Trinity College Dublin
Coláiste na Trionóide, Baile Átha Cliath
The University of Dublin

CSU 44D01 Project Specification

MySQL Carpooling Project Report

Name: SHRUTI KATHURIA

TCD ID:21355061

Table of Contents of Project Report

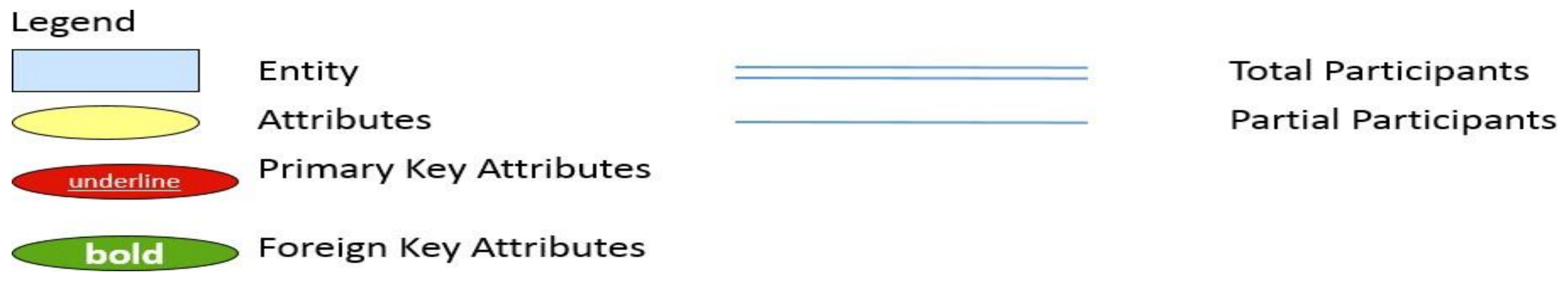
1. Application Description:

Carpool is a good option when you are traveling alone and want to share your ride. The travelers can share travel charges between them and can enjoy the ride with low cost. This database is designed for a website which can be developed for carpooling to share ride. The owner of the car share his/her ride with pickup and drop-off locations and also can add stopovers. The ride is published on the website and then traveler searches for rides. Traveler book the seat(s) for specific amount and at the end traveler can give the rating to the car owner.

Car owner and traveler can communicate with each other to send messages after booking the ride. Each car owner and riders should have valid account and they can offer or book the rides after email and mobile verifications.

2. Entity Relationship Diagram (ERD):

The following diagram represents relationship of entities (tables).



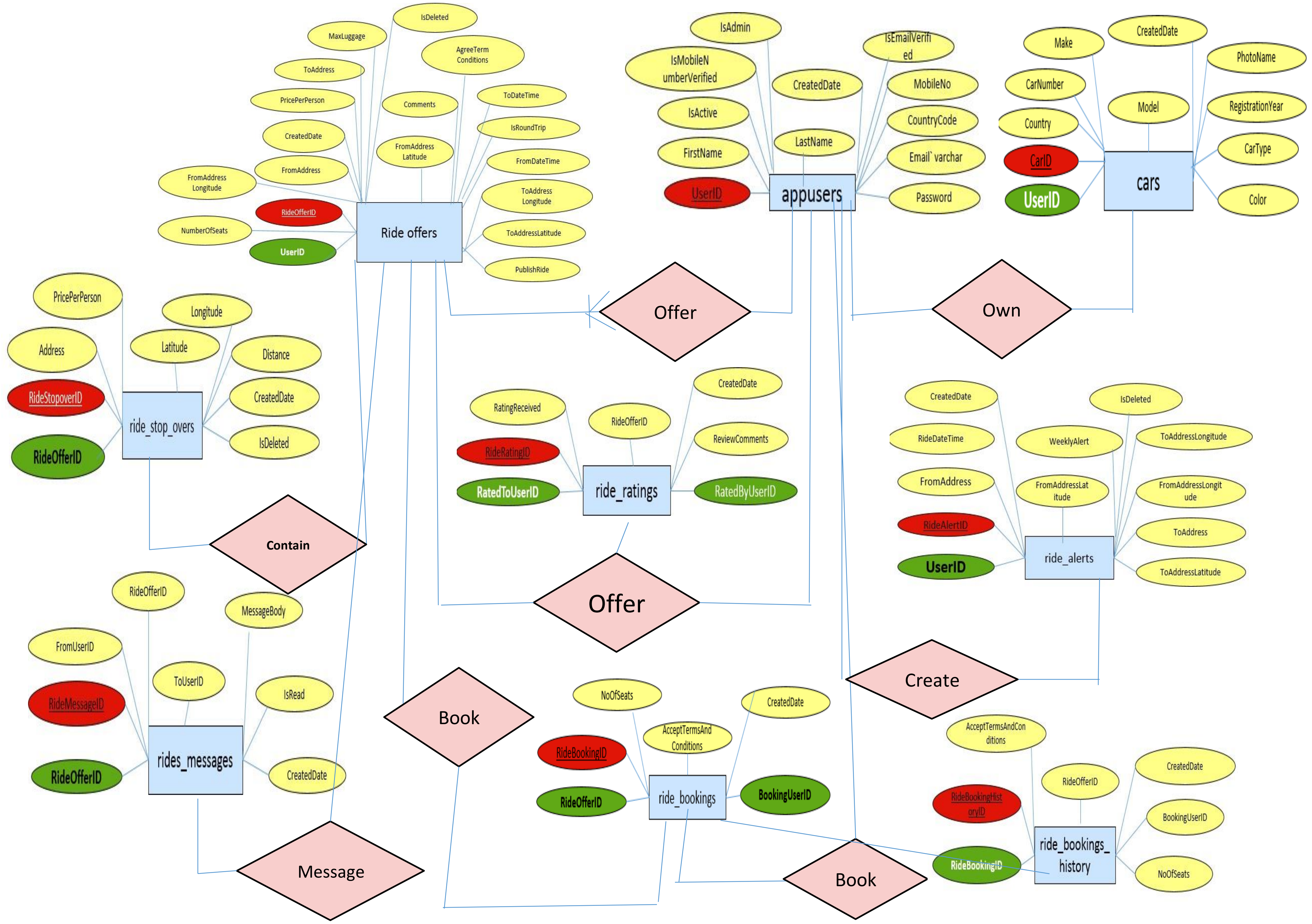
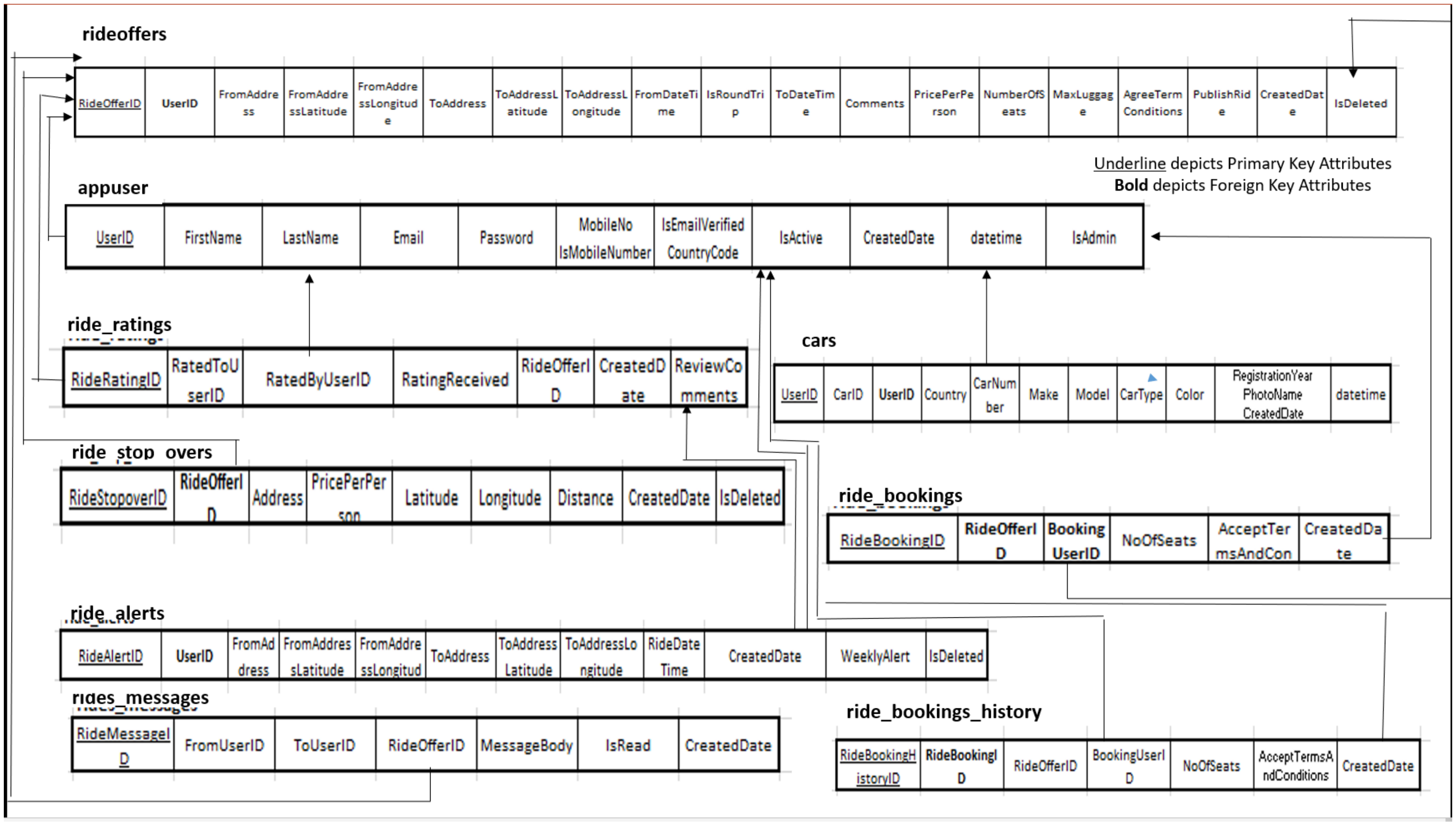

IsDeleted
MaxLuggage
AgreeTerm Conditions
ToAddress
PricePerPerson
Comments
ToDateTime
IsRoundTrip
CreatedDate
FromAddress Latitude
FromDateTime
FromAddress
FromAddress Longitude
ToAddress Longitude
RideOfferID
NumberOfSeats
Ride offers
ToAddressLatitude
UserID
PublishRide
IsAdmin
IsMobileNumberVerified
IsEmailVerified
CreatedDate
MobileNo
IsActive
CountryCode
LastName
FirstName
Email` varchar
UserID
appusers
Password
Make
CreatedDate
PhotoName
CarNumber
Model
RegistrationYear
Country
CarID
CarType
cars
UserID
Color
Offer
Own
PricePerPerson
Longitude
Address
Latitude
Distance
RideStopoverID
CreatedDate
ride_stop_overs
IsDeleted
RideOfferID
CreatedDate
RatingReceived
RideOfferID
ReviewComments
RideRatingID
RatedToUserID
ride_ratings
RatedByUserID
CreatedDate
IsDeleted
RideDateTime
WeeklyAlert
ToAddressLongitude
FromAddress
FromAddressLatitude
FromAddressLongitude
RideAlertID
ToAddress
UserID
ride_alerts
ToAddressLatitude
Contain
Offer
RideOfferID
MessageBody
FromUserID
ToUserID
IsRead
RideMessageID
rides_messages
RideOfferID
CreatedDate
Create
Book
NoOfSeats
CreatedDate
AcceptTermsAnd Conditions
RideBookingID
RideOfferID
ride_bookings
BookingUserID
AcceptTermsAndConditions
CreatedDate
RideOfferID
RideBookingHistoryID
BookingUserID
RideBookingID
ride_bookings_history
NoOfSeats
Message
Book

Figure 1. Entity Relationship Diagram

3. Mapping to Relational Schema (explaining mapping rules used for each Table)



We have created the database named as **carpooling**. Consisting following entities or tables with their respective attributes or columns.

- **appusers** – UserID is unique and auto incremented primary key for this table.

appusers table is mapped with many tables to know which user has offered the ride, booked the ride, rated the rider etc. IsAdmin column describes that if user is admin then it should show him/her admin layout and can DELETE, INACTIVE the users and can perform all admin roles.

- **cars** – CarID is unique and auto incremented primary key for this table.

This table is mapped with appusers with UserID, which describes that which user is owner of the car.

- **ride_offers** – RideOfferID is the unique id and auto incremented column for this table. This column is primary key also.

This table has foreign key UserID with appusers. Which describes that which user has offered this ride. In this table user add values regarding ride offer such as: FromAddress, ToAddress, PricePerPerson, NoOfSeats, MaxLuggage etc.

- **ride_bookings** – RideBookingID is the unique id and auto incremented column for this table. This column is primary key also.

This table is mapped with two tables **appusers** and **ride_offers**, **BookingUserId** column explains which user has booked the ride and **RideOfferID** column explains that which ride offer is booked. **NoOfSeats** explains that how much seats are booked.

- **ride_bookings_history** – This table has RideBookingHistoryID primary key column and it is auto incremented. This table is repo of ride_bookings table which create history of ride_bookings table and record is saved in this table when any record is inserted in ride_bookings table. Ride_bookings table trigger is fired after insert.

- **ride_stop_overs** – This table column RideStopOverID is primary key, unique and auto incremented. Stop overs are saved in this table. This table is mapped with **ride_offers** which explains that which ride offer is mapped with stop over. This table has **one-to-many relationship with ride_offers** table because one ride offer can have more than one stop overs.

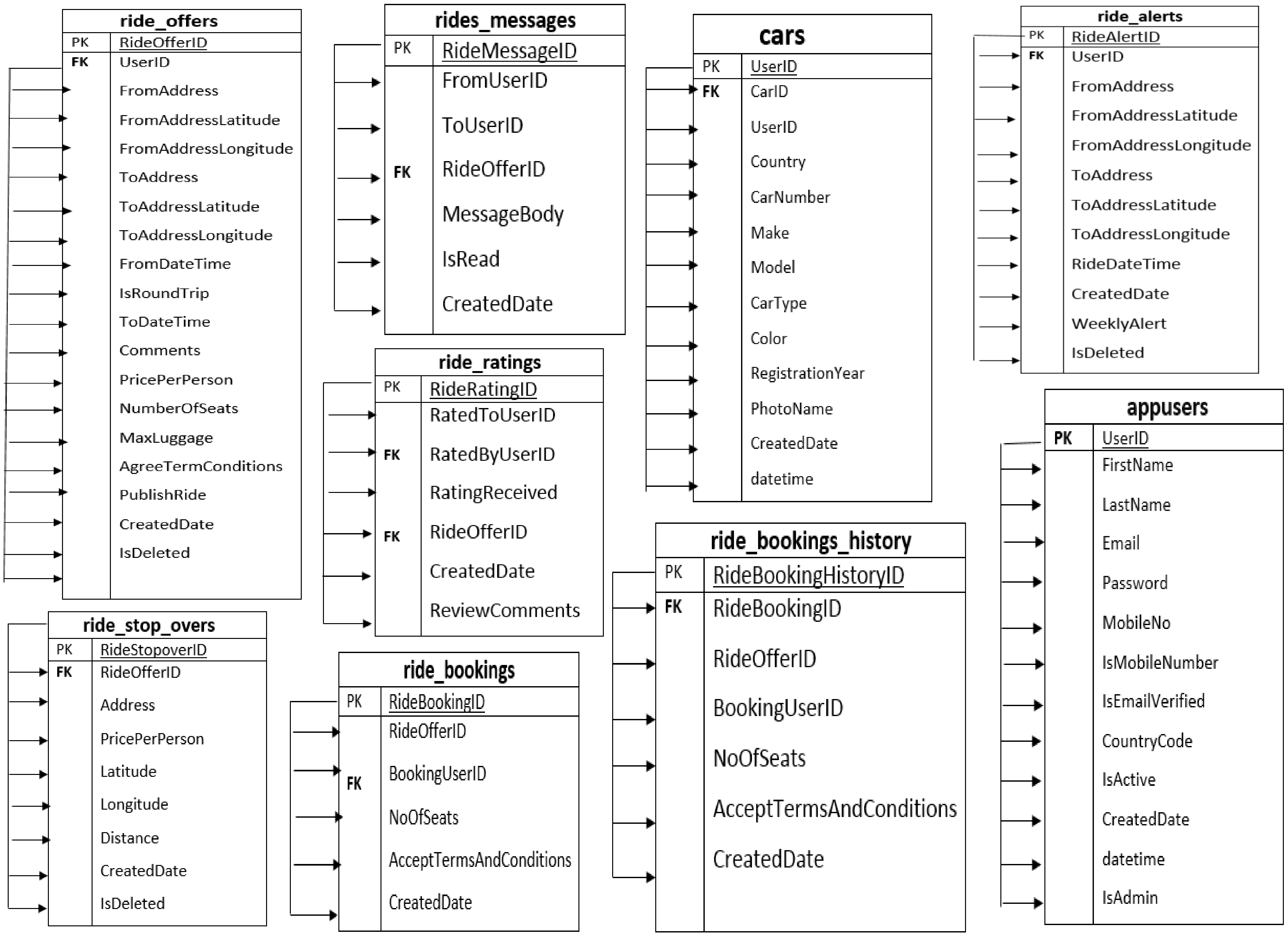
- **ride_alerts** – RideAlertID is the unique id and auto incremented column for this table. This column is primary key also. This user has foreign key with appusers table. This table saves records to show ride alerts if any user add offer and when it meets criteria with this alert then alert email can be sent to the user.

- **ride_ messages** – RideMessageID is the unique id and auto incremented column for this table. This column is primary key also. This table has foreign keys with appusers and ride_offers table to know that which user message is in table and it point to which ride.

- **ride_ratings** – RideRatingID is primary key and auto incremented column. This user is mapped with appusers table which explain that which user’s rating is added.

4. Functional Dependency Diagram

- **PK** depicts Primary Key Attributes with is underlined
- **FK** depicts Foreign Key Attributes



	ride_offers
PK	RideOfferID
FK	UserID
	FromAddress
	FromAddressLatitude
	FromAddressLongitude
	ToAddress
	ToAddressLatitude
	ToAddressLongitude
	FromDateTime
	IsRoundTrip
	ToDateTime
	Comments
	PricePerPerson
	NumberOfSeats
	MaxLuggage
	AgreeTermConditions
	PublishRide
	CreatedDate
	IsDeleted

	rides_messages
PK	RideMessageID
	FromUserID
	ToUserID
FK	RideOfferID
	MessageBody
	IsRead
	CreatedDate

	cars
PK	UserID
FK	CarID
	UserID
	Country
	CarNumber
	Make
	Model
	CarType
	Color
	RegistrationYear
	PhotoName
	CreatedDate
	datetime

	ride_alerts
PK	RideAlertID
FK	UserID
	FromAddress
	FromAddressLatitude
	FromAddressLongitude
	ToAddress
	ToAddressLatitude
	ToAddressLongitude
	RideDateTime
	CreatedDate
	WeeklyAlert
	IsDeleted

	ride_ratings
PK	RideRatingID
	RatedToUserID
FK	RatedByUserID
	RatingReceived
FK	RideOfferID
	CreatedDate
	ReviewComments

	appusers
PK	UserID
	FirstName
	LastName
	Email
	Password
	MobileNo
	IsMobileNumber
	IsEmailVerified
	CountryCode
	IsActive
	CreatedDate
	datetime
	IsAdmin

	ride_stop_overs
PK	RideStopoverID
FK	RideOfferID
	Address
	PricePerPerson
	Latitude
	Longitude
	Distance
	CreatedDate
	IsDeleted

	ride_bookings
PK	RideBookingID
	RideOfferID
FK	BookingUserID
	NoOfSeats
	AcceptTermsAndConditions
	CreatedDate

	ride_bookings_history
PK	RideBookingHistoryID
FK	RideBookingID
	RideOfferID
	BookingUserID
	NoOfSeats
	AcceptTermsAndConditions
	CreatedDate

1. Explanation and SQL Code for creating the database Tables (including any constraints)

1. Appusers table:

This table contains records of registered users using the carpooling application.

Below is the script of the table:

```
CREATE TABLE `appusers` (
  `UserID` int NOT NULL AUTO_INCREMENT,
  `FirstName` varchar(45) NOT NULL,
  `LastName` varchar(45) NOT NULL,
  `Email` varchar(45) NOT NULL,
  `Password` varchar(45) NOT NULL,
  `MobileNo` varchar(45) NOT NULL,
  `IsMobileNumberVerified` tinyint DEFAULT '0',
  `IsEmailVerified` tinyint DEFAULT '0',
  `CountryCode` varchar(45) DEFAULT NULL,
  `IsActive` tinyint DEFAULT '1',
  `CreatedDate` datetime(6) DEFAULT NULL,
  `IsAdmin` tinyint DEFAULT '0',
  PRIMARY KEY (`UserID`)
)
```

2. Cars table:

This table contains records related to cars. This table contains foreign key UserID with appusers table which explains that which user is owner of the car. Both tables have one-to-one relationship.

Below is the script of the table:

```
CREATE TABLE `cars` (
  `CarID` int NOT NULL AUTO_INCREMENT,
  `UserID` int NOT NULL,
  `Country` varchar(45) NOT NULL,
  `CarNumber` varchar(45) NOT NULL,
  `Make` varchar(45) NOT NULL,
  `Model` varchar(45) NOT NULL,
  `CarType` varchar(45) DEFAULT NULL,
  `Color` varchar(45) DEFAULT NULL,
  `RegistrationYear` int NOT NULL,
  `PhotoName` varchar(45) DEFAULT NULL,
  `CreatedDate` datetime(6) NOT NULL,
  PRIMARY KEY (`CarID`),
  KEY `appuser_cars_idx` (`UserID`),
  CONSTRAINT `appuser_cars` FOREIGN KEY (`UserID`) REFERENCES `appusers` (`UserID`)
)
```

3. **ride_offers table:**

Car owner Ride offers are saved in this table. . This table contains foreign key UserID with appusers table which explains that which user offered the ride. Both tables have one-to-one relationship. Car owner (user) add price per person, from address, to address and system calculates latitudes and longitudes for that address using google geocode API and add in ride_offers table. Also user add comments, noOfSeats, MaxLuggage etc.

Below is the script of the table:

```
CREATE TABLE `ride_offers` (
 `RideOfferID` int NOT NULL AUTO_INCREMENT,
 `UserID` int NOT NULL,
 `FromAddress` varchar(45) NOT NULL,
 `FromAddressLatitude` float NOT NULL,
 `FromAddressLongitude` float NOT NULL,
 `ToAddress` varchar(45) NOT NULL,
 `ToAddressLatitude` float NOT NULL,
 `ToAddressLongitude` float NOT NULL,
 `FromDateTime` datetime NOT NULL,
 `IsRoundTrip` tinyint NOT NULL DEFAULT '0',
 `ToDateTime` datetime DEFAULT NULL,
 `Comments` varchar(200) DEFAULT NULL,
 `PricePerPerson` int NOT NULL,
 `NumberOfSeats` int NOT NULL,
 `MaxLuggage` varchar(45) DEFAULT NULL,
 `AgreeTermConditions` tinyint NOT NULL DEFAULT '1',
 `PublishRide` tinyint NOT NULL DEFAULT '1',
 `CreatedDate` datetime(6) NOT NULL,
 `IsDeleted` tinyint NOT NULL DEFAULT '0',
 PRIMARY KEY (`RideOfferID`),
 KEY `riderOffers_UserId_idx` (`UserID`),
 CONSTRAINT `rideoffer_users` FOREIGN KEY (`UserID`) REFERENCES `appusers` (`UserID`)
)
```

4. ride_stop_overs table:

This table contains stop overs of the ride offered by car owner. This table has one-to-many relationship with ride_offers tables because there can be more than one stop overs for a long trip. Car owner (user) add price per person and address for that stop over and system calculates latitudes and longitudes for that address and add in ride_stop_overs table.

Below is the script of the table:

```
CREATE TABLE `ride_stop_overs` (
  `RideStopoverID` int NOT NULL AUTO_INCREMENT,
  `RideOfferID` int NOT NULL,
  `Address` varchar(500) NOT NULL,
  `PricePerPerson` int NOT NULL,
  `Latitude` float NOT NULL,
  `Longitude` float NOT NULL,
  `Distance` float NOT NULL,
  `CreatedDate` datetime NOT NULL,
  `IsDeleted` tinyint NOT NULL DEFAULT '0',
  PRIMARY KEY (`RideStopoverID`),
  KEY `stopover_offer_idx` (`RideOfferID`),
  CONSTRAINT `stopover_offer` FOREIGN KEY (`RideOfferID`) REFERENCES `ride_offers` (`RideOfferID`)
)
```

5. **ride_bookings table:**

This table has records of the ride bookings. When user book a ride offer then record is saved in this table. This table has RideOfferID columns which describes that which offer has been booked and BookingUserID columns has relationship with appusers table which describes that which user has booked the ride.

Below is the script of the table:

```
CREATE TABLE `ride_bookings` (
  `RideBookingID` int NOT NULL AUTO_INCREMENT,
  `RideOfferID` int NOT NULL,
  `BookingUserID` int NOT NULL,
  `NoOfSeats` int NOT NULL DEFAULT '1',
  `AcceptTermsAndConditions` tinyint NOT NULL DEFAULT '1',
  `CreatedDate` datetime(6) NOT NULL,
  PRIMARY KEY (`RideBookingID`),
  KEY `users_booking_idx` (`BookingUserID`),
  KEY `offer_booking_idx` (`RideOfferID`),
  CONSTRAINT `offer_booking` FOREIGN KEY (`RideOfferID`) REFERENCES `ride_offers` (`RideOfferID`),
  CONSTRAINT `users_booking` FOREIGN KEY (`BookingUserID`) REFERENCES `appusers` (`UserID`)
)
```

6. **rides_messages table:**

In this table messages are saved which are sent between car owner and the user who booked the ride.

Below is the script of the table:

```
CREATE TABLE `rides_messages` (
 `RideMessageID` int NOT NULL AUTO_INCREMENT,
 `FromUserID` int NOT NULL,
 `ToUserID` int NOT NULL,
 `RideOfferID` int NOT NULL,
 `MessageBody` varchar(3000) NOT NULL,
 `IsRead` tinyint NOT NULL DEFAULT '0',
 `CreatedDate` datetime(6) NOT NULL,
 PRIMARY KEY (`RideMessageID`),
 KEY `messages_offer_idx` (`RideOfferID`),
 CONSTRAINT `messages_offer` FOREIGN KEY (`RideOfferID`) REFERENCES `ride_offers` (`RideOfferID`)
)
```

7. **ride_ratings table:**

When ride has been completed then user can rate to the car owner. This is helpful for other users whether they should travel with that rider in future.

Below is the script of the table:

```
CREATE TABLE `ride_ratings` (
 `RideRatingID` int NOT NULL AUTO_INCREMENT,
 `RatedToUserID` int NOT NULL,
 `RatedByUserID` int NOT NULL,
 `RatingReceived` int NOT NULL,
 `RideOfferID` int DEFAULT NULL,
 `CreatedDate` datetime(6) NOT NULL,
 `ReviewComments` varchar(2000) DEFAULT NULL,
 PRIMARY KEY (`RideRatingID`),
 KEY `user_offer_idx` (`RideOfferID`),
 KEY `userTo_rating_idx` (`RatedToUserID`),
 KEY `userFrom_rating_idx` (`RatedByUserID`),
 CONSTRAINT `rating_offer` FOREIGN KEY (`RideOfferID`) REFERENCES `ride_offers` (`RideOfferID`),
 CONSTRAINT `userBy_rating` FOREIGN KEY (`RatedByUserID`) REFERENCES `appusers` (`UserID`),
 CONSTRAINT `userTo_rating` FOREIGN KEY (`RatedToUserID`) REFERENCES `appusers` (`UserID`)
)
```

8. **ride_alerts table:**

If any traveler travel frequently and there is no suitable ride for the user then user can add information for required ride in future. If any car owner offer a ride in application then ride alert is sent to the user then he/she can book that ride.

Below is script of the table:

```
CREATE TABLE `ride_alerts` (
 `RideAlertID` int NOT NULL AUTO_INCREMENT,
 `UserID` int NOT NULL,
 `FromAddress` varchar(500) NOT NULL,
 `FromAddressLatitude` float NOT NULL,
 `FromAddressLongitude` float NOT NULL,
 `ToAddress` varchar(500) NOT NULL,
 `ToAddressLatitude` float NOT NULL,
 `ToAddressLongitude` float NOT NULL,
 `RideDateTime` datetime(6) NOT NULL,
 `CreatedDate` datetime(6) NOT NULL,
 `WeeklyAlert` tinyint DEFAULT '1',
 `IsDeleted` tinyint DEFAULT '0',
 PRIMARY KEY (`RideAlertID`),
 KEY `appusers_ridealerts_idx` (`UserID`),
 CONSTRAINT `appusers_ridealerts` FOREIGN KEY (`UserID`) REFERENCES `appusers` (`UserID`)
)
```

9. ride_bookings_history table:

History of booking is saved in this table.

Below is the script of this table:

```
CREATE TABLE `ride_bookings_history` (
  `RideBookingHistoryID` int NOT NULL AUTO_INCREMENT,
  `RideBookingID` int NOT NULL,
  `RideOfferID` int NOT NULL,
  `BookingUserID` int NOT NULL,
  `NoOfSeats` int NOT NULL DEFAULT '1',
  `AcceptTermsAndConditions` tinyint NOT NULL DEFAULT '1',
  `CreatedDate` datetime(6) NOT NULL,
  PRIMARY KEY (`RideBookingHistoryID`),
  KEY `history_booking_idx` (`RideBookingID`),
  CONSTRAINT `history_booking` FOREIGN KEY (`RideBookingID`) REFERENCES `ride_bookings` (`RideBookingID`)
)
```

2. Explanation and SQL Code for Altering tables

1. Alter column name of the table:

Below is the script for altering cars table:

```
ALTER TABLE `carpooling`.`cars`
CHANGE COLUMN `CountryCode` `Country` VARCHAR(45) NOT NULL ;
```

2. Alter table to create foreign key constraint:

Below is the script to alter table ride_bookings to add foreign key constraint with ride_offers table on RideOfferID column. One RideOffer can have more than one bookings for more than one users (travelers).

```
ALTER TABLE `carpooling`.`ride_bookings`
ADD INDEX `rideBookings_offerid_idx` (`RideOfferID` ASC) VISIBLE;
ALTER TABLE `carpooling`.`ride_bookings`
ADD CONSTRAINT `rideBookings_offerid`
  FOREIGN KEY (`RideOfferID`)
  REFERENCES `carpooling`.`ride_offers` (`RideOfferID`)
  ON DELETE NO ACTION
```

```
 ON UPDATE NO ACTION;
```

3. **Alter table name:**

Below is the code to alter table name from ridestopovers to ride_stop_overs

```
ALTER TABLE `carpooling`.`ridestopovers`

RENAME TO  `carpooling`.`ride_stop_overs` ;
```

3. Explanation and SQL Code for Triggering operations:

Trigger is executed when INSERT, UPDATE OR DELETE operation is done on the table. Trigger can be execute before or after the operation.

Following is the syntax of creating trigger on a table:

```
CREATE TRIGGER triggerName

{BEFORE | AFTER} {INSERT | UPDATE| DELETE }

ON tableName FOR EACH ROW

trigger_body;
```

4. Explanation and SQL Code for Creating Views

View is like a table based on the result-set of SQL query. A view contains rows and columns, just like a table. The fields in a view are fields from one or more tables in the DB.

Following is SQL code syntax to create view:

```
CREATE VIEW view_name

AS SELECT column1, column2,...

FROM table_name

WHERE condition;
```

Below is SQL Code of the view which retrieve ratings of the users. It gets ratings from ride_ratings table and users details from appusers table using inner join between both tables.

```
USE `carpooling`;

CREATE  OR REPLACE VIEW `user_ratings` AS

SELECT      ride_ratings.RatedToUserID, ROUND(AVG(CAST(ride_ratings.RatingReceived AS Float)), 1) AS
UserRating,

appusers.FirstName, appusers.LastName, appusers.Email
```

```
FROM          ride_ratings

INNER JOIN appusers ON ride_ratings.RatedToUserID = appusers.UserID

GROUP BY ride_ratings.RatedToUserID, AppUsers.FirstName, AppUsers.LastName, AppUsers.Email;
```

5. Explanation and SQL Code for Populating the Tables

To insert a record into a table, use the INSERT statement. Below are INSERT statements for the tables to populate thedata.

Inserting into Table: AppUsers

```
INSERT INTO AppUsers (FirstName, LastName, Email, `Password`, MobileNo, IsMobileNumberVerified,
IsEmailVerified, CountryCode, IsActive, CreatedDate, IsAdmin) VALUES (' shruti', ' kathuria', 'user@domain.com',
'123Abc@', 03331234567, 1, 1, 'India', 1, NOW(), 1)

INSERT INTO AppUsers (FirstName, LastName, Email, `Password`, MobileNo, IsMobileNumberVerified,
IsEmailVerified, CountryCode, IsActive, CreatedDate, IsAdmin) VALUES ('TestUser', 'userLastName',
'user@domain.com', '123Abc@', 03331234567, 1, 1, 'India', 1, NOW(), 0)

INSERT INTO AppUsers (FirstName, LastName, Email, `Password`, MobileNo, IsMobileNumberVerified,
IsEmailVerified, CountryCode, IsActive, CreatedDate, IsAdmin) VALUES ('TestUser2', 'userLastName2',
'user2@domain.com', '123Abc@', 03331234567, 1, 1, 'India', 1, NOW(), 0)

INSERT INTO AppUsers (FirstName, LastName, Email, `Password`, MobileNo, IsMobileNumberVerified,
IsEmailVerified, CountryCode, IsActive, CreatedDate, IsAdmin) VALUES ('TestUser3', 'userLastName3',
'user3@domain.com', '123Abc@', 03331234567, 1, 1, 'India', 1, NOW(), 0)
```

Inserting into Table: cars

```
INSERT INTO `carpooling`.`cars`
(`UserID`,`Country`,`CarNumber`,`Make`,`Model`,`CarType`,`Color`,`RegistrationYear`,`PhotoName`,`CreatedDate`
)

VALUES (1, 'India', 'ABC-123', 'Honda', '2021', 'Civic', 'Red', '2021', 'car.png', NOW())



INSERT INTO `carpooling`.`cars`
(`UserID`,`Country`,`CarNumber`,`Make`,`Model`,`CarType`,`Color`,`RegistrationYear`,`PhotoName`,`CreatedDate`
)

VALUES (2, 'India', 'ABC-124', 'Suzuki', '2021', 'Cultus', 'Red', '2020', 'car2.png', NOW())

INSERT INTO `carpooling`.`cars`
(`UserID`,`Country`,`CarNumber`,`Make`,`Model`,`CarType`,`Color`,`RegistrationYear`,`PhotoName`,`CreatedDate`
)

VALUES (3, 'India', 'ABC-2123', 'Honda', '2022', 'Civic', 'Red', '2022', 'car3.png', NOW())
```

```
INSERT INTO `carpooling`.`cars`
(`UserID`,`Country`,`CarNumber`,`Make`,`Model`,`CarType`,`Color`,`RegistrationYear`,`PhotoName`,`CreatedDate`
)
VALUES (4, 'India', 'ABC-1233', 'Honda', '2021', 'Civic', 'White', '2021', 'car4.png', NOW())
```

Inserting into table ride_offers:

```
INSERT INTO
`carpooling`.`ride_offers`(`UserID`,`FromAddress`,`FromAddressLatitude`,`FromAddressLongitude`,`ToAddress`,`ToAddressLatitude`,`ToAddressLongitude`,`FromDateTime`,`IsRoundTrip`,`ToDateTime`,`Comments`,`PricePerPerson`,`NumberOfSeats`,`MaxLuggage`,`AgreeTermConditions`,`PublishRide`,`CreatedDate`,`IsDeleted`)
VALUES(1, 'Mumbai',19.076090,72.877426,'Delhi', '28.644800', '77.216721', '2022-11-25 20:53:17', 0, NULL, 'Ride offer',200, 3, '20KG', 1, 1, NOW(), 0)


INSERT INTO
`carpooling`.`ride_offers`(`UserID`,`FromAddress`,`FromAddressLatitude`,`FromAddressLongitude`,`ToAddress`,`ToAddressLatitude`,`ToAddressLongitude`,`FromDateTime`,`IsRoundTrip`,`ToDateTime`,`Comments`,`PricePerPerson`,`NumberOfSeats`,`MaxLuggage`,`AgreeTermConditions`,`PublishRide`,`CreatedDate`,`IsDeleted`)
VALUES(2, 'Mumbai',19.076090,72.877426,'Delhi', '28.644800', '77.216721', '2022-12-25 18:00:17', 0, NULL, 'Ride offer 2', 200, 3, '25KG', 1, 1, NOW(), 0)
```

```
INSERT INTO
`carpooling`.`ride_offers`(`UserID`,`FromAddress`,`FromAddressLatitude`,`FromAddressLongitude`,`ToAddress`,`T
oAddressLatitude`,`ToAddressLongitude`,`FromDateTime`,`IsRoundTrip`,`ToDateTime`,`Comments`,`PricePerPers
on`,`NumberOfSeats`,`MaxLuggage`,`AgreeTermConditions`,`PublishRide`,`CreatedDate`,`IsDeleted`)

VALUES(1, 'Mumbai',19.076090,72.877426,'Delhi', '28.644800', '77.216721', '2022-12-20 09:53:17', 0, NULL, 'Ride
offer3',200, 3, '40KG', 1, 1, NOW(), 0)



INSERT INTO
`carpooling`.`ride_offers`(`UserID`,`FromAddress`,`FromAddressLatitude`,`FromAddressLongitude`,`ToAddress`,`T
oAddressLatitude`,
`ToAddressLongitude`,`FromDateTime`,`IsRoundTrip`,`ToDateTime`,`Comments`,`PricePerPerson`,`NumberOfSeat
s`,`MaxLuggage`,`AgreeTermConditions`,`PublishRide`,`CreatedDate`,`IsDeleted`)VALUES(1,
Mumbai',19.076090,72.877426,'Delhi', '28.644800', '77.216721', '2022-11-30 11:00:17', 0, NULL, 'Ride offer',200,
3, '20KG', 1, 1, NOW(), 0)
```

Inserting into table ride_stop_overs:

```
INSERT INTO `carpooling`.`ride_stop_overs` (`RideOfferID`, `Address`, `PricePerPerson`, `Latitude`, `Longitude`,
`Distance`,`CreatedDate`,`IsDeleted`) VALUES(1,'Mumbai', 120, 19.076090,72.877426,80, NOW(), 0);

INSERT INTO `carpooling`.`ride_stop_overs` (`RideOfferID`, `Address`, `PricePerPerson`, `Latitude`, `Longitude`,
`Distance`,`CreatedDate`,`IsDeleted`) VALUES(2,'Mumbai', 120, 19.076090,72.877426,80, NOW(), 0);
```

```
INSERT INTO `carpooling`.`ride_stop_overs` (`RideOfferID`, `Address`, `PricePerPerson`, `Latitude`, `Longitude`,
`Distance`,`CreatedDate`,`IsDeleted`) VALUES(1,'Mumbai', 120, 19.076090,72.877426,80, NOW(), 0);


INSERT INTO `carpooling`.`ride_stop_overs` (`RideOfferID`, `Address`, `PricePerPerson`, `Latitude`, `Longitude`,
`Distance`,`CreatedDate`,`IsDeleted`) VALUES(2,'Mumbai', 120, 19.076090,72.877426,80, NOW(), 0);
```

Inserting into table ride_bookings:

```
INSERT INTO
`carpooling`.`ride_bookings`(`RideOfferID`,`BookingUserID`,`NoOfSeats`,`AcceptTermsAndConditions`,`CreatedDa
te`) VALUES(1, 1, 2, 1, NOW());
INSERT INTO
`carpooling`.`ride_bookings`(`RideOfferID`,`BookingUserID`,`NoOfSeats`,`AcceptTermsAndConditions`,`CreatedDa
te`) VALUES(1, 1, 2, 1, NOW());
INSERT INTO
`carpooling`.`ride_bookings`(`RideOfferID`,`BookingUserID`,`NoOfSeats`,`AcceptTermsAndConditions`,`CreatedDa
te`) VALUES(1, 1, 2, 1, NOW());
INSERT INTO
`carpooling`.`ride_bookings`(`RideOfferID`,`BookingUserID`,`NoOfSeats`,`AcceptTermsAndConditions`,`CreatedDa
te`) VALUES(1, 1, 2, 1, NOW());
```

Inserting into table ride_ratings:

```
INSERT INTO
`carpooling`.`ride_ratings`(`RatedToUserID`,`RatedByUserID`,`RatingReceived`,`RideOfferID`,`CreatedDate`,`Revie
wComments`) VALUES(1, 2, 5, 1, NOW(), "Nice ride");


INSERT INTO
`carpooling`.`ride_ratings`(`RatedToUserID`,`RatedByUserID`,`RatingReceived`,`RideOfferID`,`CreatedDate`,`Revie
wComments`) VALUES(1, 2, 5, 1, NOW(), "Nice ride again");

INSERT INTO
`carpooling`.`ride_ratings`(`RatedToUserID`,`RatedByUserID`,`RatingReceived`,`RideOfferID`,`CreatedDate`,`Revie
wComments`) VALUES(2, 1, 5, 2, NOW(), "Good driver");

INSERT INTO
`carpooling`.`ride_ratings`(`RatedToUserID`,`RatedByUserID`,`RatingReceived`,`RideOfferID`,`CreatedDate`,`Revie
wComments`) VALUES(1, 2, 5, 1, NOW(), "Nice ride");
```

Inserting into table ride_massages:

```
INSERT INTO `carpooling`.`rides_messages`

(`FromUserID`,`ToUserID`,`RideOfferID`,`MessageBody`,`IsRead`,`CreatedDate`)

VALUES(1, 2 ,1, "Hi, When you will arrive?", 0, NOW());
```

```
INSERT INTO `carpooling`.`rides_messages`
(`FromUserID`,`ToUserID`,`RideOfferID`,`MessageBody`,`IsRead`,`CreatedDate`)
VALUES(2, 1 ,1, "Hi, at 12pm?", 0, NOW());

INSERT INTO `carpooling`.`rides_messages`
(`FromUserID`,`ToUserID`,`RideOfferID`,`MessageBody`,`IsRead`,`CreatedDate`)
VALUES(1, 2 ,3, "Hi, When you will go?", 0, NOW());

INSERT INTO `carpooling`.`rides_messages`
(`FromUserID`,`ToUserID`,`RideOfferID`,`MessageBody`,`IsRead`,`CreatedDate`)
VALUES(2, 1 ,3, "Hi, at 9am?", 0, NOW());
```

Inserting into table ride_alerts:

```
INSERT INTO `carpooling`.`ride_alerts`
(`UserID`,`FromAddress`,`FromAddressLatitude`,`FromAddressLongitude`,`ToAddress`,`ToAddressLatitude`,`ToAd
dressLongitude`,`RideDateTime`,`CreatedDate`,`WeeklyAlert`,`IsDeleted`) VALUES(1, 'Mumbai',19.0761 ,72.8774,
'Delhi', 28.6448, 77.2167, '2022-12-25 20:53:17',NOW(), 1, 0);
```

```
INSERT INTO `carpooling`.`ride_alerts`
(`UserID`,`FromAddress`,`FromAddressLatitude`,`FromAddressLongitude`,`ToAddress`,`ToAddressLatitude`,`ToAddressLongitude`,`RideDateTime`,`CreatedDate`,`WeeklyAlert`,`IsDeleted`) VALUES(1, 'Mumbai',19.0761 ,72.8774, 'Delhi', 28.6448, 77.2167, '2022-12-25 20:53:17',NOW(), 1, 0);

INSERT INTO `carpooling`.`ride_alerts`
(`UserID`,`FromAddress`,`FromAddressLatitude`,`FromAddressLongitude`,`ToAddress`,`ToAddressLatitude`,`ToAddressLongitude`,`RideDateTime`,`CreatedDate`,`WeeklyAlert`,`IsDeleted`) VALUES(1, 'Mumbai',19.0761 ,72.8774, 'Delhi', 28.6448, 77.2167, '2022-12-25 20:53:17',NOW(), 1, 0);

INSERT INTO `carpooling`.`ride_alerts`
(`UserID`,`FromAddress`,`FromAddressLatitude`,`FromAddressLongitude`,`ToAddress`,`ToAddressLatitude`,`ToAddressLongitude`,`RideDateTime`,`CreatedDate`,`WeeklyAlert`,`IsDeleted`) VALUES(1, 'Mumbai',19.0761 ,72.8774, 'Delhi', 28.6448, 77.2167, '2022-12-25 20:53:17',NOW(), 1, 0);
```

6. Explanation and SQL Code for retrieving information from the database:

SELECT Query:

To retrieve information from database tables we use SELECT statement. SELECT query syntax is:

```
SELECT ColumName1, ColumName2, ….

FROM TableName

WHERE Condition
```

Suppose if we want to retrieve offers from ride_offers table for 'Mumbai'city then SQL query will be like this:

```
SELECT * FROM ride_offers WHERE FromAddress = 'Mumbai'
```

*will retrieve all columns.

If you want to retrieve all records of the table then remove WHERE clause.

e.g:

```
SELECT * FROM ride_offers
```

SELECT Query with JOINS:

If you want to retrieve records from more than one tables then we use joins. Both tables must be joined with foreign key.

If we want to get ratings of the user then we can write the query as below:

```
SELECT `ride_ratings`.`RatedToUserID` AS `RatedToUserID`,round(avg(cast(`ride_ratings`.`RatingReceived` as float)),1) AS `UserRating`,`appusers`.`FirstName` AS `FirstName`,`appusers`.`LastName` AS `LastName`,`appusers`.`Email` AS `Email`

FROM (`ride_ratings` join `appusers` on((`ride_ratings`.`RatedToUserID` = `appusers`.`UserID`)))
```

```
GROUP BY `ride_ratings`.`RatedToUserID`,`appusers`.`FirstName`,`appusers`.`LastName`,`appusers`.`Email`
```

Functions:

MySQL has many build-in functions. Some are String related functions (e.g: TRIM, CONCAT etc), There are also Date and Time (e.g: NOW, DAY, HOUR etc) functions, Math functions (e.g: PI, LOG, COS, ACOS, CEIL, FLOOR etc).

We can add our own user defined function which will return string, integer, float etc value.

Below is the FUNCTION which returns rating of the user.

```
USE `carpooling`;

DROP function IF EXISTS `GetUserRating`;

DELIMITER $$

USE `carpooling`$$

CREATE FUNCTION `GetUserRating` (user_Id int)

RETURNS INTEGER

BEGIN

DECLARE user_rating int DEFAULT 0;

SELECT RatingReceived INTO user_rating FROM ride_ratings WHERE RatedToUserID = user_Id LIMIT 1;

RETURN user_rating;

END$$
```

```
DELIMITER ;
```

7. Explanation and SQL Code for Triggers

As explained in Section 7 A table and a database object called a trigger go together.When a certain action is carried out on the table, it becomes active. When you runone of the table's INSERT, UPDATE, or DELETE statements, the triggers are activated. Either before or after the event, it may be revoked.

Below is the trigger which insert record in ride_bookings_history automatically when any record is inserted in ride_bookings table.

```
CREATE TRIGGER bookingHistory
    AFTER INSERT ON ride_bookings
    FOR EACH ROW
 INSERT INTO ride_bookings_history(
 RideBookingID,
  RideOfferID,
  BookingUserID,
  NoOfSeats,
  AcceptTermsAndConditions,
CreatedDate)

VALUES

(

NEW.RideBookingID,

NEW.RideOfferID,

NEW.BookingUserID,

NEW.NoOfSeats,

NEW.AcceptTermsAndConditions,

NEW.CreatedDate

)
```

8. Explanation and SQL Code for Security

To secure the database we create a specific user in database and allow that user some permissions to insert, update, select or alter the database. We can give some specific permissions or all permissions also.

To create user in MySQL, use CREATE USER syntax:

```
CREATE USER 'carpooling_user'@'localhost' IDENTIFIED BY 'carpooling123';
```

Then to grant all access to the database (e.g. carpooling), use GRANT Syntax, e.g.

```
GRANT ALL ON carpooling.* TO 'carpooling_user'@'localhost';
```

Where ALL can be replaced with specific privilege such as SELECT, INSERT, UPDATE, ALTER, etc.

Then to reload newly assigned permissions run:

```
FLUSH PRIVILEGES;
```

9. Explanation of Additional SQL Features:

SQL has stored procedures feature also. In stored procedure we can apply sql queries, if else statements, loops, switch statements etc.

For example: Below is the stored procedure which get all ratings from 1 to 5 for the specific user.

```
USE `carpooling`;

DROP procedure IF EXISTS `UserRatingsGivenTo`;

DELIMITER $$

USE `carpooling`$$

CREATE PROCEDURE `UserRatingsGivenTo` (RatedByUserID int)

BEGIN

declare rate5 float;
declare rate4 float;

declare rate3 float;

declare rate2 float;

declare rate1 float;


declare totalRateCount float;


select rate5 = COUNT(RatingReceived) from ride_Ratings WHERE RatingReceived = 5 and RatedByUserId = RatedByUserID;

select rate4 = COUNT(RatingReceived) from ride_Ratings WHERE RatingReceived = 4 and RatedByUserId = RatedByUserID;

select rate3 = COUNT(RatingReceived) from ride_Ratings WHERE RatingReceived = 3 and RatedByUserId = RatedByUserID;

select rate2 = COUNT(RatingReceived) from ride_Ratings WHERE RatingReceived = 2 and RatedByUserId = RatedByUserID;

select rate1 = COUNT(RatingReceived) from ride_Ratings WHERE RatingReceived = 1 and RatedByUserId = RatedByUserID;

Select totalRateCount = COUNT(RatingReceived) from ride_Ratings WHERE RatedByUserId = RatedByUserID;


IF(totalRateCount = 0) then

        select cast(0 as float) AS RatingFive, cast(0 as float) AS RatingFour, cast(0 as float) AS RatingThree,

        cast(0 as float) AS RatingTwo, cast(0 as float) AS RatingOne;

ELSE

        select ROUND(rate5/totalRateCount * 100, 0) AS RatingFive, ROUND(rate4/totalRateCount * 100, 0) AS RatingFour,
ROUND(rate3/totalRateCount * 100, 0) AS RatingThree,

        ROUND(rate2/totalRateCount * 100, 0) AS RatingTwo, ROUND(rate1/totalRateCount * 100, 0) AS RatingOne;
```

```
   end if;

   end$$

DELIMITER ;
```