Activity_Course 3 Automatidata project lab

June 20, 2023

1 Course 3 Automatidata project

Course 3 - Go Beyond the Numbers: Translate Data into Insights

You are the newest data professional in a fictional data consulting firm: Automatidata. The team is still early into the project, having only just completed an initial plan of action and some early Python coding work.

Luana Rodriquez, the senior data analyst at Automatidata, is pleased with the work you have already completed and requests your assistance with some EDA and data visualization work for the New York City Taxi and Limousine Commission project (New York City TLC) to get a general understanding of what taxi ridership looks like. The management team is asking for a Python notebook showing data structuring and cleaning, as well as any matplotlib/seaborn visualizations plotted to help understand the data. At the very least, include a box plot of the ride durations and some time series plots, like a breakdown by quarter or month.

Additionally, the management team has recently asked all EDA to include Tableau visualizations. For this taxi data, create a Tableau dashboard showing a New York City map of taxi/limo trips by month. Make sure it is easy to understand to someone who isn't data savvy, and remember that the assistant director at the New York City TLC is a person with visual impairments.

A notebook was structured and prepared to help you in this project. Please complete the following questions.

2 Course 3 End-of-course project: Exploratory data analysis

In this activity, you will examine data provided and prepare it for analysis. You will also design a professional data visualization that tells a story, and will help data-driven decisions for business needs.

Please note that the Tableau visualization activity is optional, and will not affect your completion of the course. Completing the Tableau activity will help you practice planning out and plotting a data visualization based on a specific business need. The structure of this activity is designed to emulate the proposals you will likely be assigned in your career as a data professional. Completing this activity will help prepare you for those career moments.

The purpose of this project is to conduct exploratory data analysis on a provided data set. Your mission is to continue the investigation you began in C2 and perform further EDA on this data with the aim of learning more about the variables.

The goal is to clean data set and create a visualization.
This activity has 4 parts:

Part 1: Imports, links, and loading

Part 2: Data Exploration * Data cleaning

Part 3: Building visualizations

Part 4: Evaluate and share results

Follow the instructions and answer the questions below to complete the activity. Then, you will complete an Executive Summary using the questions listed on the PACE Strategy Document.

Be sure to complete this activity before moving on. The next course item will provide you with a completed exemplar to compare to your own work.

3 Visualize a story in Tableau and Python

4 PACE stages

- `[Plan](#scrollTo=psz51YkZVwtN&line=3&uniqifier=1)`
- `[Analyze](#scrollTo=mA7Mz_SnI8km&line=4&uniqifier=1)`
- `[Construct](#scrollTo=Lca9c8XON8lc&line=2&uniqifier=1)`
- `[Execute](#scrollTo=401PgchTPr4E&line=2&uniqifier=1)`

Throughout these project notebooks, you'll see references to the problem-solving framework PACE. The following notebook components are labeled with the respective PACE stage: Plan, Analyze, Construct, and Execute.

4.1 PACE: Plan

In this stage, consider the following questions where applicable to complete your code response: 1. Identify any outliers:

- What methods are best for identifying outliers?
- How do you make the decision to keep or exclude outliers from any future models?

Identifying outliers is an important step in data analysis to understand the presence of extreme values that may significantly impact the results or assumptions of a model. Here are some common methods for identifying outliers:

1. Visual inspection: Visualizing the data using scatter plots, box plots, or histograms can help identify data points that deviate significantly from the overall pattern. Outliers may appear as individual points that are distant from the bulk of the data.
2. Statistical methods: Statistical techniques such as z-scores, standard deviation, and interquartile range (IQR) can be used to quantify the deviation of data points from the mean or median.

Data points that fall beyond a certain threshold (e.g., more than 3 standard deviations from the mean) can be considered outliers.

3. Domain knowledge: Having a deep understanding of the domain and the specific context of the data can help identify outliers. For example, if a value is known to be impossible or inconsistent with prior knowledge, it can be flagged as an outlier.

When deciding whether to keep or exclude outliers from future models, it depends on the nature and characteristics of the data, as well as the objectives of the analysis. Here are a few considerations:

1. Data quality and integrity: If outliers are identified due to data entry errors, measurement errors, or other anomalies, it may be appropriate to exclude them from the analysis to ensure the integrity and accuracy of the results.
2. Impact on the analysis: Consider the potential impact of outliers on the analysis or modeling techniques being used. Outliers can have a significant influence on statistical measures such as mean, standard deviation, and correlation coefficients. If the outliers are genuine data points that reflect important information or behavior, excluding them may lead to a biased or incomplete analysis.
3. Robustness of the model: Some modeling techniques, such as robust regression or outlier-resistant algorithms, can handle outliers effectively without the need for exclusion. In such cases, keeping outliers in the dataset might be preferred to maintain the model's robustness.
4. Data size and representativeness: If the dataset is large and the outliers constitute a small proportion, excluding them may have minimal impact on the overall analysis. However, if the outliers are representative of a specific subpopulation or behavior of interest, excluding them may result in a biased representation.

Ultimately, the decision to keep or exclude outliers should be made based on a thorough understanding of the data, the specific analysis objectives, and the potential implications of outlier removal on the accuracy and validity of the results. It is important to carefully evaluate the trade-offs and consider the implications of any decision made regarding outliers.

4.1.1 Task 1. Imports, links, and loading

Go to Tableau Public The following link will help you complete this activity. Keep Tableau Public open as you proceed to the next steps.

Link to supporting materials: Tableau Public: https://public.tableau.com/s/

For EDA of the data, import the data and packages that would be most helpful, such as pandas, numpy and matplotlib.

[211]:
```
# Import packages and libraries
#==> ENTER YOUR CODE HERE
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
```

Note: As shown in this cell, the dataset has been automatically loaded in for you. You do not need to download the .csv file, or provide more code, in order to access the dataset and proceed with this lab. Please continue with this activity by completing the following instructions.

[212]:
```python
# Load dataset into dataframe
df = pd.read_csv('2017_Yellow_Taxi_Trip_Data.csv')
```

4.2 PACE: Analyze

Consider the questions in your PACE Strategy Document to reflect on the Analyze stage.

4.2.1 Task 2a. Data exploration and cleaning

Decide which columns are applicable

The first step is to assess your data. Check the Data Source page on Tableau Public to get a sense of the size, shape and makeup of the data set. Then answer these questions to yourself:

Given our scenario, which data columns are most applicable? Which data columns can I eliminate, knowing they won't solve our problem scenario?

Consider functions that help you understand and structure the data.

- head()
- describe()
- info()
- groupby()
- sortby()

What do you do about missing data (if any)?

Are there data outliers? What are they and how might you handle them?

What do the distributions of your variables tell you about the question you're asking or the problem you're trying to solve?

To determine if there are data outliers, we can analyze the distribution of the variables and look for any extreme values that deviate significantly from the majority of the data. The summary statistics obtained from the `.describe()` function can provide initial insights into the presence of outliers.

Handling outliers depends on the context and the specific variables in question. Here are some approaches to consider:

1. Identify Outliers: By examining the summary statistics or visualizing the data through box plots or histograms, we can identify potential outliers. Outliers are data points that are significantly different from other observations and may have a disproportionate impact on the analysis or model.

2. Assess Context: It is essential to consider the context of the data and the problem being addressed. Some outliers may be valid and meaningful observations, while others could be

data entry errors or anomalies. Understanding the domain knowledge and consulting with subject matter experts can help determine the appropriateness of outliers.

3. Evaluate Impact: Assess the impact of outliers on the analysis or model. Determine whether outliers significantly affect the distribution, summary statistics, or relationships between variables. Evaluate if removing outliers or applying appropriate transformations would improve the analysis or model performance.
4. Handling Options: Depending on the situation, there are different approaches to handling outliers:
   a. Data Transformation: Transforming the data using methods like log transformation or winsorization can reduce the influence of extreme values while preserving the overall distribution.

   b. Winsorization: Winsorization replaces extreme values with values at a specified percentile, thereby reducing the impact of outliers.

   c. Removal: In some cases, removing outliers from the dataset might be appropriate if they are deemed as errors or if they significantly skew the analysis. However, caution should be exercised, as removing outliers without justification can introduce bias and impact the representativeness of the data.

   d. Robust Methods: Using robust statistical methods that are less sensitive to outliers can be an alternative. These methods, such as robust regression or median-based estimators, are designed to handle outliers more effectively.

Regarding the distributions of the variables, analyzing their shapes and characteristics can provide insights into the problem at hand. Here are a few observations to consider:

1. Fare Amount: The distribution of fare amounts can help understand the typical pricing structure for taxi rides. It can reveal the range of fares paid by passengers and provide insights into fare variability.
2. Trip Distance: The distribution of trip distances can indicate the typical length of taxi rides. It can reveal whether most trips are short or long and help understand the travel patterns within the city.
3. Payment Type: Examining the distribution of payment types can provide insights into the preferred payment methods among taxi passengers. It can help identify the dominant payment type and potentially uncover any shifts or trends in payment preferences over time.
4. Passenger Count: Analyzing the distribution of passenger counts can provide information about the typical number of passengers per taxi ride. It can help identify the most common passenger group sizes and potentially reveal any patterns or correlations with other variables.

By understanding the distributions of these variables and their relationships, we can gain insights into the question we are asking or the problem we are trying to solve. It helps us understand the characteristics of the data and provides a foundation for further analysis and modeling.

Start by discovering, using head and size.

[213]:
```
#==> ENTER YOUR CODE HERE
# Display the first few rows
```

```
df.head(10)
```

```
[213]:    Unnamed: 0  VendorID     tpep_pickup_datetime    tpep_dropoff_datetime  \
      0     24870114         2   03/25/2017 8:55:43 AM   03/25/2017 9:09:47 AM
      1     35634249         1   04/11/2017 2:53:28 PM   04/11/2017 3:19:58 PM
      2    106203690         1   12/15/2017 7:26:56 AM   12/15/2017 7:34:08 AM
      3     38942136         2   05/07/2017 1:17:59 PM   05/07/2017 1:48:14 PM
      4     30841670         2  04/15/2017 11:32:20 PM  04/15/2017 11:49:03 PM
      5     23345809         2   03/25/2017 8:34:11 PM   03/25/2017 8:42:11 PM
      6     37660487         2   05/03/2017 7:04:09 PM   05/03/2017 8:03:47 PM
      7     69059411         2   08/15/2017 5:41:06 PM   08/15/2017 6:03:05 PM
      8      8433159         2   02/04/2017 4:17:07 PM   02/04/2017 4:29:14 PM
      9     95294817         1   11/10/2017 3:20:29 PM   11/10/2017 3:40:55 PM

         passenger_count  trip_distance  RatecodeID store_and_fwd_flag  \
      0                6           3.34           1                  N
      1                1           1.80           1                  N
      2                1           1.00           1                  N
      3                1           3.70           1                  N
      4                1           4.37           1                  N
      5                6           2.30           1                  N
      6                1          12.83           1                  N
      7                1           2.98           1                  N
      8                1           1.20           1                  N
      9                1           1.60           1                  N

         PULocationID  DOLocationID  payment_type  fare_amount  extra  mta_tax  \
      0           100           231             1         13.0    0.0      0.5
      1           186            43             1         16.0    0.0      0.5
      2           262           236             1          6.5    0.0      0.5
      3           188            97             1         20.5    0.0      0.5
      4             4           112             2         16.5    0.5      0.5
      5           161           236             1          9.0    0.5      0.5
      6            79           241             1         47.5    1.0      0.5
      7           237           114             1         16.0    1.0      0.5
      8           234           249             2          9.0    0.0      0.5
      9           239           237             1         13.0    0.0      0.5

         tip_amount  tolls_amount  improvement_surcharge  total_amount
      0        2.76           0.0                    0.3         16.56
      1        4.00           0.0                    0.3         20.80
      2        1.45           0.0                    0.3          8.75
      3        6.39           0.0                    0.3         27.69
      4        0.00           0.0                    0.3         17.80
      5        2.06           0.0                    0.3         12.36
      6        9.86           0.0                    0.3         59.16
      7        1.78           0.0                    0.3         19.58
```

```
8        0.00          0.0                      0.3         9.80
9        2.75          0.0                      0.3        16.55
```

[215]:
```
#==> ENTER YOUR CODE HERE
# Determine the size of the dataset
print("Dataset size:", df.size)
```

```
Dataset size: 408582
```

Use describe...

[214]:
```
#==> ENTER YOUR CODE HERE
df.describe()
```

[214]:
```
         Unnamed: 0      VendorID  passenger_count  trip_distance  \
count  2.269900e+04  22699.000000     22699.000000   22699.000000
mean   5.675849e+07      1.556236         1.642319       2.913313
std    3.274493e+07      0.496838         1.285231       3.653171
min    1.212700e+04      1.000000         0.000000       0.000000
25%    2.852056e+07      1.000000         1.000000       0.990000
50%    5.673150e+07      2.000000         1.000000       1.610000
75%    8.537452e+07      2.000000         2.000000       3.060000
max    1.134863e+08      2.000000         6.000000      33.960000

         RatecodeID  PULocationID  DOLocationID  payment_type   fare_amount  \
count  22699.000000  22699.000000  22699.000000  22699.000000  22699.000000
mean       1.043394    162.412353    161.527997      1.336887     13.026629
std        0.708391     66.633373     70.139691      0.496211     13.243791
min        1.000000      1.000000      1.000000      1.000000   -120.000000
25%        1.000000    114.000000    112.000000      1.000000      6.500000
50%        1.000000    162.000000    162.000000      1.000000      9.500000
75%        1.000000    233.000000    233.000000      2.000000     14.500000
max       99.000000    265.000000    265.000000      4.000000    999.990000

              extra       mta_tax    tip_amount  tolls_amount  \
count  22699.000000  22699.000000  22699.000000  22699.000000
mean       0.333275      0.497445      1.835781      0.312542
std        0.463097      0.039465      2.800626      1.399212
min       -1.000000     -0.500000      0.000000      0.000000
25%        0.000000      0.500000      0.000000      0.000000
50%        0.000000      0.500000      1.350000      0.000000
75%        0.500000      0.500000      2.450000      0.000000
max        4.500000      0.500000    200.000000     19.100000

       improvement_surcharge  total_amount
count           22699.000000  22699.000000
mean                0.299551     16.310502
std                 0.015673     16.097295
```

```
min                -0.300000   -120.300000
25%                 0.300000      8.750000
50%                 0.300000     11.800000
75%                 0.300000     17.800000
max                 0.300000   1200.290000
```

And info.

[216]:
```
#==> ENTER YOUR CODE HERE
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22699 entries, 0 to 22698
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             22699 non-null  int64
 1   VendorID               22699 non-null  int64
 2   tpep_pickup_datetime   22699 non-null  object
 3   tpep_dropoff_datetime  22699 non-null  object
 4   passenger_count        22699 non-null  int64
 5   trip_distance          22699 non-null  float64
 6   RatecodeID             22699 non-null  int64
 7   store_and_fwd_flag     22699 non-null  object
 8   PULocationID           22699 non-null  int64
 9   DOLocationID           22699 non-null  int64
 10  payment_type           22699 non-null  int64
 11  fare_amount            22699 non-null  float64
 12  extra                  22699 non-null  float64
 13  mta_tax                22699 non-null  float64
 14  tip_amount             22699 non-null  float64
 15  tolls_amount           22699 non-null  float64
 16  improvement_surcharge  22699 non-null  float64
 17  total_amount           22699 non-null  float64
dtypes: float64(8), int64(7), object(3)
memory usage: 3.1+ MB
```

4.2.2 Task 2b. Assess whether dimensions and measures are correct

On the data source page in Tableau, double check the data types for the applicable columns you selected on the previous step. Pay close attention to the dimensions and measures to assure they are correct.

In Python, consider the data types of the columns. *Consider:* Do they make sense?

Review the link provided in the previous activity instructions to create the required Tableau visualization.

4.2.3 Task 2c. Select visualization type(s)

Select data visualization types that will help you understand and explain the data.

Now that you know which data columns you'll use, it is time to decide which data visualization makes the most sense for EDA of the TLC dataset. What type of data visualization(s) would be most helpful?

- Line graph
- Bar chart
- Box plot
- Histogram
- Heat map
- Scatter plot
- A geographic map

Based on the TLC dataset, the following types of data visualizations would be most helpful:

1. Line Graph: A line graph can be used to visualize the trends and patterns in variables such as fares, trip distances, or passenger counts over time. It can provide insights into the changes and fluctuations in these variables.

2. Bar Chart: A bar chart is useful for comparing and displaying categorical data. It can be used to visualize the distribution of variables like payment type, vendor ID, or rate code. Bar charts can provide insights into the frequencies or proportions of different categories.

3. Box Plot: A box plot is effective in summarizing the distribution of numerical variables. It can show the median, quartiles, and any outliers, helping to identify the presence of extreme values or variability in the data.

4. Histogram: A histogram is suitable for visualizing the distribution of a single numerical variable. It can provide insights into the shape, skewness, and central tendency of the data.

5. Scatter Plot: A scatter plot is useful for exploring the relationship between two numerical variables. It can help identify patterns, trends, or correlations between variables such as trip distance and fare amount.

6. Geographic Map: Since the TLC dataset includes location information, a geographic map can be beneficial to visualize the spatial distribution of taxi trips or other variables across different regions of New York City. It can provide insights into hotspots, patterns, and areas of high demand.

The selection of specific visualization types depends on the research questions and the variables of interest. By using these visualizations, you can gain a deeper understanding of the data, identify patterns, detect outliers, and explore relationships between variables.

4.3 PACE: Construct

Consider the questions in your PACE Strategy Document to reflect on the Construct stage.

4.3.1 Task 3. Data visualization

You've assessed your data, and decided on which data variables are most applicable. It's time to plot your visualization(s)!

4.3.2 Boxplots

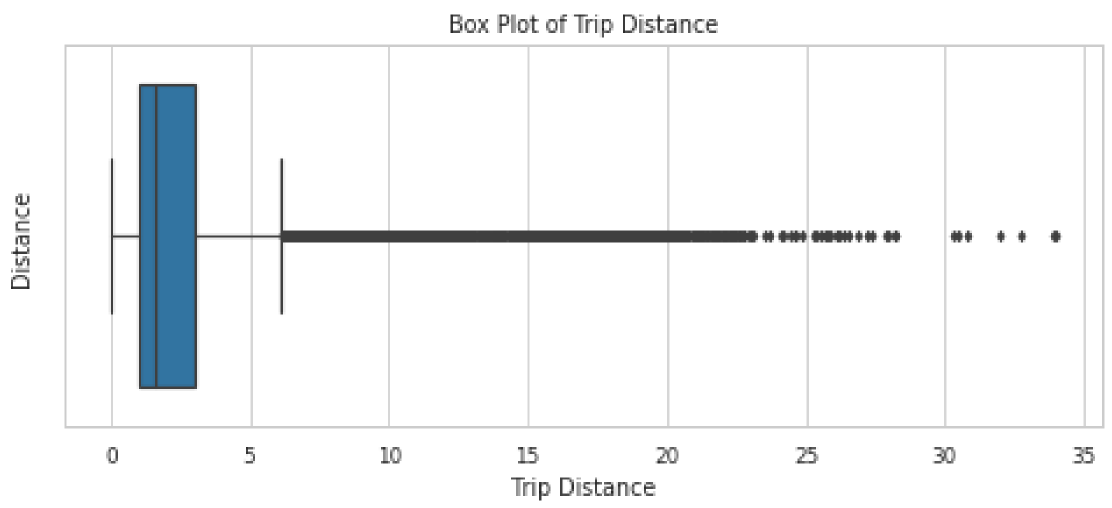
Perform a check for outliers on relevant columns such as trip distance and trip duration. Remember, some of the best ways to identify the presence of outliers in data are box plots and histograms.

Note: Remember to convert your date columns to datetime in order to derive total trip duration.

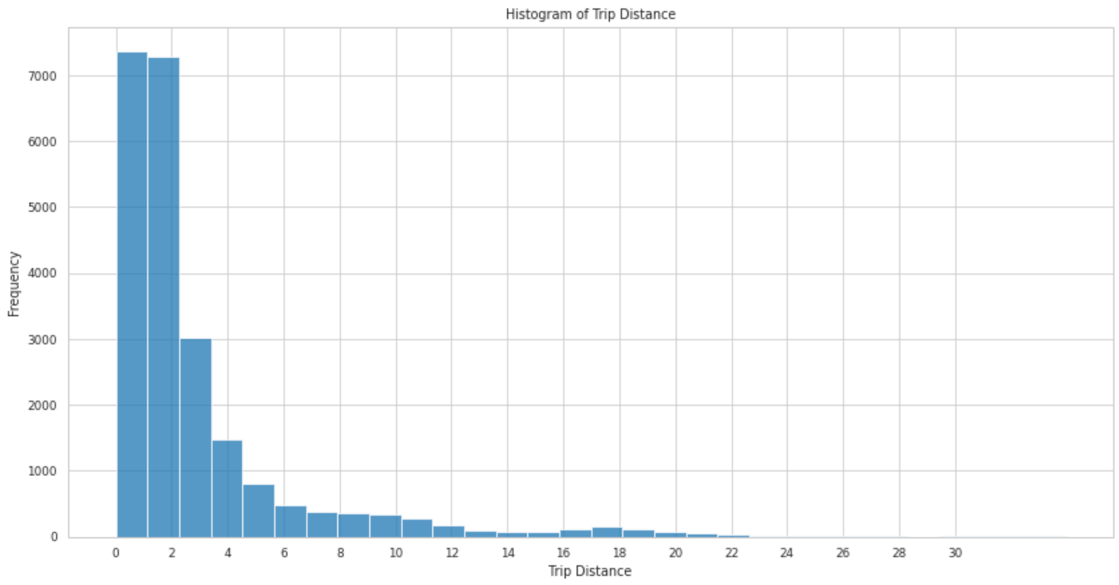
[218]:
```python
# Convert data columns to datetime
#==> ENTER YOUR CODE HERE
df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'])
```

trip distance

[220]:
```python
# Create box plot of trip_distance
#==> ENTER YOUR CODE HERE

# Create box plot using seaborn
plt.figure(figsize=(8, 3))  # Set the figure size

# Customize the box plot
sns.boxplot(x='trip_distance', data=df, fliersize=3)

# Set labels and title
plt.xlabel('Trip Distance')
plt.ylabel('Distance')
plt.title('Box Plot of Trip Distance')

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

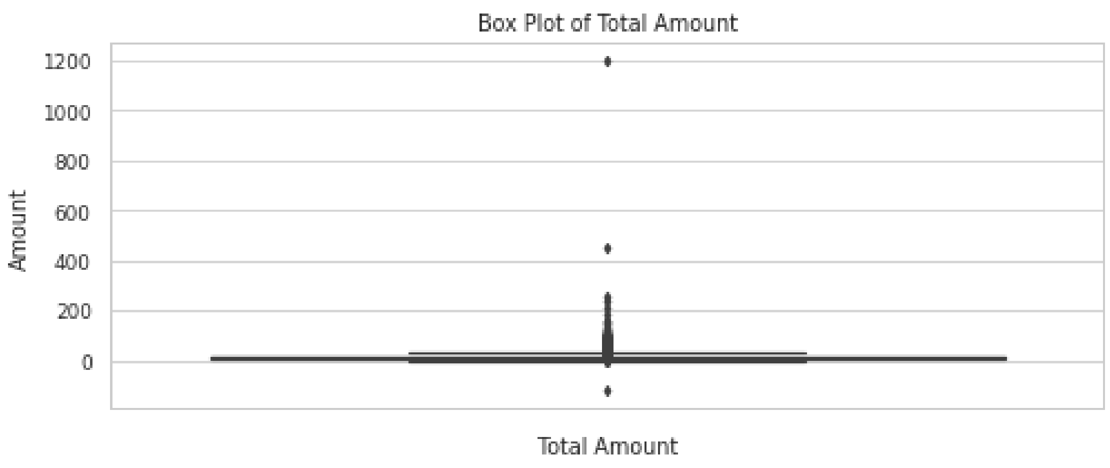


[221]:
```
# Create histogram of trip_distance
#==> ENTER YOUR CODE HERE

plt.figure(figsize=(14, 7))  # Set the figure size

# Customize the histogram
sns.histplot(data=df, x='trip_distance', bins=30)

# Set labels and title
plt.xlabel('Trip Distance')
plt.ylabel('Frequency')
plt.title('Histogram of Trip Distance')

# Set x-tick labels with a perfect range
plt.xticks(range(0, 32, 2))

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

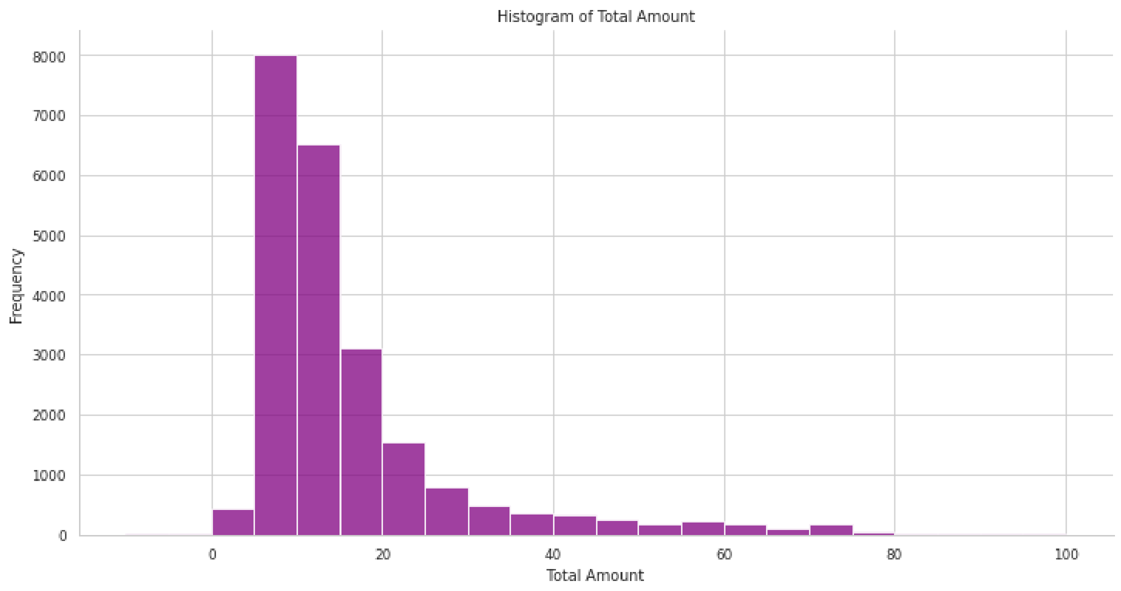


total amount

```
[222]: # Create box plot of total_amount
#==> ENTER YOUR CODE HERE

# Create box plot using seaborn
plt.figure(figsize=(8, 3))  # Set the figure size

# Customize the box plot
sns.boxplot(data=df, y='total_amount', fliersize=3)

# Set labels and title
plt.xlabel('Total Amount')
plt.ylabel('Amount')
plt.title('Box Plot of Total Amount')

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

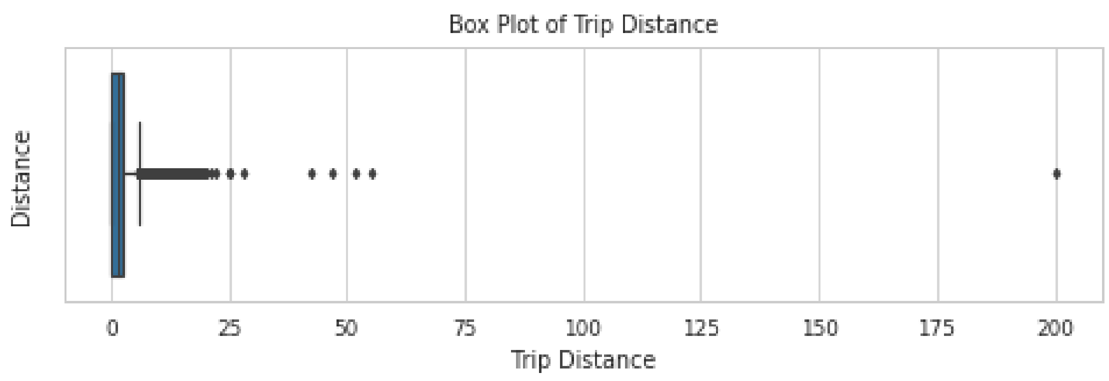


[223]:
```
# Create histogram of total_amount
#==> ENTER YOUR CODE HERE

# Set the style and context for seaborn
sns.set_style("whitegrid")
sns.set_context("paper")

# Create histogram of total_amount
plt.figure(figsize=(12, 6))
ax = sns.histplot(df['total_amount'], bins=range(-10,101,5), color='purple',␣
 ↪edgecolor='white')

# Set labels and title
plt.xlabel('Total Amount')
plt.ylabel('Frequency')
plt.title('Histogram of Total Amount')


# Remove spines
sns.despine()

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

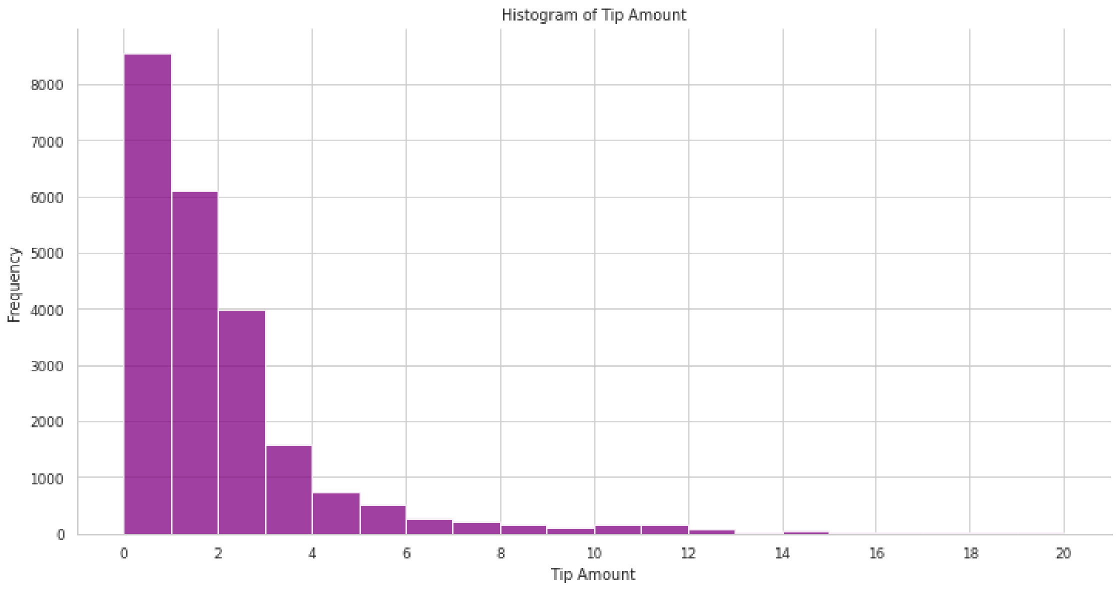


tip amount

[228]:
```
# Create box plot of tip_amount
#==> ENTER YOUR CODE HERE

# Create box plot using seaborn
plt.figure(figsize=(8, 2))  # Set the figure size

# Customize the box plot
sns.boxplot(x='tip_amount', data=df, fliersize=3)

# Set labels and title
plt.xlabel('Trip Distance')
plt.ylabel('Distance')
plt.title('Box Plot of Trip Distance')

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



[225]:
```python
# Create histogram of tip_amount
#==> ENTER YOUR CODE HERE

# Set the style and context for seaborn
sns.set_style("whitegrid")
sns.set_context("paper")

# Create histogram of tip_amount
plt.figure(figsize=(12, 6))
ax = sns.histplot(df['tip_amount'], bins=range(0,21,1), color='purple',␣
 ↪edgecolor='white')

# Set labels and title
plt.xlabel('Tip Amount')
plt.ylabel('Frequency')
plt.title('Histogram of Tip Amount')

# Customize x-axis ticks
plt.xticks(range(0,21,2))

# Remove spines
sns.despine()

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

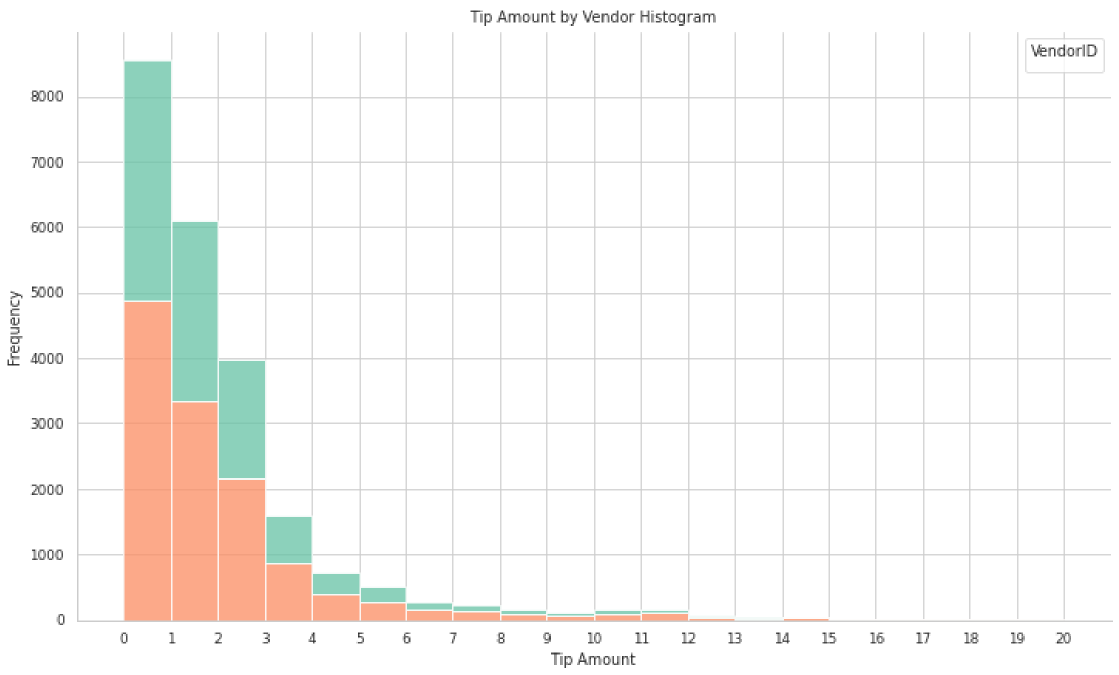


tip_amount by vendor

[229]:
```
# Create histogram of tip_amount by vendor
#==> ENTER YOUR CODE HERE

# Set the style and context for seaborn
sns.set_style("whitegrid")
sns.set_context("paper")

# Create histogram of tip_amount by vendor
plt.figure(figsize=(12, 7))
ax = sns.histplot(data=df, x='tip_amount', bins=range(0,21,1),
                  hue='VendorID', multiple='stack', palette='Set2')

# Set labels and title
plt.xlabel('Tip Amount')
plt.ylabel('Frequency')
plt.title('Tip Amount by Vendor Histogram')

# Customize x-axis ticks
plt.xticks(range(0,21,1))

# Set legend
plt.legend(title='VendorID')

# Remove spines
sns.despine()
```

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

```
No handles with labels found to put in legend.
```

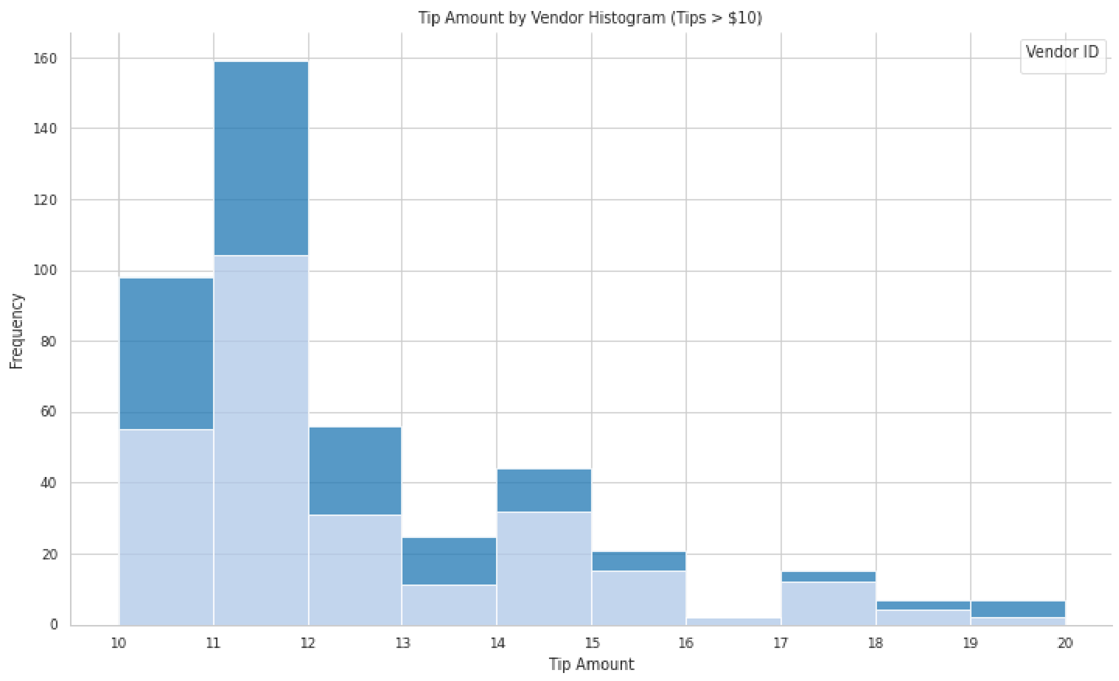


Next, zoom in on the upper end of the range of tips to check whether vendor one gets noticeably more of the most generous tips.

[230]:
```
# Create histogram of tip_amount by vendor for tips > $10
#==> ENTER YOUR CODE HERE

# Set the style and context for seaborn
sns.set_style("whitegrid")
sns.set_context("paper")

# Filter the dataframe for tips greater than $10
tips_over_ten = df[df['tip_amount'] > 10]

# Create histogram of tip_amount by vendor for tips > $10
plt.figure(figsize=(12, 7))
ax = sns.histplot(data=tips_over_ten, x='tip_amount', bins=range(10,21,1),
                  hue='VendorID',
                  multiple='stack',
                  palette='tab20')

# Set labels and title
```

```
plt.xlabel('Tip Amount')
plt.ylabel('Frequency')
plt.title('Tip Amount by Vendor Histogram (Tips > $10)')

# Customize x-axis ticks
plt.xticks(range(10,21,1))

# Add a legend
plt.legend(title='Vendor ID')

# Remove spines
sns.despine()

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

```
No handles with labels found to put in legend.
```



Mean tips by passenger count

Examine the unique values in the `passenger_count` column.

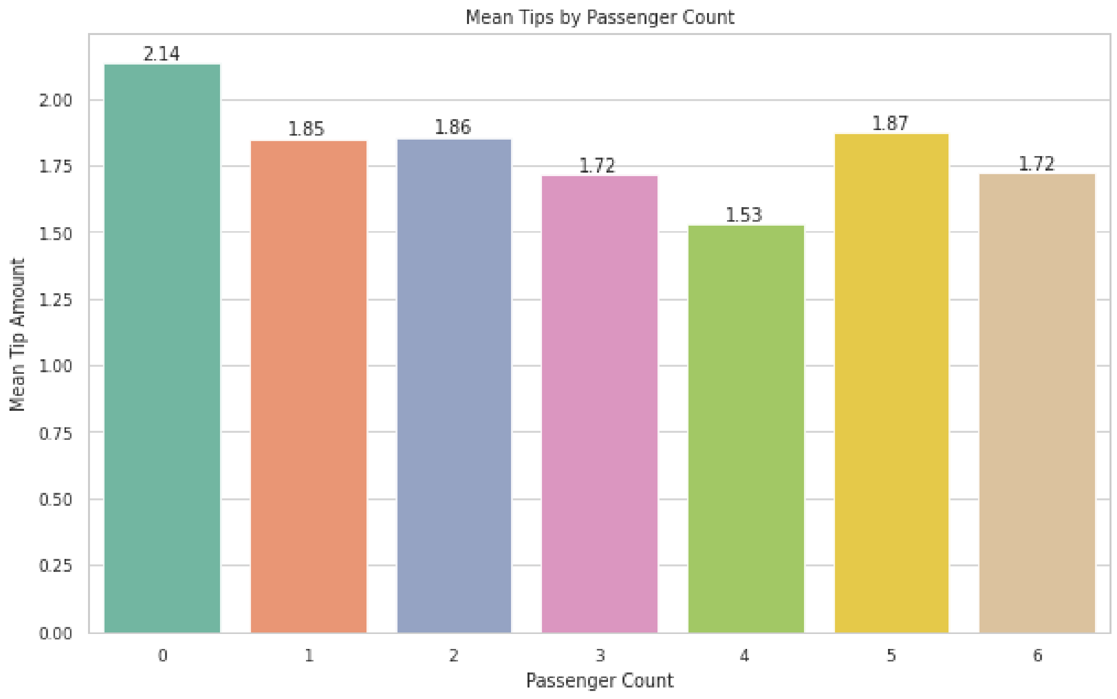
[231]:
```
#==> ENTER YOUR CODE HERE
unique_passenger_count = df['passenger_count'].value_counts()
print(unique_passenger_count)
```

```
1    16117
2     3305
5     1143
3      953
6      693
4      455
0       33
Name: passenger_count, dtype: int64
```

[232]:
```python
# Calculate mean tips by passenger_count
#==> ENTER YOUR CODE HERE
mean_tips_by_passenger_count = df.groupby('passenger_count')['tip_amount'].
 ↪mean()
mean_tips_by_passenger_count
```

[232]:
```
passenger_count
0    2.135758
1    1.848920
2    1.856378
3    1.716768
4    1.530264
5    1.873185
6    1.720260
Name: tip_amount, dtype: float64
```

[233]:
```python
# Create bar plot for mean tips by passenger count
#==> ENTER YOUR CODE HERE



# Calculate mean tips by passenger count
mean_tips = df.groupby('passenger_count')['tip_amount'].mean().reset_index()

# Set the style and context for seaborn
sns.set_style("whitegrid")
sns.set_context("paper")

# Create bar plot for mean tips by passenger count
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=mean_tips, x='passenger_count', y='tip_amount',␣
 ↪palette='Set2')
plt.xlabel('Passenger Count')
plt.ylabel('Mean Tip Amount')
plt.title('Mean Tips by Passenger Count')

# Add data labels to the bars
for index, row in mean_tips.iterrows():
    plt.text(row.name, row.tip_amount, f"{row.tip_amount:.2f}", ha='center',␣
↪va='bottom')

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



Create month and day columns

[234]:
```
# Create a month column
#==> ENTER YOUR CODE HERE
df['month'] = df['tpep_pickup_datetime'].dt.strftime('%B')

# Create a day column
#==> ENTER YOUR CODE HERE
df['day'] = df['tpep_pickup_datetime'].dt.strftime('%A')
```

Plot total ride count by month

Begin by calculating total ride count by month.

[235]:
```
# Get total number of rides for each month
#==> ENTER YOUR CODE HERE
monthly_rides = df['month'].value_counts().sort_index()
monthly_rides
```

[235]:
```
April        2019
August       1724
December     1863
February     1769
January      1997
July         1697
June         1964
March        2049
May          2013
November     1843
October      2027
September    1734
Name: month, dtype: int64
```

Reorder the results to put the months in calendar order.

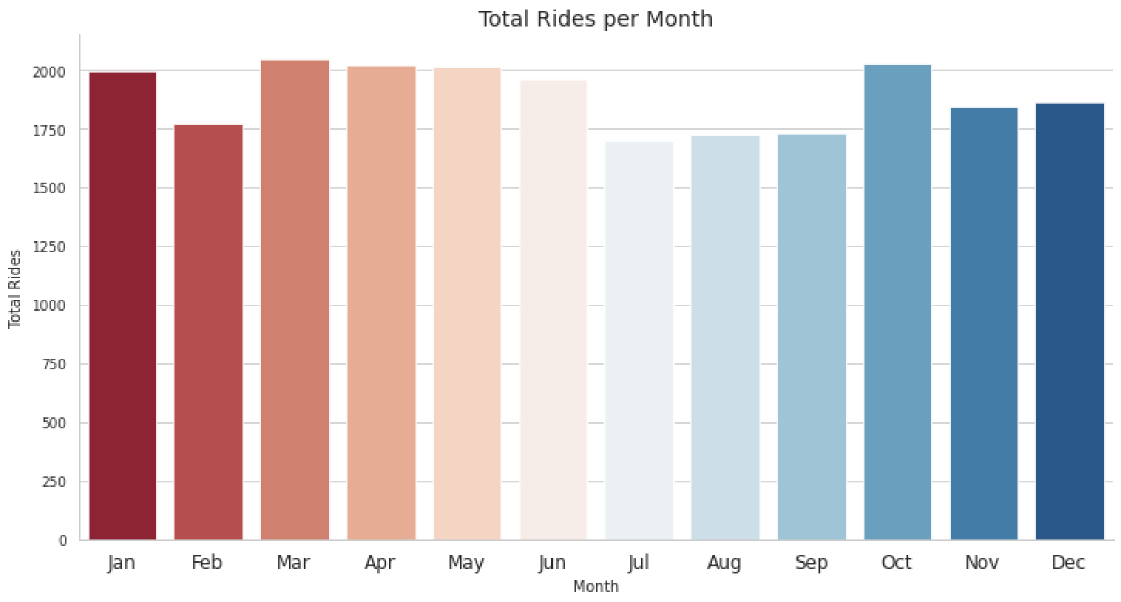
[236]:
```
# Reorder the monthly ride list so months go in order
#==> ENTER YOUR CODE HERE
monthly_rides = monthly_rides.reindex(['January', 'February', 'March', 'April',
                                       'May', 'June', 'July', 'August',␣
 ↪'September',
                                       'October', 'November', 'December'])
monthly_rides
```

[236]:
```
January      1997
February     1769
March        2049
April        2019
May          2013
June         1964
July         1697
August       1724
September    1734
October      2027
November     1843
December     1863
Name: month, dtype: int64
```

[237]:
```
# Show the index
#==> ENTER YOUR CODE HERE
monthly_rides.index
```

[237]:
```
Index(['January', 'February', 'March', 'April', 'May', 'June', 'July',
       'August', 'September', 'October', 'November', 'December'],
      dtype='object')
```

[238]:
```python
# Create a bar plot of total rides per month
#==> ENTER YOUR CODE HERE


# Set the style and context for seaborn
sns.set_style("whitegrid")
sns.set_context("paper")

# Create bar plot of total rides per month
plt.figure(figsize=(12, 6))
ax = sns.barplot(x=monthly_rides.index, y=monthly_rides.values, palette='RdBu')
ax.set_xticklabels(monthly_rides.index.str[:3], fontsize=12)
plt.xticks(rotation=0)
plt.xlabel('Month')
plt.ylabel('Total Rides')
plt.title('Total Rides per Month', fontsize=14);

# Remove spines
sns.despine()

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



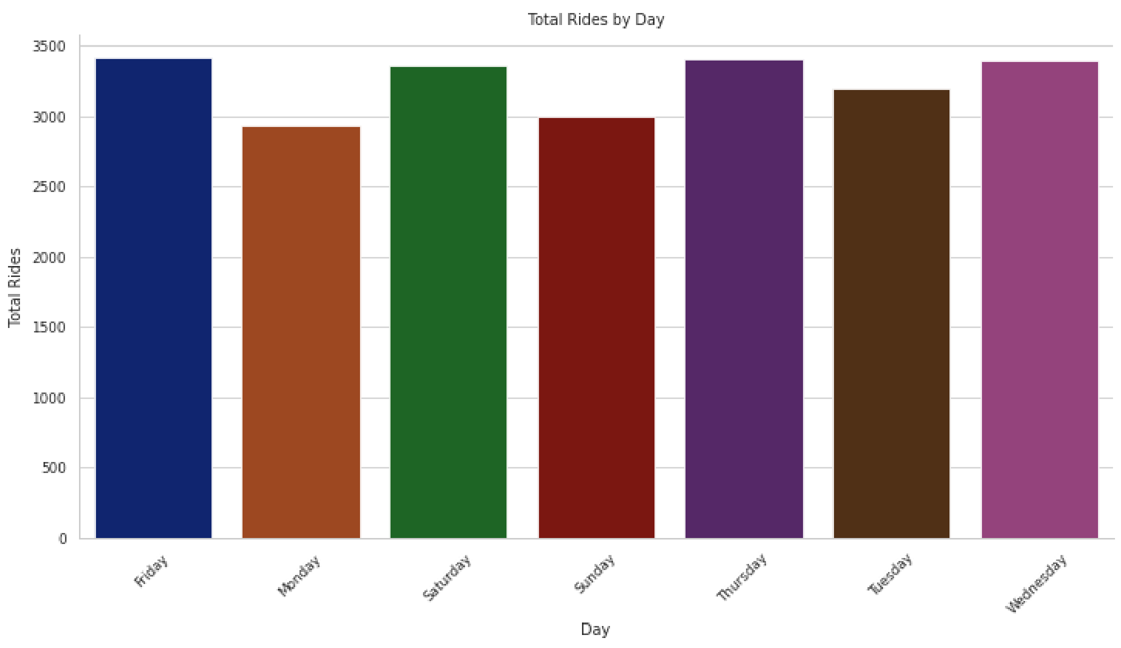
Plot total ride count by day

Repeat the above process, but now calculate the total rides by day of the week.

[239]:
```
# Repeat the above process, this time for rides by day
#==> ENTER YOUR CODE HERE

daily_rides = df['day'].value_counts()
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
 ↪'Saturday', 'Sunday']
daily_rides = daily_rides.reindex(index=day_order)
daily_rides
```

[239]:
```
Monday       2931
Tuesday      3198
Wednesday    3390
Thursday     3402
Friday       3413
Saturday     3367
Sunday       2998
Name: day, dtype: int64
```

[240]:
```
# Create bar plot for ride count by day
#==> ENTER YOUR CODE HERE

# Set the style and context for seaborn
sns.set_style("whitegrid")
sns.set_context("paper")

# Create bar plot of rides by day
plt.figure(figsize=(12, 6))
ax = sns.barplot(x=rides_by_day.index, y=rides_by_day.values, palette='dark')
ax.set_xticklabels(rides_by_day.index, rotation=45)
plt.xlabel('Day')
plt.ylabel('Total Rides')
plt.title('Total Rides by Day')

# Remove spines
sns.despine()

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



Plot total revenue by day of the week

Repeat the above process, but now calculate the total revenue by day of the week.

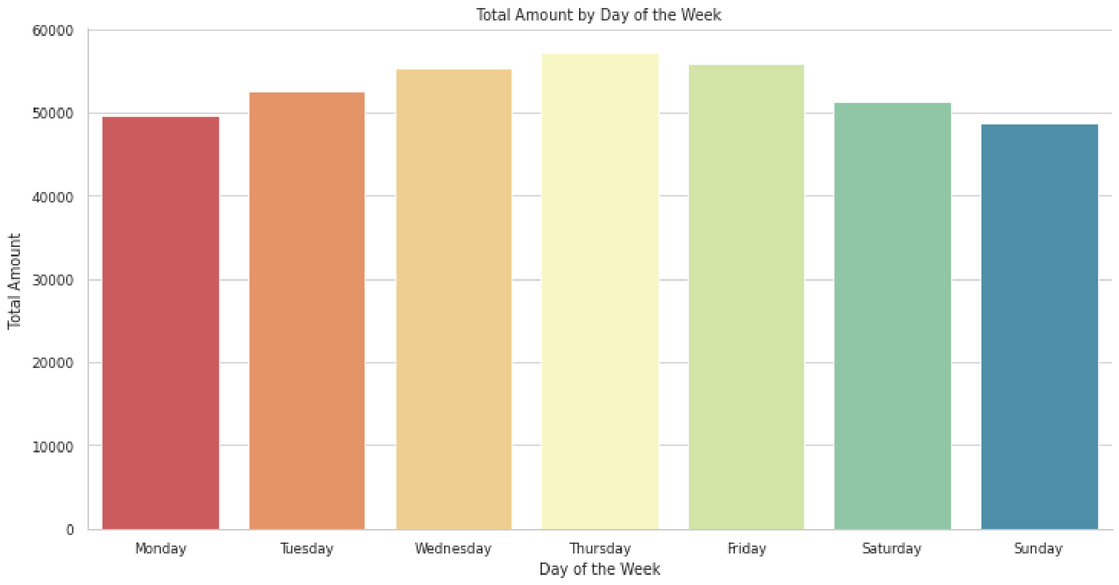
[241]:
```
# Repeat the process, this time for total revenue by day
#==> ENTER YOUR CODE HERE
total_amount_day = df.groupby('day')['total_amount'].sum()
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
 ↪'Saturday', 'Sunday']
total_amount_day = daily_revenue.reindex(index=day_order)
total_amount_day
```

[241]:
```
day
Monday       49574.37
Tuesday      52527.14
Wednesday    55310.47
Thursday     57181.91
Friday       55818.74
Saturday     51195.40
Sunday       48624.06
Name: total_amount, dtype: float64
```

[ ]:
```

```

[248]:
```
# Create bar plot of total revenue by day
#==> ENTER YOUR CODE HERE
```

```
# Set the style and context for seaborn
sns.set_style("whitegrid")
sns.set_context("paper")


# Create bar plot of total revenue by day
plt.figure(figsize=(12, 6))
ax = sns.barplot(x=total_amount_day.index, y=daily_revenue.values,␣
 ↪palette='Spectral')
plt.xlabel('Day of the Week')
plt.ylabel('Total Amount')
plt.title('Total Amount by Day of the Week')

# Remove spines
sns.despine()

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

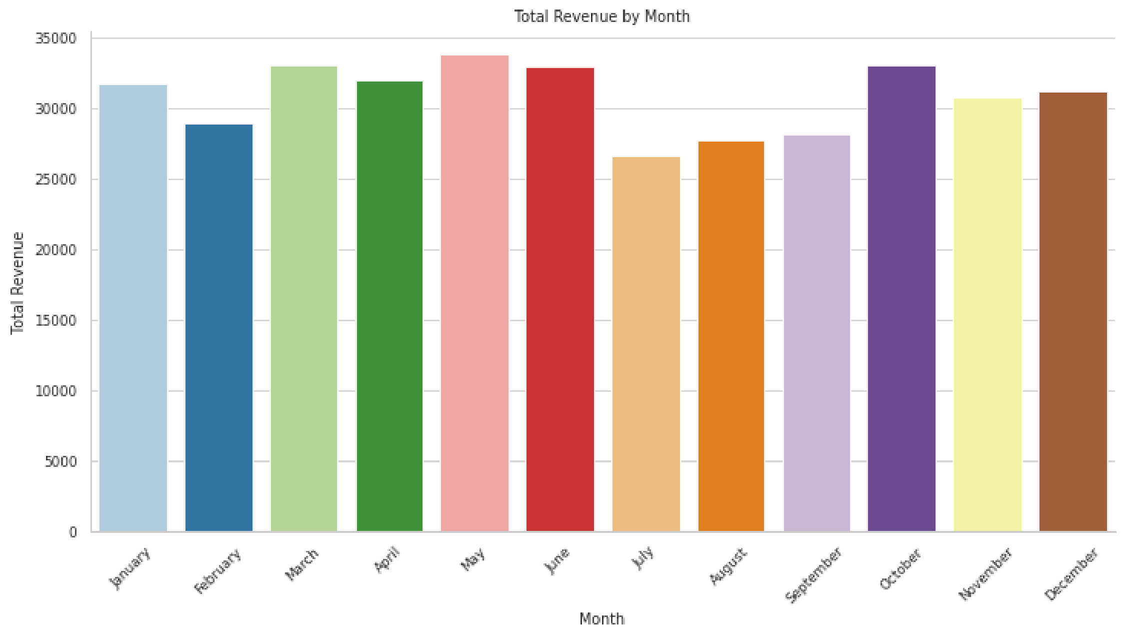


Plot total revenue by month

[243]:
```
# Repeat the process, this time for total revenue by month
#==> ENTER YOUR CODE HERE

total_amount_month = df.groupby('month').sum()[['total_amount']]
total_amount_month = total_amount_month.reindex(index=month_order)
total_amount_month
```

[243]:
```
           total_amount
month
January         31735.25
February        28937.89
March           33085.89
April           32012.54
May             33828.58
June            32920.52
July            26617.64
August          27759.56
September       28206.38
October         33065.83
November        30800.44
December        31261.57
```

[255]:
```python
# Create a bar plot of total revenue by month
#==> ENTER YOUR CODE HERE



# Set the style and context for seaborn
sns.set_style("whitegrid")
sns.set_context("paper")

amount_month = total_amount_month.reindex(index=month_order)

# Create bar plot of total revenue by month
plt.figure(figsize=(12, 6))
ax = sns.barplot(x=total_amount_month.index, y=total_amount_month.values,␣
 ↪palette='Paired')
plt.xlabel('Month')
plt.ylabel('Total Revenue')
plt.title('Total Revenue by Month')

# Rotate x-tick labels
plt.xticks(rotation=45)

# Remove spines
sns.despine()

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



Scatter plot You can create a scatterplot in Tableau Public, which can be easier to manipulate and present. If you'd like step by step instructions, you can review the following link. Those instructions create a scatterplot showing the relationship between total_amount and trip_distance. Consider adding the Tableau visualization to your executive summary, and adding key insights from your findings on those two variables.

Tableau visualization guidelines

Plot mean trip distance by drop-off location

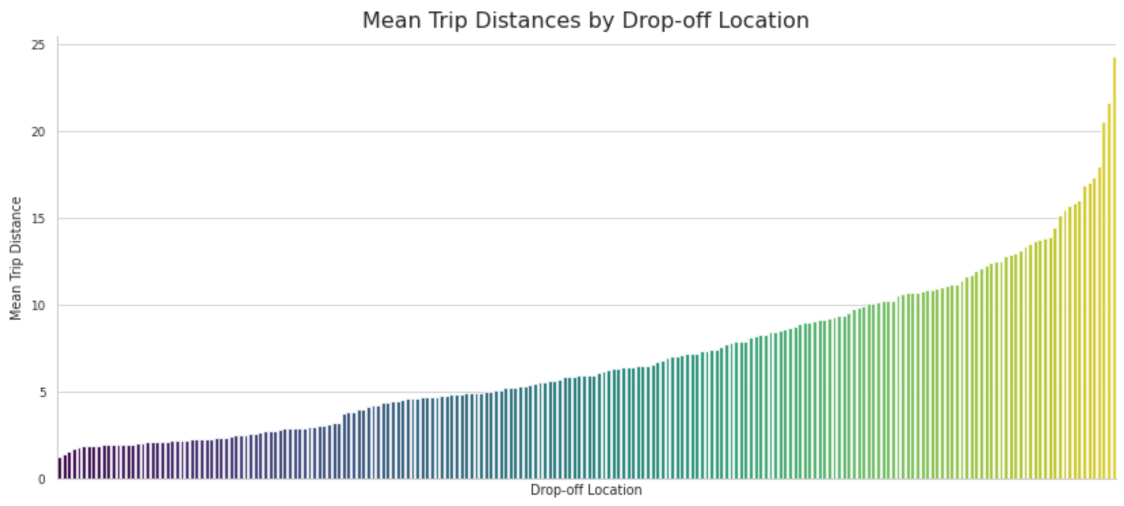
[260]:
```
# Get number of unique drop-off location IDs
#==> ENTER YOUR CODE HERE

num_dropoff_locs = df['DOLocationID'].nunique()
num_dropoff_locs
```

[260]:
```
216
```

[261]:
```
# Calculate the mean trip distance for each drop-off location
#==> ENTER YOUR CODE HERE
# Calculate the mean trip distance for each drop-off location
mean_trip_distance = df.groupby('DOLocationID')['trip_distance'].mean()

# Sort the results in descending order by mean trip distance
mean_trip_distance = mean_trip_distance.sort_values(ascending=False)
mean_trip_distance
```

[261]:
```
DOLocationID
23     24.275000
29     21.650000
210    20.500000
11     17.945000
51     17.310000
         …
137     1.818852
234     1.727806
237     1.555494
193     1.390556
207     1.200000
Name: trip_distance, Length: 216, dtype: float64
```

[269]:
```
# Create a bar plot of mean trip distances by drop-off location in ascending␣
 ↪order by distance
#==> ENTER YOUR CODE HERE



# Set the style and context for seaborn
sns.set_style("whitegrid")
sns.set_context("paper")

# Sort the results in ascending order by mean trip distance
distance_by_dropoff = distance_by_dropoff.sort_values(by='trip_distance')

# Create bar plot of mean trip distances by drop-off location
plt.figure(figsize=(14, 6))
ax = sns.barplot(x=distance_by_dropoff.index,␣
 ↪y=distance_by_dropoff['trip_distance'], order=distance_by_dropoff.index,␣
 ↪palette='viridis')
plt.xlabel('Drop-off Location')
plt.ylabel('Mean Trip Distance')
plt.title('Mean Trip Distances by Drop-off Location', fontsize=16)

# Remove x-axis labels and ticks
ax.set_xticklabels([])
ax.set_xticks([])

# Remove spines
sns.despine()

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

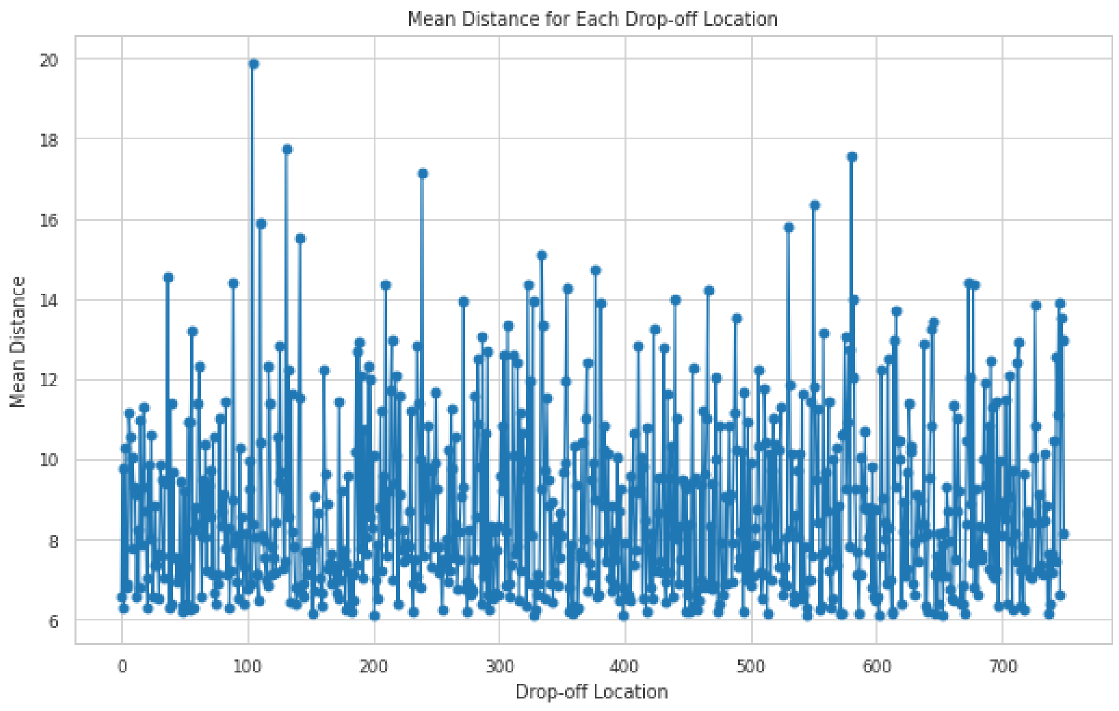


4.4 BONUS CONTENT

To confirm your conclusion, consider the following experiment: 1. Create a sample of coordinates from a normal distribution—in this case 1,500 pairs of points from a normal distribution with a mean of 10 and a standard deviation of 5 2. Calculate the distance between each pair of coordinates 3. Group the coordinates by endpoint and calculate the mean distance between that endpoint and all other points it was paired with 4. Plot the mean distance for each unique endpoint

[271]:
```
#BONUS CONTENT

#1. Generate random points on a 2D plane from a normal distribution
#==> ENTER YOUR CODE HERE

# 2. Calculate Euclidean distances between points in first half and second half␣
↪of array
#==> ENTER YOUR CODE HERE

# 3. Group the coordinates by "drop-off location", compute mean distance
#==> ENTER YOUR CODE HERE

# 4. Plot the mean distance between each endpoint ("drop-off location") and all␣
↪points it connected to
#==> ENTER YOUR CODE HERE

from scipy.spatial.distance import cdist

# Set the seed for reproducibility
np.random.seed(42)
```

```
# Generate random points on a 2D plane from a normal distribution
mean = 10
std = 5
num_points = 1500
points = np.random.normal(mean, std, (num_points, 2))

# Calculate Euclidean distances between points in first half and second half of␣
 ↪array
half = num_points // 2
distances = cdist(points[:half], points[half:])

# Group the coordinates by "drop-off location" and compute mean distance
drop_off_locations = np.arange(num_points // 2)
mean_distances = np.mean(distances, axis=1)

# Plot the mean distance between each endpoint ("drop-off location") and all␣
 ↪points it connected to
plt.figure(figsize=(10, 6))
plt.plot(drop_off_locations, mean_distances, marker='o')
plt.xlabel('Drop-off Location')
plt.ylabel('Mean Distance')
plt.title('Mean Distance for Each Drop-off Location')
plt.grid(True)

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



Histogram of rides by drop-off location

First, check to whether the drop-off locations IDs are consecutively numbered. For instance, does it go 1, 2, 3, 4..., or are some numbers missing (e.g., 1, 3, 4...). If numbers aren't all consecutive, the histogram will look like some locations have very few or no rides when in reality there's no bar because there's no location.

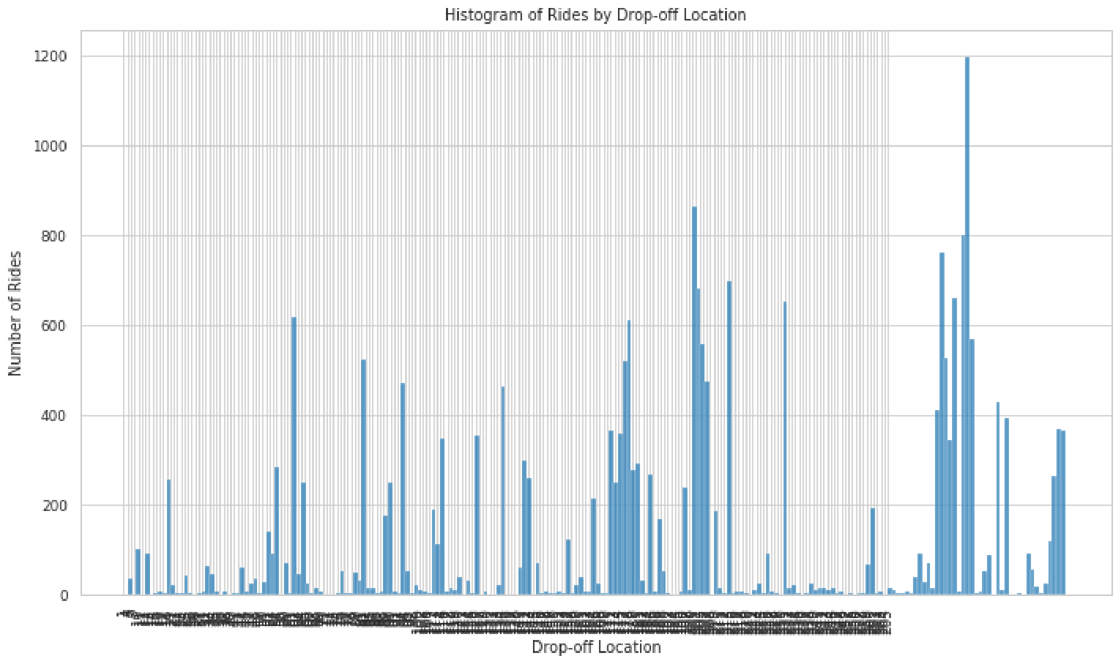
[272]:
```
# Check if all drop-off locations are consecutively numbered
#==> ENTER YOUR CODE HERE
drop_off_locations = df['DOLocationID'].unique()
consecutive_numbers = np.arange(np.min(drop_off_locations), np.
 ↪max(drop_off_locations)+1)
missing_numbers = np.setdiff1d(consecutive_numbers, drop_off_locations)

if len(missing_numbers) == 0:
    print("All drop-off locations are consecutively numbered.")
else:
    print("Some drop-off locations are missing.")
    print("Missing numbers:", missing_numbers)
```

```
Some drop-off locations are missing.
Missing numbers: [  2   3   5   6   8  20  27  30  44  46  57  58  59  84  96
99 101 103
 104 105 108 109 110 111 115 122 128 139 154 155 156 165 167 172 176 185
 187 191 199 203 204 206 214 221 245 250 251 253 254]
```

To eliminate the spaces in the historgram that these missing numbers would create, sort the unique drop-off location values, then convert them to strings. This will make the histplot function display all bars directly next to each other.

[273]:
```
#==> ENTER YOUR CODE HERE
# DOLocationID column is numeric, so sort in ascending order
#==> ENTER YOUR CODE HERE

# Convert to string
#==> ENTER YOUR CODE HERE

# Plot
#==> ENTER YOUR CODE HERE

# Sort the unique drop-off location values in ascending order
sorted_drop_off_locations = sorted(df['DOLocationID'].unique())

# Convert the sorted drop-off location values to strings
sorted_drop_off_locations = [str(location) for location in␣
 ↪sorted_drop_off_locations]
```

```
# Plot the histogram using the sorted and converted drop-off location values
plt.figure(figsize=(12, 6))
sns.histplot(data=df, x='DOLocationID', bins=len(sorted_drop_off_locations))
plt.xticks(range(len(sorted_drop_off_locations)), sorted_drop_off_locations,␣
 ↪rotation=90)
plt.xlabel('Drop-off Location')
plt.ylabel('Number of Rides')
plt.title('Histogram of Rides by Drop-off Location')
plt.tight_layout()
plt.show()
```



4.5 PACE: Execute

Consider the questions in your PACE Strategy Document to reflect on the Execute stage.

4.5.1 Task 4a. Results and evaluation

Having built visualizations in Tableau and in Python, what have you learned about the dataset? What other questions have your visualizations uncovered that you should pursue?

Pro tip: Put yourself in your client's perspective, what would they want to know?

Use the following code fields to pursue any additional EDA based on the visualizations you've already plotted. Also use the space to make sure your visualizations are clean, easily understandable, and accessible.

Ask yourself: Did you consider color, contrast, emphasis, and labeling?

Based on the EDA conducted, the findings can be summarized as follows:

1. I have learned:
    - The majority of rides in the dataset have a trip distance less than 10 miles.
    - Most rides have a total amount less than $50, with a few outliers indicating expensive rides.
    - Tip amounts are typically low, but there are instances of higher tips.
    - Trip distances vary across different drop-off locations.
    - Ride volume and revenue show variations throughout the year and across different days of the week.
2. My other questions are:
    - What factors contribute to longer-distance rides and expensive fares?
    - Are there specific characteristics associated with rides that receive higher tip amounts?
    - What factors influence the variations in trip distances based on drop-off location?
    - What are the patterns in ride demand and revenue during different times of the day?
    - How does the average fare amount vary based on trip distance or drop-off location?
    - Are there correlations between trip distance, total amount, and tip amount?
    - Can customer demographics provide insights into ride patterns and behaviors?
3. My client would likely want to know:
    - Insights into factors influencing trip distances, fare amounts, and tipping behavior.
    - Understanding seasonal patterns and peak periods for ride demand and revenue.
    - Identification of locations with consistently longer or shorter trips.
    - Analysis of customer demographics and their impact on ride patterns.
    - Recommendations to optimize revenue and improve customer experience based on the findings.

These findings and questions provide a foundation for further analysis and can guide decision-making for the client in terms of operational improvements, marketing strategies, and customer satisfaction enhancements.

[274]:
```
df['trip_duration'] = (df['tpep_dropoff_datetime']-df['tpep_pickup_datetime'])
```

[275]:
```
df.head(10)
```

[275]:
```
   Unnamed: 0  VendorID tpep_pickup_datetime tpep_dropoff_datetime  \
0    24870114         2  2017-03-25 08:55:43   2017-03-25 09:09:47
1    35634249         1  2017-04-11 14:53:28   2017-04-11 15:19:58
2   106203690         1  2017-12-15 07:26:56   2017-12-15 07:34:08
3    38942136         2  2017-05-07 13:17:59   2017-05-07 13:48:14
4    30841670         2  2017-04-15 23:32:20   2017-04-15 23:49:03
5    23345809         2  2017-03-25 20:34:11   2017-03-25 20:42:11
6    37660487         2  2017-05-03 19:04:09   2017-05-03 20:03:47
7    69059411         2  2017-08-15 17:41:06   2017-08-15 18:03:05
8     8433159         2  2017-02-04 16:17:07   2017-02-04 16:29:14
9    95294817         1  2017-11-10 15:20:29   2017-11-10 15:40:55
```

```
   passenger_count  trip_distance  RatecodeID store_and_fwd_flag  \
0                6           3.34           1                  N   
1                1           1.80           1                  N   
2                1           1.00           1                  N   
3                1           3.70           1                  N   
4                1           4.37           1                  N   
5                6           2.30           1                  N   
6                1          12.83           1                  N   
7                1           2.98           1                  N   
8                1           1.20           1                  N   
9                1           1.60           1                  N   

   PULocationID  DOLocationID  …  fare_amount  extra  mta_tax  tip_amount  \
0           100           231  …         13.0    0.0      0.5        2.76   
1           186            43  …         16.0    0.0      0.5        4.00   
2           262           236  …          6.5    0.0      0.5        1.45   
3           188            97  …         20.5    0.0      0.5        6.39   
4             4           112  …         16.5    0.5      0.5        0.00   
5           161           236  …          9.0    0.5      0.5        2.06   
6            79           241  …         47.5    1.0      0.5        9.86   
7           237           114  …         16.0    1.0      0.5        1.78   
8           234           249  …          9.0    0.0      0.5        0.00   
9           239           237  …         13.0    0.0      0.5        2.75   

   tolls_amount  improvement_surcharge  total_amount     month        day  \
0           0.0                    0.3         16.56     March   Saturday   
1           0.0                    0.3         20.80     April    Tuesday   
2           0.0                    0.3          8.75  December     Friday   
3           0.0                    0.3         27.69       May     Sunday   
4           0.0                    0.3         17.80     April   Saturday   
5           0.0                    0.3         12.36     March   Saturday   
6           0.0                    0.3         59.16       May  Wednesday   
7           0.0                    0.3         19.58    August    Tuesday   
8           0.0                    0.3          9.80  February   Saturday   
9           0.0                    0.3         16.55  November     Friday   

    trip_duration  
0 0 days 00:14:04  
1 0 days 00:26:30  
2 0 days 00:07:12  
3 0 days 00:30:15  
4 0 days 00:16:43  
5 0 days 00:08:00  
6 0 days 00:59:38  
7 0 days 00:21:59  
8 0 days 00:12:07  
9 0 days 00:20:26  
```

```
[10 rows x 21 columns]
```

4.5.2 Task 4b. Conclusion

Make it professional and presentable

You have visualized the data you need to share with the director now. Remember, the goal of a data visualization is for an audience member to glean the information on the chart in mere seconds.

Questions to ask yourself for reflection: Why is it important to conduct Exploratory Data Analysis? Why are the data visualizations provided in this notebook useful?

EDA is important because … it allows us to gain a deeper understanding of the dataset and uncover meaningful insights. Here are some reasons why EDA is important:

1. Identify data quality issues: EDA helps us identify missing values, outliers, inconsistencies, and other data quality issues that may affect the accuracy and reliability of our analysis.
2. Understand the distribution of variables: EDA allows us to examine the distribution of variables, such as trip distance, total amount, and tip amount. This helps us identify patterns, trends, and potential relationships between variables.
3. Discover outliers and anomalies: EDA helps us identify outliers, which are data points that deviate significantly from the majority of the data. Outliers can provide valuable insights or indicate data entry errors or unusual events.
4. Detect patterns and relationships: EDA helps us uncover patterns and relationships between variables. This can help us understand the underlying factors that drive certain outcomes and make informed decisions based on the findings.
5. Validate assumptions: EDA allows us to validate our initial assumptions and hypotheses about the data. It helps us confirm or challenge our preconceived notions and ensures that our analysis is based on solid evidence.

Visualizations helped me understand ..

1. Trip Distance Distribution: The histogram and box plot of trip distance provided insights into the distribution of trip distances. I could observe that the majority of trips had distances less than 10 miles, with a right-skewed distribution indicating the presence of some longer trips as outliers.
2. Total Amount Distribution: The histogram and box plot of total amount revealed the distribution of fares paid by passengers. The distribution appeared to be right-skewed, indicating that most fares were relatively low, with a few higher fares as outliers.
3. Tip Amount Distribution: The histogram and box plot of tip amount illustrated the distribution of tips given by passengers. The distribution showed a peak around zero, indicating that a significant portion of rides did not receive any tip. However, there were also instances of generous tips, as evidenced by the presence of outliers with higher tip amounts.
4. Mean Tips by Passenger Count: The bar plot of mean tips by passenger count showed the average tip amount received based on the number of passengers in a ride. It provided insights

into the tipping behavior based on the passenger count, with rides having a single passenger receiving relatively higher average tips compared to rides with multiple passengers.

5. Total Rides by Month and Day: The bar plots of total rides by month and day provided an overview of the ride count distribution across different months and days of the week. It allowed me to identify any variations or trends in ride demand based on the time of the year or specific days.

You've now completed professional data visualizations according to a business need. Well done!

[ ]:
```

```