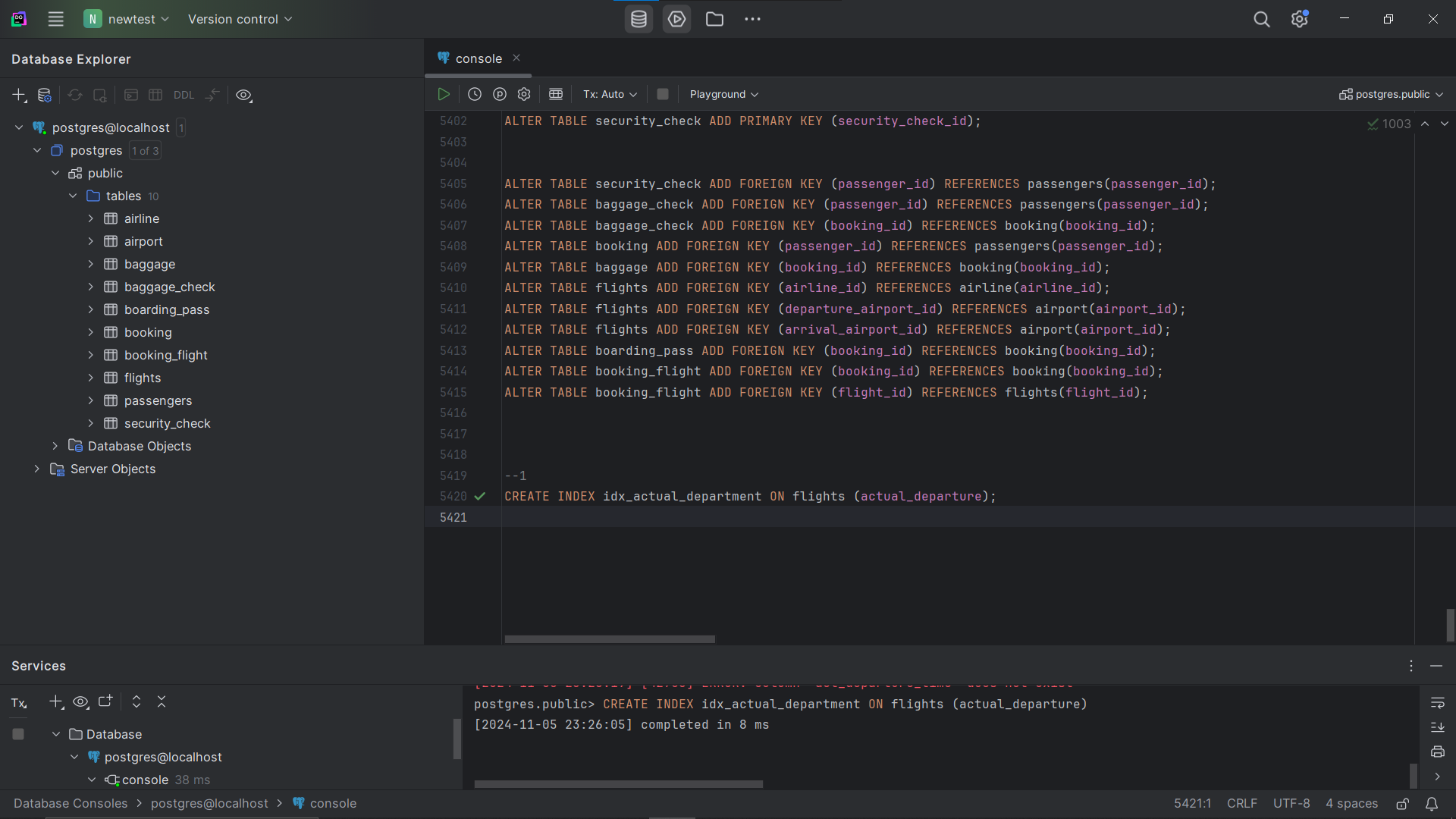
Laboratory work 7

**We continue to work with the database from the previous laboratory works.**

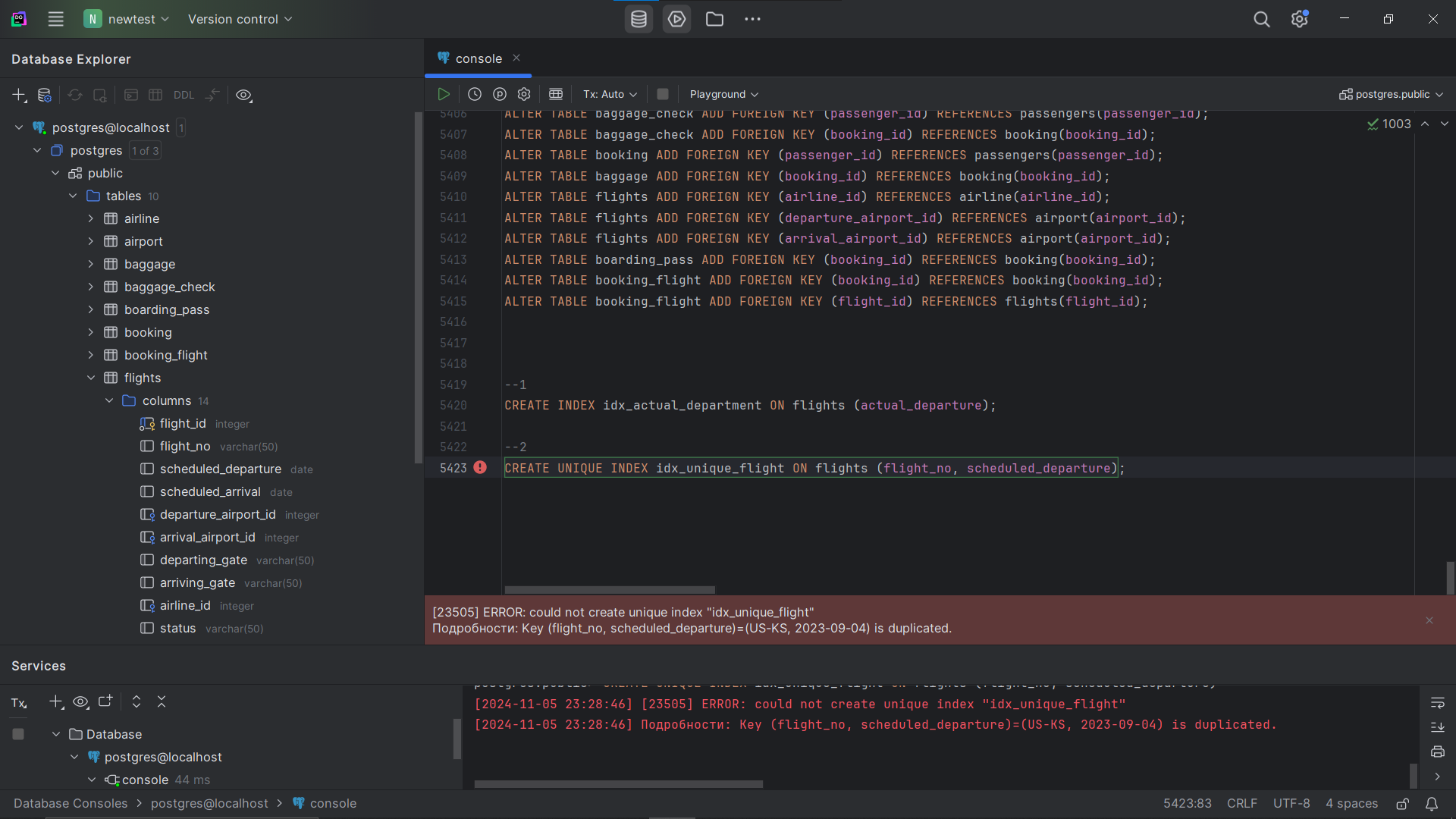
**Take a full-page screenshot that covers the code and results of each task.**

**Tasks**:

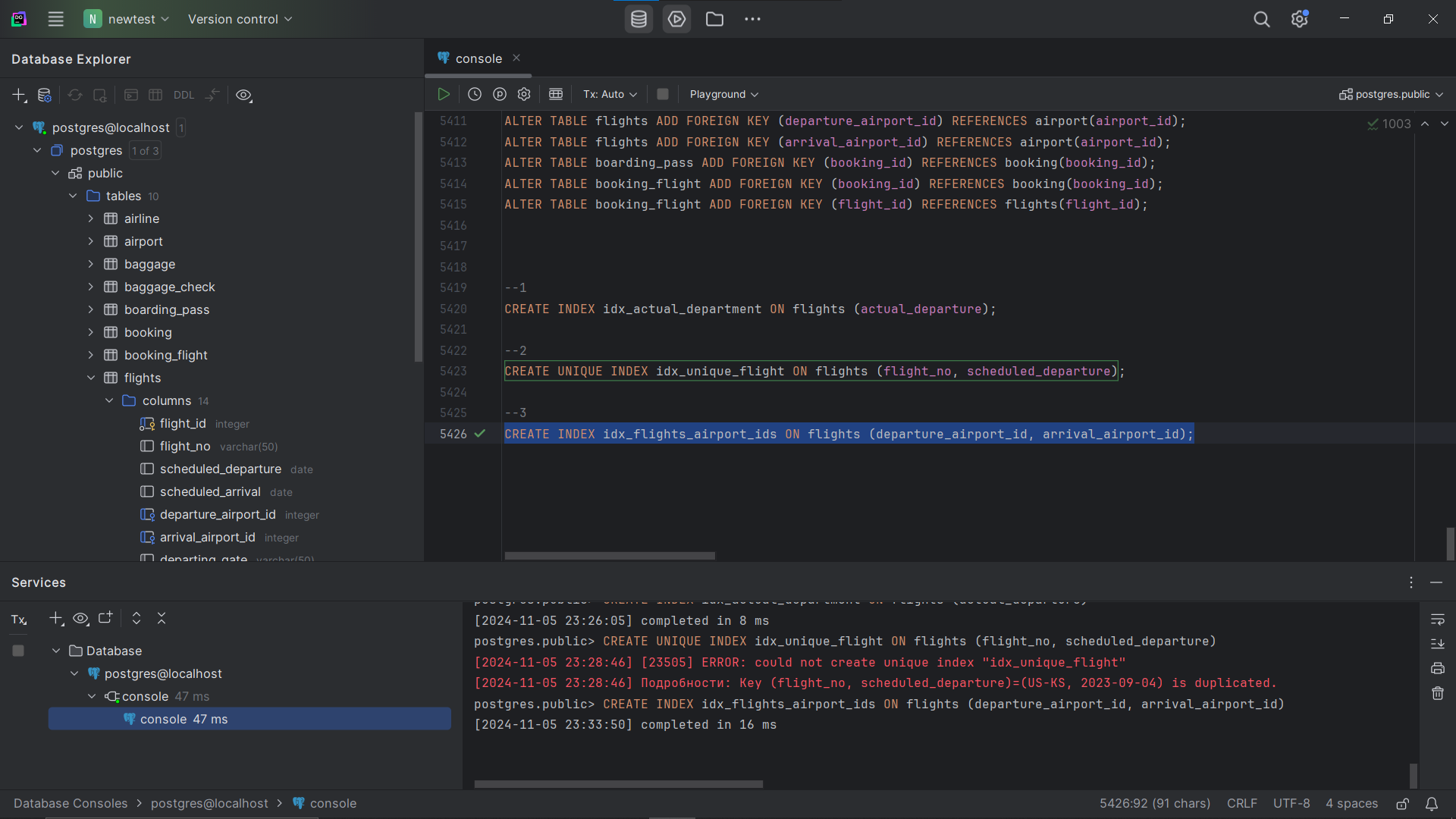
1. Create an index on the actual\_departure column in the flights table.



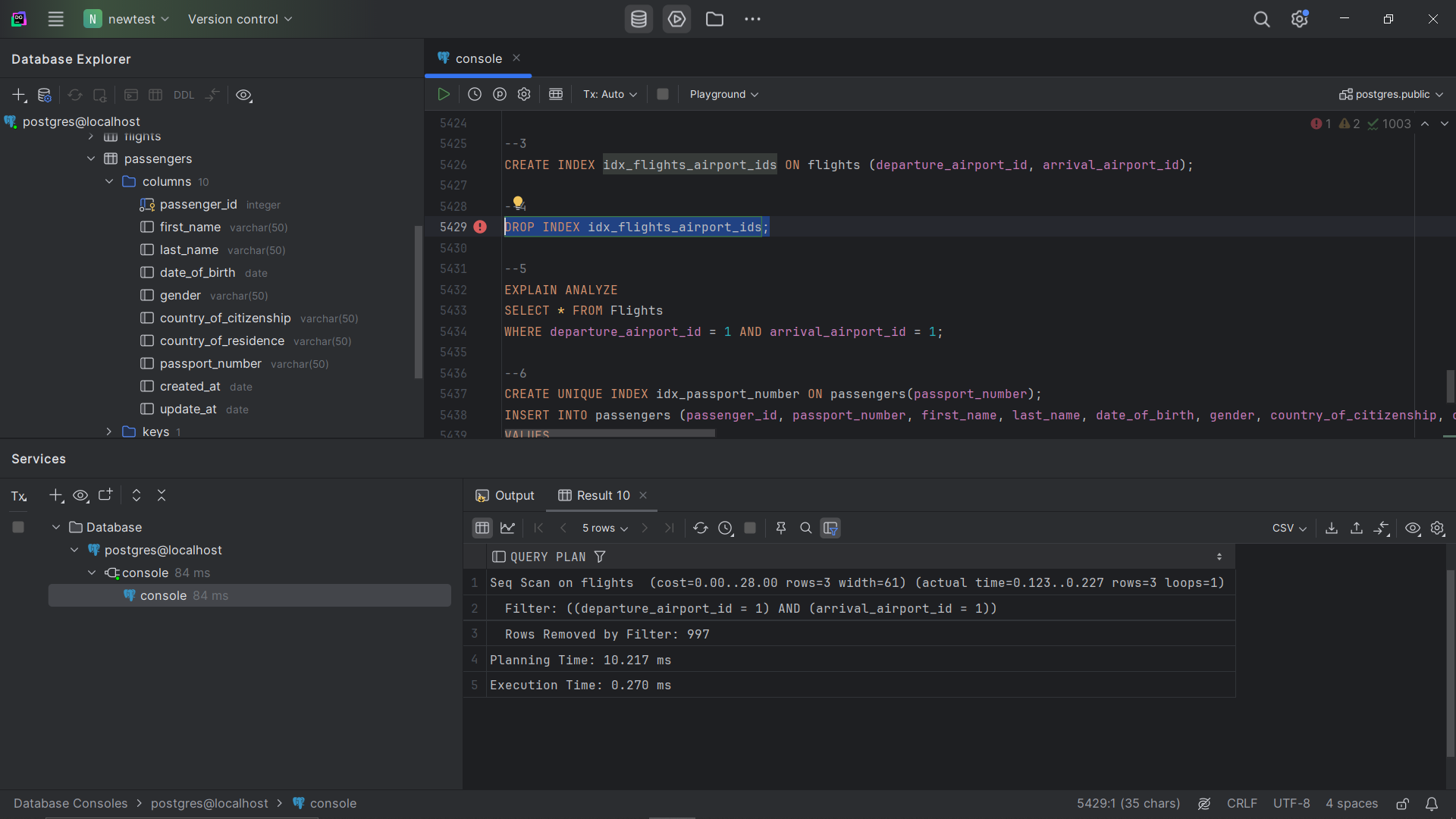
1. Create a unique index to ensure flight\_no and scheduled\_departure combinations are unique.



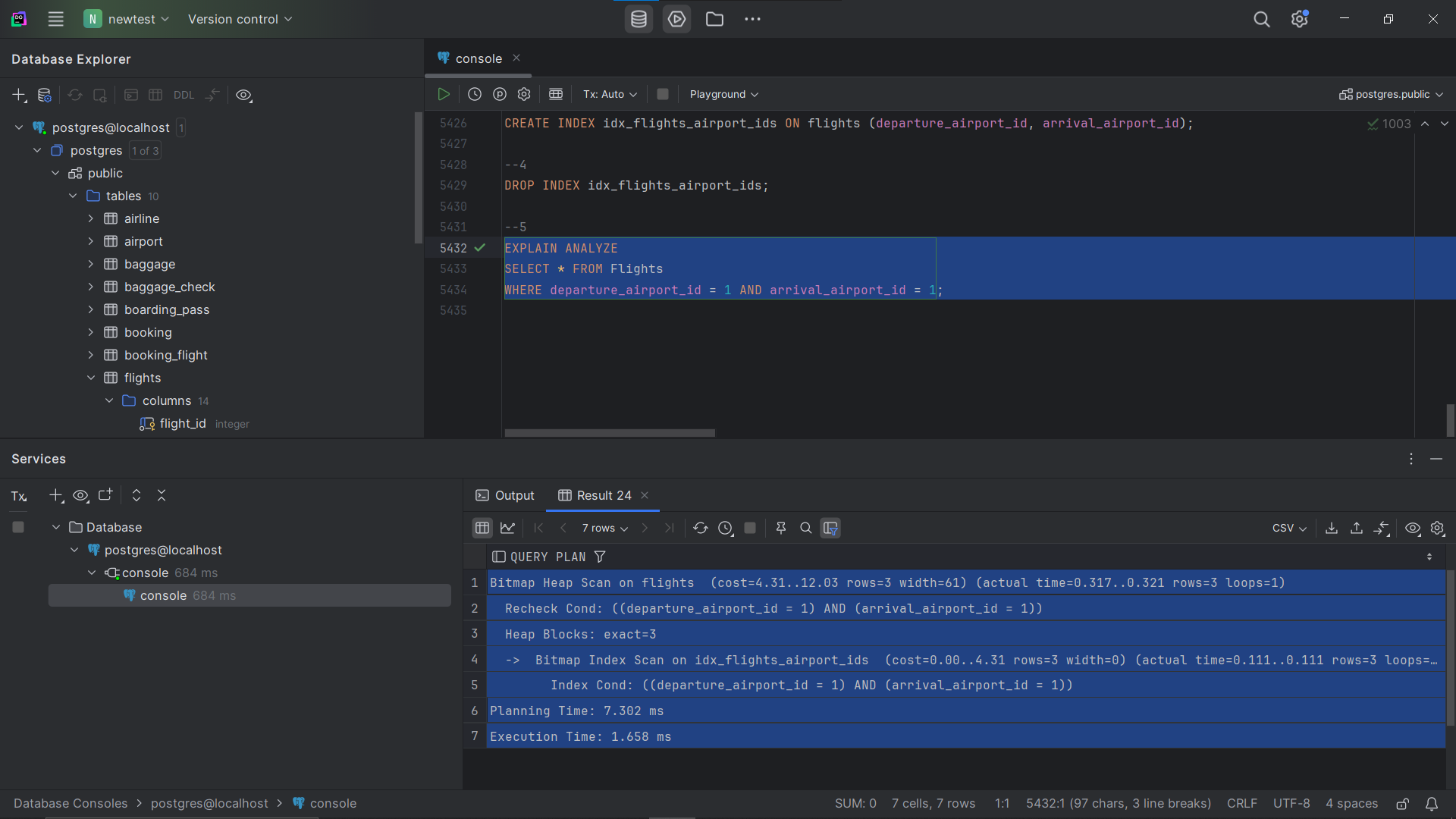
1. Create a composite index on the departure\_airport\_id and arrival\_airport\_id columns.



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

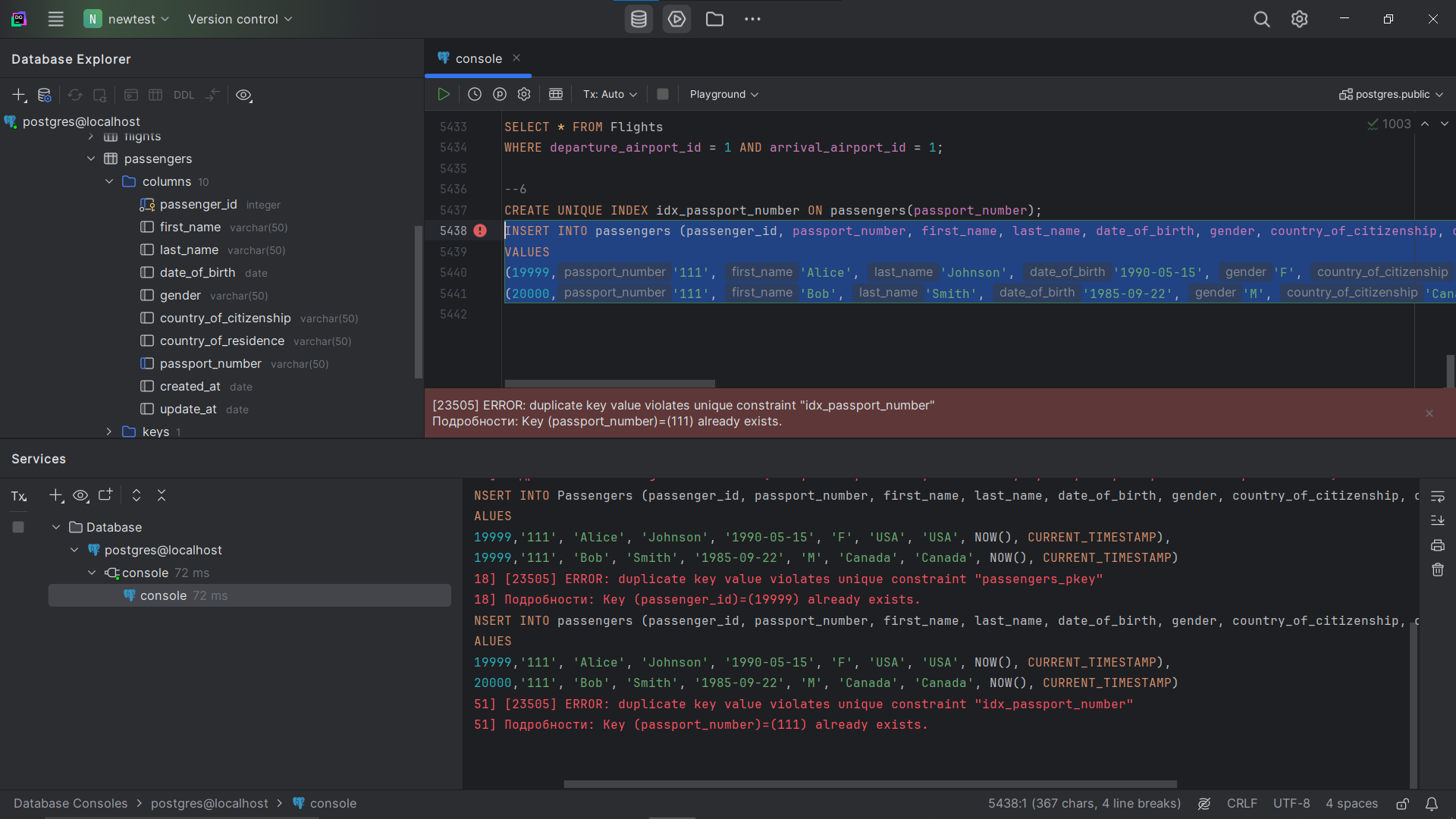


1. Use EXPLAIN ANALYZE to check index usage in a query filtering by departure\_airport and arrival\_airport.



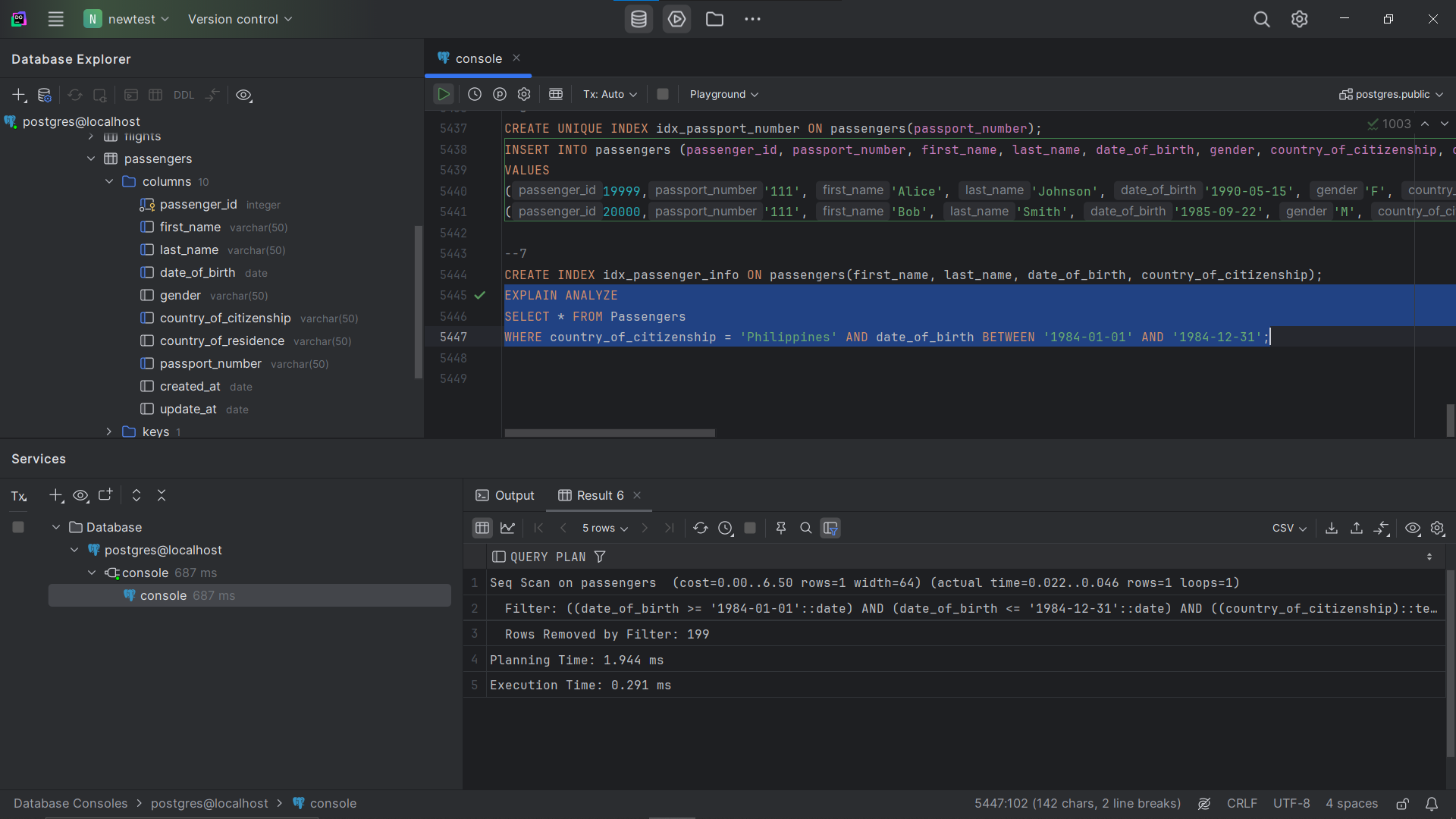
1. Create a unique index for the passport\_number of the Passengers table. Check if the index was created or not. Insert into the table two new passengers.

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



**We created unique index for passport\_number and an error occurred because we have duplicated passport\_number while inserting values for new passenger in our table**

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.



**The execution plan will show if the index is used. PostgreSQL might skip the index if filtering on country\_of\_citizenship and date\_of\_birth is not efficient, especially if these columns don’t narrow down the search enough. In such cases, a sequential scan may be faster**

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

