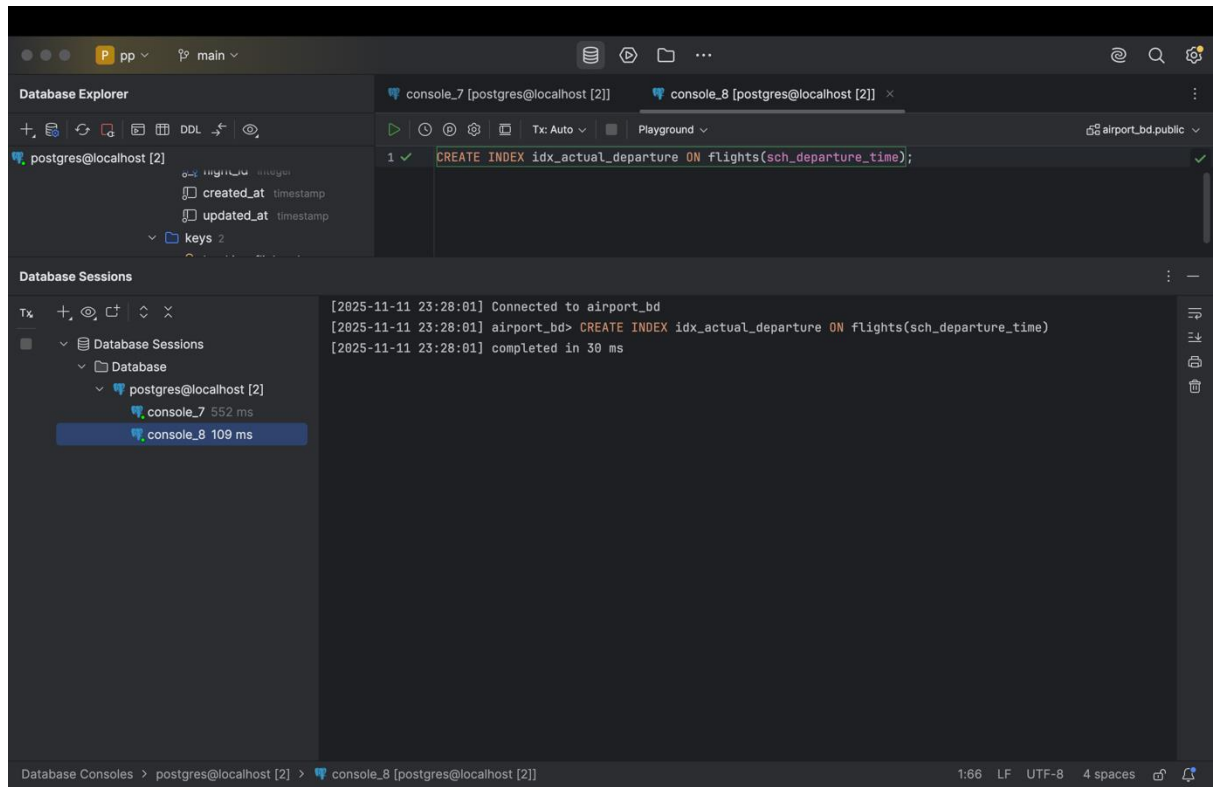
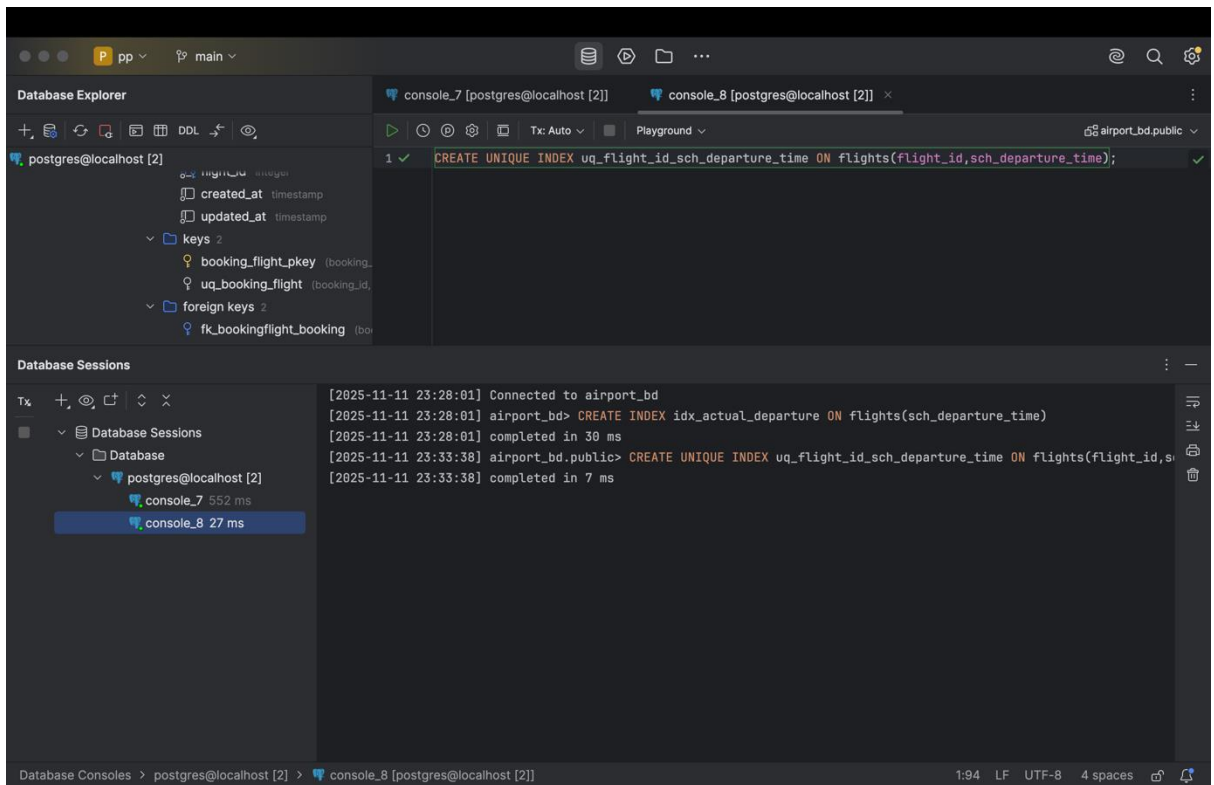


Laboratory work 7

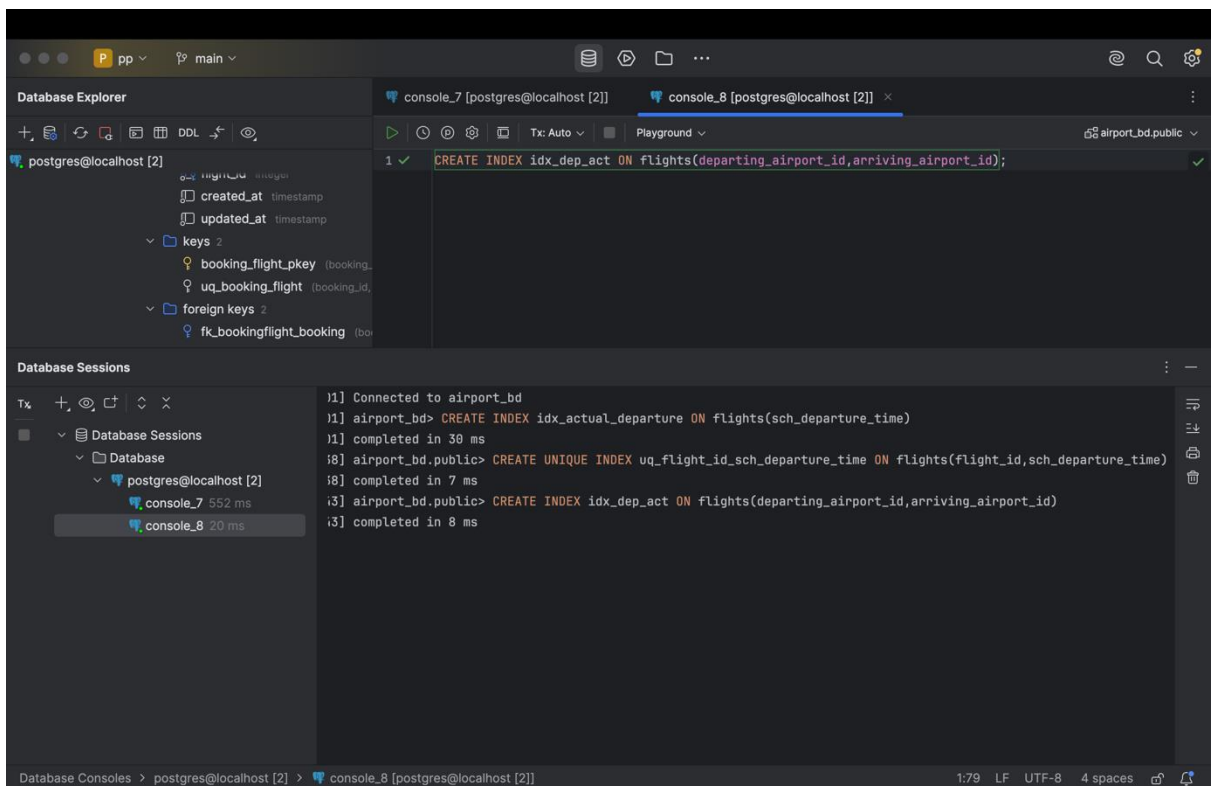
1. Create an index on the actual_departure column in the flights table.



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.

Without:

The screenshot shows a PostgreSQL IDE interface. On the left, the 'Database Explorer' pane displays the 'airports' database structure, including tables 'created_at', 'updated_at', and a 'keys' section with 'booking_flight_pkey' and 'uq_booking_flight'. The 'Database Sessions' pane shows two sessions: 'console_7' (552 ms) and 'console_8' (386 ms). The main editor shows a SQL query in 'console_8':

```
1 EXPLAIN ANALYSE
2 SELECT * FROM flights WHERE departing_airport_id = 1 AND arriving_airport_id = 3;
```

The 'Output' pane displays the query plan for 'Result 4':

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Actual Loops
1	Seq Scan on flights	(cost=0.00..7.00)	rows=1	width=70	(actual time=0.044..0.044)	rows=0	loops=1
2	Filter: ((departing_airport_id = 1) AND (arriving_airport_id = 3))						
3	Rows Removed by Filter: 200						
4	Planning Time: 1.700 ms						
5	Execution Time: 0.095 ms						

With:

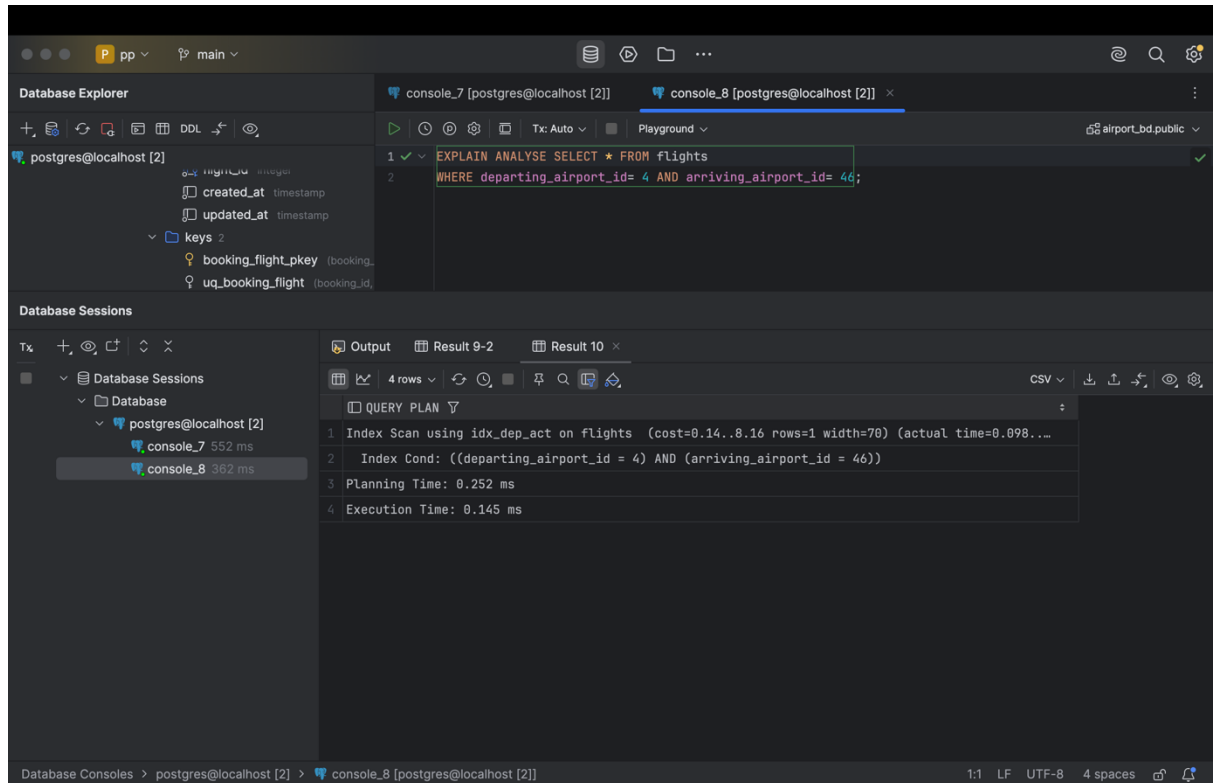
The screenshot shows the same PostgreSQL IDE interface. The 'Database Explorer' and 'Database Sessions' panes are identical to the previous screenshot. The main editor shows a SQL query in 'console_8':

```
1 SET enable_seqscan = off;
2 EXPLAIN ANALYSE SELECT * FROM flights
3 WHERE departing_airport_id=1 AND arriving_airport_id=3;
```

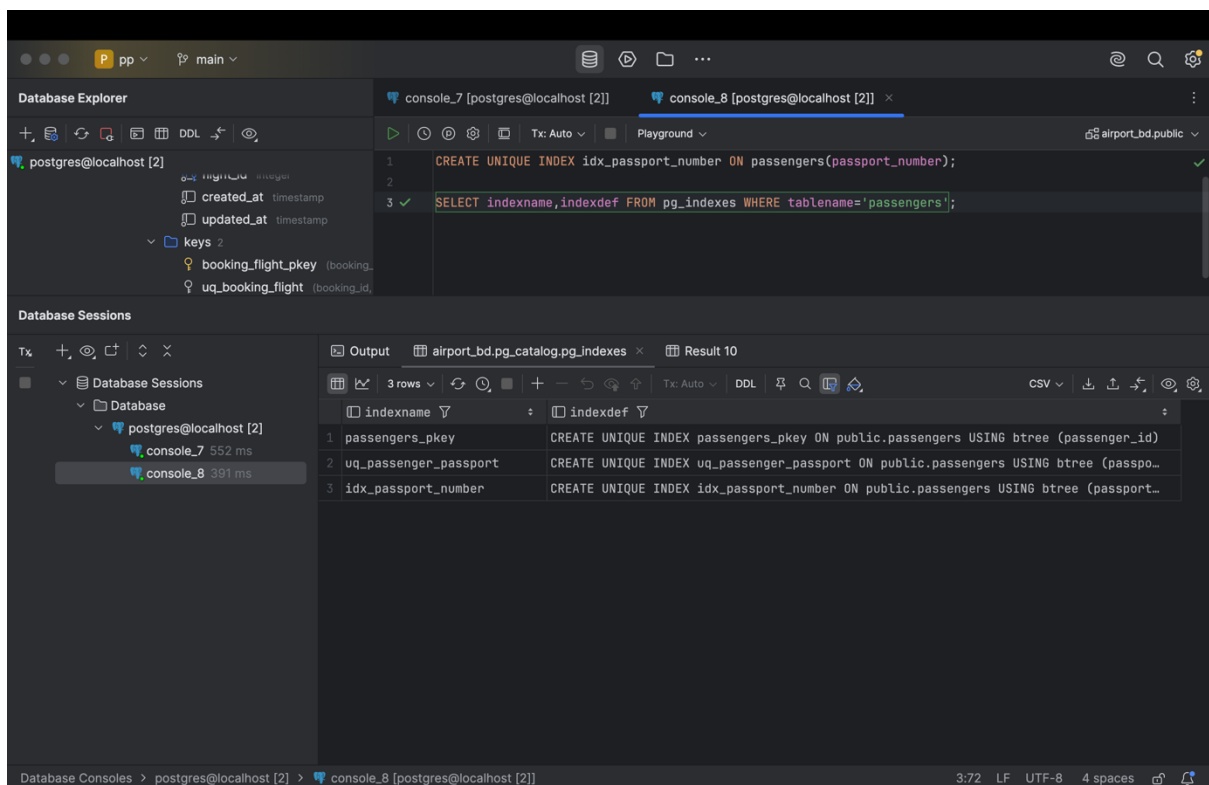
The 'Output' pane displays the query plan for 'Result 9-2':

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Actual Loops
1	Index Scan using idx_dep_act on flights	(cost=0.14..8.16)	rows=1	width=70	(actual time=0.407..0.408)	rows=0	loops=...
2	Index Cond: ((departing_airport_id = 1) AND (arriving_airport_id = 3))						
3	Planning Time: 0.150 ms						
4	Execution Time: 0.449 ms						

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



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. Explain in your own words what is going on in the output?



The image consists of two screenshots of a PostgreSQL playground interface, likely from a web browser. The top screenshot shows a successful SQL execution. The bottom screenshot shows an error message due to a duplicate key value.

Top Screenshot:

- SQL Editor:** Contains the following SQL queries:

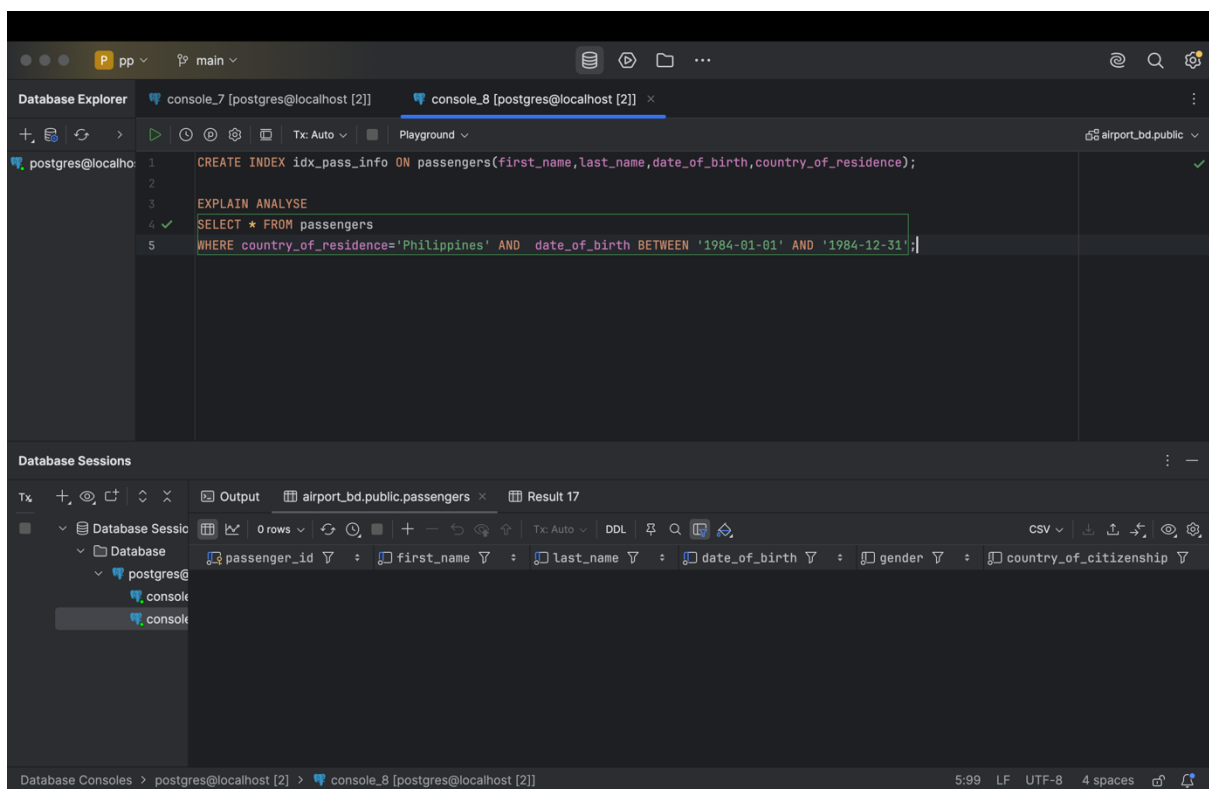
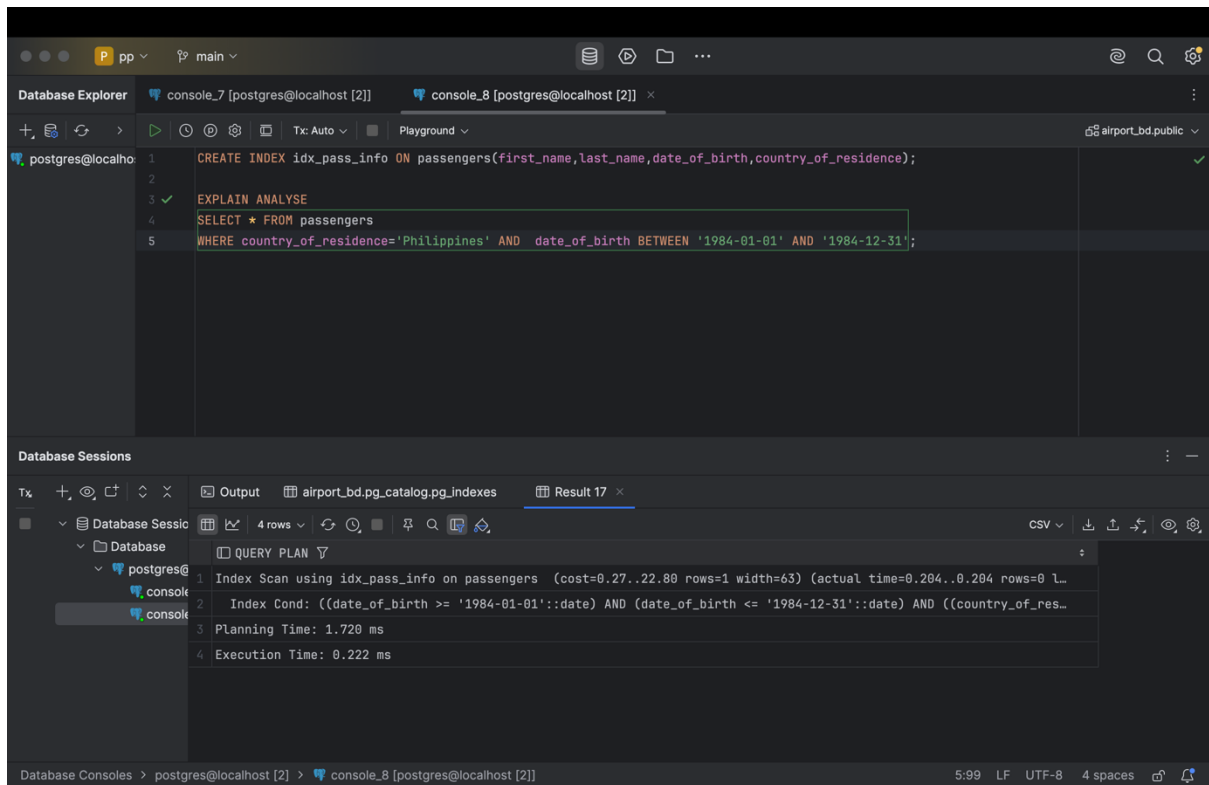
```
1 CREATE UNIQUE INDEX idx_passport_number ON passengers(passport_number);
2
3 SELECT indexname,indexdef FROM pg_indexes WHERE tablename='passengers';
4
5 INSERT INTO passengers (passenger_id, first_name, last_name, date_of_birth, gender,
6 country_of_citizenship, country_of_residence, passport_number, created_at, updated_at)
7 VALUES
8 (passenger_id 201, first_name 'Aziza', last_name 'Zhuman', date_of_birth '1999-03-05', gender 'F', country_of_citizenship 'Kazakhstan', country_of_residence 'Kazakhstan',
9 (passenger_id 202, first_name 'Bekturgan', last_name 'Dyussembaiuly', date_of_birth '2000-07-14', gender 'M', country_of_citizenship 'Kazakhstan', country_of_residence 'Kazakhstan',
```
- Database Sessions:** Shows a list of sessions for 'postgres@localhost [2]'. The active session is 'console_8' with a duration of 39 ms.
- Output:** Displays the results of the SQL queries:
 - Query 1: [2025-11-11 23:53:27] airport_bd.public> CREATE UNIQUE INDEX idx_passport_number ON passengers(passport_number) completed in 9 ms
 - Query 2: [2025-11-11 23:58:17] airport_bd.public> SELECT indexname,indexdef FROM pg_indexes WHERE tablename='passengers' 3 rows retrieved starting from 1 in 373 ms (execution: 17 ms, fetching: 356 ms)
 - Query 3: [2025-11-12 00:03:20] airport_bd.public> INSERT INTO passengers (passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence, passport_number, created_at, updated_at) VALUES (201, 'Aziza', 'Zhuman', '1999-03-05', 'F', 'Kazakhstan', 'Kazakhstan', '7. (202, 'Bekturgan', 'Dyussembaiuly', '2000-07-14', 'M', 'Kazakhstan', 'Kazakhstan', '7. 2 rows affected in 17 ms

Bottom Screenshot:

- SQL Editor:** The same SQL queries as the top screenshot, but the third query (INSERT) is highlighted with a red error icon.
- Database Sessions:** Shows a list of sessions for 'postgres@localhost [2]'. The active session is 'console_8' with a duration of 28 ms.
- Output:** Displays the results of the SQL queries, including an error message:
 - Query 1: [2025-11-12 00:03:20] 2 rows affected in 17 ms
 - Query 2: [2025-11-12 00:06:41] airport_bd.public> INSERT INTO passengers (passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence, passport_number, created_at, updated_at) VALUES (201, 'Aziza', 'Zhuman', '1999-03-05', 'F', 'Kazakhstan', 'Kazakhstan', '7. (202, 'Bekturgan', 'Dyussembaiuly', '2000-07-14', 'M', 'Kazakhstan', 'Kazakhstan', '7. 2 rows affected in 17 ms
 - Query 3: [2025-11-12 00:06:41] [23505] ERROR: duplicate key value violates unique constraint "passengers_pkey" [2025-11-12 00:06:41] Подробности: Key (passenger_id)=(201) already exists.

The index works — it doesn't allow duplication of passport numbers.

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.



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

pp main

Database Explorer console_7 [postgres@localhost [2]] console_8 [postgres@localhost [2]]

postgres@localhost

```
1 SELECT indexname, indexdef FROM pg_indexes
2 WHERE tablename='passengers';
3
4 DROP INDEX IF EXISTS idx_passport_number;
5 DROP INDEX IF EXISTS idx_pass_info;
```

Database Sessions

Output airport_bd.public.passengers airport_bd.catalog.pg_indexes

Database Sessic 4 rows Tx: Auto DDL

Database Consoles > postgres@localhost [2] > console_8 [postgres@localhost [2]]

5:36 LF UTF-8 4 spaces