Assignment - 2
Data Visualization and Data Preprocessing

Assignment Date	27November 2022
Student Name	R.Indhumathi
Student Roll Number	813019205005
Maximum Marks	2 Marks

Source Codes and corresponding outputs from Jupyter Notebook

1. Load The Necessary Libraries

In [99]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load the Dataset

[140…
```
df=pd.read_csv(r"Churn_Modelling.csv")
```

3. View the Dataset

In [141…
```
df
```

Out[141…

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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

10000 rows × 14 columns

In [142…
```
df.head()
```

Out[142…

	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

In [103…
```
df.tail()
```

Out[103…

	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

4. Data Visualization

a. Univariate Analysis

In [104…
```
list(df.columns)
```

Output:

Out[104…
```
['RowNumber',
 'CustomerId',
 'Surname',
 'CreditScore',
 'Geography',
 'Gender',
 'Age',
 'Tenure',
 'Balance',
 'NumOfProducts',
 'HasCrCard',
 'IsActiveMember',
 'EstimatedSalary',
 'Exited']
```

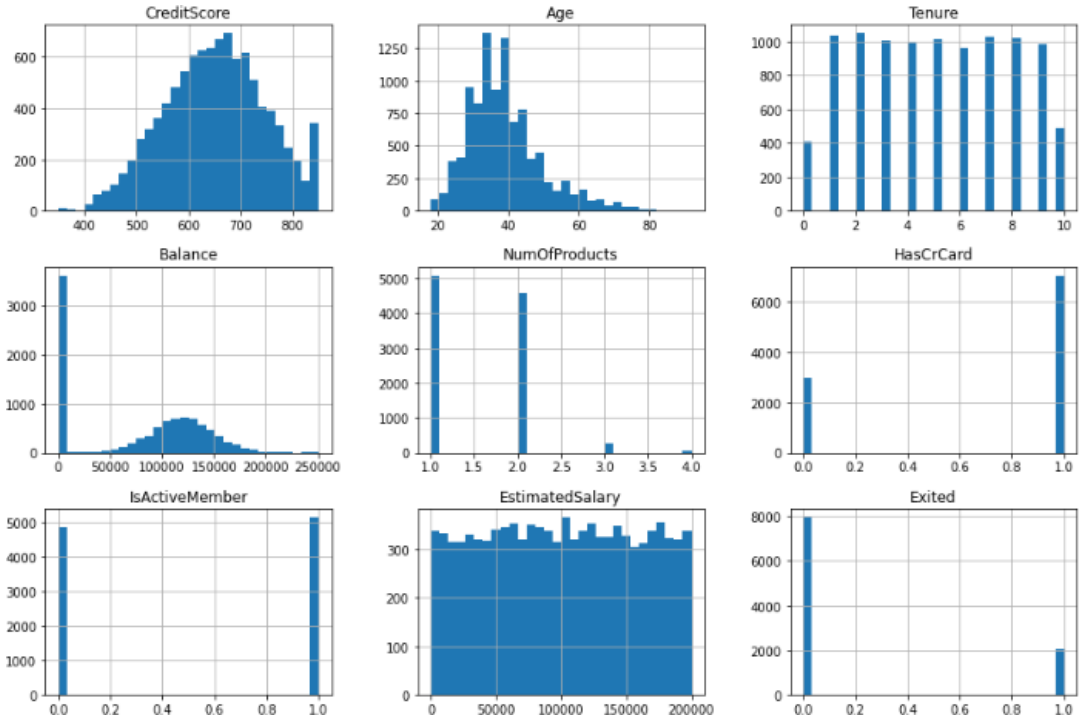
In [105…

```
df.hist(column=[
 'CreditScore',
  'Age',
 'Tenure',
 'Balance',
 'NumOfProducts',
 'HasCrCard',
 'IsActiveMember',
 'EstimatedSalary',
 'Exited'],bins=30, figsize=(15, 10))
```

Output:

Out[105…

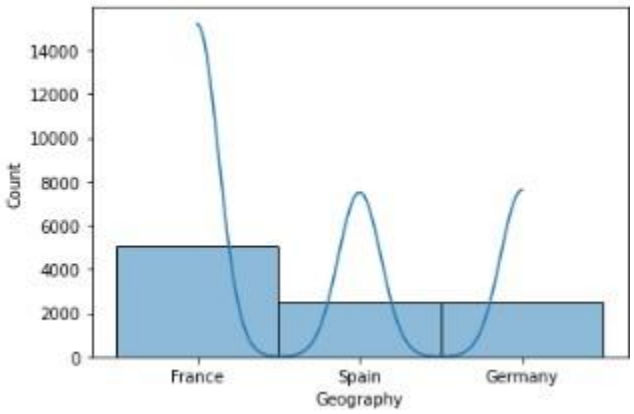
```
array([[,
        ,
        ],
       [,
        ,
        ],
       [,
        ,
        ]],
      dtype=object)
```



In [106…

```
sns.histplot(df.Geography,kde=True)
```

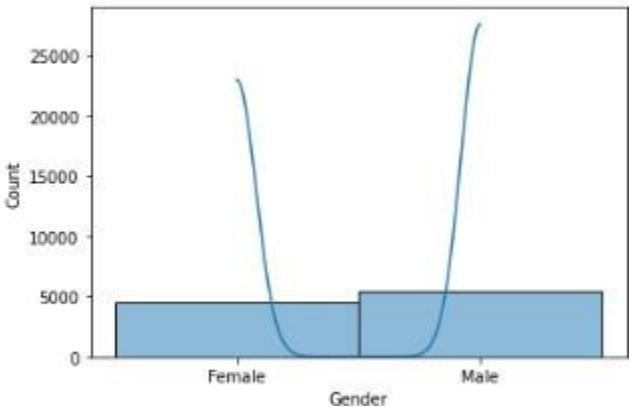
Out[106…



In [107…

```
sns.histplot(df.Gender,kde=True)
```

Out[107…

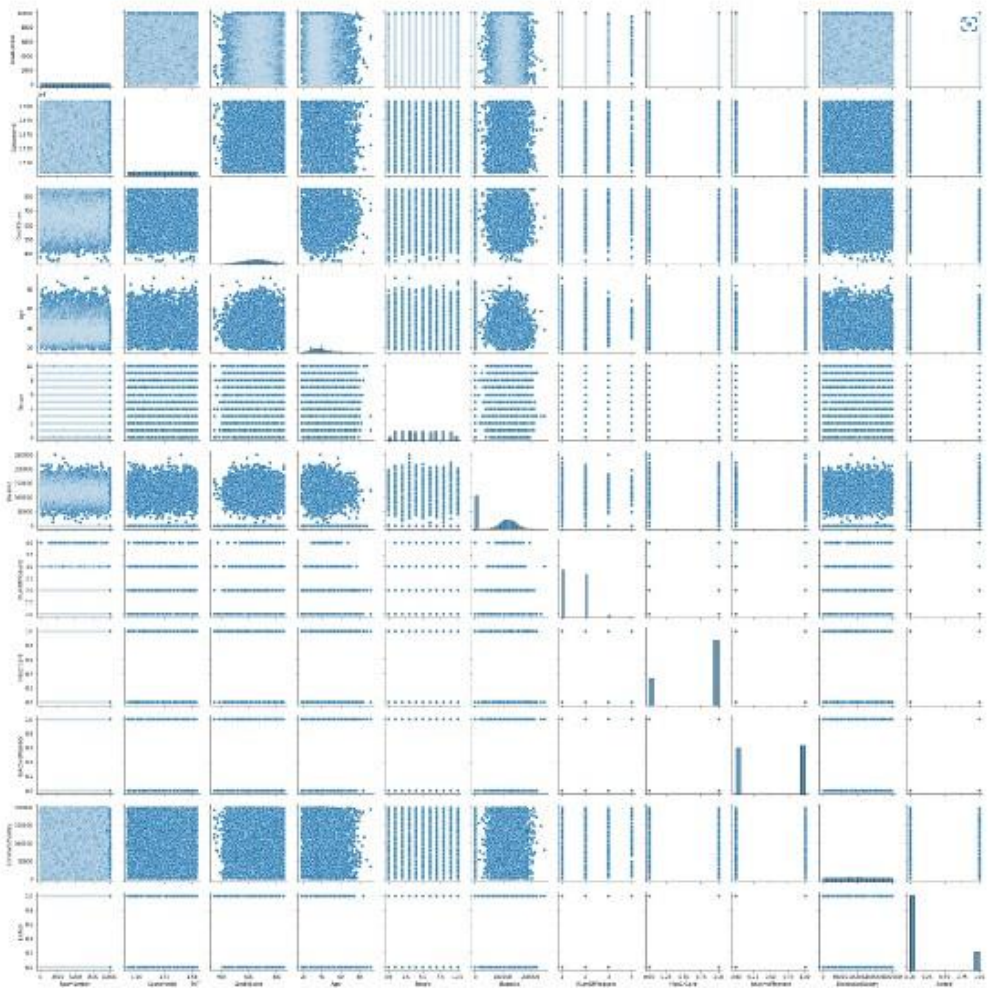


4.b. Bivariate Analysis

In [108…

```
sns.pairplot(df)
```

Out[108…

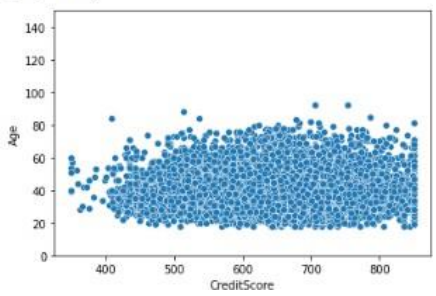


*Diagonal Plots are Univariate Plots, rest are Bivariate

In [109…

```
sns.scatterplot(df.CreditScore,df.Age)
plt.ylim(0,150)
```

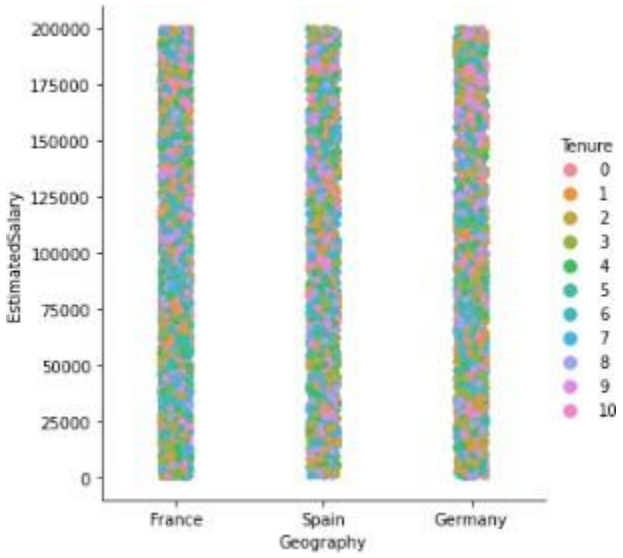
Out[109… (0.0, 150.0)



4.c. Multivariate Analysis

In [110…

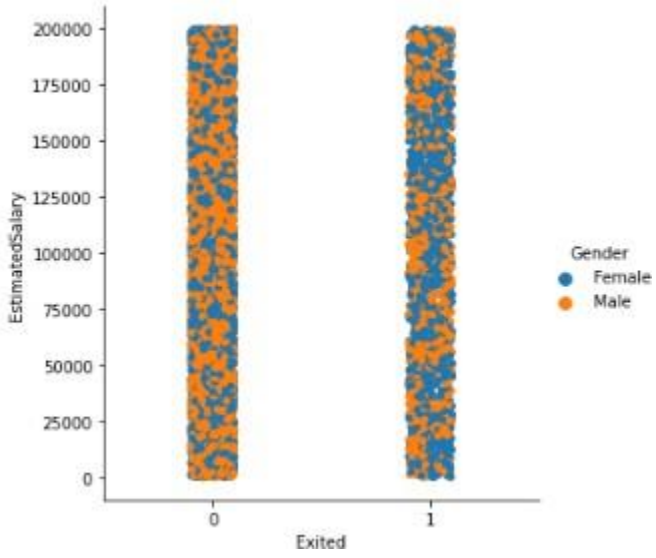
```
sns.catplot(x='Geography', y='EstimatedSalary', hue='Tenure', data=df)
```



In [111…

```
sns.catplot(x='Exited', y='EstimatedSalary', hue='Gender', data=df)
```

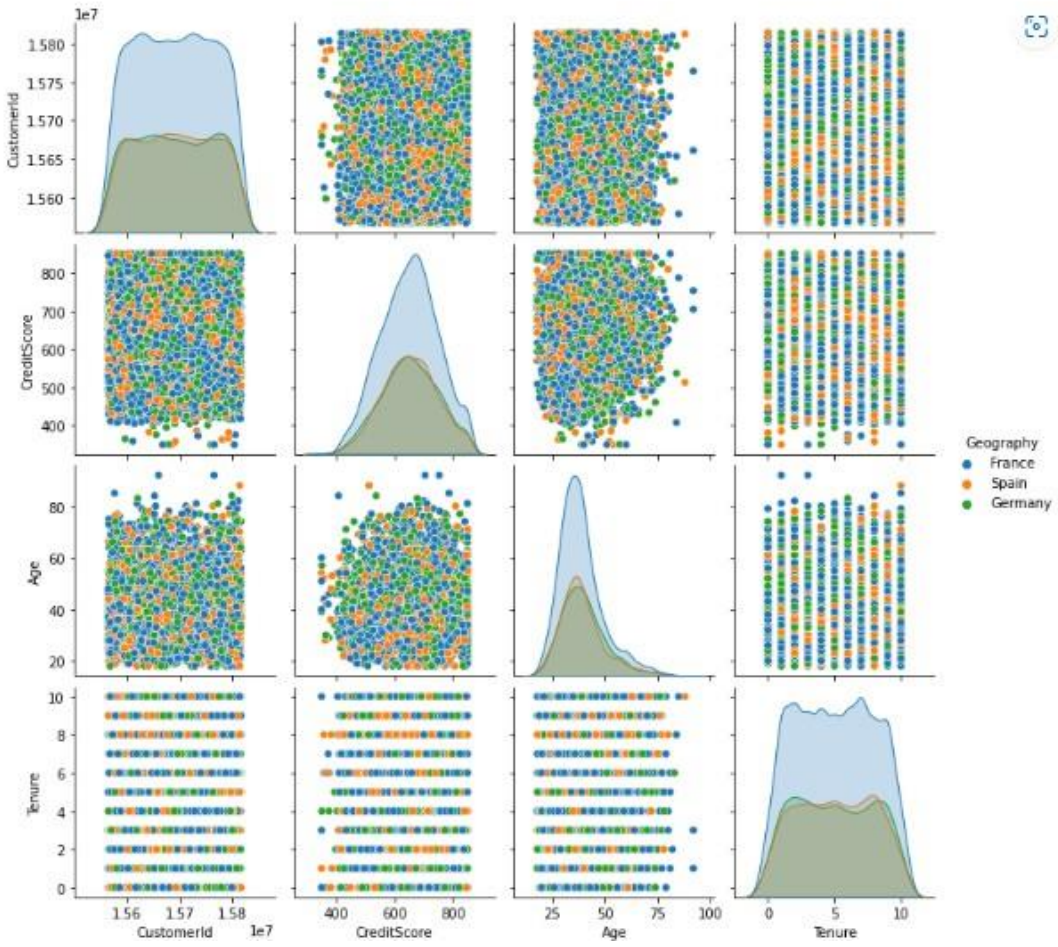
Out[111…



In [112…

```
sns.pairplot(df[['CustomerId', 'CreditScore', 'Geography','Gender', 'Age', 'Tenure']],hue='Geography')
```

Out[112…



5. Descriptive Statistics on Dataset

In [113…

```
df.describe()
```

Out[113…

	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

6. Handling Missing Values

In [114…
```
df.isnull().any()
```

Out[114…
```
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
```

df.dtypes

In [115…
```
df.isnull().sum()
```

Out[115…
```
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
```

In [116…
```
df.isnull()
```

Out[116…

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9996	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9997	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9998	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9999	False	False	False	False	False	False	False	False	False	False	False	False	False	False

10000 rows × 14 columns

***No Missing/Null values were present in the Dataset**

7. Handling Outliers

In [143…

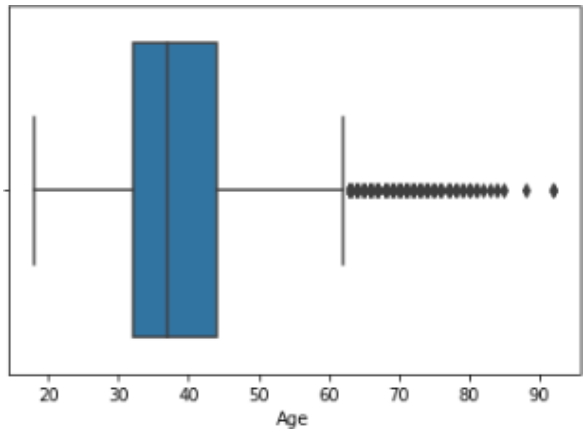
```
df.skew()
```

Out[143…

```
RowNumber           0.000
CustomerId          0.001
CreditScore        -0.071
Age                 1.011
Tenure              0.010
Balance            -0.141
NumOfProducts       0.745
HasCrCard          -0.901
IsActiveMember     -0.060
EstimatedSalary     0.002
Exited              1.471
dtype: float64
```

In [144…

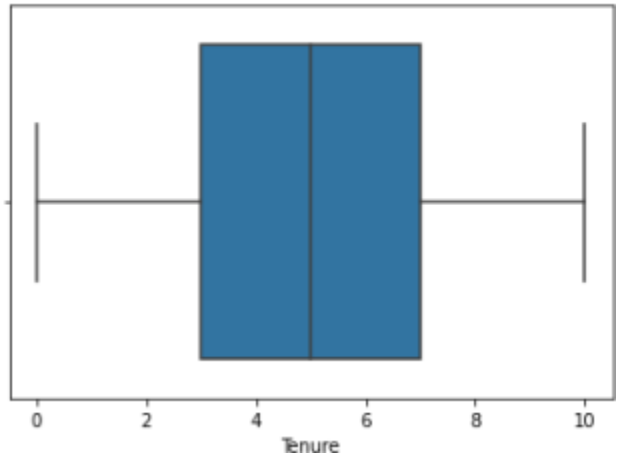
```
sns.boxplot(x=df['Age'])
```



In [145…

```
sns.boxplot(x=df['Tenure'])
```

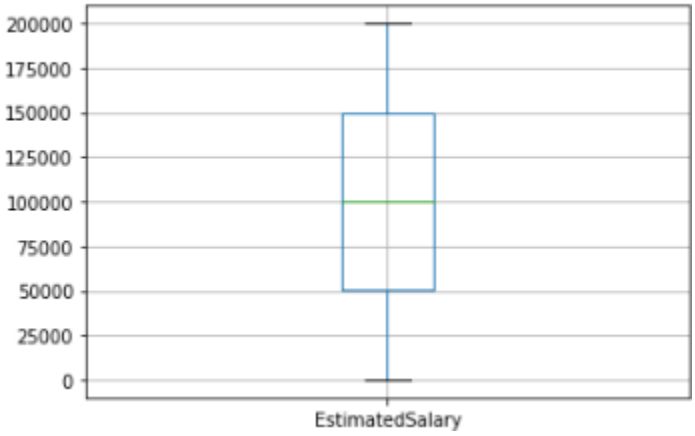
Out[145…



In [146…

```
df.boxplot(column="EstimatedSalary")
```

Out[146…

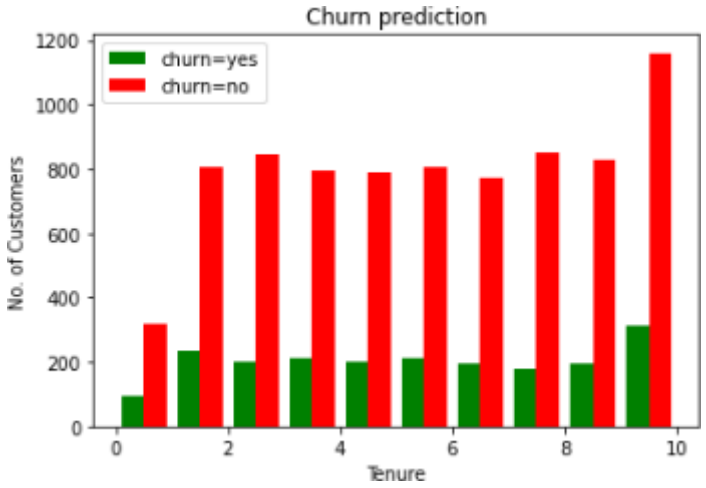


In [147…

```
churn_yes=df[df.Exited==1].Tenure
churn_no=df[df.Exited==0].Tenure
```

In [148…

```
plt.xlabel("Tenure")
plt.ylabel("No. of Customers")
plt.title('Churn prediction')
plt.hist([churn_yes,churn_no],color=["green","red"],label=["churn=yes","churn=no"])
plt.legend()
plt.show()
```



***IsExited is a target variable and can't be dealt as an outlier, age is the next highest, hence it is considered**

In [149…

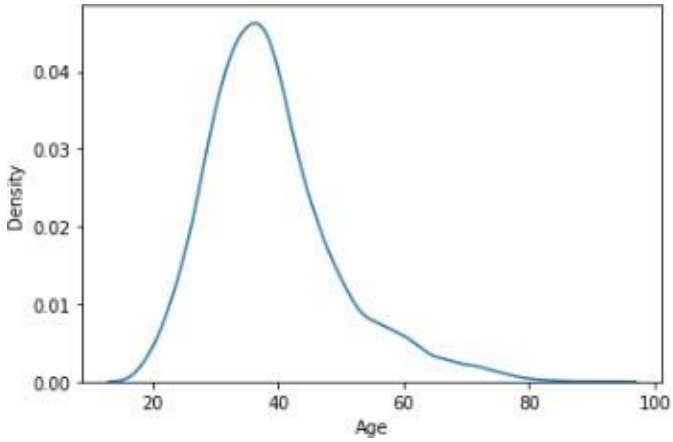
```
df.describe()
```

Out[149…

	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

In [150…

```
sns.kdeplot(df["Age"])
```



In [151…
```
q1 = df["Age"].describe()["25%"]
q1
```

Out[151… 32.0

In [152…
```
q3 = df["Age"].describe()["75%"]
q3
```

Out[152… 44.0

In [153…
```
iqr = q3-q1  # iqr
iqr
```

Out[153… 12.0

In [154…
```
l_b = q1 -(1.5*iqr)
u_b = q3 + (1.5*iqr)
```

In [155…
```
l_b
```

Out[155… 14.0

In [156…
```
u_b
```

Out[156… 62.0

In [131…
```
#handling the outliers
#1 remove the rows
#2 replace the outliers with (l_b,u_b,mean,meadian)
```

In [157…
```
df[df["Age"]<l_b]
```

Out[157…

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited

In [158…
```
df[df["Age"]>u_b]
```

Out[158…

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
58	59	15623944	T'ien	511	Spain	Female	66	4	0.00	1	1	0	1643.11	1
85	86	15805254	Ndukaku	652	Spain	Female	75	10	0.00	2	1	1	114675.75	0
104	105	15804919	Dunbabin	670	Spain	Female	65	1	0.00	1	1	1	177655.68	1
158	159	15589975	Maclean	646	France	Female	73	6	97259.25	1	0	1	104719.66	0
181	182	15789669	Hsia	510	France	Male	65	2	0.00	2	1	1	48071.61	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
9753	9754	15705174	Chiedozie	656	Germany	Male	68	7	153545.11	1	1	1	186574.68	0
9765	9766	15777067	Thomas	445	France	Male	64	2	136770.67	1	0	1	43678.06	0
9832	9833	15814690	Chukwujekwu	595	Germany	Female	64	2	105736.32	1	1	1	89935.73	1
9894	9895	15704795	Vagin	521	France	Female	77	6	0.00	2	1	1	49054.10	0
9936	9937	15653037	Parks	609	France	Male	77	1	0.00	1	0	1	18708.76	0

359 rows × 14 columns

In [138…

```
# replace the outliers with (l_b,u_b,mean,meadian)
```

In [160…

```
outlier_list = list(df[df["Age"] > u_b]["Age"])
```

In [161…

```
outlier_list
```

Out[161…

```
[66,
 75,
 65,
 73,
 65,
 72,
 67,
 67,
 79,
 80,
 68,
 75,
 66,
 66,
 70,
 63,
 72,
 64,
 64,
 70,
 67,
 82,
 63,
 69,
 65,
 69,
 64,
 65,
 74,
 67,
 66,
 67,
 63,
 70,
 71,
 72,
 67,
 74,
 76,
 66,
 63,
 66,
 68,
```

In [162…

```
outlier_dict = {}.fromkeys(outlier_list,u_b)
outlier_dict
```

Out[162…

```
{66: 62.0,
 75: 62.0,
 65: 62.0,
 73: 62.0,
 72: 62.0,
 67: 62.0,
 79: 62.0,
 80: 62.0,
 68: 62.0,
 70: 62.0,
 63: 62.0,
 64: 62.0,
 82: 62.0,
 69: 62.0,
 74: 62.0,
 71: 62.0,
 76: 62.0,
 77: 62.0,
 88: 62.0,
 85: 62.0,
 84: 62.0,
 78: 62.0,
 81: 62.0,
 92: 62.0,
 83: 62.0}
```

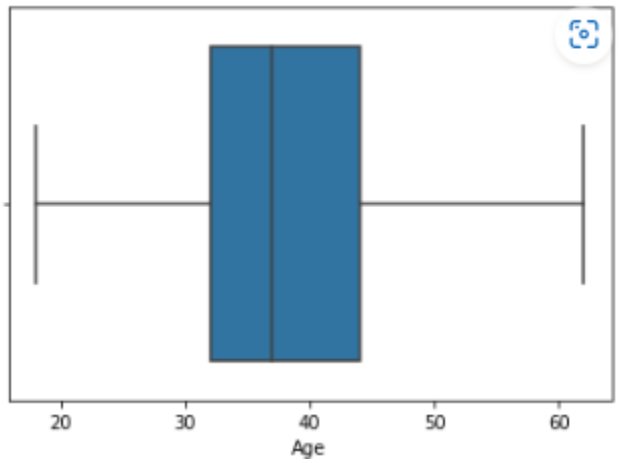
In [163…

```
df["Age"] = df["Age"].replace(outlier_dict)
```

In [165…

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

Out[165…



In [166…
```
df[df["Age"]>u_b]
```

Out[166…

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited

8.Check for Categorical columns and perform encoding

In [167…
```
df.head()
```

Out[167…

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42.0	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41.0	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42.0	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39.0	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43.0	2	125510.82	1	1	1	79084.10	0

In [168…
```
df["Geography"].unique()
```

Out[168… `array(['France', 'Spain', 'Germany'], dtype=object)`

In [169…
```
df["Gender"].unique()
```

Out[169… `array(['Female', 'Male'], dtype=object)`

***Surname is also a Categorical Field, but do not affect the result of the model, hence it's not considered**

In [170…
```
from sklearn.compose import ColumnTransformer
```

In [171…
```
from sklearn.preprocessing import OneHotEncoder
```

In [172…
```
ct=ColumnTransformer([("oh",OneHotEncoder(),[1,2])],remainder="passthrough")
```

9. Split the data into dependent and independent variables.

In [173…
```
x=df.iloc[:,3:13].values
```

In [174…
```
x.shape
```

Out[174… (10000, 10)

In [175…
```
y=df.iloc[:,13:14].values
```

In [176…
```
y.shape
```

Out[176… (10000, 1)

***Is Exited is the only Independent variable, and Row number,ID and Surname are not necessary to be considered as a dependant variable**

In [177…
```
x=ct.fit_transform(x)
```

In [178…
```
x.shape
```

Out[178… (10000, 13)

In [179…
```
x
```

Out[179…
```
array([[1.0, 0.0, 0.0, ..., 1, 1, 101348.88],
       [0.0, 0.0, 1.0, ..., 0, 1, 112542.58],
       [1.0, 0.0, 0.0, ..., 1, 0, 113931.57],
       ...,
       [1.0, 0.0, 0.0, ..., 0, 1, 42085.58],
       [0.0, 1.0, 0.0, ..., 1, 0, 92888.52],
       [1.0, 0.0, 0.0, ..., 1, 0, 38190.78]], dtype=object)
```

In [180…
```
import joblib
joblib.dump(ct,"churnct.pkl")
```

Out[180… ['churnct.pkl']

10.Split the data into training and testing

In [182…
```
from sklearn.model_selection import train_test_split
```

In [187…
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [188…
```
x_train.shape
```

Out[188… (8000, 13)

In [189…
```
x_test.shape
```

Out[189… (2000, 13)

11.Scale the independent variables

In [190…
```
from sklearn.preprocessing import StandardScaler
```

In [191…
```
sc=StandardScaler()
```

In [192…
```
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In [193…
```
joblib.dump(sc,"churnsc.pkl")
```

Out[193… ['churnsc.pkl']