PERFORMING EXPLORATORY DATA ANALYSIS ON BANKING DATA

STEP.01= IMPORTING LIBRARY(PANDAS)

In [4]:
```
# Importing Library

import pandas as pd,os
```

STEP.02 = LOADING DATASET INTO NOTEBOOK

In [5]:
```
#HERE df is dataframe in PANDAS

bank_df = pd.read_excel('C:\\Users\\abc\\Documents\\MARCH 2022 - NEW\\INTERNSHIP @ INEURON\\BANKING PROJECT\\bankfull1.xlsx')
```

STEP.03= CHECKING THE FIRST 5 AND LAST 5 ROWS OF THE DATASET, using head() & tail()

In [6]:
```
#head
bank_df.head()
```

Out[6]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

In [17]:
```
#tail
bank_df.tail()
```

Out[17]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	0	unknown	yes
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	0	unknown	yes
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	3	success	yes
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	0	unknown	no
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	11	other	no

STEP.04= CHECKING THE DATAYPE & OTHER INFO OF THE DATASET

In [7]:
```
bank_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

STEP.05= CHECKING THE NULL VALUES OF THE DATASET

In [8]:
```
bank_df.isnull().sum() #this will return the count of null from each columns.
```

Out[8]:
```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

```
Since in the given dataset it is mentioned that the null values are present as 'unknown' instead of 'nan/NaN/NuLL',
thus it is not showing the result.
```

STEP.06= LET'S RENAME THE VARIOUS COLUMNS FOR BETTER UNDERSTANDING

In [9]:
```python
#using .rename(columns = {'col1':'new_name'},inplace = true)

bank_df.rename(columns ={'age':'Age_Group','job':'Job_Types','housing':'Housing_Loan','loan':'Personal_Loan'}, inplace = True)
bank_df.rename(columns = {'duration':'Last_Call_Dur','campaign':'Current_FollowUps','pdays':'Contact_Day_Diff'}, inplace = True)
bank_df.rename(columns = {'previous':'Previous_FollowUps', 'poutcome':'Previous_Camp_Status', 'y':'Current_Camp_Status'},
                           inplace =True)
```

In [10]:
```python
bank_df.head()
```

Out[10]:

	Age_Group	Job_Types	marital	education	default	balance	Housing_Loan	Personal_Loan	contact	day	month	Last_Call_Dur	Current_FollowUps	Contact_Day_Diff	Previous_FollowUps	Previous_Camp_Status
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown

STEP.07= CATERGORIZING THE NUMERIC COLUMNS IN ORDER TO FIND KEY RELATIONSHIPS GOING AHEAD

```
Now this is a very lengthy step, since it requires all the operations to be performed column by columns.
```

STEP.07.1= CREATING FUNCTION FOR (AGE GROUP)

x(18-30):Young Adults, x(31-45):Min Age Adults, x(46-60): Veterans, x(>60): Senior Citizen

In [11]:
```python
#creating a function.

def age_group(x):
    if x >=18 and x<=30:
        return 'Young Adults'
    elif x>30 and x<=45:
        return 'Mid Age Adults'
    elif x>45 and x<=60:
        return 'Veterans'
    else:
        return 'Senior Citizen'
```

In [12]:
```python
#applying function to the column

bank_df['Age_Group'] = bank_df['Age_Group'].apply(age_group)
```

In [13]:
```python
#checking value counts after categorizing

bank_df['Age_Group'].value_counts()
```

Out[13]:
```
Mid Age Adults    23733
Veterans          13260
Young Adults       7030
Senior Citizen     1188
Name: Age_Group, dtype: int64
```

STEP.07.2= CHECKING THE VALUE COUNTS IN (JOB TYPES) & TREATING THE MISSING VALUES WITH MODE

In [14]:
```
bank_df.Job_Types.value_counts() #Or use { bank_df['Job_Types'].value_counts() }
```

Out[14]:
```
blue-collar      9732
management       9458
technician       7597
admin.           5171
services         4154
retired          2264
self-employed    1579
entrepreneur     1487
unemployed       1303
housemaid        1240
student           938
unknown           288
Name: Job_Types, dtype: int64
```

Above we can see there are 288 unknown null values available in the Job category and it may affect the outcome, so we need to treat them with Mode of the column, since the records are categorical in nature.

In [15]:
```
#Finding the mode, so that we can replace it with unknown entires.

bank_df.Job_Types.mode() #or write bank_df['Job_Types'].mode()
```

Out[15]:
```
0    blue-collar
dtype: object
```

In [16]:
```
#Replacing null value with mode column by creating the function

def unknown2bluecollar(x):
    if x == 'unknown':
        return 'blue-collar'
    else:
        return x
```

In [17]:
```
#Applying function on Job_Type

bank_df['Job_Types'] = bank_df['Job_Types'].apply(unknown2bluecollar)
```

In [18]:
```
#Checking after replacing the unknowns with mode

bank_df['Job_Types'].value_counts()
```

Out[18]:
```
blue-collar      10020
management        9458
technician        7597
admin.            5171
services          4154
retired           2264
self-employed     1579
entrepreneur      1487
unemployed        1303
housemaid         1240
student            938
Name: Job_Types, dtype: int64
```

In [19]:
```
bank_df['Job_Types'].unique()
```

Out[19]:
```
array(['management', 'technician', 'entrepreneur', 'blue-collar',
       'retired', 'admin.', 'services', 'self-employed', 'unemployed',
       'housemaid', 'student'], dtype=object)
```

STEP.07.2.1= Grouping the Job_Types into White Collar Job/ Blue Collar Job/ Entrepreneur

```
•Creating function for job group
                    •considered desk job as white collar job
                    •considerd field job as blue collar job
                    •considerd self-employed as Entrepreneur
```

In [20]:
```
def job_group(x):
    if x == 'admin.' or x == 'management' or x == 'services':
        return 'White Collar'
    elif x == 'blue-collar' or x  == 'housemaid' or x == 'technician':
        return 'Blue Collar'
    elif x == 'entrepreneur' or x == 'self-employed':
        return 'Entrepreneur'
    else:
        return x
```

In [21]:
```
#Applying the grouping function

bank_df['Job_Types'] = bank_df['Job_Types'].apply(job_group)
```

In [22]:
```
#checking the result after applying the function.

bank_df.Job_Types.value_counts()
```

Out[22]:
```
Blue Collar     18857
White Collar    18783
Entrepreneur     3066
retired          2264
unemployed       1303
student           938
Name: Job_Types, dtype: int64
```

STEP.07.3= CHECKING AND TREATING THE MISSING VALUES FROM EDUCATION COLUMN

In [23]:
```
#checking  null values from column

bank_df['education'].value_counts()
```

Out[23]:
```
secondary    23202
tertiary     13301
primary       6851
unknown       1857
Name: education, dtype: int64
```

The above result shows there are 928 unknown entries, to treat them we need to replace them with MODE of the column

In [24]:
```
#finding the mode of education

bank_df.education.mode()
```

Out[24]:
```
0    secondary
dtype: object
```

In [25]:
```
#Replacing null value with mode column by creating the function

def replace_edu(x):
    if x == 'unknown':
        return 'secondary'
    else:
        return x
```

In [26]:
```
#Applying the function to replace the unknown.

bank_df.education = bank_df.education.apply(replace_edu)
```

In [27]:
```
#checking the result after removing the null values.

bank_df['education'].value_counts()
```

Out[27]:
```
secondary    25059
tertiary     13301
primary       6851
Name: education, dtype: int64
```

STEP.07.4= CHECKING, GROUPING & TREATING THE MISSING VALUES FROM THE BALANCE COLUMN

```
Since all the entries in the Balance are nurmeric thus we will first define a function for the grouping and then apply the grouping to the balance column
```

•considered value **< 0** as **negative balance**

•considerd value **> 0** and **<= 500** as **low balance**

•considerd value **> 500** and **<= 4000** as **average balance**

•considerd value **> 4000** as **high balance**

In [28]:
```
#Creating grouping function for balance

def group_bal(y):
    if y <= 0:
        return 'Negative balance'
    elif (y > 0 and y <= 500):
        return 'Low Balance'
    elif (y > 500 and y <= 4000):
        return 'Average Balance'
    else:
        return 'High Balance'
```

In [29]:
```
#Applying the grouping function to the education column.

bank_df.balance=bank_df.balance.apply(group_bal)
```

In [30]:
```
bank_df.balance.value_counts()
```

Out[30]:
```
Average Balance     17648
Low Balance         16385
Negative balance     7280
High Balance         3898
Name: balance, dtype: int64
```

STEP.07.5= CONVERTING THE VALUES IN THE Last_Call_Duration COLUMN FROM SECONDS TO MINTUES & ROUNDING TO 0.

In [31]:
```
#dividing the column with 60 to get values in minutes and using .round(0) function.

bank_df.Last_Call_Dur = (bank_df.Last_Call_Dur / 60).round(0)
```

In [32]:
```
#checking the changes in Last_Call_Dur column

bank_df
```

Out[32]:

	Age_Group	Job_Types	marital	education	default	balance	Housing_Loan	Personal_Loan	contact	day	month	Last_Call_Dur	Current_FollowUps	Contact_Day_Diff	Previous_FollowUps	Previous_Camp_St
0	Veterans	White Collar	married	tertiary	no	Average Balance	yes	no	unknown	5	may	4.0	1	-1	0	unkı
1	Mid Age Adults	Blue Collar	single	secondary	no	Low Balance	yes	no	unknown	5	may	3.0	1	-1	0	unkı
2	Mid Age Adults	Entrepreneur	married	secondary	no	Low Balance	yes	yes	unknown	5	may	1.0	1	-1	0	unkı
3	Veterans	Blue Collar	married	secondary	no	Average Balance	yes	no	unknown	5	may	2.0	1	-1	0	unkı
4	Mid Age Adults	Blue Collar	single	secondary	no	Low Balance	no	no	unknown	5	may	3.0	1	-1	0	unkı
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	Veterans	Blue Collar	married	tertiary	no	Average Balance	no	no	cellular	17	nov	16.0	3	-1	0	unkı
45207	Senior Citizen	retired	divorced	primary	no	Average Balance	no	no	cellular	17	nov	8.0	2	-1	0	unkı
45208	Senior Citizen	retired	married	secondary	no	High Balance	no	no	cellular	17	nov	19.0	5	184	3	suc
45209	Veterans	Blue Collar	married	secondary	no	Average Balance	no	no	telephone	17	nov	8.0	4	-1	0	unkı
45210	Mid Age Adults	Entrepreneur	married	secondary	no	Average Balance	no	no	cellular	17	nov	6.0	2	188	11	

45211 rows × 17 columns

STEP.07.5.1= GROUPING THE VALUES IN THE Last_Call_Duration COLUMN

•considerd duration **>= 0** and **<= 2** as **short call time**

•considerd duration **> 2** and **<= 5** as **medium call time**

•considerd duration **> 5** as **high call time**

In [33]:
```
#Function for grouping the Last_Call_Dur

def group_LCD(z):
    if (z>=0 and z<=2):
        return 'short call time'
    elif (z>2 and z<=5):
        return 'medium call time'
    else:
        return 'high call time'
```

In [34]:
```
#applying the grouping to the column

bank_df.Last_Call_Dur = bank_df.Last_Call_Dur.apply(group_LCD)
```

In [81]:
```
#Checking the update on the column

bank_df
```

Out[81]:

	Age_Group	Job_Types	marital	education	default	balance	Housing_Loan	Personal_Loan	day	month	Last_Call_Dur	Current_FollowUps	Contact_Day_Diff	Previous_FollowUps	Previous_Camp_Status	Curr
0	Veterans	White Collar	married	tertiary	no	Average Balance	yes	no	5	may	medium call time	1	-1	0	unknown	
1	Mid Age Adults	Blue Collar	single	secondary	no	Low Balance	yes	no	5	may	medium call time	1	-1	0	unknown	
2	Mid Age Adults	Entrepreneur	married	secondary	no	Low Balance	yes	yes	5	may	short call time	1	-1	0	unknown	
3	Veterans	Blue Collar	married	secondary	no	Average Balance	yes	no	5	may	short call time	1	-1	0	unknown	
4	Mid Age Adults	Blue Collar	single	secondary	no	Low Balance	no	no	5	may	medium call time	1	-1	0	unknown	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	Veterans	Blue Collar	married	tertiary	no	Average Balance	no	no	17	nov	high call time	3	-1	0	unknown	
45207	Senior Citizen	retired	divorced	primary	no	Average Balance	no	no	17	nov	high call time	2	-1	0	unknown	
45208	Senior Citizen	retired	married	secondary	no	High Balance	no	no	17	nov	high call time	5	184	3	success	
45209	Veterans	Blue Collar	married	secondary	no	Average Balance	no	no	17	nov	high call time	4	-1	0	unknown	
45210	Mid Age Adults	Entrepreneur	married	secondary	no	Average Balance	no	no	17	nov	high call time	2	188	11	other	

45211 rows × 16 columns

In [35]:
```
bank_df.Last_Call_Dur.value_counts()
```

Out[35]:
```
short call time     18610
medium call time    15862
high call time      10739
Name: Last_Call_Dur, dtype: int64
```

STEP.07.6= DROPPING THE CONTACT COLUMN, SINCE IT IS NOT HELPING TO ANALYSE ANYTHING.

In [36]:
```
#use .drop(['col_name'], axis=1, inplace =true)

bank_df.drop(['contact'],axis=1 ,inplace = True)
```

In [37]:
```
bank_df
```
Out[37]:

	Age_Group	Job_Types	marital	education	default	balance	Housing_Loan	Personal_Loan	day	month	Last_Call_Dur	Current_FollowUps	Contact_Day_Diff	Previous_FollowUps	Previous_Camp_Status	Curr
0	Veterans	White Collar	married	tertiary	no	Average Balance	yes	no	5	may	medium call time	1	-1	0	unknown	
1	Mid Age Adults	Blue Collar	single	secondary	no	Low Balance	yes	no	5	may	medium call time	1	-1	0	unknown	
2	Mid Age Adults	Entrepreneur	married	secondary	no	Low Balance	yes	yes	5	may	short call time	1	-1	0	unknown	
3	Veterans	Blue Collar	married	secondary	no	Average Balance	yes	no	5	may	short call time	1	-1	0	unknown	
4	Mid Age Adults	Blue Collar	single	secondary	no	Low Balance	no	no	5	may	medium call time	1	-1	0	unknown	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	Veterans	Blue Collar	married	tertiary	no	Average Balance	no	no	17	nov	high call time	3	-1	0	unknown	
45207	Senior Citizen	retired	divorced	primary	no	Average Balance	no	no	17	nov	high call time	2	-1	0	unknown	
45208	Senior Citizen	retired	married	secondary	no	High Balance	no	no	17	nov	high call time	5	184	3	success	
45209	Veterans	Blue Collar	married	secondary	no	Average Balance	no	no	17	nov	high call time	4	-1	0	unknown	
45210	Mid Age Adults	Entrepreneur	married	secondary	no	Average Balance	no	no	17	nov	high call time	2	188	11	other	

45211 rows × 16 columns

STEP.07.7= GROUPING THE Current_FollowUps Column

•cosidered value **>0** and **<=5** as **upto 5 followups**

•considered value **>5** as **more than 5 followups**

In [38]:
```
#Fuction for grouping the Current_FollowUps

def group_followup(x):
    if (x>0 and x<=5):
        return 'Upto 5 followups'
    else:
        return 'More than 5 followups'
```

In [39]:
```
#Applying the function to the respective column

bank_df.Current_FollowUps = bank_df.Current_FollowUps.apply(group_followup)
```

In [40]:
```
bank_df.Current_FollowUps.value_counts()
```
Out[40]:
```
Upto 5 followups         40856
More than 5 followups     4355
Name: Current_FollowUps, dtype: int64
```

In [41]:
```
bank_df
```
Out[41]:

	Age_Group	Job_Types	marital	education	default	balance	Housing_Loan	Personal_Loan	day	month	Last_Call_Dur	Current_FollowUps	Contact_Day_Diff	Previous_FollowUps	Previous_Camp_Status	Curr
0	Veterans	White Collar	married	tertiary	no	Average Balance	yes	no	5	may	medium call time	Upto 5 followups	-1	0	unknown	
1	Mid Age Adults	Blue Collar	single	secondary	no	Low Balance	yes	no	5	may	medium call time	Upto 5 followups	-1	0	unknown	
2	Mid Age Adults	Entrepreneur	married	secondary	no	Low Balance	yes	yes	5	may	short call time	Upto 5 followups	-1	0	unknown	
3	Veterans	Blue Collar	married	secondary	no	Average Balance	yes	no	5	may	short call time	Upto 5 followups	-1	0	unknown	
4	Mid Age Adults	Blue Collar	single	secondary	no	Low Balance	no	no	5	may	medium call time	Upto 5 followups	-1	0	unknown	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	Veterans	Blue Collar	married	tertiary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	-1	0	unknown	
45207	Senior Citizen	retired	divorced	primary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	-1	0	unknown	
45208	Senior Citizen	retired	married	secondary	no	High Balance	no	no	17	nov	high call time	Upto 5 followups	184	3	success	
45209	Veterans	Blue Collar	married	secondary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	-1	0	unknown	
45210	Mid Age Adults	Entrepreneur	married	secondary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	188	11	other	

45211 rows × 16 columns

STEP.07.8= GROUPING THE Contact_Day_Diff Column

•consider values **=-1** as **Not Contacted**

•consider values **>=0** and **<=90** as **0-3 Months Back**

•consider values **>90** and **<=180** as **3-6 Months Back**

•consider values **>180** as **More Than 6 Months**

In [42]:
```
# creating function for Contact_Day_Diff column group

def group_CDF(x):
    if x == -1:
        return 'Not Contacted'
    elif x >= 0 and x <= 90:
        return '0-3 Months Back'
    elif x > 90 and x <= 180:
        return '3-6 Months Back'
    else:
        return 'More Than 6 Months'
```
In [43]:
```
#Applying the function to the column

bank_df.Contact_Day_Diff = bank_df.Contact_Day_Diff.apply(group_CDF)
```

In [44]:
```
bank_df.Contact_Day_Diff.value_counts()
```
Out[44]:
```
Not Contacted         36954
More Than 6 Months     5059
3-6 Months Back        2480
0-3 Months Back         718
Name: Contact_Day_Diff, dtype: int64
```

In [91]:
```
bank_df
```
Out[91]:

up	Job_Types	marital	education	default	balance	Housing_Loan	Personal_Loan	day	month	Last_Call_Dur	Current_FollowUps	Contact_Day_Diff	Previous_FollowUps	Previous_Camp_Status	Current_Camp_Status
ans	White Collar	married	tertiary	no	Average Balance	yes	no	5	may	medium call time	Upto 5 followups	Not Contacted	0	unknown	no
ge lts	Blue Collar	single	secondary	no	Low Balance	yes	no	5	may	medium call time	Upto 5 followups	Not Contacted	0	unknown	no
ge lts	Entrepreneur	married	secondary	no	Low Balance	yes	yes	5	may	short call time	Upto 5 followups	Not Contacted	0	unknown	no
ans	Blue Collar	married	secondary	no	Average Balance	yes	no	5	may	short call time	Upto 5 followups	Not Contacted	0	unknown	no
ge lts	Blue Collar	single	secondary	no	Low Balance	no	no	5	may	medium call time	Upto 5 followups	Not Contacted	0	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
ans	Blue Collar	married	tertiary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	0	unknown	yes
ior zen	retired	divorced	primary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	0	unknown	yes
ior zen	retired	married	secondary	no	High Balance	no	no	17	nov	high call time	Upto 5 followups	More Than 6 Months	3	success	yes
ans	Blue Collar	married	secondary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	0	unknown	no
ge lts	Entrepreneur	married	secondary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	More Than 6 Months	11	other	no

columns

STEP.07.9= GROUPING THE Previous_FollowUps Column

•cosidered value **>=0** and **<=5** as **upto 5 followups**

•considered value **>5** as **more than 5 followups**

In [48]:
```
#Function for grouping the previous followups column

def previous_followup(x):
    if x >= 0 and x <=5:
        return 'Upto 5 followups'
    else:
        return 'More Than 5 followups'
```

In [49]:
```
bank_df.Previous_FollowUps = bank_df.Previous_FollowUps.apply(previous_followup)
```

In [50]:
```
bank_df.Previous_FollowUps.value_counts()
```
Out[50]:
```
Upto 5 followups         44147
More Than 5 followups     1064
Name: Previous_FollowUps, dtype: int64
```

In [51]:
```
bank_df.Previous_FollowUps.unique()
```
Out[51]:
```
array(['Upto 5 followups', 'More Than 5 followups'], dtype=object)
```

In [52]:
```
bank_df
```
Out[52]:

up	Job_Types	marital	education	default	balance	Housing_Loan	Personal_Loan	day	month	Last_Call_Dur	Current_FollowUps	Contact_Day_Diff	Previous_FollowUps	Previous_Camp_Status	Current_Camp_Status
ans	White Collar	married	tertiary	no	Average Balance	yes	no	5	may	medium call time	Upto 5 followups	Not Contacted	Upto 5 followups	unknown	no
ge ults	Blue Collar	single	secondary	no	Low Balance	yes	no	5	may	medium call time	Upto 5 followups	Not Contacted	Upto 5 followups	unknown	no
ge ults	Entrepreneur	married	secondary	no	Low Balance	yes	yes	5	may	short call time	Upto 5 followups	Not Contacted	Upto 5 followups	unknown	no
ans	Blue Collar	married	secondary	no	Average Balance	yes	no	5	may	short call time	Upto 5 followups	Not Contacted	Upto 5 followups	unknown	no
ge ults	Blue Collar	single	secondary	no	Low Balance	no	no	5	may	medium call time	Upto 5 followups	Not Contacted	Upto 5 followups	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
ans	Blue Collar	married	tertiary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	Upto 5 followups	unknown	yes
ior zen	retired	divorced	primary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	Upto 5 followups	unknown	yes
ior zen	retired	married	secondary	no	High Balance	no	no	17	nov	high call time	Upto 5 followups	More Than 6 Months	Upto 5 followups	success	yes
ans	Blue Collar	married	secondary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	Upto 5 followups	unknown	no
ge ults	Entrepreneur	married	secondary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	More Than 6 Months	More Than 5 followups	other	no

columns

STEP.07.10= OPERATING THE Previous_camp_Status COLUMN

•We need to the check the view counts and uniqueness of the entries

•Replacing null values (i.e 'unknown') with not contacted and (others) with failure

In [55]:
```
bank_df.Previous_Camp_Status.unique()
```
Out[55]:
```
array(['unknown', 'failure', 'other', 'success'], dtype=object)
```

In [56]:
```
bank_df.Previous_Camp_Status.value_counts()
```
Out[56]:
```
unknown    36959
failure     4901
other       1840
success     1511
Name: Previous_Camp_Status, dtype: int64
```

```
The above result shows we have 36k 'unknown' records, we will consider them as Not Contacted

& 1.8k 'other' does not defines anything, thus we will consider it as a failure.
```

In [57]:
```
#replacing the unwanted records: unknown=Not Contacted, other=failure

bank_df.Previous_Camp_Status = bank_df.Previous_Camp_Status.replace('unknown','not contacted').replace('other','failure')
```

In [59]:
```
#Checking the changes after replacement

bank_df.Previous_Camp_Status.value_counts()
```

Out[59]:
```
not contacted    36959
failure           6741
success           1511
Name: Previous_Camp_Status, dtype: int64
```

In [60]:
```
#Suppose replacing the categories once again to assign meaningfull name

bank_df.Previous_Camp_Status=bank_df.Previous_Camp_Status.replace('success','P Subscribed').replace('failure','P N Subscribed')
```

In [61]:
```
#Checking the changes after replacement 2.o


bank_df.Previous_Camp_Status.value_counts()
```

Out[61]:
```
not contacted     36959
P N Subscribed     6741
P Subscribed       1511
Name: Previous_Camp_Status, dtype: int64
```

In [62]:
```
bank_df
```

Out[62]:

up	Job_Types	marital	education	default	balance	Housing_Loan	Personal_Loan	day	month	Last_Call_Dur	Current_FollowUps	Contact_Day_Diff	Previous_FollowUps	Previous_Camp_Status	Current_Camp_Status
ans	White Collar	married	tertiary	no	Average Balance	yes	no	5	may	medium call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	no
ge ults	Blue Collar	single	secondary	no	Low Balance	yes	no	5	may	medium call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	no
ge ults	Entrepreneur	married	secondary	no	Low Balance	yes	yes	5	may	short call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	no
ans	Blue Collar	married	secondary	no	Average Balance	yes	no	5	may	short call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	no
ge ults	Blue Collar	single	secondary	no	Low Balance	no	no	5	may	medium call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
ans	Blue Collar	married	tertiary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	yes
ior zen	retired	divorced	primary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	yes
ior zen	retired	married	secondary	no	High Balance	no	no	17	nov	high call time	Upto 5 followups	More Than 6 Months	Upto 5 followups	P Subscribed	yes
ans	Blue Collar	married	secondary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	no
ge ults	Entrepreneur	married	secondary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	More Than 6 Months	More Than 5 followups	P N Subscribed	no

columns

STEP.07.11= OPERATING THE Current_Camp_Status COLUMN

•**Replacing yes = Subscribed & no = Not Subscribed**

In [63]:
```
bank_df.Current_Camp_Status = bank_df.Current_Camp_Status.replace('yes','Subscribed').replace('no','Not Subscribed ')
```

In [64]:
```
bank_df.Current_Camp_Status.value_counts()
```

Out[64]:
```
Not Subscribed     39922
Subscribed          5289
Name: Current_Camp_Status, dtype: int64
```

In [65]:
```
bank_df
```

Out[65]:

up	Job_Types	marital	education	default	balance	Housing_Loan	Personal_Loan	day	month	Last_Call_Dur	Current_FollowUps	Contact_Day_Diff	Previous_FollowUps	Previous_Camp_Status	Current_Camp_Status
ans	White Collar	married	tertiary	no	Average Balance	yes	no	5	may	medium call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	Not Subscribed
ge ılts	Blue Collar	single	secondary	no	Low Balance	yes	no	5	may	medium call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	Not Subscribed
ge ılts	Entrepreneur	married	secondary	no	Low Balance	yes	yes	5	may	short call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	Not Subscribed
ans	Blue Collar	married	secondary	no	Average Balance	yes	no	5	may	short call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	Not Subscribed
ge ılts	Blue Collar	single	secondary	no	Low Balance	no	no	5	may	medium call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	Not Subscribed
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
ans	Blue Collar	married	tertiary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	Subscribed
ior zen	retired	divorced	primary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	Subscribed
ior zen	retired	married	secondary	no	High Balance	no	no	17	nov	high call time	Upto 5 followups	More Than 6 Months	Upto 5 followups	P Subscribed	Subscribed
ans	Blue Collar	married	secondary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	Not Contacted	Upto 5 followups	not contacted	Not Subscribed
ge ılts	Entrepreneur	married	secondary	no	Average Balance	no	no	17	nov	high call time	Upto 5 followups	More Than 6 Months	More Than 5 followups	P N Subscribed	Not Subscribed

; columns

```
TILL HERE, WE HAVE PERFORMED THE EXPLORATORY DATA ANALYSIS AND WE HAVE GOT THE FINAL CLEANED TABLE WHICH HAS BEEN CATEGORIZED.
```

STEP.08 = SAVE THE FILE

Now it's time to save the file, so that we can perform data visualization after hooking up to POWER BI

In [69]:
```
bank_df.to_csv('Final_Banking_file.csv')
```

In [ ]: