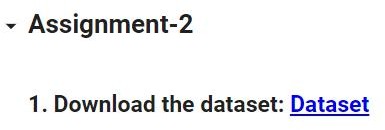
**Assignment -2**

Data Visualization and Pre-processing

|  |  |  |
| --- | --- | --- |
| Assignment Date | : | 15 October 2022 |
| Student Name | : | Kiruthiga M |
| Student Roll Number | : | 212219040061 |
| Maximum Marks | : | 2 Marks |

#### Task 1:

Download the dataset: Dataset



#### Task 2:

**Question-`1:**

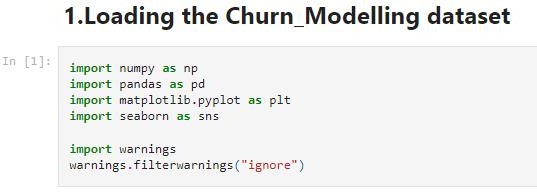
#### Loading the Churn\_Modelling dataset

Solution:

##### import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

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



Solution:

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

##### Output:



Solution:

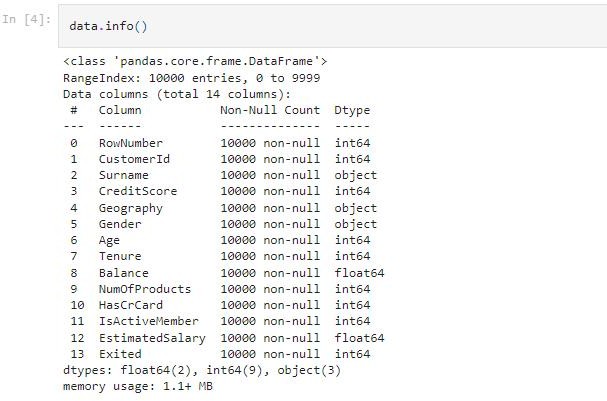
data = pd.read\_csv("/content/Churn\_Modelling.csv")

##### Output:



Solution:

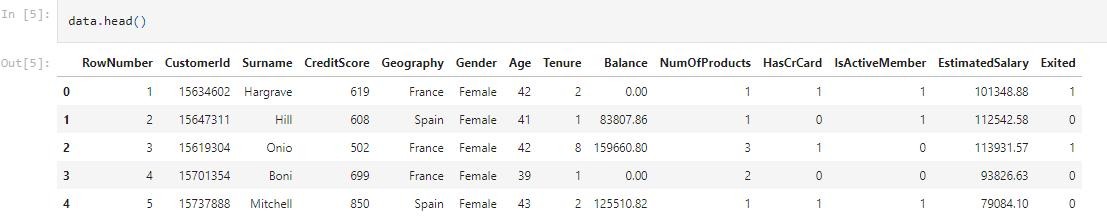
data.info()



Solution:

data.head()

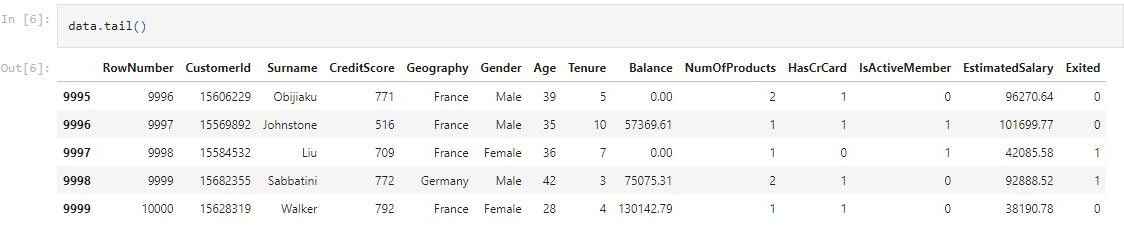
##### Output:



Solution:

data.tail()

##### Output:



Solution:

data.shape



Task 3:

**Question-2:**

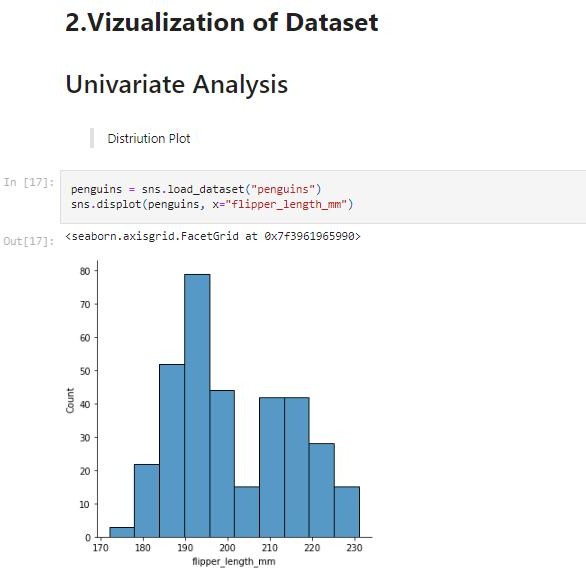
## Visualization of Dataset

### Univariate Analysis

* Distribution Plot

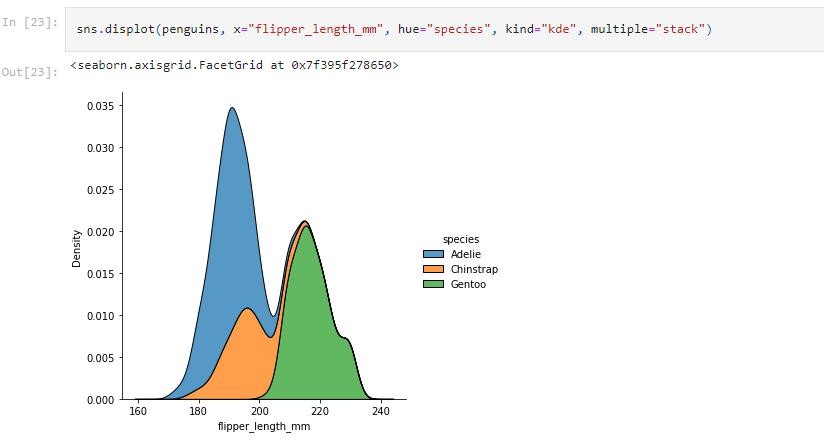
Solution:

penguins = sns.load\_dataset("penguins") sns.displot(penguins, x="flipper\_length\_mm")



Solution:

sns.displot(penguins, x="flipper\_length\_mm", hue="species", kind="kde", multiple="stack")

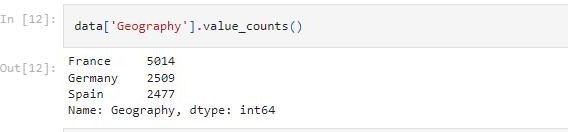


#### Histograms

Solution:

##### data['Geography'].value\_counts()

Output:

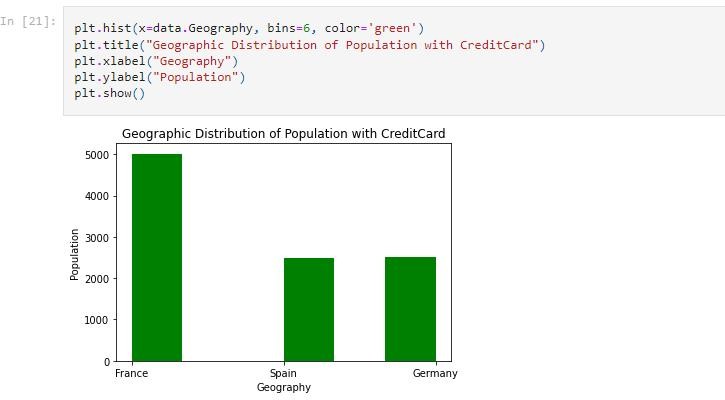


Solution:

##### 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()

##### Output:



Solution:

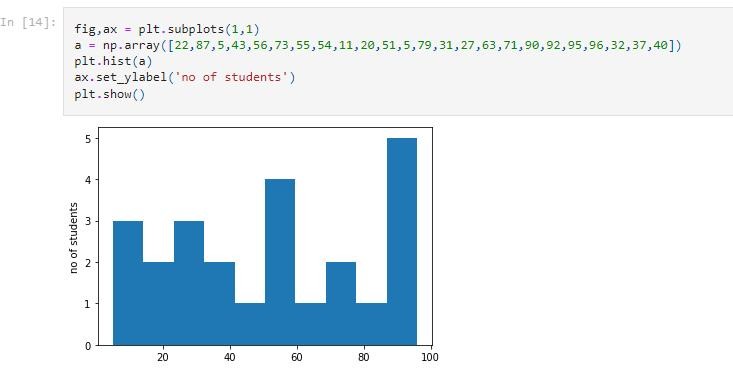
fig,ax = plt.subplots(1,1)

a = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27,63,71,90,92,95,96, 32,37,40])

plt.hist(a)

ax.set\_ylabel('no of students') plt.show()

Output:

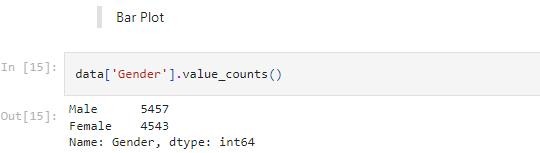


* **Bar Plot**

Solution:

data['Gender'].value\_counts()

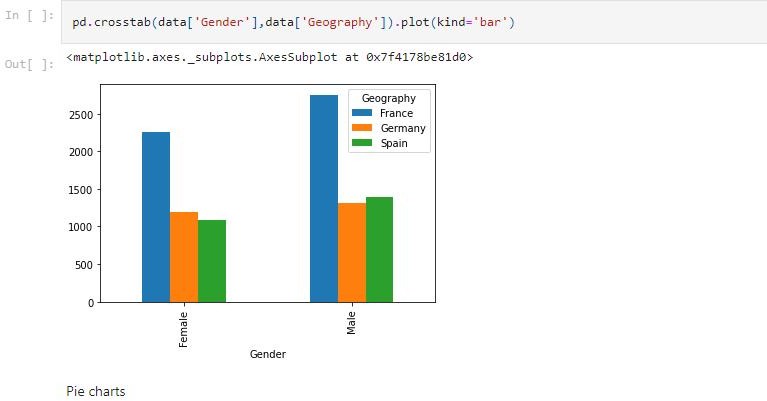
##### Output:



Solution:

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

Output:



* **Pie charts**

Solution:

##### fig=plt.figure() ax=fig.add\_axes([0,0,1,1])

courses=['machine learning','Web development','App development']

##### students\_enrolled=[50,37,42] ax.pie(students\_enrolled,labels=courses) plt.show()

Output:

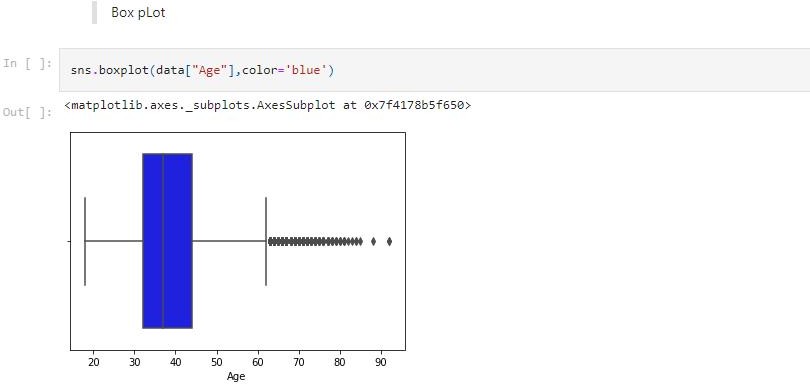


* **Box plot**

Solution:

sns.boxplot(data['Age'],color=’blue’)

Output:

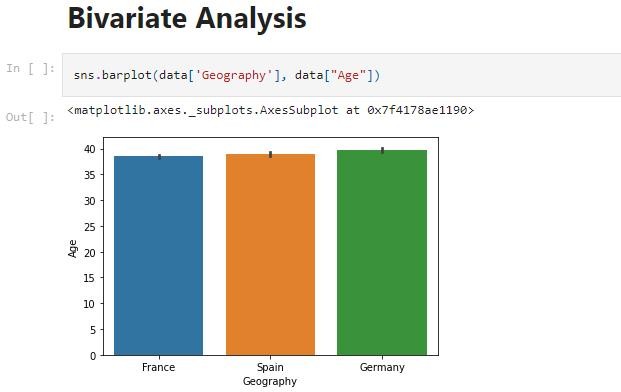


**Bivariate Analysis**

Solution:

sns.barplot(data[‘Geographgy’],data["Age"])

##### Output:



Solution:

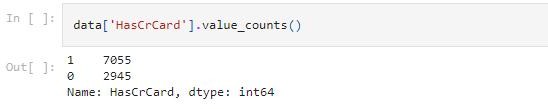
sns.barplot(data["NumOfProducts"],data["Age"])



Solution:

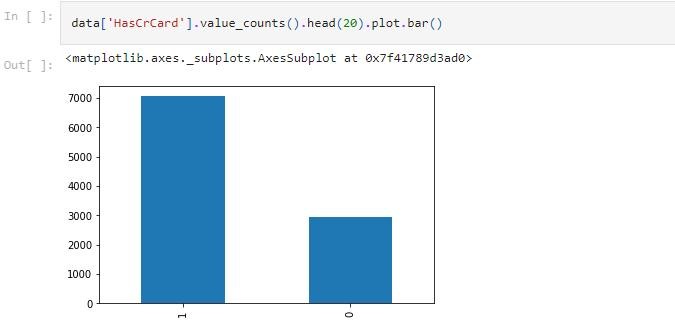
data['HasCrCard'].value\_counts()

##### Output:



Solution:

data['HasCrCard'].value\_counts().head(20).plot.bar()

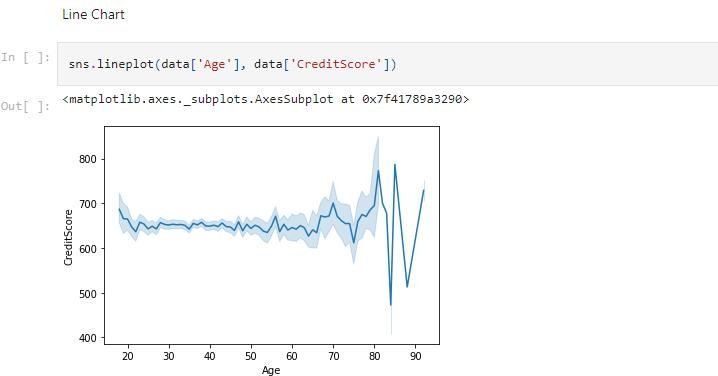


### Line Chart

Solution:

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

Output:



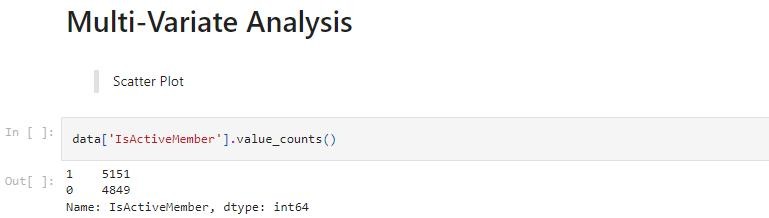
## Multi-Variate Analysis

* **Scatter Plot**

Solution:

data['IsActiveMember'].value\_counts()

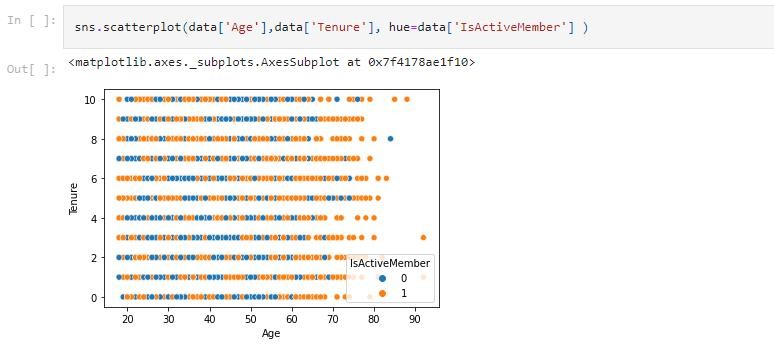
##### Output:



Solution:

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

Output:



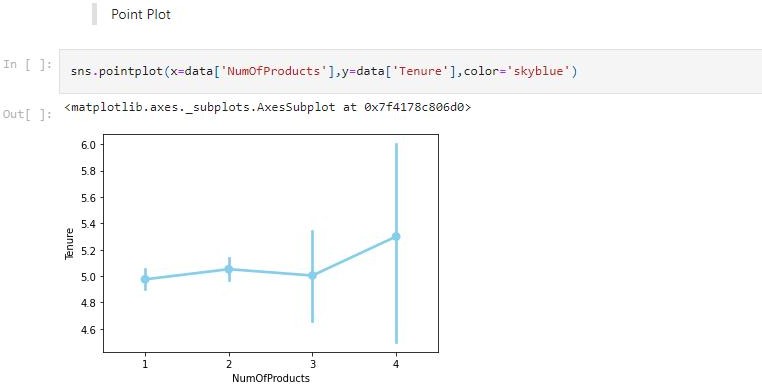
* + **Point Plot**

Solution:

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

='skyblue')

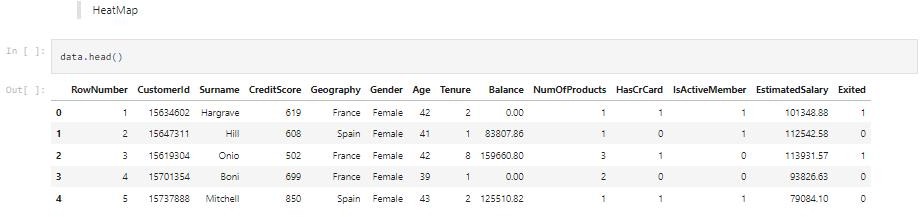
Output:



### HeatMap

Solution:

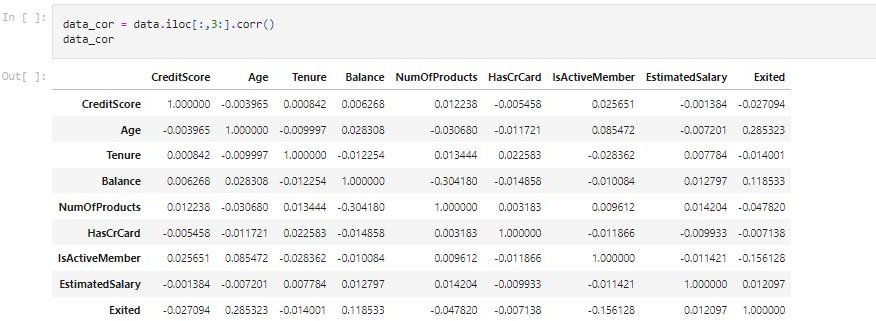
##### data.head()



Solution:

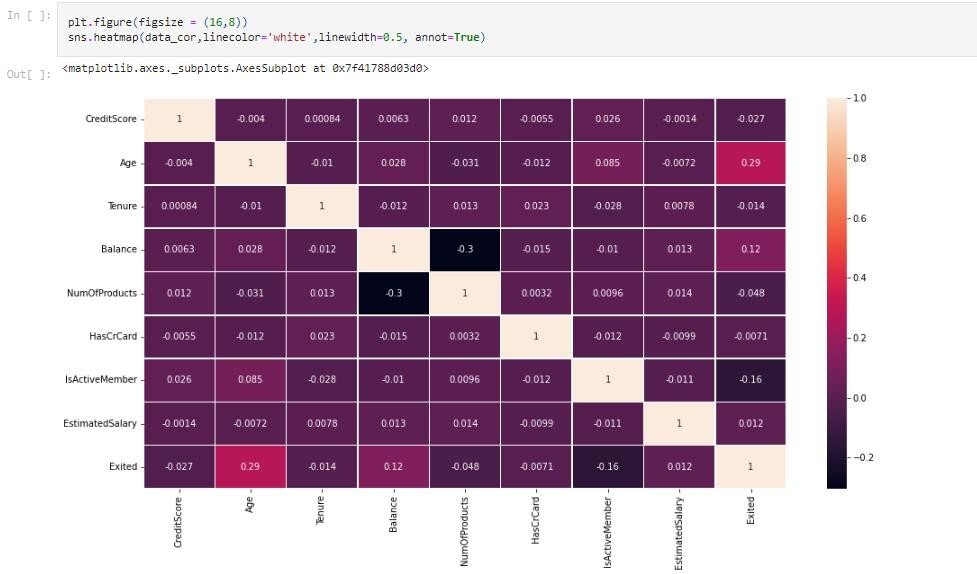
##### data\_cor = data.iloc[:,3:].corr() data\_cor

Output:



Solution:

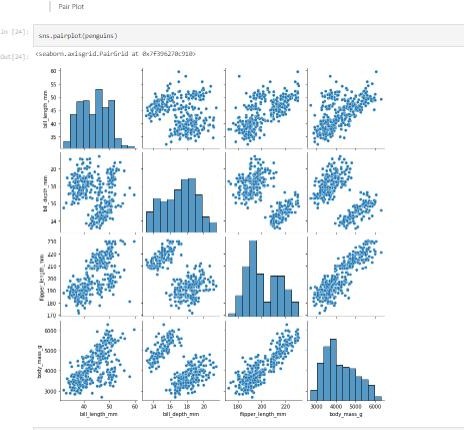
plt.figure(figsize = (16,8)) sns.heatmap(data\_cor,linecolor='white',linewidth=0.5, annot=True)



### Pair Plot

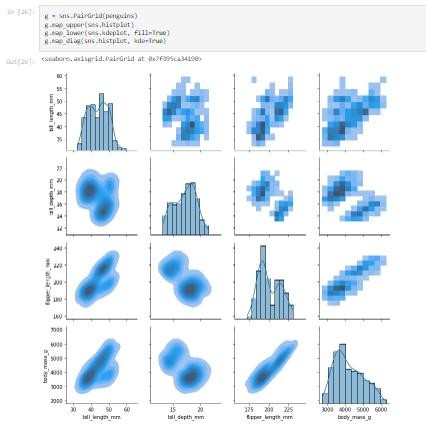
Solution:

##### sns.pairplot(penguins)



Solution:

g = sns.PairGrid(penguins) g.map\_upper(sns.histplot) g.map\_lower(sns.kdeplot, fill=True) g.map\_diag(sns.histplot, kde=True)



# Task 4:

**Question-3:**

**Descriptive Statistic Analysis**

#### Mean

1. Medium

#### Mode

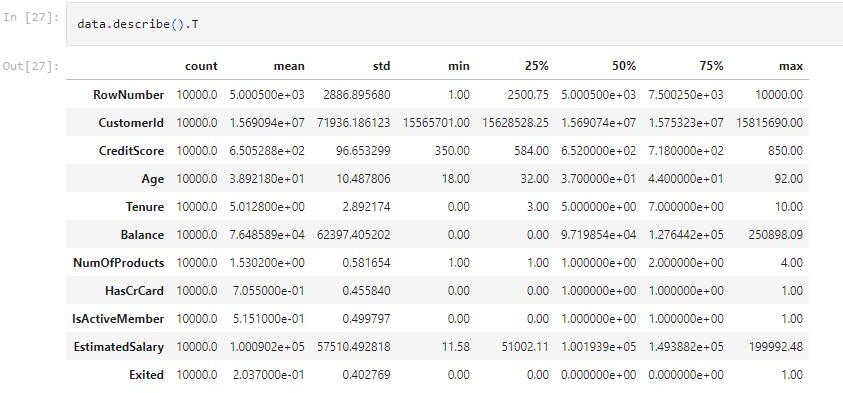
1. Standard Deviation

#### Variance

Solution:

data.describe().T

##### Output:



Solution:

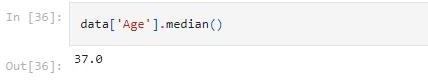
data['Age'].mean()

##### Output:



Solution:

data['Age'].median()



Solution:

data['Age'].mode()

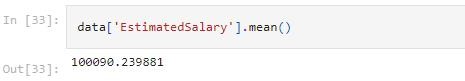
##### Output:



Solution:

data['EstimatedSalary'].mean()

##### Output:



Solution:

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



Solution:

data['EstimatedSalary'].mode())

##### Output:



Solution:

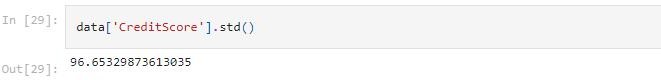
data['Balance'].mean()

##### Output:



Solution:

data['CreditScore'].std()



Solution:

data['Tenure'].var()

##### Output:



Task 5:

**Question-4:**

### Handling Missing Values

Solution:

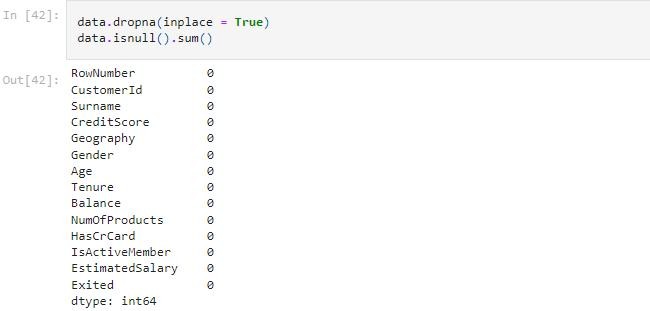
data.isna().any()



Solution:

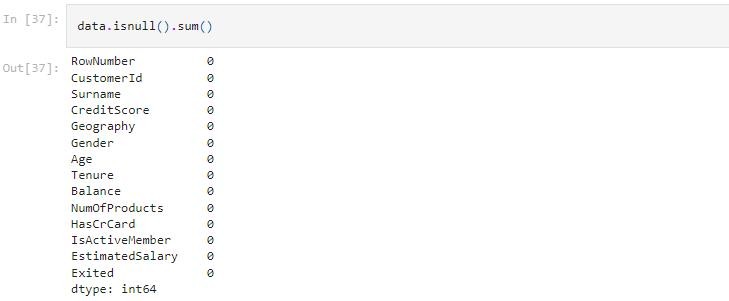
data.dropna(inplace = True) data.isnull().sum()

##### Output:



data.isnull().sum()

Output:



# Task 6:

**Question-5:**

**Finding Outliers and Replacing Them**

Solution:

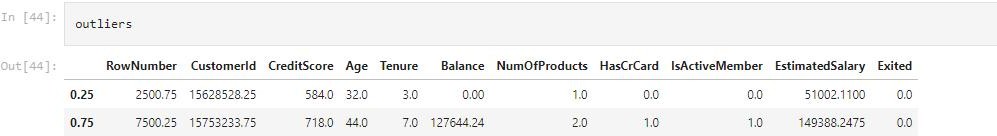
outliers = data.quantile(q=(0.25,0.75))

##### Output:



Outliers

##### Output:



Solution:

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

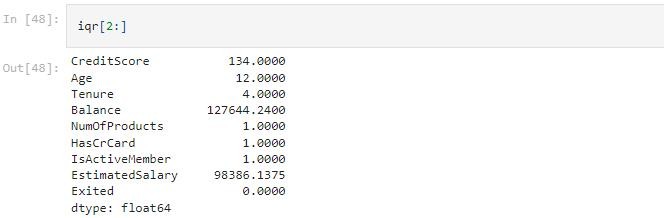
##### Output:



Solution:

iqr[2:]

##### Output:



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

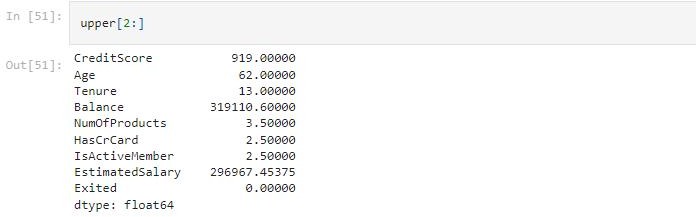
##### Output:



Solution:

upper[2:]

##### Output:



Solution:

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

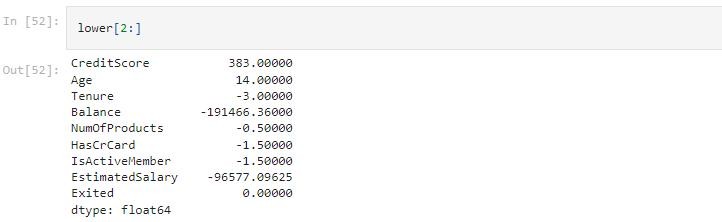
##### Output:



Solution:

lower[2:]

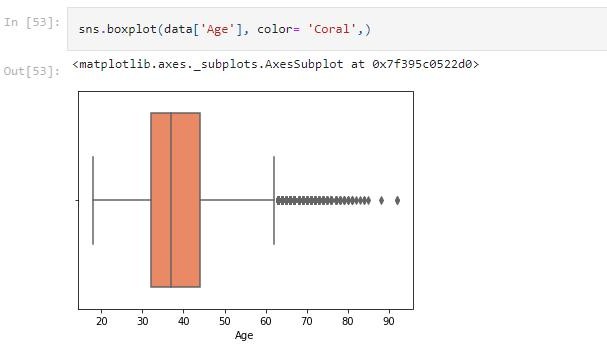
##### Output:



Solution:

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

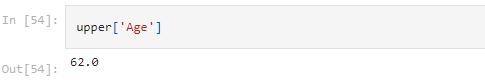
##### Output:



Solution:

upper['Age']

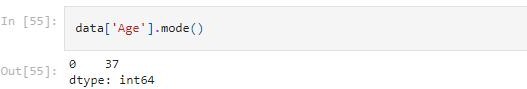
##### Output:



Solution:

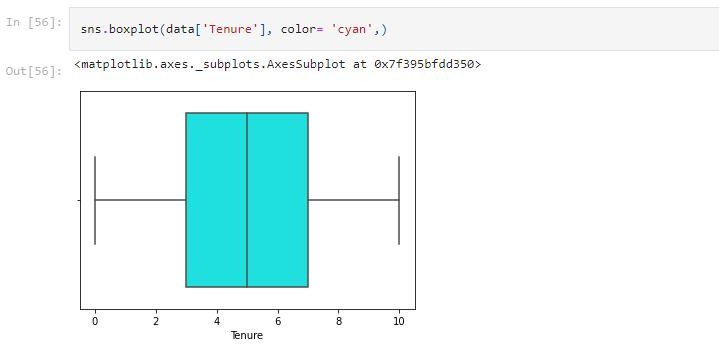
data['Age'].mode()

##### Output:



Solution:

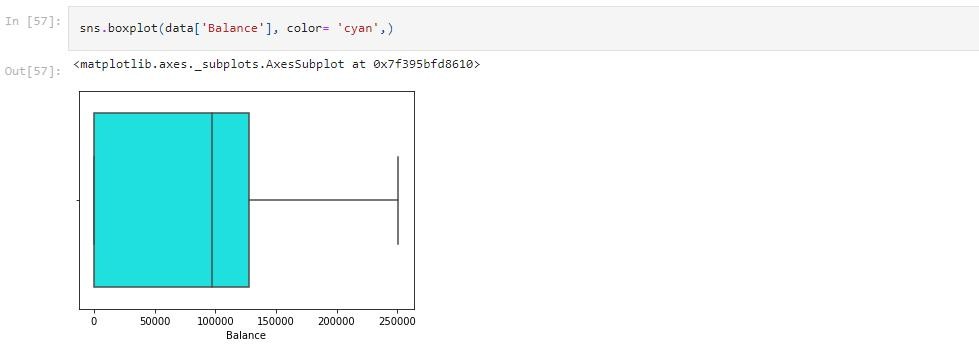
sns.boxplot(data['Tenure'], color= 'cyan',)



Solution:

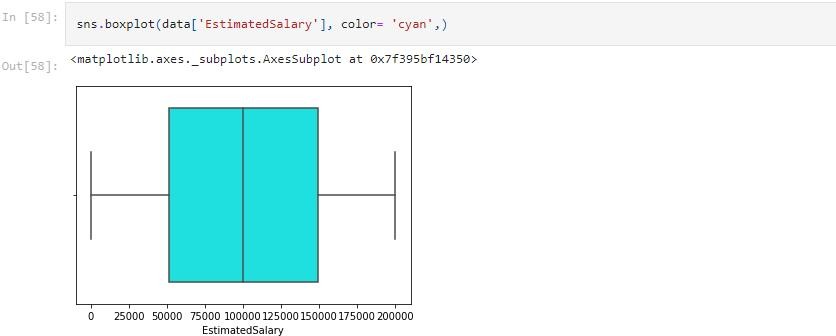
sns.boxplot(data['Balance'], color= 'cyan',)

##### Output:



Solution:

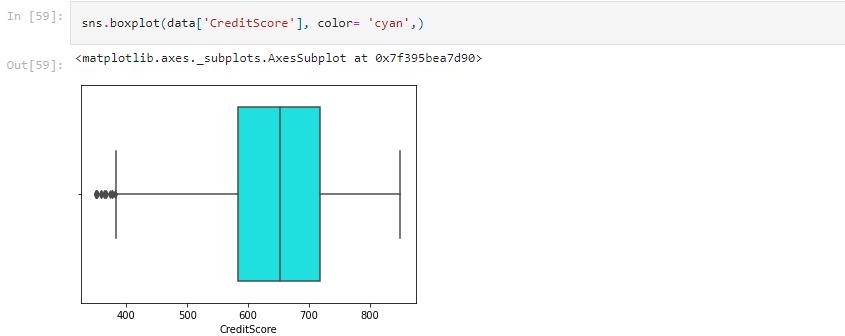
sns.boxplot(data['Estimatedsalary'], color= 'cyan',)



Solution:

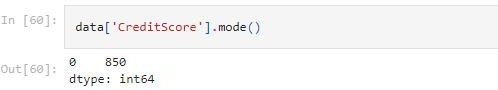
sns.boxplot(data['CreditScore'], color= 'cyan',)

##### Output:



Solution:

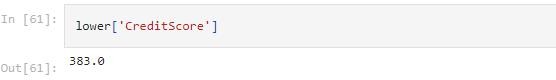
data['CreditScore'].mode()



Solution:

lower['CreditScore']

##### Output:



Solution:

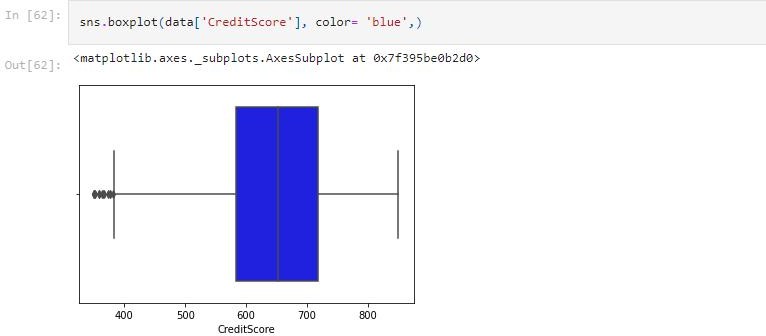
data["CreditScore"] = np.where(data["CreditScore"]<390,850,data["CreditScore"])

##### Output:



Solution:

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



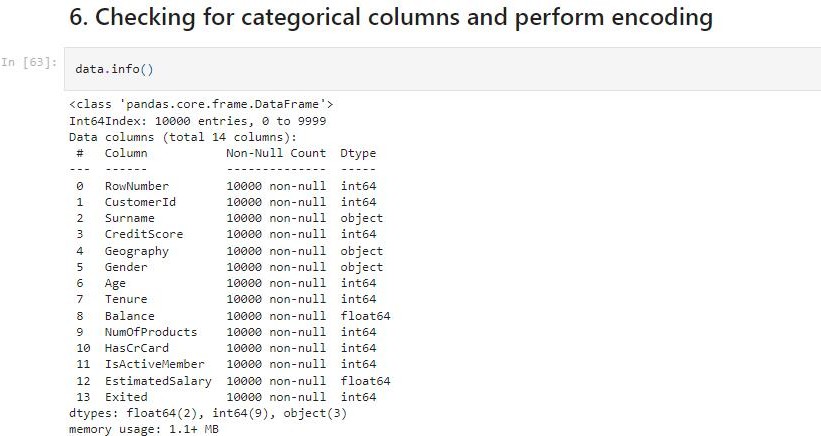
# Task 7:

**Question-6:**

**Checking for categorical columns and perform encoding**

Solution:

data.info()



Solution:

data.dtypes.value\_counts()

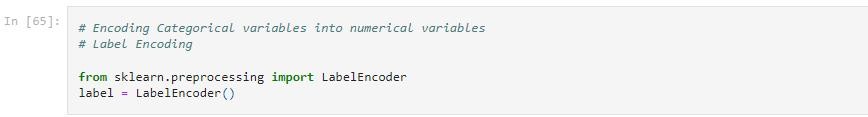
##### Output:



Solution:

# Encoding Categorical variables into numerical variables' # Label Encoding

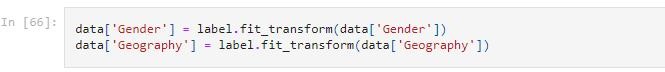
from sklearn.preprocessing import LabelEncode label = LabelEncoder()



Solution:

data['Gender'] = label.fit\_transform(data['Gender']) data['Geography'] = label.fit\_transform(data['Geography'])

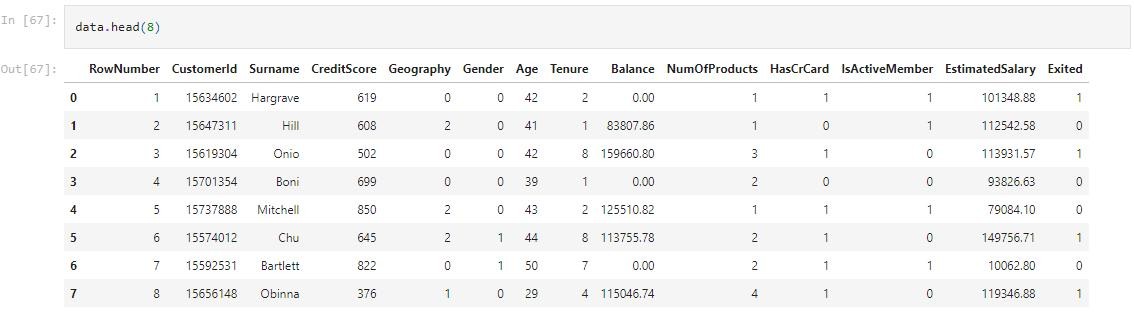
##### Output:



Solution:

data.head(8)

Output:



# Task 8:

**Question-7:**

**Split the data into dependent and independent variables**

Solution:

data\_new = data.drop(['CustomerId', 'Surname', 'RowNumber'], axis = 1)

data\_new.info()

##### Output:



Solution:

data\_new.shape

##### Output:



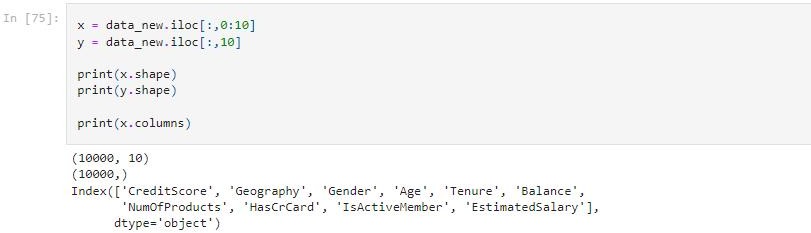
Solution:

x = data\_new.iloc[:,0:10] y = data\_new.iloc[:,10

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

print(x.columns)

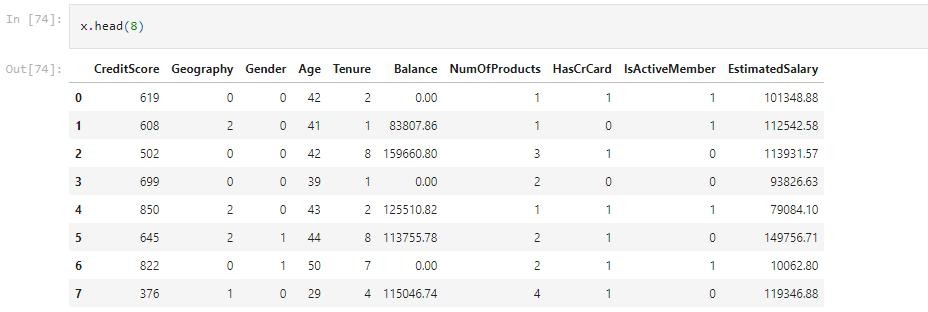
##### Output:



Solution:

x.head(8)

Output:



**Question-8:**

### 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)

Output:



**Question-9:**

### Scale the independent variables

Solution:

from sklearn.preprocessing import StandardScaler ss = StandardScaler

##### Output:



Solution:

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()

##### Output:

