



TOPS TECHNOLOGIES

Training | Outsourcing | Placement | Study Abroad

Data Analytics with Python

Modules

Level - 1

- Module 1** Python - SDLC
- Module 2** Python - Fundamentals of python language
- Module 3** Python - Collections, functions and Modules in Python

Level - 2

- Module 2)** Introduction and Getting started with SQL
- Module 3)** Getting started with Excel
- Module 4)** Statistics

Level - 3

- Module 5)** Analyzing Data with Python
- Module 10)** Data Visualization on Tableau



TOPS TECHNOLOGIES
Training | Outsourcing | Placement | Study Abroad

Level 1:

Module - 1 [Python Fundamentals]

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.

Why Python ?

- 1) Designed to be easy to learn and master
 - a) Clean, clear syntax
 - b) Very few keywords
- 2) Highly portable
 - c) Runs almost anywhere – high end servers and workstations, down to windows CE
 - d) Uses machine independent byte-code
- 3) Extensible
 - e) Designed to be extensible using C/C++,
 - f) allowing access to many external libraries

Features of Python

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

Features of Python and Installation

Python Productivity

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

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 Escape sequence

<https://github.com/TopsCode/Python/blob/master/Module1/1.1%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.

Refer This Example :

1.1.4end practical

<https://github.com/TopsCode/Python/blob/master/Module-1/1.1%20Programming%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.

Refer This Example :

1.1.5 sep 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

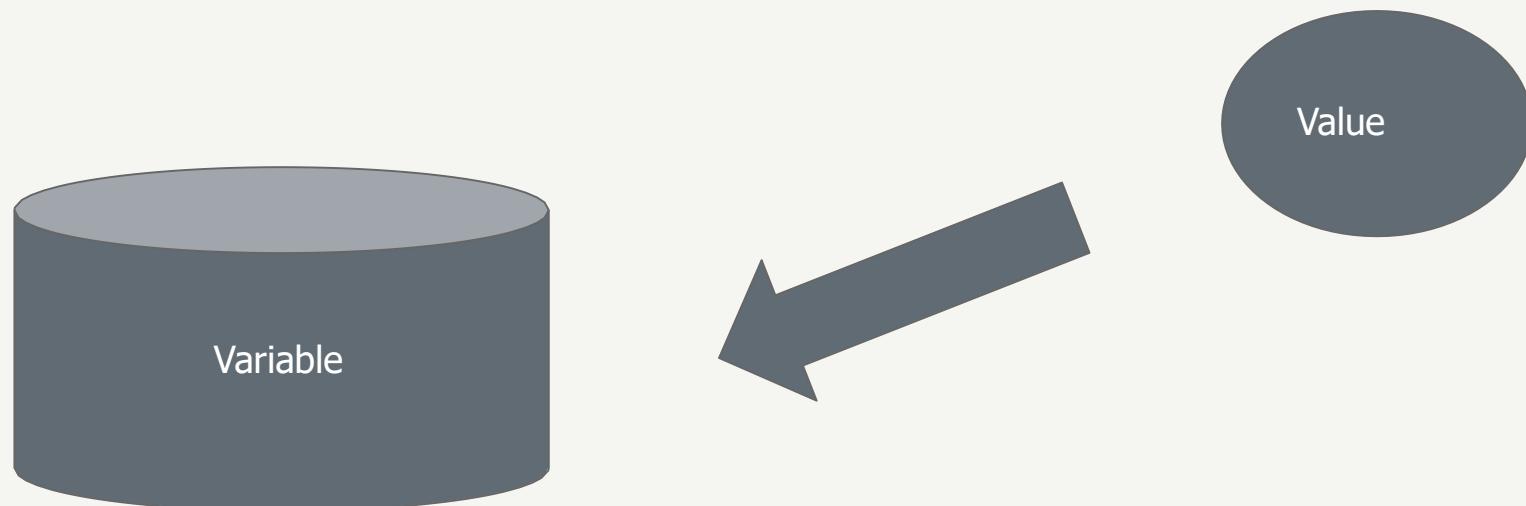
Single line comment : Indicate by #

Document comment : Indicate by """

statements : Indicate by """

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="pytho  
n"  
print(name)
```

Error :NameError: name 'name' is not defined

Refer This Examples :

6.Variable sample

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

7.Sum of two numbers

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

8.Swaping of two numbers

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

Raw String

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 Raw string

<https://github.com/TopsCode/Python/blob/master/Mod1/1.1%20Programming%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 :

1.1.10 type 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 :

1.1.1 Cast

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

12.string input

<https://github.com/TopsCode/Python/blob/master/Module1/1.1%20Programming%20Style/1.1.12%20string%20input.py>

13.int input

<https://github.com/TopsCode/Python/blob/master/Module1/1.1%20Programming%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 :

1.14 split

<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 :A+B

Here, A and B is operand

And A+B is expression

+ is an operator

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>></code>	Greater than
<code><</code>	Less than
<code>>=</code>	Greater than or equal to
	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 TRUE or FALSE as 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.

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.if else statement

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

3.elif statement

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

4.nested if 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

For Loops

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

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

If a sequence contains an expression list, it is evaluated first.

Syntax :

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. for loop

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

2. for loop with list

<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 :

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>

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>

Nested For loop

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)
```

Refer this Example :

5. nested for loop

<https://github.com/TopsCode/Python/blob/master/Module1/1.3%20Looping%20Statement/1.3.5%20nested%20for%20loop.py>

6. pattern 1

<https://github.com/TopsCode/Python/blob/master/Module1/1.3%20Looping%20Statement/1.3.6%20pattern%201.py>

7. pattern 2

<https://github.com/TopsCode/Python/blob/master/Module1/1.3%20Looping%20Statement/1.3.7%20pattern%202.py>

8. pattern 3

<https://github.com/TopsCode/Python/blob/master/Module1/1.3%20Looping%20Statement/1.3.8%20pattern%203.py>

9. pattern 4

<https://github.com/TopsCode/Python/blob/master/Module1/1.3%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 true.

Syntax :

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

1.3.10while loop

<https://github.com/TopsCode/Python/blob/master/Module1/1.3%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/Module1/1.4%20control%20statement/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/Module1/1.4%20control%20statement/1.4.1%20break%20statement.py>

Pass Statements

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/Module1/1.4%20control%20statement/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 code points.

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).

String Functions

- `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 countoccurrences 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 start with 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. String demo

<https://github.com/TopsCode/Python/blob/master/Module-1/1.5%20string/1.5.1%20StringDemo.py>

2. String Operations

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

3. String slicing

<https://github.com/TopsCode/Python/blob/master/Module-1/1.5%20string/1.5.3%20StringSlicing.py>

Level 1:

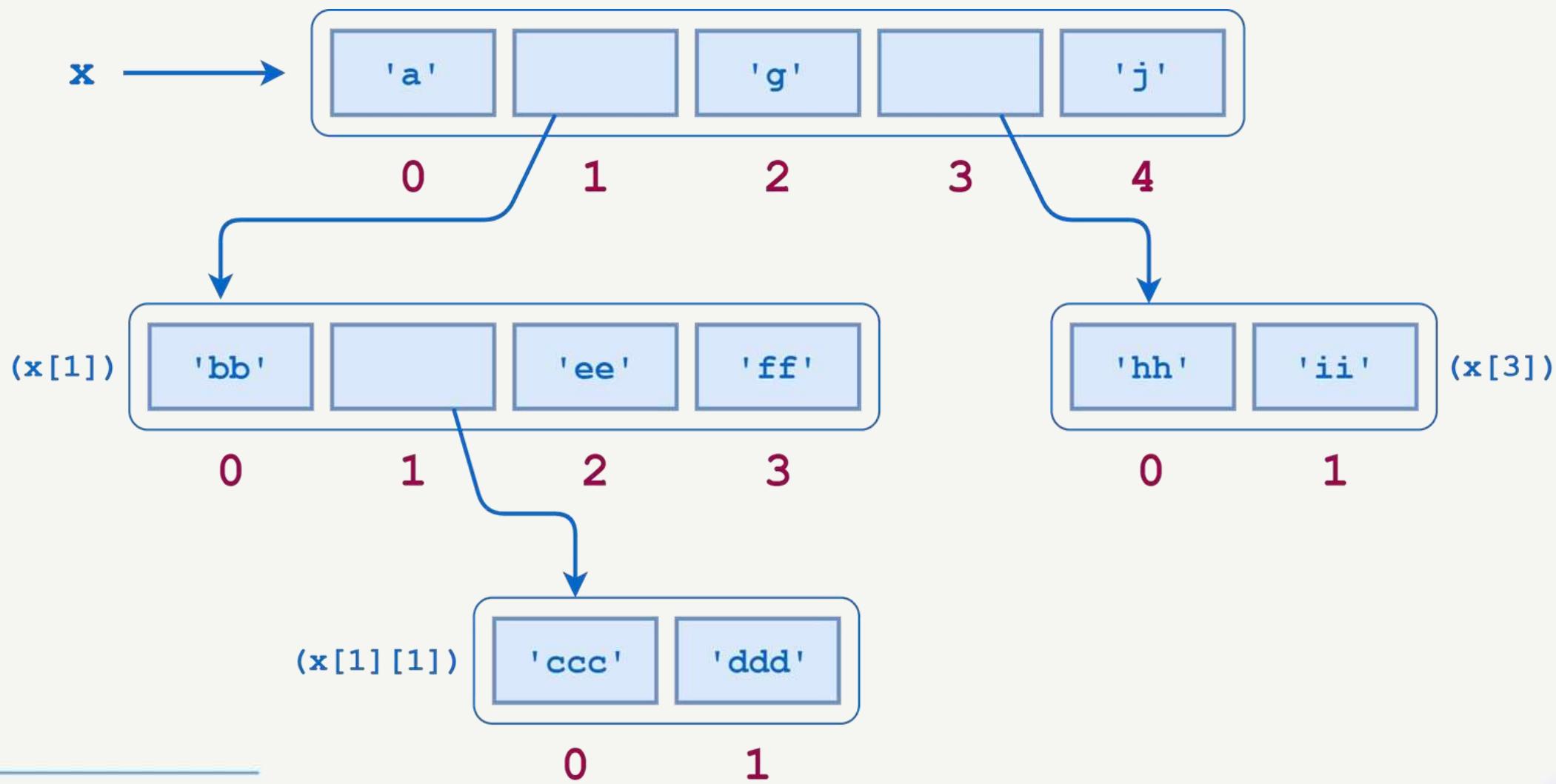
Module - 2 [Collections]

Collections

- List
Operations , Functions and methods
- Tuple
Operations , Functions and methods
- Dictionaries
Operations , Functions and methods
- Set
Operations , Functions and methods

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 hashtable 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-defined classes.
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 square brackets.
- 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(l,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=list[-1])	Removes and returns last object or obj from list.

Functions and Methods

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

- 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 :

1. List as Queue

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

2. List Demo

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

3. List operations

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

4. List pattern

<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 concat ,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 :

1. add item tuple

https://github.com/TopsCode/Python/blob/master/Module2/2.2%20Tuple/2.2.1%20add_item_tuple.py

2. convert list tuple

https://github.com/TopsCode/Python/blob/master/Module2/2.2%20Tuple/2.2.2%20convert_list_tuple.py

3. convert tuple string

https://github.com/TopsCode/Python/blob/master/Module2/2.2%20Tuple/2.2.3%20convert_tuple_string.py

4. create tuple with numbers

https://github.com/TopsCode/Python/blob/master/Module2/2.2%20Tuple/2.2.4%20create_tuple_withnumbers.py

Refer this example :

5. create tuple

https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.5%20create_tuple.py

6. Find repeat item tuple

https://github.com/TopsCode/Python/blob/master/Module-2/2.2%20Tuple/2.2.6%20find_repeat_item_tuple.py

7. slice tuple

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

Method	Description
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 dictionary dict.

Refer this example :

1. Dictionary example

<https://github.com/TopsCode/Python/blob/master/Module-2/2.3%20Dictionaries/2.3.1%20DictionaryDemo.py>

2. Dictionary method demo

<https://github.com/TopsCode/Python/blob/master/Module2/2.3%20Dictionaries/2.3.2%20DictionaryMethodDemo.py>



TOPS TECHNOLOGIES
Training | Outsourcing | Placement | Study Abroad

Level 1:

Module - 3 [Functions]

Content :

- Defining a function
- Calling function
- Types of function
- Anonymous function
- Global and local variables

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

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 other function also.

Syntax :

functionname() or

functionname(argument)

3.1 Create function

<https://github.com/TopsCode/Python/blob/master/Module-3/3.1%20Functions/3.1.1%20create%20function.py>

Types of Functions

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)	positional argument
--------------	---------------------

parrot(voltage=1000)	keyword argument
----------------------	------------------

parrot(voltage=1000000, action='VOOOOOM')	keyword arguments
--	-------------------

parrot(action='VOOOOOM', voltage=1000000)	keyword arguments
--	-------------------

parrot('a million', 'bereft of life', 'jump')	positional arguments
---	----------------------

parrot('a thousand', state='pushing up the daisies')	positional, 1 keyword
---	-----------------------

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 :

2. Function with parameter

<https://github.com/TopsCode/Python/blob/master/Module3/3.1%20Functions/3.1.2%20function%20with%20parameter.py>

3. Function with default value

<https://github.com/TopsCode/Python/blob/master/Module3/3.1%20Functions/3.1.3%20function%20with%20default%20value.py>

4. Function with return values

<https://github.com/TopsCode/Python/blob/master/Module3/3.1%20Functions/3.1.4%20function%20with%20return%20values.py>

5. tuple as parameter

<https://github.com/TopsCode/Python/blob/master/Module3/3.1%20Functions/3.1.5%20tuple%20as%20parameter.py>

6. dict as parameter

<https://github.com/TopsCode/Python/blob/master/Module-3/3.1%20Functions/3.1.6%20dict%20as%20parameter.py>

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/Module3/3.2%20Anonymous%20functions/3.2.1%20lambda%20function.py>

Global & Local variables

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 & Local variables

Local Variables

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

1. global variable

<https://github.com/TopsCode/Python/blob/master/Module-3/3.3%20Global%20Local%20Variables/3.3.1%20global%20variable.py>

2. local variable

<https://github.com/TopsCode/Python/blob/master/Module-3/3.3%20Global%20-%20Local%20Variables/3.3.1%20global%20variable.py>

3. global keyword

<https://github.com/TopsCode/Python/blob/master/Module-3/3.3%20Global%20-%20Local%20Variables/3.3.3%20global%20keyword%20use.py>

Level 1:

Module - 4 [Modules]

Content:

- Importing Module
- Random Module
- Math Module
- Packages
- Composition

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.fib1

0)

fibo.fib2

(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>

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/Module4/4.2%20random%20module/4.2.1%20RandomModulesDemo.py>

Random Functions

Function	Description
<code>random.randrange(start, stop [, step])</code>	Return a randomly selected element from <code>range(start, stop, step)</code> . This is equivalent to <code>choice(range(start, stop, step))</code> , but doesn't actually build a range object.
<code>random.randint(a, b)</code>	Return a random integer N such that $a \leq N \leq b$. Alias for <code>randrange(a, b+1)</code> .
<code>random.choice(seq)</code>	Return a random element from the non-empty sequence <code>seq</code> .
<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 x . <code>ceil ()</code> , which should return an Integral value.
<code>math.factorial(x)</code>	Return x factorial

Math Module

Function	Description
math.floor(x)	Return the floor of x , the largest integer less than or equal to x . If x is not a float, delegates to $x.\text{floor}()$, which should return an Integral value.
math.trunc(x)	Return the Real value x truncated to an Integral (usually an integer).
math.pow(x, y)	Return x raised to the power y .
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>

Packages

Packages are a way of structuring Python's module namespace by using "dotted module names".

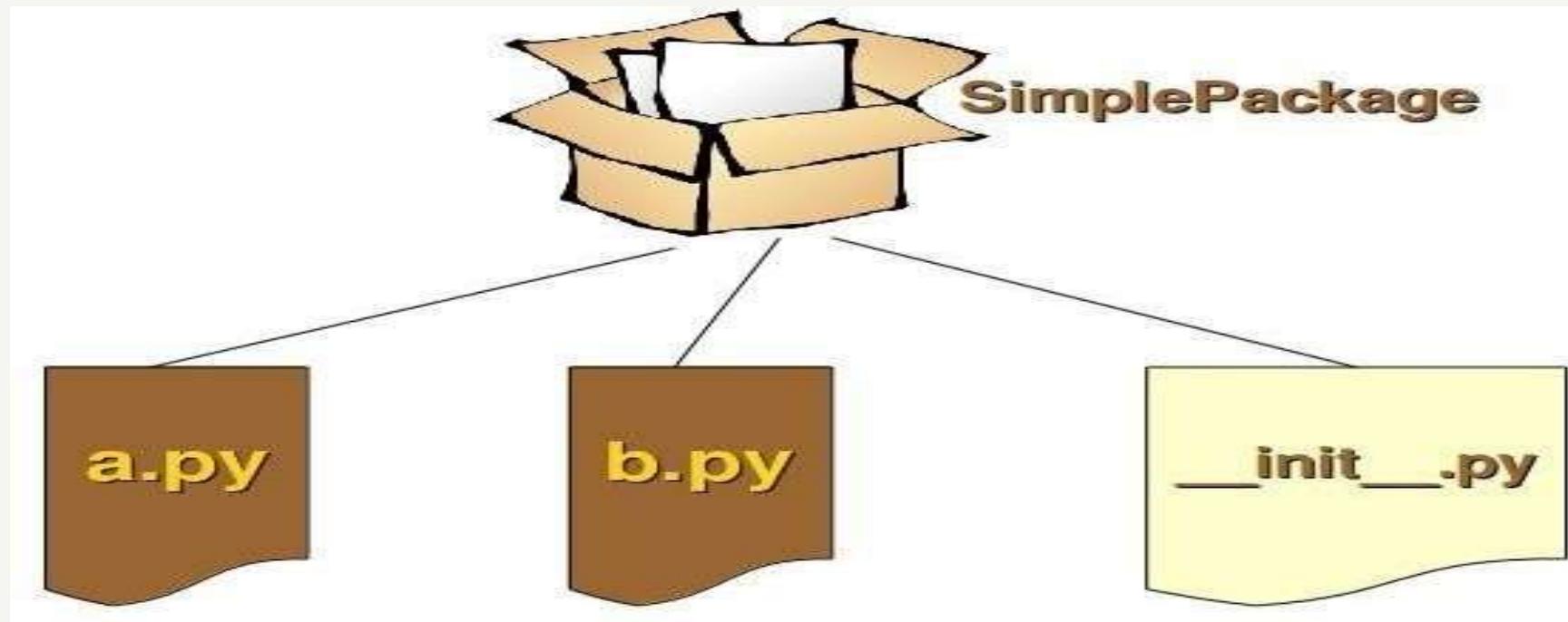
For example, the module name A.B designates a submodule named B in a package named A.

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%20sample.py>

Packages



SQL

SQL

Modules:

- DBMS and RDBMS
- E-R Relational Schema
- Types of database
- Normalization
- Algebra
- Database Programming language SQL
- Keys: Primary Key, Unique Key, Foreign Key
- SQL Statement types: DML, DDL, TQL, TCL
- Joins
- Function
- Transaction Concept (Rollback, Commit, Save Point)

DBMS

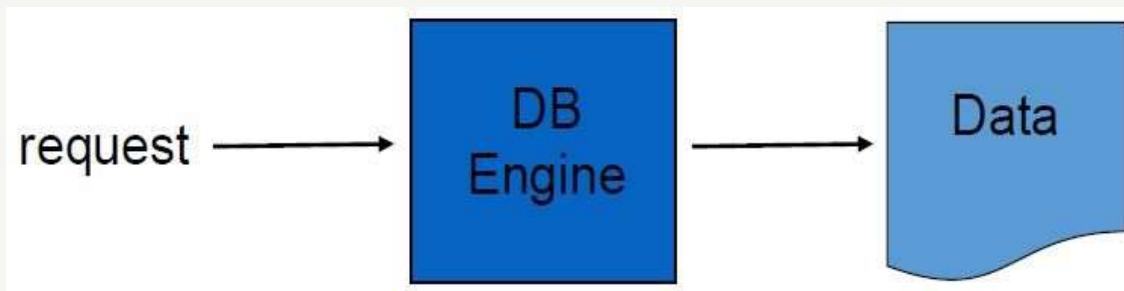
- DBMS stands for Data Base Management System.

Data + Management System

- **Database** is a collection of inter-related data and Management System is a set of programs to store and retrieve those data.
- **DBMS** is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner.
- For Example, university database organizes the data about students, faculty, and admin staff etc. which **helps in efficient retrieval, insertion and deletion of data from it**.

Database Management Systems

- A DBMS consists of 2 main pieces:
 - the data
 - the DB engine
 - the data is typically stored in one or more files



- Two most common types of DBMS are:
 - Local
 - Server

Popular DBMS Software

Here, is the list of some popular DBMS system:

- MySQL
- Microsoft Access
- Oracle
- PostgreSQL
- SQLite
- Mongo DB
- IBM DB2, etc.

What is the need of DBMS?

Database systems are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization: **Storage of data and retrieval of data.**

Storage:

- According to the principles of database systems, the data is stored in such a way that it acquires lot less space as the redundant data (duplicate data) has been removed before storage.

Fast Retrieval of data:

- Along with storing the data in an optimized and systematic manner, it is also important that we retrieve the data quickly when needed. Database systems ensure that the data is retrieved as quickly as possible.

PURPOSE OF DBMS

The main purpose of database systems is to manage the data.

Consider a university that keeps the data of students, teachers, courses, books etc. To manage this data we need to store this data somewhere where we can add new data, delete unused data, update outdated data, retrieve data, to perform these operations on data we need a Database management system that allows us to store the data in such a way so that all these operations can be performed on the data efficiently.

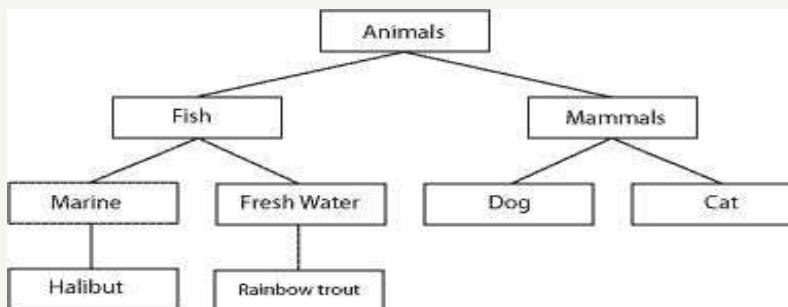
RDBMS

- Stands for "**Relational Database Management System.**"
- An RDBMS is a type of DBMS designed specifically for relational databases.
- A relational database refers to a database that stores data in a structured format, using rows and columns.
- This makes it easy to **locate and access specific values** within the database.
- It is "relational" because the values within each table are related to each other.
- **Tables may also be related to other tables.** The relational structure makes it possible to run queries across multiple tables at once.
- The RDBMS refers to the that executes queries on the data, including adding, updating, and software searching for values.
- An RDBMS may also provide a visual representation of the data. For example, **it may display data in a tables like a spreadsheet, allowing you to view and even edit individual values in the table.**

Types of DBMS ...

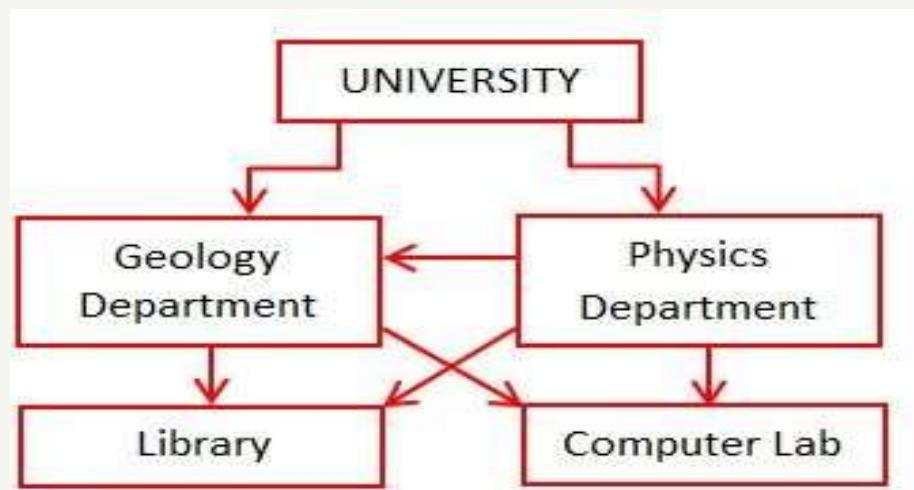
1. Hierarchical databases-

- It is very fast and simple.
- This kind of database model uses a tree-like structure which links a number of dissimilar elements to one primary record –the "owner" or "parent".
- Each record in a hierarchical database contains information about a group of parent child relationships.



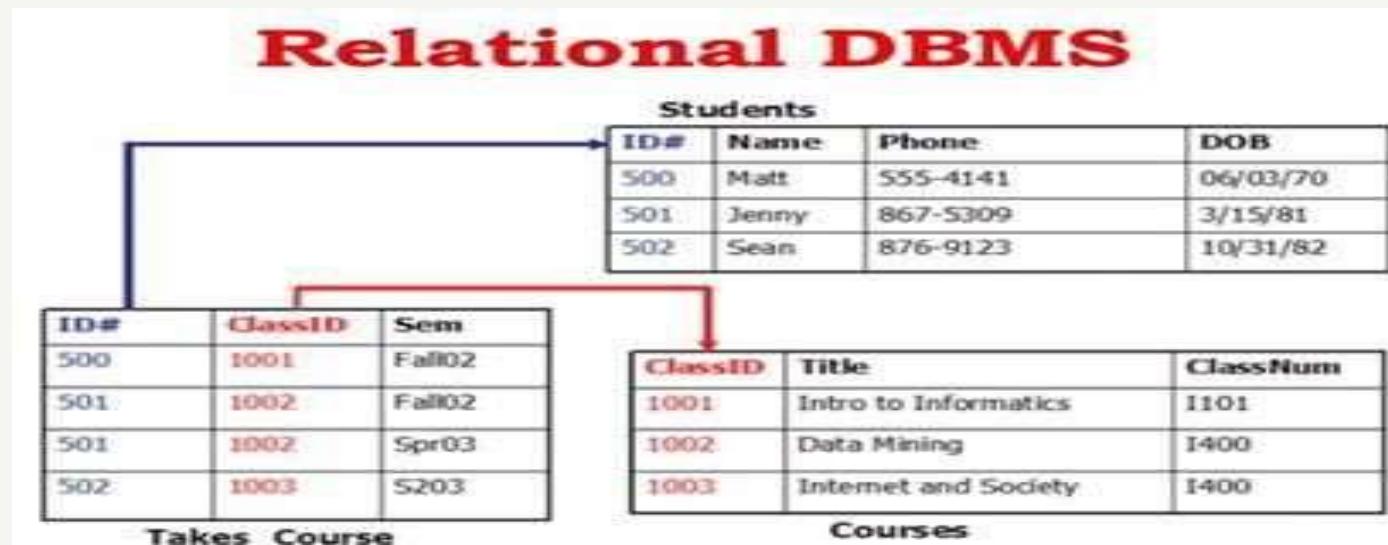
2. Network databases –

- The network database model can be viewed as a net-like form where a single element can point to multiple data elements and can itself be pointed to by multiple data elements.
- The network database **model allows each record to have multiple parents** as well as multiple child records, which can be visualized as a web-like structure of networked records.



3. Relational databases –

- A relational database is one in which data is stored in the form of tables, using **rows and columns**.
- This arrangement makes it easy to locate and access specific data within the database. It is “**relational**” because the data within each table are related to each other.



4. Object-oriented databases -

- The recent development in database technology is the incorporation of the object concept that has become significant in programming languages.
- In object-oriented databases, all data are objects. Objects may be linked to each other by an “is-part-of” relationship to represent larger, composite objects
- Object DBMS's increase the semantics of the C++ and Java



Relational Databases

- RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

- Most of today's databases are relational:
 - database contains 1 or more tables
 - table contains 1 or more records
 - record contains 1 or more fields
 - fields contain the data

- So why is it called "relational"?
 - tables are related (joined) based on common fields

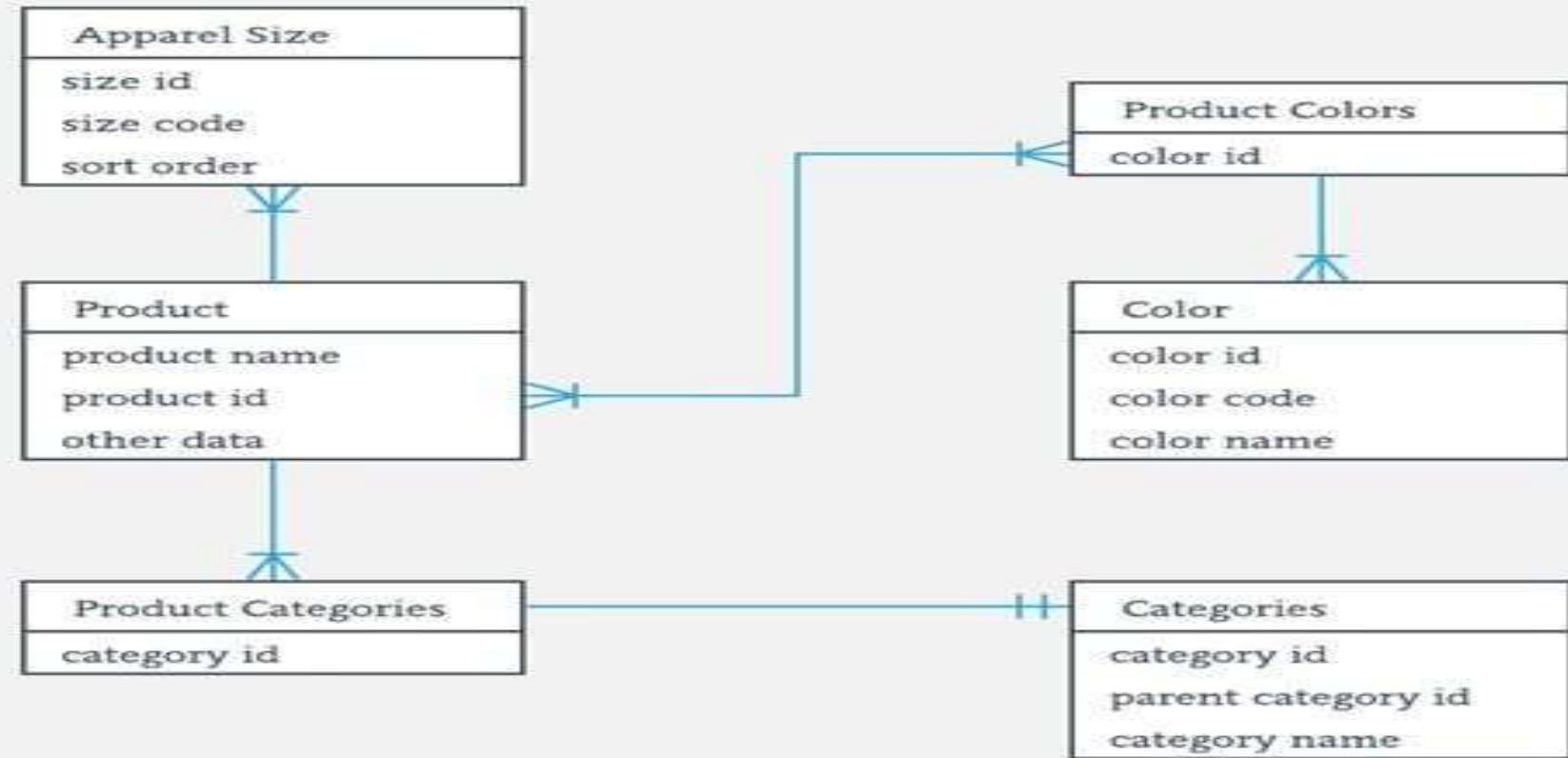
E-R Model ...

- The ER or (Entity Relational Model) is a high-level conceptual data model diagram.
- Entity-Relation model is based on the notion of real-world entities and the relationship between them.
- ER modeling helps you to analyze data requirements systematically to produce a well-designed database.
- So, it is considered a best practice to complete ER modeling before implementing your database.

ER Diagram

- Entity relationship diagram displays the relationships of entity set stored in a database.
- In other words, we can say that ER diagrams help you to explain the logical structure of databases.
- At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.

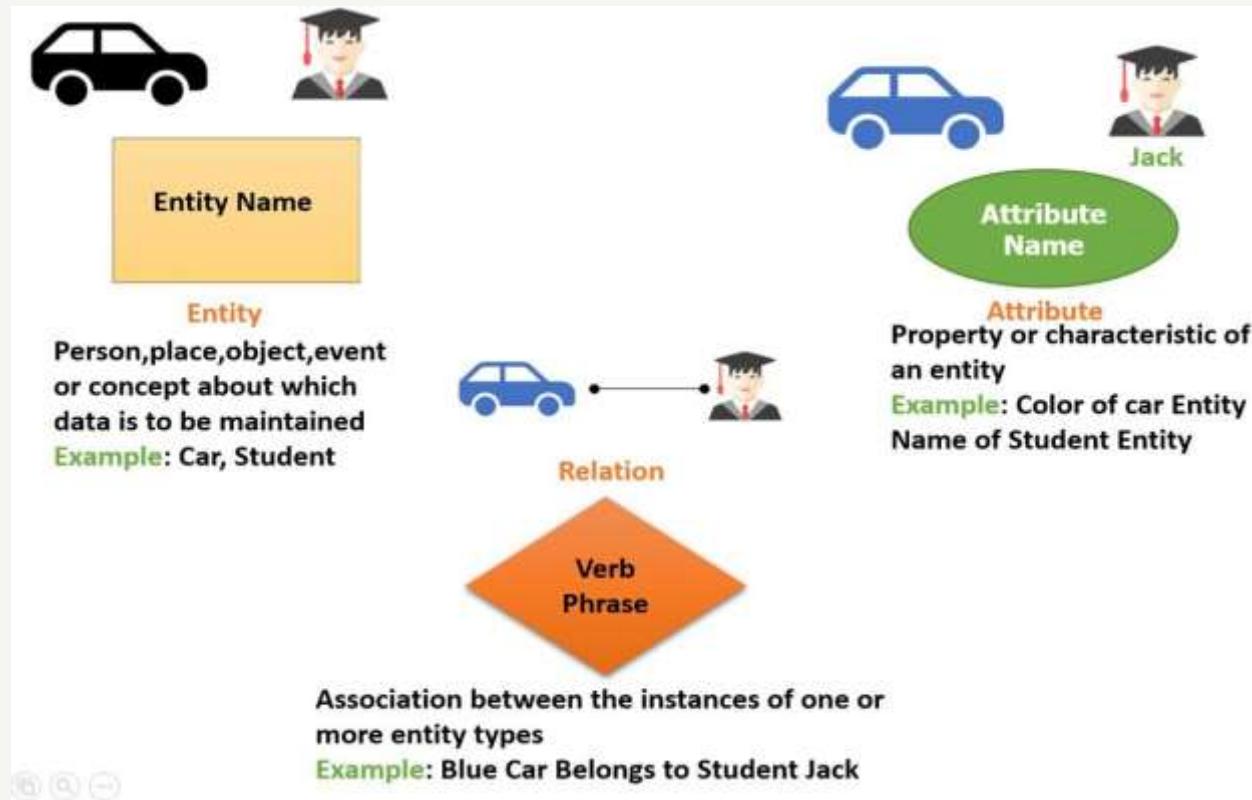
ER Diagram



Components of ER Diagram

- Entities
- Attributes
- Relationships
- For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.

Components of ER Diagram

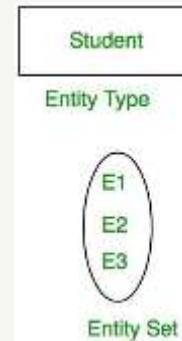


Entity

- A real-world thing either living or non-living that is easily recognizable and non recognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.
- An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some 'attributes' which represent that entity.
- Set of all entity is called **entity set**

Examples of entities:

- **Person:**Employee, Student, Patient
- **Place:**Store, Building



Attribute

Attributes are the properties which define the entity type. For example, Roll_No, Name, DOB, Age, Address, Mobile_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.

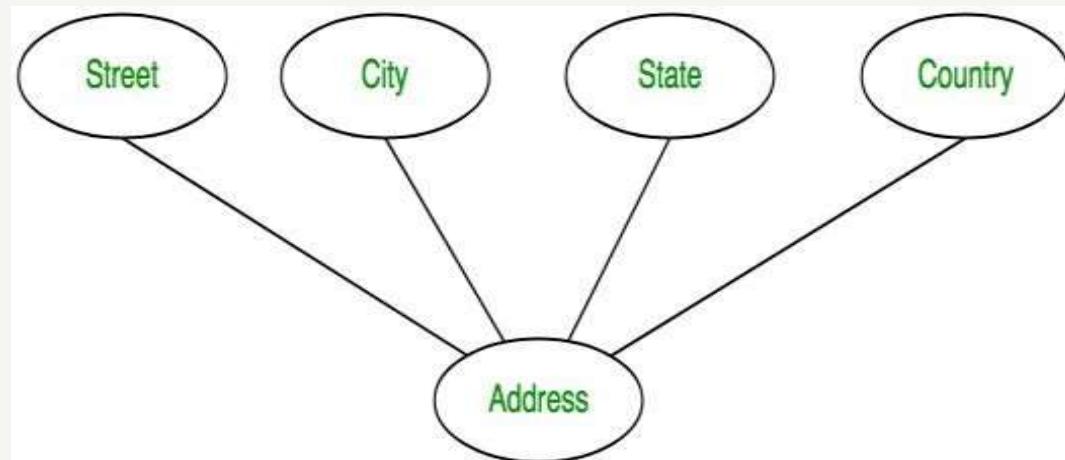
1. Key Attribute:

- The attribute which uniquely identifies each entity in the entity set is called key attribute.
- For example, Roll_No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.



2. Composite Attribute –

An attribute composed of many other attribute is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals



3. Multivalued Attribute –

An attribute consisting more than one value for a given entity. For example, Phone_No(can be more than one for a given student). In ER diagram, multivalued attribute is represented by double oval.

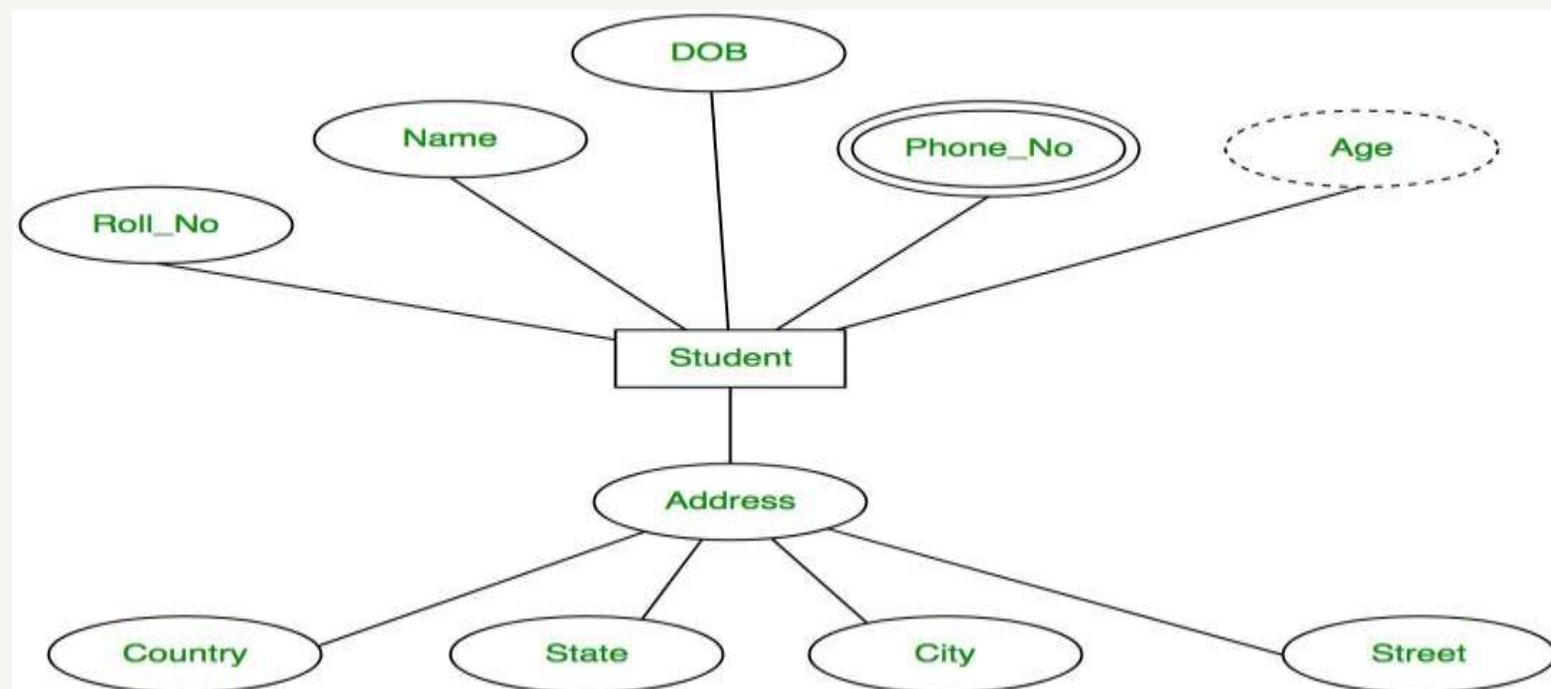


4. Derived Attribute –

An attribute which can be derived from other attributes of the entity type is known as derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, derived attribute is represented by dashed oval.



The complete entity type Student with its attributes can be represented as:

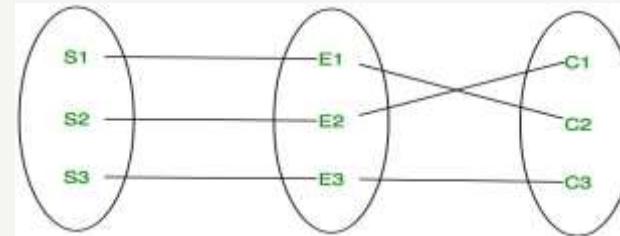


Relationship Type and Relationship Set:

A relationship type represents the association between entity types. For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course. In ER diagram, relationship type is represented by a diamond and connecting the entities with lines.



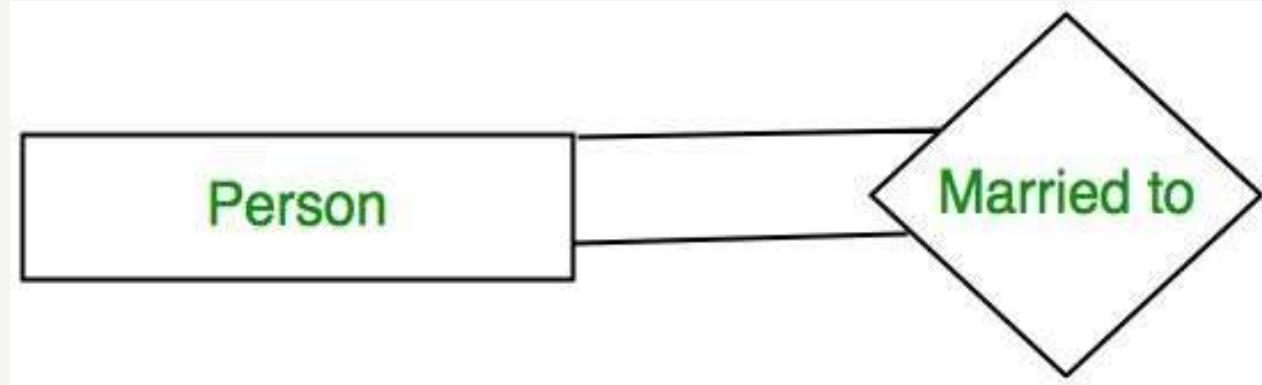
A set of relationships of same type is known as relationship set. The following relationship set depicts S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3.



Degree of a relationship set:

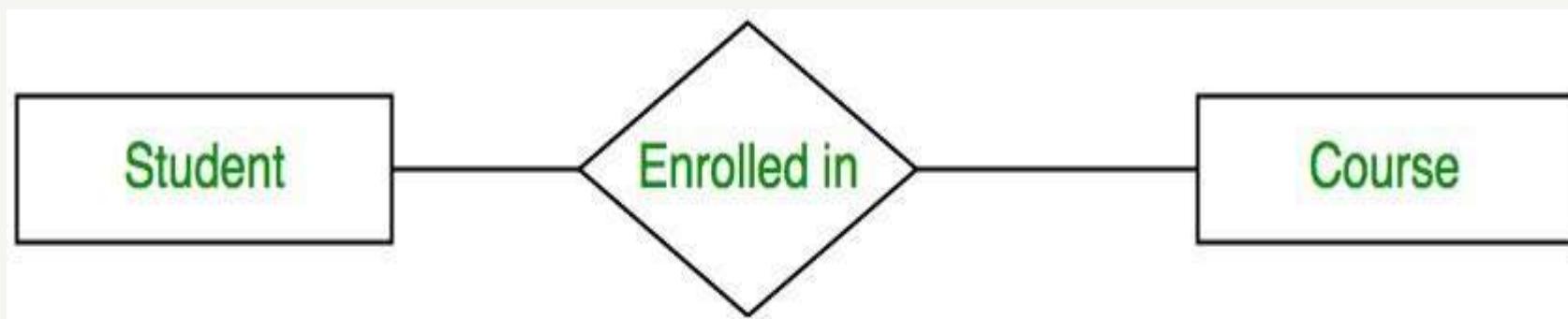
1. Unary Relationship –

When there is only ONE entity set participating in a relation, the relationship is called as unary relationship. For example, one person is married to only one person.



2. Binary Relationship –

When there are TWO entities set participating in a relation, the relationship is called as binary relationship. For example, Student is enrolled in Course.

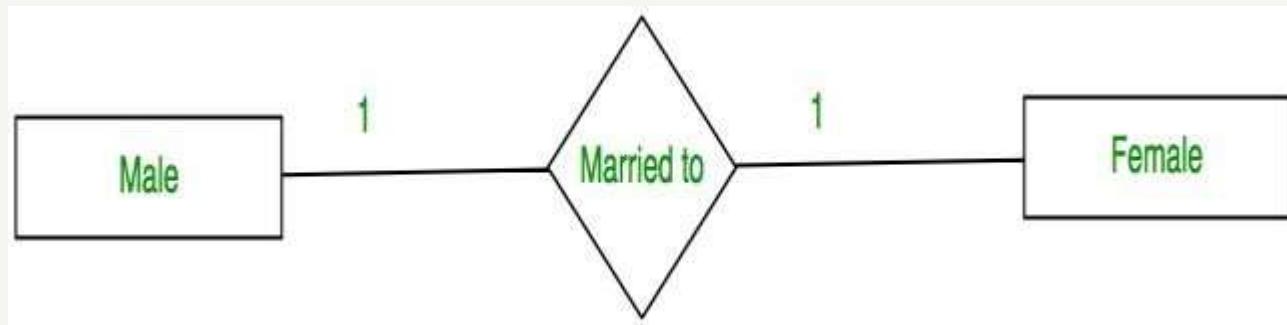


3. N-ary Relationship –

When there are n entities set participating in a relation, the relationship is called as n-ary relationship.

The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types:

One to one – When each entity in each entity set can take part only once in the relationship, the cardinality is one to one. Let us assume that a male can marry to one female and a female can marry to one male. So the relationship will be one to one.



Many to one –When entity set can take part only on entities in once in the relationship set and entities in other entity set can take part more than once in the relationship set, cardinality is many to one.



Many to many –When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.



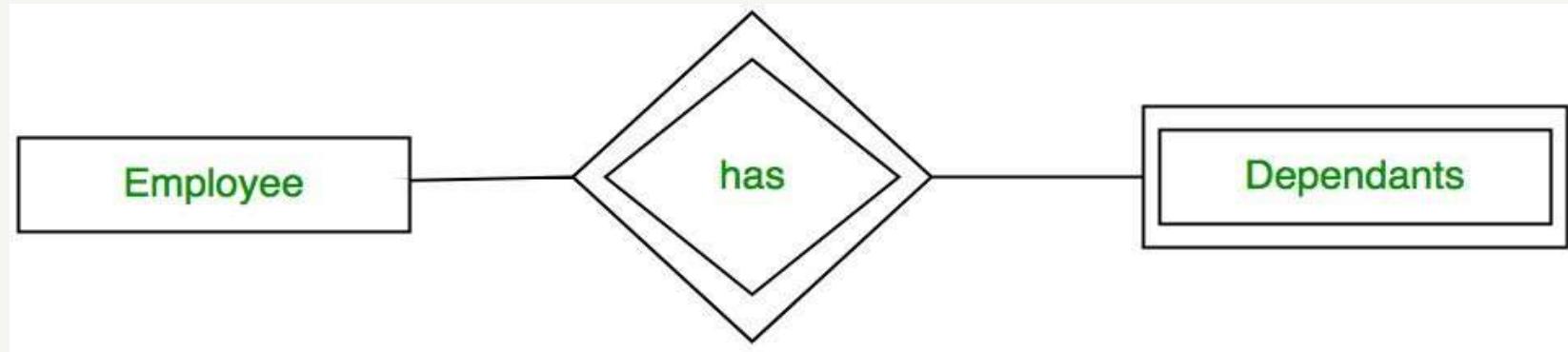
Participation Constraint: Participation Constraint is applied on the entity participating in the relationship set.

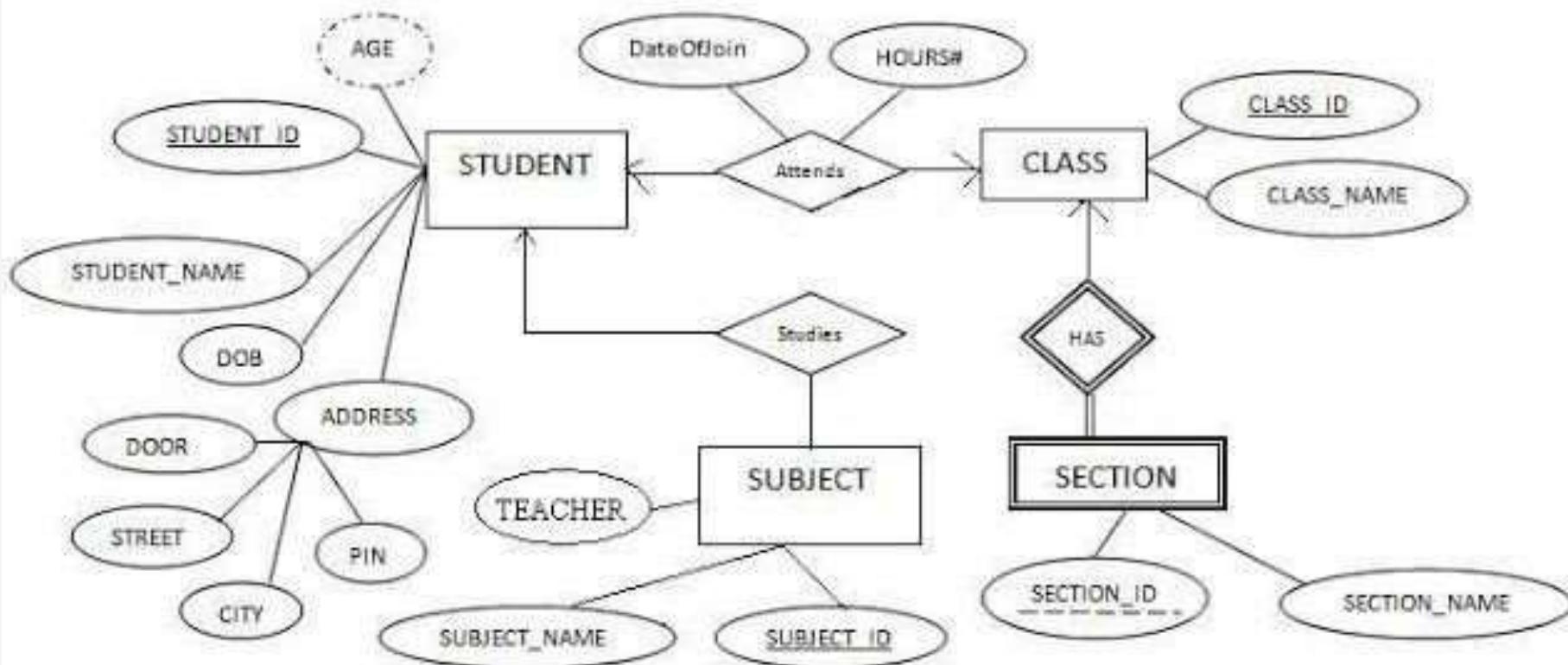
Total Participation –Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of student will be total. Total participation is shown by double line in ER diagram.

Partial Participation –The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the student, the participation of course will be partial. The diagram depicts the ‘Enrolled in’ relationship set with Student Entity set having total participation and Course Entity set having partial participation

Weak Entity Type and Identifying Relationship:

- An entity type has a key attribute which uniquely identifies each entity in the entity set.
- But there exists some entity type for which key attribute can't be defined. These are called Weak Entity type.
- For example, A company may store the information of dependants (Parents, Children, Spouse) of an Employee. But the dependants don't have existence without the employee. So Dependant will be weak entity type and Employee will be Identifying Entitytype for Dependant.





Algebra

Relational Algebra is procedural query language, which takes Relation as input and generate relation as output. Relational algebra mainly provides theoretical foundation for relational databases and SQL.

Unary Relational Operations

- SELECT (symbol: σ)
- PROJECT (symbol: π)
- RENAME (symbol: \circ)

Binary Relational Operations

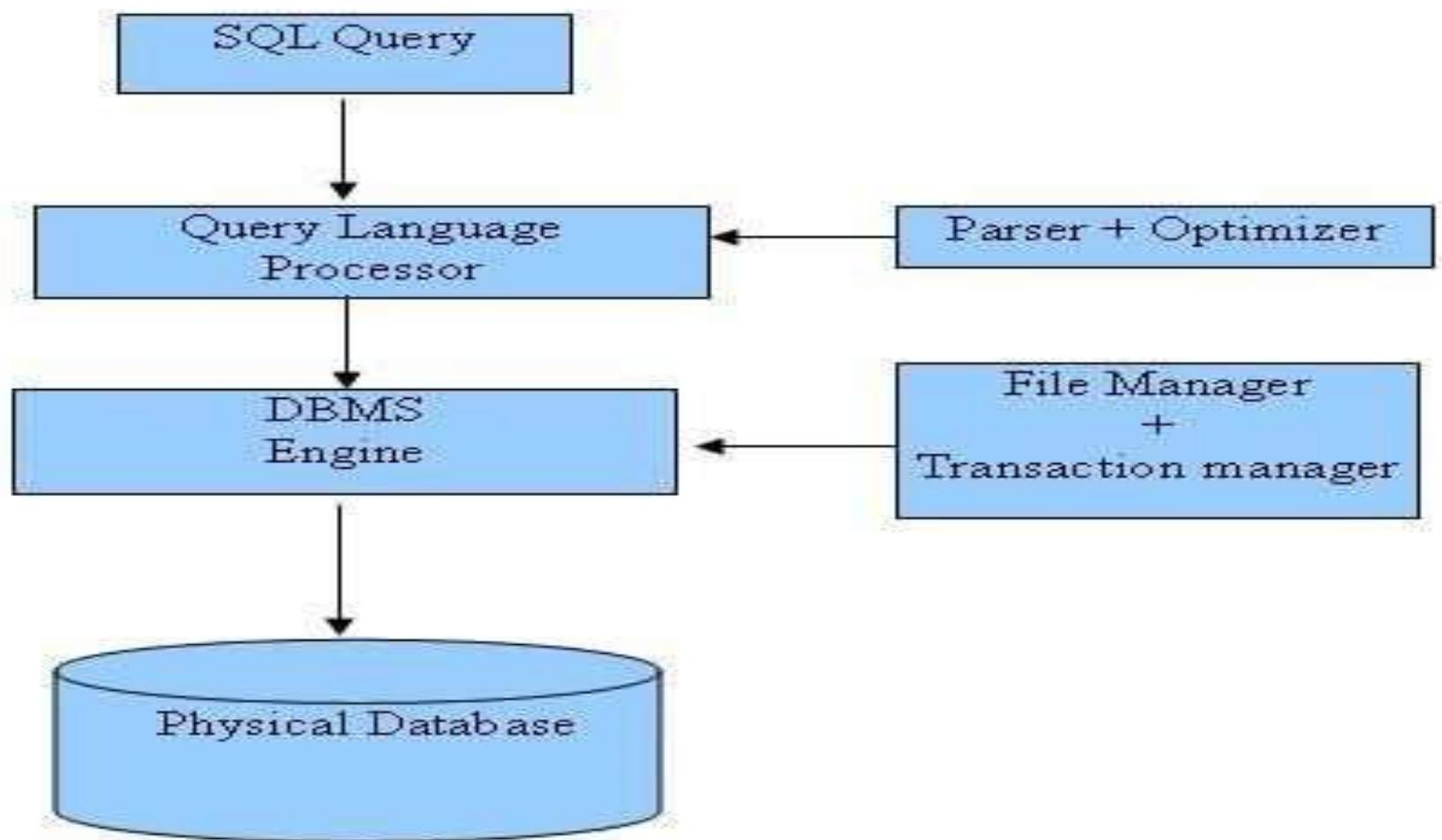
- JOIN
- DIVISION

Relational Algebra Operations From Set Theory

- UNION (\cup)
- INTERSECTION (\cap),
- DIFFERENCE ($-$)
- CARTESIAN PRODUCT (\times)

What is SQL?

- SQL is **Structured Query Language**, which is a computer language for **storing, manipulating and retrieving data** stored in relational database.
- SQL is a language of database, it includes database creation, deletion, fetching rows and modifying rows etc.
- **SQL is the standard language for Relation Database System.** All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, postgres and SQL Server use SQL as standard database language.
- Also, they are using different dialects, such as:
- MS SQL Server using T-SQL, ANSI SQL
- Oracle using PL/SQL
- MS Access version of SQL is called JET SQL (native format) etc



Why SQL?

- Allows users to access data in relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures, and views

What is SQL? (Cont...)

- SQL stands for Structured Query Language
- SQL allows you to access a database
- SQL is an ANSI standard computer language
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert new records in a database
- SQL can delete records from a database
- SQL can update records in a database
- SQL is easy to learn
- SQL is written in the form of queries
- action queries insert, update & delete data
- select queries retrieve data from DB

Keys

Primary Key:

- A primary key is a column of table which uniquely identifies each tuple (row) in that table.
- Primary key enforces integrity constraints to the table.
- **Only one primary key is** allowed to use in a table.
- The primary key does not accept the **any duplicate and NULL values**.
- The primary key value in a table changes very rarely so it is chosen with care where the changes can occur in a seldom manner.
- A primary key of one table can be referenced by foreign key of another table.

Table : Student

Roll_number	Name	Batch	Phone_number	Citizen_ID

Primary key

Keys

Unique Key:

- Unique key constraints also identifies an individual table uniquely in a relation or table.
- A table **can have more than one unique key** unlike primary key.
- Unique key constraints can accept **only one NULL value** for column.
- Unique constraints are also referenced by the foreign key of another table.
- It can be used when someone wants to enforce unique constraints on a column and a group of columns which is not a primary key.

Table : Student

Roll_number	Name	Batch	Phone_number	Citizen_ID

↑
Unique key

Keys

Foreign Key:

- When, "one" table's primary key field is added to a related "many" table in order to create the common field which relates the two tables, it is called a foreign key in the "many" table.
- In the example given below, salary of an employee is stored in salary table. Relation is established via is stored in "Employee" table. To identify the salary of "Jforeign key column "Employee_ID_Ref" which refers "Employee_ID" field in Employee table.of "Jhon" is stored in "Salary" table. But his employee info For example, salary hon", his "employee id" is stored with each salary record.

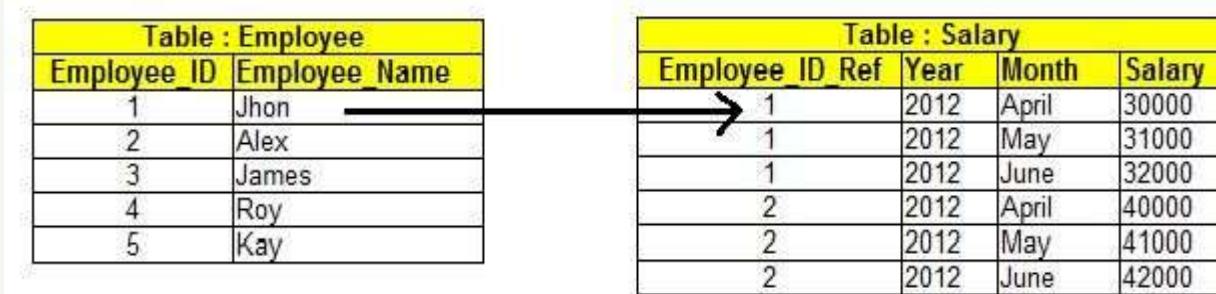


Table : Employee		Table : Salary			
Employee_ID	Employee_Name	Employee_ID_Ref	Year	Month	Salary
1	Jhon	1	2012	April	30000
2	Alex	1	2012	May	31000
3	James	1	2012	June	32000
4	Roy	2	2012	April	40000
5	Kay	2	2012	May	41000
		2	2012	June	42000

Database Normalization ...

- Normalization is the process of minimizing redundancy (duplicity) from a relation or set of relations.
- Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations.

Most Commonly used normal forms:

- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

First Normal Form

- If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute.
- A relation is in first normal form if every attribute in that relation is singled valued attribute.

Example 1 –Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD_PHONE. Its decomposition into 1NF has been shown in table 2.

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

Second Normal Form

- To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency.
- relation is in 2NF if it has No Partial Dependency,i.e.,no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.
- Partial Dependency –If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.
- The table is not in 2nf form

STUD_NO	COURSE_NO	COURSE_FEE
1	C1	1000
2	C2	1500
1	C4	2000
4	C3	1000
4	C1	1000
2	C5	2000

- Note that, there are many courses having the same course fee.
- COURSE_FEE cannot alone decide the value of COURSE_NO or STUD_NO;
- COURSE_FEE together with STUD_NO cannot decide the value of COURSE_NO;
- COURSE_FEE together with COURSE_NO cannot decide the value of STUD_NO;
- Hence, COURSE_FEE would be a non-prime attribute, as it does not belong to the one only candidate key {STUD_NO, COURSE_NO} ;
- But, COURSE_NO \rightarrow COURSE_FEE , i.e., COURSE_FEE is dependent on COURSE_NO,which is a proper subset of the candidate key.
- Non-prime attribute COURSE_FEE is dependent on a proper subset of the candidate key, which is a partial dependency and so this relation is not in 2NF.

To convert the above relation to 2NF,we need to split the table into two tables such as :Table 1: STUD_NO, COURSE_NOMTable 2: COURSE_NO, COURSE_FEE

STUD_NO	COURSE_NO	COURSE_FEE
1	C1	1000
2	C2	1500
1	C4	2000
4	C3	1000
4	C1	1000
2	C5	2000

→

Table 1		Table 2	
STUD_NO	COURSE_NO	COURSE_NO	COURSE_FEE
1	C1	C1	1000
2	C2	C2	1500
1	C4	C3	1000
4	C3	C4	2000
4	C1	C5	2000

Third Normal Form

- A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form. A relation is in 3NF if at least one of the following condition hold in every non-trivial function dependency $X \rightarrow Y$
 - X is a super key.
 - Y is a prime attribute (each element of Y is part of some candidate key).
- **Transitive dependency** – If $A \rightarrow B$ and $B \rightarrow C$ are two FDs then $A \rightarrow C$ is called transitive dependency.

STUD_NO	STUD_NAME	STUD_STATE	STUD_COUNTRY	STUD AGE
1	RAM	HARYANA	INDIA	20
2	RAM	PUNJAB	INDIA	19
3	SURESH	PUNJAB	INDIA	21

Table 4

In relation STUDENT given in Table 4, FD set: {STUD_NO \rightarrow STUD_NAME, STUD_NO \rightarrow STUD_STATE, STUD_STATE \rightarrow STUD_COUNTRY, STUD_NO \rightarrow STUD_AGE} Candidate Key: {STUD_NO}

For this relation in table 4, STUD_NO \rightarrow STUD_STATE and STUD_STATE \rightarrow STUD_COUNTRY are true. STUD_COUNTRY is transitively dependent on STUD_NO. It violates the third normal form. To convert it in third normal form, we will decompose the relation STUDENT (STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_COUNTRY, STUD_AGE) as: STUDENT (STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_AGE) STATE_COUNTRY (STATE, COUNTRY)

Boyce Codd normal form (BCNF)

- It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF.
- A table complies with BCNF if it is in 3NF and for every functional dependency $X \rightarrow Y$, X should be the super key of the table.

Example: Suppose there is a company wherein employees work in more than one department. They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

- **Functional dependencies :**

$\text{emp_id} \rightarrow \text{emp_nationality}$

$\text{emp_dept} \rightarrow \{\text{dept_type}, \text{dept_no_of_emp}\}$

- **Candidate key:** {emp_id, emp_dept}

The table is not in BCNF as neither emp_id nor emp_dept alone are keys.

- To make the table comply with BCNF we can break the table in three tables like this:

emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

SQL Process

- When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.

- There are various components included in the process.
 - These components are Query Dispatcher, Optimization Engines, Classic Query Engine and SQL Query Engine, etc.
 - Classic query engine handles all non-SQL queries but SQL query engine won't handle logical files.

SQL Statement Types

- **DDL**—Data Definition Language
- **DML**—Data Manipulation Language
- **DCL**—Data Control Language
- **DQL**—Data Query Language

DDL -Data Definition Language

Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.

DQL –Data Query Language

Command	Description
SELECT	Retrieves certain records from one or more tables

DML –Data Manipulation Language

Command	Description
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

DCL –Data Control Language

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

Create, Drop, Use Database Syntax

- SQL CREATE DATABASE STATEMENT
 - CREATE DATABASE database_name;
- SQL DROP DATABASE Statement:
 - DROP DATABASE database_name;
- SQL USE STATEMENT
 - USE DATABASE database_name;

Create, Drop, Alter Table Syntax

- SQL CREATE TABLE STATEMENT
 - CREATE TABLE table_name(column1 datatype, column2 datatype, column3 datatype, , columnN datatype, PRIMARY KEY(one or more columns));
- SQL DROP TABLE STATEMENT
 - DROP TABLE table_name;
- SQL TRUNCATE TABLE STATEMENT
 - TRUNCATE TABLE table_name;
- SQL ALTER TABLE STATEMENT
 - ALTER TABLE table_name ADD NEW_COL INT
 - ALTER TABLE table_name{ADD|DROP|MODIFY}column_name{data_ype};
- SQL ALTER TABLE STATEMENT (RENAME)
 - ALTER TABLE table_name RENAME TO new_table_name;

Insert, Update, Delete Syntax

- SQL INSERT INTO STATEMENT
 - `INSERT INTO table_name(column1, column2....columnN) VALUES (value1, value2....valueN);`
- SQL UPDATE STATEMENT
 - `UPDATE table_nameSET column1 = value1, column2 = value2....columnN=valueN[WHERE CONDITION];`
- SQL DELETE STATEMENT
 - `DELETE FROM table_nameWHERE {CONDITION};`

Select Statement Syntax

- SQL SELECT STATEMENT
 - `SELECT column1, column2....columnN FROM table_name;`
- SQL DISTINCT CLAUSE
 - `SELECT DISTINCT column1, column2....columnN FROM table_name;`
- SQL WHERE CLAUSE
 - `SELECT column1, column2....columnN FROM table_name WHERE CONDITION;`
- SQL AND/OR CLAUSE
 - `SELECT column1, column2....columnN FROM table_name WHERE CONDITION-1 {AND|OR} CONDITION-2;`

Select Statement Syntax

- SQL IN CLAUSE
 - SELECT column1, column2....columnN FROM table_name WHERE column_name IN (val-1, val-2,...val-N);
- SQL BETWEEN CLAUSE
 - SELECT column1, column2....columnN FROM table_name WHERE column_name BETWEEN val-1 AND val-2;
- SQL LIKE CLAUSE
 - SELECT column1, column2....columnN FROM table_name WHERE column_name LIKE { PATTERN };
- SQL ORDER BY CLAUSE
 - SELECT column1, column2....columnN FROM table_name WHERE CONDITION ORDER BY column_name{ASC|DESC};

Select Statement Syntax

- SQL GROUP BY CLAUSE
 - SELECT **SUM**(column_name) FROM table_name **WHERE** CONDITION
GROUP BY column_name;
- SQL COUNT CLAUSE
 - SELECT COUNT(column_name)FROM table_name WHERE CONDITION;
- SQL HAVING CLAUSE
 - SELECT SUM(column_name) FROM table_name WHERE CONDITION
GROUP BY column_name**HAVING** (arithmeticfunction condition);

Create and Drop Index Syntax

- SQL CREATE INDEX Statement :
 - ○ CREATE UNIQUE INDEX index_nameON table_name(column1, column2,...columnN);
- SQL DROP INDEX STATEMENT
- ALTER TABLE table_nameDROP INDEX index_name;
- SQL DESC Statement :
 - DESC table_name;

Commit and Rollback Syntax

- SQL COMMIT STATEMENT
 - COMMIT;

- SQL ROLLBACK STATEMENT
 - ROLLBACK;

SQL Join Types

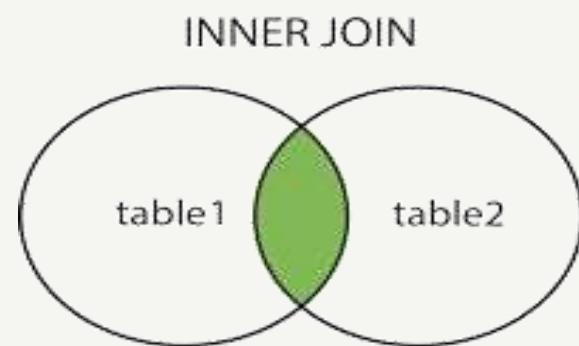
- **INNER JOIN:** returns rows when there is a match in both tables.
- **LEFT JOIN:** returns all rows from the left table, even if there are no matches in the right table.
- **RIGHT JOIN:** returns all rows from the right table, even if there are no matches in the left table.
- **FULL JOIN:** returns rows when there is a match in one of the tables.

JOIN

- A SQL Join statement is used to combine data or rows from two or more tables **based on a common field between them.**
- Different types of Joins are:
 1. INNER JOIN
 2. LEFT JOIN
 3. RIGHT JOIN
 4. FULL JOIN

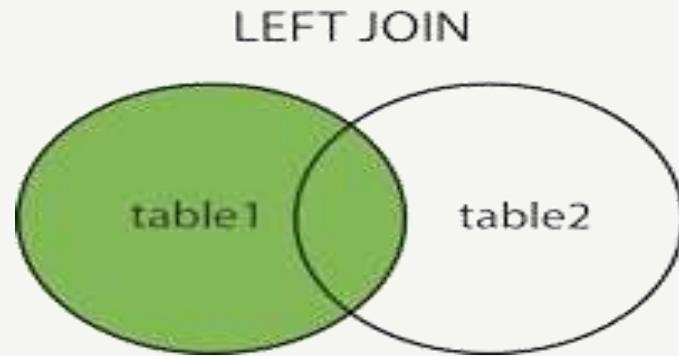
Inner Join Syntax

- The most frequently used and important of the joins is the INNER JOIN. They are also referred to as an EQUIJOIN.
-
- The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate.
- The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.
- SYNTAX:
 - `SELECT table1.column1, table2.column2...FROM table1 INNER JOIN table2 ON table1.common_filed = table2.common_field;`



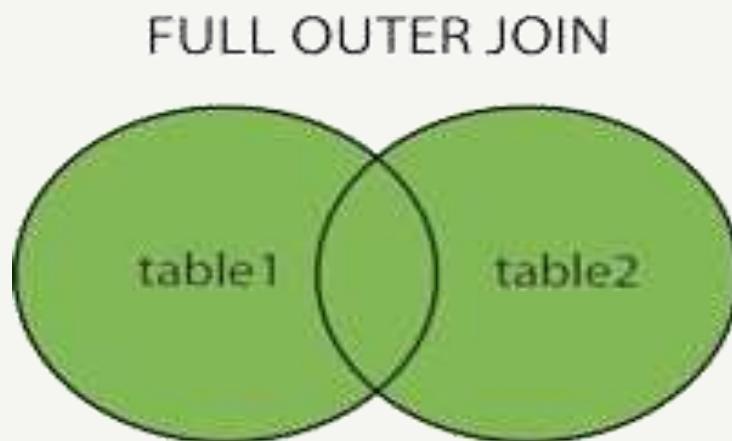
Left Join Syntax

- The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in right table, the join will still return a row in the result, but with NULL in each column from right table.
- This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.
- SYNTAX:
 - `SELECT table1.column1, table2.column2...FROM table1 LEFT JOIN table2 ON
table1.common_field = table2.common_field;`



Full Join Syntax

- The SQL FULL JOIN combines the results of both left and right outer joins.
- The joined table will contain all records from both tables, and fill in NULLs for missing matches on either side.
- SYNTAX:
 - `SELECT table1.column1, table2.column2...FROM table1 FULL JOIN table2 ON table1.common_field = table2.common_field;`



Function

SQL has many built-in functions for performing calculations on data. They are divided into 2 categories:

1. Aggregate Function
2. Scalar Function

Aggregate Function

These functions are used to do operations from the values of the column and a single value is returned.

- AVG()-Returns the average value
- COUNT()-Returns the number of rows
- FIRST()-Returns the first value
- LAST()-Returns the last value
- MAX()-Returns the largest value
- MIN()-Returns the smallest value
- SUM() -Returns the sum

1. AVG()

- Syntax:
 - `SELECT AVG(column_name) FROM table_name;`
- Example:
 - `SELECT AVG(AGE) AS AvgAge FROM Students;`
- Output:

AvgAge
19.4

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

2. COUNT()

- Syntax:
 - `SELECT COUNT(column_name) FROM table_name;`
- Example:
 - `SELECT COUNT(*) AS NumStudents FROM Students;`
- Output:

NumStudents
5

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

3. FIRST()

- Syntax:
 - `SELECT FIRST(column_name) FROM table_name;`
- Example:
 - `SELECT FIRST(MARKS) AS MarksFirst FROM Students;`

MarksFirst
90

- Output:

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

4. LAST()

- Syntax:
 - `SELECT LAST(column_name) FROM table_name;`
- Example:
 - `SELECT LAST(MARKS) AS MarksLast FROM Students;`

MarksLast
82

- Output:

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

5. MAX()

- Syntax:
 - `SELECT MAX(column_name) FROM table_name;`
- Example:
 - `SELECT MAX(MARKS) AS MaxMarks FROM Students;`

- Output:

MaxMarks
95

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

6. MIN()

- Syntax:
 - `SELECT MIN(column_name) FROM table_name;`
- Example:
 - `SELECT MIN(MARKS) AS MinMarks FROM Students;`

- Output: **MinMarks**
 50

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

Scalar functions

These functions are based on user input, these too returns single value.

- **UCASE()** -Converts a field to upper case
- **LCASE()** -Converts a field to lower case
- **MID()** -Extract characters from a text field
- **LENGTH()** -Returns the length of a text field
- **ROUND()** -Rounds a numeric field to the number of decimals specified
- **NOW()** -Returns the current system date and time
- **FORMAT()** -Formats how a field is to be displayed

1. UCASE()

- Syntax:
 - `SELECT UCASE(column_name) FROM table_name;`
- Example:
 - `SELECT UCASE(NAME) FROM Students;`

NAME
HARSH
SURESH
PRATIK
DHANRAJ
RAM

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

2. LCASE()

- Syntax:
 - `SELECT LCASE(column_name) FROM table_name;`
- Example:
 - `SELECT LCASE(NAME) FROM Students;`
- Output:

NAME
harsh
suresh
pratik
dhanraj
ram

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

3. MID()

- Syntax:
 - `SELECT MID(column_name,start,length) AS some_nameFROM table_name;`
 - specifying length is optional here, and start signifies start position (starting from 1)
- Example:
 - `SELECT MID(NAME,1,4) FROM Students;`
- Output:

NAME
HARS
SURE
PRAT
DHAN
RAM

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

4. LENGTH()

- Syntax:
 - `SELECT LENGTH(column_name) FROM table_name;`
- Example:
 - `SELECT LENGTH(NAME) FROM Students;`
- Output:

NAME
5
6
6
7
3

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

5. ROUND()

- Syntax:
 - SELECT ROUND(column_name,decimals) FROM table_name;
 - decimals-number of decimals to be fetched.
- Example:
 - SELECT ROUND(MARKS,0) FROM table_name;

- Output:

MARKS
90
50
80
95
85

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

6. NOW()

- Syntax:
 - `SELECT NOW() FROM table_name;`
- Example:
 - `SELECT NAME, NOW() AS DateTime FROM Students`

- Output:

	NAME	DateTime
	HARSH	1/13/2017 1:30:11 PM
	SURESH	1/13/2017 1:30:11 PM
	PRATIK	1/13/2017 1:30:11 PM
	DHANRAJ	1/13/2017 1:30:11 PM
	RAM	1/13/2017 1:30:11 PM

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

7. FORMAT()

- Syntax:
 - `SELECT FORMAT(column_name) FROM table_name;`
- Example:
 - `SELECT NAME, FORMAT(Now(),'YYYY-MM-DD') AS Date FROM Students;`

- Output:

	NAME	Date
	HARSH	2017-01-13
	SURESH	2017-01-13
	PRATIK	2017-01-13
	DHANRAJ	2017-01-13
	RAM	2017-01-13

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

Refer This Example

- Cursor Example:
 - https://github.com/TopsCode/Data_AnalyticsWithPython_ML_AI/blob/master/SQL/Cursor/example
- Stored Procedure with one parameter:
 - https://github.com/TopsCode/Data_AnalyticsWithPython_ML_AI/blob/master/SQL/Cursor/cursorExampleOneparam
- Stored Procedure with multiple parameter:
 - https://github.com/TopsCode/Data_AnalyticsWithPython_ML_AI/blob/master/SQL/Cursor/multipleParam

Transaction Control

The following commands are used to control transactions.

- **COMMIT**– to save the changes.
- **ROLLBACK**– to roll back the changes.
- **SAVEPOINT**– creates points within the groups of transactions in which to ROLLBACK.

1. COMMIT

- The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.
- The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command.
- The syntax for the COMMIT command is as follows:
 - COMMIT;

2. ROLLBACK

- The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.
- This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.
- The syntax for the COMMIT command is as follows:
 - ROLLBACK;

3. SAVEPOINT

- A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.
- The syntax for a SAVEPOINT command is as shown below.
 - `SAVEPOINT SAVEPOINT_NAME;`
- This command serves only in the creation of a SAVEPOINT among all the transactional statements. The ROLLBACK command is used to undo a group of transactions.
- The syntax for rolling back to a SAVEPOINT is as shown below.
 - `ROLLBACK TO SAVEPOINT_NAME;`

Excel

Modules:

- 1. Introductions to Excel**
- 2. Excel Functions**
- 3. Excel Charts**



Course Objectives

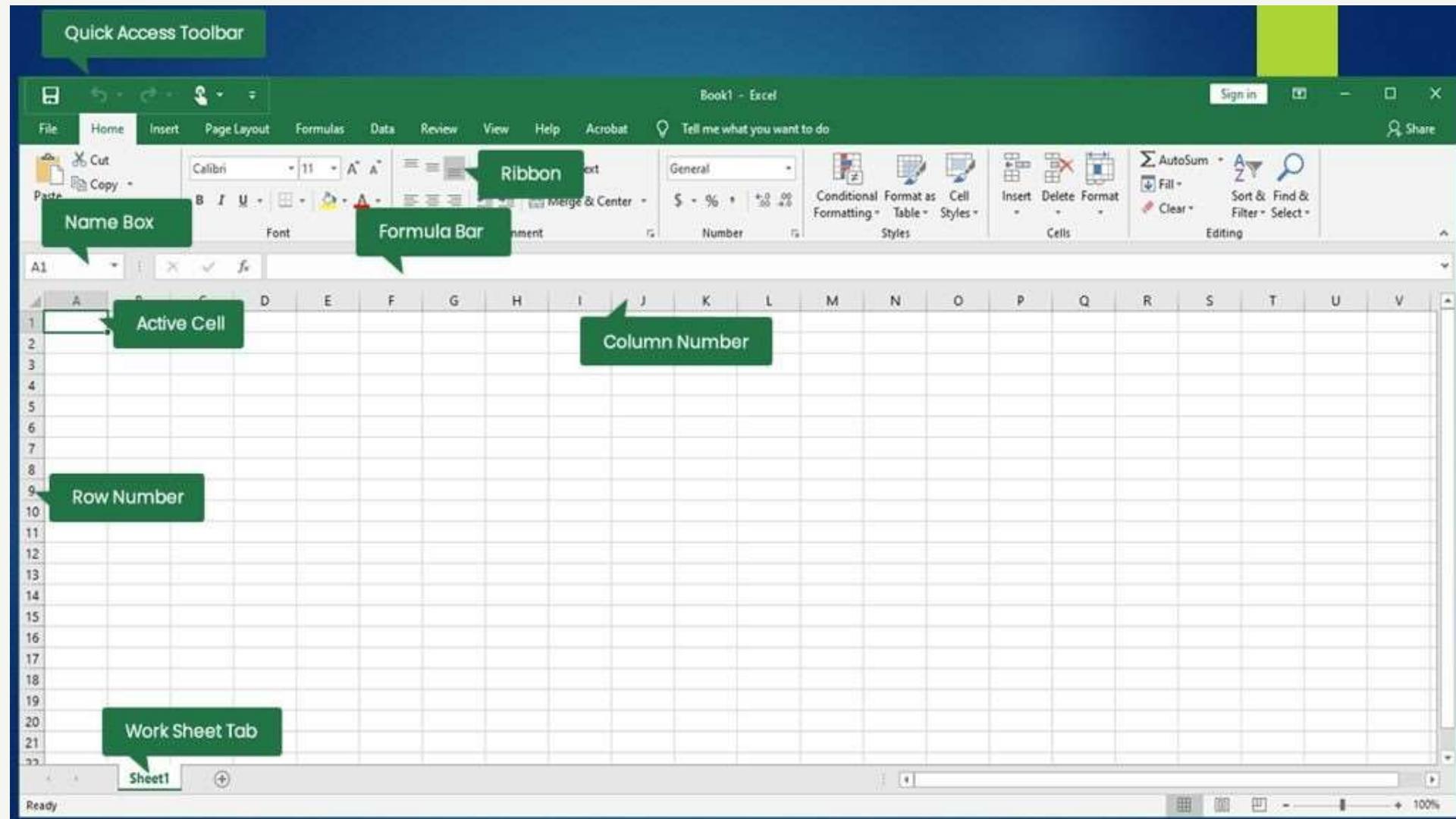
- ❖ Set up the chart.
- ❖ Differentiate between formulas and **functions** in Excel.
- ❖ Use Functions of Excel.
- ❖ Access and manipulate data.

Importance of Excel

- ❖ Microsoft Excel is one of the most important software programs ever created.
- ❖ Businesses large and small, in countries all around the world, run off of systems and processes built on and around Excel.
- ❖ So many people using Excel as part of their day-to-day jobs.
- ❖ Excel is not going anywhere, and businesses will continue to use Excel as a primary tool.
- ❖ Excel is a powerful tool that has become entrenched in business processes worldwide whether for analyzing or to Organize Data

Excel Vs Tableau

- ❖ Tableau is a Powerful **Data Visualization Tool**.
- ❖ But When it comes to Data Creation and Data Organization **Tableau is not Optimal**.
- ❖ Excel helps in generating computational solutions using built-in functions in easy way.
- ❖ User can learn to perform tasks like data manipulation, Simple Calculations and Charting just as easily, if not more easily, in Tableau.
- ❖ Learning Excel further helps you in fields like Data Analytics, Machine Learning, etc. where data preparation is a most important tasks.



Introduction to the Excel Interface

- ❖ **Title Bar** – contains the name of the workbook.
- ❖ **Worksheet Tabs** – a list of all the worksheets in the workbook.
- ❖ **Ribbon Tabs** – the top-level menu items consists of **Home, Insert, Page Layout, Formulas**, etc.
- ❖ **Ribbon** – a collection of Excel capabilities organized into **groups** corresponding to some ribbon tab.
- ❖ **Active Cell** – displays the currently referenced cell.
- ❖ **Name Box** – contains the address of the active cell.
- ❖ **Formula Bar** – contains the contents of the active cell.
- ❖ **File** – menu containing a number of options, such as open, save, and print.

CELL Reference

A **cell reference** or **cell address** is a combination of a column letter and a row number that identifies a **cell** on a worksheet.

CELL RANGE

A group of **cells** is known as a **cell range**.

Team	Country	Champions	Games play	Points earned	
Borussia	Germany	Yes	34	71	=A1:E5
Milan	Italy	Yes	38	57	
Manchester	England	Yes	38	64	
Hamburger	Germany	No	34	27	
Lazio	Italy	No	38	56	
Fiorentina	Italy	No	38	65	
Bayern	Germany	Yes	34	90	

A1:E5

Formula and Functions

C8	A	B	C	D
1				
2				
3				
4				
5			Marks	
6			89	
7			93	
8	Formula		182	
9				

C7	A	B	C
1			
2			
3			
4			Marks
5			89
6			93
7	Function		182

- ❖ Formula:
 - A **formula** is an expression which calculates the value of a cell.

- ❖ Functions
 - **Functions** are predefined **formulas** and are already available in **Excel**.

COUNT:

- ❖ Use the COUNT function to get the number of entries in a **number field**.

- ❖ Syntax:
`=Count(range)`

- ❖ Arguments:
 - **range** - The range of cells to count.

Let's use count for numbers entries.

COUNTIF:

- ❖ Use the COUNT function Only If a Condition is Satisfied
- ❖ Syntax
 - =COUNTIF(range, criteria)
- ❖ Arguments
 - **range** - The range of cells to count.
 - **criteria** - The criteria that controls which cells should be counted.

COUNTIFS

- ❖ Use the COUNT function For more than one IF Conditions.
- ❖ Syntax
 - =COUNTIFS (range1, criteria1, [range2], [criteria2], ...)
- ❖ Arguments
 - **range1** - The first range to evaluate.
 - **criteria1** - The criteria to use on range1.
 - **range2** - [optional] The second range to evaluate.
 - **criteria2** - [optional] The criteria to use on range2.

SUM:

- ❖ The Microsoft **Excel SUM function** adds all numbers in a range of cells and returns the result
- ❖ Syntax
 - =SUM (number1, [number2], [number3], ...)
- ❖ Arguments
 - **number1** - The first value to sum.
 - **number2** - [optional] The second value to sum.
 - **number3** - [optional] The third value to sum.

SUMIF

- ❖ Adds all numbers in a range of cells If Condition is True and returns the result.
- ❖ Syntax
 - =SUMIF (range, criteria, [sum_range])
- ❖ Arguments
 - **range** - The range of cells that you want to apply the criteria against.
 - **criteria** - The criteria used to determine which cells to add.
 - **sum_range** - [optional] The cells to add together. If sum_range is omitted, the cells in range are added together instead.

AVERAGE

Returns the Average (Arithmetic Mean) of the Arguments.

❖ Syntax

- `=AVERAGE (number1, [number2], ...)`

❖ Arguments

- **number1** - A number or cell reference that refers to numeric values.
- **number2** - [optional] A number or cell reference that refers to numeric values.

AVERAGEIF

RETURNS THE AVERAGE (**ARITHMETIC MEAN**) OF THE ARGUMENTS IF A CONDITION IS TRUE

❖ Syntax

- `=AVERAGEIF (range, criteria, [average_range])`

❖ Arguments

- **range** - One or more cells, including numbers or names, arrays, or references.
- **criteria** - A number, expression, cell reference, or text.
- **average_range** - [optional] The cells to average. When omitted, range is used.

AVERAGEIFS

❖ Syntax

- =AVERAGEIFS (**average_range**, range1, criteria1, [range2], [criteria2], ...)

❖ Arguments

- average_range**- The range to average.
- range1** - The first range to evaluate.
- criteria1** - The criteria to use on range1.
- range2** - [optional] The second range to evaluate.
- criteria2** - [optional] The criteria to use on range2.

IFERROR

The Microsoft Excel IFERROR function returns an alternate **value** if a formula results in an error.

❖ Syntax

- =IFERROR (value, value_if_error)

❖ Arguments

- **value** - The value, reference, or formula to check for an error.
- **value_if_error** - The value to return if an error is found.

LOOKUP

The Excel LOOKUP function performs an approximate match lookup in a one-column or one-row **range**, and returns the corresponding **value** from another one-column or one-row **range**.

- ❖ VLOOKUP
- ❖ HLOOKUP

VLOOKUP function

Use VLOOKUP when you need to find things in a table or a range by row. For example, look up a price of an automotive part by the part number, or find an employee name based on their employee ID. In its simplest form, the VLOOKUP function says:

=VLOOKUP(What you want to look up, where you want to look for it, the column number in the range containing the value to return, return an Approximate or Exact match – indicated as 1/TRUE, or 0/FALSE).

How to get started

There are four pieces of information that you will need in order to build the VLOOKUP syntax:

- 1.The value you want to look up, also called the lookup value.
- 2.The range where the lookup value is located. Remember that the lookup value should always be in the first column in the range for VLOOKUP to work correctly. For example, if your lookup value is in cell C2 then your range should start with C.
- 3.The column number in the range that contains the return value. For example, if you specify B2:D11 as the range, you should count B as the first column, C as the second, and so on.
- 4.Optionally, you can specify TRUE if you want an approximate match or FALSE if you want an exact match of the return value. If you don't specify anything, the default value will always be TRUE or approximate match.

	A	B	C	D	E
1	ID	Last name	First name	Title	Birth date
2		101 Davis	Sara	Sales Rep	12/08/68
3		102 Fontana	Olivier	VP (Sales)	02/19/52
4		103 Leal	Karina	Sales Rep	08/30/63
5		104 Patten	Michael	Sales Rep	09/19/58
6		105 Burke	Brian	Sales Manager	03/04/55
7		106 Sousa	Luis	Sales Rep	07/02/63
8					
9					
10	Formula	<code>=VLOOKUP(B3,B2:E7,2, FALSE)</code>			
11	Result	Olivier			
12					

VLOOKUP looks for *Fontana* in the first column (column B) in table_array B2:E7, and returns *Olivier* from the second column (column C) of the table_array. FALSE returns an exact match.



	A	B	C	D	E
1	ID	Last name	First name	Title	Birth date
2	101	Davis	Sara	Sales Rep	12/08/68
3	102	Fontana	Olivier	VP (Sales)	02/19/52
4	103	Leal	Karina	Sales Rep	08/30/63
5	104	Patten	Michael	Sales Rep	09/19/58
6	105	Burke	Brian	Sales Manager	03/04/55
7	106	Sousa	Luis	Sales Rep	07/02/63

IF checks to see if VLOOKUP returns *Sousa* as the last name of employee corresponding to 103 (lookup_value) in A1:E7 (table_array). Because the last name corresponding to 103 is *Leal*, the IF condition is false, and *Not found* is displayed.

8					
9					
10	Formula	=IF(VLOOKUP(103,A1:E7,2, FALSE)="Sousa","Located","Not found")			
11	Result	Not found			

Practical :- https://github.com/TopsCode/Data_Analytics/blob/main/Excel/Vlookup.xlsx

HLOOKUP function

Use HLOOKUP when your comparison values are located in a row across the top of a table of data, and you want to look down a specified number of rows.

Use VLOOKUP when your comparison values are located in a column to the left of the data you want to find.

The H in HLOOKUP stands for "Horizontal."

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

The HLOOKUP function syntax has the following arguments:

Lookup_value Required. The value to be found in the first row of the table. Lookup_value can be a value, a reference, or a text string.

Table_array Required. A table of information in which data is looked up. Use a reference to a range or a range name.

The values in the first row of table_array can be text, numbers, or logical values.

If range_lookup is TRUE, the values in the first row of table_array must be placed in ascending order: ...-2, -1, 0, 1, 2,... , A-Z, FALSE, TRUE; otherwise, HLOOKUP may not give the correct value. If range_lookup is FALSE, table_array does not need to be sorted.

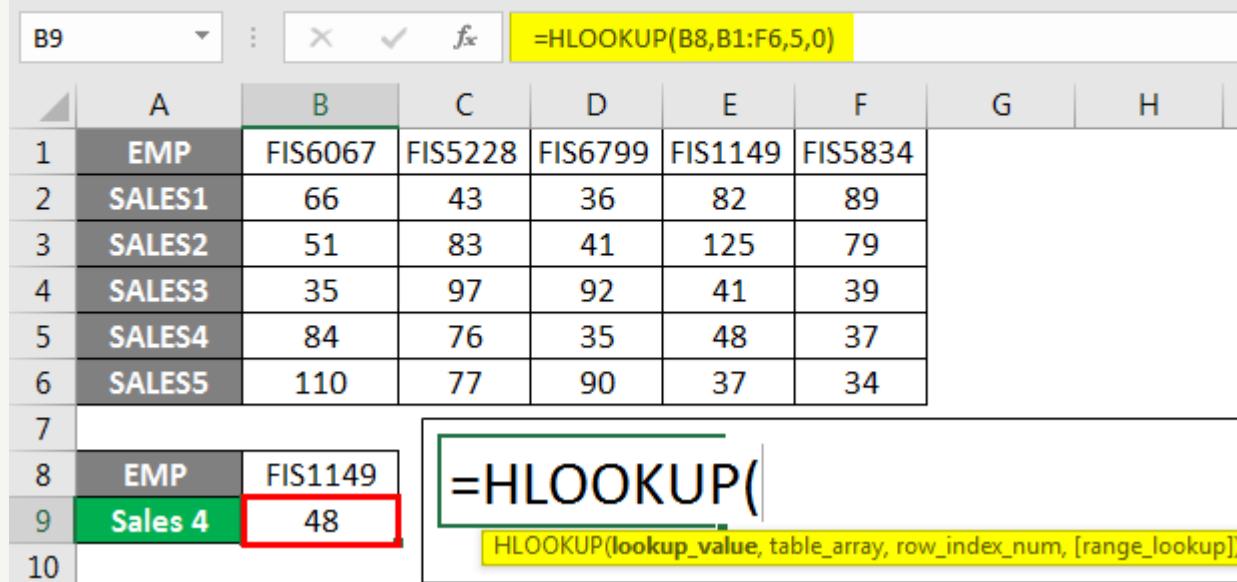
Uppercase and lowercase text are equivalent.

Sort the values in ascending order, left to right. For more information, see Sort data in a range or table.

Row_index_num Required. The row number in **table_array** from which the matching value will be returned. A **row_index_num** of 1 returns the first row value in **table_array**, a **row_index_num** of 2 returns the second row value in **table_array**, and so on. If **row_index_num** is less than 1, HLOOKUP returns the #VALUE! error value; if **row_index_num** is greater than the number of rows on **table_array**, HLOOKUP returns the #REF! error value.

Range_lookup Optional. A logical value that specifies whether you want HLOOKUP to find an exact match or an approximate match. If TRUE or omitted, an approximate match is returned. In other words, if an exact match is not found, the next largest value that is less than **lookup_value** is returned. If FALSE, HLOOKUP will find an exact match. If one is not found, the error value #N/A is returned.

HLOOKUP Formula in Excel



The screenshot shows an Excel spreadsheet with a table of data and a formula being entered.

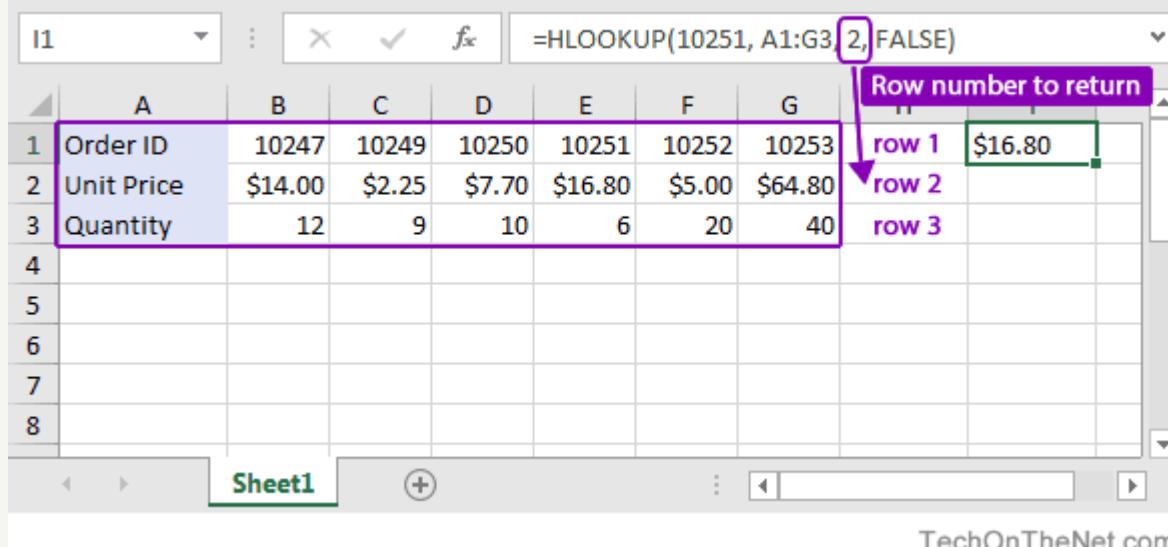
Table Data:

	A	B	C	D	E	F	G	H
1	EMP	FIS6067	FIS5228	FIS6799	FIS1149	FIS5834		
2	SALES1	66	43	36	82	89		
3	SALES2	51	83	41	125	79		
4	SALES3	35	97	92	41	39		
5	SALES4	84	76	35	48	37		
6	SALES5	110	77	90	37	34		
7								
8	EMP	FIS1149						
9	Sales 4	48						
10								

Formula Bar: =HLOOKUP(B8,B1:F6,5,0)

Cell B9: =HLOOKUP(

Formula Hint: HLOOKUP(lookup_value, table_array, row_index_num, [range_lookup])



The screenshot shows an Excel spreadsheet with a table of data. The table has columns labeled A through G and rows labeled 1 through 8. The data includes Order ID, Unit Price, and Quantity. The formula bar at the top shows =HLOOKUP(10251, A1:G3, 2, FALSE). A callout box labeled "Row number to return" points to the value \$16.80 in cell H2, which corresponds to row 2. The table data is as follows:

A	B	C	D	E	F	G	H	
1	Order ID	10247	10249	10250	10251	10252	10253	Row number to return
2	Unit Price	\$14.00	\$2.25	\$7.70	\$16.80	\$5.00	\$64.80	row 1
3	Quantity	12	9	10	6	20	40	row 2
4								row 3
5								
6								
7								
8								

TechOnTheNet.com

Practical :- https://github.com/TopsCode/Data_Analytics/blob/main/Excel/Hlookup.xlsx

Charts

A **chart** is a graphical representation of data.

A **chart** is often called a **graph**, helps to **visualize** data.

EXCEL CHARTS - TYPES

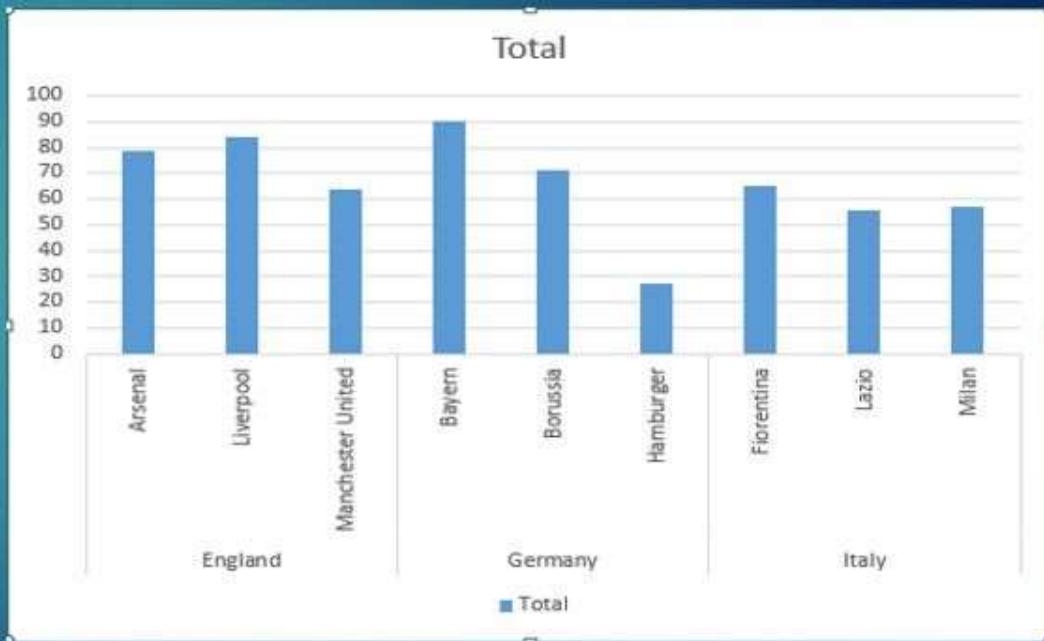
- **COLUMN** CHART
- **LINE** CHART
- **PIE** CHART
- **BAR** CHART
- **AREA** CHART
- XY (**SCATTER**) CHART

Column Chart

Column charts are used to compare values across categories by using vertical bars.

To create a column chart, follow these steps:

1. Enter data in a spreadsheet.
2. Select the data.
3. Click Insert > Insert **Column** or **Bar Chart** icon, and select a **column chart** option of your choice.



Bar Chart

A bar chart is the horizontal version of a column chart. Use a bar chart if you have large text labels.

To create a Bar chart, follow these steps:

1. Select all the data that you want included in the **bar chart**.
2. Be sure to include the column and row headers, which will become the labels in the **bar chart**.
3. Click on the Insert tab and then on Insert Column or Bar Chart button in the **Charts** group



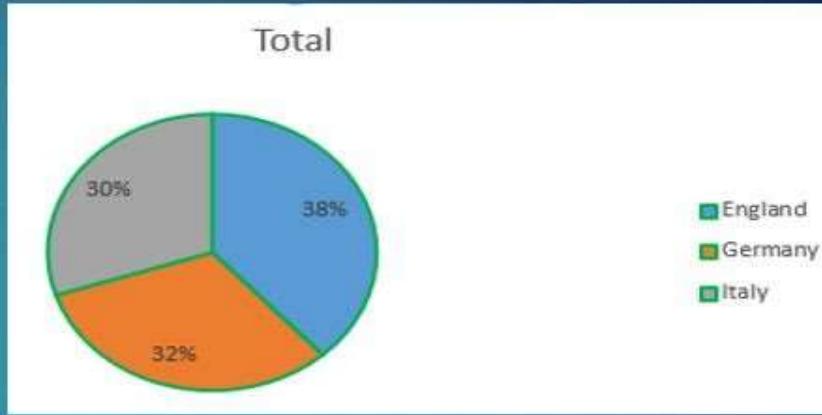
Practical- https://github.com/TopsCode/Data_Analytics/blob/main/Excel/Bar%20chart.xlsx

Pie Chart

Pie charts are used to display the contribution of each value (slice) to a total (pie). Pie charts always use one data series.

To create a **Pie chart**, follow these steps:

1. In your spreadsheet, select the data to use for your **pie chart**.
2. Click Insert > Insert **Pie or Doughnut Chart**, and then pick the **chart** you want.



https://github.com/TopsCode/Data_Analytics/blob/main/Excel/Pie%20Chart.xlsx

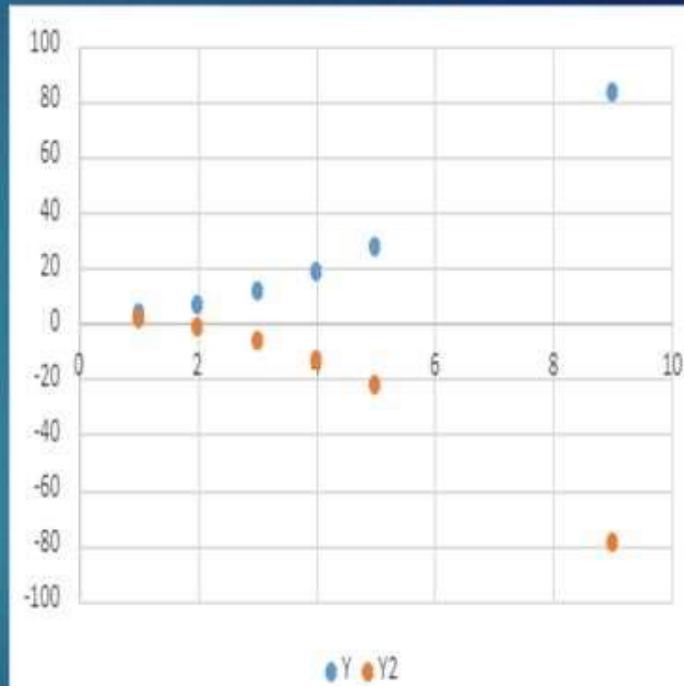
Scatter Chart

Scatter plots are often used to find out if there's a relationship between two variables suppose X and Y.

A scatterplot is used to represent a correlation between two **variables**.

There are two types of correlations: positive and negative.

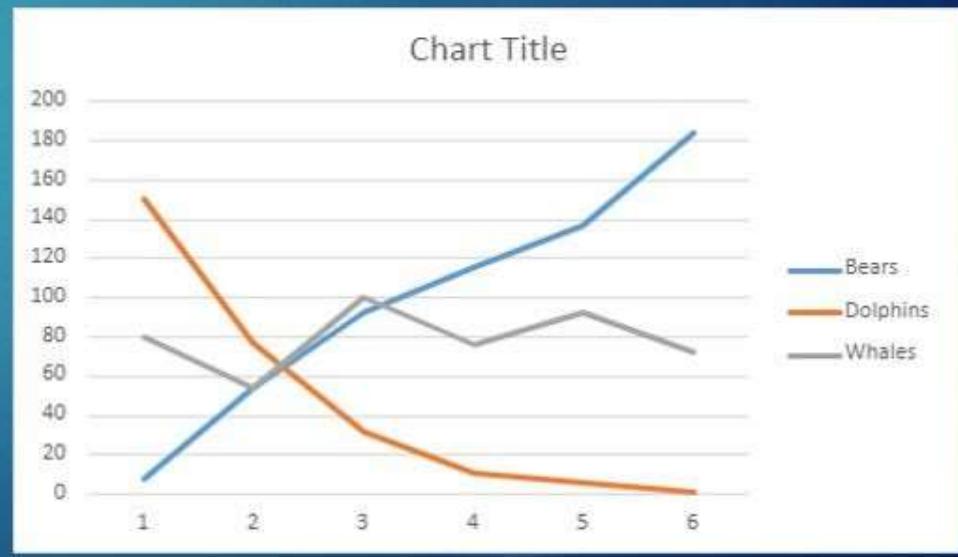
Variables that are positively correlated move in the same direction, while **variables** that are negatively correlated move in opposite directions.



Line Charts

A **line chart** is often **used to** visualize a trend in data over intervals of time – a time series.

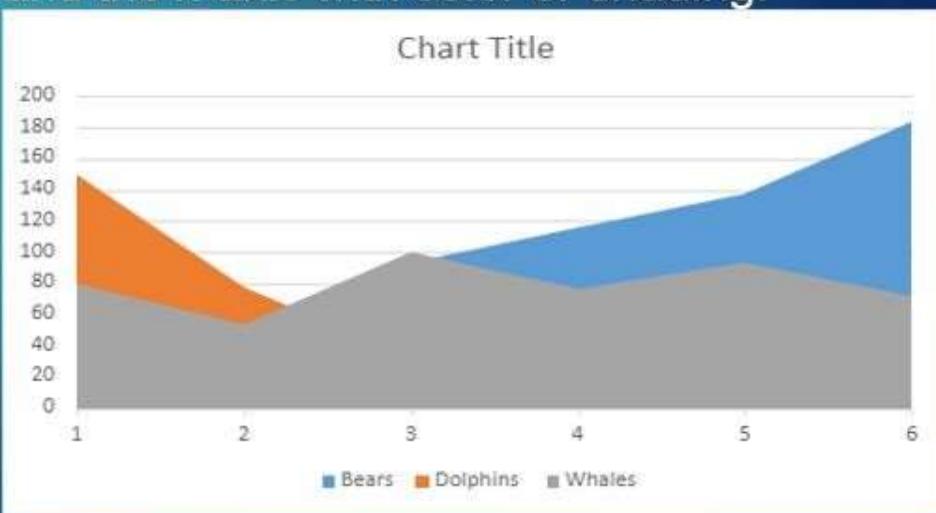
Hence **Line graphs** can be used to show how information or data change over time for ex: Trends.



Area Chart

- ❖ An **area chart** represents the change in a one or more quantities over time.
- ❖ It is made by plotting a series of data points over time, connecting those data points with line segments
- ❖ Then filling in the **area** between the line and the x-axis with color or shading.

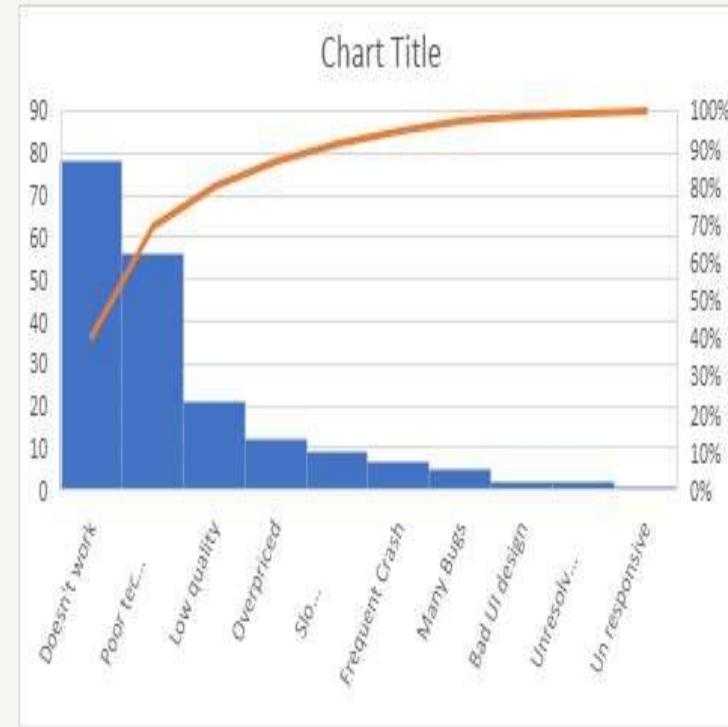
It can be seen as the Area char is a Combination of the **Bar Chart** and **Line Chart**



Pareto charts highlight the biggest factors in a set of data.

Following the idea of 80/20 analysis, they try to show which (approximately) 20% of the categories contribute 80% of the data being measured.

- Quickly highlights most important data
- Excel automatically builds histogram and adds Pareto line



https://github.com/TopsCode/Data_Analytics/blob/main/Excel/Pareto%20Chart.xlsx

Excel Dashboards

Excel dashboards make it easy to perform quick overviews of data reports rather than going through large volumes of data. Overviews help in making quick and urgent decisions since one can skim through a lot of information at once and within a short time.

The dashboards help in tracking Key Performance Indicators (KPIs) with ease, which helps organizations track the progress on their targets. They provide a high-level summary of key aspects of your data to keep everyone at par with the progress, hence giving the organization a timely indicator for necessary action in real-time.

The Excel Dashboard is used to display overviews of large data tracks.

Excel Dashboards use dashboard elements like tables, charts, and gauges to show the overviews.

The dashboards ease the decision-making process by showing the vital parts of the data in the same window.



Statistics

Modules:

- Introduction
- Population
- Sample
- Statistic and Parameter
- Introduction of Descriptive Statistics
- Measures of Central Tendency
- Measures of Spread
- Introduction of Probability
- Terminology
- Types of Probability
- Probability Distribution
- Introduction to Inferential Statistics
- Estimation and errors
- Point estimation
- Confidence Interval
- Hypothesis and its types

What is Statistics?

Statistics is a mathematical science including methods of collecting, organizing and analyzing data in such a way that meaningful conclusions can be drawn from them. In general, its investigations and analyses fall into two broad categories called descriptive and inferential statistics.

Descriptive statistics deals with the processing of data without attempting to draw any inferences from it. The data are presented in the form of tables and graphs. The characteristics of the data are described in simple terms. Events that are dealt with include everyday happenings such as accidents, prices of goods, business, incomes, epidemics, sports data, population data.

Inferential statistics is a scientific discipline that uses mathematical tools to make forecasts and projections by analyzing the given data. This is of use to people employed in such fields as engineering, economics, biology, the social sciences, business, agriculture and communications.

Introduction to Population and Sample

A **population** often consists of a large group of specifically defined elements. For example, the population of a specific country means all the people living within the boundaries of that country.

Usually, it is not possible or practical to measure data for every element of the population under study. We randomly select a small group of elements from the population and call it a **sample**. Inferences about the population are then made on the basis of several samples.

Example:

- A company is thinking about buying 50,000 electric batteries from a manufacturer. It will buy the batteries if no more than 1% of the batteries are defective. It is not possible to test each battery in the population of 50,000 batteries since it takes time and costs money. Instead, it will select few samples of 500 batteries each and test them for defects. The results of these tests will then be used to estimate the percentage of defective batteries in the population.

Quantitative Data and Qualitative Data

Data is quantitative if the observations or measurements made on a given variable of a sample or population have numerical values.

Example: height, weight, number of children, blood pressure, current, voltage.

Data is qualitative if words, groups and categories represents the observations or measurements.

Example: colors, yes-no answers, blood group.

Quantitative data is discrete if the corresponding data values take discrete values and it is continuous if the data values take continuous values.

Example of discrete data: number of children, number of cars.

Example of continuous data: speed, distance, time, pressure.

Statistic and Parameter

Many have trouble understanding the difference between parameter and statistic, but it's important to know what exactly these measures mean and how to distinguish them.

Parameter vs statistic – both are similar, yet different measures. The first one describes the whole population, while the second describes a part of the population.

What is Parameter?

It is a measure of a characteristic of an entire population (a mass of all units under consideration that share common characteristics) based on all the elements within that population. For example, all people living in one city, all-male teenagers in the world, all elements in a shopping trolley, or all students in a classroom.

If you ask all employees in a factory what kind of lunch they prefer and half of them say pasta, you get a parameter here – 50% of the employees like pasta for lunch. On the other hand, it's impossible to count how many men in the whole world like pasta for lunch, since you can't ask all of them about their choice. In that case, you'd probably survey just a representative sample (a portion) of them and extrapolate the answer to the entire population of men. This brings us to the other measure called statistic.

It's a measure of characteristic saying something about a fraction (a sample) of the population under study. A sample in statistics is a part or portion of a population. The goal is to estimate a certain population parameter. You can draw multiple samples from a given population, and the statistic (the result) acquired from different samples will vary, depending on the samples. So, using data about a sample or portion allows you to estimate the characteristics of an entire population.

Parameter vs Statistics

Can you tell the difference between statistics and parameters now?

- A parameter is a fixed measure describing the whole population (population being a group of people, things, animals, phenomena that share common characteristics.) A statistic is a characteristic of a sample, a portion of the target population.
- A parameter is fixed, unknown numerical value, while the statistic is a known number and a variable which depends on the portion of the population.
- Sample statistic and population parameters have different statistical notations:

In population parameter, population proportion is represented by P , mean is represented by μ (Greek letter mu), σ^2 represents variance, N represents population size, σ (Greek letter sigma) represents standard deviation, $\sigma_{\bar{x}}$ represents Standard error of mean, σ/μ represents Coefficient of variation, $(X-\mu)/\sigma$ represents standardized variate (z), and ϕ represents standard error of proportion.

In sample statistics, mean is represented by \bar{x} (x-bar), sample proportion is represented by \hat{p} (phat), s represents standard deviation, s^2 represents variance, sample size is represented by n , $s_{\bar{x}}$ represents Standard error of mean, s_p represents standard error of proportion, $s/(x)$ represents Coefficient of variation, and $(x-\bar{x})/s$ represents standardized variate (z).

Example of Parameters

- 20% of U.S. senators voted for a specific measure. Since there are only 100 senators, you can count what each of them voted.

Example of Statistic

- 50% of people living in the U.S. agree with the latest health care proposal. Researchers can't ask hundreds of millions of people if they agree, so they take samples, or part of the population and calculate the rest.

What Are The Differences Between Population Parameters and Sample Statistics?

The average weight of adult men in the U.S. is a parameter with an exact value – but, we don't know it.

Standard deviation and population mean are two common parameters.

A statistic is a characteristic of a group of population, or sample. You get sample statistics when you collect a sample and calculate the standard deviation and the mean. You can use sample statistics to make certain conclusions about an entire population thanks to inferential statistics. But, you need particular sampling techniques to draw valid conclusions. Using these techniques ensures that samples deliver unbiased estimates – correct on average. When it comes to based estimates, they are systematically too low or too high, so you don't need them.

To estimate population parameters in inferential statistics, you use sample statistics. For instance, if you collect a random sample of female teenagers in the U.S. and measure their weights, you can calculate the sample mean. You can use the sample mean as an unbiased estimate of the population mean.

Introduction of Descriptive Statistics

Descriptive statistics is the term given to the analysis of data that helps describe, show or summarize data in a meaningful way such that, for example, patterns might emerge from the data. Descriptive statistics do not, however, allow us to make conclusions beyond the data we have analysed or reach conclusions regarding any hypotheses we might have made. They are simply a way to describe our data.

Descriptive statistics are very important because if we simply presented our raw data it would be hard to visualize what the data was showing, especially if there was a lot of it. Descriptive statistics therefore enables us to present the data in a more meaningful way, which allows simpler interpretation of the data. For example, if we had the results of 100 pieces of students' coursework, we may be interested in the overall performance of those students. We would also be interested in the distribution or spread of the marks. Descriptive statistics allow us to do this. How to properly describe data through statistics and graphs is an important topic and discussed in other Laerd Statistics guides. Typically, there are two general types of statistic that are used to describe data

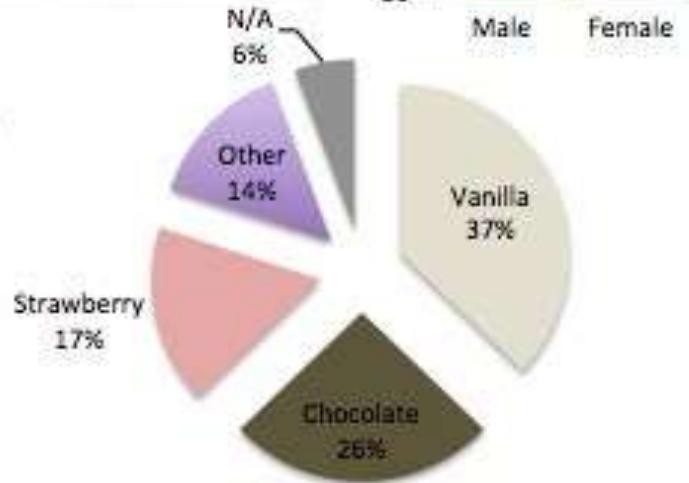
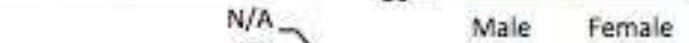
Introduction of Descriptive Statistics

- **Measures of central tendency:** these are ways of describing the central position of a frequency distribution for a group of data. In this case, the frequency distribution is simply the distribution and pattern of marks scored by the 100 students from the lowest to the highest. We can describe this central position using a number of statistics, including the mode, median, and mean.
- **Measures of spread:** these are ways of summarizing a group of data by describing how spread out the scores are. For example, the mean score of our 100 students may be 65 out of 100. However, not all students will have scored 65 marks. Rather, their scores will be spread out. Some will be lower and others higher. Measures of spread help us to summarize how spread out these scores are. To describe this spread, a number of statistics are available to us, including the range, quartiles, absolute deviation, variance and standard deviation.
- When we use descriptive statistics it is useful to summarize our group of data using a combination of tabulated description (i.e., tables), graphical description (i.e., graphs and charts) and statistical commentary (i.e., a discussion of the results).

	A	B	C	D
1	Respondent #	Age	Gender	Favorite Ice Cream Flavor
2	1	36	m	Vanilla
3	2	22	f	Chocolate
4	3	61	m	Strawberry
5	4	88	m	Other
6	5	31	m	N/A
7	6	53	m	N/A
8	7	30	f	Chocolate
9	8	64	f	Chocolate
10	9	18	m	Vanilla
11	10	16	f	Vanilla
12	11	83	m	Strawberry
13	12	16	f	Strawberry
14	13	94	m	Strawberry
15	14	55	m	Vanilla
16	15	42	f	Chocolate
17	16	18	f	Vanilla
18	17	21	#	Vanilla

Raw Data

Age	
Mean	42.6
Standard Dev.	21.9



Descriptive Statistics

Measures of Central Tendency

A **measure of central tendency** (also referred to as **measures of centre** or **central location**) is a summary measure that attempts to describe a whole set of data with a single value that represents the middle or centre of its distribution.

There are three main measures of central tendency:

- Mode
- Median
- Mean

Each of these measures describes a different indication of the typical or central value in the distribution.

Mode

The **mode** is the **most frequent occurring value** in a distribution.

Consider this dataset showing the retirement age of 11 people, in whole years:

54, 54, 54, 55, 56, 57, 57, 58, 58, 60, 60

This table shows a simple frequency distribution of the retirement age data.

Age	Frequency
54	3
55	1
56	1
57	2
58	2
60	2

Total Frequency: 11

The most commonly occurring value is 54, therefore the **mode** of this distribution is **54 years**.

Advantage of the mode:

The mode has an advantage over the median and the mean as it can be found for both numerical and categorical (non-numerical) data.

Limitations of the mode:

The are some limitations to using the mode. In some distributions, the mode may not reflect the centre of the distribution very well. When the distribution of retirement age is ordered from lowest to highest value, it is easy to see that the centre of the distribution is **57 years**, but the mode is lower, at **54 years**.

54, 54, 54, 55, 56, **57**, 57, 58, 58, 60, 60

It is also possible for there to be more than one mode for the same distribution of data, (bi-modal, or multi-modal). The presence of more than one mode can limit the ability of the mode in describing the centre or typical value of the distribution because a single value to describe the centre cannot be identified.

In some cases, particularly where the data are continuous, the distribution may have no mode at all (i.e. if all values are different).

In cases such as these, it may be better to consider using the median or mean, or group the data in to appropriate intervals, and find the modal class.

Median

The **median** is the *middle value* in distribution when the values are arranged in ascending or descending order.

The median divides the distribution in half (there are 50% of observations on either side of the median value). In a distribution with an odd number of observations, the median value is the middle value.

Median

Looking at the retirement age distribution (which has 11 observations), the median is the middle value, which is 57 years:

54, 54, 54, 55, 56, **57**, 57, 58, 58, 60, 60

When the distribution has an even number of observations, the median value is the mean of the two middle values. In the following distribution, the two middle values are 56 and 57, therefore the median equals **56.5 years**:

52, 54, 54, 54, 55, **56, 57**, 57, 58, 58, 60, 60

Advantage of the median:

The median is less affected by **outliers** and **skewed** data than the mean, and is usually the preferred measure of central tendency when the distribution is not symmetrical.

Limitations of the median:

The median cannot be identified for categorical nominal data, as it cannot be logically ordered.

Mean

The mean is the sum of the value of each observation in a dataset divided by the number of observations. This is also known as the arithmetic average.

Looking at the retirement age distribution again:

54, 54, 54, 55, 56, 57, 57, 58, 58, 60, 60

The mean is calculated by adding together all the values ($54+54+54+55+56+57+57+58+58+60+60 = 623$) and dividing by the number of observations (11) which equals 56.6 years.

Advantage of the mean:

The mean can be used for both continuous and discrete numeric data.

Limitations of the mean:

The mean cannot be calculated for categorical data, as the values cannot be summed.

As the mean includes every value in the distribution the mean is influenced by outliers and skewed distributions.

What else do I need to know about the mean?

The population mean is indicated by the Greek symbol μ (pronounced 'mu'). When the mean is calculated on a distribution from a sample it is indicated by the symbol x (pronounced X-bar).

Practical File:

https://github.com/TopsCode/Data_Analytics/blob/main/Statistics/Measure_of_Center.ipynb

Measures of Spread

Measures of spread describe how similar or varied the set of observed values are for a particular variable (data item). Measures of spread include the range, quartiles and the interquartile range, variance and standard deviation.

There are three main measures of spread:

- Range
- Quartiles
- Standard Deviation
- Variance

When can we measure spread?

The spread of the values can be measured for quantitative data, as the variables are numeric and can be arranged into a logical order with a low end value and a high end value.

Why do we measure spread?

Summarising the dataset can help us understand the data, especially when the dataset is large. As discussed in the Measures of Central Tendency , the mode, median, and mean summarise the data into a single value that is typical or representative of all the values in the dataset, but this is only part of the 'picture' that summarises a dataset. Measures of spread summarise the data in a way that shows how scattered the values are and how much they differ from the mean value.

Example

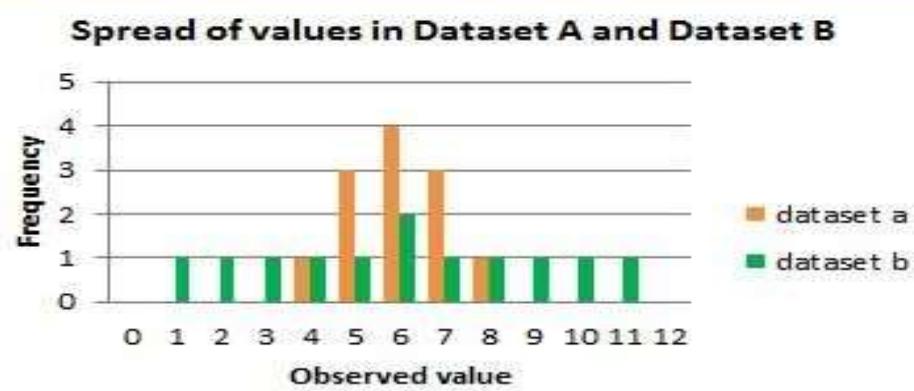
For example:

Dataset A	Dataset B
4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 8	1, 2, 3, 4, 5, 6, 6, 7, 8, 9, 10, 11

The mode (most frequent value), median (middle value*) and mean (arithmetic average) of both datasets is 6.
(*note, the median of an even numbered data set is calculated by taking the mean of the middle two observations).

If we just looked at the measures of central tendency, we may assume that the datasets are the same.

However, if we look at the spread of the values in the following graph, we can see that Dataset B is more dispersed than Dataset A. Used together, the measures of central tendency and measures of spread help us to better understand the data.



Range

The range is the difference between the smallest value and the largest value in a dataset.

Example

Calculating the Range

Dataset A

4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 8

The range is 4, the difference between the highest value (8) and the lowest value (4).

Dataset B

1, 2, 3, 4, 5, 6, 6, 7, 8, 9, 10, 11

The range is 10, the difference between the highest value (11) and the lowest value (1).

Dataset A

0	1	2	3	4	5	6	7	8	9	10	11	12	13
---	---	---	---	---	---	---	---	---	---	----	----	----	----

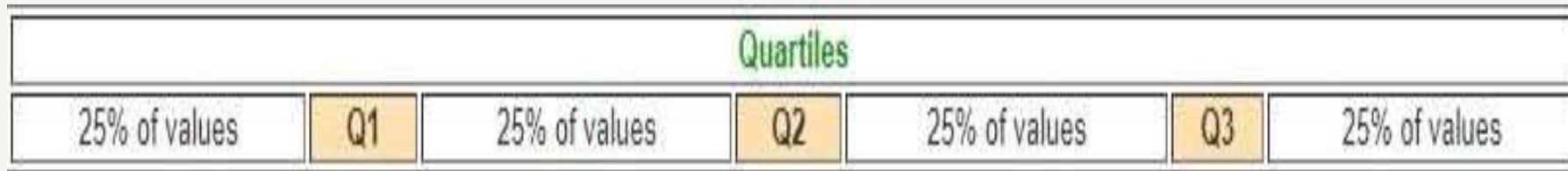
Dataset B

0	1	2	3	4	5	6	7	8	9	10	11	12	13
---	---	---	---	---	---	---	---	---	---	----	----	----	----

On a number line, you can see that the range of values for Dataset B is larger than Dataset A.

Quartiles

Quartiles divide an ordered dataset into four equal parts, and refer to the values of the point *between* the quarters. A dataset may also be divided into quintiles (five equal parts) or deciles (ten equal parts).



The **lower quartile (Q1)** is the point between the lowest 25% of values and the highest 75% of values. It is also called the **25th percentile**.

The **second quartile (Q2)** is the middle of the data set. It is also called the **50th percentile**, or the **median**.

The **upper quartile (Q3)** is the point between the lowest 75% and highest 25% of values. It is also called the **75th percentile**.

Example

Calculating Quartiles

Dataset A

4	5	5	Q1	5	6	6	Q2	6	6	7	Q3	7	7	8
---	---	---	-----------	---	---	---	-----------	---	---	---	-----------	---	---	---

As the quartile point falls between two values, the mean (average) of those values is the quartile value:

$$Q1 = (5+5) / 2 = 5$$

$$Q2 = (6+6) / 2 = 6$$

$$Q3 = (7+7) / 2 = 7$$

Dataset B

1	2	3	Q1	4	5	6	Q2	6	7	8	Q3	9	10	11
---	---	---	-----------	---	---	---	-----------	---	---	---	-----------	---	----	----

As the quartile point falls between two values, the mean (average) of those values is the quartile value:

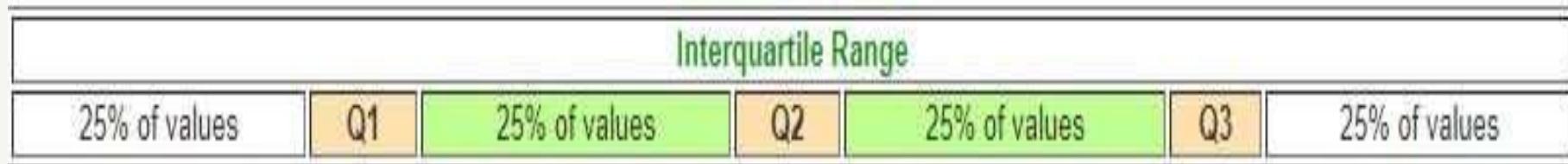
$$Q1 = (3+4) / 2 = 3.5$$

$$Q2 = (6+6) / 2 = 6$$

$$Q3 = (8+9) / 2 = 8.5$$

Interquartile Range(IQR)

The **interquartile range (IQR)** is the difference between the upper (Q3) and lower (Q1) quartiles, and describes the middle 50% of values when ordered from lowest to highest. The IQR is often seen as a better measure of spread than the range as it is not affected by outliers.



Practical File: https://github.com/TopsCode/Data_Analytics/blob/main/Statistics/Measure_of_Spread.ipynb

Introduction of Probability

How **likely** something is to happen.

Many events can't be predicted with total certainty. The best we can say is how **likely** they are to happen, using the idea of probability.

Tossing a Coin

When a coin is tossed, there are two possible outcomes:

- **Heads(H)** or
- **Tails(T)**

We say that the probability of the coin landing **H** is $\frac{1}{2}$

And the probability of the coin landing **T** is $\frac{1}{2}$



Introduction of Probability

Throwing Dice

When a single die is thrown, there are six possible outcomes: **1, 2, 3, 4, 5, 6.**

The probability of any one of them is $\frac{1}{6}$



Probability

In general:

$$\text{Probability of an event happening} = \frac{\text{Number of ways it can happen}}{\text{Total number of outcomes}}$$

Example

- The chances of rolling a "4" with a die

Number of ways it can happen: 1 (there is only 1 face with a "4" on it)

Total number of outcomes: 6 (there are 6 faces altogether)

$$\text{So the probability} = \frac{1}{6}$$

- There are 5 marbles in a bag: 4 are blue, and 1 is red. What is the probability that a blue marble gets picked?

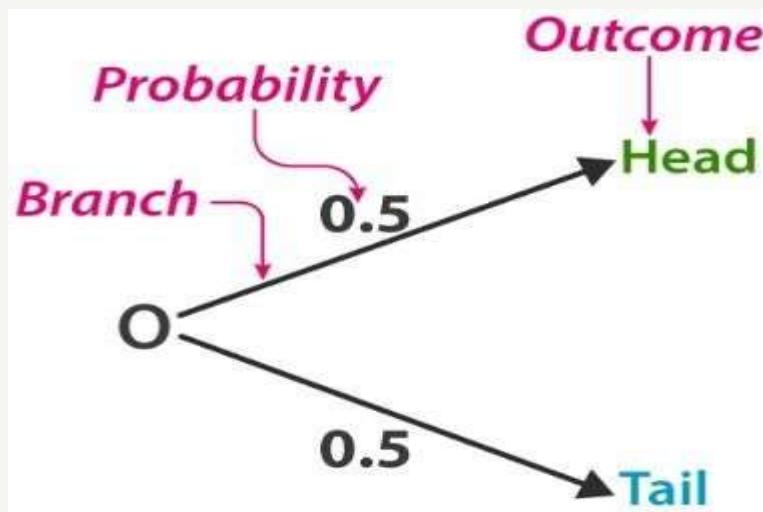
Number of ways it can happen: 4 (there are 4 blues)

Total number of outcomes: 5 (there are 5 marbles in total)

$$\text{So the probability} = \frac{4}{5} = 0.8$$

Probability Tree

The **tree diagram** helps to organize and visualize the different possible outcomes. Branches and ends of the tree are two main positions. Probability of each branch is written on the branch, whereas the ends are containing the final outcome. Tree diagram is used to figure out when to multiply and when to add. You can see below a tree diagram for the coin:



Types of Probability

There are two major types of probabilities:

- Theoretical Probability
- Experimental Probability

Theoretical Probability

Theoretical Probability is what is expected to happen based on mathematics.

$$P(\text{event}) = \frac{\text{number of favorable outcomes}}{\text{total number of possible outcomes}}$$

Example:

A coin is tossed.

$$P(\text{head}) = \frac{1}{2}$$

$$P(\text{tail}) = \frac{1}{2}$$

Experimental Probability

Experimental Probability is found by repeating an experiment and observing the outcomes.

$$P(\text{event}) = \frac{\text{number of times event occurs}}{\text{total number of trials}}$$

Example:

A Coin is tossed 10 times: A head is recorded 7 times and a tail 3 times.

$$P(\text{head}) = \frac{7}{10}$$

$$P(\text{tail}) = \frac{3}{10}$$

Practical File

https://github.com/TopsCode/Data_Analytics/blob/main/Statistics/Probability.ipynb

Probability Distribution

A probability distribution is a statistical function that describes all the possible values and likelihoods that a random variable can take within a given range. This range will be bounded between the minimum and maximum possible values, but precisely where the possible value is likely to be plotted on the probability distribution depends on a number of factors. These factors include the distribution's mean (average), standard deviation, skewness.

Types of Distributions:

1. Bernoulli Distribution
2. Uniform Distribution
3. Binomial Distribution
4. Normal Distribution
5. Exponential Distribution

Bernoulli Distribution

All you cricket junkies out there! At the beginning of any cricket match, how do you decide who is going to bat or ball? A toss! It all depends on whether you win or lose the toss, right? Let's say if the toss results in a head, you win. Else, you lose. There's no midway.

A **Bernoulli distribution** has only two possible outcomes, namely 1 (success) and 0 (failure), and a single trial. So the random variable X which has a Bernoulli distribution can take value 1 with the probability of success, say p , and the value 0 with the probability of failure, say q or $1-p$.

the occurrence of a head denotes success, and the occurrence of a tail denotes failure.

Probability of getting a head = 0.5 = Probability of getting a tail since there are only two possible outcomes.

Bernoulli Distribution

The probability mass function is given by: $p^x(1-p)^{1-x}$ where $x \in \{0, 1\}$.

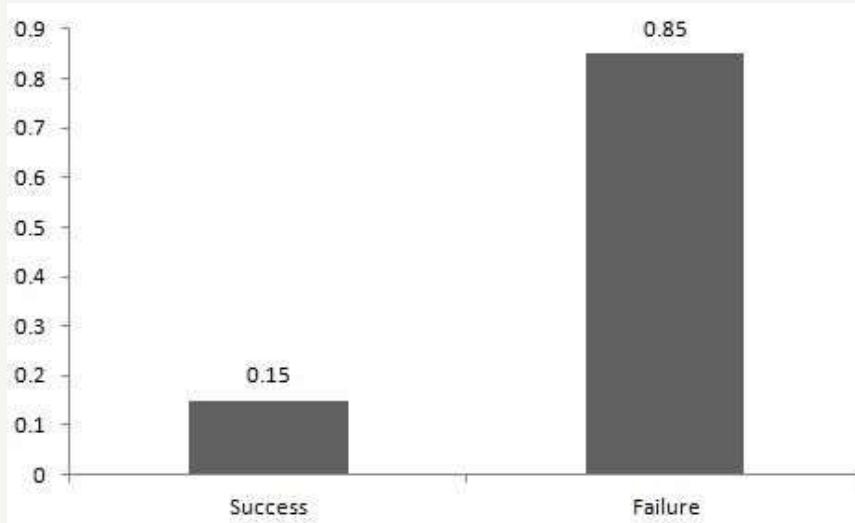
It can also be written as

$$P(x) = \begin{cases} 1 - p, & x = 0 \\ p, & x = 1 \end{cases}$$

The probabilities of success and failure need not be equally likely, like the result of a fight between me and Undertaker. He is pretty much certain to win. So in this case probability of my success is 0.15 while my failure is 0.85

Bernoulli Distribution

Here, the probability of success(p) is not same as the probability of failure. So, the chart below shows the Bernoulli Distribution of our fight.



Bernoulli Distribution

Here, the probability of success = 0.15 and probability of failure = 0.85. The expected value is exactly what it sounds. If I punch you, I may expect you to punch me back. Basically expected value of any distribution is the mean of the distribution. The expected value of a random variable X from a Bernoulli distribution is found as follows:

$$E(X) = 1*p + 0*(1-p) = p$$

The variance of a random variable from a bernoulli distribution is:

$$V(X) = E(X^2) - [E(X)]^2 = p - p^2 = p(1-p)$$

There are many examples of Bernoulli distribution such as whether it's going to rain tomorrow or not where rain denotes success and no rain denotes failure and Winning (success) or losing (failure) the game.

Binomial Distribution

Suppose that you won the toss today and this indicates a successful event. You toss again but you lost this time. If you win a toss today, this does not necessitate that you will win the toss tomorrow. Let's assign a random variable, say X , to the number of times you won the toss. What can be the possible value of X ? It can be any number depending on the number of times you tossed a coin.

There are only two possible outcomes. Head denoting success and tail denoting failure. Therefore, probability of getting a head = 0.5 and the probability of failure can be easily computed as: $q = 1 - p = 0.5$.

A distribution where only two outcomes are possible, such as success or failure, gain or loss, win or lose and where the probability of success and failure is same for all the trials is called a Binomial Distribution.

The outcomes need not be equally likely. Remember the example of a fight between me and Undertaker? So, if the probability of success in an experiment is 0.2 then the probability of failure can be easily computed as $q = 1 - 0.2 = 0.8$.

Binomial Distribution

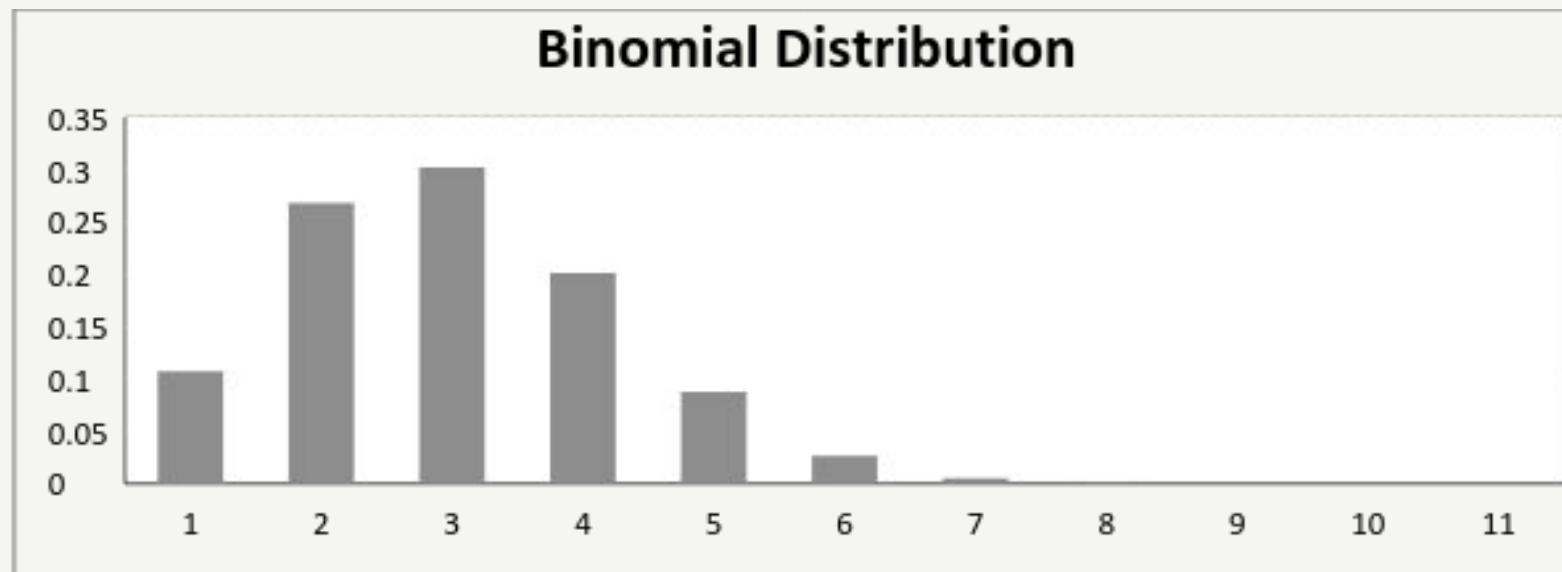
Each trial is independent since the outcome of the previous toss doesn't determine or affect the outcome of the current toss. An experiment with only two possible outcomes repeated n number of times is called binomial. The parameters of a binomial distribution are n and p where n is the total number of trials and p is the probability of success in each trial.

On the basis of the above explanation, the properties of a Binomial Distribution are

1. Each trial is independent.
2. There are only two possible outcomes in a trial- either a success or a failure.
3. A total number of n identical trials are conducted.
4. The probability of success and failure is same for all trials. (Trials are identical.)

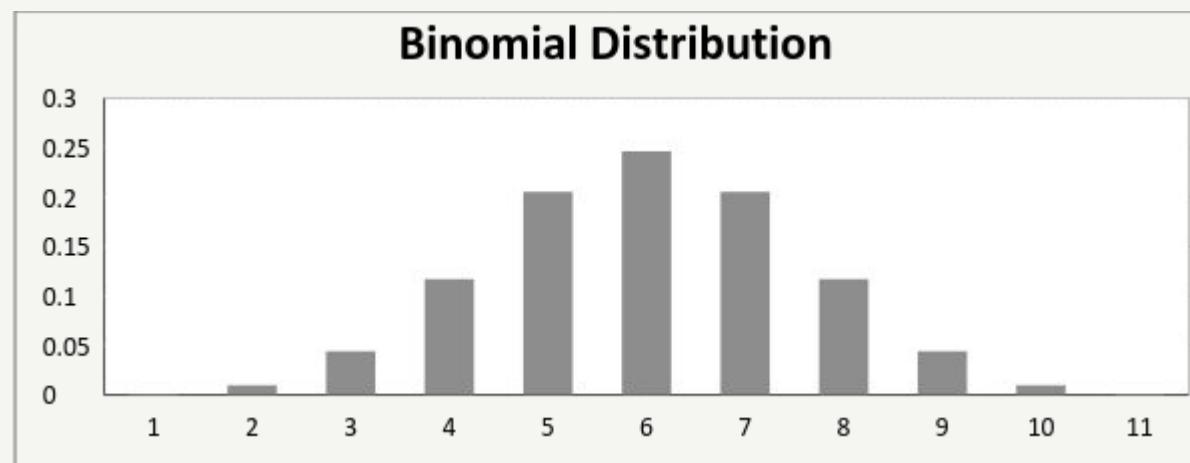
Binomial Distribution

A binomial distribution graph where the probability of success does not equal the probability of failure looks like



Binomial Distribution

Now, when probability of success = probability of failure, in such a situation the graph of binomial distribution looks like



Binomial Distribution

The mean and variance of a binomial distribution are given by:

$$\text{Mean} \rightarrow \mu = n * p$$

$$\text{Variance} \rightarrow \text{Var}(X) =$$

$$n * p * q$$

Geometric Distribution

The geometric distribution represents the number of failures before you get a success in a series of Bernoulli trials. This discrete probability distribution is represented by the probability density function.

$$f(x) = (1 - p)^{x-1} p = p * (1-p)^{(x-1)}$$

For example, you ask people outside a polling station who they voted for until you find someone that voted for the independent candidate in a local election. The geometric distribution would represent the number of people who you had to poll before you found someone who voted independent. You would need to get a certain number of failures before you got your first success.

Geometric Distribution

If you had to ask **3** people, then **X=3**; if you had to ask 4 people, then **X=4** and so on. In other words, there would be $X-1$ failures before you get your success.

If **X=n**, it means you succeeded on the n th try and failed for **n-1** tries. The probability of failing on your first try is **1-p**. For example, if **p = 0.2** then your probability of success is .2 and your probability of failure is **1 – 0.2 = 0.8**.

Independence (i.e. that the outcome of one trial does not affect the next) means that you can multiply the probabilities together. So the probability of failing on your second try is **(1-p)(1-p)** and your probability of failing on the $n-1$ tries is **(1-p)ⁿ⁻¹**. If you succeeded on your 4th try, **n = 4**, **n – 1 = 3**, so the probability of failing up to that point is **(1-p)(1-p)(1-p) = (1-p)³**.

Geometric Distribution

Example:-

If your probability of success is 0.2, what is the probability you meet an independent voter on your third try?

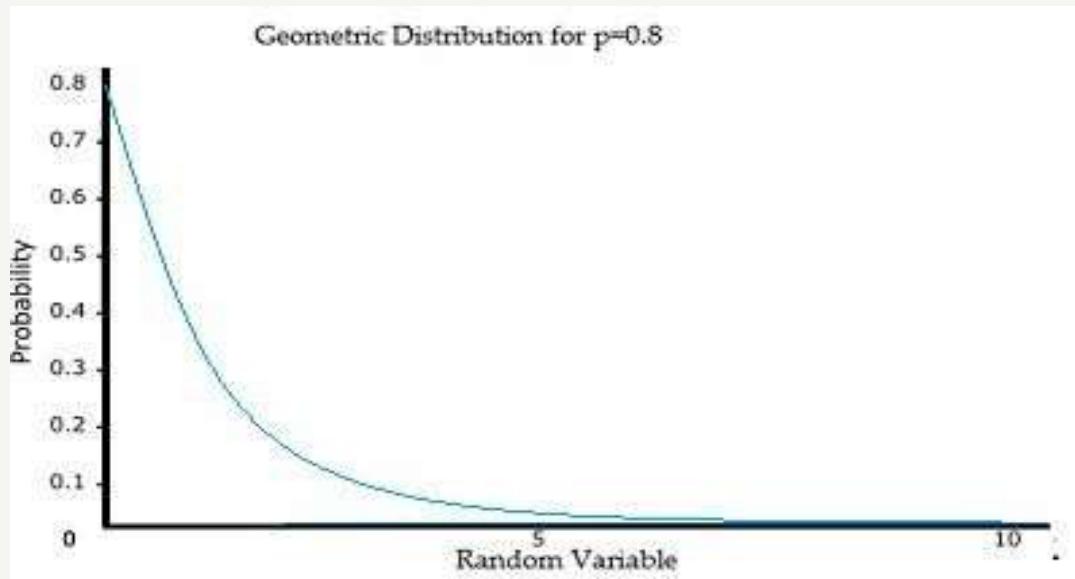
Inserting 0.2 as p and with X = 3, the probability density function becomes:

$$f(x) = (1 - p)^{x-1} * p$$

$$P(X=3) = (1 - 0.2)^3 - 1(0.2)$$

$$P(X=3) = (0.8)^2 * 0.2 = 0.128.$$

Geometric Distribution



Theoretically, there are an infinite number of geometric distributions. The value of any specific distribution depends on the value of the probability p .

Assumptions for the Geometric Distribution

The three assumptions are:

- There are two possible outcomes for each trial (success or failure).
- The trials are independent.
- The probability of success is the same for each trial.

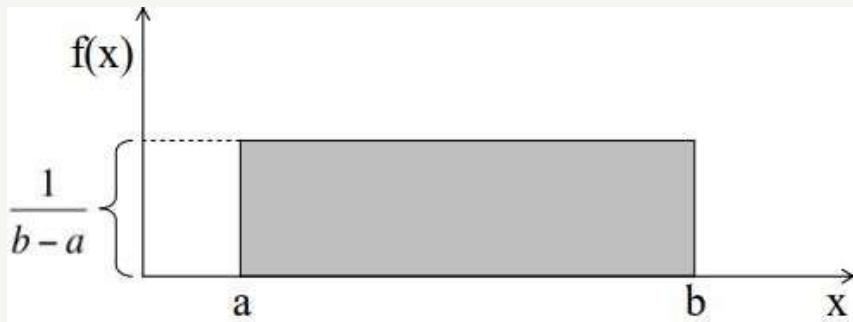
Uniform Distribution

When you roll a fair die, the outcomes are 1 to 6. The probabilities of getting these outcomes are equally likely and that is the basis of a uniform distribution. Unlike Bernoulli Distribution, all the n number of possible outcomes of a uniform distribution are equally likely.

A variable X is said to be uniformly distributed if the density function is:

$$f(x) = \frac{1}{b-a} \quad \text{for } -\infty < a \leq x \leq b < \infty$$

The graph of a uniform distribution curve looks like



Uniform Distribution

You can see that the shape of the Uniform distribution curve is rectangular, the reason why Uniform distribution is called rectangular distribution.

For a Uniform Distribution, a and b are the parameters.

Uniform Distribution

The number of bouquets sold daily at a flower shop is uniformly distributed with a maximum of 40 and a minimum of 10.

Let's try calculating the probability that the daily sales will fall between 15 and 30.

The probability that daily sales will fall between 15 and 30 is $(30-15)*(1/(40-10)) = 0.5$

Similarly, the probability that daily sales are greater than 20 is = 0.667

The mean and variance of X following a uniform distribution is:

$$\text{Mean} \rightarrow E(X) = (a+b)/2$$

$$\text{Variance} \rightarrow V(X) = (b-a)^2/12$$

The standard uniform density has parameters $a = 0$ and $b = 1$, so the PDF for standard uniform density is given by:

$$f(x) = \begin{cases} 1, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

Exponential Distribution

Let's consider the call center example one more time. What about the interval of time between the calls ? Here, exponential distribution comes to our rescue. Exponential distribution models the interval of time between the calls.

Other examples are:

1. Length of time between metro arrivals,
2. Length of time between arrivals at a gas station
3. The life of an Air Conditioner

Exponential distribution is widely used for survival analysis. From the expected life of a machine to the expected life of a human, exponential distribution successfully delivers the result.

Exponential Distribution

A random variable X is said to have an **exponential distribution** with PDF:

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

and parameter $\lambda > 0$ which is also called the rate.

For survival analysis, λ is called the failure rate of a device at any time t, given that it has survived up to t.

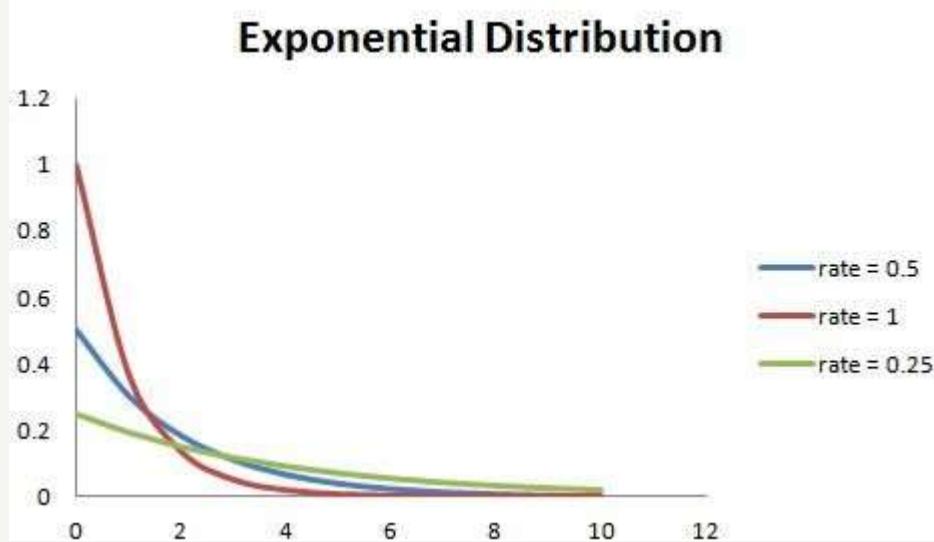
Mean and Variance of a random variable X following an exponential distribution:

$$\text{Mean} \rightarrow E(X) = 1/\lambda$$

$$\text{Variance} \rightarrow \text{Var}(X) = (1/\lambda)^2$$

Exponential Distribution

Also, the greater the rate, the faster the curve drops and the lower the rate, flatter the curve. This is explained better with the graph shown below.



Exponential Distribution

To ease the computation, there are some formulas given below.

$P\{X \leq x\} = 1 - e^{-\lambda x}$, corresponds to the area under the density curve to the left of x .

$P\{X > x\} = e^{-\lambda x}$, corresponds to the area under the density curve to the right of x .

$P\{x_1 < X \leq x_2\} = e^{-\lambda x_1} - e^{-\lambda x_2}$, corresponds to the area under the density curve between x_1 and x_2 .

Normal Distribution

Normal distribution represents the behavior of most of the situations in the universe (That is why it's called a "normal" distribution. I guess!). The large sum of (small) random variables often turns out to be normally distributed, contributing to its widespread application. Any distribution is known as Normal distribution if it has the following characteristics:

1. The mean, median and mode of the distribution coincide.
2. The curve of the distribution is bell-shaped and symmetrical about the line $x=\mu$.
3. The total area under the curve is 1.
4. Exactly half of the values are to the left of the center and the other half to the right.

A normal distribution is highly different from Binomial Distribution. However, if the number of trials approaches infinity then the shapes will be quite similar.

Normal Distribution

The PDF of a random variable X following a normal distribution is given by:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \quad \text{for } -\infty < x < \infty.$$

The mean and variance of a random variable X which is said to be normally distributed is given by:

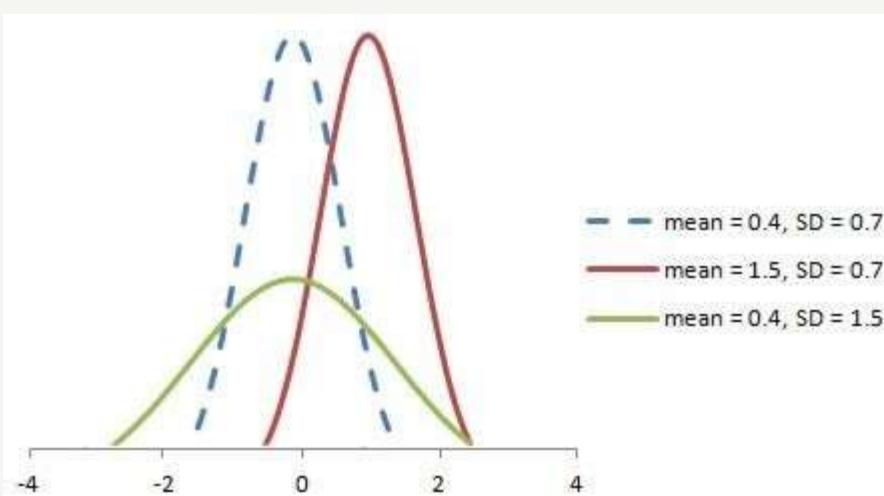
Mean -> $E(X) = \mu$

Variance -> $\text{Var}(X) = \sigma^2$

Normal Distribution

Here, μ (mean) and σ (standard deviation) are the parameters.

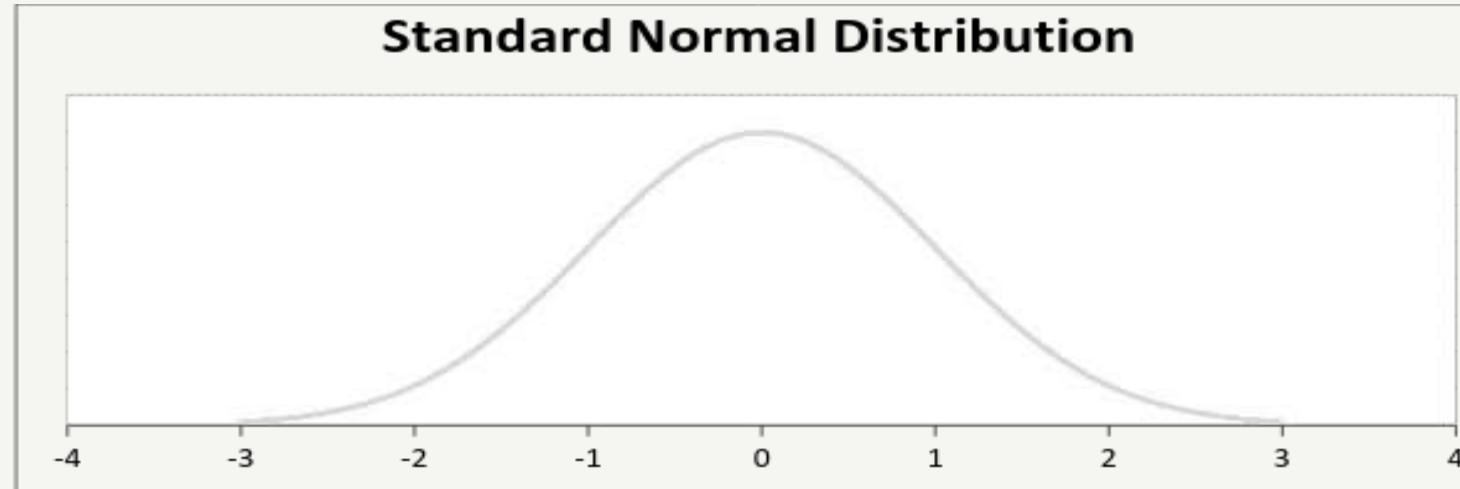
The graph of a random variable $X \sim N(\mu, \sigma)$ is shown below.



Normal Distribution

A standard normal distribution is defined as the distribution with mean 0 and standard deviation 1. For such a case, the PDF becomes:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \quad \text{for } -\infty < x < \infty$$



Practical File

https://github.com/TopsCode/Data_Analytics/blob/main/Statistics/Probability_Distribution.ipynb

Introduction of Inferential Statistics

It is about using data from sample and then making inferences about the larger population from which the sample is drawn. The goal of the inferential statistics is to draw conclusions from a sample and generalize them to the population. It determines the probability of the characteristics of the sample using probability theory. The most common methodologies used are hypothesis tests, Analysis of variance etc.

For example: Suppose we are interested in the exam marks of all the students in India. But it is not feasible to measure the exam marks of all the students in India. So now we will measure the marks of a smaller sample of students, for example 1000 students. This sample will now represent the large population of Indian students. We would consider this sample for our statistical study for studying the population from which it's deduced.



We evaluate 2 mutual exclusive statement on population data using sample data

Steps:

1. Make initial Assumption
2. Collecting data
3. Gather evidence to reject or not to reject the null Hypothesis .

What Is a Confidence Interval?

- A confidence interval, in statistics, refers to the probability that a population parameter will fall between a set of values for a certain proportion of times.
- Analysts often use confidence intervals than contain either 95% or 99% of expected observations.
- Thus, if a point estimate is generated from a statistical model of 10.00 with a 95% confidence interval of 9.50 – 10.50, it can be inferred that there is a 95% probability that the true value falls within that range.

Hypothesis Testing

Hypothesis testing is a tool for making statistical inferences about the population data. It is an analysis tool that tests assumptions and determines how likely something is within a given standard of accuracy. Hypothesis testing provides a way to verify whether the results of an experiment are valid.

A null hypothesis and an alternative hypothesis are set up before performing the hypothesis testing. This helps to arrive at a conclusion regarding the sample obtained from the population. In this article, we will learn more about hypothesis testing, its types, steps to perform the testing, and associated examples.

Alternative Hypothesis

The alternative hypothesis is an alternative to the null hypothesis. It is used to show that the observations of an experiment are due to some real effect. It indicates that there is a statistical significance between two possible outcomes and can be denoted as H_1 or H_a . For the above-mentioned example, the alternative hypothesis would be that girls are shorter than boys at the age of 5.

Types of Hypothesis Testing

Selecting the correct test for performing hypothesis testing can be confusing. These tests are used to determine a test statistic on the basis of which the null hypothesis can either be rejected or not rejected. Some of the important tests used for hypothesis testing are given below.

1. Hypothesis Testing Z Test
2. Hypothesis Testing t Test
3. Hypothesis Testing Chi Square

Hypothesis Testing Z Test

A z test is a way of hypothesis testing that is used for a large sample size ($n \geq 30$). It is used to determine whether there is a difference between the population mean and the sample mean when the population standard deviation is known. It can also be used to compare the mean of two samples.

It is used to compute the z test statistic. The formulas are given as follows:

- One sample: $Z \text{ Test} = (\bar{x} - \mu) / (\sigma / \sqrt{n})$

\bar{x} = Mean of Sample

μ = Mean of Population

σ = Standard Deviation of Population

n = Number of Observation

Hypothesis Testing t Test

The t test is another method of hypothesis testing that is used for a small sample size ($n < 30$). It is also used to compare the sample mean and population mean. However, the population standard deviation is not known. Instead, the sample standard deviation is known. The mean of two samples can also be compared using the t test.

- **One Sample Test :** $t = (\bar{x} - \mu) / (s / \sqrt{n})$

where

\bar{X} = Observed Mean of the Sample

μ = Theoretical Mean of the Population

s = Standard Deviation of the Sample

n = Sample Size

Hypothesis Testing Chi Square :

The Chi Square test is a hypothesis testing method that is used to check whether the variables in a population are independent or not.

It is used when the test statistic is chi-squared distributed.

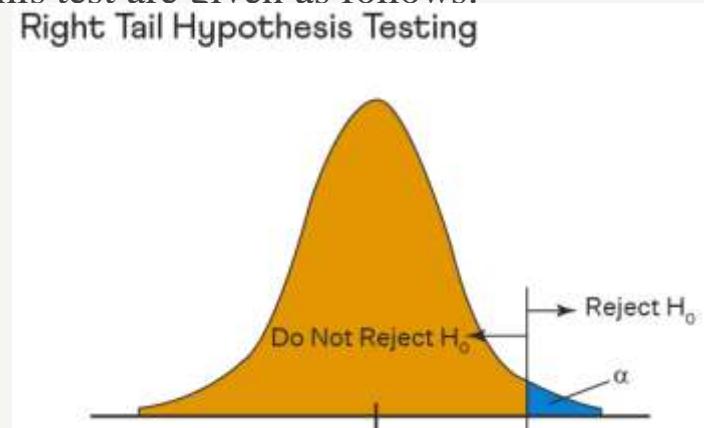
One Tailed Hypothesis Testing

One tailed hypothesis testing is done when the rejection region is only in one direction. It can also be known as directional hypothesis testing because the effects can be tested in one direction only. This type of testing is further classified into the right tailed test and left tailed test.

Right Tailed Hypothesis Testing

The right tail test is also known as the upper tail test. This test is used to check whether the population parameter is greater than some value. The null and alternative hypotheses for this test are given as follows:

Right Tail Hypothesis Testing



Left Tailed Hypothesis Testing

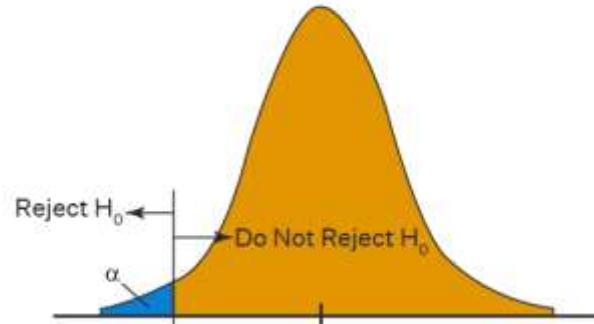
The left tail test is also known as the lower tail test. It is used to check whether the population parameter is less than some value. The hypotheses for this hypothesis testing can be written as follows:

H_0 : The population parameter is \geq some value

H_1 : The population parameter is $<$ some value.

The null hypothesis is rejected if the test statistic has a value lesser than the critical value.

Left Tail Hypothesis Testing



Two Tailed Hypothesis Testing

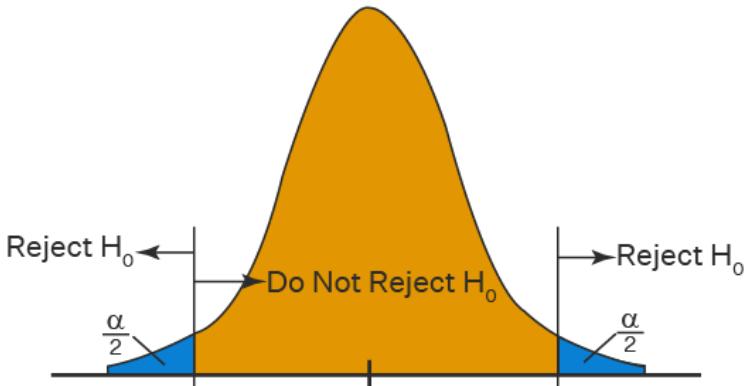
In this hypothesis testing method, the critical region lies on both sides of the sampling distribution. It is also known as a non - directional hypothesis testing method. The two-tailed test is used when it needs to be determined if the population parameter is assumed to be different than some value. The hypotheses can be set up as follows:

H_0 : the population parameter = some value

H_1 : the population parameter \neq some value

The null hypothesis is rejected if the test statistic has a value that is not equal to the critical value.

Two Tail Hypothesis Testing



Hypothesis Testing Steps :

Hypothesis testing can be easily performed in five simple steps. The most important step is to correctly set up the hypotheses and identify the right method for hypothesis testing. The basic steps to perform hypothesis testing are as follows:

Step 1: Set up the null hypothesis by correctly identifying whether it is the left-tailed, right-tailed, or two-tailed hypothesis testing.

Step 2: Set up the alternative hypothesis.

Step 3: Choose the correct significance level, α , and find the critical value.

Step 4: Calculate the correct test statistic (z, t or χ^2) and p-value.

Step 5: Compare the test statistic with the critical value or compare the p-value with α to arrive at a conclusion. In other words, decide if the null hypothesis is to be rejected or not.

Introduction to Data Analytics

- 1. What is Data Analytics?**
- 2. Importance of Data Analytics**
- 3. Types of Data Analytics**

What is Data Analytics?

**A domain focussed to
gain meaningful and
valuable insights from
the data available to do
predictions.**



Importance of Data Analytics

- Helps businesses optimize their performances.
- Analysis is done to study purchase patterns.
- This can also help in improving managerial operations and leverage organisations to next level.
- Data and information are increasing rapidly.

Types of Data Analytics

Descriptive analytics describes what has happened over a given period of time. Have the number of views gone up?

Diagnostic analytics focuses more on why something happened. Did the weather affect beer sales? Did that latest marketing campaign impact sales?

Predictive analytics moves to what is likely going to happen in the near term. What happened to sales the last time we had a hot summer?

Prescriptive analytics suggests a course of action.

Descriptive analytics

- Descriptive analytics is the interpretation of historical data to draw a better Comparison.
- Takes raw data and parses that data to draw conclusions.
- Return on Investment Capital (ROIC) is a descriptive analytic created by taking three data points:

net income, dividends, and total capital, and turning those data points into an easy-to-understand percentage that can be used to compare one company's performance to others.

Diagnostic analytics

Diagnostic analytics is a form of advanced analytics that examines data or content to answer the question, “Why did it happen?”

It is characterized by techniques such as **data discovery, data mining and correlations.**

Predictive analytics

Predictive analytics is the use of **data**, statistical algorithms and machine learning techniques to identify the likelihood of future outcomes based on historical **data**.

Prediction

Forecasting, etc



Prescriptive analytics

- Prescriptive analytics makes use of machine learning to help businesses decide a course of action based on a computer program's predictions.
- Prescriptive analytics works with predictive analytics, which uses data to determine near-term outcomes

Intro to types of Algorithms

Algorithms

1. Supervised Machine Learning

- Linear Regression
 - Simple Linear Regression
 - Multiple Linear Regression
- Logistic Regression

2. Unsupervised Machine Learning

- KNN (k-nearest neighbors)

Linear Regression

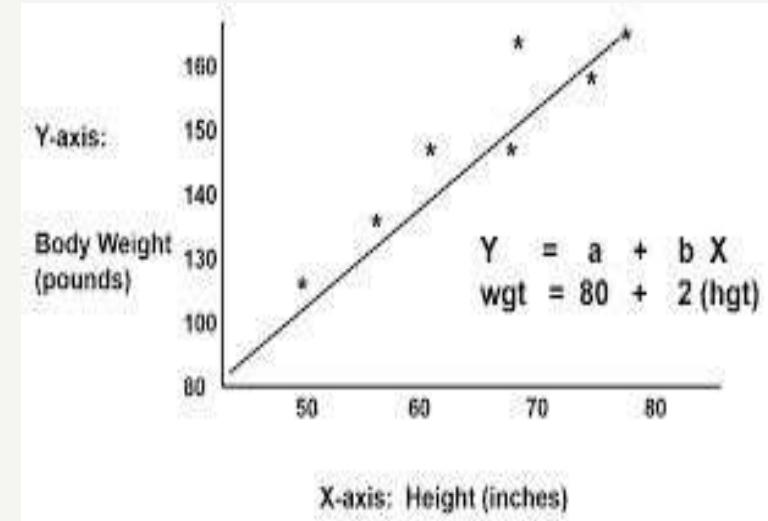
Simple Linear Regression:

y = dependent variable x = independent variable

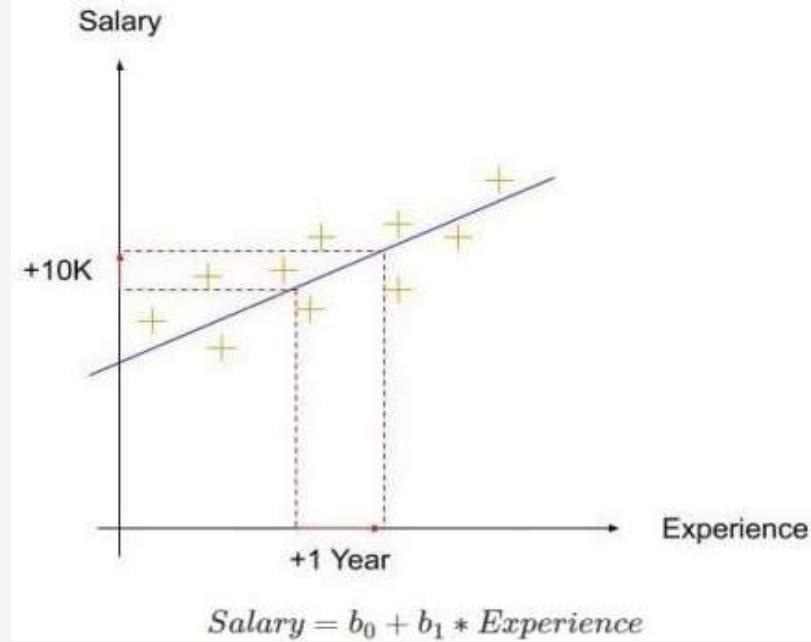
b₀ = constant

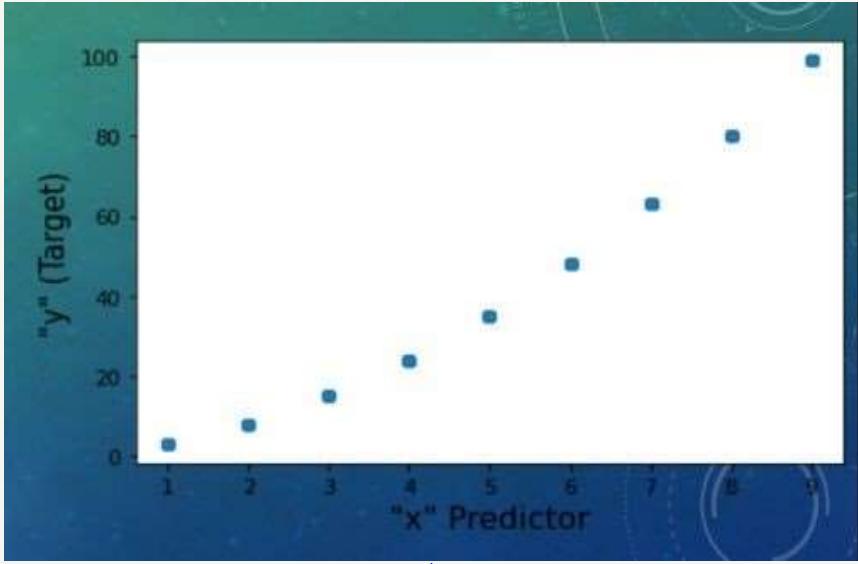
b₁ = determines how a unit change in x will make a unit change in y.

It is also known as the slope of the line which determines to which extent the change will inflate or deflate.



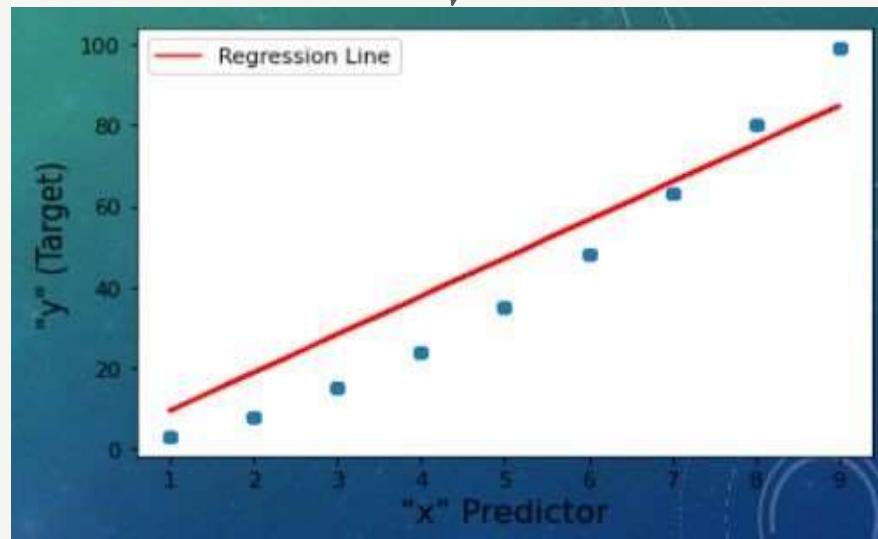
- Simple Linear Regression is used with data having only one feature and one label.
- SLR is more suited for data visualization as there are only two axes to plot the variables.





DATA

GOAL



train - test Split

```
>>>from sklearn.linear_model import train_test_split.  
x_train, x_test, y_train, y_test = train_test_split(x,y,  
test_size=0.2)
```

- Split arrays or matrices into random train and test subsets.
- “**test_size**” should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split.

Training Linear Regression

```
>>> from sklearn.linear_model import LinearRegression  
>>>model=LinearRegression()  
>>>model.fit(x_train, y_train)
```

Simple Linear Regression Practical

https://github.com/TopsCode/Data_Analytics/blob/main/machine_learning/Simple%20Linear%20.ipynb

$$MSE = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$$

- i - ith Sample
- \hat{y} - Predicted Value
- y - Actual Value

Gradients

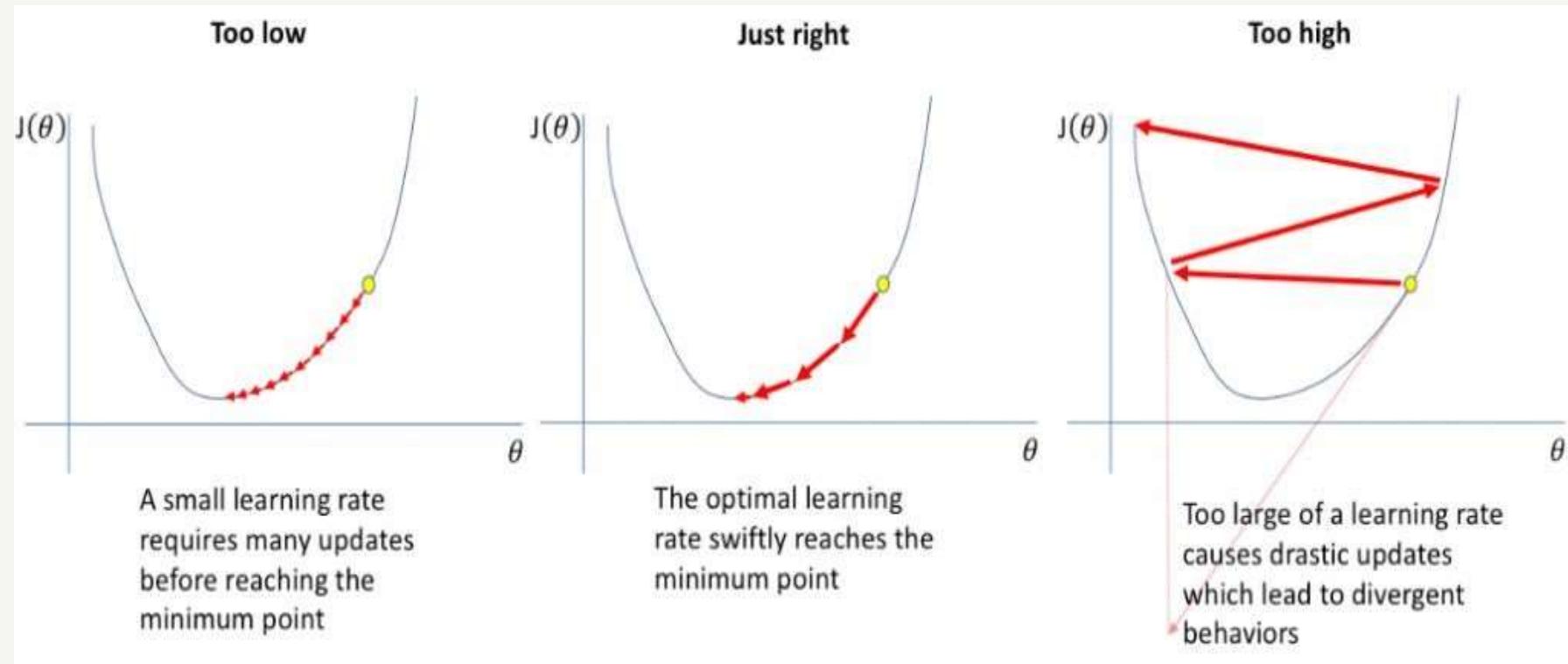
$$\frac{\partial}{\partial m} = \frac{2}{N} \sum_{i=1}^N -x_i(y_i - (mx_i + b))$$

$$\frac{\partial}{\partial b} = \frac{2}{N} \sum_{i=1}^N -(y_i - (mx_i + b))$$

predicted value = mx + b

Learning Rate (α)

Learning rate controls how quickly or slowly a model learns a problem.



Updating Parameters

```
> m = m - alpha*gradient_m  
> b = b - alpha*gradient_b
```

Practical For Gradient Descent

https://github.com/TopsCode/Data_Analytics/blob/main/machine_learning/Gradient.ipynb

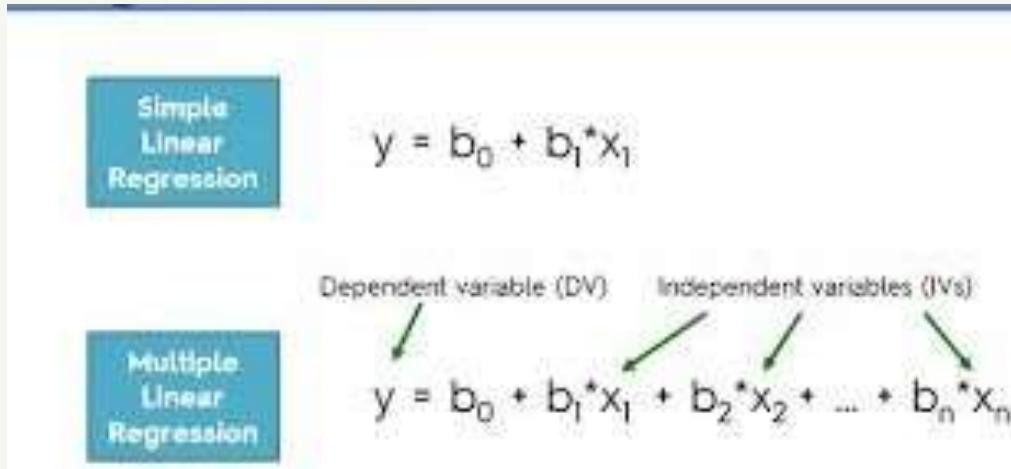
Multiple Linear Regression:

y = dependent variable

b0 = constant

b1, b2 ... bn = coefficients

x1, x2, ... xn = independent variables



Multiple Linear Regression

- Uses several explanatory(predictor) variables to predict the outcome of a response variable.

Necessary Assumption

- Linearity
- Homoscedasticity
- Multivariate Normality
- Independence of Error
- Lack of Multicollinearity

Practical for Multiple Linear Regression

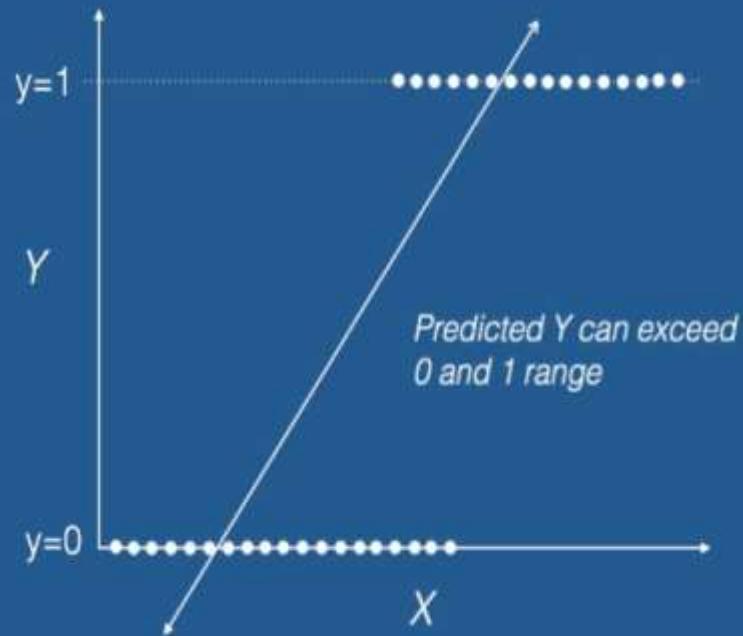
https://github.com/TopsCode/Data_Analytics/blob/main/machine_learning/MLR%2Bsubmission%2BCode.ipynb

Logistic Regression

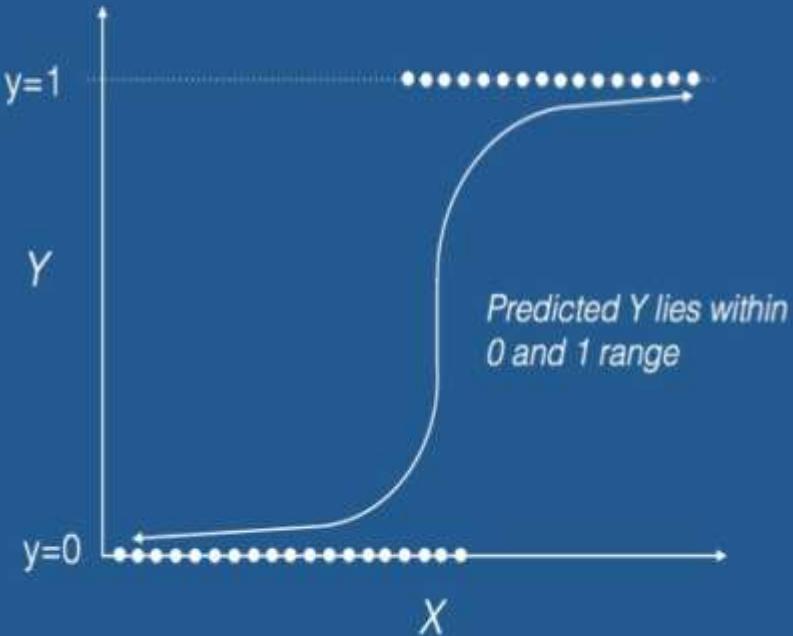
The **logistic regression** is a predictive analysis.

Logistic Regression, also known as **Logit Regression** or **Logit Model**, is a mathematical model used in statistics to estimate (guess) the **probability** of an event occurring having been given some previous data.

Linear Regression



Logistic Regression



Practical For Logistic Regression

https://github.com/TopsCode/Data_Analytics/blob/main/machine_learning/Logistic_JahnviPatel%20.ipynb

K Nearest Neighbour (KNN)

K Nearest Neighbors is a flexible approach to estimate the Bayes Classifier.

For any given X we find the K closest neighbors to X in the training data, and examine their corresponding Y.

If the Majority of the Y's are Orange we predict orange otherwise guess blue.

The Smaller that K is the more flexible the method will be.

MODULE 5

Analyzing Data with Python

Python Packages:

Python is an ocean of libraries that serve various purposes and as a Python developer, you must have sound knowledge of the best ones. To help you in this, here is an article that brings to you the Top 10 Python Libraries for machine learning which are:

1. Numpy
2. Pandas
3. Matplotlib
4. Scipy
5. Scikit-learn

1. NumPy

NumPy (Numerical Python) is the fundamental package for numerical computation in Python; it contains a powerful N-dimensional array object. It has around 18,000 comments on GitHub and an active community of 700 contributors. It's a general-purpose array-processing package that provides high-performance multidimensional objects called arrays and tools for working with them. NumPy also addresses the slowness problem partly by providing these multidimensional arrays as well as providing functions and operators that operate efficiently on these arrays.

Features:

- Provides fast, precompiled functions for numerical routines
- Array-oriented computing for better efficiency
- Supports an object-oriented approach
- Compact and faster computations with vectorization

Applications:

Extensively used in data analysis

Creates powerful N-dimensional array

Forms the base of other libraries, such as SciPy and scikit-learn

Replacement of MATLAB when used with SciPy and matplotlib

2. Pandas (Python data analysis)

- Pandas is a must in the data science life cycle. It is the most popular and widely used Python library for data science, along with NumPy in matplotlib. With around 17,000 comments on GitHub and an active community of 1,200 contributors, it is heavily used for data analysis and cleaning. Pandas provides fast, flexible data structures, such as data frame CDs, which are designed to work with structured data very easily and intuitively.

Features:

- Eloquent syntax and rich functionalities that gives you the freedom to deal with missing data
- Enables you to create your own function and run it across a series of data
- High-level abstraction
- Contains high-level data structures and manipulation tools

Applications:

- General data wrangling and data cleaning
- ETL (extract, transform, load) jobs for data transformation and data storage, as it has excellent support for loading CSV files into its data frame format
- Used in a variety of academic and commercial areas, including statistics, finance and neuroscience
- Time-series-specific functionality, such as date range generation, moving window, linear regression and date shifting.

Other Packages:

Matplotlib: This library is responsible for plotting numerical data. And that's why it is used in data analysis. It is also an open-source library and plots high-defined figures like pie charts, histograms, scatterplots, graphs, etc.

SciPy: The name “SciPy” stands for “Scientific Python”. It is an open-source library used for high-level scientific computations. This library is built over an extension of Numpy. It works with Numpy to handle complex computations. While Numpy allows sorting and indexing of array data, the numerical data code is stored in SciPy. It is also widely used by application developers and engineers.

Scikit-learn: It is a famous Python library to work with complex data. Scikit-learn is an open-source library that supports machine learning. It supports variously supervised and unsupervised algorithms like linear regression, classification, clustering, etc. This library works in association with Numpy and SciPy.

What is Data Visualization

Data Viz is The Representation of data or information in a graph, charts or Other Visual Format . its just deals with the Graphic Representation of the data .

It communicates the relationship of the data with images .

Connects thousands and millions and lines of numbers with the nice visual image to effectively analyze it .

Importance of Viz

This is important because it allows trends and patterns to be more easily seen.

With the rise of big data upon us, we need to be able to interpret increasingly larger batches of data.

Machine learning makes it easier to conduct analyses such as predictive analysis, which can then serve as helpful visualizations to present.

Data Viz is not only related with Data Scientist and Data Analytics , This Skills Can be used at almost every filed to analyse the data

Need of Visualization

We need data visualization because a visual summary of information makes it easier to identify patterns and trends than looking through thousands of rows on a spreadsheet. It's the way the human brain works.

Since the purpose of data analysis is to gain insights, data is much more valuable when it is visualized.

Even if a data analyst can pull insights from data without visualization, it will be more difficult to communicate the meaning without visualization. Charts and graphs make communicating data findings easier

Use cases of data Viz

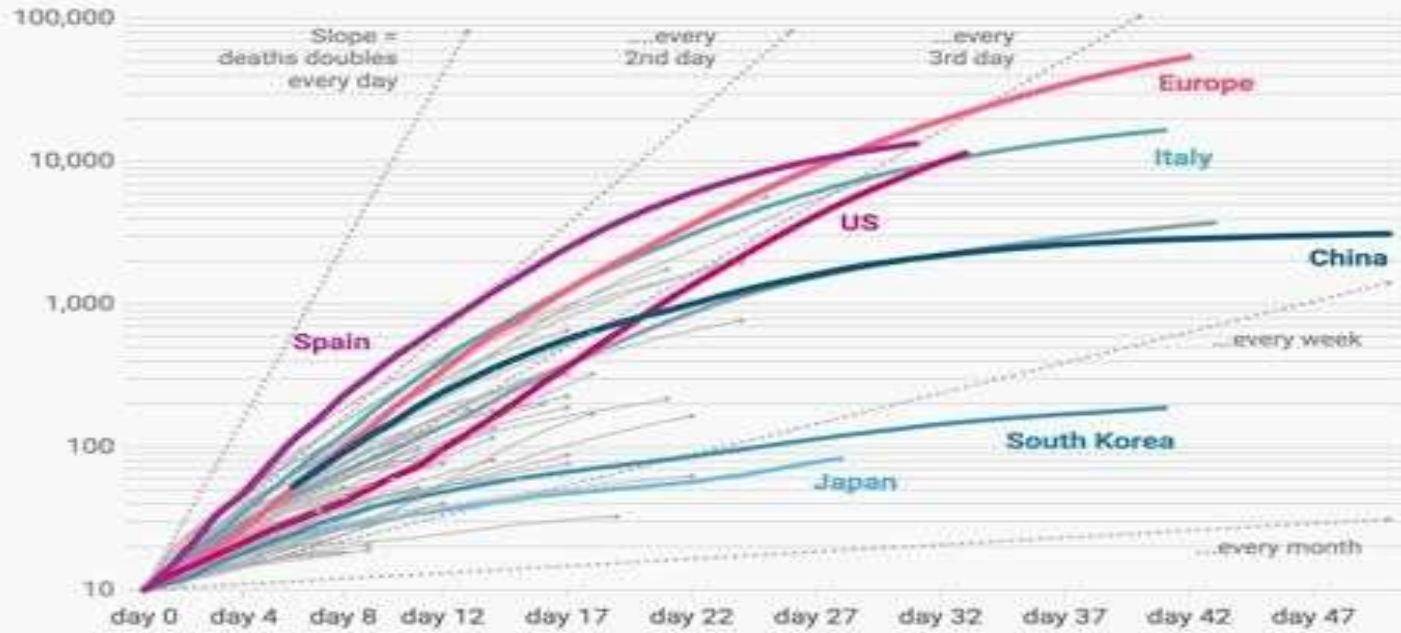
1. Identifying Trends and Spikes
2. Monitoring Goals and Results
3. Getting Notified When Changes Occur
4. Aggregating Diverse Data Sets
5. Accessing Data Displays Remotely

And Many More.....



**In the US, confirmed coronavirus deaths slow start to flatten.
China & South Korea have already a flatter curve.**

Cumulative numbers of confirmed deaths due to the COVID-19 disease, in selected countries after the 10th death, last updated with the numbers from yesterday, smoothed with a seven day rolling average. **The helper slopes don't mark corridors.** For example, Europe has a doubling rate of less than three days now – the "...every 3rd day" slope is steeper than the Europe line.



The chart stops at day 45, but more than 60 days have passed since China reported their 10th death. China reports a bit over 3,200 deaths to date. The 10th death in all countries is aligned and therefore approximated. The source didn't report values for China before March 22, so the numbers of days passed before that are estimated.

Chart: Lisa Charlotte Rost, Datawrapper · Source: Johns Hopkins CSSE · Get the data · Created with Datawrapper

We don't show cases in this chart because they're not as reliably tracked as deaths. You can find a chart showing the doubling rate of confirmed cases here and in our River. It gets updated every day, but we don't recommend using it.

What is data Visualization Tool

Its a software that takes data from a specific source and turns it into visual charts , graphs , dashboard and reports

There are multiple Data Visualization tools available in market ,

Some are..

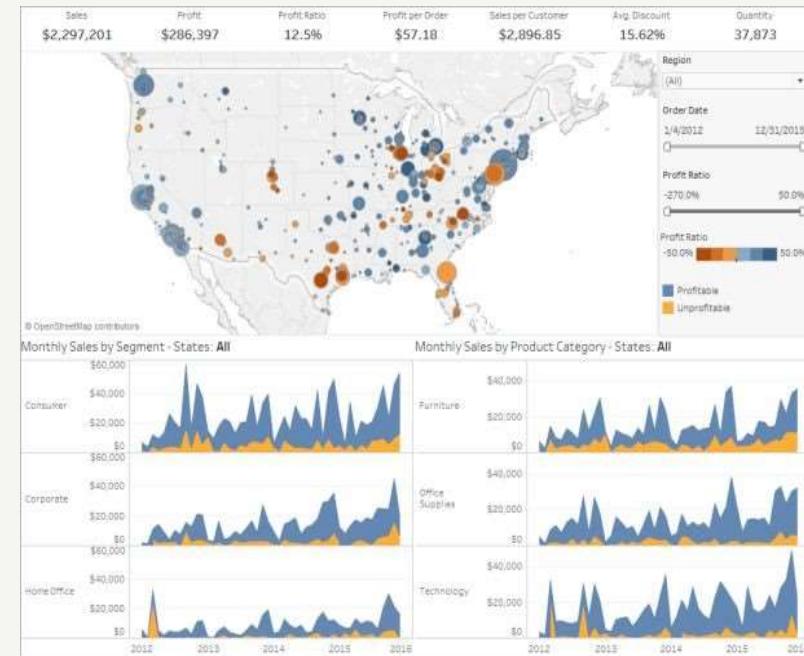
1. **Visme**
2. **Tableau**
3. **Power Bi**
4. **Infogram**
5. **Whatagraph**
6. **Sisense**
7. **DataBox**

And Many More.....

Why tableau

1. Quick And Interactive Visualization

Specialization in beautiful visualizations , it offers instantaneous insights with simple drag and drop feature , thus helping you easily analyze key data and share crucial insights .



2. Easy To use

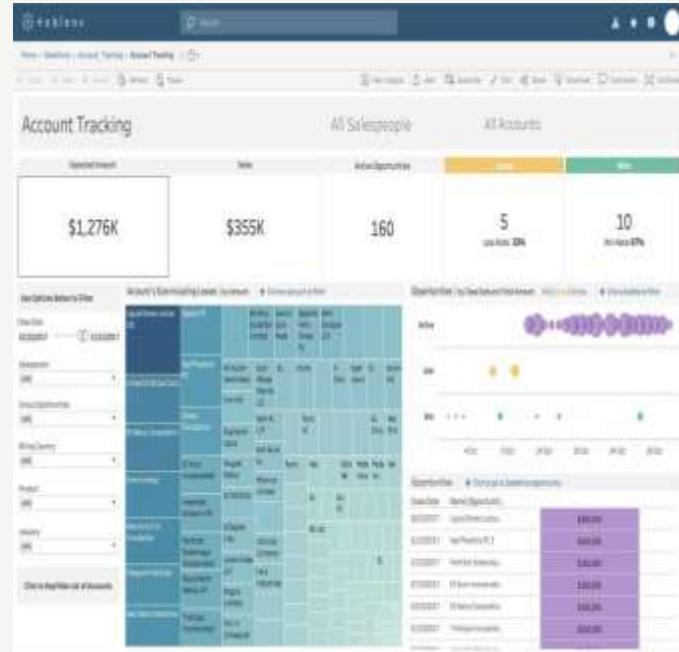
Compare to other BI tools , Tableau lets you create rich visualization in just a few seconds . It lets you perform complex tasks with simple drag-and-drop functionalities ,



3. Handles Copious amounts of data

If you need all the records of past 8-10 years ?
Tableau can handle millions of rows of data without impacting the performance of the dashboard .

It can connect to live data source to provide companies with the real-time results on some key business metrics .



4. Mobile Friendly Dashboard

Tableau dashboards can be viewed and operated on several devices such as your laptop, mobile, or even a tablet! You aren't required to perform any additional steps in order to make your dashboards mobile-friendly. Tableau automatically understands the device that you're viewing the report on and makes adjustments accordingly.

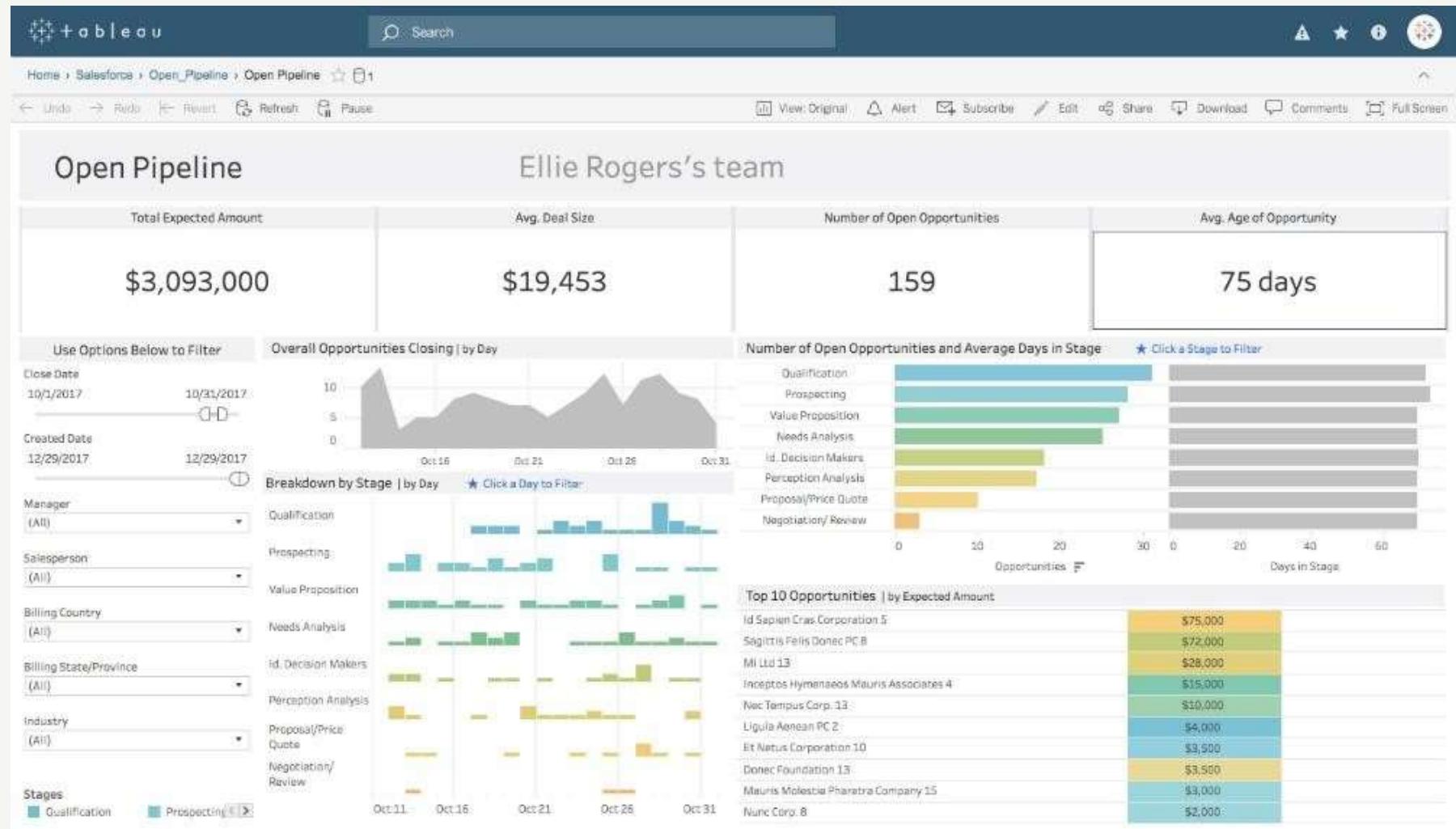


5. Integrates with Scripting Languages

The BI tool lets you integrate with R or Python, thus helping you amplify data with visual analytics.



Tableau Example dashboard



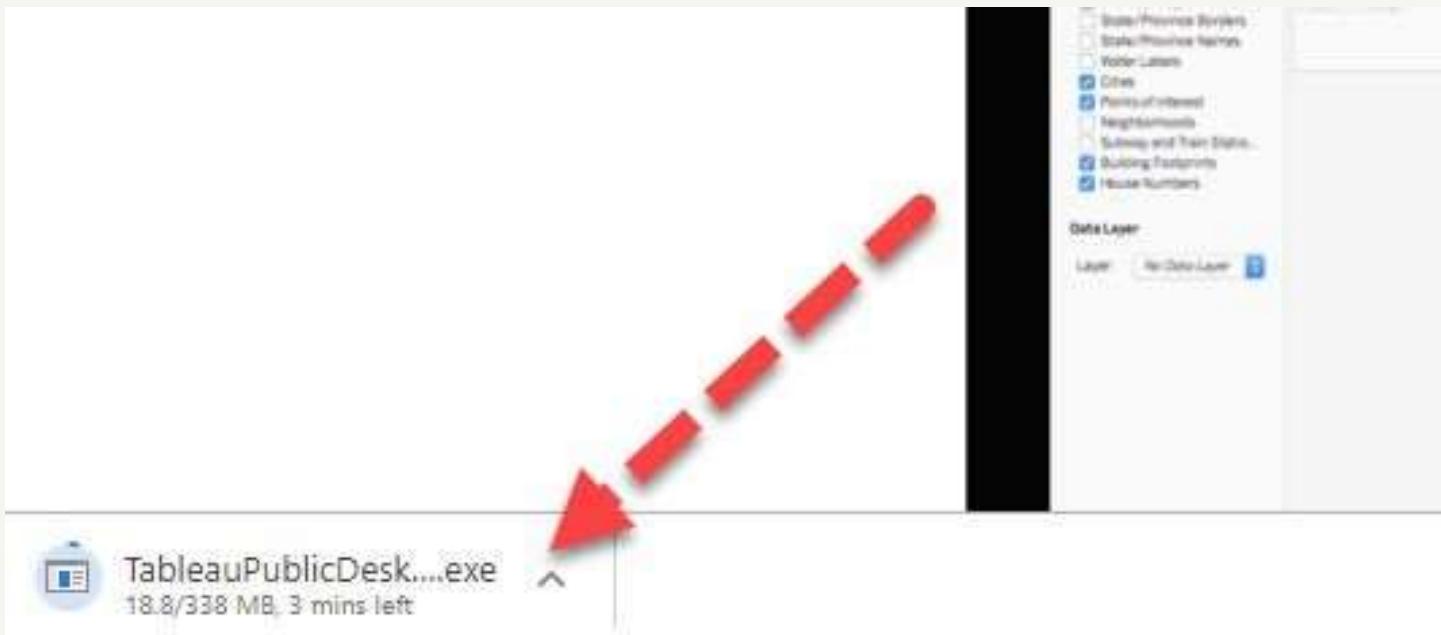


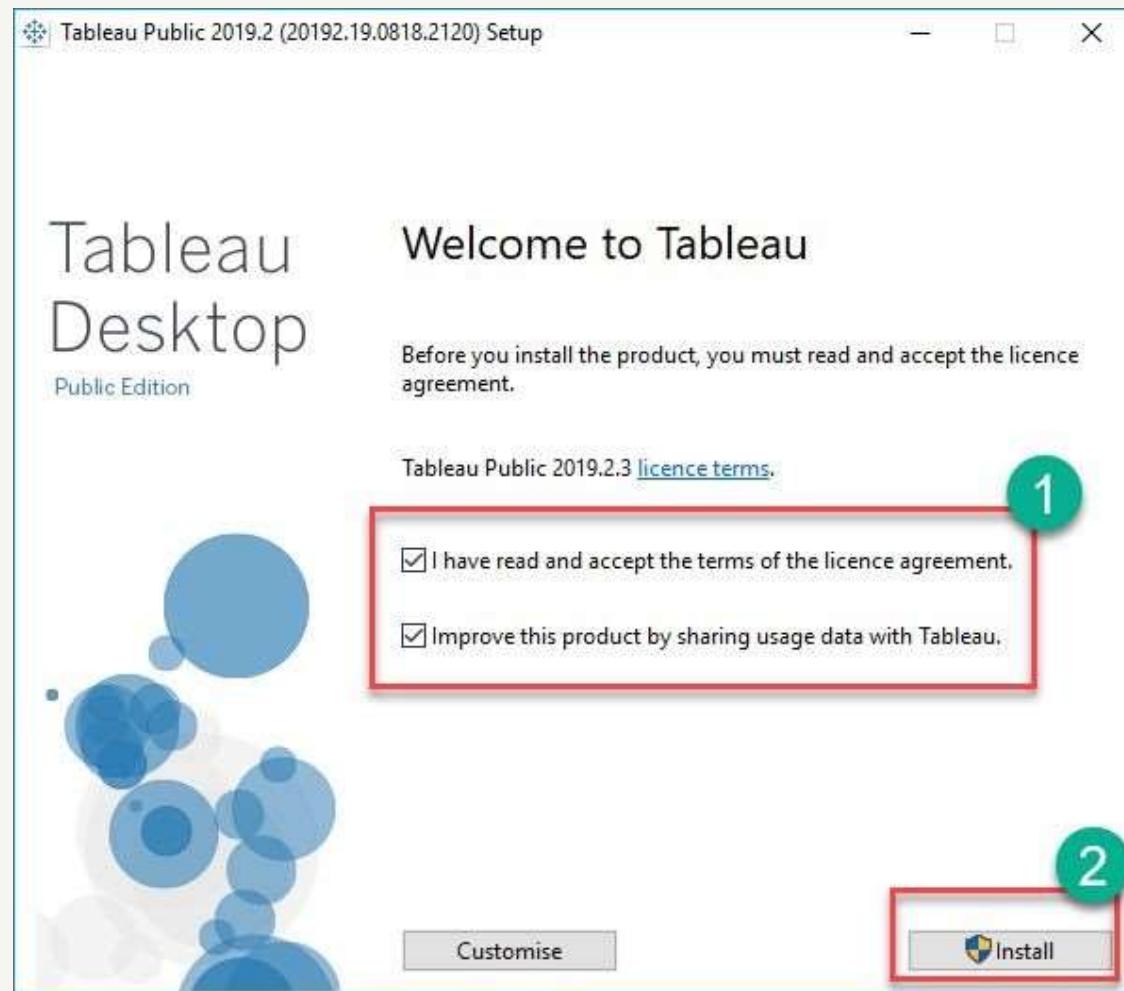
Installing Tableau

Tableau Public

GOTO : <https://public.tableau.com/en-us/s/download> and enter the email ID an download the app







Except the
terms and
conditions

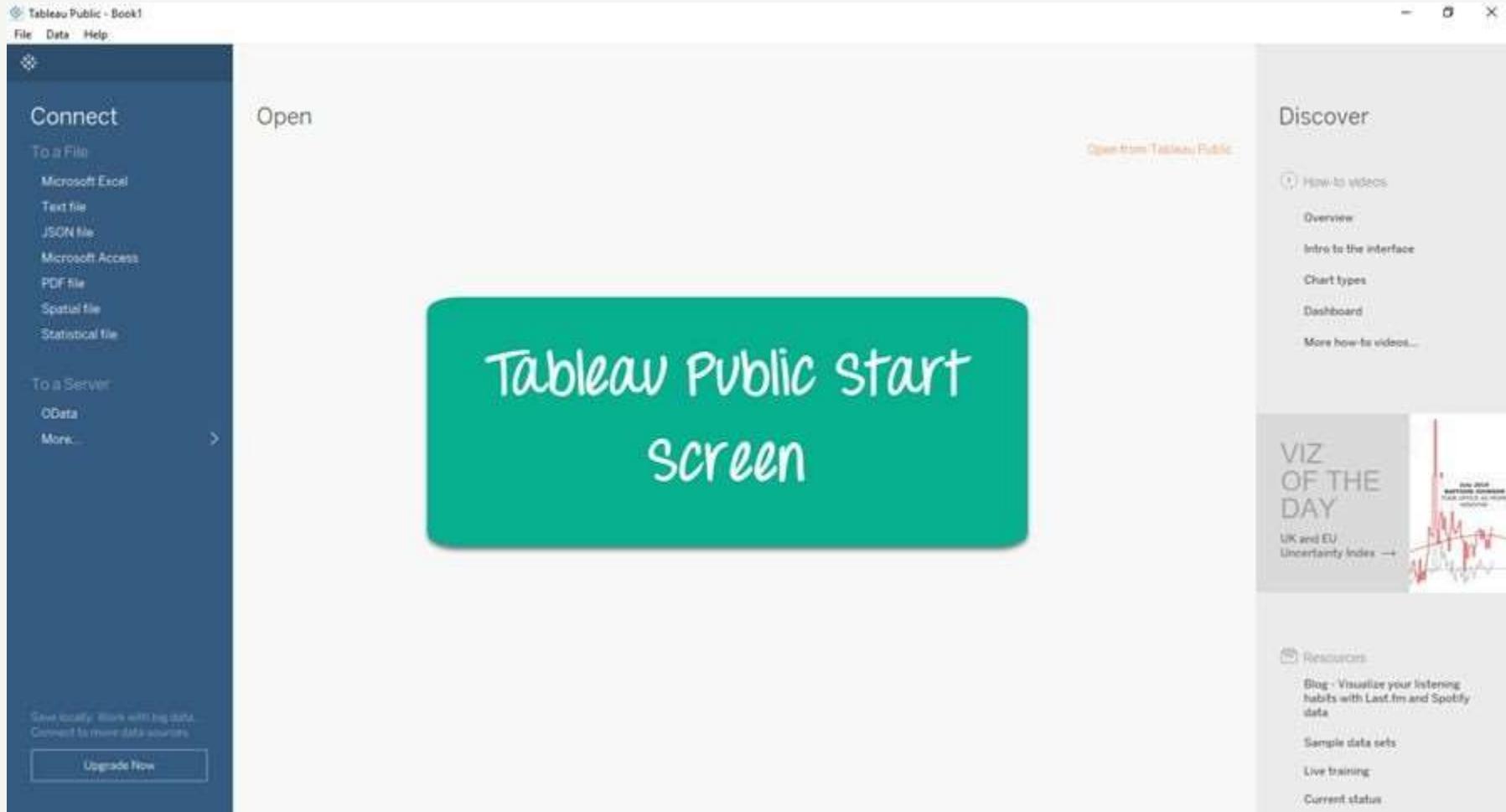
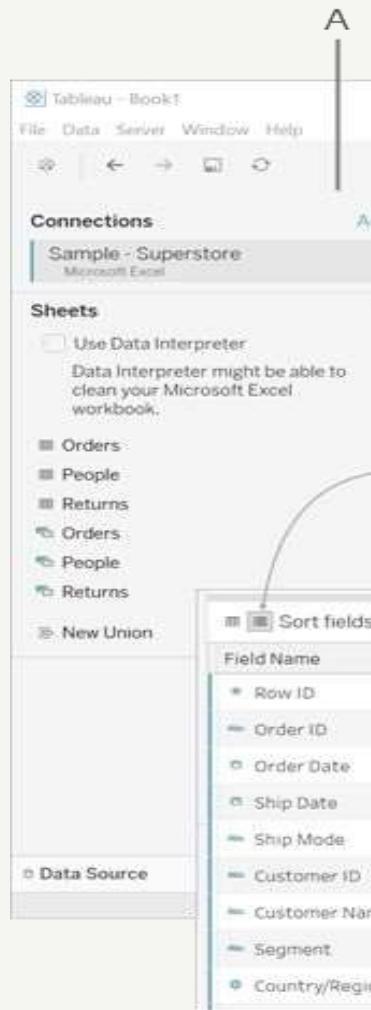


Tableau Interface

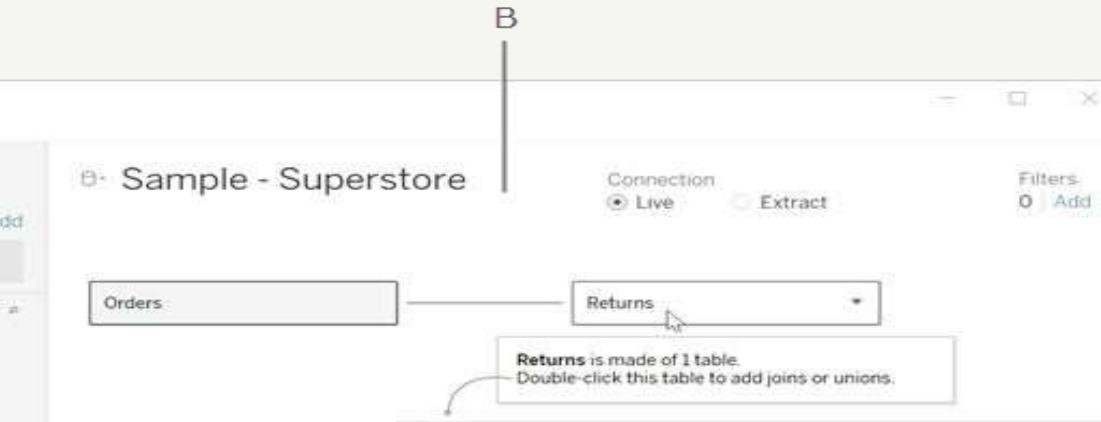
A



The screenshot shows the Tableau Data Source pane. It lists a connection named "Sample - Superstore" from "Microsoft Excel". Under "Sheets", there are several options: "Orders", "People", "Returns", "Orders", "People", "Returns", and "New Union". A checkbox for "Use Data Interpreter" is checked, with a note below stating: "Data Interpreter might be able to clean your Microsoft Excel workbook." Below this is a table titled "Data source order" with columns: Row ID, Order ID, Order Date, and Order D.

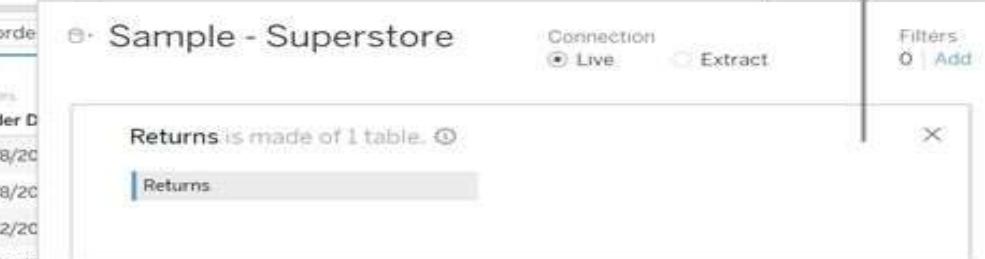
Field Name	Table	Remote Field Name
Row ID	Orders	Row ID
Order ID	Orders	Order ID
Order Date	Orders	Order Date
Ship Date	Orders	Ship Date
Ship Mode	Orders	Ship Mode
Customer ID	Orders	Customer ID
Customer Name	Orders	Customer Name
Segment	Orders	Segment
Country/Region	Orders	Country/Region

B



The screenshot shows the Tableau Data Source pane with a connection to "Sample - Superstore". The "Orders" sheet is selected. A tooltip indicates: "Returns is made of 1 table. Double-click this table to add joins or unions." A callout arrow points from this tooltip to the "Returns" sheet in the list.

C



The screenshot shows the Tableau Data Source pane with the "Returns" sheet selected. The tooltip "Returns is made of 1 table." is displayed again. A callout arrow points from this tooltip to the "Returns" table in the list.

D



The screenshot shows the Tableau Data Source pane with the "Returns" table selected. The tooltip "Returns is made of 1 table." is displayed again. A callout arrow points from this tooltip to the "Returns" table in the list.

E

The screenshot shows the Tableau Data Source pane with the "Returns" table selected. The tooltip "Returns is made of 1 table." is displayed again. A callout arrow points from this tooltip to the "Returns" table in the list.

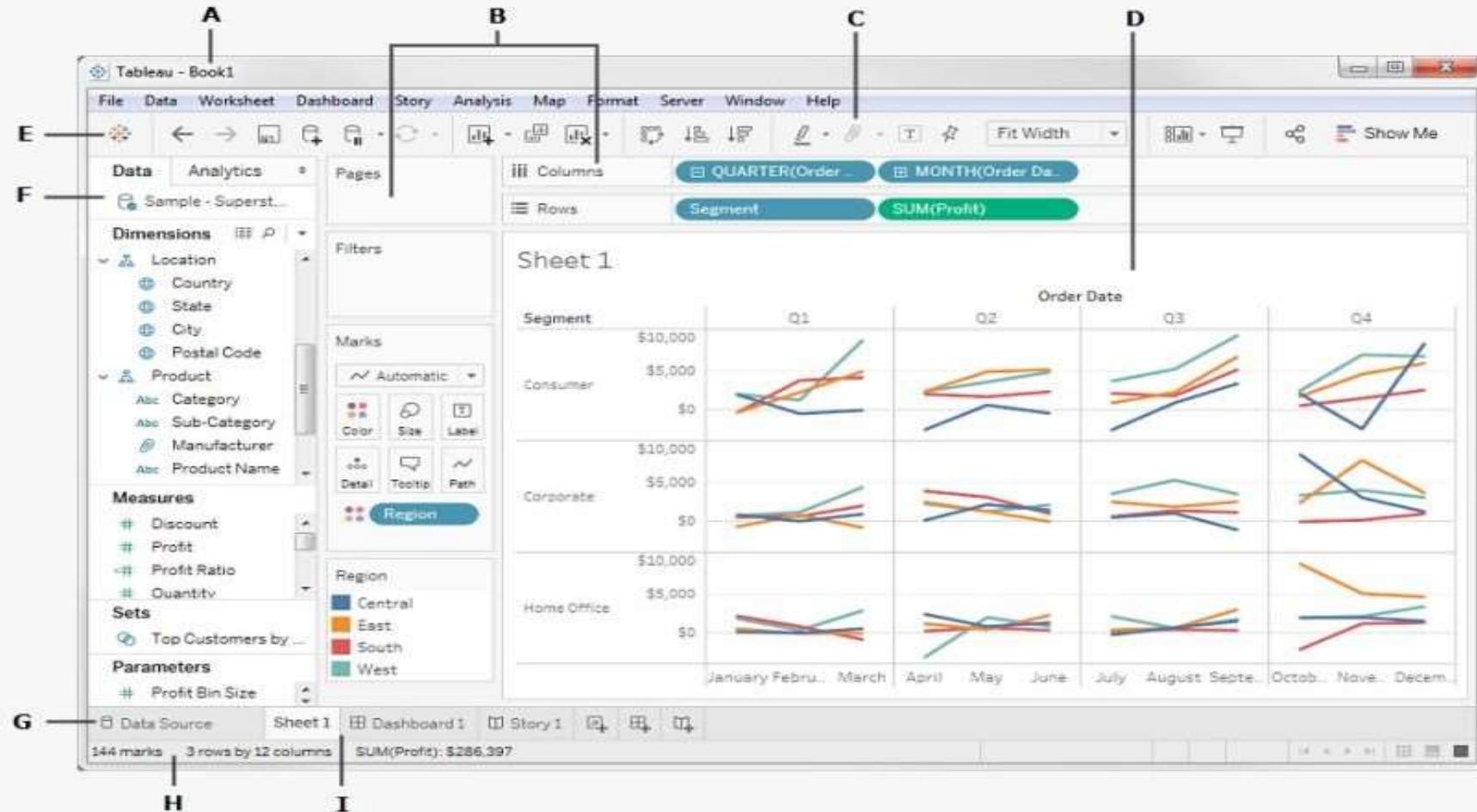
A = Left pane – Displays the connected data source and other details about your data.

B = Canvas: logical layer - The canvas opens with the logical layer, where you can create relationships between logical tables

C = Canvas: physical layer – Double-click a table in the logical layer to go to the physical layer of the canvas, where you can add joins and unions between tables.

D = Data grid – Displays first 1,000 rows of the data contained in the Tableau data source.

E = Metadata grid – Displays the fields in your data source as rows.



A = Workbook name. A workbook contains sheets. A sheet can be a worksheet, a dashboard, or a story.

B = Drag fields to the cards and shelves in the workspace to add data to your view

C = Use the toolbar to access commands and analysis and navigation tools

D = This is the canvas in the workspace where you create a visualisation (also referred to as a "viz").

E = Click this icon to go to the Start page, where you can connect to data. For more information

F = Side Bar - In a worksheet, the side bar area contains the Data pane and the Analytics pane.

G = Click this tab to go to the Data Source page and view your data.

H = Status bar - Displays information about the current view

I = Sheet tabs - Tabs represent each sheet in your workbook. This can include worksheets, dashboards and stories.

Data Types

There are primarily seven data types used in Tableau. Tableau automatically detects the data types of various fields as soon as new data gets uploaded from source to Tableau and assigns it to the fields. You can also modify these data types after uploading your data into Tableau .

1. String values
2. Number (Integer) values
3. Date values
4. Date & Time values
5. Boolean values
6. Geographic values
7. Cluster or mixed values

Data Type	Icon
Text (string) values	Abc
Date Values	📅
Date & Time Values	🕒
Numerical Values	#
Boolean Values	T F
Geographic Values	🌐
Cluster Group	⌚

Connecting To Data Source

One of the basic operation we need to learn is to connect to a data source.

Once we establish a successful connection with a data source, we can access all its data, bring some part of it in Tableau's repository (extract) and use it for our analysis.

Tableau offers a myriad of data sources such as local text files, MS Excel, PDFs, JSON or databases and servers like Tableau Server, MySQL Server, Microsoft SQL Server, etc.

Categorically, there are two types of data sources that you can connect to in Tableau;
To a file and **To a server**.

Connecting To A File

Tableau offers a variety of options to connect and get data from a file in your system.

The connection to a file section has file options such as MS Excel, MS Access, JSON, text file, PDF file, spatial file, etc.

In addition to this, with the help of the More option, you can access the data files residing in your system and connect them with Tableau.

Tableau - Book1 - Tableau license expires in 7 days

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Data Analytic Connect to Data

Dimensions

Connect

To a File

- Microsoft Excel
- Text file
- JSON file
- Microsoft Access
- PDF file
- Spatial file
- Statistical file
- More...

More...

To a Server

- Tableau Server
- Microsoft SQL Server
- MySQL
- Oracle
- Amazon Redshift
- More... >

Saved Data Sources

- World Indicators

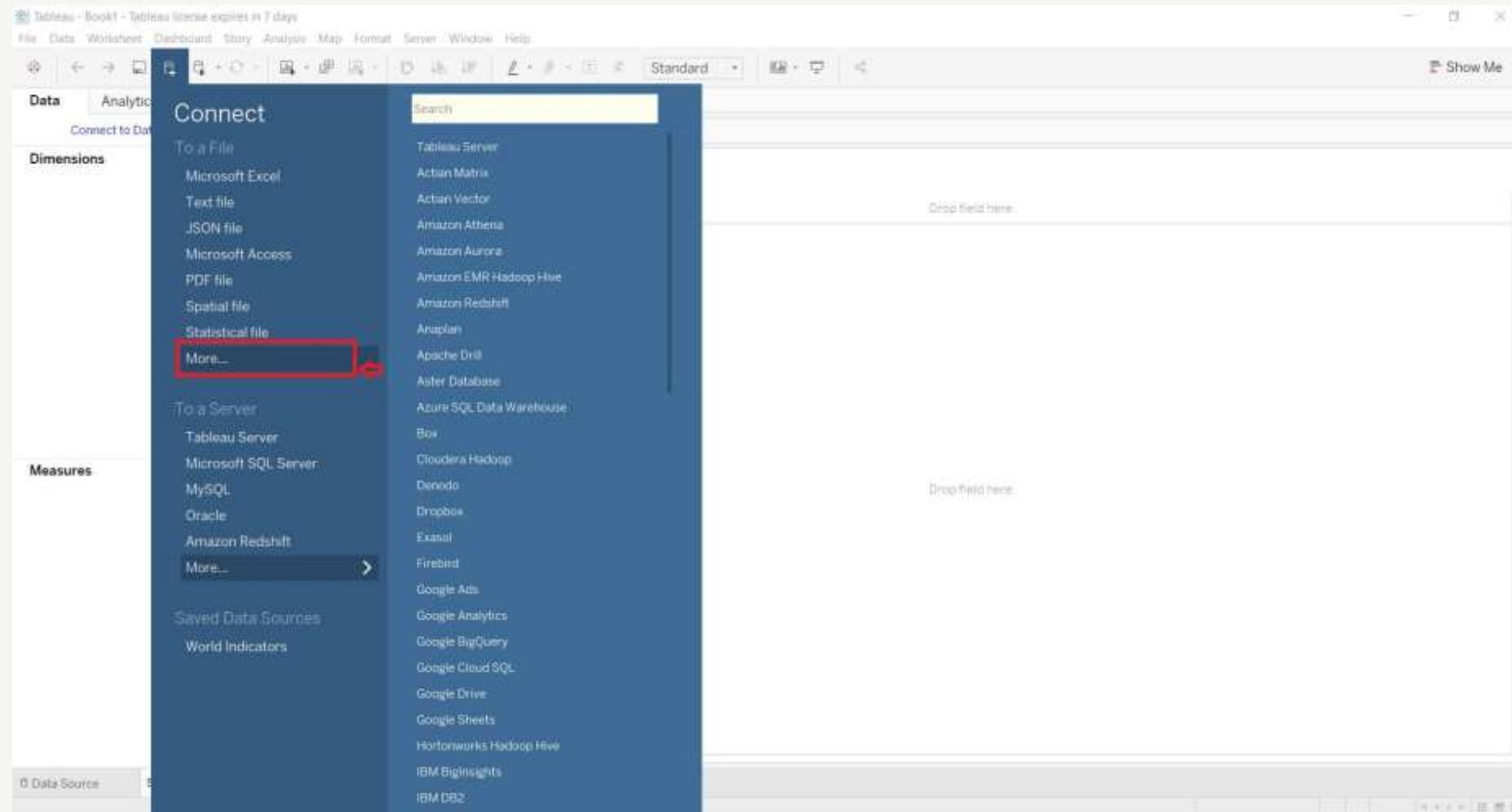
Measures

Drop field here.

Drop field here.

Drop field here.

0 Data Source



Connecting To A Server

The connection to a server section has countless options for an online data source. Here you will find connectors to different kinds of online data sources such as,

- Server-based Relational Database: Tableau Server, Microsoft SQL Server, Oracle, MySQL, Salesforce, IBM DB, Mongo DB, PostgreSQL, Maria DB, etc.
- Cloud-based Data Sources: Cloudera Hadoop, Google Cloud SQL, Amazon Aurora, etc.
- Web-based Data Sources: Web Data Connector
- Big Data Sources: Google BigQuery
- In-memory Database: *SAP HANA*
- ODBC and JDBC connections

Connect to DataBase Practically

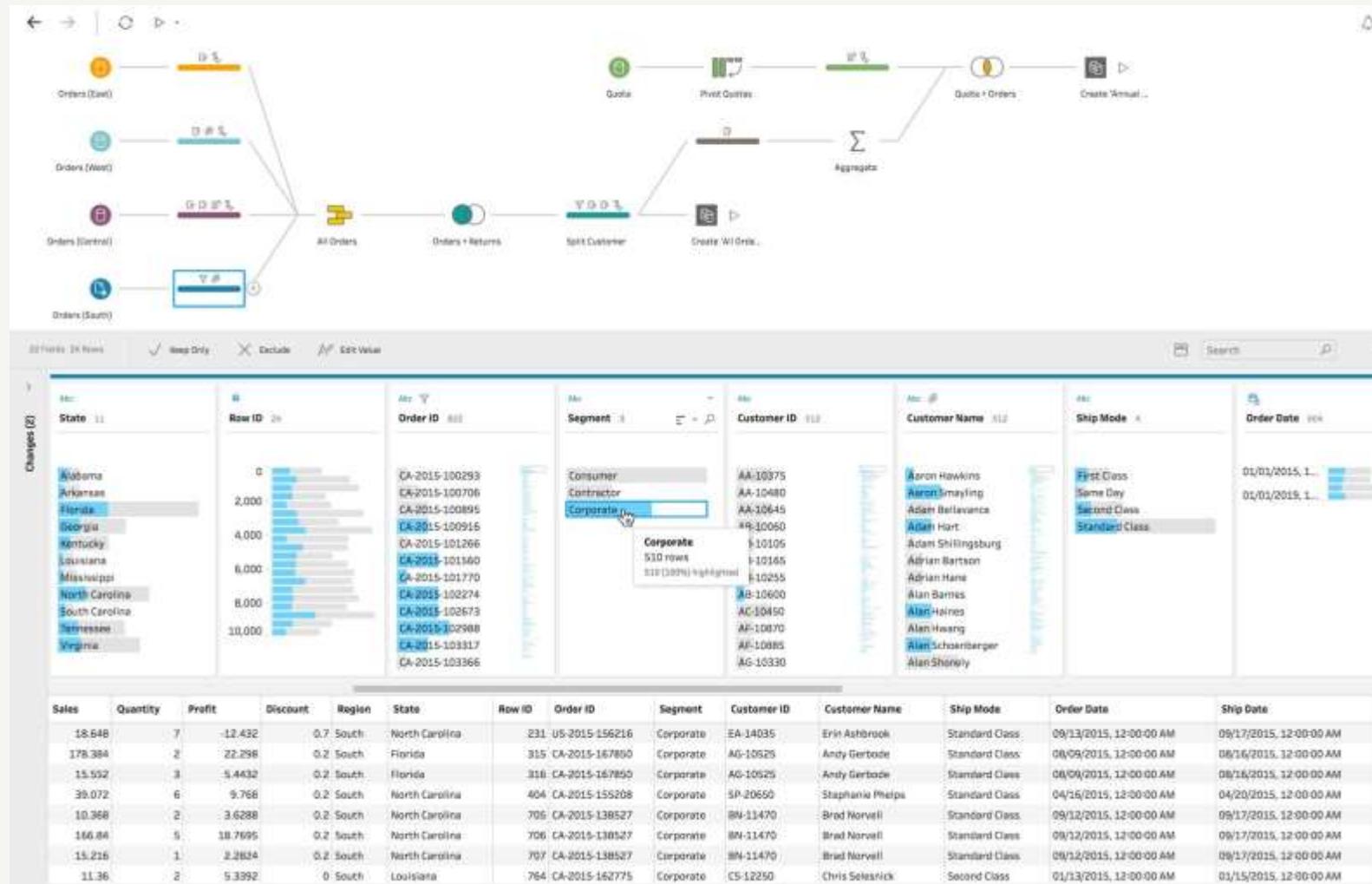
Data preparation in Tableau

As we all know that data Will Never be in our required form and every time we need to pre process that before doing any analysis .

Generally we need to stuck our head in excel for doing the same .

Tableau Provide a Better Functionality to Doing the same , we can use the Tableau Prep for Cleaning the data

Tableau Prep is designed to reduce the struggle of common yet complex tasks—such as joins, unions, pivots, and aggregations—with a drag-and-drop visual experience. No scripting required.



Joins

In general, there are four types of joins that you can use in Tableau: inner, left, right, and full outer.

Inner

:

When you use an inner join to combine tables, the result is a table that contains values that have matches in both tables.

When a value doesn't match across both tables, it is dropped entirely.

Left

:

When you use a left join to combine tables, the result is a table that contains all values from the left table and corresponding matches from the right table.

When a value in the left table doesn't have a corresponding match in the right table, you see a null value in the data grid.

Right

:

When you use a right join to combine tables, the result is a table that contains all values from the right table and corresponding matches from the left table.

When a value in the right table doesn't have a corresponding match in the left table, you see a null value in the data grid.

Full Outer

:

When you use a full outer join to combine tables, the result is a table that contains all values from both tables.

When a value from either table doesn't have a match with the other table, you see a null value in the data grid.

Union

Though union is not a type of join, union is another method for combining two or more tables by appending rows of data from one table to another. Ideally, the tables that you union have the same number of fields, and those fields have matching names and data types.

Filters

Filters are a smart way to collate and segregate data based on its dimensions and sets to reduce the overall data frequency for faster processing.

There are six different types of filters in tableau desktop based on their various objectives

1. Extract Filter

As understood by its name, the extract filters are used to extract data from the various sources,

Such methods can help in lowering the tableau queries to the data source.

2. Data Source Filter

Used mainly to restrict sensitive data from the data viewers, the data source filters are similar to the extract filters in minimizing the data feeds for faster processing.

The data source filter in tableau helps in the direct application of the filter environment to the source data and quickly uploads data that qualifies the scenario into the tableau workbook.

3. Context Filter

A context filter is a discrete filter on its own, creating datasets based on the original datasheet and the presets chosen for compiling the data. Since all the types of filters in tableau get applied to all rows in the datasheet, irrespective of any other filters, the context filter would ensure that it is first to get processed.

Despite being constrained to view all data rows, it can be implemented to choose sheets as and when required to optimize its performance by minimizing the data efficiently.

4. Dimension filter

Now that you've chosen the data, you can access the values highlighted or remove them from the selected dimension, represented as strikethrough values. You can click All or None to select or deselect based on your operation in case of multiple dimensions.

5. Measure Filter

In this filter, you can apply the various operations like Sum, Avg, Median, Standard Deviation, and other aggregate functions. In the next stage, you would be presented with four choices: Range, At least, At most, and Special for your values. Every time you drag the data you want to filter, you can do that in a specific setting.

6. Table Filter

The last filter to process is the table calculation that gets executed once the data view has been rendered. With this filter, you can quickly look into the data without any filtering of the hidden data.

Charts And Graphs

We can Create many charts in tableau depending upon our requirement

We can have

- Bar Chart**
- Line Chart**
- Pie Chart**
- Maps**
- Density Maps**
- Scatter Plot**
- Gantt Plot**
- Bubble Chart**
- Tree Map**

And many more



Calculated Field

Calculated fields allow you to create new data from data that already exists in your data source. When you create a calculated field, you are essentially creating a new field (or column) in your data source, the values or members of which are determined by a calculation that you control.

This new calculated field is saved to your data source in Tableau, and can be used to create more robust visualizations. But don't worry: your original data remains untouched.

You can use calculated fields for many, many reasons. Some examples might include:

- To segment data
- To convert the data type of a field, such as converting a string to a date.
- To aggregate data
- To filter results
- To calculate ratios

How to perform Calculation on Tableau .

1. In Tableau, select Analysis > Create Calculated Field.
2. In the Calculation Editor that opens, do the following:
 - Enter a name for the calculated field. In this example, the field is called, Discount Ratio.
 - Enter a formula. This example uses the following formula:

IF([sales] != 0 , [discount]/[sales],0)

This formula checks if sales is not equal to zero. If true, it returns the discount ratio (Discount/Sales); if false, it returns zero.

3. When finished, click **OK**.

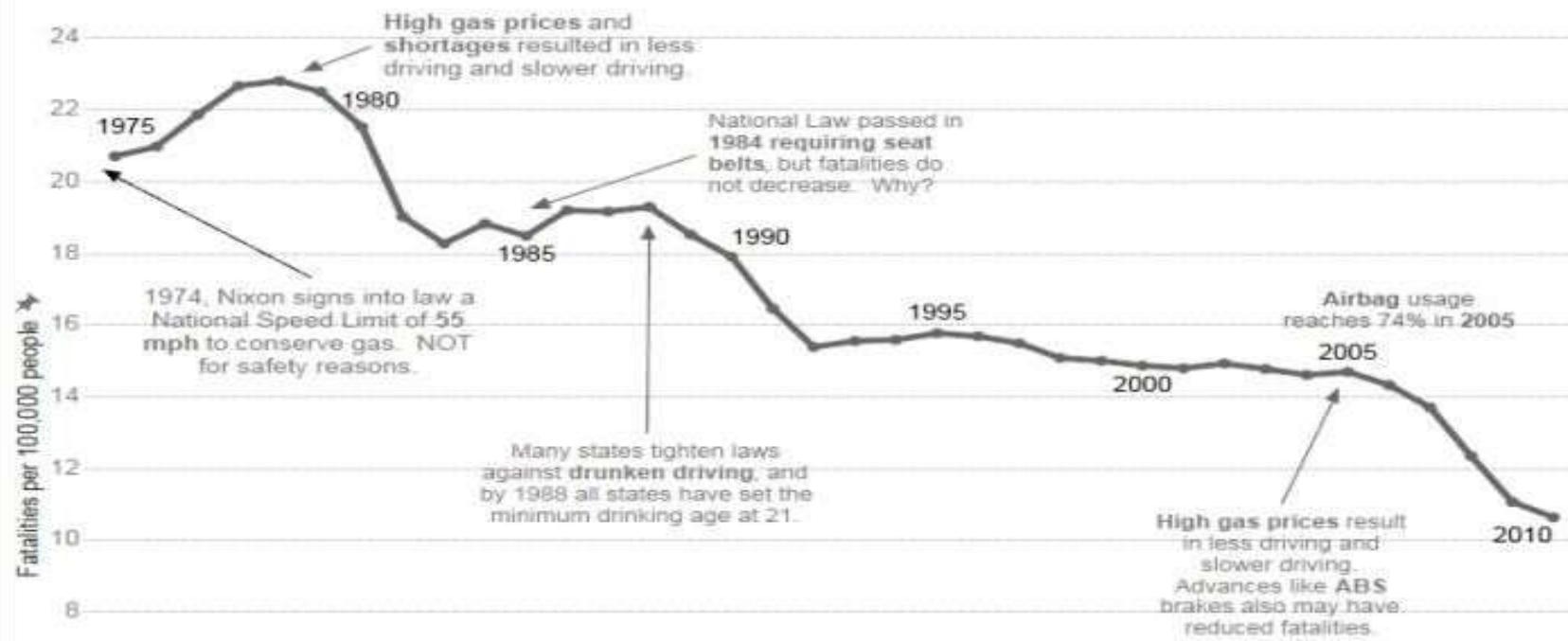
Story Making

Use stories to make your case more compelling by showing how facts are connected, and how decisions relate to outcomes. You can then publish your story to the web, or present it to an audience.

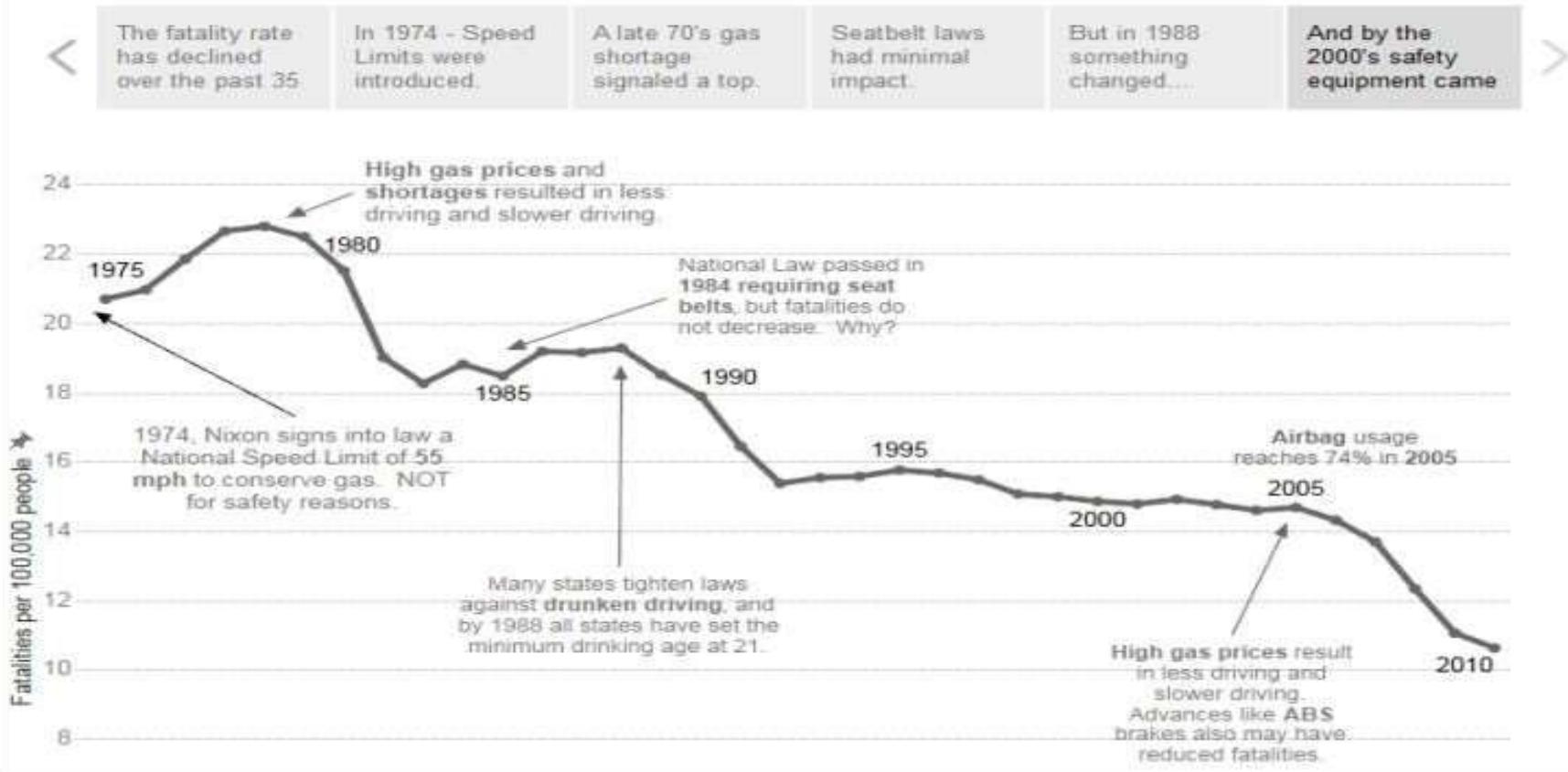
Each story point can be based on a different view or dashboard, or the entire story can be based on the same visualization seen at different stages, with different filters and annotations.

Why have driving fatalities decreased in the United States?

< The fatality rate has declined over the past 35 years. In 1974 - Speed Limits were introduced. A late 70's gas shortage signaled a top. Seatbelt laws had minimal impact. But in 1988 something changed... And by the 2000's safety equipment came >



Why have driving fatalities decreased in the United States?



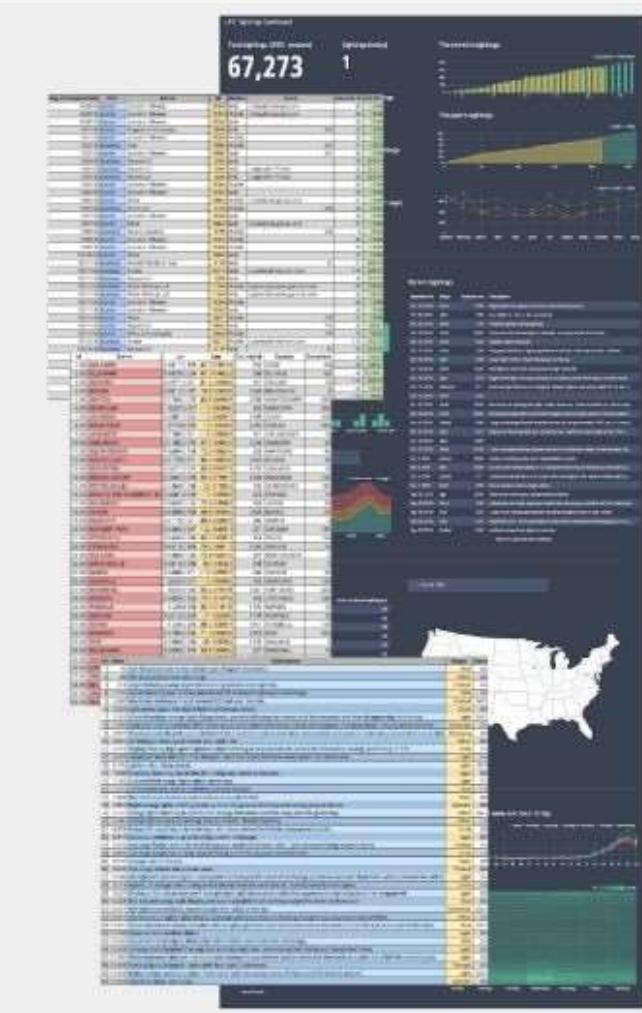
Dashboard And Reports

Reports VS Dashboard

A report is a more detailed collection of tables, charts, and graphs and it is used for a much more detailed, full analysis while a dashboard is used for monitoring what is going on. The behavior of the pieces that make up dashboards and reports are similar, but their makeup itself is different.

A dashboard answers a question in a single view and a report provides information.

The report can provide a more detailed view of the information that is presented on a dashboard.



VS.

