Assignment – 2

Data Visualization and Pre-processing

Assignment Date	26 September 2022
Student Name	Afrin Nirmala
Student Roll Number	2019PITIT103
Maximum Marks	2 Marks

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

1. Load the dataset.

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

```
df
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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

```
df.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

```
df.shape
```

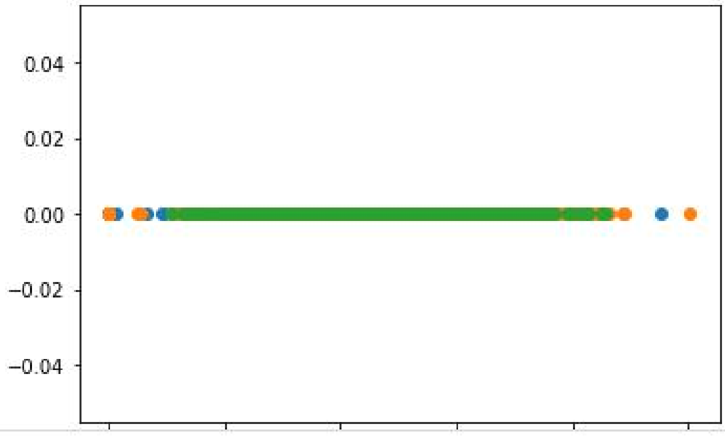
```
(10000, 14)
```

2. Perform Below Visualizations

a. Univariate Analysis

```
df_france=df.loc[df['Geography']=='France']
df_spain=df.loc[df['Geography']=='Spain']
df_germany=df.loc[df['Geography']=='Germany']
```

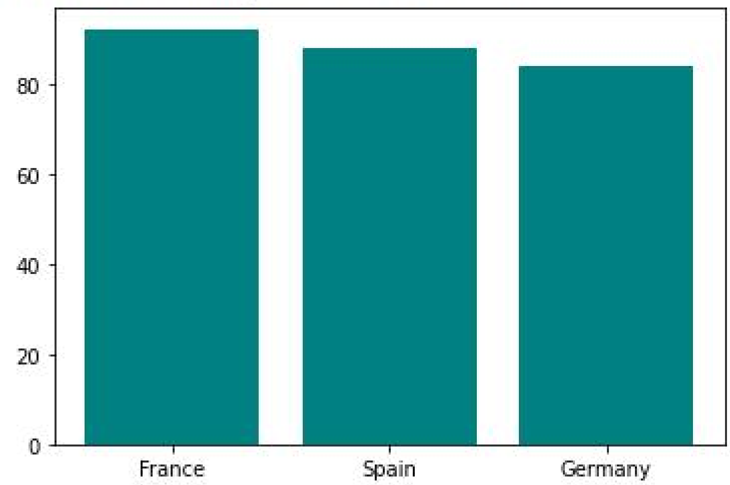
```
plt.plot(df_france['Balance'],np.zeros_like(df_france['Balance']),'o')
plt.plot(df_spain['Balance'],np.zeros_like(df_spain['Balance']),'o')
plt.plot(df_germany['Balance'],np.zeros_like(df_germany['Balance']),'o')
plt.xlabel('Age')
plt.show()
```



b. *Bivariate Analysis*

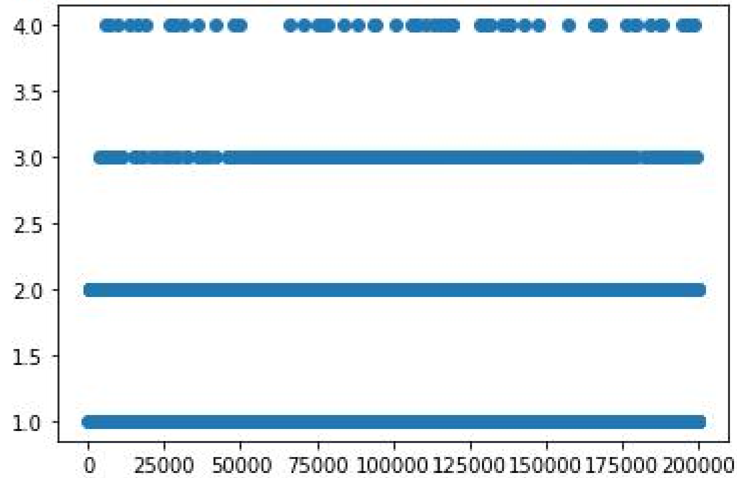
```
plt.bar('Geography','Age',data=df,color='teal')
```

```
<BarContainer object of 10000 artists>
```



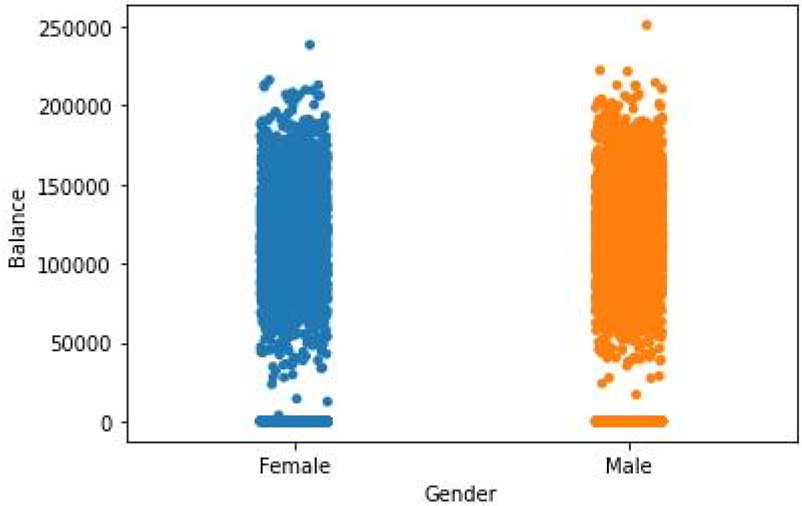
```python
plt.scatter('EstimatedSalary','NumOfProducts',data=df)
```

<matplotlib.collections.PathCollection at 0x7fcc81cb4c50>



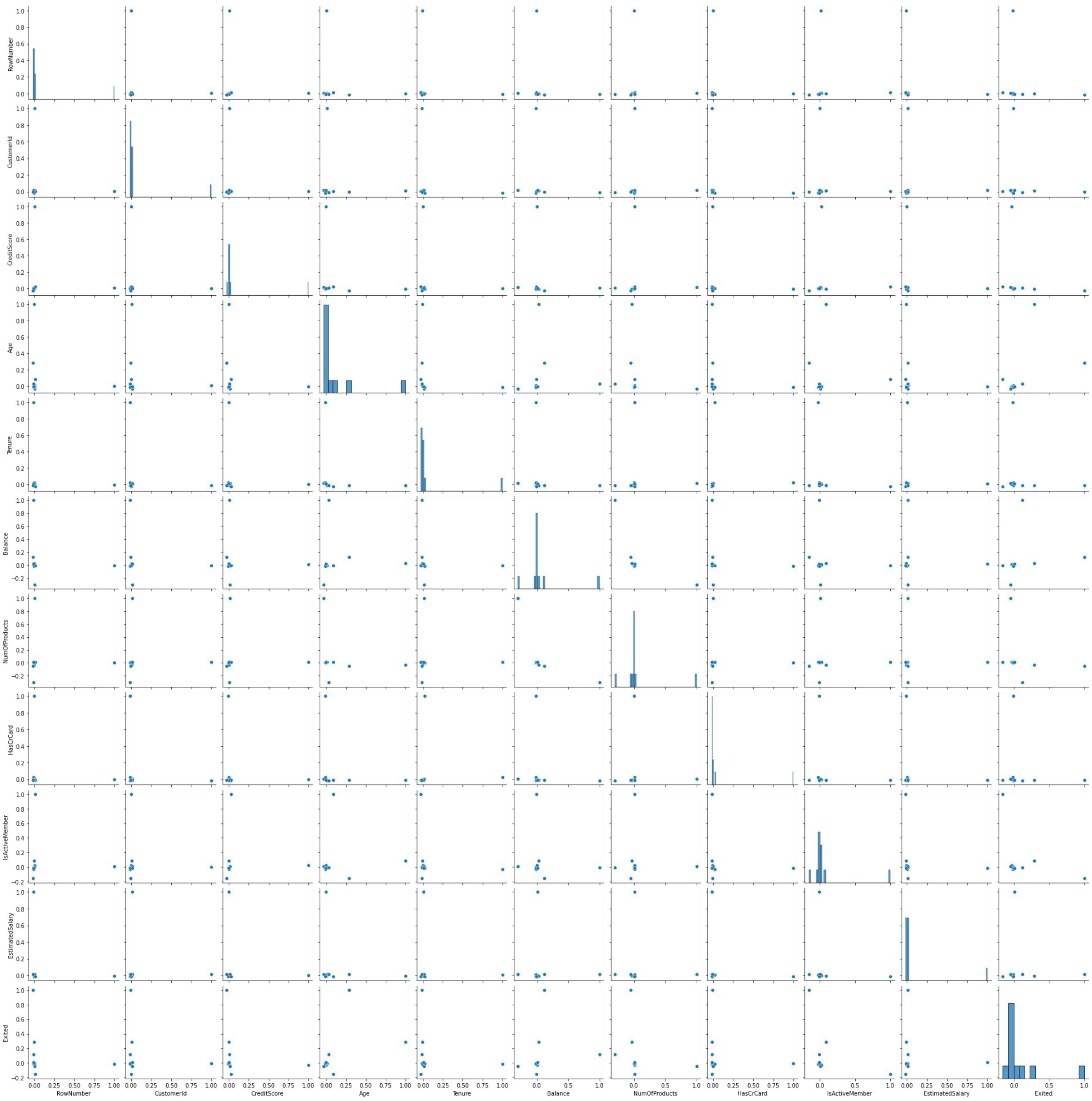

```python
sns.stripplot(x='Gender',y='Balance',data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fcc81b965d0>

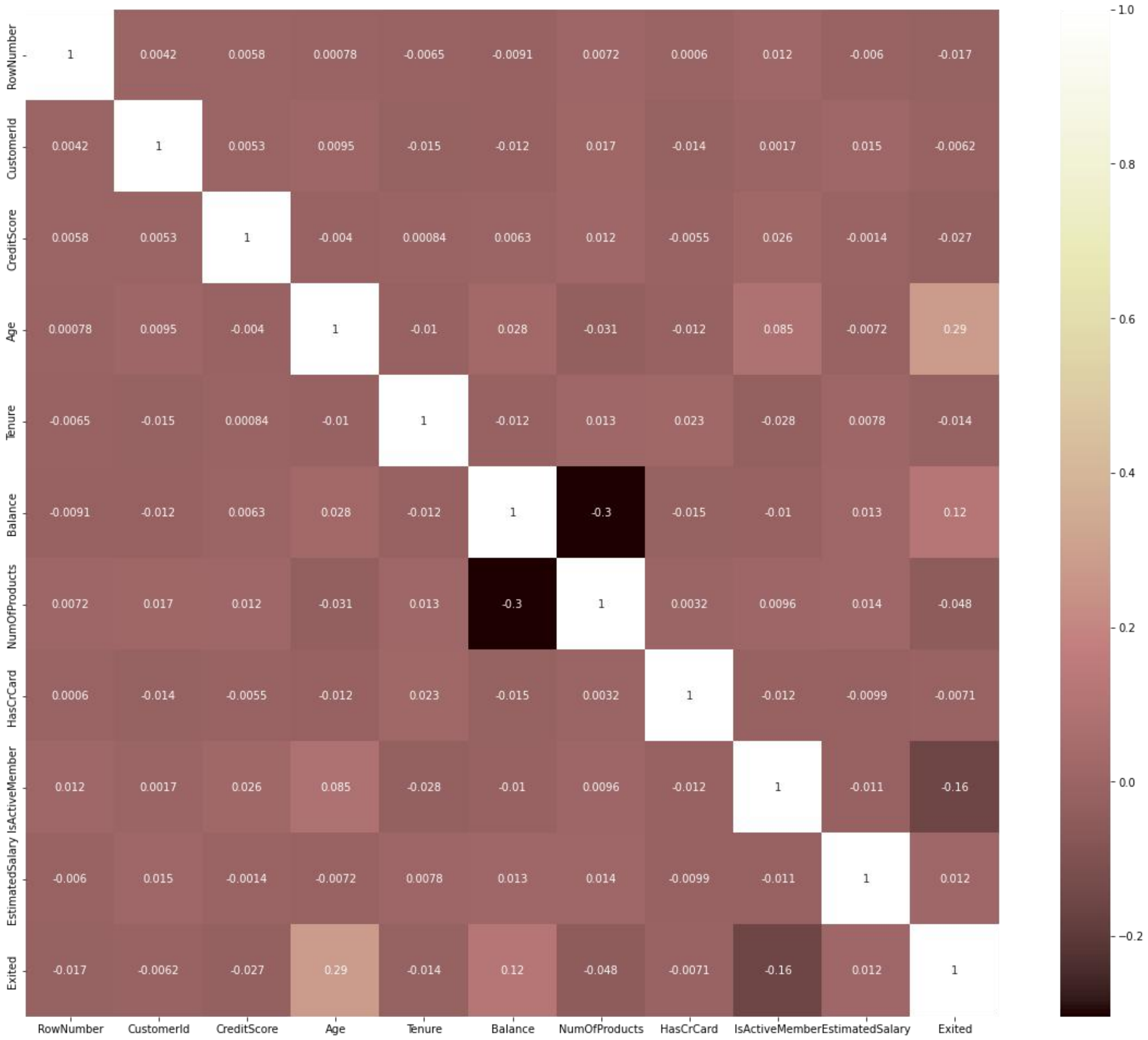


3. Multivariate Analysis

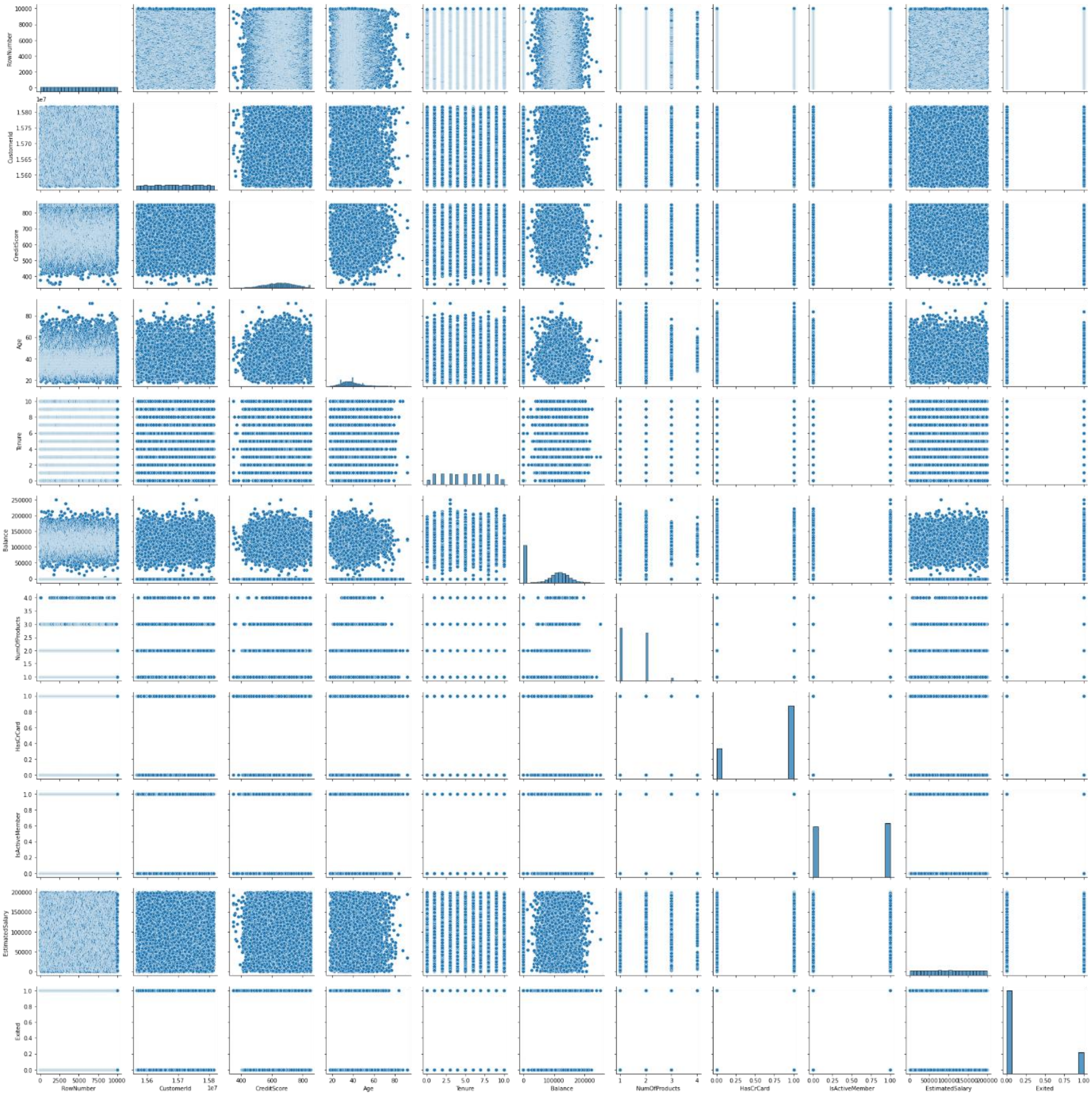
```
sns.pairplot(df.corr())
```



```
plt.subplots(figsize=(20,17))
sns.heatmap(df.corr(),annot=True,cmap='pink')
```



```
sns.pairplot(df)
```



4. Perform descriptive statistics on the dataset

```python
df.describe(include='all')
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
count	10000.00000	1.000000e+04	10000	10000.000000	10000	10000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000
unique	NaN	NaN	2932	NaN	3	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	NaN	Smith	NaN	France	Male	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	NaN	32	NaN	5014	5457	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	5000.50000	1.569094e+07	NaN	650.528800	NaN	NaN	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881
std	2886.89568	7.193619e+04	NaN	96.653299	NaN	NaN	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818
min	1.00000	1.556570e+07	NaN	350.000000	NaN	NaN	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000
25%	2500.75000	1.562853e+07	NaN	584.000000	NaN	NaN	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000
50%	5000.50000	1.569074e+07	NaN	652.000000	NaN	NaN	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000
75%	7500.25000	1.575323e+07	NaN	718.000000	NaN	NaN	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500
max	10000.00000	1.581569e+07	NaN	850.000000	NaN	NaN	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000

5. Handle the Missing values

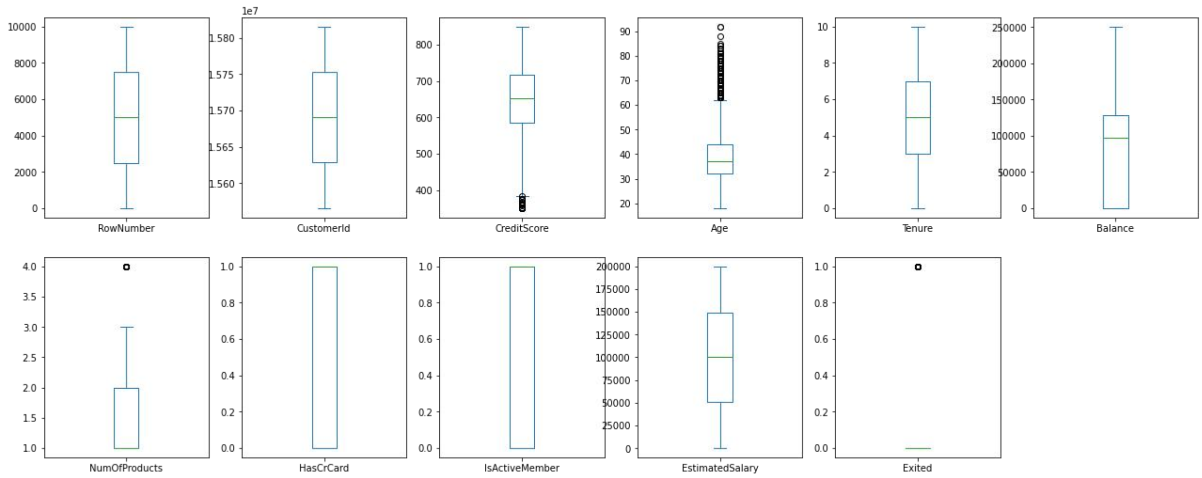
```python
df.isnull()
```

	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

6. Find the outliers and replace the outliers

```
df.plot(subplots=True,layout=(4,6),kind='box',figsize=(22,18))
```

```
RowNumber             AxesSubplot(0.125,0.71587;0.110714x0.16413)
CustomerId         AxesSubplot(0.257857,0.71587;0.110714x0.16413)
CreditScore        AxesSubplot(0.390714,0.71587;0.110714x0.16413)
Age                AxesSubplot(0.523571,0.71587;0.110714x0.16413)
Tenure             AxesSubplot(0.656429,0.71587;0.110714x0.16413)
Balance            AxesSubplot(0.789286,0.71587;0.110714x0.16413)
NumOfProducts        AxesSubplot(0.125,0.518913;0.110714x0.16413)
HasCrCard         AxesSubplot(0.257857,0.518913;0.110714x0.16413)
IsActiveMember    AxesSubplot(0.390714,0.518913;0.110714x0.16413)
EstimatedSalary   AxesSubplot(0.523571,0.518913;0.110714x0.16413)
Exited            AxesSubplot(0.656429,0.518913;0.110714x0.16413)
dtype: object
```

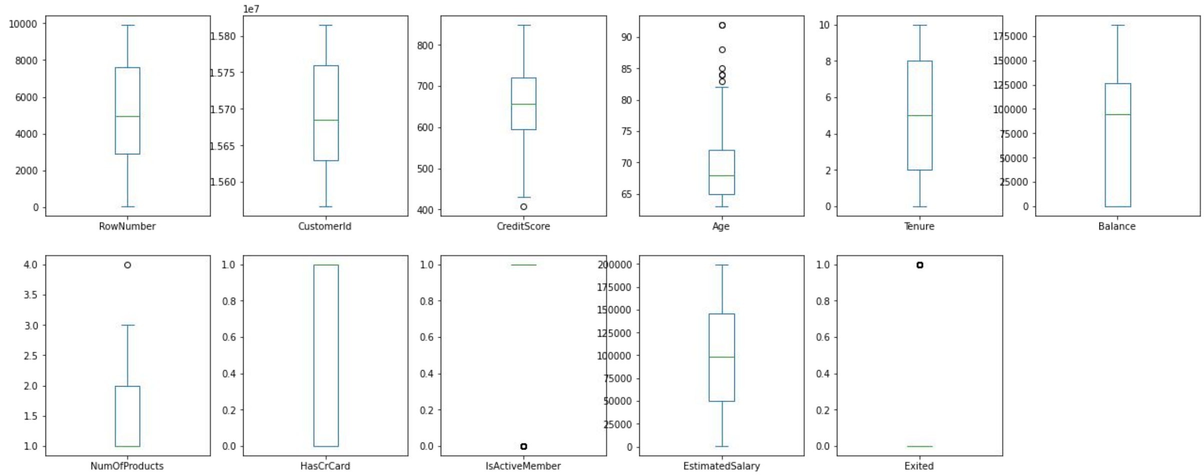



```
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1
n = 1.5
df = df[(df['Age'] < Q1 - n*IQR) | (df['Age'] > Q3 + n*IQR)]
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
58	59	15623944	T'ien	511	Spain	Female	66	4	0.00	1	1	0	1643.11	1
85	86	15805254	Ndukaku	652	Spain	Female	75	10	0.00	2	1	1	114675.75	0
104	105	15804919	Dunbabin	670	Spain	Female	65	1	0.00	1	1	1	177655.68	1
158	159	15589975	Maclean	646	France	Female	73	6	97259.25	1	0	1	104719.66	0
181	182	15789669	Hsia	510	France	Male	65	2	0.00	2	1	1	48071.61	0

```
df.plot(subplots=True,layout=(4,6),kind='box',figsize=(22,18))
```

```
RowNumber             AxesSubplot(0.125,0.71587;0.110714x0.16413)
CustomerId         AxesSubplot(0.257857,0.71587;0.110714x0.16413)
CreditScore        AxesSubplot(0.390714,0.71587;0.110714x0.16413)
Age                AxesSubplot(0.523571,0.71587;0.110714x0.16413)
Tenure             AxesSubplot(0.656429,0.71587;0.110714x0.16413)
Balance            AxesSubplot(0.789286,0.71587;0.110714x0.16413)
NumOfProducts        AxesSubplot(0.125,0.518913;0.110714x0.16413)
HasCrCard         AxesSubplot(0.257857,0.518913;0.110714x0.16413)
IsActiveMember    AxesSubplot(0.390714,0.518913;0.110714x0.16413)
EstimatedSalary   AxesSubplot(0.523571,0.518913;0.110714x0.16413)
Exited            AxesSubplot(0.656429,0.518913;0.110714x0.16413)
dtype: object
```



7. Check for Categorical columns and perform encoding

```
df.dtypes
```

```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

```python
df['Geography'].unique()
```

```
array(['Spain', 'France', 'Germany'], dtype=object)
```

```python
Geography=pd.get_dummies(df["Geography"])
df=pd.concat([df,Geography],axis=1)
df.drop(["Geography"],axis=1,inplace=True)
```

```python
df['Gender'].unique()
```

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

```python
Gender=pd.get_dummies(df["Gender"])
df=pd.concat([df,Gender],axis=1)
df.drop(["Gender"],axis=1,inplace=True)
```

```python
df.columns
```

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Age', 'Tenure',
       'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember',
       'EstimatedSalary', 'Exited', 'France', 'Germany', 'Spain', 'Female',
       'Male'],
      dtype='object')
```

```python
df['Surname'].unique()
```

```
array(["T'ien", 'Ndukaku', 'Dunbabin', 'Maclean', 'Hsia', 'Ringrose',
       'Smith', 'McIntosh', 'Matveyeva', 'Pokrovskii', 'Madukaego',
       'Cole', 'Black', 'Outhwaite', 'Ritchie', 'Teng', 'Stout',
       'Crawford', 'Fleming', 'Pugliesi', 'Romani', 'Williamson',
       'McKenzie', 'Cox', 'Alexander', 'Lloyd', 'Landry', 'Herrera',
       'Artemieva', 'Blackburn', 'Wood', "Ts'ui", 'Nepean', 'Sinclair',
       'Sanders', "Ch'iu", 'McKay', 'Lampungmeiua', 'Mitchell',
       'Lucchesi', 'Pan', 'Kung', 'Lo', 'Hsiao', 'Nnamutaezinwa',
       'Avdeeva', 'Byrne', 'Hardy', "P'eng", 'Simmons', 'Shelton',
       'Macarthur', 'Gordon', 'Hsueh', 'Sopuluchi', 'Forbes',
       'Zikoranachidimma', 'Moore', 'Mason', 'Miller', "O'Toole", 'Bogle',
       'Rosas', 'Blair', 'Shen', 'Melton', 'Oluchi', 'Kapustina', 'Hay',
       'Milne', 'Anderson', 'Becker', 'Marchesi', 'Kao', 'Rearick',
       'Genovesi', 'Hung', 'Walker', 'Pagnotto', 'Poole', 'Yuriev',
       'Iredale', 'Palerma', 'Brim', 'Powell', 'Calabrese', 'Kibble',
       'Lavrov', 'Chiekwugo', 'Yao', 'Leibius', 'Chung', 'Azikiwe',
       'Chimaoke', 'Chao', 'Fulks', 'McMorran', 'Hotchin', 'Hughes',
       'Donaldson', 'Tuan', 'Liu', 'Estep', 'Alexeyeva', 'Kilgour',
       'Lori', 'Knowles', 'Valentin', 'Maughan', 'Tochukwu', 'Tomlinson',
       'Hickey', "Ch'ang", 'Chinomso', 'Wan', 'Mazzanti', 'Thompson',
       'Onyeoruru', 'Townsend', 'Hope', 'Ives', 'Bergamaschi', 'Bateman',
       'Jarvis', 'Long', 'Panicucci', 'Monnier', 'Andreev', 'Steele',
       'Sung', 'Baker', 'Richards', 'Herbert', "Ch'ien", 'Siciliano',
```

```
       'Allan', 'Hobbs', 'Coates', 'Ignatyev', 'McConnell', 'Tang',
       'Bell', 'Nwora', 'McDonald', 'Davidson', 'Hs?', 'Onyemauchechukwu',
       'Connolly', 'King', 'Greco', 'Fanucci', 'Su', 'Woolnough',
       'Harrison', 'Isayeva', 'Martin', "Ch'en", 'Nnonso', 'Trentino',
       'Kelly', 'Sullivan', 'Thomson', 'Rogers', 'Ponomarev', 'De Luca',
       'Sheppard', 'Birk', 'Obioma', 'Iheatu', 'Dellucci', 'Hs?eh',
       'Davies', 'Fields', 'Page', 'Wu', 'Chen', 'Young', 'Sabbatini',
       'Harvey', 'Sal', 'Tan', 'Otitodilinna', 'Chinonyelum', 'Munro',
       'Sargent', 'Bianchi', 'Chiu', 'Buccho', 'Dolgorukova', 'Green',
       'Ibezimako', 'Dennis', 'Eve', 'Stradford', 'Liao', 'Abramovich',
       'Cross', 'Ko', 'Fiorentini', 'Gray', 'Lu', 'Yevseyev', 'Horton',
       'Wall', 'Cummins', 'Kodilinyechukwu', 'Scott', 'Harris', 'Ofodile',
       'Highland', 'Botts', 'Baresi', 'Afamefuna', 'Rogova', 'Bufkin',
       'Boniwell', 'Lombardi', 'Kuykendall', 'Browne', 'Niu', 'Mackie',
       'Ifeatu', 'Fan', 'Davey', 'Kennedy', 'Gibson', 'Ferguson',
       'Onyenachiya', 'Combes', 'Kent', 'Ngozichukwuka', 'Okonkwo',
       'Lappin', 'Zhdanova', 'Ibekwe', 'Pettry', 'Wright', 'Felix',
       'Nnaife', 'Schofield', 'Onwubiko', 'Rubin', 'Perry', 'Ma',
       'Akudinobi', 'Flemming', 'Spencer', 'Okwuadigbo', 'Tu',
       'Chidiebere', 'Yuan', 'Jamieson', 'Davis', 'Conway',
       'Okwudiliolisa', 'Lavrentyev', 'Muir', 'Soares', 'Lucas', 'Hudson',
       'Winter-Irving', 'North', 'Rahman', 'Samaniego', 'Mayrhofer',
       'Esomchi', 'Findlay', 'Seleznyov', 'Mickey', 'Spaull', 'Brown',
       'Evseyev', 'Vorobyova', 'McClaran', 'Mueller', 'Hao', 'Peng',
       'Sykes', 'Woronoff', 'Duncan', 'Whitson', 'Loggia', 'Parkin',
       'Tsai', 'Pugh', 'Chin', 'Chiemezie', 'Curtis', 'Russell', 'Rose',
       'Fiorentino', 'Shaw', 'Zakharov', 'Tsui', 'Bolton', 'Golubev',
       'Vanzetti', 'Tretiakova', 'Owens', 'Chifley', 'Murphy',
```

```
       'Yermakova', 'Chikelu', 'Otitodilichukwu', 'Norriss', 'Mott',
       'Henry', 'Pope', 'Romano', 'Watson', 'Ward', 'Fitzgerald',
       'Arnold', 'Wells', 'Iweobiegbunam', 'Chiedozie', 'Thomas',
       'Chukwujekwu', 'Vagin', 'Parks'], dtype=object)
```

```
df.dtypes
```

```
RowNumber             int64
CustomerId            int64
Surname              object
CreditScore           int64
Age                   int64
Tenure                int64
Balance             float64
NumOfProducts         int64
HasCrCard             int64
IsActiveMember        int64
EstimatedSalary     float64
Exited                int64
France                uint8
Germany               uint8
Spain                 uint8
Female                uint8
Male                  uint8
dtype: object
```

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

8. Split the data into dependent and independent variables

```
dependent_data=df['Exited']
```

```
independent_data=df.drop('Exited',axis=1)
```

9. Scale the independent variables

```
#minmax_scalar
df_norm = (independent_data-independent_data.min())/(independent_data.max()-independent_data.min())
```

```
df_norm.head()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	France	Germany	Spain	Female	Male
58	0.000000	0.229998	0.233032	0.103448	0.4	0.000000	0.000000	1.0	0.0	0.005731	0.0	0.0	1.0	1.0	0.0
85	0.002733	0.958335	0.552036	0.413793	1.0	0.000000	0.333333	1.0	1.0	0.573761	0.0	0.0	1.0	1.0	0.0
104	0.004657	0.956989	0.592760	0.068966	0.1	0.000000	0.000000	1.0	1.0	0.890258	0.0	0.0	1.0	1.0	0.0
158	0.010124	0.093542	0.538462	0.344828	0.6	0.520066	0.000000	0.0	1.0	0.523728	1.0	0.0	0.0	1.0	0.0
181	0.012452	0.895729	0.230769	0.068966	0.2	0.000000	0.333333	1.0	1.0	0.239051	1.0	0.0	0.0	0.0	1.0

10. Split the data into training and testing

```
X_train, X_test, y_train, y_test = train_test_split( independent_data, dependent_data, test_size=0.30, random_state=42)
```

```python
sc=StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
```

```
[[-0.11823064  0.61271238  1.23747734 ... -0.61572793  1.07879243
  -1.07879243]
 [ 0.8471792   0.31996526  0.46317624 ... -0.61572793 -0.92696238
   0.92696238]
 [-0.22875737 -0.7671311  -0.57601735 ...  1.62409395  1.07879243
  -1.07879243]
 ...
 [ 0.88831374 -1.36057018  0.70769237 ... -0.61572793  1.07879243
  -1.07879243]
 [ 1.62408552  0.21836558  1.42086444 ... -0.61572793 -0.92696238
   0.92696238]
 [-0.65405277 -1.57586185  1.27823003 ... -0.61572793 -0.92696238
   0.92696238]]
```

```python
print(X_test)
```

```
[[ 0.43654915 -0.19958681  0.30016548 ... -0.61572793 -0.92696238
   0.92696238]
 [-1.39268608 -0.45567995  0.4020472  ... -0.61572793 -0.92696238
   0.92696238]
 [ 0.9845328  -0.86183628 -0.1888668  ... -0.61572793  1.07879243
  -1.07879243]
 ...
 [ 0.43905299 -0.72549476 -0.57601735 ... -0.61572793 -0.92696238
   0.92696238]
 [ 1.64805086  0.17543652  0.02508482 ... -0.61572793 -0.92696238
   0.92696238]
 [ 0.71411789 -0.50209666 -0.42319476 ...  1.62409395  1.07879243
  -1.07879243]]
```