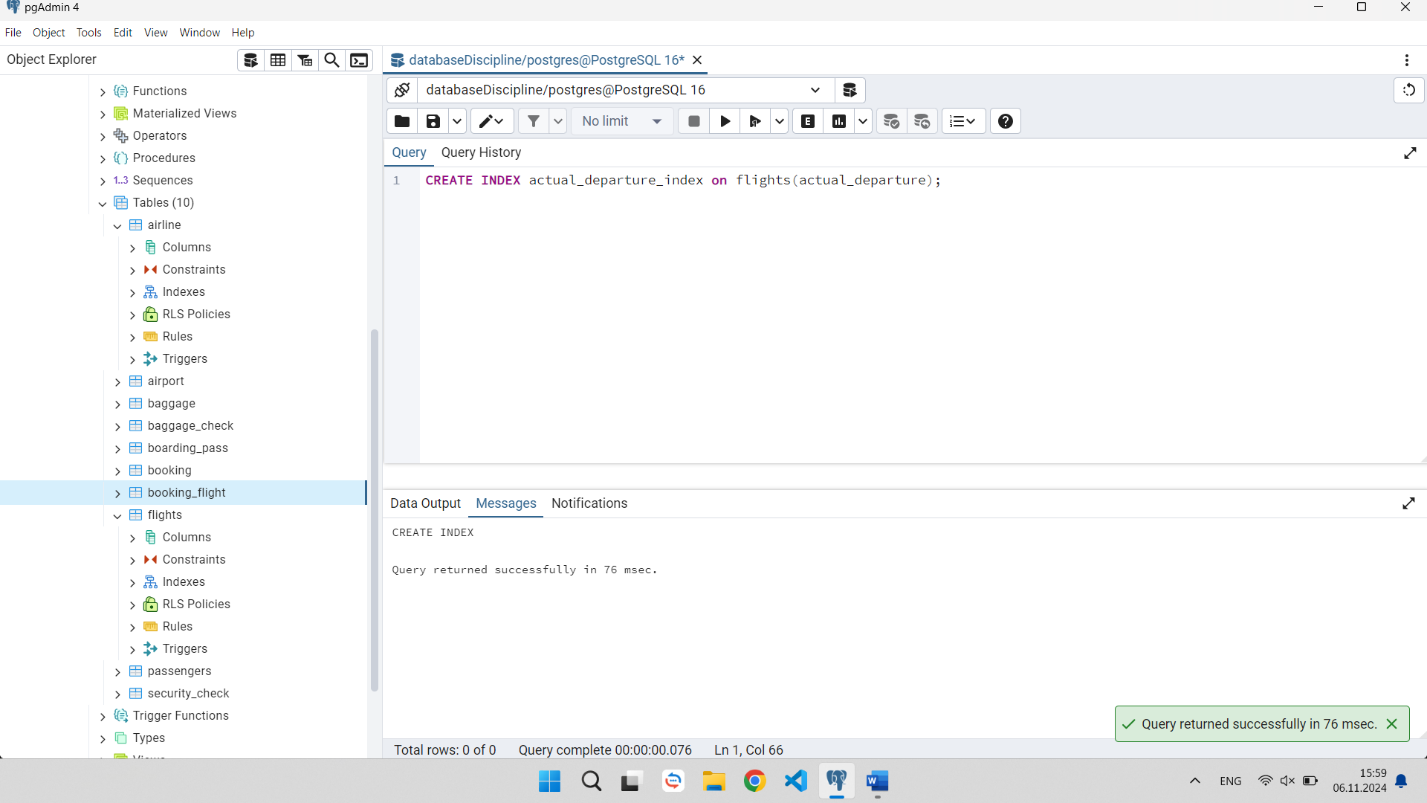
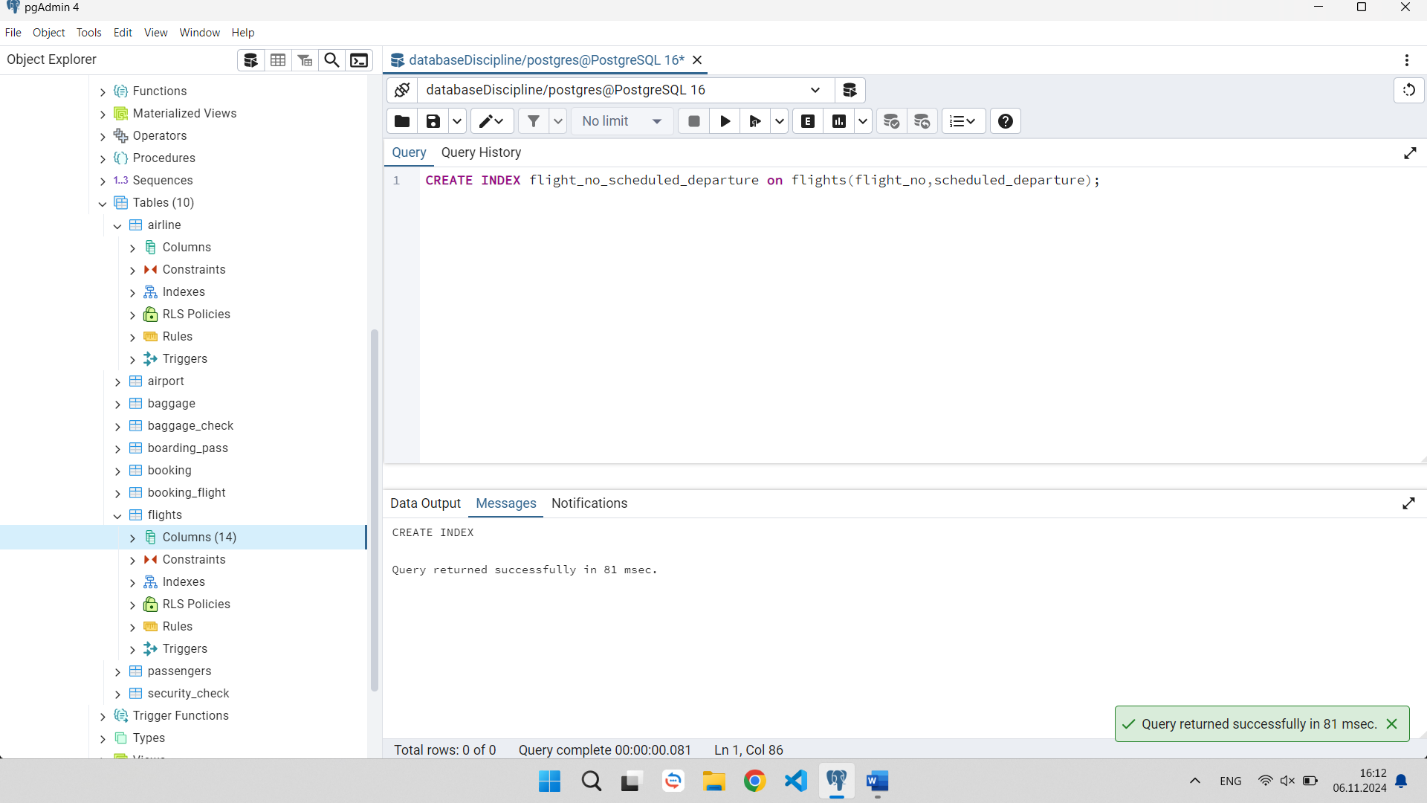
**Khakimali Zhambyl**

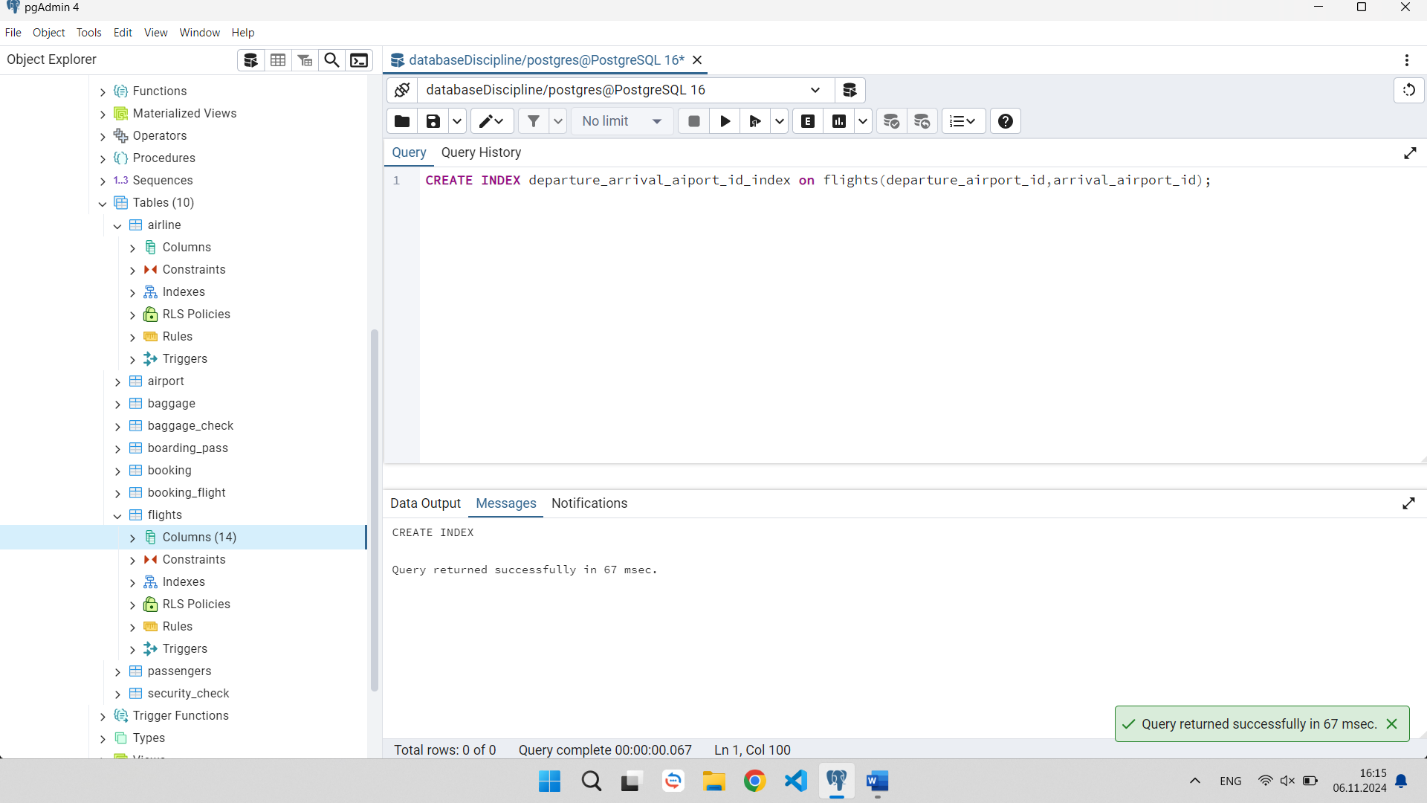
1. Create an index on the actual\_departure column in the flights table.



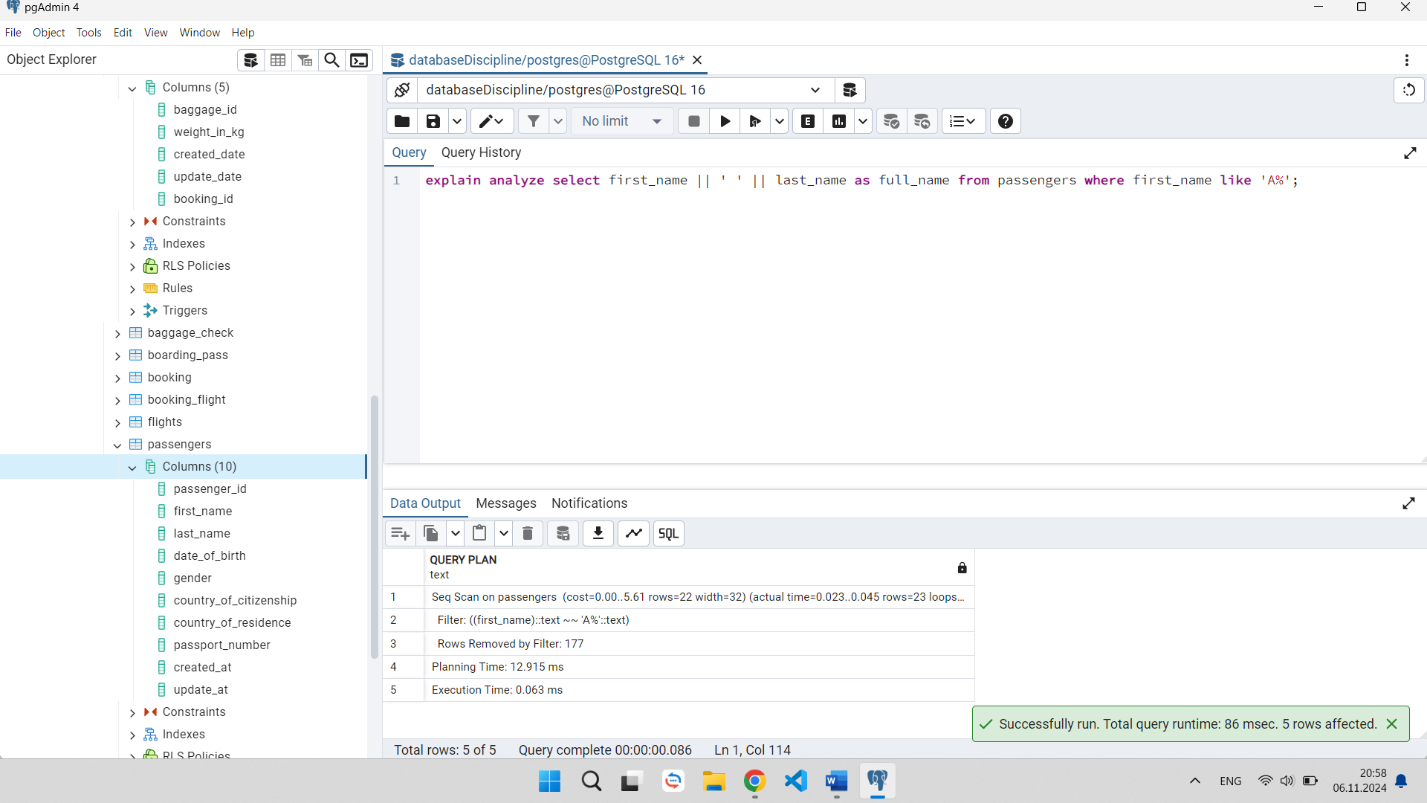
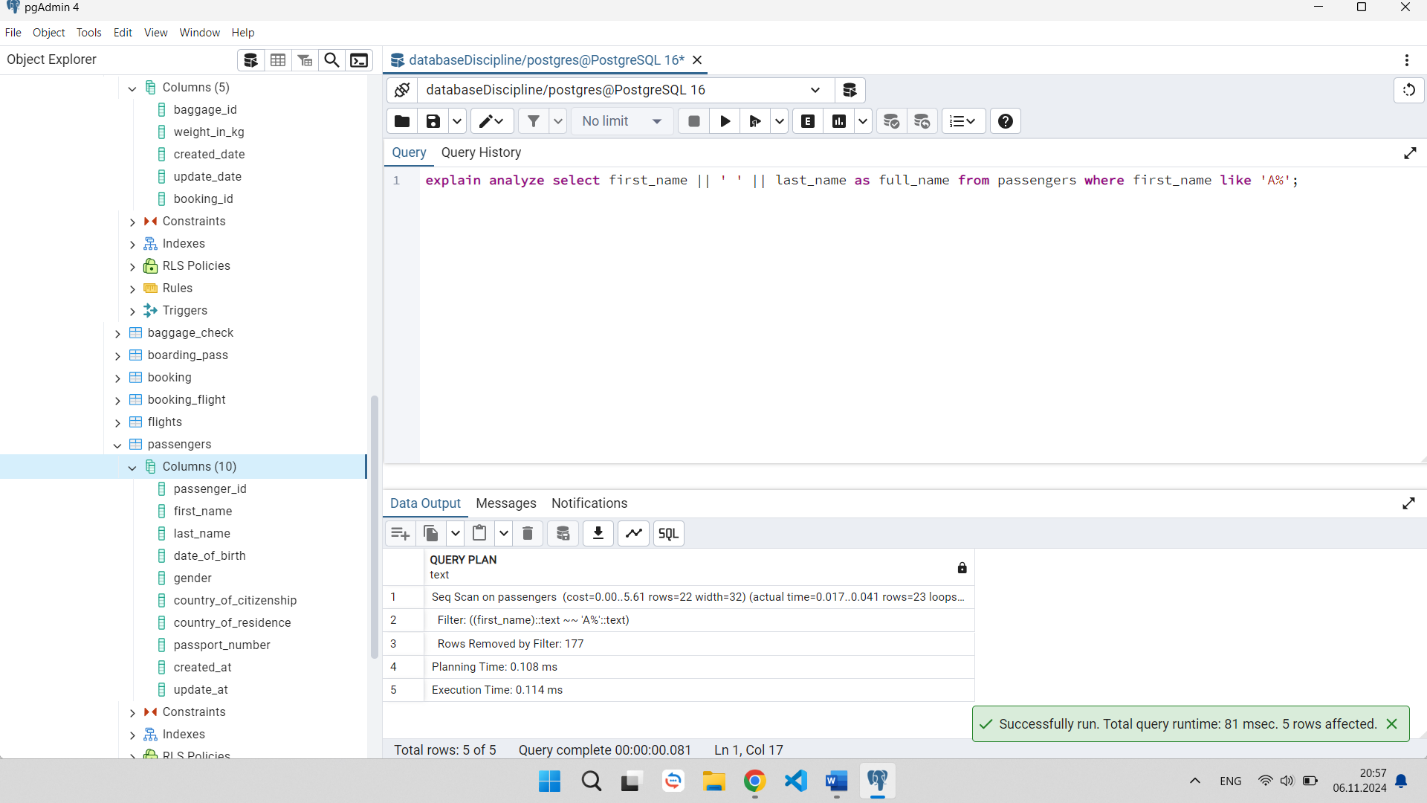
1. Create a unique index to ensure flight\_no and scheduled\_departure combinations are unique.



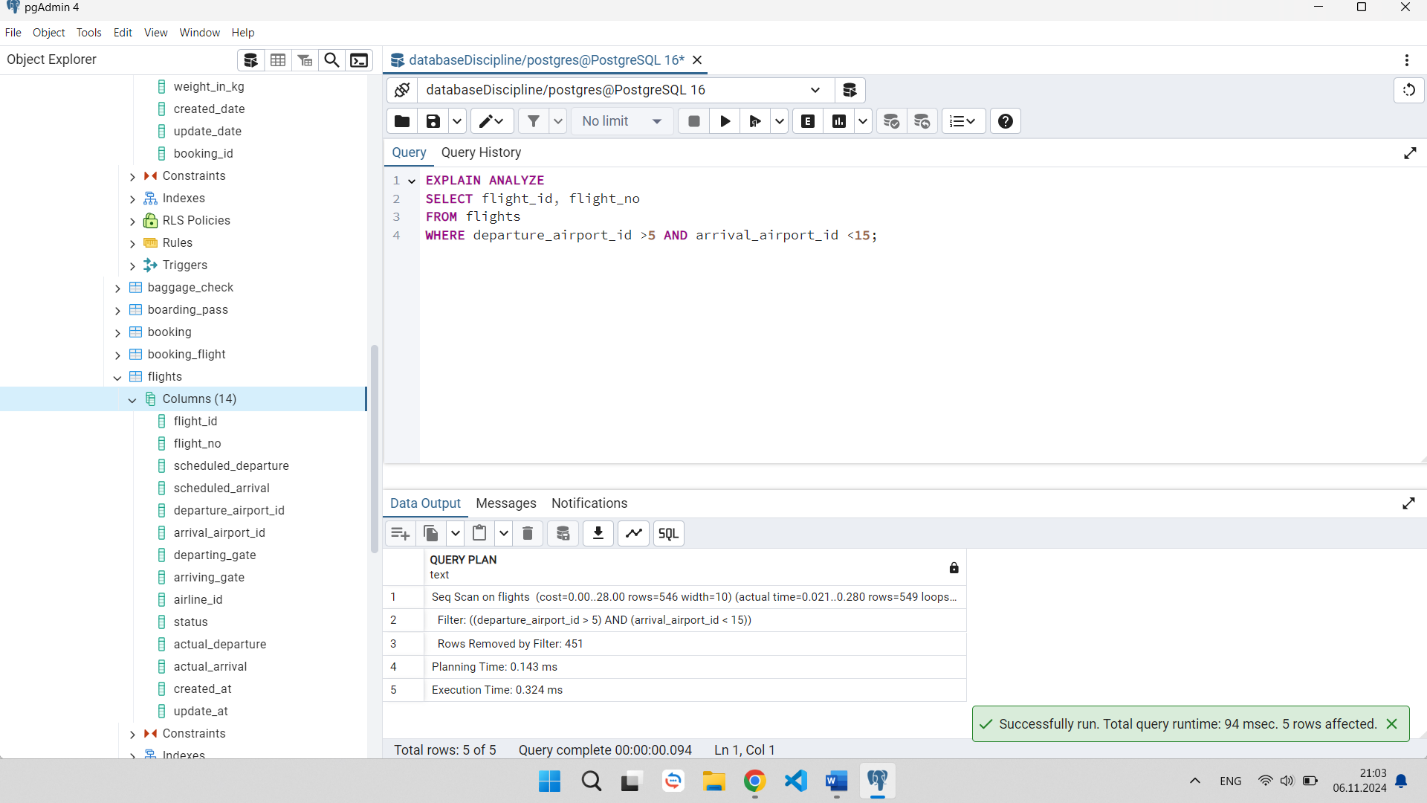
1. Create a composite index on the departure\_airport\_id and arrival\_airport\_id columns.



1. Evaluate the difference in query performance with and without indexes. Measure performance differences.

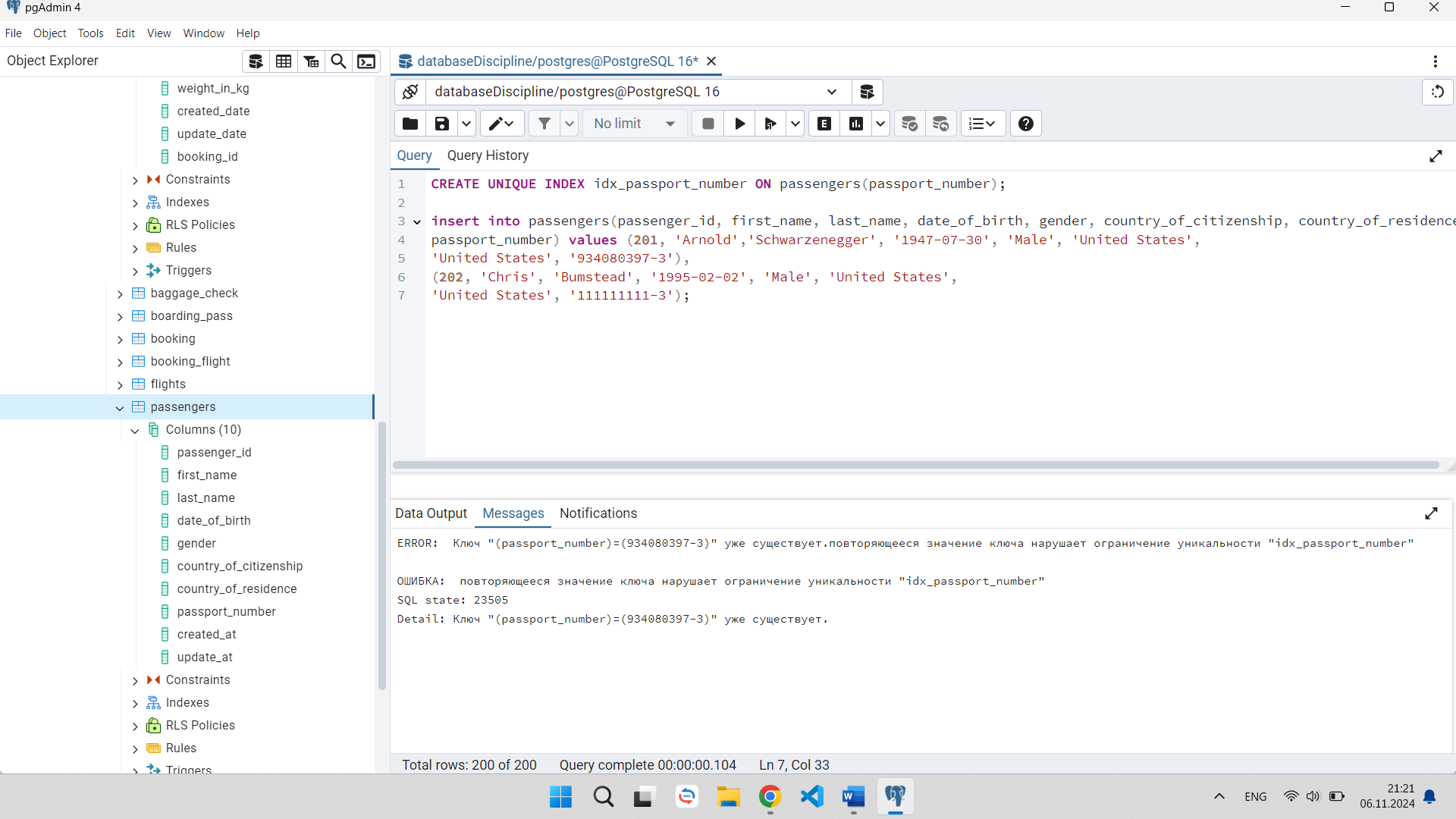


1. Use EXPLAIN ANALYZE to check index usage in a query filtering by departure\_airport and arrival\_airport.



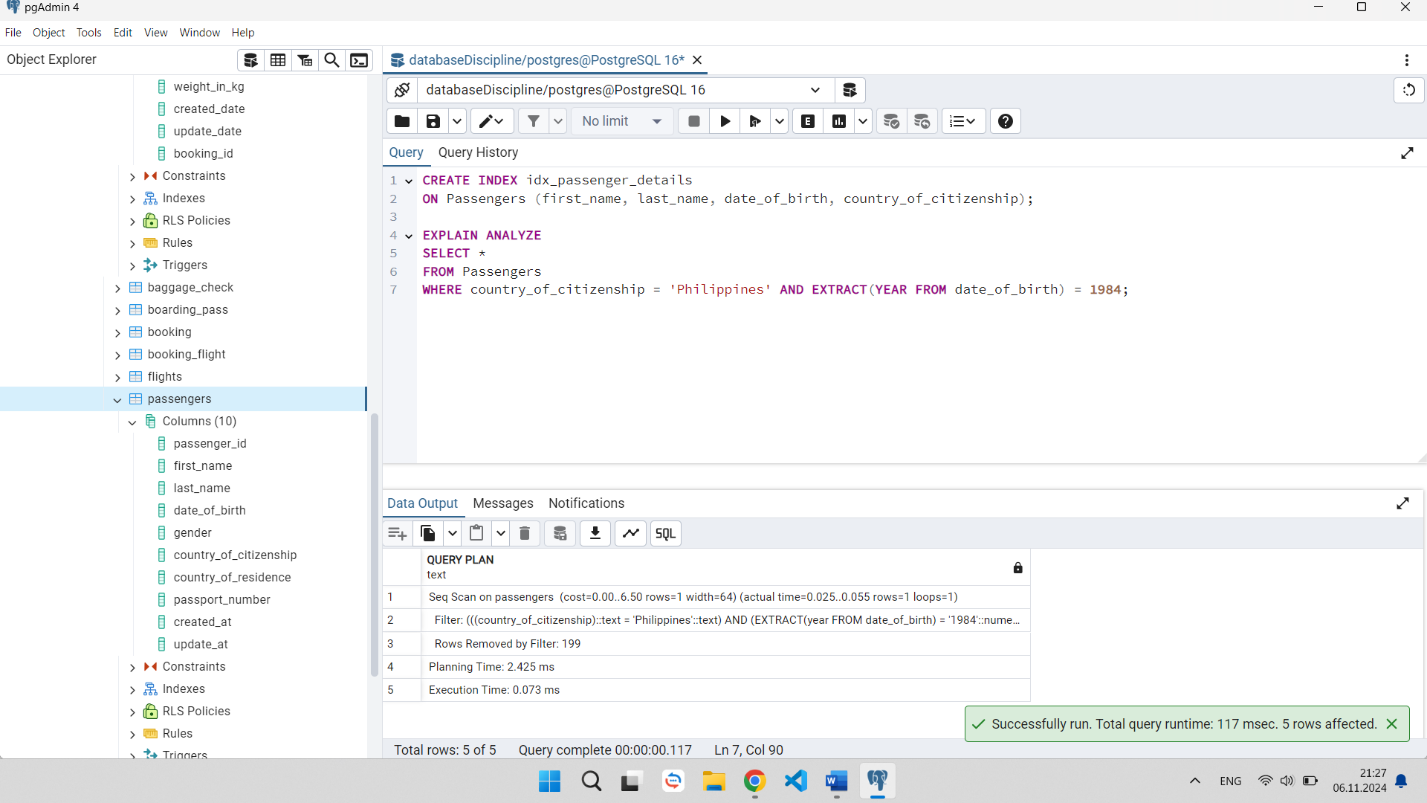
1. Create a unique index for the passport\_number of the Passengers table. Check if the index was created or not. Insert into the table two new passengers.

Explain in your own words what is going on in the output?



When we try to insert a passenger with the passport number ` 934080397-3`, the database will generate an error because the passport number already exists in the table. The unique index enforces this constraint and prevents the insertion.

1. Create an index for the Passengers table. Use for that first name, last name, date of birth and country of citizenship. Then, write a SQL query to find a passenger who was born in Philippines and was born in 1984 and check if the query uses indexes or not. Give the explanation of the results.



If the query uses multiple columns (such as country\_of\_citizenship and date\_of\_birth), the database may use a bitmap index scan. It first finds the rows matching the conditions of the index (country\_of\_citizenship = 'Philippines'), and then applies the filter on date\_of\_birth (1984).

1. Write a SQL query to list indexes for table Passengers. After delete the created indexes.

