indium

Make Technology Work

Capstone Project

Vehicle Booking Application

Database Design

By

P Hemanth Kumar Reddy

[Batch_04]

Abstract

The vehicle booking application database design provides a structured approach for managing the “bookings , vehicle , payments, reviews and ratings ”, on vehicle rentals. The abstract of this database design outlines the various entities involved in the system, including customers, admins, drivers, vehicles, booking details, and payment information. The design includes normalized tables with proper relationships, ensuring data integrity and efficient querying.

The database design aims to address the challenges of managing a vehicle booking system, including managing a vast number of booking requests, assigning drivers and vehicles, tracking payments and driver performance, and generating reports.

The database design provides a comprehensive and flexible solution that can be easily extended to accommodate changing business requirements.

Introduction

The “vehicle booking application” database design is an efficient and comprehensive solution for managing vehicle bookings and their associated data. The system is designed to meet the needs of vehicle rental companies, allowing them to manage and monitor their data of vehicles, bookings, and customers.

This database design provides a flexible solution to manage and track all the necessary information related to vehicle bookings, customers, and vehicles. By using this kind of database, vehicle rental companies can streamline their booking processes, reduce errors, and enhance customer satisfaction. This presentation is an detailed overview of the database design, its features, and functionalities.

Outcomes

This vehicle booking application database design is being implemented to provide a robust and efficient database system for a vehicle booking application. By implementing this database design, we can achieve the following benefits:

Efficient data management: The database design provides a structured way of storing and managing data related to bookings, customers, vehicles, drivers, and routes.

Improved user experience: A well-designed database can provide faster response times and reduce downtime, which will ultimately result in an improved user experience for the customers.

Scalability: The database design is scalable, which means it can accommodate large amounts of data and support future growth of the application.

Data consistency: The database design ensures that the data is consistent and accurate, reducing the chances of errors and discrepancies in the data.

Overall, the implementation of the above database design will provide a reliable and efficient foundation for the vehicle booking application, allowing for smoother operations and better customer satisfaction.

Objectives

- To create a centralized database for storing customer, driver, vehicle, and booking information.
- To enable customers to search for and book vehicles based on their preferences and requirements.
- To maintain data integrity by enforcing data constraints and validating input.
- To provide flexibility to accommodate future changes and additions to the system.
- To increase customer satisfaction by providing timely and accurate information on booking status and vehicle availability.

Existing Systems

Uber, a ride-hailing service, manages a vast amount of data related to their customers, drivers, rides, payments, and more. They use a distributed database management system to handle the scale of their operations.

Uber's database schema consists of several databases, including:

Users: This database contains information about the customers and drivers, including their names, contact details, ratings, and payment information.

Rides: This database stores information about each ride, including the pickup and drop-off locations, ride distance, ride duration, fare, and payment method.

Payments: This database contains information about all the payment transactions that occur between customers and drivers, including payment details, refunds, and cancellations.

Analytics: This database stores data related to customer behavior, driver performance, and other business metrics, which are used to optimize their operations and services.

Uber's database management system uses various technologies, including Apache Cassandra, a distributed NoSQL database, and Apache Hadoop, a distributed data processing framework. They also use several other open-source tools and platforms to manage their data, including Apache Kafka for real-time data streaming and Apache Spark for data analysis and processing.

- Uber uses a mix of SQL and NoSQL databases to manage their vast amount of data. They use MySQL for their relational database needs and Cassandra for their NoSQL database needs.
- MySQL is used to store data related to trips, drivers, riders, promotions, and payment transactions. Cassandra is used for storing real-time data such as driver and rider locations, ride requests, and ride updates.
- The Uber database schema consists of multiple tables for storing different types of data. Some of the key tables include the Trips table for storing information about completed trips, Drivers table for storing information about drivers, Riders table for storing information about riders, Payments table for storing information about payment transactions, and Promotions table for storing information about promotional offers.
- In addition to these tables, Uber also uses other supporting tables to manage various aspects of their business such as ratings and reviews, driver incentives, and fraud detection.
- The Uber database is designed to handle a massive amount of data and high traffic volume. They use various techniques such as sharding, replication, and load balancing to ensure high availability and performance.

Uber has **implemented various techniques and methodologies** to provide efficient database services to its users. Some of these include:

Sharding: Uber uses sharding to split its databases into smaller, more manageable pieces. By doing this, they can distribute the data across multiple servers, which helps to improve performance and scalability.

Replication: Uber uses replication to create copies of its databases across multiple servers. This helps to ensure that data is always available, even if one server goes down.

Caching: Uber uses caching to store frequently accessed data in memory, which helps to reduce the number of database queries and improve performance.

Load balancing: Uber uses load balancing to distribute requests across multiple servers. This helps to ensure that no single server becomes overloaded, which can lead to slower performance.

Real-time data processing: Uber uses real-time data processing techniques to process data as it is generated. This helps to ensure that data is always up-to-date and can be used to provide real-time insights and analytics.

Data partitioning: Uber uses data partitioning to divide its data into smaller, more manageable chunks. This helps to improve performance by reducing the amount of data that needs to be processed at any one time.

Overall, Uber's database schema and management system are designed to handle large-scale, real-time data processing and analysis, which is critical for their operations and growth.

Proposed Methodology

This proposed methodology is used for designing this “vehicle booking application” database
It includes the following steps:

Requirements gathering: Collecting all the necessary information about the system to be modeled, including user requirements, business rules, and constraints.

Conceptual design: Creating an abstract model of the database that represents all the entities, relationships, and attributes involved.

Logical design: Translating the conceptual model into a detailed logical model that defines all the tables, columns, and constraints needed to store and manage the data.

Physical design: Implementing the logical model in a specific database management system and choosing appropriate data types, indexing strategies, and storage options.

Implementation: Writing code to create the database, tables, and indexes, and to populate them with data.

Testing and maintenance: Verifying the correctness and completeness of the database design, testing for performance and scalability, and maintaining the database over time to ensure data integrity and availability.

Tools Used

MySQL Workbench is a visual tool used for database design, development, and administration. It provides a graphical interface to work with databases, which includes creating and modifying database schemas, writing SQL queries, and managing database users and permissions.

MySQL Server is a popular open-source relational database management system (RDBMS) that uses SQL (Structured Query Language) to interact with data. It is widely used in web applications and is known for its high performance, reliability, and scalability.

MySQL Command Line Tool, also known as MySQL Client, is a command-line interface that allows users to interact with a MySQL server through the terminal or console. It provides a way to execute SQL queries and manage databases, tables, and users using text-based commands. It is often used for automation, scripting, and system administration tasks.

MYSQL Objects Used

In this “vehicle booking application” database design process the following MYSQL objects

are used.

Tables: Tables are used to store data in the database. In the vehicle booking application database design, several tables are used such as the bookings table, customers table, drivers table, vehicles table, payments, reviews , Booking Cancellation.

Constraints: Constraints are used to define rules for data in tables. Examples of constraints include primary key constraints, foreign key constraints, and check constraints. In the vehicle booking application database design, foreign key constraints are used to enforce referential integrity between tables.

Keys: Keys are used to uniquely identify records in tables. In the vehicle booking application database design, primary keys are used to uniquely identify records in tables such as the bookings table, customers table, drivers table, vehicles table, payments, reviews, Booking Cancellation. And foreign keys are used to integrate the tables based on the business requirement.

Joins: Joins are used to combine data from multiple tables based on a common column. In the vehicle booking application database design, joins are used to combine data from tables such as the bookings table, customers table, drivers table, vehicles table, payments, reviews, Booking Cancellation.

Views: Views are virtual tables that are based on the result of a SQL query. Views are used to simplify complex queries and provide a customized view of data for users. In the vehicle booking application database design, views can be created to provide a customized view of data for users.

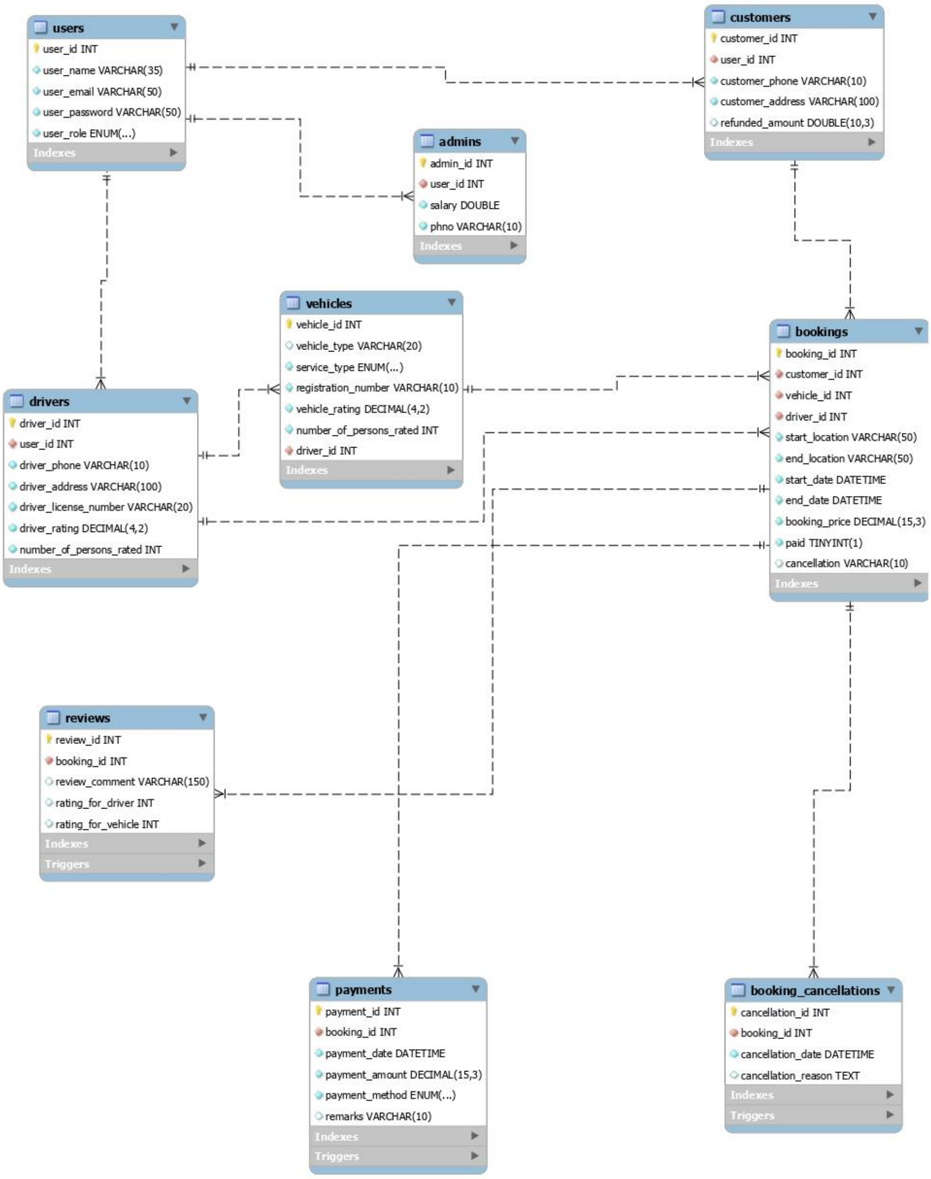
Common Table Expressions (CTEs): CTEs are temporary named result sets that are defined within a single execution of a SQL statement. CTEs can simplify complex queries and improve performance. In the vehicle booking application database design, CTEs can be used to simplify complex queries.

Triggers: Triggers are special types of stored procedures that are automatically executed in response to specific events or actions. Triggers can be used to enforce business rules, maintain data integrity, and automate database operations. In the vehicle booking application database design, triggers can be used to enforce business rules and maintain data integrity.

Design Analysis

Before implementing we analyzed the design of the database, including the normalization level, the relationships between tables, and the use of constraints and triggers. This analysis will helped to develop an optimized database with better performance and data integrity.

Database Design (ER Diagram)



The ER diagram of the vehicle booking application database design provides a visual representation of the entities, attributes, relationships, and cardinalities of the system. It consists of the following entities:

users: This entity contains attributes like user_id, user_name, user_email, password, user_role

Customers: This entity contains attributes like customer_id, user_id, customer_phone, customer_address

admins: This entity contains attributes like admin_id, user_id, salary, phno

Vehicles: This entity contains attributes like vehicle_id, vehicle_type, service_type, registration_number, vehicle_rating, number_of_persons_rated.

Drivers: This entity contains attributes like driver_id, user_id, driver_phone, driver_address , driver_license_number, driver_rating, number_of_persons_rated.

bookings: This entity contains attributes like booking_id, customer_id, vehicle_id, driver_id, start_location, end_location, start_date, end_date, booking_price, paid, cancellation.

payments: This entity contains attributes like payment_id, booking_id, payment_date, payment_amount, payment_method, remarks.

reviews: This entity contains attributes like review_id, booking_id, review_comment, rating_for_driver, rating_for_vehicle

booking_cancellation: This entity contains attributes like cancellation_id, booking_id, cancellation_date, cancellation_reason

The relationships between these entities are represented by the cardinality and the line connecting them. The ER diagram provides a clear understanding of how these entities are related to each other and how the data flows within the system. It helps in identifying the key entities and their attributes, thus facilitating the creation of an efficient and effective database schema.

Implementation Details:-

The implementation of database schema includes the following sub sections

- Creation of Tables
- Imposing constraints for data integrity
- Integrating tables (establishing relationships)
- Sample tables data
- Triggers

All the above sub sections are incorporated with in the below SQL code

Code Section:-

```
# data base creation

create database Vehicle_Booking_Application;
use Vehicle_Booking_Application;


set sql_safe_updates = 0;

# tables creation inside the database

create table users (

    user_id int auto_increment primary key,
    user_name varchar(35) not null,
    user_email varchar(50) not null,
    user_password varchar(50) not null,
    user_role enum('customer', 'driver', 'admin') not null

);


create table customers (

     customer_id int auto_increment primary key,
     user_id int not null,
     customer_phone varchar(10) not null,
     customer_address varchar(100) not null,
     foreign key (user_id) references users (user_id)
);


create table drivers (

    driver_id int auto_increment primary key,
    user_id int not null,
    driver_phone varchar(10) not null,
    driver_address varchar(100) not null,
    driver_license_number varchar(20) not null,
    driver_rating decimal(4,2) not null default 3,
    number_of_persons_rated int not null default 1,
    foreign key (user_id) references users (user_id)
);

create table admins (

    admin_id int auto_increment primary key,
    user_id int not null,
    salary double not null,
    phno varchar(10) not null unique,
    foreign key (user_id) references users(user_id)

);


create table vehicles (

     vehicle_id int auto_increment primary key,
     vehicle_type varchar(20),
     service_type enum('local', 'rental') not null,
     registration_number varchar(10) not null unique,
     vehicle_rating decimal(4,2) not null default 3,
     number_of_persons_rated int not null default 1,
     driver_id int not null,
     foreign key(driver_id) references drivers(driver_id)

);
create table bookings (

    booking_id int auto_increment primary key,
    customer_id int not null,
    vehicle_id int not null,
    driver_id int not null,
    start_location varchar(50) not null,
    end_location varchar(50) not null,
    start_date datetime not null,
    end_date datetime not null,
    booking_price decimal(15,3) not null,
    paid boolean not null default 0,

    foreign key (vehicle_id) references vehicles (vehicle_id),
    foreign key (customer_id) references customers (customer_id),
    foreign key (driver_id) references drivers(driver_id)
);


# adding a column to bookings table

alter table bookings add cancellation varchar(10);

create table payments (
     payment_id int auto_increment primary key,
     booking_id int not null,
     payment_date datetime not null,
     payment_amount decimal(15,3) not null,
    payment_method enum('cash', 'credit', 'debit', 'paypal','upi') not null,
    remarks varchar(10),
    foreign key (booking_id) references bookings (booking_id)
);

create table reviews (

    review_id int auto_increment primary key,
    booking_id int not null,
    review_comment varchar(150),
    rating_for_driver int check(rating_for_driver>0 and rating_for_driver<=5),
    rating_for_vehicle int check(rating_for_vehicle>0 and rating_for_vehicle<=5),

    foreign key (booking_id) references bookings (booking_id)
);

create table booking_cancellations (

  cancellation_id int primary key auto_increment,
  booking_id int not null,
  cancellation_date datetime not null,
  cancellation_reason text,

  foreign key(booking_id) references bookings(booking_id)

);


# to get the count of number of tables in the database

select count(*) as number_of_tables from
information_schema.tables where table_schema = 'vehicle_booking_application';


# to show the list of tables

    show tables;

# inserting records into the users table

insert into users (user_name, user_email, user_password, user_role) values

('Admin User', 'adminuser@company.com', 'Adminpassword', 'admin'),

('Admin User2', 'adminuser2@company.com', 'Adminpassword2', 'admin'),

('John', 'john@gmail.com', 'Password123', 'customer'),

('saara', 'saara@yahoo.com', 'Saara@123', 'customer'),

('manoj', 'manoj@hotmail.com', 'Manoj@$123', 'driver'),

('eric', 'eric.k@gmail.com', 'Eric@567', 'driver'),
('karthik', 'karthik@gmail.com', 'Karthik@2325', 'customer'),
('tim', 'tim@yahoo.com', 'Tim@90', 'customer'),
('bob', 'bob@hotmail.com', 'Bob@780', 'driver'),
('lokesh', 'lokesh.p@gmail.com', 'Lokesh@567', 'customer'),
('joshna', 'joshna@gmail.com', 'Joshna@123', 'customer'),
('santhosh', 'santhosh@yahoo.com', 'santhosh@123', 'customer'),
('naveen', 'naveen@hotmail.com', 'Naveen@$123', 'customer'),
('kiran', 'kiran.n@gmail.com', 'Kiran@567', 'customer'),
('rahul', 'rahul@gmail.com', 'Rahul@2325', 'driver'),
('sam', 'sam@yahoo.com', 'Sam@90', 'customer'),
('ram charan', 'ramcharan@hotmail.com', 'Rc@780', 'customer'),
('arjun', 'arjun.p@gmail.com', 'arjun@567', 'driver'),
('keerthi', 'keerthi@gmail.com', 'Keerthi@123', 'customer'),
('nani', 'nani@yahoo.com', 'Nani@123', 'customer'),
('yashwanth', 'yashwanth@hotmail.com', 'Yaswanth@$123', 'customer'),
('pavan', 'pavan.pk@gmail.com', 'Pavan@567', 'customer'),
('das', 'das@gmail.com', 'Das@2325', 'driver'),
('iqbal', 'iqbal@gmail.com', 'Iqbal@123', 'customer'),
('kishore', 'kishore@yahoo.com', 'Kishore@123', 'customer'),
('pratap', 'pratap@hotmail.com', 'Pratap@$123', 'customer'),
('bhaskar', 'Bhas.Abd@gmail.com', 'Bhaskar@567', 'customer'),
('yogi', 'yogi@gmail.com', 'Yogi@567', 'customer'),
('vijay', 'vijay@gmail.com', 'Vijay@2325', 'driver'),
('suneel', 'suneel@gmail.com', 'Suneel@2325', 'driver'),
('raana', 'raana@gmail.com', 'Raana@2325', 'driver'),
('prakash', 'prakesh@gmail.com', 'Prakesh@232', 'driver'),
('vinay', 'vinay@gmail.com', 'Vinay@235', 'driver'),
('vikram', 'vikram@gmail.com', 'Vijay@325', 'driver'),
('dileep', 'dileep@gmail.com', 'Dileep@25', 'driver'),
('bhavesh', 'bhavesh@gmail.com', 'Bhavesh@57', 'driver'),
('ganesh', 'ganesh@gmail.com', 'Ganesh@225', 'driver');

# inserting records into the customers table
insert into customers (user_id, customer_phone, customer_address)  values
(3, '9775551234', '123 Main_St,chennai'),
(4, '7885555678', '456 Elm_St,bangalore'),
(7, '6775554321', '789 Oak_St,tirupati'),
(8, '9385558765', '321 Pine_St,chennai'),
(10, '7395552468', '654 Cedar_St,kerala'),
(11, '8945551357', '987 Maple_St,bangalore'),
(12, '6305558642', '234 Birch_St, kadapa'),
(13, '7305559753', '876 Walnut_St,kurnool'),
(14, '9305553698', '135 Cherry_St,chennai'),
(16, '8305557410', '468 Juniper_St,delhi'),
(17, '9775551230', '123 t-nagar,chennai'),
(19, '7885555671', '456 auto-nagar,bangalore'),
(20, '6775554322', '789 air-bypass-road,tirupati'),
(21, '9385558763', '321 jawahar_St,chennai'),
(22, '7395552464', '654 lonar_St,kerala'),
(24, '8945551355', '987 kolar_St,bangalore'),
(25, '6305558646', '234 kondareddy_St, kadapa'),
(26, '7305559757', '876 maidkur_St,kurnool'),
(27, '9305553697', '135 sarvana_St,chennai'),
(28, '8305557454', '468 kgf_St,bangalore');

# inserting records into the drivers table
insert into drivers (user_id, driver_phone, driver_address, driver_license_number)  values
(5, '6775512341', '789 air-bypass-road,tirupati', 'AP-14-2011-0062821'),
(6, '9385512342', '321 jawahar_St,chennai','TN-14-2005-0012342'),
(9, '8305512343', '468 kgf_St,bangalore','KA-14-2011-0034567'),
(15, '6775512344', '789 Oak_St,tirupati','AP-14-2003-0056789'),
(18, '9385512345', '321 Pine_St,chennai','TN-14-2009-0023456'),
(23, '7305512346', '876 Walnut_St,kurnool','AP-14-2022-0045678'),
(29, '9305512347', '135 Cherry_St,chennai','TN-14-2000-0089012'),
(30, '3456512341', '789 air-bypass-road,tirupati', 'AP-14-2009-0052821'),
(31, '6789512342', '321 jawahar_St,chennai','TN-14-2010-0089342'),
(32, '2345512343', '468 kgf_St,bangalore','KA-14-2021-0038967'),
(33, '5068512344', '789 Oak_St,tirupati','AP-14-2004-0001789'),
(34, '9043512345', '321 Pine_St,chennai','TN-14-2008-0027856'),
(35, '7098512346', '876 Walnut_St,kurnool','AP-14-2006-0045600'),
(36, '3056512347', '135 Cherry_St,chennai','TN-14-2002-0089090'),
(37, '5089512347', '135 Cherry_St,chennai','TN-14-2014-0089034');

# inserting records into the admins table
insert into admins(user_id, salary, phno) values
(1, 35000.00, 7396762250),
(2, 70000.00, 9381358153);

# inserting records into the vehicles table
insert into vehicles (vehicle_type, service_type,registration_number,driver_id)  values
('mini', 'local', 'AP21BP7331',1),
('mini', 'local', 'KA22JP7059',7),
('mini', 'rental', 'TN31BP8441',6),
('hatchback', 'local', 'AP03BP6745',5),
('hatchback', 'rental', 'TS05KJ4389',3),
('sedan', 'local', 'TN02BK7939',2),
('sedan', 'rental', 'AP23JP5089',8),
('SUV', 'local', 'KA30HP8932',10),
('SUV', 'local', 'TN21LM0089',12),
('SUV', 'rental', 'KA33LN9067',11),
('MUV', 'local', 'MH67LJ6703',13),
('MUV', 'local', 'PB89MN8560',14),
('MUV', 'Rental', 'DL70HJ5670',15),
('pickup trucks', 'local', 'KA86BN4578',4),
('pickup trucks', 'Rental', 'JP08TJ4530',9);

# inserting records into the bookings table
INSERT INTO bookings (customer_id, vehicle_id, driver_id,
start_location, end_location, start_date, end_date, booking_price)  VALUES
(1, 1, 1, '123 Main St, chennai', '456 Park Ave,chennai',
 '2023-04-10 10:00:00', '2023-04-10 11:00:00', 350.50),
(3, 2, 7, '789 Elm St,kerala', '1011 Oak Ave, kerala',
'2023-04-11 12:00:00', '2023-04-11 13:00:00', 250.00),
(5, 8, 10, '1213 Maple St,bangalore', '1415 Pine Ave,bangalore',
'2023-04-12 14:00:00', '2023-04-12 15:00:00', 300.75),
(7, 3, 6, '123 Main St, chennai', '456 Park Ave,chennai',
'2023-04-13 16:00:00', '2023-04-13 17:00:00', 1550.25),
(9, 9, 12, '1213 Maple St,bangalore', '1415 Pine Ave,bangalore',
'2023-04-14 18:00:00', '2023-04-14 19:00:00', 450.00),
(11, 11, 13, '123 Main St, chennai', '456 Park Ave,chennai',
'2023-04-15 20:00:00', '2023-04-15 21:00:00', 650.75),
(13, 6, 2, '2829 Maple St,tirupati', '3031 Cedar Ave, tirupati',
'2023-04-16 22:00:00', '2023-04-16 23:00:00', 180.50),
(15, 12, 14, '123 Main St, chennai', '456 Park Ave,chennai',
'2023-04-17 00:00:00', '2023-04-17 01:00:00', 870.00),
(17, 10, 11, '3637 Cedar St,bangalore', '3839 Birch Ave, bangalore',
 '2023-04-18 02:00:00', '2023-04-18 03:00:00', 1750.25),
(19, 13, 15, '123 sun St, chennai', '456 dark Ave,chennai',
'2023-04-19 04:00:00', '2023-04-19 05:00:00', 1600.75),
(2, 4, 5, '123 Main St, chennai', '456 Park Ave,chennai',
'2023-04-15 20:00:00', '2023-04-15 21:00:00', 650.75),
(4, 5, 3, '123 Main St, chennai', '456 Park Ave,chennai',
'2023-04-15 20:00:00', '2023-04-15 21:00:00', 1250.75),
(6, 7, 8, '3637 Cedar St,bangalore', '3839 Birch Ave, bangalore',
'2023-04-18 02:00:00', '2023-04-18 03:00:00', 1750.25),
(8, 14, 4, '123 Main St, chennai', '456 Park Ave,chennai',
'2023-04-17 20:00:00', '2023-04-17 21:00:00', 650.75),
(10, 15, 9, '2829 Maple St,tirupati', '3031 Cedar Ave, tirupati',
'2023-04-18 22:00:00', '2023-04-18 00:00:00', 1880.50),
(12, 8, 10, '1213 Maple St,bangalore', '1415 Pine Ave,bangalore',
'2023-04-12 18:00:00', '2023-04-14 19:00:00', 2450.00),
(14, 8, 10, '1213 Maple St,bangalore', '1415 Pine Ave,bangalore',
'2023-04-19 18:00:00', '2023-04-20 19:00:00', 950.00),
(16, 2, 7, '789 Elm St,kerala', '1011 Oak Ave, kerala',
'2023-04-13 12:00:00', '2023-04-13 13:00:00', 250.00),
(18, 15, 9, '2829 Maple St,tirupati', '3031 Cedar Ave, tirupati',
 '2023-04-01 22:00:00', '2023-04-04 00:00:00', 7890.50),
(20, 8, 10, '1213 Maple St,bangalore', '1415 Pine Ave,bangalore',
'2023-04-30 18:00:00', '2023-04-14 19:00:00', 2450.00);

# inserting data into the payments table
insert into payments(booking_id, payment_date, payment_amount, payment_method) values
(3,now(), 301,'upi'),
(6, now(), 651, 'debit'),
(1,now(), 350.5,'upi'),
(2, now(), 250, 'debit'),
(4,now(), 1550.250,'cash'),
(5, now(), 450, 'credit'),
(7,now(), 180.5,'upi'),
(8,now(), 870,'upi'),
(9, now(), 1750, 'debit'),
(11, now(), 650, 'debit'),
(13, now(), 1750, 'credit'),
(19,now(), 650.750,'upi'),
(20,now(), 1880.5,'cash'),
(21, now(), 2450, 'paypal'),
(24, now(), 7890.5, 'debit'),
(25, now(), 2450, 'credit'),
(12,now(),1250.750,'upi');

# inserting data into the reviews table
insert into reviews(booking_id, review_comment, rating_for_driver,rating_for_vehicle)  values
(1, "good", 5, 5),
(2, "worst experience", 2, 1),
(3, "nice", 4, 4),
(4, "great service", 5, 5),
(5, "Average", 3, 3),
(6, "not bad", 2.7, 3),
(7, "superb", 5, 5),
(8, "bad", 2, 3),
(11, "need to improve the vehicle condition", 4, 2),
(13, "good", 4, 5);
(19, "great service", 5, 5),
(20, "Average", 3, 3),
(25, "not bad", 2.7, 3);

# inserting records into the booking_cancellation

insert into booking_cancellations(booking_id, cancellation_date,cancellation_reason)  values
```

```
(3, now(), "i have some other resource");
(7, now(), "no reason"),
(24, now(), "postponed my work"),
(2, now(), "due to holiday"),
(12, now(), "i got my friend vehicle");


# -------------------------  TRIGGERS ---------------------------------------------- #
# trigger to automatically upadate the rating of the driver and vehicle
# when the customer give review and rating and it is updated with an average value.


delimiter //
create trigger update_ratings after insert on reviews
for each row
begin
  declare d_r decimal(4, 2);
  declare vh_rating decimal(4, 2);
  declare dr_id int;
  declare cur_dr_rating int;
  declare cur_npr_driver int;
  declare cur_customer_rating_for_dr int;

  declare vh_id int;
  declare cur_vh_rating int;
  declare cur_npr_vh int;
  declare cur_customer_rating_for_vh int;


# code for updating the rating of the corresponding driver with the average of rating given in the
#reviews
  set dr_id = (select driver_id from bookings where booking_id = new.booking_id);
  set cur_dr_rating = (select driver_rating from drivers where driver_id = dr_id);
  set cur_npr_driver = (select number_of_persons_rated from drivers where driver_id = dr_id);
  set cur_customer_rating_for_dr = (select rating_for_driver from reviews where review_id =
new.review_id);
  set d_r = ((cur_npr_driver*cur_dr_rating )+ cur_customer_rating_for_dr)/(cur_npr_driver+1);

  update drivers set driver_rating = d_r where driver_id = dr_id;
update drivers set number_of_persons_rated = cur_npr_driver+1 where driver_id=dr_id;
```

```
# code for updating the rating of the corresponding driver with the average of rating given in the
#reviews

  set vh_id = (select vehicle_id from bookings where booking_id = new.booking_id);
  set cur_vh_rating = (select vehicle_rating from vehicles where vehicle_id = vh_id);
  set cur_npr_vh = (select number_of_persons_rated from vehicles where vehicle_id = vh_id);
  set cur_customer_rating_for_vh = (select rating_for_vehicle from reviews where review_id =
new.review_id);
  set vh_rating = ((cur_npr_vh*cur_vh_rating) + cur_customer_rating_for_vh)/(cur_npr_vh+1);

  update vehicles set vehicle_rating = vh_rating where vehicle_id = vh_id;
  update vehicles set number_of_persons_rated = cur_npr_vh+1 where vehicle_id = vh_id;

end//
delimiter ;


# trigger to set the payemt status of the booking automatically after the payment done.


delimiter //

create trigger set_booking_paid after insert on payments
for each row
begin
  update bookings set paid = 1 where booking_id = new.booking_id;

end//
delimiter ;

# trigger to cancel the booking and refund the amount to the customer, and the changes are

# reflected in the bookings and payments table.


delimiter //

create trigger updating_the_booking_cancellation_and_refund_payment
after insert on booking_cancellations
for each row
begin

  update bookings set cancellation = "cancelled" where booking_id = new.booking_id;
  update payments set remarks = "refunded" where booking_id = new.booking_id;
end
delimiter ;
```

Results

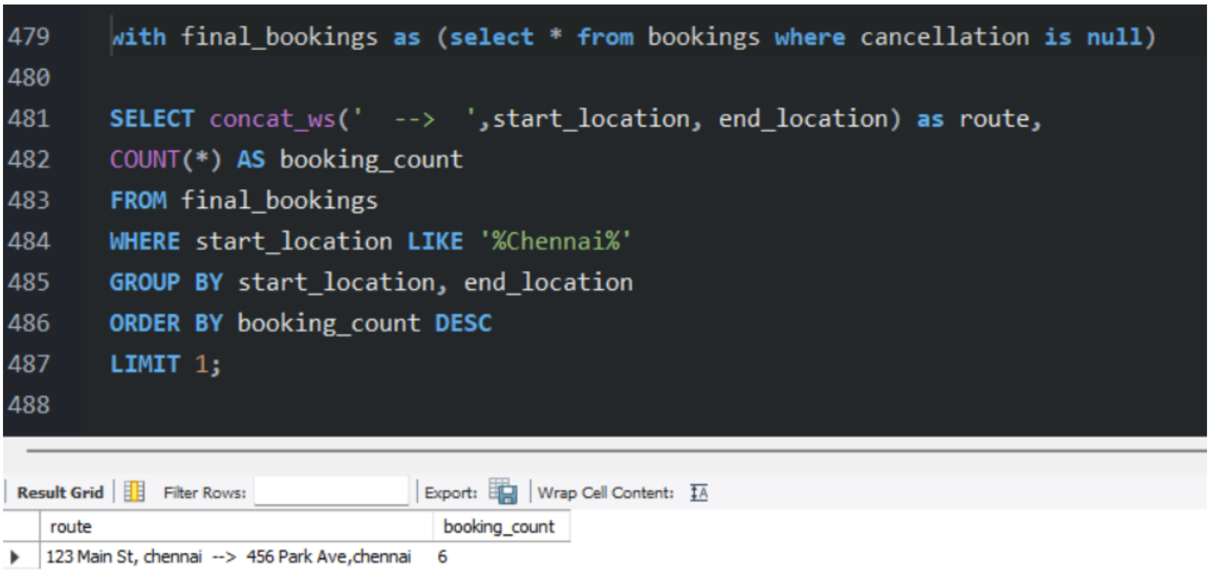
------------ **EVALUATING THE DATABASE SCHEMA WITH SOME QUERIES** -------------#

# Query-1

-- Populate the data and display the most frequently Booked Route in Chennai.

```
with final_bookings as (select * from bookings where cancellation is null)
SELECT concat_ws('  -->  ',start_location, end_location) as route,
COUNT(*) AS booking_count
FROM final_bookings
WHERE start_location LIKE '%Chennai%'
GROUP BY start_location, end_location
ORDER BY booking_count DESC
LIMIT 1;
```

# Output



```
with final_bookings as (select * from bookings where cancellation is null)

SELECT concat_ws('  -->  ',start_location, end_location) as route,
COUNT(*) AS booking_count
FROM final_bookings
WHERE start_location LIKE '%Chennai%'
GROUP BY start_location, end_location
ORDER BY booking_count DESC
LIMIT 1;
```

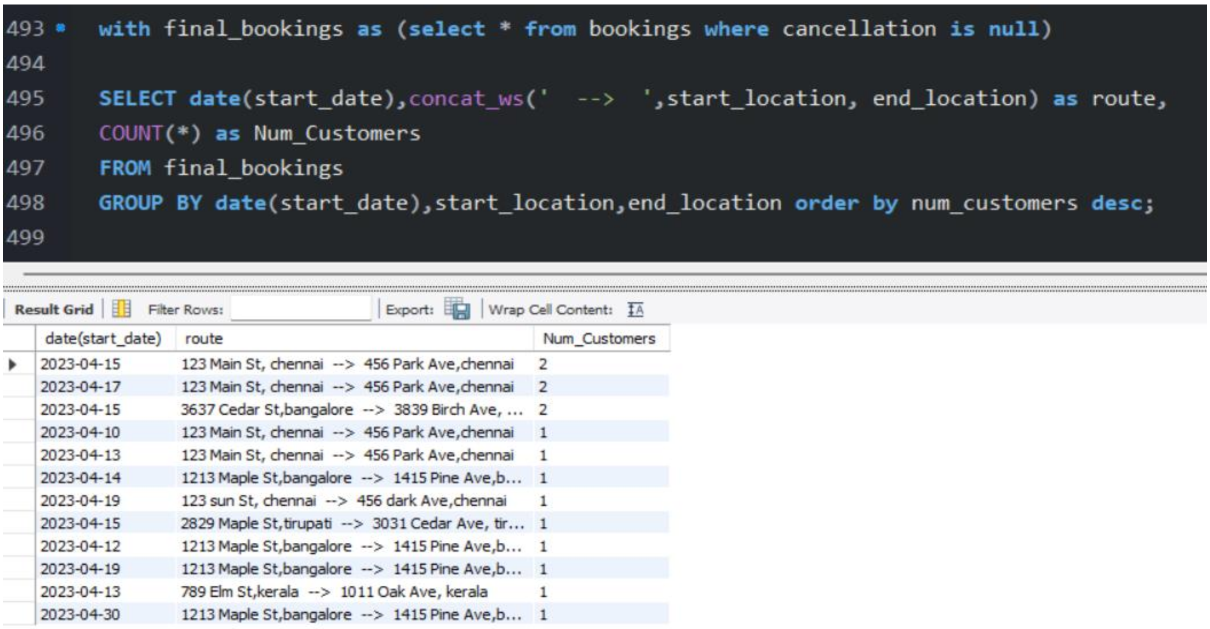
route	booking_count
123 Main St, chennai --> 456 Park Ave,chennai	6

# Query-2

-- Display the Number of Customers who travelled in "each date and in each Route".

```
with final_bookings as (select * from bookings where cancellation is null)
SELECT date(start_date),concat_ws('  -->  ',start_location, end_location) as route,
COUNT(*) as Num_Customers
FROM final_bookings
GROUP BY date(start_date),start_location,end_location order by num_customers desc;
```

# Output



```
with final_bookings as (select * from bookings where cancellation is null)

SELECT date(start_date),concat_ws('  -->  ',start_location, end_location) as route,
COUNT(*) as Num_Customers
FROM final_bookings
GROUP BY date(start_date),start_location,end_location order by num_customers desc;
```

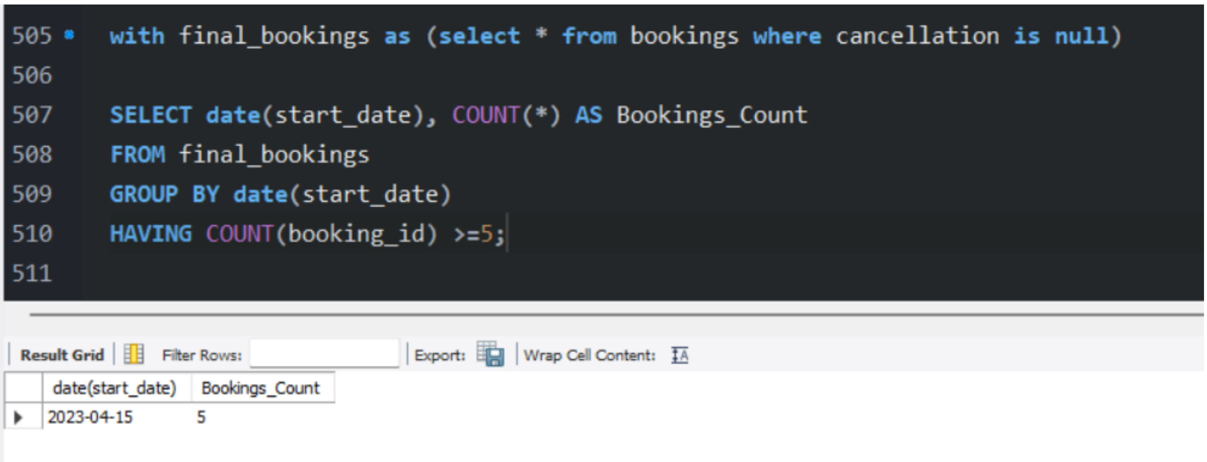
date(start_date)	route	Num_Customers
2023-04-15	123 Main St, chennai --> 456 Park Ave,chennai	2
2023-04-17	123 Main St, chennai --> 456 Park Ave,chennai	2
2023-04-15	3637 Cedar St,bangalore --> 3839 Birch Ave, ...	2
2023-04-10	123 Main St, chennai --> 456 Park Ave,chennai	1
2023-04-13	123 Main St, chennai --> 456 Park Ave,chennai	1
2023-04-14	1213 Maple St,bangalore --> 1415 Pine Ave,b...	1
2023-04-19	123 sun St, chennai --> 456 dark Ave,chennai	1
2023-04-15	2829 Maple St,tirupati --> 3031 Cedar Ave, tir...	1
2023-04-12	1213 Maple St,bangalore --> 1415 Pine Ave,b...	1
2023-04-19	1213 Maple St,bangalore --> 1415 Pine Ave,b...	1
2023-04-13	789 Elm St,kerala --> 1011 Oak Ave, kerala	1
2023-04-30	1213 Maple St,bangalore --> 1415 Pine Ave,b...	1

# Query -3

-- Display the Bookings count is >=5 in each Day.

```
with final_bookings as (select * from bookings where cancellation is null)
SELECT date(start_date), COUNT(*) AS Bookings_Count
FROM final_bookings
GROUP BY date(start_date)
HAVING COUNT(booking_id) >=5;
```

# Output



```
with final_bookings as (select * from bookings where cancellation is null)

SELECT date(start_date), COUNT(*) AS Bookings_Count
FROM final_bookings
GROUP BY date(start_date)
HAVING COUNT(booking_id) >=5;

```

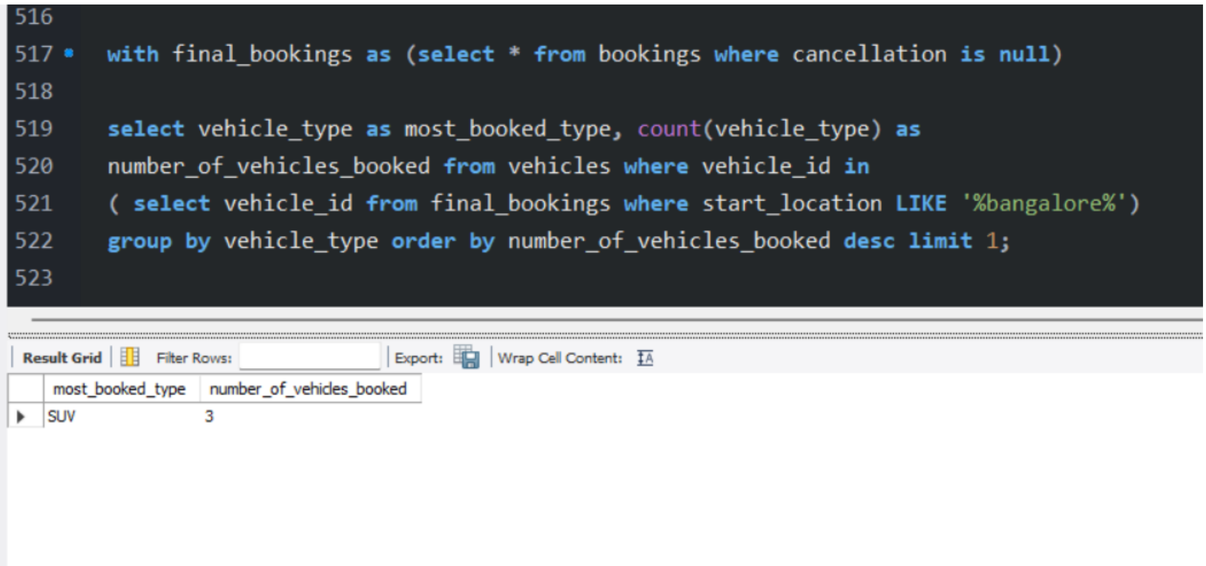
date(start_date)	Bookings_Count
2023-04-15	5

# Query -4

-- List out most booked type of Vehicle (SUV, Mini etc.,) in Bangalore City

with final_bookings as (select * from bookings where cancellation is null)
Select vehicle_type as most_booked_type, count(vehicle_type) as
number_of_vehicles_booked from vehicles where vehicle_id in
(select vehicle_id from final_bookings where start_location LIKE '%bangalore%')
group by vehicle_type order by number_of_vehicles_booked desc limit 1;

# Output



```

with final_bookings as (select * from bookings where cancellation is null)

select vehicle_type as most_booked_type, count(vehicle_type) as
number_of_vehicles_booked from vehicles where vehicle_id in
( select vehicle_id from final_bookings where start_location LIKE '%bangalore%')
group by vehicle_type order by number_of_vehicles_booked desc limit 1;

```

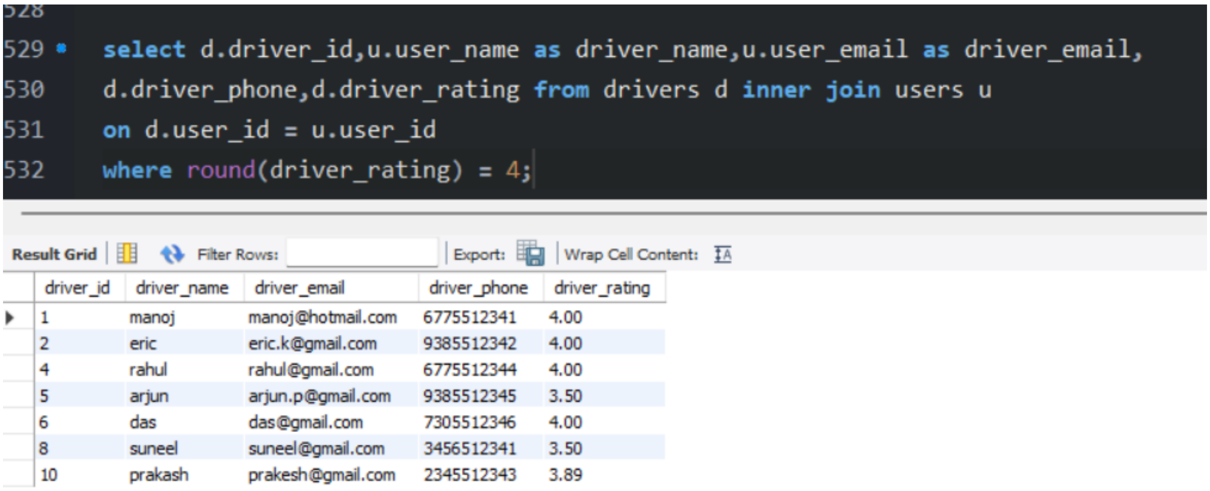
most_booked_type	number_of_vehicles_booked
SUV	3

# Query-5

-- Display the 4 star rated driver details to initiate incentives by Company.

select d.driver_id,u.user_name as driver_name,u.user_email as driver_email,
d.driver_phone,d.driver_rating from drivers d inner join users u
on d.user_id = u.user_id
where round(driver_rating) = 4;

# Output



```
select d.driver_id,u.user_name as driver_name,u.user_email as driver_email,
d.driver_phone,d.driver_rating from drivers d inner join users u
on d.user_id = u.user_id
where round(driver_rating) = 4;
```

driver_id	driver_name	driver_email	driver_phone	driver_rating
1	manoj	manoj@hotmail.com	6775512341	4.00
2	eric	eric.k@gmail.com	9385512342	4.00
4	rahul	rahul@gmail.com	6775512344	4.00
5	arjun	arjun.p@gmail.com	9385512345	3.50
6	das	das@gmail.com	7305512346	4.00
8	suneel	suneel@gmail.com	3456512341	3.50
10	prakash	prakesh@gmail.com	2345512343	3.89

Test Cases

Major test cases that are considered in designing this "vehicle booking application" database:

[1] User Registration and Logins:

Verify that a user can successfully register with valid credentials.
Verify that a user cannot register with invalid or incomplete credentials.
Verify that a user cannot register with an existing email or phone number.
Verify that a user receives a confirmation email after successful registration.
Verify that a user can reset their password in case they forget it.
Verify that the user's information is securely stored in the database.

[2] vehicle Booking:

Verify that a user can successfully book a vehicle with valid start and end locations.
Verify that a user cannot book a vehicle with invalid or incomplete start and end locations.
Verify that the user can view the estimated fare before booking.
Verify that the user receives a confirmation message after booking.
Verify that the user can cancel a vehicle within a certain time limit.

[3] Payment:

Verify that a user can successfully make a payment for a vehicle booking using valid payment methods.
Verify that a user cannot make a payment with invalid or incomplete payment methods.
Verify that a user cannot make a payment with insufficient funds or expired payment methods.
Verify that the user's payment information is encrypted and securely stored in the database.
Verify that the user receives an invoice for the vehicle booking after completing it.

[4] User Ratings:

Verify that a user can rate a driver after completing user trip or journey.
Verify that a user can rate a vehicle condition after completing user trip or journey.
Verify that the user's rating is saved and the corresponding driver and vehicle average rating is updated.
Verify that the user can view the driver's rating and feedback before booking a vehicle.

Conclusion & Future Scope

“ Vehicle booking application” database design has been designed to store all the necessary information related to users, customers, vehicles, drivers, bookings, payments, booking_cancellation, reviews.
In conclusion, the database design is capable of efficiently handling of data, ensuring data integrity, and providing fast access to information.

However, there is always room for improvement and future scope to enhance the system's features and functionalities, such as:

- Implementing a more sophisticated route optimization algorithm to improve the efficiency of vehicle routing.
- Implementing a notification system to keep customers informed about booking confirmation, driver information, and vehicle status.

Overall, the vehicle booking application database design can be extended with more features to enhance the booking experience of customers and make it more efficient for the service provider.

References and resources used

https://www.slideshare.net/sotbar7/car-rental-agency-database-mysql

https://dev.mysql.com/doc/

https://www.w3schools.com/MySQL/default.asp