In this last project you will develop an interactive data-driven web-based Python application that tells a story with real-world data. You will show your mastery of many coding concepts as you interact with real-world data. You will use Pandas and NumPy modules for managing and interacting with data, MatPlotLib, Pandas, or other charting packages for creating charts and graphs, PyDeck (or other mapping packages) for maps, and the Streamlit.io package for creating interactive web applications using Python.

The data sets we will be using this semester all come from Analyze Boston (<http://data.boston.gov>), the city of Boston’s open data hub.

|  |  |
| --- | --- |
| Name | Description |
| [Boston Parking Meters](https://data.boston.gov/dataset/parking-meters) | Looking for a place to park in Boston? This data set shows where to park in downtown Boston on each block, and hours of meter operation. |

The links for each data set provide background information and sources for the data. Please read them, as they often contain data dictionaries describing the fields or columns in each data table. [Download the data you are using from here](https://bentleyedu-my.sharepoint.com/:f:/g/personal/mfrydenberg_bentley_edu/EugKnDcvn_9OhvfHuOgFiQUBwHkQpQ3EhRlbAFEbXQp3Uw?e=SvhJpO), rather than from Analyze Boston website, as the data files provided for this project in some cases are sampled or cleaned versions of the original data files. (If there’s a file that ends with 7000\_sample, use that one. It means the original file from Analyze Boston is much larger, and a random sample of 7000 records is provided.)

# Details

## Part 1.   Design – What do you want to show and tell with your data?

The purpose of the design phase is to get you thinking about what you might do before you start coding. Identify at least three different queries or questions you can ask about your data set. Try to phrase your questions so that they can have a parameter which can come from user input.

For example: (and these queries don’t match the data sets you are given but are here to inspire you!)

* What’s the cost of the most expensive <house\_type> in <city>?
* Find all the apartments in <city> that rent for under <amount>.

Then think about the interactive widgets from Streamlit that you can bring to your application to obtain user input. For example: you can use a numeric slider to have the user enter a monthly rental amount, or display a list of cities in a dropdown list.

Next, think about how you will visually present the data or query results using charts, graphs, tables, or maps.

Be sure your web pages and visualizations are "user friendly" and as "polished" as possible. Be sure to label controls requiring user interaction, make sure your charts have titles, legends or explanations that would be helpful to the user. Think about how the user will navigate from one part of your site to another.

Feel free to add to your project as you explore Pandas and Streamlit capabilities and find cool ways to implement new or additional features. **Part of your grade will be a "complexity/originality" score.** If you use a module or do something cool that we may not have discussed in class or implement a lot more than the minimum requirements, you will receive a higher complexity score.

A complexity score of 1 means you implemented the minimum requirements for this project. A complexity score of zero means you didn’t meet the requirements.

## Part 2.  Code.

Create your Python application with a Streamlit UI and several visualizations. Write “good” Python code that incorporates the required features.

Create charts and graphs of different types with custom legends, axis labels, tick marks, colors, other features), and at least one map showing locations and data based on latitude and longitude.  Your chart should tell a story, so be sure elements are labelled appropriately, and add any narrative that will help the reader understand your visualizations and to cue the reader about which values to specify, and the purpose of each chart or graph. You may wish to add a few sentences explaining each chart as a paragraph of text on the screen.

You might also use pandas to create summary report based on the data itself (max/min values, relationships between columns, etc).

See the [documentation](https://docs.streamlit.io/library/api-reference) for how to use different Streamlit features. You might make use of [sidebars](https://docs.streamlit.io/library/api-reference/layout/st.sidebar) to place your widgets, [multi-page applications](https://docs.streamlit.io/library/get-started/multipage-apps/create-a-multipage-app), or [caching](https://docs.streamlit.io/library/advanced-features/caching) to improve performance.

Read the documentation for [PyDeck](https://deckgl.readthedocs.io/en/latest/index.html) Maps. Our examples of maps in class were PyDeck’s Scatterplot Layer or IconLayer, but PyDeck support several other styles such as Text and Heatmaps. Have a look.

To explore another chart library, consider [Seaborn](https://seaborn.pydata.org/examples/index.html) charts which have additional chart types and customization options. You might also look at [Folium](https://python-visualization.github.io/folium/) maps (here’s a [simple tutorial](https://realpython.com/python-folium-web-maps-from-data/)) if you’d like to play with a different mapping library.

If your project contains more than one Python code file (ie, one or more Python code files and images), create a zip file containing all your project files and submit it. Also be sure to **include the data files** you used as part of your submission.

## Requirements

As you write your program, be sure to include code that demonstrates each of these items. Each contributes to your project grade (see the rubric below).

**Python Features: (at least four)**

* A function with two or more parameters, one of which has a default value
* A function that returns more than one value
* A function that you call at least two different places in your program
* A list comprehension
* A dictionary

**Quality of Code:**

* Is your code easy to read, well documented, understandable/concise, accurate, efficient, maintainable?
* Do you use python language features effectively?

**Streamlit Features:**

* At least three Streamlit different [widgets](https://docs.streamlit.io/library/api-reference/widgets) (sliders, drop downs, multi-selects, text box, etc)
* Page design features (sidebar, fonts, colors, images, navigation)

**Visualizations**: (at least four, at least one of which is a map)

* At least three different charts with titles, colors, labels, legends, as appropriate
* At least one detailed map (st.map will only get you partial credit) – for full credit, include dots, icons, text that appears when hovering over a marker, or other map features

**Data Analytics Capabilities: (at least four)**

* Sorting data in ascending or descending order, by one or more columns,
* Top largest or smallest values of a column
* Filtering data by one condition
* Filtering data by two or more conditions with AND or OR
* Analyzing data with pivot tables
* Add/drop/select/create new/group columns, frequency count, other features
* Iterating through rows of a DataFrame
* Cleaning or manipulating data

**Usual rules about writing "good" code apply:**

* Make your code as modular and easy to follow as possible.
* Include a docstring, comments, and meaningful variable names.
* If you did something "cool" in your code that you are incredibly proud of, please write a comment call attention to what you did.
* If you referred to any online articles or other information beyond class examples, please be sure to list them as references / comments in your code.
* Make sure the program runs and the output is correct.

### Documentation String

Use this documentation string at the top of your Python code file:

*"""*  
*Name: Your Name*  
*CS230: Section XXX*  
*Data: Which data set you used*  
*URL: Link to your web application on Streamlit Cloud (if posted)*

*Description:*   
  
*This program ... (a few sentences about your program and the queries and charts)*  
*"""*

## Easy Extra Credit

Post your application to Streamlit Cloud. Sign up for a free Streamlit Cloud community account at <https://streamlit.io/cloud> and follow the instructions to publish your app. Share your published link on the signup sheet. The real reason you should publish your project to Streamlit Cloud is not for extra credit, but so that you can share the link with potential employers and add it to your professional portfolio. Several former CS 230 students have commented that they shared their projects at job interviews.

Python has several modules developed by other Python programmers that we have not covered during this semester. You are welcome to explore other package or modules (especially those available for visualizing data) which are available in Python's Package Index. Use one in a meaningful way in your project for extra credit. Fully document this in the code. See <https://pypi.org/> for the complete list.

|  |  |
| --- | --- |
| **Requirement** | **Points** |
| Python Coding Features (at least 4 @ 2.5 points) | 10 |
| Code Quality | 3 |
| Streamlit Features (3 controls and other page design features) | 8 |
| Visualizations (3 different charts and one map, 4 points each)   * + - 2 points for displaying the data correctly   2 points for customization (colors, grid lines, legend, etc) | 16 |
| * + - Data Analytics Capabilities (at least 4 – sort, filter, etc) 3 points each | 12 |
| Presentation | 6 |
| Complexity:  0= Your project implements less than the minimum requirements  1 = Your project meets the minimum requirements  2 = Your project includes some complex queries, charts, or UI features, or added a small number of extra features beyond those which are required  3 = Wow! You went above and beyond in requirements, ether doing more than what is required, or by including features, modules, or packages learned independently or not described in class | 3 |
| **Total:** | 60 |
| Extra Credit: Publish to Streamlit Cloud | **3** |
| Extra Credit: Use packages or functions not covered in class | **Max 3** |