# DBMS ASSIGNMENT

Nikki Gautam

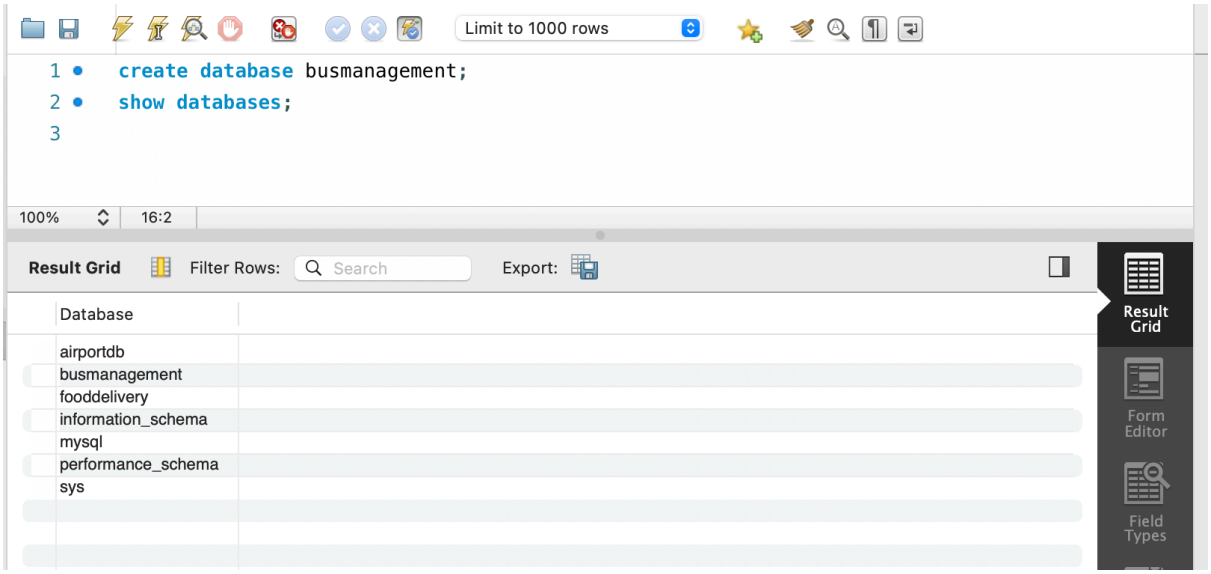
## Question 2:

Language Used (for querying): **SQL**

## Part A:

For the ER Model created in the earlier question, create the database tables, and normalize them. You are free to modify the original ER diagram with new findings / changes. You are also free to leave some tables in a denormalized state if you can justify it in the final report (part C).

**Database Created:** busmanagement

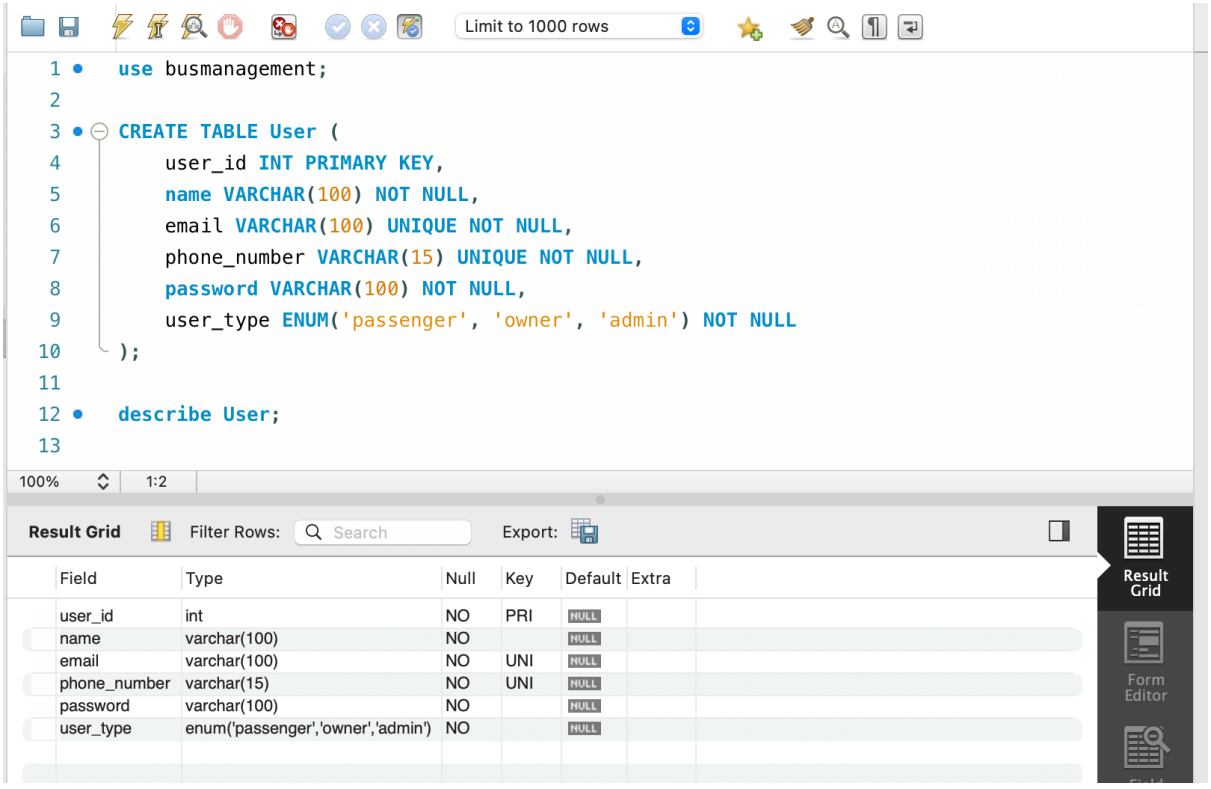


# Database Tables:

## 1. User Table

Used to store information of the users who interact with the system

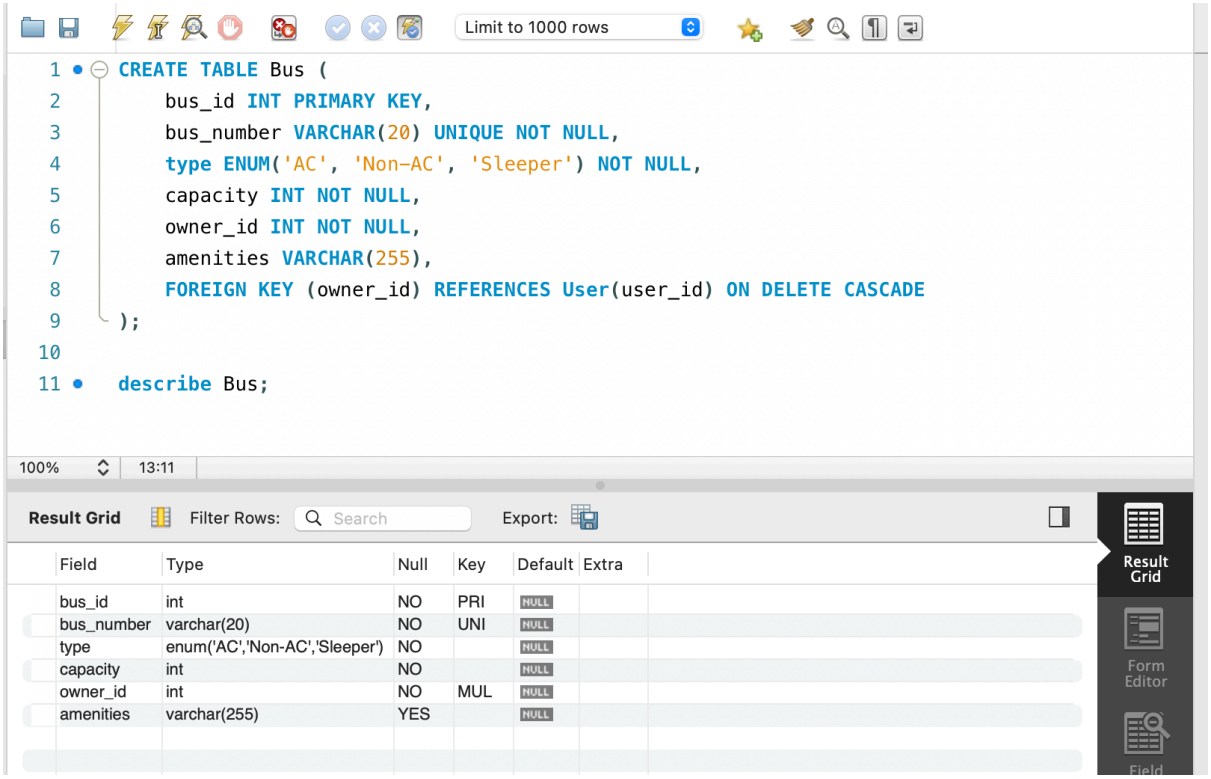
```
CREATE TABLE User (
    user_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone_number VARCHAR(15) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL,
    user_type ENUM('passenger', 'owner', 'admin') NOT
NULL
);
```



## 2. Bus Table

Used to store details about the buses owned by users

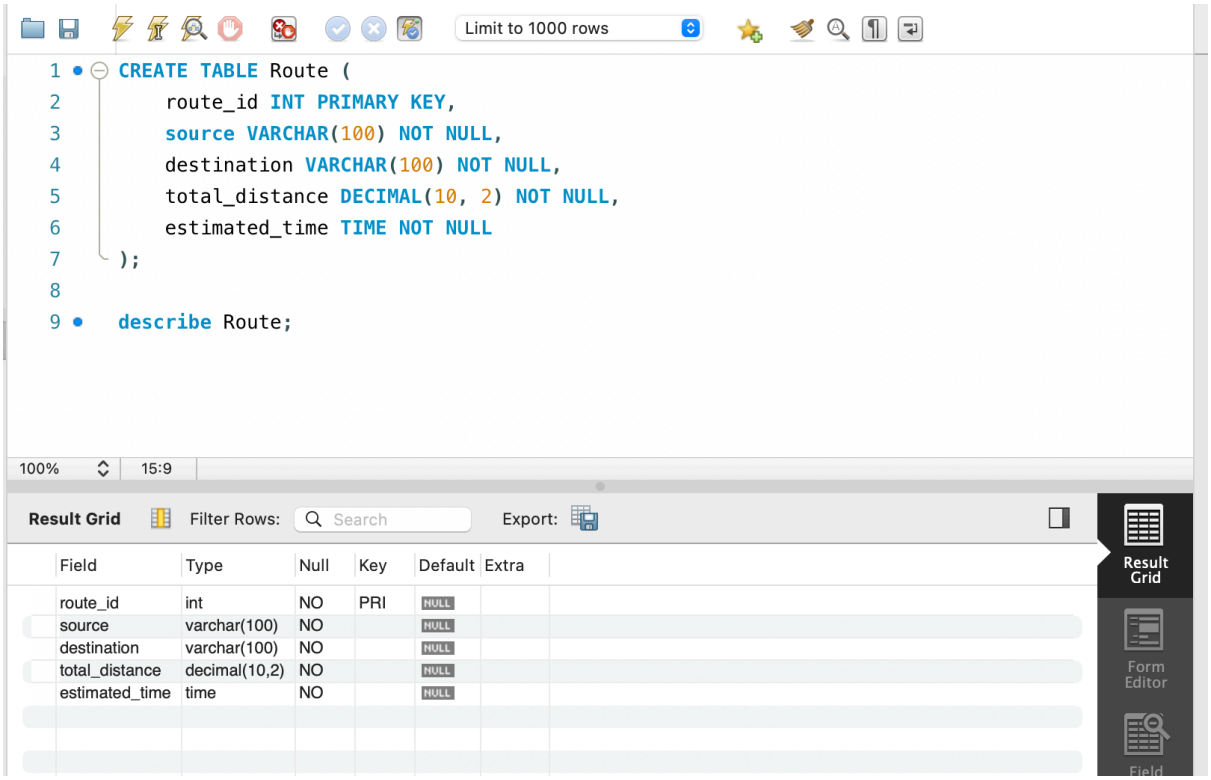
```
CREATE TABLE Bus (
    bus_id INT PRIMARY KEY,
    bus_number VARCHAR(20) UNIQUE NOT NULL,
    type ENUM('AC', 'Non-AC', 'Sleeper') NOT NULL,
    capacity INT NOT NULL,
    owner_id INT NOT NULL,
    amenities VARCHAR(255),
    FOREIGN KEY (owner_id) REFERENCES User(user_id)
ON DELETE CASCADE
);
```



## 3. Route Table

Used to store details of routes the buses take

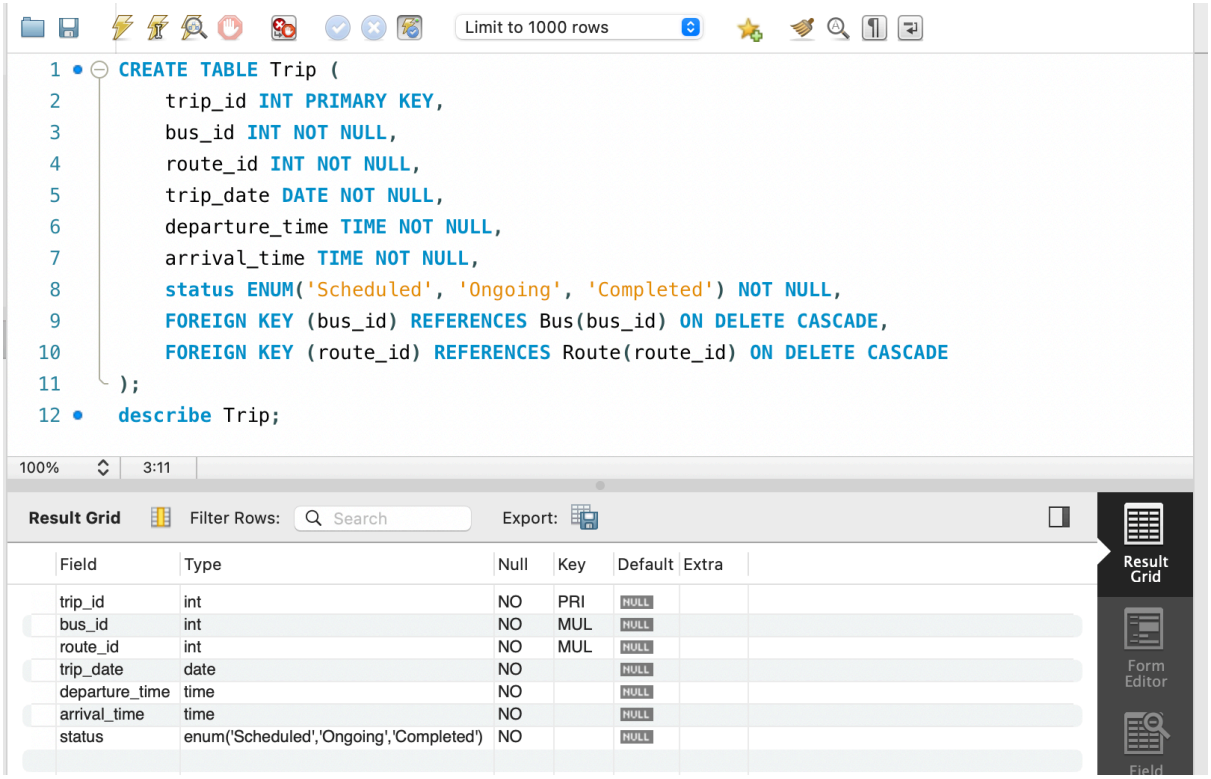
```
CREATE TABLE Route (
    route_id INT PRIMARY KEY,
    source VARCHAR(100) NOT NULL,
    destination VARCHAR(100) NOT NULL,
    total_distance DECIMAL(10, 2) NOT NULL,
    estimated_time TIME NOT NULL
);
```



## 4. Trip Table

Used to store link buses and routes to specific trips

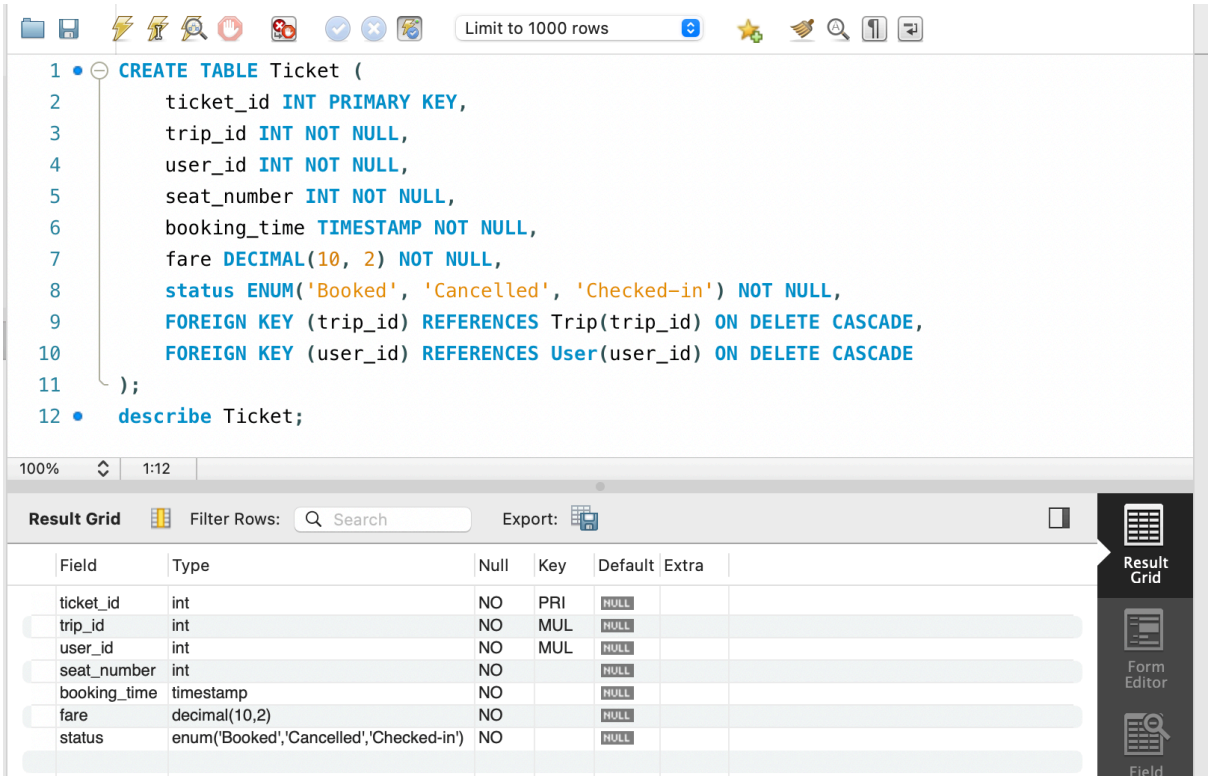
```
CREATE TABLE Trip (
    trip_id INT PRIMARY KEY,
    bus_id INT NOT NULL,
    route_id INT NOT NULL,
    trip_date DATE NOT NULL,
    departure_time TIME NOT NULL,
    arrival_time TIME NOT NULL,
    status ENUM('Scheduled', 'Ongoing', 'Completed')
NOT NULL,
    FOREIGN KEY (bus_id) REFERENCES Bus(bus_id) ON
DELETE CASCADE,
    FOREIGN KEY (route_id) REFERENCES Route(route_id)
ON DELETE CASCADE
);
```



## 5. Ticket Table

Used to store information of tickets booked by passengers for trips

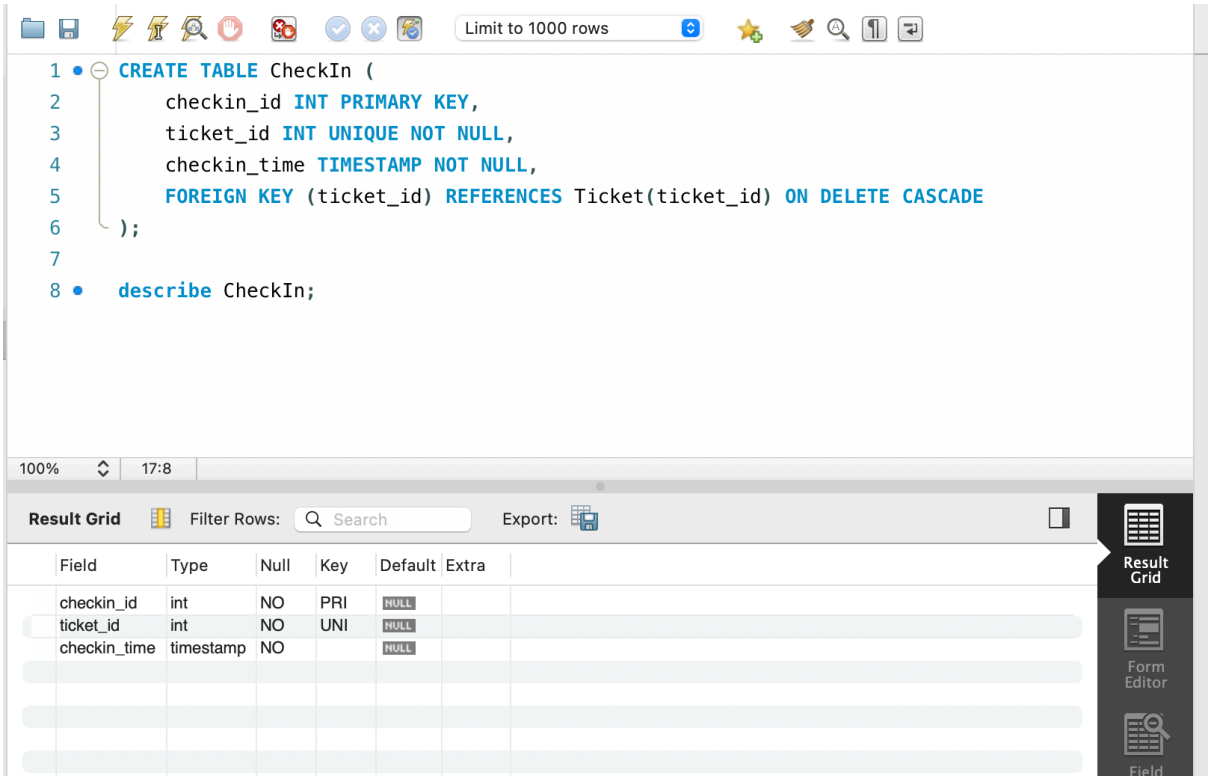
```
CREATE TABLE Ticket (
    ticket_id INT PRIMARY KEY,
    trip_id INT NOT NULL,
    user_id INT NOT NULL,
    seat_number INT NOT NULL,
    booking_time TIMESTAMP NOT NULL,
    fare DECIMAL(10, 2) NOT NULL,
    status ENUM('Booked', 'Cancelled', 'Checked-in')
NOT NULL,
    FOREIGN KEY (trip_id) REFERENCES Trip(trip_id) ON
DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON
DELETE CASCADE
);
```



## 6. Check-in Table

Used to store check-in information by passengers

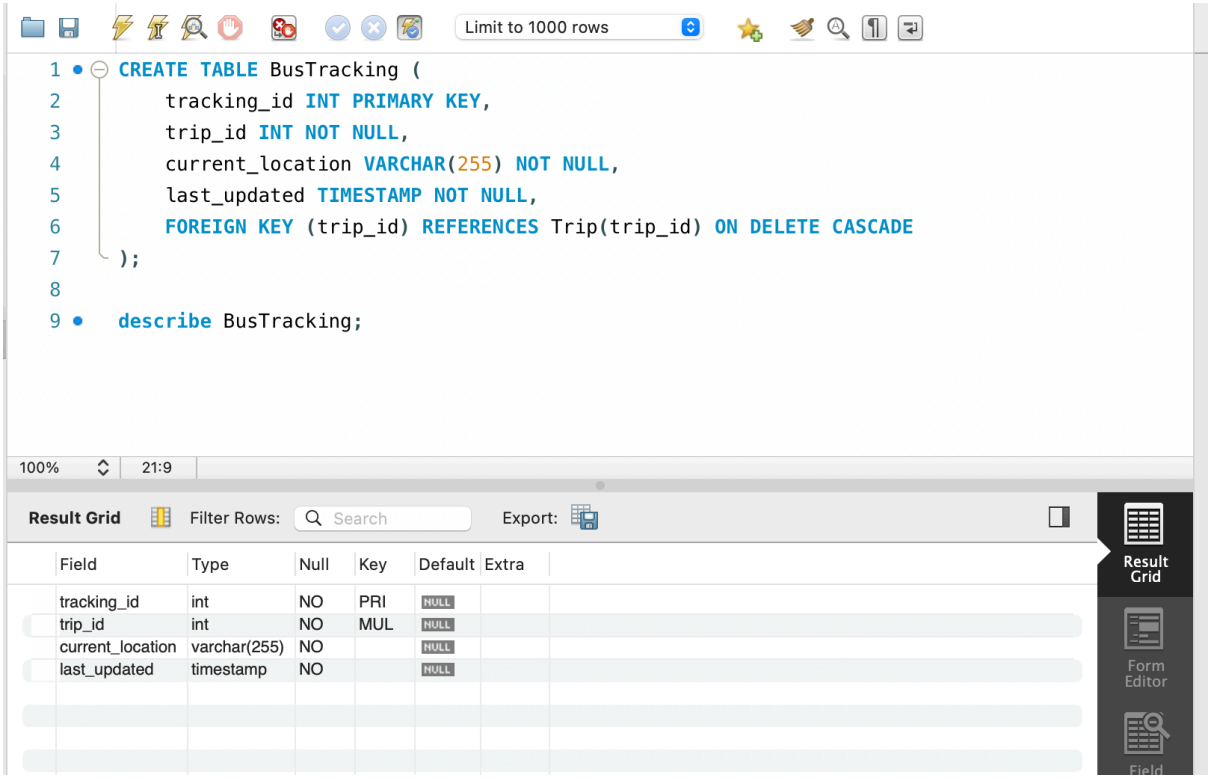
```
CREATE TABLE CheckIn (
    checkin_id INT PRIMARY KEY,
    ticket_id INT UNIQUE NOT NULL,
    checkin_time TIMESTAMP NOT NULL,
    FOREIGN KEY (ticket_id) REFERENCES
Ticket(ticket_id) ON DELETE CASCADE
);
```



## 7. Bus Tracking Table

Used to store information to track the location of buses during trips

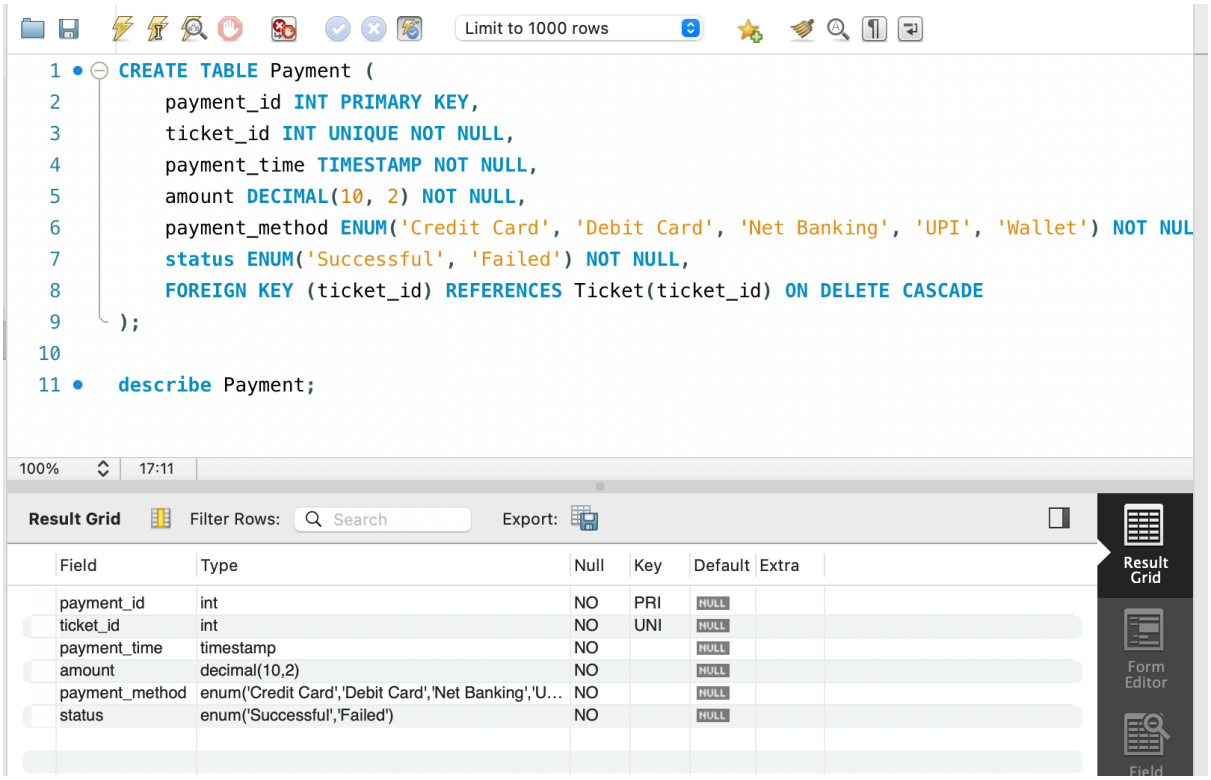
```
CREATE TABLE BusTracking (
    tracking_id INT PRIMARY KEY,
    trip_id INT NOT NULL,
    current_location VARCHAR(255) NOT NULL,
    last_updated TIMESTAMP NOT NULL,
    FOREIGN KEY (trip_id) REFERENCES Trip(trip_id) ON
DELETE CASCADE
);
```



## 8. Payment Table

Used to store information to manage payments for booked tickets

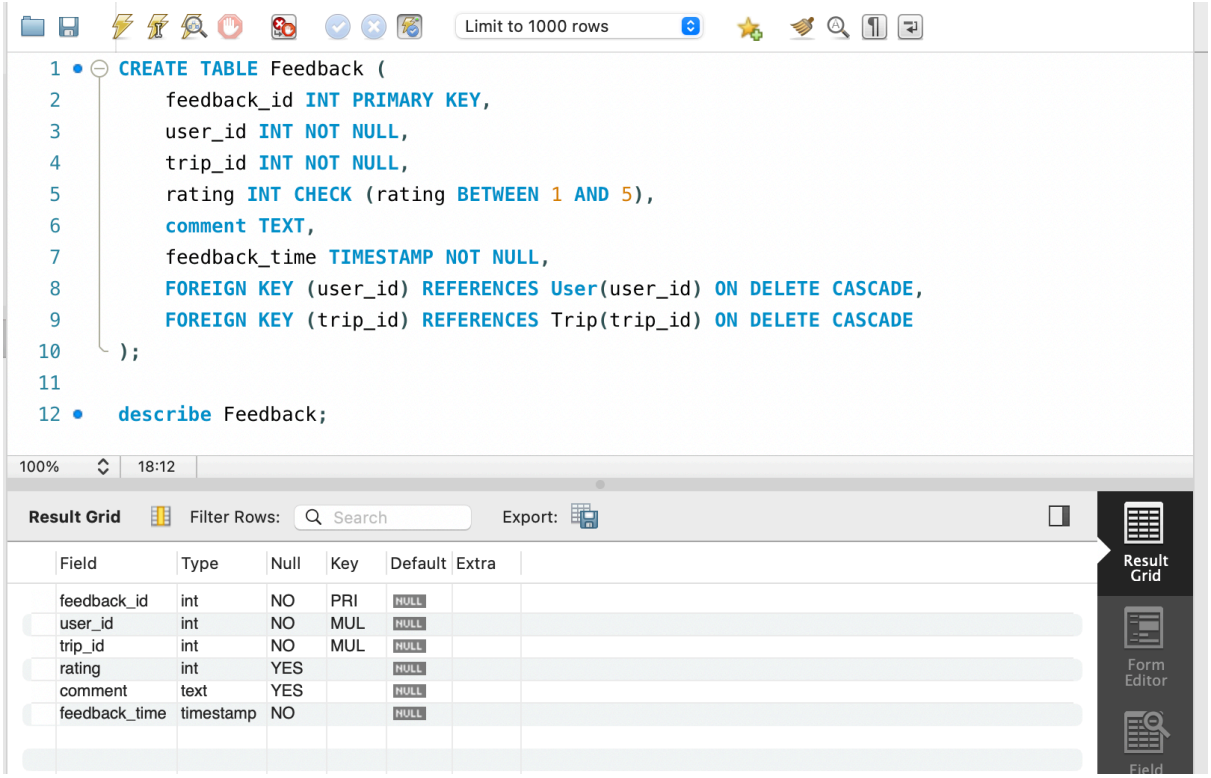
```
CREATE TABLE Payment (
    payment_id INT PRIMARY KEY,
    ticket_id INT UNIQUE NOT NULL,
    payment_time TIMESTAMP NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    payment_method ENUM('Credit Card', 'Debit Card',
'Net Banking', 'UPI', 'Wallet') NOT NULL,
    status ENUM('Successful', 'Failed') NOT NULL,
    FOREIGN KEY (ticket_id) REFERENCES
Ticket(ticket_id) ON DELETE CASCADE
);
```



## 9. Feedback Table

Used to store the feedback given by users for trips

```
CREATE TABLE Feedback (
    feedback_id INT PRIMARY KEY,
    user_id INT NOT NULL,
    trip_id INT NOT NULL,
    rating INT CHECK (rating BETWEEN 1 AND 5),
    comment TEXT,
    feedback_time TIMESTAMP NOT NULL,
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON
DELETE CASCADE,
    FOREIGN KEY (trip_id) REFERENCES Trip(trip_id) ON
DELETE CASCADE
);
```



## **Normalization (**Applied in the solution proposed**):**

### First Normal Form (1NF):

- All attributes are atomic and single-valued
- The table "**Bus**" (amenities) allow a list of amenities. To keep it atomic, a separate table can be created for bus amenities:

```
CREATE TABLE BusAmenities (
    bus_id INT NOT NULL,
    amenity VARCHAR(100) NOT NULL,
    PRIMARY KEY (bus_id, amenity),
    FOREIGN KEY (bus_id) REFERENCES Bus(bus_id)
ON DELETE CASCADE
);
```

### Second Normal Form (2NF):

- All non-key attributes are fully dependent on the primary key
- No changes are needed as the ER model already ensures no partial dependencies

### Third Normal Form (3NF):

- No transitive dependencies exist
- Tables like "**Ticket**" and "**Payment**" are already normalized, as every non-key attribute is dependent solely on the primary key

**Denormalized Tables (**Justification for keeping them as they are**):**

1. **Ticket Table**:
    - **Reason for Denormalization:** Instead of breaking "**fare**" or "**status**" into separate tables, keeping them in the "**Ticket**" table reduces join queries during frequent ticket-related operations.

2. **BusTracking Table**:
    - **Reason for Denormalization:** "**current_location**" is updated frequently, and keeping this denormalized avoids joins when fetching the latest location for display in real-time.

- **Since normalization ensures consistency and removes redundancy.** Hence, most tables are normalized to **3NF**
- **Denormalization improves performance** for frequently accessed attributes like "**current_location**" and "**fare**"
- The schema supports scalability, frequent updates, and real-time tracking while ensuring data integrity.

Final glimpse of "**busmanagement**":