Importing Packages

In [126…
```
import pandas as pd
import numpy as np
import seaborn as sb
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Loading the Dataset

In [127…
```
data=pd.read_csv("Churn_Modelling.csv")
```

In [128…
```
data.head(10)
```

Out[128]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSala
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	1	1	10062.
7	8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.
8	9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.
9	10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.

In [129…
```
data.tail(5)
```

Out[129]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimatec
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38

In [130…
```
data.describe()
```

Out[130]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	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
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000

Visualizations

Univariate Analysis

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

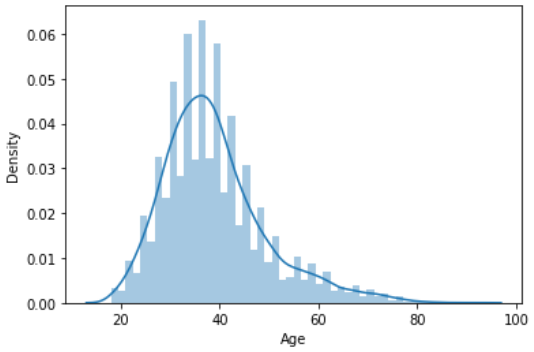
Out[9]:
```
RowNumber         -1.200000
CustomerId        -1.196113
CreditScore       -0.425726
Age                1.395347
Tenure            -1.165225
Balance           -1.489412
NumOfProducts      0.582981
HasCrCard         -1.186973
IsActiveMember    -1.996747
EstimatedSalary   -1.181518
Exited  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

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

Out[10]:
```
0       10.998778
1       10.997909
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
```

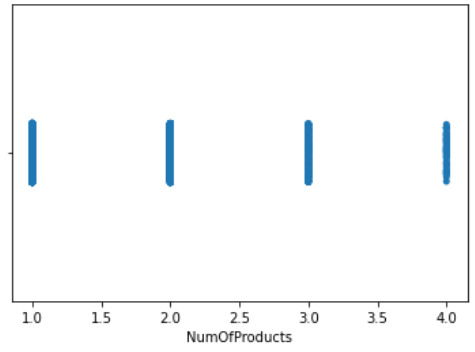
In [15]:
```
sb.distplot(data['Age'])
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffba11c6310>
```



In [19]:
```
sb.stripplot(data['NumOfProducts'])
```

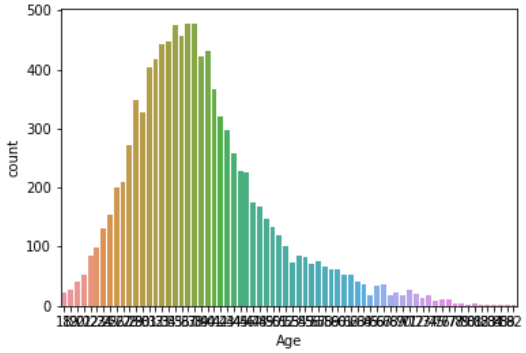
Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb9de4ac50>
```



Bivariate Analysis

In [20]:
```
sb.countplot(data["Age"])
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffba14b66d0>
```



In [22]:
```
data.skew(axis=0, skipna=True)
```

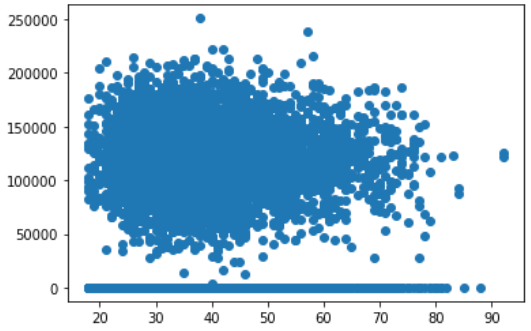
Out[22]:
```
RowNumber          0.000000
CustomerId         0.001149
CreditScore       -0.071607
Age                1.011320
Tenure             0.010991
Balance           -0.141109
NumOfProducts      0.745568
HasCrCard         -0.901812
IsActiveMember    -0.060437
EstimatedSalary    0.002085
Exited             1.471611
dtype: float64
```

In [23]:
```
data.skew(axis=1, skipna=True)
```

Out[23]:
```
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
```

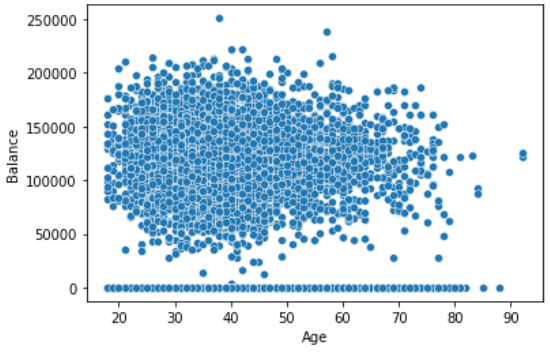
In [28]:
```
plt.scatter(data.Age, data.Balance)
```

Out[28]:
```
<matplotlib.collections.PathCollection at 0x7ffb9d2bfed0>
```



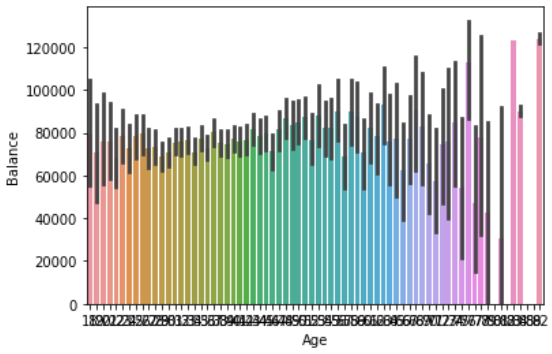
In [29]:
```
sb.scatterplot(data.Age, data.Balance)
```

Out[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb9d5ef850>
```



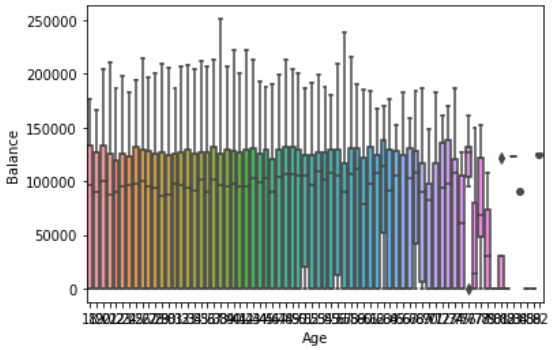
In [32]:
```
sb.barplot(data.Age, data.Balance)
```

Out[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb9ce641d0>
```



In [33]:
```
sb.boxplot(data.Age, data.Balance)
```

Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb9cb24710>
```



In [34]:
```
data.corr()
```

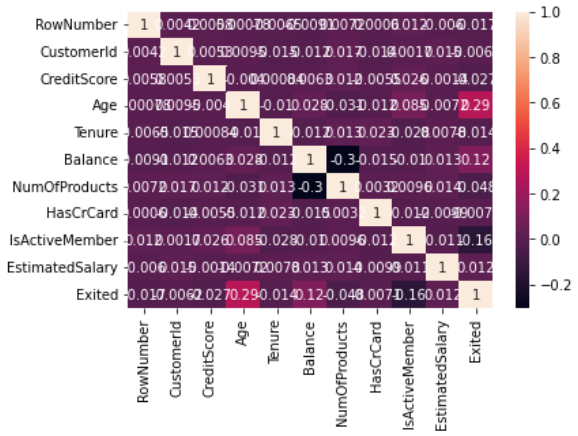
Out[34]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exit
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495	-0.009067	0.007246	0.000599	0.012044	-0.005988	-0.0165
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883	-0.012419	0.016972	-0.014025	0.001665	0.015271	-0.0062
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.0270
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.2853
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.0140
Balance	-0.009067	-0.012419	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.1185
NumOfProducts	0.007246	0.016972	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.0478
HasCrCard	0.000599	-0.014025	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.0071
IsActiveMember	0.012044	0.001665	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.1561
EstimatedSalary	-0.005988	0.015271	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.0120
Exited	-0.016571	-0.006248	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.0000

Multivariate Analysis

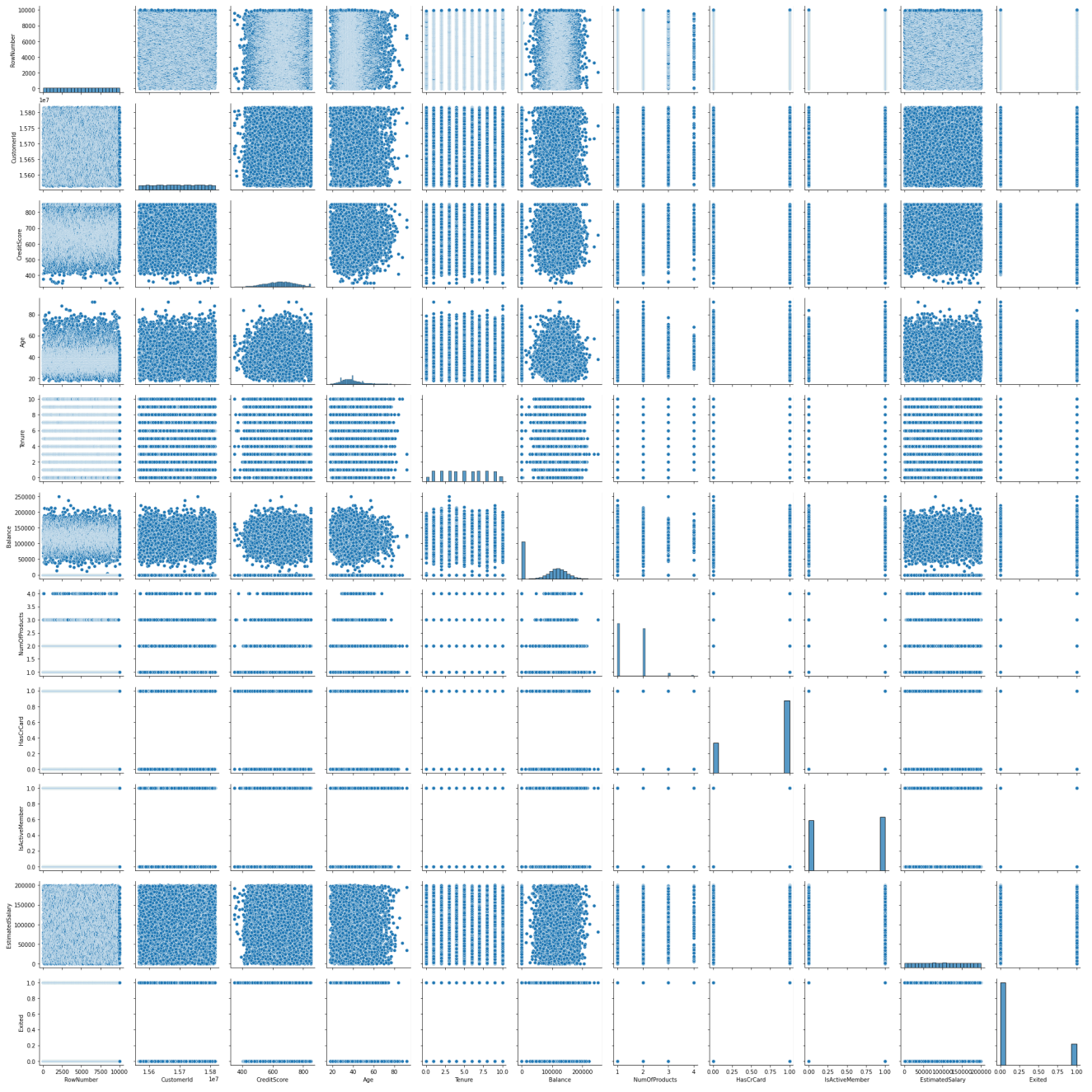
In [35]:
```
sb.heatmap(data.corr(), annot=True)
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb9d11f190>
```



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

Out[36]:
```
<seaborn.axisgrid.PairGrid at 0x7ffb9c22f150>
```



Descriptive statistics

In [37]:
```
from scipy.stats import spearmanr
```

In [39]:
```
corr = spearmanr(data)
corr
```

Out[39]:
```
SpearmanrResult(correlation=array([[ 1.00000000e+00,  4.18684789e-03,  1.82537815e-03,
         5.13017187e-03, -1.01176571e-02,  1.81963613e-02,
         4.76064421e-04, -6.93433206e-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, -1.40233299e-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,
         7.69108918e-04, -7.22407343e-03,  4.44007080e-03,
        -1.94818567e-04,  5.30920641e-02],
       [ 1.81963613e-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,
        -8.26853704e-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],
       [-6.93433206e-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,  1.11110193e-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, -1.40233299e-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, 4.88086885e-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, 2.33332702e-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, 7.63332662e-001,
        4.25266703e-001, 9.09790109e-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, 2.33332702e-002,
 5.41558890e-001, 0.00000000e+000, 8.37437458e-001,
 4.06537979e-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,
 7.63332662e-001, 8.37437458e-001, 0.00000000e+000,
 2.89407525e-003, 1.31173411e-001, 1.76909716e-001,
 1.98811127e-001, 5.64246762e-001, 2.41686809e-002,
 4.08370570e-001, 1.25850456e-026],
[9.62034639e-001, 3.80283664e-001, 8.90508036e-001,
 4.25266703e-001, 4.06537979e-004, 2.89407525e-003,
 0.00000000e+000, 2.98157345e-001, 8.65526378e-004,
 4.60240532e-009, 1.26581605e-001, 6.74797620e-005,
 8.07912562e-001, 4.60367975e-243],
[4.88086885e-001, 1.31785022e-001, 8.74364323e-002,
 9.09790109e-001, 7.06678685e-001, 1.31173411e-001,
 2.98157345e-001, 0.00000000e+000, 3.41506861e-001,
 1.96808492e-001, 2.53904935e-002, 4.13650739e-003,
 4.36732384e-001, 1.62203448e-001],
[3.67465405e-001, 1.63585747e-001, 9.36216720e-001,
 5.69634028e-001, 2.01668047e-023, 1.76909716e-001,
 8.65526378e-004, 3.41506861e-001, 0.00000000e+000,
 1.12319427e-231, 3.25429744e-001, 2.50330560e-001,
 2.38918636e-001, 7.64706959e-029],
[4.06300660e-001, 5.36514208e-002, 8.39475437e-002,
 2.08874884e-001, 9.38702072e-001, 1.98811127e-001,
 4.60240532e-009, 1.96808492e-001, 1.12319427e-231,
 0.00000000e+000, 6.99615740e-001, 1.03295766e-001,
 2.08799333e-001, 2.85374243e-036],
[9.52261425e-001, 1.60847582e-001, 3.71469037e-001,
 7.03844602e-001, 4.70093788e-001, 5.64246762e-001,
 1.26581605e-001, 2.53904935e-002, 3.25429744e-001,
 6.99615740e-001, 0.00000000e+000, 2.35441825e-001,
 3.15383179e-001, 4.75414918e-001],
[2.28461236e-001, 8.66447868e-001, 8.90503148e-001,
 1.52541637e-002, 6.57076013e-001, 2.41686809e-002,
 6.74797620e-005, 4.13650739e-003, 2.50330560e-001,
 1.03295766e-001, 2.35441825e-001, 0.00000000e+000,
 2.51464473e-001, 1.34826852e-055],
[5.48097586e-001, 1.27389774e-001, 2.38243578e-001,
 9.01602674e-001, 9.84458653e-001, 4.08370570e-001,
 8.07912562e-001, 4.36732384e-001, 2.38918636e-001,
 2.08799333e-001, 3.15383179e-001, 2.51464473e-001,
 0.00000000e+000, 2.27067756e-001],
[9.75106276e-002, 5.31116466e-001, 2.72106138e-001,
 1.98609526e-002, 1.08256524e-007, 1.25850456e-026,
 4.60367975e-243, 1.62203448e-001, 7.64706959e-029,
 2.85374243e-036, 4.75414918e-001, 1.34826852e-055,
 2.27067756e-001, 0.00000000e+000]]))
```

Handling missing values

In [24]:
```
data.isnull().any()
```

Out[24]:
```
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 [25]:
```
data.isnull().sum()
```

Out[25]:
```
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 [26]:
```
data.duplicated()
```

Out[26]:
```
0       False
1       False
2       False
3       False
4       False
        ...
9995    False
9996    False
9997    False
9998    False
9999    False
Length: 10000, dtype: bool
```

No values are NA. The data set is perfect

In [42]:
```
import statsmodels.api as sm
```

In [43]:
```
x=data[["EstimatedSalary"]]
y=data["CreditScore"]
```

In [44]:
```
model=sm.OLS(y,x)
result=model.fit()
result.summary()
```

Out[44]:

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:	Mon, 26 Sep 2022	**Prob (F-statistic):**	0.00
Time:	09:45:47	**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.

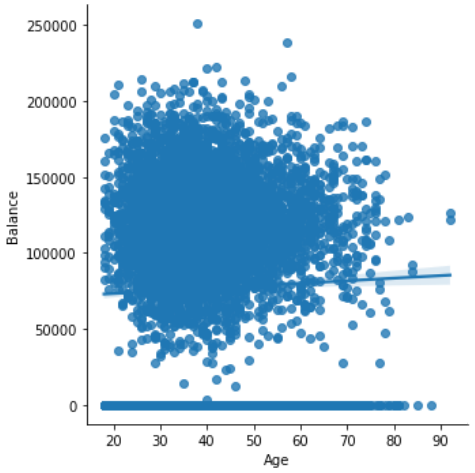
In [45]:
```
from sklearn.preprocessing import scale
```

In [46]:
```
x=scale(x)
x
```

Out[46]:
```
array([[ 0.02188649],
       [ 0.21653375],
       [ 0.2406869 ],
       ...,
       [-1.00864308],
       [-0.12523071],
       [-1.07636976]])
```

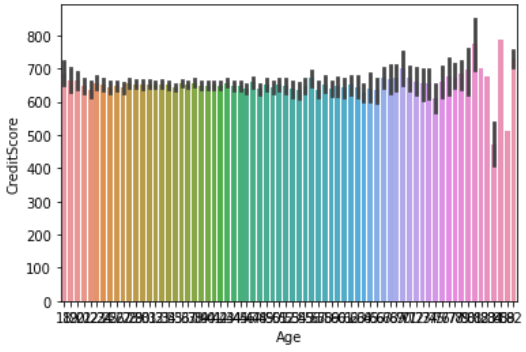
In [49]:
```
sb.lmplot(x='Age',y='Balance',data=data)
```

Out[49]:
```
<seaborn.axisgrid.FacetGrid at 0x7ffb8cb78390>
```



In [125…
```
sb.barplot(x="Age",y="CreditScore",data=data)
```

Out[125]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb8b7b3c10>
```



Finding outliers

In [51]:
```
q = data.quantile(q=[0.75,0.25])
q
```

Out[51]:

	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 [52]:
```
iqr=q.loc[0.75]-q.loc[0.25]
iqr
```

Out[52]:
```
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 [53]:
```
upper= q.loc[0.75]+1.5*iqr
upper
```

Out[53]:
```
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
```

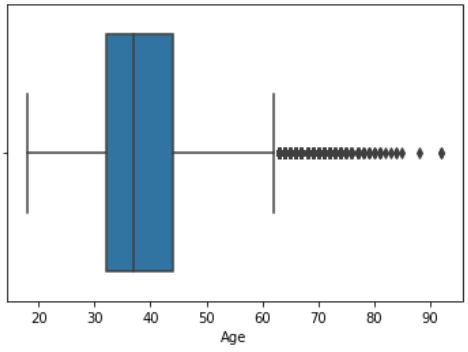
In [54]:
```
lower= q.loc[0.25]-1.5*iqr
lower
```

Out[54]:
```
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
```

Replacing outliers

In [55]:
```
sb.boxplot(data["Age"])
```

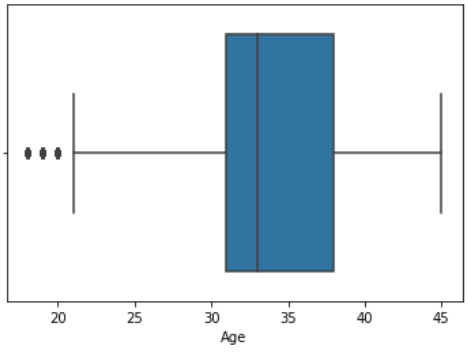
Out[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb8c214510>
```



In [56]:
```
data["Age"]= np.where(data["Age"]>45,31,data["Age"])
```

In [57]:
```
sb.boxplot(data["Age"])
```

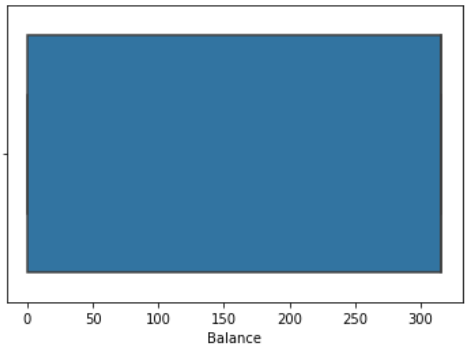
Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb8c1f23d0>
```



In [58]:
```
data["Balance"]= np.where(data["Balance"]>618,316,data["Balance"])
```

In [59]:
```
sb.boxplot(data["Balance"])
```

Out[59]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb8c15b4d0>
```



Check and replace Categorical columns

In [84]:
```
data=pd.read_csv("Churn_Modelling.csv")
```

In [85]:
```
data.head()
```

Out[85]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSala
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.

In [86]:
```
data["Gender"].replace({"Female":0, "Male":1},inplace = True)
data.head(10)
```

Out[86]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSala
0	1	15634602	Hargrave	619	France	0	42	2	0.00	1	1	1	101348.
1	2	15647311	Hill	608	Spain	0	41	1	83807.86	1	0	1	112542.
2	3	15619304	Onio	502	France	0	42	8	159660.80	3	1	0	113931.
3	4	15701354	Boni	699	France	0	39	1	0.00	2	0	0	93826.
4	5	15737888	Mitchell	850	Spain	0	43	2	125510.82	1	1	1	79084.
5	6	15574012	Chu	645	Spain	1	44	8	113755.78	2	1	0	149756.
6	7	15592531	Bartlett	822	France	1	50	7	0.00	2	1	1	10062.
7	8	15656148	Obinna	376	Germany	0	29	4	115046.74	4	1	0	119346.
8	9	15792365	He	501	France	1	44	4	142051.07	2	0	1	74940.
9	10	15592389	H?	684	France	1	27	2	134603.88	1	1	1	71725.

Label Encoding

In [87]:
```
from sklearn.preprocessing import LabelEncoder
l=LabelEncoder()
```

In [88]:
```
data["Age"]=l.fit_transform(data["Age"])
```

In [89]:
```
data.Age.unique()
```

Out[89]:
```
array([24, 23, 21, 25, 26, 32, 11,  9, 13,  6, 16,  7, 17, 27, 40, 14, 20,
       28, 18, 15, 22, 33, 43, 31, 19,  1, 48, 38,  8,  3, 37, 57,  4, 12,
       10, 47, 30, 34, 39, 55, 29, 36, 54,  2, 49, 61, 44, 35, 62, 41, 50,
        5, 42, 52, 45, 46,  0, 64, 51, 56, 53, 58, 59, 68, 67, 66, 60, 63,
       69, 65])
```

In [90]:
```
x=data.iloc[:,0:13].values
x
```

Out[90]:
```
array([[1, 15634602, 'Hargrave', ..., 1, 1, 101348.88],
       [2, 15647311, 'Hill', ..., 0, 1, 112542.58],
       [3, 15619304, 'Onio', ..., 1, 0, 113931.57],
       ...,
       [9998, 15584532, 'Liu', ..., 0, 1, 42085.58],
       [9999, 15682355, 'Sabbatini', ..., 1, 0, 92888.52],
       [10000, 15628319, 'Walker', ..., 1, 0, 38190.78]], dtype=object)
```

In [91]:
```
y=data.iloc[:,13:14].values
y
```

Out[91]:
```
array([[1],
       [0],
       [1],
       ...,
       [1],
       [1],
       [0]])
```

In [92]:
```
from sklearn.preprocessing import  OneHotEncoder
```

In [93]:
```
onehot = OneHotEncoder()
```

In [94]:
```
a=onehot.fit_transform(x[:,0:14]).toarray()
a
```

Out[94]:
```
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.]])
```

Splitting Dependent variables

In [95]:
```
y=data.iloc[:,-1].values
y
```

Out[95]:
```
array([1, 0, 1, ..., 1, 1, 0])
```

In [96]:
```
data=pd.DataFrame({"Age":[1,2,np.nan],"CreditScore":[1,np.nan,np.nan],"Balance":[1,2,3]})
data
```

Out[96]:

	Age	CreditScore	Balance
0	1.0	1.0	1
1	2.0	NaN	2
2	NaN	NaN	3

In [97]:
```
data.isnull().any()
```

Out[97]:
```
Age             True
CreditScore     True
Balance        False
dtype: bool
```

In [98]:
```
data.isnull().sum()
```

Out[98]:
```
Age            1
CreditScore    2
Balance        0
dtype: int64
```

In [99]:
```
data.dropna()
```

Out[99]:

	Age	CreditScore	Balance
0	1.0	1.0	1

In [100…
```
data.dropna(axis=1)
```

Out[100]:

	Balance
0	1
1	2
2	3

Splitting Independent variables

In [103…]
```
data=pd.read_csv("Churn_Modelling.csv")
```

In [104…]
```
x = data.iloc[:, :-1].values
x
```

Out[104]:
```
array([[1, 15634602, 'Hargrave', ..., 1, 1, 101348.88],
       [2, 15647311, 'Hill', ..., 0, 1, 112542.58],
       [3, 15619304, 'Onio', ..., 1, 0, 113931.57],
       ...,
       [9998, 15584532, 'Liu', ..., 0, 1, 42085.58],
       [9999, 15682355, 'Sabbatini', ..., 1, 0, 92888.52],
       [10000, 15628319, 'Walker', ..., 1, 0, 38190.78]], dtype=object)
```

In [131…]
```
x=data.drop(data["Age"],axis=0)
x
```

Out[131]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimatec
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38

9930 rows × 14 columns

Scalling Independent variables

In [107…]
```
x=data["CreditScore"]
s=scale(x)
s
```

Out[107]:
```
array([-0.32622142, -0.44003595, -1.53679418, ...,  0.60498839,
        1.25683526,  1.46377078])
```

In [110…]
```
x1=data["Tenure"]
s1=scale(x1)
s1
```

Out[110]:
```
array([-1.04175968, -1.38753759,  1.03290776, ...,  0.68712986,
       -0.69598177, -0.35020386])
```

Splitting the data into training and testing

In [111…]
```
from sklearn.model_selection import train_test_split
```

In [117…]
```
x=data.iloc[:,0:13].values
x
```

Out[117]:
```
array([[1, 15634602, 'Hargrave', ..., 1, 1, 101348.88],
       [2, 15647311, 'Hill', ..., 0, 1, 112542.58],
       [3, 15619304, 'Onio', ..., 1, 0, 113931.57],
       ...,
       [9998, 15584532, 'Liu', ..., 0, 1, 42085.58],
       [9999, 15682355, 'Sabbatini', ..., 1, 0, 92888.52],
       [10000, 15628319, 'Walker', ..., 1, 0, 38190.78]], dtype=object)
```

In [118…]
```
y=data.iloc[:,13:14].values
y
```

Out[118]:
```
array([[1],
       [0],
       [1],
       ...,
       [1],
       [1],
       [0]])
```

In [119…
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

In [120…
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[120]:
```
((7000, 13), (3000, 13), (7000, 1), (3000, 1))
```

In [121…
```
x_train
```

Out[121]:
```
array([[7682, 15633608, 'Black', ..., 1, 1, 55796.83],
       [9032, 15742323, 'Barese', ..., 1, 0, 19823.02],
       [3692, 15760244, 'Ives', ..., 0, 1, 13848.58],
       ...,
       [3265, 15574372, 'Hoolan', ..., 1, 0, 181429.87],
       [9846, 15664035, 'Parsons', ..., 1, 1, 148750.16],
       [2733, 15592816, 'Udokamma', ..., 1, 0, 118855.26]], dtype=object)
```

In [122…
```
x_test
```

Out[122]:
```
array([[9395, 15615753, 'Upchurch', ..., 1, 1, 192852.67],
       [899, 15654700, 'Fallaci', ..., 1, 0, 128702.1],
       [2399, 15633877, 'Morrison', ..., 1, 1, 75732.25],
       ...,
       [9308, 15680405, "P'eng", ..., 1, 1, 167400.29],
       [8395, 15597983, 'Brown', ..., 1, 1, 70849.47],
       [5234, 15591286, 'Simmons', ..., 1, 1, 33759.41]], dtype=object)
```

In [123…
```
y_train
```

Out[123]:
```
array([[1],
       [0],
       [0],
       ...,
       [0],
       [0],
       [1]])
```

In [132…
```
y_test
```

Out[132]:
```
array([[0],
       [1],
       [0],
       ...,
       [0],
       [0],
       [1]])
```