1.Download the dataset: Dataset

2.Load the dataset

In [ ]:

```
import numpy as np
import pandas as pd
df = pd.read_csv("Churn_Modelling.csv")
```

3.Perform Below Visualizations.

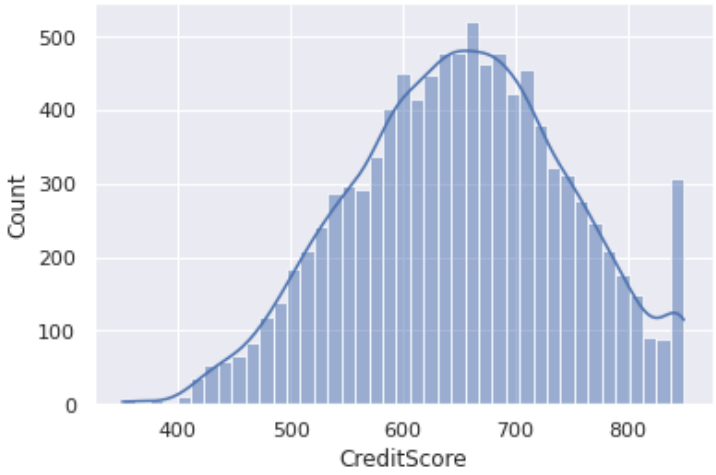
Univariate Analysis

In [ ]:

```
import seaborn as sns
sns.histplot(df.CreditScore,kde=True)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb65973a3d0>
```



Bi - Variate Analysis

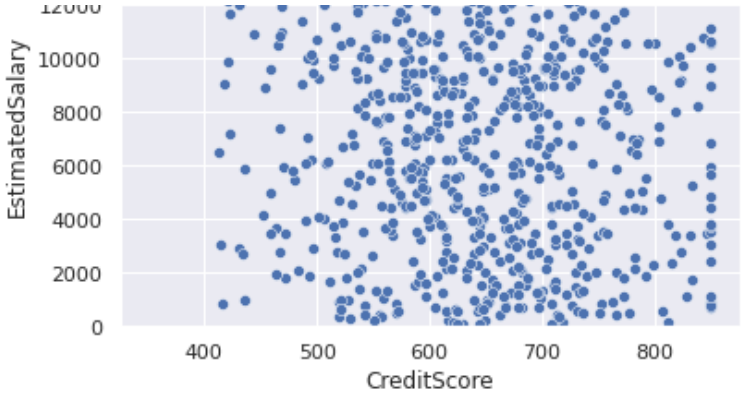
In [ ]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.scatterplot(df.CreditScore,df.EstimatedSalary)
plt.ylim(0,15000)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will resu
lt in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

```
(0.0, 15000.0)
```



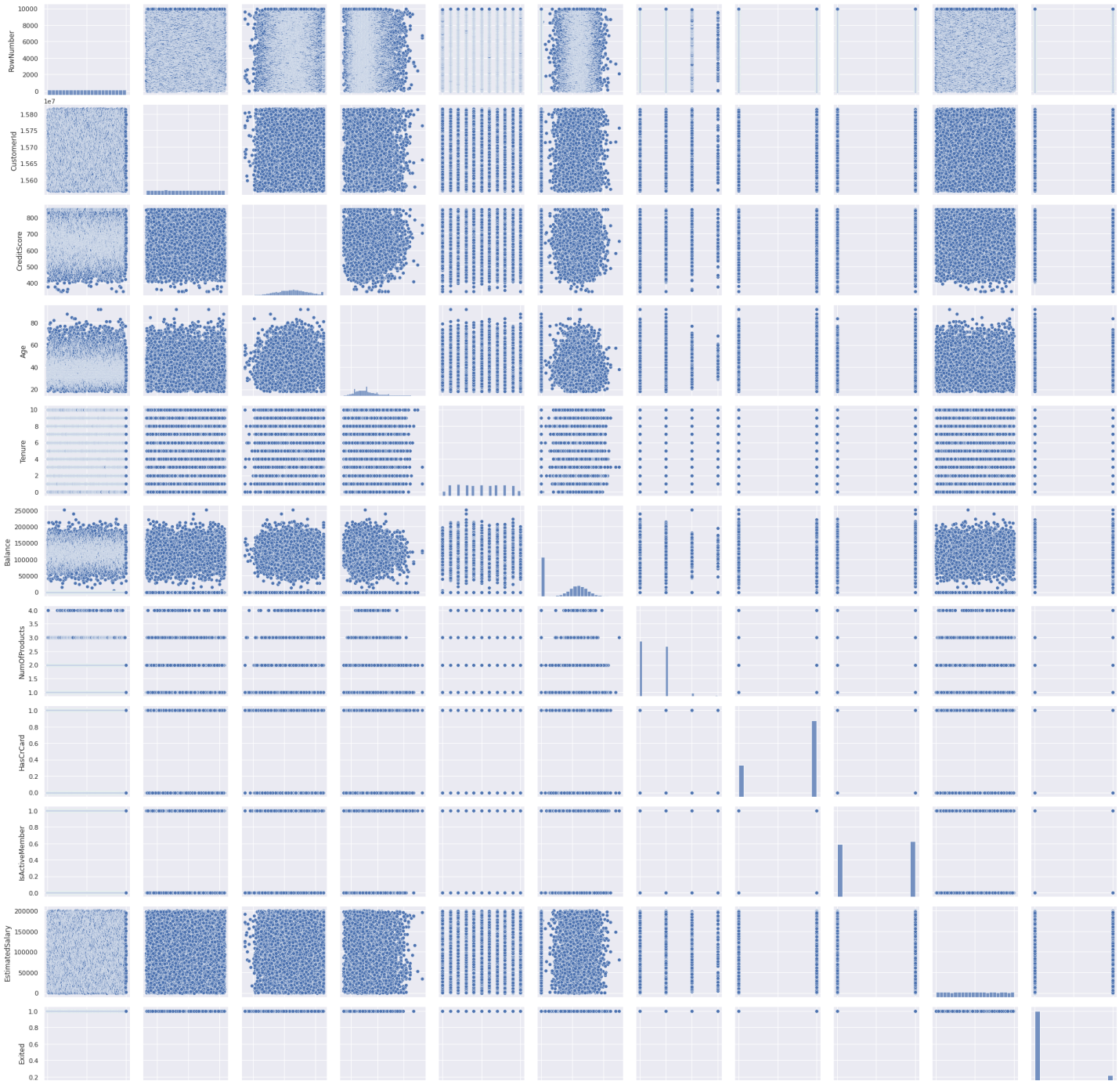
Multi - Variate Analysis

In [ ]:

```
import seaborn as sns
df=pd.read_csv("Churn_Modelling.csv")
sns.pairplot(df)
```

Out[ ]:

```
<seaborn.axisgrid.PairGrid at 0x7fb6430f1590>
```



4.Perform descriptive statistics on the dataset.

In [ ]:

```
df=pd.read_csv("Churn_Modelling.csv")
df.describe(include='all')
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
count	10000.00000	1.000000e+04	10000	10000.000000	10000	10000	10000.000000	10000.000000	10000.000000
unique	NaN	NaN	2932	NaN	3	2	NaN	NaN	NaN
top	NaN	NaN	Smith	NaN	France	Male	NaN	NaN	NaN
freq	NaN	NaN	32	NaN	5014	5457	NaN	NaN	NaN
mean	5000.50000	1.569094e+07	NaN	650.528800	NaN	NaN	38.921800	5.012800	76485.889288
std	2886.89568	7.193619e+04	NaN	96.653299	NaN	NaN	10.487806	2.892174	62397.405202
min	1.00000	1.556570e+07	NaN	350.000000	NaN	NaN	18.000000	0.000000	0.000000
25%	2500.75000	1.562853e+07	NaN	584.000000	NaN	NaN	32.000000	3.000000	0.000000
50%	5000.50000	1.569074e+07	NaN	652.000000	NaN	NaN	37.000000	5.000000	97198.540000
75%	7500.25000	1.575323e+07	NaN	718.000000	NaN	NaN	44.000000	7.000000	127644.240000
max	10000.00000	1.581569e+07	NaN	850.000000	NaN	NaN	92.000000	10.000000	250898.090000

In [ ]:

```
df.count()
```

Out[ ]:

```
RowNumber          10000
CustomerId         10000
Surname            10000
CreditScore        10000
Geography          10000
Gender             10000
Age                10000
Tenure             10000
Balance            10000
NumOfProducts      10000
HasCrCard          10000
IsActiveMember     10000
EstimatedSalary    10000
Exited             10000
dtype: int64
```

In [ ]:

```
df['Geography'].value_counts()
```

Out[ ]:

```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

5.Handle the Missing values.

In [ ]:

In [ ]:

```
from ast import increment_lineno
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(color_codes=True)
df=pd.read_csv("Churn_Modelling.csv")
df.head()
```

Out[ ]:

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

In [ ]:

```
df.isnull().sum()
```

Out[ ]:

```
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
```

No missing values here, so no need to perform further operations

6.Find the outliers and replace the outliers

In [ ]:

```
import pandas as pd
import matplotlib
from matplotlib import pyplot as pyplot
%matplotlib inline
matplotlib.rcParams['figure.figsize']=(10,4)
df=pd.read_csv("Churn_Modelling.csv")
df.sample(5)
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	Has
648	649	15633064	Stonebraker	438	France	Female	36	4	0.00	2	
4872	4873	15645937	Guerin	790	Spain	Male	32	3	0.00	1	
7431	7432	15705379	Upjohn	678	France	Male	38	3	0.00	2	

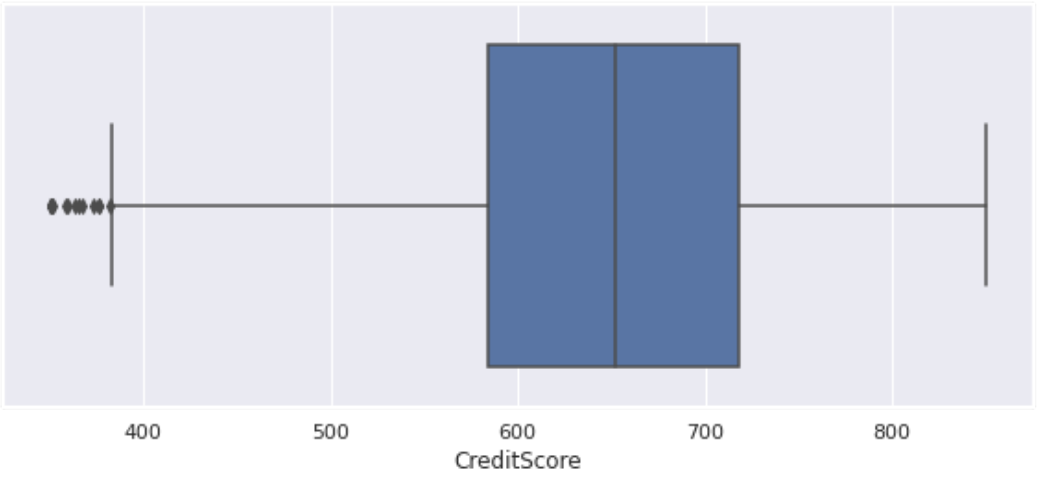
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	Has
7459	7460	15583724	Raymond	645	Spain	Female	29	4	0.00	2	
6639	6640	15583076	Deleon	588	Germany	Male	41	6	106116.56	2	

In [ ]:

```
sns.boxplot(x='CreditScore', data=df)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb63b0a8ad0>
```



7.Check for Categorical columns and perform encoding.

In [ ]:

```
df=pd.read_csv("Churn_Modelling.csv")
df.columns
import pandas as pd
import numpy as np
headers=['RowNumber','CustomerID','Surname','CreditScore','Geography',
 'Gender','Age','Tenure','Balance','NumofProducts','HasCard'
 'IsActiveMember','EstimatedSalary','Exited']
import seaborn as sns
df.head()
```

Out[ ]:

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

8.Split the data into dependent and independent variables.

In [ ]:

```
#Splitting the Dataset into the Independent Feature Matrix:
X = df.iloc[:, :-1].values
print(X)
```

```
[[1 15634602 'Hargrave' ... 1 1 101348.88]
 [2 15647311 'Hill' ... 0 1 112542.58]
 [3 15619304 'Onio' ... 1 0 113931.57]
```

```
 [3 15619304 'Onio' ... 1 0 113931.57]
 ...
 [9998 15584532 'Liu' ... 0 1 42085.58]
 [9999 15682355 'Sabbatini' ... 1 0 92888.52]
 [10000 15628319 'Walker' ... 1 0 38190.78]]
```

In [ ]:

```
#Extracting the Dataset to Get the Dependent Vector
Y = df.iloc[:, -1].values
print(Y)
```

```
[1 0 1 ... 1 1 0]
```

9.Scale the independent variables

In [ ]:

```
from sklearn.preprocessing import StandardScaler
```

In [ ]:

```
object= StandardScaler()

# standardization
scale=object.fit_transform(x)
print(scale)
```

```
[[-0.78321342]
 [-0.60653412]
 [-0.99588476]
 ...
 [-1.47928179]
 [-0.11935577]
 [-0.87055909]]
```

10.Split the data into training and testing

In [ ]:

```
from sklearn.model_selection import train_test_split

# split the dataset
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.05, random_state=0
)
```

In [ ]:

```
X_train
```

Out[ ]:

```
array([[800, 15567367, 'Tao', ..., 0, 1, 103315.74],
       [1070, 15628674, 'Iadanza', ..., 1, 0, 31904.31],
       [8411, 15609913, 'Clark', ..., 1, 0, 113436.08],
       ...,
       [3265, 15574372, 'Hoolan', ..., 1, 0, 181429.87],
       [9846, 15664035, 'Parsons', ..., 1, 1, 148750.16],
       [2733, 15592816, 'Udokamma', ..., 1, 0, 118855.26]], dtype=object)
```

In [ ]:

```
Y_train
```

Out[ ]:

```
array([0, 1, 0, ..., 0, 0, 1])
```

In [ ]:

In [ ]:

```
X_test
```

Out[ ]:

```
array([[9395, 15615753, 'Upchurch', ..., 1, 1, 192852.67],
       [899, 15654700, 'Fallaci', ..., 1, 0, 128702.1],
       [2399, 15633877, 'Morrison', ..., 1, 1, 75732.25],
       ...,
       [492, 15699005, 'Martin', ..., 1, 1, 9983.88],
       [2022, 15795519, 'Vasiliev', ..., 0, 0, 197322.13],
       [4300, 15711991, 'Chiawuotu', ..., 0, 0, 3183.15]], dtype=object)
```

In [ ]:

```
Y_test
```

Out[ ]:

```
array([0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0])
```