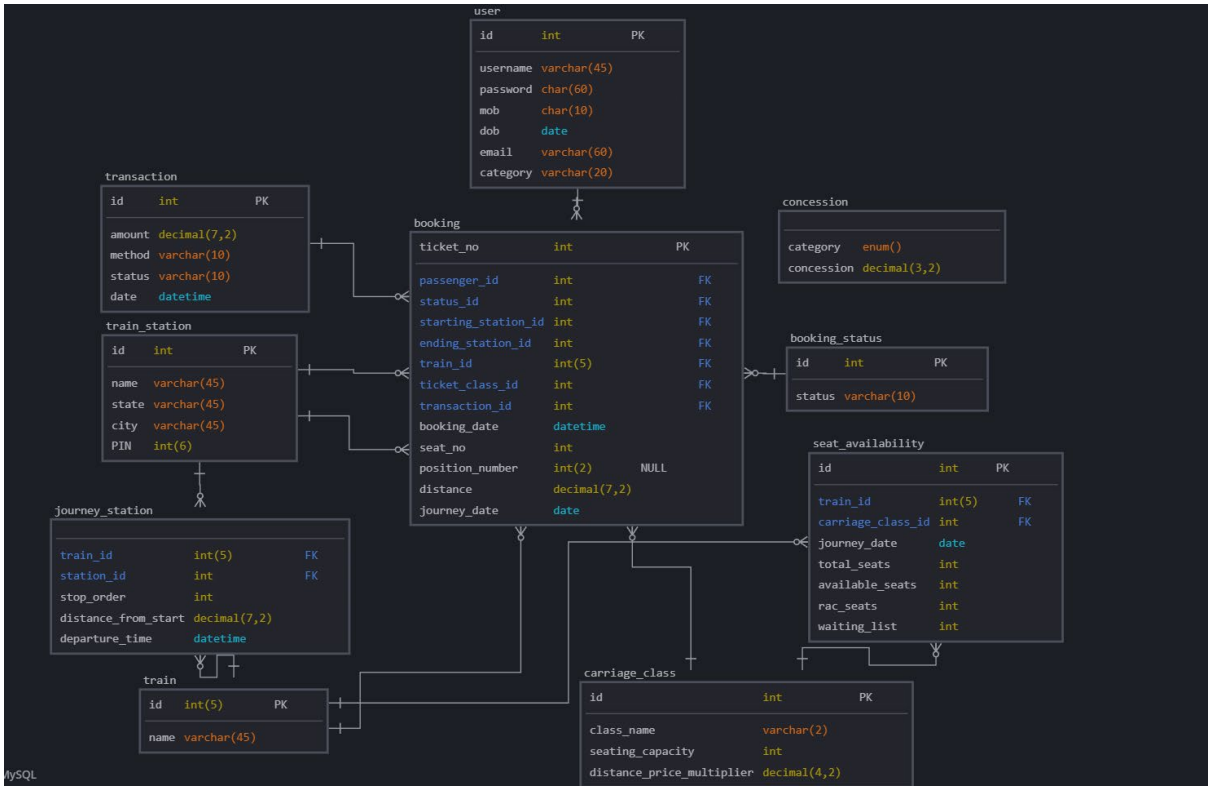
CS2202 Mini Project

ER Diagram:



Tables:

```
 1. CREATE TABLE booking_status(
 2.     id INT PRIMARY KEY AUTO_INCREMENT,
 3.     status VARCHAR(10) NOT NULL
 4. );
 5.
 6. CREATE TABLE user(
 7.     id INT PRIMARY KEY AUTO_INCREMENT,
 8.     username VARCHAR(45) NOT NULL UNIQUE,
 9.     password CHAR(60) NOT NULL,
10.     mob CHAR(10) NOT NULL,
11.     dob DATE NOT NULL,
12.     email VARCHAR(60) NOT NULL,
13.     category ENUM('STUDENT', 'SENIOR CITIZEN', 'DISABLED', 'GENERAL') NOT NULL
14. );
15.
16. CREATE TABLE concession(
17.     category ENUM('STUDENT', 'SENIOR CITIZEN', 'DISABLED', 'GENERAL') NOT NULL,
18.     concession DECIMAL(3,2)
19. );
20.
21. CREATE TABLE transaction(
22.     id INT PRIMARY KEY AUTO_INCREMENT,
23.     amount DECIMAL(7,2) NOT NULL,
24.     method varchar(10) NOT NULL,
25.     status varchar(10) NOT NULL,
26.     date DATETIME NOT NULL
27. );
28.
29. CREATE TABLE train_station(
30.     id INT PRIMARY KEY AUTO_INCREMENT,
31.     name VARCHAR(45) NOT NULL,
```

```
    state VARCHAR(45) NOT NULL,
    city VARCHAR(45) NOT NULL,
    PIN INT(6) NOT NULL
);

CREATE TABLE train(
    id INT(5) PRIMARY KEY NOT NULL,
    name VARCHAR(45) NOT NULL
);

CREATE TABLE journey_station(
    train_id INT NOT NULL,
    station_id INT NOT NULL,
    stop_order INT NOT NULL,
    distance_from_start DECIMAL(7,2) NOT NULL,
    departure_time DATETIME NOT NULL,
    CONSTRAINT station_fk FOREIGN KEY (station_id) REFERENCES train_station(id),
    CONSTRAINT train_fk FOREIGN KEY (train_id) REFERENCES train(id)
);

CREATE TABLE carriage_class(
    id INT PRIMARY KEY AUTO_INCREMENT,
    class_name CHAR(2) NOT NULL UNIQUE,
    seating_capacity INT NOT NULL,
    distance_price_multiplier DECIMAL(4,2) NOT NULL
);

CREATE TABLE booking(
    ticket_no INT AUTO_INCREMENT PRIMARY KEY,
    passenger_id INT NOT NULL,
    status_id INT NOT NULL,
    starting_station_id INT NOT NULL,
    ending_station_id INT NOT NULL,
    train_id INT(5) NOT NULL,
    ticket_class_id INT NOT NULL,
    transaction_id INT NOT NULL,
    booking_date DATETIME NOT NULL,
    seat_no INT NOT NULL,
    position_number INT(2),
    distance DECIMAL(7,2) NOT NULL,
    journey_date DATE,
    CONSTRAINT passenger_fk FOREIGN KEY (passenger_id) REFERENCES user(id),
    CONSTRAINT status_fk FOREIGN KEY (status_id) REFERENCES booking_status(id),
    CONSTRAINT starting_station_fk FOREIGN KEY (starting_station_id) REFERENCES
train_station(id),
    CONSTRAINT ending_station_fk FOREIGN KEY (ending_station_id) REFERENCES
train_station(id),
    CONSTRAINT train_id_booking_fk FOREIGN KEY (train_id) REFERENCES train(id),
    CONSTRAINT ticket_class_fk FOREIGN KEY (ticket_class_id) REFERENCES carriage_class(id),
    CONSTRAINT transaction_fk FOREIGN KEY (transaction_id) REFERENCES transaction(id)
);

CREATE TABLE seat_availability (
    id INT PRIMARY KEY AUTO_INCREMENT,
    train_id INT(5) NOT NULL,
    journey_date DATE NOT NULL,
    carriage_class_id INT NOT NULL,
    total_seats INT NOT NULL,
    available_seats INT NOT NULL,
    total_rac_seats INT NOT NULL,
    rac_seats INT NOT NULL,
    total_waiting_list INT NOT NULL,
    waiting_list INT NOT NULL,
    CONSTRAINT sa_train_fk FOREIGN KEY (train_id) REFERENCES train(id),
    CONSTRAINT sa_class_fk FOREIGN KEY (carriage_class_id) REFERENCES carriage_class(id),
    UNIQUE KEY (train_id, journey_date, carriage_class_id)
);
```

Procedures:

```
 1. DELIMITER //
 2. CREATE PROCEDURE book_ticket(
 3.     IN p_passenger_id INT,
 4.     IN p_train_id INT,
 5.     IN p_start_station INT,
 6.     IN p_end_station INT,
 7.     IN p_class_id INT,
 8.     IN p_journey_date DATE,
 9.     IN p_transaction_id INT,
10.     IN p_distance DECIMAL(7,2)
11. )
12. BEGIN
13.     DECLARE v_available_seats INT;
14.     DECLARE v_rac_seats INT;
15.     DECLARE v_waiting_list INT;
16.     DECLARE v_status_id INT;
17.     DECLARE v_position INT;
18.     DECLARE v_seat_no INT;
19.
20.     -- Get current availability
21.     SELECT available_seats, rac_seats, waiting_list
22.     INTO v_available_seats, v_rac_seats, v_waiting_list
23.     FROM seat_availability
24.     WHERE train_id = p_train_id
25.     AND journey_date = p_journey_date
26.     AND carriage_class_id = p_class_id;
27.
28.     -- Determine booking status
29.     IF v_available_seats > 0 THEN
30.         -- Confirmed ticket
31.         SELECT id INTO v_status_id FROM booking_status WHERE status = 'CONFIRMED';
32.         SET v_seat_no = (SELECT seating_capacity FROM carriage_class WHERE id = p_class_id)
- v_available_seats + 1;
33.         SET v_position = NULL;
34.
35.         -- Update available seats
36.         UPDATE seat_availability
37.         SET available_seats = available_seats - 1
38.         WHERE train_id = p_train_id
39.         AND journey_date = p_journey_date
40.         AND carriage_class_id = p_class_id;
41.
42.     ELSEIF v_rac_seats > 0 THEN
43.         -- RAC ticket
44.         SELECT id INTO v_status_id FROM booking_status WHERE status = 'RAC';
45.         SET v_seat_no = 0; -- RAC doesn't have fixed seat
46.         SET v_position = (SELECT COUNT(*) FROM booking
47.                           WHERE train_id = p_train_id
48.                           AND journey_date= p_journey_date
49.                           AND ticket_class_id = p_class_id
50.                           AND status_id = v_status_id) + 1;
51.
52.         -- Update RAC count
53.         UPDATE seat_availability
54.         SET rac_seats = rac_seats - 1
55.         WHERE train_id = p_train_id
56.         AND journey_date = p_journey_date
57.         AND carriage_class_id = p_class_id;
58.
59.     ELSE
60.         -- Waiting list ticket
61.         SELECT id INTO v_status_id FROM booking_status WHERE status = 'WL';
62.         SET v_seat_no = 0; -- WL doesn't have seat
63.         SET v_position = (SELECT COUNT(*) FROM booking
64.                           WHERE train_id = p_train_id
65.                           AND journey_date = p_journey_date
66.                           AND ticket_class_id = p_class_id
```

```
                           AND status_id = v_status_id) + 1;

        -- Update waiting list count
        UPDATE seat_availability
        SET waiting_list = waiting_list - 1
        WHERE train_id = p_train_id
        AND journey_date = p_journey_date
        AND carriage_class_id = p_class_id;
    END IF;

    -- Create booking
    INSERT INTO booking (
        passenger_id, status_id, starting_station_id,
        ending_station_id, train_id, ticket_class_id,
        transaction_id, booking_date, seat_no, position_number, distance, journey_date
    ) VALUES (
        p_passenger_id, v_status_id, p_start_station,
        p_end_station, p_train_id, p_class_id,
        p_transaction_id, NOW(), v_seat_no, v_position, p_distance, p_journey_date
    );

END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE cancel_ticket(IN p_booking_id INT)
BEGIN
    DECLARE v_train_id INT;
    DECLARE v_journey_date DATE;
    DECLARE v_class_id INT;
    DECLARE v_status_id INT;
    DECLARE v_cancelled_status_id INT;
    DECLARE v_seat_no INT;
    DECLARE v_position INT;
    DECLARE v_transaction_id INT;

    -- Get booking details
    SELECT train_id, journey_date, ticket_class_id, status_id, seat_no, position_number,
transaction_id
    INTO v_train_id, v_journey_date, v_class_id, v_status_id, v_seat_no, v_position,
v_transaction_id
    FROM booking WHERE ticket_no = p_booking_id;

    UPDATE transaction SET status="REFUND" WHERE id=v_transaction_id;

    SELECT id INTO v_cancelled_status_id FROM booking_status WHERE status = 'CANCELLED';

    -- Update booking status to cancelled
    UPDATE booking SET status_id = v_cancelled_status_id WHERE ticket_no = p_booking_id;

    -- Handle seat availability based on previous status
    IF (SELECT status FROM booking_status WHERE id = v_status_id) = 'CONFIRMED' THEN
        -- Free up a confirmed seat and upgrade RAC to confirmed if available
        UPDATE seat_availability
        SET available_seats = available_seats + 1
        WHERE train_id = v_train_id
        AND journey_date = v_journey_date
        AND carriage_class_id = v_class_id;

        UPDATE booking
        SET seat_no=seat_no-1
        WHERE train_id = v_train_id
        AND journey_date = v_journey_date
        AND ticket_class_id = v_class_id
        AND status_id = v_status_id
        AND seat_no>v_seat_no;

        -- Find first RAC ticket to upgrade
        CALL upgrade_rac_to_confirmed(v_train_id, v_journey_date, v_class_id);
        CALL upgrade_wl_to_rac(v_train_id, v_journey_date, v_class_id);
```

```
135.
136.     ELSEIF (SELECT status FROM booking_status WHERE id = v_status_id) = 'RAC' THEN
137.         -- Free up RAC and upgrade WL to RAC if available
138.         UPDATE seat_availability
139.         SET rac_seats = rac_seats + 1
140.         WHERE train_id = v_train_id
141.         AND journey_date = v_journey_date
142.         AND carriage_class_id = v_class_id;
143.
144.         UPDATE booking
145.         SET position_number = position_number -1
146.         WHERE train_id = v_train_id
147.         AND journey_date = v_journey_date
148.         AND status_id = v_status_id
149.         AND position_number > v_position;
150.
151.         -- Find first WL ticket to upgrade
152.         CALL upgrade_wl_to_rac(v_train_id, v_journey_date, v_class_id);
153.
154.     ELSEIF (SELECT status FROM booking_status WHERE id = v_status_id) = 'WL' THEN
155.         -- Free up waiting list
156.         UPDATE seat_availability
157.         SET waiting_list = waiting_list + 1
158.         WHERE train_id = v_train_id
159.         AND journey_date = v_journey_date
160.         AND carriage_class_id = v_class_id;
161.
162.         SELECT position_number INTO v_position FROM booking WHERE ticket_no = p_booking_id;
163.         -- Update positions of remaining WL tickets
164.         UPDATE booking
165.         SET position_number = position_number - 1
166.         WHERE train_id = v_train_id
167.         AND journey_date = v_journey_date
168.         AND ticket_class_id = v_class_id
169.         AND status_id = (SELECT id FROM booking_status WHERE status = 'WL')
170.         AND position_number > v_position;
171.     END IF;
172. END //
173. DELIMITER ;
174.
175. -- Procedure to upgrade RAC to confirmed
176. DELIMITER //
177. CREATE PROCEDURE upgrade_rac_to_confirmed(
178.     IN p_train_id INT,
179.     IN p_journey_date DATE,
180.     IN p_class_id INT
181. )
182. BEGIN
183.     DECLARE v_rac_status_id INT;
184.     DECLARE v_confirmed_status_id INT;
185.     DECLARE v_booking_id INT;
186.     DECLARE v_position INT;
187.     DECLARE v_seat_no INT;
188.
189.     SELECT id INTO v_rac_status_id FROM booking_status WHERE status = 'RAC';
190.     SELECT id INTO v_confirmed_status_id FROM booking_status WHERE status = 'CONFIRMED';
191.
192.     -- Find first RAC ticket
193.     SELECT ticket_no INTO v_booking_id
194.     FROM booking
195.     WHERE train_id = p_train_id
196.     AND journey_date = p_journey_date
197.     AND ticket_class_id = p_class_id
198.     AND status_id = v_rac_status_id
199.     ORDER BY position_number
200.     LIMIT 1;
201.
202.     IF v_booking_id IS NOT NULL THEN
203.
204.         SELECT (seating_capacity - available_seats + 1) INTO v_seat_no
```

```
205.         FROM carriage_class c
206.         JOIN seat_availability sa ON c.id = sa.carriage_class_id
207.         WHERE sa.train_id = p_train_id
208.         AND sa.journey_date = p_journey_date
209.         AND sa.carriage_class_id = p_class_id;
210.
211.         -- SELECT IFNULL(position_number, 0) INTO v_position FROM booking WHERE ticket_no =
v_booking_id;
212.
213.
214.         -- Update RAC positions
215.         UPDATE booking
216.         SET position_number = position_number - 1
217.         WHERE train_id = p_train_id
218.         AND journey_date = p_journey_date
219.         AND ticket_class_id = p_class_id
220.         AND status_id = v_rac_status_id;
221.         -- AND position_number > v_position;
222.
223.         -- Upgrade RAC to confirmed
224.         UPDATE booking
225.         SET status_id = v_confirmed_status_id,
226.             seat_no = v_seat_no,
227.             position_number = NULL
228.         WHERE ticket_no = v_booking_id;
229.
230.         -- Update seat availability
231.         UPDATE seat_availability
232.         SET rac_seats = rac_seats + 1,
233.             available_seats = available_seats - 1
234.         WHERE train_id = p_train_id
235.         AND journey_date = p_journey_date
236.         AND carriage_class_id = p_class_id;
237.
238.         -- Check if we need to upgrade a WL to RAC
239.         -- CALL upgrade_wl_to_rac(p_train_id, p_journey_date, p_class_id);
240.     END IF;
241. END //
242. DELIMITER ;
243.
244. -- Procedure to upgrade WL to RAC
245. DELIMITER //
246. CREATE PROCEDURE upgrade_wl_to_rac(
247.     IN p_train_id INT,
248.     IN p_journey_date DATE,
249.     IN p_class_id INT
250. )
251. BEGIN
252.     DECLARE v_wl_status_id INT;
253.     DECLARE v_rac_status_id INT;
254.     DECLARE v_booking_id INT;
255.     DECLARE v_position INT;
256.
257.     SELECT id INTO v_wl_status_id FROM booking_status WHERE status = 'WL';
258.     SELECT id INTO v_rac_status_id FROM booking_status WHERE status = 'RAC';
259.
260.     -- Find first WL ticket
261.     SELECT ticket_no INTO v_booking_id
262.     FROM booking
263.     WHERE train_id = p_train_id
264.     AND journey_date = p_journey_date
265.     AND ticket_class_id = p_class_id
266.     AND status_id = v_wl_status_id
267.     ORDER BY position_number
268.     LIMIT 1;
269.
270.     IF v_booking_id IS NOT NULL THEN
271.         -- Upgrade WL to RAC
272.         SELECT (MAX(position_number) + 1) INTO v_position
273.         FROM booking
```

```
274.         WHERE train_id = p_train_id
275.         AND journey_date = p_journey_date
276.         AND ticket_class_id = p_class_id
277.         AND status_id = v_rac_status_id;
278.
279.         UPDATE booking
280.         SET status_id = v_rac_status_id,
281.             position_number = v_position
282.         WHERE ticket_no = v_booking_id;
283.
284.         -- SELECT IFNULL(position_number, 0) INTO v_position FROM booking WHERE ticket_no =
v_booking_id;
285.
286.         -- Update WL positions
287.         UPDATE booking
288.         SET position_number = position_number - 1
289.         WHERE train_id = p_train_id
290.         AND journey_date = p_journey_date
291.         AND ticket_class_id = p_class_id
292.         AND status_id = v_wl_status_id;
293.
294.         -- Update seat availability
295.         UPDATE seat_availability
296.         SET waiting_list = waiting_list + 1,
297.             rac_seats = rac_seats - 1
298.         WHERE train_id = p_train_id
299.         AND journey_date = p_journey_date
300.         AND carriage_class_id = p_class_id;
301.     END IF;
302. END //
303. DELIMITER ;
304.
305. DELIMITER //
306. CREATE PROCEDURE initialize_seat_availability(
307.     IN p_train_id INT,
308.     IN p_journey_date DATE,
309.     IN p_class_id INT,
310.     IN p_rac_quota INT,
311.     IN p_waiting_list_quota INT
312. )
313. BEGIN
314.     DECLARE v_total_seats INT;
315.
316.     -- Get total seats for the class
317.     SELECT seating_capacity INTO v_total_seats
318.     FROM carriage_class
319.     WHERE id = p_class_id;
320.
321.     -- Insert or update seat availability
322.     INSERT INTO seat_availability (
323.         train_id, journey_date, carriage_class_id,
324.         total_seats, available_seats, total_rac_seats, rac_seats, total_waiting_list,
waiting_list
325.     ) VALUES (
326.         p_train_id, p_journey_date, p_class_id,
327.         v_total_seats, v_total_seats, p_rac_quota, p_rac_quota, p_waiting_list_quota,
p_waiting_list_quota
328.     )
329.     ON DUPLICATE KEY UPDATE
330.         total_seats = v_total_seats,
331.         available_seats = v_total_seats,
332.         rac_seats = p_rac_quota,
333.         waiting_list = p_waiting_list_quota;
334. END //
335. DELIMITER ;
```

Query for PNR tracking using ticket no.:

```
SELECT
    b.ticket_no,
    u.username AS passenger_name,
    u.category AS passenger_category,
    ts.name AS start_station,
    te.name AS end_station,
    tr.name AS train_name,
    cc.class_name AS ticket_class,
    b.seat_no,
    b.position_number,
    bs.status AS booking_status,
    b.booking_date,
    b.journey_date,
    b.distance,
    t.amount AS paid_amount,
    t.status AS transaction_status
FROM
    booking b
JOIN user u ON b.passenger_id = u.id
JOIN booking_status bs ON b.status_id = bs.id
JOIN train_station ts ON b.starting_station_id = ts.id
JOIN train_station te ON b.ending_station_id = te.id
JOIN train tr ON b.train_id = tr.id
JOIN carriage_class cc ON b.ticket_class_id = cc.id
JOIN transaction t ON b.transaction_id = t.id
WHERE
    b.ticket_no = 2;

```

Query for train schedule lookup

```
SELECT
    js.stop_order,
    ts.name AS station_name,
    ts.city,
    ts.state,
    js.distance_from_start,
    js.departure_time
FROM
    journey_station js
JOIN train_station ts ON js.station_id = ts.id
WHERE
    js.train_id =  22896
ORDER BY
    js.stop_order;

```

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

```sql
SELECT
    sa.available_seats,
    sa.rac_seats,
    sa.waiting_list,
    sa.total_seats,
    sa.total_rac_seats,
    sa.total_waiting_list
FROM
    seat_availability sa
WHERE
    sa.train_id = 22896
    AND sa.journey_date = '2025-04-19'
    AND sa.carriage_class_id = 1;
```

Query for Listing all passengers traveling on a specific train on a given date.

```sql
SELECT
    b.ticket_no,
    u.username AS passenger_name,
    u.email,
    u.mob AS mobile,
    u.category AS passenger_category,
    ts_start.name AS starting_station,
    ts_end.name AS ending_station,
    cc.class_name AS ticket_class,
    b.seat_no,
    bs.status AS booking_status,
    b.booking_date,
    b.journey_date
FROM
    booking b
JOIN user u ON b.passenger_id = u.id
JOIN train_station ts_start ON b.starting_station_id = ts_start.id
JOIN train_station ts_end ON b.ending_station_id = ts_end.id
JOIN carriage_class cc ON b.ticket_class_id = cc.id
JOIN booking_status bs ON b.status_id = bs.id
WHERE
    b.train_id = 22896
    AND b.journey_date = '2025-04-19'
ORDER BY
    b.seat_no;

```

Query for Retrieving all waitlisted passengers for a particular train.

```sql
SELECT
    b.ticket_no,
    u.username AS passenger_name,
    u.email,
    u.mob AS mobile,
    u.category AS passenger_category,
    ts_start.name AS starting_station,
    ts_end.name AS ending_station,
    cc.class_name AS ticket_class,
    b.position_number AS waiting_position,
    b.journey_date,
    b.booking_date
FROM
    booking b
JOIN user u ON b.passenger_id = u.id
JOIN train_station ts_start ON b.starting_station_id = ts_start.id
JOIN train_station ts_end ON b.ending_station_id = ts_end.id
```

```
18. JOIN carriage_class cc ON b.ticket_class_id = cc.id
19. JOIN booking_status bs ON b.status_id = bs.id
20. WHERE
21.     b.train_id = 22896
22.     AND bs.status = 'WL'
23. ORDER BY
24.     b.journey_date, b.position_number;
```

Query for Finding total amount that needs to be refunded for cancelling a train.

```
 1. SELECT
 2.     SUM(t.amount) AS total_refund_amount
 3. FROM
 4.     booking b
 5. JOIN transaction t ON b.transaction_id = t.id
 6. JOIN booking_status bs ON b.status_id = bs.id
 7. WHERE
 8.     b.train_id = 22896
 9.     AND b.journey_date = '2025-04-19'
10.     AND t.status = 'DONE';
```

Query for Total revenue generated from ticket bookings over a specified period.

```
1. SELECT
2.     SUM(t.amount) AS total_revenue
3. FROM
4.     transaction t
5. WHERE
6.     t.status = 'DONE'
7.     AND DATE(t.date) BETWEEN '2025-04-16' AND '2025-04-17';
8.
```

Query for Cancellation of records with refund status.

```
 1. SELECT
 2.     b.ticket_no,
 3.     u.username AS passenger_name,
 4.     tr.name AS train_name,
 5.     b.journey_date,
 6.     t.amount AS paid_amount,
 7.     t.status AS transaction_status,
 8.     bs.status AS booking_status,
 9.     t.date AS transaction_date
10. FROM
11.     booking b
12. JOIN user u ON b.passenger_id = u.id
13. JOIN train tr ON b.train_id = tr.id
14. JOIN transaction t ON b.transaction_id = t.id
15. JOIN booking_status bs ON b.status_id = bs.id
16. WHERE
17.     bs.status = 'CANCELLED';
18.
```

Query for Finding the busiest route based on passenger count.

```
 1. SELECT
 2.     ts_start.name AS starting_station,
 3.     ts_end.name AS ending_station,
 4.     COUNT(*) AS passenger_count
 5. FROM
 6.     booking b
 7. JOIN train_station ts_start ON b.starting_station_id = ts_start.id
 8. JOIN train_station ts_end ON b.ending_station_id = ts_end.id
 9. GROUP BY
10.     b.starting_station_id,
11.     b.ending_station_id
12. ORDER BY
13.     passenger_count DESC
14. LIMIT 1;
15.
```

Query for Generating an itemized bill for a ticket including all charges.

```
 1. SELECT
 2.     b.ticket_no,
 3.     u.username AS passenger_name,
 4.     u.category AS passenger_category,
 5.     tr.name AS train_name,
 6.     cc.class_name,
 7.     b.distance,
 8.     cc.distance_price_multiplier,
 9.     (b.distance * cc.distance_price_multiplier) AS base_fare,
10.     c.concession,
11.     ROUND((b.distance * cc.distance_price_multiplier) * (1 - c.concession), 2) AS
discounted_fare,
12.     t.amount AS final_paid,
13.     t.status AS payment_status,
14.     b.journey_date
15. FROM
16.     booking b
17. JOIN user u ON b.passenger_id = u.id
18. JOIN train tr ON b.train_id = tr.id
19. JOIN carriage_class cc ON b.ticket_class_id = cc.id
20. JOIN concession c ON u.category = c.category
21. JOIN transaction t ON b.transaction_id = t.id
22. WHERE
23.     b.ticket_no = 66;
```

Team Members:

- **Srikant Sahoo (2302CS07)**
- **Bharath Nayak (2301CS11)**
- **Ashutosh Kanojia (2301CS10)**
- **Nikhil Somara (2302CS03)**