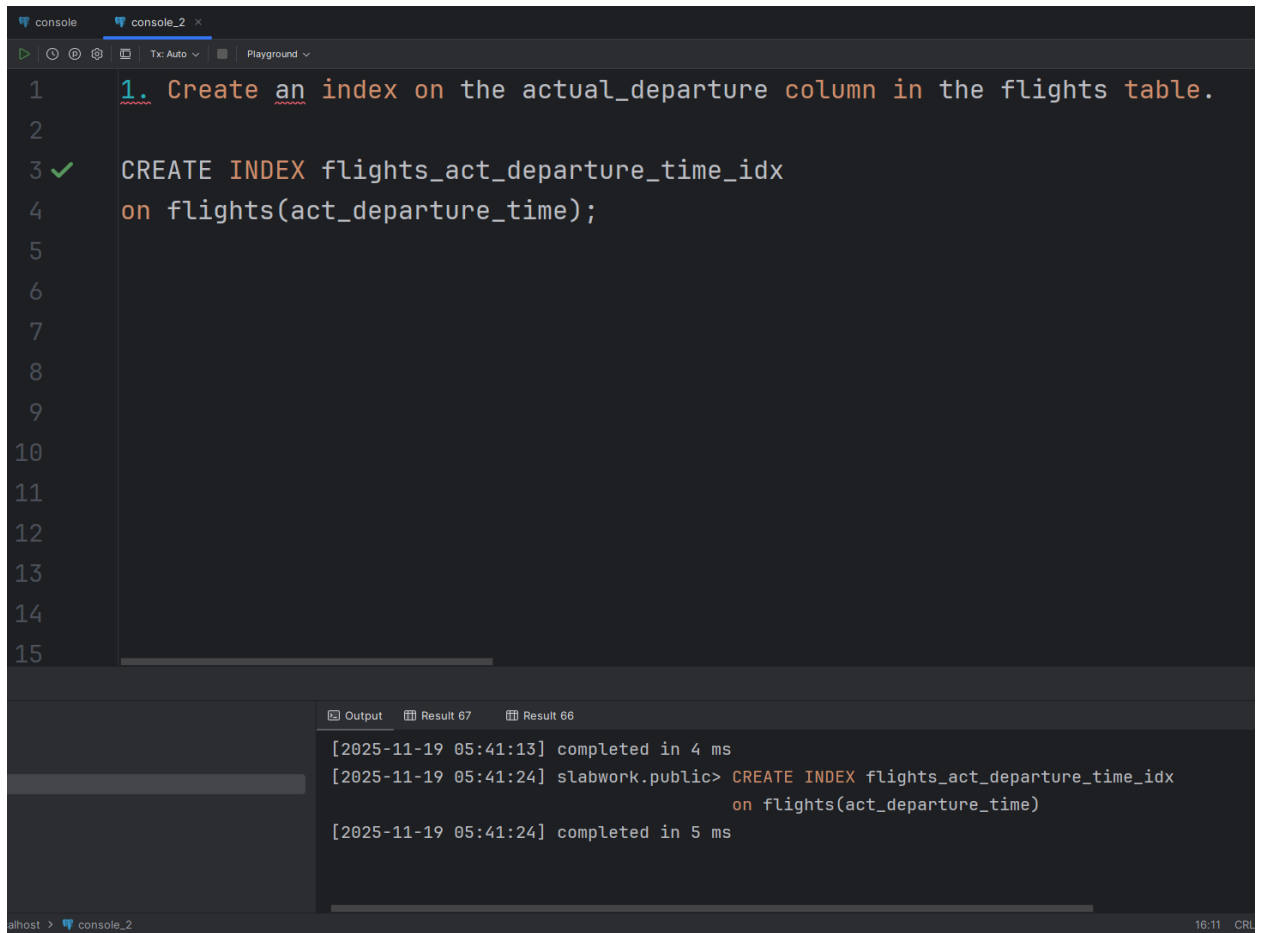


1)



The screenshot shows a SQL console interface with a dark theme. The top bar has tabs for 'console' and 'console_2'. Below the tabs are icons for running, refreshing, and other actions, along with a dropdown menu showing 'Tx: Auto' and 'Playground'. The main area contains a list of line numbers from 1 to 15 on the left. Line 1 has a blue instruction: '1. Create an index on the actual_departure column in the flights table.' Line 3 has a green checkmark icon and the SQL command: 'CREATE INDEX flights_act_departure_time_idx on flights(act_departure_time);'. The bottom panel shows the 'Output' tab with execution logs: '[2025-11-19 05:41:13] completed in 4 ms', '[2025-11-19 05:41:24] slabwork.public> CREATE INDEX flights_act_departure_time_idx on flights(act_departure_time)', and '[2025-11-19 05:41:24] completed in 5 ms'. The bottom status bar shows 'localhost > console_2' and a timestamp '16:11'.

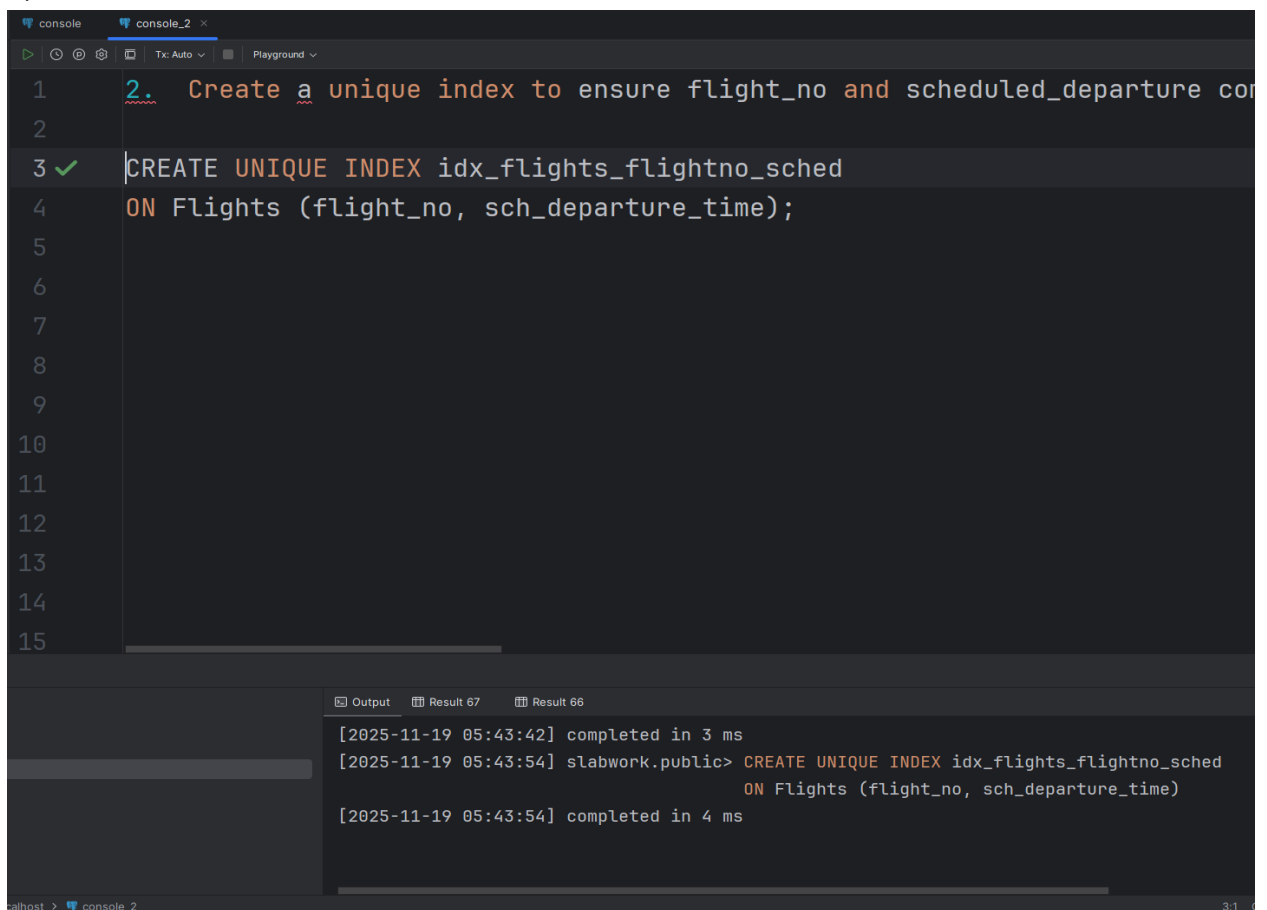
```
1 1. Create an index on the actual_departure column in the flights table.
2
3 ✓ CREATE INDEX flights_act_departure_time_idx
4   on flights(act_departure_time);
5
6
7
8
9
10
11
12
13
14
15
```

Output Result 67 Result 66

[2025-11-19 05:41:13] completed in 4 ms
[2025-11-19 05:41:24] slabwork.public> CREATE INDEX flights_act_departure_time_idx
on flights(act_departure_time)
[2025-11-19 05:41:24] completed in 5 ms

localhost > console_2 16:11 CRL

2)



The screenshot shows a SQL console interface with a dark theme. The top bar has tabs for 'console' and 'console_2'. Below the tabs are icons for running, refreshing, and other actions, along with a dropdown menu showing 'Tx: Auto' and 'Playground'. The main area contains a list of line numbers from 1 to 15 on the left. Line 1 has a blue instruction: '2. Create a unique index to ensure flight_no and scheduled_departure co'. Line 3 has a green checkmark icon and the SQL command: 'CREATE UNIQUE INDEX idx_flights_flightno_sched ON Flights (flight_no, sch_departure_time);'. The bottom panel shows the 'Output' tab with execution logs: '[2025-11-19 05:43:42] completed in 3 ms', '[2025-11-19 05:43:54] slabwork.public> CREATE UNIQUE INDEX idx_flights_flightno_sched ON Flights (flight_no, sch_departure_time)', and '[2025-11-19 05:43:54] completed in 4 ms'. The bottom status bar shows 'localhost > console_2' and a timestamp '3:1'.

```
1 2. Create a unique index to ensure flight_no and scheduled_departure co
2
3 ✓ CREATE UNIQUE INDEX idx_flights_flightno_sched
4   ON Flights (flight_no, sch_departure_time);
5
6
7
8
9
10
11
12
13
14
15
```

Output Result 67 Result 66

[2025-11-19 05:43:42] completed in 3 ms
[2025-11-19 05:43:54] slabwork.public> CREATE UNIQUE INDEX idx_flights_flightno_sched
ON Flights (flight_no, sch_departure_time)
[2025-11-19 05:43:54] completed in 4 ms

localhost > console_2 3:1

3)

```
console console_2 x
Tx: Auto Playground
1 3. Create a composite index on the departure_airport_id and arrival_airport_id
2
3 ✓ CREATE INDEX idx_flights_departure_arrival
4 ON Flights (departing_airport_id, arriving_airport_id);
5
6
7
8
9
10
11
12
13
14
15
```

```
ON Flights (flight_no, sch_departure_time)
[2025-11-19 05:43:54] completed in 4 ms
[2025-11-19 05:48:17] slabwork.public> CREATE INDEX idx_flights_departure_arrival
ON Flights (departing_airport_id, arriving_airport_id)
[2025-11-19 05:48:17] completed in 5 ms
```

localhost console_2 6:1 CRLF UTF-8

4)

```
console console_2 x
Tx: Auto Playground
1 4. Evaluate the difference in query performance with and without
2
3 ✓ EXPLAIN ANALYZE
4 SELECT *
5 FROM Passengers
6 WHERE country_of_citizenship = 'Philippines' AND
7 date_of_birth BETWEEN '1984-01-01' AND '2004-12-31';
8
9
10
11
12
13
14
15
```

```
Output Result 85 x
QUERY PLAN
1 Seq Scan on passengers (cost=0.00..6.50 rows=1 width=66) (actual time=0.033..0.033 rows=0 loops=1)
2 Filter: ((date_of_birth >= '1984-01-01'::date) AND (date_of_birth <= '2004-12-31'::date) AND ((country_of_citiz
3 Rows Removed by Filter: 200
4 Planning Time: 1.059 ms
5 Execution Time: 0.044 ms
```

5 rows

console console_2 x

Tx: Auto Playground

1 4. Evaluate the difference in query performance with and without index

2

3 ✓ CREATE INDEX idx_passengers_cs_dob ON passengers(country_of_citizenship, date_of_birth)

4

5

6

7

8

9

10

11

12

13

14 5. Use EXPLAIN ANALYZE to check index usage in a query filtering by country_of_citizenship and date_of_birth

15 6. Create a unique index for the passport_number of the Passengers table

Output Result 83

date_of_birth BETWEEN '1984-01-01' AND '2004-12-31';

[2025-11-19 05:58:29] 5 rows retrieved starting from 1 in 331 ms (execution: 3 ms, planning: 328 ms)

[2025-11-19 06:00:51] slabwork.public> CREATE INDEX idx_passengers_cs_dob ON passengers(country_of_citizenship, date_of_birth)

[2025-11-19 06:00:51] completed in 4 ms

console console_2 x

Tx: Auto Playground

1 --4. Evaluate the difference in query performance with and without indexes. Measure the execution time.

2

3 ✓ EXPLAIN ANALYZE

4 SELECT *

5 FROM Passengers

6 WHERE country_of_citizenship = 'Philippines' AND

7 date_of_birth BETWEEN '1984-01-01' AND '2004-12-31';

8

9

10

11

12

13

14

15

Output Result 113 x

QUERY PLAN

1 Index Scan using idx_passengers on passengers (cost=0.14..8.14 rows=1 width=66) (actual time=0.049..0.050 rows=0 loops=1)

2 Index Cond: (((country_of_citizenship)::text = 'Philippines'::text) AND (date_of_birth >= '1984-01-01'::date) AND (date_of_birth <= '2004-12-31'::date))

3 Planning Time: 0.148 ms

4 Execution Time: 0.063 ms

4 rows

5)

The screenshot shows a database console interface with two tabs: 'console' and 'console_2'. The 'console_2' tab is active, displaying a SQL query. The query is as follows:

```
--5. Use EXPLAIN ANALYZE to check index usage in a query

EXPLAIN ANALYZE
SELECT
    flight_id,
    departing_airport_id,
    arriving_airport_id,
    sch_departure_time,
    sch_arrival_time
FROM Flights
WHERE departing_airport_id = 3
    AND arriving_airport_id = 7;
```

Below the query, the 'QUERY PLAN' is displayed. It shows the following steps:

- 1 Bitmap Heap Scan on flights (cost=182.95..19271.12 rows=13124 width=28) (actual time=0.049..0.049 row
- 2 Recheck Cond: ((departing_airport_id = 3) AND (arriving_airport_id = 7))
- 3 -> Bitmap Index Scan on idx_fl_daiaai (cost=0.00..179.67 rows=13124 width=0) (actual time=0.039..0
- 4 Index Cond: ((departing_airport_id = 3) AND (arriving_airport_id = 7))
- 5 Planning Time: 1.283 ms
- 6 Execution Time: 0.077 ms

The bottom right of the console shows '6 rows' and a vertical ellipsis icon.

consoleconsole_2 x

12345678910111213141516

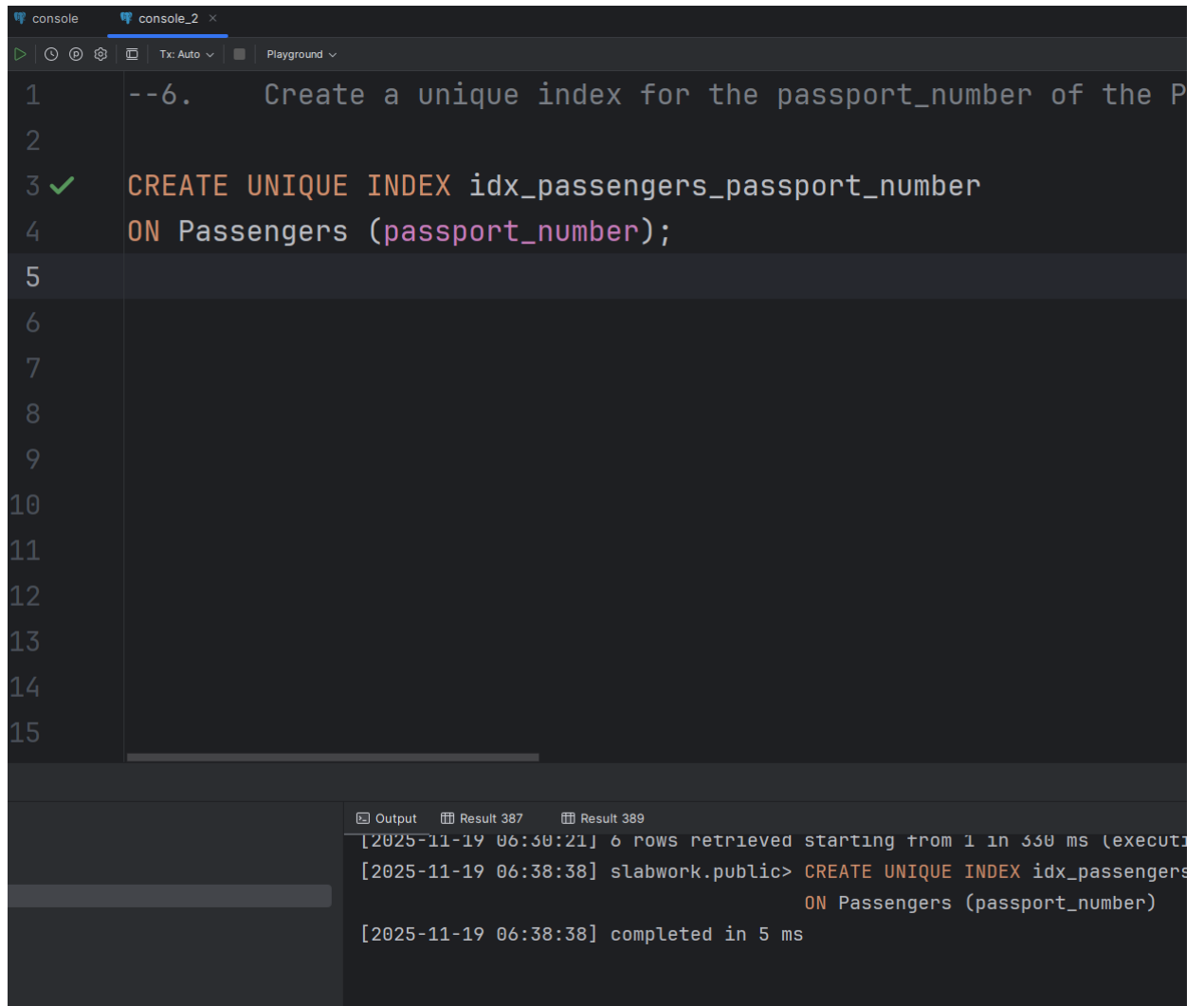
--5. Use EXPLAIN ANALYZE to check index usage in a q

CREATE INDEX idx_fl_daiaai on flights(departing_airport

OutputResult 387Result 389

WHERE departing_airport
AND arriving_airport_
[2025-11-19 06:27:57] 5 rows retrieved starting from 1 in 402
[2025-11-19 06:30:13] slabwork.public> CREate index idx_fl_dai
[2025-11-19 06:30:13] completed in 596 ms

6)



The screenshot shows a SQL console interface with a dark theme. The top bar has tabs for 'console' and 'console_2'. Below the tabs are icons for running, refreshing, and other actions, along with 'Tx: Auto' and 'Playground' dropdowns. The main editor area contains a SQL command: `--6. Create a unique index for the passport_number of the P` on line 1, followed by `CREATE UNIQUE INDEX idx_passengers_passport_number` on line 3 (marked with a green checkmark) and `ON Passengers (passport_number);` on line 4. The bottom panel shows the execution output with three tabs: 'Output', 'Result 387', and 'Result 389'. The 'Output' tab is active, displaying the following log messages: `[2025-11-19 06:30:21] 6 rows retrieved starting from 1 in 330 ms (executi`, `[2025-11-19 06:38:38] slabwork.public> CREATE UNIQUE INDEX idx_passengers`, `ON Passengers (passport_number)`, and `[2025-11-19 06:38:38] completed in 5 ms`.

```
--6. Create a unique index for the passport_number of the P
CREATE UNIQUE INDEX idx_passengers_passport_number
ON Passengers (passport_number);
```

Output Result 387 Result 389

```
[2025-11-19 06:30:21] 6 rows retrieved starting from 1 in 330 ms (executi
[2025-11-19 06:38:38] slabwork.public> CREATE UNIQUE INDEX idx_passengers
ON Passengers (passport_number)
[2025-11-19 06:38:38] completed in 5 ms
```

```
console console_2 x
Tx: Auto Playground
1 --6. Create a unique index for the passport_number of the Passen
2
3 ✓ INSERT INTO Passengers (
4     passenger_id, first_name, last_name, date_of_birth,
5     gender, country_of_citizenship, country_of_residence,
6     passport_number, created_at, updated_at
7 ) VALUES (
8     passenger_id 1001, first_name 'Test', last_name 'One', date_of_birth '
9     gender 'Male', country_of_citizenship 'USA', country_of_residence 'USA'
10    passport_number 'P999999999', created_at NOW(), updated_at NOW()
11 );
12
13
14
15
```

Output Result 387 Result 389

```
'Male', 'USA', 'USA',
'P999999999', NOW(), NOW()
)
[2025-11-19 06:39:15] 1 row affected in 4 ms
```

```
console console_2 x
Tx: Auto Playground
1 --6. Create a unique index for the passport_number of the Passengers table. Check
2
3 ! INSERT INTO Passengers (
4     passenger_id, first_name, last_name, date_of_birth,
5     gender, country_of_citizenship, country_of_residence,
6     passport_number, created_at, updated_at
7 ) VALUES (
8     passenger_id 1002, first_name 'Test', last_name 'Two', date_of_birth '1995-05-05',
9     gender 'Female', country_of_citizenship 'USA', country_of_residence 'USA',
10    passport_number 'P999999999', created_at NOW(), updated_at NOW()
11 );
12
13
14
```

[23505] ОШИБКА: повторяющееся значение ключа нарушает ограничение уникальности "idx_passengers_passport_number"
Подробности: Ключ "(passport_number)=(P999999999)" уже существует.

Output Result 387 Result 389

```
'P999999999', NOW(), NOW()
)
[23505] ОШИБКА: повторяющееся значение ключа нарушает ограничение уникальности "idx_passengers_passport_
Подробности: Ключ "(passport_number)=(P999999999)" уже существует.
```

7)


```
console console_2 x
Tx: Auto Playground
1 7. Create an index for the Passengers table. Use for that first
2
3 ✓ CREATE INDEX idx_passengers_country_dob_name
4 ON Passengers (
5     country_of_citizenship,
6     date_of_birth,
7     last_name,
8     first_name
9 );
10
11
12
13
14
15
```

Output Result 387 Result 389

```
last_name,
first_name
)
completed in 5 ms
```

```
console console_2 x
Tx: Auto Playground
1 --7. Create an index for the Passengers table. Use for that first name, last name,
2
3 ✓ EXPLAIN ANALYZE
4 SELECT
5     passenger_id,
6     first_name,
7     last_name,
8     date_of_birth,
9     country_of_citizenship
10 FROM Passengers
11 WHERE country_of_citizenship = 'Philippines'
12     AND date_of_birth BETWEEN DATE '1984-01-01' AND DATE '1985-01-01';
13
14
15
```

Output Result 394 x Result 389

QUERY PLAN

```
1 Index Scan using idx_passengers_dob on passengers (cost=0.14..8.11 rows=1 width=27) (actual time=0.004..0.004 rows=0 loops=1)
2 Index Cond: ((date_of_birth >= '1984-01-01'::date) AND (date_of_birth <= '1985-01-01'::date) AND ((country_of_citizenship)::text = 'Philippines'::text))
3 Planning Time: 1.170 ms
4 Execution Time: 0.017 ms
```

4 rows

8)

console console_2 x

Tx: Auto Playground

```
1 8. Write a SQL query to list indexes for table Passengers
2 SELECT
3     indexname,
4     indexdef
5 FROM pg_indexes
6 WHERE tablename = 'passengers';
7
8
9
10
11
12
13
14
```

Output slabwork.pg_catalog.pg_indexes x Result 396

Tx: Auto DDL

indexname	indexdef
passengers_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USI...
idx_passengers_cs_dob	CREATE INDEX idx_passengers_cs_dob ON public.passengers USIN...
idx_passengers	CREATE INDEX idx_passengers ON public.passengers USING btree...
idx_passengers_dob	CREATE INDEX idx_passengers_dob ON public.passengers USING b...
idx_passengers_passport_number	CREATE UNIQUE INDEX idx_passengers_passport_number ON public...
idx_passengers_country_dob_name	CREATE INDEX idx_passengers_country_dob_name ON public.passe...

6 rows

consoleconsole_2 x

Tx: AutoPlayground v

1

--8. Write a SQL query to list indexes for table Passengers.

2

3 ✓

DROP INDEX

4

idx_passengers_cs_dob,

5

idx_passengers,

6

idx_passengers_dob,

7

idx_passengers_passport_number,

8

idx_passengers_country_dob_name;

9

10

11

12

13

14

Outputslabwork.pg_catalog.pg_indexesResult 396

idx_passengers_cs_dob,
idx_passengers,
idx_passengers_dob,
idx_passengers_passport_number,
idx_passengers_country_dob_name

completed in 6 ms