BI Services support a large range of data sources. Click on the Get Data button, and it shows you all the available data connections.
- You can connect to different Flat files, Azure cloud, SQL database, and Web platforms, also such as Google Analytics, Facebook, and Salesforce objects.
- It includes an ODBC connection to connect to other ODBC data sources.
- Here are the available data sources in Power BI, as shown below:
 - SQL Database
 - Flat Files
 - Blank Query
 - OData Feed
 - Azure Cloud Platform
 - Online Services
 - Oracle database
 - IBM Db2 database
 - IBM Netezza
 - IBM Informix database (Beta)
 - Other data sources such as Exchange, Hadoop, or active directory

- To connect data in Power BI Desktop, you need to click on the Get Data button in the main screen. First, it shows you the most common data sources.
- Then click on the More option to see a full available list of the data sources.



- On the left side, it shows a category of all the available data sources. You also have an option to perform search operation at the top.

Let's see all the listed data sources in detail:

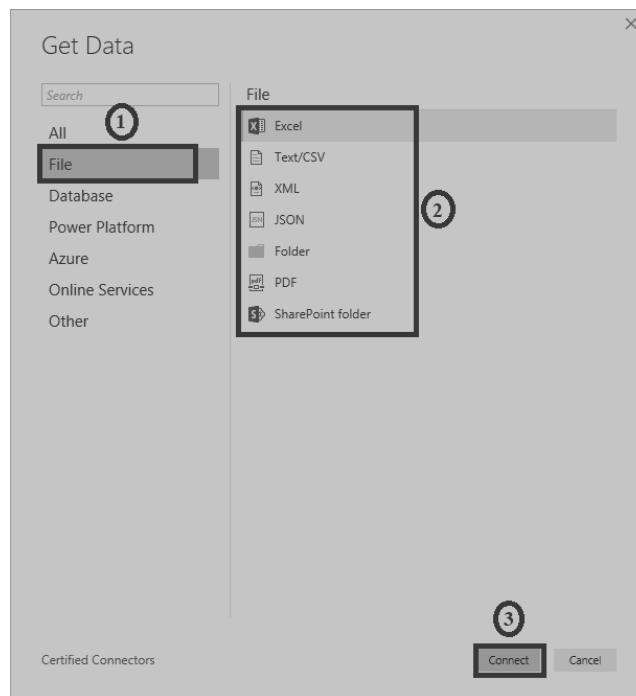
1. All

In this category, you can see all the available data sources of the Power BI desktop.

2. File

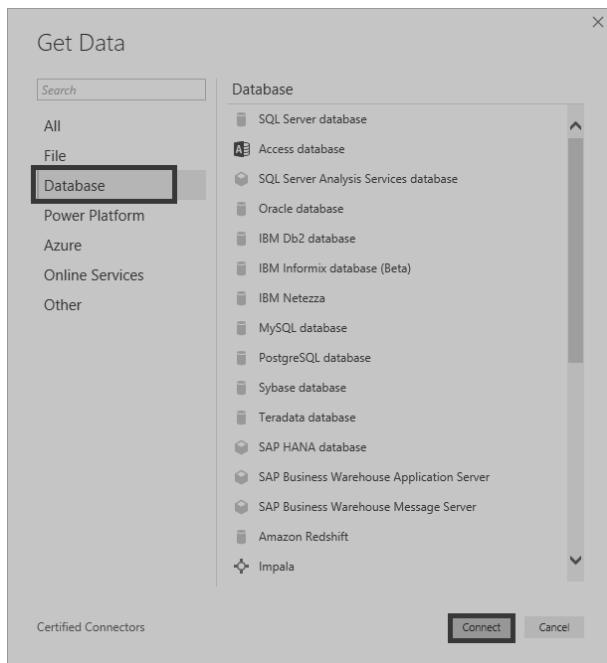
When you click on the **File** option, it shows you all the flat files supported in Power BI desktop. Select any file type from the list

3. click on the **Connect** button to connect that file.

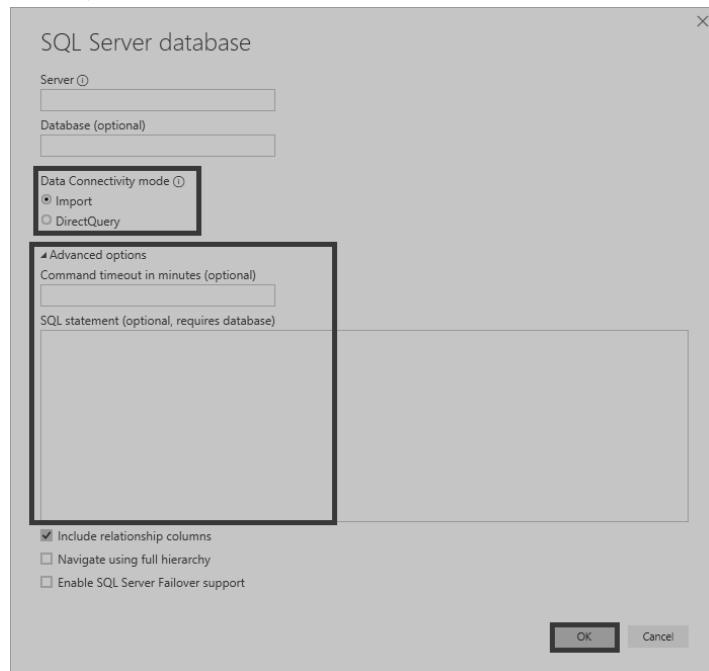


4. Database

When you click on the **Database** option, it shows you the list of all the database connections that you can connect to any database.

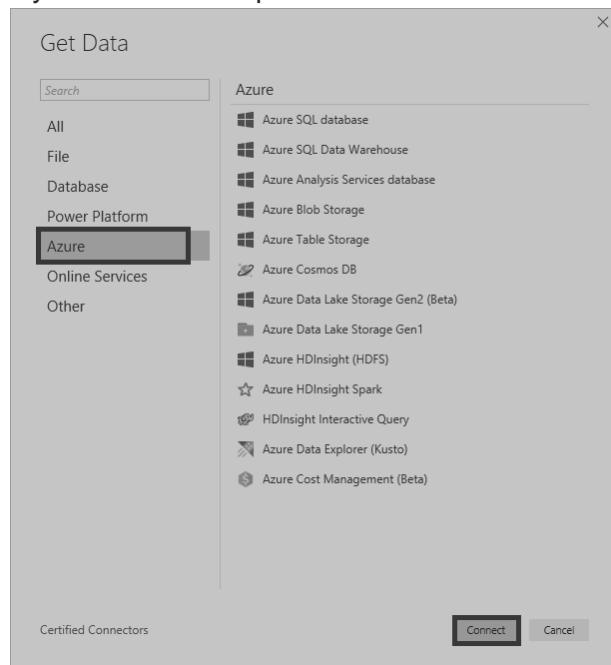


- You need to pass the server name, user name, and password to connect. Also, you can connect via a direct SQL query using the Advanced option.
- You can also select connectivity mode - Import or DirectQuery.
 - **Import:** Import method allows to perform data transformations and manipulation. When you publish the data to PBI service (limit 1 GB), it consumes and pushes data into Power BI Azure backend and data can be refreshed up to 8 times a day and a schedule can be set up for data refresh.
 - **DirectQuery:** It limits the option of data manipulation, and the data stays in the SQL database. The DirectQuery is live, and there is no need to schedule refresh as in the Import method.



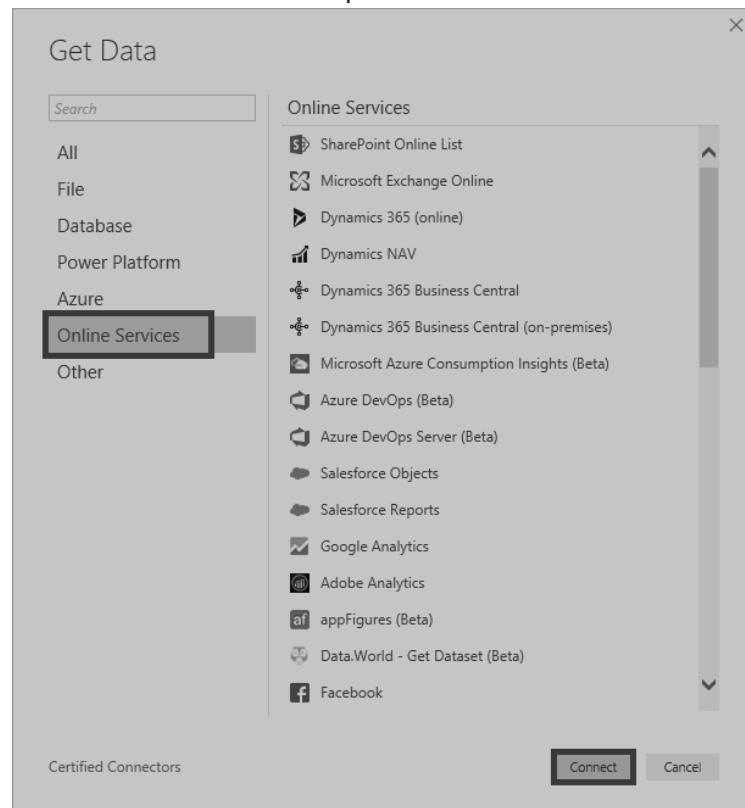
5. Azure

- Using the Azure option, you can connect with the database in the Azure cloud.
- Below screenshot shows you the various options available under the Azure category.



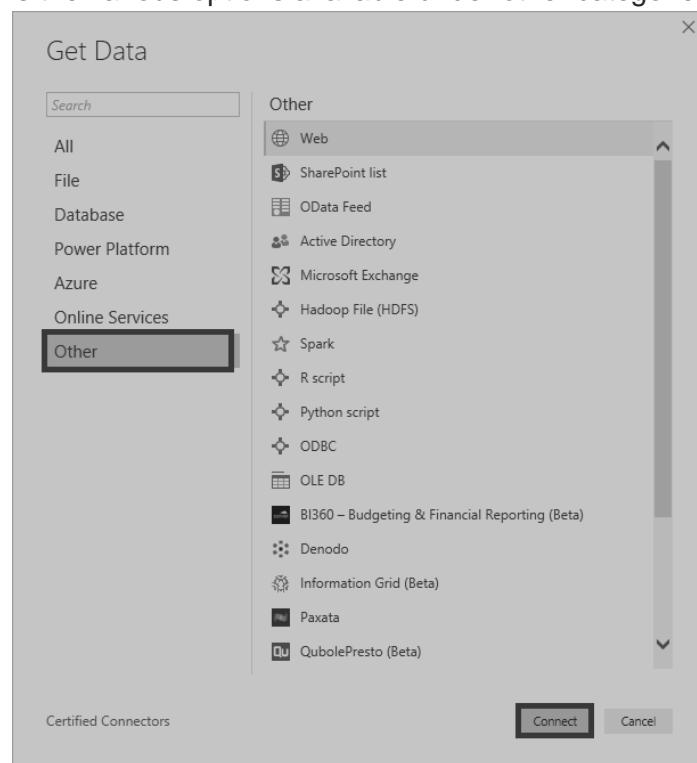
6. Online Services

- The Power BI also allows you to connect to different online services such as Exchange, Salesforce, Google Analytics, and Facebook.
- Following screenshots showed the various options available under Online Services.



7. Other

Below screenshot shows the various options available under other categories.



6.11 Power BI Embedded

The Power BI service (SaaS) and the Power BI Embedded service in Azure (PaaS) have APIs for embedding the dashboard and reports. When you are embedding the content, this gives you access to the latest Power BI features such as dashboards, gateways, and app workspaces.

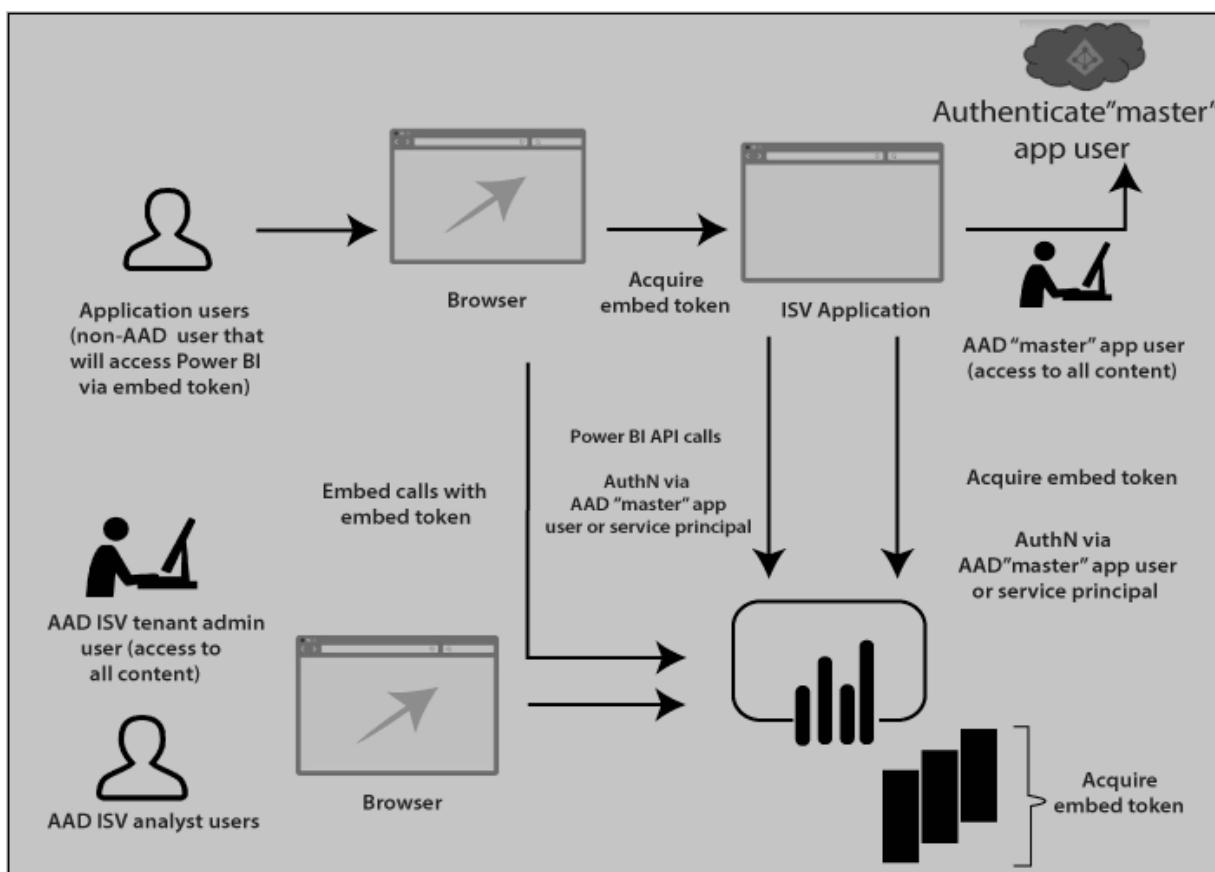
There are two scenarios for embedding Power BI content:

1. Embedding for organization's users

- (Who have Power BI license) It allows us to extend the Power BI service. It requires your application's user sign in to the Power BI service to view the content.
- After sign in, they only have access to dashboards and reports that they own or someone shared with them in the Power BI service.

2. Embedding for users and customers

- It allows you to embed the dashboards and reports for users who don't have a Power BI account. This type of embedding is also known as Power BI embedded.



- Power BI Embedded has benefits for an ISV, their developers, and the customers. For example, an ISV can start creating the visuals for free with Power BI Desktop.
 - By minimizing the visual analytic development efforts, ISVs achieve faster time to market and stand out from the competitors with differentiated data experiences.
 - Also, ISVs can opt to charge a premium for the additional value they create with embedded analytics.
 - With Power BI Embedded, your customers don't need to know anything about Power BI. You can use two different methods to create an embedded application:
 - Power BI Pro account
 - Service principle
 - The Power BI Pro account acts as the master account of your applications (think of it as a proxy account). This account allows generating embed tokens which provide access to your application's Power BI dashboards and reports.
 - Service principle can embed Power BI content into an application using an app-only token. It also allows generating embed tokens which provide access to your application's Power BI dashboards and reports.
- Note:** While embedding requires the Power BI service, customers do not need to have a Power BI account to view the application embedded content.

6.12 Power BI Gateway

- Power BI Gateway is a software which is required to access data situated in an on-premises network.
- Gateway plays a role like as a gatekeeper for the on-premises data source.
- If anyone wants to access on-premises data from the cloud or web-based app, that request goes through the gateway.
- The gateway attends all the connection requests, and access is granted based on their authentication and requirements.
- Gateway does not transfer the data from the on-premises source to the client platform. But it directly connects that platform to the on-premises data source.
- The client can directly access the data from its on-premises location to use it for making a dashboard, reports, and data analysis.
- Generally, a gateway is used to facilitate the connection between a single data source and multiple data source to the on-premises data source.

Types of the Power BI Gateway

There are two types of Power BI gateways:

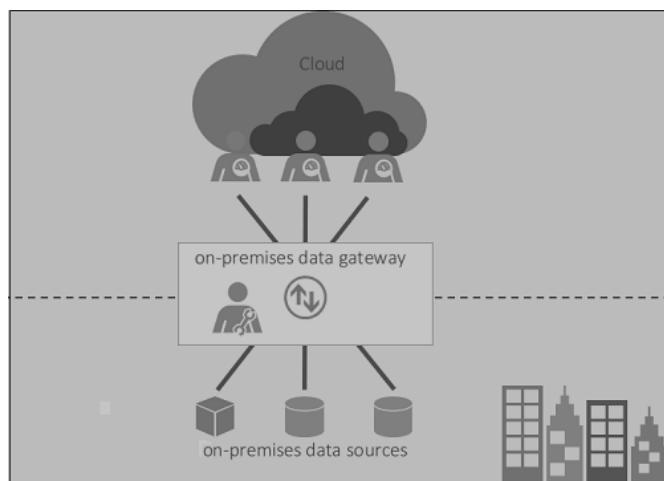
- On-premises Data Gateway (Standard Mode)
- On-premises Data Gateway (Personal Mode)

On-premises Data Gateway (Standard Mode)

- On-premises data gateway allows connection with the multiple on-premises data sources for more than one user.
- You can use the data in Power BI, Azure Logic Apps, Azure Analysis Services, PowerApps, Microsoft Flow, etc.
- You can establish direct connections to multiple data sources only installing this type of data gateway.
- This data gateway is very helpful for complex scenarios where multiple users need to access various data sources.

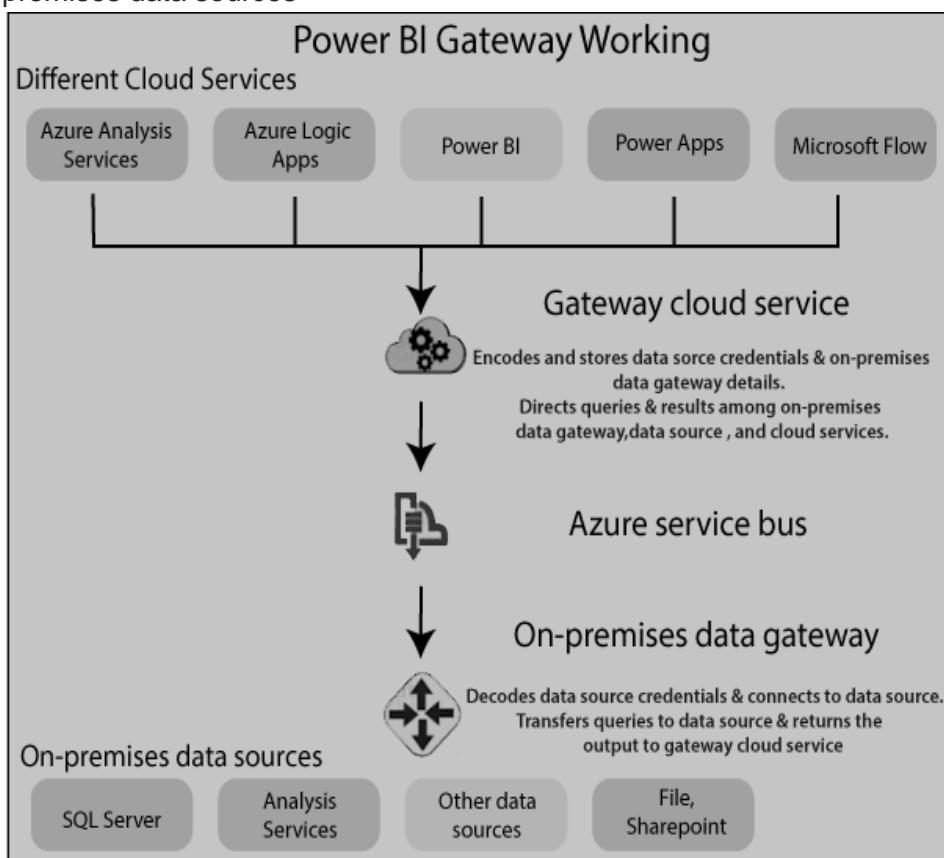
On-premises Data Gateway (Personal Mode)

- The particular mode of on-premises data gateway allows only one user to connect to different data sources.
- It is helpful when only one person needs to access the data sources.
- To create reports and the dashboards using Power BI, the user cannot share its access privilege with other users.



Power BI Gateway Architecture

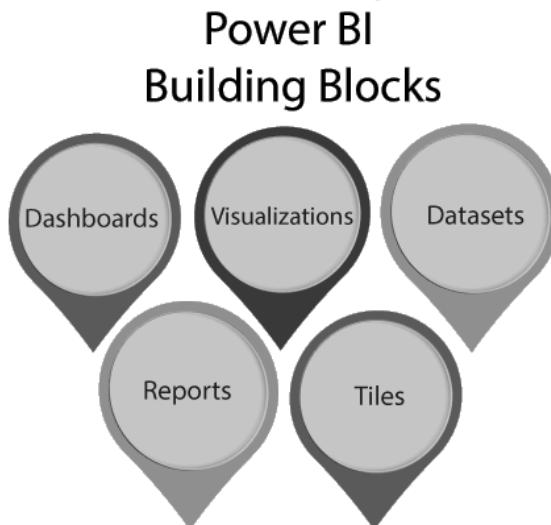
- Power BI gateway functions play a role as a mediator between the cloud services and on-premises data sources.
- The On-premises data gateways run as a Windows service. This Windows service gets registered with the Gateway Cloud Service through Azure Service Bus.
- The architecture and working of the data gateway with the help of the diagram given below.
- Power BI gateway architecture is divided into three parts:
 1. Cloud services
 2. Gateway services
 3. On-premises data sources



- According to Power BI, a cloud service creates a query which requires data from an on-premises data source.
- This query from cloud services goes to the gateway cloud service with encrypted credentials.
- The gateway cloud services process and analyze the request and then forward it to the Azure service bus.
- You don't need to configure azure service bus separately because Power BI manages it by default.
- The Azure service bus keeps all the requests to be sent forward to the on-premises data gateway.
- The on-premises data gateway decrypted credentials for the data source and connect the user to the data source.
- The on-premises data gateway forwards the query sent from the cloud service to the on-premises data source.
- The data query is executed at a data source that can be SQL Server, SharePoint, files, SSAS, etc.
- Result of the query is returned to On-premises data gateway by the data source.
- The On-premises data gateway sends the result back to the cloud service via Azure Service Bus.

6.13 Building Blocks of Power BI

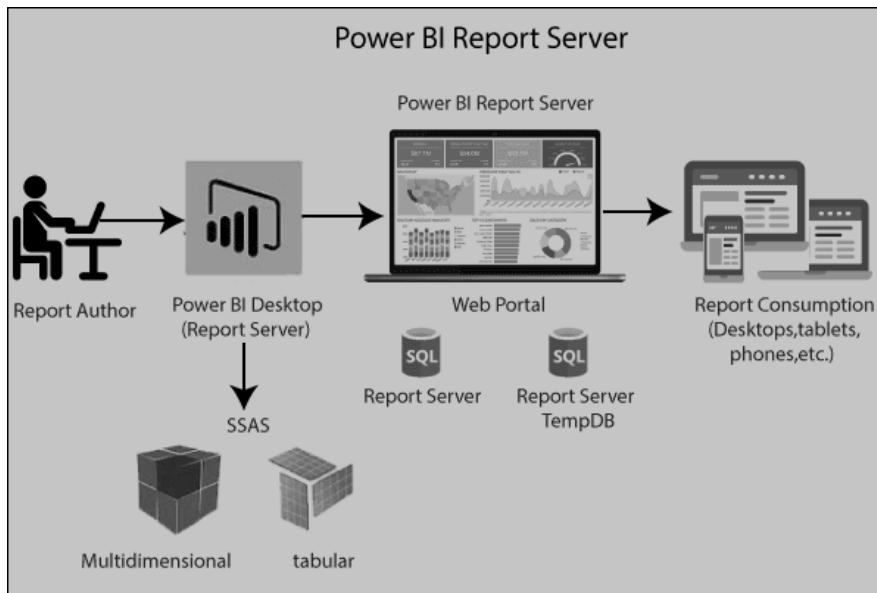
There are five building blocks, such as **Visualization**, **Reports**, **Dashboards**, **Datasets**, and **Tiles**.



- **Visualization:** The visualization is a type of chart or visuals that built by the Power BI designers. These visuals display the data from the datasets and report.
For example, line graph, pie chart, bar charts, and other graphical representation of the source data on a top geographical map, etc.
- **Reports:** A report is a collection of one or more pages of interactive visuals, text, and graphics that together make a single report.
For example, state, city report, sales by country, profit by-products report, logistic performance report, etc.
- **Dashboards:** Dashboard is a single layer presentation of multiple visualizations with interactive visuals, text, and graphics. A dashboard collects the most important metrics, on one screen, to tell a story or answer a question. The dashboard content comes from one or more datasets and one or more reports.
For example, pie charts, bar charts, and geographical maps.
- **Datasets:** The dataset is a collection of data which is used to create its visualization in Power BI.
For example, Oracle or SQL servers tables and excel sheets.
- **Tiles:** The tile is a single visualization in the report or on the dashboards.
For example, the pie chart in reports or dashboard.

6.14 Power BI Report Server

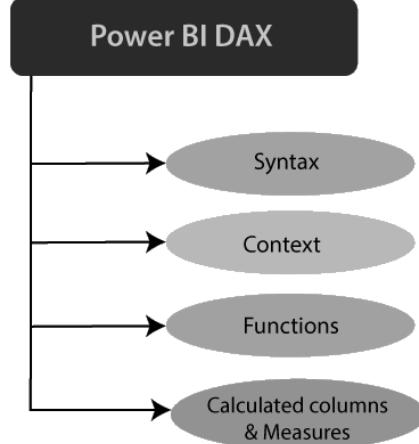
- Power BI Report Server is an on-premises report server with a web portal in which you display and manage reports and KPIs.
- Along with it come to the tools to create Power BI reports, mobile reports, paginated reports, and KPIs.
- Users can access those reports in different ways: viewing them on a web browser on any device, or as an email in their inbox.



- Power BI Report Server is a specific edition of SQL Server Reporting Services that can host Power BI reports.
- For running Power BI Report Server, you don't need to have SQL Server installation disk; the Report Server already comes with its setup files.
- You can download set up files. Power BI report server can host Power BI Reports as well as Reporting Services (SSRS) Reports.
- With Power BI report server, there will be an instance of Power BI Desktop installation.
- The Power BI Desktop edition that comes with the report server should be used to create Power BI reports. Otherwise, reports cannot be hosted on the report server.
- The Power BI Desktop report server edition is regularly updated, and its experience will be very similar to the Power BI Desktop.
- You can download the latest edition of Power BI report server from the below link. <https://powerbi.microsoft.com/en-us/report-server/>

6.15 Power BI DAX

- DAX (Data Analysis Expressions) is a formula expression language.
- It can be used in different BI and visualization tools. DAX is also known as function language in which the full code is kept inside a function.
- DAX programming formula contains two data types such as **Numeric** and **Other**.
Numeric includes currency, integers, and decimals, where **Other** includes string and a binary object.



How does it work?

For understanding the Power BI DAX, it has main three fundamental concepts such as:

- Syntax
- Context
- Functions

1. Syntax

the syntax consists of various components that make up a formula.

Total Sales = SUM (Sales [SalesAmount])

- Total Sales is the measure name.
- The equal sign (=) operator indicates the beginning of the formula.
- The DAX function SUM adds up all the numbers in the Sales[SalesAmount] column.
- Parentheses () surround an expression containing one or more arguments.
- All function requires at least one argument. An argument passes a value to a function.
- The reference table Sales.
- The referenced column [SalesAmount] in the Sales table.
- With this argument, the SUM function knows on which column to aggregate a SUM.

2. Context

- Context is one of the essential concepts of DAX.
- It is categorized into two parts;
 - Row context
 - Filter context
- The **Row-Context** is the easiest thought of as the current row.
- It applies whenever a formula has a function which uses the filters to identify a single row in a table
- The **Filter context** is a little more challenging to understand than the Row context.
- You can most easily think of the Filter-Context as one or more filters applied in a calculation.
- The Filter-Context doesn't exist in the Row-context's stead.
- Instead, it uses in addition to the former. Look at the following DAX formula.

3. Functions

- Functions are predefined and ordered formula.
- They can perform calculations using arguments passed on to them.
- These arguments can be text, numbers, logical values, or other functions.
- Some important DAX Functions:
 - Aggregate Function
 - Count Function
 - Date Time Function
 - Logical Function
 - Text Function

Types of Functions

Here are some important DAX functions:

a) Aggregate Functions

MIN

This DAX function returns the minimum numeric value in a column, or between the two scalar expressions.

Syntax

`MIN(<column>)`

MAX

This DAX function returns the maximum value in a column, including any logical values and numbers represented as text.

Syntax

`MAX(<column>)`

AVERAGE

This DAX function returns the arithmetic mean of the values in a column.

Syntax

`AVERAGE(<column>)`

SUM

This DAX function adds all the numbers in a column.

Syntax

`SUM(<column>)`

b) Count Function

COUNT

This DAX function is used to return the count of items in a column. If there are multiple numbers of the same thing, this function will count it as separate items and not a single item.

Syntax

`COUNT(<column>)`

DISTINCTCOUNT

This DAX function is used to return the distinct count of items in a column. If there are multiple numbers of the same thing, this function will count it as a single item.

Syntax

`DISTINCTCOUNT(<column>)`

c) **Date time Function****DATE**

This DAX function returns the specified date in Date-Time format.

Syntax

DATE(<year>, <month>, <day>)

HOUR

This DAX function returns the specified hour as a number from 0 to 23 (12:00 A.M. to 11:00 P.M.).

Syntax

HOUR(>datetime<)

d) **Logical Function****AND**

This DAX function performs logical AND(conjunction) on two expressions. For AND to return true, both conditions specified have to be fulfilled.

Syntax

AND(<logical argument1>,<logical argument2>)

OR

This DAX function performs logical OR(disjunction) on two expressions. For OR to return true, either of the two conditions specified has to be fulfilled.

Syntax

OR(<logical argument1>,<logical argument2>)

NOT

This DAX function performs logical NOT (negation) on given expression.

Syntax

NOT(<logical argument>)

e) **Text function****CONCATENATE**

This DAX function joins two text strings into one text string.

Syntax

CONCATENATE(<text1>, <text2>)

FIXED

This DAX function rounds a number to the specified number of decimals and returns the result as text.

Syntax

FIXED(<number>, <decimals>, <no_commas>)

REPLACE

This DAX function replaces part of a text string, based on the number of characters you specify, with a different text string.

Syntax

REPLACE(<old_text>, <start_num>, <num_chars>, <new_text>)

Calculated Columns and Measures

The Power BI DAX formulae are used in calculations, in Measures and Calculated Columns.

Calculated Columns

- When you create a data model on the Power BI Desktop, you can extend a table by creating new columns.
- The content of the columns is defined by a DAX expression, evaluated row by row or in the context of the current row across that table.

Measures

- There is another way of defining calculations in a DAX model, useful if you need to operate on aggregate values instead of on a row-by-row basis.
- These calculations are measures. One of the requirements of DAX is a measure that needs to be defined in a table.
- The action does not belong to the table. So, you can move a measure from one table to another one without losing its functionality.

6.16 Who Uses Power BI?

- Though Power BI is a self-service BI tool that brings data analytics to employees, it's mostly used by data analysts and business intelligence professionals who create the data models before disseminating reports throughout the organization.
- However, those without an analytical background can still navigate Power BI and create reports.
- Microsoft Power BI is used by both department reps and management, with reports and forecasts created to aid sales and marketing reps, while also providing data for management on how the department or individual employees are progressing toward their goals.
- In addition, Power BI offers an admin portal for administrators to help configure the implementation of Power BI, as well as usage monitoring and licenses.
- Some professionals who use Power BI tool are listed below:
 - Business and Data Analyst
 - Project and Portfolio Manager
 - IT team and IT Professionals
 - Developers and Database Administrator
 - Data Scientist
 - Consumer for end-user report

Section 3: Exercises

Exercise 1: On a plain paper, enlist Power BI Components.

Exercise 2: On a plain paper, enlist tools of Power BI.

Exercise 3: Match the column in following table.

Power BI Components	Application/Uses
1. Power Query	A. You get periodic data refreshers, expose tables, and view data feeds.
2. Power Pivot	B. It brings the data to life with interactive geographical visualization.
3. Power View	C. You can ask any questions and get an immediate response with the natural language query.
4. Power Map	D. Power pivot is used in data modelling for in-memory analytics.
5. Power BI Service	E. By using the data catalogue, you can quickly discover and reuse the queries.
6. Power BI Q&A	F. It is used to access, search, and transform public and internal data sources.
7. Data Management Gateway	G. You can share workbooks and data views which are restored from on-premises and cloud-based data sources.
8. Data Catalogue	H. By using the power view, you can analyze, visualize, and display the data as an interactive data visualization.

Exercise 4: Fill the following table.

Capabilities	Dashboards	Reports
Pages		
Data Sources		
Pinning		
Filtering		
Feature		
Set alerts		
Subscribe		
Available in Power BI desktop		
Change visualization type		

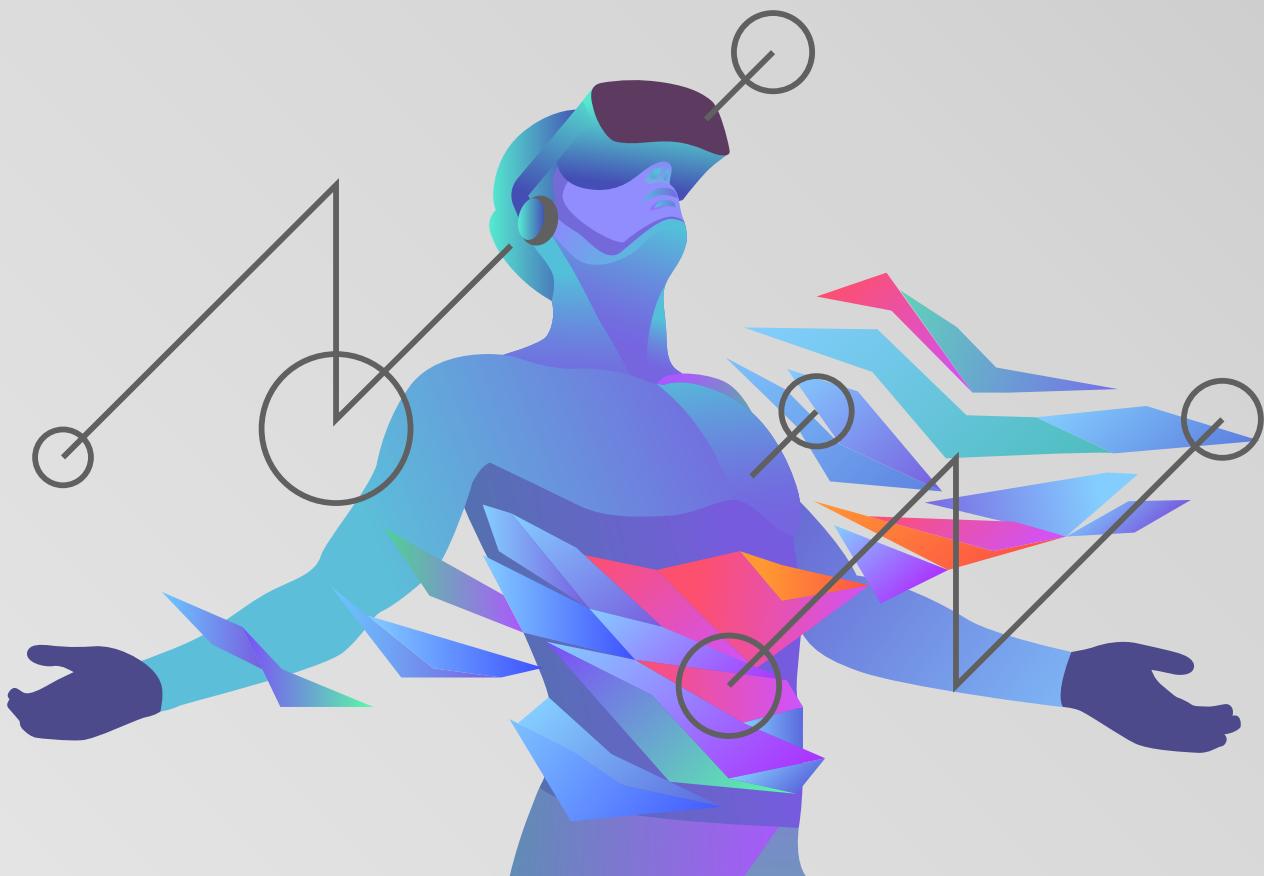
Exercise 5: Participate in a group discussion on following topics:

- a) Components and Architecture in Power BI
- b) Key Features of Power BI
- c) Advantages and Disadvantages of Power BI
- d) Power BI Data Sources
- e) Power BI Embedded and Gateway
- f) Building Blocks of Power BI

Section 4: Assessment Questionnaire

1. What are types of BI Tools?
2. What is Power BI?
3. What are the key features of Power BI?
4. What are three phases of Power BI architecture?
5. What are the advantages of Power BI?
6. Power BI dashboard is a single page, also called a _____ that uses visualization to tell the story.
7. What are various data sources in Power BI?
8. Name two scenarios for embedding Power BI content.
9. Power BI _____ is a software which is required to access data situated in an on-premises network.
10. What are two types of power gateway?
11. Power BI gateway functions play a role as a mediator between the cloud services and on-premises data sources. (True/False)
12. Power BI gateway architecture is divided into three parts, that are:
13. What are the five building blocks of Power BI?
14. Power BI Report Server is a specific edition of _____ Server Reporting Services that can host Power BI reports.
15. _____ is a formula expression language.
16. DAX programming formula contains two data types such as Numeric and Other.
 Numeric includes_____, where Other includes_____.
17. What are the fundamental concepts of Power BI DAX?
18. One of the requirements of DAX is a measure that needs to be defined in _____.

-----End of the Module-----



STL
academy

 **Empowering Youth!**

STL is one of the industry's leading integrators of digital networks providing All-in 5G solutions. Our capabilities across optical networking, services, software, and wireless connectivity place us amongst the top optical players in the world. These capabilities are built on converged architectures helping telcos, cloud companies, citizen networks, and large enterprises deliver next-gen experiences to their customers. STL collaborates with service providers globally in achieving a green and sustainable digital future in alignment with UN SDG goals. STL has a global presence in India, Italy, the UK, the US, China, and Brazil.



Skill & Assessment Partner
NASSCOM®

stl.tech | stlacad.tech