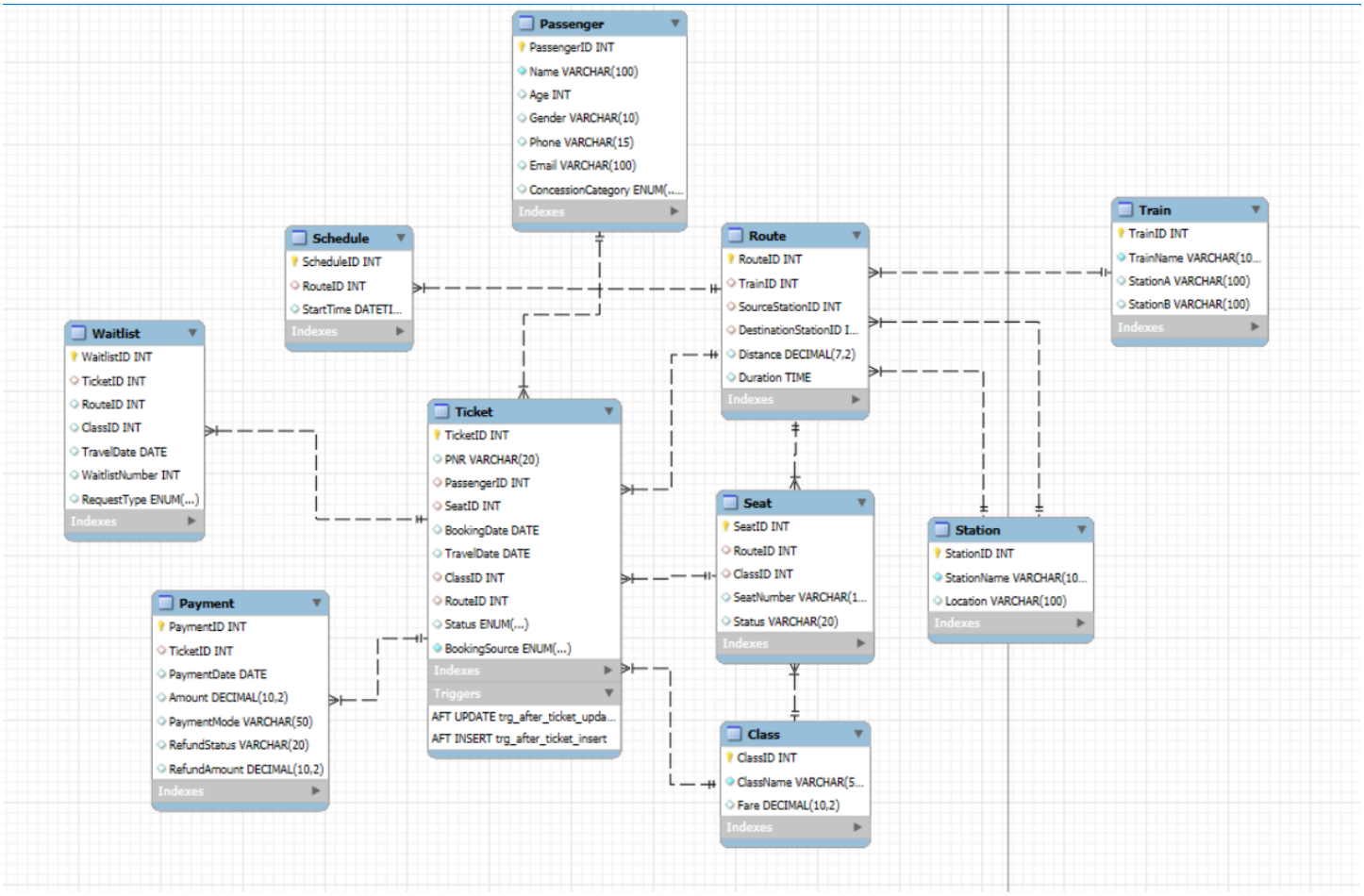
ER DIAGRAM



SCHEMA CREATION AND SAMPLE DATA INSERTION CODE

```
-- Create and use the RailwayReservation database

CREATE DATABASE IF NOT EXISTS RailwayReservation;

USE RailwayReservation;


-- 1. Passenger Table with Concession Category

DROP TABLE IF EXISTS Passenger;

CREATE TABLE Passenger (

    PassengerID INT AUTO_INCREMENT PRIMARY KEY,

    Name VARCHAR(100) NOT NULL,
```

```
    Age INT,
    Gender VARCHAR(10),
    Phone VARCHAR(15),
    Email VARCHAR(100),
    ConcessionCategory ENUM('None','Senior Citizen','Student','Disabled') DEFAULT 'None'
);

-- 2. Train Table
DROP TABLE IF EXISTS Train;
CREATE TABLE Train (
    TrainID INT AUTO_INCREMENT PRIMARY KEY,
    TrainName VARCHAR(100) NOT NULL,
    StationA VARCHAR(100),
    StationB VARCHAR(100)
);

-- 3. Station Table
DROP TABLE IF EXISTS Station;
CREATE TABLE Station (
    StationID INT AUTO_INCREMENT PRIMARY KEY,
    StationName VARCHAR(100) NOT NULL,
    Location VARCHAR(100)
);

-- 4. Route Table for managing train routes between stations
DROP TABLE IF EXISTS Route;
CREATE TABLE Route (
    RouteID INT AUTO_INCREMENT PRIMARY KEY,
    TrainID INT,
```

```
    SourceStationID INT,
    DestinationStationID INT,
    Distance DECIMAL(7,2),
    Duration TIME,
    FOREIGN KEY (TrainID) REFERENCES Train(TrainID),
    FOREIGN KEY (SourceStationID) REFERENCES Station(StationID),
    FOREIGN KEY (DestinationStationID) REFERENCES Station(StationID)
);

-- 5. Schedule Table for train timings and travel dates
DROP TABLE IF EXISTS Schedule;
CREATE TABLE Schedule (
    ScheduleID INT AUTO_INCREMENT PRIMARY KEY,
    RouteID INT,
    StartTime DATETIME,
    FOREIGN KEY (RouteID) REFERENCES Route(RouteID)
);

-- 6. Class Table
DROP TABLE IF EXISTS Class;
CREATE TABLE Class (
    ClassID INT AUTO_INCREMENT PRIMARY KEY,
    ClassName VARCHAR(50) NOT NULL,
    Fare DECIMAL(10,2)
);

-- 7. Seat Table
DROP TABLE IF EXISTS Seat;
CREATE TABLE Seat (
```

```
    SeatID INT AUTO_INCREMENT PRIMARY KEY,
    RouteID INT,
    ClassID INT,
    SeatNumber VARCHAR(10),
    Status VARCHAR(20) DEFAULT 'Available',  -- Available, Booked, etc.
    FOREIGN KEY (ClassID) REFERENCES Class(ClassID),
    FOREIGN KEY (RouteID) REFERENCES Route(RouteID)
);

-- 8. Ticket Table with new BookingSource and expanded Status (includes RAC)
DROP TABLE IF EXISTS Ticket;
CREATE TABLE Ticket (
    TicketID     INT AUTO_INCREMENT PRIMARY KEY,
    PNR          VARCHAR(20),
    PassengerID  INT,
    SeatID       INT NULL,
    BookingDate  DATE,
    TravelDate   DATE,
    ClassID      INT,
    RouteID      INT,
    Status       ENUM('Booked','Cancelled','Waitlist','RAC') DEFAULT 'Booked',
    BookingSource ENUM('Online','Counter') NOT NULL,
    INDEX idx_pnr (PNR),
    FOREIGN KEY (PassengerID) REFERENCES Passenger(PassengerID),
    FOREIGN KEY (SeatID)      REFERENCES Seat(SeatID),
    FOREIGN KEY (ClassID)     REFERENCES Class(ClassID),
    FOREIGN KEY (RouteID)     REFERENCES Route(RouteID)
);
```

```
-- 9. Payment Table with RefundAmount for refund processing
DROP TABLE IF EXISTS Payment;
CREATE TABLE Payment (
    PaymentID INT AUTO_INCREMENT PRIMARY KEY,
    TicketID INT,
    PaymentDate DATE,
    Amount DECIMAL(10,2),
    PaymentMode VARCHAR(50),  -- e.g., Credit Card, Debit Card, Net Banking, etc.
    RefundStatus VARCHAR(20) DEFAULT 'Not Refunded',  -- Not Refunded, Refunded, etc.
    RefundAmount DECIMAL(10,2) DEFAULT 0,
    FOREIGN KEY (TicketID) REFERENCES Ticket(TicketID)
);

-- 10. Waitlist Table updated with RequestType for Waitlist or RAC
DROP TABLE IF EXISTS Waitlist;
CREATE TABLE Waitlist (
    WaitlistID INT AUTO_INCREMENT PRIMARY KEY,
    TicketID INT,           -- References the waitlisted ticket from Ticket table
    RouteID INT,            -- The route for which the waitlisting applies
    ClassID INT,            -- The class of travel for which the waitlisting applies
    TravelDate DATE,        -- The travel date from the Ticket record
    WaitlistNumber INT,     -- Ordering number for prioritizing waitlisted tickets
    RequestType ENUM('Waitlist','RAC') DEFAULT 'Waitlist',
    FOREIGN KEY (TicketID) REFERENCES Ticket(TicketID)
);
```

-- Sample Data Population

-- Passenger Sample Data with ConcessionCategory specified

INSERT INTO Passenger (Name, Age, Gender, Phone, Email, ConcessionCategory) VALUES

('Ramesh Kumar', 45, 'Male', '9876543210', 'ramesh.kumar@example.com', 'None'),

('Sunita Sharma', 19, 'Female', '9123456780', 'sunita.sharma@example.com', 'Student'),

('Sushila Devi', 75, 'Female', '9988776655', 'sushila.devi@example.com', 'Senior Citizen'),

('Vijay Singh', 38, 'Male', '9876501234', 'vijay.singh@example.com', 'None'),

('Priya Patel', 17, 'Female', '9123409876', 'priya.patel@example.com', 'Student'),

('Amitabh Roy', 52, 'Male', '9876123450', 'amitabh.roy@example.com', 'None'),

('Meera Iyer', 30, 'Female', '9123678901', 'meera.iyer@example.com', 'Disabled'),

('Rahul Verma', 50, 'Male', '9876098765', 'rahul.verma@example.com', 'None'),

('Shanti Rao', 80, 'Female', '9123456701', 'shanti.rao@example.com', 'Senior Citizen'),

('Ravi Kumar', 25, 'Male', '9876509876', 'ravi.kumar@example.com', 'None'),

('Anjali Mehta', 22, 'Female', '9112233445', 'anjali.mehta@example.com', 'Student'),

('Devansh Kapoor', 65, 'Male', '9823456789', 'devansh.kapoor@example.com', 'Senior Citizen'),

('Nisha Singh', 29, 'Female', '9123344556', 'nisha.singh@example.com', 'None'),

('Kabir Joshi', 16, 'Male', '9876540011', 'kabir.joshi@example.com', 'Student'),

('Farhan Ali', 48, 'Male', '9832109876', 'farhan.ali@example.com', 'None'),

('Leela Nair', 70, 'Female', '9845671234', 'leela.nair@example.com', 'Senior Citizen'),

('Akshay Rana', 40, 'Male', '9856782345', 'akshay.rana@example.com', 'None'),

('Pooja Desai', 33, 'Female', '9867893456', 'pooja.desai@example.com', 'Disabled'),

('Manoj Tripathi', 55, 'Male', '9878904567', 'manoj.tripathi@example.com', 'None'),

('Tanya Bhatt', 19, 'Female', '9889015678', 'tanya.bhatt@example.com', 'Student');

-- Train Sample Data

INSERT INTO Train (TrainName, StationA, StationB) VALUES

('Mumbai-Delhi Express', 'Mumbai', 'Delhi'),

('Chennai-Kolkata Express', 'Chennai', 'Kolkata'),

('Chennai-Mumbai Express', 'Chennai', 'Mumbai'),

('Bangalore-Hyderabad Express', 'Bangalore', 'Hyderabad'),

('Ahmedabad-Jaipur Express', 'Ahmedabad', 'Jaipur'),

('Mumbai-Pune Express', 'Mumbai', 'Pune'),

('Mumbai-Kolkata Express', 'Mumbai', 'Kolkata'),

('Pune-Nagpur Express', 'Pune', 'Nagpur');

-- Station Sample Data

INSERT INTO Station (StationName, Location) VALUES

('Mumbai Central', 'Mumbai'),

('New Delhi', 'Delhi'),

('Chennai Central', 'Chennai'),

('Howrah', 'Kolkata'),

('Bangalore City', 'Bangalore'),

('Hyderabad Deccan', 'Hyderabad'),

('Ahmedabad Junction', 'Ahmedabad'),

('Jaipur Station', 'Jaipur'),

('Pune Junction', 'Pune'),

('Nagpur Central', 'Nagpur');

-- Route Sample Data (one per train)

INSERT INTO Route (TrainID, SourceStationID, DestinationStationID, Distance, Duration) VALUES

(1, 1, 2, 1400, '16:00:00'),

(1, 2, 1, 1400, '16:00:00'),

(2, 3, 4, 1500, '18:00:00'),

(2, 4, 3, 1500, '18:00:00'),

(3, 3, 1, 1000, '12:00:00'),

```
(3, 1, 3, 1000, '12:00:00'),
(4, 5, 6, 600, '08:00:00'),
(4, 6, 5, 600, '08:00:00'),
(5, 7, 8, 800, '10:00:00'),
(5, 8, 7, 800, '10:00:00'),
(6, 1, 9, 150, '01:30:00'),
(6, 9, 1, 150, '01:30:00'),
(7, 1, 4, 1500, '16:30:00'),
(7, 4, 1, 1500, '16:30:00'),
(8, 9, 10, 250, '02:00:00'),
(8, 10, 9, 250, '02:00:00');

-- Schedule Sample Data (2 schedules per train)
INSERT INTO Schedule (RouteID, StartTime) VALUES
(1, '2025-04-21 14:00:00'),
(2, '2025-04-22 08:00:00'),
(3, '2025-04-21 10:00:00'),
(4, '2025-04-22 06:00:00'),
(5, '2025-04-21 11:00:00'),
(6, '2025-04-21 23:30:00'),
(7, '2025-04-21 14:00:00'),
(8, '2025-04-21 23:00:00'),
(9, '2025-04-21 16:00:00'),
(10, '2025-04-22 03:00:00'),
(11, '2025-04-21 15:00:00'),
(12, '2025-04-21 17:00:00'),
(13, '2025-04-21 10:00:00'),
(14, '2025-04-22 02:00:00'),
(15, '2025-04-21 09:00:00'),
```

(16, '2025-04-21 11:30:00');

-- Class Sample Data

INSERT INTO Class (ClassName, Fare) VALUES

('Sleeper', 500),

('AC 3-tier', 1000),

('AC 2-tier', 1500),

('First Class', 2500);

-- Seat Sample Data (for each train & class; simplified example)

-- For Train 1

-- Train 1 Route 1

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES

(1, 1, 'SL01', 'Available'), (1, 1, 'SL02', 'Available'), (1, 1, 'SL03', 'Available'), (1, 1, 'SL04', 'Available'),

(1, 2, '3A01', 'Available'), (1, 2, '3A02', 'Available'), (1, 2, '3A03', 'Available'), (1, 2, '3A04', 'Available'),

(1, 3, '2A01', 'Available'), (1, 3, '2A02', 'Available'), (1, 3, '2A03', 'Available'), (1, 3, '2A04', 'Available'),

(1, 4, '1A01', 'Available'), (1, 4, '1A02', 'Available'), (1, 4, '1A03', 'Available'), (1, 4, '1A04', 'Available');

-- Train 1 Route 2

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES

(2, 1, 'SL01', 'Available'), (2, 1, 'SL02', 'Available'), (2, 1, 'SL03', 'Available'), (2, 1, 'SL04', 'Available'),

(2, 2, '3A01', 'Available'), (2, 2, '3A02', 'Available'), (2, 2, '3A03', 'Available'), (2, 2, '3A04', 'Available'),

(2, 3, '2A01', 'Available'), (2, 3, '2A02', 'Available'), (2, 3, '2A03', 'Available'), (2, 3, '2A04', 'Available'),

(2, 4, '1A01', 'Available'), (2, 4, '1A02', 'Available'), (2, 4, '1A03', 'Available'), (2, 4, '1A04', 'Available');

-- Train 2 Route 1

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES

(3, 1, 'SL01', 'Available'), (3, 1, 'SL02', 'Available'), (3, 1, 'SL03', 'Available'), (3, 1, 'SL04', 'Available'),

(3, 2, '3A01', 'Available'), (3, 2, '3A02', 'Available'), (3, 2, '3A03', 'Available'), (3, 2, '3A04', 'Available'),

(3, 3, '2A01', 'Available'), (3, 3, '2A02', 'Available'), (3, 3, '2A03', 'Available'), (3, 3, '2A04', 'Available'),

(3, 4, '1A01', 'Available'), (3, 4, '1A02', 'Available'), (3, 4, '1A03', 'Available'), (3, 4, '1A04', 'Available');

-- Train 2 Route 2

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES

(4, 1, 'SL01', 'Available'), (4, 1, 'SL02', 'Available'), (4, 1, 'SL03', 'Available'), (4, 1, 'SL04', 'Available'),

(4, 2, '3A01', 'Available'), (4, 2, '3A02', 'Available'), (4, 2, '3A03', 'Available'), (4, 2, '3A04', 'Available'),

(4, 3, '2A01', 'Available'), (4, 3, '2A02', 'Available'), (4, 3, '2A03', 'Available'), (4, 3, '2A04', 'Available'),

(4, 4, '1A01', 'Available'), (4, 4, '1A02', 'Available'), (4, 4, '1A03', 'Available'), (4, 4, '1A04', 'Available');

-- Train 3

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES

(5, 1, 'SL01', 'Available'), (5, 1, 'SL02', 'Available'), (5, 1, 'SL03', 'Available'), (5, 1, 'SL04', 'Available'),

(5, 2, '3A01', 'Available'), (5, 2, '3A02', 'Available'), (5, 2, '3A03', 'Available'), (5, 2, '3A04', 'Available'),

(5, 3, '2A01', 'Available'), (5, 3, '2A02', 'Available'), (5, 3, '2A03', 'Available'), (5, 3, '2A04', 'Available'),

(5, 4, '1A01', 'Available'), (5, 4, '1A02', 'Available'), (5, 4, '1A03', 'Available'), (5, 4, '1A04', 'Available');

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES

(6, 1, 'SL01', 'Available'), (6, 1, 'SL02', 'Available'), (6, 1, 'SL03', 'Available'), (6, 1, 'SL04', 'Available'),

(6, 2, '3A01', 'Available'), (6, 2, '3A02', 'Available'), (6, 2, '3A03', 'Available'), (6, 2, '3A04', 'Available'),

(6, 3, '2A01', 'Available'), (6, 3, '2A02', 'Available'), (6, 3, '2A03', 'Available'), (6, 3, '2A04', 'Available'),

(6, 4, '1A01', 'Available'), (6, 4, '1A02', 'Available'), (6, 4, '1A03', 'Available'), (6, 4, '1A04', 'Available');

-- Train 4

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES

(7, 1, 'SL01', 'Available'), (7, 1, 'SL02', 'Available'), (7, 1, 'SL03', 'Available'), (7, 1, 'SL04', 'Available'),

(7, 2, '3A01', 'Available'), (7, 2, '3A02', 'Available'), (7, 2, '3A03', 'Available'), (7, 2, '3A04', 'Available'),

(7, 3, '2A01', 'Available'), (7, 3, '2A02', 'Available'), (7, 3, '2A03', 'Available'), (7, 3, '2A04', 'Available'),

(7, 4, '1A01', 'Available'), (7, 4, '1A02', 'Available'), (7, 4, '1A03', 'Available'), (7, 4, '1A04', 'Available');

```
INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES
(8, 1, 'SL01', 'Available'), (8, 1, 'SL02', 'Available'), (8, 1, 'SL03', 'Available'), (8, 1, 'SL04', 'Available'),
(8, 2, '3A01', 'Available'), (8, 2, '3A02', 'Available'), (8, 2, '3A03', 'Available'), (8, 2, '3A04', 'Available'),
(8, 3, '2A01', 'Available'), (8, 3, '2A02', 'Available'), (8, 3, '2A03', 'Available'), (8, 3, '2A04', 'Available'),
(8, 4, '1A01', 'Available'), (8, 4, '1A02', 'Available'), (8, 4, '1A03', 'Available'), (8, 4, '1A04', 'Available');

-- Train 5
INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES
(9, 1, 'SL01', 'Available'), (9, 1, 'SL02', 'Available'), (9, 1, 'SL03', 'Available'), (9, 1, 'SL04', 'Available'),
(9, 2, '3A01', 'Available'), (9, 2, '3A02', 'Available'), (9, 2, '3A03', 'Available'), (9, 2, '3A04', 'Available'),
(9, 3, '2A01', 'Available'), (9, 3, '2A02', 'Available'), (9, 3, '2A03', 'Available'), (9, 3, '2A04', 'Available'),
(9, 4, '1A01', 'Available'), (9, 4, '1A02', 'Available'), (9, 4, '1A03', 'Available'), (9, 4, '1A04', 'Available');

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES
(10, 1, 'SL01', 'Available'), (10, 1, 'SL02', 'Available'), (10, 1, 'SL03', 'Available'), (10, 1, 'SL04', 'Available'),
(10, 2, '3A01', 'Available'), (10, 2, '3A02', 'Available'), (10, 2, '3A03', 'Available'), (10, 2, '3A04', 'Available'),
(10, 3, '2A01', 'Available'), (10, 3, '2A02', 'Available'), (10, 3, '2A03', 'Available'), (10, 3, '2A04', 'Available'),
(10, 4, '1A01', 'Available'), (10, 4, '1A02', 'Available'), (10, 4, '1A03', 'Available'), (10, 4, '1A04', 'Available');

-- You can continue in the same format for RouteID 11 to 16
-- Train 6
INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES
(11, 1, 'SL01', 'Available'), (11, 1, 'SL02', 'Available'), (11, 1, 'SL03', 'Available'), (11, 1, 'SL04', 'Available'),
(11, 2, '3A01', 'Available'), (11, 2, '3A02', 'Available'), (11, 2, '3A03', 'Available'), (11, 2, '3A04', 'Available'),
(11, 3, '2A01', 'Available'), (11, 3, '2A02', 'Available'), (11, 3, '2A03', 'Available'), (11, 3, '2A04', 'Available'),
(11, 4, '1A01', 'Available'), (11, 4, '1A02', 'Available'), (11, 4, '1A03', 'Available'), (11, 4, '1A04', 'Available');

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES
```

(12, 1, 'SL01', 'Available'), (12, 1, 'SL02', 'Available'), (12, 1, 'SL03', 'Available'), (12, 1, 'SL04', 'Available'),
(12, 2, '3A01', 'Available'), (12, 2, '3A02', 'Available'), (12, 2, '3A03', 'Available'), (12, 2, '3A04', 'Available'),
(12, 3, '2A01', 'Available'), (12, 3, '2A02', 'Available'), (12, 3, '2A03', 'Available'), (12, 3, '2A04', 'Available'),
(12, 4, '1A01', 'Available'), (12, 4, '1A02', 'Available'), (12, 4, '1A03', 'Available'), (12, 4, '1A04', 'Available');

-- Train 7

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES
(13, 1, 'SL01', 'Available'), (13, 1, 'SL02', 'Available'), (13, 1, 'SL03', 'Available'), (13, 1, 'SL04', 'Available'),
(13, 2, '3A01', 'Available'), (13, 2, '3A02', 'Available'), (13, 2, '3A03', 'Available'), (13, 2, '3A04', 'Available'),
(13, 3, '2A01', 'Available'), (13, 3, '2A02', 'Available'), (13, 3, '2A03', 'Available'), (13, 3, '2A04', 'Available'),
(13, 4, '1A01', 'Available'), (13, 4, '1A02', 'Available'), (13, 4, '1A03', 'Available'), (13, 4, '1A04', 'Available');

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES
(14, 1, 'SL01', 'Available'), (14, 1, 'SL02', 'Available'), (14, 1, 'SL03', 'Available'), (14, 1, 'SL04', 'Available'),
(14, 2, '3A01', 'Available'), (14, 2, '3A02', 'Available'), (14, 2, '3A03', 'Available'), (14, 2, '3A04', 'Available'),
(14, 3, '2A01', 'Available'), (14, 3, '2A02', 'Available'), (14, 3, '2A03', 'Available'), (14, 3, '2A04', 'Available'),
(14, 4, '1A01', 'Available'), (14, 4, '1A02', 'Available'), (14, 4, '1A03', 'Available'), (14, 4, '1A04', 'Available');

-- Train 8

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES
(15, 1, 'SL01', 'Available'), (15, 1, 'SL02', 'Available'), (15, 1, 'SL03', 'Available'), (15, 1, 'SL04', 'Available'),
(15, 2, '3A01', 'Available'), (15, 2, '3A02', 'Available'), (15, 2, '3A03', 'Available'), (15, 2, '3A04', 'Available'),
(15, 3, '2A01', 'Available'), (15, 3, '2A02', 'Available'), (15, 3, '2A03', 'Available'), (15, 3, '2A04', 'Available'),
(15, 4, '1A01', 'Available'), (15, 4, '1A02', 'Available'), (15, 4, '1A03', 'Available'), (15, 4, '1A04', 'Available');

INSERT INTO Seat (RouteID, ClassID, SeatNumber, Status) VALUES
(16, 1, 'SL01', 'Available'), (16, 1, 'SL02', 'Available'), (16, 1, 'SL03', 'Available'), (16, 1, 'SL04', 'Available'),
(16, 2, '3A01', 'Available'), (16, 2, '3A02', 'Available'), (16, 2, '3A03', 'Available'), (16, 2, '3A04', 'Available'),
(16, 3, '2A01', 'Available'), (16, 3, '2A02', 'Available'), (16, 3, '2A03', 'Available'), (16, 3, '2A04', 'Available'),

(16, 4, '1A01', 'Available'), (16, 4, '1A02', 'Available'), (16, 4, '1A03', 'Available'), (16, 4, '1A04', 'Available');

--Data in Ticket and Payment Table will be inserted later on booking tickets

```
mysql> select * from Passenger;
+-------------+----------------+------+--------+------------+---------------------------+--------------------+
| PassengerID | Name           | Age  | Gender | Phone      | Email                     | ConcessionCategory |
+-------------+----------------+------+--------+------------+---------------------------+--------------------+
|           1 | Ramesh Kumar   |   45 | Male   | 9876543210 | ramesh.kumar@example.com  | None               |
|           2 | Sunita Sharma  |   19 | Female | 9123456780 | sunita.sharma@example.com | Student            |
|           3 | Sushila Devi   |   75 | Female | 9988776655 | sushila.devi@example.com  | Senior Citizen     |
|           4 | Vijay Singh    |   38 | Male   | 9876501234 | vijay.singh@example.com   | None               |
|           5 | Priya Patel    |   17 | Female | 9123409876 | priya.patel@example.com   | Student            |
|           6 | Amitabh Roy    |   52 | Male   | 9876123450 | amitabh.roy@example.com   | None               |
|           7 | Meera Iyer     |   30 | Female | 9123678901 | meera.iyer@example.com    | Disabled           |
|           8 | Rahul Verma    |   50 | Male   | 9876098765 | rahul.verma@example.com   | None               |
|           9 | Shanti Rao     |   80 | Female | 9123456701 | shanti.rao@example.com    | Senior Citizen     |
|          10 | Ravi Kumar     |   25 | Male   | 9876509876 | ravi.kumar@example.com    | None               |
|          11 | Anjali Mehta   |   22 | Female | 9112233445 | anjali.mehta@example.com  | Student            |
|          12 | Devansh Kapoor |   65 | Male   | 9823456789 | devansh.kapoor@example.com| Senior Citizen     |
|          13 | Nisha Singh    |   29 | Female | 9123344556 | nisha.singh@example.com   | None               |
|          14 | Kabir Joshi    |   16 | Male   | 9876540011 | kabir.joshi@example.com   | Student            |
|          15 | Farhan Ali     |   48 | Male   | 9832109876 | farhan.ali@example.com    | None               |
|          16 | Leela Nair     |   70 | Female | 9845671234 | leela.nair@example.com    | Senior Citizen     |
|          17 | Akshay Rana    |   40 | Male   | 9856782345 | akshay.rana@example.com   | None               |
|          18 | Pooja Desai    |   33 | Female | 9867893456 | pooja.desai@example.com   | Disabled           |
|          19 | Manoj Tripathi |   55 | Male   | 9878904567 | manoj.tripathi@example.com| None               |
|          20 | Tanya Bhatt    |   19 | Female | 9889015678 | tanya.bhatt@example.com   | Student            |
+-------------+----------------+------+--------+------------+---------------------------+--------------------+
```

```
mysql> select * from Payment;
+-----------+----------+-------------+---------+-------------+--------------+--------------+
| PaymentID | TicketID | PaymentDate | Amount  | PaymentMode | RefundStatus | RefundAmount |
+-----------+----------+-------------+---------+-------------+--------------+--------------+
|         1 |        1 | 2025-04-21  | 1000.00 | Counter     | Not Refunded |         0.00 |
|         2 |        2 | 2025-04-21  |  800.00 | Counter     | Not Refunded |         0.00 |
|         3 |        3 | 2025-04-21  |  500.00 | Counter     | Refunded     |       400.00 |
|         4 |        4 | 2025-04-21  | 1000.00 | Counter     | Not Refunded |         0.00 |
|         5 |        5 | 2025-04-21  |  800.00 | Counter     | Not Refunded |         0.00 |
|         6 |        6 | 2025-04-21  |  500.00 | Counter     | Not Refunded |         0.00 |
|         7 |        7 | 2025-04-21  |  400.00 | Counter     | Not Refunded |         0.00 |
|         8 |        8 | 2025-04-21  |  250.00 | Counter     | Not Refunded |         0.00 |
|         9 |        9 | 2025-04-21  |  500.00 | Counter     | Not Refunded |         0.00 |
|        10 |       10 | 2025-04-21  |  400.00 | Counter     | Not Refunded |         0.00 |
|        11 |       11 | 2025-04-21  |  500.00 | Counter     | Not Refunded |         0.00 |
|        12 |       12 | 2025-04-21  | 1000.00 | Counter     | Refunded     |       800.00 |
+-----------+----------+-------------+---------+-------------+--------------+--------------+
```

Sample Queries

-- 1. PNR status tracking for a given ticket

SELECT t.TicketID, t.PNR, t.Status, p.Name, t.TravelDate

FROM Ticket t

JOIN Passenger p ON t.PassengerID = p.PassengerID

WHERE t.PNR = 'PNR81573';

TicketID	PNR	Status	Name	TravelDate
1	PNR81573	Booked	Ramesh Kumar	2025-04-21
2	PNR81573	Booked	Sunita Sharma	2025-04-21
3	PNR81573	Cancelled	Sushila Devi	2025-04-21
4	PNR81573	Booked	Vijay Singh	2025-04-21
5	PNR81573	Booked	Priya Patel	2025-04-21

-- 2. Train schedule lookup for a given train

```
SELECT
   t.TrainID,
   t.TrainName,
   r.RouteID,
   s.ScheduleID,
   s.StartTime,
   st1.StationName AS SourceStation,
   st2.StationName AS DestinationStation
FROM Train t
JOIN Route r ON t.TrainID = r.TrainID
JOIN Schedule s ON r.RouteID = s.RouteID
JOIN Station st1 ON r.SourceStationID = st1.StationID
JOIN Station st2 ON r.DestinationStationID = st2.StationID
WHERE t.TrainID = 1
ORDER BY s.StartTime;
```

TrainID	TrainName	RouteID	ScheduleID	StartTime	SourceStation	DestinationStation
1	Mumbai-Delhi Express	1	1	2025-04-21 14:00:00	Mumbai Central	New Delhi
1	Mumbai-Delhi Express	2	2	2025-04-22 08:00:00	New Delhi	Mumbai Central

-- 3. Available seats query for a specific train, date, and class

```
SELECT s.SeatID, s.SeatNumber
FROM Seat s
```

```
JOIN Route r ON s.RouteID = r.RouteID
JOIN Schedule sch ON r.RouteID = sch.RouteID
WHERE r.TrainID = 1
AND DATE(sch.StartTime) = '2025-04-21'
AND s.ClassID = 1
AND s.Status = 'Available';
```

SeatID	SeatNumber
1	SL01
2	SL02
3	SL03
4	SL04

```
-- 4. List all passengers traveling on a specific train on a given date
SELECT p.PassengerID, p.Name, p.Age, p.Gender, t.PNR, t.Status
FROM Ticket t
JOIN Passenger p ON t.PassengerID = p.PassengerID
JOIN Route r ON t.RouteID = r.RouteID
WHERE r.TrainID = 1
AND t.TravelDate = '2025-04-21';
```

PassengerID	Name	Age	Gender	PNR	Status
1	Ramesh Kumar	45	Male	PNR81573	Booked
2	Sunita Sharma	19	Female	PNR81573	Booked
3	Sushila Devi	75	Female	PNR81573	Cancelled
4	Vijay Singh	38	Male	PNR81573	Booked
5	Priya Patel	17	Female	PNR81573	Booked

```
-- 5. Retrieve all waitlisted/RAC passengers for a particular train
SELECT w.WaitlistID, t.TicketID, t.PNR, w.WaitlistNumber, w.RequestType
FROM Waitlist w
JOIN Ticket t ON w.TicketID = t.TicketID
```

```
JOIN Route r ON t.RouteID = r.RouteID
WHERE r.TrainID = 2
ORDER BY w.RequestType, w.WaitlistNumber;
```

WaitlistID	TicketID	PNR	WaitlistNumber	RequestType
2	10	PNR09247	1	RAC
3	11	PNR93199	2	RAC

```
-- 6. Total amount that needs to be refunded for cancelling a train
SELECT SUM(pmt.RefundAmount) AS TotalRefund
FROM Payment pmt
JOIN Ticket t ON pmt.TicketID = t.TicketID
JOIN Route r ON t.RouteID = r.RouteID
WHERE r.TrainID = 1
AND t.TravelDate = '2025-04-21'
AND t.Status = 'Booked';
```

TotalRefund
3600.00

PaymentID	TicketID	PaymentDate	Amount	PaymentMode	RefundStatus	RefundAmount
1	1	2025-04-21	1000.00	Counter	Not Refunded	0.00
2	2	2025-04-21	800.00	Counter	Not Refunded	0.00
3	3	2025-04-21	500.00	Counter	Refunded	400.00
4	4	2025-04-21	1000.00	Counter	Not Refunded	0.00
5	5	2025-04-21	800.00	Counter	Not Refunded	0.00
6	6	2025-04-21	500.00	Counter	Not Refunded	0.00
7	7	2025-04-21	400.00	Counter	Not Refunded	0.00
8	8	2025-04-21	250.00	Counter	Not Refunded	0.00
9	9	2025-04-21	500.00	Counter	Not Refunded	0.00
10	10	2025-04-21	400.00	Counter	Not Refunded	0.00
11	11	2025-04-21	500.00	Counter	Not Refunded	0.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
-- 7. Total revenue generated from ticket bookings over a specified period
SELECT SUM(pmt.Amount) AS TotalRevenue
```

```
FROM Payment pmt
WHERE pmt.PaymentDate BETWEEN '2025-04-15' AND '2025-04-21';
```

	TotalRevenue
▶	6650.00

```
-- 8. Cancellation records with refund status
SELECT t.TicketID, t.PNR, t.BookingDate, t.TravelDate, pmt.RefundStatus, pmt.RefundAmount
FROM Ticket t
JOIN Payment pmt ON t.TicketID = pmt.TicketID
WHERE t.Status = 'Cancelled';
```

	TicketID	PNR	BookingDate	TravelDate	RefundStatus	RefundAmount
▶	3	PNR81573	2025-04-21	2025-04-21	Refunded	400.00

```
-- 9. Find the busiest route based on passenger count
SELECT
  r.RouteID,
  COUNT(t.TicketID) AS PassengerCount,
  (SELECT StationName FROM Station WHERE StationID = r.SourceStationID) AS Source,
  (SELECT StationName FROM Station WHERE StationID = r.DestinationStationID) AS Destination
FROM Ticket t
JOIN Route r ON t.RouteID = r.RouteID
GROUP BY r.RouteID
HAVING COUNT(t.TicketID) = (
   SELECT MAX(pass_count)
   FROM (
      SELECT COUNT(*) AS pass_count
      FROM Ticket t2
      JOIN Route r2 ON t2.RouteID = r2.RouteID
      GROUP BY r2.RouteID
```

```
    ) AS counts
);
```

	RouteID	PassengerCount	Source	Destination
▶	3	6	Chennai Central	Howrah

```
-- 10. Generate an itemized bill for a ticket including all charges
SELECT
  t.TicketID,
  t.PNR,
  p.Name          AS PassengerName,
  s.SeatNumber    AS SeatNo,
  c.ClassName,
  c.Fare          AS BaseFare,
  pmt.Amount      AS AmountPaid,
  pmt.PaymentDate,
  pmt.PaymentMode,
  pmt.RefundStatus,
  pmt.RefundAmount
FROM Ticket t
JOIN Passenger p ON t.PassengerID = p.PassengerID
LEFT JOIN Seat     s ON t.SeatID     = s.SeatID
JOIN Class     c ON t.ClassID    = c.ClassID
JOIN Payment   pmt ON t.TicketID   = pmt.TicketID
WHERE t.PNR = 'PNR81573'

UNION ALL

SELECT
```

```
  NULL              AS TicketID,
  t.PNR,
  CONCAT('TOTAL (', COUNT(*), ' tickets)') AS PassengerName,
  ''                AS SeatNo,
  ''                AS ClassName,
  0                 AS BaseFare,
  SUM(pmt.Amount)     AS AmountPaid,
  ''                AS PaymentDate,
  ''                AS PaymentMode,
  ''                AS RefundStatus,
  SUM(pmt.RefundAmount) AS RefundAmount
FROM Ticket t
JOIN Payment pmt ON t.TicketID = pmt.TicketID
WHERE t.PNR = 'PNR81573'
GROUP BY t.PNR

ORDER BY
  -- force the TOTAL row to the bottom by treating NULL TicketID as highest
  (TicketID IS NULL),
  TicketID;
```

TicketID	PNR	PassengerName	SeatNo	ClassName	BaseFare	AmountPaid	PaymentDate	PaymentMode	RefundStatus	RefundAmount
1	PNR81573	Ramesh Kumar	3A01	AC 3-tier	1000.00	1000.00	2025-04-21	Counter	Not Refunded	0.00
2	PNR81573	Sunita Sharma	3A02	AC 3-tier	1000.00	800.00	2025-04-21	Counter	Not Refunded	0.00
3	PNR81573	Sushila Devi	NULL	AC 3-tier	1000.00	500.00	2025-04-21	Counter	Refunded	400.00
4	PNR81573	Vijay Singh	3A04	AC 3-tier	1000.00	1000.00	2025-04-21	Counter	Not Refunded	0.00
5	PNR81573	Priya Patel	3A03	AC 3-tier	1000.00	800.00	2025-04-21	Counter	Not Refunded	0.00
NULL	PNR81573	TOTAL (5 tickets)			0.00	4100.00				400.00

Additional Interesting Queries

```
-- 1. Passengers by Concession Category
SELECT p.ConcessionCategory, COUNT(*) AS TotalPassengers
```

```
FROM Passenger p
GROUP BY p.ConcessionCategory;
```

ConcessionCategory	TotalPassengers
None	9
Student	5
Senior Citizen	4
Disabled	2

```
-- 2. Top 5 Trains by Revenue in a Specified Period
SELECT tr.TrainID, tr.TrainName, SUM(pmt.Amount) AS Revenue
FROM Payment pmt
JOIN Ticket t ON pmt.TicketID = t.TicketID
JOIN Route r ON t.RouteID = r.RouteID
JOIN Train tr ON r.TrainID = tr.TrainID
WHERE pmt.PaymentDate BETWEEN '2025-04-01' AND '2025-04-30'
GROUP BY tr.TrainID, tr.TrainName
ORDER BY Revenue DESC
LIMIT 5;
```

TrainID	TrainName	Revenue
1	Mumbai-Delhi Express	4100.00
2	Chennai-Kolkata Express	2550.00

```
-- 3.  Cancellation Percentage per Train
SELECT tr.TrainID, tr.TrainName,
    ROUND(
      (SELECT COUNT(*) FROM Ticket t
        JOIN Route r2 ON t.RouteID = r2.RouteID
        WHERE r2.TrainID = tr.TrainID
          AND t.Status = 'Cancelled'
      ) /
      (SELECT COUNT(*) FROM Ticket t2
```

```
        JOIN Route r3 ON t2.RouteID = r3.RouteID
        WHERE r3.TrainID = tr.TrainID
      ) * 100, 2) AS CancellationPercentage
FROM Train tr;
```

	TrainID	TrainName	CancellationPercentage
▶	1	Mumbai-Delhi Express	20.00
	2	Chennai-Kolkata Express	0.00
	3	Chennai-Mumbai Express	NULL
	4	Bangalore-Hyderabad Express	NULL
	5	Ahmedabad-Jaipur Express	NULL
	6	Mumbai-Pune Express	NULL
	7	Mumbai-Kolkata Express	NULL
	8	Pune-Nagpur Express	NULL

FUNCTIONS/PROCEDURES/TRIGGERS

-- FARE CALCULATION

DELIMITER $$

CREATE FUNCTION fn_calculateFare(in_TicketID INT) RETURNS DECIMAL(10,2) DETERMINISTIC BEGIN DECLARE baseFare DECIMAL(10,2); DECLARE concession VARCHAR(20); DECLARE discount DECIMAL(5,2); DECLARE finalFare DECIMAL(10,2);

```
-- Get base fare from Class (using the Ticket record) Checking class of ticketID
SELECT c.Fare INTO baseFare
  FROM Ticket t
  JOIN Class c ON t.ClassID = c.ClassID
 WHERE t.TicketID = in_TicketID;

-- Get passenger concession category
SELECT p.ConcessionCategory INTO concession
  FROM Ticket t
  JOIN Passenger p ON t.PassengerID = p.PassengerID
```

```
 WHERE t.TicketID = in_TicketID;

SET discount = CASE concession
                 WHEN 'Senior Citizen' THEN 0.50
                 WHEN 'Student'        THEN 0.20
                 WHEN 'Disabled'       THEN 0.30
                 ELSE 0.00
               END;

SET finalFare = baseFare * (1 - discount);
RETURN finalFare;


END$$

DELIMITER ;
```

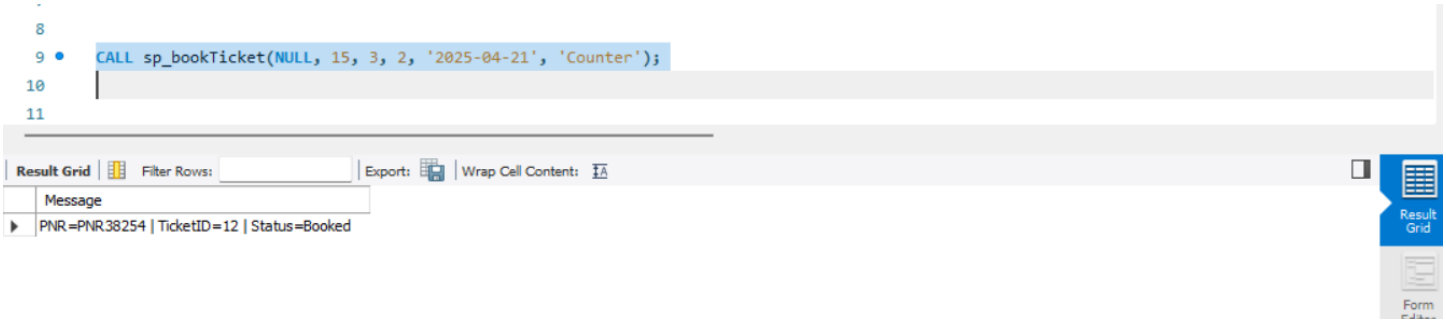
-- BOOK TICKET PROCEDURE

```
DROP PROCEDURE IF EXISTS sp_bookTicket;

DELIMITER $$

CREATE PROCEDURE sp_bookTicket(
   IN in_PNR         VARCHAR(20),
   IN in_PassengerID   INT,
   IN in_RouteID       INT,
   IN in_ClassID       INT,
   IN in_TravelDate    DATE,
   IN in_BookingSource VARCHAR(10)
```

```
)
BEGIN
   DECLARE v_TicketID   INT;
   DECLARE v_PNR_final  VARCHAR(20);
   DECLARE v_Status     VARCHAR(10) DEFAULT 'Booked';
   DECLARE v_Available  INT;
   DECLARE v_RACcount   INT;
   DECLARE v_maxRAC     INT DEFAULT 5;
   DECLARE v_waitNum    INT;
   DECLARE v_seatID     INT;

   -- Decide or reuse PNR
   IF in_PNR IS NULL THEN
      REPEAT
      SET v_PNR_final = CONCAT(
         'PNR',
         LPAD(FLOOR(RAND()*100000), 5, '0')
      );
     UNTIL NOT EXISTS (
          SELECT 1 FROM Ticket WHERE PNR = v_PNR_final
     )
   END REPEAT;
 ELSE
      SET v_PNR_final = in_PNR;
 END IF;

   -- Check seat availability AND pick one seat
   SELECT s.SeatID INTO v_seatID
    FROM Seat s
```

```
  JOIN Schedule sch ON s.RouteID = sch.RouteID
 WHERE s.RouteID = in_RouteID
   AND s.ClassID = in_ClassID
   AND DATE(sch.StartTime) = in_TravelDate
   AND s.Status = 'Available'
 LIMIT 1;

IF v_seatID IS NULL THEN
  -- No seats → RAC vs Waitlist
  SET v_Status = 'RAC';
  SELECT COUNT(*) INTO v_RACcount
    FROM Waitlist
   WHERE RouteID    = in_RouteID
     AND ClassID    = in_ClassID
     AND TravelDate = in_TravelDate
     AND RequestType = 'RAC';
  IF v_RACcount >= v_maxRAC THEN
    SET v_Status = 'Waitlist';
  END IF;
END IF;

-- Insert the ticket (with SeatID if confirmed, NULL otherwise)
INSERT INTO Ticket
  (PNR, PassengerID, SeatID, BookingDate, TravelDate, ClassID, RouteID, Status, BookingSource)
VALUES
  (v_PNR_final, in_PassengerID, v_seatID, CURDATE(), in_TravelDate,
   in_ClassID, in_RouteID, v_Status, in_BookingSource);
SET v_TicketID = LAST_INSERT_ID();
```

```
    -- Mark the seat as Booked if allocated
    IF v_seatID IS NOT NULL THEN
      UPDATE Seat SET Status = 'Booked' WHERE SeatID = v_seatID;
    END IF;

    -- Create payment
    INSERT INTO Payment (TicketID, PaymentDate, Amount, PaymentMode)
    VALUES (v_TicketID, CURDATE(), fn_calculateFare(v_TicketID), in_BookingSource);

    SELECT CONCAT('PNR=', v_PNR_final,
                  ' | TicketID=', v_TicketID,
                  ' | Status=', v_Status) AS Message;
END$$
DELIMITER ;
```



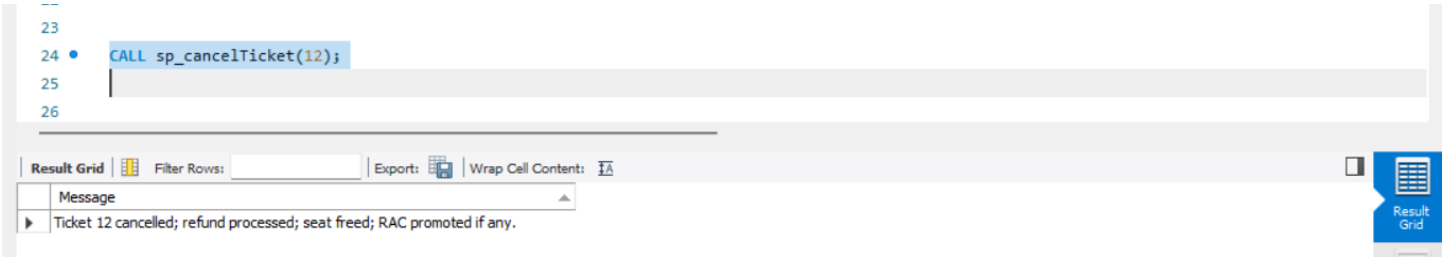
-- Procedure: Cancel Ticket

```
DROP PROCEDURE IF EXISTS sp_cancelTicket;
DELIMITER $$

CREATE PROCEDURE sp_cancelTicket(IN in_TicketID INT)
BEGIN
    DECLARE v_SeatID      INT;
    DECLARE v_RouteID     INT;
```

```
DECLARE v_ClassID     INT;
DECLARE v_TravelDate  DATE;
DECLARE v_waitTicket  INT;
DECLARE v_refundBase  DECIMAL(10,2);

-- 1) Fetch booking details
SELECT SeatID, RouteID, ClassID, TravelDate
  INTO v_SeatID, v_RouteID, v_ClassID, v_TravelDate
FROM Ticket
WHERE TicketID = in_TicketID;

-- 2) Cancel the ticket & clear its SeatID
UPDATE Ticket
  SET Status = 'Cancelled',
      SeatID = NULL
WHERE TicketID = in_TicketID;

-- 3) Calculate concession-adjusted fare, then refund 80% of that
SELECT fn_calculateFare(in_TicketID)
  INTO v_refundBase;

UPDATE Payment p
  JOIN Ticket t ON p.TicketID = t.TicketID
 SET p.RefundAmount = v_refundBase * 0.80,
     p.RefundStatus = 'Refunded'
 WHERE p.TicketID = in_TicketID;

-- 4) Free up the original seat
IF v_SeatID IS NOT NULL THEN
```

```
  UPDATE Seat
     SET Status = 'Available'
   WHERE SeatID = v_SeatID;
END IF;

-- 5) Promote earliest RAC ticket, if any
SELECT TicketID
  INTO v_waitTicket
FROM Waitlist
WHERE RouteID     = v_RouteID
  AND ClassID     = v_ClassID
  AND TravelDate  = v_TravelDate
  AND RequestType = 'RAC'
ORDER BY WaitlistNumber ASC
LIMIT 1;

IF v_waitTicket IS NOT NULL THEN
    -- a) Mark RAC ticket as Booked
    UPDATE Ticket
       SET Status = 'Booked'
     WHERE TicketID = v_waitTicket;

    -- b) Allocate an available seat
    SELECT s.SeatID
      INTO v_SeatID
    FROM Seat s
    JOIN Schedule sch ON s.RouteID = sch.RouteID
    WHERE s.RouteID   = v_RouteID
      AND s.ClassID   = v_ClassID
```

```
        AND s.Status    = 'Available'
        AND DATE(sch.StartTime) = v_TravelDate
      LIMIT 1;

      IF v_SeatID IS NOT NULL THEN
         UPDATE Seat
           SET Status = 'Booked'
          WHERE SeatID = v_SeatID;
         UPDATE Ticket
           SET SeatID = v_SeatID
          WHERE TicketID = v_waitTicket;
      END IF;

      -- c) Remove from waitlist
      DELETE FROM Waitlist
       WHERE TicketID = v_waitTicket;
   END IF;

   -- 6) Return a status message
   SELECT CONCAT('Ticket ', in_TicketID,
                 ' cancelled; refund processed; seat freed; RAC promoted if any.') AS Message;
END$$

DELIMITER ;
```



-- TRIGGER AFTER TICKET INSERT

```
DELIMITER $$

DROP TRIGGER IF EXISTS trg_after_ticket_insert;

DELIMITER $$

CREATE TRIGGER trg_after_ticket_insert

AFTER INSERT ON Ticket FOR EACH ROW

BEGIN

  DECLARE v_waitNum INT;

  IF NEW.Status IN ('RAC','Waitlist') THEN

    SELECT IFNULL(MAX(WaitlistNumber),0) + 1 INTO v_waitNum

      FROM Waitlist

     WHERE RouteID    = NEW.RouteID

       AND ClassID    = NEW.ClassID

       AND TravelDate = NEW.TravelDate

       AND RequestType= NEW.Status;

    INSERT INTO Waitlist

      (TicketID, RouteID, ClassID, TravelDate, WaitlistNumber, RequestType)

    VALUES

      (NEW.TicketID, NEW.RouteID, NEW.ClassID, NEW.TravelDate, v_waitNum, NEW.Status);

  END IF;

END$$

DELIMITER ;
```

-- TRIGGER AFTER CANCELLATION

```
DROP TRIGGER IF EXISTS trg_after_ticket_update;

DELIMITER $$


CREATE TRIGGER trg_after_ticket_update

AFTER UPDATE ON Ticket

FOR EACH ROW

BEGIN

   DECLARE baseFare DECIMAL(10,2);

   DECLARE v_seatID  INT;


   IF NEW.Status = 'Cancelled' AND OLD.Status <> 'Cancelled' THEN

      -- 1) refund 80%

      SELECT Fare INTO baseFare

        FROM Class

       WHERE ClassID = NEW.ClassID;


      UPDATE Payment

        SET RefundAmount = baseFare * 0.80,

            RefundStatus = 'Refunded'

       WHERE TicketID = NEW.TicketID;


      -- 2) free the old seat

      SET v_seatID = OLD.SeatID;  -- we already pulled it into the Ticket row
```

```
        IF v_seatID IS NOT NULL THEN
            UPDATE Seat
              SET Status = 'Available'
            WHERE SeatID = v_seatID;
        END IF;
    END IF;
END$$

DELIMITER ;
```

ER Diagram with Descriptions

Entities and Their Attributes

- **Passenger**
    - *Attributes:*

        - **PassengerID** (Primary Key)
        - **Name**
        - **Age**
        - **Gender**
        - **Phone**
        - **Email**
        - **ConcessionCategory** (ENUM: 'None', 'Senior Citizen', 'Student', 'Disabled')
    - *Description:* Stores information about railway passengers, including possible concession categories.
- **Train**
    - *Attributes:*
        - **TrainID** (Primary Key)
        - **TrainName**
        - **StationA**
        - **StationB**
    - *Description:* Contains basic information about trains and their primary stations.
- **Station**
    - *Attributes:*
        - **StationID** (Primary Key)
        - **StationName**
        - **Location**
    - *Description:* Represents train stations with details on their names and locations.
- **Route**
    - *Attributes:*
        - **RouteID** (Primary Key)
        - **TrainID** (Foreign Key referencing Train)
        - **SourceStationID** (Foreign Key referencing Station)
        - **DestinationStationID** (Foreign Key referencing Station)
        - **Distance**

        - **Duration**
    - *Relationships:*
        - Each route is associated with one train.
        - It also links two stations (source and destination).
    - *Description:* Manages train routes including the distance and travel time between two stations.
- **Schedule**
    - *Attributes:*
        - **ScheduleID** (Primary Key)
        - **RouteID** (Foreign Key referencing Route)
        - **StartTime**
    - *Description:* Stores scheduled timings for each train route.
- **Class**
    - *Attributes:*
        - **ClassID** (Primary Key)
        - **ClassName**
        - **Fare**
    - *Description:* Defines the travel classes (e.g., Sleeper, AC tiers) and their corresponding fares.
- **Seat**
    - *Attributes:*
        - **SeatID** (Primary Key)
        - **RouteID** (Foreign Key referencing Route)
        - **ClassID** (Foreign Key referencing Class)
        - **SeatNumber**
        - **Status** (e.g., 'Available', 'Booked')
    - *Relationships:*
        - Associates a seat with a specific route and travel class.
    - *Description:* Represents individual seats on a train for each route and class.
- **Ticket**
    - *Attributes:*

        - **TicketID** (Primary Key)
        - **PNR**
        - **SeatID**
        - **PassengerID** (Foreign Key referencing Passenger)
        - **BookingDate**
        - **TravelDate**
        - **ClassID** (Foreign Key referencing Class)
        - **SeatID (Foreign key referencing Seat)**
        - **RouteID** (Foreign Key referencing Route)
        - **Status** (ENUM: 'Booked', 'Cancelled', 'Waitlist', 'RAC')
        - **BookingSource** (ENUM: 'Online', 'Counter')
    - *Relationships:*
        - Links a ticket to a passenger, a travel class,a seat, and a route.
    - *Description:* Handles booking details and status for train travel.
- **Payment**
    - *Attributes:*
        - **PaymentID** (Primary Key)
        - **TicketID** (Foreign Key referencing Ticket)
        - **PaymentDate**
        - **Amount**
        - **PaymentMode** (e.g., Credit Card, Debit Card, Net Banking)
        - **RefundStatus** (e.g., 'Not Refunded', 'Refunded')
        - **RefundAmount**
    - *Relationships:*
        - Connected to a ticket to record the corresponding payment details.
    - *Description:* Tracks payment transactions including any refund processed.
- **Waitlist**
    - *Attributes:*
        - **WaitlistID** (Primary Key)

        - **TicketID** (Foreign Key referencing Ticket)
        - **RouteID**
        - **ClassID**
        - **TravelDate**
        - **WaitlistNumber**
        - **RequestType** (ENUM: 'Waitlist', 'RAC')
    - *Relationships:*
        - References a ticket and relates to specific travel route and class.
    - *Description:* Manages waitlisting of tickets when bookings exceed availability or when RAC (Reservation Against Cancellation) is applied.

Relationships Summary

- **Passenger–Ticket:** One-to-many (a passenger can have multiple tickets).
- **Train–Route:** One-to-many (a train can operate multiple routes).
- **Station–Route:** Two one-to-many relationships (each route references a source and a destination station).
- **Route–Schedule:** One-to-many (a route can have multiple schedules).
- **Route–Seat:** One-to-many (each route has several seats allocated).
- **Class–Seat:** One-to-many (each travel class designates its seats).
- **Class–Ticket:** One-to-many (tickets are associated with a specific class).
- **Ticket–Payment:** One-to-one/many (each ticket is linked to one or more payment records).
- **Ticket–Waitlist:** One-to-many (a waitlisted ticket may generate multiple waitlist entries if adjusted).

Relational Schema and Normalization Process

Relational Schema

Below is the concise relational schema derived from the SQL:

- **Passenger(PassengerID, Name, Age, Gender, Phone, Email, ConcessionCategory)**
- **Train(TrainID, TrainName, StationA, StationB)**
- **Station(StationID, StationName, Location)**
- **Route(RouteID, TrainID, SourceStationID, DestinationStationID, Distance, Duration)**
- **Schedule(ScheduleID, RouteID, StartTime)**
- **Class(ClassID, ClassName, Fare)**
- **Seat(SeatID, RouteID, ClassID, SeatNumber, Status)**
- **Ticket(TicketID, PNR, PassengerID, BookingDate, TravelDate, ClassID, RouteID, Status, BookingSource)**
- **Payment(PaymentID, TicketID, PaymentDate, Amount, PaymentMode, RefundStatus, RefundAmount)**
- **Waitlist(WaitlistID, TicketID, RouteID, ClassID, TravelDate, WaitlistNumber, RequestType)**

Normalization Process

1. First Normal Form (1NF): atomicity & no repeating groups

- **Atomic attributes**: every column holds a single, indivisible value (e.g. Name, Age, Gender in **Passenger**; StationName in **Station**).
- **No multi-valued or composite columns**: for example, instead of storing both source and destination stations in one multi-valued field on Train, we introduce a separate **Route** relation linking each TrainID to one SourceStationID and one DestinationStationID.

2. Second Normal Form (2NF): no partial dependencies

Because **every** table in our design uses a single-column primary key (*_ID), there are no composite keys—so **no** non-key attribute can depend on *part* of a key. For example:

- In **Seat**, the PK is SeatID (not the pair (RouteID,ClassID,SeatNumber)), so attributes RouteID, ClassID, SeatNumber, Status all fully depend on that one SeatID.
- In **Ticket**, BookingDate, TravelDate, Status, etc., all depend on the single PK TicketID.

This automatically satisfies 2NF.

3. Third Normal Form (3NF): no transitive dependencies

A relation is in 3NF if **every** non-key attribute depends only on the primary key, not on other non-key fields. You ensure this by:

1. **Factoring out lookup/master data**
    - Fares live only in **Class** (ClassID → Fare), not repeated in **Ticket** or **Payment**.
    - Station details live only in **Station** (StationID → StationName,Location), not repeated in **Train** or **Route**.
2. **Keeping payment details separate**
    - **Payment** holds only financial fields (Amount, PaymentMode, RefundStatus, RefundAmount) keyed by PaymentID (and linked back to **Ticket**). We don't mix passenger or train details here.
3. **Eliminating transitive chains**
    - In **Route**, TrainID → SourceStationID,DestinationStationID,Distance,Duration is direct. There's no attribute like TrainName or StationName hanging off RouteID—you join to **Train** or **Station** instead.
    - **Waitlist** does carry RouteID, ClassID, and TravelDate redundantly alongside TicketID (which itself points to those fields), but that's an intentional performance denormalization; the "pure" 3NF version could omit them and derive via a join on **Ticket**+**Route** if you wanted strict normalization.