

Assignment -2

Python Programming

Assignment Date	3 November 2022
Student Name	P Mohamed Ibrahim
Student Roll Number	212219060163
Maximum Marks	2 Marks

```
In [4]: import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
In [5]: data=pd.read_csv("Churn_Modelling_ass2.csv")
```

```
In [6]: data.head(10)
```

```
Out[6]:
```

	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
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	1
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	1	1	10062.80	0
7	8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.88	1
8	9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.50	0
9	10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73	0

```
In [7]: data.tail(10)
```

```
Out[7]:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9990	9991	15798964	Nkemakonam	714	Germany	Male	33	3	35016.60	1	1	0	53667.08	0
9991	9992	15769959	Ajuluchukwu	597	France	Female	53	4	88381.21	1	1	0	69384.71	1
9992	9993	15657105	Chukwualuka	726	Spain	Male	36	2	0.00	1	1	0	195192.40	0
9993	9994	15569266	Rahman	644	France	Male	28	7	155060.41	1	1	0	29179.52	0
9994	9995	15719294	Wood	800	France	Female	29	2	0.00	2	0	0	167773.55	0
9995	9996	15606229	Obijaku	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

```
In [8]: #describe statistics
data.describe()
```

```
Out[8]:
```

	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

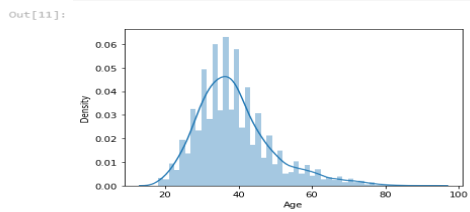
```
In [9]: data.kurt(axis=0,skipna=True)
```

```
Out[9]: RowNumber      -1.200000
CustomerId    -1.196113
CreditScore   -0.425726
Age           1.395347
Tenure        -1.165225
Balance       -1.489412
NumOfProducts  0.582981
HasCrCard     -1.186973
IsActiveMember -1.996747
EstimatedSalary -1.181518
Exited        0.165671
dtype: float64
```

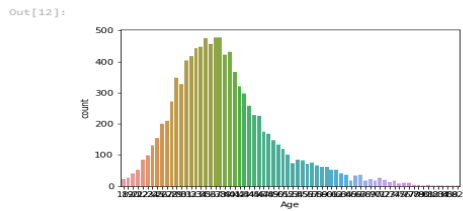
```
In [10]: data.kurt(axis=1,skipna=True)
```

```
Out[10]: 0      10.998778
1      10.997999
2      10.995886
3      10.998962
4      10.997675
...
9995    10.998908
9996    10.998551
9997    10.999788
9998    10.998530
9999    10.997973
Length: 10000, dtype: float64
```

```
In [11]: sns.distplot(data['Age'])
```



```
In [12]: sns.countplot(data["Age"])
```



```
In [13]: data.skew(axis=0,skipna=True)
```

Out[13]:

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

```
In [14]: data.skew(axis=1,skipna=True)
```

Out[14]:

0	3.316373
1	3.316193
2	3.315777
3	3.316411
4	3.316145
...	
9995	3.316399
9996	3.316325
9997	3.316581
9998	3.316321
9999	3.316207
Length:	10000, dtype: float64

```
In [15]: data.isnull().any()
```

Out[15]:

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

```
In [16]: data.isnull().sum()
```

Out[16]:

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

```
In [17]: data.duplicated()
```

Out[17]:

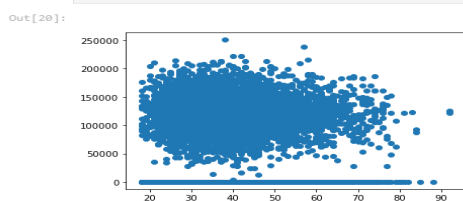
0	False
1	False
2	False
3	False
4	False
...	
9995	False
9996	False
9997	False
9998	False
9999	False
Length:	10000, dtype: bool

```
In [18]: data.duplicated().sum()
```

Out[18]: 0

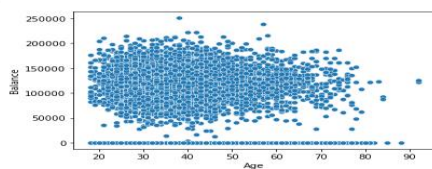
```
In [19]: ###VISUALISATION
```

```
In [20]: plt.scatter(data.Age,data.Balance)
```



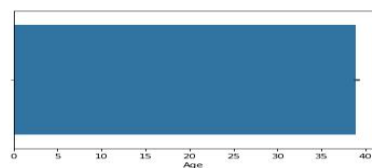
```
In [21]: sns.scatterplot(data.Age,data.Balance)
```

```
Out[21]:
```



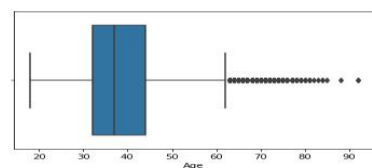
```
In [22]: sns.barplot(data['Age'])
```

```
Out[22]:
```



```
In [23]: sns.boxplot(data['Age'])
```

```
Out[23]:
```



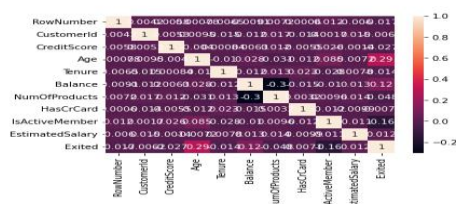
```
In [24]: data.corr()
```

```
Out[24]:
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495	-0.009067	0.007246	0.000599	0.012044	-0.005988	-0.016571
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883	-0.012419	0.016972	-0.014025	0.001665	0.015271	-0.006248
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
Balance	-0.009067	-0.012419	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
NumOfProducts	0.007246	0.016972	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
HasCrCard	0.000599	-0.014025	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
IsActiveMember	0.012044	0.001665	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
EstimatedSalary	-0.005988	0.015271	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
Exited	-0.016571	-0.006248	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

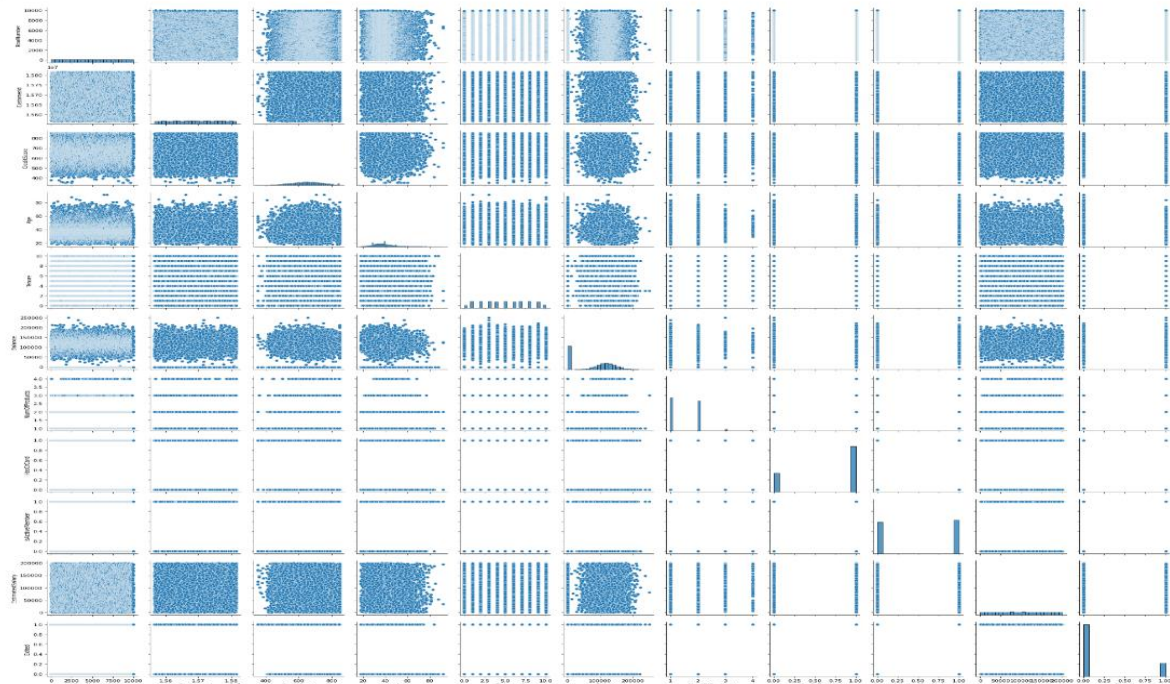
```
In [25]: sns.heatmap(data.corr(),annot=True)
```

```
Out[25]:
```



```
In [36]: sns.pairplot(data)
```

```
Out[36]:
```



```
In [39]: from scipy.stats import spearmanr
```

```
In [40]:
```

```
corr=spearmanr(data)
corr
```

```
Out[40]:
```

```
SpearmanResult(correlation=array([[ 1.00000000e+00,  4.18684789e-03,  1.82537815e-03,
 5.13017187e-03, -1.01176571e-02,  1.81953613e-02,
 4.76064421e-04, -6.9343206e-03, -9.01325568e-03,
 8.30510741e-03,  5.98746525e-04,  1.20443901e-02,
-6.00682958e-03, -1.65713715e-02],
 [ 4.18684789e-03,  1.00000000e+00,  5.31564210e-03,
 5.96746465e-03,  6.03529435e-03, -2.62440728e-03,
 8.77466555e-03,  1.50720283e-02, -1.39321914e-02,
 1.92970188e-02, -1.40232399e-02,  1.68193033e-03,
 1.52457829e-02, -6.26374782e-03],
 [ 1.82537815e-03,  5.31564210e-03,  1.00000000e+00,
 6.68503170e-03, -2.26792517e-02, -2.14337922e-03,
 1.37678535e-03, -1.70916721e-02, -8.00358124e-04,
-1.72831393e-02, -8.93818901e-03,  1.37684719e-03,
 1.17949476e-02, -1.09832944e-02],
 [ 5.13017187e-03,  5.96746465e-03,  6.68503170e-03,
 1.00000000e+00,  6.10527978e-03, -3.01144279e-03,
-7.97404431e-03,  1.13317419e-03,  5.68657057e-03,
 1.25677271e-02, -3.80181966e-03,  2.42623407e-02,
 1.23652438e-03, -2.32893966e-02],
 [-1.01176571e-02,  6.03529435e-03, -2.26792517e-02,
 6.10527978e-03,  1.00000000e+00,  2.05197803e-03,
 3.53513965e-02,  3.76366156e-03,  9.94871724e-02,
 3.69108918e-04, -7.22407343e-03,  4.44007080e-03,
-1.94818567e-04,  5.30920641e-02],
 [ 1.81953613e-02, -2.62440728e-03, -2.14337922e-03,
-3.01144279e-03,  2.05197803e-03,  1.00000000e+00,
-2.97848194e-02,  1.50959348e-02,  1.35043861e-02,
-1.28505367e-02,  5.76612437e-03,  2.25443247e-02,
-2.26853704e-03, -1.06512488e-01],
 [ 4.76064421e-04,  8.77466555e-03,  1.37678535e-03,
-7.97404431e-03,  3.53513965e-02, -2.97848194e-02,
 1.00000000e+00, -1.04049493e-02,  3.33043436e-02,
-5.85664619e-02, -1.52782371e-02,  3.98391734e-02,
-2.43149876e-03,  3.23967912e-01],
 [-6.9343206e-03, -1.50720283e-02, -1.70916721e-02,
 1.13317419e-03,  3.76366156e-03,  1.50959348e-02,
-1.04049493e-02,  1.00000000e+00, -9.51289512e-03,
 1.29080538e-02,  2.23540939e-02, -2.86732861e-02,
 7.77808376e-03, -1.39780555e-02],
 [-9.01325568e-03, -1.39321914e-02, -8.00358124e-04,
 5.68657057e-03,  9.94871724e-02,  1.35043861e-02,
 3.33043436e-02, -9.51289512e-03,  1.00000000e+00,
-3.16626558e-01, -9.83460270e-03, -1.14965258e-02,
 1.17780035e-02,  1.1110193e-01],
 [ 8.30510741e-03,  1.92970188e-02, -1.72831393e-02,
 1.25677271e-02,  7.69108918e-04, -1.28505367e-02,
-5.85664619e-02,  1.29080538e-02, -3.16626558e-01,
 1.00000000e+00,  3.85886031e-03,  1.62917706e-02,
 1.25698129e-02, -1.25282063e-01],
 [ 5.98746525e-04, -1.40232399e-02, -8.93818901e-03,
-3.80181966e-03, -7.22407343e-03,  5.76612437e-03,
-1.52782371e-02,  2.23540939e-02, -9.83460270e-03,
 3.85886031e-03,  1.00000000e+00, -1.18656369e-02,
-1.00409074e-02,  7.13776560e-03],
 [ 1.20443901e-02,  1.68193033e-03,  1.37684719e-03,
 2.42623407e-02,  4.44007080e-03,  2.25443247e-02,
 3.98391734e-02, -2.86732861e-02, -1.14965258e-02,
 1.62917706e-02, -1.18656369e-02,  1.00000000e+00,
-1.14690521e-02, -1.56128278e-01],
 [-6.00682958e-03,  1.52457829e-02,  1.17949476e-02,
 1.23652438e-03, -1.94818567e-04, -8.26853704e-03,
-2.43149876e-03,  7.77808376e-03,  1.17780035e-02,
 1.25698129e-02, -1.00409074e-02, -1.14690521e-02,
 1.00000000e+00,  1.20805366e-02],
 [-1.65713715e-02, -6.26374782e-03, -1.09832944e-02,
-2.32893966e-02,  5.30920641e-02, -1.06512488e-01,
 3.23967912e-01, -1.39780555e-02,  1.11110193e-01,
-1.25282063e-01, -7.13776560e-03, -1.56128278e-01,
 1.20805366e-02,  1.00000000e+00]], pvalue=array([[ 0.00000000e+000,  6.75483429e-001,  8.55178468e-001,
 6.07981798e-001,  3.11698199e-001,  6.88261457e-002,
 9.62034639e-001,  4.80868885e-001,  3.67465405e-001,
 4.06300660e-001,  9.52261425e-001,  2.28461236e-001,
 5.48097586e-001,  9.75106276e-002],
 [ 6.75483429e-001,  0.00000000e+000,  5.95071292e-001,
 5.50722932e-001,  5.46203060e-001,  7.93006618e-001,
 3.80283664e-001,  1.31785022e-001,  1.63585747e-001,
 5.36514208e-002,  1.60847582e-001,  8.66447868e-001,
 1.27389774e-001,  5.31116466e-001],
 [ 8.55178468e-001,  5.95071292e-001,  0.00000000e+000,
 5.03861020e-001,  2.3332702e-002,  8.30304249e-001,
 8.90508036e-001,  8.74364323e-002,  9.36216720e-001,
 8.39475437e-002,  3.71469037e-001,  8.90503148e-001,
 2.38243578e-001,  2.72106138e-001],
 [ 6.07981798e-001,  5.50722932e-001,  5.03861020e-001,
 0.00000000e+000,  5.41558890e-001,  6.3332662e-001,
 4.25266703e-001,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e-001,  5.69634028e-001,
 2.08874884e-001,  7.03844602e-001,  1.52541637e-002,
 9.01602674e-001,  1.98609526e-002],
 [ 3.11698199e-001,  5.46203060e-001,  2.3332702e-002,
 5.41558890e-001,  0.00000000e+000,  8.37437459e-001,
 4.06537579e-004,  7.06678685e-001,  2.01668047e-023,
 9.38702072e-001,  4.70093788e-001,  6.57076013e-001,
 9.84458653e-001,  1.08256524e-007],
 [ 6.88261457e-002,  7.93006618e-001,  8.30304249e-001,
 7.6332662e-001,  8.37437459e-001,  0.00000000e+000,
 2.89407525e-003,  9.09790189e
```

```
In [42]: x=data[["EstimatedSalary"]]
y=data[["CreditScore"]]
```

```
In [43]: model=sm.OLS(y,x)
result=model.fit()
result.summary()
```

Out[43]: OLS Regression Results

Dep. Variable:	CreditScore	R-squared (uncentered):	0.735			
Model:	OLS	Adj. R-squared (uncentered):	0.735			
Method:	Least Squares	F-statistic:	2.779e+04			
Date:	Sat, 24 Sep 2022	Prob (F-statistic):	0.00			
Time:	15:56:14	Log-Likelihood:	-72429.			
No. Observations:	10000	AIC:	1.449e+05			
Df Residuals:	9999	BIC:	1.449e+05			
Df Model:	1					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
EstimatedSalary	0.0049	2.93e-05	166.705	0.000	0.005	0.005
Omnibus:	1758.359	Durbin-Watson:	1.554			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	376.161			
Skew:	0.004	Prob(JB):	2.08e-82			
Kurtosis:	2.050	Cond. No.	1.00			

Notes:

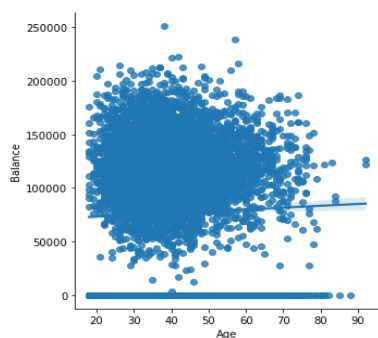
- [1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
- [2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [44]: from sklearn.preprocessing import scale
x=scale(x)
x
```

```
Out[44]: array([[ 0.02188649],
[ 0.21653375],
[ 0.2406869 ],
...,
[-1.00864308],
[-0.12523071],
[-1.07636976]])
```

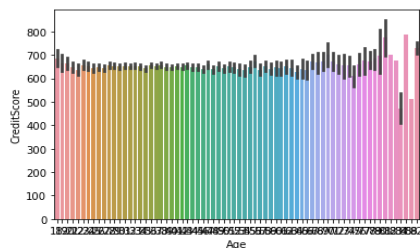
```
In [45]: sns.lmplot(x='Age',y='Balance',data=data)
```

Out[45]:



```
In [46]: sns.barplot(x="Age",y="CreditScore",data=data)
```

Out[46]:



```
In [32]: ###outlier detection
```

```
In [47]: qnt = data.quantile(q=[0.75,0.25])
qnt
```

```
Out[47]:
```

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

```
In [48]: iqr=qnt.loc[0.75]-qnt.loc[0.25]
iqr
```

```
Out[48]: RowNumber      4999.5000
CustomerId    124705.5000
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
```

```
In [51]: upper= qnt.loc[0.75]+1.5*iqr
upper
```

```
Out[51]: RowNumber      1.499950e+04
CustomerId    1.594029e+07
CreditScore    9.190000e+02
Age           6.200000e+01
Tenure        1.300000e+01
Balance      3.191106e+05
NumOfProducts    3.500000e+00
HasCrCard       2.500000e+00
IsActiveMember  2.500000e+00
EstimatedSalary 2.969675e+05
Exited         0.000000e+00
dtype: float64
```

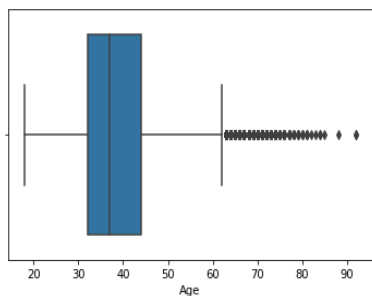
```
In [52]: lower= qnt.loc[0.25]-1.5*iqr
lower
```

```
Out[52]: RowNumber      -4.998500e+03
CustomerId    1.544147e+07
CreditScore    3.830000e+02
Age           1.400000e+01
Tenure       -3.000000e+00
Balance     -1.914664e+05
NumOfProducts   -5.000000e-01
HasCrCard      -1.500000e+00
IsActiveMember -1.500000e+00
EstimatedSalary -9.657710e+04
Exited         0.000000e+00
dtype: float64
```

```
In [36]: ###replacing outlier
```

```
In [37]: sns.boxplot(data["Age"])
```

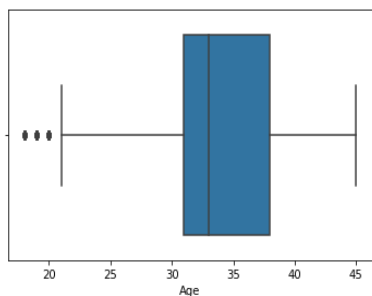
Out[37]:



```
In [53]: data["Age"]= np.where(data["Age"]>45,31,data["Age"])
```

```
In [54]: sns.boxplot(data["Age"])
```

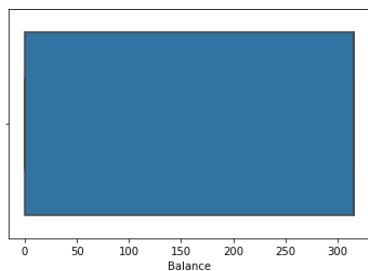
Out[54]:



```
In [55]: data["Balance"] = np.where(data["Balance"] > 618, 316, data["Balance"])
```

```
In [56]: sns.boxplot(data["Balance"])
```

Out[56]:



```
In [57]: data.head()
```

```
Out[57]:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.0	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	316.0	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	316.0	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.0	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	316.0	1	1	1	79084.10	0

```
In [58]: data["Gender"].replace({"Female":0, "Male":1}, inplace = True)
```

```
In [59]: data.head(10)
```

```
Out[59]:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	0	42	2	0.0	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	0	41	1	316.0	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	0	42	8	316.0	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	0	39	1	0.0	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	0	43	2	316.0	1	1	1	79084.10	0
5	6	15574012	Chu	645	Spain	1	44	8	316.0	2	1	0	149756.71	1
6	7	15592531	Bartlett	822	France	1	31	7	0.0	2	1	1	10062.80	0
7	8	15656148	Obinna	376	Germany	0	29	4	316.0	4	1	0	119346.88	1
8	9	15792365	He	501	France	1	44	4	316.0	2	0	1	74940.50	0
9	10	15592389	H?	684	France	1	27	2	316.0	1	1	1	71725.73	0

```
In [60]: data["HasCrCard"].replace({1:"yes",0:"no"}, inplace = True)
```

```
In [61]: data.head(10)
```

```
Out[61]:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	0	42	2	0.0	1	yes	1	101348.88	1
1	2	15647311	Hill	608	Spain	0	41	1	316.0	1	no	1	112542.58	0
2	3	15619304	Onio	502	France	0	42	8	316.0	3	yes	0	113931.57	1
3	4	15701354	Boni	699	France	0	39	1	0.0	2	no	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	0	43	2	316.0	1	yes	1	79084.10	0
5	6	15574012	Chu	645	Spain	1	44	8	316.0	2	yes	0	149756.71	1
6	7	15592531	Bartlett	822	France	1	31	7	0.0	2	yes	1	10062.80	0
7	8	15656148	Obinna	376	Germany	0	29	4	316.0	4	yes	0	119346.88	1
8	9	15792365	He	501	France	1	44	4	316.0	2	no	1	74940.50	0
9	10	15592389	H?	684	France	1	27	2	316.0	1	yes	1	71725.73	0

```
In [62]: #Label encoding
```

```
In [74]: from sklearn.preprocessing import LabelEncoder  
le=LabelEncoder()
```

```
In [75]: data["Age"] = le.fit_transform(data["Age"])
```

```
In [76]: data.Age.unique()
```

```
Out[76]: array([24, 23, 21, 25, 26, 13, 11,  9,  6, 16,  7, 17, 27, 14, 20, 18, 15,  
                22, 19,  1,  8,  3,  4, 12, 10,  2,  5,  0], dtype=int64)
```



```
In [77]: x=data.iloc[:,0:13].values
x
```

```
Out[77]: array([[1, 15634602, 'Hargrave', ..., 'yes', 1, 101348.88],
 [2, 15647311, 'Hill', ..., 'no', 1, 112542.58],
 [3, 15619304, 'Onio', ..., 'yes', 0, 113931.57],
 ...,
 [9998, 15584532, 'Liu', ..., 'no', 1, 42085.58],
 [9999, 15682355, 'Sabbatini', ..., 'yes', 0, 92888.52],
 [10000, 15628319, 'Walker', ..., 'yes', 0, 38190.78]], dtype=object)
```

```
In [78]: y=data.iloc[:,13:14].values
y
```

```
Out[78]: array([[1],
 [0],
 [1],
 ...,
 [1],
 [1],
 [0]], dtype=int64)
```

```
In [79]: data.head()
```

```
Out[79]:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	0	24	2	0.0	1	yes	1	101348.88	1
1	2	15647311	Hill	608	Spain	0	23	1	316.0	1	no	1	112542.58	0
2	3	15619304	Onio	502	France	0	24	8	316.0	3	yes	0	113931.57	1
3	4	15701354	Boni	699	France	0	21	1	0.0	2	no	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	0	25	2	316.0	1	yes	1	79084.10	0

```
In [80]: from sklearn.preprocessing import OneHotEncoder
```

```
In [81]: ohe= OneHotEncoder()
```

```
In [82]: z=ohe.fit_transform(x[:,0:14]).toarray()
z
```

```
Out[82]: array([[1., 0., 0., ..., 0., 0., 0.],
 [0., 1., 0., ..., 0., 0., 0.],
 [0., 0., 1., ..., 0., 0., 0.],
 ...,
 [0., 0., 0., ..., 0., 0., 0.],
 [0., 0., 0., ..., 0., 0., 0.],
 [0., 0., 0., ..., 0., 0., 0.]])
```

```
In [83]: ###split the data into training and testing
```

```
In [84]: from sklearn.model_selection import train_test_split
```

```
In [85]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

```
In [86]: x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
Out[86]: ((8000, 13), (2000, 13), (8000, 1), (2000, 1))
```



```

In [87]: x_train

Out[87]: array([[7390, 15676909, 'Mishin', ..., 'yes', 0, 163830.64],
               [9216, 15749265, 'Carslaw', ..., 'yes', 1, 57098.0],
               [2996, 15582492, 'Moore', ..., 'yes', 0, 185630.76],
               ...,
               [3265, 15574372, 'Hoolan', ..., 'yes', 0, 181429.87],
               [9846, 15664035, 'Parsons', ..., 'yes', 1, 148750.16],
               [2733, 15592816, 'Udokamma', ..., 'yes', 0, 118855.26]],
              dtype=object)

In [88]: x_test

Out[88]: array([[9395, 15615753, 'Upchurch', ..., 'yes', 1, 192852.67],
               [899, 15654700, 'Fallaci', ..., 'yes', 0, 128702.1],
               [2399, 15633877, 'Morrison', ..., 'yes', 1, 75732.25],
               ...,
               [9550, 15772604, 'Chiemezie', ..., 'yes', 0, 141533.19],
               [2741, 15787699, 'Burke', ..., 'yes', 1, 11276.48],
               [6691, 15579223, 'Niu', ..., 'yes', 0, 192950.6]], dtype=object)

In [89]: y_train

Out[89]: array([[0],
               [0],
               [0],
               ...,
               [0],
               [0],
               [1]], dtype=int64)

In [90]: y_test

Out[90]: array([[0],
               [1],
               [0],
               ...,
               [0],
               [0],
               [0]], dtype=int64)

In [91]: from sklearn.preprocessing import scale

In [92]: x=data["CreditScore"]
          S=scale(x)
          S

Out[92]: array([-0.32622142, -0.44003595, -1.53679418, ...,  0.60498839,
                1.25683526,  1.46377078])

In [93]: ###INDEPENDENT VARIABLE

In [94]: y=data["Age"]
          y

Out[94]: 0      24
          1      23
          2      24
          3      21
          4      25
          ..
          9995   21
          9996   17
          9997   18
          9998   24
          9999   10
          Name: Age, Length: 10000, dtype: int64

In [ ]:

In [95]: x=data.drop(data["Age"],axis=0)
          x

Out[95]:
   RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age  Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  Exited
28         29      15728693    McWilliams        574     Germany    0    25      3    316.0             1         yes             1      100187.43      0
29         30      15656300     Lucciano        411     France     1    11     0    316.0             2         yes             1      53483.21      0
30         31      15589475      Azikiwe        591     Spain     0    21     3     0.0             3         yes             0      140469.38      1
31         32      15706552  Odinakachukwu        533     France    1    18     7    316.0             1         no              1      156731.91      0
32         33      15750181     Sanderson        553     Germany    1    23     9    316.0             2         no              0       81898.81      0
...         ...         ...         ...         ...         ...     ...     ...     ...         ...         ...         ...         ...         ...
9995        9996      15606229     Obijaku        771     France     1    21     5     0.0             2         yes             0       96270.64      0
9996        9997      15569892   Johnstone        516     France     1    17    10    316.0             1         yes             1      101699.77      0
9997        9998      15584532       Liu        709     France     0    18     7     0.0             1         no              1       42085.58      1
9998        9999      15682355   Sabbatini        772     Germany    1    24     3    316.0             2         yes             0       92888.52      1
9999       10000      15628319     Walker        792     France     0    10     4    316.0             1         yes             0       38190.78      0

9972 rows x 14 columns

In [96]: ###splitting dependent variable

In [97]: y=data.iloc[:, -1].values
          y

Out[97]: array([1, 0, 1, ..., 1, 1, 0], dtype=int64)

In [98]: data=pd.DataFrame({"Age": [1,2,np.nan], "CreditScore": [1,np.nan,np.nan], "Balance": [1,2,3]})
          data

Out[98]:
   Age  CreditScore  Balance
0   1.0           1.0        1
1   2.0          NaN        2
2  NaN           NaN        3

In [99]: data.isnull().any()

Out[99]: Age           True
CreditScore       True
Balance           False
dtype: bool

```

```
In [100]: data.isnull().sum()
```

```
Out[100]: Age      1  
CreditScore  2  
Balance      0  
dtype: int64
```

```
In [101]: data.dropna()
```

```
Out[101]:
```

	Age	CreditScore	Balance
0	1.0	1.0	1

```
In [102]: data.dropna(axis=1)
```

```
Out[102]:
```

	Balance
0	1
1	2
2	3

```
In [103]: data["Age"].mean()
```

```
Out[103]: 1.5
```

```
In [ ]:
```