Activity_Course 2 Automatidata project lab

October 25, 2025

1 Automatidata project

Course 2 - Get Started with Python

Welcome to the Automatidata Project!

You have just started as a data professional in a fictional data consulting firm, Automatidata. Their client, the New York City Taxi and Limousine Commission (New York City TLC), has hired the Automatidata team for its reputation in helping their clients develop data-based solutions.

The team is still in the early stages of the project. Previously, you were asked to complete a project proposal by your supervisor, DeShawn Washington. You have received notice that your project proposal has been approved and that New York City TLC has given the Automatidata team access to their data. To get clear insights, New York TLC's data must be analyzed, key variables identified, and the dataset ensured it is ready for analysis.

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

2 Course 2 End-of-course project: Inspect and analyze data

In this activity, you will examine data provided and prepare it for analysis. This activity will help ensure the information is,

1. Ready to answer questions and yield insights
2. Ready for visualizations
3. Ready for future hypothesis testing and statistical methods

The purpose of this project is to investigate and understand the data provided.

The goal is to use a dataframe contructed within Python, perform a cursory inspection of the provided dataset, and inform team members of your findings.
This activity has three parts:

Part 1: Understand the situation * Prepare to understand and organize the provided taxi cab dataset and information.

Part 2: Understand the data

- Create a pandas dataframe for data learning, future exploratory data analysis (EDA), and statistical activities.
- Compile summary information about the data to inform next steps.

Part 3: Understand the variables

- Use insights from your examination of the summary data to guide deeper investigation into specific variables.

Follow the instructions and answer the following questions 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 Identify data types and relevant variables using Python

4 PACE stages

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

Consider the questions in your PACE Strategy Document and those below to craft your response:

4.1.1 Task 1. Understand the situation

- How can you best prepare to understand and organize the provided taxi cab information?

To best prepare for understanding and organizing the taxi data, I begin by importing the dataset into a pandas dataframe and exploring its structure. I inspect the column data types, check for missing or anomalous values, and familiarize myself with the key variables. I also review data dictionaries or documentation (if available) to understand variable meanings, which helps ensure accurate transformation and preparation for future analysis or modeling.

4.2 PACE: Analyze

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

4.2.1 Task 2a. Build dataframe

Create a pandas dataframe for data learning, and future exploratory data analysis (EDA) and statistical activities.

Code the following,

- import pandas as `pd`. pandas is used for buidling dataframes.
- import numpy as `np`. numpy is imported with pandas
- `df = pd.read_csv('Datasets\NYC taxi data.csv')`

Note: pair the data object name `df` with pandas functions to manipulate data, such as `df.groupby()`.

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.

[2]:
```
#Import libraries and packages listed
import pandas as pd
import numpy as np

# Load dataset into dataframe
df = pd.read_csv('2017_Yellow_Taxi_Trip_Data.csv')
print("done")
```

```
done
```

4.2.2 Task 2b. Understand the data - Inspect the data

View and inspect summary information about the dataframe by coding the following:

1. `df.head(10)`
2. `df.info()`
3. `df.describe()`

Consider the following two questions:

Question 1: When reviewing the `df.info()` output, what do you notice about the different variables? Are there any null values? Are all of the variables numeric? Does anything else stand out?

Question 2: When reviewing the `df.describe()` output, what do you notice about the distributions of each variable? Are there any questionable values?

1. The dataset contains 18 columns and 22,699 rows. There are no null values in any columns, which is ideal for analysis. The majority of the columns are numeric (int64 or float64), but there are also three object type columns: tpep_pickup_datetime, tpep_dropoff_datetime, and store_and_fwd_flag. These include datetime strings and categorical flags, which will need to be converted to appropriate data types before analysis. All columns appear to be fully populated.
2. Some variables contain questionable values. For example, the fare_amount column has a minimum value of -120, which is not realistic. Similarly, trip_distance includes values as low as 0.0 miles, which may indicate invalid records. The trip_duration_min ranges up to 1439

minutes (nearly 24 hours), which could be a rare but valid long trip or an anomaly. Additionally, the maximum fare_per_mile is over $5000, suggesting possible data entry errors. These outliers should be examined or removed before proceeding to modeling.

[3]:
```
df.head(10)
```

[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
5	23345809	2	03/25/2017 8:34:11 PM	03/25/2017 8:42:11 PM	6	2.30	1	N	161	236	1	9.0	0.5	0.5	2.06	0.0	0.3	12.36
6	37660487	2	05/03/2017 7:04:09 PM	05/03/2017 8:03:47 PM	1	12.83	1	N	79	241	1	47.5	1.0	0.5	9.86	0.0	0.3	59.16
7	69059411	2	08/15/2017 5:41:06 PM	08/15/2017 6:03:05 PM	1	2.98	1	N	237	114	1	16.0	1.0	0.5	1.78	0.0	0.3	19.58
8	8433159	2	02/04/2017 4:17:07 PM	02/04/2017 4:29:14 PM	1	1.20	1	N	234	249	2	9.0	0.0	0.5	0.00	0.0	0.3	9.80
9	95294817	1	11/10/2017 3:20:29 PM	11/10/2017 3:40:55 PM	1	1.60	1	N	239	237	1	13.0	0.0	0.5	2.75	0.0	0.3	16.55

```
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
```

[4]:
```
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
```

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

[5]:
```
          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
```

4.2.3 Task 2c. Understand the data - Investigate the variables

Sort and interpret the data table for two variables:`trip_distance` and `total_amount`.

Answer the following three questions:

Question 1: Sort your first variable (`trip_distance`) from maximum to minimum value, do the values seem normal?

Question 2: Sort by your second variable (`total_amount`), are any values unusual?

Question 3: Are the resulting rows similar for both sorts? Why or why not?

[10]:
```
# Step 1: Load your original CSV
import pandas as pd
import numpy as np

df = pd.read_csv('2017_Yellow_Taxi_Trip_Data.csv')
```

```
# Step 2: Clean the data - create df_cleaned
df_cleaned = df.drop(columns=['Unnamed: 0'])

# Convert datetime columns
df_cleaned['tpep_pickup_datetime'] = pd.
 ↪to_datetime(df_cleaned['tpep_pickup_datetime'])
df_cleaned['tpep_dropoff_datetime'] = pd.
 ↪to_datetime(df_cleaned['tpep_dropoff_datetime'])

# Create trip duration in minutes
df_cleaned['trip_duration_min'] = (df_cleaned['tpep_dropoff_datetime'] -␣
 ↪df_cleaned['tpep_pickup_datetime']).dt.total_seconds() / 60

# Calculate fare per mile (avoid div by zero)
df_cleaned['fare_per_mile'] = np.where(df_cleaned['trip_distance'] == 0, np.
 ↪nan, df_cleaned['fare_amount'] / df_cleaned['trip_distance'])

# Remove invalid entries
df_cleaned = df_cleaned[(df_cleaned['fare_amount'] >= 0) &␣
 ↪(df_cleaned['passenger_count'] > 0)]
```

1. When sorted by trip_distance, the majority of values seem reasonable, but the top values exceed 30 miles. While trips over 30 miles are possible (e.g., airport to outer boroughs or other cities), they are rare. The values appear plausible, but we may want to flag trips over 50 miles for closer review in future EDA.
2. Sorting by total_amount, some values are clearly extreme. For example, some fares exceed $1000, which is highly unlikely for NYC taxi rides and may indicate errors or outliers (e.g., fare of $1200 for a ~1 mile trip). These rows should be investigated or filtered before modeling. Conversely, some rows show very low or even zero total fares, which may also be problematic.
3. The rows from both sorts are not similar. Longest trips by distance are not always the ones with the highest total amount. This suggests that other factors (like tolls, tips, or flat-rate pricing) influence total fares more significantly than distance alone. Some short trips may cost more due to high surcharges or premium payment types (e.g., credit vs cash).

[11]:
```
# Sort the data by trip distance from maximum to minimum value
df_cleaned.sort_values(by='trip_distance', ascending=False).head(20)


# Sort the data by trip distance from maximum to minimum value
```

[11]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	\
9280	2	2017-06-18 23:33:25	2017-06-19 00:12:38	2	
13861	2	2017-05-19 08:20:21	2017-05-19 09:20:30	1	
6064	2	2017-06-13 12:30:22	2017-06-13 13:37:51	1	
10291	2	2017-09-11 11:41:04	2017-09-11 12:18:58	1	
29	2	2017-11-06 20:30:50	2017-11-07 00:00:00	1	

```
18130         1  2017-10-26 14:45:01   2017-10-26 16:12:49                1
5792          2  2017-08-11 14:14:01   2017-08-11 15:17:31                1
15350         2  2017-09-14 13:44:44   2017-09-14 14:34:29                1
10302         1  2017-05-15 08:11:34   2017-05-15 09:03:16                1
2592          2  2017-06-16 18:51:20   2017-06-16 19:41:42                1
20612         2  2017-08-08 21:01:00   2017-08-08 21:40:04                1
1908          2  2017-09-27 23:49:52   2017-09-28 00:26:29                2
20545         1  2017-08-24 13:03:49   2017-08-24 13:49:05                1
4138          2  2017-11-26 15:37:35   2017-11-26 16:50:22                2
15169         2  2017-08-23 16:20:15   2017-08-23 17:08:55                2
1496          2  2017-04-03 15:37:53   2017-04-03 16:44:33                1
7217          1  2017-02-20 14:28:11   2017-02-20 15:29:14                1
19483         2  2017-04-26 08:31:31   2017-04-26 11:10:50                1
908           2  2017-03-27 13:01:38   2017-03-27 13:38:44                2
4715          2  2017-09-17 20:04:24   2017-09-17 21:02:24                1

       trip_distance  RatecodeID store_and_fwd_flag  PULocationID  \
9280           33.96           5                  N           132
13861          33.92           5                  N           229
6064           32.72           3                  N           138
10291          31.95           4                  N           138
29             30.83           1                  N           132
18130          30.50           1                  N           132
5792           30.33           2                  N           132
15350          28.23           2                  N            13
10302          28.20           2                  N            90
2592           27.97           2                  N           261
20612          27.88           1                  N           132
1908           27.34           1                  N           132
20545          27.20           1                  N           132
4138           26.86           1                  N           132
15169          26.54           1                  N           132
1496           26.39           2                  N           132
7217           26.20           1                  N            13
19483          26.12           2                  N           132
908            26.12           4                  N           138
4715           25.86           4                  N            90

       DOLocationID  payment_type  fare_amount  extra  mta_tax  tip_amount  \
9280            265             2       150.00    0.0      0.0        0.00
13861           265             1       200.01    0.0      0.5       51.64
6064              1             1       107.00    0.0      0.0       55.50
10291           265             2       131.00    0.0      0.5        0.00
29               23             1        80.00    0.5      0.5       18.56
18130           220             1        90.50    0.0      0.5       19.85
5792            158             1        52.00    0.0      0.5       14.64
15350           132             1        52.00    0.0      0.5        4.40
```

10302	132	1	52.00	0.0	0.5	11.71
2592	132	2	52.00	4.5	0.5	0.00
20612	181	1	73.00	0.5	0.5	14.86
1908	181	1	72.50	0.5	0.5	14.76
20545	54	1	73.00	0.0	0.5	14.75
4138	52	1	75.00	0.0	0.5	18.95
15169	195	1	70.50	1.0	0.5	14.46
1496	261	1	52.00	0.0	0.5	11.71
7217	132	1	74.00	0.0	0.5	0.00
19483	48	2	52.00	0.0	0.5	0.00
908	265	1	100.00	0.0	0.5	15.00
4715	265	2	78.00	0.5	0.5	0.00

	tolls_amount	improvement_surcharge	total_amount	trip_duration_min	\
9280	0.00	0.3	150.30	39.216667	
13861	5.76	0.3	258.21	60.150000	
6064	16.26	0.3	179.06	67.483333	
10291	0.00	0.3	131.80	37.900000	
29	11.52	0.3	111.38	209.166667	
18130	8.16	0.3	119.31	87.800000	
5792	5.76	0.3	73.20	63.500000	
15350	5.76	0.3	62.96	49.750000	
10302	5.76	0.3	70.27	51.700000	
2592	5.76	0.3	63.06	50.366667	
20612	0.00	0.3	89.16	39.066667	
1908	0.00	0.3	88.56	36.616667	
20545	0.00	0.3	88.55	45.266667	
4138	0.00	0.3	94.75	72.783333	
15169	0.00	0.3	86.76	48.666667	
1496	5.76	0.3	70.27	66.666667	
7217	0.00	0.3	74.80	61.050000	
19483	5.76	0.3	58.56	159.316667	
908	5.76	0.3	121.56	37.100000	
4715	5.76	0.3	85.06	58.000000	

	fare_per_mile
9280	4.416961
13861	5.896521
6064	3.270171
10291	4.100156
29	2.594875
18130	2.967213
5792	1.714474
15350	1.842012
10302	1.843972
2592	1.859135
20612	2.618364

```
1908        2.651792
20545       2.683824
4138        2.792256
15169       2.656368
1496        1.970443
7217        2.824427
19483       1.990812
908         3.828484
4715        3.016241
```

[12]:
```
# Sort the data by total amount and print the top 20 values
df_cleaned.sort_values(by='total_amount', ascending=False).head(20)


# Sort the data by total amount and print the top 20 values
```

[12]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID
8476	1	2017-02-06 05:50:10	2017-02-06 05:51:08	1	2.60	5	N	226
20312	2	2017-12-19 09:40:46	2017-12-19 09:40:55	2	0.00	5	N	265
13861	2	2017-05-19 08:20:21	2017-05-19 09:20:30	1	33.92	5	N	229
12511	2	2017-12-17 18:24:24	2017-12-17 18:24:42	1	0.00	5	N	265
15474	2	2017-06-06 20:55:01	2017-06-06 20:55:06	1	0.00	5	N	265
6064	2	2017-06-13 12:30:22	2017-06-13 13:37:51	1	32.72	3	N	138
16379	2	2017-11-30 10:41:11	2017-11-30 11:31:45	1	25.50	5	N	132
3582	1	2017-01-01 23:53:01	2017-01-01 23:53:42	1	7.30	5	N	1
11269	1	2017-06-19 00:51:17	2017-06-19 00:52:12	2	0.00	5	N	265
9280	2	2017-06-18 23:33:25	2017-06-19 00:12:38	2	0.00	5	N	265
1928	1	2017-06-16 18:30:08	2017-06-16 19:18:50	2	0.00	5	N	265
10291	2	2017-09-11 11:41:04	2017-09-11 12:18:58	1	0.00	5	N	265
6708	2	2017-10-30 11:23:46	2017-10-30 11:23:49	1	0.00	5	N	265
11608	2	2017-12-19 17:00:56	2017-12-19 18:41:56	2	0.00	5	N	265
908	2	2017-03-27 13:01:38	2017-03-27 13:38:44	2	0.00	5	N	265
7281	2	2017-01-01 03:02:53	2017-01-01 03:03:02	1	0.00	5	N	265
18130	1	2017-10-26 14:45:01	2017-10-26 16:12:49	1	0.00	5	N	265
13621	1	2017-11-04 13:32:14	2017-11-04 14:18:50	2	0.00	5	N	265
13359	1	2017-01-12 07:19:36	2017-01-12 07:19:56	1	0.00	5	N	265
29	2	2017-11-06 20:30:50	2017-11-07 00:00:00	1	0.00	5	N	265

11269	0.00	5	N	265
9280	33.96	5	N	132
1928	12.50	5	N	211
10291	31.95	4	N	138
6708	0.32	5	N	264
11608	23.00	3	N	151
908	26.12	4	N	138
7281	0.00	5	N	265
18130	30.50	1	N	132
13621	19.80	5	N	265
13359	0.00	5	N	1
29	30.83	1	N	132

	DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount	\
8476	226	1	999.99	0.0	0.0	200.00	
20312	265	2	450.00	0.0	0.0	0.00	
13861	265	1	200.01	0.0	0.5	51.64	
12511	265	1	175.00	0.0	0.0	46.69	
15474	265	1	200.00	0.0	0.5	11.00	
6064	1	1	107.00	0.0	0.0	55.50	
16379	265	2	140.00	0.0	0.5	0.00	
3582	1	1	152.00	0.0	0.0	0.00	
11269	265	1	120.00	0.0	0.0	20.00	
9280	265	2	150.00	0.0	0.0	0.00	
1928	265	1	120.00	0.0	0.0	5.00	
10291	265	2	131.00	0.0	0.5	0.00	
6708	83	1	100.00	0.0	0.5	25.20	
11608	1	1	99.50	1.0	0.0	10.00	
908	265	1	100.00	0.0	0.5	15.00	
7281	265	1	100.00	0.0	0.5	20.16	
18130	220	1	90.50	0.0	0.5	19.85	
13621	230	1	105.00	0.0	0.0	8.00	
13359	1	1	75.00	0.0	0.0	18.65	
29	23	1	80.00	0.5	0.5	18.56	

	tolls_amount	improvement_surcharge	total_amount	trip_duration_min	\
8476	0.00	0.3	1200.29	0.966667	
20312	0.00	0.3	450.30	0.150000	
13861	5.76	0.3	258.21	60.150000	
12511	11.75	0.3	233.74	0.300000	
15474	0.00	0.3	211.80	0.083333	
6064	16.26	0.3	179.06	67.483333	
16379	16.26	0.3	157.06	50.566667	
3582	0.00	0.3	152.30	0.683333	
11269	11.52	0.3	151.82	0.916667	
9280	0.00	0.3	150.30	39.216667	
1928	12.50	0.3	137.80	48.700000	

```
10291          0.00                    0.3        131.80          37.900000
6708           0.00                    0.3        126.00           0.050000
11608         12.50                    0.3        123.30         101.000000
908            5.76                    0.3        121.56          37.100000
7281           0.00                    0.3        120.96           0.150000
18130          8.16                    0.3        119.31          87.800000
13621          2.64                    0.3        115.94          46.600000
13359         18.00                    0.3        111.95           0.333333
29            11.52                    0.3        111.38         209.166667

       fare_per_mile
8476      384.611538
20312            NaN
13861       5.896521
12511            NaN
15474            NaN
6064        3.270171
16379       5.490196
3582       20.821918
11269            NaN
9280        4.416961
1928        9.600000
10291       4.100156
6708      312.500000
11608       4.326087
908         3.828484
7281             NaN
18130       2.967213
13621       5.303030
13359            NaN
29          2.594875
```

[13]:
```
# Sort the data by total amount and print the bottom 20 values
df_cleaned.sort_values(by='total_amount', ascending=True).head(20)


# Sort the data by total amount and print the bottom 20 values
```

[13]:
```
       VendorID tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
10506         2  2017-03-30 03:14:26   2017-03-30 03:14:28                1
4402          2  2017-12-20 16:06:53   2017-12-20 16:47:50                1
22566         2  2017-03-07 02:24:47   2017-03-07 02:24:50                1
5722          2  2017-06-12 12:08:55   2017-06-12 12:08:57                1
19067         1  2017-07-10 14:40:09   2017-07-10 14:40:59                1
14283         1  2017-05-03 19:44:28   2017-05-03 19:44:38                1
13970         2  2017-02-23 09:21:25   2017-02-23 09:21:57                1
20133         2  2017-07-03 15:00:37   2017-07-03 15:00:58                1
```

```
18697         1  2017-12-01 12:03:08   2017-12-01 12:03:13                1
12144         2  2017-12-28 08:13:08   2017-12-28 08:13:35                1
17579         1  2017-04-09 09:25:29   2017-04-09 09:25:41                1
22035         2  2017-12-10 12:20:19   2017-12-10 12:20:56                1
13062         1  2017-10-10 09:53:00   2017-10-10 09:53:00                1
11357         2  2017-12-25 10:28:28   2017-12-25 10:28:38                2
15346         2  2017-01-17 13:18:24   2017-01-17 13:18:31                1
17747         1  2017-06-13 14:45:20   2017-06-13 14:46:07                1
14468         2  2017-09-09 13:29:37   2017-09-09 13:29:57                3
9529          1  2017-07-01 11:48:28   2017-07-01 11:48:47                1
3734          1  2017-02-20 16:58:03   2017-02-20 16:58:15                1
19838         1  2017-09-24 13:37:55   2017-09-24 13:37:55                1

       trip_distance  RatecodeID store_and_fwd_flag  PULocationID  \
10506           0.00           1                  N           264
4402            7.06           1                  N           263
22566           0.00           1                  N           264
5722            0.00           1                  N           264
19067           0.10           5                  N           261
14283           0.00           5                  N           146
13970           0.00           5                  N             7
20133           0.02           1                  N           197
18697           0.00           1                  N           264
12144           0.00           1                  N           193
17579           0.00           1                  N           132
22035           0.09           1                  N           234
13062           0.00           1                  N           186
11357           0.06           1                  N           141
15346           0.02           1                  N           246
17747           0.10           1                  N           144
14468           0.02           1                  N           162
9529            0.00           1                  N           162
3734            0.60           1                  N           265
19838           0.00           1                  N           234

       DOLocationID  payment_type  fare_amount  extra  mta_tax  tip_amount  \
10506           193             1         0.00    0.0      0.0         0.0
4402            169             2         0.00    0.0      0.0         0.0
22566           193             1         0.00    0.0      0.0         0.0
5722            193             1         0.00    0.0      0.0         0.0
19067            13             3         0.00    0.0      0.0         0.0
14283           146             3         0.01    0.0      0.0         0.0
13970             7             2         1.00    0.0      0.0         0.0
20133           197             2         2.50    0.0      0.5         0.0
18697           264             2         2.50    0.0      0.5         0.0
12144             7             1         2.50    0.0      0.5         0.0
17579           132             3         2.50    0.0      0.5         0.0
```

```
22035           234              2         2.50    0.0      0.5         0.0
13062           264              2         2.50    0.0      0.5         0.0
11357           141              2         2.50    0.0      0.5         0.0
15346           246              2         2.50    0.0      0.5         0.0
17747           144              3         2.50    0.0      0.5         0.0
14468           162              2         2.50    0.0      0.5         0.0
9529            162              4         2.50    0.0      0.5         0.0
3734            265              3         2.50    0.0      0.5         0.0
19838           264              2         2.50    0.0      0.5         0.0

       tolls_amount  improvement_surcharge  total_amount  trip_duration_min  \
10506           0.0                    0.0          0.00           0.033333
4402            0.0                    0.0          0.00          40.950000
22566           0.0                    0.0          0.00           0.050000
5722            0.0                    0.0          0.00           0.033333
19067           0.0                    0.3          0.30           0.833333
14283           0.0                    0.3          0.31           0.166667
13970           0.0                    0.3          1.30           0.533333
20133           0.0                    0.3          3.30           0.350000
18697           0.0                    0.3          3.30           0.083333
12144           0.0                    0.3          3.30           0.450000
17579           0.0                    0.3          3.30           0.200000
22035           0.0                    0.3          3.30           0.616667
13062           0.0                    0.3          3.30           0.000000
11357           0.0                    0.3          3.30           0.166667
15346           0.0                    0.3          3.30           0.116667
17747           0.0                    0.3          3.30           0.783333
14468           0.0                    0.3          3.30           0.333333
9529            0.0                    0.3          3.30           0.316667
3734            0.0                    0.3          3.30           0.200000
19838           0.0                    0.3          3.30           0.000000

       fare_per_mile
10506            NaN
4402        0.000000
22566            NaN
5722             NaN
19067       0.000000
14283            NaN
13970            NaN
20133     125.000000
18697            NaN
12144            NaN
17579            NaN
22035      27.777778
13062            NaN
11357      41.666667
```

```
15346     125.000000
17747      25.000000
14468     125.000000
9529             NaN
3734        4.166667
19838            NaN
```

[14]:
```
# How many of each payment type are represented in the data?
df_cleaned['payment_type'].value_counts()

# How many of each payment type are represented in the data?
```

[14]:
```
1    15238
2     7261
3      114
4       39
Name: payment_type, dtype: int64
```

According to the data dictionary, the payment method was encoded as follows:

1 = Credit card
2 = Cash
3 = No charge
4 = Dispute
5 = Unknown
6 = Voided trip

[21]:
```
# What is the average tip for trips paid for with credit card?
credit_card_tip_avg = df_cleaned[df_cleaned['payment_type'] == 1]['tip_amount'].
 ↪mean()
print("Average Tip (Credit Card):", credit_card_tip_avg)
# What is the average tip for trips paid for with cash?
cash_tip_avg = df_cleaned[df_cleaned['payment_type'] == 2]['tip_amount'].mean()
print("Average Tip (Cash):", cash_tip_avg)
```

```
Average Tip (Credit Card): 2.730011812573817
Average Tip (Cash): 0.0
```

[16]:
```
df_cleaned['VendorID'].value_counts()

# How many times is each vendor ID represented in the data?
```

[16]:
```
2    12612
1    10040
Name: VendorID, dtype: int64
```

[17]:
```
df_cleaned.groupby('VendorID')['total_amount'].mean()
```

```
# What is the mean total amount for each vendor?
```

[17]:
```
VendorID
1    16.293760
2    16.352747
Name: total_amount, dtype: float64
```

[20]:
```
credit_card_df = df_cleaned[df_cleaned['payment_type'] == 1]
# Filter the data for credit card payments only

credit_card_df = credit_card_df[credit_card_df['passenger_count'] > 0]

# Filter the credit-card-only data for passenger count only
```

[19]:
```
credit_card_df.groupby('passenger_count')['tip_amount'].mean()

# Calculate the average tip amount for each passenger count (credit card␣
↪payments only)
```

[19]:
```
passenger_count
1    2.714681
2    2.829949
3    2.726800
4    2.607753
5    2.762645
6    2.643326
Name: tip_amount, dtype: float64
```

4.3 PACE: Construct

Note: The Construct stage does not apply to this workflow. The PACE framework can be adapted to fit the specific requirements of any project.

4.4 PACE: Execute

Consider the questions in your PACE Strategy Document and those below to craft your response.

4.4.1 Given your efforts, what can you summarize for DeShawn and the data team?

Note for Learners: Your notebook should contain data that can address Luana's requests. Which two variables are most helpful for building a predictive model for the client: NYC TLC?

After analyzing the NYC Yellow Taxi Trip data, several insights emerged that can support the client's goals:

Tipping Behavior:

The average tip for credit card payments is approximately $2.73.

In contrast, cash payments record $0.00 average tips — likely due to underreporting or non-recorded entries.

This makes payment_type a strong candidate for predictive modeling of tips.

Passenger Count Impact:

Trips with 2 or more passengers show slightly higher average tips than solo rides, peaking around 2.83.

Thus, passenger_count may be another useful predictor for tip behavior.

Vendor Metrics:

Both major vendors (IDs 1 and 2) show similar mean total amounts (~$16.3), indicating consistent fare structures.

Recommended Predictive Model Variables:

payment_type – captures whether tipping data is likely to exist.

passenger_count – shows variation in tipping patterns by party size.

These variables are practical, interpretable, and aligned with NYC TLC's goal to better understand and possibly predict tipping behaviors based on ride characteristics.

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.