```
#1,2) Downloaded and Loaded the given Dataset

import pandas as pd
import numpy as np
#For Plotting
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from scipy import stats
from IPython.display import display, HTML
import plotly.express as px
df = pd.read_csv('Churn_Modelling.csv')
df.head()

#3) a) UNIVARIATE ANALYSIS:

plt.hist(df['Age'])
sns.countplot(df['Age'])
sns.set(rc={'figure.figsize':(22,5)})
sns.countplot(df['Geography'])

#3) b) BIVARIATE ANALYSIS:

plt.scatter(df['EstimatedSalary']
df['Age'])sns.heatmap(df.corr(), annot = True)
plt.scatter( df['Balance'], df['NumOfProducts'])
plt.xlabel('Balance')
plt.ylabel('NumofProducts')
sns.countplot(data=df,x='Geography', hue='Gender')
sns.set(rc={'figure.figsize':(10,5)})
sns.countplot(data=df, x='HasCrCard', hue='Gender')

#3)c) MULTIVARIATE ANALYSIS

sns.pairplot(data = df)

#4) DESCRIPTIVE ANALYSIS

df.describe()
df.info()
df.count()

#5) Handling Missing Values

df.isnull().sum()

#6) Finding Outliers and replacing the outliers

sns.boxplot(df['Age'])
sns.boxplot(df['CreditScore'])
df.columns
Q1=df['Age'].quantile(0.25)
Q3=df['Age'].quantile(0.75)
```

```
IQR=Q3-Q1
whisker_width = 1.5
age_outliers = df[(df['Age'] < Q1 - whisker_width*IQR) | (df['Age'] > Q3
+ whisker_width*IQR)]
age_outliers.count()
import numpy as np
lower_whisker = Q1 -(whisker_width*IQR)
upper_whisker = Q3 + (whisker_width*IQR)
df['Age']=np.where(df['Age']>upper_whisker,upper_whisker,np.where(df['Age
']<lower_whisker,lower_whisker,df['Age']))
sns.boxplot(df['Age'],data=df)
# CreditScore
Q1=df['CreditScore'].quantile(0.25)
Q3=df['CreditScore'].quantile(0.75)
IQR=Q3-Q1
whisker_width = 1.5
credscore_outliers = df[(df['CreditScore'] < Q1 - whisker_width*IQR) |
(df['CreditScore'] > Q3 + whisker_width*IQR)]
credscore_outliers.count()
lower_whisker = Q1 -(whisker_width*IQR)
upper_whisker = Q3 + (whisker_width*IQR)
df['CreditScore']=np.where(df['CreditScore']>upper_whisker,upper_whisker,
np.where(df['CreditScore']<lower_whisker,lower_whisker,df['CreditScore'])
)
sns.boxplot(df['CreditScore'])
df['CreditScore'].count()

#7) Checking for categorical columns:

df.info()
from sklearn.preprocessing import LabelEncoder
enc = LabelEncoder()
df['Geography'] = enc.fit_transform(df['Geography'])
df['Gender'] = enc.fit_transform(df[['Gender']])
df['Geography'].unique(), df['Gender'].unique()
df.head()

#8) SPLITTING DEPENDENT AND INDEPENDENT VARIABLES:

X = df.iloc[:,3:-1]
X.head()
y = df.iloc[:,-1]

#9) Scaling the Independent Variables:

from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
X = scale.fit_transform(X)
X

#10) Splitting the test and train variables

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y ,random_state=1,
shuffle=True, train_size = 0.8)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
from sklearn import linear_model
reg = linear_model.LogisticRegression()
```

```
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)
rom sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```