In [27]:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
# Importing the dataset
dataset = pd.read_csv('Churn_Modelling.csv')
```

In [28]:

```
dataset.head()
```

Out[28]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrC
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1

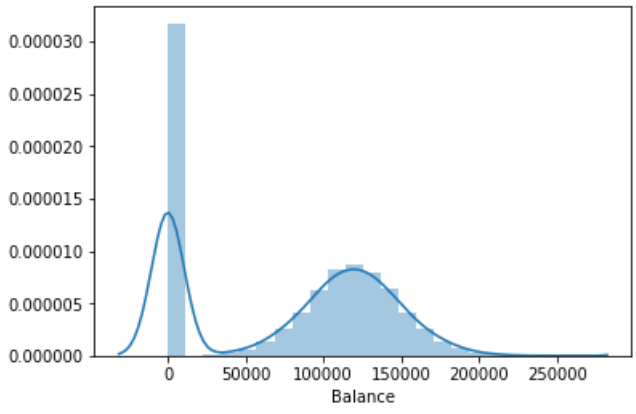
In [31]:

```
sns.distplot(dataset["Balance"])
```

```
C:\Users\Jayen\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The
'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1def2569a58>
```



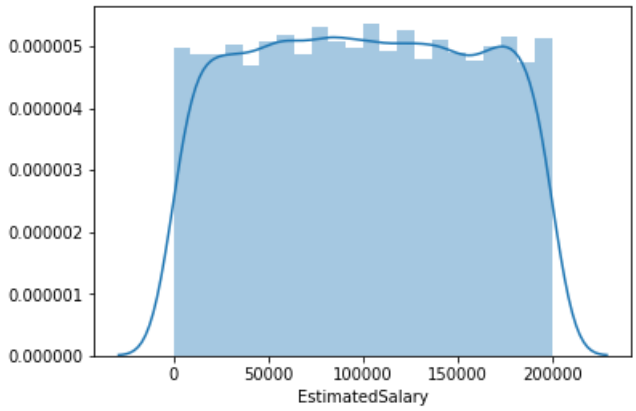
In [32]:

```
sns.distplot(dataset["EstimatedSalary"])
```

```
C:\Users\Jayen\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The
'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1def26e4d30>
```

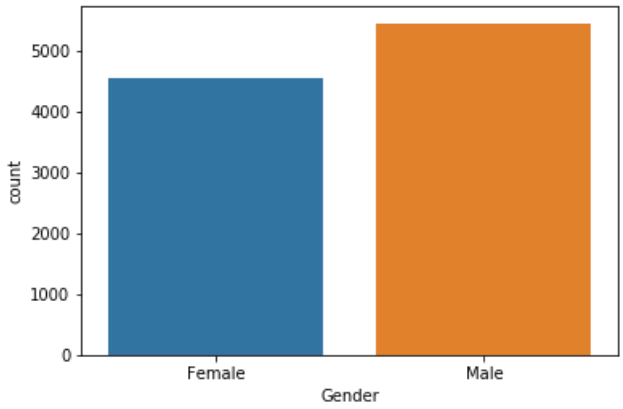


In [35]:

```
sns.countplot(x="Gender",data=dataset)
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1def28fb3c8>
```

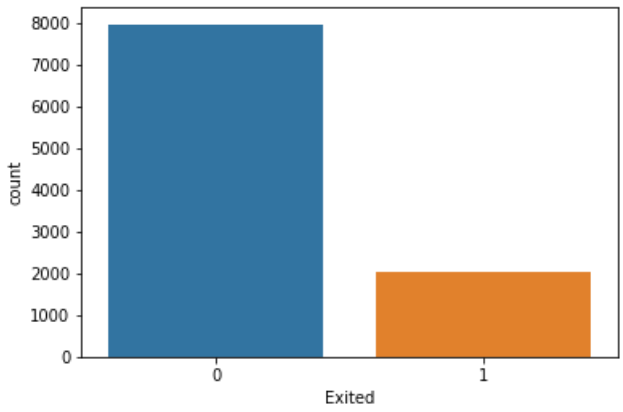


In [50]:

```
sns.countplot(x="Exited",data=dataset)
```

Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1def2093b38>
```

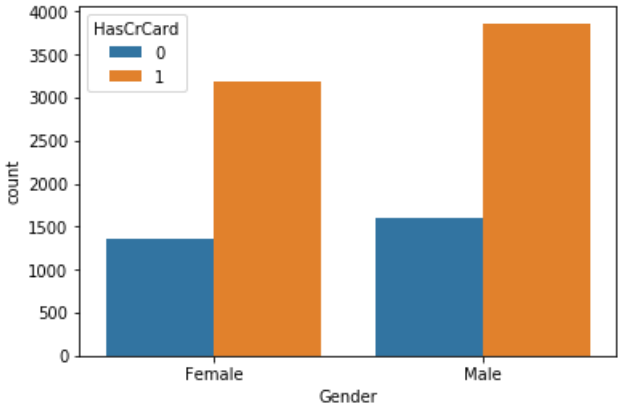


In [37]:

```
sns.countplot(x="Gender",hue="HasCrCard",data=dataset)
```

Out[37]:

<matplotlib.axes._subplots.AxesSubplot at 0x1de129d5b48>



In [44]:

```
X = dataset.iloc[:, 3:13].values
y = dataset.iloc[:, 13].values
```

In [45]:

```
# Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
onehotencoder = OneHotEncoder(categorical_features = [1])
X = onehotencoder.fit_transform(X).toarray()
X = X[:, 1:]
```

In [46]:

```
# Splitting the dataset into the Training set and Test set
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, random_state = 0)
```

In [47]:

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [48]:

```
# Part 2 - Now let's make the ANN!

# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense
```

In [13]:

```
# Initialising the ANN
classifier = Sequential()
```

In [14]:

```
# Adding the input layer and the first hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 11))
```

```
C:\Users\Jayen\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: UserWarning: Update your
```

```
`Dense` call to the Keras 2 API: `Dense(activation="relu", input_dim=11, units=6,
kernel_initializer="uniform")`
```

In [15]:

```
# Adding the input layer and the first hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 11))
```

```
C:\Users\Jayen\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: UserWarning: Update your
`Dense` call to the Keras 2 API: `Dense(activation="relu", input_dim=11, units=6,
kernel_initializer="uniform")`
```

In [16]:

```
# Adding the second hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))
```

```
C:\Users\Jayen\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: UserWarning: Update your
`Dense` call to the Keras 2 API: `Dense(activation="relu", units=6, kernel_initializer="uniform")`
```

In [17]:

```
# Adding the output layer
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
```

```
C:\Users\Jayen\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Update your
`Dense` call to the Keras 2 API: `Dense(activation="sigmoid", units=1,
kernel_initializer="uniform")`
  This is separate from the ipykernel package so we can avoid doing imports until
```

In [18]:

```
# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

In [19]:

```
# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

In [20]:

```
# Fitting the ANN to the Training set
classifier.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)
```

```
C:\Users\Jayen\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: The `nb_epoch` ar
gument in `fit` has been renamed `epochs`.
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
Epoch 1/100
8000/8000 [==============================] - 2s 217us/step - loss: 0.4814 - acc: 0.7957
Epoch 2/100
8000/8000 [==============================] - 1s 88us/step - loss: 0.4277 - acc: 0.7960
Epoch 3/100
8000/8000 [==============================] - 1s 96us/step - loss: 0.4206 - acc: 0.8135
Epoch 4/100
8000/8000 [==============================] - 1s 110us/step - loss: 0.4171 - acc: 0.8275
Epoch 5/100
8000/8000 [==============================] - 1s 156us/step - loss: 0.4150 - acc: 0.8286
Epoch 6/100
8000/8000 [==============================] - 1s 169us/step - loss: 0.4131 - acc: 0.8320
Epoch 7/100
8000/8000 [==============================] - 1s 168us/step - loss: 0.4111 - acc: 0.8337
Epoch 8/100
8000/8000 [==============================] - 1s 144us/step - loss: 0.4106 - acc: 0.8339
```

```
Epoch 9/100
8000/8000 [==============================] - 1s 119us/step - loss: 0.4096 - acc: 0.8326
Epoch 10/100
8000/8000 [==============================] - 1s 116us/step - loss: 0.4085 - acc: 0.8341
Epoch 11/100
8000/8000 [==============================] - 1s 84us/step - loss: 0.4074 - acc: 0.8346
Epoch 12/100
8000/8000 [==============================] - 1s 82us/step - loss: 0.4066 - acc: 0.8356
Epoch 13/100
8000/8000 [==============================] - 1s 78us/step - loss: 0.4067 - acc: 0.8342
Epoch 14/100
8000/8000 [==============================] - 1s 86us/step - loss: 0.4059 - acc: 0.8359
Epoch 15/100
8000/8000 [==============================] - 1s 79us/step - loss: 0.4055 - acc: 0.8339
Epoch 16/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4056 - acc: 0.8357
Epoch 17/100
8000/8000 [==============================] - 1s 77us/step - loss: 0.4047 - acc: 0.8349
Epoch 18/100
8000/8000 [==============================] - 1s 96us/step - loss: 0.4048 - acc: 0.8355
Epoch 19/100
8000/8000 [==============================] - 1s 100us/step - loss: 0.4044 - acc: 0.8359
Epoch 20/100
8000/8000 [==============================] - 1s 106us/step - loss: 0.4038 - acc: 0.8360
Epoch 21/100
8000/8000 [==============================] - 1s 82us/step - loss: 0.4034 - acc: 0.8346
Epoch 22/100
8000/8000 [==============================] - 1s 96us/step - loss: 0.4039 - acc: 0.8350
Epoch 23/100
8000/8000 [==============================] - 1s 94us/step - loss: 0.4036 - acc: 0.8349
Epoch 24/100
8000/8000 [==============================] - 1s 84us/step - loss: 0.4034 - acc: 0.8364
Epoch 25/100
8000/8000 [==============================] - 1s 72us/step - loss: 0.4033 - acc: 0.8341
Epoch 26/100
8000/8000 [==============================] - 1s 107us/step - loss: 0.4034 - acc: 0.8351
Epoch 27/100
8000/8000 [==============================] - 1s 82us/step - loss: 0.4028 - acc: 0.8361
Epoch 28/100
8000/8000 [==============================] - 1s 77us/step - loss: 0.4032 - acc: 0.8330
Epoch 29/100
8000/8000 [==============================] - 1s 110us/step - loss: 0.4025 - acc: 0.8344
Epoch 30/100
8000/8000 [==============================] - 1s 88us/step - loss: 0.4023 - acc: 0.8357
Epoch 31/100
8000/8000 [==============================] - 1s 137us/step - loss: 0.4028 - acc: 0.8332 0s - loss:
0.3965 - ac
Epoch 32/100
8000/8000 [==============================] - 1s 113us/step - loss: 0.4027 - acc: 0.8341
Epoch 33/100
8000/8000 [==============================] - 1s 95us/step - loss: 0.4026 - acc: 0.8365
Epoch 34/100
8000/8000 [==============================] - 1s 75us/step - loss: 0.4024 - acc: 0.8331
Epoch 35/100
8000/8000 [==============================] - 1s 75us/step - loss: 0.4020 - acc: 0.8351
Epoch 36/100
8000/8000 [==============================] - 1s 121us/step - loss: 0.4021 - acc: 0.8340
Epoch 37/100
8000/8000 [==============================] - 1s 75us/step - loss: 0.4019 - acc: 0.8344
Epoch 38/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4014 - acc: 0.8350
Epoch 39/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4020 - acc: 0.8331
Epoch 40/100
8000/8000 [==============================] - 1s 75us/step - loss: 0.4019 - acc: 0.8346
Epoch 41/100
8000/8000 [==============================] - 1s 72us/step - loss: 0.4023 - acc: 0.8356
Epoch 42/100
8000/8000 [==============================] - 1s 75us/step - loss: 0.4020 - acc: 0.8369
Epoch 43/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4015 - acc: 0.8336
Epoch 44/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4017 - acc: 0.8362
Epoch 45/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4015 - acc: 0.8370
Epoch 46/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4011 - acc: 0.8346
```

```
Epoch 47/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4015 - acc: 0.8349
Epoch 48/100
8000/8000 [==============================] - 1s 72us/step - loss: 0.4015 - acc: 0.8352
Epoch 49/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4017 - acc: 0.8337
Epoch 50/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4007 - acc: 0.8342
Epoch 51/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4012 - acc: 0.8360
Epoch 52/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4014 - acc: 0.8340
Epoch 53/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4016 - acc: 0.8329
Epoch 54/100
8000/8000 [==============================] - 1s 72us/step - loss: 0.4009 - acc: 0.8364
Epoch 55/100
8000/8000 [==============================] - 1s 71us/step - loss: 0.4011 - acc: 0.8367
Epoch 56/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4011 - acc: 0.8359
Epoch 57/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4011 - acc: 0.8346
Epoch 58/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4012 - acc: 0.8360
Epoch 59/100
8000/8000 [==============================] - 1s 75us/step - loss: 0.4007 - acc: 0.8357
Epoch 60/100
8000/8000 [==============================] - 1s 71us/step - loss: 0.4006 - acc: 0.8352
Epoch 61/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4008 - acc: 0.8356
Epoch 62/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4010 - acc: 0.8366
Epoch 63/100
8000/8000 [==============================] - 1s 71us/step - loss: 0.4014 - acc: 0.8354
Epoch 64/100
8000/8000 [==============================] - 1s 75us/step - loss: 0.4004 - acc: 0.8354
Epoch 65/100
8000/8000 [==============================] - 1s 72us/step - loss: 0.4007 - acc: 0.8367
Epoch 66/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4004 - acc: 0.8340
Epoch 67/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4007 - acc: 0.8359
Epoch 68/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4010 - acc: 0.8352
Epoch 69/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4004 - acc: 0.8350
Epoch 70/100
8000/8000 [==============================] - 1s 73us/step - loss: 0.4004 - acc: 0.8356
Epoch 71/100
8000/8000 [==============================] - 1s 72us/step - loss: 0.4007 - acc: 0.8362
Epoch 72/100
8000/8000 [==============================] - 1s 75us/step - loss: 0.4006 - acc: 0.8349
Epoch 73/100
8000/8000 [==============================] - ETA: 0s - loss: 0.4014 - acc: 0.834 - 1s 107us/step -
loss: 0.4009 - acc: 0.8344
Epoch 74/100
8000/8000 [==============================] - 1s 107us/step - loss: 0.4007 - acc: 0.8354
Epoch 75/100
8000/8000 [==============================] - 1s 110us/step - loss: 0.4008 - acc: 0.8350
Epoch 76/100
8000/8000 [==============================] - 1s 83us/step - loss: 0.4002 - acc: 0.8355
Epoch 77/100
8000/8000 [==============================] - 1s 115us/step - loss: 0.4010 - acc: 0.8370
Epoch 78/100
8000/8000 [==============================] - 1s 125us/step - loss: 0.4009 - acc: 0.8359
Epoch 79/100
8000/8000 [==============================] - 1s 138us/step - loss: 0.4003 - acc: 0.8352
Epoch 80/100
8000/8000 [==============================] - 1s 159us/step - loss: 0.4004 - acc: 0.8350 0s - loss:
0.
Epoch 81/100
8000/8000 [==============================] - 1s 113us/step - loss: 0.4008 - acc: 0.8342
Epoch 82/100
8000/8000 [==============================] - 1s 84us/step - loss: 0.4004 - acc: 0.8364
Epoch 83/100
8000/8000 [==============================] - 1s 77us/step - loss: 0.4002 - acc: 0.8370
Epoch 84/100
```

```
8000/8000 [==============================] - 1s 73us/step - loss: 0.4003 - acc: 0.8346
Epoch 85/100
8000/8000 [==============================] - 1s 75us/step - loss: 0.4008 - acc: 0.8347: 0s - loss:
0.3991 - ac
Epoch 86/100
8000/8000 [==============================] - 1s 75us/step - loss: 0.4005 - acc: 0.8361
Epoch 87/100
8000/8000 [==============================] - 1s 131us/step - loss: 0.4008 - acc: 0.8359
Epoch 88/100
8000/8000 [==============================] - 1s 140us/step - loss: 0.4002 - acc: 0.8367
Epoch 89/100
8000/8000 [==============================] - 1s 109us/step - loss: 0.4006 - acc: 0.8354
Epoch 90/100
8000/8000 [==============================] - 1s 107us/step - loss: 0.4000 - acc: 0.8359
Epoch 91/100
8000/8000 [==============================] - 1s 103us/step - loss: 0.4003 - acc: 0.8361
Epoch 92/100
8000/8000 [==============================] - 1s 123us/step - loss: 0.3998 - acc: 0.8342
Epoch 93/100
8000/8000 [==============================] - 1s 83us/step - loss: 0.4001 - acc: 0.8367
Epoch 94/100
8000/8000 [==============================] - 1s 117us/step - loss: 0.4006 - acc: 0.8346
Epoch 95/100
8000/8000 [==============================] - 1s 79us/step - loss: 0.4003 - acc: 0.8364
Epoch 96/100
8000/8000 [==============================] - 1s 119us/step - loss: 0.4005 - acc: 0.8347
Epoch 97/100
8000/8000 [==============================] - 1s 105us/step - loss: 0.4006 - acc: 0.8355
Epoch 98/100
8000/8000 [==============================] - 1s 118us/step - loss: 0.4006 - acc: 0.8355
Epoch 99/100
8000/8000 [==============================] - 1s 91us/step - loss: 0.3998 - acc: 0.8349
Epoch 100/100
8000/8000 [==============================] - 1s 148us/step - loss: 0.4005 - acc: 0.8350
```

Out[20]:

```
<keras.callbacks.History at 0x1def19372b0>
```

In [21]:

```
# Part 3 - Making the predictions and evaluating the model
# Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
```

In [22]:

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [23]:

```
cm
```

Out[23]:

```
array([[1546,   49],
       [ 270,  135]], dtype=int64)
```

In [26]:

```
classifier.evaluate(X,y)
```

```
10000/10000 [==============================] - 0s 30us/step
```

Out[26]:

```
[3.2832560495376586, 0.7963]
```