Name: Rutu Devang Kansara
USC ID: 8368526607

Project Final Report: Air Quality Monitoring System

Introduction

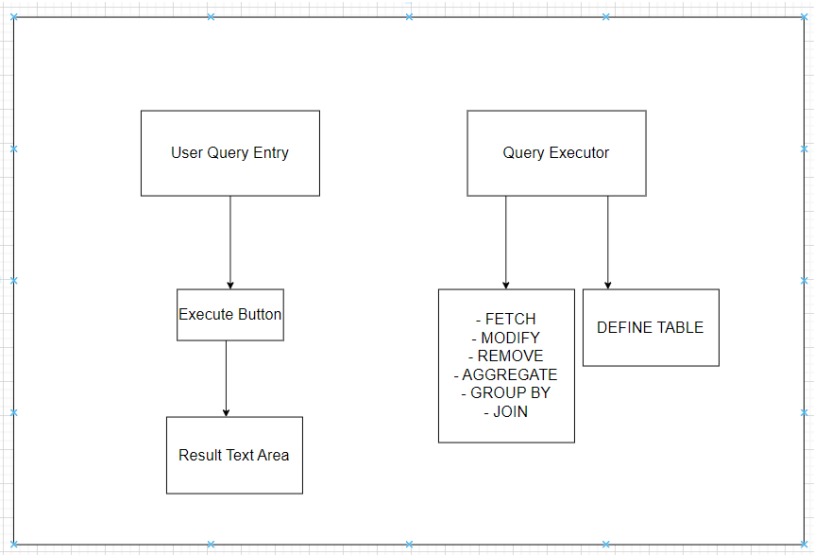
The Air Quality Monitoring System is designed to analyze and provide insights into the Sofia air quality dataset. This report outlines the project's design and implementation, emphasizing key components and functionalities.

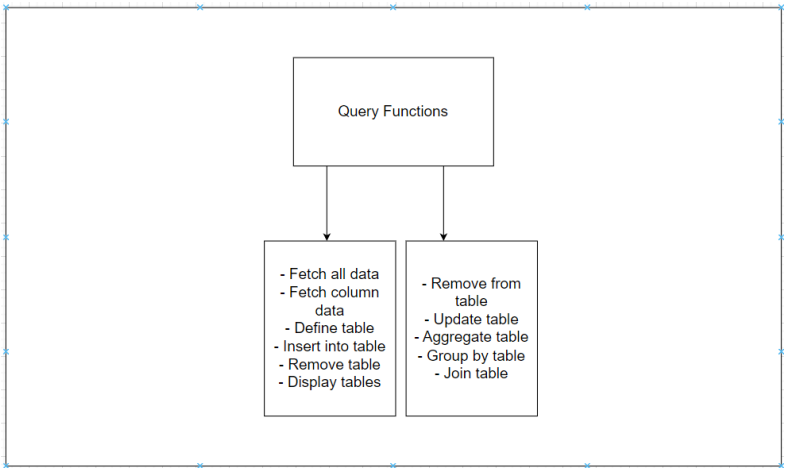
Planned Implementation (From Project Proposal)

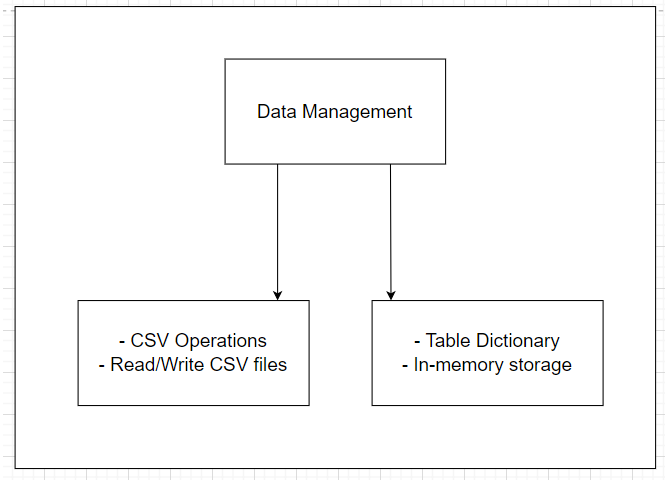
The initial plan focused on leveraging the Sofia air quality dataset from Kaggle. The project aimed to implement a relational data model for efficient data storage and advanced querying.

Choosing a relational data model was a strategic decision. This approach fulfilled immediate data needs and laid the groundwork for future expansion. The following sections will delve into the journey from this initial plan to developing the Air Quality Monitoring System.

Architecture Design







Description

The system follows a modular data import, cleaning, exploration, and database design architecture. It includes tables for sensors, measurements, label counts, datetime counts, label ranges, and geographical ranges.

The data flow starts with dataset import, cleaning, exploration, and database design. The architecture ensures flexibility in querying for information.

Implementation

1. Data Cleaning and Exploration

- Data Import:

- I imported the Sofia air quality dataset from 'air_quality_data.csv.'

- Column Management:

- Removed unnecessary columns.

- Date and Time Parsing:

- Parsed 'timestamp' into 'date' and 'time' columns.

- Redundant Column Removal:

- Eliminated redundant columns.

- Missing Data Check:

- Checked for missing data.

- Data Summary:

- Computed summary statistics.

```
data1.describe()
```

	sensor_id	location	lat	lon	temperature	humidity
count	701548.000000	701548.000000	701548.000000	701548.000000	701548.000000	701548.000000
mean	2835.419842	1426.366892	42.679722	23.333739	24.754990	48.348949
std	798.018890	403.446639	0.025568	0.039365	14.013001	20.907247
min	1764.000000	879.000000	42.622000	23.240000	-145.120000	0.000000
25%	2224.000000	1118.000000	42.665000	23.310000	20.630000	34.140000
50%	2323.000000	1172.000000	42.685000	23.332000	24.780000	48.350000
75%	3474.000000	1751.000000	42.694000	23.360000	29.870000	62.940000
max	4661.000000	2343.000000	42.738000	23.419000	61.170000	100.000000

2. Database Design and Query Language Development

- Table Definition:

- Defined tables for sensors, measurements, label counts, datetime counts, label ranges, and geographical ranges.

- Query Language Implementation:

- Developed a simple query language for data retrieval and modification.

3. Functionalities

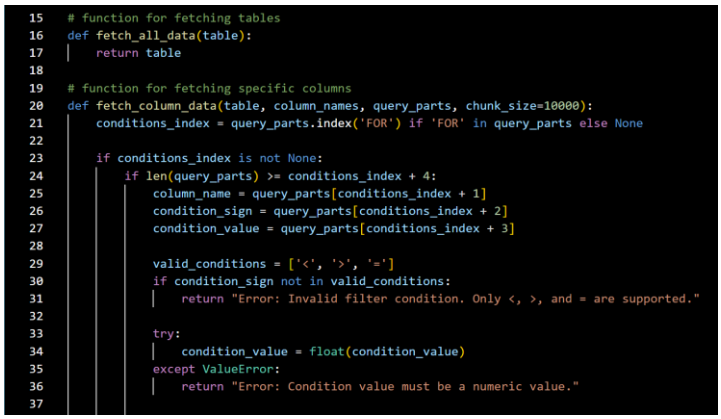
Data Retrieval

- Fetch All Data:

- Implemented functionality to retrieve all data from specified tables.

- Fetch Specific Columns:

- Retrieved specific columns with and without conditions.



```
# function for fetching tables
def fetch_all_data(table):
    return table

# function for fetching specific columns
def fetch_column_data(table, column_names, query_parts, chunk_size=10000):
    conditions_index = query_parts.index('FOR') if 'FOR' in query_parts else None

    if conditions_index is not None:
        if len(query_parts) >= conditions_index + 4:
            column_name = query_parts[conditions_index + 1]
            condition_sign = query_parts[conditions_index + 2]
            condition_value = query_parts[conditions_index + 3]

            valid_conditions = ['<', '>', '=']
            if condition_sign not in valid_conditions:
                return "Error: Invalid filter condition. Only <, >, and = are supported."

            try:
                condition_value = float(condition_value)
            except ValueError:
                return "Error: Condition value must be a numeric value."

```

Database Modification

- Insertion of Values:

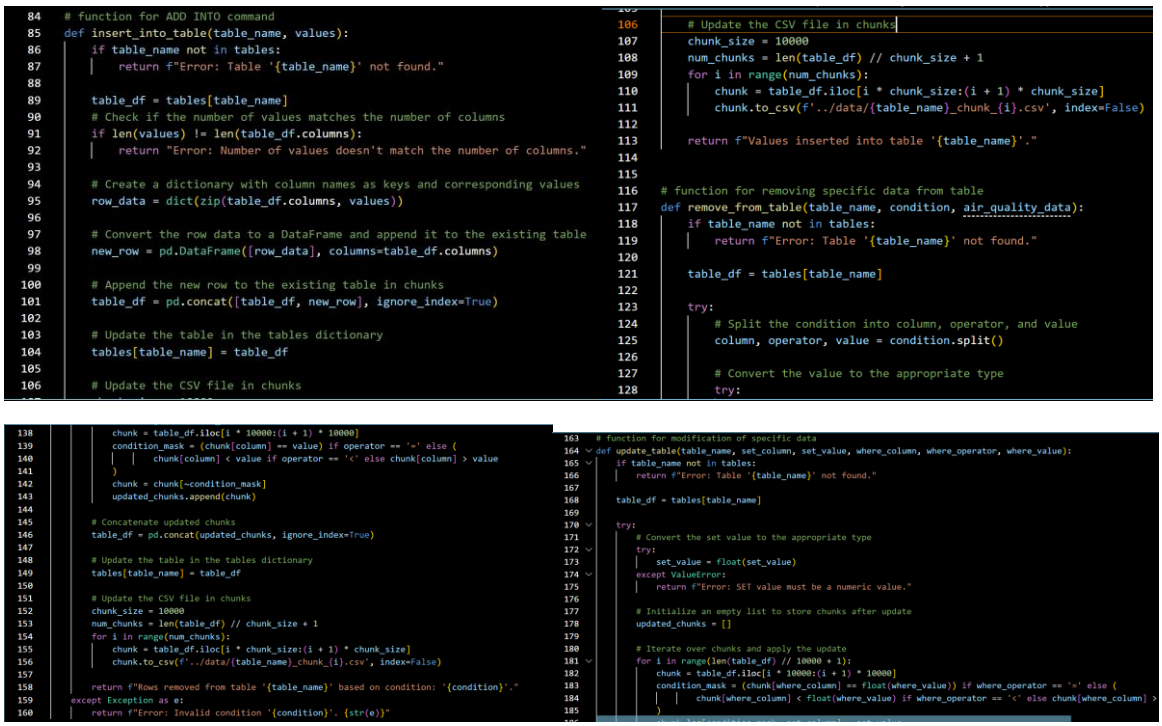
- Successfully inserted values into tables.

- Row Removal:

- Removed rows based on specified conditions.

- Value Updates:

- Updated table values based on conditions.



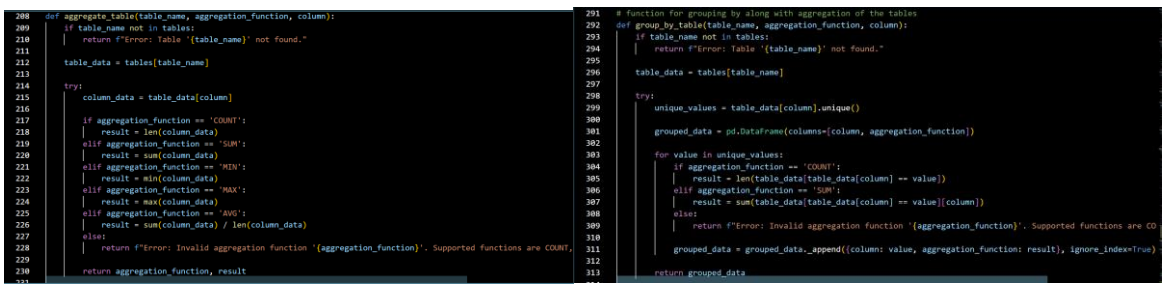
Aggregation and Grouping

- Aggregation Functions:

- Performed basic aggregation functions.

- Grouping Data:

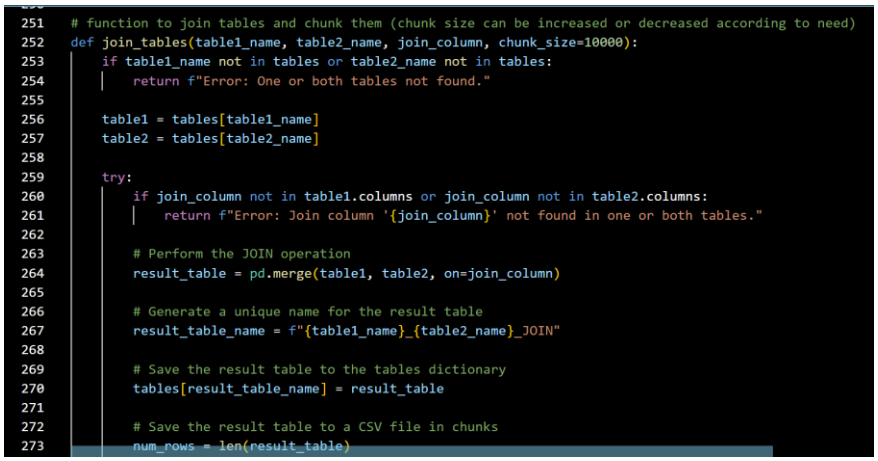
- Grouped data by specified columns.



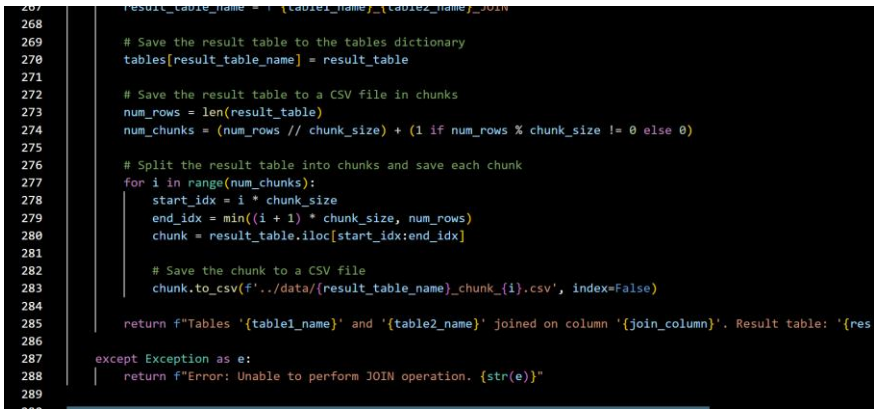
JOIN Operations

- Table Joins:

- Executed JOIN operations on specified tables.



```python
# function to join tables and chunk them (chunk size can be increased or decreased according to need)
def join_tables(table1_name, table2_name, join_column, chunk_size=10000):
    if table1_name not in tables or table2_name not in tables:
        return f"Error: One or both tables not found."

    table1 = tables[table1_name]
    table2 = tables[table2_name]

    try:
        if join_column not in table1.columns or join_column not in table2.columns:
            return f"Error: Join column '{join_column}' not found in one or both tables."

        # Perform the JOIN operation
        result_table = pd.merge(table1, table2, on=join_column)

        # Generate a unique name for the result table
        result_table_name = f"{table1_name}_{table2_name}_JOIN"

        # Save the result table to the tables dictionary
        tables[result_table_name] = result_table

        # Save the result table to a CSV file in chunks
        num_rows = len(result_table)
```



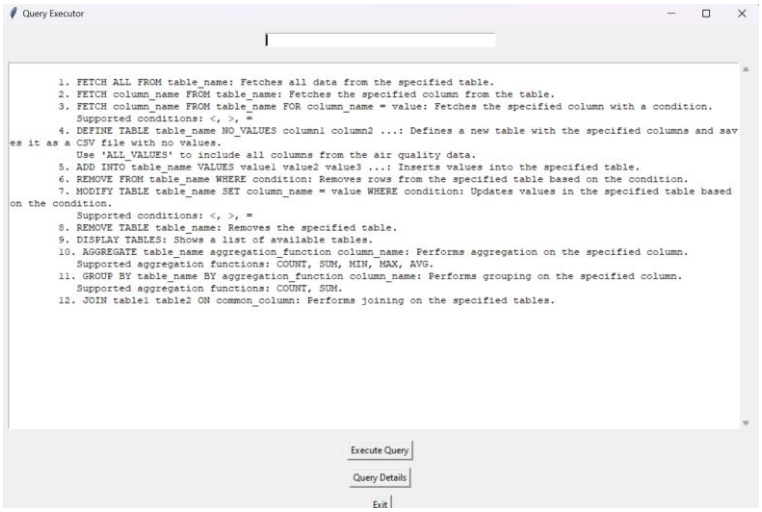
```python
        # Save the result table to the tables dictionary
        tables[result_table_name] = result_table

        # Save the result table to a CSV file in chunks
        num_rows = len(result_table)
        num_chunks = (num_rows // chunk_size) + (1 if num_rows % chunk_size != 0 else 0)

        # Split the result table into chunks and save each chunk
        for i in range(num_chunks):
            start_idx = i * chunk_size
            end_idx = min((i + 1) * chunk_size, num_rows)
            chunk = result_table.iloc[start_idx:end_idx]

            # Save the chunk to a CSV file
            chunk.to_csv(f'../data/{result_table_name}_chunk_{i}.csv', index=False)

        return f"Tables '{table1_name}' and '{table2_name}' joined on column '{join_column}'. Result table: '{res

    except Exception as e:
        return f"Error: Unable to perform JOIN operation. {str(e)}"
```

Tech Stack

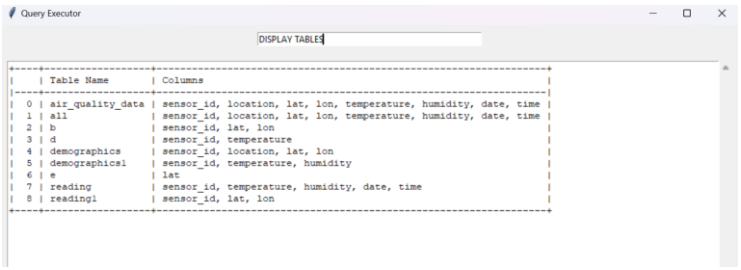
- Python
- Pandas
- Tkinter (for GUI)
- Memory Profiler
- Tabulate (for table display)

Implementation Screenshots

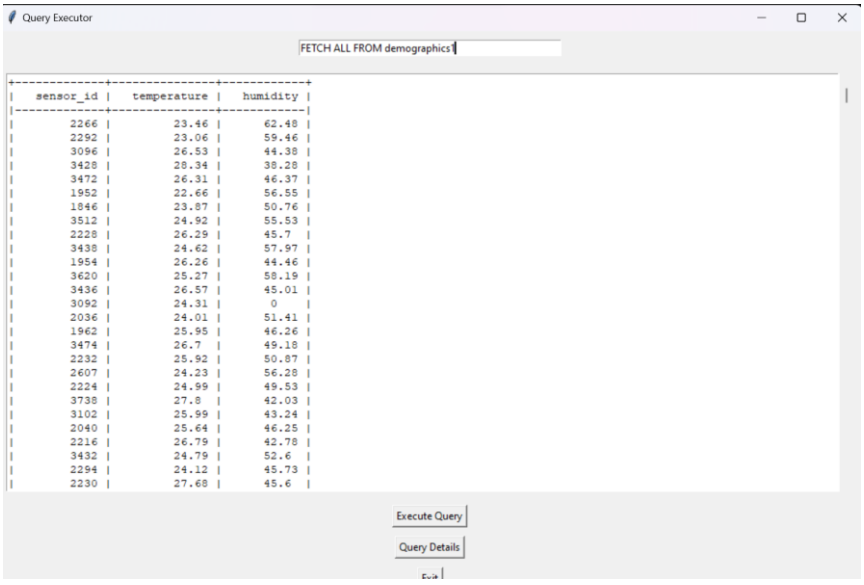
Simple interface for user-friendliness:



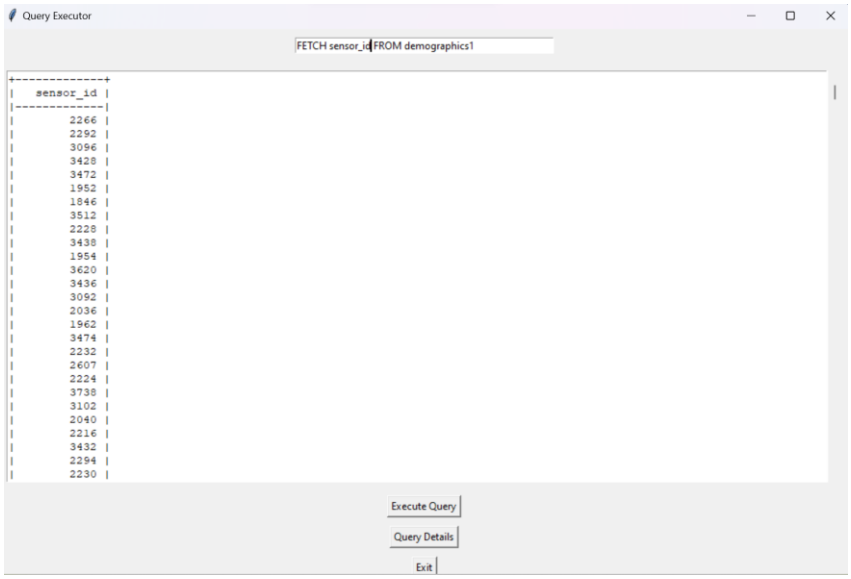
DISPLAY TABLES:



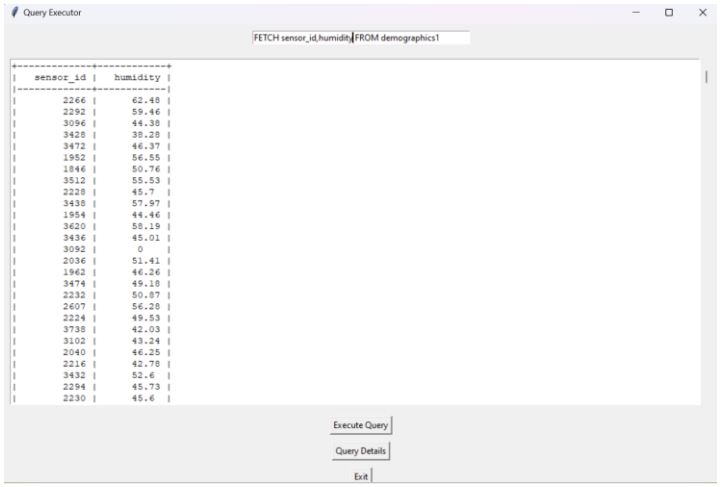
FETCH ALL FROM demographics1



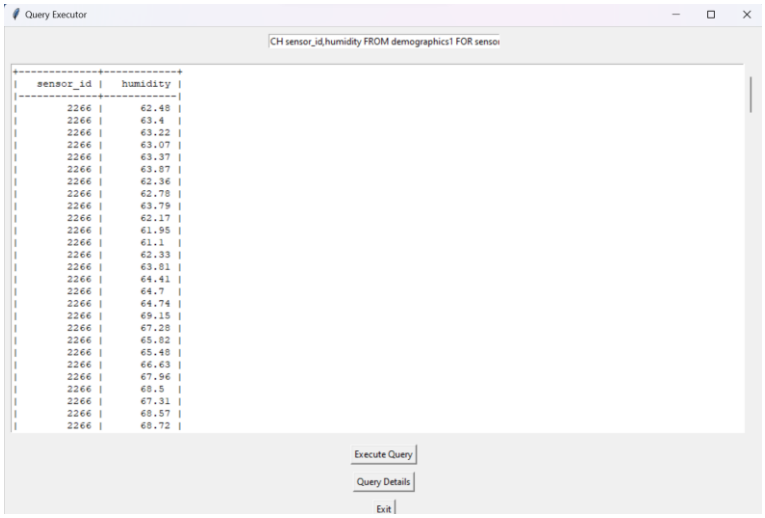
FETCH sensor_id FROM demographics1



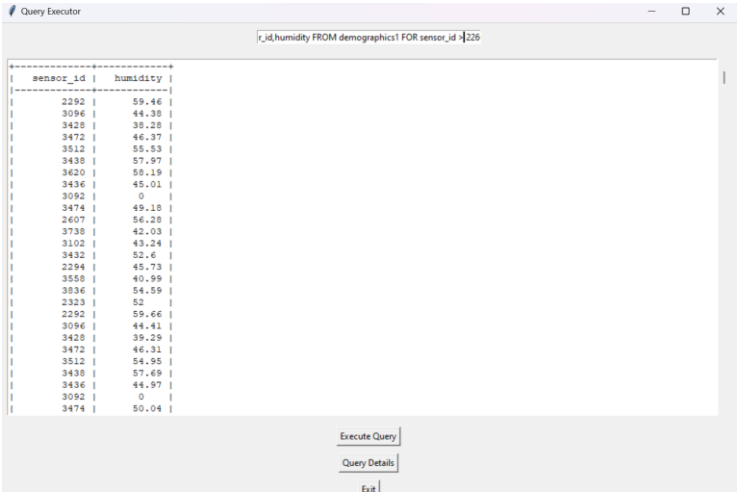
Fetch sensor_id,humidity from demographics1



FETCH sensor_id,humidity FROM demographics1 FOR sensor_id = 2266

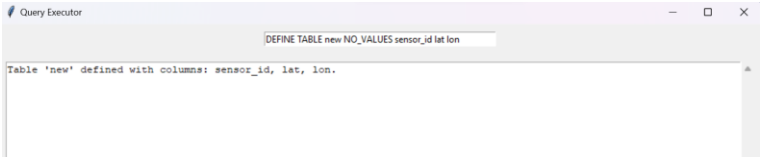


FETCH sensor_id,humidity FROM demographics1 FOR sensor_id > 2266



sensor_id	humidity
2292	59.46
3096	44.38
3428	38.28
3472	46.37
3512	55.53
3438	57.97
3620	58.19
3436	45.01
3092	0
3474	49.18
2607	56.28
3738	42.03
3102	43.24
3432	52.6
2294	45.73
3558	40.99
3836	54.59
2323	52
2292	59.66
3096	44.41
3428	39.29
3472	46.31
3512	54.95
3438	57.69
3436	44.97
3092	0
3474	50.04

DEFINE TABLE new NO_VALUES sensor_id lat lon



New table created

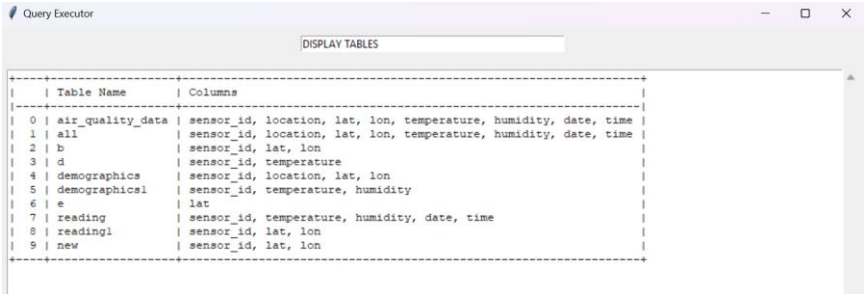
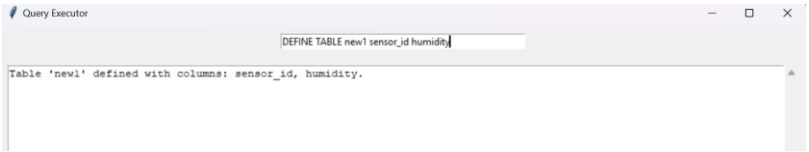
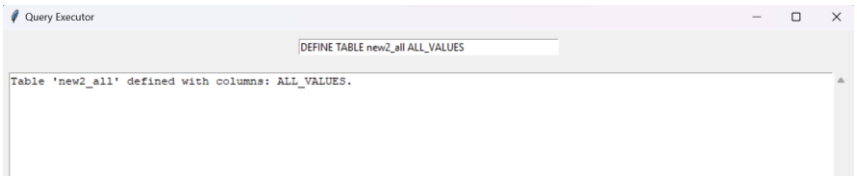


	Table Name	Columns
0	air_quality_data	sensor_id, location, lat, lon, temperature, humidity, date, time
1	all	sensor_id, location, lat, lon, temperature, humidity, date, time
2	b	sensor_id, lat, lon
3	d	sensor_id, temperature
4	demographics	sensor_id, location, lat, lon
5	demographics1	sensor_id, temperature, humidity
6	e	lat
7	reading	sensor_id, temperature, humidity, date, time
8	reading1	sensor_id, lat, lon
9	new	sensor_id, lat, lon

DEFINE TABLE new1 sensor_id humidity





Chunking

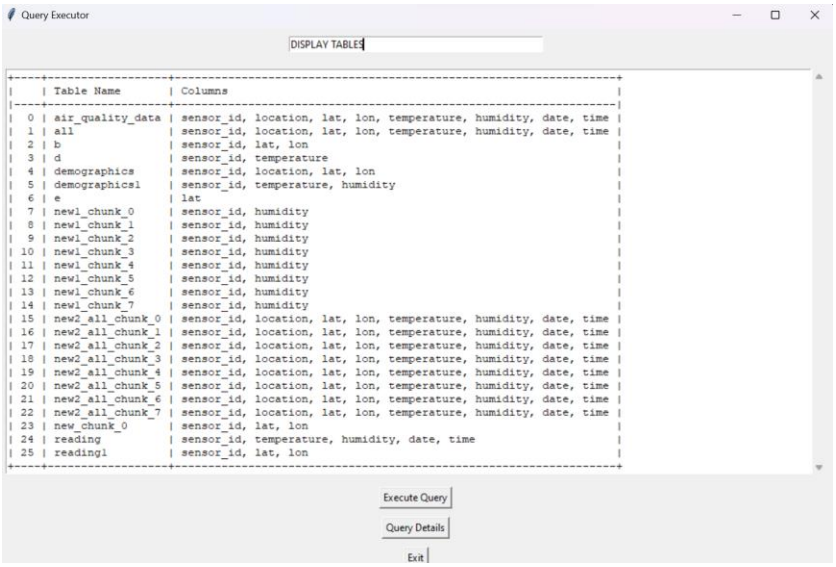
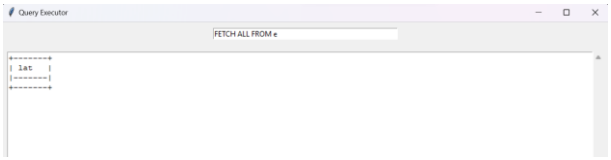


	Table Name	Columns
0	air_quality_data	sensor_id, location, lat, lon, temperature, humidity, date, time
1	all	sensor_id, location, lat, lon, temperature, humidity, date, time
2	b	sensor_id, lat, lon
3	d	sensor_id, temperature
4	demographics	sensor_id, location, lat, lon
5	demographics1	sensor_id, temperature, humidity
6	e	lat
7	new1_chunk_0	sensor_id, humidity
8	new1_chunk_1	sensor_id, humidity
9	new1_chunk_2	sensor_id, humidity
10	new1_chunk_3	sensor_id, humidity
11	new1_chunk_4	sensor_id, humidity
12	new1_chunk_5	sensor_id, humidity
13	new1_chunk_6	sensor_id, humidity
14	new1_chunk_7	sensor_id, humidity
15	new2_all_chunk_0	sensor_id, location, lat, lon, temperature, humidity, date, time
16	new2_all_chunk_1	sensor_id, location, lat, lon, temperature, humidity, date, time
17	new2_all_chunk_2	sensor_id, location, lat, lon, temperature, humidity, date, time
18	new2_all_chunk_3	sensor_id, location, lat, lon, temperature, humidity, date, time
19	new2_all_chunk_4	sensor_id, location, lat, lon, temperature, humidity, date, time
20	new2_all_chunk_5	sensor_id, location, lat, lon, temperature, humidity, date, time
21	new2_all_chunk_6	sensor_id, location, lat, lon, temperature, humidity, date, time
22	new2_all_chunk_7	sensor_id, location, lat, lon, temperature, humidity, date, time
23	new_chunk_0	sensor_id, lat, lon
24	reading	sensor_id, temperature, humidity, date, time
25	reading1	sensor_id, lat, lon

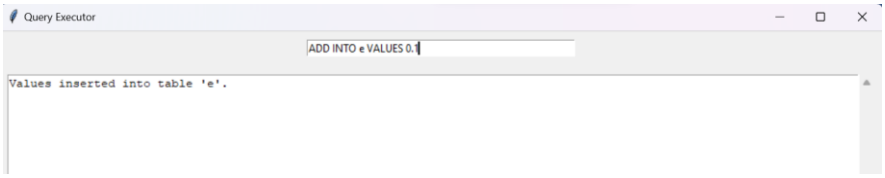
ADD INTO:

Empty table e:

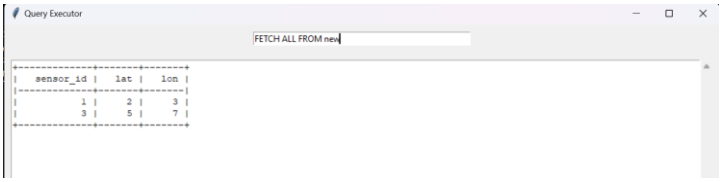


lat

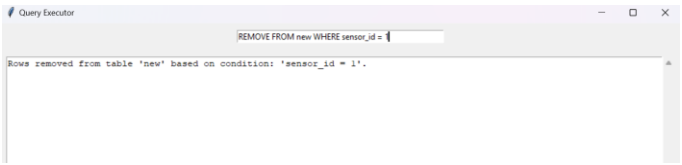
ADD INTO e VALUES 0.1

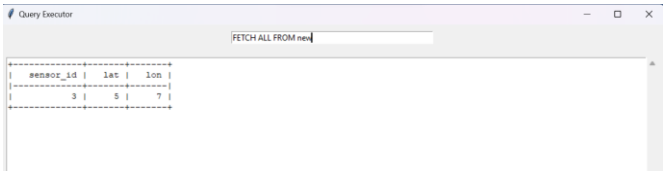


REMOVE FROM



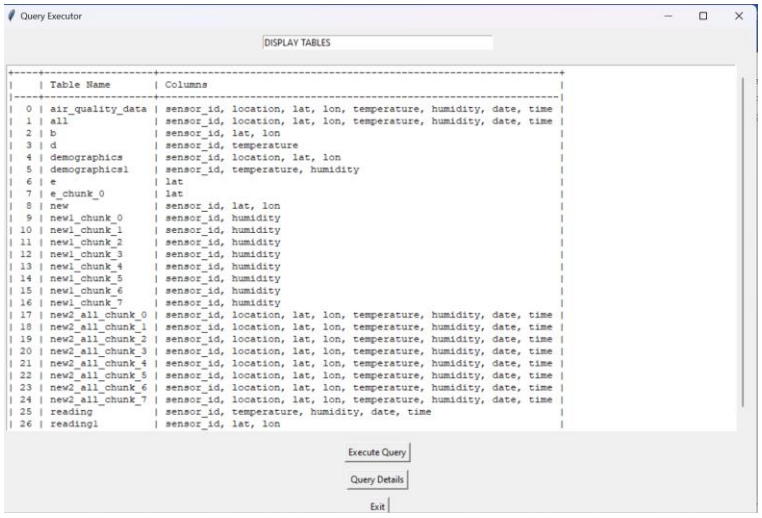
sensor_id	lat	lon
1	2	3
3	5	7

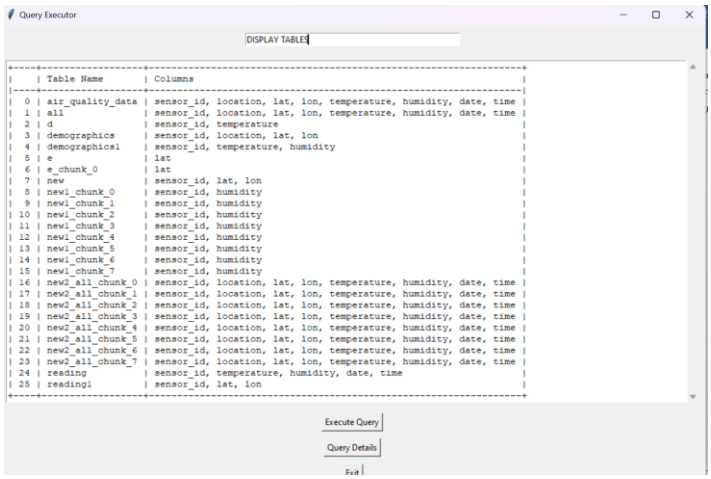




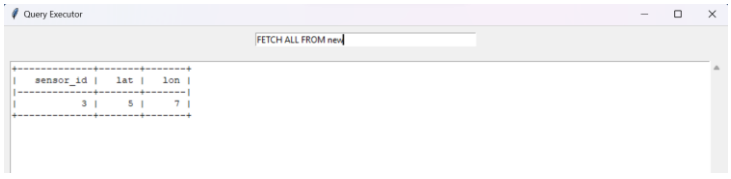
sensor_id	lat	lon
3	5	7

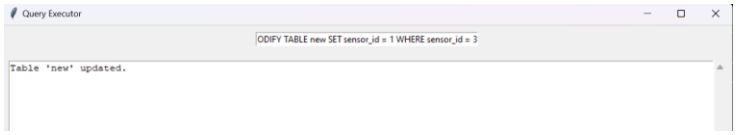
REMOVE TABLE

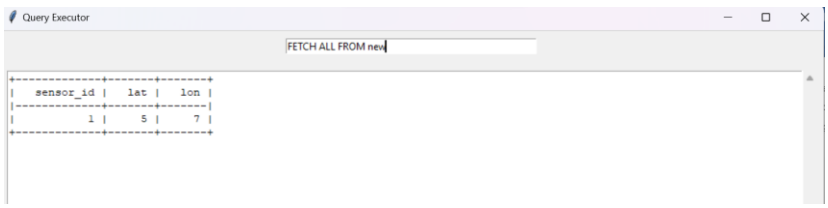




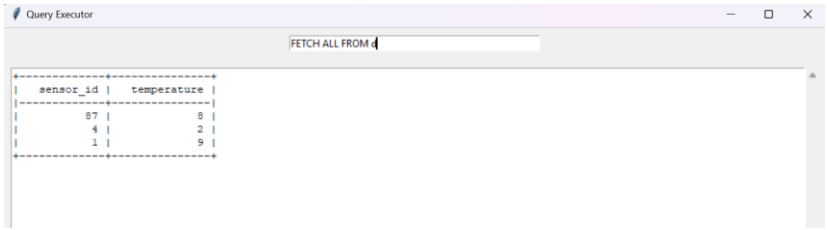
MODIFY TABLE





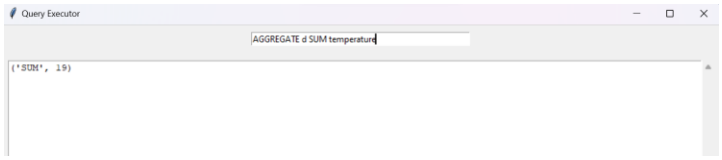


AGGREGATE

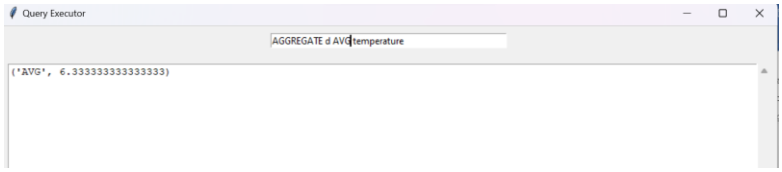


COUNT

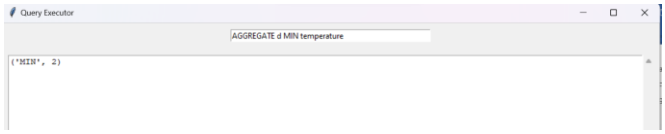
SUM



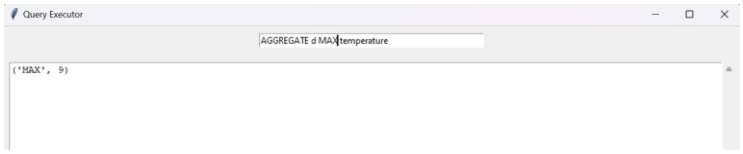
AVG



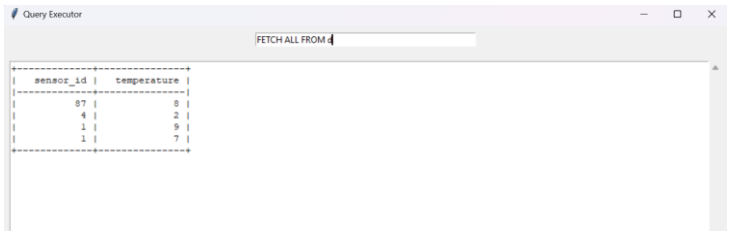
MIN

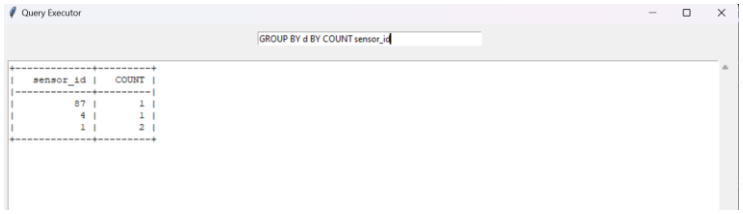


MAX

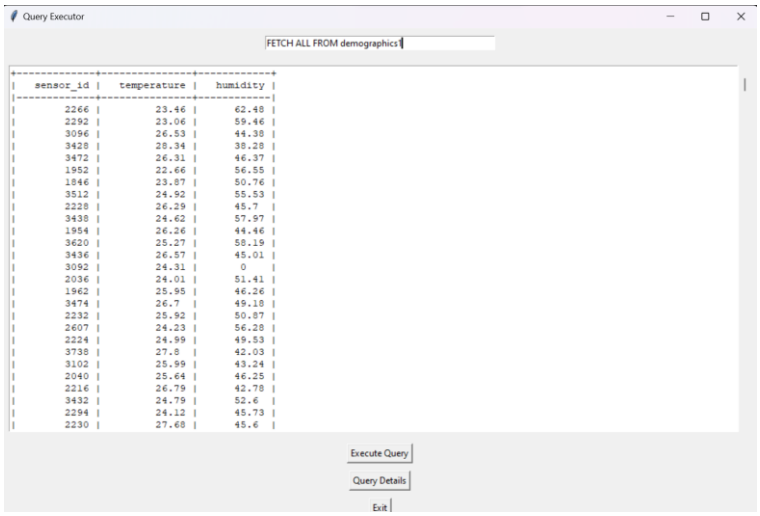


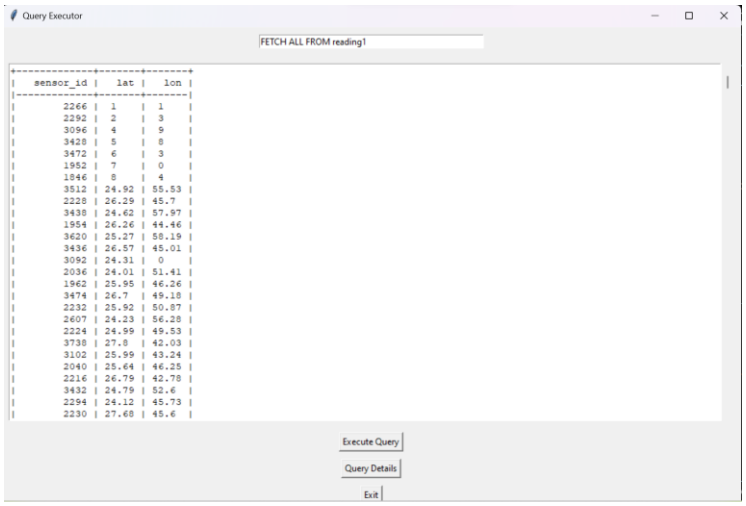
GROUP BY

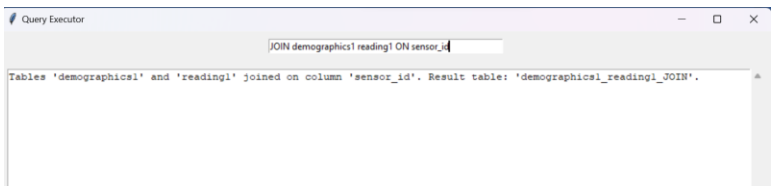




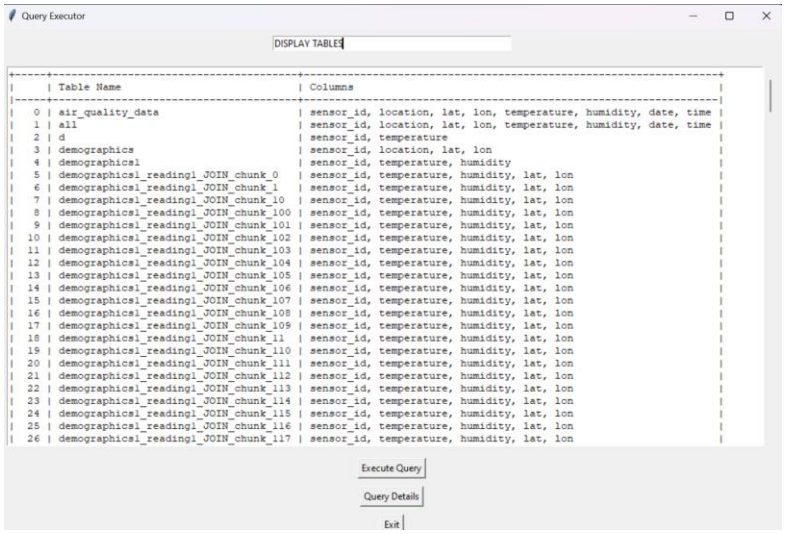
JOIN

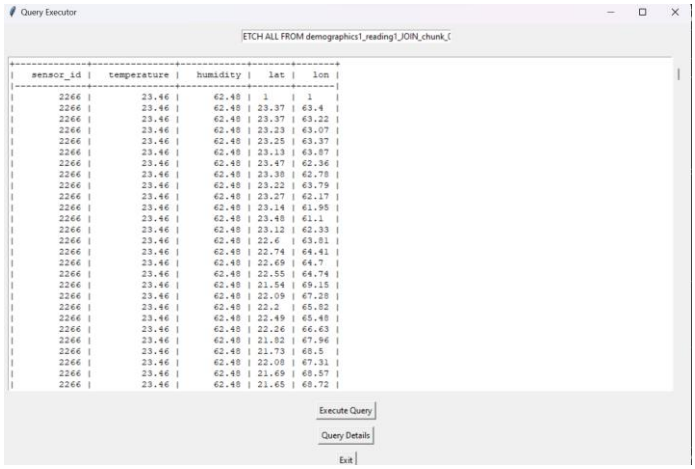






Chunks created after JOIN operation





Learning Outcomes

1. Technical Skills

Python Proficiency: Significantly enhanced skills in Python programming, utilizing it for diverse tasks within the project. This involved not only scripting logic but also effective use of Pandas for data manipulation and analysis.

GUI Development: Acquired hands-on experience in creating Graphical User Interfaces (GUIs) using Tkinter. This practical knowledge contributes to the ability to develop interactive and user-friendly applications.

2. Database Design

Relational Database Concepts: Developed a solid understanding of relational database design principles. This encompasses structuring data in a way that ensures efficiency, consistency, and meaningful relationships.

Query Languages: Gained proficiency in query languages for databases, enabling effective communication with the underlying data structures. This skill set is fundamental in extracting, updating, and managing data within the project's database.

Challenges Faced

- Interface Design:

- Not having used tkinter before, crafting a user-friendly Tkinter interface posed challenges, but I navigated them by prioritizing simplicity. The goal was to create an interface anyone could understand and interact with.

- Data Handling:

- Initially ran into some errors while managing complexities in parsing data and time for accurate analysis. Dealing with date and time intricacies in the data required meticulous handling. I tackled this challenge to ensure precise analysis.

Conclusion

To wrap up, the Air Quality Monitoring System project has achieved its goals, providing a solid foundation for further exploration of Sofia's air quality data. This marks a significant step in understanding and enhancing air quality monitoring capabilities.

Future Scope

1. Visualization

- Enhancing the visual representation of air quality data remains a key focus. By implementing more engaging graphical elements, the system can provide users with a clearer understanding of air quality trends. Additionally, exploring real-time data streaming capabilities will offer up-to-the-minute insights.

2. User Interface Enhancement

- Refine and enhance the website/interface design.

- Striving for an even more intuitive design will provide a seamless user experience.

3. Advanced Analytics

- The system's capabilities can be extended by delving into advanced analytics. Exploring machine learning models for predictive analysis opens the door to anticipating air quality changes. This proactive approach can be invaluable in decision-making and environmental planning.

Google Drive Submission Link

https://drive.google.com/drive/folders/1yTM69XO8SIthFuYAtX4zEFRk4igIHYS5?usp=drive_link

I appreciate your help and guidance throughout this course and project.

Thank you!