Activity_Course 3 Automatidata project lab

July 24, 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?

One of the best methods for identifying outliers is crafting a boxplot.

Keeping or excluding the outliers depends on the outliers kind whether it is global, contextual or collective.

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.

[139]:
```python
# Import packages and libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

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.

[140]:
```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?

There is no missing data in this dataset.

There are data outliers in columns such as [fare_amount, extra, mta_tax, improvement_surcharge and total_amount] where the minimum values are negative.

Start by discovering, using head and size.

[141]:
```
df.head(10)
```

[141]:
```
   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
```

[142]:
```
df.size
```

[142]:
```
408582
```

Use describe...

[143]:
```
df.describe()
```

[143]:
```
         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.

[144]:
```python
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

Bar chart and Box plot will be very helpful in presenting the dataset.

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

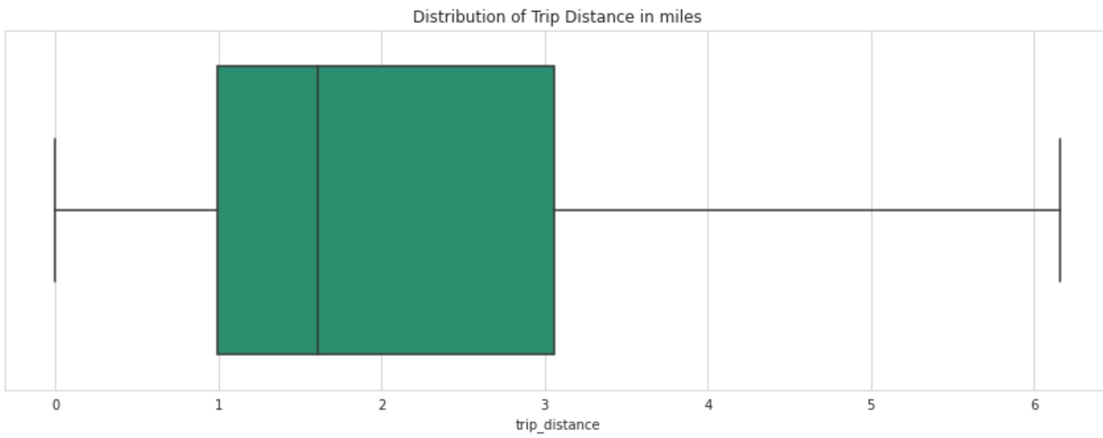
Perform a check for outliers on relevant columns such as trip distance and trip duration. Remember, some of the best ways to identify the presence of outliers in data are box plots and histograms.

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

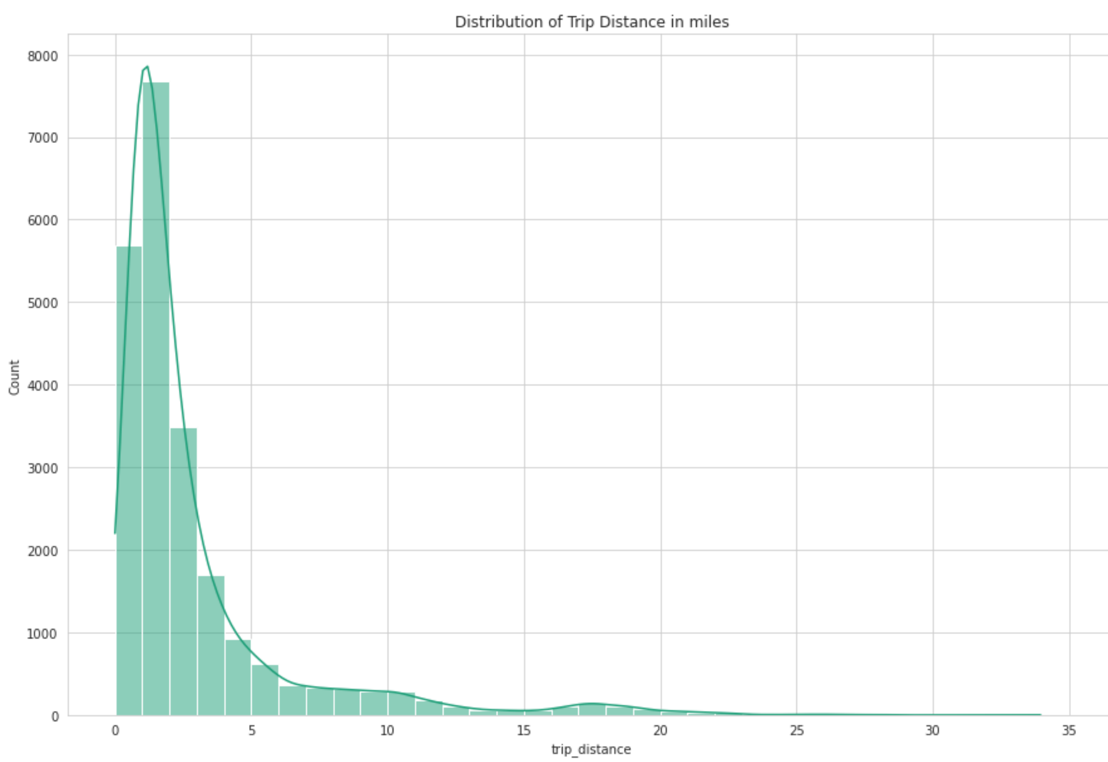
[145]:
```
# Convert data columns to datetime
df.tpep_pickup_datetime = pd.to_datetime(df.tpep_pickup_datetime)
df.tpep_dropoff_datetime = pd.to_datetime(df.tpep_dropoff_datetime)
```

trip distance

[208]:
```
# Create box plot of trip_distance
plt.figure(figsize = (15, 5))
sns.set_style("whitegrid")
sns.set_palette("Dark2")
sns.boxplot(data= df, x= "trip_distance", showfliers= False)
plt.title("Distribution of Trip Distance in miles")
plt.show()
```

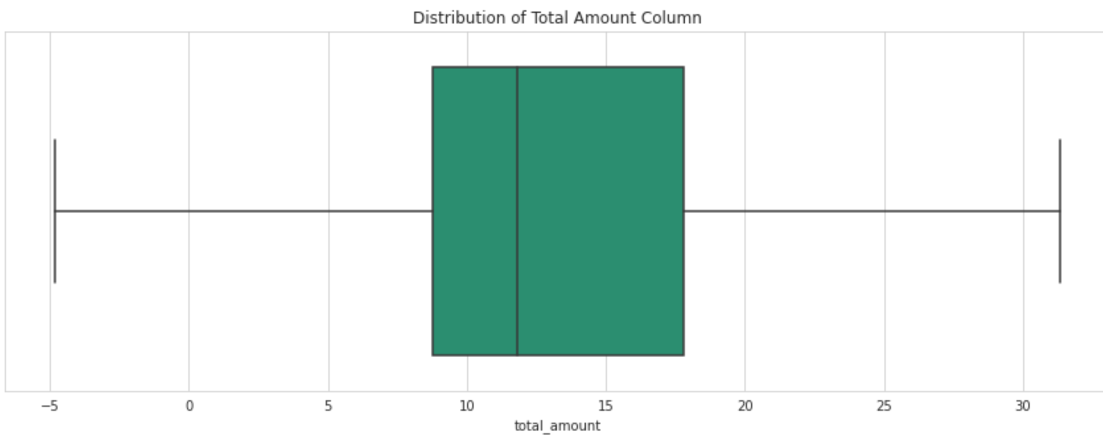


[210]:
```
# Create histogram of trip_distance
plt.figure(figsize = (15, 10))
sns.set_style("whitegrid")
sns.set_palette("Dark2")
sns.histplot(data= df, x= "trip_distance", binwidth= 1, binrange= (0, 35), kde=␣
↪True)
plt.title("Distribution of Trip Distance in miles")
plt.show()
```

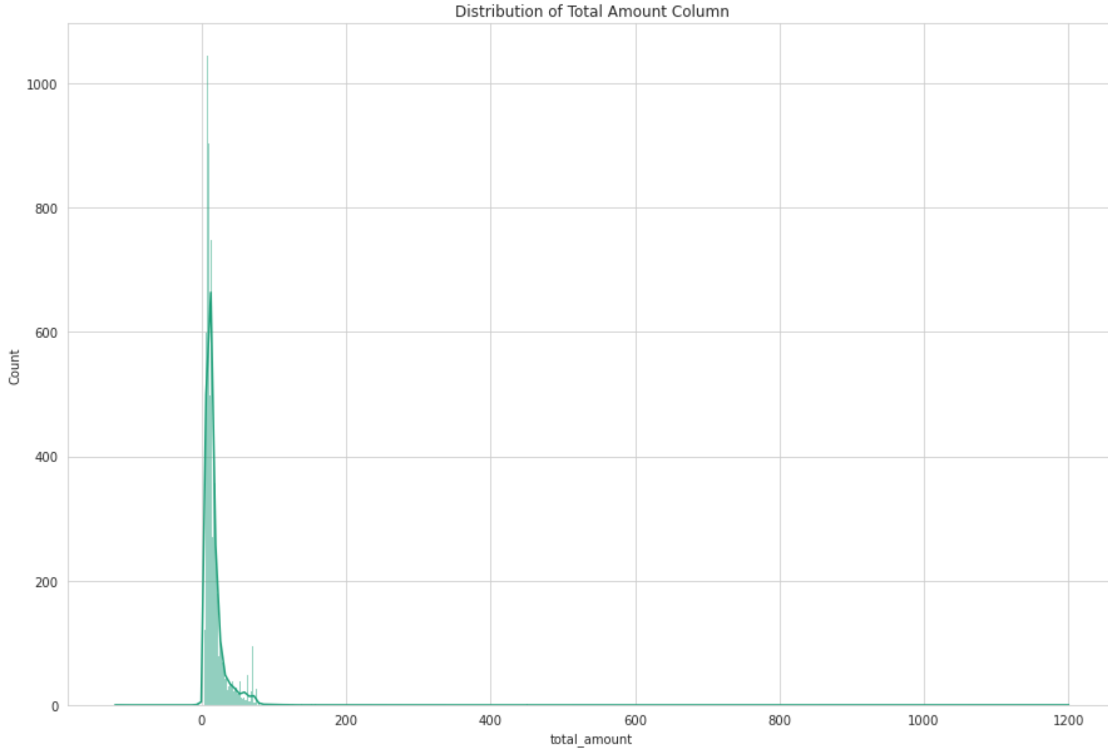


total amount

[211]:
```
# Create box plot of total_amount
plt.figure(figsize = (15, 5))
sns.boxplot(data= df, x= "total_amount", showfliers= False)
plt.title("Distribution of Total Amount Column")
plt.show()
```

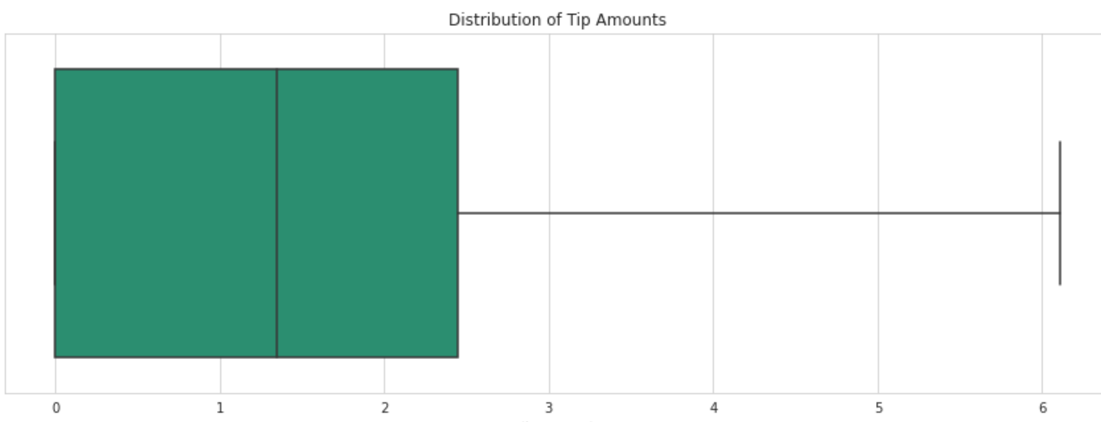


[212]:
```
# Create histogram of total_amount
plt.figure(figsize = (15, 10))
sns.histplot(data= df, x= "total_amount", binwidth= 0.5, kde= True)
plt.title("Distribution of Total Amount Column")
plt.show()
```

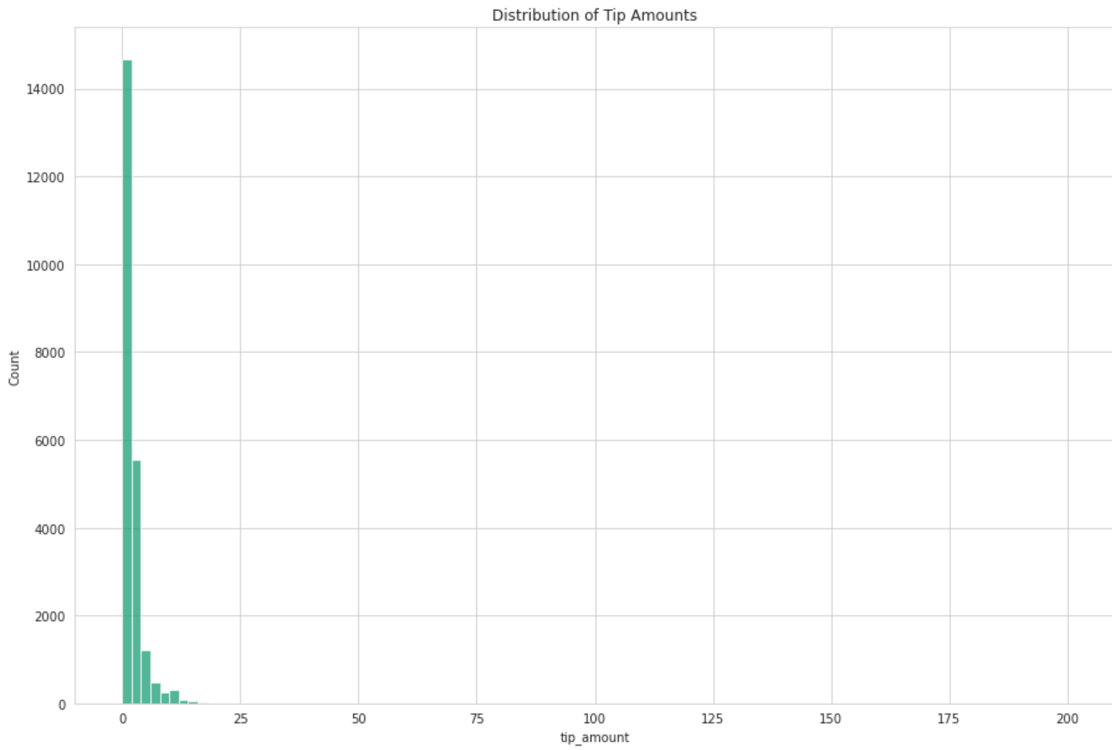


tip amount

[213]:
```
# Create box plot of tip_amount
plt.figure(figsize = (15, 5))
sns.boxplot(data= df, x= "tip_amount", showfliers= False)
plt.title("Distribution of Tip Amounts")
plt.show()
```

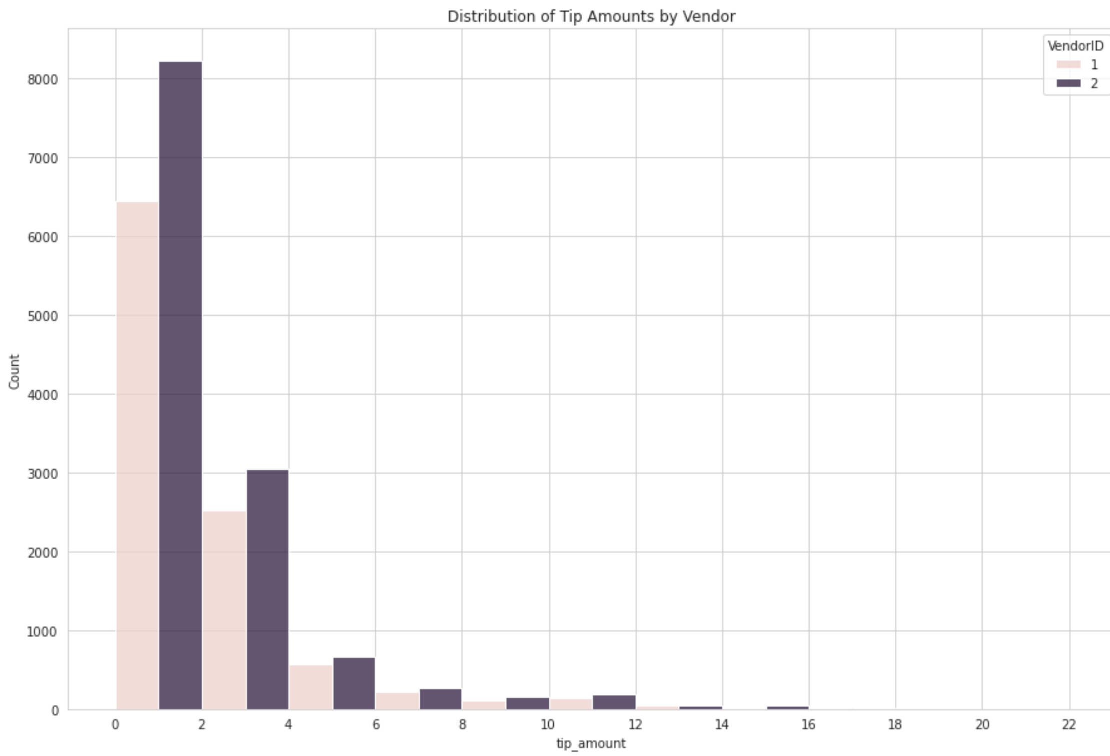


[214]:
```
# Create histogram of tip_amount
plt.figure(figsize = (15, 10))
sns.histplot(data= df, x= "tip_amount", binwidth= 2)
plt.title("Distribution of Tip Amounts")
plt.show()
```



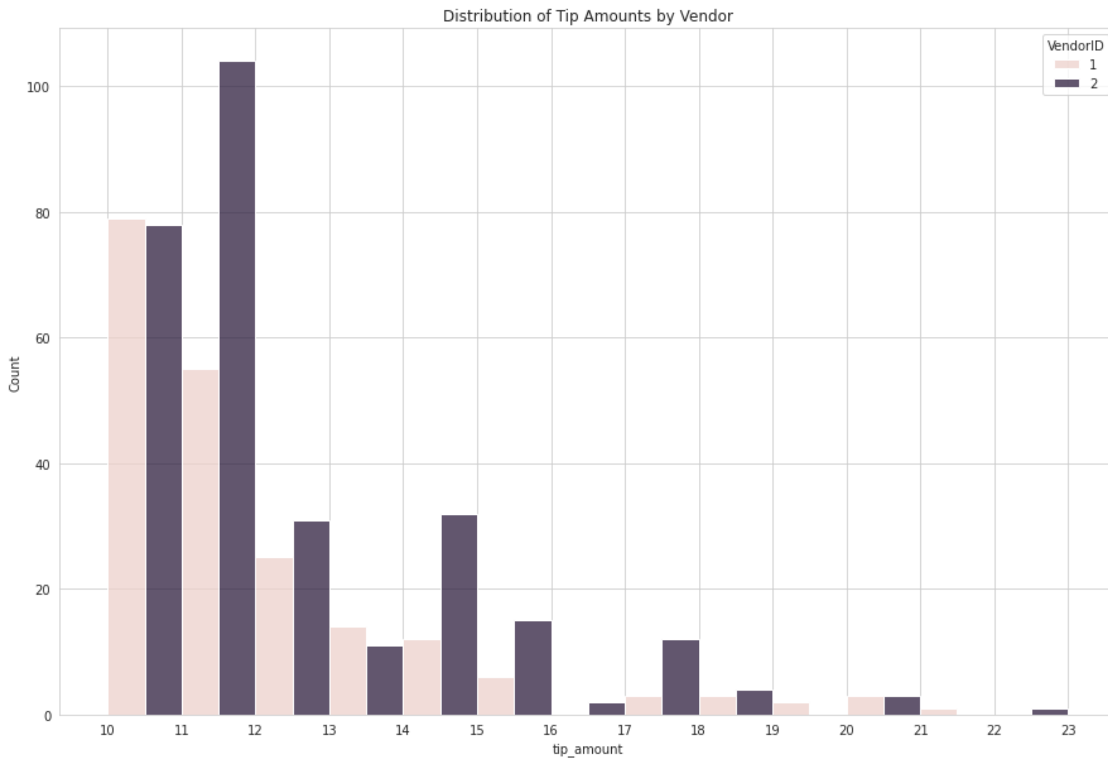
tip_amount by vendor

[215]:
```
# Create histogram of tip_amount by vendor
plt.figure(figsize = (15, 10))
ax= sns.histplot(data= df, x= "tip_amount", bins= range(0, 24, 2), hue=␣
↪"VendorID", multiple= "dodge")

ax.set_xticks(range(0, 24, 2))
ax.set_xticklabels(range(0, 24, 2))
plt.title("Distribution of Tip Amounts by Vendor")
plt.show()
```



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.

[216]:
```
# Create histogram of tip_amount by vendor for tips > $10
plt.figure(figsize = (15, 10))
ax= sns.histplot(data= df, x= "tip_amount", bins= range(10, 24, 1), hue=␣
↪"VendorID", multiple= "dodge")

ax.set_xticks(range(10, 24, 1))
ax.set_xticklabels(range(10, 24, 1))
plt.title("Distribution of Tip Amounts by Vendor")
plt.show()
```



Mean tips by passenger count

Examine the unique values in the `passenger_count` column.

[217]:
```
df["passenger_count"].value_counts()
```

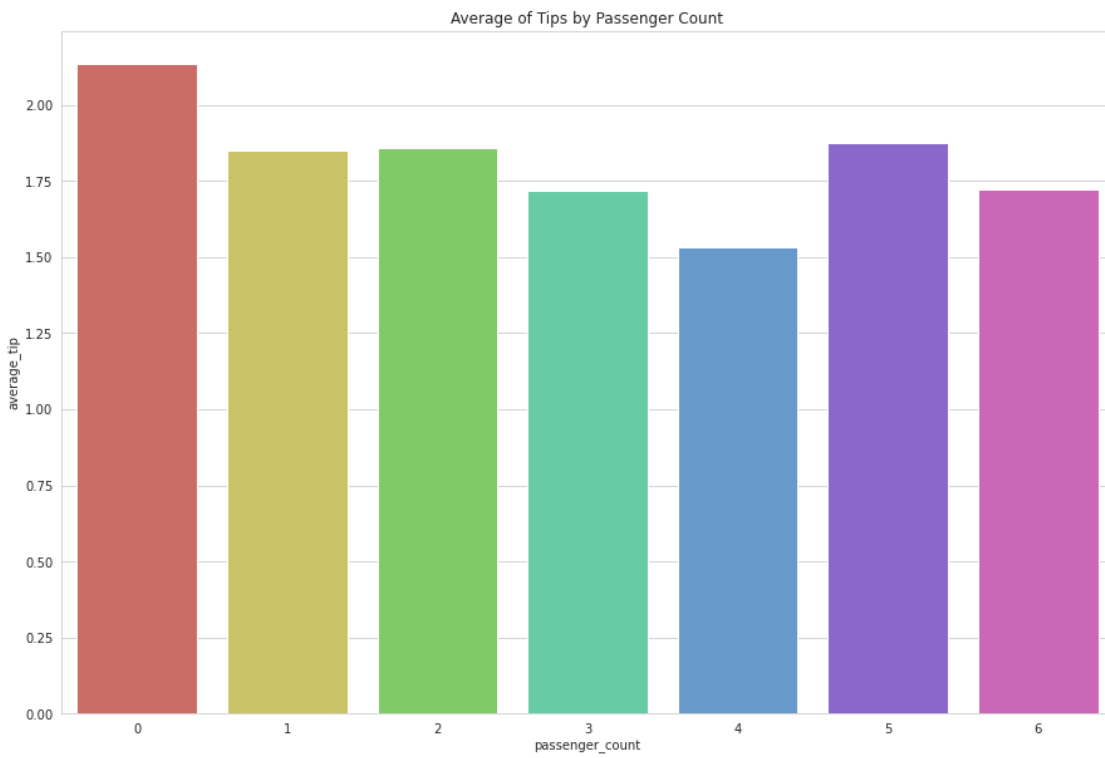
[217]:
```
1    16117
2     3305
5     1143
3      953
6      693
4      455
0       33
Name: passenger_count, dtype: int64
```

[218]:
```
# Calculate mean tips by passenger_count
mean_tips_by_pass = df.groupby(["passenger_count"])[["tip_amount"]].mean().
 ↪reset_index().rename(columns= {"tip_amount": "average_tip"})
mean_tips_by_pass
```

[218]:
```
   passenger_count  average_tip
0                0     2.135758
1                1     1.848920
2                2     1.856378
```

```
3                  3      1.716768
4                  4      1.530264
5                  5      1.873185
6                  6      1.720260
```

[219]:
```
# Create bar plot for mean tips by passenger count
plt.figure(figsize = (15, 10))
sns.barplot(data= mean_tips_by_pass, x= "passenger_count", y= "average_tip",␣
↪palette= "hls")
plt.title("Average of Tips by Passenger Count")
plt.show()
```



Create month and day columns

[220]:
```
# Create a month column
df["tpep_pickup_month"] = df["tpep_pickup_datetime"].dt.month_name().str[:3]

# Create a day column
df["tpep_pickup_day"] = df["tpep_pickup_datetime"].dt.day_name()

df.head()
```

[220]:
```
   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

   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

   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

   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

  tpep_pickup_month tpep_pickup_day
0               Mar        Saturday
1               Apr         Tuesday
2               Dec          Friday
3               May          Sunday
4               Apr        Saturday
```

Plot total ride count by month

Begin by calculating total ride count by month.

[221]:
```
# Get total number of rides for each month
df_by_month = df.groupby(["tpep_pickup_month"])[["Unnamed: 0"]].count().
↪reset_index().rename(columns= {"Unnamed: 0": "total_rides"})
df_by_month
```

[221]:
```
   tpep_pickup_month  total_rides
0                Apr         2019
1                Aug         1724
2                Dec         1863
```

```
3                 Feb        1769
4                 Jan        1997
5                 Jul        1697
6                 Jun        1964
7                 Mar        2049
8                 May        2013
9                 Nov        1843
10                Oct        2027
11                Sep        1734
```

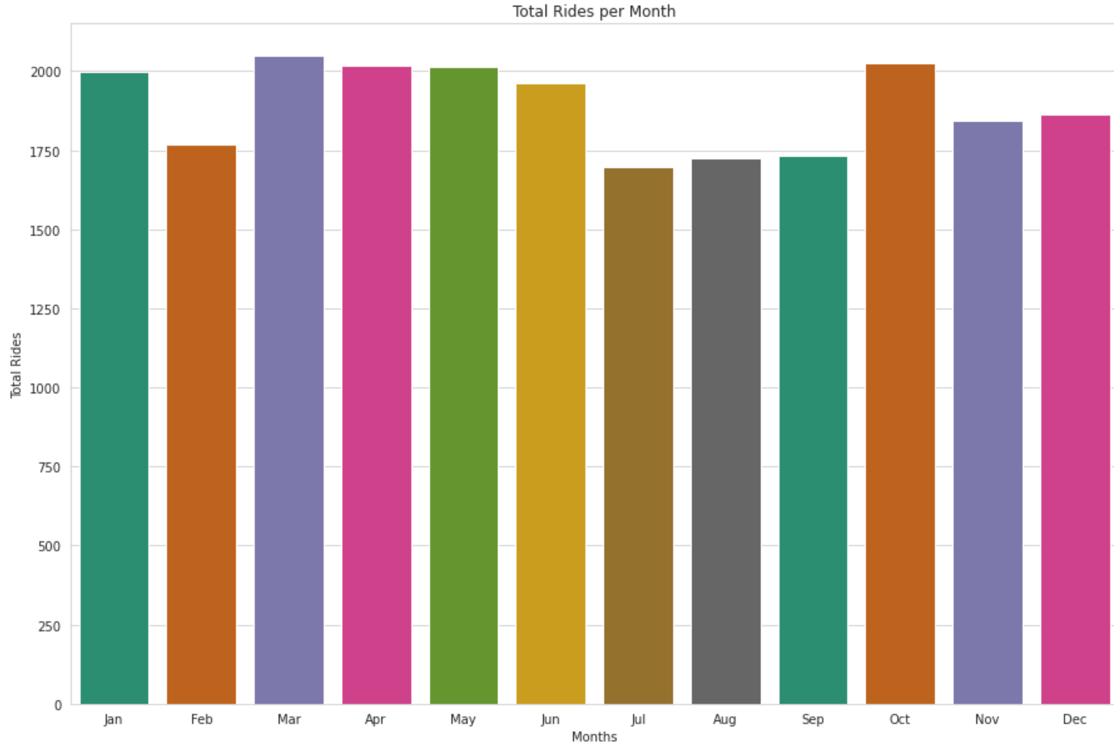
Reorder the results to put the months in calendar order.

[222]:
```
# Reorder the monthly ride list so months go in order
months_order = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep",␣
↪"Oct", "Nov", "Dec"]
```

[223]:
```
# Show the index
months_order
```

[223]:
```
['Jan',
 'Feb',
 'Mar',
 'Apr',
 'May',
 'Jun',
 'Jul',
 'Aug',
 'Sep',
 'Oct',
 'Nov',
 'Dec']
```

[224]:
```
# Create a bar plot of total rides per month
plt.figure(figsize = (15, 10))
sns.barplot(data= df_by_month, x= "tpep_pickup_month", y= "total_rides",␣
↪palette= "Dark2", order= months_order)

plt.title("Total Rides per Month")
plt.xlabel("Months")
plt.ylabel("Total Rides")
plt.show()
```



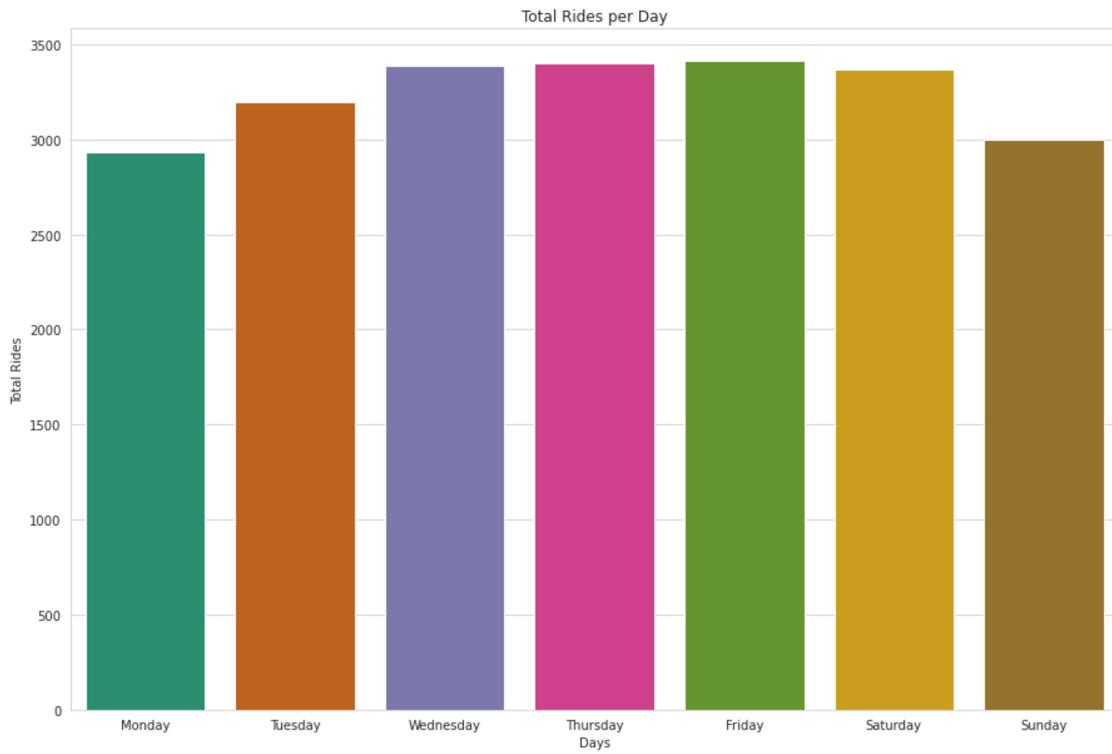
Plot total ride count by day

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

[225]:
```
# Repeat the above process, this time for rides by day
df_by_day = df.groupby(["tpep_pickup_day"])[["Unnamed: 0"]].count().
↪reset_index().rename(columns= {"Unnamed: 0": "total_rides"})
days_order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday",␣
↪"Saturday", "Sunday"]
days_order
```

[225]:
```
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
```

[226]:
```
# Create bar plot for ride count by day
plt.figure(figsize = (15, 10))
sns.barplot(data= df_by_day, x= "tpep_pickup_day", y= "total_rides", palette=␣
↪"Dark2", order= days_order)

plt.title("Total Rides per Day")
plt.xlabel("Days")
plt.ylabel("Total Rides")
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.

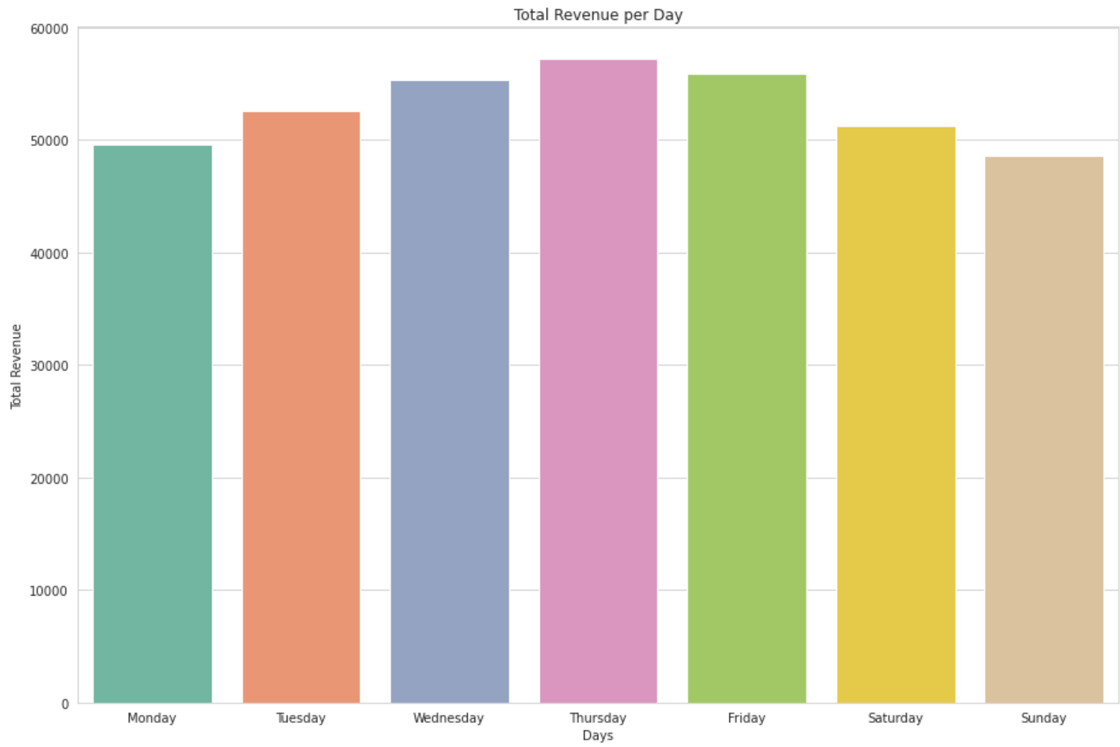
[227]:
```
# Repeat the process, this time for total revenue by day
revenue_by_day = df.groupby(["tpep_pickup_day"])[["total_amount"]].sum().
    ↪reset_index().rename(columns= {"total_amount": "total_day_revenue"})
revenue_by_day
```

[227]:

	tpep_pickup_day	total_day_revenue
0	Friday	55818.74
1	Monday	49574.37
2	Saturday	51195.40
3	Sunday	48624.06
4	Thursday	57181.91
5	Tuesday	52527.14
6	Wednesday	55310.47

[228]:
```
# Create bar plot of total revenue by day
plt.figure(figsize = (15, 10))
sns.barplot(data= revenue_by_day, x= "tpep_pickup_day", y= "total_day_revenue",␣
    ↪palette= "Set2", order= days_order)

plt.title("Total Revenue per Day")
```

```
plt.xlabel("Days")
plt.ylabel("Total Revenue")
plt.show()
```



Plot total revenue by month

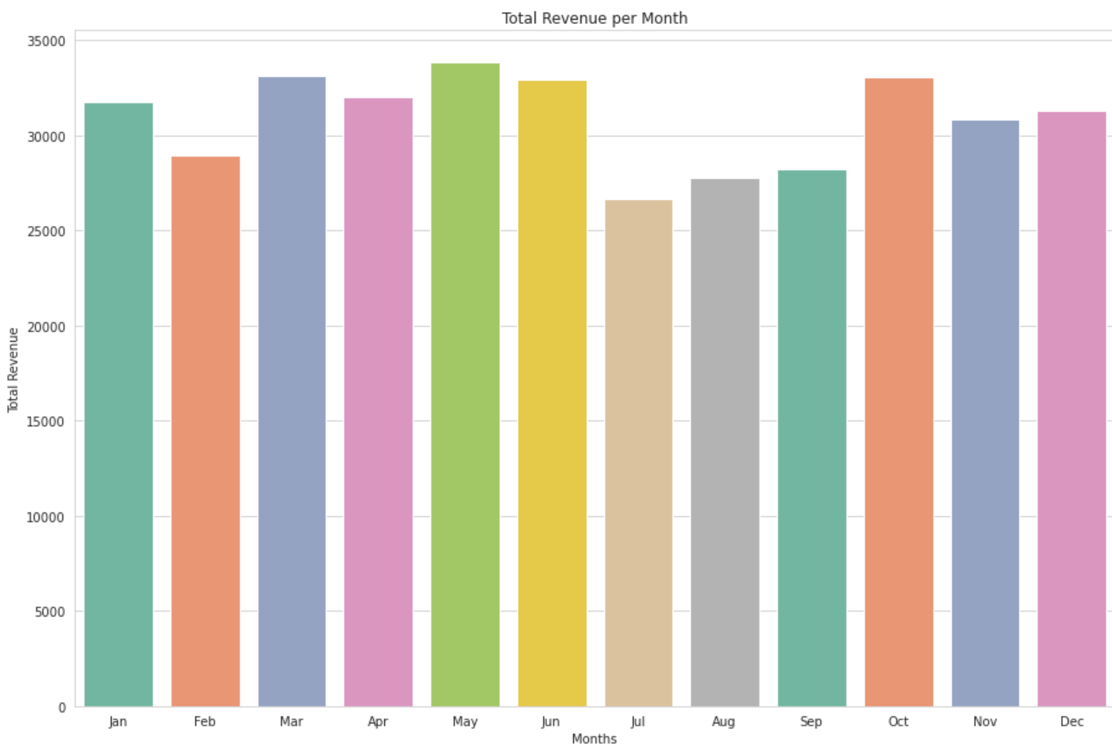
[229]:
```
# Repeat the process, this time for total revenue by month
revenue_by_month = df.groupby(["tpep_pickup_month"])[["total_amount"]].sum().
↪reset_index().rename(columns= {"total_amount": "total_month_revenue"})
revenue_by_month
```

[229]:

	tpep_pickup_month	total_month_revenue
0	Apr	32012.54
1	Aug	27759.56
2	Dec	31261.57
3	Feb	28937.89
4	Jan	31735.25
5	Jul	26617.64
6	Jun	32920.52
7	Mar	33085.89
8	May	33828.58
9	Nov	30800.44
10	Oct	33065.83

```
11                Sep             28206.38
```

[230]:
```python
# Create a bar plot of total revenue by month
plt.figure(figsize = (15, 10))
sns.barplot(data= revenue_by_month, x= "tpep_pickup_month", y=␣
↪"total_month_revenue", palette= "Set2", order= months_order)

plt.title("Total Revenue per Month")
plt.xlabel("Months")
plt.ylabel("Total Revenue")
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

[231]:
```
# Get number of unique drop-off location IDs
df["DOLocationID"].value_counts()
```

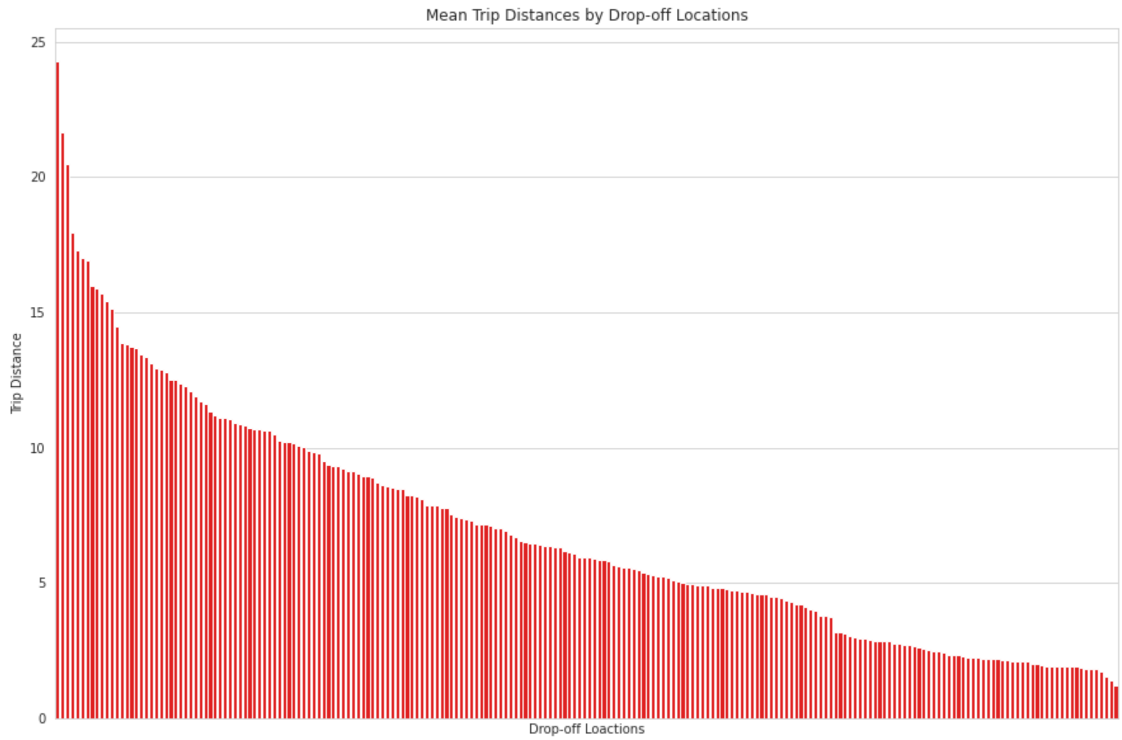
[231]:
```
161    858
236    802
230    761
237    759
170    699
      …
219      1
18       1
31       1
147      1
201      1
Name: DOLocationID, Length: 216, dtype: int64
```

[232]:
```
# Calculate the mean trip distance for each drop-off location
mean_distance_by_DOLocation = df.groupby(["DOLocationID"])[["trip_distance"]].
 ↪mean()

# Sort the results in descending order by mean trip distance
mean_distance_by_DOLocation = mean_distance_by_DOLocation.sort_values(by=␣
 ↪"trip_distance", ascending= False).reset_index()
mean_distance_by_DOLocation
```

[232]:
```
     DOLocationID  trip_distance
0              23      24.275000
1              29      21.650000
2             210      20.500000
3              11      17.945000
4              51      17.310000
..            …            …
211           137       1.818852
212           234       1.727806
213           237       1.555494
214           193       1.390556
215           207       1.200000

[216 rows x 2 columns]
```

[233]:
```
# Create a bar plot of mean trip distances by drop-off location in descending␣
 ↪order by distance
plt.figure(figsize = (15, 10))
ax = sns.barplot(data= mean_distance_by_DOLocation, x= "DOLocationID", y=␣
 ↪"trip_distance", color="red",
                 order= mean_distance_by_DOLocation["DOLocationID"])
```

```
plt.title("Mean Trip Distances by Drop-off Locations")
plt.xlabel("Drop-off Loactions")
ax.set_xticklabels([])
ax.set_xticks([])
plt.ylabel("Trip Distance")
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

[237]:
```
#BONUS CONTENT

# 1. Generate random points on a 2D plane from a normal distribution
test = np.round(np.random.normal(10, 5, (3000, 2)), 1)
midway = int(len(test)/2)
start = test[:midway]
end = test[midway:]
```

```python
# 2. Calculate Euclidean distances between points in first half and second half␣
↪of array
distances = (start - end)**2
distances = distances.sum(axis=-1)
distances = np.sqrt(distances)

# 3. Group the coordinates by "drop-off location", compute mean distance
test_df = pd.DataFrame({'start': [tuple(x) for x in start.tolist()],
                   'end': [tuple(x) for x in end.tolist()],
                   'distance': distances})
data = test_df[['end', 'distance']].groupby(['end']).mean()
data = data.sort_values(by='distance')

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

plt.figure(figsize=(15, 10))
ax = sns.barplot(x= data.index, y= data['distance'], order= data.index)

ax.set_xticks([])
ax.set_xlabel('Endpoint')
ax.set_ylabel('Mean distance to all other points')
ax.set_title('Mean distance between points taken randomly from normal␣
↪distribution')
```

[237]:
```
Text(0.5, 1.0, 'Mean distance between points taken randomly from normal
distribution')
```

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.

[235]:
```python
# Check if all drop-off locations are consecutively numbered
print("Count of Unique Drop-off Locations: ", len(df["DOLocationID"].unique()))
print("The Maximum Drop-off Location: ", df["DOLocationID"].max())
```

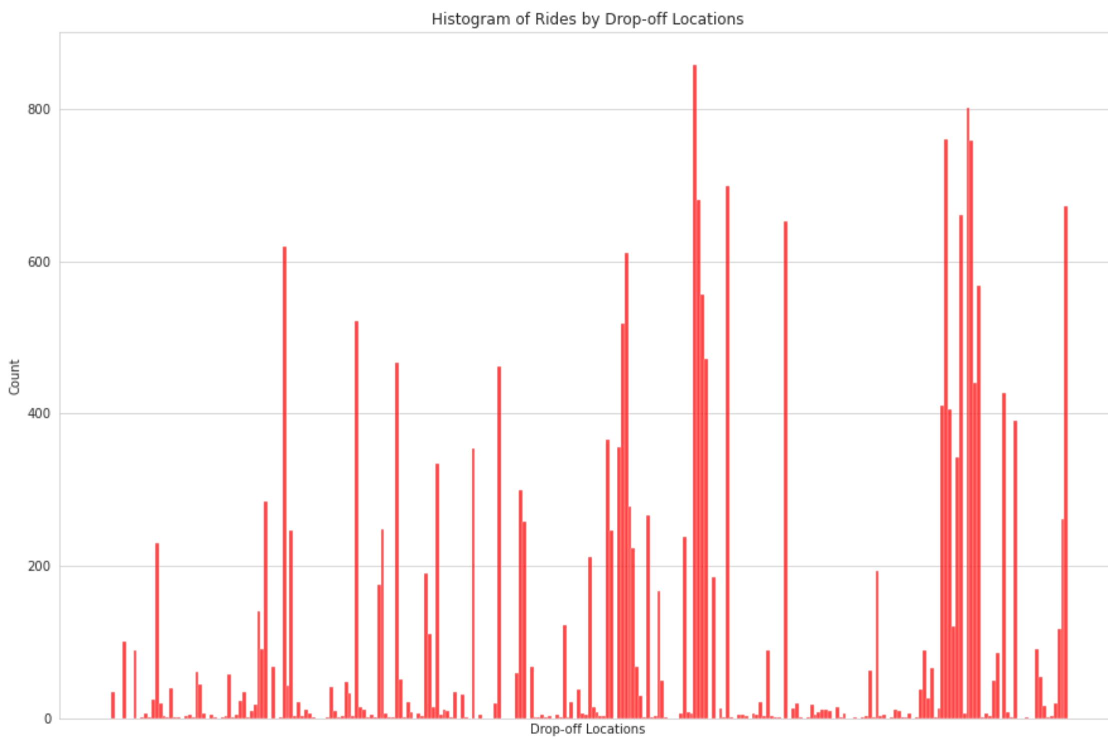
```
Count of Unique Drop-off Locations:  216
The Maximum Drop-off Location:  265
```

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.

[236]:
```python
# DOLocationID column is numeric, so sort in ascending order
sorted_dropoffs = df["DOLocationID"].sort_values()

# Convert to string
sorted_dropoffs = sorted_dropoffs.astype("object")

# Plot
```

```python
plt.figure(figsize=(15, 10))
ax = sns.histplot(sorted_dropoffs, bins= range(0, df["DOLocationID"].max(), 1),␣
 ↪color= "red")

ax.set_xticklabels([])
ax.set_xticks([])
ax.set_xlabel("Drop-off Locations")
ax.set_title("Histogram of Rides by Drop-off Locations")
```

[236]: Text(0.5, 1.0, 'Histogram of Rides by Drop-off Locations')



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?

I have learned how to craft clear visualizations.

My other questions are how the negative values entered the dataset.

My client would likely want to know more relationships between the dataset variables.

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 provides a clear understanding for the dataset.

Visualizations helped me understand the distribution of the variables and the relationships hidden between the variables.

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

Congratulations! You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged.