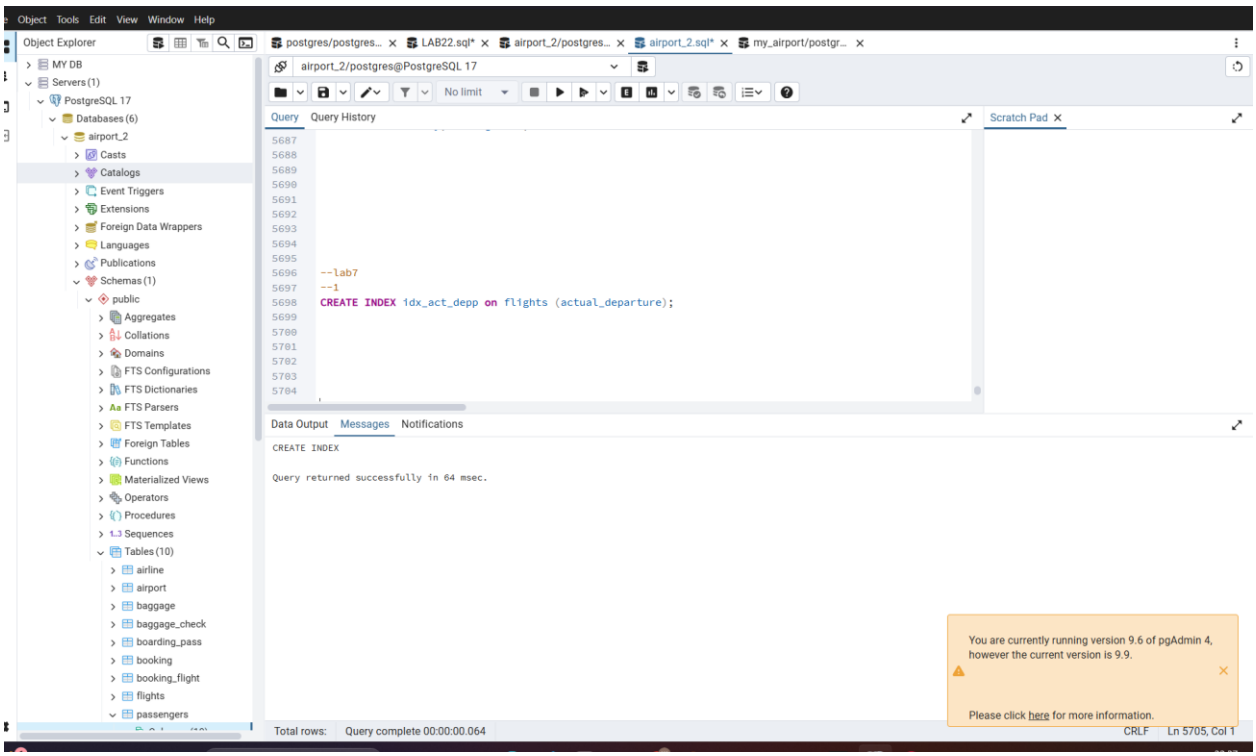
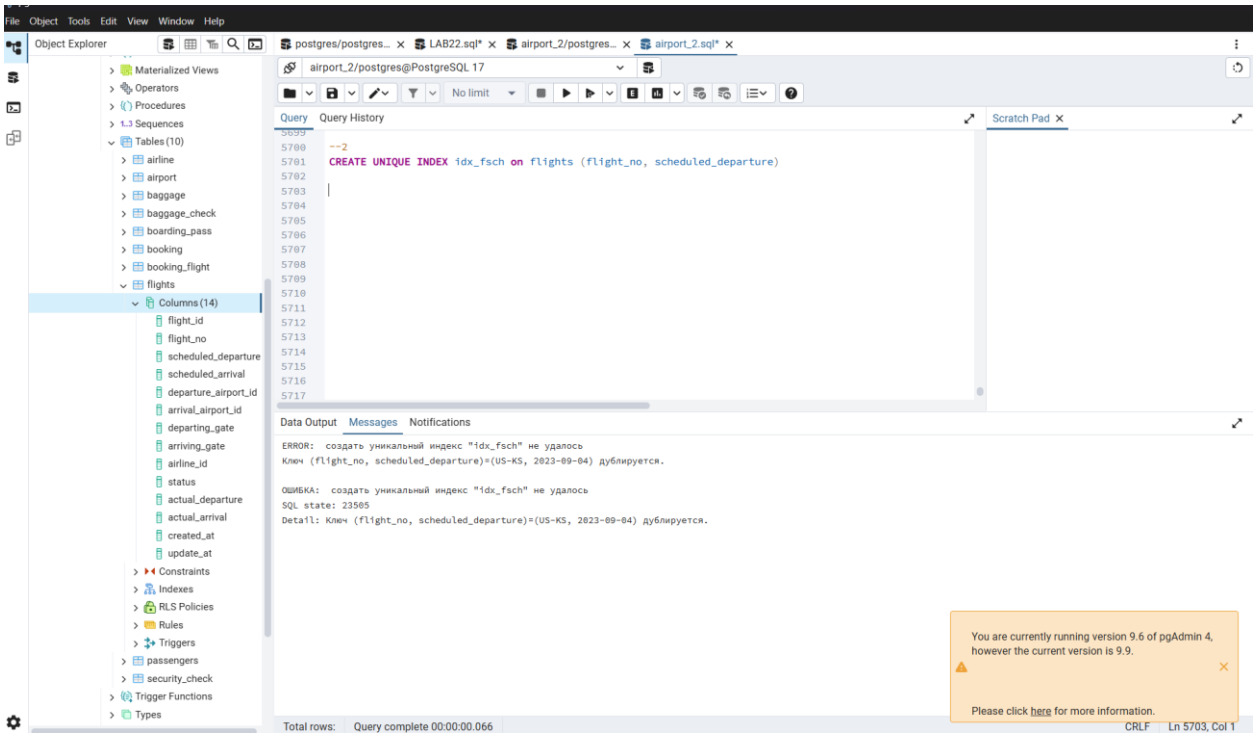
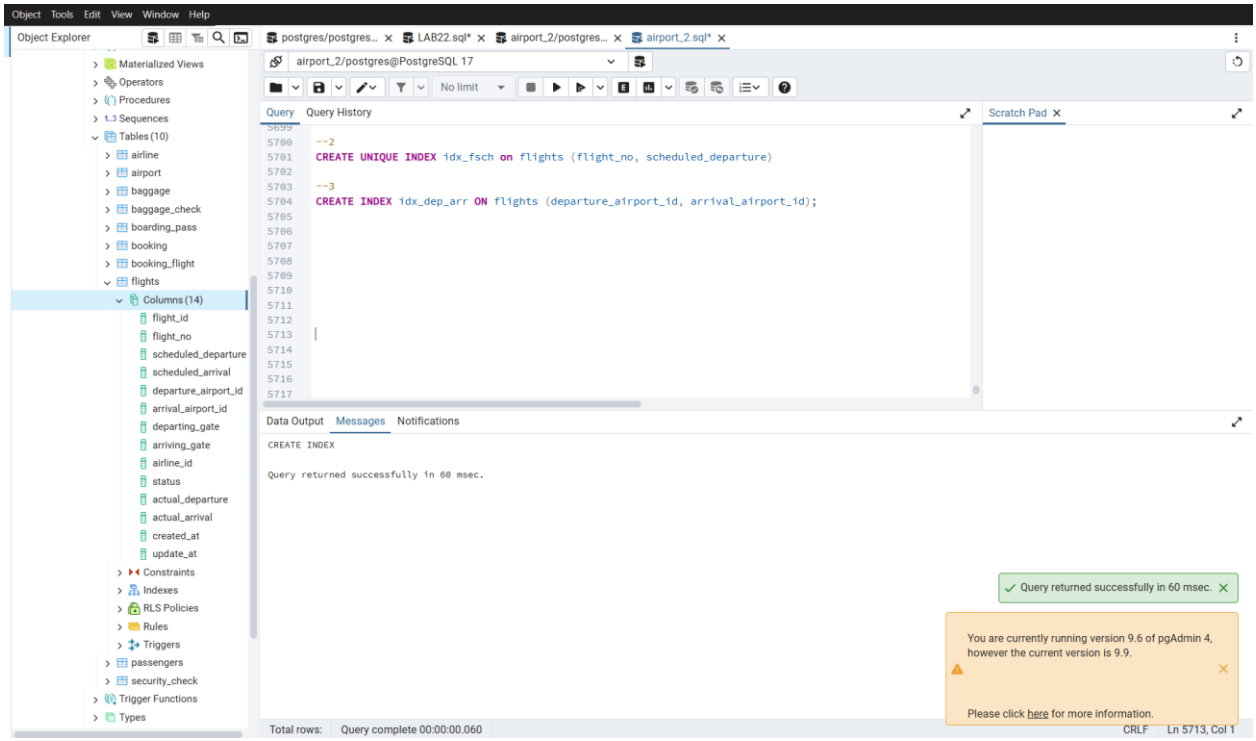
1 task


--lab7
--1
CREATE INDEX idx_act_depp on flights (actual_departure);
CREATE INDEX
Query returned successfully in 64 msec.
You are currently running version 9.6 of pgAdmin 4, however the current version is 9.9.
Please click here for more information.

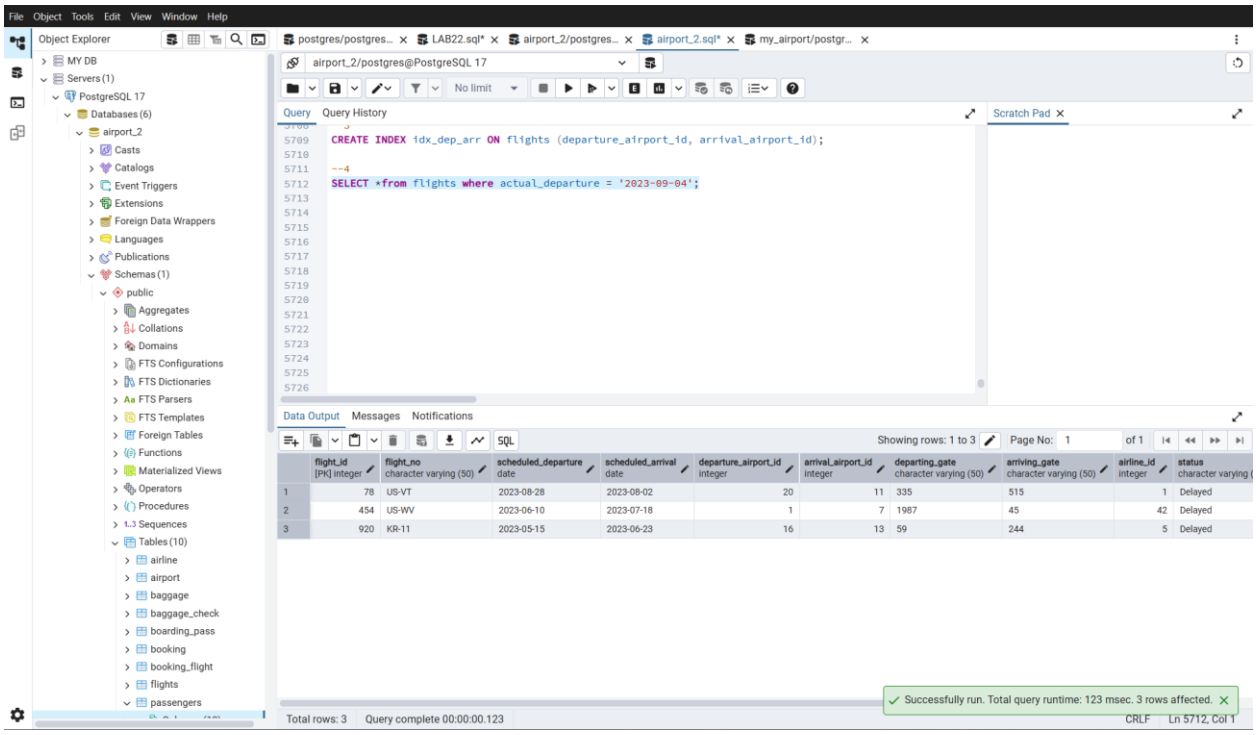
2 task

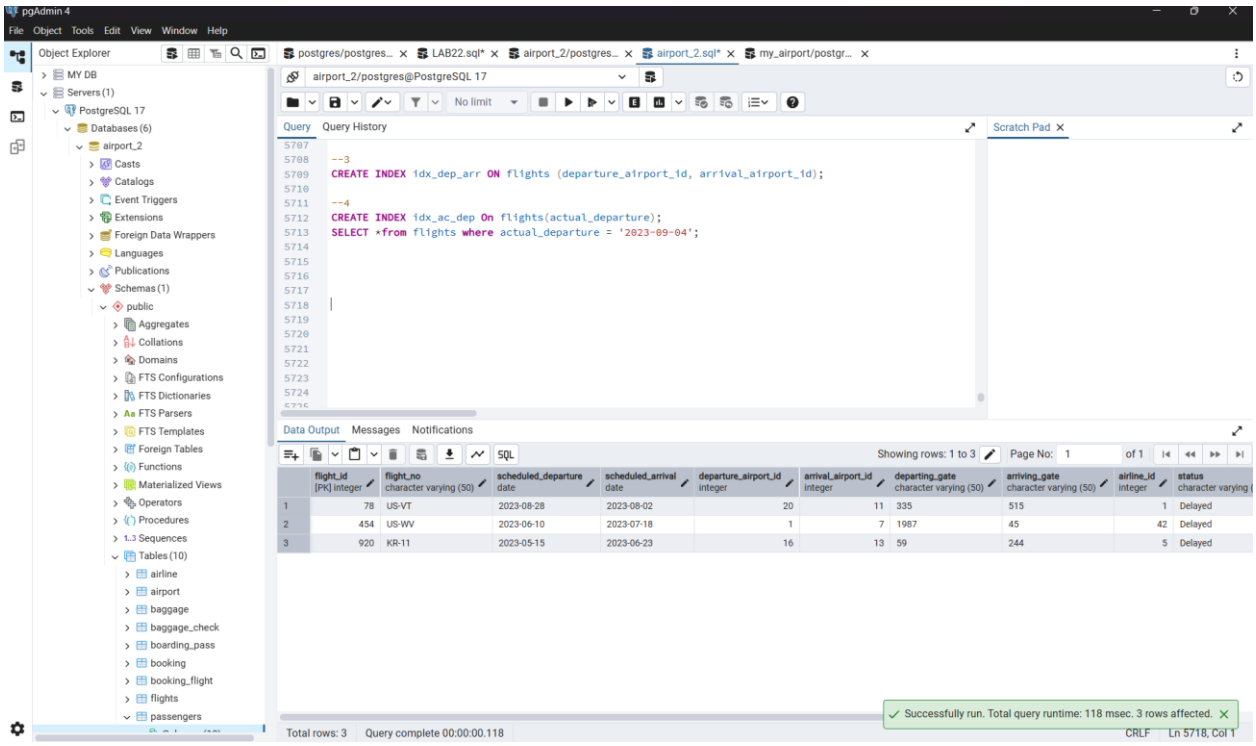

--2
CREATE UNIQUE INDEX idx_fsch on flights (flight_no, scheduled_departure)
ERROR: создать уникальный индекс "idx_fsch" не удалось
Ключ (flight_no, scheduled_departure)=(US-KS, 2023-09-04) дублируется.
ОШИБКА: создать уникальный индекс "idx_fsch" не удалось
SQL state: 23505
Detail: Ключ (flight_no, scheduled_departure)=(US-KS, 2023-09-04) дублируется.
You are currently running version 9.6 of pgAdmin 4, however the current version is 9.9.
Please click here for more information.

3 task

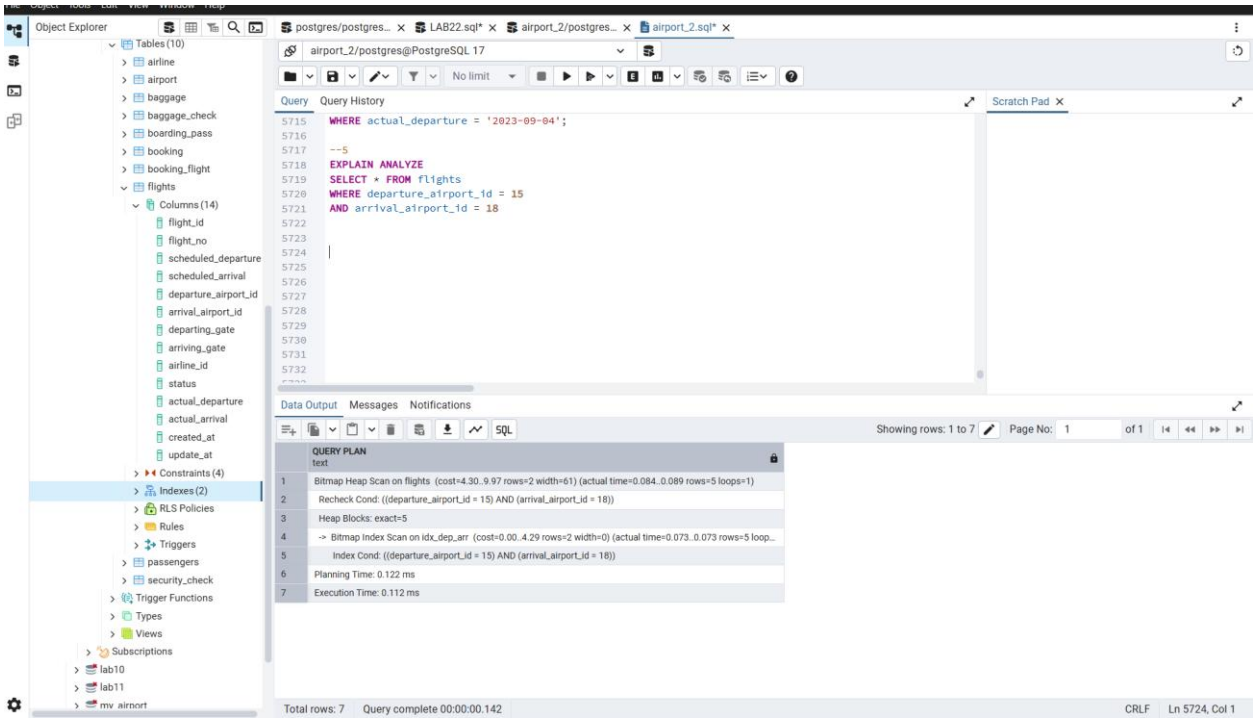

--2
CREATE UNIQUE INDEX idx_fsch on flights (flight_no, scheduled_departure)
--3
CREATE INDEX idx_dep_arr ON flights (departure_airport_id, arrival_airport_id);
CREATE INDEX
Query returned successfully in 60 msec.
Query returned successfully in 60 msec.
You are currently running version 9.6 of pgAdmin 4, however the current version is 9.9.
Please click here for more information.

4 task

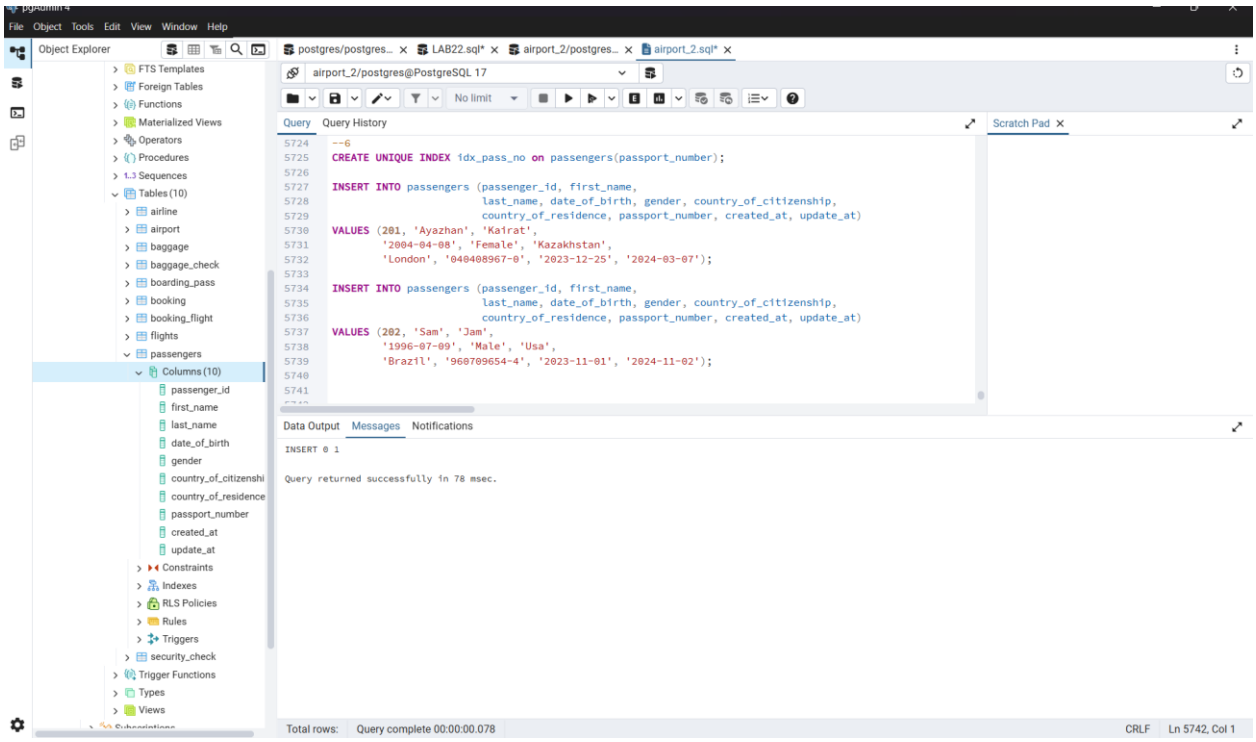

CREATE INDEX idx_dep_arr ON flights (departure_airport_id, arrival_airport_id);
--4
SELECT *from flights where actual_departure = '2023-09-04';
Successfully run. Total query runtime: 123 msec. 3 rows affected.


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
airport_2/postgres@PostgreSQL 17
Query Query History
Scratch Pad
--3
CREATE INDEX idx_dep_arr ON flights (departure_airport_id, arrival_airport_id);
--4
CREATE INDEX idx_ac_dep On flights(actual_departure);
SELECT *from flights where actual_departure = '2023-09-04';
Data Output Messages Notifications
Showing rows: 1 to 3
Page No: 1 of 1
flight_id [PK] integer
flight_no character varying (50)
scheduled_departure date
scheduled_arrival date
departure_airport_id integer
arrival_airport_id integer
departing_gate character varying (50)
arriving_gate character varying (50)
airline_id integer
status character varying
1 78 US-VT 2023-08-28 2023-08-02 20 11 335 515 1 Delayed
2 454 US-WV 2023-06-10 2023-07-18 1 7 1987 45 42 Delayed
3 920 KR-11 2023-05-15 2023-06-23 16 13 59 244 5 Delayed
Successfully run. Total query runtime: 118 msec. 3 rows affected.
Total rows: 3
Query complete 00:00:00.118
CRLF
Ln 5718, Col 1

5 task

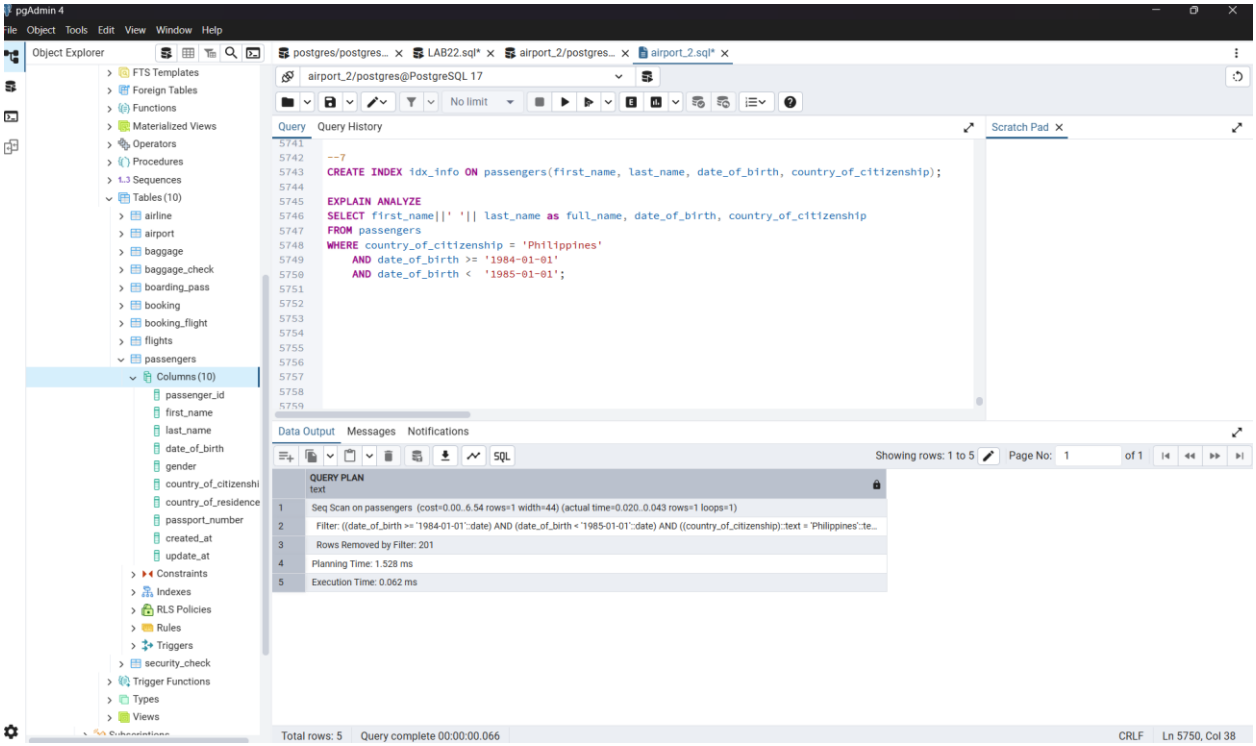

Object Explorer
airport_2/postgres@PostgreSQL 17
Query Query History
Scratch Pad
WHERE actual_departure = '2023-09-04';
--5
EXPLAIN ANALYZE
SELECT * FROM flights
WHERE departure_airport_id = 15
AND arrival_airport_id = 18
Data Output Messages Notifications
Showing rows: 1 to 7
Page No: 1 of 1
QUERY PLAN text
1 Bitmap Heap Scan on flights (cost=4.30..9.97 rows=2 width=61) (actual time=0.084..0.089 rows=5 loops=1)
2 Recheck Cond: ((departure_airport_id = 15) AND (arrival_airport_id = 18))
3 Heap Blocks: exact=5
4 -> Bitmap Index Scan on idx_dep_arr (cost=0.00..4.29 rows=2 width=0) (actual time=0.073..0.073 rows=5 loop...
5 Index Cond: ((departure_airport_id = 15) AND (arrival_airport_id = 18))
6 Planning Time: 0.122 ms
7 Execution Time: 0.112 ms
Total rows: 7
Query complete 00:00:00.142
CRLF
Ln 5724, Col 1

6 task



When I add a UNIQUE index to the *passport_number* column in the *Passengers* table, the database guarantees that no two rows can have the same passport number. This means every passport number must be different. After creating the index, I check whether it was successfully created. Then I insert two new rows with different passport numbers, and the operation works without any problems. However, if I try to insert rows with the same passport number, the database returns an error. This error indicates that the uniqueness constraint has been violated because the passport number already exists in the table.

7 task



First, we created a composite index on the columns country_of_citizenship, date_of_birth, first_name, and last_name.
These columns were chosen because the query filters by country_of_citizenship and date_of_birth, so placing them first in the index improves performance. The SQL query searches for passengers who are citizens of the Philippines and were born in 1984. To allow PostgreSQL to use the index efficiently, the date condition was written as a range:

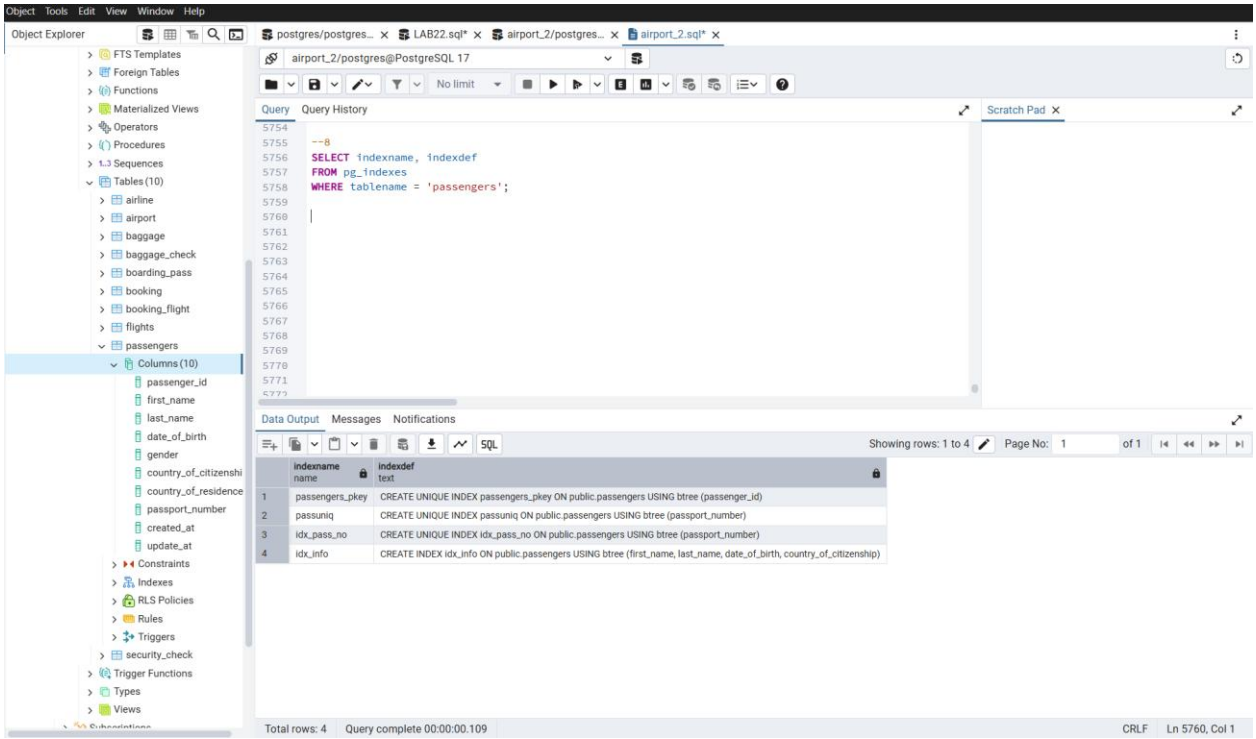
date_of_birth >= '1984-01-01'

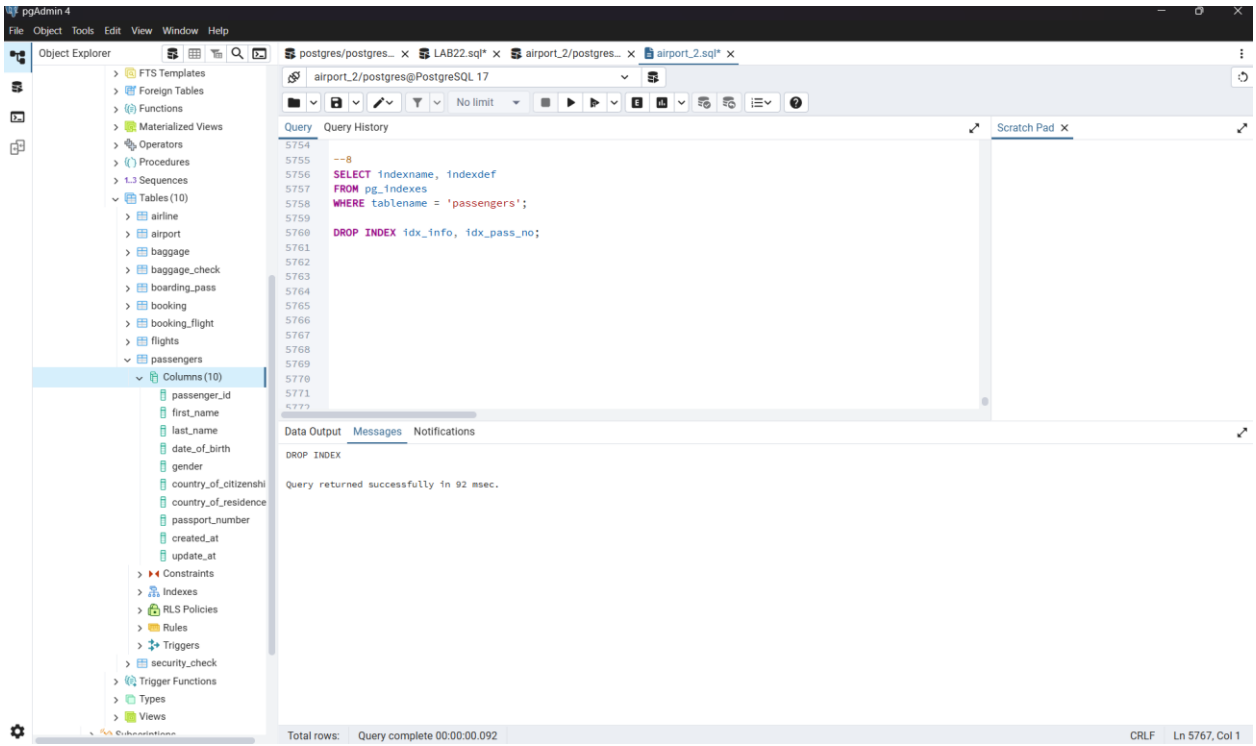
AND date_of_birth < '1985-01-01'

When running the query with EXPLAIN ANALYZE, PostgreSQL performs an **Index Scan**, which confirms that the newly created index is actually used. If the date had been written using a function such as EXTRACT(YEAR FROM date_of_birth), PostgreSQL would instead perform a **Sequential Scan**, because functions applied to indexed columns prevent index usage.

Therefore, the query with the date range and the composite index provides the most efficient execution plan.

8 task




pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
postgres/postgres...
LAB22.sql*
airport_2/postgres...
airport_2.sql*
airport_2/postgres@PostgreSQL 17
No limit
Query
Query History
Scratch Pad
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (10)
airline
airport
baggage
baggage_check
boarding_pass
booking
booking_flight
flights
passengers
Columns (10)
passenger_id
first_name
last_name
date_of_birth
gender
country_of_citizenshi
country_of_residence
passport_number
created_at
update_at
Constraints
Indexes
RLS Policies
Rules
Triggers
security_check
Trigger Functions
Types
Views
5754
5755 --8
5756 SELECT indexname, indexdef
5757 FROM pg_indexes
5758 WHERE tablename = 'passengers';
5759
5760 DROP INDEX idx_info, idx_pass_no;
5761
5762
5763
5764
5765
5766
5767
5768
5769
5770
5771
Data Output
Messages
Notifications
DROP INDEX
Query returned successfully in 92 msec.
Total rows:
Query complete 00:00:00.092
CRLF
Ln 5767, Col 1