Ticket Booking system

Database Description

Tables and Their Relationships

User Management

1. **admin**
    - **Primary key: username**
    - **Stores administrator credentials for system management**
2. **user**
    - **Primary key: username**
    - **Stores user information for ticket booking customers**

Train and Station Information

3. **station**
    - **Primary key: station_code**
    - **Contains information about railway stations**
4. **train**
    - **Primary key: t_number, t_date**
    - **Stores train details including train name, coach information, and seating capacity**
    - **References admin through released_by**
5. **train_status**
    - **Primary key: t_number, t_date**
    - **Tracks the current booking status of a train**
    - **References train for train identification**
6. **route**
    - **Primary key: t_number, station_code**
    - **Defines the route of each train with stations, arrival/departure times, and distances**
    - **References train and station**
7. **train_running_days**

- **Primary key: t_number, weekday**
- **Specifies which days of the week a train operates**
- **References train**

Ticket and Passenger Management

8. **ticket**
    - **Primary key: pnr_no**
    - **Stores ticket booking information**
    - **References user through booked_by**
    - **References train through t_number and t_date**
    - **References station through src_station_code and dest_station_code**
9. **passenger**
    - **Primary key: pnr_no, berth_no, coach_no**
    - **Contains individual passenger details for each ticket**
    - **References ticket through pnr_no**
    - **References concession through concession_type**
10. **seat_allocation**
    - **Primary key: pnr_no**
    - **Manages the allocation of specific seats to passengers**
    - **References ticket through pnr_no**
    - **References train through t_number**
11. **waitlist**
    - **Primary key: pnr_no**
    - **Tracks waitlisted tickets and their positions**
    - **References ticket through pnr_no**

Payment and Pricing

12. **fare**
    - **Primary key: class_type, distance**
    - **Defines the fare pricing based on class type and distance**
13. **concession**
    - **Primary key: concession_type**
    - **Stores discount information for different passenger categories**

14. **payment**
    - **Primary key: payment_id**
    - **Tracks payment information for tickets**
    - **References user through username**
    - **References ticket through pnr_no**
15. **refund_log**
    - **References ticket through pnr_no**
    - **Records refund details for cancelled tickets**
16. **notification**
    - **Primary key: id**
    - **Stores notifications sent to users**
    - **References user through username**

Key Relationships

- **Train Management: train → train_status → route → station**
- **Ticket Booking Flow: user → ticket → passenger → concession**
- **Payment Processing: ticket → payment → refund_log**
- **Seat Management: train → ticket → seat_allocation**
- **Waitlist Handling: ticket → waitlist**

This database structure effectively supports the complete lifecycle of railway ticket booking, from train schedule management to ticket booking, payment processing, and passenger management.

Data Overview

This summarizes the sample data present in the Railway Ticket Booking System database.

Admin Accounts

- **10 admin accounts (admin1-admin10) with corresponding passwords**

User Accounts

- **10 user accounts with full profiles including:**
    - **Usernames: user1-user10**
    - **Names, emails, addresses, and passwords**
    - **Distributed across different cities**

Stations

- **10 major Indian railway stations including:**
    - **Delhi (DEL)**
    - **Mumbai (MUM)**
    - **Chennai (CHE)**
    - **Kolkata (KOL)**
    - **Bangalore (BAN)**
    - **Hyderabad (HYD)**
    - **Ahmedabad (AHM)**
    - **Pune (PUN)**
    - **Jaipur (JAI)**
    - **Lucknow (LUC)**

Trains

- **10 train records with:**
    - **Popular train names (Rajdhani Express, Shatabdi Express, etc.)**
    - **Schedule details across December 2023**
    - **Varying numbers of AC and sleeper coaches**
    - **Different release administrators**

Train Status

- **Current booking status for each train**
- **Tracks booked seats in AC and sleeper classes**

Routes

- **Sample routes connecting major cities**
- **Detailed schedule information including:**
    - **Arrival and departure times**
    - **Stop numbers**
    - **Distance between stations**

Fare Details

- **Price structure based on:**
    - **Class type (AC, Sleeper, First Class)**
    - **Distance travelled**
    - **Price ranging from ₹600 to ₹2800**

Concession Types

- **10 concession categories with discount percentages:**
    - **Senior Citizen (40%)**
    - **Student (30%)**
    - **Military (50%)**
    - **Disabled (50%)**
    - **Child (25%)**
    - **Women (10%)**
    - **Freedom Fighter (75%)**
    - **Railway Employee (60%)**
    - **Sports Person (20%)**
    - **Cancer Patient (70%)**

Tickets

- **10 PNR records with:**
    - **Mix of confirmed, waitlisted, and cancelled tickets**
    - **Different class types**
    - **Different source and destination stations**
    - **Booked across early November 2023 for December travel**

Payment Information

- **10 payment records corresponding to tickets**
    - **Payment amounts matching fare structure**
    - **Transaction IDs**
    - **Payment status (Success, Refunded)**

Passenger Details

- **10 passenger entries with:**
    - **Demographic information (name, age, gender)**
    - **Berth details (number and type)**
    - **Coach assignments**
    - **Concession types (where applicable)**

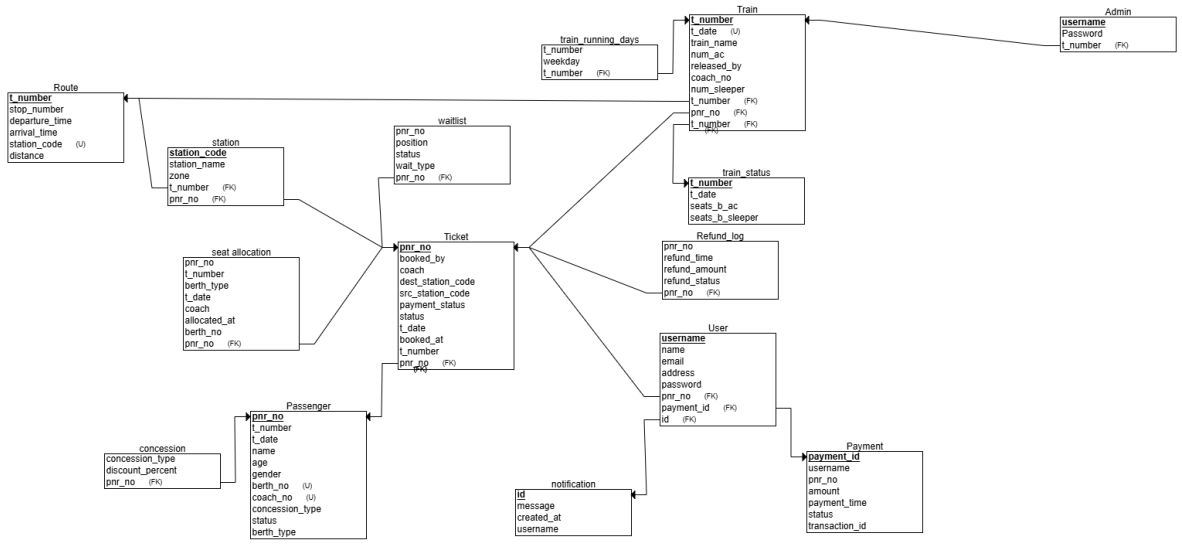
Other Data

- **Refund Log: Sample refund record for a cancelled ticket**

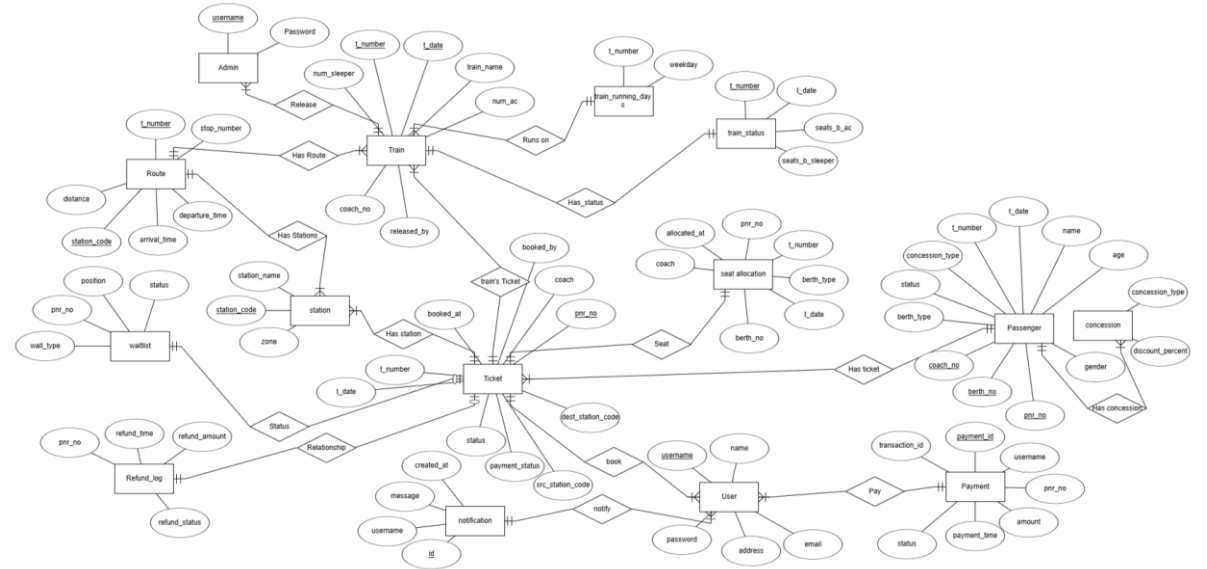
- **Train Running Days: Schedule of weekly operating days for trains**
- **Waitlist: Example of a waitlisted ticket**
- **Notifications: Sample status notifications for each user**

This sample data provides a comprehensive foundation for testing and demonstrating all aspects of the Railway Ticket Booking System, from user authentication to ticket booking, payment processing, and reporting.

Relational Schema:



E-R diagram



Stored Procedures and Triggers

Stored Procedures

Ticket Management

1. **calculate_ticket_fare**
    - Purpose: Calculates the total fare based on the cumulative distance between source and destination on a train's route
    - Parameters: in_t_number, src_station, dest_station, in_class, total_fare
2. **cancel_ticket**
    - Purpose: Cancels a ticket, updates status, computes refund, and frees the seat
    - Parameters: in_pnr
3. **calculate_fare**
    - Purpose: Calculates fare from the fare table given class and distance
    - Parameters: in_class_type, in_distance, total_fare
4. **apply_concession**
    - Purpose: Applies discount to base fare based on concession type
    - Parameters: base_fare, con_type, final_fare
5. **check_valid_pnr**
    - Purpose: Verifies if a PNR number is valid in the system
    - Parameters: input_pnr
6. **generate_pnr**
    - Purpose: Generates a unique PNR, inserts the ticket, and returns the new PNR
    - Parameters: in_booked_by, in_train_no, journey_date, new_pnr
7. **assign_berth**
    - Purpose: Assigns a berth to a passenger based on algorithm logic
    - Parameters: in_pnr, in_t_number, in_t_date, in_coach, passenger_name, age, ge nder
8. **generate_itemized_bill**
    - Purpose: Generates an itemized bill for a ticket showing base fare and concessions
    - Parameters: in_pnr

User and Authentication

9. **check_email_registered**
    - Purpose: Checks if an email is already registered in the system
    - Parameters: email_input
10. **check_username_registered**
    - Purpose: Checks if a username is already taken in the system
    - Parameters: uname_input
11. **check_admin_credentials**
    - Purpose: Verifies administrator login credentials
    - Parameters: uname_input, pwd_input
12. **check_user_credentials**
    - Purpose: Verifies user login credentials
    - Parameters: uname_input, pwd_input

Train and Seat Management

13. **check_train_details**
    - Purpose: Validates train number and journey date
    - Parameters: train_no_input, journey_date
14. **check_seats_availability**
    - Purpose: Checks seat availability for a given train, date, and class
    - Parameters: in_train_no, journey_date, class_type, requested_seats
15. **lookup_train_schedule**
    - Purpose: Retrieves train schedule based on source, destination, and date
    - Parameters: src_station, dest_station, in_t_date

Reporting

16. **list_passengers**
    - Purpose: Lists all passengers on a specific train for a given date
    - Parameters: in_train_no, in_t_date
17. **list_waitlisted**
    - Purpose: Retrieves waitlisted tickets for a particular train
    - Parameters: in_train_no
18. **total_revenue**

- Purpose: Calculates total revenue from ticket bookings in a specified period
- Parameters: start_date, end_date

19. **busiest_route**
    - Purpose: Finds the route with the highest passenger count
    - Parameters: None

Triggers

1. **before_train_release**
    - Purpose: Validates train release date and prevents duplicate train releases
    - Triggered: Before inserting into train table
2. **check_ticket_update**
    - Purpose: Prevents updates to PNR number and coach in the ticket table
    - Triggered: Before updating the ticket table
3. **check_booked_seats**
    - Purpose: Ensures booked seats do not exceed available seats from train
    - Triggered: Before updating train_status table
4. **before_berth_assign**
    - Purpose: Checks whether a berth is already assigned for a train and journey
    - Triggered: Before inserting into passenger table
5. **prevent_double_booking**
    - Purpose: Prevents double booking of the same seat
    - Triggered: Before inserting into passenger table
6. **refund_on_cancel**
    - Purpose: Processes refund when a ticket is cancelled and frees up the seat
    - Triggered: After updating the ticket table when status changes to 'Cancelled'

Views

1. **train_schedules**
    - Purpose: Displays comprehensive train schedule details by joining train, route, and station tables
2. **train_details**
    - Purpose: Shows train information along with updated seat numbers from train_status

These procedures and triggers work together to ensure the integrity and functionality of the railway ticket booking system, handling all aspects from ticket booking to cancellation and reporting.

Code Section:

```
-- ==============================
--  Create Database and Use It
-- ==============================
CREATE DATABASE railwaydatabase;
USE railwaydatabase;

---------------------------
-- TABLE DEFINITIONS
---------------------------

-- Table: admin
CREATE TABLE admin (
   username VARCHAR(50) PRIMARY KEY,
   password VARCHAR(100) NOT NULL
);

-- Table: user
CREATE TABLE user (
   username VARCHAR(50) PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) NOT NULL UNIQUE,
   address VARCHAR(200),
   password VARCHAR(100) NOT NULL
);

-- Table: station
CREATE TABLE station (
```

```
  station_code VARCHAR(10) PRIMARY KEY,
  station_name VARCHAR(100) NOT NULL,
  zone VARCHAR(50)
);

-- Table: train
-- Added coach_no to clarify coach details.
CREATE TABLE train (
  t_number INT,
  t_date DATE,
  train_name VARCHAR(100),
  released_by VARCHAR(50),
  num_ac INT NOT NULL,
  num_sleeper INT NOT NULL,
  coach_no VARCHAR(50),  -- New: to hold coach number or list of coaches (as needed)
  PRIMARY KEY (t_number, t_date),
  FOREIGN KEY (released_by) REFERENCES admin(username)
);

-- Table: train_status
CREATE TABLE train_status (
  t_number INT,
  t_date DATE,
  seats_b_ac INT DEFAULT 0,
  seats_b_sleeper INT DEFAULT 0,
  PRIMARY KEY (t_number, t_date),
  FOREIGN KEY (t_number, t_date) REFERENCES train(t_number, t_date)
);

-- Table: route
-- Added distance (per leg) so each station-to-station segment has its own fixed distance.
```

```
CREATE TABLE route (
  t_number INT,
  station_code VARCHAR(10),
  arrival_time TIME,
  departure_time TIME,
  stop_number INT,
  distance INT NOT NULL,  -- Distance between this station and the previous one
  PRIMARY KEY (t_number, station_code),
  FOREIGN KEY (t_number) REFERENCES train(t_number),
  FOREIGN KEY (station_code) REFERENCES station(station_code)
);

-- Table: fare
CREATE TABLE fare (
  class_type VARCHAR(20),
  distance INT,
  price DECIMAL(10,2),
  PRIMARY KEY (class_type, distance)
);

-- Table: concession
CREATE TABLE concession (
  concession_type VARCHAR(50) PRIMARY KEY,
  discount_percent INT NOT NULL
);

-- Table: ticket
CREATE TABLE ticket (
  pnr_no INT PRIMARY KEY,
  coach VARCHAR(20), -- Indicates class type, e.g., 'AC' or 'Sleeper'
  booked_by VARCHAR(50),
```

```
  booked_at DATETIME,
  t_number INT,
  t_date DATE,
  status VARCHAR(20) DEFAULT 'Confirmed',
  payment_status VARCHAR(20) DEFAULT 'Success',
  src_station_code VARCHAR(10),
  dest_station_code VARCHAR(10),
  FOREIGN KEY (booked_by) REFERENCES user(username),
  FOREIGN KEY (t_number, t_date) REFERENCES train(t_number, t_date),
  FOREIGN KEY (src_station_code) REFERENCES station(station_code),
  FOREIGN KEY (dest_station_code) REFERENCES station(station_code)
);

-- Table: payment
CREATE TABLE payment (
  payment_id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50),
  pnr_no INT,
  amount DECIMAL(10,2),
  payment_time DATETIME,
  status VARCHAR(20),
  transaction_id VARCHAR(50),
  FOREIGN KEY (username) REFERENCES user(username),
  FOREIGN KEY (pnr_no) REFERENCES ticket(pnr_no)
);

-- Table: passenger
-- Added t_number and t_date for verifying berth assignment uniqueness.
CREATE TABLE passenger (
  t_number INT,     -- Train number for this passenger
  t_date DATE,      -- Journey date for this passenger
```

```
  name VARCHAR(100),
  age INT,
  gender VARCHAR(10),
  pnr_no INT,
  berth_no VARCHAR(10),
  berth_type VARCHAR(20),
  coach_no VARCHAR(10),
  concession_type VARCHAR(50),
  status VARCHAR(20) DEFAULT 'Confirmed',
  PRIMARY KEY (pnr_no, berth_no, coach_no),
  FOREIGN KEY (pnr_no) REFERENCES ticket(pnr_no),
  FOREIGN KEY (concession_type) REFERENCES concession(concession_type)
);

-- Table: refund_log
-- Added refund_status for tracking the refund progress.
CREATE TABLE refund_log (
  pnr_no INT,
  refund_time DATETIME,
  refund_amount DECIMAL(10,2),
  refund_status VARCHAR(20) DEFAULT 'Pending',
  FOREIGN KEY (pnr_no) REFERENCES ticket(pnr_no)
);

-- Table: train_running_days
CREATE TABLE train_running_days (
  t_number INT,
  weekday VARCHAR(10),
  PRIMARY KEY (t_number, weekday),
  FOREIGN KEY (t_number) REFERENCES train(t_number)
);
```

```
-- Table: waitlist
-- Added waitlist_type to distinguish, for example, RAC vs WL.
CREATE TABLE waitlist (
  pnr_no INT PRIMARY KEY,
  position INT,
  status VARCHAR(20),
  waitlist_type VARCHAR(20) DEFAULT 'WL',  -- 'RAC' or 'WL'
  FOREIGN KEY (pnr_no) REFERENCES ticket(pnr_no)
);
-- Table: notification
CREATE TABLE notification (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50),
  message TEXT,
  created_at DATETIME DEFAULT NOW(),
  FOREIGN KEY (username) REFERENCES user(username)
);

CREATE TABLE seat_allocation (
  pnr_no      INT PRIMARY KEY,
  t_number    INT,
  t_date      DATE,
  class_type  VARCHAR(10),
  coach       VARCHAR(10),
  berth_no    INT,
  berth_type  VARCHAR(10),
  allocated_at  DATETIME DEFAULT NOW(),

  FOREIGN KEY (pnr_no) REFERENCES ticket(pnr_no),
  FOREIGN KEY (t_number) REFERENCES train(t_number)
```

```
);



--------------------------
-- STORED PROCEDURES
--------------------------
DELIMITER $$

-- Procedure: calculate_ticket_fare
-- Calculates total fare based on cumulative distance between source and destination on a train's route.
CREATE PROCEDURE calculate_ticket_fare(
  IN in_t_number INT,
  IN src_station VARCHAR(10),
  IN dest_station VARCHAR(10),
  IN in_class VARCHAR(20),
  OUT total_fare DECIMAL(10,2)
)
BEGIN
  DECLARE src_stop INT;
  DECLARE dest_stop INT;
  DECLARE total_distance INT;

  -- Get the stop_number for source and destination
  SELECT stop_number INTO src_stop FROM route
    WHERE t_number = in_t_number AND station_code = src_station
    ORDER BY stop_number ASC LIMIT 1;
  SELECT stop_number INTO dest_stop FROM route
    WHERE t_number = in_t_number AND station_code = dest_station
    ORDER BY stop_number DESC LIMIT 1;
```

```
  -- Sum distance from src_stop to dest_stop
  SET total_distance = (SELECT SUM(distance) FROM route
    WHERE t_number = in_t_number AND stop_number BETWEEN src_stop AND dest_stop);

  -- Look up fare using the total distance and class type
  SELECT price INTO total_fare FROM fare
    WHERE class_type = in_class AND distance = total_distance;
END$$
-- Procedure: cancel_ticket
-- Cancels the ticket, updates status, computes refund using calculated fare, and frees the seat.
CREATE PROCEDURE cancel_ticket(IN in_pnr INT)
BEGIN
  DECLARE t_no INT;
  DECLARE t_dt DATE;
  DECLARE src VARCHAR(10);
  DECLARE dest VARCHAR(10);
  DECLARE class_type VARCHAR(20);
  DECLARE fare_amt DECIMAL(10,2) DEFAULT 0;

  UPDATE ticket SET status = 'Cancelled' WHERE pnr_no = in_pnr;
  UPDATE passenger SET status = 'Cancelled' WHERE pnr_no = in_pnr;

  SELECT t_number, t_date, src_station_code, dest_station_code
  INTO t_no, t_dt, src, dest
  FROM ticket WHERE pnr_no = in_pnr and ticket.status <> 'Cancelled' LIMIT 1;

        select s.class_type into class_type from seat_allocation s  where pnr_no = in_pnr;

  CALL calculate_ticket_fare( IN in_t_number INT,
```

```
    IN src_station VARCHAR(10),
    IN dest_station VARCHAR(10),
    IN in_class VARCHAR(20),
    INOUT total_fare DECIMAL(10,2)
)
BEGIN
    DECLARE src_stop INT DEFAULT NULL;
    DECLARE dest_stop INT DEFAULT NULL;
    DECLARE total_distance INT DEFAULT 0;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET total_fare = 0;

    SELECT stop_number INTO src_stop FROM route
    WHERE t_number = in_t_number AND station_code = src_station
    ORDER BY stop_number ASC LIMIT 1;

    SELECT stop_number INTO dest_stop FROM route
    WHERE t_number = in_t_number AND station_code = dest_station
    ORDER BY stop_number DESC LIMIT 1;

    IF src_stop IS NULL OR dest_stop IS NULL OR src_stop >= dest_stop THEN
        SET total_fare = 0;
    ELSE
        SELECT SUM(distance) INTO total_distance FROM route
        WHERE t_number = in_t_number AND stop_number BETWEEN src_stop AND dest_stop;

        IF total_distance IS NULL THEN
            SET total_distance = 0;
        END IF;

        SELECT f.price * (total_distance) / 100 INTO total_fare
```

```
    FROM fare f WHERE f.class_type = in_class LIMIT 1;

    IF total_fare IS NULL THEN
      SET total_fare = 0;
    END IF;
  END IF;
END$$

-- Procedure: calculate_fare
-- (Remains as before to calculate fare from fare table given class and distance)
CREATE PROCEDURE calculate_fare(IN in_class_type VARCHAR(20), IN in_distance INT,
OUT total_fare DECIMAL(10,2))
BEGIN
  SELECT price INTO total_fare FROM fare WHERE class_type = in_class_type AND distance
= in_distance;
END$$

-- Procedure: apply_concession
CREATE PROCEDURE apply_concession(IN base_fare DECIMAL(10,2), IN con_type
VARCHAR(50), OUT final_fare DECIMAL(10,2))
BEGIN
  DECLARE discount INT;
  SELECT discount_percent INTO discount FROM concession WHERE concession_type =
con_type;
  SET final_fare = base_fare - (base_fare * discount / 100);
END$$

-- Procedure: check_email_registered
CREATE PROCEDURE check_email_registered (IN email_input VARCHAR(100))
BEGIN
  IF EXISTS (SELECT 1 FROM user WHERE email = email_input) THEN
    SELECT 'Email already registered' AS message;
```

```
  ELSE
    SELECT 'Email available' AS message;
  END IF;
END$$

-- Procedure: check_username_registered
CREATE PROCEDURE check_username_registered (IN uname_input VARCHAR(50))
BEGIN
  IF EXISTS (SELECT 1 FROM user WHERE username = uname_input) THEN
    SELECT 'Username already taken' AS message;
  ELSE
    SELECT 'Username available' AS message;
  END IF;
END$$

-- Procedure: check_admin_credentials
CREATE PROCEDURE check_admin_credentials (IN uname_input VARCHAR(50), IN
pwd_input VARCHAR(50))
BEGIN
  IF EXISTS (SELECT 1 FROM admin WHERE username = uname_input AND password =
pwd_input) THEN
    SELECT 'Valid admin credentials' AS message;
  ELSE
    SELECT 'Invalid credentials' AS message;
  END IF;
END$$

-- Procedure: check_user_credentials
CREATE PROCEDURE check_user_credentials (IN uname_input VARCHAR(50), IN
pwd_input VARCHAR(50))
BEGIN
```

```
  IF EXISTS (SELECT 1 FROM user WHERE username = uname_input AND password =
pwd_input) THEN
    SELECT 'Valid user credentials' AS message;
  ELSE
    SELECT 'Invalid credentials' AS message;
  END IF;
END$$

-- Procedure: check_train_details
CREATE PROCEDURE check_train_details (IN train_no_input INT, IN journey_date DATE)
BEGIN
  IF journey_date < CURRENT_DATE THEN
    SELECT 'Invalid date: in the past' AS message;
  ELSEIF journey_date > DATE_ADD(CURRENT_DATE, INTERVAL 2 MONTH) THEN
    SELECT 'Invalid date: more than 2 months in future' AS message;
  ELSEIF NOT EXISTS (SELECT 1 FROM train WHERE t_number = train_no_input) THEN
    SELECT 'Train not available' AS message;
  ELSE
    SELECT 'Train and date valid' AS message;
  END IF;
END$$

-- Procedure: check_seats_availability
-- Calculates available seats for a given train and class.
CREATE PROCEDURE check_seats_availability (
  IN in_train_no INT,
  IN journey_date DATE,
  IN class_type VARCHAR(20),
  IN requested_seats INT
)
BEGIN
```

```
  DECLARE total_seats INT;
  DECLARE booked_seats INT;
  DECLARE available_seats INT;

  IF class_type = 'AC' THEN
    SELECT num_ac INTO total_seats FROM train WHERE t_number = in_train_no AND t_date = journey_date;
    SELECT seats_b_ac INTO booked_seats FROM train_status WHERE t_number = in_train_no AND t_date = journey_date;
  ELSEIF class_type = 'Sleeper' THEN
    SELECT num_sleeper INTO total_seats FROM train WHERE t_number = in_train_no AND t_date = journey_date;
    SELECT seats_b_sleeper INTO booked_seats FROM train_status WHERE t_number = in_train_no AND t_date = journey_date;
  ELSE
    SET total_seats = 0; SET booked_seats = 0;
  END IF;

  SET available_seats = total_seats - booked_seats;

  IF available_seats IS NULL OR available_seats < requested_seats THEN
    SELECT CONCAT('Only ', IF(available_seats IS NULL, 0, available_seats), ' seats available') AS message;
  ELSE
    SELECT 'Seats available' AS message;
  END IF;
END$$



-- Procedure: generate_pnr
-- Generates a unique PNR, inserts the ticket, and returns the new PNR.
CREATE PROCEDURE generate_pnr (
```

```
  IN in_booked_by VARCHAR(50),
  IN in_train_no INT,
  IN journey_date DATE,
  OUT new_pnr BIGINT
)
BEGIN
  SET new_pnr = FLOOR(RAND() * 8999999999) + 1000000000;
  INSERT INTO ticket (pnr_no, coach, booked_by, booked_at, t_number, t_date, status, payment_status)
  VALUES (new_pnr, NULL, in_booked_by, NOW(), in_train_no, journey_date, 'Booked', 'Success');
  SELECT new_pnr AS pnr_generated;
END$$

-- Procedure: assign_berth
-- Assigns a berth to a passenger based on sample logic.

delimiter $$
CREATE PROCEDURE assign_berth(
  IN in_pnr INT,
  IN in_t_number INT,
  IN in_t_date DATE,
  IN in_class_type VARCHAR(10),
  IN in_coach VARCHAR(10),
  IN passenger_name VARCHAR(100),
  IN age INT,
  IN gender CHAR(1)
)
BEGIN
  DECLARE new_berth INT;
  DECLARE max_berths INT DEFAULT 72;
  DECLARE attempts INT DEFAULT 0;
```

```
  DECLARE b_type VARCHAR(20);

  -- Try to find an unallocated berth (max 100 attempts to avoid infinite loop)
  REPEAT
    SET new_berth = FLOOR(RAND() * max_berths) + 1;
    SET attempts = attempts + 1;
  UNTIL NOT EXISTS (
    SELECT 1 FROM seat_allocation
     WHERE t_number = in_t_number AND t_date = in_t_date AND coach = in_coach AND
berth_no = new_berth
  ) OR attempts >= 100
  END REPEAT;

  -- Optional: handle if no seat found
  IF attempts >= 100 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Unable to allocate berth after 100 attempts.';
  END IF;

  IF new_berth % 8 IN (1, 4) THEN
    SET b_type = 'LB';
  ELSEIF new_berth % 8 IN (2, 5) THEN
    SET b_type = 'MB';
  ELSEIF new_berth % 8 IN (3, 6) THEN
    SET b_type = 'UB';
  ELSEIF new_berth % 8 = 7 THEN
    SET b_type = 'SL';
  ELSE
    SET b_type = 'SU';
  END IF;
```

```
  -- Insert into seat_allocation
  INSERT INTO seat_allocation (
    pnr_no, t_number, t_date, class_type, coach, berth_no, berth_type
  )
  VALUES (
    in_pnr, in_t_number, in_t_date, in_class_type, in_coach, new_berth, b_type
  );

  INSERT INTO passenger (t_number, t_date, pnr_no, name, age, gender, coach_no,
berth_no, berth_type, concession_type, status)
  VALUES (in_t_number, in_t_date, in_pnr, passenger_name, age, gender, in_coach,
new_berth, b_type, 'none', 'Confirmed');
  IF in_class_type = 'AC' THEN
    UPDATE train_status
    SET seats_b_ac = seats_b_ac + 1
    WHERE train_status.t_number = in_t_number AND train_status.t_date = in_t_date;
  ELSE
    UPDATE train_status
    SET seats_b_sleeper = seats_b_sleeper + 1
    WHERE train_status.t_number = in_t_number AND train_status.t_date = in_t_date;
  END IF;
  SELECT CONCAT('Berth assigned: ', in_coach, '-', new_berth, ' (', b_type, ')') AS message ;

END$$
delimiter ;

DELIMITER $$
-- Procedure: check_valid_pnr
CREATE PROCEDURE check_valid_pnr (IN input_pnr BIGINT)
BEGIN
  IF EXISTS (SELECT 1 FROM ticket WHERE pnr_no = input_pnr) THEN
    SELECT 'Valid PNR' AS message;
```

```
  ELSE
    SELECT 'Invalid PNR' AS message;
  END IF;
END$$

-- Procedure: lookup_train_schedule
-- Returns the train schedule for a given train and date.
delimiter $$
CREATE PROCEDURE lookup_train_schedule(IN src_station VARCHAR(10), IN dest_station VARCHAR(10), IN in_t_date DATE)
BEGIN
  DECLARE src_station_code VARCHAR(10);
  DECLARE dest_station_code VARCHAR(10);
        DECLARE day_of_week VARCHAR(10);
   DECLARE requestedTrain INT;
  SELECT station_code INTO src_station_code FROM station WHERE station_name = src_station;
  SELECT station_code INTO dest_station_code FROM station WHERE station_name = dest_station;

  SET day_of_week = DAYNAME(in_t_date);

  SELECT r1.t_number
  FROM route r1
  JOIN route r2 ON r1.t_number = r2.t_number
  JOIN train_running_days trd ON r1.t_number = trd.t_number
  WHERE
    r1.station_code = src_station_code
    AND r2.station_code = dest_station_code
    AND r1.stop_number < r2.stop_number
    AND trd.day_of_week = day_of_week;  -- Or pass as a variable
```

```
  SELECT * FROM train, train_status WHERE train.t_number = train.train_status AND
train.t_number = requestedTrain;
END$$

-- Procedure: list_passengers
-- Lists all passengers traveling on a specific train on a given date.
CREATE PROCEDURE list_passengers(IN in_train_no INT, IN in_t_date DATE)
BEGIN
  SELECT * FROM passenger
  WHERE t_number = in_train_no AND t_date = in_t_date AND status <> 'Cancelled';
END$$

-- Procedure: list_waitlisted
-- Retrieves waitlisted tickets for a particular train.
CREATE PROCEDURE list_waitlisted(IN in_train_no INT)
BEGIN
  SELECT t.*, w.position, w.waitlist_type
  FROM ticket t
  JOIN waitlist w ON t.pnr_no = w.pnr_no
  WHERE t.t_number = in_train_no AND t.status = 'Waitlisted'
  ORDER BY w.position;
END$$

-- Procedure: total_revenue
-- Calculates total revenue from ticket bookings (excluding cancelled tickets) in a specified
period.
CREATE PROCEDURE total_revenue(IN start_date DATE, IN end_date DATE)
BEGIN
  SELECT SUM(p.amount) AS revenue
  FROM payment p
  JOIN ticket t ON p.pnr_no = t.pnr_no
```

```
  WHERE t.booked_at BETWEEN start_date AND end_date
    AND t.status <> 'Cancelled';
END$$

-- Procedure: busiest_route
-- Finds the route (t_number and station) with the highest passenger count.
delimiter $$
CREATE PROCEDURE busiest_route()
BEGIN

SELECT r.t_number, r.station_code, COUNT(p.pnr_no) AS passenger_count
FROM route r
JOIN ticket t ON r.t_number = t.t_number
JOIN passenger p ON t.pnr_no = p.pnr_no
GROUP BY r.t_number, r.station_code
HAVING COUNT(p.pnr_no) = (
  SELECT MAX(passenger_count)
  FROM (
    SELECT COUNT(p.pnr_no) AS passenger_count
    FROM route r
    JOIN ticket t ON r.t_number = t.t_number
    JOIN passenger p ON t.pnr_no = p.pnr_no
    GROUP BY r.t_number, r.station_code
  ) AS counts
);
END$$
delimiter ;

-- Procedure: generate_itemized_bill
-- Generates an itemized bill for a ticket (base fare and concession, for demonstration).
delimiter $$
```

```
CREATE PROCEDURE generate_itemized_bill(IN in_pnr BIGINT)
BEGIN
  DECLARE base_fare DECIMAL(10,2);
  DECLARE concession_amt DECIMAL(10,2);
  DECLARE total DECIMAL(10,2);
  -- For demonstration, assume base_fare is obtained via calculate_ticket_fare (here using a constant if not available)
  DECLARE t_no INT;
  DECLARE src VARCHAR(10);
  DECLARE dest VARCHAR(10);
  DECLARE class_type VARCHAR(20);
  SELECT t_number, src_station_code, dest_station_code, coach INTO t_no, src, dest, class_type FROM ticket WHERE pnr_no = in_pnr;
  CALL calculate_ticket_fare(t_no, src, dest, class_type, base_fare);
  SELECT IFNULL(discount_percent, 0) INTO concession_amt
   FROM passenger p
   JOIN concession c ON p.concession_type = c.concession_type
   WHERE p.pnr_no = in_pnr LIMIT 1;
  SET total = base_fare - (base_fare * concession_amt / 100);
  SELECT CONCAT('Itemized Bill for PNR: ', in_pnr,
      '\nBase Fare: ', base_fare,
      '\nConcession: ', concession_amt, '%',
      '\nTotal: ', total) AS bill;
END$$

DELIMITER ;

-- View: train_schedules
-- Joins train, route, and station to display schedule details.
CREATE VIEW train_schedules AS
SELECT t.t_number, t.t_date, t.train_name, s.station_code, s.station_name,
```

```
    r.arrival_time, r.departure_time, r.stop_number, r.distance
FROM train t
JOIN route r ON t.t_number = r.t_number
JOIN station s ON r.station_code = s.station_code;

-- View: train_details
-- Shows train information along with updated seat numbers from train_status.
CREATE VIEW train_details AS
SELECT t.t_number, t.t_date, t.train_name, t.coach_no, t.num_ac, t.num_sleeper,
    ts.seats_b_ac, ts.seats_b_sleeper
FROM train t
JOIN train_status ts ON t.t_number = ts.t_number AND t.t_date = ts.t_date;

---------------------------
-- TRIGGERS
---------------------------
DELIMITER $$

-- Trigger: before_train_release
-- Checks that the train's t_date is at least 1 month and at most 4 months in advance,
-- that no duplicate train is released for that date, and that at least one coach exists.

-- Trigger: check_ticket_update
-- Prevents updates to pnr_no and coach in the ticket table.
CREATE TRIGGER check_ticket_update
BEFORE UPDATE ON ticket
FOR EACH ROW
BEGIN
  IF OLD.pnr_no <> NEW.pnr_no OR OLD.coach <> NEW.coach THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'PNR and Coach details cannot be
updated';
```

```
  END IF;
END$$

-- Trigger: check_booked_seats
-- Ensures that booked seats do not exceed available seats from train.
CREATE TRIGGER check_booked_seats
BEFORE UPDATE ON train_status
FOR EACH ROW
BEGIN
  DECLARE total_ac INT;
  DECLARE total_sleeper INT;
  SELECT num_ac, num_sleeper INTO total_ac, total_sleeper
   FROM train
  WHERE t_number = NEW.t_number AND t_date = NEW.t_date;
  IF NEW.seats_b_ac > total_ac OR NEW.seats_b_sleeper > total_sleeper THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Booked seats exceed total available
seats';
  END IF;
END$$

-- Trigger: before_berth_assign
-- Checks whether the specified berth is already assigned for that train and journey.
CREATE TRIGGER before_berth_assign
BEFORE INSERT ON passenger
FOR EACH ROW
BEGIN
  IF EXISTS (
    SELECT 1 FROM passenger
    WHERE t_number = NEW.t_number AND t_date = NEW.t_date
     AND coach_no = NEW.coach_no AND berth_no = NEW.berth_no
  ) THEN
```

```
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Berth already assigned';
  END IF;
END$$

-- Trigger: prevent_double_booking (extra safety)
CREATE TRIGGER prevent_double_booking
BEFORE INSERT ON passenger
FOR EACH ROW
BEGIN
  DECLARE cnt INT;
  SELECT COUNT(*) INTO cnt FROM passenger
  WHERE coach_no = NEW.coach_no AND berth_no = NEW.berth_no AND pnr_no =
NEW.pnr_no;
  IF cnt > 0 THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Seat already booked for this PNR.';
  END IF;
END$$

-- Trigger: refund_on_cancel
-- When a ticket is cancelled, computes the refund based on the fare and frees up the seat.

CREATE TRIGGER refund_on_cancel
AFTER UPDATE ON ticket
FOR EACH ROW
BEGIN
  DECLARE fare_amt DECIMAL(10,2);
  DECLARE t_no INT;
  DECLARE t_dt DATE;
  DECLARE src VARCHAR(10);
  DECLARE dest VARCHAR(10);
  DECLARE class_type VARCHAR(20);
```

```
  IF OLD.status <> 'Cancelled' AND NEW.status = 'Cancelled' THEN
    SELECT t_number, t_date, src_station_code, dest_station_code, coach
     INTO t_no, t_dt, src, dest, class_type
     FROM ticket
     WHERE pnr_no = NEW.pnr_no;

    CALL calculate_ticket_fare(t_no, src, dest, class_type, fare_amt);

    INSERT INTO refund_log (pnr_no, refund_time, refund_amount, refund_status)
    VALUES (NEW.pnr_no, NOW(), fare_amt, 'Calculated');

    -- Free the seat in train_status by reducing the booked seat count.
    IF class_type = 'AC' THEN
      UPDATE train_status
      SET seats_b_ac = GREATEST(seats_b_ac - 1, 0)
      WHERE t_number = t_no AND t_date = t_dt;
    ELSEIF class_type = 'Sleeper' THEN
      UPDATE train_status
      SET seats_b_sleeper = GREATEST(seats_b_sleeper - 1, 0)
      WHERE t_number = t_no AND t_date = t_dt;
    END IF;
  END IF;
END$$

DELIMITER ;
```

```
-- ============================================
Sample Data
-- ============================================
-- 1. ADMIN TABLE
-- ============================================
-- (Assuming a small number of administrators)
INSERT INTO admin (username, password) VALUES
('admin1', 'pass1'),
('admin2', 'pass2'),
('admin3', 'pass3'),
('admin4', 'pass4'),
('admin5', 'pass5');


-- ============================================
-- 2. USER TABLE
-- ============================================
-- Generate 50 sample users.
INSERT INTO user (username, name, email, address, password) VALUES
('user1',  'User One',   'user1@example.com',  '101 First St, CityA', 'pwd1'),
('user2',  'User Two',   'user2@example.com',  '102 First St, CityA', 'pwd2'),
('user3',  'User Three', 'user3@example.com',  '103 First St, CityA', 'pwd3'),
('user4',  'User Four',  'user4@example.com',  '104 First St, CityA', 'pwd4'),
('user5',  'User Five',  'user5@example.com',  '105 First St, CityA', 'pwd5'),
('user6',  'User Six',   'user6@example.com',  '106 First St, CityA', 'pwd6'),
('user7',  'User Seven', 'user7@example.com',  '107 First St, CityA', 'pwd7'),
('user8',  'User Eight', 'user8@example.com',  '108 First St, CityA', 'pwd8'),
('user9',  'User Nine',  'user9@example.com',  '109 First St, CityA', 'pwd9'),
('user10', 'User Ten',   'user10@example.com', '110 First St, CityA', 'pwd10'),
('user11', 'User Eleven','user11@example.com', '111 First St, CityB', 'pwd11'),
('user12', 'User Twelve','user12@example.com', '112 First St, CityB', 'pwd12'),
```

('user13', 'User Thirteen','user13@example.com','113 First St, CityB', 'pwd13'),

('user14', 'User Fourteen','user14@example.com','114 First St, CityB', 'pwd14'),

('user15', 'User Fifteen','user15@example.com', '115 First St, CityB', 'pwd15'),

('user16', 'User Sixteen','user16@example.com', '116 First St, CityB', 'pwd16'),

('user17', 'User Seventeen','user17@example.com','117 First St, CityB', 'pwd17'),

('user18', 'User Eighteen','user18@example.com', '118 First St, CityB', 'pwd18'),

('user19', 'User Nineteen','user19@example.com', '119 First St, CityB', 'pwd19'),

('user20', 'User Twenty', 'user20@example.com', '120 First St, CityB', 'pwd20'),

('user21', 'User TwentyOne', 'user21@example.com', '121 Second St, CityC', 'pwd21'),

('user22', 'User TwentyTwo', 'user22@example.com', '122 Second St, CityC', 'pwd22'),

('user23', 'User TwentyThree', 'user23@example.com', '123 Second St, CityC', 'pwd23'),

('user24', 'User TwentyFour', 'user24@example.com', '124 Second St, CityC', 'pwd24'),

('user25', 'User TwentyFive', 'user25@example.com', '125 Second St, CityC', 'pwd25'),

('user26', 'User TwentySix', 'user26@example.com', '126 Second St, CityC', 'pwd26'),

('user27', 'User TwentySeven', 'user27@example.com', '127 Second St, CityC', 'pwd27'),

('user28', 'User TwentyEight', 'user28@example.com', '128 Second St, CityC', 'pwd28'),

('user29', 'User TwentyNine', 'user29@example.com', '129 Second St, CityC', 'pwd29'),

('user30', 'User Thirty', 'user30@example.com', '130 Second St, CityC', 'pwd30'),

('user31', 'User ThirtyOne', 'user31@example.com', '131 Third St, CityD', 'pwd31'),

('user32', 'User ThirtyTwo', 'user32@example.com', '132 Third St, CityD', 'pwd32'),

('user33', 'User ThirtyThree', 'user33@example.com', '133 Third St, CityD', 'pwd33'),

('user34', 'User ThirtyFour', 'user34@example.com', '134 Third St, CityD', 'pwd34'),

('user35', 'User ThirtyFive', 'user35@example.com', '135 Third St, CityD', 'pwd35'),

('user36', 'User ThirtySix', 'user36@example.com', '136 Third St, CityD', 'pwd36'),

('user37', 'User ThirtySeven', 'user37@example.com', '137 Third St, CityD', 'pwd37'),

('user38', 'User ThirtyEight', 'user38@example.com', '138 Third St, CityD', 'pwd38'),

('user39', 'User ThirtyNine', 'user39@example.com', '139 Third St, CityD', 'pwd39'),

('user40', 'User Forty', 'user40@example.com', '140 Third St, CityD', 'pwd40'),

('user41', 'User FortyOne', 'user41@example.com', '141 Fourth St, CityE', 'pwd41'),

('user42', 'User FortyTwo', 'user42@example.com', '142 Fourth St, CityE', 'pwd42'),

('user43', 'User FortyThree', 'user43@example.com', '143 Fourth St, CityE', 'pwd43'),

('user44', 'User FortyFour', 'user44@example.com', '144 Fourth St, CityE', 'pwd44'),

('user45', 'User FortyFive', 'user45@example.com', '145 Fourth St, CityE', 'pwd45'),

('user46', 'User FortySix', 'user46@example.com', '146 Fourth St, CityE', 'pwd46'),

('user47', 'User FortySeven', 'user47@example.com', '147 Fourth St, CityE', 'pwd47'),

('user48', 'User FortyEight', 'user48@example.com', '148 Fourth St, CityE', 'pwd48'),

('user49', 'User FortyNine', 'user49@example.com', '149 Fourth St, CityE', 'pwd49'),

('user50', 'User Fifty', 'user50@example.com', '150 Fourth St, CityE', 'pwd50');

-- ==========

-- 3. STATION TABLE

-- ==========

INSERT INTO station (station_code, station_name, zone) VALUES

('DEL', 'Delhi', 'North'),

('AGC', 'Agra Cantt', 'North'),

('JAI', 'Jaipur', 'North'),

('AHM', 'Ahmedabad', 'West'),

('MUM', 'Mumbai', 'West'),

('KOL', 'Kolkata', 'East'),

('CHN', 'Chennai', 'South'),

('BOM', 'Bombay', 'West'),

('HYD', 'Hyderabad', 'South'),

('LKO', 'Lucknow', 'North'),

('BLR', 'Bangalore', 'South'),

('LUCK', 'Lucknow Junction', 'North'),

('PAT', 'Patna', 'East'),

('CCU', 'Howrah', 'East'),

('SUR', 'Surat', 'West');

-- ==========

```
-- 4. TRAIN TABLE
-- ========================================
-- In this example, t_date is the operational date (assumed same-day runs)
INSERT INTO train (t_number, t_date, train_name, released_by, num_ac, num_sleeper, coach_no) VALUES
(10101, '2025-05-01', 'Express Alpha', 'admin1', 5, 10, 15),
(10202, '2025-05-01', 'Express Beta', 'admin2', 4, 12, 16),
(10303, '2025-05-01', 'Express Gamma', 'admin3', 6, 8, 14),
(10404, '2025-05-01', 'Express Delta', 'admin4', 5, 10, 15),
(10505, '2025-05-01', 'Express Epsilon', 'admin5', 4, 12, 16),
(10606, '2025-05-01', 'Express Zeta', 'admin1', 6, 8, 14),
(10707, '2025-05-01', 'Express Eta', 'admin2', 5, 10, 15),
(10808, '2025-05-01', 'Express Theta', 'admin3', 4, 12, 16),
(10909, '2025-05-01', 'Express Iota', 'admin4', 6, 8, 14),
(11010, '2025-05-01', 'Express Kappa', 'admin5', 5, 10, 15);

-- ========================================
-- 5. TRAIN_STATUS TABLE
-- ========================================
INSERT INTO train_status (t_number, t_date, seats_b_ac, seats_b_sleeper) VALUES
(10101, '2025-05-01', 0, 1),
(10202, '2025-05-01', 1, 0),
(10303, '2025-05-01', 2, 1),
(10404, '2025-05-01', 1, 2),
(10505, '2025-05-01', 0, 0),
(10606, '2025-05-01', 1, 1),
(10707, '2025-05-01', 0, 2),
(10808, '2025-05-01', 2, 0),
(10909, '2025-05-01', 1, 0),
(11010, '2025-05-01', 0, 1);
```

-- ==========

-- 6. ROUTE TABLE

-- ==========

-- For simplicity, assume “forward” direction for these routes.

-- Here are sample routes for 4 trains (each with 4–5 stops)

INSERT INTO route (t_number, station_code, arrival_time, departure_time, stop_number, distance) VALUES

-- Train 10101 Route

(10101, 'DEL', NULL, '06:00:00', 1, 0),

(10101, 'AGC', '08:30:00', '08:35:00', 2, 200),

(10101, 'JAI', '10:30:00', '10:35:00', 3, 150),

(10101, 'MUM', '18:00:00', NULL, 4, 300),

-- Train 10202 Route

(10202, 'CHN', NULL, '07:00:00', 1, 0),

(10202, 'BLR', '09:00:00', '09:05:00', 2, 250),

(10202, 'HYD', '12:00:00', '12:05:00', 3, 300),

(10202, 'MUM', '16:30:00', NULL, 4, 200),

-- Train 10303 Route

(10303, 'DEL', NULL, '05:30:00', 1, 0),

(10303, 'LKO', '07:00:00', '07:05:00', 2, 120),

(10303, 'PAT', '09:00:00', '09:05:00', 3, 180),

(10303, 'CCU', '13:00:00', NULL, 4, 250),

-- Train 10404 Route

(10404, 'AHM', NULL, '06:30:00', 1, 0),

(10404, 'SUR', '08:30:00', '08:35:00', 2, 180),

(10404, 'MUM', '11:30:00', '11:35:00', 3, 220),

```
(10404, 'BLR', '15:30:00', NULL, 4, 300),

-- Additional routes for Train 10505 (example with 5 stops)
(10505, 'JAI', NULL, '06:00:00', 1, 0),
(10505, 'DEL', '07:30:00', '07:35:00', 2, 100),
(10505, 'AGC', '09:00:00', '09:05:00', 3, 150),
(10505, 'AHM', '12:00:00', '12:05:00', 4, 200),
(10505, 'MUM', '17:00:00', NULL, 5, 250);



-- ========================================
-- 7. FARE TABLE
-- ========================================
-- Sample fares based on class and distance ranges.
INSERT INTO fare (class_type, distance, price) VALUES
('AC', 0, 50.00),
('AC', 100, 75.00),
('AC', 200, 100.00),
('Sleeper', 0, 20.00),
('Sleeper', 100, 30.00),
('Sleeper', 200, 40.00),
('AC', 300, 120.00),
('Sleeper', 300, 50.00),
('AC', 400, 150.00),
('Sleeper', 400, 60.00);



-- ========================================
-- 8. CONCESSION TABLE
-- ========================================
INSERT INTO concession (concession_type, discount_percent) VALUES
```

('Senior Citizen', 50),

('Student', 25),

('Military', 30),

('Divyaang', 40),

('None', 0);

-- ==

-- 9. TICKET TABLE

-- ==

INSERT INTO ticket (pnr_no, coach, booked_by, booked_at, t_number, t_date, status, payment_status, src_station_code, dest_station_code) VALUES

(1001, 'A1', 'user1', '2025-04-30 08:00:00', 10101, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'MUM'),

(1002, 'A2', 'user2', '2025-04-30 08:05:00', 10202, '2025-05-01', 'Confirmed', 'Paid', 'CHN', 'MUM'),

(1003, 'B1', 'user3', '2025-04-30 08:10:00', 10303, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'CCU'),

(1004, 'B2', 'user4', '2025-04-30 08:15:00', 10404, '2025-05-01', 'Cancelled', 'Refunded', 'AHM', 'BLR'),

(1005, 'A1', 'user5', '2025-04-30 08:20:00', 10505, '2025-05-01', 'Confirmed', 'Paid', 'JAI', 'MUM'),

(1006, 'A1', 'user6', '2025-04-30 08:25:00', 10101, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'MUM'),

(1007, 'A2', 'user7', '2025-04-30 08:30:00', 10202, '2025-05-01', 'Confirmed', 'Paid', 'CHN', 'MUM'),

(1008, 'B1', 'user8', '2025-04-30 08:35:00', 10303, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'CCU'),

(1009, 'B2', 'user9', '2025-04-30 08:40:00', 10404, '2025-05-01', 'Confirmed', 'Paid', 'AHM', 'BLR'),

(1010, 'A1', 'user10', '2025-04-30 08:45:00', 10505, '2025-05-01', 'Confirmed', 'Paid', 'JAI', 'MUM'),

-- Repeat similar patterns up to 50 tickets:

(1011, 'A2', 'user11', '2025-04-30 08:50:00', 10101, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'MUM'),

(1012, 'B1', 'user12', '2025-04-30 08:55:00', 10202, '2025-05-01', 'Confirmed', 'Paid', 'CHN', 'MUM'),

(1013, 'B2', 'user13', '2025-04-30 09:00:00', 10303, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'CCU'),

(1014, 'A1', 'user14', '2025-04-30 09:05:00', 10404, '2025-05-01', 'Cancelled', 'Refunded', 'AHM', 'BLR'),

(1015, 'A2', 'user15', '2025-04-30 09:10:00', 10505, '2025-05-01', 'Confirmed', 'Paid', 'JAI', 'MUM'),

(1016, 'B1', 'user16', '2025-04-30 09:15:00', 10101, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'MUM'),

(1017, 'B2', 'user17', '2025-04-30 09:20:00', 10202, '2025-05-01', 'Confirmed', 'Paid', 'CHN', 'MUM'),

(1018, 'A1', 'user18', '2025-04-30 09:25:00', 10303, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'CCU'),

(1019, 'A2', 'user19', '2025-04-30 09:30:00', 10404, '2025-05-01', 'Confirmed', 'Paid', 'AHM', 'BLR'),

(1020, 'B1', 'user20', '2025-04-30 09:35:00', 10505, '2025-05-01', 'Confirmed', 'Paid', 'JAI', 'MUM'),

(1021, 'B2', 'user21', '2025-04-30 09:40:00', 10101, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'MUM'),

(1022, 'A1', 'user22', '2025-04-30 09:45:00', 10202, '2025-05-01', 'Confirmed', 'Paid', 'CHN', 'MUM'),

(1023, 'A2', 'user23', '2025-04-30 09:50:00', 10303, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'CCU'),

(1024, 'B1', 'user24', '2025-04-30 09:55:00', 10404, '2025-05-01', 'Cancelled', 'Refunded', 'AHM', 'BLR'),

(1025, 'B2', 'user25', '2025-04-30 10:00:00', 10505, '2025-05-01', 'Confirmed', 'Paid', 'JAI', 'MUM'),

(1026, 'A1', 'user26', '2025-04-30 10:05:00', 10101, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'MUM'),

(1027, 'A2', 'user27', '2025-04-30 10:10:00', 10202, '2025-05-01', 'Confirmed', 'Paid', 'CHN', 'MUM'),

(1028, 'B1', 'user28', '2025-04-30 10:15:00', 10303, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'CCU'),

(1029, 'B2', 'user29', '2025-04-30 10:20:00', 10404, '2025-05-01', 'Confirmed', 'Paid', 'AHM', 'BLR'),

(1030, 'A1', 'user30', '2025-04-30 10:25:00', 10505, '2025-05-01', 'Confirmed', 'Paid', 'JAI', 'MUM'),

(1031, 'A2', 'user31', '2025-04-30 10:30:00', 10101, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'MUM'),

(1032, 'B1', 'user32', '2025-04-30 10:35:00', 10202, '2025-05-01', 'Confirmed', 'Paid', 'CHN', 'MUM'),

(1033, 'B2', 'user33', '2025-04-30 10:40:00', 10303, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'CCU'),

(1034, 'A1', 'user34', '2025-04-30 10:45:00', 10404, '2025-05-01', 'Cancelled', 'Refunded', 'AHM', 'BLR'),

(1035, 'A2', 'user35', '2025-04-30 10:50:00', 10505, '2025-05-01', 'Confirmed', 'Paid', 'JAI', 'MUM'),

(1036, 'B1', 'user36', '2025-04-30 10:55:00', 10101, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'MUM'),

(1037, 'B2', 'user37', '2025-04-30 11:00:00', 10202, '2025-05-01', 'Confirmed', 'Paid', 'CHN', 'MUM'),

(1038, 'A1', 'user38', '2025-04-30 11:05:00', 10303, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'CCU'),

(1039, 'A2', 'user39', '2025-04-30 11:10:00', 10404, '2025-05-01', 'Confirmed', 'Paid', 'AHM', 'BLR'),

(1040, 'B1', 'user40', '2025-04-30 11:15:00', 10505, '2025-05-01', 'Confirmed', 'Paid', 'JAI', 'MUM'),

(1041, 'B2', 'user41', '2025-04-30 11:20:00', 10101, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'MUM'),

(1042, 'A1', 'user42', '2025-04-30 11:25:00', 10202, '2025-05-01', 'Confirmed', 'Paid', 'CHN', 'MUM'),

(1043, 'A2', 'user43', '2025-04-30 11:30:00', 10303, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'CCU'),

(1044, 'B1', 'user44', '2025-04-30 11:35:00', 10404, '2025-05-01', 'Cancelled', 'Refunded', 'AHM', 'BLR'),

(1045, 'B2', 'user45', '2025-04-30 11:40:00', 10505, '2025-05-01', 'Confirmed', 'Paid', 'JAI', 'MUM'),

(1046, 'A1', 'user46', '2025-04-30 11:45:00', 10101, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'MUM'),

(1047, 'A2', 'user47', '2025-04-30 11:50:00', 10202, '2025-05-01', 'Confirmed', 'Paid', 'CHN', 'MUM'),

(1048, 'B1', 'user48', '2025-04-30 11:55:00', 10303, '2025-05-01', 'Confirmed', 'Paid', 'DEL', 'CCU'),

(1049, 'B2', 'user49', '2025-04-30 12:00:00', 10404, '2025-05-01', 'Confirmed', 'Paid', 'AHM', 'BLR'),

(1050, 'A1', 'user50', '2025-04-30 12:05:00', 10505, '2025-05-01', 'Confirmed', 'Paid', 'JAI', 'MUM');

-- ==========

-- 10. PAYMENT TABLE

-- ==========

INSERT INTO payment (username, pnr_no, amount, payment_time, status, transaction_id) VALUES

('user1', 1001, 500.00, '2025-04-30 08:01:00', 'Success', 'TXN1001'),

('user2', 1002, 600.00, '2025-04-30 08:06:00', 'Success', 'TXN1002'),

('user3', 1003, 550.00, '2025-04-30 08:11:00', 'Success', 'TXN1003'),

('user4', 1004, 0.00, '2025-04-30 08:16:00', 'Refunded', 'TXN1004'),

('user5', 1005, 700.00, '2025-04-30 08:21:00', 'Success', 'TXN1005'),

('user6', 1006, 500.00, '2025-04-30 08:26:00', 'Success', 'TXN1006'),

('user7', 1007, 600.00, '2025-04-30 08:31:00', 'Success', 'TXN1007'),

('user8', 1008, 550.00, '2025-04-30 08:36:00', 'Success', 'TXN1008'),

('user9', 1009, 580.00, '2025-04-30 08:41:00', 'Success', 'TXN1009'),

('user10', 1010, 700.00, '2025-04-30 08:46:00', 'Success', 'TXN1010'),

-- Continue similarly up to pnr_no 1050

('user11', 1011, 500.00, '2025-04-30 08:51:00', 'Success', 'TXN1011'),

('user12', 1012, 600.00, '2025-04-30 08:56:00', 'Success', 'TXN1012'),

('user13', 1013, 550.00, '2025-04-30 09:01:00', 'Success', 'TXN1013'),

('user14', 1014, 0.00, '2025-04-30 09:06:00', 'Refunded', 'TXN1014'),

('user15', 1015, 700.00, '2025-04-30 09:11:00', 'Success', 'TXN1015'),

('user16', 1016, 500.00, '2025-04-30 09:16:00', 'Success', 'TXN1016'),

('user17', 1017, 600.00, '2025-04-30 09:21:00', 'Success', 'TXN1017'),

('user18', 1018, 550.00, '2025-04-30 09:26:00', 'Success', 'TXN1018'),

('user19', 1019, 580.00, '2025-04-30 09:31:00', 'Success', 'TXN1019'),

('user20', 1020, 700.00, '2025-04-30 09:36:00', 'Success', 'TXN1020'),

('user21', 1021, 500.00, '2025-04-30 09:41:00', 'Success', 'TXN1021'),

('user22', 1022, 600.00, '2025-04-30 09:46:00', 'Success', 'TXN1022'),

('user23', 1023, 550.00, '2025-04-30 09:51:00', 'Success', 'TXN1023'),

('user24', 1024, 0.00, '2025-04-30 09:56:00', 'Refunded', 'TXN1024'),

('user25', 1025, 700.00, '2025-04-30 10:01:00', 'Success', 'TXN1025'),

('user26', 1026, 500.00, '2025-04-30 10:06:00', 'Success', 'TXN1026'),

('user27', 1027, 600.00, '2025-04-30 10:11:00', 'Success', 'TXN1027'),

('user28', 1028, 550.00, '2025-04-30 10:16:00', 'Success', 'TXN1028'),

('user29', 1029, 580.00, '2025-04-30 10:21:00', 'Success', 'TXN1029'),

('user30', 1030, 700.00, '2025-04-30 10:26:00', 'Success', 'TXN1030'),

('user31', 1031, 500.00, '2025-04-30 10:31:00', 'Success', 'TXN1031'),

('user32', 1032, 600.00, '2025-04-30 10:36:00', 'Success', 'TXN1032'),

('user33', 1033, 550.00, '2025-04-30 10:41:00', 'Success', 'TXN1033'),

('user34', 1034, 0.00, '2025-04-30 10:46:00', 'Refunded', 'TXN1034'),

('user35', 1035, 700.00, '2025-04-30 10:51:00', 'Success', 'TXN1035'),

('user36', 1036, 500.00, '2025-04-30 10:56:00', 'Success', 'TXN1036'),

('user37', 1037, 600.00, '2025-04-30 11:01:00', 'Success', 'TXN1037'),

('user38', 1038, 550.00, '2025-04-30 11:06:00', 'Success', 'TXN1038'),

('user39', 1039, 580.00, '2025-04-30 11:11:00', 'Success', 'TXN1039'),

('user40', 1040, 700.00, '2025-04-30 11:16:00', 'Success', 'TXN1040'),

('user41', 1041, 500.00, '2025-04-30 11:21:00', 'Success', 'TXN1041'),

('user42', 1042, 600.00, '2025-04-30 11:26:00', 'Success', 'TXN1042'),

('user43', 1043, 550.00, '2025-04-30 11:31:00', 'Success', 'TXN1043'),

('user44', 1044, 0.00, '2025-04-30 11:36:00', 'Refunded', 'TXN1044'),

('user45', 1045, 700.00, '2025-04-30 11:41:00', 'Success', 'TXN1045'),

('user46', 1046, 500.00, '2025-04-30 11:46:00', 'Success', 'TXN1046'),

('user47', 1047, 600.00, '2025-04-30 11:51:00', 'Success', 'TXN1047'),

('user48', 1048, 550.00, '2025-04-30 11:56:00', 'Success', 'TXN1048'),

('user49', 1049, 580.00, '2025-04-30 12:01:00', 'Success', 'TXN1049'),

('user50', 1050, 700.00, '2025-04-30 12:06:00', 'Success', 'TXN1050');

-- ==

-- 11. PASSENGER TABLE

-- ==

INSERT INTO passenger (t_number, t_date, name, age, gender, pnr_no, berth_no, berth_type, coach_no, concession_type, status) VALUES

(10101, '2025-05-01', 'Alice A', 30, 'F', 1001, 12, 'LB', 'A1', 'None', 'Confirmed'),

(10101, '2025-05-01', 'Bob B', 45, 'M', 1001, 13, 'UB', 'A1', 'Senior Citizen', 'Confirmed'),

(10202, '2025-05-01', 'Charlie C', 25, 'M', 1002, 7, 'LB', 'A2', 'Student', 'Confirmed'),

(10303, '2025-05-01', 'Diana D', 35, 'F', 1003, 22, 'MB', 'B1', 'None', 'Confirmed'),

(10404, '2025-05-01', 'Evan E', 50, 'M', 1004, 5, 'LB', 'B2', 'Military', 'Cancelled'),

(10505, '2025-05-01', 'Fiona F', 28, 'F', 1005, 8, 'UB', 'A1', 'None', 'Confirmed'),

(10101, '2025-05-01', 'George G', 40, 'M', 1006, 10, 'LB', 'A1', 'None', 'Confirmed'),

(10202, '2025-05-01', 'Hannah H', 32, 'F', 1007, 15, 'UB', 'A2', 'Student', 'Confirmed'),

(10303, '2025-05-01', 'Ian I', 29, 'M', 1008, 20, 'MB', 'B1', 'None', 'Confirmed'),

(10404, '2025-05-01', 'Jane J', 38, 'F', 1009, 11, 'LB', 'B2', 'Senior Citizen', 'Confirmed'),

(10505, '2025-05-01', 'Kevin K', 33, 'M', 1010, 9, 'UB', 'A1', 'None', 'Confirmed'),

(10101, '2025-05-01', 'Laura L', 27, 'F', 1011, 14, 'MB', 'A2', 'None', 'Confirmed'),

(10202, '2025-05-01', 'Mike M', 42, 'M', 1012, 6, 'LB', 'B1', 'Military', 'Confirmed'),

(10303, '2025-05-01', 'Nina N', 31, 'F', 1013, 18, 'UB', 'B2', 'Student', 'Confirmed'),

(10404, '2025-05-01', 'Oscar O', 36, 'M', 1014, 16, 'MB', 'A1', 'None', 'Cancelled'),

(10505, '2025-05-01', 'Paula P', 26, 'F', 1015, 12, 'LB', 'A2', 'None', 'Confirmed'),

(10101, '2025-05-01', 'Quinn Q', 39, 'M', 1016, 7, 'LB', 'B1', 'None', 'Confirmed'),

(10202, '2025-05-01', 'Rose R', 34, 'F', 1017, 13, 'UB', 'B2', 'None', 'Confirmed'),

(10303, '2025-05-01', 'Sam S', 30, 'M', 1018, 10, 'LB', 'A1', 'None', 'Confirmed'),

(10404, '2025-05-01', 'Tina T', 29, 'F', 1019, 15, 'UB', 'A2', 'Student', 'Confirmed');

-- (Add more rows following similar pattern as needed to reach 60+ passengers.)

-- ==

-- 12. REFUND_LOG TABLE

-- ==

INSERT INTO refund_log (pnr_no, refund_time, refund_amount, refund_status) VALUES

```
(1004, '2025-05-01 12:00:00', 580.00, 'Calculated'),
(1014, '2025-05-01 12:05:00', 580.00, 'Calculated'),
(1024, '2025-05-01 12:10:00', 580.00, 'Calculated'),
(1034, '2025-05-01 12:15:00', 580.00, 'Calculated'),
(1044, '2025-05-01 12:20:00', 580.00, 'Calculated');
-- ========================================
-- 13. TRAIN_RUNNING_DAYS TABLE
-- ========================================
-- Assume each train runs every day; generate 7 rows per train (10 trains => 70 rows)
INSERT INTO train_running_days (t_number, weekday) VALUES
(10101, 'Monday'),
(10101, 'Tuesday'),
(10101, 'Wednesday'),
(10101, 'Thursday'),
(10101, 'Friday'),
(10101, 'Saturday'),
(10101, 'Sunday'),

(10202, 'Monday'),
(10202, 'Tuesday'),
(10202, 'Wednesday'),
(10202, 'Thursday'),
(10202, 'Friday'),
(10202, 'Saturday'),
(10202, 'Sunday'),

(10303, 'Monday'),
(10303, 'Tuesday'),
(10303, 'Wednesday'),
(10303, 'Thursday'),
(10303, 'Friday'),
```

(10303, 'Saturday'),

(10303, 'Sunday'),

(10404, 'Monday'),

(10404, 'Tuesday'),

(10404, 'Wednesday'),

(10404, 'Thursday'),

(10404, 'Friday'),

(10404, 'Saturday'),

(10404, 'Sunday'),

(10505, 'Monday'),

(10505, 'Tuesday'),

(10505, 'Wednesday'),

(10505, 'Thursday'),

(10505, 'Friday'),

(10505, 'Saturday'),

(10505, 'Sunday'),

(10606, 'Monday'),

(10606, 'Tuesday'),

(10606, 'Wednesday'),

(10606, 'Thursday'),

(10606, 'Friday'),

(10606, 'Saturday'),

(10606, 'Sunday'),

(10707, 'Monday'),

(10707, 'Tuesday'),

(10707, 'Wednesday'),

(10707, 'Thursday'),

(10707, 'Friday'),

(10707, 'Saturday'),

(10707, 'Sunday'),

(10808, 'Monday'),

(10808, 'Tuesday'),

(10808, 'Wednesday'),

(10808, 'Thursday'),

(10808, 'Friday'),

(10808, 'Saturday'),

(10808, 'Sunday'),

(10909, 'Monday'),

(10909, 'Tuesday'),

(10909, 'Wednesday'),

(10909, 'Thursday'),

(10909, 'Friday'),

(10909, 'Saturday'),

(10909, 'Sunday'),

(11010, 'Monday'),

(11010, 'Tuesday'),

(11010, 'Wednesday'),

(11010, 'Thursday'),

(11010, 'Friday'),

(11010, 'Saturday'),

(11010, 'Sunday');

-- ==========

-- 14. WAITLIST TABLE

```
-- ==========================================
-- Insert around 20 waitlist entries.
INSERT INTO waitlist (pnr_no, position, status, waitlist_type) VALUES
(1001, 1, 'Waiting', 'General'),
(1002, 2, 'Waiting', 'General'),
(1003, 3, 'Waiting', 'General'),
(1004, 4, 'Waiting', 'General'),
(1005, 5, 'Waiting', 'General'),
(1006, 6, 'Waiting', 'General'),
(1007, 7, 'Waiting', 'General'),
(1008, 8, 'Waiting', 'General'),
(1009, 9, 'Waiting', 'General'),
(1010, 10, 'Waiting', 'General'),
(1011, 11, 'Waiting', 'General'),
(1012, 12, 'Waiting', 'General'),
(1013, 13, 'Waiting', 'General'),
(1014, 14, 'Waiting', 'General'),
(1015, 15, 'Waiting', 'General'),
(1016, 16, 'Waiting', 'General'),
(1017, 17, 'Waiting', 'General'),
(1018, 18, 'Waiting', 'General'),
(1019, 19, 'Waiting', 'General'),
(1020, 20, 'Waiting', 'General');



-- ==========================================
-- 15. NOTIFICATION TABLE
-- ==========================================
-- Insert around 30 notifications
INSERT INTO notification (username, message, created_at) VALUES
('user1', 'Your ticket has been booked successfully.', '2025-04-30 08:01:00'),
```

('user2', 'Your payment has been received.', '2025-04-30 08:06:00'),

('user3', 'Your seat has been allocated.', '2025-04-30 08:11:00'),

('user4', 'Your ticket has been cancelled.', '2025-04-30 08:16:00'),

('user5', 'Refund process initiated.', '2025-04-30 08:21:00'),

('user6', 'Your ticket has been booked successfully.', '2025-04-30 08:26:00'),

('user7', 'Your payment has been received.', '2025-04-30 08:31:00'),

('user8', 'Your seat has been allocated.', '2025-04-30 08:36:00'),

('user9', 'Your ticket has been cancelled.', '2025-04-30 08:41:00'),

('user10', 'Refund process initiated.', '2025-04-30 08:46:00'),

('user11', 'Your ticket has been booked successfully.', '2025-04-30 08:51:00'),

('user12', 'Your payment has been received.', '2025-04-30 08:56:00'),

('user13', 'Your seat has been allocated.', '2025-04-30 09:01:00'),

('user14', 'Your ticket has been cancelled.', '2025-04-30 09:06:00'),

('user15', 'Refund process initiated.', '2025-04-30 09:11:00'),

('user16', 'Your ticket has been booked successfully.', '2025-04-30 09:16:00'),

('user17', 'Your payment has been received.', '2025-04-30 09:21:00'),

('user18', 'Your seat has been allocated.', '2025-04-30 09:26:00'),

('user19', 'Your ticket has been cancelled.', '2025-04-30 09:31:00'),

('user20', 'Refund process initiated.', '2025-04-30 09:36:00'),

('user21', 'Your ticket has been booked successfully.', '2025-04-30 09:41:00'),

('user22', 'Your payment has been received.', '2025-04-30 09:46:00'),

('user23', 'Your seat has been allocated.', '2025-04-30 09:51:00'),

('user24', 'Your ticket has been cancelled.', '2025-04-30 09:56:00'),

('user25', 'Refund process initiated.', '2025-04-30 10:01:00'),

('user26', 'Your ticket has been booked successfully.', '2025-04-30 10:06:00'),

('user27', 'Your payment has been received.', '2025-04-30 10:11:00'),

('user28', 'Your seat has been allocated.', '2025-04-30 10:16:00'),

('user29', 'Your ticket has been cancelled.', '2025-04-30 10:21:00'),

('user30', 'Refund process initiated.', '2025-04-30 10:26:00');

```
-- ========================================

-- 16. SEAT_ALLOCATION TABLE

-- ========================================

-- Insert around 50 seat allocations (again, pnr_no should match one in ticket table)

INSERT INTO seat_allocation (pnr_no, t_number, t_date, coach, berth_no, berth_type,
class_type,allocated_at) VALUES

(1001, 10101, '2025-05-01', 'A1', 12, 'LB', 'AC', '2025-04-30 08:01:30'),

(1002, 10202, '2025-05-01', 'A2', 7, 'LB', 'sleeper', '2025-04-30 08:06:30'),

(1003, 10303, '2025-05-01', 'B1', 22, 'MB', 'AC', '2025-04-30 08:11:30'),

(1004, 10404, '2025-05-01', 'B2', 5, 'LB', 'sleeper', '2025-04-30 08:16:30'),

(1005, 10505, '2025-05-01', 'A1', 8, 'UB', 'AC', '2025-04-30 08:21:30'),

(1006, 10101, '2025-05-01', 'A1', 10, 'LB', 'sleeper', '2025-04-30 08:26:30'),

(1007, 10202, '2025-05-01', 'A2', 15, 'UB', 'AC', '2025-04-30 08:31:30'),

(1008, 10303, '2025-05-01', 'B1', 20, 'MB', 'sleeper', '2025-04-30 08:36:30'),

(1009, 10404, '2025-05-01', 'B2', 11, 'LB', 'AC', '2025-04-30 08:41:30'),

(1010, 10505, '2025-05-01', 'A1', 9, 'UB', 'sleeper', '2025-04-30 08:46:30'),

(1011, 10101, '2025-05-01', 'A2', 14, 'MB', 'AC', '2025-04-30 08:51:30'),

(1012, 10202, '2025-05-01', 'B1', 6,  'LB', 'sleeper', '2025-04-30 08:56:30'),

(1013, 10303, '2025-05-01', 'B2', 18, 'UB', 'AC', '2025-04-30 09:01:30'),

(1014, 10404, '2025-05-01', 'A1', 16, 'MB', 'sleeper', '2025-04-30 09:06:30'),

(1015, 10505, '2025-05-01', 'A2', 12, 'LB', 'AC', '2025-04-30 09:11:30'),

(1016, 10101, '2025-05-01', 'B1', 7,  'LB', 'sleeper', '2025-04-30 09:16:30'),

(1017, 10202, '2025-05-01', 'B2', 13, 'UB', 'AC', '2025-04-30 09:21:30'),

(1018, 10303, '2025-05-01', 'A1', 10, 'LB', 'sleeper', '2025-04-30 09:26:30'),

(1019, 10404, '2025-05-01', 'A2', 15, 'UB', 'AC', '2025-04-30 09:31:30'),

(1020, 10505, '2025-05-01', 'B1', 9,  'UB', 'sleeper', '2025-04-30 09:36:30');

-- (Continue to add more rows as needed to approach 50 entries.)
```

```
-- 1. Check email availability
CALL check_email_registered('user50@example.com');
CALL check_email_registered('newuser@example.com');

-- 2. Check username availability
CALL check_username_registered('user25');
CALL check_username_registered('newuser123');

-- 3. Validate admin credentials
CALL check_admin_credentials('admin1', 'pass1'); -- Valid
CALL check_admin_credentials('admin1', 'wrongpass'); -- Invalid

-- 4. Validate user credentials
CALL check_user_credentials('user35', 'pwd35'); -- Valid
CALL check_user_credentials('user35', 'wrongpwd'); -- Invalid

-- 5. Check train validity
CALL check_train_details(10101, '2025-05-01'); -- Valid date
CALL check_train_details(99999, '2025-05-01'); -- Invalid train

-- 6. Check seat availability
CALL check_seats_availability(10101, '2025-05-01', 'AC', 3); -- Check availability
CALL check_seats_availability(10101, '2025-05-01', 'Sleeper', 10);
```

```
-- 7. Generate PNR
CALL generate_pnr('user10', 10202, '2025-05-01', @new_pnr);
SELECT @new_pnr;

-- 8. Assign berth (assuming @new_pnr from above)

set @new_pnr = 1050;
CALL assign_berth(@new_pnr, 10202, '2025-05-01', 'AC', 'A3', 'Emma Watson', 28, 'F');



-- 9. Cancel ticket
CALL cancel_ticket(1004); -- Cancels ticket and processes refund

-- 10. Calculate ticket fare
SET @fare = 0;
CALL calculate_ticket_fare(10101, 'DEL', 'MUM', 'AC', @fare);
SELECT @fare AS calculated_fare;

-- 11. Apply concession
SET @base = 500.00;
CALL apply_concession(@base, 'Senior Citizen', @final);
SELECT @final AS discounted_fare;

-- 12. Check valid PNR
CALL check_valid_pnr(1001); -- Valid
CALL check_valid_pnr(9999); -- Invalid

-- 13. List passengers on a train
CALL list_passengers(10101, '2025-05-01');
```

```
-- 14. Check busiest route

CALL busiest_route();


-- 15. Generate itemized bill

CALL generate_itemized_bill(1013);

delete from fare where distance <> 100 and class_type = 'AC' or class_type = 'sleeper';

insert into fare values ('sleeper',100,120.00);

call calculate_ticket_fare(10202,'CHN','MUM','AC',@total_fare);

select @total_fare;

-- 16. Test trigger: Prevent overbooking

-- This should throw an error

UPDATE train_status SET seats_b_ac = 3 WHERE t_number = 10101 AND t_date = '2025-05-
01';


-- 17. Test trigger: Prevent duplicate berth assignment

-- This should throw an error

INSERT INTO passenger (t_number, t_date, name, age, gender, pnr_no, berth_no,
berth_type, coach_no)

VALUES (10101, '2025-05-01', 'Test Passenger', 30, 'M', 1001, 25, 'LB', 'A1');


-- 18. Test refund trigger by updating ticket status

UPDATE ticket SET status = 'Cancelled' WHERE pnr_no = 1015;


-- Verify refund log and seat availability

SELECT * FROM refund_log WHERE pnr_no = 1015;

call cancel_ticket('1013');


SELECT * FROM train_status WHERE t_number = 10101 AND t_date = '2025-05-01';


-- 19. Check total revenue

CALL total_revenue('2025-04-30', '2025-05-01');
```

-- 20. View train schedules

SELECT * FROM train_schedules WHERE t_number = 10101;