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.
2. **Create the following tables** to track train information, schedules, routes, passengers, and bookings:
    - Trains: To store information about trains (TrainID, TrainName, TrainType, TotalSeats).
    - Routes: To store information about routes (RouteID, StartStation, EndStation, Distance).
    - Schedules: To store train schedules (ScheduleID, TrainID, RouteID, DepartureTime, ArrivalTime).
    - Passengers: To store passenger information (PassengerID, FirstName, LastName, Age, Email).
    - Bookings: To store booking details (BookingID, PassengerID, ScheduleID, BookingDate, SeatNumber).
3. **Insert sample data** into the tables for at least:
    - 5 trains
    - 3 routes
    - 5 schedules
    - 10 passengers
    - 5 bookings

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

1. Add a **Primary Key** to each table.
2. 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.
2. **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.
3. **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.
4. **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.
2. **Query 6**: Write a query to find the **average number of passengers** booked per train.
3. **Query 7**: Write a query to find the **train with the highest number of bookings**.
4. **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.
5. **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.
2. **Write a function** to calculate the total travel time (in hours) between two stations based on departure and arrival times from the Schedules table.

Task 7: Views and Indexes

1. **Create a view** named PassengerBookingsView that combines passenger details, train information, and booking details in one query.
2. **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 availability.
    - 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.
2. 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.
2. Write a query to combine the results of all bookings made on different dates.

Task 13: Copying Tables

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

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.

Employee Table:

EmpID (PK)	FirstName	LastName	Email	PhoneNumber	Department
101	Rajesh	Sharma	rajesh@specialforce.com	9876543210	HR
102	Priya	Mehra	priya@specialforce.com	8765432109	Finance
103	Ankit	Verma	ankit@specialforce.com	7654321098	IT
104	Kavita	Gupta	kavita@specialforce.com	6543210987	IT
105	Suresh	Nair	suresh@specialforce.com	5432109876	Sales

Assignment Tasks:

1. **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.
2. **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.
3. **Primary Key**:
   What is the Primary Key for the Employee table? Explain why this key was chosen as the Primary Key.
4. **Alternate Key**:
   Identify any Alternate Keys. Alternate Keys are Candidate Keys that were not chosen as the Primary Key.
5. **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