In [39]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
import io
```

Read the Dataset

In [2]:
```
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 bank.csv to bank.csv
```

In [40]:
```
df=pd.read_csv(io.StringIO(uploaded['bank.csv'].decode('utf-8')))
df.head()
```

Out[40]:

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

2. Drop the Columns which are unique for all users

In [41]:
```
df=df.drop(['RowNumber','CustomerId','Surname'],axis=1)
df.head()
```

Out[41]:

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

In [42]:
```
df.isna().any()
```

```
df.isna().sum()
```

Out[42]:
```
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

BiVariate Analysis

In [43]:
```
print(df.shape)
df.info()
```

```
(10000, 11)
<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 [44]:
```
df.describe()
```

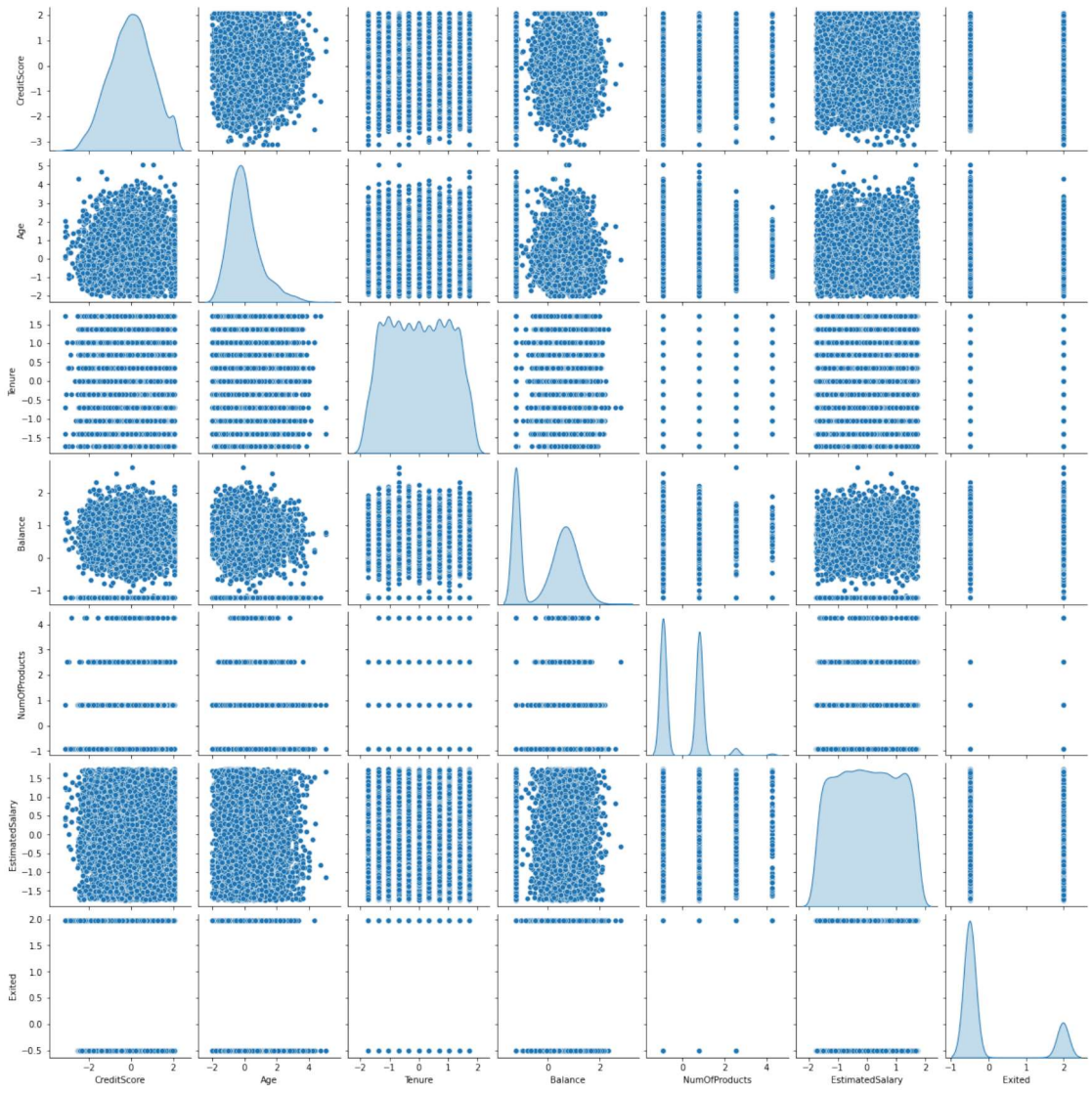
Out[44]:

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

Before performing Bivariate analysis, Lets bring all the features to the same range

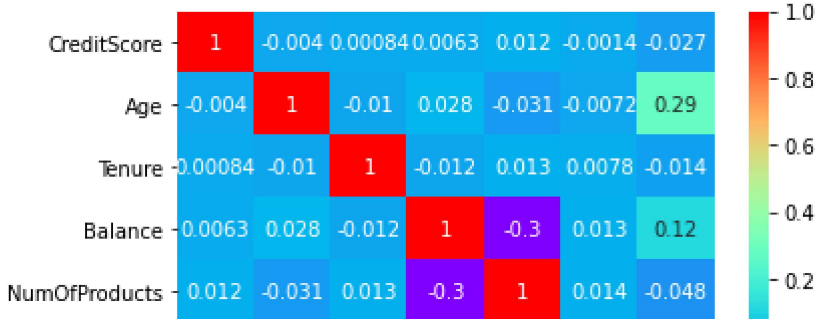
In [45]:
```
## Scale the data
scaler=StandardScaler()
## Extract only the Numerical Columns to perform Bivariate Analysis
subset=df.drop(['Geography','Gender','HasCrCard','IsActiveMember'],axis=1)
scaled=scaler.fit_transform(subset)
scaled_df=pd.DataFrame(scaled,columns=subset.columns)
sns.pairplot(scaled_df,diag_kind='kde')
```

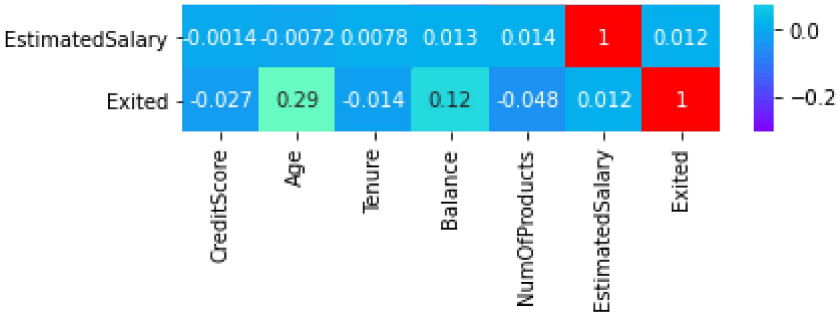
Out[45]: <seaborn.axisgrid.PairGrid at 0x7fe8126f0940>



In [46]:
```
sns.heatmap(scaled_df.corr(),annot=True,cmap='rainbow')
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe7cb4ef9b0>

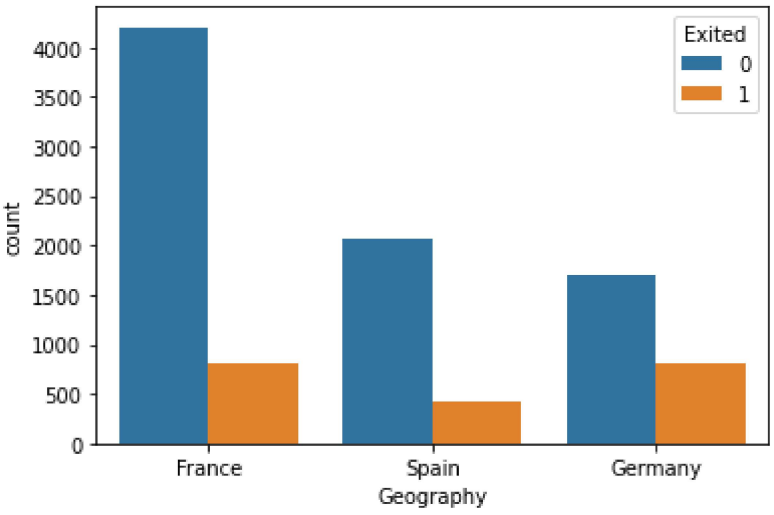


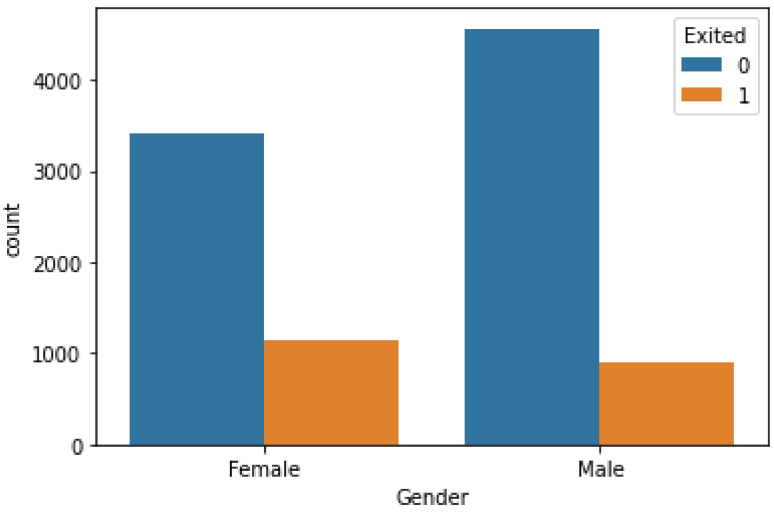


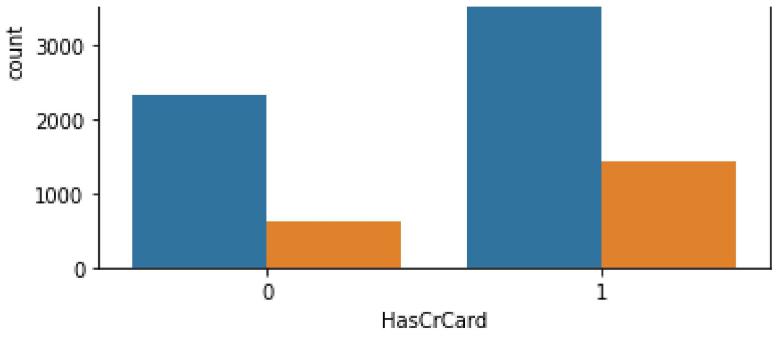
From the above plots, We can see that there is no significant Linear relationship between the features

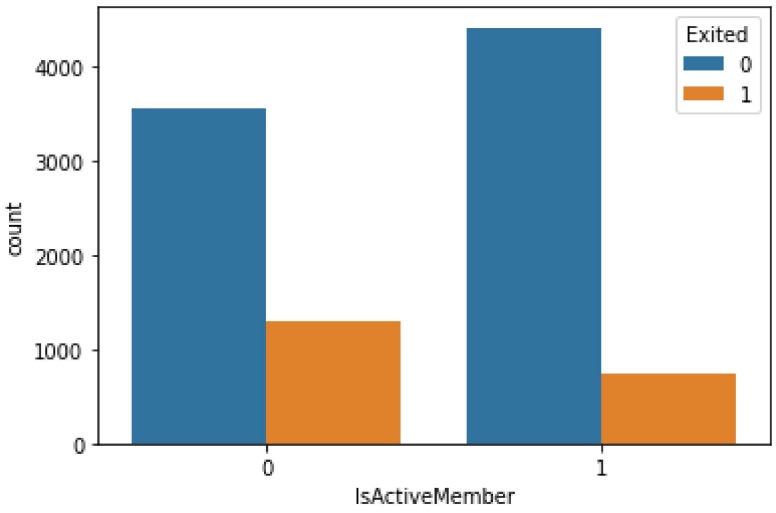
In [47]:

```
## Categorical Features vs Target Variable
sns.countplot(x='Geography',data=df,hue='Exited')
plt.show()
sns.countplot(x='Gender',data=df,hue='Exited')
plt.show()
sns.countplot(x='HasCrCard',data=df,hue='Exited')
plt.show()
sns.countplot(x='IsActiveMember',data=df,hue='Exited')
plt.show()
```







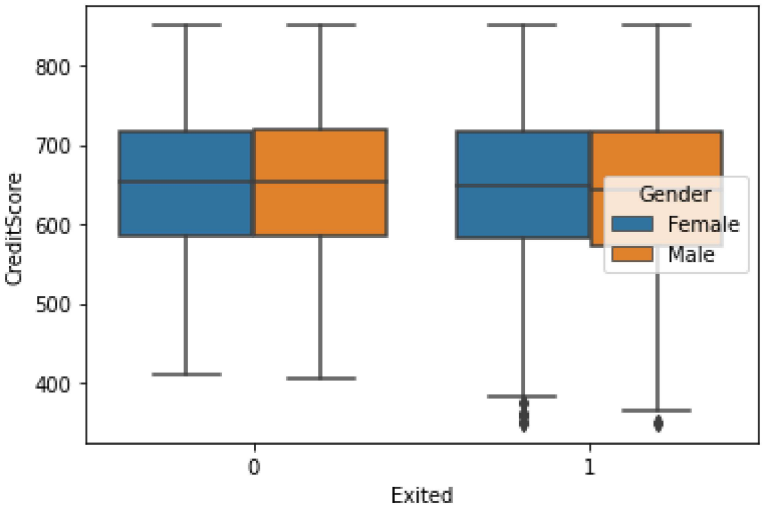


Analysing the Numerical Features relationship with the Target variable. Here 'Exited' is the Target Feature.

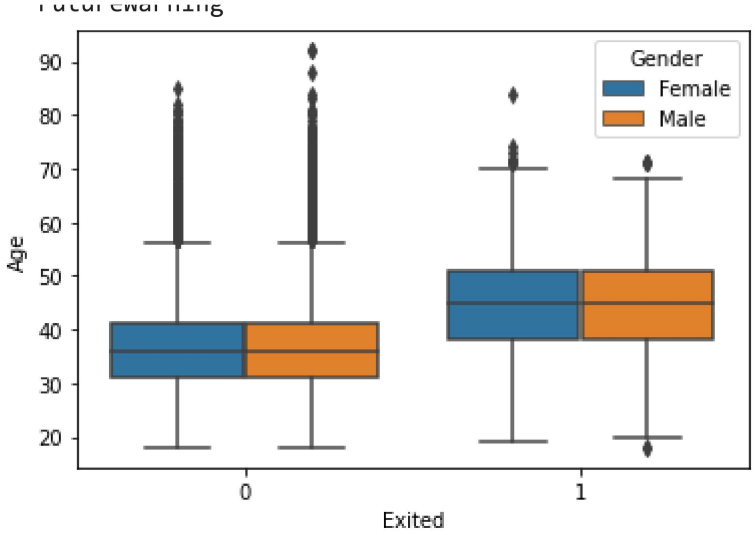
In [50]:

```
subset=subset.drop('Exited',axis=1)
for i in subset.columns:
  sns.boxplot(df['Exited'],df[i],hue=df['Gender'])
  plt.show()
```

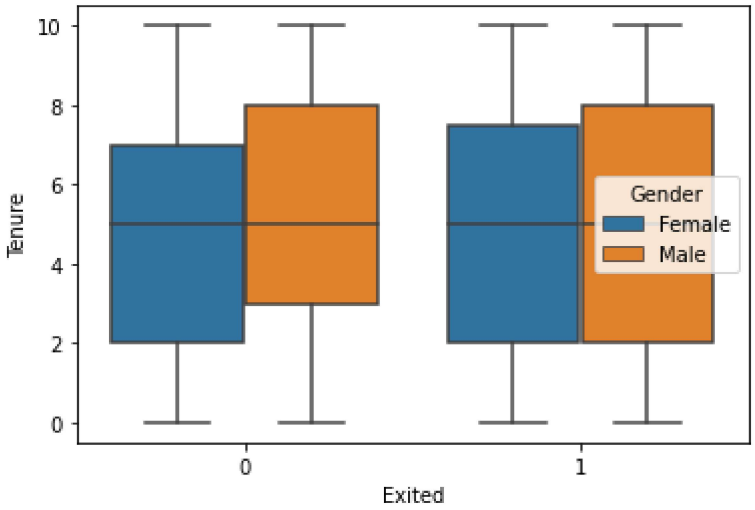
```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



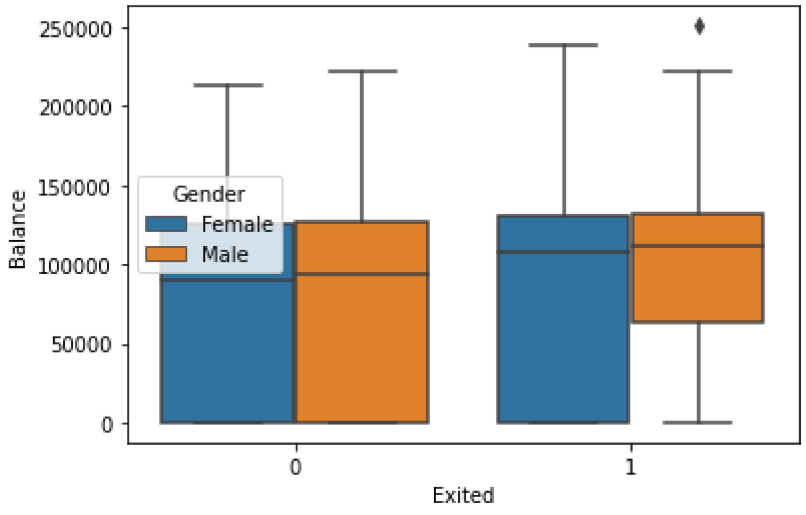

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```




```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

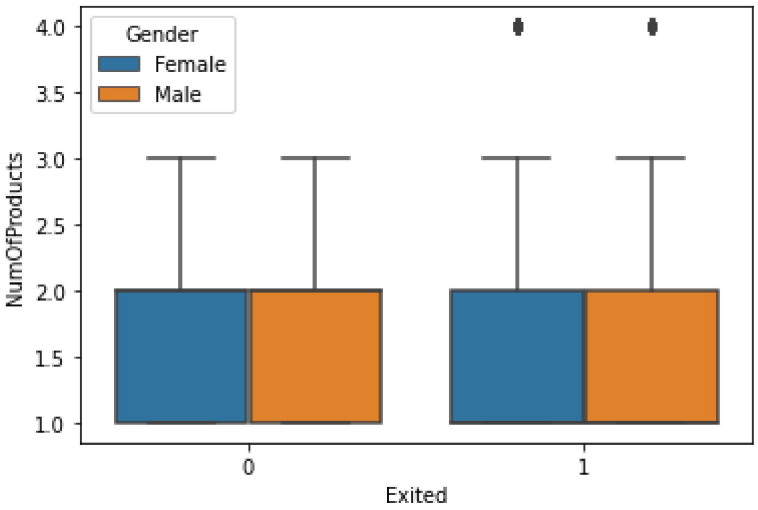



```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

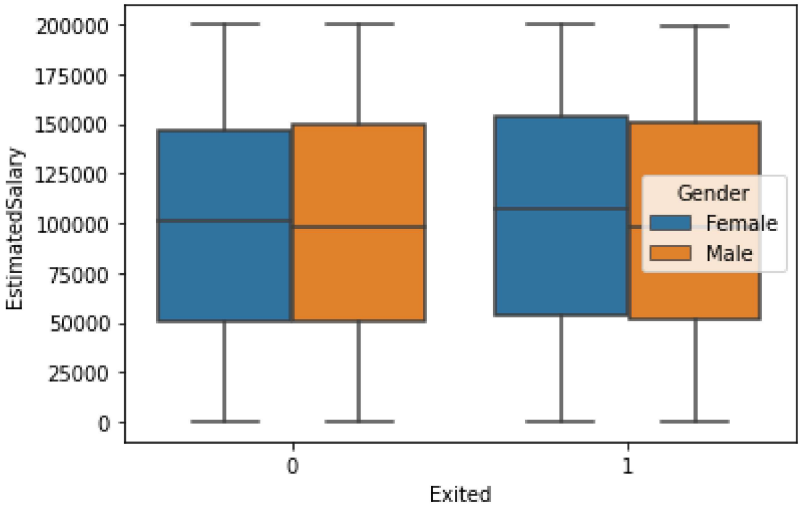



```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
```

```
FutureWarning
```




```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



Insights from Bivariate Plots

1. The Avg Credit Score seem to be almost the same for Active and Churned customers
2. Young People seem to stick to the bank compared to older people
3. The Average Bank Balance is high for Churned Customers
4. The churning rate is high with German Customers
5. The Churning rate is high among the Non-Active Members

4. Distinguish the Target and Feature Set and divide the dataset into Training and Test sets

In [51]:
```
X=df.drop('Exited',axis=1)
y=df.pop('Exited')
```

In [52]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.10,random_st
```

```
X_train,X_val,y_train,y_val=train_test_split(X_train,y_train,test_size=0.1
print("X_train size is {}".format(X_train.shape[0]))
print("X_val size is {}".format(X_val.shape[0]))
print("X_test size is {}".format(X_test.shape[0]))
```

```
X_train size is 8100
X_val size is 900
X_test size is 1000
```

In [53]:

```
## Standardising the train, Val and Test data
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
num_cols=['CreditScore','Age','Tenure','Balance','NumOfProducts','Estimated
num_subset=scaler.fit_transform(X_train[num_cols])
X_train_num_df=pd.DataFrame(num_subset,columns=num_cols)
X_train_num_df['Geography']=list(X_train['Geography'])
X_train_num_df['Gender']=list(X_train['Gender'])
X_train_num_df['HasCrCard']=list(X_train['HasCrCard'])
X_train_num_df['IsActiveMember']=list(X_train['IsActiveMember'])
X_train_num_df.head()
## Standardise the Validation data
num_subset=scaler.fit_transform(X_val[num_cols])
X_val_num_df=pd.DataFrame(num_subset,columns=num_cols)
X_val_num_df['Geography']=list(X_val['Geography'])
X_val_num_df['Gender']=list(X_val['Gender'])
X_val_num_df['HasCrCard']=list(X_val['HasCrCard'])
X_val_num_df['IsActiveMember']=list(X_val['IsActiveMember'])
## Standardise the Test data
num_subset=scaler.fit_transform(X_test[num_cols])
X_test_num_df=pd.DataFrame(num_subset,columns=num_cols)
X_test_num_df['Geography']=list(X_test['Geography'])
X_test_num_df['Gender']=list(X_test['Gender'])
X_test_num_df['HasCrCard']=list(X_test['HasCrCard'])
X_test_num_df['IsActiveMember']=list(X_test['IsActiveMember'])
```

In [54]:

```
## Convert the categorical features to numerical
X_train_num_df=pd.get_dummies(X_train_num_df,columns=['Geography','Gender']
X_test_num_df=pd.get_dummies(X_test_num_df,columns=['Geography','Gender'])
X_val_num_df=pd.get_dummies(X_val_num_df,columns=['Geography','Gender'])
X_train_num_df.head()
```

Out[54]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary
0	-1.178587	-1.041960	-1.732257	0.198686	0.820905	1.560315
1	-0.380169	-1.326982	1.730718	-0.022020	-0.907991	-0.713592
2	-0.349062	1.808258	-0.693364	0.681178	0.820905	-1.126515
3	0.625629	2.378302	-0.347067	-1.229191	0.820905	-1.682740
4	-0.203895	-1.136967	1.730718	0.924256	-0.907991	1.332535

Initialise and build the Model

In [55]:

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

```
model=Sequential()
model.add(Dense(7,activation='relu'))
model.add(Dense(10,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
```

In [56]:

```
import tensorflow as tf
optimizer=tf.keras.optimizers.Adam(0.01)
model.compile(loss='binary_crossentropy',optimizer=optimizer,metrics=['accu
```

In [57]:

```
model.fit(X_train_num_df,y_train,epochs=100,batch_size=10,verbose=1)
```

```
Epoch 1/100
810/810 [==============================] - 1s 1ms/step - loss: 0.4511 - accu
racy: 0.8054
Epoch 2/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3623 - accu
racy: 0.8493
Epoch 3/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3543 - accu
racy: 0.8541
Epoch 4/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3433 - accu
racy: 0.8561
Epoch 5/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3291 - accu
racy: 0.8692
Epoch 6/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3488 - accu
racy: 0.8560
Epoch 7/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3439 - accu
racy: 0.8644
Epoch 8/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3399 - accu
racy: 0.8615
Epoch 9/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3480 - accu
racy: 0.8602
Epoch 10/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3321 - accu
racy: 0.8683
Epoch 11/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3329 - accu
racy: 0.8614
Epoch 12/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3466 - accu
racy: 0.8575
Epoch 13/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3386 - accu
racy: 0.8640
Epoch 14/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3342 - accu
racy: 0.8667
Epoch 15/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3375 - accu
racy: 0.8652
Epoch 16/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3412 - accu
racy: 0.8574
```

```
Epoch 17/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3256 - accu
racy: 0.8652
Epoch 18/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3229 - accu
racy: 0.8687
Epoch 19/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3276 - accu
racy: 0.8670
Epoch 20/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3399 - accu
racy: 0.8581
Epoch 21/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3384 - accu
racy: 0.8661
Epoch 22/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3309 - accu
racy: 0.8638
Epoch 23/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3441 - accu
racy: 0.8592
Epoch 24/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3232 - accu
racy: 0.8697
Epoch 25/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3409 - accu
racy: 0.8651
Epoch 26/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3361 - accu
racy: 0.8622
Epoch 27/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3380 - accu
racy: 0.8596
Epoch 28/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3363 - accu
racy: 0.8611
Epoch 29/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3347 - accu
racy: 0.8636
Epoch 30/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3459 - accu
racy: 0.8623
Epoch 31/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3252 - accu
racy: 0.8658
Epoch 32/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3363 - accu
racy: 0.8621
Epoch 33/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3347 - accu
racy: 0.8638
Epoch 34/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3171 - accu
racy: 0.8725
Epoch 35/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3270 - accu
racy: 0.8666
Epoch 36/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3347 - accu
racy: 0.8623
Epoch 37/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3206 - accu
racy: 0.8671
Epoch 38/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3315 - accu
```

```
racy: 0.8645
Epoch 39/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3319 - accu
racy: 0.8590
Epoch 40/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3442 - accu
racy: 0.8556
Epoch 41/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3381 - accu
racy: 0.8623
Epoch 42/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3328 - accu
racy: 0.8673
Epoch 43/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3283 - accu
racy: 0.8655
Epoch 44/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3423 - accu
racy: 0.8582
Epoch 45/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3252 - accu
racy: 0.8674
Epoch 46/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3350 - accu
racy: 0.8618
Epoch 47/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3291 - accu
racy: 0.8644
Epoch 48/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3323 - accu
racy: 0.8651
Epoch 49/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3380 - accu
racy: 0.8621
Epoch 50/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3348 - accu
racy: 0.8596
Epoch 51/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3252 - accu
racy: 0.8685
Epoch 52/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3309 - accu
racy: 0.8688
Epoch 53/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3306 - accu
racy: 0.8652
Epoch 54/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3372 - accu
racy: 0.8628
Epoch 55/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3401 - accu
racy: 0.8618
Epoch 56/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3262 - accu
racy: 0.8619
Epoch 57/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3303 - accu
racy: 0.8661
Epoch 58/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3332 - accu
racy: 0.8674
Epoch 59/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3308 - accu
racy: 0.8612
```

```
Epoch 60/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3285 - accu
racy: 0.8644
Epoch 61/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3385 - accu
racy: 0.8626
Epoch 62/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3344 - accu
racy: 0.8650
Epoch 63/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3267 - accu
racy: 0.8632
Epoch 64/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3299 - accu
racy: 0.8666
Epoch 65/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3163 - accu
racy: 0.8749
Epoch 66/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3187 - accu
racy: 0.8727
Epoch 67/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3396 - accu
racy: 0.8591
Epoch 68/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3206 - accu
racy: 0.8675
Epoch 69/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3334 - accu
racy: 0.8644
Epoch 70/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3242 - accu
racy: 0.8672
Epoch 71/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3221 - accu
racy: 0.8696
Epoch 72/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3226 - accu
racy: 0.8671
Epoch 73/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3252 - accu
racy: 0.8694
Epoch 74/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3246 - accu
racy: 0.8605
Epoch 75/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3397 - accu
racy: 0.8633
Epoch 76/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3309 - accu
racy: 0.8647
Epoch 77/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3268 - accu
racy: 0.8630
Epoch 78/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3257 - accu
racy: 0.8694
Epoch 79/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3248 - accu
racy: 0.8667
Epoch 80/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3367 - accu
racy: 0.8622
Epoch 81/100
810/810 [==============================] - 1s 1ms/step - loss: 0.3329 - accu
```