In [1]:
```
import pandas as pd
import numpy as np
import time,datetime
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns import
calendar
```

In [2]:
```
uber_data = pd.read_csv('My Uber Drives - 2016.csv')
```

In [3]:
```
uber_data.head()
```

Out[3]:

	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 [4]:
```
#Now lets rename the columns and remove '*' from columns' name.
uber_data = uber_data.rename(columns = {uber_data.columns[0]:'START_DATE',
uber_data.columns[1]:'END_DATE',
uber_data.columns[2]:'CATEGORY',
uber_data.columns[3]:'START',
uber_data.columns[4]:'STOP',
uber_data.columns[5]: 'MILES',
uber_data.columns[6]:'PURPOSE'})
```

In [5]:
```
print(uber_data.isnull().sum())
 print(uber_data.isnull().sum().sum()) uber_data=uber_data.dropna()
```

```
START_DATE      0
END_DATE        1
CATEGORY        1
START           1
STOP            1
MILES           0
PURPOSE       503
dtype: int64 507
```

In [6]:
```
uber_data[uber_data.START.str.contains('\?') == True]
```

Out[6]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE
140	2/20/2016 14:50	2/20/2016 15:54	Business	R? walpindi	R?walpindi	23.1	Meeting
1119	12/27/2016 7:02	12/27/2016 7:14	Business	Kar?chi	Kar?chi	4.9	Temporary Site

1120	12/27/2016 8:37	12/27/2016 8:59	Business	Kar?chi	Kar?chi	5.0	Meal/Entertain
1121	12/27/2016 12:53	12/27/2016 12:57	Business	Kar?chi	Kar?chi	0.6	Meal/Entertain
1122	12/27/2016 14:49	12/27/2016 15:03	Business	Kar?chi	Unknown Location	3.1	Customer Visit
1124	12/27/2016 19:19	12/27/2016 19:50	Business	Kar?chi	Kar?chi	5.5	Customer Visit
1125	12/28/2016 8:34	12/28/2016 9:06	Business	Kar?chi	Unknown Location	10.3	Meal/Entertain
1127	12/28/2016 13:53	12/28/2016 14:01	Business	Kar?chi	Kar?chi	2.0	Errand/Supplies
1128	12/28/2016 15:04	12/28/2016 15:39	Business	Kar?chi	Unknown Location	8.5	Meal/Entertain
1130	12/28/2016 18:33	12/28/2016 18:56	Business	Kar?chi	Kar?chi	3.8	Errand/Supplies
1131	12/28/2016 22:44	12/28/2016 23:18	Business	Kar?chi	Kar?chi	5.1	Errand/Supplies
1132	12/29/2016 0:49	12/29/2016 1:06	Business	Kar?chi	Kar?chi	3.8	Errand/Supplies
1133	12/29/2016 9:44	12/29/2016 10:07	Business	Kar?chi	Unknown Location	11.6	Meal/Entertain
1135	12/29/2016 12:25	12/29/2016 12:33	Business	Kar?chi	Kar?chi	1.4	Errand/Supplies
1136	12/29/2016 13:17	12/29/2016 13:24	Business	Kar?chi	Kar?chi	1.1	Errand/Supplies
1137	12/29/2016 13:56	12/29/2016 14:11	Business	Kar?chi	Kar?chi	4.1	Airport/Travel
1138	12/29/2016 14:42	12/29/2016 14:58	Business	Kar?chi	Kar?chi	6.1	Between Offices
1139	12/29/2016 15:05	12/29/2016 15:16	Business	Kar?chi	Kar?chi	1.3	Errand/Supplies
1140	12/29/2016 18:59	12/29/2016 19:14	Business	Kar?chi	Unknown Location	3.0	Meal/Entertain
1142	12/29/2016 20:15	12/29/2016 20:45	Business	Kar?chi	Kar?chi	7.2	Meeting
1145	12/30/2016 10:15	12/30/2016 10:33	Business	Kar?chi	Kar?chi	2.8	Errand/Supplies

1146	12/30/2016 11:31	12/30/2016 11:56	Business	Kar?chi	Kar?chi	2.9	Errand/Supplies
1147	12/30/2016 15:41	12/30/2016 16:03	Business	Kar?chi	Kar?chi	4.6	Errand/Supplies
	START_DATE	**END_DATE**	**CATEGORY**	**START**	**STOP**	**MILES**	**PURPOSE**
1148	12/30/2016 16:45	12/30/2016 17:08	Business	Kar?chi	Kar?chi	4.6	Meeting
1149	12/30/2016 23:06	12/30/2016 23:10	Business	Kar?chi	Kar?chi	0.8	Customer Visit
1150	12/31/2016 1:07	12/31/2016 1:14	Business	Kar?chi	Kar?chi	0.7	Meeting
1151	12/31/2016 13:24	12/31/2016 13:42	Business	Kar?chi	Unknown Location	3.9	Temporary Site

In [7]:
```
uber_data['START'] = uber_data['START'].replace({"\?":"a"}, regex = True)
uber_data["STOP"] = uber_data["STOP"].replace({"\?":"a"}, regex = True)
```

In [8]:
```
#make string as datetime to identify individual easily
```

```
uber_data['START_DATE'] = pd.to_datetime(uber_data['START_DATE'], format="%m/%d/%Y
uber_data['END_DATE'] = pd.to_datetime(uber_data['END_DATE'], format="%m/%d/%Y %H:%

uber_data['HOUR'] =  [x.hour for x in uber_data['START_DATE']] uber_data['DAY'] =
[x.day for x in uber_data['START_DATE']] uber_data['MONTH'] =  [x.month for x in
uber_data['START_DATE']] uber_data['WEEKDAY'] =  [calendar.day_name[x.dayofweek] for
x in uber_data['START_D uber_data['DAY_OF_WEEK'] =  [x.dayofweek for x in
uber_data['START_DATE']]
```

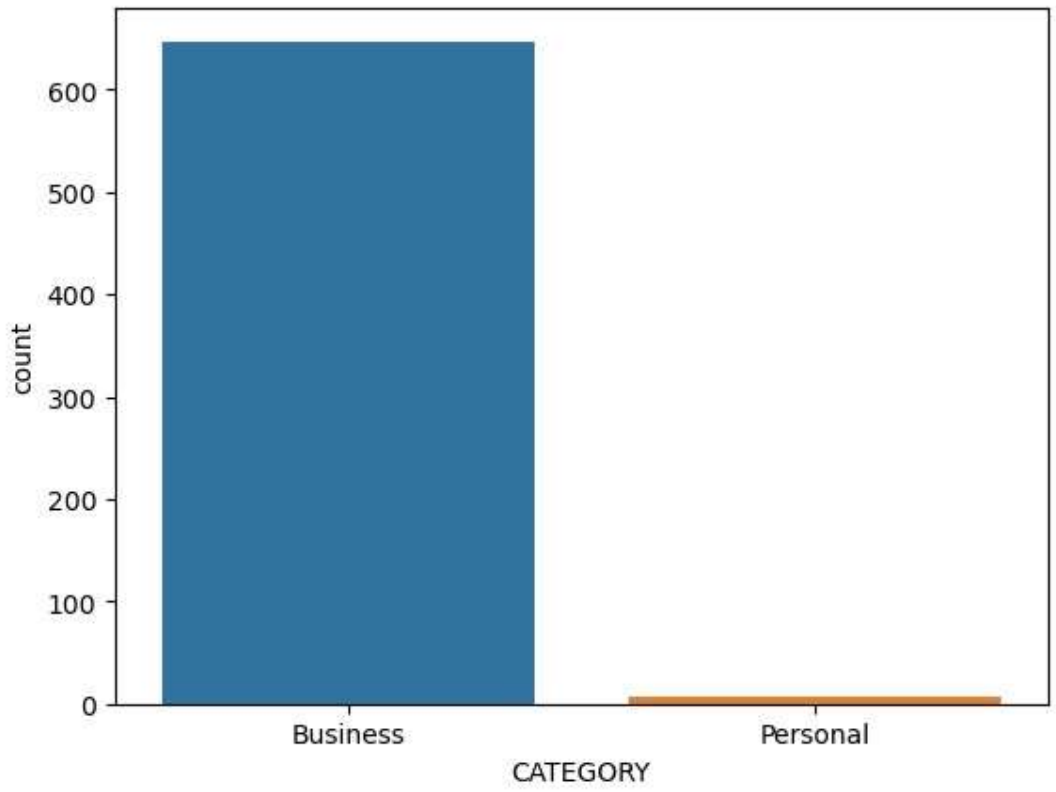
In [9]:
```
uber_data.head()
```

Out[9]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE	HOUR	DAY	MON
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	21	1	
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	20	2	
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	17	5	
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	14	6	
5	2016-01-06 17:15:00	2016-01-06 17:19:00	Business	West Palm Beach	West Palm Beach	4.3	Meal/Entertain	17	6	

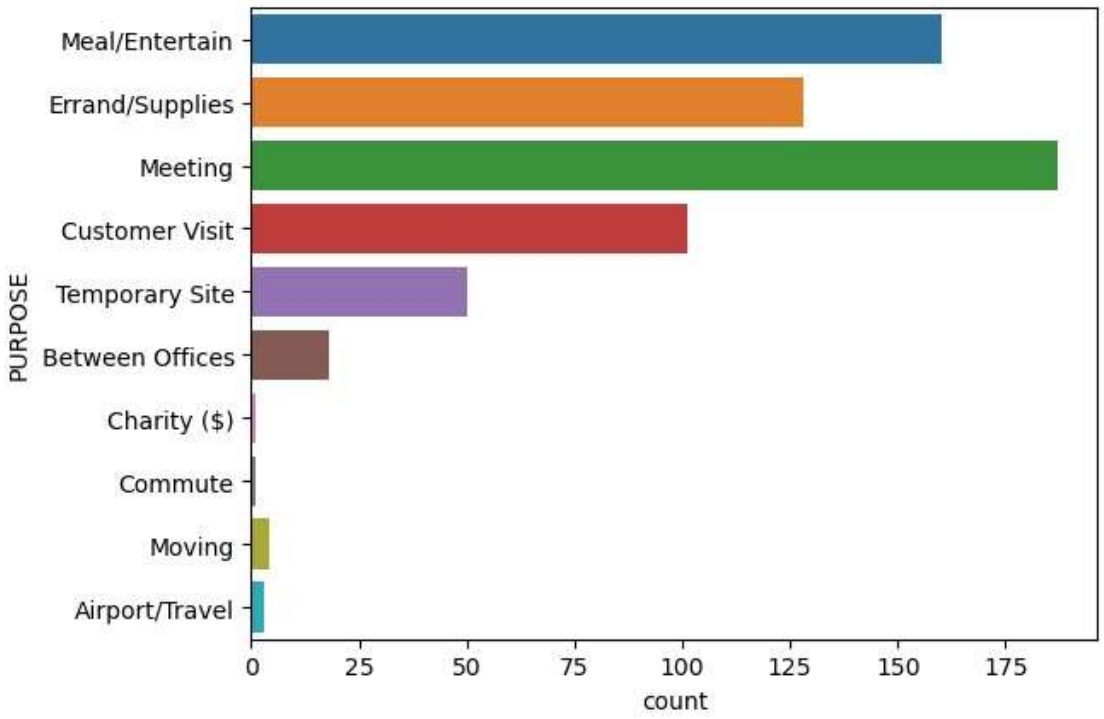
In [10]:
```
sns . countplot(x= 'CATEGORY' ,data=uber_data)
```

Out[10]:
```
<Axes: xlabel='CATEGORY', ylabel='count'>
```

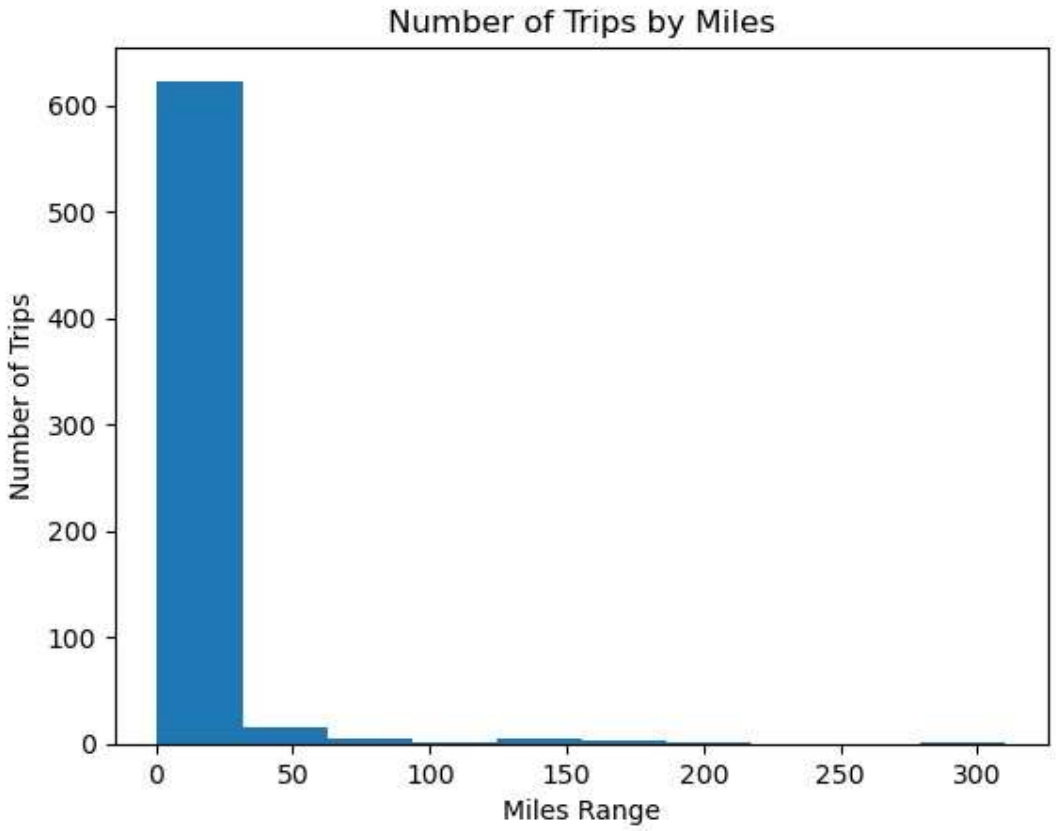


In [11]:
```
sns . countplot(y= 'PURPOSE' ,data=uber_data)
```

Out[11]:
```
<Axes: xlabel='count', ylabel='PURPOSE'>
```

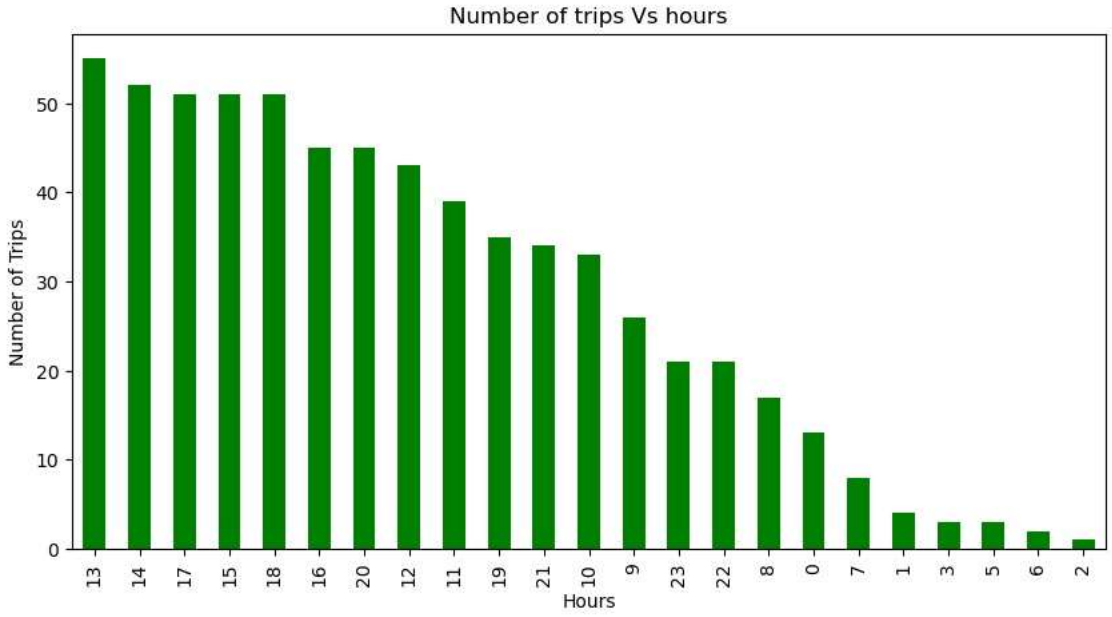


In [12]:
```
uber_data[ 'MILES' ].plot.hist()

plt.xlabel("Miles Range")
plt.ylabel("Number of Trips")
# setting y_label as price



plt.title("Number of Trips by Miles")
plt.show()
```



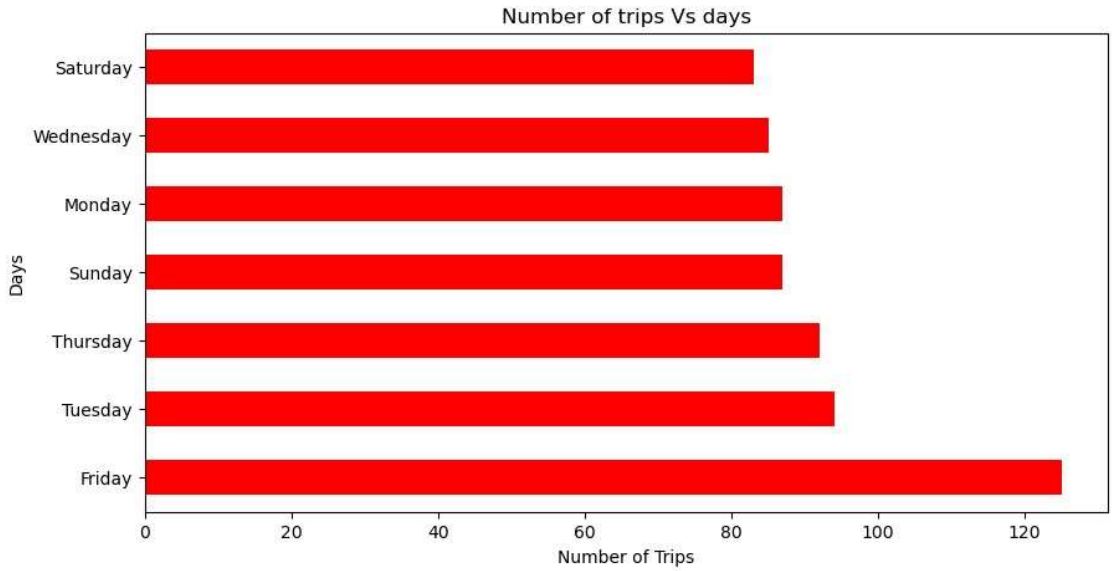
In [13]:
```
hours = uber_data['HOUR'].value_counts()
hours.plot(kind= 'bar', color= 'green', figsize= (10,5))
plt.xlabel( 'Hours' ) plt.ylabel( 'Number of Trips' )
plt.title( 'Number of trips Vs hours')
```

Out[13]:
```
Text(0.5, 1.0, 'Number of trips Vs hours')
```



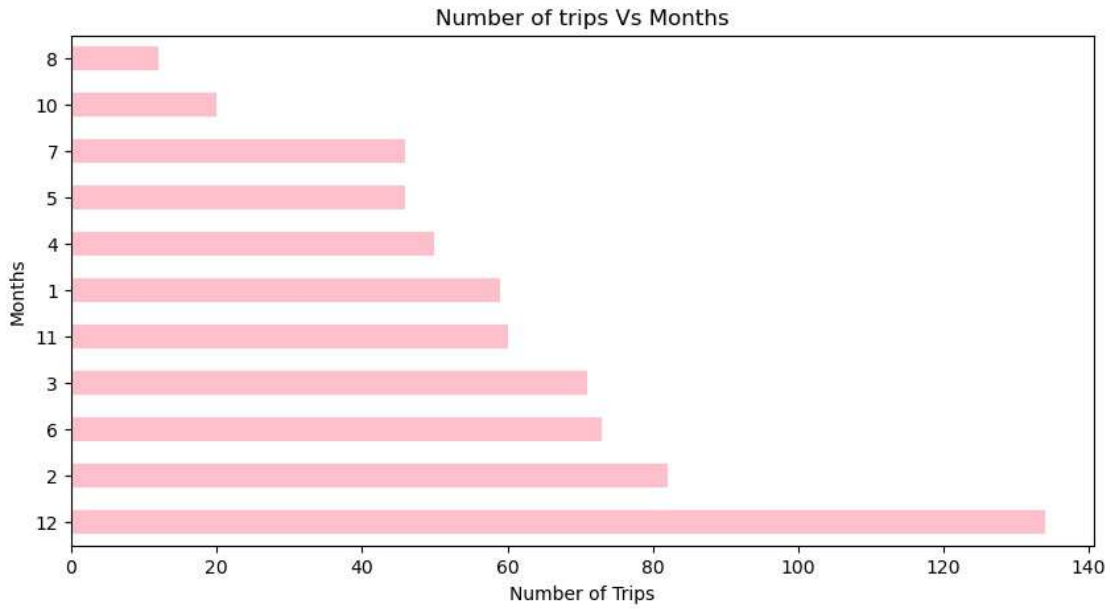
In [14]:
```
days = uber_data[ 'WEEKDAY' ].value_counts() days.plot(kind='barh',
color= 'red', figsize=(10, 5) ) plt.xlabel('Number of Trips')
plt.ylabel('Days' ) plt.title( 'Number
of trips Vs days')
```

Out[14]:
```
Text(0.5, 1.0, 'Number of trips Vs days')
```



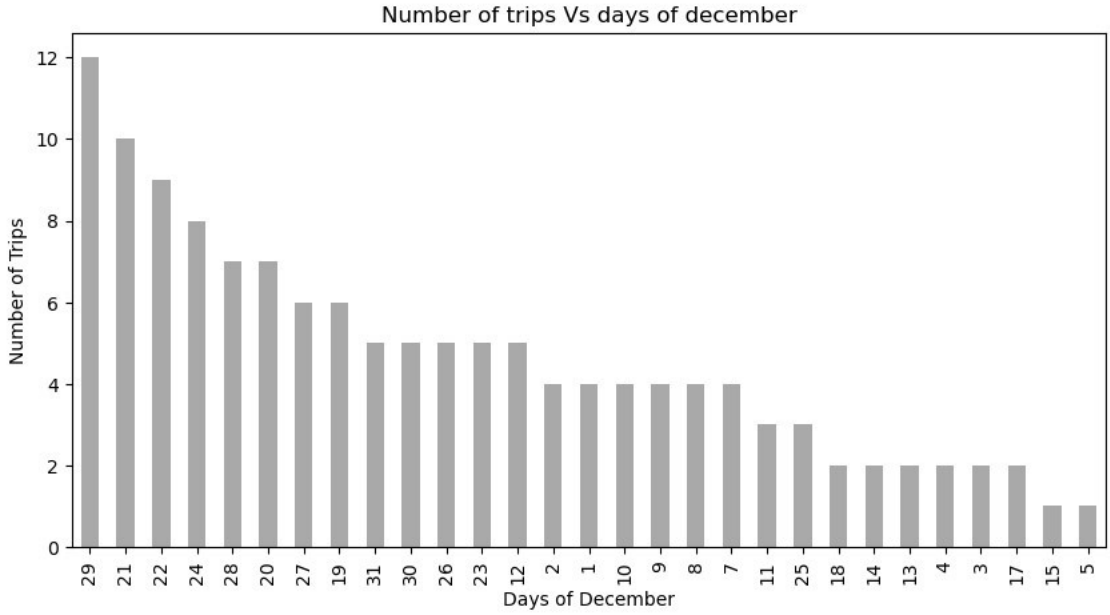
In [15]:
```
months = uber_data[ 'MONTH' ].value_counts()
months.plot(kind='barh', color= 'pink', figsize=(10, 5) )
plt.xlabel('Number of Trips') plt.ylabel('Months' )
plt.title( 'Number of trips Vs Months')
```

Out[15]:
```
Text(0.5, 1.0, 'Number of trips Vs Months')
```



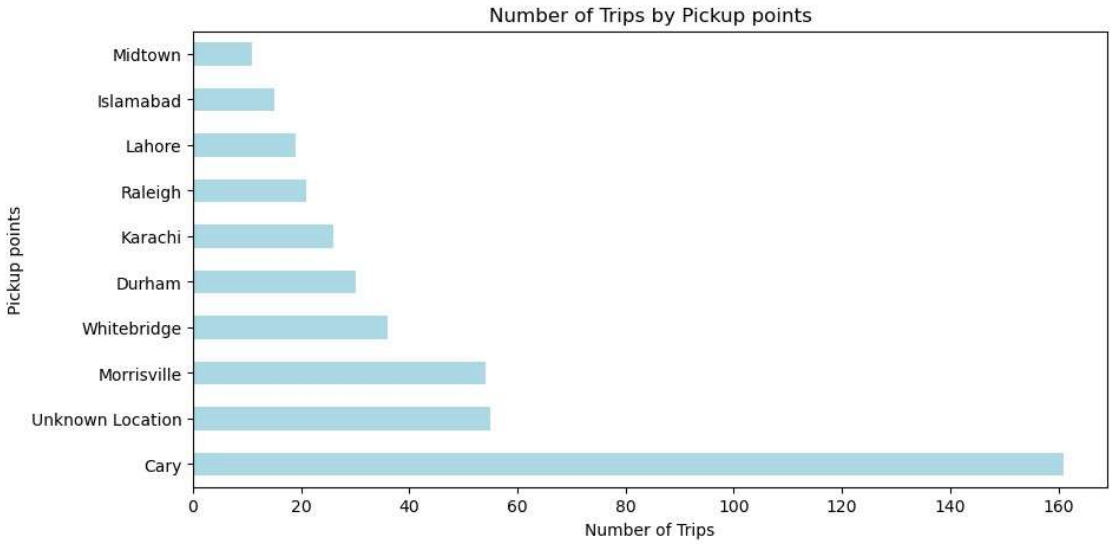

```
In [16]: months = uber_data[ 'DAY' ][uber_data['MONTH']==12].value_counts()
         months.plot(kind='bar', color= 'darkgray', figsize=(10, 5) )
         plt.xlabel('Days of December') plt.ylabel('Number of Trips' )
         plt.title( 'Number of trips Vs days of december')
```

Text(0.5, 1.0, 'Number of trips Vs days of december') Out[16]:



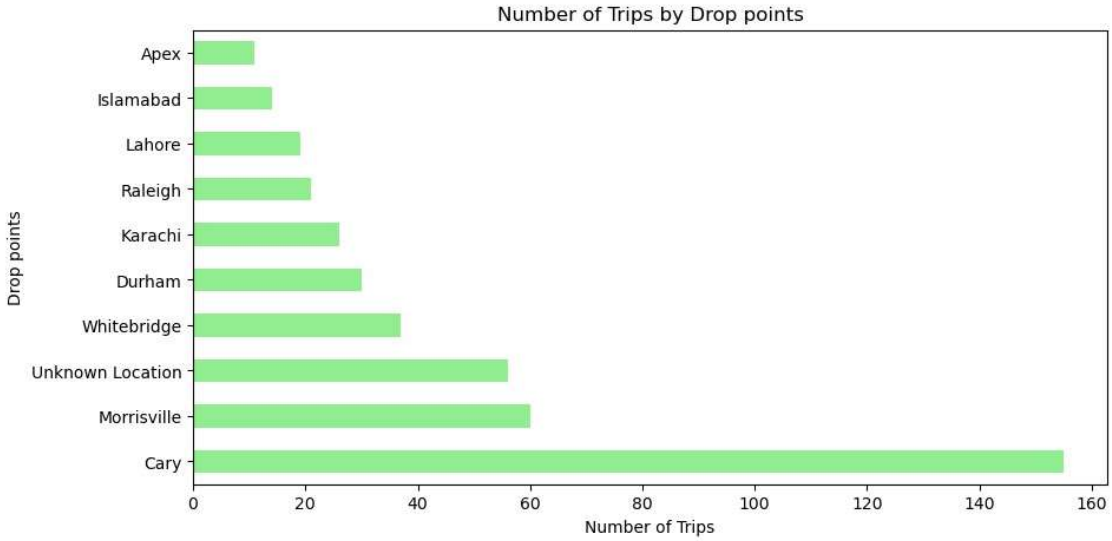

```
In [17]: pic_point = uber_data[ 'START' ].value_counts().nlargest(10)
         pic_point.plot(kind='barh', color= 'lightblue', figsize=(10, 5))
         plt.xlabel('Number of Trips') plt.ylabel('Pickup points' )
         plt.title( 'Number of Trips by Pickup points')
```

Text(0.5, 1.0, 'Number of Trips by Pickup points') Out[17]:



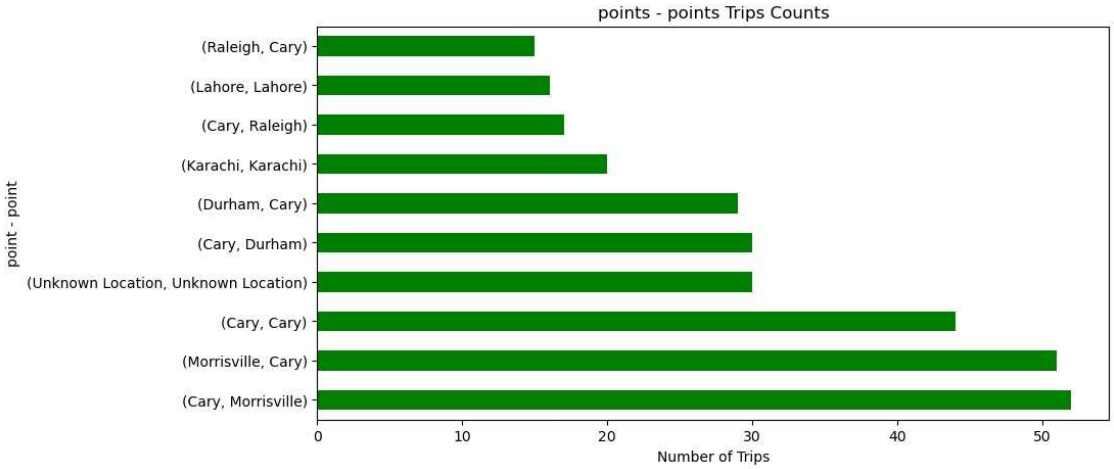
In [18]:
```
drop_point = uber_data[ 'STOP' ].value_counts().nlargest(10)
drop_point.plot(kind='barh', color= 'lightgreen', figsize=(10, 5))
plt.xlabel('Number of Trips') plt.ylabel('Drop points' ) plt.title(
'Number of Trips by Drop points')
```

Out[18]:
```
Text(0.5, 1.0, 'Number of Trips by Drop points')
```



In [19]:
```
p_p = uber_data[[ 'START', 'STOP'] ].value_counts().nlargest(10)
p_p.plot(kind='barh', color= 'green', figsize=(10, 5))
plt.xlabel('Number of Trips') plt.ylabel('point - point' )
plt.title( 'points - points Trips Counts')
```

Out[19]:
```
Text(0.5, 1.0, 'points - points Trips Counts')
```



In [20]:
```
print("\n.....Average Length of the Trip.....\n") print('Business:',
 round(uber_data[uber_data['CATEGORY'] == 'Business'].MILES.mean( print('Personal:',
 round(uber_data[uber_data['CATEGORY'] == 'Personal'].MILES.mean(
 print('Meal/Entertain:', round(uber_data[uber_data['PURPOSE'] == 'Meal/Entertain'].
```

```
.....Average Length of the Trip.....

Business: 10.97
Personal: 35.583
Meal/Entertain: 5.698
```

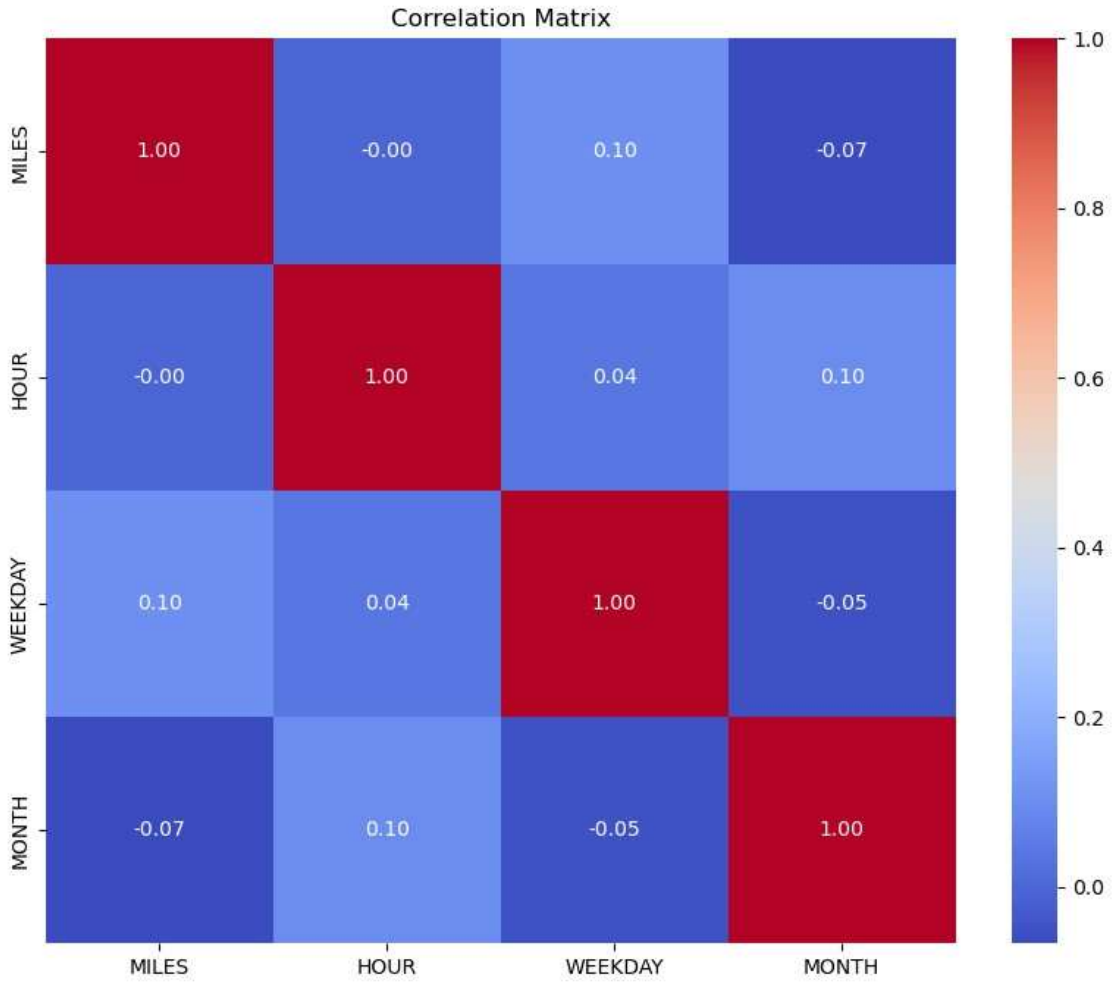
In [21]:
```
# Rename columns
uber_data = uber_data.rename(columns={
    'START_DATE*': 'START_DATE',
    'END_DATE*': 'END_DATE',
    'CATEGORY*': 'CATEGORY',
    'START*': 'START',
'STOP*': 'STOP',
    'MILES*': 'MILES',
    'PURPOSE*': 'PURPOSE'
})
```

In [22]:
```
# Data Cleaning uber_data['START_DATE'] =
 pd.to_datetime(uber_data['START_DATE']) uber_data['END_DATE'] =
 pd.to_datetime(uber_data['END_DATE']) uber_data =
 uber_data.dropna()
```

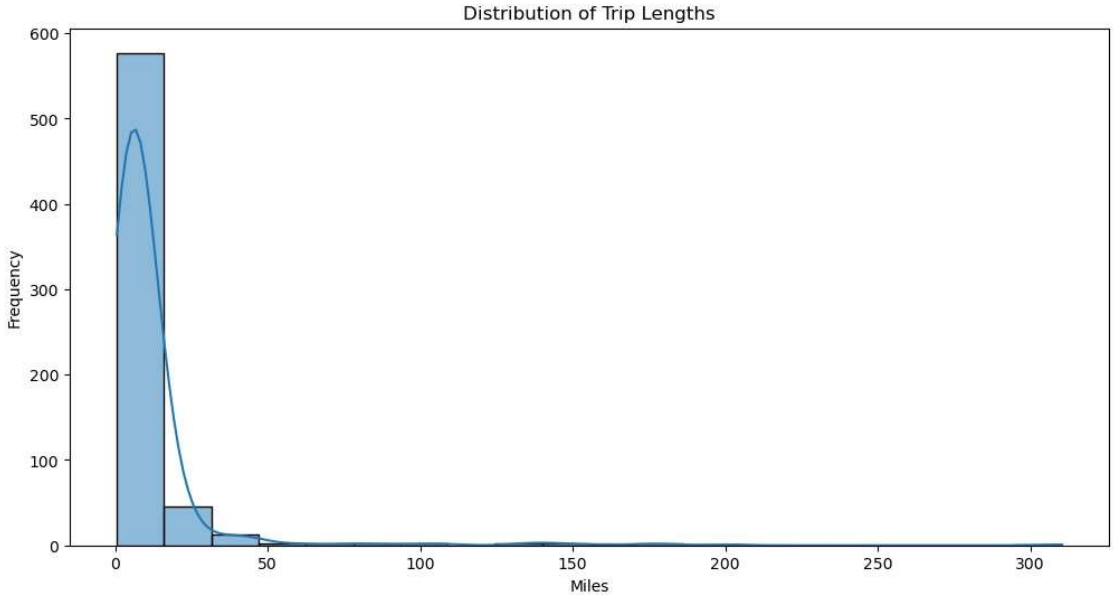
In [23]:
```
# Additional Analysis and Visualizations
```

```
# Extracting hour, weekday, month, and day of the trip
uber_data['HOUR'] = uber_data['START_DATE'].dt.hour
uber_data['WEEKDAY'] = uber_data['START_DATE'].dt.weekday
uber_data['MONTH'] = uber_data['START_DATE'].dt.month uber_data['DAY']
= uber_data['START_DATE'].dt.day
```

In [24]:
```
# Visualizing the correlation between variables
correlation_matrix = uber_data[['MILES', 'HOUR', 'WEEKDAY', 'MONTH']].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix') plt.show()
```

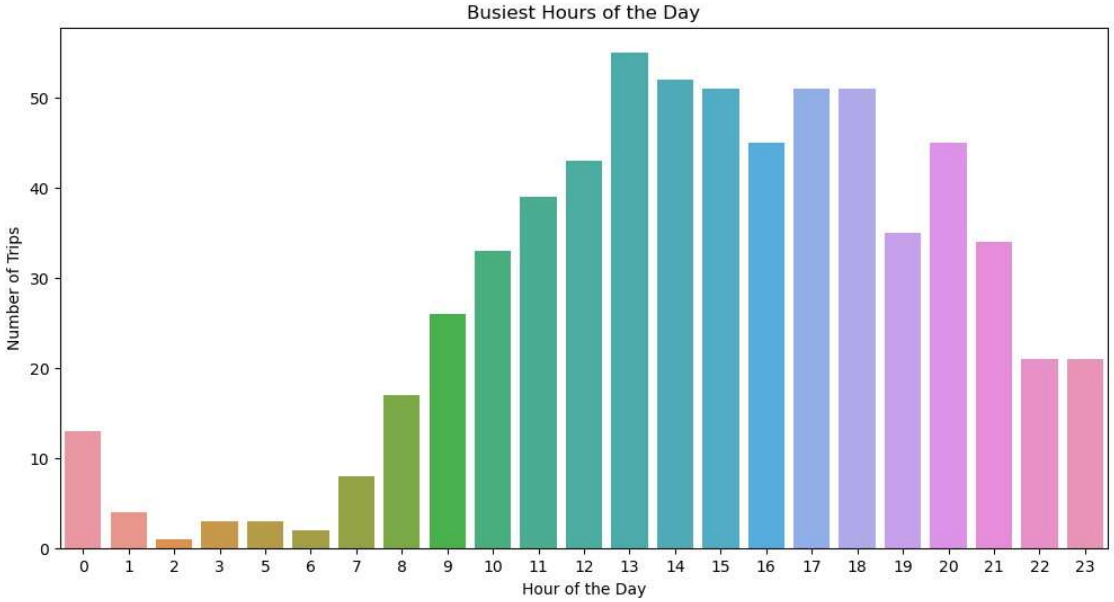



```
In [25]: # Distribution of Trip Lengths
         plt.figure(figsize=(12, 6))
         sns.histplot(uber_data['MILES'], bins=20, kde=True)
         plt.title('Distribution of Trip Lengths')
         plt.xlabel('Miles') plt.ylabel('Frequency')
         plt.show()
```



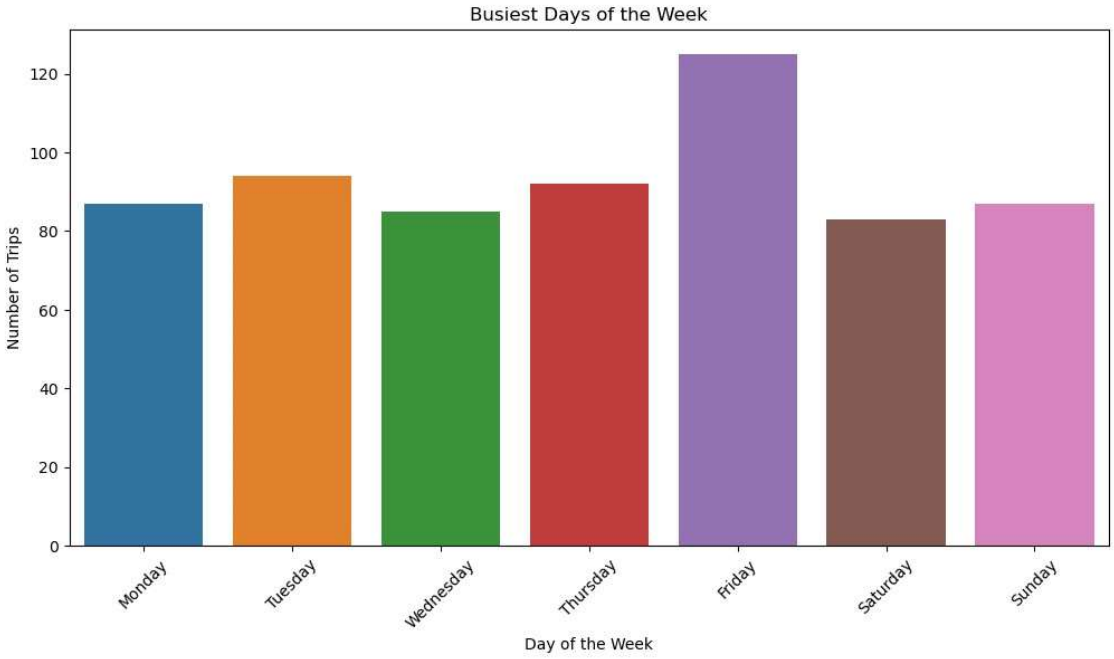
In [26]:

```
# Busiest Hours of the Day plt.figure(figsize=(12,
6))
sns.countplot(x='HOUR', data=uber_data)
plt.title('Busiest Hours of the Day')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Trips') plt.show()
```



In [27]:

```
# Busiest Days of the Week plt.figure(figsize=(12,
6))
sns.countplot(x='WEEKDAY', data=uber_data)
plt.title('Busiest Days of the Week')
plt.xlabel('Day of the Week') plt.ylabel('Number
of Trips')
plt.xticks(np.arange(7), calendar.day_name, rotation=45) plt.show()
```



1. **Data Loading and Exploration**

Description:

The data loading and exploration phase involves the initial steps of the project, where the dataset is read into a Pandas DataFrame, and a preliminary exploration is conducted to understand its structure. This phase is crucial for gaining insights into the dataset's content, identifying potential issues or anomalies, and preparing the data for further analysis.

Steps:

- Load the dataset into a Pandas DataFrame using the pd.read_csv() function or a relevant method based on the data format.
- Display the first few rows of the dataset using the head() function to get a sense of the data's structure.
- Check for missing values using isnull().sum() to assess data completeness.
- Rename columns if necessary for better clarity and consistency.

2. Data Analysis

Pattern Detection

Description:

Pattern detection involves uncovering hidden patterns in the data using various data analysis techniques. This phase may include identifying peak ride hours, popular pickup locations, or analyzing the distribution of ride durations. Statistical methods can be applied to analyze trends and relationships within the dataset.

Steps:

- Identify and analyze peak ride hours using the 'HOUR' column.
- Explore popular pickup and drop-off locations through visualizations.
- Analyze the distribution of ride durations using histograms or other relevant plots.
- Apply statistical methods to identify trends and patterns in the data.

Relationship Identification

Description:

Identify relationships between variables such as date, time, location, and ride demand. This involves exploring correlations between different columns and understanding how certain factors influence ride demand.

Steps:

- Identify and visualize relationships between variables using appropriate charts.

- Use correlation analysis to understand the degree of association between variables.
- Explore how ride demand varies with different factors, such as time of day or specific locations.

3. Data Visualization

Graphs and Charts

Description:

Data visualization is a crucial component of conveying insights effectively. Utilize tools like Matplotlib and Seaborn to create visualizations that illustrate the relationships discovered during the analysis. Visualizations enhance the interpretability of the data and make it easier for stakeholders to grasp key findings.

Steps:

- Create graphs and charts to visually represent patterns, trends, and relationships.
- Use appropriate chart types for different types of data (e.g., bar charts, histograms, scatter plots).
- Ensure proper labelling, titles, and formatting for clear communication.

Key Findings:

Based on the analysis of the Uber datasets, here are some key findings:

- The user primarily uses Uber for work-related purposes, followed by meals, errands, and customer visits.
- The distances travelled by the user are relatively short, indicating local or city travel.
- The user tends to travel during lunch hours and early evenings, possibly for client visits or lunches.
- There is regular travel throughout the week, with slightly higher trips on Fridays, which may indicate recreational activities.
- December 2016 had significantly more trips, likely due to the holiday season.
- The user frequently travels between Cary and Morrisville, suggesting a frequent commute within or between these locations.