Import Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
Churn = pd.read_csv("F:\dataset\Churn_Modelling.csv")
```

In [3]:

```
Churn.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 [4]:

```
Churn.head()
```

Out[4]:

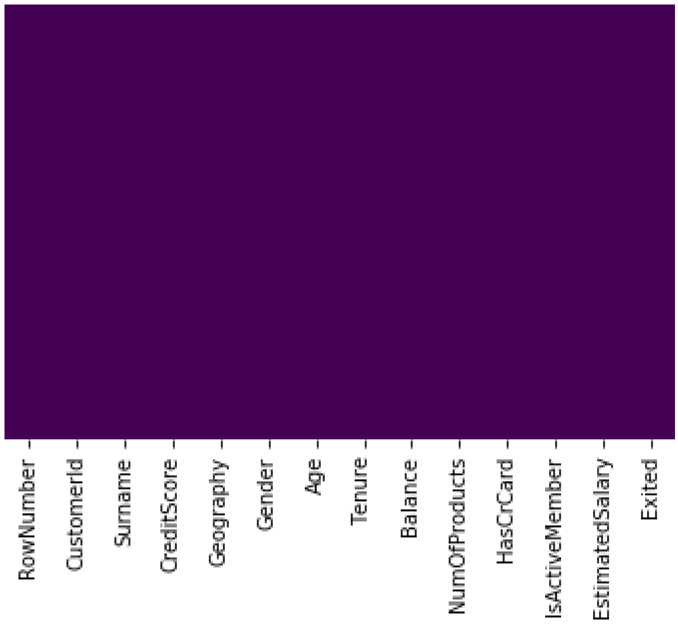
| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Bal |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | |
| **1** | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 8380 |
| **2** | 3 | 15619304 | 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]:

```
sns.heatmap(Churn.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[5]:

<matplotlib.axes._subplots.AxesSubplot at 0x265a351a2c8>

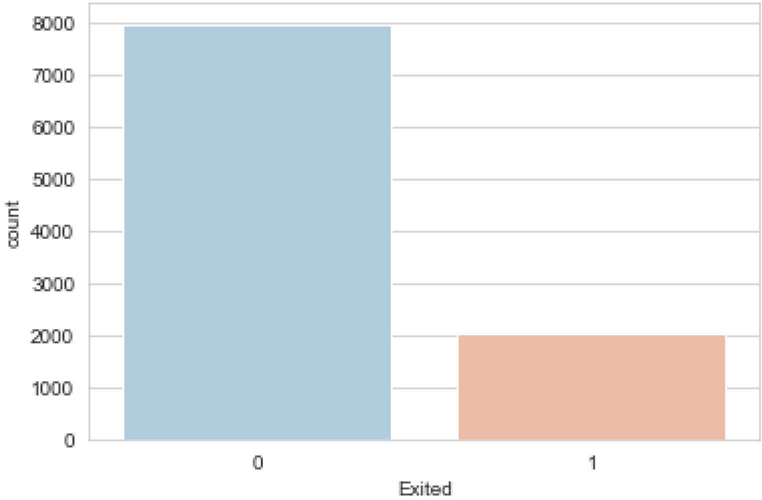



In [6]:

```
sns.set_style('whitegrid')
sns.countplot(x='Exited',data=Churn,palette='RdBu_r')
```

Out[6]:

<matplotlib.axes._subplots.AxesSubplot at 0x265a3d17f88>

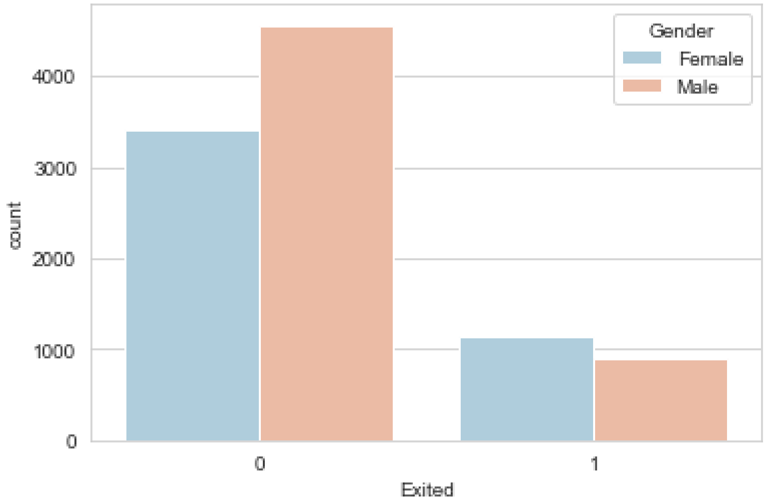


In [7]:

```
sns.set_style('whitegrid')
sns.countplot(x='Exited',hue='Gender',data=Churn,palette='RdBu_r')
```

Out[7]:

<matplotlib.axes._subplots.AxesSubplot at 0x265a3dbd288>

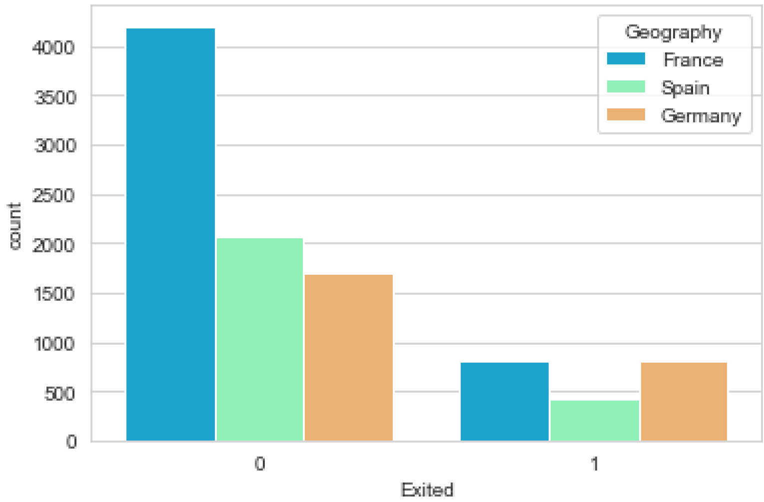



In [8]:

```
sns.set_style('whitegrid')
sns.countplot(x='Exited',hue='Geography',data=Churn,palette='rainbow')
```

Out[8]:

<matplotlib.axes._subplots.AxesSubplot at 0x265a40aa9c8>




In [9]:

```
#Churn['Age'].hist(bins=30,color='darkred',alpha=0.7)
```

# Data Cleaning

In [10]:

```
to_drop=['RowNumber','CustomerId','Surname']
Churn=Churn.drop(to_drop,axis=1)
Churn.head()
```

Out[10]:

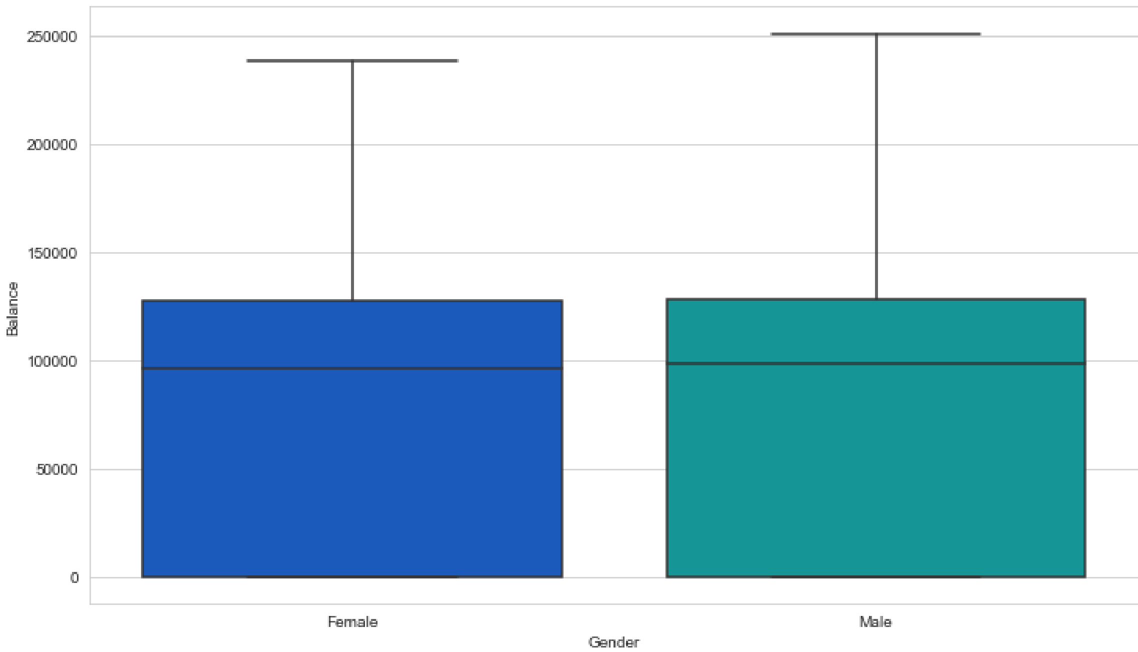
| | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | Is |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 | |
| **1** | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | |
| **2** | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 | |
| **3** | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 | |
| **4** | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 | |

In [11]:

```
plt.figure(figsize=(12, 7))
sns.boxplot(x='Gender',y='Balance',data=Churn,palette='winter')
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x265a411f108>
```

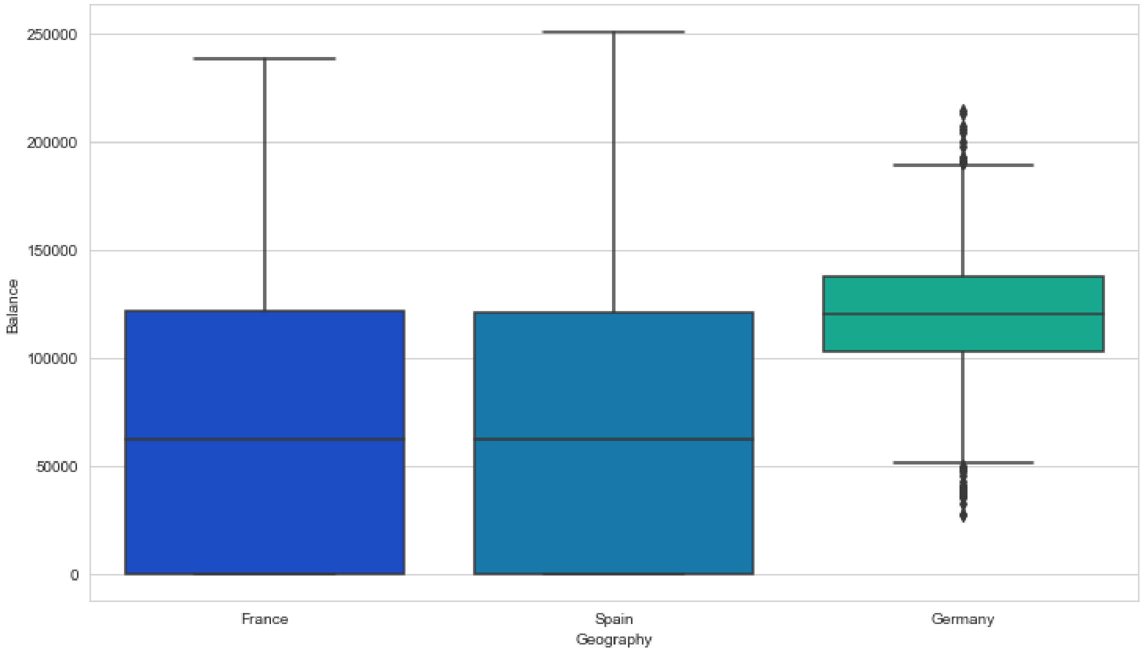


In [12]:

```
plt.figure(figsize=(12, 7))
sns.boxplot(x='Geography',y='Balance',data=Churn,palette='winter')
```

Out[12]:

<matplotlib.axes._subplots.AxesSubplot at 0x265a47e74c8>

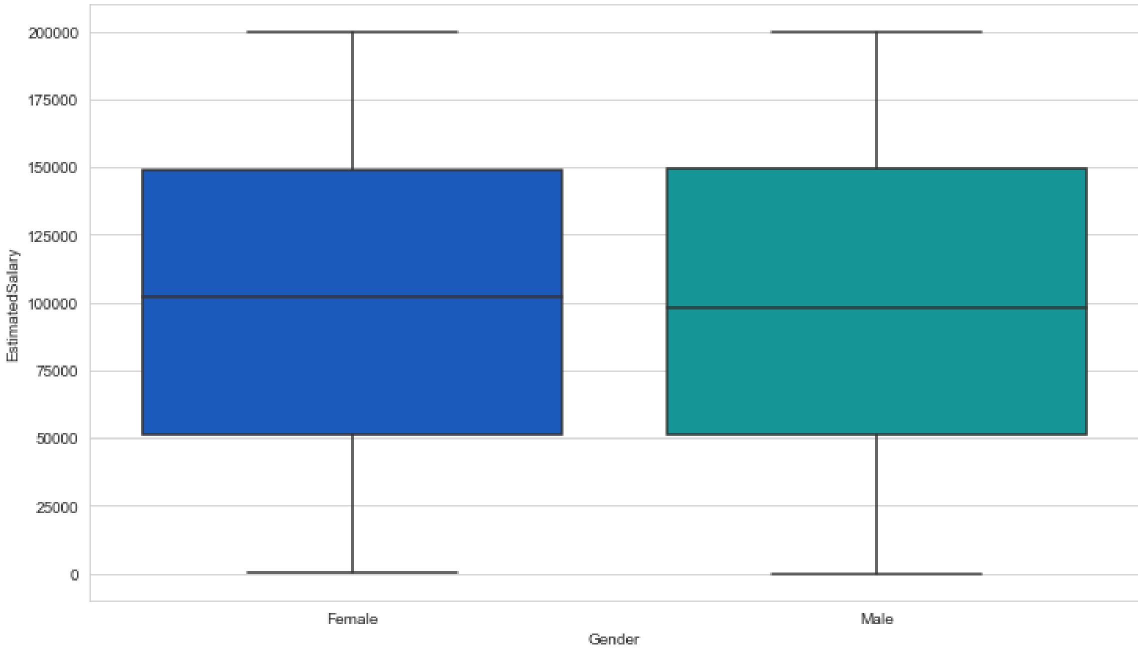


In [13]:

```
plt.figure(figsize=(12, 7))
sns.boxplot(x='Gender',y='EstimatedSalary',data=Churn,palette='winter')
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x265a411fe88>
```

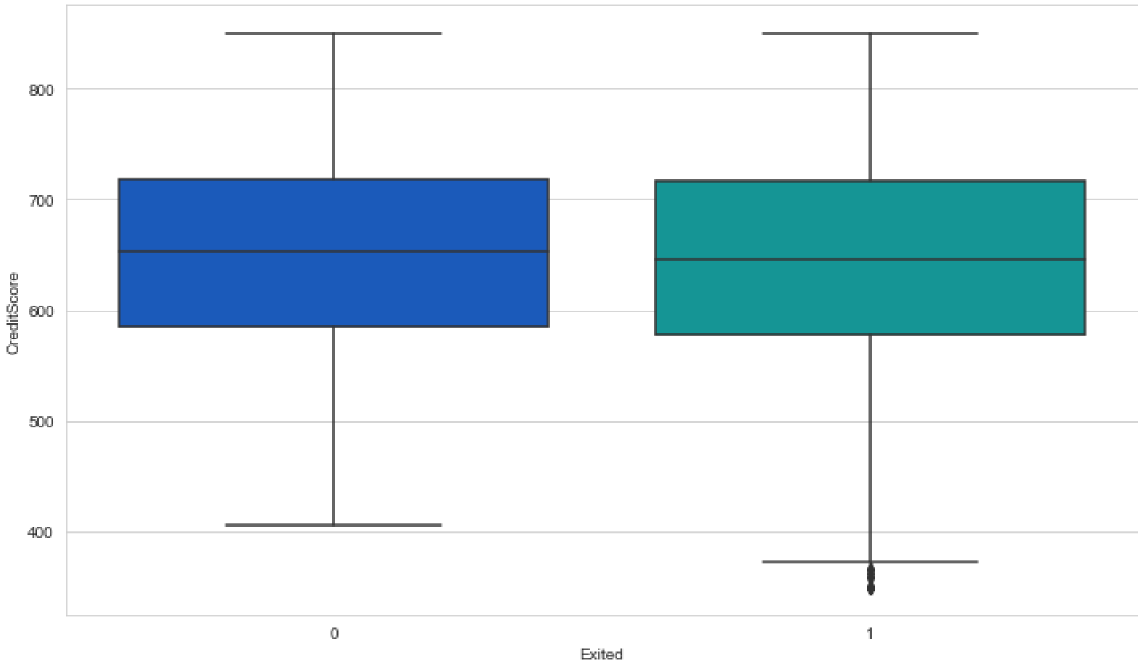



In [14]:

```
plt.figure(figsize=(12, 7))
sns.boxplot(x='Exited',y='CreditScore',data=Churn,palette='winter')
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2659e5d3848>
```




Converting Categorical Features

In [17]:

```
Churn.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  int64  
 8   IsActiveMember   10000 non-null  int64  
 9   EstimatedSalary  10000 non-null  float64
 10  Exited           10000 non-null  int64  
dtypes: float64(2), int64(7), object(2)
memory usage: 859.5+ KB
```

In [18]:

```
sex = pd.get_dummies(Churn['Gender'],drop_first=True)
sex
```

Out[18]:

| | Male |
|---|---|
| **0** | 0 |
| **1** | 0 |
| **2** | 0 |
| **3** | 0 |
| **4** | 0 |
| **...** | ... |
| **9995** | 1 |
| **9996** | 1 |
| **9997** | 0 |
| **9998** | 1 |
| **9999** | 0 |

10000 rows × 1 columns

In [19]:

```
Geo = pd.get_dummies(Churn['Geography'],drop_first=True)
Geo
```

Out[19]:

| | Germany | Spain |
|---|---|---|
| **0** | 0 | 0 |
| **1** | 0 | 1 |
| **2** | 0 | 0 |
| **3** | 0 | 0 |
| **4** | 0 | 1 |
| **...** | ... | ... |
| **9995** | 0 | 0 |
| **9996** | 0 | 0 |
| **9997** | 0 | 0 |
| **9998** | 1 | 0 |
| **9999** | 0 | 0 |

10000 rows × 2 columns

In [20]:

```
to_drop1=['Gender','Geography']
Churn=Churn.drop(to_drop1,axis=1)
Churn.head()
```

Out[20]:

| | CreditScore | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | Estim |
|---|---|---|---|---|---|---|---|---|
| **0** | 619 | 42 | 2 | 0.00 | 1 | 1 | 1 | |
| **1** | 608 | 41 | 1 | 83807.86 | 1 | 0 | 1 | |
| **2** | 502 | 42 | 8 | 159660.80 | 3 | 1 | 0 | |
| **3** | 699 | 39 | 1 | 0.00 | 2 | 0 | 0 | |
| **4** | 850 | 43 | 2 | 125510.82 | 1 | 1 | 1 | |

In [21]:

```
Churn = pd.concat([Churn,sex,Geo],axis=1)
```

In [22]:

```
Churn.head()
```

Out[22]:

| | CreditScore | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | Estim |
|---|---|---|---|---|---|---|---|---|
| **0** | 619 | 42 | 2 | 0.00 | 1 | 1 | 1 | |
| **1** | 608 | 41 | 1 | 83807.86 | 1 | 0 | 1 | |
| **2** | 502 | 42 | 8 | 159660.80 | 3 | 1 | 0 | |
| **3** | 699 | 39 | 1 | 0.00 | 2 | 0 | 0 | |
| **4** | 850 | 43 | 2 | 125510.82 | 1 | 1 | 1 | |

Building a Logistic Regression model

In [23]:

```
from sklearn.model_selection import train_test_split
```

In [24]:

```
X_train, X_test, y_train, y_test = train_test_split(Churn.drop('Exited',axis=1),
                                                    Churn['Exited'], test_size=0.20,
                                                    random_state=111)
```

In [25]:

```
print(X_train,y_train)
```

```
      CreditScore  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
27            571   44       9       0.00              2          0
3847          611   37       6       0.00              2          1
7461          596   32       4       0.00              2          0
1356          709   49       4  154344.49              2          1
4314          638   34       5  133501.36              1          0
...           ...  ...     ...        ...            ...        ...
7490          654   35       2   90865.80              1          1
8873          610   34       0  103108.17              1          0
7443          634   24       2   87413.19              1          1
4182          550   52       5  121016.23              1          1
4820          484   32       3       0.00              2          1

      IsActiveMember  EstimatedSalary  Male  Germany  Spain
27                 0         38433.35     1        0      0
3847               0        110782.88     0        0      0
7461               1        146504.35     1        0      1
1356               1         38794.57     1        0      0
4314               1        155643.04     1        0      0
...              ...              ...   ...      ...    ...
7490               1         86764.46     0        0      0
8873               0        125646.82     1        0      0
7443               0         63340.65     0        0      0
4182               1         41730.37     1        1      0
4820               1        139390.99     0        0      0

[8000 rows x 11 columns] 27      0
3847    0
7461    0
1356    0
4314    0
       ..
7490    0
8873    0
7443    0
4182    1
4820    0
Name: Exited, Length: 8000, dtype: int64
```

In [26]:

```
print(X_test,y_test)
```

```
      CreditScore  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
207           618   34       5  134954.53              1          1
1866          559   70       9       0.00              1          1
9487          850   32       5       0.00              1          1
3673          764   24       7   98148.61              1          1
7178          684   38       5  105069.98              2          1
...           ...  ...     ...        ...            ...        ...
3943          649   46       5       0.00              2          1
4007          648   43       7  139972.18              1          1
8540          484   40       7  106901.42              2          0
1906          786   29       4       0.00              2          1
488           692   30       2       0.00              2          0

      IsActiveMember  EstimatedSalary  Male  Germany  Spain
207                1        151954.39     1        0      0
1866               1        122996.76     0        0      0
9487               1          3830.59     0        0      1
3673               0         26843.76     1        0      0
7178               1        198355.28     1        0      0
...              ...              ...   ...      ...    ...
3943               1         76946.60     1        0      0
4007               0        143668.58     0        0      0
8540               0        118045.98     1        1      0
1906               0        103372.79     0        0      0
488                1        130486.57     1        0      0

[2000 rows x 11 columns] 207     0
1866    0
9487    0
3673    0
7178    0
       ..
3943    0
4007    0
8540    0
1906    0
488     0
Name: Exited, Length: 2000, dtype: int64
```

# Training and Predicting

In [27]:

```
from sklearn.linear_model import LogisticRegression
```

In [28]:

```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

Out[28]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=
0,
                   warm_start=False)
```

In [29]:

```
predictions = logmodel.predict(X_test)
```

In [30]:

```
predictions
```

Out[30]:

```
array([0, 1, 0, ..., 0, 0, 0], dtype=int64)
```

Let's move on to evaluate our model!

# Evaluation

In [31]:

```
from sklearn.metrics import classification_report,confusion_matrix
```

In [32]:

```
print(confusion_matrix(y_test,logmodel.predict(X_test)))
```

```
[[1558   34]
 [ 391   17]]
```

In [33]:

```
print(classification_report(y_test,logmodel.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.80      0.98      0.88      1592
           1       0.33      0.04      0.07       408

    accuracy                           0.79      2000
   macro avg       0.57      0.51      0.48      2000
weighted avg       0.70      0.79      0.72      2000
```

In [34]:

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, predictions)
fpr, tpr, thresholds = roc_curve(y_test, logmodel.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
#plt.savefig('Log_ROC')
plt.show()
```


In [35]:

```
roc_auc_score(y_test, logmodel.predict(X_test))
```

Out[35]:

0.5101549413735343

In [36]:

```
import statsmodels.api as sm
logit_model=sm.Logit(y_train,X_train)
result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.437496
         Iterations 6
                          Results: Logit
==================================================================
Model:              Logit            Pseudo R-squared: 0.134
Dependent Variable: Exited           AIC:              7021.9371
Date:               2020-07-06 08:42 BIC:              7098.7962
No. Observations:   8000             Log-Likelihood:   -3500.0
Df Model:           10               LL-Null:          -4043.1
Df Residuals:       7989             LLR p-value:      4.8508e-227
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
------------------------------------------------------------------
                   Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
------------------------------------------------------------------
CreditScore       -0.0034   0.0002 -16.0292 0.0000 -0.0038 -0.0030
Age                0.0586   0.0026  22.3630 0.0000  0.0534  0.0637
Tenure            -0.0456   0.0101  -4.4962 0.0000 -0.0655 -0.0257
Balance            0.0000   0.0000   1.3242 0.1854 -0.0000  0.0000
NumOfProducts     -0.3786   0.0507  -7.4704 0.0000 -0.4779 -0.2793
HasCrCard         -0.1629   0.0637  -2.5584 0.0105 -0.2877 -0.0381
IsActiveMember    -1.0876   0.0633 -17.1750 0.0000 -1.2117 -0.9635
EstimatedSalary   -0.0000   0.0000  -1.7459 0.0808 -0.0000  0.0000
Male              -0.6537   0.0597 -10.9529 0.0000 -0.7706 -0.5367
Germany            0.7855   0.0754  10.4220 0.0000  0.6378  0.9333
Spain             -0.0295   0.0774  -0.3813 0.7030 -0.1813  0.1222
==================================================================
```

In [37]:

```
to_drop2=['Balance','EstimatedSalary','Spain']
Churn1=Churn.drop(to_drop2,axis=1)
Churn1.head()
```

Out[37]:

| | CreditScore | Age | Tenure | NumOfProducts | HasCrCard | IsActiveMember | Exited | Male | Ge |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 619 | 42 | 2 | 1 | 1 | 1 | 1 | 0 | |
| **1** | 608 | 41 | 1 | 1 | 0 | 1 | 0 | 0 | |
| **2** | 502 | 42 | 8 | 3 | 1 | 0 | 1 | 0 | |
| **3** | 699 | 39 | 1 | 2 | 0 | 0 | 0 | 0 | |
| **4** | 850 | 43 | 2 | 1 | 1 | 1 | 0 | 0 | |

In [38]:

```
X_train, X_test, y_train, y_test = train_test_split(Churn1.drop('Exited',axis=1),
                                                    Churn1['Exited'], test_size=0.20,
                                                    random_state=111)
```

In [39]:

```
import statsmodels.api as sm
logit_model=sm.Logit(y_train,X_train)
result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.437789
         Iterations 6
                          Results: Logit
==================================================================
Model:              Logit            Pseudo R-squared: 0.134
Dependent Variable: Exited           AIC:              7020.6276
Date:               2020-07-06 08:42 BIC:              7076.5252
No. Observations:   8000             Log-Likelihood:   -3502.3
Df Model:           7                LL-Null:          -4043.1
Df Residuals:       7992             LLR p-value:      2.8493e-229
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
------------------------------------------------------------------
                  Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
------------------------------------------------------------------
CreditScore      -0.0034   0.0002 -17.2715 0.0000 -0.0038 -0.0030
Age               0.0584   0.0026  22.4868 0.0000  0.0533  0.0634
Tenure           -0.0457   0.0101  -4.5223 0.0000 -0.0656 -0.0259
NumOfProducts    -0.4005   0.0493  -8.1207 0.0000 -0.4972 -0.3038
HasCrCard        -0.1645   0.0636  -2.5866 0.0097 -0.2892 -0.0399
IsActiveMember   -1.0844   0.0633 -17.1393 0.0000 -1.2084 -0.9604
Male             -0.6510   0.0595 -10.9325 0.0000 -0.7677 -0.5343
Germany           0.8332   0.0641  12.9964 0.0000  0.7075  0.9589
==================================================================
```

In [40]:

```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
predictions1 = logmodel.predict(X_test)
predictions1
```

```
C:\Users\Pratima Dhar\anaconda3\lib\site-packages\sklearn\linear_model\_lo
gistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[40]:

```
array([0, 1, 0, ..., 0, 0, 0], dtype=int64)
```

In [41]:

```
from sklearn.metrics import classification_report,confusion_matrix
```

In [42]:

```
print(confusion_matrix(y_test,logmodel.predict(X_test)))
```

```
[[1534   58]
 [ 325   83]]
```

In [43]:

```
print(classification_report(y_test,logmodel.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      1592
           1       0.59      0.20      0.30       408

    accuracy                           0.81      2000
   macro avg       0.71      0.58      0.60      2000
weighted avg       0.78      0.81      0.77      2000
```

In [44]:

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, predictions1)
fpr, tpr, thresholds = roc_curve(y_test, logmodel.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
#plt.savefig('Log_ROC')
plt.show()
```