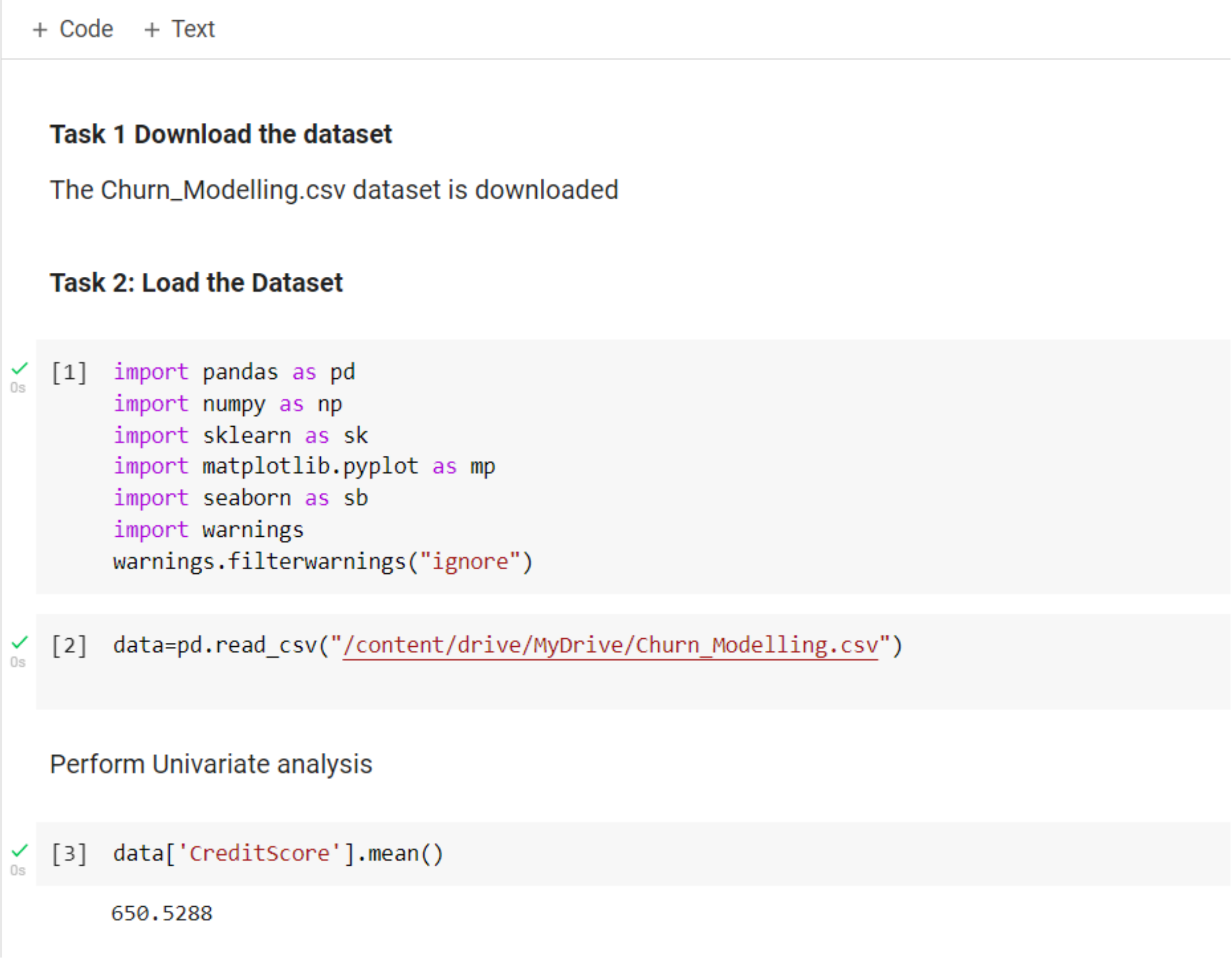
ASSIGNMENT 2



Task 1 Download the dataset

The Churn_Modelling.csv dataset is downloaded

Task 2: Load the Dataset

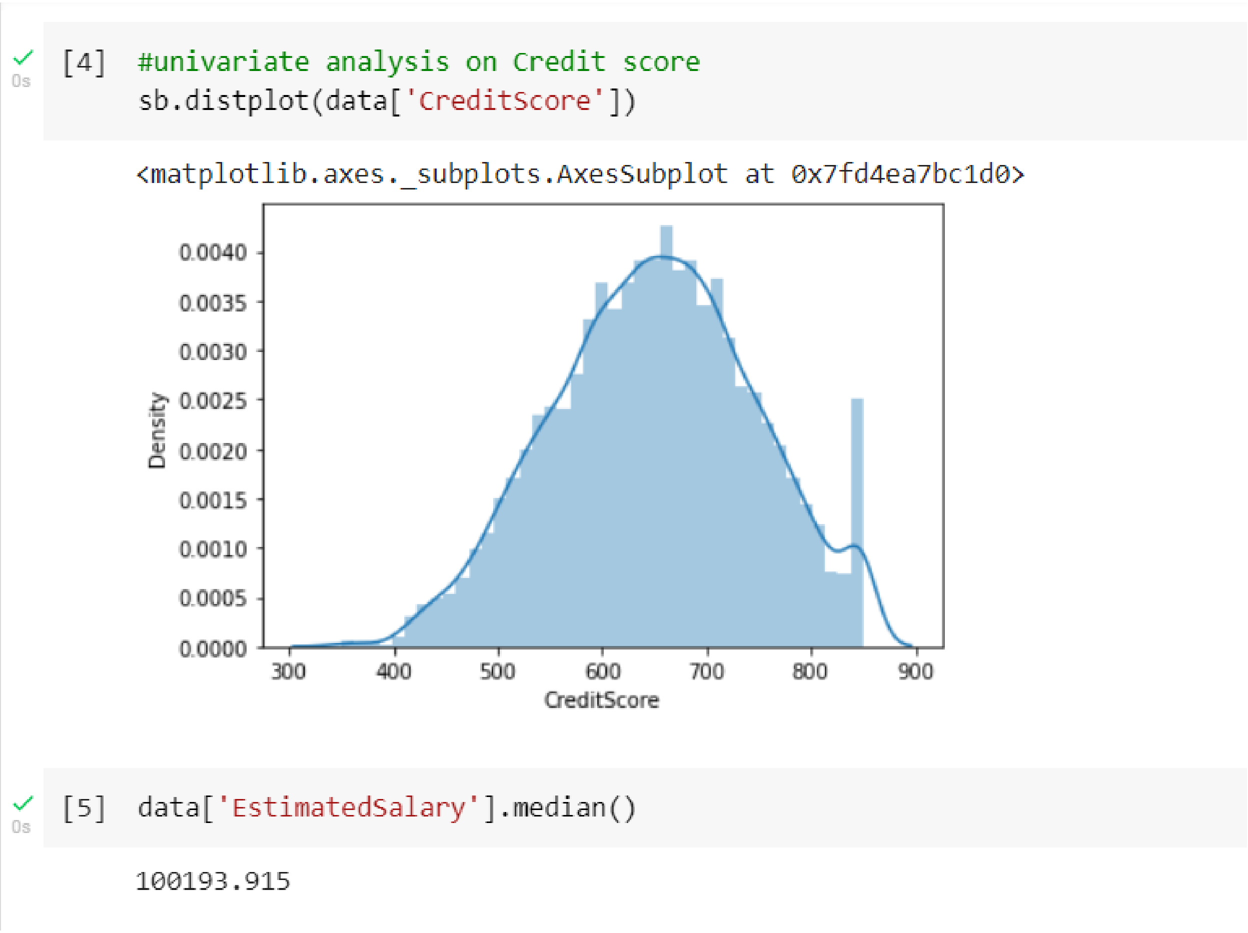
```
[1] import pandas as pd
    import numpy as np
    import sklearn as sk
    import matplotlib.pyplot as mp
    import seaborn as sb
    import warnings
    warnings.filterwarnings("ignore")
```

```
[2] data=pd.read_csv("/content/drive/MyDrive/Churn_Modelling.csv")
```

Perform Univariate analysis

```
[3] data['CreditScore'].mean()
```

```
650.5288
```

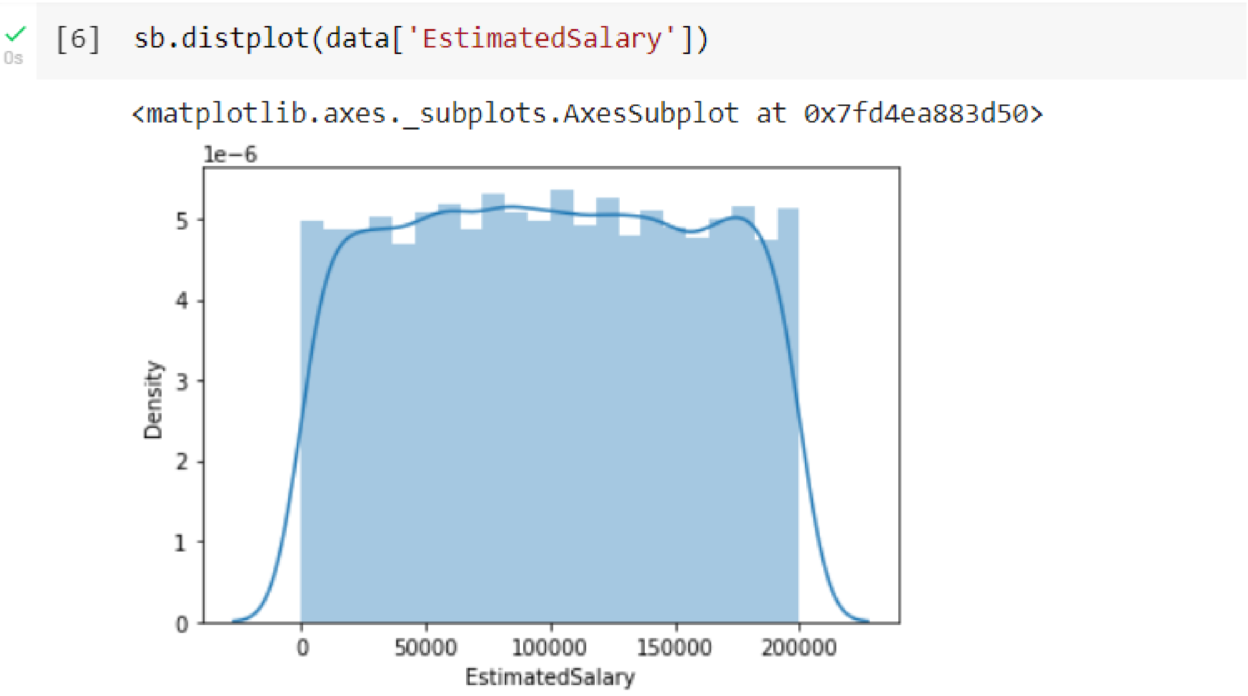


```
[4] #univariate analysis on Credit score
    sb.distplot(data['CreditScore'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd4ea7bc1d0>
```

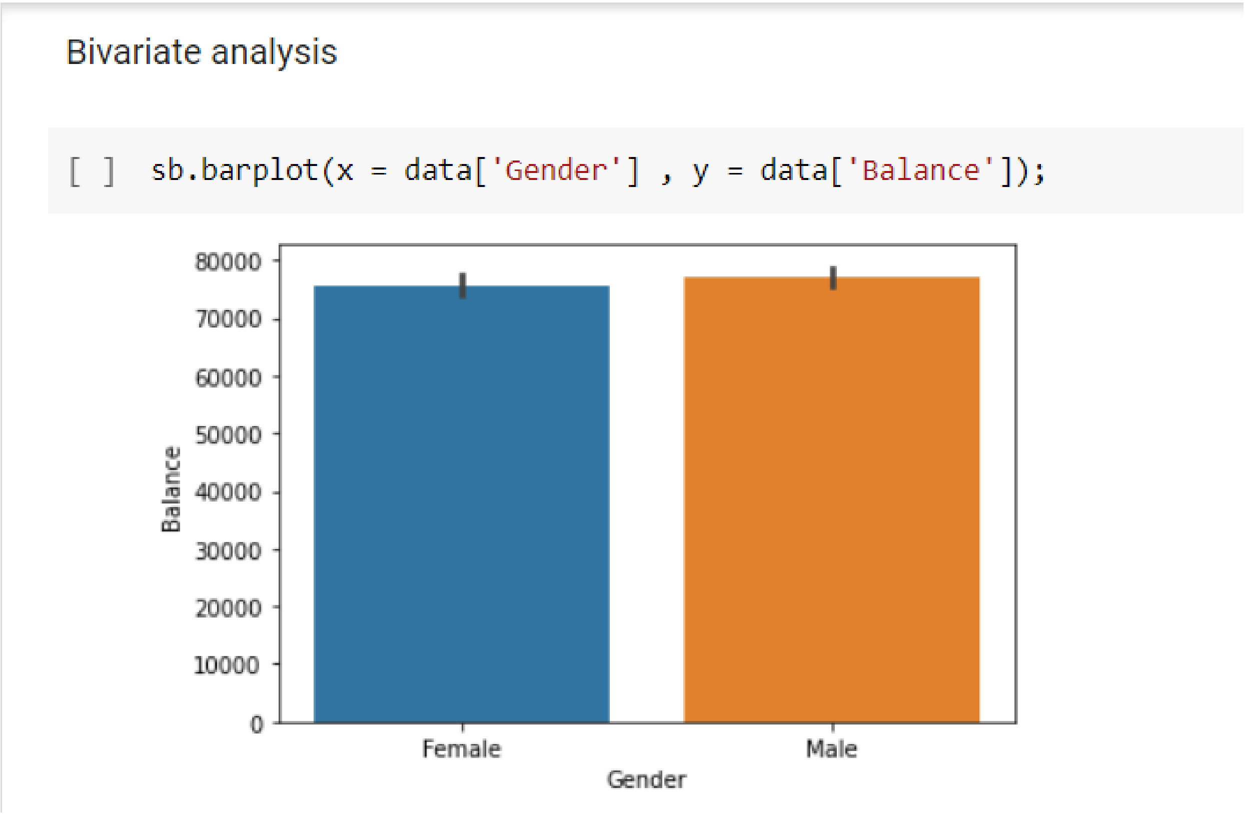
```
[5] data['EstimatedSalary'].median()
```

```
100193.915
```



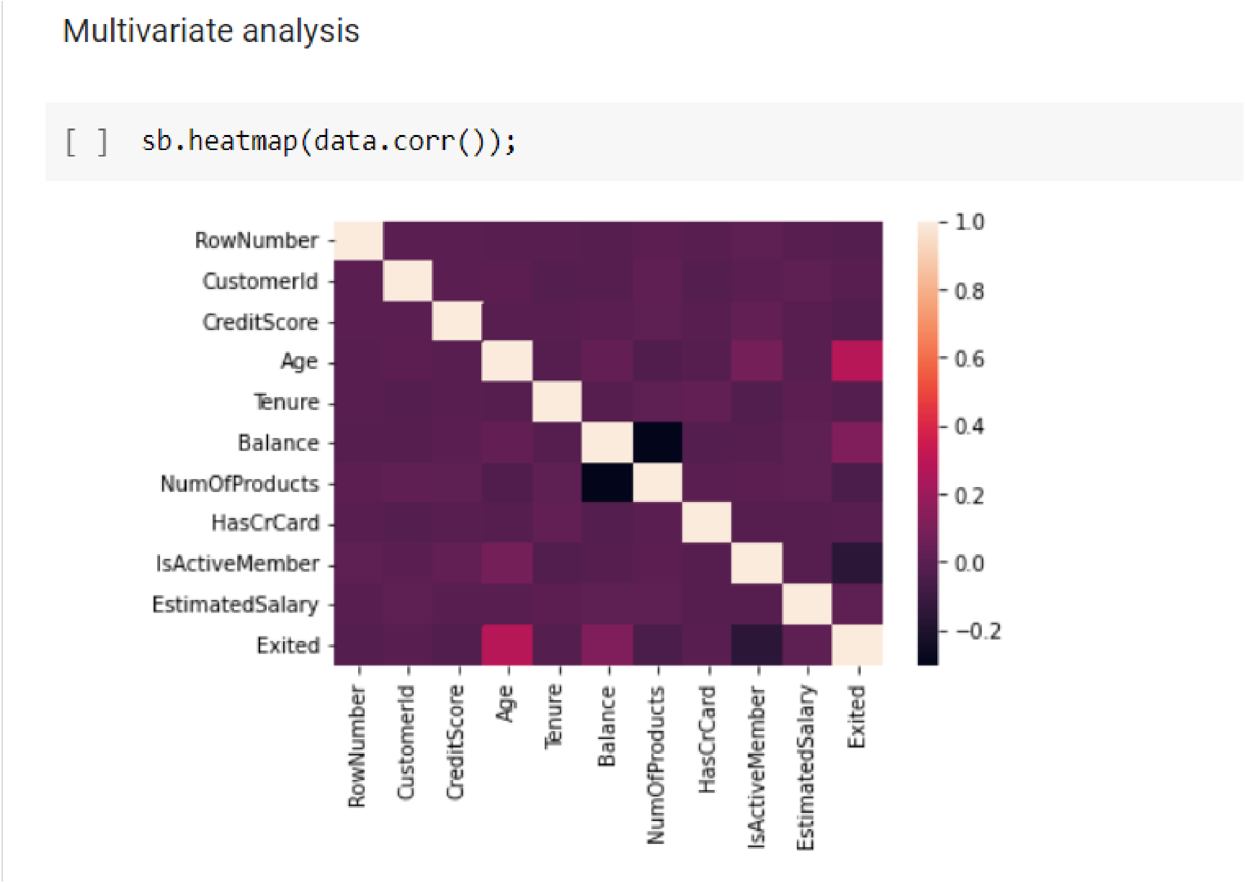
```
[6] sb.distplot(data['EstimatedSalary'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd4ea883d50>
```



Bivariate analysis

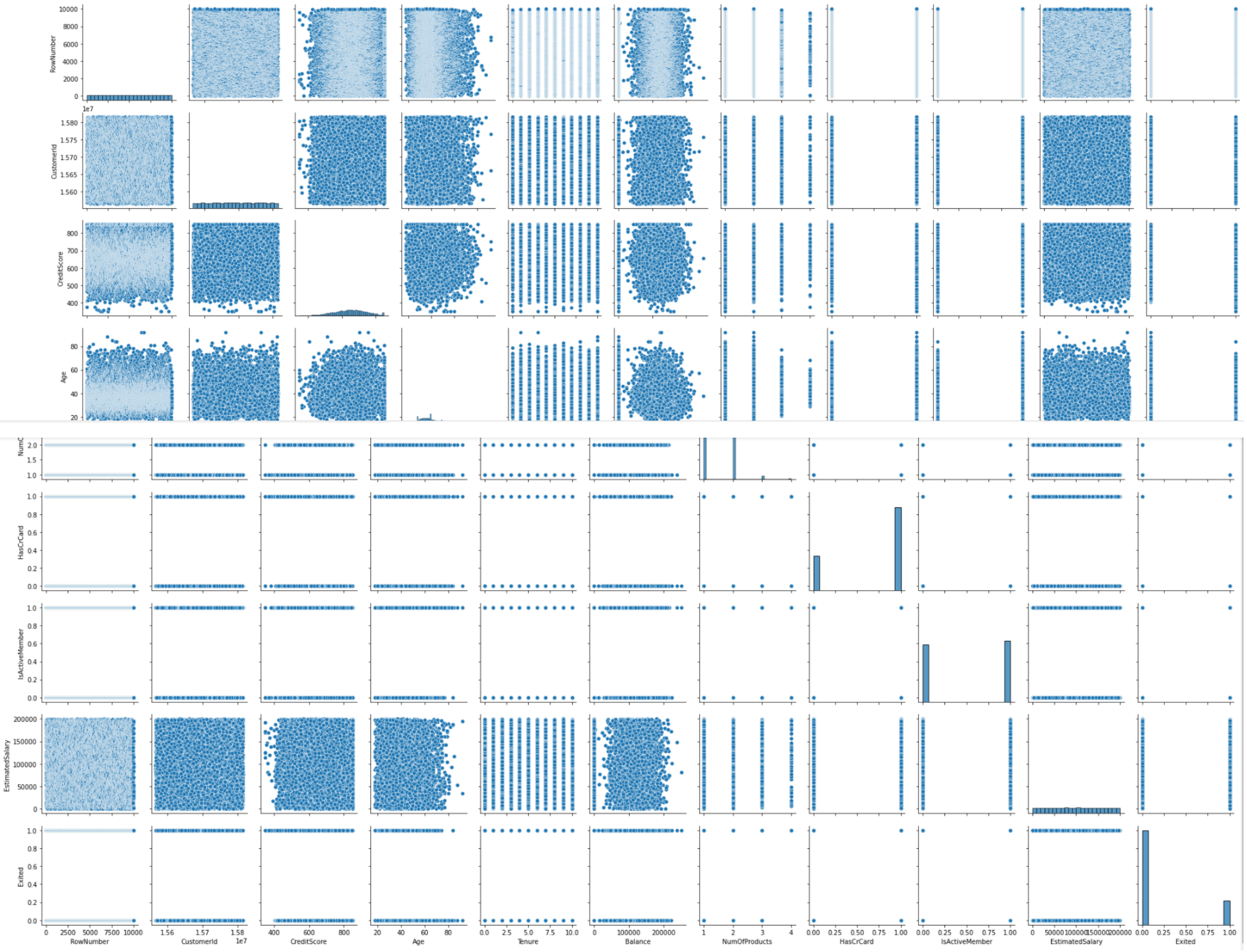
```
[ ]  sb.barplot(x = data['Gender'] , y = data['Balance']);
```



Multivariate analysis

```
[ ]  sb.heatmap(data.corr());
[ ] sb.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7efd458a6cd0>
```



Task 4 Perform descriptive statistics on the dataset

```
[ ] data.shape
```

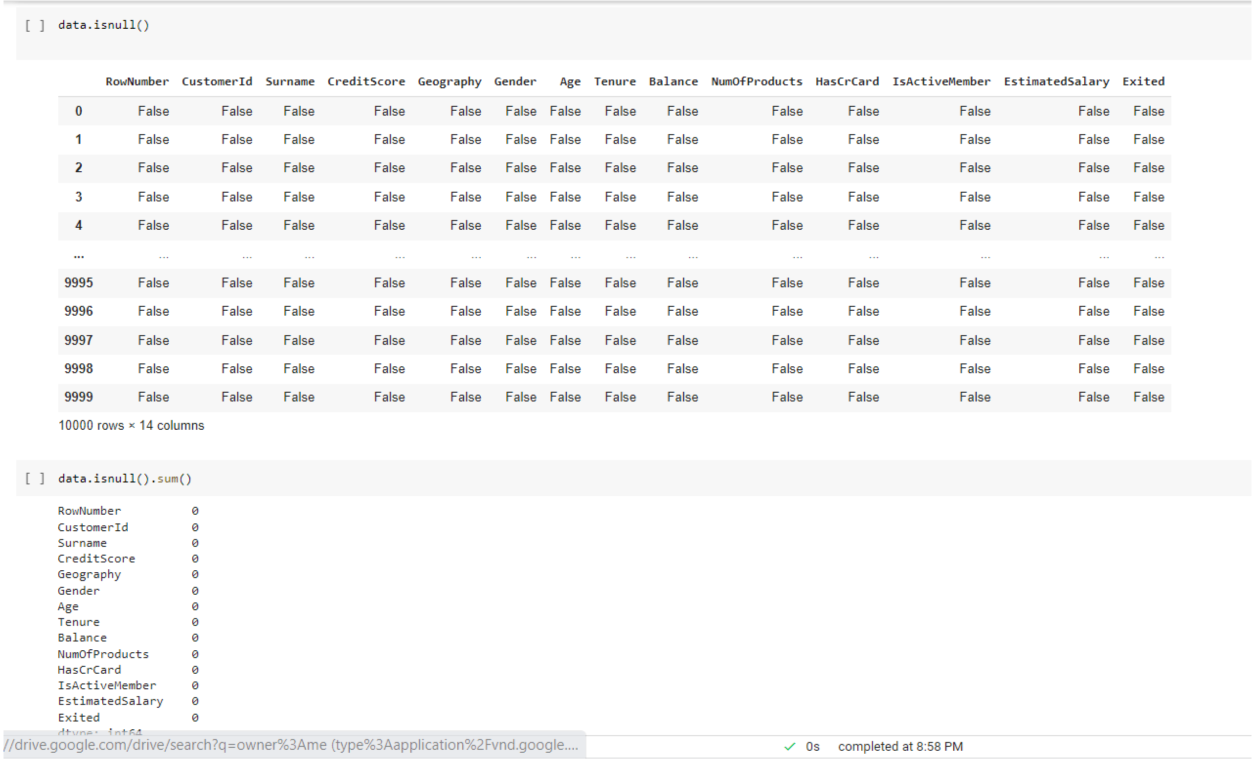
```
(10000, 14)
```

```
data.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

```
[ ] data.describe(include=['object'])
```

	Surname	Geography	Gender
count	10000	10000	10000



```
[ ] data.isnull()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9996	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9997	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9998	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9999	False	False	False	False	False	False	False	False	False	False	False	False	False	False

10000 rows × 14 columns

```
[ ] data.isnull().sum()
```

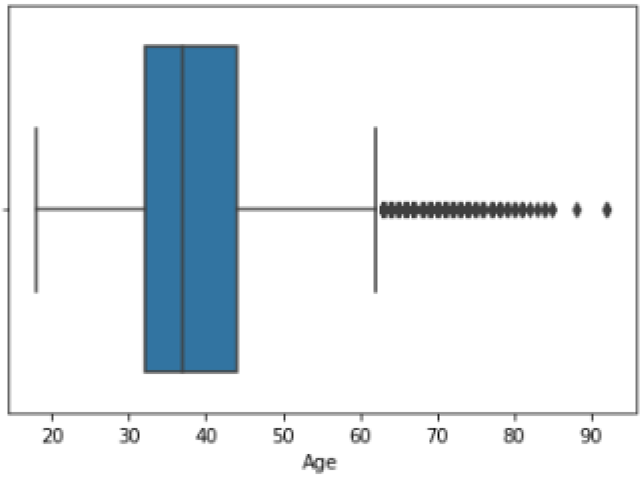
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

//drive.google.com/drive/search?q=owner%3Ame (type%3Aapplication%2Fvnd.google....

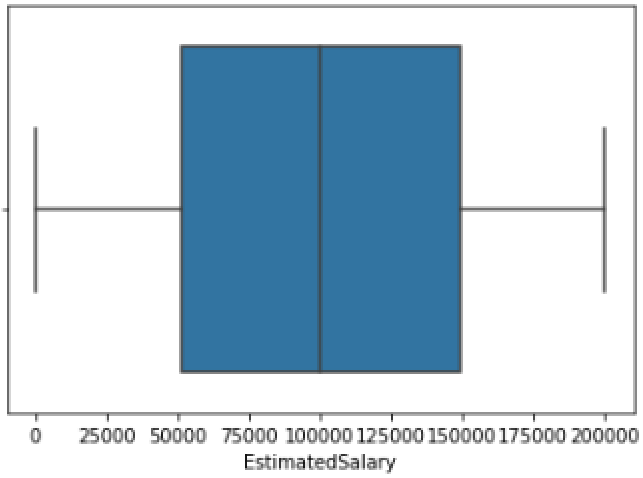
✓ 0s completed at 8:58 PM

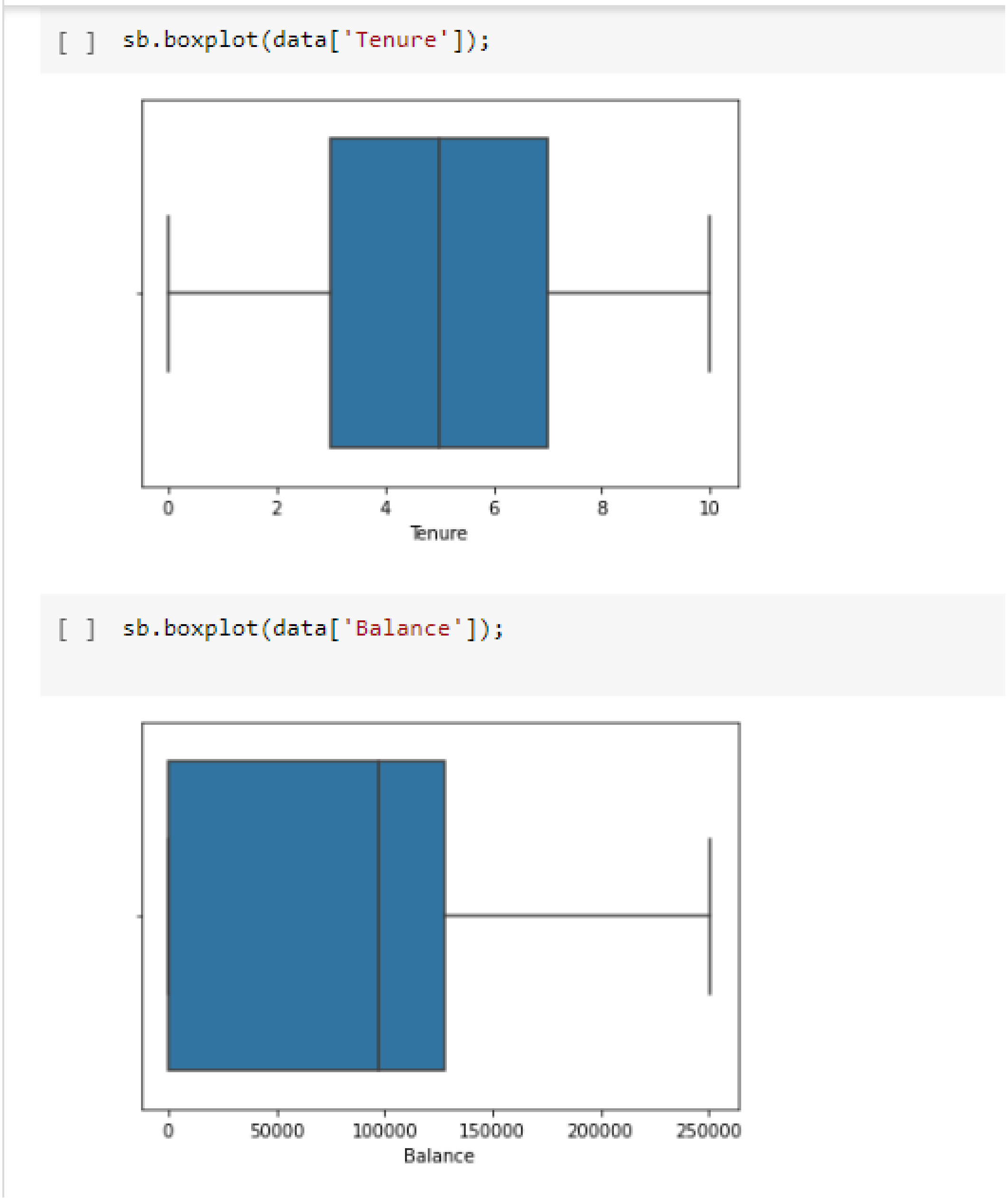
Task 6 Find the outliers and replace the outliers

```
[ ]  sb.boxplot(data['Age']);
```



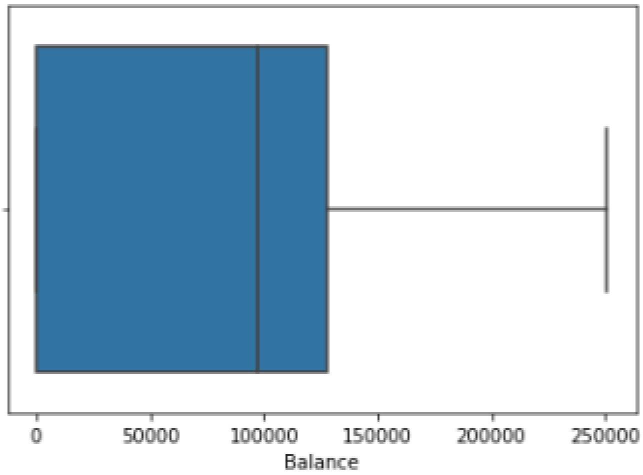
```
[ ]  sb.boxplot(data['EstimatedSalary']);
```



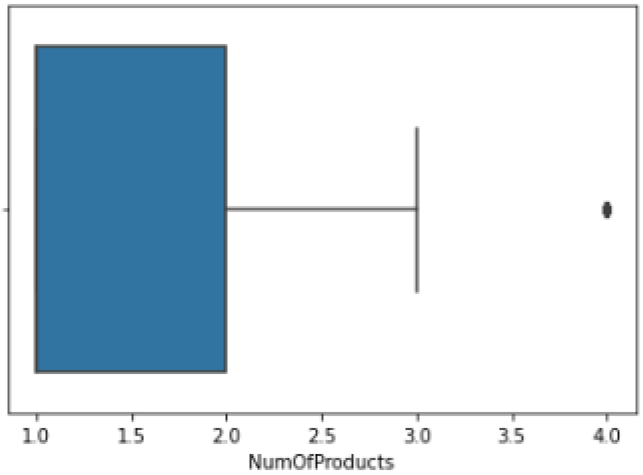


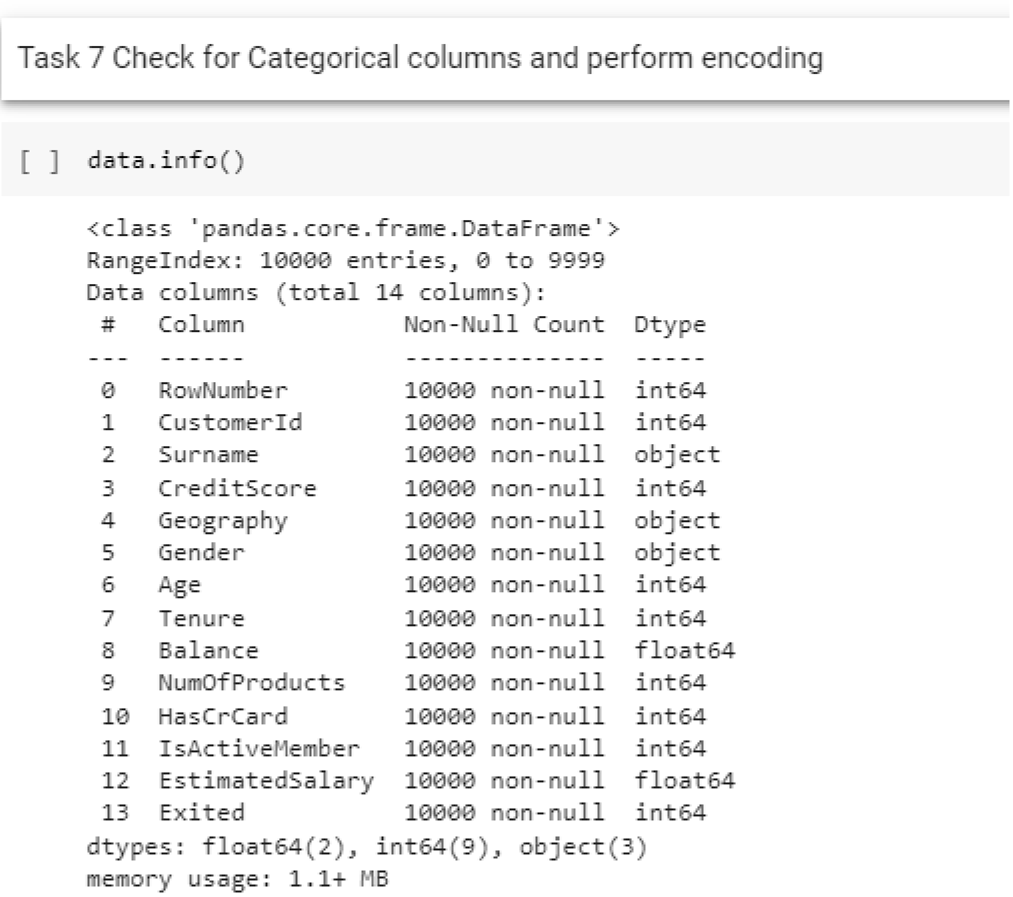
```
[ ]  sb.boxplot(data['Tenure']);
```

```
[ ]  sb.boxplot(data['Balance']);
[ ]  sb.boxplot(data['Balance']);
```



```
[ ]  sb.boxplot(data['NumOfProducts']);
```

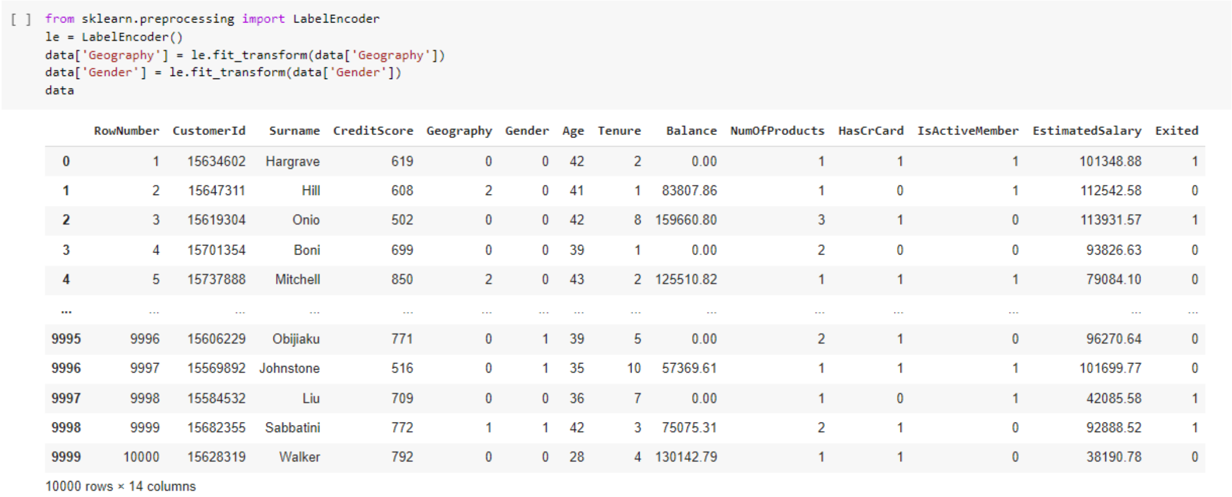




Task 7 Check for Categorical columns and perform encoding

```
[ ]  data.info()
```

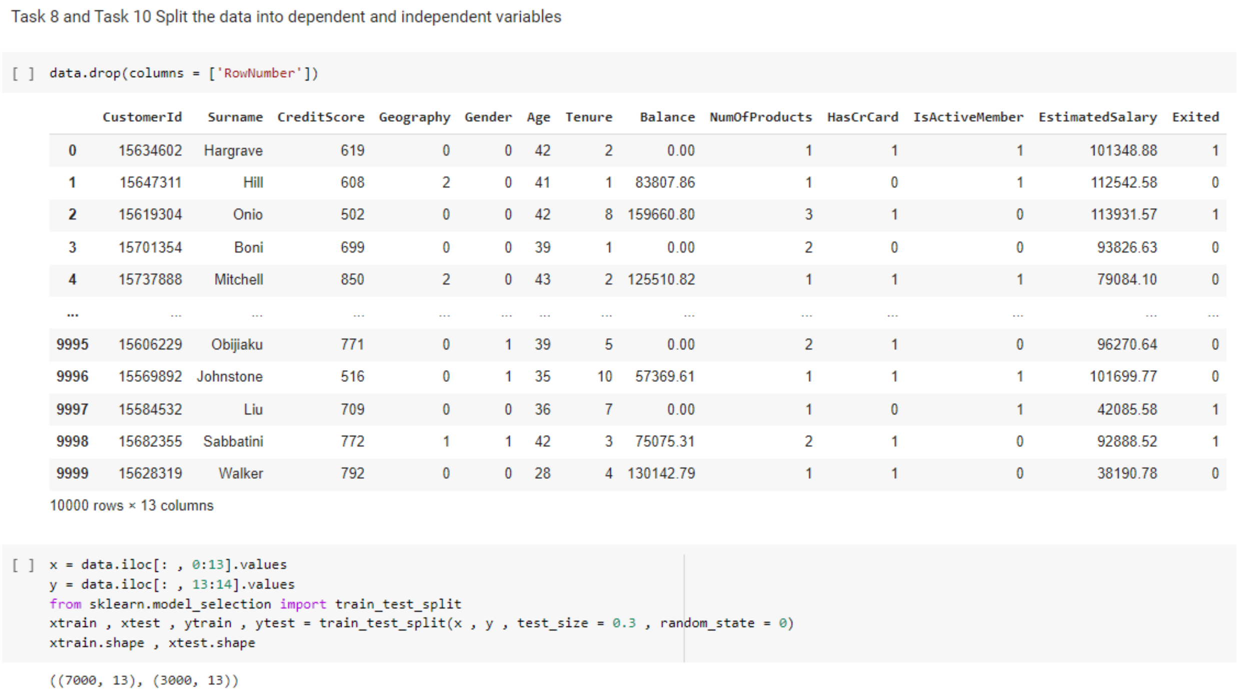
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```



```
[ ]  from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     data['Geography'] = le.fit_transform(data['Geography'])
     data['Gender'] = le.fit_transform(data['Gender'])
     data
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	2	0	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	0	1	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	0	1	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	0	0	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	1	1	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	0	0	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 14 columns



Task 8 and Task 10 Split the data into dependent and independent variables

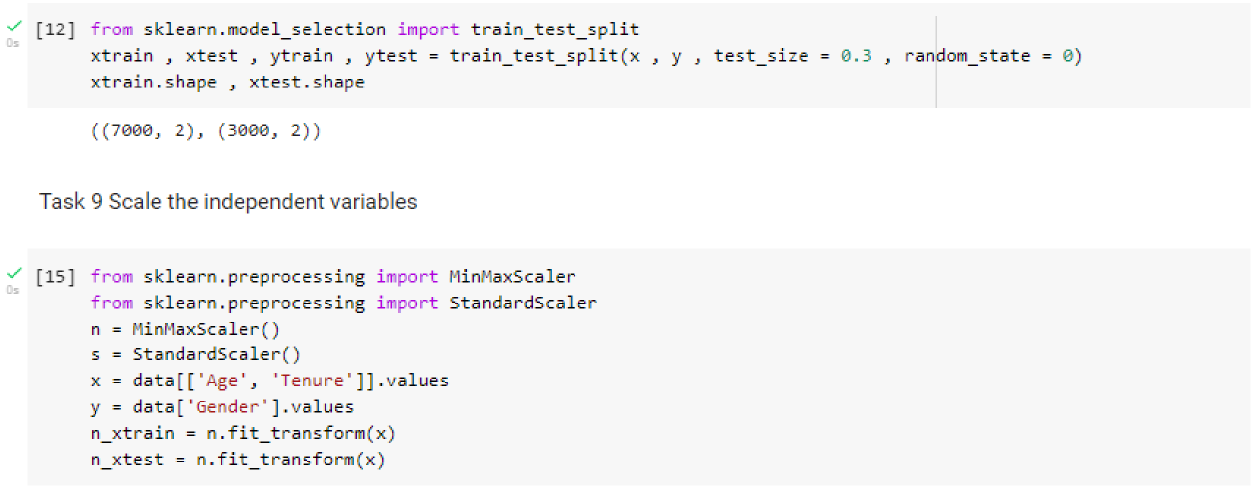
```
[ ] data.drop(columns = ['RowNumber'])
```

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	15647311	Hill	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	15619304	Onio	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	15701354	Boni	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	15737888	Mitchell	850	2	0	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15606229	Obijiaku	771	0	1	39	5	0.00	2	1	0	96270.64	0
9996	15569892	Johnstone	516	0	1	35	10	57369.61	1	1	1	101699.77	0
9997	15584532	Liu	709	0	0	36	7	0.00	1	0	1	42085.58	1
9998	15682355	Sabbatini	772	1	1	42	3	75075.31	2	1	0	92888.52	1
9999	15628319	Walker	792	0	0	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 13 columns

```
[ ] x = data.iloc[: , 0:13].values
    y = data.iloc[: , 13:14].values
    from sklearn.model_selection import train_test_split
    xtrain , xtest , ytrain , ytest = train_test_split(x , y , test_size = 0.3 , random_state = 0)
    xtrain.shape , xtest.shape

    ((7000, 13), (3000, 13))
```



```
[12] from sklearn.model_selection import train_test_split
     xtrain , xtest , ytrain , ytest = train_test_split(x , y , test_size = 0.3 , random_state = 0)
     xtrain.shape , xtest.shape

     ((7000, 2), (3000, 2))
```

Task 9 Scale the independent variables

```
[15] from sklearn.preprocessing import MinMaxScaler
     from sklearn.preprocessing import StandardScaler
     n = MinMaxScaler()
     s = StandardScaler()
     x = data[['Age', 'Tenure']].values
     y = data['Gender'].values
     n_xtrain = n.fit_transform(x)
     n_xtest = n.fit_transform(x)
```