94692 Data Science Practice - Spring 2022

UNIVERSITY OF TECHNOLOGY, SYDNEY

Group 14: Assessment Task 3

Collaborative Development of Database Explorer Web App

08 November 2022

Group 14 Members:
Huanwei Xu - 14090118
Sandipa Narayanbhai Limbachiya - 24602587
Sareem Sandeed - 24622829
Shihao Li - 12828531

Introduction:

For any data-related project, data scientists need to have a good understanding of the input datasets. They usually perform exploratory data analysis (EDA) in order to get a deep understanding of the information provided, identifying issues and limitations.

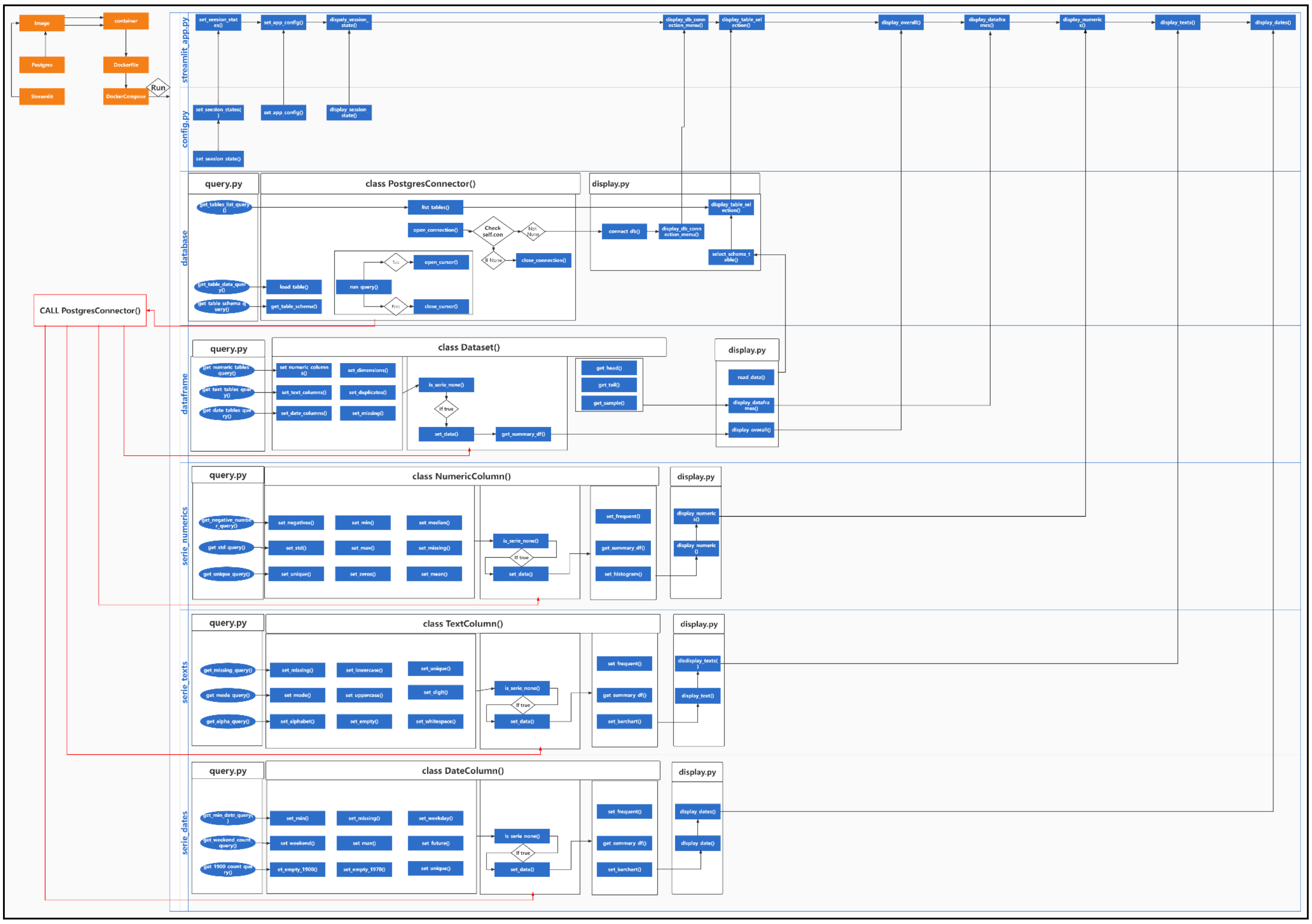
In this project, Group-14 developed an interactive web application using Streamlit and will connect to a Postgres database and perform some exploratory data analysis on selected tables. The web application needs to be containerised with Docker and will be running using python 3.8.2.

Presentation of the web application and link to the group's private GitHub repository

GitHub repository link for Group-14 is [here](#).

Video of the web application is [here](#)

Description of the design:



The structures of the web application can be split into 4 main parts:

- The series of Docker files(requirements.txt, docker file and docker-compose.yml).
- The main file(streamlit_app.py) and the configuration file (config.py)
- Database. To connect the data from Postgres.
- Dataframe, numeric series, data series, and text series. A part that contains data exploration features to display the data from various aspects(e.g., overall, type of numeric, type of date, and type of text) under different given logics.

GIT and GitHub Setup:

Steps for GIT and GitHub setup:

1. Make a GitHub account.
2. Make sure Git is installed on the machine using
   ```
   $ git --version
   ```
3. This will prompt open an installer if git is not installed. So set it up using the installer. If you have git already, it'll just show you which version of git you have installed.If still not working, use
   ```
   $ brew install git
   ```
4. Tell Git who you are
   ```
   $ git config --global user.name "YOUR_USERNAME" ##GitHub username
   $ git config --global user.email "something@xyz.com" ##GitHub email
   $ git config --global --list # To check the info you just provided
   ```
5. Generate/check your machine for existing SSH keys

6. Using the SSH protocol, you can connect and authenticate to remote servers and services. With SSH keys, you can connect to GitHub without supplying your username or password at each visit.

7. Check existing ssh key:
   ```
   $ ls -al ~/.ssh
   ```
8. If it says `ls: /Users/username/.ssh:` No such file or directory then create a new ssh key using
   `$ ssh-keygen -t rsa` keep the default location and leave the passphrase empty.

9. Copy the ssh key using the following command and add it to your GitHub account in https://github.com/-/profile/keys
   ```
   $ pbcopy < ~/.ssh/id_rsa.pub
   ```
10. Copy only the public key, never use the private key

Create a new Repository:

1. Go to GitHub and create a new project under your repository name (already existing and if not ask team leaders to create it) and use this for all data regarding this project.

2. Create different folders for different projects for better data management.
3. Whenever any code goes into production, it should be put in separate repository also outside your folder so that it is easily searchable and all information regarding a project can be directly found.

Initialize Git and clone repository:

1. Make a new folder in your desktop
   ```
   $ cd folder_name
   ```

2. To copy the repository in your local machine.
   ```
   $ git clone project_path
   ```

3. Use the git path and not the https path in your repository for this.

4. Now select the branch of the project you will be working on. Generally it will be the main branch
   ```
   $ git branch -M main
   ```

5. Go inside the repo folder before executing this to select the main branch of that repository. can this is useful as we can create multiple branches within the same repository for different versions like development , prod etc.

Create files:

1. Now do whatever you want in this folder. Add code, files data etc. This will now be your folder and everything should be done inside this folder.
2. Add files to Staging area
3. Now that you have created some files, you want to add these files to the repository on GitHub.
4. In the folder, you can check the status of the files by using:
   ```
   $ git status
   ```

5. The files in red are new or modified and need to add to tracking. For tracking these files:
   ```
   $ git add .   # Adds all the files in the local repository and stages them for commit
   ```

6. OR if you want to add a specific file
   ```
   $ git add README.md  # To add a specific file
   ```

7. Now before we commit, we need to check the status of the files that we have added.
   ```
   $ git status
   ```

8. Now, the added files will be in green colour, meaning that they are ready to be added to the repository.

Git Commit

1. Now before pushing to the repository, we need to commit towards pushing. For that:
   ```
   $ git commit -m "Commit message"
   ```

2. The message in the " " is given so that the other users can read the message and see what changes you made. This should be clearly written and precise to easily track changes.

Git push

1. This will push all the changes to the repository in the main branch.
   ```
   $ git push -u origin main
   ```

2. This will push all the changes to the repository in the selected branch.
   ```
   $ git push -u origin branch_name
   ```

Instructions to set up and launch the web application:

To run the web application, 3 files should be created and set up in advance:

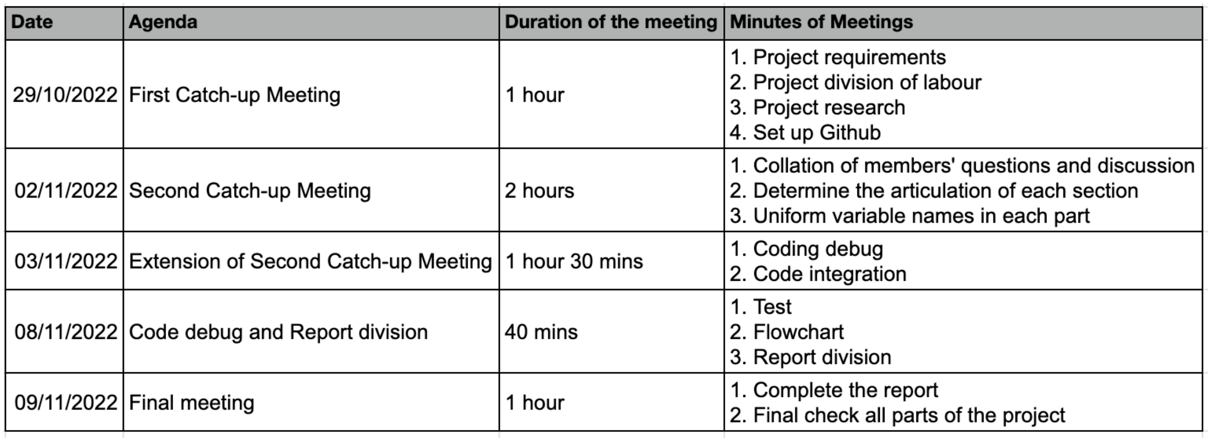
- Requirements.txt. A file to specify the versions of all the packages needed when running the application.packages includes : streamlit, pandas,altair,numpy, and psycopg2.
- Dockerfile. A file to specify the file arrangement(e.g. Path set-up, the version of python specify, and the command to run).
- Docker-compose.yml. A file to specify the images (which are Postgres and Streamlit in this case) and the default values in the environment.

How to run the web application?

1. Open new project with all the supported files in an python IDE
2. In the terminal, make sure current directory is the project file directory
3. Run **`docker build .`** in the terminal to builds Docker images from Dockerfile
4. Lastly run **`docker-compose up -d`** and compose will start
5. To run the app, open any browser and go to **`http://localhost:8501/`**
6. When web app opens, then provide database credentials to access database
7. For accessing local machine Postgres Database, Database Host should be: **`host.docker.internal`**

Group collaboration:

- Meetings brief:

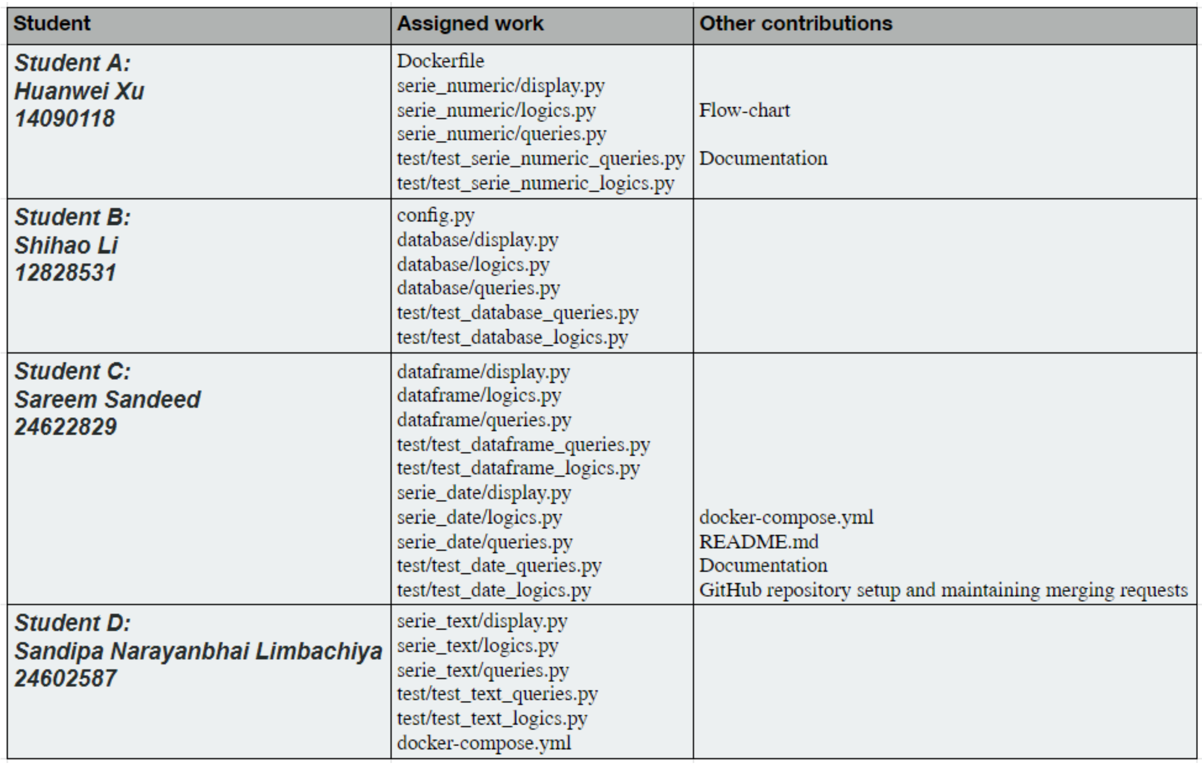


Date	Agenda	Duration of the meeting	Minutes of Meetings
29/10/2022	First Catch-up Meeting	1 hour	1. Project requirements 2. Project division of labour 3. Project research 4. Set up Github
02/11/2022	Second Catch-up Meeting	2 hours	1. Collation of members' questions and discussion 2. Determine the articulation of each section 3. Uniform variable names in each part
03/11/2022	Extension of Second Catch-up Meeting	1 hour 30 mins	1. Coding debug 2. Code integration
08/11/2022	Code debug and Report division	40 mins	1. Test 2. Flowchart 3. Report division
09/11/2022	Final meeting	1 hour	1. Complete the report 2. Final check all parts of the project

- Rules and Best practices:
  - Working with Git and GitHub repository for maintaining the codes and tracking changes
  - Each member had their own branch on repository and one member, student C, had merge rights to review and merge to master branch
  - All uploads were done through UNIX commands instead of directly uploading the files on the git repository

Group members’ Contribution/Problems faced & Implemented solutions:

Contribution of individual members:



Student	Assigned work	Other contributions
Student A: ***Huanwei Xu*** ***14090118***	Dockerfile serie_numeric/display.py serie_numeric/logics.py serie_numeric/queries.py test/test_serie_numeric_queries.py test/test_serie_numeric_logics.py	Flow-chart Documentation
Student B: ***Shihao Li*** ***12828531***	config.py database/display.py database/logics.py database/queries.py test/test_database_queries.py test/test_database_logics.py	
Student C: ***Sareem Sandeed*** ***24622829***	dataframe/display.py dataframe/logics.py dataframe/queries.py test/test_dataframe_queries.py test/test_dataframe_logics.py serie_date/display.py serie_date/logics.py serie_date/queries.py test/test_date_queries.py test/test_date_logics.py	docker-compose.yml README.md Documentation GitHub repository setup and maintaining merging requests
Student D: ***Sandipa Narayanbhai Limbachiya*** ***24602587***	serie_text/display.py serie_text/logics.py serie_text/queries.py test/test_text_queries.py test/test_text_logics.py docker-compose.yml	

Problems faces:

- *Part A: Huanwei Xu : 14090118*
  - Problem. Unable to check the serie_numeric works or not before the part of database set up well.
  - Solution. Apply psycopg2() to mock a connection for linking to Postgres.
- *Part B: Shihao Li : 12828531*
  - Problems:
    1. There is no online database for testing
    2. App config in the “config.py” file
  - Solution:
    1. Concert with the local database by using “Navicat”
- *Part C: Sareem Sandeed: 24622829*
  - Problems in solving code for display part of Dataframe and Serie_date
  - Took some time to figure out the datetime datatype issues
  - Communicating to team members about best practices of using git and GitHub
  - Running unit test and finding mock database
  - Debugging final version of web app
- *Part D: Sandipa Narayanbhai Limbachiya : 24602587*
  - I think not working in an Agile way is something I find the most difficult task. It was challenging for me to verify my code and logic because it depends on

other modules of the Application, on which we are all working very hard but in isolation.
    - While installing WSL2 as part of the Docker installation, I discovered some errors. It was a really strange problem with relation to Windows OS, but I eventually fixed it.

Suggested improvements to the code base or web application:

- Logics wise. The template have well-structured the logics of how to build up the application, but there hard to following since every developer had different understood to build up a application.
- Aesthetic wise. There are many potential better decorate for this application.
- Feature wise. To obtain a better understanding of business, is not enough from the application. There some pentential extra features(e.g, the measurements/statistics for the business purpose) should be included.
- The different parts of the code require a high degree of team coordination, with some of the tasks relying heavily on the previous parts. The division of labour could be rationalised a little more.

Reference:

1. Apply function to every row in a Pandas DataFrame. (2018, December 11). GeeksforGeeks. https://www.geeksforgeeks.org/apply-function-to-every-row-in-a-pandas-dataframe/
2. Count nan or missing values in pandas dataframe. (2020, July 1). GeeksforGeeks. https://www.geeksforgeeks.org/count-nan-or-missing-values-in-pandas-dataframe/
3. Counting number of rows with missing values in Pandas DataFrame. (n.d.). Retrieved October 20, 2022, from https://www.skytowner.com/explore/counting_number_of_rows_with_missing_value_in_pandas_dataframe
4. Find all numeric columns in postgresql database—Postgresql data dictionary queries. (n.d.). Retrieved October 22, 2022, from https://dataedo.com/kb/query/postgresql/find-all-numeric-columns
5. klin. (2018, April 5). Answer to “Include primary key on schema information in Postgres.” Stack Overflow. https://stackoverflow.com/a/49682741
6. Python | Pandas Series.dt.dayofweek. (2019, March 18). GeeksforGeeks. https://www.geeksforgeeks.org/python-pandas-series-dt-dayofweek/
7. Streamlit • The fastest way to build and share data apps. (n.d.). Retrieved November 6, 2022, from https://streamlit.io/

8. Docker documentation. (2022, November 8). Docker Documentation. https://docs.docker.com/
9. praveen. (2014, January 8). Answer to “What does os.path.abspath(Os. Path. Join(os. Path. Dirname(__file__), os. Path. Pardir)) mean? Python.” Stack Overflow. https://stackoverflow.com/a/21005918