

Assignment -2

Python Programming

Assignment Date	30 September 2022
Student Name	M.karthikeyan
Student Roll Number	621319106310
Maximum Marks	2 Marks

Question-1:

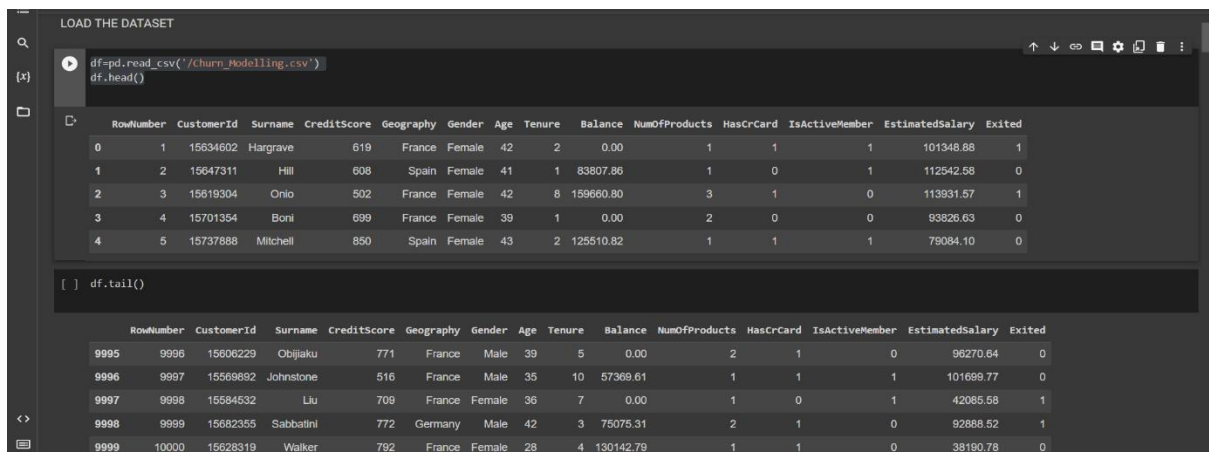
AFTER DOWNLOADING THE DATASET ,IMPORT NECESSARY LIBRARIES

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

LOAD THE DATASET

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

Solution:



LOAD THE DATASET

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

	RowNumber	CustomerId	Surname	Creditscore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9995	9996	15606229	Obijaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15589892	Johnstone	516	France	Male	35	10	57368.61	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

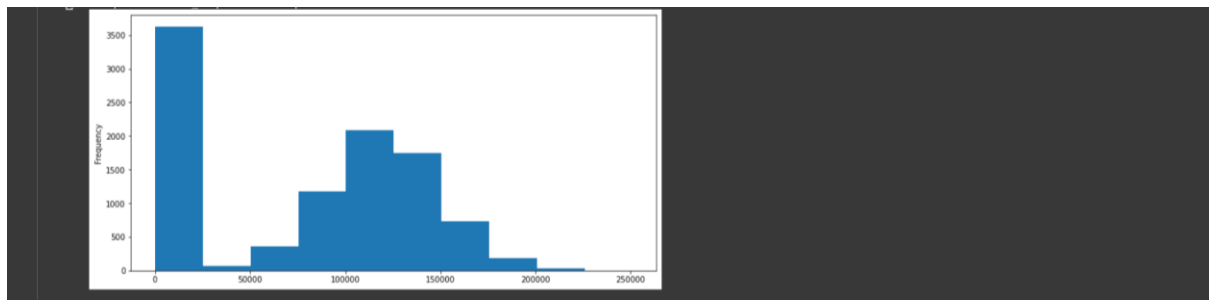
Question-2:

PERFORM VARIOUS VISUALIZATIONS

```
# UNIVARIANT ANALYSIS

plt.figure()
df.Balance.plot(kind='hist' ,figsize=(12,6))
```

Solution:



```
#distribution plot
plt.figure()
sns.distplot(df.Tenure)
```

Solution:



```
plt.figure()
sns.countplot(df.Gender)
```

Solution:



```
df.groupby(['Gender'])['EstimatedSalary'].mean().plot(kind='bar')
```

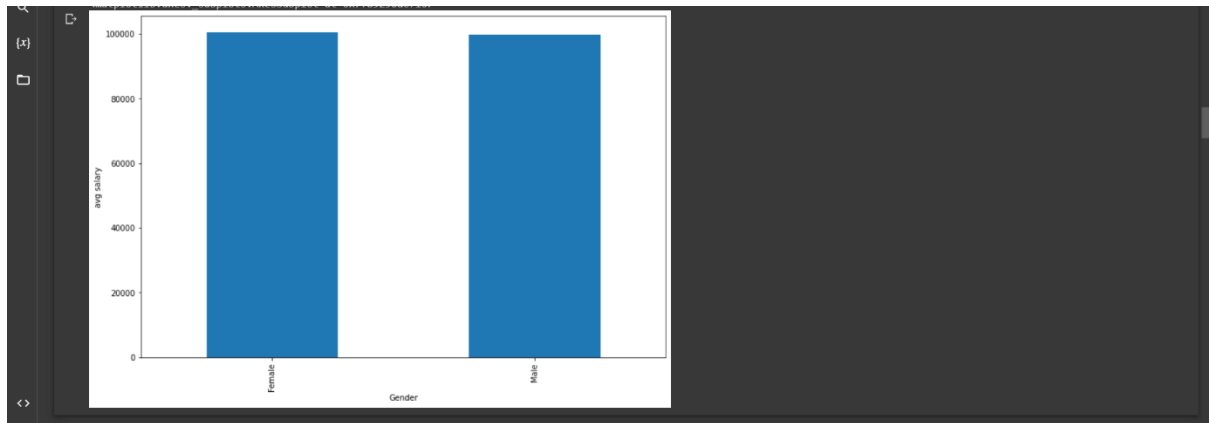
Solution:



Question-3:

```
#BIVARIANT ANALYSIS
df.groupby(['Gender'])['EstimatedSalary'].mean().plot(kind='bar',ylabel='avg salary',figsize=(12,8))
```

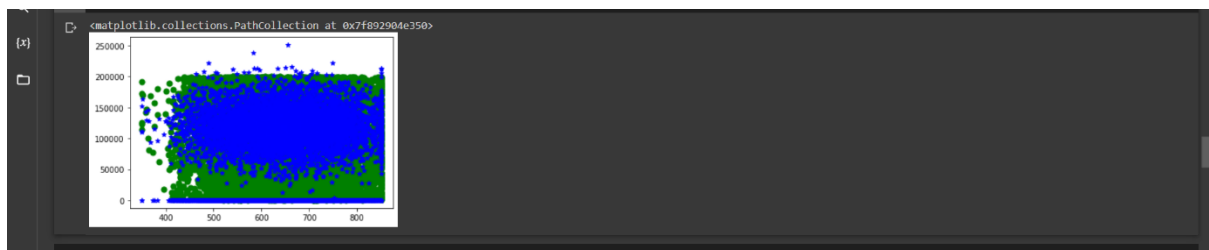
Solution:



Question-4:

```
# 3.3 MULTI-VARIANT ANALYSIS
plt.scatter(x='CreditScore',y='EstimatedSalary',data=df,c='g',s=50)
plt.scatter(x='CreditScore',y='Balance',data=df,c='b',marker='*')
```

Solution:



Question-5:

```
# 4 DESCRIPTIVE STATISTICS ON THE DATASET
df.describe()
```

Solution:

	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

Question-6:

```
# 5 HANDLING THE MISSING VALUES
df.isnull().sum()
```

Solution:

```
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
```

```
df.plot()
```

Solution:



Solution:

```
37 478
38 477
35 474
36 456
34 447
...
92 2
82 1
88 1
85 1
83 1
Name: Age, Length: 70, dtype: int64
```

```
df['NumOfProducts'].value_counts()
```

Solution:

```
df['NumOfProducts'].value_counts()
1    5084
2    4590
3     266
4      60
Name: NumOfProducts, dtype: int64
```

```
df['Age'].unique()
```

Solution:

```
[ ] df['Age'].unique()
array([42, 41, 39, 43, 44, 50, 29, 27, 31, 24, 34, 25, 35, 45, 58, 32, 38,
       46, 36, 33, 40, 51, 61, 49, 37, 19, 66, 56, 26, 21, 55, 75, 22, 30,
       28, 65, 48, 52, 57, 73, 47, 54, 72, 20, 67, 70, 62, 53, 80, 59, 68,
       23, 60, 70, 63, 64, 18, 82, 69, 74, 71, 76, 77, 88, 85, 84, 78, 81,
       92, 83])
```

```
df['NumOfProducts'].unique()
```

Solution:

```
array([1, 3, 2, 4])
```

```
df['CreditScore'].unique()
```

Solution:

```
array([619, 608, 502, 699, 850, 645, 822, 376, 501, 684, 528, 497, 476,
       549, 635, 616, 653, 587, 726, 732, 636, 510, 660, 846, 577, 756,
       571, 574, 411, 591, 533, 553, 520, 722, 475, 490, 804, 582, 472,
       465, 556, 834, 660, 776, 829, 637, 550, 698, 585, 788, 655, 601,
       656, 725, 511, 614, 742, 687, 555, 603, 751, 581, 735, 661, 675,
       738, 813, 657, 604, 519, 664, 678, 757, 416, 665, 777, 543, 506,
       493, 652, 750, 729, 646, 647, 808, 524, 769, 730, 515, 773, 814,
       718, 413, 623, 670, 622, 785, 685, 479, 685, 538, 562, 721, 628,
       668, 828, 674, 625, 432, 770, 758, 795, 686, 789, 589, 461, 584,
       579, 663, 682, 793, 691, 485, 650, 754, 535, 716, 539, 706, 586,
       631, 717, 800, 683, 704, 615, 667, 484, 480, 578, 512, 606, 597,
       778, 514, 525, 715, 580, 807, 521, 759, 516, 711, 618, 643, 671,
       689, 620, 676, 572, 695, 592, 567, 694, 547, 594, 673, 610, 767,
       763, 712, 783, 662, 699, 523, 722, 545, 634, 739, 771, 681, 544,
       696, 766, 727, 693, 557, 531, 498, 651, 791, 733, 811, 707, 714,
       782, 775, 799, 602, 744, 588, 747, 583, 627, 731, 629, 438, 642,
       806, 474, 559, 429, 680, 749, 734, 644, 626, 649, 805, 718, 840,
       638, 654, 762, 568, 613, 522, 737, 648, 443, 640, 540, 460, 593,
       801, 611, 802, 745, 483, 690, 492, 709, 705, 560, 752, 701, 537,
       487, 596, 702, 486, 724, 548, 464, 790, 534, 748, 434, 590, 468,
       590, 810, 816, 536, 793, 774, 621, 569, 650, 798, 641, 542, 692,
       639, 765, 570, 638, 599, 632, 779, 527, 564, 833, 504, 842, 508,
       417, 598, 741, 607, 761, 848, 546, 439, 755, 760, 526, 713, 700,
       666, 566, 495, 688, 612, 477, 427, 839, 819, 720, 459, 503, 624,
       529, 563, 482, 796, 445, 746, 786, 554, 672, 787, 499, 844, 450,
       815, 838, 803, 736, 633, 690, 679, 517, 792, 743, 488, 421, 841,
       708, 507, 585, 456, 435, 561, 518, 565, 728, 784, 552, 609, 764,
       697, 723, 551, 444, 719, 496, 541, 830, 812, 677, 428, 595, 617,
       809, 500, 826, 434, 513, 478, 797, 363, 399, 463, 780, 452, 575,
       837, 794, 824, 428, 823, 781, 849, 489, 431, 457, 768, 831, 359,
       820, 573, 576, 558, 817, 449, 440, 415, 821, 530, 350, 446, 425,
       740, 481, 783, 358, 845, 451, 458, 469, 423, 404, 836, 473, 835,
       466, 491, 351, 827, 843, 365, 532, 414, 453, 471, 401, 810, 832,
       470, 447, 422, 825, 430, 436, 426, 408, 847, 418, 437, 410, 454,
       407, 455, 462, 386, 405, 383, 395, 467, 433, 442, 424, 448, 441,
       367, 412, 382, 373, 419])
```

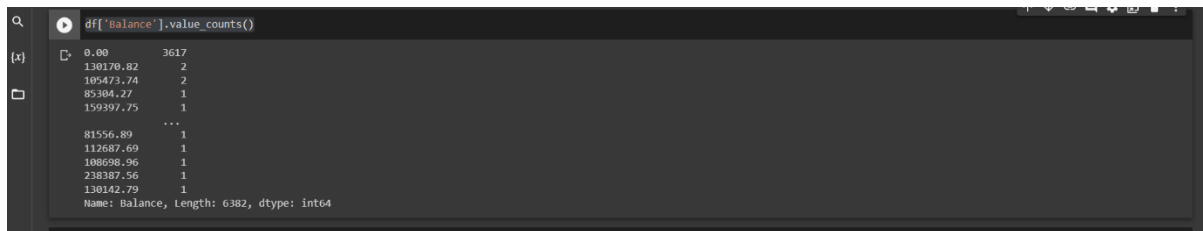
```
df['Balance'].unique()
```

Solution:

```
array([ 0. , 83807.86, 159660.8 , ..., 57369.61, 75075.31, 130142.79])
```

```
df['Balance'].value_counts()
```

Solution:



```
df['Balance'].value_counts()
0.00      3617
130170.82      2
109472.74      2
85384.27       1
159397.75       1
...
81556.89       1
112687.69       1
108698.96       1
238397.56       1
130142.79       1
Name: Balance, Length: 6382, dtype: int64
```

```
df['EstimatedSalary'].value_counts()
```

Solution:



```
[ ] df['EstimatedSalary'].value_counts()
24924.92      2
101348.88      1
55313.44      1
72500.68      1
182692.00      1
...
120893.07      1
188377.21      1
55907.93      1
4522.74       1
38190.78       1
Name: EstimatedSalary, length: 9999, dtype: int64
```

Question-7:

Solution:

```
# 6 OUTLIERS
df['EstimatedSalary'].unique()
```

```
array([101348.88, 112542.58, 113931.57, ..., 42085.58, 92888.52,
       38190.78])
```

Question-8:

Solution:

```
# 7 CHECK FOR CATEGORICAL VALUES AND PERFORM ENCODING
df.info()
```

Solution:

```
# 7. CHECK FOR CATEGORICAL VALUES AND PERFORM ENCODING
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  --
 0   RowNumber          10000 non-null  int64  
 1   CustomerId          10000 non-null  int64  
 2   Surname             10000 non-null  object  
 3   CreditScore          10000 non-null  int64  
 4   Geography           10000 non-null  object  
 5   Gender              10000 non-null  object  
 6   Age                 10000 non-null  int64  
 7   Tenure              10000 non-null  int64  
 8   Balance             10000 non-null  float64 
 9   NumOfProducts       10000 non-null  int64  
10   HasCrCard           10000 non-null  int64  
11   IsActiveMember      10000 non-null  int64  
12   EstimatedSalary     10000 non-null  float64 
13   Exited              10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler

label_encoder = LabelEncoder()

df['Gender'] = label_encoder.fit_transform(df['Gender'])
df.head()
```

Solution:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Ono	502	France	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	0	43	2	125510.62	1	1	1	79084.10	0

```
label_encoder = LabelEncoder()

df['Geography'] = label_encoder.fit_transform(df['Geography'])
df.head()
```

Solution:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Ono	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	2	0	43	2	125510.62	1	1	1	79084.10	0

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

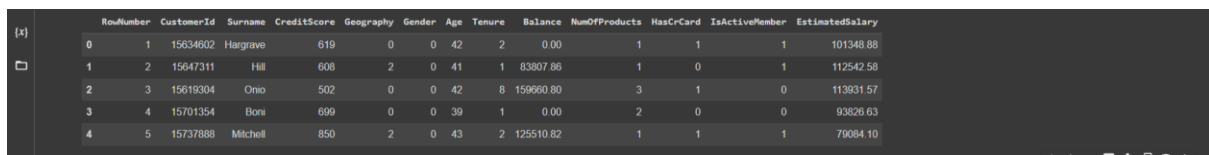
Solution:

```
array([0, 2, 1])
```

Question-9:

```
x=df.iloc[:, :-1]
y=df.iloc[:, -1]
# 8 SPLITTING DATA INTO INDEPENDENT AND DEPENDENT VARIABLES
x.head()
```

Solution:



	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619		0	0	42	2	0.00	1	1	101348.88
1	2	15647311	Hill	608		2	0	41	1	83807.86	1	0	112542.58
2	3	15619304	Onio	502		0	0	42	8	159660.80	3	1	113931.57
3	4	15701354	Boni	699		0	0	39	1	0.00	2	0	93826.63
4	5	15737888	Mitchell	850		2	0	43	2	125510.82	1	1	79084.10

```
y.head()
```

Solution:

```
0 1 1 0 2 1 3 0 4 0 Name:Exited, dtype:int64
```

```
df.info()
```



```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  --
 0   RowNumber             10000 non-null  int64  
 1   CustomerId             10000 non-null  int64  
 2   Surname                10000 non-null  object  
 3   CreditScore            10000 non-null  int64  
 4   Geography              10000 non-null  int64  
 5   Gender                 10000 non-null  int64  
 6   Age                    10000 non-null  int64  
 7   Tenure                 10000 non-null  int64  
 8   Balance                10000 non-null  float64 
 9   NumOfProducts          10000 non-null  int64  
10  HasCrCard              10000 non-null  int64  
11  IsActiveMember         10000 non-null  int64  
12  EstimatedSalary        10000 non-null  float64 
13  Exited                 10000 non-null  int64  
dtypes: float64(2), int64(11), object(1)
memory usage: 1.1+ MB
```

Question-10:

```
df.drop('Surname', axis=1, inplace=True)
x=df.iloc[:, 6:10]
y=df.iloc[:, 6:10]
#9 SPLITTING THE DATA INTO TRAIN AND TEST
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3
, random_state=0)
X_train.head()
```

Solution:



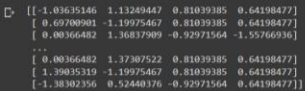
	Tenure	Balance	NumOfProducts	HasCrCard
7681	2	146193.60	2	1
9031	7	0.00	2	1
3691	5	160079.68	1	0
202	5	0.00	1	0
6625	7	143262.04	1	1

Question-11:

```
#10 SCALING INDEPENDENT VARIABLES
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train= sc.fit_transform(X_train)
print(X_train)
```

Solution:

```
#10 SCALING INDEPENDENT VARIABLES
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train= sc.fit_transform(X_train)
print(X_train)
```



```
[[-1.03635146  1.13249447  0.81039385  0.64198477]
 [ 0.69700901 -1.19975467  0.81039385  0.64198477]
 [ 0.00366482  1.36837909 -0.92971564 -1.55766936]
 ...
 [ 0.00366482  1.37307522  0.81039385  0.64198477]
 [ 1.39035319 -1.19975467  0.81039385  0.64198477]
 [-1.38302356  0.52440376 -0.92971564  0.64198477]]
```

```
#10 SCALING INDEPENDENT VARIABLES
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train= sc.fit_transform(X_train)
print(X_train)
```