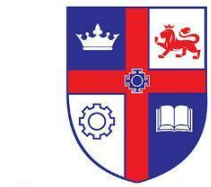
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

50% Individual Coursework

2022 Autumn

Student Name: SUMAN K.C.

London Met ID: 22015791

Assignment Submission Date: Thursday, January 5, 2023

Word Count: 6459

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

List of Figures

Table of Tables

1. Introduction

This report consists of database design and implementation for the company **PATHO NEPAL,** which was founded in Bangladesh by Fahim Ahmed. The company aims to expand its business, including Nepal, and has implemented a database to help manage and trace information related to its drivers, customer, and vehicles. The database uses advance technology and is used to support the company services, such as food delivery, ride sharing, courier delivery and package delivery which is designed to be helpful to users.

PATHAO NEPAL is a ride sharing service that allows customers to request rides from drivers through an online platform. Customer can use the platform to share rides with other users, and drivers can also offer services such as food delivery, package delivery and courier delivery. The app connects customers and drivers, allowing them to communicate and arrange rides or deliveries through the platform.

1.1 Aim and Objectives

The aim of **PATHAO NEPAL** is likely to provide convenient and cost-effective transportation option for users and to connect driver with potential customers. The other aim of this **PATHAO NEPAL COMPANY** is to: -

- Provide high-quality services to its customers.
- Expanding its business.
- Building a strong brand and reputation in the industry.
- Supporting the local community and promoting social and environmental responsibility.
- Increasing profitability and shareholder value.

Some objectives for **PATHAO NEPAL** include: -

- Improving customer satisfaction by consistently providing high-quality service and addressing any issues or concerns that arise.
- Expanding the company's reach by increasing the number of customers served or entering new markets.
- Establishing the company as a leader in the ride-sharing industry by building a strong brand and reputation through excellent customer service and innovate features.
- Making a positive impact on the local community through initiatives such as supporting local business or organizations, or implementing environmentally-friendly practices.
- Increasing profitability and shareholder value through strategies such as optimizing pricing, streamlining operations, or expanding the company's offerings.

1.2 Current Business Activities and Operations

PATHAO NEPAL is a ride sharing service that allows customers to request rides from drivers through an online platform. Customer can use the platform to share rides with other users, and drivers can also offer services such as food delivery, package delivery and courier delivery. The app connects customers and drivers, allowing them to communicate and arrange rides or deliveries through the platform.

This company is expanding its business to Nepal. They have implemented a new rule where drivers receive a 25% bonus on top of their pay for each service they provide. The remaining 75% goes to the owner. This company also has different discounts for different categories of customers. Normal Customers do not receive any discount for different categories of customers. Normal customers do not receive any discounts, staff receive a 20% discount, and VIPs receive a 50% discount. In addition to ride sharing, the company has also introduced service like courier delivery, package delivery, and food delivery.

1.2.1 Business Rules

PATHAO Nepal has certain rules to perform its business rules. They can be listed as:

- A driver may drive many vehicles, but each vehicle and a service is used by one driver at a time.
- A driver writes a single invoice for each service he provides.
- Once the costumer books the service, they cannot cancel the service.
- Service ticket is issued once the costumer books the vehicle and the service and will include details like driver name, type of service, total charge, estimated duration of destination.
- The cost of the vehicle and duration can vary depending on its type. For example, the cost of riding the motorcycle service can be cheaper than riding the car.

1.2.2 Business Assumption

Some business assumption I made while creating database are:-

- Service_Ticket_ID and Invoice_ID are two separate.

- If a customer uses a service, their reward points are increased.
  Formula to calculate reward points:
  RP = Service_Used * 0.85
  Here service_used is total service used by customer and total service_ID taken by customer.

- Customer cannot use more than one service at the same time. If they need to use multiple services, they must obtain additional Service_Ticket_IDs.

- Driver receives a salary per month, and a bonus is added on the top of this salary.

- To calculate the discount amount for a customer, the following formula is used
  Discount Amount *= (Duration * Vehicle Rate) * Customer_Discount*

- Invoice has multiple instances for single service. For example, Food delivery can have multiple invoice_Id.

- In attribute Destination there are two locations first one is starting point and second one is ending point

2. Entity Relationship Diagram (ERD)

An entity relationship diagram (ERD), also known as entity relationship model, is a graphical representation among people, object, places, concepts or events within an information technology (IT) system. An ERD uses data modelling techniques that can help define business processes and serve as the foundation for a relational database (Jacqueline, 2022).

Importance of ERD:

- ERD diagram gives a visual picture of a database's structure. This simplifies the data and relationships between distinct entities for database designers and developers
- ERDs may be used to provide a graphical representation of a database's design, making easier to verify for faults and inconsistencies.
- ERDs may be utilized to refine database needs. They give a technique to ensure that all criteria have been recorded and that there are no conflicts between them.
- ERDs can be used to document a database's structure. This improves future developers' understanding of the database architecture and structure easier.
- ERDs may be used to produce SQL code for table and relationship creation. This reduces the amount of time and effort required to create the database.
- ERDs may be used to validate database design. They can assist in identifying any issues with the database's architecture or structure. 5. ERDs may be used to produce SQL code for table and relationship creation. This reduces the amount of time and effort required to create the database.
- ERDs may be used to validate database design. They can assist in identifying any issues with the database's architecture or structure.

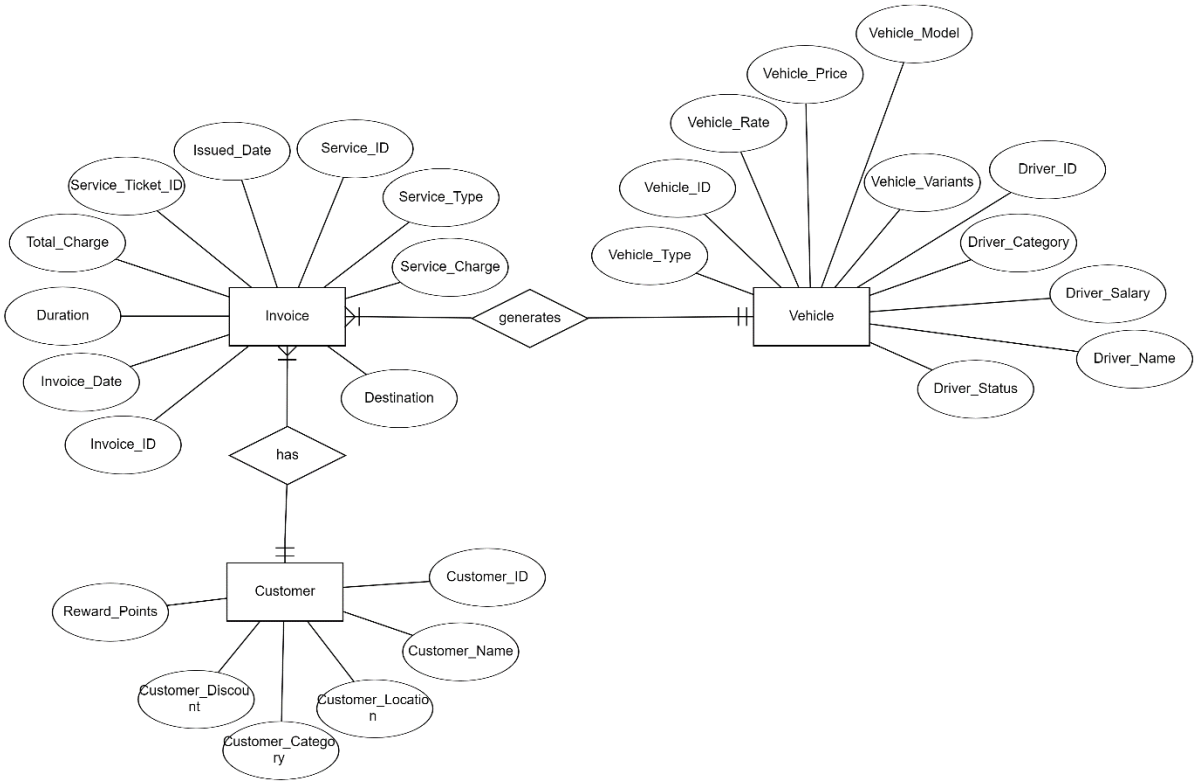


Figure 1: Initial Entity Relationship Diagram

A chasm trap occurs when two many-to-one relationships converge on a single table in a database design and a fan trap occurs when two many-to-one relationships follow one another in a primary-detail relationship. This can lead to difficulties in querying the data, as it may be difficult to determine which many-to-one relationship is the primary relationship and which is the detail But in by Initial ERD there is not the presence of both trap.

But in my attributes of intial erd there is presence of partial dependency and transitive dependency. Because of this, anamoly occurs in my erd such as – insertion, deletion and updation.

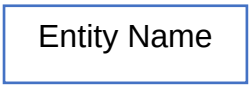
3. Identification of Entities and Attributes

3.1 Enities and Attributes

An entity is a distinct and separate objects that can be identified and recognized as a single unit. It can refer to a person, organization, system, piece of data, or a component of a system, piece of a data, or a component of a system that is considered important in its own right (techopedia, 2014).

Entity in my Initial Entity Relationship Diagram(ERD) :-

- Invoice
- Vehicle
- Customer

Symbol used for Entities :- 

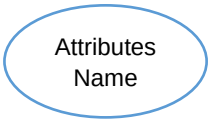
An attribute is a feature or quality that belongs to an entity. An entity may have muliple attributes, and one of them is typically designated as the primary key. In a Entity-Relation model, attributes are represented using an elliptical shape.

Attributes that are in my Initial Enitiy Relationship Diagram(ERD) :-

Attributes of Customer :- *Customer_ID, Customer_Name, Customer_Discount, Customer_Location, Customer_Category, Reward_Points*

Attributes of Invoice :- *Invoice_ID, Invoice_Date, Duration, Destination, Total_Charge, Service_Ticket_ID, Issued_Date, Service_ID, Service_Type, Service_Charge*

Attributes of Vehicle :- *Vehicle_ID, Vehicle_Rate, Vehicle_Price, Vehicle_Model, Vehicle_Variants, Vehicle_Type, Driver_ID, Driver_Name, Driver_Category, Driver_Status, Driver_Salary*

Symbol used for attributes: - 

3.1.1 Customer

Attributes	Datatype	Constraints	Description
Customer_ID	int	Primary key	This attribute stores unique customer identity.
Customer_Name	Varchar2(20)	Not Null	This attribute stores customer names.
Customer_Location	Varchar2(20)	N/A	This attribute stores customer location.
Reward_Point	Decimal (5,2)	Not Null	This attribute stores reward point of customer.
Customer_Discount	Varchar2(20)	Unique, Not Null	This attribute stores the discount percentage according to customer category.
Customer_Category	Varchar2(20)	PK, FK(Customer)	This attribute stores customer category like: - VIP, staff, customer member.

Table 1: Screenshot of Customer Table

3.1.2 Invoice

Attributes	Datatype	Constraints	Description
Invoice_ID	Varchar2(10)	Primary Key, Not Null	This field stores unique invoice id.
Invoice_Date	Date	Not Null	This field stores invoice date.
Duration	varchar2(5)	N/A	This field calculate duration.
Destination	Varchar2(20)	Not Null	This field stores the destination of customer.
Total_Charge	Varchar2(15)	Not Null	This field calculate the total charge of Customer.
Service_Ticket_ID	Int Primary Key	PK, FK(Invoice)	This field stores the unique Id of Service Ticket ID.
Issued_Date	Date	Date	This field stores the issued date of customer.
Service_ID	Varchar2(20)	PK, FK (invoice)	This field stores the unique id of service used.
Service_Type	Varchar2(20)	Unique	This field takes the unique type of customer.
Service_Charge	Int	Not Null	This field stores the service charge of customer.

Table 2: Screenshot of Invoice Table

3.1.3 Vehicle

Attributes	Datatype	Constraints	Description
Vehicle_ID	Varchar2(10)	Primary Key	This attribute stores the different vehicle id.
Vehicle_Rate	Decimal (5,2)	Not Null	This attribute stores the vehicle rate.
Vehicle_Price	int	N/A	This attribute stores the vehicle price.
Vehicle_Model	Varchar2(20)	Not Null	This attribute stores the vehicle model.
Vehicle_Variants	Varchar2(25)	N/A	This attribute stores the vehicle variants.
Vehicle_Type	Varchar2(20)	Not Null	This attribute stores the vehicle type.
Driver_ID	int	PK, FK(Vehicle)	This attribute stores the unique driver id.
Driver_Name	Varchar2(25)	Varchar2(25)	This field stores the driver's name.
Driver_Category	Varchar2(15)	Not Null	This attribute stores the driver category.
Driver_Status	Varchar2(13)	Not Null	This attribute stores the driver status.
Driver_Salary	Decimal (7,2)	Not Null	This attribute stores the driver_salary

Table 3: Screenshot of Vehicle Table.

4. Normalization

Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically (Richard, 2022).

4.1 Un-Normalized Form (UNF)

Un-Normalized Form (UNF) model stores data in a single table with multiple columns that contains repeating information i.e. (group and data). This causes data redundancy and inconsistency. For example, if a customer changes their address, multiple entries will need to update, resulting in potential data integrity issues.

Here in my initial ERD, One Vehicle can generate many Invoice and many invoice can have one customer. I started from Vehicle, so vehicle is my repeating data and Invoice and Customer are repeating group for Vehicle.

Vehicle ***(****Vehicle_ID, Vehicle_Rate, Vehicle_Price, Vehicle_Model, Vehicle_Variants, Vehicle_Type, Driver_ID, Driver_Name, Driver_Category, Driver_Status, Driver_Salary,* ***{*** *Invoice_ID, Invoice_Date, Duration, Destination ,Total_Charge, Service_Ticket_ID, Issued_Date, Service_ID, Service_Type, Service_Charge, Customer_ID, Customer_Name, Customer_Discount , Customer_Location, Customer_Category, Reward_Points , }****)***

4.2 First Normal Form (1NF)

After the first rule (UNF), First Normal Form (1NF) is applied. From UNF we are clear about Repeating Group and Repeating Data. To qualify as 1NF we have to separate Repeating Group and Repeating Data, so Vehicle and Invoice is separated from UNF, Also we have to provide Primary key of tables Vehicle is Vehicle_ID and Vehicle_ID goes on Invoice as Foreign key and Invoice has own primary key which is Invoice_ID.

Rules to achieve First Normalization Form are: -

- Remove repeating groups by separating them into separate tables.
- Each record must be unique and distinct.
- A primary key must be assigned to each tables. The primary key is a unique identifier that is used to uniquely identify each record in the tables.
- Each column in a table should have its own specific name.

Vehicle 1 ***(****Vehicle_ID(****PK****), Vehicle_Rate, Vehicle_Price, Vehicle_Model, Vehicle_Variants, Vehicle_Type, Driver_ID, Driver_Name, Driver_Category, Driver_Status, Driver_Salary****)***

Invoice 1 (*Invoice_ID(****PK****), Invoice_Date, Duration, Destination, Total_Charge, Service_Ticket_ID, Issued_Date, Service_ID, Service_Type, Service_Charge, Customer_ID, Customer_Name, Customer_Discount, Customer_Location, Customer_Category, Reward_Points, Vehicle_ID(****FK****))***

4.3 Second Normal Form (2NF)

The table must be in First Normal Form to achieve Second Normal Form (2NF).

Checking Partial Dependency for Vehicle1 Table

There should be two keys to check Second Normal Form but in Vehicle1 there is only one key so this table is already in Second Normal From.

Vehicle1(*Vehicle_ID(**PK**), Vehicle_Rate, Vehicle_Price, Vehicle_Model, Vehicle_Variants, Vehicle_Type, Driver_ID, Driver_Name, Driver_Category, Driver_Status, Driver_Salary,***)**

Checking Partial Dependency for Invoice1 Table

In table Invoice we can clearly see two keys i.e.(Invoice_ID, Vehicle_ID), so that we have to check partial dependency. Here Invoice_ID gives all the attributes of Invoice table expect Vehicle_ID.

Invoice_ID → *Invoice_Date, Duration, Destination, Total_Charge, Service_Ticket_ID, Issued_Date, Service_ID, Service_Type, Service_Charge, Customer_ID, Customer_Name, Customer_Discount, Customer_Location, Customer_Category, Reward_Points*

Vehicle_ID do not give any attributes because all attributes are given by Invoice_ID.

Vehicle_ID → *XXX*

The attributes Invoice_ID and Vehicle_ID do not provide any attributes and information on their own, but they do form a table called Vehicle_Invoice_Generator.

Invoice_ID, Vehicle_ID →

Final Table in 2NF for Invoice Table

Final table that is formed in Second Normal Form: -

Vehicle2 ***(****Vehicle_ID(****PK****), Vehicle_Rate, Vehicle_Price, Vehicle_Model, Vehicle_Variants, Vehicle_Type, Driver_ID, Driver_Name, Driver_Category, Driver_Status, Driver_Salary,****)***

Invoice2 ***(****Invoice_ID(****PK****), Invoice_Date, Duration, Destination, Total_Charge, Service_Ticket_ID, Issued_Date, Service_ID, Service_Type, Service_Charge, Customer_ID, Customer_Name, Customer_Discount, Customer_Location, Customer_Category, Reward_Points)*

Vehicle_Invoice_Generator2 (Invoice_ID(**FK**), Vehicle_ID(**FK**)**)**

4.4 Third Normalization Form(3NF)

Checking transitive dependency for Vehicle

In vehicle2 table, there is a transitive dependency since, the non-key attributes 'Driver_Name, Driver_Category, Driver_Status, Driver_Salary' is dependent on another non-key attribute 'Driver_ID'. So we need to remove transitive dependency. This can be done by placing in separate table name Driver3 with Driver_ID as primary key.

Vehicle_ID → Driver_ID(PK) ----- > Driver_Name, Driver_Category, Driver_Status, Driver_Salary

Here Driver_ID is separated

Driver_ID(PK) *--> Driver_Name, Driver_Category, Driver_Status, Driver_Salary*

After Driver3 is separated, the remaining attributes is depend on only Vehicle_ID and Driver_ID is left as foreign key.

Vehicle_ID → *Vehicle_Rate, Vehicle_Price, Vehicle_Model, Vehicle_Variants, Vehicle_Type, Driver_ID (**FK**)*

Final table in 3 NF for Vehicle

Vehicle3 ***(****Vehicle_ID (****PK****), Vehicle_Rate, Vehicle_Price, Vehicle_Model, Vehicle_Variants, Vehicle_Type, Driver_ID (****FK****)****)***

Driver3 **(***Driver_ID(****PK****), Driver_Name, Driver_Category, Driver_Status, Driver_Salary****)***

Transitive dependency is removed from table Vehicle.

Checking transitive dependency for Invoice Table

In Invoice2 table there is transitive dependency since, non-key attributes **"Service_Type, Service_Charge"** is dependent on another non-key attribute **Service_ID** and Service Table is separated from this with **Service_ID** as primary key.

Invoice_ID → Service_ID ---- > *Service_Type, Service_Charge*

Another non-key attribute **Issued_date** is dependent on another non-key attributes **Service_Ticket_ID** and **Service_Ticket_Issued** is separate from this with **Service_Ticket_ID** as primary key.

Invoice_ID → Service_Ticket_ID ---- > Issued_Date

Another non-key attribute **"Customer_Name, Customer_Location, Reward_Points"** is dependent on another non-key attribute **Customer_ID** and **Customer** table is separate from this with **Customer_ID** as primary key.

Invoice_ID → Customer_ID ---- > Customer_Name, Customer_Location, Reward_Points

In table **Customer** there is also transitive because non-key attribute **Customer_Discount** is dependent on **Customer_Category** and **Customer_Type** table is formed with primary key **Customer_Category**.

Customer_ID → Customer_Category---- > Customer_Discount

Remaining attribute of **Invoice** falls under table invoice with **Invoice_ID** as primary key and other table formed with key **Invoice_ID**, falls as a foreign key.

Invoice_ID → Invoice_Date, Duration, Destination, Total_Charge, Service_ID(FK), Service_Ticket_ID(FK), Customer_ID(FK), Vehicle_ID(FK)

Final Table in 3NF for Invoice Table

Service3 (Service_ID (PK), Service_Type, Service_Charge)

Service_Ticket_Issued3 (Service_Ticket_ID (PK), Issued_Date)

Customer3 (Customer_ID (PK), Customer_Name, Customer_Location, Reward_Points, Customer_Category (**FK**))

Customer_Category3 (Customer_Category (**PK**), Customer_Discount)

Invoice3 (Invoice_ID(**PK**), Invoice_Date, Duration, Destination, Total_Charge, Service_ID(**FK**), Service_Ticket_ID(FK), Customer_ID(FK)**)**

Vehicle_Invoice_Generator3 (Invoice_ID(**FK**), Vehicle_ID(**FK**)**)**

Final Table in Third Normal Form (3NF)

Vehicle ***(****Vehicle_ID (****PK****), Vehicle_Rate, Vehicle_Price, Vehicle_Model, Vehicle_Variants, Vehicle_Type, Driver_ID (****FK****)****)***

Driver ***(****Driver_ID(****PK****), Driver_Name, Driver_Category, Driver_Status, Driver_Salary****)***

Service *(Service_ID (PK), Service_Type, Service_Charge)*

Service_Ticket_Issued *(Service_Ticket_ID (PK), Issued_Date)*

Customer *(Customer_ID (PK), Customer_Name, Customer_Location, Reward_Points, Customer_Category (****FK****))*

Customer_Type *(Customer_Category (****PK****), Customer_Discount)*

Invoice ***(****Invoice_ID(****PK****), Invoice_Date, Duration, Destination, Total_Charge, Service_ID(****FK****), Service_Ticket_ID(FK), Customer_ID(FK)****)***

Vehicle_Invoice_Generator ***(****Invoice_ID(****FK****), Vehicle_ID(****FK****))***

5. Final Entity Relationship Diagram

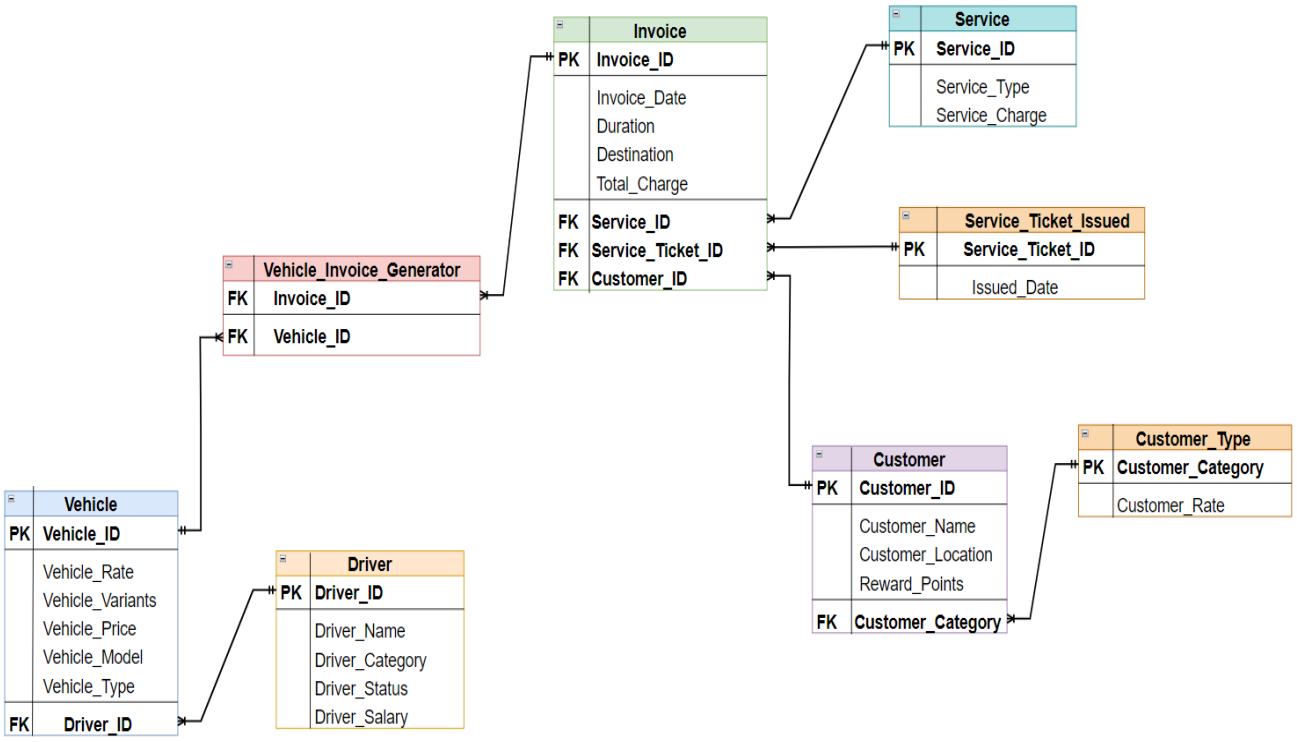


Figure 2: Final Entity Relationship Diagram

The final ER-Diagram shown above is the result of the normalization process. It consists of eight tables, each with various attributes that hold different type of data. There are also four bridge entities, which server as connections between the other tables and help establish proper relationship between them.

In final ERD logical model is used, rather than a physical one, because it represents the logical structure of the database and the relationships between the entities it contains. This means that the final ERD focuses on how the entities are connected, rather than specific hardware or software that will be used to implement the database.

Using a logical diagram for the final ERD offers several benefits. Firstly, it allows for greater flexibility in the design process, as the logical structure can be translated into various physical implementations. Additionally, it provides a clear and concise visual representation of the relationships between entities, which can aid in communication and documentation. Overall, the use of a logical ERD facilitates a more efficient and effective database design process.

After normalizing the entities and attributes form their UNF to 3NF, eight tables are formed they are: - **Vehicle, Driver, Service_Ticket_Issued, Customer, Customer_Type, Service, Invoice, Vehicle_Invoice_Generator**. And the normalized entity relationship diagram is finally constructed.

6. Implementation

In SQL PLUS, we implement the data by creating tables based on the final ER-Diagram that we obtain after the normalization process. We create a user called "coursework_suman" and grant it permissions. Then, we connect to the new user. After that, we begin the process of creating tables. We use the CREATE TABLE command to create tables. We use datatypes such as varchar2, int, decimal, and not null while creating the tables.

Creating and Granting User:

SQL> connect system/suman

Connected.

SQL> create user coursework identified by suman;

User created.

SQL> grant connect, resource to coursework;

Grant succeeded.

SQL> connect coursework/suman

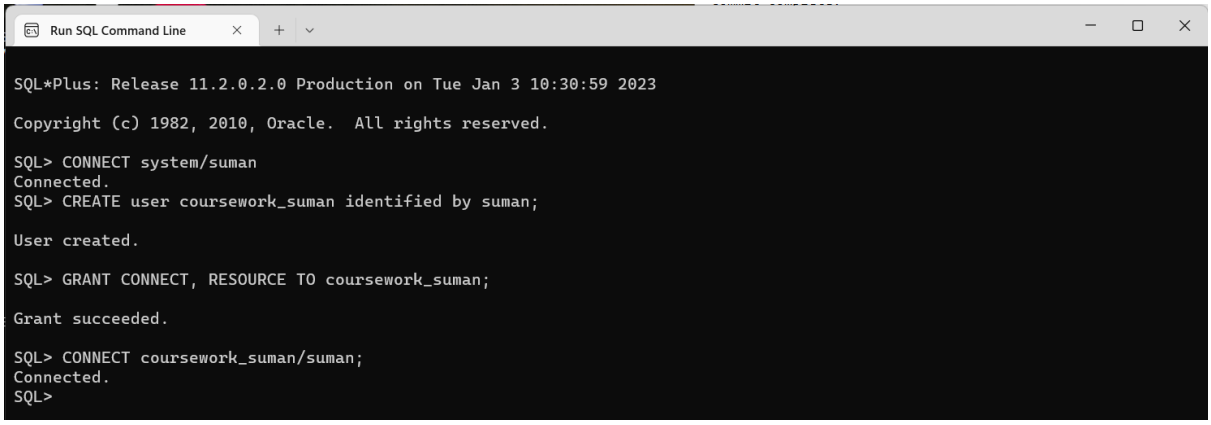


Figure 3: Creating User, Connecting and Granting to User

Creating Driver Table

```
SQL> Create table Driver(
2  Driver_ID int primary key,
3  Driver_Name varchar2(25) not null,
4  Driver_Category varchar2(15) not null,
5  Driver_Status varchar2(13) not null,
6  Driver_Salary decimal(7,2) not null );
```

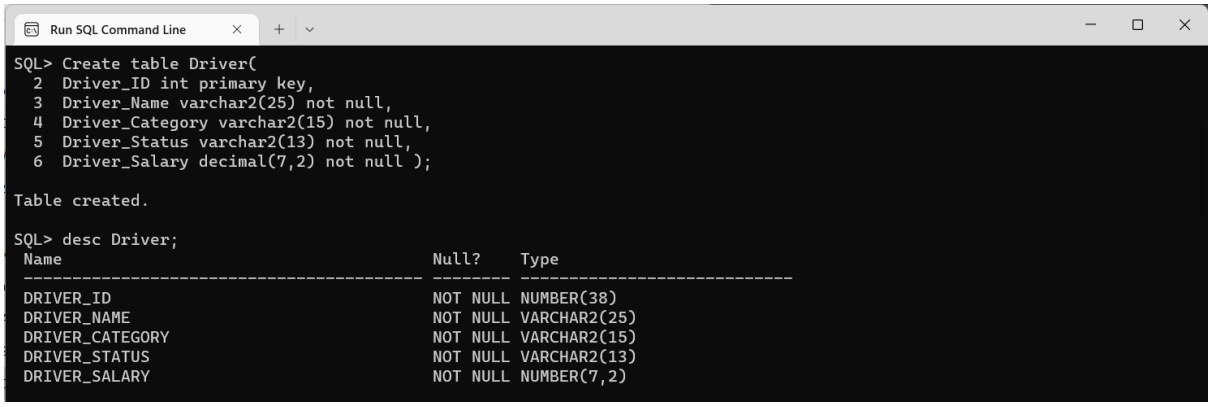


Figure 4: Screenshot of Table Driver Creation.

Data implementation of Driver Table

```
SQL> Insert into Driver Values (101, 'James Smith', 'Full Time', 'Riding', 28000.15);

1 row created.

SQL> Insert into Driver Values (102, 'Sarah Johnson', 'Part Time', 'Riding',
     15000.15);

1 row created.

SQL> Insert into Driver Values (103, 'Maria Williams', 'Full Time', 'Active',
     99910.15);

1 row created.

SQL> Insert into Driver Values (104, 'Andrew Browm', 'Full Time', 'In Active',
     50500.15);

1 row created.

SQL> Insert into Driver Values (105, 'Emily Miller', 'Full Time', 'Riding', 45000.15);

1 row created.
```

```
SQL> Insert into Driver Values (106, 'Mathew Davies', 'Full Time', 'Active', 14000.15);

1 row created.

SQL> Insert into Driver Values (107, 'Jessica Andreson', 'Part Time', 'Active', 25000.15);

SQL> Insert into Driver Values (108, 'Taylor Wilson', 'Part Time', 'Active', 25000.15);

1 row created.

SQL> Insert into Driver Values (109, 'Ryan Talor', 'Full Time', 'IN Active', 70000.15);

1 row created.
```

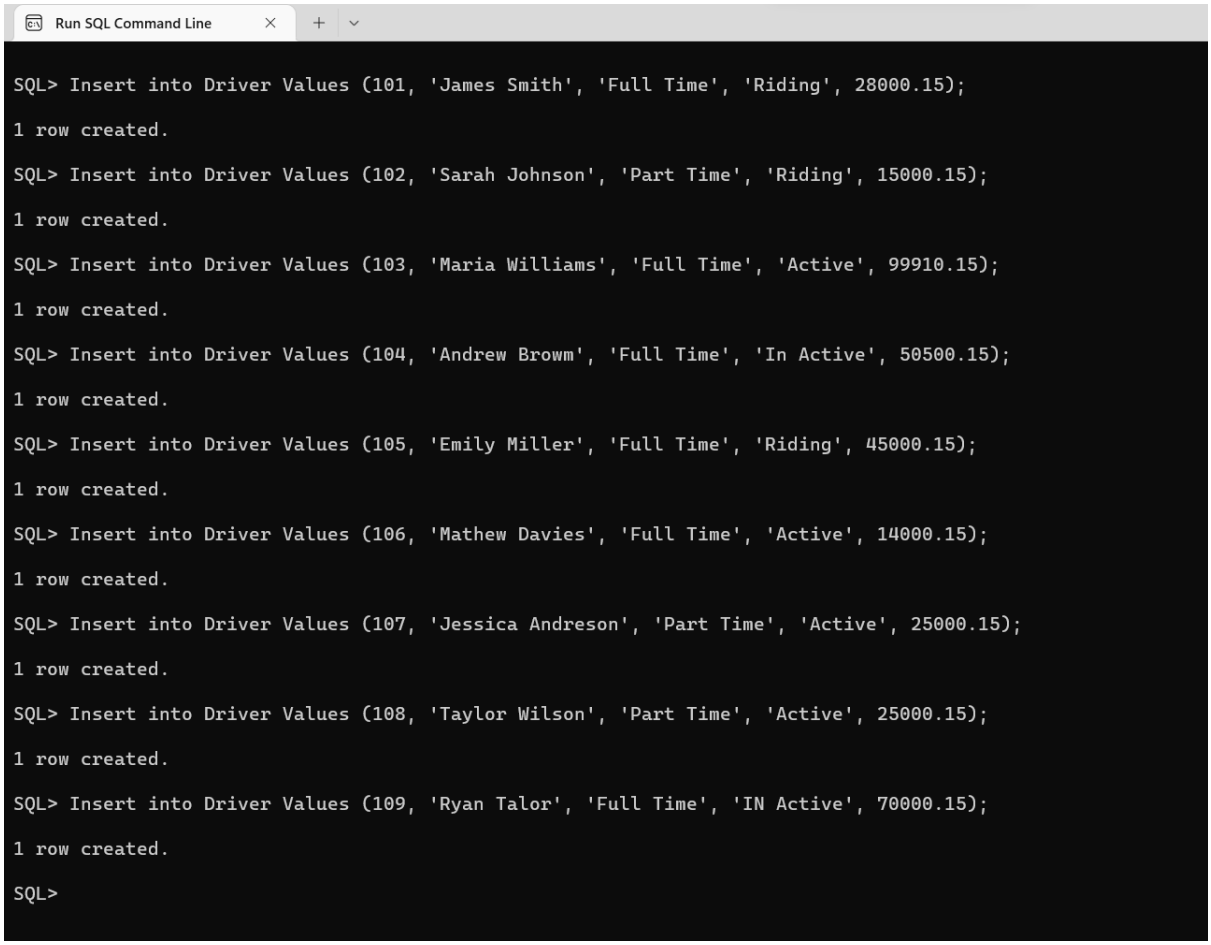
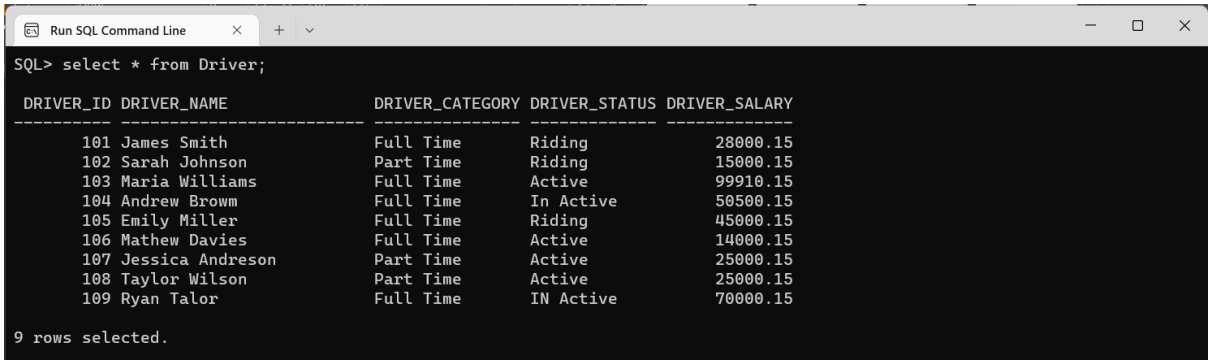


Figure 5: Screenshot of data insertion on table Driver.

SQL> select * from Driver;



```
Run SQL Command Line
SQL> select * from Driver;

 DRIVER_ID DRIVER_NAME               DRIVER_CATEGORY DRIVER_STATUS DRIVER_SALARY
---------- ------------------------- --------------- ------------- -------------
       101 James Smith               Full Time       Riding             28000.15
       102 Sarah Johnson             Part Time       Riding             15000.15
       103 Maria Williams            Full Time       Active             99910.15
       104 Andrew Browm              Full Time       In Active          50500.15
       105 Emily Miller              Full Time       Riding             45000.15
       106 Mathew Davies             Full Time       Active             14000.15
       107 Jessica Andreson          Part Time       Active             25000.15
       108 Taylor Wilson             Part Time       Active             25000.15
       109 Ryan Talor                Full Time       IN Active          70000.15

9 rows selected.
```

Figure 6: Screenshot of data of Driver.

Creating Vehicle Table

SQL> Create table Vehicle(

2 Vehicle_ID varchar2(10) primary key,

3 Vehicle_Rate decimal(5,2) not null,

4 Vehicle_Variants varchar2(25),

5 Vehicle_Price int,

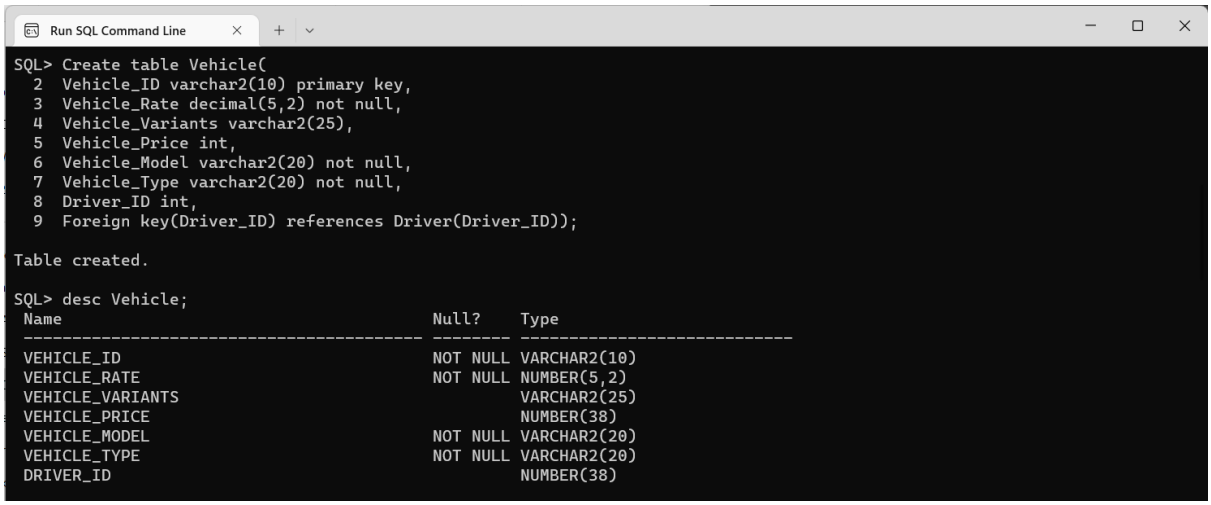
6 Vehicle_Model varchar2(20) not null,

7 Vehicle_Type varchar2(20) not null,

8 Driver_ID int,

9 Foreign key(Driver_ID) references Driver(Driver_ID));

Table created.



```
Run SQL Command Line
SQL> Create table Vehicle(
  2  Vehicle_ID varchar2(10) primary key,
  3  Vehicle_Rate decimal(5,2) not null,
  4  Vehicle_Variants varchar2(25),
  5  Vehicle_Price int,
  6  Vehicle_Model varchar2(20) not null,
  7  Vehicle_Type varchar2(20) not null,
  8  Driver_ID int,
  9  Foreign key(Driver_ID) references Driver(Driver_ID));

Table created.

SQL> desc Vehicle;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VEHICLE_ID                                NOT NULL VARCHAR2(10)
 VEHICLE_RATE                              NOT NULL NUMBER(5,2)
 VEHICLE_VARIANTS                                   VARCHAR2(25)
 VEHICLE_PRICE                                      NUMBER(38)
 VEHICLE_MODEL                             NOT NULL VARCHAR2(20)
 VEHICLE_TYPE                              NOT NULL VARCHAR2(20)
 DRIVER_ID                                          NUMBER(38)
```

Figure 7: Screenshot of Creation of table Vehicle.

Data implementation of Vehicle Table

```
SQL> insert into Vehicle values ('S2 BA 1241', 121.60, 'Bike', 250000, 'Splender',
     'Petrol', 101);

1 row created.

SQL> insert into Vehicle values ('S5 RA 2241', 150.70, 'Bike', 500000, 'Duke 250',
    'Petrol', 102);

1 row created.

SQL> insert into Vehicle values ('S6 SU 2025', 500.88, 'Cab', 2500000, 'Nissal Leaf',
     'Electric', 103);

1 row created.

SQL> insert into Vehicle values ('S1 JK 2121', 600.88, 'Cab', 3000000, 'Ford',
    'Petrol', 104);

1 row created.

SQL> insert into Vehicle values ('S3 BA 7771', 150.77, 'Scooter', 3000000, 'NUI',
     'Electric', 105);

1 row created.

SQL> insert into Vehicle values ('S3 BA 5051', 300.50, 'Bike', 300000, 'Pulsar 200',
     'Petrol', 106);

1 row created.

SQL> insert into Vehicle values ('S5 RA 2012', 130.50, 'Scooter', 5500000,
     'Renault Zoe', 'Petrol', 107);

1 row created.

SQL> insert into Vehicle values ('S2 BA 7175', 999.50, 'Cab', 3500000, 'Ford',
     'Disel', 108);

1 row created.

SQL> insert into Vehicle values ('S2 BA 8018', 850.00, 'Bike', 900000, 'Benilli',
    'Petrol', 109);

1 row created.

SQL> insert into Vehicle values ('S2 BA 5500', 750.00, 'Cab', 3000000, 'Zoe 500',
     'Electric', 101);

1 row created.
```

```
SQL> insert into Vehicle values ('S3 BA 5031', 350.00, 'Bike', 700000, 'Benelli',
        'Petrol', 103);

1 row created.
```

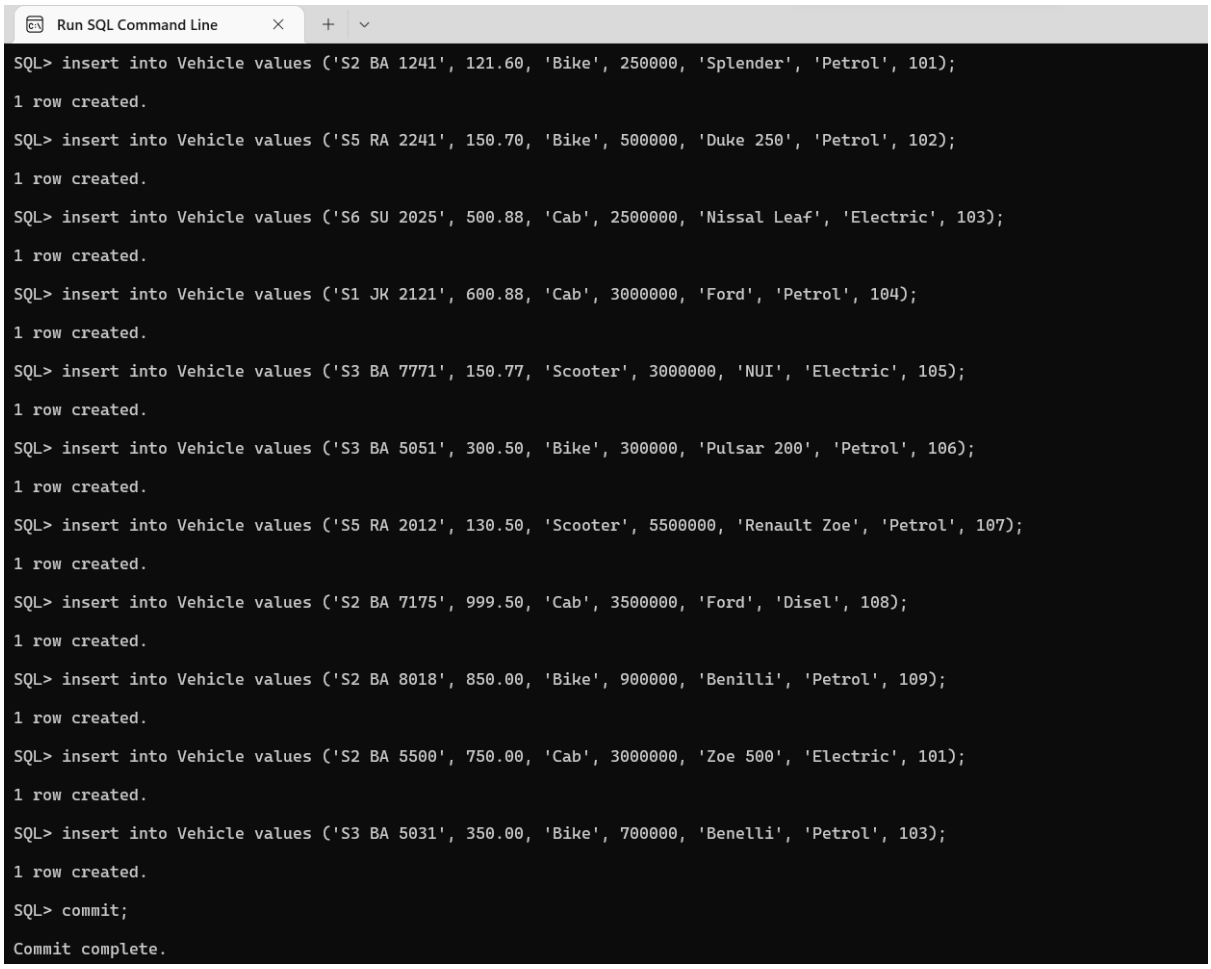


Figure 8: Screenshot of data insertion into Vehicle.

SQL > Select * from Vehicle;

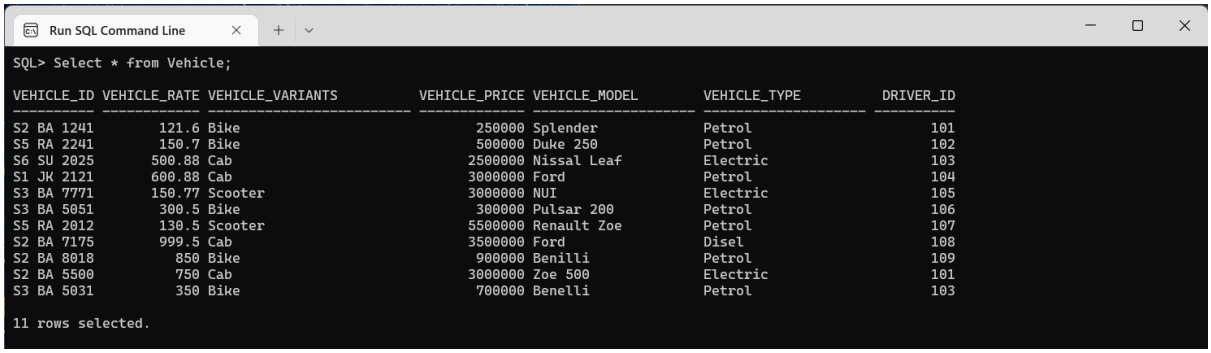


Figure 9: Screenshot of data of Vehicle.

Creating table Service

```
SQL> Create table Service(
  2  Service_ID varchar2(10) primary key,
  3  Service_Type varchar2(20) unique,
  4  Service_Charge int not null);
```

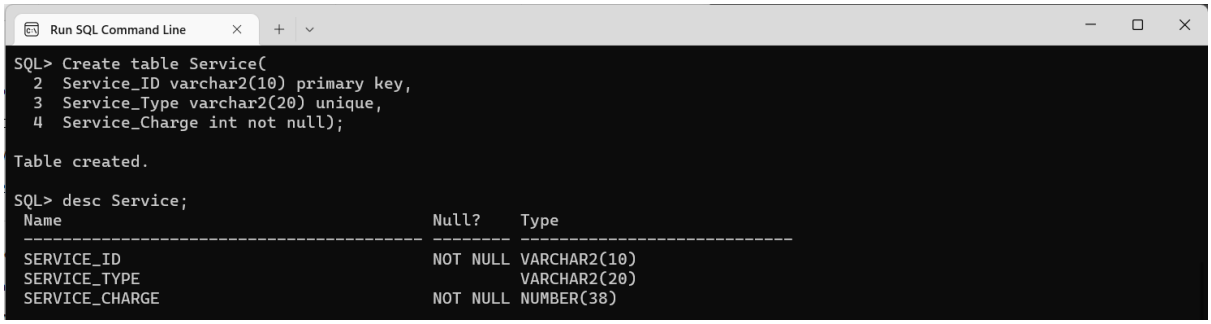


Figure 10: Screenshot of creating table Service.

Data Implementation of Service Table

```
SQL> Insert into Service values( 'RS 101', 'Ride Sharing', 200);

1 row created.

SQL> Insert into Service values( 'FD 102', 'Food Delivery', 300);

1 row created.

SQL> Insert into Service values( 'PD 103', 'Package Delivery', 360);

1 row created.

SQL> Insert into Service values( 'CD 104', 'Courier Delivery', 250);

1 row created.
```

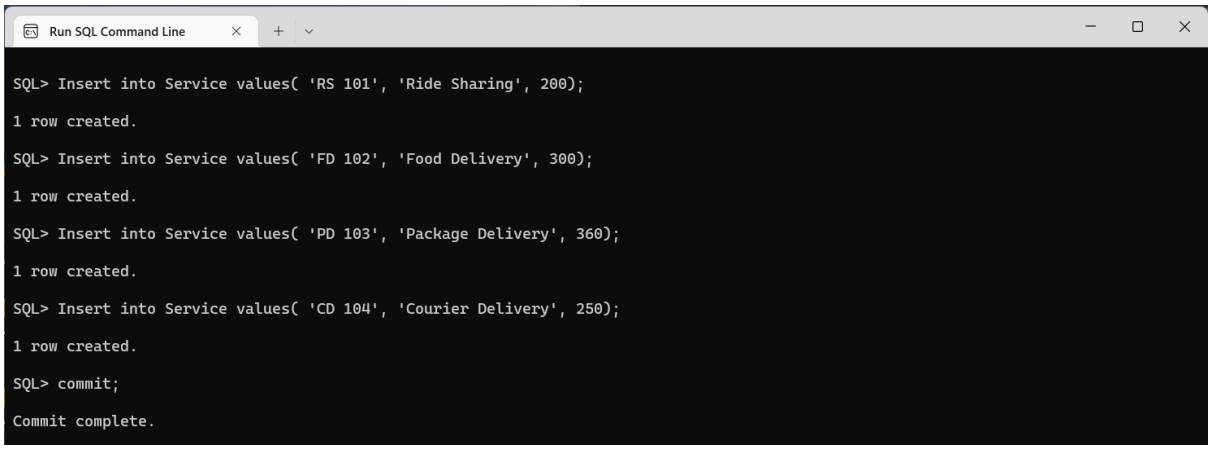


Figure 11: Screenshot of Data Insertion on Service Table.

SQL> select * from Service;

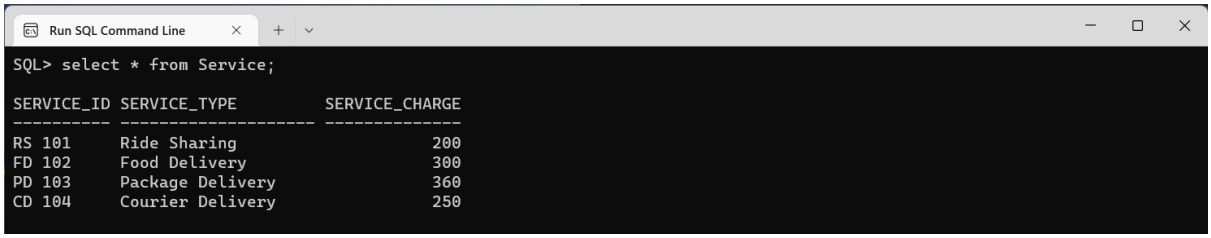


Figure 12: Screenshot of data of Service

Creating table Service_Ticket_Issued

SQL> Create table Service_Ticket_Issued(

2 Service_Ticket_ID int primary key,

3 Issued_Date date);

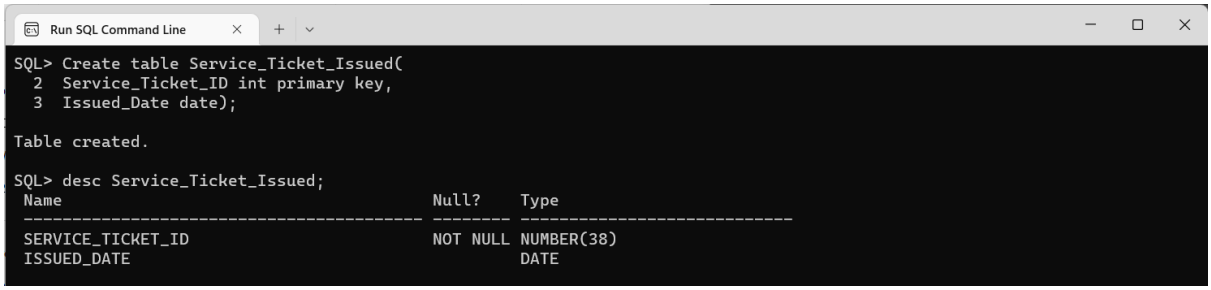


Figure 13: Screenshot of creation of table Service_Ticket_Issued.

Data Implementation of Service_Ticket_Issued

SQL> Insert into Service_Ticket_Issued values (2001,'03-FEB-2022');

1 row created.

SQL> Insert into Service_Ticket_Issued values (2002,'04-FEB-2022');

1 row created.

SQL> Insert into Service_Ticket_Issued values (2003,'05-MAR-2022');

1 row created.

SQL> Insert into Service_Ticket_Issued values (2004,'15-MAR-2022');

1 row created.

SQL> Insert into Service_Ticket_Issued values (2005,'15-Feb-2022');

1 row created.

```
SQL> Insert into Service_Ticket_Issued values (2006,'16-Feb-2022');
1 row created.
SQL> Insert into Service_Ticket_Issued values (2007,'01-APR-2022');
1 row created.
SQL> Insert into Service_Ticket_Issued values (2008,'02-APR-2022');
1 row created.
SQL> Insert into Service_Ticket_Issued values (2009,'03-APR-2022');
1 row created.
SQL> Insert into Service_Ticket_Issued values (2010,'04-APR-2022');
1 row created.
SQL> Insert into Service_Ticket_Issued values (2011,'01-JAN-2022');
1 row created.
SQL> Insert into Service_Ticket_Issued values (2012,'02-JAN-2022');
1 row created.
SQL> Insert into Service_Ticket_Issued values (2013,'03-JAN-2022');
1 row created.
SQL> Insert into Service_Ticket_Issued values (2014,'04-JAN-2022');
1 row created.
SQL> Insert into Service_Ticket_Issued values (2015,'05-JAN-2022');
1 row created.
```

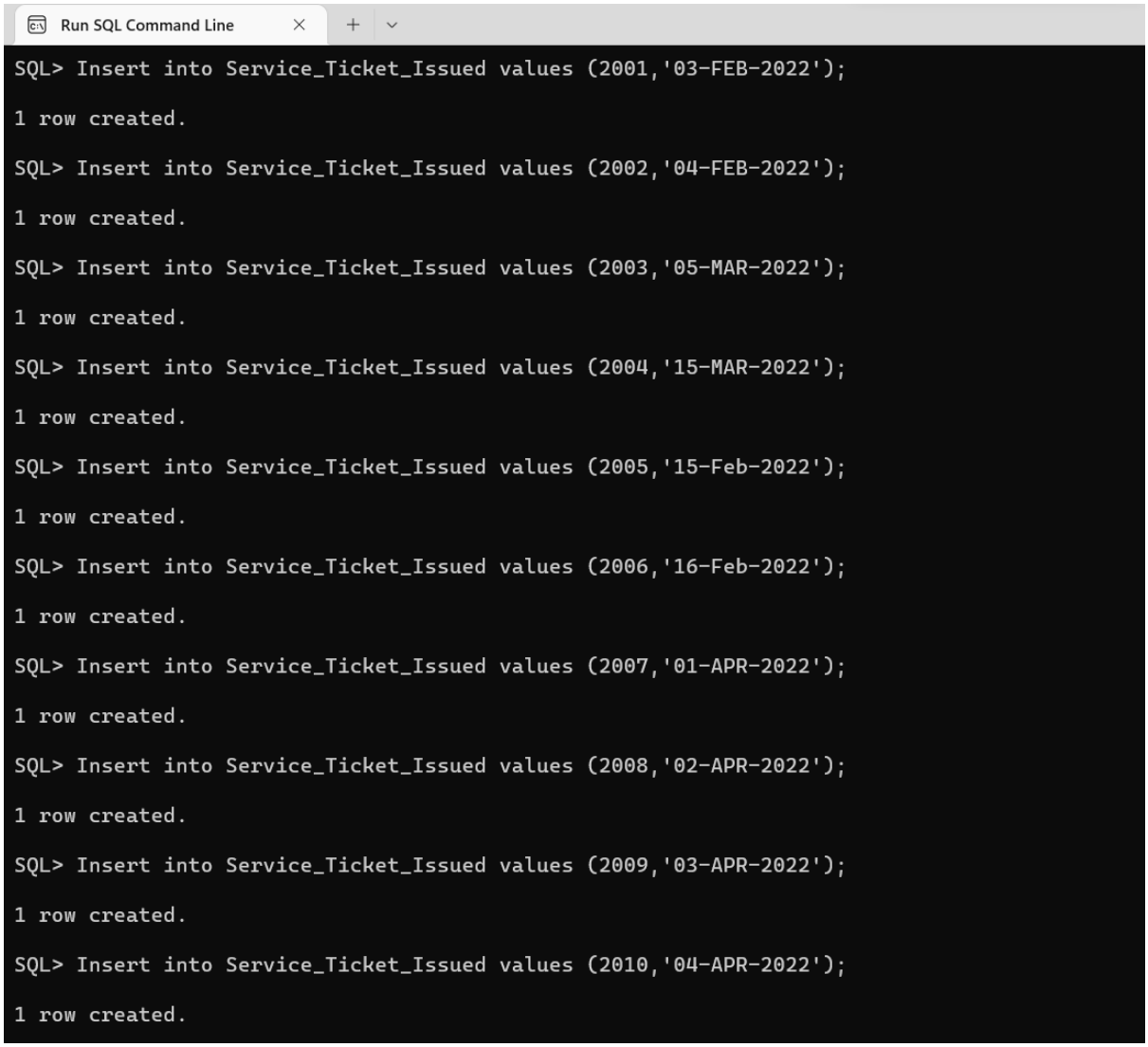


Figure 14: Screenshot of data insertion into Service_Ticket_Issued Part(A).

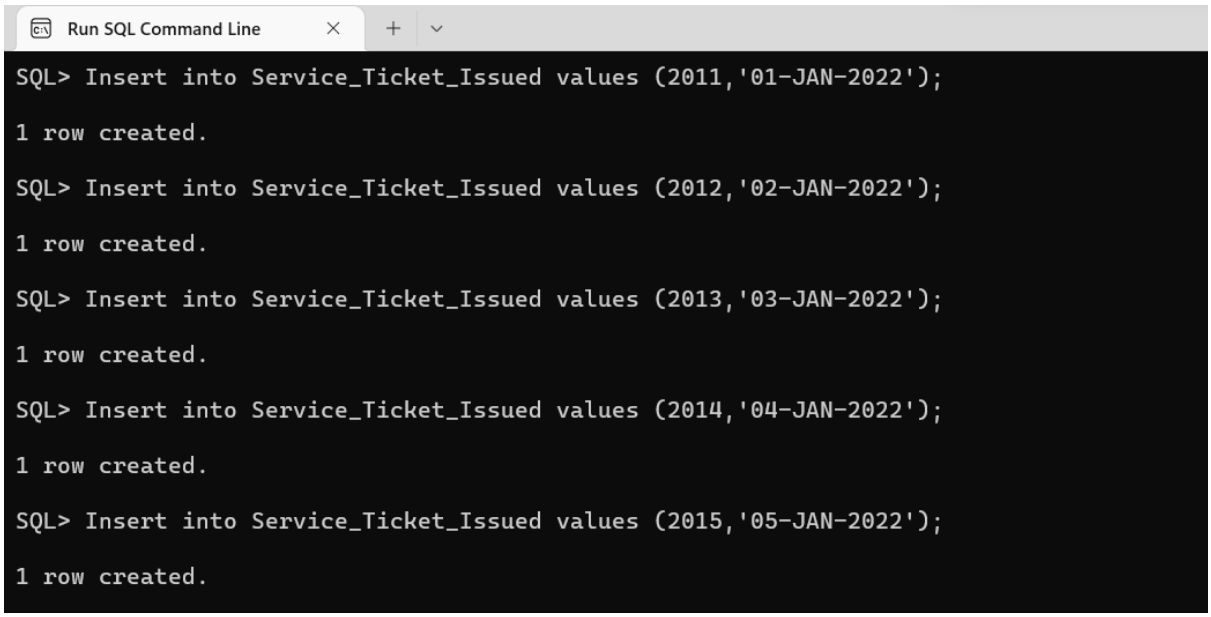


Figure 15:Screenshot of data insertion into Service_Ticket_Issued Part(B).

SQL> Select * from Service_Ticket_Issued;

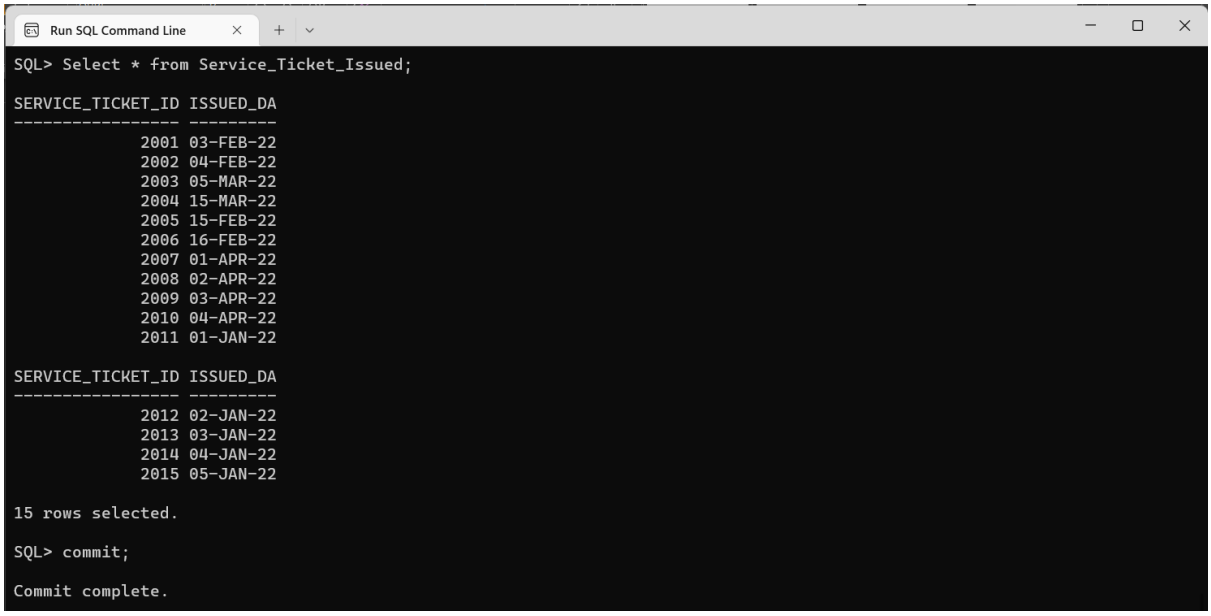


Figure 16: Screenshot of data of Service_Ticket_Issued.

Creating table Customer_Type

SQL> Create table Service_Ticket_Issued(

2 Service_Ticket_ID int primary key,

3 Issued_Date date);

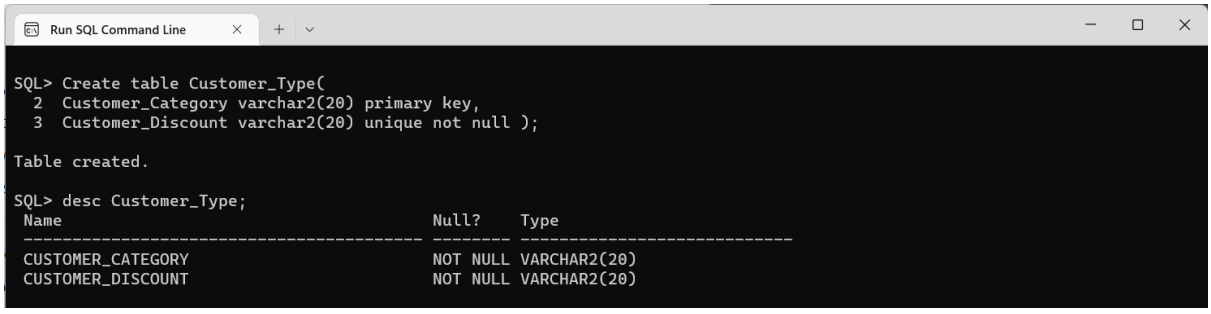


Figure 17: Screenshot of table creation of Customer_Type.

Data Implementation of Customer_Type

Insert into Customer_Type values('VIP', '50%');

1 row created.

Insert into Customer_Type values('Staff', '20%');

1 row created.

Insert into Customer_Type values('Normal Customer', '0%');

1 row created.

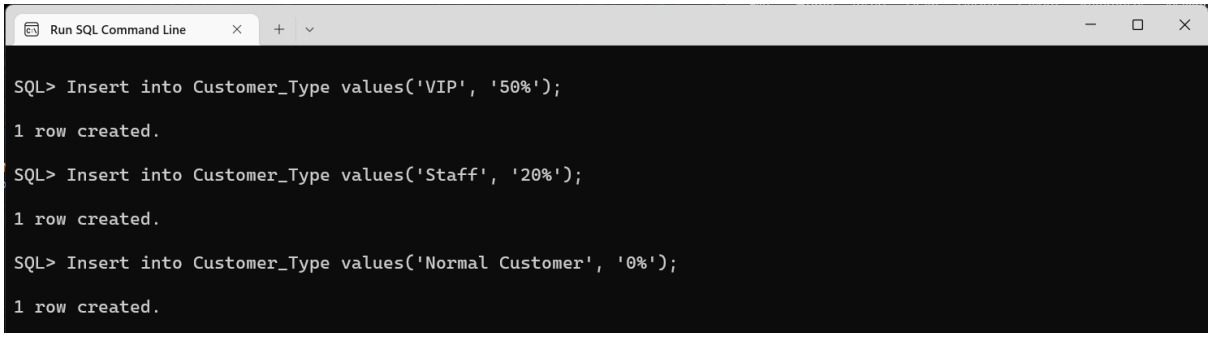


Figure 18: Screenshot of data insertion of Customer_Type

SQL> select * from Customer_Type

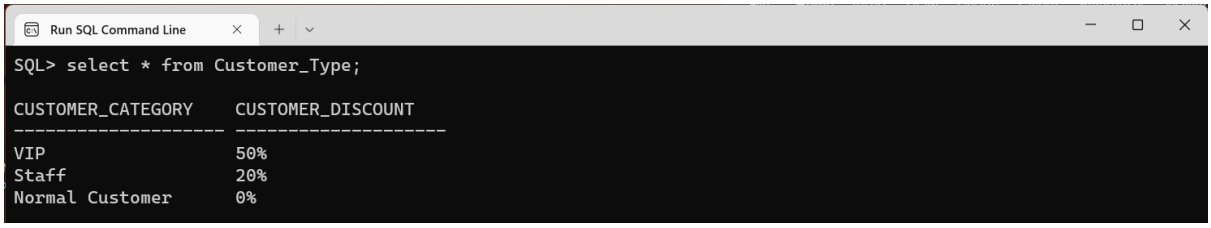


Figure 19: Screenshot of data of Customer_Type

Creation of Customer Table

```
SQL> Create table Customer(
2  Customer_ID int primary key,
3  Customer_Name varchar2(20) not null,
4  Customer_Location varchar2(20),
5  Reward_points decimal(5,2) not null,
6  Customer_Category varchar2(20),
7  Foreign Key(Customer_Category) references Customer_Type(Customer_Category));
```

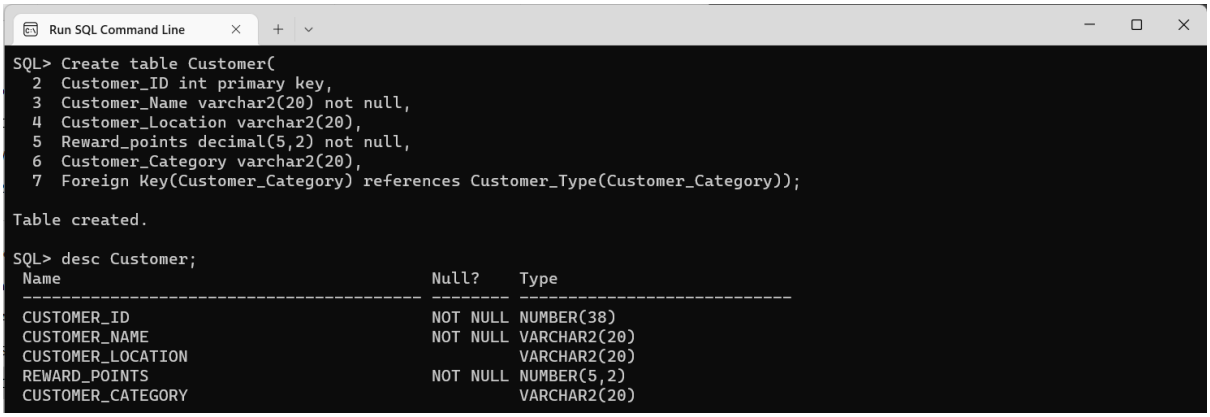


Figure 20: Screenshot of creation of Customer Table.

Data Implementation of Customer Table

```
SQL> Insert into Customer Values (2022201, 'Jennie Kim', 'Kamal Pokhari',
     120.55, 'VIP');

1 row created.

SQL> Insert into Customer Values (2022202, 'Lisa Monaban', 'Dillibazar',
     252.55, 'VIP');

1 row created.

SQL> Insert into Customer Values (2022203, 'Charlet Jr', 'Lolang', 22.55, 'Staff');

1 row created.

SQL> Insert into Customer Values (2022204, 'Marques Brownlee', 'Asan', 553.2,
     'Staff');

1 row created.
```

SQL> Insert into Customer Values (2022205, 'Ishan Kishan', 'Paknajol', 553.2, 'Normal Customer');

1 row created.

SQL> Insert into Customer Values (2022206, 'Ashley Ord', 'Dillibazar', 153.2, 'Normal Customer');

1 row created.

SQL> Insert into Customer Values (2022207, 'Elliot Choy', 'Ratnapark', 153.2, 'Normal Customer');

1 row created.

SQL> Insert into Customer Values (2022208, 'Kelly Wakasa', 'panga', 13.2, 'Normal Customer');

1 row created.

SQL> Insert into Customer Values (2022209, 'Jousha Kim', 'Dillibazar', 12.0, 'Normal Customer');

1 row created.

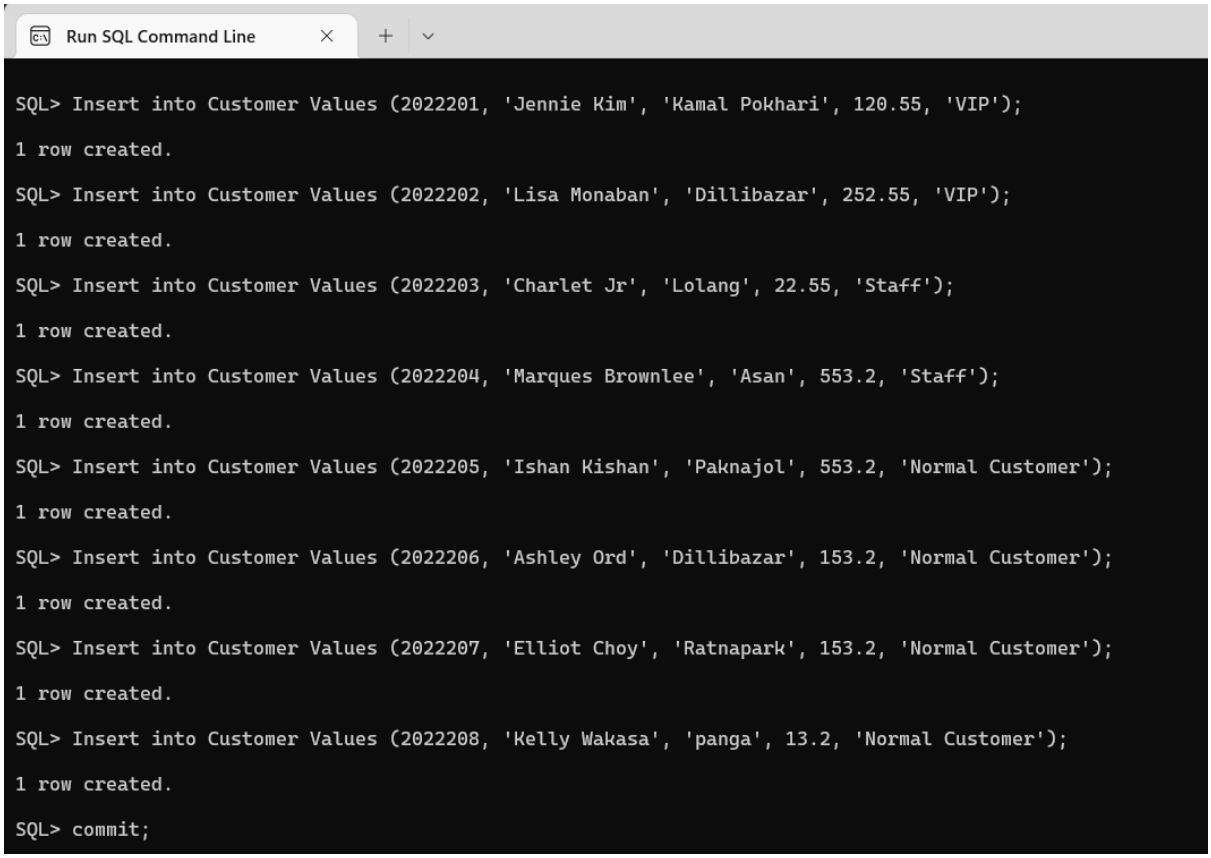


Figure 21: Screenshot of data insertion into Customer Table.

SQL> select * from Customer;

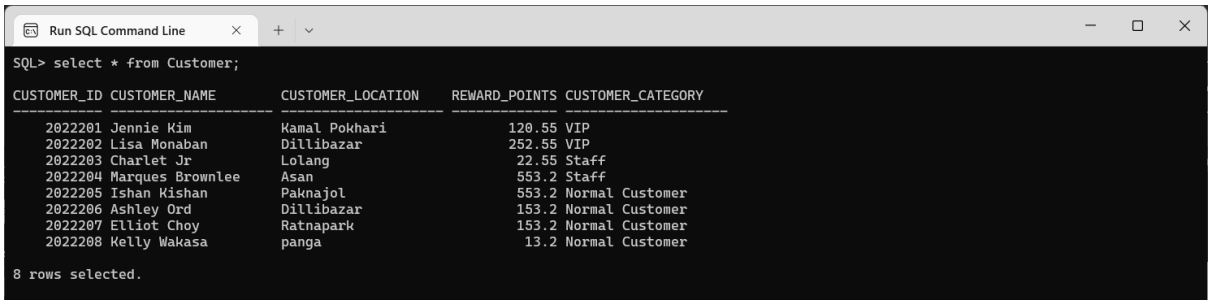


Figure 22: Screenshot of data of Customer Table

Creating Invoice Table

SQL> Create table Invoice(

2 Invoice_ID varchar2(10) primary key,

3 Invoice_Date date not null,

4 Duration varchar2(5),

5 Destination varchar2(20) not null,

6 Total_Charge varchar2(15) not null,

7 Service_ID varchar2(10),

8 Customer_ID int,

9 Service_Ticket_ID int,

10 Foreign Key(Service_ID) references Service(Service_ID),

11 Foreign Key(Customer_ID) references Customer(Customer_ID),

12 Foreign Key(Service_Ticket_ID) references Service_Ticket_Issued(Service_Ticket_ID));

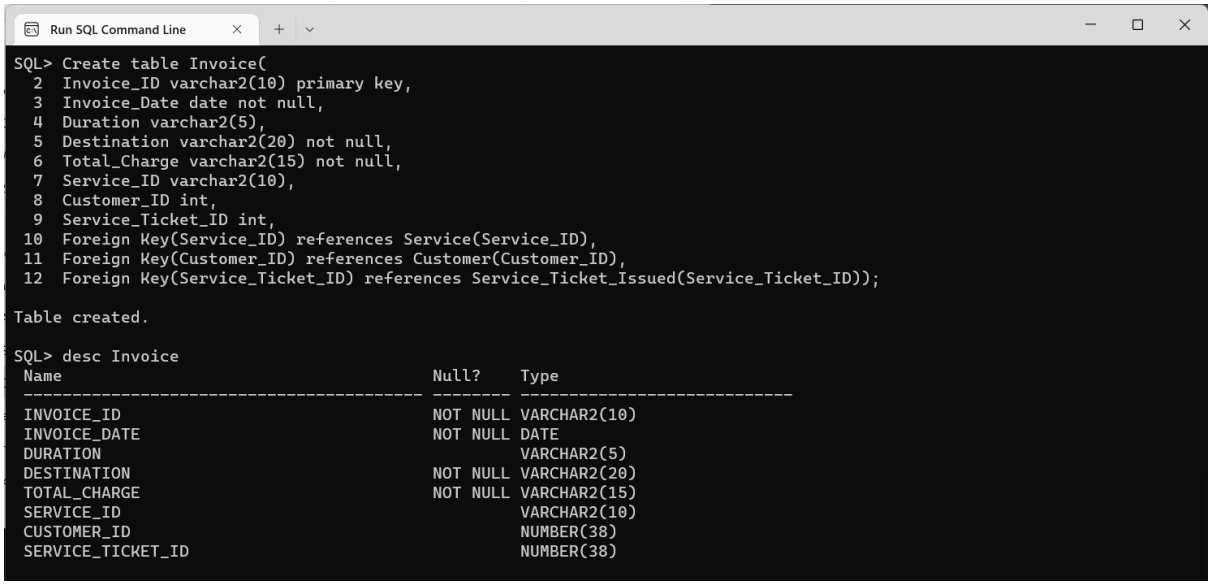


Figure 23: Screenshot of creation of Invoice Table.

Data Implementation of Invoice

SQL> Insert into Invoice values('I001','01-Jan-2021','1 hr','TIA-KamalPokari','900.50','CD 104',2022201,2001);

1 row created.

SQL> Insert into Invoice values('I002','01-Jan-2022','2 hr','TIA-Dillibazar','950.50','CD 104',2022209,2002);

1 row created.

SQL> Insert into Invoice values('I003','02-Dec-2022','25min','Kalimati-Panga','200.50','RS 101',2022206,2003);

1 row created.

SQL> Insert into Invoice values('I004','03-Dec-2022','25min','Paknajol-Asan','250.50','RS 101',2022208,2004);

1 row created.

SQL> Insert into Invoice values('I005','05-Dec-2022','55min','Ratnapark-BHKT','300.50','FD 102',2022205,2005);

1 row created.

SQL> Insert into Invoice values('I006','05-Dec-2022','55min','Ratnapark-BHKT','300.50','CD 104',2022204,2006);

1 row created.

SQL> Insert into Invoice values('I007','06-Dec-2022','55min','Ratnapark-Dillibazar','400.50','PD 103',2022202,2007);

```
1 row created.

SQL> Insert into Invoice values('I008','11-Dec-2022','2hr','TIA-
     Asan','400.50','CD 104',2022203,2008);

1 row created.

SQL> Insert into Invoice values('I009','21-Dec-2022','2hr','Dillibazar-
     lalitpur','400.50','RS 101',2022208,2009);

1 row created.

SQL> Insert into Invoice values('I010','21-Dec-2022','2hr','Dillibazar-
     lalitpur','600.50','RS 101',2022203,2010);

1 row created.

SQL> Insert into Invoice values('I011','21-Dec-2022','2hr','TIA-
     Lolang','1600.50','CD 104',2022203,2012);

1 row created.

SQL> Insert into Invoice values('I012','21-Dec-2022','5min','Asan-
     Paknajol','60.50','RS 101',2022203,2013);

1 row created.

SQL> Insert into Invoice values('I013','22-Dec-2022','5min','Asan-
     Paknajol','160.50','RS 101',2022204,2014);

1 row created.

SQL> Insert into Invoice values('I014','23-Dec-2022','5min','Dillibazar-
     Ratnapark','160.50','RS 101',2022207,2015);

1 row created.

SQL> Insert into Invoice values('I015','22-Dec-2022','5min','Dillibazar-
     Lolang','160.50','RS 101',2022206,2015);

1 row created.
```

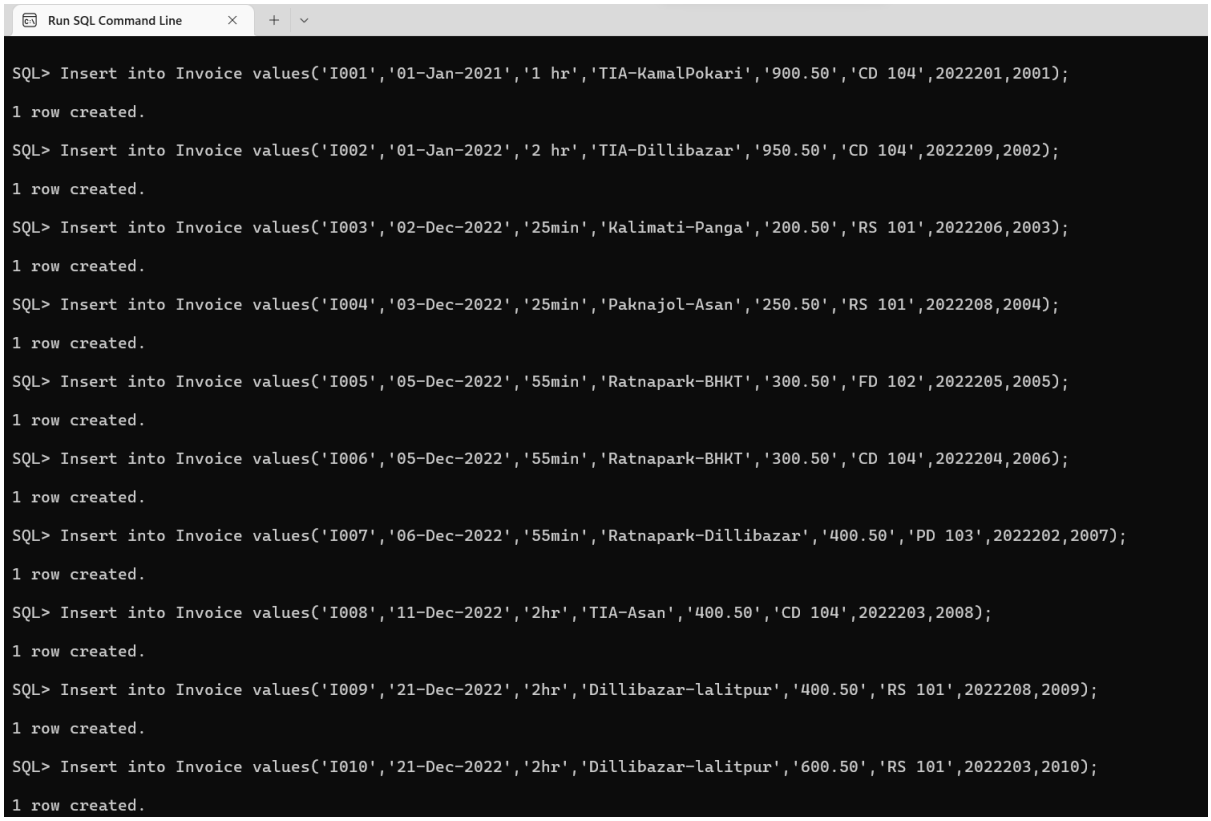


Figure 24: Screenshot of data insertion into Invoice Table Part(A).

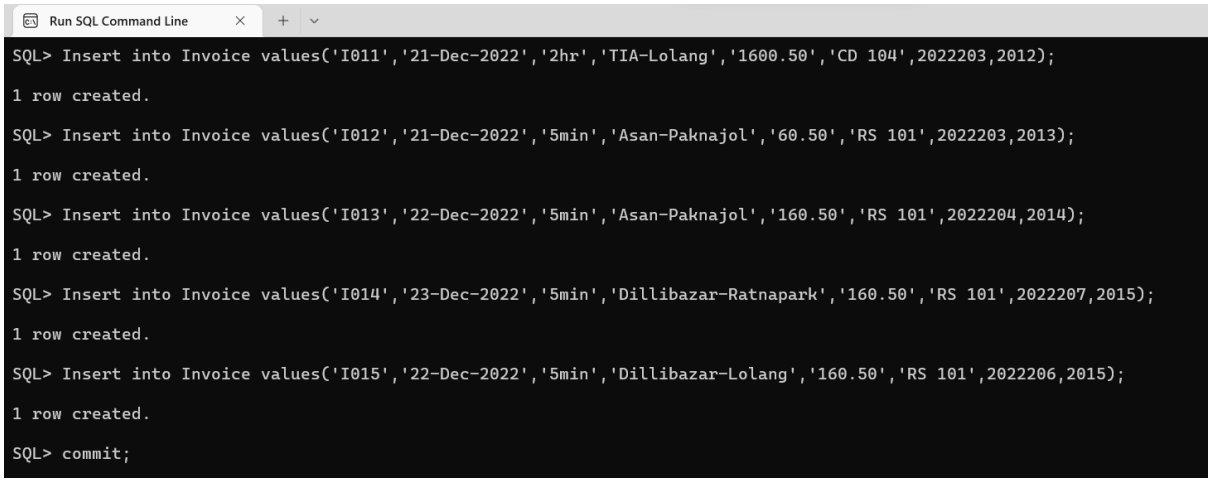
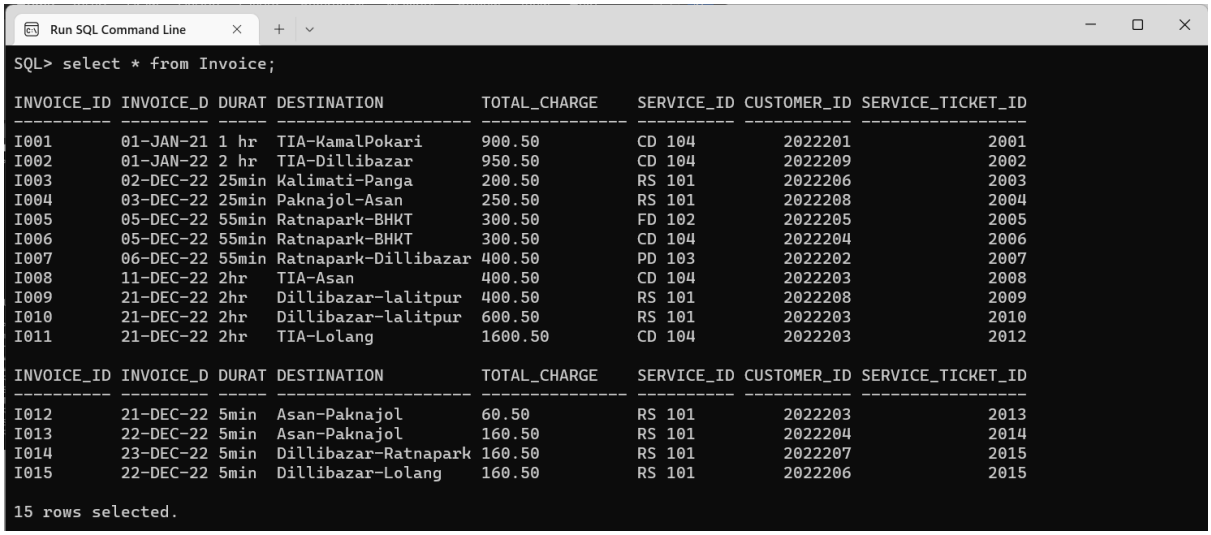


Figure 25: Screenshot of data insertion into Invoice Table Part(B).

SQL> select * from Invoice;



```
Run SQL Command Line
SQL> select * from Invoice;

INVOICE_ID INVOICE_D DURAT DESTINATION          TOTAL_CHARGE    SERVICE_ID CUSTOMER_ID SERVICE_TICKET_ID
---------- --------- ----- -------------------- --------------- ---------- ----------- ----------------
I001       01-JAN-21 1 hr  TIA-KamalPokari      900.50          CD 104          2022201             2001
I002       01-JAN-22 2 hr  TIA-Dillibazar       950.50          CD 104          2022209             2002
I003       02-DEC-22 25min Kalimati-Panga       200.50          RS 101          2022206             2003
I004       03-DEC-22 25min Paknajol-Asan        250.50          RS 101          2022208             2004
I005       05-DEC-22 55min Ratnapark-BHKT       300.50          FD 102          2022205             2005
I006       05-DEC-22 55min Ratnapark-BHKT       300.50          CD 104          2022204             2006
I007       06-DEC-22 55min Ratnapark-Dillibazar 400.50          PD 103          2022202             2007
I008       11-DEC-22 2hr   TIA-Asan             400.50          CD 104          2022203             2008
I009       21-DEC-22 2hr   Dillibazar-lalitpur  400.50          RS 101          2022208             2009
I010       21-DEC-22 2hr   Dillibazar-lalitpur  600.50          RS 101          2022203             2010
I011       21-DEC-22 2hr   TIA-Lolang           1600.50         CD 104          2022203             2012

INVOICE_ID INVOICE_D DURAT DESTINATION          TOTAL_CHARGE    SERVICE_ID CUSTOMER_ID SERVICE_TICKET_ID
---------- --------- ----- -------------------- --------------- ---------- ----------- ----------------
I012       21-DEC-22 5min  Asan-Paknajol        60.50           RS 101          2022203             2013
I013       22-DEC-22 5min  Asan-Paknajol        160.50          RS 101          2022204             2014
I014       23-DEC-22 5min  Dillibazar-Ratnapark 160.50          RS 101          2022207             2015
I015       22-DEC-22 5min  Dillibazar-Lolang    160.50          RS 101          2022206             2015

15 rows selected.
```

Figure 26: Screenshot of data on Invoice Table.

Creating table Vehicle_Invoice_Generator

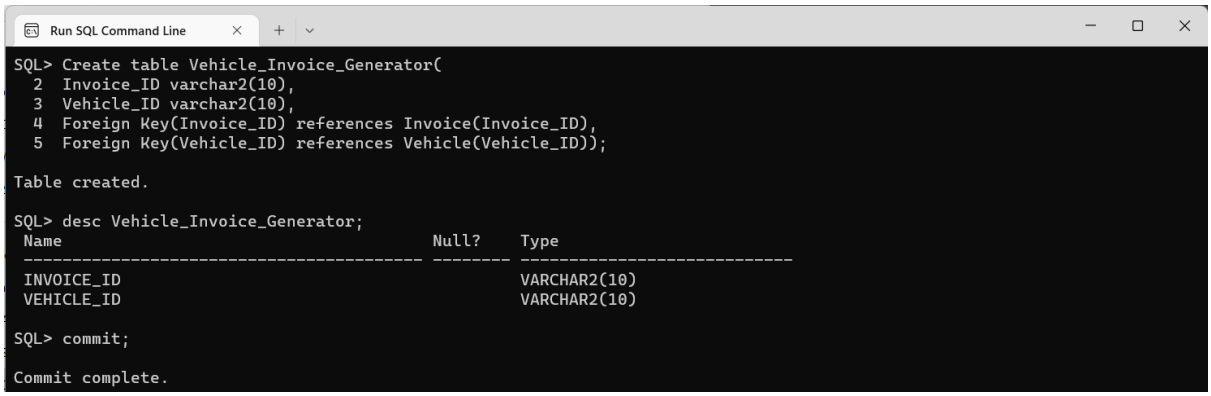
SQL> Create table Vehicle_Invoice_Generator(

2 Invoice_ID varchar2(10),

3 Vehicle_ID varchar2(10),

4 Foreign Key(Invoice_ID) references Invoice(Invoice_ID),

5 Foreign Key(Vehicle_ID) references Vehicle(Vehicle_ID));



```
Run SQL Command Line
SQL> Create table Vehicle_Invoice_Generator(
  2  Invoice_ID varchar2(10),
  3  Vehicle_ID varchar2(10),
  4  Foreign Key(Invoice_ID) references Invoice(Invoice_ID),
  5  Foreign Key(Vehicle_ID) references Vehicle(Vehicle_ID));

Table created.

SQL> desc Vehicle_Invoice_Generator;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INVOICE_ID                                         VARCHAR2(10)
 VEHICLE_ID                                         VARCHAR2(10)

SQL> commit;

Commit complete.
```

Figure 27: Screenshot of table creation of Vehicle_Invoice_Generator.

Data Implementation of Vehicle_Invoice_Generator

```
SQL> Insert into Vehicle_Invoice_Generator values('I001', 'S2 BA 1241');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I001', 'S5 RA 2012');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I003', 'S6 SU 2025');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I004', 'S3 BA 5031');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I005', 'S3 BA 5031');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I006', 'S2 BA 8018');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I009', 'S2 BA 5500');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I012', 'S3 BA 5031');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I012', 'S2 BA 7175');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I013', 'S5 RA 2012');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I014', 'S2 BA 8018');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I015', 'S3 BA 5031');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I002', 'S3 BA 5031');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I003', 'S2 BA 8018');
1 row created.
```

```
SQL> Insert into Vehicle_Invoice_Generator values('I004', 'S2 BA 5500');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I007', 'S3 BA 5031');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I008', 'S1 JK 2121');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I009', 'S2 BA 5500');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I010', 'S3 BA 7771');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I011', 'S1 JK 2121');
1 row created.
SQL> Insert into Vehicle_Invoice_Generator values('I010', 'S6 SU 2025')
;
1 row created.
```

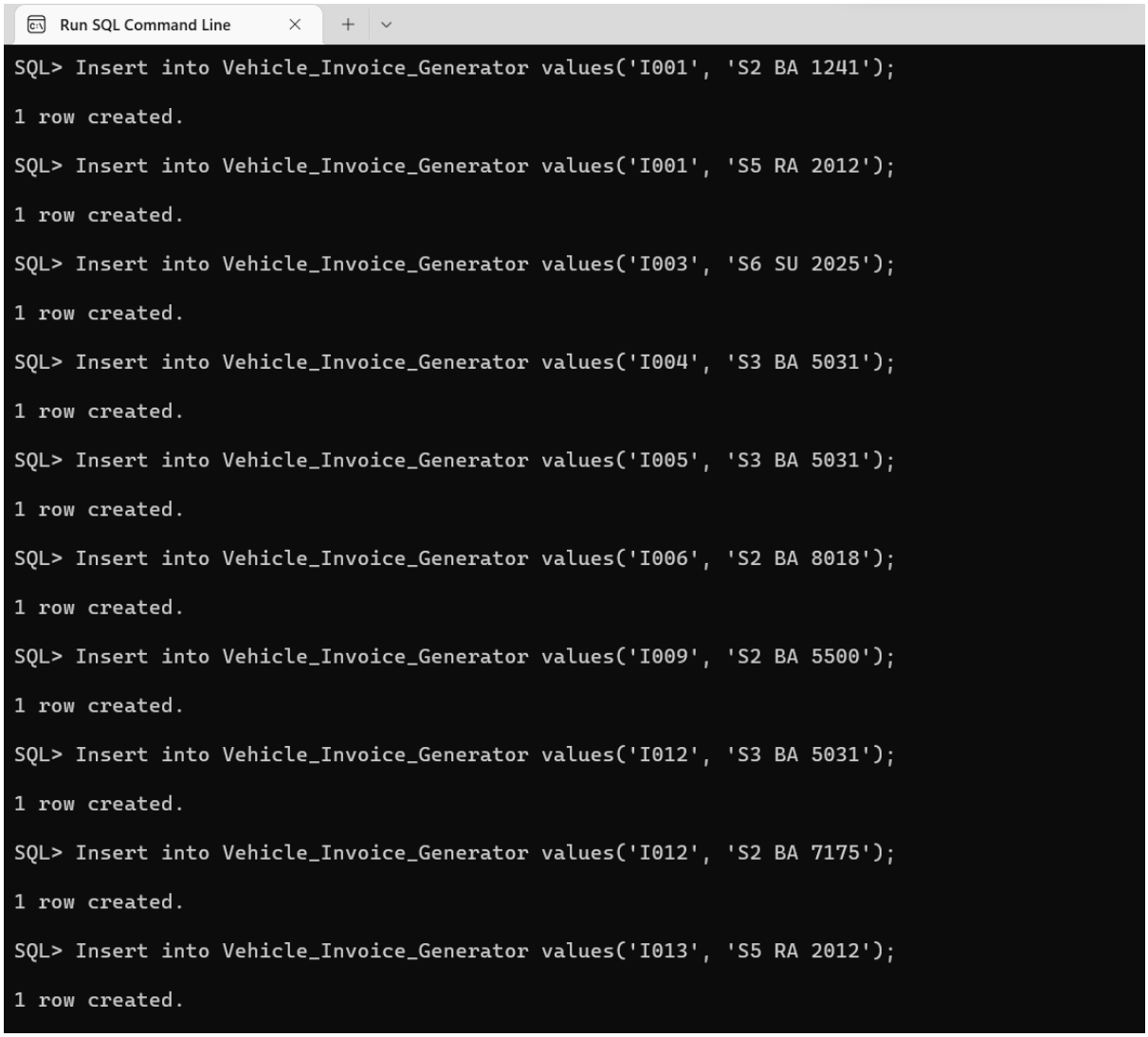


Figure 28: Screenshot of data insertion of Vehicle_Invoice_Generator Part(A).

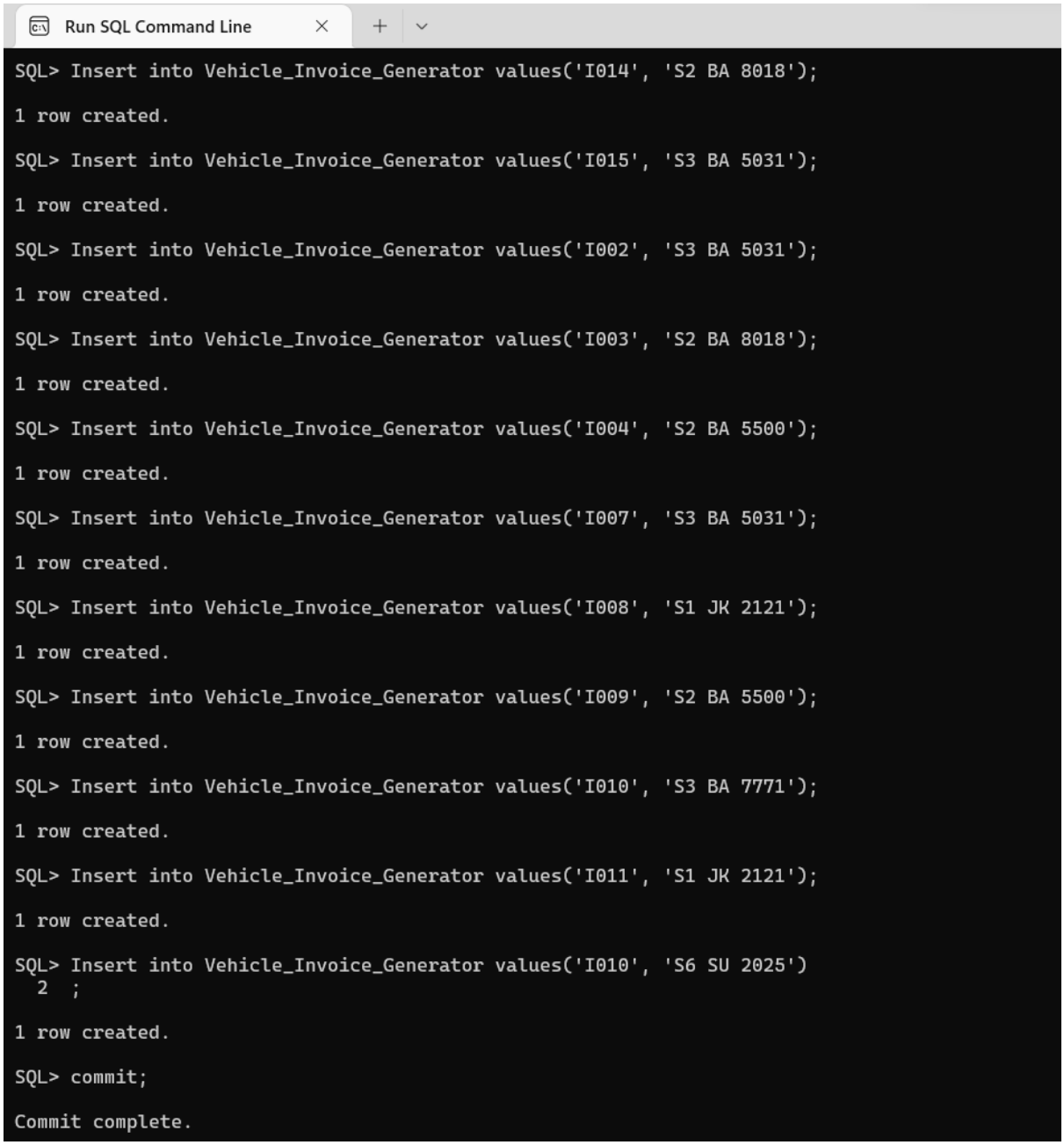


Figure 29: Screenshot of data insertion of Vehicle_Invoice_Generator part(B).

SQL> select * from Vehicle_Invoice_Generator;

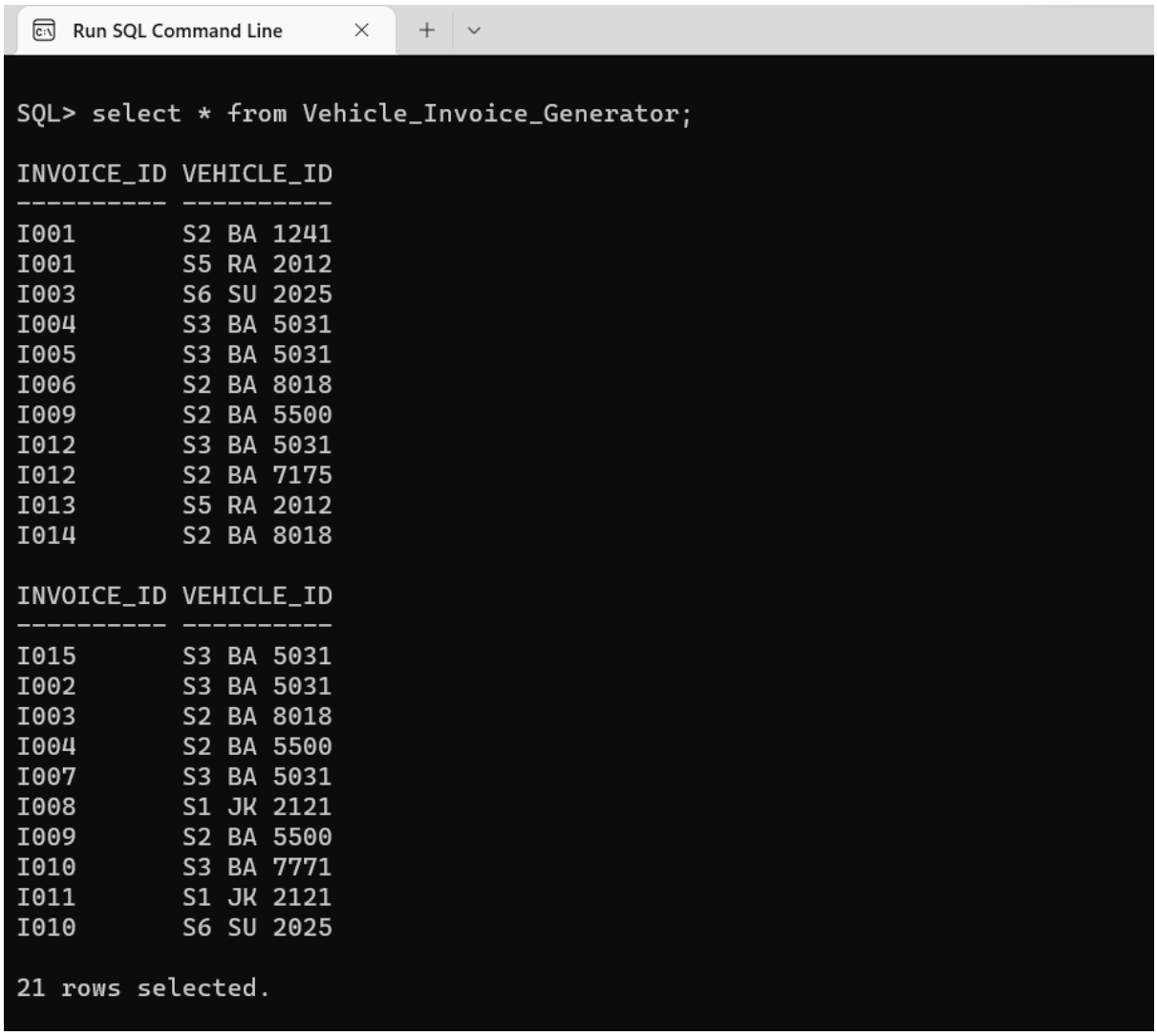


Figure 30: Screenshot of data of Vehicle_Invoice_Generator.

7. Database Querying

A quey is a question or a request expressed in a formal manner. In computer science, a query is an essentially the same thing, the only difference is the answer or retrieved information comes from a database (Hughes, 2022).

7.1.1 Informational Queries

a. List all customers according to category

Query:-

```
SQL> SELECT Customer_ID, Customer_Name, Customer_Category, Customer_Location
  2  FROM Customer
  3  ORDER BY Customer_Category;

CUSTOMER_ID CUSTOMER_NAME        CUSTOMER_CATEGORY    CUSTOMER_LOCATION
----------- -------------------- -------------------- --------------------
    2022205 Ishan Kishan         Normal Customer      Paknajol
    2022208 Kelly Wakasa         Normal Customer      panga
    2022207 Elliot Choy          Normal Customer      Ratnapark
    2022206 Ashley Ord           Normal Customer      Dillibazar
    2022209 Jousha Kim           Normal Customer      Dillibazar
    2022204 Marques Brownlee     Staff                Asan
    2022203 Charlet Jr           Staff                Lolang
    2022202 Lisa Monaban         VIP                  Dillibazar
    2022201 Jennie Kim           VIP                  Kamal Pokhari

9 rows selected.
```

Figure 31: Screenshot of listing customers according to category.

b. Find model and vehicle variants and sort by price in descending order.

Query:-

```
SQL> SELECT Vehicle_ID, Vehicle_Model, Vehicle_Variants, Vehicle_Price
  2  FROM Vehicle
  3  ORDER BY Vehicle_Price DESC;

VEHICLE_ID VEHICLE_MODEL        VEHICLE_VARIANTS          VEHICLE_PRICE
---------- -------------------- ------------------------- -------------
S5 RA 2012 Renault Zoe          Scooter                         5500000
S2 BA 7175 Ford                 Cab                             3500000
S1 JK 2121 Ford                 Cab                             3000000
S2 BA 5500 Zoe 500              Cab                             3000000
S3 BA 7771 NUI                  Scooter                         3000000
S6 SU 2025 Nissal Leaf          Cab                             2500000
S2 BA 8018 Benilli              Bike                             900000
S3 BA 5031 Benelli              Bike                             700000
S5 RA 2241 Duke 250             Bike                             500000
S3 BA 5051 Pulsar 200           Bike                             300000
S2 BA 1241 Splender             Bike                             250000

11 rows selected.
```

Figure 32: Screenshot of displaying model, variants of vehicle short by price in descending order.

c. Display the number of total vehicles that use petrol.

Query:-

```
SQL> SELECT COUNT(*)
  2  FROM Vehicle
  3  WHERE LOWER(Vehicle_Type) = 'petrol';
```

```
SQL> SELECT COUNT(*)
  2  FROM Vehicle
  3  WHERE LOWER(Vehicle_Type) = 'petrol';

  COUNT(*)
----------
         7
```

Figure 33: Screenshot of displaying the total number of vehicle that use petrol.

d. List all tickets issued from 2022/03/05 to 2022/04/05.

Query:-

```
SQL> SELECT Service_Ticket_ID, TO_CHAR(issued_date, 'YYY-MM-DD') as Issued_Date
  2  FROM Service_Ticket_Issued
  3  WHERE Issued_Date BETWEEN TO_DATE('2022-03-05', 'YYYY-MM-DD') AND TO_DATE('2022-04-05', 'YYYY-MM-DD');

SERVICE_TICKET_ID ISSUED_DA
----------------- ---------
             2003 022-03-05
             2004 022-03-15
             2007 022-04-01
             2008 022-04-02
             2009 022-04-03
             2010 022-04-04

6 rows selected.
```

Figure 34: Screenshot of listing ticket_issued from 2202/03/05 to 2022/04/05.

e. List the name of the driver who has the character ‘s’ between their names.

Query:-

```
SQL> SELECT Driver_Name
2  FROM Driver
3  WHERE Driver_Name LIKE '%s %';
```

```
SQL> SELECT Driver_Name
  2  FROM Driver
  3  WHERE Driver_Name LIKE '%s %';

DRIVER_NAME
-------------------------
James Smith
```

Figure 35: Screenshot of listing the driver name who has character ‘s’ between their names.

7.1.2 Transactional Queries Including Relational Algebra

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

Query

```
SQL> SELECT c.Customer_ID,c.Customer_Name, TO_CHAR(i.Invoice_Date,
'YYYY') as "Year", i.Total_Charge as "Total Cost", s.Service_Type
  2  FROM Customer c
  3  JOIN Invoice i ON c.Customer_ID = i.Customer_ID
  4  JOIN Service s ON s.Service_ID = s.Service_ID
  5  WHERE c.Customer_ID ='2022201';
```

Relation Algebra

π Customer_ID, Customer_Name, TO_CHAR(Invoice_Date, 'YYYY'), Total_Charge, Service_Type (σ Customer_ID='2022201' (Customer ⋈ Invoice ⋈ Service))

```
SQL> SELECT c.Customer_ID,c.Customer_Name, TO_CHAR(i.Invoice_Date, 'YYYY') as "Year", i.Total_Charge as "Total Cost", s.Service_Type
  2  FROM Customer c
  3  JOIN Invoice i ON c.Customer_ID = i.Customer_ID
  4  JOIN Service s ON s.Service_ID = s.Service_ID
  5  WHERE c.Customer_ID ='2022201';

CUSTOMER_ID CUSTOMER_NAME        Year Total Cost      SERVICE_TYPE
----------- -------------------- ---- --------------- --------------------
    2022201 Jennie Kim           2021 900.50          Ride Sharing
    2022201 Jennie Kim           2021 900.50          Food Delivery
    2022201 Jennie Kim           2021 900.50          Package Delivery
    2022201 Jennie Kim           2021 900.50          Courier Delivery
```

Figure 36: Screenshot of selecting total cost, type of service of particular customer in year.

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’.

Query

```
SQL> SELECT s.Service_ID, s.Service_Type, s.Service_Charge,
  2  d.Driver_Name,
  3  TO_CHAR(i.Invoice_Date, 'MONTH') AS "Current Month"
  4  From Service s
  5  JOIN Invoice i ON s.Service_ID = i.Service_ID
  6  JOIN Vehicle_Invoice_Generator vig ON i.Invoice_ID = vig.Invoice_ID
  7  JOIN Vehicle v ON vig.Vehicle_ID = v.Vehicle_ID
  8  JOIN Driver d ON v.Driver_ID = d.Driver_ID
  9  WHERE TO_CHAR(i.Invoice_Date, 'MM') = TO_CHAR(TO_DATE('2022-12', 'YYYY-MM'), 'MM') AND Driver_Name LIKE 'A%';
```

Relational Algebra

π Service_ID, Service_Type, Service_Charge, Driver_Name, TO_CHAR(Invoice_Date, 'MONTH') (σ TO_CHAR(Invoice_Date, 'MM') = TO_CHAR(TO_DATE('2022-12', 'YYYY-MM'), 'MM') AND Driver_Name LIKE 'A%' (Service ⋈ Invoice ⋈ Vehicle_Invoice_Generator ⋈ Vehicle ⋈ Driver))

```
SQL> SELECT s.Service_ID, s.Service_Type, s.Service_Charge,
  2  d.Driver_Name,
  3  TO_CHAR(i.Invoice_Date, 'MONTH') AS "Current Month"
  4  From Service s
  5  JOIN Invoice i ON s.Service_ID = i.Service_ID
  6  JOIN Vehicle_Invoice_Generator vig ON i.Invoice_ID = vig.Invoice_ID
  7  JOIN Vehicle v ON vig.Vehicle_ID = v.Vehicle_ID
  8  JOIN Driver d ON v.Driver_ID = d.Driver_ID
  9  WHERE TO_CHAR(i.Invoice_Date, 'MM') = TO_CHAR(TO_DATE('2022-12', 'YYYY-MM'), 'MM') AND Driver_Name LIKE 'A%';

SERVICE_ID SERVICE_TYPE         SERVICE_CHARGE DRIVER_NAME           Current Month
---------- -------------------- -------------- --------------------- ---------------------------------
CD 104     Courier Delivery               250 Andrew Browm          DECEMBER
CD 104     Courier Delivery               250 Andrew Browm          DECEMBER
```

Figure 37: Screenshot of selecting the details of service and driver name which first name start with A

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

Query

SQL> SELECT c.Customer_Name, c.Customer_Location as "Delivery of Location" , s.service_type

2 FROM Customer c

3 JOIN Invoice i ON c.customer_id = i.customer_id

4 JOIN Service s ON i.service_id = s.service_id

5 WHERE s.service_type = 'Courier Delivery';

Relational Algebra

π Customer_Name, Customer_Location, Service_Type (σ Service_Type='Courier Delivery' (Customer ⋈ Invoice ⋈ Service))

```
SQL> SELECT c.Customer_Name, c.Customer_Location as "Delivery of Location" , s.service_type
  2  FROM Customer c
  3  JOIN Invoice i ON c.customer_id = i.customer_id
  4  JOIN Service s ON i.service_id = s.service_id
  5  WHERE s.service_type = 'Courier Delivery';

CUSTOMER_NAME        Delivery of Location SERVICE_TYPE
-------------------- -------------------- --------------------
Jennie Kim           Kamal Pokhari        Courier Delivery
Charlet Jr           Lolang               Courier Delivery
Charlet Jr           Lolang               Courier Delivery
Marques Brownlee     Asan                 Courier Delivery
Jousha Kim           Dillibazar           Courier Delivery
```

Figure 38: Screenshot of selecting customer name who have used Courier Delivery.

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

Query

```
SQL> SELECT *FROM
  2 (
  3 SELECT * FROM Driver
  4 ORDER BY Driver_Salary DESC
  5 )
  6 WHERE rownum <=3
  7 ORDER BY Driver_Salary;
```

Relation Algebra

π * (π * (Driver) WHERE rownum <=3) ORDER BY Driver_Salary

```
SQL> SELECT *FROM
  2  (
  3  SELECT * FROM Driver
  4  ORDER BY Driver_Salary DESC
  5  )
  6  WHERE rownum <=3
  7  ORDER BY Driver_Salary;

 DRIVER_ID DRIVER_NAME               DRIVER_CATEGORY DRIVER_STATUS DRIVER_SALARY
---------- ------------------------ --------------- ------------- -------------
       104 Andrew Browm             Full Time       In Active          50500.15
       109 Ryan Talor               Full Time       IN Active          70000.15
       103 Maria Williams           Full Time       Active             99910.15
```

Figure 39: Screenshot of selecting the to 3 highest earning driver.

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

Before starting this query, I did not assign normal customer and staff on same destination. So I update same destination for ID I005

```
SQL> Update Invoice
  2  Set Destination =  'Asan-Paknajol'
  3  Where Invoice_ID = 'I005';

1 row updated.
```

Figure 40: Screenshot of Updating Table Invoice.

This is the result of invoice table after updating table.

```
SQL> Select * from Invoice;

INVOICE_ID INVOICE_D DURAT DESTINATION          TOTAL_CHARGE    SERVICE_ID CUSTOMER_ID SERVICE_TICKET_ID
---------- --------- ----- -------------------- --------------- ---------- ----------- ----------------
I001       01-JAN-21 1 hr  TIA-KamalPokari      900.50          CD 104         2022201             2001
I002       01-JAN-22 2 hr  TIA-Dillibazar       950.50          CD 104         2022209             2002
I003       02-DEC-22 25min Kalimati-Panga       200.50          RS 101         2022206             2003
I004       03-DEC-22 25min Paknajol-Asan        250.50          RS 101         2022208             2004
I005       05-DEC-22 55min Asan-Paknajol        300.50          FD 102         2022205             2005
I006       05-DEC-22 55min Ratnapark-BHKT       300.50          CD 104         2022204             2006
I007       06-DEC-22 55min Ratnapark-Dillibazar 400.50          PD 103         2022202             2007
I008       11-DEC-22 2hr   TIA-Asan             400.50          CD 104         2022203             2008
I009       21-DEC-22 2hr   Dillibazar-lalitpur  400.50          RS 101         2022208             2009
I010       21-DEC-22 2hr   Dillibazar-lalitpur  600.50          RS 101         2022203             2010
I011       21-DEC-22 2hr   TIA-Lolang           1600.50         CD 104         2022203             2012

INVOICE_ID INVOICE_D DURAT DESTINATION          TOTAL_CHARGE    SERVICE_ID CUSTOMER_ID SERVICE_TICKET_ID
---------- --------- ----- -------------------- --------------- ---------- ----------- ----------------
I012       21-DEC-22 5min  Asan-Paknajol        60.50           RS 101         2022203             2013
I013       22-DEC-22 5min  Asan-Paknajol        160.50          RS 101         2022204             2014
I014       23-DEC-22 5min  Dillibazar-Ratnapark 160.50          RS 101         2022207             2015
I015       22-DEC-22 5min  Dillibazar-Lolang    160.50          RS 101         2022206             2015

15 rows selected.
```

Figure 41: Screenshot of result after updating invoice table.

Query

```
SQL> SELECT v.Vehicle_ID, v.Vehicle_Type, v.Vehicle_Variants, v.Vehicle_Rate, c.Customer_ID, i.Destination, c.Customer_Category,
  2  CASE
  3    WHEN c.Customer_Category = 'Staff' THEN v.Vehicle_Rate * (1-0.2)
  4          WHEN c.Customer_Category = 'Normal Customer' THEN v.Vehicle_Rate * 1
  5    ELSE v.Vehicle_Rate
  6  END AS Discount_rate
  7  FROM Customer c
  8  JOIN Invoice i ON c.customer_ID = i.customer_id
  9  JOIN Vehicle_Invoice_Generator vig ON i.invoice_ID = vig.Invoice_Id
 10  JOIN Vehicle v ON v.Vehicle_ID = vig.vehicle_id
 11   WHERE c.Customer_Category != 'VIP' and Destination = 'Asan-Paknajol'
 12  ORDER BY customer_category;
```

Relation Algebra

π Vehicle_ID, Vehicle_Type, Vehicle_Variants, Vehicle_Rate, Customer_ID, Destination, Customer_Category, (CASE

WHEN Customer_Category='Staff' THEN Vehicle_Rate*(1-0.2)

WHEN Customer_Category='Normal Customer' THEN Vehicle_Rate*1

ELSE Vehicle_Rate

END) AS Discount_Rate

(σ Customer_Category!='VIP' and Destination='Asan-Paknajol' (Customer ⋈ Invoice ⋈ Vehicle_Invoice_Generator ⋈ Vehicle)) ORDER BY Customer_Category

```
SQL> SELECT v.Vehicle_ID, v.Vehicle_Type, v.Vehicle_Variants, v.Vehicle_Rate, c.Customer_ID, i.Destination, c.Customer_Category,
  2  CASE
  3    WHEN c.Customer_Category = 'Staff' THEN v.Vehicle_Rate * (1-0.2)
  4    WHEN c.Customer_Category = 'Normal Customer' THEN v.Vehicle_Rate * 1
  5    ELSE v.Vehicle_Rate
  6  END AS Discount_rate
  7  FROM Customer c
  8  JOIN Invoice i ON c.customer_ID = i.customer_id
  9  JOIN Vehicle_Invoice_Generator vig ON i.invoice_ID = vig.Invoice_Id
 10  JOIN Vehicle v ON v.Vehicle_ID = vig.vehicle_id
 11  WHERE c.Customer_Category != 'VIP' and Destination = 'Asan-Paknajol'
 12  ORDER BY customer_category;

VEHICLE_ID VEHICLE_TYPE         VEHICLE_VARIANTS          VEHICLE_RATE CUSTOMER_ID DESTINATION          CUSTOMER_CATEGORY    DISCOUNT_RATE
---------- -------------------- ------------------------- ------------ ----------- -------------------- -------------------- -------------
S3 BA 5031 Petrol               Bike                             350     2022205 Asan-Paknajol        Normal Customer               350
S2 BA 7175 Disel                Cab                            999.5     2022203 Asan-Paknajol        Staff                       799.6
S5 RA 2012 Petrol               Scooter                        130.5     2022204 Asan-Paknajol        Staff                       104.4
S3 BA 5031 Petrol               Bike                             350     2022203 Asan-Paknajol        Staff                         280

SQL>
```

Figure 42: Screenshot of displaying vehicle rate according to staff and normal customer.

8. File Creation

8.1 Creating Dump File

Step 1: Go to your folder where you want to create and type cmd which opens command terminal.

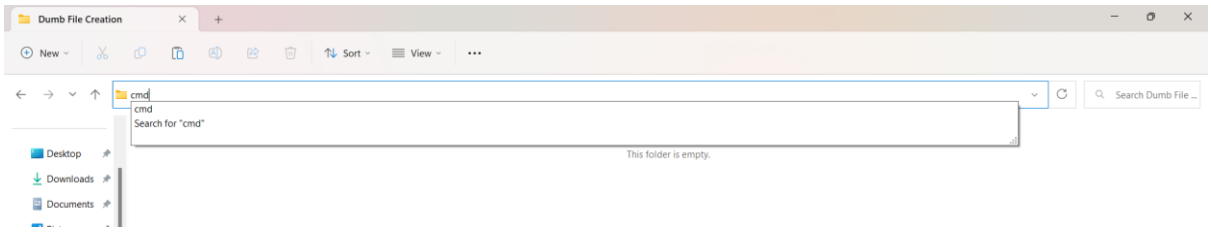


Figure 43: Screenshot of creating dump file step1

Step 2: After that type command “exp coursework_suman/suman file = coursework_suman.dmp”. After that press enter.

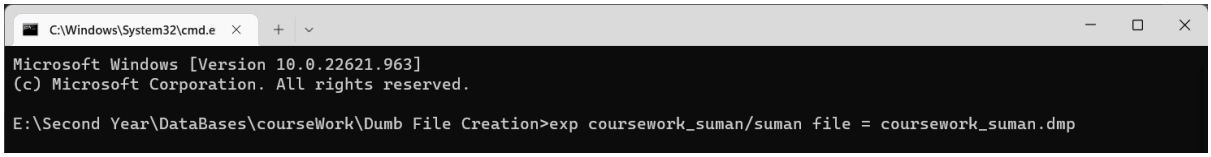


Figure 44: Screenshot of exporting dump file into coursework_suman.dmp

Step 3: After pressing enter the process takes times for creating dump file. After some time “Export terminated successfully without warning” is seen and the dump file is created.

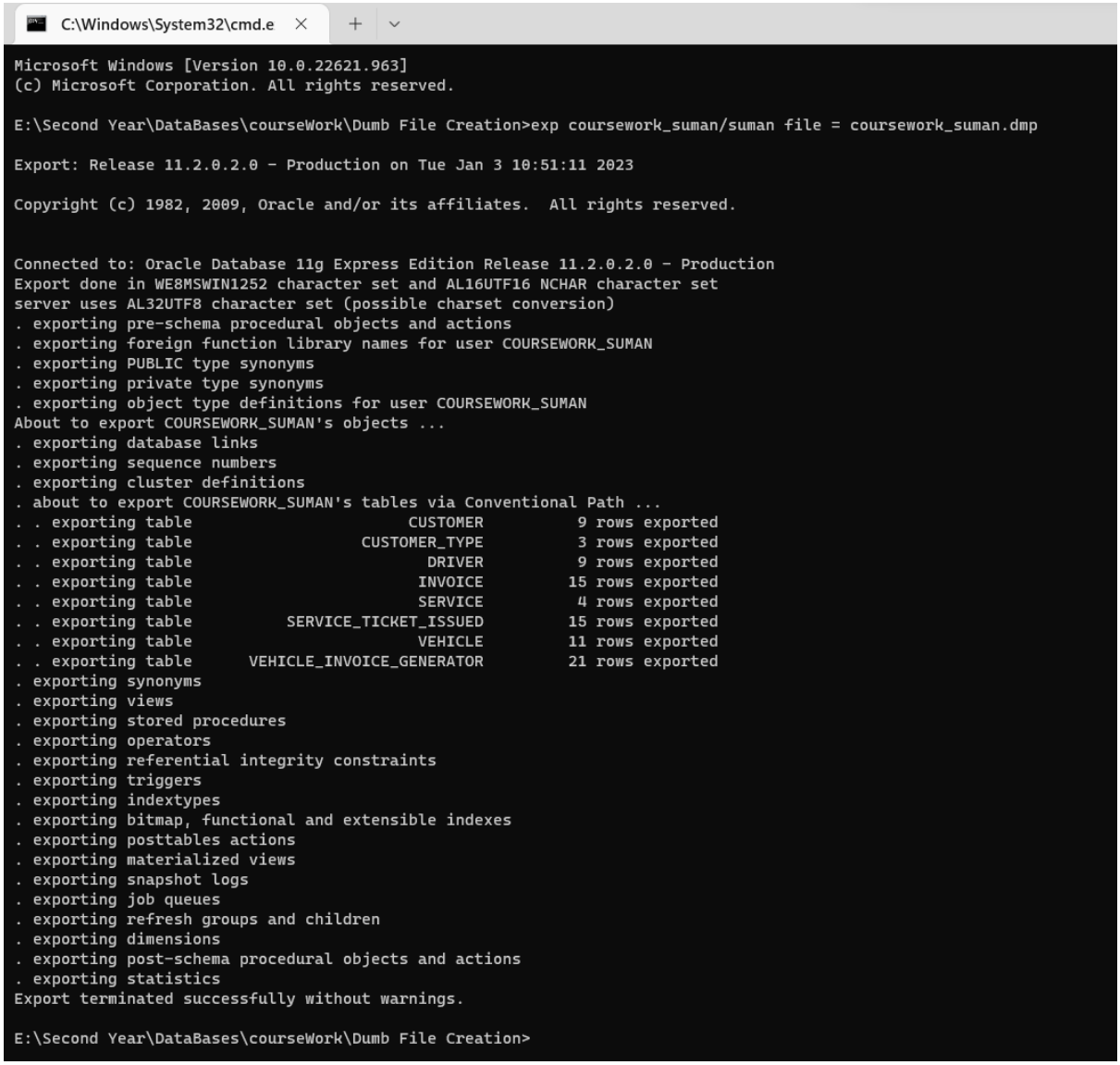


Figure 45: Screenshot of process of creating dump file.

Step 4: Here the dump file is created successfully.

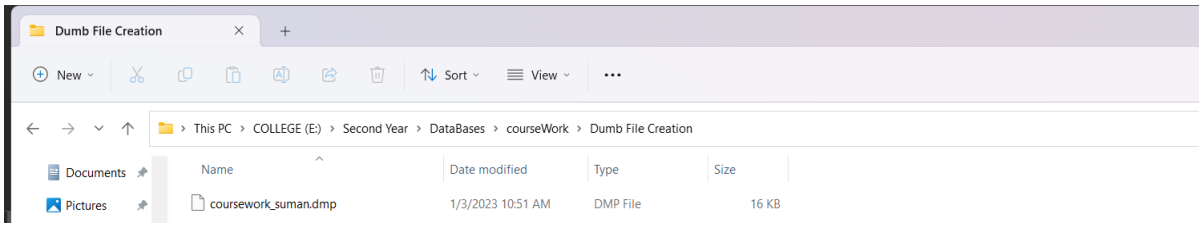


Figure 46: Screenshot of created dump file.

9. Critical Evaluation

CC5051NI- Databases is a significant semester module in Level 5 taught by outstanding lecturers and tutors. They combine practical experience with a gift for teaching. The module's primary objective is to enable students to comprehend and use approaches for Analysing, Designing, and Developing Database Systems. Similarly, the module discusses challenges related to Database System Design and Implementation. The module's primary purpose is to assist in developing an entity-relationship module from a practical problem specification. Additionally, it employs formal design techniques, such as Normalization, to generate the database schema. Similarly, it aids in extracting data from databases using Relational Algebra and SQL. Finally, it assists in designing and implementing a Database system from a conceptual Data Model. The knowledge gained in this module will aid in completing subsequent modules such as Software Engineering, Emerging Programming Platforms and Technologies, and a variety of other real-world circumstances

Nevertheless, to say, I've taken this coursework as full of fun, challenges and source of knowledge. The coursework is an example of developing the database management system. It was all about creating database for a PATHAO NEPAL. Since the coursework is all about creating database, I personally had a lot of hard time, thrilling and enjoyable moment while dealing with all the data arrangements. Creating a relational diagram, normalizing it, making final relation diagram, implementing data on oracle-based SQL program and solving the queries was not easy to deal with but, yet I had lot of fun. I've taken this project work on the basics of understanding how a understanding how a computerized database system is used to develop. Moreover, I got opportunities to learn practically about data modelling with E-R Diagram, design of relational diagram, data dictionary and how to use SQL queries for the creation of database, tables with insertion of records and manipulation of them as per the requirement. The coursework is the great

reference for the future. From this course work one can learn many knowledge, technique, have many ideas, concepts for upcoming problems in near future.

In summary, this coursework was important because it helped the student to better understand database system and how to manage them. They gained knowledge and skills that will allow them to effectively deal with any future database-related challenges or tasks.

10. Bibliography

Hughes, A., 2022. *query.* [Online]
Available at:
https://www.techtarget.com/searchdatamanagement/definition/query
[Accessed 30 December 2022].

Jacqueline, B., 2022. *techtarget.* [Online]
Available at:
https://www.techtarget.com/searchdatamanagement/definition/entity-relationship-diagram-ERD#:~:text=An%20entity%20relationship%20diagram%20(ERD,information%20technology%20(IT)%20system.
[Accessed 12 December 2022].

Richard, P., 2022. *GURU99.* [Online]
Available at: https://www.guru99.com/database-normalization.html
[Accessed 10 December 2022].

techopedia, 2014. *Entity.* [Online]
Available at: entity
[Accessed 4 January 2023].

11. Appendix

Originality report

COURSE NAME
CC5051NI - Databases

STUDENT NAME
Suman K.C. Computing

FILE NAME
22015791 SUMAN K.C.

REPORT CREATED
5 Jan 2023

Summary

Flagged passages	2	1%
Cited/quoted passages	6	2%

Web matches

techtarget.com	2	1%
studocu.com	1	0.7%
medium.com	2	0.7%
guru99.com	1	0.2%
coursehero.com	1	0.2%
scribd.com	1	0.1%

1 of 8 passages

Student passage FLAGGED

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**...

Top web match

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**...

New Microsoft Word Document - Module Code &amp - StuDocu https://www.studocu.com/en-gb/document/city-and-islington-college/professional-development-for-computing/new-microsoft-word-document/32494005

2 of 8 passages

Student passage CITED

An entity relationship diagram (ERD), also known as entity relationship model, is a graphical representation among people, object, **places, concepts or events within an information technology (IT)**…

Top web match

An entity relationship diagram (ERD), also known as an **entity relationship model, is a graphical representation** that depicts relationships **among people**, objects, **places, concepts or events within an**…

What is Entity Relationship Diagram (ERD)? - TechTarget https://www.techtarget.com/searchdatamanagement/definition/entity-relationship-diagram-ERD

---

3 of 8 passages

Student passage QUOTED

A **chasm** trap occurs when two many-to-one relationships converge on a single table **in** a database design **and a fan trap occurs when two** many-to-one relationships **follow one another in a**

Top web match

Describe how fan and **chasm** traps can occur **in** an ER model **and** how they can beresolved?Ans: **A fan trap occurs when two** “many-to-one” joins **follow one another in** primary-detail form (OrderDetails), and…

PrabhuBartaula_1002057671_C... https://www.coursehero.com/file/96945747/PrabhuBartaula-1002057671-CC204n-Midtermdocx/

---

4 of 8 passages

Student passage FLAGGED

NormalizationNormalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. **Normalization rules**…

Top web match

Normalization in DBMS **Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies**. **Normalization**…

Normalization in DBMS - Nishajha https://nisha-jha.medium.com/normalization-in-dbms-7d207d6ea662

---

5 of 8 passages

Student passage CITED

…tables into smaller tables and links them using relationships. **The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored** logically (Richard, 2022).

Top web match

The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

What is Normalization in DBMS (SQL)? 1NF, 2NF, 3NF Example https://www.guru99.com/database-normalization.html

---

6 of 8 passages

Student passage QUOTED

…create a user called "coursework_suman" and grant it permissions. **Then, we connect to the new user**. After that, we begin the process of creating tables.

Top web match

Select SQL Server Authentication and enter Login Name and Password. CONNECT TO SQL SERVER. **Then we connect to the new user** that is LOG_TBL.

GRANT And REVOKE In SQL Server - Vaishali Goilkar - Medium https://vaishaligoilkar3322.medium.com/grant-and-revoke-in-sql-server-62ef393e743

---

7 of 8 passages

Student passage QUOTED

…After that, we begin the process of creating tables. **We use the CREATE TABLE command to create tables**. We use datatypes such as varchar2, int, decimal, and…

Top web match

We use the CREATE TABLE command to create tables. For a basic database for an online store, we might have tables for customers, products, orders, product reviews,

Getting Started With MariaDB - Second Edition - Sample Chapter https://www.scribd.com/document/268933355/Getting-Started-with-MariaDB-Second-Edition-Sample-Chapter

---

8 of 8 passages

Student passage CITED

A quey **is a question or a request expressed in a formal manner**. **In computer science, a query is** an **essentially the same thing, the only difference is the answer or retrieved information comes from a**

Top web match

A query **is a question or a request** for information **expressed in a formal manner**. **In computer science, a query is essentially the same thing, the only difference is the answer or retrieved information**…

What is a database query? - TechTarget https://www.techtarget.com/searchdatamanagement/definition/query

---