Bank Customer Churn Rate Prediction using ANN

In [150]:
```python
# Importing all the required library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [151]:
```python
data  = pd.read_csv('Churn_Modelling.csv')
data.head()
```

Out[151]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balar
0	1	15634602	Hargrave	619	France	Female	42	2	0
1	2	15647311	Hill	608	Spain	Female	41	1	83807
2	3	15619304	Onio	502	France	Female	42	8	159660
3	4	15701354	Boni	699	France	Female	39	1	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510

In [152]:
```python
data.shape
```

Out[152]:
```
(10000, 14)
```

In [153]:
```python
data.columns
```

Out[153]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [154]:
```
data.dtypes
```

Out[154]:
```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

In [155]:
```
# checking the total number of missing values present in the data
data.isnull().sum()
```

Out[155]:
```
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 [156]:
```
data.duplicated().sum()
```

Out[156]:
```
0
```

In [157]:
```
data.nunique()
```

Out[157]:
```
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 [158]:
```
data.drop(['RowNumber','CustomerId','Surname'], axis=1, inplace=True)
```

Descriptive statistics for numerical columns

In [159]:
```
data.describe()
```

Out[159]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000
mean	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550
std	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584
min	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000
25%	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000
max	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000

Descriptive statistics for categorical columns

In [160]:
```
data.describe(include=[object])
```

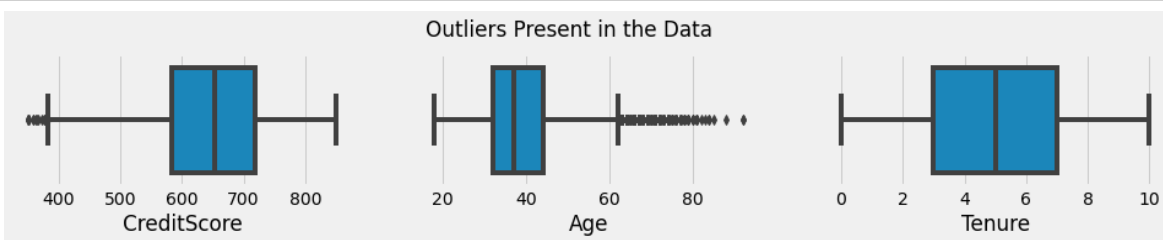
Out[160]:

	Geography	Gender
count	10000	10000
unique	3	2
top	France	Male
freq	5014	5457

In [161]:
```
data_col = data[['Exited','Gender','CreditScore','Age','Tenure','NumOfProduct
                 'EstimatedSalary','Geography']]
```

Outlier detection

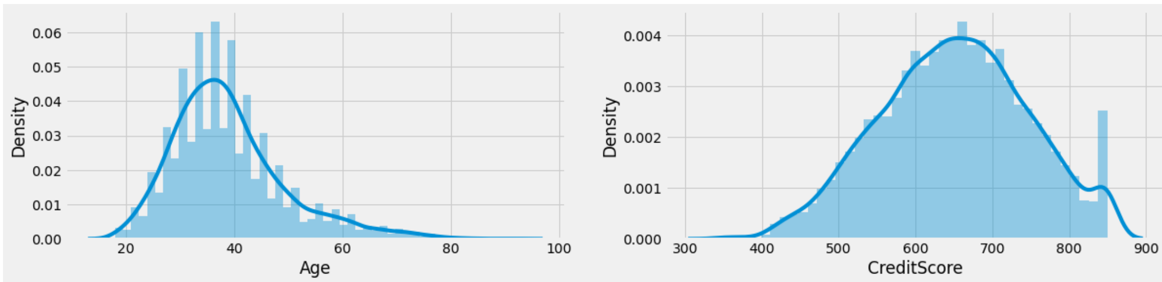
In [162]:
```
# univariate analysis on numerical columns
plt.style.use('fivethirtyeight')
plt.rcParams['figure.figsize'] = (15, 4)

plt.subplot(2, 3, 1)
sns.boxplot(data_col['CreditScore'])

plt.subplot(2, 3, 2)
sns.boxplot(data_col['Age'])

plt.subplot(2, 3, 3)
sns.boxplot(data_col['Tenure'])

plt.suptitle('Outliers Present in the Data')
plt.show()
```



Univariate Analysis on the Numerical Columns

In [163]:
```
plt.style.use('fivethirtyeight')
plt.rcParams['figure.figsize'] = (18, 4)

plt.subplot(1, 2, 1)
sns.distplot(data_col['Age'])

plt.subplot(1, 2, 2)
sns.distplot(data_col['CreditScore'])
```

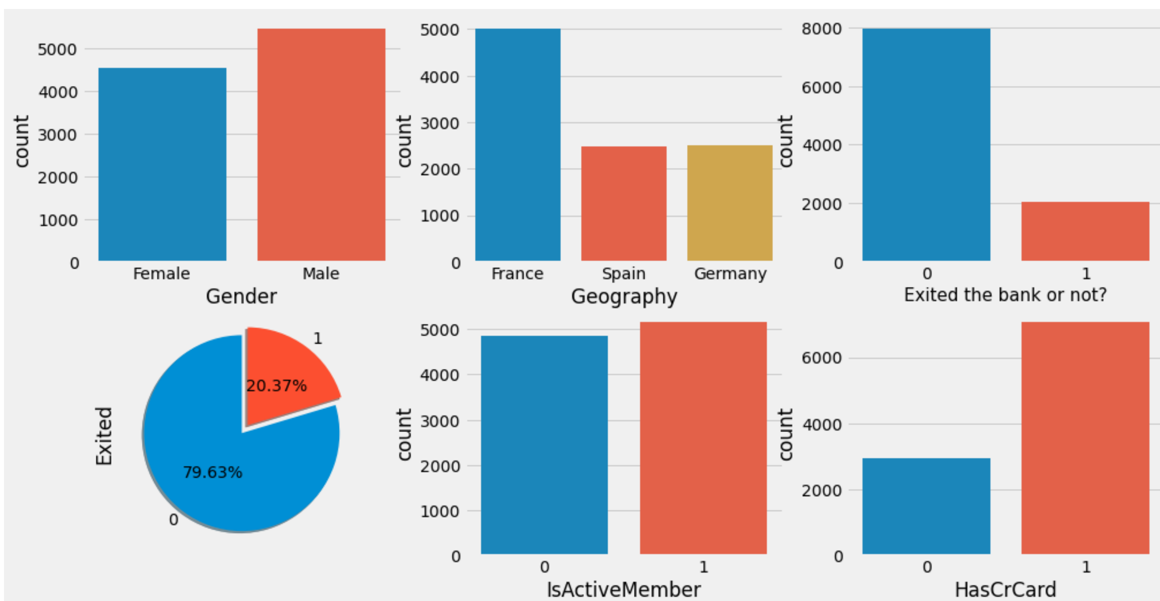
Out[163]: <AxesSubplot:xlabel='CreditScore', ylabel='Density'>



Univariate Analysis on the Categorical Columns

In [164]:
```
# univariate analysis on categorical column
plt.rcParams['figure.figsize'] = (15, 8)

plt.subplot(2, 3, 1)
sns.countplot(data['Gender'])

plt.subplot(2, 3, 2)
sns.countplot(data['Geography'])

plt.subplot(2, 3, 3)
sns.countplot(data_col['Exited'])

plt.xlabel('Exited the bank or not?', fontsize = 15)

plt.subplot(2, 3, 4)
data_col['Exited'].value_counts().plot(kind = 'pie', explode = [0, 0.1], autc
                                       labels = ['0','1'], shadow = True, pct

plt.subplot(2, 3, 5)
sns.countplot(data_col['IsActiveMember'])

plt.subplot(2, 3, 6)
sns.countplot(data_col['HasCrCard'])
```

Out[164]: <AxesSubplot:xlabel='HasCrCard', ylabel='count'>



In [165]:

```
print('Impact of Gender on Churn rate')
print(pd.crosstab(data_col['Gender'], data['Exited']))
print('\n')
```

```
Impact of Gender on Churn rate
Exited     0     1
Gender
Female  3404  1139
Male    4559   898
```

Bivariate Analysis

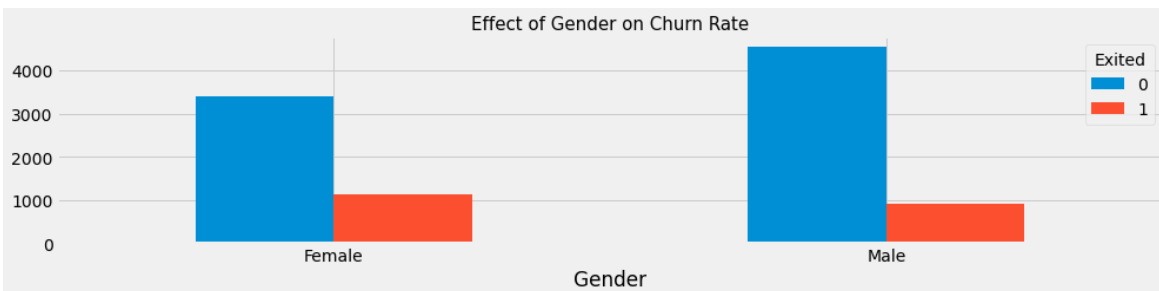
Impact of Gender on Churn Rate

In [226]:

```
plt.rcParams['figure.figsize'] = (15, 3)
x = pd.crosstab(data_col['Gender'], data_col['Exited'])
barplot = x.plot.bar(rot=0)
plt.title('Effect of Gender on Churn Rate', fontsize = 15)
```

Out[226]:

```
Text(0.5, 1.0, 'Effect of Gender on Churn Rate')
```

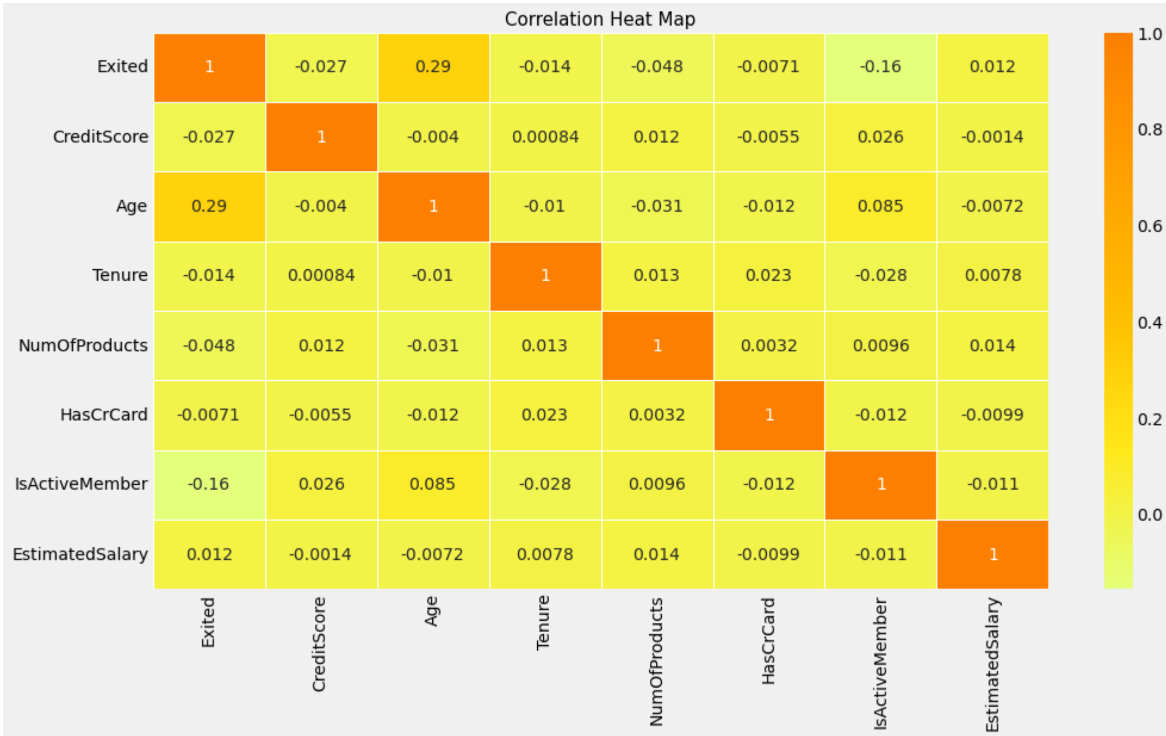



```
it appears that female customers are slightly more likely to churn from the
bank across all three countries.
```

Multivariate Analysis

In [234]:

```
# multivariate analysis
# lets check the Heat Map for the Data with respect to correlation.

plt.rcParams['figure.figsize'] = (15, 8)
sns.heatmap(data_col.corr(), annot = True, linewidth = 0.5, cmap = 'Wistia')
plt.title('Correlation Heat Map', fontsize = 15)
plt.show()
```



Encoding the categorical columns to convert them into numerical columns using LabelEncoder

In [180]:
```python
# lets start encoding these categorical columns to convert them into numerica
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data['Gender'] = le.fit_transform(data['Gender'])
data['Geography'] = le.fit_transform(data['Geography'])
```

In [181]:
```python
data.head()
```

Out[181]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsA
0	619	0	0	24	2	0.00	1	1	
1	608	2	0	23	1	83807.86	1	0	
2	502	0	0	24	8	159660.80	3	1	
3	699	0	0	21	1	0.00	2	0	
4	850	2	0	25	2	125510.82	1	1	

In [195]:
```python
x = data.drop(['Exited'], axis=1)
y = data['Exited']
```

In [196]:
```python
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, ranc

print("Shape of the x Train :", x_train.shape)
print("Shape of the y Train :", y_train.shape)
print("Shape of the x Test :", x_test.shape)
print("Shape of the y Test :", y_test.shape)
```

```
Shape of the x Train : (8000, 10)
Shape of the y Train : (8000,)
Shape of the x Test : (2000, 10)
Shape of the y Test : (2000,)
```

Feature Scalling using StandardScaler

In [198]:
```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train)
x_trained_scaler = scaler.transform(x_train)
x_test_scaler = scaler.transform(x_test)
```

In [202]:
```python
x_trained_scaler.shape
```

Out[202]: (8000, 10)

Creating the model using Artifical Neural Network

In [232]:
```
import tensorflow as tf
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Dense(10, input_shape=(10,), activation='relu'),
    keras.layers.Dense(14, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam',
              loss = 'binary_crossentropy',
              metrics=['accuracy'])

model.fit(x_trained_scaler, y_train, epochs=50)
```

```
Epoch 1/50
250/250 [==============================] - 1s 1ms/step - loss: 0.4792 - acc
uracy: 0.7945
Epoch 2/50
250/250 [==============================] - 0s 1ms/step - loss: 0.4467 - acc
uracy: 0.7945
Epoch 3/50
250/250 [==============================] - 0s 1ms/step - loss: 0.4347 - acc
uracy: 0.7946
Epoch 4/50
250/250 [==============================] - 0s 1ms/step - loss: 0.4248 - acc
uracy: 0.8069
Epoch 5/50
250/250 [==============================] - 0s 1ms/step - loss: 0.4147 - acc
uracy: 0.8216
Epoch 6/50
250/250 [==============================] - 0s 1ms/step - loss: 0.4025 - acc
uracy: 0.8276
Epoch 7/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3904 - acc
uracy: 0.8335
Epoch 8/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3796 - acc
uracy: 0.8419
Epoch 9/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3702 - acc
uracy: 0.8457
Epoch 10/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3631 - acc
uracy: 0.8479
Epoch 11/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3574 - acc
uracy: 0.8520
Epoch 12/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3533 - acc
uracy: 0.8526
Epoch 13/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3503 - acc
uracy: 0.8549
Epoch 14/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3482 - acc
```

```
uracy: 0.8571
Epoch 15/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3461 - acc
uracy: 0.8576
Epoch 16/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3448 - acc
uracy: 0.8591
Epoch 17/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3435 - acc
uracy: 0.8584
Epoch 18/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3421 - acc
uracy: 0.8602
Epoch 19/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3408 - acc
uracy: 0.8601
Epoch 20/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3405 - acc
uracy: 0.8611
Epoch 21/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3396 - acc
uracy: 0.8612
Epoch 22/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3398 - acc
uracy: 0.8611
Epoch 23/50
250/250 [==============================] - 0s 2ms/step - loss: 0.3386 - acc
uracy: 0.8608
Epoch 24/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3382 - acc
uracy: 0.8605
Epoch 25/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3379 - acc
uracy: 0.8610
Epoch 26/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3376 - acc
uracy: 0.8611
Epoch 27/50
250/250 [==============================] - 0s 2ms/step - loss: 0.3373 - acc
uracy: 0.8625
Epoch 28/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3363 - acc
uracy: 0.8615
Epoch 29/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3363 - acc
uracy: 0.8612
Epoch 30/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3356 - acc
uracy: 0.8612
Epoch 31/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3357 - acc
uracy: 0.8606
Epoch 32/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3356 - acc
uracy: 0.8616
Epoch 33/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3349 - acc
```

```
uracy: 0.8629
Epoch 34/50
250/250 [==============================] - 0s 2ms/step - loss: 0.3351 - acc
uracy: 0.8621
Epoch 35/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3346 - acc
uracy: 0.8627
Epoch 36/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3345 - acc
uracy: 0.8627
Epoch 37/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3341 - acc
uracy: 0.8648
Epoch 38/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3342 - acc
uracy: 0.8644
Epoch 39/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3336 - acc
uracy: 0.8648
Epoch 40/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3341 - acc
uracy: 0.8634
Epoch 41/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3335 - acc
uracy: 0.8627
Epoch 42/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3333 - acc
uracy: 0.8634
Epoch 43/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3329 - acc
uracy: 0.8645
Epoch 44/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3331 - acc
uracy: 0.8627
Epoch 45/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3328 - acc
uracy: 0.8641
Epoch 46/50
250/250 [==============================] - 0s 2ms/step - loss: 0.3328 - acc
uracy: 0.8648
Epoch 47/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3325 - acc
uracy: 0.8626
Epoch 48/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3318 - acc
uracy: 0.8637
Epoch 49/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3326 - acc
uracy: 0.8648
Epoch 50/50
250/250 [==============================] - 0s 1ms/step - loss: 0.3320 - acc
uracy: 0.8655
```

Out[232]: <keras.callbacks.History at 0x267e9c92850>

In [210]:
```
model.evaluate(x_test_scaler, y_test)
```

```
63/63 [==============================] - 0s 1ms/step - loss: 0.3418 - accur
acy: 0.8595
```

Out[210]:
```
[0.34178730845451355, 0.859499990940094]
```

In [213]:
```
y_predict = model.predict(x_test_scaler)
y_predict[:5]
```

```
63/63 [==============================] - 0s 1ms/step
```

Out[213]:
```
array([[0.04401844],
       [0.03105499],
       [0.08082822],
       [0.10456759],
       [0.11255805]], dtype=float32)
```

In [214]:
```
y_pred = []
for element in y_predict:
    if element > 0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)
```

In [215]:
```
y_pred[:10]
```

Out[215]:
```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
```

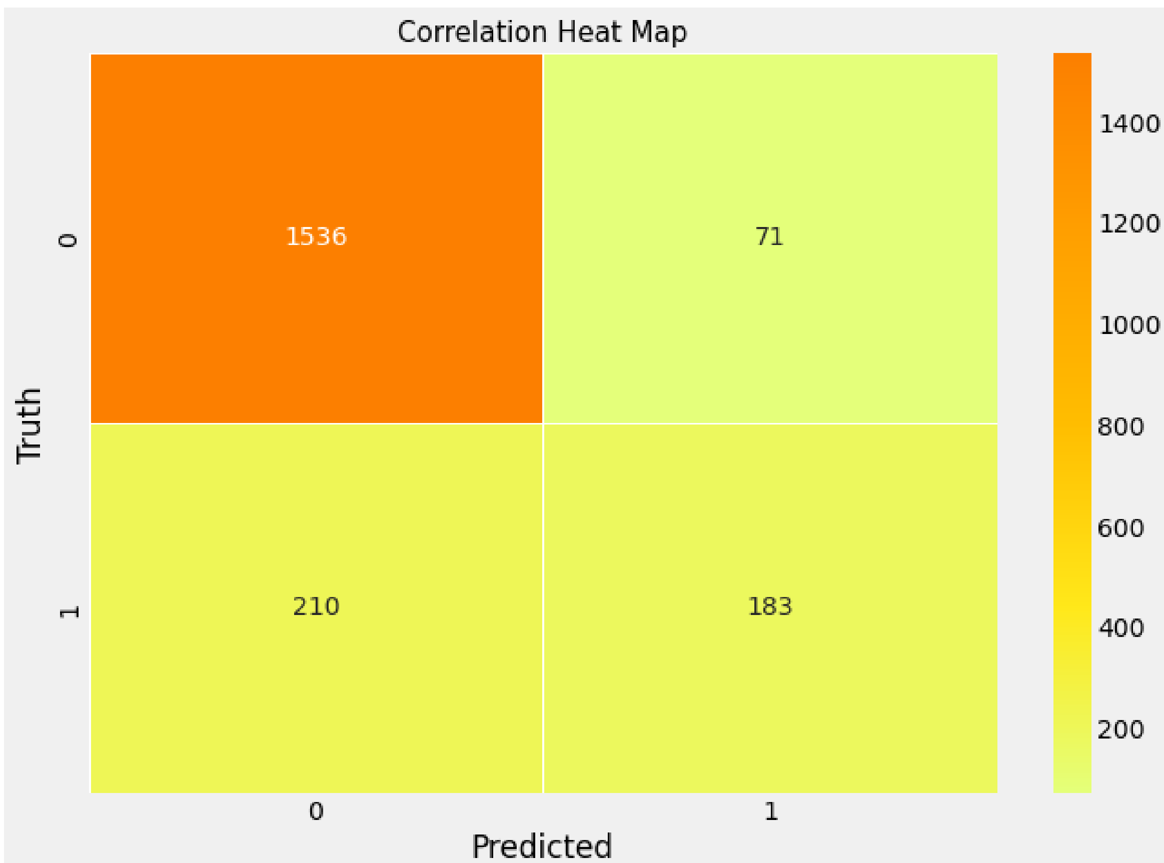
In [218]:
```
y_test[:10]
```

Out[218]:
```
6252    0
4684    0
1731    0
4742    0
4521    0
6340    0
576     0
5202    1
6363    0
439     0
Name: Exited, dtype: int64
```

In [225]:

```
from sklearn.metrics import confusion_matrix, classification_report

print(classification_report(y_test, y_pred))

cm = tf.math.confusion_matrix(labels=y_test,predictions=y_pred)

plt.rcParams['figure.figsize'] = (10, 7)
sns.heatmap(cm, annot = True, linewidth = 0.5, cmap = 'Wistia', fmt='d')
plt.title('Correlation Heat Map', fontsize = 15)
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.88      0.96      0.92      1607
           1       0.72      0.47      0.57       393

    accuracy                           0.86      2000
   macro avg       0.80      0.71      0.74      2000
weighted avg       0.85      0.86      0.85      2000
```



The Accuracy of the model using ANN is 86%

In [ ]:

```

```