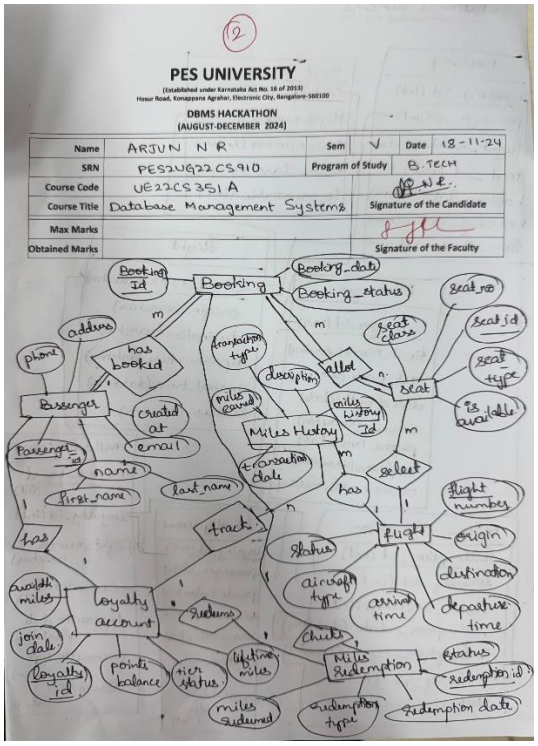
DBMS Hackathon

Set 2: Airline Reservation and Management System Version-2
Batch – 16

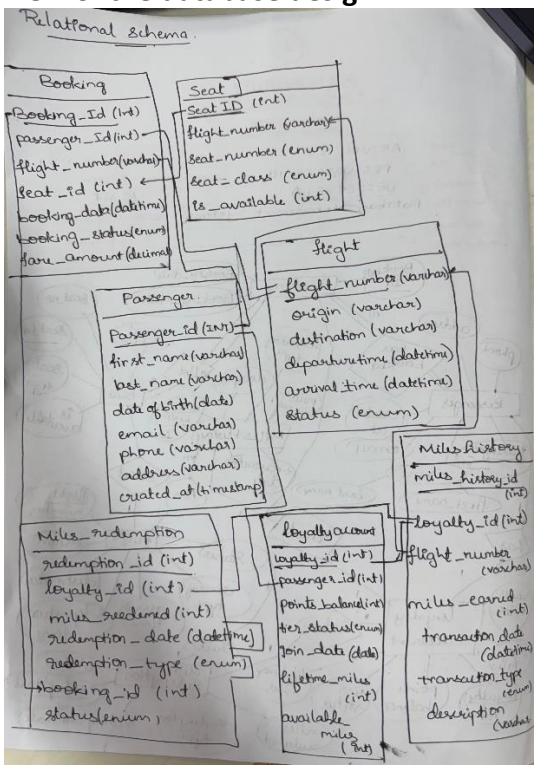
Team Members:

Arjun NR	PES2UG22CS910
Mohammed Hassan	PES2UG22CS317
Moulik Machaiah	PES2UG22CS322
Nandakishore Pramod	PES2UG22CS334

Q1) Draw an E-R Diagram for the Airline Reservation and Management System, illustrating components given in the above case study and infer the relationships and constraints within the system.



Q2) Convert the E-R Diagram into a relational schema, detailing constraints to provide a clear
view of the database design.



Q3) Develop the database with all relevant tables, establishing proper relationships and constraints. Populate each table with at least 5 records. Attach screenshots of SQL queries for table creation and the populated data.

Table Flights :

```
CREATE TABLE Flights (
        flight_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        flight_number VARCHAR(10) NOT NULL,
        departure_time DATETIME NOT NULL,
        arrival_time DATETIME NOT NULL,
        source VARCHAR(50) NOT NULL,
        destination VARCHAR(50) NOT NULL,
        class VARCHAR(20) CHECK (class IN ('Economy', 'Business', 'First')) NOT NULL, CONSTRAINT
UQ_FlightNumber UNIQUE (flight_number) );
```

```
317_322_334_910>select * from flights;
+-----------+---------------+---------------------+---------------------+---------------+---------------+----------+
| flight_id | flight_number | departure_time      | arrival_time        | source        | destination   | class    |
+-----------+---------------+---------------------+---------------------+---------------+---------------+----------+
|         1 | FL001         | 2024-11-19 08:00:00 | 2024-11-19 10:00:00 | New York      | Boston        | Economy  |
|         2 | FL002         | 2024-11-19 09:00:00 | 2024-11-19 12:00:00 | Chicago       | Dallas        | Business |
|         3 | FL003         | 2024-11-19 14:00:00 | 2024-11-19 17:00:00 | San Francisco | Seattle       | First    |
|         4 | FL004         | 2024-11-19 06:00:00 | 2024-11-19 08:30:00 | Atlanta       | Miami         | Economy  |
|         5 | FL005         | 2024-11-19 15:30:00 | 2024-11-19 18:00:00 | Los Angeles   | Denver        | Business |
|         6 | FL006         | 2024-11-20 10:00:00 | 2024-11-20 12:00:00 | Houston       | Las Vegas     | First    |
|         7 | FL007         | 2024-11-20 07:30:00 | 2024-11-20 10:00:00 | New York      | Chicago       | Economy  |
|         8 | FL008         | 2024-11-20 13:00:00 | 2024-11-20 15:30:00 | San Diego     | Phoenix       | Business |
|         9 | FL009         | 2024-11-20 08:00:00 | 2024-11-20 10:00:00 | Dallas        | Atlanta       | First    |
|        10 | FL010         | 2024-11-21 17:00:00 | 2024-11-21 19:00:00 | Orlando       | New Orleans   | Economy  |
|        11 | FL011         | 2024-11-21 12:00:00 | 2024-11-21 14:30:00 | Seattle       | San Francisco | Business |
|        12 | FL012         | 2024-11-21 20:00:00 | 2024-11-21 22:00:00 | Boston        | New York      | First    |
|        13 | FL013         | 2024-11-22 06:30:00 | 2024-11-22 09:00:00 | Miami         | Los Angeles   | Economy  |
|        14 | FL014         | 2024-11-22 13:00:00 | 2024-11-22 15:30:00 | Denver        | Houston       | Business |
|        15 | FL015         | 2024-11-22 10:30:00 | 2024-11-22 12:00:00 | Las Vegas     | San Diego     | First    |
|        16 | FL016         | 2024-11-23 16:00:00 | 2024-11-23 18:30:00 | New Orleans   | Orlando       | Economy  |
|        17 | FL017         | 2024-11-23 19:30:00 | 2024-11-23 21:00:00 | Phoenix       | Dallas        | Business |
|        18 | FL018         | 2024-11-23 09:00:00 | 2024-11-23 11:30:00 | Atlanta       | New York      | First    |
|        19 | FL019         | 2024-11-24 07:00:00 | 2024-11-24 09:30:00 | San Francisco | Seattle       | Economy  |
|        20 | FL020         | 2024-11-24 13:00:00 | 2024-11-24 15:00:00 | Chicago       | Miami         | Business |
|        21 | FL021         | 2024-11-24 17:30:00 | 2024-11-24 19:30:00 | Dallas        | Atlanta       | First    |
|        22 | FL022         | 2024-11-25 08:30:00 | 2024-11-25 11:00:00 | Boston        | Los Angeles   | Economy  |
|        23 | FL023         | 2024-11-25 12:00:00 | 2024-11-25 14:00:00 | Seattle       | Denver        | Business |
|        24 | FL024         | 2024-11-25 15:30:00 | 2024-11-25 17:30:00 | Houston       | New York      | First    |
|        25 | FL025         | 2024-11-26 06:30:00 | 2024-11-26 09:30:00 | Miami         | Chicago       | Economy  |
|        26 | FL026         | 2024-11-26 10:00:00 | 2024-11-26 12:30:00 | Orlando       | San Diego     | Business |
|        27 | FL027         | 2024-11-26 14:30:00 | 2024-11-26 16:30:00 | San Francisco | Las Vegas     | First    |
|        28 | FL028         | 2024-11-27 09:00:00 | 2024-11-27 11:00:00 | Dallas        | New Orleans   | Economy  |
|        29 | FL029         | 2024-11-27 16:30:00 | 2024-11-27 18:30:00 | Denver        | Phoenix       | Business |
|        30 | FL030         | 2024-11-27 12:00:00 | 2024-11-27 14:30:00 | Chicago       | Seattle       | First    |
+-----------+---------------+---------------------+---------------------+---------------+---------------+----------+
30 rows in set (0.00 sec)
```

Passengers Table:

```
CREATE TABLE Passengers (
    passenger_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    seat_preference VARCHAR(10) CHECK (seat_preference IN ('Window', 'Aisle', 'Middle')) NOT
    NULL,
    class VARCHAR(20) CHECK (class IN ('Economy', 'Business', 'First')) NOT NULL );
```

```
317_322_334_910>select * from passengers;
+--------------+------------+-----------+-----------------+----------+
| passenger_id | first_name | last_name | seat_preference | class    |
+--------------+------------+-----------+-----------------+----------+
|            1 | John       | Doe       | Window          | First    |
|            2 | Jane       | Smith     | Aisle           | Business |
|            3 | Emily      | Johnson   | Middle          | First    |
|            4 | Michael    | Brown     | Window          | Economy  |
|            5 | Sarah      | Davis     | Aisle           | Business |
|            6 | David      | Wilson    | Middle          | Economy  |
|            7 | Sophia     | Garcia    | Window          | First    |
|            8 | James      | Martinez  | Aisle           | Business |
|            9 | Olivia     | Hernandez | Middle          | Economy  |
|           10 | Alexander  | Lopez     | Window          | First    |
|           11 | Liam       | Anderson  | Window          | Economy  |
|           12 | Emma       | Thomas    | Aisle           | Business |
|           13 | Noah       | Moore     | Middle          | Economy  |
|           14 | Ava        | Jackson   | Window          | First    |
|           15 | Isabella   | Martin    | Aisle           | Economy  |
|           16 | Ethan      | Lee       | Middle          | Business |
|           17 | Mia        | Perez     | Window          | First    |
|           18 | Lucas      | Taylor    | Aisle           | Economy  |
|           19 | Charlotte  | Harris    | Middle          | Economy  |
|           20 | Amelia     | White     | Window          | Business |
|           21 | Oliver     | Lewis     | Aisle           | First    |
|           22 | Harper     | Walker    | Middle          | Economy  |
|           23 | William    | Hall      | Window          | Economy  |
|           24 | Elijah     | Allen     | Aisle           | Business |
|           25 | Sophia     | Young     | Middle          | First    |
|           26 | Benjamin   | Hernandez | Window          | Business |
|           27 | Evelyn     | King      | Aisle           | Economy  |
|           28 | Lucas      | Wright    | Middle          | First    |
|           29 | Mason      | Lopez     | Window          | Economy  |
|           30 | Ella       | Hill      | Aisle           | Business |
|           31 | Hassan     | Mohammed  | Window          | Economy  |
|           32 | Passenger  | Test 1    | Aisle           | Economy  |
|           33 | test       | passenger | Aisle           | Economy  |
+--------------+------------+-----------+-----------------+----------+
33 rows in set (0.00 sec)
```

Seat allocation Table:

```
CREATE TABLE Seat_Allocation (
        allocation_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        flight_id INT NOT NULL,
        seat_number VARCHAR(5) NOT NULL,
        status VARCHAR(10) CHECK (status IN ('Available', 'Booked', 'Reserved')) NOT NULL,
        FOREIGN KEY (flight_id) REFERENCES Flights(flight_id),
        CONSTRAINT UQ_FlightSeat UNIQUE (flight_id, seat_number) );
```

```
317_322_334_910>select * from seat_allocation;
+---------------+-----------+-------------+-----------+
| allocation_id | flight_id | seat_number | status    |
+---------------+-----------+-------------+-----------+
|             1 |         1 | 1A          | Booked    |
|             2 |         1 | 1B          | Booked    |
|             3 |         1 | 1C          | Reserved  |
|             4 |         2 | 2A          | Booked    |
|             5 |         2 | 2B          | Booked    |
|             6 |         2 | 2C          | Reserved  |
|             7 |         3 | 3A          | Booked    |
|             8 |         3 | 3B          | Booked    |
|             9 |         3 | 3C          | Reserved  |
|            10 |         4 | 4A          | Available |
|            11 |         4 | 4B          | Booked    |
|            12 |         4 | 4C          | Reserved  |
|            13 |         5 | 5A          | Available |
|            14 |         5 | 5B          | Booked    |
|            15 |         5 | 5C          | Reserved  |
|            16 |         6 | 6A          | Available |
|            17 |         6 | 6B          | Booked    |
|            18 |         6 | 6C          | Reserved  |
|            19 |         7 | 7A          | Available |
|            20 |         7 | 7B          | Booked    |
|            21 |         7 | 7C          | Reserved  |
|            22 |         8 | 8A          | Available |
|            23 |         8 | 8B          | Booked    |
|            24 |         8 | 8C          | Reserved  |
|            25 |         9 | 9A          | Available |
|            26 |         9 | 9B          | Booked    |
|            27 |         9 | 9C          | Reserved  |
|            28 |        10 | 10A         | Available |
|            29 |        10 | 10B         | Booked    |
|            30 |        10 | 10C         | Reserved  |
+---------------+-----------+-------------+-----------+
30 rows in set (0.00 sec)
```

Loyalty Program:

```
CREATE TABLE Loyalty_Program (
        passenger_id INT PRIMARY KEY,
        points INT DEFAULT 0,
        FOREIGN KEY (passenger_id) REFERENCES Passengers(passenger_id) );
```

```
317_322_334_910>select * from loyalty_program;
+--------------+--------+
| passenger_id | points |
+--------------+--------+
|            1 |   1000 |
|            2 |    500 |
|            3 |   1500 |
|            4 |   2000 |
|            5 |    800 |
|            6 |    400 |
|            7 |   1800 |
|            8 |    600 |
|            9 |   1000 |
|           10 |    700 |
|           11 |    900 |
|           12 |   2500 |
|           13 |   3000 |
|           14 |   1200 |
|           15 |    500 |
|           16 |    200 |
|           17 |   1000 |
|           18 |    450 |
|           19 |    700 |
|           20 |    950 |
|           21 |   1100 |
|           22 |    350 |
|           23 |   4000 |
|           24 |    800 |
|           25 |   1500 |
|           26 |    750 |
|           27 |   1300 |
|           28 |   1800 |
|           29 |    900 |
|           30 |   2100 |
+--------------+--------+
30 rows in set (0.00 sec)
```

Booking Table:

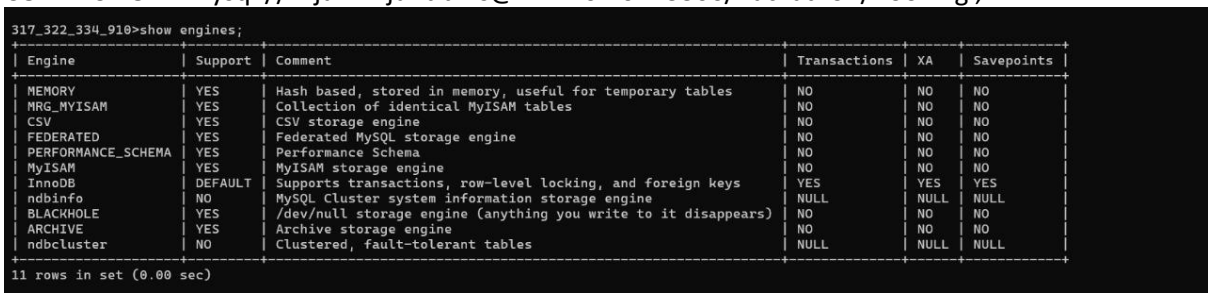
```
CREATE TABLE Booking (
   booking_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   flight_id INT NOT NULL,
   passenger_id INT NOT NULL,
   seat_number VARCHAR(5) NOT NULL,
   FOREIGN KEY (flight_id) REFERENCES Flights(flight_id),
   FOREIGN KEY (passenger_id) REFERENCES Passengers(passenger_id),
   FOREIGN KEY (flight_id, seat_number) REFERENCES Seat_Allocation(flight_id, seat_number)
);
```

```
317_322_334_910>select * from booking;
+------------+-----------+--------------+-------------+
| booking_id | flight_id | passenger_id | seat_number |
+------------+-----------+--------------+-------------+
|          1 |         1 |            1 | 1A          |
|          2 |         1 |            2 | 1B          |
|          3 |         2 |            3 | 2A          |
|          4 |         2 |            4 | 2B          |
|          5 |         3 |            5 | 3A          |
|          6 |         3 |            6 | 3B          |
|          7 |         4 |            7 | 4A          |
|          8 |         4 |            8 | 4B          |
|          9 |         5 |            9 | 5A          |
|         10 |         5 |           10 | 5B          |
|         11 |         6 |           11 | 6A          |
|         12 |         6 |           12 | 6B          |
|         13 |         7 |           13 | 7A          |
|         14 |         7 |           14 | 7B          |
|         15 |         8 |           15 | 8A          |
|         16 |         8 |           16 | 8B          |
|         17 |         9 |           17 | 9A          |
|         18 |         9 |           18 | 9B          |
|         19 |        10 |           19 | 10A         |
|         20 |        10 |           20 | 10B         |
|         21 |        11 |           21 | 11A         |
|         22 |        11 |           22 | 11B         |
|         23 |        12 |           23 | 12A         |
|         24 |        12 |           24 | 12B         |
|         25 |        13 |           25 | 13A         |
|         26 |        13 |           26 | 13B         |
|         27 |        14 |           27 | 14A         |
|         28 |        14 |           28 | 14B         |
|         29 |        15 |           29 | 15A         |
|         30 |        15 |           30 | 15B         |
|         31 |        16 |           31 | 16A         |
|         32 |        16 |           32 | 16B         |
|         33 |        17 |           33 | 17A         |
|         34 |        17 |           34 | 17B         |
|         35 |        18 |           35 | 18A         |
|         36 |        18 |           36 | 18B         |
```

Q4) SkyHigh Airlines operates with multiple branches in different locations. Each branch manages its seat allocation, and the central server maintains an authoritative record of all bookings. When a local branch cancels a booking (deletes a seat allocation), this deletion must reflect on the central server to ensure accurate and synchronized data. To ensure that seat allocation data is synchronized across the server and local branches, the Federated Engine is used. Demonstrate deletion of a record from the local branch and the reflection of the same on the central server using Federated Engine. Attach relevant screenshots from both machines.

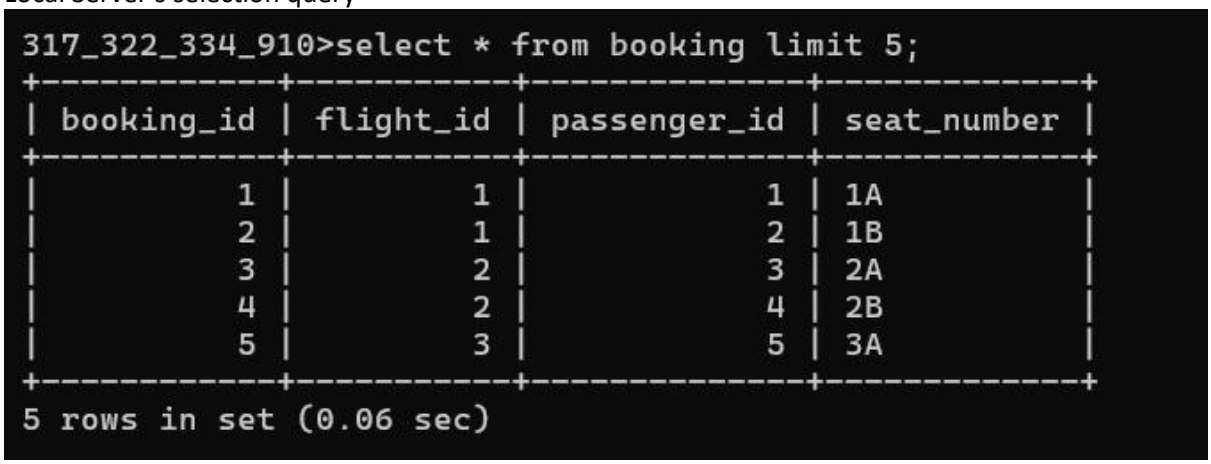
```
use hackathon;
drop table flights;
CREATE TABLE Flights (
   flight_id INT PRIMARY KEY AUTO_INCREMENT,
   flight_number VARCHAR(10) NOT NULL,
   departure_time DATETIME NOT NULL,
   arrival_time DATETIME NOT NULL,
   source VARCHAR(50) NOT NULL,
   destination VARCHAR(50) NOT NULL,
   class VARCHAR(20) CHECK (class IN ('Economy', 'Business', 'First')) NOT NULL
)
ENGINE=FEDERATED
DEFAULT CHARSET=utf8mb4
CONNECTION='mysql://test:test@172.20.10.2:3306/hackathon/flights';

drop table Passengers;
-- Create Passengers table
CREATE TABLE Passengers (
   passenger_id INT PRIMARY KEY AUTO_INCREMENT,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL,
   seat_preference VARCHAR(10) CHECK (seat_preference IN ('Window', 'Aisle', 'Middle')) NOT NULL,
   class VARCHAR(20) CHECK (class IN ('Economy', 'Business', 'First')) NOT NULL
)
ENGINE=FEDERATED
DEFAULT CHARSET=utf8mb4
CONNECTION='mysql://Arjun:Arjundbms@172.20.10.2:3306/hackathon/passengers';
drop table Seat_Allocation;
-- Create Seat_Allocation table
CREATE TABLE Seat_Allocation (
   allocation_id INT PRIMARY KEY AUTO_INCREMENT,
   flight_id INT NOT NULL,
   seat_number VARCHAR(5) NOT NULL,
   status VARCHAR(10) CHECK (status IN ('Available', 'Booked', 'Reserved')) NOT NULL,
   FOREIGN KEY (flight_id) REFERENCES Flights(flight_id)
)
ENGINE=FEDERATED
DEFAULT CHARSET=utf8mb4
CONNECTION='mysql://Arjun:Arjundbms@172.20.10.2:3306/hackathon/Seat_Allocation';

CREATE TABLE Loyalty_Program (
```

```
    passenger_id INT PRIMARY KEY,
    points INT DEFAULT 0,
    FOREIGN KEY (passenger_id) REFERENCES Passengers(passenger_id)
)
ENGINE=FEDERATED
DEFAULT CHARSET=utf8mb4
CONNECTION='mysql://Arjun:Arjundbms@172.20.10.2:3306/hackathon/Loyalty_Program';
drop table Booking;
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY auto_increment,
    flight_id INT NOT NULL,
    passenger_id INT NOT NULL,
    seat_number VARCHAR(5) NOT NULL,
    FOREIGN KEY (flight_id) REFERENCES Flights(flight_id),
    FOREIGN KEY (passenger_id) REFERENCES Passengers(passenger_id),
    FOREIGN KEY (flight_id, seat_number) REFERENCES Seat_Allocation(flight_id, seat_number)
)
ENGINE=FEDERATED
DEFAULT CHARSET=utf8mb4
CONNECTION='mysql://Arjun:Arjundbms@172.20.10.2:3306/hackathon/Booking';
```



```
317_322_334_910>show engines;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| FEDERATED          | YES     | Federated MySQL storage engine                                 | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| ndbinfo            | NO      | MySQL Cluster system information storage engine                | NULL         | NULL | NULL       |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| ndbcluster         | NO      | Clustered, fault-tolerant tables                               | NULL         | NULL | NULL       |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
11 rows in set (0.00 sec)
```

Local Server’s selection query



```
317_322_334_910>select * from booking limit 5;
+------------+-----------+--------------+-------------+
| booking_id | flight_id | passenger_id | seat_number |
+------------+-----------+--------------+-------------+
|          1 |         1 |            1 | 1A          |
|          2 |         1 |            2 | 1B          |
|          3 |         2 |            3 | 2A          |
|          4 |         2 |            4 | 2B          |
|          5 |         3 |            5 | 3A          |
+------------+-----------+--------------+-------------+
5 rows in set (0.06 sec)
```

Comment: Local Server is able to access the resources from the remote server

Q5) SkyHigh Airlines has received feedback that passengers often need to update their seat preferences after booking a flight. To address this, the airline wants to implement a feature where passengers can change their seat preferences (e.g., from window seat to an aisle seat) at any time. Create a stored procedure to handle this update efficiently. The procedure should allow airline staff to specify a passenger's ID and their new seat

preference, updating the passenger's record accordingly. Attach screenshots of an example execution of this stored procedure in the submission file.

```
CREATE PROCEDURE update_seat_preference(
   IN p_passenger_id INT,
   IN p_new_preference VARCHAR(10)
)
BEGIN
   DECLARE passenger_exists INT;

   -- Check if passenger exists
   SELECT COUNT(*) INTO passenger_exists
   FROM Passengers
   WHERE passenger_id = p_passenger_id;

   -- Validate seat preference
   IF p_new_preference NOT IN ('Window', 'Aisle', 'Middle') THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Invalid seat preference. Must be Window, Aisle, or Middle';
   END IF;

   -- Update preference if passenger exists
   IF passenger_exists > 0 THEN
      UPDATE Passengers
      SET seat_preference = p_new_preference
      WHERE passenger_id = p_passenger_id;

      SELECT 'Seat preference updated successfully' AS message;
   ELSE
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Passenger not found';
   END IF;
END //
```

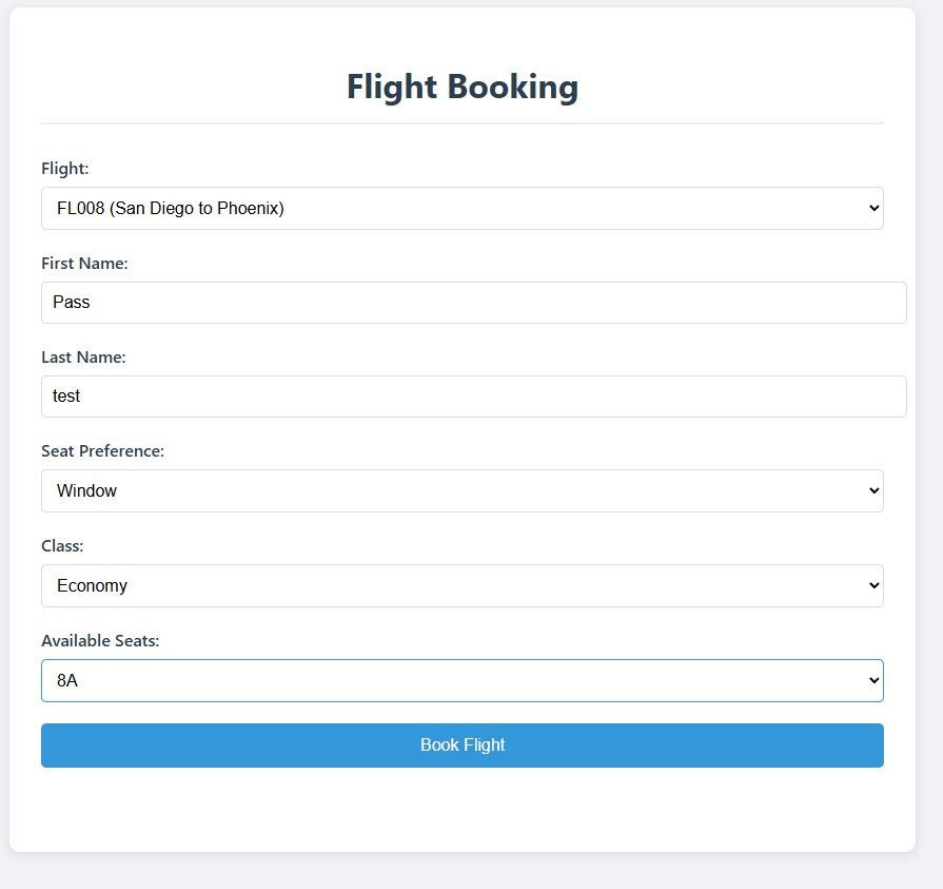
```
317_322_334_910>select * from passengers where passenger_id=1;
+--------------+------------+-----------+-----------------+---------+
| passenger_id | first_name | last_name | seat_preference | class   |
+--------------+------------+-----------+-----------------+---------+
|            1 | John       | Doe       | Window          | Economy |
+--------------+------------+-----------+-----------------+---------+
1 row in set (0.30 sec)
```

```
317_322_334_910>call update_seat_preference(1,'Aisle');
+-----------------------------------+
| message                           |
+-----------------------------------+
| Seat preference updated successfully |
+-----------------------------------+
1 row in set (0.34 sec)

Query OK, 0 rows affected (0.34 sec)
```

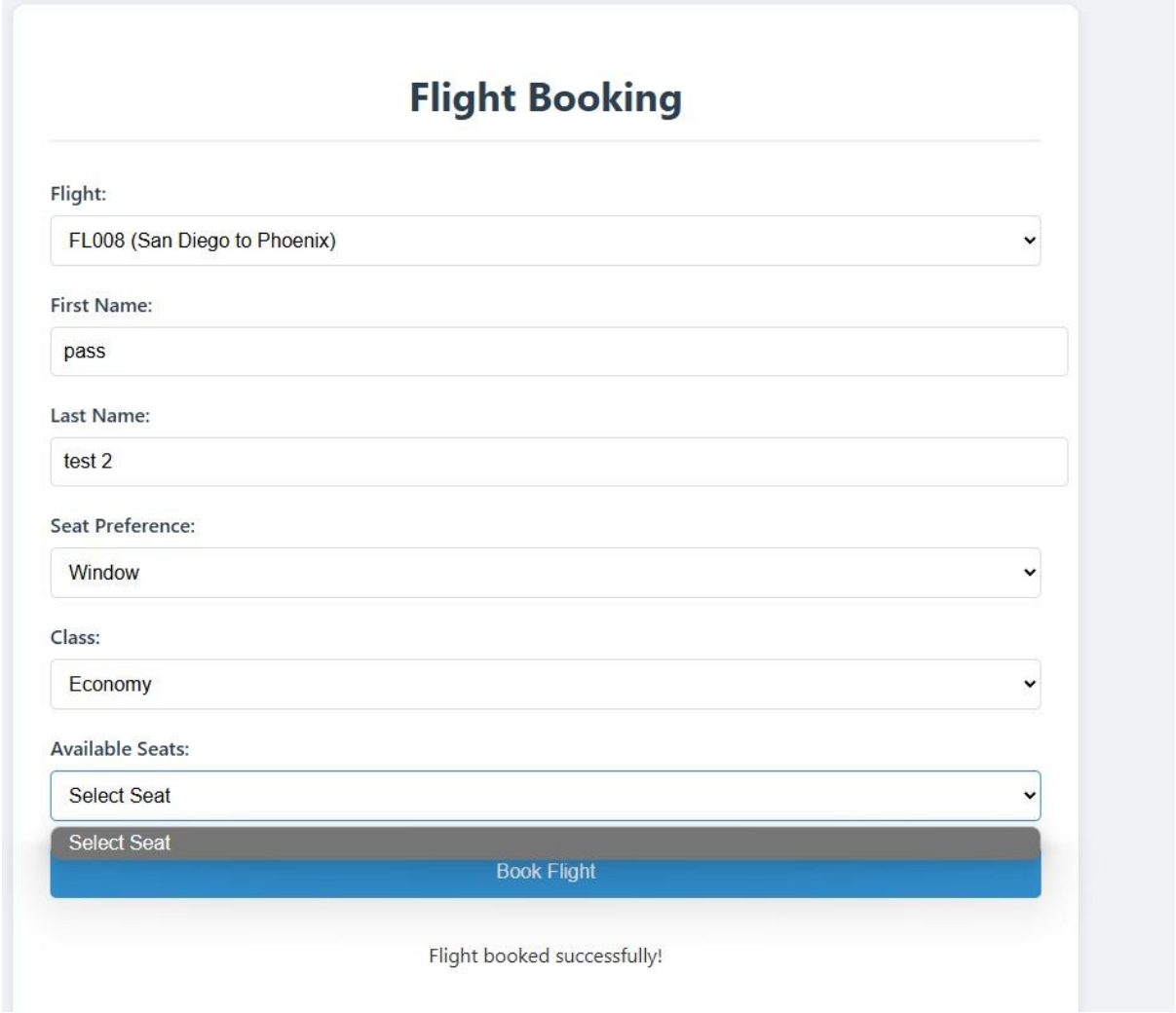
```
317_322_334_910>select * from passengers where passenger_id=1;
+--------------+------------+-----------+-----------------+---------+
| passenger_id | first_name | last_name | seat_preference | class   |
+--------------+------------+-----------+-----------------+---------+
|            1 | John       | Doe       | Aisle           | Economy |
+--------------+------------+-----------+-----------------+---------+
1 row in set (0.03 sec)
```

Q6) SkyHigh Airlines wants to automatically update the seat status to "Booked" when a new booking is made. This trigger should execute after a new row is inserted into the Booking table and update the corresponding seat status in the Seat_Allocation table. Create a front-end to demonstrate the trigger's functionality and attach screenshots showing its execution.

```
DELIMITER //

CREATE TRIGGER after_booking_insert
AFTER INSERT ON Booking
FOR EACH ROW
BEGIN
   UPDATE Seat_Allocation
   SET status = 'Booked'
   WHERE flight_id = NEW.flight_id
   AND seat_number = NEW.seat_number;
END//

DELIMITER ;
```



```
317_322_334_910>select * from seat_allocation where allocation_id=22;
+---------------+-----------+-------------+-----------+
| allocation_id | flight_id | seat_number | status    |
+---------------+-----------+-------------+-----------+
|            22 |         8 | 8A          | Available |
+---------------+-----------+-------------+-----------+
1 row in set (0.05 sec)
```




```
317_322_334_910>select * from seat_allocation where allocation_id=22;
+---------------+-----------+-------------+--------+
| allocation_id | flight_id | seat_number | status |
+---------------+-----------+-------------+--------+
|            22 |         8 | 8A          | Booked |
+---------------+-----------+-------------+--------+
1 row in set (0.03 sec)
```

Q7) SkyHigh Airlines wants to offer a discount for loyal passengers based on their accumulated
loyalty points. Implement a function that calculates a discount percentage on ticket prices based
on the total loyalty points accumulated by a passenger.
● Create a user-defined SQL function calculate_discount that takes:
1. loyalty_points (INT) – total points accumulated from the loyalty table.

2. base_ticket_price (DECIMAL) – ticket price for the flight.
3. Returns a discounted ticket price based on loyalty points.

Use the function to display for each passenger:
- **Passenger's full name**
- **Base ticket price (input)**
- **Discounted ticket price (using the function)**

```
CREATE FUNCTION calculate_discount(
   loyalty_points INT,
   base_ticket_price DECIMAL(10,2)
)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
   DECLARE discount_percentage DECIMAL(5,2);

   -- Set discount percentage based on loyalty points
   CASE
      WHEN loyalty_points <= 1000 THEN SET discount_percentage = 0;
      WHEN loyalty_points <= 5000 THEN SET discount_percentage = 5;
      WHEN loyalty_points <= 10000 THEN SET discount_percentage = 10;
      ELSE SET discount_percentage = 15;
   END CASE;

   -- Calculate and return discounted price
   RETURN base_ticket_price * (1 - discount_percentage/100);
END //

-- Create view to display passenger discount information
CREATE OR REPLACE VIEW passenger_discounts AS
SELECT
   CONCAT(p.first_name, ' ', p.last_name) as full_name,
   100.00 as base_ticket_price,
   calculate_discount(COALESCE(lp.points, 0), 100.00) as discounted_price,
   COALESCE(lp.points, 0) as loyalty_points
FROM
   Passengers p
   LEFT JOIN Loyalty_Program lp ON p.passenger_id = lp.passenger_id //
```

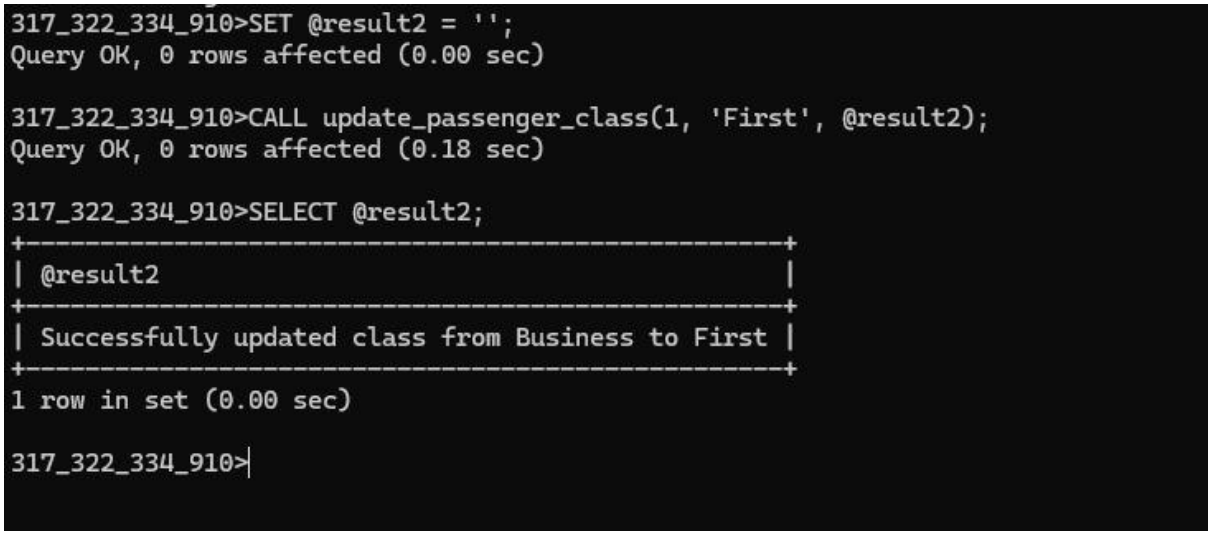
```
317_322_334_910>select * from passenger_discounts;
+--------------------+--------------------+------------------+----------------+
| full_name          | base_ticket_price  | discounted_price | loyalty_points |
+--------------------+--------------------+------------------+----------------+
| John Doe           |             100.00 |           100.00 |           1000 |
| Jane Smith         |             100.00 |           100.00 |            500 |
| Emily Johnson      |             100.00 |            95.00 |           1500 |
| Michael Brown      |             100.00 |            95.00 |           2000 |
| Sarah Davis        |             100.00 |           100.00 |            800 |
| David Wilson       |             100.00 |           100.00 |            400 |
| Sophia Garcia      |             100.00 |            95.00 |           1800 |
| James Martinez     |             100.00 |           100.00 |            600 |
| Olivia Hernandez   |             100.00 |           100.00 |           1000 |
| Alexander Lopez    |             100.00 |           100.00 |            700 |
| Liam Anderson      |             100.00 |           100.00 |            900 |
| Emma Thomas        |             100.00 |            95.00 |           2500 |
| Noah Moore         |             100.00 |            95.00 |           3000 |
| Ava Jackson        |             100.00 |            95.00 |           1200 |
| Isabella Martin    |             100.00 |           100.00 |            500 |
| Ethan Lee          |             100.00 |           100.00 |            200 |
| Mia Perez          |             100.00 |           100.00 |           1000 |
| Lucas Taylor       |             100.00 |           100.00 |            450 |
| Charlotte Harris   |             100.00 |           100.00 |            700 |
| Amelia White       |             100.00 |           100.00 |            950 |
| Oliver Lewis       |             100.00 |            95.00 |           1100 |
| Harper Walker      |             100.00 |           100.00 |            350 |
| William Hall       |             100.00 |            95.00 |           4000 |
| Elijah Allen       |             100.00 |           100.00 |            800 |
| Sophia Young       |             100.00 |            95.00 |           1500 |
| Benjamin Hernandez |             100.00 |           100.00 |            750 |
| Evelyn King        |             100.00 |            95.00 |           1300 |
| Lucas Wright       |             100.00 |            95.00 |           1800 |
| Mason Lopez        |             100.00 |           100.00 |            900 |
| Ella Hill          |             100.00 |            95.00 |           2100 |
| Hassan Mohammed    |             100.00 |           100.00 |              0 |
| Passenger  Test 1  |             100.00 |           100.00 |              0 |
| test passenger     |             100.00 |           100.00 |              0 |
+--------------------+--------------------+------------------+----------------+
33 rows in set (0.10 sec)
```

Q8) Demonstrate transaction isolation (Level-serializable) by implementing the following:
1. SkyHigh Airlines has an internal system where customer service representatives can update passengers' seat classes based on seat availability or loyalty program upgrades. Due to a system lag or simultaneous updates, two representatives attempt to change the class of a seat for the same passenger at the same time. This could lead to inconsistent data if not handled properly.
You are asked to demonstrate the impact of concurrent updates on the Passengers table and how transaction isolation can prevent issues like race conditions.

```
CREATE PROCEDURE update_passenger_class(
   IN p_passenger_id INT,
   IN p_new_class VARCHAR(20),
   OUT p_result VARCHAR(100)
)
BEGIN
   DECLARE current_class VARCHAR(20);
   DECLARE exit handler FOR SQLEXCEPTION
   BEGIN
      ROLLBACK;
```

```
    SET p_result = 'Error: Transaction rolled back';
  END;

  -- Start transaction with highest isolation level
  SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
  START TRANSACTION;

  -- Check if passenger exists and get current class
  SELECT class INTO current_class
  FROM Passengers
  WHERE passenger_id = p_passenger_id
  FOR UPDATE;  -- Lock the row

  IF current_class IS NULL THEN
    SET p_result = 'Error: Passenger not found';
    ROLLBACK;
  ELSE
    -- Validate new class
    IF p_new_class NOT IN ('Economy', 'Business', 'First') THEN
      SET p_result = 'Error: Invalid class specification';
      ROLLBACK;
    ELSE
      -- Perform the update
      UPDATE Passengers
      SET class = p_new_class
      WHERE passenger_id = p_passenger_id;

      SET p_result = CONCAT('Successfully updated class from ',
                  current_class, ' to ', p_new_class);
      COMMIT;
    END IF;
  END IF;
END //
```

```
prompt 317_322_334_910>SET @result1 = '';
Query OK, 0 rows affected (0.00 sec)

prompt 317_322_334_910>CALL update_passenger_class(1, 'Business', @result1);
Query OK, 0 rows affected (0.21 sec)

prompt 317_322_334_910>SELECT @result1;
+-----------------------------------------------------+
| @result1                                            |
+-----------------------------------------------------+
| Successfully updated class from Economy to Business |
+-----------------------------------------------------+
1 row in set (0.00 sec)

prompt 317_322_334_910>
```



Q9) Below are the queries to create and populate a simple employee database. Please copy
and paste these queries to set up the database, and then use them to answer the following
question based on the provided code.

```
-- Employee table
CREATE TABLE employees (
employee_id INT PRIMARY KEY,
employee_name VARCHAR(100)
);
-- Employee_hierarchy table
CREATE TABLE employee_hierarchy (
employee_id INT,
manager_id INT,
FOREIGN KEY (employee_id) REFERENCES employees(employee_id),
FOREIGN KEY (manager_id) REFERENCES employees(employee_id)
);
-- Insert employees
INSERT INTO employees (employee_id, employee_name) VALUES
(1, 'Alice'),
(2, 'Bob'),
(3, 'Charlie');

-- Insert employee hierarchy (relationships between employees)
INSERT INTO employee_hierarchy (employee_id, manager_id) VALUES
(2, 1), -- Bob reports to Alice
(3, 1); -- Charlie reports to Alice
```

Write a recursive query that can be used to find all employees who report to Alice directly or indirectly. Paste screenshots of the output in the submission report.

```
WITH RECURSIVE EmployeeHierarchy AS (
```

```
    -- Base case: Get Alice's direct reports
    SELECT
        e.employee_id AS EmployeeID,
        e.employee_name AS EmployeeName,
        eh.manager_id AS ManagerID
    FROM employees e
    JOIN employee_hierarchy eh ON e.employee_id = eh.employee_id
    WHERE eh.manager_id = 1

    UNION ALL

    -- Recursive case: Get employees reporting to the above set
    SELECT
        e.employee_id AS EmployeeID,
        e.employee_name AS EmployeeName,
        eh.manager_id AS ManagerID
    FROM employees e
    JOIN employee_hierarchy eh ON e.employee_id = eh.employee_id
    JOIN EmployeeHierarchy eh_recurse ON eh.manager_id = eh_recurse.EmployeeID
)
SELECT DISTINCT *
FROM EmployeeHierarchy;
```

```
prompt 317_322_334_910>WITH RECURSIVE EmployeeHierarchy AS (
    ->     -- Base case: Get Alice's direct reports
    ->     SELECT
    ->         e.employee_id AS EmployeeID,
    ->         e.employee_name AS EmployeeName,
    ->         eh.manager_id AS ManagerID
    ->     FROM employees e
    ->     JOIN employee_hierarchy eh ON e.employee_id = eh.employee_id
    ->     WHERE eh.manager_id = 1
    ->
    ->     UNION ALL
    ->
    ->     -- Recursive case: Get employees reporting to the above set
    ->     SELECT
    ->         e.employee_id AS EmployeeID,
    ->         e.employee_name AS EmployeeName,
    ->         eh.manager_id AS ManagerID
    ->     FROM employees e
    ->     JOIN employee_hierarchy eh ON e.employee_id = eh.employee_id
    ->     JOIN EmployeeHierarchy eh_recurse ON eh.manager_id = eh_recurse.EmployeeID
    -> )
    -> SELECT DISTINCT *
    -> FROM EmployeeHierarchy;
+------------+--------------+-----------+
| EmployeeID | EmployeeName | ManagerID |
+------------+--------------+-----------+
|          2 | Bob          |         1 |
|          3 | Charlie      |         1 |
+------------+--------------+-----------+
```