Activity_ Course 5 Automatidata project lab

October 26, 2023

1 Automatidata project

Course 5 - Regression Analysis: Simplify complex data relationships

The data consulting firm Automatidata has recently hired you as the newest member of their data analytics team. Their newest client, the NYC Taxi and Limousine Commission (New York City TLC), wants the Automatidata team to build a multiple linear regression model to predict taxi fares using existing data that was collected over the course of a year. The team is getting closer to completing the project, having completed an initial plan of action, initial Python coding work, EDA, and A/B testing.

The Automatidata team has reviewed the results of the A/B testing. Now it's time to work on predicting the taxi fare amounts. You've impressed your Automatidata colleagues with your hard work and attention to detail. The data team believes that you are ready to build the regression model and update the client New York City TLC about your progress.

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

2 Course 5 End-of-course project: Build a multiple linear regression model

In this activity, you will build a multiple linear regression model. As you've learned, multiple linear regression helps you estimate the linear relationship between one continuous dependent variable and two or more independent variables. For data science professionals, this is a useful skill because it allows you to consider more than one variable against the variable you're measuring against. This opens the door for much more thorough and flexible analysis to be completed.

Completing this activity will help you practice planning out and buidling a multiple linear regression model based on a specific business need. The structure of this activity is designed to emulate the proposals you will likely be assigned in your career as a data professional. Completing this activity will help prepare you for those career moments.

The purpose of this project is to demostrate knowledge of EDA and a multiple linear regression model

The goal is to build a multiple linear regression model and evaluate the model *This activity has three parts:*

Part 1: EDA & Checking Model Assumptions * What are some purposes of EDA before constructing a multiple linear regression model?

Part 2: Model Building and evaluation * What resources do you find yourself using as you complete this stage?

Part 3: Interpreting Model Results

- What key insights emerged from your model(s)?
- What business recommendations do you propose based on the models built?

3 Build a multiple linear regression model

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 to reflect on the Plan stage.

4.1.1 Task 1. Imports and loading

Import the packages that you've learned are needed for building linear regression models.

[1]:
```
# Packages for numerics + dataframes
import pandas as pd
import numpy as np

# Packages for visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Packages for date conversions for calculating trip durations
from datetime import date, datetime, timedelta

# Packages for OLS, MLR, confusion matrix
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import sklearn.metrics as metrics
```

Note: `Pandas` is used to load the NYC TLC dataset. 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]:
```python
df0 = pd.read_csv("2017_Yellow_Taxi_Trip_Data.csv")
```

4.2 PACE: Analyze

In this stage, consider the following question where applicable to complete your code response:

- What are some purposes of EDA before constructing a multiple linear regression model?

4.2.1 Task 2a. Explore data with EDA

Analyze and discover data, looking for correlations, missing data, outliers, and duplicates.

Start with `.shape` and `.info()`.

[3]:
```python
df0.shape
```

[3]:
```
(22699, 18)
```

[4]:
```python
df0.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
```

Check for missing data and duplicates using `.isna()` and `.drop_duplicates()`.

[5]:
```python
df0.isna().any(axis= 1).sum()
```

[5]:
```
0
```

[6]:
```python
df0.duplicated().sum()
```

[6]:
```
0
```

Use `.describe()`.

[7]:
```python
df0.describe(include= [np.object]).T
```

[7]:

	count	unique	top	freq
tpep_pickup_datetime	22699	22687	07/03/2017 3:45:19 PM	2
tpep_dropoff_datetime	22699	22688	10/18/2017 8:07:45 PM	2
store_and_fwd_flag	22699	2	N	22600

[8]:
```python
df0.describe(include= [np.number], percentiles= [.5]).T
```

[8]:

	count	mean	std	min \
Unnamed: 0	22699.0	5.675849e+07	3.274493e+07	12127.0
VendorID	22699.0	1.556236e+00	4.968384e-01	1.0
passenger_count	22699.0	1.642319e+00	1.285231e+00	0.0
trip_distance	22699.0	2.913313e+00	3.653171e+00	0.0
RatecodeID	22699.0	1.043394e+00	7.083909e-01	1.0
PULocationID	22699.0	1.624124e+02	6.663337e+01	1.0
DOLocationID	22699.0	1.615280e+02	7.013969e+01	1.0
payment_type	22699.0	1.336887e+00	4.962111e-01	1.0
fare_amount	22699.0	1.302663e+01	1.324379e+01	-120.0
extra	22699.0	3.332746e-01	4.630966e-01	-1.0
mta_tax	22699.0	4.974448e-01	3.946499e-02	-0.5
tip_amount	22699.0	1.835781e+00	2.800626e+00	0.0
tolls_amount	22699.0	3.125415e-01	1.399212e+00	0.0
improvement_surcharge	22699.0	2.995506e-01	1.567274e-02	-0.3
total_amount	22699.0	1.631050e+01	1.609730e+01	-120.3

	50%	max
Unnamed: 0	56731504.00	1.134863e+08
VendorID	2.00	2.000000e+00
passenger_count	1.00	6.000000e+00
trip_distance	1.61	3.396000e+01
RatecodeID	1.00	9.900000e+01

```
PULocationID             162.00  2.650000e+02
DOLocationID             162.00  2.650000e+02
payment_type               1.00  4.000000e+00
fare_amount                9.50  9.999900e+02
extra                      0.00  4.500000e+00
mta_tax                    0.50  5.000000e-01
tip_amount                 1.35  2.000000e+02
tolls_amount               0.00  1.910000e+01
improvement_surcharge      0.30  3.000000e-01
total_amount              11.80  1.200290e+03
```

4.2.2 Task 2b. Convert pickup & dropoff columns to datetime

[9]:
```
df0[["tpep_pickup_datetime", "tpep_dropoff_datetime"]].dtypes
```

[9]:
```
tpep_pickup_datetime     object
tpep_dropoff_datetime    object
dtype: object
```

[10]:
```
df0["tpep_pickup_datetime"] = pd.to_datetime(df0["tpep_pickup_datetime"])
df0["tpep_dropoff_datetime"] = pd.to_datetime(df0["tpep_dropoff_datetime"])
```

[11]:
```
df0[["tpep_pickup_datetime", "tpep_dropoff_datetime"]].dtypes
```

[11]:
```
tpep_pickup_datetime     datetime64[ns]
tpep_dropoff_datetime    datetime64[ns]
dtype: object
```

4.2.3 Task 2c. Create duration column

Create a new column called `duration` that represents the total number of minutes that each taxi ride took.

[12]:
```
df0["trip_duration"] = (df0["tpep_dropoff_datetime"] -␣
↪df0["tpep_pickup_datetime"]) / np.timedelta64(1, "m")
df0["trip_duration"].describe()
```

[12]:
```
count    22699.000000
mean        17.013777
std         61.996482
min        -16.983333
25%          6.650000
50%         11.183333
75%         18.383333
max       1439.550000
```

```
Name: trip_duration, dtype: float64
```

4.2.4 Outliers

Call `df.info()` to inspect the columns and decide which ones to check for outliers.

[13]:
```python
round(df0.describe(include= [np.number], percentiles= [.5]).T, 2)
```

[13]:
```
                         count         mean          std       min  \
Unnamed: 0             22699.0  56758486.17  32744929.49  12127.00
VendorID               22699.0         1.56         0.50      1.00
passenger_count        22699.0         1.64         1.29      0.00
trip_distance          22699.0         2.91         3.65      0.00
RatecodeID             22699.0         1.04         0.71      1.00
PULocationID           22699.0       162.41        66.63      1.00
DOLocationID           22699.0       161.53        70.14      1.00
payment_type           22699.0         1.34         0.50      1.00
fare_amount            22699.0        13.03        13.24   -120.00
extra                  22699.0         0.33         0.46     -1.00
mta_tax                22699.0         0.50         0.04     -0.50
tip_amount             22699.0         1.84         2.80      0.00
tolls_amount           22699.0         0.31         1.40      0.00
improvement_surcharge  22699.0         0.30         0.02     -0.30
total_amount           22699.0        16.31        16.10   -120.30
trip_duration          22699.0        17.01        62.00    -16.98

                               50%           max
Unnamed: 0             56731504.00  1.134863e+08
VendorID                      2.00  2.000000e+00
passenger_count               1.00  6.000000e+00
trip_distance                 1.61  3.396000e+01
RatecodeID                    1.00  9.900000e+01
PULocationID                162.00  2.650000e+02
DOLocationID                162.00  2.650000e+02
payment_type                  1.00  4.000000e+00
fare_amount                   9.50  9.999900e+02
extra                         0.00  4.500000e+00
mta_tax                       0.50  5.000000e-01
tip_amount                    1.35  2.000000e+02
tolls_amount                  0.00  1.910000e+01
improvement_surcharge         0.30  3.000000e-01
total_amount                 11.80  1.200290e+03
trip_duration                11.18  1.439550e+03
```

From above description for numerical variables, we could observe outliers in: 1. trip_distance 2. RatecodeID 3. fare_amount 4. extra 5. mta_tax 6. tip_amount 7. improvement_surcharge 8. total_amount 9. trip_duration

4.2.5 Task 2d. Box plots

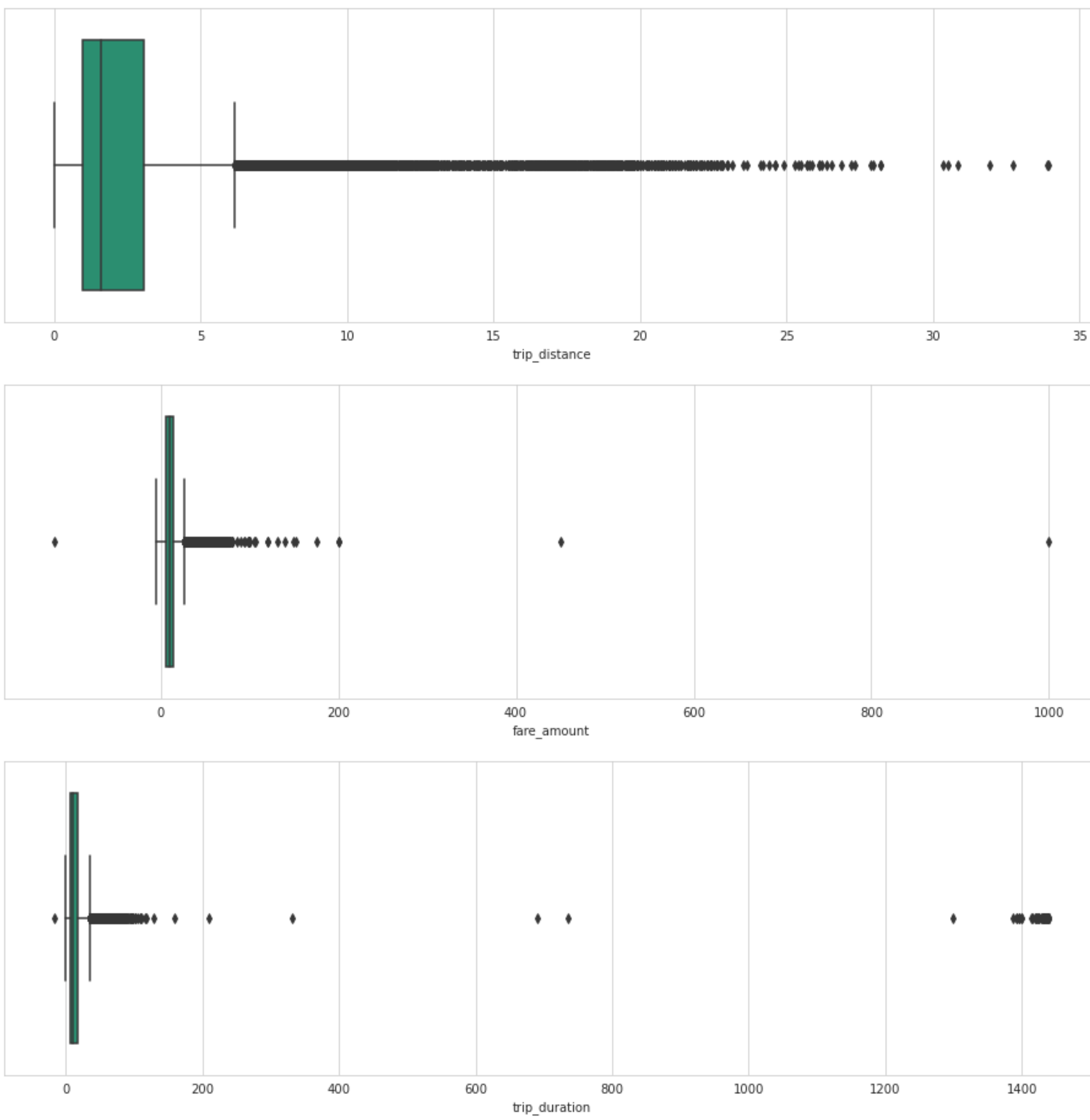
Plot a box plot for each feature: `trip_distance`, `fare_amount`, `duration`.

[14]:
```
sns.set_style("whitegrid")
sns.set_palette("Dark2")
```

[15]:
```
fig, ax= plt.subplots(3, 1, figsize= (16, 16), squeeze= True)

sns.boxplot(df0["trip_distance"], showfliers= True, ax= ax[0])
sns.boxplot(df0["fare_amount"], showfliers= True, ax= ax[1])
sns.boxplot(df0["trip_duration"], showfliers= True, ax= ax[2])

plt.show()
```



Questions: 1. Which variable(s) contains outliers?

2. Are the values in the `trip_distance` column unbelievable?

3. What about the lower end? Do distances, fares, and durations of 0 (or negative values) make sense?

1. All 3 variables contain extreme outliers.
2. Absolutely yes, it's unbelievable above the upper limit which is nearly 6 miles.
3. As well the negative values don't make any sense in this context.

4.2.6 Task 2e. Imputations

`trip_distance` outliers You know from the summary statistics that there are trip distances of 0. Are these reflective of erroneous data, or are they very short trips that get rounded down?

To check, sort the column values, eliminate duplicates, and inspect the least 10 values. Are they rounded values or precise values?

[16]:
```
# Are trip distances of 0 bad data or very short trips rounded down?
sorted(set(df0["trip_distance"]))[:5]
```

[16]:
```
[0.0, 0.01, 0.02, 0.03, 0.04]
```

The distances are captured with a high degree of precision. However, it might be possible for trips to have distances of zero if a passenger summoned a taxi and then changed their mind. Besides, are there enough zero values in the data to pose a problem?

Calculate the count of rides where the `trip_distance` is zero.

[17]:
```
df0["trip_distance"][df0["trip_distance"] == 0].count()
```

[17]:
```
148
```

[18]:
```
def impute_outliers(column_list, iqr_factor):
    '''
    Impute upper-limit values in specified columns based on their interquartile␣
↪range.

    Arguments:
        column_list: A list of columns to iterate over
        iqr_factor: A number representing x in the formula:
                    Q3 + (x * IQR). Used to determine maximum threshold,
                    beyond which a point is considered an outlier.

    The IQR is computed for each column in column_list and values exceeding
    the upper threshold for each column are imputed with the upper threshold␣
↪value.
```

```
    '''
    for column in column_list:
        # Reassign minimum to zero
        df0.loc[df0[column] < 0, column] = 0

        # Calculate upper threshold
        Q3 = df0[column].quantile(.75)
        Q1 = df0[column].quantile(.25)
        iqr = Q3 - Q1
        upper_limit = Q3 + (iqr_factor * iqr)
        print(column)
        print("Q3:", round(Q3, 3))
        print("Upper threshold:", round(upper_limit, 3))

        # Reassign values > threshold to threshold
        df0.loc[df0[column] > upper_limit, column] = upper_limit
```

fare_amount outliers

[19]:
```
round(df0["fare_amount"].describe(), 3)
```

[19]:
```
count    22699.000
mean        13.027
std         13.244
min       -120.000
25%          6.500
50%          9.500
75%         14.500
max        999.990
Name: fare_amount, dtype: float64
```

Question: What do you notice about the values in the `fare_amount` column?

We notice that the minimum values are negative, and the maximum values are graeter than 200, which are counter-intuitive.

[20]:
```
impute_outliers(["fare_amount"], 6)
```

```
fare_amount
Q3: 14.5
Upper threshold: 62.5
```

duration outliers

[21]:
```
# Call .describe() for duration outliers
round(df0["trip_duration"].describe(), 3)
```

[21]:
```
count    22699.000
mean        17.014
std         61.996
min        -16.983
25%          6.650
50%         11.183
75%         18.383
max       1439.550
Name: trip_duration, dtype: float64
```

The `duration` column has problematic values at both the lower and upper extremities.

- **Low values:** There should be no values that represent negative time. Impute all negative durations with `0`.
- **High values:** Impute high values the same way you imputed the high-end outliers for fares: `Q3 + (6 * IQR)`.

[22]:
```
impute_outliers(["trip_duration"], 6)
```

```
trip_duration
Q3: 18.383
Upper threshold: 88.783
```

4.2.7 Task 3a. Feature engineering

Create `mean_distance` column When deployed, the model will not know the duration of a trip until after the trip occurs, so you cannot train a model that uses this feature. However, you can use the statistics of trips you *do* know to generalize about ones you do not know.

In this step, create a column called `mean_distance` that captures the mean distance for each group of trips that share pickup and dropoff points.

For example, if your data were:

|Trip|Start|End|Distance| |–: |:—:|:-:| | | 1 | A | B | 1 | | 2 | C | D | 2 | | 3 | A | B |1.5 | | 4 | D | C | 3 |

The results should be:

```
A -> B: 1.25 miles
C -> D: 2 miles
D -> C: 3 miles
```

Notice that C -> D is not the same as D -> C. All trips that share a unique pair of start and end points get grouped and averaged.

Then, a new column `mean_distance` will be added where the value at each row is the average for all trips with those pickup and dropoff locations:

Trip	Start	End	Distance	mean_distance
1	A	B	1	1.25

Trip	Start	End	Distance	mean_distance
2	C	D	2	2
3	A	B	1.5	1.25
4	D	C	3	3

Begin by creating a helper column called `pickup_dropoff`, which contains the unique combination of pickup and dropoff location IDs for each row.

One way to do this is to convert the pickup and dropoff location IDs to strings and join them, separated by a space. The space is to ensure that, for example, a trip with pickup/dropoff points of 12 & 151 gets encoded differently than a trip with points 121 & 51.

So, the new column would look like this:

Trip	Start	End	pickup_dropoff
1	A	B	'A B'
2	C	D	'C D'
3	A	B	'A B'
4	D	C	'D C'

[23]:
```
# Create `pickup_dropoff` column
df0["pickup_dropoff"] = df0["PULocationID"].astype("str") + " >> " +␣
↪df0["DOLocationID"].astype("str")
df0["pickup_dropoff"].head(5)
```

[23]:
```
0    100 >> 231
1     186 >> 43
2    262 >> 236
3     188 >> 97
4      4 >> 112
Name: pickup_dropoff, dtype: object
```

[24]:
```
df0["pickup_dropoff"].describe()
```

[24]:
```
count          22699
unique          4172
top       264 >> 264
freq             277
Name: pickup_dropoff, dtype: object
```

Now, use a `groupby()` statement to group each row by the new `pickup_dropoff` column, compute the mean, and capture the values only in the `trip_distance` column. Assign the results to a variable named `grouped`.

[25]:
```
grouped_pickup_dropoff = round(df0.
↪groupby(["pickup_dropoff"])[["trip_distance"]].mean(), 2)
```

```python
grouped_pickup_dropoff.head(5)
```

[25]:
```
                 trip_distance
pickup_dropoff
1 >> 1                    2.43
10 >> 148                15.70
100 >> 1                 16.89
100 >> 100                0.25
100 >> 107                1.18
```

`grouped` is an object of the `DataFrame` class.

1. Convert it to a dictionary using the `to_dict()` method. Assign the results to a variable called `grouped_dict`. This will result in a dictionary with a key of `trip_distance` whose values are another dictionary. The inner dictionary's keys are pickup/dropoff points and its values are mean distances. This is the information you want.

Example:
`grouped_dict = {'trip_distance': {'A B': 1.25, 'C D': 2, 'D C': 3}`

2. Reassign the `grouped_dict` dictionary so it contains only the inner dictionary. In other words, get rid of `trip_distance` as a key, so:

Example:
`grouped_dict = {'A B': 1.25, 'C D': 2, 'D C': 3}`

[26]:
```python
# 1. Convert `grouped` to a dictionary
grouped_dict = grouped_pickup_dropoff.to_dict()

# 2. Reassign to only contain the inner dictionary
grouped_dict = grouped_dict["trip_distance"]
grouped_dict
```

[26]:
```
{'1 >> 1': 2.43,
 '10 >> 148': 15.7,
 '100 >> 1': 16.89,
 '100 >> 100': 0.25,
 '100 >> 107': 1.18,
 '100 >> 113': 2.02,
 '100 >> 114': 1.94,
 '100 >> 12': 4.55,
 '100 >> 125': 2.84,
 '100 >> 13': 4.2,
 '100 >> 132': 17.22,
 '100 >> 137': 1.3,
 '100 >> 138': 10.43,
 '100 >> 140': 2.75,
 '100 >> 141': 2.11,
 '100 >> 142': 1.7,
```

```
'100 >> 143': 1.58,
'100 >> 144': 3.01,
'100 >> 148': 4.11,
'100 >> 151': 3.67,
'100 >> 152': 4.9,
'100 >> 158': 1.94,
'100 >> 161': 0.98,
'100 >> 162': 1.22,
'100 >> 163': 1.27,
'100 >> 164': 0.84,
'100 >> 166': 5.2,
'100 >> 170': 0.85,
'100 >> 177': 12.0,
'100 >> 181': 9.34,
'100 >> 186': 0.64,
'100 >> 193': 4.39,
'100 >> 198': 9.01,
'100 >> 202': 5.3,
'100 >> 209': 4.43,
'100 >> 211': 2.48,
'100 >> 224': 1.95,
'100 >> 225': 7.5,
'100 >> 229': 1.78,
'100 >> 230': 0.73,
'100 >> 231': 3.52,
'100 >> 232': 3.84,
'100 >> 233': 1.25,
'100 >> 234': 1.25,
'100 >> 236': 3.34,
'100 >> 237': 2.56,
'100 >> 238': 3.36,
'100 >> 239': 2.33,
'100 >> 243': 8.77,
'100 >> 244': 7.9,
'100 >> 246': 1.17,
'100 >> 249': 1.81,
'100 >> 25': 7.36,
'100 >> 255': 6.35,
'100 >> 256': 5.86,
'100 >> 261': 3.81,
'100 >> 262': 3.82,
'100 >> 263': 3.4,
'100 >> 39': 22.6,
'100 >> 4': 2.7,
'100 >> 40': 7.23,
'100 >> 41': 4.6,
'100 >> 42': 6.78,
```

'100 >> 43': 2.03,
'100 >> 45': 3.63,
'100 >> 48': 0.85,
'100 >> 49': 7.35,
'100 >> 50': 1.18,
'100 >> 66': 4.7,
'100 >> 68': 0.99,
'100 >> 7': 4.9,
'100 >> 74': 4.53,
'100 >> 75': 4.03,
'100 >> 79': 2.61,
'100 >> 87': 5.03,
'100 >> 88': 5.5,
'100 >> 90': 1.12,
'100 >> 95': 9.0,
'106 >> 106': 0.02,
'106 >> 181': 1.1,
'106 >> 228': 1.24,
'106 >> 231': 3.8,
'106 >> 40': 0.8,
'107 >> 1': 15.55,
'107 >> 100': 1.44,
'107 >> 107': 0.49,
'107 >> 113': 0.9,
'107 >> 114': 1.21,
'107 >> 125': 1.8,
'107 >> 127': 11.57,
'107 >> 13': 3.87,
'107 >> 130': 12.43,
'107 >> 132': 16.76,
'107 >> 137': 0.68,
'107 >> 138': 10.38,
'107 >> 140': 2.8,
'107 >> 141': 2.98,
'107 >> 142': 3.23,
'107 >> 143': 4.3,
'107 >> 144': 1.62,
'107 >> 145': 3.53,
'107 >> 146': 4.3,
'107 >> 147': 8.11,
'107 >> 148': 1.73,
'107 >> 152': 6.62,
'107 >> 158': 1.78,
'107 >> 161': 1.71,
'107 >> 162': 1.57,
'107 >> 163': 2.48,
'107 >> 164': 0.75,

'107 >> 170': 1.0,
'107 >> 186': 1.43,
'107 >> 196': 7.89,
'107 >> 202': 5.86,
'107 >> 209': 3.5,
'107 >> 21': 11.5,
'107 >> 211': 1.73,
'107 >> 223': 5.7,
'107 >> 224': 0.85,
'107 >> 229': 1.91,
'107 >> 23': 17.72,
'107 >> 230': 2.11,
'107 >> 231': 3.28,
'107 >> 232': 2.14,
'107 >> 233': 1.35,
'107 >> 234': 0.68,
'107 >> 236': 3.16,
'107 >> 237': 2.25,
'107 >> 238': 4.99,
'107 >> 244': 8.84,
'107 >> 246': 1.58,
'107 >> 249': 1.4,
'107 >> 25': 4.5,
'107 >> 256': 3.87,
'107 >> 257': 9.0,
'107 >> 26': 10.33,
'107 >> 261': 2.19,
'107 >> 262': 3.57,
'107 >> 263': 3.67,
'107 >> 265': 4.5,
'107 >> 36': 5.6,
'107 >> 37': 4.9,
'107 >> 4': 1.16,
'107 >> 41': 5.98,
'107 >> 42': 7.0,
'107 >> 43': 2.8,
'107 >> 45': 2.14,
'107 >> 48': 2.55,
'107 >> 49': 5.0,
'107 >> 66': 3.58,
'107 >> 68': 1.36,
'107 >> 7': 5.95,
'107 >> 74': 5.08,
'107 >> 75': 4.93,
'107 >> 79': 0.99,
'107 >> 80': 4.7,
'107 >> 82': 7.32,

'107 >> 87': 3.5,
'107 >> 88': 3.5,
'107 >> 89': 6.97,
'107 >> 90': 0.92,
'112 >> 112': 0.7,
'112 >> 223': 4.05,
'112 >> 263': 8.0,
'112 >> 49': 3.1,
'112 >> 66': 4.57,
'112 >> 80': 0.42,
'113 >> 100': 1.98,
'113 >> 106': 4.69,
'113 >> 107': 1.05,
'113 >> 112': 4.5,
'113 >> 113': 0.83,
'113 >> 114': 0.76,
'113 >> 116': 8.55,
'113 >> 125': 1.22,
'113 >> 13': 2.21,
'113 >> 137': 1.33,
'113 >> 138': 10.4,
'113 >> 14': 15.62,
'113 >> 140': 3.3,
'113 >> 141': 3.63,
'113 >> 142': 3.6,
'113 >> 143': 4.39,
'113 >> 144': 1.13,
'113 >> 146': 5.2,
'113 >> 148': 1.19,
'113 >> 152': 8.0,
'113 >> 158': 0.95,
'113 >> 161': 2.24,
'113 >> 162': 2.21,
'113 >> 163': 2.52,
'113 >> 164': 1.48,
'113 >> 17': 4.38,
'113 >> 170': 1.57,
'113 >> 181': 5.55,
'113 >> 186': 1.41,
'113 >> 209': 2.89,
'113 >> 211': 1.02,
'113 >> 22': 12.0,
'113 >> 224': 1.46,
'113 >> 230': 2.11,
'113 >> 231': 1.74,
'113 >> 232': 2.14,
'113 >> 233': 2.15,

'113 >> 234': 0.86,
'113 >> 236': 3.99,
'113 >> 237': 3.41,
'113 >> 238': 5.32,
'113 >> 239': 5.05,
'113 >> 243': 12.4,
'113 >> 244': 9.4,
'113 >> 246': 2.21,
'113 >> 249': 0.77,
'113 >> 255': 3.7,
'113 >> 256': 3.3,
'113 >> 261': 2.0,
'113 >> 262': 4.4,
'113 >> 263': 4.0,
'113 >> 264': 0.0,
'113 >> 33': 3.83,
'113 >> 36': 6.3,
'113 >> 4': 1.09,
'113 >> 41': 9.2,
'113 >> 42': 8.34,
'113 >> 45': 2.02,
'113 >> 48': 2.3,
'113 >> 50': 2.86,
'113 >> 66': 3.2,
'113 >> 68': 1.2,
'113 >> 79': 0.78,
'113 >> 80': 5.31,
'113 >> 87': 3.14,
'113 >> 88': 2.6,
'113 >> 90': 0.85,
'113 >> 94': 12.5,
'114 >> 100': 2.4,
'114 >> 107': 1.3,
'114 >> 112': 5.13,
'114 >> 113': 0.61,
'114 >> 114': 0.55,
'114 >> 116': 9.1,
'114 >> 125': 0.75,
'114 >> 13': 1.9,
'114 >> 137': 1.84,
'114 >> 14': 11.11,
'114 >> 140': 4.1,
'114 >> 141': 3.93,
'114 >> 142': 3.88,
'114 >> 143': 4.74,
'114 >> 144': 0.85,
'114 >> 145': 6.53,

'114 >> 148': 0.91,
'114 >> 151': 7.3,
'114 >> 158': 1.36,
'114 >> 161': 2.8,
'114 >> 162': 2.71,
'114 >> 163': 3.5,
'114 >> 164': 1.71,
'114 >> 166': 7.55,
'114 >> 169': 11.6,
'114 >> 170': 2.08,
'114 >> 181': 3.77,
'114 >> 186': 1.68,
'114 >> 190': 4.7,
'114 >> 209': 1.66,
'114 >> 211': 0.45,
'114 >> 217': 2.4,
'114 >> 223': 7.6,
'114 >> 224': 5.23,
'114 >> 225': 4.41,
'114 >> 229': 3.0,
'114 >> 230': 2.88,
'114 >> 231': 1.2,
'114 >> 232': 1.34,
'114 >> 233': 2.34,
'114 >> 234': 1.35,
'114 >> 236': 4.56,
'114 >> 237': 3.72,
'114 >> 238': 5.89,
'114 >> 239': 4.99,
'114 >> 24': 6.67,
'114 >> 243': 11.23,
'114 >> 244': 8.9,
'114 >> 246': 2.01,
'114 >> 249': 0.85,
'114 >> 255': 3.69,
'114 >> 257': 4.96,
'114 >> 260': 7.08,
'114 >> 261': 1.73,
'114 >> 262': 5.5,
'114 >> 263': 4.75,
'114 >> 36': 5.98,
'114 >> 4': 1.35,
'114 >> 43': 3.6,
'114 >> 45': 1.09,
'114 >> 48': 3.4,
'114 >> 49': 3.96,
'114 >> 50': 3.32,

'114 >> 62': 5.7,
'114 >> 65': 2.8,
'114 >> 66': 3.3,
'114 >> 68': 1.7,
'114 >> 69': 10.05,
'114 >> 7': 6.4,
'114 >> 79': 1.03,
'114 >> 87': 2.04,
'114 >> 90': 1.32,
'114 >> 97': 3.7,
'116 >> 116': 0.48,
'116 >> 119': 2.9,
'116 >> 132': 19.05,
'116 >> 159': 1.64,
'116 >> 162': 6.1,
'116 >> 166': 1.33,
'116 >> 186': 6.42,
'116 >> 230': 6.38,
'116 >> 238': 3.3,
'116 >> 239': 4.52,
'116 >> 244': 1.08,
'116 >> 41': 1.72,
'116 >> 42': 1.58,
'116 >> 68': 6.31,
'116 >> 74': 2.08,
'116 >> 75': 4.23,
'116 >> 79': 9.3,
'118 >> 118': 1.43,
'12 >> 100': 4.0,
'12 >> 13': 0.9,
'12 >> 142': 5.56,
'12 >> 144': 2.08,
'12 >> 151': 8.3,
'12 >> 163': 5.5,
'12 >> 164': 5.38,
'12 >> 170': 4.9,
'12 >> 48': 4.67,
'123 >> 123': 0.93,
'125 >> 1': 14.67,
'125 >> 100': 2.11,
'125 >> 106': 5.0,
'125 >> 107': 2.13,
'125 >> 113': 0.7,
'125 >> 114': 0.82,
'125 >> 129': 8.13,
'125 >> 13': 1.3,
'125 >> 132': 19.88,

'125 >> 137': 2.68,
'125 >> 138': 10.46,
'125 >> 140': 4.96,
'125 >> 141': 6.5,
'125 >> 142': 4.8,
'125 >> 144': 0.66,
'125 >> 148': 1.39,
'125 >> 151': 5.43,
'125 >> 158': 0.74,
'125 >> 161': 2.78,
'125 >> 162': 3.24,
'125 >> 163': 3.4,
'125 >> 164': 2.62,
'125 >> 170': 4.03,
'125 >> 186': 1.87,
'125 >> 188': 5.88,
'125 >> 211': 0.66,
'125 >> 227': 9.4,
'125 >> 230': 2.71,
'125 >> 231': 1.01,
'125 >> 234': 1.72,
'125 >> 236': 4.8,
'125 >> 237': 3.47,
'125 >> 238': 6.09,
'125 >> 239': 5.05,
'125 >> 244': 9.07,
'125 >> 246': 2.26,
'125 >> 249': 0.68,
'125 >> 255': 4.05,
'125 >> 256': 3.1,
'125 >> 261': 2.01,
'125 >> 263': 5.95,
'125 >> 42': 8.16,
'125 >> 48': 2.74,
'125 >> 49': 4.21,
'125 >> 68': 1.93,
'125 >> 75': 7.2,
'125 >> 79': 1.56,
'125 >> 87': 2.09,
'125 >> 88': 2.5,
'125 >> 90': 1.44,
'125 >> 97': 4.84,
'127 >> 243': 1.92,
'128 >> 238': 7.3,
'129 >> 129': 0.81,
'129 >> 160': 6.3,
'129 >> 164': 1.96,

'129 >> 173': 2.1,
'129 >> 207': 1.2,
'129 >> 70': 1.69,
'13 >> 100': 3.98,
'13 >> 107': 4.66,
'13 >> 113': 2.75,
'13 >> 114': 2.1,
'13 >> 12': 0.9,
'13 >> 125': 0.93,
'13 >> 13': 0.52,
'13 >> 132': 24.5,
'13 >> 137': 5.04,
'13 >> 138': 15.22,
'13 >> 14': 7.1,
'13 >> 140': 6.96,
'13 >> 141': 7.44,
'13 >> 142': 5.1,
'13 >> 143': 5.15,
'13 >> 144': 2.15,
'13 >> 148': 3.31,
'13 >> 158': 2.35,
'13 >> 161': 5.9,
'13 >> 162': 6.37,
'13 >> 163': 5.17,
'13 >> 164': 6.02,
'13 >> 166': 7.46,
'13 >> 17': 5.6,
'13 >> 170': 6.17,
'13 >> 181': 3.9,
'13 >> 186': 3.77,
'13 >> 209': 1.8,
'13 >> 211': 1.73,
'13 >> 224': 4.8,
'13 >> 225': 7.51,
'13 >> 226': 8.3,
'13 >> 229': 6.34,
'13 >> 230': 4.56,
'13 >> 231': 0.96,
'13 >> 232': 3.13,
'13 >> 233': 6.2,
'13 >> 234': 3.82,
'13 >> 236': 8.33,
'13 >> 237': 7.14,
'13 >> 238': 6.7,
'13 >> 239': 5.72,
'13 >> 244': 10.58,
'13 >> 246': 2.79,

```
'13 >> 249': 2.14,
'13 >> 25': 3.0,
'13 >> 255': 5.53,
'13 >> 261': 0.78,
'13 >> 262': 8.11,
'13 >> 263': 8.07,
'13 >> 33': 3.99,
'13 >> 40': 2.8,
'13 >> 45': 2.4,
'13 >> 48': 4.15,
'13 >> 49': 5.43,
'13 >> 50': 4.1,
'13 >> 54': 4.62,
'13 >> 55': 13.26,
'13 >> 65': 3.96,
'13 >> 68': 2.7,
'13 >> 74': 10.36,
'13 >> 79': 3.96,
'13 >> 85': 10.99,
'13 >> 87': 1.21,
'13 >> 88': 0.9,
'13 >> 90': 2.7,
'13 >> 91': 7.86,
'130 >> 230': 12.8,
'130 >> 64': 6.03,
'131 >> 9': 2.1,
'132 >> 10': 3.75,
'132 >> 100': 17.6,
'132 >> 102': 7.7,
'132 >> 106': 20.2,
'132 >> 107': 17.56,
'132 >> 11': 17.94,
'132 >> 112': 15.81,
'132 >> 113': 18.3,
'132 >> 114': 21.73,
'132 >> 117': 12.2,
'132 >> 121': 10.47,
'132 >> 123': 15.65,
'132 >> 124': 5.67,
'132 >> 125': 18.74,
'132 >> 13': 20.86,
'132 >> 130': 6.72,
'132 >> 132': 2.26,
'132 >> 134': 6.58,
'132 >> 137': 16.72,
'132 >> 138': 11.69,
'132 >> 14': 20.07,
```

'132 >> 140': 19.29,
'132 >> 141': 19.14,
'132 >> 142': 20.41,
'132 >> 143': 10.9,
'132 >> 144': 18.54,
'132 >> 145': 15.84,
'132 >> 148': 17.99,
'132 >> 149': 14.32,
'132 >> 15': 14.4,
'132 >> 150': 14.85,
'132 >> 151': 19.83,
'132 >> 152': 19.1,
'132 >> 158': 22.7,
'132 >> 161': 18.6,
'132 >> 162': 17.08,
'132 >> 163': 19.23,
'132 >> 164': 18.76,
'132 >> 166': 18.6,
'132 >> 17': 10.4,
'132 >> 170': 17.2,
'132 >> 174': 21.17,
'132 >> 177': 9.2,
'132 >> 179': 15.27,
'132 >> 181': 17.36,
'132 >> 186': 18.38,
'132 >> 188': 12.15,
'132 >> 189': 12.2,
'132 >> 19': 10.5,
'132 >> 195': 26.54,
'132 >> 196': 9.35,
'132 >> 197': 6.59,
'132 >> 198': 9.9,
'132 >> 201': 12.94,
'132 >> 205': 6.0,
'132 >> 209': 21.2,
'132 >> 211': 18.91,
'132 >> 212': 16.85,
'132 >> 213': 15.2,
'132 >> 215': 4.9,
'132 >> 216': 4.49,
'132 >> 218': 4.5,
'132 >> 22': 17.9,
'132 >> 220': 30.5,
'132 >> 222': 8.21,
'132 >> 223': 13.25,
'132 >> 224': 17.59,
'132 >> 225': 11.8,

'132 >> 226': 14.96,
'132 >> 228': 23.88,
'132 >> 229': 18.49,
'132 >> 23': 30.83,
'132 >> 230': 18.57,
'132 >> 231': 20.46,
'132 >> 232': 18.3,
'132 >> 233': 17.86,
'132 >> 234': 17.65,
'132 >> 236': 19.49,
'132 >> 237': 19.54,
'132 >> 238': 20.84,
'132 >> 239': 20.9,
'132 >> 24': 19.14,
'132 >> 241': 20.5,
'132 >> 243': 22.1,
'132 >> 244': 19.9,
'132 >> 246': 18.52,
'132 >> 248': 17.22,
'132 >> 249': 18.73,
'132 >> 25': 14.81,
'132 >> 252': 11.3,
'132 >> 255': 16.47,
'132 >> 256': 17.22,
'132 >> 257': 19.81,
'132 >> 259': 20.96,
'132 >> 26': 13.92,
'132 >> 261': 22.12,
'132 >> 262': 19.16,
'132 >> 263': 19.21,
'132 >> 264': 0.0,
'132 >> 265': 14.89,
'132 >> 28': 6.31,
'132 >> 33': 18.68,
'132 >> 36': 15.89,
'132 >> 37': 14.9,
'132 >> 38': 7.3,
'132 >> 39': 9.91,
'132 >> 4': 18.59,
'132 >> 40': 14.1,
'132 >> 42': 17.95,
'132 >> 43': 18.74,
'132 >> 48': 18.76,
'132 >> 49': 11.92,
'132 >> 50': 18.74,
'132 >> 51': 19.06,
'132 >> 52': 26.86,

```
'132 >> 54': 27.2,
'132 >> 55': 17.3,
'132 >> 61': 10.69,
'132 >> 62': 13.23,
'132 >> 64': 13.6,
'132 >> 65': 15.69,
'132 >> 66': 19.3,
'132 >> 68': 18.8,
'132 >> 7': 14.78,
'132 >> 70': 11.3,
'132 >> 71': 11.34,
'132 >> 72': 10.19,
'132 >> 74': 17.25,
'132 >> 76': 9.26,
'132 >> 77': 9.0,
'132 >> 79': 19.43,
'132 >> 80': 15.61,
'132 >> 82': 10.34,
'132 >> 83': 11.15,
'132 >> 85': 13.45,
'132 >> 86': 7.8,
'132 >> 87': 19.96,
'132 >> 88': 20.6,
'132 >> 89': 14.71,
'132 >> 9': 16.51,
'132 >> 90': 18.67,
'132 >> 91': 13.84,
'132 >> 92': 10.52,
'132 >> 93': 10.26,
'132 >> 95': 8.1,
'132 >> 97': 16.35,
'133 >> 133': 4.43,
'134 >> 197': 2.2,
'135 >> 75': 12.85,
'137 >> 100': 1.46,
'137 >> 107': 0.68,
'137 >> 112': 5.1,
'137 >> 113': 1.34,
'137 >> 114': 1.76,
'137 >> 125': 2.33,
'137 >> 13': 5.48,
'137 >> 132': 22.26,
'137 >> 135': 10.36,
'137 >> 137': 0.46,
'137 >> 138': 8.4,
'137 >> 14': 12.34,
'137 >> 140': 2.19,
```

'137 >> 141': 1.83,
'137 >> 142': 3.08,
'137 >> 145': 2.7,
'137 >> 148': 1.4,
'137 >> 158': 2.66,
'137 >> 161': 1.47,
'137 >> 162': 1.17,
'137 >> 163': 1.99,
'137 >> 164': 0.68,
'137 >> 170': 0.72,
'137 >> 181': 7.65,
'137 >> 186': 1.0,
'137 >> 209': 4.09,
'137 >> 220': 12.3,
'137 >> 223': 7.03,
'137 >> 224': 0.9,
'137 >> 229': 1.26,
'137 >> 230': 1.52,
'137 >> 231': 3.32,
'137 >> 232': 2.6,
'137 >> 233': 0.89,
'137 >> 234': 1.03,
'137 >> 236': 3.0,
'137 >> 237': 2.21,
'137 >> 238': 5.4,
'137 >> 239': 4.43,
'137 >> 243': 9.69,
'137 >> 246': 2.11,
'137 >> 249': 2.16,
'137 >> 255': 4.35,
'137 >> 261': 5.49,
'137 >> 262': 3.0,
'137 >> 263': 3.53,
'137 >> 4': 1.62,
'137 >> 41': 5.12,
'137 >> 42': 6.91,
'137 >> 43': 2.9,
'137 >> 45': 2.36,
'137 >> 48': 2.15,
'137 >> 50': 2.7,
'137 >> 61': 8.13,
'137 >> 68': 1.67,
'137 >> 7': 4.2,
'137 >> 74': 4.68,
'137 >> 79': 1.31,
'137 >> 82': 5.8,
'137 >> 87': 4.34,

'137 >> 88': 4.45,
'137 >> 90': 1.3,
'138 >> 1': 32.72,
'138 >> 100': 9.76,
'138 >> 106': 11.0,
'138 >> 107': 9.46,
'138 >> 112': 7.25,
'138 >> 113': 11.09,
'138 >> 114': 11.45,
'138 >> 116': 8.02,
'138 >> 121': 6.7,
'138 >> 125': 14.57,
'138 >> 127': 10.16,
'138 >> 129': 4.01,
'138 >> 13': 14.41,
'138 >> 130': 7.21,
'138 >> 132': 12.58,
'138 >> 134': 6.86,
'138 >> 137': 8.75,
'138 >> 138': 0.95,
'138 >> 14': 16.18,
'138 >> 140': 8.88,
'138 >> 141': 9.37,
'138 >> 142': 10.85,
'138 >> 143': 10.47,
'138 >> 144': 12.34,
'138 >> 145': 7.72,
'138 >> 146': 4.13,
'138 >> 148': 11.54,
'138 >> 15': 8.1,
'138 >> 151': 9.15,
'138 >> 152': 8.86,
'138 >> 158': 13.9,
'138 >> 160': 6.68,
'138 >> 161': 10.13,
'138 >> 162': 9.67,
'138 >> 163': 10.43,
'138 >> 164': 9.65,
'138 >> 166': 8.26,
'138 >> 17': 9.24,
'138 >> 170': 8.98,
'138 >> 171': 7.33,
'138 >> 174': 14.1,
'138 >> 175': 9.3,
'138 >> 177': 10.95,
'138 >> 178': 18.23,
'138 >> 179': 3.93,

'138 >> 180': 11.2,
'138 >> 181': 11.78,
'138 >> 182': 10.29,
'138 >> 186': 11.03,
'138 >> 188': 13.36,
'138 >> 189': 13.43,
'138 >> 192': 4.66,
'138 >> 196': 5.61,
'138 >> 197': 7.5,
'138 >> 198': 9.96,
'138 >> 200': 12.3,
'138 >> 209': 12.95,
'138 >> 210': 20.5,
'138 >> 211': 12.8,
'138 >> 220': 13.39,
'138 >> 223': 3.05,
'138 >> 224': 9.87,
'138 >> 225': 9.82,
'138 >> 226': 4.58,
'138 >> 229': 10.01,
'138 >> 230': 10.6,
'138 >> 231': 13.14,
'138 >> 232': 11.48,
'138 >> 233': 8.78,
'138 >> 234': 10.53,
'138 >> 236': 8.83,
'138 >> 237': 9.47,
'138 >> 238': 9.2,
'138 >> 239': 10.17,
'138 >> 243': 10.42,
'138 >> 244': 10.09,
'138 >> 246': 10.52,
'138 >> 249': 11.16,
'138 >> 25': 10.89,
'138 >> 252': 5.23,
'138 >> 255': 7.43,
'138 >> 256': 8.68,
'138 >> 257': 16.4,
'138 >> 260': 4.27,
'138 >> 261': 16.1,
'138 >> 262': 8.06,
'138 >> 263': 8.69,
'138 >> 265': 20.55,
'138 >> 29': 21.65,
'138 >> 33': 11.03,
'138 >> 36': 7.73,
'138 >> 37': 8.56,

'138 >> 4': 10.49,
'138 >> 41': 8.06,
'138 >> 42': 7.01,
'138 >> 43': 10.44,
'138 >> 48': 10.38,
'138 >> 49': 9.82,
'138 >> 50': 10.78,
'138 >> 51': 13.8,
'138 >> 52': 12.3,
'138 >> 53': 5.3,
'138 >> 56': 4.1,
'138 >> 61': 13.31,
'138 >> 62': 10.11,
'138 >> 64': 12.36,
'138 >> 65': 9.93,
'138 >> 66': 10.6,
'138 >> 68': 11.48,
'138 >> 69': 7.5,
'138 >> 7': 3.61,
'138 >> 70': 1.43,
'138 >> 74': 6.79,
'138 >> 75': 8.15,
'138 >> 79': 10.97,
'138 >> 80': 6.99,
'138 >> 81': 14.47,
'138 >> 82': 3.52,
'138 >> 83': 3.34,
'138 >> 87': 13.81,
'138 >> 88': 15.39,
'138 >> 89': 15.11,
'138 >> 90': 10.94,
'138 >> 92': 3.65,
'138 >> 93': 4.29,
'138 >> 95': 5.08,
'138 >> 97': 11.11,
'138 >> 98': 8.7,
'14 >> 14': 0.38,
'140 >> 107': 3.12,
'140 >> 113': 4.2,
'140 >> 125': 8.28,
'140 >> 13': 7.64,
'140 >> 132': 18.7,
'140 >> 135': 14.48,
'140 >> 137': 2.58,
'140 >> 138': 9.3,
'140 >> 140': 0.6,
'140 >> 141': 0.75,

'140 >> 142': 1.78,
'140 >> 143': 2.32,
'140 >> 151': 2.83,
'140 >> 161': 1.84,
'140 >> 162': 1.5,
'140 >> 163': 1.6,
'140 >> 164': 2.47,
'140 >> 166': 4.2,
'140 >> 170': 2.12,
'140 >> 179': 4.22,
'140 >> 186': 3.32,
'140 >> 193': 4.15,
'140 >> 209': 6.17,
'140 >> 211': 5.53,
'140 >> 223': 6.4,
'140 >> 224': 3.1,
'140 >> 226': 3.62,
'140 >> 229': 1.11,
'140 >> 230': 2.41,
'140 >> 231': 7.45,
'140 >> 232': 5.75,
'140 >> 233': 1.74,
'140 >> 234': 3.5,
'140 >> 236': 1.22,
'140 >> 237': 0.97,
'140 >> 238': 2.2,
'140 >> 239': 2.35,
'140 >> 24': 4.36,
'140 >> 243': 6.84,
'140 >> 244': 7.56,
'140 >> 246': 4.95,
'140 >> 249': 5.07,
'140 >> 260': 4.82,
'140 >> 262': 0.87,
'140 >> 263': 0.92,
'140 >> 4': 3.82,
'140 >> 43': 1.7,
'140 >> 45': 5.0,
'140 >> 48': 2.67,
'140 >> 50': 3.0,
'140 >> 52': 7.66,
'140 >> 65': 7.12,
'140 >> 66': 7.5,
'140 >> 68': 4.1,
'140 >> 7': 4.96,
'140 >> 74': 2.97,
'140 >> 75': 1.81,

```
'140 >> 79': 4.2,
'140 >> 83': 5.7,
'140 >> 85': 11.32,
'140 >> 87': 5.89,
'140 >> 88': 6.11,
'140 >> 90': 3.68,
'140 >> 95': 7.6,
'140 >> 97': 9.0,
'141 >> 100': 2.51,
'141 >> 107': 2.6,
'141 >> 112': 4.4,
'141 >> 113': 3.75,
'141 >> 114': 3.9,
'141 >> 116': 6.41,
'141 >> 13': 7.11,
'141 >> 130': 13.8,
'141 >> 132': 19.41,
'141 >> 133': 11.54,
'141 >> 137': 2.31,
'141 >> 138': 9.28,
'141 >> 140': 0.93,
'141 >> 141': 0.82,
'141 >> 142': 1.71,
'141 >> 143': 1.99,
'141 >> 145': 2.9,
'141 >> 148': 5.65,
'141 >> 151': 3.12,
'141 >> 158': 4.8,
'141 >> 161': 1.6,
'141 >> 162': 1.14,
'141 >> 163': 1.21,
'141 >> 164': 2.23,
'141 >> 166': 4.17,
'141 >> 170': 1.79,
'141 >> 173': 6.45,
'141 >> 178': 14.0,
'141 >> 186': 2.74,
'141 >> 193': 2.72,
'141 >> 196': 7.63,
'141 >> 209': 6.66,
'141 >> 211': 5.2,
'141 >> 220': 10.23,
'141 >> 224': 3.0,
'141 >> 226': 2.48,
'141 >> 229': 0.94,
'141 >> 230': 1.86,
'141 >> 231': 7.25,
```

'141 >> 233': 1.33,
'141 >> 234': 2.98,
'141 >> 236': 1.14,
'141 >> 237': 0.61,
'141 >> 238': 2.38,
'141 >> 239': 2.03,
'141 >> 24': 3.9,
'141 >> 243': 7.63,
'141 >> 244': 7.68,
'141 >> 246': 4.55,
'141 >> 249': 5.66,
'141 >> 255': 5.13,
'141 >> 261': 6.88,
'141 >> 262': 0.82,
'141 >> 263': 0.9,
'141 >> 4': 4.67,
'141 >> 42': 4.05,
'141 >> 43': 1.23,
'141 >> 48': 2.53,
'141 >> 50': 2.68,
'141 >> 65': 8.0,
'141 >> 68': 3.2,
'141 >> 7': 3.53,
'141 >> 74': 3.14,
'141 >> 75': 1.9,
'141 >> 79': 3.95,
'141 >> 80': 5.56,
'141 >> 88': 7.26,
'141 >> 90': 4.05,
'142 >> 100': 1.62,
'142 >> 107': 3.22,
'142 >> 113': 3.2,
'142 >> 114': 3.74,
'142 >> 116': 4.56,
'142 >> 125': 3.99,
'142 >> 127': 8.96,
'142 >> 129': 5.68,
'142 >> 13': 5.06,
'142 >> 132': 20.77,
'142 >> 137': 2.98,
'142 >> 138': 9.13,
'142 >> 140': 2.29,
'142 >> 141': 1.7,
'142 >> 142': 0.63,
'142 >> 143': 0.84,
'142 >> 144': 4.48,
'142 >> 145': 3.6,

```
'142 >> 148': 7.87,
'142 >> 151': 2.03,
'142 >> 158': 2.7,
'142 >> 161': 1.43,
'142 >> 162': 1.68,
'142 >> 163': 0.83,
'142 >> 164': 2.38,
'142 >> 166': 2.69,
'142 >> 17': 8.27,
'142 >> 170': 2.32,
'142 >> 174': 12.6,
'142 >> 181': 8.14,
'142 >> 186': 1.86,
'142 >> 209': 7.3,
'142 >> 211': 5.0,
'142 >> 220': 9.0,
'142 >> 223': 5.8,
'142 >> 224': 3.88,
'142 >> 225': 8.8,
'142 >> 229': 1.63,
'142 >> 230': 1.05,
'142 >> 231': 4.84,
'142 >> 233': 2.29,
'142 >> 234': 2.92,
'142 >> 236': 2.03,
'142 >> 237': 1.36,
'142 >> 238': 1.45,
'142 >> 239': 1.0,
'142 >> 24': 2.16,
'142 >> 243': 7.75,
'142 >> 244': 6.06,
'142 >> 246': 2.08,
'142 >> 249': 2.98,
'142 >> 261': 6.45,
'142 >> 262': 2.69,
'142 >> 263': 2.29,
'142 >> 264': 0.4,
'142 >> 41': 2.92,
'142 >> 42': 3.94,
'142 >> 43': 1.1,
'142 >> 48': 1.0,
'142 >> 50': 1.08,
'142 >> 68': 1.88,
'142 >> 74': 3.89,
…}
```

1. Create a `mean_distance` column that is a copy of the `pickup_dropoff` helper column.

2. Use the `map()` method on the `mean_distance` series. Pass `grouped_dict` as its argument. Reassign the result back to the `mean_distance` series. When you pass a dictionary to the `Series.map()` method, it will replace the data in the series where that data matches the dictionary's keys. The values that get imputed are the values of the dictionary.

```
Example:
df['mean_distance']
```

mean_distance
'A B'
'C D'
'A B'
'D C'
'E F'

```
grouped_dict = {'A B': 1.25, 'C D': 2, 'D C': 3}
df['mean_distance`] = df['mean_distance'].map(grouped_dict)
df['mean_distance']
```

mean_distance
1.25
2
1.25
3
NaN

When used this way, the `map()` `Series` method is very similar to `replace()`, however, note that `map()` will impute `NaN` for any values in the series that do not have a corresponding key in the mapping dictionary, so be careful.

[27]:
```
# 1. Create a mean_distance column that is a copy of the pickup_dropoff helper␣
↪column
df0["mean_distance"] = df0["pickup_dropoff"]

# 2. Map `grouped_dict` to the `mean_distance` column
df0["mean_distance"] = df0["mean_distance"].map(grouped_dict)

# Confirm that it worked
df0["mean_distance"].head(5)
```

[27]:
```
0    3.52
1    3.11
2    0.88
3    3.70
4    4.44
Name: mean_distance, dtype: float64
```

Create `mean_duration` column Repeat the process used to create the `mean_distance` column to create a `mean_duration` column.

[28]:
```python
grouped_pickup_dropoff = round(df0.
 ↪groupby(["pickup_dropoff"])[["trip_duration"]].mean(), 2)
grouped_pickup_dropoff.head(5)
```

[28]:

pickup_dropoff	trip_duration
1 >> 1	0.47
10 >> 148	69.37
100 >> 1	48.18
100 >> 100	3.13
100 >> 107	11.20

[29]:
```python
# 1. Convert `grouped` to a dictionary
grouped_dict = grouped_pickup_dropoff.to_dict()

# 2. Reassign to only contain the inner dictionary
grouped_dict = grouped_dict["trip_duration"]
grouped_dict
```

[29]:
```
{'1 >> 1': 0.47,
 '10 >> 148': 69.37,
 '100 >> 1': 48.18,
 '100 >> 100': 3.13,
 '100 >> 107': 11.2,
 '100 >> 113': 14.45,
 '100 >> 114': 15.32,
 '100 >> 12': 17.98,
 '100 >> 125': 17.61,
 '100 >> 13': 31.9,
 '100 >> 132': 35.22,
 '100 >> 137': 9.29,
 '100 >> 138': 44.78,
 '100 >> 140': 19.65,
 '100 >> 141': 13.62,
 '100 >> 142': 11.6,
 '100 >> 143': 10.88,
 '100 >> 144': 22.86,
 '100 >> 148': 33.42,
 '100 >> 151': 15.41,
 '100 >> 152': 19.77,
 '100 >> 158': 11.84,
 '100 >> 161': 8.99,
 '100 >> 162': 11.12,
 '100 >> 163': 11.47,
```

```
'100 >> 164': 9.52,
'100 >> 166': 15.82,
'100 >> 170': 8.28,
'100 >> 177': 58.53,
'100 >> 181': 28.55,
'100 >> 186': 5.77,
'100 >> 193': 15.57,
'100 >> 198': 53.9,
'100 >> 202': 39.35,
'100 >> 209': 29.73,
'100 >> 211': 14.34,
'100 >> 224': 10.42,
'100 >> 225': 44.8,
'100 >> 229': 18.77,
'100 >> 230': 8.27,
'100 >> 231': 22.85,
'100 >> 232': 17.83,
'100 >> 233': 10.12,
'100 >> 234': 10.39,
'100 >> 236': 22.04,
'100 >> 237': 21.57,
'100 >> 238': 14.29,
'100 >> 239': 16.24,
'100 >> 243': 26.15,
'100 >> 244': 18.6,
'100 >> 246': 9.62,
'100 >> 249': 11.52,
'100 >> 25': 28.23,
'100 >> 255': 15.65,
'100 >> 256': 30.92,
'100 >> 261': 21.97,
'100 >> 262': 20.18,
'100 >> 263': 26.6,
'100 >> 39': 60.05,
'100 >> 4': 15.99,
'100 >> 40': 37.73,
'100 >> 41': 19.23,
'100 >> 42': 17.67,
'100 >> 43': 17.67,
'100 >> 45': 22.45,
'100 >> 48': 6.59,
'100 >> 49': 53.13,
'100 >> 50': 8.85,
'100 >> 66': 24.02,
'100 >> 68': 8.56,
'100 >> 7': 21.67,
'100 >> 74': 15.7,
```

```
'100 >> 75': 18.23,
'100 >> 79': 17.69,
'100 >> 87': 33.48,
'100 >> 88': 29.36,
'100 >> 90': 8.32,
'100 >> 95': 18.75,
'106 >> 106': 0.17,
'106 >> 181': 4.03,
'106 >> 228': 6.17,
'106 >> 231': 21.7,
'106 >> 40': 4.18,
'107 >> 1': 27.55,
'107 >> 100': 17.81,
'107 >> 107': 4.39,
'107 >> 113': 5.78,
'107 >> 114': 8.17,
'107 >> 125': 12.07,
'107 >> 127': 25.65,
'107 >> 13': 16.3,
'107 >> 130': 49.85,
'107 >> 132': 50.75,
'107 >> 137': 4.49,
'107 >> 138': 31.05,
'107 >> 140': 18.38,
'107 >> 141': 25.54,
'107 >> 142': 16.22,
'107 >> 143': 20.95,
'107 >> 144': 13.2,
'107 >> 145': 12.42,
'107 >> 146': 10.4,
'107 >> 147': 21.83,
'107 >> 148': 12.43,
'107 >> 152': 19.63,
'107 >> 158': 13.72,
'107 >> 161': 12.62,
'107 >> 162': 10.03,
'107 >> 163': 12.92,
'107 >> 164': 7.77,
'107 >> 170': 8.03,
'107 >> 186': 12.71,
'107 >> 196': 26.57,
'107 >> 202': 18.0,
'107 >> 209': 12.15,
'107 >> 21': 34.77,
'107 >> 211': 12.6,
'107 >> 223': 24.08,
'107 >> 224': 6.12,
```

'107 >> 229': 13.8,
'107 >> 23': 35.85,
'107 >> 230': 15.92,
'107 >> 231': 15.06,
'107 >> 232': 13.4,
'107 >> 233': 11.61,
'107 >> 234': 6.61,
'107 >> 236': 21.25,
'107 >> 237': 18.79,
'107 >> 238': 25.69,
'107 >> 244': 32.83,
'107 >> 246': 13.84,
'107 >> 249': 14.93,
'107 >> 25': 28.03,
'107 >> 256': 19.33,
'107 >> 257': 18.27,
'107 >> 26': 20.92,
'107 >> 261': 6.65,
'107 >> 262': 18.75,
'107 >> 263': 24.74,
'107 >> 265': 29.35,
'107 >> 36': 23.03,
'107 >> 37': 30.35,
'107 >> 4': 6.49,
'107 >> 41': 21.11,
'107 >> 42': 15.87,
'107 >> 43': 10.57,
'107 >> 45': 14.86,
'107 >> 48': 17.53,
'107 >> 49': 30.72,
'107 >> 66': 24.7,
'107 >> 68': 11.85,
'107 >> 7': 22.41,
'107 >> 74': 18.33,
'107 >> 75': 17.51,
'107 >> 79': 8.74,
'107 >> 80': 29.9,
'107 >> 82': 23.95,
'107 >> 87': 12.72,
'107 >> 88': 16.66,
'107 >> 89': 23.17,
'107 >> 90': 7.25,
'112 >> 112': 5.07,
'112 >> 223': 13.38,
'112 >> 263': 22.85,
'112 >> 49': 7.83,
'112 >> 66': 7.68,

```
'112 >> 80': 2.6,
'113 >> 100': 13.97,
'113 >> 106': 30.77,
'113 >> 107': 9.66,
'113 >> 112': 16.57,
'113 >> 113': 5.34,
'113 >> 114': 6.36,
'113 >> 116': 22.62,
'113 >> 125': 8.47,
'113 >> 13': 12.69,
'113 >> 137': 10.91,
'113 >> 138': 38.57,
'113 >> 14': 49.0,
'113 >> 140': 32.25,
'113 >> 141': 21.74,
'113 >> 142': 22.53,
'113 >> 143': 22.85,
'113 >> 144': 9.09,
'113 >> 146': 28.27,
'113 >> 148': 11.4,
'113 >> 152': 21.18,
'113 >> 158': 8.01,
'113 >> 161': 17.82,
'113 >> 162': 13.36,
'113 >> 163': 14.02,
'113 >> 164': 10.33,
'113 >> 17': 19.08,
'113 >> 170': 13.88,
'113 >> 181': 34.42,
'113 >> 186': 11.33,
'113 >> 209': 12.8,
'113 >> 211': 7.86,
'113 >> 22': 53.75,
'113 >> 224': 10.78,
'113 >> 230': 19.25,
'113 >> 231': 10.74,
'113 >> 232': 14.63,
'113 >> 233': 16.33,
'113 >> 234': 6.48,
'113 >> 236': 23.27,
'113 >> 237': 25.81,
'113 >> 238': 25.46,
'113 >> 239': 23.43,
'113 >> 243': 43.98,
'113 >> 244': 24.68,
'113 >> 246': 13.56,
'113 >> 249': 6.46,
```

'113 >> 255': 25.88,
'113 >> 256': 9.57,
'113 >> 261': 13.08,
'113 >> 262': 20.1,
'113 >> 263': 23.75,
'113 >> 264': 0.0,
'113 >> 33': 20.18,
'113 >> 36': 28.0,
'113 >> 4': 8.4,
'113 >> 41': 32.22,
'113 >> 42': 38.12,
'113 >> 45': 14.73,
'113 >> 48': 19.56,
'113 >> 50': 16.69,
'113 >> 66': 19.65,
'113 >> 68': 8.98,
'113 >> 79': 6.51,
'113 >> 80': 28.55,
'113 >> 87': 13.11,
'113 >> 88': 9.89,
'113 >> 90': 6.26,
'113 >> 94': 29.13,
'114 >> 100': 15.78,
'114 >> 107': 8.72,
'114 >> 112': 18.02,
'114 >> 113': 4.44,
'114 >> 114': 4.34,
'114 >> 116': 21.95,
'114 >> 125': 6.65,
'114 >> 13': 10.0,
'114 >> 137': 10.35,
'114 >> 14': 23.0,
'114 >> 140': 15.2,
'114 >> 141': 23.68,
'114 >> 142': 28.57,
'114 >> 143': 15.34,
'114 >> 144': 6.49,
'114 >> 145': 33.73,
'114 >> 148': 9.38,
'114 >> 151': 39.6,
'114 >> 158': 8.7,
'114 >> 161': 22.03,
'114 >> 162': 17.32,
'114 >> 163': 30.85,
'114 >> 164': 11.33,
'114 >> 166': 25.51,
'114 >> 169': 78.37,

```
'114 >> 170': 16.92,
'114 >> 181': 16.58,
'114 >> 186': 9.39,
'114 >> 190': 27.83,
'114 >> 209': 13.82,
'114 >> 211': 6.05,
'114 >> 217': 10.03,
'114 >> 223': 22.13,
'114 >> 224': 35.12,
'114 >> 225': 19.38,
'114 >> 229': 13.82,
'114 >> 230': 25.55,
'114 >> 231': 8.65,
'114 >> 232': 12.56,
'114 >> 233': 14.82,
'114 >> 234': 9.55,
'114 >> 236': 26.34,
'114 >> 237': 19.78,
'114 >> 238': 26.43,
'114 >> 239': 24.92,
'114 >> 24': 24.98,
'114 >> 243': 28.72,
'114 >> 244': 22.05,
'114 >> 246': 10.99,
'114 >> 249': 7.07,
'114 >> 255': 19.13,
'114 >> 257': 17.27,
'114 >> 260': 14.75,
'114 >> 261': 16.45,
'114 >> 262': 23.62,
'114 >> 263': 19.41,
'114 >> 36': 27.48,
'114 >> 4': 12.7,
'114 >> 43': 27.72,
'114 >> 45': 8.58,
'114 >> 48': 16.68,
'114 >> 49': 18.0,
'114 >> 50': 24.01,
'114 >> 62': 27.45,
'114 >> 65': 18.72,
'114 >> 66': 21.86,
'114 >> 68': 11.67,
'114 >> 69': 39.03,
'114 >> 7': 33.75,
'114 >> 79': 8.61,
'114 >> 87': 15.68,
'114 >> 90': 9.48,
```

'114 >> 97': 20.15,
'116 >> 116': 2.8,
'116 >> 119': 8.25,
'116 >> 132': 27.93,
'116 >> 159': 10.47,
'116 >> 162': 17.08,
'116 >> 166': 6.39,
'116 >> 186': 24.3,
'116 >> 230': 19.65,
'116 >> 238': 12.48,
'116 >> 239': 16.35,
'116 >> 244': 4.86,
'116 >> 41': 7.28,
'116 >> 42': 10.93,
'116 >> 68': 25.48,
'116 >> 74': 15.08,
'116 >> 75': 26.57,
'116 >> 79': 41.27,
'118 >> 118': 8.97,
'12 >> 100': 22.37,
'12 >> 13': 7.85,
'12 >> 142': 24.17,
'12 >> 144': 16.92,
'12 >> 151': 40.78,
'12 >> 163': 31.68,
'12 >> 164': 32.02,
'12 >> 170': 15.37,
'12 >> 48': 37.25,
'123 >> 123': 5.4,
'125 >> 1': 34.33,
'125 >> 100': 16.06,
'125 >> 106': 26.25,
'125 >> 107': 16.89,
'125 >> 113': 2.57,
'125 >> 114': 7.4,
'125 >> 129': 21.48,
'125 >> 13': 9.46,
'125 >> 132': 88.78,
'125 >> 137': 10.93,
'125 >> 138': 38.48,
'125 >> 140': 27.75,
'125 >> 141': 27.12,
'125 >> 142': 22.13,
'125 >> 144': 6.72,
'125 >> 148': 12.6,
'125 >> 151': 15.25,
'125 >> 158': 4.39,

```
'125 >> 161': 23.27,
'125 >> 162': 27.5,
'125 >> 163': 14.04,
'125 >> 164': 16.39,
'125 >> 170': 20.64,
'125 >> 186': 10.26,
'125 >> 188': 20.8,
'125 >> 211': 6.73,
'125 >> 227': 23.1,
'125 >> 230': 21.35,
'125 >> 231': 7.26,
'125 >> 234': 13.06,
'125 >> 236': 17.53,
'125 >> 237': 19.48,
'125 >> 238': 25.82,
'125 >> 239': 20.83,
'125 >> 244': 33.17,
'125 >> 246': 16.03,
'125 >> 249': 4.82,
'125 >> 255': 21.82,
'125 >> 256': 16.27,
'125 >> 261': 9.69,
'125 >> 263': 42.4,
'125 >> 42': 26.75,
'125 >> 48': 16.95,
'125 >> 49': 23.15,
'125 >> 68': 12.76,
'125 >> 75': 26.62,
'125 >> 79': 12.83,
'125 >> 87': 12.1,
'125 >> 88': 12.53,
'125 >> 90': 7.58,
'125 >> 97': 23.33,
'127 >> 243': 11.34,
'128 >> 238': 15.15,
'129 >> 129': 5.12,
'129 >> 160': 18.28,
'129 >> 164': 10.65,
'129 >> 173': 9.6,
'129 >> 207': 4.25,
'129 >> 70': 9.13,
'13 >> 100': 25.58,
'13 >> 107': 19.3,
'13 >> 113': 20.42,
'13 >> 114': 13.9,
'13 >> 12': 5.0,
'13 >> 125': 5.75,
```

```
'13 >> 13': 5.71,
'13 >> 132': 53.82,
'13 >> 137': 14.07,
'13 >> 138': 36.88,
'13 >> 14': 18.33,
'13 >> 140': 16.54,
'13 >> 141': 18.77,
'13 >> 142': 23.43,
'13 >> 143': 17.5,
'13 >> 144': 11.38,
'13 >> 148': 15.06,
'13 >> 158': 11.95,
'13 >> 161': 25.87,
'13 >> 162': 25.49,
'13 >> 163': 34.2,
'13 >> 164': 30.17,
'13 >> 166': 25.37,
'13 >> 17': 23.22,
'13 >> 170': 24.66,
'13 >> 181': 12.17,
'13 >> 186': 23.76,
'13 >> 209': 4.83,
'13 >> 211': 12.34,
'13 >> 224': 13.72,
'13 >> 225': 24.98,
'13 >> 226': 39.02,
'13 >> 229': 17.46,
'13 >> 230': 26.69,
'13 >> 231': 6.28,
'13 >> 232': 16.52,
'13 >> 233': 22.43,
'13 >> 234': 22.76,
'13 >> 236': 34.71,
'13 >> 237': 26.72,
'13 >> 238': 22.53,
'13 >> 239': 21.99,
'13 >> 244': 41.83,
'13 >> 246': 11.65,
'13 >> 249': 12.38,
'13 >> 25': 19.67,
'13 >> 255': 21.0,
'13 >> 261': 12.58,
'13 >> 262': 24.4,
'13 >> 263': 27.47,
'13 >> 33': 14.0,
'13 >> 40': 8.72,
'13 >> 45': 12.33,
```

'13 >> 48': 20.28,
'13 >> 49': 24.28,
'13 >> 50': 14.95,
'13 >> 54': 19.45,
'13 >> 55': 33.93,
'13 >> 65': 21.67,
'13 >> 68': 10.85,
'13 >> 74': 20.0,
'13 >> 79': 14.93,
'13 >> 85': 44.17,
'13 >> 87': 8.09,
'13 >> 88': 8.28,
'13 >> 90': 17.0,
'13 >> 91': 33.57,
'130 >> 230': 39.48,
'130 >> 64': 11.77,
'131 >> 9': 7.53,
'132 >> 10': 8.99,
'132 >> 100': 68.73,
'132 >> 102': 25.17,
'132 >> 106': 40.48,
'132 >> 107': 50.46,
'132 >> 11': 32.71,
'132 >> 112': 37.44,
'132 >> 113': 40.71,
'132 >> 114': 69.38,
'132 >> 117': 23.87,
'132 >> 121': 16.12,
'132 >> 123': 22.75,
'132 >> 124': 15.38,
'132 >> 125': 53.12,
'132 >> 13': 51.34,
'132 >> 130': 23.67,
'132 >> 132': 3.02,
'132 >> 134': 19.47,
'132 >> 137': 29.2,
'132 >> 138': 27.68,
'132 >> 14': 35.61,
'132 >> 140': 36.79,
'132 >> 141': 50.4,
'132 >> 142': 50.37,
'132 >> 143': 76.88,
'132 >> 144': 37.0,
'132 >> 145': 40.76,
'132 >> 148': 46.34,
'132 >> 149': 28.98,
'132 >> 15': 28.93,

'132 >> 150': 27.22,
'132 >> 151': 48.53,
'132 >> 152': 38.42,
'132 >> 158': 49.74,
'132 >> 161': 46.48,
'132 >> 162': 44.0,
'132 >> 163': 52.94,
'132 >> 164': 59.56,
'132 >> 166': 40.95,
'132 >> 17': 45.25,
'132 >> 170': 37.11,
'132 >> 174': 28.97,
'132 >> 177': 22.53,
'132 >> 179': 39.88,
'132 >> 181': 42.7,
'132 >> 186': 56.44,
'132 >> 188': 33.52,
'132 >> 189': 43.68,
'132 >> 19': 14.38,
'132 >> 195': 48.67,
'132 >> 196': 25.58,
'132 >> 197': 17.7,
'132 >> 198': 28.22,
'132 >> 201': 28.43,
'132 >> 205': 16.28,
'132 >> 209': 56.2,
'132 >> 211': 45.62,
'132 >> 212': 24.28,
'132 >> 213': 35.88,
'132 >> 215': 10.3,
'132 >> 216': 12.31,
'132 >> 218': 14.03,
'132 >> 22': 33.62,
'132 >> 220': 87.8,
'132 >> 222': 18.03,
'132 >> 223': 34.0,
'132 >> 224': 46.87,
'132 >> 225': 33.15,
'132 >> 226': 34.41,
'132 >> 228': 41.62,
'132 >> 229': 47.61,
'132 >> 23': 88.78,
'132 >> 230': 59.6,
'132 >> 231': 51.76,
'132 >> 232': 37.55,
'132 >> 233': 43.58,
'132 >> 234': 49.83,

```
'132 >> 236': 50.25,
'132 >> 237': 56.47,
'132 >> 238': 43.78,
'132 >> 239': 44.86,
'132 >> 24': 38.27,
'132 >> 241': 33.18,
'132 >> 243': 36.43,
'132 >> 244': 43.95,
'132 >> 246': 66.32,
'132 >> 248': 31.82,
'132 >> 249': 48.12,
'132 >> 25': 49.07,
'132 >> 252': 31.33,
'132 >> 255': 37.05,
'132 >> 256': 43.65,
'132 >> 257': 59.65,
'132 >> 259': 28.5,
'132 >> 26': 41.38,
'132 >> 261': 51.49,
'132 >> 262': 49.93,
'132 >> 263': 39.98,
'132 >> 264': 0.0,
'132 >> 265': 30.07,
'132 >> 28': 13.63,
'132 >> 33': 48.92,
'132 >> 36': 34.45,
'132 >> 37': 38.99,
'132 >> 38': 16.21,
'132 >> 39': 23.2,
'132 >> 4': 57.06,
'132 >> 40': 58.72,
'132 >> 42': 25.97,
'132 >> 43': 41.92,
'132 >> 48': 58.25,
'132 >> 49': 38.62,
'132 >> 50': 50.14,
'132 >> 51': 43.39,
'132 >> 52': 72.78,
'132 >> 54': 45.27,
'132 >> 55': 30.87,
'132 >> 61': 32.33,
'132 >> 62': 31.58,
'132 >> 64': 20.25,
'132 >> 65': 69.99,
'132 >> 66': 71.07,
'132 >> 68': 52.55,
'132 >> 7': 36.53,
```

```
'132 >> 70': 22.25,
'132 >> 71': 23.0,
'132 >> 72': 27.48,
'132 >> 74': 37.3,
'132 >> 76': 28.23,
'132 >> 77': 16.88,
'132 >> 79': 47.28,
'132 >> 80': 31.96,
'132 >> 82': 32.57,
'132 >> 83': 28.42,
'132 >> 85': 38.12,
'132 >> 86': 17.8,
'132 >> 87': 50.69,
'132 >> 88': 57.73,
'132 >> 89': 42.31,
'132 >> 9': 35.25,
'132 >> 90': 43.62,
'132 >> 91': 21.27,
'132 >> 92': 24.51,
'132 >> 93': 21.98,
'132 >> 95': 25.6,
'132 >> 97': 46.97,
'133 >> 133': 25.53,
'134 >> 197': 9.25,
'135 >> 75': 40.87,
'137 >> 100': 12.72,
'137 >> 107': 5.74,
'137 >> 112': 21.85,
'137 >> 113': 10.17,
'137 >> 114': 9.84,
'137 >> 125': 15.09,
'137 >> 13': 17.45,
'137 >> 132': 44.8,
'137 >> 135': 29.1,
'137 >> 137': 3.52,
'137 >> 138': 17.05,
'137 >> 14': 27.43,
'137 >> 140': 11.18,
'137 >> 141': 15.73,
'137 >> 142': 22.71,
'137 >> 145': 8.93,
'137 >> 148': 10.83,
'137 >> 158': 12.13,
'137 >> 161': 13.44,
'137 >> 162': 8.65,
'137 >> 163': 14.15,
'137 >> 164': 8.69,
```

'137 >> 170': 6.84,
'137 >> 181': 32.12,
'137 >> 186': 12.07,
'137 >> 209': 11.78,
'137 >> 220': 22.57,
'137 >> 223': 31.55,
'137 >> 224': 4.24,
'137 >> 229': 10.46,
'137 >> 230': 12.97,
'137 >> 231': 18.4,
'137 >> 232': 9.52,
'137 >> 233': 7.04,
'137 >> 234': 8.74,
'137 >> 236': 15.65,
'137 >> 237': 14.72,
'137 >> 238': 24.0,
'137 >> 239': 14.97,
'137 >> 243': 51.68,
'137 >> 246': 14.64,
'137 >> 249': 12.73,
'137 >> 255': 16.52,
'137 >> 261': 11.93,
'137 >> 262': 17.24,
'137 >> 263': 14.13,
'137 >> 4': 9.6,
'137 >> 41': 39.26,
'137 >> 42': 17.32,
'137 >> 43': 29.75,
'137 >> 45': 15.98,
'137 >> 48': 24.98,
'137 >> 50': 12.52,
'137 >> 61': 29.87,
'137 >> 68': 12.57,
'137 >> 7': 17.5,
'137 >> 74': 32.38,
'137 >> 79': 6.76,
'137 >> 82': 18.17,
'137 >> 87': 16.32,
'137 >> 88': 12.37,
'137 >> 90': 9.63,
'138 >> 1': 67.48,
'138 >> 100': 36.66,
'138 >> 106': 27.93,
'138 >> 107': 29.85,
'138 >> 112': 29.33,
'138 >> 113': 34.38,
'138 >> 114': 36.17,

```
'138 >> 116': 22.7,
'138 >> 121': 11.62,
'138 >> 125': 49.23,
'138 >> 127': 28.52,
'138 >> 129': 20.11,
'138 >> 13': 34.59,
'138 >> 130': 14.4,
'138 >> 132': 34.11,
'138 >> 134': 30.73,
'138 >> 137': 18.26,
'138 >> 138': 17.73,
'138 >> 14': 46.32,
'138 >> 140': 24.0,
'138 >> 141': 25.29,
'138 >> 142': 40.42,
'138 >> 143': 44.51,
'138 >> 144': 47.98,
'138 >> 145': 24.28,
'138 >> 146': 16.98,
'138 >> 148': 51.9,
'138 >> 15': 16.21,
'138 >> 151': 26.09,
'138 >> 152': 29.51,
'138 >> 158': 32.78,
'138 >> 160': 19.83,
'138 >> 161': 44.22,
'138 >> 162': 33.54,
'138 >> 163': 39.75,
'138 >> 164': 38.3,
'138 >> 166': 39.92,
'138 >> 17': 33.28,
'138 >> 170': 32.09,
'138 >> 171': 14.22,
'138 >> 174': 21.65,
'138 >> 175': 24.6,
'138 >> 177': 22.15,
'138 >> 178': 49.35,
'138 >> 179': 26.38,
'138 >> 180': 22.27,
'138 >> 181': 33.12,
'138 >> 182': 24.75,
'138 >> 186': 43.6,
'138 >> 188': 38.42,
'138 >> 189': 40.9,
'138 >> 192': 10.05,
'138 >> 196': 14.69,
'138 >> 197': 16.01,
```

'138 >> 198': 21.09,
'138 >> 200': 20.62,
'138 >> 209': 30.9,
'138 >> 210': 29.55,
'138 >> 211': 43.58,
'138 >> 220': 57.48,
'138 >> 223': 11.17,
'138 >> 224': 29.52,
'138 >> 225': 28.68,
'138 >> 226': 14.19,
'138 >> 229': 35.02,
'138 >> 230': 42.08,
'138 >> 231': 41.91,
'138 >> 232': 30.95,
'138 >> 233': 22.26,
'138 >> 234': 43.03,
'138 >> 236': 28.46,
'138 >> 237': 38.95,
'138 >> 238': 28.11,
'138 >> 239': 30.0,
'138 >> 243': 30.92,
'138 >> 244': 31.48,
'138 >> 246': 54.85,
'138 >> 249': 37.68,
'138 >> 25': 34.55,
'138 >> 252': 12.97,
'138 >> 255': 24.9,
'138 >> 256': 29.18,
'138 >> 257': 41.4,
'138 >> 260': 15.62,
'138 >> 261': 50.21,
'138 >> 262': 32.11,
'138 >> 263': 31.13,
'138 >> 265': 39.38,
'138 >> 29': 44.78,
'138 >> 33': 42.14,
'138 >> 36': 29.08,
'138 >> 37': 23.48,
'138 >> 4': 34.52,
'138 >> 41': 26.05,
'138 >> 42': 19.02,
'138 >> 43': 38.91,
'138 >> 48': 41.27,
'138 >> 49': 33.42,
'138 >> 50': 43.15,
'138 >> 51': 30.58,
'138 >> 52': 21.01,

'138 >> 53': 21.71,
'138 >> 56': 18.03,
'138 >> 61': 39.92,
'138 >> 62': 67.0,
'138 >> 64': 27.82,
'138 >> 65': 35.34,
'138 >> 66': 40.56,
'138 >> 68': 40.54,
'138 >> 69': 22.52,
'138 >> 7': 17.75,
'138 >> 70': 6.74,
'138 >> 74': 20.03,
'138 >> 75': 31.24,
'138 >> 79': 31.92,
'138 >> 80': 24.41,
'138 >> 81': 24.52,
'138 >> 82': 19.3,
'138 >> 83': 9.93,
'138 >> 87': 44.28,
'138 >> 88': 64.32,
'138 >> 89': 26.17,
'138 >> 90': 33.49,
'138 >> 92': 14.03,
'138 >> 93': 13.48,
'138 >> 95': 16.54,
'138 >> 97': 29.71,
'138 >> 98': 28.85,
'14 >> 14': 2.22,
'140 >> 107': 9.52,
'140 >> 113': 15.57,
'140 >> 125': 31.55,
'140 >> 13': 20.06,
'140 >> 132': 33.6,
'140 >> 135': 24.63,
'140 >> 137': 11.82,
'140 >> 138': 17.75,
'140 >> 140': 4.58,
'140 >> 141': 5.59,
'140 >> 142': 12.93,
'140 >> 143': 13.85,
'140 >> 151': 13.8,
'140 >> 161': 14.06,
'140 >> 162': 11.23,
'140 >> 163': 11.45,
'140 >> 164': 20.73,
'140 >> 166': 29.9,
'140 >> 170': 11.27,

```
'140 >> 179': 17.4,
'140 >> 186': 25.35,
'140 >> 193': 23.43,
'140 >> 209': 19.16,
'140 >> 211': 33.65,
'140 >> 223': 16.25,
'140 >> 224': 11.68,
'140 >> 226': 18.48,
'140 >> 229': 7.92,
'140 >> 230': 22.72,
'140 >> 231': 28.52,
'140 >> 232': 10.82,
'140 >> 233': 12.05,
'140 >> 234': 18.32,
'140 >> 236': 10.28,
'140 >> 237': 7.74,
'140 >> 238': 16.8,
'140 >> 239': 17.06,
'140 >> 24': 22.75,
'140 >> 243': 19.58,
'140 >> 244': 19.57,
'140 >> 246': 38.72,
'140 >> 249': 24.28,
'140 >> 260': 16.63,
'140 >> 262': 5.1,
'140 >> 263': 4.8,
'140 >> 4': 15.78,
'140 >> 43': 14.15,
'140 >> 45': 10.78,
'140 >> 48': 21.25,
'140 >> 50': 23.38,
'140 >> 52': 22.15,
'140 >> 65': 16.12,
'140 >> 66': 18.25,
'140 >> 68': 23.36,
'140 >> 7': 28.03,
'140 >> 74': 10.8,
'140 >> 75': 6.33,
'140 >> 79': 16.3,
'140 >> 83': 22.22,
'140 >> 85': 49.92,
'140 >> 87': 18.06,
'140 >> 88': 13.63,
'140 >> 90': 29.77,
'140 >> 95': 41.13,
'140 >> 97': 39.95,
'141 >> 100': 15.51,
```

```
'141 >> 107': 15.38,
'141 >> 112': 18.3,
'141 >> 113': 26.31,
'141 >> 114': 22.08,
'141 >> 116': 24.18,
'141 >> 13': 22.2,
'141 >> 130': 34.92,
'141 >> 132': 62.26,
'141 >> 133': 20.98,
'141 >> 137': 13.63,
'141 >> 138': 17.54,
'141 >> 140': 6.89,
'141 >> 141': 4.56,
'141 >> 142': 11.44,
'141 >> 143': 12.58,
'141 >> 145': 13.92,
'141 >> 148': 18.55,
'141 >> 151': 19.6,
'141 >> 158': 35.57,
'141 >> 161': 13.52,
'141 >> 162': 8.62,
'141 >> 163': 11.71,
'141 >> 164': 17.43,
'141 >> 166': 19.78,
'141 >> 170': 12.78,
'141 >> 173': 19.63,
'141 >> 178': 39.15,
'141 >> 186': 17.16,
'141 >> 193': 10.48,
'141 >> 196': 25.7,
'141 >> 209': 20.25,
'141 >> 211': 22.28,
'141 >> 220': 34.12,
'141 >> 224': 11.2,
'141 >> 226': 7.33,
'141 >> 229': 7.11,
'141 >> 230': 16.49,
'141 >> 231': 30.07,
'141 >> 233': 8.13,
'141 >> 234': 15.14,
'141 >> 236': 7.67,
'141 >> 237': 4.93,
'141 >> 238': 16.01,
'141 >> 239': 13.17,
'141 >> 24': 16.05,
'141 >> 243': 19.43,
'141 >> 244': 28.93,
```

'141 >> 246': 23.05,
'141 >> 249': 37.48,
'141 >> 255': 31.33,
'141 >> 261': 22.26,
'141 >> 262': 5.02,
'141 >> 263': 4.12,
'141 >> 4': 11.86,
'141 >> 42': 16.04,
'141 >> 43': 8.89,
'141 >> 48': 19.23,
'141 >> 50': 16.94,
'141 >> 65': 27.14,
'141 >> 68': 30.35,
'141 >> 7': 15.76,
'141 >> 74': 10.49,
'141 >> 75': 8.16,
'141 >> 79': 15.35,
'141 >> 80': 26.37,
'141 >> 88': 17.97,
'141 >> 90': 22.14,
'142 >> 100': 12.3,
'142 >> 107': 21.99,
'142 >> 113': 17.48,
'142 >> 114': 22.42,
'142 >> 116': 24.43,
'142 >> 125': 30.7,
'142 >> 127': 36.14,
'142 >> 129': 23.5,
'142 >> 13': 24.06,
'142 >> 132': 48.95,
'142 >> 137': 17.84,
'142 >> 138': 42.18,
'142 >> 140': 11.58,
'142 >> 141': 11.2,
'142 >> 142': 4.21,
'142 >> 143': 6.18,
'142 >> 144': 40.13,
'142 >> 145': 20.4,
'142 >> 148': 21.83,
'142 >> 151': 11.77,
'142 >> 158': 16.28,
'142 >> 161': 12.58,
'142 >> 162': 15.1,
'142 >> 163': 8.11,
'142 >> 164': 17.32,
'142 >> 166': 11.36,
'142 >> 17': 56.8,

```
'142 >> 170': 16.76,
'142 >> 174': 29.78,
'142 >> 181': 23.82,
'142 >> 186': 14.74,
'142 >> 209': 34.52,
'142 >> 211': 37.13,
'142 >> 220': 17.47,
'142 >> 223': 22.76,
'142 >> 224': 33.11,
'142 >> 225': 29.23,
'142 >> 229': 12.58,
'142 >> 230': 8.53,
'142 >> 231': 19.32,
'142 >> 233': 21.9,
'142 >> 234': 16.57,
'142 >> 236': 12.43,
'142 >> 237': 10.11,
'142 >> 238': 7.66,
'142 >> 239': 6.28,
'142 >> 24': 7.41,
'142 >> 243': 21.26,
'142 >> 244': 19.99,
'142 >> 246': 15.08,
'142 >> 249': 15.24,
'142 >> 261': 31.2,
'142 >> 262': 12.59,
'142 >> 263': 15.7,
'142 >> 264': 1.5,
'142 >> 41': 15.16,
'142 >> 42': 21.82,
'142 >> 43': 5.83,
'142 >> 48': 7.32,
'142 >> 50': 7.41,
'142 >> 68': 15.97,
'142 >> 74': 34.26,
…}
```

[30]:
```
# 1. Create a mean_distance column that is a copy of the pickup_dropoff helper␣
↪column
df0["mean_duration"] = df0["pickup_dropoff"]

# 2. Map `grouped_dict` to the `mean_distance` column
df0["mean_duration"] = df0["mean_duration"].map(grouped_dict)

# Confirm that it worked
df0["mean_duration"].head(5)
```

[30]:
```
0    22.85
1    24.47
2     7.25
3    30.25
4    14.62
Name: mean_duration, dtype: float64
```

Create `day` and `month` columns Create two new columns, `day` (name of day) and `month` (name of month) by extracting the relevant information from the `tpep_pickup_datetime` column.

[31]:
```
# Create 'day' col
df0["day"] = df0["tpep_pickup_datetime"].dt.day_name().str.lower()

# Create 'month' col
df0["month"] = df0["tpep_pickup_datetime"].dt.month_name().str.lower()
```

Create `rush_hour` column Define rush hour as: * Any weekday (not Saturday or Sunday) AND * Either from 06:00–10:00 or from 16:00–20:00

Create a binary `rush_hour` column that contains a 1 if the ride was during rush hour and a 0 if it was not.

[32]:
```
# Create 'rush_hour' col
df0["rush_hour"] = df0["tpep_pickup_datetime"].dt.hour

# If day is Saturday or Sunday, impute 0 in `rush_hour` column
df0.loc[df0["day"].isin(["saturday", "sunday"]), "rush_hour"] = 0

df0["rush_hour"].head(5)
```

[32]:
```
0     0
1    14
2     7
3     0
4     0
Name: rush_hour, dtype: int64
```

[33]:
```
def rush_hourizer(hour):
    if 6 <= hour <= 10:
        val = 1
    elif 16 <= hour <= 20:
        val = 1
    else:
        val = 0
    return val
```

[34]:
```
df0["rush_hour"] = df0["rush_hour"].apply(rush_hourizer)
df0["rush_hour"].head(5)
```

[34]:
```
0    0
1    0
2    1
3    0
4    0
Name: rush_hour, dtype: int64
```

[35]:
```
df0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22699 entries, 0 to 22698
Data columns (total 25 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  datetime64[ns]
 3   tpep_dropoff_datetime  22699 non-null  datetime64[ns]
 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
 18  trip_duration          22699 non-null  float64
 19  pickup_dropoff         22699 non-null  object
 20  mean_distance          22699 non-null  float64
 21  mean_duration          22699 non-null  float64
 22  day                    22699 non-null  object
 23  month                  22699 non-null  object
 24  rush_hour              22699 non-null  int64
dtypes: datetime64[ns](2), float64(11), int64(8), object(4)
memory usage: 4.3+ MB
```

[36]:
```
df1 = df0[["VendorID", "tpep_pickup_datetime", "tpep_dropoff_datetime", "day",␣
 ↪"month", "rush_hour",\
```

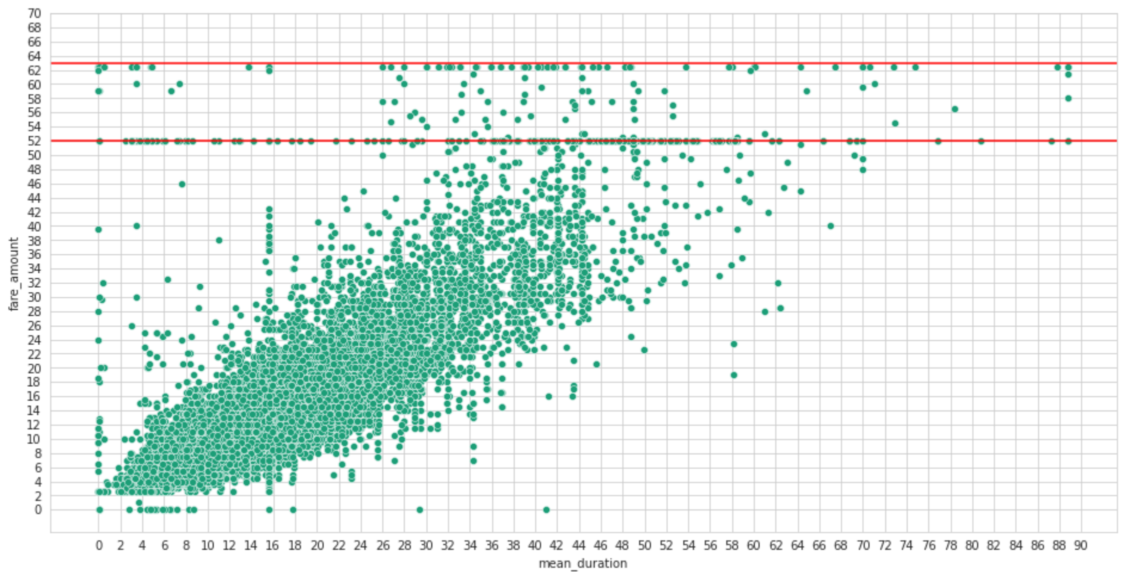
```
            "passenger_count", "trip_distance", "trip_duration","RatecodeID",␣
↪"store_and_fwd_flag",\
            "PULocationID", "DOLocationID", "pickup_dropoff", "mean_distance",␣
↪"mean_duration", "payment_type",\
            "fare_amount", "extra", "mta_tax", "tip_amount", "tolls_amount",␣
↪"improvement_surcharge", "total_amount"]]
df1.shape
```

[36]: (22699, 24)

4.2.8 Task 4. Scatter plot

Create a scatterplot to visualize the relationship between `mean_duration` and `fare_amount`.

[37]:
```
plt.figure(figsize= (16, 8))
sns.scatterplot(data= df1, x= "mean_duration", y= "fare_amount")
plt.axhline(52, color= "red")
plt.axhline(63, color= "red")
plt.xticks(range(0, 91, 2))
plt.yticks(range(0, 71, 2))
plt.show()
```



The `mean_duration` variable correlates with the target variable. But what are the horizontal lines around fare amounts of 52 dollars and 63 dollars? What are the values and how many are there?

You know what one of the lines represents. 62 dollars and 50 cents is the maximum that was imputed for outliers, so all former outliers will now have fare amounts of $62.50. What is the other line?

Check the value of the rides in the second horizontal line in the scatter plot.

[38]:
```
df1[df1["fare_amount"] > 50]["fare_amount"].value_counts().head(3)
```

[38]:
```
52.0    514
62.5     84
59.0      9
Name: fare_amount, dtype: int64
```

Examine the first 30 of these trips.

[39]:
```
# Set pandas to display all columns
pd.set_option("display.max_columns", None)
df1[df1["fare_amount"] == 52].head(30)
```

[39]:
```
      VendorID tpep_pickup_datetime tpep_dropoff_datetime        day  \
11           2  2017-03-05 19:15:30   2017-03-05 19:52:18     sunday
110          1  2017-06-03 14:24:57   2017-06-03 15:31:48   saturday
161          2  2017-11-11 20:16:16   2017-11-11 20:17:14   saturday
247          2  2017-12-06 23:37:08   2017-12-07 00:06:19  wednesday
379          2  2017-09-24 23:45:45   2017-09-25 00:15:14     sunday
388          1  2017-02-28 18:30:05   2017-02-28 19:09:55    tuesday
406          2  2017-06-05 12:51:58   2017-06-05 13:07:35     monday
449          2  2017-08-03 22:47:14   2017-08-03 23:32:41   thursday
468          2  2017-09-26 13:48:26   2017-09-26 14:31:17    tuesday
520          2  2017-04-23 21:34:48   2017-04-23 22:46:23     sunday
569          2  2017-11-22 21:31:32   2017-11-22 22:00:25  wednesday
572          2  2017-07-18 13:29:06   2017-07-18 13:29:19    tuesday
586          2  2017-06-26 13:39:12   2017-06-26 14:34:54     monday
692          2  2017-11-07 22:15:00   2017-11-07 22:45:32    tuesday
717          1  2017-12-06 05:19:50   2017-12-06 05:53:52  wednesday
719          1  2017-08-04 17:53:34   2017-08-04 18:50:56     friday
782          2  2017-06-09 09:31:25   2017-06-09 10:24:10     friday
816          2  2017-02-21 06:11:03   2017-02-21 06:59:39    tuesday
818          2  2017-06-20 08:15:18   2017-06-20 10:24:37    tuesday
835          2  2017-01-10 22:29:47   2017-01-10 23:06:46    tuesday
840          2  2017-10-27 21:50:00   2017-10-27 22:35:04     friday
861          1  2017-12-16 06:39:59   2017-12-16 07:07:59   saturday
881          2  2017-12-30 05:25:29   2017-12-30 06:01:29   saturday
958          1  2017-10-15 22:39:12   2017-10-15 23:14:22     sunday
970          2  2017-02-17 20:39:42   2017-02-17 21:13:29     friday
984          1  2017-08-23 18:23:26   2017-08-23 19:18:29  wednesday
1082         2  2017-02-07 17:20:19   2017-02-07 17:34:41    tuesday
1097         2  2017-08-14 23:01:15   2017-08-14 23:03:35     monday
1110         1  2017-09-06 10:46:17   2017-09-06 11:44:41  wednesday
1179         2  2017-06-19 06:23:13   2017-06-19 07:03:53     monday

          month  rush_hour  passenger_count  trip_distance  trip_duration  \
```

11	march	0	2	18.90	36.800000
110	june	0	1	18.00	66.850000
161	november	0	1	0.23	0.966667
247	december	0	1	18.93	29.183333
379	september	0	1	17.99	29.483333
388	february	1	1	18.40	39.833333
406	june	0	1	4.73	15.616667
449	august	0	2	18.21	45.450000
468	september	0	1	17.27	42.850000
520	april	0	6	18.34	71.583333
569	november	0	1	18.65	28.883333
572	july	0	1	0.00	0.216667
586	june	0	1	17.76	55.700000
692	november	0	2	16.97	30.533333
717	december	0	1	20.80	34.033333
719	august	1	1	21.60	57.366667
782	june	1	2	18.81	52.750000
816	february	1	5	16.94	48.600000
818	june	1	1	17.77	88.783333
835	january	0	1	18.57	36.983333
840	october	0	1	22.43	45.066667
861	december	0	2	17.80	28.000000
881	december	0	6	18.23	36.000000
958	october	0	1	21.80	35.166667
970	february	1	1	19.57	33.783333
984	august	1	1	16.70	55.050000
1082	february	1	1	1.09	14.366667
1097	august	0	5	2.12	2.333333
1110	september	1	1	19.10	58.400000
1179	june	1	6	19.77	40.666667

	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	\
11	2	N	236	132	
110	2	N	132	163	
161	2	N	132	132	
247	2	N	132	79	
379	2	N	132	234	
388	2	N	132	48	
406	2	N	228	88	
449	2	N	132	48	
468	2	N	186	132	
520	2	N	132	148	
569	2	N	132	144	
572	2	N	230	161	
586	2	N	211	132	
692	2	N	132	170	
717	2	N	132	239	

719	2	N	264	264
782	2	N	163	132
816	2	N	132	170
818	2	N	132	246
835	2	N	132	48
840	2	N	132	163
861	2	N	75	132
881	2	N	68	132
958	2	N	132	261
970	2	N	132	140
984	2	N	132	230
1082	2	N	170	48
1097	2	N	265	265
1110	2	N	239	132
1179	2	N	238	132

	pickup_dropoff	mean_distance	mean_duration	payment_type	fare_amount	\
11	236 >> 132	19.21	40.50	1	52.0	
110	132 >> 163	19.23	52.94	1	52.0	
161	132 >> 132	2.26	3.02	2	52.0	
247	132 >> 79	19.43	47.28	2	52.0	
379	132 >> 234	17.65	49.83	1	52.0	
388	132 >> 48	18.76	58.25	2	52.0	
406	228 >> 88	4.73	15.62	2	52.0	
449	132 >> 48	18.76	58.25	2	52.0	
468	186 >> 132	17.10	42.92	2	52.0	
520	132 >> 148	17.99	46.34	1	52.0	
569	132 >> 144	18.54	37.00	1	52.0	
572	230 >> 161	0.69	7.97	1	52.0	
586	211 >> 132	16.58	61.69	1	52.0	
692	132 >> 170	17.20	37.11	1	52.0	
717	132 >> 239	20.90	44.86	1	52.0	
719	264 >> 264	3.19	15.62	1	52.0	
782	163 >> 132	17.28	52.34	1	52.0	
816	132 >> 170	17.20	37.11	1	52.0	
818	132 >> 246	18.52	66.32	1	52.0	
835	132 >> 48	18.76	58.25	1	52.0	
840	132 >> 163	19.23	52.94	2	52.0	
861	75 >> 132	18.44	36.20	1	52.0	
881	68 >> 132	18.78	58.04	2	52.0	
958	132 >> 261	22.12	51.49	2	52.0	
970	132 >> 140	19.29	36.79	1	52.0	
984	132 >> 230	18.57	59.60	1	52.0	
1082	170 >> 48	1.27	14.14	2	52.0	
1097	265 >> 265	0.75	3.41	2	52.0	
1110	239 >> 132	19.80	50.56	1	52.0	
1179	238 >> 132	19.47	53.86	1	52.0	

	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	\
11	0.0	0.5	14.58	5.54	0.3	
110	0.0	0.5	0.00	0.00	0.3	
161	0.0	0.5	0.00	0.00	0.3	
247	0.0	0.5	0.00	0.00	0.3	
379	0.0	0.5	14.64	5.76	0.3	
388	4.5	0.5	0.00	5.54	0.3	
406	0.0	0.5	0.00	5.76	0.3	
449	0.0	0.5	0.00	5.76	0.3	
468	0.0	0.5	0.00	5.76	0.3	
520	0.0	0.5	5.00	0.00	0.3	
569	0.0	0.5	10.56	0.00	0.3	
572	0.0	0.5	11.71	5.76	0.3	
586	0.0	0.5	11.71	5.76	0.3	
692	0.0	0.5	11.71	5.76	0.3	
717	0.0	0.5	5.85	5.76	0.3	
719	4.5	0.5	12.60	5.76	0.3	
782	0.0	0.5	13.20	0.00	0.3	
816	0.0	0.5	2.00	5.54	0.3	
818	0.0	0.5	11.71	5.76	0.3	
835	0.0	0.5	13.20	0.00	0.3	
840	0.0	0.5	0.00	5.76	0.3	
861	0.0	0.5	6.00	5.76	0.3	
881	0.0	0.5	0.00	0.00	0.3	
958	0.0	0.5	0.00	0.00	0.3	
970	0.0	0.5	11.67	5.54	0.3	
984	4.5	0.5	42.29	0.00	0.3	
1082	4.5	0.5	0.00	5.54	0.3	
1097	0.0	0.5	0.00	0.00	0.3	
1110	0.0	0.5	15.80	0.00	0.3	
1179	0.0	0.5	17.57	5.76	0.3	

	total_amount
11	72.92
110	52.80
161	52.80
247	52.80
379	73.20
388	62.84
406	58.56
449	58.56
468	58.56
520	57.80
569	63.36
572	70.27
586	70.27

```
692         70.27
717         64.41
719         75.66
782         66.00
816         60.34
818         70.27
835         66.00
840         58.56
861         64.56
881         52.80
958         52.80
970         70.01
984         99.59
1082        62.84
1097        52.80
1110        68.60
1179        76.13
```

Question: What do you notice about the first 30 trips?

It seems that almost all of the trips in the first 30 rows where the fare amount was $52 either begin or end at location 132, and all of them have a `RatecodeID` of 2.

There is no readily apparent reason why PULocation 132 should have so many fares of 52 dollars. They seem to occur on all different days, at different times, with both vendors, in all months. However, there are many toll amounts of $5.76 and \$5.54. This would seem to indicate that location 132 is in an area that frequently requires tolls to get to and from. It's likely this is an airport.

4.2.9 Task 5. Isolate modeling variables

Drop features that are redundant, irrelevant, or that will not be available in a deployed environment.

[40]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22699 entries, 0 to 22698
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   VendorID               22699 non-null  int64
 1   tpep_pickup_datetime   22699 non-null  datetime64[ns]
 2   tpep_dropoff_datetime  22699 non-null  datetime64[ns]
 3   day                    22699 non-null  object
 4   month                  22699 non-null  object
 5   rush_hour              22699 non-null  int64
 6   passenger_count        22699 non-null  int64
 7   trip_distance          22699 non-null  float64
```

```
 8   trip_duration          22699 non-null  float64
 9   RatecodeID             22699 non-null  int64
 10  store_and_fwd_flag     22699 non-null  object
 11  PULocationID           22699 non-null  int64
 12  DOLocationID           22699 non-null  int64
 13  pickup_dropoff         22699 non-null  object
 14  mean_distance          22699 non-null  float64
 15  mean_duration          22699 non-null  float64
 16  payment_type           22699 non-null  int64
 17  fare_amount            22699 non-null  float64
 18  extra                  22699 non-null  float64
 19  mta_tax                22699 non-null  float64
 20  tip_amount             22699 non-null  float64
 21  tolls_amount           22699 non-null  float64
 22  improvement_surcharge  22699 non-null  float64
 23  total_amount           22699 non-null  float64
dtypes: datetime64[ns](2), float64(11), int64(7), object(4)
memory usage: 4.2+ MB
```

[41]:
```
df2 = df1.copy()

df2 = df2.drop(["tpep_pickup_datetime", "tpep_dropoff_datetime", "day",␣
 ↪"month", "trip_distance", "trip_duration",\
                "RatecodeID", "store_and_fwd_flag", "PULocationID",␣
 ↪"DOLocationID", "pickup_dropoff", "payment_type",\
                "extra", "mta_tax", "tip_amount", "tolls_amount",␣
 ↪"improvement_surcharge", "total_amount"], axis= 1)
df2.info()
```

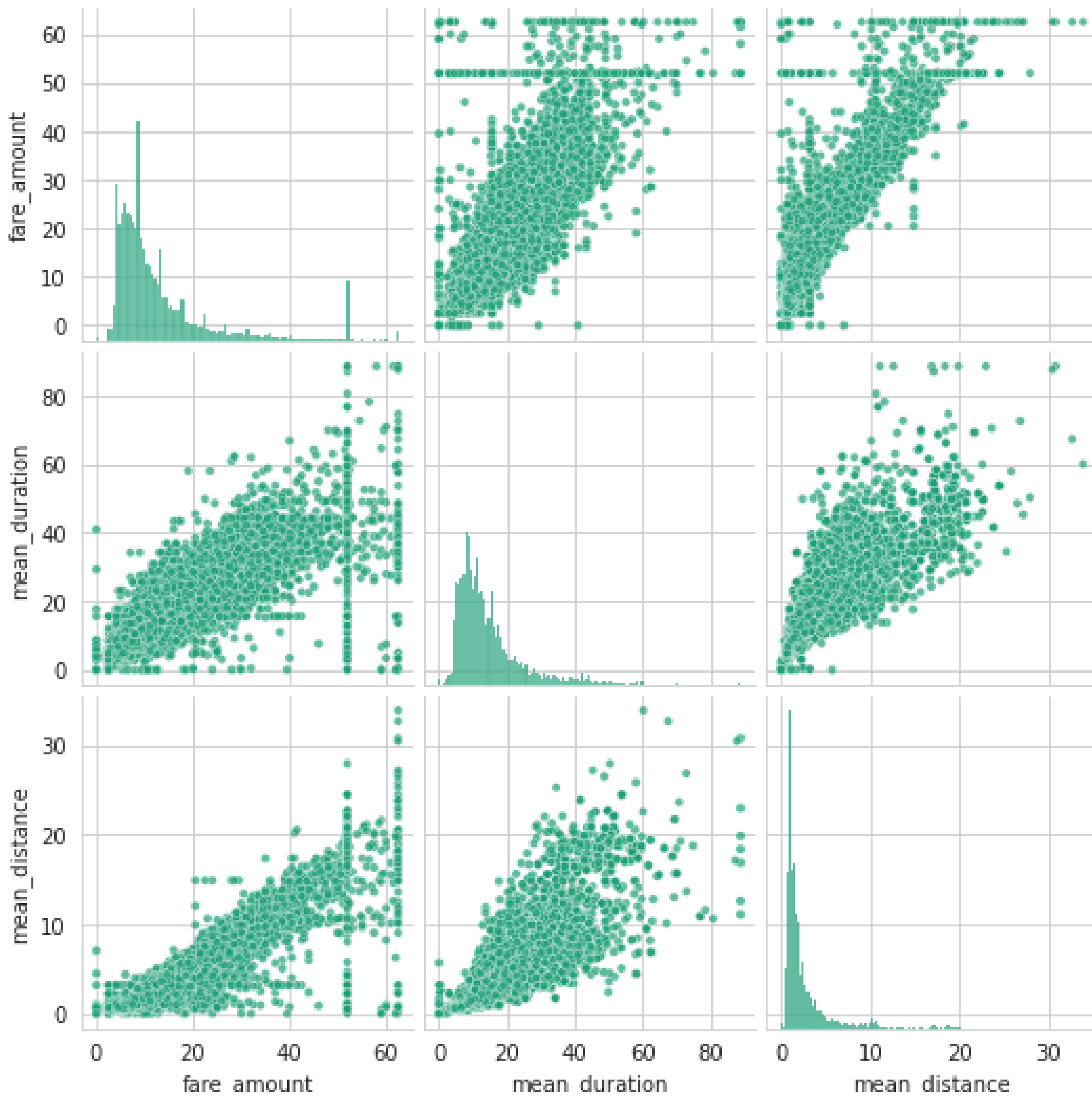
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22699 entries, 0 to 22698
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   VendorID         22699 non-null  int64
 1   rush_hour        22699 non-null  int64
 2   passenger_count  22699 non-null  int64
 3   mean_distance    22699 non-null  float64
 4   mean_duration    22699 non-null  float64
 5   fare_amount      22699 non-null  float64
dtypes: float64(3), int64(3)
memory usage: 1.0 MB
```

4.2.10 Task 6. Pair plot

Create a pairplot to visualize pairwise relationships between `fare_amount`, `mean_duration`, and `mean_distance`.

[42]:

```
# Create a pairplot to visualize pairwise relationships between variables in
 ↪the data
sns.pairplot(data= df2[["fare_amount", "mean_duration", "mean_distance"]],
 ↪plot_kws= {"alpha": 0.7, "size": 0.7})
plt.show()
```



These variables all show linear correlation with each other. Investigate this further.

4.2.11 Task 7. Identify correlations

Next, code a correlation matrix to help determine most correlated variables.

[43]:
```python
# Correlation matrix to help determine most correlated variables
df2.corr(method= "pearson")
```

[43]:
```
                 VendorID  rush_hour  passenger_count  mean_distance  \
VendorID         1.000000  -0.000752         0.266463       0.004737
rush_hour       -0.000752   1.000000        -0.024283      -0.046799
passenger_count  0.266463  -0.024283         1.000000       0.013433
mean_distance    0.004737  -0.046799         0.013433       1.000000
mean_duration    0.001874  -0.027496         0.015849       0.874863
fare_amount      0.001045  -0.025901         0.014942       0.910187

                 mean_duration  fare_amount
VendorID              0.001874     0.001045
rush_hour            -0.027496    -0.025901
passenger_count       0.015849     0.014942
mean_distance         0.874863     0.910187
mean_duration         1.000000     0.859104
fare_amount           0.859104     1.000000
```

Visualize a correlation heatmap of the data.

[44]:
```python
# Create correlation heatmap
plt.figure(figsize= (16, 8))
sns.heatmap(data= df2.corr(method= "pearson"), vmin= -1, vmax= 1, annot= True,␣
↪cmap= "YlGnBu")
plt.show()
```

Question: Which variable(s) are correlated with the target variable of `fare_amount`?

The variables which are strongly correlated with the target variable are mean_distance and mean_duration, while rush_hour and passenger_count are weekly correlated.

4.3 PACE: Construct

After analysis and deriving variables with close relationships, it is time to begin constructing the model. Consider the questions in your PACE Strategy Document to reflect on the Construct stage.

4.3.1 Task 8a. Split data into outcome variable and features

Set your X and y variables. X represents the features and y represents the outcome (target) variable.

[45]:
```
# Remove the target column from the features
X = df2.drop(columns='fare_amount')

# Set y variable
y = df2[["fare_amount"]]

# Display first few rows
X.head(3)
```

[45]:
```
   VendorID  rush_hour  passenger_count  mean_distance  mean_duration
0         2          0                6           3.52          22.85
1         1          0                1           3.11          24.47
2         1          1                1           0.88           7.25
```

4.3.2 Task 8b. Pre-process data

Dummy encode categorical variables

[46]:
```
# Convert VendorID to string
X["VendorID"] = X["VendorID"].astype("str")

# Get dummies
X = pd.get_dummies(X, drop_first= False)
X.head(3)
```

[46]:
```
   rush_hour  passenger_count  mean_distance  mean_duration  VendorID_1  \
0          0                6           3.52          22.85           0
1          0                1           3.11          24.47           1
2          1                1           0.88           7.25           1

   VendorID_2
0           1
1           0
```

2 0

4.3.3 Split data into training and test sets

Create training and testing sets. The test set should contain 20% of the total samples. Set `random_state=0`.

[47]:
```
# Create training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2,␣
↪random_state= 0)
```

4.3.4 Standardize the data

Use `StandardScaler()`, `fit()`, and `transform()` to standardize the `X_train` variables. Assign the results to a variable called `X_train_scaled`.

[48]:
```
# Standardize the X variables
scaler = StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
print("X_train_scaled:\n", X_train_scaled)
```

```
X_train_scaled:
 [[-0.77153979 -0.50301524  0.86949026  0.17649076 -0.89286563  0.89286563]
 [ 1.29610943 -0.50301524 -0.59914394 -0.69876345 -0.89286563  0.89286563]
 [-0.77153979  0.27331093 -0.4788558  -0.57301992  1.11998936 -1.11998936]
 …
 [-0.77153979 -0.50301524 -0.45088182 -0.67896132  1.11998936 -1.11998936]
 [ 1.29610943 -0.50301524 -0.59075174 -0.8571805   1.11998936 -1.11998936]
 [-0.77153979  1.82596329  0.83592148  1.13194361 -0.89286563  0.89286563]]
```

4.3.5 Fit the model

Instantiate your model and fit it to the training data.

[49]:
```
# Fit your model to the training data
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)
```

[49]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

4.3.6 Task 8c. Evaluate model

4.3.7 Train data

Evaluate your model performance by calculating the residual sum of squares and the explained variance score (R^2). Calculate the Mean Absolute Error, Mean Squared Error, and the Root Mean Squared Error.

[50]:
```python
# Evaluate the model performance on the training data
y_pred_train = lr.predict(X_train_scaled)

print("R^2:", r2_score(y_train, y_pred_train))
print("MAE:", mean_absolute_error(y_train, y_pred_train))
print("MSE:", mean_squared_error(y_train, y_pred_train))
print("RMSE:", np.sqrt(mean_squared_error(y_train, y_pred_train)))
```

```
R^2: 0.8398698730002254
MAE: 2.186209417536295
MSE: 17.886782423534164
RMSE: 4.22927682039544
```

4.3.8 Test data

Calculate the same metrics on the test data. Remember to scale the `X_test` data using the scaler that was fit to the training data. Do not refit the scaler to the testing data, just transform it. Call the results `X_test_scaled`.

[51]:
```python
# Scale the X_test data
X_test_scaled = scaler.transform(X_test)
```

[52]:
```python
# Evaluate the model performance on the testing data
y_pred_test = lr.predict(X_test_scaled)

print("R^2:", r2_score(y_test, y_pred_test))
print("MAE:", mean_absolute_error(y_test, y_pred_test))
print("MSE:", mean_squared_error(y_test, y_pred_test))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_test)))
```

```
R^2: 0.8682443054061614
MAE: 2.1337521175002836
MSE: 14.327983000717454
RMSE: 3.785232225467475
```

4.4 PACE: Execute

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

4.4.1 Task 9a. Results

Use the code cell below to get `actual`,`predicted`, and `residual` for the testing set, and store them as columns in a `results` dataframe.

[53]:
```
# Create a `results` dataframe
results = pd.DataFrame(data= {"actual": y_test["fare_amount"], "predicted":␣
 ↪y_pred_test.ravel()})
results.head(5)
```

[53]:

	actual	predicted
5818	14.0	12.338959
18134	28.0	16.540539
4655	5.5	6.701320
7378	15.5	16.201496
13914	9.5	10.506846

[54]:
```
results["residual"] = results["actual"] - results["predicted"]
results.head(5)
```

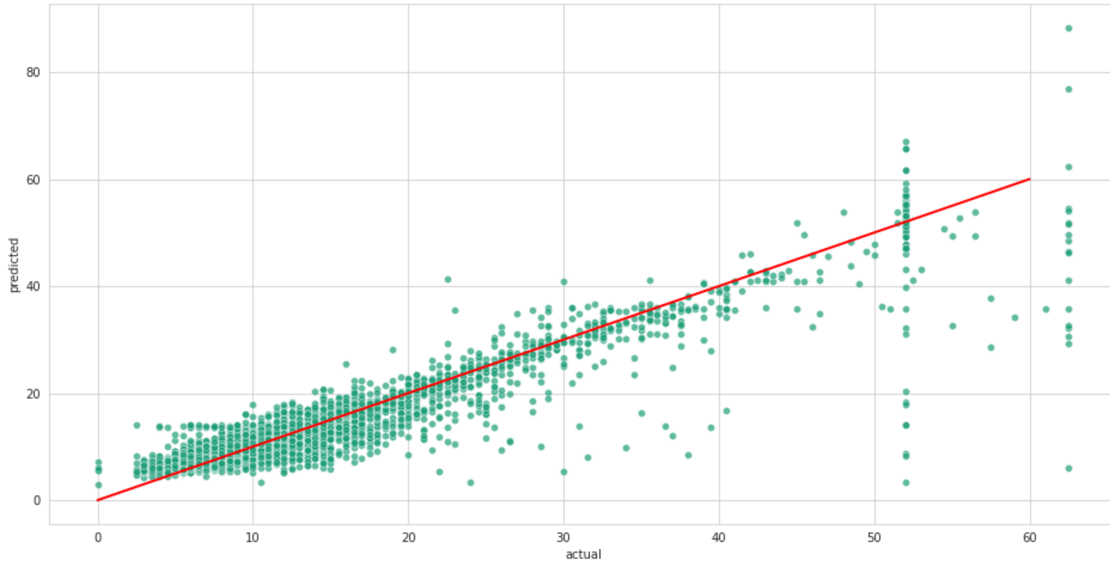
[54]:

	actual	predicted	residual
5818	14.0	12.338959	1.661041
18134	28.0	16.540539	11.459461
4655	5.5	6.701320	-1.201320
7378	15.5	16.201496	-0.701496
13914	9.5	10.506846	-1.006846

4.4.2 Task 9b. Visualize model results

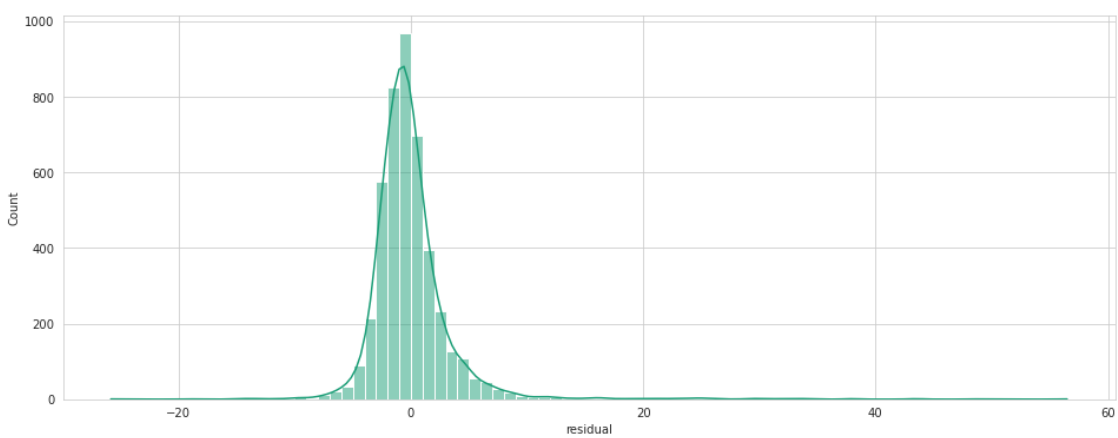
Create a scatterplot to visualize `actual` vs. `predicted`.

[55]:
```
# Create a scatterplot to visualize `predicted` over `actual`
plt.figure(figsize= (16, 8))
sns.scatterplot(data= results, x= "actual", y= "predicted", alpha= 0.7)
plt.plot([0, 60], [0, 60], c= "red", linewidth= "2")
plt.show()
```



Visualize the distribution of the `residuals` using a histogram.

[56]:
```
# Visualize the distribution of the `residuals`
plt.figure(figsize= (16, 6))
sns.histplot(data= results, x= "residual", kde= True, bins= np.arange(-20, 20,␣
↪1))
plt.show()
```

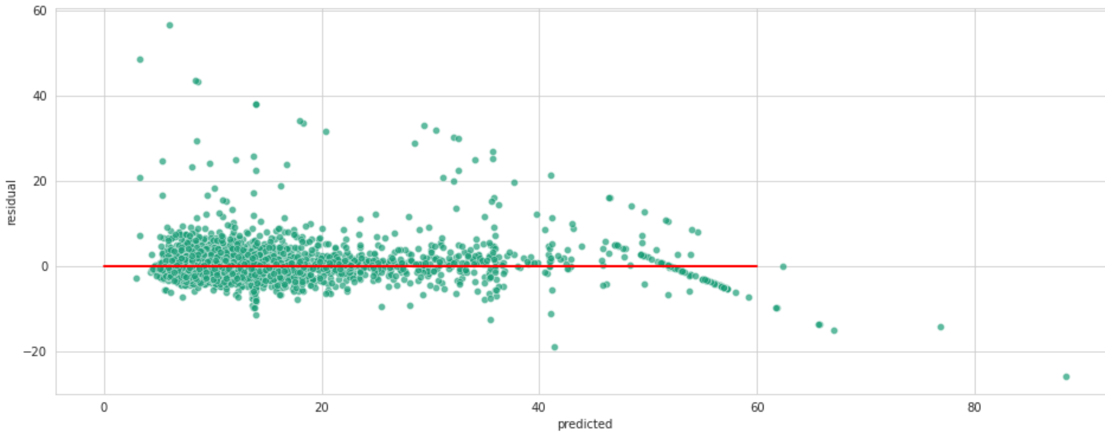


[57]:
```
# Calculate residual mean
print("Residual Mean =", round(results["residual"].mean(), 4))
```

```
Residual Mean = -0.0153
```

Create a scatterplot of `residuals` over `predicted`.

[58]:
```python
# Create a scatterplot of `residuals` over `predicted`
plt.figure(figsize= (16, 6))
sns.scatterplot(data= results, x= "predicted", y= "residual", alpha= 0.7)
plt.plot([0, 60], [0, 0], c= "red", linewidth= "2")
plt.show()
```



4.4.3 Task 9c. Coefficients

Use the `coef_` attribute to get the model's coefficients. The coefficients are output in the order of the features that were used to train the model. Which feature had the greatest effect on trip fare?

[59]:
```python
print("Coefficients:\n", lr.coef_)
```

```
Coefficients:
 [[ 0.12151169  0.03150761  7.13589815  2.81145665  0.02727586 -0.02727586]]
```

[60]:
```python
coefficients = round(pd.DataFrame(data= lr.coef_, columns= X.columns) * 100, 4)
coefficients
```

[60]:

```
   rush_hour  passenger_count  mean_distance  mean_duration  VendorID_1  \
0    12.1512           3.1508       713.5898       281.1457      2.7276

   VendorID_2
0    -2.7276
```

What do these coefficients mean? How should they be interpreted?

These coefficients represent the relative increase in fare_amount for every unit increased in standard deviation of these variables. To simplify the interpretation, let's calculate the standard deviation of the most effective variables:

[61]:
```
print(X_train['mean_distance'].std())

print(7.133867 / X_train['mean_distance'].std())
```

```
3.574848361859646
1.99557191743063
```

Now you can make a more intuitive interpretation: for every 3.57 miles traveled, the fare increased by a mean of \$7.13. Or, for every 1 mile traveled, the fare increased by a mean of \$2.00.

[62]:
```
print(X_train['mean_duration'].std())

print(2.811456 / X_train['mean_duration'].std())
```

```
10.10020090816442
0.27835644316019315
```

For every 10.10 minutes traveled, the fare increased by a mean of \$2.81. Or, for every 1 minute traveled, the fare increased by a mean of \$0.28.

4.4.4 Task 9d. Conclusion

1. What are the key takeaways from this notebook?
2. What results can be presented from this notebook?

What are the key takeaways from this notebook?

1. Multiple linear regression is a powerful tool to estimate a dependent continous variable from several independent variables.
2. You can discuss meeting linear regression assumptions, and you can present the MAE and RMSE scores obtained from the model.

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.