**Supta Richard Philip**

**Lecturer, Faculty, Computer Science**

Ride Sharing Application management System

Submitted To

dear professor, supta richard philip

we extend our gratitude for your guidance. our team comprises of ashiqur rahman saron, hasin almas sifat, chaity rani mondol and soumodip madhu. together, we are excited to apply your teachings to our database project and create something exceptional.

best regards from

GROUP-5

group members

|  |  |  |
| --- | --- | --- |
| Name | ID | Contribution |
| HASIN ALMAS SIFAT | **22-48679-3** | **ER DIAGRAM** |
| ASHIQUR RAHMAN SARON | **22-48697-3** | **NORMALIZATION & FINALIZATION** |
| SOUMODIP MADHU | **22-48707-3** | **TABLE CREATION** |
| CHAITY RANI MONDOL | **22-48731-3** | **Query** |

*Table of Contents*

[***Introduction 2***](#_Toc142766252)

[***Case Study 4***](#_Toc142766253)

[***ER Diagram 5***](#_Toc142766254)

[***Normalization 6***](#_Toc142766255)

[***Finalization 14***](#_Toc142766256)

[***Table Description 16***](#_Toc142766257)

[***Table Data Insertion 24***](#_Toc142766258)

[***Query 37***](#_Toc142766259)

[***Conclusion 45***](#_Toc142766260)

# *Introduction*

In the ever-evolving world of transportation, ride-sharing apps have emerged as a revolutionary solution, providing seamless and efficient travel experiences for millions of people worldwide. To efficiently manage and organize this dynamic ecosystem, we present the "AIUB Ride Sharing App Management System". This comprehensive database system aims to streamline the operations of the ride-sharing platform application, facilitating smooth interactions among passengers, riders, and administrators.

* 1. .**KEY FEATURES OF THE APPLICATION**

**User Management:** The database will meticulously store essential user details for both passengers and riders, including their names, contact information, and unique user IDs generated during the registration process. This information will enable effective communication and identification of users throughout the platform.

**Ride History and Payment Tracking:** The database will maintain a comprehensive record of every ride taken by both passengers and riders. Each ride entry will include relevant details such as pick-up and drop-off locations, ride duration, fare, and payment method. This feature will allow users to review their travel history and admins to monitor and analyze ride statistics.

**Payment History and Transactions:** All financial transactions associated with ride payments will be securely stored within the database. This includes payment amounts, dates, and relevant transaction IDs. The system will offer a transparent and organized view of payment histories for both passengers and riders.

**Admin Privileges and Control:** Administrators will have special access to manage user data, ride records, and payment information. The admin interface will empower them to monitor the platform's health, resolve issues, and maintain data integrity effectively.

**Security and Data Protection:** The database project will adhere to industry-standard security protocols to safeguard user information and payment details. Encryption and secure authentication mechanisms will be implemented to prevent unauthorized access and data breaches.

**Scalability and Performance:** The database system will be designed to handle a growing user base and accommodate increased ride activity efficiently. Measures such as indexing, caching, and optimized queries will ensure smooth performance even during peak usage periods.

# *Case Study*

In the "AIUB RIDEZ" ride-sharing service application, users can request rides to travel within the city. Each ride is associated with a specific user and is taken by a driver. Every ride is associated with a specific vehicle, and the availability of vehicles is closely tied to the concept of ride service. Vehicles are categorized into different types, such as car, SUV, or bike. Each ride request must be assigned to a vehicle available within the fleet.

Drivers, who act as the service providers, are responsible for completing the rides. Both drivers and vehicles are essential components of the ride-sharing service. Driver information includes their name, contact details, license number, and vehicle details. Vehicles are defined by their type, license plate number, and capacity.

Users of the "AIUB RIDEZ" application can register for the service and create accounts. Each user is identified by a unique user ID. The registration process includes providing personal details such as name, email, and phone number. Users can also upload a profile picture for identification purposes.

For each ride, the application records details such as the ride ID, start and end locations, date, time, distance, fare, and ride status. Additionally, users have the option to rate and provide feedback for completed rides, which helps improve the overall service quality.

To enhance the user experience, "AIUB RIDEZ" also offers promotional offers and discounts in the form of "Coupons". Users can apply these coupons to their rides to avail discounts.

The ride-sharing service application also allows drivers to set up a digital wallet, known as the "Driver Wallet". This wallet enables riders to collect funds for seamless payments and quick ride booking. The wallet is associated with the rider's account and tracks the income of the rider.

To summarize, "AIUB RIDEZ" is a ride-sharing service application where it offers a convenient and efficient ride-sharing experience.

# *ER DIAGRAM*

# *Normalization*

Normalization is the process of organizing relational data in databases to eliminate redundancy, enhance data integrity, and ensure efficient storage and retrieval. In a ride-sharing app management system, normalization reduces data duplication, maintains accuracy, improves query performance, aids scalability, simplifies maintenance, and ensures consistent, reliable data for optimal functioning and growth. By performing various normalization forms (such as 1NF, 2NF, 3NF, BCNF, etc.) in the relational schema the database management system will be able to perform much more efficiently. The "1st Normal Form" is denoted as "1NF," the "2nd Normal Form" is represented as "2NF," the "3rd Normal Form" is designated by "3NF," and this pattern continues for subsequent normal forms. Notably, the primary key within each table is underscored and table name is presented in bold typeface.

* 1. **MAPPING ER DIAGRAM TO RELATIONAL SCHEMA**
* **Admin** { Admin\_Id, First\_Name, Last\_Name, Email, Phone\_Number, Password }
* **User** {User\_Id, First\_Name, Last\_Name, Email, Nid\_Number, Phone\_Number, Password, Registration\_Date}
* **Payment History** {Payment\_Id, User\_Id, Driver\_Id, Amount, Payment\_Method, Payment\_Date\_&\_Time}

FOREIGN KEY (User\_Id, Driver\_Id) REFRENCES (User, Driver)

* **Payment Type** {Payment\_Type\_Id, Payment\_Type\_Name}

FOREIGN KEY (User\_Id) REFRENCES (User)

* **Card** { Card\_Id, Primary\_Method\_Id, Expiry\_Date, Card\_Id,Card\_Holder\_Name, CVV }

FOREIGN KEY (Primary\_Method\_Id) REFRENCES (Payment\_Id)

* **Payment Method** {Payment\_Method\_Id, Card\_Number, Payment\_Type\_Id, User\_Id}

FOREIGN KEY (User\_Id) REFRENCES (User)

* **Location** { Location\_Id, Name, Address}
* **Destination** {Destination\_Id, Name, Address}
* **Ride** {Ride\_Id, User\_Id, Driver\_Id, Vehicle\_Id, Feedback, Feedback, Pick\_Up\_Location, Ride\_Status, Start\_Time, End\_Time, Distance, Rating}

FOREIGN KEY (User\_Id, Driver\_Id, Vehicle\_Id, Pick\_Up\_Location, Distance) REFRENCES (User, Rider, Vehicle, Destination, Location)

* **Vehicle** {Vehicle\_Id, Vehicle\_Type, Driver\_Id, Capacity}

FOREIGN KEY (Driver\_Id) REFRENCES (Rider)

* **Insurance\_Info** {Insurance\_No., Insurance\_Type, Insurance\_Expiry}
* **Wallet** {Wallet\_Id, User\_Id, User\_Balance}

FOREIGN KEY (User\_Id) REFRENCES (User)

* **Rider** {Driver\_Id, First\_Name, Last\_Name, Email, License\_Number, Phone\_Number, Password}
* **Rating & Feedback** {Rating\_Id, User\_Id, Ride\_Id, Driver\_Id, Rating, Feedback}

FOREIGN KEY (User\_Id, Ride\_Id, Driver\_Id) REFRENCES (User, Rider, Ride)

* **Promotions & Coupons** {Coupon\_Id, Discount\_Percentage, Expiring\_Date}

1. **IMPLEMENTING THE PROCESS OF NORMALIZATION**
2. **Admin Manages Users & Riders :**

**UNF:**

**Admin And User Data (**Admin\_Id, Admin\_First\_Name, Admin\_Last\_Name, Admin\_Email, Admin\_Phone\_Number, Admin\_Password, User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Password, User\_Registration\_Date, Driver\_Id, Driver\_First\_Name, Driver\_Last\_Name, Driver\_Email, Driver\_License\_Number, Driver\_Phone\_Number, Driver\_Password)

**1NF:** Creating new User And Rider table to remove multivalued attributes from Admin User Data table.

**Admin** (Admin\_Id, Admin\_First\_Name, Admin\_Last\_Name, Admin\_Email, Admin\_Phone\_Number, Admin\_Password)

**User** (User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Password, User\_Registration\_Date)

**Rider** (Driver\_Id, Driver\_First\_Name, Driver\_Last\_Name, Driver\_Email, Driver\_License\_Number, Driver\_Phone\_Number, Driver\_Password)

**2NF:** No changes are needed in 1NF since there are no partial dependencies.

**Admin** (Admin\_Id, Admin\_First\_Name, Admin\_Last\_Name, Admin\_Email, Admin\_Phone\_Number, Admin\_Password)

**User** (User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Password, User\_Registration\_Date)

**Driver** (Driver\_Id, Driver\_First\_Name, Driver\_Last\_Name, Driver\_Email, Driver\_License\_Number, Driver\_Phone\_Number, Driver\_Password)

**3NF:** No changes are needed in 2NF since there are no transitive dependencies.

**Admin** (Admin\_Id, Admin\_First\_Name, Admin\_Last\_Name, Admin\_Email, Admin\_Phone\_Number, Admin\_Password)

**User** (User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Registration\_Date, User\_Password)

**Driver** (Driver\_Id, Driver\_First\_Name, Driver\_Last\_Name, Driver\_Email, Driver\_License\_Number, Driver\_Phone\_Number, Driver\_Password)

1. **Rider Owns Vehicle :**

**UNF:**

**Rider Vehicle Data (**Driver\_Id, Driver\_First\_Name, Driver\_Last\_Name, Driver\_Email, Driver\_License\_Number, Driver\_Phone\_Number, Driver\_Password, Vehicle\_Id, Vehicle\_Type, Capacity, Insurance\_No., Insurance\_Type, Insurance\_Expiry)

**1NF:** Creating new Vehicle and Rider table to remove multivalued attributes from Rider Vehicle Data table.

**Driver** (Driver\_Id, Driver\_First\_Name, Driver\_Last\_Name, Driver\_Email, Driver\_License\_Number, Driver\_Phone\_Number, Driver\_Password)

**Vehicle**(Vehicle\_Id, Vehicle\_Type, Capacity, Insurance\_No., Insurance\_Type, Insurance\_Expiry)

**2NF:** No changes are needed in 1NF since there are no partial dependencies.

**Driver** (Driver\_Id, Driver\_First\_Name, Driver\_Last\_Name, Driver\_Email, Driver\_License\_Number, Driver\_Phone\_Number, Driver\_Password)

**Vehicle** (Vehicle\_Id, Vehicle\_Type, Capacity, Insurance\_No., Insurance\_Type, Insurance\_Expiry)

**3NF:** No changes are needed in 2NF since there are no transitive dependencies.

**Driver** (Driver\_Id, Driver\_First\_Name, Driver\_Last\_Name, Driver\_Email, Driver\_License\_Number, Driver\_Phone\_Number, Driver\_Password)

**Vehicle** (Vehicle\_Id, Vehicle\_Type, Capacity)

Insurance Info (Insurance\_No., Insurance\_Type, Insurance\_Expiry)

1. **Ride Has Location & Destination :**

**UNF :**

**Ride Info (**Ride\_Id, User\_Id, Driver\_Id, Vehicle\_Id, Feedback, Feedback, Pick\_Up\_Location, Ride\_Status, Start\_Time, End\_Time, Distance, Rating, Location\_Id, Name, Address, Destination\_Id, Name, Address**)**

**1NF:** No changes are needed in UNF since there are no multivalued attributes.

**Ride Info** (Ride\_Id, User\_Id, Driver\_Id, Vehicle\_Id, Feedback, Feedback, Pick\_Up\_Location, Ride\_Status, Start\_Time, End\_Time, Distance, Rating, Location\_Id, Name, Address, Destination\_Id, Name, Address)

**2NF:** Creating new Destination, Location and Ride table to remove partial dependencies from Rider Info Data table to remove partial dependencies.

**Ride Info** (Ride\_Id, User\_Id, Driver\_Id, Vehicle\_Id, Feedback, Feedback, Pick\_Up\_Location, Ride\_Status, Start\_Time, End\_Time, Distance, Rating, Location\_Id, Destination\_Id)

**Location** (Location\_Id, Name, Address)

**Destination** (Destination\_Id, Name, Address)

**3NF:** No changes are needed in 2NF since there are no transitive dependencies.

**Ride Info** (Ride\_Id, User\_Id, Driver\_Id, Vehicle\_Id, Feedback, Pick\_Up\_Location, Ride\_Status, Start\_Time, End\_Time, Distance, Rating, Location\_Id, Destination\_Id)

**Location** (Location\_Id, Name, Address)

**Destination** (Destination\_Id, Name, Address)

1. **User Makes Payment :**

**UNF :**

**User Payment** (User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Registration\_Date, User\_Password, Payment\_Type\_Id, Payment\_Type\_Name , Payment\_Method\_Id, Card\_Number, Payment\_Type\_Id, User\_Id, Card\_Id, Primary\_Method\_Id, Expiry\_Date, Card\_Id,Card\_Holder\_Name, CVV)

**1NF:** Creating new User and Payment Info table to remove multivalued attributes from User Payment Data table.

**User** (User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Registration\_Date, User\_Password)

**Payment Info**(Payment\_Type\_Id, Payment\_Type\_Name, Payment\_Method\_Id, Card\_Number, User\_Id, Card\_Id, Primary\_Method\_Id, Expiry\_Date, Card\_Holder\_Name, CVV)

**2NF:** Creating new Payment Type, Card and Payment Method table from Payment Info Table to remove partial dependencies from User Payment Info table.

**User** (User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Registration\_Date, User\_Password)

**Payment Type** (Payment\_Type\_Id, Payment\_Type\_Name)

**Payment Method** (Payment\_Method\_Id, Card\_Number, Payment\_Type\_Id, User\_Id)

**Card**(Card\_Id,Primary\_Method\_Id,Expiry\_Date,Card\_Holder\_Name,CVV)

**3NF:** No changes are needed in 2NF since there are no transitive dependencies.

**User** (User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Registration\_Date, User\_Password)

**Payment Type**(Payment\_Type\_Id, Payment\_Type\_Name)

**Payment Method**(Payment\_Method\_Id, Card\_Number, Payment\_Type\_Id, User\_Id)

**Card**(Card\_Id,Primary\_Method\_Id,Expiry\_Date,Card\_Holder\_Name,CVV)

1. **User Has Payment History :**

**UNF:**

**User Payment History**(User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Registration\_Date, User\_Password, Payment\_Id, Driver\_Id, User\_Id, Payment\_Method, Amount, Payment\_Date\_&\_Time)

**1NF:** Creating new User and Payment History table to remove multivalued attributes from User Payment History data table.

**User**(User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Registration\_Date, User\_Password)

**Payment History**(Payment\_Id, Driver\_Id, User\_Id, Payment\_Method, Amount, Payment\_Date\_&\_Time)

**2NF:** No changes are needed in 1NF since there are no partial dependencies.

**User**(User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Registration\_Date, User\_Password)

**Payment History**(Payment\_Id, Driver\_Id, User\_Id, Payment\_Method, Amount, Payment\_Date\_&\_Time)

**3NF:** No changes are needed in 2NF since there are no transitive dependencies.

**User**(User\_Id, User\_First\_Name, User\_Last\_Name, User\_Email, User\_Nid\_Number, User\_Phone\_Number, User\_Registration\_Date, User\_Password)

**Payment History**(Payment\_Id, Driver\_Id, User\_Id, Payment\_Method, Amount, Payment\_Date\_&\_Time)

1. **Admin Launches Offers And Coupons :**

**UNF:**

**Promotions & Coupons** (Coupon\_Id, Discount\_Percentage, Expiring\_Date **)**

**1NF:** No changes are needed in UNF since there are no multivalued attributes.

**Promotions & Coupons** (Coupon\_Id, Discount\_Percentage, Expiring\_Date **)**

**2NF:** No changes are needed in 1NF since there are no partial dependencies.

**Promotions & Coupons** (Coupon\_Id, Discount\_Percentage, Expiring\_Date **)**

**3NF:** No changes are needed in 2NF since there are no transitive dependencies.

**Promotions & Coupons** (Coupon\_Id, Discount\_Percentage, Expiring\_Date **)**

1. **User Gives Ratings & Feedback :**

**UNF:**

Rating & Feedback(Rating\_Id, User\_Id, Ride\_Id, Driver\_Id, Rating, Feedback)

**1NF:** No changes are needed in UNF since there are no multivalued attributes.

**Rating & Feedback** (Rating\_Id, User\_Id, Ride\_Id, Driver\_Id, Rating, Feedback)

**2NF:** No changes are needed in 1NF since there are no partial dependencies.

**Rating & Feedback** (Rating\_Id, User\_Id, Ride\_Id, Driver\_Id, Rating, Feedback)

**3NF:** No changes are needed in 2NF since there are no transitive dependencies.

**Rating & Feedback** (Rating\_Id, User\_Id, Ride\_Id, Driver\_Id, Rating, Feedback)

# *Finalization*

Finalization in a ride-sharing application ensures a well-structured, efficient, and organized database design. It's vital for maintaining data integrity, reducing redundancy, and enabling smooth operations. By defining clear relationships, normalizing data, and establishing foreign keys, finalization ensures accurate information storage, retrieval, and updates. In a ride-sharing service, finalization optimizes user experiences, enhances payment processing, streamlines ride management, and facilitates promotional activities. The ultimate iteration of the relational schema is presented herewith.

**Entities and Their Attributes:**

1. **Admin**: Admin\_Id, First\_Name, Last\_Name, Email, Phone\_Number, Password
2. **User**: User\_Id, First\_Name, Last\_Name, Email, Nid\_Number, Phone\_Number, Password, Registration\_Date
3. **Payment Type**: Payment\_Type\_Id, Payment\_Type\_Name
4. **Card**: Card\_Id, Primary\_Method\_Id, Expiry\_Date, Card\_Holder\_Name, CVV
5. **Payment Method**: Payment\_Method\_Id, Card\_Number, Payment\_Type\_Id, User\_Id
6. **Payment History**: Payment\_Id, User\_Id, Driver\_Id, Amount, Payment\_Method, Payment\_Date\_&\_Time
7. **Location**: Location\_Id, Name, Address
8. **Destination**: Destination\_Id, Name, Address
9. **Ride**: Ride\_Id, User\_Id, Driver\_Id, Vehicle\_Id, Feedback, Feedback, Pick\_Up\_Location, Ride\_Status, Start\_Time, End\_Time, Distance, Rating
10. **Vehicle**: Vehicle\_Id, Vehicle\_Type, Driver\_Id, Capacity, Insurance\_No
11. **Insurance\_Info**: Insurance\_No., Insurance\_Type, Insurance\_Expiry
12. **Wallet**: Wallet\_Id, User\_Id, User\_Balance
13. **Rider**: Driver\_Id, First\_Name, Last\_Name, Email, License\_Number, Phone\_Number, Password
14. **Rating & Feedback**: Rating\_Id, User\_Id, Ride\_Id, Driver\_Id, Rating, Feedback
15. **Promotions & Coupons**: Coupon\_Id, Discount\_Percentage, Expiring\_Date

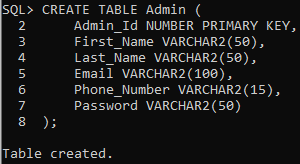
**Relationships:**

* **Payment History** (User\_Id, Driver\_Id) REFERENCE User(User\_Id), Rider(Driver\_Id)
* **Payment Type** (User\_Id) REFERENCE User(User\_Id)
* **Card** (Primary\_Method\_Id) REFERENCE Payment History(Payment\_Id)
* **Payment Method** (User\_Id) REFERENCE User(User\_Id)
* **Payment History** (User\_Id, Driver\_Id, Payment\_Method) REFERENCE User(User\_Id), Rider(Driver\_Id), Payment Method(Payment\_Method\_Id)
* **Ride** (User\_Id, Driver\_Id, Vehicle\_Id, Pick\_Up\_Location, Distance) REFERENCE User(User\_Id), Rider(Driver\_Id), Vehicle(Vehicle\_Id), Destination(Pick\_Up\_Location), Location(Distance)
* **Vehicle** (Driver\_Id) REFERENCE Rider(Driver\_Id)
* **Wallet** (User\_Id) REFERENCE User(User\_Id)
* **Rating & Feedback** (User\_Id, Ride\_Id, Driver\_Id) REFERENCE User(User\_Id), Rider(Ride\_Id), Ride(Driver\_Id)

# *Table Description*

Utilizing the concluded version of the database schema, progression to table creation becomes feasible. Oracle 10g software is employed for this purpose, employing SQL (Standard Query Language) for table generation. Below is the depiction of table creation, inclusive of the query language employed for its realization.

1. **ADMIN TABLE:**

CREATE TABLE Admin (

*Figure 6.1*

Admin\_Id NUMBER PRIMARY KEY,

First\_Name VARCHAR2(50),

Last\_Name VARCHAR2(50),

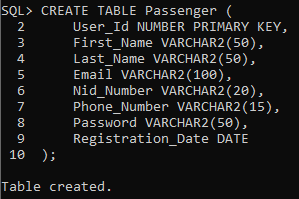
Email VARCHAR2(100),

Phone\_Number VARCHAR2(15),

Password VARCHAR2(50)

);

1. **USER TABLE:**

CREATE TABLE Passenger (

*Figure 6.2*

User\_Id NUMBER PRIMARY KEY,

First\_Name VARCHAR2(50),

Last\_Name VARCHAR2(50),

Email VARCHAR2(100),

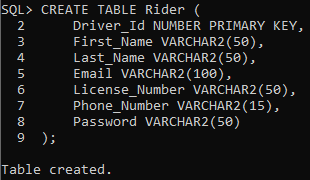
Nid\_Number VARCHAR2(20),

Phone\_Number VARCHAR2(15),

Password VARCHAR2(50),

Registration\_Date DATE

);

1. **RIDER TABLE:**

*Figure 6.3*

CREATE TABLE Rider (

Driver\_Id NUMBER PRIMARY KEY,

First\_Name VARCHAR2(50),

Last\_Name VARCHAR2(50),

Email VARCHAR2(100),

License\_Number VARCHAR2(50),

Phone\_Number VARCHAR2(15),

Password VARCHAR2(50)

);

1. **RIDE TABLE:**

CREATE TABLE Ride (

Ride\_Id NUMBER PRIMARY KEY,

User\_Id NUMBER,

Driver\_Id NUMBER,

Vehicle\_Id NUMBER,

Feedback VARCHAR2(1000),

Pick\_Up\_Location NUMBER,

Ride\_Status VARCHAR2(50),

Start\_Time TIMESTAMP,

End\_Time TIMESTAMP,

Distance NUMBER,

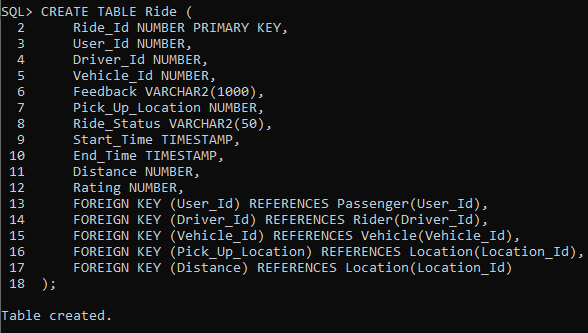
Rating NUMBER,

FOREIGN KEY (User\_Id) REFERENCES Passenger(User\_Id),

FOREIGN KEY (Driver\_Id) REFERENCES Rider(Driver\_Id),

FOREIGN KEY (Vehicle\_Id) REFERENCES Vehicle(Vehicle\_Id),

FOREIGN KEY (Pick\_Up\_Location) REFERENCES Location(Location\_Id),

 FOREIGN KEY (Distance) REFERENCES Location(Location\_Id) );

*Figure 6.4*

1. **VEHICLE TABLE:**

CREATE TABLE Vehicle (

Vehicle\_Id NUMBER PRIMARY KEY,

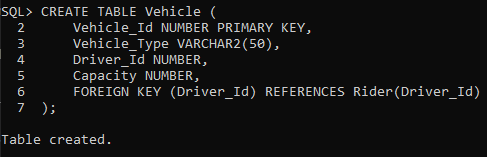
Vehicle\_Type VARCHAR2(50),

Driver\_Id NUMBER,

Capacity NUMBER,

FOREIGN KEY (Driver\_Id) REFERENCES Rider(Driver\_Id)

);



*Figure 6.5*

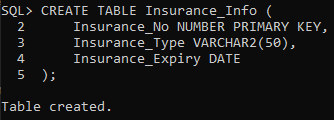
1. **INSURANCE TABLE:**

CREATE TABLE Insurance\_Info (

Insurance\_No NUMBER PRIMARY KEY,

Insurance\_Type VARCHAR2(50),

Insurance\_Expiry DATE

);

*Figure 6.6*

1. **WALLET TABLE:**

CREATE TABLE Wallet (

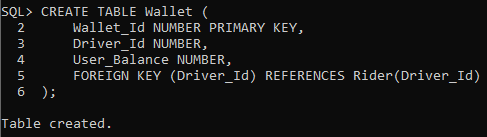
Wallet\_Id NUMBER PRIMARY KEY,

Driver\_Id NUMBER,

User\_Balance NUMBER,

FOREIGN KEY (Driver\_Id) REFERENCES Rider(Driver\_Id)

);



*Figure 6.7*

1. **PAYMENT HISTORY TABLE:**

CREATE TABLE Payment\_History (

Payment\_Id NUMBER PRIMARY KEY,

User\_Id NUMBER,

Driver\_Id NUMBER,

Amount NUMBER,

Payment\_Method NUMBER,

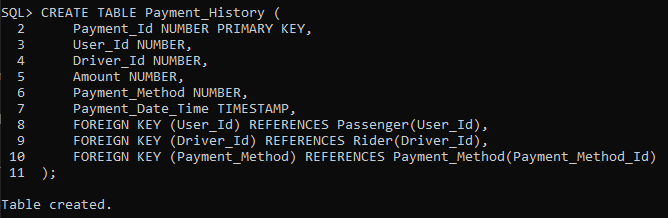
Payment\_Date\_Time TIMESTAMP,

FOREIGN KEY (User\_Id) REFERENCES Passenger(User\_Id),

FOREIGN KEY (Driver\_Id) REFERENCES Rider(Driver\_Id),

FOREIGN KEY (Payment\_Method) REFERENCES Payment\_Method(Payment\_Method\_Id)

);



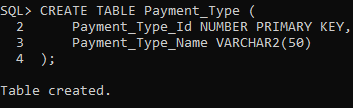
*Figure 6.8*

1. **PAYMENT TYPE TABLE:**

CREATE TABLE Payment\_Type (

Payment\_Type\_Id NUMBER PRIMARY KEY,

Payment\_Type\_Name VARCHAR2(50) );



*Figure 6.9*

1. **PAYMENT METHOD TABLE:**

CREATE TABLE Payment\_Method (

Payment\_Method\_Id NUMBER PRIMARY KEY,

Card\_Number VARCHAR2(50),

Payment\_Type\_Id NUMBER,

User\_Id NUMBER,

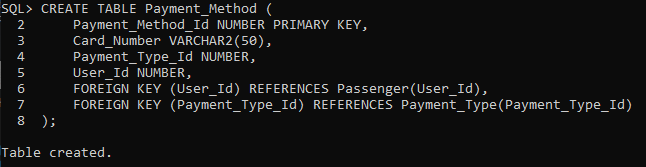
FOREIGN KEY (User\_Id)

REFERENCES Passenger(User\_Id),

FOREIGN KEY (Payment\_Type\_Id)

REFERENCES Payment\_Type(Payment\_Type\_Id)

);



*Figure 6.10*

1. **CARD TABLE:**

CREATE TABLE Card (

Card\_Id NUMBER PRIMARY KEY,

Primary\_Method\_Id NUMBER,

Expiry\_Date DATE,

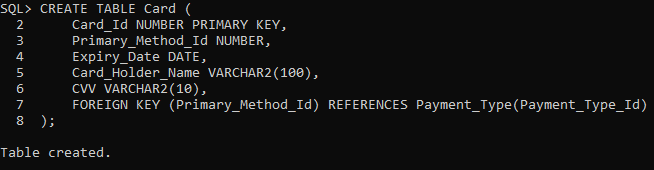
Card\_Holder\_Name VARCHAR2(100),

CVV VARCHAR2(10),

FOREIGN KEY (Primary\_Method\_Id)

REFERENCES Payment\_Type (Payment\_Type\_Id)

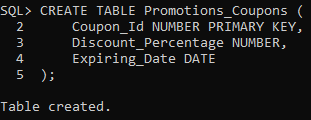
);



*Figure 6.11*

1. **PROMOTIONS & COUPONS TABLE:**

CREATE TABLE Promotions\_Coupons (

 Coupon\_Id NUMBER PRIMARY KEY,

Discount\_Percentage NUMBER,

Expiring\_Date DATE

);

*Figure 6.12*

1. **RATING & FEEDBACK TABLE:**

CREATE TABLE Feedback (

Rating\_Id NUMBER PRIMARY KEY,

User\_Id NUMBER,

Ride\_Id NUMBER,

Driver\_Id NUMBER,

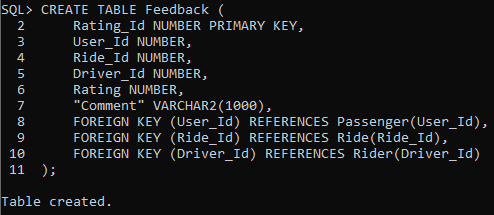
Rating NUMBER,

"Comment" VARCHAR2(1000),

FOREIGN KEY (User\_Id) REFERENCES Passenger(User\_Id),

FOREIGN KEY (Ride\_Id) REFERENCES Ride(Ride\_Id),

FOREIGN KEY (Driver\_Id) REFERENCES Rider(Driver\_Id)

);

*Figure 6.13*

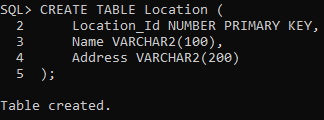
1. **PICK UP LOCATION TABLE:**

CREATE TABLE Location (

Location\_Id NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Address VARCHAR2(200) );



*Figure 6.14*

1. **DESTINATION:**

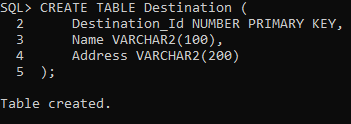
CREATE TABLE Destination (

Destination\_Id NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Address VARCHAR2(200)

);



*Figure 6.15*

# *Table Data Insertion*

Following the establishment of the table in accordance with the relational schema, the focus transitions to the phase of data insertion. Once again, SQL (Standard Query Language) will be employed for this purpose.

1. **Inserting Data Into The "Admin" Table:**

INSERT INTO Admin (Admin\_Id, First\_Name, Last\_Name, Email, Phone\_Number, Password)

VALUES (1, 'Admin', 'Smith', 'admin@example.com', '1234567890', 'adminpass');

INSERT INTO Admin (Admin\_Id, First\_Name, Last\_Name, Email, Phone\_Number, Password)

VALUES (2, 'Admin', 'Johnson', 'admin2@example.com', '9876543210', 'adminsecure');

INSERT INTO Admin (Admin\_Id, First\_Name, Last\_Name, Email, Phone\_Number, Password)

VALUES (3, 'Admin', 'Doe', 'admin3@example.com', '5678901234', 'admin123');

INSERT INTO Admin (Admin\_Id, First\_Name, Last\_Name, Email, Phone\_Number, Password)

VALUES (4, 'Admin', 'Brown', 'admin4@example.com', '0123456789', 'admin456');

*Figure 7.1*

1. **Inserting Data Into The "User" Table:**

INSERT INTO Passenger (User\_Id, First\_Name, Last\_Name, Email, Nid\_Number, Phone\_Number, Password, Registration\_Date)

VALUES (1, 'John', 'Doe', 'john@example.com', '1234567890', '1234567890', 'password123', TO\_DATE('2023-08-06', 'YYYY-MM-DD'));

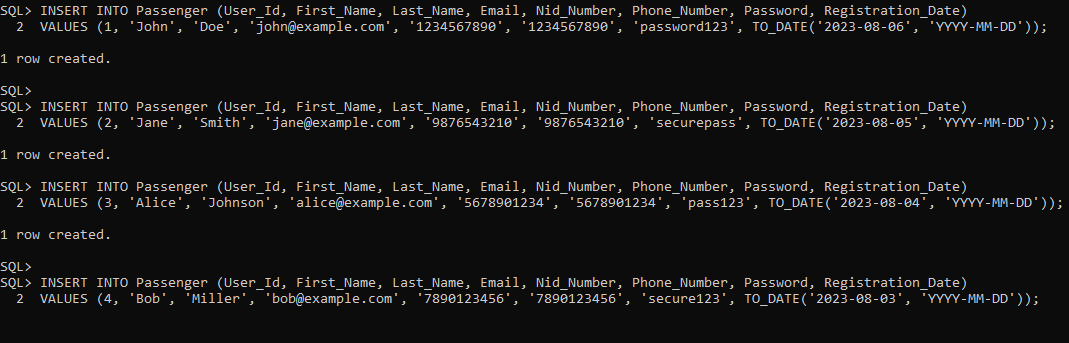
INSERT INTO Passenger (User\_Id, First\_Name, Last\_Name, Email, Nid\_Number, Phone\_Number, Password, Registration\_Date)

VALUES (2, 'Jane', 'Smith', 'jane@example.com', '9876543210', '9876543210', 'securepass', TO\_DATE('2023-08-05', 'YYYY-MM-DD'));

INSERT INTO Passenger (User\_Id, First\_Name, Last\_Name, Email, Nid\_Number, Phone\_Number, Password, Registration\_Date)

VALUES (3, 'Alice', 'Johnson', 'alice@example.com', '5678901234', '5678901234', 'pass123', TO\_DATE('2023-08-04', 'YYYY-MM-DD'));

INSERT INTO Passenger (User\_Id, First\_Name, Last\_Name, Email, Nid\_Number, Phone\_Number, Password, Registration\_Date)

VALUES (4, 'Bob', 'Miller', 'bob@example.com', '7890123456', '7890123456', 'secure123', TO\_DATE('2023-08-03', 'YYYY-MM-DD'));

*Figure 7.2*

1. **Inserting Data Into The "Rider" Table:**

INSERT INTO Rider (Driver\_Id, First\_Name, Last\_Name, Email, License\_Number, Phone\_Number, Password)

VALUES (1, 'Michael', 'Johnson', 'michael@example.com', 'ABCD1234', '9876543210', 'riderpass');

INSERT INTO Rider (Driver\_Id, First\_Name, Last\_Name, Email, License\_Number, Phone\_Number, Password)

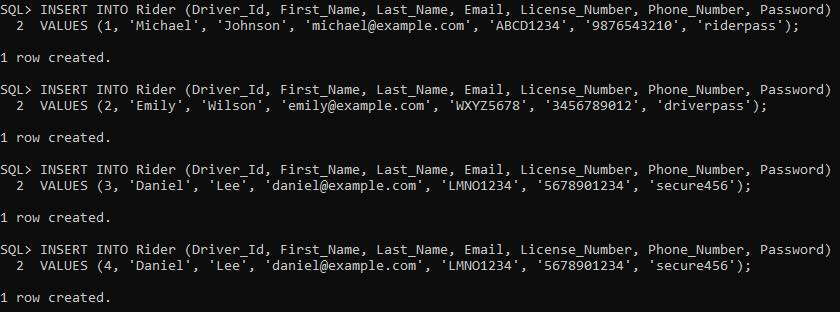
VALUES (2, 'Emily', 'Wilson', 'emily@example.com', 'WXYZ5678', '3456789012', 'driverpass');

INSERT INTO Rider (Driver\_Id, First\_Name, Last\_Name, Email, License\_Number, Phone\_Number, Password)

VALUES (3, 'Daniel', 'Lee', 'daniel@example.com', 'LMNO1234', '5678901234', 'secure456');

INSERT INTO Rider (Driver\_Id, First\_Name, Last\_Name, Email, License\_Number, Phone\_Number, Password)

VALUES (4, 'Daniel', 'Lee', 'daniel@example.com', 'LMNO1234', '5678901234', 'secure456');



*Figure 7.3*

1. **Inserting Data Into The "Ride" Table:**

INSERT INTO Ride (Ride\_Id, User\_Id, Driver\_Id, Vehicle\_Id, Feedback, Pick\_Up\_Location, Ride\_Status, Start\_Time, End\_Time, Distance, Rating, Destination)

VALUES (1, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL);

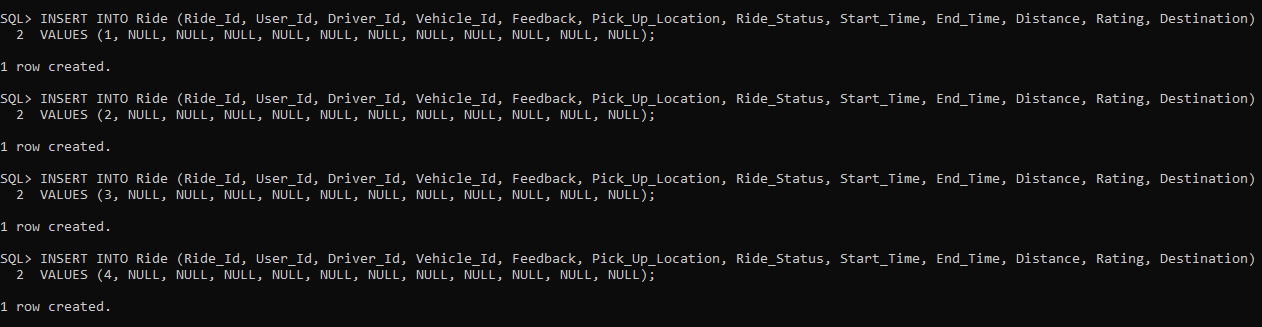
INSERT INTO Ride (Ride\_Id, User\_Id, Driver\_Id, Vehicle\_Id, Feedback, Pick\_Up\_Location, Ride\_Status, Start\_Time, End\_Time, Distance, Rating, Destination)

VALUES (2, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO Ride (Ride\_Id, User\_Id, Driver\_Id, Vehicle\_Id, Feedback, Pick\_Up\_Location, Ride\_Status, Start\_Time, End\_Time, Distance, Rating, Destination)

VALUES (3, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO Ride (Ride\_Id, User\_Id, Driver\_Id, Vehicle\_Id, Feedback, Pick\_Up\_Location, Ride\_Status, Start\_Time, End\_Time, Distance, Rating, Destination)

VALUES (4, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL);

*Figure 7.4*

1. **Inserting Data Into The "Vehicle" Table:**

INSERT INTO Vehicle (Vehicle\_Id, Vehicle\_Type, Driver\_Id, Capacity)

VALUES (1, 'Sedan', 2, 4);

INSERT INTO Vehicle (Vehicle\_Id, Vehicle\_Type, Driver\_Id, Capacity)

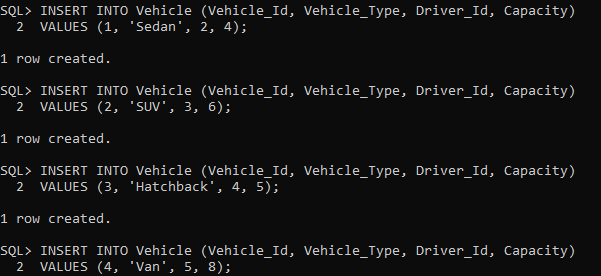
VALUES (2, 'SUV', 3, 6);

INSERT INTO Vehicle (Vehicle\_Id, Vehicle\_Type, Driver\_Id, Capacity)

VALUES (3, 'Hatchback', 4, 5);

INSERT INTO Vehicle (Vehicle\_Id, Vehicle\_Type, Driver\_Id, Capacity)

VALUES (4, 'Van', 1, 8);



*Figure 7.5*

1. **Inserting Data Into The "Insurance\_Info" Table:**

INSERT INTO Insurance\_Info (Insurance\_No, Insurance\_Type, Insurance\_Expiry)

VALUES ('INS123', 'Auto', TO\_DATE('2023-12-31', 'YYYY-MM-DD'));

INSERT INTO Insurance\_Info (Insurance\_No, Insurance\_Type, Insurance\_Expiry)

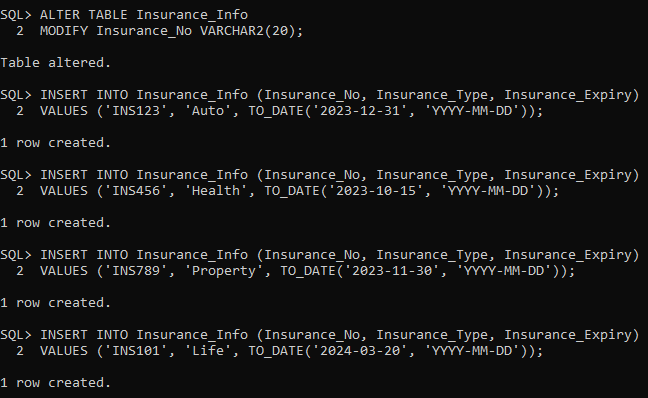
VALUES ('INS456', 'Health', TO\_DATE('2023-10-15', 'YYYY-MM-DD'));

INSERT INTO Insurance\_Info (Insurance\_No, Insurance\_Type, Insurance\_Expiry)

VALUES ('INS789', 'Property', TO\_DATE('2023-11-30', 'YYYY-MM-DD'));

INSERT INTO Insurance\_Info (Insurance\_No, Insurance\_Type, Insurance\_Expiry)

VALUES ('INS101', 'Life', TO\_DATE('2024-03-20', 'YYYY-MM-DD'));



*Figure 7.6*

1. **Inserting Data Into The "Payment\_Method" Table:**

INSERT INTO Payment\_Method (Payment\_Method\_Id, Card\_Number, Payment\_Type\_Id, User\_Id)

VALUES (1, '1234-5678-9012-3456', 1, 1);

INSERT INTO Payment\_Method (Payment\_Method\_Id, Card\_Number, Payment\_Type\_Id, User\_Id)

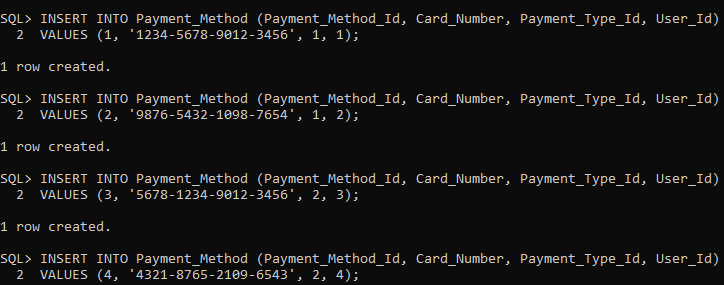
VALUES (2, '9876-5432-1098-7654', 1, 2);

INSERT INTO Payment\_Method (Payment\_Method\_Id, Card\_Number, Payment\_Type\_Id, User\_Id)

VALUES (3, '5678-1234-9012-3456', 2, 3);

INSERT INTO Payment\_Method (Payment\_Method\_Id, Card\_Number, Payment\_Type\_Id, User\_Id)

VALUES (4, '4321-8765-2109-6543', 2, 4);



*Figure 7.7*

1. **Inserting Data Into The "Payment\_Type" Table:**

INSERT INTO Payment\_Type (Payment\_Type\_Id, Payment\_Type\_Name)

VALUES (1, 'Credit Card');

INSERT INTO Payment\_Type (Payment\_Type\_Id, Payment\_Type\_Name)

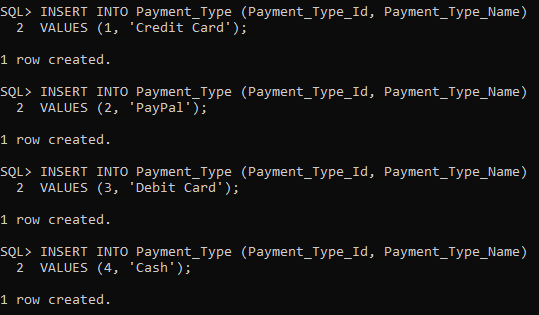
VALUES (2, 'PayPal');

INSERT INTO Payment\_Type (Payment\_Type\_Id, Payment\_Type\_Name)

VALUES (3, 'Debit Card');

INSERT INTO Payment\_Type (Payment\_Type\_Id, Payment\_Type\_Name)

VALUES (4, 'Cash');

***Figure 7.8***

1. **Inserting Data Into The "Payment\_History" Table:**

INSERT INTO Payment\_History (Payment\_Id, User\_Id, Driver\_Id, Amount, Payment\_Method, Payment\_Date\_Time)

VALUES (1, 1, 2, 25.00, 1, TO\_TIMESTAMP('2023-08-01 15:30:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Payment\_History (Payment\_Id, User\_Id, Driver\_Id, Amount, Payment\_Method, Payment\_Date\_Time)

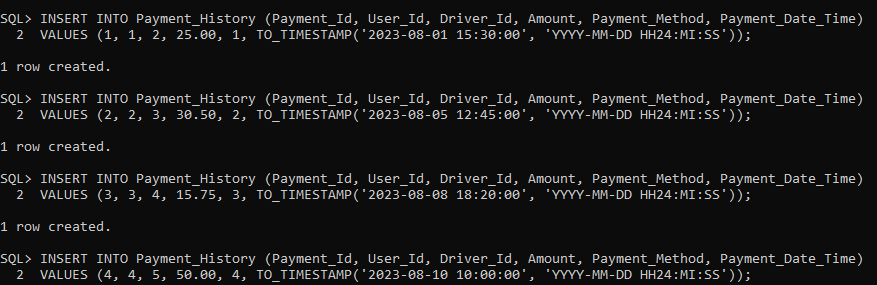
VALUES (2, 2, 3, 30.50, 2, TO\_TIMESTAMP('2023-08-05 12:45:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Payment\_History (Payment\_Id, User\_Id, Driver\_Id, Amount, Payment\_Method, Payment\_Date\_Time)

VALUES (3, 3, 4, 15.75, 3, TO\_TIMESTAMP('2023-08-08 18:20:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Payment\_History (Payment\_Id, User\_Id, Driver\_Id, Amount, Payment\_Method, Payment\_Date\_Time)

VALUES (4, 4, 1, 50.00, 4, TO\_TIMESTAMP('2023-08-10 10:00:00', 'YYYY-MM-DD HH24:MI:SS'));



*Figure 7.9*

1. **Inserting Data Into The "Destination" Table:**

INSERT INTO Destination (Destination\_Id, Name, Address)

VALUES (1, 'Airport', '456 Airport Road');

INSERT INTO Destination (Destination\_Id, Name, Address)

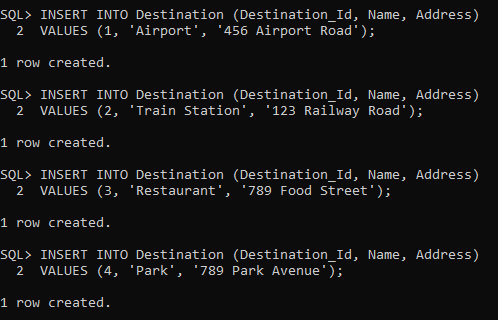
VALUES (2, 'Train Station', '123 Railway Road');

INSERT INTO Destination (Destination\_Id, Name, Address)

VALUES (3, 'Restaurant', '789 Food Street');

INSERT INTO Destination (Destination\_Id, Name, Address)

VALUES (4, 'Park', '789 Park Avenue');



*Figure 7.10*

1. **Inserting Data Into The "Feedback" Table:**

INSERT INTO Feedback (Rating\_Id, User\_Id, Ride\_Id, Driver\_Id, Rating, "Comment")

VALUES (1, 1, 1, 2, 4.5, 'The driver was very courteous and the ride was comfortable.');

INSERT INTO Feedback (Rating\_Id, User\_Id, Ride\_Id, Driver\_Id, Rating, "Comment")

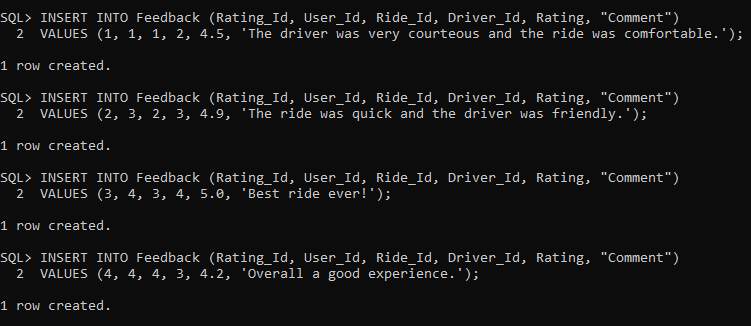
VALUES (2, 3, 2, 3, 4.9, 'The ride was quick and the driver was friendly.');

INSERT INTO Feedback (Rating\_Id, User\_Id, Ride\_Id, Driver\_Id, Rating, "Comment")

VALUES (3, 4, 3, 4, 5.0, 'Best ride ever!');

INSERT INTO Feedback (Rating\_Id, User\_Id, Ride\_Id, Driver\_Id, Rating, "Comment")

VALUES (4, 4, 4, 3, 4.2, 'Overall a good experience.');



*Figure 7.11*

1. **Inserting Data Into The "Location" Table:**

INSERT INTO Location (Location\_Id, Name, Address)

VALUES (1, 'Central Park', '123 Main Street');

INSERT INTO Location (Location\_Id, Name, Address)

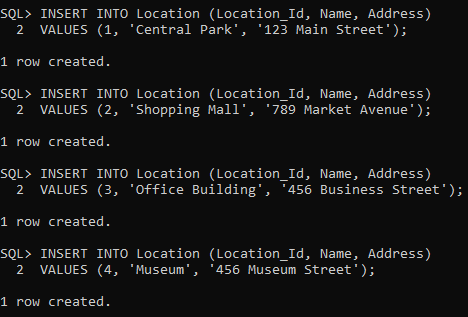
VALUES (2, 'Shopping Mall', '789 Market Avenue');

INSERT INTO Location (Location\_Id, Name, Address)

VALUES (3, 'Office Building', '456 Business Street');

INSERT INTO Location (Location\_Id, Name, Address)

VALUES (4, 'Museum', '456 Museum Street');



*Figure 7.12*

1. **Inserting Data Into The "Promotions\_Coupons" Table:**

INSERT INTO Promotions\_Coupons (Coupon\_Id, Discount\_Percentage, Expiring\_Date)

VALUES (1, 10, TO\_DATE('2023-09-30', 'YYYY-MM-DD'));

INSERT INTO Promotions\_Coupons (Coupon\_Id, Discount\_Percentage, Expiring\_Date)

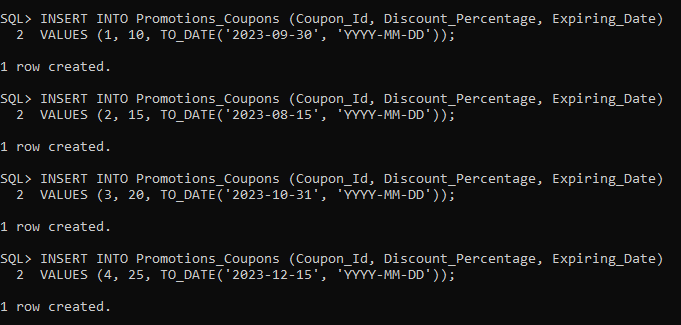
VALUES (2, 15, TO\_DATE('2023-08-15', 'YYYY-MM-DD'));

INSERT INTO Promotions\_Coupons (Coupon\_Id, Discount\_Percentage, Expiring\_Date)

VALUES (3, 20, TO\_DATE('2023-10-31', 'YYYY-MM-DD'));

INSERT INTO Promotions\_Coupons (Coupon\_Id, Discount\_Percentage, Expiring\_Date)

VALUES (4, 25, TO\_DATE('2023-12-15', 'YYYY-MM-DD'));



*Figure 7.13*

1. **Inserting Data Into The "Card" Table:**

INSERT INTO Card (Card\_Id, Primary\_Method\_Id, Expiry\_Date, Card\_Holder\_Name, CVV)

VALUES (1, 1, TO\_DATE('2025-12-31', 'YYYY-MM-DD'), 'John Smith', '123');

INSERT INTO Card (Card\_Id, Primary\_Method\_Id, Expiry\_Date, Card\_Holder\_Name, CVV)

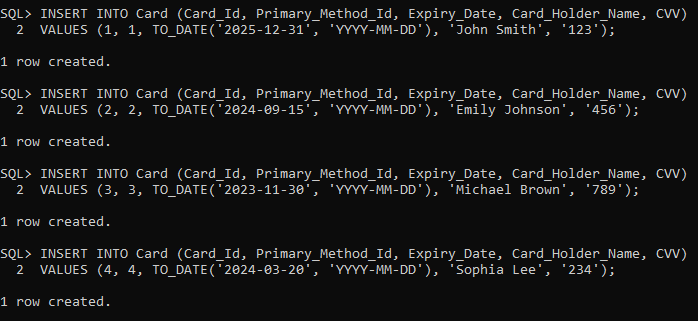
VALUES (2, 2, TO\_DATE('2024-09-15', 'YYYY-MM-DD'), 'Emily Johnson', '456');

INSERT INTO Card (Card\_Id, Primary\_Method\_Id, Expiry\_Date, Card\_Holder\_Name, CVV)

VALUES (3, 3, TO\_DATE('2023-11-30', 'YYYY-MM-DD'), 'Michael Brown', '789');

INSERT INTO Card (Card\_Id, Primary\_Method\_Id, Expiry\_Date, Card\_Holder\_Name, CVV)

VALUES (4, 4, TO\_DATE('2024-03-20', 'YYYY-MM-DD'), 'Sophia Lee', '234');



*Figure 7.14*

1. **Inserting Data Into The "Wallet" Table:**

INSERT INTO Wallet (Wallet\_Id, Driver\_Id, User\_Balance)

VALUES (1, 1, 500.00);

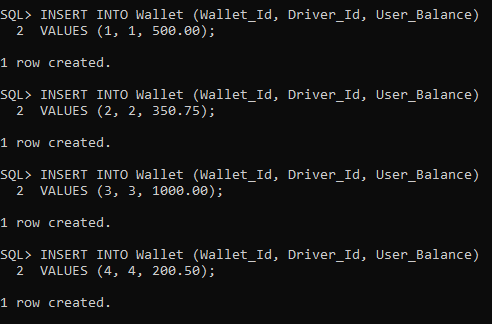
INSERT INTO Wallet (Wallet\_Id, Driver\_Id, User\_Balance)

VALUES (2, 2, 350.75);

INSERT INTO Wallet (Wallet\_Id, Driver\_Id, User\_Balance)

VALUES (3, 3, 1000.00);

INSERT INTO Wallet (Wallet\_Id, Driver\_Id, User\_Balance)

VALUES (4, 4, 20

*Figure 7.15*

# *Query*

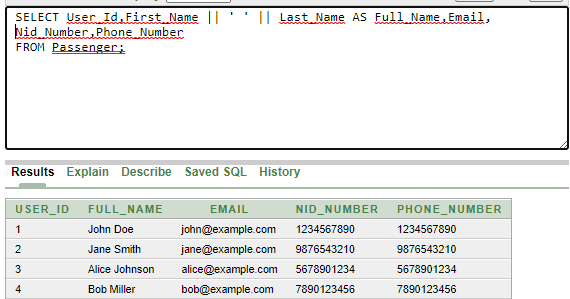
Several distinct types of intricate queries have been judiciously executed on the database schema to thoroughly assess its operational functionality. The subsequent section presents these queries, accompanied by the outcomes they have yielded, providing a comprehensive overview of the system's performance.

1. **Single row function:** Executing a singular row function query within the database to extract comprehensive user information, encompassing their full name, email address, national identification number (NID), and contact phone number.

**Example:**

SELECT User\_Id,First\_Name || ' ' || Last\_Name

AS Full\_Name,Email, Nid\_Number,Phone\_Number

FROM Passenger;

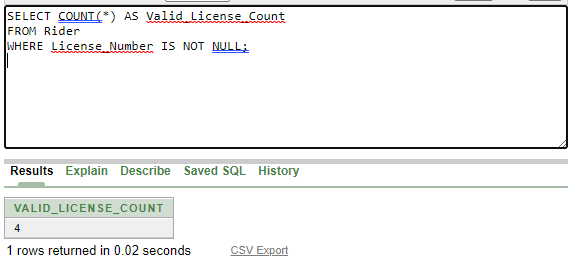
*Figure 8.1*

1. **Aggregate Function:** Employing aggregate functions, the administrator gains the valuable ability to query the database effectively. This empowers them to sift through user data, identify law-abiding individuals with valid licenses, and consequently sift out riders who might be contravening legal norms. The judicious use of aggregate functions enhances the system's capacity for insightful decision-making and regulatory compliance.

**Example:**

SELECT COUNT(\*) AS Valid\_License\_Count

FROM Rider

WHERE License\_Number IS NOT NULL;

*Figure 8.2*

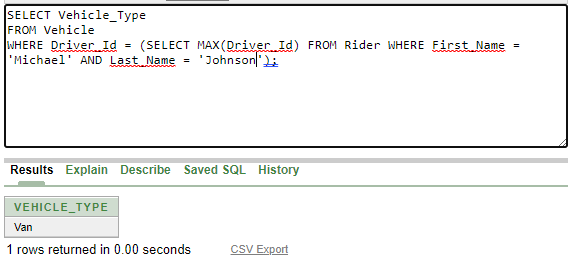
1. **Single Row Sub-Query:** To aid in the identification of a particular driver's vehicle, the administrator can make use of the single-row subquery function.

**Example:**

SELECT Vehicle\_Type

FROM Vehicle

WHERE Driver\_Id = (SELECT MAX(Driver\_Id) FROM Rider WHERE First\_Name = 'Michael' AND Last\_Name = 'Johnson');



*Figure 8.3*

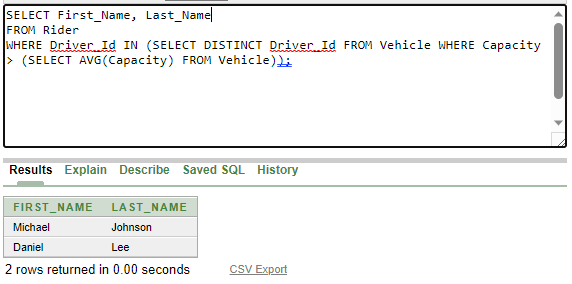
1. **Multiple Row Sub-Query:** Through the utilization of multiple-row subqueries, users are enabled to conduct searches for available drivers possessing vehicles with the capacity to accommodate multiple passengers simultaneously. This functionality enhances user convenience by providing a means to identify suitable transportation options aligned with varying passenger requirements.

**Example:**

SELECT First\_Name, Last\_Name

FROM Rider

WHERE Driver\_Id IN (SELECT DISTINCT Driver\_Id FROM Vehicle WHERE Capacity > (SELECT AVG(Capacity) FROM Vehicle));



*Figure 8.4*

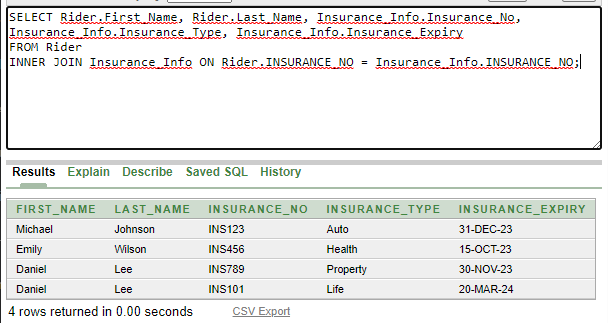
1. **Inner Join:** The utilization of an inner join operation facilitates the amalgamation of two interrelated tables. This capability enables riders to ascertain their insurance type and expiry date by executing a join operation between the driver table and the insurance table.

**Example:**

SELECT Rider.First\_Name, Rider.Last\_Name, Insurance\_Info.Insurance\_No, Insurance\_Info.Insurance\_Type, Insurance\_Info.Insurance\_Expiry

FROM Rider

INNER JOIN Insurance\_Info ON Rider.INSURANCE\_NO = Insurance\_Info.INSURANCE\_NO;



*Figure 8.5*

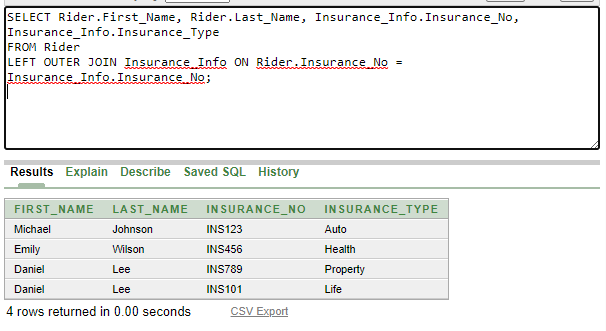
1. **Outer Joining:** Through the application of an outer join operation on interconnected tables, information retrieval is enabled. As an illustration, the display of riders' insurance details aligned with their other pertinent information can be achieved through the utilization of outer joins.

**Example:**

**SELECT Rider.First\_Name, Rider.Last\_Name, Insurance\_Info.Insurance\_No, Insurance\_Info.Insurance\_Type**

**FROM Rider**

**LEFT OUTER JOIN Insurance\_Info ON Rider.Insurance\_No = Insurance\_Info.Insurance\_No;**



*Figure 8.6*

1. **Self Join:** A self-join operation affords the creation of distinct virtual instances of a single table, each endowed with a unique alias. This approach facilitates the establishment of relationships between rows within the confines of the same table.

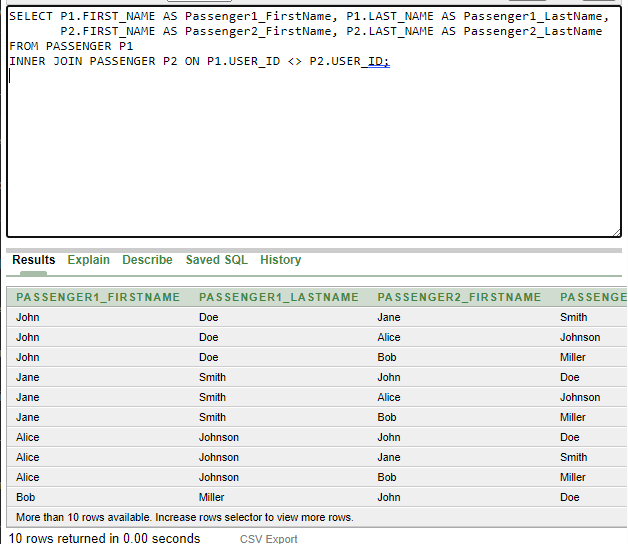
**Example:**

**SELECT P1.FIRST\_NAME AS Passenger1\_FirstName, P1.LAST\_NAME AS Passenger1\_LastName,**

**P2.FIRST\_NAME AS Passenger2\_FirstName, P2.LAST\_NAME AS Passenger2\_LastName**

**FROM PASSENGER P1**

**INNER JOIN PASSENGER P2 ON P1.USER\_ID <> P2.USER\_ID;**



*Figure 8.7*

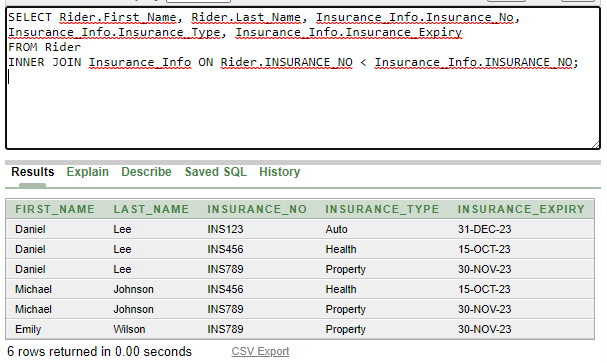
1. **Non Equi Join:** A non-equi join introduces join conditions that encompass operators beyond the equality operator (=). In lieu of confining comparisons to equality, non-equi joins encompass varied conditions such as greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=). An apt application of a non-equi join query involves seeking combinations of rider and insurance information, specifically where the rider's insurance number is of lesser value than the corresponding insurance number within the "Insurance\_Info" table.

**Example:**

**SELECT Rider.First\_Name, Rider.Last\_Name, Insurance\_Info.Insurance\_No, Insurance\_Info.Insurance\_Type, Insurance\_Info.Insurance\_Expiry**

**FROM Rider**

**INNER JOIN Insurance\_Info ON Rider.INSURANCE\_NO < Insurance\_Info.INSURANCE\_NO;**

****

*Figure 8.8*

# *Conclusion*

In summation, the "AIUB RIDEZ" ride-sharing database application schema demonstrates a meticulous and comprehensive approach to address the complex requirements of modern transportation services. The architecture encompasses a well-structured web of entities, relationships, and attributes that cater to various stakeholders, including users, drivers, and administrators. By implementing normalization techniques, the schema ensures data integrity and minimizes the risk of anomalies, fostering a dependable environment for data manipulation. The seamless integration of features like ride history, payment methods, and vehicle information reflects the commitment to offering a seamless user experience, while the utilization of various join types showcases the versatility and power of relational database management systems. Overall, the "AIUB RIDEZ" schema stands as a robust foundation for a sophisticated and user-friendly ride-sharing application that is poised to enhance urban mobility and redefine transportation convenience.