1.Importing the headers

In [ ]:
```
import io
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statistics as st
import sklearn
```

In [ ]:
```
```

2.Downloading and loading the churn_modelling dataset

In [ ]:
```
from google.colab import files
uploaded = files.upload()
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Churn_Modelling.csv to Churn_Modelling.csv
```

In [71]:
```
dataset = pd.read_csv(io.BytesIO(uploaded['Churn_Modelling.csv']))
```

In [72]:
```
df = pd.read_csv('Churn_Modelling.csv')
df.head()
```

Out[72]:

	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 [73]:
```
df = df.drop(columns=['RowNumber', 'CustomerId', 'Surname'])
df.head()
```

Out[73]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

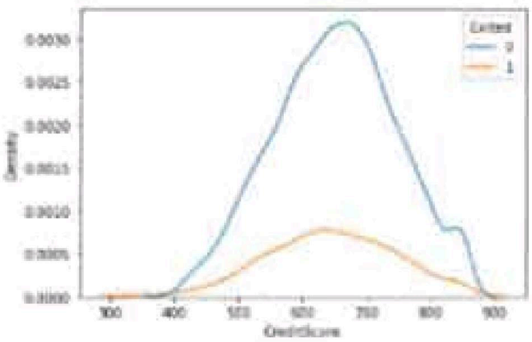
In [74]:

```
df['IsActiveMember'] = df['IsActiveMember'].astype('category')
df['Exited'] = df['Exited'].astype('category')
df['HasCrCard'] = df['HasCrCard'].astype('category')
```

3a.Univariate Analysis

In [75]:

```
sns.kdeplot(x='CreditScore', data = df , hue = 'Exited')
plt.show()
```



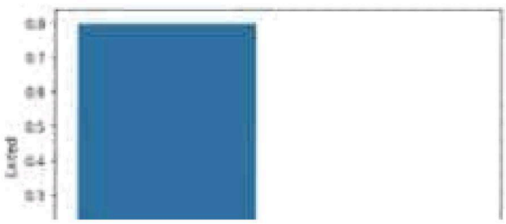
3b.Bivariate Analysis

In [76]:

```
density = df['Exited'].value_counts(normalize=True).reset_index()
sns.barplot(data=density, x='index', y='Exited', )
density
```

Out[76]:

	index	Exited
0	0	0.7963
1	1	0.2037



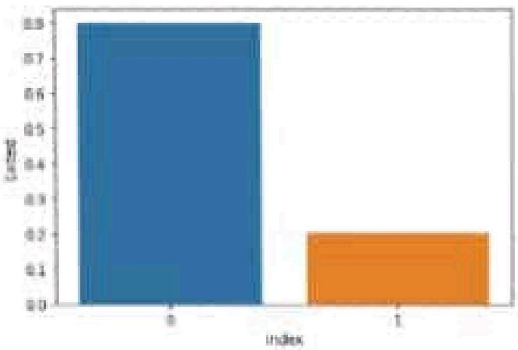
3b.Bivariate Analysis

In [76]:

```
density = df['Exited'].value_counts(normalize=True).reset_index()
sns.barplot(data=density, x='index', y='Exited', );
density
```

Out[76]:

	index	Exited
0	0	0.7963
1	1	0.2037

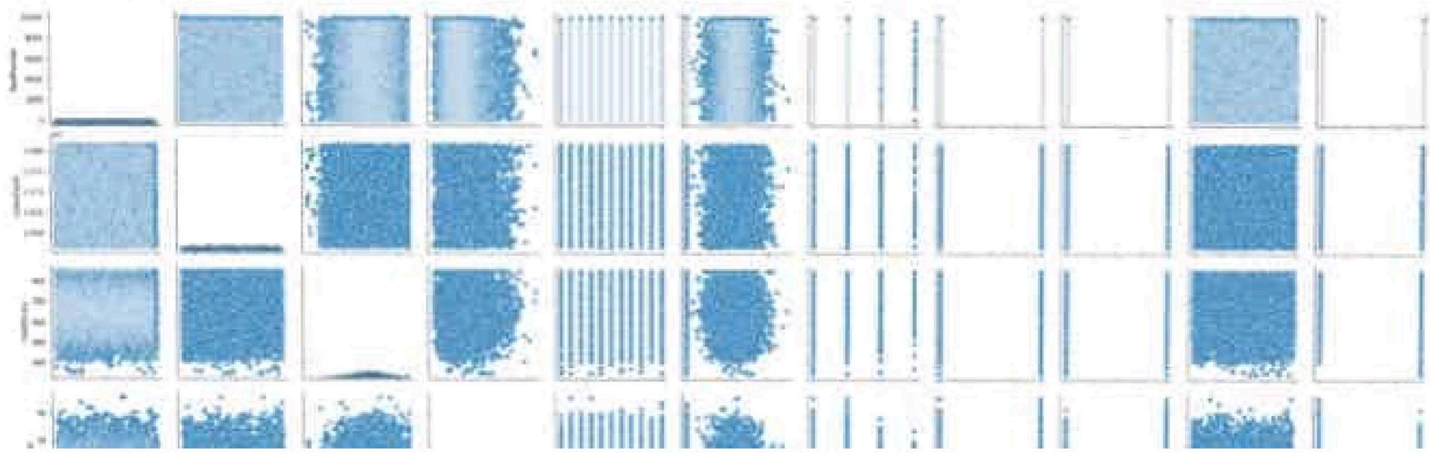


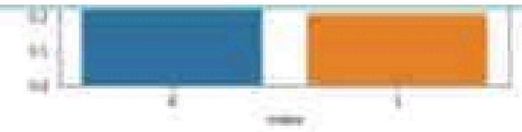
3c.Multivariate Analysis

In [77]:

```
sns.pairplot(dataset)
```

Out[77]: <seaborn.axisgrid.PairGrid at 0x7f3968c57f90>

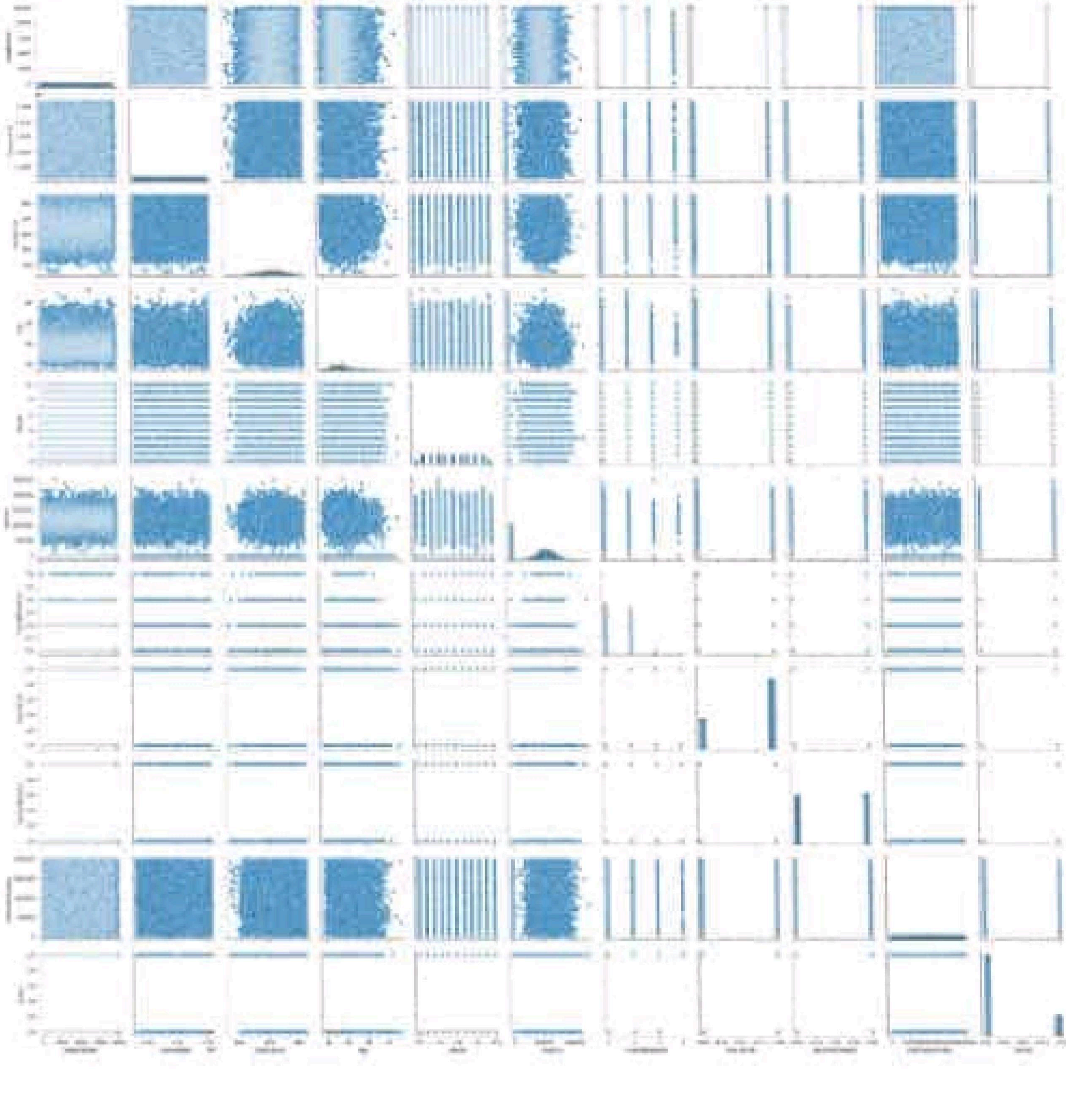




3c.Multivariate Analysis

```
sns.pairplot(dataset)
```

```
<seaborn.axisgrid.PairGrid at 0x[illegible]>
```



4.Discriptive Statistics bold text

In [78]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   CreditScore      10000 non-null  int64
 1   Geography        10000 non-null  object
 2   Gender           10000 non-null  object
 3   Age              10000 non-null  int64
 4   Tenure           10000 non-null  int64
 5   Balance          10000 non-null  float64
 6   NumOfProducts    10000 non-null  int64
 7   HasCrCard        10000 non-null  category
 8   IsActiveMember   10000 non-null  category
 9   EstimatedSalary  10000 non-null  float64
 10  Exited           10000 non-null  category
dtypes: category(3), float64(2), int64(4), object(2)
memory usage: 654.8+ KB
```

In [79]:
```
df.describe()
```

Out[79]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	650.528800	38.921800	5.012800	76485.889288	1.530200	100090.239881
std	96.653299	10.487806	2.892174	62397.405202	0.581654	57510.492818
min	350.000000	18.000000	0.000000	0.000000	1.000000	11.580000
25%	584.000000	32.000000	3.000000	0.000000	1.000000	51002.110000
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	100193.915000
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	149388.247500
max	850.000000	92.000000	10.000000	250898.090000	4.000000	199992.480000

5.Haldle Missing Values

In [80]:
```
df.isna().sum()
```

Out[80]:
```
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
```

6.Find the Outliers and Replace the Outlier

Finding outliers

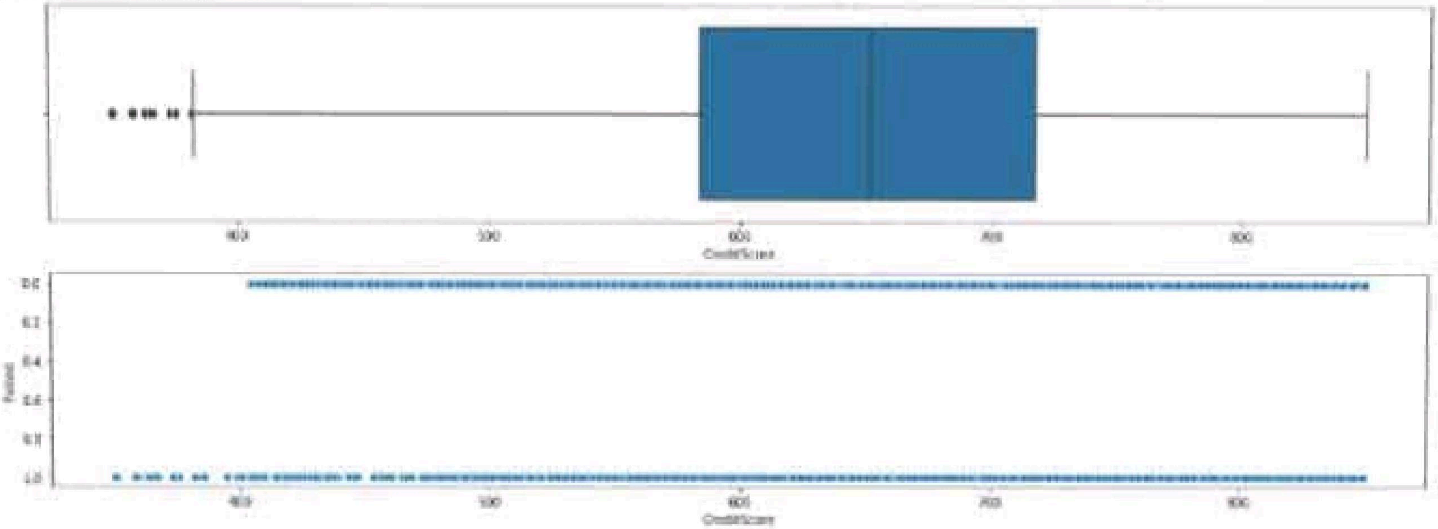
In [82]:

```
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 [83]:

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

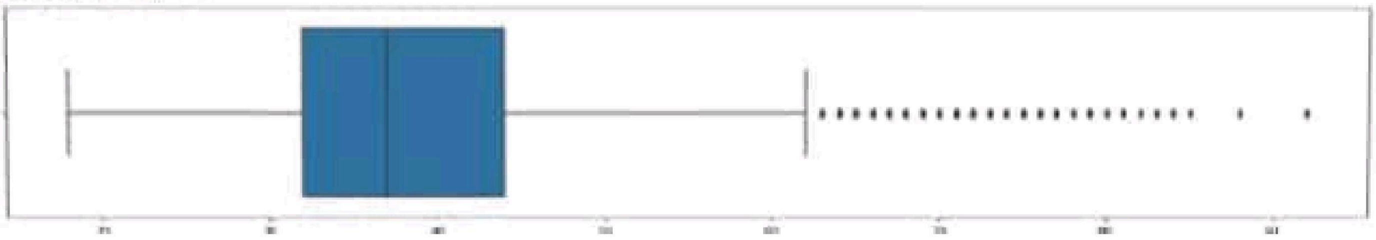
```
# of Bivariate Outliers: 19
```

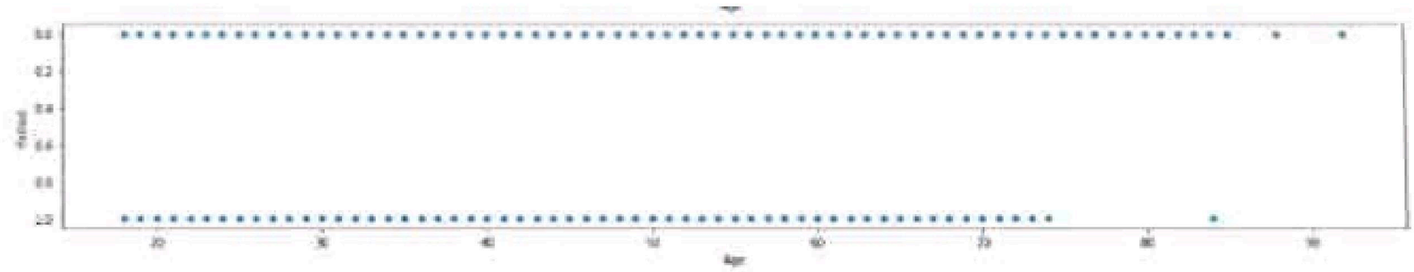


In [84]:

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

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

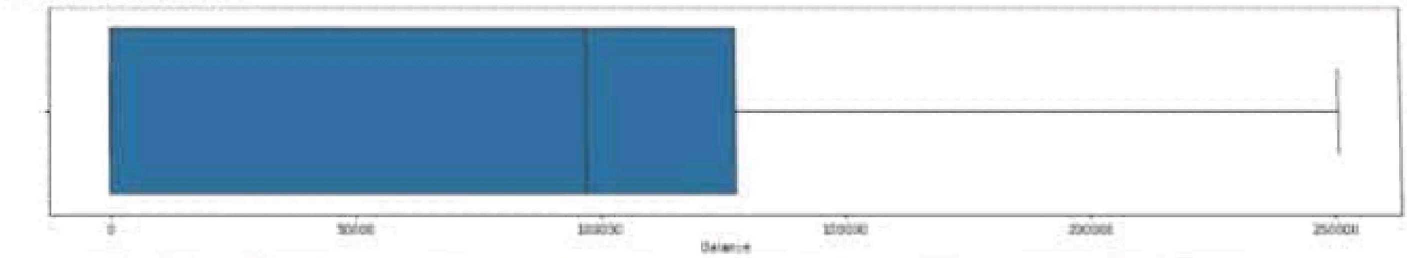


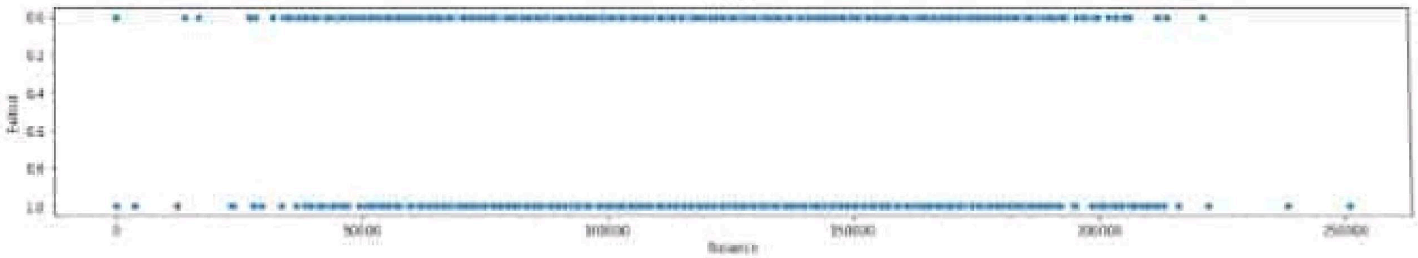


In [84]:

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

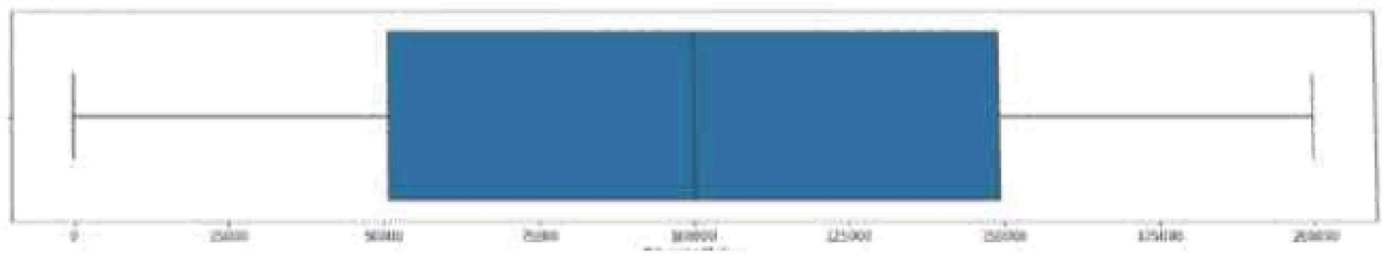
```
# of Bivariate Outliers: 4
```

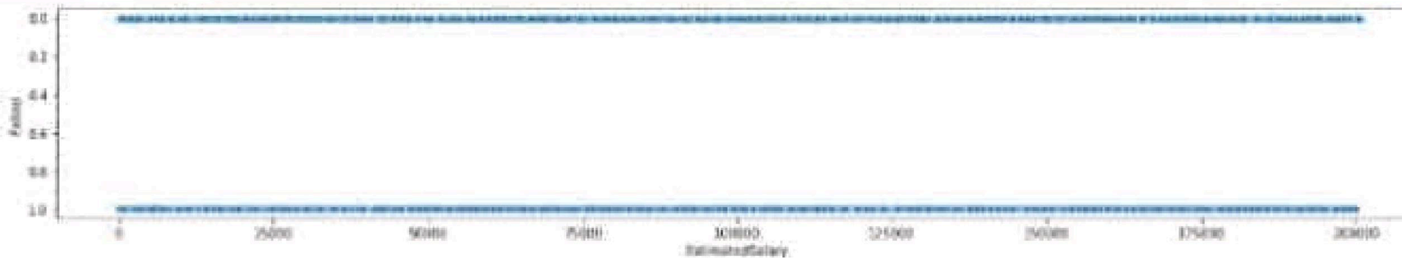




In [85]:

```
box_scatter(df,'EstimatedSalary','Exited');
plt.tight_layout()
```





Removing the Outliers

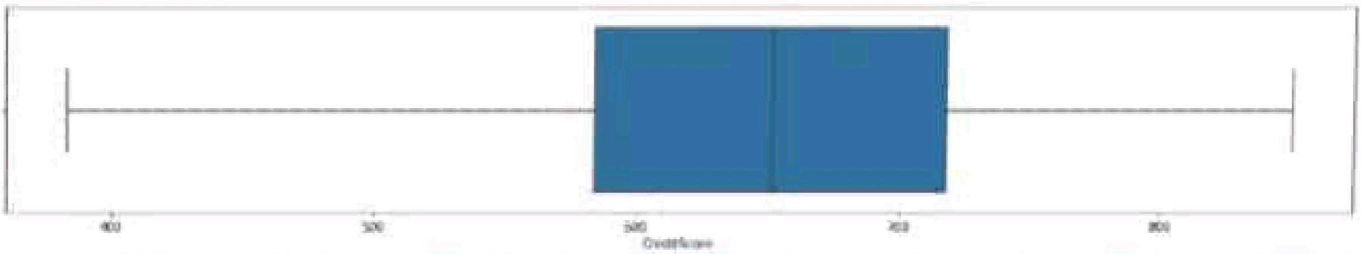
In [86]:

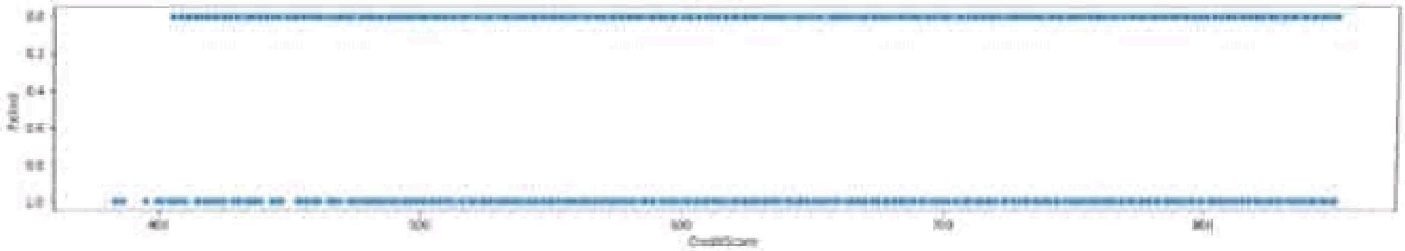
```python
for i in df:
    if df[i].dtype=='int64' or df[i].dtypes=='float64':
        q1=df[i].quantile(0.25)
        q3=df[i].quantile(0.75)
        iqr=q3-q1
        upper=q3+1.5*iqr
        lower=q1-1.5*iqr
        df[i]=np.where(df[i] >upper, upper, df[i])
        df[i]=np.where(df[i] <lower, lower,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

In [87]:

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

```
# of Bivariate Outliers: 18
```

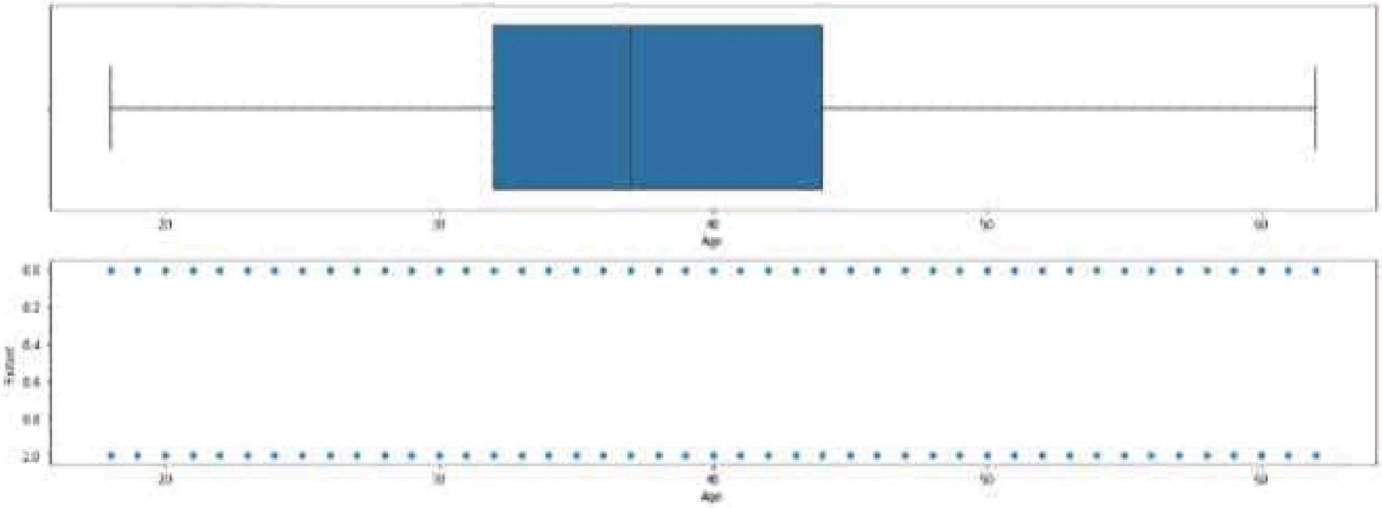




In [88]:

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

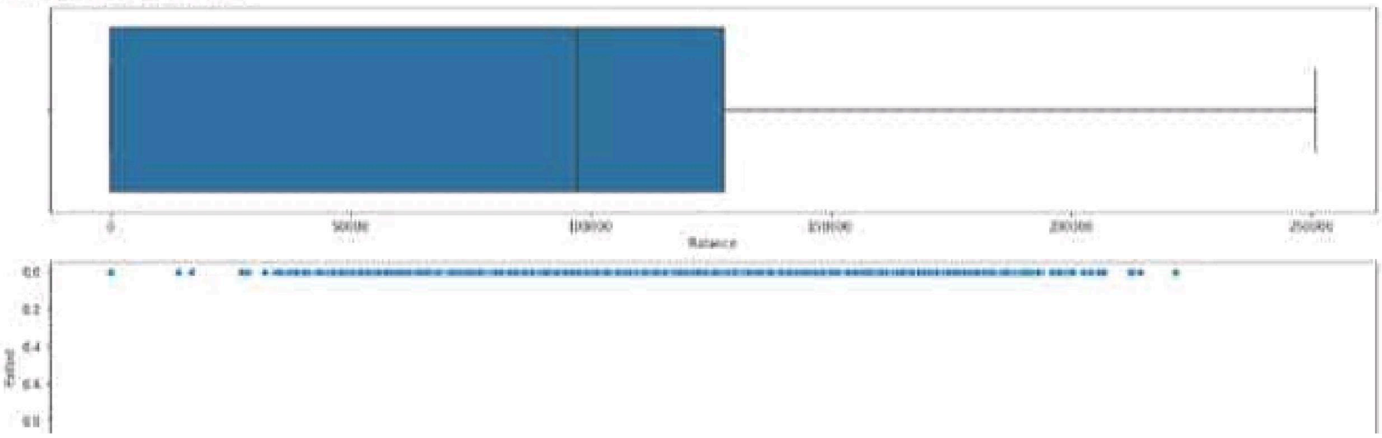
```
# of Bivariate Outliers: 0
```



In [89]:

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

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



M-EPBL/IBM-Project-32294-1660209123/blob/main/Karan%20assignment_2_.ipynb

Balance

7.Categorical Encoding

In [16]:

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

8.Split the data into dependent and indipendent variables

In [17]:

```
X = dataset
X.drop(['RowNumber', 'Exited'], axis=1)

Y = pd.DataFrame(dataset['Exited'])
```

In [108...

```
print (X.columns)
print (Y.columns)
```

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

9.Scale and independent variables

In [109...

```
print (Y)
```

```
      Exited
0          1
1          0
2          1
3          0
4          0
...      ...
9995       0
9996       0
9997       1
9998       1
9999       0

[10000 rows x 1 columns]
```

In [112...

```
from sklearn import preprocessing

normalized_Y = preprocessing.normalize(Y)

print (normalized_Y)
```

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

In [103]:

```python
standard_Y = Y.copy()

from sklearn import preprocessing

ss = preprocessing.StandardScaler()
ss.fit(standard_Y)

print (standard_Y)
```

```
      Exited
0          1
1          0
2          1
3          0
4          0
...      ...
9995       0
9996       0
9997       1
9998       1
9999       0

[10000 rows x 1 columns]
```

10.Perform Train Test Split

In [104]:

```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split (X, Y)
```

In [105]:

```python
x_train
```

Out[105]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
6811	6812	15771222	Oguejiofor	779	France	Female	42	5	0.00	2	0	0	25951.91	0
2974	2975	15746731	Barnes	583	Germany	Male	58	8	69128.58	1	1	1	170705.53	0
952	953	15600149	Wood	571	Germany	Female	66	8	111577.01	1	0	1	188271.90	0
5643	5644	15782096	Volkova	616	Spain	Female	36	6	0.00	1	1	1	12916.32	1
7061	7062	15789201	Thomson	603	Germany	Female	35	9	145623.36	1	1	0	163181.62	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4653	4654	15579017	Sinclair	489	France	Female	51	3	0.00	2	0	1	174098.28	1
1317	1318	15720702	Shih	789	France	Male	37	3	0.00	1	1	0	121883.87	1

In [186...

```
x_test
```

Out[186...

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exite
7220	7221	15700637	Chang	718	Spain	Male	40	5	0.00	2	0	0	121527.81	
1154	1155	15766887	Iadanza	538	Spain	Male	39	2	122773.50	2	1	1	58467.08	
3156	3157	15509641	Sung	692	Germany	Female	41	8	135701.29	1	1	0	59354.24	
4514	4515	15658670	Chim	669	France	Female	38	8	0.00	2	1	0	84048.16	
2017	2018	15768244	Macleod	538	Spain	Female	30	8	0.00	2	1	1	41192.95	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3443	3444	15606755	Moretti	597	Spain	Female	46	4	0.00	2	1	0	58607.16	
7123	7124	15682686	Chukwuemeka	722	France	Female	38	3	0.00	2	0	1	167984.72	
1997	1998	15624781	Mbanefo	672	France	Female	34	1	142151.75	2	1	1	108753.34	
5735	5736	15756076	Greenwood	585	Spain	Female	44	4	0.00	2	0	1	101728.46	
4348	4349	15574387	Ni	635	Germany	Female	44	2	79064.85	2	0	1	113291.75	

2500 rows × 14 columns

In [187...

```
y_train
```

Out[187...

	Exited
6011	0
2974	0
952	0
5643	1
7061	0
...	...
4653	1
1317	1
3661	0
1002	0
5961	0

7500 rows × 1 columns

In [188...

```
y_test
```

952	0
5643	1
7061	0
...	...
4653	1
1317	1
3661	0
1002	0
5961	0

7500 rows × 1 columns

In [[illegible]]:

```
y_test
```

Out[[illegible]]:

	Exited
7220	0
1154	0
3156	1
4514	0
2017	0
...	...
3443	1
7123	0
1997	0
5735	0
4348	0

2500 rows × 1 columns

In [ ]: