1. Downloading the Dataset and importing the Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn as sk
```

2. Load the dataset

In [2]:
```
data = pd.read_csv("Churn_Modelling.csv")
```

In [3]:
```
pwd
```

Out[3]: 'C:\\Users\\amirt\\Desktop\\IBM Assignments\\Week-2 Assignment\\Team Lead'

In [4]:
```
#display first five rows

data.head()
```

Out[4]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	1	15634602	Hargrave	619	France	Female	42	2	(
1	2	15647311	Hill	608	Spain	Female	41	1	8380
2	3	15619304	Onio	502	France	Female	42	8	15966
3	4	15701354	Boni	699	France	Female	39	1	(
4	5	15737888	Mitchell	850	Spain	Female	43	2	12551

In [5]:
```
#display first 10 rows

data.head(10)
```

Out[5]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	1	15634602	Hargrave	619	France	Female	42	2	(
1	2	15647311	Hill	608	Spain	Female	41	1	8380
2	3	15619304	Onio	502	France	Female	42	8	15966
3	4	15701354	Boni	699	France	Female	39	1	(
4	5	15737888	Mitchell	850	Spain	Female	43	2	12551
5	6	15574012	Chu	645	Spain	Male	44	8	11375
6	7	15592531	Bartlett	822	France	Male	50	7	(
7	8	15656148	Obinna	376	Germany	Female	29	4	11504
8	9	15792365	He	501	France	Male	44	4	14205
9	10	15592389	H?	684	France	Male	27	2	13460

In [14]:
```
data.columns
```

Out[14]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [15]:
```
#unique feature - gender

data["Geography"].unique()
```

Out[15]:
```
array(['France', 'Spain', 'Germany'], dtype=object)
```

In [39]:
```
data.nunique()
```

Out[39]:
```
RowNumber          10000
CustomerId         10000
Surname             2932
CreditScore          460
Geography              3
Gender                 2
Age                   70
Tenure                11
Balance             6382
NumOfProducts          4
HasCrCard              2
IsActiveMember         2
EstimatedSalary     9999
Exited                 2
dtype: int64
```

In [45]:
```
data["IsActiveMember"].unique()
```

Out[45]:
```
array([1, 0], dtype=int64)
```

In [16]:
```
# To display the bottom of the dataset

data.tail()
```

Out[16]:

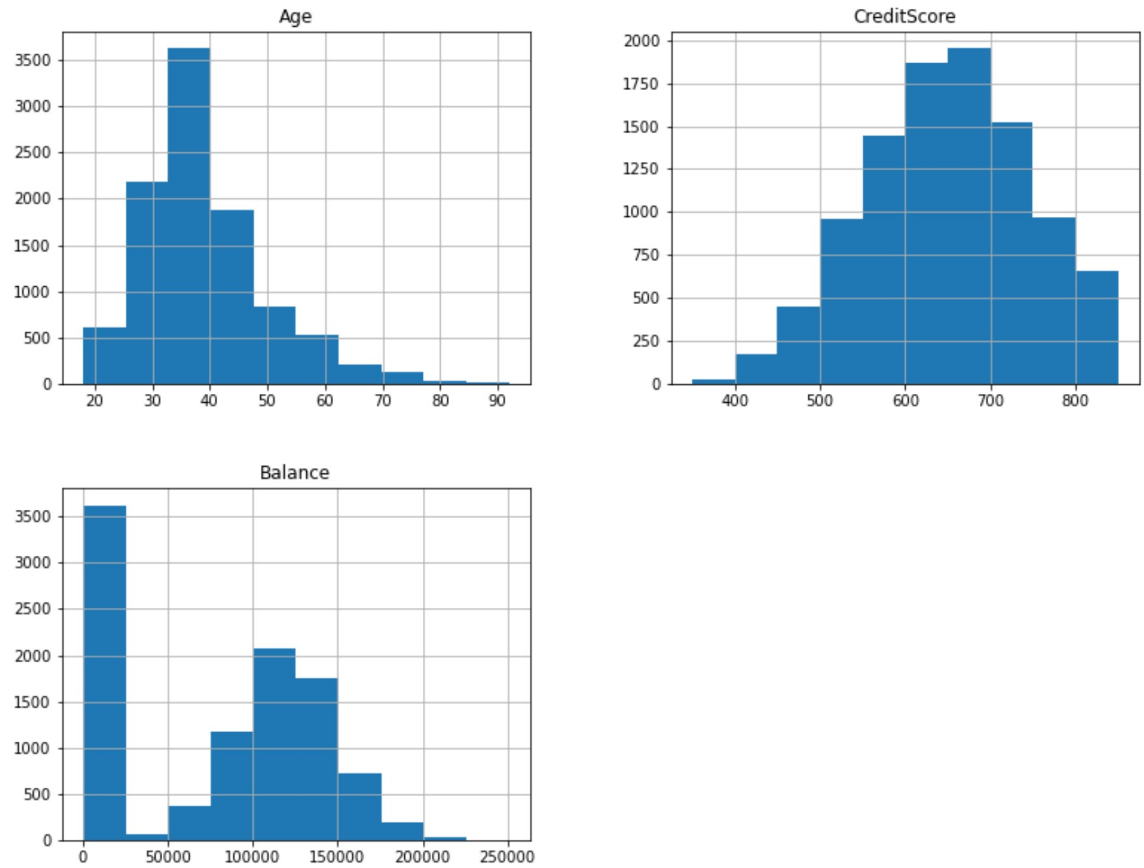
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
9995	9996	15606229	Obijiaku	771	France	Male	39	5
9996	9997	15569892	Johnstone	516	France	Male	35	10
9997	9998	15584532	Liu	709	France	Female	36	7
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3
9999	10000	15628319	Walker	792	France	Female	28	4

3. Performing (EDA) Visulization

- Univariate Analysis
- Bi - Variate Analysis
- Multi - Variate Analysis

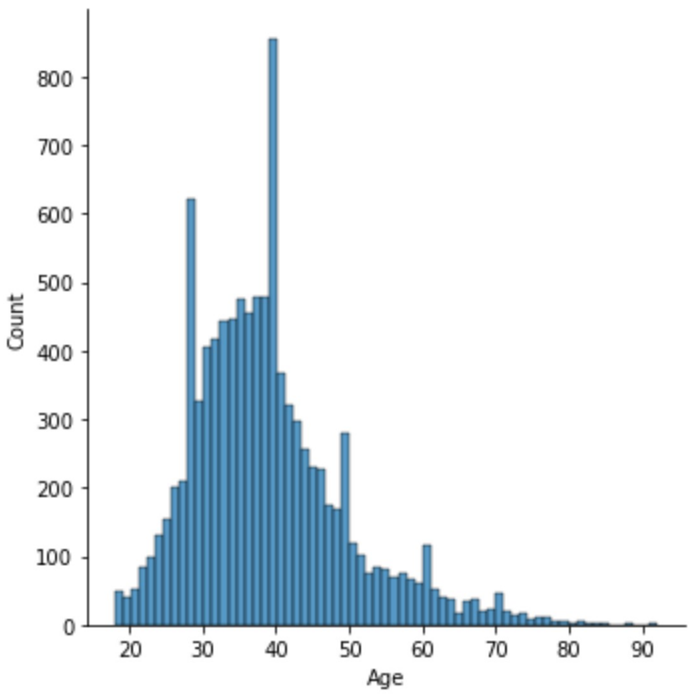
(i) Uni-variate Analysis

In [7]:
```
features =['Age','CreditScore', 'Balance']
data[features].hist(figsize=(13, 10));
```



In [8]:
```
sns.displot(data["Age"])
```

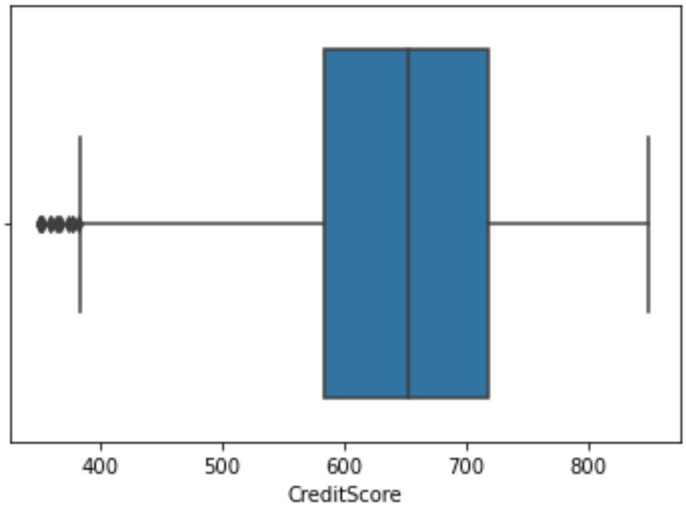
Out[8]: <seaborn.axisgrid.FacetGrid at 0x27b7b849700>



In [47]:
```
import warnings
warnings.filterwarnings("ignore")
```

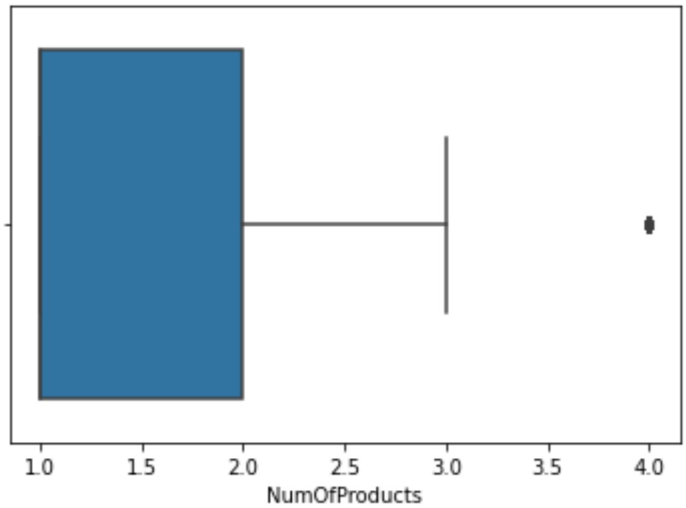
In [48]:
```
sns.boxplot(data["CreditScore"])
```

Out[48]: <AxesSubplot:xlabel='CreditScore'>



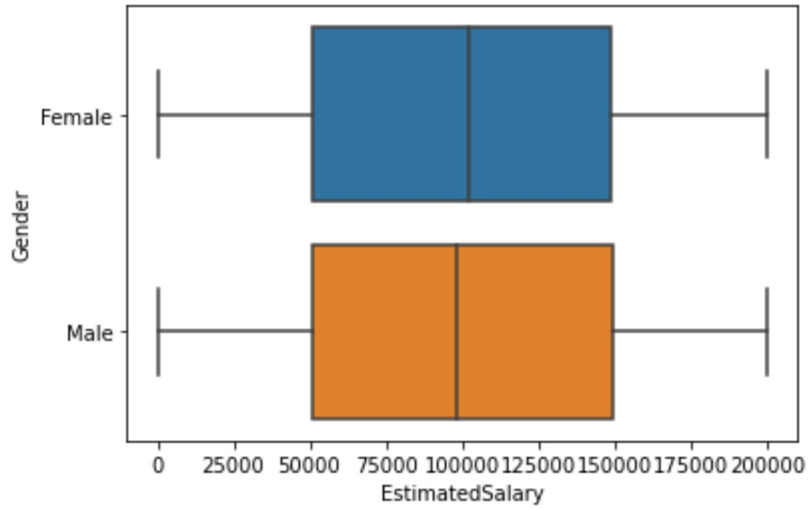
In [52]:
```
sns.boxplot(data["NumOfProducts"])
```

Out[52]: <AxesSubplot:xlabel='NumOfProducts'>

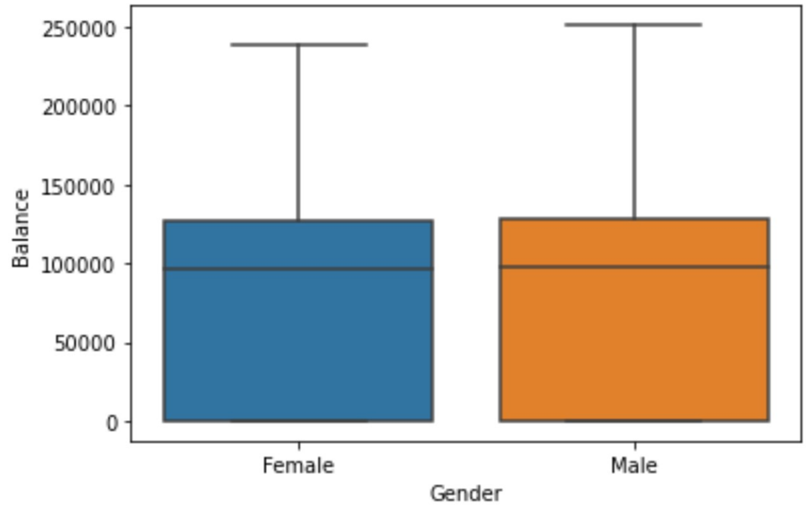


(ii) Bi-variate Analysis

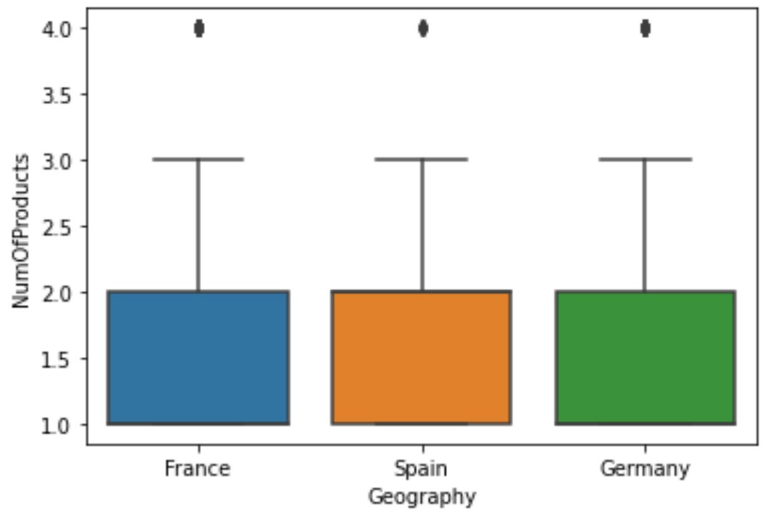
In [9]:
```
sns.boxplot(x = data['EstimatedSalary'], y = data['Gender'] );
```



In [13]:
```
sns.boxplot(x=data['Gender'],y=data['Balance']);
```



In [18]:
```
sns.boxplot(x=data['Geography'],y=data['NumOfProducts']);
```



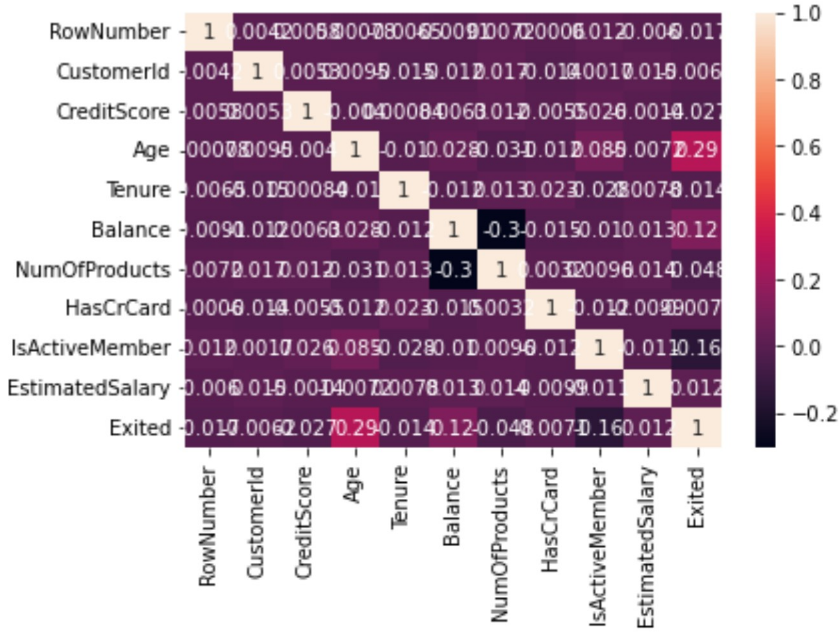
(iii) Multi-variate Analysis

In [20]:
```
# Correlation for "NumOfProducts","EstimatedSalary","Balance"

df= pd.DataFrame(data,columns=['NumOfProducts','EstimatedSalary','Balance']
corrMatrix = df.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()
```

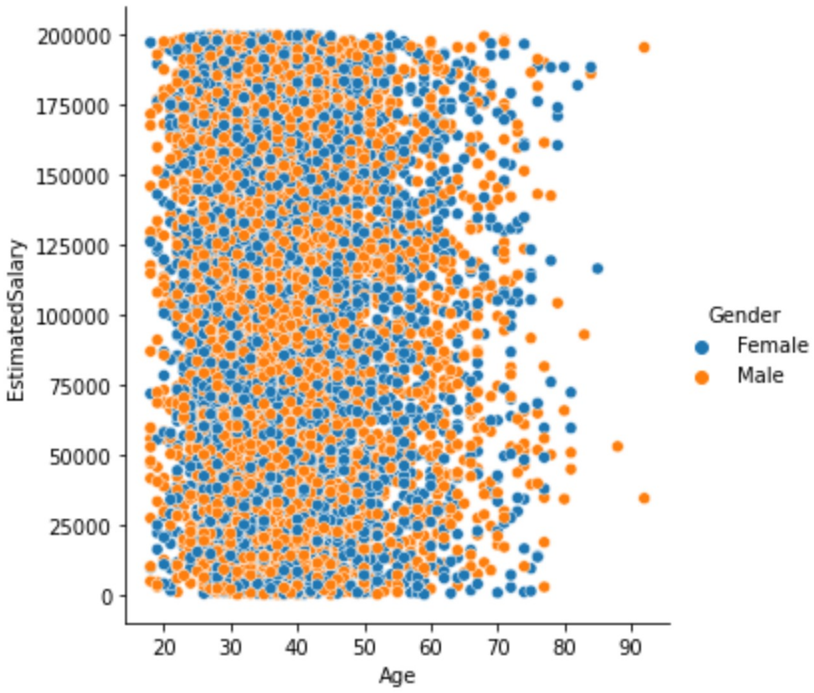
In [21]:
```
#correlation for all elements in dataset
sns.heatmap(data.corr(),annot = True)
```

Out[21]: <AxesSubplot:>



In [28]:
```
sns.relplot(x = "Age",y ="EstimatedSalary",hue="Gender",data=data)
```

Out[28]:
```
<seaborn.axisgrid.FacetGrid at 0x27b0df3a520>
```



4. Performing descriptive statistics on the dataset.

In [150]:
```
data[['CreditScore','Balance','EstimatedSalary']].mean()
```

Out[150]:
```
CreditScore           650.561300
Balance             76485.889288
EstimatedSalary    100090.239881
dtype: float64
```

In [151]:
```
#median

data[['CreditScore','Balance','EstimatedSalary']].median()
```

Out[151]:
```
CreditScore           652.000
Balance             97198.540
EstimatedSalary    100193.915
dtype: float64
```

In [152]:
```
#mode

data[['CreditScore','Balance','EstimatedSalary']].mode()
```

Out[152]:

	CreditScore	Balance	EstimatedSalary
0	850.0	0.0	24924.92

In [153]:
```
#quantile

data[['CreditScore','Balance','EstimatedSalary']].quantile()
```

Out[153]:
```
CreditScore           652.000
Balance             97198.540
EstimatedSalary    100193.915
Name: 0.5, dtype: float64
```

In [154]:
```
#standard Deivation

data[['CreditScore','Balance','EstimatedSalary']].std()
```

Out[154]:
```
CreditScore           96.558702
Balance            62397.405202
EstimatedSalary    57510.492818
dtype: float64
```

In [155]:
```
#min

data[['CreditScore','Balance','EstimatedSalary']].min()
```

Out[155]:
```
CreditScore        383.00
Balance              0.00
EstimatedSalary     11.58
dtype: float64
```

In [156]:
```
#max

data[['CreditScore','Balance','EstimatedSalary']].max()
```

Out[156]:
```
CreditScore           850.00
Balance            250898.09
EstimatedSalary    199992.48
dtype: float64
```

In [157]:
```
#skew

data[['CreditScore','Balance','EstimatedSalary']].skew()
```

Out[157]:
```
CreditScore       -0.064255
Balance           -0.141109
EstimatedSalary    0.002085
dtype: float64
```

In [26]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
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
```

In [158]:
```
data.shape
```

Out[158]: (10000, 14)

In [31]:
```
data.describe()
```

Out[31]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000

5. Handling the Missing values.

In [34]:
```
data.isnull().sum()
```

Out[34]:
```
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
```

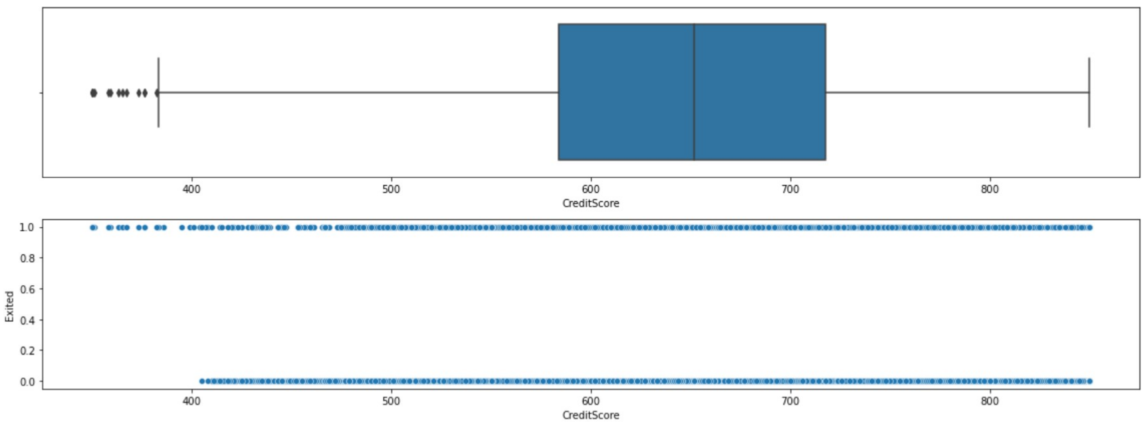
The above result shows that there is no missing values in the dataset

6. Find the outliers and replace the outliers

In [55]:
```
def box_scatter(data, x, y):
    fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(16,6))
    sns.boxplot(data=data, x=x, ax=ax1)
    sns.scatterplot(data=data, x=x,y=y,ax=ax2)
```

In [58]:
```
#Scatter and box plot
box_scatter(data,'CreditScore','Exited');
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(data.loc[data['CreditScore'] < 400])}
```

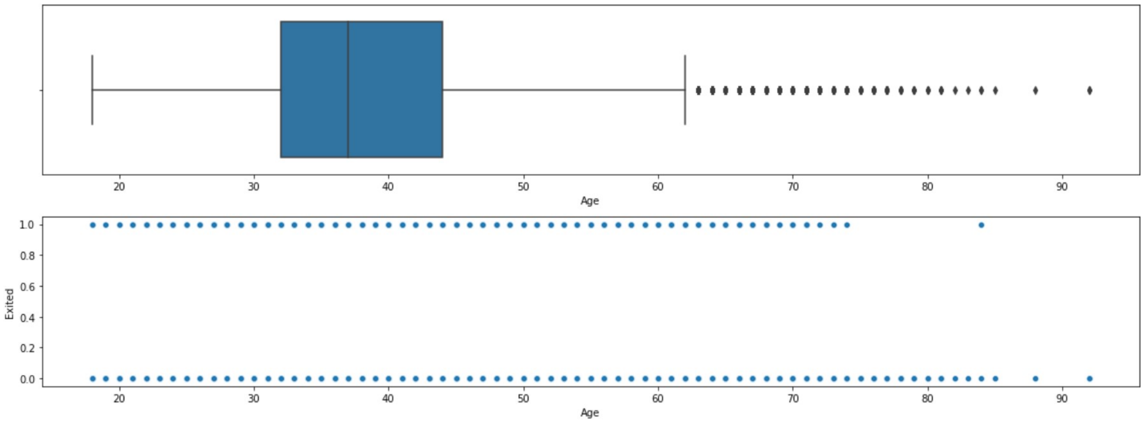
```
# of Bivariate Outliers: 19
```



In the analysis view,there are 19 outliers

In [60]:
```
box_scatter(data,'Age','Exited');
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(data.loc[data['Age'] > 87])}")
```

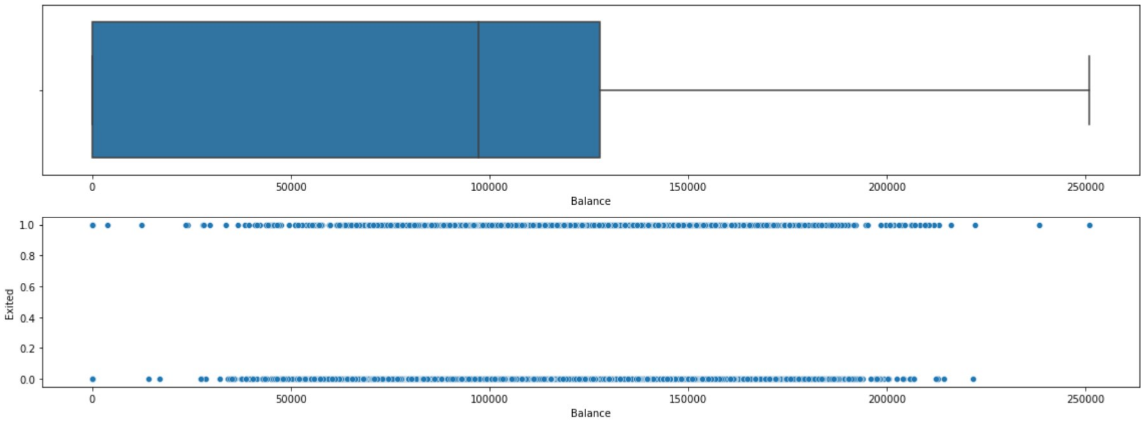
```
# of Bivariate Outliers: 3
```



In the above,there are 3 outliers

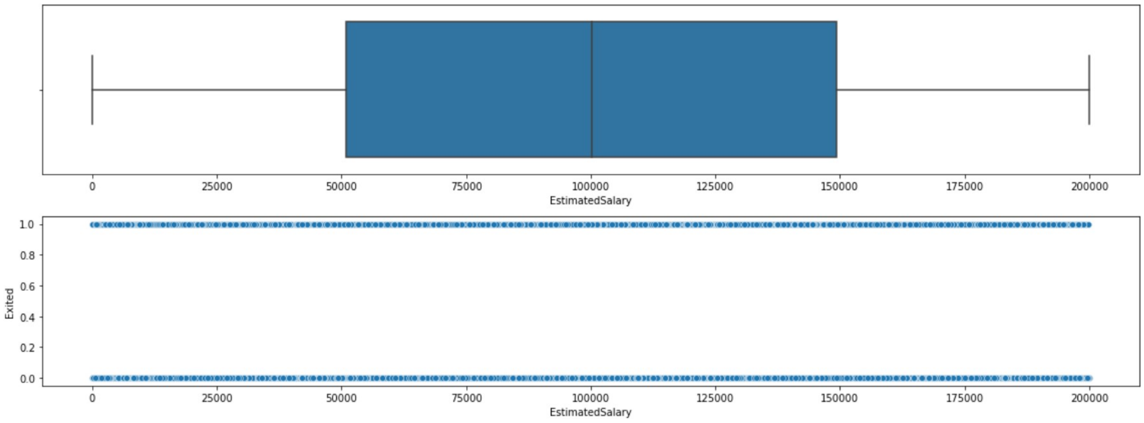
In [61]:
```
box_scatter(data,'Balance','Exited');
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(data.loc[data['Balance'] > 220000])}"
```

```
# of Bivariate Outliers: 4
```



Again ,there are 4 outliers

In [62]:
```
box_scatter(data,'EstimatedSalary','Exited');
plt.tight_layout()
```

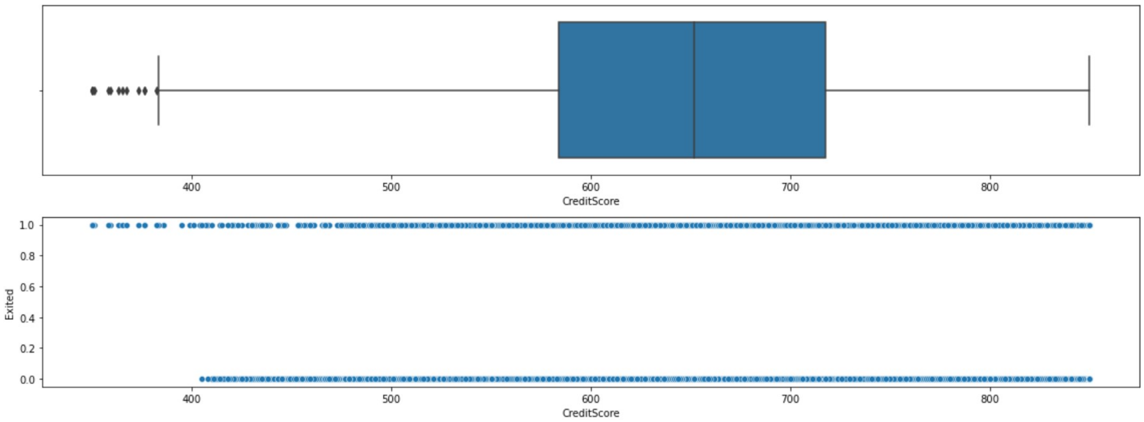


Removing the Outliers

In [63]:
```
for i in df:
    if df[i].dtype=='int64' or df[i].dtypes=='float64':
        q1=df[i].quantile(0.25)
        q3=df[i].quantile(0.75)
        iqr=q3-q1
        upper=q3+1.5*iqr
        lower=q1-1.5*iqr
        df[i]=np.where(df[i] >upper, upper, df[i])
        df[i]=np.where(df[i] <lower, lower, df[i])
```

In [65]:
```
box_scatter(data,'CreditScore','Exited');
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(data.loc[data['CreditScore'] < 400])}")
```

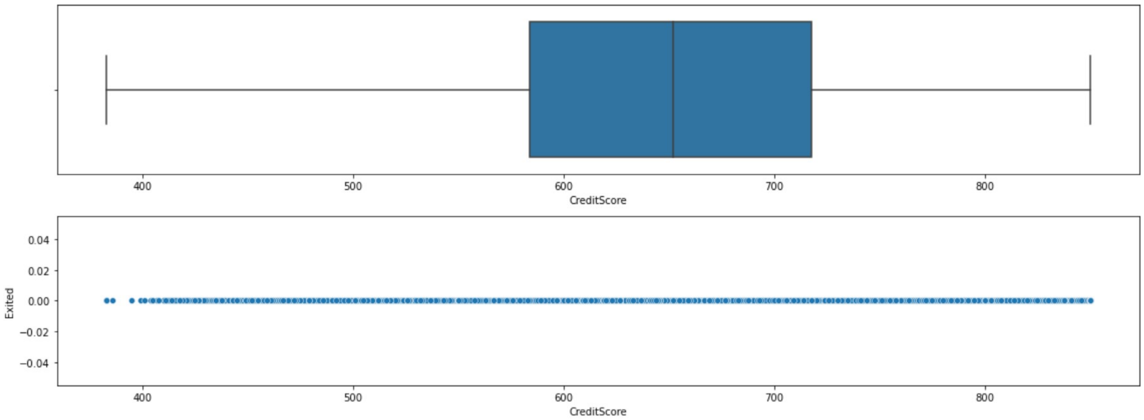
```
# of Bivariate Outliers: 19
```



In [68]:
```
for i in data:
    if data[i].dtype=='int64' or data[i].dtypes=='float64':
        q1=data[i].quantile(0.25)
        q3=data[i].quantile(0.75)
        iqr=q3-q1
        upper=q3+1.5*iqr
        lower=q1-1.5*iqr
        data[i]=np.where(data[i] >upper, upper, data[i])
        data[i]=np.where(data[i] <lower, lower, data[i])
```

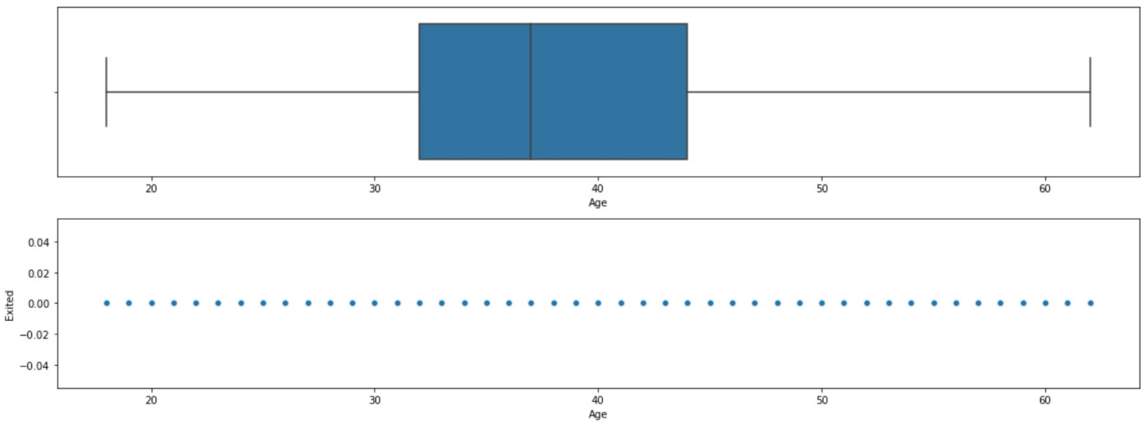
In [69]:
```
box_scatter(data,'CreditScore','Exited');
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(data.loc[data['CreditScore'] < 400])}
```

```
# of Bivariate Outliers: 19
```



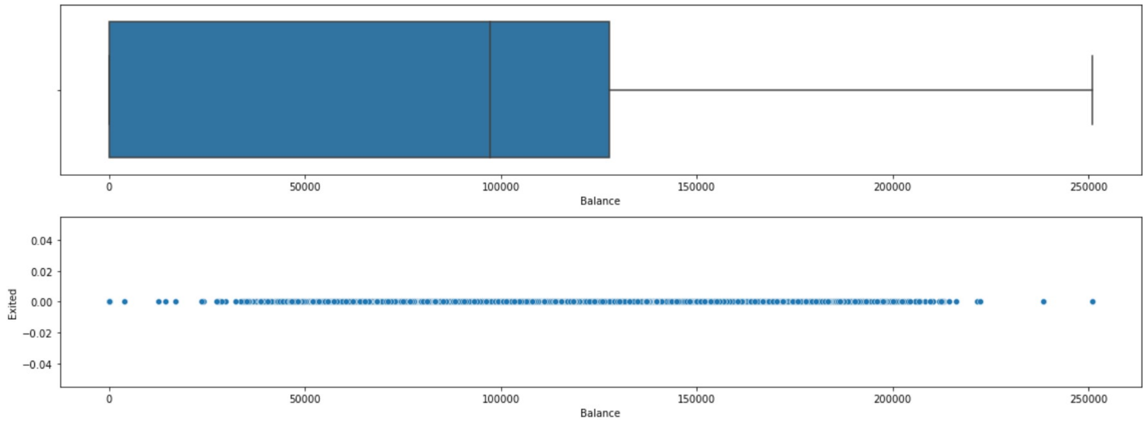
In [70]:
```
box_scatter(data,'Age','Exited');
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(data.loc[data['Age'] > 87])}")
```

```
# of Bivariate Outliers: 0
```



In [71]:
```
box_scatter(data,'Balance','Exited');
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(data.loc[data['Balance'] > 220000])}")
```

```
# of Bivariate Outliers: 4
```



7. Checking for Categorical columns and performing encoding.

In [77]:
```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
for i in data:
    if data[i].dtype=='object' or data[i].dtype=='category':
        data[i]=encoder.fit_transform(data[i])
```

8. Split the data into dependent and independent variables

In [119]:
```
x = data.iloc[:,:-1]
x.head()
```

Out[119]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	1.0	15634602.0	1115	619.0	0	0	42.0	2.0	
1	2.0	15647311.0	1177	608.0	2	0	41.0	1.0	8380
2	3.0	15619304.0	2040	502.0	0	0	42.0	8.0	15966
3	4.0	15701354.0	289	699.0	0	0	39.0	1.0	
4	5.0	15737888.0	1822	850.0	2	0	43.0	2.0	12551

In [120]:
```
y=data.iloc[:-1]
y.head()
```

Out[120]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	1.0	15634602.0	1115	619.0	0	0	42.0	2.0	
1	2.0	15647311.0	1177	608.0	2	0	41.0	1.0	8380
2	3.0	15619304.0	2040	502.0	0	0	42.0	8.0	15966
3	4.0	15701354.0	289	699.0	0	0	39.0	1.0	
4	5.0	15737888.0	1822	850.0	2	0	43.0	2.0	12551

9. Scaling the independent variables

In [144]:
```
#scaling
names=X.columns
names
```

Out[144]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'Geography', 'Gender', 'Age',
       'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember
',
       'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [145]:
```
from sklearn.preprocessing import scale
x= scale(X)
x
```

Out[145]:
```
array([[-1.73187761, -0.78321342, -0.46418322, ...,  0.97024255,
         0.02188649,  0.        ],
       [-1.7315312 , -0.60653412, -0.3909112 , ...,  0.97024255,
         0.21653375,  0.        ],
       [-1.73118479, -0.99588476,  0.62898807, ..., -1.03067011,
         0.2406869 ,  0.        ],
       ...,
       [ 1.73118479, -1.47928179,  0.07353887, ...,  0.97024255,
        -1.00864308,  0.        ],
       [ 1.7315312 , -0.11935577,  0.98943914, ..., -1.03067011,
        -0.12523071,  0.        ],
       [ 1.73187761, -0.87055909,  1.4692527 , ..., -1.03067011,
        -1.07636976,  0.        ]])
```

In [146]:
```
X = pd.DataFrame(x,columns = names)
X
```

Out[146]:

	RowNumber	CustomerId	Surname	Geography	Gender	Age	Tenure	Bala
0	-1.731878	-0.783213	-0.464183	-0.901886	-1.095988	0.342615	-1.041760	-1.225
1	-1.731531	-0.606534	-0.390911	1.515067	-1.095988	0.240011	-1.387538	0.117
2	-1.731185	-0.995885	0.628988	-0.901886	-1.095988	0.342615	1.032908	1.333
3	-1.730838	0.144767	-1.440356	-0.901886	-1.095988	0.034803	-1.387538	-1.225
4	-1.730492	0.652659	0.371354	1.515067	-1.095988	0.445219	-1.041760	0.785
...	...	...	...	...	...	...	...	
9995	1.730492	-1.177652	0.580534	-0.901886	0.912419	0.034803	-0.004426	-1.225
9996	1.730838	-1.682806	-0.203004	-0.901886	0.912419	-0.375612	1.724464	-0.306
9997	1.731185	-1.479282	0.073539	-0.901886	-1.095988	-0.273008	0.687130	-1.225
9998	1.731531	-0.119356	0.989439	0.306591	0.912419	0.342615	-0.695982	-0.022
9999	1.731878	-0.870559	1.469253	-0.901886	-1.095988	-1.093840	-0.350204	0.859

10000 rows × 13 columns

10. Splitting the data into Training and Testing

In [125]:
```
from sklearn.model_selection import train_test_split
```

In [147]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_s
```

In [148]:
```
X_train.head()
```

Out[148]:

	RowNumber	CustomerId	Surname	Geography	Gender	Age	Tenure	Bala
7389	0.827747	-0.195066	0.366627	1.515067	-1.095988	-0.478216	-0.004426	-1.225
9275	1.481077	0.810821	-1.292630	0.306591	0.912419	0.342615	-1.387538	-0.012
2995	-0.694379	-1.507642	0.391445	-0.901886	-1.095988	-0.991236	-1.041760	0.575
5316	0.109639	1.243462	-0.744271	1.515067	0.912419	0.137407	-0.004426	0.467
356	-1.608556	-1.100775	1.117074	1.515067	-1.095988	1.881674	1.032908	0.806

In [149]:
```
X_train.shape,y_train.shape,X_test.shape,y_test.shape
```

Out[149]: ((8000, 13), (8000,), (2000, 13), (2000,))