IBM NALAIYA THIRAN

Data Visualization and Pre-processing

Assignment -2

Team ID	PNT2022TMID33620
Project Name	AI based discourse for Banking Industry
Student Name	Sukesh S
Student Roll Number	922519106163
Maximum Marks	2 Marks

1.Download the dataset: Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2.Load the dataset:

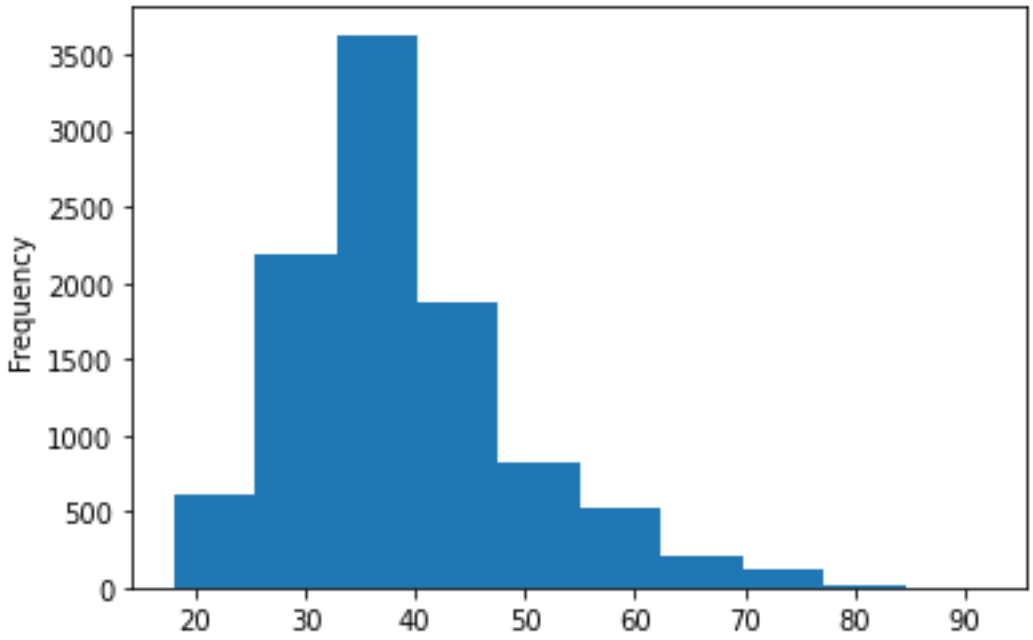
```
df=pd.read_csv('/content/Churn_Modelling.csv')

df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

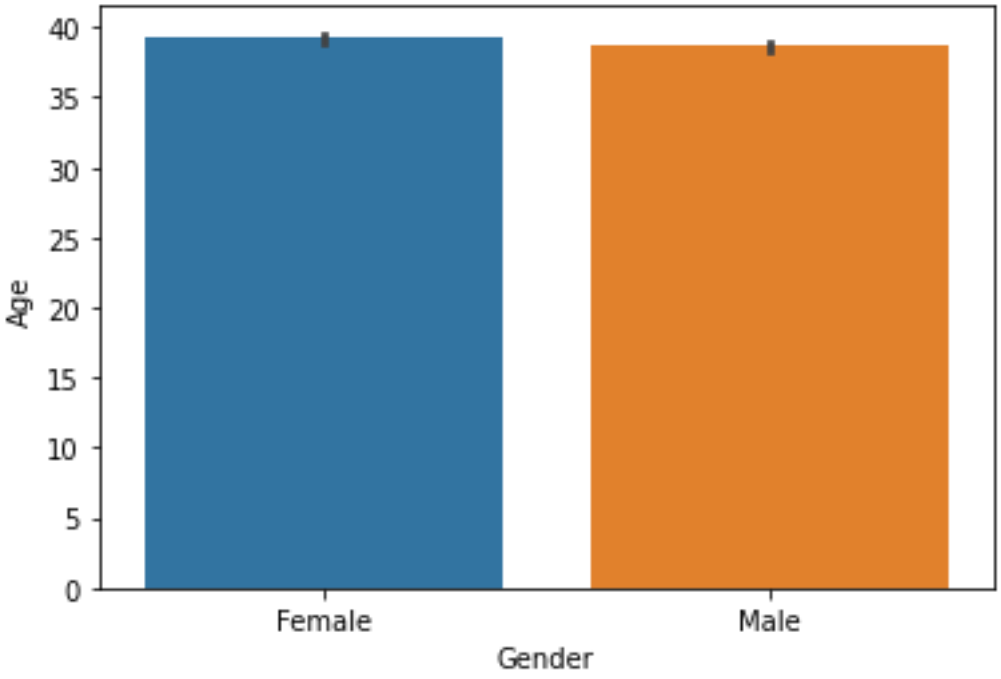
3.Perform Below Visualizations:

- Univariate Analysis:

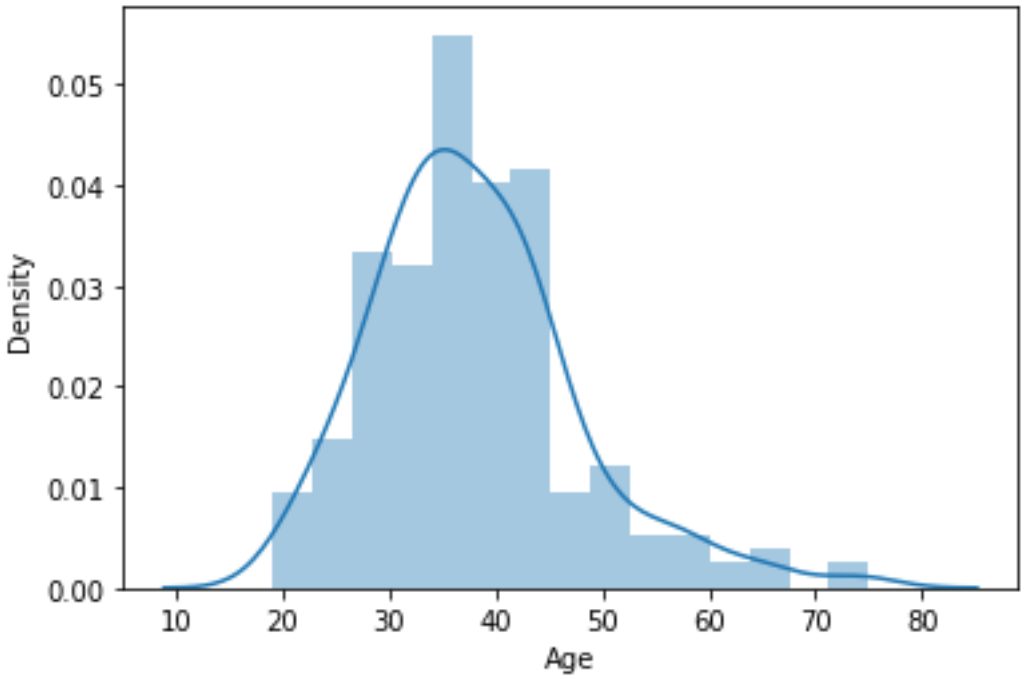
```
import matplotlib.pyplot as plt
%matplotlib inline
df['Age'].plot.hist()
```




```
sns.barplot(df['Gender'], df['Age'])
```

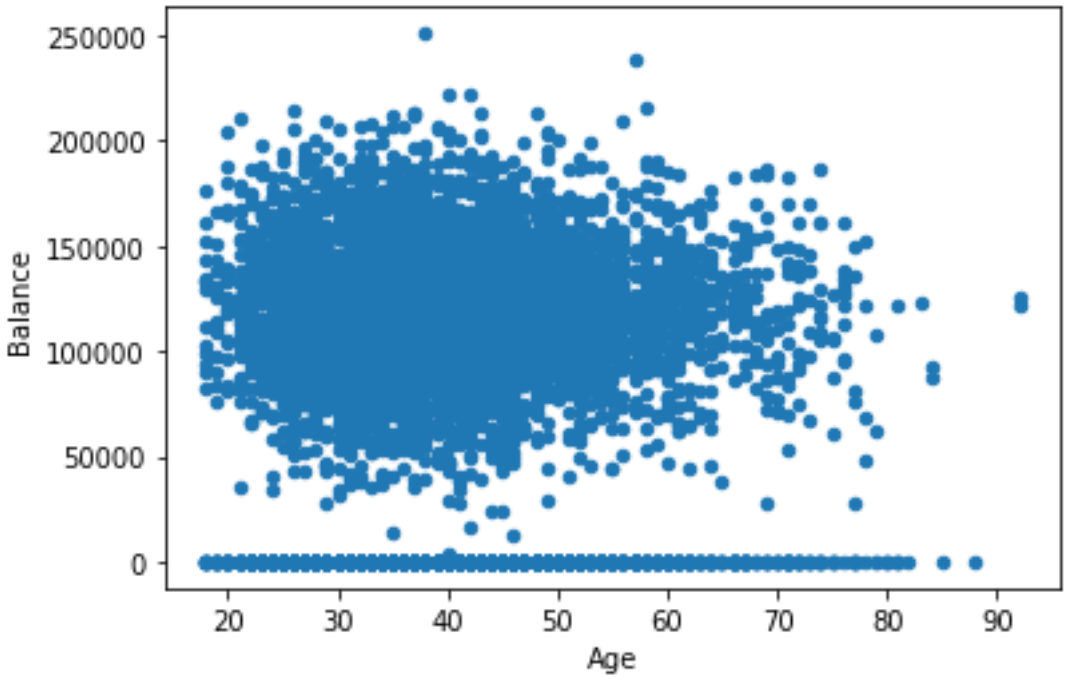



```
sns.distplot(df['Age'].head(200))
```

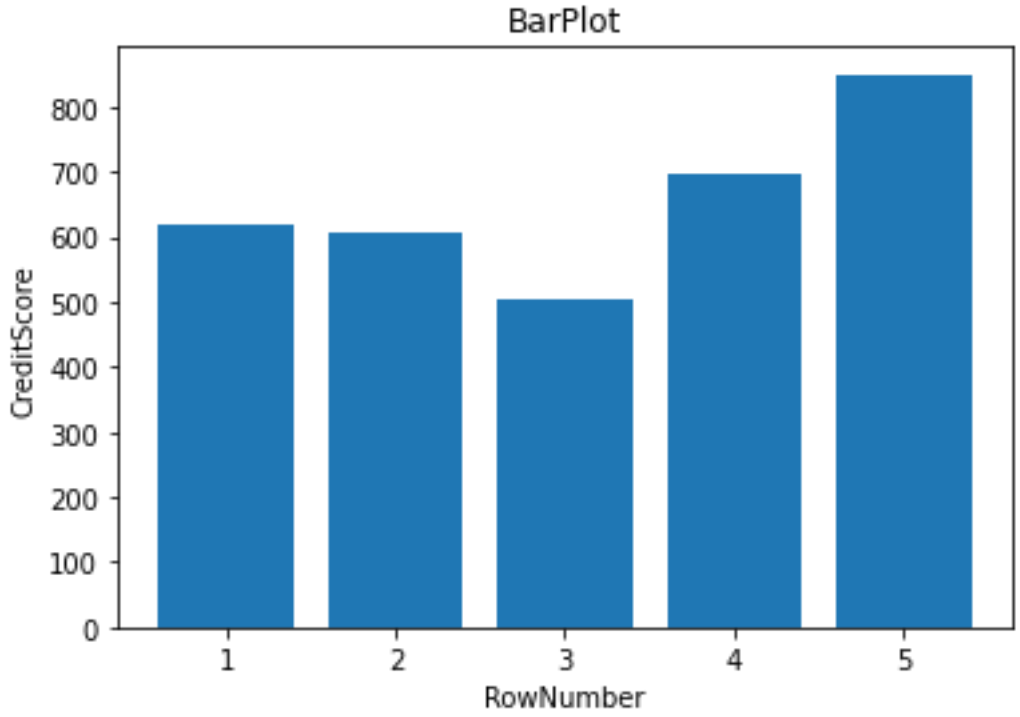


- Bivariate Analysis

```
import matplotlib.pyplot as plt
%matplotlib inline
df.plot.scatter('Age','Balance')
```



```
plt.bar(df['RowNumber'].head(),df['CreditScore'].head(),)
plt.title('BarPlot')
plt.xlabel('RowNumber')
plt.ylabel('CreditScore')
```

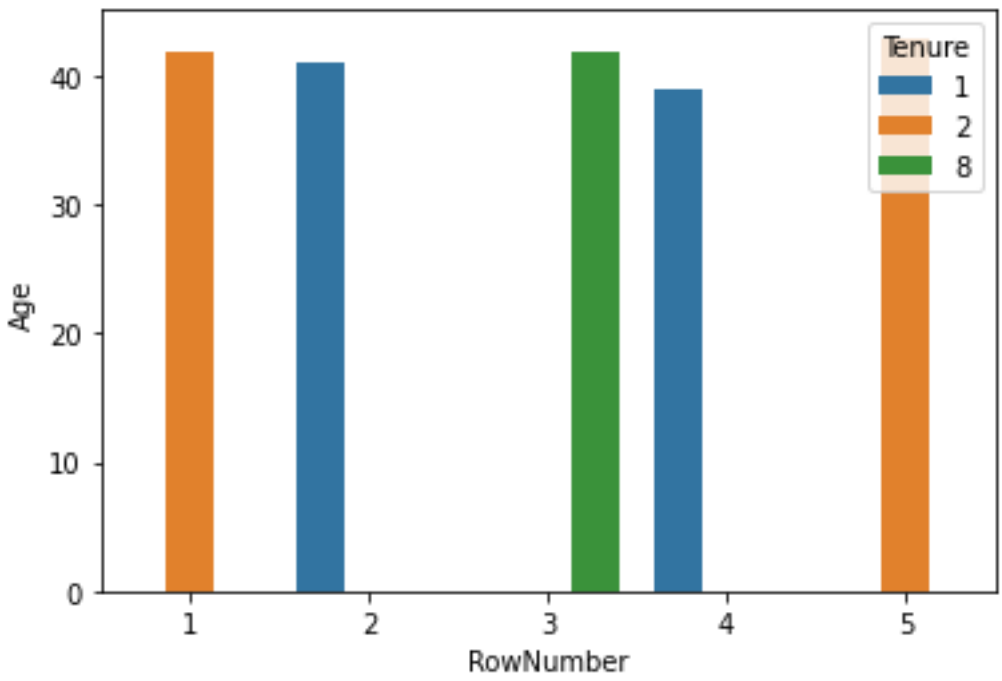



```
df.head()
```

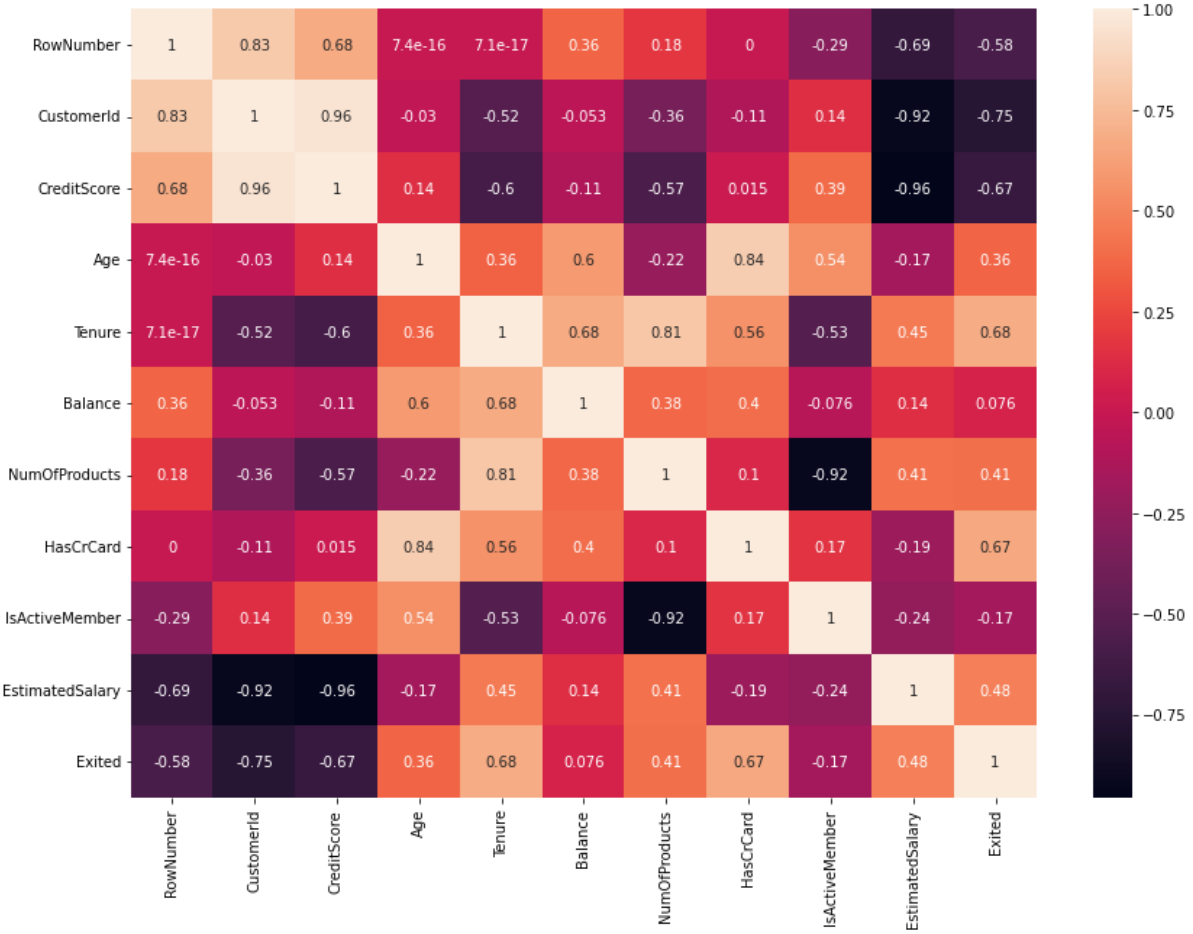
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

- Multi-variate Analysis:

```
sns.barplot('RowNumber','Age',hue='Tenure', data=df.head(
))
```




```
fig= plt.figure(figsize  =(14,10))
sns.heatmap(df.head().corr(), annot = True)
```



4.Perform descriptive statics on the dataset:

```
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

```
df.tail()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

```
df.shape
```

(1000, 14)

```
df.info()
```

RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):

#	Column	Non-Null Count	Dtype
0	RowNumber	10000 non-null	int64
1	CustomerId	10000 non-null	int64
2	Surname	10000 non-null	object
3	CreditScore	10000 non-null	int64
4	Geography	10000 non-null	object
5	Gender	10000 non-null	object
6	Age	10000 non-null	int64
7	Tenure	10000 non-null	int64
8	Balance	10000 non-null	float64
9	NumOfProducts	10000 non-null	int64
10	HasCrCard	10000 non-null	int64
11	IsActiveMember	10000 non-null	int64
12	EstimatedSalary	10000 non-null	float64
13	Exited	10000 non-null	int64

dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB

```
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

5.Handle the missing values:

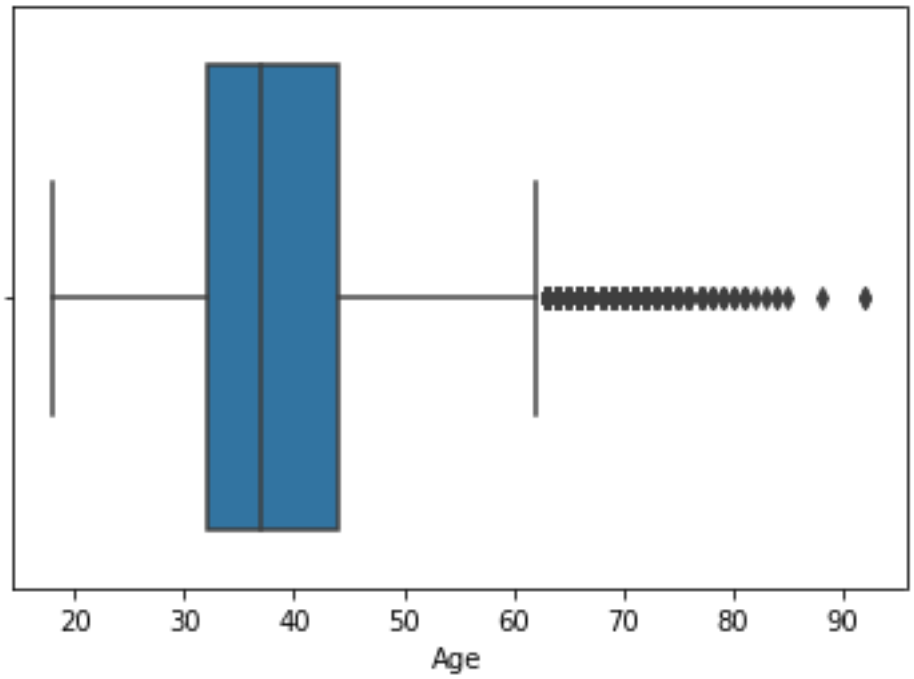
```
df.isna().any()
```

RowNumber False
CustomerId False
Surname False
CreditScore False
Geography False
Gender False
Age False
Tenure False
Balance False
NumOfProducts False
HasCrCard False
IsActiveMember False
EstimatedSalary False
Exited False dtype: bool

No missing values

6.Find the outliers and replace the outliers:

```
sns.boxplot(df['Age'])
```




```
df.mean()
```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance columns in..
"""Entry point for launching an IPython kernel.

RowNumber	5.000500e+03
CustomerId	1.569094e+07
CreditScore	6.505288e+02
Age	3.892180e+01
Tenure	5.012800e+00
Balance	7.648589e+04
NumOfProducts	1.530200e+00
HasCrCard	7.055000e-01
IsActiveMember	5.151000e-01
EstimatedSalary	1.000902e+05
Exited	2.037000e-01

dtype: float64

```
qut= df.quantile(q=[0.25,0.75])
qut
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.25	2500.75	15628528.25	584.0	32.0	3.0	0.00	1.0	0.0	0.0	51002.1100	0.0
0.75	7500.25	15753233.75	718.0	44.0	7.0	127644.24	2.0	1.0	1.0	149388.2475	0.0

```
irq=qut.loc[0.75]- qut.loc[0.25] # q3 and q1
irq
```

RowNumber 4999.5000

CustomerId	124705.5000
CreditScore	134.0000
Age	12.0000
Tenure	4.0000
Balance	127644.2400
NumOfProducts	1.0000
HasCrCard	1.0000
IsActiveMember	1.0000
EstimatedSalary	98386.1375
Exited	0.0000

dtype: float64

```
# lower
lower= qut.loc[0.25]+(1.5*irq)
lower
```

RowNumber	1.000000e+04
CustomerId	1.581559e+07
CreditScore	7.850000e+02
Age	5.000000e+01
Tenure	9.000000e+00
Balance	1.914664e+05
NumOfProducts	2.500000e+00
HasCrCard	1.500000e+00
IsActiveMember	1.500000e+00
EstimatedSalary	1.985813e+05
Exited	0.000000e+00

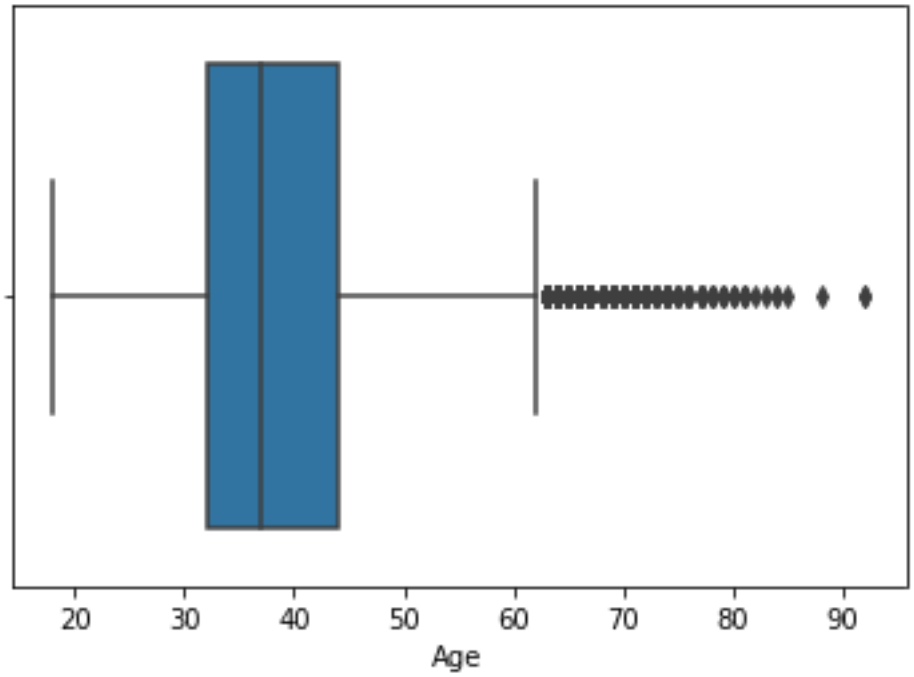
dtype: float64

```
#upper
upper= qut.loc[0.75]+(1.5*irq)
upper
```

RowNumber	1.499950e+04
CustomerId	1.594029e+07
CreditScore	9.190000e+02
Age	6.200000e+01
Tenure	1.300000e+01
Balance	3.191106e+05
NumOfProducts	3.500000e+00
HasCrCard	2.500000e+00
IsActiveMember	2.500000e+00
EstimatedSalary	2.969675e+05
Exited	0.000000e+00

dtype: float64

```
sns.boxplot(df['Age'])
```

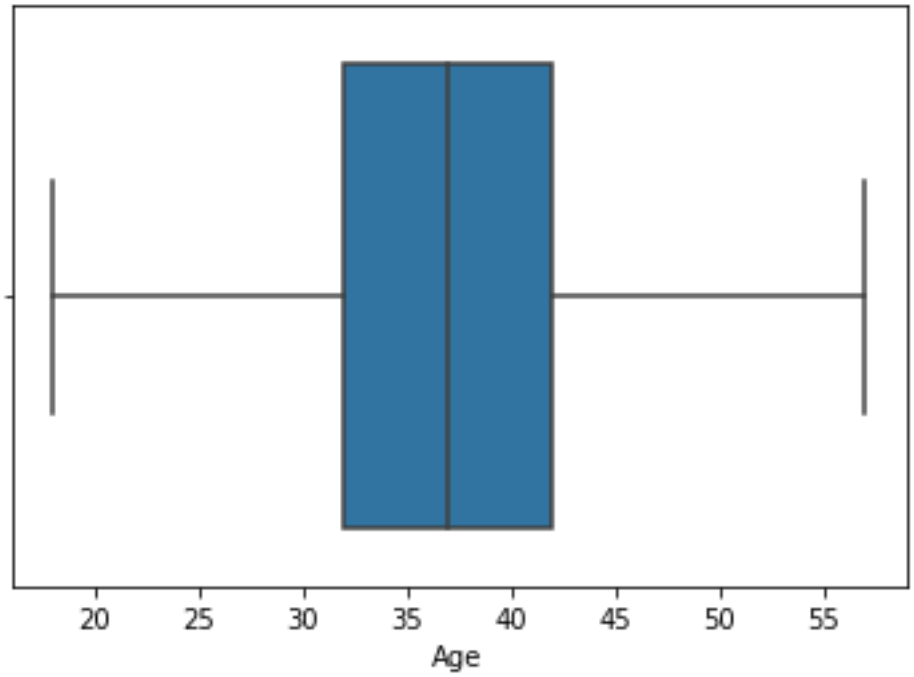



```
df['Age'].mean()
```

38.9218

```
df['Age']=np.where(df['Age']>57,39, df['Age'])
```

```
sns.boxplot(df['Age'])
```



7.Check for categorical column and perform encoding:

```
df.info()
```

RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):

#	Column	Non-Null Count	Dtype
0	RowNumber	10000 non-null	int64
1	CustomerId	10000 non-null	int64
2	Surname	10000 non-null	object
3	CreditScore	10000 non-null	int64
4	Geography	10000 non-null	object
5	Gender	10000 non-null	object
6	Age	10000 non-null	int64
7	Tenure	10000 non-null	int64
8	Balance	10000 non-null	float64
9	NumOfProducts	10000 non-null	int64
10	HasCrCard	10000 non-null	int64
11	IsActiveMember	10000 non-null	int64
12	EstimatedSalary	10000 non-null	float64
13	Exited	10000 non-null	int64

dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB

```
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

```
df.Geography.unique()
```

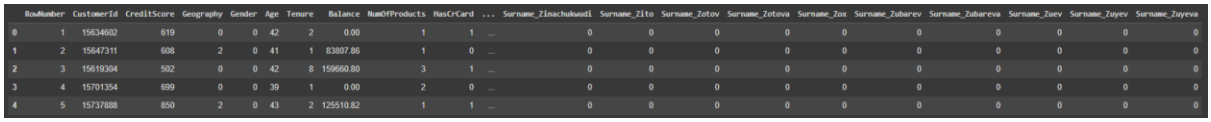
array(['France', 'Spain', 'Germany'], dtype=object)

```
df['Gender'].replace({'Female':0, 'Male': 1 }, inplace=True)
df['Geography'].replace({'France':0,'Germany':1, 'Spain':2}, i
nplace=True)
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	2	0	43	2	125510.82	1	1	1	79084.10	0

```
# using dummy values
df_d= pd.get_dummies(df,columns = ['Surname'])
```

```
df_d.head()
```



	RowNumber	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	...	Surname_Zinachukwudi	Surname_Zito	Surname_Zotov	Surname_Zotova	Surname_Zox	Surname_Zubarev	Surname_Zubareva	Surname_Zuev	Surname_Zuyev	Surname_Zuyeva
0	1	15634602	619	0	0	42	2	0.00	1	1	...	0	0	0	0	0	0	0	0	0	0
1	2	15647311	608	2	0	41	1	83807.86	1	0	...	0	0	0	0	0	0	0	0	0	0
2	3	15619304	502	0	0	42	8	159660.80	3	1	...	0	0	0	0	0	0	0	0	0	0
3	4	15701354	699	0	0	39	1	0.00	2	0	...	0	0	0	0	0	0	0	0	0	0
4	5	15737888	850	2	0	43	2	125510.82	1	1	...	0	0	0	0	0	0	0	0	0	0

8. Split the data into dependent and independent variables:

```
#splitting the dataset into x(independent variable) and y(depe
ndent variable)

x=df.iloc[:,0:10]
y=df.iloc[:,10]

print(x.shape)
print(y.shape)

print(x.columns)
#print(y)
```

(10000, 10)
(10000,)
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts'], dtype='object')

9.Scale the independent variable:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x = scale(x)
x
```

array([0.64609167, -1.54776799, 0.64609167, ..., -1.54776799, 0.64609167, 0.64609167])

10.Split the data into training and testing:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test
_size=0.25, random_state=0)

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

(7500,)
(7500,)
(2500,)
(2500,)