In [26]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [27]:
```
df = pd.read_csv(r"C:\Users\ippil\Downloads\uberdrive (1).csv")
```

In [28]:
```
df
```

Out[28]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE
0	01-01-2016 21:11	01-01-2016 21:17	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertai
1	01-02-2016 01:25	01-02-2016 01:37	Business	Fort Pierce	Fort Pierce	5.0	Na
2	01-02-2016 20:25	01-02-2016 20:38	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplie
3	01-05-2016 17:31	01-05-2016 17:45	Business	Fort Pierce	Fort Pierce	4.7	Meetin
4	01-06-2016 14:42	01-06-2016 15:49	Business	Fort Pierce	West Palm Beach	63.7	Customer Vis
...	...	...	...	...	...	...	
1150	12/31/2016 1:07	12/31/2016 1:14	Business	Karachi	Karachi	0.7	Meetin
1151	12/31/2016 13:24	12/31/2016 13:42	Business	Karachi	Unknown Location	3.9	Temporary Sit
1152	12/31/2016 15:03	12/31/2016 15:38	Business	Unknown Location	Unknown Location	16.2	Meetin
1153	12/31/2016 21:32	12/31/2016 21:50	Business	Katunayake	Gampaha	6.4	Temporary Sit
1154	12/31/2016 22:08	12/31/2016 23:51	Business	Gampaha	Ilukwatta	48.2	Temporary Sit

1155 rows × 7 columns

Data Inspection

- To understand the structure and content of the dataset.

In [34]:
```
df.columns
```

Out[34]:
```
Index(['START_DATE*', 'END_DATE*', 'CATEGORY*', 'START*', 'STOP*', 'MILES*',
       'PURPOSE*'],
      dtype='object')
```

In [36]:
```
df.shape
```

Out[36]:
```
(1155, 7)
```

In [38]:
```
df.size
```

Out[38]:
```
8085
```

In [40]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1155 entries, 0 to 1154
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   START_DATE*  1155 non-null   object 
 1   END_DATE*    1155 non-null   object 
 2   CATEGORY*    1155 non-null   object 
 3   START*       1155 non-null   object 
 4   STOP*        1155 non-null   object 
 5   MILES*       1155 non-null   float64
 6   PURPOSE*     653 non-null    object 
dtypes: float64(1), object(6)
memory usage: 63.3+ KB
```

In [42]:
```
df.describe()
```

Out[42]:

	MILES*
count	1155.000000
mean	10.566840
std	21.579106
min	0.500000
25%	2.900000
50%	6.000000
75%	10.400000
max	310.300000

In [44]:
```
df.head()
```

Out[44]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
0	01-01-2016 21:11	01-01-2016 21:17	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain
1	01-02-2016 01:25	01-02-2016 01:37	Business	Fort Pierce	Fort Pierce	5.0	NaN
2	01-02-2016 20:25	01-02-2016 20:38	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies
3	01-05-2016 17:31	01-05-2016 17:45	Business	Fort Pierce	Fort Pierce	4.7	Meeting
4	01-06-2016 14:42	01-06-2016 15:49	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit

In [46]:
```
df.tail()
```

Out[46]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
1150	12/31/2016 1:07	12/31/2016 1:14	Business	Karachi	Karachi	0.7	Meeting
1151	12/31/2016 13:24	12/31/2016 13:42	Business	Karachi	Unknown Location	3.9	Temporary Site
1152	12/31/2016 15:03	12/31/2016 15:38	Business	Unknown Location	Unknown Location	16.2	Meeting
1153	12/31/2016 21:32	12/31/2016 21:50	Business	Katunayake	Gampaha	6.4	Temporary Site
1154	12/31/2016 22:08	12/31/2016 23:51	Business	Gampaha	Ilukwatta	48.2	Temporary Site

Questions

Q1. Show the last 10 records of the dataset.

In [50]:
```
df.tail(10)
```

Out[50]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE
1145	12/30/2016 10:15	12/30/2016 10:33	Business	Karachi	Karachi	2.8	Errand/Supplie
1146	12/30/2016 11:31	12/30/2016 11:56	Business	Karachi	Karachi	2.9	Errand/Supplie
1147	12/30/2016 15:41	12/30/2016 16:03	Business	Karachi	Karachi	4.6	Errand/Supplie
1148	12/30/2016 16:45	12/30/2016 17:08	Business	Karachi	Karachi	4.6	Meetin
1149	12/30/2016 23:06	12/30/2016 23:10	Business	Karachi	Karachi	0.8	Customer Vis
1150	12/31/2016 1:07	12/31/2016 1:14	Business	Karachi	Karachi	0.7	Meetin
1151	12/31/2016 13:24	12/31/2016 13:42	Business	Karachi	Unknown Location	3.9	Temporary Sit
1152	12/31/2016 15:03	12/31/2016 15:38	Business	Unknown Location	Unknown Location	16.2	Meetin
1153	12/31/2016 21:32	12/31/2016 21:50	Business	Katunayake	Gampaha	6.4	Temporary Sit
1154	12/31/2016 22:08	12/31/2016 23:51	Business	Gampaha	Ilukwatta	48.2	Temporary Sit

Q2. Show the first 10 records of the dataset.

In [52]:
```
df.head(10)
```

Out[52]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
0	01-01-2016 21:11	01-01-2016 21:17	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain
1	01-02-2016 01:25	01-02-2016 01:37	Business	Fort Pierce	Fort Pierce	5.0	NaN
2	01-02-2016 20:25	01-02-2016 20:38	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies
3	01-05-2016 17:31	01-05-2016 17:45	Business	Fort Pierce	Fort Pierce	4.7	Meeting
4	01-06-2016 14:42	01-06-2016 15:49	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit
5	01-06-2016 17:15	01-06-2016 17:19	Business	West Palm Beach	West Palm Beach	4.3	Meal/Entertain
6	01-06-2016 17:30	01-06-2016 17:35	Business	West Palm Beach	Palm Beach	7.1	Meeting
7	01-07-2016 13:27	01-07-2016 13:33	Business	Cary	Cary	0.8	Meeting
8	01-10-2016 08:05	01-10-2016 08:25	Business	Cary	Morrisville	8.3	Meeting
9	01-10-2016 12:17	01-10-2016 12:44	Business	Jamaica	New York	16.5	Customer Visit

Q3. Show the dimension(number of rows and columns) of the dataset.

In [60]:
```
print(df.shape)
print("IT shows that the dataset contains 1155 rows and 7 columns")
```

```
(1155, 7)
IT shows that the dataset contains 1155 rows and 7 columns
```

Q4. Show the size (Total number of elements) of the dataset.

In [62]:
```
df.size
```

Out[62]: 8085

Q5. Display the information about all the variables of the data set. What can you infer from the output?

In [66]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1155 entries, 0 to 1154
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   START_DATE*  1155 non-null   object 
 1   END_DATE*    1155 non-null   object 
 2   CATEGORY*    1155 non-null   object 
 3   START*       1155 non-null   object 
 4   STOP*        1155 non-null   object 
 5   MILES*       1155 non-null   float64
 6   PURPOSE*     653 non-null    object 
dtypes: float64(1), object(6)
memory usage: 63.3+ KB
```

Q6. Check for missing values.

In [72]:
```
df.isna()
```

Out[72]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	True
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
1150	False	False	False	False	False	False	False
1151	False	False	False	False	False	False	False
1152	False	False	False	False	False	False	False
1153	False	False	False	False	False	False	False
1154	False	False	False	False	False	False	False

1155 rows × 7 columns

Q7. How many missing values are present in the entire dataset?

In [76]:
```
df.isna().sum()
```

Out[76]:
```
START_DATE*      0
END_DATE*        0
CATEGORY*        0
START*           0
STOP*            0
MILES*           0
PURPOSE*       502
dtype: int64
```

Q8. Get the summary of the original data.

In [78]:
```
df.describe(include='all')
```

Out[78]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
count	1155	1155	1155	1155	1155	1155.000000	653
unique	1154	1154	2	176	187	NaN	10
top	6/28/2016 23:34	6/28/2016 23:59	Business	Cary	Cary	NaN	Meeting
freq	2	2	1078	201	203	NaN	187
mean	NaN	NaN	NaN	NaN	NaN	10.566840	NaN
std	NaN	NaN	NaN	NaN	NaN	21.579106	NaN
min	NaN	NaN	NaN	NaN	NaN	0.500000	NaN
25%	NaN	NaN	NaN	NaN	NaN	2.900000	NaN
50%	NaN	NaN	NaN	NaN	NaN	6.000000	NaN
75%	NaN	NaN	NaN	NaN	NaN	10.400000	NaN
max	NaN	NaN	NaN	NaN	NaN	310.300000	NaN

Q9. Drop the missing values and store the data in a new dataframe (name it"df")

In [84]:
```
df1 = df.dropna()
df1
```

Out[84]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE
0	01-01-2016 21:11	01-01-2016 21:17	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertai
2	01-02-2016 20:25	01-02-2016 20:38	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplie
3	01-05-2016 17:31	01-05-2016 17:45	Business	Fort Pierce	Fort Pierce	4.7	Meetin
4	01-06-2016 14:42	01-06-2016 15:49	Business	Fort Pierce	West Palm Beach	63.7	Customer Vis
5	01-06-2016 17:15	01-06-2016 17:19	Business	West Palm Beach	West Palm Beach	4.3	Meal/Entertai
...	...	...	...	...	...	...	
1150	12/31/2016 1:07	12/31/2016 1:14	Business	Karachi	Karachi	0.7	Meetin
1151	12/31/2016 13:24	12/31/2016 13:42	Business	Karachi	Unknown Location	3.9	Temporary Sit
1152	12/31/2016 15:03	12/31/2016 15:38	Business	Unknown Location	Unknown Location	16.2	Meetin
1153	12/31/2016 21:32	12/31/2016 21:50	Business	Katunayake	Gampaha	6.4	Temporary Sit
1154	12/31/2016 22:08	12/31/2016 23:51	Business	Gampaha	Ilukwatta	48.2	Temporary Sit

653 rows × 7 columns

Q10. Check the information of the dataframe(df).

In [90]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 653 entries, 0 to 1154
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   START_DATE*  653 non-null    object 
 1   END_DATE*    653 non-null    object 
 2   CATEGORY*    653 non-null    object 
 3   START*       653 non-null    object 
 4   STOP*        653 non-null    object 
 5   MILES*       653 non-null    float64
 6   PURPOSE*     653 non-null    object 
dtypes: float64(1), object(6)
memory usage: 40.8+ KB
```

Q11. Get the unique start locations.

In [92]:
```
df['START*'].unique()
```

Out[92]:
```
array(['Fort Pierce', 'West Palm Beach', 'Cary', 'Jamaica', 'New York',
       'Elmhurst', 'Midtown', 'East Harlem', 'Flatiron District',
       'Midtown East', 'Hudson Square', 'Lower Manhattan',
       "Hell's Kitchen", 'Downtown', 'Gulfton', 'Houston', 'Eagan Park',
       'Morrisville', 'Durham', 'Farmington Woods', 'Lake Wellingborough',
       'Fayetteville Street', 'Raleigh', 'Whitebridge', 'Hazelwood',
       'Fairmont', 'Meredith Townes', 'Apex', 'Chapel Hill', 'Northwoods',
       'Edgehill Farms', 'Eastgate', 'East Elmhurst', 'Long Island City',
       'Katunayaka', 'Colombo', 'Nugegoda', 'Unknown Location',
       'Islamabad', 'R?walpindi', 'Noorpur Shahan', 'Preston',
       'Heritage Pines', 'Tanglewood', 'Waverly Place', 'Wayne Ridge',
       'Westpark Place', 'East Austin', 'The Drag', 'South Congress',
       'Georgian Acres', 'North Austin', 'West University', 'Austin',
       'Katy', 'Sharpstown', 'Sugar Land', 'Galveston', 'Port Bolivar',
       'Washington Avenue', 'Briar Meadow', 'Latta', 'Jacksonville',
       'Lake Reams', 'Orlando', 'Kissimmee', 'Daytona Beach', 'Ridgeland',
       'Florence', 'Meredith', 'Holly Springs', 'Chessington', 'Burtrose',
       'Parkway', 'Mcvan', 'Capitol One', 'University District',
       'Seattle', 'Redmond', 'Bellevue', 'San Francisco', 'Palo Alto',
       'Sunnyvale', 'Newark', 'Menlo Park', 'Old City', 'Savon Height',
       'Kilarney Woods', 'Townes at Everett Crossing', 'Huntington Woods',
       'Weston', 'Seaport', 'Medical Centre', 'Rose Hill', 'Soho',
       'Tribeca', 'Financial District', 'Oakland', 'Emeryville',
       'Berkeley', 'Kenner', 'CBD', 'Lower Garden District', 'Storyville',
       'New Orleans', 'Chalmette', 'Arabi', 'Pontchartrain Shores',
       'Metairie', 'Summerwinds', 'Parkwood', 'Banner Elk', 'Boone',
       'Stonewater', 'Lexington Park at Amberly', 'Winston Salem',
       'Asheville', 'Topton', 'Renaissance', 'Santa Clara', 'Ingleside',
       'West Berkeley', 'Mountain View', 'El Cerrito', 'Krendle Woods',
       'Fuquay-Varina', 'Rawalpindi', 'Lahore', 'Karachi', 'Katunayake',
       'Gampaha'], dtype=object)
```

Q12. What is the total number of unique start locations?

In [102…
```
df['START*'].nunique()
```

Out[102… 176

Q13. What is the total number of unique stop locations.

In [106…
```
df['STOP*'].nunique()
```

Out[106… 187

Q14. Display all Uber trips that has the starting point as San Francisco.

In [108…
```
df[df['START*'] =='San Francisco']
```

Out[108…

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
362	05-09-2016 14:39	05-09-2016 15:06	Business	San Francisco	Palo Alto	20.5	Between Offices
440	6/14/2016 16:09	6/14/2016 16:39	Business	San Francisco	Emeryville	11.6	Meeting
836	10/19/2016 14:02	10/19/2016 14:31	Business	San Francisco	Berkeley	10.8	NaN
917	11-07-2016 19:17	11-07-2016 19:57	Business	San Francisco	Berkeley	13.2	Between Offices
919	11-08-2016 12:16	11-08-2016 12:49	Business	San Francisco	Berkeley	11.3	Meeting
927	11-09-2016 18:40	11-09-2016 19:17	Business	San Francisco	Oakland	12.7	Customer Visit
933	11-10-2016 15:17	11-10-2016 15:22	Business	San Francisco	Oakland	9.9	Temporary Site
966	11/15/2016 20:44	11/15/2016 21:00	Business	San Francisco	Berkeley	11.8	Temporary Site

Q15. What is the most popular starting point for the Uber drivers?

In [110…
```
df['START*'].value_counts(ascending = False).head(1)
```

Out[110…
```
START*
Cary    201
Name: count, dtype: int64
```

Q16. What is the most popular dropping point for the Uber drivers?

In [112…
```
df['STOP*'].value_counts(ascending = False).head(1)
```

Out[112…
```
STOP*
Cary    203
Name: count, dtype: int64
```

Q17. What is the most frequent route taken by Uber drivers.

In [120…
```
df.groupby(['START*','STOP*'])['MILES*'].size().sort_values(ascending=False).head(1
```

Out[120…
```
START*            STOP*
Unknown Location  Unknown Location         86
Morrisville       Cary                    75
Cary              Morrisville             67
                  Cary                    53
                  Durham                  36
                                          ..
Midtown           Midtown                  2
New Orleans       Metairie                 2
Huntington Woods  Weston                   2
The Drag          Congress Ave District    2
Metairie          New Orleans              2
Name: MILES*, Length: 100, dtype: int64
```

Q18. Display all types of purposes for the trip in an array.

In [122…
```
df['PURPOSE*']
```

Out[122…
```
0        Meal/Entertain
1                   NaN
2       Errand/Supplies
3               Meeting
4        Customer Visit
            ...
1150            Meeting
1151     Temporary Site
1152            Meeting
1153     Temporary Site
1154     Temporary Site
Name: PURPOSE*, Length: 1155, dtype: object
```

Q19. Plot a bar graph of Purpose vs Miles(Distance). What can you infer from the plot

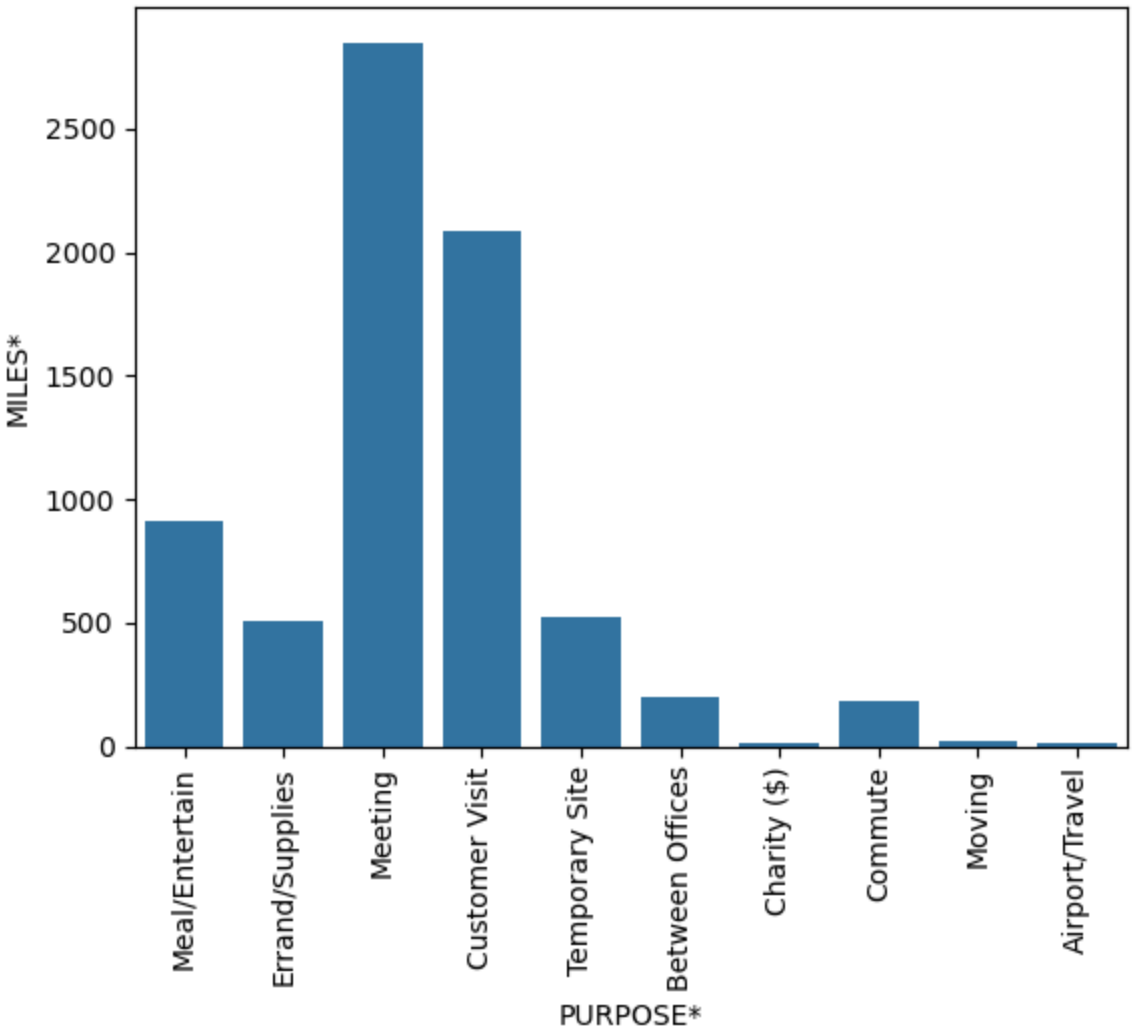
In [132…
```
sns.barplot(data =df,y = 'MILES*',x='PURPOSE*',ci=False,estimator='sum')
plt.xticks(rotation =90)
```

```
C:\Users\ippil\AppData\Local\Temp\ipykernel_12732\2818505978.py:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=('ci', False)` for the same effect.

  sns.barplot(data =df,y = 'MILES*',x='PURPOSE*',ci=False,estimator='sum')
```

Out[132…
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
 [Text(0, 0, 'Meal/Entertain'),
  Text(1, 0, 'Errand/Supplies'),
  Text(2, 0, 'Meeting'),
  Text(3, 0, 'Customer Visit'),
  Text(4, 0, 'Temporary Site'),
  Text(5, 0, 'Between Offices'),
  Text(6, 0, 'Charity ($)'),
  Text(7, 0, 'Commute'),
  Text(8, 0, 'Moving'),
  Text(9, 0, 'Airport/Travel')])
```



- From the bar graph we can conclude that the purpose of travelling is more for mettings and for customer visit compared to the remaining purpose.

Q20. Display a dataframe of Purpose and the total distance travelled for that particular Purpose.

In [165…
```
df1
```

Out[165…

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
0	01-01-2016 21:11	01-01-2016 21:17	Business	San Francisco	Fort Pierce	5.1	Meal/Entertain
2	01-02-2016 20:25	01-02-2016 20:38	Business	San Francisco	Fort Pierce	4.8	Errand/Supplies
3	01-05-2016 17:31	01-05-2016 17:45	Business	San Francisco	Fort Pierce	4.7	Meeting
4	01-06-2016 14:42	01-06-2016 15:49	Business	San Francisco	West Palm Beach	63.7	Customer Visit
5	01-06-2016 17:15	01-06-2016 17:19	Business	San Francisco	West Palm Beach	4.3	Meal/Entertain
…	…	…	…	…	…	…	…
1150	12/31/2016 1:07	12/31/2016 1:14	Business	San Francisco	Karachi	0.7	Meeting
1151	12/31/2016 13:24	12/31/2016 13:42	Business	San Francisco	Unknown Location	3.9	Temporary Site
1152	12/31/2016 15:03	12/31/2016 15:38	Business	San Francisco	Unknown Location	16.2	Meeting
1153	12/31/2016 21:32	12/31/2016 21:50	Business	San Francisco	Gampaha	6.4	Temporary Site
1154	12/31/2016 22:08	12/31/2016 23:51	Business	San Francisco	Ilukwatta	48.2	Temporary Site

653 rows × 7 columns

In [167…

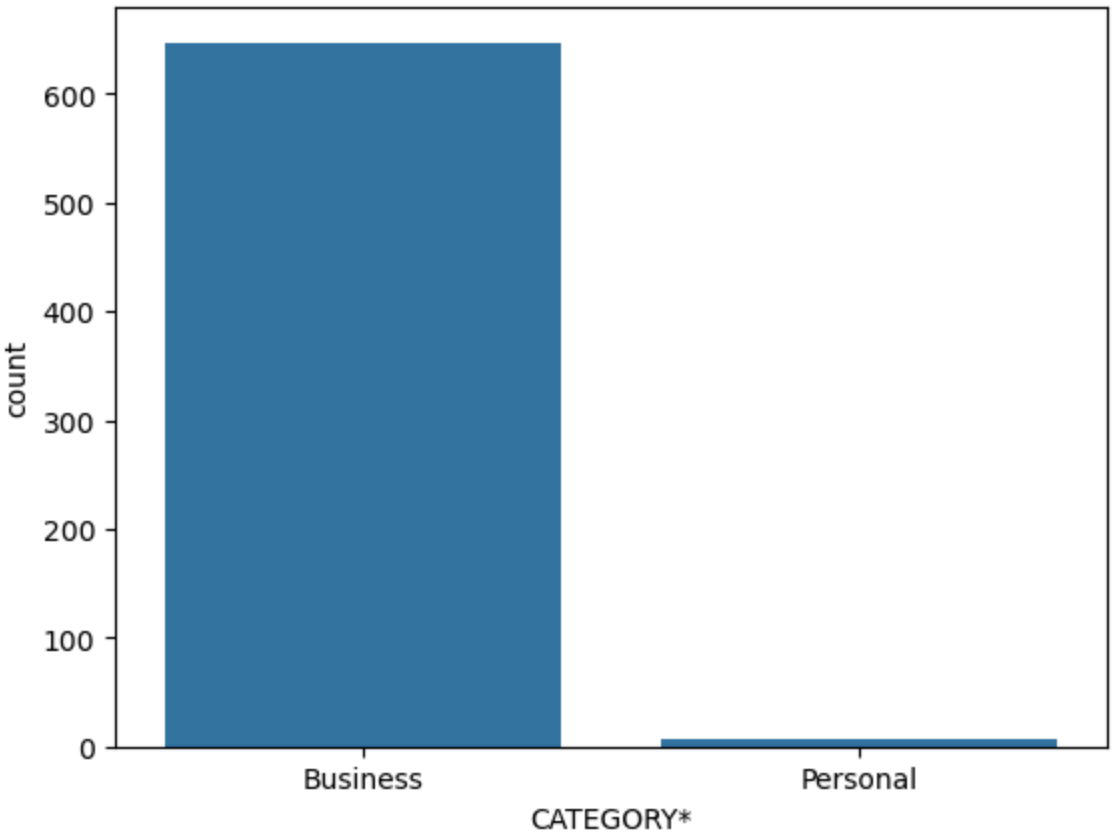
```
df2 =df1.groupby('PURPOSE*')['MILES*'].sum()
print(df2)
```

```
PURPOSE*
Airport/Travel       16.5
Between Offices     197.0
Charity ($)          15.1
Commute             180.2
Customer Visit     2089.5
Errand/Supplies     508.0
Meal/Entertain      911.7
Meeting            2851.3
Moving               18.2
Temporary Site      523.7
Name: MILES*, dtype: float64
```

Q21. Generate a plot showing count of trips vs category of trips. What can you infer from the plot

In [175…
```
sns.countplot(x = df1['CATEGORY*'])
```

Out[175…
```
<Axes: xlabel='CATEGORY*', ylabel='count'>
```

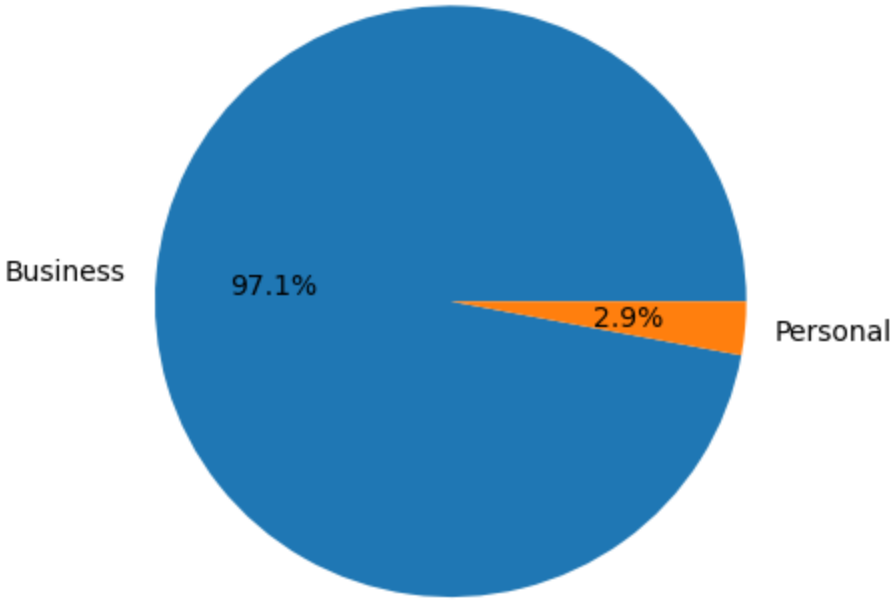


- We can conclude that based on the count plot most of the trips are of business trips compared to personal trips

Q22. What percentage of Miles were clocked under Business Category and what percentage of Miles were clocked under Personal Category ?

In [179…
```
plt.pie(df1.groupby('CATEGORY*')['MILES*'].sum(),labels=df1.groupby('CATEGORY*')['M
```

Out[179…
```
([<matplotlib.patches.Wedge at 0x1b4b7722e70>,
  <matplotlib.patches.Wedge at 0x1b4b7834380>],
 [Text(-1.0953743213574358, 0.100772496797177, 'Business'),
  Text(1.095374329613059, -0.10077240706037442, 'Personal')],
 [Text(-0.5974769025586013, 0.05496681643482381, '97.1%'),
  Text(0.5974769070616684, -0.054966767487476954, '2.9%')])
```



- 97.1% miles were clocked under business category and the remaining 2.9% are of personal category.

In [ ]: