DAV 6100 IA Final Project Report Spring 2024

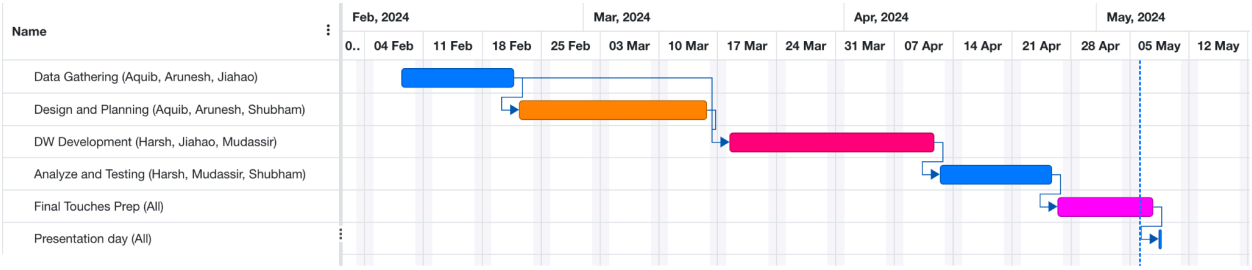
1. Introduction

- **Project Name: Mission Green**
- **GIT Repository: Link**
  Project Members: Aquib Hussain Mohammed, Arunesh Kumar Rai, Harshkumar Jeetendra Vaghmaria, Jiahao Li, Mudassir Imam, Shubham Pant
- **Professor:** Brandon Chiazza

- **Problem Statement:** In today's world, as the population increases, we face a significant problem: finding places where the air is **clean to breathe and the water is safe to drink**. To tackle this issue, we've come together to locate areas with clean air and drinkable water.
- **Goal Statement:** Our goal is to provide vital information to our stakeholders using the given data analyzed, our stakeholders will be able to make informed decisions.
- **Our Stakeholders:** Local governments, businesses, non-profits, Educational institutions, development agencies, and international organizations.

- **Business/ Use Cases:**
  - Healthcare Planning: Healthcare providers can use this data to identify regions with high rates of respiratory or waterborne illnesses attributable to poor air and water quality. This information can inform healthcare planning and resource allocation, such as the distribution of medical supplies or the implementation of preventive health programs.
  - Business Site Selection: Businesses, especially those in the hospitality or wellness industries, can use this data to choose locations for new branches, resorts, or wellness centers. They can market their services as being in areas with high-quality air and water, attracting health-conscious customers.
  - Urban Planning and Development: Local governments can use this information to plan new infrastructure projects, such as parks or residential areas, in locations with clean air and safe water sources. This ensures that residents have access to healthy living environments.
  - Real Estate Agents: The Areas with good AQI and Water Quality are encouraged to rent or purchase as they pose a higher demand in the Market.

- **Timeline, Roles, and Responsibilities for this initiative:**



- ❖ Project Management Lead: Mudassir
- ❖ Data Management Lead: Harshkumar

- **Assumptions:**

a. Data Accuracy:
   - Data acquired from sources like the EPA and city records is assumed to be accurate and reliable.
   - Any discrepancies or errors could impact the project's outcomes.

b. Data Integration:
   - Integration of air quality and drinking water data will be seamless.
   - Challenges may arise in reconciling differences in data formats or standards.

c. Frequency of Data Updates:
   - Monthly data updates via AWS Lambda are assumed to be consistent.
   - Regular updates ensure stakeholders have access to the latest information.
   - Limitations for the "Mission Green" Project:

- **Limitations:**

a. Geographic Coverage:
   - Findings may not generalize to areas beyond New York City.
   - Stakeholders outside these regions may not benefit directly.

b. Data Granularity:
   - Varying levels of data granularity could limit analysis depth.
   - The specificity of recommendations may be constrained.

2. Data Acquisition:

A. Air Quality Data:
   - Method to Download: CSV Download
   - Structured Dataset
   - Link:https://www.epa.gov/outdoor-air-quality-data/download-daily-data
   - Code file: Air_Script.ipynb
   - Output:

```
df_CO = read_data_from_folder('mission-green-bucket', 'Air_Quality_Data/CO')
df_NO2 = read_data_from_folder('mission-green-bucket', 'Air_Quality_Data/NO2')
df_Ozone = read_data_from_folder('mission-green-bucket', 'Air_Quality_Data/Ozone')
df_PM10 = read_data_from_folder('mission-green-bucket', 'Air_Quality_Data/PM10')
df_PM2 = read_data_from_folder('mission-green-bucket', 'Air_Quality_Data/PM2.5')
df_SO2 = read_data_from_folder('mission-green-bucket', 'Air_Quality_Data/SO2')
```

```
File 'Air_Quality_Data/CO/ad_viz_plotval_data_CO_2020.csv' read and stored as DataFrame.
File 'Air_Quality_Data/CO/ad_viz_plotval_data_CO_2021.csv' read and stored as DataFrame.
File 'Air_Quality_Data/CO/ad_viz_plotval_data_CO_2022.csv' read and stored as DataFrame.
File 'Air_Quality_Data/CO/ad_viz_plotval_data_CO_2023.csv' read and stored as DataFrame.
File 'Air_Quality_Data/NO2/ad_viz_plotval_data_NO2_2020.csv' read and stored as DataFrame.
File 'Air_Quality_Data/NO2/ad_viz_plotval_data_NO2_2021.csv' read and stored as DataFrame.
File 'Air_Quality_Data/NO2/ad_viz_plotval_data_NO2_2022.csv' read and stored as DataFrame.
File 'Air_Quality_Data/NO2/ad_viz_plotval_data_NO2_2023.csv' read and stored as DataFrame.
File 'Air_Quality_Data/Ozone/ad_viz_plotval_data_Ozone_2020.csv' read and stored as DataFrame.
File 'Air_Quality_Data/Ozone/ad_viz_plotval_data_Ozone_2021.csv' read and stored as DataFrame.
File 'Air_Quality_Data/Ozone/ad_viz_plotval_data_Ozone_2022.csv' read and stored as DataFrame.
File 'Air_Quality_Data/Ozone/ad_viz_plotval_data_Ozone_2023.csv' read and stored as DataFrame.
File 'Air_Quality_Data/Ozone/ad_viz_plotval_data_Ozone_2024.csv' read and stored as DataFrame.
File 'Air_Quality_Data/PM10/ad_viz_plotval_data_PM10_2020.csv' read and stored as DataFrame.
File 'Air_Quality_Data/PM10/ad_viz_plotval_data_PM10_2021.csv' read and stored as DataFrame.
File 'Air_Quality_Data/PM10/ad_viz_plotval_data_PM10_2022.csv' read and stored as DataFrame.
File 'Air_Quality_Data/PM10/ad_viz_plotval_data_PM10_2023.csv' read and stored as DataFrame.
File 'Air_Quality_Data/PM10/ad_viz_plotval_data_PM10_2024.csv' read and stored as DataFrame.
File 'Air_Quality_Data/PM2.5/ad_viz_plotval_data_PM2.5_2020.csv' read and stored as DataFrame.
File 'Air_Quality_Data/PM2.5/ad_viz_plotval_data_PM2.5_2021.csv' read and stored as DataFrame.
File 'Air_Quality_Data/PM2.5/ad_viz_plotval_data_PM2.5_2022.csv' read and stored as DataFrame.
File 'Air_Quality_Data/PM2.5/ad_viz_plotval_data_PM2.5_2023.csv' read and stored as DataFrame.
File 'Air_Quality_Data/PM2.5/ad_viz_plotval_data_PM2.5_2024.csv' read and stored as DataFrame.
File 'Air_Quality_Data/SO2/ad_viz_plotval_data_SO2_2020.csv' read and stored as DataFrame.
File 'Air_Quality_Data/SO2/ad_viz_plotval_data_SO2_2021.csv' read and stored as DataFrame.
File 'Air_Quality_Data/SO2/ad_viz_plotval_data_SO2_2022.csv' read and stored as DataFrame.
File 'Air_Quality_Data/SO2/ad_viz_plotval_data_SO2_2023.csv' read and stored as DataFrame.
```

- Data Dictionary: Air Quality Data Description.xlsx
- Integration: Batch Migration

B. Drinking Water Inspection Data:

- Method To Download: API Get
- Unstructured Dataset
- Link:https://data.cityofnewyork.us/Health/Self-Reported-Drinking-Water-Tank-Inspection-Resul/gjm4-k24g/about_data
- Code file: Water_Script.ipynb
- Output:

Drinking_Water_Quality_Data/Self_Reported_Drinking_Water_Tank_Inspection_Results.csv

[11]:
```
dataframes = read_s3_files_as_dataframes('mission-green-bucket','Drinking_Water_Quality_Data')
for i, df in enumerate(dataframes):
    final_water_df = df
```

File 'Drinking_Water_Quality_Data/Self_Reported_Drinking_Water_Tank_Inspection_Results.csv' read and stored as DataFrame.

[12]:
```
pd.set_option('display.max_columns', 50)
final_water_df
```

t[12]:

	bin	borough	zip	house_num	street_name	block	lot	confirmation_num	reporting_year	tank_num	in
0	1035071.0	MANHATTAN	10019.0	9	West 57th Street	1273	22	WTI8972067865	2020	1	
1	1080609.0	MANHATTAN	10018.0	1372	BROADWAY	813	23	WTI0835829410	2021	1	
2	1087665.0	MANHATTAN	10128.0	333	East 91 Street	1554	23	WTI2822978287	2020	1	
3	1003881.0	MANHATTAN	10002.0	50	Eldridge street	300	8	WTI8346971831	2018	1	
4	4030340.0	QUEENS	11377.0	40-33	69TH STREET	1301	10	WTI9251583006	2016	1	
...	...	...	...	...	...	...	...	...	...	...	...
49795	1014412.0	MANHATTAN	10001.0	212	WEST 35 STREET	784	51	WTI4702638830	2022	1	
49796	1085671.0	MANHATTAN	10017.0	450	Lexington Avenue	1280	90	WTI5762173305	2024	1	
49797	1015862.0	MANHATTAN	10118.0	350	5th Ave	835	41	WTI1896490394	2019	6	

- Data Dictionary: DOHMH_WaterTankInspection_DataDictionary.xlsx
- Integration: Batch Migration

C. Git link for both datasets. - Data

D. Data Profiling - Data_Profiling.xlsx

3. Platform to use:

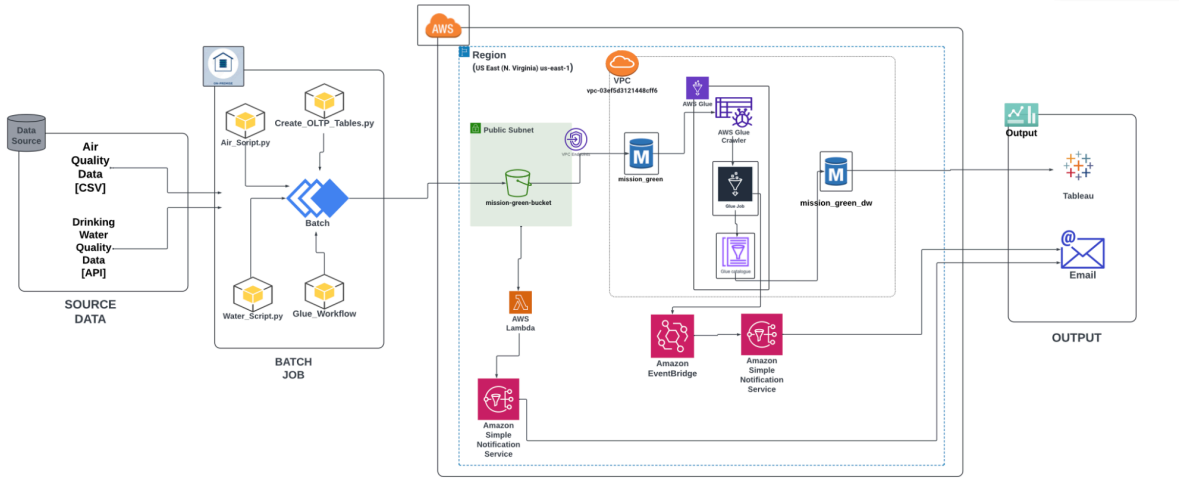
A. Amazon Web Services

- Services to be used:
  - AWS SNS
  - AWS EventBridge
  - AWS IAM
  - AWS RDS
  - AWS VPC
  - AWS S3
  - AWS Glue
  - AWS Lambda

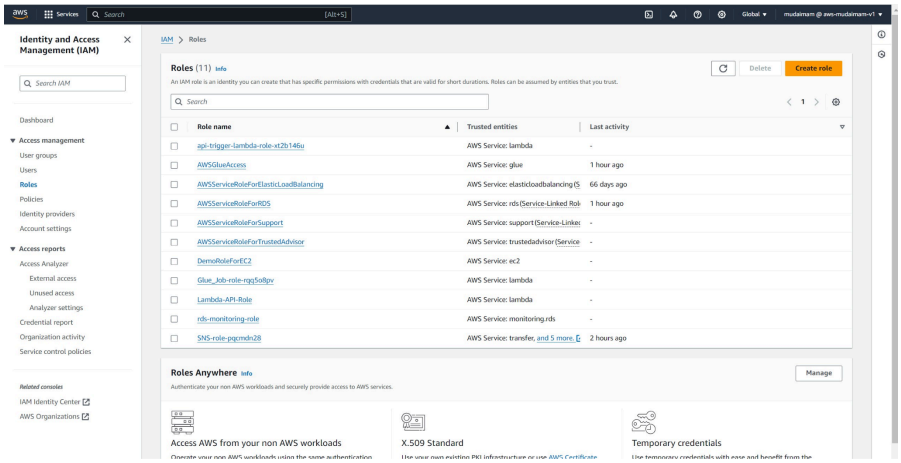
B. Tableau for Visualization

DAV 6100 IA Final Project Report Spring 2024

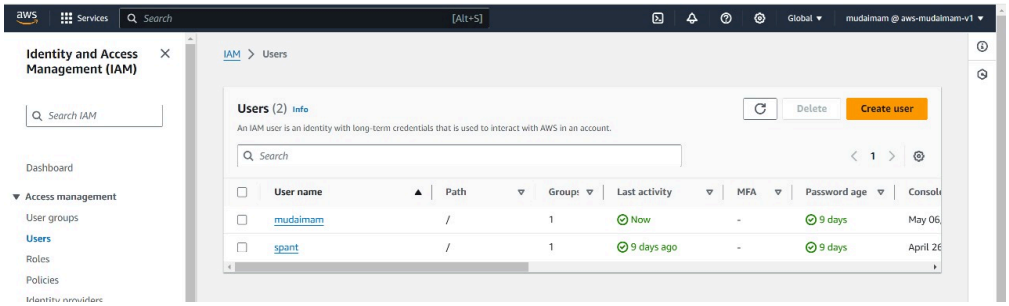
4. AWS Architecture:

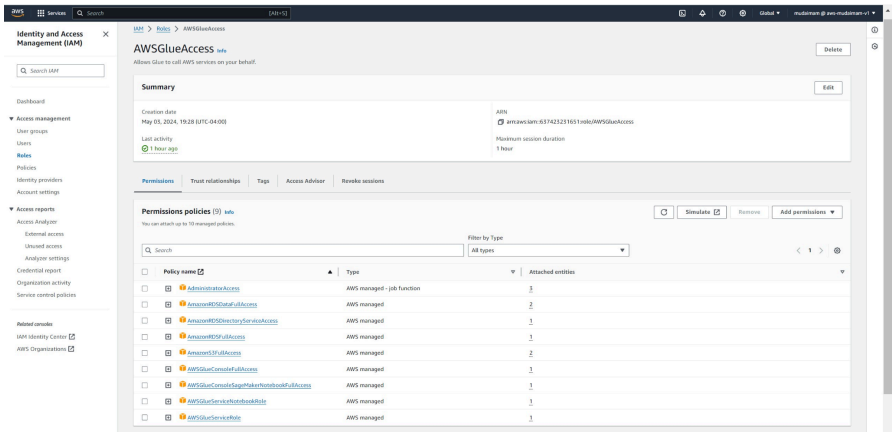


5. IAM Roles:



- **IAM users:**



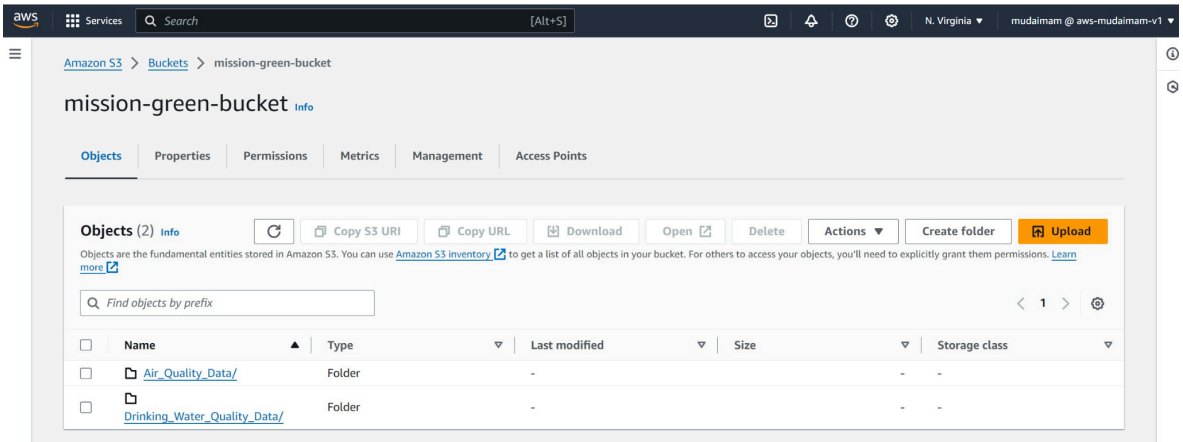


6. Data loading to AWS S3:

- Code: [Data loading to S3]
- Output:

```
In [6]: # Validate the files stored in s3 bucket
        print_bucket_content("mission-green-bucket")
```

```
Contents of the mission-green-bucket bucket
Air_Quality_Data/CO/ad_viz_plotval_data_CO_2020.csv
Air_Quality_Data/CO/ad_viz_plotval_data_CO_2021.csv
Air_Quality_Data/CO/ad_viz_plotval_data_CO_2022.csv
Air_Quality_Data/CO/ad_viz_plotval_data_CO_2023.csv
Air_Quality_Data/NO2/ad_viz_plotval_data_NO2_2020.csv
Air_Quality_Data/NO2/ad_viz_plotval_data_NO2_2021.csv
Air_Quality_Data/NO2/ad_viz_plotval_data_NO2_2022.csv
Air_Quality_Data/NO2/ad_viz_plotval_data_NO2_2023.csv
Air_Quality_Data/Ozone/ad_viz_plotval_data_Ozone_2020.csv
Air_Quality_Data/Ozone/ad_viz_plotval_data_Ozone_2021.csv
Air_Quality_Data/Ozone/ad_viz_plotval_data_Ozone_2022.csv
Air_Quality_Data/Ozone/ad_viz_plotval_data_Ozone_2023.csv
Air_Quality_Data/Ozone/ad_viz_plotval_data_Ozone_2024.csv
Air_Quality_Data/PM10/ad_viz_plotval_data_PM10_2020.csv
Air_Quality_Data/PM10/ad_viz_plotval_data_PM10_2021.csv
Air_Quality_Data/PM10/ad_viz_plotval_data_PM10_2022.csv
Air_Quality_Data/PM10/ad_viz_plotval_data_PM10_2023.csv
Air_Quality_Data/PM10/ad_viz_plotval_data_PM10_2024.csv
Air_Quality_Data/PM2.5/ad_viz_plotval_data_PM2.5_2020.csv
Air_Quality_Data/PM2.5/ad_viz_plotval_data_PM2.5_2021.csv
Air_Quality_Data/PM2.5/ad_viz_plotval_data_PM2.5_2022.csv
Air_Quality_Data/PM2.5/ad_viz_plotval_data_PM2.5_2023.csv
Air_Quality_Data/PM2.5/ad_viz_plotval_data_PM2.5_2024.csv
Air_Quality_Data/SO2/ad_viz_plotval_data_SO2_2020.csv
Air_Quality_Data/SO2/ad_viz_plotval_data_SO2_2021.csv
Air_Quality_Data/SO2/ad_viz_plotval_data_SO2_2022.csv
Air_Quality_Data/SO2/ad_viz_plotval_data_SO2_2023.csv
Drinking_Water_Quality_Data/Self_Reported_Drinking_Water_Tank_Inspection_Results.csv
```



7. Storing both raw datasets to RDS - (mission_green) OLTP:

- Code: Air_Script.ipynb
- Output:

```
# Connection details for RDS (PostgreSQL example)
db_type = "mysql"
db_user = ""
db_password = ""
db_host = "database-1.cnq4kwysqjdb.us-east-1.rds.amazonaws.com"  # RDS endpoint
db_port = 3306  # Default PostgreSQL port
db_name = "mission_green"

# Create a database engine using SQLAlchemy
connection_string = f"{db_type}://{db_user}:{db_password}@{db_host}:{db_port}/{db_name}"
engine = create_engine(connection_string)

# Write DataFrame to the RDS table
table_name1 = "Air_Quality_Data"  # Name of the RDS table to write to

# Write DataFrame to the SQL database
merged_df.to_sql(table_name1, engine, if_exists="replace", index=False)  # index=False to avoid creating index co

print(f"DataFrame written to {table_name1} in RDS database.")
```

```
DataFrame written to Air_Quality_Data in RDS database.
```

- Code: Water_Script.ipynb
- Output:

```
# Connection details for RDS (PostgreSQL example)
db_type = "mysql"
db_user = ""
db_password = ""
db_host = "database-1.cnq4kwysqjdb.us-east-1.rds.amazonaws.com"  # RDS endpoint
db_port = 3306  # Default PostgreSQL port
db_name = "mission_green"

# Create a database engine using SQLAlchemy
connection_string = f"{db_type}://{db_user}:{db_password}@{db_host}:{db_port}/{db_name}"
engine = create_engine(connection_string)

table_name2 = "Drinking_Water_Quality_Data"  # Name of the RDS table to write to

# Write DataFrame to the SQL database
final_water_df.to_sql(table_name2, engine, if_exists="replace", index=False)  # index=False to avoid creating inc

print(f"DataFrame written to {table_name2} in RDS database.")
```

```
DataFrame written to Drinking_Water_Quality_Data in RDS database.
```

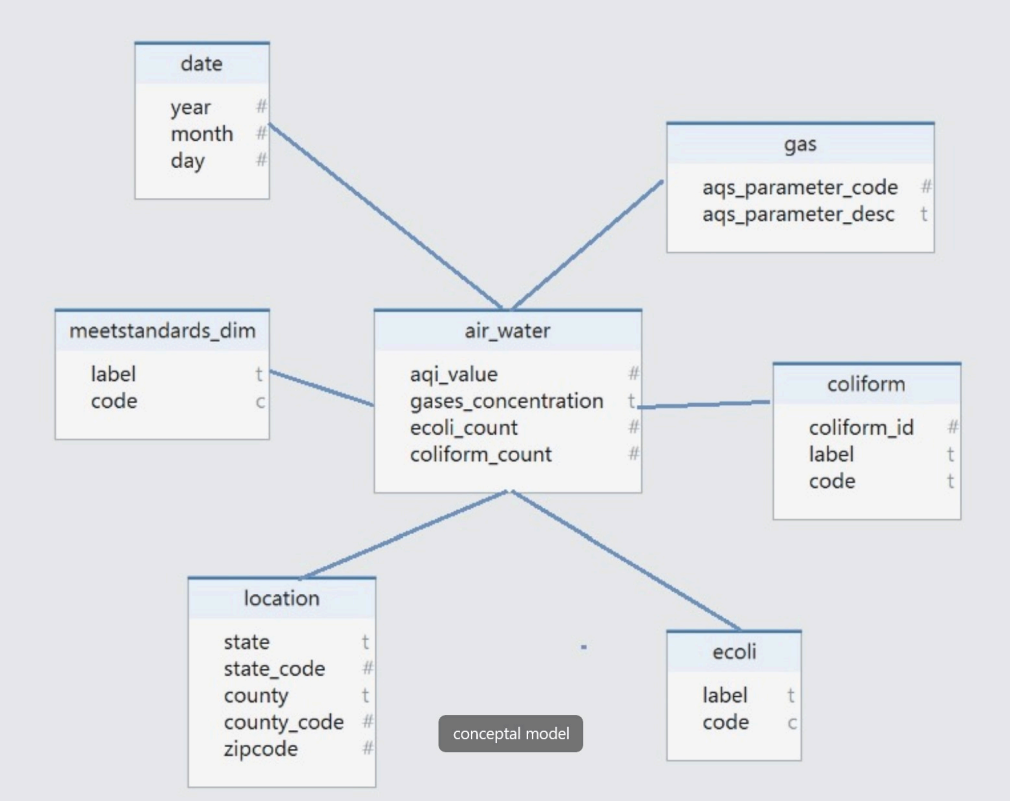
8. Data Modeling:

- Kimball's 4 step process to create dimensional data modeling:
  - A. Selecting the Business Process:
    - The business process selected addresses the challenge of locating areas with clean air and drinkable water.
  - B. Declare the Grain:
    - Define the level of detail required for analysis. For this project, daily air quality measurements and inspection results for drinking water tanks.
  - C. Choose the Dimensions:
    - Date

- Gas
- Location
- E Coli
- Coliform
- Meet Standards

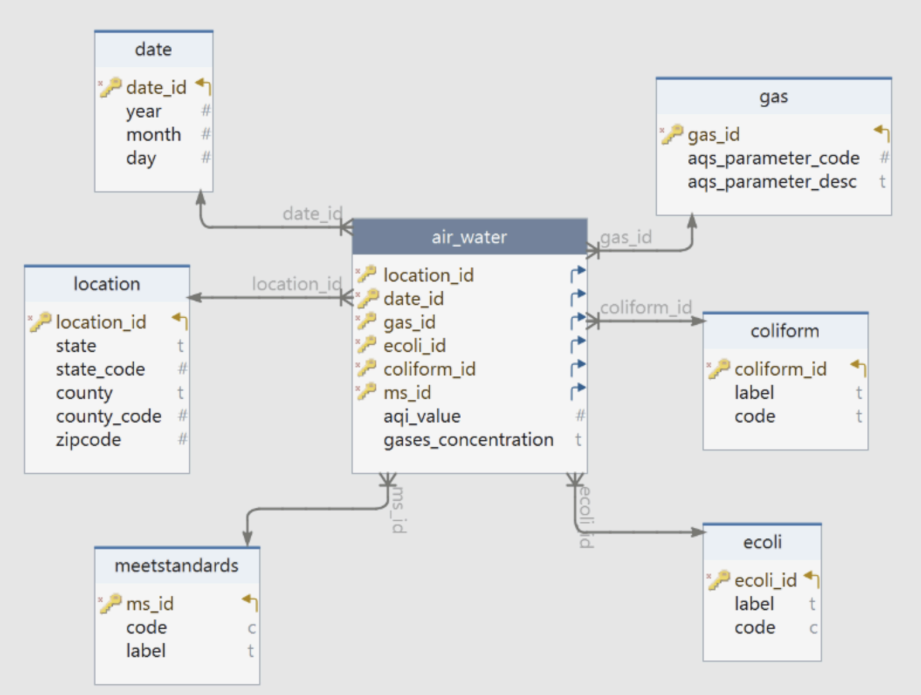
D. Identify the Facts:
- AQI and Water Inspection Results

- Conceptual Model:

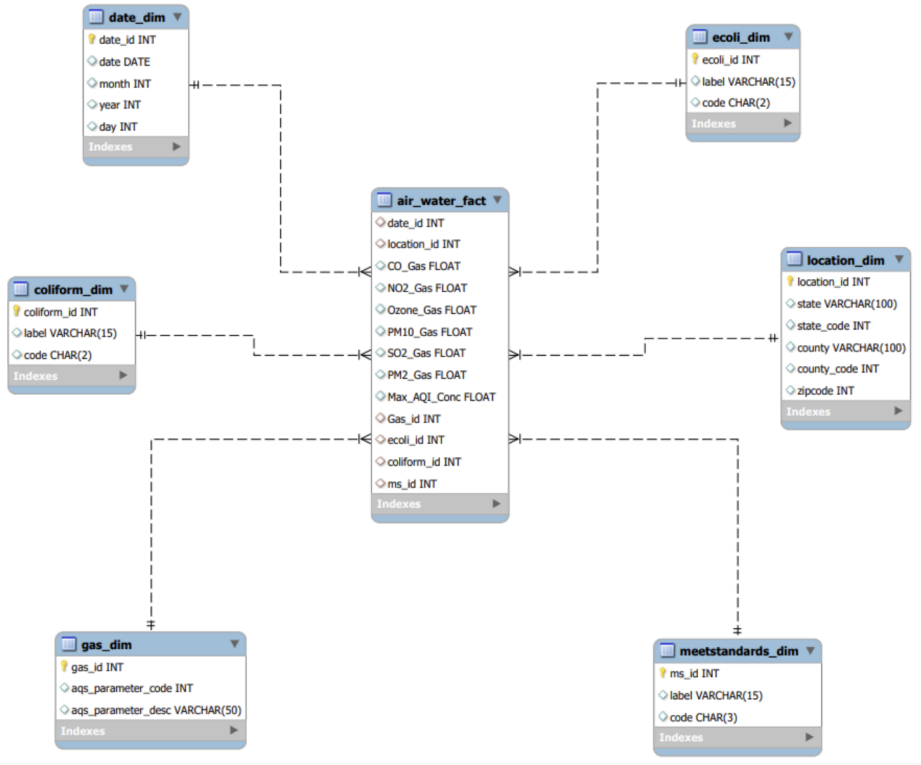


DAV 6100 IA Final Project Report Spring 2024

- Logical Model:



- Physical Model:

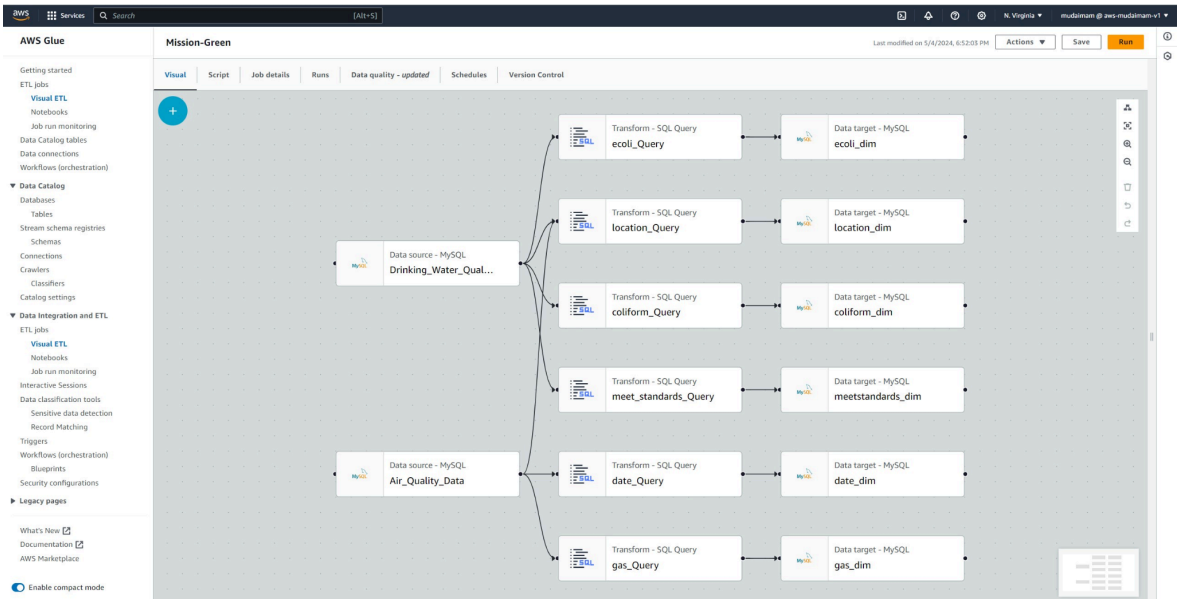


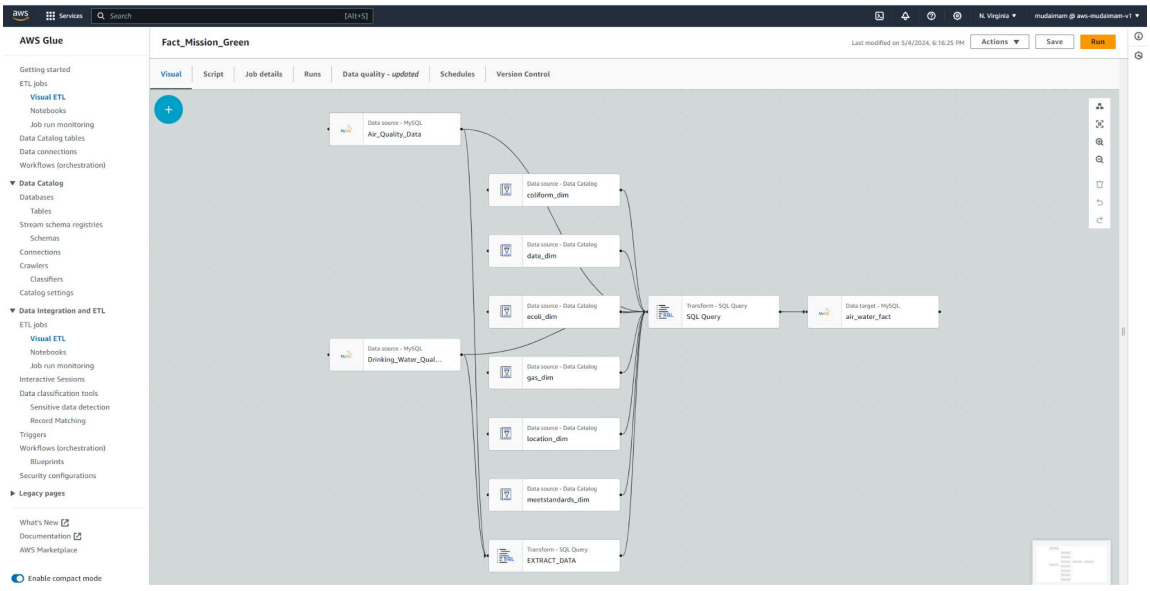
DAV 6100 IA Final Project Report Spring 2024

Bus Matrix: [Bus Matrix Group 3.xls]

- ***Note:*** Implementing **Slowly Changing Dimensions (SCD)** in this project does not align with the problem statement due to the high volatility of air quality and drinking water inspection data, which could result in complexity, performance issues, and potential compromise of data quality.

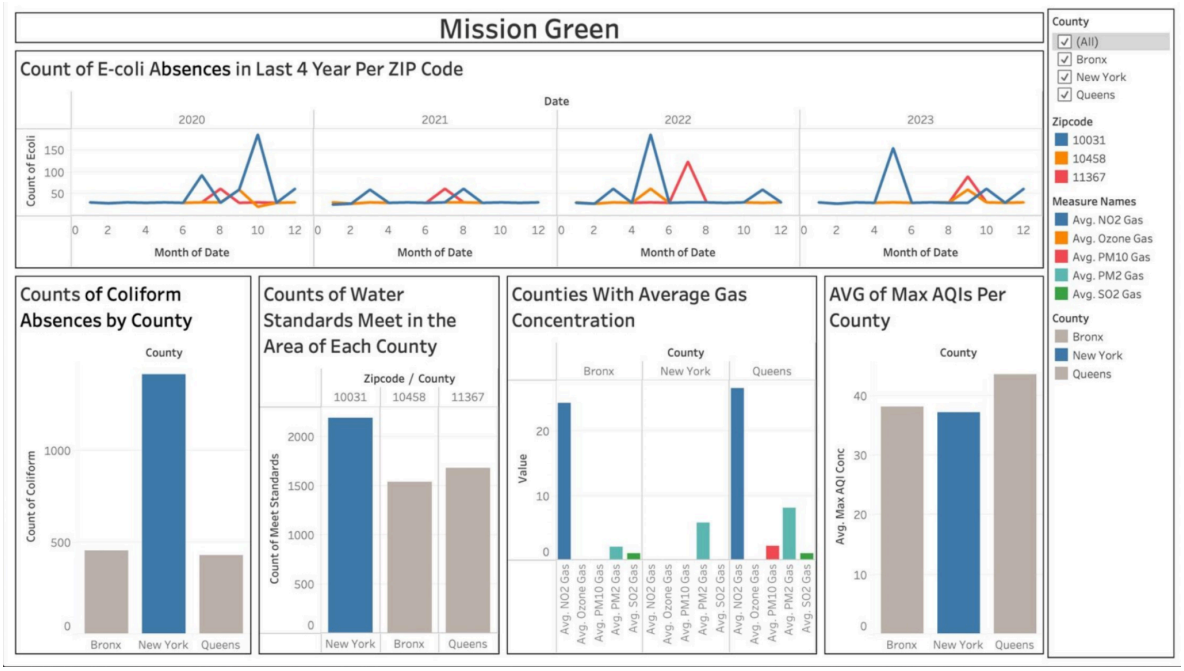
9. AWS Glue - ETL Process: OLTP to OLAP





DAV 6100 IA Final Project Report Spring 2024

10. VIsualization using Tableau: [IA Final Project Group 3.twbx]



11. Recommendations from visualization:

- According to the visualization, Mission Green has effectively implemented the proposed architecture in the area. Our recommendation to stakeholders is to focus on New York County, as it exhibits the lowest presence of E. coli samples among all localities. Additionally, New York County boasts lower Air Quality Index (AQI) levels for major pollutants compared to other areas in the state.

12. Challenges:

- AWS Lambda integration: Troubleshooting and monitoring distributed lambda functions across multiple services can be complex.
- Data scarcity: With insufficient data, it's challenging to extract meaningful patterns, resulting in incomplete or inaccurate analyses.
- API limitations: Restrictions on the usage of zip conversion APIs may hinder data processing and integration, impacting the efficiency of workflows.
- AWS Glue budget constraints: Limited budget allocations for AWS Glue services can restrict data transformation capabilities and scalability, posing challenges in managing large datasets effectively.

13. Git Progression: Project versioning

- During the initial development phase, Git will track progress from feature implementation to finalization, all consolidated within the 'main' branch, ensuring seamless collaboration and version control.

14. Lessons learned and improvement:

- Lambda for ETL: Leveraging lambda for ETL tasks can enhance flexibility and scalability compared to traditional approaches, allowing for more efficient data processing.
- GLUE for migration: Limiting GLUE usage to migration tasks helps streamline the ETL process and avoids unnecessary complexity in the data pipeline.
- Data acquisition: Actively seeking and acquiring more data can enrich analytics and insights, contributing to better decision-making and problem-solving.
- SNS notification optimization: Optimizing SNS notifications at the RDS level instead of S3 can improve monitoring efficiency and reduce noise in alerts.