DSCI 551 – Fall 2025

Data2App: From Data Parsing, Operations, to Data-Driven Applications

Project Guideline

Motivations: Python libraries such as Pandas and json are commonly used in practice to process structured (e.g., CSV) and semi-structured data (e.g., JSON). We have seen both libraries in class far. For example, we show that Pandas has read_csv function that reads data from a CSV file into a dataframe in main memory; we have also seen how the json library loads a JSON file and converts its value (e.g., JSON object or array) into a proper Python object (e.g., dictionary or list). For example, the JSON file for a NoSQL database (such as MongoDB and Firebase Firestore database) may store JSON data in a collection. A collection contains an array of JSON objects, where each object is analogous to a row in a table.

As an example, consider two CSV files: country and language. Country CSV file stores information about country code, country name, continent, GNP, and population; language CSV file stores information about country code (USA), and language spoken in each country (e.g., English).

Pandas also provides functions for filtering (selecting a subset of rows) and projecting (selecting a subset of columns) of a dataframe, grouping a dataframe by certain attributes and computing aggregates, or joining a dataframe with another dataframe on some attributes. For example,

- Find names of countries whose GNP > 100000:
  - country[country.GNP > 100000][['Name']]
  - which answers SQL query:
    - select Name from country where GNP > 100000
- For each continent, computes the maximum GNP of countries in the continent:
  - country.groupby('Continent').agg({'GNP': 'max'})
  - SQL query:
    - select Continent, max(GNP) from country group by Continent
- Find names of countries and languages spoken in the countries
  - country.merge(countrylanguage, left_on = "Code", right_on = "CountryCode")[['Name', 'Language']]
  - SQL query:

        - Select Name, Language from country join countrylanguage on country.Code = countrylanguage.CountryCode

As another example, MongoDB provides functions for similar operations on JSON data. For example, assume that country and countrylanguage data are stored in collections, e.g., country collection may be:

[{“Name”: ”United States”, “Code”: “USA”, “GNP”: 200000, “Population”: 3000000000}, …]

And a countrylanguage collection may be:

[{“CountryCode”: “USA”, “Language”: “English”}, …]

It also provides functions for filtering, projection, grouping, aggregation, and join. Note that, the exact syntax of such functions is not important to the project. But you can find out more details later in class and here: https://www.mongodb.com/docs/manual/crud/.

Project objectives: The goals of the project are to understand what happens under the hood of these libraries and their functions, and provide you with hands-on experiences in developing and using these functions.

Project options:

- SQL:
    - Parsing: develop your own parser for CSV file, similar to read_csv. It should have basic parsing functions but does not need to be as complex as read_csv. It should support the specification of CSV file name, and column separators.
    - Functions: Filtering, projection, group by, aggregation, and join. The data may be loaded into main memory, becoming a dataframe. In class, we discussed the ideas of how to implement your own DataFrame class and how data are stored in the frame (e.g., as a dictionary where each column is stored as key-values pair where values is a Python list or numpy array). You might also want to look up __getitem__ function in Python if you want to implement []-style retrieval of data, e.g., df[‘Name’].
    - Applications: Develop an application that utilizes all the functions above. Your application may be a search app or an app that analyzes data. Your final report should clearly state how each function is used in your application. You should use real-world datasets. You can use internet resources, e.g., https://datasetsearch.research.google.com/ to find the datasets. Kaggle.com is also an option.
- NoSQL:

- Parsing: develop your own JSON parser that works like json.load(). We have provided you with a sample code, attached to this guideline, that you can take as a starting point. Note that sample code only (partially) implements the loading of JSON object. It needs to be extended to handle other types of JSON values, e.g., array, and also nested value (e.g., object inside object or array, and so on). JSON data, once loaded, should be properly represented in Python, e.g., object stored in a dictionary, array in a list, etc.
- Operations: You should implement functions for filtering, projection, group by, aggregation, and join. You can decide on the exact syntax of such functions (they do not have to be the same as that in MongoDB, for example).
- Applications: The requirements on the applications and datasets are similar to that for SQL option.

Requirements:

- One-person group: You can choose either SQL or NoSQL option described above.
- Two-person group: In addition to requirements for one-person group, you should also handle the scaling issue of datasets, that is, what happens if the dataset is too large to be entirely loaded into main memory. You may implement options similar to chunk_size in read_csv. For NoSQL, this chunk size may correspond to the number of JSON documents/objects loaded into memory at the same time.
- Three-person group: in addition to requirements for two-person group, you need to implement both SQL and NoSQL options. But it is ok to consider scaling only for one option: SQL or NoSQL.
- No groups with over 3 people are permitted.
- All members of the project group should be from the same 551 section.
- You may leverage internet resources and take them as reference for your project. However, **you should write your own codes**. Coping and pasting existing implementations from the Internet will result in a failing grade.
- You should NOT use libraries that implemented the required functions of your project. These libraries include, but are not limited to, pandas, json, csv, and libraries that provide similar functions as pd.read_csv() and json.load().

Deliverables & Deadlines: (due times, except for demos, are 11:59pm on the due date. No late submissions will be accepted.).

- Proposal (5 points, due **9/19**, Friday):

    - Your chosen options for the project. Changing your options (SQL vs. NoSQL) after proposal will not be permitted.
    - How you plan to implement the option(s) chosen. Indicate the timelines for your milestones (which week completes which component, etc.).
    - Group formation for the project and division of labor among group members.
    - **Note**: choose your group members wisely as once you form your group, the project will be a collaborative effort and your project will be graded as such. You will share the responsibility if other members fail to contribute to the project!
- Midterm progress report (5 points, due **10/17**, Friday):
    - For each item in your plan and milestones, indicate the status (pending or completed).
    - If you fail to complete a milestone, explain why.
    - Detail the contribution of each project member to the progress so far.
- Project demo (10 points, in-class, **11/24** Monday (MW section), **11/25** (Tuesday section):
    - You will give a live demo of your project on the demo day. Your demo should cover:
        - data loading and parsing
        - each required function
        - your application that uses your implemented functions
    - All members of the group should be present during live demo. Each member will be presenting the portion of the project he/she has been responsible for. Failing to present may lead us to assume that you have not contributed to the project and will be graded accordingly.
    - Failure to demo your project live will result in a failing grade of the project. Recorded presentations will **NOT** be accepted.
    - Failing to give successful demonstrations of items above will result in deduction of your implementation points (see below). No makeup demos will be permitted.
- Project implementation (70 points, due **11/23**, Sunday):
    - You should have completed the implementation of all components of your project by the demo time.
    - Your implementation will be evaluated by examining your codes and your demonstration. Grading rubrics:
        - data loading and parsing (20 points)
        - required functions (30 points)

        - your application that uses your implemented functions (20 points)
    - Submission entries will be provided for you to upload your codes on Brightspace. Details on this will be announced separately.
- Project final report (10 points, due **12/15**, Monday):
    - Final report typically runs from 5-8 pages (depending on your project group size and contents).
    - This will be a comprehensive report on your project:
        - What was your project about?
        - Describe algorithms for loading, parsing, and operations.
        - Describe your datasets and applications.
        - Describe your learning experiences.
        - Describe each member's contribution to the project.