In [ ]:

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

Loading the dataset

In [ ]:

```
df=pd.read_csv('/content/Churn_Modelling.csv')
```

Visualization

In [ ]:

```
df.head(10)
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	
7	8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	
8	9	15792365	He	501	France	Male	44	4	142051.07	2	
9	10	15592389	H?	684	France	Male	27	2	134603.88	1	

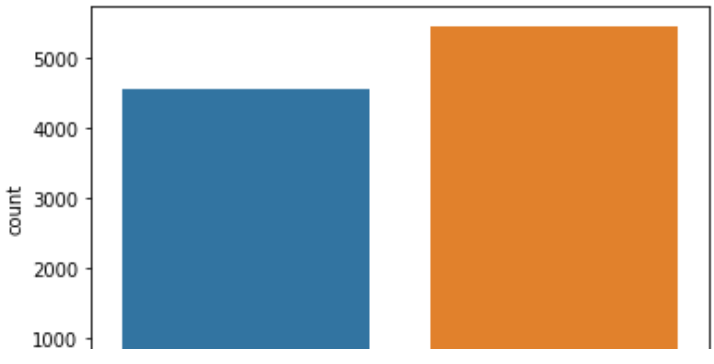
In [ ]:

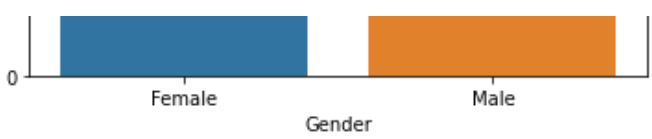
```
sns.countplot(df.Gender)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fac8ae5e290>
```



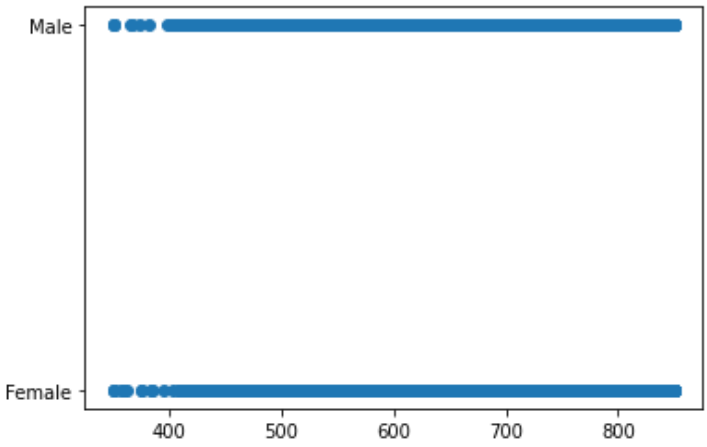


In [ ]:

```
plt.scatter(df.CreditScore,df.Gender)
```

Out[ ]:

```
<matplotlib.collections.PathCollection at 0x7fac8b3d7350>
```



In [ ]:

```
sns.pairplot(df,hue="Gender",size=3)
```

Description Statistics on the dataset

In [ ]:

```
df.describe()
```

Out[ ]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	Is
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	

Handle the Missing values

In [ ]:

```
df.info
```

Out[ ]:

```
<bound method DataFrame.info of       RowNumber  CustomerId    Surname  CreditScore Geogr
aphy  Gender  Age  \
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
...         ...         ...        ...          ...       ...     ...  ...
9995       9996    15606229   Obijiaku          771    France    Male   39
9996       9997    15569892  Johnstone          516    France    Male   35
9997       9998    15584532        Liu          709    France  Female   36
9998       9999    15682355  Sabbatini          772   Germany    Male   42
9999      10000    15628319     Walker          792    France  Female   28

      Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0          2       0.00              1          1               1
1          1   83807.86              1          0               1
2          8  159660.80              3          1               0
3          1       0.00              2          0               0
4          2  125510.82              1          1               1
...      ...        ...            ...        ...             ...
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0

      EstimatedSalary  Exited
0           101348.88       1
1           112542.58       0
2           113931.57       1
3            93826.63       0
4            79084.10       0
...               ...     ...
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0

[10000 rows x 14 columns]>
```

In [ ]:

```
df.isnull().sum()
```

Out[ ]:

```
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 [ ]:

```
df.isna()
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrC
0	False	False	False	False	False	False	False	False	False	False	Fa
1	False	False	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	False	Fa

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrC
3	False	False	False	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	False	False	False	Fa
...	...	...	...	...	...	...	...	...	...	...	
9995	False	False	False	False	False	False	False	False	False	False	Fa
9996	False	False	False	False	False	False	False	False	False	False	Fa
9997	False	False	False	False	False	False	False	False	False	False	Fa
9998	False	False	False	False	False	False	False	False	False	False	Fa
9999	False	False	False	False	False	False	False	False	False	False	Fa

10000 rows × 14 columns

In [ ]:

```
df.notnull()
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCa
0	True	True	True	True	True	True	True	True	True	True	Tr
1	True	True	True	True	True	True	True	True	True	True	Tr
2	True	True	True	True	True	True	True	True	True	True	Tr
3	True	True	True	True	True	True	True	True	True	True	Tr
4	True	True	True	True	True	True	True	True	True	True	Tr
...	...	...	...	...	...	...	...	...	...	...	
9995	True	True	True	True	True	True	True	True	True	True	Tr
9996	True	True	True	True	True	True	True	True	True	True	Tr
9997	True	True	True	True	True	True	True	True	True	True	Tr
9998	True	True	True	True	True	True	True	True	True	True	Tr
9999	True	True	True	True	True	True	True	True	True	True	Tr

10000 rows × 14 columns

In [ ]:

```
df.notna()
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCa
0	True	True	True	True	True	True	True	True	True	True	Tr
1	True	True	True	True	True	True	True	True	True	True	Tr
2	True	True	True	True	True	True	True	True	True	True	Tr
3	True	True	True	True	True	True	True	True	True	True	Tr
4	True	True	True	True	True	True	True	True	True	True	Tr
...	...	...	...	...	...	...	...	...	...	...	
9995	True	True	True	True	True	True	True	True	True	True	Tr
9996	True	True	True	True	True	True	True	True	True	True	Tr
9997	True	True	True	True	True	True	True	True	True	True	Tr
9998	True	True	True	True	True	True	True	True	True	True	Tr
9999	True	True	True	True	True	True	True	True	True	True	Tr

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCa

10000 rows × 14 columns

Check the categorial columns and perform encoding

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
```

In [ ]:

```
le= LabelEncoder()
```

In [ ]:

```
df['Geography']=le.fit_transform(df['Geography'])
df.head()
```

Out[ ]:

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

Split the data into dependent and independent variable

In [ ]:

```
x=df.iloc[:,0:13].values
x
```

Out[ ]:

```
array([[1, 15634602, 'Hargrave', ..., 1, 1, 101348.88],
       [2, 15647311, 'Hill', ..., 0, 1, 112542.58],
       [3, 15619304, 'Onio', ..., 1, 0, 113931.57],
       ...,
       [9998, 15584532, 'Liu', ..., 0, 1, 42085.58],
       [9999, 15682355, 'Sabbatini', ..., 1, 0, 92888.52],
       [10000, 15628319, 'Walker', ..., 1, 0, 38190.78]], dtype=object)
```

In [ ]:

```
y=df.iloc[:,13:].values
y
```

Out[ ]:

```
array([[1],
       [0],
       [1],
       ...,
       [1],
       [1],
       [0]])
```

Scale the idependent variable

In [ ]:

```
from sklearn.preprocessing import StandardScaler
```

In [ ]:

```
ss=StandardScaler()
```

In [ ]:

```
y=ss.fit_transform(y)
```

In [ ]:

```
df.head()
```

Split tranining and testing data

In [ ]:

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

In [ ]:

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=(0.33),random_state=42)
```

In [ ]:

```
xtrain.shape,xtest.shape
```

Out[ ]:

```
((6700, 13), (3300, 13))
```