

Kazakh-British Technical University
DATABASES
Laboratory work №10

Alexandr Kislitsin
23B031861

2.Create a stored procedure to update the status of a flight.

The screenshot shows the MySQL Workbench interface with the following details:

- Database Explorer:** Shows the schema of the `airport_db` database, including tables like `Airline`, `Flight`, and `Passenger`.
- Code Editor:** The code for the stored procedure `update_flight_status` is displayed. It takes parameters `p_flight_id` and `p_new_status`. It updates the `Flights` table, sets the new status and update time, and returns a result message or an error message if the flight is not found.
- Output Window:** Shows the execution of the stored procedure with the command `CALL update_flight_status(p_flight_id 1, p_new_status 'boarding');`. The output shows the flight record being updated.
- Notifications:** A timeline on the right shows several notifications related to the current session.

3.Create a stored procedure that returns a list of flights departing from a specific airport.

The screenshot shows the MySQL Workbench interface with the following details:

- Database Explorer:** Shows the schema of the `airport_db` database, including tables like `Airline`, `Flight`, and `Passenger`.
- Code Editor:** The code for the stored procedure `get_flights_from_airport` is displayed. It joins multiple tables to find flights departing from a specific airport, counts the total number of flights, and returns the count.
- Output Window:** Shows the execution of the stored procedure with the command `CALL get_flights_from_airport(p_airport_id 1);`. The output shows the total number of flights.
- Notifications:** A timeline on the right shows several notifications related to the current session.

4.Create a function to calculate the average delay time of flights arriving at a specific airport.

```

CREATE FUNCTION calculate_avg_delay(
    p_airport_id INT
)
RETURNS DECIMAL(10,2)
BEGIN
    RETURN IFNULL((SELECT AVG(actual_arrival - scheduled_arrival)
    FROM Flights
    WHERE actual_arrival > scheduled_arrival
    AND p_airport_id = airport_id), 0);
END;

```

```

SELECT calculate_avg_delay(1) AS avg_delay_hours;

```

airport_id	airport_name	avg_delay_hours
1	Akunaq Heliport	3087.72
2	Alert Bay Airport	3689.00
3	Armidale Airport	3162.46
11	Bermuda Dunes Airport	3161.00
18	Darchula Airport	2845.85

5.Create a stored procedure that lists all passengers for a given flight number.

```

CREATE PROCEDURE get_passengers_for_flight(
    p_flight_id INT
)
BEGIN
    SELECT COUNT(DISTINCT p.passenger_id) AS total_passengers
    FROM Passengers p
    JOIN Booking b ON p.passenger_id = b.passenger_id
    LEFT JOIN Boarding_pass bp ON b.booking_id = bp.booking_id
    WHERE b.flight_id = p_flight_id
    ORDER BY p.last_name, p.first_name;
END;

```

```

CALL get_passengers_for_flight(1);

```

total_passengers
8

6. Create a stored procedure to find the passenger who has taken the greatest number of flights.

The screenshot shows the DataGrip IDE interface. The Database Explorer panel on the left lists tables like `Boarding_pass`, `Booking`, `Booking_flight`, and `Flights`. The main code editor window contains the following SQL code:

```
CREATE PROCEDURE find_most_frequent_passenger()
BEGIN
    SELECT
        p.passenger_id,
        CONCAT(p.first_name, ' ', p.last_name) AS full_name,
        p.passport_number,
        p.country_of_citizenship,
        COUNT(b.booking_id) AS total_flights
    FROM Passengers p
        JOIN Booking b ON p.passenger_id = b.passenger_id
    GROUP BY p.passenger_id, p.first_name, p.last_name, p.passport_number, p.country_of_citizenship
    ORDER BY total_flights DESC
    LIMIT 1;
END;
CALL find_most_frequent_passenger();
```

The Services panel shows a transaction with a single query: `CALL find_most_frequent_passenger();`. The Output panel displays the result of the query:

	passenger_id	full_name	passport_number	country_of_citizenship	total_flights
1	68	Sheela Roux	014456243-X	Thailand	1

Notifications on the right show several notifications from the user "C on n ec te d".

7. Create a stored procedure to find all flights that are delayed by more than 24 hours.

The screenshot shows the DataGrip IDE interface. The Database Explorer panel on the left lists tables like `Boarding_pass`, `Booking`, `Booking_flight`, and `Flights`. The main code editor window contains the following SQL code:

```
CREATE PROCEDURE find_delayed_flights_24h()
-- Дополнительно выводим статистику
SELECT
    COUNT(*) AS total_delayed_flights,
    AVG(TIMESTAMPDIFF(HOUR, scheduled_arrival, actual_arrival)) AS avg_delay_hours,
    MAX(TIMESTAMPDIFF(HOUR, scheduled_arrival, actual_arrival)) AS max_delay_hours
FROM Flights
WHERE TIMESTAMPDIFF(HOUR, scheduled_arrival, actual_arrival) > 24;
END;
CALL find_delayed_flights_24h();
```

The Services panel shows a transaction with a single query: `CALL find_delayed_flights_24h();`. The Output panel displays the result of the query:

	total_delayed_flights	avg_delay_hours	max_delay_hours
1	587	2890.9349	8280

Notifications on the right show several notifications from the user "C on n ec te d".

8.Create a function that counts the number of flights for each airline.

The screenshot shows the MySQL Workbench interface with the following details:

- Database Explorer:** Shows the schema structure of the `@localhost` database, including tables like `Boarding_pass`, `Booking`, and `Flights`.
- Editor:** Displays the SQL code for creating a function:

```
1670
1671
1672
1673     SELECT
1674         a.airline_id,
1675         a.airline_name,
1676         a.airline_country,
1677         count_flights_by_airline( p_airline_id a.airline_id ) AS total_flights
1678     FROM Airline a
1679     ORDER BY total_flights DESC;
1680
1681     SELECT count_flights_by_airline( p_airline_id 1 ) AS flights_count;
```
- Output:** Shows the result of the function call `flights_count` with a value of 32.
- Notifications:** A sidebar showing a timeline of notifications related to the current session.

9.Create a stored procedure to calculate the average ticket price for a specific flight.

The screenshot shows the MySQL Workbench interface with the following details:

- Database Explorer:** Shows the schema structure of the `@localhost` database.
- Editor:** Displays the SQL code for creating a stored procedure:

```
1689
1690     CREATE PROCEDURE calculate_avg_ticket_price(
1691         v_min_price AS min_price,
1692         v_max_price AS max_price,
1693         v_booking_count AS total_bookings
1694     )
1695     ELSE
1696         SELECT
1697             p_flight_id AS flight_id,
1698             'Нет бронирований для этого рейса' AS message;
1699     END IF;
1700
1701     END;
```



```
1726     CALL calculate_avg_ticket_price( p_flight_id 1 );
```
- Output:** Shows the result of the stored procedure call with a message: "Нет бронирований для этого рейса".
- Notifications:** A sidebar showing a timeline of notifications related to the current session.

10.Create a stored procedure to find the flight with the highest ticket price. The procedure should return the flight number, the departure and arrival airports, and the ticket price for the most expensive flight.

The screenshot shows a SQL development environment with the following interface elements:

- Database Explorer:** Shows the database structure at `@localhost`, including tables like `Boarding_pass`, `Booking`, `Booking_flight`, `Flights`, `Passengers`, and `Security_check`.
- Query Editor:** Displays the SQL code for creating a stored procedure named `find_most_expensive_flight`. The code performs a multi-table join across `Booking`, `Flights`, `Airport`, and `Airline` tables to find the top 5 flights by ticket price.
- Notifications:** A sidebar showing activity logs with entries like "Created" and "Completed".
- Services:** A sidebar showing active database connections and their execution times.
- Output:** A results grid showing the output of the stored procedure call, which is empty ("0 rows").
- Bottom Status:** Shows the file path (`Desktop > DATABASES > last_labs.sql`), the line number (1783), and encoding information (LF, UTF-8).

```
CREATE PROCEDURE find_most_expensive_flight()
    FROM Booking b
        JOIN Flights f ON b.flight_id = f.flight_id
        JOIN Airport dep ON f.departure_airport_id = dep.airport_id
        JOIN Airport arr ON f.arrival_airport_id = arr.airport_id
        JOIN Airline a ON f.airline_id = a.airline_id
    ORDER BY b.ticket_price DESC
    LIMIT 5;
END;

CALL find_most_expensive_flight();
```