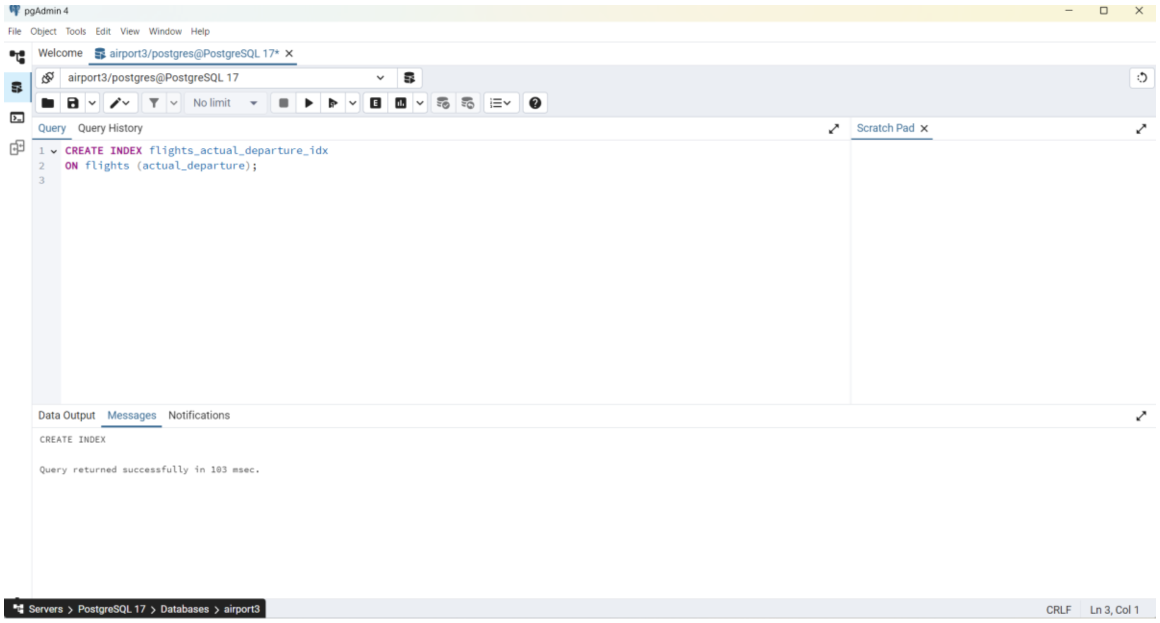
Laboratory Work 7

1. Create an index on actual_departure in flights

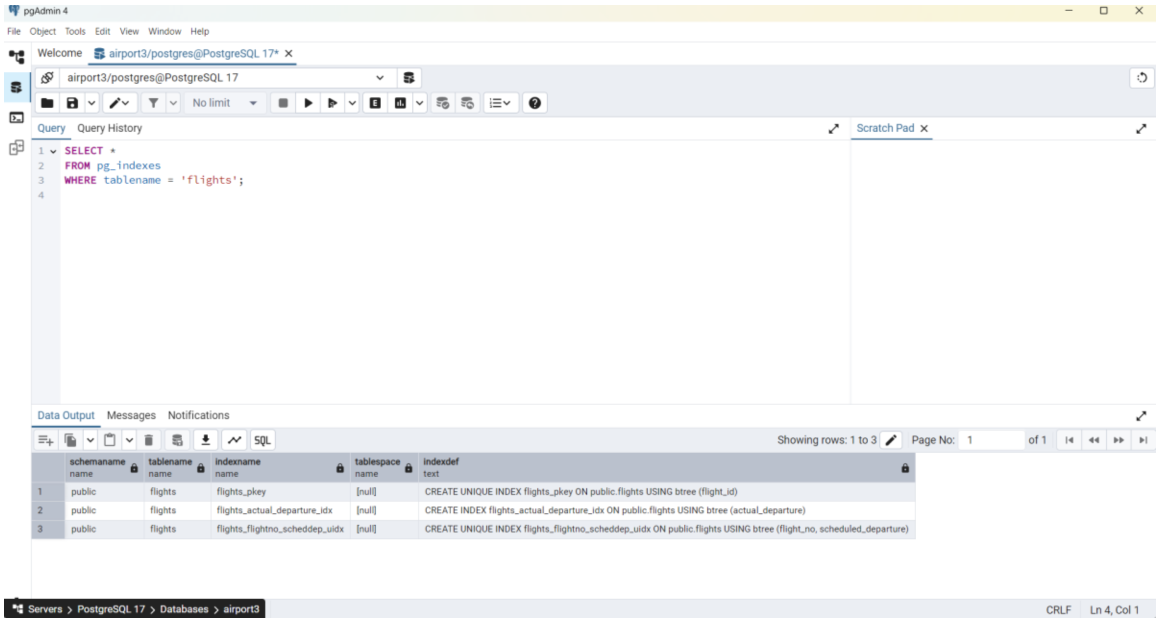


```sql
CREATE INDEX flights_actual_departure_idx
ON flights (actual_departure);
```

```
CREATE INDEX

Query returned successfully in 103 msec.
```

2. Create unique index flight_no+scheduled_departure

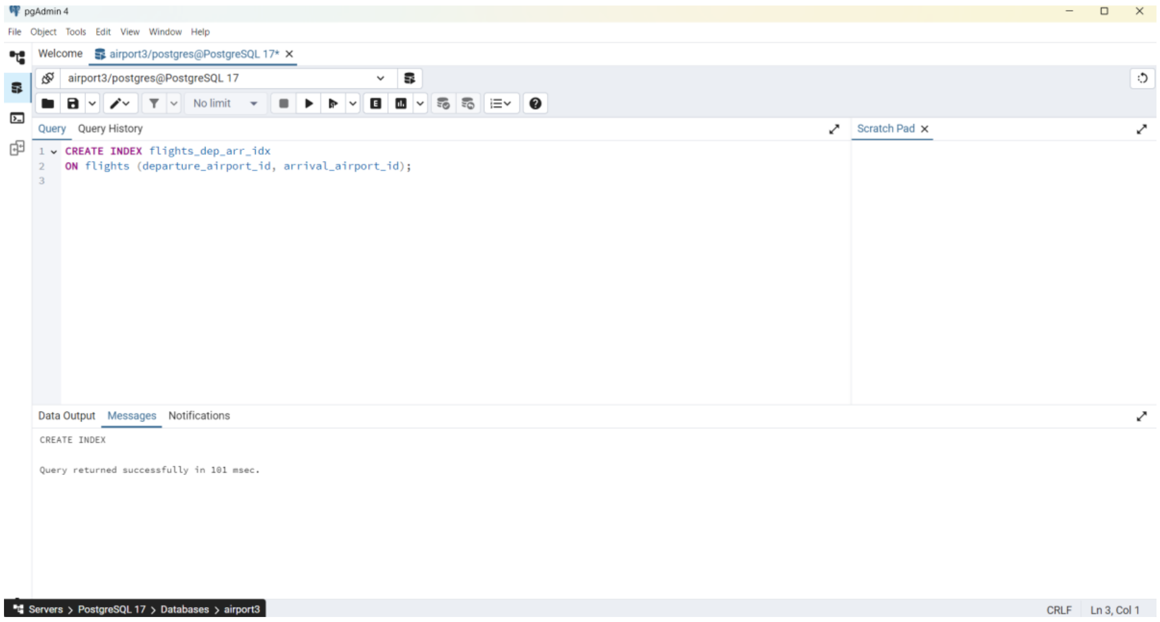


```sql
SELECT *
FROM pg_indexes
WHERE tablename = 'flights';
```

Showing rows: 1 to 3

	schemaname name	tablename name	indexname name	tablespace name	indexdef text
1	public	flights	flights_pkey	[null]	CREATE UNIQUE INDEX flights_pkey ON public.flights USING btree (flight_id)
2	public	flights	flights_actual_departure_idx	[null]	CREATE INDEX flights_actual_departure_idx ON public.flights USING btree (actual_departure)
3	public	flights	flights_flightno_scheddep_uidx	[null]	CREATE UNIQUE INDEX flights_flightno_scheddep_uidx ON public.flights USING btree (flight_no, scheduled_departure)

3. Create composite index departure_airport_id + arrival_airport_id



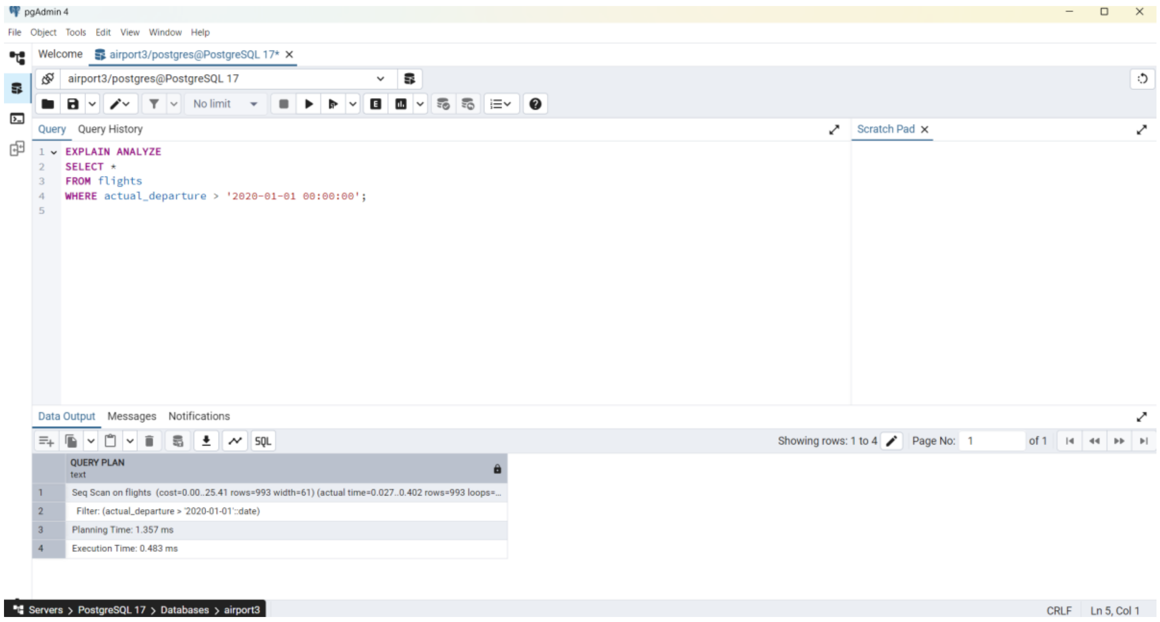
```sql
CREATE INDEX flights_dep_arr_idx
ON flights (departure_airport_id, arrival_airport_id);
```

```
CREATE INDEX

Query returned successfully in 101 msec.
```

4. Evaluate performance before/after indexes

BEFORE:

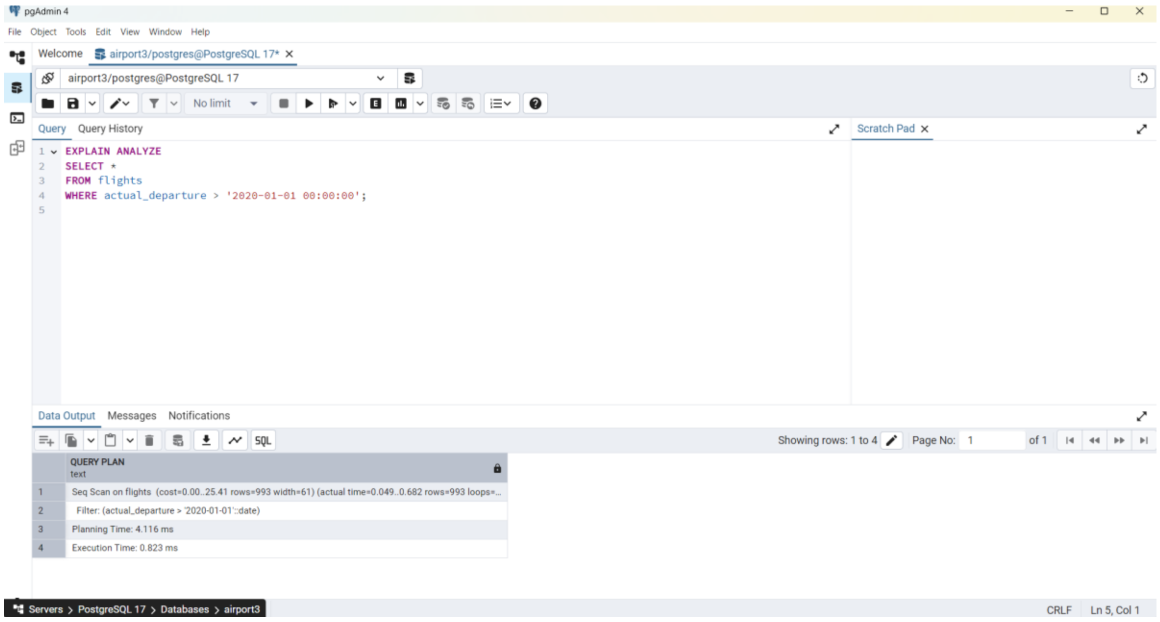


```sql
EXPLAIN ANALYZE
SELECT *
FROM flights
WHERE actual_departure > '2020-01-01 00:00:00';
```

Showing rows: 1 to 4 Page No: 1 of 1

	QUERY PLAN text
1	Seq Scan on flights (cost=0.00..25.41 rows=993 width=61) (actual time=0.027..0.402 rows=993 loops=...
2	Filter: (actual_departure > '2020-01-01'::date)
3	Planning Time: 1.357 ms
4	Execution Time: 0.483 ms

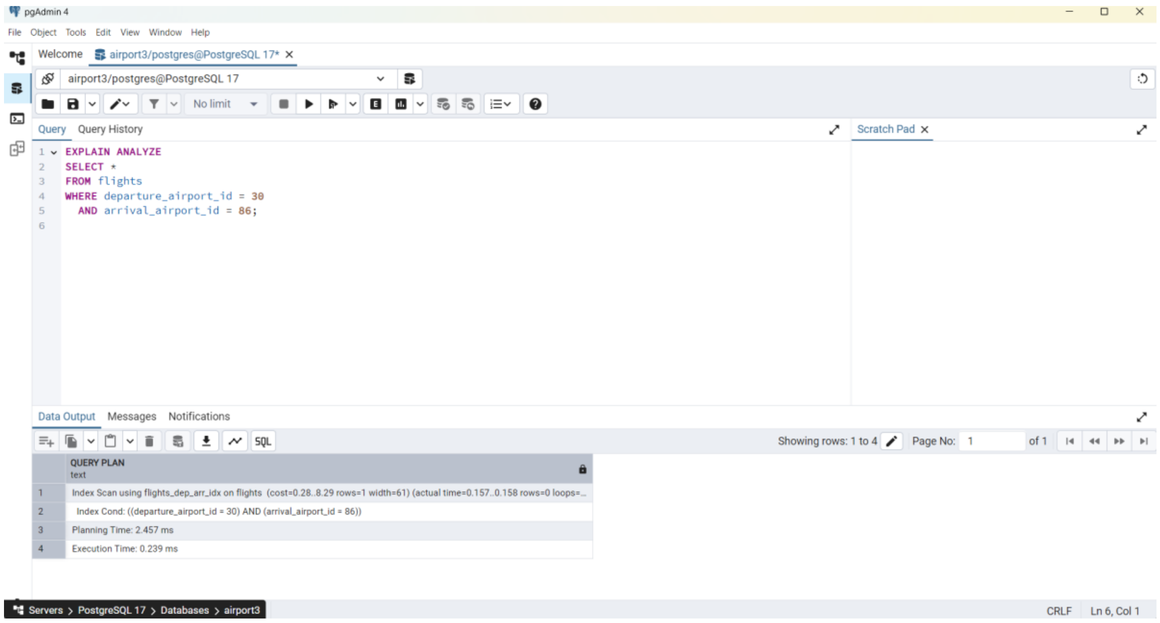
AFTER:



```
EXPLAIN ANALYZE
SELECT *
FROM flights
WHERE actual_departure > '2020-01-01 00:00:00';
```

	QUERY PLAN text
1	Seq Scan on flights (cost=0.00..25.41 rows=993 width=61) (actual time=0.049..0.682 rows=993 loops=...
2	Filter: (actual_departure > '2020-01-01'::date)
3	Planning Time: 4.116 ms
4	Execution Time: 0.823 ms

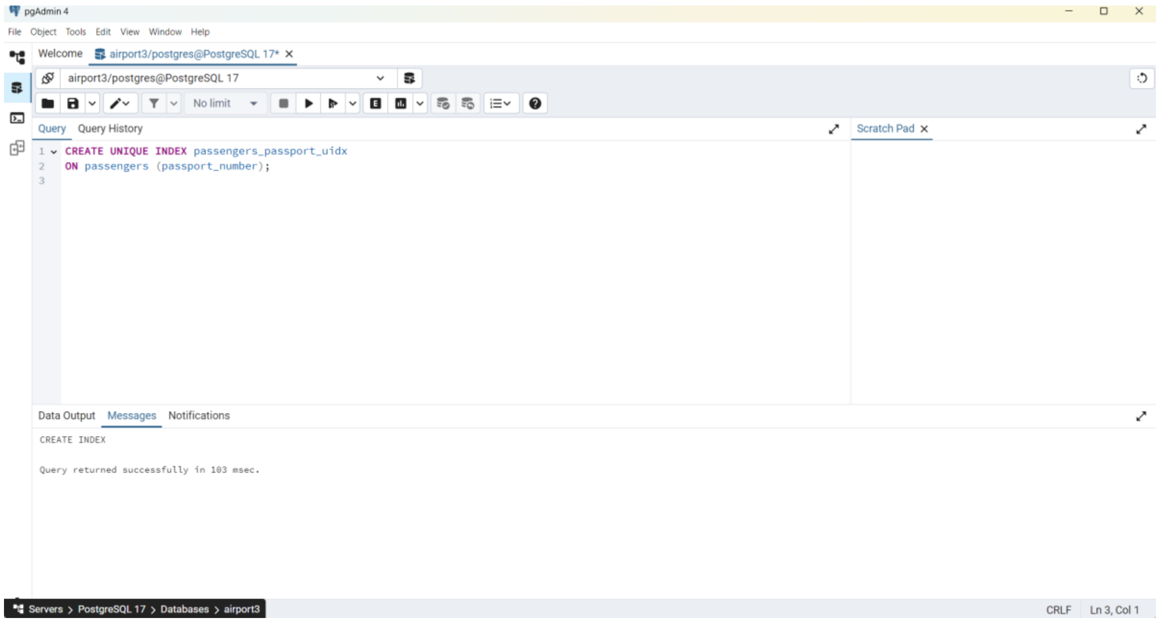
5. Use EXPLAIN ANALYZE for airport filters



```
EXPLAIN ANALYZE
SELECT *
FROM flights
WHERE departure_airport_id = 30
  AND arrival_airport_id = 86;
```

	QUERY PLAN text
1	Index Scan using flights_dep_arr_idx on flights (cost=0.28..8.29 rows=1 width=61) (actual time=0.157..0.158 rows=0 loops=...
2	Index Cond: ((departure_airport_id = 30) AND (arrival_airport_id = 86))
3	Planning Time: 2.457 ms
4	Execution Time: 0.239 ms

6. Unique index on passport_number + test inserts

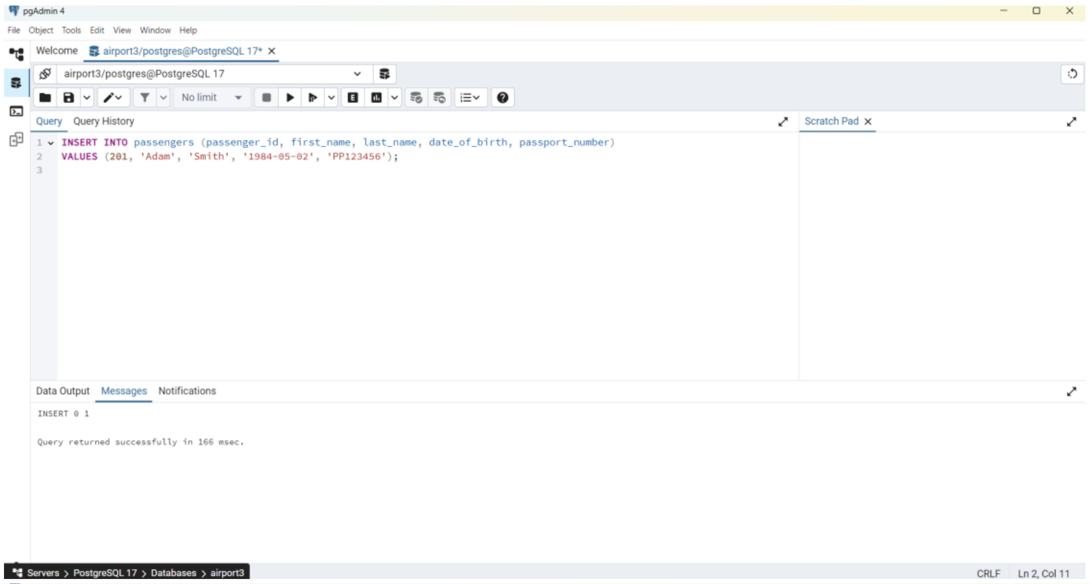


```
CREATE UNIQUE INDEX passengers_passport_uidx
ON passengers (passport_number);
```

```
CREATE INDEX

Query returned successfully in 103 msec.
```

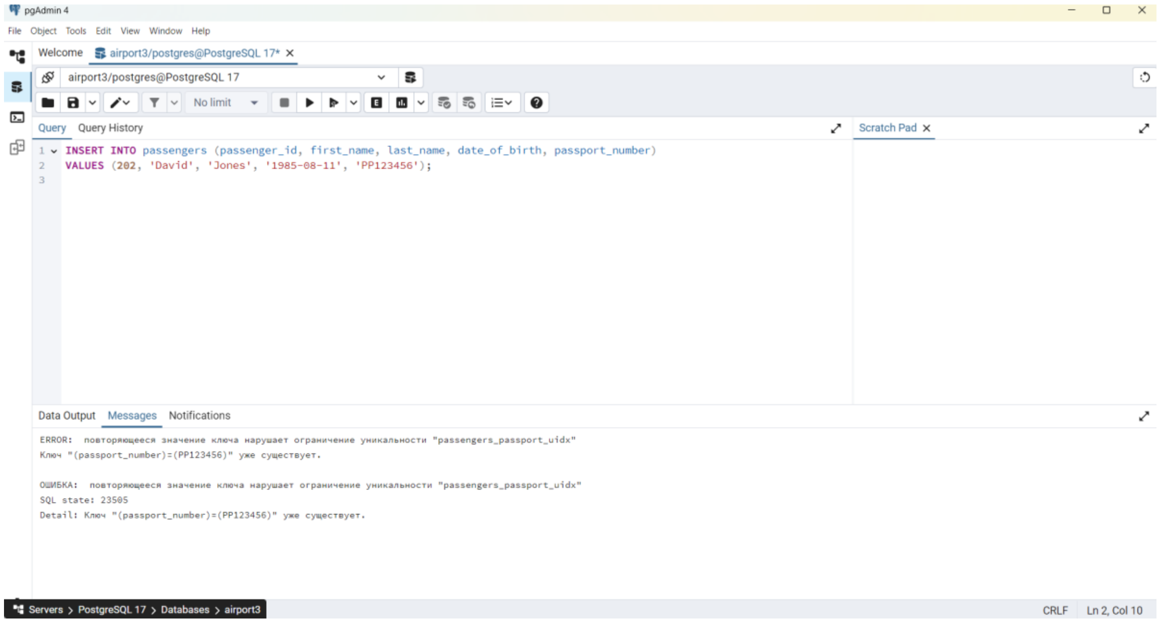
pgAdmin 4
File Object Tools Edit View Window Help
Welcome
airport3/postgres@PostgreSQL 17*
airport3/postgres@PostgreSQL 17
No limit
Query
Query History
Scratch Pad
Data Output
Messages
Notifications
Servers > PostgreSQL 17 > Databases > airport3
CRLF
Ln 3, Col 1



```sql
INSERT INTO passengers (passenger_id, first_name, last_name, date_of_birth, passport_number)
VALUES (201, 'Adam', 'Smith', '1984-05-02', 'PP123456');
```

```
INSERT 0 1

Query returned successfully in 166 msec.
```

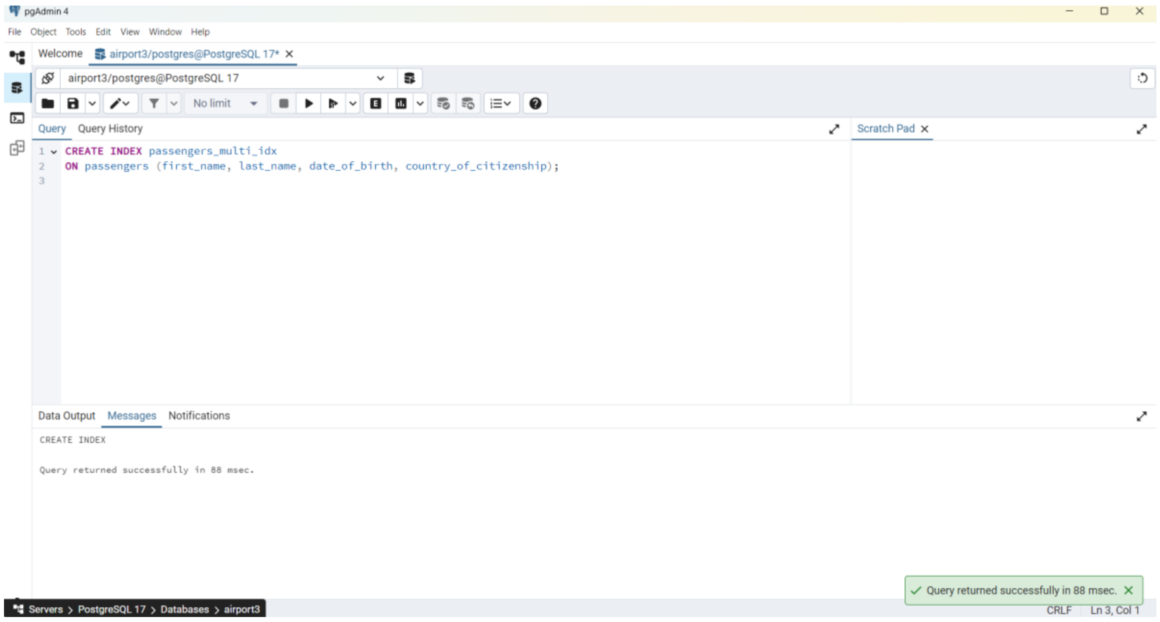


```sql
INSERT INTO passengers (passenger_id, first_name, last_name, date_of_birth, passport_number)
VALUES (202, 'David', 'Jones', '1985-08-11', 'PP123456');
```

```
ERROR:  повторяющееся значение ключа нарушает ограничение уникальности "passengers_passport_uidx"
Ключ "(passport_number)=(PP123456)" уже существует.

ОШИБКА:  повторяющееся значение ключа нарушает ограничение уникальности "passengers_passport_uidx"
SQL state: 23505
Detail: Ключ "(passport_number)=(PP123456)" уже существует.
```

7. Composite index on passenger fields + EXPLAIN ANALYZE

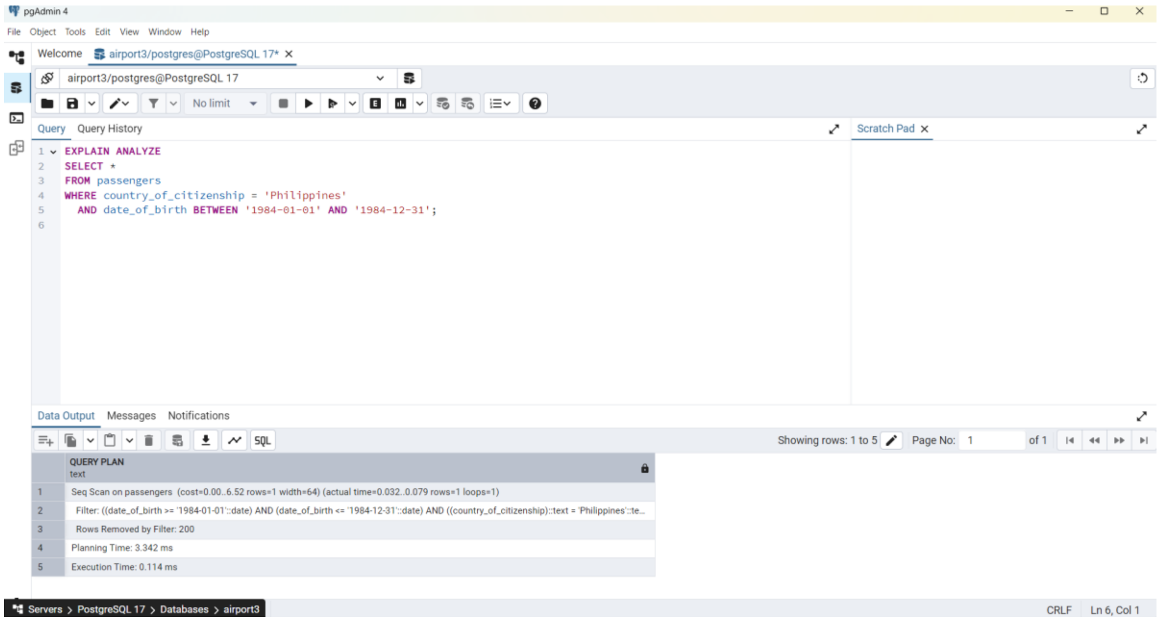


```sql
CREATE INDEX passengers_multi_idx
ON passengers (first_name, last_name, date_of_birth, country_of_citizenship);
```

```
CREATE INDEX

Query returned successfully in 88 msec.
```

pgAdmin 4
File Object Tools Edit View Window Help
Welcome
airport3/postgres@PostgreSQL 17*
No limit
Query
Query History
Scratch Pad
Data Output
Messages
Notifications
Query returned successfully in 88 msec.
Servers > PostgreSQL 17 > Databases > airport3
CRLF
Ln 3, Col 1

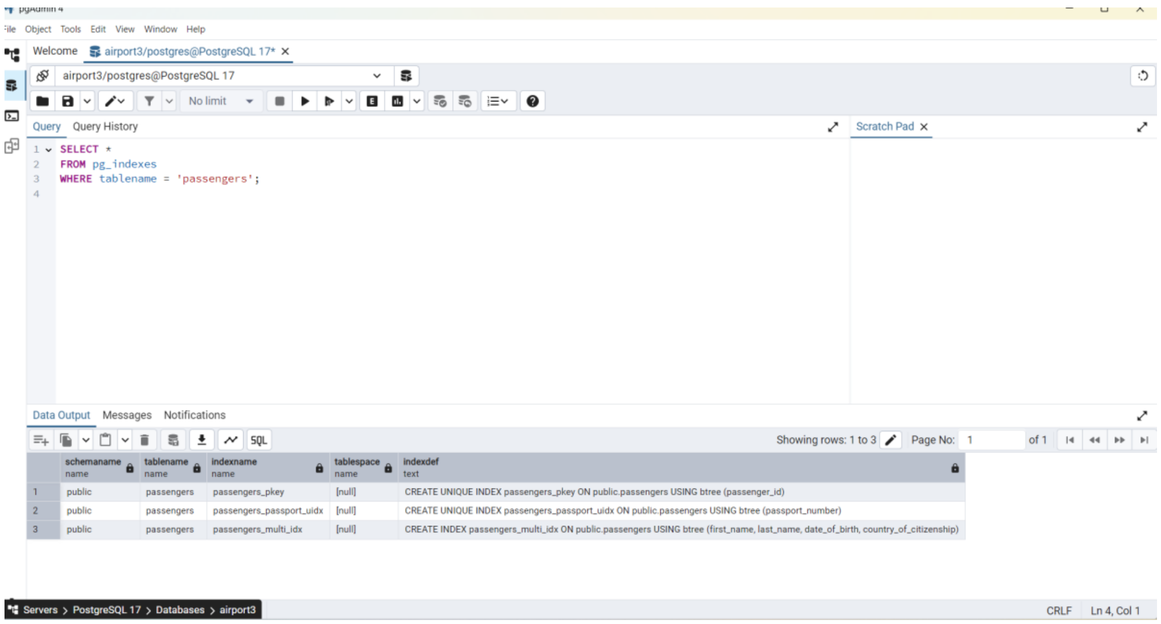


```sql
EXPLAIN ANALYZE
SELECT *
FROM passengers
WHERE country_of_citizenship = 'Philippines'
  AND date_of_birth BETWEEN '1984-01-01' AND '1984-12-31';
```

	QUERY PLAN text
1	Seq Scan on passengers (cost=0.00..6.52 rows=1 width=64) (actual time=0.032..0.079 rows=1 loops=1)
2	Filter: ((date_of_birth >= '1984-01-01'::date) AND (date_of_birth <= '1984-12-31'::date) AND ((country_of_citizenship)::text = 'Philippines'::te...
3	Rows Removed by Filter: 200
4	Planning Time: 3.342 ms
5	Execution Time: 0.114 ms

PostgreSQL does not use the composite index because the query filters by the last columns of the index, not by the first ones, so it performs a sequential scan instead.

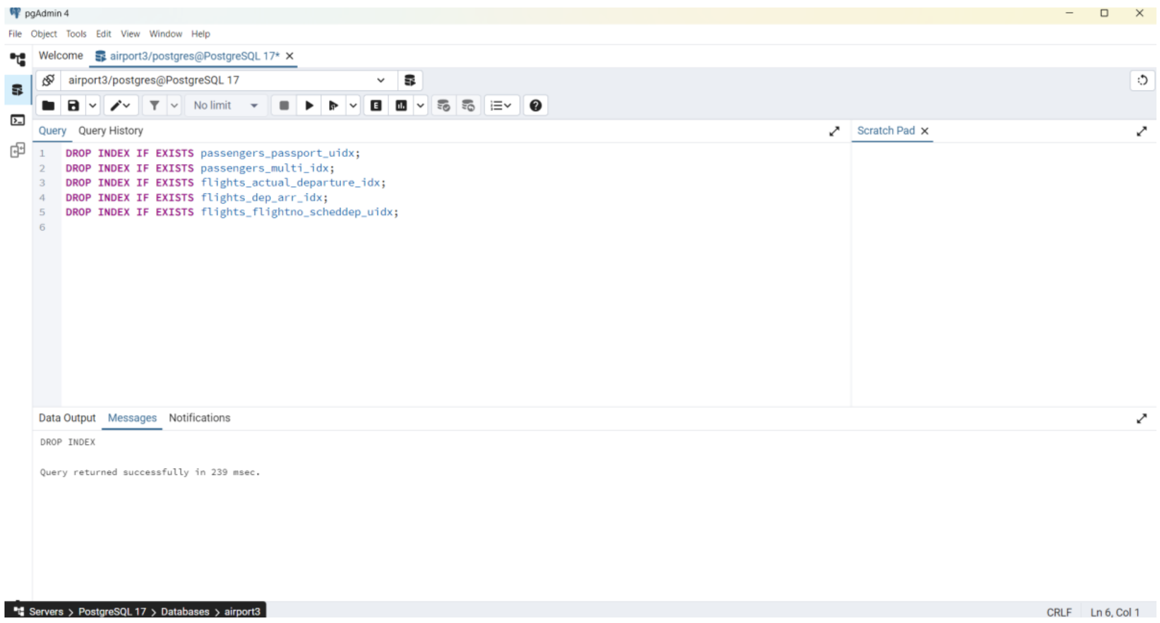
8. List indexes & then delete them



```sql
SELECT *
FROM pg_indexes
WHERE tablename = 'passengers';
```

Showing rows: 1 to 3

	schemaname name	tablename name	indexname name	tablespace name	indexdef text
1	public	passengers	passengers_pkey	[null]	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
2	public	passengers	passengers_passport_uidx	[null]	CREATE UNIQUE INDEX passengers_passport_uidx ON public.passengers USING btree (passport_number)
3	public	passengers	passengers_multi_idx	[null]	CREATE INDEX passengers_multi_idx ON public.passengers USING btree (first_name, last_name, date_of_birth, country_of_citizenship)



```sql
DROP INDEX IF EXISTS passengers_passport_uidx;
DROP INDEX IF EXISTS passengers_multi_idx;
DROP INDEX IF EXISTS flights_actual_departure_idx;
DROP INDEX IF EXISTS flights_dep_arr_idx;
DROP INDEX IF EXISTS flights_flightno_scheddep_uidx;
```

```
DROP INDEX

Query returned successfully in 239 msec.
```