online-food-delivery

November 29, 2024

[2]:
```
import pandas as pd

# Provide the path to your CSV file
file_path = file_path = r'C:\Users\divaa\OneDrive\Desktop\pri\FoodDB1.csv'


# Load the dataset
df = pd.read_csv(file_path)

# Display the first few rows of the dataframe
df.head()
```

[2]:

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	\
0	0xcdcd	DEHRES17DEL01	36.0	4.2	
1	0xd987	KOCRES16DEL01	21.0	4.7	
2	0x2784	PUNERES13DEL03	23.0	4.7	
3	0xc8b6	LUDHRES15DEL02	34.0	4.3	
4	0xdb64	KNPRES14DEL02	24.0	4.7	

	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	\
0	30.327968	78.046106	30.397968	
1	10.003064	76.307589	10.043064	
2	18.562450	73.916619	18.652450	
3	30.899584	75.809346	30.919584	
4	26.463504	80.372929	26.593504	

	Delivery_location_longitude	Order_Date	Time_Orderd	Time_Order_picked	\
0	78.116106	12-02-2022	21:55	22:10	
1	76.347589	13-02-2022	14:55	15:05	
2	74.006619	04-03-2022	17:30	17:40	
3	75.829346	13-02-2022	09:20	09:30	
4	80.502929	14-02-2022	19:50	20:05	

	Weather_conditions	Road_traffic_density	Vehicle_condition	Type_of_order	\
0	Fog	Jam	2	Snack	
1	Stormy	High	1	Meal	
2	Sandstorms	Medium	1	Drinks	

```
3         Sandstorms                  Low                    0        Buffet
4                Fog                  Jam                    1         Snack

  Type_of_vehicle  multiple_deliveries Festival           City  \
0      motorcycle                  3.0       No  Metropolitian
1      motorcycle                  1.0       No  Metropolitian
2         scooter                  1.0       No  Metropolitian
3      motorcycle                  0.0       No  Metropolitian
4         scooter                  1.0       No  Metropolitian

   Time_taken (min)
0                46
1                23
2                21
3                20
4                41
```

[3]:
```
# View column names
df.columns
```

[3]:
```
Index(['ID', 'Delivery_person_ID', 'Delivery_person_Age',
       'Delivery_person_Ratings', 'Restaurant_latitude',
       'Restaurant_longitude', 'Delivery_location_latitude',
       'Delivery_location_longitude', 'Order_Date', 'Time_Orderd',
       'Time_Order_picked', 'Weather_conditions', 'Road_traffic_density',
       'Vehicle_condition', 'Type_of_order', 'Type_of_vehicle',
       'multiple_deliveries', 'Festival', 'City', 'Time_taken (min)'],
      dtype='object')
```

[4]:
```
import pandas as pd

# Load the dataset
file_path = r'C:\Users\divaa\OneDrive\Desktop\pri\FoodDB1.csv'
df = pd.read_csv(file_path)

# View column names
print("Column Names: \n", df.columns)

# View data types
print("\nData Types: \n", df.dtypes)

# View dataset summary
print("\nDataset Info: \n")
df.info()

# View first few rows of the dataset
print("\nFirst 5 rows: \n")
```

```
print(df.head())
```

```
Column Names:
 Index(['ID', 'Delivery_person_ID', 'Delivery_person_Age',
       'Delivery_person_Ratings', 'Restaurant_latitude',
       'Restaurant_longitude', 'Delivery_location_latitude',
       'Delivery_location_longitude', 'Order_Date', 'Time_Orderd',
       'Time_Order_picked', 'Weather_conditions', 'Road_traffic_density',
       'Vehicle_condition', 'Type_of_order', 'Type_of_vehicle',
       'multiple_deliveries', 'Festival', 'City', 'Time_taken (min)'],
      dtype='object')

Data Types:
 ID                             object
Delivery_person_ID              object
Delivery_person_Age            float64
Delivery_person_Ratings        float64
Restaurant_latitude            float64
Restaurant_longitude           float64
Delivery_location_latitude     float64
Delivery_location_longitude    float64
Order_Date                      object
Time_Orderd                     object
Time_Order_picked               object
Weather_conditions              object
Road_traffic_density            object
Vehicle_condition                int64
Type_of_order                   object
Type_of_vehicle                 object
multiple_deliveries            float64
Festival                        object
City                            object
Time_taken (min)                 int64
dtype: object

Dataset Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45584 entries, 0 to 45583
Data columns (total 20 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   ID                           45584 non-null  object
 1   Delivery_person_ID           45584 non-null  object
 2   Delivery_person_Age          43730 non-null  float64
 3   Delivery_person_Ratings      43676 non-null  float64
 4   Restaurant_latitude          45584 non-null  float64
```

```
 5   Restaurant_longitude         45584 non-null  float64
 6   Delivery_location_latitude   45584 non-null  float64
 7   Delivery_location_longitude  45584 non-null  float64
 8   Order_Date                   45584 non-null  object
 9   Time_Orderd                  43853 non-null  object
 10  Time_Order_picked            45584 non-null  object
 11  Weather_conditions           44968 non-null  object
 12  Road_traffic_density         44983 non-null  object
 13  Vehicle_condition            45584 non-null  int64
 14  Type_of_order                45584 non-null  object
 15  Type_of_vehicle              45584 non-null  object
 16  multiple_deliveries          44591 non-null  float64
 17  Festival                     45356 non-null  object
 18  City                         44384 non-null  object
 19  Time_taken (min)             45584 non-null  int64
dtypes: float64(7), int64(2), object(11)
memory usage: 7.0+ MB

First 5 rows:

       ID Delivery_person_ID  Delivery_person_Age  Delivery_person_Ratings  \
0  0xcdcd      DEHRES17DEL01                 36.0                      4.2
1  0xd987      KOCRES16DEL01                 21.0                      4.7
2  0x2784     PUNERES13DEL03                 23.0                      4.7
3  0xc8b6     LUDHRES15DEL02                 34.0                      4.3
4  0xdb64      KNPRES14DEL02                 24.0                      4.7

   Restaurant_latitude  Restaurant_longitude  Delivery_location_latitude  \
0            30.327968             78.046106                   30.397968
1            10.003064             76.307589                   10.043064
2            18.562450             73.916619                   18.652450
3            30.899584             75.809346                   30.919584
4            26.463504             80.372929                   26.593504

   Delivery_location_longitude  Order_Date Time_Orderd Time_Order_picked  \
0                    78.116106  12-02-2022       21:55             22:10
1                    76.347589  13-02-2022       14:55             15:05
2                    74.006619  04-03-2022       17:30             17:40
3                    75.829346  13-02-2022       09:20             09:30
4                    80.502929  14-02-2022       19:50             20:05

  Weather_conditions Road_traffic_density  Vehicle_condition Type_of_order  \
0                Fog                  Jam                  2         Snack
1             Stormy                 High                  1          Meal
2         Sandstorms               Medium                  1        Drinks
3         Sandstorms                  Low                  0        Buffet
4                Fog                  Jam                  1         Snack
```

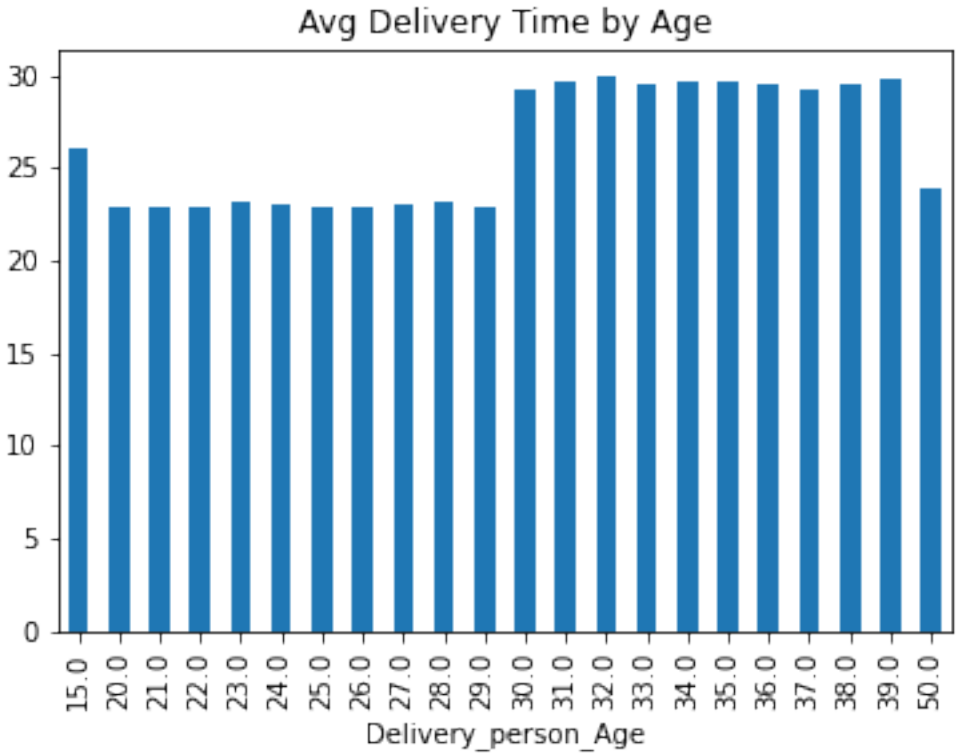
```
  Type_of_vehicle  multiple_deliveries Festival           City  \
0      motorcycle                  3.0       No  Metropolitian
1      motorcycle                  1.0       No  Metropolitian
2         scooter                  1.0       No  Metropolitian
3      motorcycle                  0.0       No  Metropolitian
4         scooter                  1.0       No  Metropolitian

   Time_taken (min)
0                46
1                23
2                21
3                20
4                41
```

[5]:
```
# Get descriptive statistics for numerical columns
df.describe()

# Check for missing values
df.isnull().sum()
```

[5]:
```
ID                             0
Delivery_person_ID             0
Delivery_person_Age         1854
Delivery_person_Ratings     1908
Restaurant_latitude            0
Restaurant_longitude           0
Delivery_location_latitude     0
Delivery_location_longitude    0
Order_Date                     0
Time_Orderd                 1731
Time_Order_picked              0
Weather_conditions           616
Road_traffic_density         601
Vehicle_condition              0
Type_of_order                  0
Type_of_vehicle                0
multiple_deliveries          993
Festival                     228
City                        1200
Time_taken (min)               0
dtype: int64
```

[69]:
```
# Average delivery time by delivery person's age
df.groupby('Delivery_person_Age')['Time_taken (min)'].mean().plot(kind='bar',␣
 ↪title='Avg Delivery Time by Age')
```

[69]: <AxesSubplot:title={'center':'Avg Delivery Time by Age'},
xlabel='Delivery_person_Age'>



[70]:
```
# Minimum and Maximum Delivery Person Age
min_age = df['Delivery_person_Age'].min()
max_age = df['Delivery_person_Age'].max()

print("Minimum Age:", min_age)
print("Maximum Age:", max_age)
```

```
Minimum Age: 15.0
Maximum Age: 50.0
```

[71]:
```
# Print average delivery time by delivery person's age (in numbers)
avg_delivery_time_by_age
```

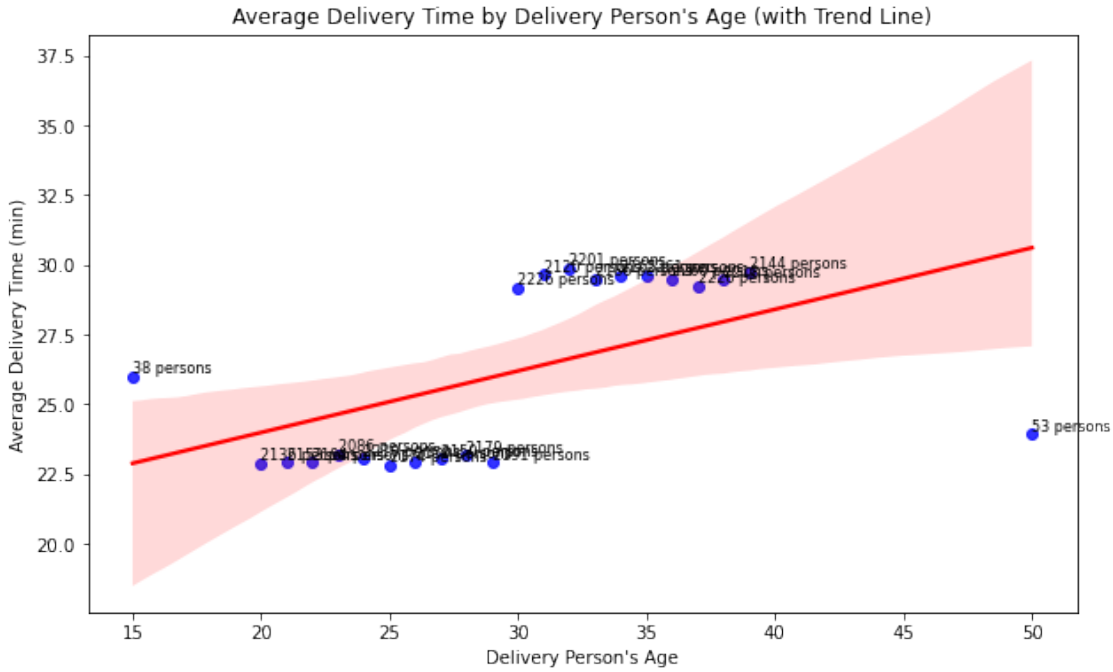
[71]:
```
Delivery_person_Age
15.0    26.000000
20.0    22.898876
21.0    22.908500
22.0    22.927985
23.0    23.196069
24.0    23.059729
```

```
25.0    22.819227
26.0    22.948124
27.0    23.053023
28.0    23.166590
29.0    22.916933
30.0    29.170710
31.0    29.659906
32.0    29.885961
33.0    29.489936
34.0    29.605543
35.0    29.607253
36.0    29.500885
37.0    29.250674
38.0    29.451758
39.0    29.739739
50.0    23.943396
Name: Time_taken (min), dtype: float64
```

[72]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Calculate the average delivery time by delivery person's age
avg_delivery_time_by_age = df.groupby('Delivery_person_Age')['Time_taken␣
 ↪(min)'].mean()

# Count the number of delivery persons in each age group
person_count_by_age = df['Delivery_person_Age'].value_counts().sort_index()

# Identify any missing age groups (if there are gaps in the ages)
all_possible_ages = range(int(df['Delivery_person_Age'].min()),␣
 ↪int(df['Delivery_person_Age'].max()) + 1)
missing_ages = [age for age in all_possible_ages if age not in␣
 ↪person_count_by_age]

# Output the missing ages (if any)
if missing_ages:
    print("Missing age groups that are affecting the analysis:", missing_ages)
else:
    print("No missing age groups affecting the analysis.")

# Display the number of persons in each age group
print("\nNumber of persons in each age group:")
print(person_count_by_age)

# Now, let's visualize using a scatter plot with a regression line and annotate␣
 ↪the number of persons for each age group
plt.figure(figsize=(10, 6))
```

```
sns.regplot(x=avg_delivery_time_by_age.index, y=avg_delivery_time_by_age.
 ↪values, scatter_kws={'color':'blue'}, line_kws={'color':'red'})

# Annotating the number of persons for each age group
for age, avg_time, count in zip(avg_delivery_time_by_age.index,␣
 ↪avg_delivery_time_by_age.values, person_count_by_age):
    plt.text(age, avg_time, f'{count} persons', horizontalalignment='left',␣
 ↪verticalalignment='bottom', fontsize=8)

# Set title and labels
plt.title('Average Delivery Time by Delivery Person\'s Age (with Trend Line)')
plt.xlabel('Delivery Person\'s Age')
plt.ylabel('Average Delivery Time (min)')
plt.show()
```

```
Missing age groups that are affecting the analysis: [16, 17, 18, 19, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49]

Number of persons in each age group:
15.0      38
20.0    2136
21.0    2153
22.0    2194
23.0    2086
24.0    2210
25.0    2174
26.0    2159
27.0    2150
28.0    2179
29.0    2191
30.0    2226
31.0    2120
32.0    2201
33.0    2186
34.0    2165
35.0    2261
36.0    2260
37.0    2226
38.0    2218
39.0    2144
50.0      53
Name: Delivery_person_Age, dtype: int64
```



[73]:
```
# Step 1: Identify rows where 'Delivery_person_Age' is missing
missing_age_rows = df[df['Delivery_person_Age'].isnull()]

# Step 2: Calculate the average delivery time for rows where␣
 ↪'Delivery_person_Age' is missing
avg_time_missing_age = missing_age_rows['Time_taken (min)'].mean()

print(f"Average delivery time for rows with missing 'Delivery_person_Age':␣
 ↪{avg_time_missing_age:.2f} minutes")

# Step 3: Calculate overall average delivery time for all delivery persons␣
 ↪(including those with missing age)
overall_avg_time = df['Time_taken (min)'].mean()

print(f"Overall average delivery time (including missing ages):␣
 ↪{overall_avg_time:.2f} minutes")

# Step 4: Calculate average delivery time for rows where 'Delivery_person_Age'␣
 ↪is not missing
avg_time_with_age = df[df['Delivery_person_Age'].notnull()]['Time_taken (min)'].
 ↪mean()

print(f"Average delivery time for rows with non-missing 'Delivery_person_Age':␣
 ↪{avg_time_with_age:.2f} minutes")
```

```
# Step 5: Display how many rows have missing 'Delivery_person_Age'
missing_age_count = missing_age_rows.shape[0]
print(f"\nTotal number of missing 'Delivery_person_Age': {missing_age_count}")
```

```
Average delivery time for rows with missing 'Delivery_person_Age': 26.46 minutes
Overall average delivery time (including missing ages): 26.29 minutes
Average delivery time for rows with non-missing 'Delivery_person_Age': 26.29
minutes

Total number of missing 'Delivery_person_Age': 1854
```

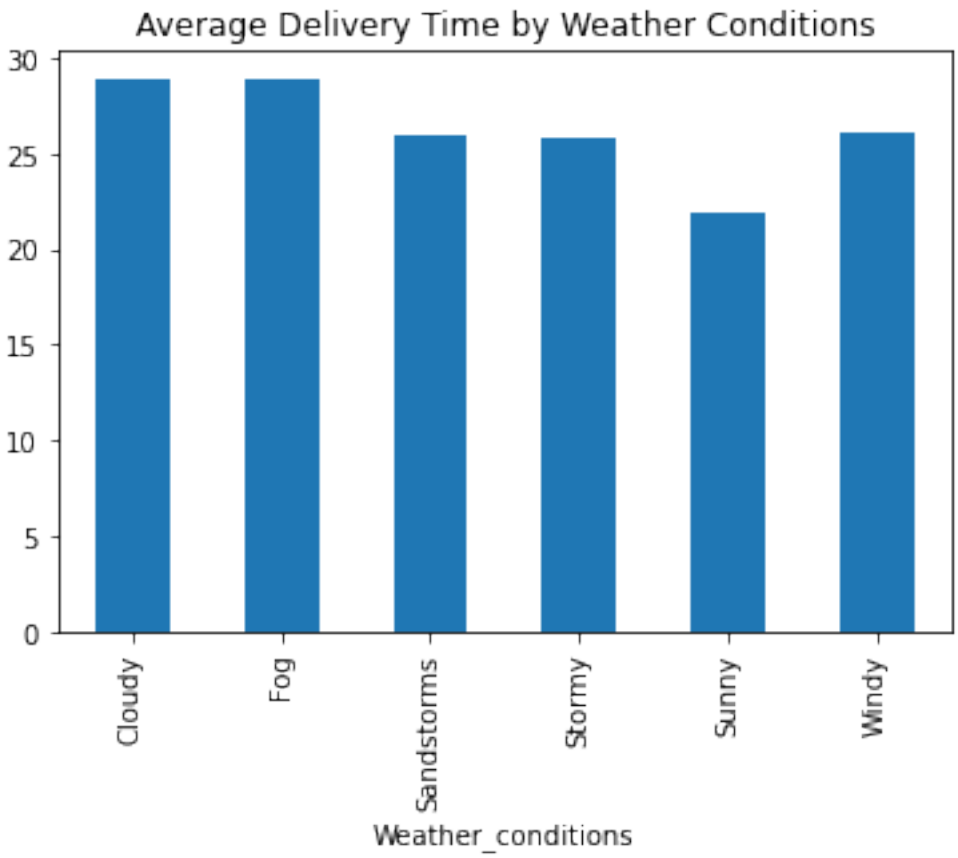
Minimal Impact of Missing Age on Delivery Time, The difference between the average delivery time for missing ages (26.46 minutes) and the overall average (26.29 minutes) is quite small (0.17 minutes). This suggests that the missing data in the Delivery_person_Age column does not have a significant impact on the average delivery time.

[74]:
```
# Step 1: Remove rows where 'Delivery_person_Age' or 'Time_taken (min)' are␣
 ↪missing
df_cleaned = df.dropna(subset=['Delivery_person_Age', 'Time_taken (min)'])

# Step 2: Find the minimum delivery time and the respective age
min_time = df_cleaned['Time_taken (min)'].min()
min_time_age = df_cleaned[df_cleaned['Time_taken (min)'] ==␣
 ↪min_time]['Delivery_person_Age'].values[0]

# Step 3: Find the maximum delivery time and the respective age
max_time = df_cleaned['Time_taken (min)'].max()
max_time_age = df_cleaned[df_cleaned['Time_taken (min)'] ==␣
 ↪max_time]['Delivery_person_Age'].values[0]

# Step 4: Print the results
print(f"Minimum delivery time: {min_time} minutes, by person aged␣
 ↪{min_time_age}")
print(f"Maximum delivery time: {max_time} minutes, by person aged␣
 ↪{max_time_age}")
```

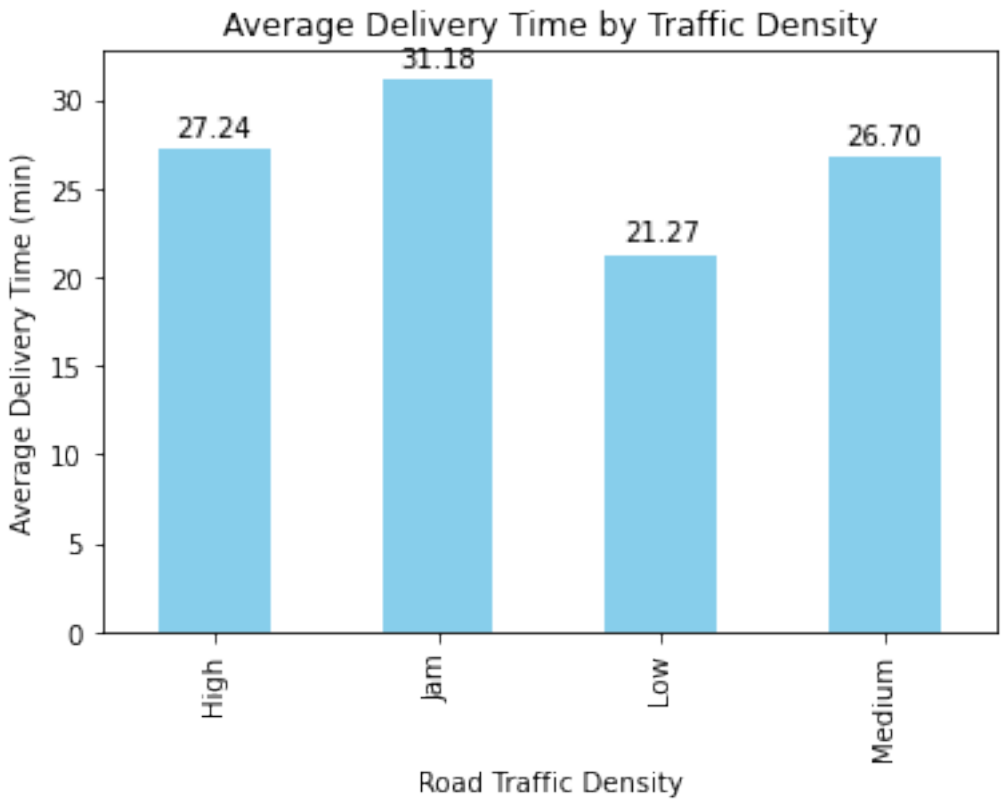
```
Minimum delivery time: 10 minutes, by person aged 20.0
Maximum delivery time: 54 minutes, by person aged 38.0
```

[75]:
```
df.groupby('Weather_conditions')['Time_taken (min)'].mean().plot(kind='bar',␣
 ↪title='Average Delivery Time by Weather Conditions')
```

[75]:
```
<AxesSubplot:title={'center':'Average Delivery Time by Weather Conditions'},
xlabel='Weather_conditions'>
```



[76]:
```
import matplotlib.pyplot as plt

# Calculate the average delivery time by traffic density
traffic_data = df.groupby('Road_traffic_density')['Time_taken (min)'].mean()

# Create the bar plot
ax = traffic_data.plot(kind='bar', title='Average Delivery Time by Traffic␣
 ↪Density', color='skyblue')

# Add labels for data values on top of the bars
for i in ax.containers:
    ax.bar_label(i, fmt='%.2f', label_type='edge', padding=3)

# Display the plot
plt.ylabel('Average Delivery Time (min)')
plt.xlabel('Road Traffic Density')
plt.show()
```



[77]:
```
# Count orders by user (Delivery_person_ID)
order_frequency = df['Delivery_person_ID'].value_counts()

# Classify users as repeat or one-time users
df['user_type'] = df['Delivery_person_ID'].apply(lambda x: 'Repeat' if␣
 ↪order_frequency[x] > 1 else 'One-time')

# Summary of user types
user_type_summary = df['user_type'].value_counts()
print(user_type_summary)
```

```
Repeat    45584
Name: user_type, dtype: int64
```

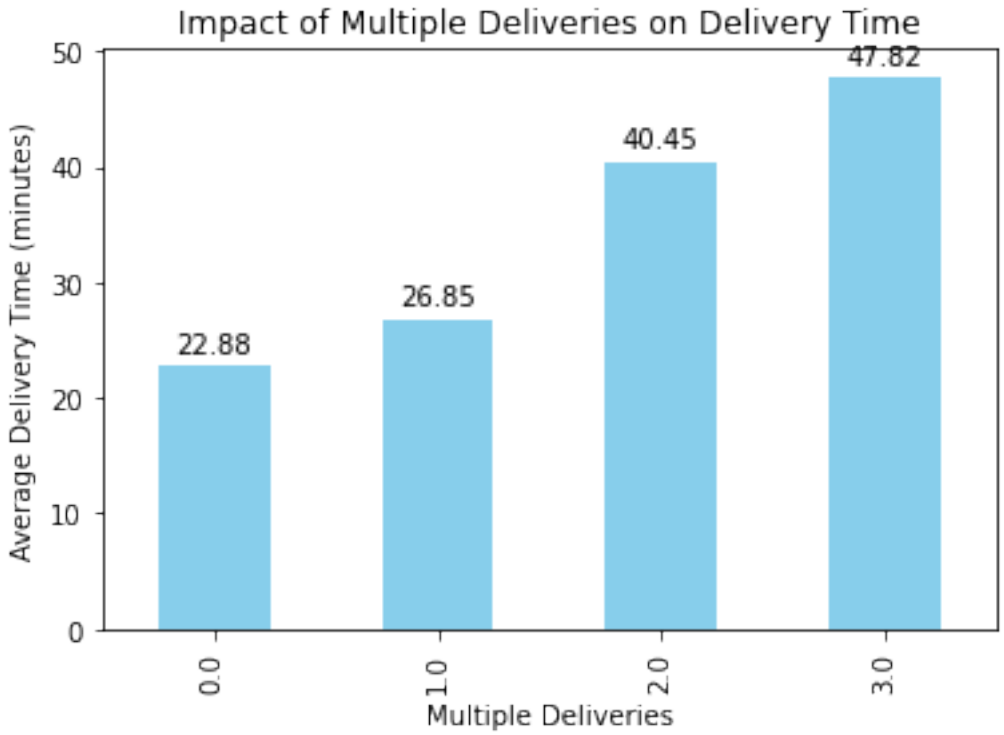
[78]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the figure size for better clarity
plt.figure(figsize=(10, 6))

# Create a scatter plot for Customer Ratings vs. Delivery Time
```

```
sns.scatterplot(x='Time_taken (min)', y='Delivery_person_Ratings', data=df,␣
 ↪hue='Road_traffic_density', palette='coolwarm')

# Add labels and title for the plot
plt.title('Customer Ratings vs. Delivery Time', fontsize=16)
plt.xlabel('Delivery Time (minutes)', fontsize=12)
plt.ylabel('Customer Ratings', fontsize=12)

# Display the plot
plt.show()
```

Ratings stay relatively high for delivery times up to 30 minutes, across all traffic densities. Delivery times beyond 30 minutes show a clear drop in customer ratings, particularly under low and medium traffic conditions. Traffic jams seem to cause longer delivery times, but customer ratings remain more forgiving, likely because customers expect delays in heavy traffic.

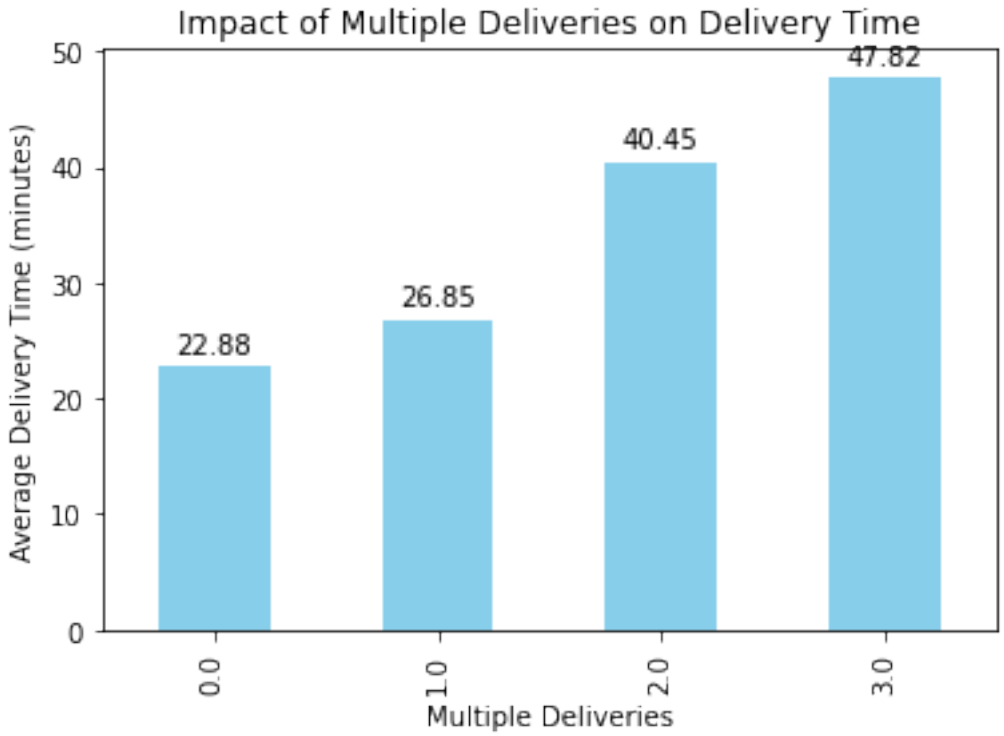
[79]:
```
import matplotlib.pyplot as plt

# Calculate the average delivery time for each number of multiple deliveries
delivery_data = df.groupby('multiple_deliveries')['Time_taken (min)'].mean()

# Create the bar plot
ax = delivery_data.plot(kind='bar', title='Impact of Multiple Deliveries on␣
 ↪Delivery Time', color='skyblue')
```

```
# Add data labels on top of each bar
for i in ax.containers:
    ax.bar_label(i, fmt='%.2f', label_type='edge', padding=3)

# Set labels for the axes
plt.ylabel('Average Delivery Time (minutes)')
plt.xlabel('Multiple Deliveries')

# Show the plot
plt.show()
```



[80]:
```
import matplotlib.pyplot as plt

# Calculate the average delivery time for each number of multiple deliveries
delivery_data = df.groupby('multiple_deliveries')['Time_taken (min)'].mean()

# Create the bar plot
ax = delivery_data.plot(kind='bar', title='Impact of Multiple Deliveries on␣
 ↪Delivery Time', color='skyblue')

# Add data labels on top of each bar
for i in ax.containers:
```

```
    ax.bar_label(i, fmt='%.2f', label_type='edge', padding=3)

# Set labels for the axes
plt.ylabel('Average Delivery Time (minutes)')
plt.xlabel('Multiple Deliveries')

# Show the plot
plt.show()
```



[81]:
```
import pandas as pd

# Convert 'Time_Orderd' and 'Time_Order_picked' to datetime
df['Time_Orderd'] = pd.to_datetime(df['Time_Orderd'], format='%H:%M')
df['Time_Order_picked'] = pd.to_datetime(df['Time_Order_picked'], format='%H:
 ↪%M')

# Calculate the time difference in minutes
df['Time_difference'] = (df['Time_Order_picked'] - df['Time_Orderd']).dt.
 ↪total_seconds() / 60

# Calculate the minimum, maximum, and average time difference
min_time_difference = df['Time_difference'].min()
max_time_difference = df['Time_difference'].max()
```

```
avg_time_difference = df['Time_difference'].mean()

# Display the results
print(f"Minimum time between order and pickup: {min_time_difference} minutes")
print(f"Maximum time between order and pickup: {max_time_difference} minutes")
print(f"Average time between order and pickup: {avg_time_difference:.2f}␣
 ↪minutes")
```

```
Minimum time between order and pickup: -1435.0 minutes
Maximum time between order and pickup: 15.0 minutes
Average time between order and pickup: -17.30 minutes
```

Order Frequency:

[82]:
```
# Count orders by user (Delivery_person_ID)
order_frequency = df['Delivery_person_ID'].value_counts()

# Classify users as repeat or one-time users
df['user_type'] = df['Delivery_person_ID'].apply(lambda x: 'Repeat' if␣
 ↪order_frequency[x] > 1 else 'One-time')

# Summary of user types
user_type_summary = df['user_type'].value_counts()
print(user_type_summary)
```

```
Repeat    45584
Name: user_type, dtype: int64
```

[89]:
```
# Convert 'Order_Date' to datetime format
df['Order_Date'] = pd.to_datetime(df['Order_Date'], format='%d-%m-%Y')

# Convert 'Time_Orderd' to datetime, handle both HH:MM and HH:MM:SS formats
df['Time_Orderd'] = pd.to_datetime(df['Time_Orderd'], format='%H:%M:%S',␣
 ↪errors='coerce').dt.time

# Extract the hour directly from 'Time_Orderd'
df['order_hour'] = df['Time_Orderd'].apply(lambda x: x.hour if pd.notnull(x)␣
 ↪else None)

# Extract the day of the week from 'Order_Date'
df['order_day'] = df['Order_Date'].dt.day_name()

# Find the peak order hours (number of orders per hour)
peak_hours = df['order_hour'].value_counts().sort_index()

# Find the peak order days (number of orders per day of the week)
peak_days = df['order_day'].value_counts()
```

```
# Display the peak hours and peak days
print("Peak Hours:\n", peak_hours)
print("Peak Days:\n", peak_days)
```

```
Peak Hours:
 0.0      430
8.0     1817
9.0     1947
10.0    1991
11.0    1961
12.0     892
13.0     783
14.0     791
15.0     873
16.0     709
17.0    4277
18.0    4479
19.0    4593
20.0    4538
21.0    4685
22.0    4576
23.0    4511
Name: order_hour, dtype: int64
Peak Days:
 Wednesday    7093
Friday       7028
Tuesday      6374
Thursday     6348
Saturday     6287
Sunday       6248
Monday       6206
Name: order_day, dtype: int64
```

[96]:
```
# Extract distinct types of food from 'Type_of_order' column
distinct_food_types = df['Type_of_order'].unique()

# Display the distinct food types
print("Distinct types of food ordered:\n", distinct_food_types)
```

```
Distinct types of food ordered:
 ['Snack' 'Meal' 'Drinks' 'Buffet']
```

[97]:
```
# Convert 'Order_Date' to datetime format, assuming 'Order_Date' is already a␣
  ↪string
df['Order_Date'] = pd.to_datetime(df['Order_Date'], format='%d-%m-%Y')

# Extract the day of the week (0 = Monday, 6 = Sunday)
```

```
df['day_of_week'] = df['Order_Date'].dt.dayofweek

# Filter rows where the order is placed on a weekend (5 = Saturday, 6 = Sunday)
weekend_orders = df[df['day_of_week'].isin([5, 6])]

# Get the count of each type of order on weekends
weekend_order_types = weekend_orders['Type_of_order'].value_counts()

# Display the results
print("Types of food ordered on weekends:\n", weekend_order_types)
```

```
Types of food ordered on weekends:
 Snack     3178
Drinks    3175
Meal      3154
Buffet    3028
Name: Type_of_order, dtype: int64
```

[95]:
```
# Count the occurrences of each type of food ordered
food_counts = df['Type_of_order'].value_counts()

# Display the type of food that was ordered the most
most_ordered_food = food_counts.idxmax()
most_ordered_count = food_counts.max()

print(f"The most ordered type of food is: '{most_ordered_food}' with␣
 ↪{most_ordered_count} orders.")
```

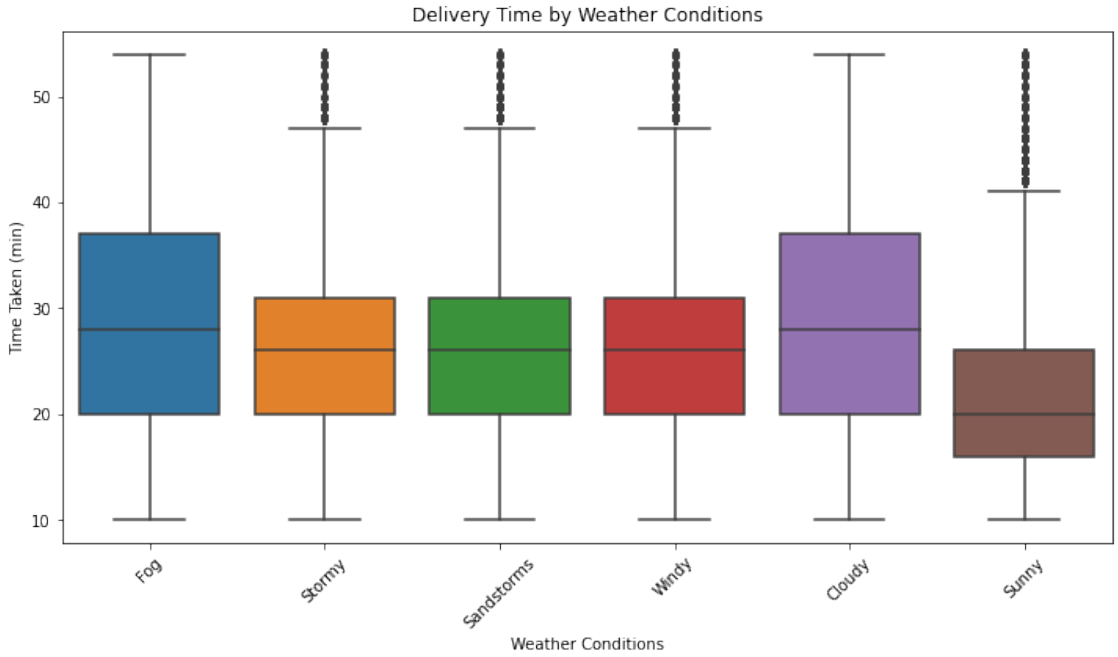
```
The most ordered type of food is: 'Snack' with 11530 orders.
```

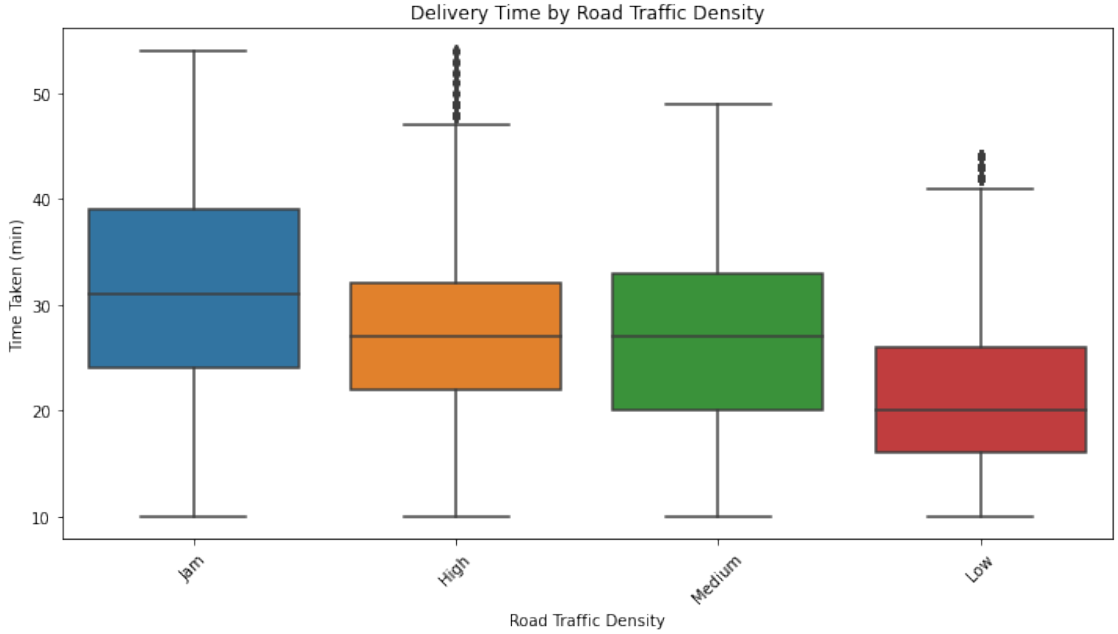
[98]:
```
# Descriptive statistics for delivery time based on weather conditions
weather_stats = df.groupby('Weather_conditions')['Time_taken (min)'].describe()
print("Delivery Time Statistics by Weather Conditions:")
print(weather_stats)

# Descriptive statistics for delivery time based on road traffic density
traffic_stats = df.groupby('Road_traffic_density')['Time_taken (min)'].
 ↪describe()
print("\nDelivery Time Statistics by Road Traffic Density:")
print(traffic_stats)
```

```
Delivery Time Statistics by Weather Conditions:
                     count       mean        std   min   25%   50%   75%   max
Weather_conditions
Cloudy              7533.0  28.917164  10.086234  10.0  20.0  28.0  37.0  54.0
Fog                 7653.0  28.914674  10.129321  10.0  20.0  28.0  37.0  54.0
Sandstorms          7494.0  25.875500   8.619009  10.0  20.0  26.0  31.0  54.0
Stormy              7584.0  25.868803   8.473481  10.0  20.0  26.0  31.0  54.0
```

```
Sunny                7282.0  21.856770   8.328310  10.0  16.0  20.0  26.0  54.0
Windy                7422.0  26.118836   8.615702  10.0  20.0  26.0  31.0  54.0

Delivery Time Statistics by Road Traffic Density:
                         count       mean       std   min   25%   50%   75%  \
Road_traffic_density
High                    4423.0  27.240109  8.397667  10.0  22.0  27.0  32.0
Jam                    14139.0  31.176038  9.941898  10.0  24.0  31.0  39.0
Low                    15476.0  21.266671  6.799175  10.0  16.0  20.0  26.0
Medium                 10945.0  26.699680  8.560638  10.0  20.0  27.0  33.0

                         max
Road_traffic_density
High                    54.0
Jam                     54.0
Low                     44.0
Medium                  49.0
```

[8]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Box plot for Delivery Time by Weather Conditions
plt.figure(figsize=(12, 6))
sns.boxplot(x='Weather_conditions', y='Time_taken (min)', data=df)
plt.title('Delivery Time by Weather Conditions')
plt.ylabel('Time Taken (min)')
plt.xlabel('Weather Conditions')
plt.xticks(rotation=45)
plt.show()

# Box plot for Delivery Time by Road Traffic Density
plt.figure(figsize=(12, 6))
sns.boxplot(x='Road_traffic_density', y='Time_taken (min)', data=df)
plt.title('Delivery Time by Road Traffic Density')
plt.ylabel('Time Taken (min)')
plt.xlabel('Road Traffic Density')
plt.xticks(rotation=45)
plt.show()
```





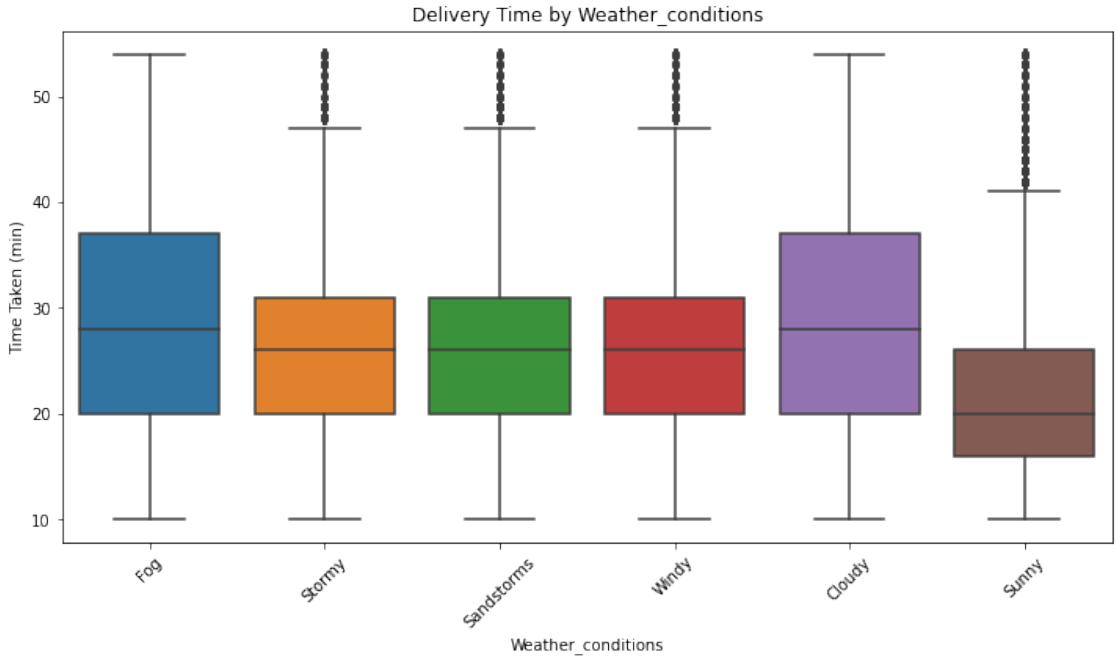
[100]:
```
# Descriptive statistics for each factor
factors = ['Weather_conditions', 'Road_traffic_density', 'Vehicle_condition',␣
 ↪'multiple_deliveries', 'Festival', 'City']
```

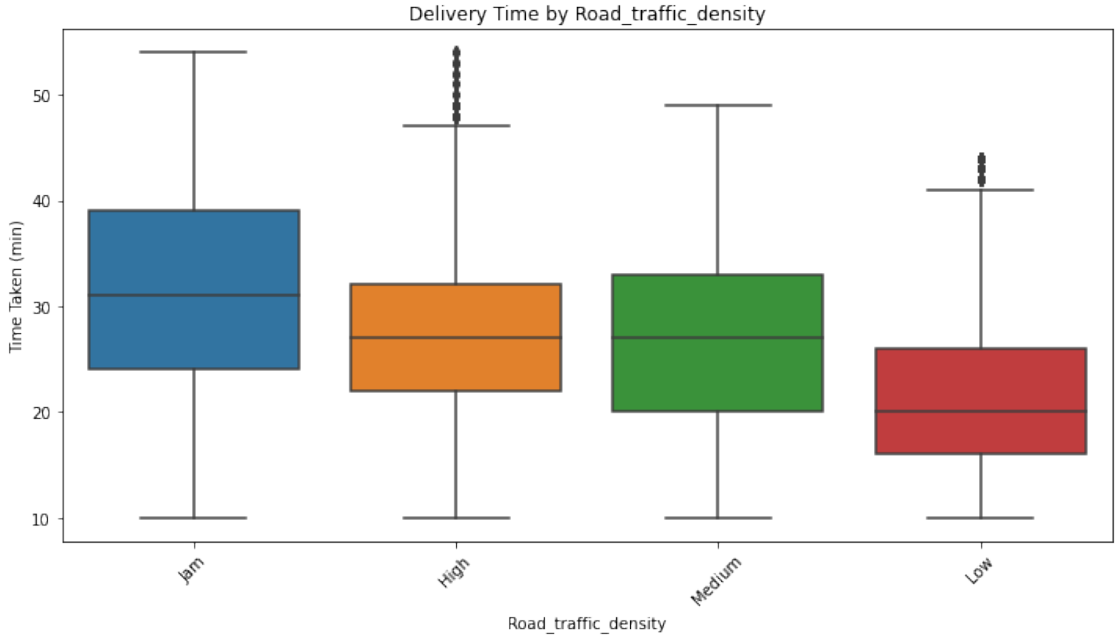
```python
for factor in factors:
    print(f"Delivery Time Statistics by {factor}:")
    print(df.groupby(factor)['Time_taken (min)'].describe(), "\n")
```

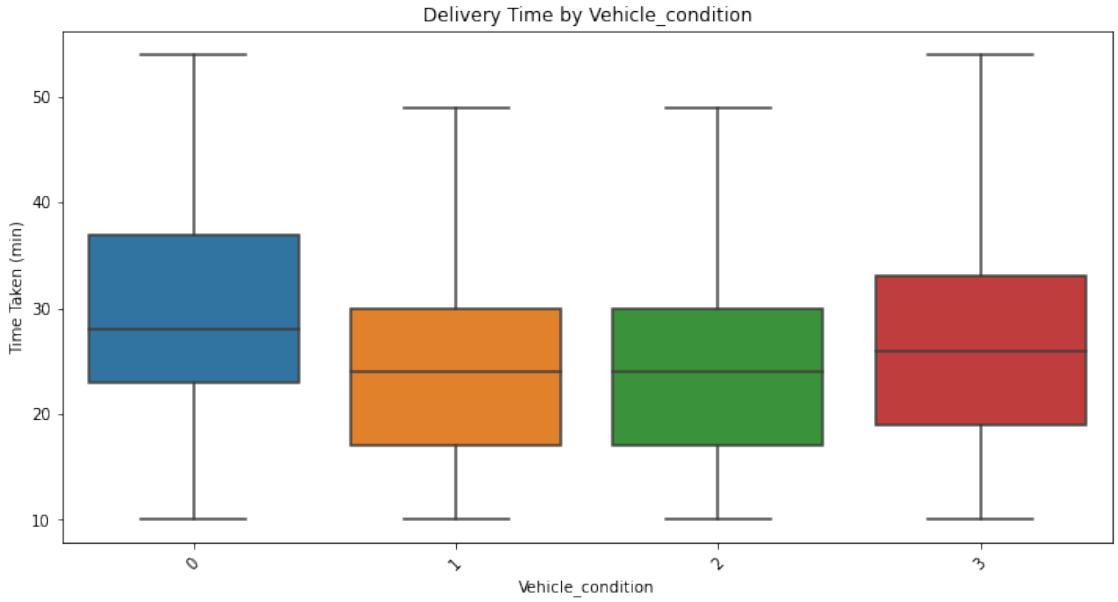
```
Delivery Time Statistics by Weather_conditions:
                     count       mean        std   min   25%   50%   75%   max
Weather_conditions
Cloudy              7533.0  28.917164  10.086234  10.0  20.0  28.0  37.0  54.0
Fog                 7653.0  28.914674  10.129321  10.0  20.0  28.0  37.0  54.0
Sandstorms          7494.0  25.875500   8.619009  10.0  20.0  26.0  31.0  54.0
Stormy              7584.0  25.868803   8.473481  10.0  20.0  26.0  31.0  54.0
Sunny               7282.0  21.856770   8.328310  10.0  16.0  20.0  26.0  54.0
Windy               7422.0  26.118836   8.615702  10.0  20.0  26.0  31.0  54.0

Delivery Time Statistics by Road_traffic_density:
                       count       mean       std   min   25%   50%   75%  \
Road_traffic_density
High                  4423.0  27.240109  8.397667  10.0  22.0  27.0  32.0
Jam                  14139.0  31.176038  9.941898  10.0  24.0  31.0  39.0
Low                  15476.0  21.266671  6.799175  10.0  16.0  20.0  26.0
Medium               10945.0  26.699680  8.560638  10.0  20.0  27.0  33.0

                      max
Road_traffic_density
High                 54.0
Jam                  54.0
Low                  44.0
Medium               49.0

Delivery Time Statistics by Vehicle_condition:
                     count       mean       std   min   25%   50%   75%   max
Vehicle_condition
0                  15005.0  30.073109  9.597452  10.0  23.0  28.0  37.0  54.0
1                  15028.0  24.353673  8.718856  10.0  17.0  24.0  30.0  49.0
2                  15031.0  24.454394  8.641767  10.0  17.0  24.0  30.0  49.0
3                    520.0  26.492308  9.383703  10.0  19.0  26.0  33.0  54.0

Delivery Time Statistics by multiple_deliveries:
                       count       mean       std   min   25%   50%   75%  \
multiple_deliveries
0.0                  14094.0  22.876188  8.767457  10.0  16.0  22.0  28.0
1.0                  28151.0  26.854925  8.537959  10.0  20.0  26.0  32.0
2.0                   1985.0  40.454912  4.921368  31.0  37.0  40.0  44.0
3.0                    361.0  47.819945  3.481417  42.0  45.0  48.0  50.0

                      max
multiple_deliveries
```

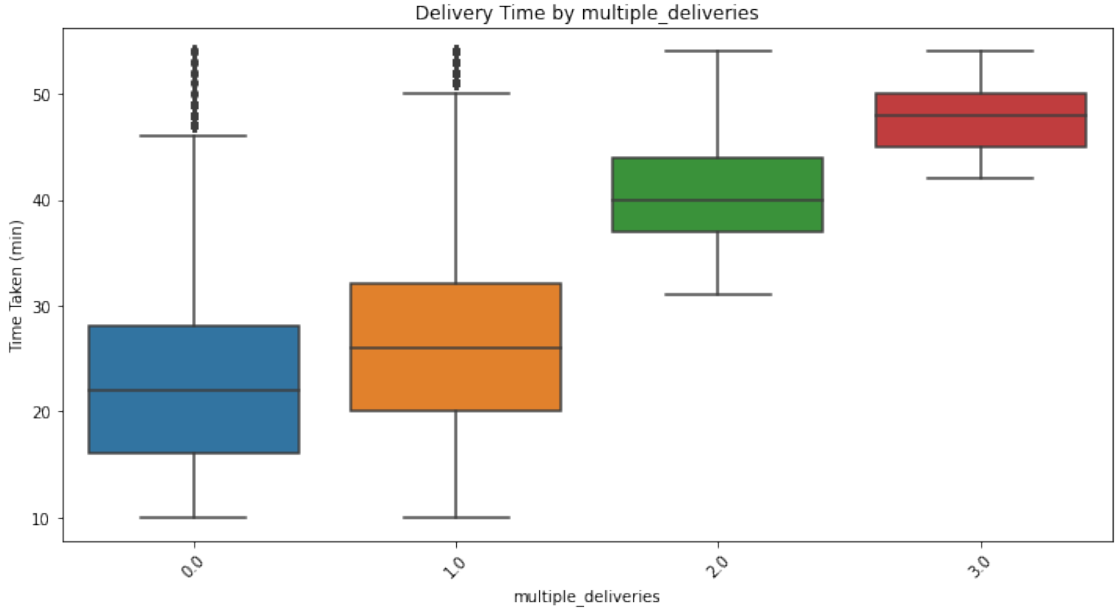
```
0.0                  54.0
1.0                  54.0
2.0                  54.0
3.0                  54.0

Delivery Time Statistics by Festival:
            count       mean       std   min   25%   50%   75%   max
Festival
No        44460.0  25.984121  9.013540  10.0  19.0  25.0  32.0  54.0
Yes         896.0  45.517857  4.001915  38.0  43.0  45.0  48.0  54.0

Delivery Time Statistics by City:
                 count       mean       std   min   25%   50%   75%   max
City
Metropolitian  34087.0  27.314460  9.184753  10.0  20.0  26.0  33.0  54.0
Semi-Urban       164.0  49.731707  2.693089  44.0  48.0  49.0  52.0  54.0
Urban          10133.0  22.983322  8.867217  10.0  16.0  22.0  28.0  54.0
```

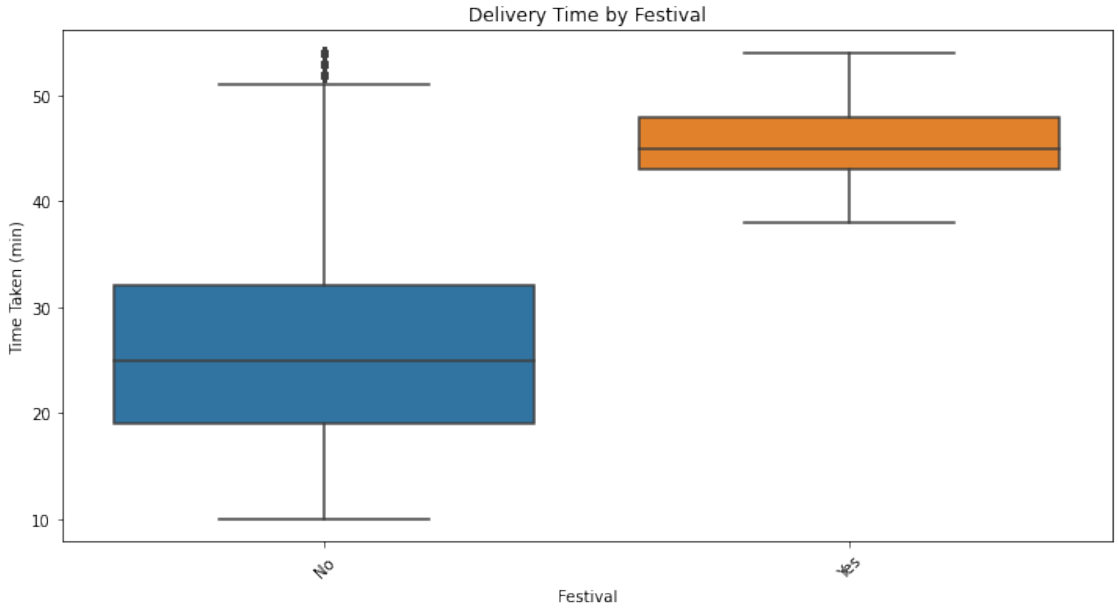
[101]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Function to create box plots for each factor
def plot_delivery_time_by_factor(factor):
    plt.figure(figsize=(12, 6))
    sns.boxplot(x=factor, y='Time_taken (min)', data=df)
    plt.title(f'Delivery Time by {factor}')
    plt.ylabel('Time Taken (min)')
    plt.xlabel(factor)
    plt.xticks(rotation=45)
    plt.show()

# Create box plots for each factor
for factor in factors:
    plot_delivery_time_by_factor(factor)
```

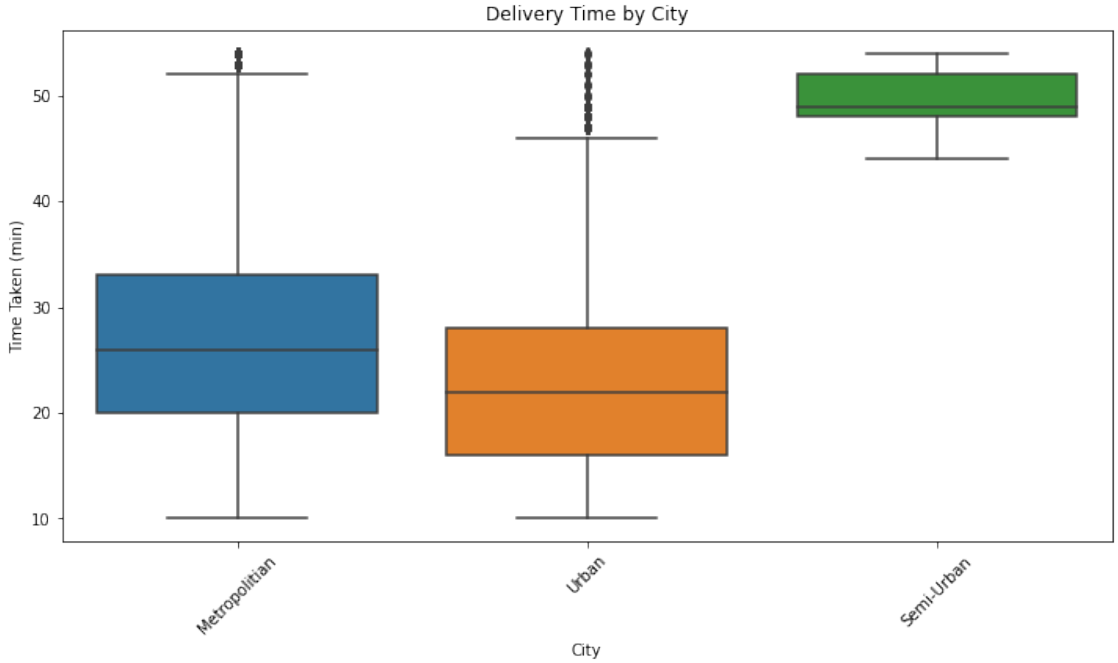

Delivery Time by Weather_conditions
Time Taken (min)
10
20
30
40
50
Fog
Stormy
Sandstorms
Windy
Cloudy
Sunny
Weather_conditions


Delivery Time by Road_traffic_density
Time Taken (min)
10
20
30
40
50
Jam
High
Medium
Low
Road_traffic_density


Delivery Time by Vehicle_condition
Time Taken (min)
50
40
30
20
10
0
1
2
3
Vehicle_condition


Delivery Time by multiple_deliveries
Time Taken (min)
50
40
30
20
10
0.0
1.0
2.0
3.0
multiple_deliveries





[104]:
```
# Calculate the correlation matrix
correlation_matrix = df_encoded.corr()
print(correlation_matrix['Time_taken (min)'])
```

```
Vehicle_condition              -0.234456
```

```
multiple_deliveries              0.387042
Weather_conditions_Fog           0.125441
Weather_conditions_Sandstorms   -0.019779
Weather_conditions_Stormy       -0.020240
Weather_conditions_Sunny        -0.206170
Weather_conditions_Windy        -0.008230
Road_traffic_density_Jam         0.348852
Road_traffic_density_Low        -0.384083
Road_traffic_density_Medium      0.024303
Festival_Yes                     0.290070
City_Semi-Urban                  0.150078
City_Urban                      -0.188612
Time_taken (min)                 1.000000
Name: Time_taken (min), dtype: float64
```

[105]:
```
# Check the unique types of vehicles
unique_vehicle_types = df['Type_of_vehicle'].unique()
print("Unique vehicle types:", unique_vehicle_types)
```

```
Unique vehicle types: ['motorcycle' 'scooter' 'electric_scooter' 'bicycle']
```

[12]:
```
# Count the number of deliveries for each type of vehicle
vehicle_counts = df['Type_of_vehicle'].value_counts()
print(vehicle_counts)
```

```
motorcycle          26429
scooter             15273
electric_scooter     3814
bicycle                68
Name: Type_of_vehicle, dtype: int64
```

[108]:
```
import pandas as pd

# Assuming df is your DataFrame
# Group by 'Type_of_vehicle' and aggregate counts and average delivery times
vehicle_analysis = df.groupby('Type_of_vehicle').agg(
    delivery_count=('Type_of_vehicle', 'size'),  # Count of deliveries
    average_delivery_time=('Time_taken (min)', 'mean')  # Average delivery time
).reset_index()

# Sort the results by delivery count
vehicle_analysis = vehicle_analysis.sort_values(by='delivery_count',␣
 ↪ascending=False)

# Display the results
print(vehicle_analysis)
```

```
   Type_of_vehicle  delivery_count  average_delivery_time
```

2	motorcycle	26429	27.605774
3	scooter	15273	24.478819
1	electric_scooter	3814	24.470110
0	bicycle	68	26.426471

[109]:
```
import pandas as pd

# Assuming your dataset is already loaded into a DataFrame called df
# If not, load the dataset:
# df = pd.read_csv('your_dataset.csv')

# Convert the 'Order_Date' column to datetime if it's not already
df['Order_Date'] = pd.to_datetime(df['Order_Date'], errors='coerce')

# Extract the distinct months from the 'Order_Date' column
df['Month'] = df['Order_Date'].dt.month

# Get distinct months
distinct_months = df['Month'].unique()

# Display the distinct months
distinct_months.sort()
print("Distinct Months:", distinct_months)
```

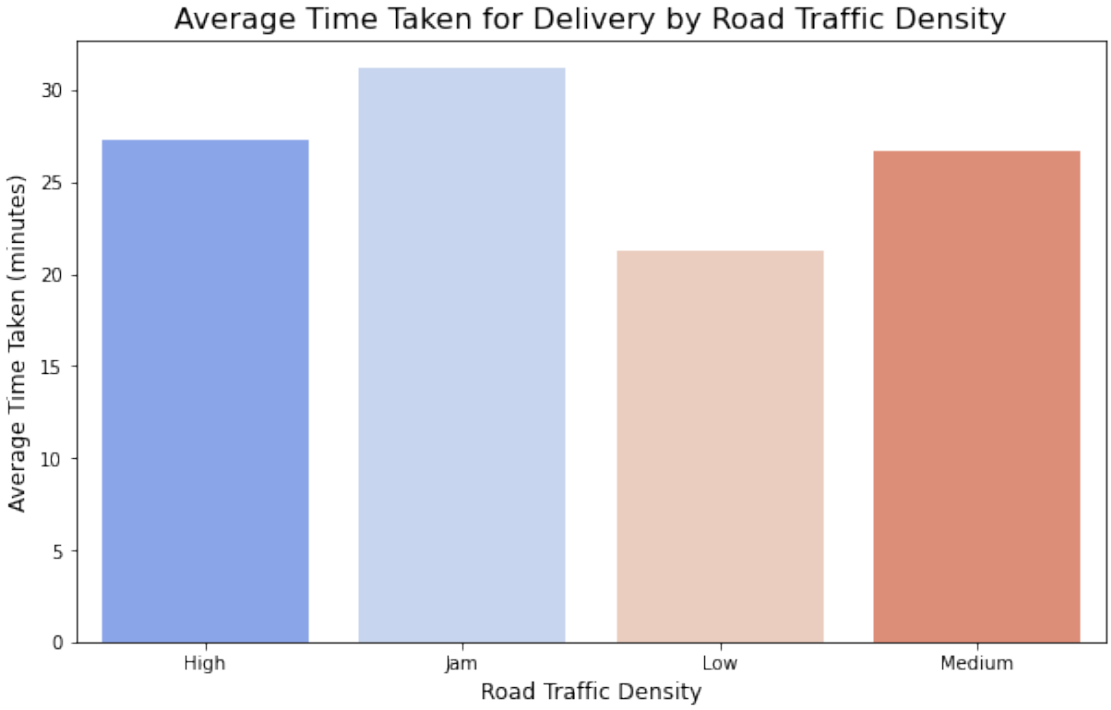
```
Distinct Months: [2 3 4]
```

[6]:
```
# Group by Road_traffic_density and calculate the mean time taken for delivery
traffic_delivery_time = df.groupby('Road_traffic_density')['Time_taken (min)'].
 ↪mean().reset_index()

# Display the result
traffic_delivery_time
```

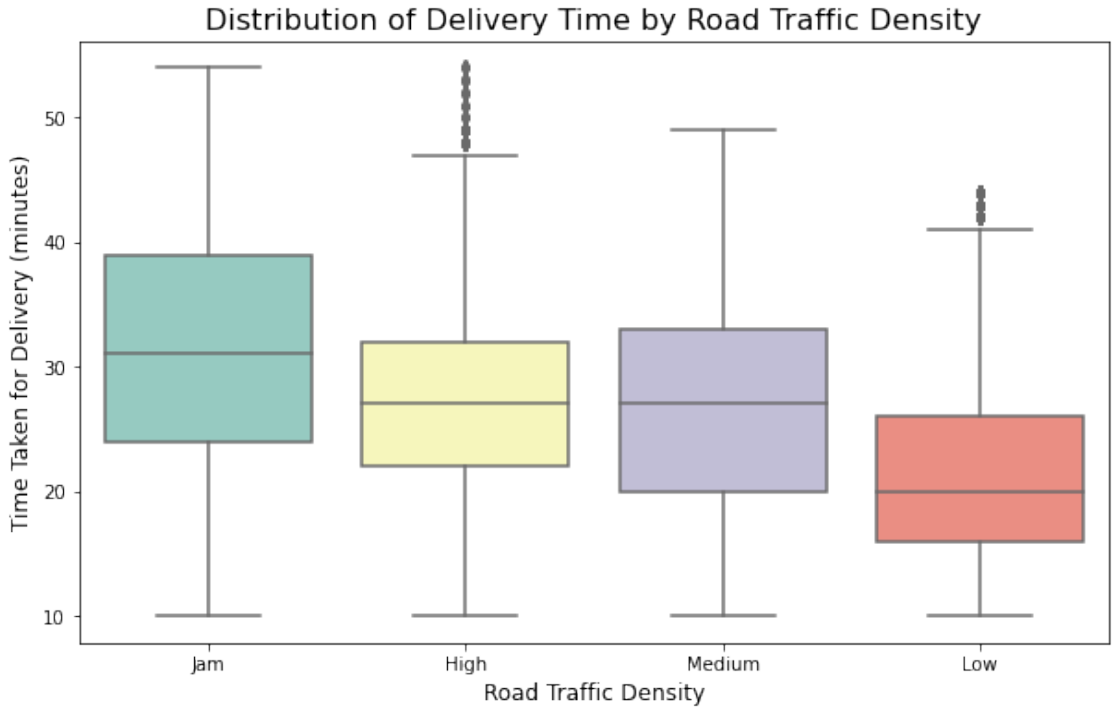
[6]:
	Road_traffic_density	Time_taken (min)
0	High	27.240109
1	Jam	31.176038
2	Low	21.266671
3	Medium	26.699680

[9]:
```
# Set the plot size for better visibility
plt.figure(figsize=(10, 6))

# Create a bar plot for the average delivery time by traffic density
sns.barplot(x='Road_traffic_density', y='Time_taken (min)',␣
 ↪data=traffic_delivery_time, palette='coolwarm')

# Add titles and labels
```

```
plt.title('Average Time Taken for Delivery by Road Traffic Density',␣
 ↪fontsize=16)
plt.xlabel('Road Traffic Density', fontsize=12)
plt.ylabel('Average Time Taken (minutes)', fontsize=12)

# Display the plot
plt.show()
```



[11]:
```
# Set the plot size
plt.figure(figsize=(10, 6))

# Create a boxplot to show the distribution of delivery times
sns.boxplot(x='Road_traffic_density', y='Time_taken (min)', data=df,␣
 ↪palette='Set3')

# Add titles and labels
plt.title('Distribution of Delivery Time by Road Traffic Density', fontsize=16)
plt.xlabel('Road Traffic Density', fontsize=12)
plt.ylabel('Time Taken for Delivery (minutes)', fontsize=12)

# Display the plot
plt.show()
```


Distribution of Delivery Time by Road Traffic Density
Time Taken for Delivery (minutes)
50
40
30
20
10
Jam
High
Medium
Low
Road Traffic Density

[ ]: