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.

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.

Identify data types and relevant variables using Python

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.

PACE: Plan

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

Task 1. Understand the situation

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

1. Ensure that the variables within the dataset provided are understood and any outstanding questions about the data are asked before any analysis begins.
2. Once you've conducted an initial examination of the data, identify ways to structure it for easier analysis. Consider renaming columns, combining or disregarding certain values, and determining the appropriate data types for each column.

PACE: Analyze

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

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.

In [2]:
```
#Import libraries and packages listed above
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
```

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. There do not appear to be any null values within the data frame. Not all of the variables are numeric; some datetime and object variables are also present.
2. The fare_amount variable indicates a min value of -120. This could pertain to refunds where the system shows a negative balance, but further research to determine this will be required. There also appears to be a fare amount of 999, which could be an error and should be further investigated.
3. The summary table shows that some variables may have a higher distribution skew than others. However, with such a high number of observations, the skew inherent within these distributions should be insignificant to impact analysis negatively.

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

Out[3]:

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

In [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
```

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

Out[5]:

	Unnamed: 0	VendorID	passenger_count	trip_distance	RatecodeID	PULocation
count	2.269900e+04	22699.000000	22699.000000	22699.000000	22699.000000	22699.0000
mean	5.675849e+07	1.556236	1.642319	2.913313	1.043394	162.4123
std	3.274493e+07	0.496838	1.285231	3.653171	0.708391	66.6333
min	1.212700e+04	1.000000	0.000000	0.000000	1.000000	1.0000
25%	2.852056e+07	1.000000	1.000000	0.990000	1.000000	114.0000
50%	5.673150e+07	2.000000	1.000000	1.610000	1.000000	162.0000
75%	8.537452e+07	2.000000	2.000000	3.060000	1.000000	233.0000
max	1.134863e+08	2.000000	6.000000	33.960000	99.000000	265.0000

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?

1. Sorting trip_distance from highest to lowest indicates a value of 33.92, which seems very high and out of place. This could be an error and requires further investigation, but aligns with our previous discovery of 33 miles.
2. Sorting by total_amount indicates a value of 1200.29 and other relatively high values of 450.30 and 282.21, respectively. Looking at trip_distance in conjunction with total_amount, there are discrepancies, as the values seem much lower than they should, considering the cost of the trip.
3. The most expensive rides don't appear to necessarily be the longest ones.

In [10]:
```
# Sort the data by trip distance from maximum to minimum value

trip_distance_sorted = df.sort_values(by='trip_distance', ascending = False)
trip_distance_sorted.head(20)
```

Out[10]:

	Unnamed: 0	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count
9280	51810714	2	06/18/2017 11:33:25 PM	06/19/2017 12:12:38 AM	2
13861	40523668	2	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	1
6064	49894023	2	06/13/2017 12:30:22 PM	06/13/2017 1:37:51 PM	1
10291	76319330	2	09/11/2017 11:41:04 AM	09/11/2017 12:18:58 PM	1
29	94052446	2	11/06/2017 8:30:50 PM	11/07/2017 12:00:00 AM	1
18130	90375786	1	10/26/2017 2:45:01 PM	10/26/2017 4:12:49 PM	1
5792	68023798	2	08/11/2017 2:14:01 PM	08/11/2017 3:17:31 PM	1
15350	77309977	2	09/14/2017 1:44:44 PM	09/14/2017 2:34:29 PM	1
10302	43431843	1	05/15/2017 8:11:34 AM	05/15/2017 9:03:16 AM	1
2592	51094874	2	06/16/2017 6:51:20 PM	06/16/2017 7:41:42 PM	1
20612	67238347	2	08/08/2017 9:01:00 PM	08/08/2017 9:40:04 PM	1
1908	81396251	2	09/27/2017 11:49:52 PM	09/28/2017 12:26:29 AM	2
20545	71466504	1	08/24/2017 1:03:49 PM	08/24/2017 1:49:05 PM	1
4138	100097208	2	11/26/2017 3:37:35 PM	11/26/2017 4:50:22 PM	2
15169	71224174	2	08/23/2017 4:20:15 PM	08/23/2017 5:08:55 PM	2
1496	27465882	2	04/03/2017 3:37:53 PM	04/03/2017 4:44:33 PM	1
7217	13572254	1	02/20/2017 2:28:11 PM	02/20/2017 3:29:14 PM	1
908	25075013	2	03/27/2017 1:01:38 PM	03/27/2017 1:38:44 PM	2
19483	34486843	2	04/26/2017 8:31:31 AM	04/26/2017 11:10:50 AM	1
4715	78418799	2	09/17/2017 8:04:24 PM	09/17/2017 9:02:24 PM	1

In [15]:
```
# Sort the data by total amount and print the top 20 values

total_amount_sorted_top20 = df.sort_values(by='total_amount', ascending = False)
total_amount_sorted_top20.head(20)
```

Out[15]:

	Unnamed: 0	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count
8476	11157412	1	02/06/2017 5:50:10 AM	02/06/2017 5:51:08 AM	1
20312	107558404	2	12/19/2017 9:40:46 AM	12/19/2017 9:40:55 AM	2
13861	40523668	2	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	1
12511	107108848	2	12/17/2017 6:24:24 PM	12/17/2017 6:24:42 PM	1
15474	55538852	2	06/06/2017 8:55:01 PM	06/06/2017 8:55:06 PM	1
6064	49894023	2	06/13/2017 12:30:22 PM	06/13/2017 1:37:51 PM	1
16379	101198443	2	11/30/2017 10:41:11 AM	11/30/2017 11:31:45 AM	1
3582	111653084	1	01/01/2017 11:53:01 PM	01/01/2017 11:53:42 PM	1
11269	51920669	1	06/19/2017 12:51:17 AM	06/19/2017 12:52:12 AM	2
9280	51810714	2	06/18/2017 11:33:25 PM	06/19/2017 12:12:38 AM	2
1928	51087145	1	06/16/2017 6:30:08 PM	06/16/2017 7:18:50 PM	2
10291	76319330	2	09/11/2017 11:41:04 AM	09/11/2017 12:18:58 PM	1
6708	91660295	2	10/30/2017 11:23:46 AM	10/30/2017 11:23:49 AM	1
11608	107690629	2	12/19/2017 5:00:56 PM	12/19/2017 6:41:56 PM	2
908	25075013	2	03/27/2017 1:01:38 PM	03/27/2017 1:38:44 PM	2
7281	111091850	2	01/01/2017 3:02:53 AM	01/01/2017 3:03:02 AM	1
18130	90375786	1	10/26/2017 2:45:01 PM	10/26/2017 4:12:49 PM	1
13621	93330154	1	11/04/2017 1:32:14 PM	11/04/2017 2:18:50 PM	2
13359	3055315	1	01/12/2017 7:19:36 AM	01/12/2017 7:19:56 AM	1
29	94052446	2	11/06/2017 8:30:50 PM	11/07/2017 12:00:00 AM	1

In [59]:
```
# Sort the data by total amount and print the bottom 20 values

total_amount_sorted_top20.tail(20)
```

Out[59]:

	Unnamed: 0	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count
14283	37675840	1	05/03/2017 7:44:28 PM	05/03/2017 7:44:38 PM	1
19067	58713019	1	07/10/2017 2:40:09 PM	07/10/2017 2:40:59 PM	1
10506	26005024	2	03/30/2017 3:14:26 AM	03/30/2017 3:14:28 AM	1
5722	49670364	2	06/12/2017 12:08:55 PM	06/12/2017 12:08:57 PM	1
4402	108016954	2	12/20/2017 4:06:53 PM	12/20/2017 4:47:50 PM	1
22566	19022898	2	03/07/2017 2:24:47 AM	03/07/2017 2:24:50 AM	1
1646	57337183	2	07/05/2017 11:02:23 AM	07/05/2017 11:03:00 AM	1
18565	43859760	2	05/22/2017 3:51:20 PM	05/22/2017 3:52:22 PM	1
314	105454287	2	12/13/2017 2:02:39 AM	12/13/2017 2:03:08 AM	6
5758	833948	2	01/03/2017 8:15:23 PM	01/03/2017 8:15:39 PM	1
5448	28459983	2	04/06/2017 12:50:26 PM	04/06/2017 12:52:39 PM	1
4423	97329905	2	11/16/2017 8:13:30 PM	11/16/2017 8:14:50 PM	2
10281	55302347	2	06/05/2017 5:34:25 PM	06/05/2017 5:36:29 PM	2
8204	91187947	2	10/28/2017 8:39:36 PM	10/28/2017 8:41:59 PM	1
20317	75926915	2	09/09/2017 10:59:51 PM	09/09/2017 11:02:06 PM	1
11204	58395501	2	07/09/2017 7:20:59 AM	07/09/2017 7:23:50 AM	1
14714	109276092	2	12/24/2017 10:37:58 PM	12/24/2017 10:41:08 PM	5
17602	24690146	2	03/24/2017 7:31:13 PM	03/24/2017 7:34:49 PM	1
20698	14668209	2	02/24/2017 12:38:17 AM	02/24/2017 12:42:05 AM	1
12944	29059760	2	04/08/2017 12:00:16 AM	04/08/2017 11:15:57 PM	1

In [143]:
```
# How many of each payment type are represented in the data?

df['payment_type'].value_counts()
```

Out[143]:
```
1    15265
2     7267
3      121
4       46
Name: payment_type, dtype: int64
```

In [138]:
```
# What is the average tip for trips paid for with credit card?

mean_tip_amount_cc = df[df['payment_type'] == 1]['tip_amount'].mean()
rounded_mean_tip_amount_cc = round(mean_tip_amount_cc, 2)
print("Mean tip amount for payment type 1:", rounded_mean_tip_amount_cc)

# What is the average tip for trips paid for with cash?

mean_tip_amount_cash = df[df['payment_type'] == 2]['tip_amount'].mean()
rounded_mean_tip_amount_cash = round(mean_tip_amount_cash, 2)
print("Mean tip amount for payment type 2:", rounded_mean_tip_amount_cas
h)
```

```
Mean tip amount for payment type 1: 2.73
Mean tip amount for payment type 2: 0.0
```

In [86]:
```
# How many times is each vendor ID represented in the data?

vendor_type_grouped = df.groupby('VendorID')['VendorID'].count()
print(vendor_type_grouped)
```

```
VendorID
1    10073
2    12626
Name: VendorID, dtype: int64
```

In [142]:
```
# What is the mean total amount for each vendor?

vendor_ids = [1, 2]

for vendor_id in vendor_ids:
    mean_total_amount = df[df['VendorID'] == vendor_id]['total_amount'].m
ean()
    print(f"Mean total_amount for VendorID {vendor_id}: {mean_total_amoun
t:.2f}")

df.groupby(['VendorID']).mean(numeric_only=True)[['total_amount']]
```

```
Mean total_amount for VendorID 1: 16.30
Mean total_amount for VendorID 2: 16.32
```

Out[142]:

VendorID	total_amount
1	16.298119
2	16.320382

In [127]:
```
# Filter the data for credit card payments only
credit_card = df[df['payment_type']==1]

# Filter the data for passenger count only
credit_card['passenger_count'].value_counts()
```

Out[127]:
```
1    10977
2     2168
5      775
3      600
6      451
4      267
0       27
Name: passenger_count, dtype: int64
```

In [129]:
```
# Calculate the average tip amount for each passenger count (credit card payments only)

credit_card.groupby(['passenger_count']).mean(numeric_only=True)[['tip_amount']]
```

Out[129]:

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

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.

PACE: Execute

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

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?

The two most helpful variables for creating a predictive model are trip_distance and total_amount. Analyzing and performing inference on these variables should allow predicting trip fares based on the approximate distance travelled.

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.