CS 623 - Database Management Systems

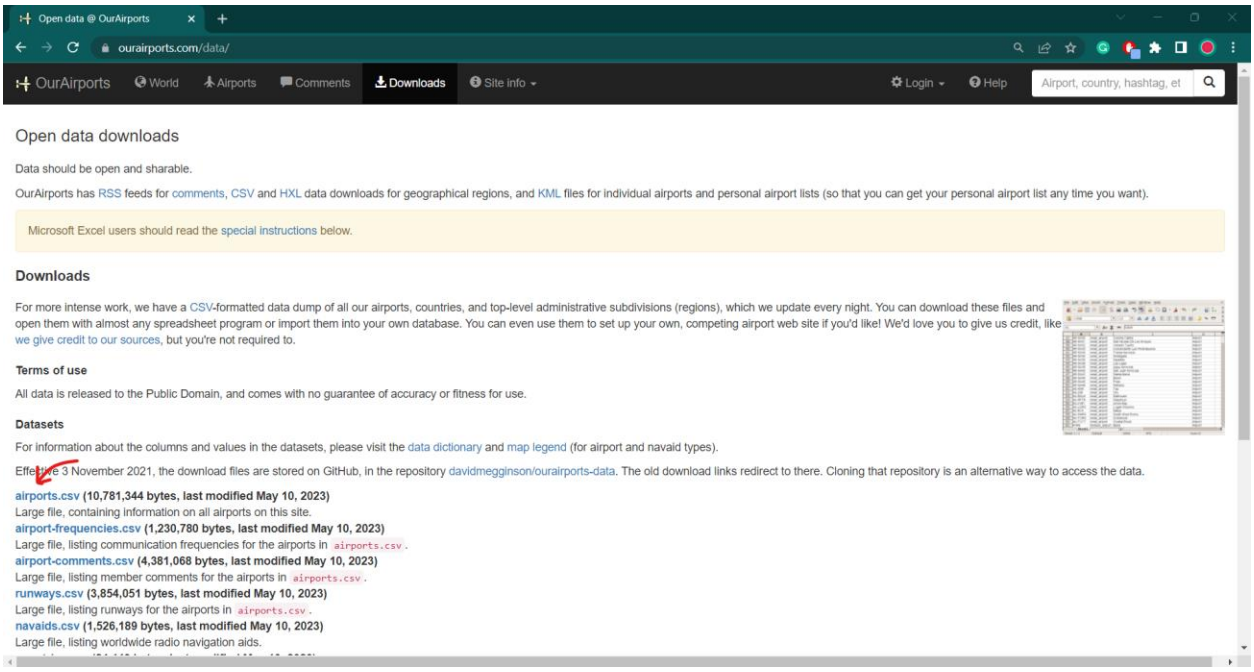
Group Project

Team Members:

Pruthvi Raj Pudi
Vamshi Krishna Bangaru
Adusumalli Lokesh

Our Dataset Source:

We're using data from OurAirports (https://ourairports.com/data/), a platform where users share and download geospatial data related to airports worldwide. The file, airports.csv, contains comprehensive geospatial data for all the airports listed on the platform, updated as of May 9, 2023. It's in a CSV format, which is easily imported into various software for further spatial analysis.



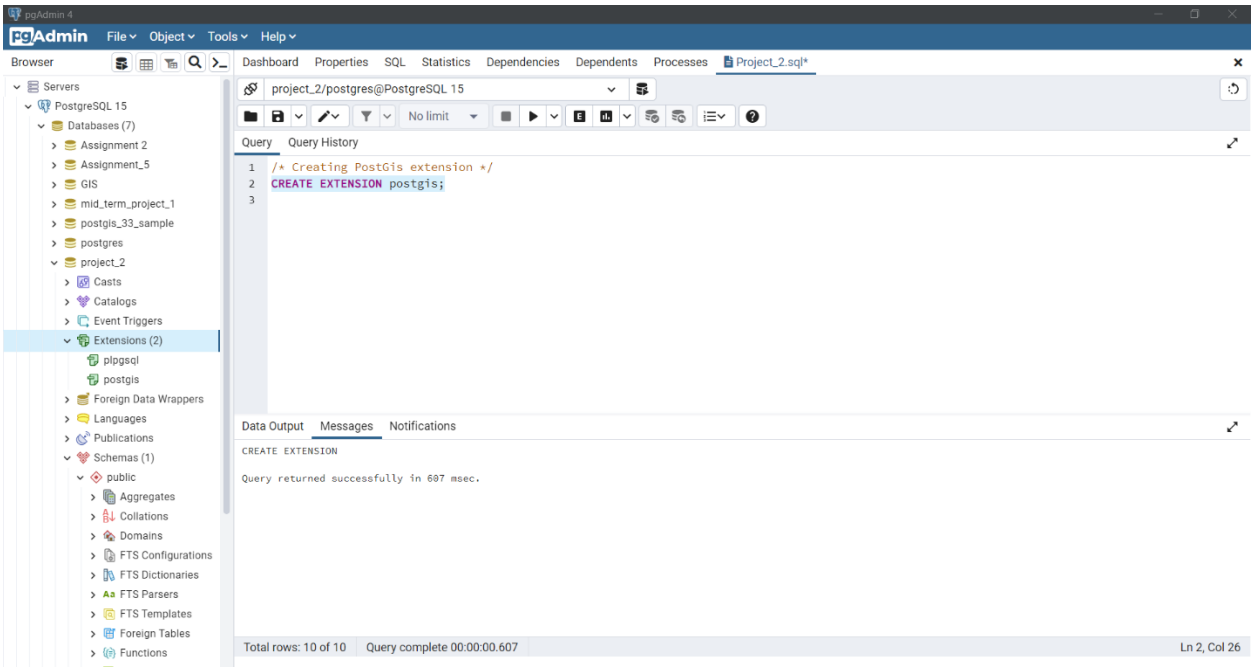
CS 623 - Database Management Systems

Group Project

Creating the PostGIS Extension:

```
CREATE EXTENSION postgis;
```

This command is used to add the PostGIS extension to your PostgreSQL database. If it's not already installed, this command will install it.



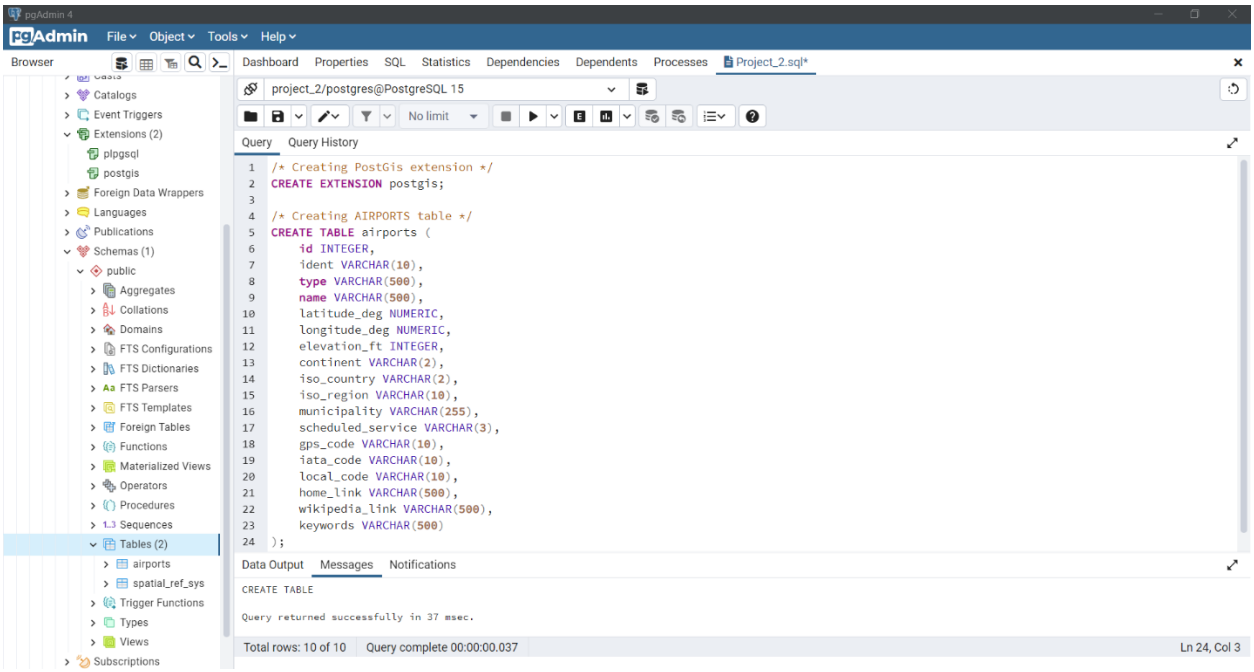
PostGIS is an extension of PostgreSQL and is the common standard for vector geodata in open source databases. It adds support for geographic objects, allowing the database to perform spatial queries and analyses.

CS 623 - Database Management Systems

Group Project

Creating the Airports Table:

```
CREATE TABLE airports (
        id INTEGER,
        ident VARCHAR(10),
        type VARCHAR(500),
        name VARCHAR(500),
        latitude_deg NUMERIC,
        longitude_deg NUMERIC,
        elevation_ft INTEGER,
        continent VARCHAR(2),
        iso_country VARCHAR(2),
        iso_region VARCHAR(10),
        municipality VARCHAR(255),
        scheduled_service VARCHAR(3),
        gps_code VARCHAR(10),
        iata_code VARCHAR(10),
        local_code VARCHAR(10),
        home_link VARCHAR(500),
        wikipedia_link VARCHAR(500),
        keywords VARCHAR(500));
```

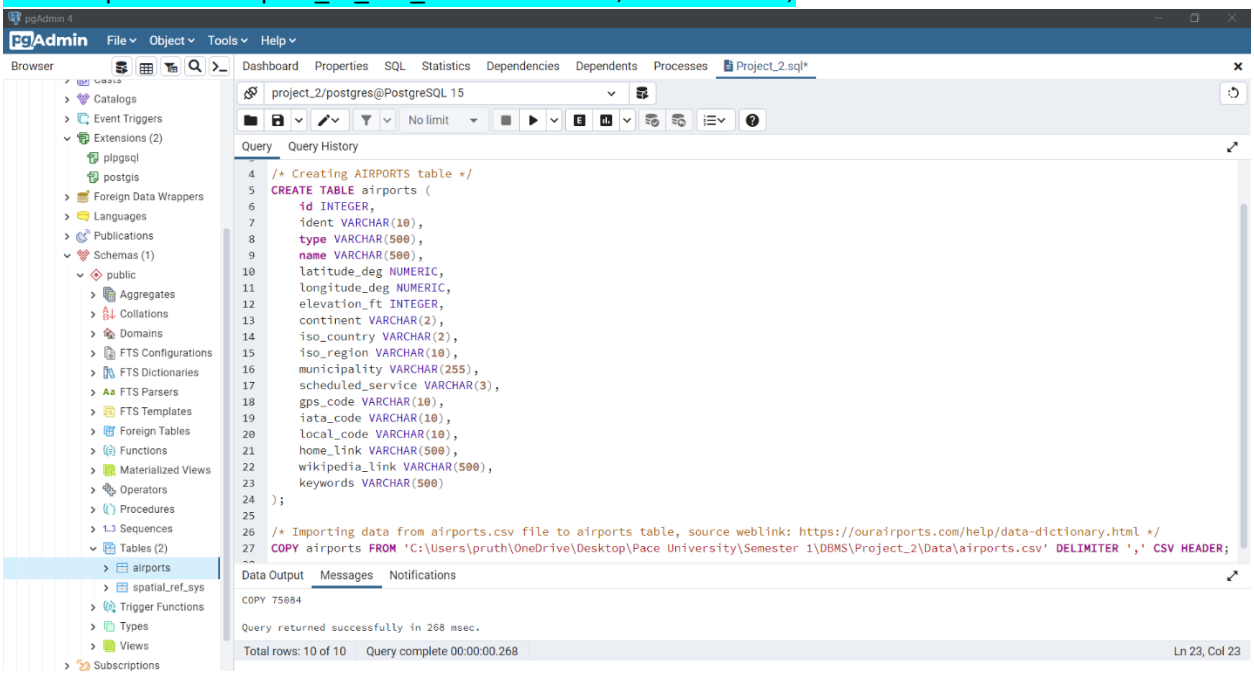


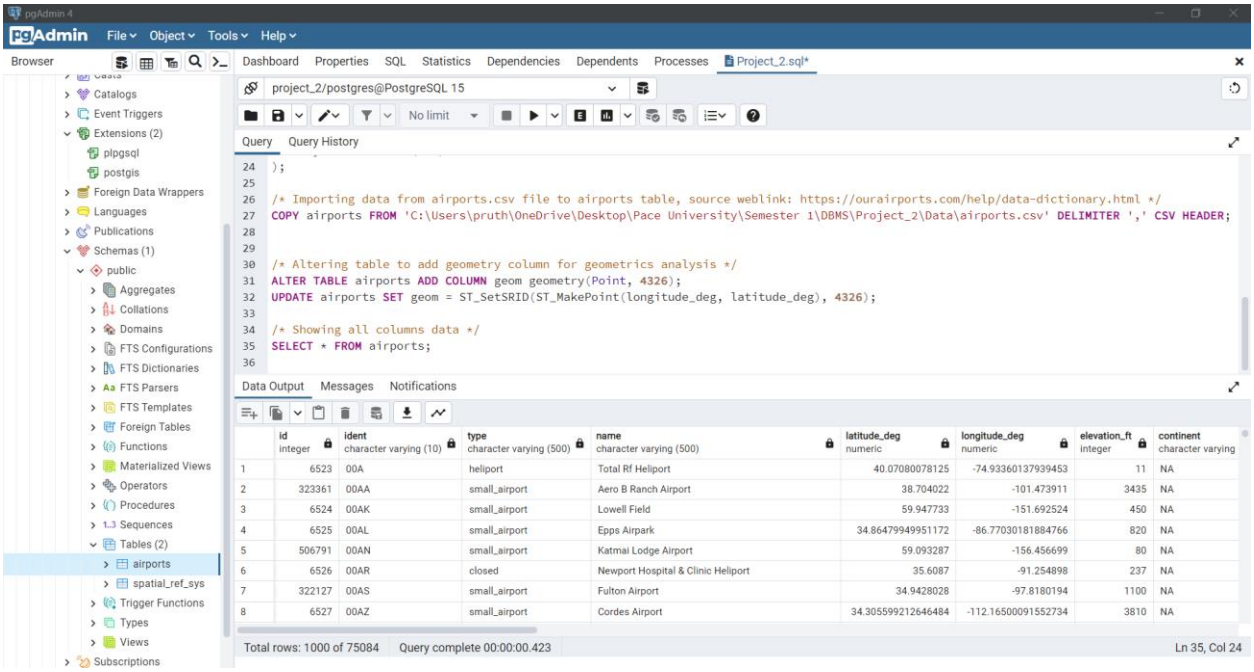
The CREATE TABLE statement defines a structure to hold our data. Each attribute of the airport entity (like id, name, type, location, etc.) is assigned a corresponding column in the airports table. Different data types, like INTEGER, VARCHAR, and NUMERIC, are used based on the nature of the data each

column will hold. VARCHAR is used for text, INTEGER for whole numbers, and NUMERIC for decimal or floating-point numbers.

Importing data from a CSV file:

COPY airports FROM 'path_to_our_file' DELIMITER ',' CSV HEADER;





CS 623 - Database Management Systems

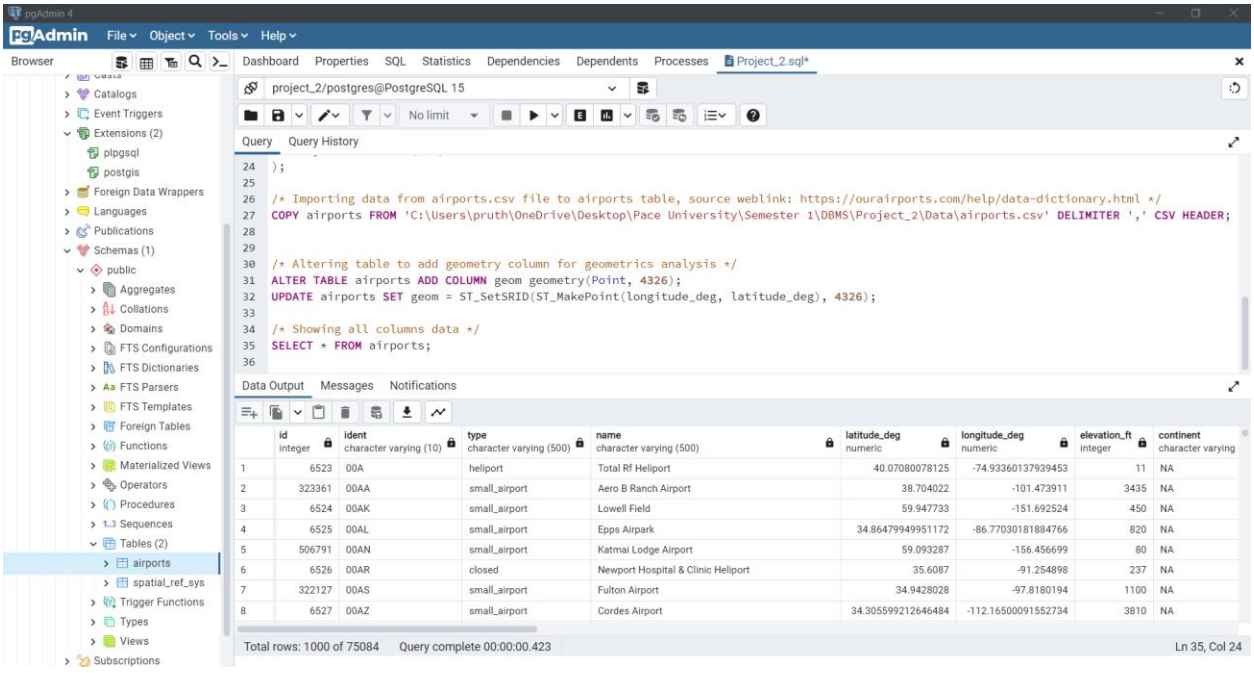
Group Project

The COPY command is a powerful tool in PostgreSQL, allowing for bulk import/export of data to and from a table. It's more efficient than using multiple INSERT commands when dealing with a large amount of data. The DELIMITER ',' CSV HEADER part of the command specifies that the data in the file is comma-separated and that the file includes a header row.

Adding a Geometry Column and Updating it:

```
ALTER TABLE airports ADD COLUMN geom geometry(Point, 4326);
UPDATE airports SET geom = ST_SetSRID(ST_MakePoint(longitude_deg, latitude_deg), 4326);
```

These commands are used to add a new column called "geom" to the airports table that will store the geographical coordinates of the airports. Then, the geographical points are created from the longitude and latitude columns and stored in the geom column.



	id integer	ident character varying (10)	type character varying (500)	name character varying (500)	latitude_deg numeric	longitude_deg numeric	elevation_ft integer	continent character varying
1	6523	00A	heliport	Total Rf Heliport	40.07080078125	-74.93360137939453	11	NA
2	323361	00AA	small_airport	Aero B Ranch Airport	38.704022	-101.473911	3435	NA
3	6524	00AK	small_airport	Lowell Field	59.947733	-151.692524	450	NA
4	6525	00AL	small_airport	Epps Airpark	34.86479949951172	-86.77030181884766	820	NA
5	506791	00AN	small_airport	Katmai Lodge Airport	59.093287	-156.456699	80	NA
6	6526	00AR	closed	Newport Hospital & Clinic Heliport	35.6087	-91.254898	237	NA
7	322127	00AS	small_airport	Fulton Airport	34.9428028	-97.8180194	1100	NA
8	6527	00AZ	small_airport	Cordes Airport	34.305599212646484	-112.16500091552734	3810	NA

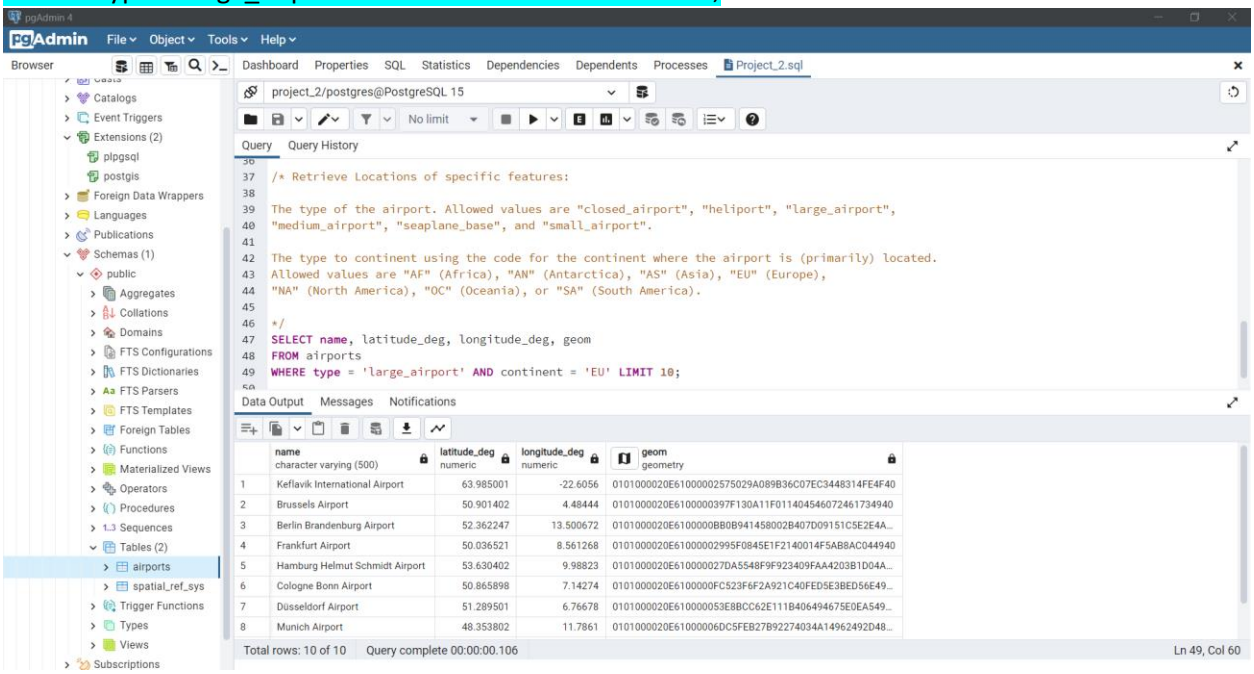
The addition of a geom column to the airports table is significant. This column is of the geometry data type, a spatial data type from the PostGIS extension. The geometry(Point, 4326) argument specifies that the column will hold point data in the EPSG:4326 coordinate system (WGS84). The UPDATE command uses PostGIS functions ST_SetSRID and ST_MakePoint to convert longitude and latitude degrees into a geometric point.

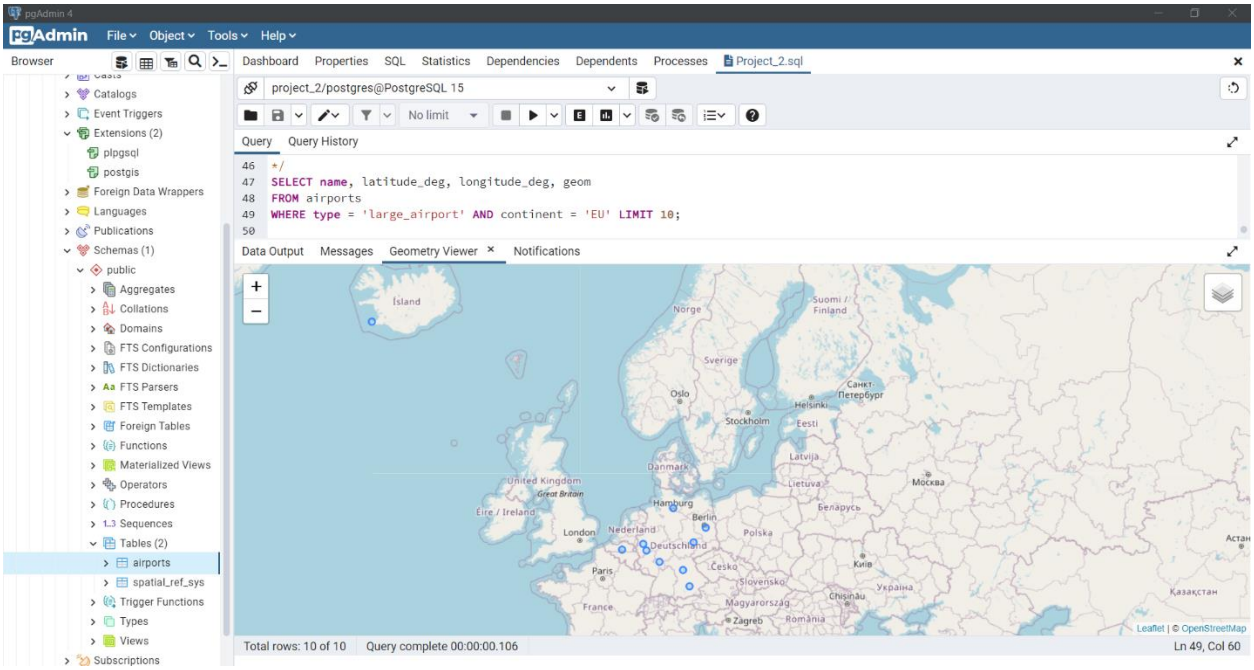
CS 623 - Database Management Systems

Group Project

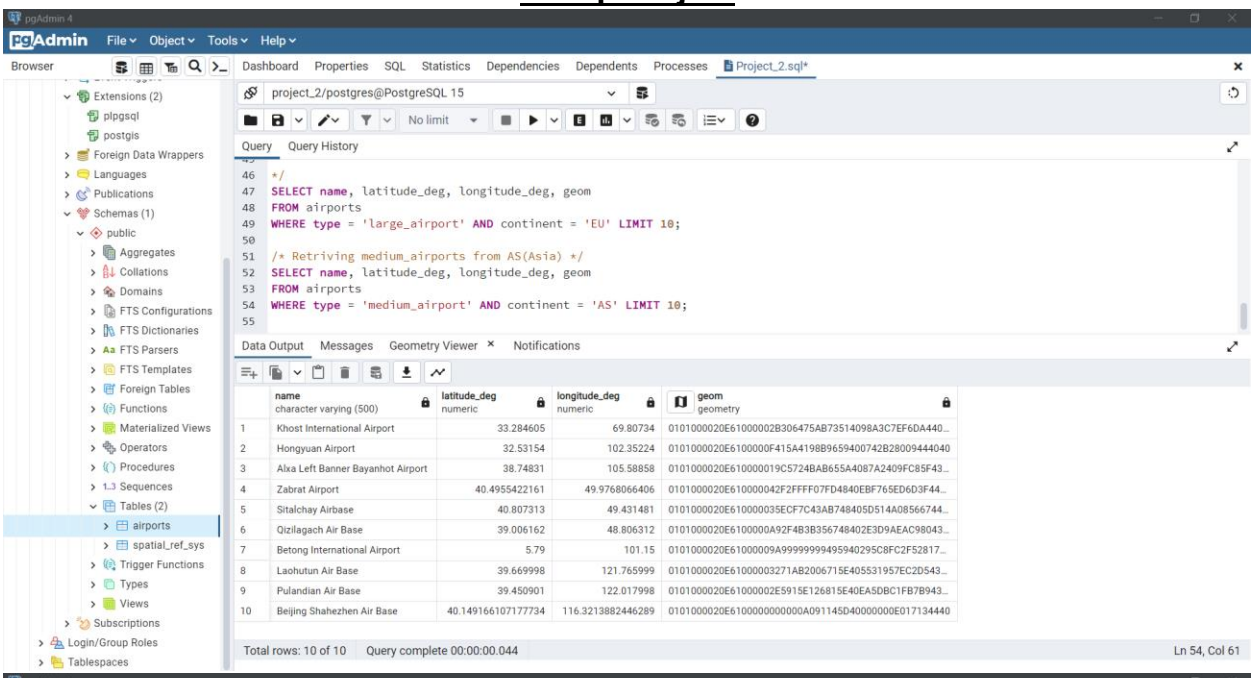
Retrieve Locations of specific features:

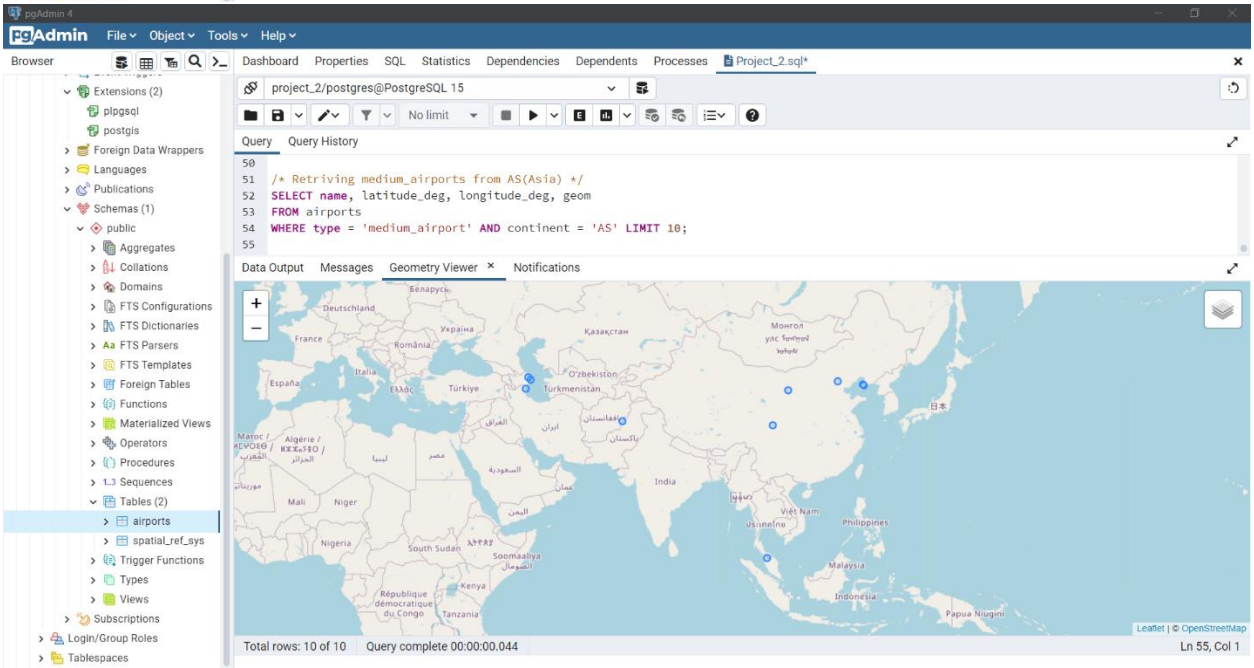
```
SELECT * FROM airports;
SELECT name, latitude_deg, longitude_deg, geom
FROM airports
WHERE type = 'large_airport' AND continent = 'EU' LIMIT 10;
```





Group Project





These commands are used to query data from the airports table. The first query selects all data from the table. The second query selects specific columns (name, latitude, longitude, geom) from the table for airports that are of type 'large_airport' and are located in Europe ('EU').

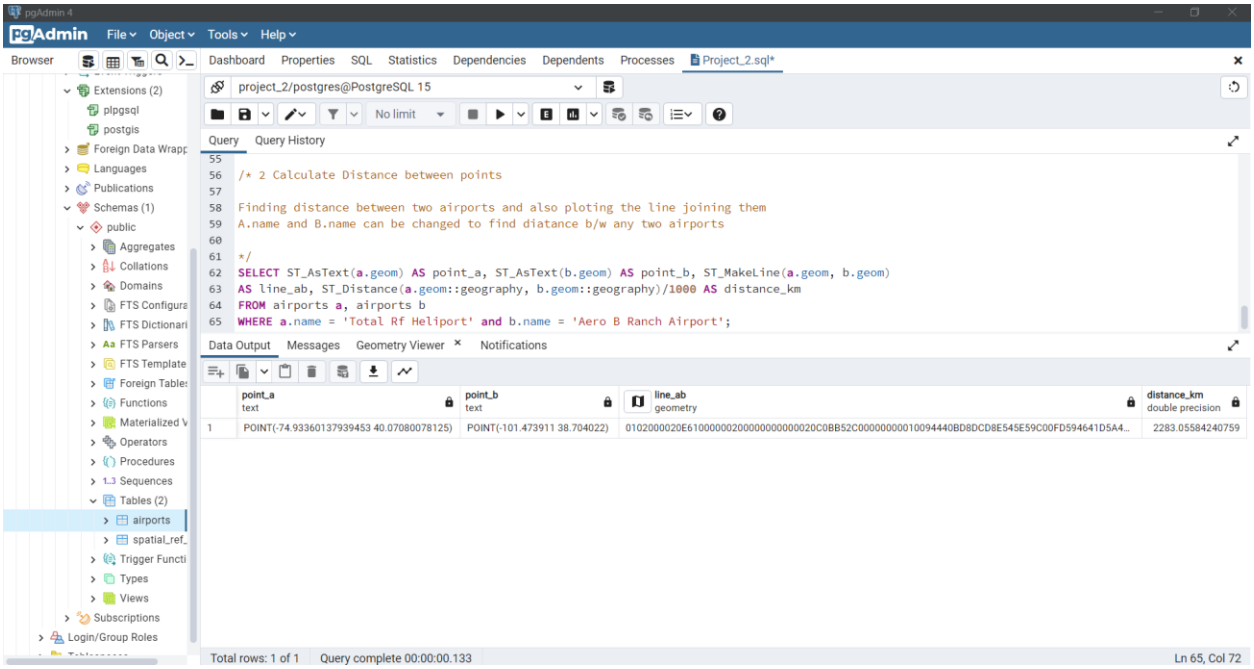
The SELECT command is used to retrieve data from the database. The WHERE clause filters records that satisfy a particular condition. The LIMIT clause restricts the output to a specified number of records. For instance, LIMIT 10 will only return the first 10 records that meet the conditions.

CS 623 - Database Management Systems

Group Project

Calculating Distance between Points:

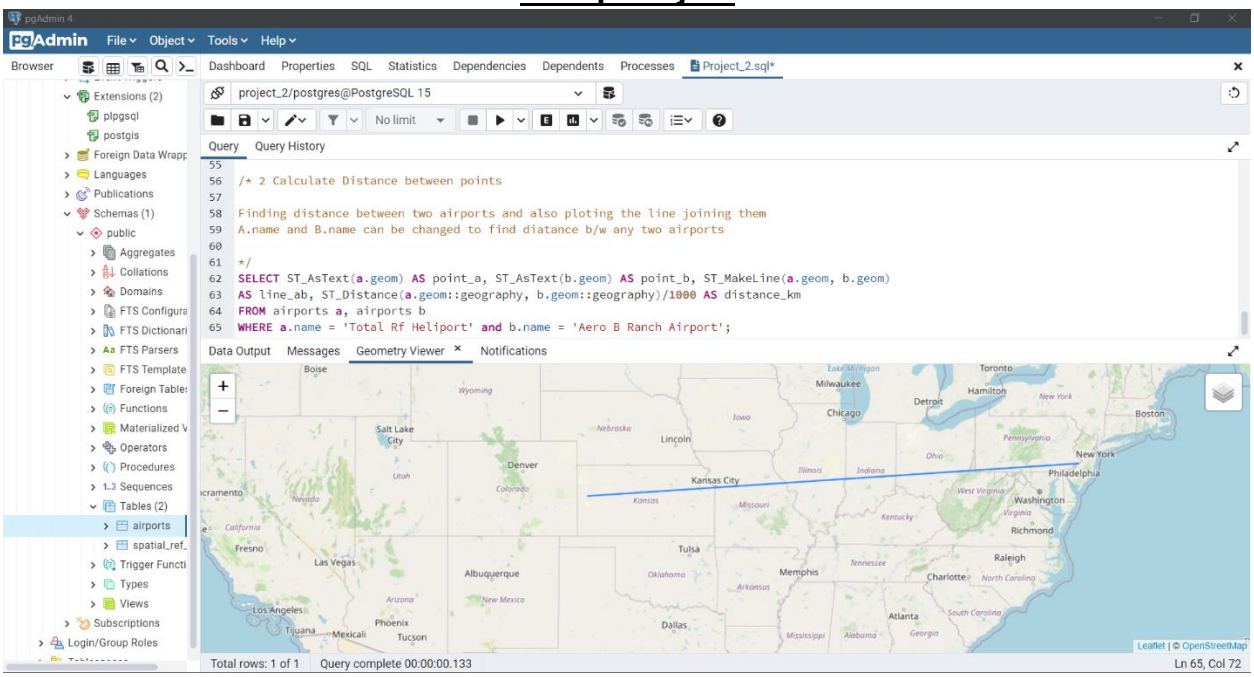
```
SELECT ST_AsText(a.geom) AS point_a, ST_AsText(b.geom) AS point_b, ST_MakeLine(a.geom, b.geom)
AS line_ab, ST_Distance(a.geom::geography, b.geom::geography)/1000 AS distance_km
FROM airports a, airports b
WHERE a.name = 'Total Rf Heliport' and b.name = 'Aero B Ranch Airport';
```



This query calculates the distance between two airports, represented by their geographical points. It also creates a line between these two points.

CS 623 - Database Management Systems

Group Project



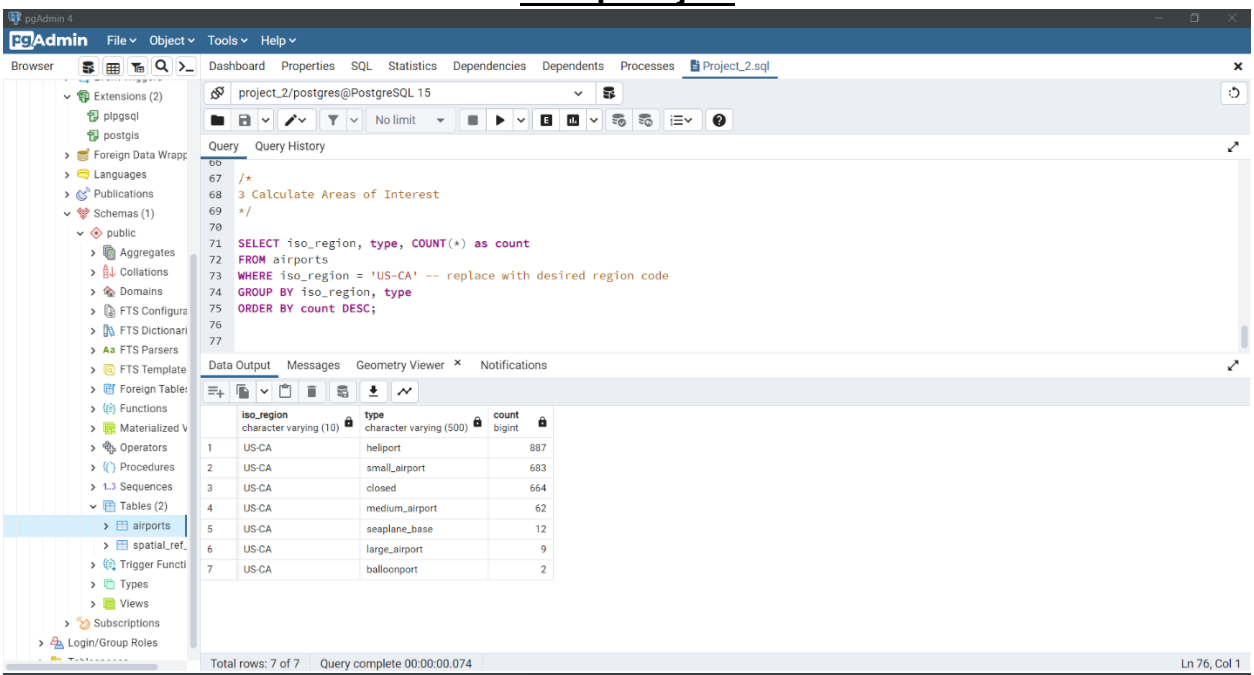
This part involves spatial functions from the PostGIS extension. ST_AsText converts geometric data into readable string format. ST_MakeLine creates a line that joins two points. ST_Distance calculates the distance between two points, and ::geography casts the geometry data type to geography, which is necessary for accurate distance calculations over long distances and global scales.

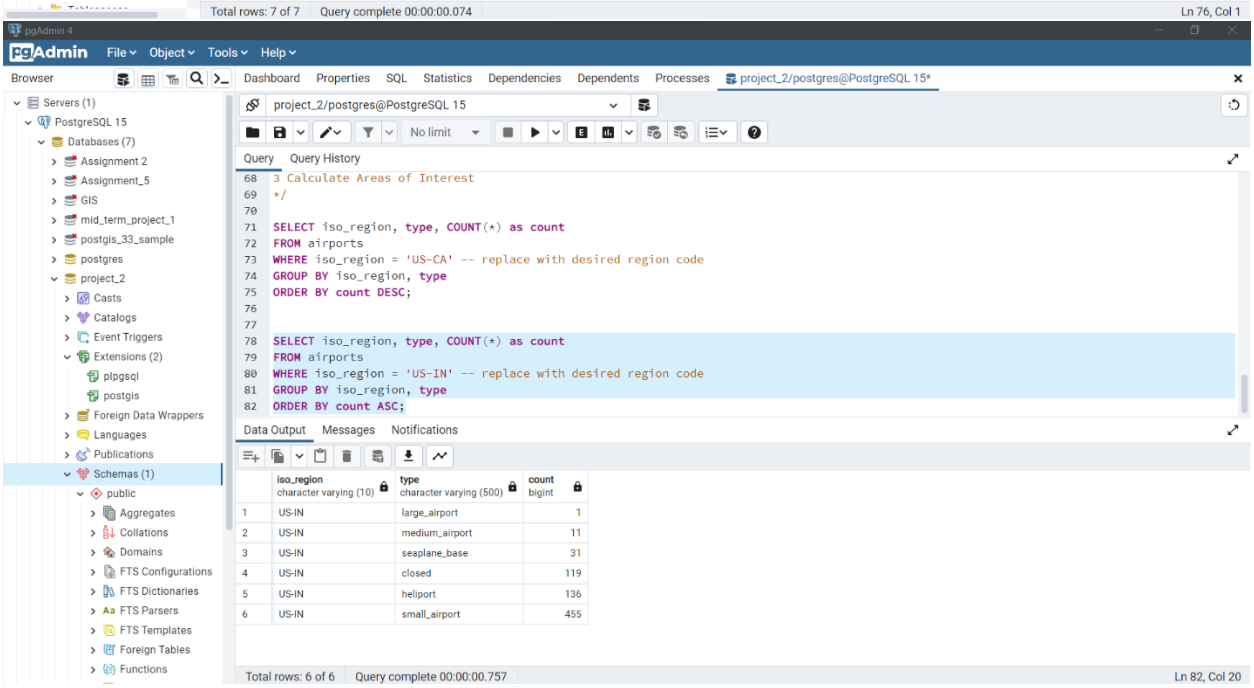
Calculating Areas of Interest:

```
SELECT iso_region, type, COUNT(*) as count
FROM airports
WHERE iso_region = 'US-CA'
GROUP BY iso_region, type
ORDER BY count DESC;
```

This command is used to count the number of each type of airport in a specific region ('US-CA' for California, USA). The results are grouped by region and type and ordered in descending order by the count.

Group Project





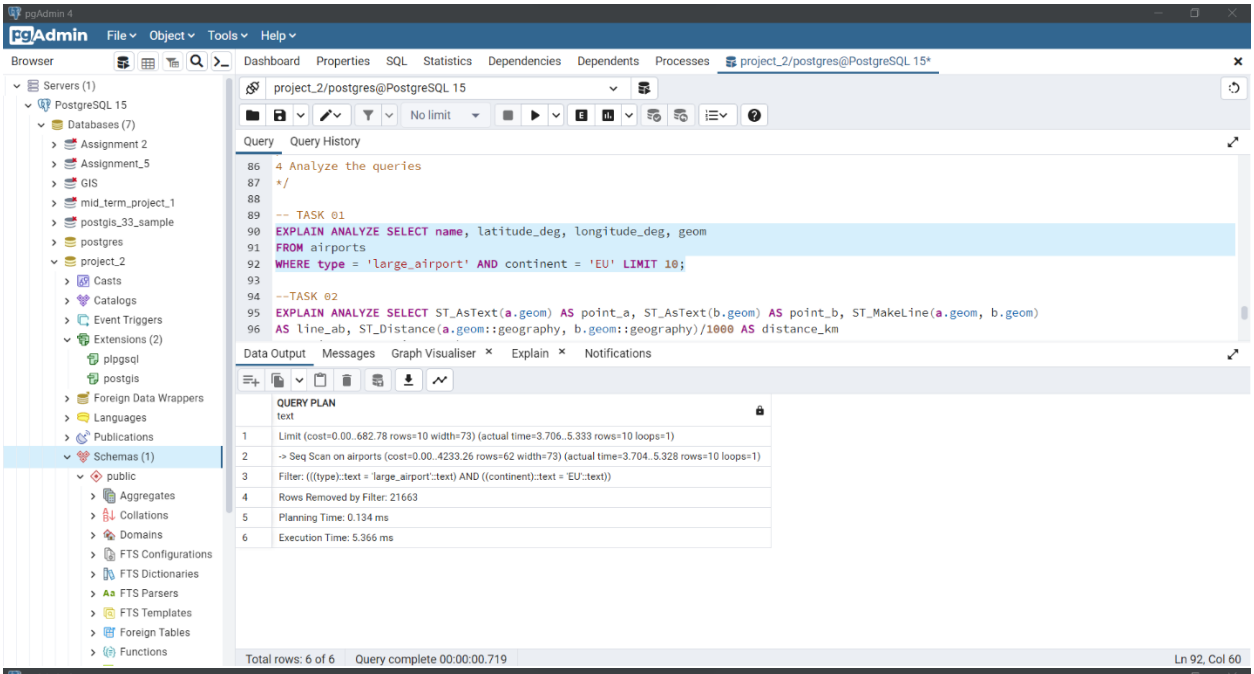
These queries use aggregate function COUNT(*) to count the number of airports in each type for a specific region. GROUP BY groups the result by the unique combinations of the specified columns, and ORDER BY sorts the result by the specified column.

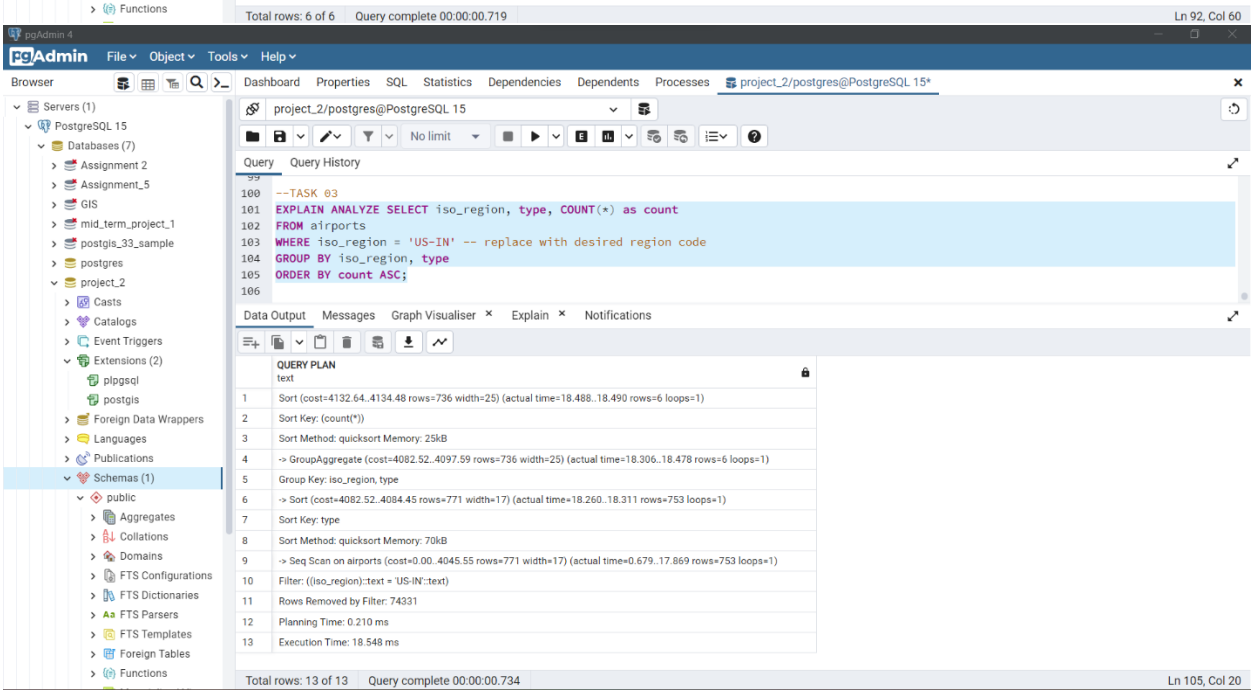
CS 623 - Database Management Systems

Group Project

Analyzing the Queries:

```
EXPLAIN ANALYZE SELECT name, latitude_deg, longitude_deg, geom
FROM airports
WHERE type = 'large_airport' AND continent = 'EU' LIMIT 10;
```





CS 623 - Database Management Systems

Group Project

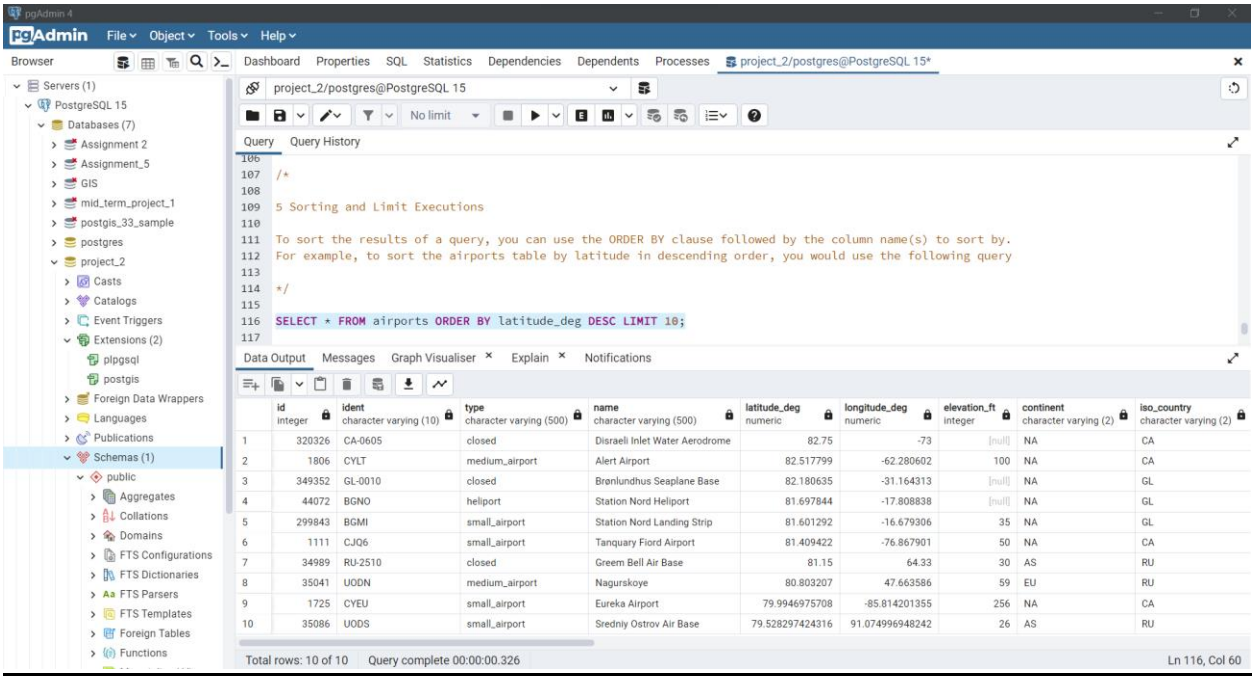
The EXPLAIN ANALYZE command is used to display the execution plan that the PostgreSQL planner generates for the supplied statement.

Sorting and Limit Executions:

- SELECT * FROM airports ORDER BY latitude_deg DESC LIMIT 10;
- SELECT * FROM airports ORDER BY elevation_ft ASC LIMIT 10;

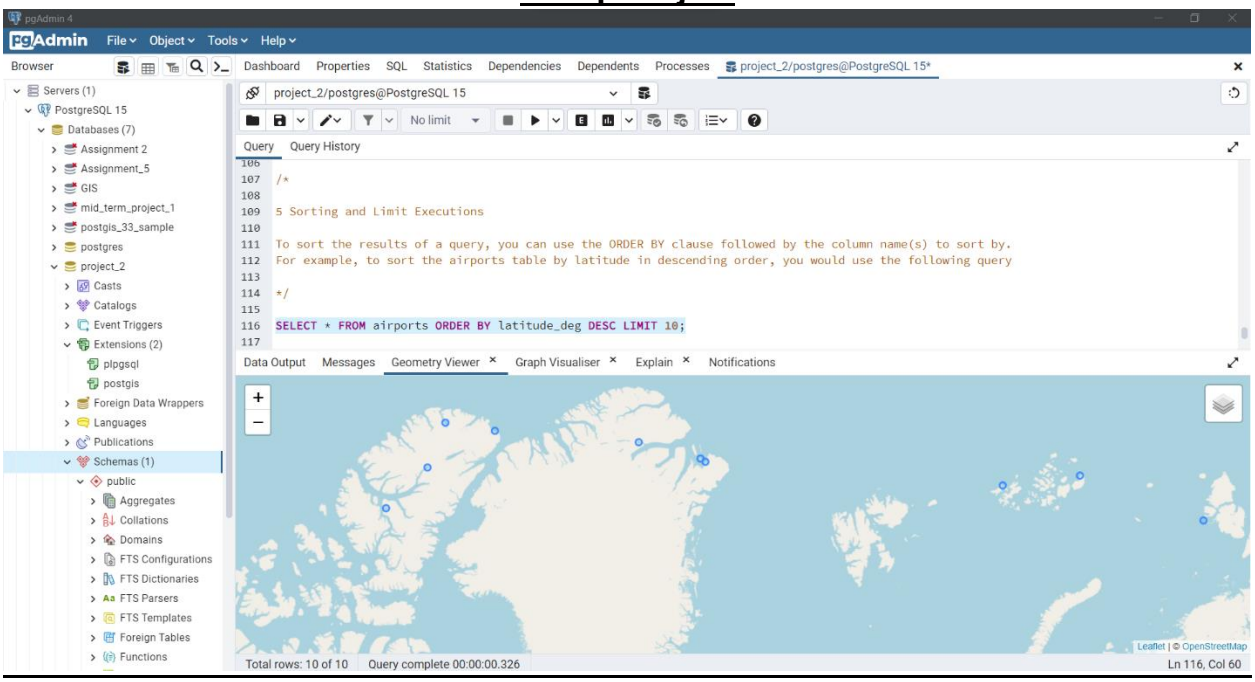
These two commands sort the data from the airports table. The first one sorts the data in descending order by the latitude of the airports, and the second one sorts the data in ascending order by the elevation of the airports. Both commands only return the top 10 results due to the LIMIT clause.

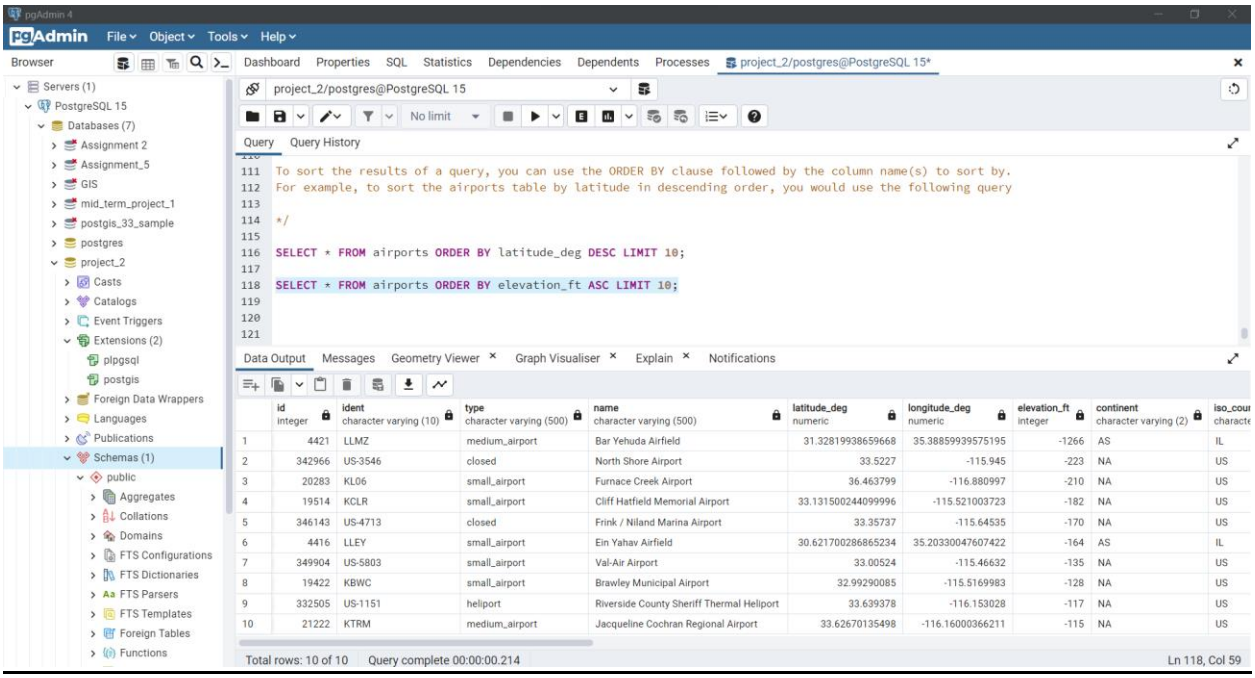
.



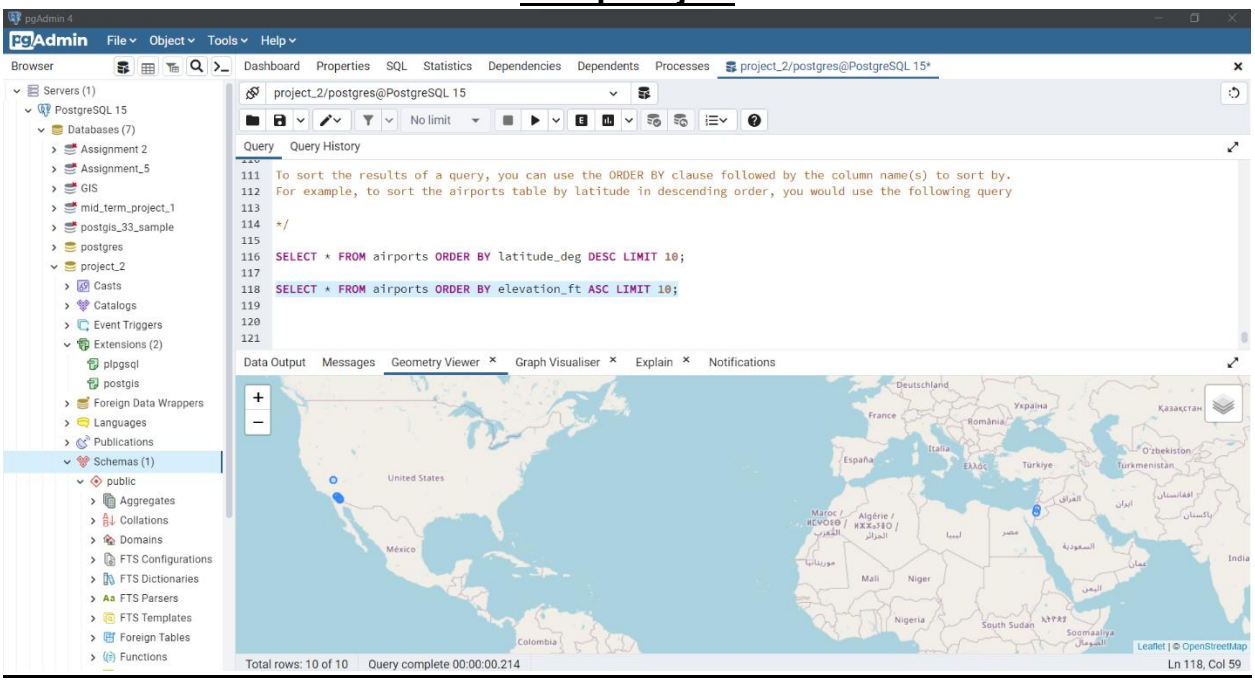
CS 623 - Database Management Systems

Group Project





Group Project



The ORDER BY clause sorts the records in ascending or descending order based on one or multiple columns. The LIMIT clause restricts the number of records returned, which can help reduce computational load when dealing with large datasets.

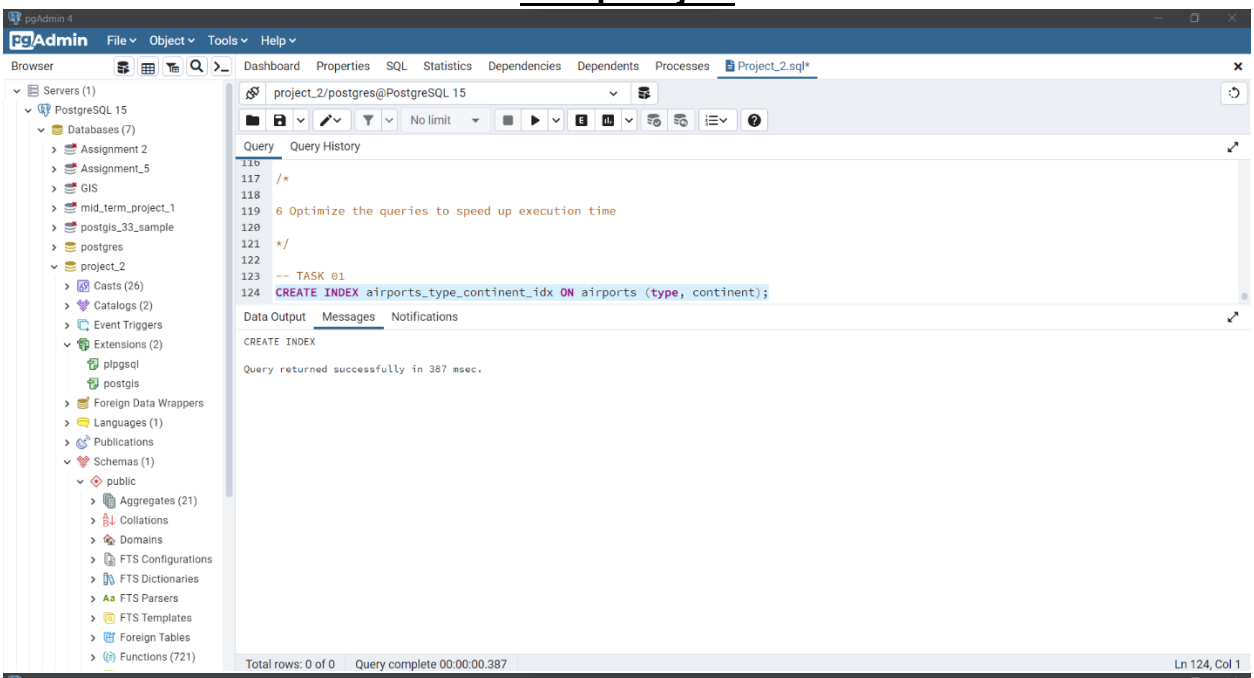
Optimizing the Queries to Speed Up Execution Time:

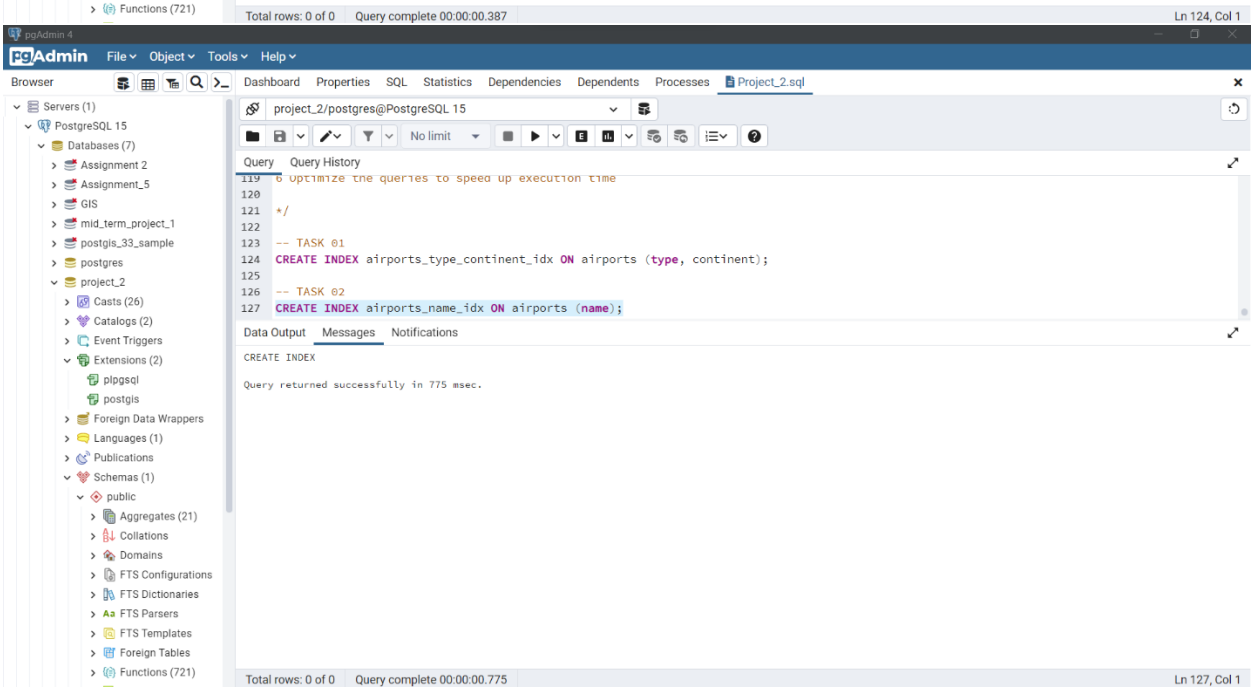
- CREATE INDEX airports_type_continent_idx ON airports (type, continent);
- CREATE INDEX airports_name_idx ON airports (name);
- CREATE INDEX airports_iso_region_idx ON airports (iso_region);
- CREATE INDEX airports_latitude_deg_idx ON airports (latitude_deg);

These commands are used to create indexes on the specified columns of the airports table. Indexes can significantly speed up data retrieval times on a database.

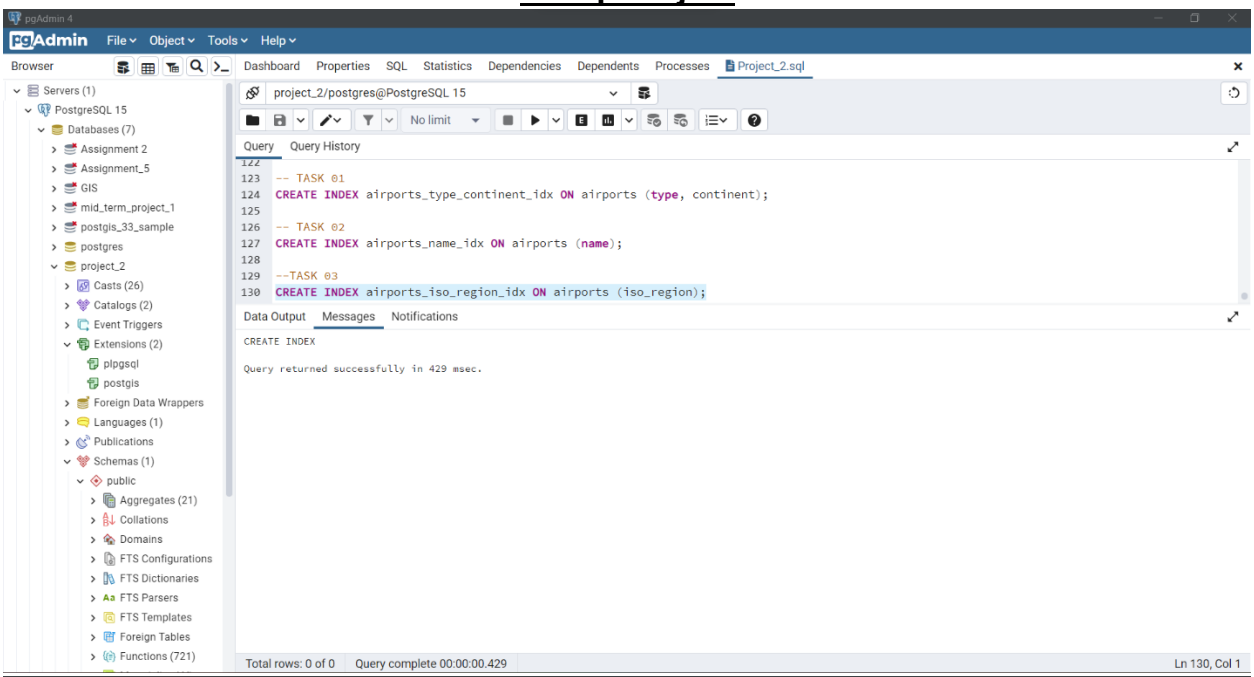
CS 623 - Database Management Systems

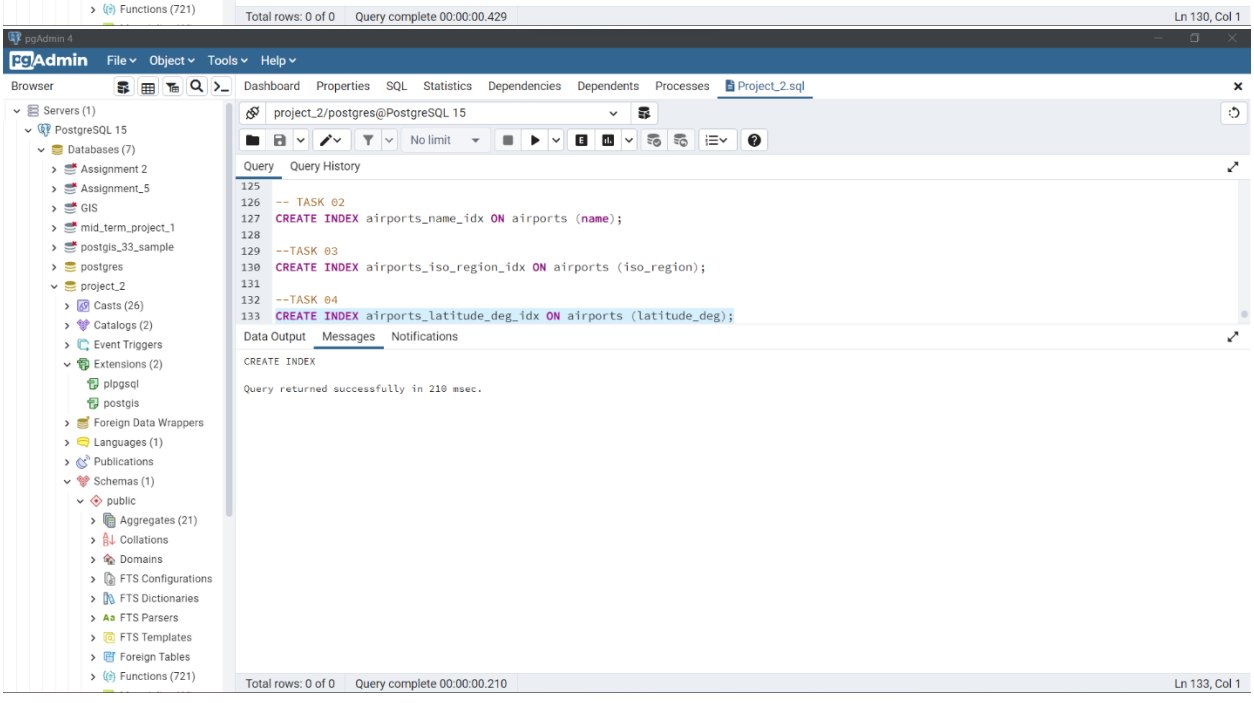
Group Project





Group Project





Indexing speeds up data retrieval by providing swift access to rows in the database tables. It's similar to the index of a book, which provides quick access to the content based on the index entries. In this script, CREATE INDEX is used to create indexes on certain columns, such as type, continent, name, iso_region, and latitude_deg in the airports table. An index can significantly improve query performance, especially for large tables. However, it's worth noting that while indexes speed up data retrieval, they can slow down data insertion, deletion, and updates because the index also needs to be updated. Therefore, indexes should be used judiciously.

CS 623 - Database Management Systems

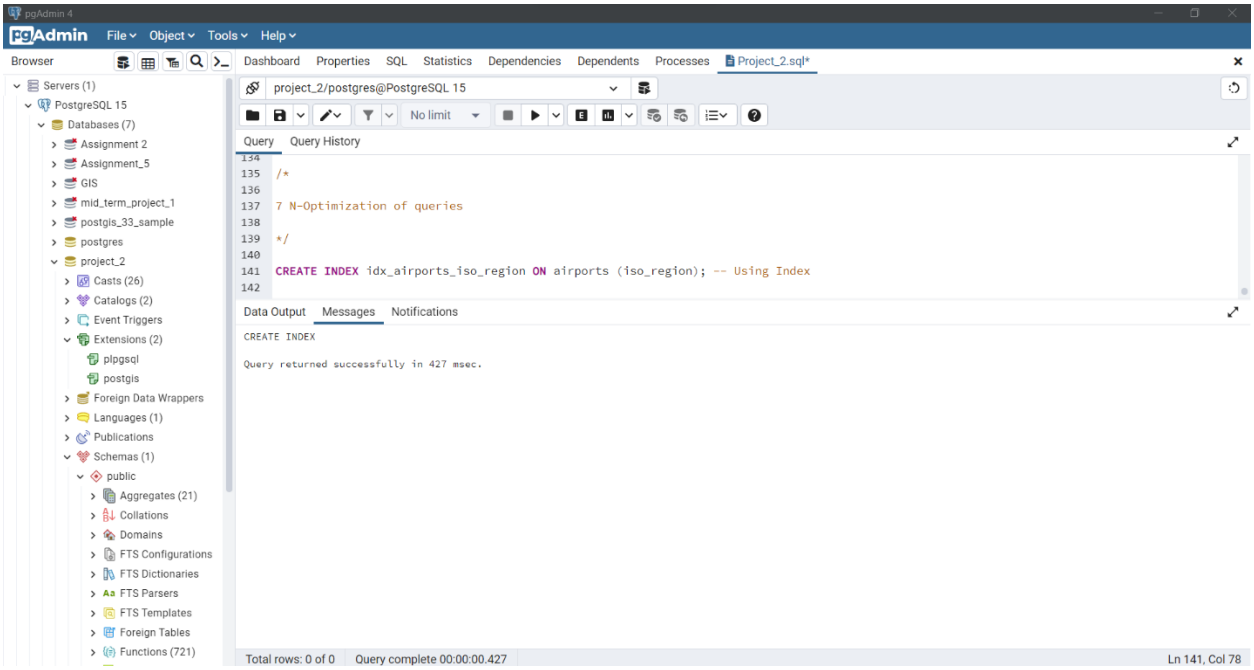
Group Project

N-Optimization of Queries:

N-Optimization of queries refers to a set of techniques and best practices used to optimize SQL queries in order to improve their performance and reduce their execution time. Some of the common techniques used for query optimization include:

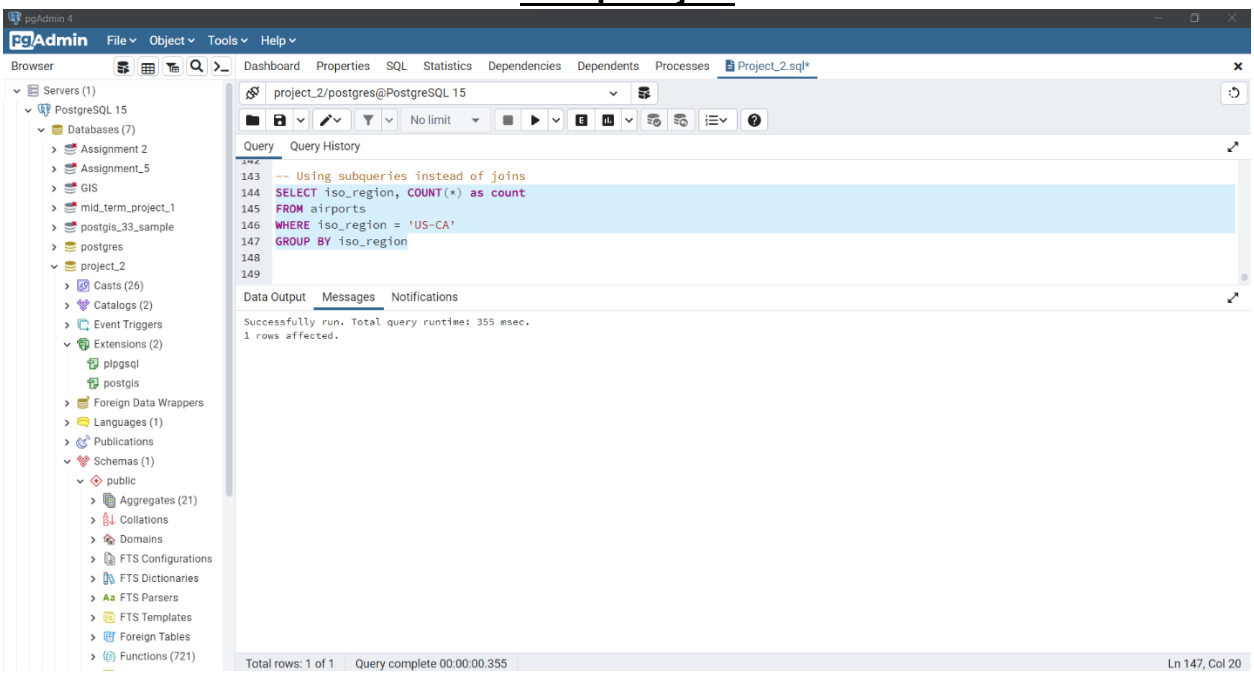
- **Creating an index on the iso_region column:** This will speed up queries that involve a search or sort operation on this column.

```
CREATE INDEX idx_airports_iso_region ON airports (iso_region);
```



- **Using subqueries instead of joins**: Depending on the scenario, using a subquery can sometimes be more efficient than a join. In this case, a subquery is used to count the number of airports in a specific region.

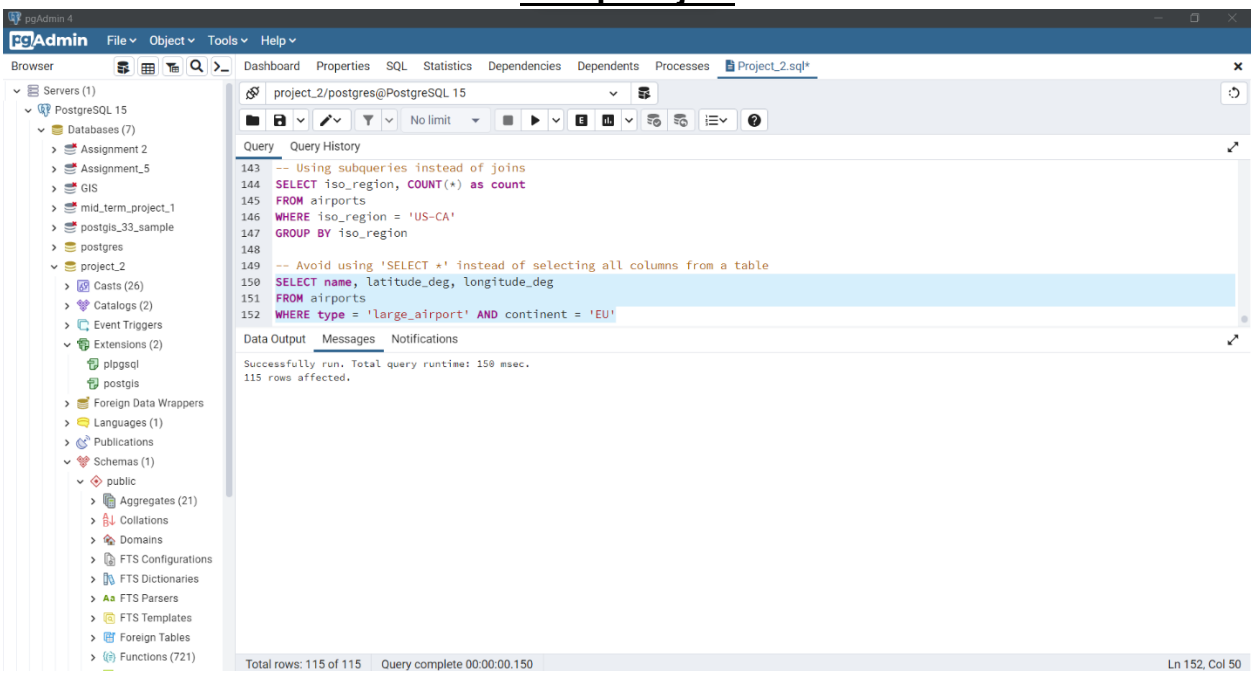
```
SELECT iso_region, COUNT(*) as count
FROM airports
WHERE iso_region = 'US-CA'
GROUP BY iso_region;
```



- **Avoiding 'SELECT *'**: The 'SELECT *' query can be quite inefficient, especially for large tables with many columns. By specifying the exact columns you need in your SELECT clause, you can reduce the amount of data that needs to be read from the disk and transferred from the database server to your application.

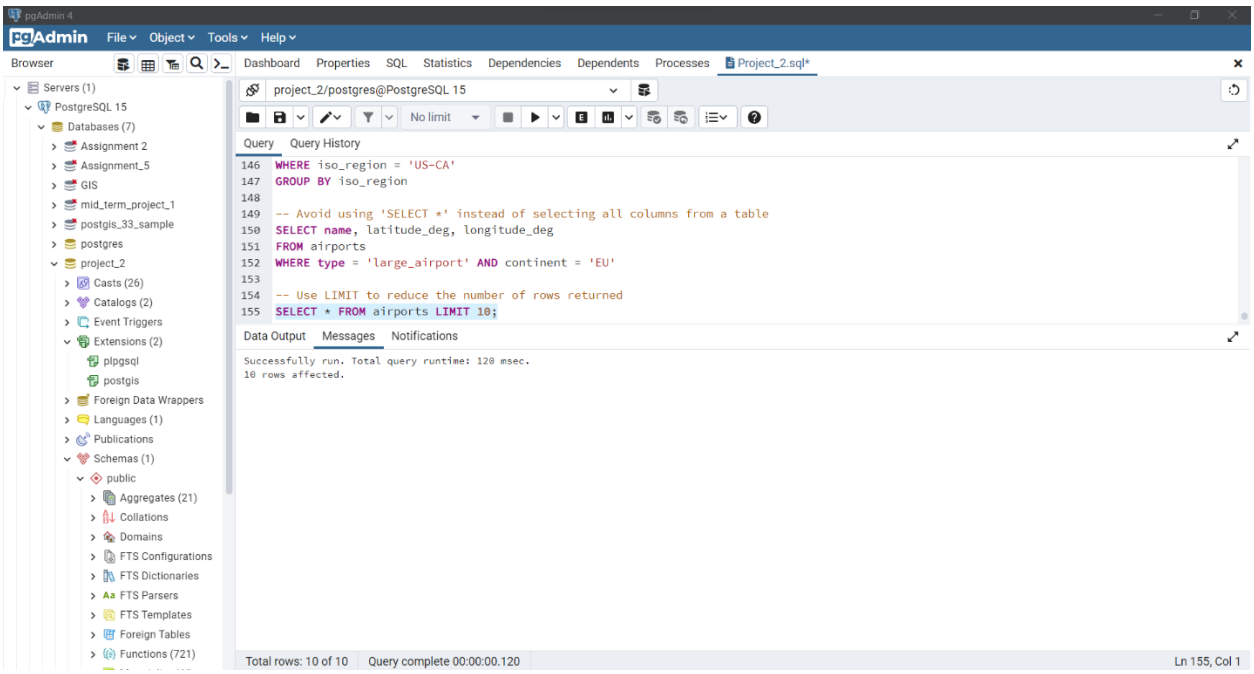
```
SELECT name, latitude_deg, longitude_deg
FROM airports
WHERE type = 'large_airport' AND continent = 'EU';
```

Group Project



- **Using LIMIT:** The LIMIT clause can significantly improve performance by reducing the number of rows returned by a query. This can be especially beneficial when you only need a certain number of rows from a large table.

SELECT * FROM airports LIMIT 10;



CS 623 - Database Management Systems

Group Project

Summary:

This project has been a significant exploration into the world of geospatial databases using PostgreSQL and its PostGIS extension. We sourced comprehensive airport data from OurAirports and successfully integrated this extensive dataset into our own database. This allowed us to perform various queries, calculations, and analyses on the data, exploring the capabilities of PostGIS extension and learning how to use SQL for geospatial analysis.

By creating indexes and implementing N-Optimization techniques, we were able to optimize our queries, leading to faster and more efficient data retrieval. We gained practical insights into handling and optimizing large datasets, understanding the balance between query performance and the computational cost of maintaining indexes.

However, our project only scratches the surface of what's possible with geospatial databases. There's a plethora of analyses and operations that can be conducted on this data, such as pathfinding, spatial joins, spatial clustering, and much more.