# CS306 Database Systems – Project Phase II Report

**Group Members:**

- Mert Güngör (34159)
- Kaan Berk Karabıyık (34424)

## 1. Introduction

In this second phase of our project, we improved our Airline Operations System. We added **database triggers** and **stored procedures** to make the system more automatic and secure.

We used **MariaDB** with **HeidiSQL** for this project. We preferred this setup because we have experience with it from our other projects. The SQL code we wrote works perfectly with the project requirements.

## 2. Triggers

We created two triggers. One prevents mistakes (like overbooking), and the other keeps a record of deleted data.

### 2.1. Trigger 1: `check_capacity_before_insert`

- **What it does:** This trigger runs `BEFORE` a new passenger is added to the `Booking` table. It checks the aircraft's capacity. If the flight is full, the trigger stops the action and shows an error message. This makes sure we do not sell more tickets than the seat count.
- Evidence:
  The screenshot below shows that when we try to add a 3rd passenger to a flight with a capacity of 2, the system gives an error: "ERROR: Flight is full! Cannot accept new booking."

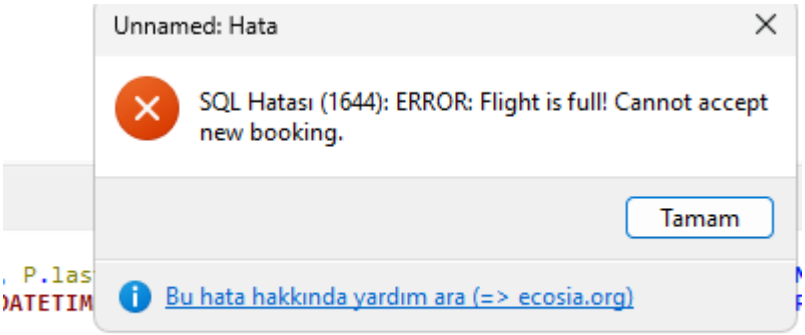



Figure 1: The trigger prevents adding a new passenger because the flight is full.

### 2.2. Trigger 2: `log_booking_delete`

- **What it does:** This trigger runs `AFTER` a booking is deleted. It is used for security and auditing. When a reservation is cancelled, this trigger automatically saves the flight number, seat number, and the deletion time into a separate table called `Booking_Log`.

- Evidence:
  Below, you can see the log table is empty at first. After we delete a booking, the log table automatically gets the new record.

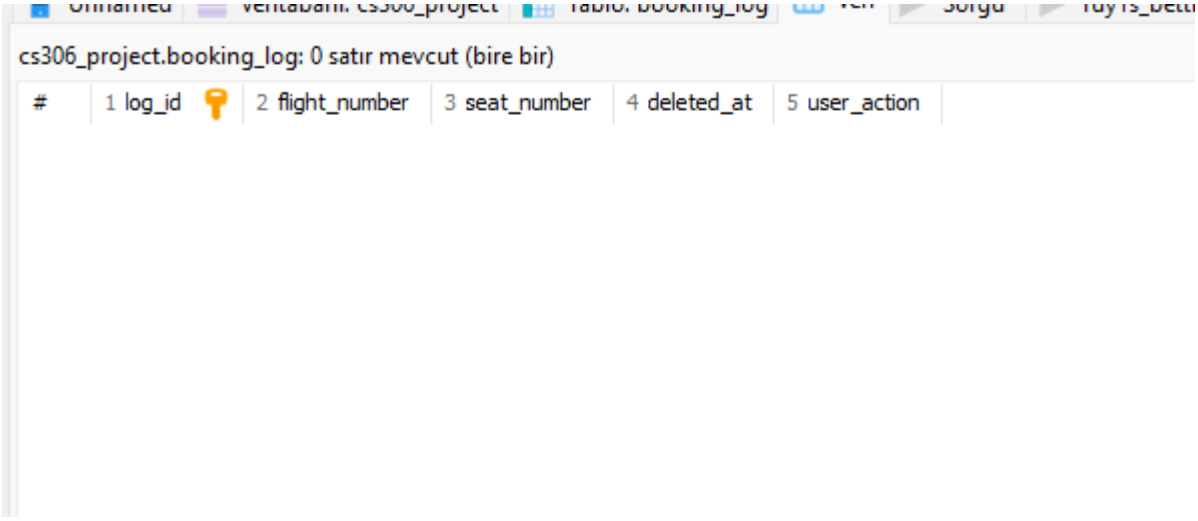
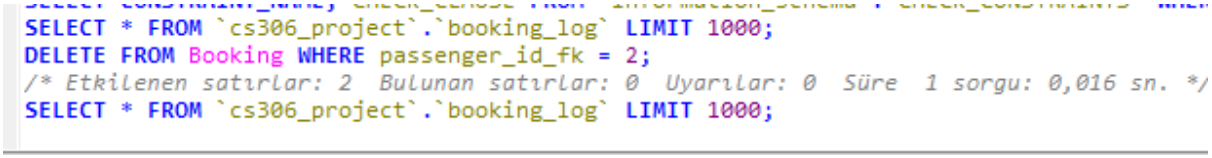
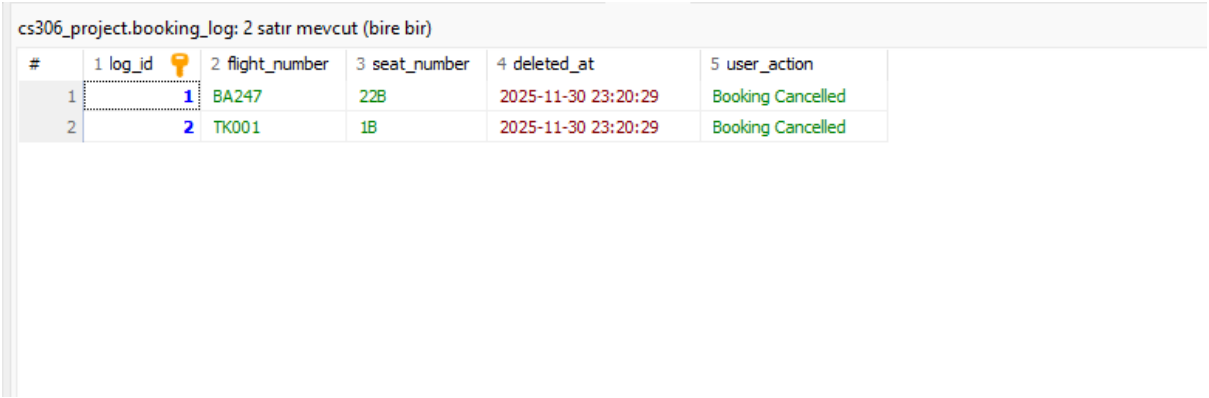



Figure 2: The log table is empty before deletion.

Figure




3: We execute the delete command.




- Figure 4: The deleted booking information appears in the log table.

### 3. Stored Procedures

We wrote two stored procedures to make complex tasks easier and faster.

#### 3.1. Procedure 1: `GetPassengerManifest`

- **What it does:** This procedure takes a `flight_number` as input. It joins the `Booking` and `Passenger` tables. It returns a simple list of all passengers (Names and Seat Numbers) for that specific flight. This is useful for the flight crew.

- Evidence:
  The screenshot below shows the passenger list for flight TK001.

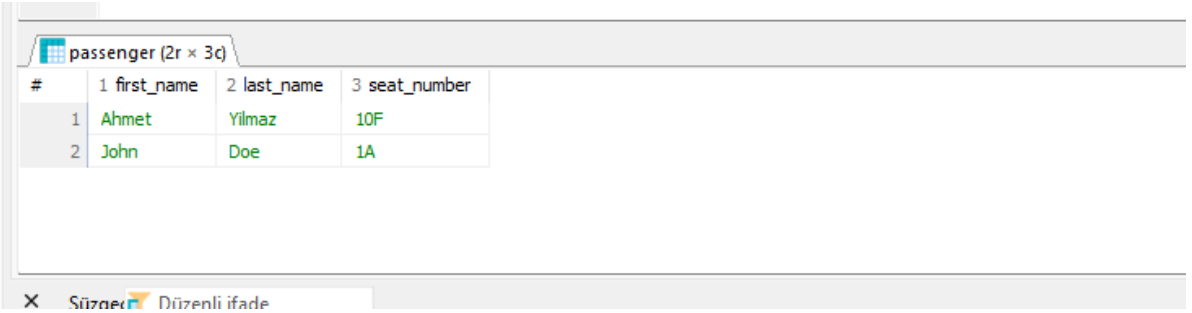



Figure 5: The procedure shows the passenger list.

### 3.2. Procedure 2: ScheduleFlight

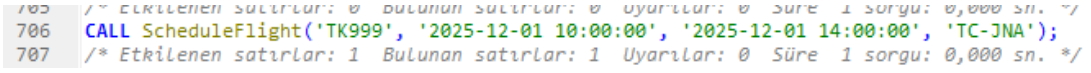
- **What it does:** This procedure makes it easy to add a new flight. Instead of writing a long INSERT command, we just call this procedure with the flight details (Flight Number, Time, Plane). It automatically fills in the route (IST to LHR) and airline information.
- Evidence:
  The screenshot below shows that a new flight is successfully created with a simple command.
- 


Figure 6: A new flight is scheduled successfully.