DATABASE DESIGN AND DEVELOPMENT ON YYC CAR RENTAL

-Harmanjot Kaur

TABLE OF CONTENT

Table of Contents

ABOUT COMPANY

YYC Rentals is a leading Canadian-based car rental company, committed to delivering reliable, accessible, and secure car rental services in Calgary, Canada. Our mission is to provide an exceptional rental experience by offering a diverse range of vehicles that cater to every customer's unique needs—whether it's a budget-friendly car for a weekend getaway or a luxury vehicle for a special occasion. With branches in major cities, YYC Rentals ensures easy access and convenience for customers nationwide.

Customer satisfaction is at the core of YYC Rentals. We prioritize user privacy, providing a secure platform for all transactions. Our technology-driven approach streamlines the booking process, offering personalized services, transparent pricing, and special promotions. Through our user loyalty program and ongoing customer feedback system, we are continuously improving our services to ensure every rental experience is smooth and enjoyable.

YYC Rentals is more than just a car rental company—it is a customer-first business dedicated to building lasting relationships. Our team of highly trained professionals is committed to providing outstanding customer service, ensuring that every client receives the support they need, from reservation to return. By blending affordability, innovation, and quality service, YYC Rentals is setting a new standard for the car rental industry in Canada.

MISSION

Our mission is to provide a seamless, reliable, and secure car rental experience for customers across Canada. We strive to make car rentals accessible to everyone by offering a diverse fleet of vehicles that meet various needs and preferences. Our priority is ensuring customer satisfaction through exceptional service, affordable pricing, and robust security measures. We are committed to protecting user privacy and delivering a smooth, user-friendly rental process through a secure and innovative platform.

OBJECTIVES

1. Vast Vehicle Selection:

Our goal is to expand our fleet continuously to offer a wide range of vehicles, from economy options to luxury cars, to cater to different customer preferences and needs. By providing a versatile selection, we ensure that customers can find the perfect vehicle for every occasion, whether for business trips, family vacations, or city commuting.

2.Affordable Pricing:

We aim to offer competitive pricing, making car rentals accessible to a wide range of customers. By utilizing data-driven pricing models, we can optimize rental costs while maintaining profitability. Offering discounts and loyalty rewards further enhances affordability, creating value for our customers.

3.Exceptional Client Service:

Delivering high-quality client service is central to our business. From the moment customers visit our platform to when they return the vehicle, we aim to provide assistance and support that exceeds expectations. By leveraging technology, we ensure real-time support, easy reservations, and transparent communication with our clients.

4.Technology-Driven Insights:

We utilize data analytics to continually improve our services. Insights gained from user preferences, rental patterns, and customer feedback help us make informed decisions, streamline operations, and improve customer experience. This data-centric approach allows us to enhance vehicle availability, rental speed, and client satisfaction.

5.Secure and Reliable System:

In an era of heightened data security concerns, we are committed to maintaining a secure platform that ensures customer data privacy and transaction integrity. Our system is designed with multi-layered security protocols, audit trails, and encrypted transactions to build trust and reliability with our users.

DATABASE DESIGN

Why There is a Need for a Database at YYC Rentals?

The implementation of a database at YYC Rentals is essential for several reasons, all of which contribute to the company's ability to operate effectively and achieve the desired objectives. Here are the key reasons Highlighting the necessity of a database for the company:

1.Organized Data Storage:

As YYC Rentals manages numerous customer profiles, vehicle inventories, reservations, and branch details, a database allows for structured and organized storage of this information. This organization facilitates easy retrieval, updates, and management of data, reducing the risk of errors associated with manual processes

2.Improved Efficiency:

A database automates many administrative tasks related to bookings, inventory management, and customer service. This efficiency minimizes the time spent on manual data entry, increases productivity, and allows employees to focus more on customer interactions and service enhancements.

3.Real-Time Data Access:

With a centralized database, employees can access real-time information regarding vehicle availability, customer reservations, and user details. This immediate access is crucial for making informed decisions, such as confirming bookings and managing inventory effectively

4.Enhanced Customer Experience:

A database enables YYC Rentals to track customer preferences, rental history, and feedback, allowing for a more personalized service experience. Understanding customer needs and preferences helps the company tailor offerings, promotions, and communication, ultimately improving customer satisfaction and loyalty.

5.Data Integrity and Security:

The database ensures that customer and operational data is stored securely and maintained accurately. With built-in security measures, sensitive information is protected from unauthorized access and breaches, fostering trust among customers and compliance with legal regulations regarding data privacy.

6.Scalability:

As the company grows, the volume of data will also increase. A well-designed database can easily scale to accommodate additional users, vehicles, branches, and services, ensuring that YYC Rentals can adapt to changing market demands without compromising efficiency or performance.

7.Data-Driven Decision Making:

The database can facilitate advanced data analysis and reporting, enabling the management team to make informed decisions based on insights derived from rental patterns, customer feedback, and market trends. This capability supports strategic planning and operational improvements.

8.Error Reduction:

By automating processes and maintaining centralized records, the database minimizes the potential for human error that can occur with manual data handling. This reliability is crucial for maintaining accurate records and enhancing overall service quality.

TABLES

The database for YYC Rentals has been designed with specific tables to address the core operational needs of the company, aligning with its mission to provide reliable, secure, and customer-focused car rental services. Each table in the database plays a vital role in ensuring efficient, streamlined processes and a superior customer experience.

The tables required for the database are:-

1.Customer Table:

This table stores critical information about the users of the car rental system. It ensures privacy and accessibility by keeping detailed user data such as names, email addresses, hashed passwords, contact information, and user activity status. By storing encrypted passwords and allowing each user to control their information, we ensure high levels of security and privacy. The table also enables better customer service by providing easy access to user details for support teams.

2.Vehicles Table:

The Vehicles table is fundamental to maintaining a wide selection of cars available for rent. It holds detailed information about each vehicle in the fleet, including the make, model, year, vehicle type, mileage, and availability status. Additionally, it tracks the daily rental rates and the branch where the vehicle is located. This allows for the real-time management of available vehicles, ensuring customers can find the right car for their specific needs at the most competitive price.

3.Reservations Table:

The Reservations table tracks each rental reservation from the moment a customer books a vehicle until the rental period ends. It includes details such as the user ID, vehicle ID, reservation date, rental start and end dates, total price, and reservation status. This table is designed to improve customer satisfaction by simplifying the booking process, offering technology-driven insights into rental history, and optimizing fleet availability.

4.Branches / Locations Table:

To ensure customers have convenient access to rental services, the Branches table stores data about the company's rental locations. This table includes each branch's name, address, contact information, and city details. By associating vehicles with specific branches, the table ensures customers can easily find and rent vehicles near their preferred location.

5.Employees Table:

The Employees table manages details of employees working in different branches. It tracks employee names, roles (such as customer service representatives or mechanics), branch assignments, and contact information. By organizing employee data in this way, the company can assign tasks efficiently and ensure high levels of client service at all branches.

6.Vehicle Insurance Table:

This table ensures that all vehicles in the fleet are covered by appropriate insurance policies. It stores information such as the insurance provider, policy number, coverage amount, and policy validity dates for each vehicle. This information ensures that each vehicle meets legal requirements and that accidents or damages are covered, contributing to a secure and reliable rental system.

7.Feedbacks and Ratings Table:

Customer feedback is essential to improving service. The Feedbacks and Ratings table captures user opinions about the vehicles they rented, including their ratings on a scale of 1 to 5 and any additional comments they may provide. This data is critical for technology-driven insights, helping the company make informed decisions about vehicle maintenance, customer preferences, and overall service quality.

8.Accident Reports Table:

The Accident Reports table is designed to store comprehensive details regarding any accidents involving vehicles from YYC Rentals. This table is crucial for maintaining a secure and reliable rental system, as it helps the company track incidents, manage risk, and ensure that appropriate actions are taken in response to accidents. It facilitates better decision-making regarding vehicle safety, insurance claims, and customer support while enhancing overall operational transparency.

DATA DICTIONARY

A **data dictionary** is a centralized repository that provides detailed information about the data elements in a database, including their meanings, relationships, constraints, and usage The data dictionary for YYC CAR RENTALS Tables are:

1.**Customer Table:**

Attribute	**Description**	**Data Type**	**Constraints**
user_id	Unique identifier for each user	INT	PRIMARYKEY, AUTO_INCREMENT
first_name	User's first name	VARCHAR(50)	NOT NULL
last_name	User's last name	VARCHAR(50)	NOT NULL
email	User's email address (used for login)	VARCHAR(100)	NOT NULL, UNIQUE

password_hash	Encrypted password for secure login	VARCHAR(255)	NOT NULL
phone_number	User's contact number	VARCHAR(20)	
address	User's physical address	TEXT	
created_at	Timestamp of user account creation	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

2. **Vehicle Table**

Attribute	**Description**	**Data Type**	**Constraints**
vehicle_id	Unique identifier for each vehicle	INT	PRIMARY KEY, AUTO_INCREMENT
Insurance_id	Insurance of the vehicle	INT	FOREIGN KEY
model	Model name of the vehicle	VARCHAR(50)	NOT NULL
year	Manufacturing year of the vehicle	INT	
vehicle_type	Type of vehicle (e.g., SUV, Sedan, etc.)	VARCHAR(50)	
daily_rate	Rental price per day	DECIMAL(10, 2)	
availability_status	Availability of the vehicle (e.g., available, rented, maintenance)	VARCHAR(20)	NOT NULL
mileage	Current mileage of the vehicle	INT	
branch_id	Foreign key referencing the branch where the vehicle is located	INT	FOREIGN KEY

3. Reservation Table

Attribute	Description	Data Type	Constraints
reservation_id	Unique identifier for each reservation	INT	PRIMARY KEY, AUTO_INCREMENT
user_id	Foreign key referencing the user making the reservation	INT	FOREIGN KEY, NOT NULL
reservation_date	Date the reservation was made	DATE	NOT NULL
start_date	Start date of the rental period	DATE	NOT NULL
end_date	End date of the rental period	DATE	NOT NULL
total_price	Total cost of the rental	DECIMAL(10, 2)	
status	Reservation status (e.g., confirmed, cancelled, completed)	VARCHAR(20)	

4. Branches/Location Table:

Attribute	Description	Data Type	Constraints
branch_id	Unique identifier for each branch	INT	PRIMARY KEY, AUTO_INCREMENT
branch_name	Name of the rental branch	VARCHAR(100)	NOT NULL
address	Physical address of the branch	TEXT	NOT NULL
city	City where the branch is located	VARCHAR(50)	

state	State where the branch is located	VARCHAR(50)	
zip_code	Postal code for the branch	VARCHAR(10)	
phone_number	Contact phone number for the branch	VARCHAR(20)	
email	Email address for the branch	VARCHAR(100)	

5. Employee Table:

Attribute	Description	Data Type	Constraints
employee_id	Unique identifier for each employee	INT	PRIMARY KEY, AUTO_INCREMENT
first_name	Employee's first name	VARCHAR(50)	NOT NULL
last_name	Employee's last name	VARCHAR(50)	NOT NULL
role	Role of the employee (e.g., customer service, mechanic)	VARCHAR(50)	
branch_id	Foreign key referencing the employee's branch	INT	FOREIGN KEY
hire_date	Date the employee was hired	DATE	
email	Employee's email address	VARCHAR(100)	
phone_number	Employee's contact number	VARCHAR(20)	

6. Vehicle Insurance Table:

Attribute	Description	Data Type	Constraints
insurance_id	Unique identifier for each insurance policy	INT	PRIMARY KEY, AUTO_INCREMENT
provider	Insurance provider's name	VARCHAR(100)	
policy_number	Unique insurance policy number	VARCHAR(50)	
Valid_from	Start date of the insurance policy	DATE	
valid_until	End date of the insurance policy	DATE	
coverage_amount	Insurance coverage amount	DECIMAL(10, 2)	

7. Feedbacks and Rating Table:

Attribute	Description	Data Type	Constraints
review_id	Unique identifier for each feedback or rating	INT	PRIMARY KEY, AUTO_INCREMENT
user_id	Foreign key referencing the user who left the feedback	INT	FOREIGN KEY, NOT NULL
vehicle_id	Foreign key referencing the rated vehicle	INT	FOREIGN KEY, NOT NULL

review_rating	Rating of the vehicle (1-5 scale)	INT	CHECK (rating BETWEEN 1 AND 5)
review_text	Additional text feedback provided by the user	TEXT	
feedback_date	Date the feedback was submitted	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

8. Accident report table

Attribute	Data Type	Description	Constraints
report_id	INT	Unique identifier for each accident report.	Primary Key, Not Null, AUTO_INCREMENT
reservation_id	INT	Links to the Reservations table to specify which reservation is associated with the accident.	Foreign Key, Not Null
vehicle_id	INT	Links to the Vehicles table to specify which vehicle was involved in the accident.	Foreign Key, Not Null
insurance_id	INT	Links to the Vehicle Insurance table to specify the insurance details related to the accident.	Foreign Key, Nullable
accident_date	DATE	The date when the accident occurred.	Not Null
description	TEXT	A detailed description of the circumstances surrounding the accident.	Not Null
damage_cost	DECIMAL(10, 2)	Estimated cost of damages incurred during the accident.	Nullable
is_resolved	BOOLEAN	Indicates whether the accident report has been resolved.	Not Null, Default FALSE

RELATIONSHIPS

In a Relational Database Management System (RDBMS), relationships refer to the associations between tables. These relationships are established through the use of foreign keys, which link one table to another. Understanding and implementing relationships is crucial for ensuring data integrity, maintaining organization, and enabling efficient data retrieval.

Importance of Relationships in RDBMS are:

The importance od relationships in database is explained as:

1.Data Integrity:

Relationships help maintain data accuracy and consistency. By enforcing referential integrity, RDBMS ensures that relationships between tables are valid, preventing orphaned records (records that refer to non-existent data).

2.Efficient Data Retrieval:

Relationships enable complex queries to be performed across multiple tables. This allows for more efficient data retrieval and analysis, as related data can be fetched in a single query using joins.

3.Reduced Data Redundancy:

Properly defining relationships minimizes data duplication by allowing data to be stored in a normalized form. This leads to a more efficient use of storage and simplifies data maintenance.

4.Simplified Data Management:

Relationships make it easier to manage and manipulate related data. Changes made in one table can automatically reflect in related tables due to the established foreign key constraints.Enhanced

5.Data Modeling:

Relationships provide a clear structure to the database design, making it easier to understand and model real-world scenarios. This clarity aids in better communication among stakeholders and developers.Facilitates

6.Business Logic:

Many business processes rely on relationships between entities. By modeling these relationships in the database, organizations can ensure that their applications align with the underlying business logic.

Types of relationships

There are mainly three types of relationships among tables which are explained as:

One to One Relationship (1:1): It is used to create a relationship between two tables in which a single row of the first table can only be related to one and only one records of a second table. Similarly, the row of a second table can also be related to anyone row of the first table.

One to Many Relationship(1:N): It is used to create a relationship between two tables. Any single rows of the first table can be related to one or more rows of the second tables, but the rows of second tables can only relate to the only row in the first table. It is also known as a **many to one** relationship.

Many to Many Relationship(M:N): It is **many to many** relationships that create a relationship between two tables. Each record of the first table can relate to any records (or no records) in the second table. Similarly, each record of the second table can also relate to more than one record of the first table. It is also represented an **N:N** relationship.

The relationship between tables are:

Source Table	Relationship Type	Target Table	Foreign Key	Explanation
Customer	*One-to-Many*	Reservations	user_id	One customer can make multiple reservations.

Source Table	**Relationship Type**	**Target Table**	**Foreign Key**	**Explanation**
Customer	*One-to-Many*	VehicleReviews	user_id	One customer can leave multiple reviews for different vehicles.
Vehicles	*Many-to-One*	Branches	branch_id	Each vehicle belongs to one branch, while a branch can have many vehicles.
Vehicles	*One-to-One*	VehicleInsurance	insurance_id	Each vehicle can have one insurance policy; multiple vehicles can share the same insurance.
Reservations	*Many-to-One*	Vehicles	vehicle_id	Each reservation is for one vehicle, but a vehicle can have multiple reservations.
Reservations	*One-to-Many*	AccidentReports	rental_id	Each reservation can be linked to multiple accident reports.

Source Table	**Relationship Type**	**Target Table**	**Foreign Key**	**Explanation**
Branches	*One-to-Many*	Vehicles	branch_id	A branch can have multiple vehicles available for rent.
Branches	*One-to-Many*	Employees	branch_id	A branch can employ multiple employees.
Vehicles	*One-to-Many*	Reservations	vehicle_id	One vehicle can be reserved multiple times.
Vehicles	*One-to-Many*	VehicleReviews	vehicle_id	One vehicle can receive multiple reviews from different customers.

Source Table	**Relationship Type**	**Target Table**	**Foreign Key**	**Explanation**
Vehicles	*One-to-Many*	AccidentReports	vehicle_id	A vehicle can be involved in multiple accident reports.
Reservations	*Many-to-One*	Customer	user_id	Each reservation is made by one customer.
VehicleReviews	*Many-to-One*	Customer	user_id	Each review is submitted by one customer, while a customer can submit multiple reviews.
VehicleReviews	*Many-to-One*	Vehicles	vehicle_id	Each review is for one vehicle, but a vehicle can have multiple reviews.

Source Table	**Relationship Type**	**Target Table**	**Foreign Key**	**Explanation**
AccidentReports	*Many-to-One*	Reservations	rental_id	Each accident report can be associated with one reservation.
AccidentReports	*Many-to-One*	Vehicles	vehicle_id	Each accident report involves one vehicle.
AccidentReports	*Many-to-One*	VehicleInsurance	insurance_id	Each accident report can relate to one insurance policy.
Employees	*Many-to-One*	Branches	branch_id	Each employee works for one branch, while a branch can have many employees.

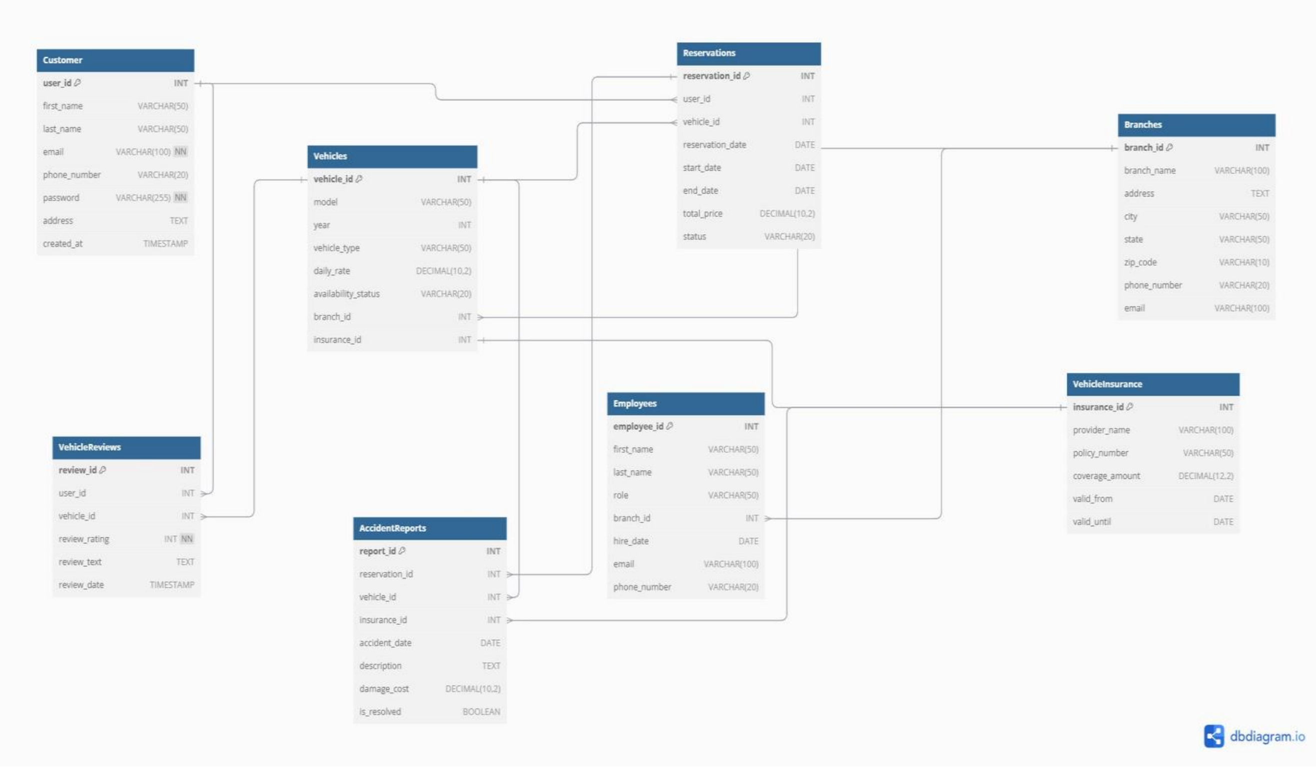
ER DIAGRAM

Peter Chen developed the ER diagram in 1976. The ER model was created to provide a simple and understandable model for representing the structure and logic of databases. It has since evolved into variations such as the Enhanced ER Model and the Object Relationship Model.
The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.

Why Use ER Diagrams In DBMS?

- ER diagrams represent the E-R model in a database, making them easy to convert into relations (tables).
- ER diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
- ER diagrams require no technical knowledge and no hardware support.
- These diagrams are very easy to understand and easy to create even for a naive user.
- It gives a standard solution for visualizing the data logically.

The ER Diagram for this database is:



Customer		
user_id	**INT**	
first_name	VARCHAR(50)	
last_name	VARCHAR(50)	
email	VARCHAR(100)	NN
phone_number	VARCHAR(20)	
password	VARCHAR(255)	NN
address	TEXT	
created_at	TIMESTAMP	

Vehicles	
vehicle_id	**INT**
model	VARCHAR(50)
year	INT
vehicle_type	VARCHAR(50)
daily_rate	DECIMAL(10,2)
availability_status	VARCHAR(20)
branch_id	INT
insurance_id	INT

Reservations	
reservation_id	**INT**
user_id	INT
vehicle_id	INT
reservation_date	DATE
start_date	DATE
end_date	DATE
total_price	DECIMAL(10,2)
status	VARCHAR(20)

Branches	
branch_id	**INT**
branch_name	VARCHAR(100)
address	TEXT
city	VARCHAR(50)
state	VARCHAR(50)
zip_code	VARCHAR(10)
phone_number	VARCHAR(20)
email	VARCHAR(100)

VehicleReviews		
review_id	**INT**	
user_id	INT	
vehicle_id	INT	
review_rating	INT	NN
review_text	TEXT	
review_date	TIMESTAMP	

AccidentReports	
report_id	**INT**
reservation_id	INT
vehicle_id	INT
insurance_id	INT
accident_date	DATE
description	TEXT
damage_cost	DECIMAL(10,2)
is_resolved	BOOLEAN

Employees	
employee_id	**INT**
first_name	VARCHAR(50)
last_name	VARCHAR(50)
role	VARCHAR(50)
branch_id	INT
hire_date	DATE
email	VARCHAR(100)
phone_number	VARCHAR(20)

VehicleInsurance	
insurance_id	**INT**
provider_name	VARCHAR(100)
policy_number	VARCHAR(50)
coverage_amount	DECIMAL(12,2)
valid_from	DATE
valid_until	DATE

DATABASE DEVELOPMENT

We create a sample database for the rental car system. Below is the sample query run during the development.

Example:

Create Database:

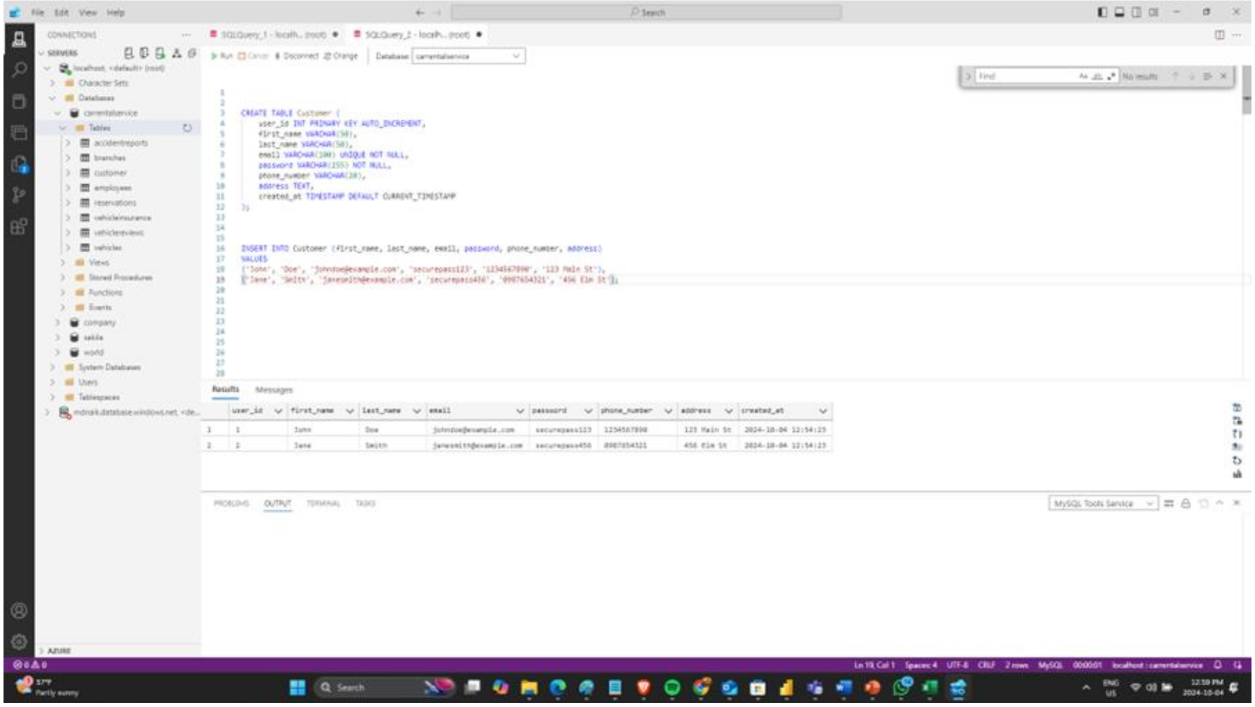
```
CREATE DATABASE CarRentalService;
```

Create table and Insert query for Customer table:

1. Customer Table

```
CREATE TABLE Customer (

   user_id INT PRIMARY KEY AUTO_INCREMENT,

   first_name VARCHAR(50),

   last_name VARCHAR(50),

   email VARCHAR(100) UNIQUE NOT NULL,

   password VARCHAR(255) NOT NULL,

   phone_number VARCHAR(20),

   address TEXT,

   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

);


INSERT INTO Customer (first_name, last_name, email, password,
phone_number, address)

VALUES

('John', 'Doe', 'johndoe@example.com', 'securepass123', '1234567890', '123 Main
St'),
```

('Jane', 'Smith', 'janesmith@example.com', 'securepass456', '0987654321', '456 Elm St');



```
CREATE TABLE Customer (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    phone_number VARCHAR(20),
    address TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);



INSERT INTO Customer (first_name, last_name, email, password, phone_number, address)
VALUES
('John', 'Doe', 'johndoe@example.com', 'securepass123', '1234567890', '123 Main St'),
('Jane', 'Smith', 'janesmith@example.com', 'securepass456', '0987654321', '456 Elm St');
```

	user_id	first_name	last_name	email	password	phone_number	address	created_at
1	1	John	Doe	johndoe@example.com	securepass123	1234567890	123 Main St	2024-10-04 12:54:23
2	2	Jane	Smith	janesmith@example.com	securepass456	0987654321	456 Elm St	2024-10-04 12:54:23

SQLQuery_1 - localh... (root)
SQLQuery_2 - localh... (root)
Run
Cancel
Disconnect
Change
Database: carrentalservice
Results
Messages
PROBLEMS
OUTPUT
TERMINAL
TASKS

Inner Join Query

SELECT

Customer.first_name,

Customer.last_name,

COUNT(Reservations.reservation_id) AS total_reservations

FROM

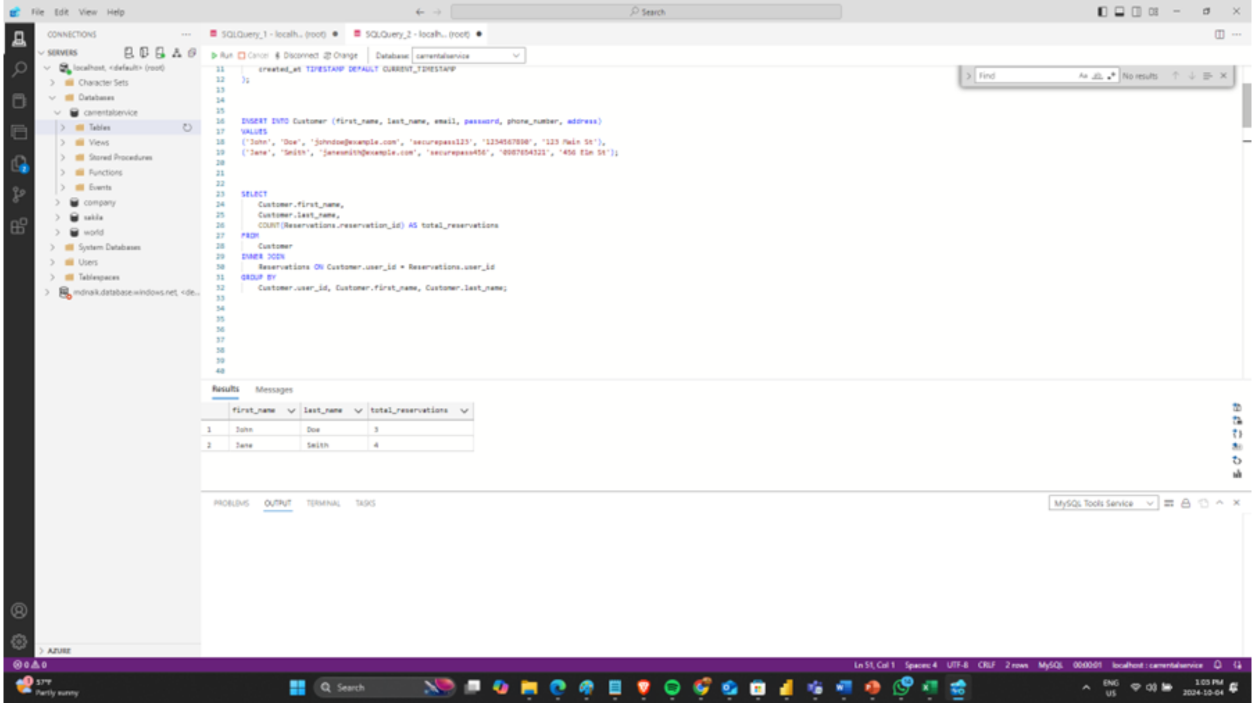
Customer INNER JOIN

Reservations ON Customer.user_id = Reservations.user_id

GROUP BY

Customer.user_id, Customer.first_name, Customer.last_name;

Explanation: We are trying to get the number of bookings by customer and displaying their first name and last name by matching the reservation id in reservation table and customer id from customer table.



```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);



INSERT INTO Customer (first_name, last_name, email, password, phone_number, address)
VALUES
('John', 'Doe', 'johndoe@example.com', 'securepass123', '1234567890', '123 Main St'),
('Jane', 'Smith', 'janesmith@example.com', 'securepass456', '0987654321', '456 Elm St');



SELECT
    Customer.first_name,
    Customer.last_name,
    COUNT(Reservations.reservation_id) AS total_reservations
FROM
    Customer
INNER JOIN
    Reservations ON Customer.user_id = Reservations.user_id
GROUP BY
    Customer.user_id, Customer.first_name, Customer.last_name;
```

	first_name	last_name	total_reservations
1	John	Doe	3
2	Jane	Smith	4

Group By Query

SELECT

    Vehicles.model,

    SUM(Reservations.total_price) AS total_revenue

FROM

    Vehicles

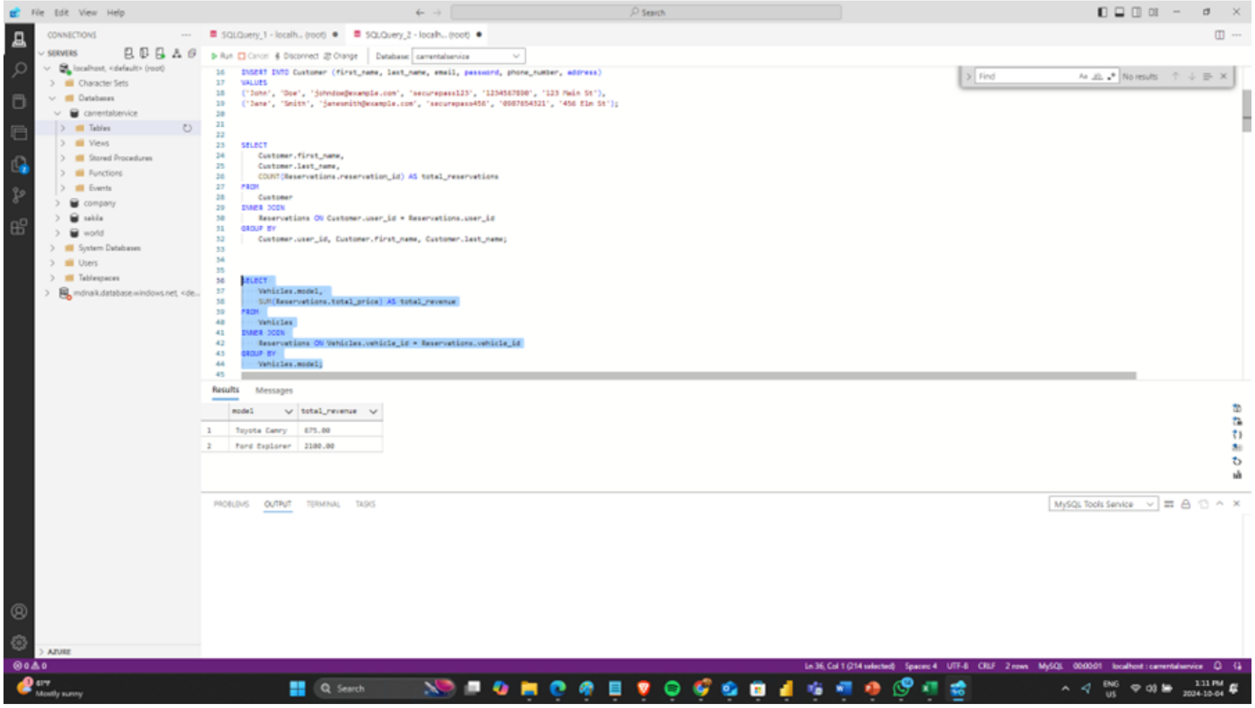
INNER JOIN

    Reservations ON Vehicles.vehicle_id = Reservations.vehicle_id

```
GROUP BY
    Vehicles.model;
```

Explanation:

We are trying to get the revenue generated by each vehicle using group by.



```sql
INSERT INTO Customer (first_name, last_name, email, password, phone_number, address)
VALUES
('John', 'Doe', 'johndoe@example.com', 'securepass123', '1234567890', '123 Main St'),
('Jane', 'Smith', 'janesmith@example.com', 'securepass456', '0987654321', '456 Elm St');



SELECT
    Customer.first_name,
    Customer.last_name,
    COUNT(Reservations.reservation_id) AS total_reservations
FROM
    Customer
INNER JOIN
    Reservations ON Customer.user_id = Reservations.user_id
GROUP BY
    Customer.user_id, Customer.first_name, Customer.last_name;



SELECT
    Vehicles.model,
    SUM(Reservations.total_price) AS total_revenue
FROM
    Vehicles
INNER JOIN
    Reservations ON Vehicles.vehicle_id = Reservations.vehicle_id
GROUP BY
    Vehicles.model;
```

	model	total_revenue
1	Toyota Camry	675.00
2	Ford Explorer	2100.00

CONCLUSION

The development of the database for **YYC Rentals** is key to supporting the company's mission of providing reliable, accessible, and secure car rental services with a focus on user privacy and satisfaction. Through effective **database design and development**, essential business entities like **Users**, **Vehicles**, **Reservations**, and **Feedbacks** were structured into relational tables, ensuring the efficient organization and retrieval of data.

By establishing relationships between these tables, the database ensures seamless data flow, reduces redundancy, and maintains data integrity across operations such as reservations, customer interactions, and vehicle management. This relational design allows for better decision-making through technology-driven insights, helping the company meet its objectives of offering a wide vehicle selection, competitive pricing, and exceptional customer service.

Overall, the database provides the foundation for YYC Rentals to manage its operations efficiently, improve customer satisfaction, and support long-term growth while maintaining data security and privacy.

APPENDIX

An appendix is a supplementary section at the end of a document or report that provides additional information, data, or resources relevant to the main content. It typically contains detailed explanations, charts, tables, or any other material that supports the main body of the work but is too lengthy or detailed to include within the main text.

Importance of an Appendix:

1.Enhances Clarity:By placing supplementary material in an appendix, the main text remains clear and concise. This separation allows readers to focus on the core message without being overwhelmed by excessive details.

2.Provides Additional Resources:

An appendix offers an opportunity to present supplementary information such as charts, graphs, data sets, or extensive examples that enhance the reader's understanding of the topic.

3.Facilitates Reference:

Readers can easily refer to the appendix for specific data or explanations without interrupting the flow of the main text. This organization helps maintain a logical structure.**Supports Validation**:

Including additional data, such as calculations, surveys, or research findings, can support the claims made in the main body of the document. This adds credibility and validates the research or findings presented.

4.Accommodates Lengthy Information:

For reports that require extensive data or lengthy discussions, the appendix allows authors to provide this information without disrupting the narrative of the main text.

Appendix: Data Types and Descriptions

Data Type	**Description**
INT	Integer data type used for unique identifiers (IDs) and numerical fields without decimals.
VARCHAR(n)	Variable-length character string, used for text fields like names, emails, and other short text values.
TEXT	Large text data type used for storing long text content such as addresses or detailed descriptions.
DECIMAL(10,2)	Decimal data type with fixed precision, used for financial values like prices and damage costs.
BOOLEAN	Logical data type used for true/false values, often indicating status (e.g., availability, resolution).
DATE	Date data type used for storing calendar dates, such as rental or accident dates.
TIMESTAMP	Date and time data type, typically used for automatically recording the time of record creation or update.

Appendix: Constraints and Descriptions

Constraint	Description
Primary Key (PK)	Ensures that each record in a table is uniquely identified by a specific column (or set of columns).
Foreign Key (FK)	Creates a relationship between two tables, ensuring referential integrity by linking one table's column to another.
Auto Increment	Automatically generates a unique sequential value for a column, typically used for primary keys.
Not Null	Ensures that a column cannot have a NULL (empty) value, meaning it must always contain data.
Unique	Ensures that all values in a column are different across all rows, preventing duplicates.
Default	Assigns a default value to a column if no value is specified during record creation.
Check	Enforces a specific condition on column values, ensuring that data entered satisfies the condition.
Foreign Key Cascade	Deletes or updates records in related tables if the corresponding record in the parent table is deleted or updated.