```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 %matplotlib inline
5 import seaborn as sns
6 import math
```

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

```
1 df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	France	Female	42	2	0.(
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8
3	4	15701354	Boni	699	France	Female	39	1	0.(
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8

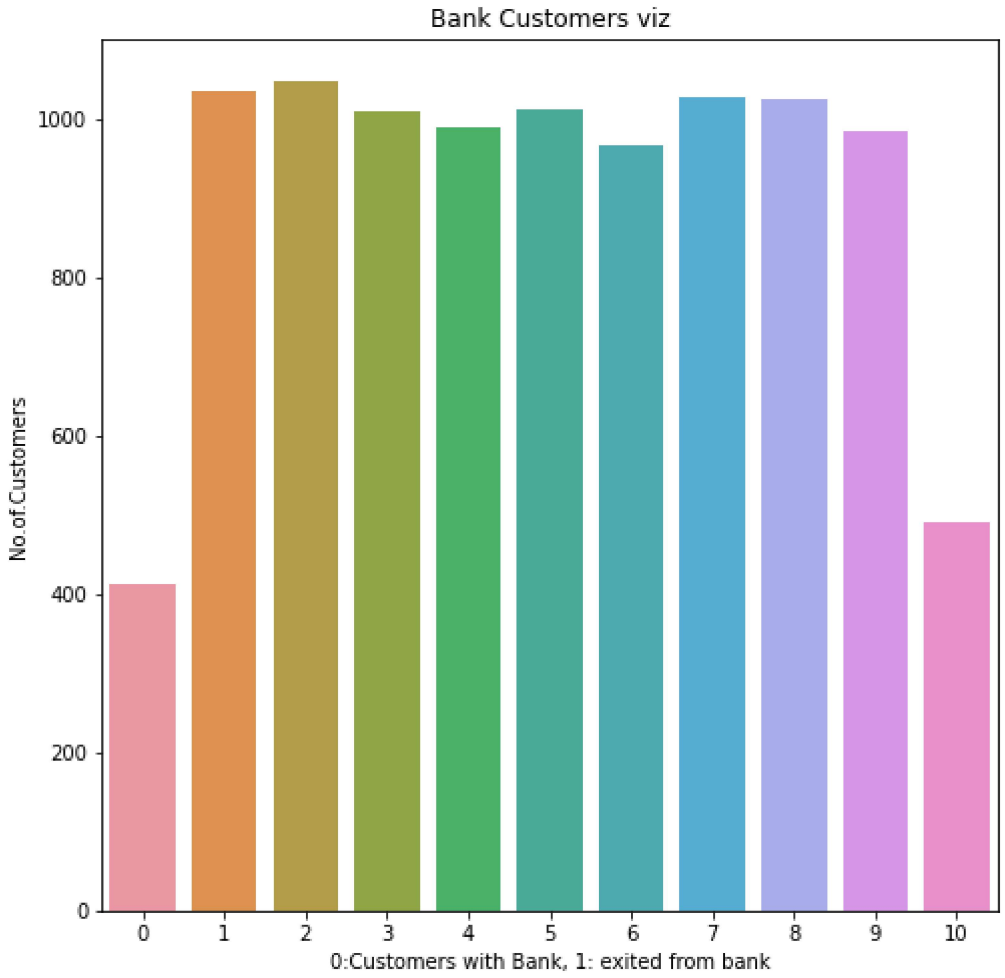
```
1 df.drop(["RowNumber","CustomerId","Surname"],axis=1,inplace=True)
```

```
1 df.info()
2
```

```
<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
```

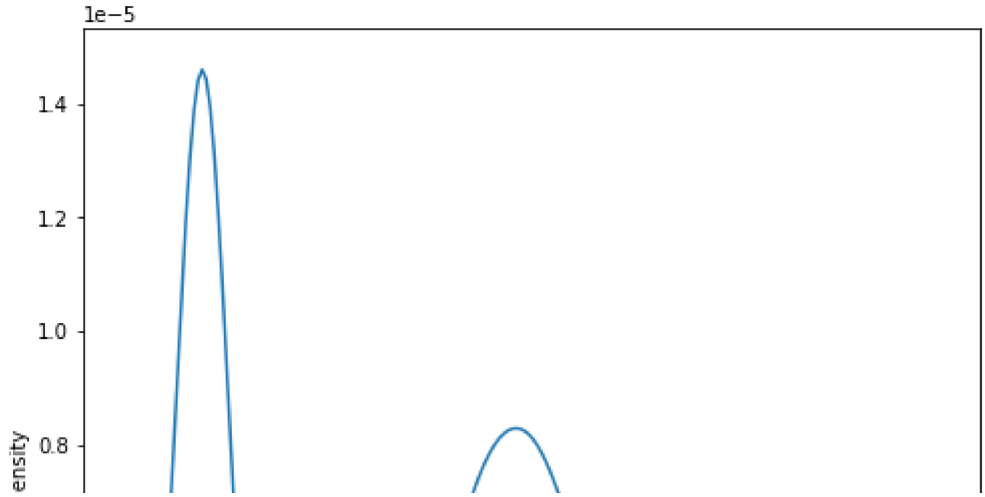
```
1 #Perform Univariate Analysis
2 plt.figure(figsize=(8,8))
```

```
sns.countplot(x='Tenure',data=df)
plt.xlabel('0:Customers with Bank, 1: exited from bank')
plt.ylabel('No.of.Customers')
plt.title("Bank Customers viz")
plt.show()
```



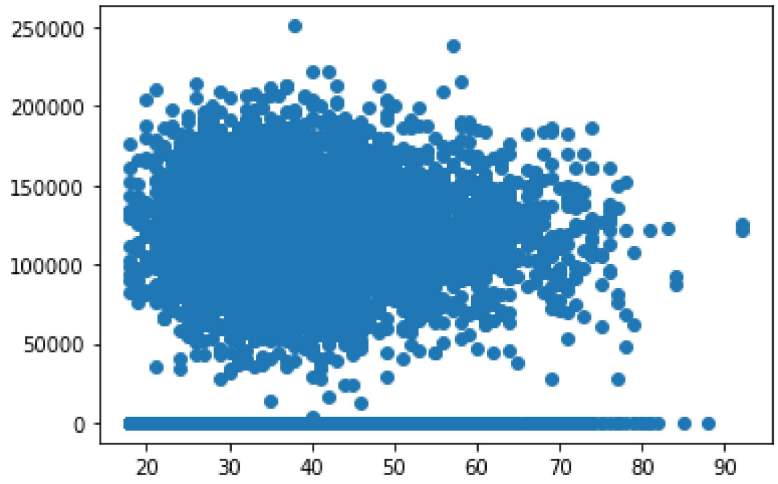

```
#Perform Univariate Analysis
plt.figure(figsize=(8,8))
sns.kdeplot(x=df['Balance'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d77d59550>
```




```
1 #Perform Bivariate Analysis
2 plt.scatter(df.Age,df.Balance)
```

```
<matplotlib.collections.PathCollection at 0x7f3d77c43b10>
```




```
1 #Perform Bivariate Analysis
2 df.corr()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	Is
CreditScore	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	
Age	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	

```
#Perform Bivariate Analysis
import statsmodels.api as sm

#define response variable
y = df['CreditScore']

#define explanatory variable
x = df[['EstimatedSalary']]

#add constant to predictor variables
x = sm.add_constant(x)

#fit linear regression model
model = sm.OLS(y, x).fit()

#view model summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            CreditScore   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.000
Method:                 Least Squares   F-statistic:                    0.01916
Date:                Wed, 28 Sep 2022   Prob (F-statistic):              0.890
Time:                        10:09:48   Log-Likelihood:                -59900.
No. Observations:               10000   AIC:                         1.198e+05
Df Residuals:                    9998   BIC:                         1.198e+05
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            650.7617      1.940    335.407      0.000     646.958     654.565
EstimatedSalary -2.326e-06   1.68e-05     -0.138      0.890   -3.53e-05    3.06e-05
==============================================================================
Omnibus:                      132.939   Durbin-Watson:                   2.014
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               84.242
Skew:                          -0.072   Prob(JB):                     5.10e-19
Kurtosis:                       2.574   Cond. No.                     2.32e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
[2] The condition number is large, 2.32e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142: FutureWarning: I
  x = pd.concat(x[::order], 1)
```

```
#Perform Multivariate Analysis
plt.figure(figsize=(4,4))
sns.pairplot(data=df[["Balance","CreditScore","EstimatedSalary","NumOfProducts","Tenure","
```

```
<seaborn.axisgrid.PairGrid at 0x7f3d79a25210>
<Figure size 288x288 with 0 Axes>
```


```
1 #Perform Descriptive Statistics
2 df=pd.DataFrame(df)
3 print(df.sum())
```

```
CreditScore                                                6505288
Geography          FranceSpainFranceFranceSpainSpainFranceGermany...
Gender             FemaleFemaleFemaleFemaleFemaleMaleMaleFemaleMa...
Age                                                         389218
Tenure                                                       50128
Balance                                                764858892.88
NumOfProducts                                                15302
HasCrCard                                                     7055
IsActiveMember                                                5151
EstimatedSalary                                        1000902398.81
Exited                                                        2037
dtype: object
```


```
1 #Perform Descriptive Statistics
2 print("----Sum Value-----")
3 print(df.sum(1))
4 print("---------------------------------")
5 print("-----Product Value-----")
6 print(df.prod())
7 print("---------------------------------")
```

```
----Sum Value-----
0       102015.88
1       197002.44
2       274149.37
3        94567.63
4       205492.92
           ...
9995     97088.64
9996    159633.38
9997     42840.58
9998    168784.83
9999    169159.57
Length: 10000, dtype: float64
---------------------------------
-----Product Value-----
CreditScore        0.0
Age                0.0
Tenure             0.0
Balance            0.0
NumOfProducts      0.0
HasCrCard          0.0
IsActiveMember     0.0
EstimatedSalary    inf
Exited             0.0
dtype: float64
```

```
----------------------------------
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: FutureWarning: Dropping
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.7/dist-packages/numpy/core/_methods.py:52: RuntimeWarning: overfl
  return umr_prod(a, axis, dtype, out, keepdims, initial, where)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: FutureWarning: Dropping
```

```
1 #Handling with missing Values
2 df.isnull()#Checking values are null
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
9995	False	False	False	False	False	False	False	False
9996	False	False	False	False	False	False	False	False
9997	False	False	False	False	False	False	False	False
9998	False	False	False	False	False	False	False	False
9999	False	False	False	False	False	False	False	False

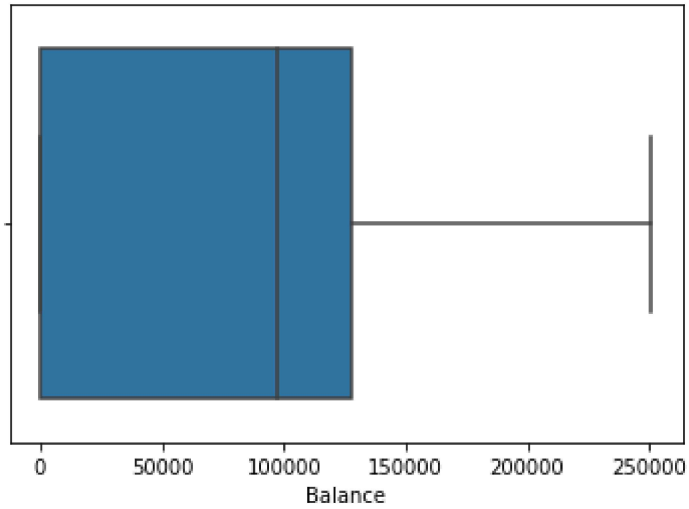
10000 rows × 11 columns

```
1 #Handling with missing Values
2 df.notnull()#Checking values are not null
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	I
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	

```
1 #Find outliers & replace the outliers
2 sns.boxplot(df['Balance'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d69c562d0>
```




```
1 #Find outliers & replace the outliers
2 print(np.where(df['Balance']>100000))
```

```
(array([   2,    4,    5, ..., 9987, 9993, 9999]),)
```

```
1 #Find outliers & replace the outliers
2 from scipy import stats
3 import numpy as np
4
5 z = np.abs(stats.zscore(df["EstimatedSalary"]))
6 print(z)
```

```
0       0.021886
1       0.216534
2       0.240687
3       0.108918
4       0.365276
          ...
9995    0.066419
9996    0.027988
```

```
9997    1.008643
9998    0.125231
9999    1.076370
Name: EstimatedSalary, Length: 10000, dtype: float64
```

```
1 #Check for categorical columns & performs encoding
2 from sklearn.preprocessing import LabelEncoder
3 df['Gender'].unique()
```

```
array(['Female', 'Male'], dtype=object)
```

```
1 #Check for categorical columns & performs encoding
2 df['Gender'].value_counts()
```

```
Male      5457
Female    4543
Name: Gender, dtype: int64
```

```
1 #Check for categorical columns & performs encoding
2 encoding=LabelEncoder()
3 df["Gender"]=encoding.fit_transform(df.iloc[:,1].values)
4 df
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	619	France	0	42	2	0.00	1	1
1	608	Spain	2	41	1	83807.86	1	0
2	502	France	0	42	8	159660.80	3	1
3	699	France	0	39	1	0.00	2	0
4	850	Spain	2	43	2	125510.82	1	1
...	...	...	...	...	...	...	...	...
9995	771	France	0	39	5	0.00	2	1
9996	516	France	0	35	10	57369.61	1	1
9997	709	France	0	36	7	0.00	1	0
9998	772	Germany	1	42	3	75075.31	2	1
9999	792	France	0	28	4	130142.79	1	1

10000 rows × 11 columns

```
1 #Split the data into Dependent & Independent Variables
```

```
print("----------Dependent Variables----------")
X=df.iloc[:,1:4]
print(X)
print("---------------------------------------")
print("---------Independent Variables---------")
Y=df.iloc[:,4]
print(Y)
print("---------------------------------------")
```

```
----------Dependent Variables----------
     Geography  Gender  Age
0       France       0   42
1        Spain       2   41
2       France       0   42
3       France       0   39
4        Spain       2   43
...        ...     ...  ...
9995    France       0   39
9996    France       0   35
9997    France       0   36
9998   Germany       1   42
9999    France       0   28

[10000 rows x 3 columns]
---------------------------------------
---------Independent Variables---------
0        2
1        1
2        8
3        1
4        2
        ..
9995     5
9996    10
9997     7
9998     3
9999     4
Name: Tenure, Length: 10000, dtype: int64
---------------------------------------
```

```
#Split the data into training & testing
from sklearn.model_selection import train_test_split
```

```
#Split the data into training & testing
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=4,random_state=4)
x_train
```

	const	EstimatedSalary
2558	1.0	137903.54
7642	1.0	121765.00
8912	1.0	109470.34
3319	1.0	2923.61
6852	1.0	7312.25
...	...	...
456	1.0	7666.73
6017	1.0	9085.00
709	1.0	147794.63

```
1 #Split the data into training & testing
2 x_test
```

	const	EstimatedSalary
1603	1.0	23305.85
8713	1.0	41248.80
4561	1.0	143317.42
6600	1.0	174123.16

```
1 #Split the data into training & testing
2 y_train
```

```
2558    727
7642    811
8912    623
3319    430
6852    600
       ...
456     733
6017    487
709     686
8366    637
1146    614
Name: CreditScore, Length: 9996, dtype: int64
```

```
1 #Split the data into training & testing
2 x_test
```

	const	EstimatedSalary
1603	1.0	23305.85
8713	1.0	41248.80
4561	1.0	140017.42

```
#Split the data into training & testing
y_test
```

```
1603    576
8713    786
4561    562
6600    505
Name: CreditScore, dtype: int64
```