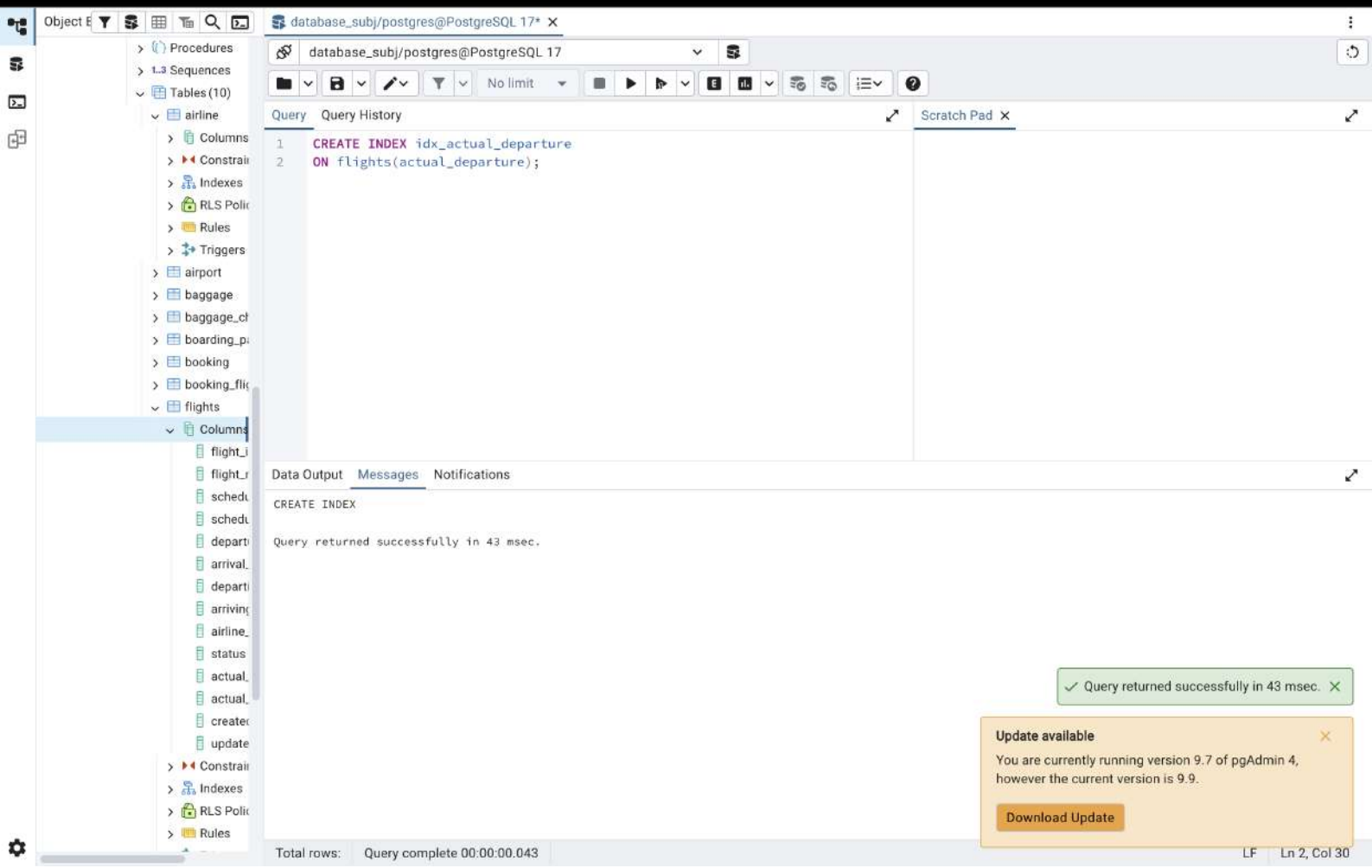
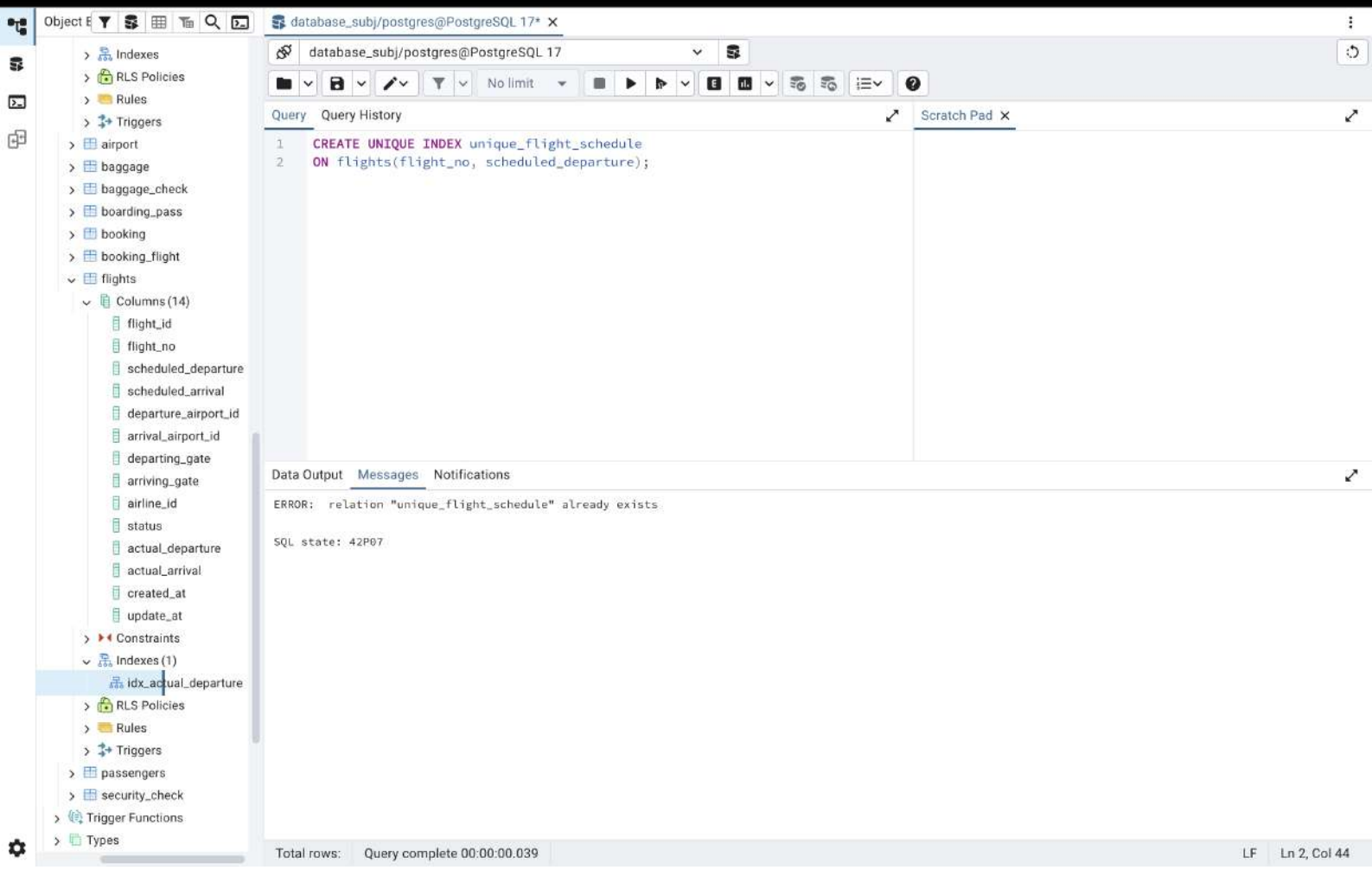
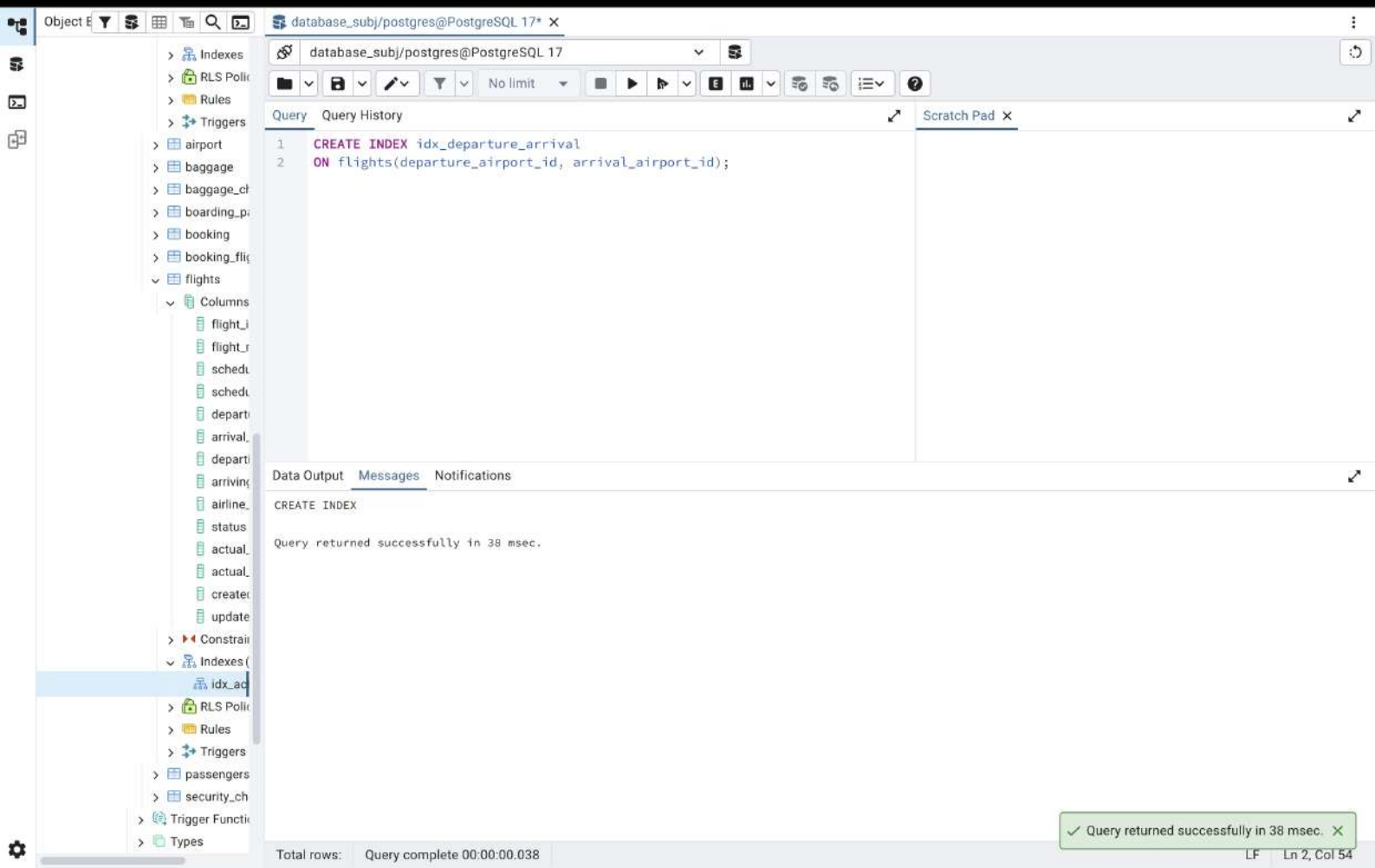
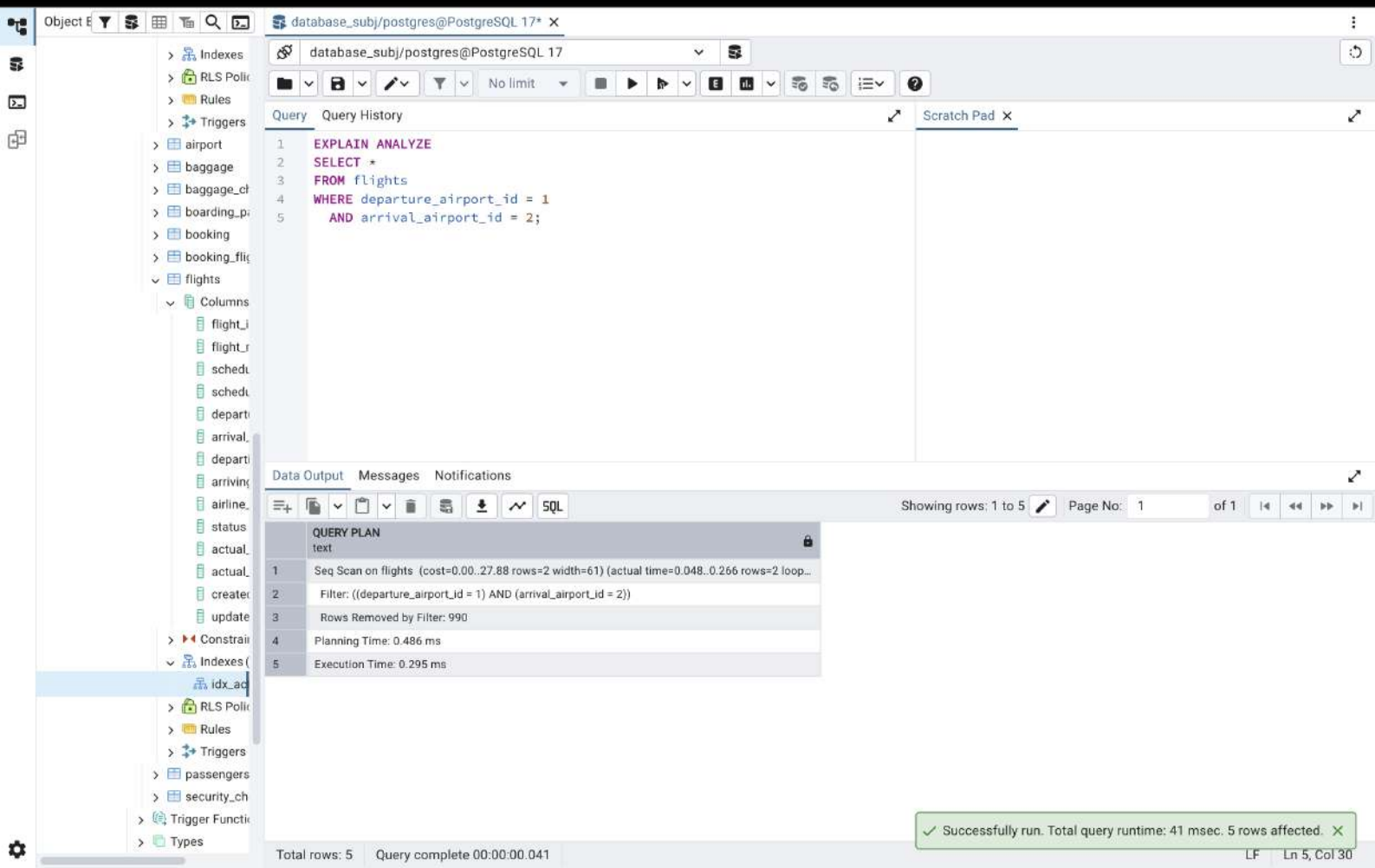
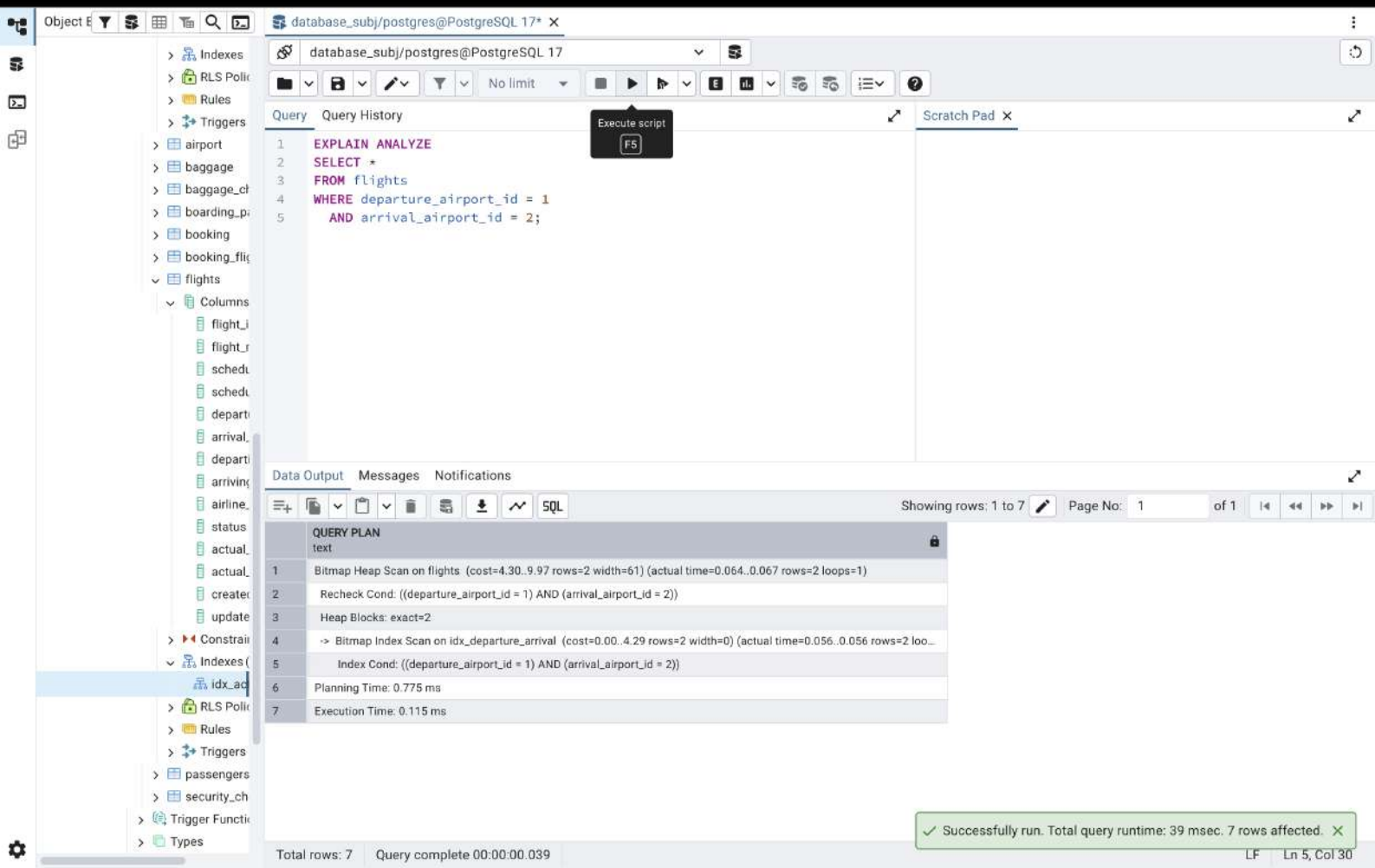
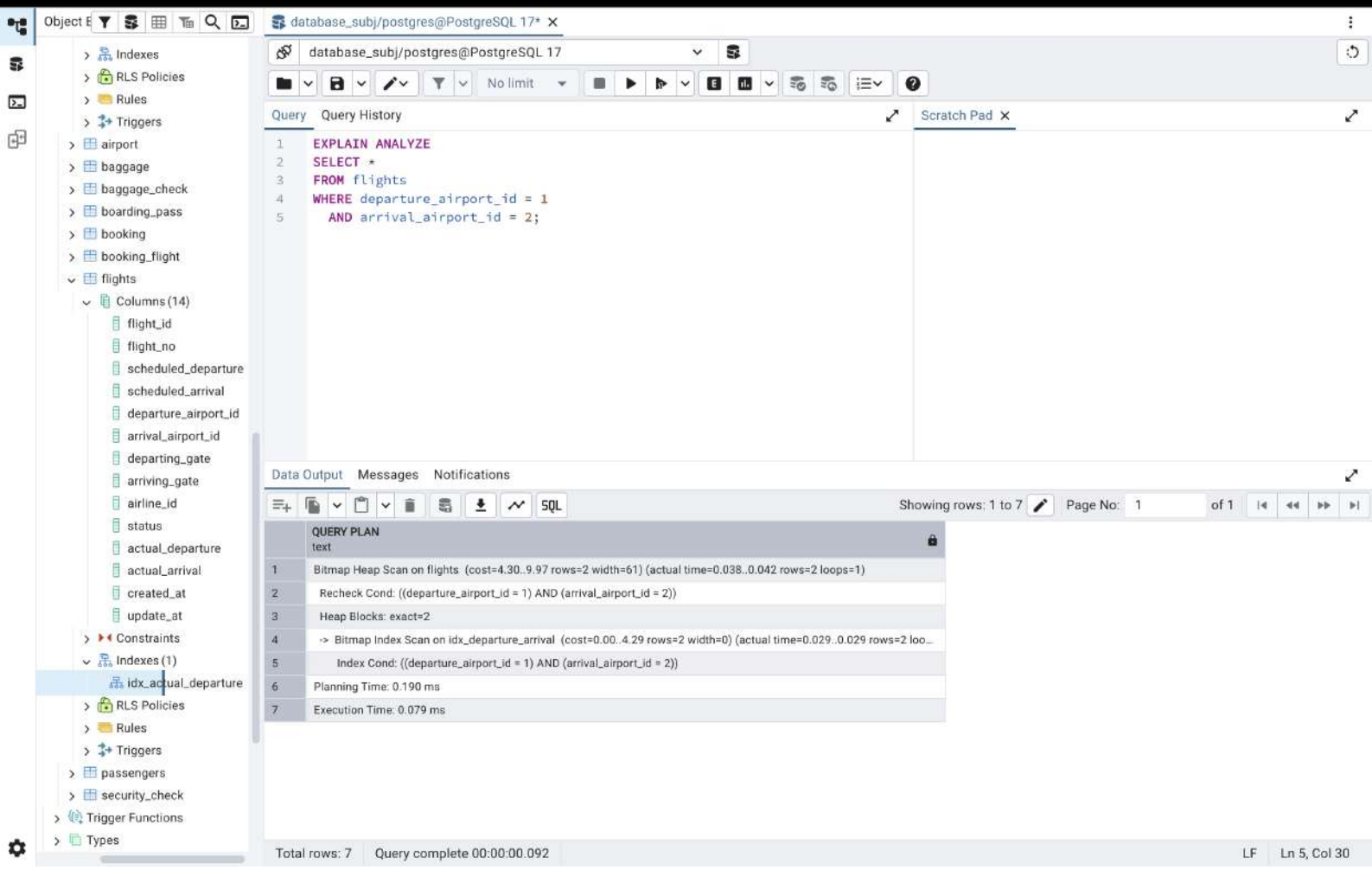

database_subj/postgres@PostgreSQL 17*
database_subj/postgres@PostgreSQL 17
No limit
Query
Query History
Scratch Pad
1 CREATE INDEX idx_actual_departure
2 ON flights(actual_departure);
Data Output
Messages
Notifications
CREATE INDEX
Query returned successfully in 43 msec.
Query returned successfully in 43 msec.
Update available
You are currently running version 9.7 of pgAdmin 4, however the current version is 9.9.
Download Update
Total rows:
Query complete 00:00:00.043
LF
Ln 2, Col 30

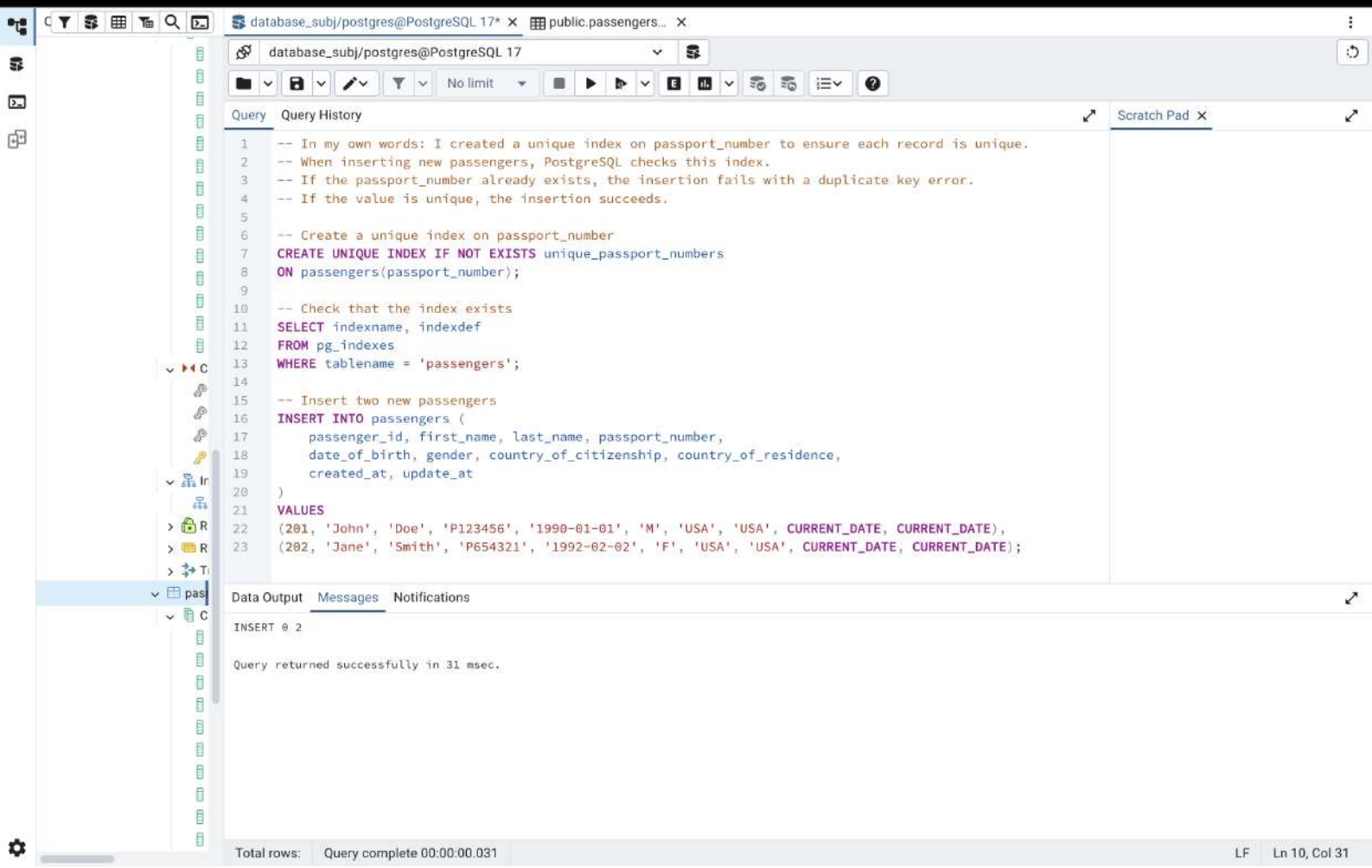

Object E
database_subj/postgres@PostgreSQL 17*
database_subj/postgres@PostgreSQL 17
No limit
Query
Query History
Scratch Pad
1 CREATE UNIQUE INDEX unique_flight_schedule
2 ON flights(flight_no, scheduled_departure);
Indexes
RLS Policies
Rules
Triggers
airport
baggage
baggage_check
boarding_pass
booking
booking_flight
flights
Columns (14)
flight_id
flight_no
scheduled_departure
scheduled_arrival
departure_airport_id
arrival_airport_id
departing_gate
arriving_gate
airline_id
status
actual_departure
actual_arrival
created_at
update_at
Constraints
Indexes (1)
idx_actual_departure
RLS Policies
Rules
Triggers
passengers
security_check
Trigger Functions
Types
Data Output
Messages
Notifications
ERROR: relation "unique_flight_schedule" already exists
SQL state: 42P07
Total rows:
Query complete 00:00:00.039
LF
Ln 2, Col 44

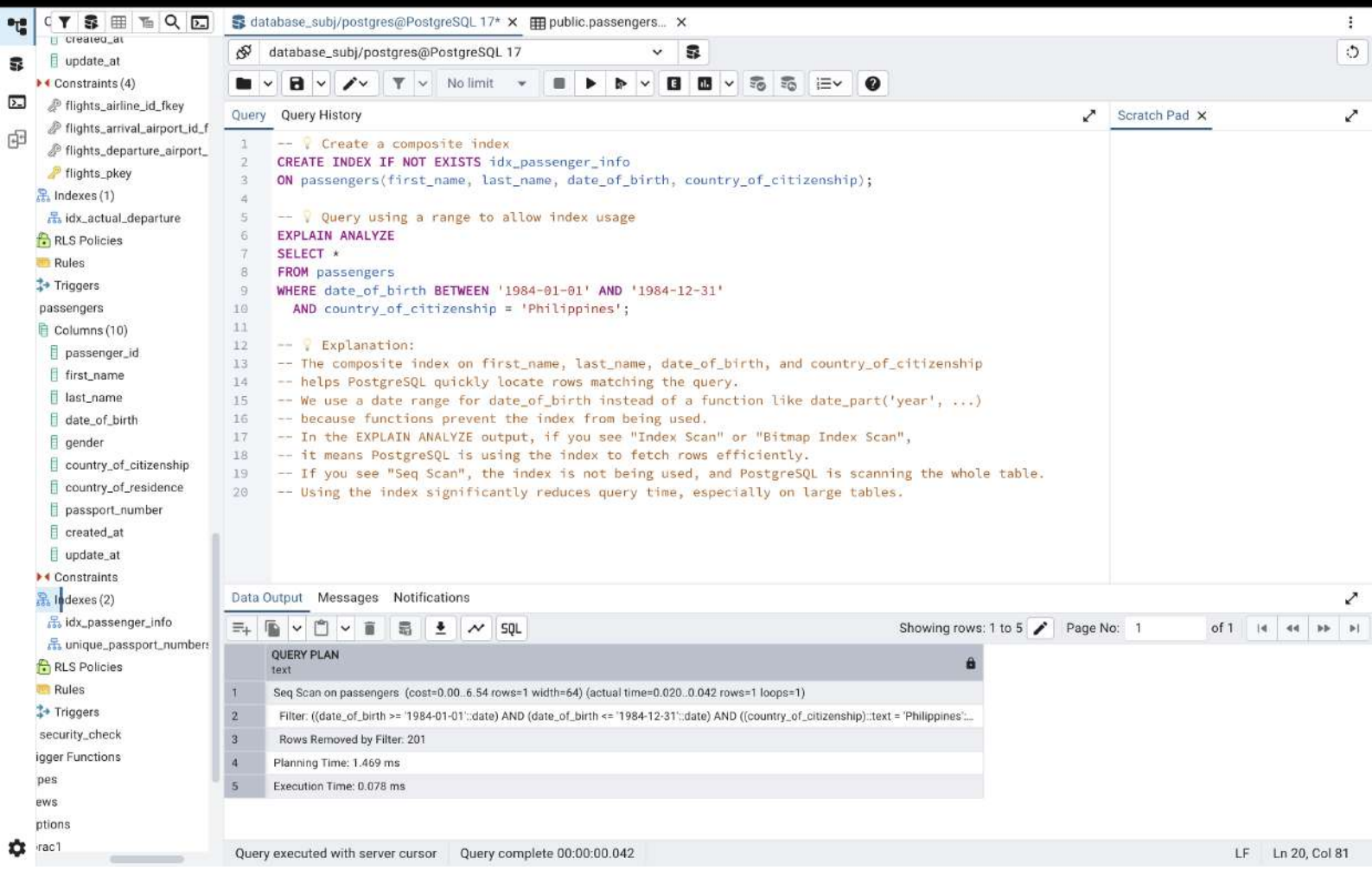

database_subj/postgres@PostgreSQL 17*
database_subj/postgres@PostgreSQL 17
No limit
Query
Query History
Scratch Pad
CREATE INDEX idx_departure_arrival
ON flights(departure_airport_id, arrival_airport_id);
Data Output
Messages
Notifications
CREATE INDEX
Query returned successfully in 38 msec.
Total rows:
Query complete 00:00:00.038
Query returned successfully in 38 msec.
Ln 2, Col 54

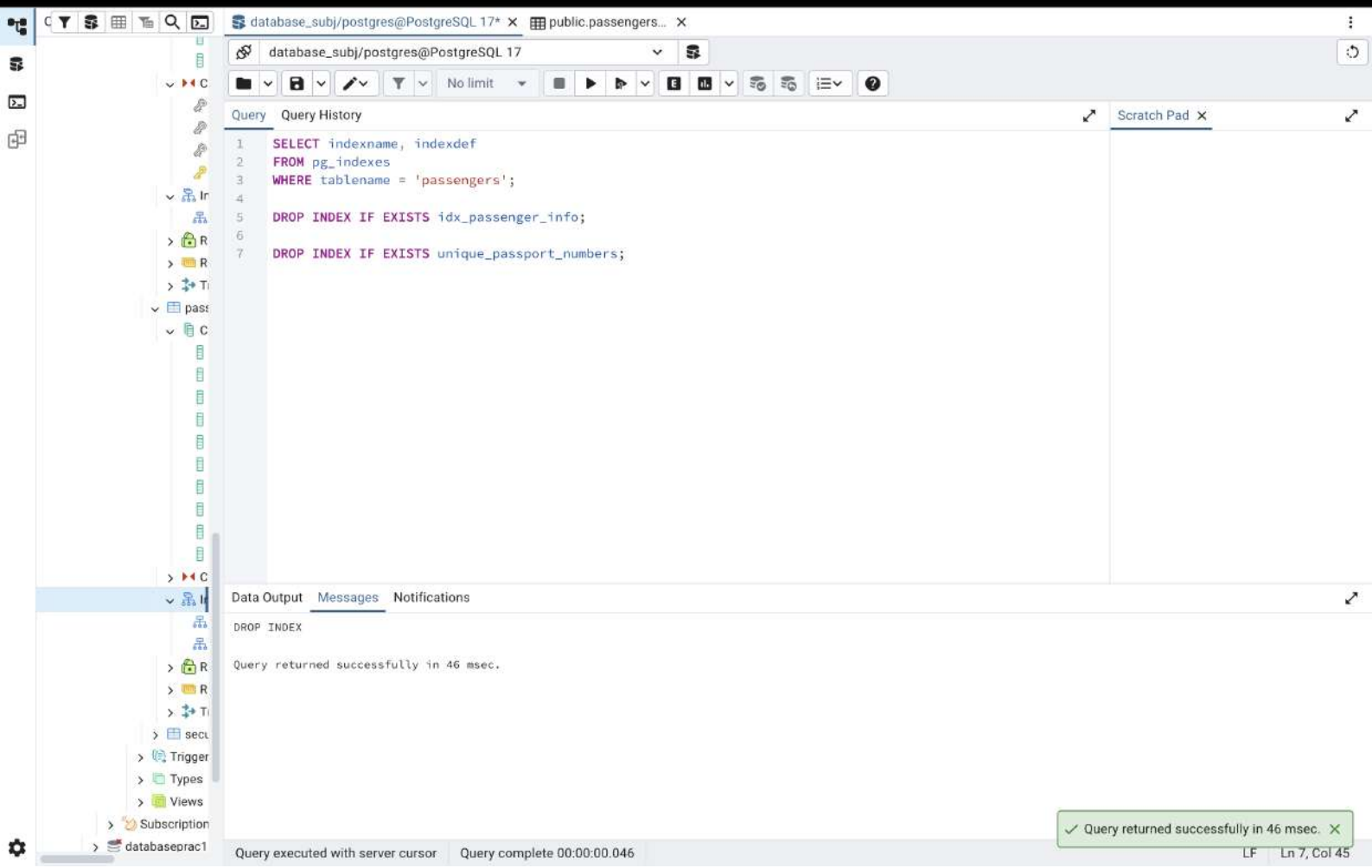

database_subj/postgres@PostgreSQL 17*
database_subj/postgres@PostgreSQL 17
No limit
Query
Query History
Scratch Pad
EXPLAIN ANALYZE
SELECT *
FROM flights
WHERE departure_airport_id = 1
AND arrival_airport_id = 2;
Data Output
Messages
Notifications
Showing rows: 1 to 5
Page No: 1
of 1
QUERY PLAN
text
Seq Scan on flights (cost=0.00..27.88 rows=2 width=61) (actual time=0.048..0.266 rows=2 loop...
Filter: ((departure_airport_id = 1) AND (arrival_airport_id = 2))
Rows Removed by Filter: 990
Planning Time: 0.486 ms
Execution Time: 0.295 ms
Successfully run. Total query runtime: 41 msec. 5 rows affected.
Total rows: 5
Query complete 00:00:00.041
LF
Ln 5, Col 30


database_subj/postgres@PostgreSQL 17*
database_subj/postgres@PostgreSQL 17
No limit
Execute script
F5
Query
Query History
Scratch Pad
1 EXPLAIN ANALYZE
2 SELECT *
3 FROM flights
4 WHERE departure_airport_id = 1
5 AND arrival_airport_id = 2;
Data Output
Messages
Notifications
SQL
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=0.064..0.067 rows=2 loops=1)
2 Recheck Cond: ((departure_airport_id = 1) AND (arrival_airport_id = 2))
3 Heap Blocks: exact=2
4 -> Bitmap Index Scan on idx_departure_arrival (cost=0.00..4.29 rows=2 width=0) (actual time=0.056..0.056 rows=2 loo...
5 Index Cond: ((departure_airport_id = 1) AND (arrival_airport_id = 2))
6 Planning Time: 0.775 ms
7 Execution Time: 0.115 ms
Successfully run. Total query runtime: 39 msec. 7 rows affected.
Total rows: 7
Query complete 00:00:00.039
LF
Ln 5, Col 30


database_subj/postgres@PostgreSQL 17*
database_subj/postgres@PostgreSQL 17
No limit
Query
Query History
Scratch Pad
EXPLAIN ANALYZE
SELECT *
FROM flights
WHERE departure_airport_id = 1
AND arrival_airport_id = 2;
Data Output
Messages
Notifications
Showing rows: 1 to 7
Page No: 1
of 1
QUERY PLAN
text
Bitmap Heap Scan on flights (cost=4.30..9.97 rows=2 width=61) (actual time=0.038..0.042 rows=2 loops=1)
Recheck Cond: ((departure_airport_id = 1) AND (arrival_airport_id = 2))
Heap Blocks: exact=2
-> Bitmap Index Scan on idx_departure_arrival (cost=0.00..4.29 rows=2 width=0) (actual time=0.029..0.029 rows=2 loo...
Index Cond: ((departure_airport_id = 1) AND (arrival_airport_id = 2))
Planning Time: 0.190 ms
Execution Time: 0.079 ms
Total rows: 7
Query complete 00:00:00.092
LF
Ln 5, Col 30
Indexes
RLS Policies
Rules
Triggers
airport
baggage
baggage_check
boarding_pass
booking
booking_flight
flights
Columns (14)
flight_id
flight_no
scheduled_departure
scheduled_arrival
departure_airport_id
arrival_airport_id
departing_gate
arriving_gate
airline_id
status
actual_departure
actual_arrival
created_at
update_at
Constraints
Indexes (1)
idx_actual_departure
RLS Policies
Rules
Triggers
passengers
security_check
Trigger Functions
Types


database_subj/postgres@PostgreSQL 17*
public.passengers...
database_subj/postgres@PostgreSQL 17
No limit
Query
Query History
Scratch Pad
-- In my own words: I created a unique index on passport_number to ensure each record is unique.
-- When inserting new passengers, PostgreSQL checks this index.
-- If the passport_number already exists, the insertion fails with a duplicate key error.
-- If the value is unique, the insertion succeeds.
-- Create a unique index on passport_number
CREATE UNIQUE INDEX IF NOT EXISTS unique_passport_numbers
ON passengers(passport_number);
-- Check that the index exists
SELECT indexname, indexdef
FROM pg_indexes
WHERE tablename = 'passengers';
-- Insert two new passengers
INSERT INTO passengers (
passenger_id, first_name, last_name, passport_number,
date_of_birth, gender, country_of_citizenship, country_of_residence,
created_at, update_at
)
VALUES
(201, 'John', 'Doe', 'P123456', '1990-01-01', 'M', 'USA', 'USA', CURRENT_DATE, CURRENT_DATE),
(202, 'Jane', 'Smith', 'P654321', '1992-02-02', 'F', 'USA', 'USA', CURRENT_DATE, CURRENT_DATE);
Data Output
Messages
Notifications
INSERT 0 2
Query returned successfully in 31 msec.
Total rows:
Query complete 00:00:00.031
LF
Ln 10, Col 31


database_subj/postgres@PostgreSQL 17*
public.passengers...
database_subj/postgres@PostgreSQL 17
No limit
Query
Query History
Scratch Pad
-- 💡 Create a composite index
CREATE INDEX IF NOT EXISTS idx_passenger_info
ON passengers(first_name, last_name, date_of_birth, country_of_citizenship);
-- 💡 Query using a range to allow index usage
EXPLAIN ANALYZE
SELECT *
FROM passengers
WHERE date_of_birth BETWEEN '1984-01-01' AND '1984-12-31'
AND country_of_citizenship = 'Philippines';
-- 💡 Explanation:
-- The composite index on first_name, last_name, date_of_birth, and country_of_citizenship
-- helps PostgreSQL quickly locate rows matching the query.
-- We use a date range for date_of_birth instead of a function like date_part('year', ...)
-- because functions prevent the index from being used.
-- In the EXPLAIN ANALYZE output, if you see "Index Scan" or "Bitmap Index Scan",
-- it means PostgreSQL is using the index to fetch rows efficiently.
-- If you see "Seq Scan", the index is not being used, and PostgreSQL is scanning the whole table.
-- Using the index significantly reduces query time, especially on large tables.
Data Output
Messages
Notifications
Showing rows: 1 to 5
Page No: 1
of 1
QUERY PLAN
text
1 Seq Scan on passengers (cost=0.00..6.54 rows=1 width=64) (actual time=0.020..0.042 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'::...
3 Rows Removed by Filter: 201
4 Planning Time: 1.469 ms
5 Execution Time: 0.078 ms
Query executed with server cursor
Query complete 00:00:00.042
LF
Ln 20, Col 81


database_subj/postgres@PostgreSQL 17*
public.passengers...
database_subj/postgres@PostgreSQL 17
No limit
Query
Query History
Scratch Pad
1 SELECT indexname, indexdef
2 FROM pg_indexes
3 WHERE tablename = 'passengers';
4
5 DROP INDEX IF EXISTS idx_passenger_info;
6
7 DROP INDEX IF EXISTS unique_passport_numbers;
Data Output
Messages
Notifications
DROP INDEX
Query returned successfully in 46 msec.
Trigger
Types
Views
Subscription
databaseprac1
Query returned successfully in 46 msec.
Query executed with server cursor
Query complete 00:00:00.046
LF
Ln 7, Col 45