Question-1:

Download the dataset:

The dataset “Churn_Modelling.csv” was downloaded Successfully

Question-2:

Load the Dataset:

```
[ ] #load the dataset
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

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

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

```
[4] data.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

Question-3:

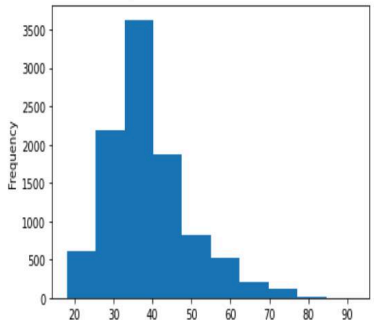
Perform Below Visualization:

Univariate Analysis

```
[5]  #Univariate Analysis for Numerical data
```

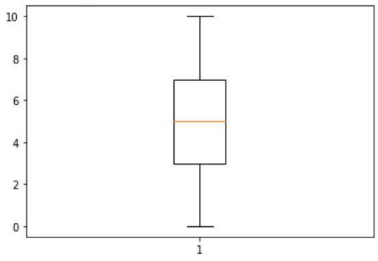
```
[6] #Histogram
    data['Age'].plot(kind='hist')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f65a0462590>
```




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

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f659f8e02d0>,
  <matplotlib.lines.Line2D at 0x7f659f8e0810>],
 'caps': [<matplotlib.lines.Line2D at 0x7f659f8e0d50>,
  <matplotlib.lines.Line2D at 0x7f659f8e62d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f659f959d10>],
 'medians': [<matplotlib.lines.Line2D at 0x7f659f8e6850>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f659f8e6d90>],
 'means': []}
```

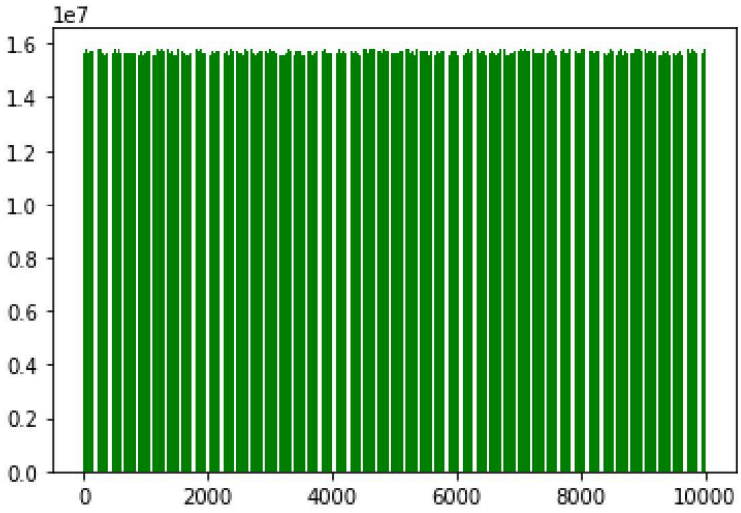



```
[ ] #Univariate Analysis for Categorical Data
```

```
[14] #Bar Chart
     df = pd.DataFrame(data)

     X = list(df.iloc[:, 0])
     Y = list(df.iloc[:, 1])
     plt.bar(X, Y, color='g')
```

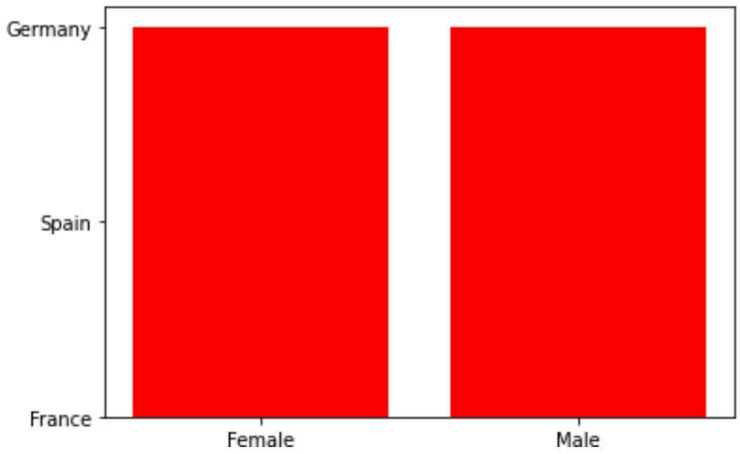
```
<BarContainer object of 10000 artists>
```



Bivariate Analysis

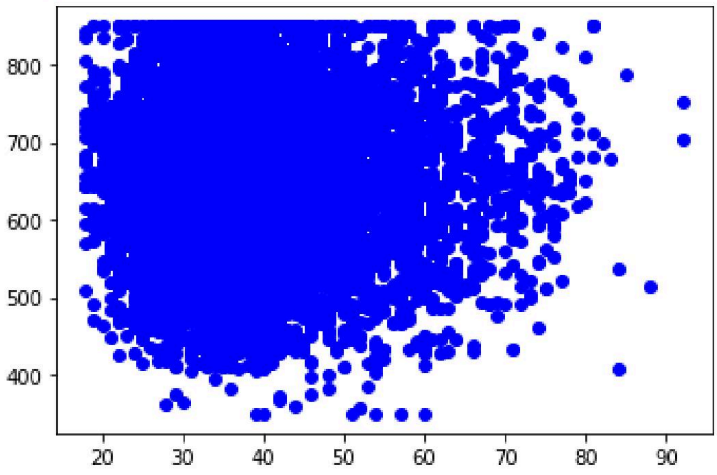
```
[23] #Bivariate Analysis for Categorical Data
     #Stacked Bar chart
     plt.bar(data['Gender'], data['Geography'], color='r')
```

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




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

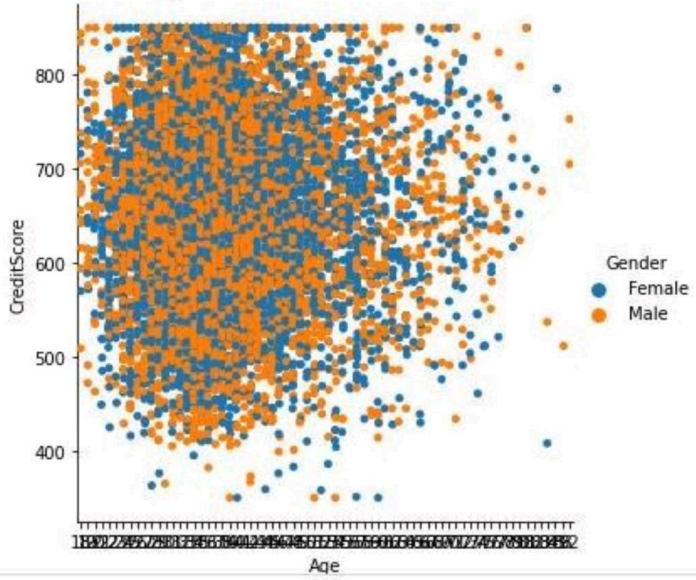
```
<matplotlib.collections.PathCollection at 0x7f6589f606d0>
```



Multivariate Analysis

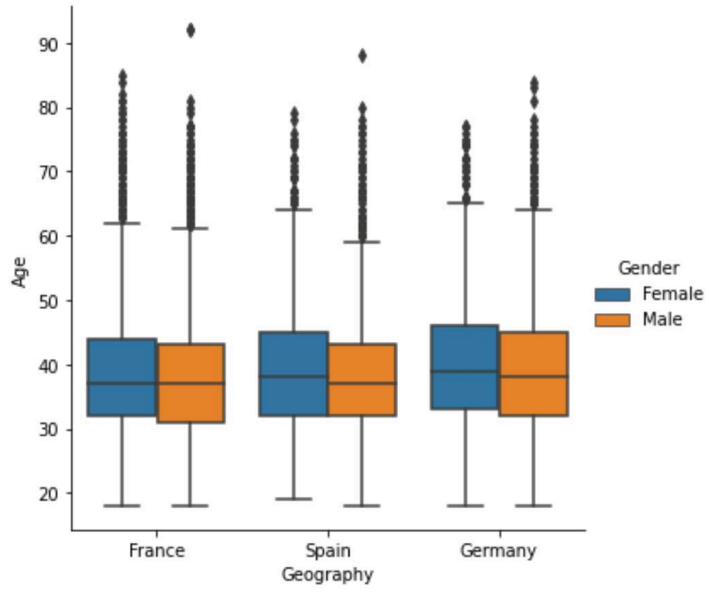
```
#Multivariate Analysis for 2 Numerical and 1 Categorical Data
#Scatter Plot
import seaborn as sns
sns.catplot(data=data, x="Age", y="CreditScore", hue="Gender")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f657aab5d90>
```




```
[32] #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 0x7f6575c43490>
```



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:
  """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
```

```
[34] data.median()
```

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

```
[36] data.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

```
[38] data.shape
```

```
(10000, 14)
```

Question-5:

Handle the Missing values:

```
[39] #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
```

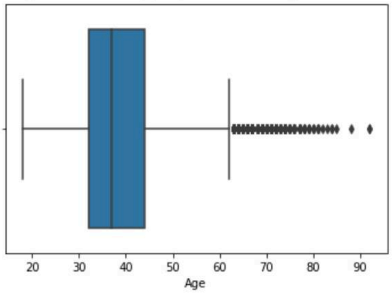
Question-6:

Find the outliers and replace the outliers:

```
[ ] #Find the Outliers and replace the outliers
```

```
[40] sns.boxplot(data['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x.
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f6575aed650>
```



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

	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

```
[42] 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
```

```
[43] 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
```

```
[44] 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
```

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

```
[50] data.shape
```

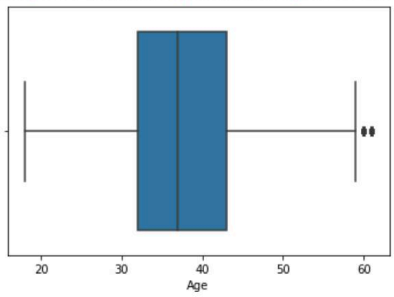
```
(10000, 14)
```

```
[49] df2.shape
```

```
(9589, 14)
```

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x.
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f6573caad10>
```



Question-7:

Check for Categorical columns and perform Encoding:

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

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

```
[76] df2.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

Question-8:

Split the data into dependent and independent variables:

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

```
[78] x.head()
```

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

Question-9:

Scale the independent variables:

```
[80] #Scale the independent variables
     names=z.columns
     names
```

```
Index(['CreditScore', 'Geography', 'Gender', 'Age'], dtype='object')
```

```
[81] from sklearn.preprocessing import scale
```

```
[82] z=scale(z)
     z
```

```
array([[-0.32370448, -0.90175758, -1.09674455,  0.50205394],
       [-0.43751069,  1.51663241, -1.09674455,  0.38636147],
       [-1.53418871, -0.90175758, -1.09674455,  0.50205394],
       ...,
       [ 0.60743723, -0.90175758, -1.09674455, -0.19210091],
       [ 1.25923643,  0.30743742,  0.91178935,  0.50205394],
       [ 1.46615681, -0.90175758, -1.09674455, -1.1176407 ]])
```

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

```
z.head()
```

	CreditScore	Geography	Gender	Age
0	-0.323704	-0.901758	-1.096745	0.502054
1	-0.437511	1.516632	-1.096745	0.386361
2	-1.534189	-0.901758	-1.096745	0.502054
3	0.503977	-0.901758	-1.096745	0.154977
4	2.066226	1.516632	-1.096745	0.617746

Question-10:

Split the data into training and testing:

```
[ ] #Split The data into Training and Testing
```

```
[89] from sklearn.model_selection import train_test_split
```

```
[90] x_train,x_test,y_train,y_test = train_test_split(z,y,test_size=0.2)
```

```
[92] x_train
```

	CreditScore	Geography	Gender	Age
1060	-0.634085	-0.901758	0.911789	-1.117641
6074	-0.437511	1.516632	-1.096745	-0.307793
7328	-1.347960	0.307437	-1.096745	-1.696103
5165	0.907472	1.516632	0.911789	0.733439
2750	-0.489241	-0.901758	-1.096745	1.196209
...	...	...	...	...
4092	1.010932	-0.901758	0.911789	0.733439
996	-0.261628	-0.901758	-1.096745	-1.580411
8690	1.186814	0.307437	-1.096745	0.964824
8099	0.131520	1.516632	-1.096745	-0.539178
1888	0.048752	1.516632	-1.096745	-0.192101

```
y_train
```

```
1104    151645.96
6334    143463.28
7638     37577.66
5392     43018.82
2851    100478.60
           ...
4269     2048.55
1037    180969.55
9056    166896.01
8440     36864.05
1960     86013.96
Name: EstimatedSalary, Length: 7671, dtype: float64
```

```
[94] x_test
```

	CreditScore	Geography	Gender	Age
962	0.772974	0.307437	0.911789	0.154977
5257	1.248890	1.516632	-1.096745	0.386361
7515	-0.841005	0.307437	-1.096745	-0.654871
6844	0.959202	-0.901758	-1.096745	-0.886256
4102	-0.996196	1.516632	-1.096745	0.386361
...	...	...	...	...
60	0.379825	0.307437	-1.096745	-1.233333
5555	0.503977	-0.901758	0.911789	-0.076408
5112	1.704115	1.516632	-1.096745	2.237441
138	0.131520	-0.901758	0.911789	-0.423486
4973	0.328095	-0.901758	-1.096745	2.353134

1918 rows × 4 columns

```
[95] y_test
```

```
1002    184023.54
5486     92914.67
7838    132038.65
7133    138780.89
4281     36242.19
           ...
61      126494.82
5797     83263.04
5337     38941.44
141     180427.24
5191       706.50
Name: EstimatedSalary, Length: 1918, dtype: float64
```