

Laboratory work 7

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

The screenshot shows the DataGrip IDE interface. In the top right, there are tabs for 'main' and 'pp'. Below them, the 'Database Explorer' shows a connection to 'postgres@localhost [2]' with a schema named 'high5'. Under 'high5', there are tables 'flights' and 'keys'. The 'flights' table has columns: id (int), flight_no (text), sch_departure_time (timestamp), and actual_departure (timestamp). The 'keys' folder contains two entries: 'created_at timestamp' and 'updated_at timestamp'. In the center, the 'Playground' tab is active, displaying the SQL command: 'CREATE INDEX idx_actual_departure ON flights(sch_departure_time);'. Below the playground, the 'Database Sessions' panel shows a single session: 'postgres@localhost [2]'. The session details show a connection to 'airport_bd' at [2025-11-11 23:28:01]. A query was run: 'CREATE INDEX idx_actual_departure ON flights(sch_departure_time)'. The execution completed in 30 ms. At the bottom, the status bar indicates the file is saved ('main'), the encoding is UTF-8, and the code has 4 spaces per indentation level.

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

The screenshot shows the DataGrip IDE interface. On the left, the Database Explorer panel displays the schema of the 'flights' table, which includes columns like 'flight_id' (integer), 'created_at' (timestamp), 'updated_at' (timestamp), and several indexes and foreign keys. In the center-right, the main editor window shows a SQL command: 'CREATE UNIQUE INDEX uq_flight_id_sch_departure_time ON flights(flight_id,sch_departure_time);'. Below the editor, the Database Sessions panel shows two active sessions: 'console_7' (552 ms) and 'console_8' (27 ms). The status bar at the bottom indicates the date and time as '2025-11-11 23:28:01'.

3. Create a composite index on the departure_airport_id and arrival_airport_id columns.

The screenshot shows the DataGrip IDE interface. Similar to the previous screenshot, it displays the Database Explorer with the 'flights' table schema. The main editor window now contains a different SQL command: 'CREATE INDEX idx_dep_act ON flights(departing_airport_id,arriving_airport_id);'. The Database Sessions panel shows the same two sessions ('console_7' and 'console_8'). The status bar at the bottom indicates the date and time as '2025-11-11 23:33:38'.

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

Without:

The screenshot shows the pgAdmin interface with two database consoles. The top console, 'console_8', displays the query EXPLAIN ANALYSE SELECT * FROM flights WHERE departing_airport_id = 1 AND arriving_airport_id = 3;. The output shows a sequential scan on the 'flights' table. The bottom console, 'console_7', shows the execution time for this query.

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

```
Seq Scan on flights  (cost=0.00..7.00 rows=1 width=70) (actual time=0.044..0.044 rows=0 loops=1)
  Filter: ((departing_airport_id = 1) AND (arriving_airport_id = 3))
  Rows Removed by Filter: 200
Planning Time: 1.700 ms
Execution Time: 0.095 ms
```

With:

The screenshot shows the pgAdmin interface with two database consoles. The top console, 'console_8', displays the query SET enable_seqscan = off; followed by EXPLAIN ANALYSE SELECT * FROM flights WHERE departing_airport_id = 1 AND arriving_airport_id = 3;. The output shows an index scan using an index named idx_dep_act. The bottom console, 'console_7', shows the execution time for this query.

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

```
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=1)
  Index Cond: ((departing_airport_id = 1) AND (arriving_airport_id = 3))
  Planning Time: 0.150 ms
  Execution Time: 0.449 ms
```

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

The screenshot shows the pgAdmin interface with two database consoles. The left console (postgres@localhost [2]) displays the schema for the 'flights' table, which includes columns like flight_id, created_at, and updated_at. The right console (console_8) contains the following SQL code:

```
EXPLAIN ANALYZE SELECT * FROM flights
WHERE departing_airport_id = 4 AND arriving_airport_id = 46;
```

The results pane shows the execution plan and timing details:

- Index Scan using idx_dep_act on flights (cost=0.14..8.16 rows=1 width=70) (actual time=0.098...)
- Index Cond: ((departing_airport_id = 4) AND (arriving_airport_id = 46))
- Planning Time: 0.252 ms
- Execution Time: 0.145 ms

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 screenshot shows the pgAdmin interface with two database consoles. The left console (postgres@localhost [2]) displays the schema for the 'passenger' table, which includes columns like passenger_id, created_at, and updated_at. The right console (console_8) contains the following SQL code:

```
CREATE UNIQUE INDEX idx_passport_number ON passengers(passport_number);
SELECT indexname, indexdef FROM pg_indexes WHERE tablename='passenger';
```

The results pane shows the output of the second query, listing three indexes:

indexname	indexdef
passenger_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passenger USING btree (passenger_id)
uq_passenger_passport	CREATE UNIQUE INDEX uq_passenger_passport ON public.passenger USING btree (passport_number)
idx_passport_number	CREATE UNIQUE INDEX idx_passport_number ON public.passenger USING btree (passport_number)

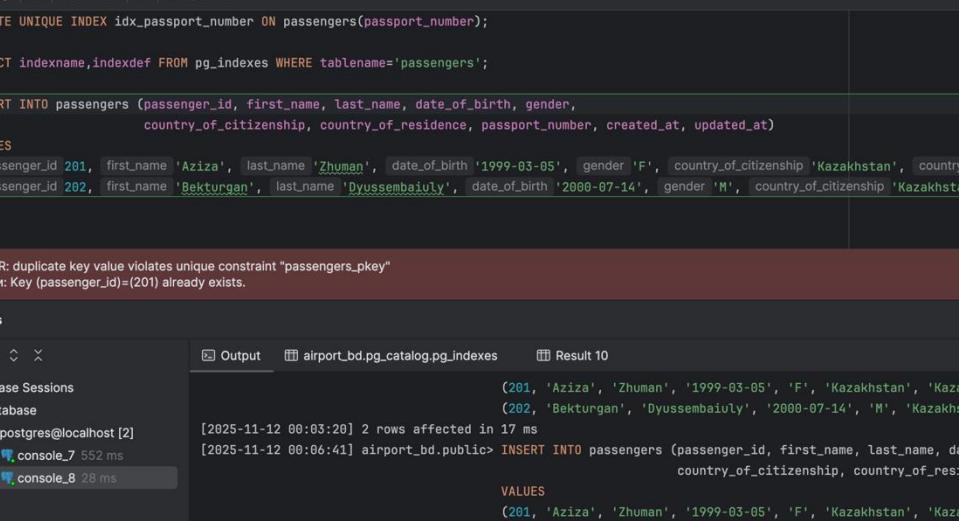
The screenshot shows the pgAdmin 4 interface with two database consoles open:

- console_7 [postgres@localhost [2]]**: Contains the following SQL code:

```
CREATE UNIQUE INDEX idx_passport_number ON passengers(passport_number);
SELECT indexname,indexdef FROM pg_indexes WHERE tablename='passengers';
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
( 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',
( passenger_id 202, first_name 'Bekturgan', last_name 'Dyussembeuly', date_of_birth '2000-07-14', gender 'M', country_of_citizenship 'Kazakhstan', country_of_residence 'Kazakhstan')
```
- console_8 [postgres@localhost [2]]**: Shows the output of the executed commands:

```
[2025-11-11 23:53:27] airport_bd.public> CREATE UNIQUE INDEX idx_passport_number ON passengers(passport_number)
[2025-11-11 23:53:27] completed in 9 ms
[2025-11-11 23:58:17] airport_bd.public> SELECT indexname,indexdef FROM pg_indexes WHERE tablename='passengers'
[2025-11-11 23:58:18] 3 rows retrieved starting from 1 in 373 ms (execution: 17 ms, fetching: 356 ms)
[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, pass
VALUES
(201, 'Aziza', 'Zhuman', '1999-03-05', 'F', 'Kazakhstan', 'Kazakhstan', '7-2025-11-12 00:03:20)
(202, 'Bekturgan', 'Dyussembeuly', '2000-07-14', 'M', 'Kazakhstan', 'Kazakhstan', '7-2025-11-12 00:03:20)
[2025-11-12 00:03:20] 2 rows affected in 17 ms
```

The bottom left pane shows the "Database Sessions" tree, which includes "Database Sessions", "Database", and two sessions under "postgres@localhost [2]": "console_7" (552 ms) and "console_8" (39 ms). The bottom right pane shows the "Output" tab with the command history and results.



```
CREATE UNIQUE INDEX idx_passport_number ON passengers(passport_number);

SELECT indexname,indexdef FROM pg_indexes WHERE tablename='passengers';

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
( 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'
( passenger_id 202, first_name 'Bekturgen', last_name 'Dyussembeiluly', date_of_birth '2000-07-14', gender 'M', country_of_citizenship 'Kazakhstan', country_of_residence 'Kazakhstan'

[23505] ERROR: duplicate key value violates unique constraint "passengers_pkey"
Подробности: Key (passenger_id)=(201) already exists.
```

Database Sessions

Tx	Output	Result 10
Database Sessions	airport_bd.pg_catalog.pg_indexes	(201, 'Aziza', 'Zhuman', '1999-03-05', 'F', 'Kazakhstan', 'Kazakhstan', '202, 'Bekturgen', 'Dyussembeiluly', '2000-07-14', 'M', 'Kazakhstan', 'Kazakhstan')
postgres@localhost [2]		[2025-11-12 00:03:20] 2 rows affected in 17 ms
console_7		[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, pass
console_8		VALUES (201, 'Aziza', 'Zhuman', '1999-03-05', 'F', 'Kazakhstan', 'Kazakhstan', '202, 'Bekturgen', 'Dyussembeiluly', '2000-07-14', 'M', 'Kazakhstan', 'Kazakhstan') [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.

The screenshot shows a PostgreSQL database interface. In the top-left pane, the command `CREATE INDEX idx_pass_info ON passengers(first_name, last_name, date_of_birth, country_of_residence);` is run. Below it, the command `EXPLAIN ANALYSE` is run followed by a query: `SELECT * FROM passengers WHERE country_of_residence='Philippines' AND date_of_birth BETWEEN '1984-01-01' AND '1984-12-31';`. The results show an Index Scan using the created index on the `passengers` table.

```
CREATE INDEX idx_pass_info ON passengers(first_name, last_name, date_of_birth, country_of_residence);
EXPLAIN ANALYSE
SELECT * FROM passengers
WHERE country_of_residence='Philippines' AND date_of_birth BETWEEN '1984-01-01' AND '1984-12-31';
```

The screenshot shows a PostgreSQL database interface. In the top-left pane, the command `CREATE INDEX idx_pass_info ON passengers(first_name, last_name, date_of_birth, country_of_residence);` is run. Below it, the command `EXPLAIN ANALYSE` is run followed by a query: `SELECT * FROM passengers WHERE country_of_residence='Philippines' AND date_of_birth BETWEEN '1984-01-01' AND '1984-12-31';`. The results show an Index Scan using the created index on the `passenger` table.

```
CREATE INDEX idx_pass_info ON passengers(first_name, last_name, date_of_birth, country_of_residence);
EXPLAIN ANALYSE
SELECT * FROM passengers
WHERE country_of_residence='Philippines' AND date_of_birth BETWEEN '1984-01-01' AND '1984-12-31';
```

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

The screenshot shows a PostgreSQL database management interface with the following details:

- Database Explorer:** A sidebar on the left showing the database structure:
 - Postgres@localhost
 - Tables: airport, lab2_bk, phonet, passengers, postgr, schema, Data, ranks.
- Query Editor:** The main area contains a code editor with the following SQL script:

```
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:** A bottom panel showing the current session details:
 - Database: postgres@localhost [2]
 - Session: console_8 [postgres@localhost [2]]
 - Output tab (selected)
 - Time: 5:36
 - Encoding: UTF-8
 - Format: 4 spaces