

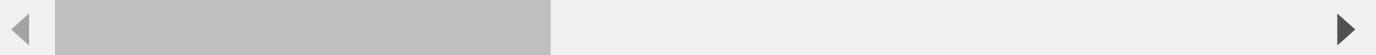
```
In [1]: import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
import datetime as dt  
import seaborn as sns
```

```
In [3]: df=pd.read_csv('2017_Yellow_Taxi_Trip_Data.csv')
```

```
In [4]: df.head(10)
```

```
Out[4]:
```

	Unnamed: 0	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance
0	24870114	2	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	6	3.34
1	35634249	1	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	1	1.80
2	106203690	1	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	1	1.00
3	38942136	2	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	1	3.70
4	30841670	2	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	1	4.37
5	23345809	2	03/25/2017 8:34:11 PM	03/25/2017 8:42:11 PM	6	2.30
6	37660487	2	05/03/2017 7:04:09 PM	05/03/2017 8:03:47 PM	1	12.83
7	69059411	2	08/15/2017 5:41:06 PM	08/15/2017 6:03:05 PM	1	2.98
8	8433159	2	02/04/2017 4:17:07 PM	02/04/2017 4:29:14 PM	1	1.20
9	95294817	1	11/10/2017 3:20:29 PM	11/10/2017 3:40:55 PM	1	1.60



```
In [5]: df.size
```

```
Out[5]: 408582
```

```
In [6]: df.describe()
```

Out[6]:

	Unnamed: 0	VendorID	passenger_count	trip_distance	RatecodeID	PULocationID	DOLc
count	2.269900e+04	22699.000000	22699.000000	22699.000000	22699.000000	22699.000000	2269
mean	5.675849e+07	1.556236	1.642319	2.913313	1.043394	162.412353	16
std	3.274493e+07	0.496838	1.285231	3.653171	0.708391	66.633373	7
min	1.212700e+04	1.000000	0.000000	0.000000	1.000000	1.000000	
25%	2.852056e+07	1.000000	1.000000	0.990000	1.000000	114.000000	11
50%	5.673150e+07	2.000000	1.000000	1.610000	1.000000	162.000000	16
75%	8.537452e+07	2.000000	2.000000	3.060000	1.000000	233.000000	23
max	1.134863e+08	2.000000	6.000000	33.960000	99.000000	265.000000	26



In [7]:

```
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
```

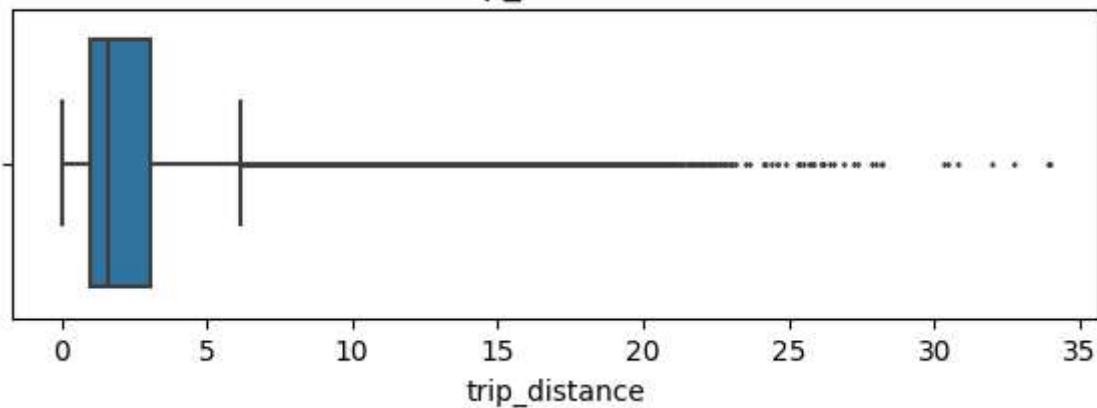
In [8]:

```
# 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'])
```

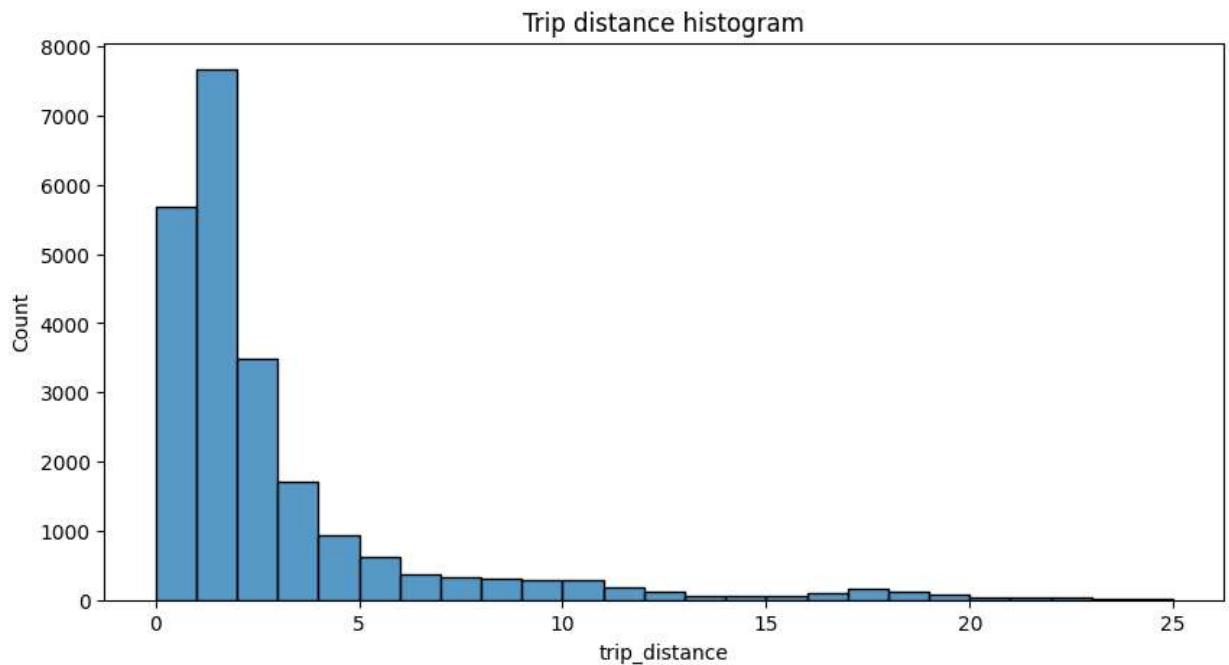
In [9]:

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

trip_distance

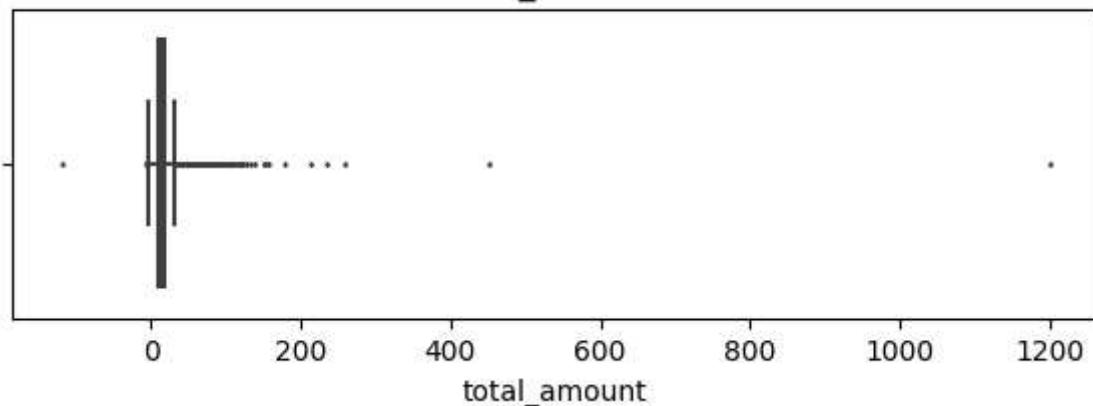


```
In [10]: # Create histogram of trip_distance
plt.figure(figsize=(10,5))
sns.histplot(df['trip_distance'], bins=range(0,26,1))
plt.title('Trip distance histogram');
```



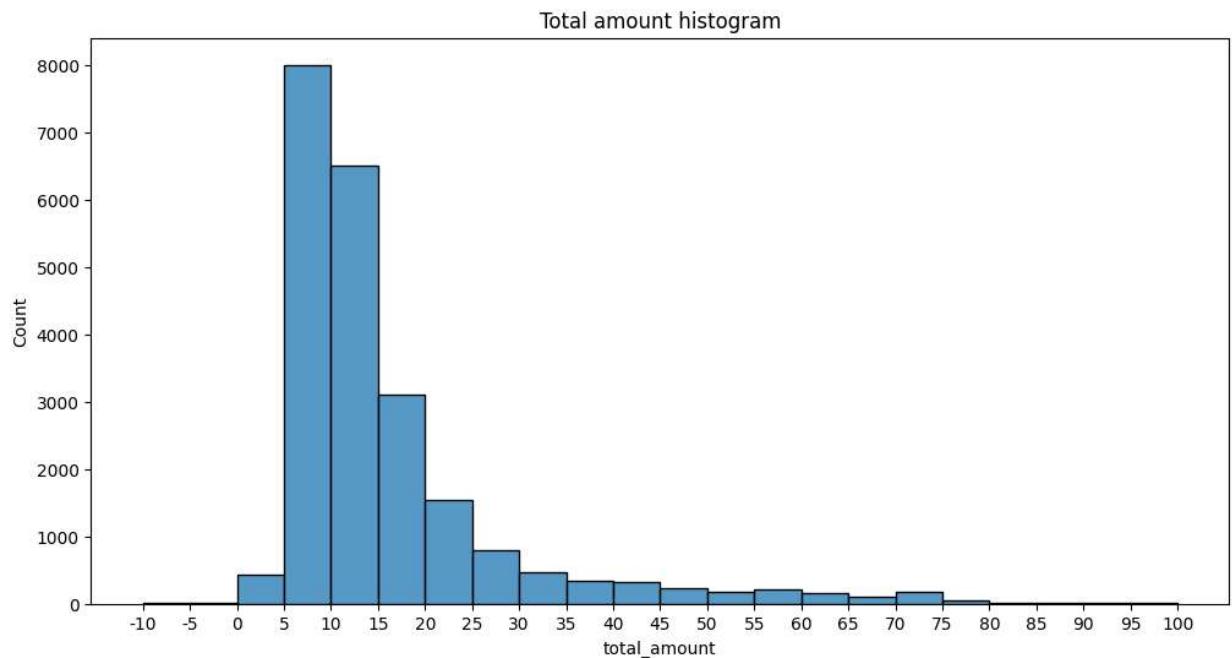
```
In [11]: # Create box plot of total_amount
plt.figure(figsize=(7,2))
plt.title('total_amount')
sns.boxplot(x=df['total_amount'], fliersize=1);
```

total_amount



```
In [12]: # Create histogram of total_amount
```

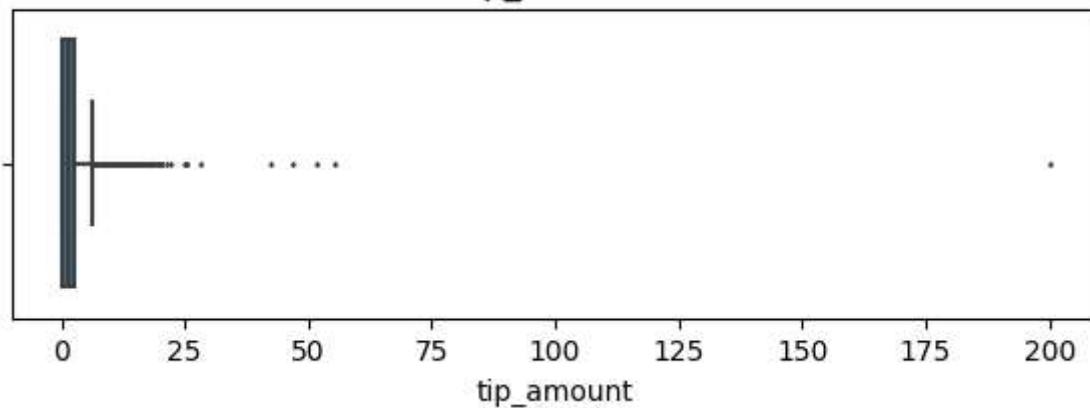
```
plt.figure(figsize=(12,6))
ax = sns.histplot(df['total_amount'], bins=range(-10,101,5))
ax.set_xticks(range(-10,101,5))
ax.set_xticklabels(range(-10,101,5))
plt.title('Total amount histogram');
```



```
In [13]: # Create box plot of tip_amount
```

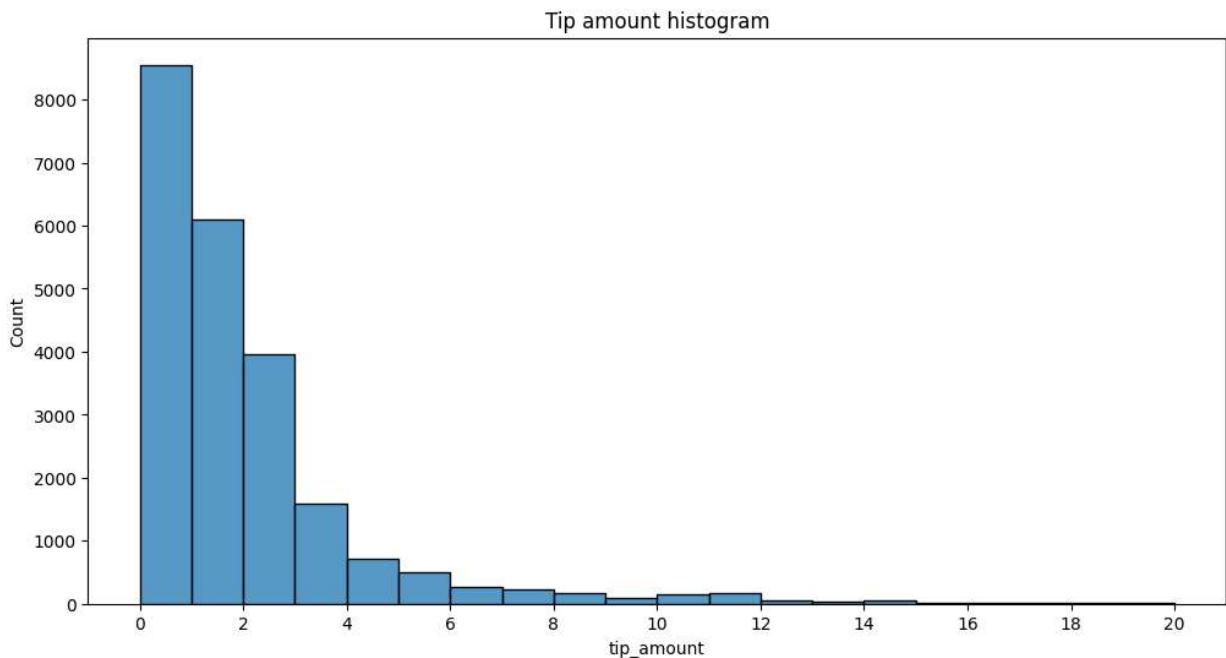
```
plt.figure(figsize=(7,2))
plt.title('tip_amount')
sns.boxplot(x=df['tip_amount'], fliersize=1);
```

tip_amount



In [14]:

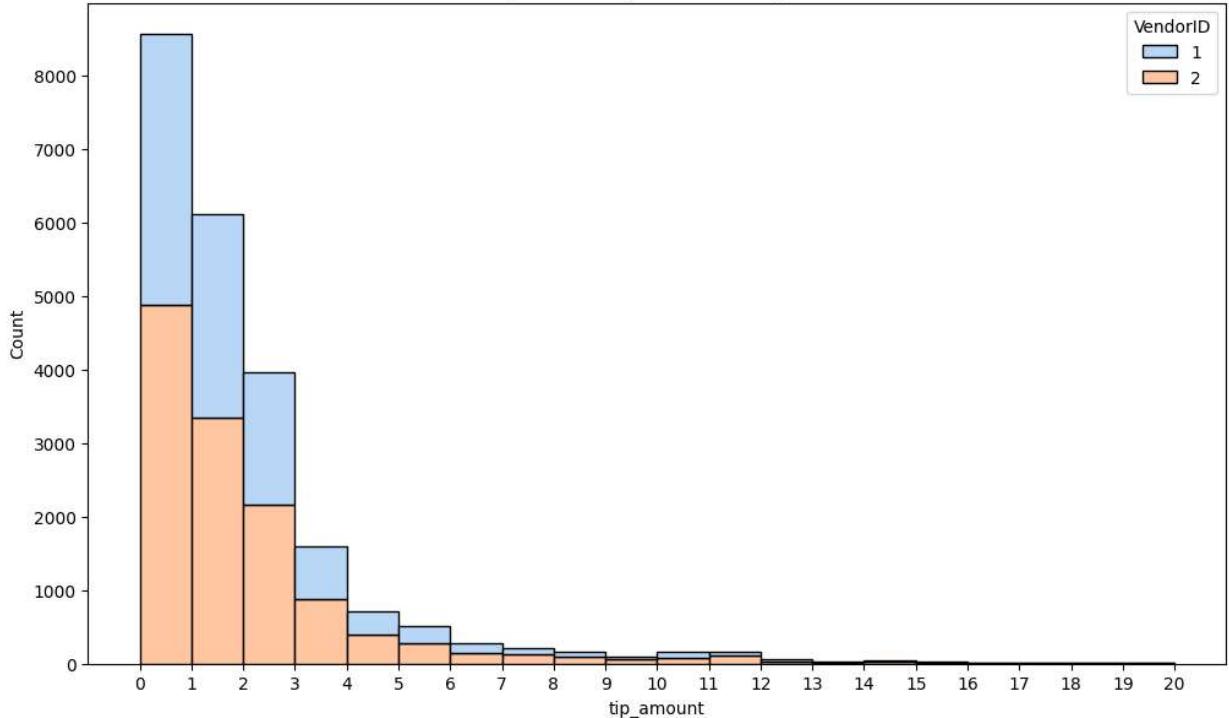
```
# Create histogram of tip_amount
plt.figure(figsize=(12,6))
ax = sns.histplot(df['tip_amount'], bins=range(0,21,1))
ax.set_xticks(range(0,21,2))
ax.set_xticklabels(range(0,21,2))
plt.title('Tip amount histogram');
```



In [15]:

```
# 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='pastel')
ax.set_xticks(range(0,21,1))
ax.set_xticklabels(range(0,21,1))
plt.title('Tip amount by vendor histogram');
```

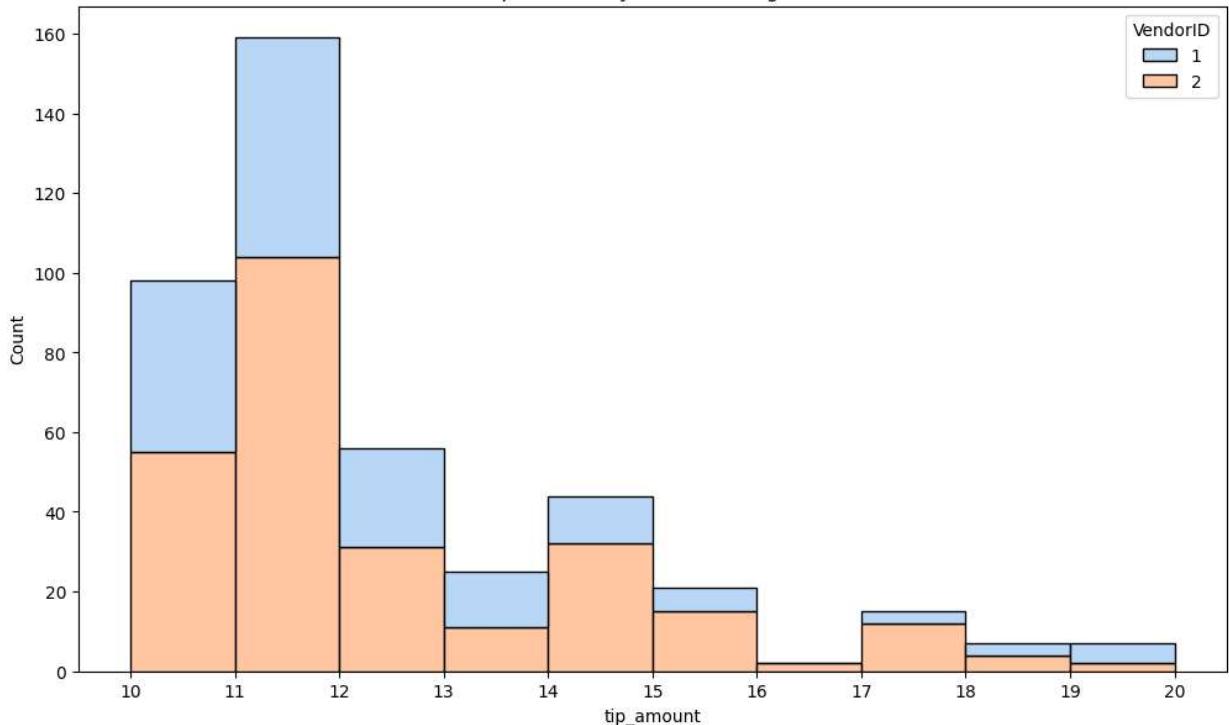
Tip amount by vendor histogram



```
In [16]: # Create histogram of tip_amount by vendor for tips > $10
```

```
tips_over_ten = df[df['tip_amount'] > 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='pastel')
ax.set_xticks(range(10,21,1))
ax.set_xticklabels(range(10,21,1))
plt.title('Tip amount by vendor histogram');
```

Tip amount by vendor histogram



```
In [17]: df['passenger_count'].value_counts()
```

```
Out[17]: 1    16117
         2    3305
         5    1143
         3    953
         6    693
         4    455
         0    33
Name: passenger_count, dtype: int64
```

```
In [18]: # Calculate mean tips by passenger_count
```

```
mean_tips_by_passenger_count = df.groupby(['passenger_count']).mean()[['tip_amount']]
mean_tips_by_passenger_count
```

C:\Users\bussy\AppData\Local\Temp\ipykernel_67848\1943984880.py:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.

```
mean_tips_by_passenger_count = df.groupby(['passenger_count']).mean()[['tip_amount']]
```

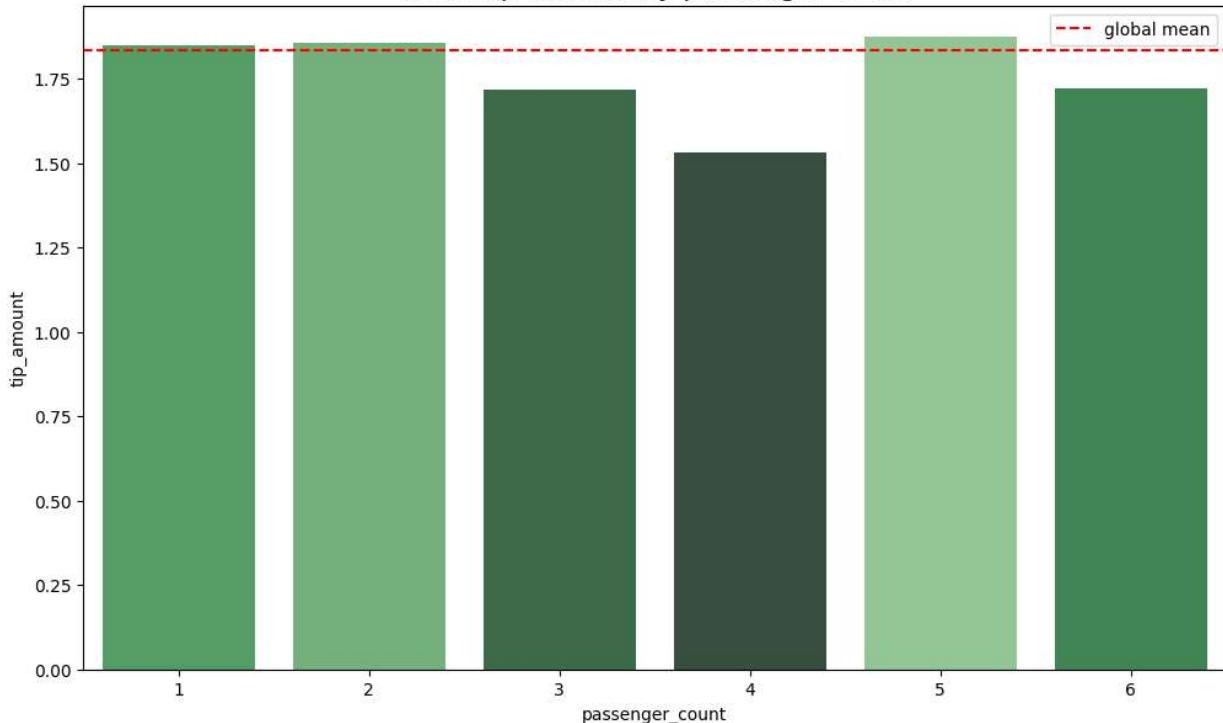
```
Out[18]: tip_amount
```

passenger_count	tip_amount
0	2.135758
1	1.848920
2	1.856378
3	1.716768
4	1.530264
5	1.873185
6	1.720260

```
In [19]: # 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);
```

Mean tip amount by passenger count



```
In [20]: # Create a month column
df['month'] = df['tpep_pickup_datetime'].dt.month_name()
# Create a day column
df['day'] = df['tpep_pickup_datetime'].dt.day_name()
```

```
In [21]: # Get total number of rides for each month
monthly_rides = df['month'].value_counts()
monthly_rides
```

```
Out[21]: March      2049
October    2027
April      2019
May        2013
January    1997
June       1964
December   1863
November   1843
February   1769
September  1734
August     1724
July       1697
Name: month, dtype: int64
```

```
In [22]: # Reorder the monthly ride list so months go in order
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
               'August', 'September', 'October', 'November', 'December']

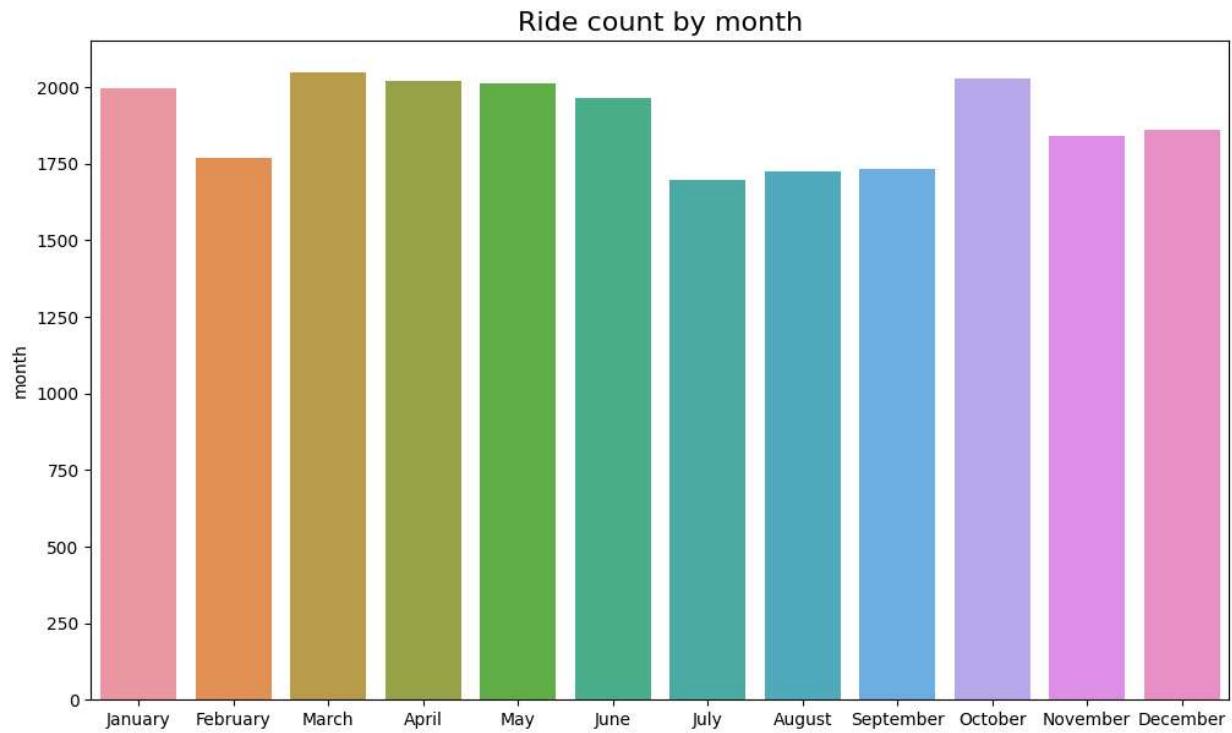
monthly_rides = monthly_rides.reindex(index=month_order)
monthly_rides
```

```
Out[22]: 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
```

```
In [23]: # Show the index
monthly_rides.index
```

```
Out[23]: Index(['January', 'February', 'March', 'April', 'May', 'June', 'July',
               'August', 'September', 'October', 'November', 'December'],
               dtype='object')
```

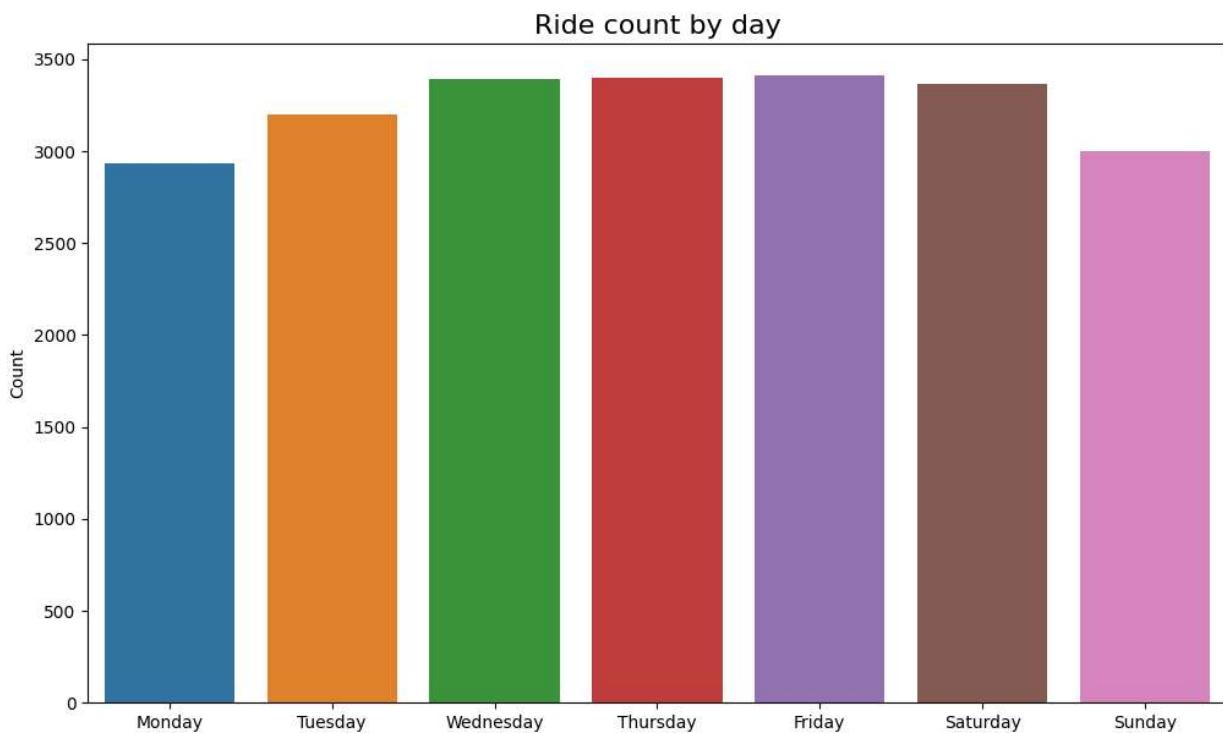
```
In [24]: # Create a bar plot of total rides per month
plt.figure(figsize=(12,7))
ax = sns.barplot(x=monthly_rides.index, y=monthly_rides)
ax.set_xticklabels(month_order)
plt.title('Ride count by month', fontsize=16);
```



```
In [25]: # 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
```

```
Out[25]: Monday    2931
          Tuesday   3198
          Wednesday  3390
          Thursday   3402
          Friday    3413
          Saturday   3367
          Sunday     2998
Name: day, dtype: int64
```

```
In [26]: # Create bar plot for ride count by day
plt.figure(figsize=(12,7))
ax = sns.barplot(x=daily_rides.index, y=daily_rides)
ax.set_xticklabels(day_order)
ax.set_ylabel('Count')
plt.title('Ride count by day', fontsize=16);
```



```
In [27]: # Repeat the process, this time for total revenue by day
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
total_amount_day = df.groupby('day').sum()[['total_amount']]
total_amount_day = total_amount_day.reindex(index=day_order)
total_amount_day
```

C:\Users\bussy\AppData\Local\Temp\ipykernel_67848\4104135426.py:3: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.

```
total_amount_day = df.groupby('day').sum()[['total_amount']]
```

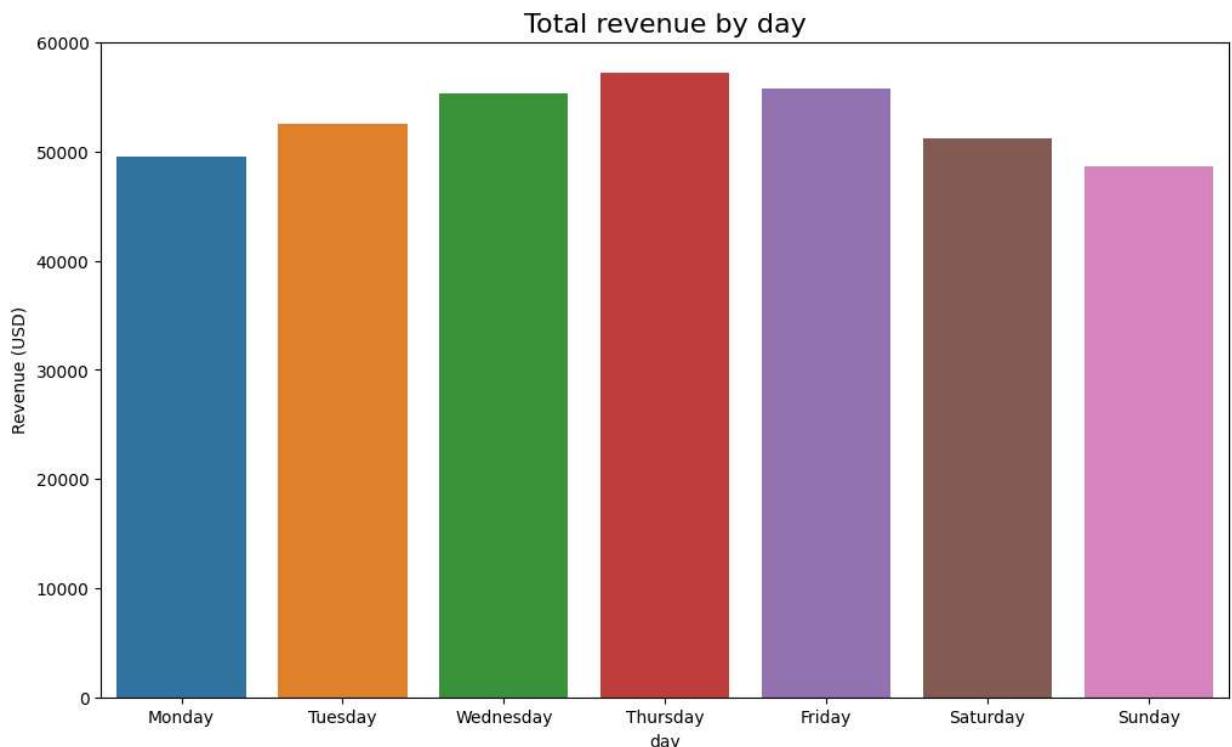
Out[27]:

total_amount

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

In [28]:

```
# Create bar plot of total revenue by day
plt.figure(figsize=(12,7))
ax = sns.barplot(x=total_amount_day.index, y=total_amount_day['total_amount'])
ax.set_xticklabels(day_order)
ax.set_ylabel('Revenue (USD)')
plt.title('Total revenue by day', fontsize=16);
```



In [29]:

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

C:\Users\bussy\AppData\Local\Temp\ipykernel_67848\749544936.py:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.

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

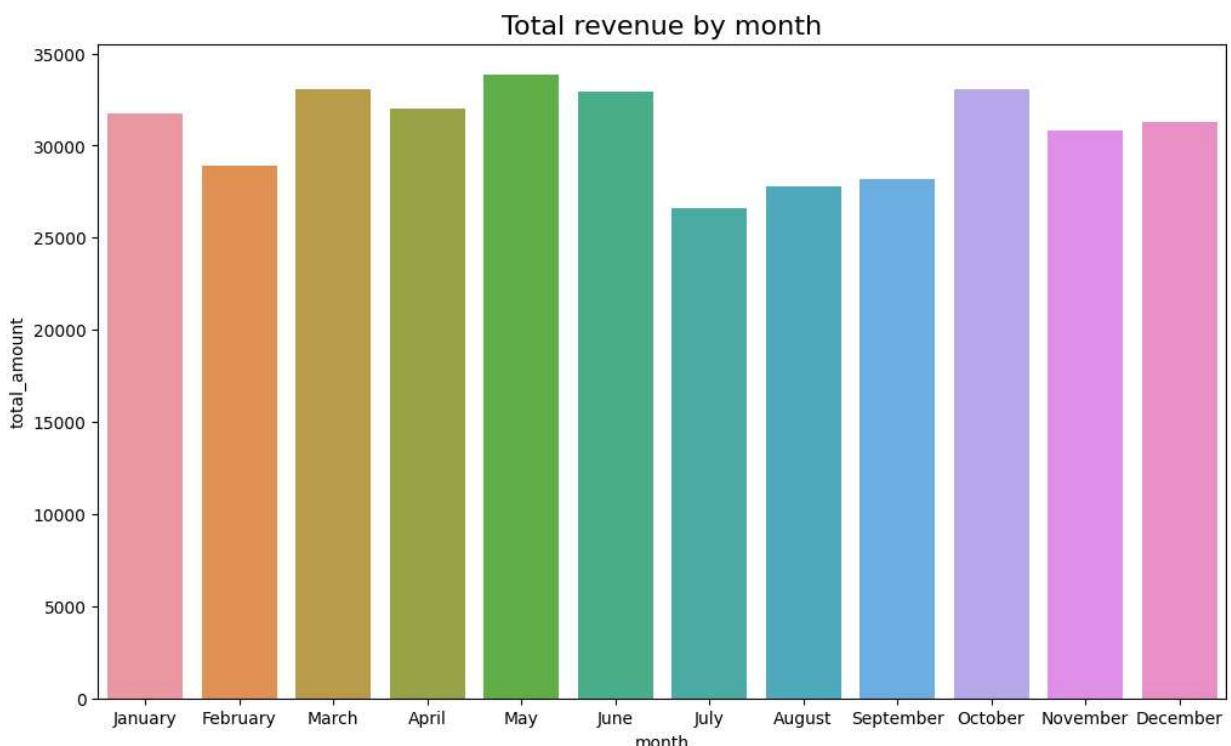
Out[29]:

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

In [30]:

```
# Create a bar plot of total revenue by month
plt.figure(figsize=(12,7))
ax = sns.barplot(x=total_amount_month.index, y=total_amount_month['total_amount'])
plt.title('Total revenue by month', fontsize=16);
```



In [31]:

```
# Get number of unique drop-off Location IDs
df['DOlocationID'].nunique()
```

Out[31]:

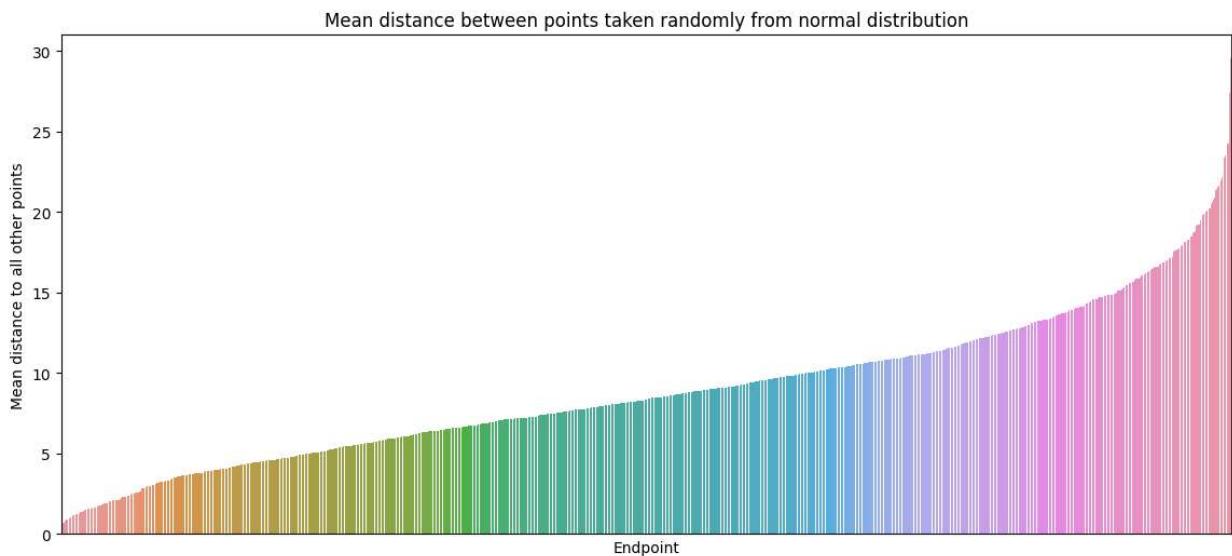
216

```
In [32]: # 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) # Calculate midpoint of the array of coordinates
start = test[:midway] # Isolate first half of array ("pick-up Locations")
end = test[midway:] # Isolate second half of array ("drop-off Locations")

# 2. Calculate Euclidean distances between points in first half and second half of arr
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
plt.figure(figsize=(14,6))
ax = sns.barplot(x=data.index,
                  y=data['distance'],
                  order=data.index)
ax.set_xticklabels([])
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');
```

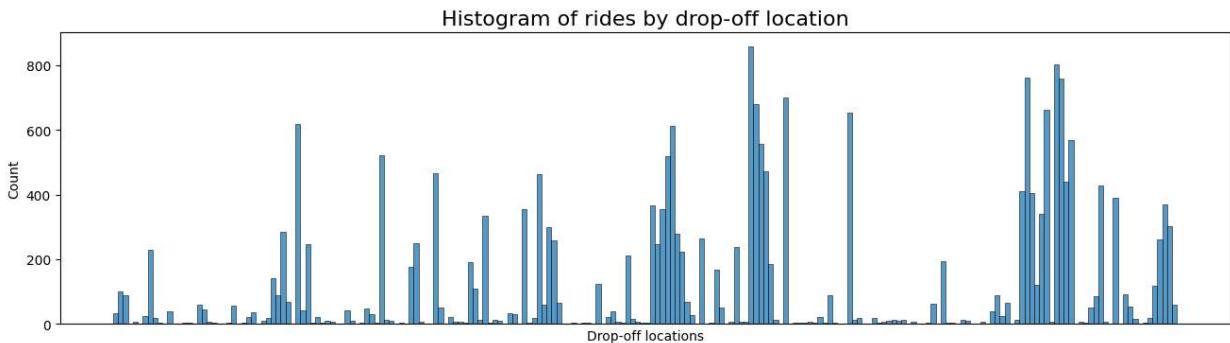


```
In [33]: # Check if all drop-off Locations are consecutively numbered
df['DOLocationID'].max() - len(set(df['DOLocationID']))
```

Out[33]: 49

```
In [34]: plt.figure(figsize=(16,4))
# DOLocationID column is numeric, so sort in ascending order
sorted_dropoffs = df['DOLocationID'].sort_values()
# Convert to string
sorted_dropoffs = sorted_dropoffs.astype('str')
# Plot
```

```
sns.histplot(sorted_dropoffs, bins=range(0, df['DOLocationID'].max()+1, 1))
plt.xticks([])
plt.xlabel('Drop-off locations')
plt.title('Histogram of rides by drop-off location', fontsize=16);
```



In [35]: `df['trip_duration'] = (df['tpep_dropoff_datetime']-df['tpep_pickup_datetime'])`

In [36]: `df.head(10)`

Out[36]:

	Unnamed: 0	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance
0	24870114	2	2017-03-25 08:55:43	2017-03-25 09:09:47	6	3.34
1	35634249	1	2017-04-11 14:53:28	2017-04-11 15:19:58	1	1.80
2	106203690	1	2017-12-15 07:26:56	2017-12-15 07:34:08	1	1.00
3	38942136	2	2017-05-07 13:17:59	2017-05-07 13:48:14	1	3.70
4	30841670	2	2017-04-15 23:32:20	2017-04-15 23:49:03	1	4.37
5	23345809	2	2017-03-25 20:34:11	2017-03-25 20:42:11	6	2.30
6	37660487	2	2017-05-03 19:04:09	2017-05-03 20:03:47	1	12.83
7	69059411	2	2017-08-15 17:41:06	2017-08-15 18:03:05	1	2.98
8	8433159	2	2017-02-04 16:17:07	2017-02-04 16:29:14	1	1.20
9	95294817	1	2017-11-10 15:20:29	2017-11-10 15:40:55	1	1.60

10 rows × 21 columns

