In [1]:

```
# Importing important libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt # plots
import seaborn as sns ## generates enhanced plots
```

Reading the datasets

In [2]:

```
df1 = pd.read_csv('Customer_ID.csv')
df1.head()
```

Out[2]:

	Customer ID	Gender	Age	Income (USD/Month)
0	29290	Male	28	10813
1	27703	Male	27	9237
2	28712	Male	53	11242
3	28020	Male	23	23327
4	27182	Male	33	8536

In [3]:

```
df2 = pd.read_csv('City.csv')
df2.head()
```

Out[3]:

	City	Population	Users
0	NEW YORK NY	8,405,837	302,149
1	CHICAGO IL	1,955,130	164,468
2	LOS ANGELES CA	1,595,037	144,132
3	MIAMI FL	1,339,155	17,675
4	SILICON VALLEY	1,177,609	27,247

In [4]:

```
df3 = pd.read_csv('Transaction_ID.csv')
df3.head()
```

Out[4]:

	Transaction ID	Customer ID	Payment_Mode
0	10000011	29290	Card
1	10000012	27703	Card
2	10000013	28712	Cash
3	10000014	28020	Cash
4	10000015	27182	Card

In [5]:

```
df4 = pd.read_csv('Cab_Data.csv')
df4.head()
```

Out[5]:

	Transaction ID	Date of Travel	Company	City	KM Travelled	Price Charged	Cost of Trip
0	10000011	42377	Pink Cab	ATLANTA GA	30.45	370.95	313.635
1	10000012	42375	Pink Cab	ATLANTA GA	28.62	358.52	334.854
2	10000013	42371	Pink Cab	ATLANTA GA	9.04	125.20	97.632
3	10000014	42376	Pink Cab	ATLANTA GA	33.17	377.40	351.602
4	10000015	42372	Pink Cab	ATLANTA GA	8.73	114.62	97.776

We can add dataframe 1 and 3 through 'Customer ID', dataframe 2 and 4 through 'City' column, table 3 and 4 through 'Transaction ID'.

How many rows and columns do we have?

Find the number of rows and columns in any dataframe by using the shape function in Pandas/Numpy

In [6]:

```
print("The number of rows is %d" %(df1.shape[0]))
print("The number of columns is %d" %(df1.shape[1]))
```

```
The number of rows is 49171
The number of columns is 4
```

In [7]:

```
print("The number of rows is %d" %(df2.shape[0]))
print("The number of columns is %d" %(df2.shape[1]))
```

```
The number of rows is 20
The number of columns is 3
```

In [8]:

```
print("The number of rows is %d" %(df3.shape[0]))
print("The number of columns is %d" %(df3.shape[1]))
```

```
The number of rows is 440098
The number of columns is 3
```

In [9]:

```
print("The number of rows is %d" %(df4.shape[0]))
print("The number of columns is %d" %(df4.shape[1]))
```

```
The number of rows is 359392
The number of columns is 7
```

Let's get the type of data

In [10]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49171 entries, 0 to 49170
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Customer ID         49171 non-null  int64 
 1   Gender              49171 non-null  object
 2   Age                 49171 non-null  int64 
 3   Income (USD/Month)  49171 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 1.5+ MB
```

In [11]:

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   City        20 non-null     object
 1   Population  20 non-null     object
 2   Users       20 non-null     object
dtypes: object(3)
memory usage: 608.0+ bytes
```

In [12]:

```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440098 entries, 0 to 440097
Data columns (total 3 columns):
 #   Column          Non-Null Count   Dtype 
---  ------          --------------   ----- 
 0   Transaction ID  440098 non-null  int64 
 1   Customer ID     440098 non-null  int64 
 2   Payment_Mode    440098 non-null  object
dtypes: int64(2), object(1)
memory usage: 10.1+ MB
```

In [13]:

```
df4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 359392 entries, 0 to 359391
Data columns (total 7 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   Transaction ID  359392 non-null  int64  
 1   Date of Travel  359392 non-null  int64  
 2   Company         359392 non-null  object 
 3   City            359392 non-null  object 
 4   KM Travelled    359392 non-null  float64
 5   Price Charged   359392 non-null  float64
 6   Cost of Trip    359392 non-null  float64
dtypes: float64(3), int64(2), object(2)
memory usage: 19.2+ MB
```

In [14]:

```
## generate descriptive statistics which is very helpful

df1.describe()
```

Out[14]:

	Customer ID	Age	Income (USD/Month)
count	49171.000000	49171.000000	49171.000000
mean	28398.252283	35.363121	15015.631856
std	17714.137333	12.599066	8002.208253
min	1.000000	18.000000	2000.000000
25%	12654.500000	25.000000	8289.500000
50%	27631.000000	33.000000	14656.000000
75%	43284.500000	42.000000	21035.000000
max	60000.000000	65.000000	35000.000000

In [15]:

```
df2.describe()
```

Out[15]:

	City	Population	Users
count	20	20	20
unique	20	20	20
top	NEW YORK NY	8,405,837	302,149
freq	1	1	1

In [16]:

```
df3.describe()
```

Out[16]:

	Transaction ID	Customer ID
count	4.400980e+05	440098.000000
mean	1.022006e+07	23619.513120
std	1.270455e+05	21195.549816
min	1.000001e+07	1.000000
25%	1.011004e+07	3530.000000
50%	1.022006e+07	15168.000000
75%	1.033008e+07	43884.000000
max	1.044011e+07	60000.000000

In [17]:

```
df4.describe()
```

Out[17]:

	Transaction ID	Date of Travel	KM Travelled	Price Charged	Cost of Trip
count	3.593920e+05	359392.000000	359392.000000	359392.000000	359392.000000
mean	1.022076e+07	42964.067998	22.567254	423.443311	286.190113
std	1.268058e+05	307.467197	12.233526	274.378911	157.993661
min	1.000001e+07	42371.000000	1.900000	15.600000	19.000000
25%	1.011081e+07	42697.000000	12.000000	206.437500	151.200000
50%	1.022104e+07	42988.000000	22.440000	386.360000	282.480000
75%	1.033094e+07	43232.000000	32.960000	583.660000	413.683200
max	1.044011e+07	43465.000000	48.000000	2048.030000	691.200000

Get columns with missing data using Pandas isna

In [18]:

```
df1.isna().any(axis = 0)
```

Out[18]:

```
Customer ID           False
Gender                False
Age                   False
Income (USD/Month)    False
dtype: bool
```

In [19]:

```
df2.isna().any(axis = 0)
```

Out[19]:

```
City          False
Population    False
Users         False
dtype: bool
```

In [20]:

```
df3.isna().any(axis = 0)
```

Out[20]:

```
Transaction ID    False
Customer ID       False
Payment_Mode      False
dtype: bool
```

In [21]:

```
df4.isna().any(axis = 0)
```

Out[21]:

```
Transaction ID    False
Date of Travel    False
Company           False
City              False
KM Travelled      False
Price Charged     False
Cost of Trip      False
dtype: bool
```

There is no nan values in any dataframe

Let's drop duplicates

In [22]:

```
df1.drop_duplicates()
```

Out[22]:

	Customer ID	Gender	Age	Income (USD/Month)
0	29290	Male	28	10813
1	27703	Male	27	9237
2	28712	Male	53	11242
3	28020	Male	23	23327
4	27182	Male	33	8536
...	...	...	...	...
49166	12490	Male	33	18713
49167	14971	Male	30	15346
49168	41414	Male	38	3960
49169	41677	Male	23	19454
49170	39761	Female	32	10128

49171 rows × 4 columns

In [23]:

```
df2.drop_duplicates()
```

Out[23]:

	City	Population	Users
0	NEW YORK NY	8,405,837	302,149
1	CHICAGO IL	1,955,130	164,468
2	LOS ANGELES CA	1,595,037	144,132
3	MIAMI FL	1,339,155	17,675
4	SILICON VALLEY	1,177,609	27,247
5	ORANGE COUNTY	1,030,185	12,994
6	SAN DIEGO CA	959,307	69,995
7	PHOENIX AZ	943,999	6,133
8	DALLAS TX	942,908	22,157
9	ATLANTA GA	814,885	24,701
10	DENVER CO	754,233	12,421
11	AUSTIN TX	698,371	14,978
12	SEATTLE WA	671,238	25,063
13	TUCSON AZ	631,442	5,712
14	SAN FRANCISCO CA	629,591	213,609
15	SACRAMENTO CA	545,776	7,044
16	PITTSBURGH PA	542,085	3,643
17	WASHINGTON DC	418,859	127,001
18	NASHVILLE TN	327,225	9,270
19	BOSTON MA	248,968	80,021

In [24]:

```
df3.drop_duplicates()
```

Out[24]:

	Transaction ID	Customer ID	Payment_Mode
0	10000011	29290	Card
1	10000012	27703	Card
2	10000013	28712	Cash
3	10000014	28020	Cash
4	10000015	27182	Card
...	...	...	...
440093	10440104	53286	Cash
440094	10440105	52265	Cash
440095	10440106	52175	Card
440096	10440107	52917	Card
440097	10440108	51587	Card

440098 rows × 3 columns

In [25]:

```
df4.drop_duplicates()
```

Out[25]:

	Transaction ID	Date of Travel	Company	City	KM Travelled	Price Charged	Cost of Trip
0	10000011	42377	Pink Cab	ATLANTA GA	30.45	370.95	313.6350
1	10000012	42375	Pink Cab	ATLANTA GA	28.62	358.52	334.8540
2	10000013	42371	Pink Cab	ATLANTA GA	9.04	125.20	97.6320
3	10000014	42376	Pink Cab	ATLANTA GA	33.17	377.40	351.6020
4	10000015	42372	Pink Cab	ATLANTA GA	8.73	114.62	97.7760
...	...	...	...	...	...	...	...
359387	10440101	43108	Yellow Cab	WASHINGTON DC	4.80	69.24	63.3600
359388	10440104	43104	Yellow Cab	WASHINGTON DC	8.40	113.75	106.8480
359389	10440105	43105	Yellow Cab	WASHINGTON DC	27.75	437.07	349.6500
359390	10440106	43105	Yellow Cab	WASHINGTON DC	8.80	146.19	114.0480
359391	10440107	43102	Yellow Cab	WASHINGTON DC	12.76	191.58	177.6192

359392 rows × 7 columns

It looks like there isn't any duplicate values in any dataframe

In [26]:

```
mergedRes1 = pd.merge(df1, df3, on = 'Customer ID')
mergedRes1
```

Out[26]:

	Customer ID	Gender	Age	Income (USD/Month)	Transaction ID	Payment_Mode
0	29290	Male	28	10813	10000011	Card
1	29290	Male	28	10813	10351127	Cash
2	29290	Male	28	10813	10412921	Card
3	27703	Male	27	9237	10000012	Card
4	27703	Male	27	9237	10320494	Card
...	...	...	...	...	...	...
440093	12490	Male	33	18713	10439799	Cash
440094	14971	Male	30	15346	10439820	Card
440095	41414	Male	38	3960	10439838	Card
440096	41677	Male	23	19454	10439840	Cash
440097	39761	Female	32	10128	10439846	Card

440098 rows × 6 columns

In [27]:

```
mergedRes2 = pd.merge(df2, df4, on = 'City')
mergedRes2
```

Out[27]:

	City	Population	Users	Transaction ID	Date of Travel	Company	KM Travelled	Price Charged	Co
0	NEW YORK NY	8,405,837	302,149	10000139	42377	Pink Cab	17.85	242.90	198.
1	NEW YORK NY	8,405,837	302,149	10000140	42378	Pink Cab	25.30	407.21	255.
2	NEW YORK NY	8,405,837	302,149	10000141	42375	Pink Cab	16.32	236.41	186.
3	NEW YORK NY	8,405,837	302,149	10000142	42373	Pink Cab	12.43	194.61	144.
4	NEW YORK NY	8,405,837	302,149	10000143	42378	Pink Cab	29.70	434.57	350.
...	...	...	...	...	...	...	...	...	
359387	BOSTON MA	248,968	80,021	10437872	43104	Yellow Cab	1.96	26.23	25.
359388	BOSTON MA	248,968	80,021	10437873	43105	Yellow Cab	23.52	380.22	338.
359389	BOSTON MA	248,968	80,021	10437874	43106	Yellow Cab	29.10	393.33	391.
359390	BOSTON MA	248,968	80,021	10437875	43108	Yellow Cab	3.33	42.31	46.
359391	BOSTON MA	248,968	80,021	10437876	43103	Yellow Cab	23.75	349.91	290.

359392 rows × 9 columns

In [28]:

```
mergedRes3 = pd.merge(df3, df4, on = 'Transaction ID')
mergedRes3
```

Out[28]:

	Transaction ID	Customer ID	Payment_Mode	Date of Travel	Company	City	KM Travelled	Ch
0	10000011	29290	Card	42377	Pink Cab	ATLANTA GA	30.45	
1	10000012	27703	Card	42375	Pink Cab	ATLANTA GA	28.62	
2	10000013	28712	Cash	42371	Pink Cab	ATLANTA GA	9.04	
3	10000014	28020	Cash	42376	Pink Cab	ATLANTA GA	33.17	
4	10000015	27182	Card	42372	Pink Cab	ATLANTA GA	8.73	
...	...	...	...	...	...	...	...	
359387	10440101	52392	Cash	43108	Yellow Cab	WASHINGTON DC	4.80	
359388	10440104	53286	Cash	43104	Yellow Cab	WASHINGTON DC	8.40	
359389	10440105	52265	Cash	43105	Yellow Cab	WASHINGTON DC	27.75	
359390	10440106	52175	Card	43105	Yellow Cab	WASHINGTON DC	8.80	
359391	10440107	52917	Card	43102	Yellow Cab	WASHINGTON DC	12.76	

359392 rows × 9 columns

Now, combining the mergedRes dataframes.

In [29]:

```
data1 = pd.merge(mergedRes1, mergedRes2, on = 'Transaction ID')
data1
```

Out[29]:

	Customer ID	Gender	Age	Income (USD/Month)	Transaction ID	Payment_Mode	City	Populati
0	29290	Male	28	10813	10000011	Card	ATLANTA GA	814,8
1	29290	Male	28	10813	10351127	Cash	ATLANTA GA	814,8
2	29290	Male	28	10813	10412921	Card	ATLANTA GA	814,8
3	27703	Male	27	9237	10000012	Card	ATLANTA GA	814,8
4	27703	Male	27	9237	10320494	Card	ATLANTA GA	814,8
...	...	...	...	...	...	...	...	
359387	38520	Female	42	19417	10439790	Card	SEATTLE WA	671,2
359388	12490	Male	33	18713	10439799	Cash	SILICON VALLEY	1,177,6
359389	41414	Male	38	3960	10439838	Card	TUCSON AZ	631,4
359390	41677	Male	23	19454	10439840	Cash	TUCSON AZ	631,4
359391	39761	Female	32	10128	10439846	Card	TUCSON AZ	631,4

359392 rows × 14 columns

In [30]:

```
data = pd.merge(mergedRes1, mergedRes2, on = 'Transaction ID')
data
```

Out[30]:

	Customer ID	Gender	Age	Income (USD/Month)	Transaction ID	Payment_Mode	City	Populati
0	29290	Male	28	10813	10000011	Card	ATLANTA GA	814,8
1	29290	Male	28	10813	10351127	Cash	ATLANTA GA	814,8
2	29290	Male	28	10813	10412921	Card	ATLANTA GA	814,8
3	27703	Male	27	9237	10000012	Card	ATLANTA GA	814,8
4	27703	Male	27	9237	10320494	Card	ATLANTA GA	814,8
...	...	...	...	...	...	...	...	
359387	38520	Female	42	19417	10439790	Card	SEATTLE WA	671,2
359388	12490	Male	33	18713	10439799	Cash	SILICON VALLEY	1,177,6
359389	41414	Male	38	3960	10439838	Card	TUCSON AZ	631,4
359390	41677	Male	23	19454	10439840	Cash	TUCSON AZ	631,4
359391	39761	Female	32	10128	10439846	Card	TUCSON AZ	631,4

359392 rows × 14 columns