

Kazakh-British Technical University

Databases

Laboratory work №2

Kislitsin Alexandr

23B031861

The purpose of the work

Creating tables with keys and relationships;

Filling the database with data;

Performing SELECT queries for information analysis;

Development of filtering, sorting, grouping and aggregate functions.

Task 1

All the data of passengers whose last name is same as first name.

The screenshot displays a database management tool interface with the following components:

- Database Explorer:** Shows a tree view of the database structure. The 'lab_db' database is selected, showing tables: 'Airline', 'Airport', 'Baggage', 'Booking_flight', 'Booking_pass', 'Booking_ticket', 'Passengers', and 'Related_data'. The 'Airline' table is expanded, showing columns: 'airline_id' (int, auto-increment), 'airline_code' (varchar(30)), 'airline_name' (varchar(50)), 'airline_country' (varchar(50)), 'created_at' (timestamp), and 'updated_at' (timestamp).
- Console:** Contains a list of SQL queries. The selected query is:

```
DELETE FROM Flights
WHERE YEAR(sch_arrival_time) = 2024;

UPDATE booking
SET ticket_price = ticket_price * 1.15
WHERE ticket_price > 0;

DELETE FROM Booking
WHERE ticket_price < 10000;

SELECT *
FROM Passengers
WHERE first_name = last_name;
```
- Output:** Displays the results of the selected query. The table has 5 columns: 'passenger_id', 'first_name', 'last_name', 'date_of_birth', 'gender', and 'country_of_origin'. The results are currently empty, showing '0 rows'.
- Services:** A list of database services and their execution times: 'Booking_flight' (358 ms), 'Booking' (363 ms), 'Baggage_check' (366 ms), 'Airline' (352 ms), 'Boarding_pass' (348 ms), and 'Baggage' (340 ms).

Task 2

The last name of all passangers without duplicates.

The screenshot displays a database management interface with the following components:

- Database Explorer:** Shows the database structure for `@localhost` and `lab_db`. The `lab_db` database contains tables: `airline`, `airport`, `baggage`, `flights`, and `passengers`. The `airline` table has columns: `airline_id` (int, auto-increment), `airline_code` (varchar(30)), `airline_name` (varchar(50)), `airline_country` (varchar(50)), `created_at` (timestamp), and `updated_at` (timestamp).
- Console:** Contains the following SQL queries:

```
192 UPDATE booking
193 SET ticket_price = ticket_price * 1.15
194 WHERE ticket_price > 0;
195
196 DELETE FROM Booking
197 WHERE ticket_price < 10000;
198
199 SELECT *
200 FROM Passengers
201 WHERE first_name = last_name;
202
203 SELECT DISTINCT last_name
204 FROM Passengers;
205
206 SELECT *
207 FROM Passengers
208 WHERE gender = 'Male'
209 AND date_of_birth BETWEEN '1990-01-01' AND '2000-12-31';
```
- Services:** A list of database services and their execution times:
 - Database: 403 ms
 - Booking_flight: 358 ms
 - Booking: 363 ms
 - Baggage_check: 366 ms
 - Airline: 352 ms
 - Boarding_pass: 348 ms
 - Baggage: 340 ms
 - Flights: 338 ms
 - Airport: 394 ms
- Output:** A table showing the results of the SQL queries. The first query returns the last names of passengers where the first name equals the last name. The second query returns the last names of male passengers born between 1990 and 2000.

last_name
Silva
Karinova
Wilson
Bekova
Fischer
Yamamoto
Romanov
Murnhv

Task 3

All male passengers born between 1990 and 2000.

The screenshot displays a database management interface with the following components:

- Database Explorer:** Shows a local database named 'lab_db' with tables 'Airline', 'Airport', 'Baggage', and 'Flights'. The 'Airline' table structure is visible, including columns like 'airline_id', 'airline_code', 'airline_name', 'airline_country', 'created_at', and 'updated_at'.
- Console:** Contains a SQL script with the following queries:

```
DELETE FROM Booking
WHERE ticket_price < 10000;

SELECT *
FROM Passengers
WHERE first_name = last_name;

SELECT DISTINCT last_name
FROM Passengers;

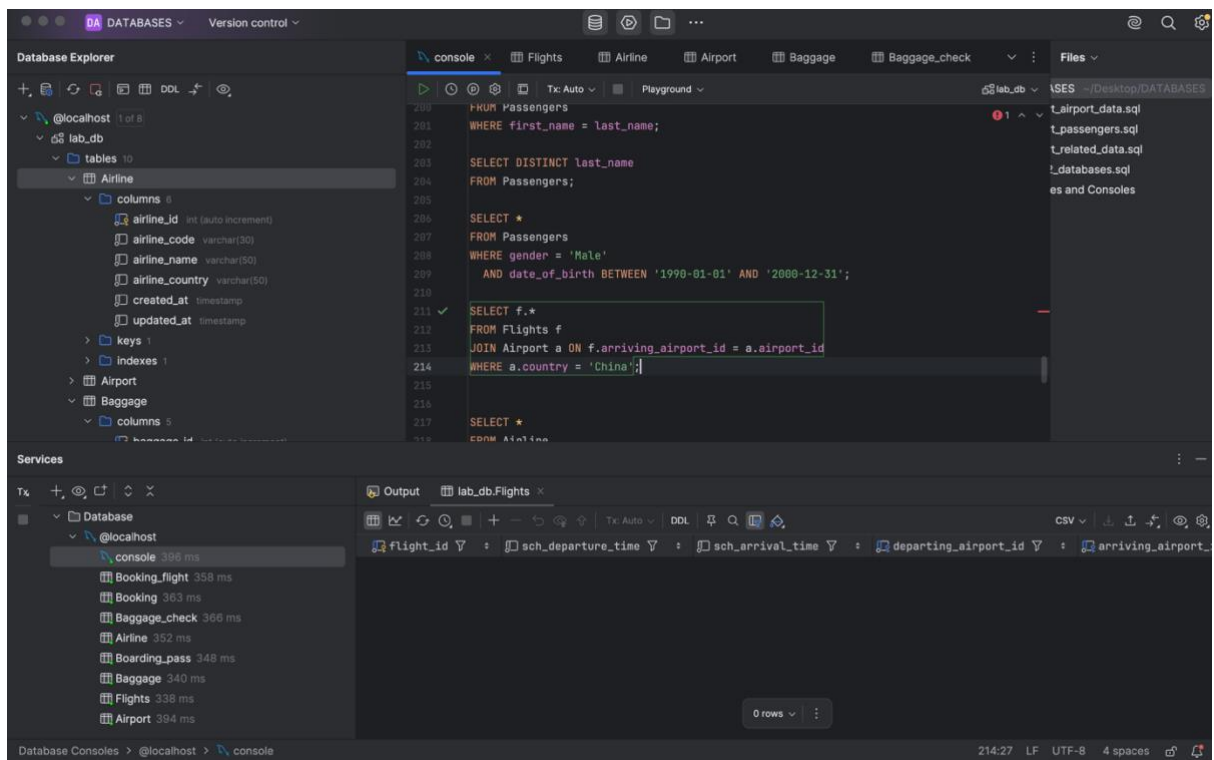
SELECT *
FROM Passengers
WHERE gender = 'Male'
AND date_of_birth BETWEEN '1990-01-01' AND '2000-12-31';

SELECT f.*
FROM Flights f
JOIN Airport a ON f.arriving_airport_id = a.airport_id
```
- Output:** Displays the results of the query filtering for male passengers born between 1990 and 2000. The results are shown in a table with 6 columns: passenger_id, first_name, last_name, date_of_birth, gender, and country_of_citizen.

passenger_id	first_name	last_name	date_of_birth	gender	country_of_citizen
194	George	Karimova	1993-01-02	Male	Australia
203	Elena	Ali	1999-05-30	Male	China
214	Mei	Li	1996-02-10	Male	Egypt
215	Askar	Kuznetsov	1991-06-27	Male	Japan
225	Sanzhar	Murphy	1995-07-10	Male	Qatar
249	Aruzhan	Tanaka	1994-11-22	Male	Egypt
253	Murat	Anderson	1990-10-14	Male	Turkey

Task 4

Create a query that shows all flights flying to 'China'.



Task 5

Airlines from any of: ('France','Portugal','Poland') created between '2023-11-01' and '2024-03-31'.

The screenshot displays a database management tool interface with the following components:

- Database Explorer:** Shows a tree view of the database structure. The 'Airline' table is selected, showing columns: `airline_id` (int, auto-increment), `airline_code` (varchar(30)), `airline_name` (varchar(50)), `airline_country` (varchar(50)), `created_at` (timestamp), and `updated_at` (timestamp).
- Console:** Contains a SQL query with line numbers 208 to 225. The query is:

```
208 WHERE gender = 'Male'
209 AND date_of_birth BETWEEN '1990-01-01' AND '2000-12-31';
210
211 SELECT f.*
212 FROM Flights f
213 JOIN Airport a ON f.arriving_airport_id = a.airport_id
214 WHERE a.country = 'China';
215
216
217 ✓ SELECT *
218 FROM Airline
219 WHERE airline_country IN ('France','Portugal','Poland')
220 AND created_at BETWEEN '2023-11-01' AND '2024-03-31';
221
222 SELECT airline_name
223 FROM Airline
224 WHERE airline_country = 'Kazakhstan';
225
```
- Services:** A list of database services and their execution times:
 - console: 358 ms
 - Booking_flight: 358 ms
 - Booking: 363 ms
 - Baggage_check: 366 ms
 - Airline: 352 ms
 - Boarding_pass: 348 ms
 - Baggage: 340 ms
 - Flights: 338 ms
 - Airport: 394 ms
- Output:** A table showing the results of the query. The columns are: `airline_id`, `airline_code`, `airline_name`, `airline_country`, `created_at`, and `updated_at`. The table is currently empty, showing 0 rows.

Task 6

All airline names based in Kazakhstan.

The screenshot displays a database management tool interface with the following components:

- Database Explorer:** Shows the database structure for `lab_db`. The `Airline` table is expanded, showing columns: `airline_id` (int, auto increment), `airline_code` (varchar(30)), `airline_name` (varchar(50)), `airline_country` (varchar(50)), `created_at` (timestamp), and `updated_at` (timestamp). It also shows keys and indexes.
- Console:** Contains SQL queries:

```
211 SELECT f.*
212 FROM Flights f
213 JOIN Airport a ON f.arriving_airport_id = a.airport_id
214 WHERE a.country = 'China';
215
216
217 SELECT *
218 FROM Airline
219 WHERE airline_country IN ('France','Portugal','Poland')
220 AND created_at BETWEEN '2023-11-01' AND '2024-03-31';
221
222 ✓ SELECT airline_name
223 FROM Airline
224 WHERE airline_country = 'Kazakhstan';
225
226
227 UPDATE Booking
228 SET ticket_price = ticket_price * 0.9
229 WHERE created_at < '2023-11-01';
```
- Services:** A list of database services and their execution times:
 - console: 388 ms
 - Booking_flight: 358 ms
 - Booking: 363 ms
 - Baggage_check: 366 ms
 - Airline: 352 ms
 - Boarding_pass: 348 ms
 - Baggage: 340 ms
 - Flights: 338 ms
 - Airport: 394 ms
- Output:** Shows the result of the query in line 224, displaying the column `airline_name`.
- Files:** A list of files in the project, including `t_airport_data.sql`, `t_passengers.sql`, `t_related_data.sql`, `t_databases.sql`, and `es and Consoles`.

Task 7

Reduce the cost of booking price by 10% created before '11-01-2023'.

The screenshot displays a database management interface with the following components:

- Database Explorer:** Shows the database structure for 'lab_db', including tables like 'Airline', 'Airport', 'Baggage', 'Flights', 'Passengers', and 'Booking'. The 'Airline' table is selected, showing columns: 'airline_id' (int, auto-increment), 'airline_code' (varchar(30)), 'airline_name' (varchar(50)), 'airline_country' (varchar(50)), 'created_at' (timestamp), and 'updated_at' (timestamp).
- Console:** Contains the following SQL queries:

```
220 AND created_at BETWEEN '2023-11-01' AND '2024-03-31';
221
222 SELECT airline_name
223 FROM Airline
224 WHERE airline_country = 'Kazakhstan';
225
226
227 ✓ UPDATE Booking
228 SET ticket_price = ticket_price * 0.9
229 WHERE created_at < '2023-11-01';
230
231
232 SELECT * FROM Passengers;
233
234 SELECT *
235 FROM Baggage
236 WHERE weight_in_kg > 25
237 ORDER BY weight_in_kg DESC
238 LIMIT 3;
```
- Services:** A list of database services and their execution times:
 - Database
 - @localhost
 - console 65 ms
 - Booking_flight 358 ms
 - Booking 363 ms
 - Baggage_check 366 ms
 - Airline 352 ms
 - Boarding_pass 348 ms
 - Baggage 340 ms
 - Flights 338 ms
 - Airport 394 ms
- Output:** Shows the execution results of the queries:

```
WHERE airline_country IN ('France', 'Portugal', 'Poland')
AND created_at BETWEEN '2023-11-01' AND '2024-03-31'
[2025-09-30 12:25:06] 0 rows retrieved in 357 ms (execution: 12 ms, fetching: 345 ms)
[2025-09-30 12:25:42] lab_db> SELECT airline_name
FROM Airline
WHERE airline_country = 'Kazakhstan'
[2025-09-30 12:25:42] 0 rows retrieved in 354 ms (execution: 4 ms, fetching: 350 ms)
[2025-09-30 12:26:33] lab_db> UPDATE Booking
SET ticket_price = ticket_price * 0.9
WHERE created_at < '2023-11-01'
[2025-09-30 12:26:33] completed in 33 ms
```

Task 8

Find top3 overweighted baggage with more than 25kg.

The screenshot displays a database management tool interface with the following components:

- Database Explorer:** Shows a tree view of the database structure. The 'Airline' table is selected, showing columns: `airline_id` (int, auto increment), `airline_code` (varchar(30)), `airline_name` (varchar(50)), `airline_country` (varchar(50)), `created_at` (timestamp), and `updated_at` (timestamp).
- Console:** Contains the following SQL queries:

```
UPDATE Booking
SET ticket_price = ticket_price * 0.9
WHERE created_at < '2023-11-01';

SELECT * FROM Passengers;

SELECT *
FROM Baggage
WHERE weight_in_kg > 25
ORDER BY weight_in_kg DESC
LIMIT 3;

SELECT first_name, last_name
FROM Passengers
ORDER BY date of birth DESC;
```
- Services:** A list of database services with their execution times:
 - console: 389 ms
 - Booking_flight: 358 ms
 - Booking: 363 ms
 - Baggage_check: 366 ms
 - Airline: 352 ms
 - Boarding_pass: 348 ms
 - Baggage: 340 ms
 - Flights: 338 ms
 - Airport: 394 ms
- Output:** A table titled 'lab.db.Baggage' showing the results of the third query. The table has columns: `baggage_id`, `weight_in_kg`, `created_at`, `updated_at`, and `booking_id`. The table is currently empty, showing '0 rows'.

Task 9

Find the youngest passengers' full name.

The screenshot shows a database management interface with the following components:

- Database Explorer:** Displays the database structure for 'lab_db', including tables like 'Airline', 'Airport', 'Baggage', and 'Passengers'.
- Console:** Contains the following SQL queries:

```
232 SELECT * FROM Passengers;  
233  
234 SELECT *  
235 FROM Baggage  
236 WHERE weight_in_kg > 25  
237 ORDER BY weight_in_kg DESC  
238 LIMIT 3;  
239  
240  
241 ✓ SELECT first_name, last_name  
242 FROM Passengers  
243 ORDER BY date_of_birth DESC  
244 LIMIT 1;  
245  
246 SELECT booking_platform, MIN(ticket_price) AS cheapest_price  
247 FROM Booking  
248 GROUP BY booking_platform;  
249  
250 SELECT *
```
- Output:** Displays the results of the last query (line 241), showing the first name and last name of the youngest passenger:

first_name	last_name
Victor	Martin
- Services:** A list of database services and their execution times, including 'console' (395 ms), 'Booking_flight' (358 ms), 'Booking' (363 ms), 'Baggage_check' (366 ms), 'Airline' (352 ms), 'Boarding_pass' (348 ms), 'Baggage' (340 ms), 'Flights' (338 ms), and 'Airport' (394 ms).

Task 10

Find the cheapest booking price on each booking platform.

The screenshot shows a database management tool interface with the following components:

- Database Explorer:** Displays the database structure for 'lab_db' on 'localhost'. It shows tables: Airline, Airport, Baggage, and Baggage_check. The 'Airline' table is expanded, showing columns: airline_id (int, auto-increment), airline_code (varchar(30)), airline_name (varchar(50)), airline_country (varchar(50)), created_at (timestamp), and updated_at (timestamp).
- Console:** Contains a SQL query:

```
236 WHERE weight_in_kg > 25
237 ORDER BY weight_in_kg DESC
238 LIMIT 3;

241 SELECT first_name, last_name
242 FROM Passengers
243 ORDER BY date_of_birth DESC
244 LIMIT 1;

246 SELECT booking_platform, MIN(ticket_price) AS cheapest_price
247 FROM Booking
248 GROUP BY booking_platform;

250 SELECT *
251 FROM Airline
252 WHERE airline_code REGEXP '[0-9]';
```
- Services:** A list of database services and their execution times:
 - console: 394 ms
 - Booking_flight: 358 ms
 - Booking: 363 ms
 - Baggage_check: 366 ms
 - Airline: 352 ms
 - Boarding_pass: 348 ms
 - Baggage: 340 ms
 - Flights: 338 ms
 - Airport: 394 ms
- Output:** Displays the result of the SQL query in a table with two columns: 'booking_platform' and 'cheapest_price'. The result is:

booking_platform	cheapest_price
Online	17184.00

Task 11

Return airlines whose airline_code contains a digit.

The screenshot shows a database IDE interface with the following components:

- Database Explorer:** Shows the database structure for 'lab_db'. The 'Airline' table is selected, showing columns: airline_id (int, auto-increment), airline_code (varchar(30)), airline_name (varchar(50)), airline_country (varchar(50)), created_at (timestamp), and updated_at (timestamp).
- Console:** Contains a SQL query that filters for airline codes containing digits using a REGEXP pattern. The query is as follows:

```
236 WHERE weight_in_kg > 25
237 ORDER BY weight_in_kg DESC
238 LIMIT 3;
239
240
241 SELECT first_name, last_name
242 FROM Passengers
243 ORDER BY date_of_birth DESC
244 LIMIT 1;
245
246 SELECT booking_platform, MIN(ticket_price) AS cheapest_price
247 FROM Booking
248 GROUP BY booking_platform;
249
250 ✓ SELECT *
251 FROM Airline
252 WHERE airline_code REGEXP '[0-9]';
253
```
- Output:** Displays the results of the SQL query in a table with 6 columns: airline_id, airline_code, airline_name, airline_country, created_at, and updated_at. The results are as follows:

airline_id	airline_code	airline_name	airline_country	created_at	updated_at
1	KZ001	KazAir	Turkey	2025-09-28 18:45:40	2025-09-28 18:45:40
2	FR001	AirEasy	France	2025-09-28 18:45:50	2025-09-28 18:45:50
3	BR001	FlyHigh	Brazil	2025-09-28 18:45:50	2025-09-28 18:45:50
4	PL001	FlyFly	Poland	2025-09-28 18:45:50	2025-09-28 18:45:50
- Services:** A list of database services and their execution times, including console (411 ms), Booking_flight (358 ms), Booking (363 ms), Baggage_check (366 ms), Airline (352 ms), Boarding_pass (348 ms), Baggage (340 ms), Flights (338 ms), and Airport (394 ms).

Task 12

Baggage checks where update_at is in the same month as created_at but occurs earlier than created_at.

The screenshot shows a database management tool interface with the following components:

- Database Explorer:** Shows a tree view of the database structure. The 'Airline' table is selected, showing columns: airline_id (int, auto-increment), airline_code (varchar(30)), airline_name (varchar(50)), airline_country (varchar(50)), created_at (timestamp), and updated_at (timestamp).
- Console:** Contains SQL queries. The first query finds the cheapest price for each booking platform. The second query filters the 'Airline' table for codes matching a regex '[0-9]'. The third query, which is highlighted, selects all data from the 'Airline' table, ordered by 'created_at' in descending order, with a limit of 5 rows.
- Output:** Displays the results of the third query in a table with 6 columns: airline_id, airline_code, airline_name, airline_country, created_at, and updated_at. It shows 5 rows of data.
- Services:** A list of database services and their execution times, including 'console' (1 s 258 ms), 'Booking_flight' (358 ms), 'Booking' (363 ms), 'Baggage_check' (366 ms), 'Airline' (352 ms), 'Boarding_pass' (348 ms), 'Baggage' (340 ms), 'Flights' (338 ms), and 'Airport' (394 ms).

airline_id	airline_code	airline_name	airline_country	created_at	updated_at
1	TK	TurkishAir	Turkey	2025-09-29 13:47:35	2025-09-29 13:47:35
2	FR001	AirEasy	France	2025-09-28 18:45:50	2025-09-28 18:45:50
3	BR001	FlyHigh	Brazil	2025-09-28 18:45:50	2025-09-28 18:45:50
4	PL001	FlyFly	Poland	2025-09-28 18:45:50	2025-09-28 18:45:50
5	KZ001	KazAir	Turkey	2025-09-28 18:45:40	2025-09-28 18:45:40

Conclusion

In the course of the work, the Airport database was created and filled in, and the relationships between the tables were configured. Data sampling requests were performed, including filtering by criteria, sorting, grouping, and aggregation functions.