**UNIVERSITY OF GREENWICH**

**NAME: OLUWASEGUNN LAWAL**

**STUDENT ID: 001276114**

**MSC Big Data and Business Intelligence**

**Course: Data Warehousing and Business Intelligence Coursework**

**COMP1848**

**GROUP NUMBER: 14**

| NAME | ID | CONTRIBUTION |
|---|---|---|
| Oluwasegun Lawal | 001276114 | Schema creation including staging table |
| Kehinde Ajani | 1356803 | Data Cleaning |
| Mahendra Ramlal Choudhary | 1366460 | SQL |
| Toluwaleyi Abayomi Oke | 1421167 | SQL |

**Course Leader: Hooman Oroojeni**

# Table of Contents

## INTRODUCTION

In this project, my team and I worked together to develop a data warehouse for the Air Quality data in Oracle SQL Developer. Our task cantered around designing a star schema to facilitate efficient data analysis and reporting. This schema comprises a central fact table, **FACT_AIR_QUALITY**, and several supporting dimension tables, such as ***DIM_LOCATION*****, DIM_TIME, *****DIM_WEATHER*****, *****DIM_CONTROL_LEVEL*****, *****DIM_POLLUTANT*****, and *****DIM_TRAFFIC_POPULATION***. These tables collectively structure the data in a manner that optimizes query performance and enables detailed insights into air quality metrics. The primary objective of this coursework is to construct a data warehouse capable of effectively monitoring air quality within metropolitan regions. By leveraging various attributes of the data stored in **Tpm2.5**, I aim to enable comprehensive and efficient analyses. This data warehouse will serve as a pivotal tool in supporting efforts to monitor and enhance environmental well-being through robust air quality control measures.

## Group Task: STAR SCHEMA CREATION

To design and implement a comprehensive data warehouse for analyzing air quality, my team and I adopted a star schema architecture, which centralizes measurable metrics in a fact table while contextualizing data with dimension tables. This schema ensures analytical flexibility, efficient querying, and support for multi-dimensional analysis. Additionally, a **staging table** was introduced to preprocess and clean raw data before populating the dimension and fact tables.

### Staging Table Design

The **STAGING_AIR_QUALITY** table served as a temporary repository to facilitate data preprocessing and transformation. This table was designed to hold raw data extracted from the source system, including all relevant fields such as timestamps, pollutant names, pollutant values, weather conditions, and station information. The schema of the staging table included columns for:

1. **Quantitative Attributes**: Pollutant levels, wind speed, visibility, temperature, and humidity.
2. **Qualitative Attributes**: Weather conditions, traffic levels, and control levels.
3. **Geographical Data**: Station names and coordinates (latitude, longitude).
4. **Identifiers**: Record IDs and redundant station information for traceability.

The preprocessing phase involved several key steps:

- Identifying and handling missing values by replacing null values in numerical columns with their respective means (average value), and categorical columns with their modes.
- Ensuring data consistency by aligning the raw data structure with the schema design.
- Removing duplicates and redundant records to maintain data integrity.

### Fact Table Design

The **FACT_AIR_QUALITY** table is the central component of the star schema and was designed to consolidate statistical and measurable data related to air quality. The schema includes:

- **Pollutant Levels**: The quantitative air quality metrics being analysed.
- **Foreign Keys**: Links to dimension tables such as LOCATION_ID, WEATHER_ID, TRAFFIC_ID, CONTROL_LEVEL_ID, POLLUTANT_ID, and TIME_ID. These references ensure that the fact table integrates seamlessly with contextual information from the dimension tables.

The fact table plays a pivotal role in enabling comprehensive analyses, such as evaluating pollutant trends, studying seasonal variations, and assessing the impact of human activity and environmental factors on air quality.

**Dimension Table Design**

To provide rich context to the fact table, six-dimension tables were developed:

1. **DIM_LOCATION**:

   Captures station-level geographical data, including station names, latitude, and longitude, enabling spatial analyses.

2. **DIM_POLLUTANT**:

   Stores pollutant-specific details, such as names and placeholder descriptions, facilitating pollutant-specific analyses.

3. **DIM_WEATHER**:

   Includes environmental attributes such as temperature, humidity, wind speed, visibility, and weather conditions. This table supports analyses of how weather impacts air quality.

4. **DIM_TRAFFIC_POPULATION**:

   Focuses on human activity metrics like traffic levels, vehicle counts, and population density, providing insights into anthropogenic influences on pollution.

5. **DIM_CONTROL_LEVEL**:

   Stores data related to air quality control measures, including levels and descriptions of implemented interventions.

6. **DIM_TIME**:

   Provides a temporal framework for analysis, with attributes such as day, month, and hour. This table enables time-based queries, such as identifying monthly trends or seasonal variations in pollution.

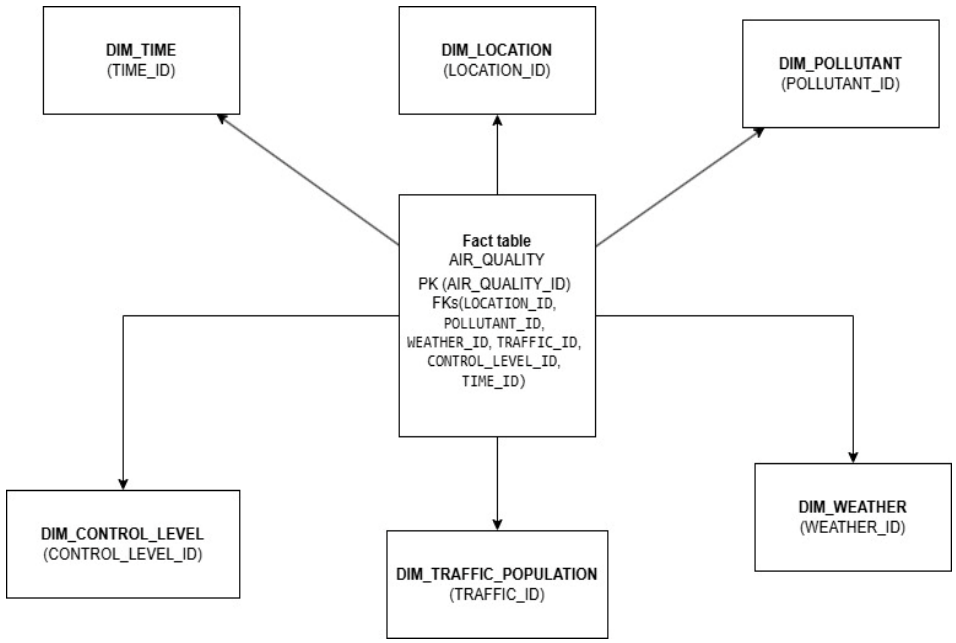



FIG: Star schema diagram illustrating the fact table, and dimension tables.

The star schema centers around the **AIR_QUALITY** fact table, linked to six-dimension tables via foreign keys (e.g., LOCATION_ID, TIME_ID). Dimension tables provide descriptive data, like location, weather, and time, while the fact table consolidates measurable metrics. This design simplifies analysis and ensures efficient querying.

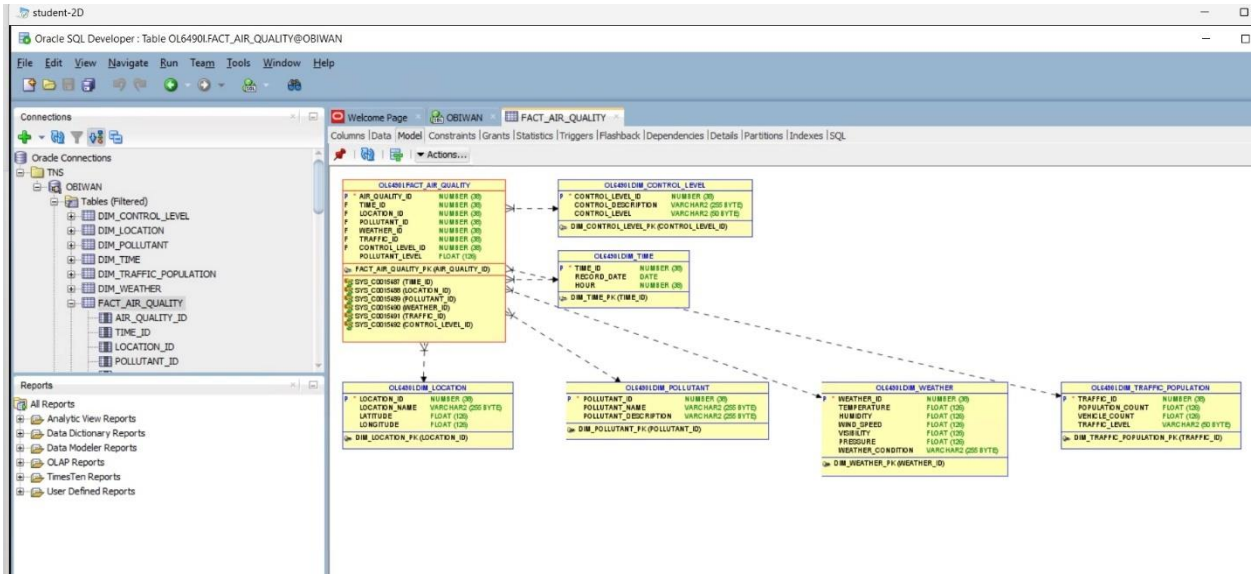



Fig: Screenshot of the fact table in SQL Developer (fact table and dimensions table).

## BUS PLAN FOR THE AIR QUALITY DATA WAREHOUSE

A bus plan is crucial for establishing a clear and consistent framework for integrating and organizing data across various business processes, ensuring alignment with analytical goals and scalability in data warehousing. The bus plan I created includes the following:

### Context of the Business:

Air quality monitoring plays a vital role in assessing pollution levels, identifying trends, and predicting future conditions. My primary objective in this project is to design a data warehouse that integrates sensor data from various locations into a robust schema to enable comprehensive analysis. This includes understanding pollution trends over time, pinpointing critical hotspots, forecasting future pollution levels, and ensuring compliance with environmental regulations. Given the challenges posed by noisy and incomplete raw sensor data, I focused on implementing an ETL process to clean, transform, and organize the data, ensuring its reliability and accuracy for analysis.

### Grain of The Fact Table:

I defined the grain of the AIR_QUALITY fact table as one record per monitoring station, pollutant type, and timestamp. This level of granularity enables detailed tracking of pollutant readings across specific times and locations, ensuring precise and comprehensive analysis.

**Dimensions:**

| Dimension Table | Attributes |
|---|---|
| DIM_LOCATION | Location_ID, Station_ID, Station_Name, Latitude, Longitude, Installation_Date |
| DIM_TIME | Time_ID, Date, Hour, Day, Month, Year, Season, Timestamp |
| DIM_POLLUTANT | Pollutant_ID, Pollutant_Type (e.g., PM2.5, $NO_2$, CO), Unit, Pollutant_Threshold |
| DIM_WEATHER | Weather_ID, Temperature, Humidity, Wind_Speed, Visibility, Pressure, Weather_Condition |
| DIM_TRAFFIC_POPULATION | Traffic_ID, Traffic_Level, Vehicle_Count, Population_Density, Nearby_Factories |
| DIM_CONTROL_LEVEL | Control_ID, Control_Level, Measures_Implemented, Density_Absorb, GCC |

The table above enriches the fact table with contextual information.

**Fact Table**

The **AIR_QUALITY** fact table consolidates quantitative data linked to dimensions:

- **Attributes**: Air_Quality_ID (Primary Key), Location_ID (FK), Pollutant_ID (FK), Weather_ID (FK), Traffic_ID (FK), Control_ID (FK), Time_ID (FK), Pollution_Value.

**Business Process**

The core business processes supported by the data warehouse are:

| Business Process | Fact Table Grain | Dimensions Involved |
|---|---|---|
| Monitoring Pollution Trends | Per-station, pollutant, and timestamp | Time, Location, Pollutant |
| Weather Impact on Pollution | Per station and timestamp | Time, Location, Weather |
| Traffic and Population Influence | Per station, daily traffic, and population | Time, Location, Traffic and Population |
| Control Measures Effectiveness | Per station and timestamp | Time, Location, Control Level |
| Regulatory Compliance | Per pollutant and location over time | Time, Location, Pollutant |

**ETL and Data Cleaning Considerations:**

In my initial approach to data cleaning, I aimed at addressed several key issues to enhance data integrity. For missing values, I input numerical columns like Pollution_Value and Temperature with their mean, and categorical columns like Weather_Condition and Traffic_Level with their mode. I ensured all readings fell within valid ranges, validating pollutant thresholds, temperatures (-50°C to 60°C), and humidity levels (0% to 100%). Duplicate records were removed to avoid redundancy. Additionally, irrelevant data, such as the Redundant_Station_ID and Redundant_Station_Name columns, can be dropped. Finally, I enforced data consistency by aligning units across records, ensuring pollutants were uniformly measured in $\mu g/m^3$.

**Bus Matrix**

| Dimension Table | Monitoring Pollution Trends | Weather Impact Analysis | Traffic/Population Analysis | Control Effectiveness | Compliance |
|---|---|---|---|---|---|
| DIM_LOCATION | ✓ | ✓ | ✓ | ✓ | ✓ |
| DIM_TIME | ✓ | ✓ | ✓ | ✓ | ✓ |
| DIM_POLLUTANT | ✓ | | | | ✓ |
| DIM_WEATHER | | ✓ | | | |
| DIM_TRAFFIC_POPULATION | | | ✓ | | |
| DIM_CONTROL_LEVEL | | | | ✓ | |

In designing the bus matrix, I mapped dimensions to key business processes to ensure comprehensive analytical coverage. The matrix highlights how each dimension supports processes such as monitoring pollution trends, analyzing weather impacts, examining traffic and population influences, evaluating control

measures, and ensuring compliance. Core dimensions like DIM_LOCATION and DIM_TIME are pivotal across all processes, while others, such as DIM_POLLUTANT and DIM_WEATHER, provide targeted insights. This structured approach ensures the data warehouse is aligned with analytical goals, enabling a nuanced understanding of air quality dynamics.

## ETA (Extract, Transform, Loading): Data Exporting into the Oracle (obiwan)

My group and I began by converting the dataset from an ACCDB file to a CSV format as our initial step. Afterward, we carefully inspected the dataset and reviewed the CSV file to understand its structure and contents. We identified and categorized the information into appropriate tables, mapping columns such as **Station_ID** and **Timestamp** to the **DIM_LOCATION** and **DIM_TIME** dimension tables, respectively. We also categorized **Pollutant** to the **DIM_POLLUTANT** table, ensuring each dataset was grouped logically into dimension and fact tables, a crucial step before loading the CSV into our Oracle Developer environment, OBIWAN, for further processing.

The ETL (Extract, Transform, Load) process was designed to ensure the raw air quality data was accurately transformed into a clean, consistent format for analytical purposes within a star schema-based data warehouse. This process involved three stages: **data staging and extraction**, **data cleaning and transformation**, and **data integration into dimension and fact tables**.

### 1. Data Staging and Extraction

The staging table, STAGING_AIR_QUALITY, was created to serve as a temporary repository for raw data. It was designed with a comprehensive schema to accommodate all key attributes, such as station details, pollutant measurements, weather conditions, traffic data, and geographical coordinates. The staging table provided a workspace where the data could be cleaned and pre-processed before populating the dimension and fact tables. The below figure illustrates the staging table model in the SQL developer.

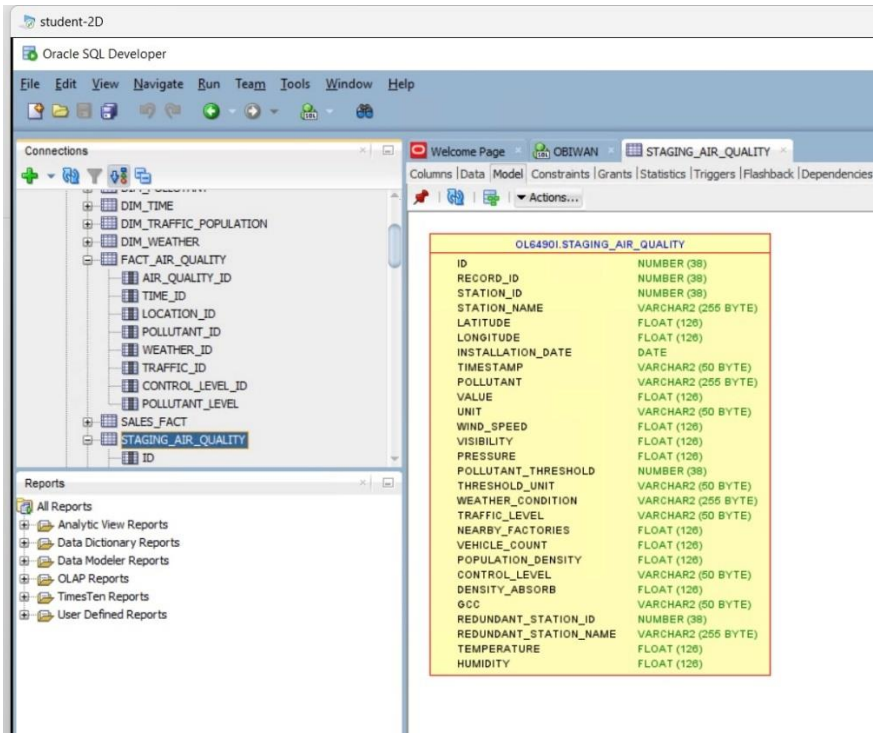



Fig: Staging table.

## 2. Data Cleaning

Before commencing the data cleaning process, I performed a comprehensive query to assess the prevalence of missing or invalid readings across all stations within the dataset. This preliminary analysis uncovered notable data quality issues, including missing or anomalous values for key attributes such as pollutant levels, temperature, and humidity. For instance, the North London station exhibited the highest number of anomalies (34), followed by Central London (24) and South London (23). These anomalies included null pollutant readings, temperatures falling outside the acceptable range of -50 to 60 degrees Celsius, and humidity levels exceeding logical boundaries (below 0% or above 100%).

To ensure the quality and integrity of the data, a systematic cleaning process was conducted:

- **Handling Missing Values:**
    - Numerical attributes like VALUE, TEMPERATURE, and WIND_SPEED were updated with their mean values, ensuring the imputation of meaningful data for analysis.

- Categorical attributes, including WEATHER_CONDITION and TRAFFIC_LEVEL, were replaced with their mode (most frequent value), maintaining representational accuracy.

- **Validating Data Ranges:**
  - Key metrics were validated to ensure they fell within reasonable and acceptable ranges. For instance, temperatures were restricted to -50°C to 60°C, and humidity levels to 0% to 100%. Any outliers were identified and addressed accordingly.
- **Removing Duplicates:**
  - Duplicate records across both the staging and dimension tables were identified and removed to prevent redundancy and ensure data uniqueness.
- **Dropping Irrelevant Data:**
  - Columns such as REDUNDANT_STATION_ID and REDUNDANT_STATION_NAME, which were not relevant for the analysis, were removed to streamline the data structure.
- **Ensuring Consistency:**
  - Units of measurement were standardized across all records. For example, pollutant concentrations were aligned to a uniform unit (µg/m³), ensuring comparability and consistency.

By transitioning from the raw table to the staging table and implementing thorough data cleaning measures, I ensured the dataset's robustness and readiness for subsequent transformations and analyses. This meticulous approach laid the groundwork for generating accurate insights and making well-informed decisions in the later stages of the project.

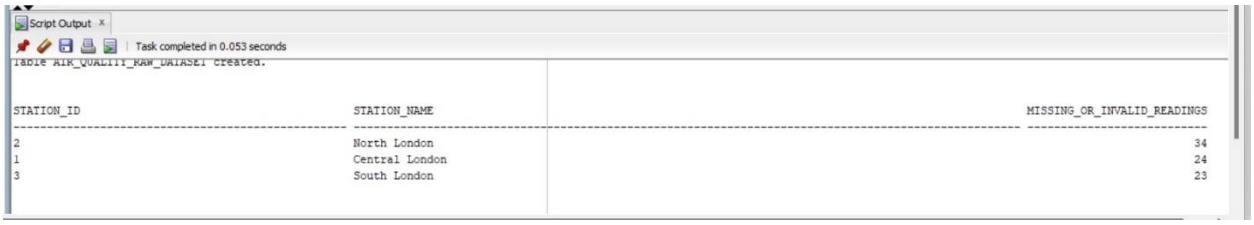



Fig: Screenshot illustrating the number of missing or invalid readings per station before cleaning.

**3. Data Integration (Organize Data into Fact and Dimension Tables)**

After the data was cleaned and validated in the staging table, it was systematically loaded into the dimension tables. Each dimension table was designed to store descriptive information:

- DIM_LOCATION for station details and geographical attributes.
- DIM_WEATHER for environmental conditions.
- DIM_POLLUTANT for pollutant-specific information.
- DIM_TRAFFIC_POPULATION for traffic and population metrics.
- DIM_CONTROL_LEVEL for air quality control measures.
- DIM_TIME for temporal data such as date, hour, and season.
  The fact table, FACT_AIR_QUALITY, was then populated by referencing the dimension tables through their respective foreign keys. This ensured a cohesive integration of measurable metrics (e.g., pollutant levels) with their contextual dimensions (e.g., time, location, weather).

**Confirming the validity of the Warehouse to Perform the following queries:**

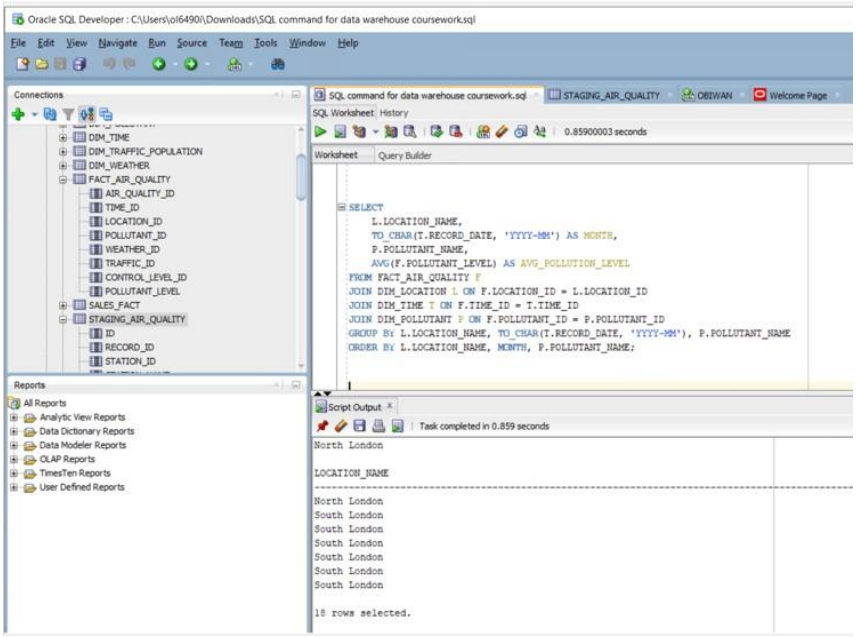
1. **Average pollution level (PM2.5, $NO_2$, etc.) by location per month.**

Using the SQL Command:

```
SELECT
    L.LOCATION_NAME,
    TO_CHAR(T.RECORD_DATE, 'YYYY-MM') AS MONTH,
    P.POLLUTANT_NAME,
    AVG(F.POLLUTANT_LEVEL) AS AVG_POLLUTION_LEVEL
FROM FACT_AIR_QUALITY F
JOIN DIM_LOCATION L ON F.LOCATION_ID = L.LOCATION_ID
JOIN DIM_TIME T ON F.TIME_ID = T.TIME_ID
JOIN DIM_POLLUTANT P ON F.POLLUTANT_ID = P.POLLUTANT_ID
GROUP    BY    L.LOCATION_NAME,    TO_CHAR(T.RECORD_DATE,    'YYYY-MM'),
P.POLLUTANT_NAME
ORDER BY L.LOCATION_NAME, MONTH, P.POLLUTANT_NAME;
```

To analyse air quality trends, I wrote an SQL query to calculate the average pollution levels by location and month. This query joins the fact table, FACT_AIR_QUALITY, with dimension tables (DIM_LOCATION, DIM_TIME, and DIM_POLLUTANT) to aggregate pollutant data (e.g., CO, $NO_2$, PM10) for each location, grouped by month. The resulting dataset offers insights into pollution patterns, enabling a better understanding of air quality variations across regions and time periods.

However, due to the extensive nature of the results, capturing the entire table in a single screenshot was not feasible, as it exceeded the display capacity of the interface. To address this, I condensed the data into a shorter table format, highlighting key findings for clarity and ease of interpretation. This summarized table still retains the most critical details, including location names, pollutant types, and their average pollution levels, ensuring it effectively supports the objectives of this analysis.



### 2. Highest Pollution Levels by Day for Each Pollutant

To identify the highest pollution levels by day for each pollutant, I followed a systematic approach using SQL. First, I utilized a SELECT statement to retrieve key information, including the record date, pollutant name, and the maximum pollution level. I applied the MAX() function to determine the highest pollution levels recorded for each pollutant on a given day.

To ensure accurate grouping, I incorporated a GROUP BY clause, which allowed me to aggregate data by date and pollutant name. This grouping was essential for calculating the maximum values specific to each pollutant on each day. Finally, I added an ORDER BY clause to organize the results chronologically by date and alphabetically by pollutant name, making the output more readable and easier to interpret.

This query enabled me to effectively identify the most severe pollution levels across all pollutants and days in the dataset, providing a valuable foundation for analysing extreme pollution events and their potential causes.

SQL Command:

```
SELECT
    T.RECORD_DATE,
    P.POLLUTANT_NAME,
    MAX(F.POLLUTANT_LEVEL) AS MAX_POLLUTION_LEVEL
FROM FACT_AIR_QUALITY F
JOIN DIM_TIME T ON F.TIME_ID = T.TIME_ID
JOIN DIM_POLLUTANT P ON F.POLLUTANT_ID = P.POLLUTANT_ID
GROUP BY T.RECORD_DATE, P.POLLUTANT_NAME
ORDER BY T.RECORD_DATE, P.POLLUTANT_NAME;
```

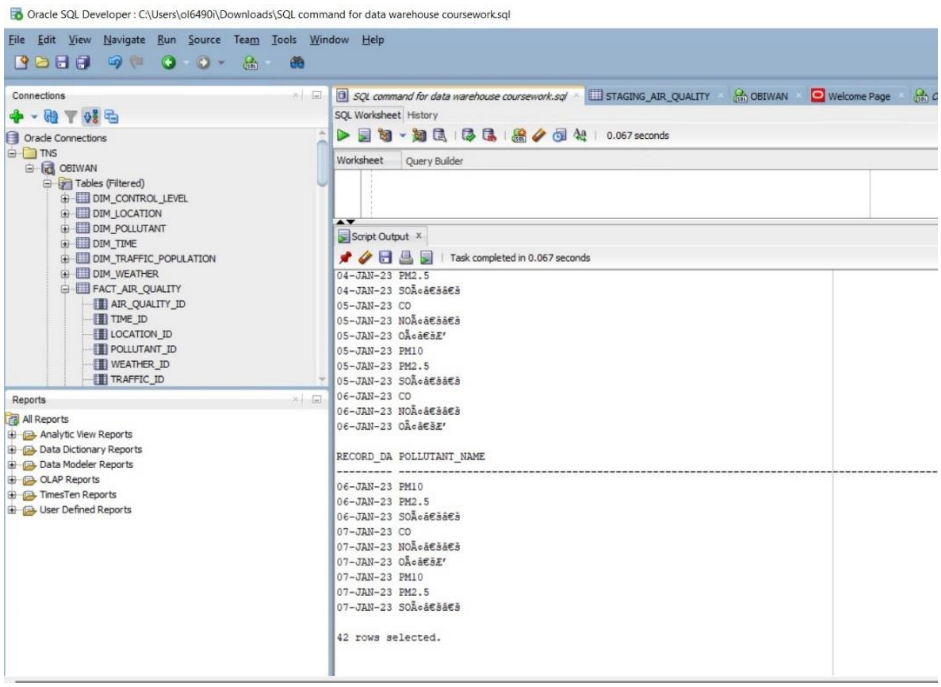



Fig: Showing the result of the highest pollutions level by day for each pollutants

**Explanation of results after running this query:**

The result of the query reveals the highest pollution levels recorded for each pollutant on specific dates. For example, on January 1, 2023, the highest recorded pollution levels include 24.76 µg/m³ for CO, 79.4 µg/m³ for $NO_2$, and 52.2 µg/m³ for $O_3$. Similarly, January 2, 2023, shows that the highest levels were 1.4 µg/m³ for CO and 76.4 µg/m³ for $NO_2$.

This analysis highlights the most severe pollution concentrations by pollutant on a daily basis. It provides a critical understanding of peak pollution events, which can be used for environmental monitoring and decision-making. These maximum values help identify specific days with extreme pollution levels and allow for the investigation of potential causes, such as weather conditions, traffic density, or industrial activities.

### 3. Analyzing Seasonal Trends in Pollution Levels Using SQL

SQL Command:

```
SELECT
    TO_CHAR(T.RECORD_DATE, 'YYYY-MM') AS MONTH,
    P.POLLUTANT_NAME,
    AVG(F.POLLUTANT_LEVEL) AS AVG_POLLUTION_LEVEL
FROM FACT_AIR_QUALITY F
JOIN DIM_TIME T ON F.TIME_ID = T.TIME_ID
JOIN DIM_POLLUTANT P ON F.POLLUTANT_ID = P.POLLUTANT_ID
GROUP BY TO_CHAR(T.RECORD_DATE, 'YYYY-MM'), P.POLLUTANT_NAME
ORDER BY MONTH, P.POLLUTANT_NAME;
```

This query calculates the average pollution levels for each pollutant across months, providing insights into trends and seasonal variations. By joining the FACT_AIR_QUALITY table with DIM_TIME and DIM_POLLUTANT, I accessed time-specific and pollutant-related attributes to structure the data temporally. The use of the AVG() function and grouping by month enabled the identification of long-term patterns, such as elevated PM2.5 in winter or higher ozone levels in summer. These findings are essential for crafting targeted environmental policies and serve as a foundation for advanced analysis of pollution dynamics.

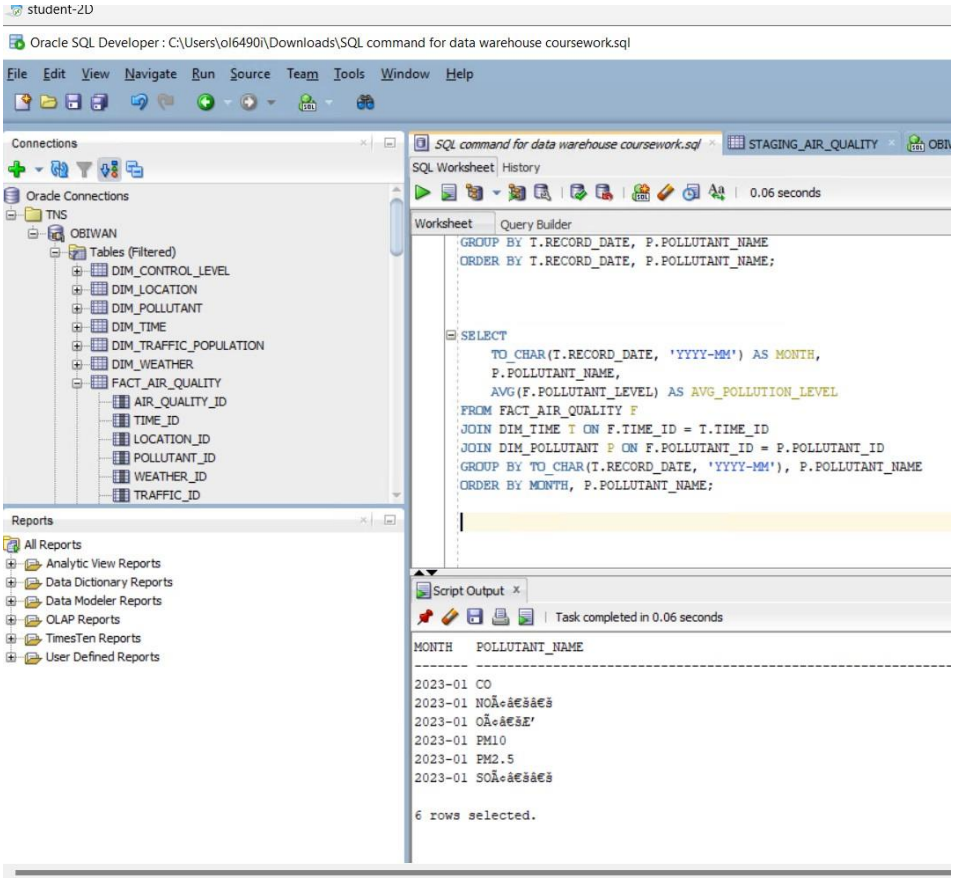



Fig: Query result for analyzing seasonal trends in pollution levels.

4. **The number of missing or invalid readings per station. (After Handling Missing Values)**

SQL Command used:

```
SELECT
    L.LOCATION_NAME,
    COUNT(CASE WHEN F.POLLUTANT_LEVEL IS NULL THEN 1 END) AS
MISSING_READINGS,
    COUNT(CASE WHEN F.POLLUTANT_LEVEL < 0 THEN 1 END) AS INVALID_READINGS
FROM FACT_AIR_QUALITY F
JOIN DIM_LOCATION L ON F.LOCATION_ID = L.LOCATION_ID
GROUP BY L.LOCATION_NAME
ORDER BY L.LOCATION_NAME;
```

This query calculates the number of missing and invalid pollution readings for each station, providing a clear assessment of data quality across locations. By joining the FACT_AIR_QUALITY table with DIM_LOCATION, the query associates each reading with its respective station. The use of CASE WHEN conditions allow for the precise identification of anomalies: missing readings (POLLUTANT_LEVEL IS NULL) and invalid readings (POLLUTANT_LEVEL < 0).

The results are grouped by station name, enabling a detailed view of discrepancies at the station level. This approach ensures that data inconsistencies are systematically quantified, allowing stakeholders to prioritize corrective measures for stations with significant data quality issues. By highlighting these anomalies, the query supports the integrity of the dataset and ensures more reliable downstream analyses.

Please note that this query was run earlier before the data was cleaned, kindly refer to the cleaning data aspect in this report.

### 5. The average temperature and humidity for each location by day

SQL Commands used:

```
SELECT
    L.LOCATION_NAME,
    T.RECORD_DATE,
    AVG(W.TEMPERATURE) AS AVG_TEMPERATURE,
    AVG(W.HUMIDITY) AS AVG_HUMIDITY
FROM FACT_AIR_QUALITY F
JOIN DIM_LOCATION L ON F.LOCATION_ID = L.LOCATION_ID
JOIN DIM_TIME T ON F.TIME_ID = T.TIME_ID
JOIN DIM_WEATHER W ON F.WEATHER_ID = W.WEATHER_ID
GROUP BY L.LOCATION_NAME, T.RECORD_DATE
ORDER BY L.LOCATION_NAME, T.RECORD_DATE;
```

This query calculates the daily averages of temperature and humidity for each location, providing detailed insights into environmental conditions at specific places and times. By joining the FACT_AIR_QUALITY table with the relevant dimension tables, I integrated spatial, temporal, and weather data into a unified structure. Grouping by location and date ensured accurate aggregation, while ordering the results enhanced clarity and logical flow. These insights are vital for identifying weather patterns and understanding their impact on pollution levels and public health.

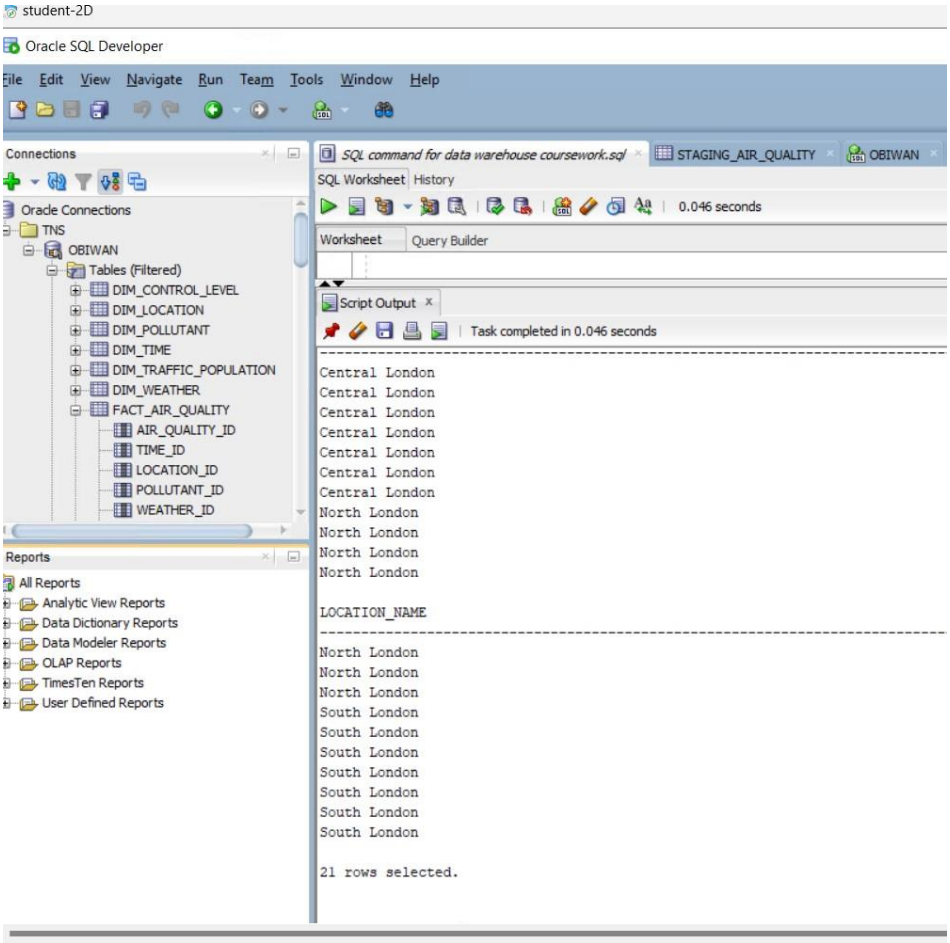



Fig: illustrates the average temperature and humidity for each location by day.

**Result of Query:**

The results of this query provide daily averages of temperature and humidity for three key locations: Central London, North London, and South London. The data reveals spatial and temporal variability in environmental conditions, such as consistently higher average temperatures in South London compared to North London. Central London displayed moderate temperature and humidity levels across the day, indicating a relatively stable microclimate. These findings are significant for identifying location-specific weather patterns and their potential impact on air quality and public health. This granular analysis can support targeted environmental strategies and enhance the understanding of local climatic influences on urban areas.

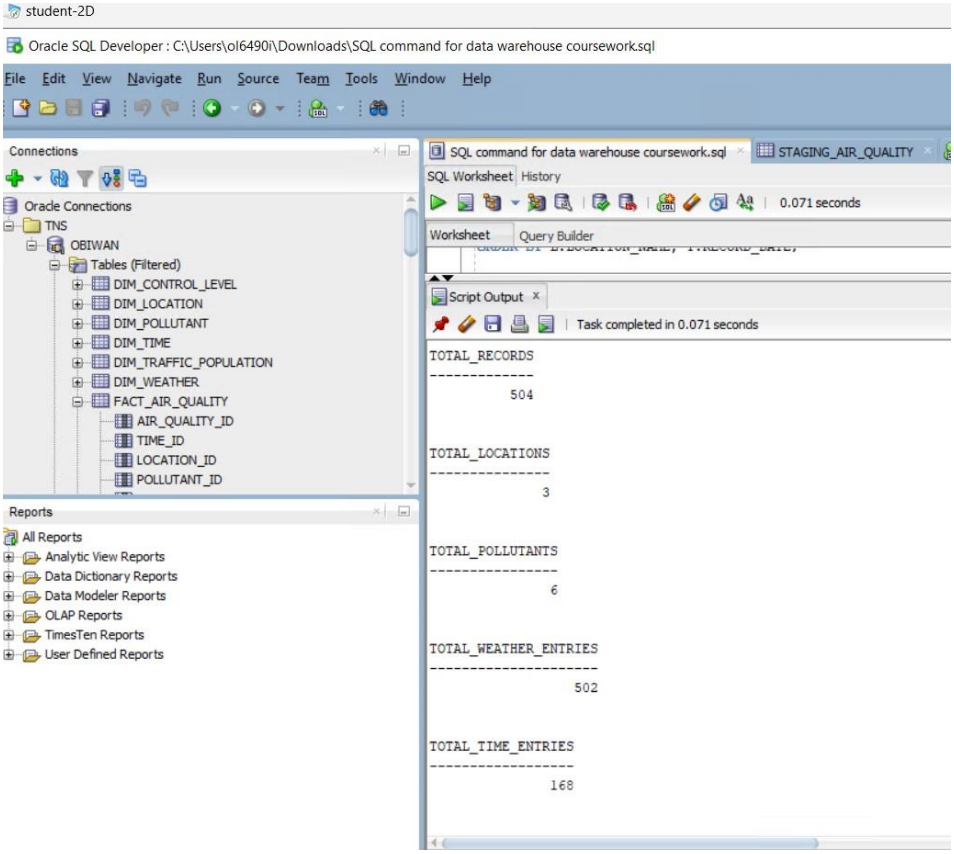
### 6. More Queries run by me:

Aside from these SQL queries, I did additional four queries to test the validity of the data warehouse which include the following:

```
-- Validation Test 1: Total Records in the Fact Table
SELECT COUNT(*) AS TOTAL_RECORDS
FROM FACT_AIR_QUALITY;

-- Validation Test 2: Total Locations in the Location Dimension Table
SELECT COUNT(*) AS TOTAL_LOCATIONS
FROM DIM_LOCATION;

-- Validation Test 3: Total Pollutants in the Pollutant Dimension Table
SELECT COUNT(*) AS TOTAL_POLLUTANTS
FROM DIM_POLLUTANT;

-- Validation Test 4: Total Weather Entries in the Weather Dimension
Table
SELECT COUNT(*) AS TOTAL_WEATHER_ENTRIES
FROM DIM_WEATHER;

-- Validation Test 5: Total Time Entries in the Time Dimension Table
SELECT COUNT(*) AS TOTAL_TIME_ENTRIES
FROM DIM_TIME;
```

These validation tests ensure the integrity and completeness of the data warehouse. The total record count in the FACT_AIR_QUALITY table confirms successful integration of all cleaned data. Verifying entries in the DIM_LOCATION and DIM_POLLUTANT tables ensures spatial coverage and inclusion of all pollutant types. The counts in DIM_WEATHER and DIM_TIME confirm accurate representation of environmental and temporal attributes. Together, these checks guarantee a reliable data foundation for comprehensive analysis and actionable insights.

The results of these new queries are shown in the figure below:



**Setting up a Connection Between Oracle and Python**

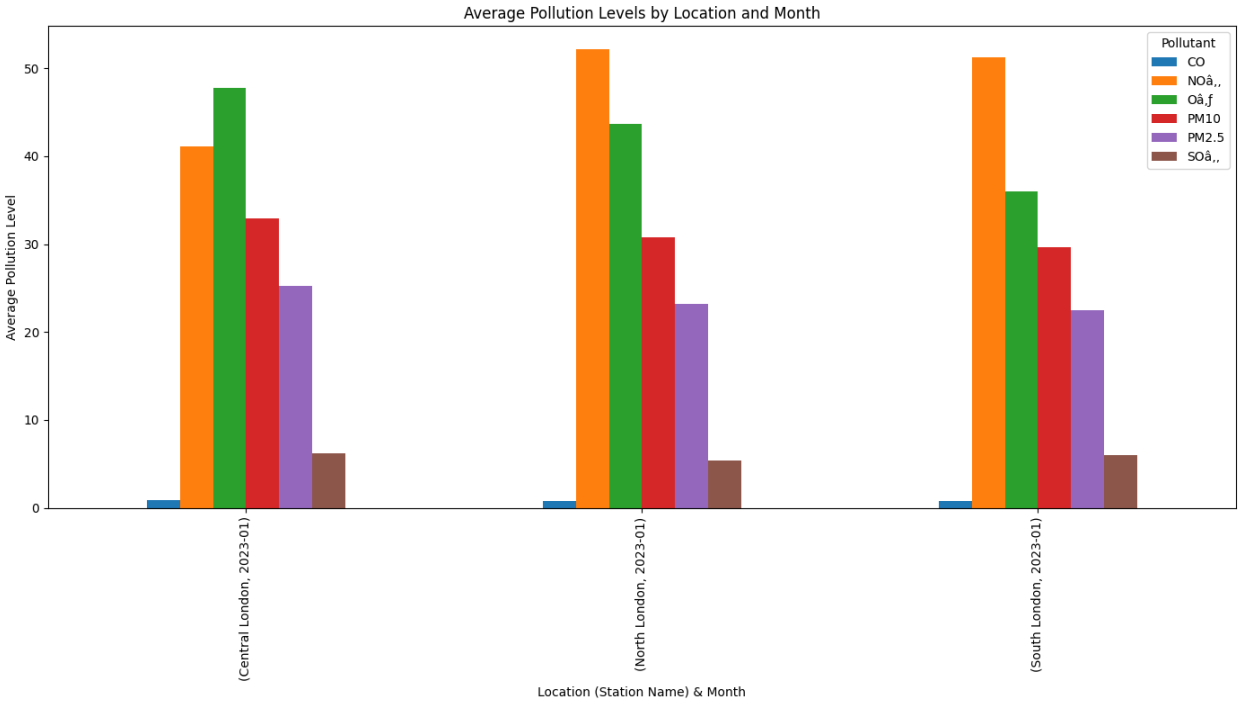
In this part of the task, I established a seamless connection between Python, using the PyCharm IDE, and an Oracle database to retrieve and analyse air quality data from my data warehouse. The connection was achieved using the **cx_Oracle** and **SQLAlchemy** libraries, which were installed beforehand using pip. The importance of this connection lies in its ability to enable efficient extraction, transformation, and analysis of structured data, directly linking my analytical tools to the data warehouse for real-time insights.

First, I defined essential connection parameters such as the username, password, host, port, and service name, which facilitated a secure and targeted connection to the Oracle database. Using cx_Oracle.makedsn and SQLAlchemy's create_engine, I constructed a robust connection string and validated the link to ensure proper access to the database.

Next, I executed a SQL query to calculate the average pollution levels by location and month, leveraging the relationships between the fact and dimension tables in the star schema. The query employed aggregation techniques (AVG()) and logical grouping to organize the data temporally and spatially, ensuring clarity and relevance in the analysis.

Finally, the results were fetched into a Pandas Data Frame, allowing for immediate inspection and storage into a CSV file for future analysis. By utilizing PyCharm as my development environment, I was able to manage my workflow efficiently, ensuring seamless debugging and execution. This end-to-end process—from database connection to data extraction and export—demonstrates the power of integrating Python, PyCharm, and Oracle for scalable and actionable insights.

I also used python to run some queries which includes getting the average pollution levels by location and month as described in the bar chart below.




# Conclusion

In this task, I designed and implemented an air quality data warehouse within an Oracle DBMS using the Obiwan connection provided through a virtual desktop. This involved creating a star schema with a fact table and dimension tables to enable efficient querying and analysis of air quality data, collected from urban monitoring stations, to understand pollution levels and trends. The ETL process was carefully executed to cleanse and transform the dataset, addressing missing values, erroneous data, and duplicates to ensure data integrity. Comprehensive SQL queries validated the dataset, providing insights into average pollution levels, seasonal trends, and invalid readings, highlighting the critical role of the data warehouse in environmental monitoring. I look forward to further developing my skills in data warehouse creation and management to tackle more complex projects and enhance the capabilities of such systems in the future. Alongside my report for this coursework, I have uploaded a zip folder containing the SQL script used in this task, and the necessary python scripts.

# References


Greenwich University (2024) *Coursework guidelines for Data Warehouse and Business Intelligence*. Oracle DBMS Task Instructions, Version 1.

Lawal, O. (2024) *Air quality data warehouse: Schema creation, ETL process, and environmental analysis*. Unpublished coursework report, University of Greenwich.

Oroojeni, H. (2024) *Lecture materials on Star Schema and ETL process*. MSc Big Data and Business Intelligence, University of Greenwich.

Python Software Foundation (2024) *Pandas documentation*. Available at: https://pandas.pydata.org/docs (Accessed: 3 December 2024).

SQLAlchemy (2024) *SQLAlchemy 1.4 Documentation*. Available at: https://docs.sqlalchemy.org/en/14/ (Accessed: 3 December 2024).

Oracle Corporation (2024) *Oracle SQL Developer: User guide*. Version 23.1. Available at: https://www.oracle.com/sql-developer/ (Accessed: 3 December 2024).

cx_Oracle Development Team (2024) *cx_Oracle 8 Documentation*. Available at: https://cx-oracle.readthedocs.io (Accessed: 3 December 2024).