

# Introduction to Data Analytics

## Module 1

# Data Analytics Vs Data Science

- **Data analysts** examine large data sets to identify trends, develop charts, and create visual presentations to help businesses make more strategic decisions.
- **Data scientists**, on the other hand, design and construct new processes for data modeling and production using prototypes, algorithms, predictive models, and custom analysis.

# What Is Data Analytics?

- Data analytics is the science of analyzing raw data to make conclusions about that information.
- Data analytics help a business optimize its performance, perform more efficiently, maximize profit, or make more strategically-guided decisions.
- The techniques and processes of data analytics have been automated into mechanical processes and algorithms that work over raw data for human consumption.

# What Is Data Analytics?

- Various approaches to data analytics include descriptive analytics, diagnostic analytics, predictive analytics, and prescriptive analytics.
- Data analytics relies on a variety of software tools including spreadsheets, data visualization, reporting tools, data mining programs, and open-source languages.

# Types of Data Analytics

- **Descriptive analytics:** This describes what has happened over a given period of time. Have the number of views gone up? Are sales stronger this month than last?
- **Diagnostic analytics:** This focuses more on why something happened. It involves more diverse data inputs and a bit of hypothesizing. Did the weather affect beer sales? Did that latest marketing campaign impact sales?

# Types of Data Analytics

- Predictive analytics: This moves to what is likely going to happen in the near term. What happened to sales the last time we had a hot summer? How many weather models predict a hot summer this year?
- Prescriptive analytics: This suggests a course of action. For example, we should add an evening shift to the brewery and rent an additional tank to increase output if the likelihood of a hot summer is measured as an average of these five weather models and the average is above 58%,

# Data Collection

- This step involves collecting or gathering data and information from across a broad spectrum of sources. Various forms of information are then recreated into the same format so they can eventually be analyzed. The process can take a good bit of time, more than any other step.
- Collecting new data from internet and other sources
  - Using the previously collected and stored data
  - Reusing someone else's data
  - Purchasing data

# Data Cleaning

- Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.
- Data cleaning is not simply about erasing information to make space for new data, but rather finding a way to maximize a data set's accuracy without necessarily deleting information.
- This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results.

# Data Exploration

- **Data exploration** is an approach similar to initial data analysis, whereby a data analyst uses visual exploration to understand what is in a dataset and the characteristics of the data, rather than through traditional data management systems.
- These characteristics can include size or amount of data, completeness of the data, correctness of the data, possible relationships amongst data elements or files/tables in the data.

# Data Preprocessing

- Data preprocessing can refer to manipulation, filtration or augmentation of data before it is analyzed, and is often an important step in the data mining process.
- Data collection methods are often loosely controlled, resulting in out-of-range values, impossible data combinations, and missing values, amongst other issues.

# Data Analysis

- Data analysis is the science of examining data to conclude the information to make decisions or expand knowledge on various subjects. It consists of subjecting data to operations.
- This process happens to obtain precise conclusions to help us achieve our goals, such as operations that cannot be previously defined since data collection may reveal specific difficulties.

# Data Interpretation

- Data interpretation is the process of reviewing data and arriving at relevant conclusions using various analytical research methods.
- Data analysis assists researchers in categorizing, manipulating data, and summarizing data to answer critical questions.

# **Introduction to Statistics**

## **Module 1**

# What is Statistics?

- Statistics is a branch of applied mathematics that involves the collection, description, analysis, and inference of conclusions from quantitative data.
- The mathematical theories behind statistics rely heavily on differential and integral calculus, linear algebra, and probability theory.

# Key Points

- Statistics is the study and manipulation of data, including ways to gather, review, analyze, and draw conclusions from data.
- The two major areas of statistics are descriptive and inferential statistics.
- Several sampling techniques can be used to compile statistical data, including simple random, systematic, stratified, or cluster sampling.
- Statistics are present in almost every department of every company and are an integral part of investing.

# Statistics

## DESCRIPTIVE STATISTICS

- . Arrangement of data
- . Presentation of data

graphical

numerical

## STATISTICAL INFERENCE

Gives methods of formulating conclusions concerning the object of studies (general population) based on a smaller sample

# Descriptive Statistics

- Descriptive statistics mostly focus on the central tendency, variability, and distribution of sample data.
- Central tendency means the estimate of the characteristics, a typical element of a sample or population. It includes descriptive statistics such as mean, median, and mode.

# Descriptive Statistics

- Variability refers to a set of statistics that show how much difference there is among the elements of a sample or population along the characteristics measured.
  
- It includes metrics such as range, variance, and standard deviation.

# Key Points

- Descriptive statistics summarizes or describes the characteristics of a data set.
- Descriptive statistics consists of three basic categories of measures: measures of central tendency, measures of variability (or spread), and frequency distribution.

# Inferential Statistics

- Inferential statistics enable you to draw inferences and make predictions based on your data, whereas descriptive statistics summarize the properties of a data collection.
- It is an area of mathematics that enables us to identify trends and patterns in a large number of numerical data.

# Types of Inferential Statistics

- Hypothesis testing.
- Regression analysis

# Hypothesis testing

- Testing hypotheses and drawing generalizations about the population from the sample data are examples of inferential statistics.
- Creating a null hypothesis and an alternative hypothesis, then performing a statistical test of significance are required.

# Z Test

- Null Hypothesis:  $H_0: \mu = \mu_0$
- Alternate hypothesis:  $H_1: \mu > \mu_0$
- Test Statistic:  $Z \text{ Test} = (\bar{x} - \mu) / (\sigma / \sqrt{n})$   
where,  $\bar{x}$  = sample mean  
 $\mu$  = population mean  
 $\sigma$  = standard deviation of the population  
 $n$  = sample size
- Decision Criteria: If the z statistic  $>$  z critical value, reject the null hypothesis.

# T Test

- Null Hypothesis:  $H_0: \mu = \mu_0$
- Alternate Hypothesis:  $H_1: \mu > \mu_0$
- Test Statistic:  $t = \bar{x} - \mu / s\sqrt{n}$
- The representations  $\bar{x}$ ,  $\mu$ , and  $n$  are the same as stated for the z-test. The letter “s” represents the standard deviation of the sample.
- Decision Criteria: If the t statistic  $>$  t critical value, reject the null hypothesis.

# F Test

- Null Hypothesis:  $H_0 : \sigma^2_1 = \sigma^2_2$
- Alternate Hypothesis:  $H_1 : \sigma^2_1 > \sigma^2_2$
- Test Statistic:  $f = \sigma^2_1 / \sigma^2_2$ , where  $\sigma^2_1$  is the variance of the first population, and  $\sigma^2_2$  is the variance of the second population.
- Decision Criteria: Deciding Criteria: Reject the null hypothesis if f test statistic > critical value.

# Random Variable

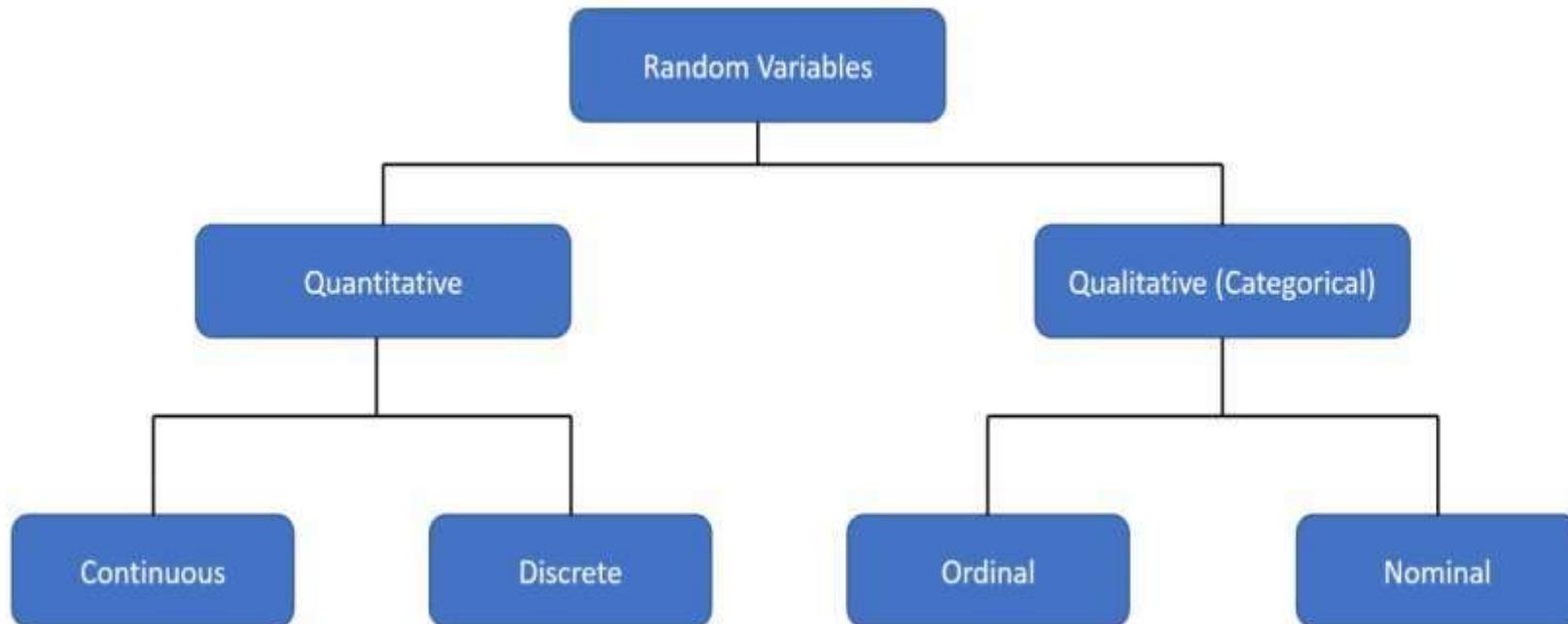
- A random variable is a variable whose value is unknown or a function that assigns values to each of an experiment's outcomes.
- Random variables are often designated by letters and can be classified as discrete, which are variables that have specific values, or continuous, which are variables that can have any values within a continuous range.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/module%201\(Introduction%20to%20Statistics/Randomness\\_and\\_Reproducibility.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/module%201(Introduction%20to%20Statistics/Randomness_and_Reproducibility.ipynb)

# Key Points

- A random variable is a variable whose value is unknown or a function that assigns values to each of an experiment's outcomes.
- A random variable can be either discrete (having specific values) or continuous (any value in a continuous range).

# Types of Random Variable



# Quantitative Random Variables

- Continuous Random Variable: The random variable that can be measured as a rational/decimal number are continuous random variables. For example, the height of a student, marks in an exam, etc.
- Discrete Random Variable: The random variable that can be measured as an integer number are discrete random variables. For example, the number of calls received in a day, the outcome of rolling a dice, etc.

# Qualitative Random Variables

- Ordinal Random Variable: If the outcomes of random experiments have inherent ordering, then the outcomes of those random experiments are called ordinal random variables.
- Nominal Random Variable: If a random variable is a name or label or category, that does not have order is called a nominal random variable..

# Mean

- The mean is the average of a set of numbers. To calculate the mean, begin by adding up all of the data points and dividing by the total number of data points.
- Formula:

$$\text{Mean } \bar{x} = \Sigma x_i / N$$

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/module%201\(Introduction%20to%20Statistics/Mean\\_Median\\_Mode.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/module%201(Introduction%20to%20Statistics/Mean_Median_Mode.ipynb)

# Median

- The median is the data point in the middle of a set. To find the median, the numbers in the set must be arranged from smallest to largest
- Formula :  
$$\text{Median} = (N+1)\text{th} / 2 \text{ term; when } N \text{ is odd}$$
$$[ N\text{th} / 2 \text{ term} + ( N / 2 + 1)\text{th} \text{ term } ] / 2 ; \text{ when } N \text{ is even}$$

# What Is Variance?

- The term variance refers to a statistical measurement of the spread between numbers in a data set.
- More specifically, variance measures how far each number in the set is from the mean (average), and thus from every other number in the set. Variance is often depicted by this symbol:  $\sigma^2$ .
- The square root of the variance is the standard deviation (SD or  $\sigma$ ), which helps determine the consistency of an investment's returns over a period of time.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/module\\_201\(Introduction%20to%20Statistics/Methods\\_of\\_Variability.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/module_201(Introduction%20to%20Statistics/Methods_of_Variability.ipynb)

## Key Points

- Variance is a measurement of the spread between numbers in a data set.
- In particular, it measures the degree of dispersion of data around the sample's mean.
- Investors use variance to see how much risk an investment carries and whether it will be profitable.
- Variance is also used in finance to compare the relative performance of each asset in a portfolio to achieve the best asset allocation.

# Understanding Variance

## Sample Variance

$$s^2 = \frac{\sum(x - \bar{x})^2}{n - 1}$$

## Sample Standard Deviation

$$s = \sqrt{\frac{\sum(x - \bar{x})^2}{n - 1}}$$

# Probability Distribution

- In probability theory and statistics, a probability distribution is the mathematical function that gives the probabilities of occurrence of different possible outcomes for an experiment.
- It is a mathematical description of a random phenomenon in terms of its sample space and the probabilities of events (subsets of the sample space).

# What Is a Normal Distribution?

- Normal distribution, also known as the Gaussian distribution, is a probability distribution that is symmetric about the mean, showing that data near the mean are more frequent in occurrence than data far from the mean.
- The most commonly used distribution is the normal distribution, which is used frequently in finance, investing, science, and engineering. The normal distribution is fully characterized by its mean and standard deviation, meaning the distribution is not skewed and does not exhibit kurtosis.

# Gamma Function

- The gamma function is defined as an improper definite integral. First, an integral represents the antiderivative of a function and the approximate area between curves based on the infinite summation of the areas of thin vertical rectangles. Integration is the process used to find the value of an integral.

## Key Points :

- The Factorial Function: One of these properties is that the value of the gamma function of  $x$  plus one is equal to  $x$  times the value of the gamma function of  $x$ . This property leads to the definition of the complex factorial function, which is that  $x!$  is equal to the value of the gamma function of  $x + 1$ .
- Euler's Reflection Formula: This formula is also known as the complement formula. It relates a number and its complement to trigonometry, specifically, to the value of sine of  $\pi$  times  $x$ .

# Chi-Square Distribution

- The chi-squared distribution is defined as the distribution of a sum of the squares of  $k$  independent standard normal random variables.
- A standard normal random variable has a mean of 0 and a variance of 1. The parameter  $k$  is known as the degrees of freedom of the chi-squared distribution.

# What Is Discrete Distribution?

- A discrete probability distribution counts occurrences that have countable or finite outcomes.
- Discrete distributions contrast with continuous distributions, where outcomes can fall anywhere on a continuum.
- Common examples of discrete distribution include the binomial, Poisson, and Bernoulli distributions.
- These distributions often involve statistical analyses of "counts" or "how many times" an event occurs.

# Types of Discrete Distribution

- The most common discrete distributions used by statisticians or analysts include the binomial, Poisson, Bernoulli, and multinomial distributions.
- Others include the negative binomial, geometric, and hypergeometric distributions

# Binomial Distribution

- The binomial distribution, for example, evaluates the probability of an event occurring several times over a given number of trials and given the event's probability in each trial.
- It may be generated, for example, by keeping track of how many free throws a basketball player makes in a game, where 1 = a basket and 0 = a miss

# Uniform Distribution

- Uniform distributions are probability distributions with equally likely outcomes.
- In a discrete uniform distribution, outcomes are discrete and have the same probability.
- In a continuous uniform distribution, outcomes are continuous and infinite.
- In a normal distribution, data around the mean occur more frequently.
- The frequency of occurrence decreases the farther you are from the mean in a normal distribution.

# Bernoulli Distribution

- The Bernoulli distribution is the most basic discrete distribution. A variable that follows the distribution can take one of two possible values, 1 (usually called a success) or 0 (failure), where the probability of success is  $p$ ,  $0 < p < 1$ .
- An example of a Bernoulli random variable (that is a variable that follows the Bernoulli distribution) is the outcome of a coin toss, where the outcome is either a head (success) or a tail (failure) and the probability of a head is a number between 0 and 1.

# Geometric Distribution

- The geometric probability distribution is a special type of discrete probability distribution.
- A geometric random variable is a random variable that counts the number of dichotomous (Bernoulli) trials before a success occurs. The corresponding probability distribution is a geometric probability distribution.

# Poisson Distribution

- In statistics, a Poisson distribution is a discrete probability distribution that tells how many times an event is likely to occur over a specified period.
- It is a count distribution, the parameter of which is lambda ( $\lambda$ ); the mean number of events in the specific interval.

# Examples :

[https://github.com/TopsCode/Data Analysis 2024/t  
ree/main/module%201\(Introduction%20to%20Stati  
stics\)](https://github.com/TopsCode/Data_Analysis_2024/tree/main/module%201(Introduction%20to%20Statistics))

# **Module 2 DA - Introduction to Microsoft MS Excel**

[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/  
main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data)

# Intro to MS Excel

- **Excel Microsoft**

Excel is the first spreadsheet application in the Microsoft Office Suite that allowed the user to define the appearance of spreadsheets.

- **Spreadsheet –**

A table of values arranged in rows and columns where each value can have a predefined relationship to the other values that sit in their respective cell.

# Excel Advantages:

- It has intelligent cell recomputation.
- Excel allows large numbers of calculations to be carried out simultaneously.
- It also has extensive graphing capabilities, and enables users to perform mail merge.

# Workbook

- **Workbook - This is** also called a spreadsheet and a unique file created by Excel.
- Automatically shows in the workspace when you open Microsoft Excel .
- Each workbook contains three worksheets (labeled Sheet1, Sheet2, and Sheet3).
- A workbook must contain at least one worksheet.

# Worksheet -

- A grid of cells, consisting of 65,536 rows by 256 columns.
- Sheet Tabs –
- separate a workbook into specific worksheets.
- Navigation buttons –
- allow you to move to another worksheet in an Excel workbook and is used to display the first, previous, next or last worksheets in the workbook. MS Excel

# Excel Advantages:

- It has intelligent cell recomputation.
- Excel allows large numbers of calculations to be carried out simultaneously.
- It also has extensive graphing capabilities, and enables users to perform mail merge.

- **Name box** - shows the address of the current selection or active cell.
- **Formula bar** - displays information being entered as you type-in the current or active cell.
- **Cells** - little boxes that are formed from the intersection of columns and rows. The contents of a cell can also be edited in the Formula bar. **Cell address** - name designated to each cell which is comprised of two parts:
  - a) the column letter; and
  - b) the row number.
- **Active cell** - refers to the cell that can be acted upon or receives the data or command you give it which reveals a dark border. All other cells reveal a light gray border. MS Excel

# MS Excel

**Different spreadsheet information** can be entered into a cell such as text, numbers or mathematical formulas.

- **Text - any entry that is not a number or formula.** MS Excel
- **Numbers - values used when making calculations.**
- **Formulas - mathematical calculations.**

[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/  
main/Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data)

# MS Excel

- **Entering Data in the Cell:**
- This can be done by clicking on the cell where you want to type the information.
- An insertion point appears in the cell as the data is typed.
- The data being typed appear both in the active cell or in the Formula bar.

# MS Excel

- **Editing Information in the Cell:**
- **Method 1: Direct Cell Editing** This can be done by double-clicking on the cell that contains the information to be changed.
- **Method 2: Formula Bar Editing** This can be done by single clicking on the cell that contains the information and edit it in the formula bar. MS Excel

# MS Excel

- **Creating Simple Formula (MDAS Operation):**
- **Step 1: Click the cell where the formula will be defined.**
- **Step 2: Type the equal sign (=) to let Excel know a formula is being defined.**
- **Step 3: Type the first number to be included in the operation. If it involves numeric value contained in another cell, click the cell in which the number is contained. Using Formula**

# MS Excel

- **Step 4:** Type the mathematical operator/s (\*, /, +, -) to let Excel know that an operation is to be performed.
- **Step 5:** Type the second number to be included in the operation the way it is done in step 2 especially if it involves a number in another cell.
- **Step 6:** Press Enter to complete the formula. Using Formula

[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/  
main/Module%203%20DA%20-  
%20Introduction%20to%20Excel%20/Lecture%20Data](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data)

# Using Formula

$$120/(8-5)*4-2$$



Perform the operation in parentheses  
first:  $8-5=3$

formula becomes

$$120/3*4-2$$



Because the division comes before the  
multiplication, divide  $120/3=40$

formula becomes

$$40*4-2$$



Next, the multiplication takes place  
before the subtraction:  $40*4=160$

formula becomes

$$160-2$$



Finally,  $160-2=158$

The final answer is 158

# Using Formula

## Applying Formulas to other Cells:

Formula are essential element in using Excel especially when your dealing with a range of cells in a given column or row.

Two ways to apply formula to other cells:

1. Copy and Paste Method
2. Fill Formula Method Using Formula

- **Fill Formula Method** - allows you to copy a formula and fill it into many different consecutive cells at the same time. Fill Handle Using Formula

	F	G
7		
8	<i>Expenses (Jan - Mar)</i>	<i>Net Income</i>
9	50,000.00	
10	3,000.00	
11	3,455.00	

A screenshot of a Microsoft Excel spreadsheet illustrating the Fill Handle feature. The spreadsheet has three columns: F and G, and rows numbered 7 through 11. Row 8 contains the labels 'Expenses (Jan - Mar)' and 'Net Income'. Row 9 contains the value '50,000.00'. A red circle highlights the 'Fill Handle' located at the bottom-right corner of the cell containing '50,000.00'. An arrow points from the text 'Fill Handle' to this red circle.

# VLOOKUP

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Mo  
dule%203%20DA%20-%  
%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%2  
0Lecture-2.docx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Lecture-2.docx)

VLOOKUP LOOKS FOR  
A VALUE IN THE  
LEFTMOST COLUMN OF  
A TABLE

THEN RETURNS A  
VALUE IN THE SAME  
ROW

FROM A COLUMN YOU  
SPECIFY

**IN OTHER WORDS**

VLOOKUP FUNCTION IS  
USED TO SEARCH A  
VALUE IN A TABLE

IF FOUND, RETURN  
THE CORRESPONDING  
ROW VALUE OF THAT  
TABLE

FOR THE SPECIFIED  
COLUMN

# Syntax :

```
=VLOOKUP(LOOKUP_VALUE, TABLE_ARRAY,  
COL_INDEX_NUM, [RANGE_LOOKUP])
```

LOOKUP\_VALUE

VALUE YOU WANT TO LOOK UP

TABLE\_ARRAY

THE RANGE WHERE THE LOOKUP VALUE IS LOCATED

COL\_INDEX\_NUM

COLUMN NUMBER IN THE RANGE THAT CONTAINS THE RETURN VALUE

RANGE\_LOOKUP

OPTIONALLY, YOU CAN SPECIFY 'TRUE' FOR AN APPROPRIATE MATCH OR  
'FALSE' FOR AN EXACT MATCH OF THE RETURN VALUE

# Types :

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20D%A%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Lecture-2.docx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20D%A%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Lecture-2.docx)

## RANGE LOOKUP

USED WHEN YOU WANT TO SEARCH FOR RANGES, IT WILL LOOK FOR NEAREST MINIMUM VALUE FROM THE FIRST COLUMN OF THE TABLE

IF TABLE CONSIST OF 40, 50, 60, 70 AND YOU SEARCH FOR 55, THEN IT WILL DESTINED TO 50 AS IT IS THE NEAREST MINIMUM VALUE

IT IS AN APPROPRIATE MATCH. INDICATED AS TRUE OR 1

## EXACT LOOKUP

USED WHEN YOU WANT TO SEARCH FOR EXACT VALUE

IF TABLE CONSIST OF 40, 50, 60, 70 AND YOU SEARCH FOR 55, THEN IT WILL SHOW NOTHING I.E. NA.

IT IS AN EXACT MATCH. INDICATED AS FALSE OR 0

# HLOOKUP:

HLOOKUP LOOKS FOR  
A VALUE IN THE TOP  
ROW OF A TABLE

THEN RETURNS A  
VALUE IN THE SAME  
COLUMN

FROM A ROW YOU  
SPECIFY

**IN OTHER WORDS**

HLOOKUP FUNCTION IS  
USED TO SEARCH A  
VALUE IN A TABLE

IF FOUND, RETURN  
THE CORRESPONDING  
COLUMN VALUE OF  
THAT TABLE

FOR THE SPECIFIED  
ROW

# Syntax:

```
=HLOOKUP(LOOKUP_VALUE, TABLE_ARRAY,  
ROW_INDEX_NUM, [RANGE_LOOKUP])
```

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx)

# Index and Match :

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Lecture-5.docx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Lecture-5.docx)

INDEX MATCH IS A COMBINATION OF TWO FUNCTIONS

INDEX

MATCH

RETURNS A VALUE OR REFERENCE OF THE CELL AT THE INTERSECTION OF A PARTICULAR ROW AND COLUMN, IN A GIVEN RANGE.

RETURNS THE RELATIVE POSITION OF AN ITEM IN AN ARRAY THAT MATCHES A SPECIFIED VALUE IN A SPECIFIED ORDER.

WHEN WE USE INDEX AND MATCH FUNCTION TOGETHER, THE MATCH FUNCTION FINDS THE LOOKUP\_VALUE'S ROW INDEX OR COLUMN INDEX NUMBER AND THEN PASSES THE INFORMATION TO THE INDEX FUNCTION, WHICH RETURNS THE INFORMATION WHICH WE ACTUALLY WANT.

# Syntax

SYNTAX	
<code>=INDEX(ARRAY, ROW_NUM, [COLUMN_NUM])</code>	

INDEX

ARRAY : A RANGE OF CELLS
ROW_NUM : ROW IN AN ARRAY FROM WHICH TO RETURN A VALUE
COLUMN_NUM : COLUMN IN AN ARRAY FROM WHICH TO RETURN A VALUE

A	B
1	
2 PRODUCT	QUNATITY
3 SHIRTS	54
4 TIES	47
5 COATS	18
6 SHORTS	68
7 T-SHIRTS	74
8 SHOES	38
9 SOCKS	46
10 JEANS	86
11 WATCHES	55
12 TROUSER	42

THIS RESULTS VALUE AT THE INTERSECTION OF ROW 3 AND COLUMN 2 IN THE ARRAY A3:B12

THIS RESULTS VALUE AT THE INTERSECTION OF ROW 5 AND COLUMN 1 IN THE ARRAY A3:B12

THIS RESULTS VALUE AT THE INTERSECTION OF ROW 9 AND COLUMN 1 IN THE ARRAY A3:B12

A	B
13	
14 FORMULA	=INDEX(A3:B12,3,2)
15 RESULT	18

A	B
13	
14 FORMULA	=INDEX(A3:B12,5,1)
15 RESULT	T-SHIRTS

A	B
13	
14 FORMULA	=INDEX(A3:B12,9,1)
15 RESULT	WATCHES

# Syntax

## SYNTAX

=MATCH(LOOKUP\_VALUE, LOOKUP\_ARRAY, [MATCH\_TYPE])

## MATCH

LOOKUP\_VALUE : VALUE YOU USE TO FIND A VALUE YOU WANT IN A TABLE

LOOKUP\_ARRAY : A RANGE OF CELLS IN WHICH TO LOOK FOR VALUES

MATCH\_TYPE : THE NUMBER 0 FOR EXACT MATCH & 1,-1 FOR APPROX MATCH

A
1
2 PRODUCT
3 SHIRTS
4 TIES
5 COATS
6 SHORTS
7 T-SHIRTS
8 SHOES
9 SOCKS
10 JEANS
11 WATCHES
12 TROUSER

THIS LOOKUP FOR "COATS" IN THE RANGE A3:A12 AND  
RETURN ITS POSITION

A	B
13	
14	FORMULA =MATCH("COATS",A3:A12,0)
15	RESULT 3

THIS LOOKUP FOR "JEANS" IN THE RANGE A3:A12 AND  
RETURN ITS POSITION

A	B
13	
14	FORMULA =MATCH("JEANS",A3:A12,0)
15	RESULT 8

THIS LOOKUP FOR "SHOES" IN THE RANGE A3:A12 AND  
RETURN ITS POSITION

A	B
13	
14	FORMULA =MATCH("SHOES",A3:A12,0)
15	RESULT 6

# Index Match vs Lookups

1

INDEX MATCH RETURNS A REFERENCE RATHER THAN A VALUE, WHICH ALLOW US TO DO MORE THINGS IN ONE FORMULA

2

INDEX MATCH CAN LOOKUP EITHER VERTICAL OR HORIZONTAL DATA, WITH NO NEED TO CHANGE FUNCTIONS

5

THERE CAN BE TWO REASONS YOU USE VLOOKUP/HLOOKUP INSTEAD OF INDEX MATCH. EITHER YOU DON'T KNOW HOW TO USE INDEX MATCH OR YOU KNOW NOTHING ABOUT INDEX MATCH

3

INDEX MATCH CAN RETURN APPROXIMATE MATCHES FROM DATA SORTED FROM LARGEST TO SMALLEST. WE CAN'T DO SAME WITH LOOKUP FUNCTION

4

AT IT WORST, INDEX MATCH IS SLIGHTLY FASTER THAN VLOOKUP AND AT ITS BEST, IT IS MANY TIMES FASTER

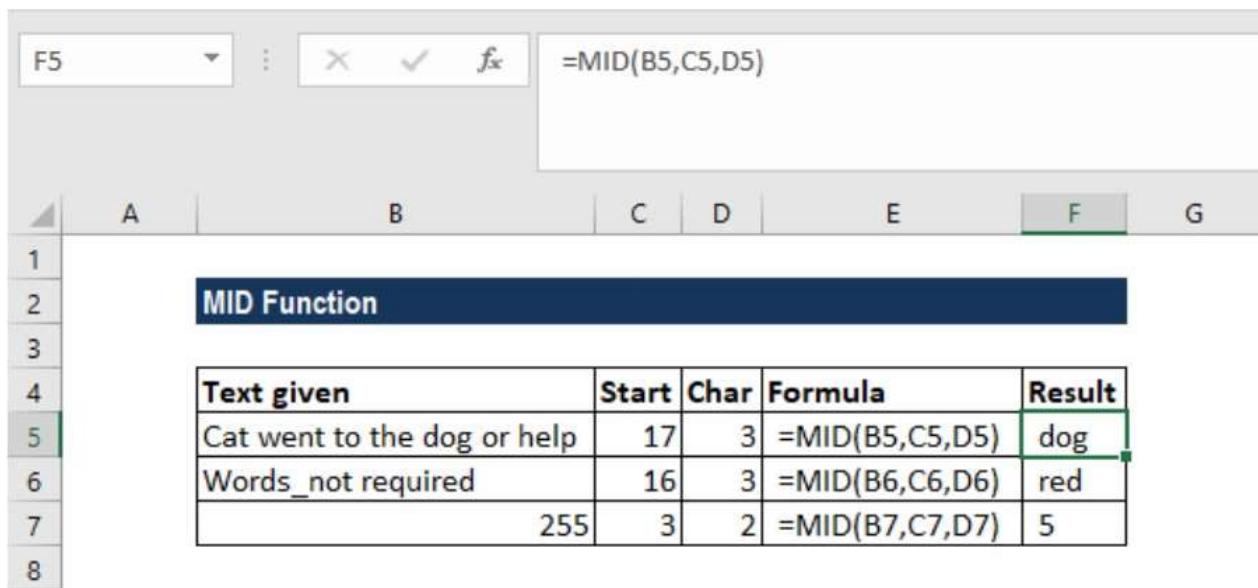
[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20-%203.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20-%203.xlsx)

# MID Function

- **=MID(text,start\_num,num\_chars)**
- The MID function uses the following arguments:
  1. **Text (required argument)** – The original text string.
  2. **Start\_num (required argument)** – This is an integer that specifies the position of the first character that you want to be returned.
  3. **Num\_chars (required argument)** – Specifies the number of characters, starting with start\_num, to be returned from the start of the given text. It is the number of characters to be extracted.

[https://github.com/TopsCode/Data\\_Analys s\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lectur e%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx](https://github.com/TopsCode/Data_Analys s_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lectur e%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx)

# MID Function



The screenshot shows a Microsoft Excel interface. The formula bar at the top contains the formula `=MID(B5,C5,D5)`. The spreadsheet area below has columns A through G and rows 1 through 8. Row 2 contains the text "MID Function". Row 5 contains the formula `=MID(B5,C5,D5)` and its result "dog". Row 6 contains the formula `=MID(B6,C6,D6)` and its result "red". Row 7 contains the formula `=MID(B7,C7,D7)` and its result "5".

	A	B	C	D	E	F	G
1							
2		MID Function					
3							
4	Text given	Start	Char	Formula	Result		
5	Cat went to the dog or help	17	3	=MID(B5,C5,D5)	dog		
6	Words _not required	16	3	=MID(B6,C6,D6)	red		
7		255	3	=MID(B7,C7,D7)	5		
8							

# OFFSET Function

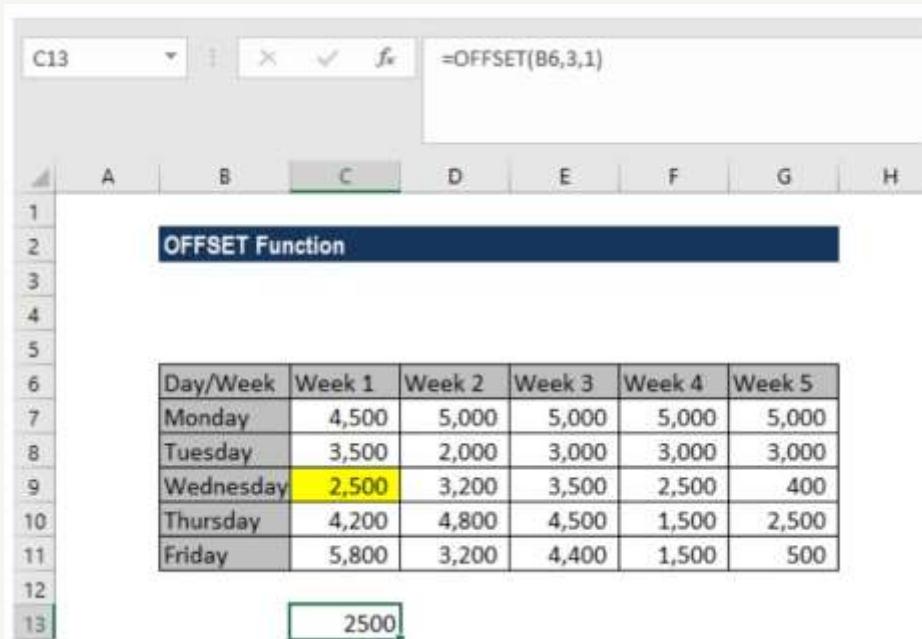
= OFFSET(reference, rows, cols, [height], [width])

The OFFSET function uses the following arguments:

1. Reference (required argument) – This is the cell range that is to be offset. It can be either single cell or multiple cells
2. Rows (required argument) – This is the number of rows from the start (upper left) of the supplied reference, to the start of the returned range.
3. Cols (required argument) – The number of columns from the start (upper left) of the supplied reference, to the start of the returned range.
4. Height (optional argument) – Specifies the height of the returned range. If omitted, the returned range is the same height as the supplied reference argument.
5. Width (optional argument) – This specifies the width of the returned range. If omitted, the returned range is the same width as the supplied reference.

[https://github.com/TopsCode/Data\\_Analys s\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lectur e%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx](https://github.com/TopsCode/Data_Analys s_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lectur e%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx)

# OFFSET FUNCTION



The screenshot shows a Microsoft Excel spreadsheet with the following details:

- Cell C13:** Contains the formula `=OFFSET(B6,3,1)`.
- Table Headers:** Row 6 contains the headers "Day/Week", "Week 1", "Week 2", "Week 3", "Week 4", and "Week 5".
- Data Rows:** Rows 7 through 11 contain data for Monday, Tuesday, Wednesday, Thursday, and Friday respectively, across the six weeks.
- Wednesday's Value:** The value "2,500" is highlighted in yellow in the "Week 1" column of Wednesday's row.
- Result in C13:** The result of the formula, "2500", is displayed in the cell C13.

# What is the Excel TEXT Function

- The Excel TEXT Function is used to convert numbers to text within a spreadsheet.
- Essentially, the function will convert a numeric value into a text string.
- TEXT is available in all versions of Excel.

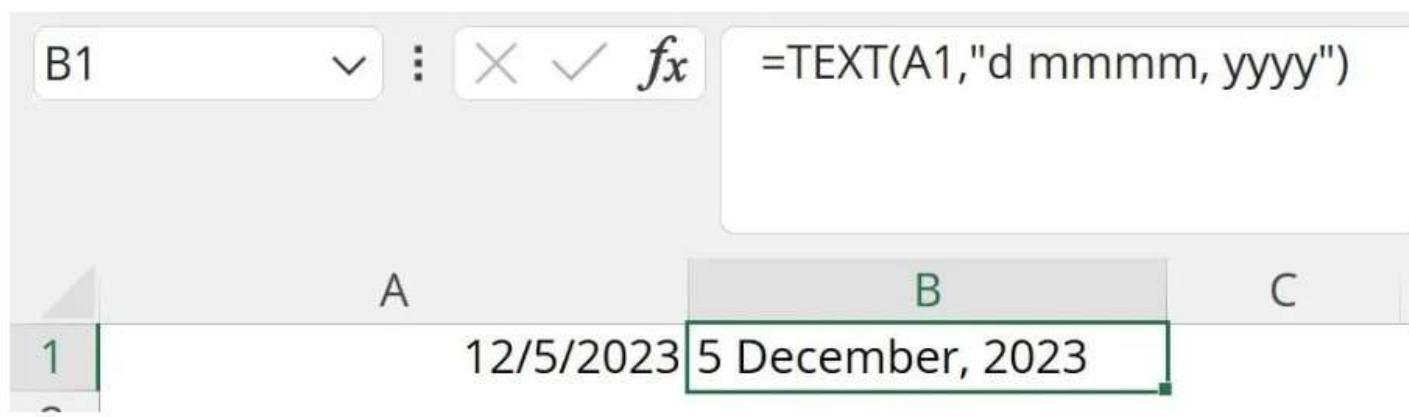
[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx)

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx)

# Examples

## 1. Basic example - Excel Text Function

With the following data, I need to convert the data to “d mmmm, yyyy” format. When we insert the text function, the result would look as follows:



The screenshot shows the Microsoft Excel interface. In the formula bar, the cell reference "B1" is selected, along with the formula `=TEXT(A1,"d mmmm, yyyy")`. Below the formula bar, the worksheet area shows column headers A, B, and C. Cell A1 contains the number "1". Cell B1 contains the date "12/5/2023", which has been converted to the text "5 December, 2023" using the specified formula. The text in cell B1 is highlighted with a green border.

# LEN Function

=LEN(text)

The LEN function uses only one argument:

1. **Text (required argument)** – This is the text for which we wish to calculate the length. We can provide the text argument for the function:
  - Directly
  - As a string returned from another formula
  - As a reference to a cell containing a string

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx)

# TRIM Function

- **=TRIM(text)**
- **Text (required argument) – This is the text from which we wish to remove spaces.**

[https://github.com/TopsCode/Data\\_Analyses\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20-%201.xlsx](https://github.com/TopsCode/Data_Analyses_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20-%201.xlsx)

# UPPER Function

- **=UPPER(Text)**
- **Text (required argument) – This is the text that we want to convert to uppercase. Text can be a text string or a reference to a cell.**

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20-%201.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20-%201.xlsx)

# LOWER Function

- **=LOWER(text)**
- This function uses one argument, that is, text. It is a required argument here. It is the text that we wish to convert to lowercase. The function will not change the characters in the text string that are not letters. So, for example, numbers and punctuations will remain unaffected when we use LOWER.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx)

# Find and Replace in Excel

- Find and Replace in Excel allows you to quickly search all cells and formulas in a spreadsheet for all instances that match your search criteria. This guide will cover how to search in Excel and use find and replace in Excel.

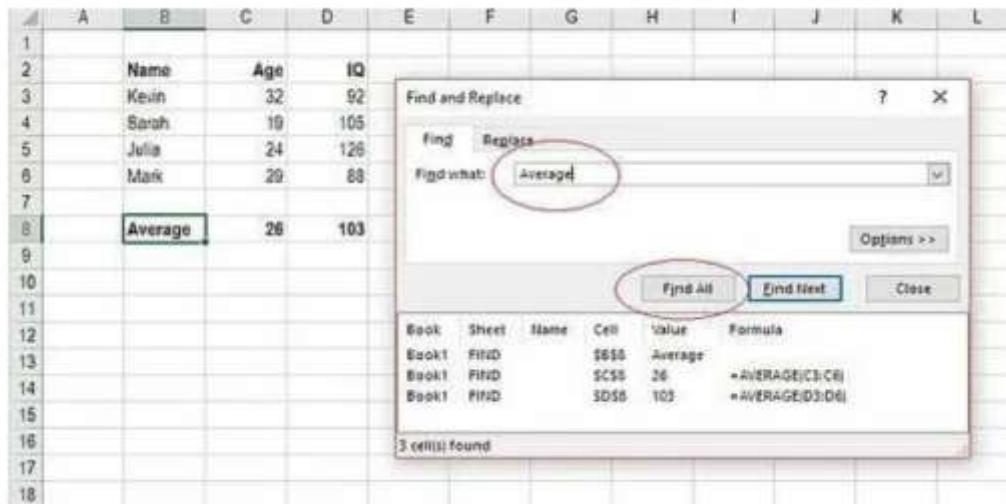
[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data/Excel%20Practice%20Sheet%20%20-%201.xlsx)

## Part 1: Find a single data point

1. Press Ctrl + F
  2. Type "Sarah"
  3. Click Find Next

## Part 2: Search in Excel all instances of something

1. Press Ctrl + F
2. Type "Average"
3. Click Find All



Book	Sheet	Name	Cell	Value	Formula
Book1	FIND		\$E\$5	Average	
Book1	FIND		\$C\$5	26	=AVERAGE(C3:C6)
Book1	FIND		\$D\$5	103	=AVERAGE(D3:D6)

3 cell(s) found.

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		Name	Age	IQ								
3		Kevin	32	92								
4		Sarah	19	105								
5		Julia	24	126								
6		Mark	29	88								
7		<b>Median</b>	26.5	99								
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												

Find and Replace

Find

Find what:

Replace with:

Replace Find All Find Next Close

Book:	Sheet:	Name:	Cell:	Value:	Formula:
Book1	FIND		\$B\$8	Median	=MEDIAN(C3:C6)
Book1	FIND		\$C\$8	26.5	=MEDIAN(C3:C6)
Book1	FIND		\$D\$8	99	=MEDIAN(D3:D6)

3 cell(s) found

Microsoft Excel

All done. We made 3 replacements.

# Pivot Table

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data/Excel%20Practice%20Sheet%20-%202.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data/Excel%20Practice%20Sheet%20-%202.xlsx)

# What is Pivot Table

- Many Excel users are not familiar with, or are intimidated by Pivot Tables, one of the most powerful features in Excel. This presentation describes Pivot Tables and Few Features of Pivot Table.
- Creating Pivot Table A pivot table is a great reporting tool that sorts and sums independent of the original data layout in the spreadsheet. If you never used one, this below example will most interesting for you.
- First, set up / create some data, in a specific range in Excel, like below slide,

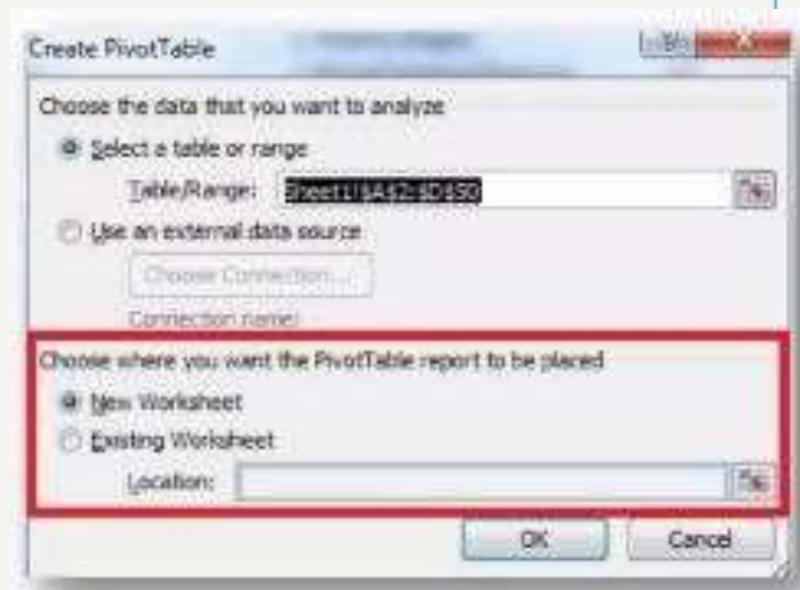
# Selecting Data Range for Pivot Table

List of Weekly Payments to Building Workers.			
	Name of Worker	Week	Nature of Payment
1	X		1. Washing Allowance
2	X		1. Conveyance Expenses
3	X		1. Weekly Wages
4	X		1. Miscellaneous Expense:
5	X		1. Washing Allowance
6	X		1. Conveyance Expenses
7	Y		1. Weekly Wages
8	Y		1. Miscellaneous Expense:
9	Y		1. Washing Allowance
10	Y		1. Conveyance Expenses
11	Z		1. Weekly Wages
12	Z		1. Miscellaneous Expense:
13	Z		1. Washing Allowance
14	Z		1. Conveyance Expenses
15	Z		1. Weekly Wages
16	Z		1. Miscellaneous Expense:
17	X		2. Washing Allowance
18	X		2. Conveyance Expenses
19	X		2. Weekly Wages
20	X		2. Miscellaneous Expense:
21	Y		2. Washing Allowance
22	Y		2. Conveyance Expenses
23	Y		2. Weekly Wages
24	Y		2. Miscellaneous Expense:
25	Z		2. Washing Allowance
26	Z		2. Conveyance Expenses
27	Z		2. Weekly Wages
28	X		3. Miscellaneous Expense:
29	X		3. Washing Allowance
30	X		3. Conveyance Expenses
			3. Weekly Wages
			3. Miscellaneous Expense:

- As given in example I have created data of 3 Workers x,y,z and their, weekly payments in Various Segments. Selected a Range of (A1:D50)

# Pivot Table Range

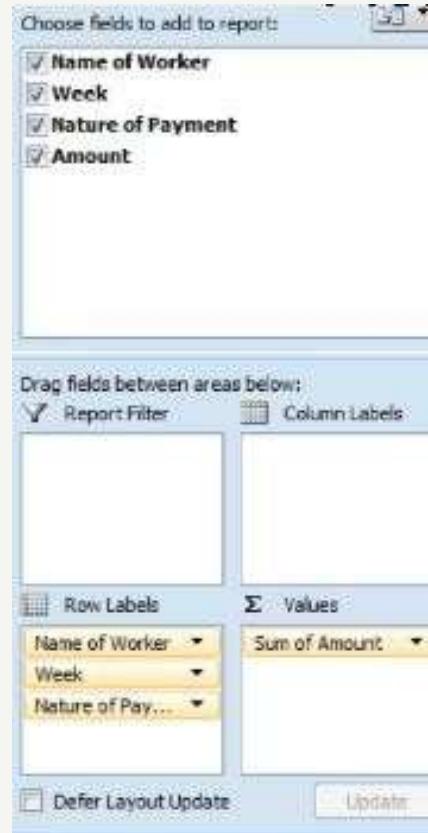
- Here We need to understand the data range. Excel suggests the table as shown in above Slide .
- If you expect to add data in the future, set the data range to include as many rows as you think you will ever need.
- Rather than A1:D50, you may want to specify \$A\$1:\$D\$500. One more suggestion is , as shown in Graphic you can define the Destination of Pivot Table as New Sheet or Existing Sheet



# Pivot Table Field

List Report Filter Column Labels Row  
Labels  $\sum$  Values

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20-%202.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Practice%20Sheet%20-%202.xlsx)



# **Pivot Table Filed List Definitions**

- **Report Filter** : Use a report filter to conveniently display a subset of data in a PivotTable report or Pivot Chart Report. A report filter helps to manage the display of large amounts of data, and to focus on a subset of data in the report, such as a product line, a time span, a Geographic region Column
- **Labels** : A field that is assigned a column orientation in a PivotTable report.
- **Row Labels** : A field that is assigned a row orientation in a Pivot Table report.

# Introduction to Power Query Editor

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Lecture-6.docx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Lecture-6.docx)

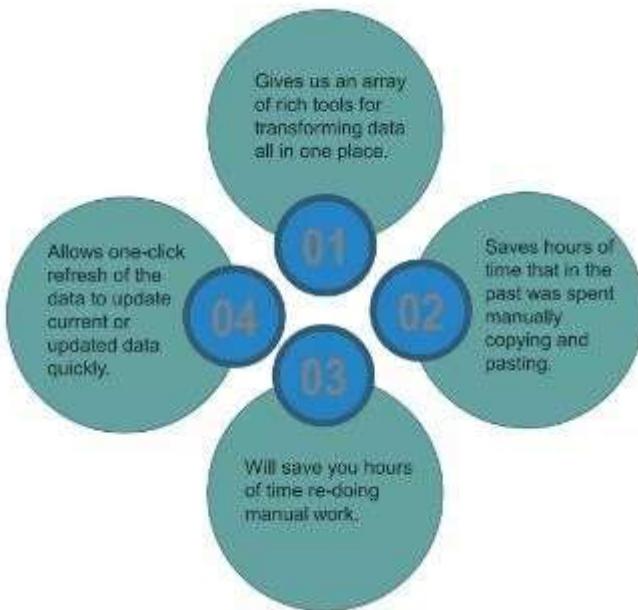
# Introduction to Power Query

- Power Query is an Excel add-in that you can use for ETL. That means, you can **extract data** from different sources, **transform** it, and **then load** it to the worksheet.

Power Query is a **business intelligence tool available in Excel** that allows you to import data from many different sources and then clean, transform and reshape your data as needed. It allows you to set up a query once and then reuse it with a simple refresh.



# Purpose and Advantages



# Getting to know Power Query & Editor

1

Find and connect  
data across a wide  
variety of sources.

2

Processing large  
amount of data  
(where Excel is  
limited to one million  
rows)

3

Merge and shape  
data sources to  
match your data  
analysis  
requirements or  
prepare it for further  
analysis and  
modelling by tools  
such as Power Pivot  
and Power View.

4

Manipulating,  
cleaning and  
combining several  
tables, files or even  
folders !

5

Create custom views  
over data.

# Why is Power Query a great alternative to Excel VBA?

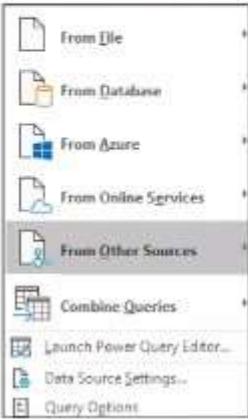
- 01** Process your tables simply by clicking on buttons. No coding skills needed !
- 02** Visualize your operations step by step without running a macro.
- 03** Easily edit the steps order of different steps through dragging and dropping!  
Also, you can add and delete unnecessary steps by a simple click.

# Loading data with Power Query

On the Data tab, in the Get & Transform Data group, click Get Data.



Click on the source



[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Lecture-6.docx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Excel%20Lecture-6.docx)

# What are Excel Interactive Dashboards

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Dashboard%20Created.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Dashboard%20Created.xlsx)

Abhishek Wavhal  
DA / DS Trainer Tops Technologies

# **Definition What are they**

**Excel Dashboards are interactive data visualization tools which connect perfectly with all Excel tools and give you a graphical overview of your whole dataset**



## What can be found in an Excel Dashboard?

- Several Charts & Graphs
- Slicers or Timelines
- Pivot Tables

**Remember to give an overview of your dataset and present mostly summarized information**



## Which tools can with an Excel Dashboard?

- Power Query
- Hyperlinks
- Custom & Conditional Formatting

Hyperlinks to other sheets are the best way to move around your Excel Dashboard



## Charts & Graphs

Create an overview of your Dataset using a combination of  
these Data Visualization tools



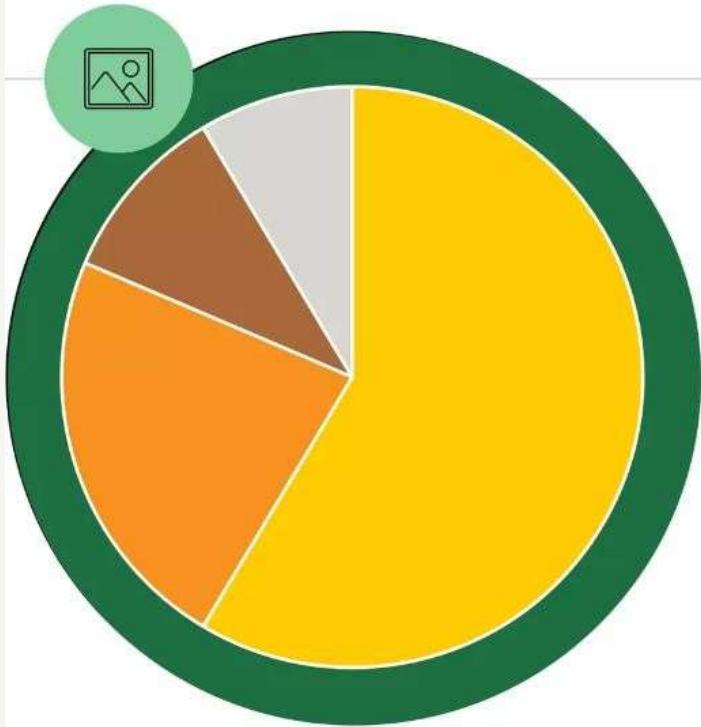
## What are they?

### Charts

"A drawing that shows information in a simple way, often using lines and curves to show amounts"

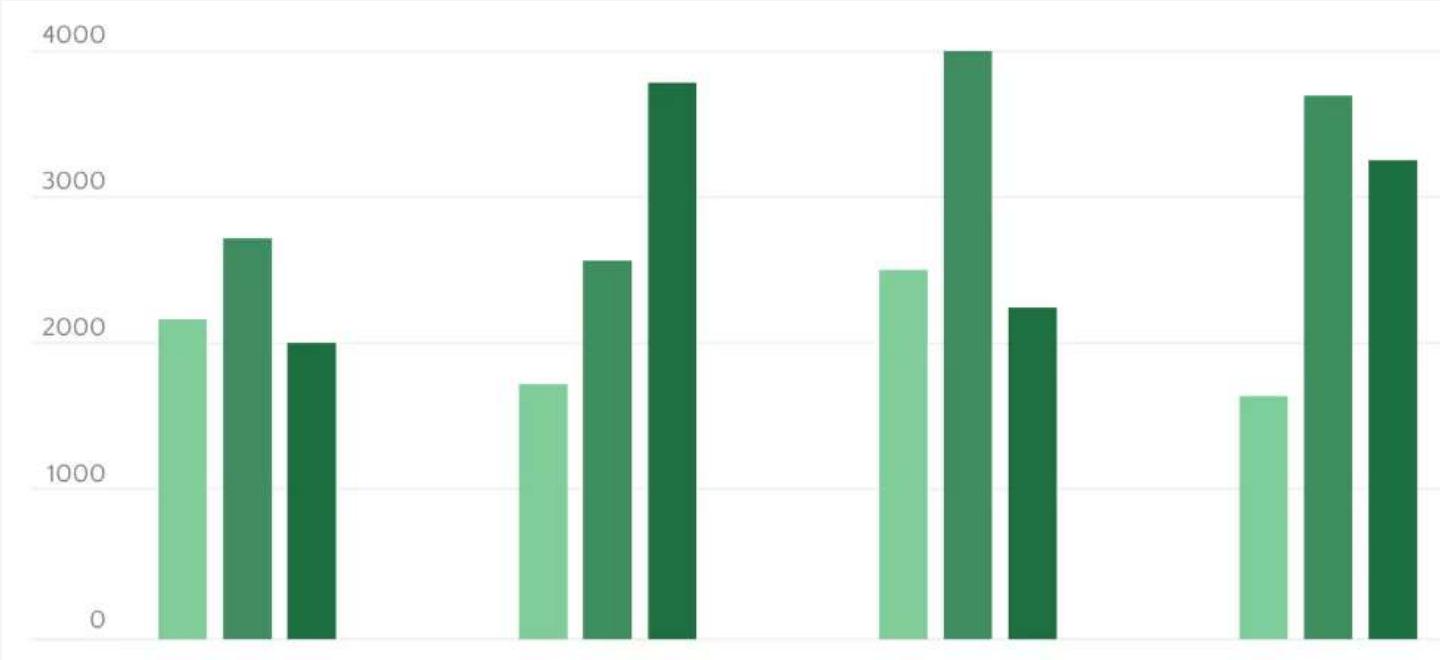
### Graphs

"A picture that shows how two sets of information or variables (...) are related, usually by lines or curves"



Charts let you present your Data Set in a modern and easy to understand way

Add Slicers to make your Excel Dashboard interactive



**Graphs let you present forecasts & tendencies**



You can create Maps

# Excel Interactive Dashboards

## Intuitive

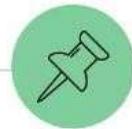
The main objective of an Excel Dashboard is that the Data has to be presented in an user friendly interface.

## Overview

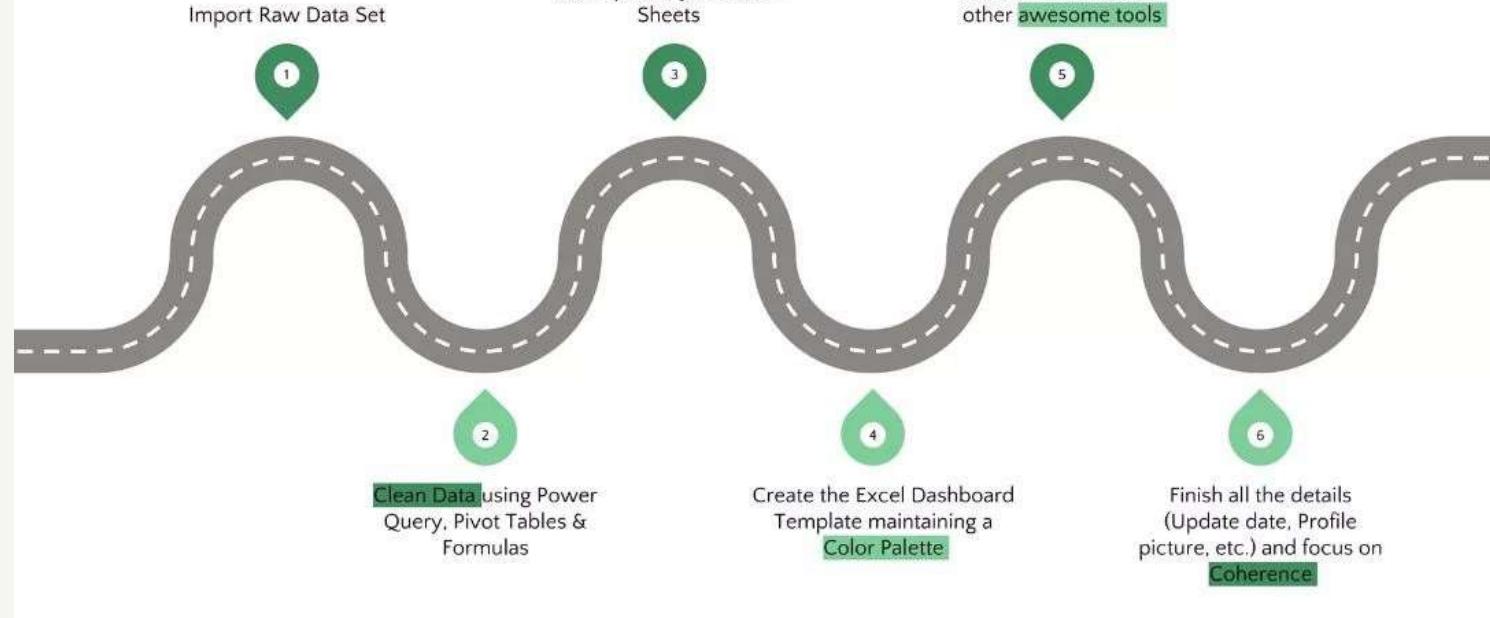
Remember that each piece of the Excel Dashboard has to be synchronized with the overall ecosystem.

## Conclusions

Venture into drawing conclusions and highlighting the most relevant information.



# Roadmap





[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Dashboard%20Created.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/Dashboard%20Created.xlsx)

# Samples :



[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data/Dashboard%20Created%202.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data/Dashboard%20Created%202.xlsx)

# Samples :



[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data/Dashboard%20Created%202.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data/Dashboard%20Created%202.xlsx)

# Samples :



# Introduction to Microsoft MS Excel – Macros

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/  
Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data/Lec%20  
Macros%201.1%20\(2\).xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20Lecture%20Data/Lec%20Macros%201.1%20(2).xlsx)

Abhishek Wavhal  
DA / DS Trainer Tops Technologies

# Content :

Introduction to Macros

Introduction to visual basic application

Recording a Macro

Looking at the Code of Recorded Macro



# Why Macros :

If you have tasks in Microsoft Excel that you do repeatedly, you can record a macro to automate those tasks.

# Why Macros :

- Excel is itself a tool which helps users to work with worksheets & workbooks with ease.
- This tool is one of the ultimate timesaver, but there are numerous events when you have to do repetitive tasks which are of a very little value but consume your precious time; yet engage you in a less productive activity, which Excel may do automatically.
- Microsoft Excel macro is the feature which eliminates or at-least minimize these events which take your precious time in repetitive tasks, and you will be able to capitalize this

# What is Macros :

A macro is an action or a set of actions that you can run as many times as you want.

When you create a macro, you are recording your mouse clicks and keystrokes.

After you create a macro, you can edit it to make minor changes to the way it works.



## Macros :

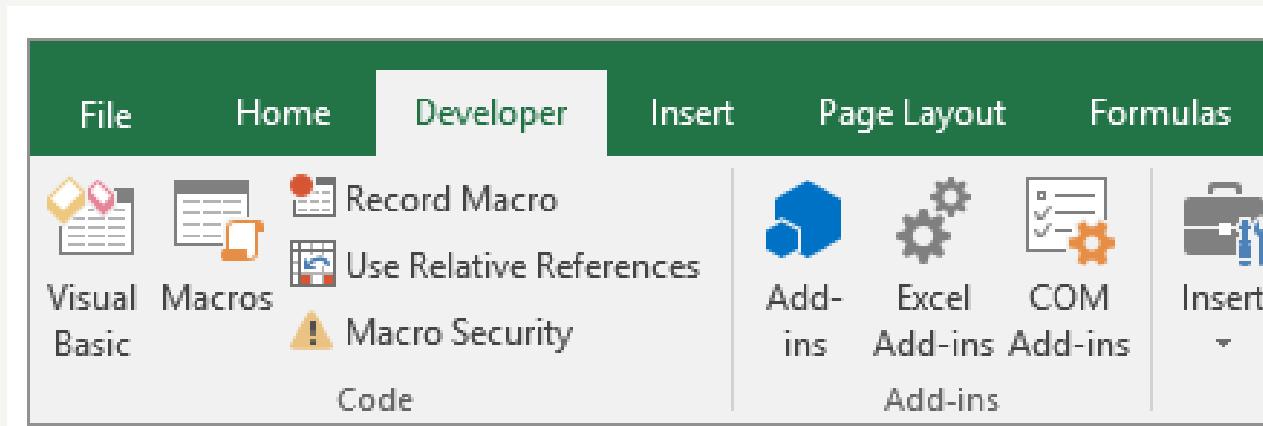
- Suppose that every month, you create a report for your accounting manager.
- You want to format the names of the customers with overdue accounts in red, and also apply bold formatting.
- You can create and then run a macro that quickly applies these formatting changes to the cells you select.



# How :

**Step 1 : Before you record a macro**

**Macros and VBA tools can be found on the Developer tab, which is hidden by default, so the first step is to enable it.**



## How :

Step 2 : In the Code group on the Developer tab, click Record Macro.

Step 3 : Optionally, enter a name for the macro in the Macro name box, enter a shortcut key in the Shortcut key box, and a description in the Description box, and then click OK to start recording.



## How :

**Step 4 : Perform the actions you want to automate, such as entering boilerplate text or filling down a column of data.**

**Step 5 : On the Developer tab, click Stop Recording.**



## Exercise : Create A macro that will convert this CSV file into this fine table

	A	B	C
1.	Aaradhyा, Gupta, 50000, 30		
2.	Advik, Patil, 60000, 35		
3.	Anika, Sharma, 75000, 40		
4.	Aryan, Desai, 55000, 28		
5.	Dia, Patel, 80000, 45		
6.	Ishan, Singh, 65000, 33		
7.	Kavya, Chatterjee, 70000, 38		
8.	Mihir, Mishra, 60000, 32		
9.	Myra, Chopra, 85000, 42		
10.	Navya, Rao, 70000, 36		
11.	Parth, Shah, 60000, 31		
12.	Prisha, Verma, 55000, 29		
13.	Reyansh, Kumar, 90000, 48		
14.	Saisha, Reddy, 65000, 34		
15.	Shaurya, Gupta, 75000, 39		
16.	Shreya, Sharma, 70000, 37		
17.	Viihaan, Khan, 80000, 41		
18.	Zara, Jha, 60000, 33		
19.	Aahana, Banerjee, 70000, 38		
20.	Advalt, Chopra, 65000, 35		
21.			

	A	B	C	D	E	F
	Sr. No.	First Name	Last Name	Salary	Age	
1.						
2.	1	Aaradhyा	Gupta	50000	30	
3.	2	Advik	Patil	60000	35	
4.	3	Anika	Sharma	75000	40	
5.	4	Aryan	Desai	55000	28	
6.	5	Dia	Patel	80000	45	
7.	6	Ishan	Singh	65000	33	
8.	7	Kavya	Chatterjee	70000	38	
9.	8	Mihir	Mishra	60000	32	
10.	9	Myra	Chopra	85000	42	
11.	10	Navya	Rao	70000	36	
12.	11	Parth	Shah	60000	31	
13.	12	Prisha	Verma	55000	29	
14.	13	Reyansh	Kumar	90000	48	
15.	14	Saisha	Reddy	65000	34	
16.	15	Shaurya	Gupta	75000	39	
17.	16	Shreya	Sharma	70000	37	
18.	17	Viihaan	Khan	80000	41	
19.	18	Zara	Jha	60000	33	
20.	19	Aahana	Banerjee	70000	38	
21.	20	Advalt	Chopra	65000	35	
22.						
23.						

# **Introduction to Visual Basic for Application :**

[https://github.com/TopsCode/Data\\_Analysis\\_2024/b  
lob/main/Module%203%20DA%20-  
%20Introduction%20to%20Excel%20/Lecture%20Da  
ta/VBA%201.1.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%203%20DA%20-%20Introduction%20to%20Excel%20/Lecture%20Data/VBA%201.1.xlsx)

## What is Visual Basic?

- Visual Basic is a language which is inherited from a very popular language BASIC. BASIC stands for Beginners All-purpose Symbolic Instruction Code. Visual Basic is called visual as you can do most of the program by click and go i.e. visually. It's an event driven & object based language.



# Introduction to Visual Basic for Application :

- **Visual Basic for Application** Visual Basic for Application is called VBA as it uses Visual Basic language and is capable of using application specific objects i.e. if we talk about VBA for Excel it can use Cell Object, Range Object, Worksheet Object, Workbook Object etc.

# Exercise :

## Step 1:

Order ID,Name,Product,Quantity,Price,Total
11280,Bill Smith,Volkswagen Golf,15,30000,450000
11281,Kennedi Singh,Toyota Yaris,10,25000,250000
11282,Harley Fritz,Seat Panda,150,28000,4200000
11283,Nyla Novak,Ford Focus,12,30000,360000
11284,Ivan Hines,Vauxhall Corsa,20,30000,600000
11285,Jonah Higgins,Volkswagen Polo,20,26000,520000
11286,Jordan Boone,Kia Sportage,2,30000,60000
11287,Kylee Townsend,Ford Fiesta,5,35000,175000
11288,Nora Rollins,Seat Ibiza,80,30000,2400000
11289,Brendan Walls,Renault Clio,100,26000,2600000
11290,Steven Michael,Honda CR-V,20,35000,700000
11291,Lucia Mckay,GMC Sierra,25,40000,1000000
11292,Josue Roach,Seat Panda,10,28000,280000
11293,Franklin Wright,Ford Focus,10,30000,300000
11294,Denzel Flores,Vauxhall Corsa,5,30000,150000
11295,Bruno Cordova,Volkswagen Polo,180,26000,4680000
11296,Jaylynn Knapp,Kia Sportage,5,30000,150000
11297,Bruce Rich,Ford Fiesta,250,35000,8750000

1= Convert The CSV file into Table
2= With Text To Columns
3= Add Discount Column
4= Do formatting And coloring
5= Create a Function to give Discount of 20 % if Buying Quantiy more than 20

# Exercise :

## **Step 2: Final Function should be discount of 20 %**

# Conclusion:

**Macros are used to automate tasks or you can use macros to develop user interfaces**

**Excel Macro use a language called VBA (Visual Basic for Applications), which is both Event Driven and Object Based**  
**The easiest way to look at macro is by using the Excel's record Macro Feature**

**You can even edit Excel's recorded macro by yourself**

# Applied Statistics in Excel

## Module 3

# Applied Statistics

- As the root of data analysis, the study of applied statistics prepares professionals for careers as statisticians, data scientists, data analysts, and more. Applied statistics is a foundation upon which data science has been built.
- Through statistical methods, analysis, and an emphasis on real-world data, applied statisticians seek concrete solutions to tangible problems. Individuals with a strong background in applied statistics may then become data scientists, but the relationship doesn't work inversely—those who study data science exclusively would not necessarily be prepared for careers as applied statisticians.

# Frequency Distribution

- A Frequency Distribution is a summary of how often each value occurs by grouping values together.

<b>Sections</b>	<b>Students</b>
Section A	12
Section B	13
Section C	8
Section D	6
Section E	12
Section F	14
Section G	19
Section H	22
Section I	14
Section K	15
Section L	20
Section M	27

Excel : Frequency Distribution

# Frequency Distribution

- There are multiple ways to calculate frequency distribution (table) with Excel.
  - With **COUNTIFS** Function
  - With **FREQUENCY** Function
- **Frequency Distribution Using COUNTIFS Function**
- **Formula :**
  - `=COUNTIFS($C$4:$C$15,">="&F5,$C$4:$C$15,"<="&G5)`

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/module%203\(Applied\\_Statistics\)/frequency%20distribution.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/module%203(Applied_Statistics)/frequency%20distribution.xlsx)

# Frequency Formula

- **Formula :**
- **=FREQUENCY(data\_array, bins\_array)**
- The FREQUENCY function uses the following arguments:  
**Data arrays** (It is a required argument) – This is an array or reference to a set of values for which you want to count frequencies.  
**Bins array** (It is a required argument) – This is an array of intervals (“bins”) for grouping values.

# Correlation Matrix

- **Correlation** is a measure that describes the strength and direction of a relationship between two variables. It is commonly used in statistics, economics and social sciences for budgets, business plans and the like.
- The method used to study how closely the variables are related is called **correlation analysis**.
- The number of calories you eat and your weight (positive correlation)

# Correlation Matrix

- **Correlation** is a measure that describes the strength and direction of a relationship between two variables. It is commonly used in statistics, economics and social sciences for budgets, business plans and the like.
- The method used to study how closely the variables are related is called **correlation analysis**.
- The number of calories you eat and your weight (positive correlation)

# Correlation Matrix

- Excel CORREL function
- The CORREL function returns the Pearson correlation coefficient for two sets of values. Its syntax is very easy and straightforward:  
**CORREL(array1, array2)**

Where:

**Array1** is the first range of values.

**Array2** is the second range of values.

The two arrays should have equal length.

# Normal Distribution Using Excel

- A normal distribution graph in Excel represents the normal distribution phenomenon of a given data.
- This graph is made after calculating the mean and standard deviation for the data and then calculating the normal deviation over it.
- Although Excel 2019 versions, it is easy to plot the normal distribution graph as it has a built-in function to calculate the normal distribution and standard deviation. The graph is very similar to the bell curve.

# Example

	A	B	C
1	Particulars	Amt	
2	Stock Price	115	
3	Overall Average Stock Price	90	
4	Standard Deviation of the Stock Price	16	
5	Cumulative Distibution	=NORM.DIST(B2,B3,B4,1)	
6			
7			

# Sample Size Calculator In Excel

- Several variables must be considered for this:
  - The size of the target population
  - The margin of error
  - The confidence level

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/module%203\(Applied\\_Statistics\)/Sample%20Size%20Calculator.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/module%203(Applied_Statistics)/Sample%20Size%20Calculator.xlsx)

# Sample Size Calculator In Excel

<i>Determine Sample Size</i>	
Select Confidence Level	95%
Enter Population Size	2000
Enter Sample size	375
Margin of error	4.56%
An estimate of the proportion of people falling into the group in which you are interested in the population	50%
Sample Size	375

# Sample Size Calculator In Excel

## Sample Size Formula

$$n = \frac{t^2 \times p(1-p)}{m^2}$$

m<sup>2</sup>

n : minimum required sample size

t : confidence level at x% level of significance

p : estimate of the proportion of people falling into the group in which you are interested

m : margin of error

# Compute Correlation Matrix

- The Correlation Matrix in Excel is a square and symmetrical table containing the correlation coefficients of the specified variables.
- And the matrix helps analyze the degree of the linear relationship between all the possible variable pairs
- Ensure the given dataset does not have non-numeric values. Otherwise, we will get a warning message and be unable to obtain the Correlation Matrix in Excel.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/module\\_%203\(Applied\\_Statistics\)/Covariance-Matrix-In-Excel.xlsx](https://github.com/TopsCode/Data_Analysis_2024/blob/main/module_%203(Applied_Statistics)/Covariance-Matrix-In-Excel.xlsx)

# Compute partial Correlation Matrix

- Partial correlation, calculates the correlation between two variables, while excluding the effect of a third variable. This makes it possible to find out whether the correlation  $r_{xy}$  between variables  $x$  and  $y$  is produced by the variable  $z$ .
- The partial correlation  $r_{xy,z}$  tells how strongly the variable  $x$  correlates with the variable  $y$ , if the correlation of both variables with the variable  $z$  is calculated out

# Compute partial Correlation Matrix

## Calculate Partial Correlation

For the calculation of the partial correlation, the three correlations between the individual variables are required. The partial correlation then results in

$$r_{xy,z} = \frac{r_{xy} - r_{xz} \cdot r_{yz}}{\sqrt{(1 - r_{xz}^2) \cdot (1 - r_{yz}^2)}}$$

- $r_{xy}$  = Correlation between variable x and y
- $r_{xz}$  = Correlation of the third variable z with the variable x
- $r_{yz}$  = Correlation of the third variable z with the variable y

$r_{xy}$

0.63

$r_{xz}$

0.57

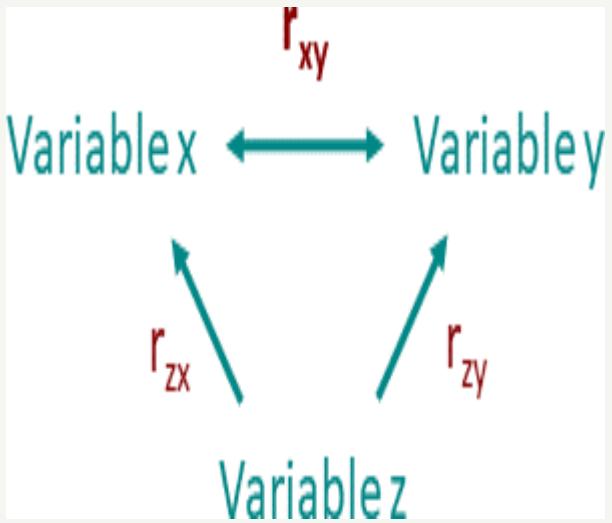
$r_{yz}$

0.88

$r_{xy,z}$

= 0.329

# Compute partial Correlation Matrix



# **MODULE-4**

## **DA-INTRODUCTION TO SQL**

## Core Components:

- **DML (Data Manipulation Language):** Used for managing data within schema objects. Examples include SELECT, INSERT, UPDATE, and DELETE.
- **DDL (Data Definition Language):** Used for defining and managing all schema objects, including creating, altering, and dropping tables and other database structures. Examples include CREATE, ALTER, and DROP.
- **DCL (Data Control Language):** Used to control access to data within the database. Examples include GRANT and REVOKE.
- **TCL (Transaction Control Language):** Used to manage the changes made by DML statements. Examples include COMMIT, ROLLBACK, and SAVEPOINT

# Syntax for querying and Managing the database

:-

## Querying Data:

- One of the most common uses of SQL is to query data from a database.
- This is typically done using the SELECT statement, which can retrieve data based on specific criteria.

## Syntax:-

```
SELECT column1, column2  
FROM table_name  
WHERE condition;
```

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/module%204.1.sql>

## **Updating data:-**

- SQL allows for updating data within the database

## **Syntax:-**

```
UPDATE table_name  
SET column1 = value1, column2 = value2  
WHERE condition;
```

Refer this Example:-

<https://github.com/TopsCode/SQL/blob/master/module%204.1.sql>

## **Inserting Data:**

- SQL is used to add new records to a database.

## **Syntax:-**

```
INSERT INTO table_name  
(column1, column2)  
VALUES (value1, value2);
```

Refer this example:-

<https://github.com/TopsCode/SQL/blob/master/module%204.1.sql>

## **Deleting Data:**

- SQL can also remove records from a database.

## **Syntax:-**

DELETE FROM table\_name WHERE condition;

## **Creating Tables:**

- SQL can create new tables to store data.

## **Syntax:-**

CREATE TABLE table\_name ( column1 datatype, column2 datatype, ..);

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/module%204.1.sql>

## **DATA TYPES IN SQL:-**

- In SQL, data types define the kind of data that can be stored in a column of a table. Different database management systems (DBMS) might have slight variations in their data types, but the basic categories remain consistent. Here are the primary data types used in SQL:

## String datatype:-

1. **CHAR(size)**:- A fixed-length string that can contain letters, numbers, and special characters.
  - The size parameter defines the length of the string.
  - Example: CHAR(10) reserves 10 characters, padding with spaces if necessary.
2. **VARCHAR(size)**: A variable-length string.
  - The size parameter specifies the maximum length of the string.
  - Example: VARCHAR(255) allows up to 255 characters.
3. **TEXT**: Used for large strings of text.
  - Typically does not have a size limit.
4. **BINARY(size) and VARBINARY(size)**: Similar to CHAR and VARCHAR, but for binary data.

Refer this example:-

<https://github.com/TopsCode/SQL/blob/master/MODULE%204.2.1.sql>

# Integer and Numerical Datatype:-

**1. INT or INTEGER:** A standard integer.

Example: 3,4.

**2. SMALLINT:** A smaller range integer.

Example: 150

**3. BIGINT:** A larger range integer.

Example: 983429852

Refer this example:-

<https://github.com/TopsCode/SQL/blob/master/Module%204.2.2.sql>

#### 4. **DECIMAL(p, s) or NUMERIC(p, s)**: Fixed-point numbers.

- p (precision) is the total number of digits.
- s (scale) is the number of digits to the right of the decimal point.
- Example: 10.22 i.e. DECIMAL(10, 2) allows for a number with up to 10 digits, 2 of which can be after the decimal point.

#### 5. **FLOAT**: A floating-point number.

- Less precise than DECIMAL.
- Example: 10.2

Refer this Example:-

<https://github.com/TopsCode/SQL/blob/master/Module%204.2.2.sql>

# Date and Time datatype:-

1. **DATE:** Stores a date value in the format YYYY-MM-DD.  
Example: 2024-02-22
2. **TIME:** Stores a time value in the format HH:MI:SS.  
Example: 22:13:12
3. **DATETIME:** Stores both date and time in the format YYYY-MM-DD HH:MI:SS.  
Example: 2024-02-22 22:13:12
4. **TIMESTAMP:** Similar to DATETIME, but often used to track changes in records.  
Example: 2024-02-22 22:13:12
5. **YEAR:** Stores a year in a two-digit or four-digit format.  
Example: YEAR(4)

Refer this example:-

<https://github.com/TopsCode/SQL/blob/master/Module%204.2.3.sql>

## Other Datatypes:-

**1. NUM:** A string object with a value chosen from a list of permitted values.

Example: ENUM('value1', 'value2', 'value3')

**2. SET:** A string object that can have zero or more values chosen from a list of permitted values.

Example: SET('value1', 'value2', 'value3')

**3. BOOLEAN:** Stores TRUE or FALSE.

Example: BOOLEAN

**4. BLOB (Binary Large Object):** Used for storing large binary data, like images or multimedia files.

Example: BLOB

# OPERATORS IN SQL:-

## 1. Arithmetic operators:-

Arithmetic operators are used to perform mathematical operations on numeric data.

(a) Addition:-

```
SELECT salary + bonus FROM employees;
```

(b) Subtraction:-

```
SELECT salary - deductions FROM employees;
```

(c) Multiplication:-

```
SELECT salary * 1.1 FROM employees;
```

(d) Division:-

```
SELECT salary / 12 FROM employees;
```

(e) Modulus:-Return the remainder of a division operation

```
SELECT salary % 2 FROM employees;
```

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.3.sql>

## 2) Logical operators

logical operators are used to combine multiple conditions in a WHERE clause

- (a) AND**:- returns True if both the conditions are True

Example:-

```
SELECT * FROM employees  
WHERE department = 'Sales' AND salary > 50000;
```

- (b) OR**:- returns True either one condition is True

Example:-

```
SELECT * FROM employees  
WHERE department = 'Sales' OR salary > 50000;
```

- (c) NOT**:-returns True if condition is False

Example:-

```
SELECT * FROM employees  
WHERE NOT department = 'Sales';
```

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.3.sql>

# Comparison Operators:-

Comparison Operator are used to compare two values.

- (a) **EQUAL TO (“=” ) :**

```
SELECT * FROM EMPLOYEES  
WHERE DEPARTMENT=“SALES”;
```

- (b) **NOT EQUAL TO (“<>”, “!=“ ) :**

```
SELECT * FROM EMPLOYEES  
WHERE DEPARTMENT <> “SALES”;
```

- (c) **Greater than (>) :**

```
SELECT * FROM EMPLOYEES  
WHERE SALARY>5000;
```

- (d) **Less than (<) :**

```
SELECT * FROM EMPLOYEES  
WHERE SALARY<5000;
```

**refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.3.sql>

**(e) Greater than equal to (“>=“) :-**

Example:-

```
SELECT * FROM EMPLOYEES  
WHERE SALARY>=5000;
```

**(f) Less than equal to (“<=“) :-**

Example:-

```
SELECT * FROM EMPLOYEES  
WHERE SALARY<=5000;
```

## **SET OPERATORS:-**

Set operators combine the results of two or more queries.

- (a) **UNION**:- Combines the results of two queries and removes duplicates.

```
SELECT first_name FROM employees  
UNION SELECT first_name  
FROM customers;
```

- (b) **UNION ALL**:- Combines the results of two queries without removing duplicates.

```
SELECT first_name FROM employees UNION ALL SELECT first_name FROM customers;
```

- (c) **INTERSECT**:- Returns only the common records between two queries.

```
SELECT first_name FROM employees  
INTERSECT  
SELECT first_name FROM customers;
```

**(d) EXCEPT (OR MINUS IN SOME DATABASE):-** Returns records from the first query that are not present in the second query.

**Example:-**

```
SELECT first_name FROM employees  
EXCEPT SELECT first_name FROM customers;
```

## OTHER OPERATORS:-

**(a) IN :-** Checks if a value is within a set of values.

- Example:-
- `SELECT * FROM employees WHERE department IN ('Sales', 'Marketing');`

**(b) BETWEEN :-** Check if a value is within a range.

- Example:-
- `SELECT * FROM employees where salary BETWEEN 40000 AND 50000;`

**(c) Like Operator:-** searches for the specified pattern in the column

Example:-

`SELECT * FROM employees WHERE first_name LIKE 'A%';`

**(d) IS NULL Operator :-** Check for null Values

Example:-

`SELECT * FROM employees  
WHERE middle_name IS NULL;`

# SELECT STATEMENT

- The SELECT statement in SQL is fundamental for querying data from a database. Below is an explanation of the SELECT statement and its variations, including the WHERE, GROUP BY, and HAVING clauses.

## **Basic Select Statement:-**

- The SELECT statement is used to select data from a database. The data returned is stored in a result table, sometimes called the result set.  
Syntax:- `SELECT column1, column2, ... FROM table_name;`

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.4.sql>

# **SELECT WITH WHERE CLAUSE:-**

- The WHERE clause is used to filter records. It is used to extract only those records that fulfil a specified condition.

## **syntax:-**

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

### **Example:-**

```
SELECT first_name, last_name FROM employees WHERE department = 'Sales';
```

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.4.sql>

# SELECT WITH GROUP BY CLAUSE:-

- The GROUP BY clause is used to arrange identical data into groups. It is often used with aggregate functions like COUNT, MAX, MIN, SUM, and AVG to group the result set by one or more columns.

## Syntax:-

```
SELECT column1, aggregate_function(column2) FROM table_name GROUP BY column1;
```

## Example:-

```
SELECT department, COUNT(*) FROM employees GROUP BY department;
```

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.4.sql>

# **SELECT CLAUSE WITH HAVING CLAUSE :-**

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions. The HAVING clause is used in combination with the GROUP BY clause to filter groups based on a condition.

## **Syntax:-**

```
SELECT column1, aggregate_function(column2) FROM table_name GROUP BY column1  
HAVING condition;
```

## **Example:-**

```
SELECT department, COUNT(*) FROM employees GROUP BY department HAVING  
COUNT(*) > 10;
```

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.4.sql>

## **COMBINING SELECT WITH WHERE , GROUP BY AND HAVING CLAUSE:-**

You can combine all these clauses in a single SELECT statement to filter records, group them, and then apply additional filters on the groups.

### **Example:-**

```
SELECT department, COUNT(*) FROM employees  
WHERE hire_date > '2020-01-01'  
GROUP BY department  
HAVING COUNT(*) > 5;
```

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.4.sql>

In this example:-

The WHERE clause filters employees hired after January 1, 2020.

The GROUP BY clause groups the remaining employees by department.

The HAVING clause filters out departments that have 5 or fewer employees hired after January 1, 2020.

# AGGREGATE FUNCTION:-

Aggregation functions in SQL are used to perform calculations on multiple rows of a single column of a table and return a single value. They are often used with the GROUP BY clause. Here are some common aggregation functions, including COUNT, SUM, and DISTINCT, and how they are used to summarize data.

## 1. COUNT:-

The COUNT function returns the number of rows that match a specified condition.

Example:-`SELECT COUNT(employee_id) FROM employees WHERE department = 'Sales';`

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.5.sql>

## 2. SUM FUNCTION:-

The SUM function returns the total sum of a numeric column.

```
SELECT SUM(salary) FROM employees WHERE department ='Engineering';
```

3. Similarly we can use **MIN**, **MAX** , **AVG** function to find the minimum ,maximum and Average of the data.

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.5.sql>

# DISTINCT FUNCTION:-

The DISTINCT keyword is used in conjunction with an aggregate function to return the unique values in a column.

## Syntax:-

```
SELECT DISTINCT column_name FROM table_name;
```

## Example:-

```
SELECT DISTINCT department FROM employees;
```

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.5.sql>

# LIMIT CLAUSE:-

The LIMIT clause in MySQL is used to restrict the number of rows returned by a query. This is especially useful when you need to retrieve a specific subset of data, such as the top results, a specific range of rows, or when paginating results. Here's how to use the LIMIT clause effectively.

## **Example:-**

- **Limit with where:-**

```
SELECT first_name, last_name FROM employees WHERE department = 'Sales' LIMIT 5;
```

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.6.sql>

## OFFSET CLAUSE:-

The OFFSET clause is used to specify the number of rows to skip before starting to return the rows. It is often used in conjunction with LIMIT to paginate results.

### **Example:-**

```
SELECT first_name, last_name FROM employees LIMIT 10 OFFSET 10;
```

### **Explanation:**

LIMIT 10: This limits the result to 10 rows.

OFFSET 10: This skips the first 10 rows in the result set.

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.6.sql>

## **SELECT WITH ALIAS NAME :-**

The AS keyword in SQL is used to create an alias for a column or an expression in a SELECT query. An alias is a temporary name given to a column or an expression in the result set. Aliases make column names more readable and are useful for giving descriptive names to computed columns.

### **Example:-**

```
SELECT first_name AS "First Name", last_name AS "Last Name", salary AS "Employee  
Salary" FROM employees;  
SELECT first_name, last_name, (salary * 12) AS "Annual Salary" FROM employees;
```

**Refer this Example:-**<https://github.com/TopsCode/SQL/blob/master/Module%204.8.sql>

# JOINS IN SQL:-

Joins in SQL are used to combine rows from two or more tables based on a related column between them. Here's an overview of different types of SQL joins, including INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN.

## INNER JOIN:-

An INNER JOIN returns only the rows that have matching values in both tables. If there is no match, the row will not be included in the result set.

### Example:-

```
SELECT employees.first_name, employees.last_name, departments.department_name FROM  
employees INNER JOIN departments ON employees.department_id = departments.department_id;
```

**Explanation:-** This query returns only the employees who have a matching department in the departments table.

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.9.sql>

# LEFT JOIN(OR LEFT OUTER JOIN):-

A LEFT JOIN returns all rows from the left table (table1), and the matched rows from the right table (table2). If there is no match, NULL values are returned for columns from the right table.

## Example:-

```
SELECT employees.first_name, employees.last_name, departments.department_name FROM  
employees LEFT JOIN departments ON employees.department_id = departments.department_id;
```

**Explanation:-**This query returns all employees, including those who do not have a matching department. For those without a match, NULL will appear in the department\_name column.

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.9.sql>

# RIGHT JOIN (OR RIGHT OUTER JOIN):-

A RIGHT JOIN returns all rows from the right table (table2), and the matched rows from the left table (table1). If there is no match, NULL values are returned for columns from the left table.

## Example:-

```
SELECT employees.first_name, employees.last_name, departments.department_name FROM  
employees RIGHT JOIN departments ON employees.department_id = departments.department_id;
```

## Explanation:-

This query returns all departments, including those that do not have any employees.

For those without a match, NULL will appear in the first\_name and last\_name columns.

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.9.sql>

# FULL JOIN( OR FULL OUTER JOIN):-

A FULL JOIN returns all rows when there is a match in either the left table or the right table. If there is no match, NULL values are returned for columns from the table without a match.

## Example:-

```
SELECT employees.first_name, employees.last_name, departments.department_name FROM  
employees FULL JOIN departments ON employees.department_id = departments.department_id;
```

## Explanation:-

This query returns all employees and all departments. If an employee does not have a department, NULL will appear in the department\_name column, and if a department does not have any employees, NULL will appear in the first\_name and last\_name columns.

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.9.sql>

## Order By clause:-

The ORDER BY clause is a SQL command used to sort the result set of a query in ascending or descending order based on one or more columns. It is typically used in conjunction with the SELECT statement to organize the output of a query in a specific sequence.

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.10.sql>

# ADVANCE SQL QUERIES:-

1. **Subqueries:-** Subqueries are queries nested within another query.

**Example:-**

```
SELECT first_name, last_name FROM employees WHERE department_id = (SELECT department_id FROM departments WHERE department_name = 'Sales');
```

**Explanation:-**This query selects employees who work in the Sales department.

**Refer this example:**

<https://github.com/TopsCode/SQL/blob/master/Module%204.11.sql>

## COALESCE FUNCTION:-

The COALESCE function returns the first non-NULL value from a list of arguments.

```
SELECT first_name, last_name, COALESCE(department_name, 'No Department') AS  
department_name FROM employees LEFT JOIN departments ON employees.department_id  
= departments.department_id;
```

### Explanation:-

This query returns employees with their department names, replacing NULL with 'No Department'.

# Working with Dates:-

## Date Functions:-

SQL provides several functions to work with dates.

**Current Date:-** This will return the current date

**syntax:-** Select CURRENT\_DATE;

**DATEADD and DATEDIFF:-** DATEADD adds a specified interval to a date, and DATEDIFF calculates the difference between two dates.

**syntax:-**

```
SELECT DATEADD(year, 1, '2023-01-01') AS NextYear; SELECT DATEDIFF(day, '2023-01-01', '2023-12-31')  
AS DaysDifference;
```

**(c) Extracting Parts of Dates:-**This query extracts the year and month from the hire\_date column.

**Syntax:-**

```
SELECT EXTRACT(year FROM hire_date) AS HireYear, EXTRACT(month FROM hire_date) AS  
HireMonth FROM employees;
```

IN dept\_id INT: An input parameter of type INT.

• IN dept\_id INT: An input parameter of type INT.



## STORED PROCEDURES :-

A stored procedure is a set of SQL statements that can be saved and reused. Stored procedures can accept parameters, execute SQL statements, and return results.

Suppose we want to create a stored procedure to get employee details by department.

### Example:-

```
CREATE PROCEDURE GetEmployeesByDepartment(IN dept_id INT) BEGIN SELECT first_name,  
last_name, department_id FROM employees WHERE department_id = dept_id; END;
```

**Explanation:-** IN dept\_id INT: An input parameter of type INT.

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.12.sql>

### Syntax to Call the Stored Procedure:-

```
CALL GetEmployeesByDepartment(2);
```

### Explanation:-

This calls the GetEmployeesByDepartment stored procedure with 2 as the department ID.

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.12.sql>

# Managing Stored Procedures:-

- **Show Procedures:-** SHOW PROCEDURE STATUS;
- **Dropping a stored Procedure:-** DROP PROCEDURE IF EXISTS GetEmployeesByDepartment;

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.12.sql>

## **VIEW:-**

A view is a virtual table based on the result set of an SQL query. Views do not store data themselves but provide a way to look at data in one or more tables.

**Example:-** Suppose we want to create a view to see employee details along with their department names.

**Query:-** CREATE VIEW EmployeeDepartment AS SELECT e.first\_name, e.last\_name, d.department\_name FROM employees e JOIN departments d ON e.department\_id = d.department\_id;

**Query to call View:-** SELECT \* FROM EmployeeDepartment;

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.12.sql>

# MANAGING VIEWS:-

## (a) Show views:-

```
SHOW FULL TABLES IN database_name  
WHERE TABLE_TYPE = 'VIEW';
```

## (b) Dropping view:-

```
DROP VIEW IF EXISTS EmployeeDepartment;
```

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.12.sql>

# Triggers :-

A trigger is a special kind of stored procedure that automatically executes (or "fires") in response to certain events on a particular table or view. Triggers can be used to enforce business rules, maintain audit trails, and perform validations and calculations.

## Types of Triggers:-

- (a) **BEFORE Trigger:** Executes before an operation (INSERT, UPDATE, DELETE).
- (b) **AFTER Trigger:** Executes after an operation (INSERT, UPDATE, DELETE).
- (c) **INSTEAD OF Trigger:** Used in views to perform actions instead of the triggering event

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.13.sql>

# Practical Implementation Trigger:-

**Syntax to create employee table:-**

```
CREATE TABLE EMP  
(EID INT PRIMARY KEY,  
ENAME VARCHAR(30),  
SALARY INT);
```

**Refer this example:-**

<https://github.com/TopsCode/SQL/blob/master/Module%204.13.sql>

## Syntax to insert the data into employee table:-

```
INSERT INTO EMP(EID,  
ENAME, SALARY)  
VALUES(1,'JOHN',30000),  
(2,'SHAME', 31000),  
(3, 'BOB', 42000),  
(4,'JAMES',43000),  
(5,'JASMINE', 45000),  
(6,'GARIMA',23000);
```

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.13.sql>

## Syntax to create the empty table in which Trigger result will stored:-

```
CREATE TABLE SALARY_AUDIT  
(OLD_SALARY INT,  
NEW_SALARY INT,  
SALARY_DIFFERENCE INT );
```

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.13.sql>

## Syntax to create the Trigger:-

DELIMITER //

```
CREATE TRIGGER SALARY_DIFFERENCE_TRIGGER
BEFORE UPDATE ON EMP
FOR EACH ROWBEGIN
DECLARE SALARY_DIFFERENCE DECIMAL(10,2);
SET SALARY_DIFFERENCE= NEW.SALARY -IFNULL(OLD.SALARY,0);
INSERT INTO SALARY_AUDIT(OLD_SALARY, NEW_SALARY,      SALARY_DIFFERENCE)
VALUES(IFNULL(OLD.SALARY,0), NEW.SALARY, SALARY_DIFFERENCE);
END //
```

Refer this example:- <https://github.com/TopsCode/SQL/blob/master/Module%204.13.sql>

**Syntax:-**

UPDATE EMP SET SALARY = 3000 WHERE EID= 3;

Now calling salary audit table , as trigger is created for Emp table and its result stored in salary audit table

**Syntax:-**

SELECT \* FROM SALARY\_AUDIT;

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.13.sql>

# Managing Triggers:-

**Show Triggers:-** SHOW TRIGGERS;

**Dropping a Triggers:-** DROP TRIGGER IF EXISTS trigger\_name;

**Conclusion:-** Triggers are a powerful feature in SQL that allow you to automate actions in response to changes in your database tables. They can help maintain data integrity, enforce business rules, and keep audit trails. By understanding and implementing triggers, you can ensure that your database actions are consistent and reliable, reducing the need for manual intervention and checks.

**Refer this example:-** <https://github.com/TopsCode/SQL/blob/master/Module%204.13.sql>

# NORMALIZATION :-

Normalization is a process in database design that organizes tables to minimize redundancy and dependency. It involves decomposing a database into smaller, related tables without losing data integrity. The main goals of normalization are to eliminate redundancy, ensure data dependencies make sense, and reduce the potential for anomalies during data operations like insertions, deletions, and updates.

Normalization involves several normal forms (NF), each with specific rules. Here are the first three normal forms, which are the most commonly applied:

# **First Normal Form (1NF):**

- Ensure that the table has a primary key.
- Each column must contain atomic (indivisible) values, and each column must contain values of a single type.
- Each record must be unique.

## **SECOND NORMAL FORM(2NF):-**

- Meet all the requirements of 1NF.
- Remove partial dependencies; no non-primary key attribute should depend on a part of a composite primary key.

## Third Normal Form (3NF):

- Meet all the requirements of 2NF.
- Remove transitive dependencies; no non-primary key attribute should depend on another non-primary key attribute.

# IMPORTING AND EXPORTING DATA IN SQL:-

## STEPS TO EXPORT THE DATA:-

- Open MySQL Workbench and connect to your database.
- Run the desired query to fetch data. Syntax:- Select \* from table\_name.
- Right-click on the result grid and choose Export Result Set....
- Select CSV format and save the file.
- Open the CSV file in Excel and save it as an Excel workbook.

# STEPS TO IMPORT THE DATA IN SQL:-

Importing data into MySQL Workbench can be done using the Import Data functionality. Here are the steps to import data into MySQL Workbench:

## Step 1:

- Open MySQL Workbench and Connect to Your Database
- Launch MySQL Workbench.
- Connect to your MySQL database server by clicking on the appropriate connection in the "MySQL Connections" window and entering your credentials.

## Step 2:

- Select the Table to Import Data Into
- In the "Navigator" panel on the left side of MySQL Workbench, expand your database schema to view the list of tables.
- Right-click on the table into which you want to import data.
- From the context menu, choose Table Data Import Wizard.

### **Step 3:**

- Configure the Import Settings
- The "Table Data Import Wizard" window will open.
- Click on the Import from Self-Contained File option if your data is stored in a file.
- Click on the Import from Disk button and browse to select the file containing the data you want to import.

**step 4:**

- Start the Import Process
- Once you have configured the import settings, click Next to proceed.
- Review the import settings and click Start Import.
- MySQL Workbench will import the data into the selected table

### **Step 5:-**

- Verify the Imported Data
- After the import process completes, you can verify the imported data by running SQL queries on the table.
- You can also use MySQL Workbench's data browsing features to view the imported data directly

## Step 6:

- Close the Import Wizard
- Once you have verified the imported data, you can close the "Table Data Import Wizard" window.
- **Conclusion:-**By following these steps, you can easily import data into MySQL Workbench from a file and populate your database tables with the desired data. Ensure that your data file is properly formatted and compatible with MySQL's data types to avoid import errors.

## ER MODEL:-

Entity-Relationship (ER) modeling is a fundamental technique used in database design to visually represent the structure of a database. It helps to organize and understand the relationships between different entities within a system. Here's an overview of ER modeling principles:

# Basic Concepts of ER MODEL:-

## 1. Entities:-

An entity is a real-world object or concept that exists independently and has attributes that describe its properties.

**Example:** In a university database, entities could include Student, Course, Professor, etc.

## 2. Attributes:-

Attributes are properties or characteristics of entities that describe them.  
Each entity has a set of attributes associated with it.

**Example:** For the Student entity, attributes could include StudentID, Name, DateOfBirth, etc.

### 3. Relationships:-

- Relationships describe how entities are related to each other.
- They can be one-to-one, one-to-many, or many-to-many.

**Example:** A Student entity may be related to a Course entity through an enrollment relationship.

#### 4. Keys :-

- Keys uniquely identify instances of an entity within a database.
- Primary keys are unique identifiers for each record in a table.
- Foreign keys establish relationships between tables.

Example: StudentID could be a primary key in the Student table and a foreign key in the Enrollment table.

- **Identify Entities:** Determine the main entities in the system you are modeling.
- **Define Attributes:** Identify and define the attributes for each entity.
- **Establish Relationships:** Determine how entities are related to each other and define the relationships between them.
- **Refine the Model:** Review and refine the ER diagram to ensure it accurately represents the system.
- **Normalize the Model (Optional):** If necessary, normalize the model to eliminate redundancy and improve efficiency.

# Module 5) Creating Dashboard with Visualization Tool

## • Introduction To Power BI , What is Power Bi , Why is Power BI and Power BI Installation and set up

- Capability to make intelligent business decisions.
- Makes information Helps you in analysis of the data.
- Provides data insights to everyone from any type / size / source of data
- Power BI is a self-service cloud based BI tool, which can be used for reporting and data analysis from a variety of data sources.
- Power BI provides a simple and user-friendly drag and drop interface so that users can work with it and get benefits from it.
- Power BI produces highly interactive and beautiful graphs, dashboards, and reports
- in a matter of minutes.

# Why Power BI?

- You can create a report and perform data manipulation easily.
- Power BI provides a 360-degree view to visualize the data and publish the resulting content.
- It supports natural query language.
- It allows you to fetch data from 60 different data sources.
- You can connect your data live or save it on your local machine.
- The product updates happen frequently and regularly.
- Power BI offers a simple learning curve.
- Performance of Power BI is much better as compared to the other BI tools in market.



**TOPS TECHNOLOGIES**

Training | Outsourcing | Placement | Study Abroad

Open a browser and navigate to the Power BI page.



Click on the **Download Free** button and enter the required details.



Click the **Get Started** button.



Locate the installed file and run it.



Follow the on-screen instructions.

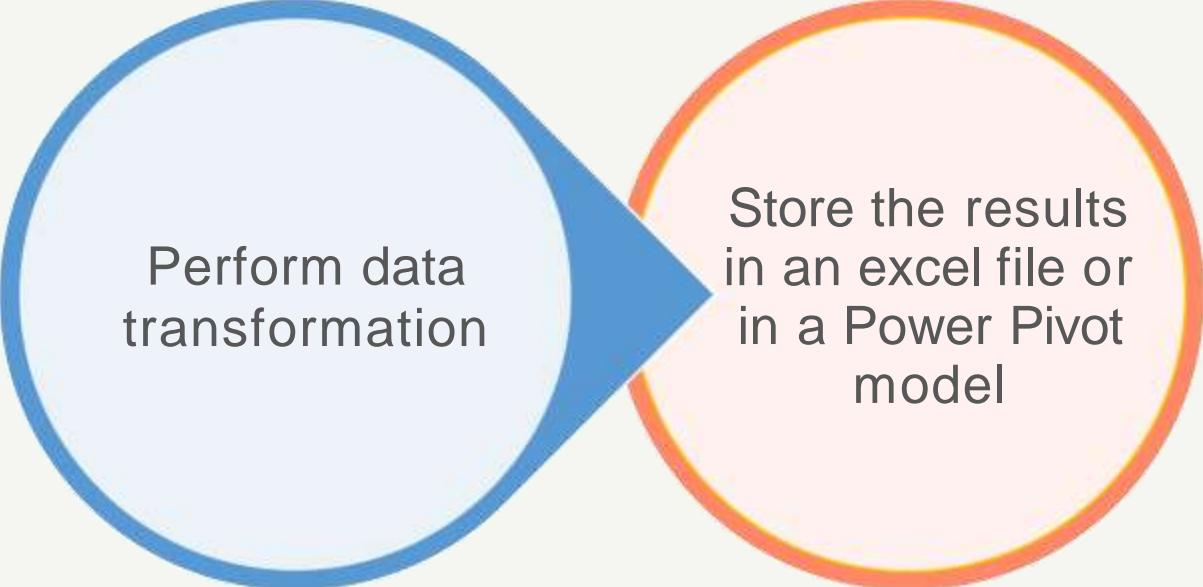
# Components Of Power BI

Power Query

Power Pivot

Power View

## **Power BI Desktop uses Power Query to:**



Perform data transformation

Store the results  
in an excel file or  
in a Power Pivot  
model

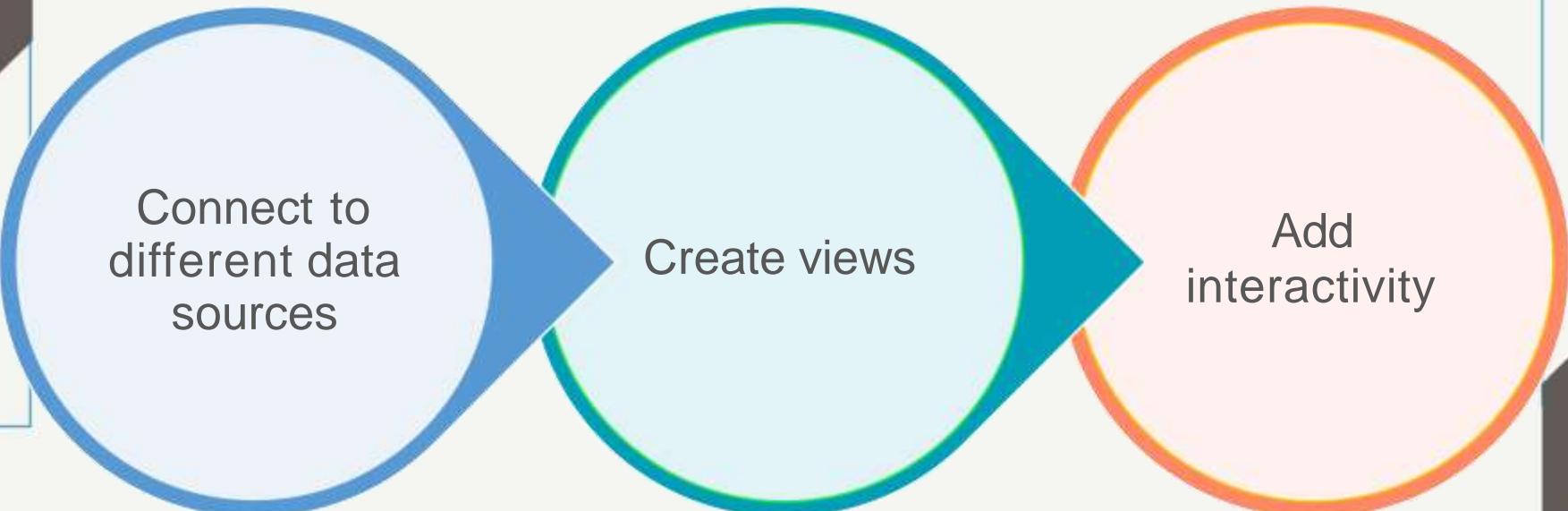
## With Power Pivot you can:

Perform data modeling

Build a schema, relationships, and so on

Create user-defined fields

## **Power View allows you to:**

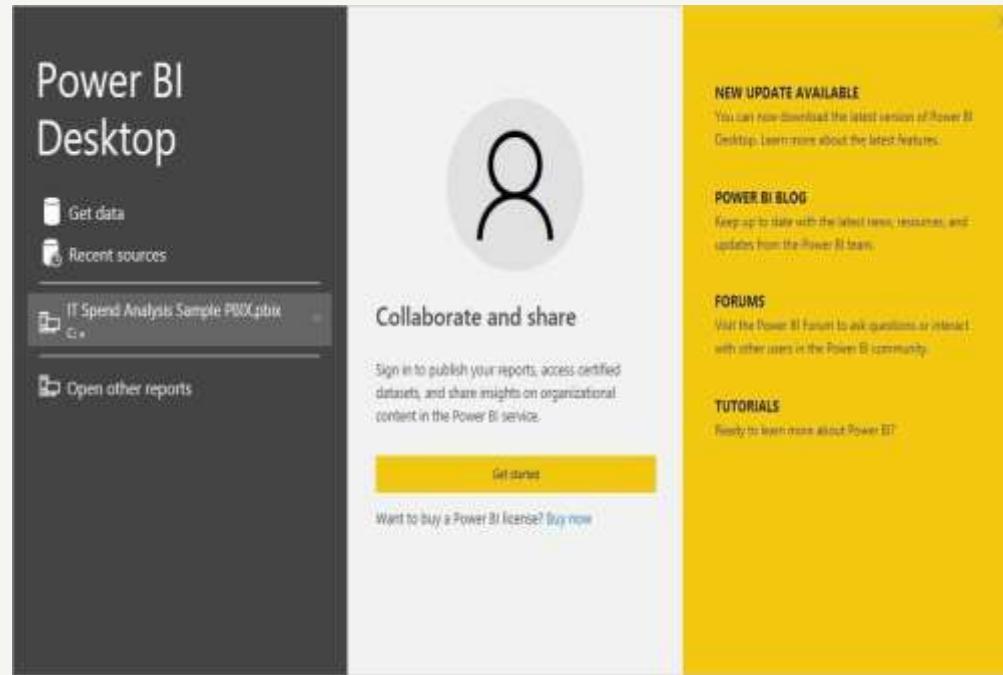


Connect to  
different data  
sources

Create views

Add  
interactivity

# Power BI Interface



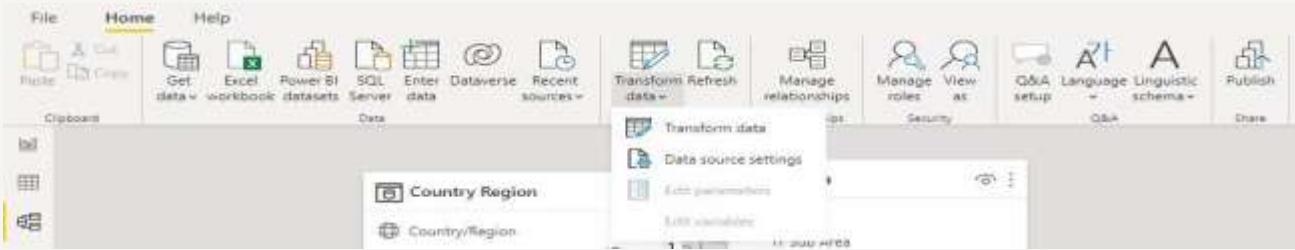
## Welcome Screen



# Power Query (ETL tool) Overview of the Query Editor

# Getting Familiar to Query Editor

You can use the Navigator window, or Home tab to launch the QueryEditor



Transform  
Menu



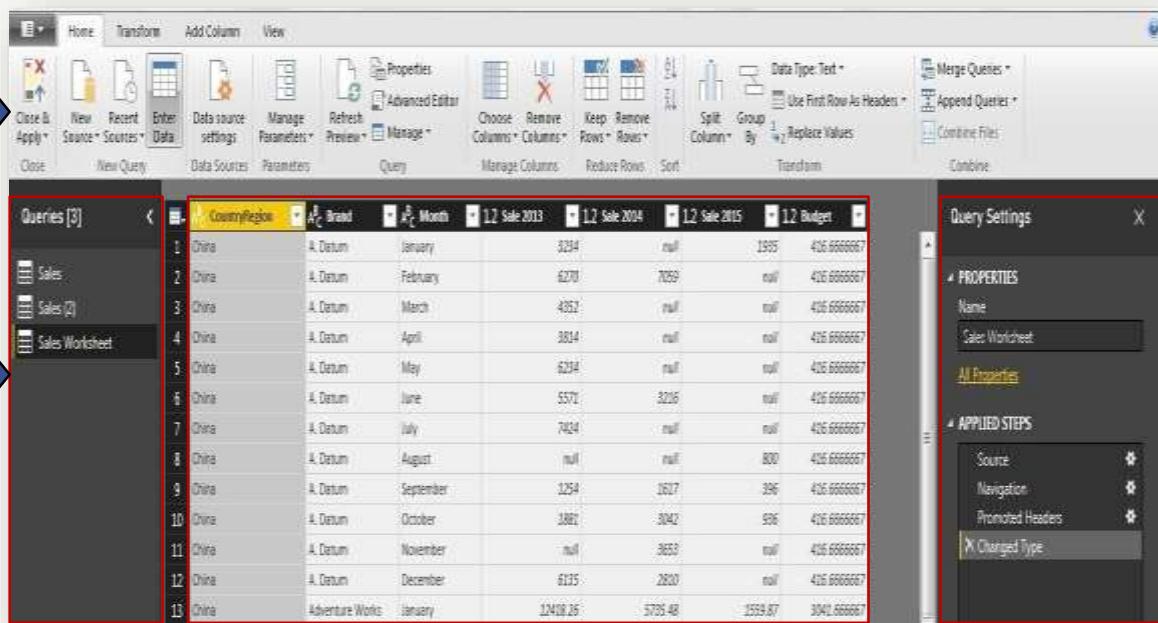
Left Pane



Query Settings



Center Pane



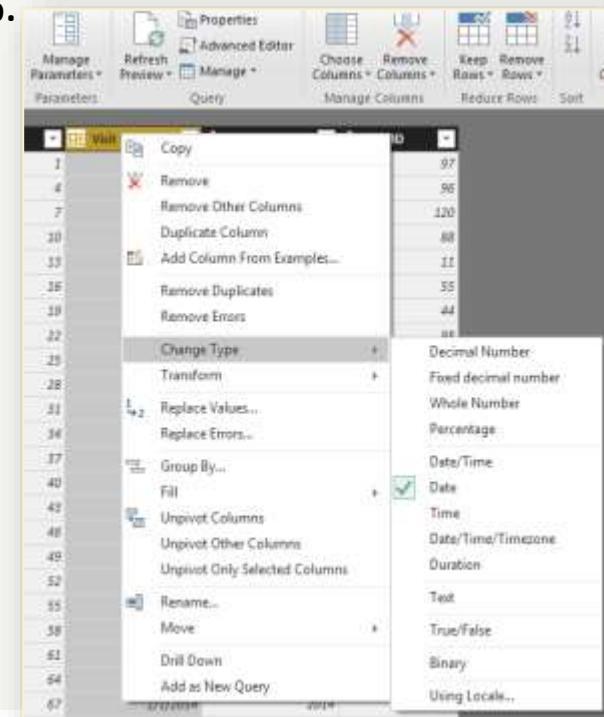
Row	CountryRegion	A Datum	B Month	C 12 Sale 2013	D 12 Sale 2014	E 12 Sale 2015	F 12 Budget
1	China	A. Datum	January	3294	null	1935	426.66666667
2	China	A. Datum	February	6270	7059	null	426.66666667
3	China	A. Datum	March	4952	null	null	426.66666667
4	China	A. Datum	April	3814	null	null	426.66666667
5	China	A. Datum	May	6234	null	null	426.66666667
6	China	A. Datum	June	5571	3216	null	426.66666667
7	China	A. Datum	July	7404	null	null	426.66666667
8	China	A. Datum	August	null	null	800	426.66666667
9	China	A. Datum	September	1254	1627	396	426.66666667
10	China	A. Datum	October	3881	3042	936	426.66666667
11	China	A. Datum	November	null	3653	null	426.66666667
12	China	A. Datum	December	6135	2820	null	426.66666667
13	Adventure Works	A. Datum	January	12408.26	5735.48	1559.87	3041.66666667

# Shaping Data: Change Data Type

As soon as the data is imported in Power BI, the system automatically attempts to convert the data type of the source column into a data type that better supports more efficient storage, calculations, and data visualization. For example, a column without any fractional values would be converted to a “Whole Number” data type by Power BI Desktop.

Following are the data types supported in Power BI:

- Decimal number
- Fixed decimal number
- Whole number
- Percentage
- Date/Time
- Date
- Time
- Date/Time/Timezone
- Duration
- Text
- True/False



Power BI allows you to split a column either by,

- Delimiter

There are additional options available as well to split at the,

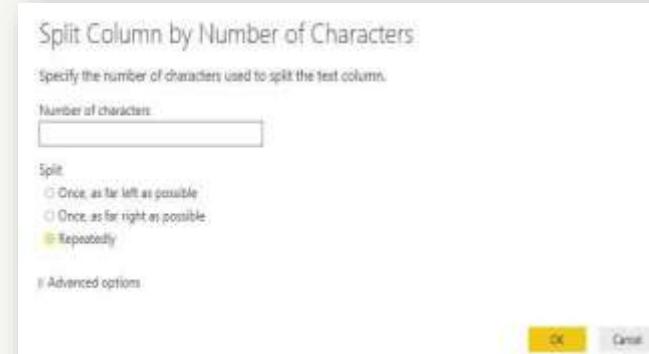
- ✓ Left most delimiter
- ✓ Right most delimiter or
- ✓ Each occurrence of the delimiter



- Number of characters

Power BI allows you to split a column by number of characters either,

- ✓ Once as far left as possible
- ✓ Once as far right as possible or
- ✓ Repeatedly



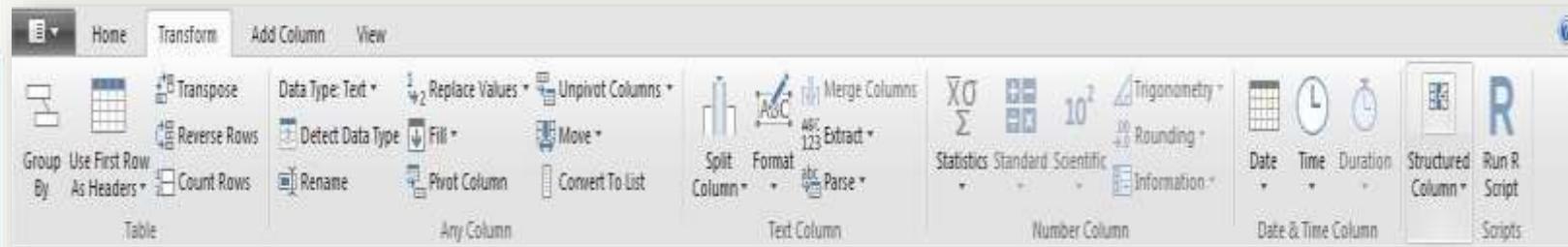
# Shaping Data – Text Transforms

Following are some of the commonly used text transformations available in Power BI:

- Lowercase
- Uppercase
- Capitalize each word
- Trim
- Clean
- Length
- Transpose
- Fill
- Reverse rows



Training | Outsourcing | Placement | Study Abroad



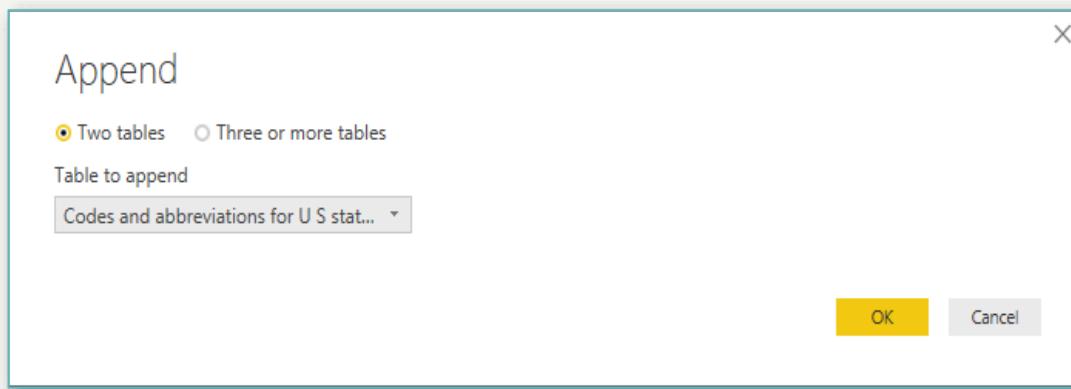
[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205\\_%20Creating%20Dashboard%20with%20Visualization%20Tool/Powerquery%20Editor](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205_%20Creating%20Dashboard%20with%20Visualization%20Tool/Powerquery%20Editor)



## Combine Data — Append Queries

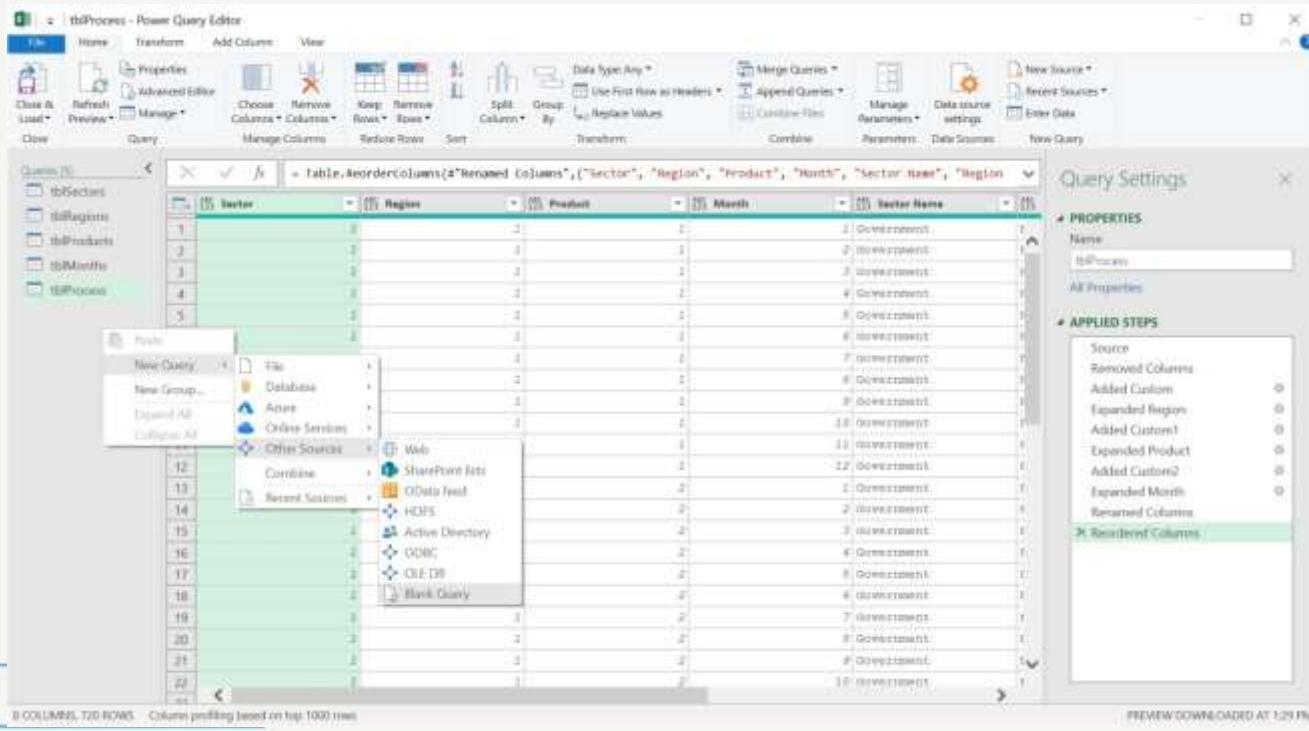
- You can append queries when you want to add one or more rows to an existing query.
- Using the “Append Queries” option, you can append different set of data rows coming in from different queries.
- It's important to have the same number and name of columns for this option to work accurately.

## Data Transformation



## Create a Blank Query:-

If the PQ Editor is not already open, use the keyboard shortcut Alt-A PN L to launch it.



The screenshot shows the Microsoft Power Query Editor interface. The main area displays a table with columns: Sector, Region, Product, Month, and Sector Name. The 'Sector' column has values 1 through 5. The 'Region' column has values 1 through 5. The 'Product' column has values 1 through 5. The 'Month' column has values 1 through 5. The 'Sector Name' column lists 'Government' repeated 10 times. On the left, the 'Queries' pane shows several tables: tbSector, tbRegions, tbProducts, tbMonths, and tbPhotos. A context menu is open over the 'tbPhotos' entry, with 'Blank Query' highlighted. The top ribbon bar shows 'File', 'Home', 'Transform', 'Add Column', and 'View'. The 'Transform' tab is selected. The 'Applied Steps' pane on the right shows a single step: 'Renamed Columns'.

Once inside the PQ Editor find the Queries pane on the left then right click inside that pane and use pop-up menu option:-

***New Query > Other Source > Blank Query.***

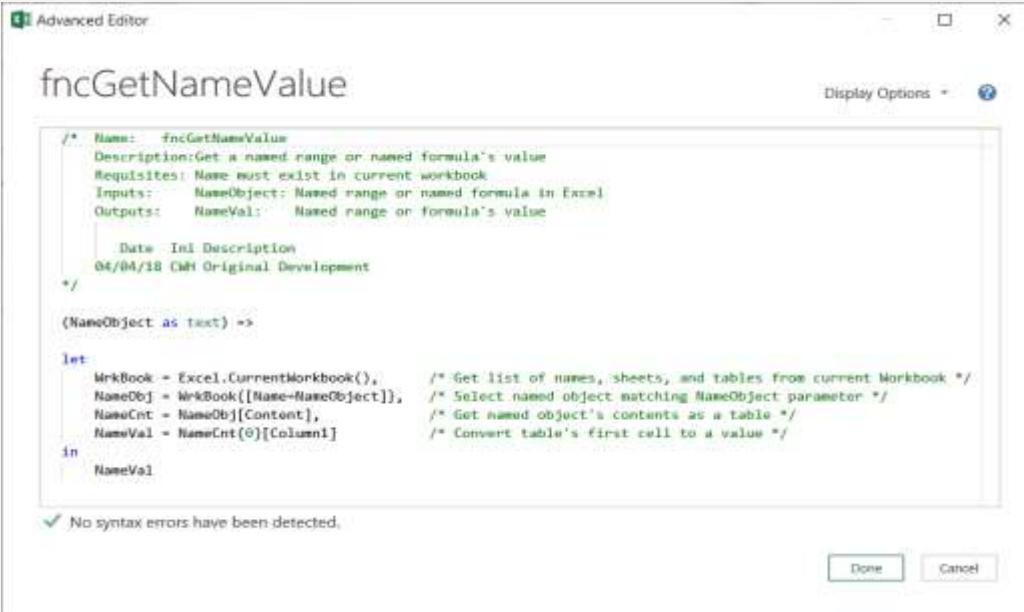
This creates **Query1** in the Queries pane.

## **Rename it**

Right click on **Query1** and select ***Rename***. Change the name to ***fncGetNameValue***.

**Open it in Advanced Editor**

**Click on fncGetNameValue and then, from PQ's ribbon, take menu option *Home > Advanced Editor*.**



The screenshot shows the Microsoft Power Query Advanced Editor window. The title bar says "Advanced Editor". The main area displays the following VBA code:

```
/* Name: fncGetNameValue
Description: Get a named range or named formula's value
Requisites: Name must exist in current workbook
Inputs: NameObject: Named range or named formula in Excel
Outputs: NameVal: Named range or formula's value

Date: 04/04/18 Author: Original Development
*/
(NameObject as text) =>

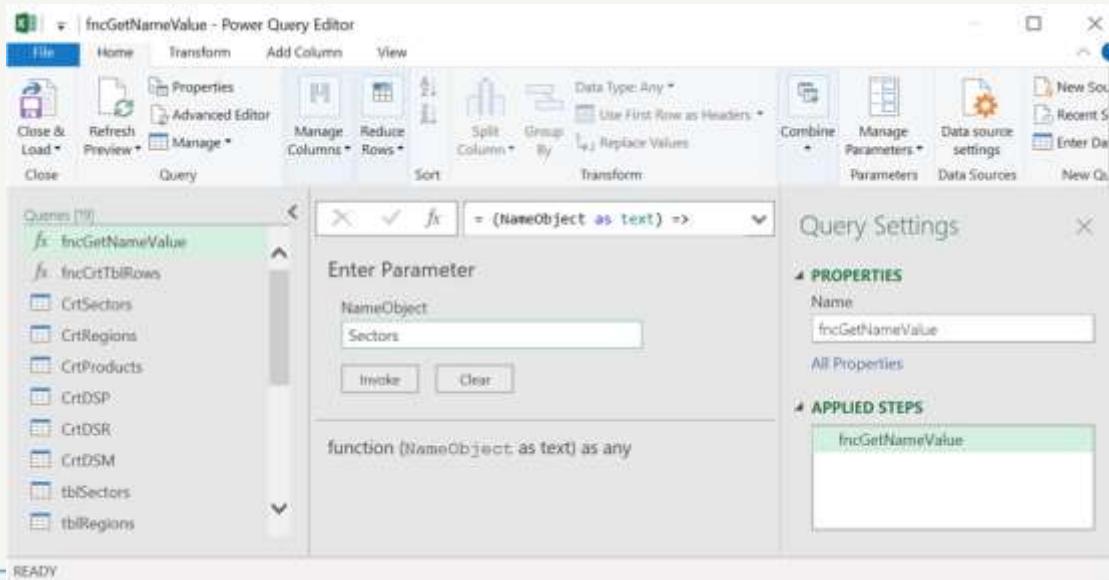
let
    WrkBook = Excel.CurrentWorkbook(),
    NameObj = WrkBook([Name=NameObject]),
    NameCnt = NameObj[Content],
    NameVal = NameCnt{0}[Column1]
in
    NameVal
```

At the bottom left, there is a green checkmark icon followed by the text "No syntax errors have been detected." At the bottom right, there are "Done" and "Cancel" buttons.

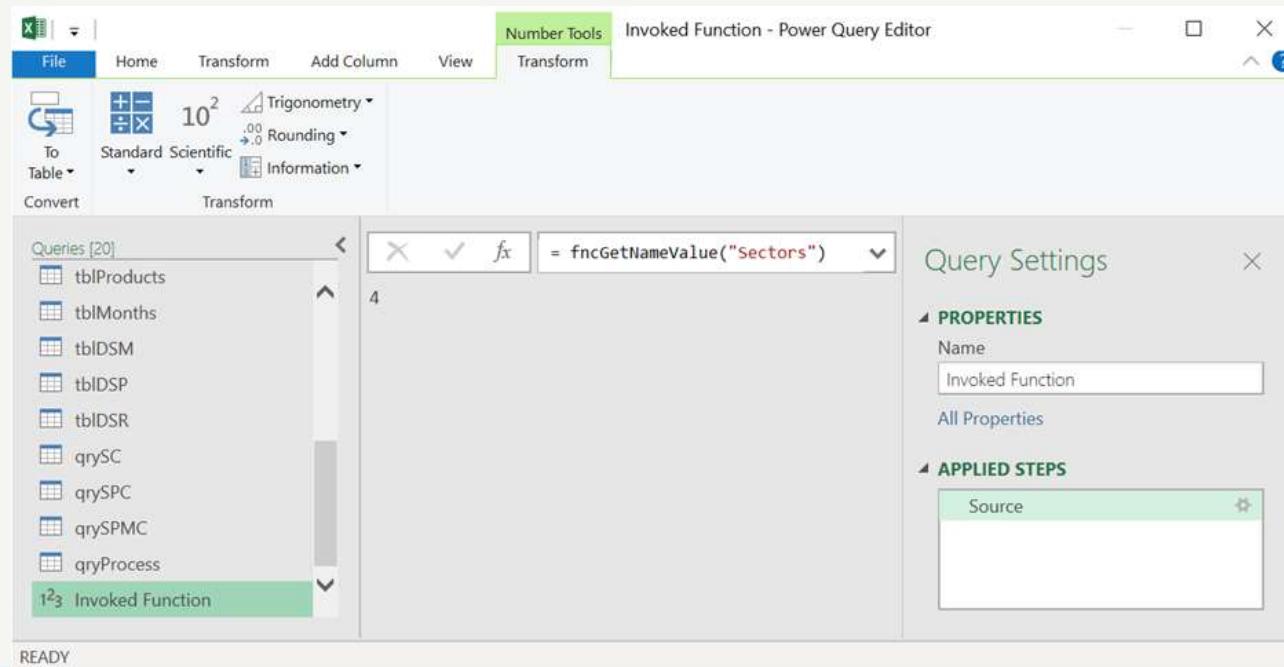
**Find code from here:-**

# Test

Click on the new function in the Queries pane then enter *Sectors* in the Name Object box.



Click **Invoke**. 4 should appear. When you are through admiring your work, delete **Invoked Function**.



## Power Pivot Table

“Pivot Table” in Power BI but there is a “Matrix Visualization” which is almost equivalent to a Pivot Table in Excel. It offers nearly the same features as Pivot does in Excel but they are named differently and of course placed differently.

To create a Pivot, pick up the “Matrix Visual” and NOT the Table visual



Matrix Visual is equivalent  
of Pivot Tables

Don't confuse it with Tables  
which is not like Pivot Tables



As soon as you create a Matrix, you'll get similar options like you do in Excel i.e. Rows, Columns and Values

The screenshot shows a data visualization interface. On the left is a table with two columns: 'Customer' and 'Amount'. The table contains 13 rows of data, including a 'Total' row at the bottom. On the right is a sidebar with three sections: 'Rows', 'Columns', and 'Values'. The 'Rows' section contains the word 'Customer'. The 'Columns' section has a placeholder 'Add data fields here'. The 'Values' section contains the word 'Amount'. Red arrows point from the text 'You can drag your fields here, just like you do in a pivot' to each of the three sections in the sidebar.

Customer	Amount
Boston Consultants	568050
Data Tronics	534900
Good Fly	704500
India Trotters	657900
Jindle Power Works	566600
MNTL	563750
Namint Enterprises	720600
Shah Associates	622550
Sharma & Co	621200
Shyam & Sharma Co	663000
VCC	634950
White Associates	547200
<b>Total</b>	<b>7405200</b>

Customer

Amount

Rows

Customer

Columns

Add data fields here

Values

Amount

You can drag your fields here,  
just like you do in a pivot

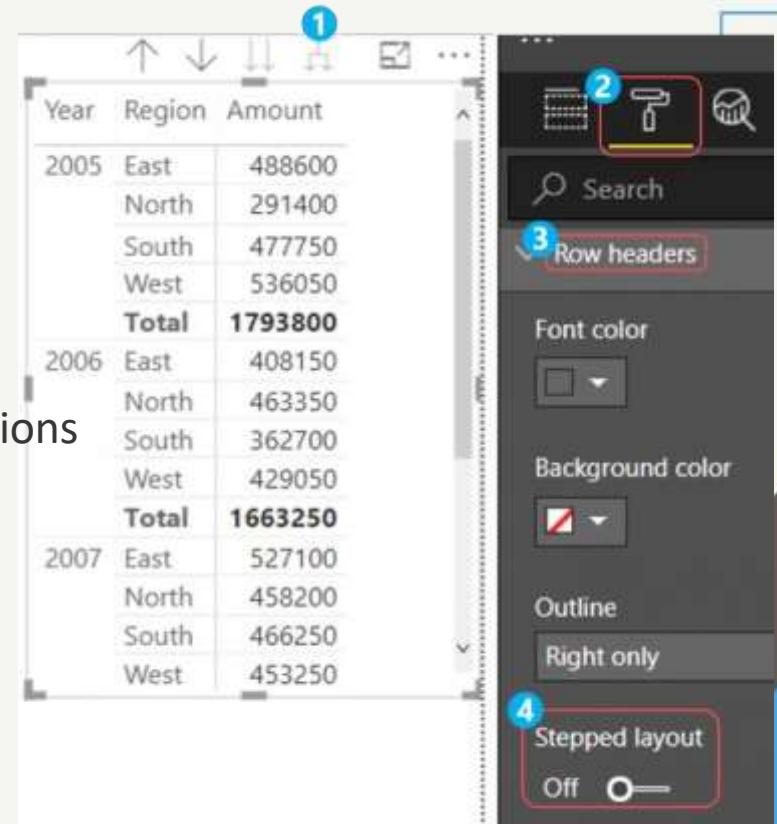
**Creating a Tabular / Classic View** – Any pivot veteran won't be able to stand a pivot table without this. If you don't know, Tabular / Classic View allows each field in rows to occupy a separate column.

Here is how a Tabular View looks in a Pivot Table Years and Region – placed in row labels are occupying different columns

Years	Region	Sum of Amount
2005	North	291400
	South	477750
	West	536050
	East	488600
<b>2005 Total</b>		<b>1793800</b>
2006	North	463350
	South	362700
	West	429050
	East	408150
<b>2006 Total</b>		<b>1663250</b>
2007	North	458200
	South	466250
	West	453250
	East	527100
<b>2007 Total</b>		<b>1904800</b>
2008	North	624100
	South	528350
	West	437950
	East	452950
<b>2008 Total</b>		<b>2043350</b>
<b>Grand Total</b>		<b>7405200</b>

This can be achieved in 4 simple steps in Power BI

1. After you create the Pivot and drag years and region in Rows, click on the double arrow icon to expand the Pivot
2. Then click on the “Format Tab” in the Visualizations Pane
3. Roll down to Row Headers
4. And turn off Stepped Layout



The screenshot shows a Pivot Table in Power BI with the following data:

Year	Region	Amount
2005	East	488600
	North	291400
	South	477750
	West	536050
	<b>Total</b>	<b>1793800</b>
2006	East	408150
	North	463350
	South	362700
	West	429050
	<b>Total</b>	<b>1663250</b>
2007	East	527100
	North	458200
	South	466250
	West	453250

To the right, the Format pane is open with the following steps highlighted:

1. Double arrow icon (Expand/Collapse)
2. Format tab icon
3. Row headers section
4. Stepped layout section, with "Off" selected

## Adding / Removing Subtotals and Grand Totals

Year	Region	Amount
2006	South	362700
	West	429050
	Total	<b>1663250</b>
2007	East	527100
	North	458200
	South	466250
	West	453250
	Total	<b>1904800</b>
2008	East	452950
	North	624100
	South	528350
	West	437950
	Total	<b>2043350</b>
	Total	<b>7405200</b>

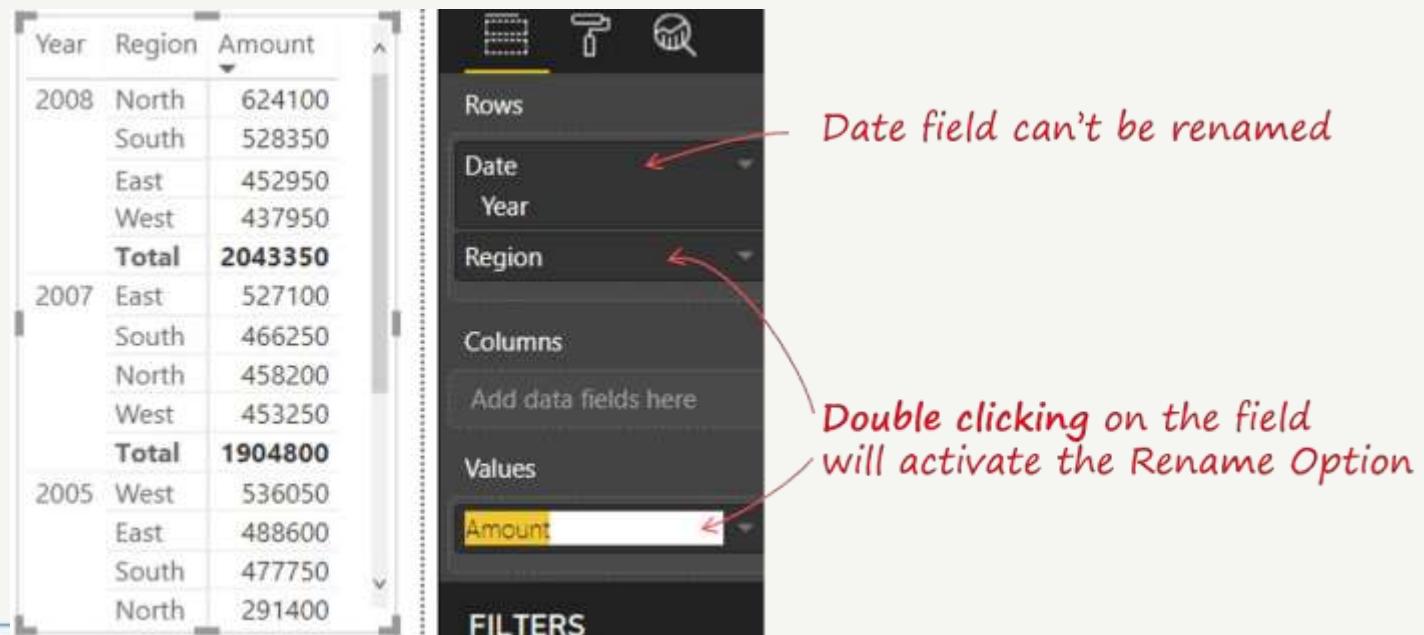


1. Simply click on the Format Tab once again
2. Roll down over to Subtotals and Grand total
  1. Apart from turning them on/off
  2. You'll also find options to play with font, styles, text size, color, label etc..

[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205\\_%20Creating%20Dashboard%20with%20Visualization%20Tool/Power%20pivot](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205_%20Creating%20Dashboard%20with%20Visualization%20Tool/Power%20pivot)

**Renaming the fields – Often to customize report users like to give custom names to field headers.**

**After dragging the fields in Rows/Columns/Values, just double click on the field to Rename it**



The screenshot shows a Power BI report interface. On the left is a table with three columns: Year, Region, and Amount. The table data is as follows:

	Year	Region	Amount
2008	North	624100	
	South	528350	
	East	452950	
	West	437950	
<b>Total</b>	<b>2043350</b>		
2007	East	527100	
	South	466250	
	North	458200	
	West	453250	
<b>Total</b>	<b>1904800</b>		
2005	West	536050	
	East	488600	
	South	477750	
	North	291400	

On the right is a Fields pane with sections for Rows, Columns, Values, and Filters. The Values section contains the field "Amount". Red arrows point from two red annotations to the "Date" field in the Rows section and the "Amount" field in the Values section. The annotations are:

- Date field can't be renamed
- Double clicking on the field will activate the Rename Option

## Power View (Visualization Charts)

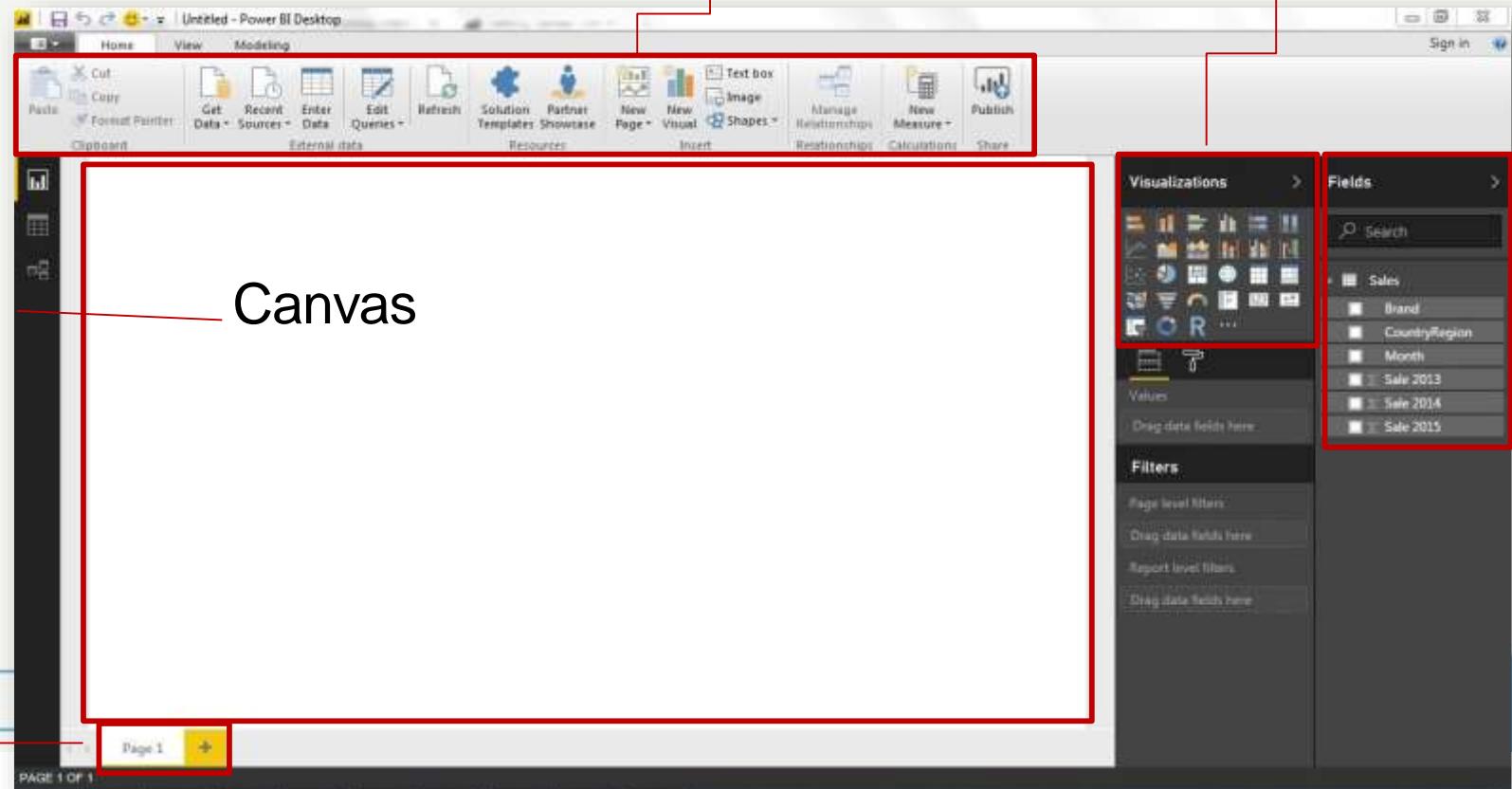
Ribbon

Visualizations

Fields

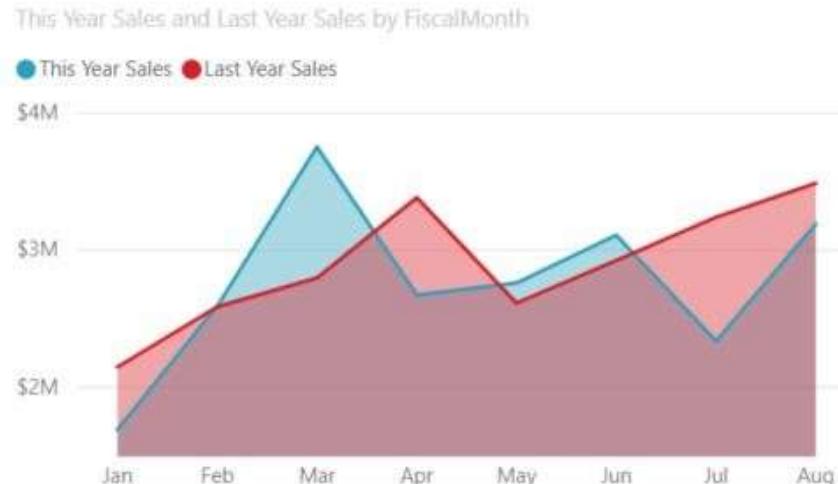
Canvas

Pages

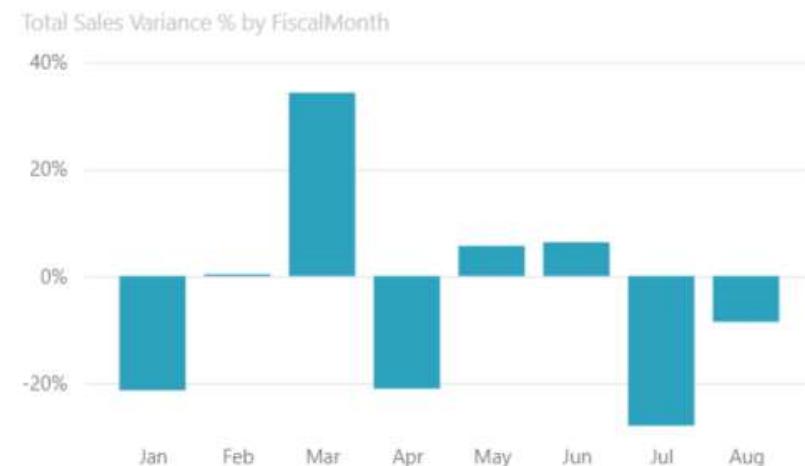




## Area charts: Basic (Layered) and Stacked



## Bar and column charts





**TOPS TECHNOLOGIES**

Training | Outsourcing | Placement | Study Abroad

## Cards Multi row

030-Kids

\$5.30

Average Unit Price

## Single number

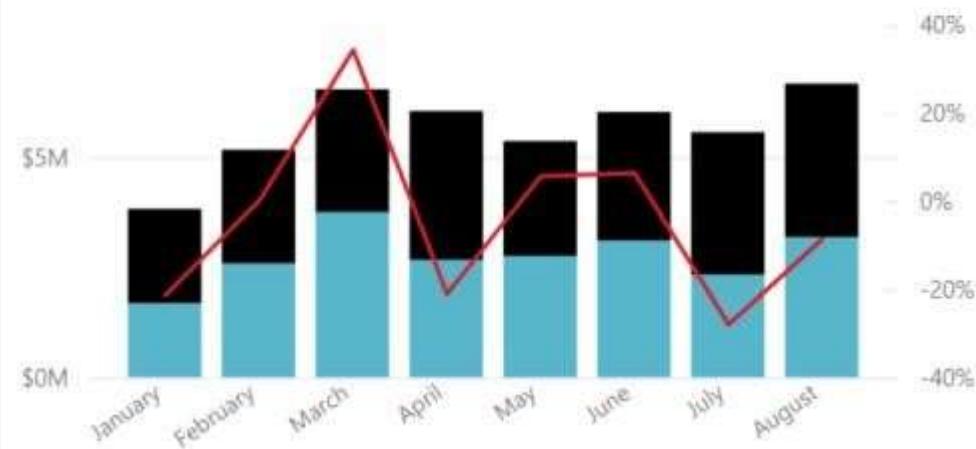
104

Total Stores

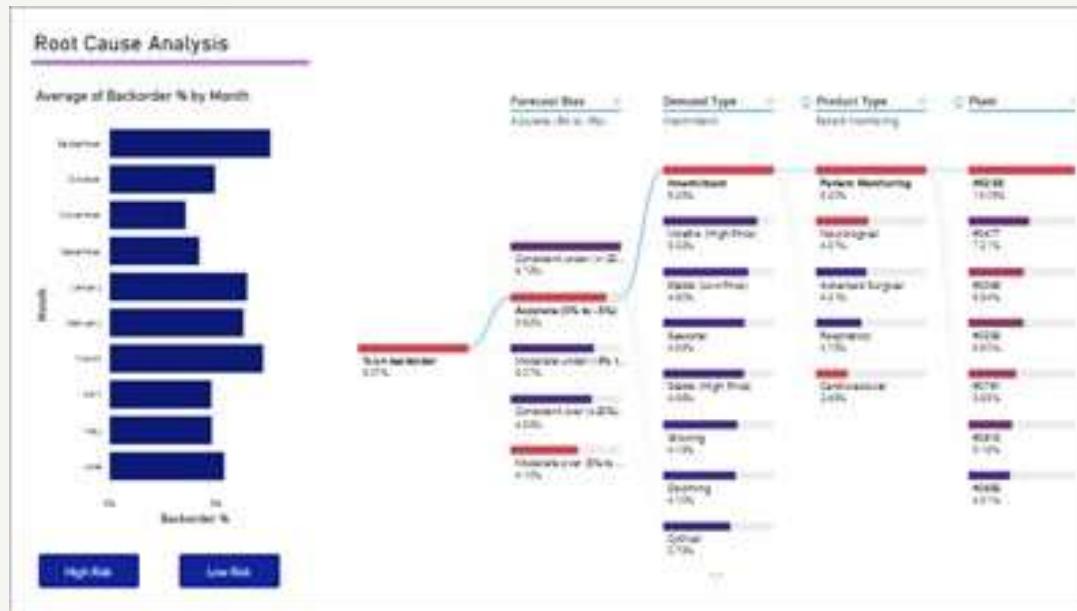
## Combo charts

This Year Sales, Last Year Sales and Total Sales Variance % by Month

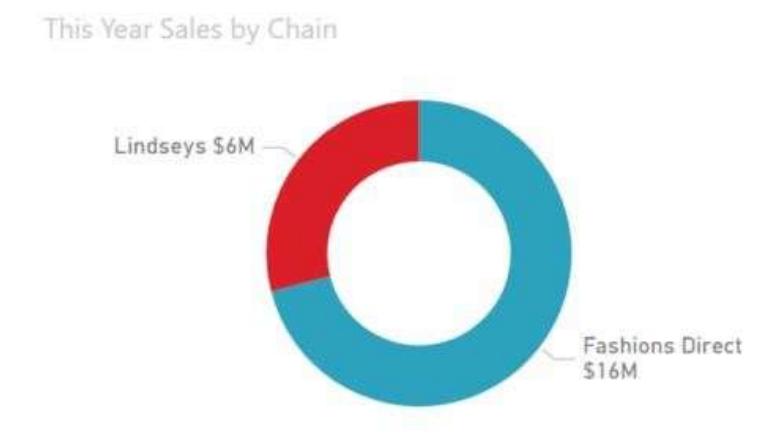
● This Year Sales ● Last Year Sales ● Total Sales Variance %



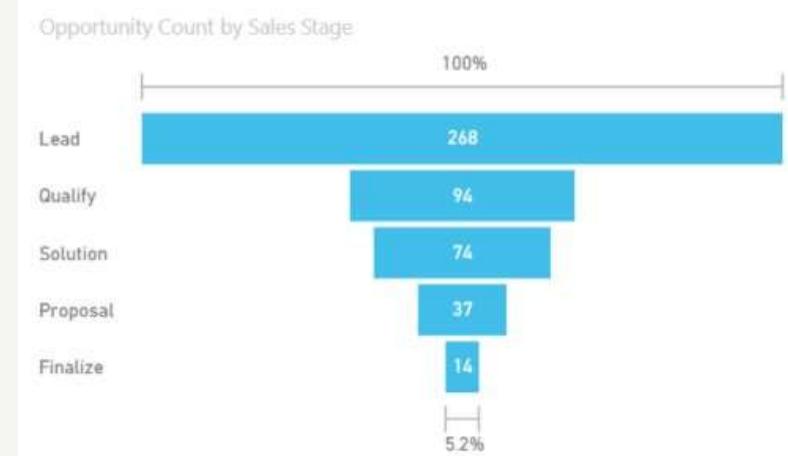
# Decomposition tree



## Doughnut charts



## Funnel charts





## Gauge charts

Average of Gross Sales



## Line charts

This Year Sales and Last Year Sales by FiscalMonth

This Year Sales    Last Year Sales

\$4M

\$3M

\$2M

Jan Feb Mar Apr May Jun Jul Aug

[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205%20Creating%20Dashboard%20with%20Visualization%20Tool/Charts](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205%20Creating%20Dashboard%20with%20Visualization%20Tool/Charts)

## Power BI Report View ,Model View , Power BI Table View

### Report View

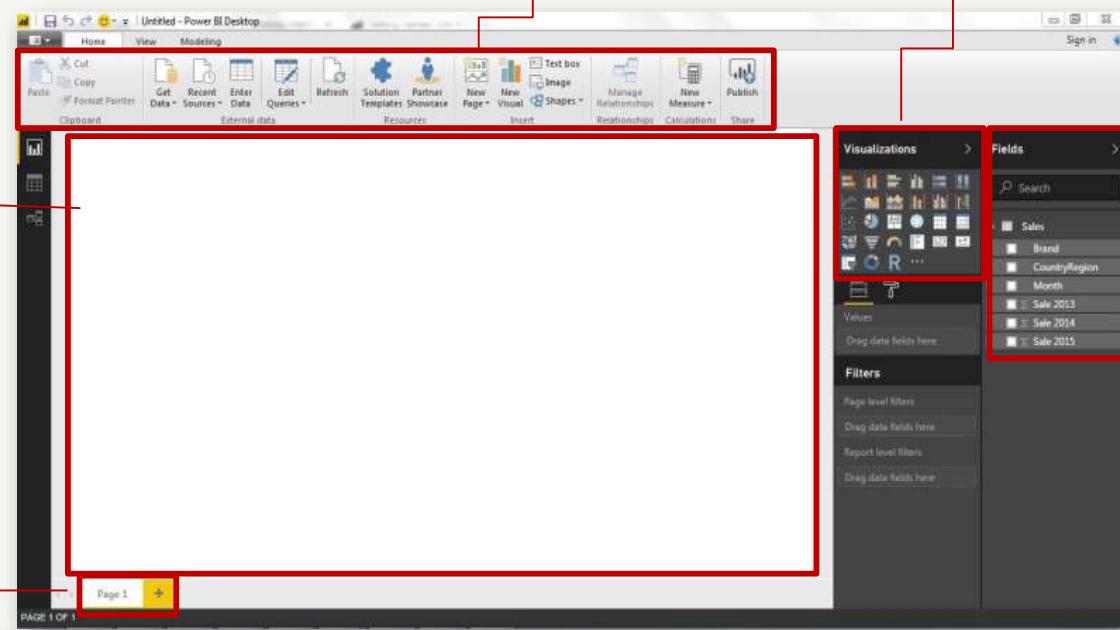
Canvas

Ribbon

Pages

Visualizations

Fields

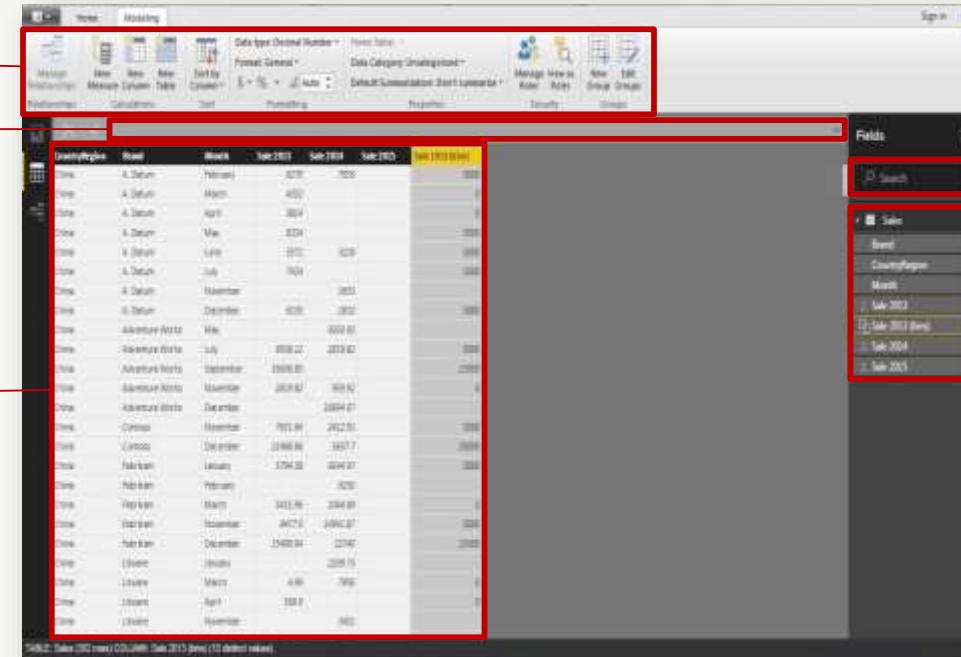


# Data View

Modeling ribbon

Formula bar

Grid

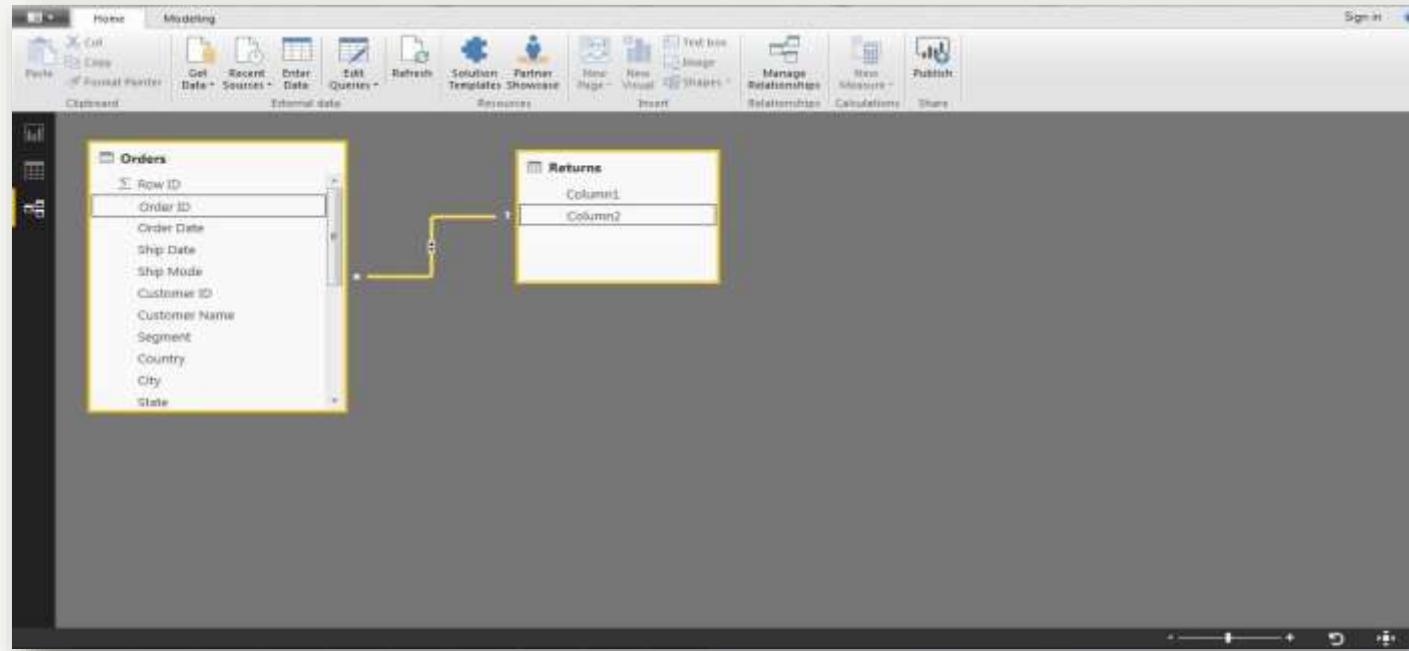


Customer	Region	Month	Sale 2013	Sale 2014	Sale 2015	Total 2013-2015
A. Sales	February	325	225	200	0	750
A. Sales	March	400	0	0	0	400
A. Sales	April	334	0	0	0	334
A. Sales	May	324	0	0	0	324
A. Sales	June	357	329	0	0	686
A. Sales	July	364	0	0	0	364
A. Sales	November	365	0	0	0	365
A. Sales	December	400	360	0	0	760
A. Sales	January	3600.81	0	0	0	3600.81
A. Sales	February	2000.22	2223.82	0	0	4224.04
A. Sales	March	2000.21	0	0	0	2000.21
A. Sales	April	2000.21	0	0	0	2000.21
A. Sales	May	2000.21	0	0	0	2000.21
A. Sales	June	2000.21	0	0	0	2000.21
A. Sales	July	2000.21	0	0	0	2000.21
A. Sales	August	2000.21	0	0	0	2000.21
A. Sales	September	2000.21	0	0	0	2000.21
A. Sales	October	2000.21	0	0	0	2000.21
A. Sales	November	2000.21	0	0	0	2000.21
A. Sales	December	2000.21	0	0	0	2000.21
B. Sales	February	325	225	200	0	750
B. Sales	March	400	204.81	0	0	604.81
B. Sales	April	3472.5	14962.81	0	0	18435.31
B. Sales	May	3472.5	14962.81	0	0	18435.31
B. Sales	June	3472.5	14962.81	0	0	18435.31
B. Sales	July	3472.5	14962.81	0	0	18435.31
B. Sales	August	3472.5	14962.81	0	0	18435.31
B. Sales	September	3472.5	14962.81	0	0	18435.31
B. Sales	October	3472.5	14962.81	0	0	18435.31
B. Sales	November	3472.5	14962.81	0	0	18435.31
B. Sales	December	3472.5	14962.81	0	0	18435.31

Search bar

Fields list

# Model View



## How to make Relations in Two or more tables in Power BI

### Why to Set up Relationships?

- When you work on multiple tables, chances are you're going to do some analysis using data from all those tables.
- You must set up relationships between those tables in order to calculate results and display the correct data in your dashboards. Power BI Desktop makes creating those relationships easy.
- Power BI Desktop allows you to set up relationships using two ways.
- You can use automatic, or manual method to create relationships between multiple tables.

## Auto Detection and Custom Relations

**Objective:** To manage relationships using Auto Detect and Custom Relations.

**Access:** To execute the practice, follow these steps:

Create the two tables or datasets in the Excel Sheet as shown below.

You can use any of the other datasets.

Date	Members	Car Loan Balance
9/17/2019	301	12087
9/16/2019	302	14567
9/15/2019	303	12233
9/14/2019	304	6534
9/13/2019	305	7658

Date	Members	House Loan Balance
9/12/2019	301	445678
9/13/2019	302	456732
9/10/2019	303	65342
9/11/2019	304	87675
9/9/2019	305	87235

## Auto Detection and Custom Relations

**Objective:** To manage relationships using Auto Detect and Custom Relations.

**Step 1:** Load the two tables or datasets to Power BI.

**Step 2:** After loading the data, click on the **Manage Relationships** tab on the top menu bar.

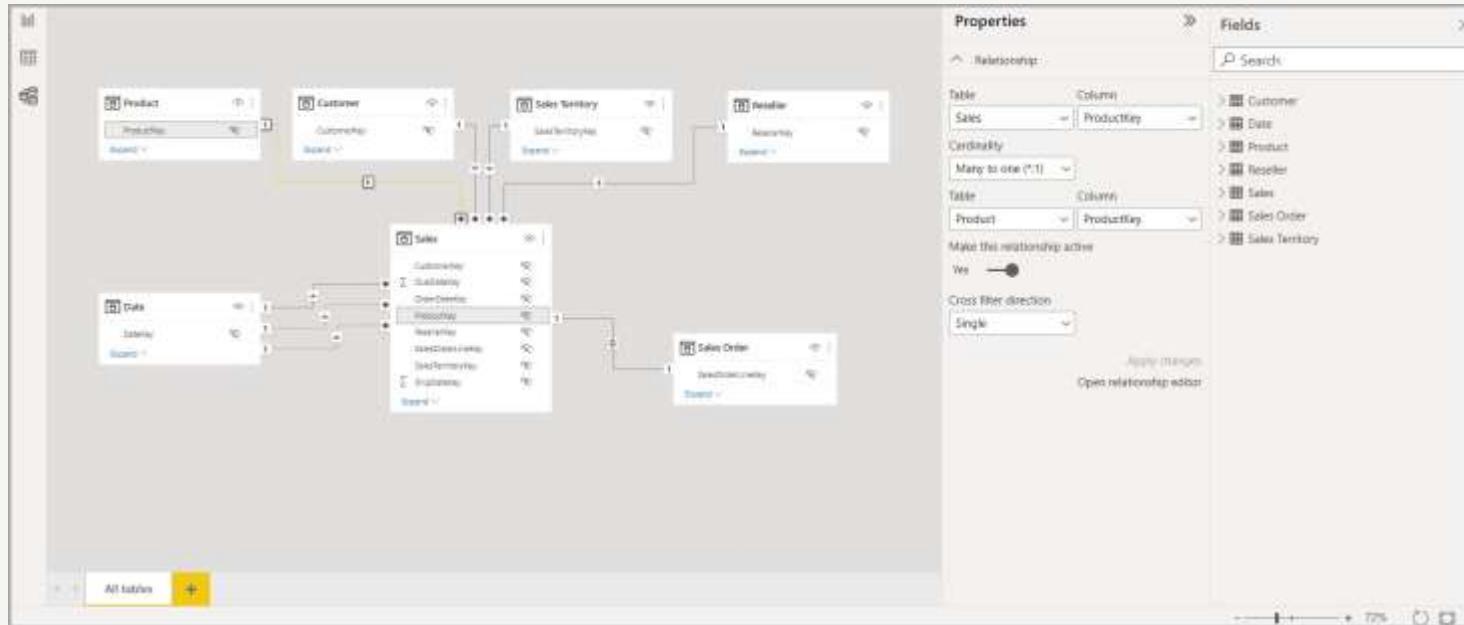
**Step 3:** Click on the **Autodetect** button. It will automatically detect the relations, if any.

**Step 4:** To view the relation, click on the **model** icon on the left pane.

**Step 5:** You will now see that the relationship is detected and is being represented.

## Create a relationship manually:-

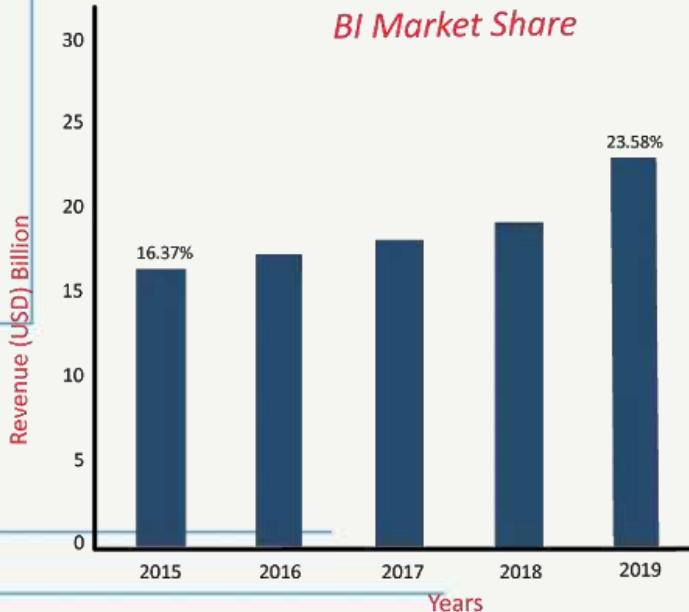
1. On the Modeling tab, select Manage relationships > New.
2. In the Create relationship dialog box, in the first table drop-down list, select a table. Select the column you want to use in the relationship.
3. In the second table drop-down list, select the other table you want in the relationship. Select the other column you want to use, and then select OK.



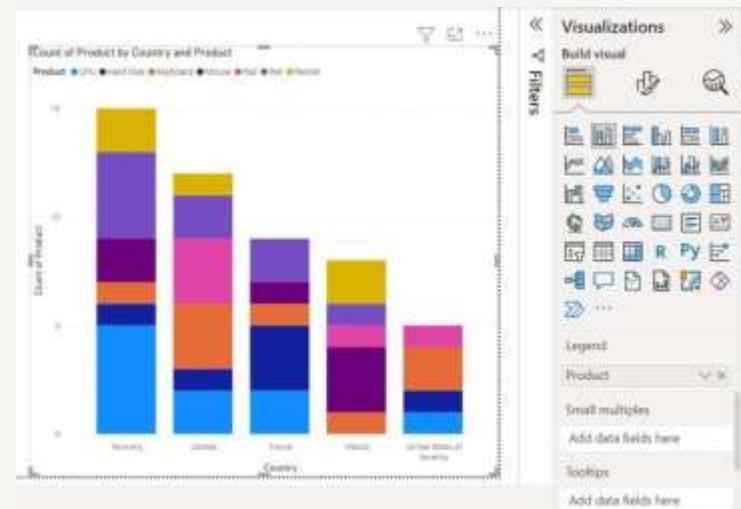
[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205%20Creating%20Dashboard%20with%20Visualization%20Tool/Data%20modeling](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205%20Creating%20Dashboard%20with%20Visualization%20Tool/Data%20modeling)

# Power BI Basic Power Charts

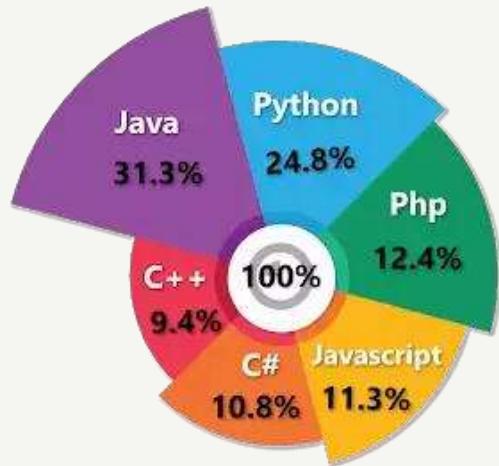
## Column Chart



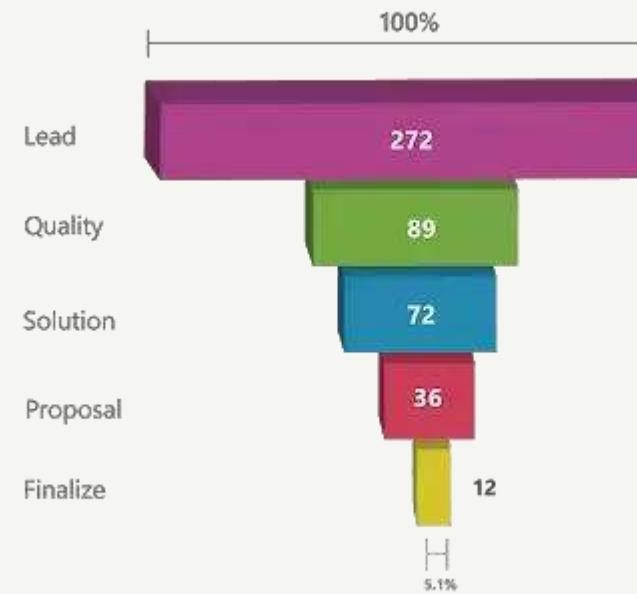
## Stacked Chart



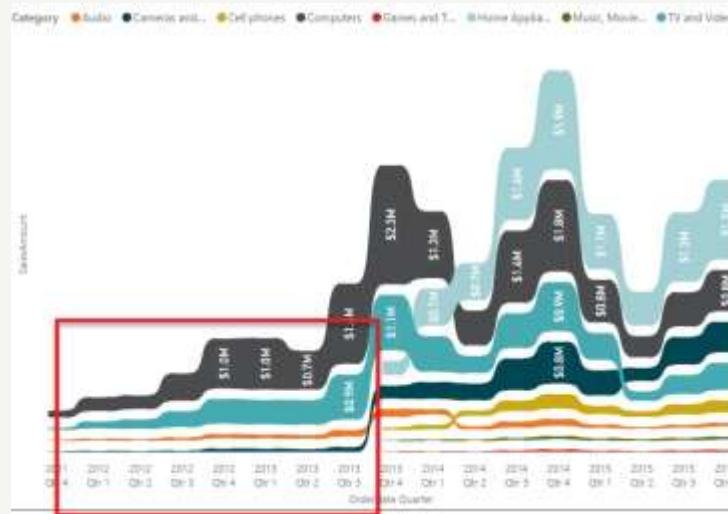
## Pie Chart



## Funnel Chart



## Ribbon Chart



[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205%20Creating%20Dashboard%20with%20Visualization%20Tool/Charts](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205%20Creating%20Dashboard%20with%20Visualization%20Tool/Charts)

# Types of Data connection power BI

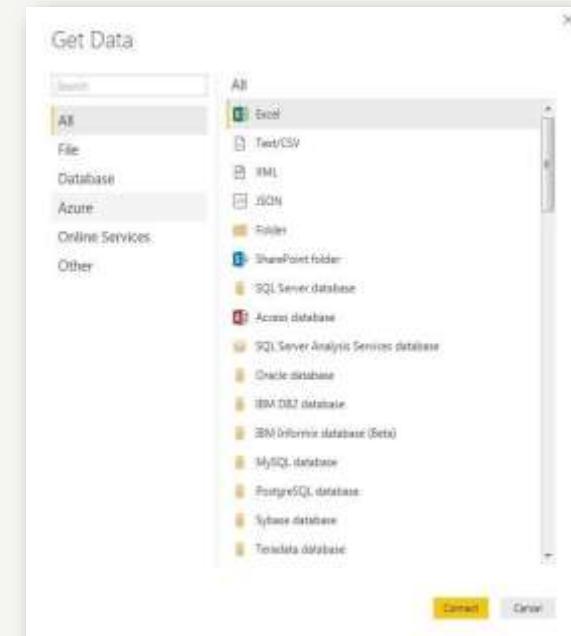
Power BI Desktop allows you connect to about 60 different types of data sources.



ORACLE®

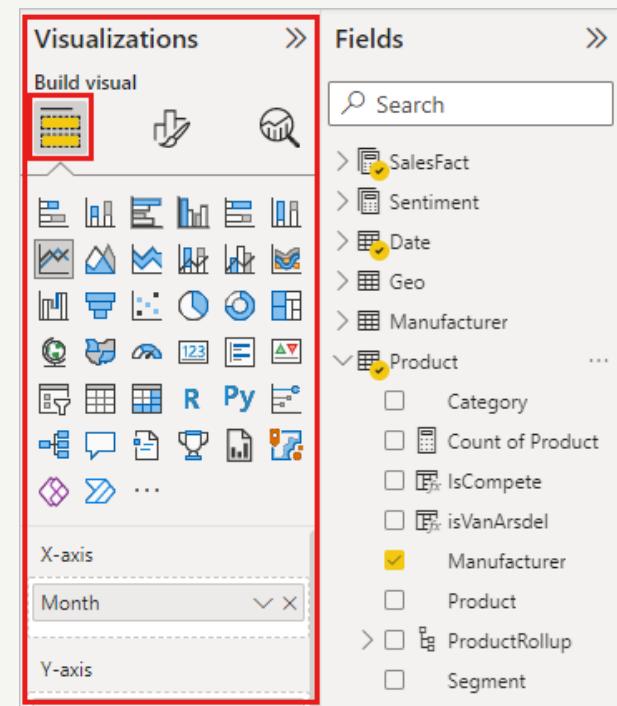


MySQL



## Format Tools in Power BI for Charts and Visualization

When you're editing a report and you select a visualization, the Visualizations pane appears. Use this pane to change visualizations. Directly the Visualizations pane, there are three icons: the Fields icon (a stack of bars), the Format icon (a paint brush), and the Analytics icon (a magnifying glass). In the image below, the Fields icon is selected, indicated by a yellow bar below the icon.





**When you select Format, the area below the icon displays the customizations available for the currently selected visualization.**



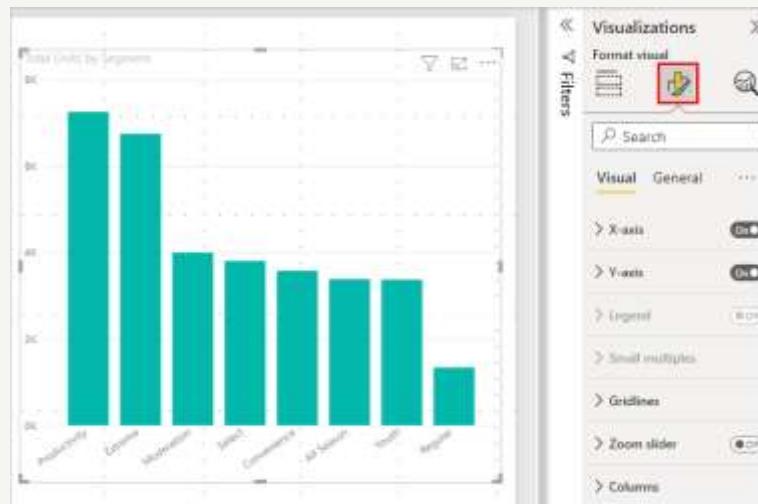
You can customize many elements of each visualization. The options available depend on the visual selected. Some of those options are:

- Legend
- Background
- X-axis
- Lock aspect
- Y-axis
- Border
- Data colors
- Shadow
- Data labels
- Tooltip
- Total labels
- Visual header
- Shapes
- Shapes
- Plot area
- Position
- Title
- Zoom

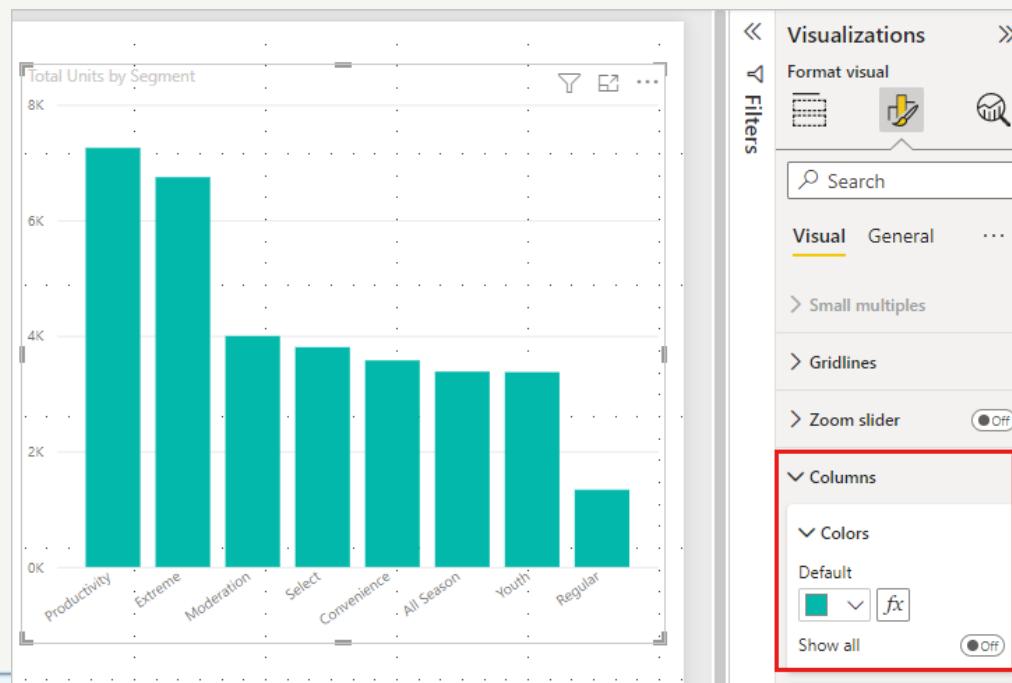
## Change colors in a visual

Let's walk through the steps necessary to customize colors on a visualization.

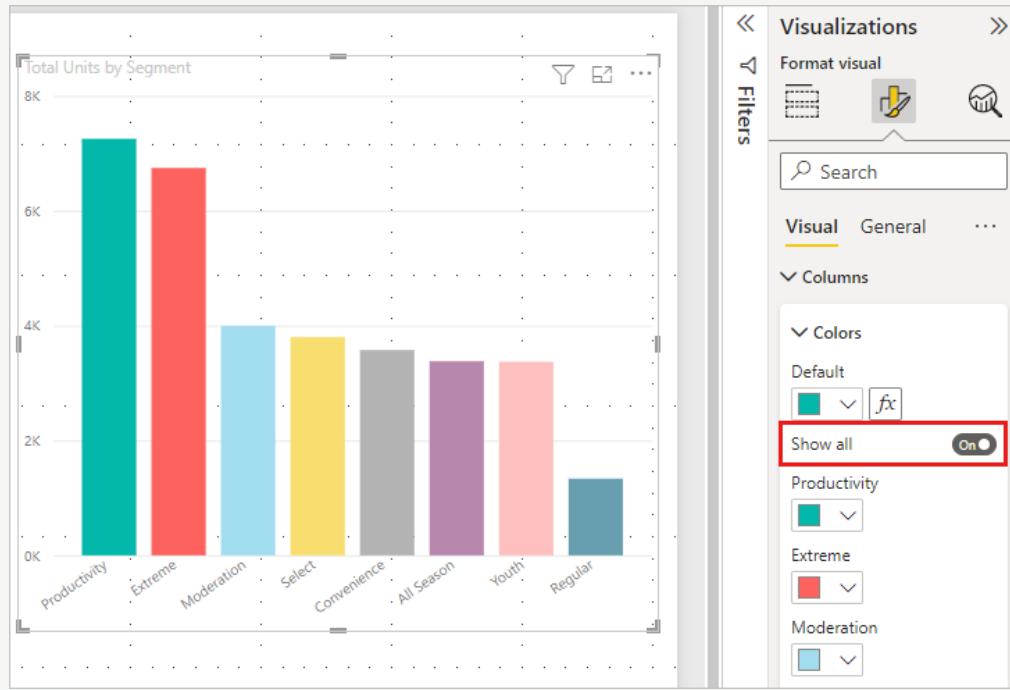
1. Select a visualization to make it active.
2. Select the paint brush icon to open the Formatting tab. The Formatting tab displays all the formatting elements available for the selected visual.



### 3. Select Colors to expand its available customizations.



**Change Show all to On, and select different colors for columns, rows, and lines, depending on the visual type.**



Here are a few tips for working with colors. The numbers in the following list are also shown on the following image, indicating where you can access and change these useful elements:-

- 1.Don't like the color? No problem, just select the down-arrow to open the color palette and select a new one.
- 2.Don't like any of the color changes? Select **Reset to default** from the bottom of the **Data color** section, and your colors revert to the default settings.
- 3.Want a color you don't see in the palette? Just select **More colors...**, and choose from the spectrum.



**TOPSTECHNOLOGIES**

Training | Outsourcing | Placement | Study Abroad

Visualizations >> Fields >>

Fields

Theme colors

White	Black	Cyan	Dark Gray	Red	Yellow	Gray	Light Blue	Orange	Purple
Light Gray	Dark Gray	Cyan	Gray	Red	Yellow	Gray	Light Blue	Orange	Purple
Light Gray	Dark Gray	Cyan	Gray	Red	Yellow	Gray	Light Blue	Orange	Purple
Gray	Black	Cyan	Dark Gray	Red	Yellow	Dark Gray	Light Blue	Orange	Purple
Gray	Black	Cyan	Dark Gray	Red	Yellow	Dark Gray	Light Blue	Orange	Purple
Gray	Black	Cyan	Dark Gray	Red	Yellow	Dark Gray	Light Blue	Orange	Purple

Recent colors

Red	Blue	Light Blue	Purple	Gray	Yellow	Red	Cyan	Dark Gray	White
-----	------	------------	--------	------	--------	-----	------	-----------	-------

 More colors...

 ▾

Regular  ▾

> Spacing

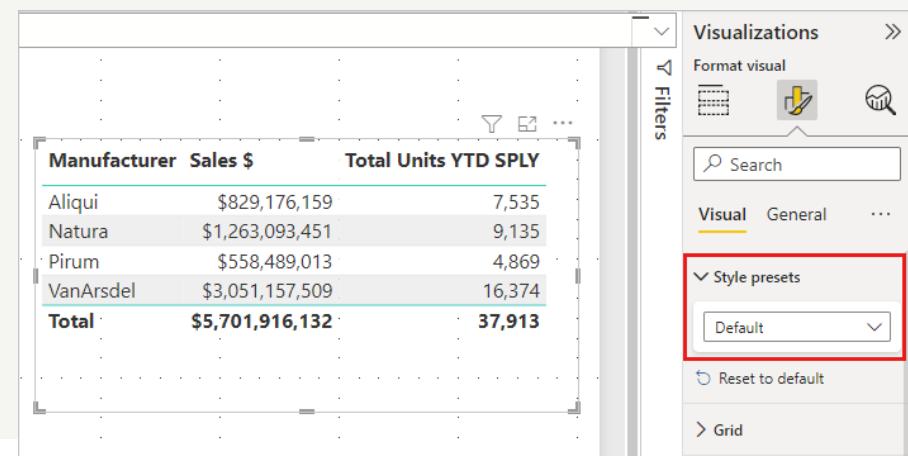
 Reset to default

Segment 

## Apply a style to a table

Some Power BI visualizations have a **Style** option. One click applies a full set of formatting options to your visualization, all at once.

1. Select a table or matrix to make it active.
2. Open the Formatting tab and select **Style presets**.



The screenshot shows a Power BI report with a table visualization. The table has columns: Manufacturer, Sales \$, Total Units, YTD, and SPLY. The data includes various manufacturers like Aliqui, Natura, Pirum, VanArsdel, and totals. The styling ribbon on the right is open, showing the 'Visualizations' tab selected. Under 'Style presets', a dropdown menu is open with 'Default' selected. Other options include 'Reset to default' and 'Grid'.

Manufacturer	Sales \$	Total Units	YTD	SPLY
Aliqui	\$829,176,159	7,535		
Natura	\$1,263,093,451	9,135		
Pirum	\$558,489,013	4,869		
VanArsdel	\$3,051,157,509	16,374		
<b>Total</b>	<b>\$5,701,916,132</b>	<b>37,913</b>		



Training | Outsourcing | Placement | Study Abroad

Manufacturer	Sales \$	Total Units YTD SPLY
Aliqui	\$829,176,159	7,535
Natura	\$1,263,093,451	9,135
Pirum	\$558,489,013	4,869
VanArsdel	\$3,051,157,509	16,374
<b>Total</b>	<b>\$5,701,916,132</b>	<b>37,913</b>

...

...

Visualizations

Format visual

Filters

Visualizations

Format visual

Filters

Visual General ...

Style presets

Default

Reset to default

Grid

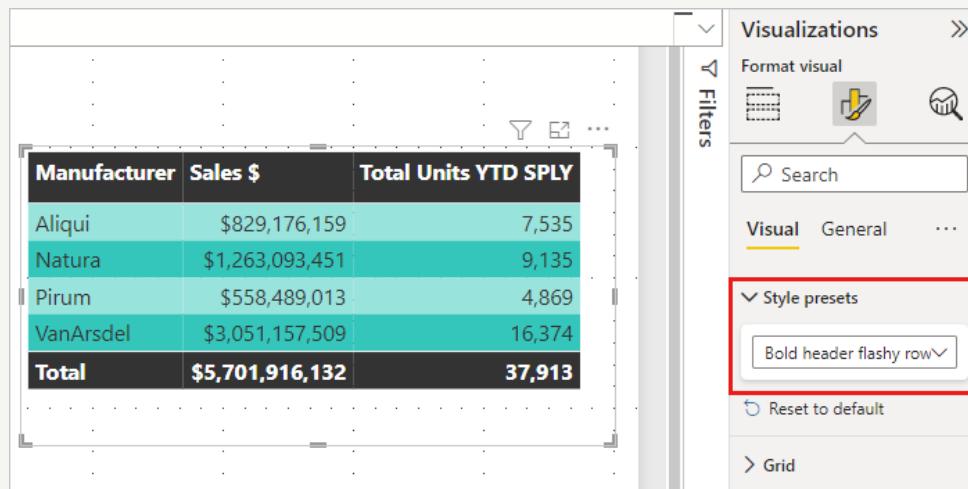
Style presets

Default

Reset to default

Grid

Select a style from the dropdown.



The screenshot shows a Microsoft Power BI interface. On the left is a table with the following data:

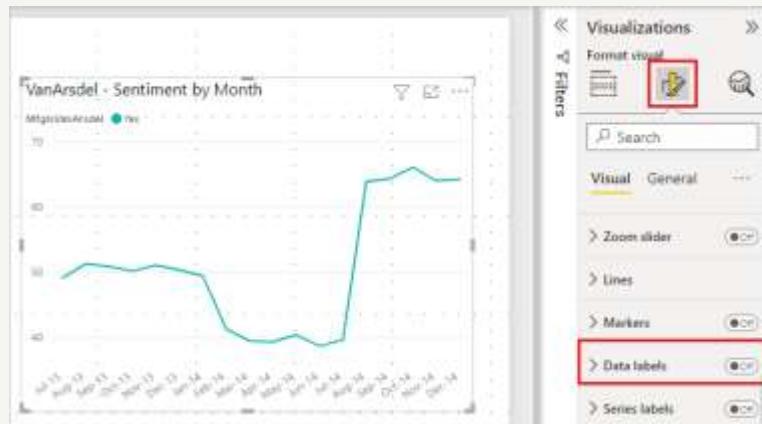
Manufacturer	Sales \$	Total Units YTD SPLY
Aliqui	\$829,176,159	7,535
Natura	\$1,263,093,451	9,135
Pirum	\$558,489,013	4,869
VanArsdel	\$3,051,157,509	16,374
<b>Total</b>	<b>\$5,701,916,132</b>	<b>37,913</b>

On the right, the 'Format visual' pane is open, showing the 'Visualizations' section and a 'Filters' section. The 'Style presets' dropdown menu is highlighted with a red box. It contains options like 'Bold header flashy row' (which is selected), 'Reset to default', and 'Grid'.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205\\_%20Creating%20Dashboard%20with%20Visualization%20Tool/Formate%20table](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205_%20Creating%20Dashboard%20with%20Visualization%20Tool/Formate%20table)

## Add data labels

Here is the *before* picture.



Here is the *after* picture.



# Customize data labels

## Add total labels



## Create Tables in Power BI

For example, imagine you're a personnel manager who has a table of Northwest Employees and another table of Southwest Employees. You want to combine the two tables into a single table called Western Region Employees.

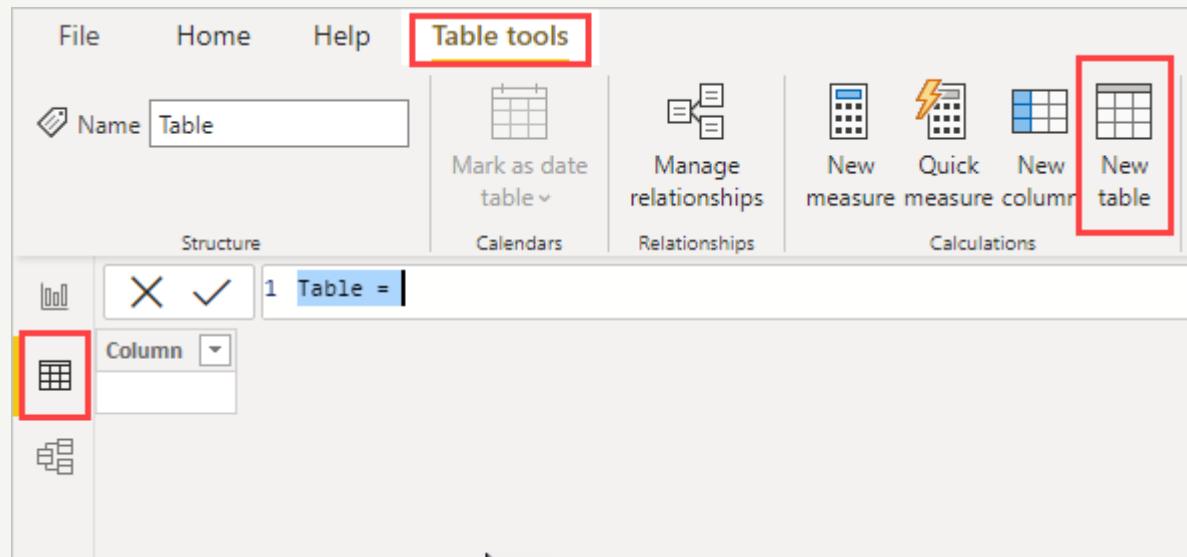
**Northwest Employees**

Employee	City	State	Tenure
Allen, Kerry	Eugene	OR	1
Baker, Cameron	Portland	OR	15
Morin, Max	Redmond	WA	10
Ramirez, Riley	Portland	OR	3
Rocha, Kim	Redmond	WA	15
Smith, Avery	Redmond	WA	15

**Southwest Employees**

Employee	City	State	Tenure
Connors, Morgan	San Diego	CA	10
Irwin, Jesse	Phoenix	AZ	3
Nguyen, Rory	Los Angeles	CA	3
Torres, Devon	Los Angeles	CA	2

In Report View, Data View, or Model View of Power BI Desktop, in the Calculations group select New table. It's a bit easier to do in Table tools in the Data View, because then you can immediately see your new calculated table.



Enter the following formula in the formula bar:

**Western Region Employees = UNION('Northwest Employees', 'Southwest Employees')**

X ✓ 1 Western Region Employees = UNION('Northwest Employees', 'Southwest Employees')

Employee	City	State	Tenure
Allen, Kerry	Eugene	OR	1
Baker, Cameron	Portland	OR	15
Morin, Max	Redmond	WA	10
Ramirez, Riley	Portland	OR	3
Rocha, Kim	Redmond	WA	15
Smith, Avery	Redmond	WA	15
Connors, Morgan	San Diego	CA	10
Irwin, Jesse	Phoenix	AZ	3
Nguyen, Rory	Los Angeles	CA	3
Torres, Devon	Los Angeles	CA	2

Fields >

Search

Northwest Employees

Southwest Employees

Western Region Employees

City

Employee

State

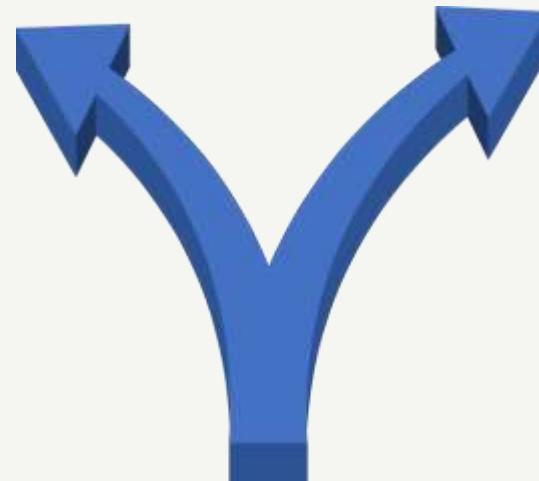
Tenure

## Data Analysis Expressions DAX

### Types of DAX Calculations

**Calculated Columns**

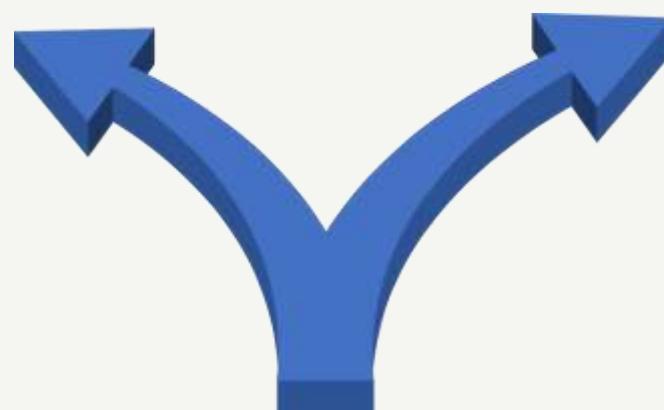
**Calculated Measures**





When you want to calculate the percentages or ratios, or you need complex aggregations to be created in the column. To accomplish this, you need:

**Measure name**



**At least one function or expression**

Example: ColumnName = Function(TableName[ColumnName])

## Data Analysis Expressions DAX

Aggregation

Counting

Logical

Text

Date

Filter

## Using DAX: Aggregation Functions

SUM

MeasureName = SUM(TableName[ColumnName])

AVERAGE

MeasureName = AVERAGE(TableName[ColumnName])

MIN

MeasureName = MIN(TableName[ColumnName])

MAX

MeasureName = MAX(TableName[ColumnName])

SUMX

MeasureName =  
SUMX(TableName,[ColumnName1]\*[ColumnName1])

## Using DAX: Counting Functions

COUNT

MeasureName = COUNT(TableName[ColumnName])

COUNTA

MeasureName = COUNTA(TableName[ColumnName])

COUNTBLANK

MeasureName = COUNTBLANK(TableName[ColumnName])

COUNTROWS

MeasureName = COUNTROWS(TableName)

DISTINCTCOUNT

MeasureName = DISTINCTCOUNT(TableName[ColumnName])

[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205\\_%20Creating%20Dashboard%20with%20Visualization%20Tool/Dax](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205_%20Creating%20Dashboard%20with%20Visualization%20Tool/Dax)

## Using DAX: Date Functions

DATE = DATE(2017,10,28)

HOUR = HOUR([ColumnName])

NOW = NOW()

EOMONTH = EOMONTH("3/10/2017",1)

WEEKDAY = WEEKDAY("3/10/2017",1)

[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205\\_%20Creating%20Dashboard%20with%20Visualization%20Tool/Dax](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205_%20Creating%20Dashboard%20with%20Visualization%20Tool/Dax)

## Using DAX: Filter Functions

- The filter functions are the most powerful DAX functions.
- The lookup functions operate using tables and relationships, like a database.
- To create dynamic calculations, the filtering functions let you manipulate data context.

CALCULATE

```
Books Revenue = CALCULATE(SUM([Revenue]),'Category Sales Report'[Category]="Books")
```

RELATED

```
Cost = RELATED('Subcategory Master'[Cost])
```

FILTER

```
High Revenue = FILTER('Category Sales Report','Category Sales Report'[Revenue]>10000)
```

[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205\\_%20Creating%20Dashboard%20with%20Visualization%20Tool/Dax](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205_%20Creating%20Dashboard%20with%20Visualization%20Tool/Dax)



# Text Functions

CONCATENATE

= CONCATENATE([ColumnName], "Value")

REPLACE

= REPLACE('New Products'[Product Code],1,2,"OB")

SEARCH

= SEARCH("com",[Subcategory],1,0)

UPPER

= UPPER([ColumnName])

FIXED

FIXED([Revenue Forecast],2,0)



## Creating IF OR and IF AND functions

AND

= AND([Country] = "USA", [Medal] = "Gold")

OR

= OR([Country] = "USA", [Medal] = "Gold")

NOT

= NOT([Country] = "USA")

IF

= IF([Country] = "USA", "American", "Other")

IFERROR

= IFERROR (5/0, "Div by zero")

[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205\\_%20Creating%20Dashboard%20with%20Visualization%20Tool>If%20Or%20and](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205_%20Creating%20Dashboard%20with%20Visualization%20Tool>If%20Or%20and)

# Overview of M Language

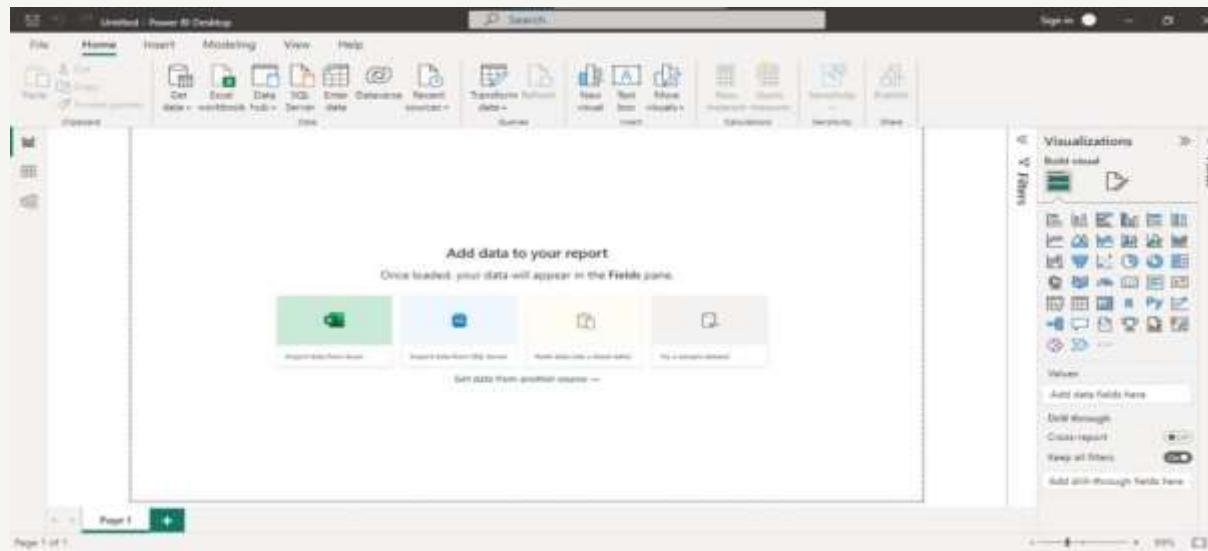
[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205\\_%20Creating%20Dashboard%20with%20Visualization%20Tool/Mlanguage](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205_%20Creating%20Dashboard%20with%20Visualization%20Tool/Mlanguage)



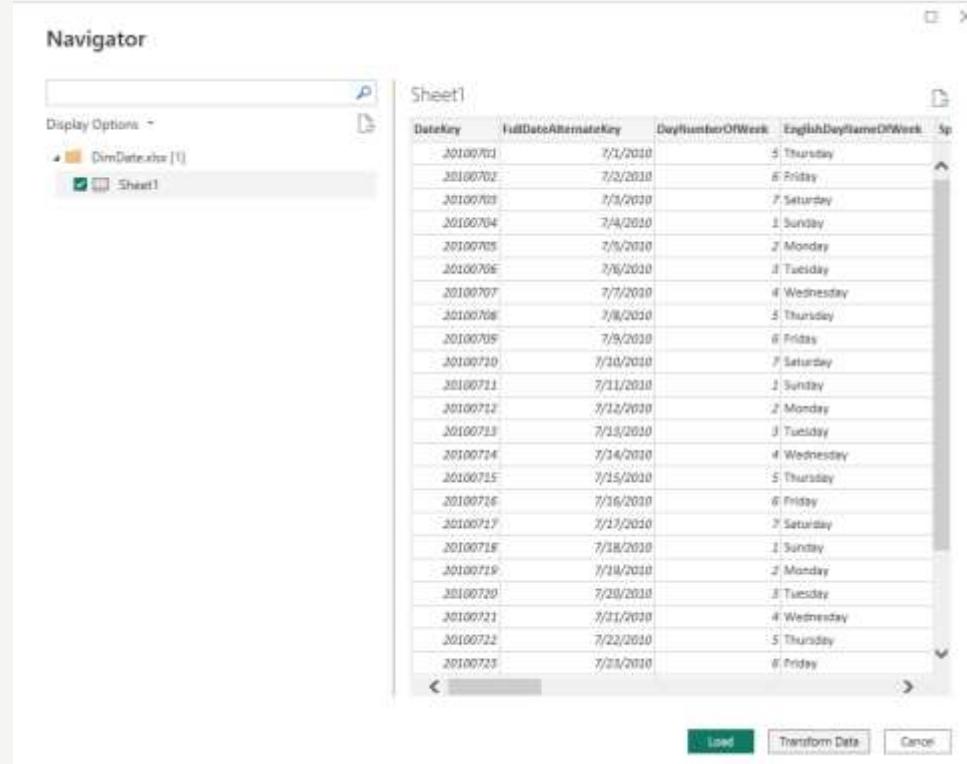
## What are the main differences between DAX and M Language?

## Creating Power Query M formula language queries :-

**First of all, let's just open the Power Query Editor if your Power BI Desktop appears as shown below-**



**Go to Get Data tab or Recent sources, select the Dataset and click on “Transform Data”.**

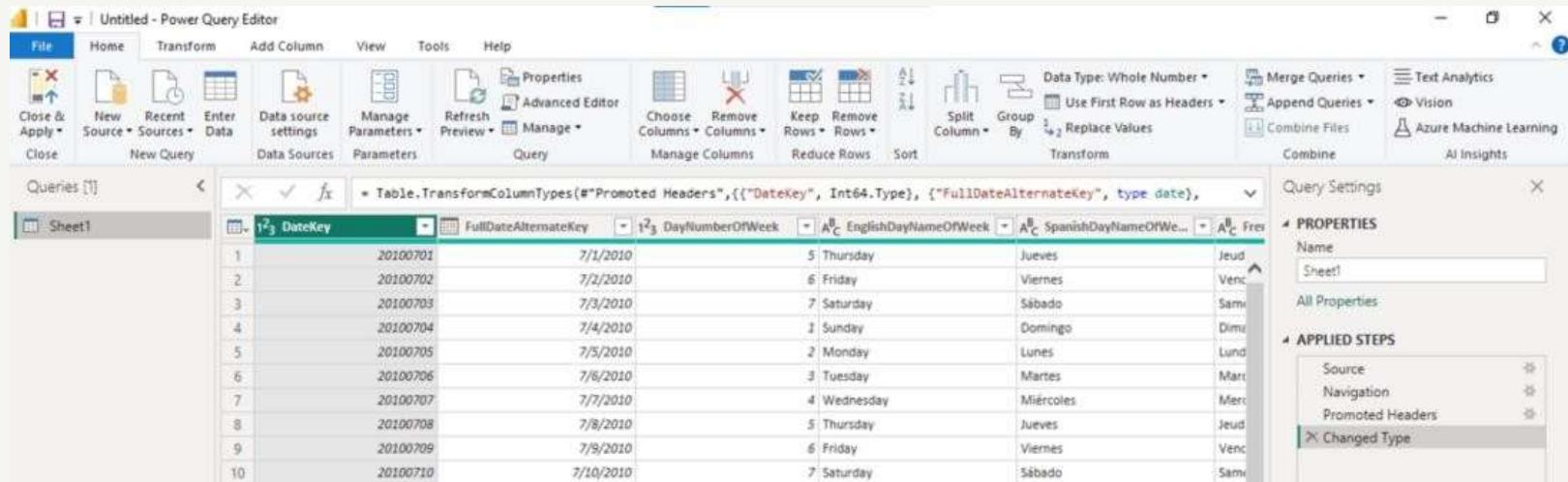


The screenshot shows the Microsoft Power BI 'Get Data' interface. On the left, the 'Navigator' pane displays 'DimDate.xlsx [1]' and 'Sheet1'. The main area, titled 'Sheet1', shows a preview of data from July 2010. The columns are: DateKey, FullDateAlternateKey, DayNumberOfWeek, EnglishDayNameOfWeek, and Sp. The data is as follows:

DateKey	FullDateAlternateKey	DayNumberOfWeek	EnglishDayNameOfWeek	Sp
20100701	7/1/2010	5	Thursday	
20100702	7/2/2010	6	Friday	
20100703	7/3/2010	7	Saturday	
20100704	7/4/2010	1	Sunday	
20100705	7/5/2010	2	Monday	
20100706	7/6/2010	3	Tuesday	
20100707	7/7/2010	4	Wednesday	
20100708	7/8/2010	5	Thursday	
20100709	7/9/2010	6	Friday	
20100710	7/10/2010	7	Saturday	
20100711	7/11/2010	1	Sunday	
20100712	7/12/2010	2	Monday	
20100713	7/13/2010	3	Tuesday	
20100714	7/14/2010	4	Wednesday	
20100715	7/15/2010	5	Thursday	
20100716	7/16/2010	6	Friday	
20100717	7/17/2010	7	Saturday	
20100718	7/18/2010	1	Sunday	
20100719	7/19/2010	2	Monday	
20100720	7/20/2010	3	Tuesday	
20100721	7/21/2010	4	Wednesday	
20100722	7/22/2010	5	Thursday	
20100723	7/23/2010	6	Friday	

At the bottom right, there are buttons for 'Load', 'Transform Data', and 'Cancel'.

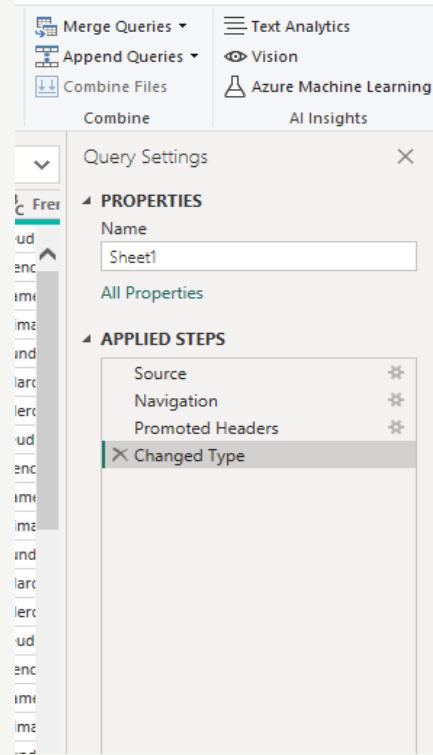
It will open a Power Query Window.



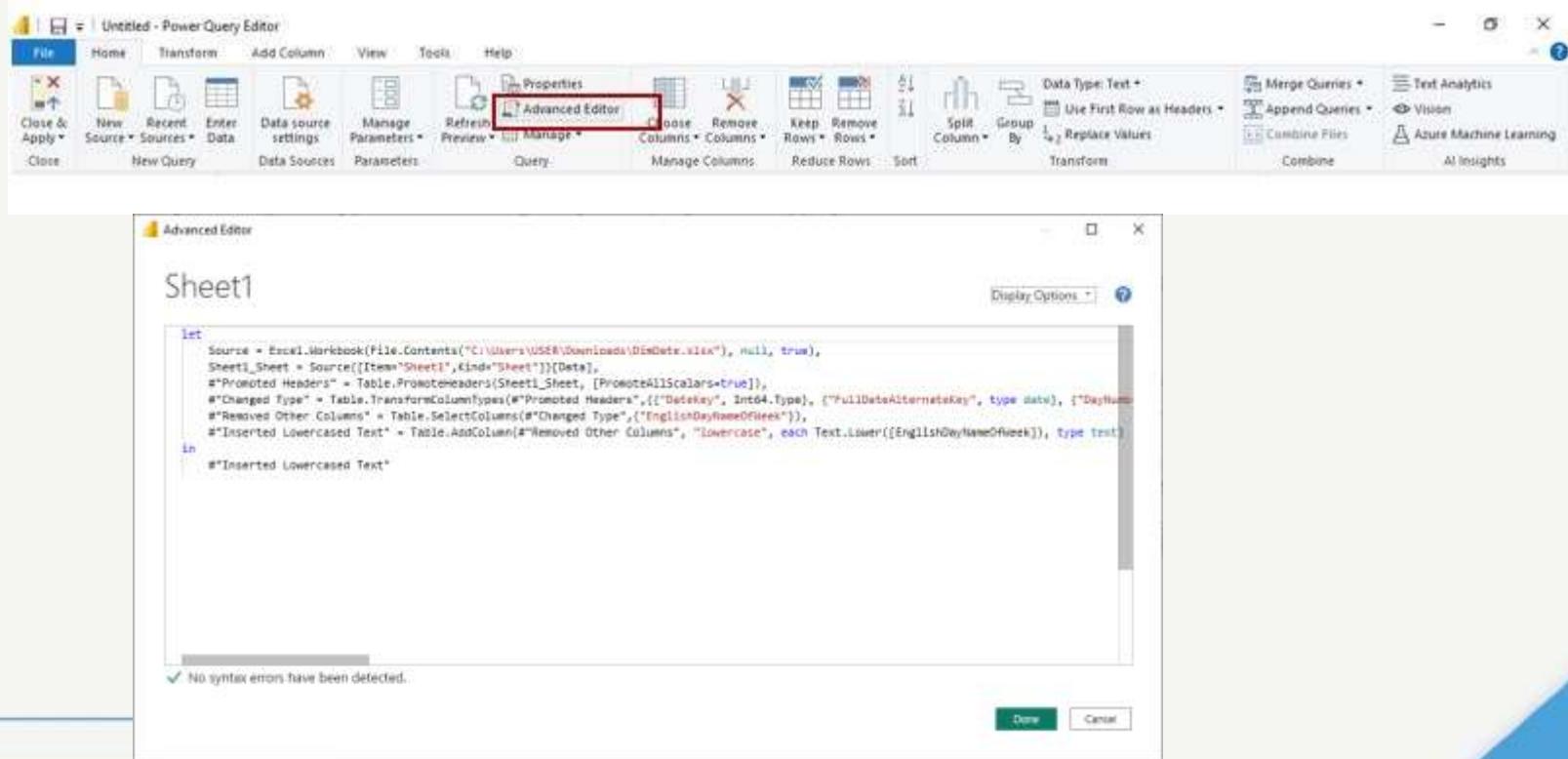
The screenshot shows the Microsoft Power Query Editor interface. The ribbon menu at the top includes File, Home, Transform, Add Column, View, Tools, and Help. The Home tab is selected. The toolbar below the ribbon contains various icons for managing queries, such as Close & Apply, New Source, Refresh, Manage Parameters, and Data Type. The main workspace displays a table with 10 rows of data, labeled 'Sheet1'. The columns are: DateKey, FullDateAlternateKey, DayNumberOfWeek, EnglishDayNameOfWeek, SpanishDayNameOfWeek, and FrenchDayName. The data shows dates from July 1, 2010, to July 10, 2010, with corresponding day names in English, Spanish, and French. To the right of the table, the 'Query Settings' pane is open, showing the 'Name' field set to 'Sheet1'. The 'APPLIED STEPS' pane lists the transformation steps: 'Source', 'Navigation', 'Promoted Headers', and 'Changed Type'. The 'Changed Type' step is highlighted with a red border.

DateKey	FullDateAlternateKey	DayNumberOfWeek	EnglishDayNameOfWeek	SpanishDayNameOfWeek	FrenchDayName
20100701	7/1/2010	5	Thursday	Jueves	Jeudi
20100702	7/2/2010	6	Friday	Viernes	Vendredi
20100703	7/3/2010	7	Saturday	Sábado	Samedi
20100704	7/4/2010	1	Sunday	Domingo	Dimanche
20100705	7/5/2010	2	Monday	Lunes	Lundi
20100706	7/6/2010	3	Tuesday	Martes	Mardi
20100707	7/7/2010	4	Wednesday	Miercoles	Mardi
20100708	7/8/2010	5	Thursday	Jueves	Jeudi
20100709	7/9/2010	6	Friday	Viernes	Vendredi
20100710	7/10/2010	7	Saturday	Sábado	Samedi

**Power query is where the M language is written, and you can see on the right that these four stages were generated automatically.**



## Advanced query formed can be seen in Advance editor tab-



The screenshot shows the Microsoft Power Query Editor interface. The ribbon at the top has tabs like File, Home, Transform, Add Column, View, Tools, and Help. The 'Transform' tab is currently selected. A toolbar below the ribbon contains various icons for operations such as Close & Apply, New Source, Refresh, Manage Parameters, and Advanced Editor. The 'Advanced Editor' button is highlighted with a red box. The main area is titled 'Sheet1' and displays the M code for the query:

```
let
    Source = Excel.Workbook(File.Contents("C:\Users\USEA\Downloads\DimDistr.xlsx"), null, true),
    Sheet1_Sheet = Source[[Item="Sheet1",Kind="Sheet"]]{Data},
    #"Promoted Headers" = Table.PromoteHeaders(Sheet1_Sheet, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Datekey", Int64.Type}, {"FullDateAlternateKey", type date}, {"DayNum", Int64.Type}, {"Removed Other Columns" = Table.SelectColumns(#"Changed Type",{"EnglishDayNameOfWeek"})}},
    #"Inserted Lowercased Text" = Table.AddColumn(#"Removed Other Columns", "lowercase", each Text.Lower([EnglishDayNameOfWeek]), type text)
in
    #"Inserted Lowercased Text"
```

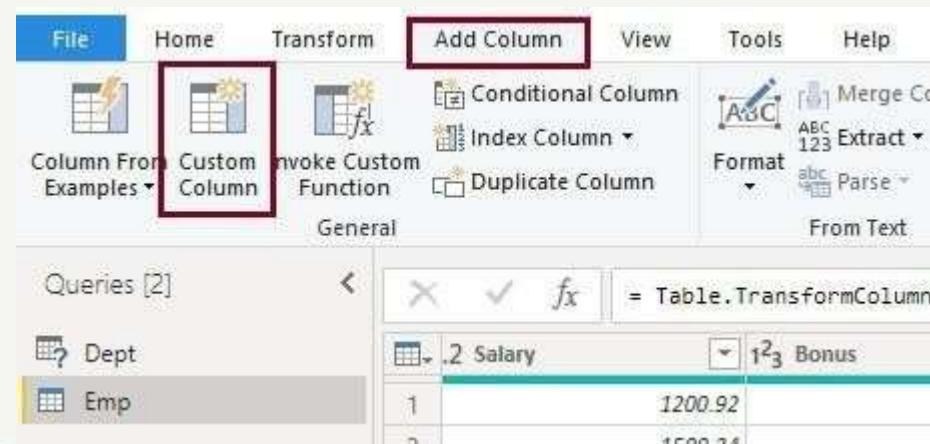
At the bottom of the Advanced Editor window, there is a message: '✓ No syntax errors have been detected.' and two buttons: 'Done' and 'Cancel'.

## How to Add a Custom Column using M Language?

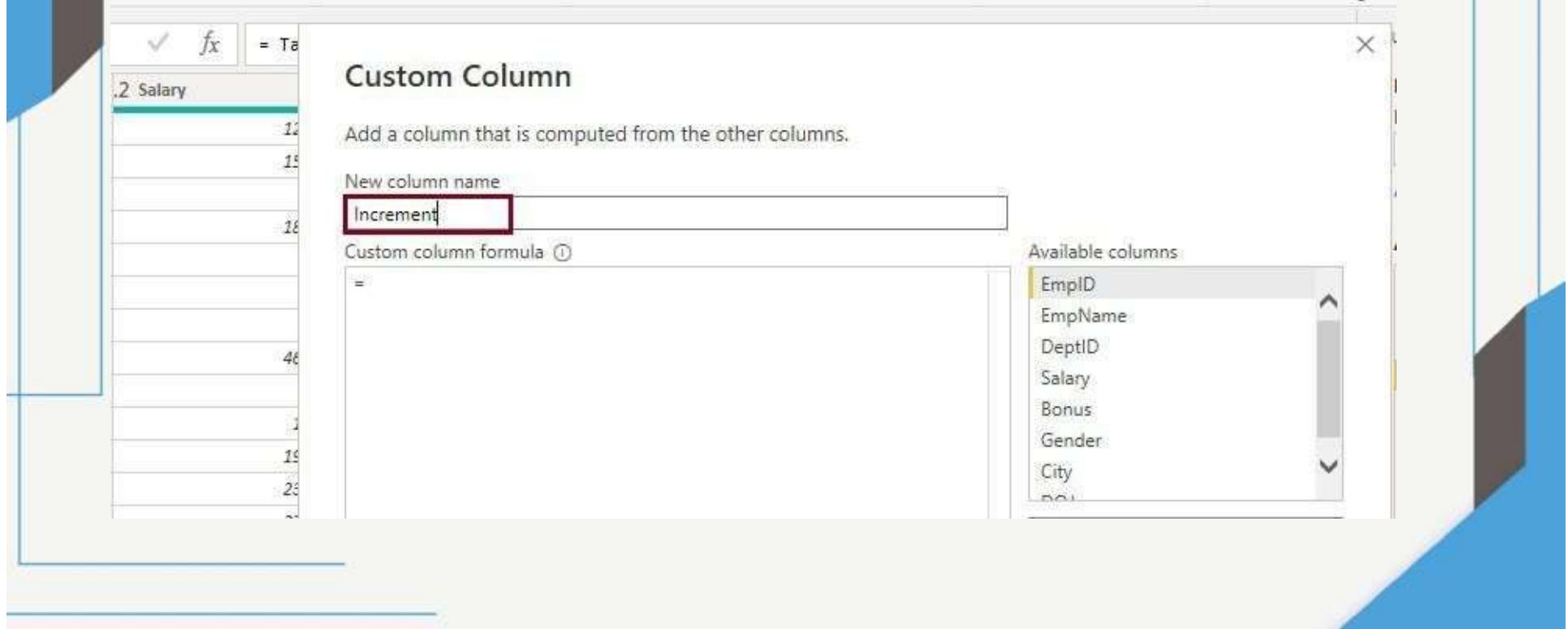
**Open Power BI desktop >> Get Data >> Choose any data sources (Ex: Excel workbook) >> choose required file >> select required tables >> load.**

**Then**

**Step 1: Select a custom column from the query editor. Go to Add new column >> select custom column as shown in the picture.**



**Step 2: Give your new column a name.** Here we wanted to create a new Increment column on the employee table.



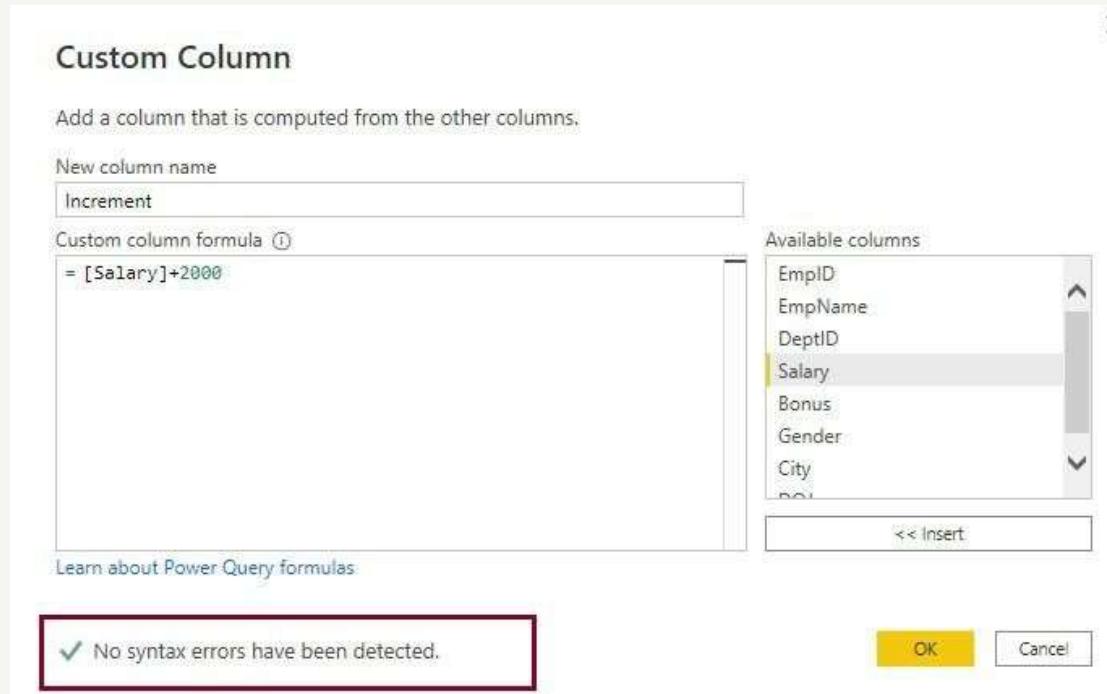
The screenshot shows a spreadsheet application with a 'Custom Column' dialog box open. On the left, there is a preview of a table with a single visible row containing salary data. The main area of the dialog has a title 'Custom Column' and a sub-instruction 'Add a column that is computed from the other columns.' Below this, a 'New column name' input field contains the text 'Increment', which is highlighted with a red rectangle. A 'Custom column formula' input field below it contains the text '='. To the right of these fields is a vertical list of 'Available columns' with the following items: EmpID, EmpName, DeptID, Salary, Bonus, Gender, and City. The 'Salary' item is at the top of this list.

### Step 3: Create a formula for the column.

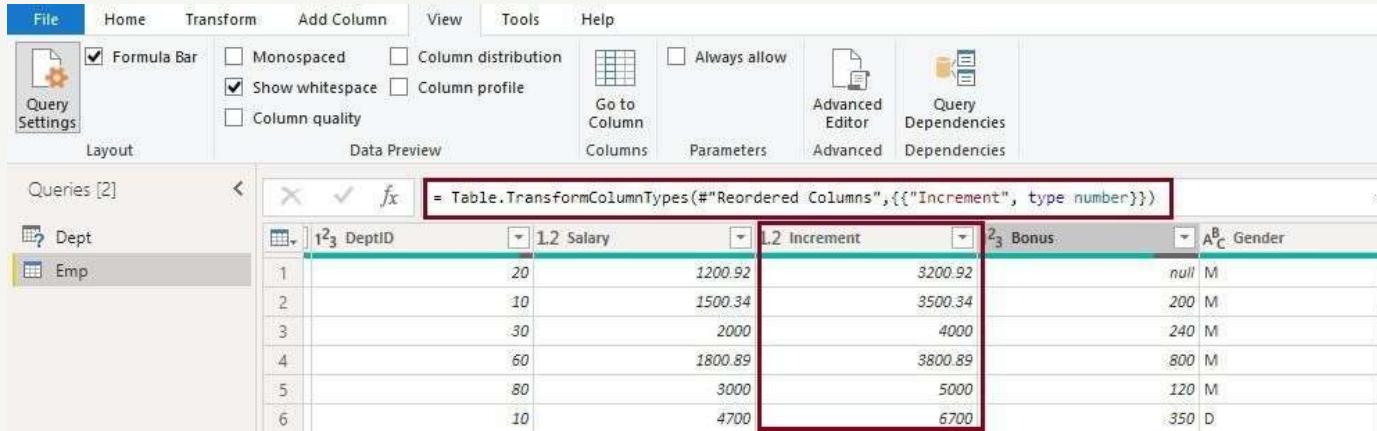
For this, we selected the “salary” column from the available columns option, because we wanted to add an Increment column on the base of the salary of the employee. After selecting the required column from the available columns option select ” Insert “. then go to the custom column formula box and try to write the formula (Ex: [salary] + 2000).



## Step 4: Check that there are no syntax errors in your formula then press OK.



**Now we created a new column using M language as shown in the picture.**



The screenshot shows the Microsoft Power Query Editor interface. The ribbon at the top includes File, Home, Transform, Add Column, View, Tools, and Help. The 'Query Settings' icon is selected in the left pane. The 'Advanced Editor' button is highlighted. The formula bar displays the M code: `= Table.TransformColumnTypes(#"Reordered Columns",{{"Increment", type number}})`. The 'Data Preview' pane shows two tables: 'Dept' and 'Emp'. The 'Emp' table has columns: DeptID, Salary, Increment, Bonus, and Gender. The 'Increment' column is highlighted with a red border. The data in the 'Increment' column is as follows:

	DeptID	Salary	Increment	Bonus	Gender
1	20	1200.92	3200.92	null	M
2	10	1500.34	3500.34	200	M
3	30	2000	4000	240	M
4	60	1800.89	3800.89	800	M
5	80	3000	5000	120	M
6	10	4700	6700	350	D

## How to Change Background in Power BI Map

Add map visuals and customize them

After preparing and loading the dataset, simply drag-and-drop and customize the preferred map in Power BI.



Go to the Visualizations → Format visual. Here you will find 2 options to personalize the map: Visuals and General settings.

- The Visual settings are different depending on the type of map you choose.
- The General settings include the customization options for titles, effects, header icons, showing/hiding tooltips, using ALT text, etc. These general properties are the same for all types of Power BI maps.



**TOPS TECHNOLOGIES**

Training | Outsourcing | Placement | Study Abroad

### Visualizations »

Format visual



Search

Visual General ...

› Properties

› Title On

› Effects

› Header icons On

› Tooltips On

› Alt text

## How to Create a Map in Power BI

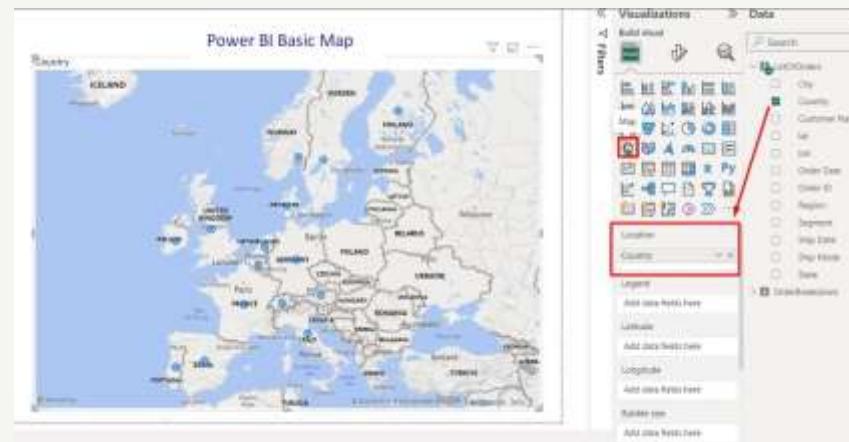
A variety of map types are available for you to explore, including:

- Basic Map
- Filled Map
- Shape Map
- ArcGIS Maps

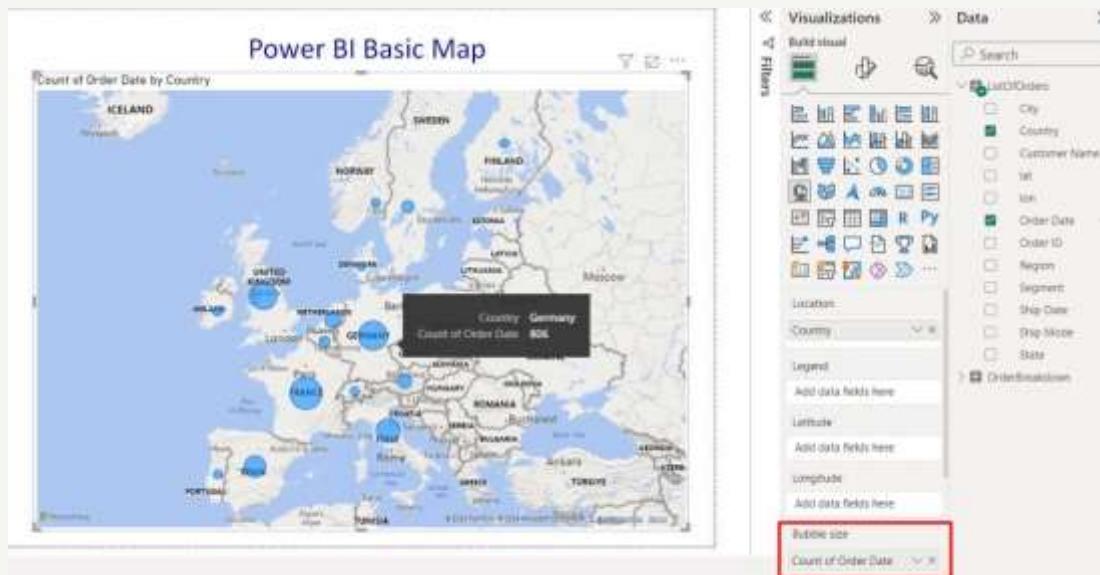
# Basic Map

## How to create Power BI basic maps

- After loading a map dataset inside your Power BI, you can see the **Visualizations** pane on the right side of the canvas.
- Here, you'll find the **icon of Map (basic)**. Drag and drop the basic map icon inside the Power BI report. You can expand the map for a better view and analytics.
- Next, select the country or city and drag it inside the **Location** tab. Within seconds, you can see the bubbles identifying the European countries.



In the Power BI basic map, you can also show the bubble size of the geographical locations. For instance, drag the Order ID in the bubble Size tab to show how many orders the shop gets from each country.

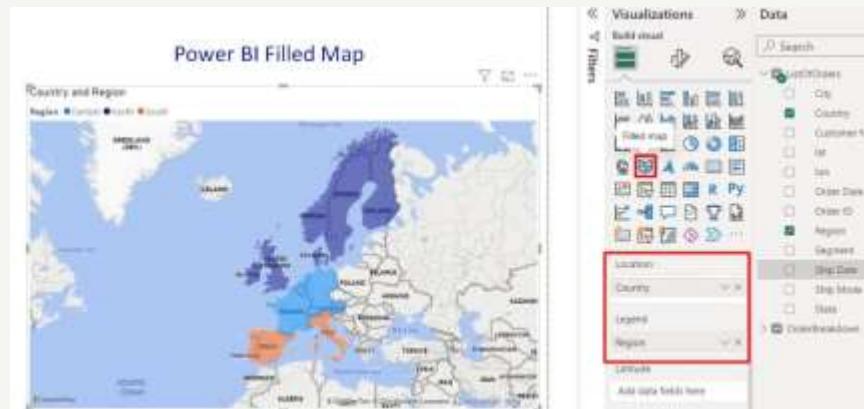


[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205\\_%20Creating%20Dashboard%20with%20Visualization%20Tool/Map%20data](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205_%20Creating%20Dashboard%20with%20Visualization%20Tool/Map%20data)

## Filled maps

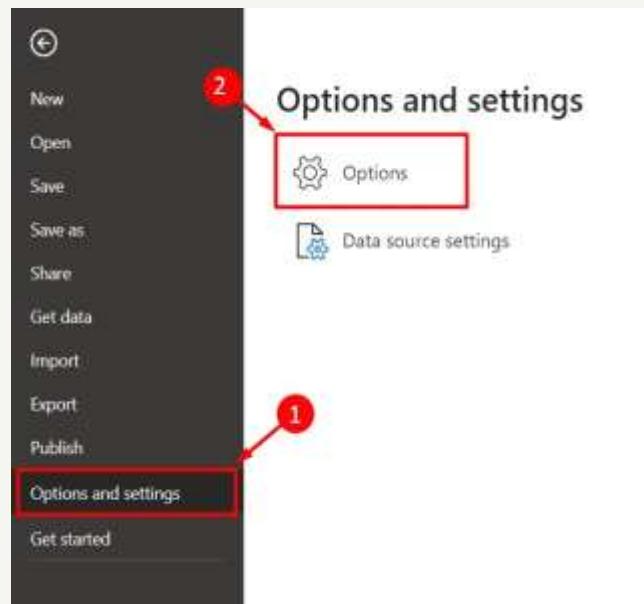
### How to create Power BI filled maps

- Go to the Visualizations pane. Next, locate the filled map icon and drag it onto your report canvas.
- To display the geographical map, assign Location (e.g. country) and data fields such as Legend (Region). The legend in Power BI Filled Map provides a key to interpret the colors representing data categories or ranges.
- Here the Central, North, and South regions are categorized by filling 3 different colors.

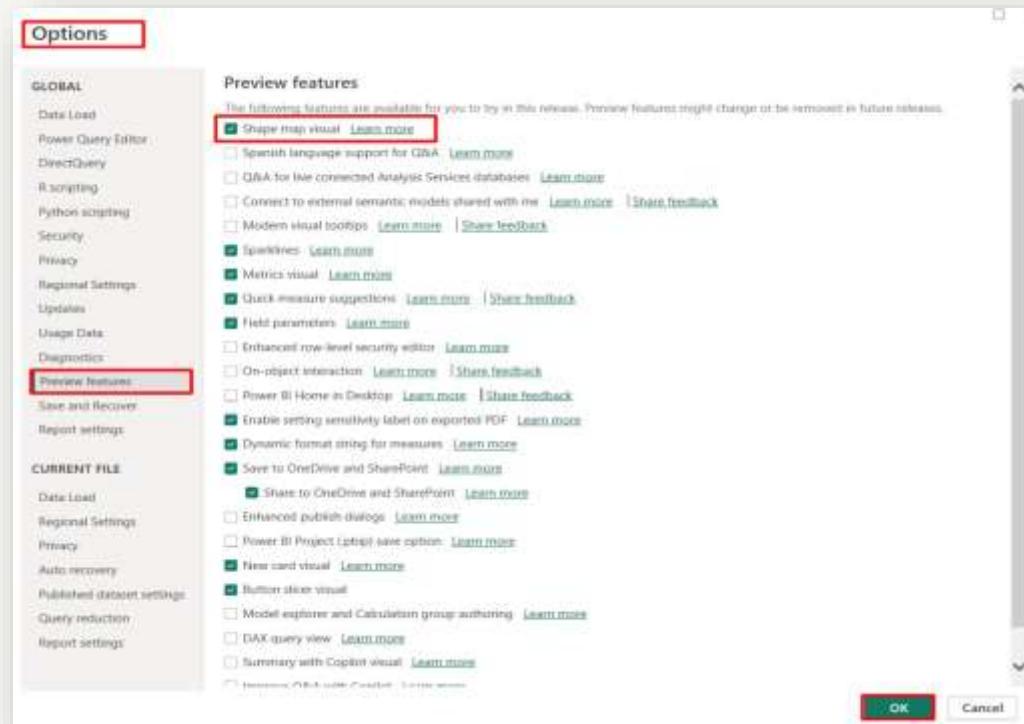


## How to create Power BI shape maps

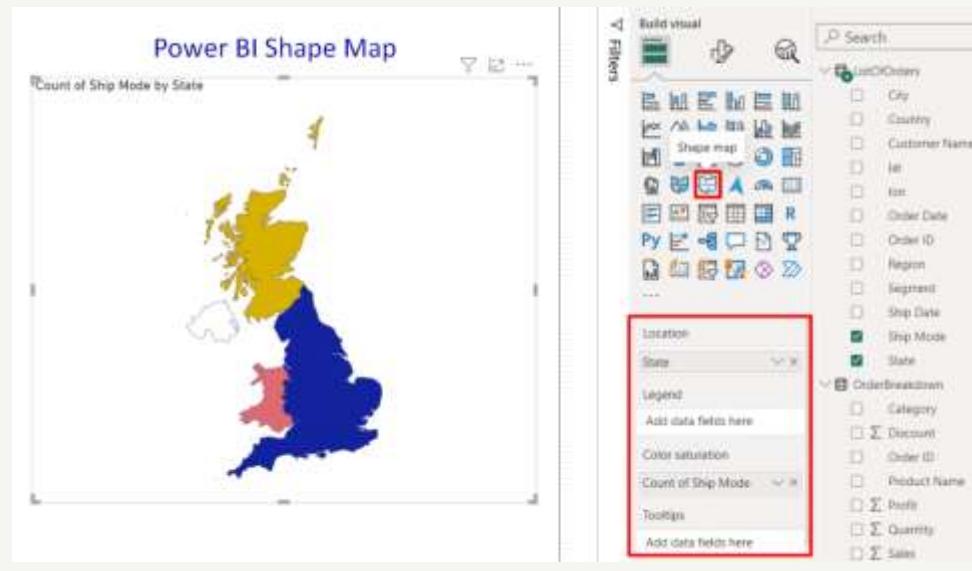
- To create a Shape map in Power BI, there's an initial step since it's currently a preview feature. First, you'll need to enable it.
- Head over to File, then Options and Settings, followed by Options.



- Under Preview features, find the Shape map visual checkbox and tick it. Click OK and don't forget to restart Power BI Desktop for the changes to take effect.

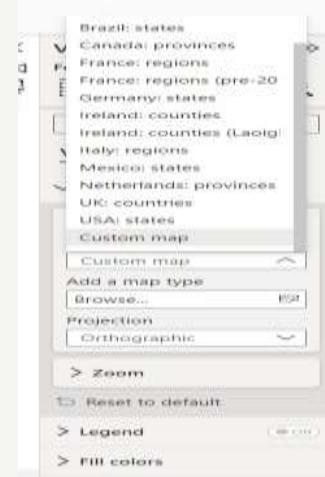


- Now you're ready to build your map! Open your report and navigate to the **Visualizations** pane. Look for the **Shape map** icon and select it.
- Then, drag your data field, the Ship Mode of the product, to the **Color saturation** field. Power BI will automatically color each region based on your data values, visually representing your information.

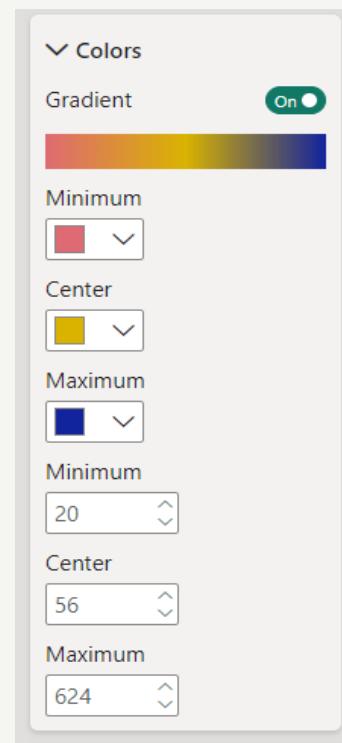


## Customize shape map in Power BI

- Go to the Visual → Map Setting → Map type
- Select the map type from the drop-down e.g. USA: states, Canada: provinces, etc
- If your desired map isn't in the drop-down, simply browse the JSON file of the custom map type.
- Choose the Projection type such as Orthographic, Equirectangular, and Mercator.
- This will help you to adjust the map of how the 3D Earth is flattened onto a 2D map to best represent data.



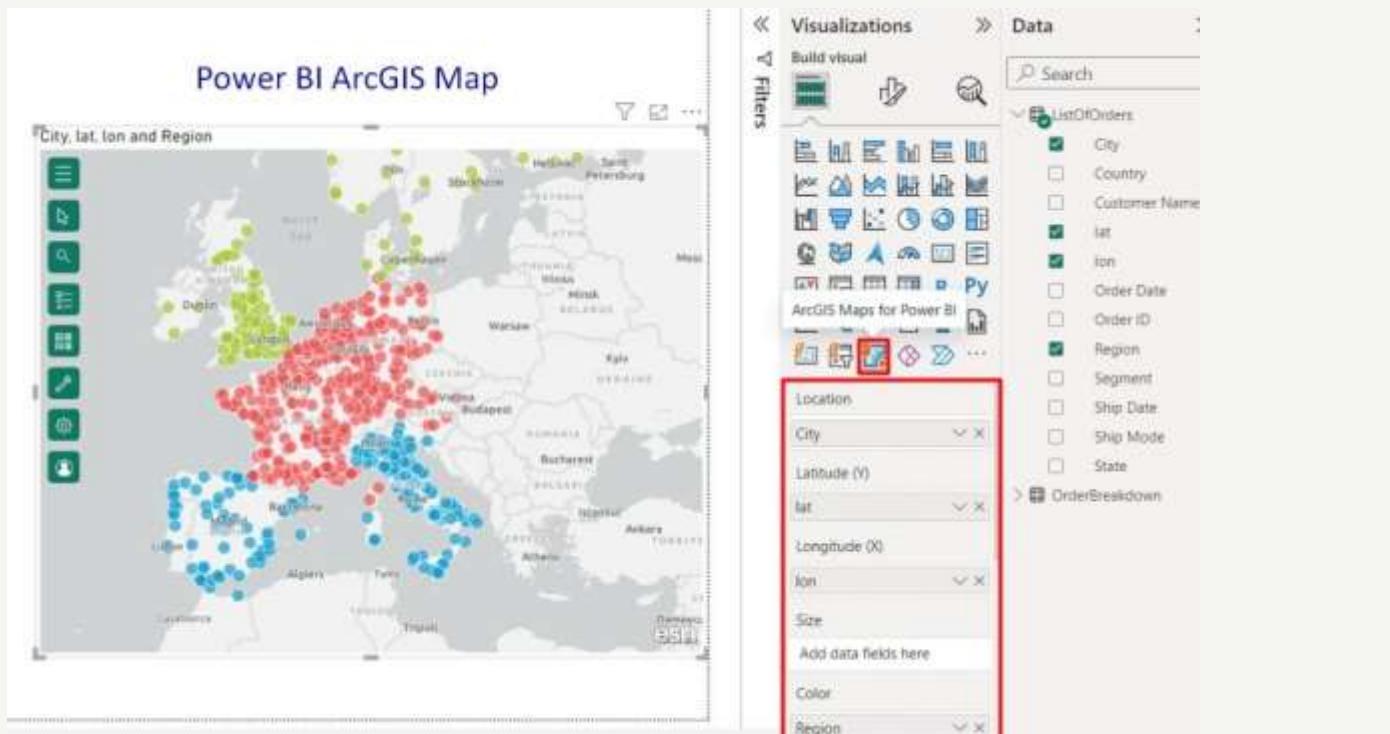
Additionally, you can customize the zoom and legend options for shape maps. You can also set general or gradient colors. Also, select the fill colors to choose the minimum, medium, and maximum values.



## ArcGIS maps

### How to make ArcGIS maps for Power BI

- Open an existing report or create a new one in Power BI Desktop. In the Visualizations pane on the right side, find the ArcGIS for Power BI icon and click it. An empty map template will appear on your report canvas.
- On the right side, there will be fields for Location and Latitude and Longitude depending on your data format.
- Drag and drop your location field into the Location field or enter the appropriate latitude and longitude fields. You can also categorize based on color by adding value in the Color field.
- Power BI will automatically populate the map with your data points with spatial data to visualize and gain advanced demographic insights.



[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205\\_%20Creating%20Dashboard%20with%20Visualization%20Tool/Map%20data](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205_%20Creating%20Dashboard%20with%20Visualization%20Tool/Map%20data)

## Subtotal & Total in Matrix

### Simple Matrix with grand total

Location	Online	Reseller	Store	Total
Asia	\$558,161,919.882	\$362,335,320.82	\$864,865,641.158	<b>\$1,785,362,881.86</b>
Europe	\$513,363,375.0855	\$337,908,291.1344	\$772,295,801.23	<b>\$1,623,567,467.4499</b>
North America	\$614,545,814.916	\$404,305,088.911	\$3,217,538,155.937	<b>\$4,236,389,059.764</b>



# Matrix with multiple columns and rows

## How to turn off grand total in Power BI Matrix

- You can easily turn the grand total off (and turn it on again) by using the '**Column subtotals**' and '**Row subtotals**' options in the **Format visual** tab.
- The label 'subtotals' here might not seem intuitive at first, but it makes sense when you consider the structure of the Matrix.

### Steps to turn off the grand total:

- Click the Matrix visual to select it.
- Click the **Format visual** tab in the **Visualizations** pane. You'll find the **Row subtotals** and **Column subtotal** sections.
- Toggle the button to **Off**, Depending on which grand total you want to turn off.
- Optionally, you can apply the setting individually for each row or column.



# TOPS TECHNOLOGIES

Training | Outsourcing | Placement | Study Abroad

Drill on [Columns](#)

Year	2011		2012	
	ProductCategory	SalesQuantity	SalesAmount	SalesQuantity
<b>Audio</b>				
Bluetooth Headphones	86067	\$3,773,927.4859	108882	
MP4&MP3	111448	\$15,951,269.3459	194770	
Recording Pen	55124	\$10,009,475.177	80620	
<b>Cameras and camcorders</b>				
Camcorders	735790	\$565,099,766.06	617692	
Cameras & Camcorders Accessories	266763	\$11,511,262.0934	293072	
Digital Cameras	788680	\$164,036,668.479	535729	
Digital SLR Cameras	793246	\$361,246,218.826	545705	
<b>Cell phones</b>				
Cell phones Accessories	1889140	\$19,432,285.6013	2670160	
Home & Office Phones	723929	\$21,977,458.3508	493327	
Smart phones & PDAs	694083	\$196,066,926.45	441862	
Touch Screen Phones	451952	\$126,370,920.68	322371	
<b>Computers</b>				
Computers Accessories	792800	\$25,102,979.6835	938331	
Desktops	671634	\$235,958,217.1285	460394	
Laptops	308841	\$207,603,669.5815	423364	
Monitors	273775	\$62,030,603.23	394980	
Printers, Scanners & Fax	789034	\$129,652,592.36	523708	
Projectors & Screens	610694	\$486,121,934.59	533417	
<b>Music, Movies and Audio Books</b>				
Movie DVD	650102	\$74,975,760.8291	463809	
<b>TV and Video</b>				
Car Video	335364	\$120,587,054.45	260501	
Home Theater System	511759	\$232,837,702.159	531275	
Televisions	198049	\$61,693,262.7022	237443	
VCD & DVD	113654	\$11,353,334.95	118967	
<b>Total</b>	<b>11851928</b>	<b>\$3,144,393,292.1311</b>	<b>11270399</b>	

«» **Visualizations** »»

Format visual

Filters

Search

Visual General ...

Row headers

Column subtotals  **Off**

Row subtotals  **On**

Apply settings to

Per row level  **On**

Row level  **ProductSubcategory**

Rows

Show subtotal  **Off**

Subtotal label

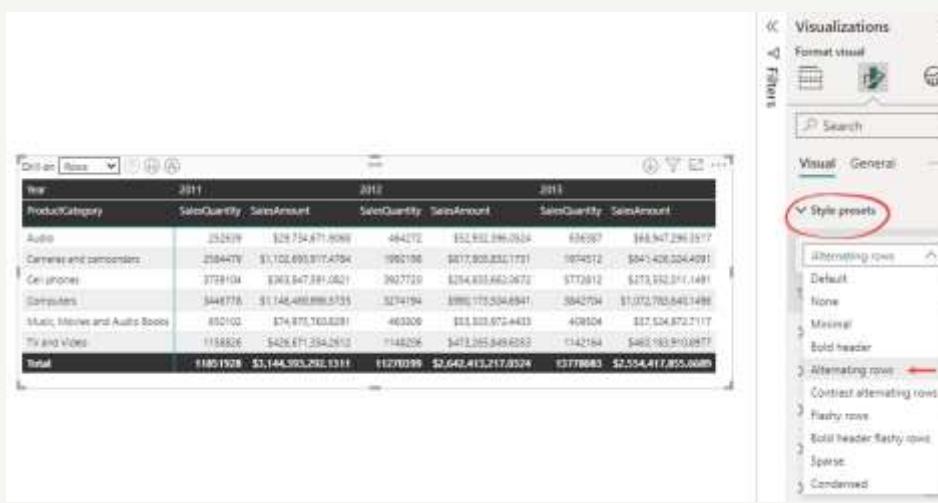
Position  **Top**

## How to format Matrix in Power BI

Formatting a Matrix in Power BI can significantly enhance its appearance by making it more readable, engaging, and informative.

Here's how:

1. Click the Matrix to select it.
2. Click the Format visual tab in the Visualization pane.
3. Configure the formatting options as you want.

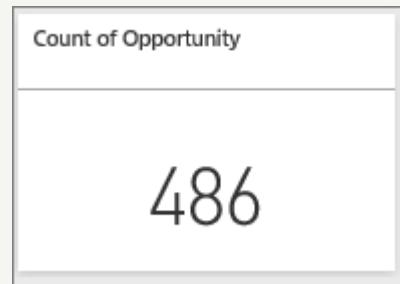


The screenshot shows a Power BI report with a Matrix visualization. The Matrix has three columns labeled 'Year' (2011, 2012, 2013), 'ProductCategory' (Auto, Cameras and camcorders, Cell phones, Computers, Music, Movies and Audio Books, TV and Video), and two rows for SalesQuantity and SalesAmount. The data includes values like 25,217 for Auto in 2011 and \$1,145,993,292 for Total sales in 2011. To the right, the 'Format visual' pane is open, specifically the 'Style' tab under 'Visualizations'. A red circle highlights the 'Style presets' section, which contains various options like 'Default', 'None', 'Minimal', 'Bold header', and 'Alternating rows'. Under 'Alternating rows', another red circle highlights the '2 Alternating rows' option.

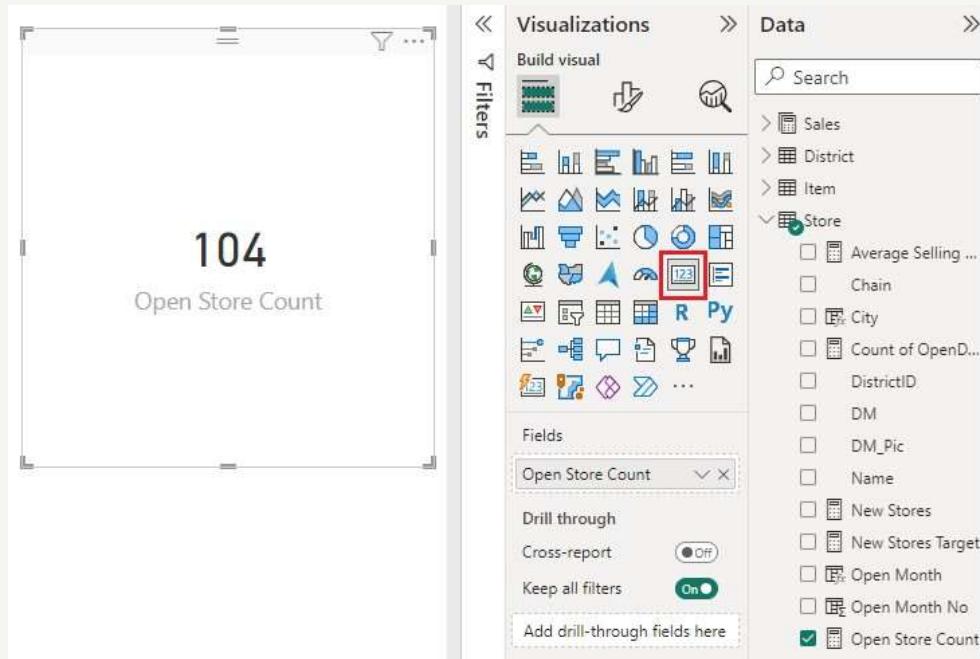
Year	2011	2012	2013			
ProductCategory	SalesQuantity	SalesAmount	SalesQuantity	SalesAmount	SalesQuantity	SalesAmount
Auto	25,217	\$1,145,993,292	48,472	\$5,931,296,052	63,937	\$8,367,291,577
Cameras and camcorders	218,447	\$7,102,690,971	4754	\$9,821,803,832	1,731	\$641,408,524,401
Cell phones	372,914	\$363,647,391,021	392,729	\$254,833,662,267	372,812	\$271,352,311,149
Computers	544,778	\$1,148,488,888,875	527,194	\$980,175,534,894	582,704	\$1,073,783,845,749
Music, Movies and Audio Books	852,102	\$74,875,703,628	463,206	\$31,323,672,443	40,854	\$27,524,872,217
TV and Video	1,188,826	\$426,671,332,262	714,826	\$413,365,849,633	13,421	\$482,183,910,697
Total	1,185,192	\$3,144,993,292,131	1,127,809	\$2,642,413,217,852	1,377,988	\$2,554,417,855,868

## Create card visualizations

A single number, such as total sales, market share year over year, or total opportunities, is sometimes the most important thing you want to track. A type of visualization in Power BI called a *card* might be the best way to view that number. As with almost all of the native Power BI visualizations, you can create cards with the report editor or Q&A. Here's an example of a card visualization:



## Create a card using the report editor



The screenshot displays the Power BI Report Editor interface. On the left, a card visual shows the value "104" and the caption "Open Store Count". To the right of the card is the "Visualizations" pane, which contains a grid of icons for different chart types. One icon, a small square with a number inside, is highlighted with a red box. Below the grid is a section titled "Fields" containing a dropdown menu set to "Open Store Count". The "Data" pane on the right shows a hierarchical structure of fields under "Store", with "Open Store Count" checked. Other fields listed include Sales, District, Item, Average Selling ..., Chain, City, Count of OpenD..., DistrictID, DM, DM\_Pic, Name, New Stores, New Stores Target, Open Month, Open Month No, and another entry for Open Store Count.

Visualizations

Build visual

Filters

Open Store Count

Data

Search

Sales

District

Item

Store

Average Selling ...

Chain

City

Count of OpenD...

DistrictID

DM

DM\_Pic

Name

New Stores

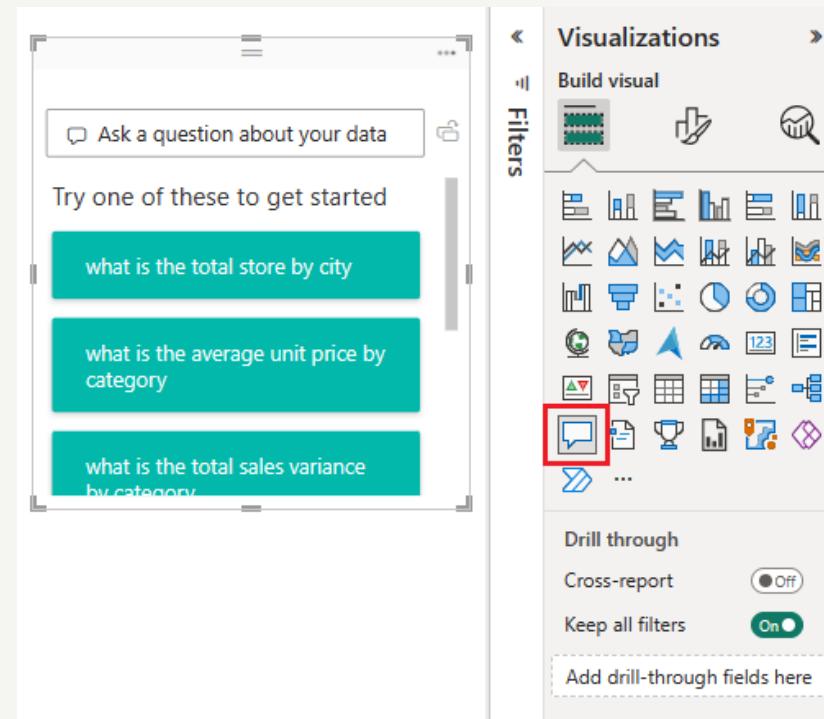
New Stores Target

Open Month

Open Month No

Open Store Count

## Create a card from the Q&A question box



The screenshot shows the Power BI desktop interface. On the left, there's a 'Q&A' card with the heading 'Ask a question about your data'. Below it, under 'Try one of these to get started', are three teal-colored cards with questions:

- what is the total store by city
- what is the average unit price by category
- what is the total sales variance by category

To the right of the Q&A card is the 'Visualizations' pane. It includes sections for 'Build visual' (with icons for matrix, bar chart, line chart, etc.), 'Filters' (with a dropdown arrow), and 'Drill through'. Under 'Drill through', there are options for 'Cross-report' (radio button set to 'Off') and 'Keep all filters' (radio button set to 'On'). A red box highlights the icon for a 'Card' visualization, which is the fourth icon in the first row of the 'Build visual' section.

# Slicers in Power BI

## When to use a slicer:-

Slicers are a great choice when you want to:

- Display commonly used or important filters on the report canvas for easier access.
- Make it easier to see the current filtered state without having to open a drop-down list.
- Filter by columns that are unneeded and hidden in the data tables.
- Create more focused reports by putting slicers next to important visuals.

Power BI slicers don't support:

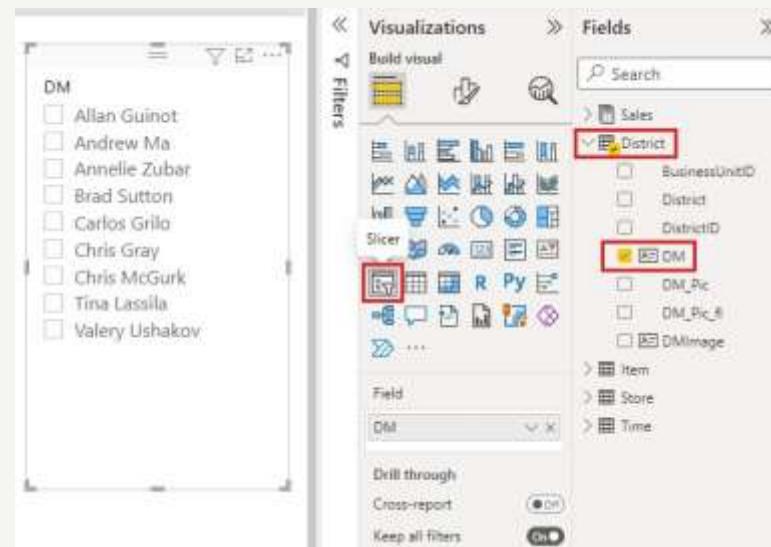
- Input fields
- Drill-down options



## Create a slicer

- 1. Open Power BI Desktop, and from the menu bar, select File > Open report.**
- 2. Browse to the Retail Analysis Sample PBIX.pbix file, then select Open.**
- 3. On the left pane, select the Report icon to open the file in report view.**
- 4. On the Overview page, with nothing selected on the report canvas, select the Slicer icon in the Visualizations pane to create a new slicer.**

The new slicer is now populated with a list of district manager names and their selection boxes.

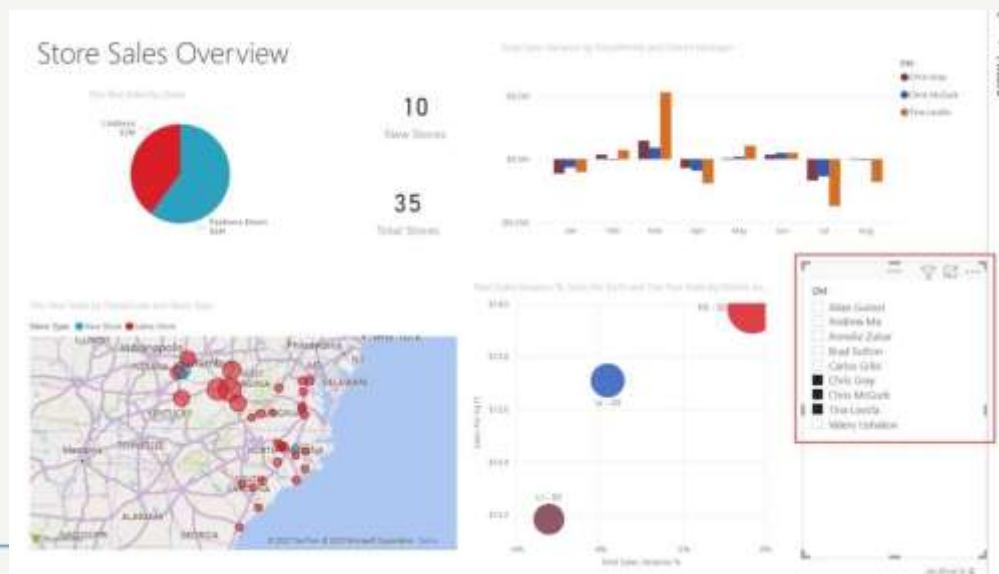


The screenshot shows the Power BI desktop interface. On the left, there is a Slicer visual titled "DM" containing a list of names: Allan Guinot, Andrew Ma, Annelie Zubat, Brad Sutton, Carlos Grilo, Chris Gray, Chris McGurk, Tina Lassila, and Valery Ushakov. On the right, the "Fields" pane is open, showing the "District" field selected. Other fields listed include BusinessUnitID, District, DistrictID, DM (selected), DM\_Pic, DM\_Pic\_B, and DMImage. The "Field" dropdown at the bottom is set to "DM".

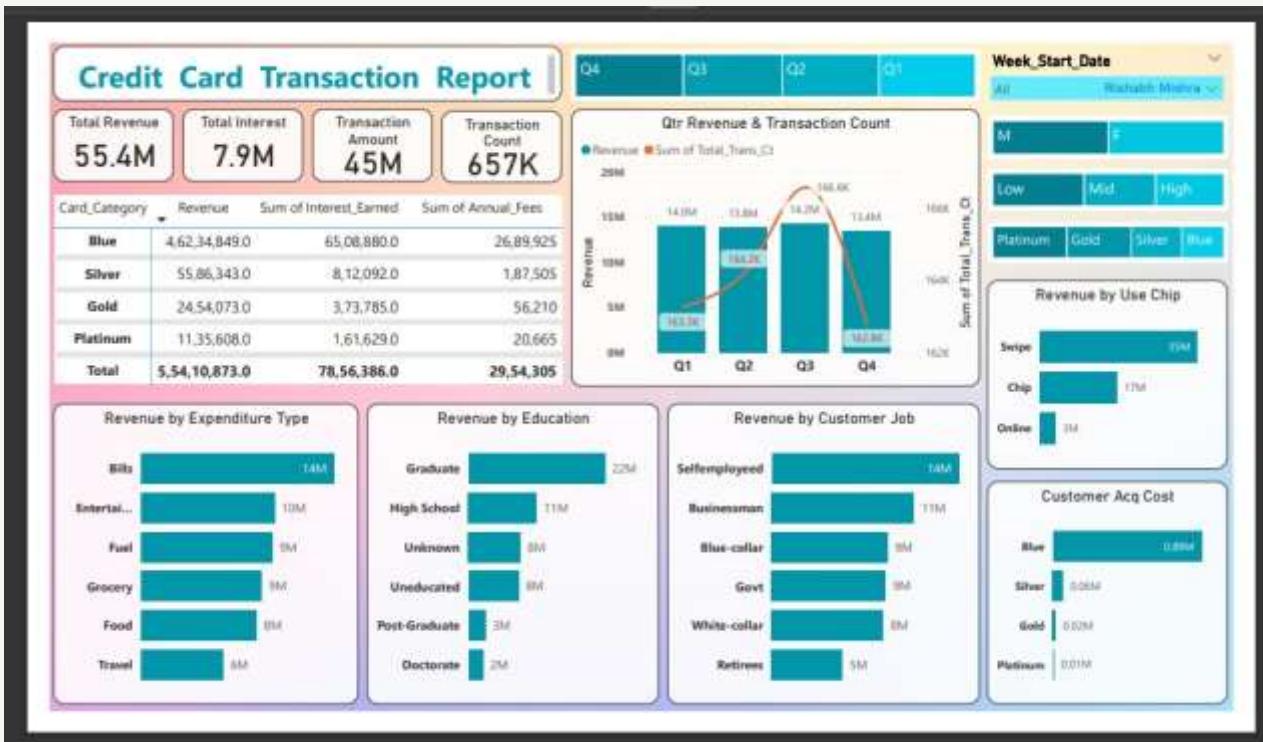
[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205\\_%20Creating%20Dashboard%20with%20Visualization%20Tool/Slicer%20and%20Filter](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205_%20Creating%20Dashboard%20with%20Visualization%20Tool/Slicer%20and%20Filter)

Select names on the slicer and notice the effects on the other visualizations on the page. Select names again to deselect them, or hold down the Ctrl key to select more than one name. Selecting all names has the same effect as selecting none.

Screenshot showing vertical slicer with three names selected and report filtered.



# Creating Dashboard with POWER BI





[https://github.com/TopsCode/Data\\_Analysis\\_2024/tree/main/Module%205%20Creating%20Dashboard%20with%20Visualization%20Tool/Reports%20and%20dashbord](https://github.com/TopsCode/Data_Analysis_2024/tree/main/Module%205%20Creating%20Dashboard%20with%20Visualization%20Tool/Reports%20and%20dashbord)

# **Module- 7**

## **DA - Introduction to Python**

# Introduction of Python

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
- Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development.

# Introduction of Python

- Python supports modules and packages, which encourages program modularity and code reuse.
- The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

- Designed to be easy to learn and master
  - ❖ Clean, clear syntax
  - ❖ Very few keywords
- Highly portable
  - ❖ Runs almost anywhere - high end servers and workstations, down to windows CE
  - ❖ Uses machine independent byte- code
- Extensible
  - Designed to be extensible using C/C++,
  - allowing access to many external libraries

# Features of Python

- ❖ Clean syntax plus high-level data types
  - Leads to fast coding (First language in many universities abroad!)
- ❖ Uses white-space to delimit blocks
  - Humans generally do, so why not the language?
  - Try it, you will end up liking it
- ❖ Uses white-space to delimit blocks
  - Variables do not need declaration
  - Although not a type-less language

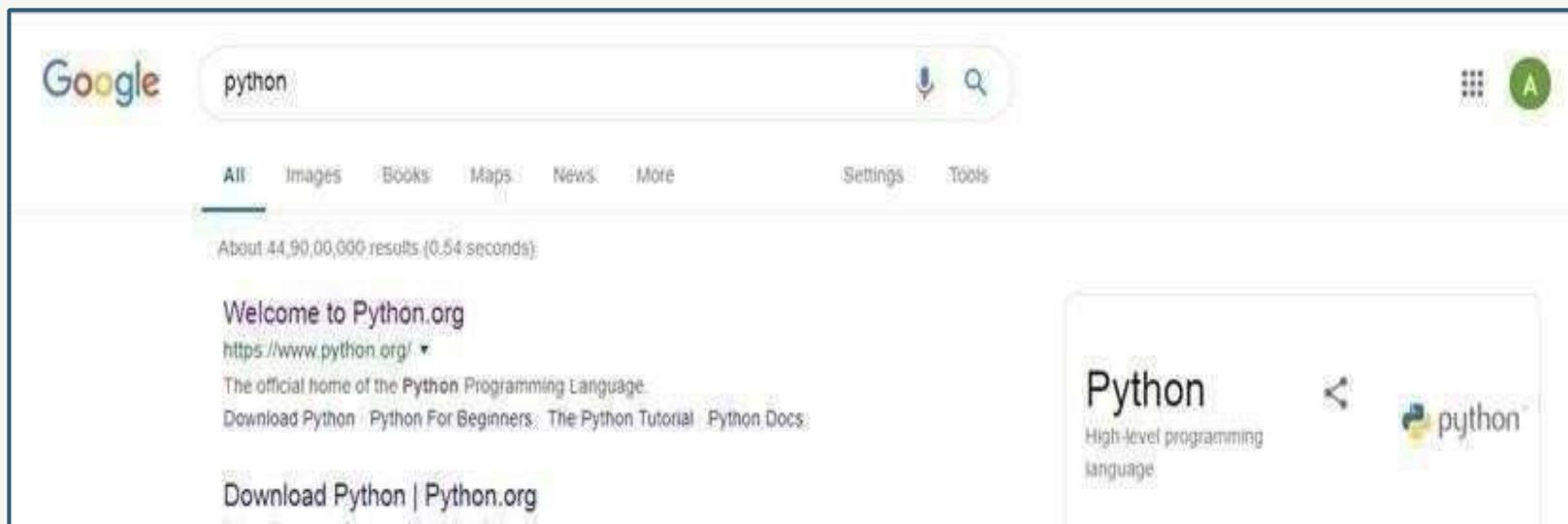
# Features of Python

## Python Productivity

- Reduced development time
  - Code is 2-10x shorter than C, C++, Java
- Improved program maintenance
  - Code is extremely readable
- Less training
  - Language is very easy to learn

# Installation

- **step: 1**



# Installation

- **step : 2 download latest version from python official site**



# Installation

- **step : 3 double click on setup**



# Programming Style

Python programs/ modules are written as text files with traditionally a .py extension.

Each Python module has its own discrete namespace.

Name space within a Python module is a global one.

Python modules and programs are differentiated only by the way they are called.

# Programming Style

- py files executed directly are programs (often referred to as scripts)
- .py files referenced via the import statement are modules.
- Thus, the same .py file can be a program/script, or a module.

# Programming Style

- Python programs/ modules are written as text files with traditionally a .py extension.
- Each Python module has its own discrete namespace.
- Name space within a Python module is a global one.
- Python modules and programs are differentiated only by the way they are called.

# **print() function**

- The print function in Python is a function that outputs to your console window whatever you say you want to print out.
- At first blush, it might appear that the print function is rather useless for programming, but it is actually one of the most widely used functions in all of python. The reason for this is that it makes for a great debugging tool.

# Refer this example :

## 1. Print sample

<https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.1%20print.py>

## 2. single quotation and double quotation

<https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.2%20Single-Double%20Quotation%20print.py>

# **Difference between Python2 and Python3**

One difference between Python 2 and Python 3 is the print statement. In Python 2, the “print” statement is not a function, and therefore can be invoked without a parenthesis. However, in Python 3, it is a function, and must be invoked with parentheses.

## **Python 2**

```
print "Hello"
```

## **Python 3**

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

# Escape Sequences

Escape Sequence	use
\'	Single quote
\\"	Double quote
\	backslash
\n	New line
\t	tab
\b	backspace

## 1.1.3 Escapes sequence

<https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.3%20Escape%20sequence.py>

## **end=“ “**

- The end=' ' is just to say that you want a space after the end of the statement instead of a new line character.

## **ReferThis Example :**

### **1.1.4end practical**

<https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.4%20end%20function%20.py>

## **sep=“ “**

- The separator between the arguments to print() function in Python is space by default (softspace feature) , which can be modified and can be made to any character, integer or string as per our choice.
- The ‘sep’ parameter is used to achieve the same, it is found only in python 3.x or later. It is also used for formatting the output strings.

# ReferThis Example :

## 1.1.5sep practical

<https://github.com/TopsCode/Python/blob/master/Module - 1/1.1%20Programming%20Style/1.1.5%20sep.py>

# Comments

- A comment is a programmer- readable explanation or annotation in the source code of a computer program. They are added with the purpose of making the source code easier for humans to understand, and are generally ignored by compilers and interpreters.

# Types of comments :

Single line comment

Indicate by #

Document comment

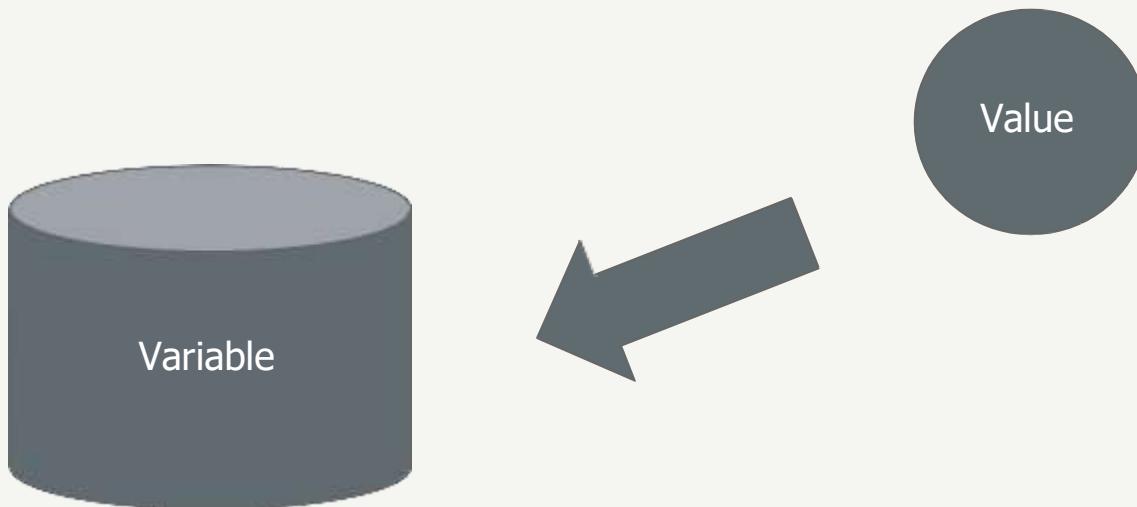
|||||

statements

|||||

# Variables

- Variable :A name which can store a value



- Unlike other programming languages, Python has no command for declaring a variable.
- A variable is created the moment you first assign a value to it.
- E.g.
  - number=20
  - age = 21
- Note : Variables do not need to be declared with any particular type

# Variable Declaration Rules:

- A variable can have a short name (like a and b) or a more descriptive name (age,username,product\_price).
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number  
`10name = "python"`
- A variable name can only contain alphanumeric characters and underscores (A-z, 0-9, and \_ )  
`@name="Python"`

# Variable Declaration Rules:

- Variable names are case-sensitive (age, Age and AGE are three different variables)
- NAME="python" print( name)

Error : NameError: name 'name' is not defined

# ReferThis Examples :

## 1.1.6 Variable sample

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.6%20Variable.py>

## 1.1.7 Sum of two numbers

- [https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.7%20sum%20of%20two%20numbers\(variable\).py](https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.7%20sum%20of%20two%20numbers(variable).py)

## 1.1.8 Swaping of two numbers

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.8%20swaping%20of%20two%20numbers.py>

# RawString

- Python raw string is created by prefixing a string literal with 'r' or 'R'. Python raw string treats backslash (\) as a literal character.

## Refer this Example :

### 1.1.9 Rawstring

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.9%20raw%20string.py>

# **type()**

- Python have a built-in method called as type which generally come in handy while figuring out the type of variable used in the program in the runtime.

## **Refer this Example :**

### **1.1.10type function**

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.10%20type%20function.py>

# Casting()

- Convert one data type value into another data type. In python Casting done with function such as int() or float() or str() .

## Refer this Example :

### 1.1.11 Cast

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.11%20casting.py>

# Taking input from keyboard

Python provides us inbuilt function to read the input from the keyword.

## **input() :**

- This function first takes the input from the user and then evaluates the expression, which means Python automatically identifies whether user entered a string or a number or list.
- If the input provided is not correct then either syntax error or exception is raised by python.

# How input function works:

- When `input()` function executes program flow will be stopped until the user has given an input.
- The text or message display on the output screen to ask a user to enter input value is optional i.e. the prompt, will be printed on the screen is optional.
- Whatever you enter as input, `input` function convert it into a string. if you enter an integer value still `input()` function convert it into a string. You need to explicitly convert it into an integer in your code using typecasting.

# Refer this Example:

## 1.1.12 string input

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.12%20string%20input.py>

## 1.1.13 int input

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.13%20int%20input.py>

# split()

- Using of split() function at some point, need to break a large string down into smaller chunks, or strings.

## Refer this Example :

### 1.1.14split

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.14%20split.py>

# **format()**

- In Python 3, which allows multiple substitutions and value formatting. This method lets us concatenate elements within a string through positional formatting.

## **Refer this Example :**

### **1.1.15 format**

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programing%20Style/1.1.15%20format.py>

# Operators in Python

- To perform specific operations we need to use some symbols ..  
that symbols are operator

## Example :

**A + B**

# Arithmetic Operators

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Exponentiation
//	Floor division

# Assignment Operators

Operator	Example
=	a=10
+=	a+=10
*=	a*=10
/=	a/=10
%=	a%=10
**=	a**=10
//=	a//=10

# Logical Operators

Operator	name
and	And operator
or	Or operator
not	Not operator

# Comparison Operators

Operator	Name
<code>==</code>	Equal
<code>!=</code>	Not equal
<code>&gt;</code>	Greater than
<code>&lt;</code>	Less than
<code>&gt;=</code>	Greater than or equal to
<code>&lt;=</code>	Less than or equal to

# Identity Operators

Operator	Example
is	A is B
is not	A is not B

# Membership Operators

Operator	Example
in	A in student_list
Not in	A not in student_list

# Conditional Statements

- Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions
- Decision structures evaluate multiple expressions which produce TRUEor FALSEas outcome.

# Types of conditional Statements

- If . Statement
- If.. else Statement
- If..Elif..else Statement
- Nested if Statement

# If Statements

- It is similar to that of other languages.
- The if statement contains a logical expression using which data is compared and a decision is made based on the result of the comparison.
- Syntax :  
**if    condition:  
              statements**

# If.. else statement

- It is similar to that of other languages.
- It is frequently the case that you want one thing to happen when a condition is true, and something else to happen when it is false.
- For that we have the if else statement.
- Syntax :

```
if condition:
    statements
else:
    statement(s)
```

# If..elif..else statement

- It is similar to that of other languages.
- The elif is short for else if. It allows us to check for multiple expressions.
- If the condition for if is False, it checks the condition of the next elif block and so on.
- If all the conditions are False, body of else is executed.
- Only one block among the several if...elif...else blocks is executed according to the condition.

# If..elif..else statement

- Syntax :

```
If condition:  
    statement  
Elif condition:  
    Statement
```

# Nested if....else statement

- There may be a situation when you want to check for another condition after a condition resolves to true.
- In such a situation, you can use the nested if construct.
- Syntax :

```
if    condition:  
      statements  
      if    condition:  
        statements  
      else:  
        statement(s)
```

# Refer this Example:

## 1. if statement

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.2%20Conditional%20Statements/1.2.1%20if%20statement.py>

## 2. ifelsestatement

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.2%20Conditional%20Statements/1.2.2%20if%20else%20statement.py>

### 1.2.3 elifstatement

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.2%20Conditional%20Statements/1.2.3%20elif%20statement.py>

### 1.2.4 nestedif statement

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.2%20Conditional%20Statements/1.2.4%20nested%20if%20statement.py>

# Looping Statements

- A loop statement allows us to execute a statement or group of statements multiple times.
- Python programming language provides following types of loops to handle looping requirements.
  - **For Loop**
  - **While Loop**

# ForLoops

- For loop has the ability to iterate over the items of any sequence, such as a list or a string.
- Syntax :

```
for iterating_var insequence:  
    statements(s)
```
- If a sequence contains an expression list, it is evaluated first.
- Then, the first item in the sequence is assigned to the iterating variable `iterating_var`.

- Next, the statements block is executed.
- Each item in the list is assigned to *iterating\_var*, and the statement( s) block is executed until the entire sequence is exhausted

### **Refer this Example :**

#### **1.3.1forloop**

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.3%20Looping%20Statement/1.3.1%20for%20loop%20with%20string.py>

#### **1.3.2 for loop withlist**

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.3%20Looping%20Statement/1.3.2%20for%20loop%20with%20list.py>

# range() function

- To loop through a set of code a specified number of times, we can use the range()function,
- The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1(by default), and ends at a specified number.
- The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6):

# Refer this Example:

## 1.3.3 for loop with range

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.3%20Looping%20Statement/1.3.3%20for%20loop%20with%20range.py>

## 1.3.4 for loop with decrement range

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.3%20Looping%20Statement/1.3.4%20for%20loop%20with%20decrement.py>

# NestedForloop

- Python programming language allows to use one loop inside another loop.
- Syntax :

```
for iterating_var in sequence:    for  
    iterating_var in sequence:  
        statements(s)  
    statements(s)
```

# Referthis Example:

## 1.3.5nested for loop

- [https://github.com/TopsCode/Python/blob/master/Module\\_1/1.3%20Looping%20Statement/1.3.5%20nested%20for%20loop.py](https://github.com/TopsCode/Python/blob/master/Module_1/1.3%20Looping%20Statement/1.3.5%20nested%20for%20loop.py)

## 1.3.6pattern 1

- [https://github.com/TopsCode/Python/blob/master/Module\\_1/1.3%20Looping%20Statement/1.3.6%20pattern%201.py](https://github.com/TopsCode/Python/blob/master/Module_1/1.3%20Looping%20Statement/1.3.6%20pattern%201.py)

## 1.3.7pattern 2

- [https://github.com/TopsCode/Python/blob/master/Module\\_1/1.3%20Looping%20Statement/1.3.7%20pattern%202.py](https://github.com/TopsCode/Python/blob/master/Module_1/1.3%20Looping%20Statement/1.3.7%20pattern%202.py)

## 1.3.8pattern 3

- [https://github.com/TopsCode/Python/blob/master/Module\\_1/1.3%20Looping%20Statement/1.3.8%20pattern%203.py](https://github.com/TopsCode/Python/blob/master/Module_1/1.3%20Looping%20Statement/1.3.8%20pattern%203.py)

## 1.3.9pattern 4

- [https://github.com/TopsCode/Python/blob/master/Module\\_1/1.3%20Looping%20Statement/1.3.9%20pattern%204.py](https://github.com/TopsCode/Python/blob/master/Module_1/1.3%20Looping%20Statement/1.3.9%20pattern%204.py)

# While Loop

- A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.
- Here, statement( s) may be a single statement or a block of statements. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is
- **Syntax :**

```
while expression:  
    statement(s)
```

## 1.3.10 whileloop

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.3%20Looping%20Statement/1.3.10%20while%20loop.py>

# Control Statements

- Loop control statements change execution from its normal sequence.
- When execution leaves a scope, all automatic objects that were created in that scope are destroyed.
- Python supports the following control statements.
  - Break
  - Continue
  - Pass

# Break Statements

- It brings control out of the loop and transfers execution to the statement immediately following the loop.
- Syntax :  
`break`

## Refer this Example :

### 1.4.1 break statement

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.4%20control%20statament/1.4.1%20break%20statement.py>

# Continue Statements

- It continues with the next iteration of the loop.
- Syntax :

continue

## Refer this Example:

### 1.4.2 continue statement

- <https://github.com/TopsCode/Python/blob/master/Module - 1/1.4%20control%20statament/1.4.1%20break%20statement.py>

# PassStatements

- The pass statement does nothing.
- It can be used when a statement is required syntactically but the program requires no action.

Syntax :

```
pass
```

## Refer this Example :

### 1.4.3 pass statement

- <https://github.com/TopsCode/Python/blob/master/Module-1/1.4%20control%20statament/1.4.3%20pass%20statement.py>

# String Manipulation

- Textual data in Python is handled with "str" objects, or strings. Strings are immutable( fixed/ rigid) sequences of Unicode codepoints.
- String literals are written in a variety of ways:
  - Single quotes: 'allows embedded "double" quotes'
  - Double quotes: "allows embedded 'single' quotes".
  - Triple quoted:  
    """Three single quotes""", """Three double quotes"""

Note : Triple quoted strings may span multiple lines - all associated whitespace will be included in the string literal.

# String Functions

- str.capitalize()

Return a copy of the string with its first character capitalized and the rest lowercase.

- str.casefold()

Return a case folded copy of the string. Case folded strings may be used for caseless matching.

- str.center(width[, fillchar])

Return centered in a string of length width. Padding is done using the specified fillchar (default is an ASCII space). The original string is returned if width is less than or equal to len(s).

- str.count(sub[, start[, end]])

Return the number of non-overlapping occurrences of substring sub in the range [start, end].

Optional arguments start and end are interpreted as in slice notation.

- str.endswith(suffix[, start[, end]])

Return True if the string ends with the specified suffix, otherwise return False.

suffix can also be a tuple of suffixes to look for. With optional start, test beginning at that position. With optional end, stop comparing at that position.

# String Functions

- `str.find(sub[, start[, end]])`

Return the lowest index in the string where substring sub is found within the slice s[start:end].

Optional arguments start and end are interpreted as in slice notation. Return -1 if sub is not found.

Note:

The `find()` method should be used only if you need to know the position of sub. To check if sub is a substring or not, use the "in"operator:

```
>>>>>'Py' in 'Python' True
```

# String Functions

- `str.format(*args, **kwargs)`

Perform a string formatting operation. The string on which this method is called can contain literal text or replacement fields delimited by braces{}.

Each replacement field contains either the numeric index of a positional argument, or the name of a keyword argument. Returns a copy of the string where each replacement field is replaced with the string value of the corresponding argument.

# String Functions

- `str.index(sub[, start[, end]])`)
- Like `find()`, but raise `ValueError` when the substring is not found.
- `str.isalnum()`  
Return true if all characters in the string are alphanumeric and there is at least one character, false otherwise.
- A character `c` is alphanumeric if one of the following returns True:
- `c.isalpha(), c.isdecimal(), c.isdigit(), or c.isnumeric().`

# String Functions

- `str.isidentifier()`

Return true if the string is a valid identifier according to the language definition

- `str.islower()`

Return true if all cased characters in the string are lowercase and there is at least one cased character, false otherwise.

- `str.istitle()`

Return true if the string is a title cased string and there is at least one character, for example uppercase characters may only follow uncased characters and lowercase characters only cased ones.

Return false otherwise.

# String Functions

- **str.isupper()**

Return true if all cased characters in the string are uppercase and there is at least one cased character, false otherwise.

- **str.join(*iterable*)**

Return a string which is the concatenation of the strings in the iterable iterable. A TypeError will be raised if there are any non- string values in iterable.

- **str.ljust(*width[, fillchar]*)**

Return the string left justified in a string of length width. Padding is done using the specified fillchar (default is an ASCII space). The original string is returned if width is less than or equal to len(s).

# String Functions

- `str.partition(sep)`

Split the string at the first occurrence of *sep*, and return a 3 - tuple containing the part before the separator, the separator itself, and the part after the separator.

If the separator is not found, return a 3-tuple containing the string itself, followed by two empty strings.

- `str.replace(old, new[, count])`

Return a copy of the string with all occurrences of substring *old* replaced by *new*.

If the optional argument *count* is given, only the first *count* occurrences are replaced.

# String Functions

- `str.split(sep=None, maxsplit=-1)`

:

Return a list of the words in the string, using sep as the delimiter string. If maxsplit is given, at most maxsplit splits are done (thus, the list will have at most maxsplit+1 elements).

If maxsplit is not specified or -1, then there is no limit on the number of splits

- `str.replace(old, new[, count])`

Return a copy of the string with all occurrences of substring old replaced by new.

If the optional argument count is given, only the first count occurrences are replaced.

# String Functions

- `str.swapcase()`
  - Return a copy of the string with uppercase characters converted to lowercase and vice versa.
  
- `str.title()`
  - Return a titlecased version of the string where words startwith an uppercase character and the remaining characters are lowercase.

# String Slicing

- Like other programming languages, it's possible to access individual characters of a string by using array-like indexing syntax.
- In this we can access each and every element of string through their index number and the indexing starts from 0.
- Python does index out of bound checking.
- So, we can obtain the required character using syntax, `string_name[index_position]`:
- The positive `index_position` denotes the element from the starting(0) and the negative index shows the index from the end(-1).

# String Slicing

- To extract substring from the whole string then we use the syntax like
- `string_name[beginning: end :step]` beginning represents the starting index of string end denotes the end index of string which is not inclusive steps denotes the distance between the two words.

# Refer this example :

## 1.5.1 String demo

- <https://github.com/TopsCode/Python/blob/master/Module1/1.5%20string/1.5.1%20StringDemo.py>

## 1.5.2 String Operations

- <https://github.com/TopsCode/Python/blob/master/Module1/1.5%20string/1.5.2%20StringOperation.py>

## 1.5.3 String slicing

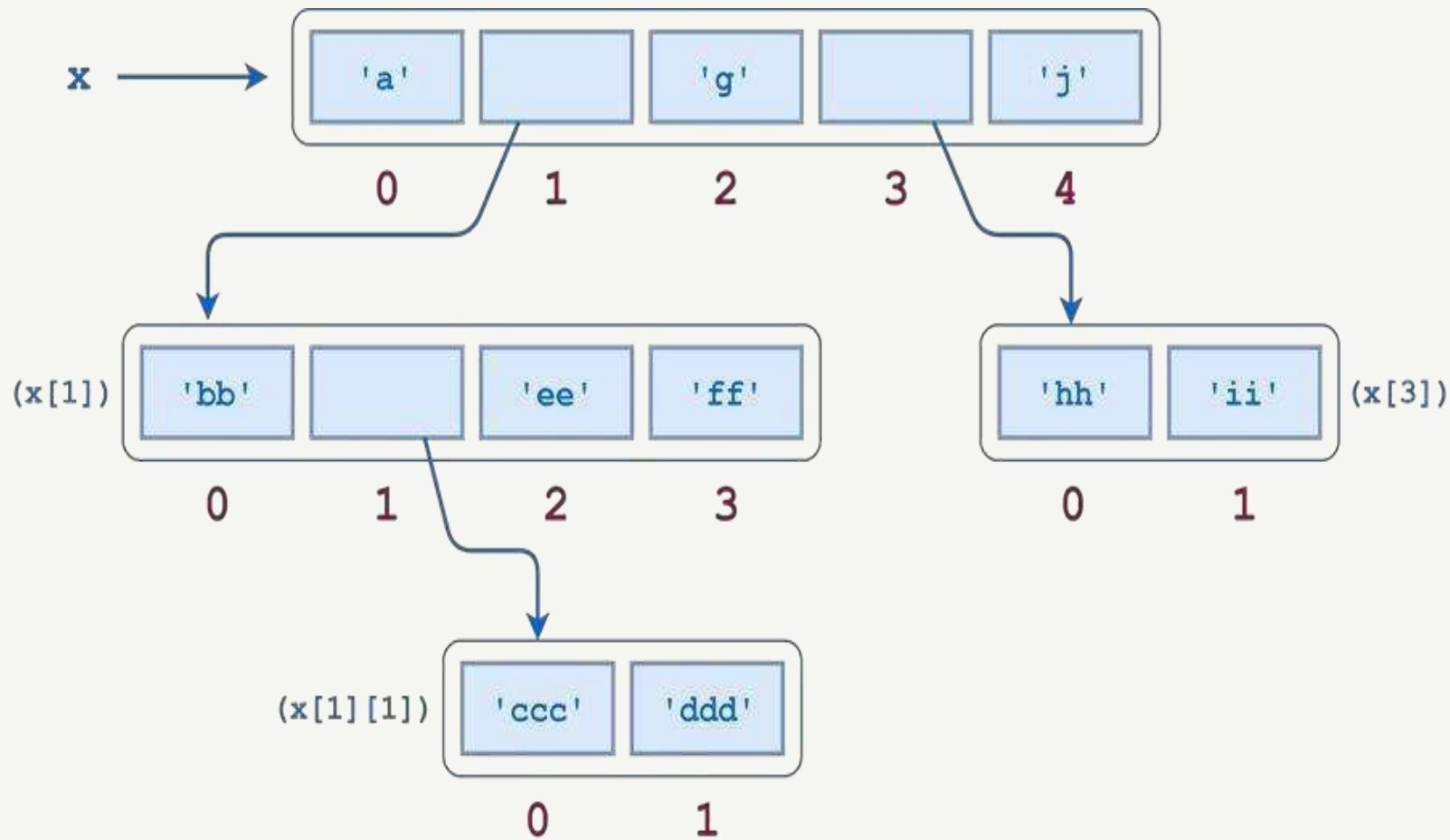
- <https://github.com/TopsCode/Python/blob/master/Module1/1.5%20string/1.5.3%20StringSlicing.py>

# **Module- 2**

# **Collection, Function And Modules**

# List [ ]

- Introduction
- Accessing list
- Operations
- Working with lists
- Function and Methods



# Built-in Types

Types	Description
numerics	There are three distinct numeric types: <i>integers</i> , <i>floating point numbers</i> , and <i>complex numbers</i> .
Sequences	There are three basic sequence types: lists, tuples, and range objects. Additional sequence types tailored for processing of binary data and text strings.

Mappings	A mapping object maps hashable values to arbitrary objects. Mappings are mutable objects. There is currently only one standard mapping type, the <i>dictionary</i> .
Classes	Python classes provide all the standard features of Object Oriented Programming.
Instances	instances of user-definedclasses.
exceptions	All exceptions must be instances of a class.

# Introduction

- Python knows a number of compound data types, used to group together other values.
- The most versatile is the list, which can be written as a list of comma-separated values (items) between squarebrackets.
- Lists might contain items of different types, but usually the items all have the same type.

# Accessing List

- Accessing List

Like strings (and all other built-in sequence type), lists can be indexed and sliced

Example:

fruits[0]

Example

:fruits[-3:-1]

Unlike strings, which are immutable, lists are a mutable type, i.e. it is possible to change their content.

# Operations

- **“in” operator:-** This operator is used to check if an element is present in the list or not. Returns true if element is present in list else returns false.
- **“not in” operator:-** This operator is used to check if an element is not present in the list or not.
- Returns true if element is not present in list else returns false.

# Operations

- Common applications are to make new lists where each element is the result of some operations applied to each member of another sequence or iterable, or to create a subsequence of those elements that satisfy a certain condition.

# Functions and Methods

Name	Description
len(list)	Gives the total length of the list.
max(list)	Returns item from the list with max value.
min(list)	Returns item from the list with min value.
list(seq)	Converts a tuple into list.

# Functions and Methods

Methods	Description
list.append(x) )	Add an item to the end of the list. Equivalent to a [len(a):]=[x].
list.extend(L)	Appends the contents of L to list
list.insert(I,x)	Insert an item at a given position. The first argument is the index of the element before which to insert, so a.insert(0,x) inserts at the front of the list, and a.insert(len(a),x) is equivalent to a.append(x).

list.count(obj )	Returns count of how many times obj occurs in list
list.index(obj )	Returns the lowest index in list that obj appears
List.pop(obj =l ist[-1]	Removes and returns last object or obj from list.

# Functions and Methods

Name	Description
list.reverse()	Reverses objects of list in place
list.sort([fun])	Sorts objects of list, use compare <b>fun</b> if given
list.remove(obj)	Removes object obj from list

# Using Lists as Stacks

- The list methods make it very easy to use a list as a stack, where the last element added is the first element retrieved("last-in,first-out") . To add an item to the top of the stack, use append().
- To retrieve an item from the top of the stack, use pop() without an explicit index.

# Using Lists as Queue

- It is also possible to use a list as a queue, where the first element added is the first element retrieved("first-in, first-out"); however, lists are not efficient for this purpose. While appends and pops from the end of list are fast, doing inserts or pops from the beginning of a list is slow(because all of the other elements have to be shifted by one).

# Refer this example:

## 2.1.1 Listas Queue

- <https://github.com/TopsCode/Python/blob/master/Module - 2/2.1%20List/2.1.1%20ListAsQueue.py>

## 2.1.2 ListDemo

- <https://github.com/TopsCode/Python/blob/master/Module - 2/2.1%20List/2.1.2%20ListDemo.py>

## 2.1.3 Listoperations

- <https://github.com/TopsCode/Python/blob/master/Module - 2/2.1%20List/2.1.3%20ListOperation.py>

## 2.1.4 Listpattern

- <https://github.com/TopsCode/Python/blob/master/Module - 2/2.1%20List/2.1.4%20ListPattern.py>

# Tuple ( )

- Introduction
- Accessing tuples
- Operations
- Working
- Functions and Methods

# Introduction

- A tuple is a sequence of immutable Python objects.
- Tuples are sequences, just like lists.
- The differences between tuples and lists are, the tuples cannot be
  - changed unlike lists and tuples use parentheses, whereas lists use
  - square brackets.
- Eg fruits=("Mango","Banana","Oranges",23,44)
- Eg numbers=(11,22,33,44)
- Eg fruits="Mango","Banana","Oranges"

# Introduction

- Unlike lists, tuples are immutable. This means that elements of a tuple cannot be changed once it has been assigned.
- But if the element is itself a mutable data type like list, its nested items can be changed.
- We can also assign a tuple to different values (reassignment).
- Also We cannot delete or remove items from a tuple.
- But deleting the tuple entirely is possible using the keyword del.

# Accessing tuples

- There are various ways in which we can access the elements of a tuple.
- We can use the index operator [] to access an item in a tuple.
- Index starts from 0. The index must be an integer.
- Python allows negative indexing for its sequences.
- The index of -1 refers to the last item, -2 to the second last item and so on.
- We can access a range of items in a tuple by using the slicing operator (colon).
- Eg. fruits [2:]

# Operations

- With Tuples we can do concate ,repetition,iterations etc...

# Functions

Name	Description
len(tuple)	Gives the total length of the tuple.
max(tuple)	Returns item from the tuple with max value.
min(tuple)	Returns item from the tuple with min value.
tuple(seq)	Converts a seq into tuple.

# Methods

Method	Description
count(obj)	Returns count of how many times obj occurs in tuple
index(obj)	Returns the lowest index in tuple that obj appears

# Refer this example :

## 2.2.1 add item tuple

- [https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.1%20add\\_item\\_tuple.py](https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.1%20add_item_tuple.py)

## 2.2.2 convert listtuple

- [https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.2%20convert\\_list\\_tuple.py](https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.2%20convert_list_tuple.py)

-

# Refer this example :

## 2.2.5 create tuple

- [https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.5%20create\\_tuple.py](https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.5%20create_tuple.py)

## 2.2.6 Find repeat item tuple

- [https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.6%20find\\_repaeat\\_item\\_tuple.py](https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.6%20find_repaeat_item_tuple.py)

## 2.2.7 slicetuple

- [https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.7%20slice\\_tuple.py](https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.7%20slice_tuple.py)

# Dictionaries

- Introduction
- Accessing values in dictionaries
- Working with dictionaries
- Properties
- Functions

# Introduction

- Dictionaries are sometimes found in other languages as “associative memories” or “associative arrays”.
- Unlike sequences, which are indexed by a range of numbers, dictionaries are indexed by *keys*, which can be any immutable type; strings and numbers can always be keys.
- Tuples can be used as keys if they contain only strings, numbers, or tuples; if a tuple contains any mutable object either directly or indirectly, it cannot be used as a key.

# Introduction

- You can't use lists as keys, since lists can be modified in place using index assignments, slice assignments, or methods like `append()` and `extend()`.
- The main operations on a dictionary are storing a value with some key and extracting the value given the key.
- Like lists they can be easily changed, can be shrunk and grown ad libitum at run time. They shrink and grow without the necessity of making copies. Dictionaries can be contained in lists and vice versa.

# Introduction

- A list is an ordered sequence of objects, whereas dictionaries are unordered sets.
- But the main difference is that items in dictionaries are accessed via keys and not via their position.

# Accessing Values

- To access dictionary elements, we can use the familiar square brackets along with the key to obtain its value.
- It is an error to extract a value using a non-existent key.
- We can also create a dictionary using the built-in class dict() (constructor).
- We can test if a key is in a dictionary or not using the keyword in.
- The membership test is for keys only, not for values.

# Properties

- **Properties of Dictionaries**

- Dictionary values have no restrictions.
- They can be any arbitrary Python object, either standard objects or user-defined objects.
- However, same is not true for the keys.

- **More than one entry per key not allowed.**

- Which means no duplicate key is allowed.
- When duplicate keys encountered during assignment, the last assignment wins.

# Properties

- **Keys must be immutable.**
  - Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed.

# Methods and Functions

<b>Method</b>	<b>Description</b>
dist.copy()	A dictionary can be copied with the method copy():.
dist.update()	It merges the keys and values of one dictionary into another, overwriting values of the same key.
dist.values()	It returns the list of dictionary dict's values.
dist.keys()	It returns the list of dictionary dict's keys.
dist.items()	It returns the list of dictionary dict's keys,values in tuple pairs.
dist.clear()	Removes all elements of dictionarydict.

# Refer this example :

## 2.3.1 Dictionary example

- <https://github.com/TopsCode/Python/blob/master/Module-2/2.3%20Dictionaries/2.3.1%20DictionaryDemo.py>

## 2.3.2 Dictionary method demo

- <https://github.com/TopsCode/Python/blob/master/Module-2/2.3%20Dictionaries/2.3.2%20DictionaryMethodDemo.py>

# Functions

- A function is a block of organized, reusable code that is used to perform a single, related action.
- Functions provide better modularity for your application and a high degree of code reusing.

# Defining a Functions

- **Defining a function**
  - Python gives us many built-in functions like print(), etc. but we can also create our own functions.
  - A function in Python is defined by a def statement. The general syntax looks like this:

```
def function-name( Parameter list):  
    statements, i.e. the function  
    body  
  
    return [expression]
```

# Defining a Functions

- The keyword “def” introduces a function definition.
- It must be followed by the function name and the parenthesized list of formal parameters. The statements that form the body of the function start at the next line, and must be indented.
- The “return” statement returns with a value from a function.“return” without an expression argument returns None.
- Falling off the end of a function also returns None.

# Calling a Functions

- Once function define, we can call it directly or in any otherfunction also.
- Syntax :

function  
name()  
or  
functionname( argument)

## 3.1.1 Create function

- [https://github.com/TopsCode/Python  
/blob/master /Module-  
3/3.1%20Functions/3.1.1%20create%20fu  
nction.py](https://github.com/TopsCode/Python/blob/master /Module-3/3.1%20Functions/3.1.1%20create%20function.py)

# Types of Functions

## Functions can be of

### Built-in functions

Functions that come built into the Python language itself are called built-in functions and are readily available to us.  
Eg: `input()`, `eval()`, `print()` etc...

### User defined functions

Functions that we define ourselves to do certain specific task are referred as user-defined functions.  
Eg: `checkNoEvenOdd(20)`

# Function Arguments

- It is possible to define functions with a variable number of arguments.
- The function arguments can be
  - Default arguments values
  - Keyword arguments

# Function Arguments

- Default arguments values
  - The most useful form is to specify a default value for one or more arguments.
  - This creates a function that can be called with fewer arguments than it is defined to allow.
  - Eg. def employeeDetails(name,gender='male',age=35)
  - This function can be called in several ways:
  - giving only the mandatory argument :  
`employeeDetails("Ramesh")`

- giving one of the optional arguments:  
`employeeDetails("Ramesh",'Female')`
  
- or even giving all arguments :  
`employeeDetails("Ramesh",'Female',31)`

# Function Arguments

- Note : The default value is evaluated only once. This makes a difference when the default is a mutable object such as a list, dictionary, or instances of most classes. For example, the following function accumulates the arguments passed to it on subsequent calls:

# Function Arguments

- **Keyword Arguments**

- Functions can also be called using keyword arguments of the form `kwarg=value`.
- For instance, the following function:
- `def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):`
- accepts one required argument (`voltage`) and three optional arguments (`state`, `action`, and `type`).

# Function Arguments

```
def parrot(voltage, state='a stiff', action='voom',
           type='Norwegian Blue'):
```

parrot(1000)	<i>positional argument</i>
--------------	----------------------------

parrot(voltage=1000)	<i>keyword argument</i>
----------------------	-------------------------

parrot(voltage=1000000,            action='VOOOOOM')	<i>keyword arguments</i>
--	--------------------------

parrot(action='VOOOOOM',            voltage=1000000)	<i>keyword arguments</i>
--	--------------------------

parrot('a million', 'bereft of life',            'jump')	<i>positional arguments</i>
--	-----------------------------

parrot('a thousand', state='pushing            up thedaisies')	<i>positional, 1 keyword</i>
--	------------------------------

# Function Arguments

- **Arbitrary Argument Lists**
  - These arguments will be wrapped up in a tuple . Before the variable number of arguments, zero or more normal arguments may occur.
  - Eg :

```
def write_multiple_items(file, separator, *args): file.write(separator.join(args))
```
  - Normally, these variadic arguments will be last in the list of formal parameters, because they scoop up all remaining input arguments that are passed to the function.

# Function Arguments

- Any formal parameters which occur after the \*args parameter are 'keyword-only' arguments, meaning that they can only be used as keywords rather than positional arguments.

```
def concat(*args, sep="/"):  
    return sep.join(args)
```

```
concat("earth", "mars", "venus") O/P-  
>'earth/mars/venus' concat("earth", "mars", "venus",  
sep=".") O/P-  
>'earth.mars.venus'
```

# Refer this example :

## 3.1.2 Function with parameter

- <https://github.com/TopsCode/Python/blob/master/Module-3/3.1%20Functions/3.1.2%20function%20with%20parameter.py>

## 3.1.3 Function with default value

- <https://github.com/TopsCode/Python/blob/master/Module-3/3.1%20Functions/3.1.3%20function%20with%20default%20value.py>
- <https://github.com/TopsCode/Python/blob/master/Module-3/3.1%20Functions/3.1.6%20dict%20as%20parameter.py>

### **3.1.4 Function with return values**

- <https://github.com/TopsCode/Python/blob/master/Module-3/3.1%20Functions/3.1.4%20function%20with%20return%20values.py>

### **3.1.5 tuple as parameter**

- <https://github.com/TopsCode/Python/blob/master/Module-3/3.1%20Functions/3.1.5%20tuple%20as%20parameter.py>

### **3.1.6 dict as parameter**

- <https://github.com/TopsCode/Python/blob/master/Module->

# Function Arguments

- **Anonymous functions**

- In Python, anonymous function is a function that is defined without a name.
- While normal functions are defined using the def keyword, in Python anonymous functions are defined using the lambda keyword.
- Hence, anonymous functions are also called lambda functions.
- A lambda function has the following syntax.
- lambda arguments: expression

# Function Arguments

- Lambda functions can have any number of arguments but only one expression.
- The expression is evaluated and returned.
- Lambda functions can be used wherever function objects are required.
- Python supports a style of programming called functional programming where you can pass functions to other functions to do stuff.

### **3.2.1 lambda function**

- [https://github.com/TopsCode/Python/blob/  
master/Module-  
3/3.2%20Anonymous%20functions/3.2.1%20la  
mbda%20function.py](https://github.com/TopsCode/Python/blob/master/Module-3/3.2%20Anonymous%20functions/3.2.1%20lambda%20function.py)

# Global & Localvariables

## • **Global Variables**

- Defining a variable on the module level makes it a global variable, you don't need to global keyword.
- The global keyword is needed only if you want to reassign the global variables in the function/method.
- Defining a variable on the module level makes it a global variable, you don't need to global keyword.
- The global keyword is needed only if you want to reassign the global variables in the function/method.

# Global & Localvariables

- **LocalVariables**

- If a variable is assigned a value anywhere within the function's body, it's assumed to be a local unless explicitly declared as global.
- Local variables of functions can't be accessed from outside

### **3.3.1 global variable**

- <https://github.com/TopsCode/Python/blob/master/Module%20-3/3.3%20Global%20-%20Local%20Variables/3.3.1%20global%20variable.py>

### **3.3.2 local variable**

- <https://github.com/TopsCode/Python/blob/master/Module%20-3/3.3%20Global%20-%20Local%20Variables/3.3.1%20global%20variable.py>

### **3.3.3 global keyword**

- <https://github.com/TopsCode/Python/blob/master/Module%20-3/3.3%20Global%20-%20Local%20Variables/3.3.3%20global%20keyword%20use.py>

# Modules

- A module is a file containing Python definitions and statements.
- The file name is the module name with the suffix .py appended.
- Within a module, the module's name (as a string) is available as the value of the global variable `name` .

# Importing Module

- Modules can import other modules.
- It is customary but not required to place all import statements at the beginning of a module (or script, for that matter).
- The imported module names are placed in the importing module's global symbol table.

**Eg : import fibo**

- Using the module name we can access functions.

fibo.fib

(10)

fibo.fib

2(10)

# Importing Module

- There is a variant of the import statement that imports names from a module directly into the importing module's symbol table.

For example:

```
from fibo import fib,  
fib2 fib(500)  
another way to import  
is from fibo import *
```

## 4.1.1 module example

- [https://github.com/TopsCode/Python/blob/  
master/Module-  
4/4.1%20module/4.1.1%20sample.py](https://github.com/TopsCode/Python/blob/master/Module-4/4.1%20module/4.1.1%20sample.py)

# Random Module

- Python offers random module that can generate random numbers.
- These are pseudo-random number as the sequence of number generated depends on the seed.

## 4.2.1 Random module

- <https://github.com/TopsCode/Python/blob/master/Module-4/4.2%20random%20module/4.2.1%20RandomModulesDemo.py>

# Random Functions

Function	Description
<code>random.randrange( s tart,stop[, step])</code>	Return a randomlyselected element from range(start, stop, step). This is equivalent to choice(range(start, stop, step)), butdoesn't actually build a rangeobject.
<code>random.randint(a, b)</code>	Return a random integer $N$ such that $a \leq N \leq b$ . Alias for randrange(a, b+1).
<code>random.choice(seq)</code>	Return a random element from the non- empty sequence seq.
<code>random. random()</code>	Return the next random floating point number in the range [0.0, 1.0).
And many more...	

# Math Module

- This module is always available.
- It provides access to the mathematical functions defined by the C standard.
- These functions are divided into some categories like Number-theoretic and representation functions, Power and logarithmic functions, Trigonometric functions, Angular conversion, Hyperbolic functions, Special functions.
- Constants
- `math.pi` : The mathematical constant  $\pi = 3.141592\dots$ , to available precision.
- `math.e` : The mathematical constant  $e = 2.718281 \dots$ , to available precision.,

# Math Module

- `math.inf`: A floating-point positive infinity. (For negative infinity, use `-math.inf`.) Equivalent to the output of `float('inf')`.
- `math.nan` : A floating-point “not a number” (NaN) value. Equivalent to the output of `float('nan')`

Function	Description
<code>math.ceil(x)</code>	Return the ceiling of $x$ , the smallest integer greater than or equal to $x$ . If $x$ is not a float, delegates to <code>x.ceil()</code> , which should return an Integralvalue.
<code>math.factorial(x)</code>	Return $x$ factorial

# Math Module

Function	Description
math.floor( <i>x</i> )	Return the floor of <i>x</i> , the largest integer less than or equal to <i>x</i> . If <i>x</i> is not a float, delegates to <i>x</i> . floor(), which should return an Integral value.
math.trunc( <i>x</i> )	Return the Real value <i>x</i> truncated to an Integral (usually an integer).
math.pow( <i>x</i> , <i>y</i> )	Return <i>x</i> raised to the power <i>y</i> .
And so more...	

## 4.3.1 Math module

- <https://github.com/TopsCode/Python/blob/master/Module-4/4.3%20math%20module/4.3.1%20MathModuleDemo.py>

# Built-in Libraries

- [random](#)
- [math](#)
- [calendar](#)
- [os](#)
- [datetime](#)
- [platform](#)
- [keyword](#)
- [requests](#)

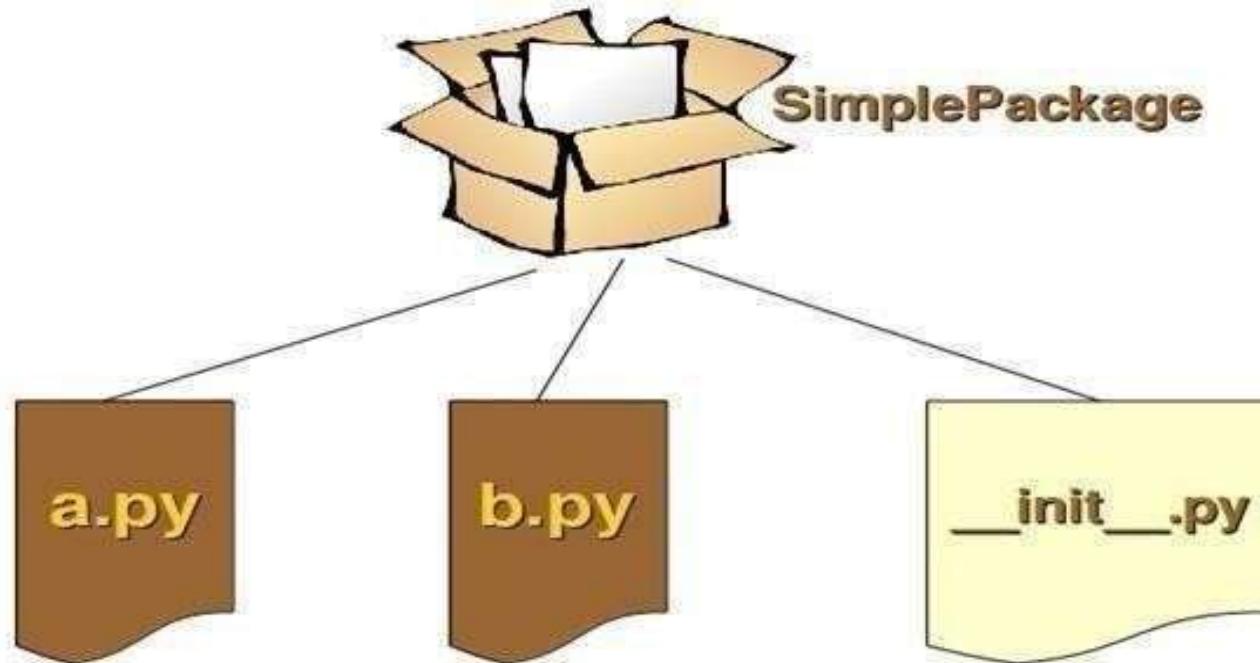
# Packages

- Packages are a way of structuring Python's module namespace by using "dotted module names".
- For example, the module name A.B designates a submodule named B in a package named A.
- A package is basically a directory with Python files.

## 4.4.1 Package example

- [https://github.com/TopsCode/Python/blob/  
master/Module-  
4/4.4%20packages/4.4.1%20package%20sam  
ple.py](https://github.com/TopsCode/Python/blob/master/Module-4/4.4%20packages/4.4.1%20package%20sample.py)

# Packages



# External Libraries

## Installation

Before using an external library, you need to install it. You can typically install libraries using a package manager like ***pip (Python's package installer)***.

**Example:** To install the *requests* library for making HTTP requests, you would run ***pip install requests*** in your command line.

## Importing:

Once installed, you import the library into your Python script or interactive environment using the import statement.

**Example:** *import requests*

# External Libraries

- [pandas](#)
- [instaloader](#)
- [pyqrCode](#)
- [requests](#)
- [pywhatkit](#)
- [pytube](#)

# Printing on screen

- “print()” is use to print on screen.
- `print(value, ..,sep=' ',end='\n', file=sys.stdout)`
- prints the values to a stream, or to sys.stdout by default.
- **Optional keyword arguments:**
  - file: a file-like object (stream); defaults to the current sys.stdout.
  - sep: string inserted between values, default a space.
  - end: string appended after the last value, default a newline.

# Reading from keyboard

- To read data from keyboard “`input()`” is use.
- The input of the user will be returned as a string without any changes.
- If this raw input has to be transformed into another data type needed by the algorithm, we can use either a casting function or the `eval` function.

# Opening and Closing file

- To read data from keyboard “input()” is use.
- “open()” is use to open the file.
- It returns file object.
- syntax
- open(fileName,mode).
- fileName: Name of the file that we wants to open.
- mode: ‘r’ (only for reading), ‘w’ (only for writing), ‘a’ (for append) ,r+ (for read and write).
- Normally, files are opened in text mode, that means, you read and write strings from and to the file, which are encoded in a specific encoding.
- If encoding is not specified, the default is platformdependent

# Opening and Closing file

- **Close the file**

call `f.close()` to close it and free up any system resources taken up by the open file.

# Reading and writing files

- `file.read(size)` :This function is use to read a file's contents.
- If size is omitted or negative then the entire content is returned.
- If the end of the file has been reached, `f.read()` will return an empty string (' ').
- `f.readline()` reads a single line from the file; a newline character (`\n`) is left at the end of the string, and is only omitted on the last line of the file if the file doesn't end in a newline.
- For reading lines from a file, you can loop over the file object.
- This is memoryefficient, fast, and leads to simple code:

# Reading and writing files

- **f.write(string):** writes the contents of string to the file, returning the number of characters written.
- Other types of objects need to be converted – either to a string (in text mode) or a bytes object (in binary mode) –before writing them.
- **f.tell():** It returns an integer giving the file object's current position in the file represented as number of bytes from the beginning of the file when in binary mode and an opaque number when in text mode.
- **f.seek():** To change the file object's position.  
`f.seek(offset, from_what)`

# Refer this examples

## 5.1.1 File example

- <https://github.com/TopsCode/Python/blob/master/Module -5/5.1%20File%20-%20input%20output/5.1.1%20FileDemo.py>

## 5.2.1 Create folder

- <https://github.com/TopsCode/Python/blob/master/Module-5/5.2%20Folder/5.2.1%20create%20directory.py>

## 5.2.2 delete folder

- <https://github.com/TopsCode/Python/blob/master/Module-5/5.2%20Folder/5.2.2%20delete%20directory.py>

# Exception

- In python , there are two distinguishable kinds of errors: syntax errors and exceptions.
- Syntax errors, also known as parsing errors
- Eg: for i in range(1,10)  
      print(i)  
Here missing ":" after for is syntax error.
- Exceptions :Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it.
- Errors detected during execution are called exceptions

# Exception

- Exceptions handling in Python is very similar to Java.
- But whereas in Java exceptions are caught by catch clauses, we have statements introduced by an "except" keyword in Python.
- Eg :

```
n = int(input("Please enter a number: "))
```

Please enter a number: 23.50  
Exception occurs like  

```
ValueError: invalid literal for int() with base 10:'23.5'
```

# Exceptclause

- A try statement may have more than one except clause for different exceptions.
- But at most one except clause will be executed.

## 6.1.1 Exception example 1

- <https://github.com/TopsCode/Python/blob/master/Module-6/6.1%20Exception%20Handling/6.1.1%20ExceptionDemo.py>

## 6.1.2 Exception example 2

- <https://github.com/TopsCode/Python/blob/master/Module-6/6.1%20Exception%20Handling/6.1.2%20ExceptionDemo2.py>

# Try.....finally clause

- try statement had always been paired with except clauses. But there is another way to use it as well.
- The try statement can be followed by a finallyclause.
- Finally clauses are called clean- up or termination clauses, because they must be executed under all circumstances, i.e. a "finally" clause is always executed regardless if an exception occurred in a try block or not.

## 6.2.1 Try finally use

- <https://github.com/TopsCode/Python/blob/master/Module-6/6.2%20Try%20Finally%20clause/6.2.1%20Try%20finally%20use.py>

# Userdefined Exception

- Python also allows you to create your own exceptions by deriving classes from the standard built-in exceptions.

```
class  
    MyNewError(Except  
        ion): pass  
  
    raise MyNewError("Something happened in my program")
```

# Class And Object

- Python classes provide all the standard features of Object Oriented Programming: the class inheritance mechanism allows multiple base classes, a derived class can override any methods of its base class or classes, and a method can call the method of a base class with the same name.
- The class definition looks like : class ClassName:

Statement 1

Statement 2

.....

State

ment

N

# Class And Object

- The statements inside a class definition will usually be function definitions, but other statements are also allowed.
- When a class definition is entered, a new namespace is created, and used as the local scope—thus, all assignments to local variables go into this new namespace.
- In particular, function definitions bind the name of the new function here.

# Member methods in class

- The class\_suite consists of all the component statements defining class members, data attributes and functions.
- The class attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- Eg. `displayDetails()`

# Staticmethods in Python

- Static methods in Python are similar to those found in Java or C++.
- A static method does not receive an implicit first argument.
- To declare a static method, use this idiom:

```
class C:  
    def f(arg1, arg2,  
        ...):.... f =  
        staticmethod(f)
```

- `staticmethod(function)` -> method
- Convert a function to be a staticmethod.
- It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`).
- The instance is ignored except for its class.

# Object

- Class objects support two kinds of operations: attributereferences and instantiation.
- Attribute references use the standard syntax used for all attribute references in Python: obj.name
- Valid attribute names are all the names that were in the class's namespace when the class object was created..

# Object

- Class objects support two kinds of operations: attributereferences and instantiation.
- Class instantiation uses function notation.
- Just pretend that the class object is a parameterless function that returns a new instance of the class.

## 7.1.1 class and object example1

- <https://github.com/TopsCode/Python/blob/master/Module-7/7.1%20class%20and%20object/7.1.1%20ClassDemo.py>

## 7.1.2 class and object example2

- <https://github.com/TopsCode/Python/blob/master/Module-7/7.1%20class%20and%20object/7.1.2%20ClassObjectDemo.py>

# Inheritance

- Python supports inheritance, it even supports multiple inheritance.
- Classes can inherit from other classes.
- A class can inherit attributes and behaviour methods from another class, called the superclass.
- A class which inherits from a superclass is called a subclass, also called heir class or child class.
- Superclasses are sometimes called ancestors as well.

## 7.2.1 Inheritance

- [https://github.com/TopsCode/Python/blob/  
master/Module-  
7/7.2%20Inheritance/7.2.1%20InheritenceDem  
o.py](https://github.com/TopsCode/Python/blob/master/Module-7/7.2%20Inheritance/7.2.1%20InheritenceDemo.py)

# Overloading

- Python supports operator and function overloading.
- **Method overloading**
  - Overloading is the ability to define the same method, with the same name but with a different number of arguments and types.
  - It's the ability of one function to perform different tasks, depending on the number of parameters or the types of the parameters.
  - Python operators work for built-in classes.
  - But same operator behaves differently with different types.

# Overloading

- For example, the + operator will, perform arithmetic addition on two numbers, merge two lists and concatenate two strings.
- This feature in Python, that allows same operator to have different meaning according to the context is called operator overloading.

## 7.3.1 Operator overloading example1

- <https://github.com/TopsCode/Python/blob/master/Module-7/7.3%20Overloading/7.3.1%20Operator%20overloading.py>

### **7.3.2 Operator overloading example2**

- <https://github.com/TopsCode/Python/blob/master/Module-7/7.3%20Overloading/7.3.2%20Operator%20overloading2.py>

### **7.3.3 Operator overloading example3**

- <https://github.com/TopsCode/Python/blob/master/Module-7/7.3%20Overloading/7.3.2%20Operator%20overloading3.py>

# Module 8

## DA- Working with NumPy (python)

## 8.1 Difference Between EDA, AI, ML, and DL

**Exploratory data analysis (EDA)** : EDA is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

**Artificial Intelligence (AI)**: Developing machines to mimic human intelligence and behavior.

**Machine Learning (ML)**: Algorithms that learn from structured data to predict outputs and discover patterns in that data.

**Deep Learning (DL)**: Algorithms based on highly complex neural networks that mimic the way a human brain works to detect patterns in large unstructured data sets

## What is Data Analysis ?

Data analysis is the process of inspecting, cleansing, transforming, and modeling data to uncover useful information, inform conclusions, and support decision-making. It involves examining data sets to identify patterns, trends, relationships, and anomalies that can provide insights into various phenomena or help solve problems.

Data analysis can be conducted using a variety of techniques, including statistical analysis, machine learning, data mining, and visualization. The goal of data analysis is to extract actionable insights from raw data, which can be used to make informed decisions, optimize processes, improve performance, or gain a better understanding of a particular subject or domain.

## The Differences between EDA (Exploratory Data Analysis), AI (Artificial Intelligence), ML (Machine Learning), and DL (Deep Learning):

### ➤ Exploratory Data Analysis (EDA):

**Definition:** EDA is the process of analyzing data sets to summarize their main characteristics, often with visual methods.

**Purpose:** EDA helps understand the data, identify patterns, detect outliers, and formulate hypotheses for further analysis.

**Methods:** Techniques such as summary statistics, data visualization (e.g., histograms, scatter plots), and dimensionality reduction are commonly used in EDA.

➤ **Artificial Intelligence (AI):**

**Definition:** AI refers to the simulation of human intelligence in machines, enabling them to perform tasks that typically require human intelligence, such as learning, reasoning, problem-solving, perception, and language understanding.

**Scope:** AI encompasses various subfields, including machine learning, natural language processing, computer vision, robotics, and expert systems.

**Applications:** AI is applied across different domains, including healthcare, finance, transportation, and entertainment, to automate tasks, improve decision-making, and enhance user experiences.

## ➤ Machine Learning (ML):

**Definition:** ML is a subset of AI that focuses on the development of algorithms and models that enable computers to learn from and make predictions or decisions based on data, without being explicitly programmed.

**Types:** ML algorithms can be categorized into supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning, depending on the nature of the learning process and the availability of labeled data.

**Examples:** ML is used for various tasks, including classification, regression, clustering, anomaly detection, and recommendation systems.

➤ **Deep Learning (DL):**

**Definition:** DL is a subfield of ML that uses neural networks with many layers (deep architectures) to learn representations of data, typically achieving higher performance than traditional ML techniques in tasks such as image recognition, natural language processing, and speech recognition.

**Characteristics:** DL models can automatically discover hierarchical patterns and features from raw data, without the need for manual feature extraction, making them well-suited for processing large, complex datasets.

**Examples:** DL has been successfully applied in areas such as computer vision (e.g., image classification, object detection), natural language processing (e.g., sentiment analysis, machine translation), and speech recognition.

## 8.1 What is NumPy ?

NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements.

**Travis Oliphant** created NumPy package in 2005



It is an extension module of Python which is mostly written in C.

It provides various functions which are capable of performing the numeric computations with a high speed.

NumPy provides various powerful data structures, implementing multi-dimensional arrays and matrices. These data structures are used for the optimal computations regarding arrays and matrices.

## 8.2. Math Refresher

### Vectors and Arrays:

- In mathematics, vectors are quantities that have magnitude and direction. In NumPy, vectors are represented as one-dimensional arrays.
- Arrays in NumPy are data structures that store elements of the same data type in a contiguous block of memory. They can have one or more dimensions.
- NumPy arrays allow efficient storage and manipulation of large datasets, making them ideal for mathematical operations.
- [https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_8\\_Numpy%20with%20Python/numpy\\_8.2.\\_Math\\_Refresher.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_8_Numpy%20with%20Python/numpy_8.2._Math_Refresher.ipynb)

### 8.3 Array Creation :

➤ Learn to create 1D arrays :

To create a numerical array you can pass the array function a list of values. we passed array a list of values and created a NumPy ndarray holding those values.

The array function will automatically try and identify the number's type and as we passed a list of integers it has set the array's dtype attribute to int64 (which stands for a 64 bit integer).

If any of the numbers were a float the type for all numbers would be a float:

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_8\\_Numpy%20with%20Python/Numpy\\_8.3Numpy\\_creation.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_8_Numpy%20with%20Python/Numpy_8.3Numpy_creation.ipynb)

## 8.4 NumPy functions:

1. `zeros()` :This function creates an array filled with zeros.
2. `ones()`: This function creates an array filled with ones.
3. `arange()`:This function creates an array with evenly spaced values within a given interval.
4. `linspace()`: This function creates an array with evenly spaced values over a specified interval. The system implicitly calculates the step size.

## 5. eye() :

This method returns the resultant array with shape NxM with all elements as 0 except the kth diagonal where all elements are 1.

The eye() method takes the following arguments:

- N - Number of rows in the output (can be int)
- M (optional) - Number of columns in the output (can be int)
- k (optional) - the diagonal in question (can be int)
- dtype (optional) - the datatype of the resultant array
- order (optional) - the memory layout of the array (can be 'C' or 'F')

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_8\\_Numpy%20with%20Python/Numpy\\_8.3Numpy\\_creation.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_8_Numpy%20with%20Python/Numpy_8.3Numpy_creation.ipynb)

## 8.5 NumPy Attributes:

- 1) **shape**: This attribute returns a tuple representing the dimensions of the array.  
For example, if you have a 2D array with 3 rows and 4 columns, the shape would be (3, 4).
- 2) **size**: It returns the total number of elements in the array. For instance, if you have a 2D array with a shape of (3, 4), the size would be  $3 * 4 = 12$ .
- 3) **dtype**: This attribute indicates the data type of the elements in the array. It could be int, float, etc., and it also specifies the number of bytes per element.
- 4) **ndim**: This attribute returns the number of array dimensions. For example, a 1D array would have a dimensionality of 1, a 2D array would have a dimensionality of 2, and so on.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%208%20Numpy%20with%20Python/MODULE%208.5%20NUMPY.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%208%20Numpy%20with%20Python/MODULE%208.5%20NUMPY.ipynb)

## 7. Arithmetic Operations:

### 1. Addition:

The add() function sums the content of two arrays, and return the results in a new array.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_8\\_Numpy%20with%20Python/8.7\\_numpy\\_arithmetic\\_operations.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_8_Numpy%20with%20Python/8.7_numpy_arithmetic_operations.ipynb)

### 2. Subtraction:

The subtract() function subtracts the values from one array with the values from another array, and return the results in a new array.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_8\\_Numpy%20with%20Python/8.7\\_numpy\\_arithmetic\\_operations.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_8_Numpy%20with%20Python/8.7_numpy_arithmetic_operations.ipynb)

### 8.7.3 Multiplication

The multiply() function multiplies the values from one array with the values from another array, and return the results in a new array.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_8\\_Numpy%20with%20Python/8.7\\_numpy\\_arithmetic\\_operations.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_8_Numpy%20with%20Python/8.7_numpy_arithmetic_operations.ipynb)

### 8.7.4 Division

The divide() function divides the values from one array with the values from another array, and return the results in a new array.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_8\\_Numpy%20with%20Python/8.7\\_numpy\\_arithmetic\\_operations.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_8_Numpy%20with%20Python/8.7_numpy_arithmetic_operations.ipynb)

## **8.8 Broadcasting and upcasting :**

Broadcasting and upcasting are fundamental concepts in Python, especially when dealing with arrays or tensors in libraries like NumPy.

### **1. Broadcasting:**

Broadcasting allows NumPy to perform arithmetic operations on arrays of different shapes. When operating on two arrays, NumPy compares their shapes element-wise. It starts with the trailing dimensions and works its way forward. Two dimensions are compatible when:

1. They are equal, or
2. One of them is

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_8\\_Numpy%20with%20Python/Module\\_8.8.1\\_Numpy\\_upcasting.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_8_Numpy%20with%20Python/Module_8.8.1_Numpy_upcasting.ipynb)

### 8.8.2 Upcasting:

Upcasting occurs when NumPy promotes the data types of arrays involved in an operation to a common data type that can accommodate all the values without loss of precision.

For example, if you add an array of integers to an array of floats, the integers will be upcasted to floats to maintain precision.

Understanding broadcasting rules and upcasting is crucial for efficiently working with arrays, especially when dealing with multidimensional data or when performing operations involving arrays of different shapes or data types.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_8\\_Numpy%20with%20Python/Module\\_8.8.1\\_Numpy\\_upcasting.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_8_Numpy%20with%20Python/Module_8.8.1_Numpy_upcasting.ipynb)

## 8.9 Comparison Operators as Ufuncs :

Computation on NumPy Arrays: Universal Functions introduced ufuncs, and focused in particular on arithmetic operators.

We saw that using `+`, `-`, `*`, `/`, and other operators on arrays leads to element-wise operations.

NumPy also implements comparison operators such as `<` (less than) and `>` (greater than) as element-wise ufuncs.

The result of these comparison operators is always an array with a Boolean data type.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_8\\_Numpy%20with%20Python/Module\\_8.9\\_Conditional\\_Operators.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_8_Numpy%20with%20Python/Module_8.9_Conditional_Operators.ipynb)

## 8.10 Array Indexing and Slicing:

In NumPy, indexing and slicing are fundamental operations for accessing elements or subarrays within arrays. Here's a brief overview:

**1. Indexing:** Indexing refers to accessing individual elements of an array. In NumPy, indexing starts from 0, similar to Python lists.

**2. Slicing:** Slicing allows you to extract a portion of the array. It follows the format `start:stop:step`. If any of these parameters are not specified, they default to the values that allow the entire array to be sliced.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_8\\_Numpy%20with%20Python/Module\\_8.10\\_Numpy\\_slicing\\_indexing.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_8_Numpy%20with%20Python/Module_8.10_Numpy_slicing_indexing.ipynb)

# **Module 9**

## **DA - Working with Pandas**

## **Pandas :**

Pandas is a powerful and popular open-source Python library for data manipulation and analysis.

It provides easy-to-use data structures and functions that allow users to work efficiently with structured data like tabular, time series, and matrix data. Here's an overview of some key aspects of the Pandas library:

### **➤ Data Structures:**

1. Series: A one-dimensional labeled array capable of holding any data type (e.g., integers, strings, floats, Python objects). It's similar to a Python list or NumPy array, but with added features like index labels.
  
2. Data Frame: A two-dimensional labeled data structure with columns of potentially different types. It's like a spreadsheet or SQL table, and it's the primary data structure for Pandas.

### **Key Features of Pandas :**

1. Data Alignment: Automatically aligns data based on label indices.
2. Handling Missing Data: Provides methods to handle missing data, represented as NaN (Not a Number).
3. Data Operations: Offers a wide range of operations for filtering, selecting, merging, grouping, and reshaping data.
4. Time Series Functionality: Includes specific functionalities for working with time series data.
5. Input/Output: Supports reading from and writing to various file formats like CSV, Excel, SQL databases, and more.

6. Basic Functionality: Indexing and Slicing: Accessing specific rows or columns of data.
7. Data Manipulation: Methods for sorting, filtering, adding, deleting, and modifying data.
8. Aggregation and Grouping: Tools for aggregating data based on specified criteria.
9. Merging and Joining: Combining data from different sources based on common columns or indices.

## 1. Handling missing and categorical data :

Handling missing and categorical data in pandas is crucial for data preprocessing and analysis.

### ➤ Handling Missing Data:

Pandas provides several methods to handle missing data:

#### 1. Removing missing data:

➤ `dropna()`: Drops rows or columns with missing values.

#### 2. Imputing missing data:

➤ `fillna()`: Fills missing values with a specified value.

➤ Imputation techniques like mean, median, mode can be used.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.1\\_handling\\_missing\\_data.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.1_handling_missing_data.ipynb)

### 9.1.2 Missing data

The processing of missing data is one of the most important imperfections in a dataset. Several methods for dealing with missing data are provided by the pandas package in Python, including dropna() and fillna().

- 1) The dropna() method is used to eliminate any columns or rows that have missing values. For instance, the code below will eliminate all rows with at least one missing value:
  
- 2) The fillna() function can be used to fill in missing values with a specific value or method. For example, the following code will fill in missing values in the 'age' column with the mean age of the data:

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.1\\_2\\_missing\\_data.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.1_2_missing_data.ipynb)

### 9.1.3 Handling outliers

Handling outliers is a typical data cleaning activity. Values that diverge greatly from the rest of the data are considered outliers. These factors should be managed carefully since they have a significant influence on a model's performance.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.1.3\\_Outlier.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.1.3_Outlier.ipynb)

## Feature Engineering :

Feature selection and feature engineering are essential components of data cleaning.

The process of choosing only the relevant features in a dataset is referred to as feature selection, whereas the process of building new features from already existing ones is known as feature engineering.

The code below is an illustration of feature engineering:

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.1.3\\_Outlier.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.1.3_Outlier.ipynb)

### **Explanation:**

1. We import pandas as pd, NumPy as np, and relevant modules from scikit-learn.
2. We create a sample Data Frame df with two features 'A' and 'B'. We perform feature engineering by creating two new features 'A squared' and 'Cubed', which are the squares and cubes of the existing features 'A' and 'B', respectively.
3. We split the data into features (X) and the target variable (y).
4. We split the data into training and testing sets using train\_test\_split.
5. We choose a Random Forest Classifier as our model for demonstration purposes.
6. We train the model using the training data.
7. We make predictions on the test data.
8. We evaluate the model's performance using accuracy.

## 9.2 Pandas Series Objects:

`pd.Series(data, index=index)`

where `index` is an optional argument, and `data` can be one of many entities.

For example, `data` can be a list or NumPy array, in which case `index` defaults to an integer sequence:

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.3\\_Pandas\\_Series.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.3_Pandas_Series.ipynb)

### 9.3 Pandas Series

- Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.).

If Series is an analog of a one-dimensional array with explicit indices, a Data Frame is an analog of a two-dimensional array with explicit row and column indices.

Just as you might think of a two-dimensional array as an ordered sequence of aligned one-dimensional columns, you can think of a Data Frame as a sequence of aligned Series objects. Here, by "aligned" we mean that they share the same index.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.3\\_Pandas\\_Series.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.3_Pandas_Series.ipynb)

### 9.3 Pandas Series

- Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.).

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%209%20with%20Python/Module%209.3%20Pandas%20Series.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%209%20with%20Python/Module%209.3%20Pandas%20Series.ipynb)

### 9.4 Model\_Selection

- If Series is an analog of a one-dimensional array with explicit indices, a Data Frame is an analog of a two-dimensional array with explicit row and column indices. Just as you might think of a two-dimensional array as an ordered sequence of aligned one-dimensional columns, you can think of a Data Frame as a sequence of aligned Series objects. Here, by "aligned" we mean that they share the same index.

## 9.5 Mult-indexing in Data Frame

A far more common pattern for handling higher-dimensional data is to make use of hierarchical indexing (also known as multi-indexing) to incorporate multiple index levels within a single index. In this way, higher-dimensional data can be compactly represented within the familiar one-dimensional Series and two-dimensional Data Frame objects.

The direct creation of MultiIndex objects; considerations when indexing, slicing, and computing statistics across multiply indexed data; and useful routines for converting between simple and hierarchically indexed representations of data.

We begin with the standard imports:

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.5\\_Multi\\_Indexing.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.5_Multi_Indexing.ipynb)

## **6. Dropping Level, Transposing:**

In Pandas, dropping levels from a MultiIndex and transposing Data Frame objects are common operations. Let me guide you through both processes:

### **1. Dropping Levels from MultiIndex :**

To drop levels from the MultiIndex, you can use the drop level function:

### **2. Transposing Data Frame :**

Transposing a Data Frame is simply flipping rows and columns. You can do this using the transpose function or simply using .T:

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.6\\_Dropping%20Level%2C%20Transposing.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.6_Dropping%20Level%2C%20Transposing.ipynb)

## 9.7 Accessing Rows, Adding and Removing Columns:

To manipulate a dataset with a URL as a data source, you can use the `pandas.read_csv()` function directly with the URL to load the dataset into a Data Frame.

Here's how you can add a column, delete a column, and perform basic manipulations:

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.7\\_Accessing%20Rows%2C%20Adding%20and%20Removing%20Columns.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.7_Accessing%20Rows%2C%20Adding%20and%20Removing%20Columns.ipynb)

## 9.8 Querying and Sorting Data Frame

Sorting is a fundamental operation in data manipulation and analysis that involves arranging data in a specific order.

Sorting is crucial for tasks such as organizing data for better readability, identifying patterns, making comparisons, and facilitating further analysis.

### ➤ Sort Data Frame in Pandas :

In Pandas, we can use the `sort_values()` function to sort a Data Frame. For example,

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.8%20Querying%20and%20Sorting%20DataFrame.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.8%20Querying%20and%20Sorting%20DataFrame.ipynb)

## 9. Operations on Data frame :

### 1. Rename Column Name :

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.9%20Operations%20on%20DataFrame.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.9%20Operations%20on%20DataFrame.ipynb)

### 2. Create derived column:

It's very common to add new columns using derived data. You just need to assign to a new column:

### 3. Number of NaNs in column :

.isnull() considers both np.NaN and None as null values

Use .isnull() to check which values are null/NaN and then call .sum()

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.9%20Operations%20on%20DataFrame.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.9%20Operations%20on%20DataFrame.ipynb)

## **9.10 Merging and Joining Data Frames:**

One important feature offered by Pandas is its high-performance, in-memory join and merge operations, which you may be familiar with if you have ever worked with databases.

The main interface for this is the pd.merge function, and we'll see a few examples of how this can work in practice.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.10\\_Merging\\_and\\_joining.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.10_Merging_and_joining.ipynb)

## 9.11 Grouping and aggregating

Grouping and aggregating data in Pandas is a fundamental operation for data analysis.

You can group data based on one or more columns and then perform aggregation functions on the grouped data. Here's how you can do it:

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_9\\_Pandas%20with%20Python/Module\\_9.11.Grouping%20and%20Aggregating%20Data.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_9_Pandas%20with%20Python/Module_9.11.Grouping%20and%20Aggregating%20Data.ipynb)

# MODULE 10

**DA – Visualization of Data with Matplotlib and Seaborn (Python)**

## 1. Matplotlib :

Matplotlib is a multiplatform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

It was conceived by John Hunter in 2002, originally as a patch to IPython for enabling interactive MATLAB-style plotting via gnuplot from the IPython command line.

Just as we use the np shorthand for NumPy and the pd shorthand for Pandas, we will use some standard shorthand for Matplotlib imports:

- `import matplotlib as mpl`
- `import matplotlib.pyplot as plt`

## 10.1 Simple Line Plots

Perhaps the simplest of all plots is the visualization of a single function .

Here we will take a first look at creating a simple plot of this type. As in all the following chapters, we'll start by setting up the notebook for plotting and importing the packages we will use:

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_10\\_Matplotlib%20%26%20Seaborn/Module\\_10.1\\_Matplot\\_Simple\\_line.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_10_Matplotlib%20%26%20Seaborn/Module_10.1_Matplot_Simple_line.ipynb)

## 10.2. Visualizing Relationships between Variables using Scatter Plots:

Another commonly used plot type is the simple scatter plot, a close cousin of the line plot.

Instead of points being joined by line segments, here the points are represented individually with a dot, circle, or other shape.

We'll start by setting up the notebook for plotting and importing the packages we will use:

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_10\\_Matplotlib%20%26%20Seaborn/Module\\_10.2\\_Visualizing%20Relationships%20between%20Variables%20using%20Scatter%20Plots.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_10_Matplotlib%20%26%20Seaborn/Module_10.2_Visualizing%20Relationships%20between%20Variables%20using%20Scatter%20Plots.ipynb)

### 10.3 Studying Distributions of Variables using Histograms & Bar Charts

- A simple histogram can be a great first step in understanding a dataset. Earlier, we saw a preview of Matplotlib's histogram function (discussed in Comparisons, Masks, and Boolean Logic), which creates a basic histogram in one line, once the normal boilerplate imports are done (see the following figure):
- The plt.hist docstring has more information on other available customization options. I find this combination of histtype='stepfilled' along with some transparency alpha to be helpful when comparing histograms of several distributions (see the following figure):

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module%2010%20Seaborn/04.05-Histogram-and-Bining.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module%2010%20Seaborn/04.05-Histogram-and-Bining.ipynb)

## 10.4 Visualizing Two-Dimensional Data using Heatmaps :

The Heatmap is just a colored representation of a matrix. However, heatmap has a very large use case. We can use heatmaps for the following purpose.

It is used to see the correlation between columns of a dataset where we can use a darker color for columns having a high correlation.

We can also use heatmaps for plotting various time series and finance-related data where the Y-axis will be the month and X-axis will be the year and the element of the heatmap will be our data.

We can also use the Seaborn library to plot heatmaps even plotting heatmaps using Seaborn is comparatively easier than the matplotlib library. To plot the heatmap using Seaborn we will use `sns.heatmap()` function from the Seaborn library.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_10\\_Matplotlib%20%26%20Seaborn/Module\\_10.4\\_Heat\\_Map.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_10_Matplotlib%20%26%20Seaborn/Module_10.4_Heat_Map.ipynb)

## 10.5 Box Plots and Violin Plots for Visualizing Distributions :

- Violin Plot is a method to visualize the distribution of numerical data of different variables. It is quite similar to Box Plot but with a rotated plot on each side, giving more information about the density estimate on the y-axis.
- The density is mirrored and flipped over, and the resulting shape is filled in, creating an image resembling a violin. The advantage of a violin plot is that it can show nuances in the distribution that aren't perceptible in a boxplot. On the other hand, the boxplot more clearly shows the outliers in the data.
- Violin Plots hold more information than box plots, they are less popular.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_10\\_Matplotlib%20%26%20Seaborn/Module\\_10.5\\_Violin\\_Plot.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_10_Matplotlib%20%26%20Seaborn/Module_10.5_Violin_Plot.ipynb)

## 10.6 Pair Plots and Joint Plots for Exploring Relationships :

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn helps resolve the two major issues while working with Matplotlib:

- Default Matplotlib parameters
- Working with data frames

### ▪ **Seaborn.pairplot() :**

To plot multiple pairwise bivariate distributions in a dataset, you can use the .pairplot() function.

The diagonal plots are the univariate plots, and this displays the relationship for the (n, 2) combination of variables in a Data Frame as a matrix of plots.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_10\\_Matplotlib%20%26%20Seaborn/10.6%20E2%80%A2%20Pair%20Plots%20and%20Joint%20Plots%20for%20Exploring%20Relationships%20.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_10_Matplotlib%20%26%20Seaborn/10.6%20E2%80%A2%20Pair%20Plots%20and%20Joint%20Plots%20for%20Exploring%20Relationships%20.ipynb)

## 10.7 Exploratory Data Analysis (EDA) on a Dataset

Basic example of performing Exploratory Data Analysis (EDA) on a dataset using Python with Pandas, Matplotlib, and Seaborn libraries:

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_10\\_Matplotlib%20%26%20Seaborn/Module\\_10.7\\_EDA.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_10_Matplotlib%20%26%20Seaborn/Module_10.7_EDA.ipynb)

Also this

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_10\\_Matplotlib%20%26%20Seaborn/visualization\\_categorical.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_10_Matplotlib%20%26%20Seaborn/visualization_categorical.ipynb)

# MODULE 11

## Web Scrapping

## 11.1 What is Web Scraping?

Web scraping is an automated method used to extract large amounts of data from websites. This data is often unstructured HTML data which is then converted into a structured format for analysis or use in other applications.

Web scraping is a technique used to collect content and data from the internet.

This data is usually saved in a local file so that it can be manipulated and analyzed as needed.

If you've ever copied and pasted content from a website into an Excel spreadsheet, this is essentially what web scraping is, but on a very small scale.

## Overview of web scraping techniques and tools.

### Techniques for Web Scraping :

#### 1. Manual Scraping :

Description: Manually copying and pasting data from a website.

Use Case: Suitable for small-scale data extraction tasks.

Limitations: Time-consuming and not feasible for large datasets.

#### 2. HTTP Requests :

Description: Sending HTTP requests to fetch web page content.

Example Tool: Requests library in Python.

Use Case: Ideal for static web pages where content doesn't change dynamically.

## 11.2 Introduction to Data Scraping

### **Definition and Significance of Data Scraping:**

- Data scraping, also known as web scraping, refers to the automated process of extracting information from websites. This process involves fetching the web content, parsing the data, and converting it into a structured format that can be easily analyzed or utilized.
- The significance of data scraping lies in its ability to gather vast amounts of data from the web quickly and efficiently, which can then be used for various purposes.

### **11.3 Why Data Scraping is Important for Gathering Information from the Web:**

#### **1. Efficiency and Speed:**

1. Manually collecting data from websites is time-consuming and prone to errors. Data scraping automates this process, allowing for the rapid gathering of large datasets.

#### **2. Access to Real-Time Data:**

1. Data scraping can be set up to run at regular intervals, providing access to the most current data available on websites. This is particularly useful for monitoring prices, tracking trends, or keeping up with news updates.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_11\\_Scrapping/Module\\_11.3\\_Extracting%20data%20from%20HTML%20web%20pages.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_11_Scrapping/Module_11.3_Extracting%20data%20from%20HTML%20web%20pages.ipynb)

## **Applications Of Web Scrapping :**

### **1. Market Analysis and Competitive Intelligence:**

Businesses use data scraping to gather information about competitors, such as pricing, product availability, and customer reviews. This helps in making informed decisions and staying competitive in the market.

### **2. Research and Academic Purposes:**

Researchers and academics utilize data scraping to collect data for studies and experiments. This can include scraping scientific articles, statistical data, and other relevant information from the web.

### **3. Automating Tedium Tasks:**

Repetitive tasks such as data entry and updating databases can be automated through data scraping, freeing up time and resources for more strategic activities.

➤ In conclusion, data scraping is a powerful tool that enables the efficient and effective gathering of information from the web. Its importance spans various industries and applications, making it an essential practice in the digital age.

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/Module\\_11\\_Scrapping/Module\\_11.6\\_Extracting%20Data%20with%20BeautifulSoup.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_11_Scrapping/Module_11.6_Extracting%20Data%20with%20BeautifulSoup.ipynb)

- Case studies and examples of successful web scraping projects :

[https://github.com/TopsCode/Data\\_Analysis\\_2024/blob/main/  
Module 11 Scrapping/11.7 Scraping using csv.ipynb](https://github.com/TopsCode/Data_Analysis_2024/blob/main/Module_11_Scrapping/11.7_Scraping_using_csv.ipynb)



**TOPS TECHNOLOGIES**

Training | Outsourcing | Placement | Study Abroad

# THANK YOU