In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df = pd.read_csv('UberDataset.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	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 [5]:
```
df.shape
```

Out[5]:
```
(1156, 7)
```

In [6]:
```
df.dtypes
```

Out[6]:
```
START_DATE     object
END_DATE       object
CATEGORY       object
START          object
STOP           object
MILES         float64
PURPOSE        object
dtype: object
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
START_DATE      0
END_DATE        1
CATEGORY        1
START           1
STOP            1
MILES           0
PURPOSE       503
dtype: int64
```

In [12]:
```
df.describe()
```

Out[12]:

	MILES
count	1156.000000
mean	21.115398
std	359.299007
min	0.500000
25%	2.900000
50%	6.000000
75%	10.400000
max	12204.700000

In [13]:
```
duplicate = df.duplicated()
duplicate.sum()
```

Out[13]:
```
1
```

In [14]:
```
df.drop_duplicates(inplace=True)
df.duplicated().sum()
```

Out[14]:
```
0
```

In [8]:
```
df.info()
```

```
<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 [15]:
```
null_data =df[df['PURPOSE'].isnull()]
null_data
```

Out[15]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE
1	01-02-2016 01:25	01-02-2016 01:37	Business	Fort Pierce	Fort Pierce	5.0	NaN
32	1/19/2016 9:09	1/19/2016 9:23	Business	Whitebridge	Lake Wellingborough	7.2	NaN
85	02-09-2016 10:54	02-09-2016 11:07	Personal	Whitebridge	Northwoods	5.3	NaN
86	02-09-2016 11:43	02-09-2016 11:50	Personal	Northwoods	Tanglewood	3.0	NaN
87	02-09-2016 13:36	02-09-2016 13:52	Personal	Tanglewood	Preston	5.1	NaN
...	...	...	...	...	...	...	...
1066	12/19/2016 14:37	12/19/2016 14:50	Business	Unknown Location	Unknown Location	5.4	NaN
1069	12/19/2016 19:05	12/19/2016 19:17	Business	Islamabad	Unknown Location	2.2	NaN
1071	12/20/2016 8:49	12/20/2016 9:24	Business	Unknown Location	Rawalpindi	12.0	NaN
1143	12/29/2016 20:53	12/29/2016 21:42	Business	Kar?chi	Unknown Location	6.4	NaN
1155	Totals	NaN	NaN	NaN	NaN	12204.7	NaN

503 rows × 7 columns

In [16]:
```
df['PURPOSE'].fillna("Not Mentioned",inplace=True)
```

In [17]:
```
null_data =df[df['PURPOSE'].isnull()]
null_data
```

Out[17]:

START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE

In [18]:
```
df.isnull().sum()
```

Out[18]:
```
START_DATE    0
END_DATE      1
CATEGORY      1
START         1
STOP          1
MILES         0
PURPOSE       0
dtype: int64
```

In [19]:
```
df.dropna(inplace =True)
```

In [20]:
```
df.isnull().sum()
```

Out[20]:
```
START_DATE    0
END_DATE      0
CATEGORY      0
START         0
STOP          0
MILES         0
PURPOSE       0
dtype: int64
```

In [21]:
```
df.dtypes
```

Out[21]:
```
START_DATE     object
END_DATE       object
CATEGORY       object
START          object
STOP           object
MILES         float64
PURPOSE        object
dtype: object
```

In [22]:
```
#changing the Datatypes
df['START_DATE'] = pd.to_datetime(df['START_DATE'], errors='coerce')
df['END_DATE'] = pd.to_datetime(df['END_DATE'], errors='coerce')
```

In [23]:
```
df.dtypes
```

Out[23]:
```
START_DATE    datetime64[ns]
END_DATE      datetime64[ns]
CATEGORY              object
START                 object
STOP                  object
MILES                float64
PURPOSE               object
dtype: object
```

In [24]:
```
import datetime as dt

df["START_TIME"]=df['START_DATE'].dt.strftime('%H:%M')
df["END_TIME"]=df['END_DATE'].dt.strftime('%H:%M')
df.head()
```

Out[24]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE	START_TIME	END_TIME
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	21:11	21:17
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	Not Mentioned	01:25	01:37
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	20:25	20:38
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	17:31	17:45
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	14:42	15:49

In [25]:
```
#duration=data['END_TIME']-data['START_TIME']
df['DURATION']=df['END_DATE']-df['START_DATE']
df.head()
```

Out[25]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE	START_TIME	END_TIME	DURATION
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	21:11	21:17	0 days 00:06:00
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	Not Mentioned	01:25	01:37	0 days 00:12:00
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	20:25	20:38	0 days 00:13:00
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	17:31	17:45	0 days 00:14:00
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	14:42	15:49	0 days 01:07:00

In [28]:
```
df['START'] = df['START'].replace("Kar?chi", "Karachi")
df['STOP'] = df['STOP'].replace("Kar?chi", "Karachi")
```

Exploratory Data Analysis

In [26]:
```
df.CATEGORY.unique()
```

Out[26]:
```
array(['Business', 'Personal'], dtype=object)
```

In [27]:
```
df.PURPOSE.unique()
```

Out[27]:
```
array(['Meal/Entertain', 'Not Mentioned', 'Errand/Supplies', 'Meeting',
       'Customer Visit', 'Temporary Site', 'Between Offices',
       'Charity ($)', 'Commute', 'Moving', 'Airport/Travel'], dtype=object)
```

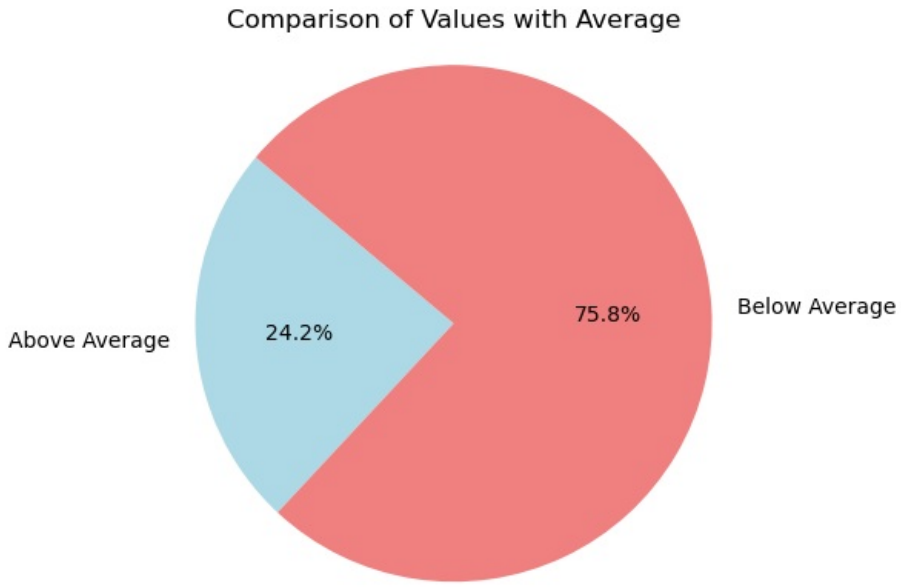
In [29]:
```
average_distance_travelled = df.groupby('PURPOSE')['MILES'].mean()
average_distance_travelled
```

Out[29]:
```
PURPOSE
Airport/Travel       5.500000
Between Offices     10.944444
Charity ($)         15.100000
Commute            180.200000
Customer Visit      20.688119
Errand/Supplies      3.968750
Meal/Entertain       5.698125
Meeting             15.276344
Moving               4.550000
Not Mentioned        9.748008
Temporary Site      10.474000
Name: MILES, dtype: float64
```

Comparison with the average distance travelled by passengers

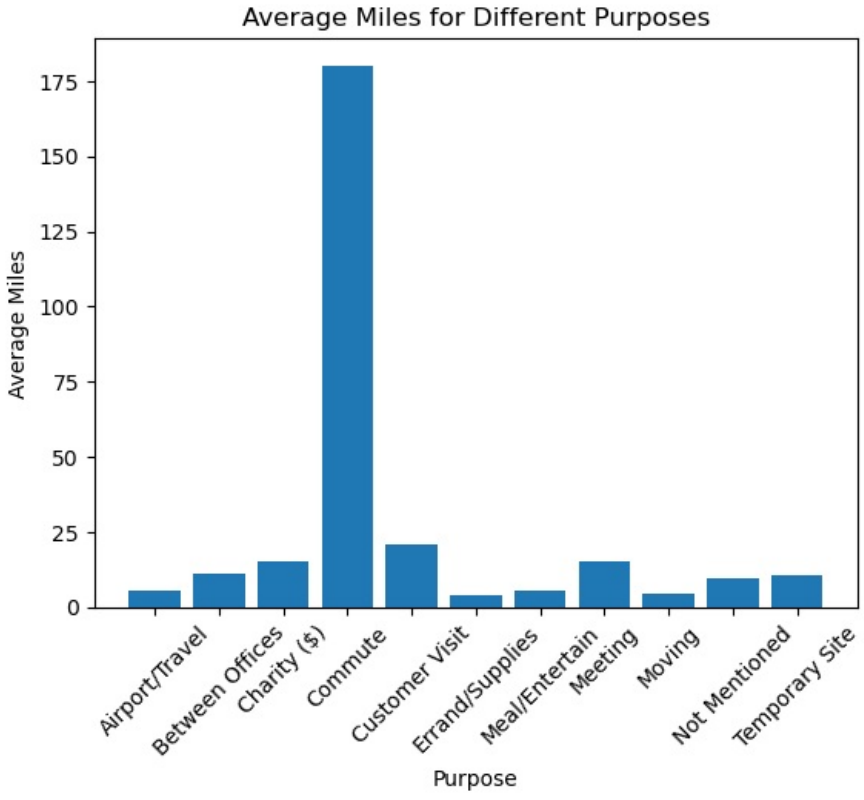
In [30]:
```
average_value = df['MILES'].mean()

# Compare values with the average and categorize them
above_average = df[df['MILES'] > average_value].shape[0]
```

```
below_average = df[df['MILES'] <= average_value].shape[0]

# Data for the pie chart
sizes = [above_average, below_average]
labels = ['Above Average', 'Below Average']
colors = ['lightblue', 'lightcoral']

# Create a pie chart
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.axis('equal')

plt.title('Comparison of Values with Average')
plt.show()
```



Average miles travelled for different purposes

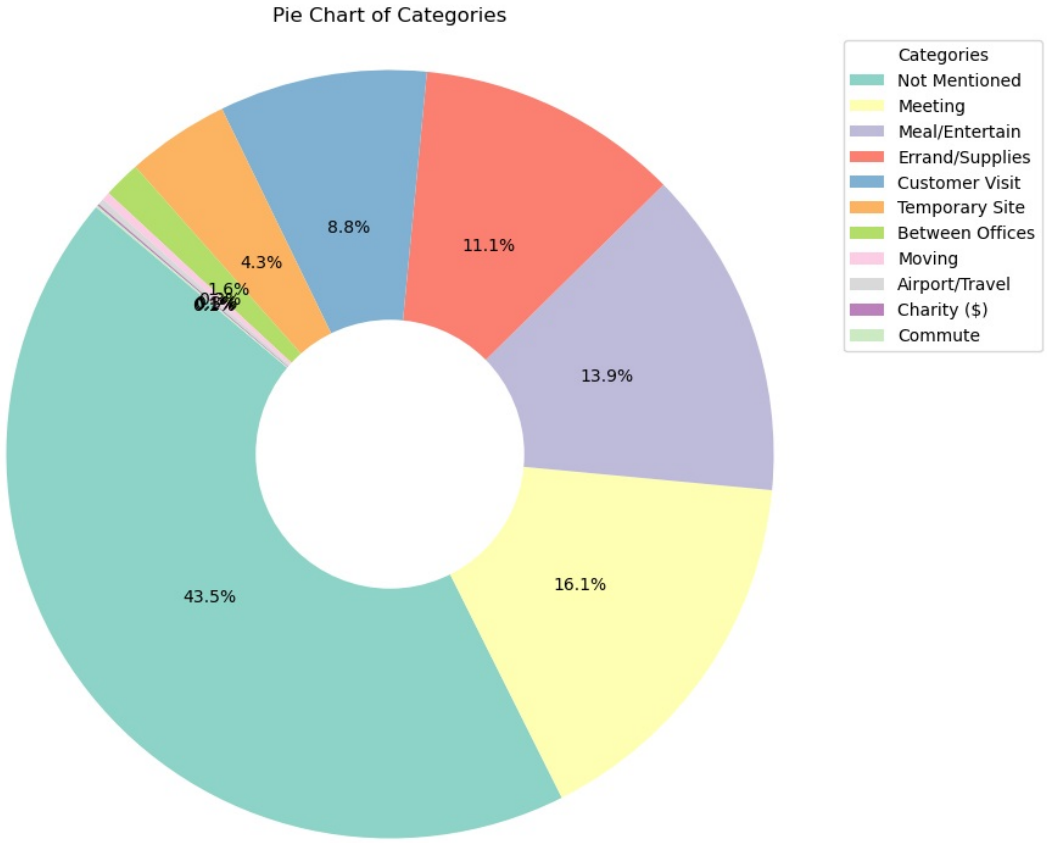
In [31]:
```
plt.bar(average_distance_travelled.index, average_distance_travelled.values)
plt.xlabel('Purpose')
plt.ylabel('Average Miles')
plt.title('Average Miles for Different Purposes')
plt.xticks(rotation=45)
plt.show()
```



Type of Passengers Category Wise

In [32]:
```
# Count the occurrences of each category
category_counts = df['PURPOSE'].value_counts()
```

```
# Data for the pie chart
sizes = category_counts.values
labels = category_counts.index
colors = plt.cm.Set3.colors  # Choose a colormap

# Create a pie chart
plt.figure(figsize=(14,9))
plt.pie(sizes, colors=colors, autopct='%1.1f%%', startangle=140)
plt.axis('equal')  # Equal aspect ratio ensures the pie chart is circular.

# Adding white circle in the center to make it look like a donut chart (optional)
centre_circle = plt.Circle((0,0),0.35,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)


plt.legend(labels, title='Categories', loc='best')
plt.title('Pie Chart of Categories')
plt.show()
```

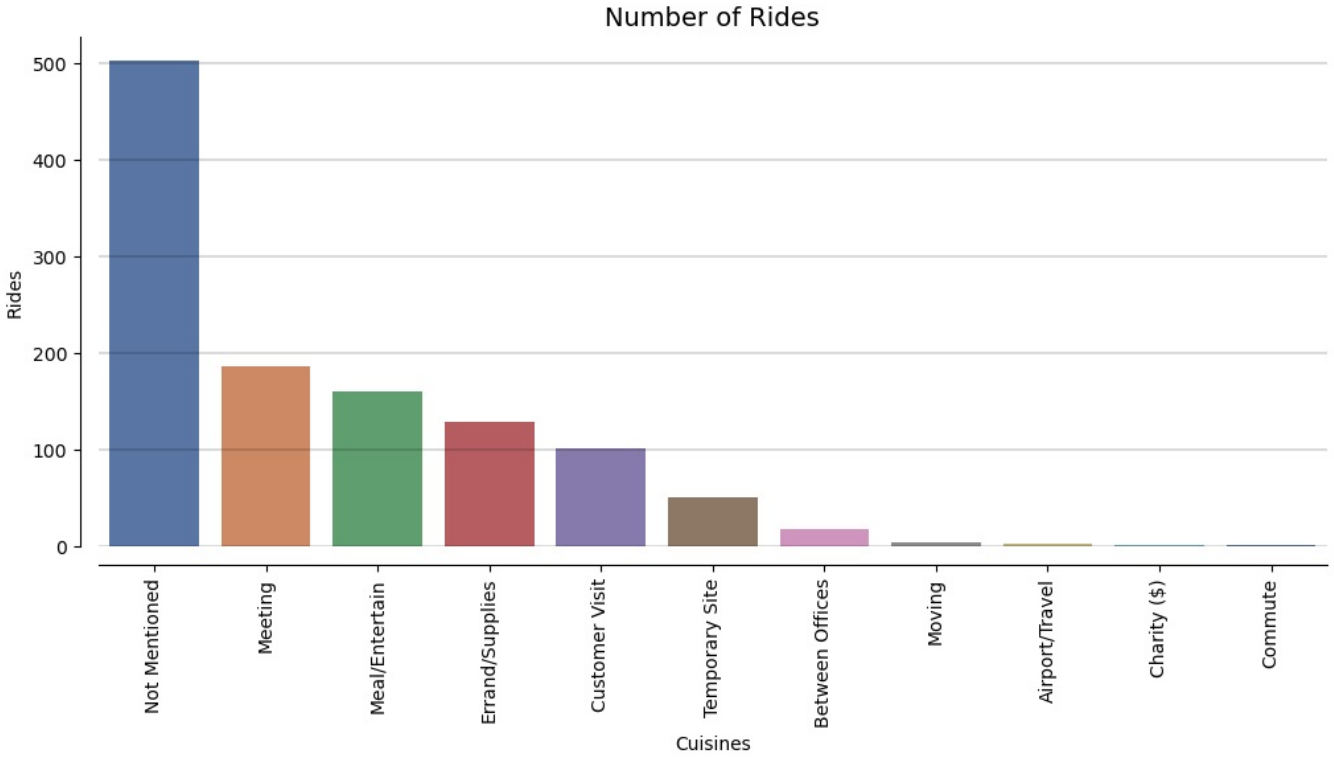


In [33]:
```
purpose=df['PURPOSE'].value_counts()
purpose
```

Out[33]:
```
Not Mentioned      502
Meeting            186
Meal/Entertain     160
Errand/Supplies    128
Customer Visit     101
Temporary Site      50
Between Offices     18
Moving               4
Airport/Travel       3
Charity ($)          1
Commute              1
Name: PURPOSE, dtype: int64
```

In [34]:
```
c = df.PURPOSE.value_counts().reset_index().rename({'index':'PURPOSE','PURPOSE':'frequency'}, axis =1)

plt.figure(figsize=(12,5))

#color of graph
sns.barplot(data = c.head(20),
            x='PURPOSE',
            y = 'frequency',
            palette= "deep",
           )


#differentiating lines
plt.grid(True, axis = 'y',
         color = 'black',
```

```
         linestyle = '-',
         linewidth=0.2
        )

plt.grid(False, axis = 'x')

#rotate the x-axis labels
sns.despine(offset=10, trim=False)
plt.xticks(rotation = 90)

#fontsize
plt.xlabel("Cuisines", fontsize = 10)
plt.ylabel("Rides", fontsize = 10)
plt.title('Number of Rides', fontsize = 14)
plt.show()
```



In [35]:
```
average_miles_by_category = df.groupby('CATEGORY')['MILES'].mean()
average_miles_by_category
```

Out[35]:
```
CATEGORY
Business    10.656546
Personal     9.320779
Name: MILES, dtype: float64
```

In [36]:
```
# least 5 start stations
least_5_start_stations = df['START'].value_counts().nsmallest(5)
least_5_start_stations
```

Out[36]:
```
Fuquay-Varina           1
Wake Co.                1
NOMA                    1
Santa Clara             1
North Berkeley Hills    1
Name: START, dtype: int64
```

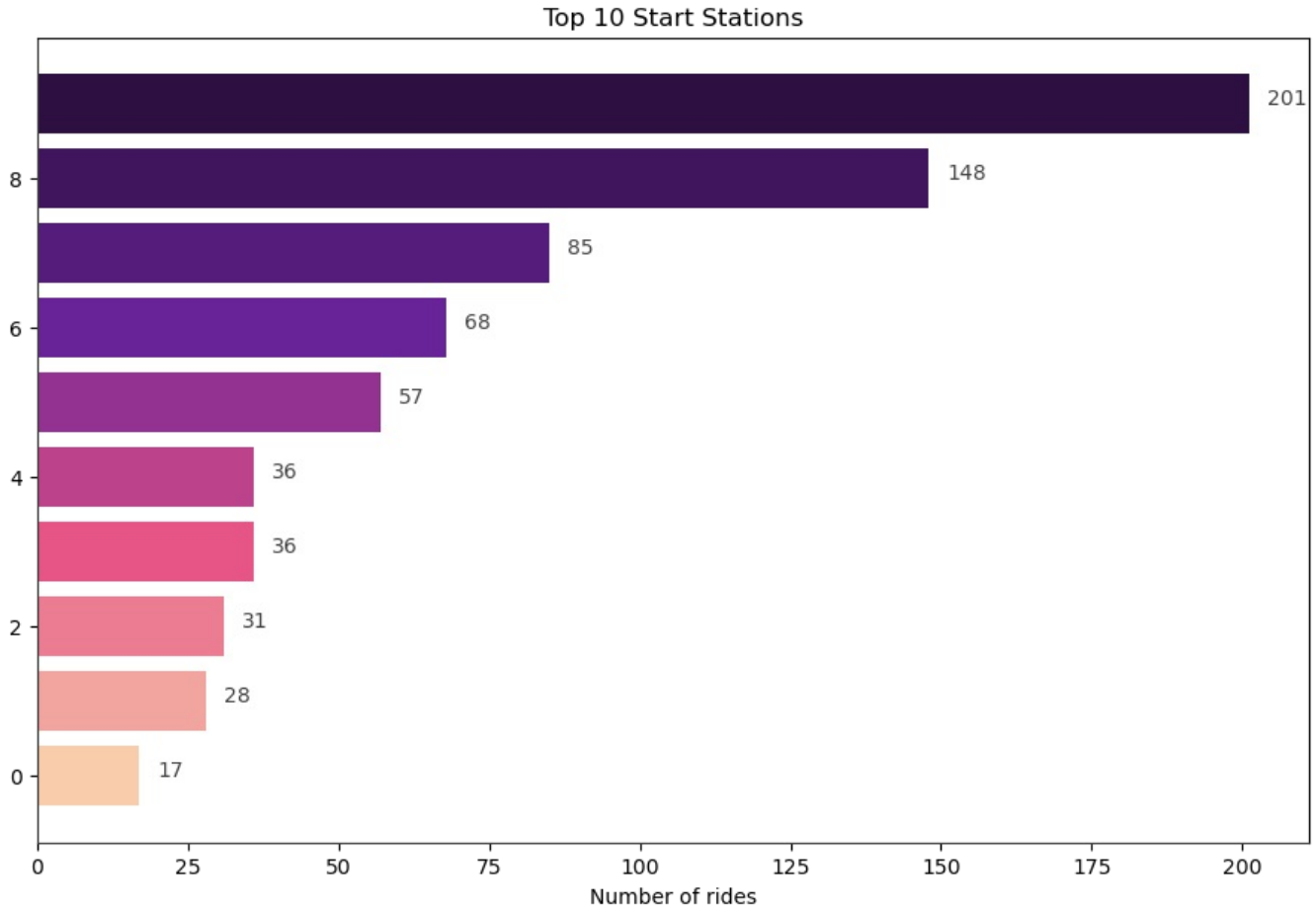
In [37]:
```
# least 5 stop stations
least_5_stop_stations = df['STOP'].value_counts().nsmallest(5)
least_5_stop_stations
```

Out[37]:
```
Arlington Park at Amberly    1
Stonewater                   1
Elk Park                     1
Summerwinds                  1
Parkwood                     1
Name: STOP, dtype: int64
```

Performance of top start and stop stations

In [38]:
```
top10_startstations = df["START"].value_counts()[:10].sort_values(ascending=True)

height = top10_startstations.values
bars = top10_startstations.index
y_pos = np.arange(len(bars))

fig = plt.figure(figsize=[11,7], frameon=False)
ax = fig.gca()
ax.spines["top"].set_visible("#424242")
```

```
ax.spines["right"].set_visible("#424242")
ax.spines["left"].set_color("#424242")
ax.spines["bottom"].set_color("#424242")

colors = ["#f9cdac","#f2a49f","#ec7c92","#e65586","#bc438b","#933291","#692398","#551c7b","#41155e","#2d0f41"]
plt.barh(y_pos, height, color=colors)

plt.xlabel("Number of rides")

for i, v in enumerate(height):
    ax.text(v+3, i, str(v), color='#424242')
plt.title("Top 10 Start Stations")


plt.show()
```



In [39]:
```
top10_stopstations = df["START"].value_counts()[:10].sort_values(ascending=True)
top10_stopstations
```

Out[39]:
```
Westpark Place       17
Raleigh              28
Karachi              31
Lahore               36
Durham               36
Islamabad            57
Whitebridge          68
Morrisville          85
Unknown Location    148
Cary                201
Name: START, dtype: int64
```

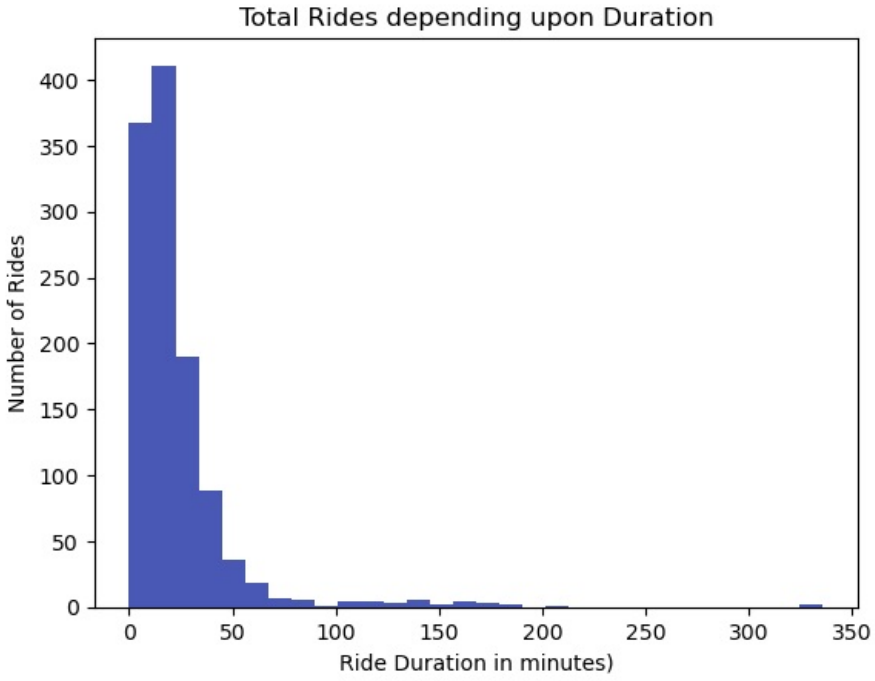
Ride Durations

In [40]:
```
# calculating ride durations by subtracting start time by end time
df['ride_duration'] = df['END_DATE'] - df['START_DATE']

# Calculating min, max, and average ride durations
min_duration = df['ride_duration'].min()
max_duration = df['ride_duration'].max()
average_duration = df['ride_duration'].mean()

print("Minimum ride duration:", min_duration)
print("Maximum ride duration:", max_duration)
print("Average ride duration:", average_duration)
```

```
Minimum ride duration: 0 days 00:00:00
Maximum ride duration: 0 days 05:36:00
Average ride duration: 0 days 00:23:14.506065857
```

In [41]:
```
#plotting the values in histogram
```

```
plt.hist(df['ride_duration'].dt.total_seconds() / 60, bins=30 , color= '#4958B5')
plt.xlabel('Ride Duration in minutes)')
plt.ylabel('Number of Rides')
plt.title('Total Rides depending upon Duration')
plt.show()
```



In [ ]: