What is Data Analysis?

Data Analysis is the process of inspecting, cleaning, transforming, and modeling data to discover useful information, draw conclusions, and support decision-making. It is a crucial step in any data-driven approach, helping organizations and individuals make informed decisions by interpreting data patterns, trends, and insights.

Steps in Data Analysis:

Data Collection: Gathering raw data from various sources such as databases, APIs, surveys, or logs.

Data Cleaning: Removing or correcting inaccuracies, duplicates, and inconsistencies in the data.

Exploratory Data Analysis (EDA): Summarizing the main characteristics of the data using statistical methods and visualization tools.

Data Transformation: Preparing the data for analysis by normalizing, aggregating, or structuring it appropriately.

Analysis and Modeling: Applying techniques like statistical methods, machine learning, or predictive modeling to extract insights.

Visualization and Reporting: Presenting the results through dashboards, charts, graphs, or reports to communicate findings effectively.

```
In [ ]:
```

Tools: Excel, Python (Pandas, NumPy, Matplotlib, Seaborn):

```
In [ ]:
```

Applications of Data Analysis:

Business: Market trend analysis, customer segmentation, and performance evaluation.

Healthcare: Patient diagnosis, medical research, and drug effectiveness studies.

Finance: Fraud detection, risk assessment, and investment strategies.

Education: Analyzing student performance and improving learning outcomes.

Sports: Player performance evaluation and game strategy optimization.

```
In [ ]:
```

Simple Scenario:

A retail company wants to analyze its sales data to understand trends and improve sales performance.

1. Data Collection

Example: Collect sales data for the past year from the company’s point-of-sale (POS) system.

Data Includes:

1. Date of sale
2. Product category
3. Quantity sold
4. Revenue
5. Customer demographics (age, location)

Purpose: Gather raw data that answers questions like "Which products sell the most?" or "What regions are underperforming?"

```
In [ ]:
```

2. Data Cleaning

Example: Inspect the dataset for issues.

1. Remove duplicate sales entries.

2. Correct inconsistencies in product names (e.g., "t-shirt" vs. "T-shirt").

3. Handle missing data, such as revenue values for some transactions.

Why?: Clean data ensures accurate and reliable analysis.

```
In [ ]:
```

3. Exploratory Data Analysis (EDA)

Example: Use descriptive statistics and visualizations to explore the data.

1. Find the total sales revenue.
2. Identify which product categories generate the most revenue.

3. Plot sales trends over time (e.g., sales increase during the holiday season).

Tool: Use Python (Matplotlib, Pandas) or Excel to create charts and summaries.

Outcome:

"Electronics" is the top-selling category.

Sales peak in December and dip in February.

```
In [ ]:
```

4. Data Transformation

Example: Prepare the data for deeper analysis.

1. Group data by month to analyze monthly trends.
2. Aggregate data by customer age groups to understand customer segmentation.

   Why?: It makes patterns and relationships easier to identify.

```
In [ ]:
```

5. Analysis and Modeling

Example: Answer key business questions:

1. Use trend analysis to predict next year's sales during peak seasons.
2. Apply clustering to group customers by purchase behavior.
3. Perform a correlation analysis to check if discounts lead to higher sales.

Outcome:

1. Discounts are most effective for electronics during the holiday season.
2. Younger customers (ages 18–25) prefer fashion-related products.

```
In [ ]:
```

6. Visualization and Reporting

Example: Present findings to the management team.

1. Create a bar chart showing monthly sales revenue.
2. Use a pie chart to represent sales by product category.
3. Build a dashboard in Tableau or Power BI for interactive exploration.

Insights Shared:

1. Focus on stocking electronics in December for maximum sales.
2. Offer targeted discounts for fashion products to younger customers.

```
In [1]: import pandas as pd
        ud=pd.read_csv("Uber.csv")
        type(ud)

        #to load a csv file and view its type
```

```
Out[1]: pandas.core.frame.DataFrame
```

```
In [2]: ud.head()

        #to view first 5 lines of a file
```

Out[2]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
0	1/1/2016 21:11	1/1/2016 21:17	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain
1	1/2/2016 1:25	1/2/2016 1:37	Business	Fort Pierce	Fort Pierce	5.0	NaN
2	1/2/2016 20:25	1/2/2016 20:38	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies
3	1/5/2016 17:31	1/5/2016 17:45	Business	Fort Pierce	Fort Pierce	4.7	Meeting
4	1/6/2016 14:42	1/6/2016 15:49	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit

In [3]:

```
ud.tail()

#to view last five lines of a file
```

Out[3]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
1151	12/31/2016 13:24	12/31/2016 13:42	Business	Kar?chi	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
1155	Totals	NaN	NaN	NaN	NaN	12204.7	NaN

In [4]:

```
ud.shape

#to view the comple size as in matrix form
```

Out[4]:

```
(1156, 7)
```

In [5]:

```
ud.describe

#to get the complete description of our data
```

Out[5]:
```
<bound method NDFrame.describe of            START_DATE*         END_DATE* CATEGOR
Y*             START*  \
0       1/1/2016 21:11    1/1/2016 21:17  Business       Fort Pierce
1        1/2/2016 1:25     1/2/2016 1:37  Business       Fort Pierce
2       1/2/2016 20:25    1/2/2016 20:38  Business       Fort Pierce
3       1/5/2016 17:31    1/5/2016 17:45  Business       Fort Pierce
4       1/6/2016 14:42    1/6/2016 15:49  Business       Fort Pierce
...                ...               ...       ...               ...
1151  12/31/2016 13:24  12/31/2016 13:42  Business           Kar?chi
1152  12/31/2016 15:03  12/31/2016 15:38  Business  Unknown Location
1153  12/31/2016 21:32  12/31/2016 21:50  Business        Katunayake
1154  12/31/2016 22:08  12/31/2016 23:51  Business           Gampaha
1155            Totals               NaN       NaN               NaN

                 STOP*   MILES*         PURPOSE*
0          Fort Pierce      5.1  Meal/Entertain
1          Fort Pierce      5.0              NaN
2          Fort Pierce      4.8  Errand/Supplies
3          Fort Pierce      4.7          Meeting
4      West Palm Beach     63.7   Customer Visit
...                ...      ...              ...
1151  Unknown Location      3.9   Temporary Site
1152  Unknown Location     16.2          Meeting
1153           Gampaha      6.4   Temporary Site
1154         Ilukwatta     48.2   Temporary Site
1155               NaN  12204.7              NaN

[1156 rows x 7 columns]>
```

In [6]:
```
ud.columns

#to get info about the columns in out file
```

Out[6]:
```
Index(['START_DATE*', 'END_DATE*', 'CATEGORY*', 'START*', 'STOP*', 'MILES*',
       'PURPOSE*'],
      dtype='object')
```

In [7]:
```
ud.dtypes

#to get info about the datatypes of each column
```

Out[7]:
```
START_DATE*     object
END_DATE*       object
CATEGORY*       object
START*          object
STOP*           object
MILES*         float64
PURPOSE*        object
dtype: object
```

In [8]:
```
print(ud.isnull().sum())
```

```
START_DATE*      0
END_DATE*        1
CATEGORY*        1
START*           1
STOP*            1
MILES*           0
PURPOSE*       503
dtype: int64
```

In [9]:
```
ud.loc[:,["START_DATE*"]]

#loc is used to get the detailes of specific columns, it uses slicing works with bo
```

Out[9]:

	START_DATE*
0	1/1/2016 21:11
1	1/2/2016 1:25
2	1/2/2016 20:25
3	1/5/2016 17:31
4	1/6/2016 14:42
...	...
1151	12/31/2016 13:24
1152	12/31/2016 15:03
1153	12/31/2016 21:32
1154	12/31/2016 22:08
1155	Totals

1156 rows × 1 columns

In [10]:

```
ud.iloc[20:40,[0,1,2]]

#iloc is used to get the detailes of specific columns, it uses slicing works with c
```

Out[10]:

	START_DATE*	END_DATE*	CATEGORY*
20	1/12/2016 15:13	1/12/2016 15:28	Business
21	1/12/2016 15:42	1/12/2016 15:54	Business
22	1/12/2016 16:02	1/12/2016 17:00	Business
23	1/13/2016 13:54	1/13/2016 14:07	Business
24	1/13/2016 15:00	1/13/2016 15:28	Business
25	1/14/2016 16:29	1/14/2016 17:05	Business
26	1/14/2016 21:39	1/14/2016 21:45	Business
27	1/15/2016 0:41	1/15/2016 1:01	Business
28	1/15/2016 11:43	1/15/2016 12:03	Business
29	1/15/2016 13:26	1/15/2016 13:44	Business
30	1/18/2016 14:55	1/18/2016 15:06	Business
31	1/18/2016 16:13	1/18/2016 16:24	Business
32	1/19/2016 9:09	1/19/2016 9:23	Business
33	1/19/2016 10:55	1/19/2016 11:09	Business
34	1/20/2016 10:36	1/20/2016 11:11	Business
35	1/20/2016 11:48	1/20/2016 12:19	Business
36	1/20/2016 13:25	1/20/2016 14:19	Business
37	1/21/2016 14:25	1/21/2016 14:29	Business
38	1/21/2016 14:43	1/21/2016 14:51	Business
39	1/21/2016 16:01	1/21/2016 16:06	Business

In [11]:
```
ud.loc[3,["START_DATE*","END_DATE*"]]
```

Out[11]:
```
START_DATE*     1/5/2016 17:31
END_DATE*       1/5/2016 17:45
Name: 3, dtype: object
```

In [12]:
```
ud.head(100)

#to get specific number of lines we use head(n) where n is number of lines
```

Out[12]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
0	1/1/2016 21:11	1/1/2016 21:17	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain
1	1/2/2016 1:25	1/2/2016 1:37	Business	Fort Pierce	Fort Pierce	5.0	NaN
2	1/2/2016 20:25	1/2/2016 20:38	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies
3	1/5/2016 17:31	1/5/2016 17:45	Business	Fort Pierce	Fort Pierce	4.7	Meeting
4	1/6/2016 14:42	1/6/2016 15:49	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit
...	...	...	...	...	...	...	...
95	2/12/2016 8:21	2/12/2016 8:42	Business	Cary	Durham	8.5	Temporary Site
96	2/12/2016 10:45	2/12/2016 10:52	Business	Durham	Morrisville	2.6	Temporary Site
97	2/12/2016 11:14	2/12/2016 11:35	Business	Morrisville	Raleigh	17.0	Customer Visit
98	2/12/2016 13:02	2/12/2016 13:36	Business	Raleigh	Cary	18.0	Meeting
99	2/12/2016 14:49	2/12/2016 15:06	Business	Cary	Morrisville	8.4	Meeting

100 rows × 7 columns

In [13]:
```
ud.info()

#info()- similar to stypes and descibe
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1156 entries, 0 to 1155
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   START_DATE*  1156 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*       1156 non-null   float64
 6   PURPOSE*     653 non-null    object 
dtypes: float64(1), object(6)
memory usage: 63.3+ KB
```

In [14]:
```
uni_s=ud['START*'].unique()
print(uni_s)

#unique()- used to obtain the unique values in our column
```

```
['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'
 'Whitebridge' 'Lake Wellingborough' 'Fayetteville Street' 'Raleigh'
 'Hazelwood' 'Fairmont' 'Meredith Townes' 'Apex' 'Chapel Hill'
 'Northwoods' 'Edgehill Farms' 'Tanglewood' 'Preston' 'Eastgate'
 'East Elmhurst' 'Jackson Heights' 'Long Island City' 'Katunayaka'
 'Unknown Location' 'Colombo' 'Nugegoda' 'Islamabad' 'R?walpindi'
 'Noorpur Shahan' 'Heritage Pines' 'Westpark Place' 'Waverly Place'
 'Wayne Ridge' 'Weston' 'East Austin' 'West University' 'South Congress'
 'The Drag' 'Congress Ave District' 'Red River District' 'Georgian Acres'
 'North Austin' 'Coxville' 'Convention Center District' 'Austin' 'Katy'
 'Sharpstown' 'Sugar Land' 'Galveston' 'Port Bolivar' 'Washington Avenue'
 'Briar Meadow' 'Latta' 'Jacksonville' 'Couples Glen' 'Kissimmee'
 'Lake Reams' 'Orlando' 'Sand Lake Commons' 'Sky Lake' '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' 'Seaport'
 'Medical Centre' 'Rose Hill' 'Soho' 'Tribeca' 'Financial District'
 'Oakland' 'Emeryville' 'Berkeley' 'Kenner' 'CBD' 'Lower Garden District'
 'Lakeview' 'Storyville' 'New Orleans' 'Metairie' 'Chalmette' 'Arabi'
 'Pontchartrain Shores' 'Marigny' 'Covington' 'Mandeville'
 'Jamestown Court' 'Summerwinds' 'Parkwood' 'Pontchartrain Beach'
 'St Thomas' 'Banner Elk' 'Elk Park' 'Newland' 'Boone' 'Stonewater'
 'Lexington Park at Amberly' 'Arlington Park at Amberly' 'Arlington'
 'Kalorama Triangle' 'K Street' 'West End' 'Connecticut Avenue'
 'Columbia Heights' 'Washington' 'Wake Forest' 'Lahore' 'Karachi'
 'SOMISSPO' 'West Berkeley' 'North Berkeley Hills' 'San Jose' 'Eagle Rock'
 'Winston Salem' 'Asheville' 'Topton' 'Hayesville' 'Bryson City' 'Almond'
 'Mebane' 'Agnew' 'Cory' 'Renaissance' 'Santa Clara' 'NOMA' 'Sunnyside'
 'Ingleside' 'Central' 'Tenderloin' 'College Avenue' 'South' 'Southside'
 'South Berkeley' 'Mountain View' 'El Cerrito' 'Krendle Woods' 'Wake Co.'
 'Fuquay-Varina' 'Rawalpindi' 'Kar?chi' 'Katunayake' 'Gampaha' nan]
```

In [15]:
```
ud['START*'].value_counts()

#value_count= to get the count of element in our column
```

Out[15]:
```
START*
Cary                201
Unknown Location    148
Morrisville          85
Whitebridge          68
Islamabad            57
                   ...
Florence              1
Ridgeland             1
Daytona Beach         1
Sky Lake              1
Gampaha               1
Name: count, Length: 177, dtype: int64
```

In [16]:
```
miles_gt_50 =ud['MILES*']>50
print(miles_gt_50)

#to obtain the rows where miles value is greater than 50 -boolean form in result
```

```
0       False
1       False
2       False
3       False
4        True
        ...
1151    False
1152    False
1153    False
1154    False
1155     True
Name: MILES*, Length: 1156, dtype: bool
```

In [17]:
```
miles_gt50=ud[ud['MILES*']>50]
print(miles_gt50)

#to obtain the same but in list form instead of boolean
```

```
             START_DATE*          END_DATE* CATEGORY*            START*  \
4        1/6/2016 14:42    1/6/2016 15:49  Business       Fort Pierce   
232     3/17/2016 12:52   3/17/2016 15:11  Business            Austin   
251     3/19/2016 19:33   3/19/2016 20:39  Business         Galveston   
268     3/25/2016 13:24   3/25/2016 16:22  Business              Cary   
269     3/25/2016 16:52   3/25/2016 22:22  Business             Latta   
270     3/25/2016 22:54    3/26/2016 1:39  Business      Jacksonville   
295      4/2/2016 12:21    4/2/2016 14:47  Business         Kissimmee   
296      4/2/2016 16:57    4/2/2016 18:09  Business     Daytona Beach   
297      4/2/2016 19:38    4/2/2016 22:36  Business      Jacksonville   
298      4/2/2016 23:11     4/3/2016 1:34  Business         Ridgeland   
299       4/3/2016 2:00     4/3/2016 4:16  Business          Florence   
546     7/14/2016 16:39   7/14/2016 20:05  Business       Morrisville   
559     7/17/2016 12:20   7/17/2016 15:25  Personal             Boone   
707     8/24/2016 13:01   8/24/2016 15:25  Business  Unknown Location   
710     8/25/2016 17:19   8/25/2016 19:20  Business  Unknown Location   
726     8/27/2016 14:01   8/27/2016 15:44  Business            Lahore   
727     8/27/2016 16:15   8/27/2016 19:13  Business  Unknown Location   
751      9/6/2016 17:49    9/6/2016 17:49  Business  Unknown Location   
776     9/27/2016 21:01    9/28/2016 2:37  Business  Unknown Location   
788     10/6/2016 17:23   10/6/2016 17:40  Business        R?walpindi   
869    10/28/2016 15:53  10/28/2016 17:59  Business              Cary   
870    10/28/2016 18:13  10/28/2016 20:07  Business     Winston Salem   
871    10/28/2016 20:13  10/28/2016 22:00  Business         Asheville   
873    10/29/2016 17:13  10/29/2016 19:19  Business        Hayesville   
880    10/30/2016 13:24  10/30/2016 14:37  Business      Bryson City   
881    10/30/2016 15:22  10/30/2016 18:23  Business         Asheville   
1088   12/21/2016 20:56  12/21/2016 23:42  Business        Rawalpindi   
1155             Totals               NaN       NaN               NaN   

                  STOP*   MILES*         PURPOSE*  
4       West Palm Beach     63.7   Customer Visit  
232                Katy    136.0   Customer Visit  
251             Houston     57.0   Customer Visit  
268               Latta    144.0   Customer Visit  
269        Jacksonville    310.3   Customer Visit  
270           Kissimmee    201.0          Meeting  
295       Daytona Beach     77.3   Customer Visit  
296        Jacksonville     80.5   Customer Visit  
297           Ridgeland    174.2   Customer Visit  
298            Florence    144.0          Meeting  
299                Cary    159.3          Meeting  
546         Banner Elk    195.3              NaN  
559                Cary    180.2          Commute  
707    Unknown Location     96.2              NaN  
710    Unknown Location     50.4              NaN  
726    Unknown Location     86.6              NaN  
727    Unknown Location    156.9              NaN  
751    Unknown Location     69.1              NaN  
776    Unknown Location    195.6              NaN  
788    Unknown Location    112.6              NaN  
869       Winston Salem    107.0          Meeting  
870           Asheville    133.6          Meeting  
871              Topton     91.8          Meeting  
873              Topton     75.7              NaN  
880           Asheville     68.4              NaN  
881              Mebane    195.9              NaN  
1088   Unknown Location    103.0          Meeting  
1155                NaN  12204.7              NaN  
```

In [18]:
```python
miles_gt_50_v=ud[ud['MILES*']>50][['MILES*']]
print(miles_gt_50_v)

#obtaining only the miles column instead of all columns for which miles >50
```

```
        MILES*
4         63.7
232      136.0
251       57.0
268      144.0
269      310.3
270      201.0
295       77.3
296       80.5
297      174.2
298      144.0
299      159.3
546      195.3
559      180.2
707       96.2
710       50.4
726       86.6
727      156.9
751       69.1
776      195.6
788      112.6
869      107.0
870      133.6
871       91.8
873       75.7
880       68.4
881      195.9
1088     103.0
1155   12204.7
```

In [19]:

```
count_milesgt50=(ud['MILES*']>50).sum()
print(count_milesgt50)

#count of rows where miles >50
```

```
28
```

In [20]:

```
miles_bw50_100=ud[(ud['MILES*']>50) & (ud['MILES*']<100)]
print(miles_bw50_100)

#data where miles >50 and <100
```

```
            START_DATE*          END_DATE* CATEGORY*            START*  \
4      1/6/2016 14:42    1/6/2016 15:49  Business       Fort Pierce   
251   3/19/2016 19:33   3/19/2016 20:39  Business         Galveston   
295    4/2/2016 12:21    4/2/2016 14:47  Business         Kissimmee   
296    4/2/2016 16:57    4/2/2016 18:09  Business     Daytona Beach   
707   8/24/2016 13:01   8/24/2016 15:25  Business  Unknown Location   
710   8/25/2016 17:19   8/25/2016 19:20  Business  Unknown Location   
726   8/27/2016 14:01   8/27/2016 15:44  Business            Lahore   
751    9/6/2016 17:49    9/6/2016 17:49  Business  Unknown Location   
871  10/28/2016 20:13  10/28/2016 22:00  Business         Asheville   
873  10/29/2016 17:13  10/29/2016 19:19  Business        Hayesville   
880  10/30/2016 13:24  10/30/2016 14:37  Business       Bryson City   

                STOP*  MILES*         PURPOSE*  
4     West Palm Beach    63.7  Customer Visit  
251           Houston    57.0  Customer Visit  
295     Daytona Beach    77.3  Customer Visit  
296      Jacksonville    80.5  Customer Visit  
707  Unknown Location    96.2             NaN  
710  Unknown Location    50.4             NaN  
726  Unknown Location    86.6             NaN  
751  Unknown Location    69.1             NaN  
871            Topton    91.8         Meeting  
873            Topton    75.7             NaN  
880         Asheville    68.4             NaN  
```

In [21]:
```
miles_bw50_100=ud[(ud['MILES*']>50) & (ud['MILES*']<100)][['MILES*']]
print(miles_bw50_100)

#data where miles >50 and <100 but only miles column
```

```
     MILES*
4      63.7
251    57.0
295    77.3
296    80.5
707    96.2
710    50.4
726    86.6
751    69.1
871    91.8
873    75.7
880    68.4
```

In [22]:
```
miles_bw50_100.count()

#count of values in miles_bw50_100
```

Out[22]:
```
MILES*    11
dtype: int64
```

In [23]:
```
miles_bw50_100=ud[(ud['MILES*']>50) & (ud['MILES*']<100)][['MILES*','START*','STOP*
print(miles_bw50_100)

#data where miles >50 and <100 with specific column data
```

```
     MILES*            START*             STOP*
4      63.7       Fort Pierce   West Palm Beach
251    57.0         Galveston           Houston
295    77.3         Kissimmee     Daytona Beach
296    80.5     Daytona Beach      Jacksonville
707    96.2  Unknown Location  Unknown Location
710    50.4  Unknown Location  Unknown Location
726    86.6            Lahore  Unknown Location
751    69.1  Unknown Location  Unknown Location
871    91.8         Asheville            Topton
873    75.7       Hayesville            Topton
880    68.4      Bryson City         Asheville
```

In [24]:
```
START_NEWYORK=ud[ud['START*']=="New York"]
print(START_NEWYORK)

#to get data where start city is newyork
```

```
         START_DATE*         END_DATE* CATEGORY*    START*            STOP*  \
10   1/10/2016 15:08  1/10/2016 15:51  Business  New York           Queens
22   1/12/2016 16:02  1/12/2016 17:00  Business  New York    Queens County
106  2/14/2016 16:35  2/14/2016 17:02  Business  New York  Long Island City
423  6/10/2016 15:19  6/10/2016 16:28  Business  New York          Jamaica

     MILES* PURPOSE*
10     10.8  Meeting
22     15.1  Meeting
106    13.0  Meeting
423    16.3  Meeting
```

In [25]:
```
isin_F=ud.loc[ud['START*'].isin(['New York','Lahore']),'START*']
print(isin_F)

#isin function to check where our column contain the values given
```

```
10      New York
22      New York
106     New York
423     New York
712       Lahore
716       Lahore
717       Lahore
718       Lahore
720       Lahore
721       Lahore
722       Lahore
724       Lahore
725       Lahore
726       Lahore
774       Lahore
775       Lahore
792       Lahore
793       Lahore
795       Lahore
796       Lahore
797       Lahore
1096      Lahore
1097      Lahore
1098      Lahore
1099      Lahore
1103      Lahore
1105      Lahore
1106      Lahore
1107      Lahore
1108      Lahore
1109      Lahore
1110      Lahore
1111      Lahore
1112      Lahore
1113      Lahore
1114      Lahore
1115      Lahore
1116      Lahore
1117      Lahore
1118      Lahore
Name: START*, dtype: object
```

In [26]:
```
cond = ud.loc[ (ud['START*'].isin(['New York', 'Jamaica ',   'Downtown'])) &
              (ud['STOP*'].isin(['New York', 'Queens', 'Gulfton'])) &
              (ud['MILES*'] > 10) & (ud['MILES*'] < 20) ]
print(cond)

#to display record whose start city is say A,B,C and stop city is say X,Y,Z and mil
```

```
        START_DATE*         END_DATE* CATEGORY*    START*    STOP*  MILES*  \
10  1/10/2016 15:08  1/10/2016 15:51  Business  New York   Queens    10.8   
23  1/13/2016 13:54  1/13/2016 14:07  Business  Downtown  Gulfton    11.2   

   PURPOSE*  
10  Meeting  
23  Meeting  
```

In [27]:
```
cond_2 = ud.loc[ (ud['START*'].isin(['New York', 'Jamaica ',   'Downtown'])) &
              (ud['STOP*'].isin(['New York', 'Queens', 'Gulfton'])) &
              (ud['MILES*'] > 10) & (ud['MILES*'] < 20) ,['START*','STOP*','MILES*'
print(cond_2)

#to display recordof start ,stop and miles  whose start city is say A,B,C and stop
```

```
       START*    STOP*  MILES*
10  New York   Queens    10.8
23  Downtown  Gulfton    11.2
```

In [28]:
```
ud.dtypes
```

Out[28]:
```
START_DATE*     object
END_DATE*       object
CATEGORY*       object
START*          object
STOP*           object
MILES*         float64
PURPOSE*        object
dtype: object
```

In [54]:
```
ud['START_DATE*'] = pd.to_datetime(ud['START_DATE*'], errors='coerce')
```

In [55]:
```
ud['END_DATE*'] = pd.to_datetime(ud['END_DATE*'])
```

In [56]:
```
ud.dtypes
```

Out[56]:
```
START_DATE*    datetime64[ns]
END_DATE*      datetime64[ns]
CATEGORY*              object
START*                 object
STOP*                  object
MILES*                float64
PURPOSE*               object
MILES_CAT              object
dtype: object
```

In [57]:
```
#to print the sales of january 2016

cond3 = ud.loc[ (ud['START_DATE*'] >= '2016-01-01') & (ud['END_DATE*'] <= '2016-01-
print(cond3)
```

```
           START_DATE*           END_DATE* CATEGORY*       START*  \
0  2016-01-01 21:11:00 2016-01-01 21:17:00  Business  Fort Pierce
1  2016-01-02 01:25:00 2016-01-02 01:37:00  Business  Fort Pierce
2  2016-01-02 20:25:00 2016-01-02 20:38:00  Business  Fort Pierce
3  2016-01-05 17:31:00 2016-01-05 17:45:00  Business  Fort Pierce
4  2016-01-06 14:42:00 2016-01-06 15:49:00  Business  Fort Pierce
..                 ...                 ...       ...          ...
56 2016-01-29 13:24:00 2016-01-29 13:47:00  Business       Durham
57 2016-01-29 18:31:00 2016-01-29 18:52:00  Business         Cary
58 2016-01-29 21:21:00 2016-01-29 21:40:00  Business         Apex
59 2016-01-30 16:21:00 2016-01-30 16:33:00  Business         Cary
60 2016-01-30 18:09:00 2016-01-30 18:24:00  Business         Apex

              STOP*  MILES*         PURPOSE*   MILES_CAT
0       Fort Pierce     5.1   Meal/Entertain  short trip
1       Fort Pierce     5.0              NaN  short trip
2       Fort Pierce     4.8  Errand/Supplies  short trip
3       Fort Pierce     4.7          Meeting  short trip
4   West Palm Beach    63.7   Customer Visit  short trip
..              ...     ...              ...         ...
56             Cary    10.1          Meeting  short trip
57             Apex     5.8  Errand/Supplies  short trip
58             Cary     5.5   Meal/Entertain  short trip
59             Apex     5.7  Errand/Supplies  short trip
60             Cary     5.7   Customer Visit  short trip

[61 rows x 8 columns]
```

In [58]:
```
#sales in jan 2016 and start city is cary
cond4 = ud.loc[ (ud['START_DATE*'] >= '2016-01-01') & (ud['END_DATE*'] <= '2016-01-
print(cond4)
print(cond4.count())
```

```
            START_DATE*            END_DATE* CATEGORY* START*        STOP*  \
7  2016-01-07 13:27:00 2016-01-07 13:33:00  Business   Cary         Cary
8  2016-01-10 08:05:00 2016-01-10 08:25:00  Business   Cary  Morrisville
28 2016-01-15 11:43:00 2016-01-15 12:03:00  Business   Cary       Durham
30 2016-01-18 14:55:00 2016-01-18 15:06:00  Business   Cary         Cary
34 2016-01-20 10:36:00 2016-01-20 11:11:00  Business   Cary      Raleigh
37 2016-01-21 14:25:00 2016-01-21 14:29:00  Business   Cary         Cary
38 2016-01-21 14:43:00 2016-01-21 14:51:00  Business   Cary         Cary
39 2016-01-21 16:01:00 2016-01-21 16:06:00  Business   Cary         Cary
43 2016-01-26 17:17:00 2016-01-26 17:22:00  Business   Cary         Cary
44 2016-01-26 17:27:00 2016-01-26 17:29:00  Business   Cary         Cary
45 2016-01-27 09:24:00 2016-01-27 09:31:00  Business   Cary         Cary
46 2016-01-27 10:19:00 2016-01-27 10:48:00  Business   Cary      Raleigh
50 2016-01-28 12:28:00 2016-01-28 13:00:00  Business   Cary      Raleigh
53 2016-01-29 09:31:00 2016-01-29 09:45:00  Business   Cary         Cary
54 2016-01-29 10:56:00 2016-01-29 11:07:00  Business   Cary         Cary
55 2016-01-29 11:43:00 2016-01-29 12:03:00  Business   Cary       Durham
57 2016-01-29 18:31:00 2016-01-29 18:52:00  Business   Cary         Apex
59 2016-01-30 16:21:00 2016-01-30 16:33:00  Business   Cary         Apex

    MILES*         PURPOSE*   MILES_CAT
7      0.8          Meeting  short trip
8      8.3          Meeting  short trip
28    10.4   Meal/Entertain  short trip
30     4.8   Meal/Entertain  short trip
34    17.1          Meeting  short trip
37     1.6  Errand/Supplies  short trip
38     2.4   Meal/Entertain  short trip
39     1.0   Meal/Entertain  short trip
43     1.4  Errand/Supplies  short trip
44     0.5  Errand/Supplies  short trip
45     1.8          Meeting  short trip
46    18.7   Customer Visit  short trip
50    19.0   Temporary Site  short trip
53     4.6   Customer Visit  short trip
54     5.2          Meeting  short trip
55    10.4          Meeting  short trip
57     5.8  Errand/Supplies  short trip
59     5.7  Errand/Supplies  short trip
START_DATE*    18
END_DATE*      18
CATEGORY*      18
START*         18
STOP*          18
MILES*         18
PURPOSE*       18
MILES_CAT      18
dtype: int64
```

In [59]:
```
step_con = ud.loc[(ud['START_DATE*'] >= '2016-01-01') & (ud['END_DATE*'] <= '2016-0
print(step_con)

#asigning step size using iloc
```

```
             START_DATE*           END_DATE* CATEGORY*               START*  \
1  2016-01-02 01:25:00 2016-01-02 01:37:00  Business          Fort Pierce   
3  2016-01-05 17:31:00 2016-01-05 17:45:00  Business          Fort Pierce   
5  2016-01-06 17:15:00 2016-01-06 17:19:00  Business      West Palm Beach   
7  2016-01-07 13:27:00 2016-01-07 13:33:00  Business                 Cary   
9  2016-01-10 12:17:00 2016-01-10 12:44:00  Business              Jamaica   
11 2016-01-10 18:18:00 2016-01-10 18:53:00  Business             Elmhurst   
13 2016-01-11 08:55:00 2016-01-11 09:21:00  Business          East Harlem   
15 2016-01-11 13:32:00 2016-01-11 13:46:00  Business              Midtown   
17 2016-01-12 12:33:00 2016-01-12 12:49:00  Business              Midtown   
19 2016-01-12 14:42:00 2016-01-12 14:56:00  Business      Lower Manhattan   
21 2016-01-12 15:42:00 2016-01-12 15:54:00  Business       Hell's Kitchen   
23 2016-01-13 13:54:00 2016-01-13 14:07:00  Business             Downtown   
25 2016-01-14 16:29:00 2016-01-14 17:05:00  Business              Houston   
27 2016-01-15 00:41:00 2016-01-15 01:01:00  Business          Morrisville   
29 2016-01-15 13:26:00 2016-01-15 13:44:00  Business               Durham   
31 2016-01-18 16:13:00 2016-01-18 16:24:00  Business     Farmington Woods   
33 2016-01-19 10:55:00 2016-01-19 11:09:00  Business  Lake Wellingborough   
35 2016-01-20 11:48:00 2016-01-20 12:19:00  Business  Fayetteville Street   
37 2016-01-21 14:25:00 2016-01-21 14:29:00  Business                 Cary   
39 2016-01-21 16:01:00 2016-01-21 16:06:00  Business                 Cary   
41 2016-01-26 12:33:00 2016-01-26 12:41:00  Business            Hazelwood   
43 2016-01-26 17:17:00 2016-01-26 17:22:00  Business                 Cary   
45 2016-01-27 09:24:00 2016-01-27 09:31:00  Business                 Cary   
47 2016-01-27 12:34:00 2016-01-27 12:44:00  Business             Fairmont   
49 2016-01-27 14:46:00 2016-01-27 15:08:00  Business              Raleigh   
51 2016-01-28 15:11:00 2016-01-28 15:31:00  Business      Meredith Townes   
53 2016-01-29 09:31:00 2016-01-29 09:45:00  Business                 Cary   
55 2016-01-29 11:43:00 2016-01-29 12:03:00  Business                 Cary   
57 2016-01-29 18:31:00 2016-01-29 18:52:00  Business                 Cary   
59 2016-01-30 16:21:00 2016-01-30 16:33:00  Business                 Cary   

                STOP*  MILES*         PURPOSE*   MILES_CAT  
1         Fort Pierce     5.0              NaN  short trip  
3         Fort Pierce     4.7          Meeting  short trip  
5     West Palm Beach     4.3   Meal/Entertain  short trip  
7                Cary     0.8          Meeting  short trip  
9            New York    16.5   Customer Visit  short trip  
11           New York     7.5          Meeting  short trip  
13              NoMad     6.4   Temporary Site  short trip  
15       Midtown East     1.7   Meal/Entertain  short trip  
17      Hudson Square     1.9   Meal/Entertain  short trip  
19      Hudson Square     1.8  Errand/Supplies  short trip  
21            Midtown     2.0  Errand/Supplies  short trip  
23            Gulfton    11.2          Meeting  short trip  
25            Houston    21.9   Customer Visit  short trip  
27               Cary     8.0  Errand/Supplies  short trip  
29               Cary    10.4   Meal/Entertain  short trip  
31        Whitebridge     4.7   Meal/Entertain  short trip  
33        Whitebridge     7.6   Temporary Site  short trip  
35            Umstead    15.1          Meeting  short trip  
37               Cary     1.6  Errand/Supplies  short trip  
39               Cary     1.0   Meal/Entertain  short trip  
41        Whitebridge     2.3  Errand/Supplies  short trip  
43               Cary     1.4  Errand/Supplies  short trip  
45               Cary     1.8          Meeting  short trip  
47    Meredith Townes     3.4   Customer Visit  short trip  
49               Cary    12.9   Customer Visit  short trip  
51  Leesville Hollow     14.7          Meeting  short trip  
53               Cary     4.6   Customer Visit  short trip  
55             Durham    10.4          Meeting  short trip  
57               Apex     5.8  Errand/Supplies  short trip  
59               Apex     5.7  Errand/Supplies  short trip  
```

In [60]:
```
df = ud.loc[(ud['START_DATE*'] >= '2016-01-01') & (ud['END_DATE*'] <= '2016-01-31')
df.reset_index(inplace=True, drop=False)
print(df)

#drop=false index is present
```

```
    index          START_DATE*            END_DATE* CATEGORY*       START*  \
0       0 2016-01-01 21:11:00 2016-01-01 21:17:00  Business  Fort Pierce   
1       1 2016-01-02 01:25:00 2016-01-02 01:37:00  Business  Fort Pierce   
2       2 2016-01-02 20:25:00 2016-01-02 20:38:00  Business  Fort Pierce   
3       3 2016-01-05 17:31:00 2016-01-05 17:45:00  Business  Fort Pierce   
4       4 2016-01-06 14:42:00 2016-01-06 15:49:00  Business  Fort Pierce   
..    ...                 ...                 ...       ...          ...   
56     56 2016-01-29 13:24:00 2016-01-29 13:47:00  Business       Durham   
57     57 2016-01-29 18:31:00 2016-01-29 18:52:00  Business         Cary   
58     58 2016-01-29 21:21:00 2016-01-29 21:40:00  Business         Apex   
59     59 2016-01-30 16:21:00 2016-01-30 16:33:00  Business         Cary   
60     60 2016-01-30 18:09:00 2016-01-30 18:24:00  Business         Apex   

              STOP*  MILES*         PURPOSE*   MILES_CAT  
0       Fort Pierce     5.1   Meal/Entertain  short trip  
1       Fort Pierce     5.0              NaN  short trip  
2       Fort Pierce     4.8  Errand/Supplies  short trip  
3       Fort Pierce     4.7          Meeting  short trip  
4   West Palm Beach    63.7   Customer Visit  short trip  
..              ...     ...              ...         ...  
56             Cary    10.1          Meeting  short trip  
57             Apex     5.8  Errand/Supplies  short trip  
58             Cary     5.5   Meal/Entertain  short trip  
59             Apex     5.7  Errand/Supplies  short trip  
60             Cary     5.7   Customer Visit  short trip  

[61 rows x 9 columns]
```

In [61]:
```
df1 = ud.loc[(ud['START_DATE*'] >= '2016-01-01') & (ud['END_DATE*'] <= '2016-01-31'
df1.reset_index(inplace=True, drop=True)
print(df1)

#drop=true index is removed
```

```
            START_DATE*           END_DATE* CATEGORY*       START*  \
0  2016-01-01 21:11:00 2016-01-01 21:17:00  Business  Fort Pierce   
1  2016-01-02 01:25:00 2016-01-02 01:37:00  Business  Fort Pierce   
2  2016-01-02 20:25:00 2016-01-02 20:38:00  Business  Fort Pierce   
3  2016-01-05 17:31:00 2016-01-05 17:45:00  Business  Fort Pierce   
4  2016-01-06 14:42:00 2016-01-06 15:49:00  Business  Fort Pierce   
..                 ...                 ...       ...          ...   
56 2016-01-29 13:24:00 2016-01-29 13:47:00  Business       Durham   
57 2016-01-29 18:31:00 2016-01-29 18:52:00  Business         Cary   
58 2016-01-29 21:21:00 2016-01-29 21:40:00  Business         Apex   
59 2016-01-30 16:21:00 2016-01-30 16:33:00  Business         Cary   
60 2016-01-30 18:09:00 2016-01-30 18:24:00  Business         Apex   

              STOP*  MILES*         PURPOSE*   MILES_CAT  
0       Fort Pierce     5.1   Meal/Entertain  short trip  
1       Fort Pierce     5.0              NaN  short trip  
2       Fort Pierce     4.8  Errand/Supplies  short trip  
3       Fort Pierce     4.7          Meeting  short trip  
4   West Palm Beach    63.7   Customer Visit  short trip  
..              ...     ...              ...         ...  
56             Cary    10.1          Meeting  short trip  
57             Apex     5.8  Errand/Supplies  short trip  
58             Cary     5.5   Meal/Entertain  short trip  
59             Apex     5.7  Errand/Supplies  short trip  
60             Cary     5.7   Customer Visit  short trip  

[61 rows x 8 columns]
```

In [62]:
```
ud.sort_values(by='MILES*')

#sort_values used to sort data based on a column values
```

Out[62]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*	M
420	2016-06-08 17:16:00	2016-06-08 17:18:00	Business	Soho	Tribeca	0.5	Errand/Supplies	
44	2016-01-26 17:27:00	2016-01-26 17:29:00	Business	Cary	Cary	0.5	Errand/Supplies	
120	2016-02-17 16:38:00	2016-02-17 16:43:00	Business	Katunayaka	Katunayaka	0.5	Errand/Supplies	
1111	2016-12-25 00:10:00	2016-12-25 00:14:00	Business	Lahore	Lahore	0.6	Errand/Supplies	
1110	2016-12-24 22:04:00	2016-12-24 22:09:00	Business	Lahore	Lahore	0.6	Errand/Supplies	
...	...	...	...	...	...	...	...	
776	2016-09-27 21:01:00	2016-09-28 02:37:00	Business	Unknown Location	Unknown Location	195.6	NaN	
881	2016-10-30 15:22:00	2016-10-30 18:23:00	Business	Asheville	Mebane	195.9	NaN	
270	2016-03-25 22:54:00	2016-03-26 01:39:00	Business	Jacksonville	Kissimmee	201.0	Meeting	
269	2016-03-25 16:52:00	2016-03-25 22:22:00	Business	Latta	Jacksonville	310.3	Customer Visit	
1155	NaT	NaT	NaN	NaN	NaN	12204.7	NaN	

1156 rows × 8 columns

In [63]:

```
ud.sort_values(by='MILES*',ascending=False)

#ascending fasle=rsults in descending order
```

Out[63]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*	M
1155	NaT	NaT	NaN	NaN	NaN	12204.7	NaN	
269	2016-03-25 16:52:00	2016-03-25 22:22:00	Business	Latta	Jacksonville	310.3	Customer Visit	
270	2016-03-25 22:54:00	2016-03-26 01:39:00	Business	Jacksonville	Kissimmee	201.0	Meeting	
881	2016-10-30 15:22:00	2016-10-30 18:23:00	Business	Asheville	Mebane	195.9	NaN	
776	2016-09-27 21:01:00	2016-09-28 02:37:00	Business	Unknown Location	Unknown Location	195.6	NaN	
...	...	...	...	...	...	...	...	
1121	2016-12-27 12:53:00	2016-12-27 12:57:00	Business	Kar?chi	Kar?chi	0.6	Meal/Entertain	
1110	2016-12-24 22:04:00	2016-12-24 22:09:00	Business	Lahore	Lahore	0.6	Errand/Supplies	
44	2016-01-26 17:27:00	2016-01-26 17:29:00	Business	Cary	Cary	0.5	Errand/Supplies	
420	2016-06-08 17:16:00	2016-06-08 17:18:00	Business	Soho	Tribeca	0.5	Errand/Supplies	
120	2016-02-17 16:38:00	2016-02-17 16:43:00	Business	Katunayaka	Katunayaka	0.5	Errand/Supplies	

1156 rows × 8 columns

In [64]:

```
df2 = ud.sort_values(by=["START*", "STOP*"], ascending=[True, False])
print(df2)

#sorting data in increasing order of start anf descreasing order of stop
```

```
                START_DATE*           END_DATE*  CATEGORY*         START*  \
908  2016-11-05 08:34:00 2016-11-05 08:43:00  Business         Agnew   
911  2016-11-06 10:50:00 2016-11-06 11:04:00  Business         Agnew   
906  2016-11-04 21:04:00 2016-11-04 21:20:00  Business         Agnew   
910  2016-11-05 19:20:00 2016-11-05 19:28:00  Business         Agnew   
879  2016-10-30 12:58:00 2016-10-30 13:18:00  Business        Almond   
...                  ...                 ...       ...           ...   
572  2016-07-19 17:14:00 2016-07-19 17:24:00  Business   Whitebridge   
332  2016-04-27 13:30:00 2016-04-27 13:40:00  Business   Whitebridge   
612  2016-08-01 12:47:00 2016-08-01 13:04:00  Business   Whitebridge   
870  2016-10-28 18:13:00 2016-10-28 20:07:00  Business  Winston Salem   
1155                 NaT                 NaT       NaN           NaN   

                          STOP*   MILES*         PURPOSE*   MILES_CAT  
908                Renaissance      2.2              NaN  short trip  
911                Renaissance      2.4              NaN  short trip  
906                       Cory      4.3              NaN  short trip  
910                      Agnew      2.2              NaN  short trip  
879                Bryson City     15.2              NaN  short trip  
...                        ...      ...              ...         ...  
572                Chessington      3.9  Errand/Supplies  short trip  
332                   Burtrose      4.9  Between Offices  short trip  
612     Arlington Park at Amberly      6.2              NaN  short trip  
870                 Asheville    133.6          Meeting   Long trip  
1155                       NaN  12204.7              NaN   Long trip  

[1156 rows x 8 columns]
```

In [65]:
```python
import numpy as np
ud.loc[:,'MILES_CAT']=np.where(ud['MILES*']>100,'Long trip','short trip')
ud.head()

#categorizing as long and short trips based on miles
```

Out[65]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*	MILES_CAT
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	short trip
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	NaN	short trip
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	short trip
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	short trip
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	short trip

In [66]:
```python
ud['nc']=10
ud
```

Out[66]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*	MILI
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	sh
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	NaN	sh
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	sh
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	sh
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	sh
...	...	...	...	...	...	...	...	
1151	2016-12-31 13:24:00	2016-12-31 13:42:00	Business	Kar?chi	Unknown Location	3.9	Temporary Site	sh
1152	2016-12-31 15:03:00	2016-12-31 15:38:00	Business	Unknown Location	Unknown Location	16.2	Meeting	sh
1153	2016-12-31 21:32:00	2016-12-31 21:50:00	Business	Katunayake	Gampaha	6.4	Temporary Site	sh
1154	2016-12-31 22:08:00	2016-12-31 23:51:00	Business	Gampaha	Ilukwatta	48.2	Temporary Site	sh
1155	NaT	NaT	NaN	NaN	NaN	12204.7	NaN	Lc

1156 rows × 9 columns

In [76]:

```python
ud['Trip']=np.where(ud['MILES*']<=100,
                    "Short trips",
                    np.where(ud['MILES*']<=200,"Medium Trip","Long Trip"))
ud

#categorizing as long,medium and short trips based on miles
```

Out[76]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*	MILI
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	sh
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	NaN	sh
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	sh
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	sh
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	sh
...	...	...	...	...	...	...	...	
1151	2016-12-31 13:24:00	2016-12-31 13:42:00	Business	Kar?chi	Unknown Location	3.9	Temporary Site	sh
1152	2016-12-31 15:03:00	2016-12-31 15:38:00	Business	Unknown Location	Unknown Location	16.2	Meeting	sh
1153	2016-12-31 21:32:00	2016-12-31 21:50:00	Business	Katunayake	Gampaha	6.4	Temporary Site	sh
1154	2016-12-31 22:08:00	2016-12-31 23:51:00	Business	Gampaha	Ilukwatta	48.2	Temporary Site	sh
1155	NaT	NaT	NaN	NaN	NaN	12204.7	NaN	Lc

1156 rows × 10 columns

In [77]:

```
Short_trips=ud[ud['Trip']=="Short trips"]
Short_trips

#finding number of short trips
```

Out[77]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*	MILE
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	sh
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	NaN	sh
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	sh
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	sh
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	sh
...	...	...	...	...	...	...	...	
1150	2016-12-31 01:07:00	2016-12-31 01:14:00	Business	Kar?chi	Kar?chi	0.7	Meeting	sh
1151	2016-12-31 13:24:00	2016-12-31 13:42:00	Business	Kar?chi	Unknown Location	3.9	Temporary Site	sh
1152	2016-12-31 15:03:00	2016-12-31 15:38:00	Business	Unknown Location	Unknown Location	16.2	Meeting	sh
1153	2016-12-31 21:32:00	2016-12-31 21:50:00	Business	Katunayake	Gampaha	6.4	Temporary Site	sh
1154	2016-12-31 22:08:00	2016-12-31 23:51:00	Business	Gampaha	Ilukwatta	48.2	Temporary Site	sh

1139 rows × 10 columns

In [78]:
```python
Long_trips=ud[ud['Trip']=="Long Trip"]
Long_trips

#finding number of long trips
```

Out[78]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*	MILES_
269	2016-03-25 16:52:00	2016-03-25 22:22:00	Business	Latta	Jacksonville	310.3	Customer Visit	Long
270	2016-03-25 22:54:00	2016-03-26 01:39:00	Business	Jacksonville	Kissimmee	201.0	Meeting	Long
1155	NaT	NaT	NaN	NaN	NaN	12204.7	NaN	Long

In [82]:
```python
Med_trips=ud[ud['Trip']=="Medium Trip"]
Med_trips

#finding number of medium trips
```

Out[82]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*	MILES_C
232	2016-03-17 12:52:00	2016-03-17 15:11:00	Business	Austin	Katy	136.0	Customer Visit	Long t
268	2016-03-25 13:24:00	2016-03-25 16:22:00	Business	Cary	Latta	144.0	Customer Visit	Long t
297	2016-04-02 19:38:00	2016-04-02 22:36:00	Business	Jacksonville	Ridgeland	174.2	Customer Visit	Long t
298	2016-04-02 23:11:00	2016-04-03 01:34:00	Business	Ridgeland	Florence	144.0	Meeting	Long t
299	2016-04-03 02:00:00	2016-04-03 04:16:00	Business	Florence	Cary	159.3	Meeting	Long t
546	2016-07-14 16:39:00	2016-07-14 20:05:00	Business	Morrisville	Banner Elk	195.3	NaN	Long t
559	2016-07-17 12:20:00	2016-07-17 15:25:00	Personal	Boone	Cary	180.2	Commute	Long t
727	2016-08-27 16:15:00	2016-08-27 19:13:00	Business	Unknown Location	Unknown Location	156.9	NaN	Long t
776	2016-09-27 21:01:00	2016-09-28 02:37:00	Business	Unknown Location	Unknown Location	195.6	NaN	Long t
788	2016-10-06 17:23:00	2016-10-06 17:40:00	Business	R?walpindi	Unknown Location	112.6	NaN	Long t
869	2016-10-28 15:53:00	2016-10-28 17:59:00	Business	Cary	Winston Salem	107.0	Meeting	Long t
870	2016-10-28 18:13:00	2016-10-28 20:07:00	Business	Winston Salem	Asheville	133.6	Meeting	Long t
881	2016-10-30 15:22:00	2016-10-30 18:23:00	Business	Asheville	Mebane	195.9	NaN	Long t
1088	2016-12-21 20:56:00	2016-12-21 23:42:00	Business	Rawalpindi	Unknown Location	103.0	Meeting	Long t

In [83]:
```
Med_trips.count()
```

Out[83]:
```
START_DATE*     14
END_DATE*       14
CATEGORY*       14
START*          14
STOP*           14
MILES*          14
PURPOSE*         9
MILES_CAT       14
nc              14
Trip            14
dtype: int64
```

In [85]:
```
a=ud['Trip'].value_counts()
a

#count of no trips and there types
```

Out[85]:
```
Trip
Short trips    1139
Medium Trip      14
Long Trip         3
Name: count, dtype: int64
```

In [87]:
```
ud.groupby('START*')['MILES*'].agg('mean')

#groub by cluse for start and avg of miles value
```

Out[87]:
```
START*
Agnew                   2.775000
Almond                 15.200000
Apex                    5.341176
Arabi                  17.000000
Arlington               4.900000
                         ...
West University         2.200000
Weston                  4.000000
Westpark Place          2.182353
Whitebridge             4.020588
Winston Salem         133.600000
Name: MILES*, Length: 177, dtype: float64
```

In [94]:
```
#average miles of each purpose

grouped = ud.groupby('CATEGORY*')['MILES*'].agg(['sum', 'mean', 'max'])
grouped
```

Out[94]:

CATEGORY*	sum	mean	max
Business	11487.0	10.655844	310.3
Personal	717.7	9.320779	180.2

In [95]:
```
ud.groupby('START*')['MILES*'].agg(['sum', 'mean', 'max'])
```

Out[95]:

START*	sum	mean	max
Agnew	11.1	2.775000	4.3
Almond	15.2	15.200000	15.2
Apex	90.8	5.341176	9.0
Arabi	17.0	17.000000	17.0
Arlington	4.9	4.900000	4.9
...	...	...	...
West University	4.4	2.200000	2.3
Weston	8.0	4.000000	4.2
Westpark Place	37.1	2.182353	4.2
Whitebridge	273.4	4.020588	9.0
Winston Salem	133.6	133.600000	133.6

177 rows × 3 columns

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [26]:

```
temp=pd.DataFrame({
    'A':[1,2,3,4],
    'B':[10,20,30,40],
    'C':['2023-1-29','2025-1-21','2025-1-11','2025-1-22']
})

#to create a dataframe
```

In [27]:

```
temp.info()

# to display info about the data frame
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   A       4 non-null      int64
 1   B       4 non-null      int64
 2   C       4 non-null      object
dtypes: int64(2), object(1)
memory usage: 228.0+ bytes
```

In [28]:
```
print(temp)
```

```
   A   B          C
0  1  10  2023-1-29
1  2  20  2025-1-21
2  3  30  2025-1-11
3  4  40  2025-1-22
```

In [30]:
```
temp['C']=pd.to_datetime(temp['C'])

#converting the object data type to datetime
```

In [31]:
```
temp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   A       4 non-null      int64
 1   B       4 non-null      int64
 2   C       4 non-null      datetime64[ns]
dtypes: datetime64[ns](1), int64(2)
memory usage: 228.0 bytes
```

In [49]:
```
temp['C']=pd.to_datetime(temp['C'],format="%d-%m-%Y")
```

In [50]:
```
temp.dtypes
```

Out[50]:
```
A             int64
B             int64
C    datetime64[ns]
dtype: object
```

In [48]:
```
print(temp)
```

```
   A   B          C
0  1  10 2023-01-29
1  2  20 2025-01-21
2  3  30 2025-01-11
3  4  40 2025-01-22
```

In [56]:
```
temp['A'] = temp['A'].astype(str)

#converting int to str which results in object datatype
```

In [57]:
```
temp.dtypes
```

Out[57]:
```
A            object
B             int64
C    datetime64[ns]
dtype: object
```

```
In [ ]:
```