

DBMS ASSIGNMENT

Nikki Gautam

Question 2:

Part B:

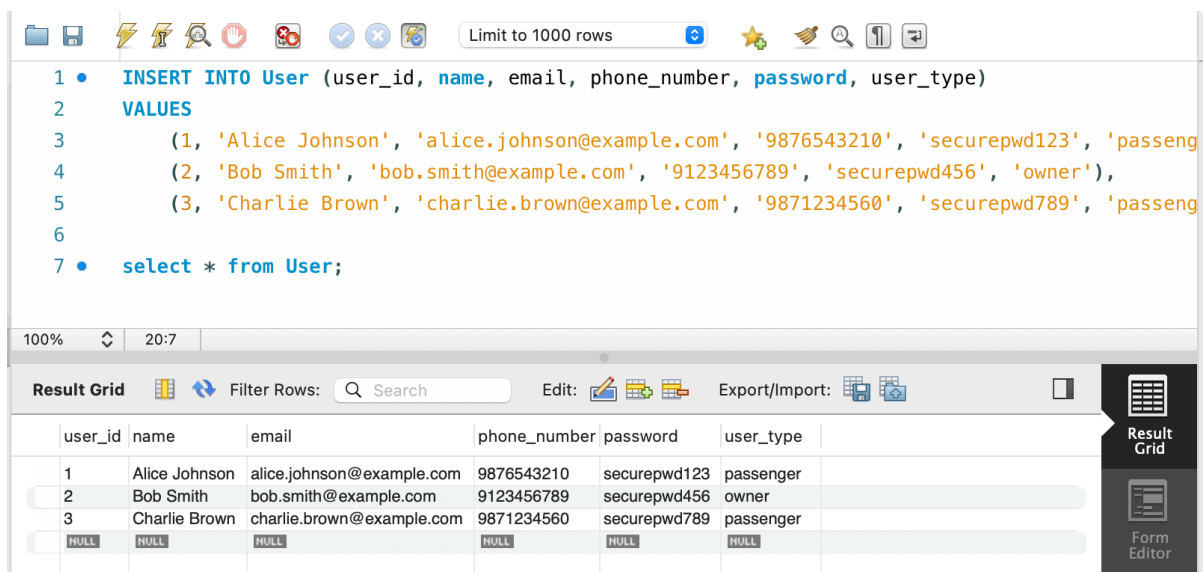
Once you have created the database and tables, we want you populate your tables with some mock data. You can use any programming language for this or produce a SQL script/statement to insert the data.

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');
```

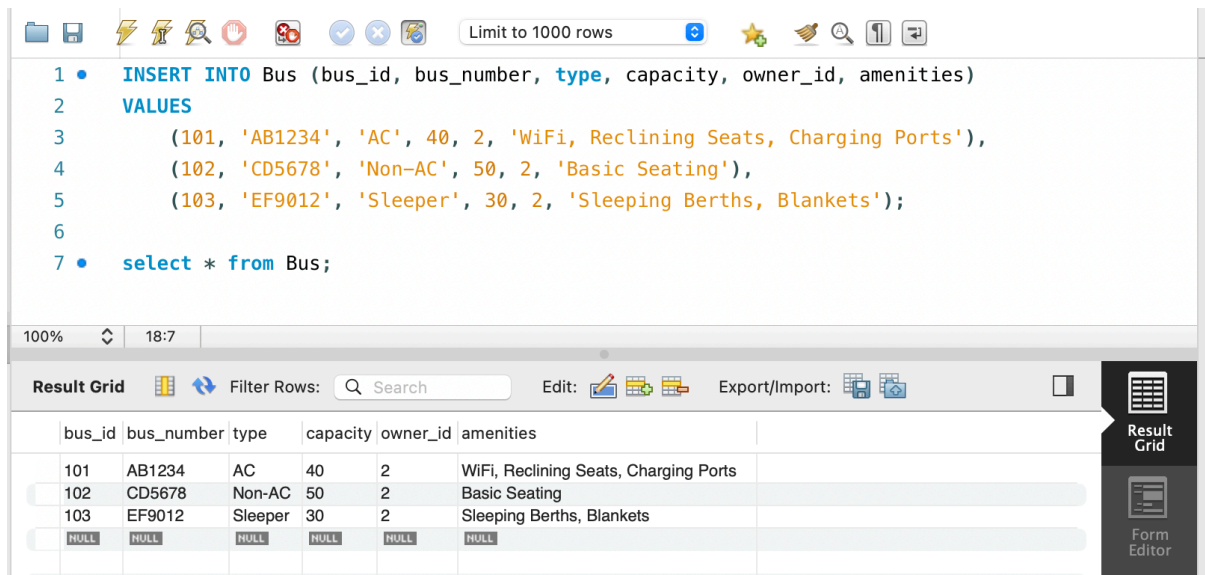


The screenshot shows a database management interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, the SQL script is displayed in a text area, numbered 1 through 7. The script consists of an INSERT statement followed by three rows of data, and a SELECT statement. Below the script, there's a 'Result Grid' section showing the data inserted into the 'User' table. The grid has columns for user_id, name, email, phone_number, password, and user_type. The data is displayed in a table format with alternating row colors. On the right side of the grid, there are buttons for 'Result Grid' and 'Form Editor'.

| user_id | name | email | phone_number | password | user_type |
|---------|---------------|---------------------------|--------------|--------------|-----------|
| 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 |
| NULL | NULL | NULL | NULL | NULL | NULL |

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');
```



The screenshot shows a database management interface. The top toolbar includes icons for file operations, a search bar, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following code:

```
1 • INSERT INTO Bus (bus_id, bus_number, type, capacity, owner_id, amenities)
2   VALUES
3     (101, 'AB1234', 'AC', 40, 2, 'WiFi, Reclining Seats, Charging Ports'),
4     (102, 'CD5678', 'Non-AC', 50, 2, 'Basic Seating'),
5     (103, 'EF9012', 'Sleeper', 30, 2, 'Sleeping Berths, Blankets');
6
7 • select * from Bus;
```

Below the editor, the 'Result Grid' is displayed, showing the data inserted into the 'Bus' table. The grid has columns for bus_id, bus_number, type, capacity, owner_id, and amenities. The first three rows contain the data from the INSERT statement, and the last row shows NULL values.

| bus_id | bus_number | type | capacity | owner_id | amenities |
|--------|------------|---------|----------|----------|---------------------------------------|
| 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 |
| NULL | NULL | NULL | NULL | NULL | NULL |

On the right side of the interface, there are buttons for 'Result Grid' and 'Form Editor'.

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');
```

Limit to 1000 rows

```

1 • INSERT INTO Route (route_id, source, destination, total_distance, estimated_time)
2   VALUES
3     (1, 'New York', 'Boston', 216.5, '04:00:00'),
4     (2, 'San Francisco', 'Los Angeles', 383.0, '06:30:00'),
5     (3, 'Chicago', 'Detroit', 281.0, '04:45:00');
6
7 • select * from Route;

```

100% 1:6

Result Grid Filter Rows: Search Edit: Export/Import:

| 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 |

Result Grid Form Editor

4. Trip table

```

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');

```

Limit to 1000 rows

```

1 • INSERT INTO Trip (trip_id, bus_id, route_id, trip_date, departure_time, arrival_time, status)
2   VALUES
3     (1001, 101, 1, '2025-01-20', '08:00:00', '12:00:00', 'Scheduled'),
4     (1002, 102, 2, '2025-01-21', '10:00:00', '16:30:00', 'Scheduled'),
5     (1003, 103, 3, '2025-01-22', '06:15:00', '11:00:00', 'Scheduled');
6
7 • select * from Trip;

```

100% 1:6

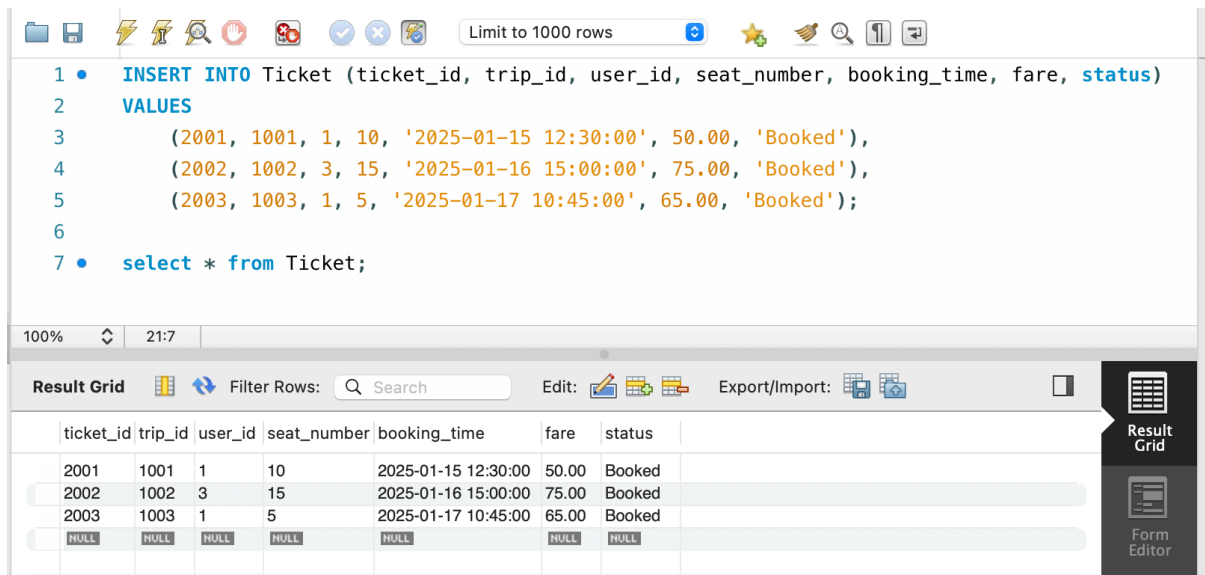
Result Grid Filter Rows: Search Edit: Export/Import:

| 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 |

Result Grid Form Editor

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');
```



The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Limit to 1000 rows' dropdown. The SQL editor contains the following code:

```
1 • INSERT INTO Ticket (ticket_id, trip_id, user_id, seat_number, booking_time, fare, status)
2 VALUES
3     (2001, 1001, 1, 10, '2025-01-15 12:30:00', 50.00, 'Booked'),
4     (2002, 1002, 3, 15, '2025-01-16 15:00:00', 75.00, 'Booked'),
5     (2003, 1003, 1, 5, '2025-01-17 10:45:00', 65.00, 'Booked');
6
7 • select * from Ticket;
```

Below the editor, the 'Result Grid' tab is active, displaying the following data:

| 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 |

On the right side of the interface, there are buttons for 'Result Grid' and 'Form Editor'.

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');
```

Limit to 1000 rows

```

1 • INSERT INTO CheckIn (checkin_id, ticket_id, checkin_time)
2   VALUES
3     (3001, 2001, '2025-01-20 07:45:00'),
4     (3002, 2002, '2025-01-21 09:45:00');
5
6 • select * from CheckIn;

```

100% 1:5

Result Grid Filter Rows: Search Edit: Export/Import:

| checkin_id | ticket_id | checkin_time |
|------------|-----------|---------------------|
| 3001 | 2001 | 2025-01-20 07:45:00 |
| 3002 | 2002 | 2025-01-21 09:45:00 |
| NULL | NULL | NULL |

Result Grid Form Editor

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');

```

Limit to 1000 rows

```

1 • INSERT INTO BusTracking (tracking_id, trip_id, current_location, last_updated)
2   VALUES
3     (4001, 1001, 'Hartford', '2025-01-20 10:30:00'),
4     (4002, 1002, 'Bakersfield', '2025-01-21 13:45:00');
5
6 • select * from BusTracking;

```

100% 26:6

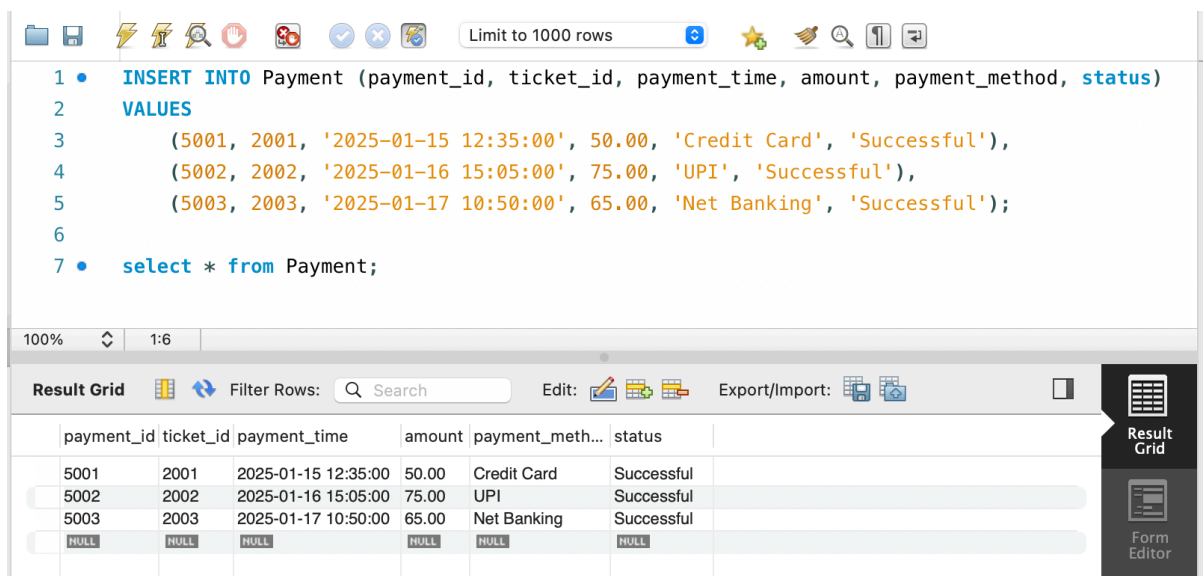
Result Grid Filter Rows: Search Edit: Export/Import:

| tracking_... | trip_id | current_locati... | last_updated |
|--------------|---------|-------------------|---------------------|
| 4001 | 1001 | Hartford | 2025-01-20 10:30:00 |
| 4002 | 1002 | Bakersfield | 2025-01-21 13:45:00 |
| NULL | NULL | NULL | NULL |

Result Grid Form Editor

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');
```



The screenshot shows a database management interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, the SQL query is displayed in a text area. The query consists of an INSERT statement followed by a SELECT statement. The results of the SELECT statement are shown in a table below the query. The table has columns for payment_id, ticket_id, payment_time, amount, payment_method, and status. The results show three rows of data, all with a status of 'Successful'. On the right side of the interface, there are buttons for 'Result Grid' and 'Form Editor'.

```
1 • INSERT INTO Payment (payment_id, ticket_id, payment_time, amount, payment_method, status)
2 VALUES
3     (5001, 2001, '2025-01-15 12:35:00', 50.00, 'Credit Card', 'Successful'),
4     (5002, 2002, '2025-01-16 15:05:00', 75.00, 'UPI', 'Successful'),
5     (5003, 2003, '2025-01-17 10:50:00', 65.00, 'Net Banking', 'Successful');
6
7 • select * from Payment;
```

| payment_id | ticket_id | payment_time | amount | payment_meth... | status |
|------------|-----------|---------------------|--------|-----------------|------------|
| 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 |
| NULL | NULL | NULL | NULL | NULL | NULL |

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');
```


The screenshot shows a database management interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, a SQL query is entered in a text area:

```

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

```

Below the query editor, there's a 'Result Grid' section. It includes a search bar, 'Filter Rows:' label, and 'Edit:' and 'Export/Import:' buttons. The result grid displays the following data:

| 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 |

On the right side of the result grid, there are two buttons: 'Result Grid' and 'Form Editor'.

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.