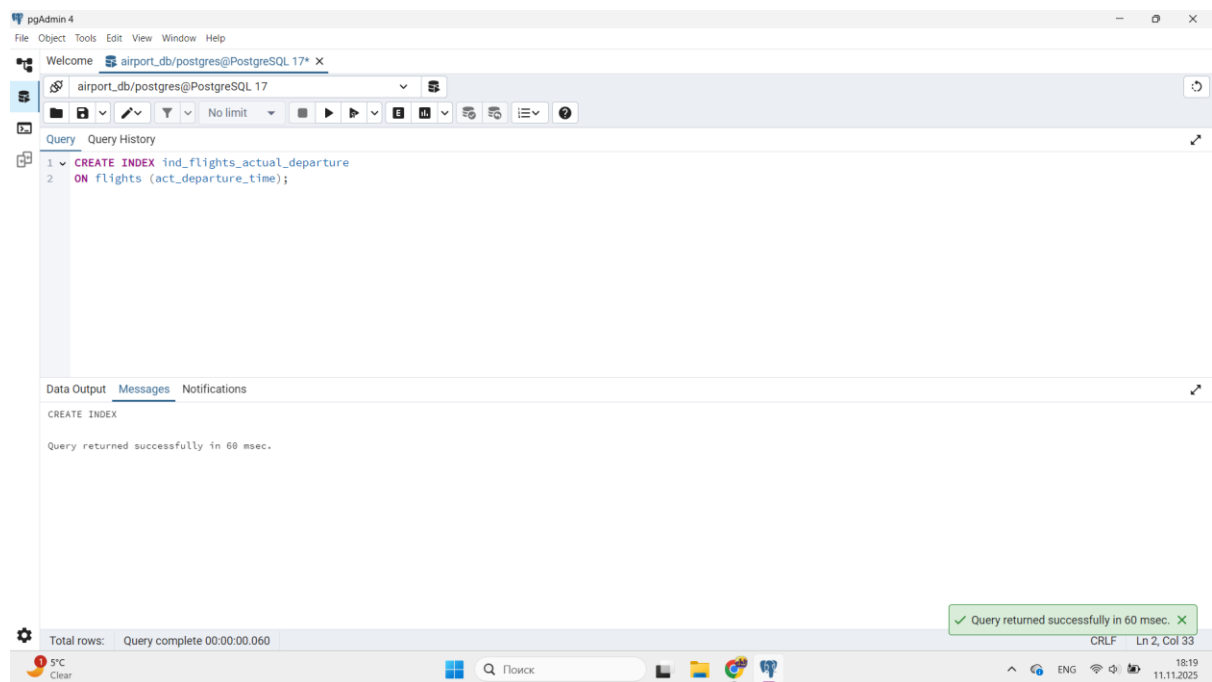
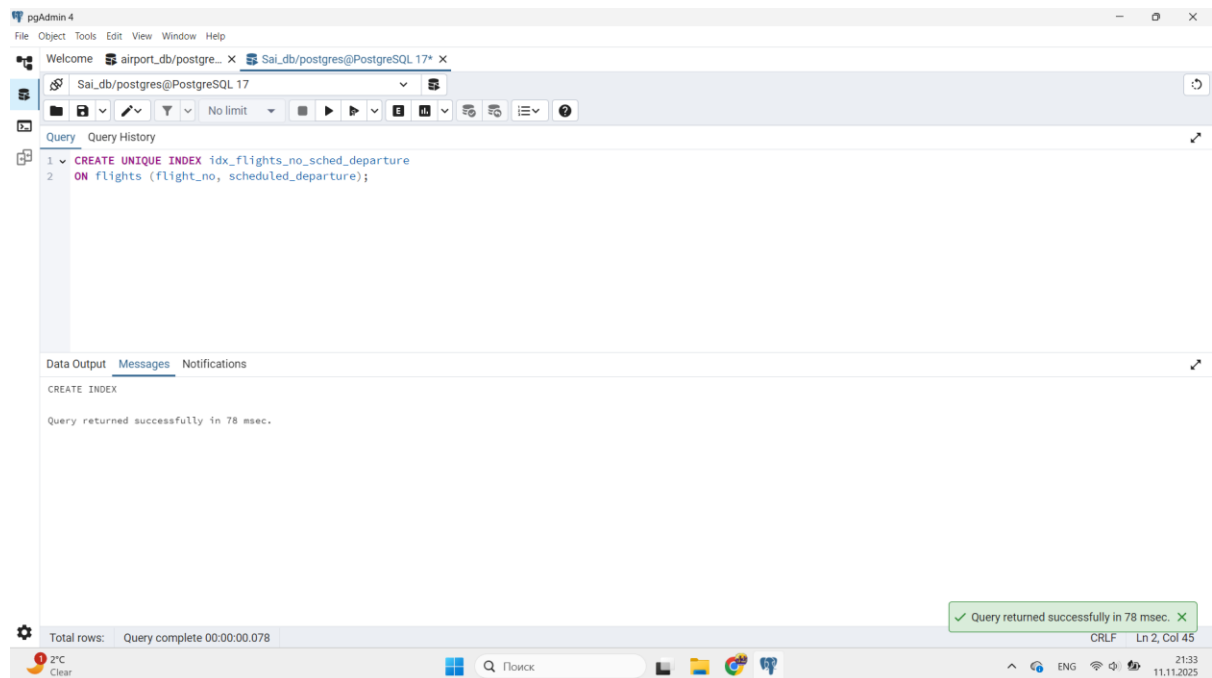


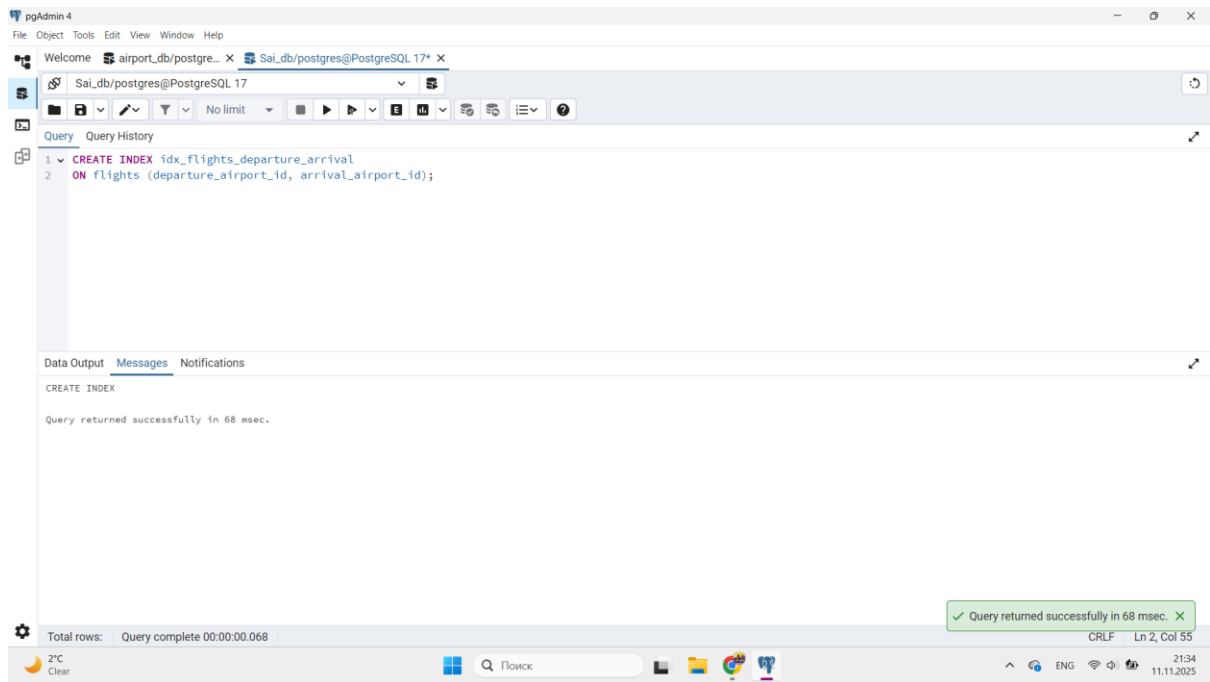
1. Create an index on the actual_departure column in the flights table.



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... X Sai_db/postgres@PostgreSQL 17* X

Sai_db/postgres@PostgreSQL 17

Query Query History

```
1 SET enable_seqscan = ON;
2 SET enable_indexscan = OFF;
3
4 EXPLAIN ANALYZE
5 SELECT *
6 FROM flights
7 WHERE departure_airport_id = 3
8 AND arrival_airport_id = 7;
```

Data Output Messages Notifications

Showing rows: 1 to 5 Page No: 1 of 1

QUERY PLAN
Seq Scan on flights (cost=0.00..27.89 rows=2 width=61) (actual time=0.164..0.309 rows=1 loops=1)
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

Total rows: 5 Query complete 00:00:00.084

Successfully run. Total query runtime: 84 msec. 5 rows affected.

pgAdmin 4

File Object Tools Edit View Window Help

Welcome airport_db/postgre... X Sai_db/postgres@PostgreSQL 17* X

Sai_db/postgres@PostgreSQL 17

Query Query History

```
1 CREATE INDEX idx_flights_departure_arrival
2 ON flights (departure_airport_id, arrival_airport_id);
```

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 137 msec.

Total rows: Query complete 00:00:00.137

Query returned successfully in 137 msec.

pgAdmin 4

Welcome | airport_db/postgre... X | Sai_db/postgres@PostgreSQL 17* X

Sai_db/postgres@PostgreSQL 17

Query | Query History

```
1 SET enable_seqscan = OFF;
2 SET enable_indexscan = ON;
3
4 EXPLAIN ANALYZE
5 SELECT *
6 FROM flights
7 WHERE departure_airport_id = 3
8 AND arrival_airport_id = 7;
```

Data Output | Messages | Notifications

Showing rows: 1 to 7 | Page No: 1 of 1

QUERY PLAN	
text	
1	Bitmap Heap Scan on flights (cost=4.30..9.97 rows=2 width=61) (actual time=0.097..0.098 rows=1 loops=1)
2	Recheck Cond: ((departure_airport_id = 3) AND (arrival_airport_id = 7))
3	Heap Blocks: exact=1
4	-> Bitmap Index Scan on idx_flights_departure_arrival (cost=0.00..4.29 rows=2 width=0) (actual time=0.075..0.075 rows=1 loop=1)
5	Index Cond: ((departure_airport_id = 3) AND (arrival_airport_id = 7))
6	Planning Time: 2.776 ms
7	Execution Time: 0.140 ms

Successfully run. Total query runtime: 82 msec. 7 rows affected.

Total rows: 7 | Query complete 00:00:00.082

2°C Clear | Поиск | 21:52 11.11.2025

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

pgAdmin 4

Welcome airport_db/postgre... Sai_db/postgres@PostgreSQL 17* X

Sai_db/postgres@PostgreSQL 17

Query Query History

```

1 SELECT indexname, indexdef
2 FROM pg_indexes
3 WHERE tablename = 'flights';

```

Data Output Messages Notifications

Indexname	Indexdef
flights_pkey	CREATE UNIQUE INDEX flights_pkey ON public.flights USING btree (flight_id)
idx_flights_no_scheduled_departure	CREATE UNIQUE INDEX idx_flights_no_scheduled_departure ON public.flights USING btree (flight_no, scheduled_dep...
idx_flights_departure_arrival	CREATE INDEX idx_flights_departure_arrival ON public.flights USING btree (departure_airport_id, arrival_airport_id)
idx_flights_actual_departure	CREATE INDEX idx_flights_actual_departure ON public.flights USING btree (actual_departure)

Total rows: 4 Query complete 00:00:00.109

Successfully run. Total query runtime: 109 msec. 4 rows affected.

pgAdmin 4

Welcome airport_db/postgre... Sai_db/postgres@PostgreSQL 17* X

Sai_db/postgres@PostgreSQL 17

Query Query History

```

1 EXPLAIN ANALYZE
2 SELECT *
3 FROM flights
4 WHERE departure_airport_id = 3
5 AND arrival_airport_id = 7;

```

Data Output Messages Notifications

QUERY PLAN

1	Bitmap Heap Scan on flights (cost=4.30..9.97 rows=2 width=61) (actual time=0.040..0.041 rows=1 loops=1)
2	Recheck Cond: ((departure_airport_id = 3) AND (arrival_airport_id = 7))
3	Heap Blocks: exact=1
4	-> Bitmap Index Scan on idx_flights_departure_arrival (cost=0.00..4.29 rows=2 width=0) (actual time=0.025..0.026 rows=1 loop=1)
5	Index Cond: ((departure_airport_id = 3) AND (arrival_airport_id = 7))
6	Planning Time: 0.200 ms
7	Execution Time: 0.098 ms

Total rows: 7 Query complete 00:00:00.108

Successfully run. Total query runtime: 108 msec. 7 rows affected.

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... X Sai_db/postgres@PostgreSQL 17* X

Sai_db/postgres@PostgreSQL 17

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.

Total rows: Query complete 00:00:00.116

2°C Clear

Поиск

22:09 11.11.2025

pgAdmin 4

File Object Tools Edit View Window Help

Welcome airport_db/postgre... X Sai_db/postgres@PostgreSQL 17* X

Sai_db/postgres@PostgreSQL 17

Query Query History

```
1 SELECT indexname, indexdef
2 FROM pg_indexes
3 WHERE tablename = 'passengers';
```

Data Output Messages Notifications

	indexname	indexdef
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)

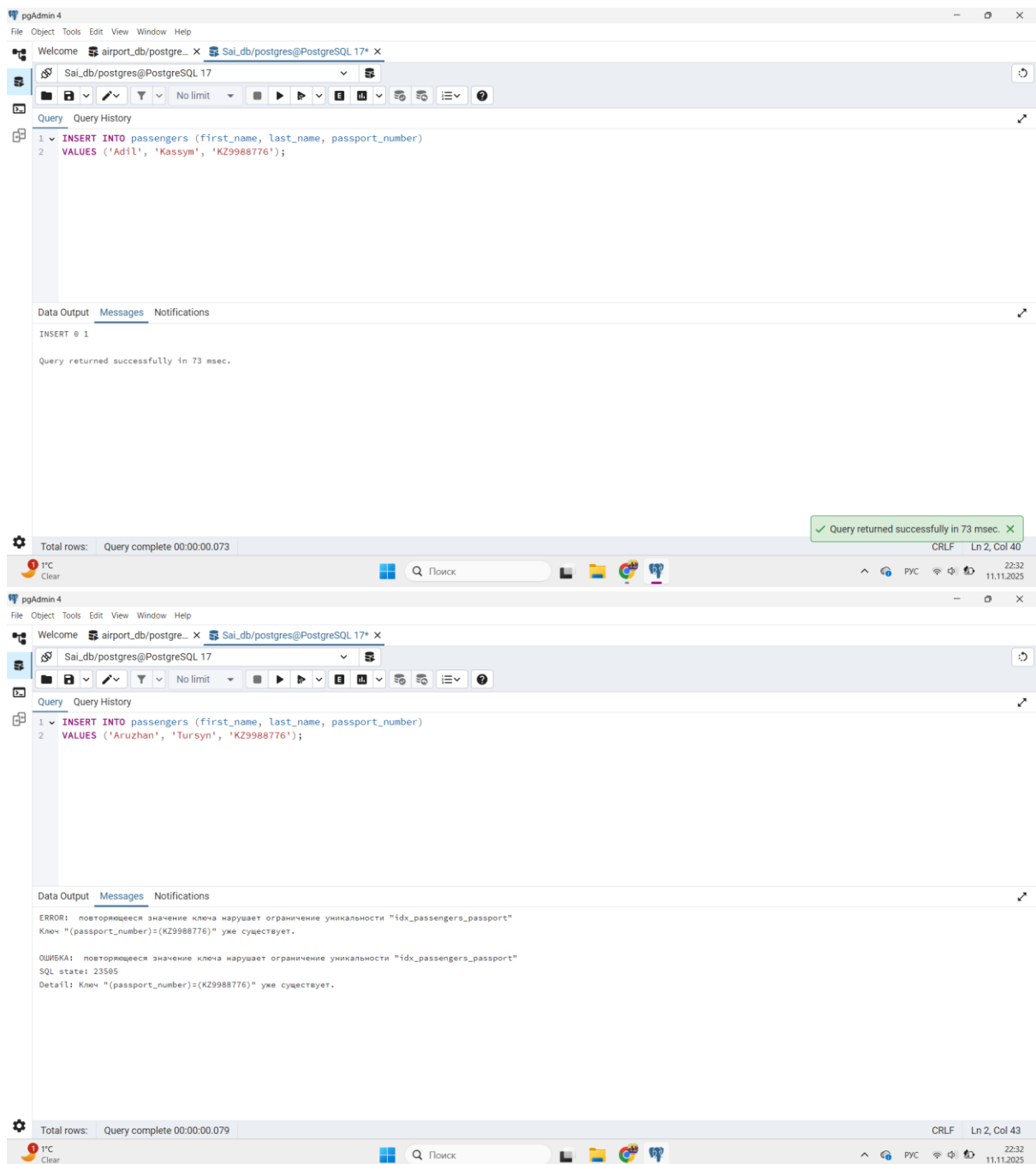
Showing rows: 1 to 2 Page No: 1 of 1

Total rows: 2 Query complete 00:00:00.130

1°C Clear

Поиск

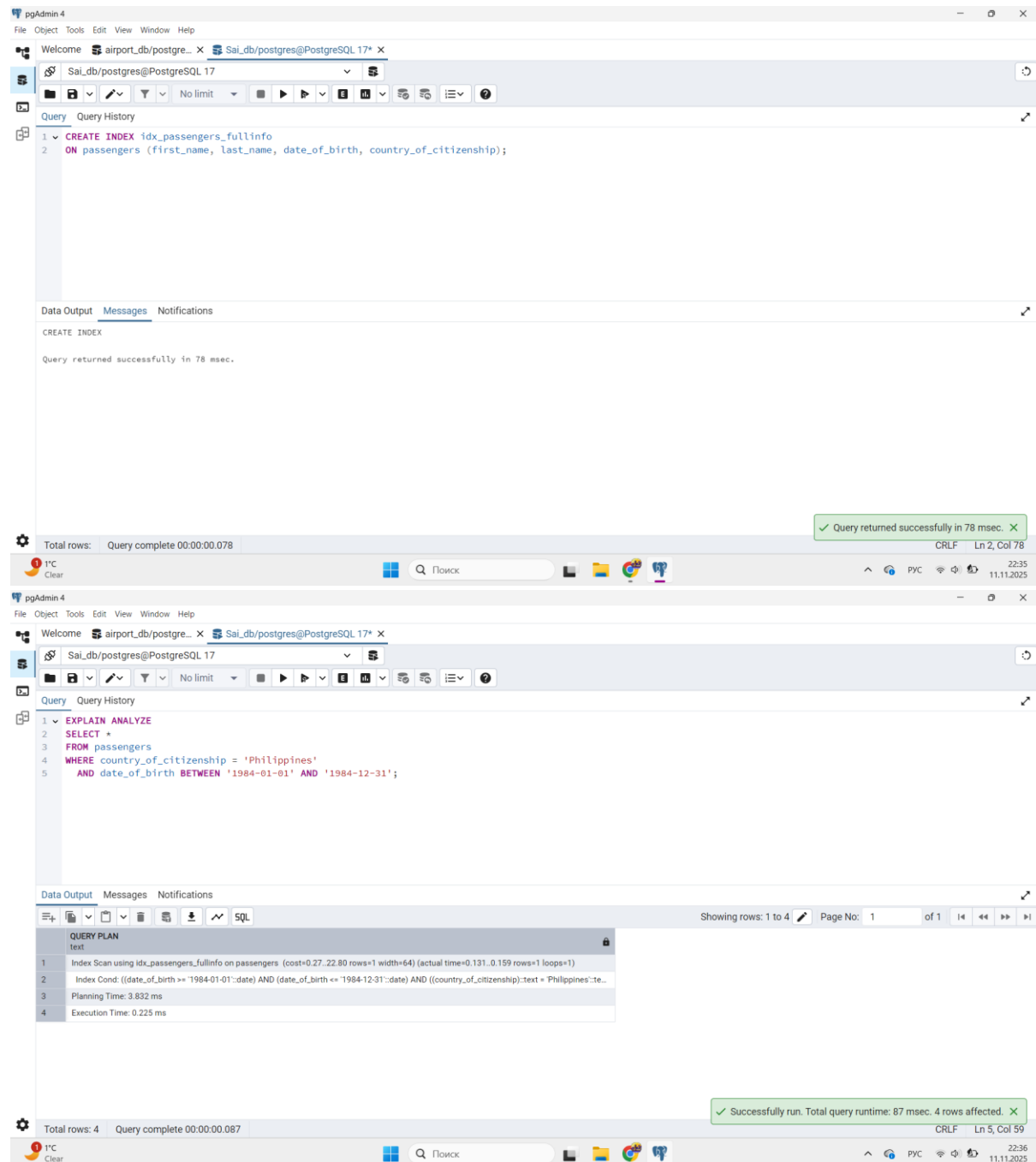
22:10 11.11.2025



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.



The first screenshot shows the pgAdmin 4 interface with the following SQL query entered in the query editor:

```
1 CREATE INDEX idx_passengers_fullinfo
2 ON passengers (first_name, last_name, date_of_birth, country_of_citizenship);
```

The Data Output tab shows the message: "CREATE INDEX" and "Query returned successfully in 78 msec."

The second screenshot shows the same pgAdmin 4 interface with the following SQL query entered:

```
1 EXPLAIN ANALYZE
2 SELECT *
3 FROM passengers
4 WHERE country_of_citizenship = 'Philippines'
5 AND date_of_birth BETWEEN '1984-01-01' AND '1984-12-31';
```

The Data Output tab shows the message: "Successfully run. Total query runtime: 87 msec. 4 rows affected."

The Query Plan tab shows the following execution plan:

Step	Plan
1	Index Scan using idx_passengers_fullinfo on passengers (cost=0.27..22.80 rows=1 width=64) (actual time=0.131..0.159 rows=1 loops=1)
2	Index Cond: ((date_of_birth >= '1984-01-01':date) AND (date_of_birth <= '1984-12-31':date) AND ((country_of_citizenship)::text = 'Philippines':te...
3	Planning Time: 3.832 ms
4	Execution Time: 0.225 ms

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

Welcome | airport_db/postgre... X | Sai_db/postgres@PostgreSQL 17* X

Sai_db/postgres@PostgreSQL 17

Query | Query History

```
1 SELECT indexname, indexdef
2 FROM pg_indexes
3 WHERE tablename = 'passengers';
```

Data Output | Messages | Notifications

	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)

Showing rows: 1 to 3 | Page No: 1 of 1

✓ Successfully run. Total query runtime: 155 msec. 3 rows affected. ✕

Total rows: 3 | Query complete 00:00:00.155

1°C Clear | Поиск | 22:41 11.11.2025

pgAdmin 4

Welcome | airport_db/postgre... X | Sai_db/postgres@PostgreSQL 17* X

Sai_db/postgres@PostgreSQL 17

Query | Query History

```
1 DROP INDEX idx_passengers_passport;
2 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

1°C Clear | Поиск | 22:42 11.11.2025