

# Activity\_Course 2 Automatidata project lab

December 21, 2023

## 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 City 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 constructed 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?

==> ENTER YOUR RESPONSE HERE

##### 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 building 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.

```
[4]: #Import libraries and packages listed above
    ### YOUR CODE HERE ###
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?

==> ENTER YOUR RESPONSE TO QUESTIONS 1 & 2 HERE

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

```
[5]:   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

```
[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

```

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

```

[7]:      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?

==> ENTER YOUR RESPONSES TO QUESTION 1-3 HERE

```
[8]: # ==> ENTER YOUR CODE HERE
df_sort = df.sort_values(by=['trip_distance'], ascending=False)
df_sort.head(10)

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

```
[8]: Unnamed: 0 VendorID tpep_pickup_datetime tpep_dropoff_datetime \
9280 51810714 2 06/18/2017 11:33:25 PM 06/19/2017 12:12:38 AM
13861 40523668 2 05/19/2017 8:20:21 AM 05/19/2017 9:20:30 AM
6064 49894023 2 06/13/2017 12:30:22 PM 06/13/2017 1:37:51 PM
10291 76319330 2 09/11/2017 11:41:04 AM 09/11/2017 12:18:58 PM
29 94052446 2 11/06/2017 8:30:50 PM 11/07/2017 12:00:00 AM
18130 90375786 1 10/26/2017 2:45:01 PM 10/26/2017 4:12:49 PM
5792 68023798 2 08/11/2017 2:14:01 PM 08/11/2017 3:17:31 PM
15350 77309977 2 09/14/2017 1:44:44 PM 09/14/2017 2:34:29 PM
10302 43431843 1 05/15/2017 8:11:34 AM 05/15/2017 9:03:16 AM
2592 51094874 2 06/16/2017 6:51:20 PM 06/16/2017 7:41:42 PM
```

	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	\
9280	2	33.96	5	N	
13861	1	33.92	5	N	

6064	1	32.72	3	N
10291	1	31.95	4	N
29	1	30.83	1	N
18130	1	30.50	1	N
5792	1	30.33	2	N
15350	1	28.23	2	N
10302	1	28.20	2	N
2592	1	27.97	2	N

	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	\
9280	132	265	2	150.00	0.0	0.0	
13861	229	265	1	200.01	0.0	0.5	
6064	138	1	1	107.00	0.0	0.0	
10291	138	265	2	131.00	0.0	0.5	
29	132	23	1	80.00	0.5	0.5	
18130	132	220	1	90.50	0.0	0.5	
5792	132	158	1	52.00	0.0	0.5	
15350	13	132	1	52.00	0.0	0.5	
10302	90	132	1	52.00	0.0	0.5	
2592	261	132	2	52.00	4.5	0.5	

	tip_amount	tolls_amount	improvement_surcharge	total_amount
9280	0.00	0.00	0.3	150.30
13861	51.64	5.76	0.3	258.21
6064	55.50	16.26	0.3	179.06
10291	0.00	0.00	0.3	131.80
29	18.56	11.52	0.3	111.38
18130	19.85	8.16	0.3	119.31
5792	14.64	5.76	0.3	73.20
15350	4.40	5.76	0.3	62.96
10302	11.71	5.76	0.3	70.27
2592	0.00	5.76	0.3	63.06

```
[9]: #==> ENTER YOUR CODE HERE
total_amount_sorted = df.sort_values(
    ['total_amount'], ascending=False)['total_amount']
total_amount_sorted.head(20)
# Sort the data by total amount and print the top 20 values
```

```
[9]: 8476      1200.29
20312      450.30
13861      258.21
12511      233.74
15474      211.80
6064       179.06
16379      157.06
3582       152.30
```

```
11269    151.82
9280     150.30
1928     137.80
10291    131.80
6708     126.00
11608    123.30
908      121.56
7281     120.96
18130    119.31
13621    115.94
13359    111.95
29       111.38
Name: total_amount, dtype: float64
```

```
[10]: ##=> ENTER YOUR CODE HERE
total_amount_sorted.tail(20)
# Sort the data by total amount and print the bottom 20 values
```

```
[10]: 14283     0.31
19067     0.30
10506     0.00
5722      0.00
4402      0.00
22566     0.00
1646     -3.30
18565    -3.80
314       -3.80
5758     -3.80
5448     -4.30
4423     -4.30
10281    -4.30
8204     -4.80
20317    -4.80
11204    -5.30
14714    -5.30
17602    -5.80
20698    -5.80
12944   -120.30
Name: total_amount, dtype: float64
```

```
[11]: ##=> ENTER YOUR CODE HERE
df['payment_type'].value_counts()
# How many of each payment type are represented in the data?
```

```
[11]: 1    15265
2     7267
3     121
```



```
4      46
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
```

```
[13]: ===> ENTER YOUR CODE HERE
avg_cc_tip = df[df['payment_type']==1]['tip_amount'].mean()
print('Avg. cc tip:', avg_cc_tip)
# What is the average tip for trips paid for with credit card?

===> ENTER YOUR CODE HERE
avg_cash_tip = df[df['payment_type']==2]['tip_amount'].mean()
print('Avg. cash tip:', avg_cash_tip)
# What is the average tip for trips paid for with cash?
```

```
Avg. cc tip: 2.7298001965279934
Avg. cash tip: 0.0
```

```
[14]: ===> ENTER YOUR CODE HERE
df['VendorID'].value_counts()
# How many times is each vendor ID represented in the data?
```

```
[14]: 2    12626
      1    10073
Name: VendorID, dtype: int64
```

```
[15]: ===> ENTER YOUR CODE HERE
df.groupby(['VendorID']).mean(numeric_only=True)[['total_amount']]
# What is the mean total amount for each vendor?
```

```
[15]:          total_amount
VendorID
1          16.298119
2          16.320382
```

```
[16]: ===> ENTER YOUR CODE HERE
credit_card = df[df['payment_type']==1]
# Filter the data for credit card payments only

===> ENTER YOUR CODE HERE
credit_card['passenger_count'].value_counts()
```

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

```
[16]: 1    10977
      2     2168
      5     775
      3     600
      6     451
      4     267
      0      27
      Name: passenger_count, dtype: int64
```

```
[17]: #==> ENTER YOUR CODE HERE

# Calculate the average tip amount for each passenger count (credit card_
↳payments only)
credit_card.groupby(['passenger_count']).mean(numeric_only=True)[['tip_amount']]
```

```
[17]:
```

passenger_count	tip_amount
0	2.610370
1	2.714681
2	2.829949
3	2.726800
4	2.607753
5	2.762645
6	2.643326

### 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?*

==> ENTER YOUR RESPONSE HERE

**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.