Assignment: Database Operations for Railway Management System

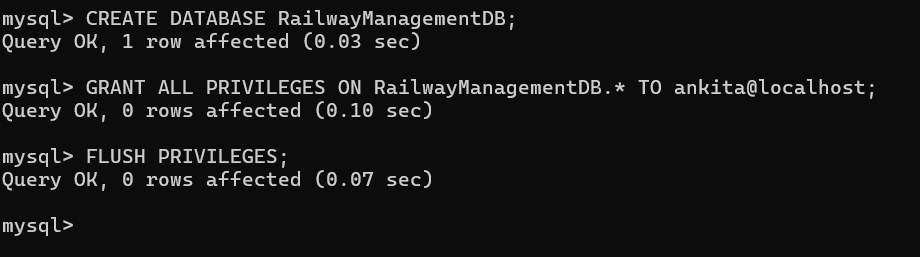
## Duration: 2 Hours

**Scenario**:

The Railway Management System is a critical part of the transportation sector that manages trains, schedules, routes, passengers, and ticket bookings. You have been appointed as the database consultant to design and implement a database system that efficiently manages the day-to-day operations of the railway system.

# Task 1: Database Creation and Table Setup

1. **Create a database** named RailwayManagementDB.



1. **Create the following tables** to track train information, schedules, routes, passengers, and bookings:
   * Trains: To store information about trains (TrainID, TrainName, TrainType, TotalSeats).

Ans:

mysql> CREATE TABLE TRAINS (

-> TRAINID INT,

-> TRAINNAME VARCHAR (50),

-> TRAINTYPE VARCHAR (20),

-> TOTALSEATS INT

-> );

Query OK, 0 rows affected (0.08 sec)

* + Routes: To store information about routes (RouteID, StartStation, EndStation, Distance).

Ans: mysql> CREATE TABLE ROUTES (

-> ROUTEID INT,

-> STARTSTATION VARCHAR (50),

-> ENDSTATION VARCHAR (50),

-> DISTANCE\_KM INT

-> );

Query OK, 0 rows affected (0.03 sec)

* + Schedules: To store train schedules (ScheduleID, TrainID, RouteID, DepartureTime, ArrivalTime).

Ans: mysql> CREATE TABLE SCHEDULES (

-> SCHEDULEID INT,

-> TRAINID INT,

-> ROUTEID INT,

-> DEPARTURETIME DATETIME,

-> ARRIVALTIME DATETIME

-> );

Query OK, 0 rows affected (0.02 sec)

* + Passengers: To store passenger information (PassengerID, FirstName, LastName, Age, Email).

Ans:

mysql> CREATE TABLE PASSENGERS (

-> PASSENGERID INT,

-> FIRSTNAME VARCHAR (30),

-> LASTNAME VARCHAR (30),

-> AGE INT,

-> EMAIL VARCHAR (100)

-> );

Query OK, 0 rows affected (0.03 sec)

* + Bookings: To store booking details (BookingID, PassengerID, ScheduleID, BookingDate, SeatNumber).

Ans: mysql> CREATE TABLE BOOKINGS (

-> BOOKINGID INT,

-> PASSENGERID INT,

-> SCHEDULEID INT,

-> BOOKINGDATE DATE,

-> SEATNUMBER INT

-> );

Query OK, 0 rows affected (0.02 sec)

1. **Insert sample data** into the tables for at least:
   * 5 trains

mysql> INSERT INTO TRAINS (TRAINID, TRAINNAME, TRAINTYPE, TOTALSEATS)

-> VALUES

-> (1, 'Rajdhani Express', 'Express', 300),

-> (2, 'Tejas Express', 'Superfast', 200),

-> (3, 'Shatabdi Express', 'Passenger', 250),

-> (4, 'Duronto Express', 'Superfast', 150),

-> (5, 'Garib Rath', 'Express', 350);

Query OK, 5 rows affected (0.06 sec)

Records: 5 Duplicates: 0 Warnings: 0

* + 3 routes

mysql> INSERT INTO ROUTES (ROUTEID,

-> STARTSTATION, ENDSTATION, DISTANCE\_KM)

-> VALUES

-> (1, 'DELHI', 'MUMBAI', 1400),

-> (2, 'KOLKATA', 'CHENNAI', 1650),

-> (3, 'JAIPUR', 'AHMEDABAD', 650);

Query OK, 3 rows affected (0.01 sec)

Records: 3 Duplicates: 0 Warnings: 0

* + 5 schedules

mysql> INSERT INTO SCHEDULES (SCHEDULEID,

-> TRAINID, ROUTEID, DEPARTURETIME,

-> ARRIVALTIME)

-> VALUES

-> (1, 1, 1, '2024-10-20 09.00.00', '2024-10-20 21.00.00'),

-> (2, 2, 2, '2024-10-21 08.30.00', '2024-10-21 22.00.00'),

-> (3, 3, 3, '2024-10-22 06.00.00', '2024-10-22 14.00.00'),

-> (4, 4, 1, '2024-10-23 10.00.00', '2024-10-23 22.00.00'),

-> (5, 5, 2, '2024-10-24 07.00.00', '2024-10-24 21.00.00');

Query OK, 5 rows affected, 10 warnings (0.01 sec)

Records: 5 Duplicates: 0 Warnings: 10

* + 10 passengers

mysql> INSERT INTO PASSENGERS (PASSENGERID, FIRSTNAME, LASTNAME, AGE, EMAIL)

-> VALUES

-> (1, 'Rajesh', 'Sharma', 45, 'rajesh.sharma@specialforce.com'),

-> (2, 'Priya', 'Mehra', 32, 'priya.mehra@specialforce.com'),

-> (3, 'Ankit', 'Verma', 29, 'ankit.verma@specialforce.com'),

-> (4, 'Kavita', 'Gupta', 40, 'kavita.gupta@specialforce.com'),

-> (5, 'Arun', 'Patel', 50, 'arun.patel@specialforce.com'),

-> (6, 'Neha', 'Joshi', 27, 'neha.joshi@specialforce.com'),

-> (7, 'Suresh', 'Nair', 33, 'suresh.nair@specialforce.com'),

-> (8, 'Pooja', 'Reddy', 36, 'pooja.reddy@specialforce.com'),

-> (9, 'Vikram', 'Singh', 42, 'vikram.singh@specialforce.com'),

-> (10, 'Aarti', 'Desai', 25, 'aarti.desai@specialforce.com');

Query OK, 10 rows affected (0.04 sec)

Records: 10 Duplicates: 0 Warnings: 0

* + 5 bookings

mysql> INSERT INTO BOOKINGS (BOOKINGID, PASSENGERID, SCHEDULEID, BOOKINGDATE, SEATNUMBER)

-> VALUES

-> (1, 1, 1, '2024-10-10', 12),

-> (2, 2, 1, '2024-10-11', 34),

-> (3, 3, 2, '2024-10-12', 56),

-> (4, 4, 3, '2024-10-13', 18),

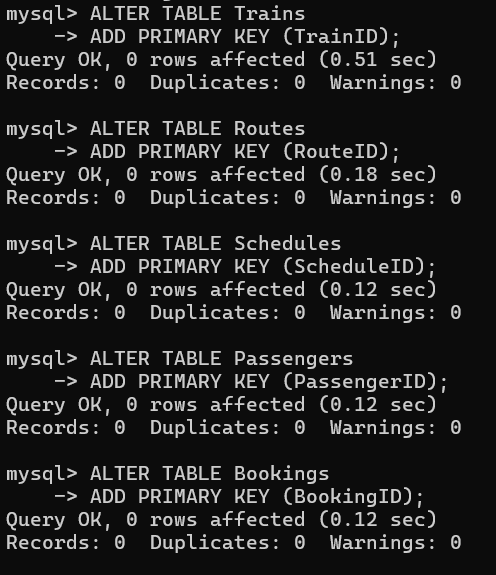
-> (5, 5, 4, '2024-10-14', 22);

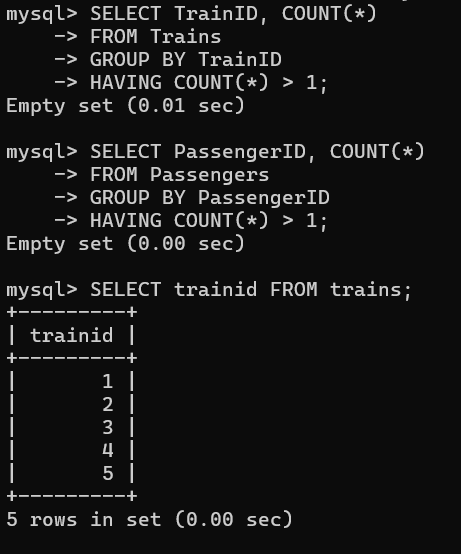
Query OK, 5 rows affected (0.01 sec)

Records: 5 Duplicates: 0 Warnings: 0

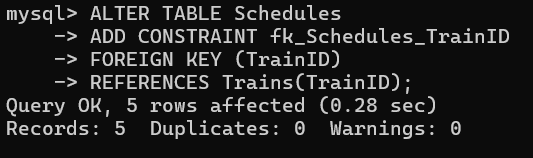
# Task 2: Add Constraints After Data Insertion (Strictly write after data insertion)

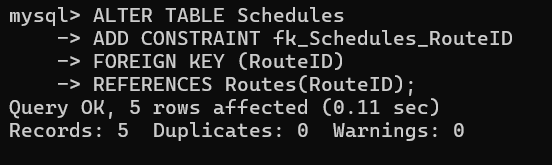
1. Add a **Primary Key** to each table.

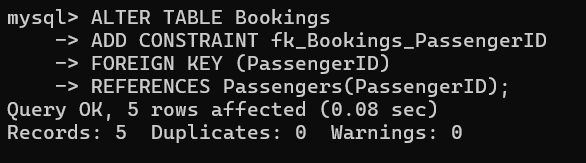


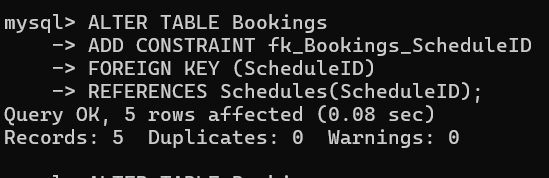


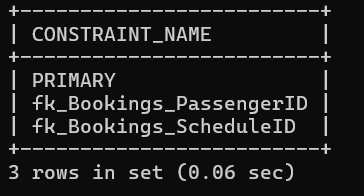
1. Add **Foreign Keys** to establish relationships between:
   * Schedules and Trains (on TrainID).
   * Schedules and Routes (on RouteID).
   * Bookings and Passengers (on PassengerID).
   * Bookings and Schedules (on ScheduleID).







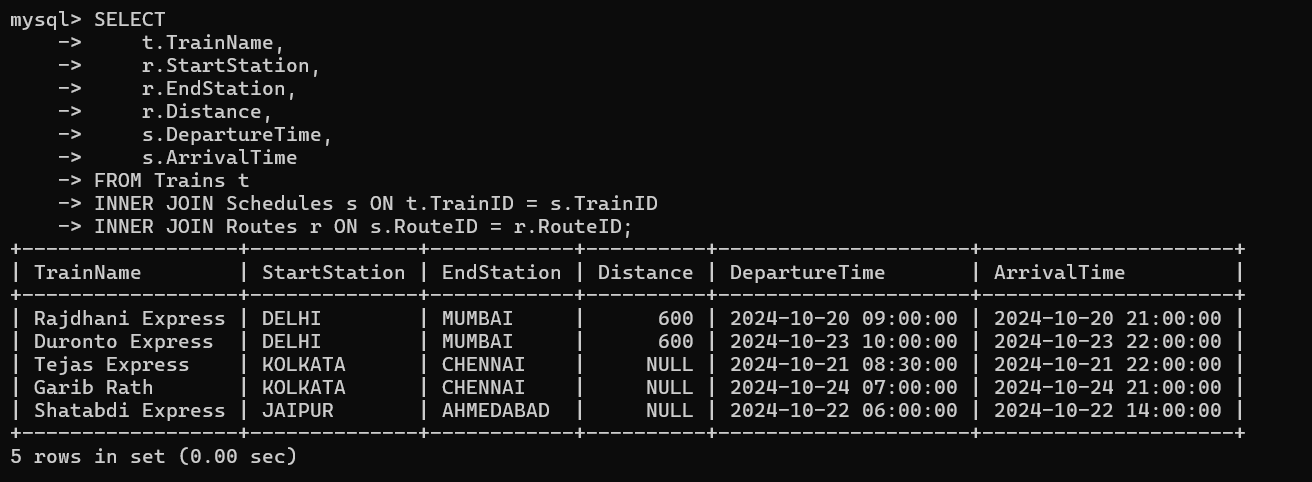




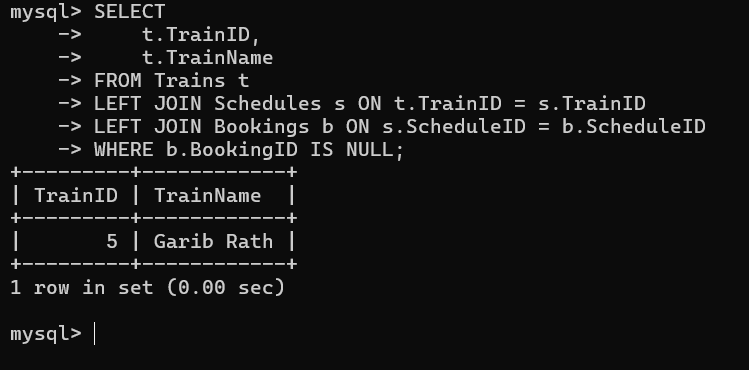
# Task 3: Joins and Queries

1. **Query 1**: Write a query to retrieve the train name, route details, and schedule for all trains using an

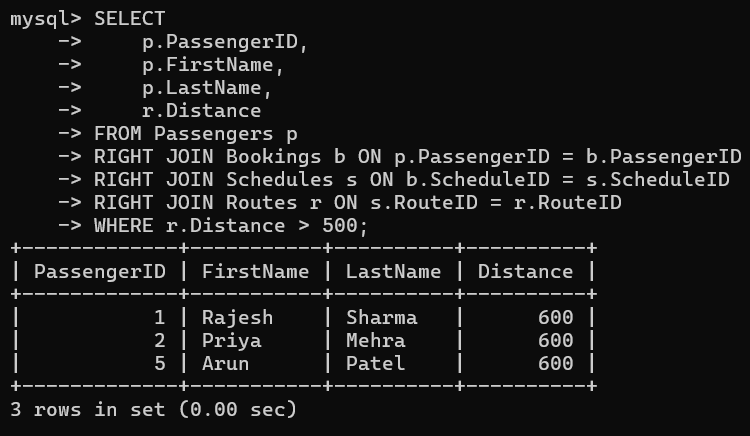
**INNER JOIN** between the Trains, Routes, and Schedules tables.



1. **Query 2**: Write a query to retrieve all trains that don't have any bookings using a **LEFT JOIN** between the Trains and Bookings tables.

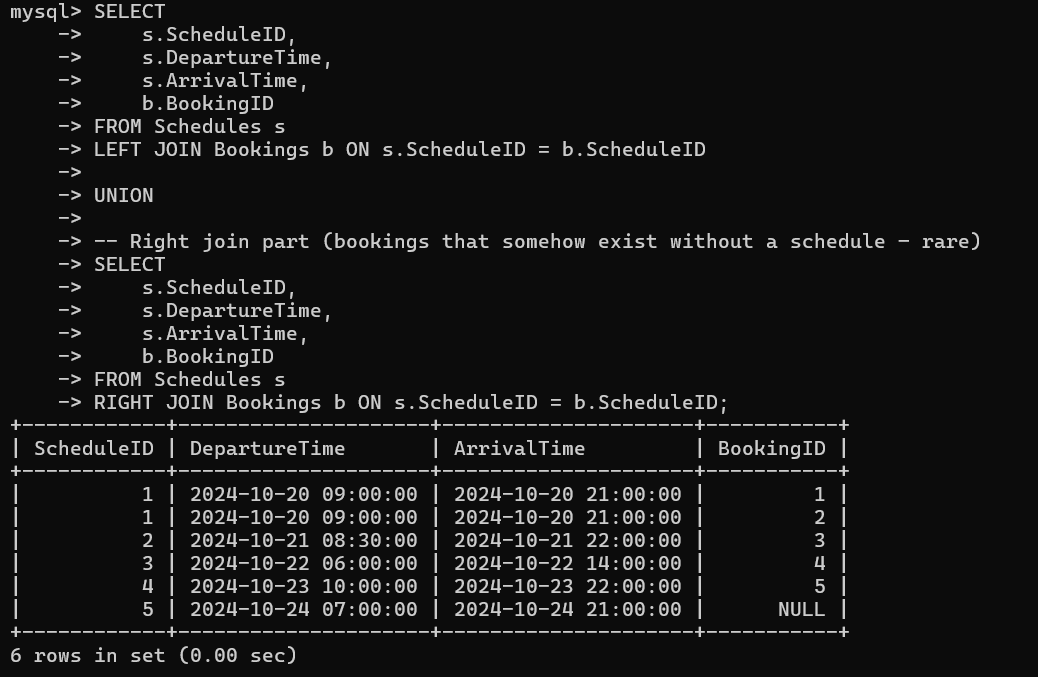


**Query 3**: Write a query to find all passengers who have booked seats for trains traveling a distance of more than 500 km using a **RIGHT JOIN** and subquery.



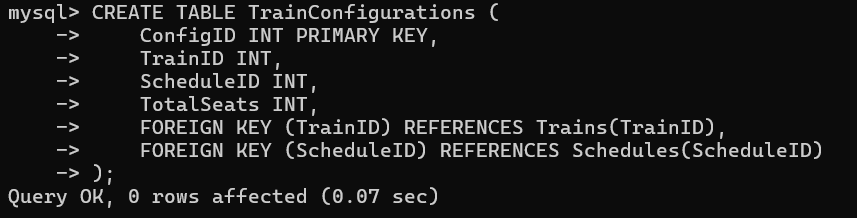
**Query 4**: Write a query to list all train schedules, even if there are no passengers booked, using an

**OUTER JOIN**.



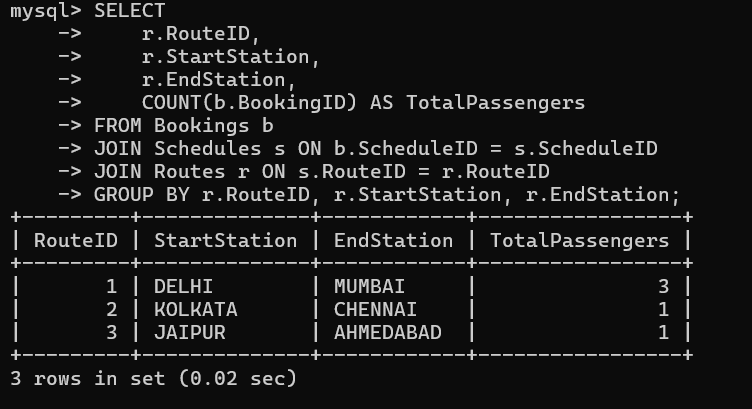
# Task 4: Normalization

1. **Normalize** the tables to the **3rd Normal Form (3NF)** to eliminate redundancy and ensure data integrity.

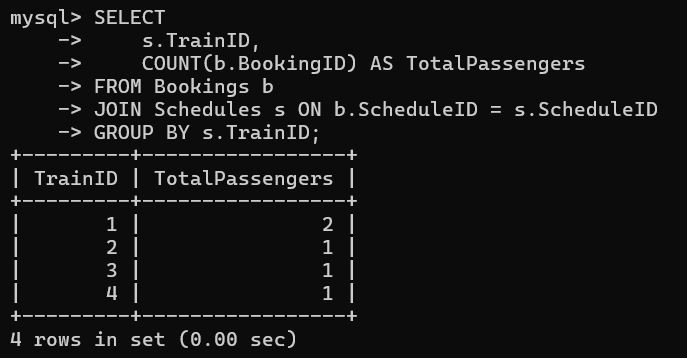


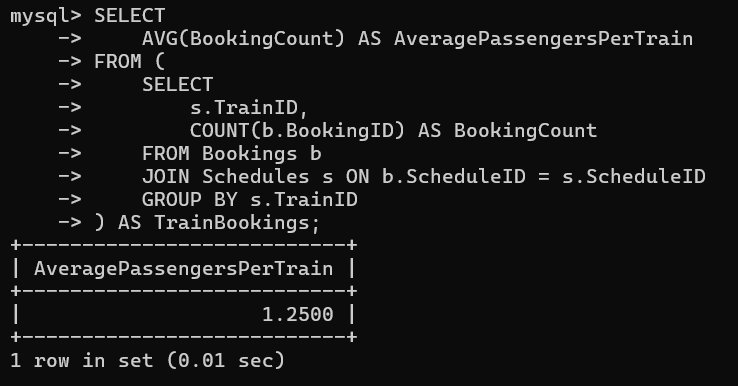
# Task 5: Sub Queries

1. **Query 5**: Write a query to calculate the **total number of passengers** for each train route.

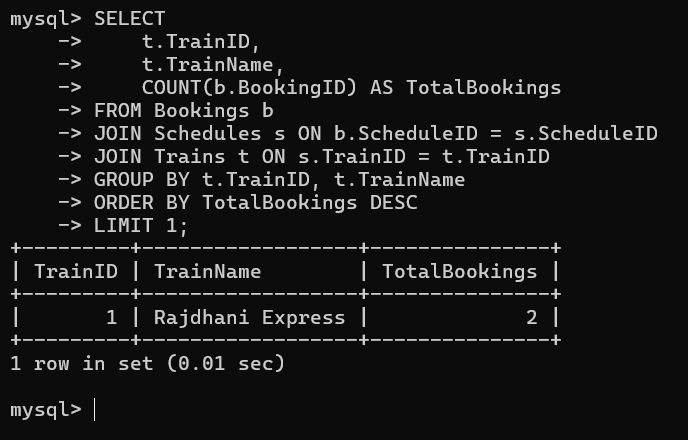


1. **Query 6**: Write a query to find the **average number of passengers** booked per train.

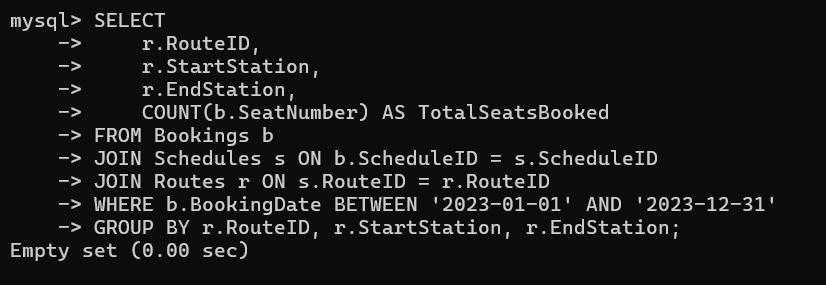




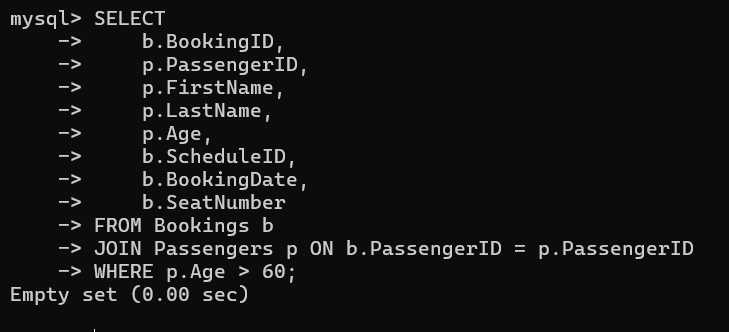
1. **Query 7**: Write a query to find the **train with the highest number of bookings**.



1. **Query 8**: Write a query to find the **total seats booked per train route** where the booking date is between 01-Jan-2023 and 31-Dec-2023.

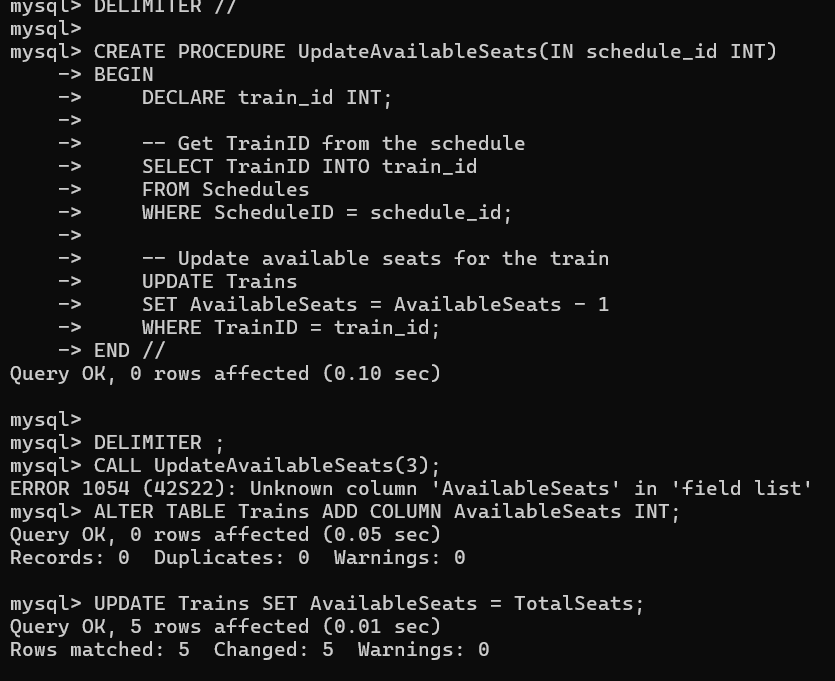


1. **Query 9**: Write a query to list all bookings where the passenger's age is greater than 60.



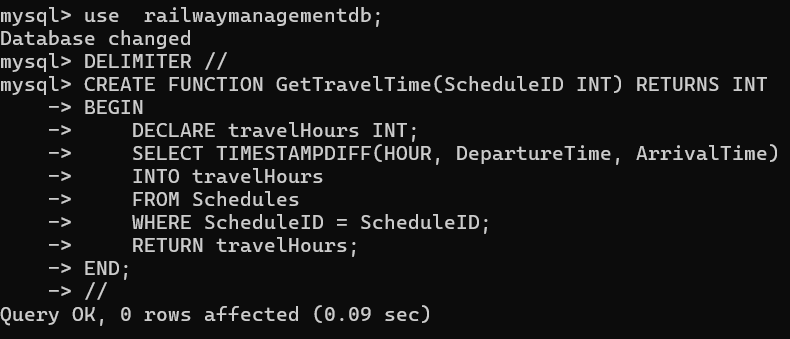
# Task 6: Stored Procedures and Functions

1. **Write a stored procedure** to update the number of available seats in a train after a booking has been made.



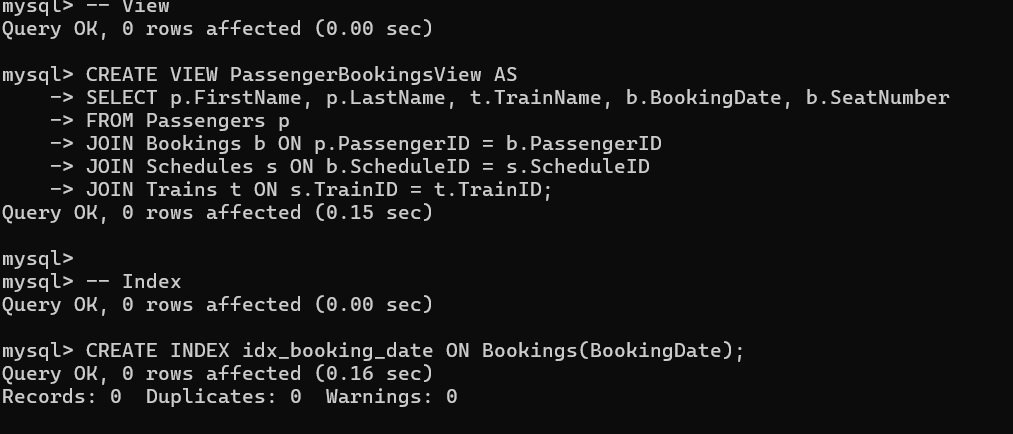
1. **Write a function** to calculate the total travel time (in hours) between two stations based on departure and arrival times from the Schedules table.

**Ans:**

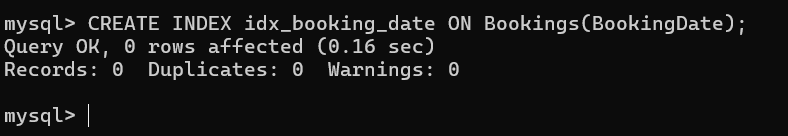


# Task 7: Views and Indexes

1. **Create a view** named PassengerBookingsView that combines passenger details, train information, and booking details in one query.



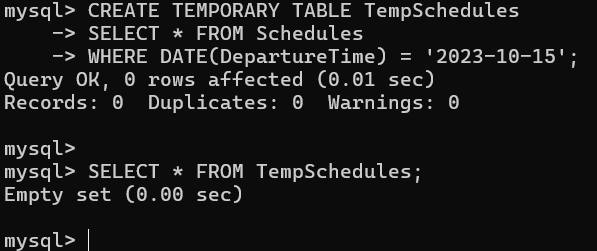
1. **Create an index** on the Bookings table to improve the performance of queries filtering by BookingDate.



# Task 8: Temporary Tables

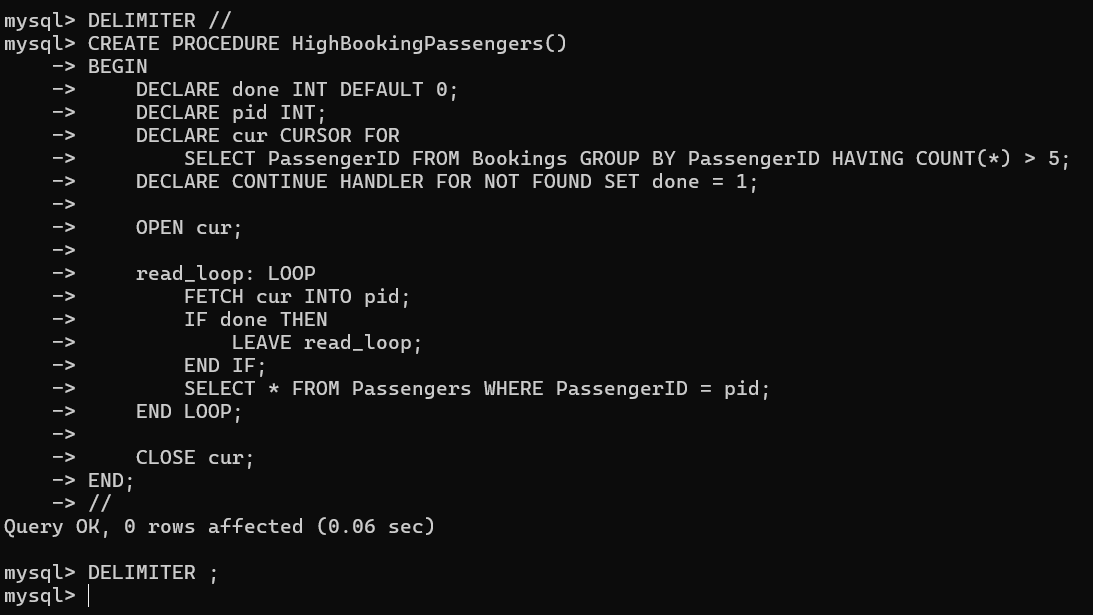
1. **Create a temporary table** to store the schedule of all trains departing on a specific day (for example,

15-Oct-2023), and then query it.



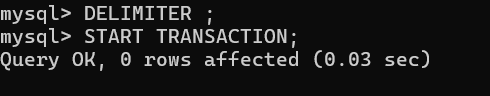
# Task 9: Cursors

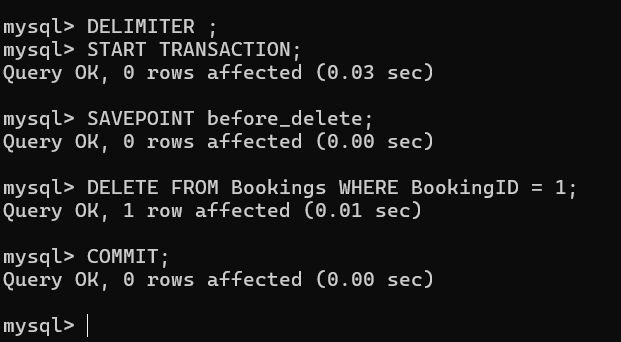
1. Write a **basic cursor** to iterate over the passengers who have booked more than 5 tickets.



# Task 10: ACID Properties and Transactions

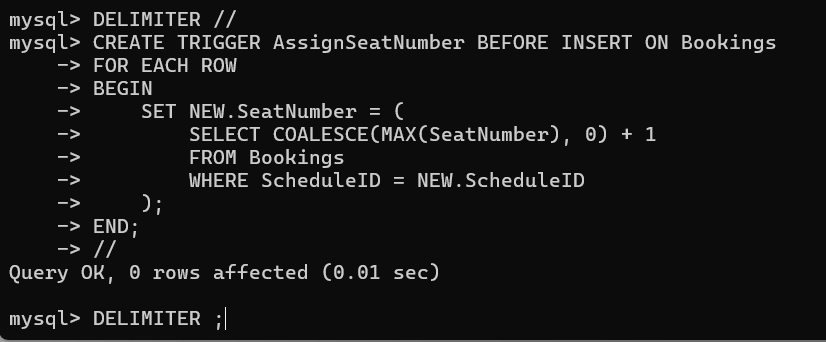
1. Ensure the database follows **ACID properties** by using **transactions**:
   * **Begin a transaction** before updating seat availabilit
   * Use a **savepoint** before deleting any booking records.
   * **Rollback** if an error occurs, ensuring no changes are made to the database.
   * **Commit** the transaction only after all changes have been successfully applied.



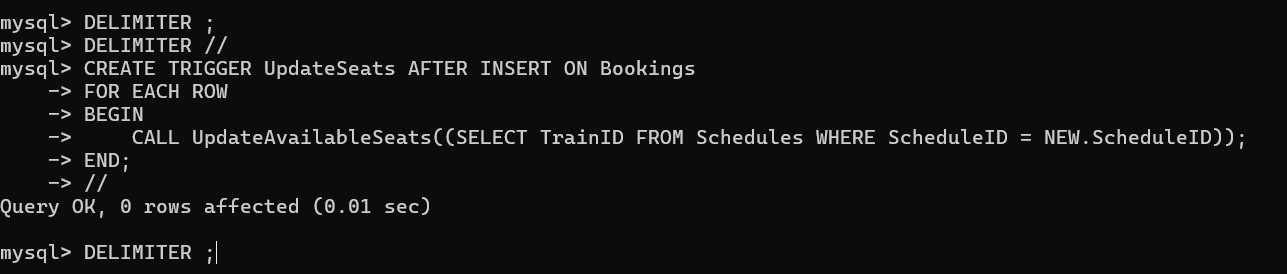


# Task 11: Triggers (Understand which trigger to use when and for what)

1. Create a **Trigger** that automatically assigns a seat number to passengers when a booking is created.

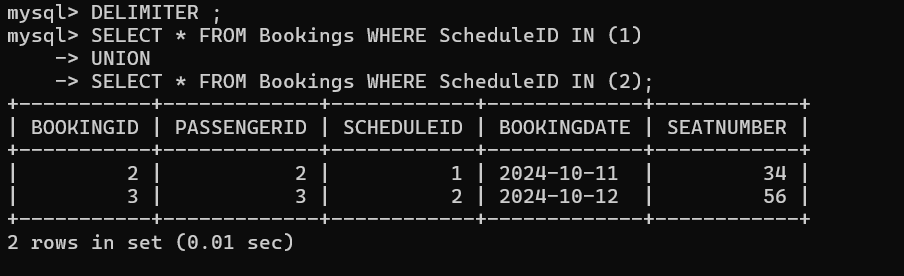


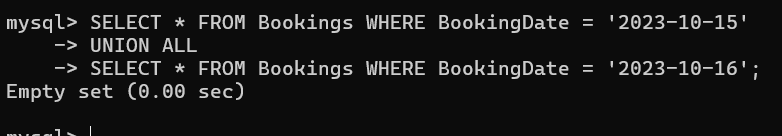
1. Create a **Trigger** that updates the total available seats in the train after a booking is confirmed.



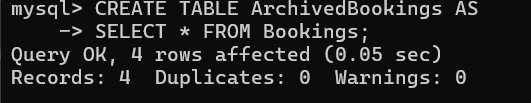
# Task 12: UNION and UNION ALL

1. Write a query to combine the results of two queries that return passengers booked on trains for two different routes.



1. Write a query to combine the results of all bookings made on different dates.
2. 

# Task 13: Copying Tables

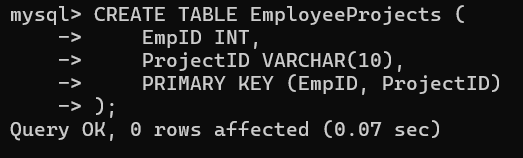
1. **Copy the structure** of the Passengers table into a new table named OldPassengers.
2. 
3. **Copy all the data** from the Bookings table into another table named ArchivedBookings.
4. 

# Additional Assignment: Keys Practice (You should know exact difference and how to use them)

## Duration: 15-20 Minutes Scenario:

SpecialForce Private Limited has an Employee Management System that stores employee data. The management system uses various keys to maintain data integrity and ensure fast retrieval of information. You are tasked with identifying and implementing different types of keys in the database schema provided below.

2222



# Employee Table:

**EmpID (PK) FirstName LastName Email PhoneNumber Department**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 101 | Rajesh | Sharma | [rajesh@specialforce.com](mailto:rajesh@specialforce.com) | 9876543210 | HR |
| 102 | Priya | Mehra | [priya@specialforce.com](mailto:priya@specialforce.com) | 8765432109 | Finance |
| 103 | Ankit | Verma | [ankit@specialforce.com](mailto:ankit@specialforce.com) | 7654321098 | IT |
| 104 | Kavita | Gupta | [kavita@specialforce.com](mailto:kavita@specialforce.com) | 6543210987 | IT |
| 105 | Suresh | Nair | [suresh@specialforce.com](mailto:suresh@specialforce.com) | 5432109876 | Sales |
|  |  |  |  |  |  |

# Assignment Tasks:

## Super Key:

Identify all possible Super Keys from the Employee table. Remember, a Super Key is any combination of columns that uniquely identifies each record.

ANS: Super Keys: {EmpID}, {EmpID, Email}, {EmpID, FirstName}, {EmpID, PhoneNumber}, etc.

## Candidate Key:

Determine which of the Super Keys qualify as Candidate Keys. A Candidate Key is a minimal Super Key, meaning it contains no unnecessary columns.

ANS:Candidate Key: {EmpID}

## Primary Key:

What is the Primary Key for the Employee table? Explain why this key was chosen as the Primary Key.

ANS: Primary Key: EmpID (unique, not null, best for identification)

## Alternate Key:

Identify any Alternate Keys. Alternate Keys are Candidate Keys that were not chosen as the Primary Key.

ANS: Alternate Key: Email (can also be unique)

## Composite Key:

Suppose the company wants to track employees working on multiple projects, where both the EmpID and

ProjectID are needed to uniquely identify records in a new EmployeeProjects table. Define a Composite Key for this table using EmpID and ProjectID.

## EmployeeProjects Table: EmpID ProjectID

101 P01

102 P02

103 P03

104 P01

105 P03