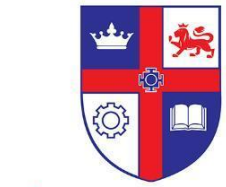
LONDON METROPOLITAN UNIVERSITY



islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

50% Individual Coursework

2022 Autumn

Student Name: Anurag Anand

London Met ID: 21039635

Assignment Submission Date: Thursday, January 5, 2023

Word Count: 5700

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

Table of Figures

Table of Figures

Part 1: Introduction

1.1: Grab Company

Grab is a taxi-booking mobile app intended for use in Southeast Asia was launched in 2012 in Malaysia, but e headquarters in Singapore now (Grab, 2022). After establishment, their main goal was to make taxi rides safer as well as to compete in the market dominated by Uber. It also expanded to the Philippines, Singapore, Thailand, and Indonesia within a couple of years. It has taken the safety of its riders rigorously as compared to other taxi services and this is the grandest strength of the business. Now, it has expanded its services to demands of food delivery, courier delivery, and e-commerce (products delivery). They also have vision of operating their business into South Asian countries like Nepal, India, China, Pakistan, and Bangladesh. Companies like inDriver, Pathao and other commonly used mobile-app based delivery and ride sharing companies are in-affective here. They are actively looking to design the very best way to easily provide these much-needed services to their end-users, hence, they have decided to design a database system capable of handling all four of their services effectively in the South Asian market.

1.2: Aim of Grab

The aims of Grab are to make its services accessible for all, empower millions of Micro Businesses, and provide safety and trust to all its consumers using its main services like booking rides, packages delivery, food delivery, and ecommerce products delivery and expand to more densely populated parts of Asia.

1.3: Objectives

- Providing safe and well trusted ride booking services.
- Designing products and services that are easily affordable for people of all levels of income.
- Allowing one hundred million micro businesses an opportunity to create income and provide its services to the consumers.
- Allow delivery of packages to/from a consumer to another.
- Fast delivery of foods from restaurants and other products from shops.

- Create job opportunities – drivers, app-developers, IT professionals, merchant related jobs, admin jobs, etc.
- Becoming eco-friendly by using greener less carbon-emitting vehicles in their delivering their services.

1.4: Current Business Activities and Operations

Right now, this business has expanded its activities quite broadly, as compared to 2012 when Grab came to existence, through its executives making good decisions by researching the market demands well.

Back in 2012, only the ride-hailing service was available which became possible by creating an efficient app that allowed people to easily navigate and book cabs.

In 2021, it was shown there are over six million rides completed through the Grab app (Code Brew, 2021) and about 5 million drivers employed for all the delivery operations for its business throughout the countries where it's available. (Chen Lin, 2022)

It has about 2.8 million merchants across the nations where Grab is available to use. (Code Brew, 2021).

Because of its services having a massive demand, their First Quarter 2022 Results showed an estimated $1.3 billion of revenue in the year 2022. (Grab, 2022)

The business activities that are now done through those merchants are: food delivery, products delivery, courier delivery, and the most importantly– rides services.

1.5: The current business rules are:

- A driver may drive many vehicles, but each vehicle is used by only one driver at a time.
- A driver writes a single invoice for each service he provides which is related to the service itself.
- One or more drivers can do many services at once like they can combine doing food and product delivery at the same time using same vehicle.
- One or more customers can have many services used at the same time. But they can only book one ride service at a time.
- Once the customer books the service, they cannot cancel the service.
- Only drivers are able to cancel the service when needed.
- Service ticket is issued once the customer books the vehicle and the service and will include details like driver name, type of service, total charge, estimated duration of the destination and more information.
- The cost of the vehicle and duration can vary depending on its type, eg: cost of riding motorbikes is cheaper than riding cars/vans.
- Customers can order food from different restaurants, products from stores, send courier, or take ride from their present location to their specified destination.
- Customers can choose the option to pay in cash, card, or through Grab apps.
- A package will be returned to the sender if destination cannot be reached for some reason, and this will be a new service the user makes and need to pay for the total charge for the return delivery.
- Multiple packages can be delivered to the same or different destination at the same time.
- The total charge only calculates the delivery of the services and amount need to be paid for the ride service, the actual product or food costs are handled through another system in the app.

Part 2: Initial ERD

On the basis of the business rules of my company I have made this initial ERD diagram. I have considered that a vehicle is used for multiple services and multiple services can be used by multiple customers at a time. This diagram shows that relation as well as the relation between services and drivers, invoice generation, and ticket generation which are all kept within the service entity in my case.

2.a: Identification of Entities and Attributes

The entities that I've visioned forming for the database of Grab in its initial form are Vehicle, Service, and Customer. These entities are based on simple reasoning, there will be further breakdown of them into more when we will move into forming the final ERD.

2.a.1: Vehicle Table:

As shown in the table below of Vehicle Entity, all those attributes contain relevant information about the vehicle used for the services provided by Grab company that the customers use on a day-to-day basis. The Vehicle attributes are shown with their data types, constraints, and brief descriptions about what values they hold.

Attributes	Data Type	Constraints	Description
vehicleID	VARCHAR2 (4)	PRIMARY KEY, UNIQUE	This is the unique ID to distinguish the vehicles used for the services.
vehicleType	VARCHAR2 (20)	NOT NULL	This is the type of vehicle like a bike, car, etc.
vehicleCost	NUMBER (10)	NOT NULL	This is the price of the vehicle when purchased.
fuelType	VARCHAR2 (10)	NOT NULL	Indicated what type of fuel is used
vehicleModel	VARCHAR2 (30)	NOT NULL	Attribute that stores model of the vehicle.

vehicleVariant	VARCHAR2 (20)	NOT NULL	Indicated the variant of the model of the vehicle.
vehicleBaseCost	NUMBER (5)	NOT NULL	This is the base price of a vehicle when they are used in a service.

Table 1: Attributes of Vehicle Table - Initial ERD

2.a.2: Service Table

In the table below for Service Entity, all those attributes listed contain relevant information about the services provided by Grab company that the customers use on a day-to-day basis as well as has details concerning the drivers that delivered the service, the ticket information for when the customer places orders, and the relevant invoice that customer gets once the service is completed. In this table, the service attributes are shown with their data types, constraints, and brief descriptions about what values they hold.

Attributes	Data Type	Constraints	Description
serviceID	VARCHAR2(5)	PRIMARY KEY, UNIQUE	This is the unique ID to distinguish the service used by a customer.
vehicleID	VARCHAR2(5)	FOREIGN KEY, NOT NULL	This is the unique ID to distinguish the vehicles used for the services.
serviceBaseCharge	NUMBER (2)	NOT NULL	The base price for a service used.
chargePerKm	NUMBER (3)	NOT NULL	The total cost per km for the specific service used.
serviceCategory	VARCHAR2 (40)	NOT NULL	The names of the services that are available to use.
ticketNo	VARCHAR2 (4)	NOT NULL	This is the unique ID to distinguish the tickets

			generated for each services customer uses.
issuedDate	DATE	NOT NULL	The date when the service has begun.
issuedTime	VARCHAR2 (10)	NOT NULL	The time when the service has begun.
totalDistance	NUMBER (3,1)	NOT NULL	The total distance from initial location to destination.
destination	VARCHAR2 (20)	NOT NULL	The location where the service comes to an end.
initialLocation	VARCHAR2(20)	NOT NULL	The location where the service is started from.
serviceDuration	NUMBER (3)	NOT NULL	Total amount of time for the service to complete once begun.
totalCharge	NUMBER (4)	NOT NULL	The total amount for the service which takes account of base vehicle price, service charge, discounts, etc.
rewardPoints	NUMBER (5)	NOT NULL	The amount of points customer gets for using a service.
transactionID	VARCHAR2 (4)	NOT NULL	This is the unique ID to distinguish invoices generated for the services.
paymentMethod	VARCHAR2 (50)	NOT NULL	Details of payment method and amount paid.
driverID	VARCHAR2 (4)	NOT NULL	This is the unique ID to distinguish the driver for the services.
driverName	VARCHAR2 (40)	NOT NULL	Full name of the driver.

driverAddress	VARCHAR2 (60)	NOT NULL	The home address of the driver.
driverEmail	VARCHAR2 (60)	NOT NULL	The email address of the driver.
driverContactNo	VARCHAR2 (15)	NOT NULL	The phone number of the driver.
driverSalary	NUMBER (8)	NOT NULL	The annual salary of the driver.
driverDOB	DATE	NOT NULL	The date of birth of the driver.
driverGender	VARCHAR2 (10)	NOT NULL	The gender of the driver.
driverCategory	VARCHAR2 (20)	NOT NULL	Used to show if a driver works full-time or part-time.

Table 2:Attributes of Service Table - Initial ERD

2.a.3: Customer Table

The table below is for the Customer Entity, all those attributes listed there contain relevant information about the vehicle used for the services provided by Grab company that the customers use on a day-to-day basis. These Customer attributes are shown with their data types, constraints, and brief descriptions about what values they hold.

Attributes	Data Type	Constraints	Description
customerID	VARCHAR2 (4)	PRIMARY KEY, UNIQUE	This is the unique ID to distinguish the customers that use the services.
vehicleID	VARCHAR2 (4)	FOREIGN KEY, NOT NULL	This is the unique ID to distinguish the vehicles used for the services.
serviceID	VARCHAR2 (4)	FOREIGN KEY, NOT NULL	This is the unique ID to distinguish the services used by a customer.
customerName	VARCHAR2 (40)	NOT NULL	The full name of the customer.
customerAddress	VARCHAR2 (60)	NOT NULL	The home address of the customer.
customerPhone	VARCHAR2 (15)	NOT NULL	The phone number of the customer.

customerEmail	VARCHAR2 (60)	NOT NULL	The email address of the customer.
customerCategory	VARCHAR2 (15)	NOT NULL	The type of customer.
customerDiscount	VARCHAR2 (5)	NOT NULL	The discount percent the customer gets.
customerDOB	DATE	NOT NULL	The birthdate of the customer
customerGender	VARCHAR2 (10)	NOT NULL	The gender of the customer.

Table 3: Attributes of Customer Table - Initial ERD

2.b: Identification and representation of the Primary Keys and Foreign Keys.

We identify the primary keys by checking if those attributes are unique and can define a row of data uniquely. Whereas foreign keys are columns that create relationships between two tables. These keys can be used together to remove redundancies and create easier processing of data in the database management system. (geeksforgeeks, 2022)

In our initial ERD diagram, we can see that customerID, vehicleID, and serviceID act as primary key in Customer table, Vehicle table, and Service table respectively. The vehicleID, and serviceID are foreign keys in Customer table. The vehicleID is the foreign key in Vehicle table. And the vehicle table doesn't have any foreign keys.

2.c: Initial Entity Relationship Diagram (ERD) identified with its attributes and relationships.

There is a one-to-many relationship between a vehicle and service tables. This means one vehicle is mandatory for a service and carries out multiple services. Also, there is a many-to-many relationship between service and customer. This means one or many customers can use one or more services at a time.

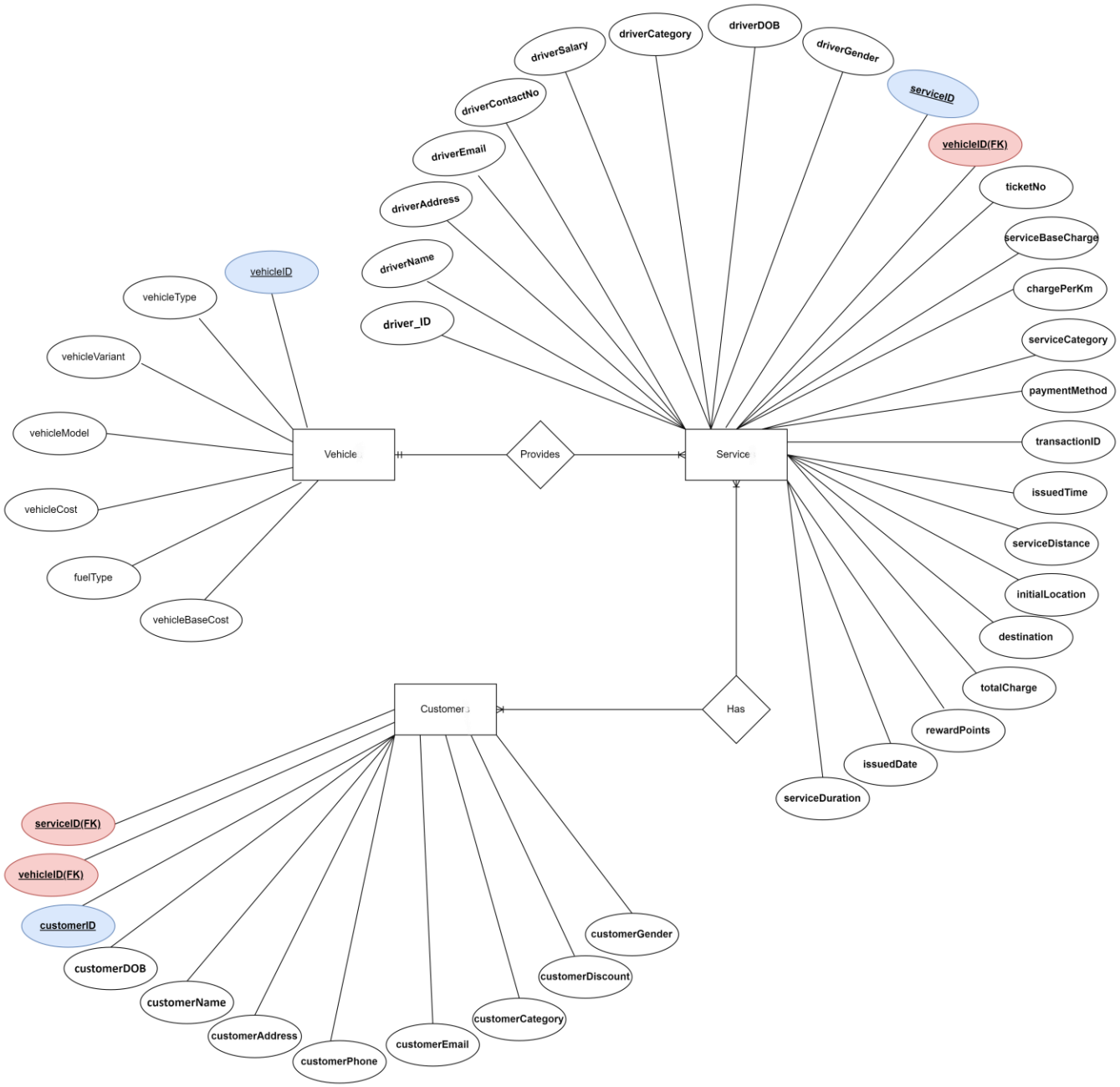


Figure 1: Initial ERD for Grab company. Blue represents 'Primary Key' and Red represents 'Foreign key'

2.d: Assumptions

- One customer uses one or more service at a time.
- Drivers only drive one vehicle at a time, but they can drive many vehicles on different days/months/years.
- One driver gives one or more services at a time.
- One transaction is made for one specific service.
- One ticket is generated for a specific service requested by the customer.
- Use of different services also generates respective rewards points for the customer which will be linked to service ticket.
- The totalCharge for different services includes calculation using baseCost of vehicle used and totalDistance, chargePerKm, customerDiscount and serviceBaseCharge.
- Even if the customer/driver home addresses are in different countries, we assume they are present in Kathmandu using the service.
- Invoice shows the user all the details regarding the payments they have made.
- Ticket has details concerning which service has been booked by the customer mainly.
- We have assumed that there isn’t a direct relation between the driver and customer, their relation is through the service usage.

Part 3: Normalization

The technique of effectively organizing a database's data is called normalization. The normalization process has two objectives: removing redundant data (such as keeping the same data in multiple tables) and ensuring that dependencies make sense (just storing interrelated data in a specific table). Both of these objectives are worthwhile ones because they help a database use less space and make sure the data is stored logically. These in turn help avoid any kind of anomalies related to inserting, updating, and deleting data. (Kormos, 2022)

Note: Underlined attribute signifies Primary Key. Underline and (FK) together signify Foreign Keys.

3.a: Unnormalized form (UNF)

In this form, we have a basic database data model (how data is organized in a database) that fails to adhere to any of the relational model's requirements for database normalization. (DBpedia, 2023)

The UNF form for our attributes will be written in the following way based on the initial ERD we had.

UNF: vehicle (vehicleID, vehicleType, vehicleCost, fuelType, vehicleModel, vehicleVariant, vehicleBaseCost, **{**vehicleID, serviceBaseCharge, chargePerKm, serviceCategory, ticketNo**,** issuedDate, issuedTime, totalDistance, destination, initialLocation, serviceDuration, totalCharge, rewardPoints, transactionID, paymentMethod, driverID, driverName, driverAddress, driverEmail, driverContactNo, driverSalary, driverDOB, driverGender, driverCategory, {customerID, customerName, customerAddress, customerPhone, customerEmail, customerCategory, customerDiscount, customerDOB, customerGender}})

Here, we have considered the vehicle to be the unique data which is repeating. The service and customer attributes are placed in the repeating groups. The customer repeating group is linked to both the vehicle data and service attributes.

3.b: First Normal form (1NF)

The first normal form (1NF) specifies the ground conditions for a structured database (Kormos, 2022):

Remove away redundant columns in the same table.

Make unique tables (connect repeating groups table with repeating data table as needed) for each group of connected data and give each row its own column or collection of columns (using introduced primary key).

From our UNF table vehicle, I have derived three tables by separating the repeating data into a table (vehicle - 1), and the two repeating groups into two different tables (service - 1 and customer - 1). Also shown relation among them using the foreign key attributes specification.

1NF tables:

vehicle -1 (vehicleID, vehicleType, fuelType, vehicleModel, vehicleVariant, vehicleBaseCost, vehicleCost)

service -1 (serviceID, vehicleID(FK), serviceBaseCharge, chargePerKm, serviceCategory, ticketNo**,** issuedDate, issuedTime, totalDistance, destination, initialLocation, serviceDuration, totalCharge, rewardPoints, transactionID, paymentMethod, driverID, driverName, driverAddress, driverEmail, driverContactNo, driverSalary, driverDOB, driverGender, driverCategory)

customer – 1 (customerID, vehicleID(FK), serviceID(FK), customerName, customerAddress, customerPhone, customerEmail, customerCategory, customerDiscount, customerDOB, customerGender)

3.c: Second Normal form (2NF)

The table must first be in 1NF in order for it to be normalized to 2NF. There should not be partial dependencies in the table. The proper subset of the candidate key should produce a non-prime attribute because of the partial reliance in this situation.

2NF process:

vehicle -2 (vehicleID, vehicleType, vehicleType, vehicleModel, vehicleVariant, vehicleBaseCost, vehicleCost)

Checking for partial dependencies in Service – 1:

serviceID → serviceBaseCharge, chargePerKm, serviceCategory, ticketNo**,** issuedDate, issuedTime, totalDistance, destination, initialLocation, serviceDuration, totalCharge, rewardPoints, transactionID, paymentMethod

serviceID alone gives all those attributes shown above which indicates partial dependency so this kept into a table called ‘service - 2’.

vehicleID(FK) → X ()

serviceID, vehicleID(FK) → driverID, driverName, driverAddress, driverEmail, driverContactNo, driverSalary, driverDOB, driverGender, driverCategory

Here, the above attributes are determined by the composite key. Hence, a new entity called ‘driver – 2’ is introduced.

service -2 (serviceID, serviceBaseCharge, chargePerKm, serviceCategory, ticketNo**,** issuedDate, issuedTime, totalDistance, destination, initialLocation, serviceDuration, totalCharge, rewardPoints, transactionID, paymentMethod)

driver – 2 (serviceID, vehicleID(FK), driverID, driverName, driverAddress, driverEmail, driverContactNo, driverSalary, driverDOB, driverGender, driverCategory)

Checking for Partial dependencies in Customer – 1:

customerID → customerName, customerAddress, customerPhone, customerEmail, customerCategory, customerDiscount, customerDOB, customerGender

customerID, vehicleID* → X

customerID, serviceID(FK) → X (This relation must be present)

vehicleID(FK), serviceID(FK) → X

customerID, vehicleID(FK), serviceID(FK) → X

We get customer – 2 table and CustomerOrder – 2 table from the 2NF process of customer – 1 table. Here, we got customer - 2 table by removing partial dependency on the composite key. And we got customerOrder – 2 which acts as a bridge entity between customer – 2 and service – 2 tables.

We ignore other bridge entities as they are not needed according to our assumptions for the database model.

customer – 2 (customerID, customerName, customerAddress, customerPhone, customerEmail, customerCategory, customerDiscount, customerDOB, customerGender)

customerOrder -2 (customerID(FK), serviceID(FK))

After 2NF process we get these tables:

vehicle -2 (vehicleID, vehicleType, fuelType, vehicleModel, vehicleVariant, vehicleBaseCost, vehicleCost)

service -2 (serviceID, serviceBaseCharge, chargePerKm, serviceCategory, ticketNo**,** issuedDate, issuedTime, totalDistance, destination, initialLocation, serviceDuration, totalCharge, rewardPoints, transactionID, paymentMethod)

driver – 2 (serviceID, vehicleID(FK), driverID, driverName, driverAddress, driverEmail, driverContactNo, driverSalary, driverDOB, driverGender, driverCategory)

customer – 2 (customerID, customerName, customerAddress, customerPhone, customerEmail, customerCategory, customerDiscount, customerDOB, customerGender)

customerOrder -2 (customerID(FK), serviceID(FK))

3.d: Third Normal form (3NF)

Third normal form is the next process we can undertake on tables that are in 2NF form, here we separate non-key attributes from the table to a new table if they are depended on a non-key value which is dependent on a key value. That means checking for transitive dependencies and creating new tables for that kind of dependency.

3NF:

We will check for transitive dependencies in tables that consist of one or more keys,

In vehicle – 2 table:

We will look for the transitive dependencies below

vehicleID → gives all other non-key attributes and not even one of those non-key attributes are depended on each other.

So, a new table isn't formed. vehicle - 3 is same as vehicle - 2

vehicle -3 (vehicleID, vehicleType, vehicleType, vehicleModel, vehicleVariant, vehicleBaseCost, vehicleCost)

Now, we will do the same in Service – 2 table:

serviceID → ticketNo (non-key) → issuedDate, issuedTime, totalDistance, destination, initialLocation, serviceDuration, totalCharge, rewardPoints (non-key values)

serviceID → transactionID(non-key) → paymentMethod (non-key)

ticketNo and transactionID are two non-keys that have other non-key values dependent on. So, we must separate these out into two new tables.

service – 3 (serviceID, ticketNo(FK), transactionID(FK), serviceBaseCharge, chargePerKm, serviceCategory)

ticket – 3 (ticketNo, issuedDate, issuedTime, totalDistance, destination, initialLocation, serviceDuration, totalCharge, rewardPoints)

invoice – 3(transationID, paymentMethod)

Repeating the same checking process in Driver – 2 Table:

serviceID, vehicleID(FK) → driverID(non-key) → driverName, driverAddress, driverEmail, driverContactNo, driverSalary, driverDOB, driverGender, driverCategory

The composite key in Driver – 2 table, gives driverID which is a non-key that has other non-key values dependent on it. So, we must separate these out into different table.

driver – 3 (driverID, driverName, driverAddress, driverEmail, driverContactNo, driverSalary, driverDOB, driverGender, driverCategory)

serviceDeliverer – 3 (serviceID(FK), vehicleID(FK), driverID(FK))

The Customer – 2 and CustomerOrder – 2 tables do not have transitive dependencies in them. So they remain unchanged from 2NF process:

Customer – 3 (customerID, customerName, customerAddress, customerPhone, customerEmail, customerCategory, customerDiscount, customerDOB, customerGender)

customerOrder -3 (customerID(FK), serviceID(FK))

Final tables from Normalization processes undergone above (UNF to 3NF) are below:

vehicle -3 (vehicleID, vehicleType, fuelType, vehicleModel, vehicleVariant, vehicleBaseCost, vehicleCost)

service – 3 (serviceID, ticketNo(FK), transactionID(FK), serviceBaseCharge, chargePerKm, serviceCategory)

ticket – 3 (ticketNo, issuedDate, issuedTime, totalDistance, destination, initialLocation, serviceDuration, totalCharge, rewardPoints)

invoice – 3(transationID, paymentMethod)

driver – 3 (driverID, driverName, driverAddress, driverEmail, driverContactNo, driverSalary, driverDOB, driverGender, driverCategory)

serviceDeliverer – 3 (serviceID(FK), vehicleID(FK), driverID(FK))

customer – 3 (customerID, customerName, customerAddress, customerPhone, customerEmail, customerCategory, customerDiscount, customerDOB, customerGender)

customerOrder -3 (customerID(FK), serviceID(FK))

Part 4: Final Entity Relationship Diagram (ERD)

This is the final ERD achieved after the normalization of initial ERD based on all the assumptions listed after the initial ERD was made in Part 2 of this coursework.

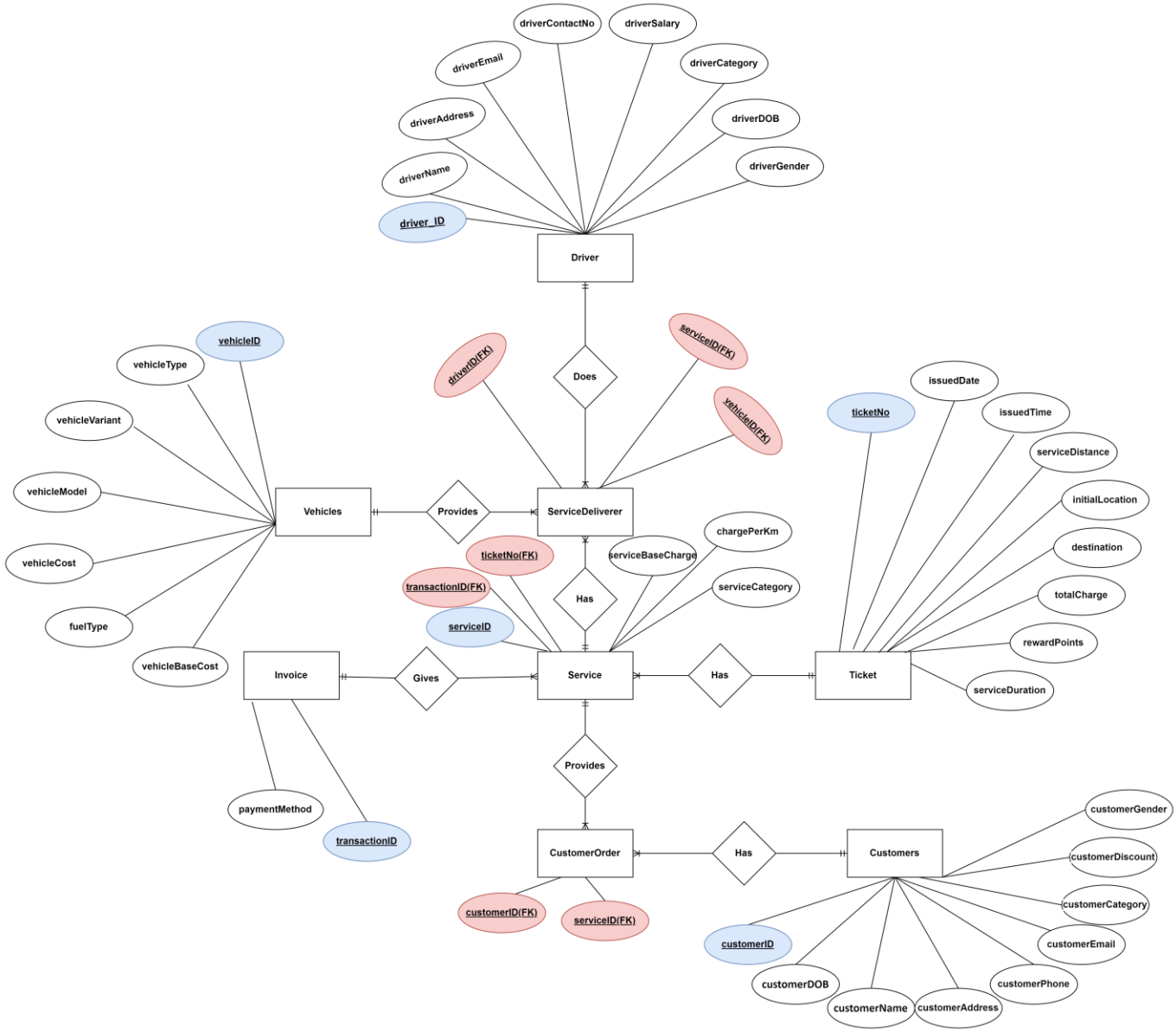


Figure 2: The final ERD of Grab Company database

Relations between entities:

There is One-to-Many relation between customers and customerOrder entities. One-to-Many relation between service and CustomerOrder entities. Many-to-One relation between service and ticket as well as service and invoice entities. One-to-Many relation between service and serviceDeliverer entities. Many-to-One relation between serviceDeliverer and vehicle as well as service and driver entities.

Part 5: Implementation

In order to implement our database schema, we first need to create a new user then make our data base. The screenshot below shows how my user was made:

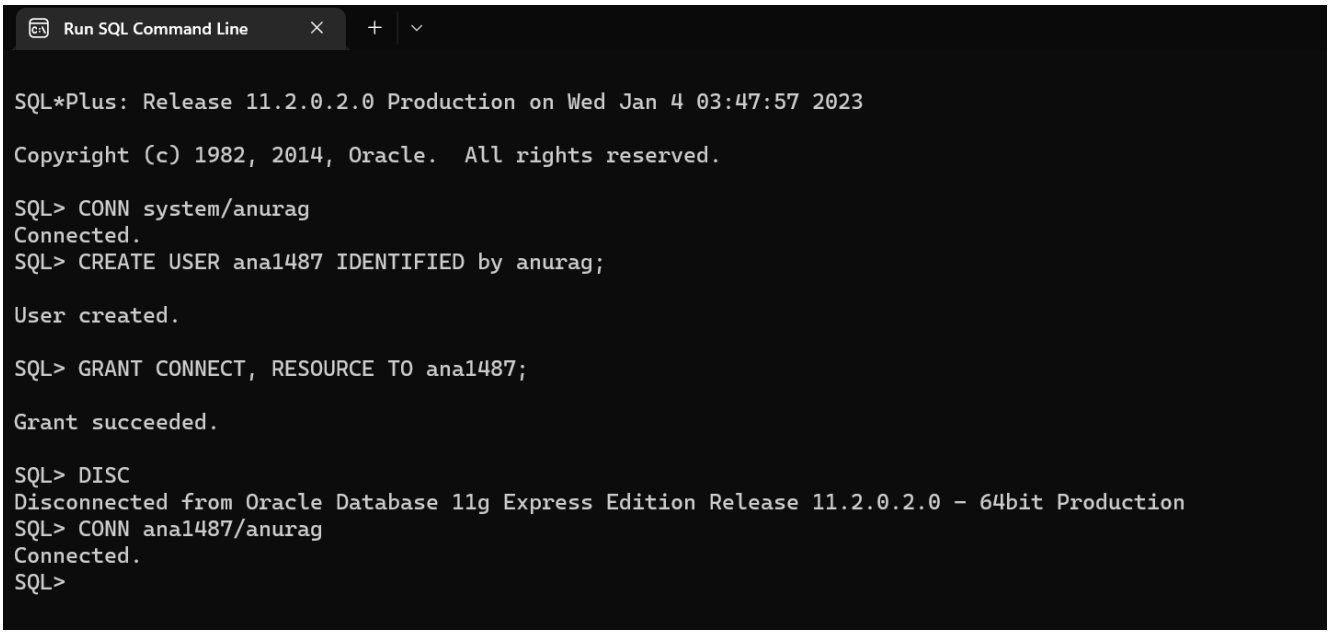


Figure 3: Creating user ana1487 to write the SQL commands

In the pages below of the subsections of Part 5(a), we have shown the tables of all entities formed after the normalization process. They will be used accordingly to make their respective tables on Oracle SQL database.

5.a Create Relations and Tables for the database with SQL Command

5.a.1: customer Table: Creation of table based on these attributes and its structure shown

Attributes	Data Type	Constraints	Description
customerID	VARCHAR2 (4)	PRIMARY KEY, UNIQUE	This is the unique ID to distinguish the customers that use the services.
customerName	VARCHAR2 (40)	NOT NULL	The full name of the customer.
customerAddress	VARCHAR2 (60)	NOT NULL	The home address of the customer.
customerPhone	VARCHAR2 (15)	NOT NULL	The phone number of the customer.
customerEmail	VARCHAR2 (60)	NOT NULL	The email address of the customer.
customerCategory	VARCHAR2 (15)	NOT NULL	The type of customer.
customerDiscount	VARCHAR2 (5)	NOT NULL	The discount percent the customer gets.
customerDOB	DATE	NOT NULL	The birthdate of the customer
customerGender	VARCHAR2 (10)	NOT NULL	The gender of the customer.

Table 4: The table customer with all of its attributes

Creating customer table:

```
SQL> CREATE TABLE customer(customerID VARCHAR2(4) PRIMARY KEY, customerName VARCHAR2(40) NOT NULL,
  2  customerAddress VARCHAR2(60) NOT NULL, customerPhone VARCHAR2(15) NOT NULL,
  3  customerEmail VARCHAR2(60) NOT NULL, customerCategory VARCHAR2(15) NOT NULL,
  4  customerDiscount VARCHAR2(5) NOT NULL, customerDOB DATE NOT NULL,
  5  customerGender VARCHAR2(10));

Table created.
```

Figure 4: Creating customer table

Seeing structure of customer table:

```
SQL> DESC customer;
 Name              Null?    Type
 ----------------- -------- ------------
 CUSTOMERID        NOT NULL VARCHAR2(4)
 CUSTOMERNAME      NOT NULL VARCHAR2(40)
 CUSTOMERADDRESS   NOT NULL VARCHAR2(60)
 CUSTOMERPHONE     NOT NULL VARCHAR2(15)
 CUSTOMEREMAIL     NOT NULL VARCHAR2(60)
 CUSTOMERCATEGORY  NOT NULL VARCHAR2(15)
 CUSTOMERDISCOUNT  NOT NULL VARCHAR2(5)
 CUSTOMERDOB       NOT NULL DATE
 CUSTOMERGENDER             VARCHAR2(10)

SQL>
```

Figure 5: Seeing structure of customer table

5.a.2: driver Table: Creation of table based on these attributes and its structure is shown

Attributes	Data Type	Constraints	Description
driverID	VARCHAR2 (4)	PRIMARY KEY, UNIQUE	This is the unique ID to distinguish the driver for the services.
driverName	VARCHAR2 (40)	NOT NULL	Full name of the driver.
driverAddress	VARCHAR2 (60)	NOT NULL	The home address of the driver.
driverEmail	VARCHAR2 (60)	NOT NULL	The email address of the driver.
driverContactNo	VARCHAR2 (15)	NOT NULL	The phone number of the driver.
driverSalary	NUMBER (8)	NOT NULL	The annual salary of the driver.
driverDOB	DATE	NOT NULL	The date of birth of the driver.
driverGender	VARCHAR2 (10)	NOT NULL	The gender of the driver.
driverCategory	VARCHAR2 (20)	NOT NULL	Used to show if a driver works full-time or part-time.

Table 5:The table driver with all of its attributes

Creating table driver and seeing its structure.

```
SQL> CREATE TABLE driver(driverID VARCHAR2(4) PRIMARY KEY, driverName VARCHAR2(40) NOT NULL,
  2  driverAddress VARCHAR2(60) NOT NULL, diverEmail VARCHAR2(60) NOT NULL,
  3  driverContactNo VARCHAR2(15) NOT NULL, driverDOB DATE NOT NULL, driverGender VARCHAR2(10) NOT NULL,
  4  driverCategory VARCHAR2(20) NOT NULL, driverSalary NUMBER(8) NOT NULL);

Table created.

SQL> DESC driver;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DRIVERID                                  NOT NULL VARCHAR2(4)
 DRIVERNAME                                NOT NULL VARCHAR2(40)
 DRIVERADDRESS                             NOT NULL VARCHAR2(60)
 DIVEREMAIL                                NOT NULL VARCHAR2(60)
 DRIVERCONTACTNO                           NOT NULL VARCHAR2(15)
 DRIVERDOB                                 NOT NULL DATE
 DRIVERGENDER                              NOT NULL VARCHAR2(10)
 DRIVERCATEGORY                            NOT NULL VARCHAR2(20)
 DRIVERSALARY                              NOT NULL NUMBER(8)

SQL>
```

Figure 6: Creating driver table and seeing its structure

5.a.3: vehicle Table: Creation of table based on these attributes and its structure is shown

Attributes	Data Type	Constraints	Description
vehicleID	VARCHAR2 (4)	PRIMARY KEY, UNIQUE	This is the unique ID to distinguish the vehicles used for the services.
vehicleType	VARCHAR2 (20)	NOT NULL	This is the type of vehicle like a bike, car, etc.
vehicleCost	NUMBER (10)	NOT NULL	This is the price of the vehicle when purchased.
fuelType	VARCHAR2 (10)	NOT NULL	Indicated what type of fuel is used
vehicleModel	VARCHAR2 (30)	NOT NULL	Attribute that stores model of the vehicle.
vehicleVariant	VARCHAR2 (20)	NOT NULL	Indicated the variant of the model of the vehicle.
vehicleBaseCost	NUMBER (5)	NOT NULL	This is the base price of a vehicle when they are used in a service.

Table 6:The table driver with all of its attributes

Creating vehicle table and seeing its structure.

```
SQL> CREATE TABLE vehicle(vehicleID VARCHAR2(4) PRIMARY KEY, vehicleType VARCHAR2(20) NOT NULL,
  2  fuelType VARCHAR2(10) NOT NULL, vehicleModel VARCHAR2(30) NOT NULL, vehicleVariant VARCHAR2(20) NOT NULL,
  3  vehicleBaseCost NUMBER(5) NOT NULL, vehicleCost NUMBER(10) NOT NULL);

Table created.

SQL> DESC vehicle;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VEHICLEID                                 NOT NULL VARCHAR2(4)
 VEHICLETYPE                               NOT NULL VARCHAR2(20)
 FUELTYPE                                  NOT NULL VARCHAR2(10)
 VEHICLEMODEL                              NOT NULL VARCHAR2(30)
 VEHICLEVARIANT                            NOT NULL VARCHAR2(20)
 VEHICLEBASECOST                           NOT NULL NUMBER(5)
 VEHICLECOST                               NOT NULL NUMBER(10)

SQL>
```

Figure 7: Creating vehicle table and seeing its structure

5.a.4: invoice Table: Creation of table based on these attributes and its structure is shown

Attributes	Data Type	Constraints	Description
transactionID	VARCHAR2 (4)	PRIMARY KEY, UNIQUE	This is the unique ID to distinguish invoices generated for the services.
paymentMethod	VARCHAR (50)	NOT NULL	Details of payment method and amount paid.

Table 7: Table invoice attributes list

Creating table invoice and seeing its structure

```
SQL> CREATE TABLE invoice(transactionID VARCHAR2(4) PRIMARY KEY, paymentMethod VARCHAR2(50) NOT NULL);

Table created.

SQL> DESC invoice;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TRANSACTIONID                             NOT NULL VARCHAR2(4)
 PAYMENTMETHOD                             NOT NULL VARCHAR2(50)

SQL>
```

Figure 8: Creating invoice table and seeing its structure

5.a.5: ticket Table: Creation of table based on these attributes and its structure is shown

Attributes	Data Type	Constraints	Description
ticketNo	VARCHAR2 (4)	PRIMARY KEY, UNIQUE	This is the unique ID to distinguish the tickets generated for each services customer uses.
issuedDate	DATE	NOT NULL	The date when the service has begun.
issuedTime	VARCHAR2 (10)	NOT NULL	The time when the service has begun.
totalDistance	NUMBER (3,1)	NOT NULL	The total distance from initial location to destination.
destination	VARCHAR2 (20)	NOT NULL	The location where the service comes to an end.
initialLocation	VARCHAR2(20)	NOT NULL	The location where the service is started from.
serviceDuration	NUMBER (3)	NOT NULL	Total amount of time for the service to complete once begun.
totalCharge	NUMBER (4)	NOT NULL	The total amount for the service which takes account of base vehicle price, service charge, discounts, etc.
rewardPoints	NUMBER (5)	NOT NULL	The amount of points customer gets for using a service.

Table 8: All the attributes of ticket table

Creating ticket table and seeing its structure.

```
SQL> CREATE TABLE ticket(ticketNo VARCHAR2(4) PRIMARY KEY, issuedDate DATE NOT NULL,
  2  issuedTime VARCHAR2(10) NOT NULL, totalDistance NUMBER(3,1) NOT NULL,
  3  destination VARCHAR2(20) NOT NULL, initialLocation VARCHAR2(20) NOT NULL,
  4  serviceDuration NUMBER(3) NOT NULL, totalCharge NUMBER(4) NOT NULL,
  5  rewardPoints NUMBER(5) NOT NULL);

Table created.

SQL> DESC ticket;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TICKETNO                                  NOT NULL VARCHAR2(4)
 ISSUEDDATE                                NOT NULL DATE
 ISSUEDTIME                                NOT NULL VARCHAR2(10)
 TOTALDISTANCE                             NOT NULL NUMBER(3,1)
 DESTINATION                               NOT NULL VARCHAR2(20)
 INITIALLOCATION                           NOT NULL VARCHAR2(20)
 SERVICEDURATION                           NOT NULL NUMBER(3)
 TOTALCHARGE                               NOT NULL NUMBER(4)
 REWARDPOINTS                              NOT NULL NUMBER(5)

SQL>
```

Figure 9: Creating ticket table and seeing its structure

5.a.6: service Table: Creation of table based on these attributes and its structure is shown

Attributes	Data Type	Constraints	Description
serviceID	VARCHAR2 (4)	PRIMARY KEY, UNIQUE	This is the unique ID to distinguish the service used by a customer.
ticketNo	VARCHAR2 (4)	FOREIGN KEY, NOT NULL	This is the unique ID to distinguish the tickets generated for each services customer uses.
transactionID	VARCHAR2 (4)	FOREIGN KEY, NOT NULL	This is the unique ID to distinguish invoices generated for the services.
serviceBaseCharge	NUMBER (2)	NOT NULL	The base price for a service used.
chargePerKm	NUMBER (3)	NOT NULL	The total cost per km for the specific service used.
serviceCategory	VARCHAR2 (40)	NOT NULL	The names of the services that are available to use.

Table 9: Attributes of service table

Creating service table and seeing its structure

```
SQL> CREATE TABLE service(serviceID VARCHAR2(4) PRIMARY KEY, ticketNo VARCHAR2(4), transactionID VARCHAR2(4),
  2  serviceCategory VARCHAR2(40) NOT NULL, serviceBaseCharge NUMBER(2) NOT NULL, chargePerKm NUMBER(3) NOT NULL,
  3  FOREIGN KEY (ticketNo) REFERENCES ticket(ticketNo),
  4  FOREIGN KEY (transactionID) REFERENCES invoice(transactionID));

Table created.

SQL> DESC service
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SERVICEID                                 NOT NULL VARCHAR2(4)
 TICKETNO                                           VARCHAR2(4)
 TRANSACTIONID                                      VARCHAR2(4)
 SERVICECATEGORY                           NOT NULL VARCHAR2(40)
 SERVICEBASECHARGE                         NOT NULL NUMBER(2)
 CHARGEPERKM                               NOT NULL NUMBER(3)

SQL>
```

Figure 10: Creating service table and seeing its structure

5.a.7: customerOrder Table: Creation of table based on these attributes and its structure is shown

Attributes	Data Type	Constraints	Description
serviceID	VARCHAR2 (4)	FOREIGN KEY, UNIQUE	This is the unique ID to distinguish the service used by a customer.
customerID	VARCHAR2 (4)	FOREIGN KEY, NOT NULL	This is the unique ID to distinguish the customers that use the services.

Table 10: Atributes of customerOrder table

Creating customerOrder table and seeing its structure

```
SQL> CREATE TABLE customerorder(serviceID VARCHAR2(4), customerID VARCHAR2(4),
  2  FOREIGN KEY (serviceID) REFERENCES service(serviceID),
  3  FOREIGN KEY (customerID) REFERENCES customer(customerID));

Table created.

SQL> DESC customerorder;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SERVICEID                                          VARCHAR2(4)
 CUSTOMERID                                         VARCHAR2(4)

SQL>
```

Figure 11: Creating customerOrder table and seeing its structure

5.a.8: serviceDeliverer Table: Creation of table based on these attributes and its structure is shown

Attributes	Data Type	Constraints	Description
serviceID	VARCHAR2 (4)	FOREIGN KEY, NOT NULL	This is the unique ID to distinguish the service used by a customer.
driverID	VARCHAR2 (4)	FOREIGN KEY, NOT NULL	This is the unique ID to distinguish the driver for the services.
vehicleID	VARCHAR2 (4)	FOREIGN KEY, NOT NULL	This is the unique ID to distinguish the vehicles used for the services.

Table 11: serviceDeliverer Attributes table

Creating serviceDeliverer table and seeing its structure

```
SQL> CREATE TABLE servicedeliverer(serviceID VARCHAR2(4), driverID VARCHAR2(4), vehicleID VARCHAR2(4),
  2  FOREIGN KEY (serviceID) REFERENCES service(serviceID),
  3  FOREIGN KEY (driverID) REFERENCES driver(driverID),
  4  FOREIGN KEY (vehicleID) REFERENCES vehicle(vehicleID));

Table created.

SQL> DESC servicedeliverer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SERVICEID                                          VARCHAR2(4)
 DRIVERID                                           VARCHAR2(4)
 VEHICLEID                                          VARCHAR2(4)

SQL>
```

Figure 12: Creating serviceDeliverer table and seeing its structure

5.b: Screenshots of the SQL Command and the overall rows of table data (Data Entry and Table Data Display).

5.b.1: customer Table: Data entry and display.

Entering the following data into the customer table using INSERT INTO customer VALUES(); command.

```
SQL> INSERT INTO customer VALUES('C001', 'Mia Ratkovic', 'Sydney, Australia', '444-020-8991', 'mia.ratk1@gmail.com', 'Staff', '20%', '11-Jan-98', 'Female');

1 row created.

SQL> INSERT INTO customer VALUES('C002', 'Chelsea Souter', 'Canberra, Australia', '412-100-8991', 'chelsea12souter@yahoo.com', 'General', '0%', '28-Feb-97', 'Female');

1 row created.

SQL> INSERT INTO customer VALUES('C003', 'Madison Chisolmn', 'Winnipeg, Canada', '431-220-1202', 'madi.chisolm11@gmail.com', 'Staff', '20%', '20-Sep-98', 'Female');

1 row created.

SQL> INSERT INTO customer VALUES('C004', 'Anna Verbytska', 'Toronto, Canada', '128-230-9999', 'verbytska.anna@gmail.com', 'General', '0%', '19-Oct-96', 'Female');

1 row created.

SQL> INSERT INTO customer VALUES('C005', 'Samadhi Waduge', 'Colombo, Sri Lanka', '209-910-8788', 'samWaduge69@gmail.com', 'General', '0%', '22-Aug-00', 'Female');

1 row created.

SQL> INSERT INTO customer VALUES('C006', 'Rosaria Battista', 'Amsterdamn, Netherland', '116-000-9991', 'rose22Battista@outlook.com', 'General', '0%', '15-Aug-97', 'Female');

1 row created.

SQL> INSERT INTO customer VALUES('C007', 'Pharista Poudel', 'Auckland, New Zealand', '129-080-1000', 'angel-phar24@gmail.com', 'Staff', '20%', '21-Mar-98', 'Female');

1 row created.

SQL> INSERT INTO customer VALUES('C008', 'Aarzo Sapkota', 'Auckland, New Zealand', '129-020-8888', 'arzu.sap99@gmail.com', 'General', '0%', '01-Jan-99', 'Female');

1 row created.

SQL>
```

Figure 13: Inserting 8 rows of data in customer table

Command Used: SELECT * FROM customer; to display the customer table data.

```
SQL> SELECT * FROM customer;

CUST CUSTOMERNAME                    CUSTOMERADDRESS                                                  CUSTOMERPHONE   CUSTOMEREMAIL                                                    CUSTOMERCATEGOR CUSTO CUSTOMERD CUSTOMERGE
---- ------------------------------- ---------------------------------------------------------------- --------------- ---------------------------------------------------------------- --------------- ----- --------- ----------
C001 Mia Ratkovic                    Sydney, Australia                                                444-020-8991    mia.ratk1@gmail.com                                              Staff           20%   11-JAN-98 Female
C002 Chelsea Souter                  Canberra, Australia                                              412-100-8991    chelsea12souter@yahoo.com                                        General         0%    28-FEB-97 Female
C003 Madison Chisolmn                Winnipeg, Canada                                                 431-220-1202    madi.chisolm11@gmail.com                                         Staff           20%   20-SEP-98 Female
C004 Anna Verbytska                  Toronto, Canada                                                  128-230-9999    verbytska.anna@gmail.com                                         General         0%    19-OCT-96 Female
C005 Samadhi Waduge                  Colombo, Sri Lanka                                               209-910-8788    samWaduge69@gmail.com                                            General         0%    22-AUG-00 Female
C006 Rosaria Battista                Amsterdamn, Netherland                                           116-000-9991    rose22Battista@outlook.com                                       General         0%    15-AUG-97 Female
C007 Pharista Poudel                 Auckland, New Zealand                                            129-080-1000    angel-phar24@gmail.com                                           Staff           20%   21-MAR-98 Female
C008 Aarzo Sapkota                   Auckland, New Zealand                                            129-020-8888    arzu.sap99@gmail.com                                             General         0%    01-JAN-99 Female

8 rows selected.

SQL>
```

Figure 14: Displaying all rows of data present in customer table

5.b.2: driver Table: Data entry and display.

Entering the following data into the driver table using INSERT INTO driver VALUES(); command.

```
SQL> INSERT INTO driver VALUES('D001', 'Deven Gurung', 'Baluwatar, Kathmandu', 'devengurung2000@gmail.com', '9818780607', '11-Nov-00', 'Male', 'Full-Time', 500000);

1 row created.

SQL> INSERT INTO driver VALUES('D002', 'Kaman Diwash Shrestha', 'Tokha, Kathmandu', 'kaman.shrestha@gmail.com', '9846253095', '20-Aug-02', 'Male', 'Full-Time', 550000);

1 row created.

SQL> INSERT INTO driver VALUES('D003', 'Samir Man Shrestha', 'Pharping, Kathmandu', 'samirshresyha@gmail.com', '9812423322', '29-Feb-01', 'Male', 'Part-Time', 300000);
INSERT INTO driver VALUES('D003', 'Samir Man Shrestha', 'Pharping, Kathmandu', 'samirshresyha@gmail.com', '9812423322', '29-Feb-01', 'Male', 'Part-Time', 300000)
                                                                                                                    *
ERROR at line 1:
ORA-01839: date not valid for month specified


SQL> INSERT INTO driver VALUES('D003', 'Samir Man Shrestha', 'Pharping, Kathmandu', 'samirshresyha@gmail.com', '9812423322', '28-Feb-01', 'Male', 'Part-Time', 300000);

1 row created.

SQL> INSERT INTO driver VALUES('D004', 'Prashant Khanal', 'Lazimpat, Kathmandu', 'khanalprashant72@gmail.com', '9846253095', '12-Mar-97', 'Male', 'Part-Time', 350000);

1 row created.

SQL> INSERT INTO driver VALUES('D005', 'Pukar Krishna Karmacharya', 'Hatiban, Lalitpur', 'krishnaKar@outlook.com', '9846900012', '13-Jun-90', 'Male', 'Full-Time', 600000);

1 row created.

SQL> INSERT INTO driver VALUES('D006', 'Biswash Sherpa', 'New Baneshwor, Kathmandu', 'malla.abhaas96@gmail.com', '9862787799', '28-Jul-96', 'Male', 'Full-Time', 650000);

1 row created.

SQL> INSERT INTO driver VALUES('D007', 'Mayand Malla', 'London, UK', 'mrmario@gmail.com', '982800535', '21-Feb-01', 'Male', 'Full-Time', 580000);

1 row created.

SQL> INSERT INTO driver VALUES('D008', 'Abhishek Anand', 'New York City, United States','abhishek1997@gmail.com', '200-122-0000', '26-Feb-97', 'Male', 'Full- Time', 800000);

1 row created.

SQL>
```

Figure 15: Inserting 8 rows of data into driver table

Command Used: SELECT * FROM driver; to display the driver table data.

```
SQL> SELECT * FROM driver;

DRIV DRIVERNAME                    DRIVERADDRESS                    DRIVEREMAIL                    DRIVERCONTACTNO DRIVERDOB DRIVERGEND DRIVERCATEGORY       DRIVERSALARY
---- ----------------------------- -------------------------------- ------------------------------ --------------- --------- ---------- -------------------- ------------
D001 Deven Gurung                  Baluwatar, Kathmandu             devengurung2000@gmail.com      9818780607      11-NOV-00 Male       Full-Time                  500000
D002 Kaman Diwash Shrestha         Tokha, Kathmandu                 kaman.shrestha@gmail.com       9846253095      20-AUG-02 Male       Full-Time                  550000
D003 Samir Man Shrestha            Pharping, Kathmandu              samirshresyha@gmail.com        9812423322      28-FEB-01 Male       Part-Time                  300000
D004 Prashant Khanal               Lazimpat, Kathmandu              khanalprashant72@gmail.com     9846253095      12-MAR-97 Male       Part-Time                  350000
D005 Pukar Krishna Karmacharya     Hatiban, Lalitpur                krishnaKar@outlook.com         9846900012      13-JUN-90 Male       Full-Time                  600000
D006 Biswash Sherpa                New Baneshwor, Kathmandu         malla.abhaas96@gmail.com       9862787799      28-JUL-96 Male       Full-Time                  650000
D007 Mayand Malla                  London, UK                       mrmario@gmail.com              982800535       21-FEB-01 Male       Full-Time                  580000
D008 Abhishek Anand                New York City, United States     abhishek1997@gmail.com         200-122-0000    26-FEB-97 Male       Full- Time                 800000

8 rows selected.

SQL>
```

Figure 16: Displaying all rows of data present in driver table

5.b.3: vehicle Table: Data entry and display.

Entering the following data into the vehicle table using INSERT INTO vehicle VALUES(); command.

```
SQL> INSERT INTO vehicle VALUES ('V001', 'Car', 'Electric', 'Tesla X', 'Model X', 200, 18000000);

1 row created.

SQL> INSERT INTO vehicle VALUES ('V002', 'Motorbike', 'Petrol', 'KTM Duke' , '390', 100, 580000);

1 row created.

SQL> INSERT INTO vehicle VALUES('V003', 'Car', 'Electric', 'Tesla Y', 'Model Y', 220, 15000000);

1 row created.

SQL> INSERT INTO vehicle VALUES('V004', 'Car', 'Petrol', 'Volkswagen Polo', 'Highline 1.6', 150, 3900000);

1 row created.

SQL> INSERT INTO vehicle VALUES('V005', 'Motorbike', 'Petrol', 'CF Moto NK', '250', 110, 750000);

1 row created.

SQL> INSERT INTO vehicle VALUES('V006', 'Van', 'Diesel', 'Toyota Hiace', 'Hiace Pickup', 265, 7100000);

1 row created.

SQL> INSERT INTO vehicle VALUES('V007', 'Motorbike', 'Petrol', 'Hero Honda Splendor', '125', 80, 300000);

1 row created.

SQL> INSERT INTO vehicle VALUES('V008', 'Car', 'Eletric', 'Toyota Prius', 'Prius Hybrid', 95, 3500000);

1 row created.

SQL> INSERT INTO vehicle VALUES('V009', 'Motorbike', 'Petrol', 'Honda Shine', '160', 85, 350000);

1 row created.

SQL> INSERT INTO vehicle VALUES('V010', 'Motorbike', 'Petrol', 'Yamaha R series', 'R1', 150, 800000);

1 row created.

SQL>
```

Figure 17:Inserting 10 rows of data into vehicle table

Command Used: SELECT * FROM vehicle; to display the vehicle table data.

```
SQL> SET linesize 300;
SQL> SELECT * FROM vehicle;

VEHI VEHICLETYPE          FUELTYPE   VEHICLEMODEL                  VEHICLEVARIANT       VEHICLEBASECOST VEHICLECOST
---- -------------------- ---------- ----------------------------- -------------------- --------------- -----------
V001 Car                  Electric   Tesla X                       Model X                          200    18000000
V002 Motorbike            Petrol     KTM Duke                      390                              100      580000
V003 Car                  Electric   Tesla Y                       Model Y                          220    15000000
V004 Car                  Petrol     Volkswagen Polo               Highline 1.6                     150     3900000
V005 Motorbike            Petrol     CF Moto NK                    250                              110      750000
V006 Van                  Diesel     Toyota Hiace                  Hiace Pickup                     265     7100000
V007 Motorbike            Petrol     Hero Honda Splendor           125                               80      300000
V008 Car                  Eletric    Toyota Prius                  Prius Hybrid                      95     3500000
V009 Motorbike            Petrol     Honda Shine                   160                               85      350000
V010 Motorbike            Petrol     Yamaha R series               R1                               150      800000

10 rows selected.

SQL>
```

Figure 18: Displaying all rows of data present in vehicle table

5.b.4: ticket Table: Data entry and display.

Entering the following data into the driver table using INSERT INTO driver VALUES(); command.

```
SQL> INSERT INTO ticket VALUES('T001', '11-Aug-20', '09:00', 05.0, 'Tokha', 'Putalisadak', 20, 405, 200);

1 row created.

SQL> INSERT INTO ticket VALUES('T002', '06-Mar-21', '23:10', 05.0, 'Tokha', 'Putalisadak', 20, 355, 200);

1 row created.

SQL> INSERT INTO ticket VALUES('T003', '08-Mar-21', '13:26', 22.3, 'Naxal', 'Patlekhet', 130, 358, 1300);

1 row created.

SQL> INSERT INTO ticket VALUES('T004', '05-Mar-22', '13:26', 22.3, 'Naxal', 'Patlekhet', 130, 286, 1300);

1 row created.

SQL> INSERT INTO ticket VALUES('T005', '06-Mar-22', '15:30', 11.0, 'Samakhusi', 'Naxal', 40, 245 , 400);

1 row created.

SQL> INSERT INTO ticket VALUES('T006', '09-Mar-22', '12:22', 11.0, 'Tokha', 'Putalisadak', 35, 288, 350);

1 row created.

SQL> INSERT INTO ticket VALUES('T007', '09-Mar-22', '11:20', 28.6, 'Baluwatar', 'Boudha', 55, 551, 550);

1 row created.

SQL> INSERT INTO ticket VALUES('T008', '05-Apr-22', '15:30', 03.5, 'Maharajgunj', 'Baluwatar', 15, 165, 150);

1 row created.

SQL> INSERT INTO ticket VALUES('T009', '09-Apr-22', '15:30', 20.5, 'Budhanilkantha', 'Koteshwor', 50, 518, 500);

1 row created.

SQL> INSERT INTO ticket VALUES('T010', '10-Apr-22', '21:18', 08.0, 'Nagpokhari', 'Maharajgunj', 45, 280, 450);

1 row created.

SQL> INSERT INTO ticket VALUES('T011', '27-Jun-22', '17:59', 08.0, 'Nagpokhari', 'Maharajgunj', 45, 350, 450);

1 row created.

SQL> INSERT INTO ticket VALUES('T012', '22-Sep-22', '16:49', 12.4, 'New Baneshwor', 'Putalisadak', 40, 350, 400);

1 row created.

SQL> INSERT INTO ticket VALUES('T013', '31-Dec-22', '13:45', 30.5, 'Bhaisepati', 'Budhanilkantha', 120, 620, 1200);

1 row created.

SQL> INSERT INTO ticket VALUES('T014', '01-Jan-23', '22:30', 30.5, 'Bhaisepati', 'Budhanilkantha', 120, 567, 1200);

1 row created.

SQL> INSERT INTO ticket VALUES('T015', '01-Jan-23', '22:30', 17.2, 'New Baneshower', 'Tokha', 60, 428, 600);

1 row created.

SQL> INSERT INTO ticket VALUES('T016', '02-Jan-23', '00:15', 19.6, 'Chobhar', 'Putalisadak', 75, 454, 750);

1 row created.

SQL> INSERT INTO ticket VALUES('T017', '04-Jan-23', '12:30', 01.7, 'Kamal Pokhari', 'Putalisadak', 5, 145, 50);

1 row created.

SQL>
```

Figure 19: Inserting 17 rows of data into ticket table

Command Used: SELECT * FROM ticket; to display the ticket table data.

```
SQL> SELECT * FROM ticket;

TICK ISSUEDDAT ISSUEDTIME TOTALDISTANCE DESTINATION          INITIALLOCATION      SERVICEDURATION TOTALCHARGE REWARDPOINTS
---- --------- ---------- ------------- -------------------- -------------------- --------------- ------------ ------------
T001 11-AUG-20 09:00                  5 Tokha                Putalisadak                       20          405          200
T002 06-MAR-21 23:10                  5 Tokha                Putalisadak                       20          355          200
T003 08-MAR-21 13:26               22.3 Naxal                Patlekhet                        130          358         1300
T004 05-MAR-22 13:26               22.3 Naxal                Patlekhet                        130          286         1300
T005 06-MAR-22 15:30                 11 Samakhusi            Naxal                             40          245          400
T006 09-MAR-22 12:22                 11 Tokha                Putalisadak                       35          288          350
T007 09-MAR-22 11:20               28.6 Baluwatar            Boudha                            55          551          550
T008 05-APR-22 15:30                3.5 Maharajgunj          Baluwatar                         15          165          150
T009 09-APR-22 15:30               20.5 Budhanilkantha       Koteshwor                         50          518          500
T010 10-APR-22 21:18                  8 Nagpokhari           Maharajgunj                       45          280          450
T011 27-JUN-22 17:59                  8 Nagpokhari           Maharajgunj                       45          350          450
T012 22-SEP-22 16:49               12.4 New Baneshwor        Putalisadak                       40          350          400
T013 31-DEC-22 13:45               30.5 Bhaisepati           Budhanilkantha                   120          620         1200
T014 01-JAN-23 22:30               30.5 Bhaisepati           Budhanilkantha                   120          567         1200
T015 01-JAN-23 22:30               17.2 New Baneshower       Tokha                             60          428          600
T016 02-JAN-23 00:15               19.6 Chobhar              Putalisadak                       75          454          750
T017 04-JAN-23 12:30                1.7 Kamal Pokhari        Putalisadak                        5          145           50

17 rows selected.

SQL>
```

Figure 20: Displaying all the rows of data in ticket table

5.b.5: invoice Table: Data entry and display.

Entering the following data into the invoice table using INSERT INTO invoice VALUES(); command.

```
SQL> INSERT INTO invoice VALUES('I001', 'Credit Card: Rs. 405 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I002', 'Credit Card: Rs. 355 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I003', 'Esewa: Rs. 358 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I004', 'Esewa: Rs. 286 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I005', 'FonePay: Rs. 245 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I006', 'Cash: Rs. 288 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I007', 'FonePay: Rs. 551 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I008', 'Credit Card: Rs. 165 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I009', 'Esewa: Rs. 518 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I010', 'Cash: Rs. 280 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I011', 'Cash: Rs. 350 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I012', 'FonePay: Rs. 350 paid');

1 row created.
```

```
SQL> INSERT INTO invoice VALUES('I013', 'Credit Card: Rs. 620 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I014', 'Debit Card: Rs. 567 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I015', 'Esewa: Rs. 428 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I016', 'Cash: Rs. 454 paid');

1 row created.

SQL> INSERT INTO invoice VALUES('I017', 'Credit Card: Rs. 145 paid');

1 row created.

SQL>
```

Figure 21: Inserting 17 rows of data into invoice table

Command Used: SELECT * FROM service; to display the service table data.

```
SQL> SELECT * FROM invoice;

TRAN PAYMENTMETHOD
---- --------------------------------------------------
I001 Credit Card: Rs. 405 paid
I002 Credit Card: Rs. 355 paid
I003 Esewa: Rs. 358 paid
I004 Esewa: Rs. 286 paid
I005 FonePay: Rs. 245 paid
I006 Cash: Rs. 288 paid
I007 FonePay: Rs. 551 paid
I008 Credit Card: Rs. 165 paid
I009 Esewa: Rs. 518 paid
I010 Cash: Rs. 280 paid
I011 Cash: Rs. 350 paid
I012 FonePay: Rs. 350 paid
I013 Credit Card: Rs. 620 paid
I014 Debit Card: Rs. 567 paid
I015 Esewa: Rs. 428 paid
I016 Cash: Rs. 454 paid
I017 Credit Card: Rs. 145 paid

17 rows selected.
```

Figure 22: Displaying all the rows of data in invoice table

5.b.6: service Table: Data entry and display.

Entering the following data into the service table using INSERT INTO service VALUES(); command.

```
SQL> INSERT INTO service VALUES('S001', 'T001', 'I001', 'Product Delivery', 80, 25);

1 row created.

SQL> INSERT INTO service VALUES('S002', 'T002', 'I002', 'Product Delivery', 80, 25);

1 row created.

SQL> INSERT INTO service VALUES('S003', 'T003', 'I003', 'Rides', 50, 10);

1 row created.

SQL> INSERT INTO service VALUES('S004', 'T004', 'I004', 'Rides', 50, 10);

1 row created.

SQL> INSERT INTO service VALUES('S005', 'T005', 'I005', 'Rides', 50, 10);

1 row created.

SQL> INSERT INTO service VALUES('S006', 'T006', 'I006', 'Courier Delivery', 40, 20);

1 row created.

SQL> INSERT INTO service VALUES('S007', 'T007', 'I007', 'Food Delivery', 60, 15);

1 row created.

SQL> INSERT INTO service VALUES('S008', 'T008', 'I008', 'Rides', 50, 10);

1 row created.

SQL> INSERT INTO service VALUES('S009', 'T009', 'I009', 'Food Delivery', 60, 15);

1 row created.

SQL> INSERT INTO service VALUES('S010', 'T010', 'I010', 'Rides', 50, 10);

1 row created.

SQL> INSERT INTO service VALUES('S011', 'T011', 'I011', 'Rides', 50, 10);

1 row created.

SQL> INSERT INTO service VALUES('S012', 'T012', 'I012', 'Courier Delivery', 40, 20);

1 row created.

SQL> INSERT INTO service VALUES('S013', 'T013', 'I013', 'Rides', 50, 10);
```

```
SQL> INSERT INTO service VALUES('S013', 'T013', 'I013', 'Rides', 50, 10);

1 row created.

SQL> INSERT INTO service VALUES('S014', 'T014', 'I014', 'Rides', 50, 10);

1 row created.

SQL> INSERT INTO service VALUES('S015', 'T015', 'I015', 'Food Delivery', 60, 15);

1 row created.

SQL> INSERT INTO service VALUES('S016', 'T016', 'I016', 'Food Delivery', 60, 15);

1 row created.

SQL> INSERT INTO service VALUES('S017', 'T017', 'I017', 'Rides', 50, 10);

1 row created.

SQL>
```

Figure 23: Inserting 17 rows of data into service

Command Used: SELECT * FROM service; to display the service table data.

```
SQL> SELECT * FROM service;

SERV TICK TRAN SERVICECATEGORY                       SERVICEBASECHARGE CHARGEPERKM
---- ---- ---- ---------------------------------------- ----------------- -----------
S001 T001 I001 Product Delivery                                      80          25
S002 T002 I002 Product Delivery                                      80          25
S003 T003 I003 Rides                                                 50          10
S004 T004 I004 Rides                                                 50          10
S005 T005 I005 Rides                                                 50          10
S006 T006 I006 Courier Delivery                                      40          20
S007 T007 I007 Food Delivery                                         60          15
S008 T008 I008 Rides                                                 50          10
S009 T009 I009 Food Delivery                                         60          15
S010 T010 I010 Rides                                                 50          10
S011 T011 I011 Rides                                                 50          10
S012 T012 I012 Courier Delivery                                      40          20
S013 T013 I013 Rides                                                 50          10
S014 T014 I014 Rides                                                 50          10
S015 T015 I015 Food Delivery                                         60          15
S016 T016 I016 Food Delivery                                         60          15
S017 T017 I017 Rides                                                 50          10

17 rows selected.
```

Figure 24: Displaying all the rows of data in service table

5.b.7: customerOrder Table: Data entry and display.

Entering the following data into the customerOrder table using INSERT INTO customerOrder VALUES(); command.

```
SQL> INSERT INTO customerorder VALUES('S001', 'C004');

1 row created.

SQL> INSERT INTO customerorder VALUES('S002', 'C002');

1 row created.

SQL> INSERT INTO customerorder VALUES('S003', 'C005');

1 row created.

SQL> INSERT INTO customerorder VALUES('S004', 'C007');

1 row created.

SQL> INSERT INTO customerorder VALUES('S005', 'C006');

1 row created.

SQL> INSERT INTO customerorder VALUES('S006', 'C001');

1 row created.

SQL> INSERT INTO customerorder VALUES('S007', 'C007');

1 row created.

SQL> INSERT INTO customerorder VALUES('S008', 'C008');

1 row created.

SQL> INSERT INTO customerorder VALUES('S009', 'C006');

1 row created.

SQL> INSERT INTO customerorder VALUES('S010', 'C007');

1 row created.

SQL> INSERT INTO customerorder VALUES('S011', 'C006');

1 row created.

SQL> INSERT INTO customerorder VALUES('S012', 'C003');

1 row created.
```

Figure 25: Inserting first 12 rows of data into customerOrder table

```
SQL> INSERT INTO customerorder VALUES('S013', 'C004');

1 row created.

SQL> INSERT INTO customerorder VALUES('S014'. 'C007');
INSERT INTO customerorder VALUES('S014'. 'C007')
                                        *
ERROR at line 1:
ORA-00917: missing comma


SQL> INSERT INTO customerorder VALUES('S014', 'C007');

1 row created.

SQL> INSERT INTO customerorder VALUES('S015', 'C005');

1 row created.

SQL> INSERT INTO customerorder VALUES('S016', 'C005');

1 row created.

SQL> INSERT INTO customerorder VALUES('S017', 'C005');

1 row created.

SQL>
```

Figure 26: Entering 5 more rows of data into CustomerOrder table

Command Used: SELECT * FROM customerOrder; to display the customerOrder table data.

```
SQL> SELECT * FROM customerOrder;

SERV CUST
---- ----
S001 C004
S002 C002
S003 C005
S004 C007
S005 C006
S006 C001
S007 C007
S008 C008
S009 C006
S010 C007
S011 C006
S012 C003
S013 C004
S014 C007
S015 C005
S016 C005
S017 C005

17 rows selected.

SQL>
```

Figure 27: Displaying all rows of data from the customerOrder table

5.b.8: serviceDeliverer Table: Data entry and display.

Entering the following data into the serviceDeliverer table using INSERT INTO serviceDeliverer VALUES(); command.

```
SQL> INSERT INTO servicedeliverer VALUES('S001', 'D002', 'V001');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S002', 'D004', 'V010');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S003', 'D001', 'V009');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S004', 'D005', 'V009');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S005', 'D003', 'V005');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S006', 'D006', 'V002');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S007', 'D004', 'V001');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S008', 'D004', 'V008');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S009', 'D005', 'V004');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S010', 'D007', 'V003');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S011', 'D008', 'V003');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S012', 'D006', 'V010');

1 row created.
```

Figure 28: Inserting first 12 rows of data into the serviceDeliverer table

```
SQL> INSERT INTO servicedeliverer VALUES('S013', 'D008', 'V006');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S014'. 'D007', 'V006');
INSERT INTO servicedeliverer VALUES('S014'. 'D007', 'V006')
                                       *
ERROR at line 1:
ORA-00917: missing comma


SQL> INSERT INTO servicedeliverer VALUES('S014', 'D007', 'V006');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S015', 'D008', 'V005');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S016', 'D008', 'V002');

1 row created.

SQL> INSERT INTO servicedeliverer VALUES('S017', 'D008', 'V007');

1 row created.

SQL>
```

Figure 29: Inserting 5 more rows of data into serviceDeliverer Table

Command Used: SELECT * FROM serviceDeliverer; to display the serviceDeliverer table data.

```
SQL> SELECT * FROM servicedeliverer;

SERV DRIV VEHI
---- ---- ----
S001 D002 V001
S002 D004 V010
S003 D001 V009
S004 D005 V009
S005 D003 V005
S006 D006 V002
S007 D004 V001
S008 D004 V008
S009 D005 V004
S010 D007 V003
S011 D008 V003
S012 D006 V010
S013 D008 V006
S014 D007 V006
S015 D008 V005
S016 D008 V002
S017 D008 V007

17 rows selected.

SQL>
```

Figure 30: Displaying all rows of data in serviceDeliverer table

Part 6: Database Querying

6.1: Informational Queries

6.1.a: Listing customers according to category:

SQL Command: SELECT * FROM customer ORDER BY customerCategory;

```
SQL> SELECT * FROM customer ORDER BY customerCategory;

CUST CUSTOMERNAME                    CUSTOMERADDRESS                                    CUSTOMERPHONE   CUSTOMEREMAIL                                      CUSTOMERCATEGOR CUSTO CUSTOMERD CUSTOMERGE
---- ------------------------------- -------------------------------------------------- --------------- -------------------------------------------------- --------------- ----- --------- ---------
C004 Anna Verbytska                  Toronto, Canada                                    128-230-9999    verbytska.anna@gmail.com                           General         0%    19-OCT-96 Female
C006 Rosaria Battista                Amsterdamn, Netherland                             116-000-9991    rose22Battista@outlook.com                         General         0%    15-AUG-97 Female
C005 Samadhi Wadugo                  Colombo, Sri Lanka                                 209-910-8788    samWadugo69@gmail.com                              General         0%    22-AUG-00 Female
C002 Chelsea Souter                  Canberra, Australia                                412-100-8991    chelsea12souter@yahoo.com                          General         0%    28-FEB-97 Female
C008 Aarzo Sapkota                   Auckland, New Zealand                              129-020-8888    arzu.sap99@gmail.com                               General         0%    01-JAN-99 Female
C003 Madison Chisolmn                Winnipeg, Canada                                   431-220-1202    madi.chisolm11@gmail.com                           Staff           20%   20-SEP-98 Female
C007 Pharista Poudel                 Auckland, New Zealand                              129-080-1000    angel-phar24@gmail.com                             Staff           20%   21-MAR-98 Female
C001 Mia Ratkovic                    Sydney, Australia                                  444-020-8991    mia.ratk1@gmail.com                                Staff           20%   11-JAN-98 Female

8 rows selected.

SQL>
```

Figure 31: Displaying details of customer according to their category

Output Analysis: Normal customers are shown first as ‘general’ then the ‘staff’ are shown.

6.1.b: Finding model and vehicle variants and sort by price in descending order.

SQL Command: SELECT vehicleModel, vehicleVariant FROM vehicle ORDER BY vehicleCost DESC;

```
SQL> SELECT vehicleModel, vehicleVariant, vehicleCost FROM vehicle ORDER BY vehicleCost DESC;

VEHICLEMODEL                   VEHICLEVARIANT       VEHICLECOST
------------------------------ -------------------- -----------
Tesla X                        Model X                 18000000
Tesla Y                        Model Y                 15000000
Toyota Hiace                   Hiace Pickup             7100000
Volkswagen Polo                Highline 1.6             3900000
Toyota Prius                   Prius Hybrid             3500000
Yamaha R series                R1                        800000
CF Moto NK                     250                       750000
KTM Duke                       390                       580000
Honda Shine                    160                       350000
Hero Honda Splendor            125                       300000

10 rows selected.

SQL>
```

Figure 32: Seeing vehicle model and variant by descending order

Output Analysis: We are shown the vehicles model and variant in order of most expensive to cheaper vehicle cost.

6.1.c: Displaying the number of total vehicles that use petrol.

SQL Command: SELECT COUNT(*) PetrolType FROM vehicle WHERE fuelType = 'Petrol';

```
SQL> SELECT COUNT(*) PetrolType FROM vehicle WHERE fuelType ='Petrol';

PETROLTYPE
----------
         6

SQL>
```

Figure 33: Seeing number of vehicles that use petrol

Output Analysis: We are shown the total number of vehicles that use petrol as their fuel type

6.1.d: Listing all tickets issued from 2022/03/05 to 2022/04/05.

SQL Command: SELECT * FROM ticket WHERE issuedDate BETWEEN '05-Mar-22' AND '05-Apr-22';

```
SQL> SELECT * FROM ticket WHERE issuedDate BETWEEN '05-Mar-22' AND '05-Apr-22';

TICK ISSUEDDAT ISSUEDTIME TOTALDISTANCE DESTINATION          INITIALLOCATION      SERVICEDURATION TOTALCHARGE REWARDPOINTS
---- --------- ---------- ------------- -------------------- -------------------- --------------- ----------- ------------
T004 05-MAR-22 13:26               22.3 Naxal                Patlekhet                       130         286         1300
T005 06-MAR-22 15:30                 11 Samakhusi            Naxal                           40          245          400
T006 09-MAR-22 12:22                 11 Tokha                Putalisadak                     35          288          350
T007 09-MAR-22 11:20               28.6 Baluwatar            Boudha                          55          551          550
T008 05-APR-22 15:30                3.5 Maharajgunj          Baluwatar                       15          165          150
```

Figure 34: Displays ticket issued for 1 month period in 2022

Output Analysis: This lists all the service tickets issued between Mar 5th 2022 to Apr 5th 2022.

6.1.e: Listing the name of the driver who has the character ‘s’ between their names.

SQL Command: SELECT driverName FROM driver WHERE driverName LIKE '%_s_%';

```
SQL> SELECT driverName FROM driver WHERE driverName LIKE '%_s_%';

DRIVERNAME
----------------------------------------
Kaman Diwash Shrestha
Samir Man Shrestha
Prashant Khanal
Pukar Krishna Karmacharya
Biswash Sherpa
Abhishek Anand

6 rows selected.

SQL>
```

Figure 35:Displays name of drivers that have 's' character between their names

Output Analysis: This shows a all the drivers who have ‘s’ character between their names.

6.2: Transactional Queries

6.2.a: Show the total cost and the type of service of a particular customer in a year that has used the service.

SQL Command:

SELECT t.totalCharge, s.serviceCategory FROM ticket t

JOIN service s

ON t.ticketNo = s.ticketNo

JOIN customerOrder cu

ON s.serviceID = cu.serviceID

JOIN customer c

ON cu.customerID = c.customerID

WHERE cu.customerID = 'C007' AND EXTRACT(YEAR FROM t.issuedDate) = '2022';

```
SQL> SELECT t.totalCharge, s.serviceCategory FROM ticket t
  2    JOIN service s
  3    ON t.ticketNo = s.ticketNo
  4    JOIN customerOrder cu
  5    ON s.serviceID = cu.serviceID
  6    JOIN customer c
  7    ON cu.customerID = c.customerID
  8    WHERE cu.customerID = 'C007' AND EXTRACT(YEAR FROM t.issuedDate) = '2022';

TOTALCHARGE SERVICECATEGORY
----------- ----------------------------------------
        286 Rides
        551 Food Delivery
        280 Rides

SQL>
```

Figure 36: Shows total cost of the services a customer used in 2022

Output Analysis: This shows us the total cost of the services that a customer with customerID ‘C007’ used in the year 2022.

6.2.b: List the details of services that have been provided by a driver for the current month whose first name starts with the letter ‘A’.

SQL Command:

SELECT s.serviceID, s.serviceCategory, s.serviceBaseCharge, s.chargePerKm, t.issuedDate FROM service s

JOIN serviceDeliverer sd

ON sd.serviceID = s.serviceID

JOIN Driver d

ON d.driverID = sd.driverID

JOIN ticket t

ON t.ticketNo = s.ticketNo

WHERE d.driverName LIKE 'A_%' AND EXTRACT(MONTH FROM Sysdate) = EXTRACT(MONTH FROM t.issuedDate);

```
SQL> SELECT s.serviceID, s.serviceCategory, s.serviceBaseCharge, s.chargePerKm, t.issuedDate FROM service s
  2  JOIN serviceDeliverer sd
  3  ON sd.serviceID = s.serviceID
  4  JOIN Driver d
  5  ON d.driverID = sd.driverID
  6  JOIN ticket t
  7  ON t.ticketNo = s.ticketNo
  8  WHERE d.driverName LIKE 'A_%' AND EXTRACT(MONTH FROM Sysdate) = EXTRACT(MONTH FROM t.issuedDate);

SERV SERVICECATEGORY                        SERVICEBASECHARGE CHARGEPERKM ISSUEDDAT
---- ---------------------------------------- ----------------- ----------- ---------
S015 Food Delivery                                         60          15 01-JAN-23
S016 Food Delivery                                         60          15 02-JAN-23
S017 Rides                                                 50          10 04-JAN-23

SQL>
```

Figure 37: Shows service provided by driver whose name begins with ‘A’ in Jan 2023

Output Analysis: Displaying the details of services provided by the driver ‘Abhishek Anand’ who is the only driver that has name beginning with letter ‘A’ in the database during the current month, i.e: Jan 2023.

6.2.c: List the details of customers who have used only courier service and their location of delivery.

SQL Command:

```
SELECT c.*, t.destination FROM customer c

JOIN customerOrder cu

ON cu.customerID = c.customerID

JOIN service s

ON s.serviceID = cu.serviceID

JOIN ticket t

ON s.ticketNo = t.ticketNo

WHERE s.serviceCategory = 'Courier Delivery';
```

Figure 38: Courier delivery customer details and delivery location/destination

Output Analysis: This command displays all the customers details that used courier delivery only with their delivery location which is the destination in our database.

6.2.d: List all the details of the top 3 highest earning drivers.

SQL Command:

```
SELECT * FROM

(SELECT * FROM driver ORDER BY driverSalary DESC)

WHERE ROWNUM<=3;
```

Figure 39: Top 3 highest paid drivers

Output Analysis: Shows top 3 highest paid drivers.

6.2.e: Display the rate of all vehicles for staff and normal customers on a particular destination.

SQL Command:

SELECT c.customerID, t.totalCharge, c.customerCategory, t.initialLocation, t.destination,v.vehicleID, v.vehicleType, v.vehicleModel FROM

Customer c

JOIN customerOrder

ON c.customerID = customerOrder.customerID

JOIN Service s

ON customerOrder.serviceID = s.serviceID

JOIN serviceDeliverer sd

ON s.serviceID = sd.serviceID

JOIN vehicle v

ON sd.vehicleID = v.vehicleID

JOIN ticket t

ON s.ticketNo = t.ticketNo

WHERE t.destination = 'Naxal';

```
SQL> SELECT c.customerID, t.totalCharge, c.customerCategory, t.initialLocation, t.destination,v.vehicleID, v.vehicleType, v.vehicleModel FROM
  2     Customer c
  3     JOIN customerOrder
  4     ON c.customerID = customerOrder.customerID
  5     JOIN Service s
  6     ON customerOrder.serviceID = s.serviceID
  7     JOIN serviceDeliverer sd
  8     ON s.serviceID = sd.serviceID
  9     JOIN vehicle v
 10     ON sd.vehicleID = v.vehicleID
 11     JOIN ticket t
 12     ON s.ticketNo = t.ticketNo
 13     WHERE t.destination = 'Naxal';

CUST TOTALCHARGE CUSTOMERCATEGOR INITIALLOCATION      DESTINATION          VEHI VEHICLETYPE          VEHICLEMODEL
---- ----------- --------------- -------------------- -------------------- ---- -------------------- ------------------------------
C005         358 General         Patlekhet            Naxal                V009 Motorbike            Honda Shine
C007         286 Staff           Patlekhet            Naxal                V009 Motorbike            Honda Shine

SQL>
```

Figure 40: Variation in price for staff and general customer going to same destination

Output Analysis: This command displays the cost of different vehicles for the staff and general customer and also discounted vehicle rates for the “Staff” heading to same destination.

Note: You can see difference in price of the vehicles based on other common destinations for the ride.

Part 7: File Creation

I have created the dump file for this database of Grab company. It is under the username ‘ana1487’ and the password is set as ‘anurag’.

```
Microsoft Windows [Version 10.0.22621.963]
(c) Microsoft Corporation. All rights reserved.

C:\Users\anura>cd C:\Users\anura\OneDrive\DumpFile

C:\Users\anura\OneDrive\DumpFile>exp ana1487/anurag file = 21039635AnuragAnand.dmp

Export: Release 11.2.0.2.0 - Production on Wed Jan 4 15:29:52 2023

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.


Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user ANA1487
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user ANA1487
About to export ANA1487's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export ANA1487's tables via Conventional Path ...
. . exporting table                       CUSTOMER          8 rows exported
. . exporting table                  CUSTOMERORDER         17 rows exported
. . exporting table                         DRIVER          8 rows exported
. . exporting table                        INVOICE         17 rows exported
. . exporting table                        SERVICE         17 rows exported
. . exporting table               SERVICEDELIVERER         17 rows exported
. . exporting table                         TICKET         17 rows exported
. . exporting table                        VEHICLE         10 rows exported
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
. exporting posttables actions
. exporting materialized views
. exporting snapshot logs
. exporting job queues
. exporting refresh groups and children
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully without warnings.

C:\Users\anura\OneDrive\DumpFile>
```

Figure 41:Dump file creation through command prompt

Part 8: Critical Evaluation

In this Databases module, the main learning outcomes were for us to create Entity-Relationship model/model from realistic scenarios where databases are needed to be implemented, use of formal design techniques like normalization to create a relational database schema, also designing and implementing a database system, using Oracle DBMS, from a conceptual data model as well as to manipulate and extract data stored in databases using relational algebra and SQL queries. The module did deliver in these aspects, and provided more in-depth information about all the types of relations tables can have with each other, ways to determine key attributes, and to understand why databases are a useful tool to design, especially relational databases which are used by most companies these days – there's an estimation that between 2021-2027 it's usage is expected to increase by at least 41% (businesswire, 2022). However, there are few things the module did not teach in detail as we only saw brief content on them such as normalization beyond 3NF. It would have been better if we could have used Boyce Codd Normal Form (BNF) to reduce all the data redundancies. However, the module did deliver on its learning outcomes well, and it has a relation with all other fields of study is present because we need to design and create databases for any kind of data collection be it in the field of science, technology, business, market, arts, etc in order to use those data as per need.

In the same way, this coursework has helped fulfil all those learning outcomes of this module. It has helped me understand the basics for designing Entity-relationship diagram for Grab company that provides four different services based on their business rules, allowed me to create its database schema on the basis of final ERD which can be achieved after the normalization process done looking at the initial ERD design and assumptions made for the company, allowed me to implement data and carry out informational and transactional queries as per the requirements and finally I was also able to create a dump file that saved the database schema in a file which can be shared to others. But, I realized later on that just using up to 3NF form of normalization, my service table isn't fully redundance free, hence if we were told to implement BCNF form in my database for Grab, it would have been fully redundancy free. In all, this coursework has broadened my view about the usage of databases in real life situations where companies use them and implement to do analysis on a large scale, I believe this coursework will enable me to create a proper database system for my Final Year Project.

References

businesswire. (2022, Feburary 10). *Worldwide Relational Database Industry to 2027 - Key Drivers, Restraints and Opportunities* . Retrieved from businesswire: https://www.businesswire.com/news/home/20220210005654/en/Worldwide-Relational-Database-Industry-to-2027---Key-Drivers-Restraints-and-Opportunities---ResearchAndMarkets.com

Chen Lin, A. D. (2022, 9 27). *Grab expects to break even by the second half of 2024*. Retrieved from Reuters: https://www.reuters.com/technology/grab-expects-break-even-its-digibank-operations-by-2026-2022-09-27/

Code Brew. (2021, 6 21). *How does grab work, business and revenue mode explained*. Retrieved from Code Brew Labs: https://www.code-brew.com/how-does-grab-work-business-and-revenue-model-explained/

DBpedia. (2023, January 5). *About: Unnormalized form* . Retrieved from DBpedia: https://dbpedia.org/page/Unnormalized_form

geeksforgeeks. (2022, October 28). *Difference Between Primary Key and Foreign Key*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/difference-between-primary-key-and-foreign-key/

Grab. (2022, 12 07). *About Us*. Retrieved from Grab: https://www.grab.com/sg/about/

Grab. (2022, May 19). *Press Release Details*. Retrieved from Grab Reports First Quarter 2022 Results: https://investors.grab.com/news-releases/news-release-details/grab-reports-first-quarter-2022-results

Kormos, J. (2022, Feburary 22). *The basics of database normalization*. Retrieved from lifewire: https://www.lifewire.com/database-normalization-basics-1019735