```
#load the dataset
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
```

```
data = pd.read_csv("/content/drive/MyDrive/Churn_Modelling.csv")
```

```
data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Ba
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	838
2	3	15619304	Onio	502	France	Female	42	8	1596
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	1255

Perform Below Visualization

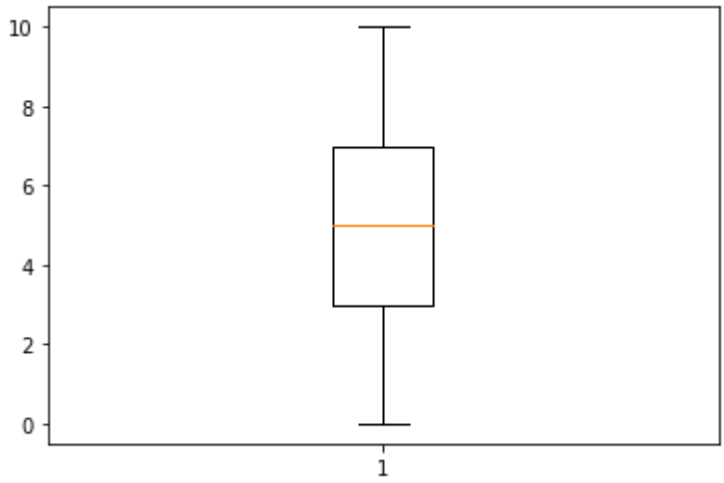
```
#univariate Analysis for Numerical data
```

```
#Histogram
data['Age'].plot(kind='hist')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7138b72e50>
```

```
#Box Plot
plt.boxplot(data['Tenure'])
```

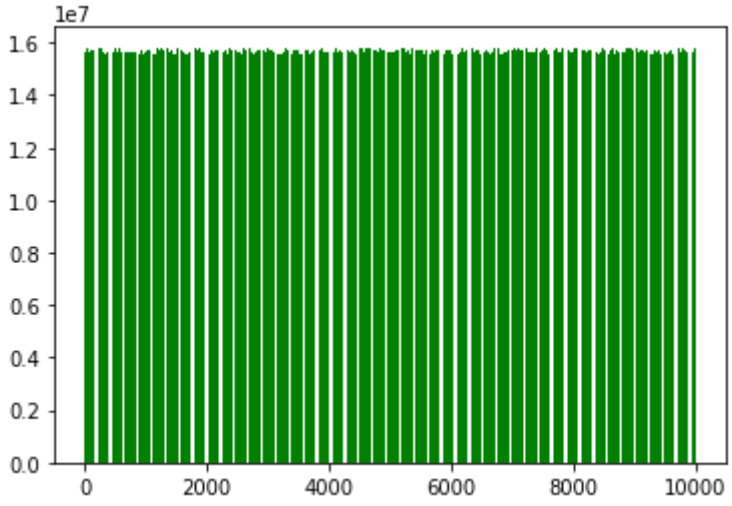
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f7138583ed0>,
  <matplotlib.lines.Line2D at 0x7f7138587450>],
 'caps': [<matplotlib.lines.Line2D at 0x7f7138587990>,
  <matplotlib.lines.Line2D at 0x7f7138587ed0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f7138583950>],
 'medians': [<matplotlib.lines.Line2D at 0x7f713858e490>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f713858e9d0>],
 'means': []}
```




```
#univariate Analysis for Categorical Data
```

```
#Bar Chart
df = pd.DataFrame(data)
X = list(df.iloc[:,0])
Y = list(df.iloc[:,1])
plt.bar(X,Y,color='g')
```

```
<BarContainer object of 10000 artists>
```



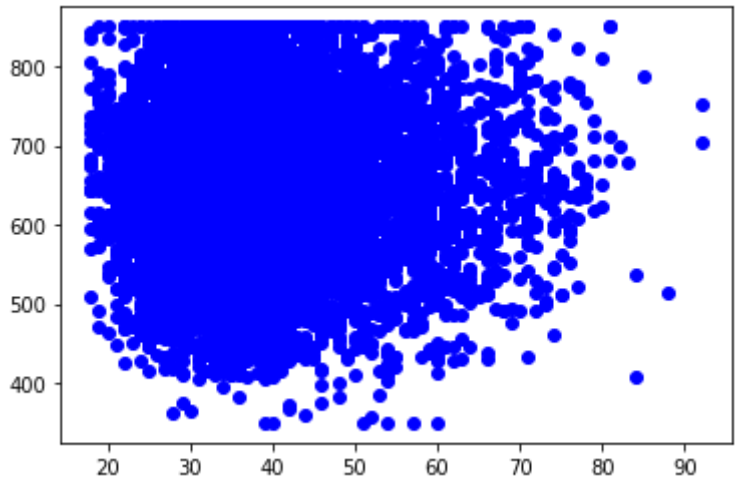
Bivariate Analysis

```
#Bivariate Analysis for Categorical Data
#Stacked Bar chart
plt.bar(data['Gender'],data['Geography'],color='r')
```

```
<BarContainer object of 10000 artists>
```


```
#Bivariate Analysis for Numerical Data
plt.scatter(data['Age'],data['CreditScore'],color='b')
```

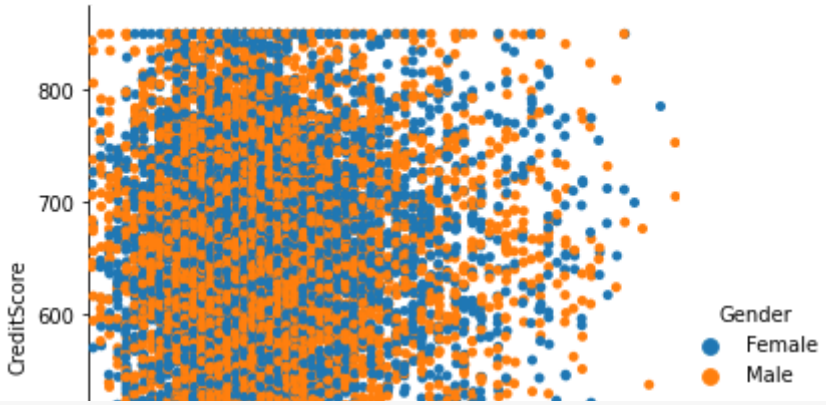
```
<matplotlib.collections.PathCollection at 0x7f713858eb50>
```



Multivariate Analysis

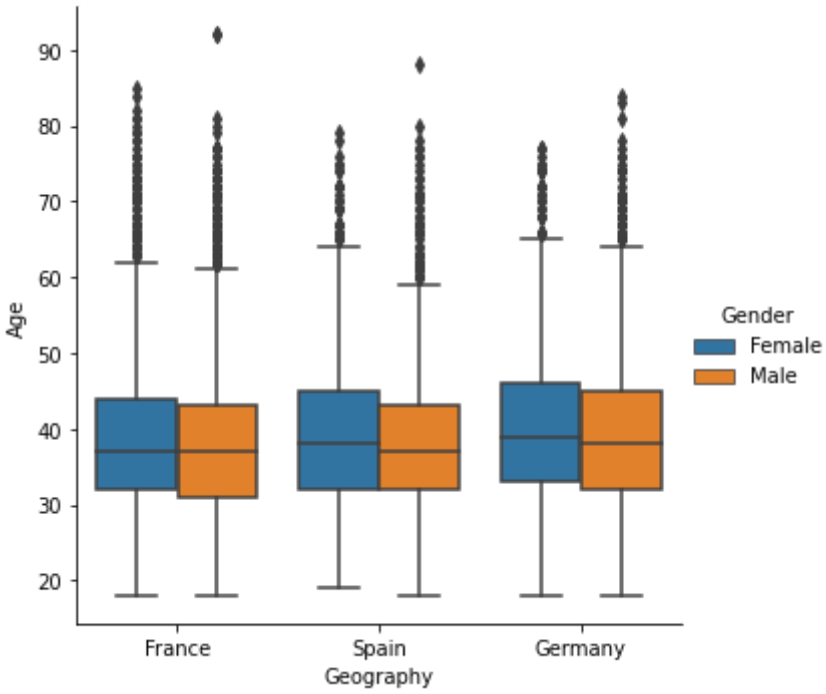
```
#Multivariate Analysis for 2 Numerical and 1 Categorical Data
#Scatter Plot
import seaborn as sns
sns.catplot(data=data, x="Age", y="CreditScore", hue="Gender")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f712a994b90>
```




```
#Multivariate Analysis for 2 Categorical and 1 Numerical Data
#Box Plot
sns.catplot(data=data, x="Geography", y="Age", hue="Gender", kind="box")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f71297ab590>
```



Question-4: Perform Descriptive Statistics on the dataset:

```
#Perform Descriptive Statistics on the Dataset
```

```
data.mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Droppi
  """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
```

```
data.median()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Droppi
  """Entry point for launching an IPython kernel.
RowNumber          5.000500e+03
CustomerId         1.569074e+07
CreditScore        6.520000e+02
Age                3.700000e+01
Tenure             5.000000e+00
Balance            9.719854e+04
NumOfProducts      1.000000e+00
HasCrCard          1.000000e+00
IsActiveMember     1.000000e+00
EstimatedSalary    1.001939e+05
Exited             0.000000e+00
dtype: float64
```

```
data.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balar
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090

```
data.shape
```

```
(10000, 14)
```

Question-5: Handle the Missing values:

```
#Handling the missing values
data.isnull().sum()
```

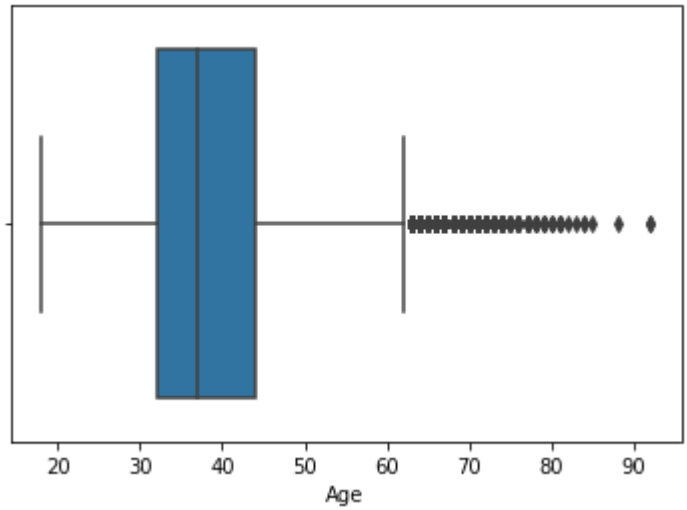
```
RowNumber          0
CustomerId         0
```

```
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

Question-6: Find the outliers and replace the outliers:

```
#Find the OutLiers and replace the outliers
sns.boxplot(data['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f7129b3ea10>
```




```
qnt=data.quantile(q=[0.25,0.75])
qnt
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	Has
0.25	2500.75	15628528.25	584.0	32.0	3.0	0.00	1.0	
0.75	7500.25	15753233.75	718.0	44.0	7.0	127644.24	2.0	

```
IQR = qnt.loc[0.75] - qnt.loc[0.25]
IQR
```

```
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
```

```
upper_extreme = qnt.loc[0.75]+1.5*IQR
upper_extreme
```

```
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
```

```
lower_extreme = qnt.loc[0.25]-1.5*IQR
lower_extreme
```

```
RowNumber         -4.998500e+03
CustomerId         1.544147e+07
CreditScore        3.830000e+02
Age                1.400000e+01
Tenure            -3.000000e+00
Balance           -1.914664e+05
NumOfProducts     -5.000000e-01
HasCrCard         -1.500000e+00
IsActiveMember    -1.500000e+00
EstimatedSalary   -9.657710e+04
Exited             0.000000e+00
dtype: float64
```

```
df2 = data[(data['Age']<upper_extreme['Age']) & (data['Age']>lower_extreme['Age'])]
```

```
data.shape
```

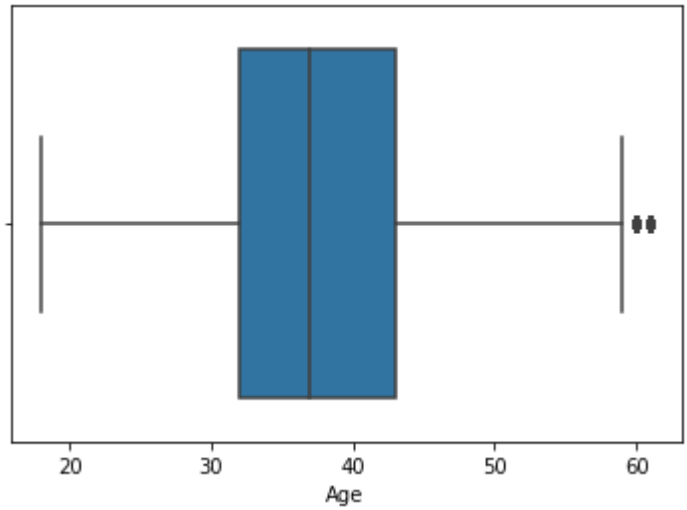
```
(10000, 14)
```

```
df2.shape
```

```
(9589, 14)
```

```
sns.boxplot(df2['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f7124cd43d0>
```



Question-7: Check for Categorical columns and perform Encoding:

```
#Check for Categorical columns and perform encoding
#Categorical are Geography and Gender
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
df2['Geography'] = le.fit_transform(df2['Geography'])
df2['Gender'] = le.fit_transform(df2['Gender'])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarnir
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarnir
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
df2.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Ba

Question-8: split the data into dependent and independent variables:

1	2	15647311	Hill	608	2	0	41	1	838

```
#Split the data into dependent and independent variables.
y=df2['EstimatedSalary']
x=df2.drop(columns=['EstimatedSalary'],axis=1)
```

4	5	15737888	Mitchell	850	2	0	43	2	1255

```
x.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Ba
0	1	15634602	Hargrave	619	0	0	42	2	
1	2	15647311	Hill	608	2	0	41	1	838
2	3	15619304	Onio	502	0	0	42	8	1596
3	4	15701354	Boni	699	0	0	39	1	
4	5	15737888	Mitchell	850	2	0	43	2	1255

Question-9: Scale the independent variables:

```
#Scale the independent variables
names=x.columns
names
```

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'Exited'],
      dtype='object')
```

```
from sklearn.preprocessing import scale
```

```
x=scale(x)
x
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_data.py:220: RuntimeWar
  mean_ = np.nanmean(X, axis)
/usr/local/lib/python3.7/dist-packages/numpy/lib/nanfunctions.py:1671: RuntimeWarning
  keepdims=keepdims)
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_data.py:228: RuntimeWar
  mean_1 = np.nanmean(Xr, axis=0)
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_data.py:236: UserWarnir
  "Numerical issues were encountered "
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_data.py:247: RuntimeWar
  mean_2 = np.nanmean(Xr, axis=0)
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_data.py:255: UserWarnir
  "Numerical issues were encountered "
array([[nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
```

```
       [nan, nan, nan, ..., nan, nan, nan],
       ...,
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan]])
```

```
x =pd.DataFrame(x, columns =[names])
```

```
x.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

Question-10: split the data into training and testing:

```
#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=0.2)
```

```
x_train
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Ter
2508	2509	15661974	Pirozzi	677	0	1	46	

```
y_train
```

```
2508    158531.01
1292     74275.08
2861     89566.64
2512    116912.45
3721     88783.59
          ...
4390    148579.43
7721     42581.09
4213     94767.77
1216     41139.05
4194    195771.95
Name: EstimatedSalary, Length: 7671, dtype: float64
```

```
x_test
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	
4429	4430	15686463	Fu	626	0	1	38	7	1
8377	8378	15567147	Ratten	802	2	1	40	4	
7357	7358	15570947	Bruny	615	2	0	29	7	1
4680	4681	15729582	Fu	676	1	1	48	3	
1069	1070	15628674	Iadanza	844	0	1	40	7	1
...	...	...	...	...	...	...	...	...	
5121	5122	15708422	Hsiung	677	2	0	35	0	
1570	1571	15607133	Shih	717	2	0	49	1	1
1789	1790	15773017	Todd	763	2	0	37	6	
5039	5040	15775490	Downie	660	0	0	38	5	1
2841	2842	15748473	Curnow	801	0	1	38	5	

1918 rows × 13 columns

```
y_test
```

```
4429     52795.56
8377     81908.09
7357    126396.01
4680    101397.86
1069     31904.31
          ...
5121     76637.38
1570    124532.90
1789    149705.25
```

```
5039    195906.59
2841     66256.27
Name: EstimatedSalary, Length: 1918, dtype: float64
```