Module 5 Challenge

Start Assignment

Due Wednesday by 11:59pm **Points** 100 **Submitting** a text entry box or a website url

Background

V. Isualize has given you and Omar a brand-new assignment. Using your Python skills and knowledge of Pandas, you'll create a summary DataFrame of the ride-sharing data by city type. Then, using Pandas and Matplotlib, you'll create a multiple-line graph that shows the total weekly fares for each city type. Finally, you'll submit a written report that summarizes how the data differs by city type and how those differences can be used by decision-makers at PyBer.

What You're Creating

This new assignment consists of two technical analysis deliverables and a written report to present your results. You will submit the following:

- Deliverable 1: A ride-sharing summary DataFrame by city type
- Deliverable 2: A multiple-line chart of total fares for each city type
- Deliverable 3: A written report for the PyBer analysis (README.md)

Files

Use the following link to download the Challenge starter code.

Download challenge starter code (https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/v2/module_5/PyBer_Challenge_starter_code.ipynb)

Deliverable 1: A ride-sharing summary DataFrame by city type (35 points)

Deliverable 1 Instructions

Using the Pandas `groupby()` function with the `count()` and `sum()` methods on PyBer DataFrame columns, get the total number of rides, total number of drivers, and the total fares for each city type. Then, calculate the average fare per ride and average fare per driver for each city type. Finally, add this data to a new DataFrame, then format the columns.

REWIND

For this deliverable, you've already done the following in this module:

- **Lesson 5.3.2:** Use the `groupby()` function
- **Lesson 5.3.2:** Use the `count()` method
- **Lesson 5.4.1:** Format numbers and strings
- **Lesson 5.5.1:** Use the `sum()` method

1. Download the `PyBer_Challenge_starter_code.ipynb` file into your PyBer_Analysis folder and rename it `PyBer_Challenge.ipynb`.
2. Use the step-by-step instructions below to add code where indicated by the numbered comments in the starter code file.
3. In Step 1, use the `groupby()` function to create a Series of data that has the type of city as the index, then apply the `count()` method to the "ride_id" column.
4. In Step 2, use the `groupby()` function to create a Series of data that has the type of city as the index, then apply the `sum()` method to the "driver_count" column.
5. In Step 3, use the `groupby()` function to create a Series of data that has the type of city as the index, then apply the `sum()` method to the "fare" column.
6. In Step 4, calculate the average fare per ride by city type by dividing the sum of all the fares by the total rides.
7. In Step 5, calculate the average fare per driver by city type by dividing the sum of all the fares by the total drivers.
8. In Step 6, create a PyBer summary DataFrame with all the data gathered from Steps 1-5, using the column names shown below:.

type	Total Rides	Total Drivers	Total Fares	Average Fare per Ride	Average Fare per Driver
Rural	125	78	4327.93	34.623440	55.486282
Suburban	625	490	19356.33	30.970128	39.502714
Urban	1625	2405	39854.38	24.525772	16.571468

9. In Step 7, use the provided code snippet to remove the index name ("type") from the PyBer summary DataFrame.
10. In Step 8, format the columns of the Pyber summary DataFrame to look like this:

	Total Rides	Total Drivers	Total Fares	Average Fare per Ride	Average Fare per Driver
Rural	125	78	$4,327.93	$34.62	$55.49
Suburban	625	490	$19,356.33	$30.97	$39.50
Urban	1,625	2,405	$39,854.38	$24.53	$16.57

Deliverable 1 Requirements

You will earn a perfect score for Deliverable 1 by completing all requirements below:

- The total number of rides for each city type is retrieved. **(5 pt)**
- The total number of drivers for each city type is retrieved. **(5 pt)**
- The sum of the fares for each city type is retrieved. **(5 pt)**
- The average fare per ride for each city type is calculated. **(5 pt)**
- The average fare per driver for each city type is calculated. **(5 pt)**
- A PyBer summary DataFrame is created. **(5 pt)**
- The PyBer summary DataFrame is formatted as shown in the example. **(5 pt)**

Deliverable 2: A multiple-line chart of total fares for each city type (45 points)

Deliverable 2 Instructions

Using your Pandas skills and two new functions, `pivot()` and `resample()`, create a multiple-line graph that shows the total fares for each week by city type.

REWIND

For this deliverable, you've already done the following in this module:

- **Lesson 5.1.3:** Create a line chart using the object-oriented interface method
- **Lesson 5.1.4:** Annotate charts

- **Lesson 5.1.10:** Graph a Pandas DataFrame
- **Lesson 5.3.2:** Use the `groupby()` function
- **Lesson 5.5.1:** Use the `sum()` method

Use the step-by-step instructions below to add code where indicated by the numbered comments in the starter code file:

1. In Step 1, create a new DataFrame with multiple indices using the `groupby()` function on the "type" and "date" columns of the `pyber_data_df` DataFrame, then apply the `sum()` method on the "fare" column to show the total fare amount for each date.
2. In Step 2, use the provided code snippet to reset the index. This is needed to use the `pivot()` function in the next step (Step 3).
3. In Step 3, use the `pivot()` function to convert the DataFrame from the previous step so that the index is the "date," each column is a city "type," and the values are the "fare."
    - After this step, you'll see that each cell has the total fare for the date and time, as shown in the following image.

Note: In cells where there is no fare to be summed for that row, the cell will be filled with NaNs.

type date	Rural	Suburban	Urban
2019-01-01 00:08:16	NaN	NaN	37.91
2019-01-01 00:46:46	NaN	47.74	NaN
2019-01-01 02:07:24	NaN	24.07	NaN
2019-01-01 03:46:50	NaN	NaN	7.57
2019-01-01 05:23:21	NaN	NaN	10.75
2019-01-01 09:45:36	43.69	NaN	NaN
2019-01-01 12:32:48	NaN	25.56	NaN
2019-01-01 14:40:14	NaN	NaN	5.42
2019-01-01 14:42:25	NaN	NaN	12.31
2019-01-01 14:52:06	NaN	31.15	NaN

If you'd like a hint on how to create a pivot table from a DataFrame, that's totally okay. If not, that's great too. You can always revisit this later if you change your mind.

HIDE HINT

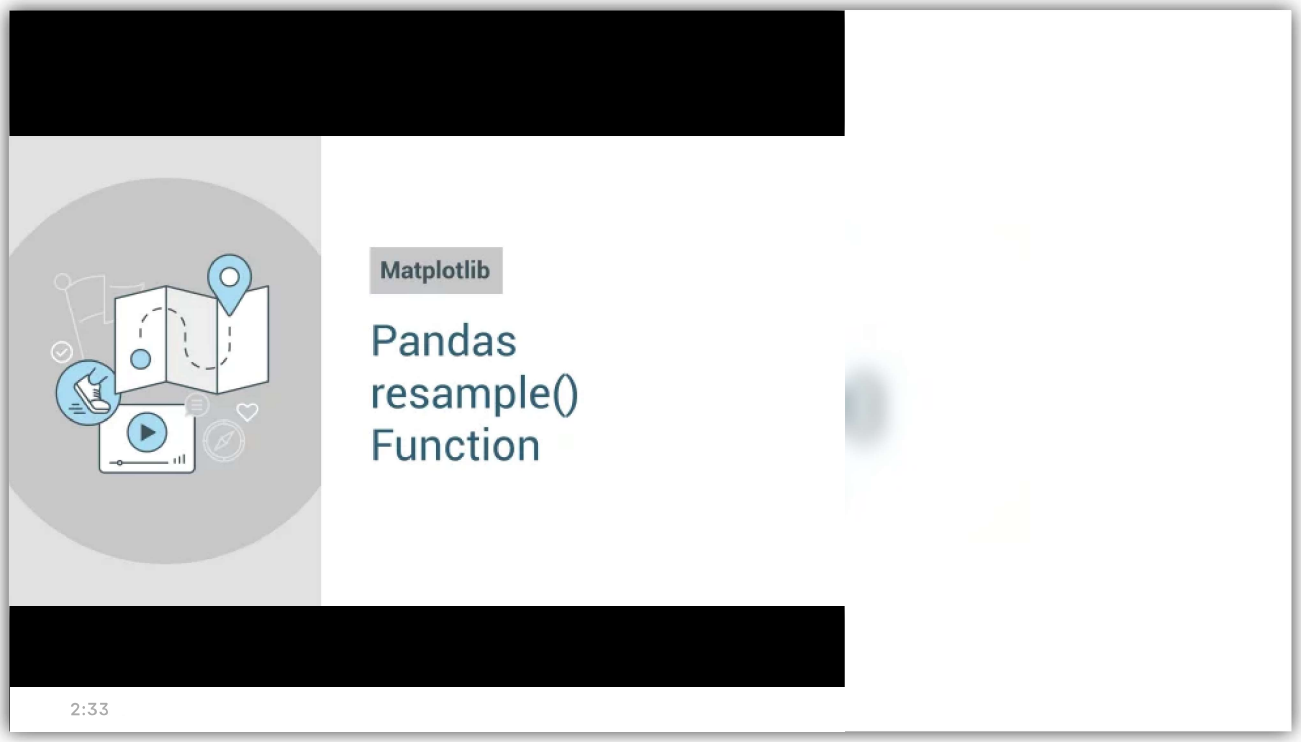
The `pivot()` function will return a reshaped DataFrame that is organized by given index and column values. In addition to this **Pandas documentation (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.pivot.html)**, watch the video for a detailed overview of how to use the `pivot()` function on a DataFrame.

4. In Step 4, create a new DataFrame by using the loc method on the following date range: 2019-01-01 through 2019-04-28.
5. In Step 5, use the provided code snippet to reset the index of the DataFrame from the previous step (Step 4) to a datetime data type. This is necessary to use the `resample()` method in Step 7.
6. In Step 6, use the provided code snippet, `df.info()`, to check that the "date" is a datetime data type.
7. In Step 7, create a new DataFrame by applying the `resample()` function to the DataFrame you modified in Step 5. Resample the data in weekly bins, then apply the `sum()` method to get the total fares for each week.

If you'd like a hint on how to create a pivot table from a DataFrame, that's totally okay. If not, that's great too. You can always revisit this later if you change your mind.

HIDE HINT

The `resample()` function will resample time-series data, but the data to be resampled must have a datetime-like index, such as `DatetimeIndex`, `PeriodIndex`, or `TimedeltaIndex`. In addition to this **Pandas documentation (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.resample.html)**, watch the video for a detailed overview of how to use the `resample()` function on a DataFrame.

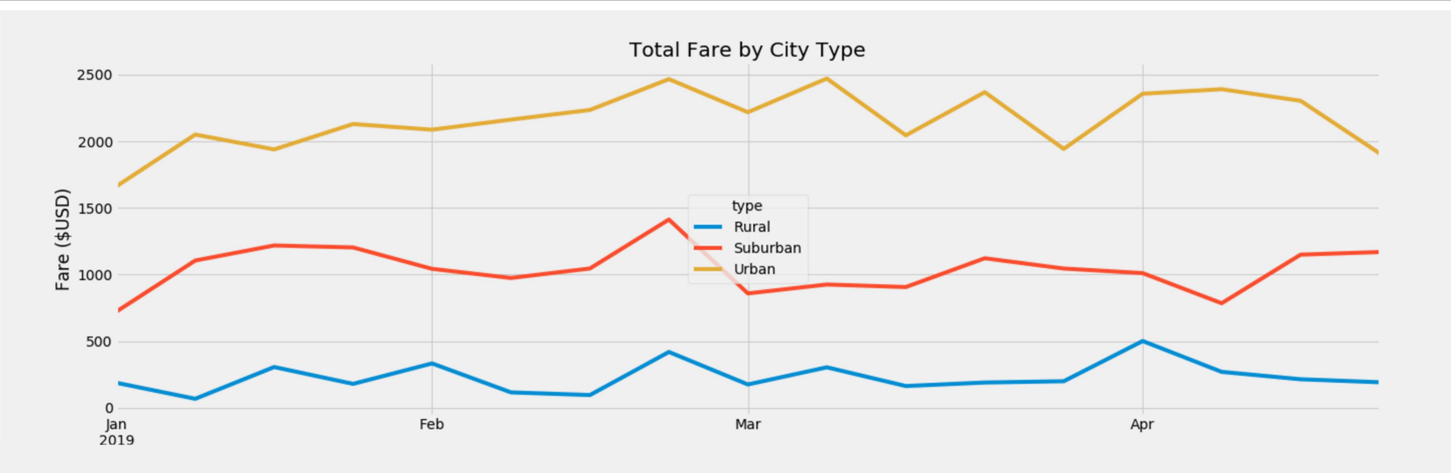


- After creating the resampled DataFrame in Step 7, confirm that your DataFrame looks like this:

type date	Rural	Suburban	Urban
2019-01-06	187.92	721.60	1661.68
2019-01-13	67.65	1105.13	2050.43
2019-01-20	306.00	1218.20	1939.02
2019-01-27	179.69	1203.28	2129.51
2019-02-03	333.08	1042.79	2086.94
2019-02-10	115.80	974.34	2162.64
2019-02-17	95.82	1045.50	2235.07
2019-02-24	419.06	1412.74	2466.29
2019-03-03	175.14	858.46	2218.20
2019-03-10	303.94	925.27	2470.93

8. Finally, in Step 8, graph the resampled DataFrame from Step 7 using the object-oriented interface method and the `df.plot()` method, as well as the Matplotlib `"fivethirtyeight"` graph style code snippet provided in the starter code. Annotate the y-axis label and the title, then use the appropriate code to save the figure as `PyBer_fare_summary.png` in your "analysis" folder.

    - Confirm that your multiple-line chart looks like the following image, where each week is a peak or dip in the line graphs.



Deliverable 2 Requirements

You will earn a perfect score for Deliverable 2 by completing all requirements below:

- A DataFrame was created using the `groupby()` function on the "type" and "date" columns, and the `sum()` method is applied on the "fare" column to show the total fare amount for each date and time. **(10 pt)**
- A DataFrame was created using the `pivot()` function where the index is the "date," the columns are the city "type," and the values are the "fare." **(10 pt)**
- A DataFrame was created using the `loc` method on the date range: 2019-01-01 through 2019-04-28. **(5 pt)**
- A DataFrame was created using the `resample()` function in weekly bins and shows the sum of the fares for each week. **(10 pt)**
- An annotated chart showing the total fares by city type is created and saved to the "analysis" folder. **(10 pt)**

Deliverable 3: A written report for the PyBer analysis (20 points)

Deliverable 3 Instructions

Use your repository README file to write your analysis of how to address any disparities in the ride-sharing data among the city types.

The analysis should contain the following:

1. **Overview of the analysis:** Explain the purpose of the new analysis.
2. **Results:** Using images from the summary DataFrame and multiple-line chart, describe the differences in ride-sharing data among the different city types.
3. **Summary:** Based on the results, provide three business recommendations to the CEO for addressing any disparities among the city types.

Deliverable 3 Requirements

Structure, Organization, and Formatting (6 points)

The written analysis has the following structure, organization, and formatting:

- There is a title, and there are multiple sections. **(2 pt)**
- Each section has a heading and subheading. **(2 pt)**
- Links to images are working and displayed correctly. **(2 pt)**

Analysis (14 points)

The written analysis has the following:

1. Overview of the analysis:
    - The purpose of the new analysis is well defined. **(3 pt)**

2. Results:

- There is a description of the differences in ride-sharing data among the different city types. Ride-sharing data include the total rides, total drivers, total fares, average fare per ride and driver, and total fare by city type. **(7 pt)**

3. Summary:
    - There is a statement summarizing three business recommendations to the CEO for addressing any disparities among the city types. **(4 pt)**

Submission

Once you're ready to submit, make sure to check your work against the rubric to ensure you are meeting the requirements for this Challenge one final time. It's easy to overlook items when you're in the zone!

As a reminder, the deliverables for this Challenge are as follows:

- Deliverable 1: A ride-sharing summary DataFrame by city type.
- Deliverable 2: A multiple-line chart of total fares for each city type.
- Deliverable 3: A written report for the PyBer analysis (README.md).

Upload the following to your PyBer_Analysis GitHub repository:

1. The `PyBer_Challenge.ipynb` file.
    - The results need to be kept populated in the `PyBer_Challenge.ipynb` file. Do not clear the output from the `PyBer_Challenge.ipynb` file before uploading to GitHub.
2. The "Resources" folder with the `city_data.csv` and `ride_data.csv` files.
3. The "analysis" folder with the `PyBer_fare_summary.png`.
4. An updated README.md that has your written analysis.

To submit your challenge assignment for grading in Bootcamp Spot, click Start Assignment, click the Website URL tab, then provide the URL of your PyBer_Analysis GitHub repository, and then click Submit. Comments are disabled for graded submissions in BootCampSpot. If you have questions about your feedback, please notify your instructional staff or the Student Success Manager. If you would like to resubmit your work for an improved grade, you can use the **Re-Submit Assignment** button to upload new links. You may resubmit up to 3 times for a total of 4 submissions.

IMPORTANT

Once you receive feedback on your Challenge, make any suggested updates or adjustments to your work. Then, add this week's Challenge to your professional portfolio.

NOTE

You are allowed to miss up to two Challenge assignments and still earn your certificate. If you complete all Challenge assignments, your lowest two grades will be dropped. If you wish to skip this assignment, click Next, and move on to the next Module.