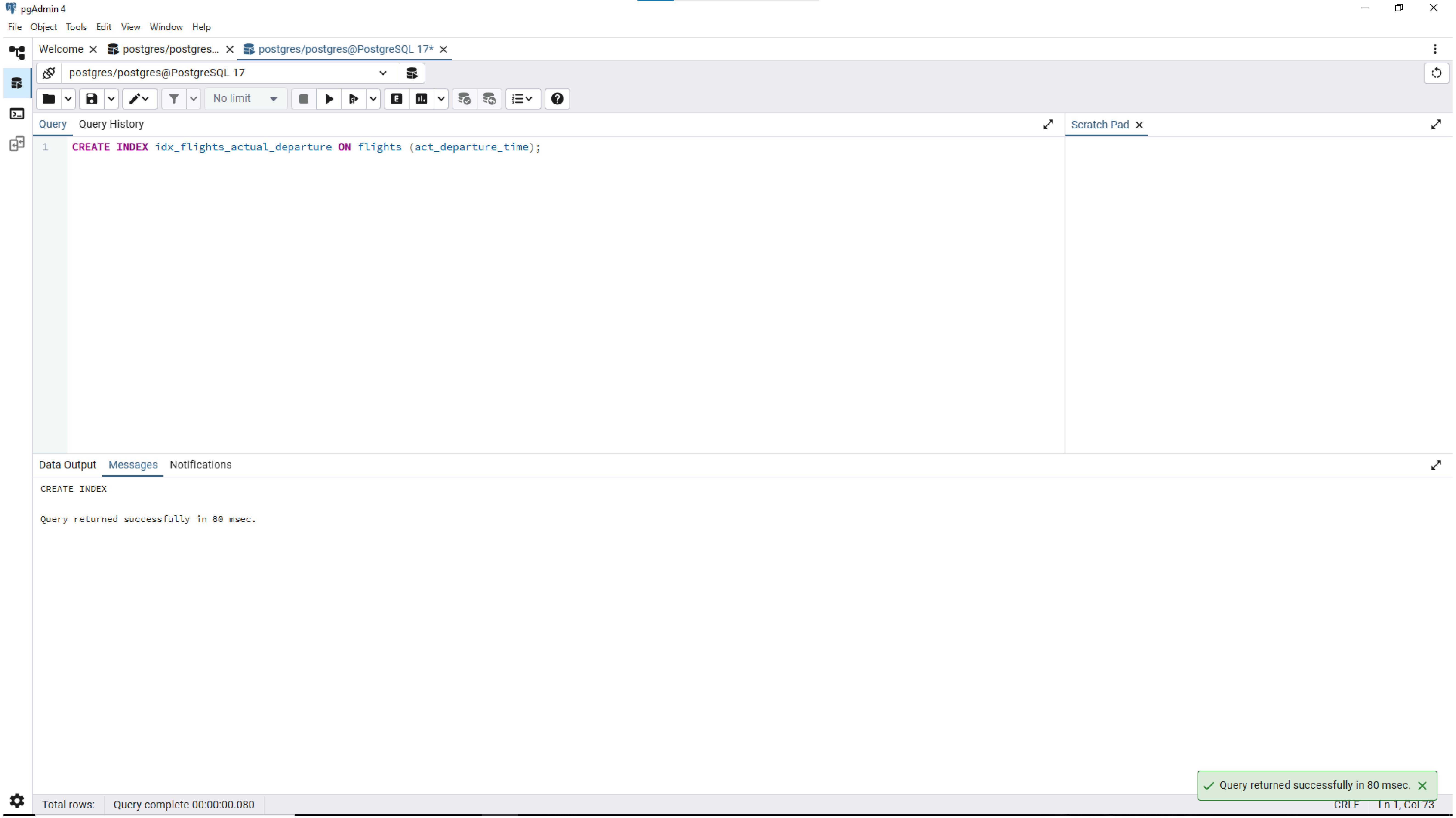


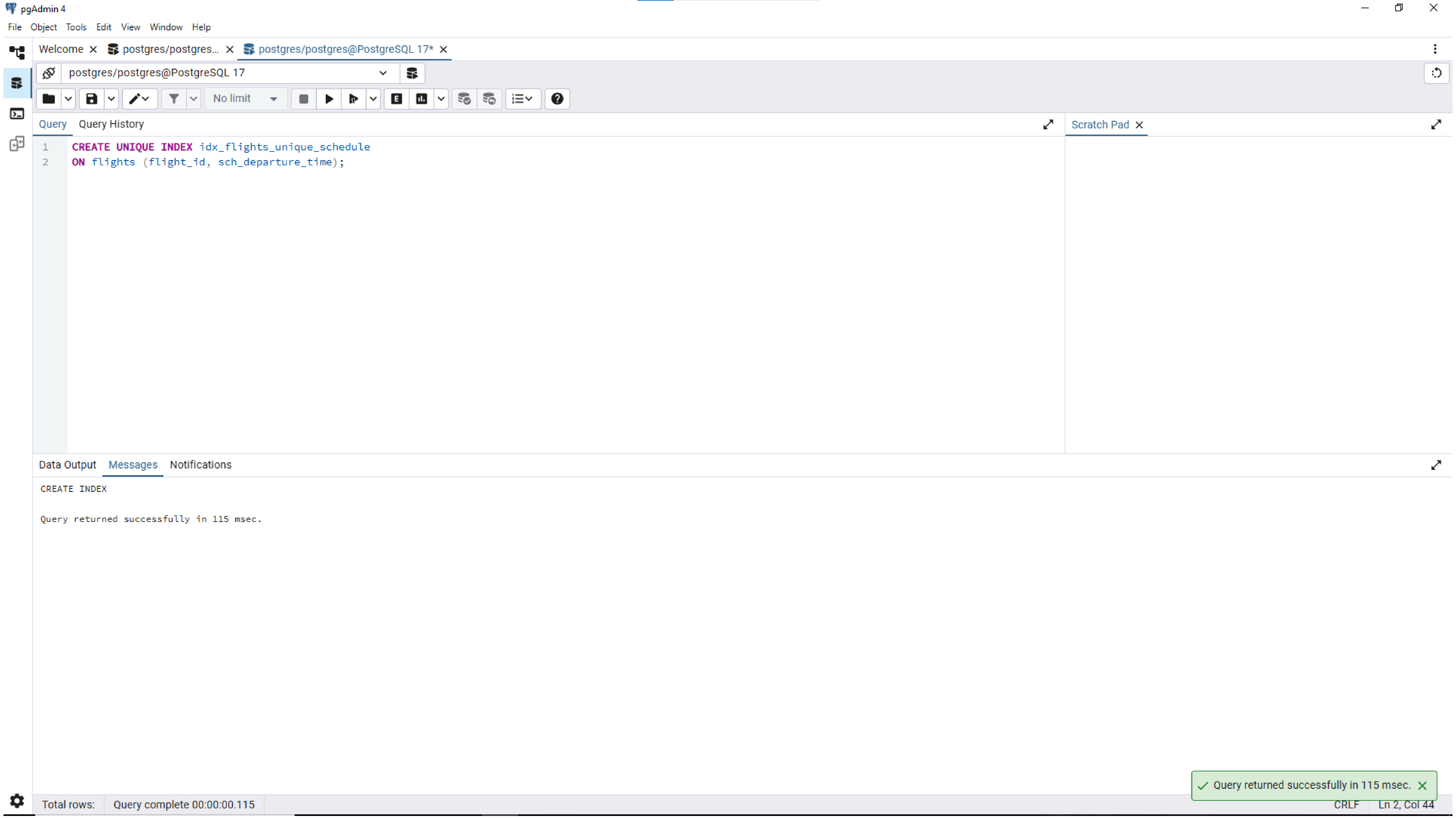
```
CREATE INDEX idx_flights_actual_departure ON flights (act_departure_time);
```

```
CREATE INDEX

Query returned successfully in 80 msec.
```

Query returned successfully in 80 msec.

Total rows: Query complete 00:00:00.080 CRLF Ln 1, Col 73

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



```sql
CREATE UNIQUE INDEX idx_flights_unique_schedule
ON flights (flight_id, sch_departure_time);
```

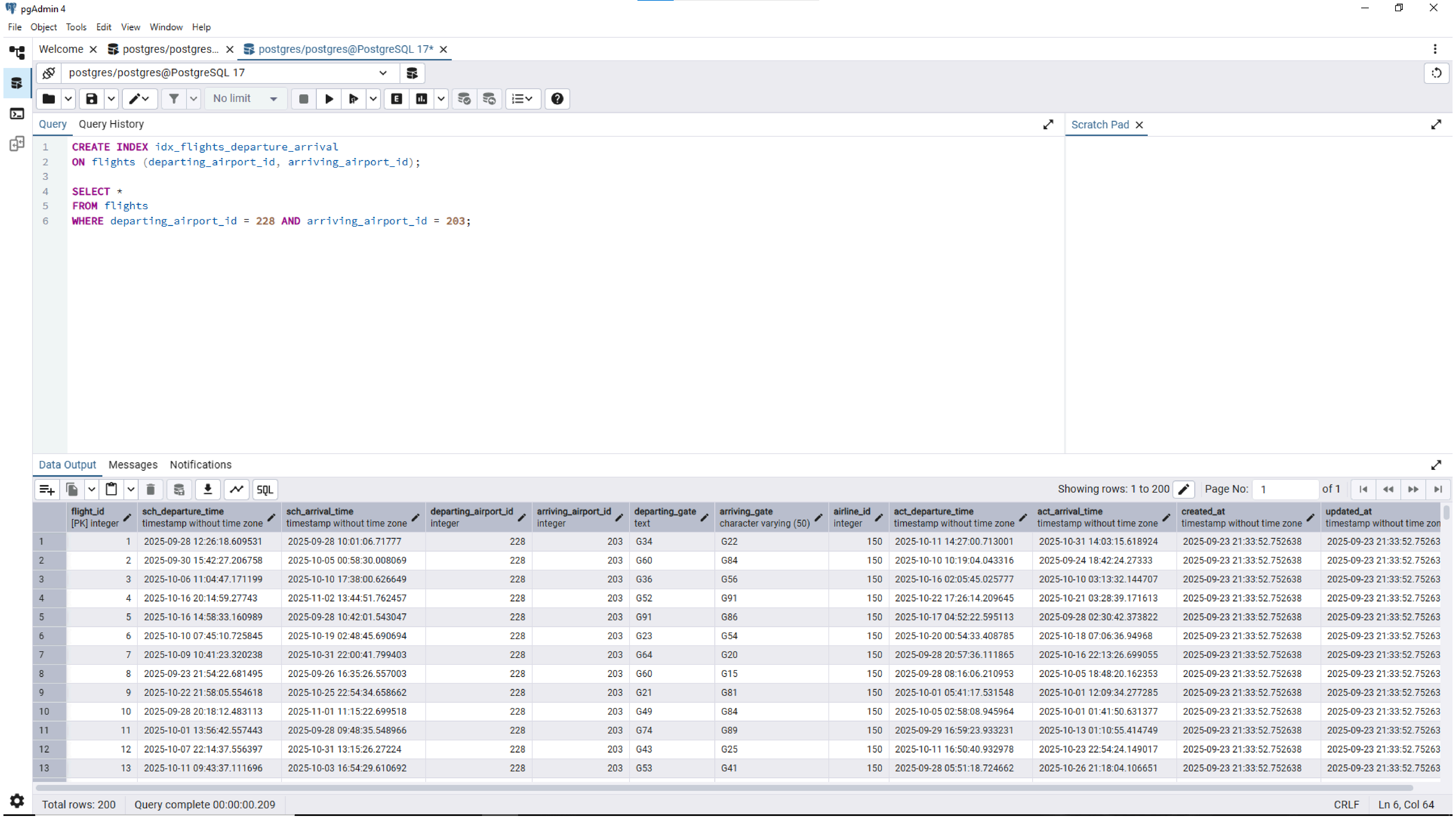
```
CREATE INDEX

Query returned successfully in 115 msec.
```

Query returned successfully in 115 msec.

Total rows: Query complete 00:00:00.115

CRLF Ln 2, Col 44

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



```sql
CREATE INDEX idx_flights_departure_arrival
ON flights (departing_airport_id, arriving_airport_id);

SELECT *
FROM flights
WHERE departing_airport_id = 228 AND arriving_airport_id = 203;
```

Showing rows: 1 to 200 Page No: 1 of 1

	flight_id [PK] integer	sch_departure_time timestamp without time zone	sch_arrival_time timestamp without time zone	departing_airport_id integer	arriving_airport_id integer	departing_gate text	arriving_gate character varying (50)	airline_id integer	act_departure_time timestamp without time zone	act_arrival_time timestamp without time zone	created_at timestamp without time zone	updated_at timestamp without time zon
1	1	2025-09-28 12:26:18.609531	2025-09-28 10:01:06.71777	228	203	G34	G22	150	2025-10-11 14:27:00.713001	2025-10-31 14:03:15.618924	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
2	2	2025-09-30 15:42:27.206758	2025-10-05 00:58:30.008069	228	203	G60	G84	150	2025-10-10 10:19:04.043316	2025-09-24 18:42:24.27333	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
3	3	2025-10-06 11:04:47.171199	2025-10-10 17:38:00.626649	228	203	G36	G56	150	2025-10-16 02:05:45.025777	2025-10-10 03:13:32.144707	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
4	4	2025-10-16 20:14:59.27743	2025-11-02 13:44:51.762457	228	203	G52	G91	150	2025-10-22 17:26:14.209645	2025-10-21 03:28:39.171613	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
5	5	2025-10-16 14:58:33.160989	2025-09-28 10:42:01.543047	228	203	G91	G86	150	2025-10-17 04:52:22.595113	2025-09-28 02:30:42.373822	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
6	6	2025-10-10 07:45:10.725845	2025-10-19 02:48:45.690694	228	203	G23	G54	150	2025-10-20 00:54:33.408785	2025-10-18 07:06:36.94968	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
7	7	2025-10-09 10:41:23.320238	2025-10-31 22:00:41.799403	228	203	G64	G20	150	2025-09-28 20:57:36.111865	2025-10-16 22:13:26.699055	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
8	8	2025-09-23 21:54:22.681495	2025-09-26 16:35:26.557003	228	203	G60	G15	150	2025-09-28 08:16:06.210953	2025-10-05 18:48:20.162353	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
9	9	2025-10-22 21:58:05.554618	2025-10-25 22:54:34.658662	228	203	G21	G81	150	2025-10-01 05:41:17.531548	2025-10-01 12:09:34.277285	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
10	10	2025-09-28 20:18:12.483113	2025-11-01 11:15:22.699518	228	203	G49	G84	150	2025-10-05 02:58:08.945964	2025-10-01 01:41:50.631377	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
11	11	2025-10-01 13:56:42.557443	2025-09-28 09:48:35.548966	228	203	G74	G89	150	2025-09-29 16:59:23.933231	2025-10-13 01:10:55.414749	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
12	12	2025-10-07 22:14:37.556397	2025-10-31 13:15:26.27224	228	203	G43	G25	150	2025-10-11 16:50:40.932978	2025-10-23 22:54:24.149017	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263
13	13	2025-10-11 09:43:37.111696	2025-10-03 16:54:29.610692	228	203	G53	G41	150	2025-09-28 05:51:18.724662	2025-10-26 21:18:04.106651	2025-09-23 21:33:52.752638	2025-09-23 21:33:52.75263

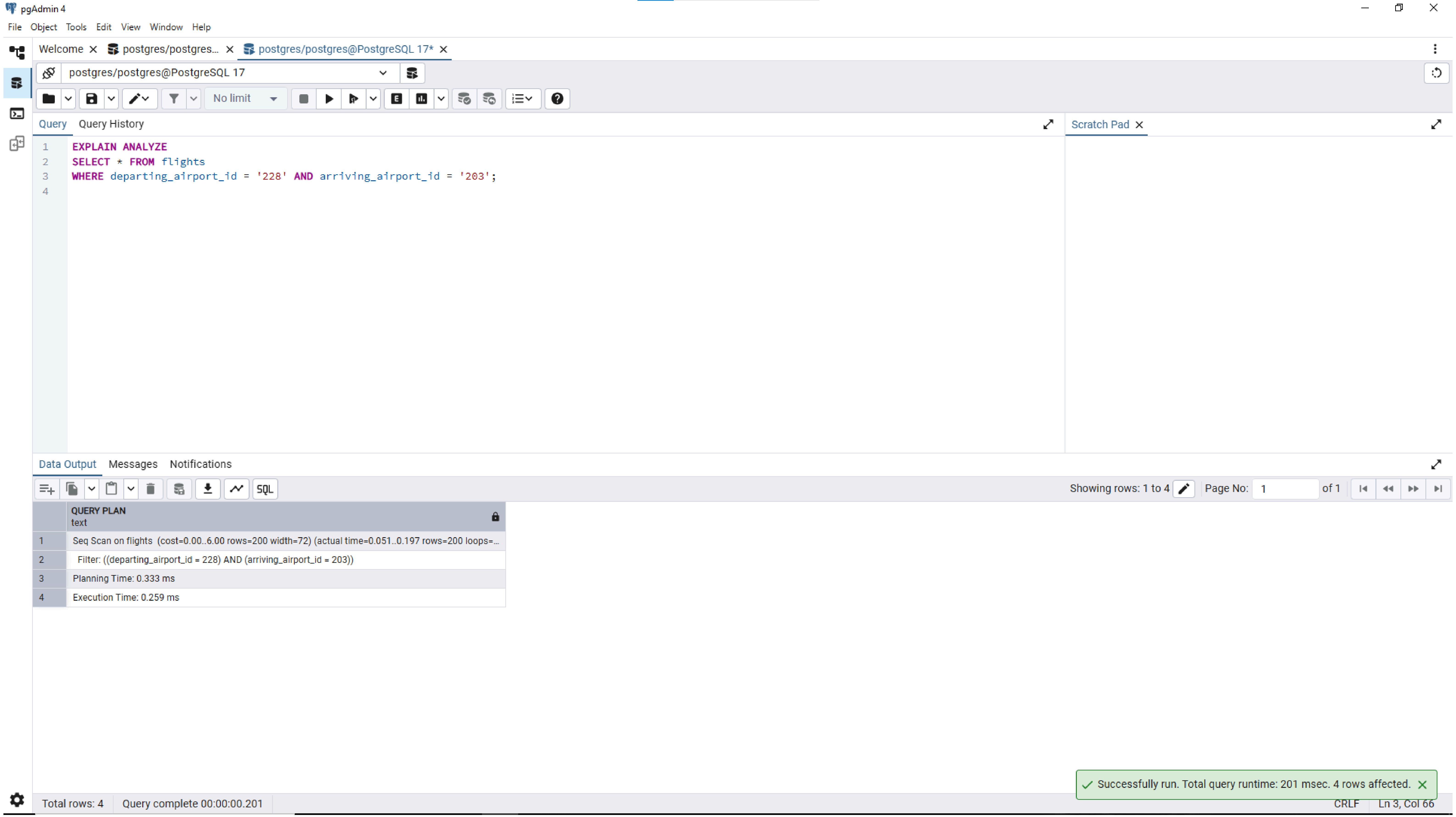
Total rows: 200 Query complete 00:00:00.209

CRLF Ln 6, Col 64

Create a composite index on the departure_airport_id and arrival_airport_id columns.

Exercices	Difference	with index	without Index
1	49	146 msec	195 msec
2	109	168 msec	277 msec
3	54	184 msec	238 msec

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



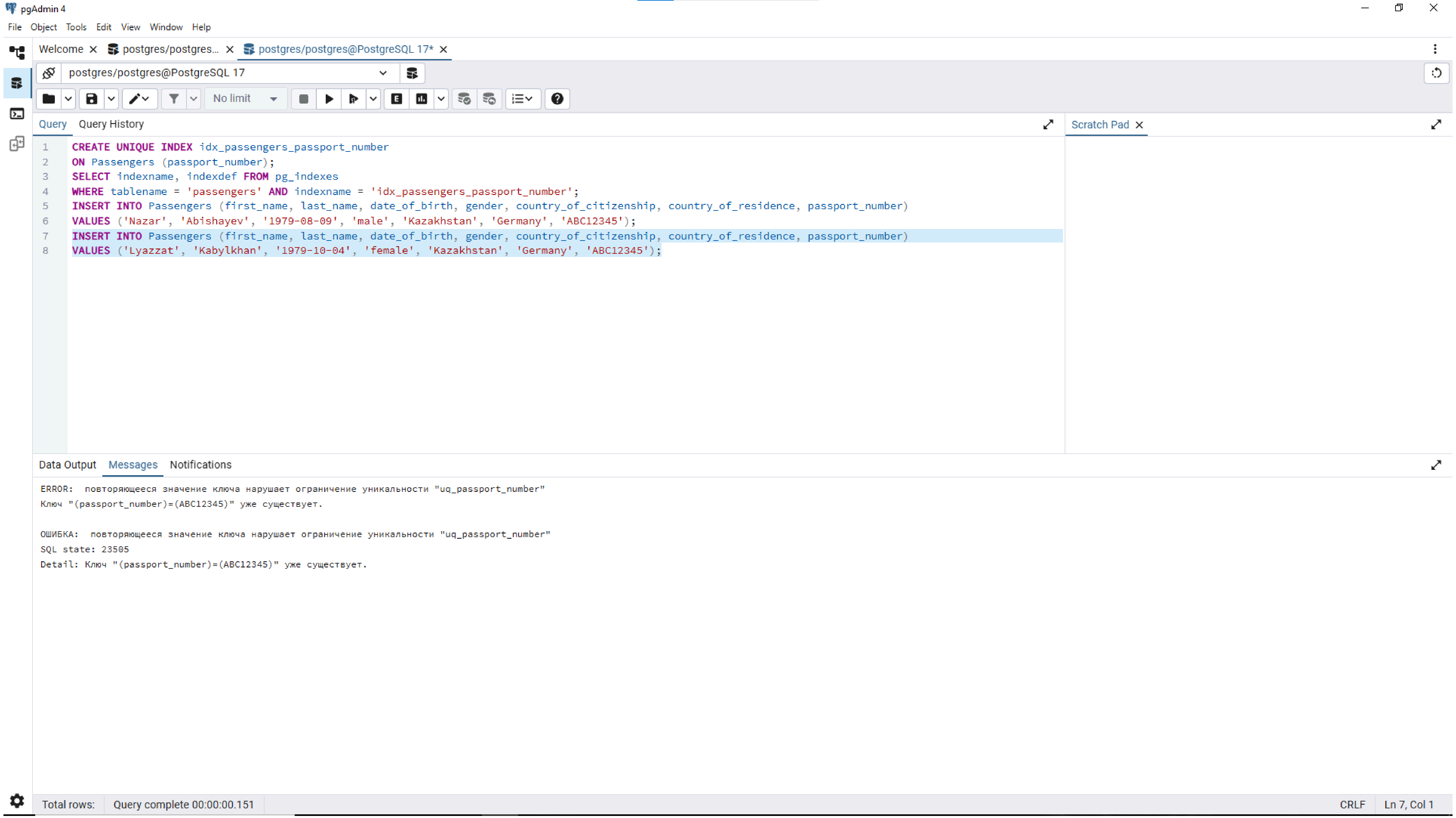
```
EXPLAIN ANALYZE
SELECT * FROM flights
WHERE departing_airport_id = '228' AND arriving_airport_id = '203';
```

	QUERY PLAN text
1	Seq Scan on flights (cost=0.00..6.00 rows=200 width=72) (actual time=0.051..0.197 rows=200 loops=...
2	Filter: ((departing_airport_id = 228) AND (arriving_airport_id = 203))
3	Planning Time: 0.333 ms
4	Execution Time: 0.259 ms

Successfully run. Total query runtime: 201 msec. 4 rows affected.

Total rows: 4   Query complete 00:00:00.201   CRLF   Ln 3, Col 66

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

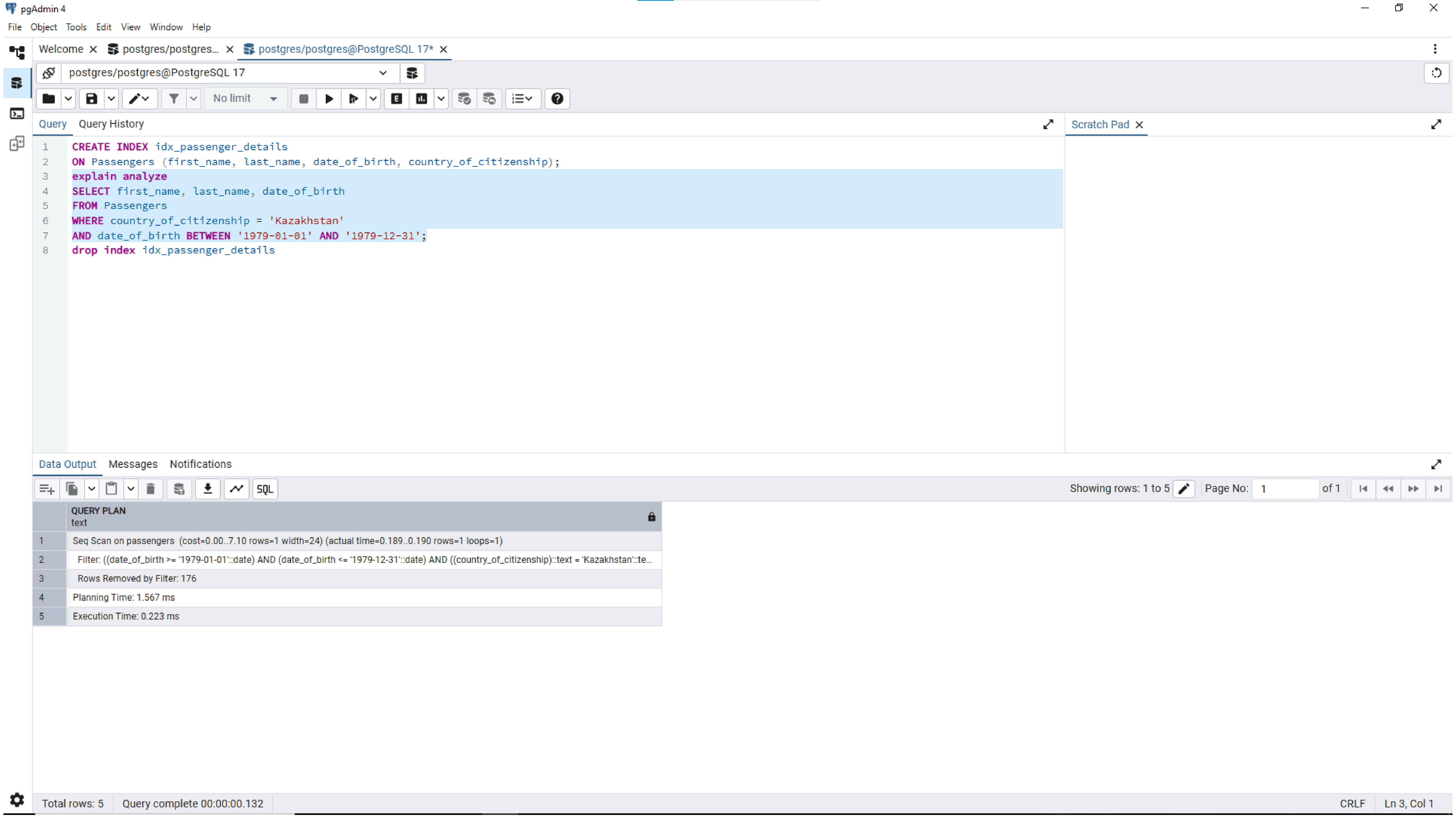


```sql
CREATE UNIQUE INDEX idx_passengers_passport_number
ON Passengers (passport_number);
SELECT indexname, indexdef FROM pg_indexes
WHERE tablename = 'passengers' AND indexname = 'idx_passengers_passport_number';
INSERT INTO Passengers (first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence, passport_number)
VALUES ('Nazar', 'Abishayev', '1979-08-09', 'male', 'Kazakhstan', 'Germany', 'ABC12345');
INSERT INTO Passengers (first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence, passport_number)
VALUES ('Lyazzat', 'Kabylkhan', '1979-10-04', 'female', 'Kazakhstan', 'Germany', 'ABC12345');
```

```
ERROR:  повторяющееся значение ключа нарушает ограничение уникальности "uq_passport_number"
Ключ "(passport_number)=(ABC12345)" уже существует.

ОШИБКА:  повторяющееся значение ключа нарушает ограничение уникальности "uq_passport_number"
SQL state: 23505
Detail: Ключ "(passport_number)=(ABC12345)" уже существует.
```

Total rows: Query complete 00:00:00.151 CRLF Ln 7, Col 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?

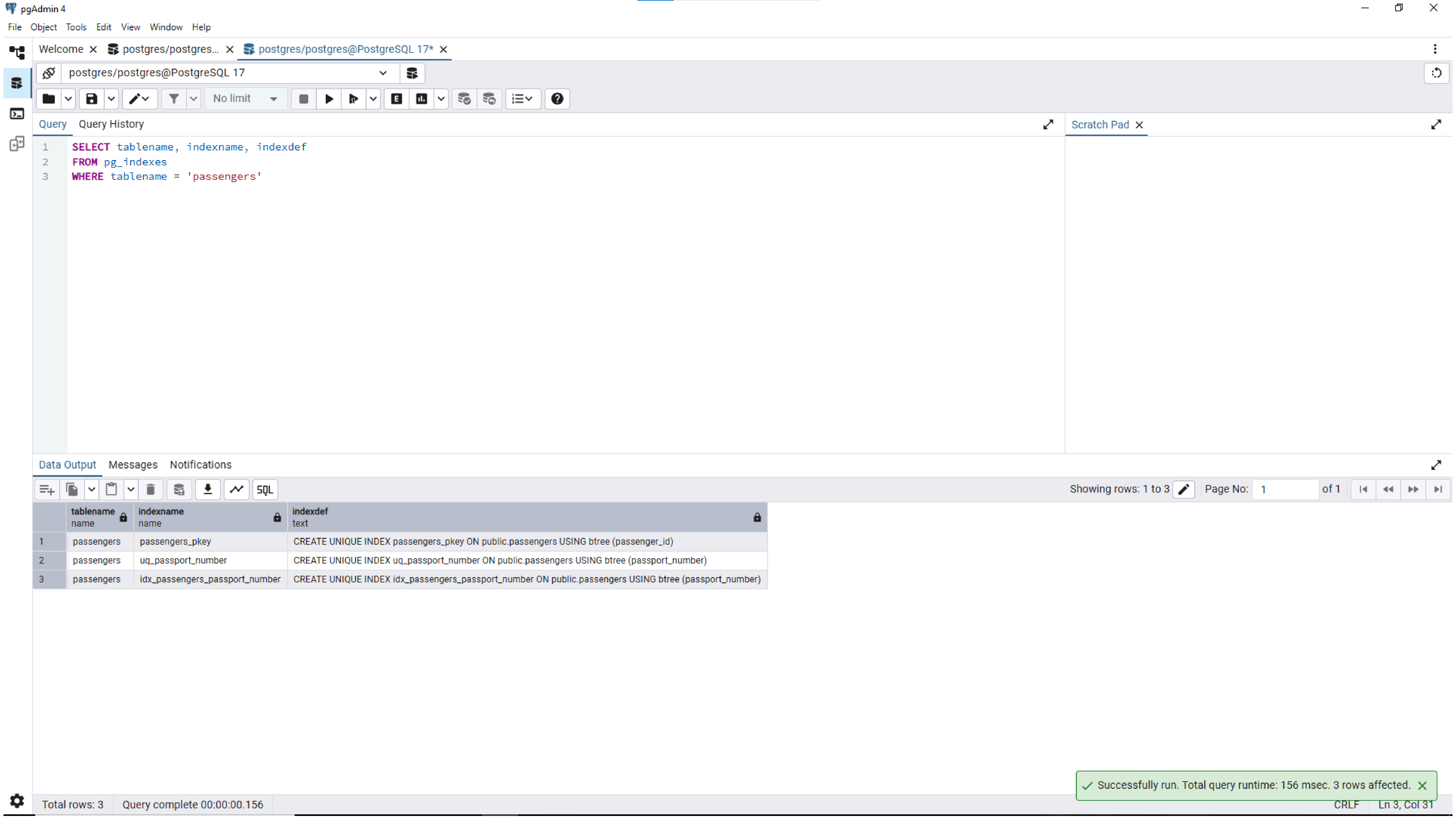


```sql
CREATE INDEX idx_passenger_details
ON Passengers (first_name, last_name, date_of_birth, country_of_citizenship);
explain analyze
SELECT first_name, last_name, date_of_birth
FROM Passengers
WHERE country_of_citizenship = 'Kazakhstan'
AND date_of_birth BETWEEN '1979-01-01' AND '1979-12-31';
drop index idx_passenger_details
```

	QUERY PLAN text
1	Seq Scan on passengers (cost=0.00..7.10 rows=1 width=24) (actual time=0.189..0.190 rows=1 loops=1)
2	Filter: ((date_of_birth >= '1979-01-01'::date) AND (date_of_birth <= '1979-12-31'::date) AND ((country_of_citizenship)::text = 'Kazakhstan'::te...
3	Rows Removed by Filter: 176
4	Planning Time: 1.567 ms
5	Execution Time: 0.223 ms

Total rows: 5   Query complete 00:00:00.132   CRLF   Ln 3, Col 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 Kazakhstan and was born in 1979 and check if the query uses indexes or not. Give the explanation of the results.



```sql
SELECT tablename, indexname, indexdef
FROM pg_indexes
WHERE tablename = 'passengers'
```

	tablename name	indexname name	indexdef text
1	passengers	passengers_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
2	passengers	uq_passport_number	CREATE UNIQUE INDEX uq_passport_number ON public.passengers USING btree (passport_number)
3	passengers	idx_passengers_passport_number	CREATE UNIQUE INDEX idx_passengers_passport_number ON public.passengers USING btree (passport_number)

✓ Successfully run. Total query runtime: 156 msec. 3 rows affected.

Total rows: 3   Query complete 00:00:00.156

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