Practical 9

Identify the column(s) of a given DataFrame which have at least one missing value, count the number of missing values in each column and drop the rows and columns with missing values. Check for the null values. Also remove the duplicate values from the DataFrame. Handle outliers in the Data Frame.

In [24]:
```
import pandas as pd
import numpy as np
import scipy as sp
df= pd.read_csv("clv_data.csv")
df.head()
```

Out[24]:

	Unnamed: 0	id	age	gender	income	days_on_platform	city	purchases
0	0	0	NaN	Male	126895.0	14.0	San Francisco	0
1	1	1	NaN	Male	161474.0	14.0	Tokyo	0
2	2	2	24.0	Male	104723.0	34.0	London	1
3	3	3	29.0	Male	NaN	28.0	London	2
4	4	4	18.0	Female	132181.0	26.0	London	2

In [25]:
```
df.shape
```

Out[25]:
```
(5000, 8)
```

In [28]:
```
df.describe()
```

Out[28]:

count	5000.000000	5000.000000	2554.000000	4958.000000	4859.000000	5000.00
mean	2499.500000	2499.500000	30.202036	79619.163372	24.389998	1.10
std	1443.520003	1443.520003	12.129439	60314.847584	18.153388	1.18
min	0.000000	0.000000	10.000000	4.000000	1.000000	0.00
25%	1249.750000	1249.750000	19.000000	32741.500000	10.000000	0.00
50%	2499.500000	2499.500000	30.000000	66109.500000	21.000000	1.00
75%	3749.250000	3749.250000	41.000000	115437.500000	35.000000	2.00

max	4999.000000	4999.000000	50.000000	388572.000000	111.000000	6.00

1. Identify the column(s) of a given DataFrame which have at least one missing value

In [30]:
```
df.isna()
```

Out[30]:

	Unamed: 0	id	age	gender	income	days_on_platform	city	purchases
0	False	False	True	False	False	False	False	False
1	False	False	True	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	True	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
4995	False	False	True	False	False	False	False	False
4996	False	False	True	False	False	False	False	False
4997	False	False	True	False	False	False	False	False
4998	False	False	True	False	False	False	False	False
4999	False	False	True	False	False	False	False	False

5000 rows × 8 columns

```
df.isnull()
```

Out[31]:

	Unamed: 0	id	age	gender	income	days_on_platform	city	purchases
0	False	False	True	False	False	False	False	False
1	False	False	True	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	True	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
4995	False	False	True	False	False	False	False	False
4996	False	False	True	False	False	False	False	False
4997	False	False	True	False	False	False	False	False
4998	False	False	True	False	False	False	False	False
4999	False	False	True	False	False	False	False	False

5000 rows × 8 columns

In [32]:
```
df.isna().any()
```

Out[32]:
```
Unnamed: 0          False
id                  False
age                  True
gender              False
income               True
days_on_platform     True
city                False
purchases           False
dtype: bool
```

2. Count the number of missing values in each column

In [34]:
```
df.isna().sum()
```

Out[34]:
```
Unnamed: 0            0
id                    0
age                2446
gender                0
income               42
days_on_platform    141
city                  0
purchases             0
dtype: int64
```

3. Drop the rows and columns with missing values. Check for the null values.

In [35]:
```
df.dropna()
```

Out[35]:

	Unnamed: 0	id	age	gender	income	days_on_platform	city	purchases
2	2	2	24.0	Male	104723.0	34.0	London	1
4	4	4	18.0	Female	132181.0	26.0	London	2
5	5	5	23.0	Male	12315.0	14.0	New York City	0
8	8	8	46.0	Male	129157.0	23.0	New York City	0
9	9	9	49.0	Female	76842.0	19.0	Tokyo	2
...	...	...	...	...	...	...	...	...
4986	4986	4986	23.0	Male	75425.0	6.0	London	1
4989	4989	4989	47.0	Female	84987.0	30.0	Tokyo	0
4990	4990	4990	33.0	Male	3020.0	89.0	New York City	0
4991	4991	4991	36.0	Female	26173.0	34.0	Tokyo	0
4992	4992	4992	26.0	Male	88858.0	14.0	Miami	3

2457 rows × 8 columns

In [36]:
```
df.fillna(method='pad')
```

Out[36]:

	Unnamed: 0	id	age	gender	income	days_on_platform	city	purchases
0	0	0	NaN	Male	126895.0	14.0	San Francisco	0
1	1	1	NaN	Male	161474.0	14.0	Tokyo	0
2	2	2	24.0	Male	104723.0	34.0	London	1
3	3	3	29.0	Male	104723.0	28.0	London	2
4	4	4	18.0	Female	132181.0	26.0	London	2
...	...	...	...	...	...	...	...	...
4995	4995	4995	26.0	Female	212261.0	28.0	San Francisco	1
4996	4996	4996	26.0	Male	70228.0	12.0	San Francisco	0
4997	4997	4997	26.0	Male	64995.0	14.0	New York City	0
4998	4998	4998	26.0	Male	56144.0	4.0	New York City	2
4999	4999	4999	26.0	Female	110977.0	29.0	London	0

5000 rows × 8 columns

In [37]:
```
df.fillna(method='bfill')
```

Out[37]:

	Unnamed: 0	id	age	gender	income	days_on_platform	city	purchases
0	0	0	24.0	Male	126895.0	14.0	San Francisco	0
1	1	1	24.0	Male	161474.0	14.0	Tokyo	0
2	2	2	24.0	Male	104723.0	34.0	London	1
3	3	3	29.0	Male	132181.0	28.0	London	2
4	4	4	18.0	Female	132181.0	26.0	London	2
...	...	...	...	...	...	...	...	...
4995	4995	4995	NaN	Female	212261.0	28.0	San Francisco	1
4996	4996	4996	NaN	Male	70228.0	12.0	San Francisco	0
4997	4997	4997	NaN	Male	64995.0	14.0	New York City	0
4998	4998	4998	NaN	Male	56144.0	4.0	New York City	2
4999	4999	4999	NaN	Female	110977.0	29.0	London	0

5000 rows × 8 columns

In [38]:
```
df.replace(to_replace=np.nan,  value=-99)
```

Out[38]:

	Unnamed: 0	id	age	gender	income	days_on_platform	city	purchases
0	0	0	-99.0	Male	126895.0	14.0	San Francisco	0
1	1	1	-99.0	Male	161474.0	14.0	Tokyo	0
2	2	2	24.0	Male	104723.0	34.0	London	1
3	3	3	29.0	Male	-99.0	28.0	London	2
4	4	4	18.0	Female	132181.0	26.0	London	2
...	...	...	...	...	...	...	...	...
4995	4995	4995	-99.0	Female	212261.0	28.0	San Francisco	1
4996	4996	4996	-99.0	Male	70228.0	12.0	San Francisco	0
4997	4997	4997	-99.0	Male	64995.0	14.0	New York City	0
4998	4998	4998	-99.0	Male	56144.0	4.0	New York City	2
4999	4999	4999	-99.0	Female	110977.0	29.0	London	0

5000 rows × 8 columns

4. Also remove the duplicate values from the DataFrame.

In [20]:
```
df.drop_duplicates()
```

Out[20]:

	Unnamed: 0	id	age	gender	income	days_on_platform	city	purchases
2	2	2	24.0	Male	104723.0	34.0	London	1
4	4	4	18.0	Female	132181.0	26.0	London	2
5	5	5	23.0	Male	12315.0	14.0	New York City	0
8	8	8	46.0	Male	129157.0	23.0	New York City	0
9	9	9	49.0	Female	76842.0	19.0	Tokyo	2
...	...	...	...	...	...	...	...	...
4986	4986	4986	23.0	Male	75425.0	6.0	London	1
4989	4989	4989	47.0	Female	84987.0	30.0	Tokyo	0
4990	4990	4990	33.0	Male	3020.0	89.0	New York City	0
4991	4991	4991	36.0	Female	26173.0	34.0	Tokyo	0
4992	4992	4992	26.0	Male	88858.0	14.0	Miami	3

2457 rows × 8 columns

Handle outliers in the Data Frame.

In [35]:
```
def outliers(df, columns):
    for col in columns:
        if df[col].dtype.kind in 'biufc':
            Q1 = df[col].quantile(0.25)
            Q3 = df[col].quantile(0.75)
            IQR = Q3 - Q1
            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR
            df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df

numeric_cols = df.select_dtypes(include=np.number).columns.tolist()
df_no_outliers = outliers(df_no_duplicates, numeric_cols)
print("\nDataFrame after removing outliers:\n", df_no_outliers)
```

```
DataFrame after removing outliers:
       Unnamed: 0    id   age  gender    income  days_on_platform   \
2               2     2  24.0    Male  104723.0              34.0
4               4     4  18.0  Female  132181.0              26.0
5               5     5  23.0    Male   12315.0              14.0
8               8     8  46.0    Male  129157.0              23.0
9               9     9  49.0  Female   76842.0              19.0
...           ...   ...   ...     ...       ...               ...
4984         4984  4984  24.0  Female  225155.0               8.0
4986         4986  4986  23.0    Male   75425.0               6.0
4989         4989  4989  47.0  Female   84987.0              30.0
4991         4991  4991  36.0  Female   26173.0              34.0
4992         4992  4992  26.0    Male   88858.0              14.0

                city  purchases
2             London          1
4             London          2
5      New York City          0
8      New York City          0
9              Tokyo          2
...              ...        ...
4984   San Francisco          2
4986          London          1
4989           Tokyo          0
4991           Tokyo          0
4992           Miami          3
```

```
[2362 rows x 8 columns]
```

Data normalization

In [22]:
```
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
```

In [23]:
```
df_cars = pd.DataFrame([[120000, 11], [250000, 11.5], [175000, 15.8], [350000, 17],
                        columns=['odometer_reading', 'fuel_economy'])
print(df_cars)
```

```
   odometer_reading  fuel_economy
0            120000          11.0
1            250000          11.5
2            175000          15.8
3            350000          17.0
4            400000          10.0
```

In [25]:
```
scaler = MinMaxScaler()
df_norm = pd.DataFrame(scaler.fit_transform(df_cars), columns=df_cars.columns)

df_norm
```

Out[25]:

	odometer_reading	fuel_economy
0	0.000000	0.142857
1	0.464286	0.214286
2	0.196429	0.828571
3	0.821429	1.000000
4	1.000000	0.000000

In [26]:
```
def z_score(df):
    df_std = df.copy()
    for column in df_std.columns:
        df_std[column] = (df_std[column] - df_std[column].mean()) / df_std[column].

    return df_std

df_cars_standardized = z_score(df_cars)
df_cars_standardized
```

Out[26]:

	odometer_reading	fuel_economy
0	-1.189512	-0.659120
1	-0.077019	-0.499139
2	-0.718842	0.876693

3	0.778745	1.260647
4	1.206628	-0.979081

In [27]:
```
from sklearn.preprocessing import StandardScaler

std_scaler = StandardScaler()
std_scaler
df_std = pd.DataFrame(std_scaler.fit_transform(df_cars), columns=df_cars.columns)

df_std
```

Out[27]:

	odometer_reading	fuel_economy
0	-1.329915	-0.736918
1	-0.086110	-0.558055
2	-0.803690	0.980173
3	0.870664	1.409446
4	1.349051	-1.094646