README file for the Challenge Task

Candidate Name: **Ram Kumar Ruppa Surulinathan**

I thank you very much for offering this challenge. It was fun and at the same time thought provoking.

I have shared detailed information about different steps of the task along with some intermediate results.

---

1. **Mandatory Part (A)**

The Sourcer:

I started with the webscraping of the suggested 'tripadvisor' site but unfortunately, I was kept blocking even after I tried with different useragents and different device I have. [Response 403]

Then, I tried webscraping of Yelp website. This time it worked but the response it perfectly encrypted so that the python package was unable to decipher it. [Response 200]

Then, I Googled "berlin restaurants". I came across "Quandoo" webpage. I tried webscraping it and was successful.

For webscraping, I created a script called "quandoo_webscrapper_app.py" inside the directory "1. quandoo_restaurants_scraper"

I tried to handle different edge cases:

a. Best Case Scenario – Berlin – where there are 33 pages of results.
b. Best Case Scenario – Frankfurt - where there are 5 pages of results.
c. Edge Case Scenario – Rostock – where there is just 1 page result.
d. Worst Case Scenario - Paris – No Result.

As of Dec 16, Quandoo has 647 restaurants enlisted. The webscraping also generated the same number of results.

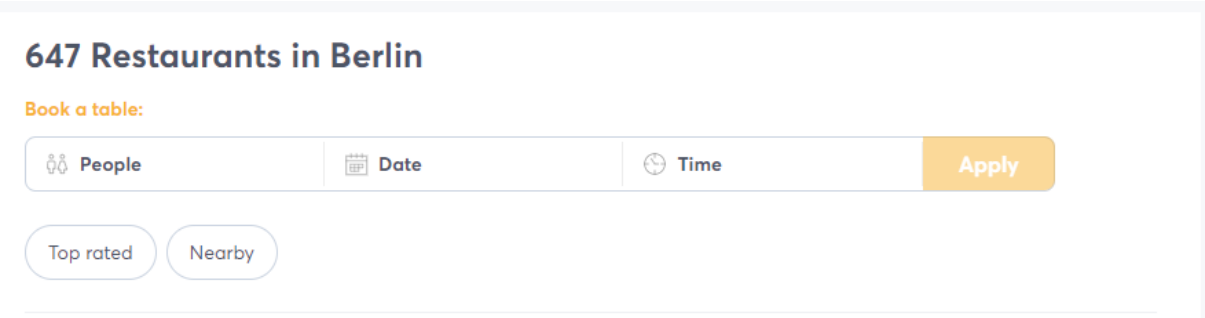


Fig 1: Berlin Restaurant Webpage (https://www.quandoo.de/en/result?destination=berlin) taken on 16.11.23

I added logging to know the progress of the scraping task and helped me to navigate to the right page in case of failure in the scraping process due to unexpected behavior (missing data).

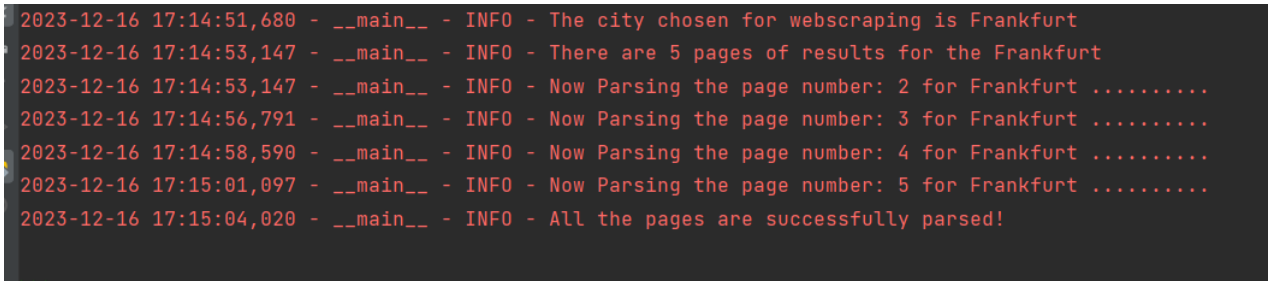


Fig 2: Logging records generated while scraping the restaurant data for Frankfurt

To run the webscraping script:

a. Please navigate to the directory '1.quandoo_restaurants_scraper'
b. It contains 'quandoo_webscraper_app.py' script, Dockerfile and 'scraped_data' directory to store the result files generated after scraping.
c. Docker commands to run:
    1. docker build -t python_quandoo_scraper .
    2. docker run -it -v "$(pwd)/:/quandoo_webscraper" python_quandoo_scraper
    3. The script can be run independently too, no input required.
d. The generated files are placed automatically inside the folder called "scraped_data" .

Extra improvements suggestions:

a. To have a schedule for the sourcer to run(daily/hourly/monthly)

I can think of orchestration tools like Airflow or Mage. Here I have created a DAG file called "quandoo_webscraping_orchestration.py" It is just a template and didn't execute it.

b. Think how to fetch only new restaurant data on each run

I will try to compare the new results with the previous results and identify those new restaurants name that are not present and scrape only those and add it to the results. [This could be iterative approach].

The Database:

A. Deploy a database. The database is expected to be supported by DBT:
B. Build the initial database layer to store the sourcer data

I have no prior experience in deploying a real database. My knowledge was limited to pgadmin. For this task, I tried using a docker-compose method referring Youtube tutorials.

To spin up a postgres database:

a. Please navigate to '2.database_setup' directory.
b. It contains docker-compose file, postgres_connection.py script and quandoo_berlin_results.csv file.
c. Docker commands to run:

1. docker-compose up -d
2. Please log on to localhost:8080 to see the database homepage.

d. The database login page appears like this:

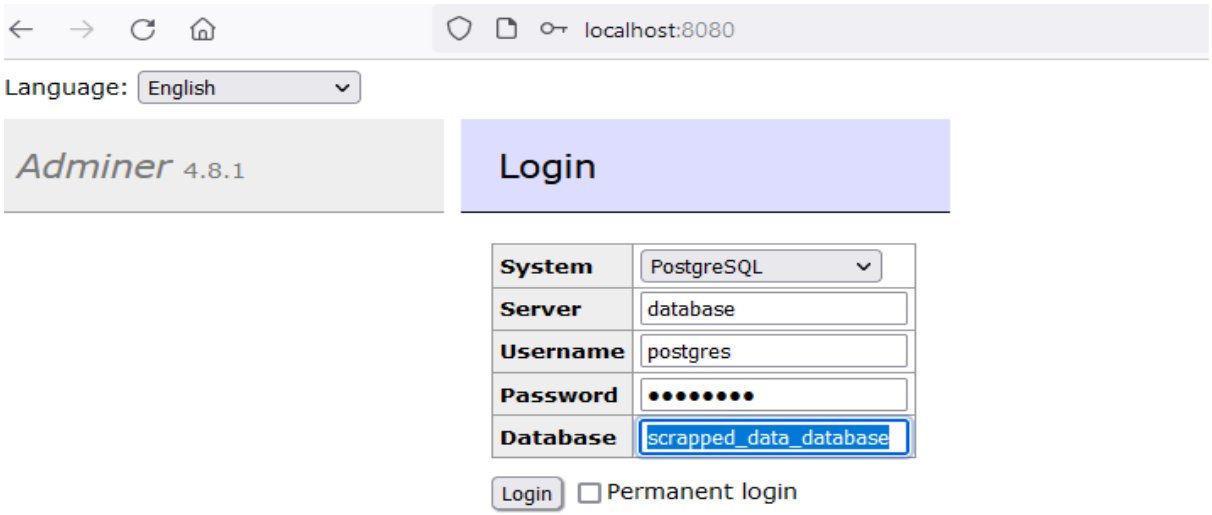


Fig 3: Loginpage of the database

e. After login, I created a table called “berlin_restaurants_table” in the database.

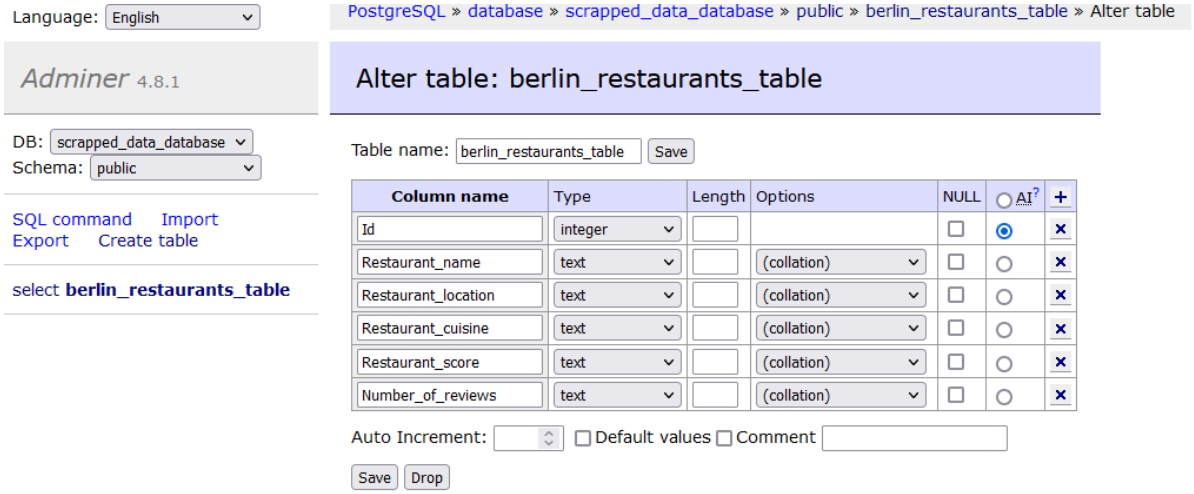
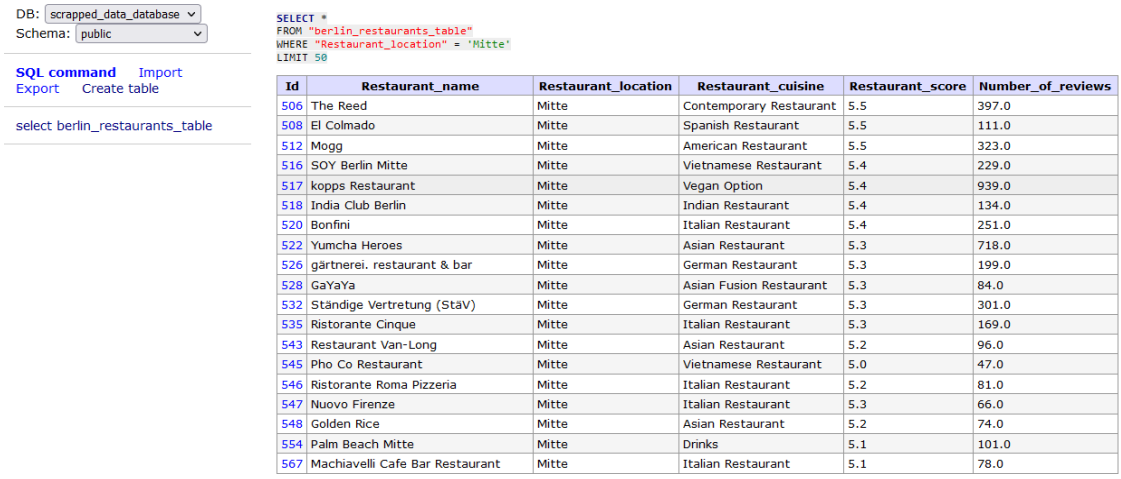


Fig 4: berlin_restaurants_table in the database

f. After populating the table with the data from the csv file, it looked like:



Id	Restaurant_name	Restaurant_location	Restaurant_cuisine	Restaurant_score	Number_of_reviews
506	The Reed	Mitte	Contemporary Restaurant	5.5	397.0
508	El Colmado	Mitte	Spanish Restaurant	5.5	111.0
512	Mogg	Mitte	American Restaurant	5.5	323.0
516	SOY Berlin Mitte	Mitte	Vietnamese Restaurant	5.4	229.0
517	kopps Restaurant	Mitte	Vegan Option	5.4	939.0
518	India Club Berlin	Mitte	Indian Restaurant	5.4	134.0
520	Bonfini	Mitte	Italian Restaurant	5.4	251.0
522	Yumcha Heroes	Mitte	Asian Restaurant	5.3	718.0
526	gärtnerei. restaurant & bar	Mitte	German Restaurant	5.3	199.0
528	GaYaYa	Mitte	Asian Fusion Restaurant	5.3	84.0
532	Ständige Vertretung (StäV)	Mitte	German Restaurant	5.3	301.0
535	Ristorante Cinque	Mitte	Italian Restaurant	5.3	169.0
543	Restaurant Van-Long	Mitte	Asian Restaurant	5.2	96.0
545	Pho Co Restaurant	Mitte	Vietnamese Restaurant	5.0	47.0
546	Ristorante Roma Pizzeria	Mitte	Italian Restaurant	5.2	81.0
547	Nuovo Firenze	Mitte	Italian Restaurant	5.3	66.0
548	Golden Rice	Mitte	Asian Restaurant	5.2	74.0
554	Palm Beach Mitte	Mitte	Drinks	5.1	101.0
567	Machiavelli Cafe Bar Restaurant	Mitte	Italian Restaurant	5.1	78.0

Fig 5: Simple querying in the database

g. The script called 'postgres_connection.py' can be used to add data and fetch the data.

C. Build Datamarts layer to store calculated data
D. Add more layers if it's needed

E. Use DBT to process data from the raw to the Datamarts layers

F. Consider how to update Datamarts on a certain schedule(hourly, daily)

Unfortunately, I have no prior exposure to Datamarts and building it. I tried to refer to some materials but didn't succeed in that. Therefore, I had skipped this segment and downstream segment associated with this.

The Optional Part (B & C)

Visualising the Data(B)

For this task, I tried to create a simple dashboard using Plotly dash with the scraped data of Berlin. (Berlin city maybe you can relate more)

The dashboard has:

1. Data table to see/filter the scraped results.
2. Scatter plot that represents the Berliner's top 10 highest review for each kind of cuisine.
3. Scatter plot that represents the Berliner's top 10 highest review for each locality.
4. Bar plot that represents the highest number of restaurants in each locality.

To run the visualization script:

a. Please navigate to '3.quandoo_dashboard'.
b. It contains 'berlin_restaurants_viz.py' script, Dockerfile and the directory 'scraped_data' containing the source file.
c. Docker commands to run:
    1. docker build -t python_quandoo_viz .
    2. docker run -h localhost -p 9002:9000 -d --name dash-app python_quandoo_viz

3. Please log on to http://127.0.0.1:9002/ see the dashboard live.
4. The script also can be run independently.

Quandoo Restaurants in Berlin

Restaurant_name	Restaurant_location	Restaurant_cuisine	Restaurant_score	Number_of_reviews
filter data...				
Taleh Thai	Prenzlauer Berg	Thai Restaurant	5	360
The Reed	Mitte	Contemporary Restaurant	5.5	397
GOLVET	Tiergarten	International Restaurant	5.9	68
El Colmado	Mitte	Spanish Restaurant	5.5	111
Vineria del Este	Friedrichshain	Spanish Restaurant	5.7	166
Delizie D'Italia	Prenzlauer Berg	Italian Restaurant	5	135
Ha-An	Prenzlauer Berg	Vietnamese Restaurant	5.5	369
Mogg	Mitte	American Restaurant	5.5	323
Feel Seoul Good	Pankow	Asian Restaurant	5.5	114
Mitho Chai P-Berg	Prenzlauer Berg	Asian Restaurant	5.5	131
Osteria Fiorello	Prenzlauer Berg	Italian Restaurant	5	75
SOY Berlin Mitte	Mitte	Vietnamese Restaurant	5.4	229
kopps Restaurant	Mitte	Vegan Option	5.4	939
India Club Berlin	Mitte	Indian Restaurant	5.4	134
Anjou	Prenzlauer Berg	Vietnamese Restaurant	5.4	162

Fig 6: Data table in the dashboard

Fig 7: Plots in the dashboard

Add Data Lake layer(C) - Unfortunately skipped this segment.

References:

1. https://python.plainenglish.io/how-to-run-a-python-file-inside-a-docker-container-e17668891c85
2. https://python.plainenglish.io/dockerizing-plotly-dash-5c23009fc10b
3. https://www.youtube.com/watch?v=nlk0QlPdbcY

Thanks for your time in reading my assignment.