NYC TLC Project Part 2

November 5, 2024

1 NYC TLC Project

Data structuring and cleaning, as well as matplotlib/seaborn visualizations plotted to help understand the data.

2 Exploratory data analysis

The purpose of this project is to conduct exploratory data analysis on the data set.

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

3 Visualize a story in Tableau and Python

[1]:
```
# Import packages and libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# Load dataset into dataframe
df = pd.read_csv('2017_Yellow_Taxi_Trip_Data.csv')
```

3.0.1 Task 2a. Data exploration and cleaning

Start by discovering, using head and size.

[3]:
```
df.head()
```

[3]:

	Unnamed: 0	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount
0	24870114	2	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	6	3.34	1	N	100	231	1	13.0	0.0	0.5	2.76	0.0	0.3	16.56
1	35634249	1	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	1	1.80	1	N	186	43	1	16.0	0.0	0.5	4.00	0.0	0.3	20.80
2	106203690	1	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	1	1.00	1	N	262	236	1	6.5	0.0	0.5	1.45	0.0	0.3	8.75
3	38942136	2	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	1	3.70	1	N	188	97	1	20.5	0.0	0.5	6.39	0.0	0.3	27.69
4	30841670	2	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	1	4.37	1	N	4	112	2	16.5	0.5	0.5	0.00	0.0	0.3	17.80

[4]:
```
df.size
```

[4]:
```
408582
```

Use describe...

[5]:
```
df.describe()
```

[5]:

	Unnamed: 0	VendorID	passenger_count	trip_distance	RatecodeID	PULocationID	DOLocationID	payment_type	fare_amount
count	2.269900e+04	22699.000000	22699.000000	22699.000000	22699.000000	22699.000000	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					

```
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.

[6]:
```
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
```

3.0.2 Task 2b. Select visualization type(s)

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

3.0.3 Task 3. Data visualization

3.0.4 Boxplots

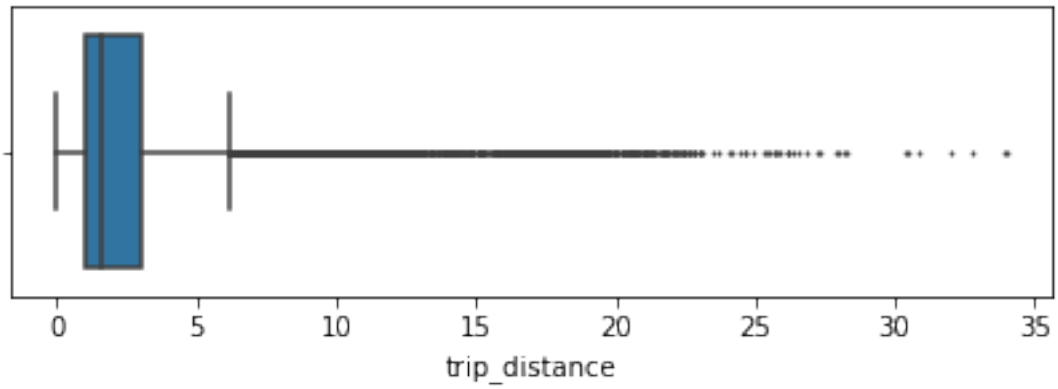
Perform a check for outliers on relevant columns such as trip distance and trip duration.

[20]:
```
# 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

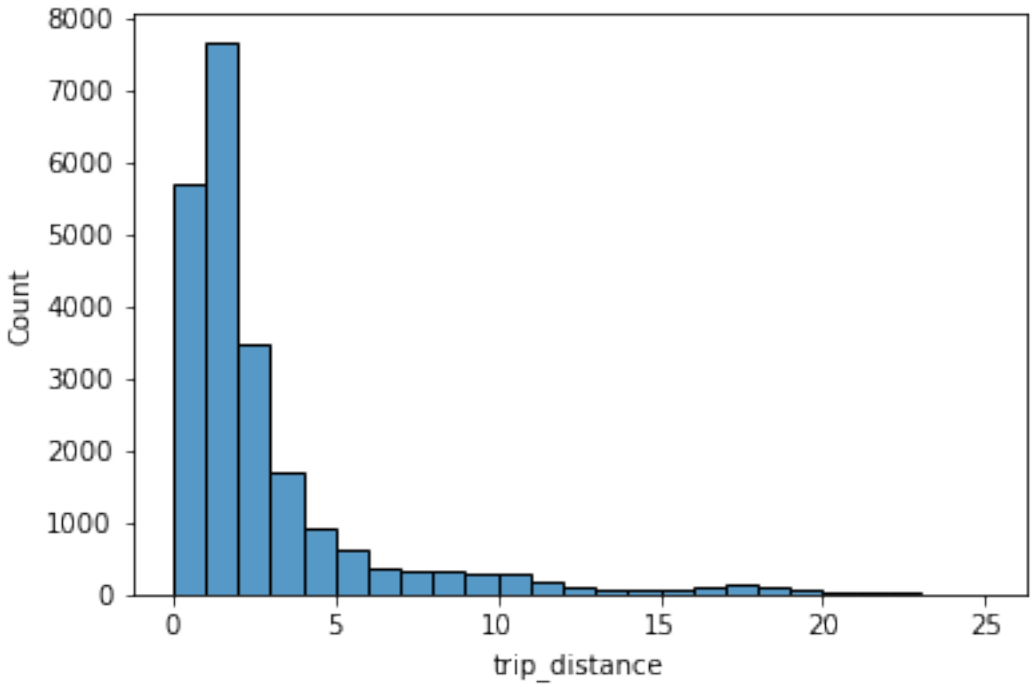
[8]:
```
# Create box plot of trip_distance
import seaborn as sns
plt.figure(figsize=(7,2))
sns.boxplot(data=None, x=df['trip_distance'], fliersize=1)
```

[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee4d608690>
```



[9]:
```
# Create histogram of trip_distance
sns.histplot(x=df['trip_distance'], bins=range(0,26,1))
```

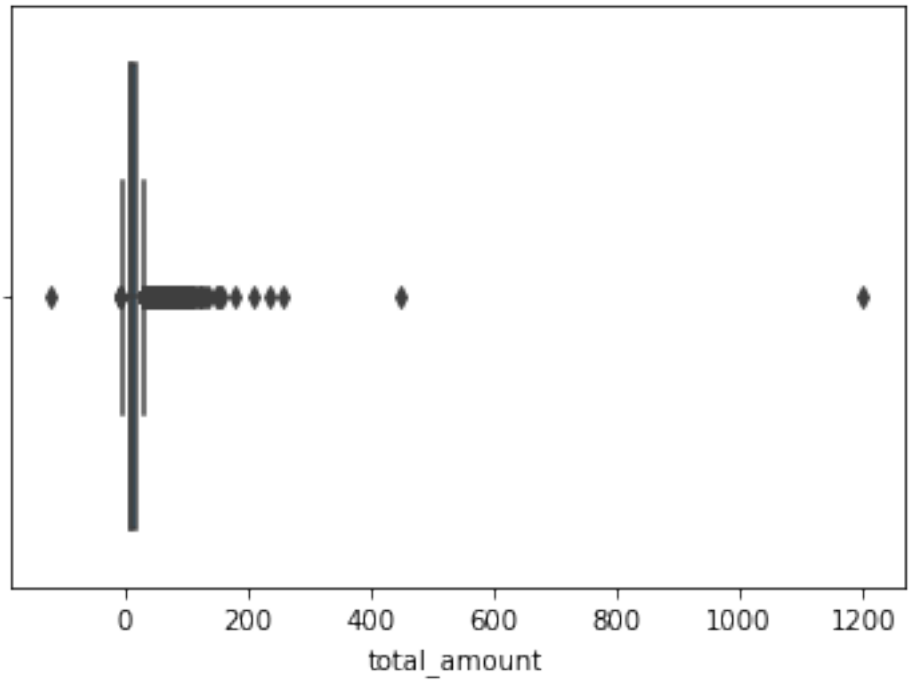
[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7fee4d383250>



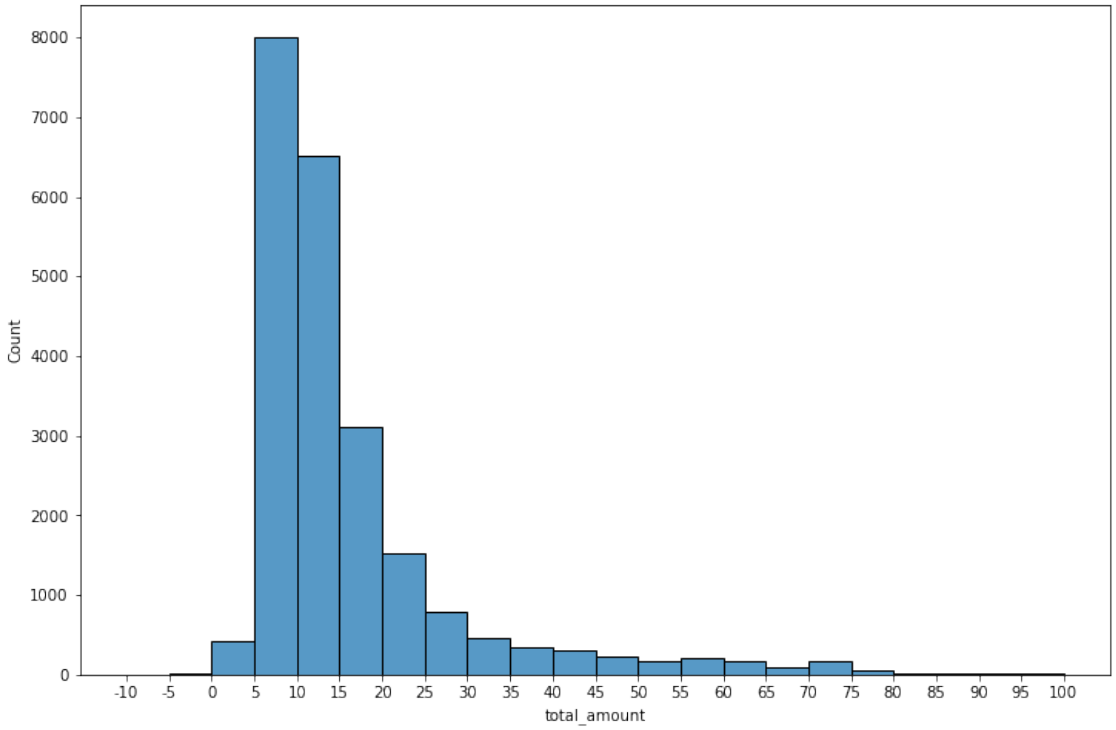
total amount

[10]:
```
# Create box plot of total_amount
sns.boxplot(data=None, x=df['total_amount'])
```

[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7fee4d09f1d0>



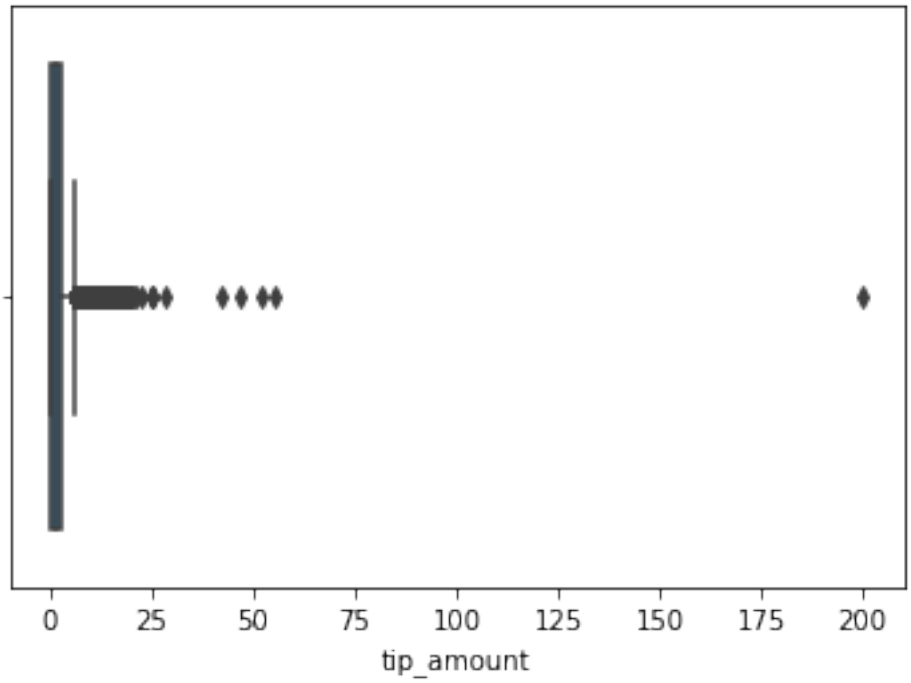
[25]:
```
# Create histogram of total_amount
plt.figure(figsize=(12,8))
ax = sns.histplot(x=df['total_amount'], bins=range(-10,101,5))
ax.set_xticks(range(-10,101,5))
ax.set_xticklabels(range(-10,101,5));
```



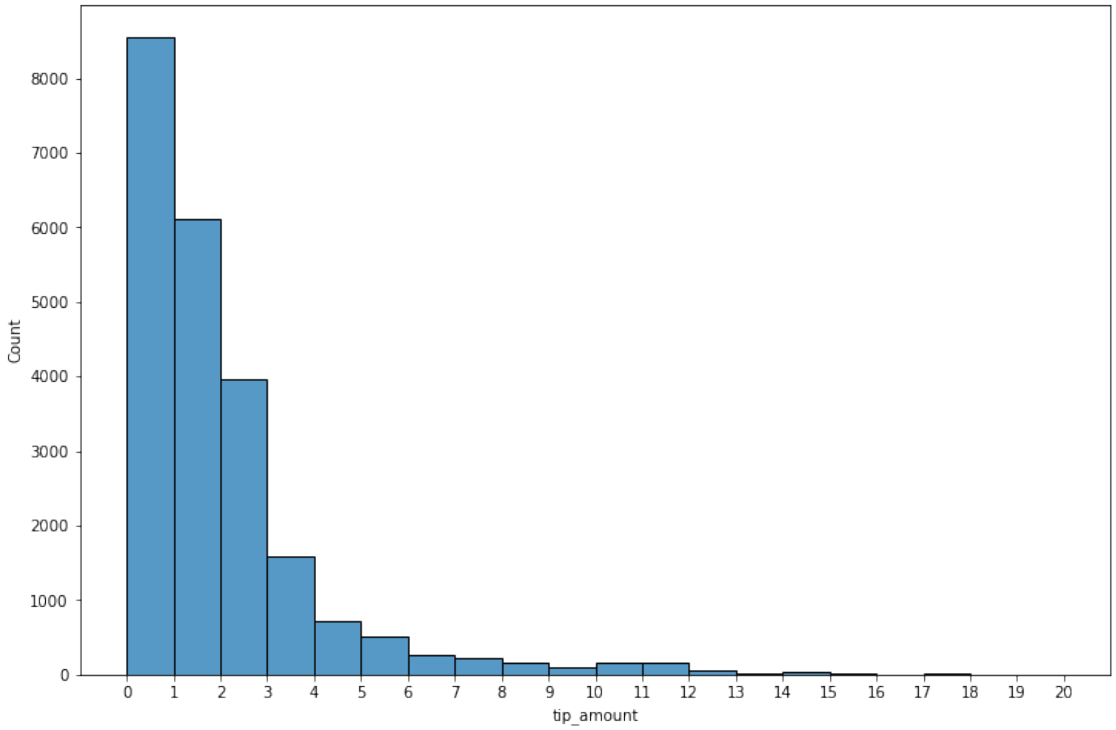
tip amount

[11]:
```
# Create box plot of tip_amount
sns.boxplot(df['tip_amount'])
```

[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7fee4d085190>

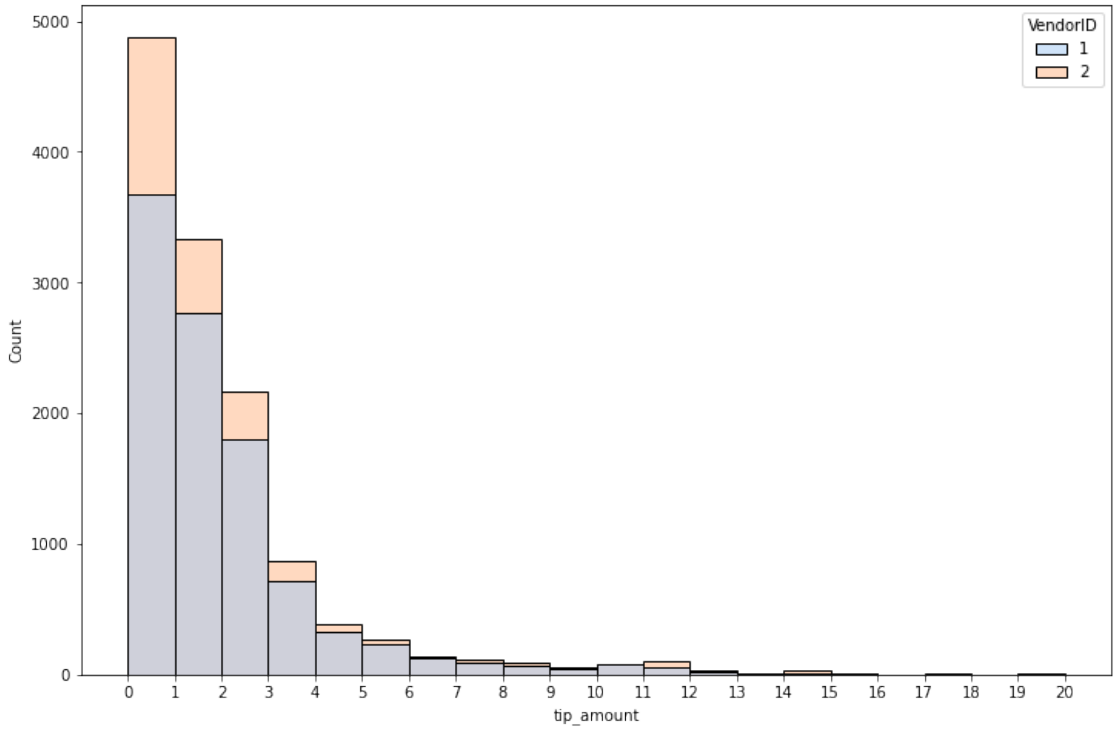


[12]:
```
# Create histogram of tip_amount
plt.figure(figsize=(12,8))
ax = sns.histplot(x=df['tip_amount'], bins=range(0,21,1))
ax.set_xticks(range(0,21,1))
ax.set_xticklabels(range(0,21,1));
```

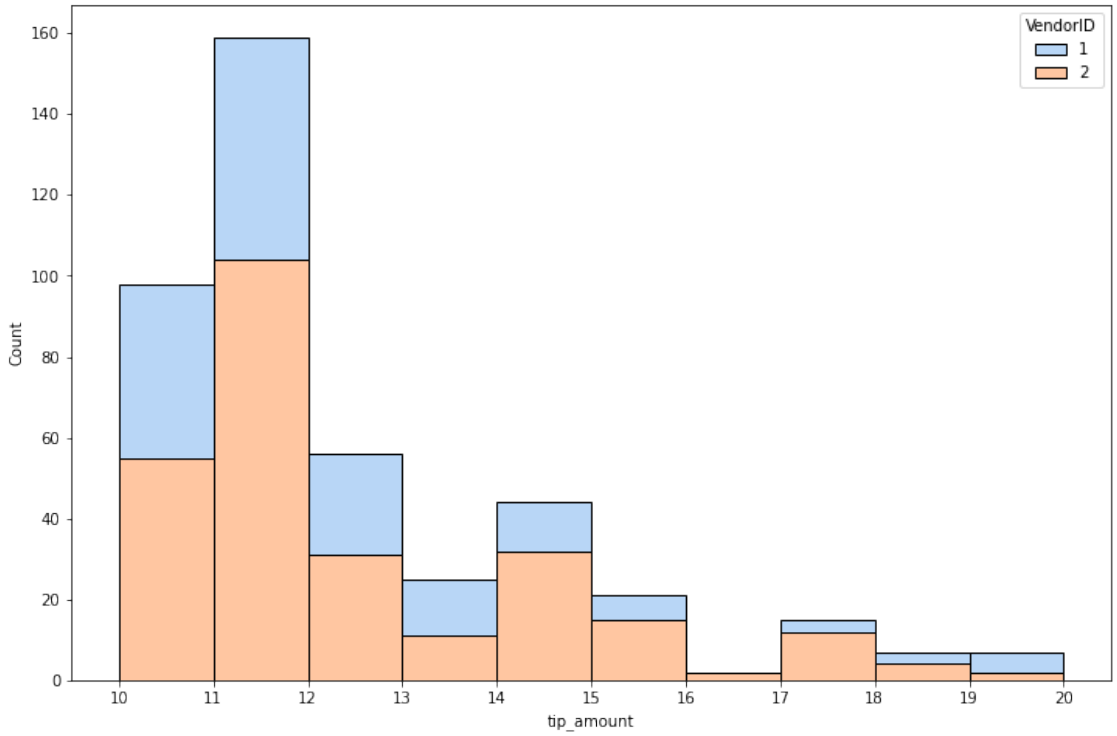


tip_amount by vendor

[13]:
```
# Create histogram of tip_amount by vendor
plt.figure(figsize=(12,8))
ax = sns.histplot(data=df, x=df['tip_amount'], bins=range(0,21,1),
                  hue='VendorID',
                  palette = 'pastel')
ax.set_xticks(range(0,21,1))
ax.set_xticklabels(range(0,21,1));
```



[14]:
```
# Create histogram of tip_amount by vendor for tips > $10
data = df[df['tip_amount']>10]
plt.figure(figsize=(12,8))
ax = sns.histplot(data=data, x='tip_amount', bins=range(10,21,1),
                  hue='VendorID',
                   multiple='stack',
                  palette = 'pastel')
ax.set_xticks(range(10,21,1))
ax.set_xticklabels(range(10,21,1));
```



Mean tips by passenger count

Examine the unique values in the `passenger_count` column.

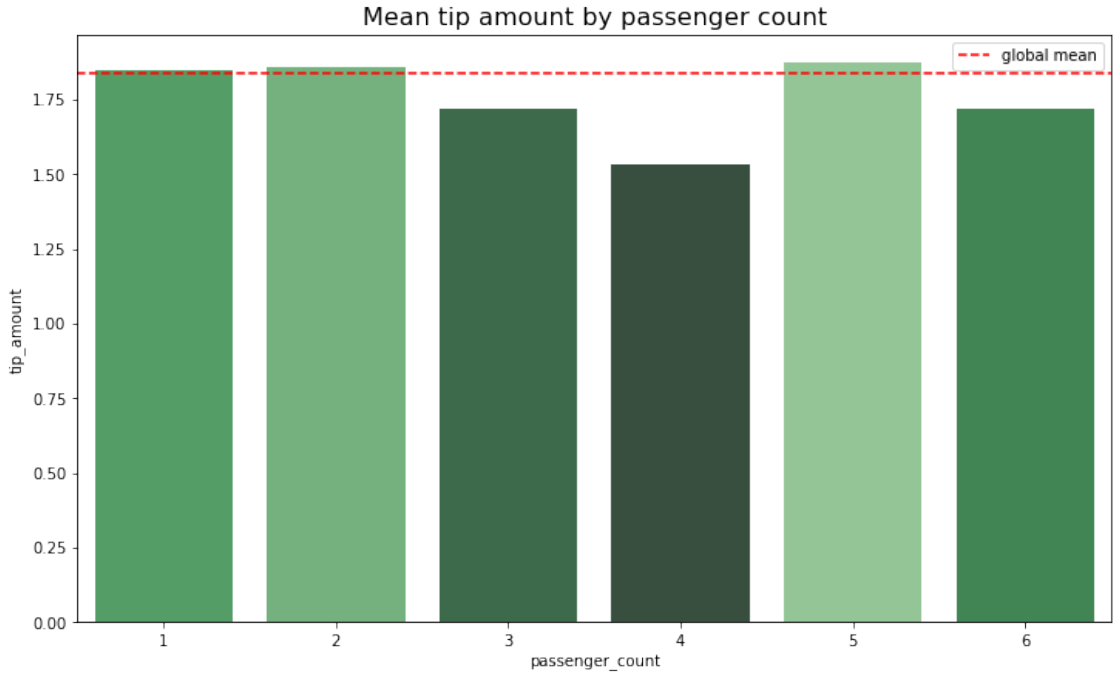
[15]:
```
df['passenger_count'].value_counts()
```

[15]:
```
1    16117
2     3305
5     1143
3      953
6      693
4      455
0       33
Name: passenger_count, dtype: int64
```

[16]:
```
# Calculate mean tips by passenger_count
mean_tips_by_passenger_count = df.groupby('passenger_count').
 ↪mean()[['tip_amount']]
```

[17]:
```
# Create bar plot for mean tips by passenger count
data = mean_tips_by_passenger_count.tail(-1)
pal = sns.color_palette("Greens_d", len(data))
rank = data['tip_amount'].argsort().argsort()
plt.figure(figsize=(12,7))
```

```
ax = sns.barplot(x=data.index,
            y=data['tip_amount'],
            palette=np.array(pal[::-1])[rank])
ax.axhline(df['tip_amount'].mean(), ls='--', color='red', label='global mean')
ax.legend()
plt.title('Mean tip amount by passenger count', fontsize=16);
```



Create month and day columns

[24]:
```
# Create a month column
df['month'] = df['tpep_pickup_datetime'].dt.month_name()
# Create a day column
#==> ENTER YOUR CODE HERE
df['day'] = df['tpep_pickup_datetime'].dt.day_name()
```

Plot total ride count by month

[30]:
```
# Get total number of rides for each month
monthly_rides = df['month'].value_counts()
```

Reorder the results to put the months in calendar order.

[31]:
```
# Reorder the monthly ride list so months go in order
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
         'August', 'September', 'October', 'November', 'December']
```

```
monthly_rides = monthly_rides.reindex(index=month_order)
monthly_rides
```

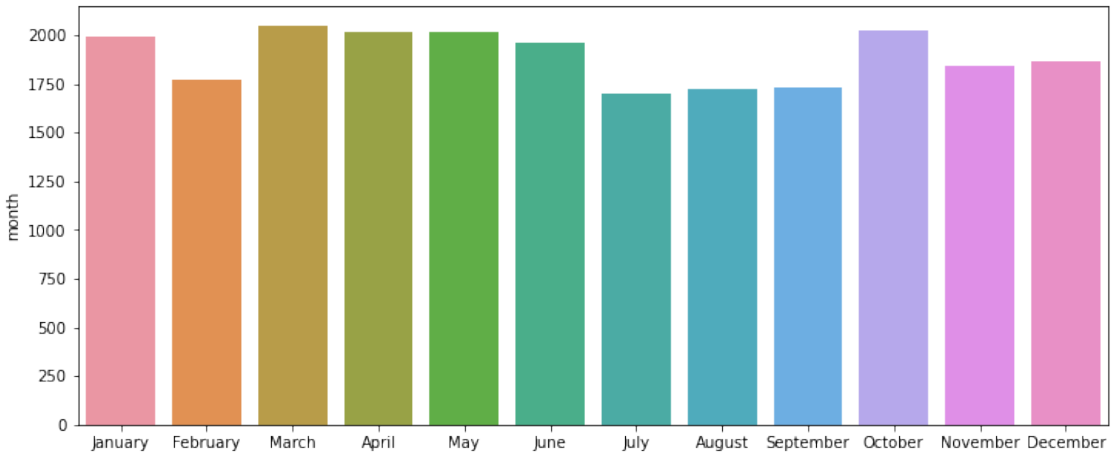
[31]:
```
January      1997
February     1769
March        2049
April        2019
May          2013
June         1964
July         1697
August       1724
September    1734
October      2027
November     1843
December     1863
Name: month, dtype: int64
```

[32]:
```
# Show the index
monthly_rides.index
```

[32]:
```
Index(['January', 'February', 'March', 'April', 'May', 'June', 'July',
       'August', 'September', 'October', 'November', 'December'],
      dtype='object')
```

[35]:
```
# Create a bar plot of total rides per month
plt.figure(figsize=(12,5))
sns.barplot(x=monthly_rides.index, y=monthly_rides)
```

[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee4c8be950>
```



Plot total ride count by day

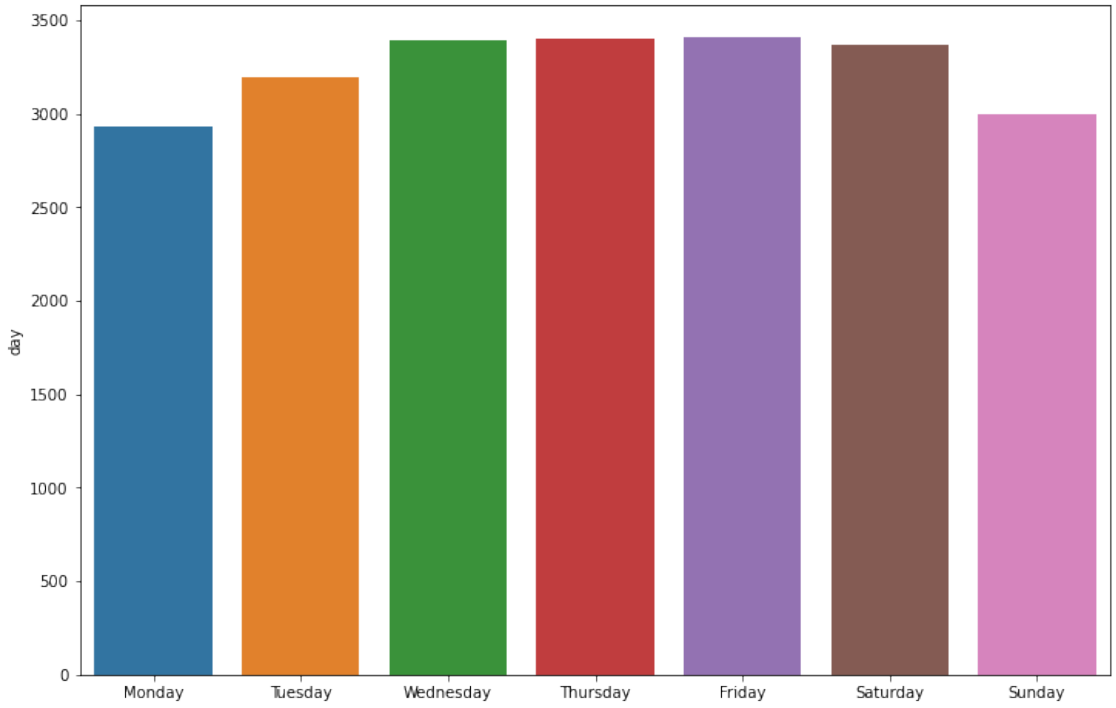
Repeat the above process, but now calculate the total rides by day of the week.

[36]:
```
# Repeat the above process, this time for rides by day
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
```

[36]:
```
Monday       2931
Tuesday      3198
Wednesday    3390
Thursday     3402
Friday       3413
Saturday     3367
Sunday       2998
Name: day, dtype: int64
```

[37]:
```
# Create bar plot for ride count by day
plt.figure(figsize=(12,8))
sns.barplot(x=daily_rides.index,y=daily_rides)
```

[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee4c6316d0>
```



Plot total revenue by day of the week

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

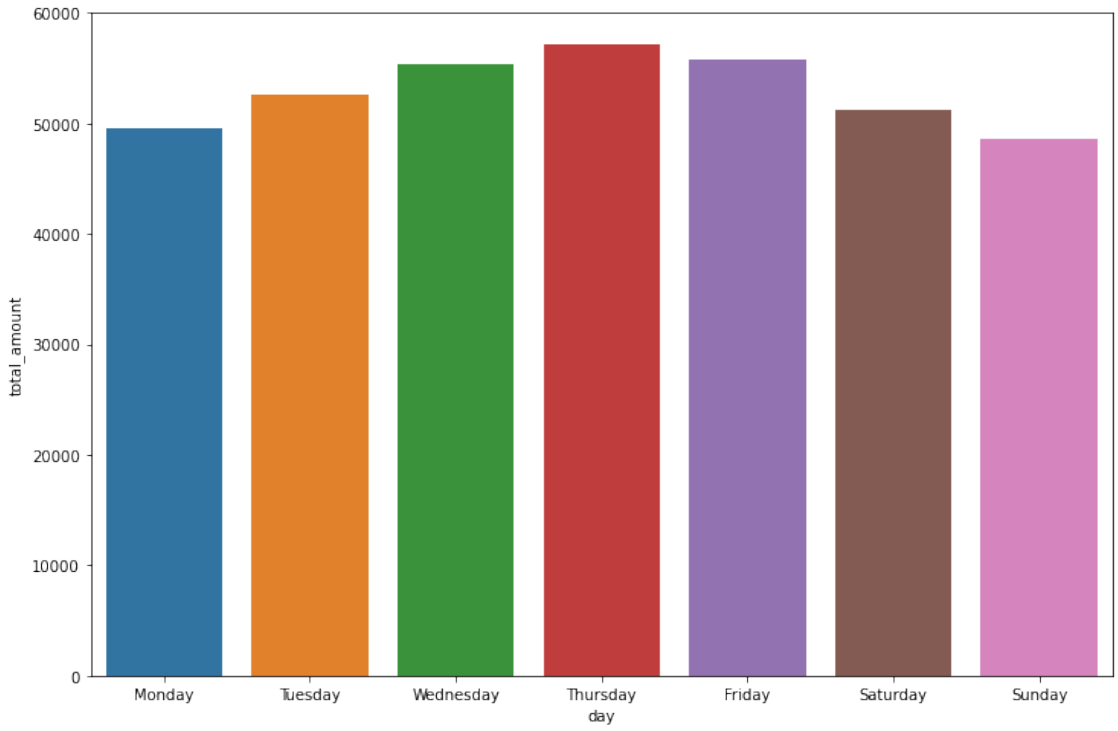
[38]:
```
# Repeat the process, this time for total revenue by day
revenue_by_day = df.groupby('day').sum()[['total_amount']]
revenue_by_day = revenue_by_day.reindex(index=day_order)
revenue_by_day
```

[38]:

day	total_amount
Monday	49574.37
Tuesday	52527.14
Wednesday	55310.47
Thursday	57181.91
Friday	55818.74
Saturday	51195.40
Sunday	48624.06

[41]:
```
# Create bar plot of total revenue by day
plt.figure(figsize=(12,8))
sns.barplot(x=revenue_by_day.index,y=revenue_by_day.total_amount)
```

[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee4c390750>
```



Plot total revenue by month

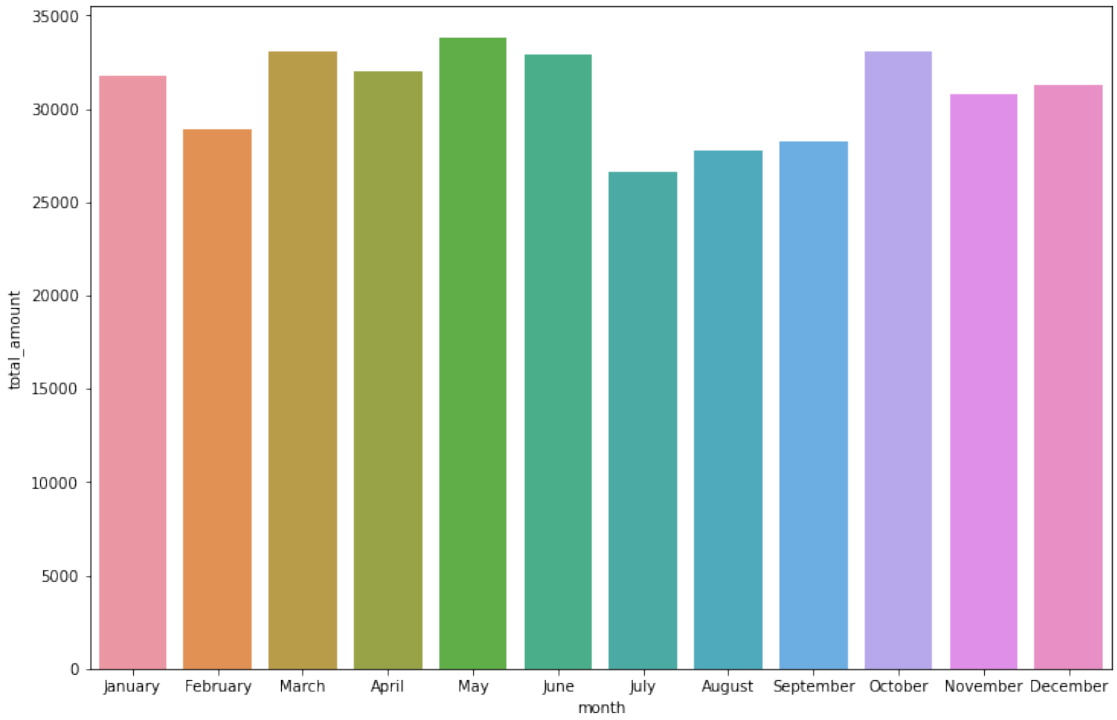
[42]:
```
# Repeat the process, this time for total revenue by month
revenue_by_month = df.groupby('month').sum()[['total_amount']]
revenue_by_month = revenue_by_month.reindex(index=month_order)
revenue_by_month
```

[42]:

month	total_amount
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

[43]:
```
# Create a bar plot of total revenue by month
plt.figure(figsize=(12,8))
sns.barplot(x=revenue_by_month.index,y=revenue_by_month.total_amount)
```

[43]: <matplotlib.axes._subplots.AxesSubplot at 0x7fee4c390390>



Plot mean trip distance by drop-off location

[46]:
```
# Get number of unique drop-off location IDs
df['DOLocationID'].nunique()
```

[46]:
```
216
```

[48]:
```
# Calculate the mean trip distance for each drop-off location
distance_by_dropoff = df.groupby('DOLocationID').mean()[['trip_distance']]

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

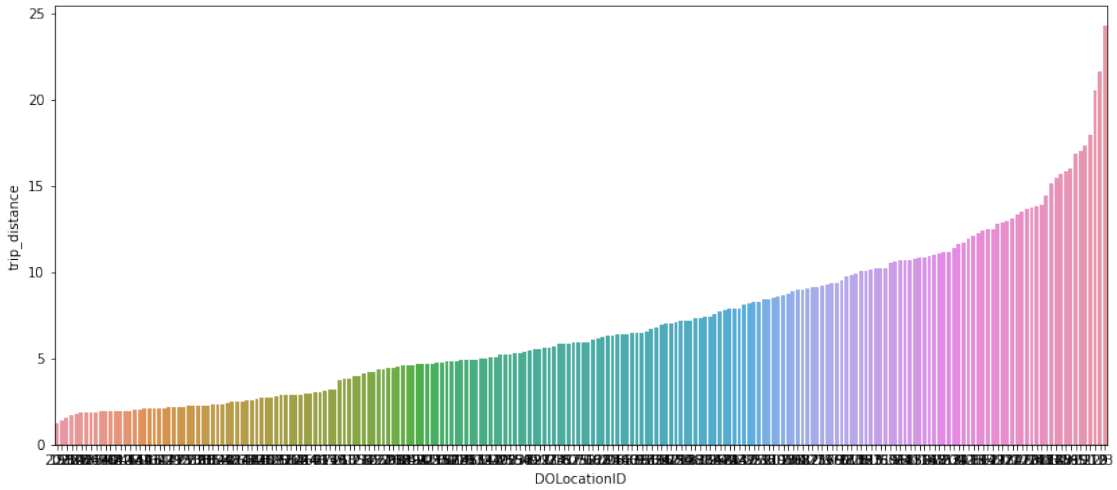
[48]:

DOLocationID	trip_distance
207	1.200000
193	1.390556
237	1.555494
234	1.727806
137	1.818852
…	…
51	17.310000
11	17.945000
210	20.500000
29	21.650000
23	24.275000

[216 rows x 1 columns]

[49]:
```
# Create a bar plot of mean trip distances by drop-off location in ascending␣
 ↪order by distance

plt.figure(figsize=(14,6))
ax = sns.barplot(x=distance_by_dropoff.index,
                 y=distance_by_dropoff['trip_distance'],
                 order=distance_by_dropoff.index)
```



This plot presents a characteristic curve related to the cumulative density function of a normal distribution. In other words, it indicates that the drop-off points are relatively evenly distributed over the terrain. This is good to know, because geographic coordinates were not included in this dataset, so there was no obvious way to test for the distibution of locations.