**Assignment**

DATABASE DESIGN AND DEVELOPMENT CSE4005

**Name:** Jayasuriya Arachchige Don Prabath Udayanga

**Student ID:** CL/HDCSE/CMU/128/14

**Module:** Database Design and Development - CSE4005

**Batch:** 128

**Course:**Higher Diploma in Computing and Software Engineering - CMU

**Date of Submission:** 17/04/2025

**Feedback Form**

**International College of Business & Technology**

**Module:** CSE4005 - Database Design and Development

**Student:**

**Assessor:**

**Assignment:** Database Management System for a Ride-Hailing Application

**Strong features of your work:**

**Areas for improvement:**

**Marks Awarded:**

**Assignment Cover Sheet**

|  |  |  |
| --- | --- | --- |
| **Qualification** | | **Module Number and Title** |
| HD in Computing and Software Engineering | | CSE4005 - **Database Design and Development** |
| **Student Name & No.** | | **Assessor** |
| Prabath Udayanga (CL/HDCSE/CMU/128/14) | |  |
| **Hand out date** | | **Submission Date** |
|  | | 17/04/2025 |
| **Assessment type**  WRIT1-Coursework | **Duration/Length of**  **Assessment Type**  3000 Words | **Weighting of Assessment**  100% |

|  |
| --- |
| **Learner declaration** |
| I Prabath Udayanga (CL/HDCSE/CMU/128/14) certify that the work submitted for this assignment is my own and research sources are fully acknowledged. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Marks Awarded** | | | |
| First assessor | |  | |
| IV marks | |  | |
| Agreed grade | |  | |
| Signature of the assessor |  | Date |  |

A screenshot of a cell phone

AI-generated content may be incorrect.

**Acknowledgment**

I would like to extend my heartfelt gratitude to Mr. Chanuka for his invaluable guidance and support throughout the Database Design and Development module. His clear explanations, practical insights, and engaging teaching approach have greatly enhanced my understanding of database concepts, design principles, and implementation techniques.

I truly appreciate the effort and dedication he has put into making complex topics more accessible and ensuring that we develop both theoretical knowledge and practical skills essential for database development.

Thank you for your encouragement and for sharing your expertise with us.

Table of Contents

Introduction.………………………………………………………………………… 5

Task 1 …………………………………………………………………......…………5

Task 2 ………………………………….………………………………………....….8

Task 3 ….…………………………………………………………………………….9

Task 4 ….……………………………………………………………………………11

Task 5 ….……………………………………………………………………………26

Task 6 ….……………………………………………………………………………29

Conclusion ………………………………………….………………………………44

References ……...…...………………………………………………………………45

**Introduction**

Urban transportation has undergone a remarkable change because ride-hailing applications developed rapidly. Infor Pvt Ltd needs a solid relational database structure for their new ride-hailing app so we will carry out this database system design and development. The database contains a structure which enables managing extensive amounts of driver, user, ride and payment information and delivers secure functionality with sustainable performance. The database implementation includes creating an ERD as well as performing normalization followed by SQL development and query execution before extensive testing for a complete optimized system.

**Task 1: Data model, comparing different data models and why older data models are being replaced by new data models**

**What is Data Model?**

A data model represents a conceptual system which determines how data structures into the physical storage of a database and controls data treatment operations. The data model specifies relationships between elements while maintaining rules which defend both integrity and consistency of data.

There are 3 main types of data models,

1. **Hierarchical Data Model**
2. **Network Data Model**
3. **Relational Data Model**

**Comparison of Data Models**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Hierarchical Model** | **Network Model** | **Relational Model** |
| Structure | Tree-like (Parent-Child) | Graph-like (Many-to-Many) | Table-based (Rows & Columns) |
| Relationships | One-to-many (strict hierarchy) | Many-to-many (more flexible) | Many-to-many (through keys & joins) |
| Data Retrieval | Requires traversing from root to child | More flexible but still requires complex navigation | Uses SQL, allowing flexible queries |
| Complexity | High (complex to modify and maintain) | High (complex pointers and links) | Moderate (structured yet flexible) |
| Flexibility | Rigid (difficult to restructure) | More flexible than hierarchical but still challenging | Highly flexible |
| Scalability | Poor (not suitable for large data) | Better than hierarchical but complex | Highly scalable |
| Ease of Use | Difficult (requires hierarchical navigation) | Difficult (pointer-based structure) | Easy (uses SQL and relationships) |
| Storage Efficiency | High (redundant data minimized) | Moderate (some redundancy due to pointers) | Efficient (normalization removes redundancy) |

**Why Are Older Data Models Being Replaced?**

The relational model has displaced older data models such as hierarchical and network because of various reasons such as,

1. **Better Flexibility and Scalability** - The design of relational databases allows efficient processing of large and complex datasets which display multiple interrelated components. Lowered scalability emerges in older models because their structures maintain rigid definitions.
2. **Ease of Data Retrieval** - Users retrieve data efficiently through Structured Query Language (SQL) in the relational model whereas navigating complex tree or graph structures is not required.
3. **Normalization and Data Integrity** - The relational database normalization process reduces both storage needs and improves data consistency by eliminating unnecessary repetition of information.
4. **Better Maintenance and Modifications** - The modification process for relational databases stays simple due to foreign keys and joins while older models need extensive structural changes when relationships evolve.
5. **Business Adoption and Standardization** - Most modern applications (banking, e-commerce, ride-hailing apps) use relational databases like MySQL, PostgreSQL, and SQL Server because of their compatibility, security, and efficiency.

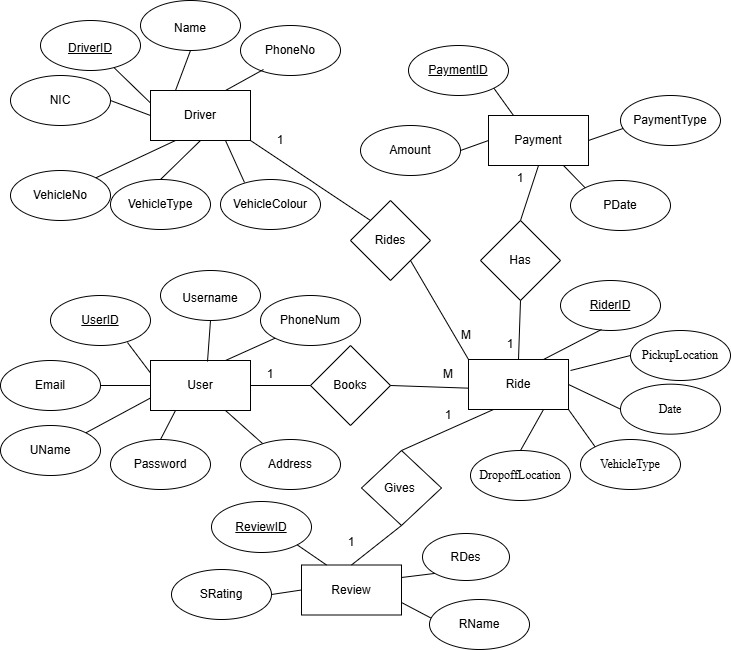
**Conclusion**

Modern data management applications have chosen the relational model instead of hierarchical and network models because it delivers flexibility and scalability and enables simple data access and manageability of complex interrelationships between data. Ride-hailing systems alongside other modern large-scale applications demand efficient interrelated data processing that hierarchical and network models cannot adequately provide due to their lack of adaptability.

**Task 2: Entity relationship diagram and identifying important keys and represent different types of attributes and relationships**

**Entities and their Attributes**

|  |  |
| --- | --- |
| **Entity** | **Attributes** |
| **Driver** | DriverID, Name, PhoneNo, NIC, VehicleNo, VehicleType, VehicleColour |
| **User** | UserID, Username, UName, PhoneNum, Email, Password, Address |
| **Ride** | RiderID, PickupLocation, DropoffLocation, Date, VehicleType |
| **Payment** | PaymentID, PaymentType, Amount, PDate |
| **Review** | ReviewID, SRating, RDes, RName |

**ER Diagram**

**Task 3: Relational Schema for ERD**

**Entities and Attributes**

**1. User Table**

User (UserID, Username, UName, PhoneNum, Password, Email, AddressLineOne, AddressLineTwo)

**2. Driver Table**

Driver (DriverID, Name, NIC, PhoneNo, VehicleNo, VehicleTypeID, VehicleColourID)

**3. Ride Table**

Ride (RideID, UserID [FK], DriverID [FK], PickupLocation, DropoffLocation, Date)

**4. Payment Table**

Payment (PaymentID , RideID [FK], PaymentTypeID, Amount, PDate)

**5. Review Table**

Review (ReviewID, RideID [FK], SRating, RDes, RName)

**Normalization Process (to 3NF)**

**1.) 0NF to 1NF**

All entities are in **1NF** because of atomic attributes, and there are no repeating groups.

**2.) 1NF to 2NF**

Eliminating **partial dependencies**. Most tables have single attribute Primary keys, so **2NF is satisfied**.

**3.) 2NF to 3NF**

Eliminating **transitive dependencies**. All entities are satisfied with **3NF**.

**First Normal Form (1NF):**

A relation is in **1NF** if:

* It contains only **atomic values**.
* Each attribute contains only a **single value**, not a set or list.
* Each record is **unique**.

**Second Normal Form (2NF):**

A relation is in **2NF** if:

* It is in **1NF**.
* And **every non-prime attribute** is **fully functionally dependent** on the **entire primary key**, not just part of it.

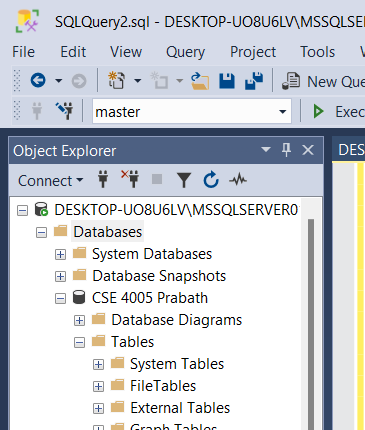
**Third Normal Form (3NF):**

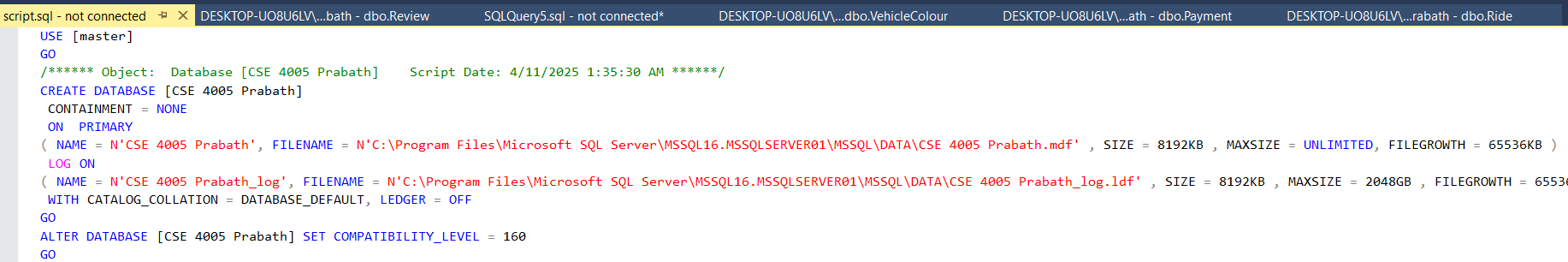
A relation is in **3NF** if,

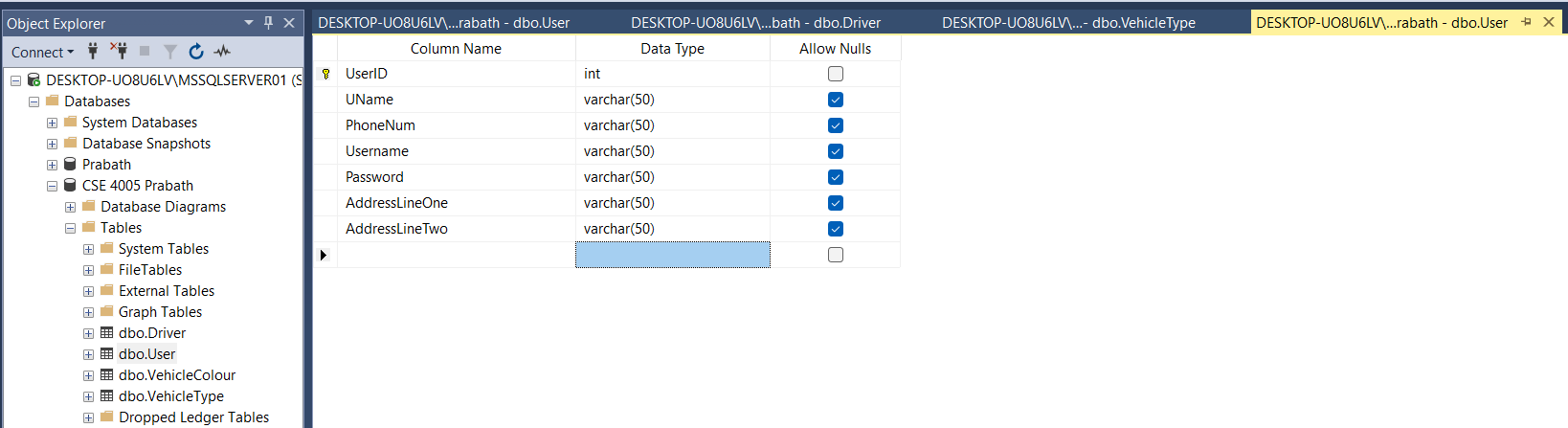
* It is in **2NF**.
* And there are **no transitive dependencies**

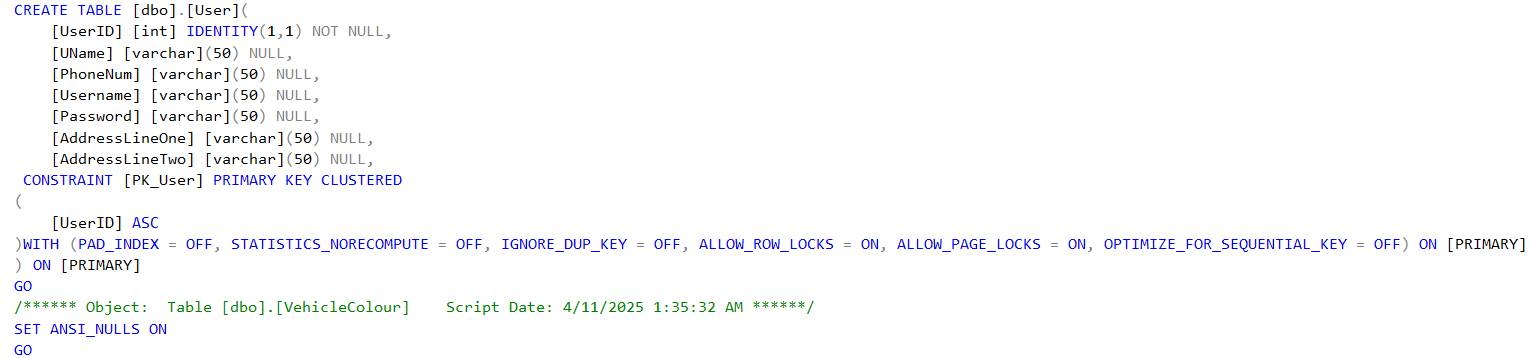
**Task 4: The database using SQL server**

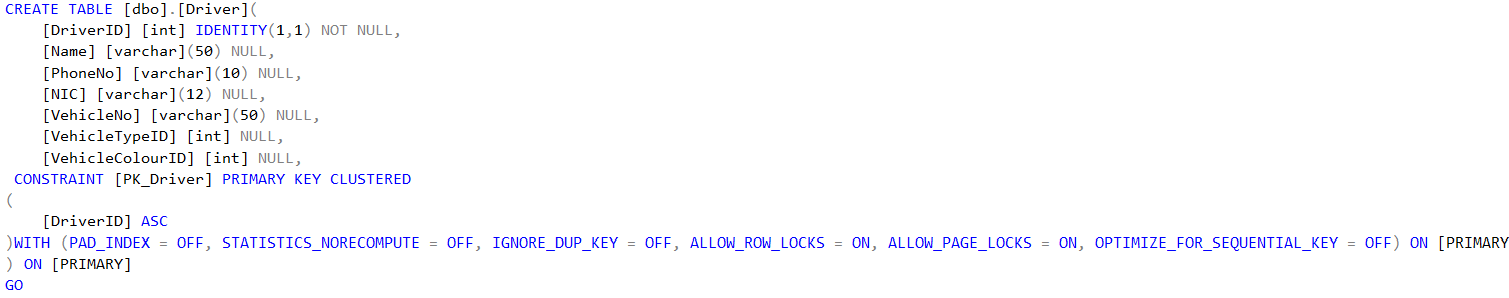
1. **Creating a new Database ‘CSE 4005 Prabath’**



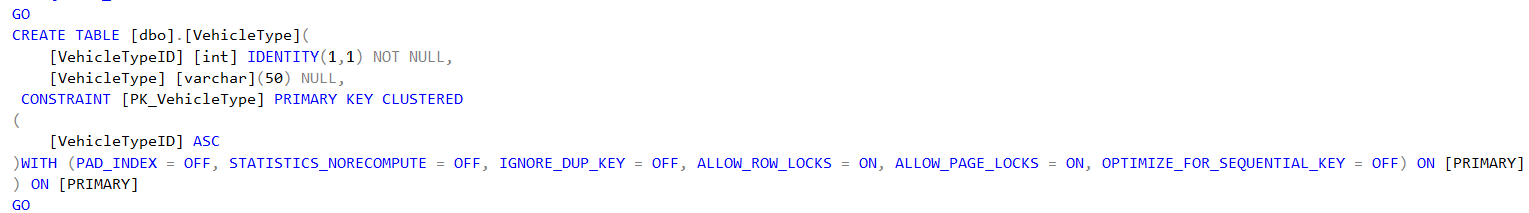
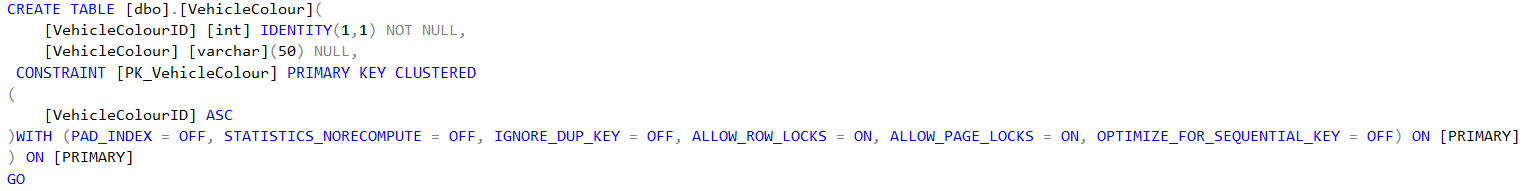


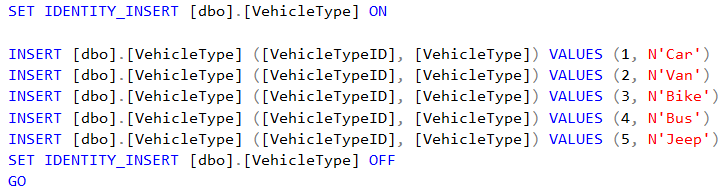
1. **Creating User Table**

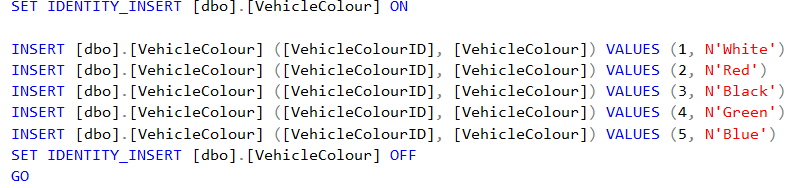


1. **A screenshot of a computer

   AI-generated content may be incorrect.Creating Driver Table**
2. **A screenshot of a computer

   AI-generated content may be incorrect.Creating VehicleType Table**
3. **A screenshot of a computer

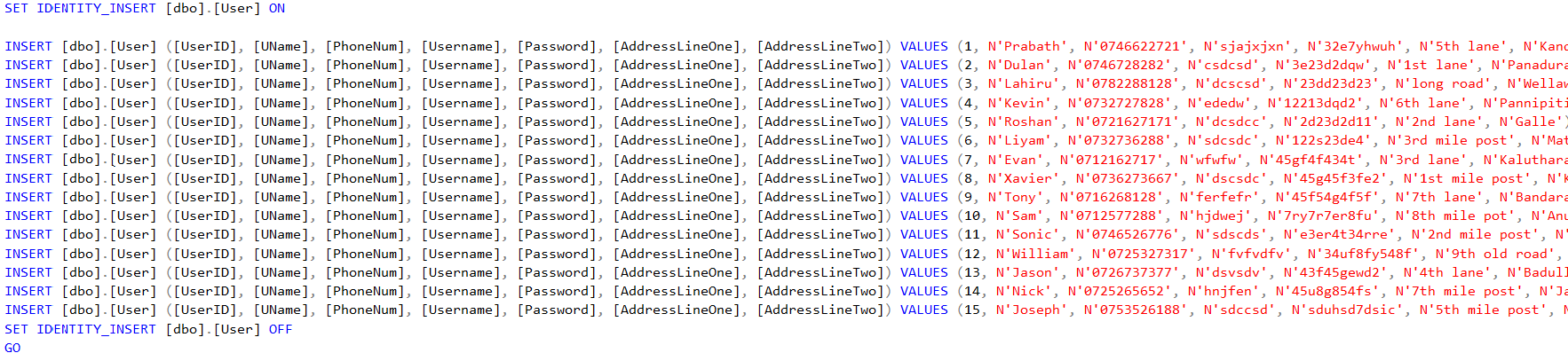
   AI-generated content may be incorrect.Creating VehicleColour Table**
4. **A computer screen shot of a computer

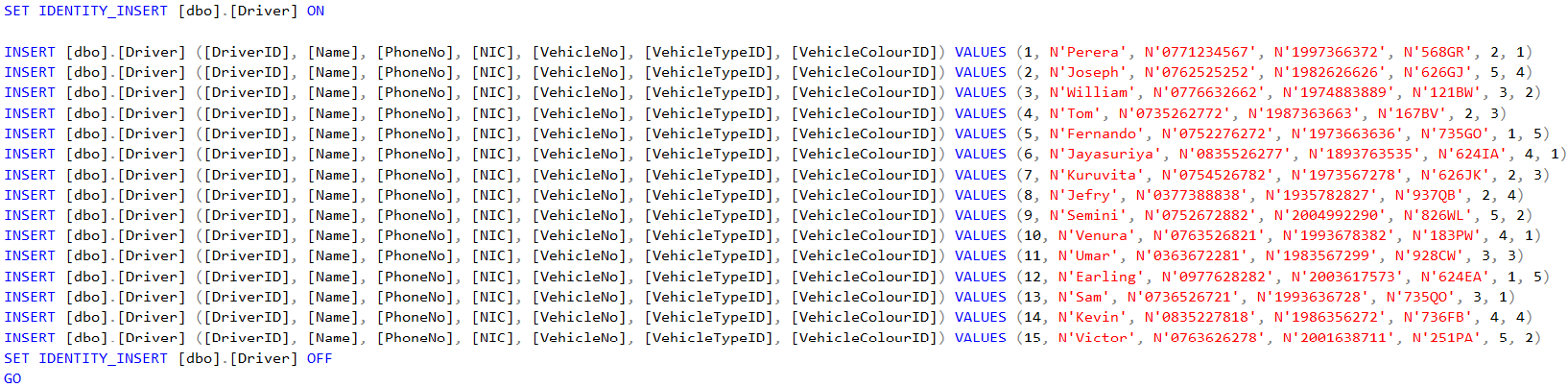
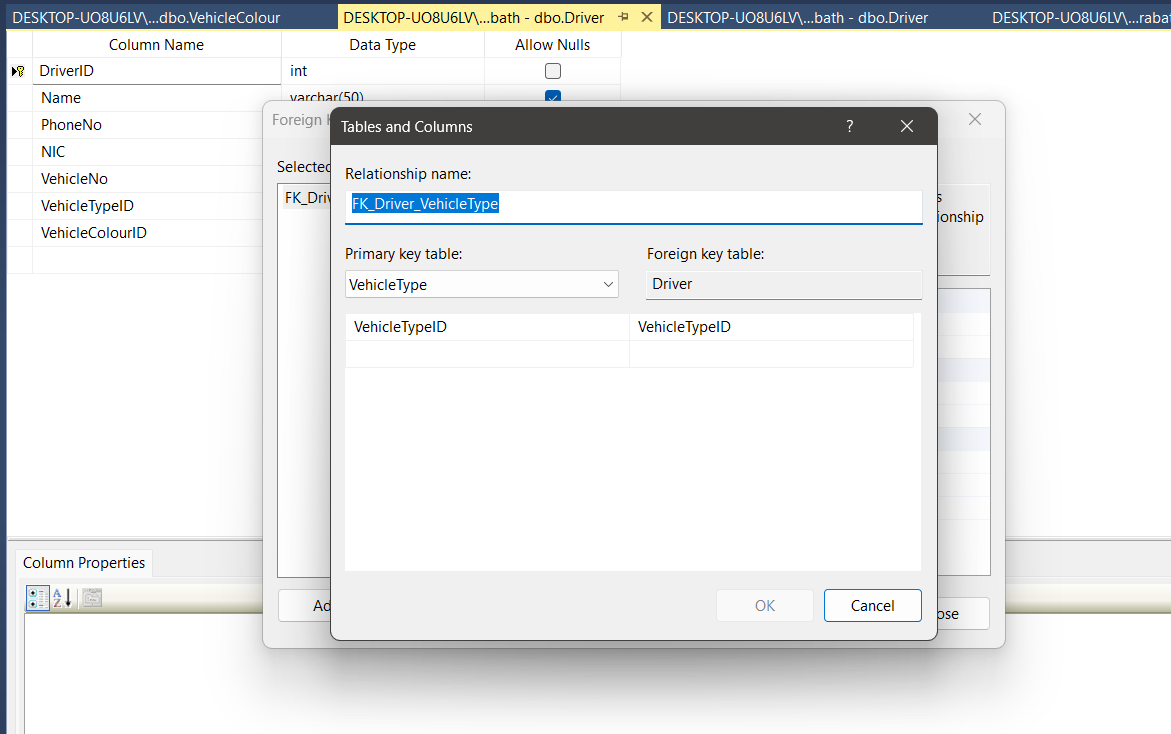
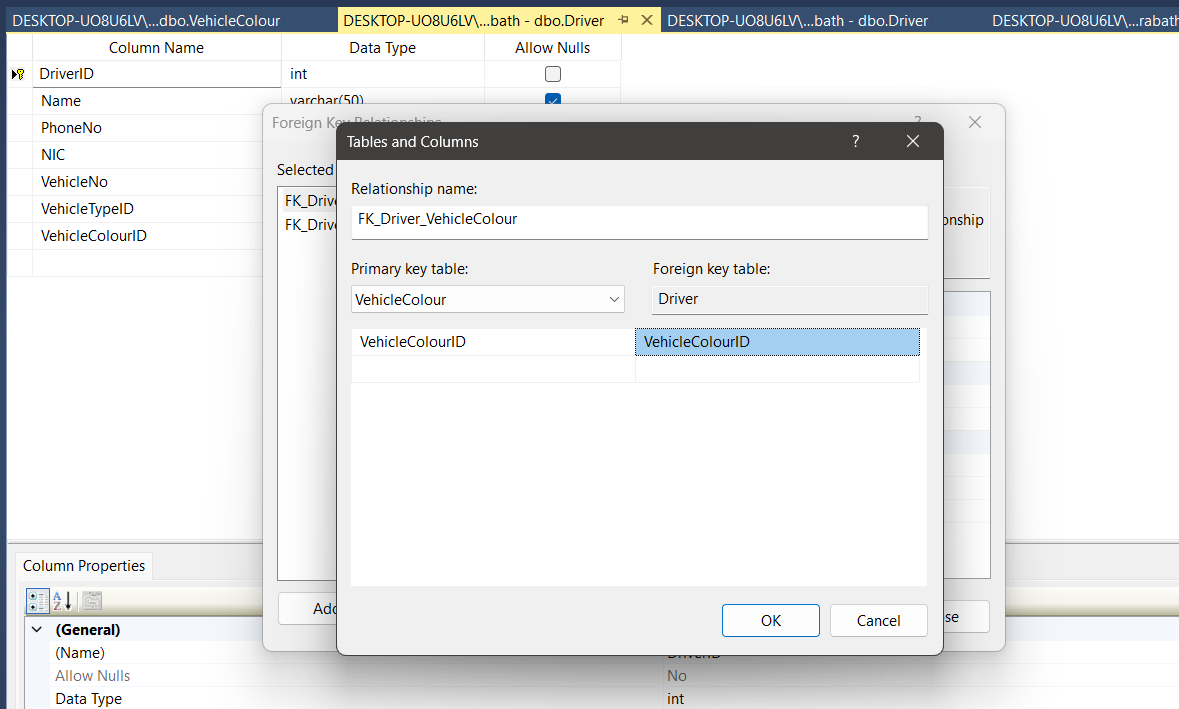
   AI-generated content may be incorrect.Inserting Data into VehicleType Table**
5. **A computer screen shot of a computer

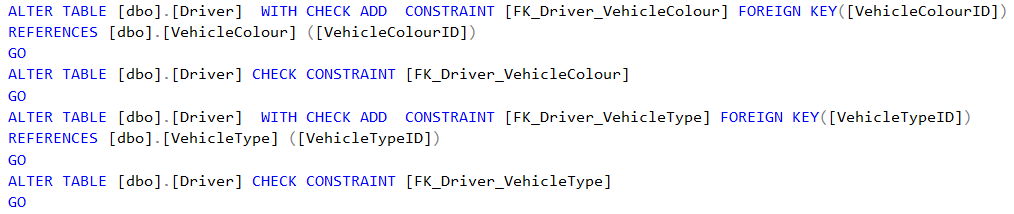
   AI-generated content may be incorrect.Inserting Data into VehicleColour Table**

**A screenshot of a computer

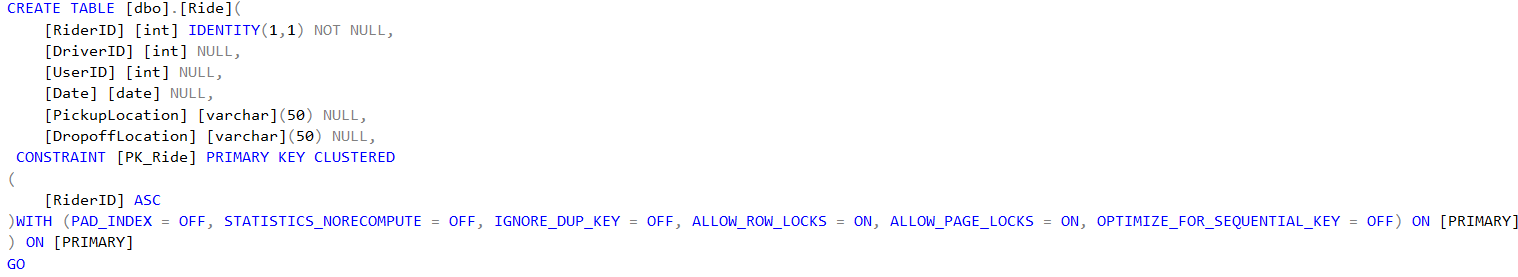
AI-generated content may be incorrect.**

1. **Inserting Data into Driver Table**
2. A screenshot of a computer

   AI-generated content may be incorrect.**Inserting Data into User Table**
3. ** Foreign Key Relationships Between Driver, VehicleType, and VehicleColour Tables**

A screenshot of a computer

AI-generated content may be incorrect.

1. A screenshot of a computer

   AI-generated content may be incorrect. **Create Ride Table**
2. **Foreign Key Relationships Between Ride, Driver, and User Tables**

**A screenshot of a computer

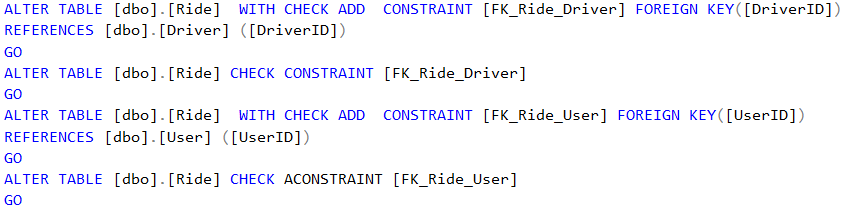
AI-generated content may be incorrect.**

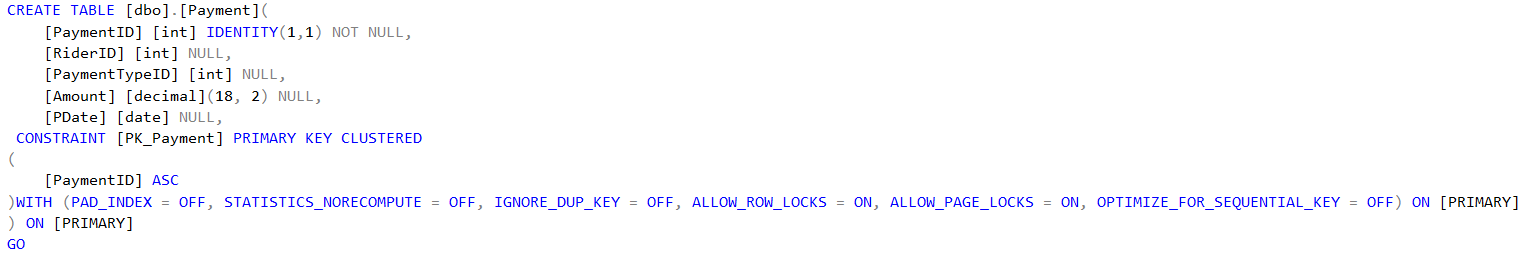
A screenshot of a computer

AI-generated content may be incorrect.

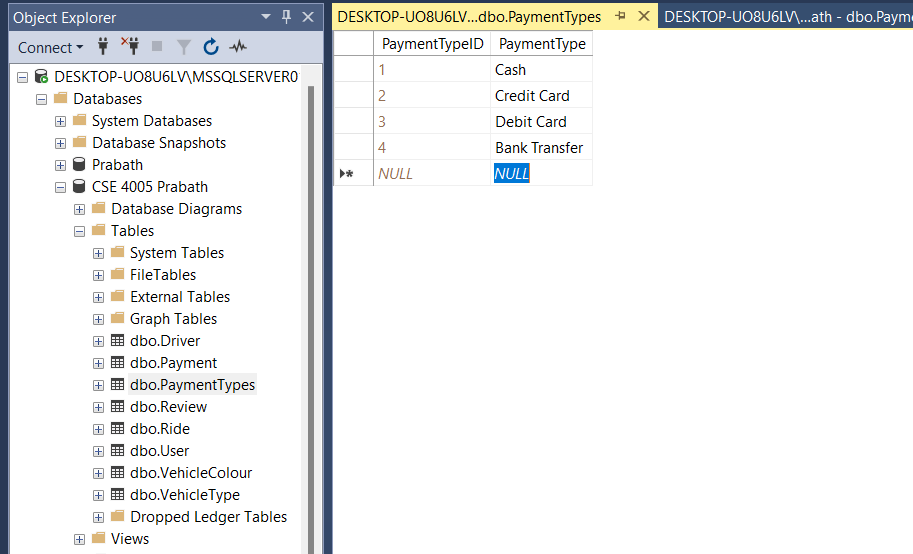
A screenshot of a computer

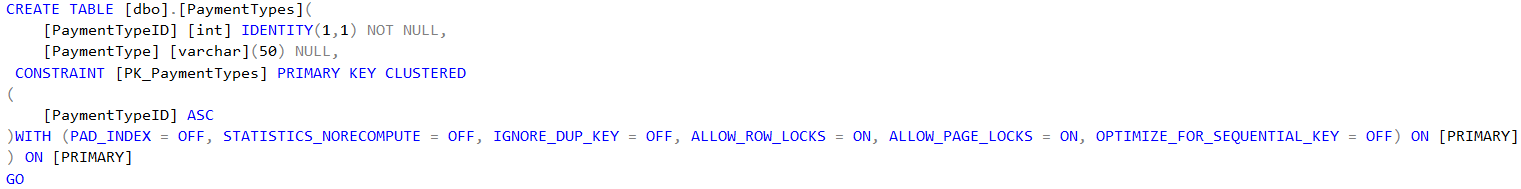
AI-generated content may be incorrect.

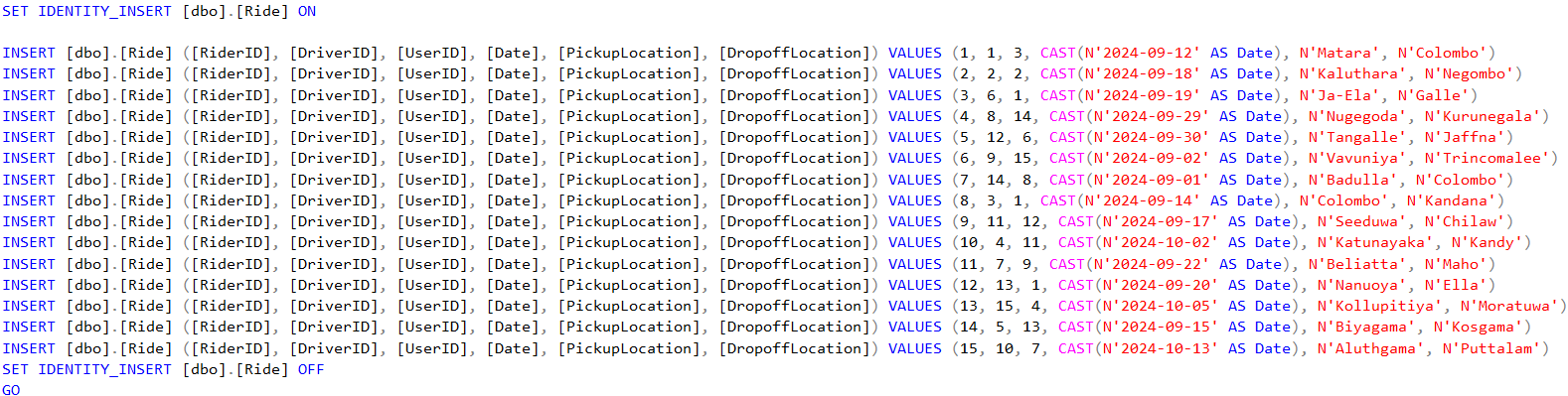


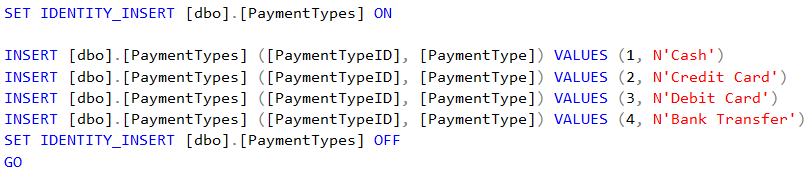
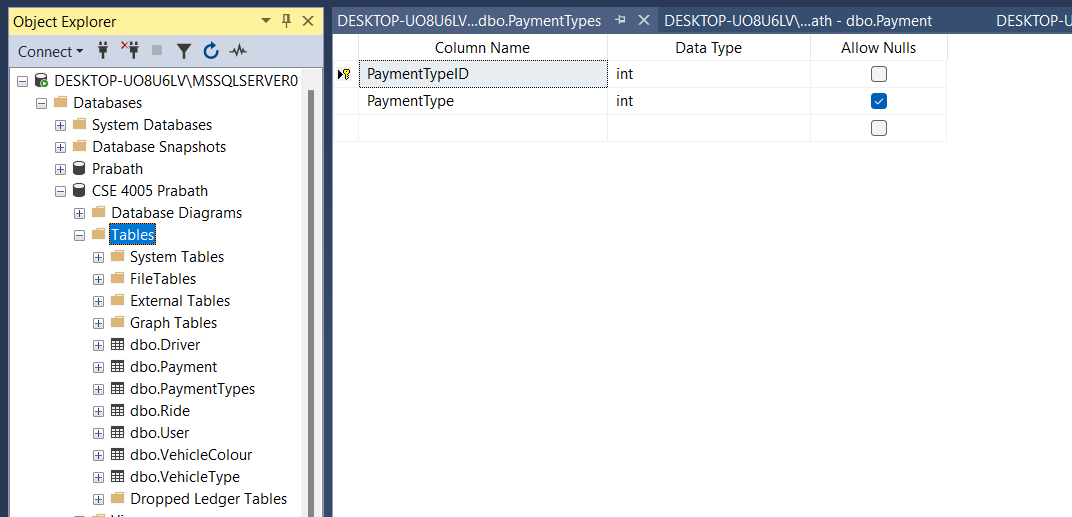
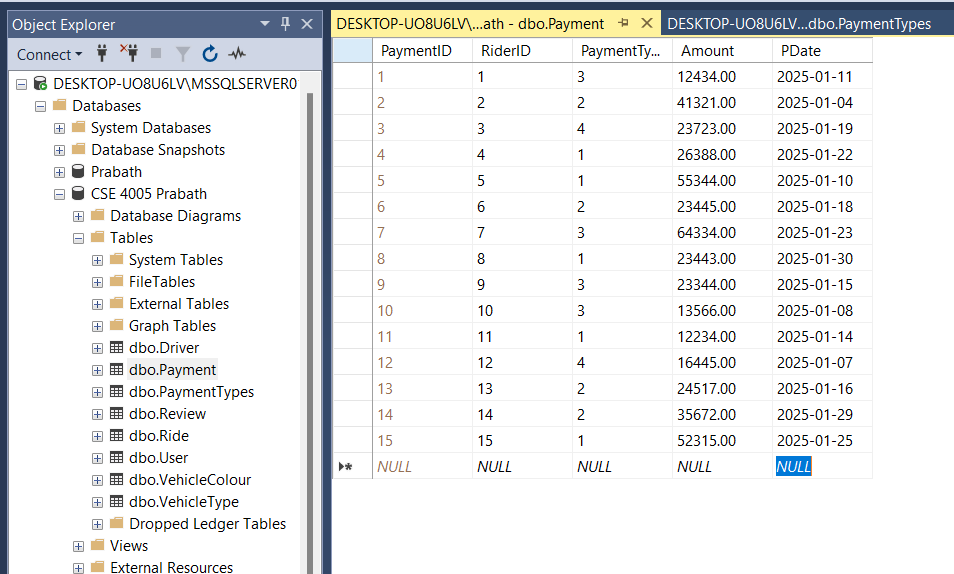
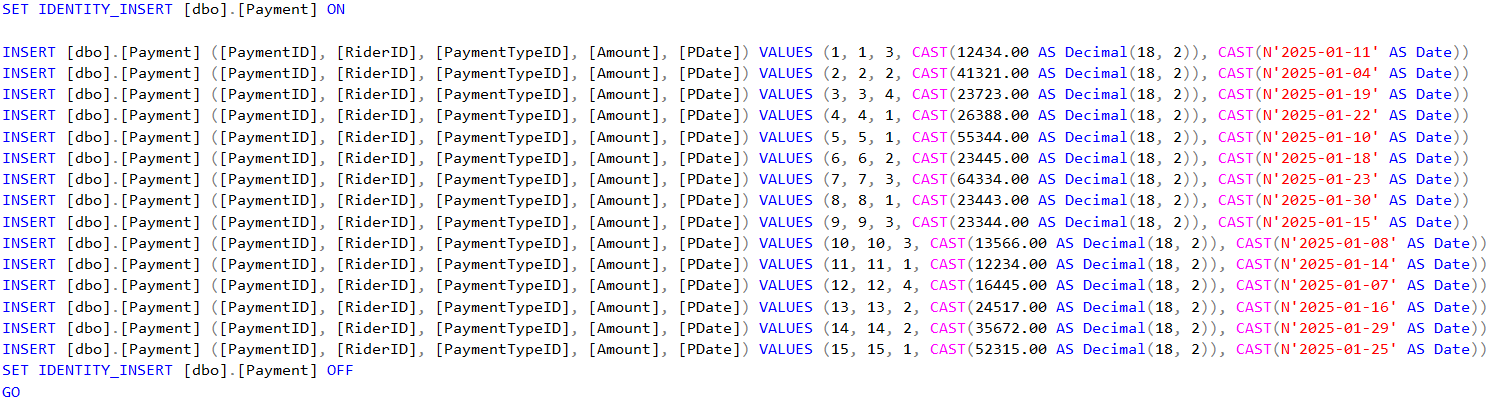
1. **A screenshot of a computer

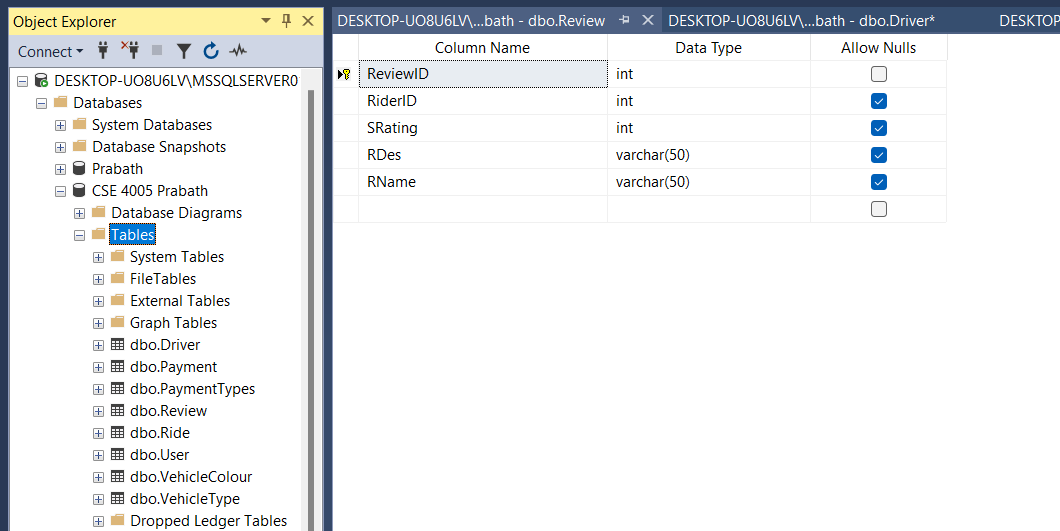
   AI-generated content may be incorrect. Create Payment Table**
2. **Create PaymentTypes Table**

****

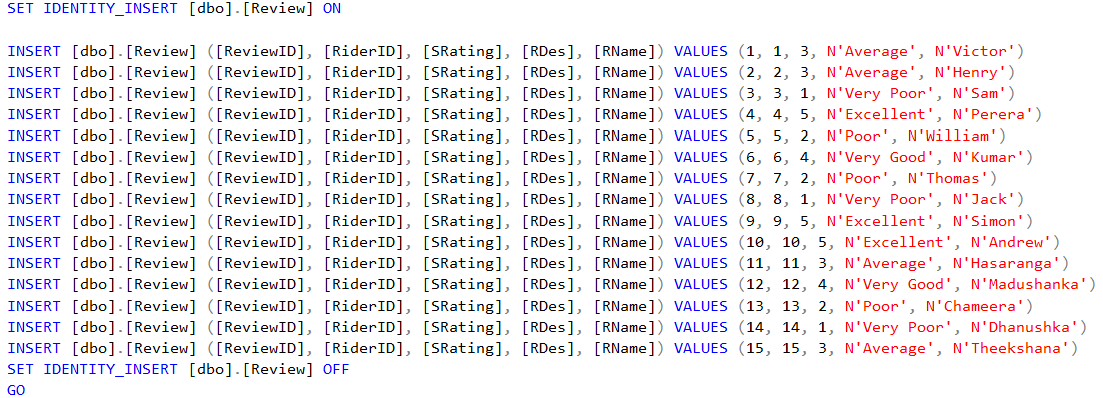


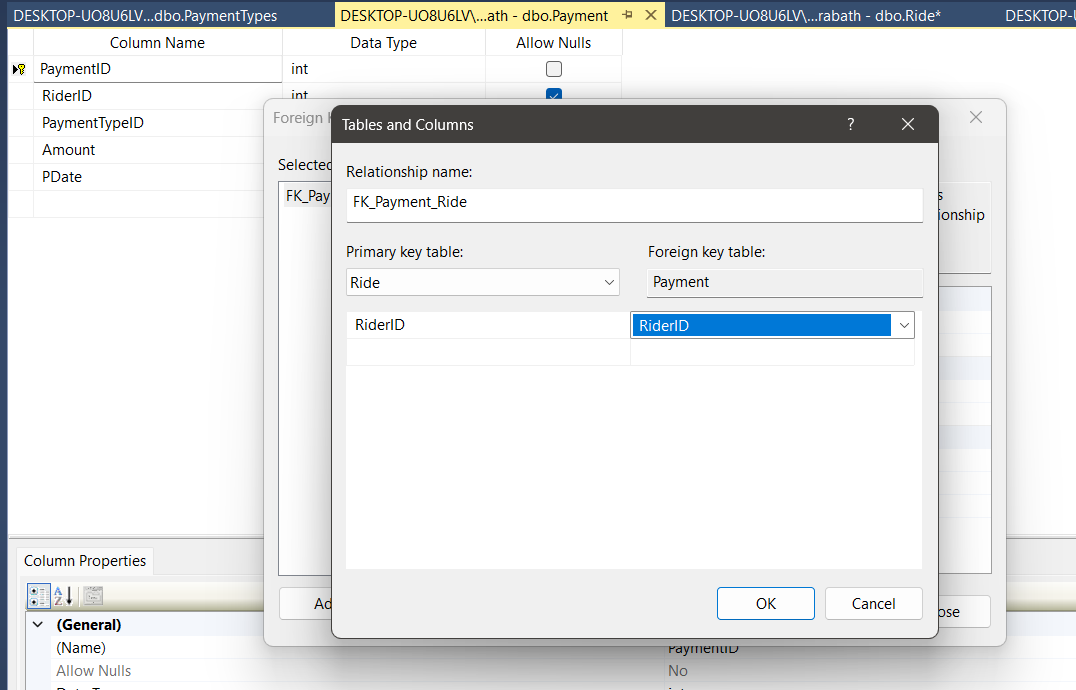
1. ****A screenshot of a computer

   AI-generated content may be incorrect. **Inserting Data into Ride Table**
2. ** Inserting Data into PaymentTypes Table**
3.  **Inserting Data into Payment Table**
4. A white background with blue text

   AI-generated content may be incorrect. **Create Review Table**
5. A screenshot of a computer

   AI-generated content may be incorrect. **Inserting Data into Review Table**



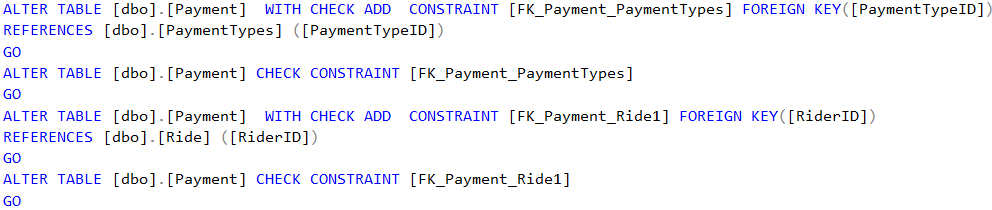
1.  **Foreign Key Relationships Between Payment, Ride and PaymentTypes Tables**

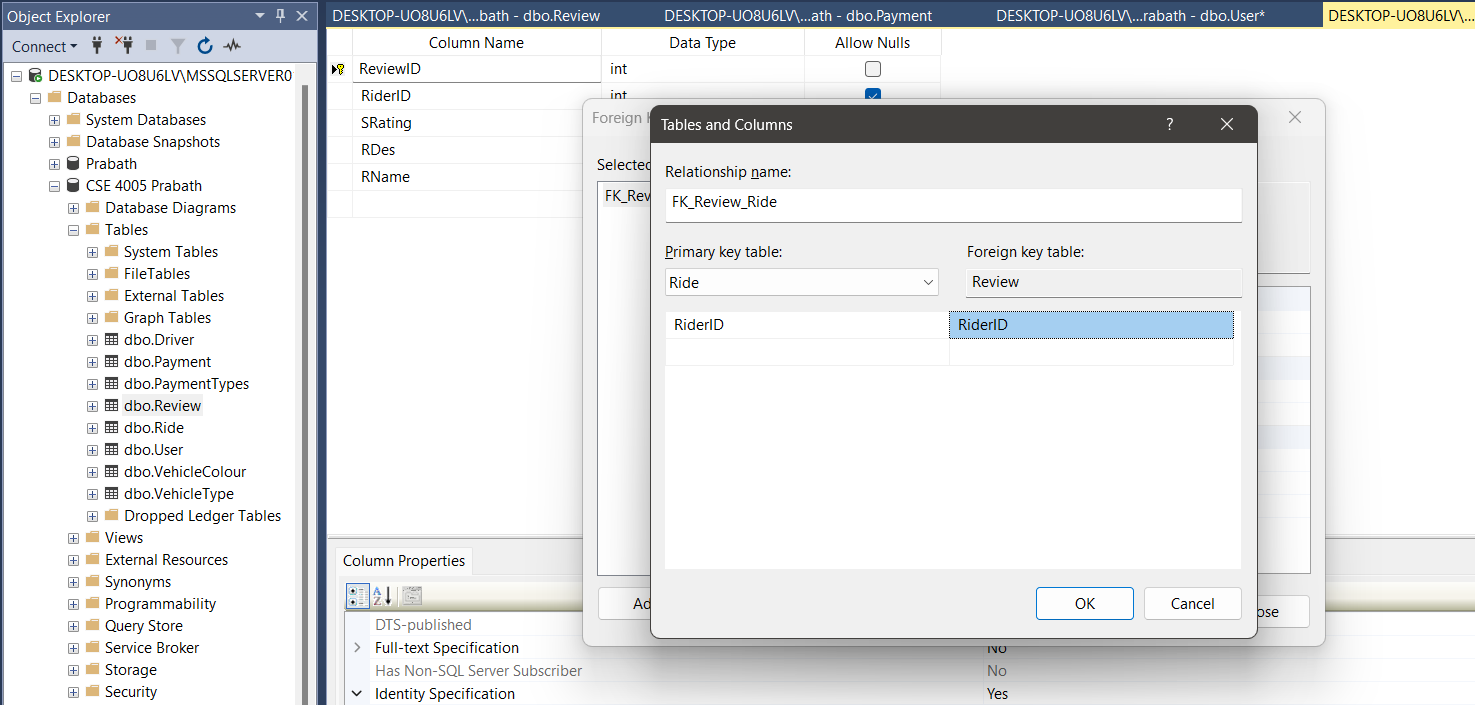
A screenshot of a computer

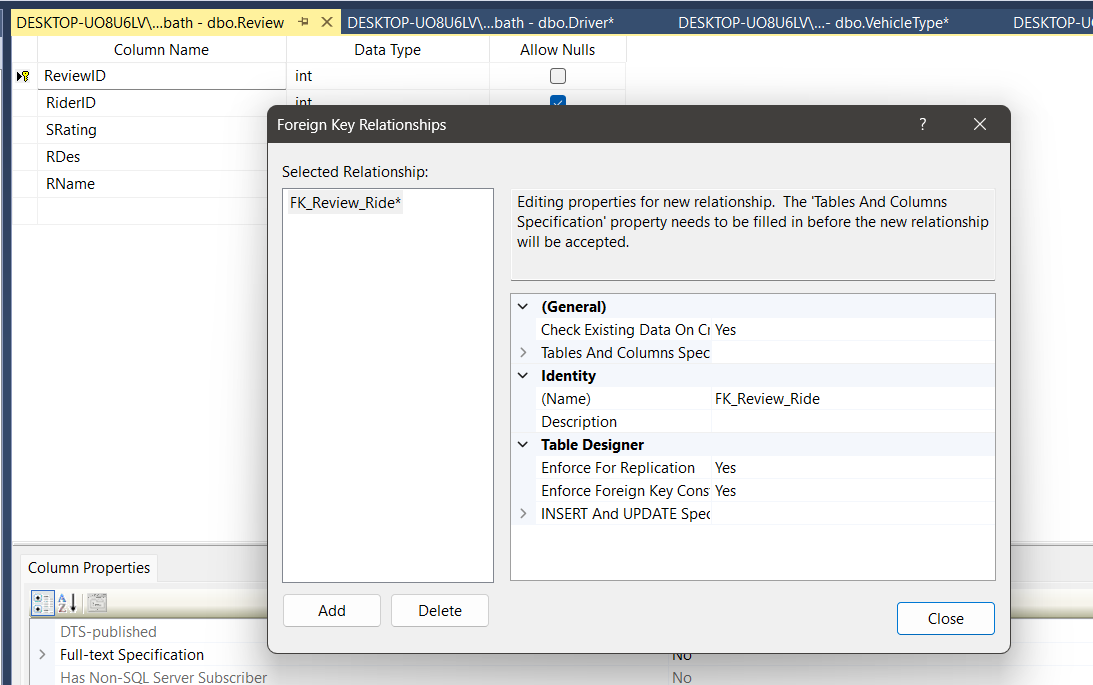
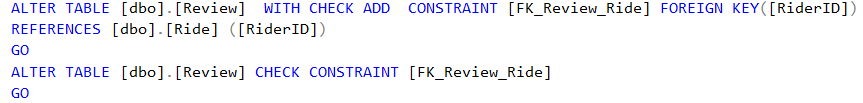
AI-generated content may be incorrect.

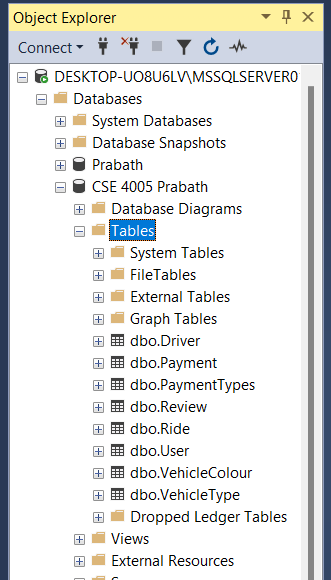
A screenshot of a computer

AI-generated content may be incorrect.



1.  **Foreign Key Relationships Between Review and Ride Tables**

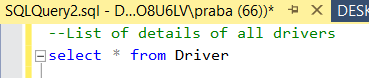


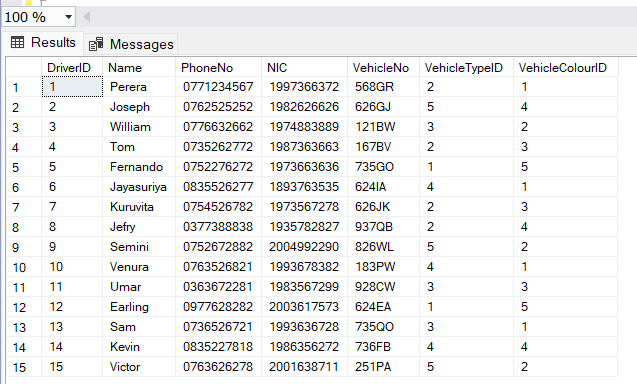
1.  **Object Explorer at the end**
2. **A screenshot of a computer

   AI-generated content may be incorrect. Database Diagram**

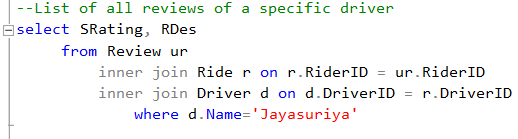
**Task 5: SQL Queries**

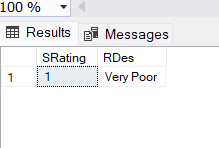
* 1. **List of details of all drivers**

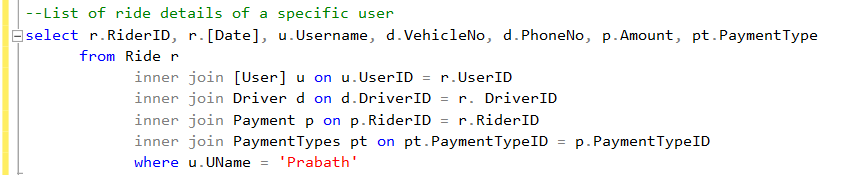


**Output:**

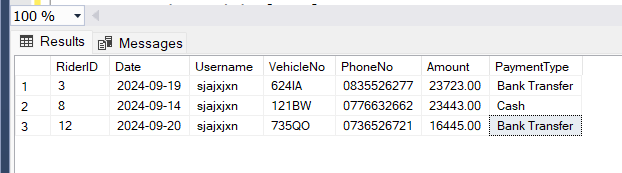
* 1. **List of all reviews of a specific driver**

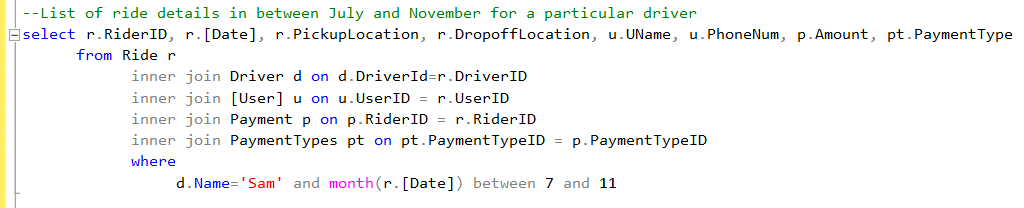


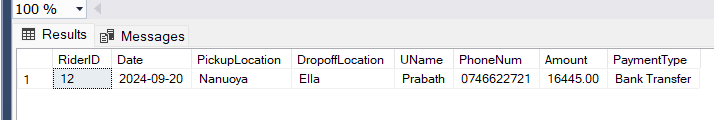
 **Output:**

* 1. **List of ride details of a specific user**

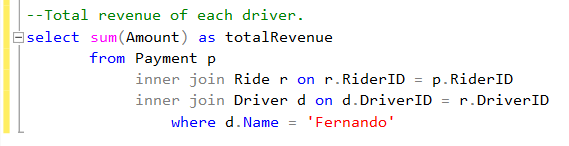
**Output:**

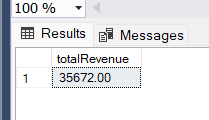


* 1. **List of ride details between July and November for a particular driver.**

** Output:**

* 1. **Total revenue of each driver.**



 **Output:**

**Task 6: Test plan, Test strategy and Test cases**

**1. Objective**

Ensure the ride-hailing database system performs accurately, securely, and efficiently. The goal is to verify data integrity, relational consistency, query accuracy, and conformance to requirements.

**2. Scope**

Includes testing of:

* Database schema (tables, keys, constraints)
* Data integrity and referential relationships
* SQL query outputs
* CRUD operations
* Edge cases, input validations, and performance

**3. Resources**

* Microsoft SQL Server
* SQL Server Management Studio (SSMS)
* Sample test data
* Test environment

**Test Strategy**

**Types of Testing**

* 1. Unit Testing
  2. Integration Testing
  3. Functional Testing
  4. Regression Testing
  5. Security Testing
  6. Performance Testing

**Testing Techniques**

* + Equivalence Partitioning
  + Boundary Value Analysis
  + Error Guessing

**Test Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Test Case Id** | **Test Case Name** | **Test Data** | **Expected Results** | **Test Result** |
| 1 | TC 1 | Insert valid driver data | INSERT INTO Driver (Name, PhoneNo, NIC, VehicleNo, VehicleTypeID, VehicleColourID) VALUES ('Amal', '0771234567', '123456789V', 'WP1234', 1, 2); | Row inserted successfully | Pass |
| 2 | TC 2 | Insert a Driver without a VehicleColourID | INSERT INTO Driver (Name, PhoneNo, NIC, VehicleNo, VehicleTypeID) VALUES ('Sunil', '0777777777', '917234567V', 'WP9999', 1); | Row inserted successfully | Pass |
| 3 | TC 3 | Insert a new Vehicle Type | INSERT INTO VehicleType (TypeName) VALUES ('Tuk Tuk'); | Row inserted successfully | Pass |
| 4 | TC 4 | Prevent inserting Ride with non-existing DriverID | INSERT INTO Ride (DriverID, UserID, PickupLocation, DropoffLocation, Date) VALUES (999, 1, 'Colombo', 'Kandy', '2025-04-10'); | Fails due to Foreign Key constraint | Pass |
| 5 | TC 5 | Check FK on delete | DELETE FROM [User] WHERE UserID = 1; | Fails due to FK constraint in Ride table | Pass |
| 6 | TC 6 | FK reference from PaymentType | INSERT INTO Payment (RiderID, PaymentTypeID, Amount, PDate) VALUES (1, 1, 2500, '2025-04-10'); | Row inserted if IDs are valid | Pass |
| 7 | TC 7 | Delete driver data | DELETE FROM Driver WHERE NIC = '123456789V'; | Row deleted successfully | Pass |
| 8 | TC 8 | Insert a new payment mode | INSERT INTO PaymentTypes (PaymentType) VALUES ('Free'); | Row inserted successfully | Pass |
| 9 | TC 9 | Insert a new vehicle Colour | INSERT INTO VehicleColour(VehicleColour) VALUES ('Brown'); | Row inserted successfully | Pass |
| 10 | TC 10 | Retrieve all Rides with INNER JOIN User & Driver | SELECT R.RiderID, U.UName, D.Name, R.PickupLocation FROM Ride R JOIN [User] U ON R.UserID = U.UserID JOIN Driver D ON R.DriverID = D.DriverID; | Joined rows shown | Pass |
| 11 | TC 11 | Check the average rating for a driver | SELECT D.DriverID, D.Name, AVG(Review.SRating) AS AvgRating FROM Review JOIN Ride R ON Review.RiderID = R.RiderID JOIN Driver D ON R.DriverID = D.DriverID GROUP BY D.DriverID, D.Name; | Avg shown per driver | Pass |
| 12 | TC 12 | Insert valid review | INSERT INTO Review (RiderID, SRating, RDes, RName) VALUES (1, 4, 'Poor', 'Nilu'); | Row added successfully | Pass |
| 13 | TC 13 | Fetch Rides by Date range | SELECT \* FROM Ride WHERE Date BETWEEN '2024-09-01' AND '2024-09-10'; | Only rides in that range | Pass |
| 14 | TC 14 | Validate FK from Payment to Ride | INSERT INTO Payment (RiderID, PaymentTypeID, Amount, PDate) VALUES (999, 1, 1200, '2025-04-11'); | Foreign key error | Pass |
| 15 | TC 15 | Attempt to delete PaymentType in use | DELETE FROM PaymentTypes WHERE PaymentTypeID = 5; | Row deleted successfully | Pass |
| 16 | TC 16 | Check case sensitivity on Username | SELECT \* FROM [User] WHERE Username = 'NICK'; | No results (case-sensitive) | Pass |
| 17 | TC 17 | Validate Username length constraint | INSERT INTO [User] (Username, UName, PhoneNum, Password, AddressLineOne, AddressLineTwo) VALUES ('xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx...', 'Kamal', '0712233637', 'camcbasc', 'nagoda', 'ragama'); | Fails (Username exceeds max length) | Pass |
| 18 | TC 18 | List Users with no Rides | SELECT \* FROM [User] WHERE UserID NOT IN (SELECT UserID FROM Ride); | Returns users with no rides | Pass |

**Test Cases**

**Test Case 1:**

|  |  |
| --- | --- |
| Test Case ID | 1 |
| Test Objectives | Insert valid driver data |
| Test Data | INSERT INTO Driver ([Name], PhoneNo, NIC, VehicleNo, VehicleTypeID, VehicleColourID) VALUES ( 'Amal', '0771234567', '123456789V', 'WP1234', 1, 2); |
| Expected Results | Row inserted successfully |
| Actual results | Row inserted successfully |
| Test Result | Pass |
| Screenshot |  |

**Test Case 2:**

|  |  |
| --- | --- |
| Test Case ID | 2 |
| Test Objectives | Insert a Driver without a VehicleColourID |
| Test Data | INSERT INTO Driver (Name, PhoneNo, NIC, VehicleNo, VehicleTypeID) VALUES ('Sunil', '0777777777', '917234567V', 'WP9999', 1); |
| Expected Results | Row inserted successfully |
| Actual results | Row inserted successfully |
| Test Result | Pass |
| Screenshot |  |

**Test Case 3:**

|  |  |
| --- | --- |
| Test Case ID | 3 |
| Test Objectives | Insert a new Vehicle Type |
| Test Data | INSERT INTO VehicleType (TypeName) VALUES ('Tuk Tuk'); |
| Expected Results | Row inserted successfully |
| Actual results | Row inserted successfully |
| Test Result | Pass |
| Screenshot |  |

**Test Case 4:**

|  |  |
| --- | --- |
| Test Case ID | 4 |
| Test Objectives | Prevent inserting Ride with non-existing DriverID |
| Test Data | INSERT INTO Ride (DriverID, UserID, PickupLocation, DropoffLocation, Date) VALUES (999, 1, 'Colombo', 'Kandy', '2025-04-10'); |
| Expected Results | Fails due to Foreign Key constraint |
| Actual results | Fails due to Foreign Key constraint |
| Test Result | Pass |
| Screenshot | |

**Test Case 5:**

|  |  |
| --- | --- |
| Test Case ID | 5 |
| Test Objectives | Check FK on delete |
| Test Data | DELETE FROM [User] WHERE UserID = 1; |
| Expected Results | Should fail due to FK constraint in Ride table |
| Actual results | Should fail due to FK constraint in Ride table |
| Test Result | Pass |
| Screenshot | |

**Test Case 6:**

|  |  |
| --- | --- |
| Test Case ID | 6 |
| Test Objectives | FK reference from PaymentType |
| Test Data | INSERT INTO Payment (RiderID, PaymentTypeID, Amount, PDate) VALUES (1, 1, 2500, '2025-04-10'); |
| Expected Results | Row inserted if IDs are valid |
| Actual results | Row inserted if IDs are valid |
| Test Result | Pass |
| Screenshot |  |

**Test Case 7:**

|  |  |
| --- | --- |
| Test Case ID | 7 |
| Test Objectives | Delete driver data |
| Test Data | DELETE FROM Driver WHERE NIC = '123456789V'; |
| Expected Results | Row deleted successfully |
| Actual results | Row deleted successfully |
| Test Result | Pass |
| Screenshot |  |

**Test Case 8:**

|  |  |
| --- | --- |
| Test Case ID | 8 |
| Test Objectives | Insert a new payment mode |
| Test Data | INSERT INTO PaymentTypes (PaymentType) VALUES ('Free'); |
| Expected Results | Row inserted successfully |
| Actual results | Row inserted successfully |
| Test Result | Pass |
| Screenshot |  |

**Test Case 9:**

|  |  |
| --- | --- |
| Test Case ID | 9 |
| Test Objectives | Insert a new vehicle Colour |
| Test Data | INSERT INTO VehicleColour(VehicleColour) VALUES ('Brown'); |
| Expected Results | Row inserted successfully |
| Actual results | Row inserted successfully |
| Test Result | Pass |
| Screenshot |  |

**Test Case 10:**

|  |  |
| --- | --- |
| Test Case ID | 10 |
| Test Objectives | Retrieve all Rides with INNER JOIN User & Driver |
| Test Data | SELECT R.RiderID, U.UName, D.Name, R.PickupLocation FROM Ride R  JOIN [User] U ON R.UserID = U.UserID  JOIN Driver D ON R.DriverID = D.DriverID; |
| Expected Results | Joined rows shown |
| Actual results | Joined rows shown |
| Test Result | Pass |
| Screenshot |  |

**Test Case 11:**

|  |  |
| --- | --- |
| Test Case ID | 11 |
| Test Objectives | Check average rating for a driver |
| Test Data | SELECT D.DriverID, D.Name, AVG(Review.SRating) AS AvgRating FROM Review  JOIN Ride R ON Review.RiderID = R.RiderID  JOIN Driver D ON R.DriverID = D.DriverID  GROUP BY D.DriverID, D.Name; |
| Expected Results | Avg shown per driver |
| Actual results | Avg shown per driver |
| Test Result | Pass |
| Screenshot |  |

**Test Case 12:**

|  |  |
| --- | --- |
| Test Case ID | 12 |
| Test Objectives | Insert valid review |
| Test Data | INSERT INTO Review (RiderID, SRating, RDes, RName) VALUES (1, 4, 'Poor', 'Nilu'); |
| Expected Results | Row added |
| Actual results | Row added successfully |
| Test Result | Pass |
| Screenshot |  |

**Test Case 13:**

|  |  |
| --- | --- |
| Test Case ID | 13 |
| Test Objectives | Fetch Rides by Date range |
| Test Data | SELECT \* FROM Ride WHERE Date BETWEEN '2024-09-01' AND '2024-09-10'; |
| Expected Results | Only rides in that range. |
| Actual results | Only rides in that range. |
| Test Result | Pass |
| Screenshot |  |

**Test Case 14:**

|  |  |
| --- | --- |
| Test Case ID | 14 |
| Test Objectives | Validate FK from Payment to Ride |
| Test Data | INSERT INTO Payment (RiderID, PaymentTypeID, Amount, PDate) VALUES (999, 1, 1200, '2025-04-11'); |
| Expected Results | Foreign key error. |
| Actual results | Foreign key error. |
| Test Result | Pass |
| Screenshot | |

**Test Case 15:**

|  |  |
| --- | --- |
| Test Case ID | 15 |
| Test Objectives | Attempt to delete PaymentType in use |
| Test Data | DELETE FROM PaymentTypes WHERE PaymentTypeID = 5; |
| Expected Results | Row deleted successfully |
| Actual results | Row deleted successfully |
| Test Result | Pass |
| Screenshot |  |

**Test Case 16:**

|  |  |
| --- | --- |
| Test Case ID | 16 |
| Test Objectives | Check case sensitivity on Username |
| Test Data | SELECT \* FROM [User] WHERE Username = 'NICK'; |
| Expected Results | No results as it is not case-insensitive |
| Actual results | No results as it is not case-insensitive |
| Test Result | Pass |
| Screenshot |  |

**Test Case 17:**

|  |  |
| --- | --- |
| Test Case ID | 17 |
| Test Objectives | Fetch Rides by Date range |
| Test Data | INSERT INTO [User] (Username, UName, PhoneNum, Password, AddressLineOne, AddressLineTwo)  VALUES ('xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  xxxxxxxxx', 'Kamal', '0712233637', 'camcbasc', 'nagoda', 'ragama'); |
| Expected Results | Fails as Username exceeds max length defined in schema |
| Actual results | Fails as Username exceeds max length defined in schema |
| Test Result | Pass |
| Screenshot | |

**Test Case 18:**

|  |  |
| --- | --- |
| Test Case ID | 18 |
| Test Objectives | List Users with no Rides |
| Test Data | SELECT \* FROM [User] WHERE UserID NOT IN (SELECT UserID FROM Ride); |
| Expected Results | Returns users who haven't taken any rides |
| Actual results | Returns users who haven't taken any rides |
| Test Result | Pass |
| Screenshot |  |

**Conclusion**

A design and development process of a ride-hailing application relational database succeeded through normalization standards (up to 3NF) along with data integrity maintained by relational constraints. Data management throughout the SQL Server implementation process proved to be effective from establishing database tables up to running queries against them. The complete testing phase confirmed system functionality and security as well as performance capabilities. The project illustrated the benefits of relational databases when compared to earlier models which now people can observe in their flexibility along with their scalability and user-friendly nature. The system demonstrates compliance with all requirements that track user and driver data together with trip information as well as payment transactions and customer reviews therefore establishing a sound base for the ride-sharing application.

**References**

* dzsquared (2025). *Install SQL Server Data Tools (SSDT) - SQL Server Data Tools (SSDT)*. [online] Microsoft.com. Available at: <https://learn.microsoft.com/en-us/sql/ssdt/download-sql-server-data-tools-ssdt?view=sql-server-ver16&tabs=vs2022> [Accessed 14 Apr. 2025].
* GeeksforGeeks. (2023). *Complete Reference to Databases in Designing Systems - Learn System Design*. [online] Available at: <https://www.geeksforgeeks.org/complete-reference-to-databases-in-designing-systems/>.
* Lifewire. (n.d.). *An Intro to Databases That’s Suitable for the Brand New Beginner*. [online] Available at: <https://www.lifewire.com/databases-for-beginners-1019643>.
* LucidChart (2024). *What is an Entity Relationship Diagram (ERD)?* [online] Lucidchart. Available at: <https://www.lucidchart.com/pages/er-diagrams>.
* MikeRayMSFT (2023). *Data types (Transact-SQL) - SQL Server*. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16>.
* Pratik Satasiya (2019). *Challenges of Database Testing*. [online] dzone.com. Available at: <https://dzone.com/articles/importance-amp-challenges-of-database-testing> [Accessed 14 Apr. 2025].
* Red (2025). *SQL Test - SQL Server Unit Testing Tool From Redgate*. [online] Red-gate.com. Available at: <https://www.red-gate.com/products/sql-test/> [Accessed 14 Apr. 2025].
* rwestMSFT (2025). *SQL Server Management Studio (SSMS)*. [online] Microsoft.com. Available at: <https://learn.microsoft.com/en-us/ssms/sql-server-management-studio-ssms>.
* SQL Server Tutorial. (n.d.). *SQL Server Tutorial*. [online] Available at: <https://www.sqlservertutorial.net/>.
* Tutorialspoint.com. (2019). *DBMS - Data Models - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/dbms/dbms_data_models.htm>.
* w3schools (2019). *SQL Tutorial*. [online] W3schools.com. Available at: <https://www.w3schools.com/sql/>.
* www.guru99.com. (2023). *Top 25 Database Testing Interview Questions & Answers (2023)*. [online] Available at: <https://www.guru99.com/database-testing-interview-questions.html>.