Assignment –2
Data Visualization and Pre-processing

Assignment Date	25 September 2022
Student Name	Mr.Veerenthiran.S
Student Roll Number	820419104080
Maximum Marks	2 Marks

1. Downloaded the Dataset Churn_Modelling.csv and Uploaded into content folder:

Importing Required Libraries:

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

2.Loading the dataset:

```
[ ] ds=pd.read_csv(r'/content/Churn_Modelling.csv')

[ ] ds.shape
    (10000, 14)

[ ] ds.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

3.Performing Visualization on Datasets

3.1 Univariate Analysis

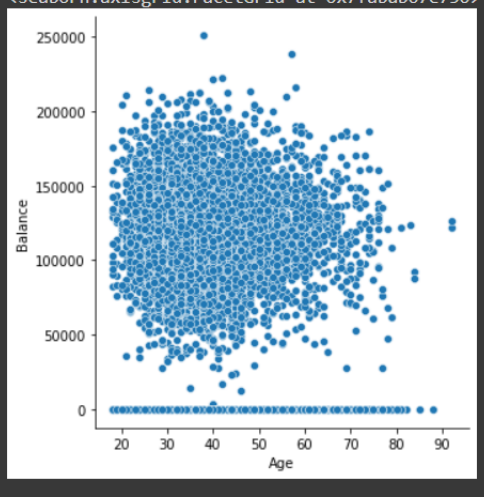
```
sns.distplot(ds['EstimatedSalary'],hist=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fabab6ed690>
```

3.2 Bi-variate Analysis

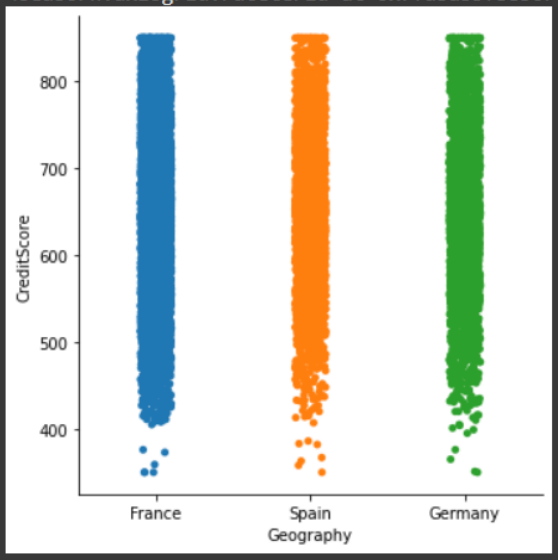
```
[ ]  sns.relplot(x='Age',y='Balance',data=ds)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fabab67e750>
```



```
[ ]  #categorical data
     sns.catplot(x='Geography',y='CreditScore',data=ds)
```

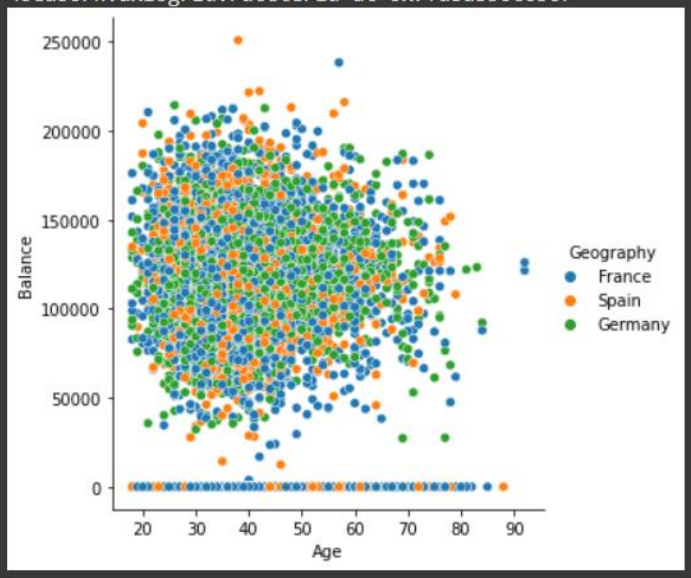
```
<seaborn.axisgrid.FacetGrid at 0x7fabab5fb850>
```



3.3 Multivariate Analysis

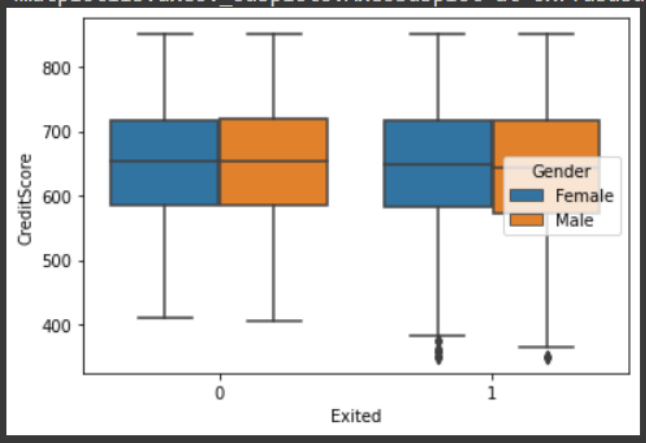
```
[ ]  sns.relplot(x='Age',y='Balance',hue='Geography',data=ds)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fabab56c650>
```



```
[ ]  #categorical data
     sns.boxplot(x='Exited',y='CreditScore',hue='Gender',data=ds)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fababd473d0>
```



4.Performing Descriptive Statistics on the Dataset

```
[ ]  ds.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

5.Handling the Missing values

```
[ ]  ds.isnull().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
```

```
[ ]  ds.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
```

```
[ ]  #no null values found , so no need to handle.
```

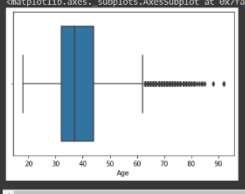
6.Finding the outliers and Replace the outliers:

```
[ ] ds.skew()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions
  """Entry point for launching an IPython kernel.
RowNumber          0.000000
CustomerId         0.001149
CreditScore       -0.071607
Age                1.011320
Tenure             0.010991
Balance           -0.141109
NumOfProducts      0.745568
HasCrCard         -0.901812
IsActiveMember    -0.060437
EstimatedSalary    0.002085
Exited             1.471611
dtype: float64
```

```
[ ] sns.boxplot(ds["Age"])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fabab455610>
```




```
[ ] q0 = ds["Age"].describe()["25%"]

[ ] q1 = ds["Age"].describe()["75%"]

[ ] iqr=q1-q0

[ ] lb = q0 -(1.5*iqr)
    ub = q1 + (1.5*iqr)

[ ] ds[ds["Age"]<lb]
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited

```
[ ] ds[ds["Age"]>ub]
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9753	9754	15705174	Chiedozie	656	Germany	Male	68	7	153545.11	1	1	1	186574.68	0
9765	9766	15777067	Thomas	445	France	Male	64	2	136770.67	1	0	1	43678.06	0
9832	9833	15814690	Chukwujekwu	595	Germany	Female	64	2	105736.32	1	1	1	89935.73	1
9894	9895	15704795	Vagin	521	France	Female	77	6	0.00	2	1	1	49054.10	0
9936	9937	15653037	Parks	609	France	Male	77	1	0.00	1	0	1	18708.76	0

359 rows × 14 columns

```
[ ] #Replacing the outlier
    outlier_list = list(ds[ds["Age"] > ub]["Age"])

[ ] print(outlier_list)
```

```
[66, 75, 65, 73, 65, 72, 67, 67, 79, 80, 68, 75, 66, 66, 70, 63, 72, 64, 64, 70, 67, 82, 63, 69, 65, 69, 64, 65, 74, 67, 66, 67, 63, 70, 71, 72, 67, 74, 76, 66, 63, 66, 68, 67, 63, 71,
```

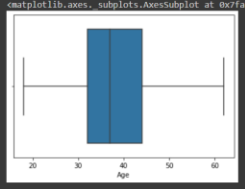
```
[ ] outlier_dict = {}.fromkeys(outlier_list,ub)

[ ] print(outlier_dict)
```

```
{66: 62.0, 75: 62.0, 65: 62.0, 73: 62.0, 72: 62.0, 67: 62.0, 79: 62.0, 80: 62.0, 68: 62.0, 70: 62.0, 63: 62.0, 64: 62.0, 82: 62.0, 69: 62.0, 74: 62.0, 71: 62.0, 76: 62.0, 77: 62.0, 88:
```

```
[ ] ds["Age"] = ds["Age"].replace(outlier_dict)
```

```
sns.boxplot(ds["Age"])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fabab310650>
```




```
[ ] ds[ds["Age"]>ub]
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited

7.Check for categorical columns and perform coding:

```
[ ] from sklearn.compose import ColumnTransformer
    from sklearn.preprocessing import OneHotEncoder
    ct=ColumnTransformer([('oh',OneHotEncoder(),[1,2])],remainder='passthrough')
    x=ct.fit_transform(x)
    print(x.shape)
```

```
(10000, 13)
```

```
[ ] # saving the data
    import joblib
    joblib.dump(ct,"churnct.pkl")
```

```
['churnct.pkl']
```

8.Split the data into dependent and independent variables

```
[ ] x=ds.iloc[:,3:13].values
    print(x.shape)
    y=ds.iloc[:,13:14].values
    print(y.shape)
```

```
(10000, 10)
(10000, 1)
```

9.Scale the independent variables:

```
[ ] 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	10	11	12
0	-1.014607	-0.569844	1.743090	1.091687	-1.091687	0.169582	-0.464608	0.006661	-1.215717	0.809503	0.642595	-1.032270	1.106432
1	-1.014607	1.754865	-0.573694	-0.916013	0.916013	-2.304559	0.301026	-1.377440	-0.006312	-0.921591	0.642595	0.968738	-0.748664
2	0.985604	-0.569844	-0.573694	1.091687	-1.091687	-1.191196	-0.943129	-1.031415	0.579935	-0.921591	0.642595	-1.032270	1.485335
3	-1.014607	-0.569844	1.743090	-0.916013	0.916013	0.035566	0.109617	0.006661	0.473128	-0.921591	0.642595	-1.032270	1.276528
4	-1.014607	-0.569844	1.743090	1.091687	-1.091687	2.056114	1.736588	1.044737	0.810193	0.809503	0.642595	0.968738	0.558378

10.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=0.2,random_state=0)
    print(x_train.shape)
    print(x_test.shape)
```

```
(8000, 13)
(2000, 13)
```