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

**Table of Contents**

1. **Introduction**
2. **Application Design Requirements**
   * Login Screen
   * Driver Register Page
   * Customer Register Page
   * Customer Profile Screen
   * Book Ride Page
   * Driver Accept Ride Page
   * Payment Screen
   * Customer Previous Rides
3. **Database Normalization**

* Atomicity
* Elimination of Repeating Groups
* Unique Rows
* Elimination of Partial Dependencies
* Elimination of Transitive Dependencies
  + Referential Integrity

1. **Database Schema Overview**

* Roles Table
* Login Details Table
* Vehicles Table
* Drivers Table
* Customers Table
* Rides Table
* Ride Geolocations Table
* Payment Details Table
* Rewards Table

1. **SQL Script for Creating the MobilityServiceDB**
2. **Analytics Requirements**

**Introduction**

The Mobility Service System is a platform designed to make ride-booking and transportation services simple and efficient for both customers and drivers. Customers can create accounts, book rides, and check their ride history, while drivers can register, manage their profiles, and accept ride requests. The system supports different payment options like cash, credit cards, debit cards, and wallets, making it convenient for users. It is built with a strong database to store and manage information about users, drivers, vehicles, rides, and payments efficiently. The system also includes geolocation tracking to provide accurate ride details and ensure smooth operations. Customers and drivers can earn rewards through rides, referrals, and promotions, which adds extra benefits. With a focus on ease of use and reliability, this platform helps both users and drivers enjoy a hassle-free experience. It is designed to handle large amounts of data and can be updated or scaled as needed, making it a long-term solution for growing transportation needs.

**Application design Requirements:**

**Mock up screens**

1. **Login Screen**

This is the login page where the user will enter their username and password. If they don't have an account, they can click on "Register" to create one.

**A screenshot of a login form

Description automatically generated**

**Driver Register Page:**

To register as a driver, the user needs to select the "Register as Driver" radio button and their personal information, including username, password, first name, last name, email, phone number, and license number. Additionally, they need to enter their vehicle details in the "Vehicle Details" section, such as the vehicle type (e.g., car, motorcycle, van), license plate number, model, and capacity.

**A screenshot of a computer

Description automatically generated**

**Customer Register Page:**

To register as a customer, the user needs to select the "Register as Customer" radio button and provide their details, including username, password, first name, last name, email, and phone number. After entering the required information, they can click on "Submit" to save the details.

**A screenshot of a login form

Description automatically generated**

**Customer Profile Screen:**

On the customer profile screen, the user can view their details, including username, first name, last name, email, and phone number. They can make the changes, and then clicking "Update" to save them. The username must be unique; if the user tries to modify it to one that is already in use, an error message will be displayed.

**A screenshot of a computer

Description automatically generated**

**Book Ride Page:**

On the "Book Ride" page, the customer selects the pickup location, drop location, and vehicle type (e.g., motorcycle, cab, van). Based on the selected vehicle type, the maximum capacity and fare will be displayed. The customer can then click on "Ride" to confirm the booking. Upon confirmation, a new record will be created in the database with the status set as "Booked."

**A screenshot of a computer

Description automatically generated**

**Driver Accept Ride Page:**

After a customer books a ride, the driver can view all requested rides on the "Driver Accept Ride" page. The page displays details such as the pickup location, drop location, distance, fare, and vehicle type. The driver can review the list of available rides and accept one by clicking on the "Accept" button for a specific ride. Once the driver accepts a ride, the ride's status in the database is updated to "Accepted," and the ride is assigned to the driver. Once the driver begins the ride (picks up the customer), the status updates to **"InTransit."** When the ride is completed and the customer is dropped off at the destination, the status changes to **"Completed".** Upon completion, the customer is redirected to the **Payment Screen**,

**Screens screenshot of a phone

Description automatically generated**

**Payment Screen**

On the Payment Page, the customer can view the fare for the completed ride. They must select a payment option from the available methods: Cash, Credit Card, Debit Card, or Wallet. After selecting the desired payment method, the customer clicks on the "Pay" button to proceed. Upon successful payment, the payment details, including the selected payment method, fare, and transaction status and number, are recorded in the database.

A screenshot of a computer

Description automatically generated

**Customer Previous Rides:**

**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.

**Entity Relation Diagram (ERD):**

**A diagram of a diagram

Description automatically generated**

**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 ('Booked','Accepted','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\_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:**

**A screenshot of a computer

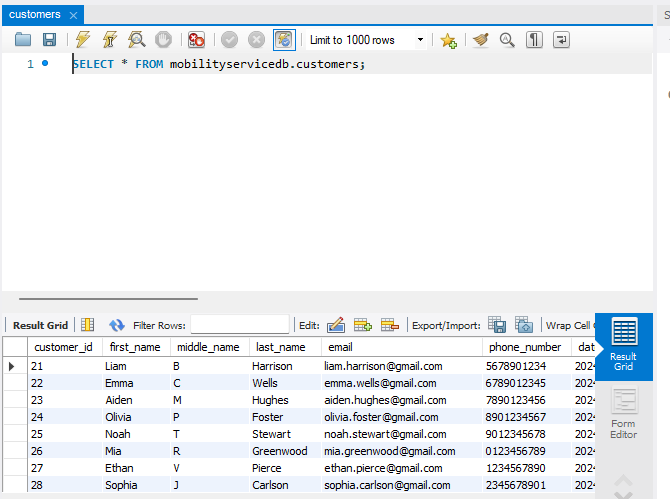
Description automatically generated**

**Login Details table**

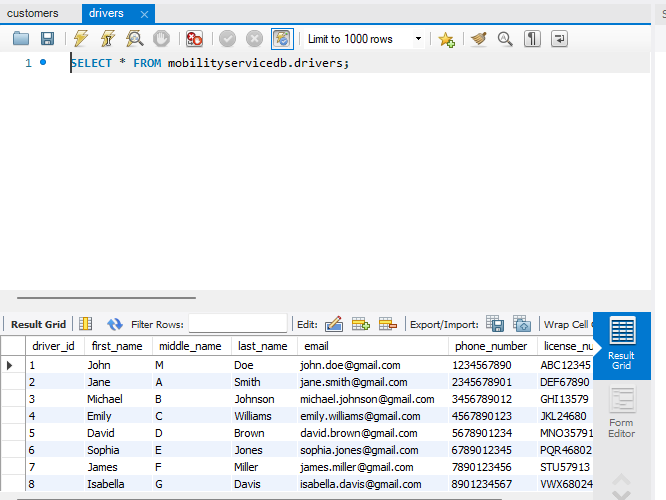
**A screenshot of a computer

Description automatically generated**

**Customers Table:**

****

**Drivers Table:**

****

**Rides Table:**

**A screenshot of a computer

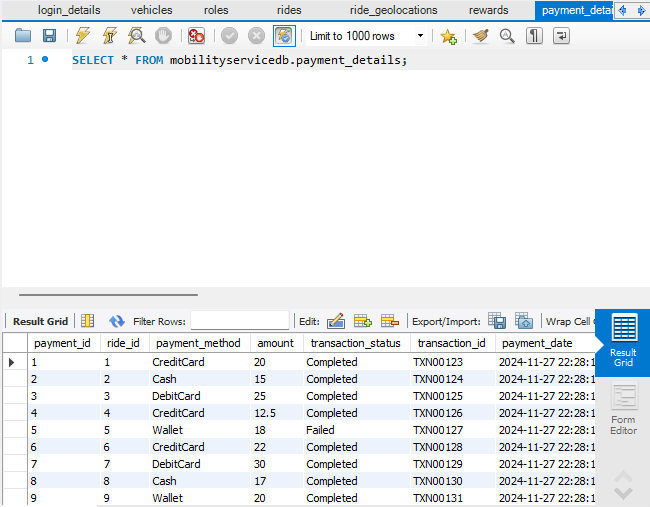
Description automatically generated**

**Ride Geo locations Table:**

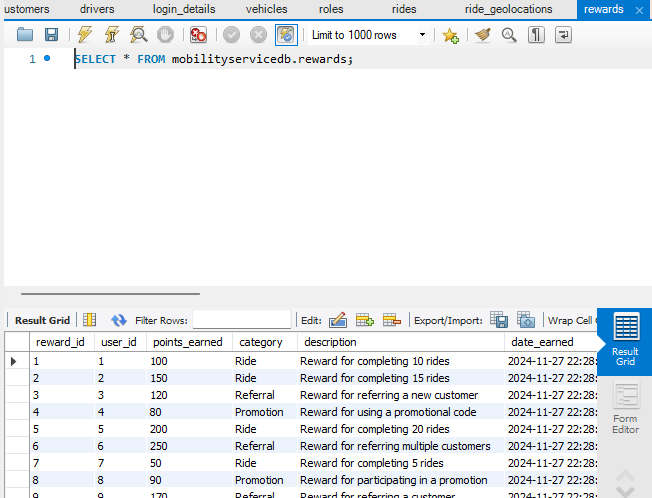
**A screenshot of a computer

Description automatically generated**

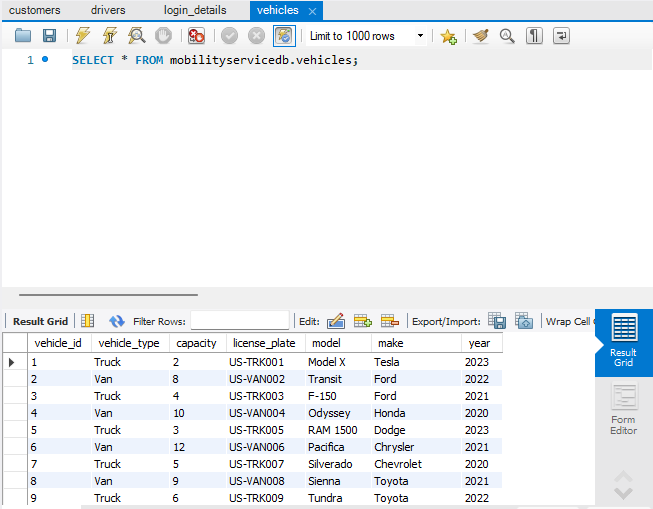
**Payment Details Table:**

****

**Rewards table:**

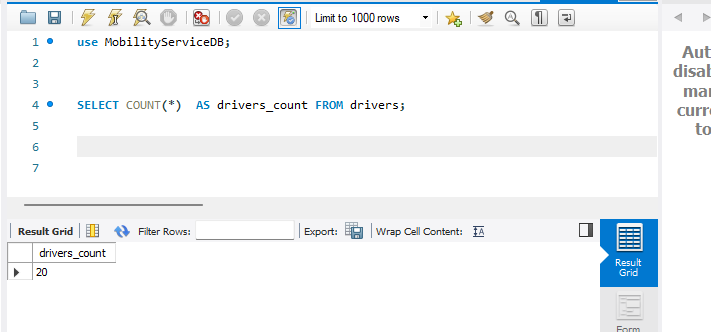
****

**Vehicle Table:**

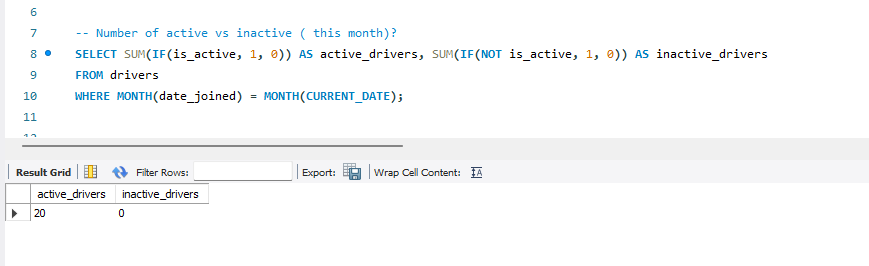


**Analytics Requirement**

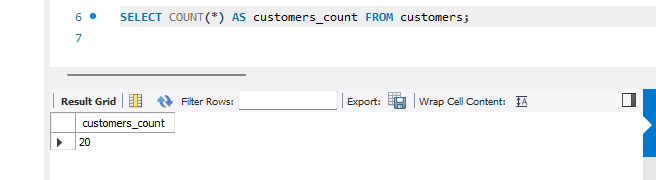
**How many drivers are registered?**

****

**2. Number of active vs inactive drivers this month?**

****

**How many customers have you registered?**

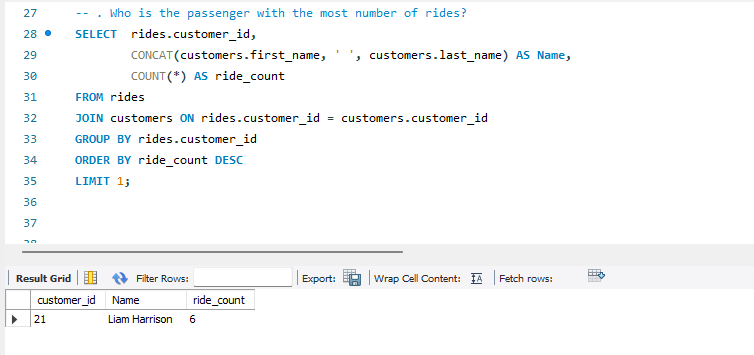


**Who is the driver with the most number of rides?**

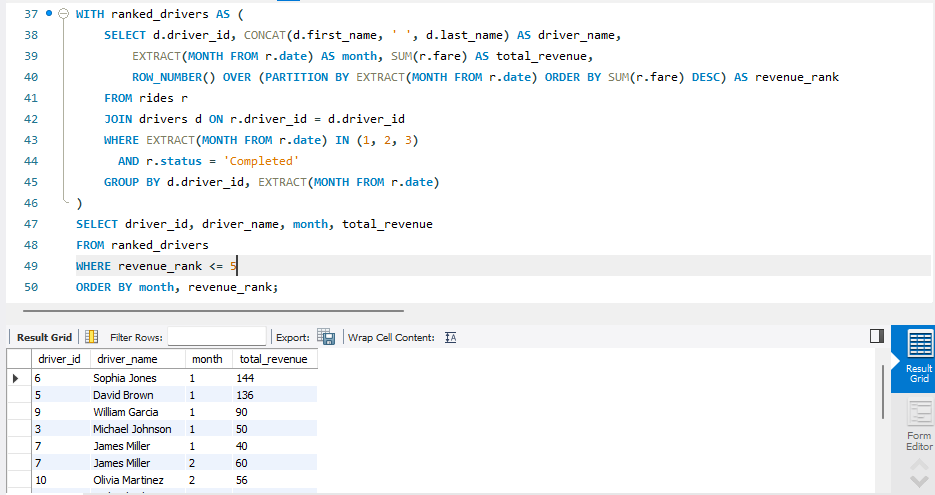
**A screenshot of a computer

Description automatically generated**

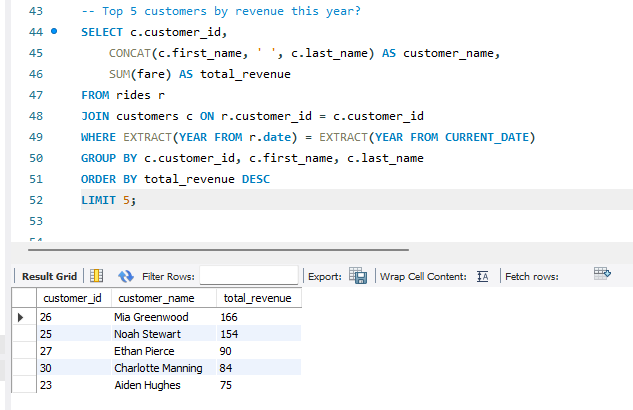
**Who is the passenger with the most number of rides?**

****

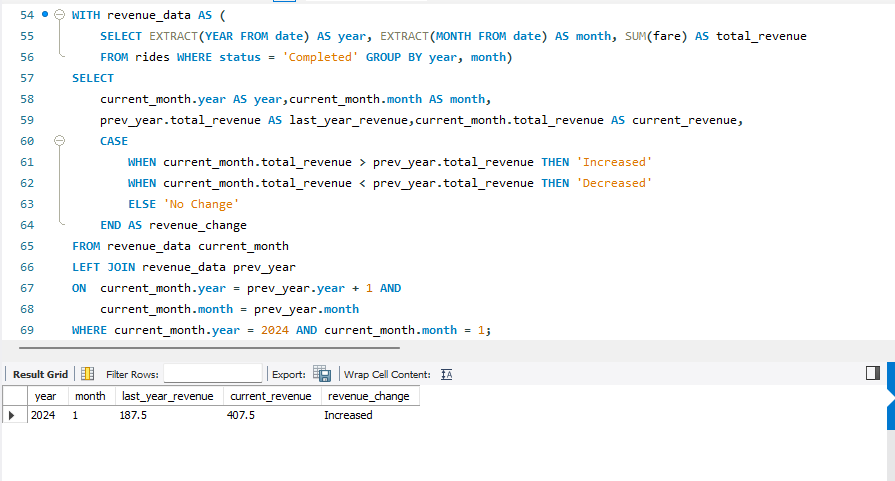
**Top 5 drivers by revenue by month (January, February, March)?**

****

**Top 5 customers by revenue this year?**

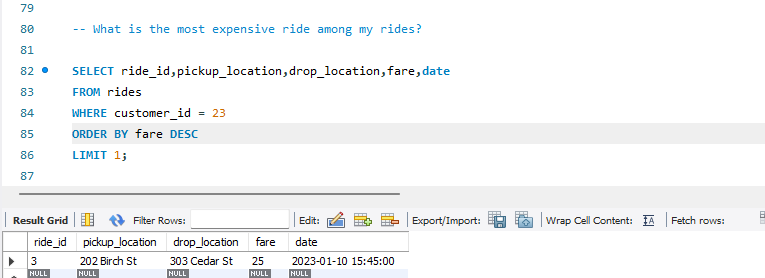
****

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



**Customer Questions:**

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

****

**How many rides did I take with your business?**

**A screenshot of a computer

Description automatically generated**

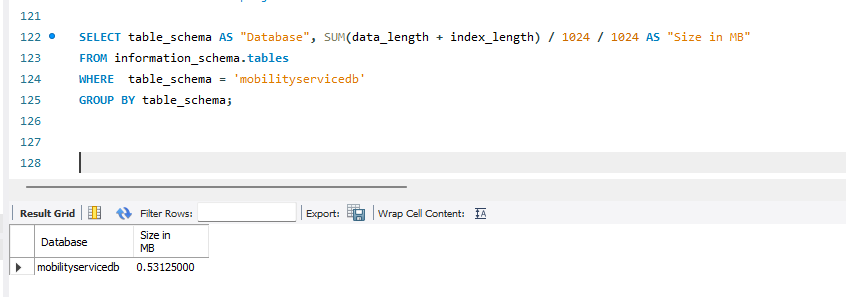
**How much money did I spend with your business year-to-date?**

A screenshot of a computer

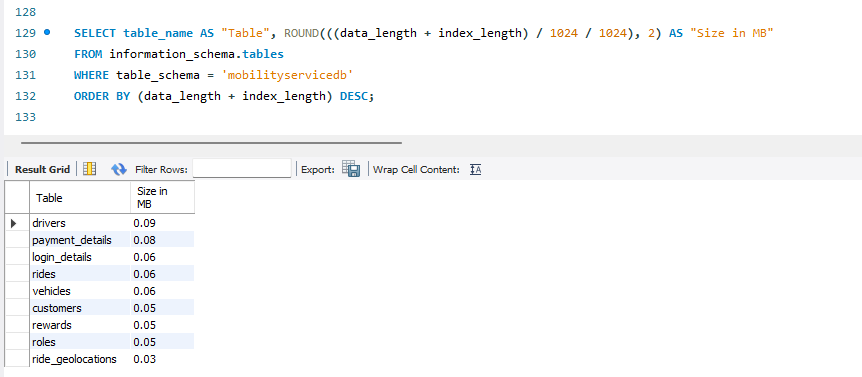
Description automatically generated

**Operations questions**

**what is the size of project database?**



**what is the size of each table?**



**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.

**Conclusion:**

The Mobility Service System is designed to streamline ride-booking and transportation services, making them simple and efficient for both customers and drivers. By focusing on a well-organized database structure and proper data management, the system ensures that essential information—such as user details, vehicle information, ride records, and payment data can be stored securely and accessed easily. Through normalization, we have avoided data redundancy and maintained the integrity of the database, which supports consistent and reliable service.

With features like user registration, profile management, and a clear process for booking and accepting rides, the platform provides a smooth experience for users. The integration of payment options and reward programs enhances the overall user experience and encourages user engagement.

**References:**