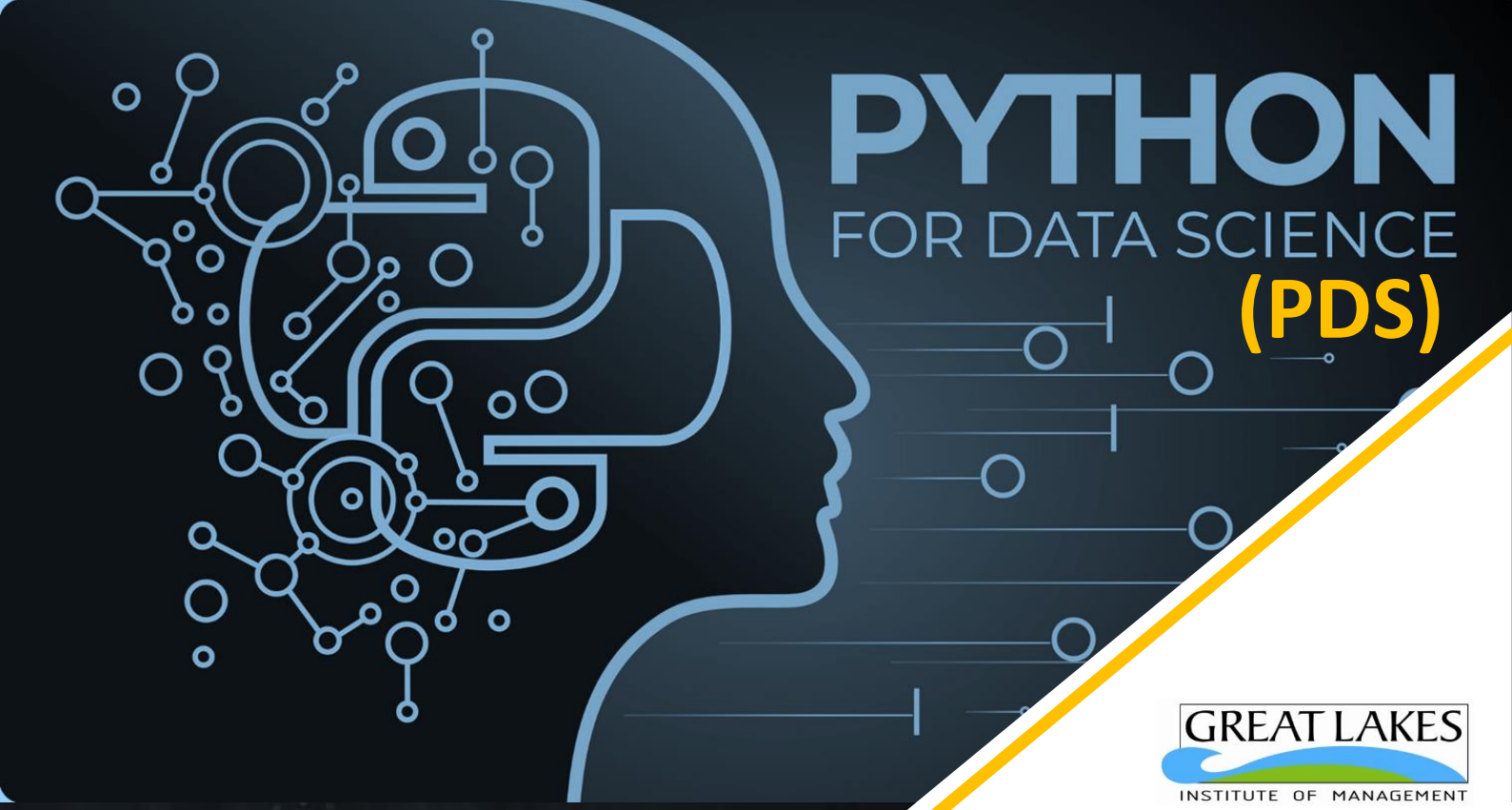


PYTHON
FOR DATA SCIENCE
(PDS)

GREAT LAKES
INSTITUTE OF MANAGEMENT

Group Assignment

greatlearning
Learning for Life

Team Name: PDS Group1

Members:

1. Rekha Chankramath
2. Nikhil Panchal
3. Rajvardhan
4. Nandita Ray
5. Abhijeet Singh

Oct 31st, 2020

About Us

Before we move on to the PDS group assignment solution, we will cover brief introduction about us.
We are seasoned professionals with diverse experience.

Rekha

Overall **20+ years of corporate and management experience**. Currently (from 2018) working as **Strategy Consultant (Independently)** for firms and companies that range from hospitality, manufacturing and retail industries. Before moving to an independent role was with **KPMG Global Services for 12 years**. Hands-on manager with expertise in accounting **systems development, fiscal management and financial reporting**. Proven record of developing and implementing financial and operational controls that **improve P&L scenario and competitively position firm**.

Nikhil

Overall **13+ years of cross functional experience** in FMCG and Alcobev industry. Working with **Diageo PLC** since 2017, looking after **Data Analytics CoE for Global Audit & Risk department**. Before to that worked with **Coca-Cola for 10 years** and completed stint in operations, supply chain (direct & indirect), Master data maintenance (SAP techno functional role) and Data analytics (Internal Audit).

Rajvardhan

Overall **6+ years of experience in IT industry** worked in **financial and aerospace domain**. Working **with Infosys since June 2018 as a technology analyst** before that was working in **HCL technologies**.

Nandita

Overall **7+ years of experience** in **Online marketing**. At present working with **VMware to optimize the Knowledge Base website to enable customers to self-solve the software related issues**. Dealing with lots of Customer data and looking forward to insightfully deal with the data. Also, as a **Citizen Philanthropist working with many NGOs** to utilise my skills towards betterment of the community.

Abhijeet

Overall **5+ years of experience in software Java development**. Currently working with **CGI under Insurance project**. Before joining CGI, worked for **TCS for 4 years** in software development using **Java and web services**.

Load the necessary libraries. Import and load the dataset with a name uber_drives .

In [1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [2]:
```
sns.set()
```

In [3]:
```
class color: ## to define output or conclusion in below format or color.
   PURPLE = '\033[95m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[92m'
   YELLOW = '\033[93m'
   RED = '\033[91m'
   BOLD = '\033[1m'
   UNDERLINE = '\033[4m'
   END = '\033[0m'
```

In [4]:
```
# Reading the data set
uber_drives=pd.read_csv('uberdrive-2.csv')
```

Q1. Show the last 10 records of the dataset. (2 point)

In [5]:
```
# This function helps to display last 10 records of the dataset.
uber_drives.tail(10)
```

Out[5]:

	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/Supplies
1146	12/30/2016 11:31	12/30/2016 11:56	Business	Karachi	Karachi	2.9	Errand/Supplies
1147	12/30/2016 15:41	12/30/2016 16:03	Business	Karachi	Karachi	4.6	Errand/Supplies
1148	12/30/2016 16:45	12/30/2016 17:08	Business	Karachi	Karachi	4.6	Meeting
1149	12/30/2016 23:06	12/30/2016 23:10	Business	Karachi	Karachi	0.8	Customer Visit
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

---

Q2. Show the first 10 records of the dataset. (2 points)

In [6]:
```
# This function helps to display first 10 records of the dataset.
uber_drives.head(10)
```

Out[6]:

	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. (2 points)

In [7]:
```
# This function displays the dimension of the dataset.
uber_drives.shape
```

Out[7]: (1155, 7)

In [8]:
```
print(color.BOLD + color.PURPLE + 'In this dataset number of rows are    : ', uber_drives.shape[0], color.END)
print("")
print(color.BOLD + color.PURPLE + 'In this dataset number of columns are : ', uber_drives.shape[1], color.END)
```

```
In this dataset number of rows are    :  1155

In this dataset number of columns are :  7
```

Q4. Show the size (Total number of elements) of the dataset. (2 points)

In [9]:
```
## The total number of elements of dataset is stored in the size attribute which is equal to the row count * column co
unt.
uber_drives.size
```

Out[9]: 8085

In [10]:
```
print(color.BOLD + color.PURPLE + 'In this dataset total number of element are : ', uber_drives.size, color.END)
```

```
In this dataset total number of element are :  8085
```

Q5. Display the information about all the variables of the data set. (2 points)

Hint: Information includes - Total number of columns,variable data-types, number of non-null values in a variable, and usage

In [11]:
```
## This function helps to to get a concise summary of the dataset
## It comes really handy when doing exploratory analysis of the data.
uber_drives.info()
```

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

Q6. Check for missing values. (2 points) - Note: Output should be boolean only.

In [12]:
```
## This function helps to get to know which column from the dataset has the null value.
uber_drives.isnull().any()
```

Out[12]:
```
START_DATE*    False
END_DATE*      False
CATEGORY*      False
START*         False
STOP*          False
MILES*         False
PURPOSE*        True
dtype: bool
```

Here we can see that "PURPOSE" field in the dataset has the missing values.

In [13]:
```
## This function helps to display the missing value and non-missing value row-wise in boolean.
uber_drives['PURPOSE*'].isnull()
```

Out[13]:
```
0       False
1        True
2       False
3       False
4       False
        ...
1150    False
1151    False
1152    False
1153    False
1154    False
Name: PURPOSE*, Length: 1155, dtype: bool
```

Q7. How many missing values are present? (3 points)

Hint: Find out the total number of missing values across all the variables

In [14]:
```
## This function helps to display the total count of missing value in the dataset.
uber_drives.isnull().sum().sum()
```

Out[14]:
```
502
```

In [15]:
```
print(color.BOLD + color.PURPLE +
      'Total number of missing values across all the variables are :',uber_drives.isnull().sum().sum(), color.END)
```

```
Total number of missing values across all the variables are : 502
```

Q8. Get the summary of the original data. (3 points).

Hint: Summary includes- Count,Mean, Std, Min, 25%,50%,75% and max

Note:Outcome will contain only numerical column.

In this dataset we have only one numerical column which is MILES.

In [16]:
```
## This function helps to get the summary of the numerical data which is MILES.
uber_drives.describe().T
```

Out[16]:

	count	mean	std	min	25%	50%	75%	max
MILES*	1155.0	10.56684	21.579106	0.5	2.9	6.0	10.4	310.3

Q9. Drop the missing values and store data in a new dataframe (name it"df") (4-points)

Note: Dataframe "df" will not contain any missing value

In [17]:
```
## This function has created a copy of uber_drives into df dataset.
df=uber_drives.copy()
```

In [18]:
```
## IN the df dataset we have dropped all the NULL values permanently by using the parameter inplace = True
df.dropna(axis=0,inplace=True)
```

In [19]:
```
## By using this function we can see the ROWS with NULL value has been permanently dropped from the df dataset.
df.shape
```

Out[19]: (653, 7)

Q10. Check the information of the dataframe(df). (2 points)

Hint: Information includes - Total number of columns,variable data-types, number of non-null values in a variable, and usage

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 653 entries, 0 to 1154
Data columns (total 7 columns):
START_DATE*    653 non-null object
END_DATE*      653 non-null object
CATEGORY*      653 non-null object
START*         653 non-null object
STOP*          653 non-null object
MILES*         653 non-null float64
PURPOSE*       653 non-null object
dtypes: float64(1), object(6)
memory usage: 40.8+ KB
```

Now the no. of ROWS got reduced to 653 as we have dropped the null rows (i.e. 502) from the df dataset.

Q11. Get the unique start destinations. (3 points)

Note: This question is based on the dataframe with no 'NA' values

Hint- You need to print the unique destination place names in this and not the count.

In [21]:
```
## This function helps to get the unique START destinations considering the dataset 'df' (excluding null rows).
df['START*'].unique()
```

Out[21]:
```
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)
```

In [22]:
```
print(color.BOLD + color.PURPLE + "The total number of unique start destinations in the df dataset :",df['START*'].nun
ique(),
     color.END)
```

The total number of unique start destinations in the df dataset : 131

Q12. What is the total number of unique start destinations? (3 points)

Note: Use the original dataframe without dropping 'NA' values

In [23]:
```
print(color.BOLD + color.PURPLE + "The total number of unique start destinations in the original dataset(uber_drives)
 :"
      ,uber_drives['START*'].nunique(), color.END)
```

The total number of unique start destinations in the original dataset(uber_drives) : 176

In [24]:
```
## This function helps to get the unique START destinations considering the dataset 'original dataset(uber_drives)'.
uber_drives['START*'].unique()
```

Out[24]:
```
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', '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', 'Katunayake', 'Gampaha'],
      dtype=object)
```

Q13. What is the total number of unique stop destinations. (2 points)

Note: Use the original dataframe without dropping 'NA' values.

In [25]:
```
print(color.BOLD + color.PURPLE + "The total number of unique stop destinations in the original dataset(uber_drives)
:"
      ,uber_drives['STOP*'].nunique(), color.END)
```

The total number of unique stop destinations in the original dataset(uber_drives) : 187

In [26]:
```
## This function helps to get the unique STOP destinations considering the dataset 'original dataset(uber_drives)'.
uber_drives['STOP*'].unique()
```

Out[26]:
```
array(['Fort Pierce', 'West Palm Beach', 'Palm Beach', 'Cary',
       'Morrisville', 'New York', 'Queens', 'East Harlem', 'NoMad',
       'Midtown', 'Midtown East', 'Hudson Square', 'Lower Manhattan',
       "Hell's Kitchen", 'Queens County', 'Gulfton', 'Downtown',
       'Houston', 'Jamestown Court', 'Durham', 'Whitebridge',
       'Lake Wellingborough', 'Raleigh', 'Umstead', 'Hazelwood',
       'Westpark Place', 'Meredith Townes', 'Leesville Hollow', 'Apex',
       'Chapel Hill', 'Williamsburg Manor', 'Macgregor Downs',
       'Edgehill Farms', 'Northwoods', 'Tanglewood', 'Preston',
       'Walnut Terrace', 'Jackson Heights', 'East Elmhurst',
       'Midtown West', 'Long Island City', 'Jamaica', 'Unknown Location',
       'Colombo', 'Nugegoda', 'Katunayaka', 'Islamabad', 'R?walpindi',
       'Noorpur Shahan', 'Heritage Pines', 'Waverly Place', 'Wayne Ridge',
       'Depot Historic District', 'Weston', 'West University',
       'South Congress', 'Arts District', 'Congress Ave District',
       'Red River District', 'The Drag', 'Convention Center District',
       'North Austin', 'Coxville', 'Katy', 'Alief', 'Sharpstown',
       'Sugar Land', 'Galveston', 'Port Bolivar', 'Washington Avenue',
       'Briar Meadow', 'Greater Greenspoint', 'Latta', 'Jacksonville',
       'Kissimmee', 'Isles of Buena Vista', 'Orlando', 'Lake Reams',
       'Vista East', 'Sky Lake', 'Sand Lake Commons', 'Daytona Beach',
       'Ridgeland', 'Florence', 'Cedar Hill', 'Holly Springs',
       'Harden Place', 'Chessington', 'Burtrose', 'Parkway',
       'Capitol One', 'University District', 'Redmond', 'Bellevue',
       'Seattle', 'Mcvan', 'Palo Alto', 'Sunnyvale', 'Newark',
       'Menlo Park', 'San Francisco', 'Parkway Museums', 'Hog Island',
       'Savon Height', 'Kildaire Farms', 'Kilarney Woods',
       'Gramercy-Flatiron', 'Tudor City', 'Soho', 'Tribeca',
       'Financial District', 'Kips Bay', 'Emeryville', 'Berkeley',
       'Oakland', 'Bay Farm Island', 'New Orleans',
       'Lower Garden District', 'Lakeview', 'Storyville',
       'Faubourg Marigny', 'Metairie', 'Kenner', 'Bywater', 'Chalmette',
       'Arabi', 'Pontchartrain Shores', 'Marigny', 'Covington',
       'Mandeville', 'Summerwinds', 'Parkwood', 'Pontchartrain Beach',
       'CBD', 'St Thomas', 'Banner Elk', 'Elk Park', 'Newland', 'Boone',
       'Stonewater', 'Lexington Park at Amberly',
       'Arlington Park at Amberly', 'Washington', 'K Street',
       'Kalorama Triangle', 'Northwest Rectangle', 'Columbia Heights',
       'Arlington', 'Farmington Woods', 'Wake Forest', 'Lahore',
       'Karachi', 'French Quarter', 'North Berkeley Hills', 'Southside',
       'San Jose', 'Eagle Rock', 'Huntington Woods', 'Winston Salem',
       'Asheville', 'Topton', 'Hayesville', 'Bryson City', 'Almond',
       'Mebane', 'Santa Clara', 'Cory', 'Agnew', 'Renaissance',
       'West Berkeley', 'Central', 'Sunnyside', 'Ingleside',
       'Potrero Flats', 'SOMISSPO', 'Tenderloin', 'College Avenue',
       'South', 'Southwest Berkeley', 'South Berkeley', 'Mountain View',
       'El Cerrito', 'Wake Co.', 'Fuquay-Varina', 'Rawalpindi', 'Gampaha',
       'Ilukwatta'], dtype=object)
```

Q14. Display all the Uber trips that has the starting point of San Francisco. (3 points)

Note: Use the original dataframe without dropping the 'NA' values.

Hint: You need to display the rows which has starting point of San Francisco. Try using loc function

In [27]:
```
## This function helps to get the details around starting point San Francisco.
A14=uber_drives[uber_drives.loc[:,'START*']=="San Francisco"]
A14
```

Out[27]:

	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

In [28]:
```
print(color.BOLD + color.PURPLE + 'Uber trips that has the starting point of San Francisco in the original dataset are
: '
      , A14['START*'].count(), color.END)
```

```
Uber trips that has the starting point of San Francisco in the original dataset are :  8
```

Q15. What is the most popular starting point for the Uber drivers? (2 points)

Note: Use the original dataframe without dropping the 'NA' values.

Hint:Popular means the place that is visited the most

In [29]:
```
## By using this function :
## Step 1 : Value_counts gets the summary of STARTING points, sorted by descending order.
## Step 2 : Head() displays the first (most popular starting point) value from the dataset.
uber_drives['START*'].value_counts().head(1)
```

Out[29]:
```
Cary    201
Name: START*, dtype: int64
```

In [30]:
```
print(color.BOLD + color.PURPLE + "Most popular starting point for the Uber drivers is"
      , uber_drives['START*'].value_counts().idxmax()
      ,'with the count',uber_drives['START*'].value_counts().head(1)[0], color.END)
```

```
Most popular starting point for the Uber drivers is Cary with the count 201
```

Q16. What is the most popular dropping point for the Uber drivers? (2 points)

Note: Use the original dataframe without dropping the 'NA' values.

Hint: Popular means the place that is visited the most

In [31]:
```
## By using this function :
## Step 1 : Value_counts gets the summary of DROPPING points, sorted by descending order.
## Step 2 : Head() displays the first (most popular dropping point) value from the dataset.
uber_drives['STOP*'].value_counts().head(1)
```

Out[31]:
```
Cary    203
Name: STOP*, dtype: int64
```

In [32]:
```
print(color.BOLD + color.PURPLE + "Most popular dropping point for the Uber drivers is",
      uber_drives['STOP*'].value_counts().idxmax(),'with the count',
      uber_drives['STOP*'].value_counts().head(1)[0], color.END)
```

```
Most popular dropping point for the Uber drivers is Cary with the count 203
```

Q17. List the most frequent route taken by Uber drivers. (4 points)

Note: This question is based on the new dataframe with no 'na' values.

Hint-Print the most frequent route taken by Uber drivers (Route= combination of START & END points present in the Data set). One may use Groupby function

In [33]:
```
## Create a new field called Route to merge start and stop columns.
Route = df['START*']+ " -> " + df['STOP*']
Route
```

Out[33]:
```
0                 Fort Pierce -> Fort Pierce
2                 Fort Pierce -> Fort Pierce
3                 Fort Pierce -> Fort Pierce
4             Fort Pierce -> West Palm Beach
5         West Palm Beach -> West Palm Beach
                         ...
1150                    Karachi -> Karachi
1151             Karachi -> Unknown Location
1152    Unknown Location -> Unknown Location
1153                  Katunayake -> Gampaha
1154                   Gampaha -> Ilukwatta
Length: 653, dtype: object
```

In [34]:
```
## This function adds the Route field into the df dataset.
df['Route']=Route
```

In [50]:
```
## First select Route and start date from the df dataset and then group it by Route.
## Next step is to count Route points and sort the count in decending order
d17 = df[['Route','START_DATE*']].groupby('Route').count().sort_values('START_DATE*',ascending = False)
## Reset the index to get details in proper format
d17.reset_index(inplace = True)
d17
```

Out[50]:

	Route	START_DATE*
0	Cary -> Morrisville	52
1	Morrisville -> Cary	51
2	Cary -> Cary	44
3	Unknown Location -> Unknown Location	30
4	Cary -> Durham	30
...	...	...
234	Katunayaka -> Katunayaka	1
235	Katunayaka -> Unknown Location	1
236	Katunayake -> Gampaha	1
237	Katy -> Houston	1
238	Winston Salem -> Asheville	1

239 rows × 2 columns

In [49]:
```
print(color.BOLD + color.PURPLE + 'The most frequent route taken by Uber drivers is : ', d17.iloc[0,0]
      , ' with count : ', d17.iloc[0,1], color.END)
```

The most frequent route taken by Uber drivers is : Cary -> Morrisville with count : 52

Q18. Display all types of purposes for the trip in an array. (3 points)

Note: This question is based on the new dataframe with no 'NA' values.

In [37]:
```
print(color.BOLD + color.PURPLE + "The unique types of purposes for the trip in the new dataset (df) :"
      ,df['PURPOSE*'].nunique(), color.END)
```

The unique types of purposes for the trip in the new dataset (df) : 10

In [38]:
```
## This function helps to get the unique types of purposes in the dataset 'new dataset(df)'.
df['PURPOSE*'].unique()
```

Out[38]:
```
array(['Meal/Entertain', 'Errand/Supplies', 'Meeting', 'Customer Visit',
       'Temporary Site', 'Between Offices', 'Charity ($)', 'Commute',
       'Moving', 'Airport/Travel'], dtype=object)
```

Q19. Plot a bar graph of Purpose vs Miles(Distance). (3 points)

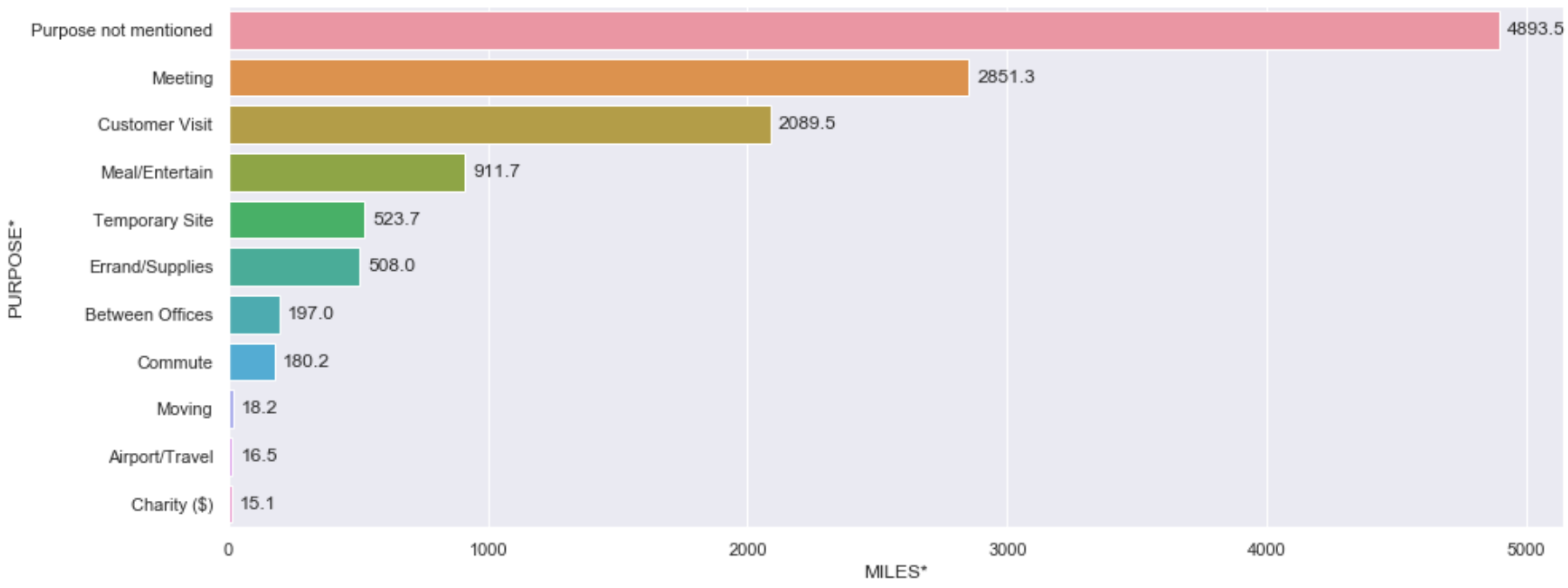
Note: Use the original dataframe without dropping the 'NA' values.

In [75]:
```
## Create a copy of the original dataframe as d19
d19 = uber_drives.copy()
## By using fillna function replace the "NA" value in purpose field to " Puprose not mentioned"
d19.fillna(value='Purpose not mentioned',inplace=True)
## Using groupby function get the total miles purpose-wise in descending order and save it in a19 dataframe.
a19=d19[['PURPOSE*','MILES*']].groupby('PURPOSE*').sum().sort_values('MILES*',ascending = False)
## Reset the index to get details in proper format
a19.reset_index(inplace = True)
a19
```

Out[75]:

	PURPOSE*	MILES*
0	Purpose not mentioned	4893.5
1	Meeting	2851.3
2	Customer Visit	2089.5
3	Meal/Entertain	911.7
4	Temporary Site	523.7
5	Errand/Supplies	508.0
6	Between Offices	197.0
7	Commute	180.2
8	Moving	18.2
9	Airport/Travel	16.5
10	Charity ($)	15.1

In [92]:
```
## This function helps to get the barplot for miles and purpose including the NA value which was replaced as
## "Purpose not defined"
plt.figure(figsize=(15, 6))

ax = sns.barplot(x='MILES*',y='PURPOSE*',data=a19)


### This is custom made function to get annotation in the dataset.

annotate = ax.patches

# For each bar: Place a label
for annot in annotate:
    # Get X and Y placement of label from rect.
    x_value = annot.get_width()
    y_value = annot.get_y() + annot.get_height() / 2

    # Number of points between bar and label.
    space = 5
    # Vertical alignment for positive values
    ha = 'left'

    # If value of bar is negative: Place label left of bar
    if x_value < 0:
        # Invert space to place label to the left
        space *= -1
        # Horizontally align label at right
        ha = 'right'

    # Use X value as label and format number with one decimal place
    label = "{:.1f}".format(x_value)

    # Create annotation
    plt.annotate(
        label,                      # Use `label` as label
        (x_value, y_value),         # Place label at end of the bar
        xytext=(space, 0),          # Horizontally shift label by `space`
        textcoords="offset points", # Interpret `xytext` as offset in points
        va='center',                # Vertically center label
        ha=ha)                      # Horizontally align label differently for
                                    # positive and negative values.
```



Q20. Display a dataframe of Purpose and the distance travelled for that particular Purpose. (3 points)

Note: Use the original dataframe without dropping "NA" values

In [41]:
```
## Create a copy of the original dataframe as d20
d20 = uber_drives.copy()
## By using fillna function replace the "NA" value in purpose field to " Puprose not mentioned"
d20.fillna(value='Purpose not mentioned',inplace=True)
## Using groupby function get the total miles purpose-wise in descending order and save it in a20 dataframe.
a20=d20[['PURPOSE*','MILES*']].groupby('PURPOSE*').sum().sort_values('MILES*',ascending = False)
## Reset the index to get details in proper format
a20.reset_index(inplace = True)
a20
```

Out[41]:

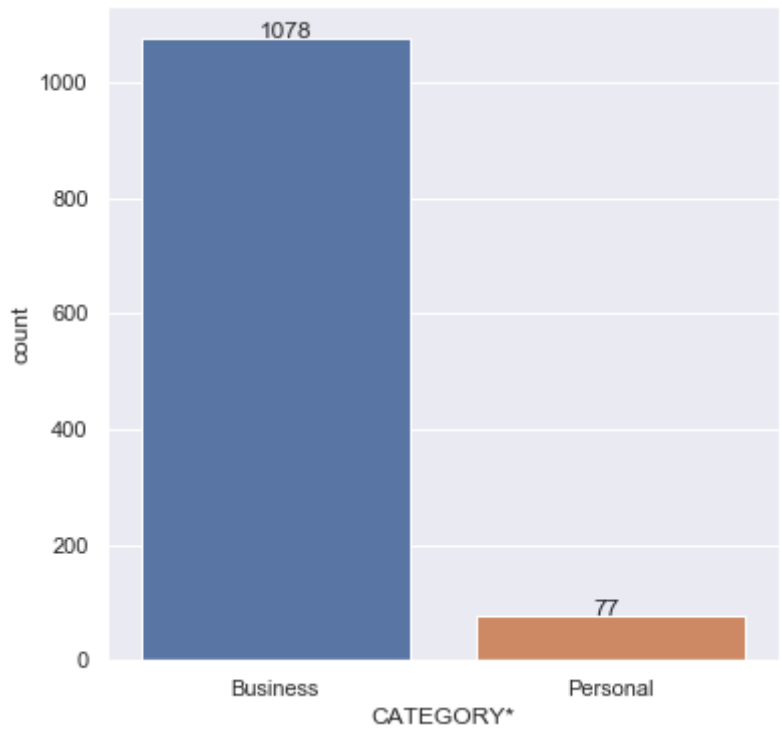
	PURPOSE*	MILES*
0	Purpose not mentioned	4893.5
1	Meeting	2851.3
2	Customer Visit	2089.5
3	Meal/Entertain	911.7
4	Temporary Site	523.7
5	Errand/Supplies	508.0
6	Between Offices	197.0
7	Commute	180.2
8	Moving	18.2
9	Airport/Travel	16.5
10	Charity ($)	15.1

Q21. Plot number of trips vs Category of trips. (4 points)

Note: Use the original dataframe without dropping the 'NA' values.

Hint : You can make a countplot or barplot.

In [89]:
```
## This function displays the number of trips category-wise for the original dataset(uber_drives)

plt.figure(figsize=(6, 6))
a21 = sns.countplot(x='CATEGORY*',data=uber_drives)


## This FOR loop function helps to get the annotate of the count of the category in the count plot
for p, label in zip(a21.patches, uber_drives["CATEGORY*"].value_counts()):
    a21.annotate(label, (p.get_x()+0.35, p.get_height()+0.35))
```



Q22. What is proportion of miles that are covered as Business trips and what is the proportion of miles that are covered as Personal trips? (4 points)

Note:Use the original dataframe without dropping the 'NA' values. The proportion calculation is with respect to the 'miles' variable.

Hint: Proportion of miles covered as business trips= (Total Miles clocked as Business Trips)/ (Total Miles)

Proportion of miles covered as personal trips= (Total Miles clocked as Personal Trips)/ (Total Miles)

In [43]:
```
## Create variable "TotalMiles" for getting total miles covered in the dataset.
TotalMiles=uber_drives['MILES*'].sum()
TotalMiles
```

Out[43]: 12204.7

In [44]:
```
## Create variable "BusinessTripMiles" for getting total miles covered under "Business" category in the dataset.
BusinessTripMiles=uber_drives[uber_drives['CATEGORY*']=='Business']['MILES*'].sum()
BusinessTripMiles
```

Out[44]: 11487.0

In [45]:
```
## Create variable "PersonalTripMiles" for getting total miles covered under "Personal" category in the dataset.
PersonalTripMiles=uber_drives[uber_drives['CATEGORY*']=='Personal']['MILES*'].sum()
PersonalTripMiles
```

Out[45]: 717.6999999999999

In [46]:
```
print(color.BOLD + color.PURPLE + "Proportion of miles covered as business trips :"
      ,((BusinessTripMiles/TotalMiles)*100).round(2),'%', color.END)
```

Proportion of miles covered as business trips : 94.12 %

In [47]:
```
print(color.BOLD, color.PURPLE + "Proportion of miles covered as personal trips :"
      ,((PersonalTripMiles/TotalMiles)*100).round(2),'%', color.END)
```

Proportion of miles covered as personal trips : 5.88 %
