**Project Goals**

You will work to write several functions that investigate a dataset of *Jeopardy!* questions and answers. Filter the dataset for topics that you’re interested in, compute the average difficulty of those questions, and train to become the next Jeopardy champion!

**Setup Instructions**

If you choose to do this project on your computer instead of Codecademy, you can download what you’ll need by clicking the “Download” button below. **Note that if you choose to download the project, we will provide you with a slightly bigger dataset**. If you need help setting up your computer, be sure to check out our [setup guide](https://www.codecademy.com/articles/install-python3).

[Download](https://content.codecademy.com/PRO/independent-practice-projects/jeopardy/jeopardy_starting.zip)

**Tasks**

**2/9 Complete**

Mark the tasks as complete by checking them off

**Prerequisites**

**1.**

In order to complete this project, you should have completed the Pandas lessons in the [Analyze Data with Python Skill Path](https://www.codecademy.com/learn/paths/analyze-data-with-python" \t "_blank). You can also find those lessons in the [Data Analysis with Pandas course](https://www.codecademy.com/learn/data-processing-pandas) or the [Data Scientist Career Path](https://www.codecademy.com/learn/paths/data-science/).

Finally, the [Practical Data Cleaning](https://www.codecademy.com/learn/practical-data-cleaning) course may also be helpful.

**Project Requirements**

**2.**

We’ve provided a csv file containing data about the game show *Jeopardy!* in a file named jeopardy.csv. Load the data into a DataFrame and investigate its contents. Try to print out specific columns.

Note that in order to make this project as “real-world” as possible, we haven’t modified the data at all — we’re giving it to you exactly how we found it. As a result, this data isn’t as “clean” as the datasets you normally find on Codecademy. More specifically, there’s something odd about the column names. After you figure out the problem with the column names, you may want to rename them to make your life easier the rest of the project.

In order to disply the full contents of a column, we’ve added this line of code to the top of your file:

pd.set\_option('display.max\_colwidth', -1)

Hint

The column names all have a leading space. For example, the column that contains the answers is " Answer". Notice the space at the front of the column name.

**3.**

Write a function that filters the dataset for questions that contains all of the words in a list of words. For example, when the list ["King", "England"] was passed to our function, the function returned a DataFrame of 152 rows. Every row had the strings "King" and "England" somewhere in its " Question".

Note that in this example, we found 152 rows by filtering the *entire* dataset. You can download the entire dataset at the start or end of this project. The dataset used on Codecademy is only a fraction of the dataset so you won’t find as many rows.

Test your function by printing out the column containing the question of each row of the dataset.

Hint

You could use a lambda function to do this. Our lambda function uses the [all function](https://docs.python.org/3/library/functions.html#all) to check if all elements of the input list are in the question.

**4.**

Test your original function with a few different sets of words to try to find some ways your function breaks. Edit your function so it is more robust.

For example, think about capitalization. We probably want to find questions that contain the word "King" *or* "king".

You may also want to check to make sure you don’t find rows that contain substrings of your given words. For example, our function found a question that didn’t contain the word "king", however it did contain the word "viking" — it found the "king" inside "viking". Note that this also comes with some drawbacks — you would no longer find questions that contained words like "England's".

Hint

We fixed the capitalization problem by lowercasing the question string and the words in the input array.

**5.**

We may want to eventually compute aggregate statistics, like .mean() on the " Value" column. But right now, the values in that column are strings. Convert the " Value" column to floats. If you’d like to, you can create a new column with the float values.

Now that you can filter the dataset of question, use your new column that contains the float values of each question to find the “difficulty” of certain topics. For example, what is the average value of questions that contain the word "King"?

Make sure to use the dataset that contains the float values as the dataset you use in your filtering function.

Hint

In order to convert these values, you’ll have to remove the dollar sign and commas from the string and then cast that string as a float. Be careful — some of the values in that column are "None" (those rows represent Final Jeopardy questions). You’ll have to do something different for those rows — if you try to remove the non-existent dollar sign and commas from those rows, you’ll get an error!

We used the [replace](https://docs.python.org/3/library/stdtypes.html#str.replace) method to remove the dollar sign and commas.

To find the average value of the new column, we used the [mean](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.mean.html) method.

**6.**

Write a function that returns the count of the unique answers to all of the questions in a dataset. For example, after filtering the entire dataset to only questions containing the word "King", we could then find all of the unique answers to those questions. The answer “Henry VIII” appeared 3 times and was the most common answer.

Hint

You could write your own function that does this, but the [value\_counts](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.value_counts.html" \t "_blank) method can help!

**7.**

Explore from here! This is an incredibly rich dataset, and there are so many interesting things to discover. There are a few columns that we haven’t even started looking at yet. Here are some ideas on ways to continue working with this data:

* Investigate the ways in which questions change over time by filtering by the date. How many questions from the 90s use the word "Computer" compared to questions from the 2000s?
* Is there a connection between the round and the category? Are you more likely to find certain categories, like "Literature" in Single Jeopardy or Double Jeopardy?
* Build a system to quiz yourself. Grab random questions, and use the [input](https://docs.python.org/3/library/functions.html#input) function to get a response from the user. Check to see if that response was right or wrong. Note that you can’t do this on the Codecademy platform — to do this, download the data, and write and run the code on your own computer!

**Solution**

**8.**

Compare your program to our [sample solution code](https://content.codecademy.com/PRO/independent-practice-projects/jeopardy/jeopardy_solution.zip) - remember, that your program might look different from ours (and probably will) and that’s okay!

**Solution**

**9.**

Great work! Visit [our forums](https://discuss.codecademy.com/t/this-is-jeopardy-challenge-project-python-pandas/462365) to compare your project to our sample solution code. You can also learn how to host your own solution on GitHub so you can share it with other learners! Your solution might look different from ours, and that’s okay! There are multiple ways to solve these projects, and you’ll learn more by seeing others’ code.