DBMS ASSIGNMENT

Nikki Gautam

Question 2:

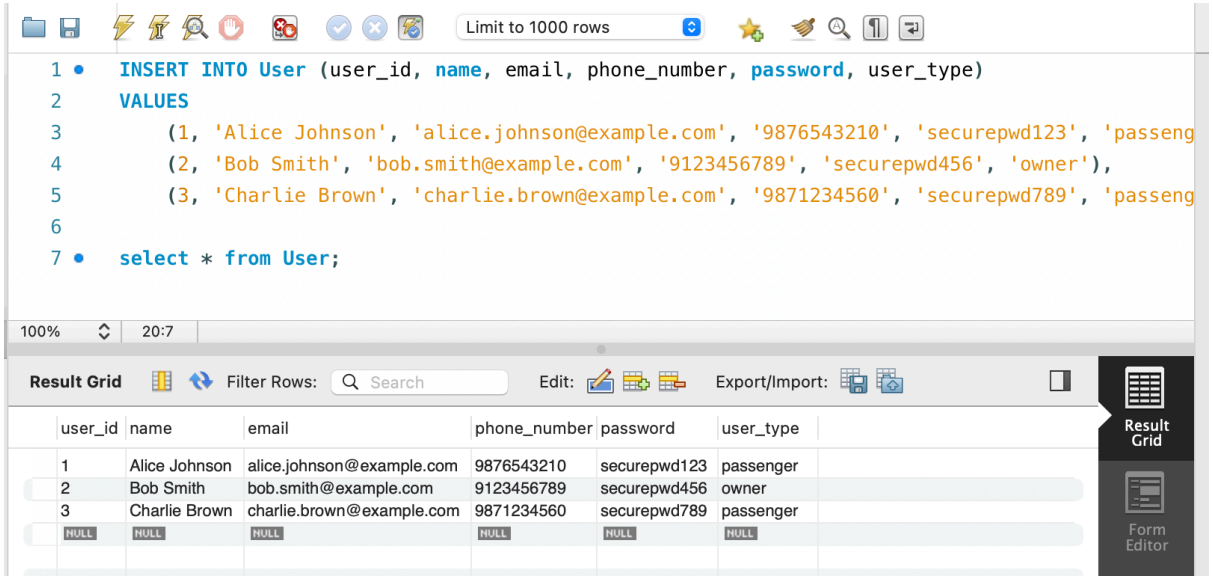
Part B:

Once you have created the database and tables, we want you populate your tables with some mock data. You can use any programming language for this or produce a SQL script/statement to insert the data.
We want you think about the data types, lengths of fields, constraints, keys etc. while you are at it and make it as real-world ready as it can be.

Inserting Mock Data in the tables: (Using SQL Queries)

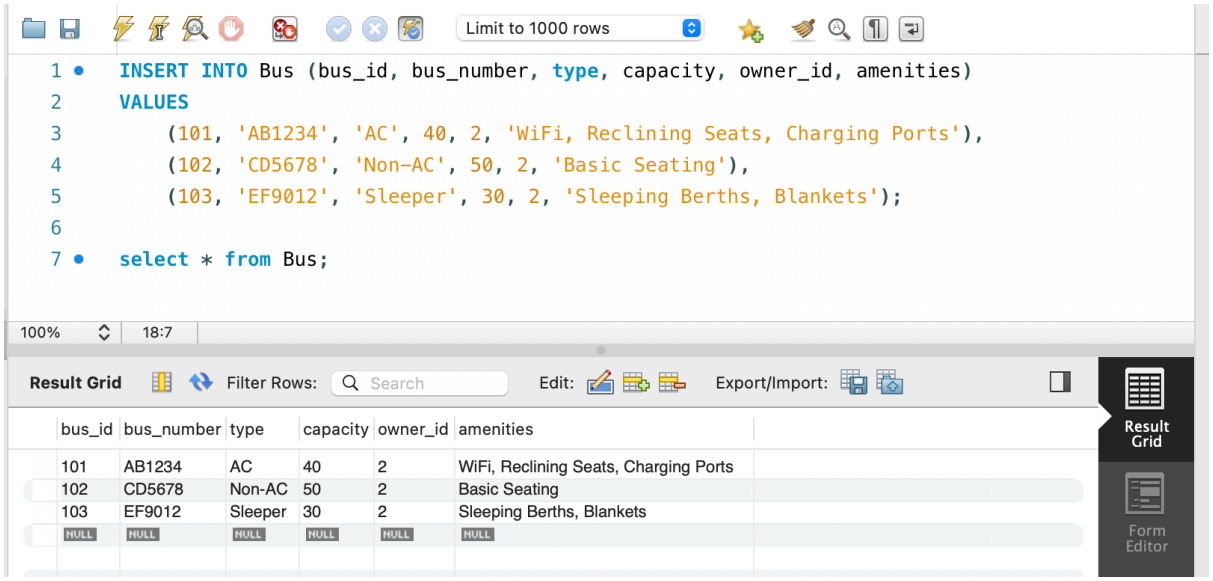
1. User table

```
INSERT INTO User (user_id, name, email, phone_number,
password, user_type)
VALUES
    (1, 'Alice Johnson', 'alice.johnson@example.com',
'9876543210', 'securepwd123', 'passenger'),
    (2, 'Bob Smith', 'bob.smith@example.com',
'9123456789', 'securepwd456', 'owner'),
    (3, 'Charlie Brown', 'charlie.brown@example.com',
'9871234560', 'securepwd789', 'passenger');
```



2. Bus table

```
INSERT INTO Bus (bus_id, bus_number, type, capacity,
owner_id, amenities)
VALUES
    (101, 'AB1234', 'AC', 40, 2, 'WiFi, Reclining
Seats, Charging Ports'),
    (102, 'CD5678', 'Non-AC', 50, 2, 'Basic
Seating'),
    (103, 'EF9012', 'Sleeper', 30, 2, 'Sleeping
Berths, Blankets');
```



3. Route table

```
INSERT INTO Route (route_id, source, destination,
total_distance, estimated_time)
VALUES
    (1, 'New York', 'Boston', 216.5, '04:00:00'),
    (2, 'San Francisco', 'Los Angeles', 383.0,
'06:30:00'),
    (3, 'Chicago', 'Detroit', 281.0, '04:45:00');
```

```sql
INSERT INTO Route (route_id, source, destination, total_distance, estimated_time)
VALUES
    (1, 'New York', 'Boston', 216.5, '04:00:00'),
    (2, 'San Francisco', 'Los Angeles', 383.0, '06:30:00'),
    (3, 'Chicago', 'Detroit', 281.0, '04:45:00');

select * from Route;
```

Result Grid

route_id	source	destination	total_distan...	estimated_ti...
1	New York	Boston	216.50	04:00:00
2	San Francisco	Los Angeles	383.00	06:30:00
3	Chicago	Detroit	281.00	04:45:00
NULL	NULL	NULL	NULL	NULL

4. Trip table

```sql
INSERT INTO Trip (trip_id, bus_id, route_id, trip_date, departure_time, arrival_time, status)
VALUES
    (1001, 101, 1, '2025-01-20', '08:00:00', '12:00:00', 'Scheduled'),
    (1002, 102, 2, '2025-01-21', '10:00:00', '16:30:00', 'Scheduled'),
    (1003, 103, 3, '2025-01-22', '06:15:00', '11:00:00', 'Scheduled');
```

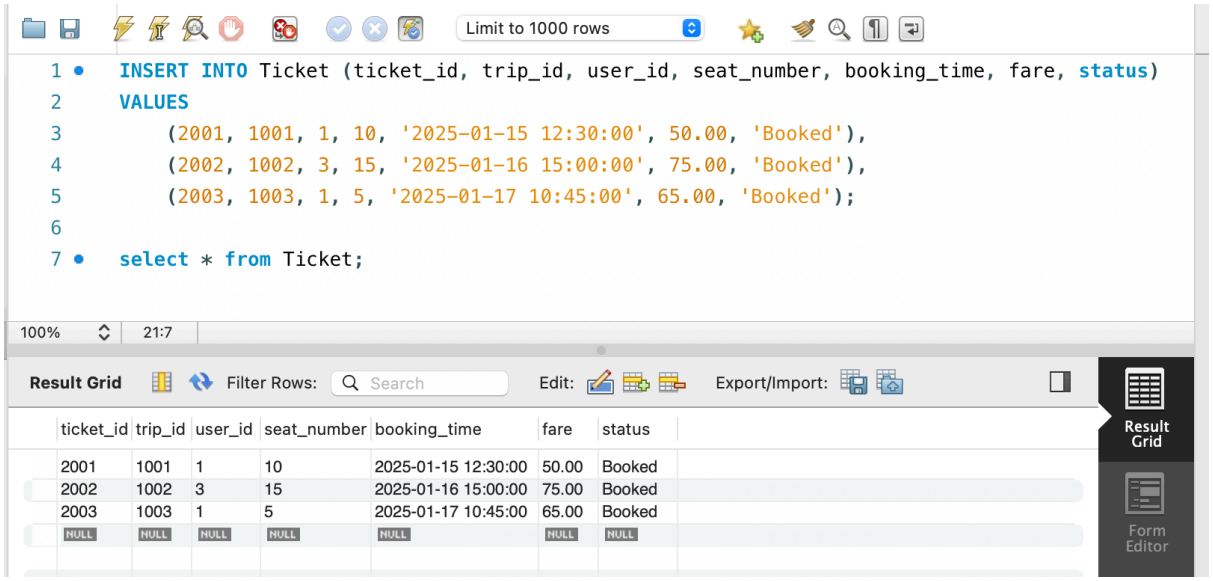
```sql
INSERT INTO Trip (trip_id, bus_id, route_id, trip_date, departure_time, arrival_time, status
VALUES
    (1001, 101, 1, '2025-01-20', '08:00:00', '12:00:00', 'Scheduled'),
    (1002, 102, 2, '2025-01-21', '10:00:00', '16:30:00', 'Scheduled'),
    (1003, 103, 3, '2025-01-22', '06:15:00', '11:00:00', 'Scheduled');

select * from Trip;
```

Result Grid

trip_id	bus_id	route_id	trip_date	departure_time	arrival_time	status
1001	101	1	2025-01-20	08:00:00	12:00:00	Scheduled
1002	102	2	2025-01-21	10:00:00	16:30:00	Scheduled
1003	103	3	2025-01-22	06:15:00	11:00:00	Scheduled
NULL	NULL	NULL	NULL	NULL	NULL	NULL

5. Ticket table

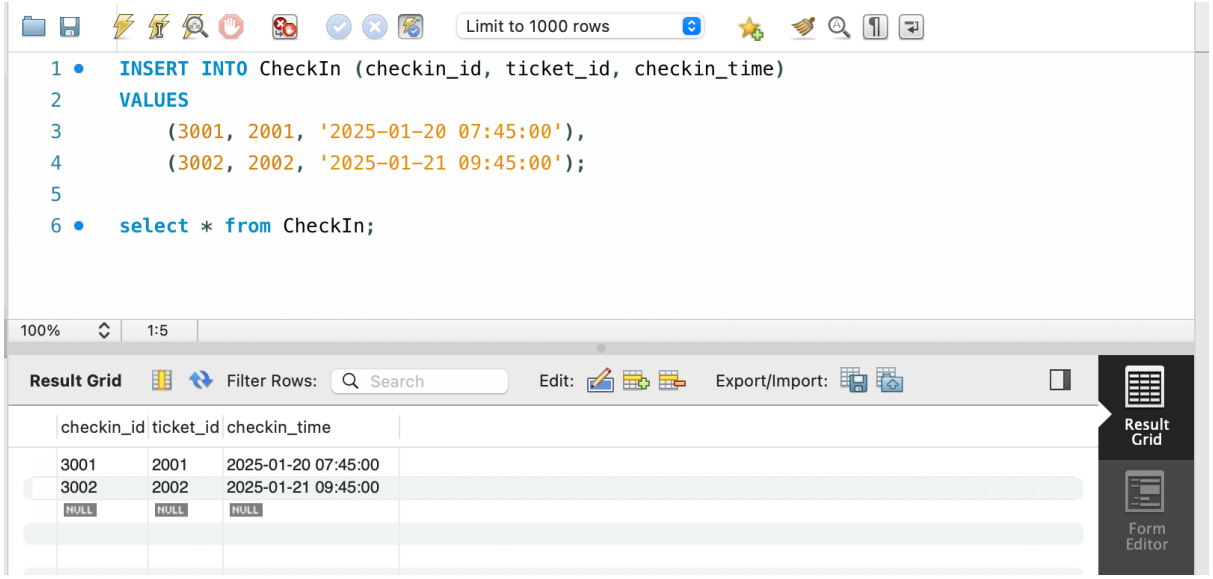
```
INSERT INTO Ticket (ticket_id, trip_id, user_id,
seat_number, booking_time, fare, status)
VALUES
    (2001, 1001, 1, 10, '2025-01-15 12:30:00', 50.00,
'Booked'),
    (2002, 1002, 3, 15, '2025-01-16 15:00:00', 75.00,
'Booked'),
    (2003, 1003, 1, 5, '2025-01-17 10:45:00', 65.00,
'Booked');
```



ticket_id	trip_id	user_id	seat_number	booking_time	fare	status
2001	1001	1	10	2025-01-15 12:30:00	50.00	Booked
2002	1002	3	15	2025-01-16 15:00:00	75.00	Booked
2003	1003	1	5	2025-01-17 10:45:00	65.00	Booked
NULL	NULL	NULL	NULL	NULL	NULL	NULL

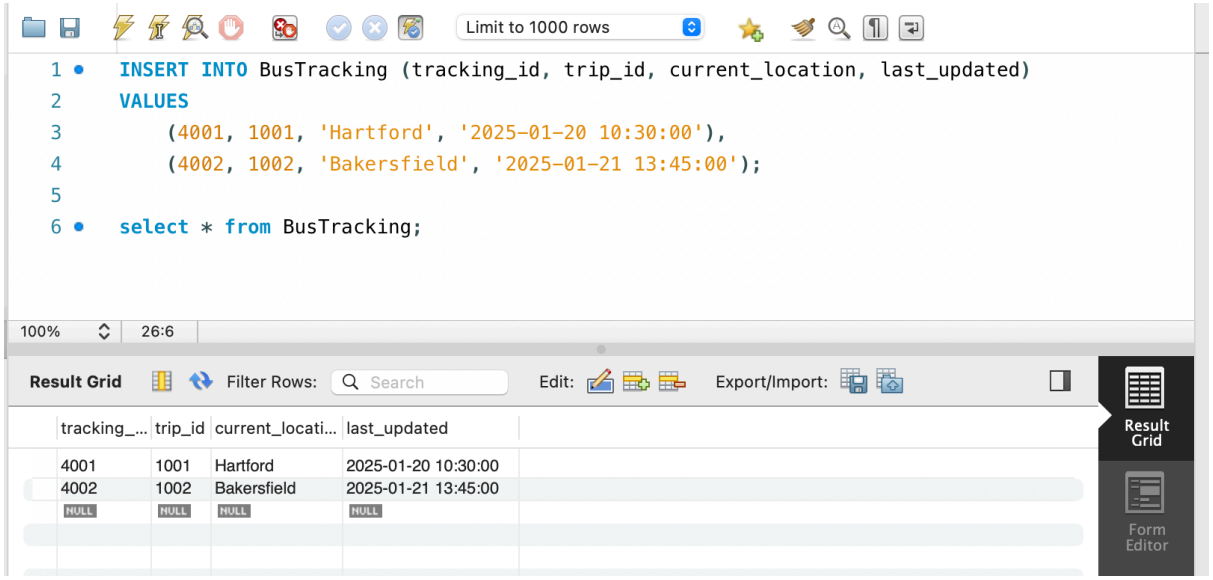
6. CheckIn table

```
INSERT INTO CheckIn (checkin_id, ticket_id,
checkin_time)
VALUES
    (3001, 2001, '2025-01-20 07:45:00'),
    (3002, 2002, '2025-01-21 09:45:00');
```



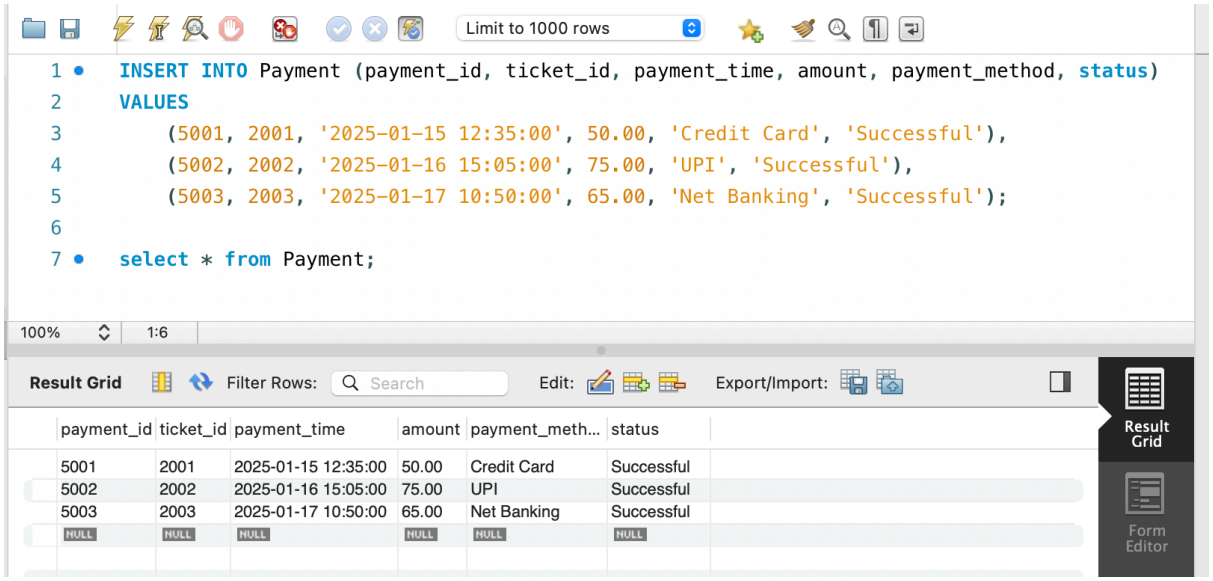
7. BusTracking table

```
INSERT INTO BusTracking (tracking_id, trip_id,
current_location, last_updated)
VALUES
    (4001, 1001, 'Hartford', '2025-01-20 10:30:00'),
    (4002, 1002, 'Bakersfield', '2025-01-21
13:45:00');
```



8. Payment table

```
INSERT INTO Payment (payment_id, ticket_id,
payment_time, amount, payment_method, status)
VALUES
    (5001, 2001, '2025-01-15 12:35:00', 50.00,
'Credit Card', 'Successful'),
    (5002, 2002, '2025-01-16 15:05:00', 75.00, 'UPI',
'Successful'),
    (5003, 2003, '2025-01-17 10:50:00', 65.00, 'Net
Banking', 'Successful');
```



9. Feedback table

```
INSERT INTO Feedback (feedback_id, user_id, trip_id,
rating, comment, feedback_time)
VALUES
    (6001, 1, 1001, 5, 'Very comfortable ride!',
'2025-01-20 18:00:00'),
    (6002, 3, 1002, 4, 'Great experience, but the bus
was late.', '2025-01-21 20:00:00');
```

```sql
INSERT INTO Feedback (feedback_id, user_id, trip_id, rating, comment, feedback_time)
VALUES
    (6001, 1, 1001, 5, 'Very comfortable ride!', '2025-01-20 18:00:00'),
    (6002, 3, 1002, 4, 'Great experience, but the bus was late.', '2025-01-21 20:00:00');

select * from Feedback;
```

feedback_id	user_id	trip_id	rating	comment	feedback_time
6001	1	1001	5	Very comfortable ride!	2025-01-20 18:00:00
6002	3	1002	4	Great experience, but the bus was late.	2025-01-21 20:00:00
NULL	NULL	NULL	NULL	NULL	NULL

Data Distribution Justifications

1. **User Table:**
    - Unique constraints on email and phone to ensure no duplication
    - Passwords are securely stored (hashing can be used in practice)
2. **Bus Table:**
    - Amenities is stored as a comma-separated string to simplify frequent queries (Normalization can be applied later if required)
3. **Route Table:**
    - Distance and time are kept accurately to reflect real-world scenarios
4. **Trip Table:**
    - Each trip is scheduled with a specific bus and route
    - Status values ensure proper trip lifecycle management
5. **Ticket Table:**
    - Each ticket links users to trips
    - fare and status are included for efficient ticket management
6. **Check-in Table:**
    - Check-ins are recorded only for passengers who board the bus
7. **Bus Tracking Table:**
    - Tracks real-time location updates for ongoing trips

8. **Payment Table:**
    - Payment details are critical for ticket management
9. **Feedback Table:**
    - Feedback entries ensure trip quality monitoring

Improvements for the future (also mentioned in Part C)

- Use hashed passwords in "**User**" for security.
- Implement "**BusAmenities**" as a normalized table for large data.
- Add unique constraints and foreign keys to ensure referential integrity.