Ride-Booking System: A Transactional Database for Mobility Service Providers

Design Requirements:

In the Design Requirements, we focused on planning how the Mobility Service system will work by designing its database structure. The goal was to make sure the system can handle important information like users, vehicles, drivers, customers, rides, and payments in an organized way. We ensured that each piece of data is stored properly, with no unnecessary repetition, and that all information is easy to update or delete when needed.

Database Normalization:

1. Atomicity:
    - Each column in every table holds atomic values. For example, in the drivers and customers tables, columns like phone_number and email store only one phone number or email per record, adhering to the principle of atomicity.
2. Elimination of Repeating Groups:
    - There are no repeating groups in the tables. Each column stores a single value per record. In the rides table, for instance, the columns like pickup_location, drop_location, and fare each hold a distinct value per ride. This ensures compliance with 1NF.
3. Unique Rows:
    - Each table uses primary keys to maintain uniqueness. For instance, the driver_id in the drivers table guarantees each driver is unique, and the same applies to other tables like rides and customers.
4. Elimination of Partial Dependencies:
    - In the drivers and customers tables, all non-key attributes depend entirely on their respective primary keys (driver_id and customer_id). For example, attributes like first_name, email, and phone_number in the drivers table depend solely on driver_id. This satisfies the second normal form (2NF).

5. Elimination of Transitive Dependencies:
    - The design avoids transitive dependencies. For example, in the drivers table, attributes like email, phone_number, and vehicle_id depend directly on driver_id and not on other non-key attributes. This ensures compliance with 3NF.
6. Referential Integrity:
    - Foreign keys are correctly set up to maintain referential integrity across all tables. For example, ride_id in the payment_details table references rides(ride_id), ensuring that payment details are tied to valid rides. Similarly, user_id in the rewards table ensures that rewards are linked to valid users.

Database Schema Overview:

1. Roles Table

- **Columns**:
    - role_id (Primary Key) - Unique identifier for each role.
    - role_name - Name of the role (e.g., 'Admin', 'Driver', 'Customer').
- **Primary Key**:
    - role_id
- **Foreign Keys**: None

2. Login Details Table

- **Columns**:
    - user_id (Primary Key) - Unique identifier for each user.
    - username - User's login name (must be unique).
    - password - User's password.
    - last_login_at - Timestamp for the last time the user logged in.
    - role_id (Foreign Key) - References roles(role_id), indicating the user's role.

- **Primary Key**:
    - user_id
- **Foreign Keys**:
    - role_id references roles(role_id)

3. Vehicles Table

- **Columns**:
    - vehicle_id (Primary Key) - Unique identifier for each vehicle.
    - vehicle_type - Type of vehicle (e.g., 'Sedan', 'SUV').
    - capacity - Maximum capacity of passengers the vehicle can hold.
    - license_plate - Unique vehicle license plate number.
    - model - Model of the vehicle.
    - make - Make or manufacturer of the vehicle.
    - year - Year of manufacture.
- **Primary Key**:
    - vehicle_id
- **Foreign Keys**: None

4. Drivers Table

- **Columns**:
    - driver_id (Primary Key) - Unique identifier for each driver.
    - first_name - Driver's first name.
    - middle_name - Driver's middle name (optional).
    - last_name - Driver's last name.

- email - Driver's email address (unique).
- phone_number - Driver's phone number.
- license_number - Unique license number for the driver.
- vehicle_id (Foreign Key) - References vehicles(vehicle_id), indicating which vehicle the driver is assigned to.
- is_active - Indicates whether the driver is active (boolean).
- date_joined - Timestamp of when the driver joined.

- **Primary Key**:
  - driver_id
- **Foreign Keys**:
  - vehicle_id references vehicles(vehicle_id)
  - user_id (from login_details) references login_details(user_id) as a foreign key for authentication.

5. Customers Table

- **Columns**:
  - customer_id (Primary Key) - Unique identifier for each customer.
  - first_name - Customer's first name.
  - middle_name - Customer's middle name (optional).
  - last_name - Customer's last name.
  - email - Customer's email address (unique).
  - phone_number - Customer's phone number.
  - date_registered - Timestamp of when the customer registered.
  - is_active - Indicates whether the customer is active (boolean).
- **Primary Key**:

- customer_id

- **Foreign Keys**:
  - user_id (from login_details) references login_details(user_id) as a foreign key for authentication.

6. Rides Table

- **Columns**:
  - ride_id (Primary Key) - Unique identifier for each ride.
  - driver_id (Foreign Key) - References drivers(driver_id), indicating the driver for the ride.
  - customer_id (Foreign Key) - References customers(customer_id), indicating the customer for the ride.
  - pickup_location - Location where the ride starts.
  - drop_location - Location where the ride ends.
  - distance - Distance traveled during the ride (in kilometers or miles).
  - fare - Fare charged for the ride.
  - date - Timestamp of when the ride was booked or completed.
  - customer_rating - Rating given by the customer (if applicable).
  - comments - Optional comments provided by the customer.
  - status - Status of the ride (e.g., 'InTransit', 'Completed', 'Cancelled').
- **Primary Key**:
  - ride_id
- **Foreign Keys**:
  - driver_id references drivers(driver_id)
  - customer_id references customers(customer_id)

7. Ride Geolocations Table

- **Columns**:
    - route_id (Primary Key) - Unique identifier for each geolocation record.
    - ride_id (Foreign Key) - References rides(ride_id), linking geolocation data to a ride.
    - latitude - Latitude of the location point.
    - longitude - Longitude of the location point.
    - timestamp - Timestamp of when the geolocation data was recorded.
- **Primary Key**:
    - route_id
- **Foreign Keys**:
    - ride_id references rides(ride_id)

8. Payment Details Table

- **Columns**:
    - payment_id (Primary Key) - Unique identifier for each payment.
    - ride_id (Foreign Key) - References rides(ride_id), linking payment details to a ride.
    - payment_method - Method used for payment (e.g., 'Cash', 'CreditCard', 'DebitCard', 'Wallet').
    - amount - Amount paid for the ride.
    - transaction_status - Status of the transaction (e.g., 'Pending', 'Completed', 'Failed').
    - transaction_id - Unique transaction identifier for the payment.

- payment_date - Timestamp of when the payment was made.

- **Primary Key**:
  - payment_id
- **Foreign Keys**:
  - ride_id references rides(ride_id)

9. Rewards Table

- **Columns**:
  - reward_id (Primary Key) - Unique identifier for each reward.
  - user_id (Foreign Key) - References login_details(user_id), linking the reward to the user.
  - points_earned - Number of points earned for the reward.
  - category - Category of the reward (e.g., 'Ride', 'Referral', 'Promotion').
  - description - Description of how the reward was earned.
  - date_earned - Timestamp of when the reward was earned.
- **Primary Key**:
  - reward_id
- **Foreign Keys**:
  - user_id references login_details(user_id) (for drivers, customers, and rewards)

SQL script for creating the MobilityServiceDB:

```
CREATE DATABASE MobilityServiceDB;

USE MobilityServiceDB;
```

```
-- Table Creation
CREATE TABLE roles (
    role_id INTEGER PRIMARY KEY AUTO_INCREMENT,
    role_name VARCHAR(50) UNIQUE NOT NULL
);

CREATE TABLE login_details (
    user_id INTEGER PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    last_login_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    role_id INTEGER NOT NULL,  -- Ensure role_id is INTEGER
    FOREIGN KEY (role_id) REFERENCES roles(role_id) ON DELETE CASCADE
);



CREATE TABLE vehicles (
    vehicle_id INTEGER PRIMARY KEY AUTO_INCREMENT,
    vehicle_type VARCHAR(20) NOT NULL,
    capacity INTEGER NOT NULL,
    license_plate VARCHAR(20) UNIQUE NOT NULL,
```

```
    model VARCHAR(50) NOT NULL,
    make VARCHAR(50) NOT NULL,
    year INTEGER NOT NULL
);




CREATE TABLE drivers (
    driver_id INTEGER PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    middle_name VARCHAR(50),
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone_number VARCHAR(15) NOT NULL,
    license_number VARCHAR(50) UNIQUE NOT NULL,
    vehicle_id INTEGER,
    is_active BOOLEAN DEFAULT TRUE,
    date_joined TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (driver_id) REFERENCES login_details(user_id) ON DELETE CASCADE,
    FOREIGN KEY (vehicle_id) REFERENCES vehicles(vehicle_id) ON DELETE SET NULL
);
```

```
CREATE TABLE customers (
    customer_id INTEGER PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    middle_name VARCHAR(50),
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone_number VARCHAR(15) NOT NULL,
    date_registered TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    is_active BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (customer_id) REFERENCES login_details(user_id) ON DELETE CASCADE
);



CREATE TABLE rides (
    ride_id INTEGER PRIMARY KEY AUTO_INCREMENT,
    driver_id INTEGER NOT NULL,
    customer_id INTEGER NOT NULL,
    pickup_location VARCHAR(255) NOT NULL,
    drop_location VARCHAR(255) NOT NULL,
    distance FLOAT NOT NULL,
    fare FLOAT NOT NULL,
```

```
    date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    customer_rating FLOAT,
    comments TEXT,
    status VARCHAR(50) CHECK (status IN ('InTransit', 'Completed', 'Cancelled')) NOT NULL,
    FOREIGN KEY (driver_id) REFERENCES drivers(driver_id) ON DELETE CASCADE,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
);

CREATE TABLE ride_geolocations (
    route_id INTEGER PRIMARY KEY AUTO_INCREMENT,
    ride_id INTEGER NOT NULL,
    latitude FLOAT NOT NULL,
    longitude FLOAT NOT NULL,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (ride_id) REFERENCES rides(ride_id) ON DELETE CASCADE
);

CREATE TABLE payment_details (
    payment_id INTEGER PRIMARY KEY AUTO_INCREMENT,
```

```
    ride_id INTEGER NOT NULL,
    payment_method VARCHAR(20) CHECK (payment_method IN ('Cash', 'CreditCard', 'DebitCard', 'Wallet')) NOT NULL,
    amount FLOAT NOT NULL,
    transaction_status VARCHAR(20) CHECK (transaction_status IN ('Pending', 'Completed', 'Failed')) NOT NULL,
    transaction_id VARCHAR(100) UNIQUE,
    payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (ride_id) REFERENCES rides(ride_id) ON DELETE CASCADE
);

CREATE TABLE rewards (
    reward_id INTEGER PRIMARY KEY AUTO_INCREMENT,
    user_id INTEGER NOT NULL,
    points_earned INTEGER NOT NULL,
    category VARCHAR(20) CHECK (category IN ('Ride', 'Referral', 'Promotion')) NOT NULL,
    description TEXT,
    date_earned TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES login_details(user_id) ON DELETE CASCADE
);

-- Index Creation
CREATE INDEX idx_role_name ON roles (role_name);
```

```
CREATE INDEX idx_username ON login_details (username);

CREATE INDEX idx_role_id ON login_details (role_id);

CREATE INDEX idx_license_plate ON vehicles (license_plate);

CREATE INDEX idx_vehicle_type ON vehicles (vehicle_type);

CREATE INDEX idx_email ON drivers (email);

CREATE INDEX idx_license_number ON drivers (license_number);

CREATE INDEX idx_vehicle_id ON drivers (vehicle_id);

CREATE INDEX idx_customer_email ON customers (email);

CREATE INDEX idx_driver_id ON rides (driver_id);

CREATE INDEX idx_customer_id ON rides (customer_id);

CREATE INDEX idx_status ON rides (status);

CREATE INDEX idx_ride_id ON ride_geolocations (ride_id);

CREATE INDEX idx_ride_id_payment ON payment_details (ride_id);

CREATE INDEX idx_transaction_status ON payment_details (transaction_status);

CREATE INDEX idx_payment_method ON payment_details (payment_method);

CREATE INDEX idx_user_id ON rewards (user_id);

CREATE INDEX idx_category ON rewards (category);




-- Data insert statements


INSERT INTO roles (role_name) VALUES ('Admin'), ('Driver'), ('Customer');
```

```
INSERT INTO vehicles (vehicle_type, capacity, license_plate, model, make, year)
VALUES
('Truck', 2, 'US-TRK001', 'Model X', 'Tesla', 2023),
('Van', 8, 'US-VAN002', 'Transit', 'Ford', 2022),
('Truck', 4, 'US-TRK003', 'F-150', 'Ford', 2021),
('Van', 10, 'US-VAN004', 'Odyssey', 'Honda', 2020),
('Truck', 3, 'US-TRK005', 'RAM 1500', 'Dodge', 2023),
('Van', 12, 'US-VAN006', 'Pacifica', 'Chrysler', 2021),
('Truck', 5, 'US-TRK007', 'Silverado', 'Chevrolet', 2020),
('Van', 9, 'US-VAN008', 'Sienna', 'Toyota', 2021),
('Truck', 6, 'US-TRK009', 'Tundra', 'Toyota', 2022),
('Van', 7, 'US-VAN010', 'Sprinter', 'Mercedes-Benz', 2023),
('Truck', 2, 'US-TRK011', 'Ram 2500', 'Dodge', 2022),
('Van', 10, 'US-VAN012', 'Viano', 'Mercedes-Benz', 2020),
('Truck', 3, 'US-TRK013', 'Hino 300', 'Hino', 2021),
('Van', 8, 'US-VAN014', 'Metris', 'Mercedes-Benz', 2022),
('Truck', 4, 'US-TRK015', 'Isuzu D-Max', 'Isuzu', 2020),
('Van', 6, 'US-VAN016', 'Transit Connect', 'Ford', 2023),
('Truck', 5, 'US-TRK017', 'Ford Ranger', 'Ford', 2021),
('Van', 9, 'US-VAN018', 'Express 3500', 'Chevrolet', 2020),
('Truck', 7, 'US-TRK019', 'Freightliner M2', 'Freightliner', 2023),
('Van', 8, 'US-VAN020', 'NV3500', 'Nissan', 2022);
```

```
-- Inserting login details for drivers with passwords based on username and special characters

INSERT INTO login_details (username, password, role_id)

VALUES

('john_doe', 'john_doe@123', 2),

('jane_smith', 'jane_smith#456', 2),

('michael_johnson', 'michael_johnson!789', 2),

('emily_williams', 'emily_williams$012', 2),

('david_brown', 'david_brown%345', 2),

('sophia_jones', 'sophia_jones^678', 2),

('james_miller', 'james_miller&910', 2),

('isabella_davis', 'isabella_davis*112', 2),

('william_garcia', 'william_garcia+131', 2),

('olivia_martinez', 'olivia_martinez@141', 2),

('benjamin_hernandez', 'benjamin_hernandez#151', 2),

('mia_lopez', 'mia_lopez$161', 2),

('lucas_gonzalez', 'lucas_gonzalez^171', 2),

('charlotte_perez', 'charlotte_perez!181', 2),

('alexander_wilson', 'alexander_wilson@191', 2),

('amelia_anderson', 'amelia_anderson#202', 2),

('ethan_thomas', 'ethan_thomas*212', 2),

('harper_taylor', 'harper_taylor+222', 2),

('logan_moore', 'logan_moore$232', 2),

('grace_jackson', 'grace_jackson!242', 2);
```

-- Inserting login details for customers with passwords based on username and special characters

INSERT INTO login_details (username, password, role_id)

VALUES

('liam_parker', 'liam_parker@252', 3),

('emma_roberts', 'emma_roberts#262', 3),

('aiden_mitchell', 'aiden_mitchell!272', 3),

('olivia_harris', 'olivia_harris$282', 3),

('noah_thompson', 'noah_thompson^292', 3),

('mia_garcia', 'mia_garcia*302', 3),

('ethan_lee', 'ethan_lee+312', 3),

('sophia_martin', 'sophia_martin@322', 3),

('lucas_taylor', 'lucas_taylor#332', 3),

('charlotte_young', 'charlotte_young!342', 3),

('jackson_moore', 'jackson_moore$352', 3),

('amelia_adams', 'amelia_adams^362', 3),

('oliver_carter', 'oliver_carter*372', 3),

('harper_baker', 'harper_baker+382', 3),

('evan_scott', 'evan_scott@392', 3),

('chloe_nelson', 'chloe_nelson#402', 3),

('mason_hill', 'mason_hill!412', 3),

('ella_green', 'ella_green$422', 3),

('gabriel_king', 'gabriel_king^432', 3),

('grace_wright', 'grace_wright*442', 3);

INSERT INTO drivers (driver_id, first_name, middle_name, last_name, email, phone_number, license_number, vehicle_id, is_active, date_joined)

VALUES

(1, 'John', 'M', 'Doe', 'john.doe@gmail.com', '1234567890', 'ABC12345', 1, TRUE, NOW()),

(2, 'Jane', 'A', 'Smith', 'jane.smith@gmail.com', '2345678901', 'DEF67890', 2, TRUE, NOW()),

(3, 'Michael', 'B', 'Johnson', 'michael.johnson@gmail.com', '3456789012', 'GHI13579', 3, TRUE, NOW()),

(4, 'Emily', 'C', 'Williams', 'emily.williams@gmail.com', '4567890123', 'JKL24680', 4, TRUE, NOW()),

(5, 'David', 'D', 'Brown', 'david.brown@gmail.com', '5678901234', 'MNO35791', 5, TRUE, NOW()),

(6, 'Sophia', 'E', 'Jones', 'sophia.jones@gmail.com', '6789012345', 'PQR46802', 6, TRUE, NOW()),

(7, 'James', 'F', 'Miller', 'james.miller@gmail.com', '7890123456', 'STU57913', 7, TRUE, NOW()),

(8, 'Isabella', 'G', 'Davis', 'isabella.davis@gmail.com', '8901234567', 'VWX68024', 8, TRUE, NOW()),

(9, 'William', 'H', 'Garcia', 'william.garcia@gmail.com', '9012345678', 'YZA79135', 9, TRUE, NOW()),

(10, 'Olivia', 'I', 'Martinez', 'olivia.martinez@gmail.com', '0123456789', 'BCD80246', 10, TRUE, NOW()),

(11, 'Benjamin', 'J', 'Hernandez', 'benjamin.hernandez@gmail.com', '1234567890', 'EFG91357', 11, TRUE, NOW()),

(12, 'Mia', 'K', 'Lopez', 'mia.lopez@gmail.com', '2345678901', 'HIJ02468', 12, TRUE, NOW()),

(13, 'Lucas', 'L', 'Gonzalez', 'lucas.gonzalez@gmail.com', '3456789012', 'KLM13579', 13, TRUE, NOW()),

(14, 'Charlotte', 'M', 'Perez', 'charlotte.perez@gmail.com', '4567890123', 'NOP24680', 14, TRUE, NOW()),

(15, 'Alexander', 'N', 'Wilson', 'alexander.wilson@gmail.com', '5678901234', 'QRS35791', 15, TRUE, NOW()),

(16, 'Amelia', 'O', 'Anderson', 'amelia.anderson@gmail.com', '6789012345', 'TUV46802', 16, TRUE, NOW()),

(17, 'Ethan', 'P', 'Thomas', 'ethan.thomas@gmail.com', '7890123456', 'WXY57913', 17, TRUE, NOW()),

(18, 'Harper', 'Q', 'Taylor', 'harper.taylor@gmail.com', '8901234567', 'ZAB68024', 18, TRUE, NOW()),

(19, 'Logan', 'R', 'Moore', 'logan.moore@gmail.com', '9012345678', 'CDE79135', 19, TRUE, NOW()),

(20, 'Grace', 'S', 'Jackson', 'grace.jackson@gmail.com', '0123456789', 'FGH80246', 20, TRUE, NOW());

INSERT INTO customers (customer_id, first_name, middle_name, last_name, email, phone_number, date_registered, is_active)

VALUES

(21, 'Liam', 'B', 'Harrison', 'liam.harrison@gmail.com', '5678901234', NOW(), TRUE),

(22, 'Emma', 'C', 'Wells', 'emma.wells@gmail.com', '6789012345', NOW(), TRUE),

(23, 'Aiden', 'M', 'Hughes', 'aiden.hughes@gmail.com', '7890123456', NOW(), TRUE),

(24, 'Olivia', 'P', 'Foster', 'olivia.foster@gmail.com', '8901234567', NOW(), TRUE),

(25, 'Noah', 'T', 'Stewart', 'noah.stewart@gmail.com', '9012345678', NOW(), TRUE),

(26, 'Mia', 'R', 'Greenwood', 'mia.greenwood@gmail.com', '0123456789', NOW(), TRUE),

(27, 'Ethan', 'V', 'Pierce', 'ethan.pierce@gmail.com', '1234567890', NOW(), TRUE),

(28, 'Sophia', 'J', 'Carlson', 'sophia.carlson@gmail.com', '2345678901', NOW(), TRUE),

(29, 'Lucas', 'K', 'Dunn', 'lucas.dunn@gmail.com', '3456789012', NOW(), TRUE),

(30, 'Charlotte', 'N', 'Manning', 'charlotte.manning@gmail.com', '4567890123', NOW(), TRUE),

(31, 'Jackson', 'F', 'Wagner', 'jackson.wagner@gmail.com', '5678901234', NOW(), TRUE),

(32, 'Amelia', 'S', 'Burns', 'amelia.burns@gmail.com', '6789012345', NOW(), TRUE),

(33, 'Oliver', 'Q', 'Douglas', 'oliver.douglas@gmail.com', '7890123456', NOW(), TRUE),

(34, 'Harper', 'B', 'Lloyd', 'harper.lloyd@gmail.com', '8901234567', NOW(), TRUE),

(35, 'Evan', 'D', 'Byrne', 'evan.byrne@gmail.com', '9012345678', NOW(), TRUE),

(36, 'Chloe', 'H', 'Fowler', 'chloe.fowler@gmail.com', '0123456789', NOW(), TRUE),

(37, 'Mason', 'T', 'Henderson', 'mason.henderson@gmail.com', '1234567890', NOW(), TRUE),

(38, 'Ella', 'N', 'Riley', 'ella.riley@gmail.com', '2345678901', NOW(), TRUE),

(39, 'Gabriel', 'W', 'Wells', 'gabriel.wells@gmail.com', '3456789012', NOW(), TRUE),

(40, 'Grace', 'M', 'Morgan', 'grace.morgan@gmail.com', '4567890123', NOW(), TRUE);

-- Insert sample ride data

-- Insert sample ride data for the previous year (2023)

INSERT INTO rides (driver_id, customer_id, pickup_location, drop_location, distance, fare, status, date)

VALUES

(1, 21, '123 Main St', '456 Elm St', 10.5, 10.0, 'Completed', '2023-01-01 10:00:00'),

(2, 22, '789 Oak St', '101 Pine St', 8.0, 5.0, 'Completed', '2023-01-05 12:30:00'),

(3, 23, '202 Birch St', '303 Cedar St', 12.3, 25.0, 'Completed', '2023-01-10 15:45:00'),

(4, 24, '404 Maple St', '505 Willow St', 5.0, 12.5, 'Completed', '2023-01-15 14:00:00'),

(5, 25, '606 Aspen St', '707 Redwood St', 6.8, 18.0, 'Completed', '2023-01-20 08:00:00'),

(6, 26, '808 Palm St', '909 Oakwood St', 9.5, 22.0, 'Completed', '2023-01-25 16:30:00'),

(7, 27, '1010 Beach St', '1111 Forest St', 15.0, 10.0, 'Completed', '2023-01-28 13:20:00'),

(8, 28, '1212 River St', '1313 Valley St', 7.2, 7.0, 'Completed', '2023-01-30 09:00:00'),

(9, 29, '1414 Mountain St', '1515 Hill St', 10.0, 70.0, 'Completed', '2023-01-30 18:00:00'),

(10, 30, '1616 Desert St', '1717 Ocean St', 14.0, 8.0, 'Completed', '2023-01-31 11:50:00'),

(1, 21, '123 Main St', '456 Elm St', 10.5, 20.0, 'Completed', '2023-02-01 10:00:00'),

(2, 22, '789 Oak St', '101 Pine St', 8.0, 15.0, 'Completed', '2023-02-05 12:30:00'),

(3, 23, '202 Birch St', '303 Cedar St', 12.3, 25.0, 'Completed', '2023-02-10 15:45:00'),

(4, 24, '404 Maple St', '505 Willow St', 5.0, 12.5, 'Completed', '2023-02-14 14:00:00'),

(5, 25, '606 Aspen St', '707 Redwood St', 6.8, 18.0, 'Completed', '2023-02-18 08:00:00'),

(6, 26, '808 Palm St', '909 Oakwood St', 9.5, 22.0, 'Completed', '2023-02-21 16:30:00'),

(7, 27, '1010 Beach St', '1111 Forest St', 15.0, 30.0, 'Completed', '2023-02-23 13:20:00'),

(8, 28, '1212 River St', '1313 Valley St', 7.2, 17.0, 'Completed', '2023-02-25 09:00:00'),

(9, 29, '1414 Mountain St', '1515 Hill St', 10.0, 20.0, 'Completed', '2023-02-28 18:00:00'),

(10, 30, '1616 Desert St', '1717 Ocean St', 14.0, 28.0, 'Completed', '2023-02-28 11:50:00'),

(1, 21, '123 Main St', '456 Elm St', 10.5, 20.0, 'Completed', '2023-03-01 10:00:00'),

(2, 22, '789 Oak St', '101 Pine St', 8.0, 15.0, 'Completed', '2023-03-05 12:30:00'),

(3, 23, '202 Birch St', '303 Cedar St', 12.3, 25.0, 'Completed', '2023-03-10 15:45:00'),

(4, 24, '404 Maple St', '505 Willow St', 5.0, 12.5, 'Completed', '2023-03-15 14:00:00'),

(5, 25, '606 Aspen St', '707 Redwood St', 6.8, 18.0, 'Completed', '2023-03-20 08:00:00'),

(6, 26, '808 Palm St', '909 Oakwood St', 9.5, 22.0, 'Completed', '2023-03-25 16:30:00'),

(7, 27, '1010 Beach St', '1111 Forest St', 15.0, 30.0, 'Completed', '2023-03-28 13:20:00'),

(8, 28, '1212 River St', '1313 Valley St', 7.2, 17.0, 'Completed', '2023-03-30 09:00:00'),

(9, 29, '1414 Mountain St', '1515 Hill St', 10.0, 20.0, 'Completed', '2023-03-30 18:00:00'),

(10, 30, '1616 Desert St', '1717 Ocean St', 14.0, 28.0, 'Completed', '2023-03-31 11:50:00'),

(1, 21, '123 Main St', '456 Elm St', 10.5, 20.0, 'Completed', '2024-01-01 10:00:00'),

(2, 22, '789 Oak St', '101 Pine St', 8.0, 15.0, 'Completed', '2024-01-05 12:30:00'),

(3, 23, '202 Birch St', '303 Cedar St', 12.3, 25.0, 'Completed', '2024-01-10 15:45:00'),

(4, 24, '404 Maple St', '505 Willow St', 5.0, 12.5, 'Completed', '2024-01-15 14:00:00'),

(5, 25, '606 Aspen St', '707 Redwood St', 6.8, 118.0, 'Completed', '2024-01-20 08:00:00'),

(6, 26, '808 Palm St', '909 Oakwood St', 9.5, 122.0, 'Completed', '2024-01-25 16:30:00'),

(7, 27, '1010 Beach St', '1111 Forest St', 15.0, 30.0, 'Completed', '2024-01-28 13:20:00'),

(8, 28, '1212 River St', '1313 Valley St', 7.2, 17.0, 'Completed', '2024-01-30 09:00:00'),

(9, 29, '1414 Mountain St', '1515 Hill St', 10.0, 20.0, 'Completed', '2024-01-30 18:00:00'),

(10, 30, '1616 Desert St', '1717 Ocean St', 14.0, 28.0, 'Completed', '2024-01-31 11:50:00'),

(1, 21, '123 Main St', '456 Elm St', 10.5, 20.0, 'Completed', '2024-02-01 10:00:00'),

(2, 22, '789 Oak St', '101 Pine St', 8.0, 15.0, 'Completed', '2024-02-05 12:30:00'),

(3, 23, '202 Birch St', '303 Cedar St', 12.3, 25.0, 'Completed', '2024-02-10 15:45:00'),

(4, 24, '404 Maple St', '505 Willow St', 5.0, 12.5, 'Completed', '2024-02-14 14:00:00'),

(5, 25, '606 Aspen St', '707 Redwood St', 6.8, 18.0, 'Completed', '2024-02-18 08:00:00'),

(6, 26, '808 Palm St', '909 Oakwood St', 9.5, 22.0, 'Completed', '2024-02-21 16:30:00'),

(7, 27, '1010 Beach St', '1111 Forest St', 15.0, 30.0, 'Completed', '2024-02-23 13:20:00'),

(8, 28, '1212 River St', '1313 Valley St', 7.2, 17.0, 'Completed', '2024-02-25 09:00:00'),

(9, 29, '1414 Mountain St', '1515 Hill St', 10.0, 20.0, 'Completed', '2024-02-28 18:00:00'),

(10, 30, '1616 Desert St', '1717 Ocean St', 14.0, 28.0, 'Completed', '2024-02-28 11:50:00'),

(1, 21, '123 Main St', '456 Elm St', 10.5, 20.0, 'Completed', '2024-03-01 10:00:00'),

(2, 22, '789 Oak St', '101 Pine St', 8.0, 15.0, 'Completed', '2024-03-05 12:30:00'),

(3, 23, '202 Birch St', '303 Cedar St', 12.3, 25.0, 'Completed', '2024-03-10 15:45:00'),

(4, 24, '404 Maple St', '505 Willow St', 5.0, 12.5, 'Completed', '2024-03-15 14:00:00'),

(5, 25, '606 Aspen St', '707 Redwood St', 6.8, 18.0, 'Completed', '2024-03-20 08:00:00'),

(6, 26, '808 Palm St', '909 Oakwood St', 9.5, 22.0, 'Completed', '2024-03-25 16:30:00'),

(7, 27, '1010 Beach St', '1111 Forest St', 15.0, 30.0, 'Completed', '2024-03-28 13:20:00'),

(8, 28, '1212 River St', '1313 Valley St', 7.2, 17.0, 'Completed', '2024-03-30 09:00:00'),

(9, 29, '1414 Mountain St', '1515 Hill St', 10.0, 20.0, 'Completed', '2024-03-30 18:00:00'),

(10, 30, '1616 Desert St', '1717 Ocean St', 14.0, 28.0, 'Completed', '2024-03-31 11:50:00');

INSERT INTO ride_geolocations (ride_id, latitude, longitude)

VALUES

(1, 40.7128, -74.0060),

(1, 40.7306, -73.9352),

(1, 40.7550, -73.9830),

(1, 40.7580, -73.9855),

(1, 40.7790, -73.9800),

(1, 40.7128, -74.0060),

(2, 34.0522, -118.2437),

(2, 34.0632, -118.2500),

(2, 34.0750, -118.2700),

(2, 34.0812, -118.2900),

(2, 34.0980, -118.3000),

(2, 34.0522, -118.2437),

(3, 41.8781, -87.6298),

(3, 41.8800, -87.6200),

(3, 41.8900, -87.6100),

(3, 41.9000, -87.6000),

(3, 41.9050, -87.5900),

(3, 41.8781, -87.6298),

(4, 29.7604, -95.3698),

(4, 29.7700, -95.3600),

(4, 29.7800, -95.3500),

(4, 29.7900, -95.3400),

(4, 29.8000, -95.3300),

(4, 29.7604, -95.3698),

(5, 51.5074, -0.1278),

(5, 51.5150, -0.1400),

(5, 51.5250, -0.1450),

(5, 51.5300, -0.1500),

(5, 51.5400, -0.1550),

(5, 51.5074, -0.1278),

(6, 48.8566, 2.3522),

(6, 48.8600, 2.3500),

(6, 48.8700, 2.3400),

(6, 48.8800, 2.3300),

(6, 48.8900, 2.3200),

(6, 48.8566, 2.3522),

(7, 52.5200, 13.4050),

(7, 52.5300, 13.4000),

(7, 52.5400, 13.3950),

(7, 52.5500, 13.3900),

(7, 52.5600, 13.3850),

(7, 52.5200, 13.4050),

(8, 37.7749, -122.4194),

(8, 37.7800, -122.4300),

(8, 37.7900, -122.4400),

(8, 37.8000, -122.4500),

(8, 37.8100, -122.4600),

(8, 37.7749, -122.4194),

(9, 34.0522, -118.2437),

(9, 34.0600, -118.2500),

(9, 34.0700, -118.2600),

(9, 34.0800, -118.2700),

(9, 34.0900, -118.2800),

(9, 34.0522, -118.2437),

(10, 40.7306, -73.9352),

(10, 40.7400, -73.9300),

(10, 40.7500, -73.9200),

(10, 40.7600, -73.9100),

(10, 40.7700, -73.9000),

(10, 40.7306, -73.9352);

```
INSERT INTO payment_details (ride_id, payment_method, amount, transaction_status, transaction_id)
VALUES
(1, 'CreditCard', 20.0, 'Completed', 'TXN00123'),
(2, 'Cash', 15.0, 'Completed', 'TXN00124'),
(3, 'DebitCard', 25.0, 'Completed', 'TXN00125'),
(4, 'CreditCard', 12.5, 'Completed', 'TXN00126'),
(5, 'Wallet', 18.0, 'Failed', 'TXN00127'),
(6, 'CreditCard', 22.0, 'Completed', 'TXN00128'),
(7, 'DebitCard', 30.0, 'Completed', 'TXN00129'),
(8, 'Cash', 17.0, 'Completed', 'TXN00130'),
(9, 'Wallet', 20.0, 'Completed', 'TXN00131'),
(10, 'CreditCard', 28.0, 'Completed', 'TXN00132');
```

```
INSERT INTO rewards (user_id, points_earned, category, description)
VALUES
(1, 100, 'Ride', 'Reward for completing 10 rides'),
(2, 150, 'Ride', 'Reward for completing 15 rides'),
(3, 120, 'Referral', 'Reward for referring a new customer'),
(4, 80, 'Promotion', 'Reward for using a promotional code'),
(5, 200, 'Ride', 'Reward for completing 20 rides'),
```

(6, 250, 'Referral', 'Reward for referring multiple customers'),

(7, 50, 'Ride', 'Reward for completing 5 rides'),

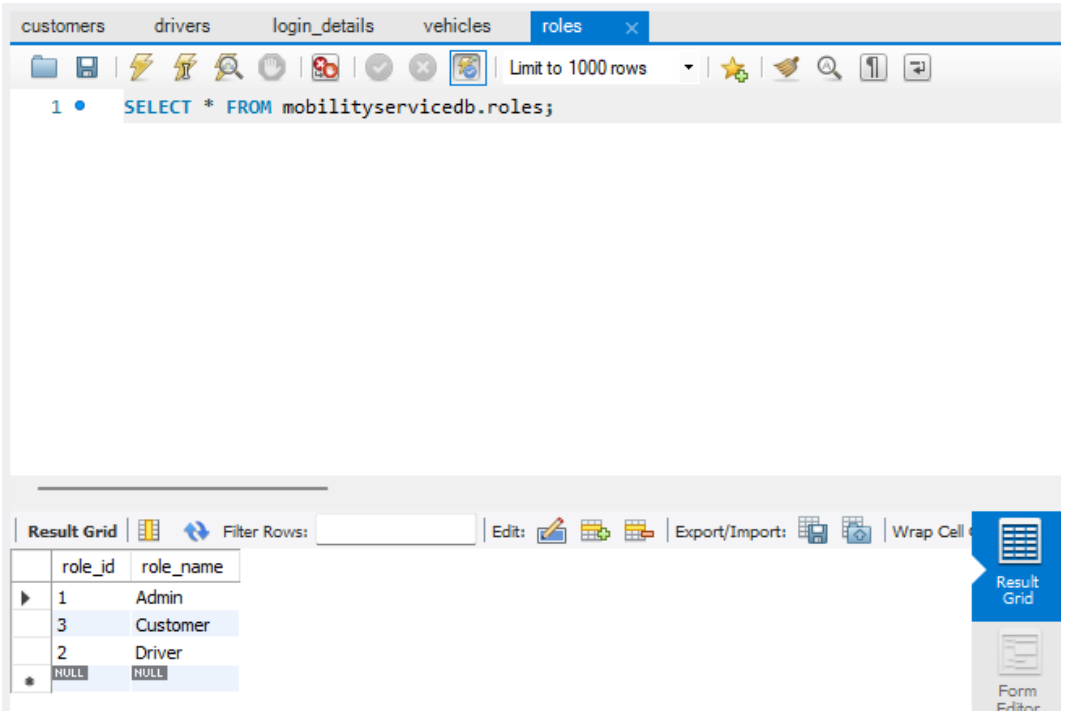
(8, 90, 'Promotion', 'Reward for participating in a promotion'),

(9, 170, 'Referral', 'Reward for referring a customer'),

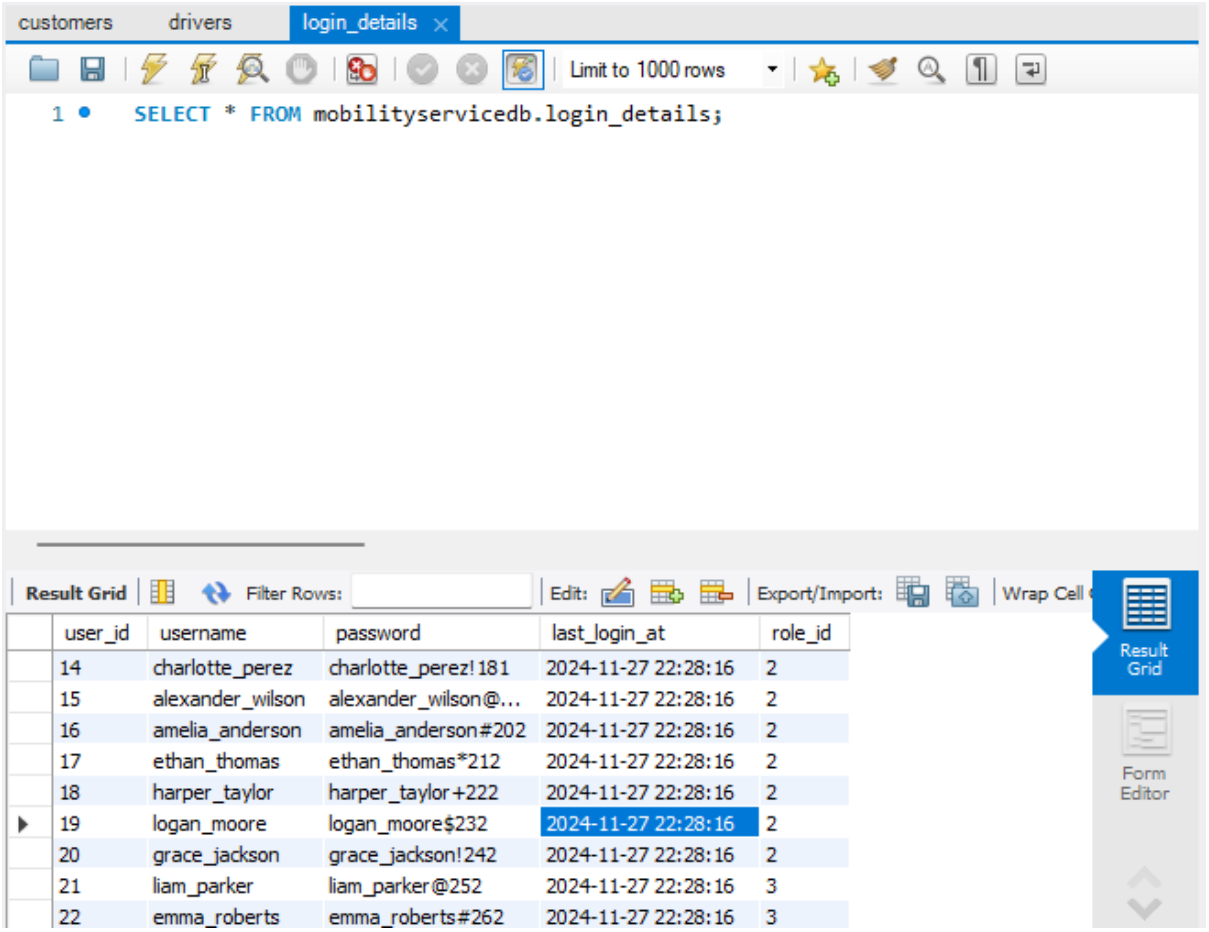
(10, 60, 'Ride', 'Reward for completing 6 rides');

Table Output Screenshots:

Roles Table:



Login Details table



user_id	username	password	last_login_at	role_id
14	charlotte_perez	charlotte_perez!181	2024-11-27 22:28:16	2
15	alexander_wilson	alexander_wilson@...	2024-11-27 22:28:16	2
16	amelia_anderson	amelia_anderson#202	2024-11-27 22:28:16	2
17	ethan_thomas	ethan_thomas*212	2024-11-27 22:28:16	2
18	harper_taylor	harper_taylor+222	2024-11-27 22:28:16	2
19	logan_moore	logan_moore$232	2024-11-27 22:28:16	2
20	grace_jackson	grace_jackson!242	2024-11-27 22:28:16	2
21	liam_parker	liam_parker@252	2024-11-27 22:28:16	3
22	emma_roberts	emma_roberts#262	2024-11-27 22:28:16	3

Customers Table:

customers

Limit to 1000 rows

```
1 • SELECT * FROM mobilityservicedb.customers;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

customer_id	first_name	middle_name	last_name	email	phone_number	dat
21	Liam	B	Harrison	liam.harrison@gmail.com	5678901234	202
22	Emma	C	Wells	emma.wells@gmail.com	6789012345	202
23	Aiden	M	Hughes	aiden.hughes@gmail.com	7890123456	202
24	Olivia	P	Foster	olivia.foster@gmail.com	8901234567	202
25	Noah	T	Stewart	noah.stewart@gmail.com	9012345678	202
26	Mia	R	Greenwood	mia.greenwood@gmail.com	0123456789	202
27	Ethan	V	Pierce	ethan.pierce@gmail.com	1234567890	202
28	Sophia	J	Carlson	sophia.carlson@gmail.com	2345678901	202

Result Grid | Form Editor

Drivers Table:

customers | drivers

Limit to 1000 rows

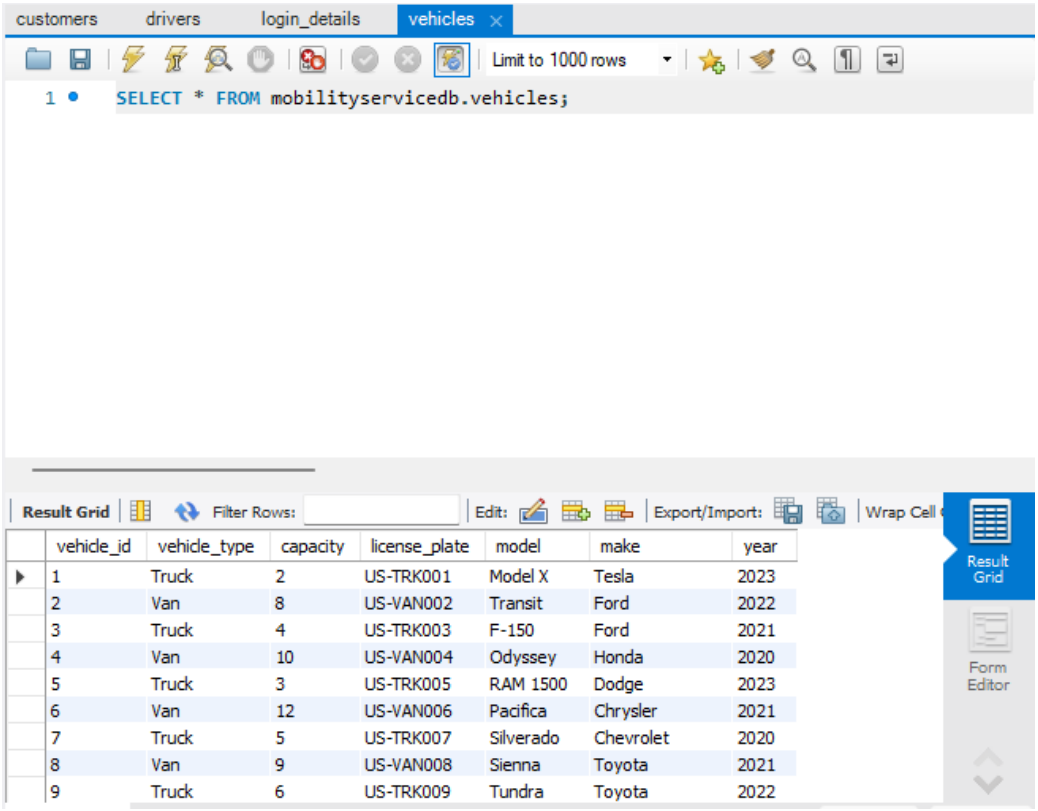
```
1 • SELECT * FROM mobilityservicedb.drivers;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

driver_id	first_name	middle_name	last_name	email	phone_number	license_nu
1	John	M	Doe	john.doe@gmail.com	1234567890	ABC12345
2	Jane	A	Smith	jane.smith@gmail.com	2345678901	DEF67890
3	Michael	B	Johnson	michael.johnson@gmail.com	3456789012	GHI13579
4	Emily	C	Williams	emily.williams@gmail.com	4567890123	JKL24680
5	David	D	Brown	david.brown@gmail.com	5678901234	MNO35791
6	Sophia	E	Jones	sophia.jones@gmail.com	6789012345	PQR46802
7	James	F	Miller	james.miller@gmail.com	7890123456	STU57913
8	Isabella	G	Davis	isabella.davis@gmail.com	8901234567	VWX68024

Result Grid | Form Editor

Rides Table:



vehicle_id	vehicle_type	capacity	license_plate	model	make	year
1	Truck	2	US-TRK001	Model X	Tesla	2023
2	Van	8	US-VAN002	Transit	Ford	2022
3	Truck	4	US-TRK003	F-150	Ford	2021
4	Van	10	US-VAN004	Odyssey	Honda	2020
5	Truck	3	US-TRK005	RAM 1500	Dodge	2023
6	Van	12	US-VAN006	Pacifica	Chrysler	2021
7	Truck	5	US-TRK007	Silverado	Chevrolet	2020
8	Van	9	US-VAN008	Sienna	Toyota	2021
9	Truck	6	US-TRK009	Tundra	Toyota	2022

Ride Geo locations Table:

customers drivers login_details vehicles roles rides ride_geolocations

Limit to 1000 rows

```
1 • SELECT * FROM mobilityservicedb.ride_geolocations;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

route_id	ride_id	latitude	longitude	timestamp
1	1	40.7128	-74.006	2024-11-27 22:28:17
2	1	40.7306	-73.9352	2024-11-27 22:28:17
3	1	40.755	-73.983	2024-11-27 22:28:17
4	1	40.758	-73.9855	2024-11-27 22:28:17
5	1	40.779	-73.98	2024-11-27 22:28:17
6	1	40.7128	-74.006	2024-11-27 22:28:17
7	2	34.0522	-118.244	2024-11-27 22:28:17
8	2	34.0632	-118.25	2024-11-27 22:28:17
9	2	34.075	-118.27	2024-11-27 22:28:17

Result Grid | Form Editor

Payment Details Table:

login_details vehicles roles rides ride_geolocations rewards payment_deta

Limit to 1000 rows

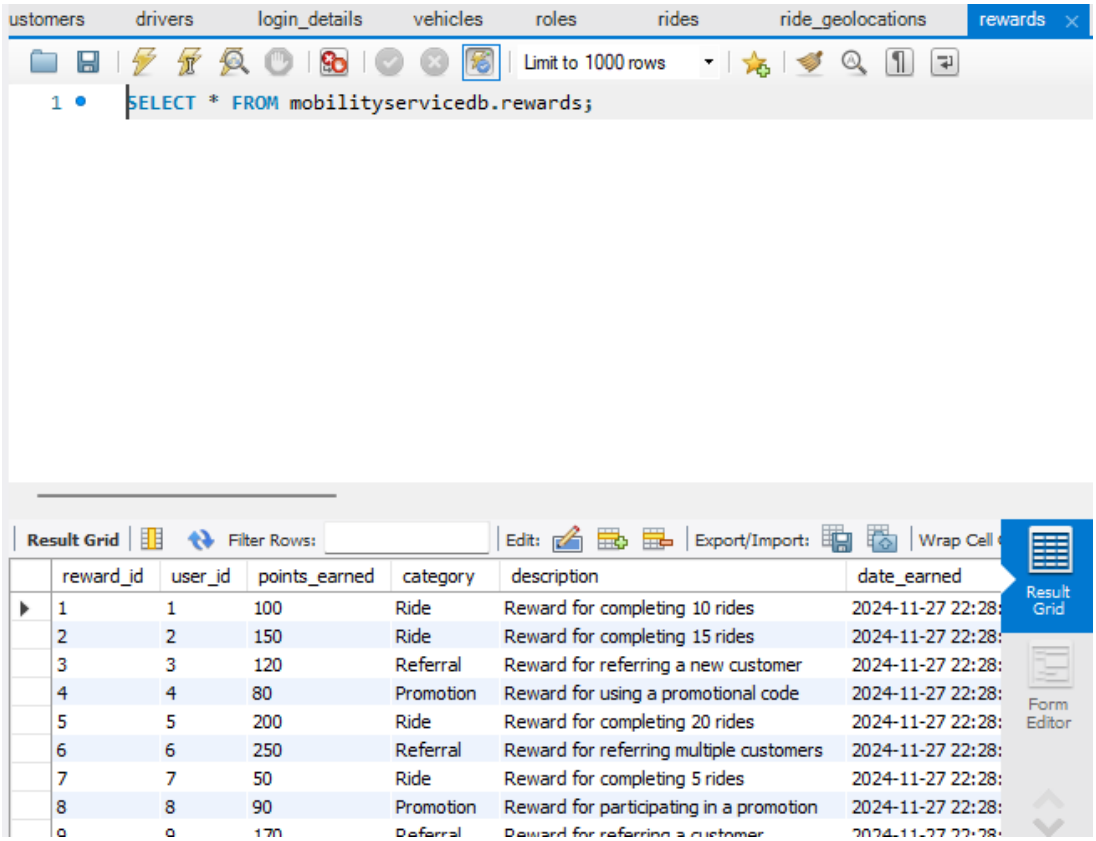
```
1 • SELECT * FROM mobilityservicedb.payment_details;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

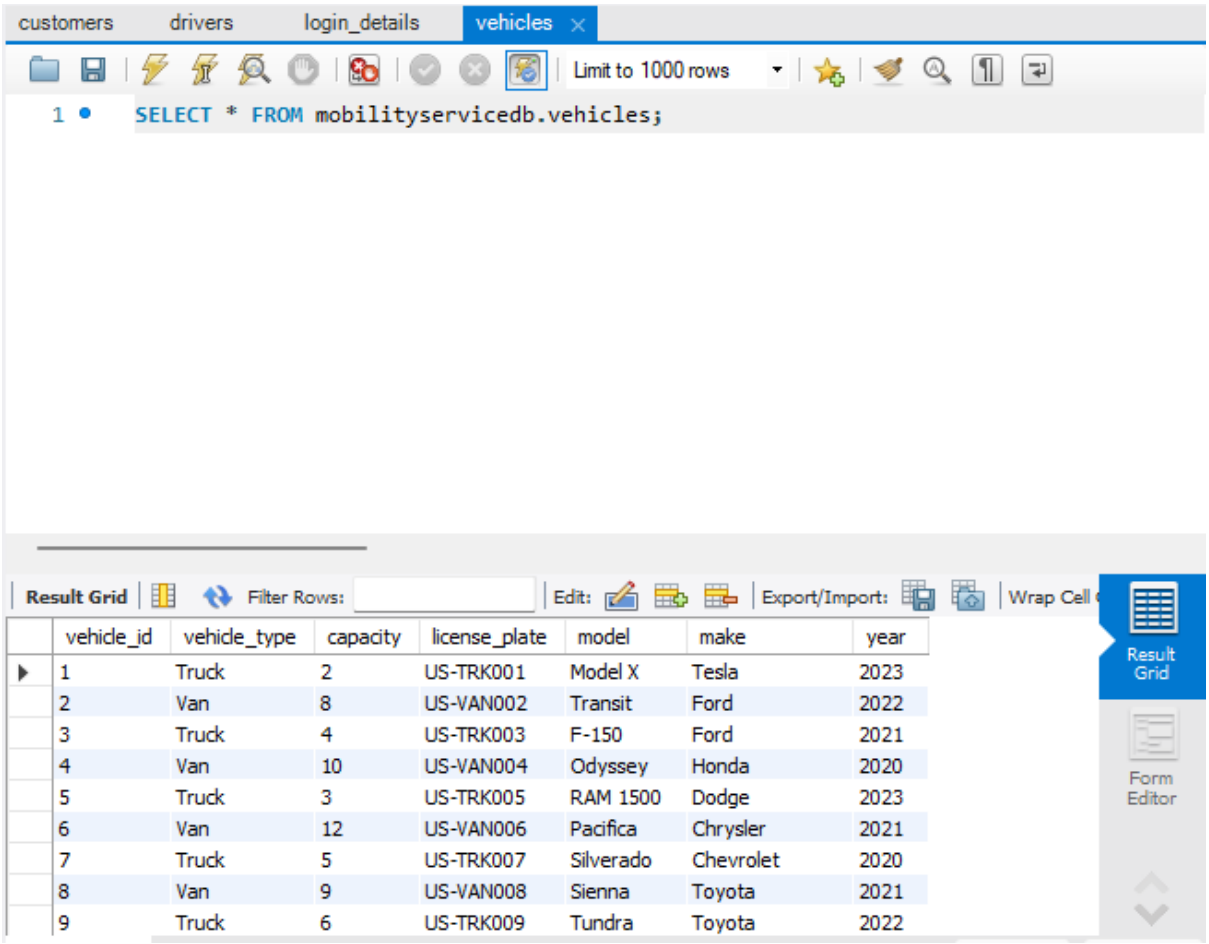
payment_id	ride_id	payment_method	amount	transaction_status	transaction_id	payment_date
1	1	CreditCard	20	Completed	TXN00123	2024-11-27 22:28:1
2	2	Cash	15	Completed	TXN00124	2024-11-27 22:28:1
3	3	DebitCard	25	Completed	TXN00125	2024-11-27 22:28:1
4	4	CreditCard	12.5	Completed	TXN00126	2024-11-27 22:28:1
5	5	Wallet	18	Failed	TXN00127	2024-11-27 22:28:1
6	6	CreditCard	22	Completed	TXN00128	2024-11-27 22:28:1
7	7	DebitCard	30	Completed	TXN00129	2024-11-27 22:28:1
8	8	Cash	17	Completed	TXN00130	2024-11-27 22:28:1
9	9	Wallet	20	Completed	TXN00131	2024-11-27 22:28:1

Result Grid | Form Editor

Rewards table:



Vehicle Table:



vehicle_id	vehicle_type	capacity	license_plate	model	make	year
1	Truck	2	US-TRK001	Model X	Tesla	2023
2	Van	8	US-VAN002	Transit	Ford	2022
3	Truck	4	US-TRK003	F-150	Ford	2021
4	Van	10	US-VAN004	Odyssey	Honda	2020
5	Truck	3	US-TRK005	RAM 1500	Dodge	2023
6	Van	12	US-VAN006	Pacifica	Chrysler	2021
7	Truck	5	US-TRK007	Silverado	Chevrolet	2020
8	Van	9	US-VAN008	Sienna	Toyota	2021
9	Truck	6	US-TRK009	Tundra	Toyota	2022

Analytics Requirement

How many drivers are registered?

```
use MobilityServiceDB;


SELECT COUNT(*)  AS drivers_count FROM drivers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

drivers_count
20

2. Number of active vs inactive drivers this month?

```
-- Number of active vs inactive ( this month)?
SELECT SUM(IF(is_active, 1, 0)) AS active_drivers, SUM(IF(NOT is_active, 1, 0)) AS inactive_drivers
FROM drivers
WHERE MONTH(date_joined) = MONTH(CURRENT_DATE);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

active_drivers	inactive_drivers
20	0

How many customers have you registered?

```
SELECT COUNT(*) AS customers_count FROM customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customers_count
20

Who is the driver with the most number of rides?

```sql
-- Who is the driver with the most number of rides?
SELECT  rides.driver_id,CONCAT(drivers.first_name, ' ', drivers.last_name) AS Name,
    COUNT(*) AS ride_count
FROM rides
JOIN drivers ON rides.driver_id = drivers.driver_id
GROUP BY rides.driver_id
ORDER BY ride_count DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

driver_id	Name	ride_count
1	John Doe	6

Who is the passenger with the most number of rides?

```sql
-- . Who is the passenger with the most number of rides?
SELECT  rides.customer_id,
        CONCAT(customers.first_name, ' ', customers.last_name) AS Name,
        COUNT(*) AS ride_count
FROM rides
JOIN customers ON rides.customer_id = customers.customer_id
GROUP BY rides.customer_id
ORDER BY ride_count DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

customer_id	Name	ride_count
21	Liam Harrison	6

Top 5 drivers by revenue by month (January, February, March)?

```sql
WITH ranked_drivers AS (
    SELECT d.driver_id, CONCAT(d.first_name, ' ', d.last_name) AS driver_name,
        EXTRACT(MONTH FROM r.date) AS month, SUM(r.fare) AS total_revenue,
        ROW_NUMBER() OVER (PARTITION BY EXTRACT(MONTH FROM r.date) ORDER BY SUM(r.fare) DESC) AS revenue_rank
    FROM rides r
    JOIN drivers d ON r.driver_id = d.driver_id
    WHERE EXTRACT(MONTH FROM r.date) IN (1, 2, 3)
      AND r.status = 'Completed'
    GROUP BY d.driver_id, EXTRACT(MONTH FROM r.date)
)
SELECT driver_id, driver_name, month, total_revenue
FROM ranked_drivers
WHERE revenue_rank <= 5
ORDER BY month, revenue_rank;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

driver_id	driver_name	month	total_revenue
6	Sophia Jones	1	144
5	David Brown	1	136
9	William Garcia	1	90
3	Michael Johnson	1	50
7	James Miller	1	40
7	James Miller	2	60
10	Olivia Martinez	2	56

Top 5 customers by revenue this year?

```sql
-- Top 5 customers by revenue this year?
SELECT c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    SUM(fare) AS total_revenue
FROM rides r
JOIN customers c ON r.customer_id = c.customer_id
WHERE EXTRACT(YEAR FROM r.date) = EXTRACT(YEAR FROM CURRENT_DATE)
GROUP BY c.customer_id, c.first_name, c.last_name
ORDER BY total_revenue DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

customer_id	customer_name	total_revenue
26	Mia Greenwood	166
25	Noah Stewart	154
27	Ethan Pierce	90
30	Charlotte Manning	84
23	Aiden Hughes	75

Has our revenue increased last month compared to the same month last year?

```sql
WITH revenue_data AS (
    SELECT EXTRACT(YEAR FROM date) AS year, EXTRACT(MONTH FROM date) AS month, SUM(fare) AS total_revenue
    FROM rides WHERE status = 'Completed' GROUP BY year, month)
SELECT
    current_month.year AS year,current_month.month AS month,
    prev_year.total_revenue AS last_year_revenue,current_month.total_revenue AS current_revenue,
    CASE
        WHEN current_month.total_revenue > prev_year.total_revenue THEN 'Increased'
        WHEN current_month.total_revenue < prev_year.total_revenue THEN 'Decreased'
        ELSE 'No Change'
    END AS revenue_change
FROM revenue_data current_month
LEFT JOIN revenue_data prev_year
ON  current_month.year = prev_year.year + 1 AND
    current_month.month = prev_year.month
WHERE current_month.year = 2024 AND current_month.month = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

year	month	last_year_revenue	current_revenue	revenue_change
2024	1	187.5	407.5	Increased

Customer Questions:

What is the most expensive ride among my rides?

```sql

    -- What is the most expensive ride among my rides?

    SELECT ride_id,pickup_location,drop_location,fare,date
    FROM rides
    WHERE customer_id = 23
    ORDER BY fare DESC
    LIMIT 1;

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

ride_id	pickup_location	drop_location	fare	date
3	202 Birch St	303 Cedar St	25	2023-01-10 15:45:00
NULL	NULL	NULL	NULL	NULL

How many rides did I take with your business?

```sql
-- How many rides did I take with your business?

SELECT COUNT(*) AS total_rides
FROM rides
WHERE customer_id = 23;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_rides
6

How much money did I spend with your business year-to-date?

```sql
-- How much money did I spend with your business year-to-date?
SELECT SUM(fare) AS total_spent
FROM rides
WHERE customer_id = 23 AND status = 'Completed' AND EXTRACT(YEAR FROM date) = EXTRACT(YEAR FROM CURRENT_DATE);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_spent
75

Operations questions

what is the size of project database?

```sql
SELECT table_schema AS "Database", SUM(data_length + index_length) / 1024 / 1024 AS "Size in MB"
FROM information_schema.tables
WHERE  table_schema = 'mobilityservicedb'
GROUP BY table_schema;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Database	Size in MB
mobilityservicedb	0.53125000

what is the size of each table?

```
128
129 •  SELECT table_name AS "Table", ROUND(((data_length + index_length) / 1024 / 1024), 2) AS "Size in MB"
130    FROM information_schema.tables
131    WHERE table_schema = 'mobilityservicedb'
132    ORDER BY (data_length + index_length) DESC;
133
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Size in MB
drivers	0.09
payment_details	0.08
login_details	0.06
rides	0.06
vehicles	0.06
customers	0.05
rewards	0.05
roles	0.05
ride_geolocations	0.03

Impact of Index Creation on Query Performance: Before and After Analysis

Before Creating the Index:

- The query SELECT * FROM mobilityservicedb.payment_details ORDER BY payment_method LIMIT 0, 1000 was executed without an index on the payment_method column.
- Since there was no index, the database had to perform a full table scan to retrieve and order the data based on the payment_method column. This process is slower, especially as the table grows larger.

After Creating the Index:

- An index was created on the payment_method column using the command:

  CREATE INDEX idx_payment_method ON payment_details (payment_method);

- The database now has an optimized structure to quickly locate rows based on the payment_method column. Instead of scanning the entire table, the database can efficiently access the indexed data.

Execution Time Comparison:

- Before index creation: The query took 0.00053950 seconds.
- After index creation: The query took 0.00047650 seconds.

The execution time improved a little after the index was created, meaning the index helped the database find and sort the data faster. However, since there are only 10 records in the table, the time difference is small. The impact of the index would be much bigger if the table had more records, making data retrieval much faster as the dataset grows.