In [ ]:
```
#EDA - Exploratry data Analysis
#ETL - exteact,transpose, load - sql(ETL- informatica, ibm watson)
#covertfrom row - clean technique are called EDA
'''1.variabke identification
2.univariate analysis'
3.bivariate analysis
4.outlier treatement
5.missing value treatment
6.imputation or tranformer
7.variable creation'''
```

In [ ]:
```
#1.variabke identification
'''ex
family(loan,kid education,house propety)
   father(government employee) -- dependent variabe or target attribute or tar
   mother(home) independent variable - non target variable - non predictable v
   son(2nd) - independent variable - non target variable - non predictable var
   daughter(5th)independent variable - non target variable - non predictable v
   y = x1 + x2+x3
   y = m1x1 + m2x2 + m3x3
   family
      father - y
      mother - x1

      linear equation - y = mx + c(linear regression algorithm)
1. Variable identificaton
independent variable  = x1,x2,x3
dependent varibale = y'''

'''
mechine learning
1.regression - dependent variable are contineous
2.classificaton - dependent variable are binary or cetagorical
3. clustring -  no dependent variable but group

1. Regression model
contineous variable
ex - gold prie , petrol price, flight fare price, eletricity bill price , land
food price, vegetable price , etc

1. Simple linear regression
2. multiple linear regression
3. polynomial
4.gradient descent, stocastic, batch
4. k - nearest
5. svr
6.descision tree
7.xgboost
8.light gradiant boosting
9.time series
10.Ann
```

```
2.Classifiction - dependent variable is binary or category
algorithm
1.logistic regression
2.support vector machine
3.knn classifier
4.naive bayes(bayesian theorem)
5. decission tree
6. random forest
7.xgboost
8.lgbm
9.ann classifier

3. Clustering - grouping
1.pca(principle component analysis)
2.k-means
3.hierarchical
4.dbscan

ml build/ work wih relevent attribute does not work on irrelevent atrribute
every time being with ds we need to look for relevent attribute

if you build the ml model with irrelevent attribute then overfitting or multic

'''
```

In [ ]:

```
'''
2.Univarite Analysis(plot graph using one variable)
3.Bivariate Analysis(Plot graph using two variable)
     Corelation - relation among attributes in the dataset
     range of corelaton is - to 1
        3 parts:
          1.positive corelation - range is 0 to 1(y=mx+c)
          2.negative corelation - range is -1 to 0
          3.no corelation - range is 0(neither positive or nor negative)

4.Missing value treatment -
dataset build with numerical data or categorical data

if numerical data missing then we need to implement:
mean strategy
median strategy
mode strategy

if text data missing then we need to implement:
mode strategy
knn algorithm

5. Outliers treatment --> by plot the graph
6.Imputation or Tranformers
    Transformer: transform categorical data numerical dara to build ml
    transformer also called as imputation
    NLP--> embbeding
        1.Dummy variable --> in form of 0 and 1
```

```
        2.One hot encoder -->
        3.Label encoder --> count of particular type of values

7.Variable creation
from 1 variable will create more attributes

*** EDA also called as feature engineering technique ***

Does outelier impact mean,median or mode strategy ?
ans --> mean strategy impacted by outliers
'''
```

In [2]:
```
import pandas as pd
emp = pd.read_excel(r"/content/Rawdata.xlsx")
```

In [3]:
```
emp
```

Out[3]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy^	Testing	45' yr	Bangalore	10%%000	<3
2	Uma#r	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam*	Statistics	67-yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

In [4]:
```
emp.shape
```

Out[4]:
```
(6, 6)
```

In [5]:
```
id(emp)
```

Out[5]:
```
135480149860544
```

In [6]:
```
emp.columns
```

Out[6]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')
```

In [7]:
```
emp.head()
```

Out[7]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy^	Testing	45' yr	Bangalore	10%%000	<3
2	Uma#r	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam*	Statistics	67-yr	NaN	30000-	5+ year

In [8]:
```
emp.isnull()
```

Out[8]:

	Name	Domain	Age	Location	Salary	Exp
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	True	True	False	False
3	False	False	True	False	False	True
4	False	False	False	True	False	False
5	False	False	False	False	False	False

In [9]:
```
emp['Name']
```

Out[9]:

	Name
0	Mike
1	Teddy^
2	Uma#r
3	Jane
4	Uttam*
5	Kim

dtype: object

In [10]:
```
emp['Name'] = emp['Name'].str.replace(r'\W','', regex=True)
```

In [ ]:
```
# W is non word character
```

In [11]:
```
emp['Name']
```

Out[11]:

	Name
0	Mike
1	Teddy
2	Umar
3	Jane
4	Uttam
5	Kim

dtype: object

In [12]:
```
emp
```

Out[12]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy	Testing	45' yr	Bangalore	10%%000	<3
2	Umar	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam	Statistics	67-yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

In [ ]:
```
emp.isnull().sum()
```

Out[ ]:

	0
Name	0
Domain	0
Age	2
Location	2
Salary	0
Exp	1

dtype: int64

In [13]:
```
emp['Domain']
```

Out[13]:

	Domain
0	Datascience#$
1	Testing
2	Dataanalyst^^#
3	Ana^^lytics
4	Statistics
5	NLP

dtype: object

In [14]:
```
emp['Domain'] = emp['Domain'].str.replace(r'\W','', regex=True)
```

In [15]:
```
emp['Domain']
```

Out[15]:

	Domain
0	Datascience
1	Testing
2	Dataanalyst
3	Analytics
4	Statistics
5	NLP

dtype: object

In [16]:
```
emp
```

Out[16]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34 years	Mumbai	5^00#0	2+
1	Teddy	Testing	45' yr	Bangalore	10%%000	<3
2	Umar	Dataanalyst	NaN	NaN	1$5%000	4> yrs
3	Jane	Analytics	NaN	Hyderbad	2000^0	NaN
4	Uttam	Statistics	67-yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

In [17]:
```
emp['Age'] = emp['Age'].str.replace(r'\W','', regex=True)
```

In [18]:
```
emp['Age']
```

Out[18]:

	Age
0	34years
1	45yr
2	NaN
3	NaN
4	67yr
5	55yr

dtype: object

In [19]:
```
emp
```

Out[19]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34years	Mumbai	5^00#0	2+
1	Teddy	Testing	45yr	Bangalore	10%%000	<3
2	Umar	Dataanalyst	NaN	NaN	1$5%000	4> yrs
3	Jane	Analytics	NaN	Hyderbad	2000^0	NaN
4	Uttam	Statistics	67yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

In [20]:
```
emp['Age'] = emp['Age'].str.extract('(\\d+)')
```

In [21]:
```
print(emp["Age"])
```

```
0     34
1     45
2    NaN
3    NaN
4     67
5     55
Name: Age, dtype: object
```

In [22]:
```
emp
```

Out[22]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5^00#0	2+
1	Teddy	Testing	45	Bangalore	10%%000	<3
2	Umar	Dataanalyst	NaN	NaN	1$5%000	4> yrs
3	Jane	Analytics	NaN	Hyderbad	2000^0	NaN
4	Uttam	Statistics	67	NaN	30000-	5+ year
5	Kim	NLP	55	Delhi	6000^$0	10+

In [23]:
```
emp['Location'] = emp['Location'].str.replace(r'\W','', regex=True)
```

In [24]:
```
print(emp['Location'])
```

```
0       Mumbai
1    Bangalore
2          NaN
3     Hyderbad
4          NaN
5        Delhi
Name: Location, dtype: object
```

In [25]:
```
emp
```

Out[25]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5^00#0	2+
1	Teddy	Testing	45	Bangalore	10%%000	<3
2	Umar	Dataanalyst	NaN	NaN	1$5%000	4> yrs
3	Jane	Analytics	NaN	Hyderbad	2000^0	NaN
4	Uttam	Statistics	67	NaN	30000-	5+ year
5	Kim	NLP	55	Delhi	6000^$0	10+

In [26]:
```
emp['Salary'] = emp['Salary'].str.replace(r'\W','', regex=True)
```

In [27]:
```
print(emp["Salary"])
```

```
0     5000
1    10000
2    15000
3    20000
4    30000
5    60000
Name: Salary, dtype: object
```

In [28]:
```
emp
```

Out[28]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2+
1	Teddy	Testing	45	Bangalore	10000	<3
2	Umar	Dataanalyst	NaN	NaN	15000	4> yrs
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5+ year
5	Kim	NLP	55	Delhi	60000	10+

In [29]:
```
emp.head()
```

Out[29]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2+
1	Teddy	Testing	45	Bangalore	10000	<3
2	Umar	Dataanalyst	NaN	NaN	15000	4> yrs
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5+ year

In [30]:
```
emp['Exp'] = emp['Exp'].str.extract('(\\d+)')
```

In [31]:
```
print(emp['Exp'])
```

```
0      2
1      3
2      4
3    NaN
4      5
5     10
Name: Exp, dtype: object
```

In [32]:
```
emp
```

Out[32]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	NaN	NaN	15000	4
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [33]:
```
clean_data = emp.copy()
```

In [34]:
```
clean_data
```

Out[34]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	NaN	NaN	15000	4
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [35]:
```
clean_data.isnull().sum()
```

Out[35]:

	0
Name	0
Domain	0
Age	2
Location	2
Salary	0
Exp	1

dtype: int64

In [36]:
```
clean_data["Age"]
```

Out[36]:

	Age
0	34
1	45
2	NaN
3	NaN
4	67
5	55

dtype: object

In [37]:
```
import numpy as np
```

In [ ]:

In [38]:
```
clean_data['Age'] = clean_data['Age'].fillna(np.mean(pd.to_numeric(clean_data[
```

In [39]:
```
clean_data
```

Out[39]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50.25	NaN	15000	4
3	Jane	Analytics	50.25	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [40]:
```
clean_data['Exp'] = clean_data['Exp'].fillna(np.mean(pd.to_numeric(clean_data[
```

In [41]:
```
clean_data['Exp']
```

Out[41]:

	Exp
0	2
1	3
2	4
3	4.8
4	5
5	10

dtype: object

In [42]:
```
clean_data
```

Out[42]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50.25	NaN	15000	4
3	Jane	Analytics	50.25	Hyderbad	20000	4.8
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [43]:
```
clean_data['Location'] = clean_data['Location'].fillna(clean_data['Location'].
```

In [44]:
```
clean_data['Location']
```

Out[44]:

	Location
0	Mumbai
1	Bangalore
2	Bangalore
3	Hyderbad
4	Bangalore
5	Delhi

dtype: object

In [45]:
```
clean_data
```

Out[45]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50.25	Bangalore	15000	4
3	Jane	Analytics	50.25	Hyderbad	20000	4.8
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [46]:
```
clean_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Name      6 non-null      object
 1   Domain    6 non-null      object
 2   Age       6 non-null      object
 3   Location  6 non-null      object
 4   Salary    6 non-null      object
 5   Exp       6 non-null      object
dtypes: object(6)
memory usage: 420.0+ bytes
```

In [47]:
```
clean_data['Age'] = clean_data['Age'].astype(int)
```

In [48]:
```
clean_data
```

Out[48]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4.8
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [49]:
```
clean_data['Name'] = clean_data['Name'].astype('category')
clean_data['Domain'] = clean_data['Domain'].astype('category')
clean_data['Location'] = clean_data['Location'].astype('category')
```

In [50]:
```
clean_data
```

Out[50]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4.8
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [51]:
```
clean_data.to_csv('clean_data.csv')
```

In [52]:
```
import os
os.getcwd()# from the os give saved current working directly
```

Out[52]:
```
'/content'
```

In [53]:
```
clean_data
```

Out[53]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4.8
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [ ]:
```
#Python Dunction - it is a collection of statement
'''
Two type of functuon
1.Inbuild Function
2.User Define Function

Syantax of Function
def function_name():
  print()
'''
```

In [55]:
```
def hello():
  print("Good Evening")
hello()
hello()
hello()
```

```
Good Evening
Good Evening
Good Evening
```

In [61]:
```
def add(x,y):
  z = x + y
  return z
add(2,3)
```

Out[61]:
```
5
```

In [58]:
```
def Arithmetic(x,y):
  z = x + y
  print("Addition :",z)
  e = x -y
  print("Sub:", e)
```

```
Arithmetic(6,4)
```

```
Addition : 10
Sub: 2
```

In [59]:
```
def add(x,y,z):
  c = x+y+z
  print(c)
add(2,3,4)
```

```
9
```

In [60]:
```
def add(x,y,z):
  c = x+y+z
  return c
add(2,3,4)
```

Out[60]: 9

In [62]:
```
def hello():
  print("Good Evening")

def add(x,y):
  z = x + y
  return z


hello()
add(2,3)
```

```
Good Evening
```

Out[62]: 5

In [69]:
```
def hello():
  print("Hello !")
  print("Good Evening")


def add(,b):
  z = a + b
  print(z)

def sub(x,y,z):
  s = x - y - z
  print(s)


hello()
add(2,3)
sub(6,3,1)
```

```
Hello !
Good Evening
5
2
```

In [75]:
```
def add_sub(x,y):
  c = x + y
  d = x - y
  return c, d

result = add_sub(4,5)
print(result)
print(type(result))
```

```
(9, -1)
<class 'tuple'>
```

In [78]:
```
def add_sub(x,y):
  c = x + y
  d = x - y
  return c, d

result, result1 = add_sub(4,5)
print(result)
print(result1)
```

```
9
-1
```

In [80]:
```
def add_sub_mul(x,y):
  c = x + y
  d = x - y
  e =x * y
  return c, d, e

result, result1, result2 = add_sub_mul(4,5)
print(result)
print(result1)
print(result2)

print(type(result))
print(type(result1))
print(type(result2))
```

```
9
-1
20
<class 'int'>
<class 'int'>
<class 'int'>
```

In [ ]:
```
#function agrgument has two parts
#1. Formal argument - mention at the time of definition
#2. Actual argument -  while we are calling
```

In [81]:
```
def update():
  x =6
  print(x)
update()
```

```
6
```

In [82]:
```
def update():
  x =6
  print(x)
update()
```

```
6
```

In [83]:
```
def add(x,y): # x & y are called as formal argument
  z = x + y
  print(z)
add(2,3) #2,3 are actual argument
```

```
5
```

In [ ]:
```
#actual argument has 4 parts
'''1. positional arguments
2. keyword arguments
3. default arguments
4. Variable length arguments'''
```

In [85]:
```
#Positional Argument
def person(name,age):
  print(name)
  print(age)

person('nit', 22)
```

```
nit
22
```

In [86]:
```
def person(name,age):
  print(name)
  print(age)

person(22, 'nit')
```

```
22
nit
```

In [88]:
```
def person(name,age):
  print(name)
  print(age - 1)

person('nit', 22)
```

```
nit
21
```

In [90]:
```
def person(name,age):
```

```
  print(name)
  print(age + 1)

person('nit', 22)
```

```
nit
23
```

In [91]:
```
def person(name,age):
  print(name)
  print(age + 1)

person(22)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-682854463.py in <cell line: 0>()
      3   print(age + 1)
      4
----> 5 person(22)

TypeError: person() missing 1 required positional argument: 'age'
```

In [92]:
```
def person(name):
  print(name)
  print(age + 1)

person('nit', 22)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-3821191632.py in <cell line: 0>()
      3   print(age + 1)
      4
----> 5 person('nit', 22)

TypeError: person() takes 1 positional argument but 2 were given
```

In [95]:
```
#keyWord argument
def person(name,age,phone):
  print(name)
  print(age + 1)
  print(phone)

person(age = 22, name = 'nit', phone = 2345)
```

```
nit
23
2345
```

In [99]:
```
#Default argument
def person(name,age = 18):
  print(name)
  print(age)
```

```
person('nit')
```

```
nit
19
```

In [ ]:
```
#variable Length argument
```