Name :- MOHAN SAI KOTHAPALLI

Emp ID:- 2112951

ASSESSMENT- Getting and Knowing your Data

Step 1. Import the necessary libraries

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
```

Step 2. Import the dataset from this address.

In [2]:
```
Data = pd.read_csv("https://raw.githubusercontent.com/mohan-sai-12/Cognizant_Intern/main/Assignment1.cs
v")
Data.head()
```

Out[2]:

	user_id	age	gender	occupation	zip_code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

Step 3. Assign it to a variable called users and use the 'user_id' as index

In [3]:
```
Users = Data
Users = Users.set_index('user_id', drop = True)
Users.head()
```

Out[3]:

user_id	age	gender	occupation	zip_code
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213

Step 4. See the first 25 entries

In [4]:
```
Users.head(25)
```

Out[4]:

user_id	age	gender	occupation	zip_code
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213
6	42	M	executive	98101
7	57	M	administrator	91344
8	36	M	administrator	5201
9	29	M	student	1002
10	53	M	lawyer	90703
11	39	F	other	30329
12	28	F	other	6405
13	47	M	educator	29206
14	45	M	scientist	55106
15	49	F	educator	97301
16	21	M	entertainment	10309
17	30	M	programmer	6355
18	35	F	other	37212
19	40	M	librarian	2138
20	42	F	homemaker	95660
21	26	M	writer	30068
22	25	M	writer	40206
23	30	F	artist	48197
24	21	F	artist	94533
25	39	M	engineer	55107

Step 5. See the last 10 entries

In [5]:
```
Users.tail(10)
```

Out[5]:

user_id	age	gender	occupation	zip_code
934	61	M	engineer	22902
935	42	M	doctor	66221
936	24	M	other	32789
937	48	M	educator	98072
938	38	F	technician	55038
939	26	F	student	33319
940	32	M	administrator	2215
941	20	M	student	97229
942	48	F	librarian	78209
943	22	M	student	77841

Step 6. What is the number of observations in the dataset?

In [6]:
```
Users.count(axis=0)
```

Out[6]:
```
age           943
gender        943
occupation    943
zip_code      943
dtype: int64
```

Step 7. What is the number of columns in the dataset?

In [7]:
```
len(Users.axes[1])
```

Out[7]:
```
4
```

Step 8. Print the name of all the columns.

In [8]:
```
list(Users.columns)
```

Out[8]:
```
['age', 'gender', 'occupation', 'zip_code']
```

Step 9. How is the dataset indexed?

In [9]:
```
Users.index
```

Out[9]:
```
Int64Index([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,
            ...
            934, 935, 936, 937, 938, 939, 940, 941, 942, 943],
           dtype='int64', name='user_id', length=943)
```

Step 10. What is the data type of each column?

In [10]:
```
Users.dtypes
```

Out[10]:
```
age            int64
gender        object
occupation    object
zip_code      object
dtype: object
```

Step 11. Print only the occupation column

In [11]:
```
Users.occupation
```

Out[11]:
```
user_id
1         technician
2              other
3             writer
4         technician
5              other
           ...
939          student
940    administrator
941          student
942        librarian
943          student
Name: occupation, Length: 943, dtype: object
```

Step 12. How many different occupations are in this dataset?

In [12]:
```
Users['occupation'].value_counts().count()
```

Out[12]:
```
21
```

Step 13. What is the most frequent occupation?

In [13]:
```
Users['occupation'].value_counts()
```

Out[13]:
```
student          196
other            105
educator          95
administrator     79
engineer          67
programmer        66
librarian         51
writer            45
executive         32
scientist         31
artist            28
technician        27
marketing         26
entertainment     18
healthcare        16
retired           14
lawyer            12
salesman          12
none               9
homemaker          7
doctor             7
Name: occupation, dtype: int64
```

Step 14. Summarize the DataFrame.

In [14]:
```
Users.describe()
```

Out[14]:

	age
count	943.000000
mean	34.051962
std	12.192740
min	7.000000
25%	25.000000
50%	31.000000
75%	43.000000
max	73.000000

Step 15. Summarize all the columns

In [15]:
```
Users.describe(include='all')
```

Out[15]:

	age	gender	occupation	zip_code
count	943.000000	943	943	943
unique	NaN	2	21	795
top	NaN	M	student	55414
freq	NaN	670	196	9
mean	34.051962	NaN	NaN	NaN
std	12.192740	NaN	NaN	NaN
min	7.000000	NaN	NaN	NaN
25%	25.000000	NaN	NaN	NaN
50%	31.000000	NaN	NaN	NaN
75%	43.000000	NaN	NaN	NaN
max	73.000000	NaN	NaN	NaN

Step 16. Summarize only the occupation column

In [16]:
```
Users['occupation'].describe()
```

Out[16]:
```
count         943
unique         21
top       student
freq          196
Name: occupation, dtype: object
```

Step 17. What is the mean age of users?

In [17]:
```
Users['age'].mean()
```

Out[17]:
```
34.05196182396607
```

Step 18. What is the age with least occurrence?

In [18]:
```
Users['age'].value_counts().tail()
```

Out[18]:
```
11    1
10    1
73    1
66    1
7     1
Name: age, dtype: int64
```

Step 19. Write a lambda function to convert a string column to a numerical column

In [19]:
```
users=Users
users.dtypes
```

Out[19]:
```
age            int64
gender        object
occupation    object
zip_code      object
dtype: object
```

In [20]:
```
d = defaultdict(LabelEncoder)
users = users.apply(lambda x: d[x.name].fit_transform(x))
print(users)
users.dtypes
```

```
         age  gender  occupation  zip_code
user_id
1         14       1          19       617
2         43       0          13       688
3         13       1          20       223
4         14       1          19       288
5         23       0          13        51
...      ...     ...         ...       ...
939       16       0          18       236
940       22       1           0       146
941       10       1          18       743
942       38       0          10       554
943       12       1          18       549

[943 rows x 4 columns]
```

Out[20]:
```
age           int64
gender        int32
occupation    int32
zip_code      int32
dtype: object
```

In [ ]: