Module 4 Challenge

Start Assignment

* **Due** Thursday by 23:59

* **Points** 100

* **Submitting** a text entry box or a website url

In this assignment, you’ll create and manipulate Pandas DataFrames to analyse school and standardised test data.

**Background**

You are the new Chief Data Scientist for your local government area. In this capacity, you'll be helping the school board and mayor make strategic decisions regarding future school budgets and priorities.

As a first task, you've been asked to analyse the area-wide standardised test results. You'll be given access to every student's maths and reading scores, as well as various information on the schools they attend. Your task is to aggregate the data to showcase obvious trends in school performance.

**Before You Begin**

1. Create a new repository for this project called pandas-challenge. **Do not add this assignment to an existing repository**.
2. Clone the new repository to your computer.
3. Inside your local Git repository, create a folder for this assignment and name it PyCitySchools.
4. Add your Jupyter notebook to this folder. This will be the main script to run for analysis.
5. Push these changes to GitHub or GitLab.

**Files**

Download the following files to help you get started:

[Module 4 Challenge filesLinks to an external site.](https://static.bc-edx.com/data/dla-1-2/m4/lms/starter/Starter_Code.zip)

**Instructions**

Using Pandas and Jupyter Notebook, create a report that includes the following data. Your report must include a written description of at least two observable trends based on the data.

**Hint:** Check out the sample solution called PyCitySchools\_starter.ipynb located in the .zip file to review the desired format for this assignment.

**Local Government Area (LGA) Summary**

Perform the necessary calculations and then create a high-level snapshot of the local government area's key metrics in a DataFrame.

Include the following:

* Total number of unique schools
* Total students
* Total budget
* Average maths score
* Average reading score
* % passing maths (the percentage of students who passed maths)
* % passing reading (the percentage of students who passed reading)
* % overall passing (the percentage of students who passed maths AND reading)

**Note:** A passing grade is 50 or higher.

**School Summary**

Perform the necessary calculations and then create a DataFrame that summarises key metrics about each school.

Include the following:

* School name
* School type
* Total students
* Total school budget
* Per student budget
* Average maths score
* Average reading score
* % passing maths (the percentage of students who passed maths)
* % passing reading (the percentage of students who passed reading)
* % overall passing (the percentage of students who passed maths AND reading)

**Highest-Performing Schools (by % Overall Passing)**

Sort the schools by % Overall Passing in descending order and display the top 5 rows.

Save the results in a DataFrame called "top\_schools".

**Lowest-Performing Schools (by % Overall Passing)**

Sort the schools by % Overall Passing in ascending order and display the top 5 rows.

Save the results in a DataFrame called "bottom\_schools".

**Maths Scores by Year**

Perform the necessary calculations to create a DataFrame that lists the average maths score for students of each year level (9, 10, 11, 12) at each school.

**Reading Scores by Year**

Create a DataFrame that lists the average reading score for students of each year level (9, 10, 11, 12) at each school.

**Scores by School Spending**

Create a table that breaks down school performance based on average spending ranges (per student).

Use the code provided below to create four bins with reasonable cutoff values to group school spending.

spending\_bins = [0, 585, 630, 645, 680]

labels = ["<$585", "$585-630", "$630-645", "$645-680"]

Use pd.cut to categorise spending based on the bins.

Use the following code to then calculate mean scores per spending range.

spending\_math\_scores = school\_spending\_df.groupby(["Spending Ranges (Per Student)"])["Average Math Score"].mean()

spending\_reading\_scores = school\_spending\_df.groupby(["Spending Ranges (Per Student)"])["Average Reading Score"].mean()

spending\_passing\_math = school\_spending\_df.groupby(["Spending Ranges (Per Student)"])["% Passing Math"].mean()

spending\_passing\_reading = school\_spending\_df.groupby(["Spending Ranges (Per Student)"])["% Passing Reading"].mean()

overall\_passing\_spending = school\_spending\_df.groupby(["Spending Ranges (Per Student)"])["% Overall Passing"].mean()

Use the scores above to create a DataFrame called spending\_summary.

Include the following metrics in the table:

* Average maths score
* Average reading score
* % passing maths (the percentage of students who passed maths)
* % passing reading (the percentage of students who passed reading)
* % overall passing (the percentage of students who passed maths AND reading)

**Scores by School Size**

Use the following code to bin the per\_school\_summary.

size\_bins = [0, 1000, 2000, 5000]

labels = ["Small (<1000)", "Medium (1000-2000)", "Large (2000-5000)"]

Use pd.cut on the "Total Students" column of the per\_school\_summary DataFrame.

Create a DataFrame called size\_summary that breaks down school performance based on school size (small, medium, or large).

**Scores by School Type**

Use the per\_school\_summary DataFrame from the previous step to create a new DataFrame called type\_summary.

This new DataFrame should show school performance based on the "School Type".

**Requirements**

**Local Government Area Summary (20 points)**

* Calculate the total number of unique schools (2 points)
* Calculate the total number of students (2 points)
* Calculate the total budget (2 points)
* Calculate the average (mean) maths score (2 points)
* Calculate the average (mean) reading score (2 points)
* Use the code provided to calculate the percentage of students who passed maths (2 points)
* Calculate the percentage of students who passed reading (2 points)
* Use the code provided to calculate the percentage of students that passed both maths and reading (2 points)
* Create a new DataFrame for the above calculations called area\_summary (4 points)

**School Summary (20 points)**

* Use the code provided to select the school type (2 points)
* Calculate the total student count (2 points)
* Use the code provided to calculate the per capita spending (2 points)
* Calculate the average test scores (2 points)
* Calculate the number of schools with maths scores of 50 or higher (2 points)
* Calculate the number of schools with reading scores of 50 or higher (2 points)
* Use the provided code to calculate the schools that passed both maths and reading with scores of 50 or higher (2 points)
* Use the provided code to calculate the passing rates (2 points)
* Create a new DataFrame for the above calculations called per\_school\_summary (4 points)

**Highest-Performing Schools by Percentage of Overall Passing (5 points)**

* Sort the schools in the per\_school\_summary DataFrame by % Overall Passing in descending order (2 points)
* Save the results to a DataFrame called top\_schools (2 points)
* Display the first 5 rows (1 point)

**Lowest-Performing Schools by Percentage of Overall Passing (5 points)**

* Sort the schools in the per\_school\_summary DataFrame by % Overall Passing in ascending order (2 points)
* Save the results to a DataFrame called bottom\_schools (2 points)
* Display the first 5 rows (1 point)

**Maths Scores by Year (10 points)**

* Use the code provided to separate the data by year (1 points)
* Group by "school\_name" and take the mean of each (4 points)
* Use the code to select only the maths\_score (1 points)
* Combine each of the scores above into single DataFrame called maths\_scores\_by\_year (4 points)

**Reading Scores by Year (10 points)**

* Use the code provided to separate the data by year (1 points)
* Group by "school\_name" and take the mean of each (4 points)
* Use the code to select only the reading\_score (1 points)
* Combine each of the scores above into single DataFrame called reading\_scores\_by\_year (4 points)

**Scores by School Spending (5 points)**

* Use pd.cut with the provided code to bin the data by the spending ranges (2 points)
* Use the code provided to calculate the averages (1 points)
* Create the spending\_summary DataFrame using the binned and averaged spending data (2 points)

**Scores by School Size (5 points)**

* Use pd.cut with the provided code to bin the data by the school sizes (2 points)
* Use the code provided to calculate the averages (1 points)
* Create the size\_summary DataFrame using the binned and averaged size data (2 points)

**Scores by School Type (5 points)**

* Group the per\_school\_summary DataFrame by "School Type" and average the results (2 points)
* Use the code provided to select the new column data (1 point)
* Create a new DataFrame called type\_summary that uses the new column data (2 points)

**Written Report (15 points)**

To receive all points, the written report presents a cohesive written analysis that:

* Summarises the analysis (5 points)
* Draws two correct conclusions or comparisons from the calculations (10 points)

**Grading**

This assignment will be evaluated against the requirements and assigned a grade according to the following table:

| **Grade** | **Points** |
| --- | --- |
| A (+/-) | 90+ |
| B (+/-) | 80–89 |
| C (+/-) | 70–79 |
| D (+/-) | 60–69 |
| F (+/-) | < 60 |

**Submission**

To submit your Challenge assignment, click Submit, and then provide the URL of your GitHub repository for grading.

**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.

Comments are disabled for graded submissions in Bootcamp Spot. If you have questions about your feedback, please notify your instructional staff or your Student Success Advisor. If you would like to resubmit your work for an additional review, you can use the Resubmit Assignment button to upload new links. You may resubmit up to three times for a total of four submissions.

**IMPORTANT**

**It is your responsibility to include a note in the README section of your repo specifying code source and its location within your repo**. This applies if you have worked with a peer on an assignment, used code in which you did not author or create sourced from a forum such as Stack Overflow, or you received code outside curriculum content from support staff such as an Instructor, TA, Tutor, or Learning Assistant. This will provide visibility to grading staff of your circumstance in order to avoid flagging your work as plagiarized.

If you are struggling with a Challenge or any aspect of the curriculum, please remember that there are student support services available for you:

1. Office hours facilitated by your TA(s)
2. [Tutoring GuidelinesLinks to an external site.](https://docs.google.com/document/d/1hTldEfWhX21B_Vz9ZentkPeziu4pPfnwiZbwQB27E90/edit?usp=sharing) (schedule a session in the "Tutor Sessions" section of Bootcampspot)
3. Ask the class Slack channel/get peer support
4. AskBCS Learning Assistants

**References**

Data generated by [Mockaroo, LLCLinks to an external site.](https://mockaroo.com/), (2022). Realistic Data Generator. Data for this dataset was generated by edX Boot Camps LLC, and is intended for educational purposes only.

[Previous](https://bootcampspot.instructure.com/courses/4525/modules/items/1134073)[Next](https://bootcampspot.instructure.com/courses/4525/modules/items/1134075)