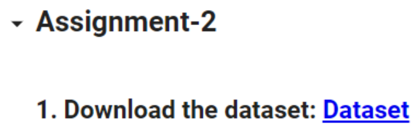
Assignment -2

Data Visualization and Pre-processing

Assignment Date	27 September 2022
Student Name	R. Vignesh Kumar
Student Roll Number	912419104038
Maximum Marks	2 Marks

Task 1:

1. Download the dataset: [Dataset]



▾ Assignment-2

1. Download the dataset: Dataset

Task 2:

2.Loading the Churn_Modelling dataset

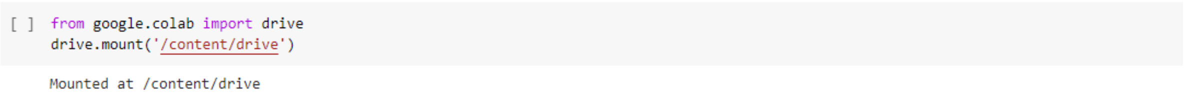
Solution:

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

import warnings
warnings.filterwarnings("ignore")
```

▾ 2.Loading the Churn_Modelling dataset

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

import warnings
warnings.filterwarnings("ignore")
```

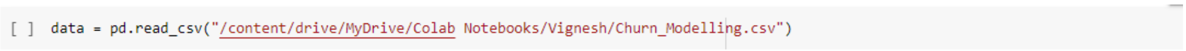
```
from google.colab import drive
drive.mount('/content/drive')
```



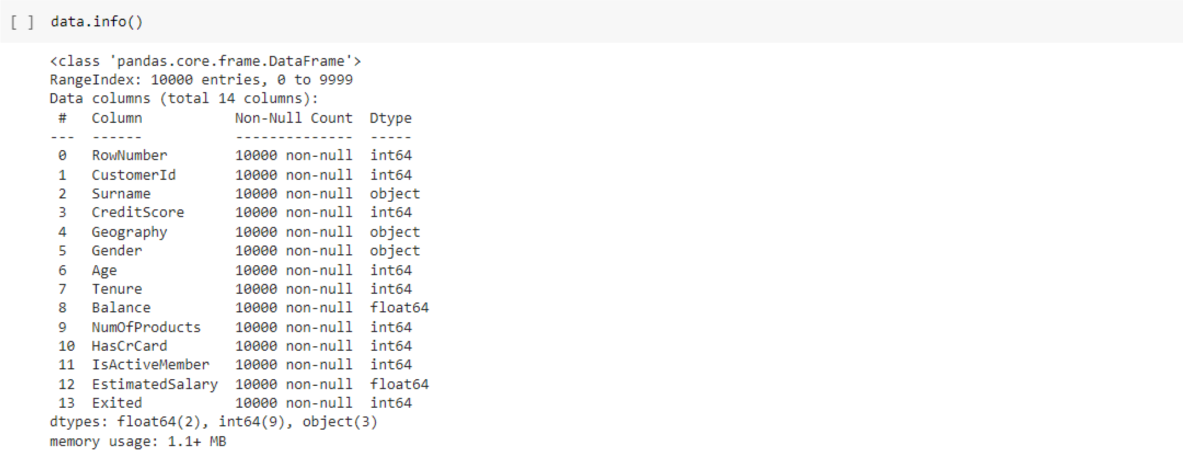
```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

data = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Vignesh/Churn_Modelling.csv")

data.info()

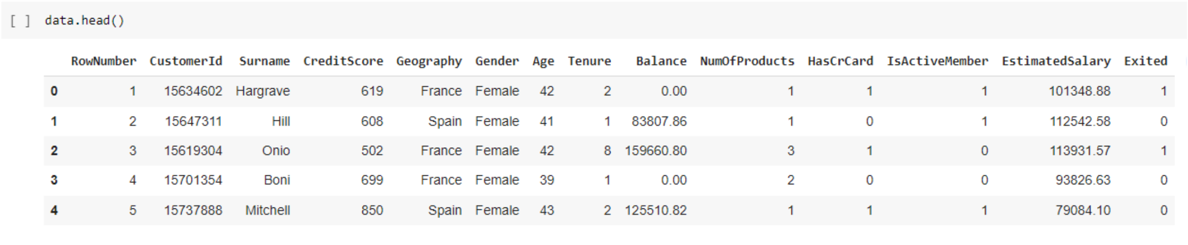


```
[ ] data = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Vignesh/Churn_Modelling.csv")
```



```
[ ] 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
```

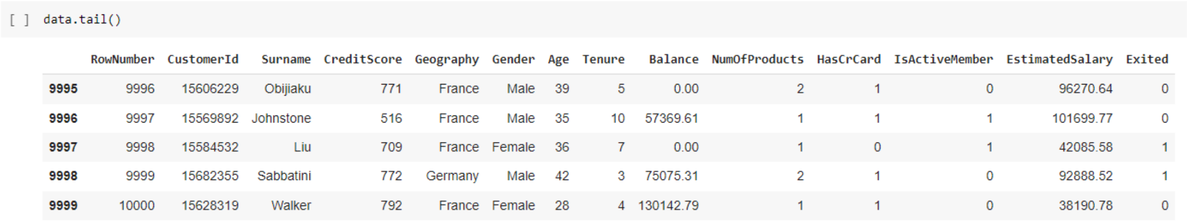
data.head()



```
[ ] data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

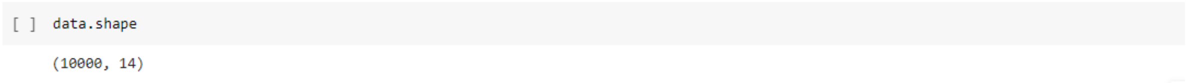
data.head()



```
[ ] data.tail()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

data.shape



```
[ ] data.shape

    (10000, 14)
```

Task 3:

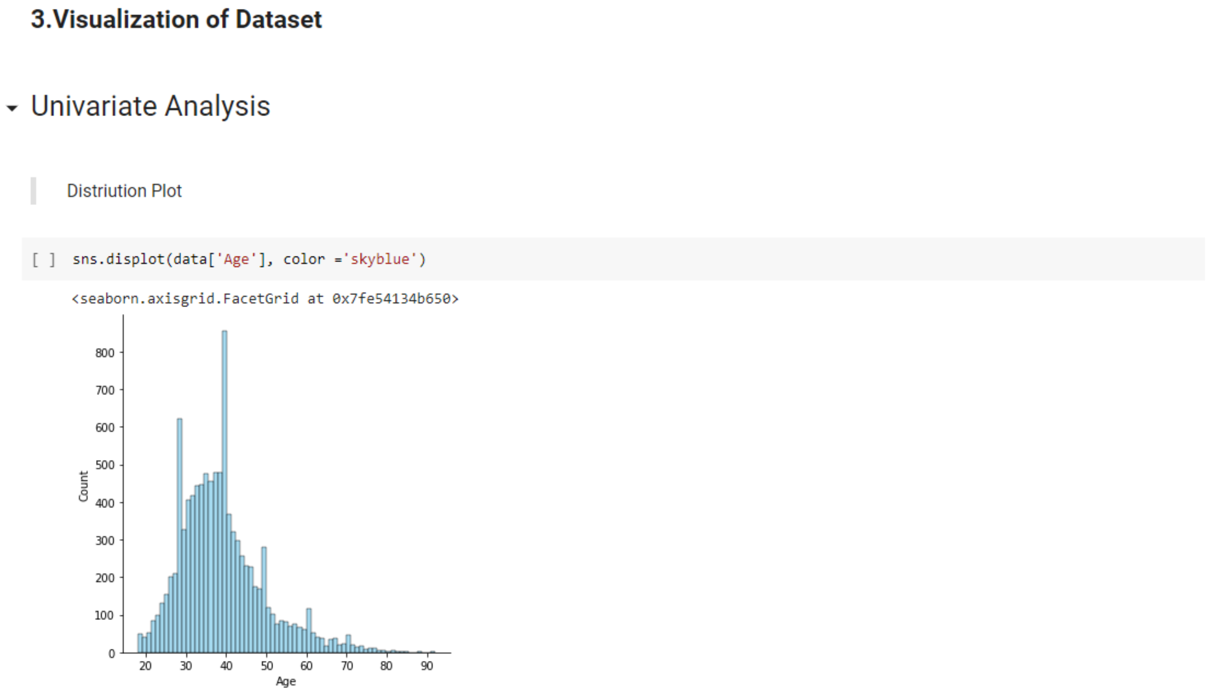
3. Visualization of Dataset

Univariate Analysis

- Distribution Plot

Solution:

sns.displot(data['Age'], color ='skyblue')



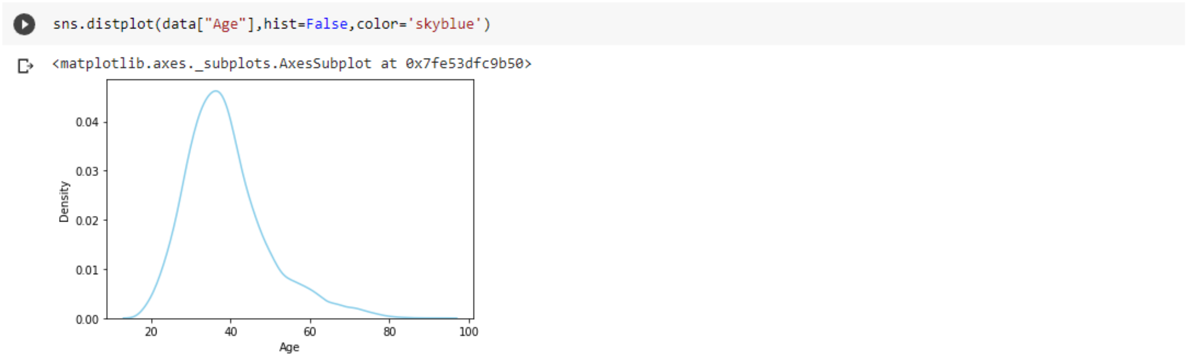
3.Visualization of Dataset

Univariate Analysis

Distriution Plot

```
[ ] sns.displot(data['Age'], color ='skyblue')

    <seaborn.axisgrid.FacetGrid at 0x7fe54134b650>
```

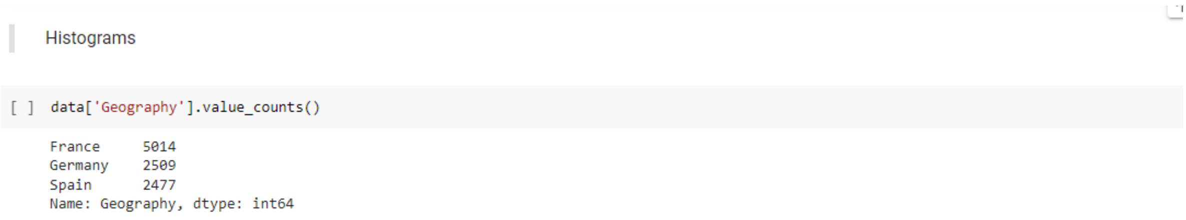
sns.distplot(data["Age"],hist=False,color='skyblue')



```
sns.distplot(data["Age"],hist=False,color='skyblue')

<matplotlib.axes._subplots.AxesSubplot at 0x7fe53dfc9b50>
```

- Histograms

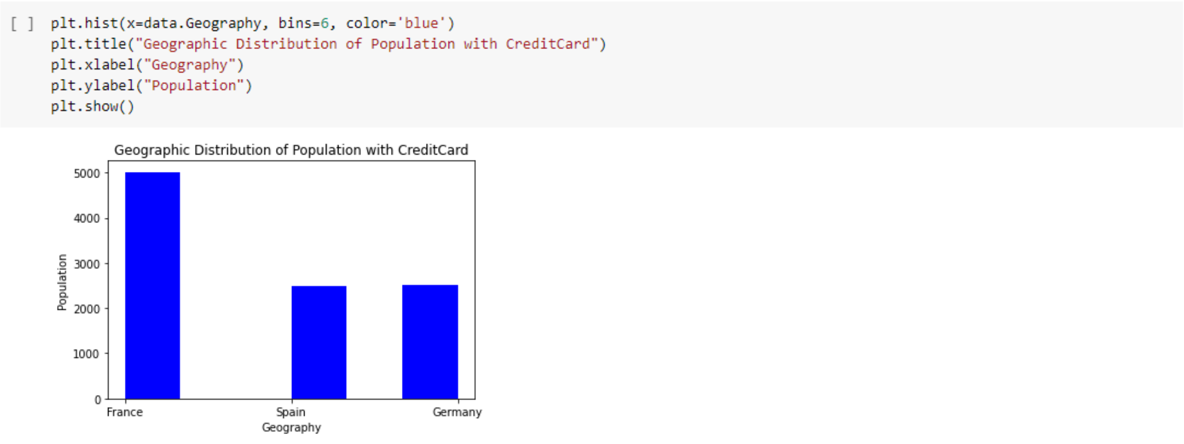
data['Geography'].value_counts()



Histograms

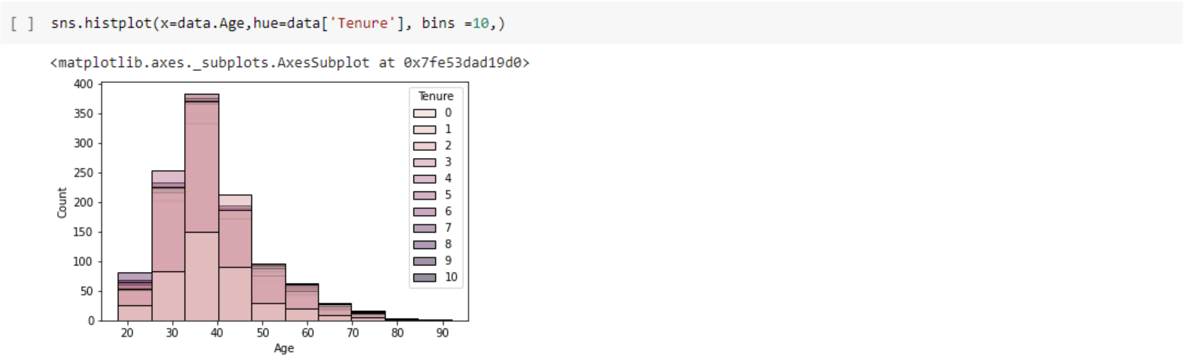
```
[ ] data['Geography'].value_counts()

    France     5014
    Germany    2509
    Spain      2477
    Name: Geography, dtype: int64
```

plt.hist(x=data.Geography, bins=6, color='blue')
plt.title("Geographic Distribution of Population with CreditCard")
plt.xlabel("Geography")
plt.ylabel("Population")
plt.show()



```
[ ] plt.hist(x=data.Geography, bins=6, color='blue')
    plt.title("Geographic Distribution of Population with CreditCard")
    plt.xlabel("Geography")
    plt.ylabel("Population")
    plt.show()
```

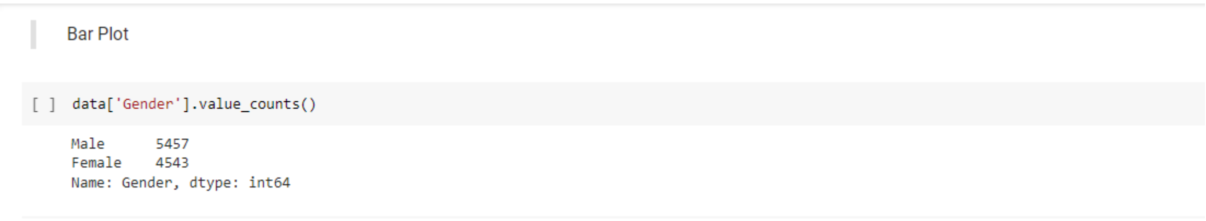
sns.histplot(x=data.Age,hue=data['Tenure'], bins =10,)



```
[ ] sns.histplot(x=data.Age,hue=data['Tenure'], bins =10,)

    <matplotlib.axes._subplots.AxesSubplot at 0x7fe53dad19d0>
```

- Bar Plot

data['Gender'].value_counts()

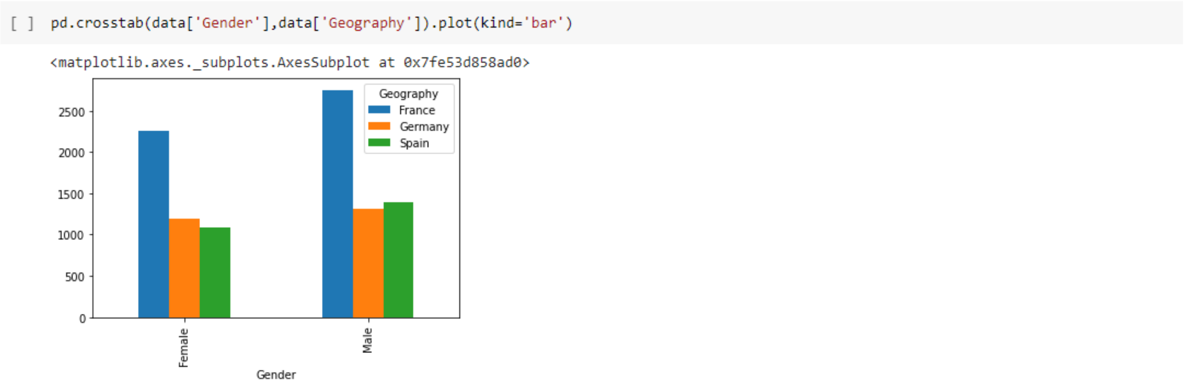


Bar Plot

```
[ ]  data['Gender'].value_counts()
```

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

pd.crosstab(data['Gender'],data['Geography']).plot(kind='bar')

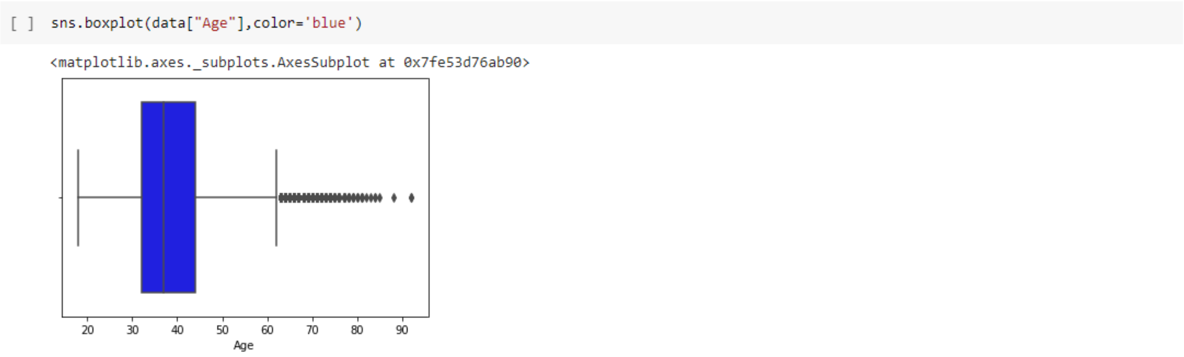


```
[ ]  pd.crosstab(data['Gender'],data['Geography']).plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe53d858ad0>
```

- Box Plot

sns.boxplot(data["Age"],color='blue')

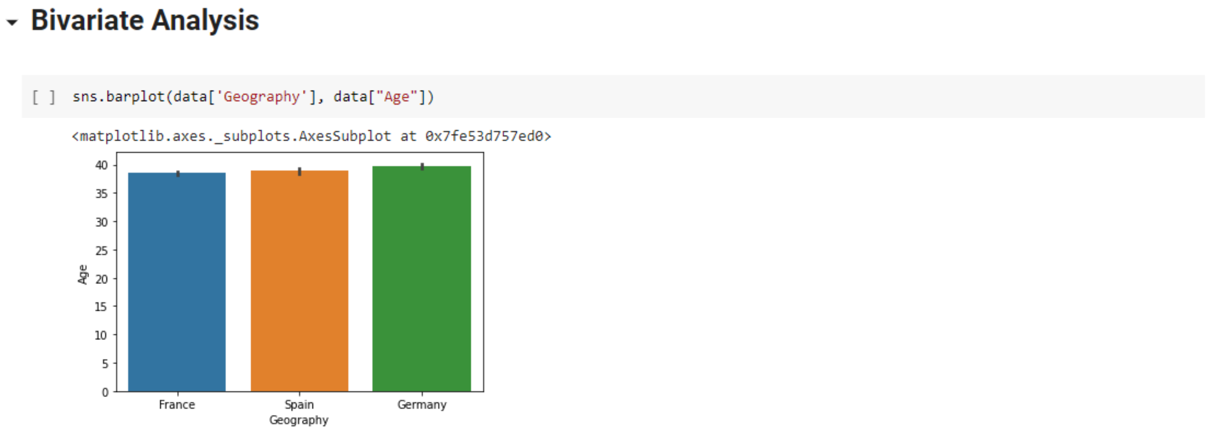


```
[ ]  sns.boxplot(data["Age"],color='blue')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe53d76ab90>
```

Bivariate Analysis

sns.barplot(data['Geography'], data["Age"])

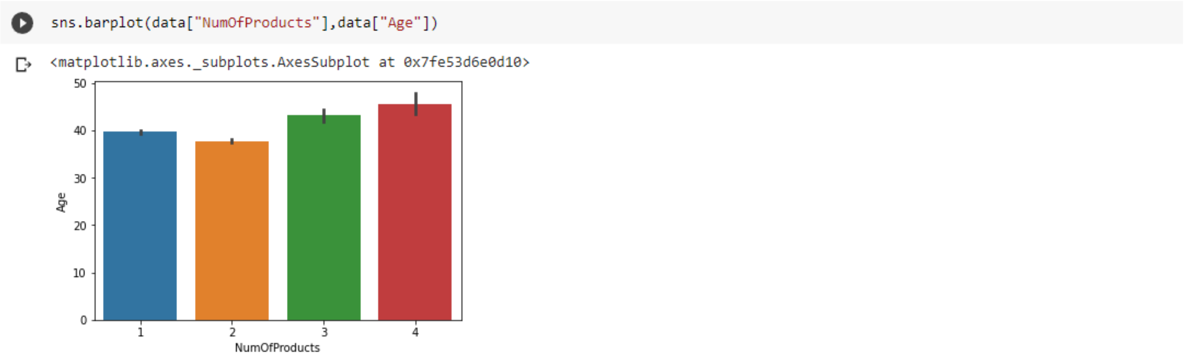


Bivariate Analysis

```
[ ] sns.barplot(data['Geography'], data["Age"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe53d757ed0>
```

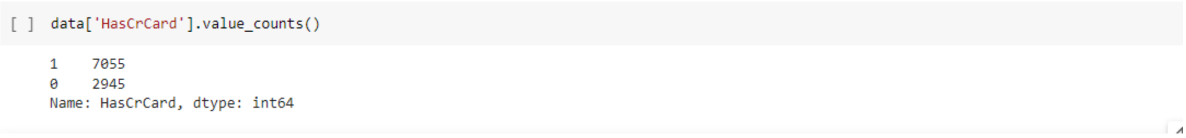
sns.barplot(data["NumOfProducts"],data["Age"])



```
sns.barplot(data["NumOfProducts"],data["Age"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe53d6e0d10>
```

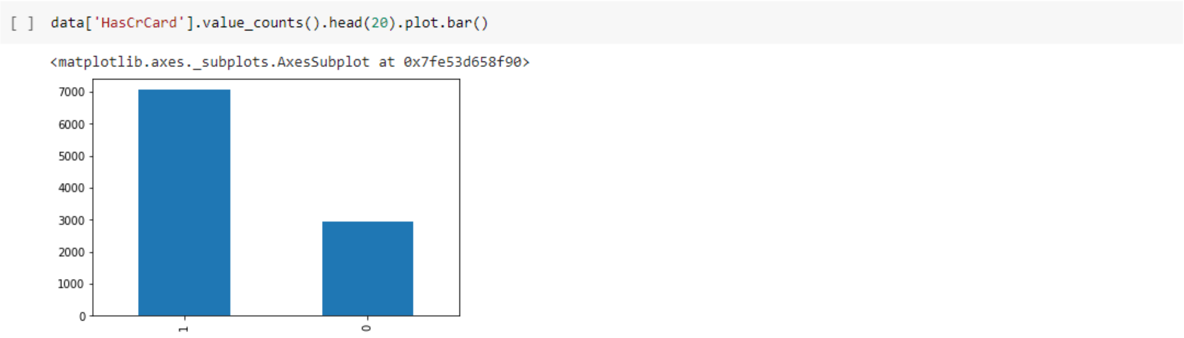
data['HasCrCard'].value_counts()



```
[ ] data['HasCrCard'].value_counts()
```

```
1    7055
0    2945
Name: HasCrCard, dtype: int64
```

data['HasCrCard'].value_counts().head(20).plot.bar()

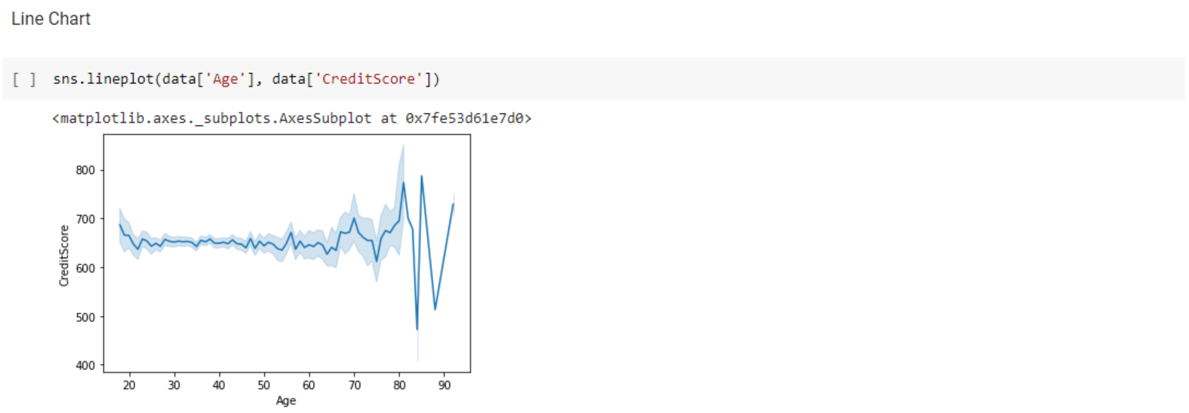


```
[ ] data['HasCrCard'].value_counts().head(20).plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe53d658f90>
```

- Line Chart

sns.lineplot(data['Age'], data['CreditScore'])



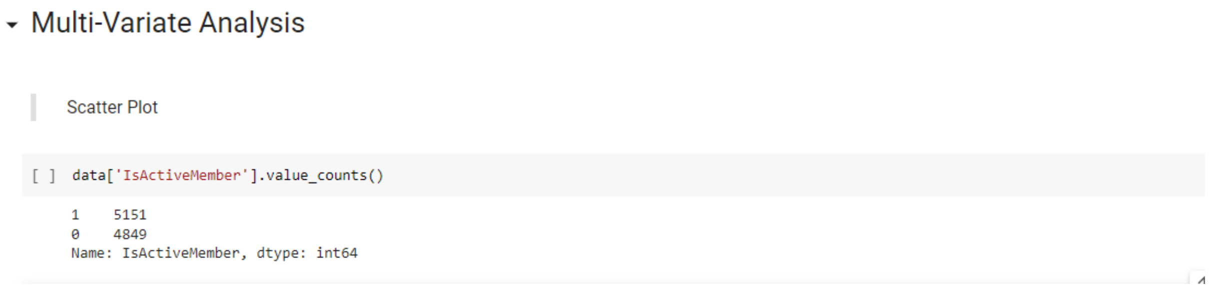
Line Chart

```
[ ]  sns.lineplot(data['Age'], data['CreditScore'])

     <matplotlib.axes._subplots.AxesSubplot at 0x7fe53d61e7d0>
```

Multi-Variate Analysis

- Scatter Plot

data['IsActiveMember'].value_counts()

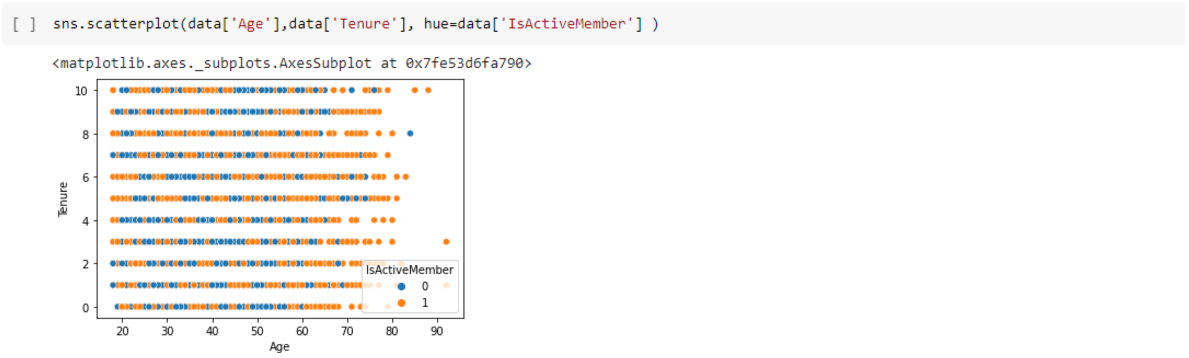


Multi-Variate Analysis

Scatter Plot

```
[ ]  data['IsActiveMember'].value_counts()

     1    5151
     0    4849
     Name: IsActiveMember, dtype: int64
```

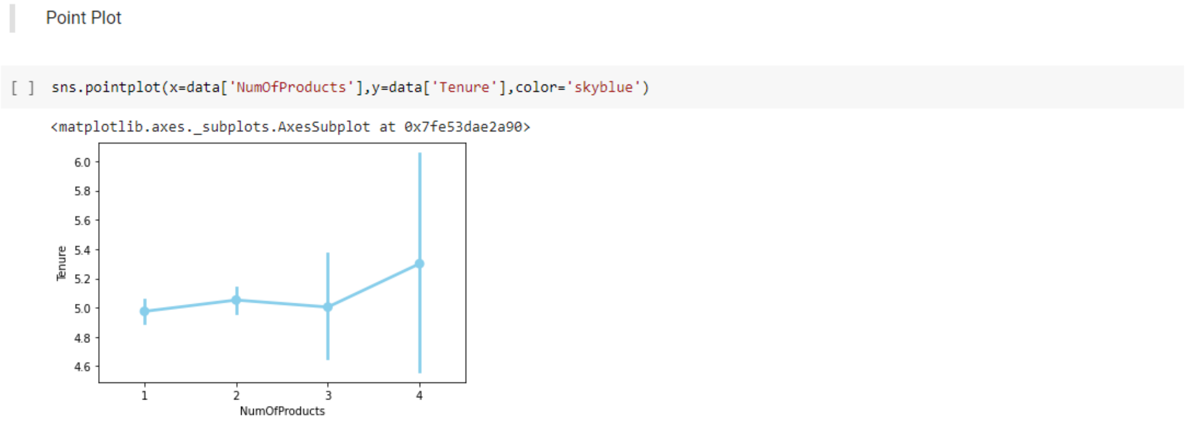
sns.scatterplot(data['Age'],data['Tenure'], hue=data['IsActiveMember'])



```
[ ]  sns.scatterplot(data['Age'],data['Tenure'], hue=data['IsActiveMember'] )

     <matplotlib.axes._subplots.AxesSubplot at 0x7fe53d6fa790>
```

- Point Plot

sns.pointplot(x=data['NumOfProducts'],y=data['Tenure'],color='skyblue')



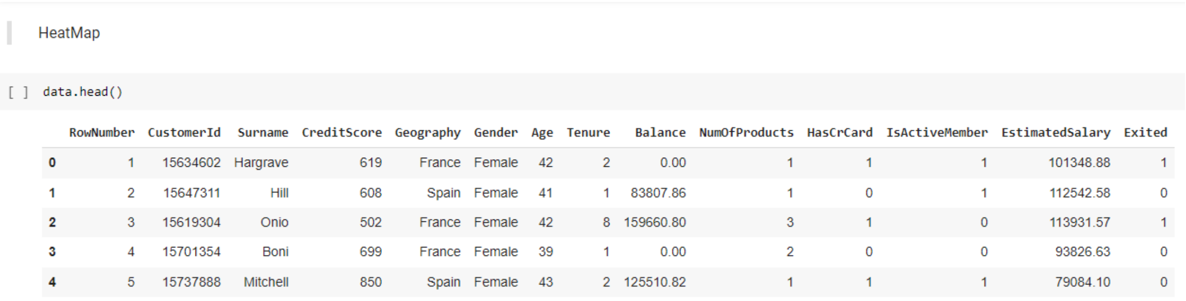
> Point Plot

```
[ ]  sns.pointplot(x=data['NumOfProducts'],y=data['Tenure'],color='skyblue')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe53dae2a90>
```

- HeatMap

data.head()

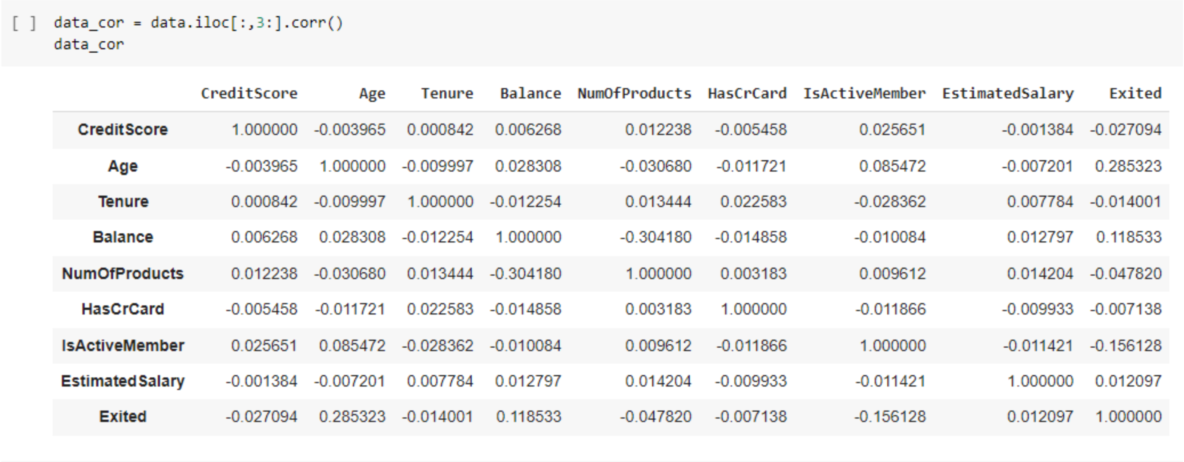


> HeatMap

```
[ ]  data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

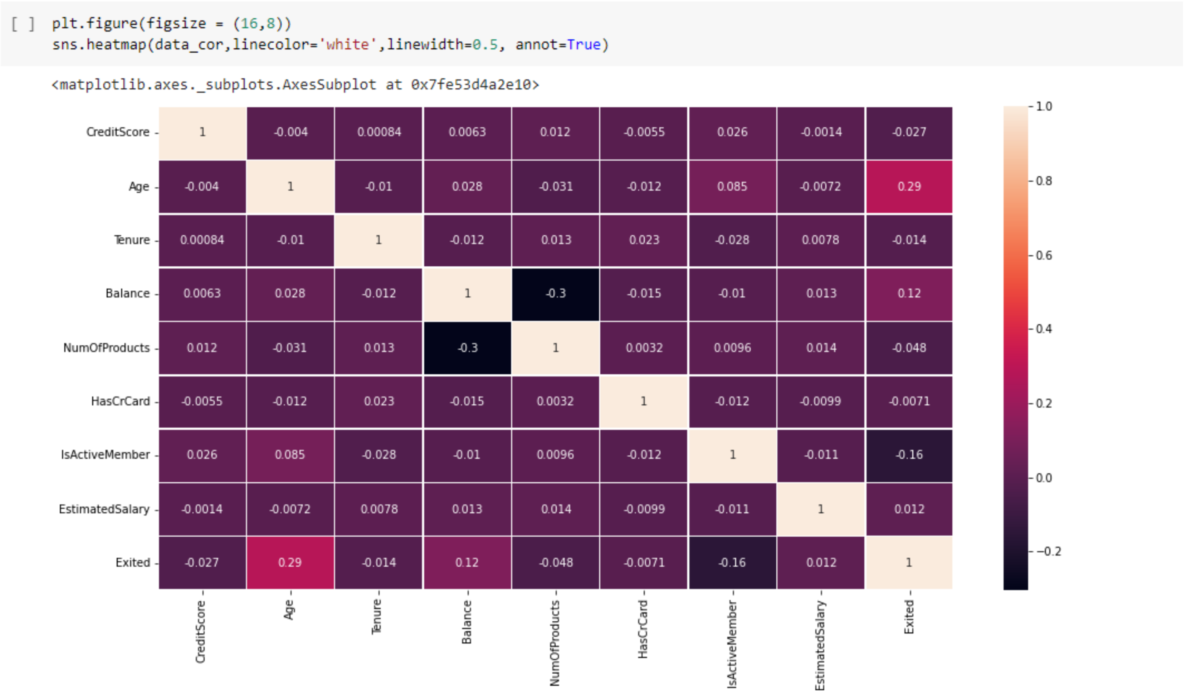
data_cor = data.iloc[:,3:].corr()
data_cor



```
[ ]  data_cor = data.iloc[:,3:].corr()
     data_cor
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
CreditScore	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
Age	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
Tenure	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
Balance	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
NumOfProducts	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
HasCrCard	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
IsActiveMember	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
EstimatedSalary	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
Exited	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

plt.figure(figsize = (16,8))
sns.heatmap(data_cor,linecolor='white',linewidth=0.5, annot=True)



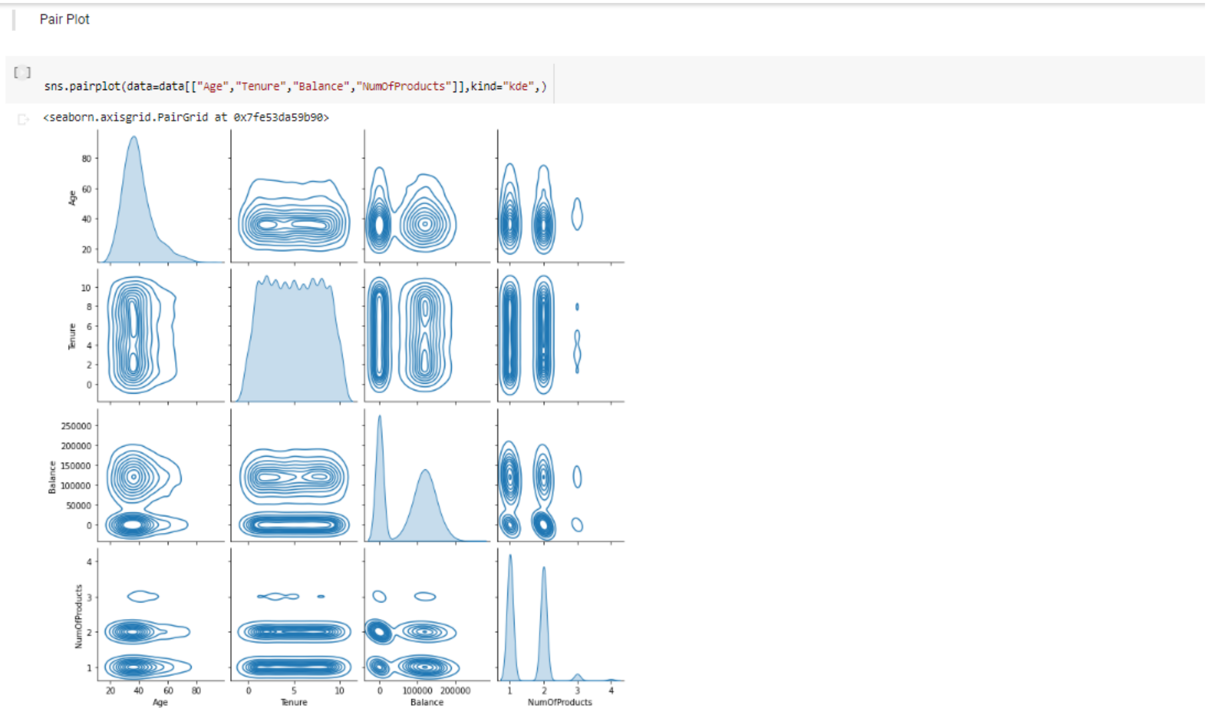
```
[ ]  plt.figure(figsize = (16,8))
     sns.heatmap(data_cor,linecolor='white',linewidth=0.5, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe53d4a2e10>
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
CreditScore	1	-0.004	0.00084	0.0063	0.012	-0.0055	0.026	-0.0014	-0.027
Age	-0.004	1	-0.01	0.028	-0.031	-0.012	0.085	-0.0072	0.29
Tenure	0.00084	-0.01	1	-0.012	0.013	0.023	-0.028	0.0078	-0.014
Balance	0.0063	0.028	-0.012	1	-0.3	-0.015	-0.01	0.013	0.12
NumOfProducts	0.012	-0.031	0.013	-0.3	1	0.0032	0.0096	0.014	-0.048
HasCrCard	-0.0055	-0.012	0.023	-0.015	0.0032	1	-0.012	-0.0099	-0.0071
IsActiveMember	0.026	0.085	-0.028	-0.01	0.0096	-0.012	1	-0.011	-0.16
EstimatedSalary	-0.0014	-0.0072	0.0078	0.013	0.014	-0.0099	-0.011	1	0.012
Exited	-0.027	0.29	-0.014	0.12	-0.048	-0.0071	-0.16	0.012	1

- Pair Plot

data.head()sns.pairplot(data=data[["Age","Tenure","Balance","NumOfProducts"]],kind="kde",)



Pair Plot

```
[ ]  sns.pairplot(data=data[["Age","Tenure","Balance","NumOfProducts"]],kind="kde",)
```

```
<seaborn.axisgrid.PairGrid at 0x7fe53da59b90>
```

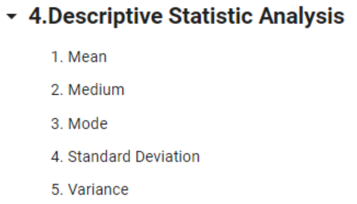
Task 4:

4. Descriptive Statistic Analysis

1. Mean
2. Medium
3. Mode
4. Standard Deviation
5. Variance

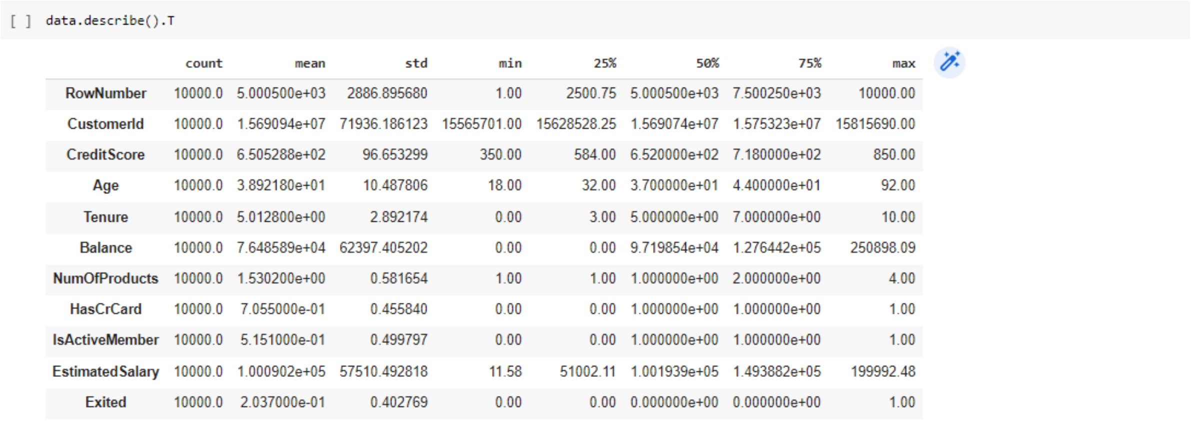
Solution:

data.describe().T



4.Descriptive Statistic Analysis

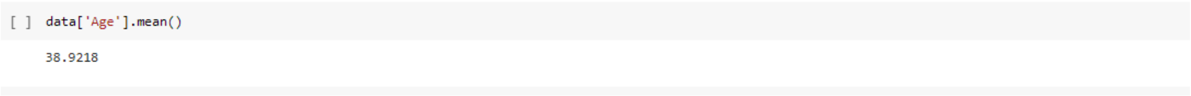
1. Mean
2. Medium
3. Mode
4. Standard Deviation
5. Variance



```
[ ]  data.describe().T
```

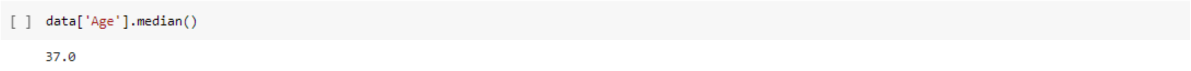
	count	mean	std	min	25%	50%	75%	max
RowNumber	10000.0	5.000500e+03	2886.895680	1.00	2500.75	5.000500e+03	7.500250e+03	10000.00
CustomerId	10000.0	1.569094e+07	71936.186123	15565701.00	15628528.25	1.569074e+07	1.575323e+07	15815690.00
CreditScore	10000.0	6.505288e+02	96.653299	350.00	584.00	6.520000e+02	7.180000e+02	850.00
Age	10000.0	3.892180e+01	10.487806	18.00	32.00	3.700000e+01	4.400000e+01	92.00
Tenure	10000.0	5.012800e+00	2.892174	0.00	3.00	5.000000e+00	7.000000e+00	10.00
Balance	10000.0	7.648589e+04	62397.405202	0.00	0.00	9.719854e+04	1.276442e+05	250898.09
NumOfProducts	10000.0	1.530200e+00	0.581654	1.00	1.00	1.000000e+00	2.000000e+00	4.00
HasCrCard	10000.0	7.055000e-01	0.455840	0.00	0.00	1.000000e+00	1.000000e+00	1.00
IsActiveMember	10000.0	5.151000e-01	0.499797	0.00	0.00	1.000000e+00	1.000000e+00	1.00
EstimatedSalary	10000.0	1.000902e+05	57510.492818	11.58	51002.11	1.001939e+05	1.493882e+05	199992.48
Exited	10000.0	2.037000e-01	0.402769	0.00	0.00	0.000000e+00	0.000000e+00	1.00

data['Age'].mean()



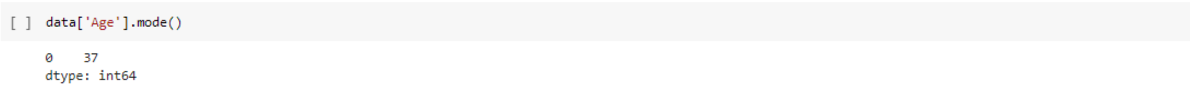
```
[ ]  data['Age'].mean()

     38.9218
```

data['Age'].median()



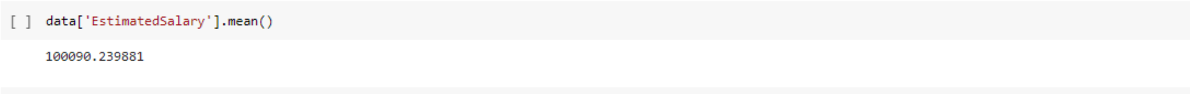
```
[ ]  data['Age'].median()

     37.0
```

data['Age'].mode()



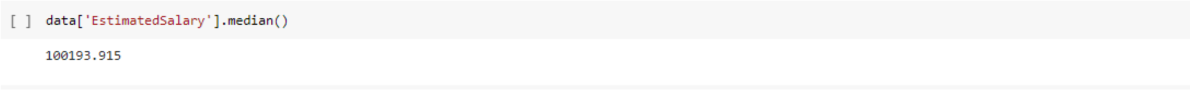
```
[ ]  data['Age'].mode()

     0    37
     dtype: int64
```

data['EstimatedSalary'].mean()



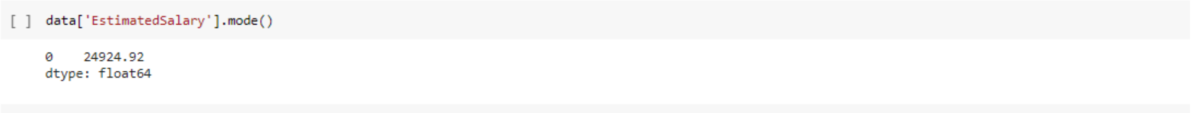
```
[ ] data['EstimatedSalary'].mean()

    100090.239881
```

data['EstimatedSalary'].median(),)



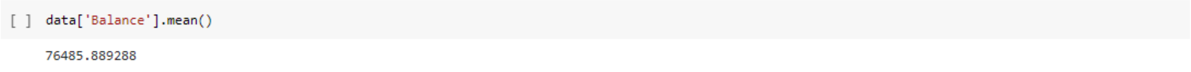
```
[ ] data['EstimatedSalary'].median()

    100193.915
```

data['EstimatedSalary'].mode())



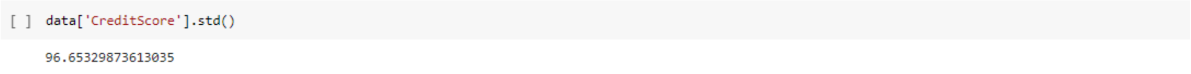
```
[ ] data['EstimatedSalary'].mode()

    0    24924.92
    dtype: float64
```

data['Balance'].mean()



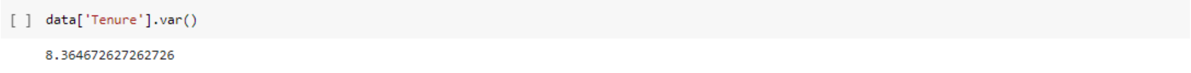
```
[ ] data['Balance'].mean()

    76485.889288
```

data['CreditScore'].std()



```
[ ] data['CreditScore'].std()

    96.65329873613035
```

data['Tenure'].var()



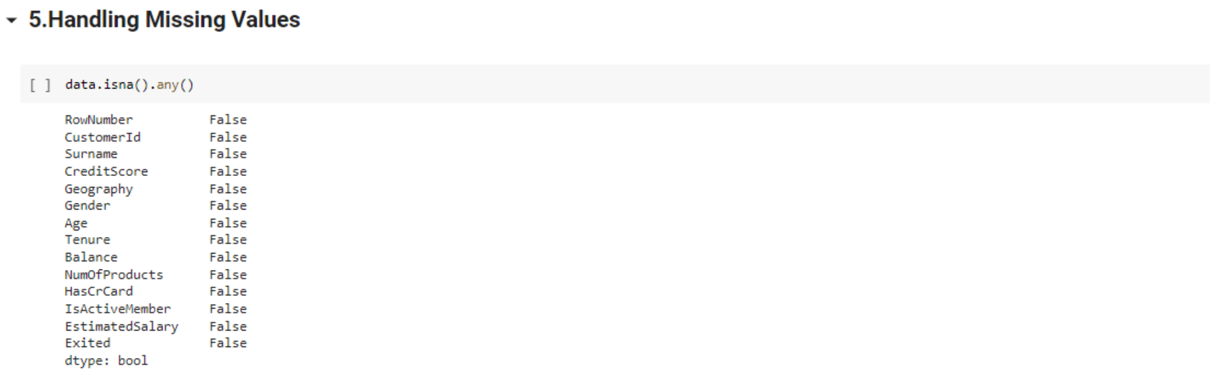
```
[ ] data['Tenure'].var()

    8.364672627262726
```

Task 5:

5.Handling Missing Values

Solution:

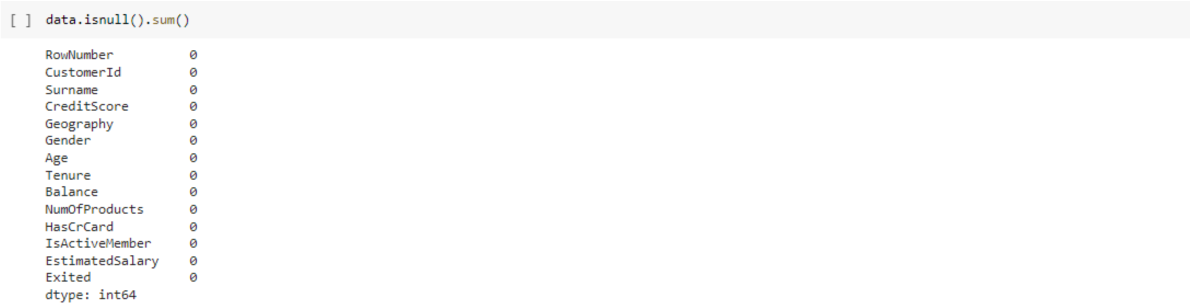
data.isna().any()



▾ 5.Handling Missing Values

```
[ ] data.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
```

```
data.isnull().sum()
```



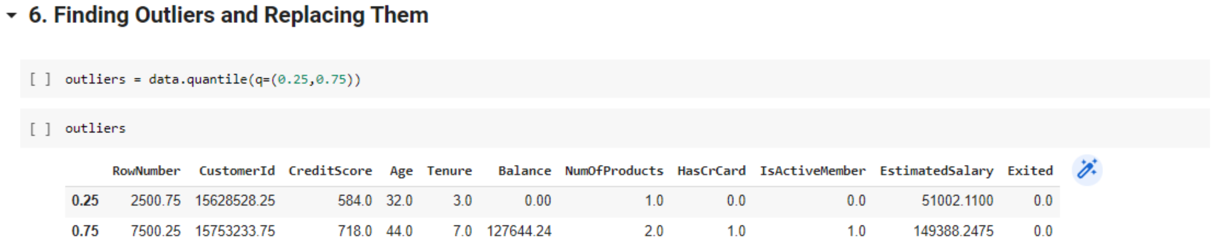
```
[ ] data.isnull().sum()

    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
```

Task 6:

6. Finding Outliers and Replacing Them

Solution:

```
outliers = data.quantile(q=(0.25,0.75))
outliers
```

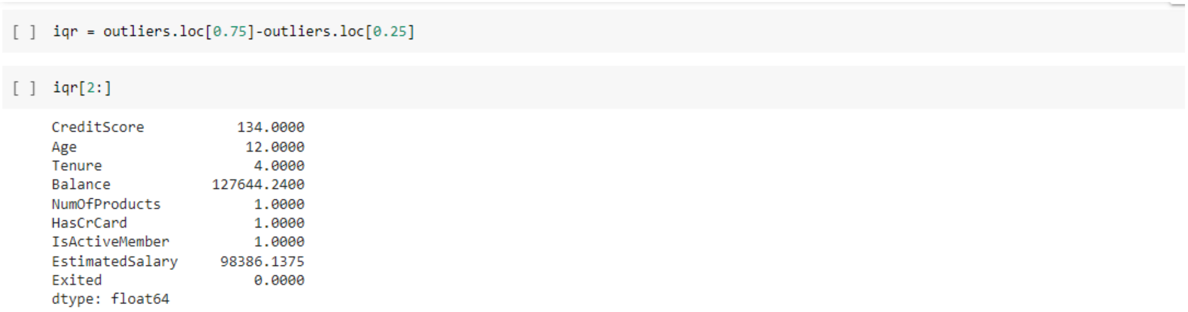


6. Finding Outliers and Replacing Them

```
[ ] outliers = data.quantile(q=(0.25,0.75))

[ ] outliers
```

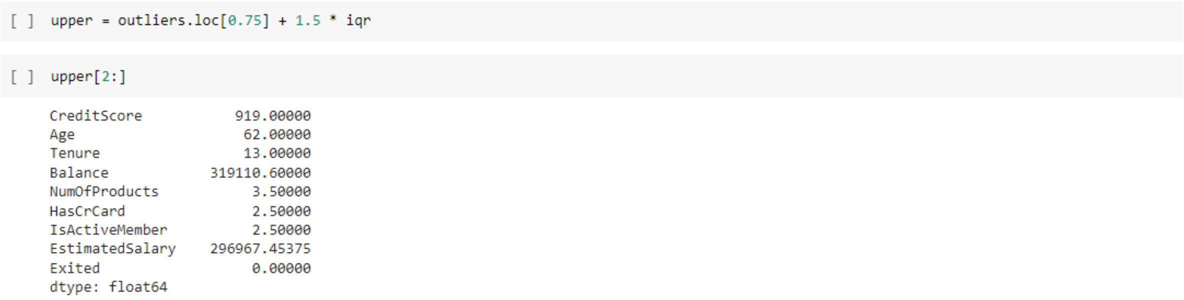
	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.25	2500.75	15628528.25	584.0	32.0	3.0	0.00	1.0	0.0	0.0	51002.1100	0.0
0.75	7500.25	15753233.75	718.0	44.0	7.0	127644.24	2.0	1.0	1.0	149388.2475	0.0

```
iqr = outliers.loc[0.75]-outliers.loc[0.25]
iqr[2:]
```



```
[ ] iqr = outliers.loc[0.75]-outliers.loc[0.25]

[ ] iqr[2:]

    CreditScore           134.0000
    Age                    12.0000
    Tenure                  4.0000
    Balance            127644.2400
    NumOfProducts           1.0000
    HasCrCard               1.0000
    IsActiveMember          1.0000
    EstimatedSalary     98386.1375
    Exited                  0.0000
    dtype: float64
```

upper = outliers.loc[0.75] + 1.5 * iqr
upper[2:]

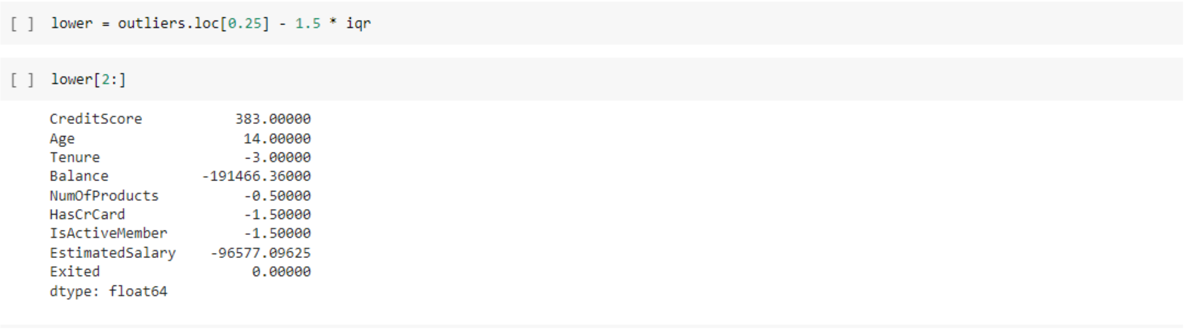


```
[ ]  upper = outliers.loc[0.75] + 1.5 * iqr
```

```
[ ]  upper[2:]
```

```
CreditScore           919.00000
Age                    62.00000
Tenure                 13.00000
Balance            319110.60000
NumOfProducts           3.50000
HasCrCard               2.50000
IsActiveMember          2.50000
EstimatedSalary    296967.45375
Exited                  0.00000
dtype: float64
```

lower = outliers.loc[0.25] - 1.5 * iqr
lower[2:]

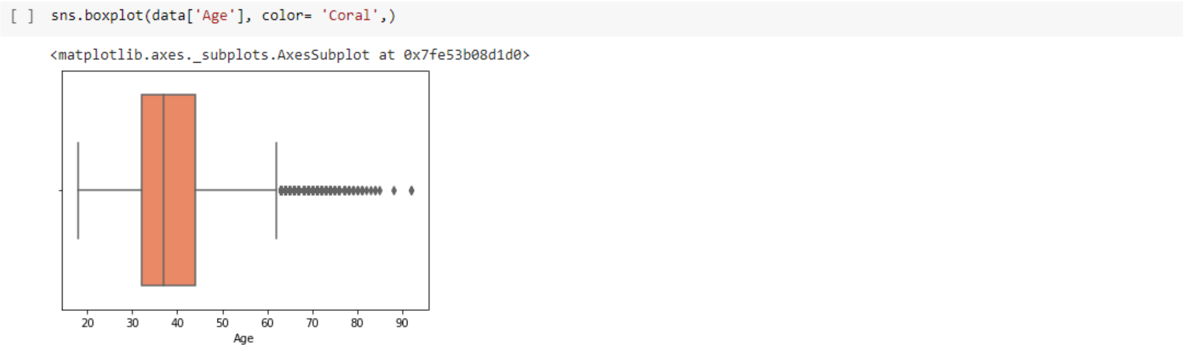


```
[ ]  lower = outliers.loc[0.25] - 1.5 * iqr
```

```
[ ]  lower[2:]
```

```
CreditScore           383.00000
Age                    14.00000
Tenure                 -3.00000
Balance           -191466.36000
NumOfProducts          -0.50000
HasCrCard              -1.50000
IsActiveMember         -1.50000
EstimatedSalary    -96577.09625
Exited                  0.00000
dtype: float64
```

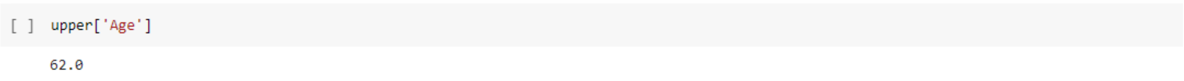
sns.boxplot(data['Age'], color= 'Coral',)



```
[ ]  sns.boxplot(data['Age'], color= 'Coral',)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe53b08d1d0>
```

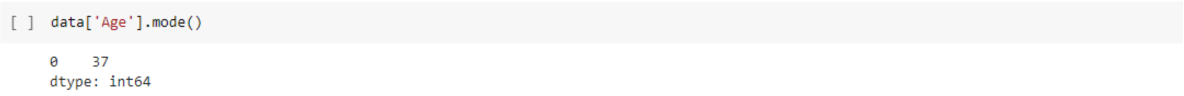
upper['Age']



```
[ ]  upper['Age']
```

```
62.0
```

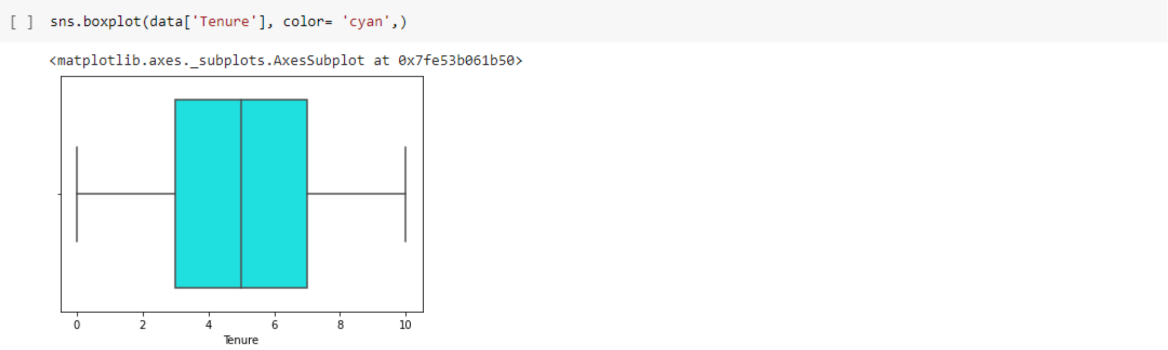
data['Age'].mode()



```
[ ]  data['Age'].mode()
```

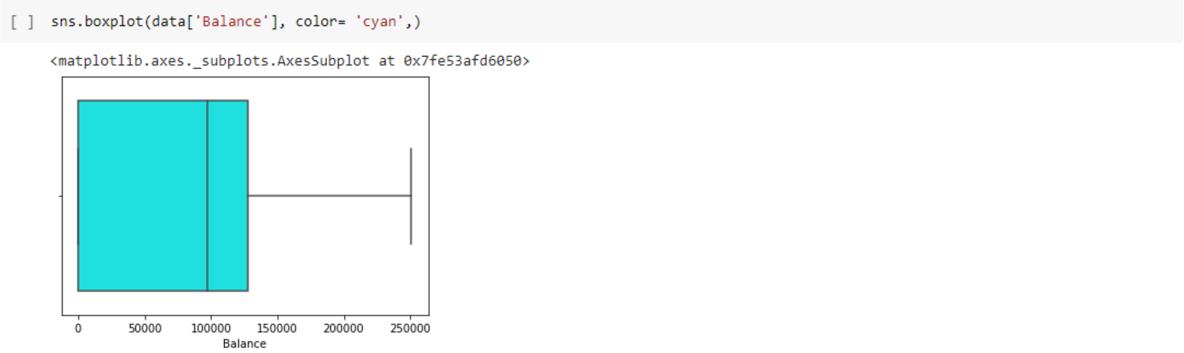
```
0    37
dtype: int64
```

sns.boxplot(data['Tenure'], color= 'cyan',)



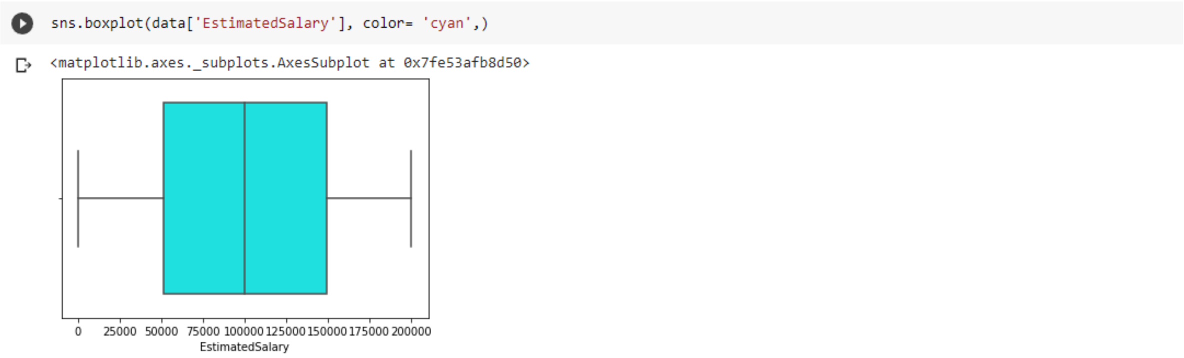
```
[ ] sns.boxplot(data['Tenure'], color= 'cyan',)

<matplotlib.axes._subplots.AxesSubplot at 0x7fe53b061b50>
```

sns.boxplot(data['EstimatedSalary'], color= 'cyan',)



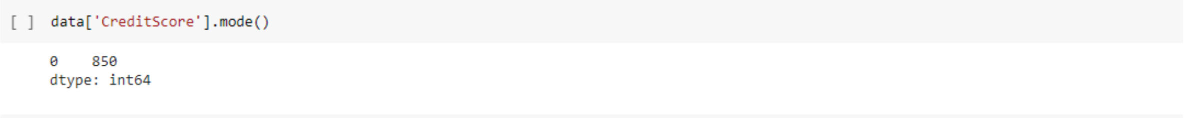
```
[ ] sns.boxplot(data['Balance'], color= 'cyan',)

<matplotlib.axes._subplots.AxesSubplot at 0x7fe53afd6050>
```

sns.boxplot(data['CreditScore'], color= 'cyan',)



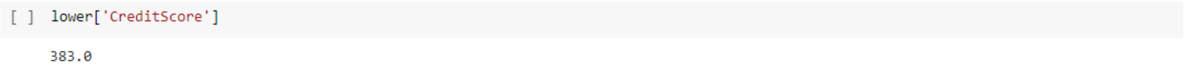
```
sns.boxplot(data['EstimatedSalary'], color= 'cyan',)

<matplotlib.axes._subplots.AxesSubplot at 0x7fe53afb8d50>
```

data['CreditScore'].mode()



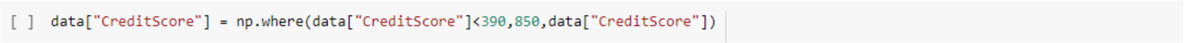
```
[ ] data['CreditScore'].mode()

0    850
dtype: int64
```

lower['CreditScore']

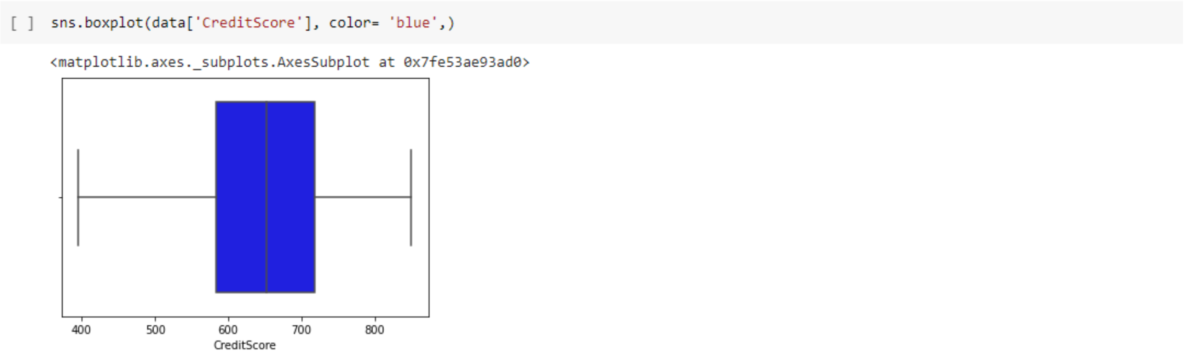


```
[ ] lower['CreditScore']

383.0
```

data["CreditScore"] = np.where(data["CreditScore"]<390,850,data["CreditScore"])
sns.boxplot(data['CreditScore'], color= 'blue',)



```
[ ]  data["CreditScore"] = np.where(data["CreditScore"]<390,850,data["CreditScore"])
```



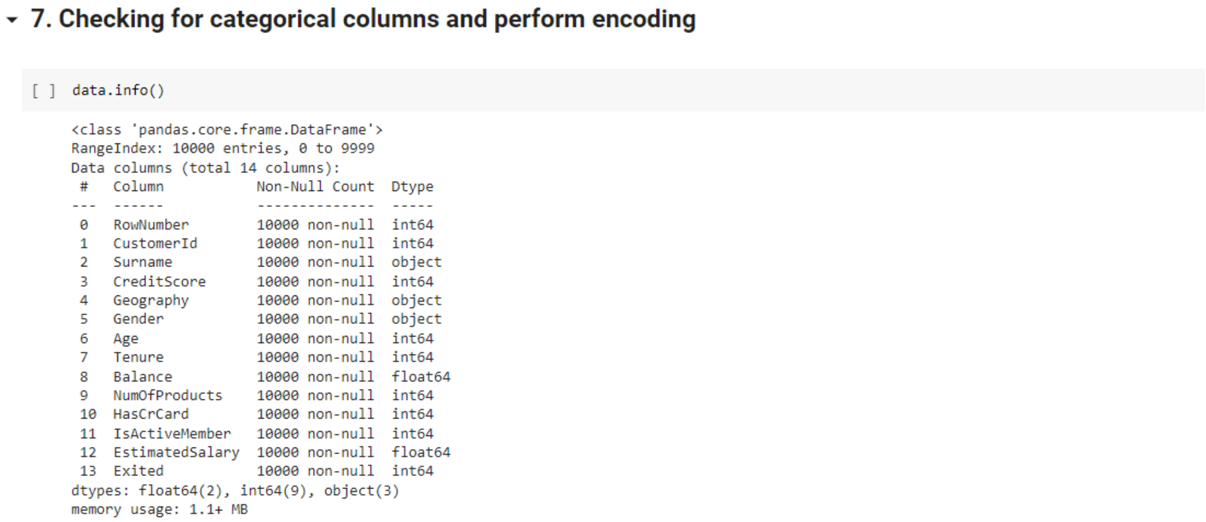
```
[ ]  sns.boxplot(data['CreditScore'], color= 'blue',)

     <matplotlib.axes._subplots.AxesSubplot at 0x7fe53ae93ad0>
```

Task 7:

7. Checking for categorical columns and perform encoding

Solution:

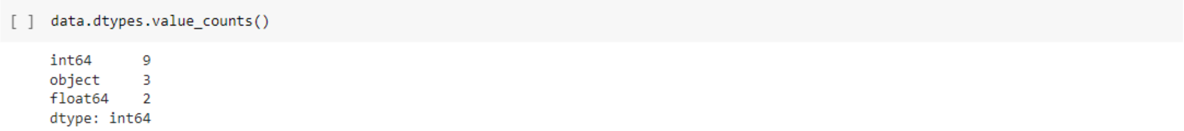
data.info()



7. Checking for categorical columns and perform encoding

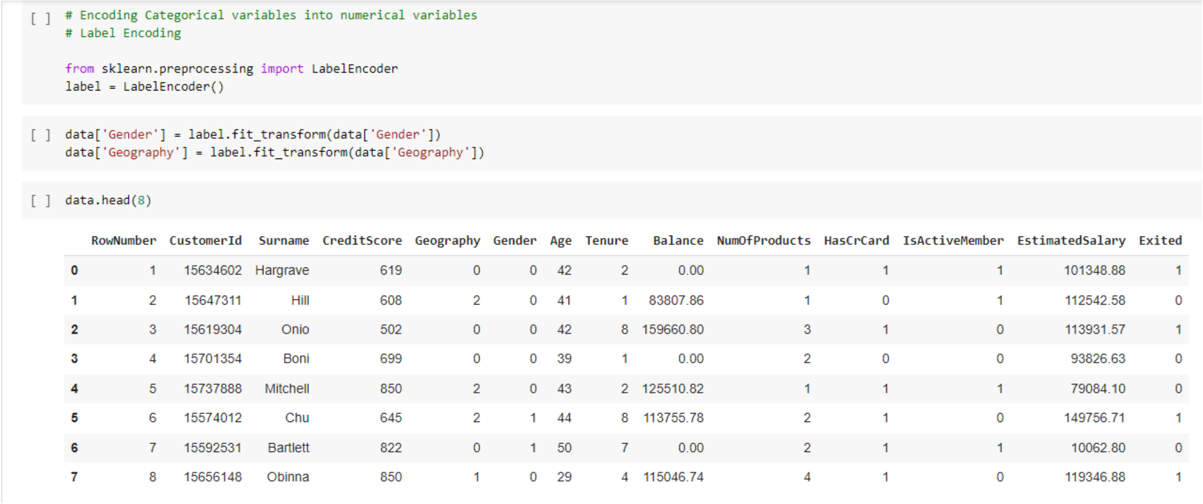
```
[ ]  data.info()

     <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
```

data.dtypes.value_counts()



```
[ ]  data.dtypes.value_counts()

     int64      9
     object     3
     float64    2
     dtype: int64
```

```
# Encoding Categorical variables into numerical variables'
# Label Encoding

from sklearn.preprocessing import LabelEncode
label = LabelEncoder()

data['Gender'] = label.fit_transform(data['Gender'])
data['Geography'] = label.fit_transform(data['Geography'])
data.head(8)
```



```
[ ] # Encoding Categorical variables into numerical variables
    # Label Encoding

    from sklearn.preprocessing import LabelEncoder
    label = LabelEncoder()
```

```
[ ] data['Gender'] = label.fit_transform(data['Gender'])
    data['Geography'] = label.fit_transform(data['Geography'])
```

```
[ ] data.head(8)
```

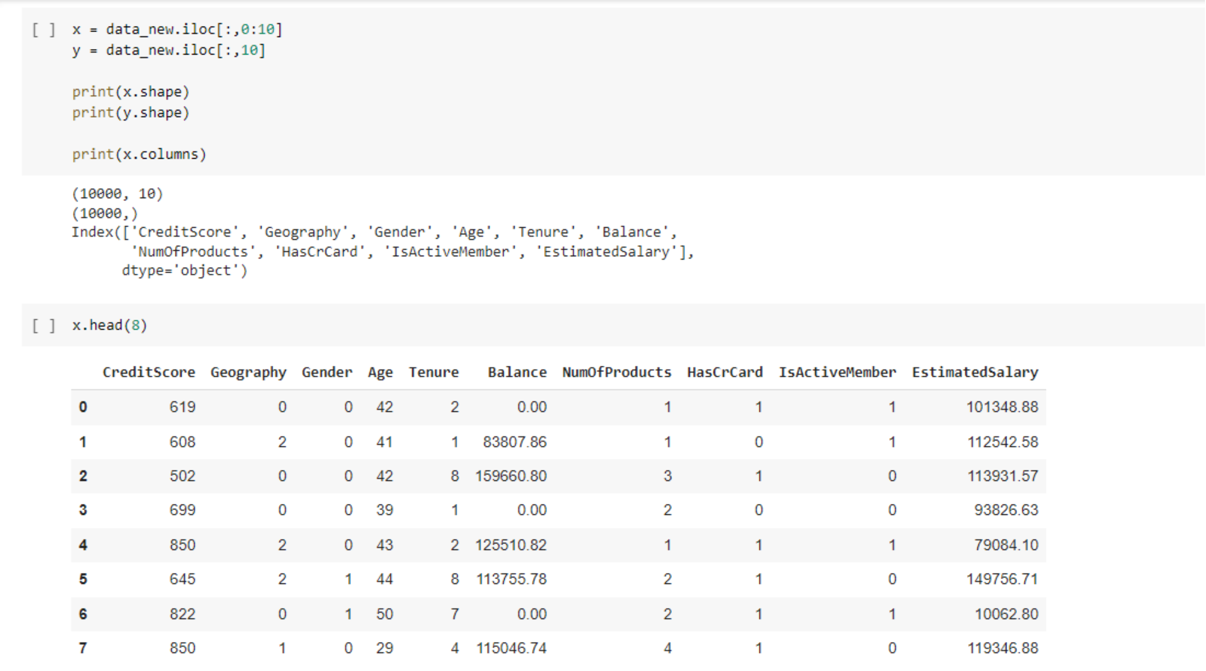
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	0	0	42	8	159660.80	3	1	0	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
5	6	15574012	Chu	645	2	1	44	8	113755.78	2	1	0	149756.71	1
6	7	15592531	Bartlett	822	0	1	50	7	0.00	2	1	1	10062.80	0
7	8	15656148	Obinna	850	1	0	29	4	115046.74	4	1	0	119346.88	1

Task 8:

8. Split the data into dependent and independent variables

Solution:

```
data_new = data.drop(['CustomerId', 'Surname', 'RowNumber'], axis = 1)
data_new.info()

data_new.shape

x = data_new.iloc[:,0:10]
y = data_new.iloc[:,10
print(x.shape)
print(y.shape)
print(x.columns)

x.head(8)
```



```
[ ] x = data_new.iloc[:,0:10]
    y = data_new.iloc[:,10]

    print(x.shape)
    print(y.shape)

    print(x.columns)
```

```
(10000, 10)
(10000,)
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary'],
      dtype='object')
```

```
[ ] x.head(8)
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	0	0	42	2	0.00	1	1	1	101348.88
1	608	2	0	41	1	83807.86	1	0	1	112542.58
2	502	0	0	42	8	159660.80	3	1	0	113931.57
3	699	0	0	39	1	0.00	2	0	0	93826.63
4	850	2	0	43	2	125510.82	1	1	1	79084.10
5	645	2	1	44	8	113755.78	2	1	0	149756.71
6	822	0	1	50	7	0.00	2	1	1	10062.80
7	850	1	0	29	4	115046.74	4	1	0	119346.88

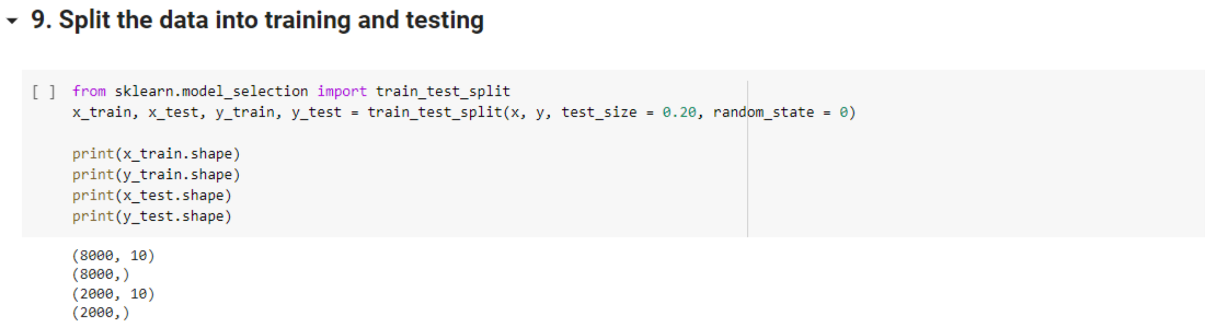
Task 9:

9. Split the data into training and testing

Solution:

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)



9. Split the data into training and testing

```
[ ] from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)

    print(x_train.shape)
    print(y_train.shape)
    print(x_test.shape)
    print(y_test.shape)
```

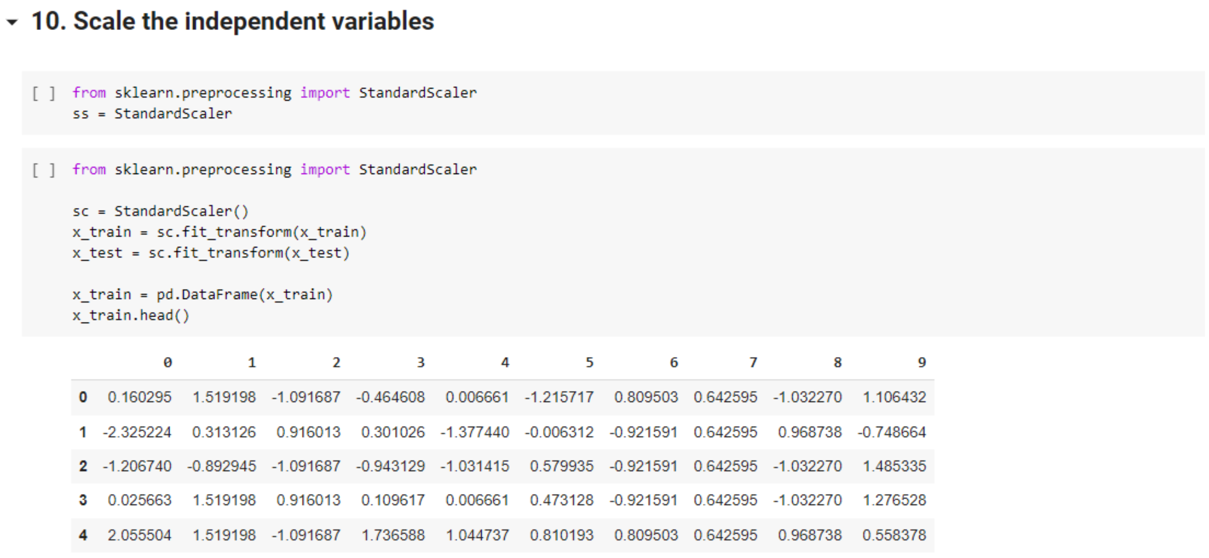
```
(8000, 10)
(8000,)
(2000, 10)
(2000,)
```

Task 10:

10. Scale the independent variables

Solution:

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)

x_train = pd.DataFrame(x_train)
x_train.head()
```



10. Scale the independent variables

```
[ ] from sklearn.preprocessing import StandardScaler
    ss = StandardScaler
```

```
[ ] from sklearn.preprocessing import StandardScaler

    sc = StandardScaler()
    x_train = sc.fit_transform(x_train)
    x_test = sc.fit_transform(x_test)

    x_train = pd.DataFrame(x_train)
    x_train.head()
```

	0	1	2	3	4	5	6	7	8	9
0	0.160295	1.519198	-1.091687	-0.464608	0.006661	-1.215717	0.809503	0.642595	-1.032270	1.106432
1	-2.325224	0.313126	0.916013	0.301026	-1.377440	-0.006312	-0.921591	0.642595	0.968738	-0.748664
2	-1.206740	-0.892945	-1.091687	-0.943129	-1.031415	0.579935	-0.921591	0.642595	-1.032270	1.485335
3	0.025663	1.519198	0.916013	0.109617	0.006661	0.473128	-0.921591	0.642595	-1.032270	1.276528
4	2.055504	1.519198	-1.091687	1.736588	1.044737	0.810193	0.809503	0.642595	0.968738	0.558378