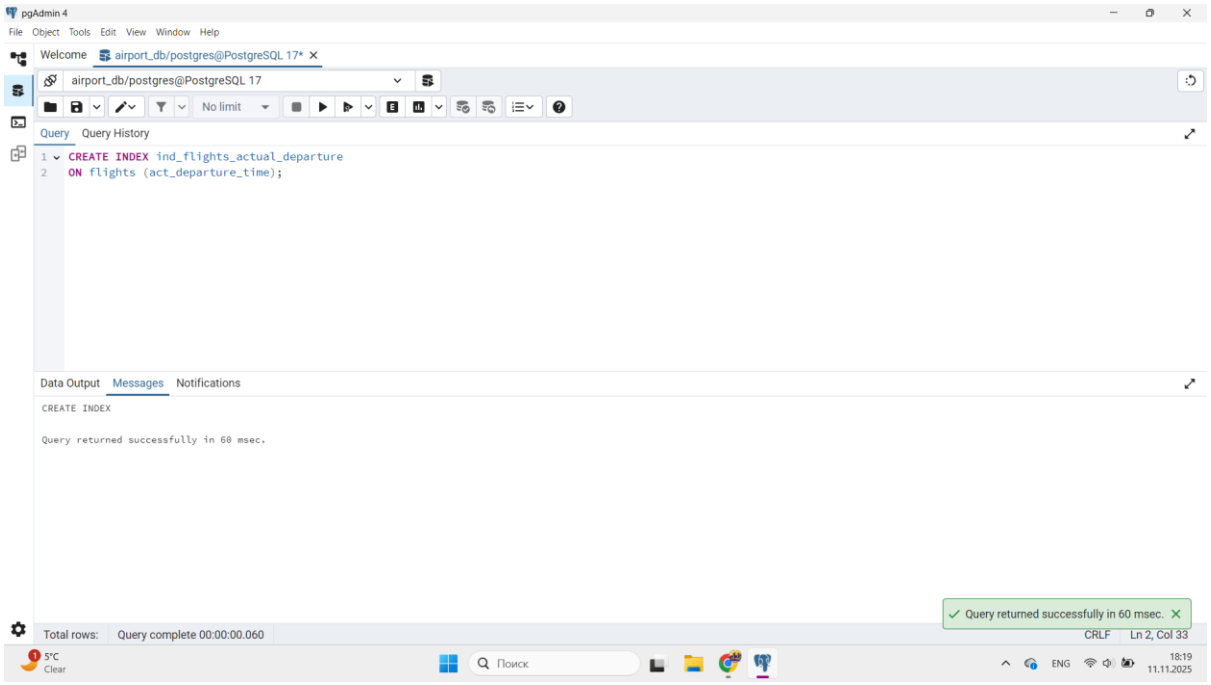
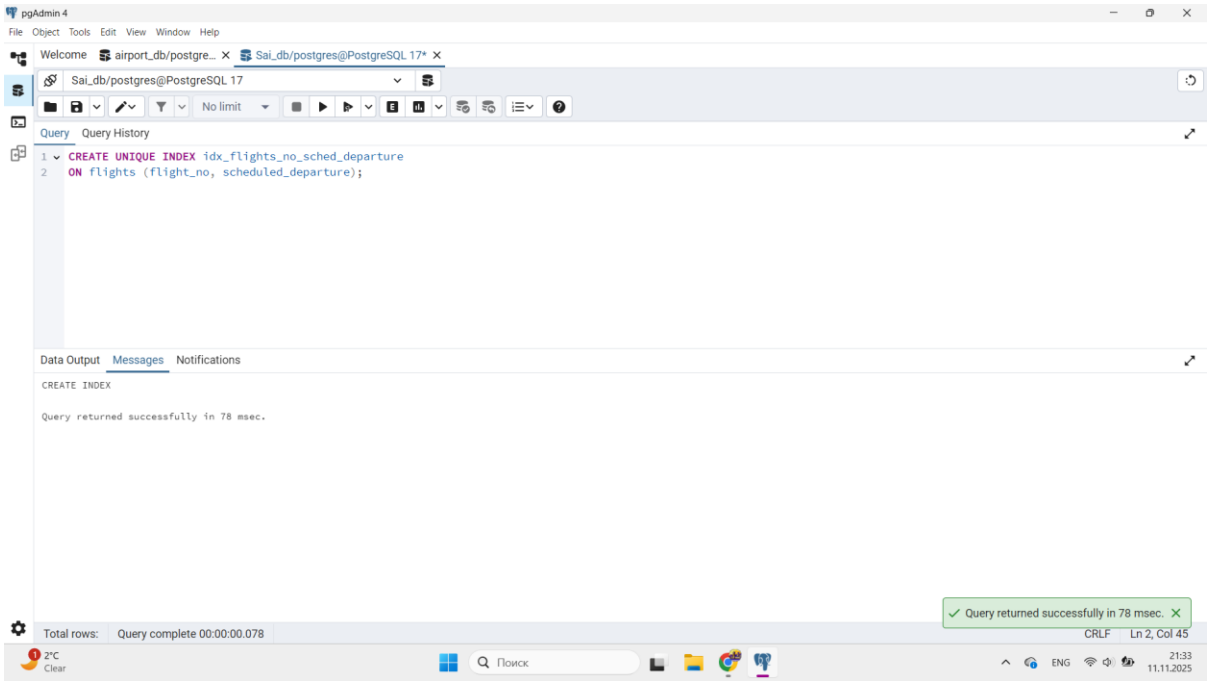
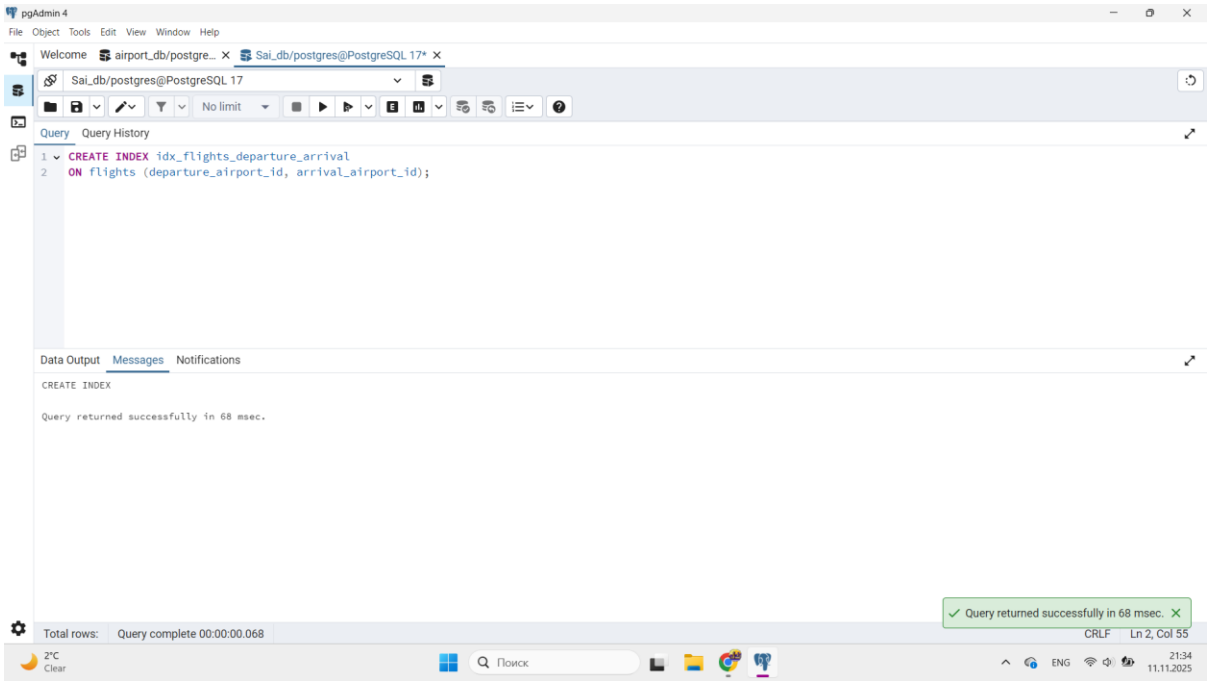
1. Create an index on the actual_departure column in the flights table.



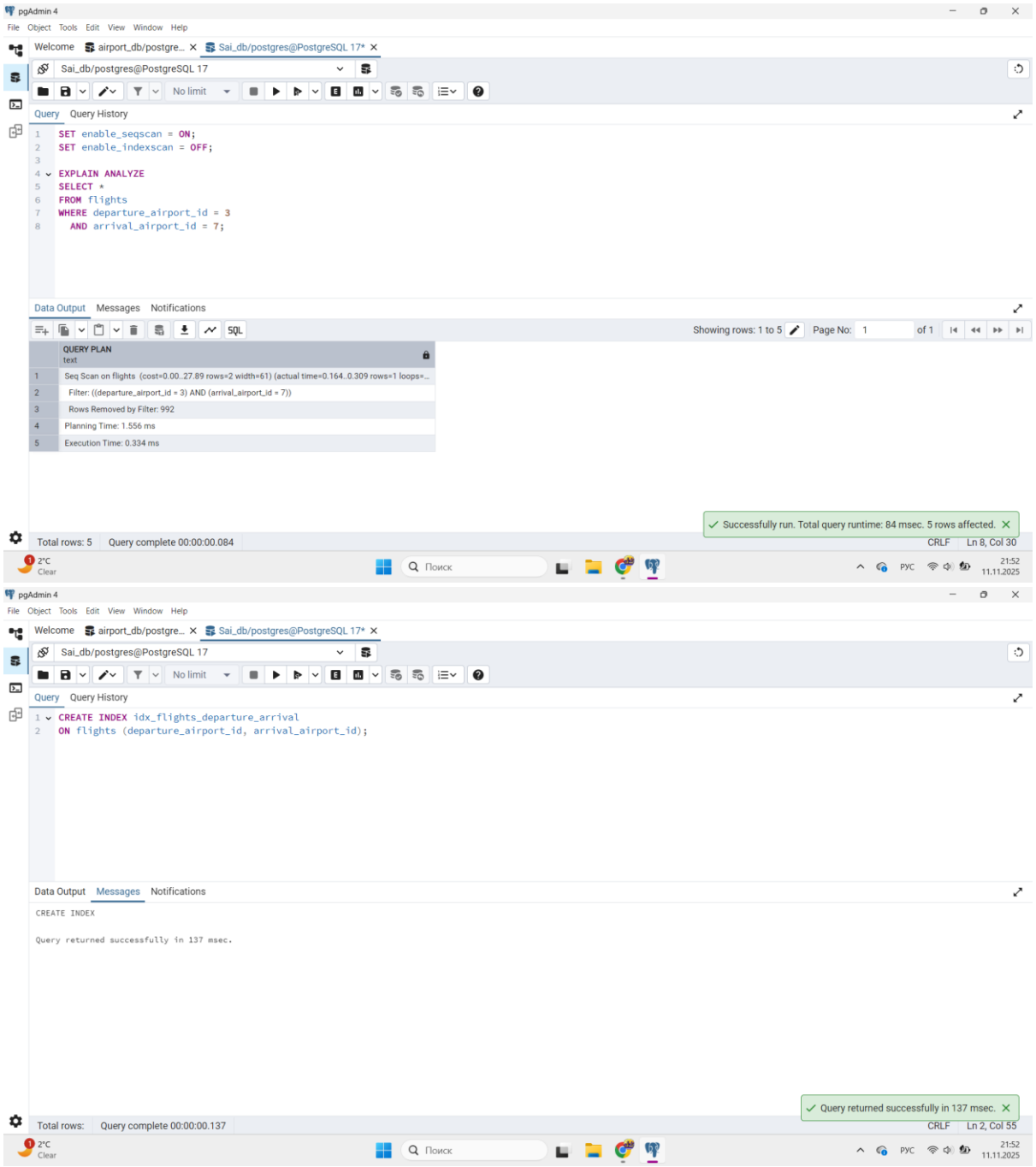
2. Create a unique index to ensure flight_no and scheduled_departure combinations are unique.



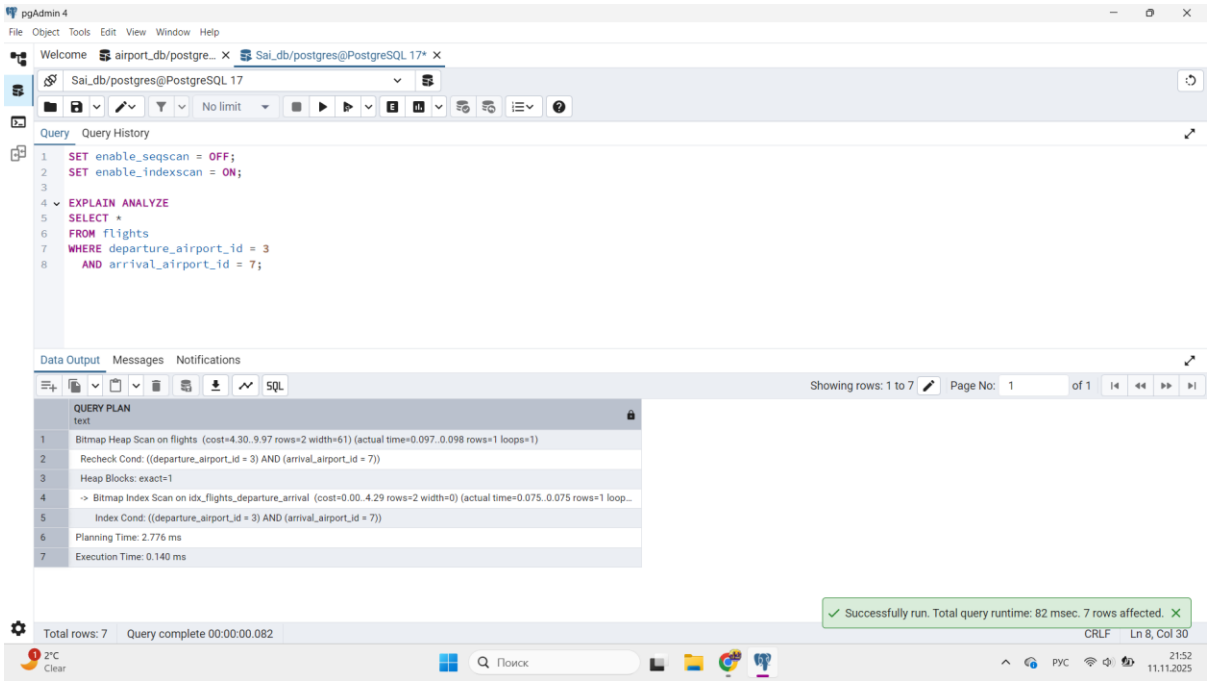
3. Create a composite index on the departure_airport_id and arrival_airport_id columns.



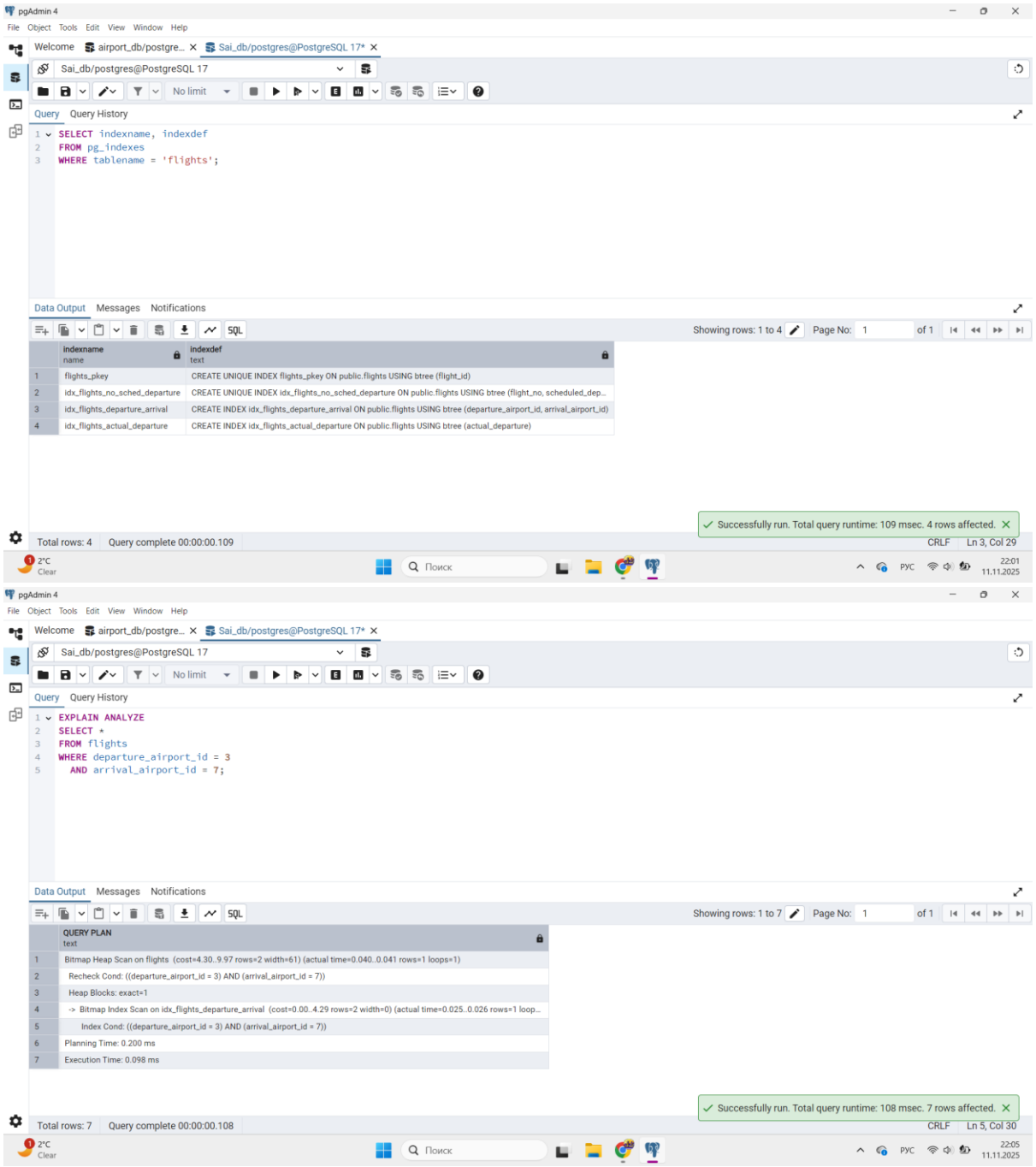
4. Evaluate the difference in query performance with and without indexes. Measure performance differences.


pgAdmin 4
File Object Tools Edit View Window Help
Welcome
airport_db/postgre...
Sai_db/postgres@PostgreSQL 17*
Sai_db/postgres@PostgreSQL 17
No limit
Query
Query History
SET enable_seqscan = ON;
SET enable_indexscan = OFF;
EXPLAIN ANALYZE
SELECT *
FROM flights
WHERE departure_airport_id = 3
AND arrival_airport_id = 7;
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 5
Page No: 1
of 1
QUERY PLAN
text
Seq Scan on flights (cost=0.00..27.89 rows=2 width=61) (actual time=0.164..0.309 rows=1 loops=...
Filter: ((departure_airport_id = 3) AND (arrival_airport_id = 7))
Rows Removed by Filter: 992
Planning Time: 1.556 ms
Execution Time: 0.334 ms
Successfully run. Total query runtime: 84 msec. 5 rows affected.
Total rows: 5
Query complete 00:00:00.084
CRLF
Ln 8, Col 30
2°C
Clear
Поиск
РУС
21:52
11.11.2025

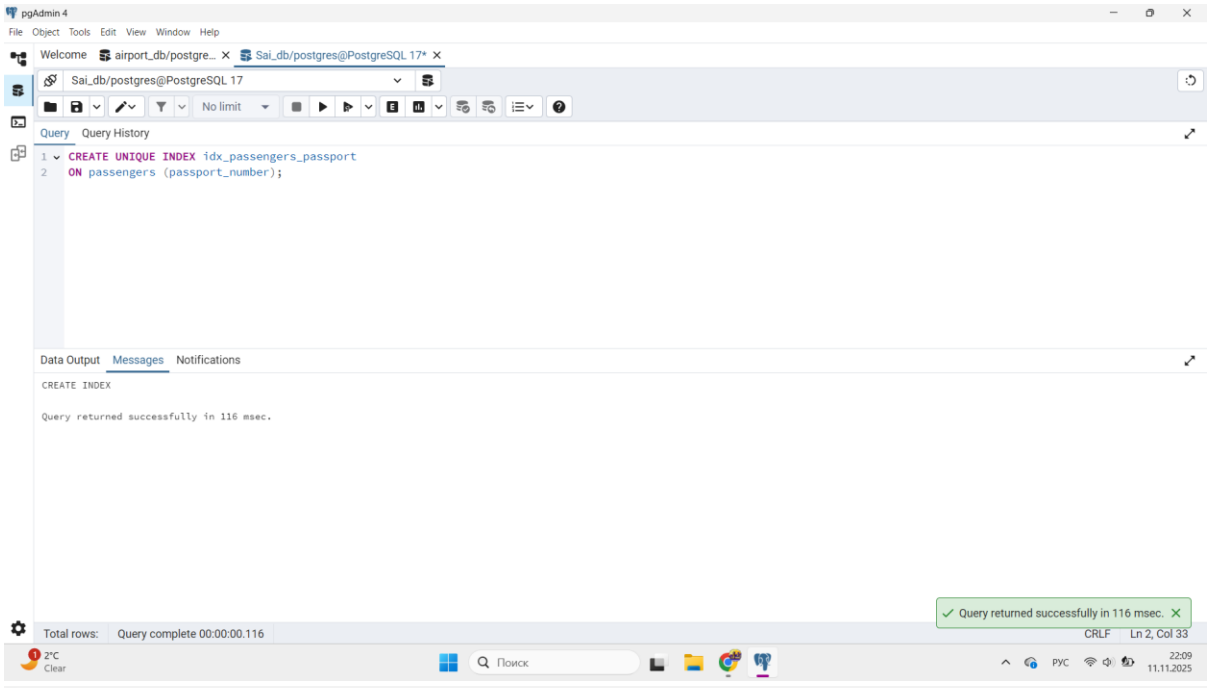
pgAdmin 4
File Object Tools Edit View Window Help
Welcome
airport_db/postgre...
Sai_db/postgres@PostgreSQL 17*
Sai_db/postgres@PostgreSQL 17
No limit
Query
Query History
CREATE INDEX idx_flights_departure_arrival
ON flights (departure_airport_id, arrival_airport_id);
Data Output
Messages
Notifications
CREATE INDEX
Query returned successfully in 137 msec.
Query returned successfully in 137 msec.
Total rows:
Query complete 00:00:00.137
CRLF
Ln 2, Col 55
2°C
Clear
Поиск
РУС
21:52
11.11.2025

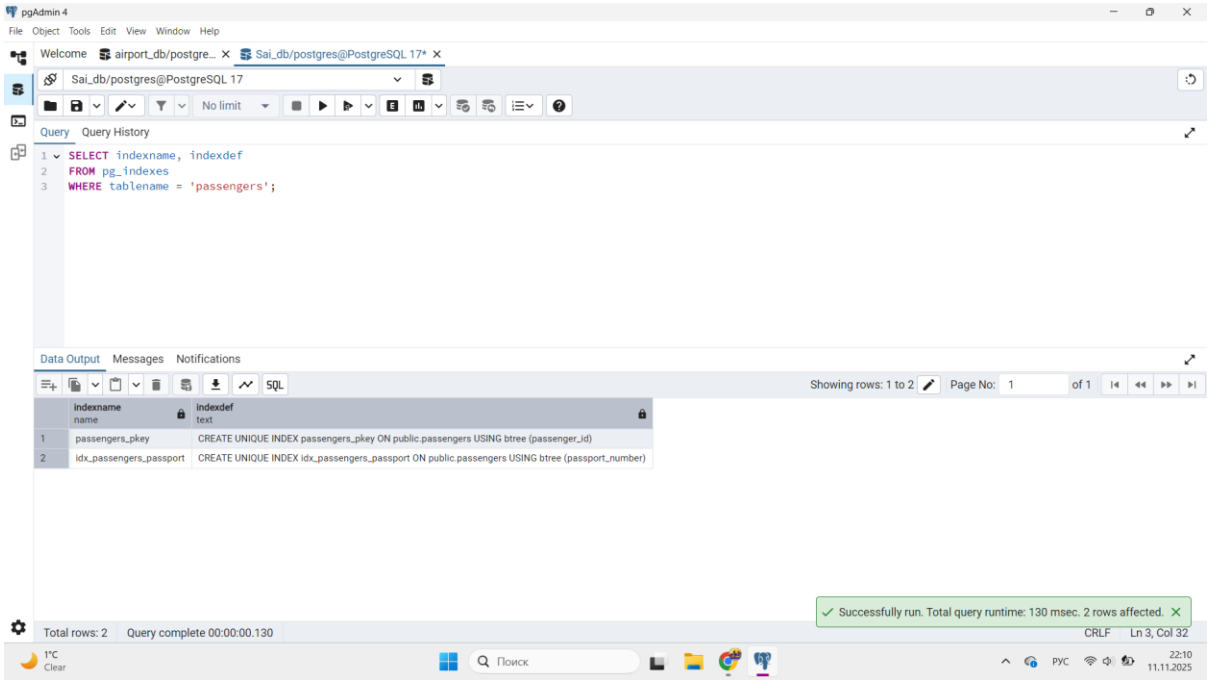


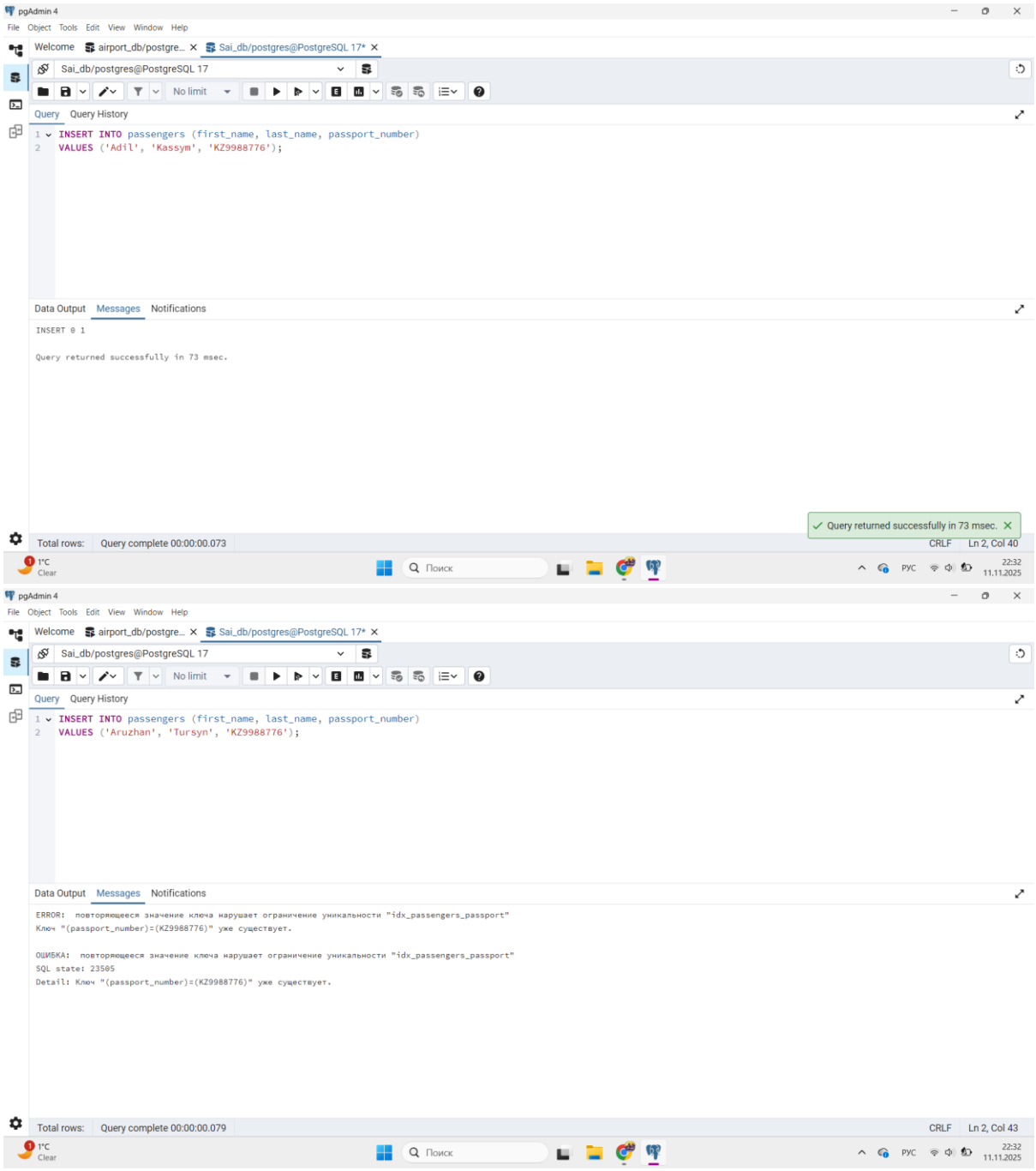
5. Use EXPLAIN ANALYZE to check index usage in a query filtering by departure_airport and arrival_airport.



6. 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.


pgAdmin 4
File Object Tools Edit View Window Help
Welcome airport_db/postgre... Sai_db/postgres@PostgreSQL 17*
Sai_db/postgres@PostgreSQL 17
No limit
Query Query History
1 CREATE UNIQUE INDEX idx_passengers_passport
2 ON passengers (passport_number);
Data Output Messages Notifications
CREATE INDEX
Query returned successfully in 116 msec.
Query returned successfully in 116 msec.
Total rows: Query complete 00:00:00.116
CRLF Ln 2, Col 33
2°C Clear
Поиск
РУС 22:09 11.11.2025

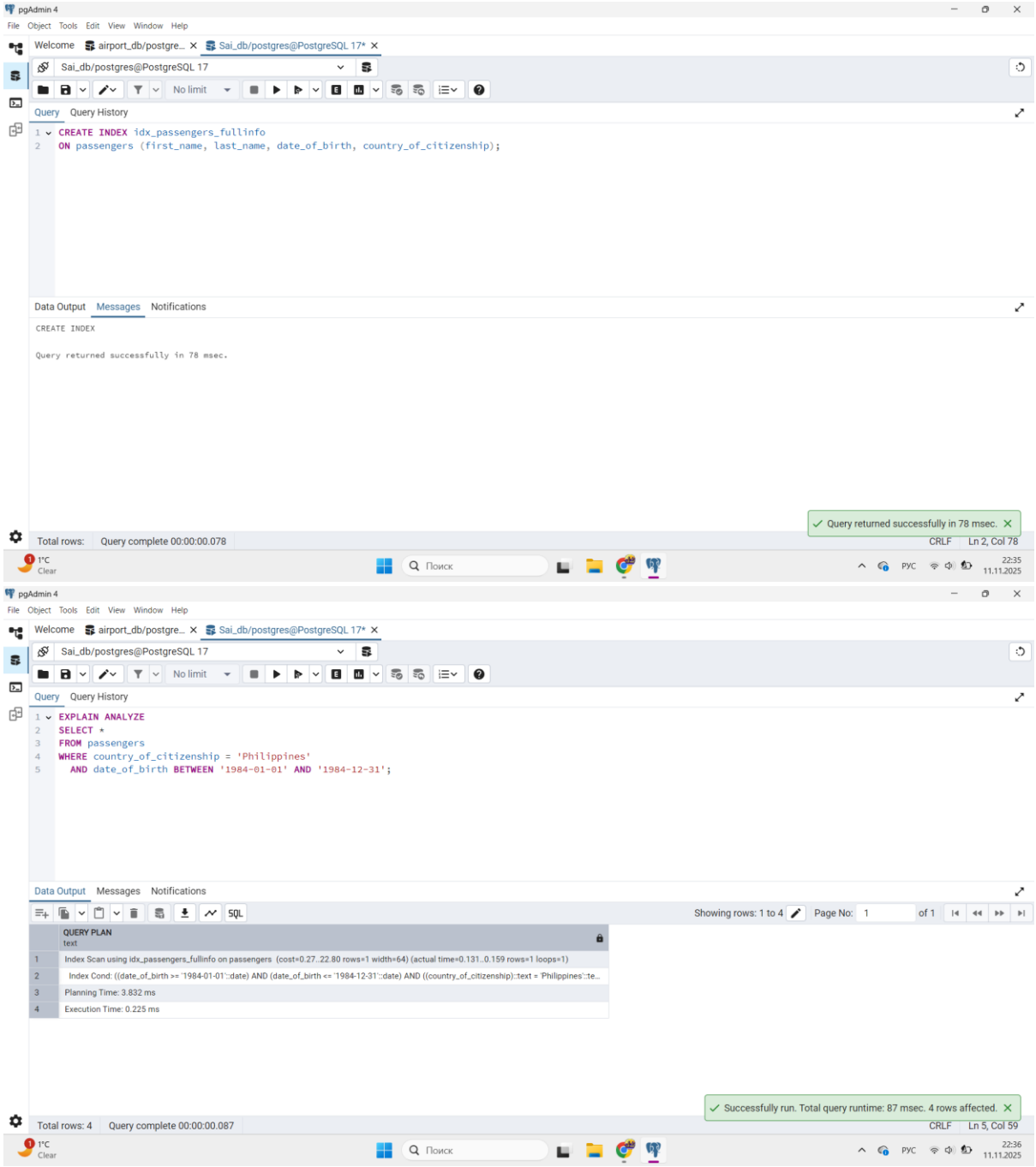

pgAdmin 4
File Object Tools Edit View Window Help
Welcome airport_db/postgre... Sai_db/postgres@PostgreSQL 17*
Sai_db/postgres@PostgreSQL 17
No limit
Query Query History
1 SELECT indexname, indexdef
2 FROM pg_indexes
3 WHERE tablename = 'passengers';
Data Output Messages Notifications
SQL
Showing rows: 1 to 2 Page No: 1 of 1
indexname name
indexdef text
1 passengers_pkey CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
2 idx_passengers_passport CREATE UNIQUE INDEX idx_passengers_passport ON public.passengers USING btree (passport_number)
Successfully run. Total query runtime: 130 msec. 2 rows affected.
Total rows: 2 Query complete 00:00:00.130
CRLF Ln 3, Col 32
1°C Clear
Поиск
РУС 22:10 11.11.2025



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

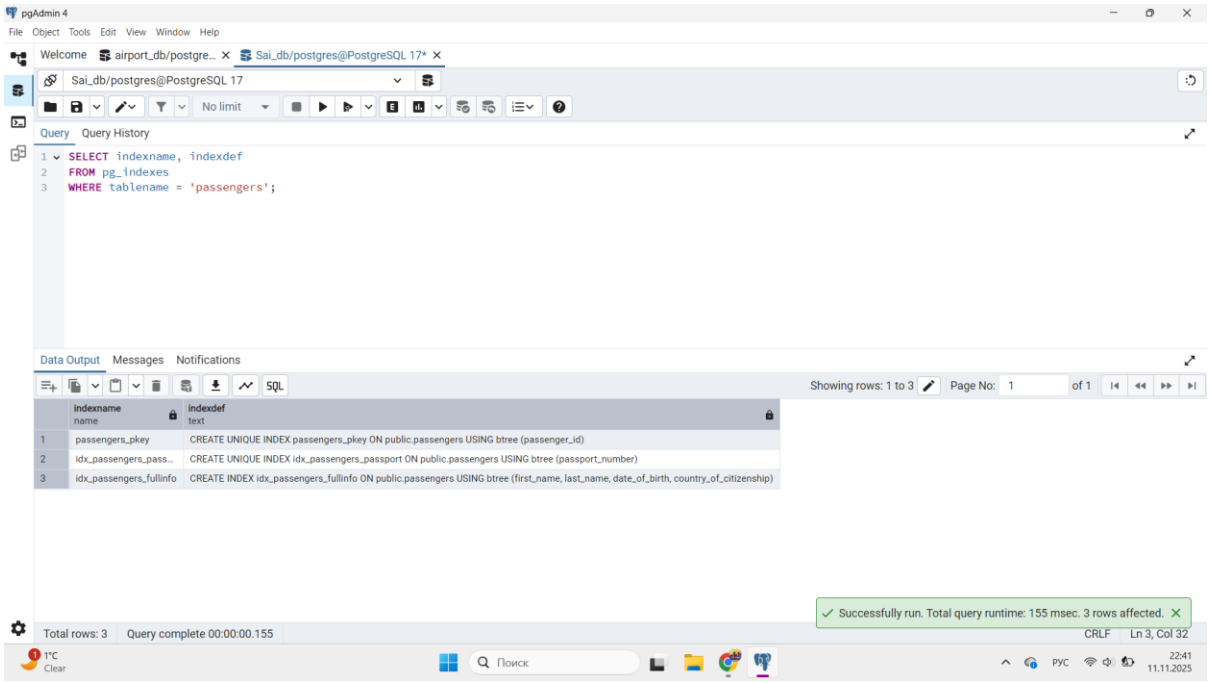
When we inserted the first passenger, the database accepted it — because the passport number was new. When we tried to insert the second passenger with the same passport number, PostgreSQL showed an error. This happened because we created a **unique index** on the `passport_number` column. It means every passenger must have a **different** passport number. So the database **stopped** the second insert to keep the data correct and avoid duplicates.

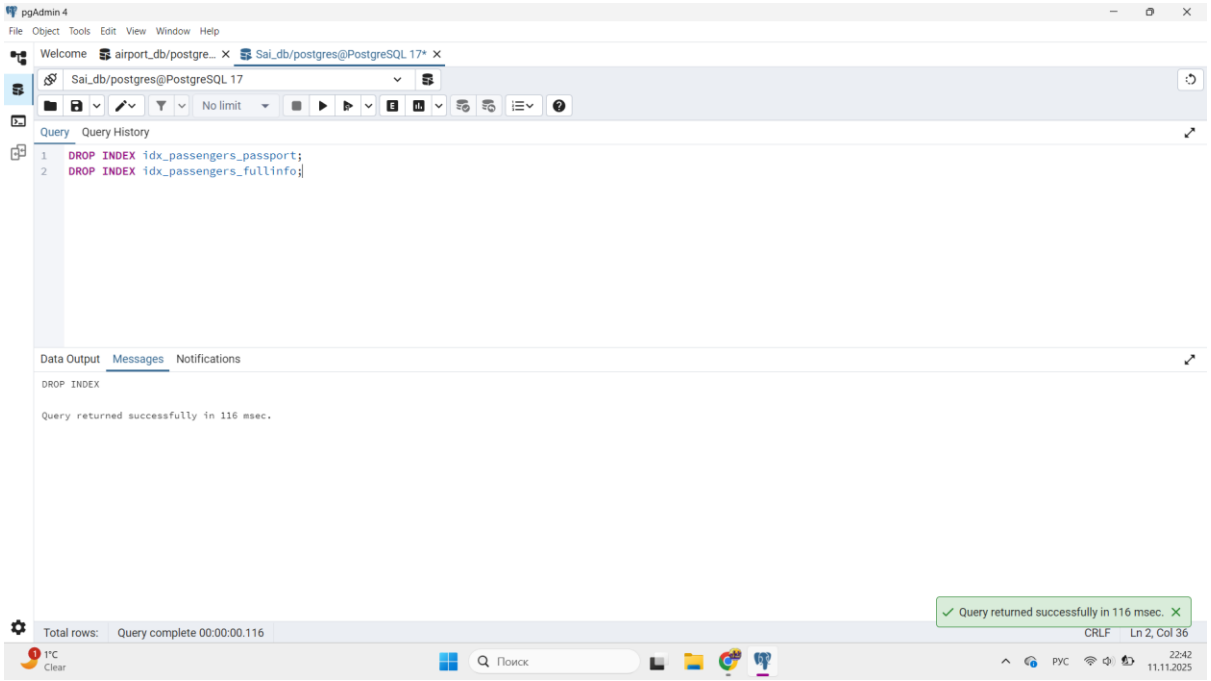
7. Create an index for the Passengers table. Use for that first name, last name, date of birth and country of citizenship. Then, write a SQL query to find a passenger who was born in Philippines and was born in 1984 and check if the query uses indexes or not. Give the explanation of the results.



PostgreSQL used the composite index when we searched for a passenger by birth year and citizenship. The query was much faster because the database did not scan the whole table – it used the index to find only matching rows. If there is no index or the query does not match its columns, PostgreSQL does a sequential scan and becomes slower.

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


pgAdmin 4
File Object Tools Edit View Window Help
Welcome airport_db/postgre... Sai_db/postgres@PostgreSQL 17*
Sai_db/postgres@PostgreSQL 17
No limit
Query Query History
SELECT indexname, indexdef
FROM pg_indexes
WHERE tablename = 'passengers';
Data Output Messages Notifications
SQL
Showing rows: 1 to 3 Page No: 1 of 1
indexname name
indexdef text
1 passengers_pkey CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
2 idx_passengers_pass... CREATE UNIQUE INDEX idx_passengers_passport ON public.passengers USING btree (passport_number)
3 idx_passengers_fullinfo CREATE INDEX idx_passengers_fullinfo ON public.passengers USING btree (first_name, last_name, date_of_birth, country_of_citizenship)
Successfully run. Total query runtime: 155 msec. 3 rows affected.
Total rows: 3 Query complete 00:00:00.155
CRLF Ln 3, Col 32
1°C Clear
Поиск
РУС
22:41 11.11.2025


pgAdmin 4
File Object Tools Edit View Window Help
Welcome airport_db/postgre... Sai_db/postgres@PostgreSQL 17*
Sai_db/postgres@PostgreSQL 17
No limit
Query Query History
DROP INDEX idx_passengers_passport;
DROP INDEX idx_passengers_fullinfo;
Data Output Messages Notifications
DROP INDEX
Query returned successfully in 116 msec.
Query returned successfully in 116 msec.
Total rows: Query complete 00:00:00.116
CRLF Ln 2, Col 36
1°C Clear
Поиск
РУС
22:42 11.11.2025