```
import numpy as np
```

```
import pandas as pd
```

```
df=pd.read_csv("Churn_Modelling.csv")
```

```
df
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
0	1	15634602	Hargrave	619	France	Female	42	2
1	2	15647311	Hill	608	Spain	Female	41	1
2	3	15619304	Onio	502	France	Female	42	8
3	4	15701354	Boni	699	France	Female	39	1
4	5	15737888	Mitchell	850	Spain	Female	43	2
...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5
9996	9997	15569892	Johnstone	516	France	Male	35	10
9997	9998	15584532	Liu	709	France	Female	36	7
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3
9999	10000	15628319	Walker	792	France	Female	28	4

10000 rows × 14 columns

```
import matplotlib.pyplot as plt
import seaborn as sns
```

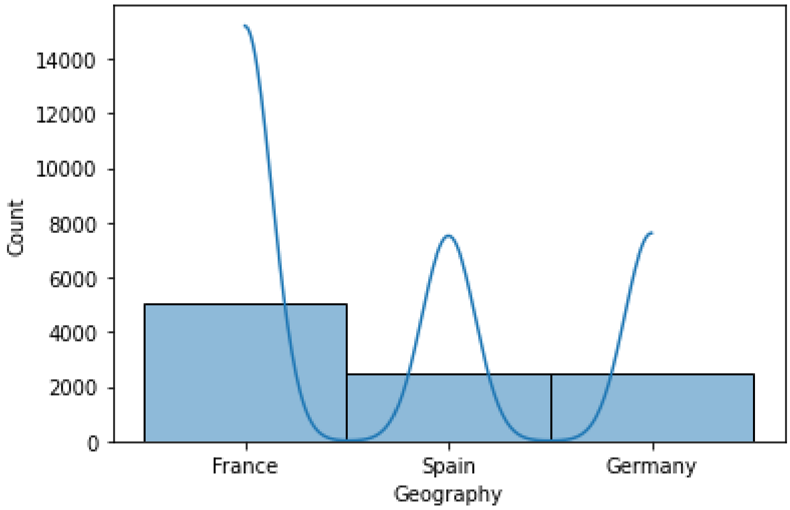
```
%matplotlib inline
```

```
df[['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure']].describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800

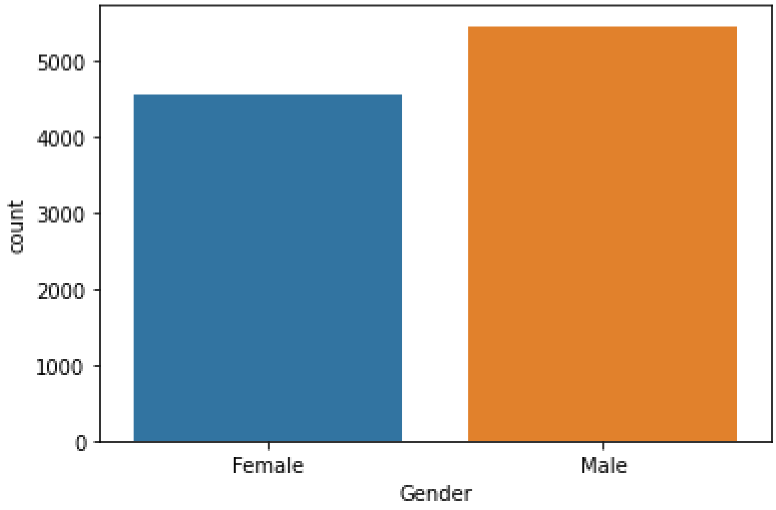
```
sns.histplot(df.Geography,kde= True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f08d7826750>
```



```
# plot count plot for the gender column
sns.countplot(df.Gender)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: P
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f08d76ef610>
```

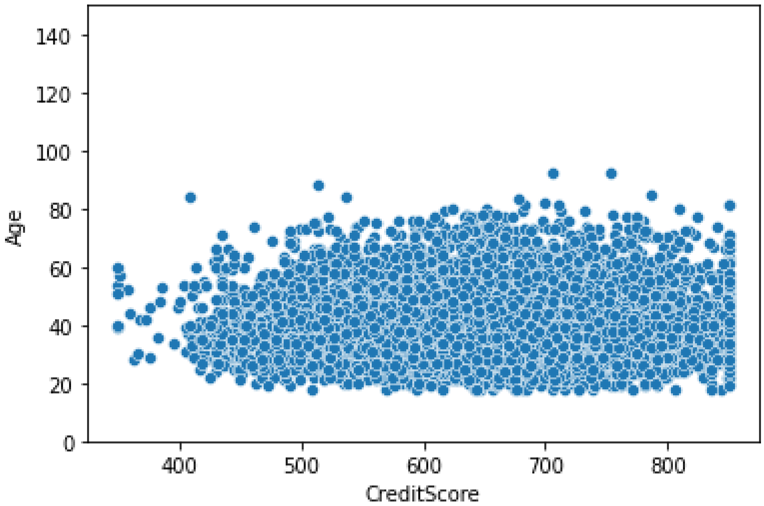


```
df[['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure']].corr()
```

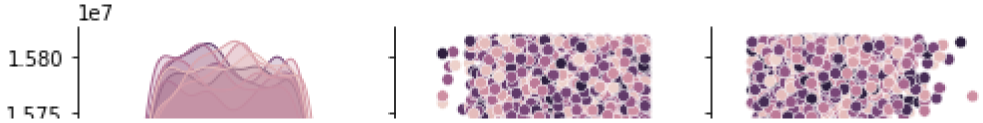
	RowNumber	CustomerId	CreditScore	Age	Tenure
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883

```
sns.scatterplot(df.CreditScore,df.Age)
plt.ylim(0,150)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: P
  FutureWarning
(0.0, 150.0)
```



```
sns.pairplot(data=df[['CustomerId', 'Surname', 'CreditScore', 'Geography','Gender', 'Age',
<seaborn.axisgrid.PairGrid at 0x7f08d7202250>
```



```
# summary statistics
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Ba
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.0
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.8
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.4
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.0
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.0
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.5
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.2
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.0

```
df.dtypes
```

```
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
```

```
# mode
df['Age'].mode()
```

```
0    37
dtype: int64
```

```
# calculation of the mean
df["Age"].mean()
```

```
38.9218
```

```
# calculation of the mean and round the result
round(df["Age"].mean(), 3)
```

```
38.922
```

```
# calculation of the median
df["Age"].median()
```

```
37.0
```

```
df.isna().any()
```

```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

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

```
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
```

```
df.isnull()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenur
0	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	.
9995	False	False	False	False	False	False	False	Fals
9996	False	False	False	False	False	False	False	Fals
9997	False	False	False	False	False	False	False	Fals
9998	False	False	False	False	False	False	False	Fals
9999	False	False	False	False	False	False	False	Fals

10000 rows × 14 columns

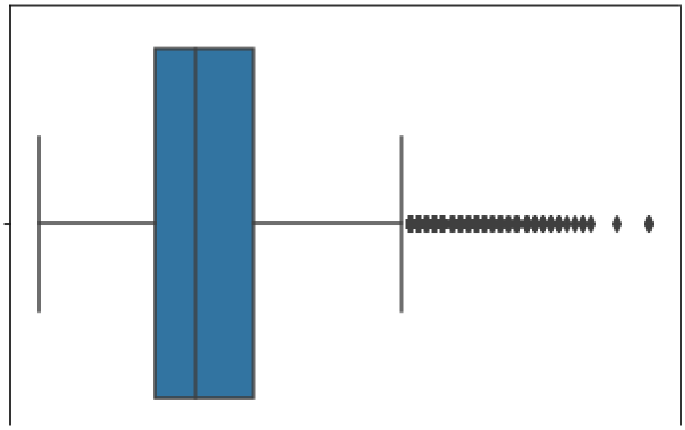
```
df.notnull()
```

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

10000 rows × 14 columns

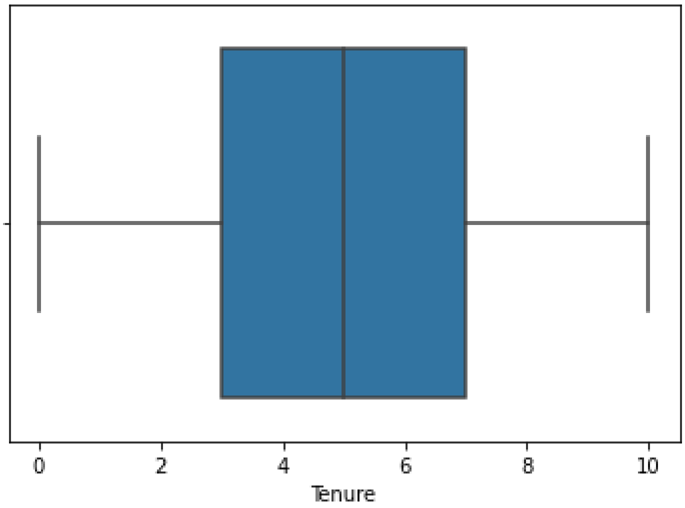
```
sns.boxplot(x=df['Age'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f08d6ffcd10>
```



```
sns.boxplot(x=df['Tenure'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f08d719f550>
```



```
a=df.columns
```

```
b=df._get_numeric_data().columns
```

```
b
```

```
Index(['RowNumber', 'CustomerId', 'CreditScore', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary',
       'Exited'],
      dtype='object')
```

```
list(set(a) - set(b))
```

```
['Geography', 'Surname', 'Gender']
```

```
# x  -Independent
# y  -Dependent
x =df.drop('Exited',axis=1)
y=df['Exited']
```

```
x.head()
```

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

```
y.head()
```

```
0    1
1    0
2    1
3    0
4    0
Name: Exited, dtype: int64
```

```
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
scale = StandardScaler()
x=df[['Age','Tenure']]
scaledx = scale.fit_transform(x)
print(scaledx)
```

```
[[ 0.29351742 -1.04175968]
 [ 0.19816383 -1.38753759]
 [ 0.29351742  1.03290776]
 ...
 [-0.27860412  0.68712986]
 [ 0.29351742 -0.69598177]
 [-1.04143285 -0.35020386]]
```

```
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)
```

```
print('X Train shape:{},Y.Train SHape:{}'.format(x_train.shape,y_train.shape))
```

```
X Train shape:(8000, 2),Y.Train SHape:(8000,)
```

```
print('X Test Shape :{},Y Test SHape:{}'.format(x_test.shape,y_test.shape))
```

```
X Test Shape :(2000, 2),Y Test SHape:(2000,)
```