LAPTOP ORDERING AND SELLING DATABASE SYSTEM

A report submitted to the

Department of Electrical and Information Engineering

Faculty of Engineering

University of Ruhuna

Sri Lanka

On 11th of September 2023

In completing the individual mini for the module

EE 4202 Database Systems

By

Dissanayake D.M.M.I.T -EG/2020/3912

Dissanayake P.K -EG/2020/3916

Dissanayake D.K.R.C.K. -EG/2020/3910

Abstract

Database Management System is use for managing all the data in proper manner. It allows all CRUD operations of the data (Create, Read, Update, Delete). It acts as an interface between the database and end user for implementation and the easy access of the data.

In this report it consists of all the procedure of creating Database Management System suing MySQL for a laptop import and export system.

Preface

A good education can change anyone. A good teacher can change everything. However, one cannot acquire knowledge without the kind guidance of gurus.

There are many people who helped us to complete this report successfully. First of all, we would like to express our gratitude to the Department of Electrical and Information Engineering, Faculty of Engineering, Ruhuna University, for preparing such a module for undergraduate students. Also, I would like to express my gratitude to Dr. Nadeesha Sandamali, module coordinator of EE4202 Database Systems. Without her excellent guidance and excellent supervision this report would have failed. We have learned a lot from her excellent teaching and due to these facts, this project has been successful. Last but not the least, our sincere gratitude goes to our loved ones who were behind the success of our education.

Table of contents

List of figures

1 Introduction

In today's fast-paced digital landscape, businesses involved in the laptop import and export industry face a number of challenges. A systematic and efficient approach to managing extensive data related to laptops, customers, orders, warranties and more. This report serves as a detailed documentation of our journey to develop a robust database management system (DBMS) using MySQL specifically tailored for laptop ordering and selling operations.

The laptop industry has experienced exponential growth in recent years, driven by rapid technological advancements and rising global demand for portable computing devices. In this dynamic environment, the need for streamlined data management has never been more critical. Our database system aims to meet these needs by providing a structured, scalable and user-friendly platform for handling the complex web of information associated with laptop transactions.

Throughout this report, we aim to provide a comprehensive account of our efforts in creating a powerful tool for the laptop ordering and selling industry. Our MySQL-based DBMS represents an important milestone in data management, simplifying complex operations and contributing to the success of businesses in this dynamic sector.

This report serves as a valuable resource for anyone interested in understanding the intricacies of developing a database system that fits the unique needs of the laptop import and export industry. We believe the insights and methodologies presented here will empower organizations to harness the full potential of their data, increasing their competitiveness in this ever-evolving marketplace.

2 CHAPTER 1: RQUIREMENT ANALYSIS

2.1 Requirement analysis

For the database design for laptop import export system, first it was identified all required attributes, entities and relationship with conceptual database model. After that conceptual model was normalized and physically implemented into the DBMS using MySQL.

Requirement analysis can be divided as functional requirements and data requirements.

2.1.1 Functional requirements

The actions that database system is capable of doing. At the end of this project the target is to make the ability of the database to follow the given tasks. This allocates performances and other limiting requirements to all functional levels.

- Should have user friendly interface.
- User should be able to upload data directly to the database.
- Easy data retrieval of customer details, orders, shipping details, warrenty, reviews, laptops, seller details, payment details and the invoices.
- User must be able to view previous point sources.
- Order entity should have direct relationships with customer, invoice, laptop and shipping entities.
- Previously added data should be able to modify, update and delete.
- Missing data should be indicated or filled accordingly.
- Regular data backups should be there to avoid data losses.

2.1.2 Data Requirements

Entity	Attributes
Customer	Address Phone_No Customer_ID Customer_Name Email
Review	Rating Comments Review_Date Review_ID
Payment	Payment_Method Payment_ID Payment_Date Amount
Invoice	Invoice_No Total_Amount Date Order_ID
Order	Order_ID Type Quantity
Shipping	Shipping_ID Shipping_Date Shipping_Address Traching_No Shipping_Type
Laptop	Model Brand Laptop_ID Price Specification
Warranty	Starting_Date Warrenty_ID Laptop_ID End_Date Duration
Seller	Seller_Name Seller_ID Seller_Email Company Seller_PhoneNo

3 CHAPTER 2: CONCEPTUAL DESIGN

3.1 Conceptual Design

This conceptual data model has been designed using the visual paradigm tool to represent the all attributes and entities and relationships of the database.

3.1.1 ER Diagram

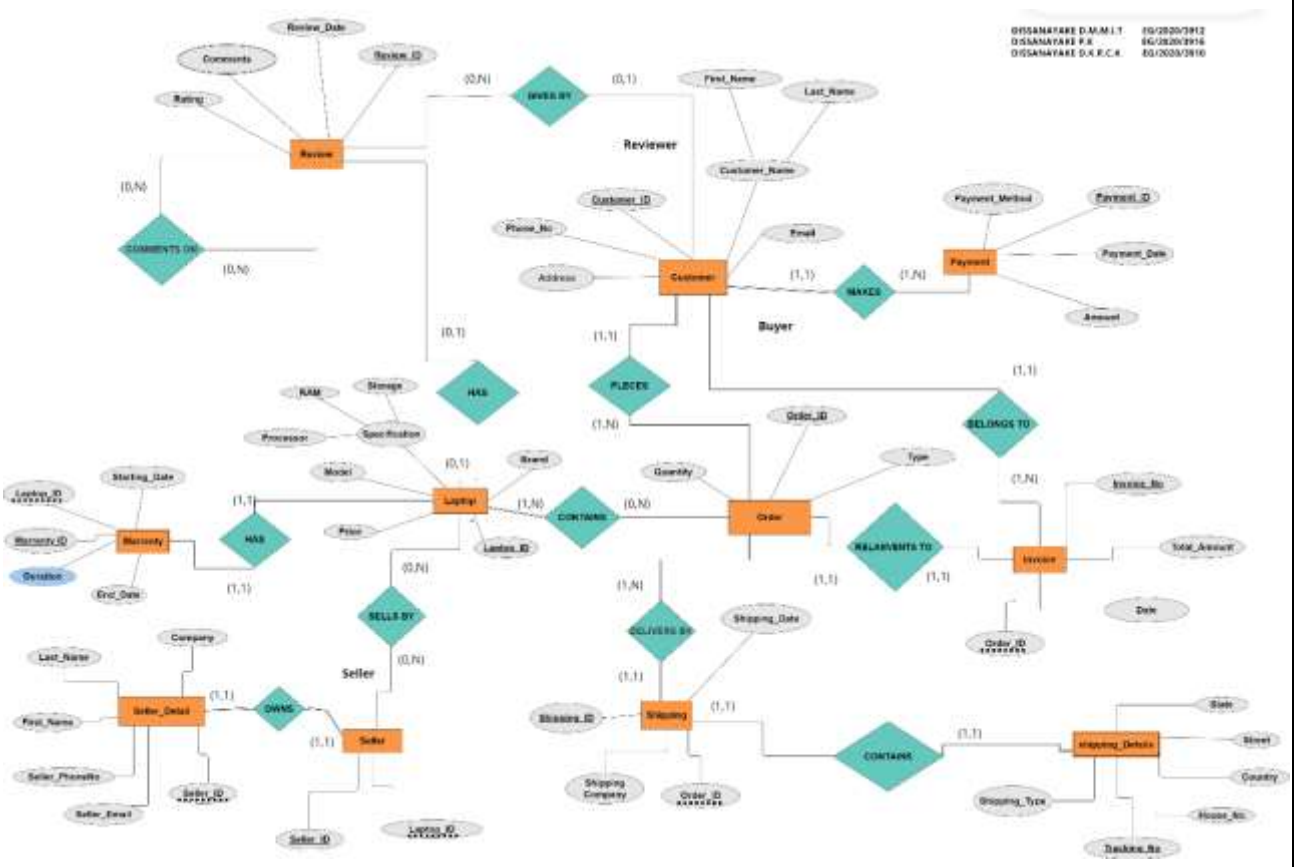


FIGURE 1 ER MODEL

3.1.2 UML MODEL

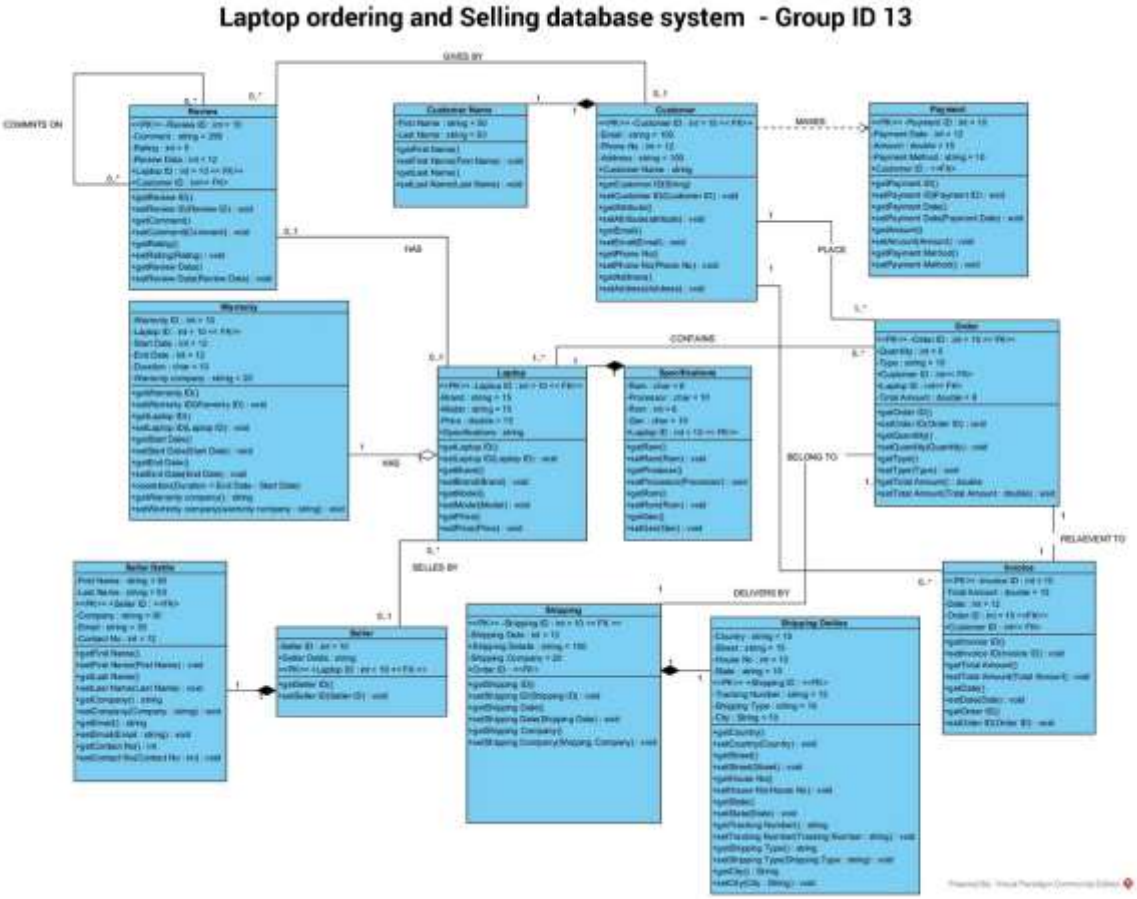


FIGURE 2 UML MODEL

4 CHAPTER 3:IMPLEMETATION

4.1 Normalization

1. 1st normal form

In the first normal form, Composite attributes were taken separately and added to the table as separate columns. As an example, In the Customer table, Name is a composite key with First_Name and Last_Name attributes. It was added to the table as separate columns like below.

Customer_ID	First_name	Last_name	Address_Line1	Address_Line2	City	Email	Phone_no
1	John	Doe	123 Main Street	Apt 4B	New York	john@example.com	555-123-4567
2	Jane	Smith	456 Elm St	NULL	Los Angeles	jane@example.com	555-987-6543
3	Bob	Johnson	789 Oak St	NULL	Chicago	bob@example.com	555-567-8901
4	Alice	Brown	321 Elm St	Apt 2C	San Francisco	alice@example.com	555-789-0123
5	Eva	Williams	555 Pine St	NULL	Houston	eva@example.com	555-234-5678
6	David	Lee	888 Oak St	Apt 3D	Miami	david@example.com	555-876-5432

FIGURE 3 1ST NORMAL FORM

2. 2nd normal form

In the second normal form there shouldn't be any partial dependencies. Therefore tables with partial dependencies were broken down into two tables as no any prime attribute make functional dependency with a non prime attribute. After that added foreign key to add the connection between two tables.

4.2 MySQL Implementation

4.2.1 Creating the schema (COMPUTERS)

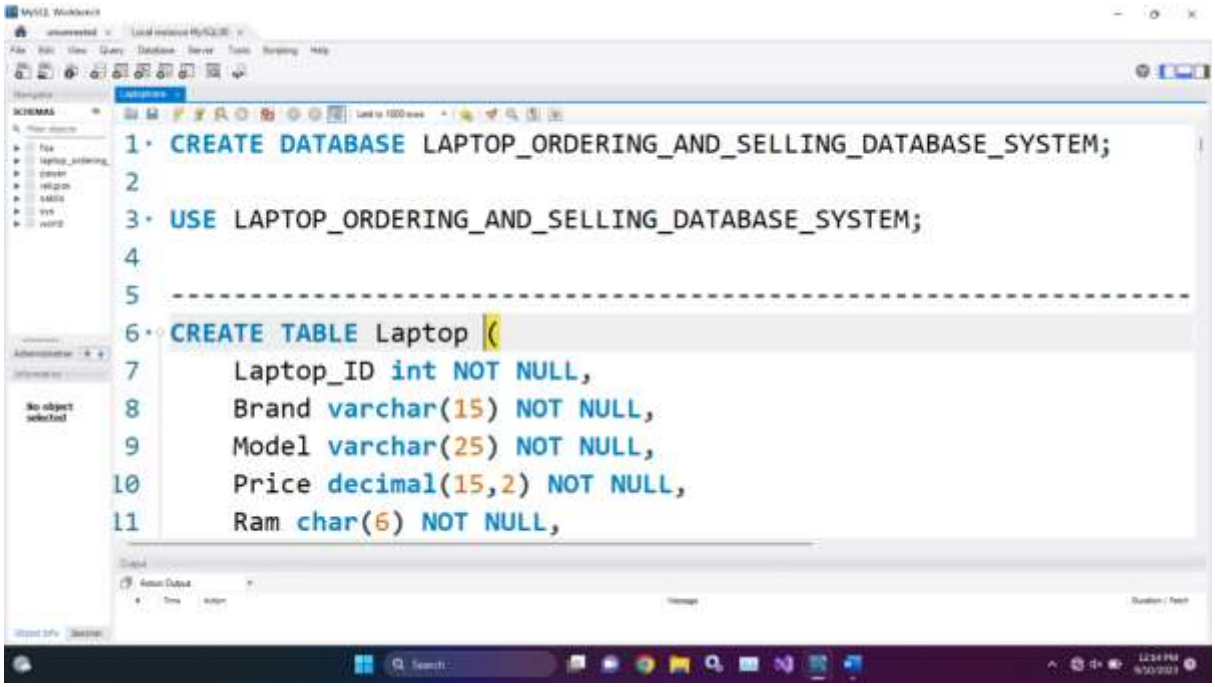
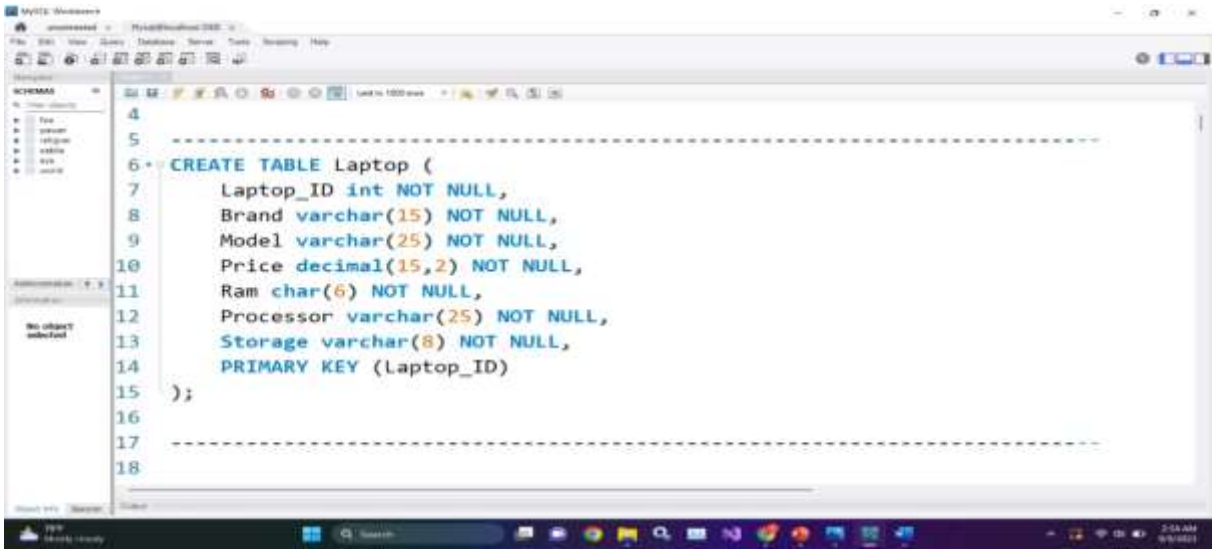


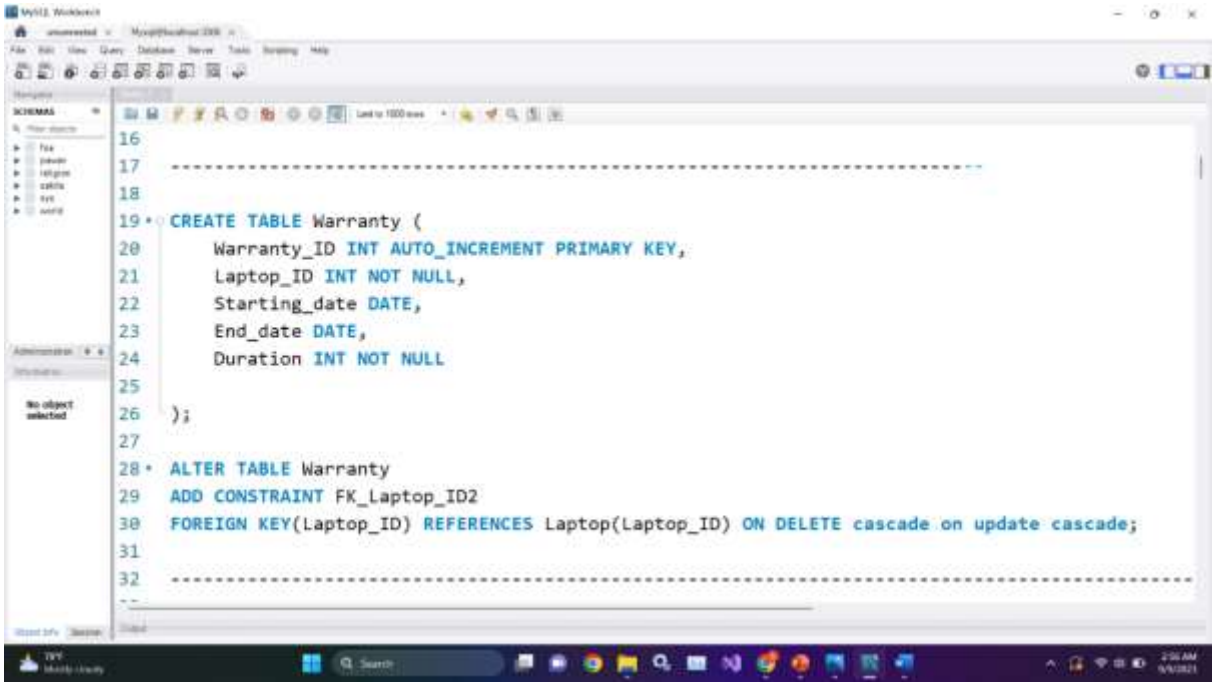
FIGURE 4 DATABASE CREATION

4.2.2 Table Creation



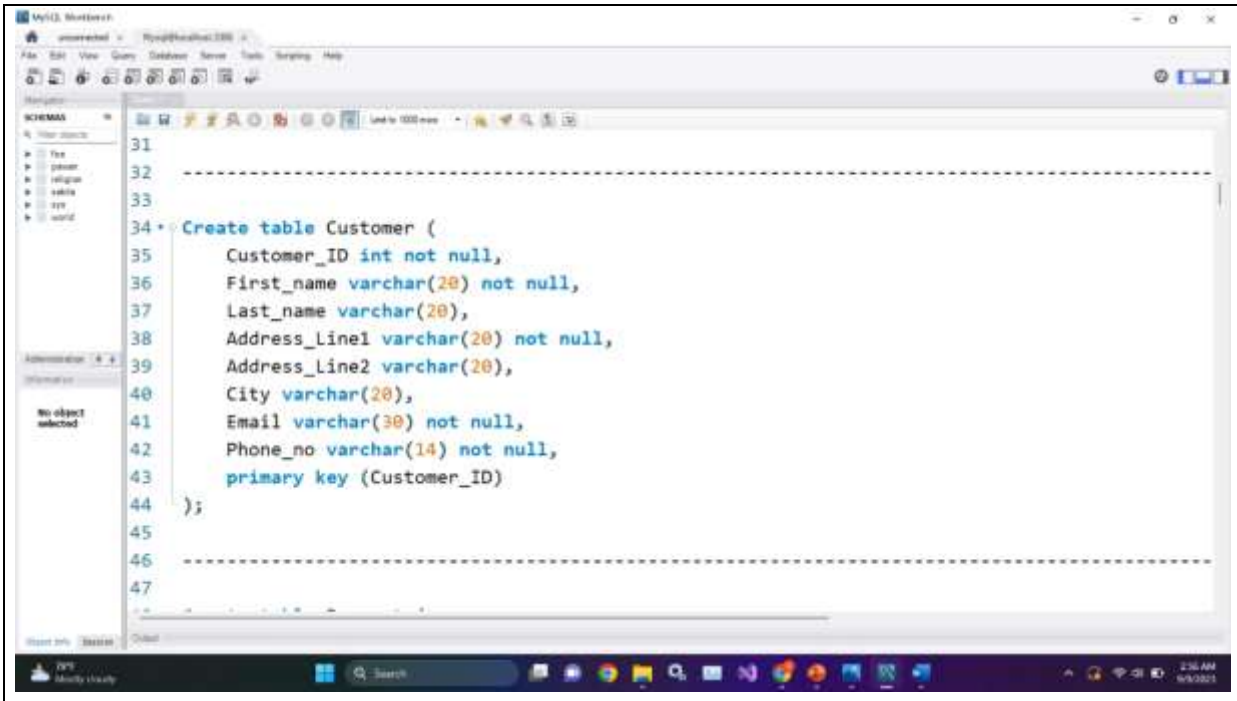
```
---------------------------------------------------------------------------
CREATE TABLE Laptop (
    Laptop_ID int NOT NULL,
    Brand varchar(15) NOT NULL,
    Model varchar(25) NOT NULL,
    Price decimal(15,2) NOT NULL,
    Ram char(6) NOT NULL,
    Processor varchar(25) NOT NULL,
    Storage varchar(8) NOT NULL,
    PRIMARY KEY (Laptop_ID)
);

---------------------------------------------------------------------------
```

FIGURE 5 CREATING LAPTOP TABLE



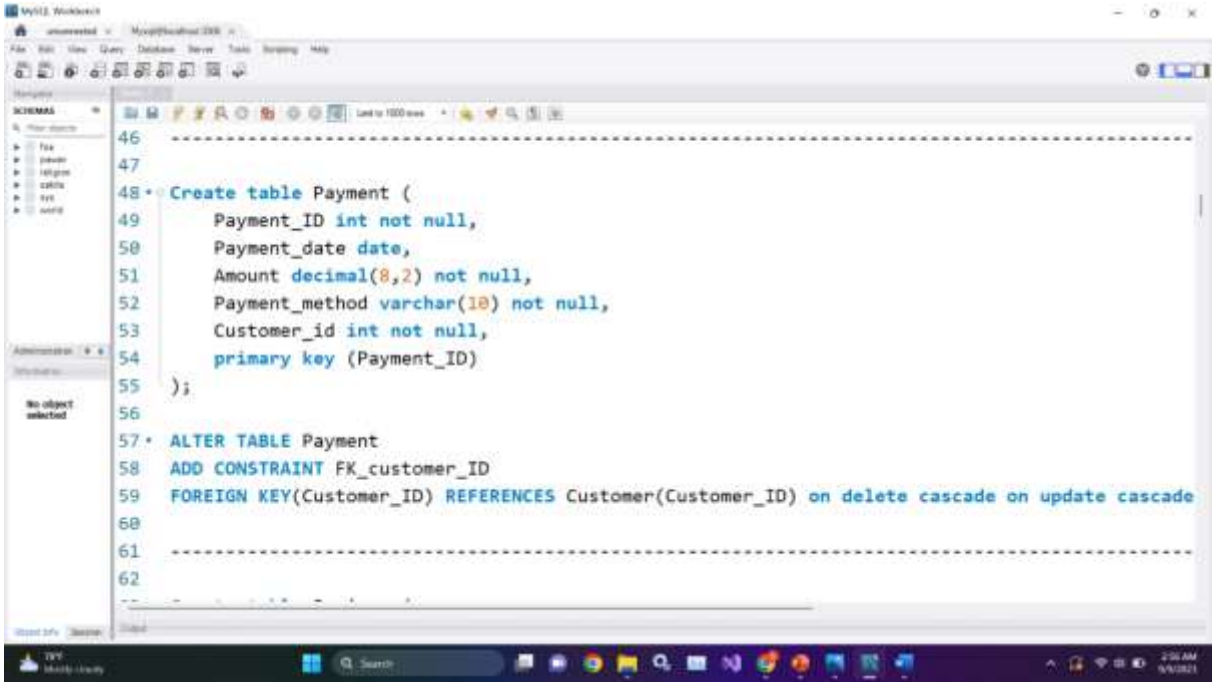
```
---------------------------------------------------------------------------

CREATE TABLE Warranty (
    Warranty_ID INT AUTO_INCREMENT PRIMARY KEY,
    Laptop_ID INT NOT NULL,
    Starting_date DATE,
    End_date DATE,
    Duration INT NOT NULL

);

ALTER TABLE Warranty
ADD CONSTRAINT FK_Laptop_ID2
FOREIGN KEY(Laptop_ID) REFERENCES Laptop(Laptop_ID) ON DELETE cascade on update cascade;

---------------------------------------------------------------------------
```

FIGURE 6 WARRANTY TABLE



```
---------------------------------------------------------------------------------------------

Create table Customer (
    Customer_ID int not null,
    First_name varchar(20) not null,
    Last_name varchar(20),
    Address_Line1 varchar(20) not null,
    Address_Line2 varchar(20),
    City varchar(20),
    Email varchar(30) not null,
    Phone_no varchar(14) not null,
    primary key (Customer_ID)
);

---------------------------------------------------------------------------------------------
```

FIGURE 7 CUSTOMER TABLE



```
---------------------------------------------------------------------------------------------

Create table Payment (
    Payment_ID int not null,
    Payment_date date,
    Amount decimal(8,2) not null,
    Payment_method varchar(10) not null,
    Customer_id int not null,
    primary key (Payment_ID)
);

ALTER TABLE Payment
ADD CONSTRAINT FK_customer_ID
FOREIGN KEY(Customer_ID) REFERENCES Customer(Customer_ID) on delete cascade on update cascade

---------------------------------------------------------------------------------------------
```

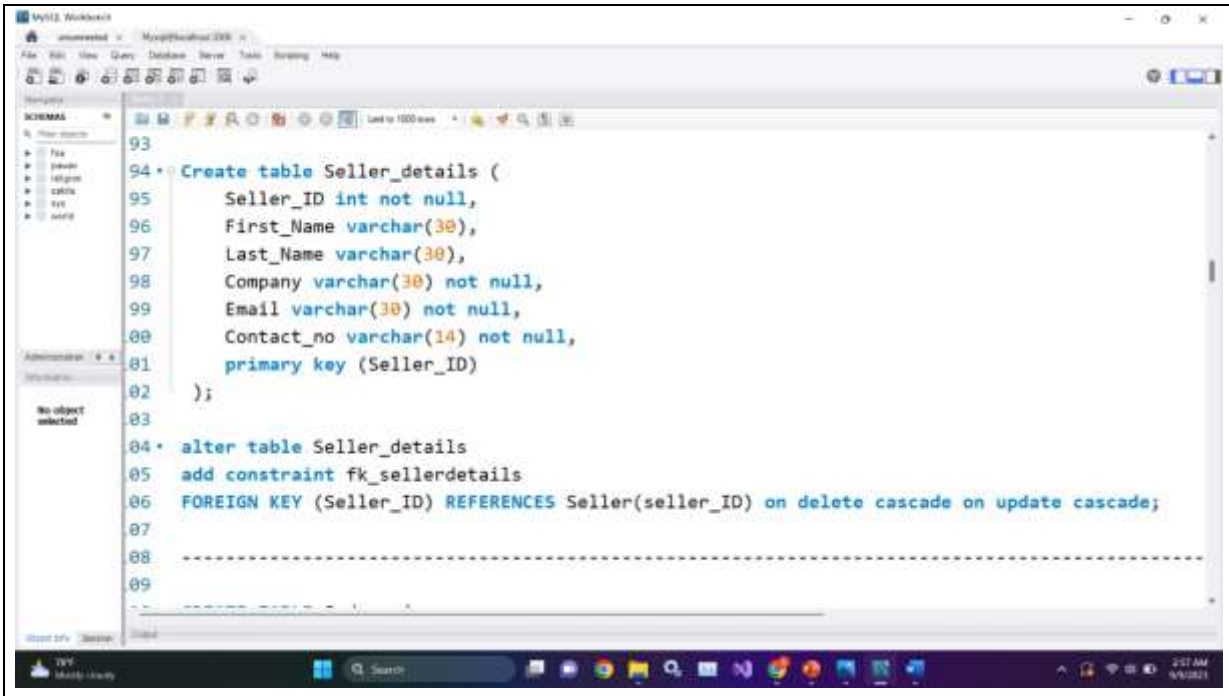
FIGURE 8 PAYMENT TABLE

```
Create table Reviews (
    Review_ID int not null,
    Comments varchar(256),
    Rating int,
    Review_date date,
    Laptop_ID int not null,
    Customer_ID int not null,

    primary key (Review_ID)
);

ALTER TABLE reviews
ADD CONSTRAINT fk_laptopid
FOREIGN KEY (Laptop_id) REFERENCES Laptop (Laptop_id) ON DELETE cascade on update cascade,
ADD CONSTRAINT fk_customer
FOREIGN KEY (customer_id)REFERENCES customer(customer_id) ON delete cascade on update cascade;
```

FIGURE 9 REVIEWS TABLE

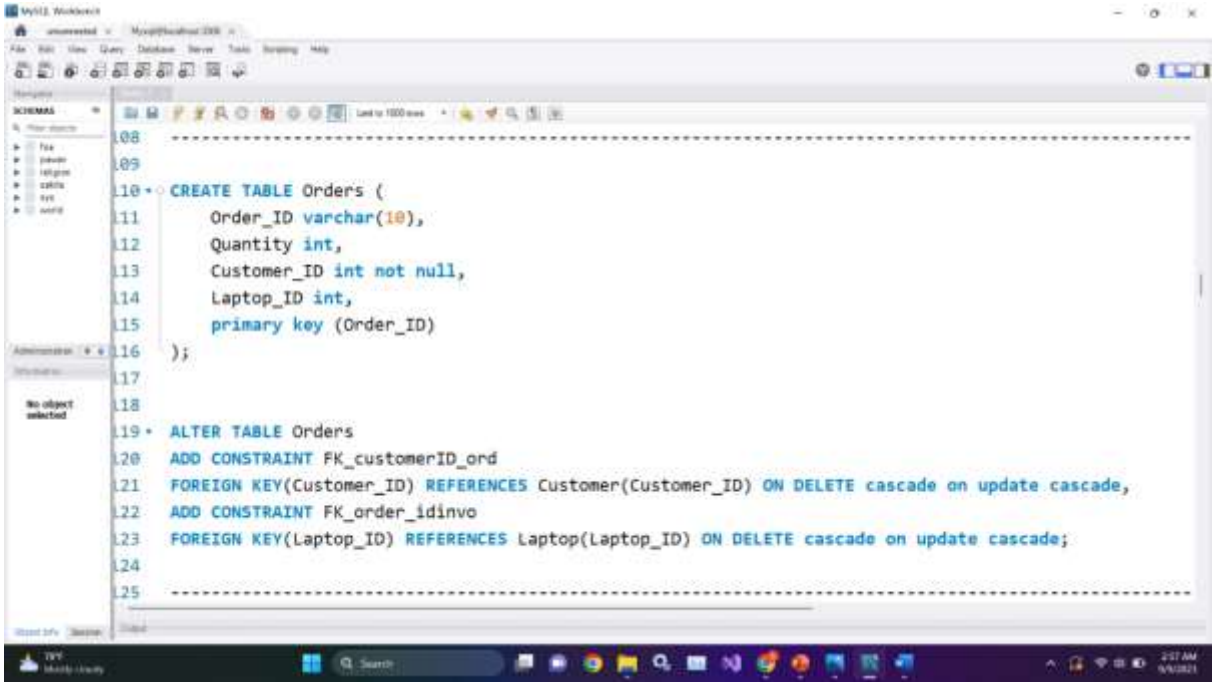
```
FOREIGN KEY (Laptop_id) REFERENCES Laptop (Laptop_id) ON DELETE cascade on update cascade,
ADD CONSTRAINT fk_customer
FOREIGN KEY (customer_id)REFERENCES customer(customer_id) ON delete cascade on update cascade;

--------------------------------------------------------------------------------------------------

Create table Seller (
    laptop_ID int not null,
    seller_ID int not null,
    primary key (laptop_ID)
);

ALTER TABLE Seller
ADD CONSTRAINT fk_seller
FOREIGN KEY (Laptop_id) REFERENCES Laptop (Laptop_id) ON DELETE cascade on update cascade;
alter table Seller add constraint uniq unique(Seller_ID);
--------------------------------------------------------------------------------------------------
```

FIGURE 10 SELLER TABLE



```
Create table Seller_details (
    Seller_ID int not null,
    First_Name varchar(30),
    Last_Name varchar(30),
    Company varchar(30) not null,
    Email varchar(30) not null,
    Contact_no varchar(14) not null,
    primary key (Seller_ID)
 );

alter table Seller_details
add constraint fk_sellerdetails
FOREIGN KEY (Seller_ID) REFERENCES Seller(seller_ID) on delete cascade on update cascade;

-----------------------------------------------------------------------------------------
```

FIGURE 11 :SELLER_DETAILS TABLE



```
-----------------------------------------------------------------------------------------

CREATE TABLE Orders (
    Order_ID varchar(10),
    Quantity int,
    Customer_ID int not null,
    Laptop_ID int,
    primary key (Order_ID)
 );


ALTER TABLE Orders
ADD CONSTRAINT FK_customerID_ord
FOREIGN KEY(Customer_ID) REFERENCES Customer(Customer_ID) ON DELETE cascade on update cascade,
ADD CONSTRAINT FK_order_idinvo
FOREIGN KEY(Laptop_ID) REFERENCES Laptop(Laptop_ID) ON DELETE cascade on update cascade;

-----------------------------------------------------------------------------------------
```

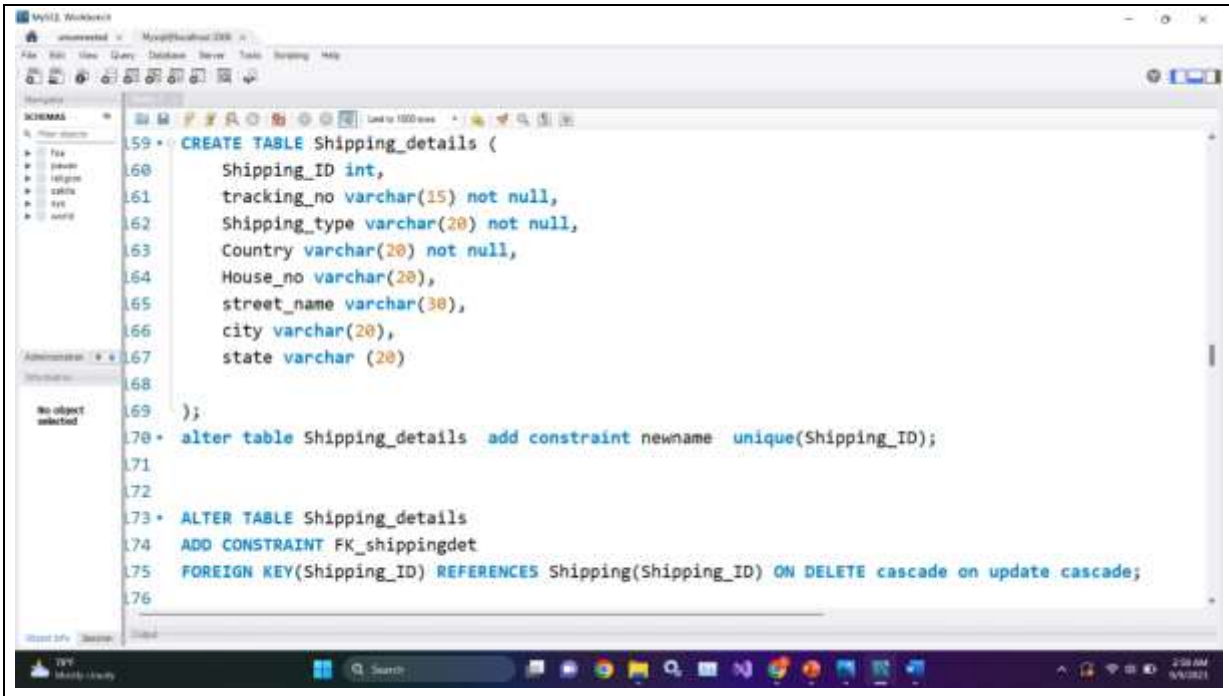
FIGURE 12 ORDERS TABLE

```
CREATE TABLE Invoice (
    Invoice_ID bigint not null,
    Total_Amount decimal(8,2) not null,
    Invoice_Date date,
    Customer_ID int not null,
    Order_ID varchar(10),
    primary key (invoice_ID)
);

ALTER TABLE Invoice
ADD CONSTRAINT FK_customerIDinvo
FOREIGN KEY(Customer_ID) REFERENCES Customer(Customer_ID) ON DELETE no action on update cascade,

ADD CONSTRAINT FK_order_idinvoice
FOREIGN KEY(Order_ID) REFERENCES Orders(Order_ID) ON DELETE cascade on update cascade;

-----------------------------------------------------------------------------------------------
```

FIGURE 13 INVOICE TABLE

```
-----------------------------------------------------------------------------------------------

CREATE TABLE Shipping (
    Shipping_ID int,
    Order_ID varchar(10),
    Shipping_Company varchar(20),
    Shipping_date date,
    primary key (shipping_ID)
);
ALTER TABLE shipping
ADD CONSTRAINT FK_shipping
FOREIGN KEY(Order_ID) REFERENCES Orders(order_ID) ON DELETE cascade on update cascade;


-----------------------------------------------------------------------------------------------

```

FIGURE 14 SHIPPING TABLE



```
CREATE TABLE Shipping_details (
    Shipping_ID int,
    tracking_no varchar(15) not null,
    Shipping_type varchar(20) not null,
    Country varchar(20) not null,
    House_no varchar(20),
    street_name varchar(30),
    city varchar(20),
    state varchar (20)

);
alter table Shipping_details  add constraint newname  unique(Shipping_ID);


ALTER TABLE Shipping_details
ADD CONSTRAINT FK_shippingdet
FOREIGN KEY(Shipping_ID) REFERENCES Shipping(Shipping_ID) ON DELETE cascade on update cascade;
```

FIGURE 15 SHIPPING_DETAILS TABLE

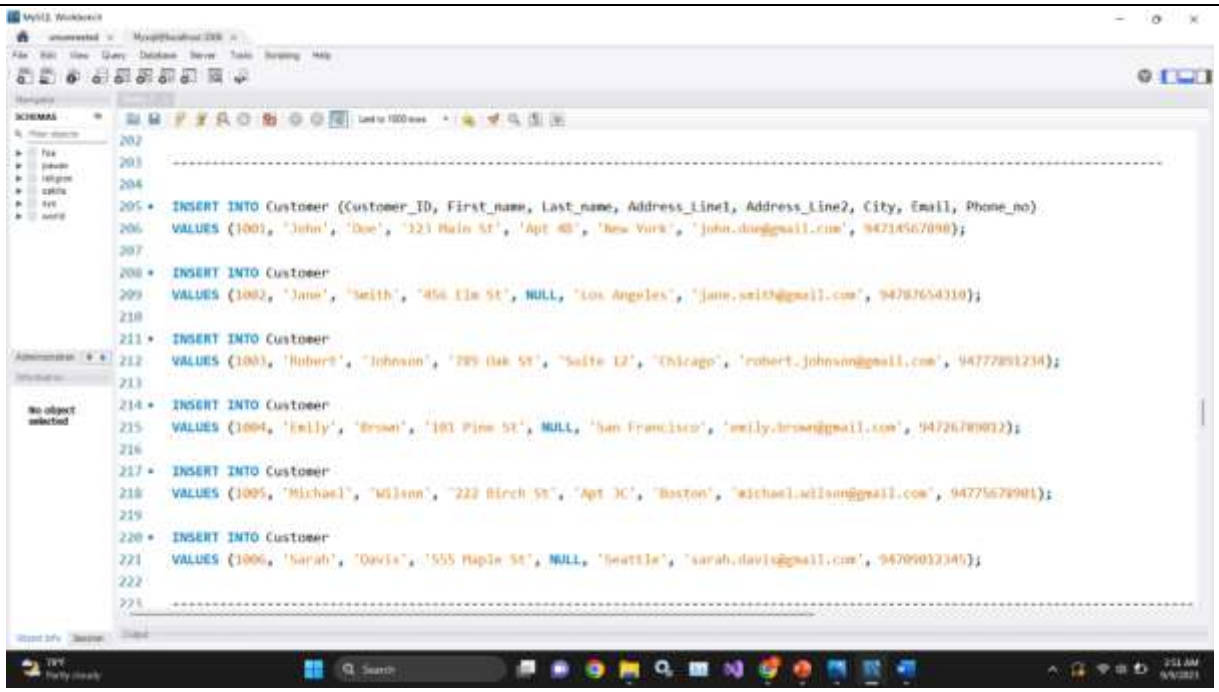
4.2.3 Inserting data into Tables

```
FOREIGN KEY(Shipping_ID) REFERENCES Shipping(Shipping_ID) ON DELETE cascade on update cascade;

-----------------------------------------------------------------------------------------------------

-----------------------------------------------------------------/*insertion*/-----------------------

INSERT INTO Laptop (Laptop_ID, Brand, Model, Price,Ram,Processor,Storage)
VALUES (21001, 'Asus', 'ROG Zephyrus', 455000.00,'8GB','Intel i9 9000','1TB');

INSERT INTO Laptop (Laptop_ID, Brand, Model, Price,Ram,Processor,Storage)
VALUES (21002, 'Dell', 'Inspiron 3532', 255000.00,'8GB','Intel i9 9000','1TB'),
       (21003, 'Dell', 'Inspiron 5532', 305000.00,'8GB','Intel i9 9000','1TB'),
       (21004, 'Huawei', 'Matebook X', 405000.00,'8GB','Intel i9 9000','1TB'),
       (21005, 'Acer', 'Nitro 15', 335000.00,'8GB','Intel i9 9000','1TB'),
       (21006, 'MSI', 'Stealth 15', 505000.00,'8GB','Intel i9 9000','1TB');
```

FIGURE 16 INSERTING INTO THE LAPTOP TABLE

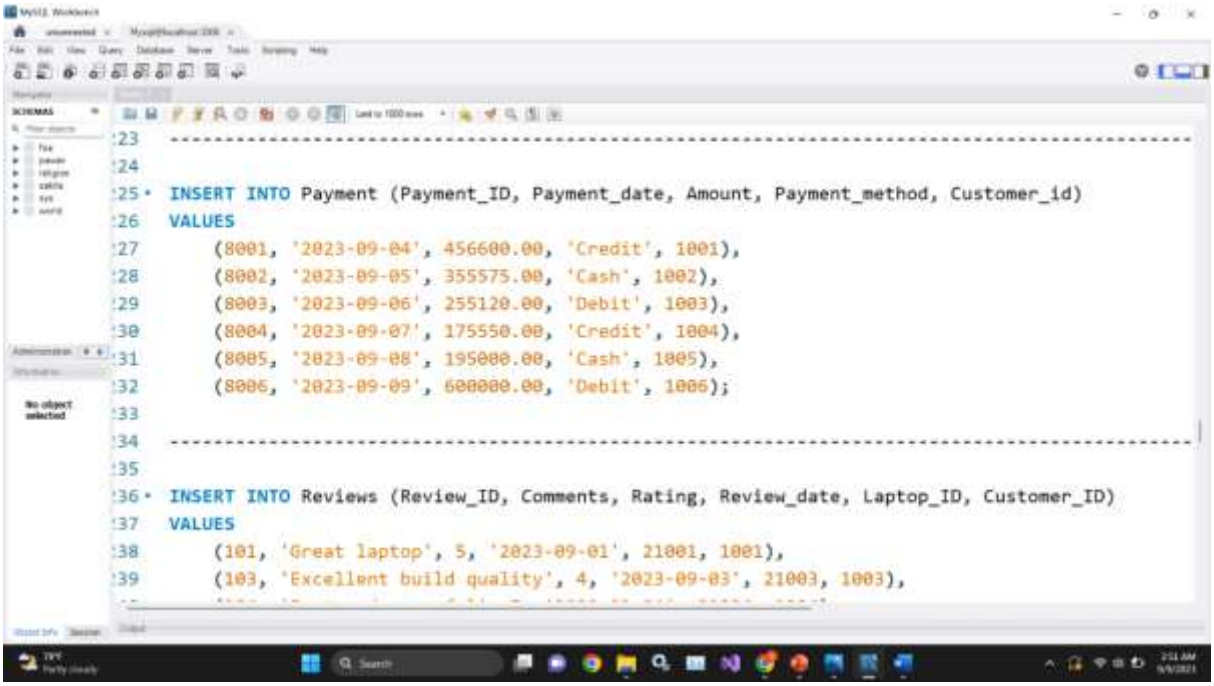
```
       (21003, 'Dell', 'Inspiron 5532', 305000.00,'8GB','Intel i9 9000','1TB'),
       (21004, 'Huawei', 'Matebook X', 405000.00,'8GB','Intel i9 9000','1TB'),
       (21005, 'Acer', 'Nitro 15', 335000.00,'8GB','Intel i9 9000','1TB'),
       (21006, 'MSI', 'Stealth 15', 505000.00,'8GB','Intel i9 9000','1TB');


-----------------------------------------------------------------------------------------------------
INSERT INTO Warranty ( Warranty_ID,Laptop_ID, Starting_date, End_date, Duration)
VALUES
    ( 1,21001, '2023-01-01', '2023-12-31', 12),
    (2, 21002, '2023-02-15', '2025-02-14', 24),
    (3, 21003, '2023-03-10', '2026-03-09', 36),
    (4, 21004, '2023-04-20', '2024-04-19', 12),
    (5,21005, '2023-05-05', '2025-05-04', 24),
    ( 6,21006, '2023-06-30', '2026-06-29', 36);

-----------------------------------------------------------------------------------------------------
```

FIGURE 17 INSERTING INTO THE WARRANTY TABLE



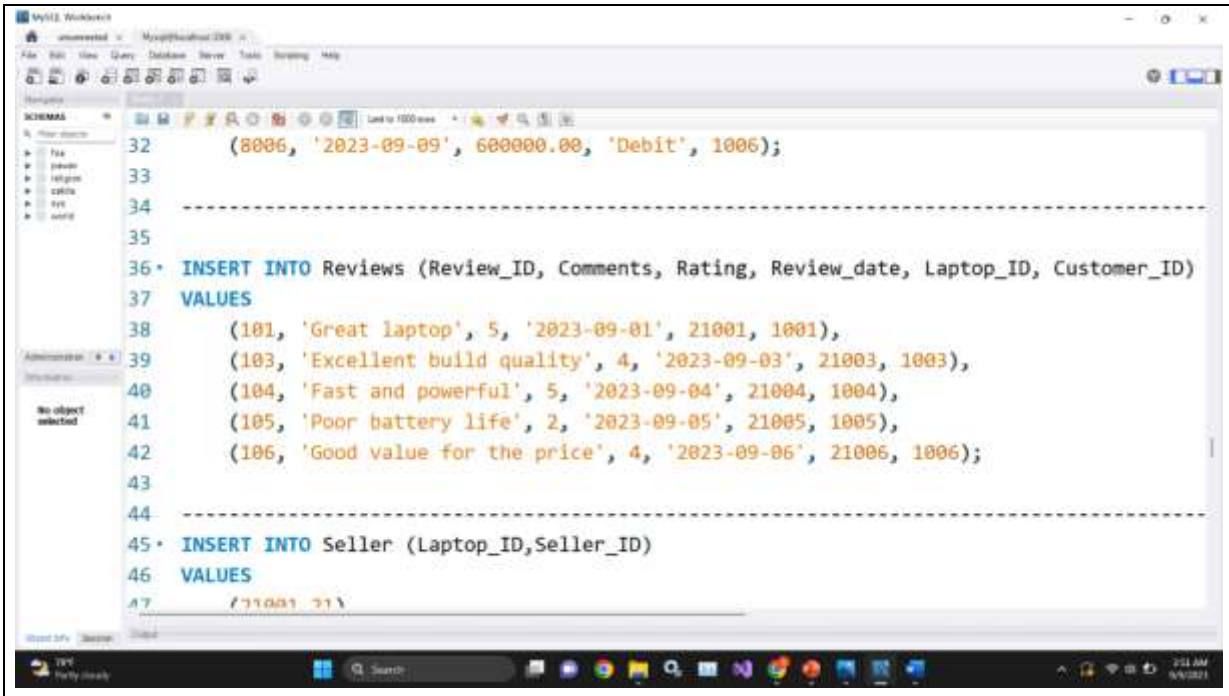
```
INSERT INTO Customer (Customer_ID, First_name, Last_name, Address_Line1, Address_Line2, City, Email, Phone_no)
VALUES (1001, 'John', 'Doe', '121 Main St', 'Apt 4B', 'New York', 'john.doe@gmail.com', 9471456789);

INSERT INTO Customer
VALUES (1002, 'Jane', 'Smith', '456 Elm St', NULL, 'Los Angeles', 'jane.smith@gmail.com', 9478765431);

INSERT INTO Customer
VALUES (1003, 'Robert', 'Johnson', '789 Oak St', 'Suite 12', 'Chicago', 'robert.johnson@gmail.com', 9477789123);

INSERT INTO Customer
VALUES (1004, 'Emily', 'Brown', '101 Pine St', NULL, 'San Francisco', 'emily.brown@gmail.com', 9472678901);

INSERT INTO Customer
VALUES (1005, 'Michael', 'Wilson', '222 Birch St', 'Apt 3C', 'Boston', 'michael.wilson@gmail.com', 9477567890);

INSERT INTO Customer
VALUES (1006, 'Sarah', 'Davis', '555 Maple St', NULL, 'Seattle', 'sarah.davis@gmail.com', 9470901234);
```

FIGURE 18 INSERTING INTO THE CUSTOMER TABLE



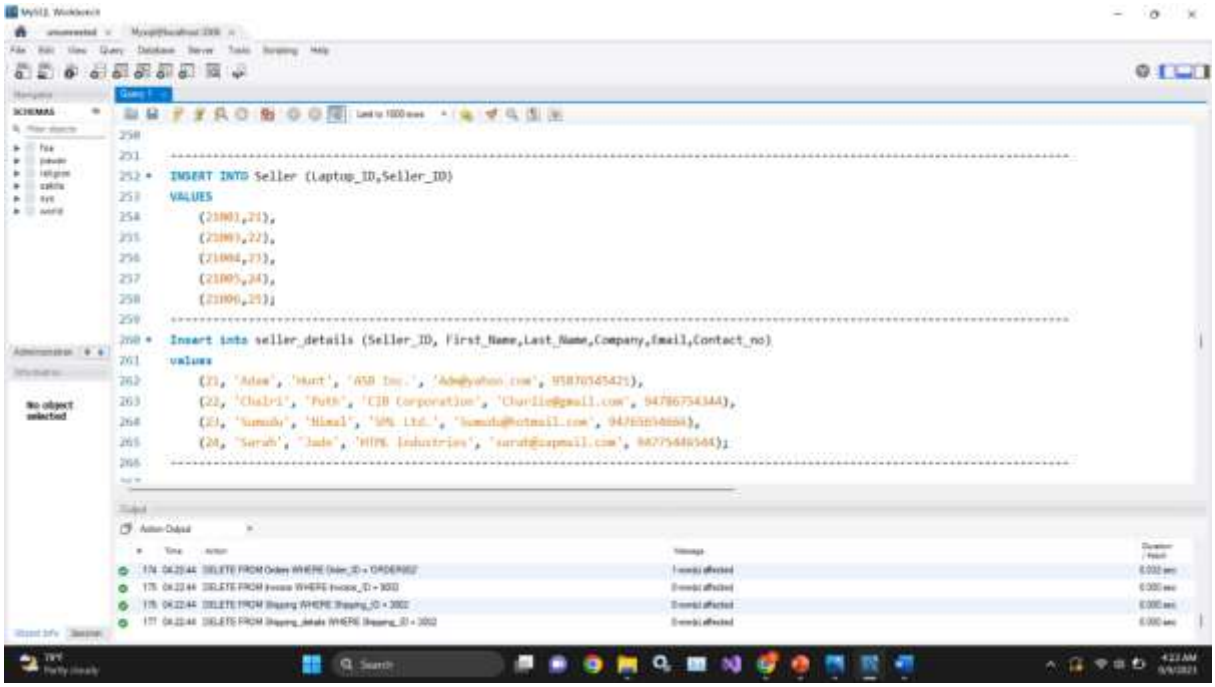
```
INSERT INTO Payment (Payment_ID, Payment_date, Amount, Payment_method, Customer_id)
VALUES
    (8001, '2023-09-04', 456600.00, 'Credit', 1001),
    (8002, '2023-09-05', 355575.00, 'Cash', 1002),
    (8003, '2023-09-06', 255120.00, 'Debit', 1003),
    (8004, '2023-09-07', 175550.00, 'Credit', 1004),
    (8005, '2023-09-08', 195000.00, 'Cash', 1005),
    (8006, '2023-09-09', 600000.00, 'Debit', 1006);

INSERT INTO Reviews (Review_ID, Comments, Rating, Review_date, Laptop_ID, Customer_ID)
VALUES
    (101, 'Great laptop', 5, '2023-09-01', 21001, 1001),
    (103, 'Excellent build quality', 4, '2023-09-03', 21003, 1003),
```

FIGURE 19 INSERTING INTO THE PAYMENT TABLE



```
    (8006, '2023-09-09', 600000.00, 'Debit', 1006);

----------------------------------------------------------------------------------------

INSERT INTO Reviews (Review_ID, Comments, Rating, Review_date, Laptop_ID, Customer_ID)
VALUES
    (101, 'Great laptop', 5, '2023-09-01', 21001, 1001),
    (103, 'Excellent build quality', 4, '2023-09-03', 21003, 1003),
    (104, 'Fast and powerful', 5, '2023-09-04', 21004, 1004),
    (105, 'Poor battery life', 2, '2023-09-05', 21005, 1005),
    (106, 'Good value for the price', 4, '2023-09-06', 21006, 1006);

----------------------------------------------------------------------------------------

INSERT INTO Seller (Laptop_ID,Seller_ID)
VALUES
    (21001,21),
```

FIGURE 20 INSERTING INTO THE REVIEWS TABLE



```
----------------------------------------------------------------------------------------

INSERT INTO Seller (Laptop_ID,Seller_ID)
VALUES
    (21001,21),
    (21003,22),
    (21004,23),
    (21005,24),
    (21006,25);
----------------------------------------------------------------------------------------

Insert into seller_details (Seller_ID, First_Name,Last_Name,Company,Email,Contact_no)
values
    (21, 'Adam', 'Hurt', 'ASD Inc.', 'Adam@yahoo.com', 9587654321),
    (22, 'Charlie', 'Puth', 'CIB Corporation', 'Charlie@gmail.com', 9478675434A),
    (23, 'Sumudu', 'Himal', 'SPL Ltd.', 'Sumudu@hotmail.com', 9476354664),
    (24, 'Sarah', 'Jade', 'HIPK Industries', 'sarah@gmail.com', 9477546564);
----------------------------------------------------------------------------------------
```

FIGURE 21 INSERTING INTO THE SELLER AND SELLER_DETAILS TABLE

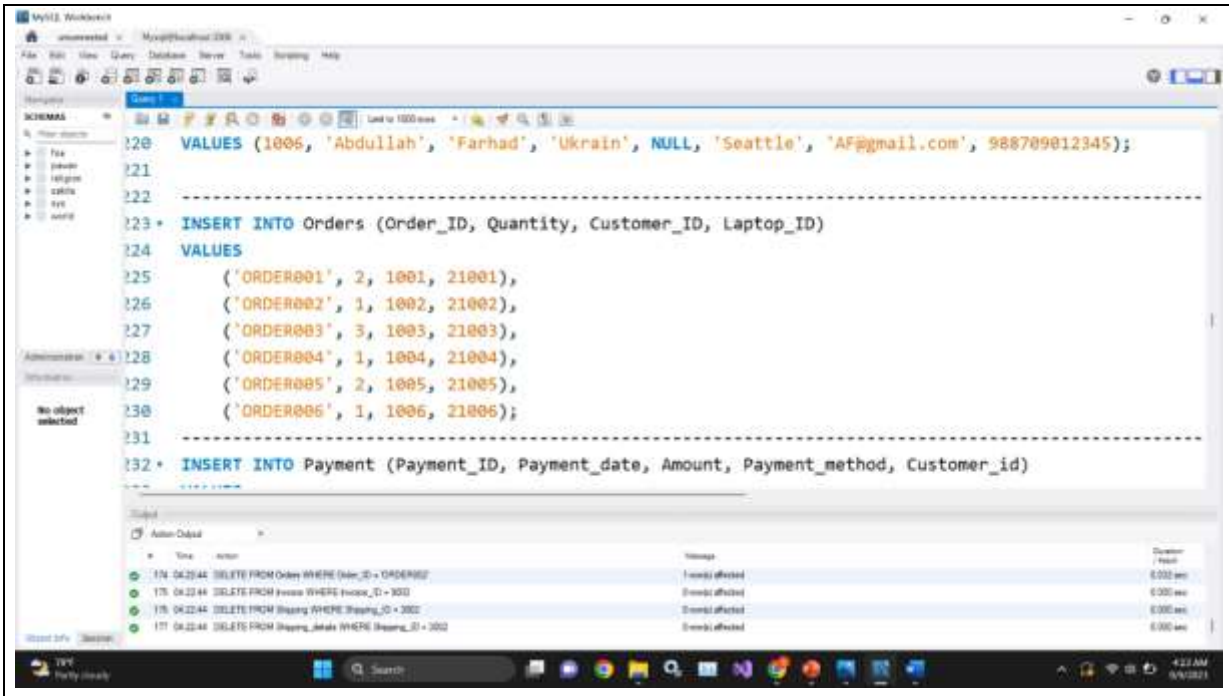


FIGURE 22 INSERTING INTO THE ORERS TABLE

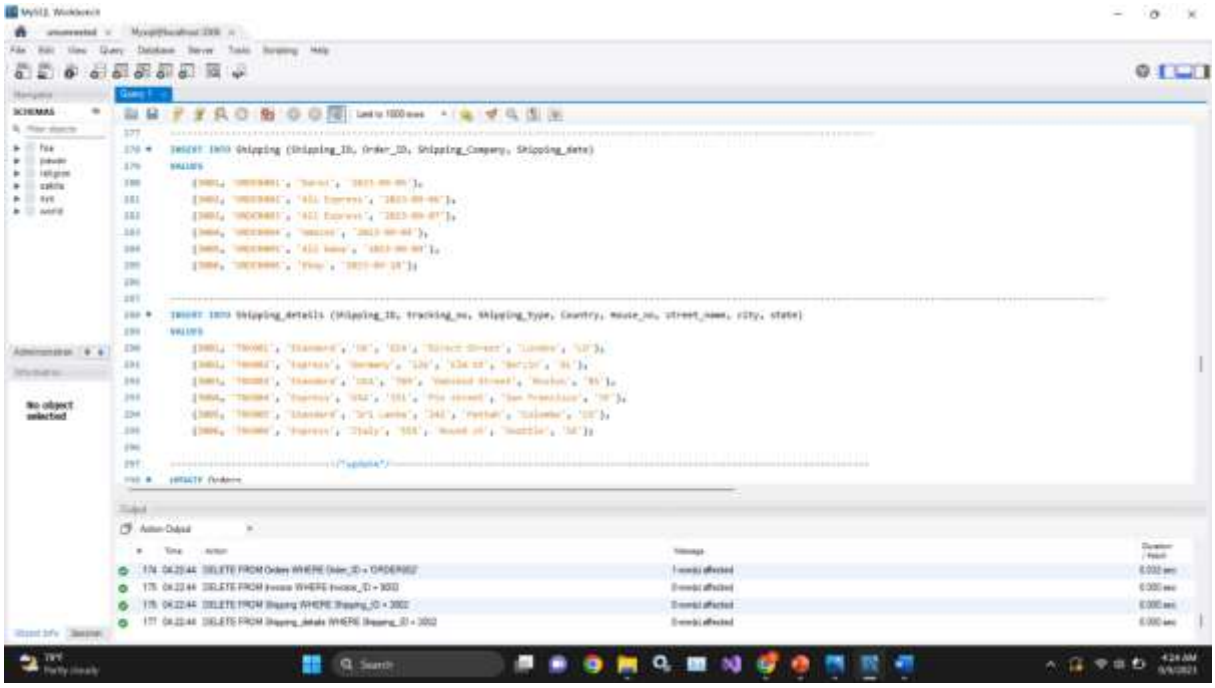


FIGURE 23 INSERTING INTO THE SHIPPING AND SHIPPING_DETAILS TABLE

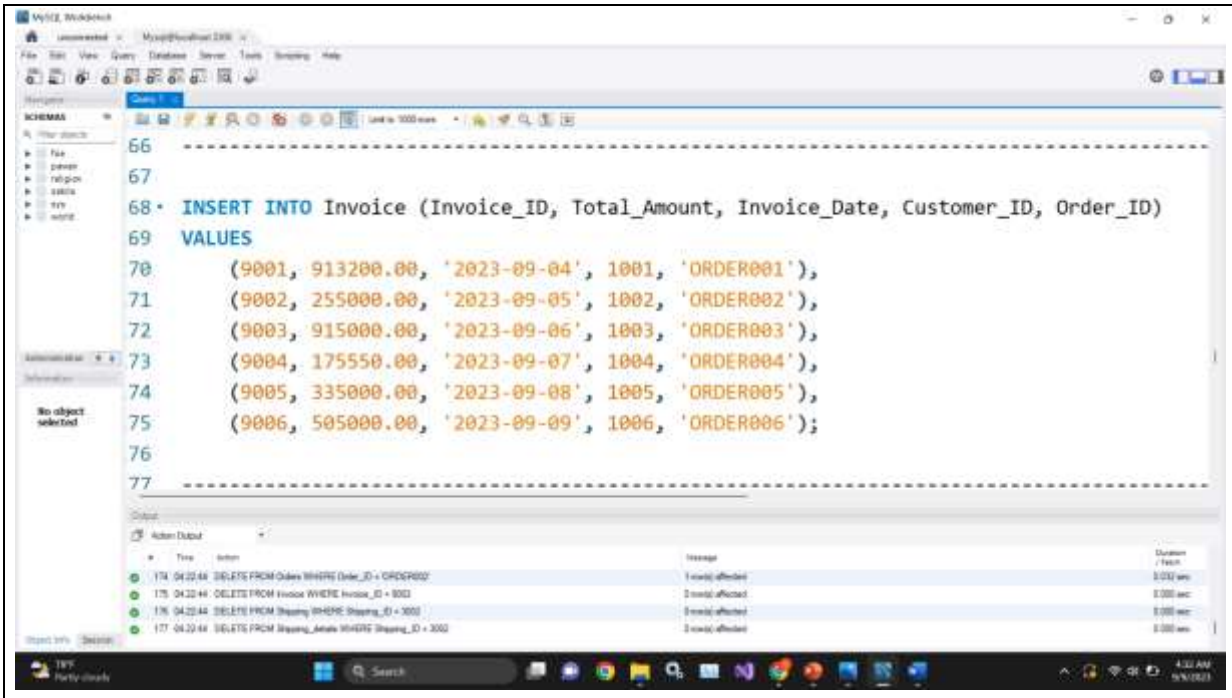


FIGURE 24 INSERTING INTO THE INVOICE TABLE

4.2.4 Update data from the Tables

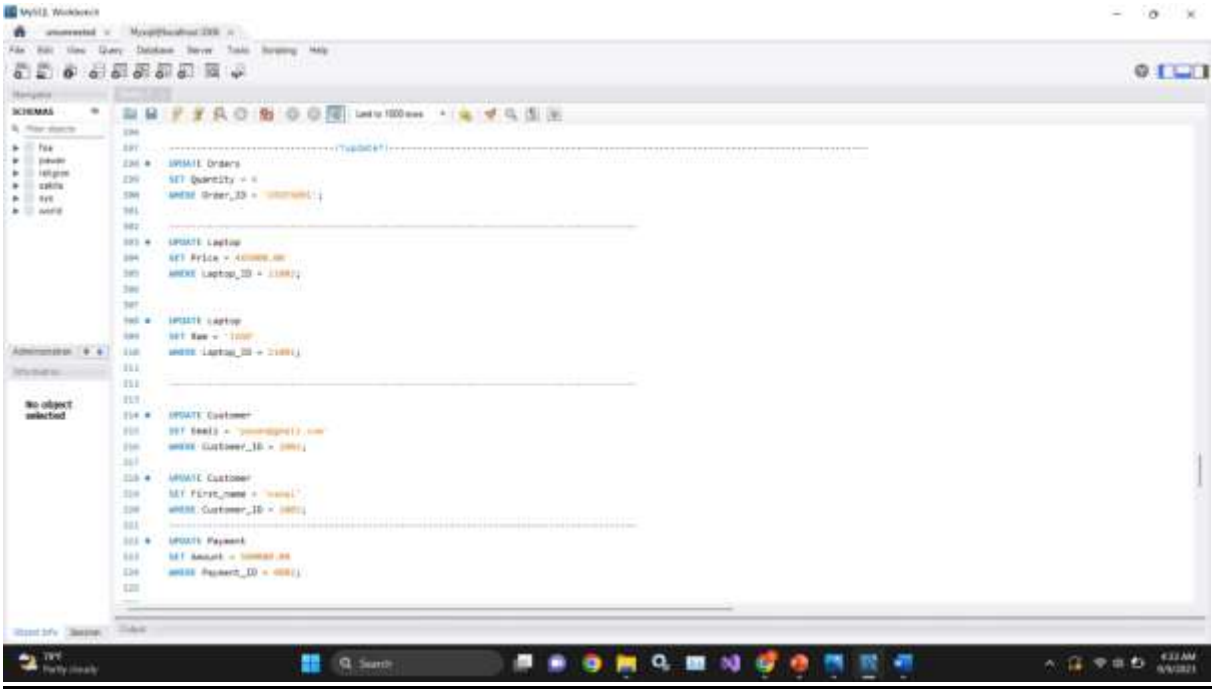


FIGURE 25:UPDATING DATA FROM TABLES-1

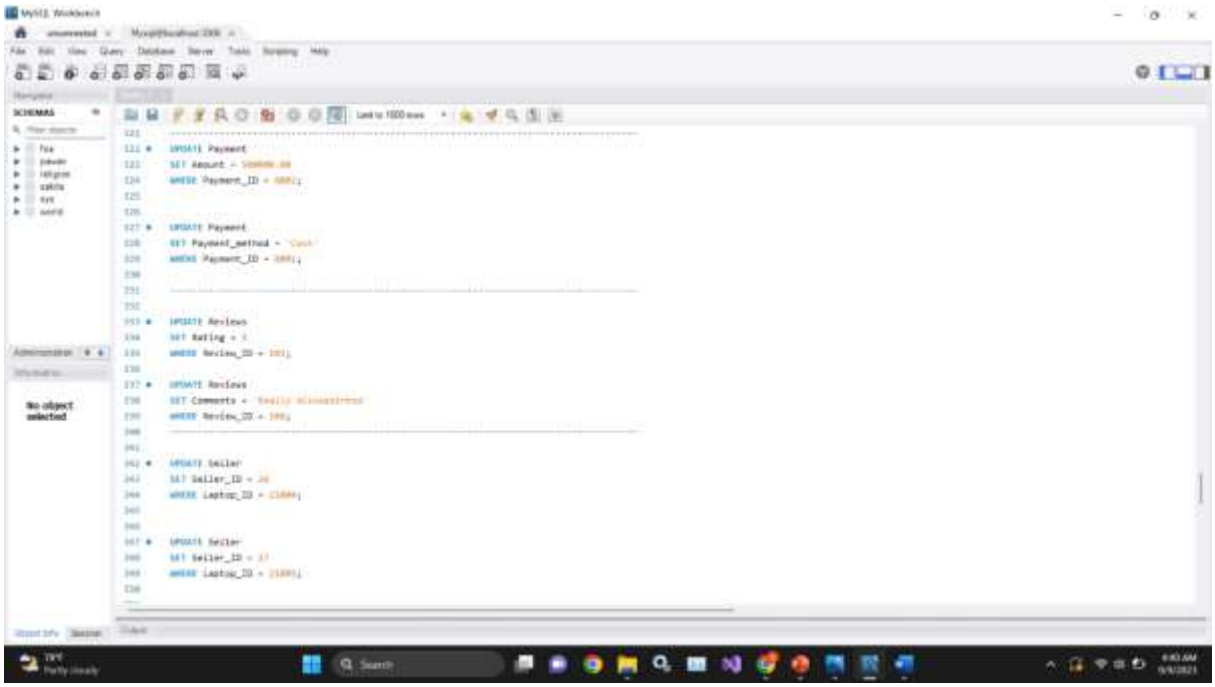


FIGURE 26:UPDATING DATA FROM TABLES-2

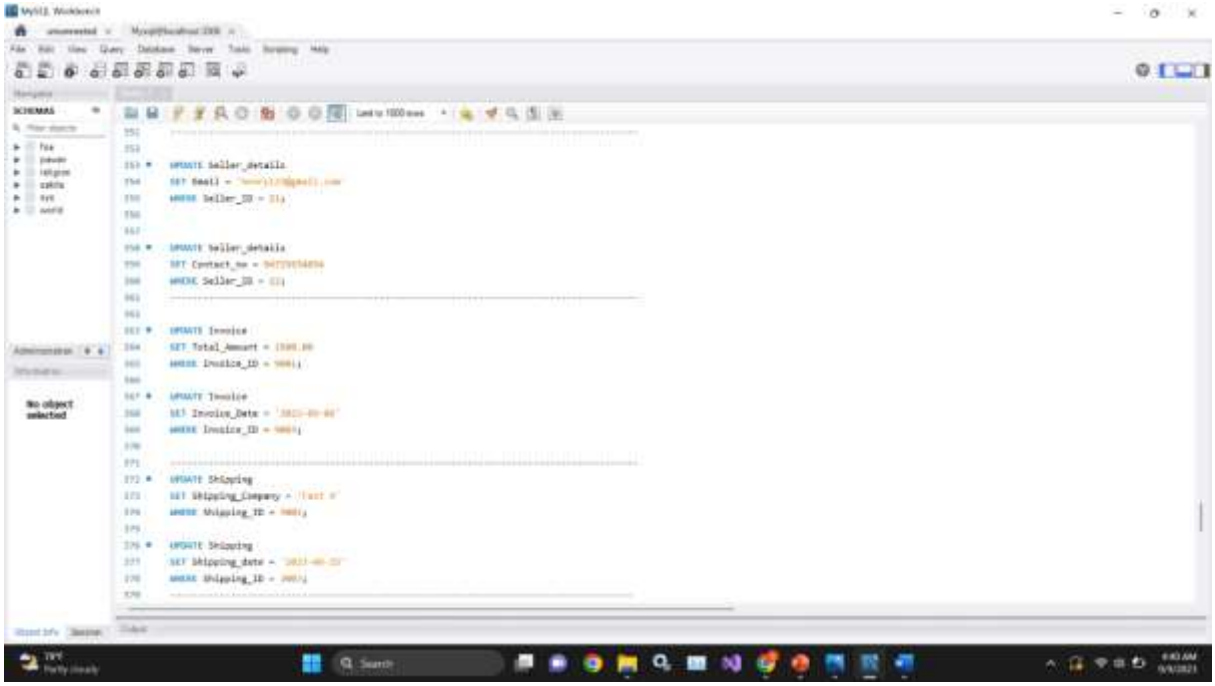


FIGURE 27:UPDATING DATA FROM TABLES-3

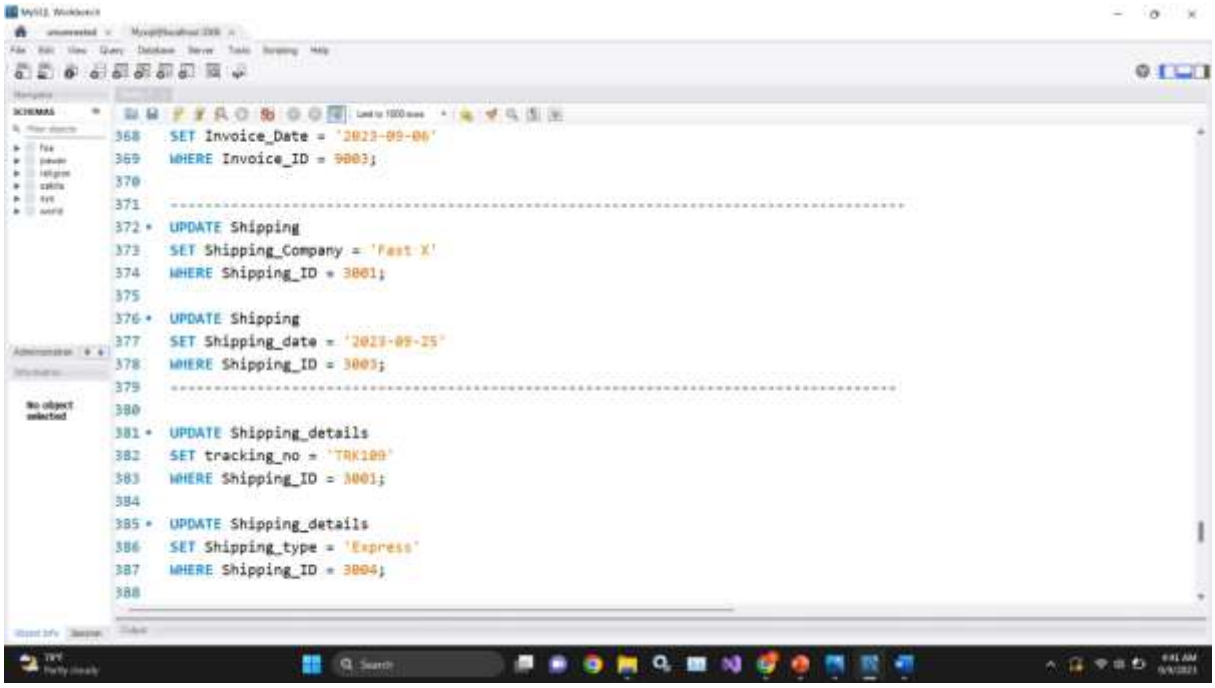


FIGURE 28:UPDATING DATA FROM TABLES-4

4.2.5 Delete data from the Tables

```
----------------------------------/*Deletion*/----------------------------------------------

DELETE FROM Laptop
WHERE Laptop_ID = 21001;


DELETE FROM Customer
WHERE Customer_ID = 1001;


DELETE FROM Payment
WHERE Payment_ID = 8001;


DELETE FROM Reviews
WHERE Review_ID = 104;


DELETE FROM Seller
WHERE Laptop_ID = 21006;
```

FIGURE 29:DELETE DATA FROM TABLES-1

```
DELETE FROM Seller_details
WHERE Seller_ID = 23;


DELETE FROM Orders
WHERE Order_ID = 'ORDER002';


DELETE FROM Invoice
WHERE Invoice_ID = 9002;


DELETE FROM Shipping
WHERE Shipping_ID = 3002;


DELETE FROM Shipping_details
WHERE Shipping_ID = 3002;
```

FIGURE 30:DELETE DATA FROM TABLES-2

5 CHAPTER 4: TRANSACTIONS

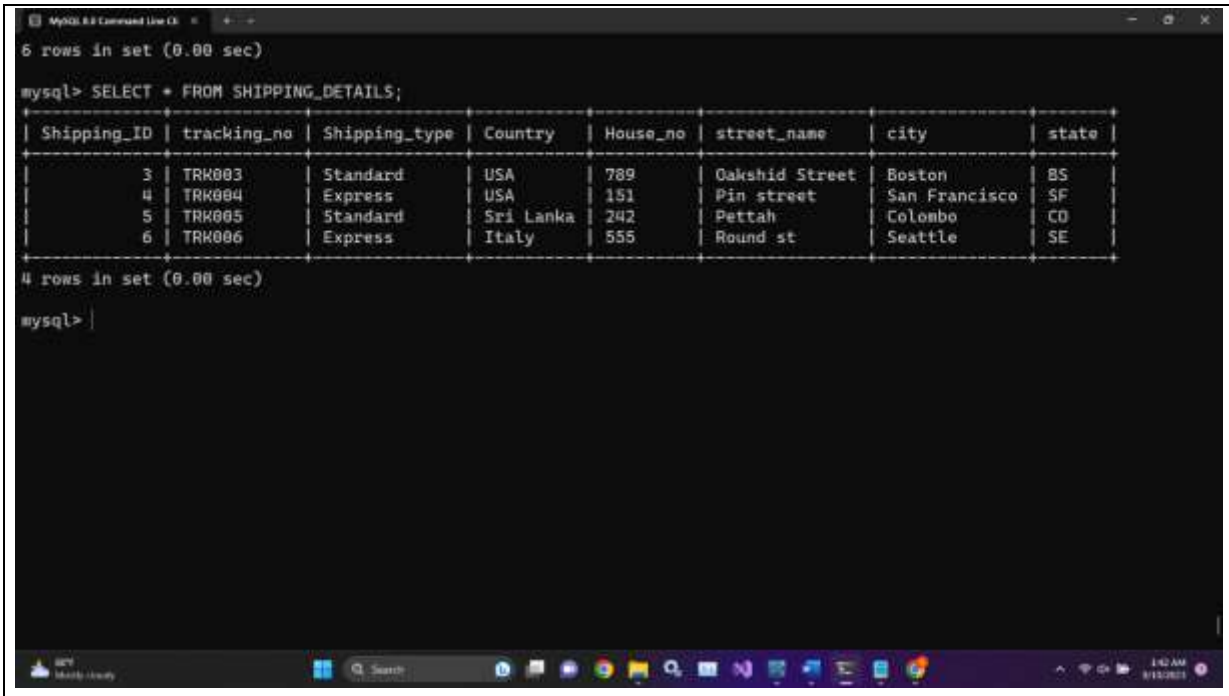
5.1 Simple queries

```
| pawan                                      |
| performance_schema                         |
| religion                                   |
| sakila                                     |
| sys                                        |
| world                                      |
+--------------------------------------------+
10 rows in set (0.00 sec)

mysql> use  laptop_ordering_and_selling_database_system;
Database changed
mysql> select * from laptop;
+-----------+--------+---------------+-----------+-----+---------------+---------+
| Laptop_ID | Brand  | Model         | Price     | Ram | Processor     | Storage |
+-----------+--------+---------------+-----------+-----+---------------+---------+
|     21002 | Dell   | Inspiron 3532 | 465000.00 | 8GB | Intel i9 9000 | 1TB     |
|     21003 | Dell   | Inspiron 5532 | 305000.00 | 8GB | Intel i9 9000 | 1TB     |
|     21004 | Huawei | Matebook X    | 405000.00 | 8GB | Intel i9 9000 | 1TB     |
|     21005 | Acer   | Nitro 15      | 335000.00 | 8GB | Intel i9 9000 | 1TB     |
|     21006 | MSI    | Stealth 15    | 505000.00 | 8GB | Intel i9 9000 | 1TB     |
+-----------+--------+---------------+-----------+-----+---------------+---------+
5 rows in set (0.00 sec)

mysql>
```

FIGURE 31:RETRIEVING ALL DATA FROM LAPTOP TABLE

```
+-----------+---------+---------------+-----------+-----+-------------
--+----------+
5 rows in set (0.00 sec)

mysql> select Payment_ID,amount,Payment_method from Payment;
+-----------+-----------+----------------+
| Payment_ID | amount    | Payment_method |
+-----------+-----------+----------------+
|       8002 | 500000.00 | Cash           |
|       8003 | 255120.00 | Debit          |
|       8004 | 175550.00 | Credit         |
|       8005 | 195000.00 | Cash           |
|       8006 | 600000.00 | Debit          |
+-----------+-----------+----------------+
5 rows in set (0.00 sec)

mysql>
```

FIGURE 32: RETRIEVING SELECTED DATA FROM PAYMENT TABLE



```
6 rows in set (0.00 sec)

mysql> SELECT * FROM SHIPPING_DETAILS;
+-------------+-------------+---------------+-----------+----------+-----------------+---------------+-------+
| Shipping_ID | tracking_no | Shipping_type | Country   | House_no | street_name     | city          | state |
+-------------+-------------+---------------+-----------+----------+-----------------+---------------+-------+
|           3 | TRK003      | Standard      | USA       | 789      | Oakshid Street  | Boston        | BS    |
|           4 | TRK004      | Express       | USA       | 151      | Pin street      | San Francisco | SF    |
|           5 | TRK005      | Standard      | Sri Lanka | 242      | Pettah          | Colombo       | CO    |
|           6 | TRK006      | Express       | Italy     | 555      | Round st        | Seattle       | SE    |
+-------------+-------------+---------------+-----------+----------+-----------------+---------------+-------+
4 rows in set (0.00 sec)

mysql>
```

FIGURE 33:SELECT OPERATION OF SHIPPING DETAILS TABLE

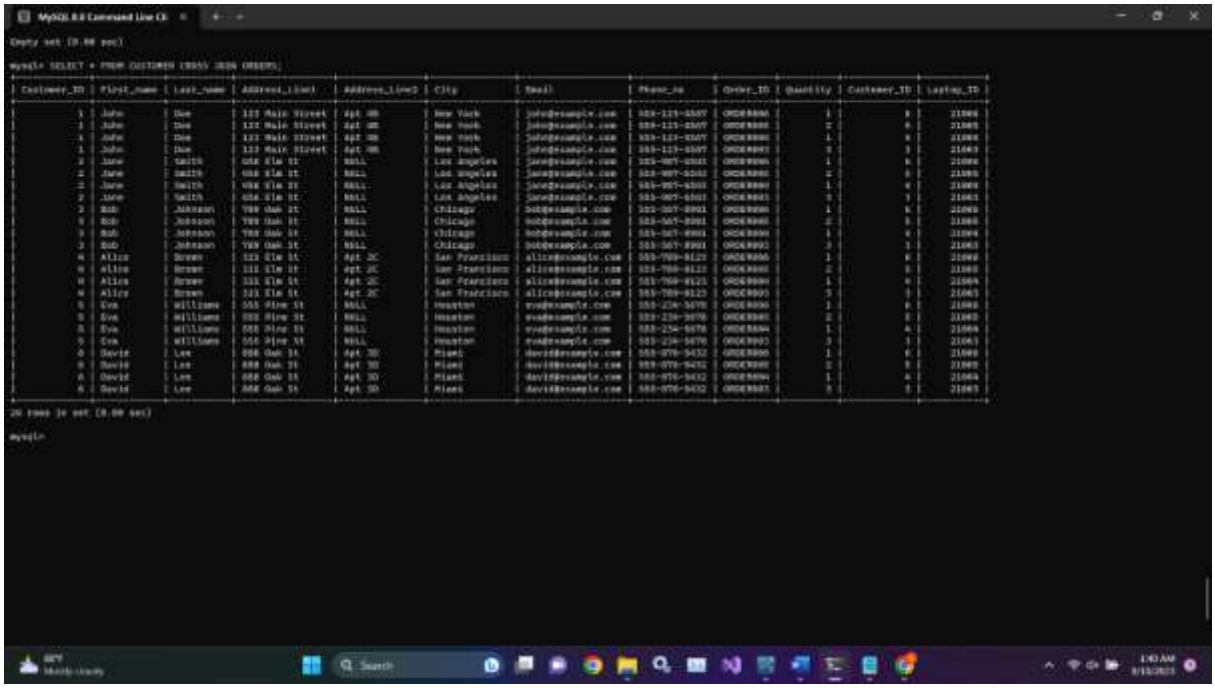


FIGURE 34:CARTESIAN PRODUCT OF CUSTOMER AND ORDERS

```
mysql> create view Delllaptops as select Laptop_ID,Model,price from Laptop where Brand = 'Dell
';
Query OK, 0 rows affected (0.12 sec)

mysql> select*from Delllaptops;
+-----------+---------------+-----------+
| Laptop_ID | Model         | price     |
+-----------+---------------+-----------+
|     21002 | Inspiron 3532 | 465000.00 |
|     21003 | Inspiron 5532 | 305000.00 |
+-----------+---------------+-----------+
2 rows in set (0.00 sec)

mysql>
```

FIGURE 35:CREATING USER VIEW

```
+-----------+---------------+-----------+
2 rows in set (0.00 sec)

mysql> select Review_ID , Comments as Review, Rating from Reviews;
+-----------+-------------------------+--------+
| Review_ID | Review                  | Rating |
+-----------+-------------------------+--------+
|       103 | Excellent build quality |      4 |
|       105 | Poor battery life       |      2 |
|       106 | Really dissapointed     |      4 |
+-----------+-------------------------+--------+
3 rows in set (0.00 sec)

mysql>
```

FIGURE 36:RENAME OPERATION

```
d' at line 1
mysql>

mysql> select Brand,AVG(Price) as Average_Price from Laptop group by Brand;
+---------+----------------+
| Brand   | Average_Price  |
+---------+----------------+
| Dell    | 385000.000000  |
| Huawei  | 405000.000000  |
| Acer    | 335000.000000  |
| MSI     | 505000.000000  |
+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```

FIGURE 37:USING AGGREGATION FUNCTIONS

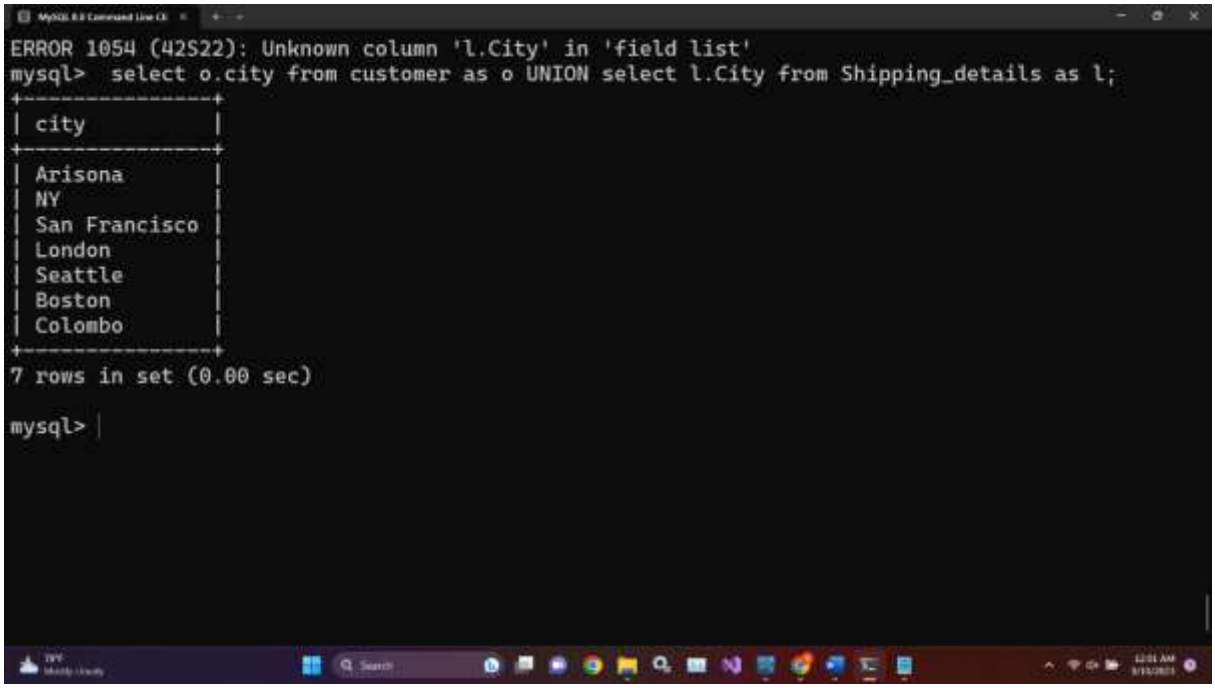
```
mysql> SELECT brand,model,price from Laptop order by price asc;
+--------+---------------+------------+
| brand  | model         | price      |
+--------+---------------+------------+
| Dell   | Inspiron 5532 | 305000.00  |
| Acer   | Nitro 15      | 335000.00  |
| Huawei | Matebook X    | 405000.00  |
| Dell   | Inspiron 3532 | 465000.00  |
| MSI    | Stealth 15    | 505000.00  |
+--------+---------------+------------+
5 rows in set (0.00 sec)

mysql>
```

FIGURE 38:ORDER BY OPERATION

```
4 rows in set (0.00 sec)

mysql> select First_Name,Last_Name,Email from Customer where Email like '%gmail.com';
+------------+------------+--------------------+
| First_Name | Last_Name  | Email              |
+------------+------------+--------------------+
| nipum      | lanka      | pawan@gmail.com    |
| kamal      | Jade       | robert@gmail.com   |
| john       | Snow       | jsnow@gmail.com    |
| Michael    | henry      | michael@gmail.com  |
| Abdullah   | Farhad     | AF@gmail.com       |
+------------+------------+--------------------+
5 rows in set (0.00 sec)

mysql>
```

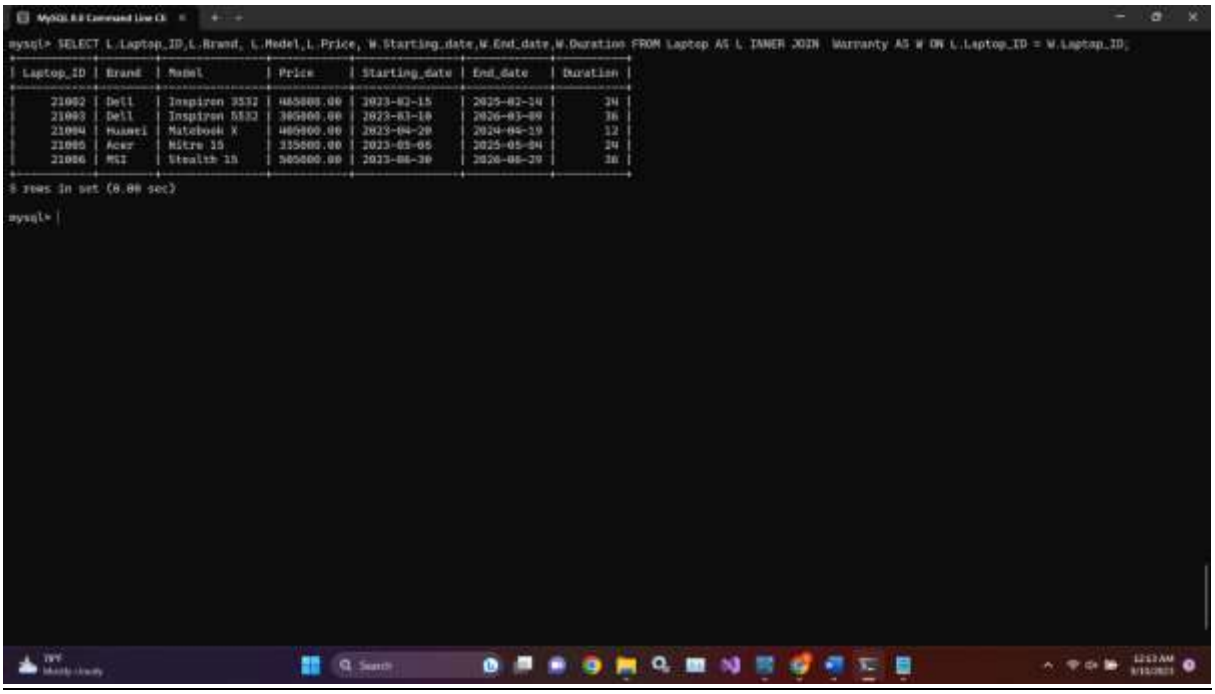
FIGURE 39:LIKE OPERATION

5.2 Complex queries



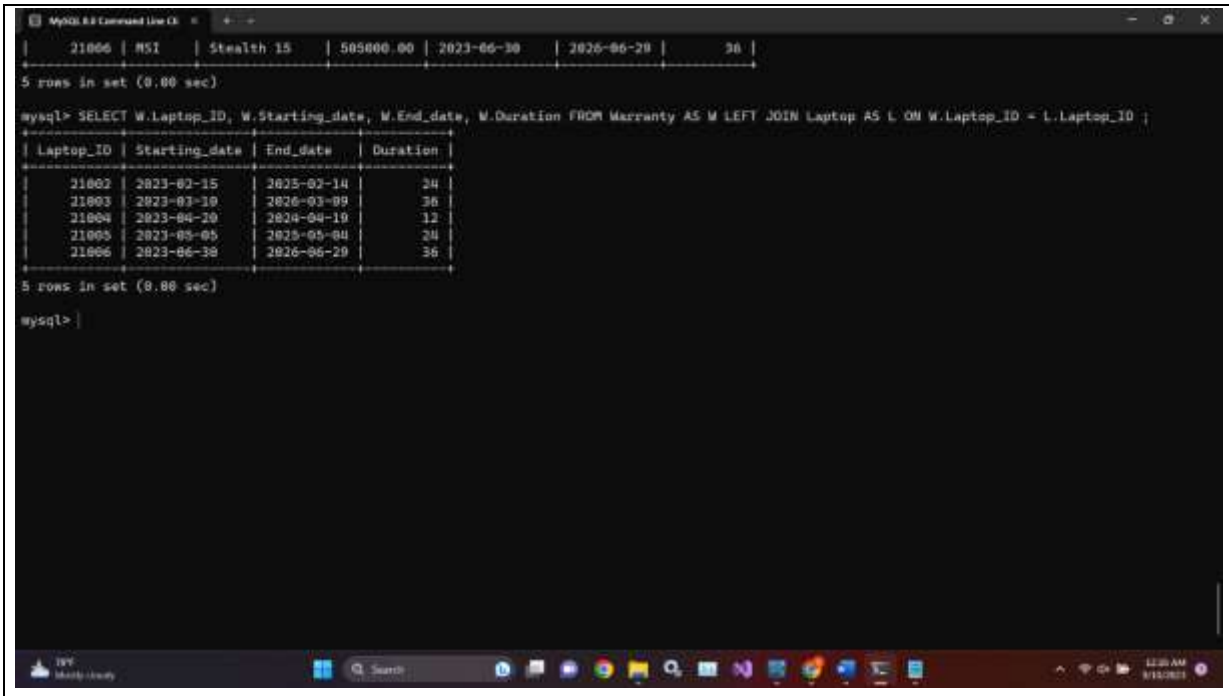
```
ERROR 1054 (42S22): Unknown column 'l.City' in 'field list'
mysql>  select o.city from customer as o UNION select l.City from Shipping_details as l;
+----------------+
| city           |
+----------------+
| Arisona        |
| NY             |
| San Francisco  |
| London         |
| Seattle        |
| Boston         |
| Colombo        |
+----------------+
7 rows in set (0.00 sec)

mysql>
```

FIGURE 40:UNION



```
mysql> SELECT L.Laptop_ID,L.Brand, L.Model,L.Price, W.Starting_date,W.End_date,W.Duration FROM Laptop AS L INNER JOIN  Warranty AS W ON L.Laptop_ID = W.Laptop_ID;
+------------+--------+----------------+-----------+---------------+-------------+----------+
| Laptop_ID  | Brand  | Model          | Price     | Starting_date | End_date    | Duration |
+------------+--------+----------------+-----------+---------------+-------------+----------+
|      21002 | Dell   | Inspiron 3532  | 465000.00 | 2023-02-15    | 2025-02-14  |       24 |
|      21003 | Dell   | Inspiron 5532  | 385000.00 | 2023-03-10    | 2026-03-09  |       36 |
|      21004 | Huawei | Matebook X     | 480000.00 | 2023-04-20    | 2024-04-19  |       12 |
|      21005 | Acer   | Nitro 15       | 315000.00 | 2023-05-05    | 2025-05-04  |       24 |
|      21006 | MSI    | Stealth 15     | 585000.00 | 2023-06-30    | 2026-06-29  |       36 |
+------------+--------+----------------+-----------+---------------+-------------+----------+
5 rows in set (0.00 sec)

mysql>
```

FIGURE 41:INTERSECTION



```
|      21006 | MSI     | Stealth 15     | 505000.00 | 2023-06-30     | 2026-06-29 |        36 |
+------------+---------+----------------+-----------+----------------+------------+-----------+
5 rows in set (0.00 sec)

mysql> SELECT W.Laptop_ID, W.Starting_date, W.End_date, W.Duration FROM Warranty AS W LEFT JOIN Laptop AS L ON W.Laptop_ID = L.Laptop_ID ;
+------------+----------------+------------+-----------+
| Laptop_ID | Starting_date | End_date   | Duration |
+------------+----------------+------------+-----------+
|      21002 | 2023-02-15     | 2025-02-14 |        24 |
|      21003 | 2023-03-10     | 2026-03-09 |        36 |
|      21004 | 2023-04-20     | 2024-04-19 |        12 |
|      21005 | 2023-05-05     | 2025-05-04 |        24 |
|      21006 | 2023-06-30     | 2026-06-29 |        36 |
+------------+----------------+------------+-----------+
5 rows in set (0.00 sec)

mysql>
```

FIGURE 42:SET DIFFERENCE

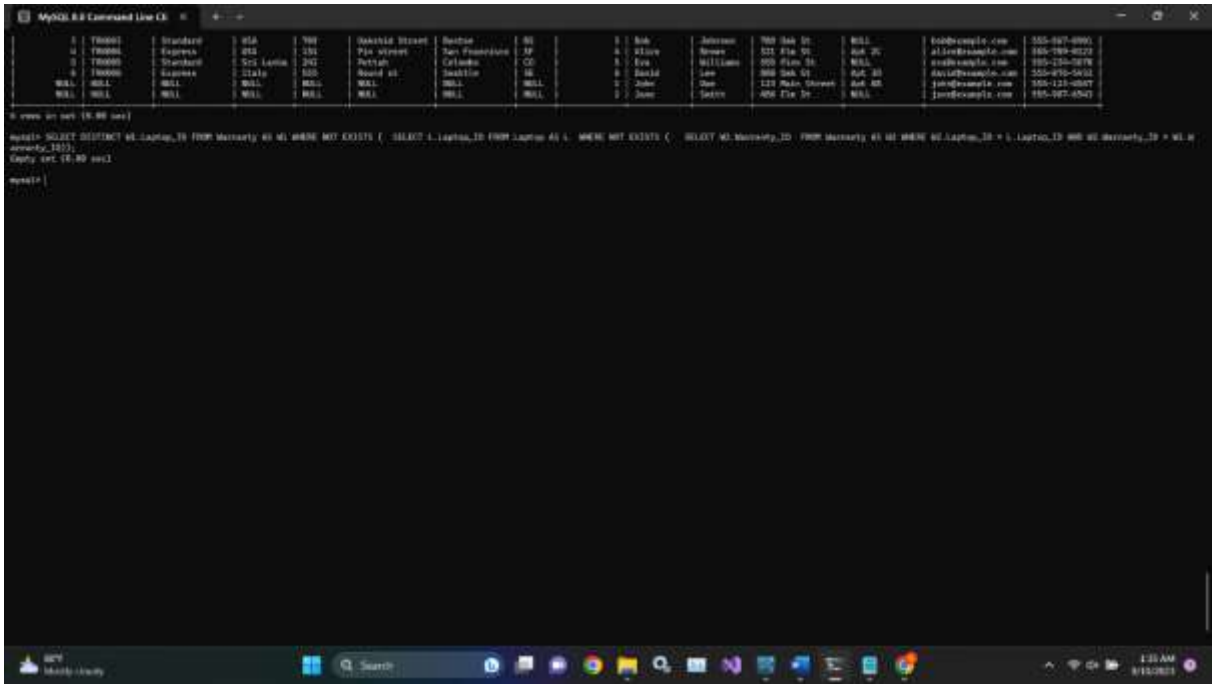


FIGURE 43:DIVISION

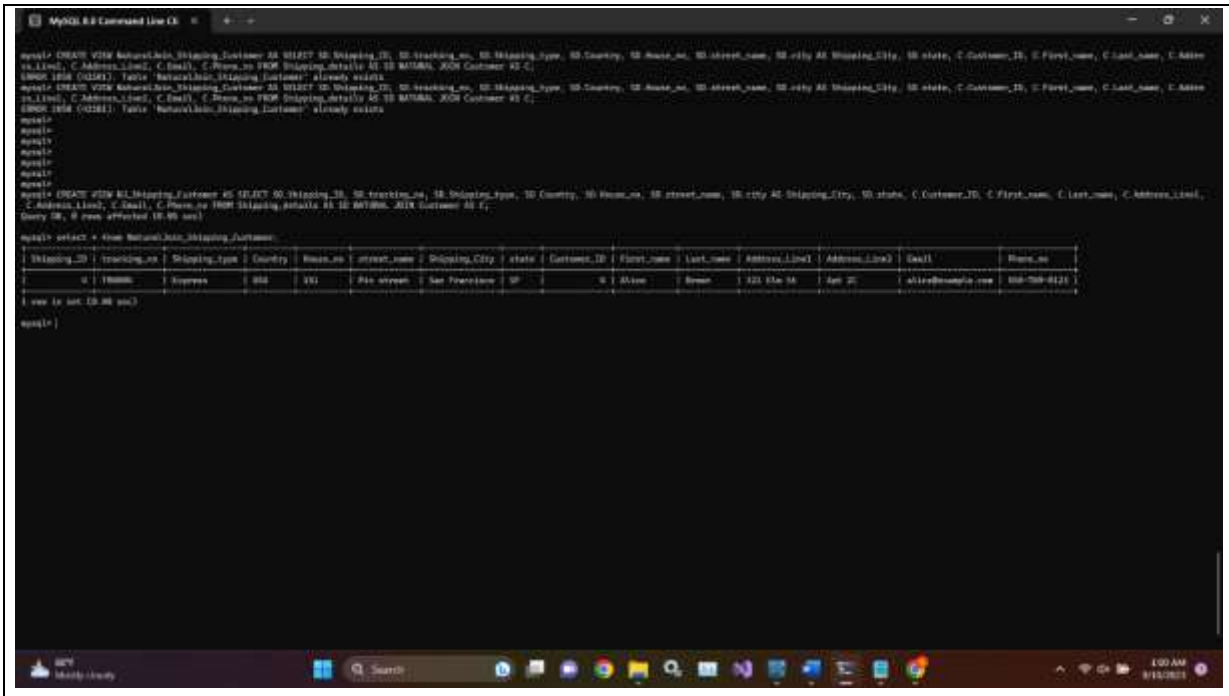


FIGURE 44:NATURAL JOIN

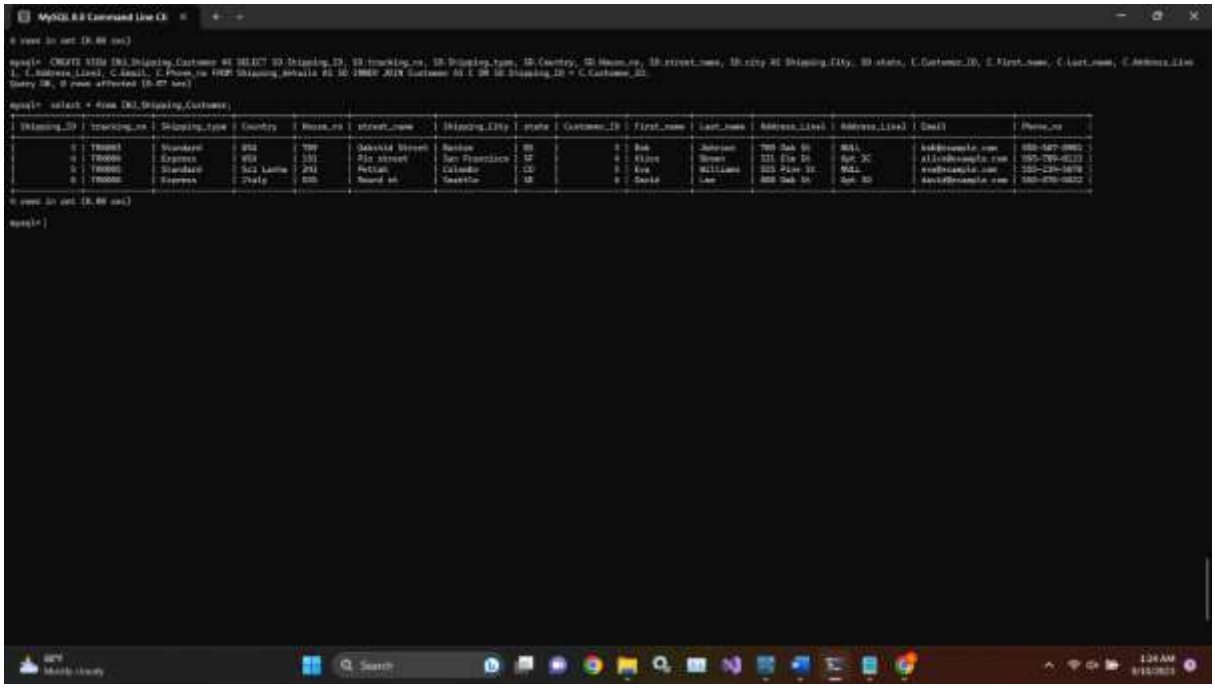


FIGURE 45:INNER JOIN

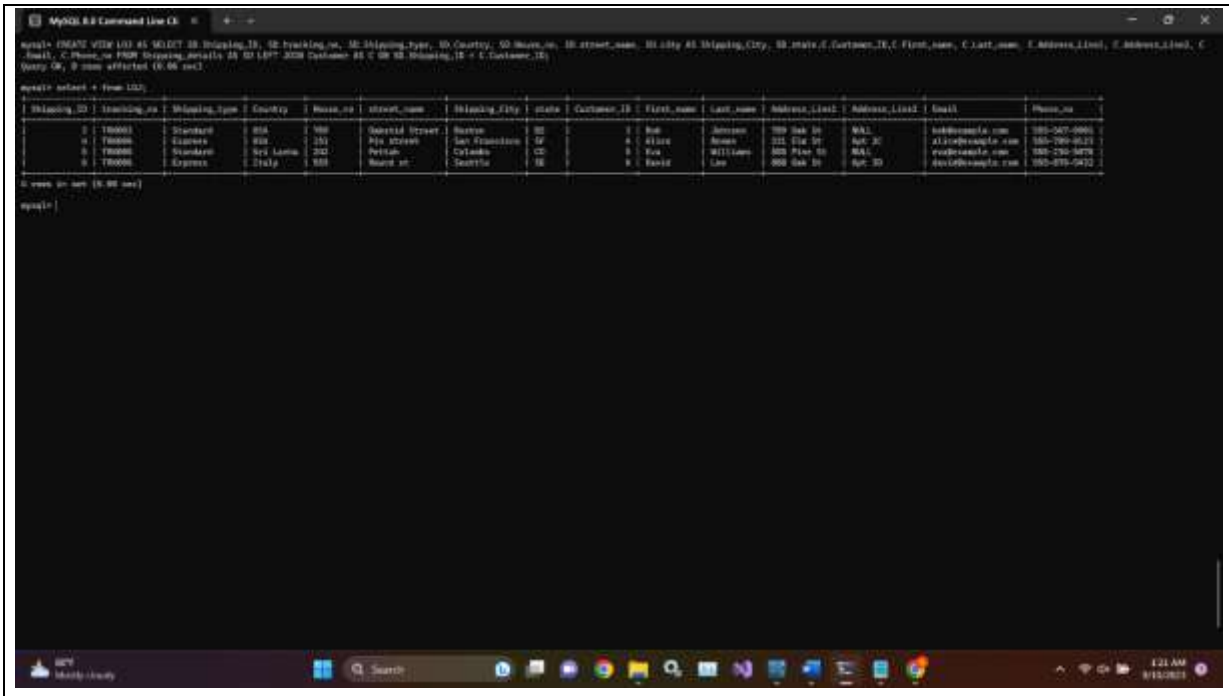


FIGURE 46:LEFT OUTER JOIN

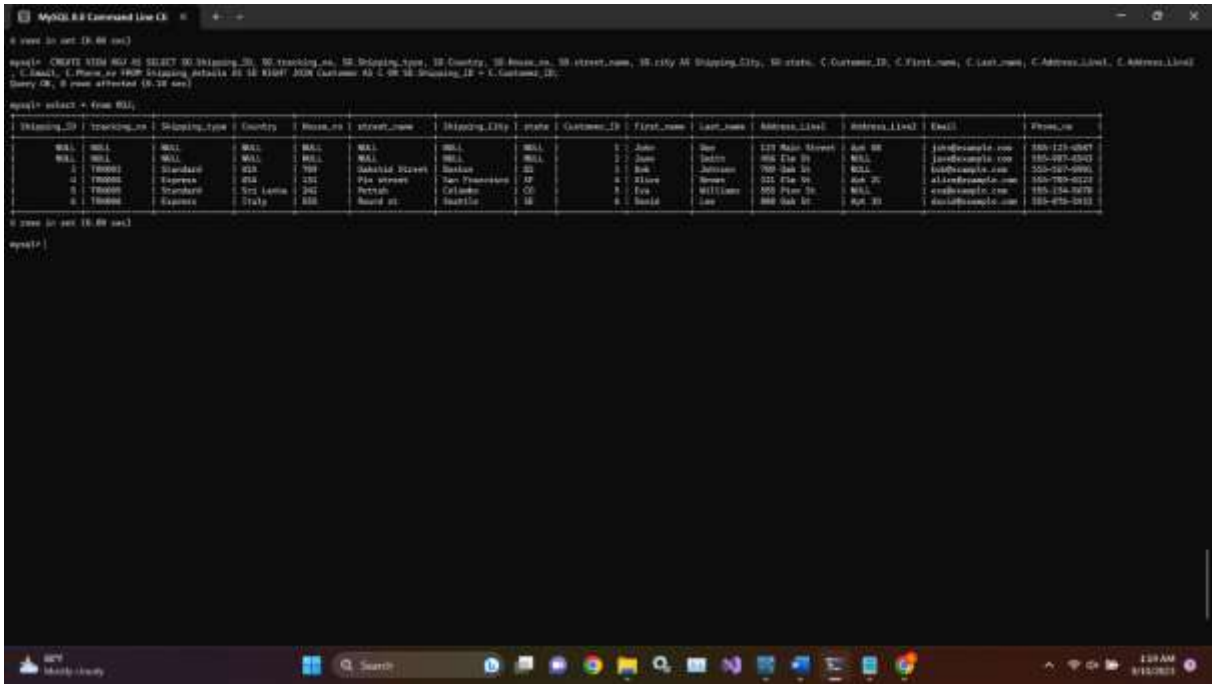


FIGURE 47 RIGHT OUTER JOIN

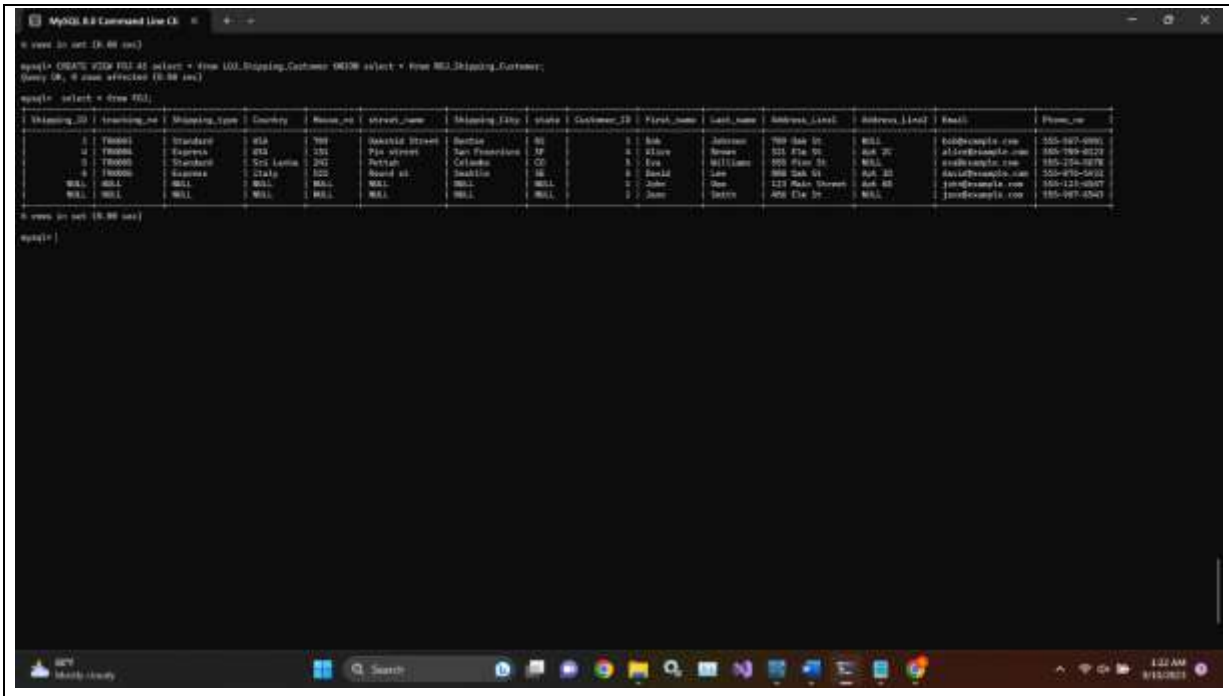


FIGURE 48:FULL OUTR JOIN

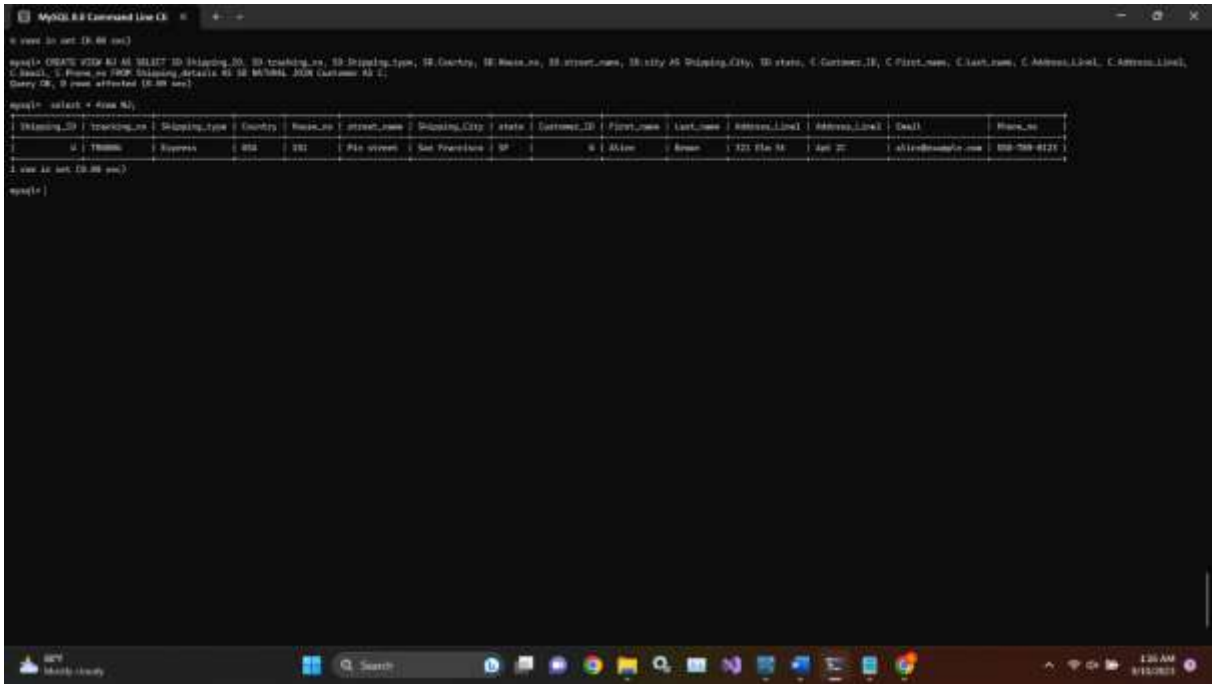


FIGURE 49:OUTER UNION

6 CHAPTER 5: TUNING

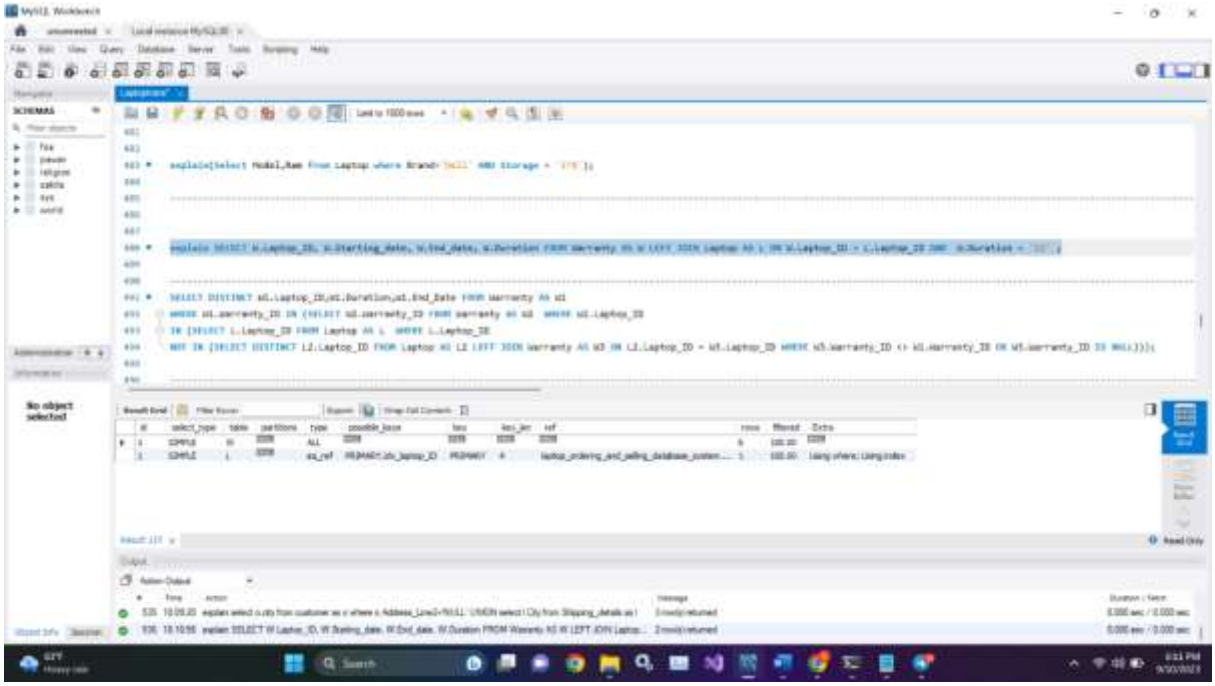


FIGURE 50:BEFORE TUNING DIFFERENCE OPERATION

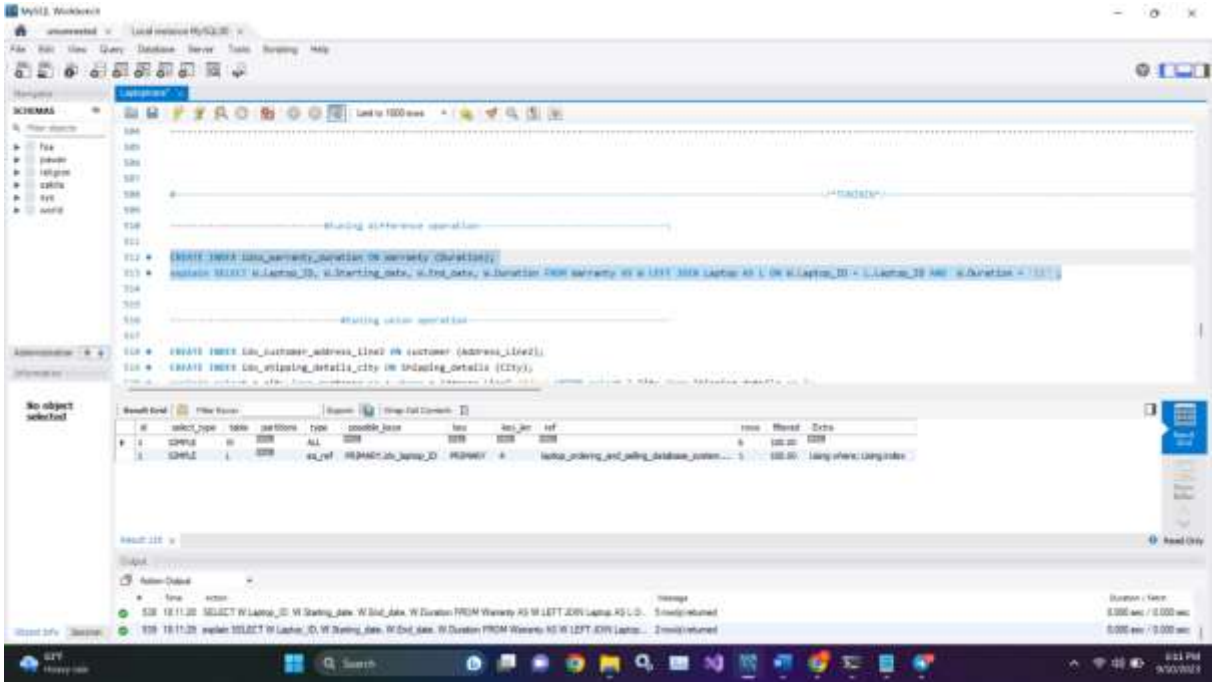


FIGURE 51:AFTER TUNING DIFFERENCE OPERATION

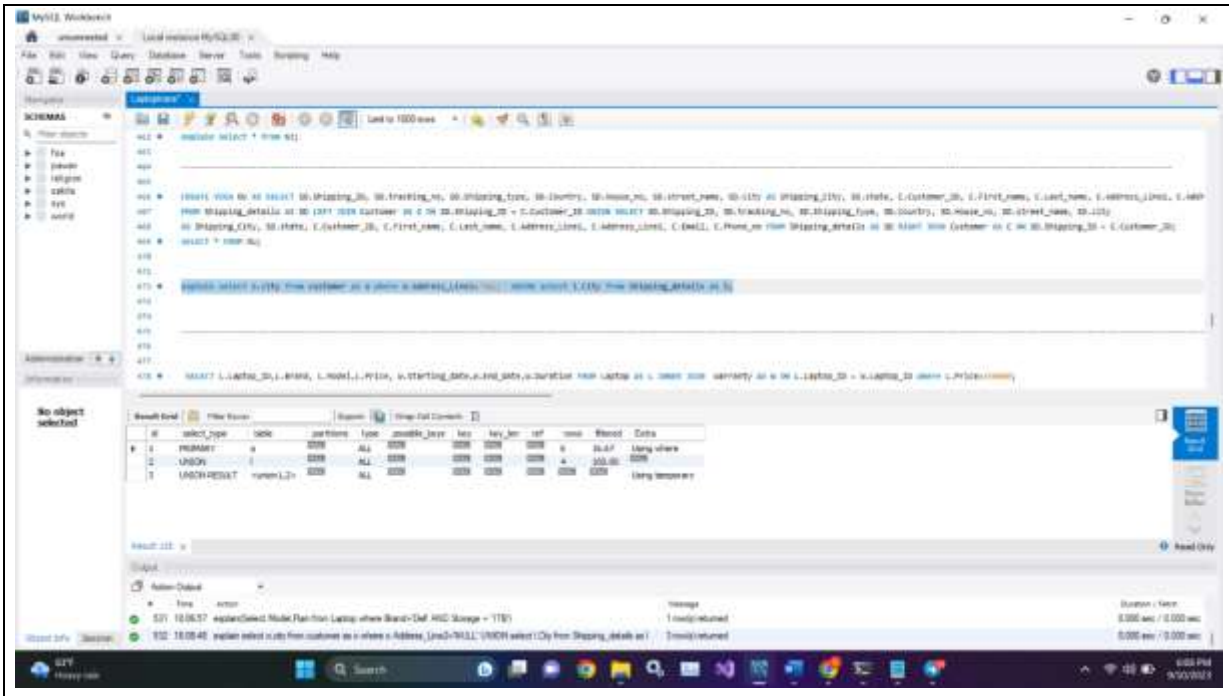


FIGURE 52: BEFORE TUNING UNION OPERATION

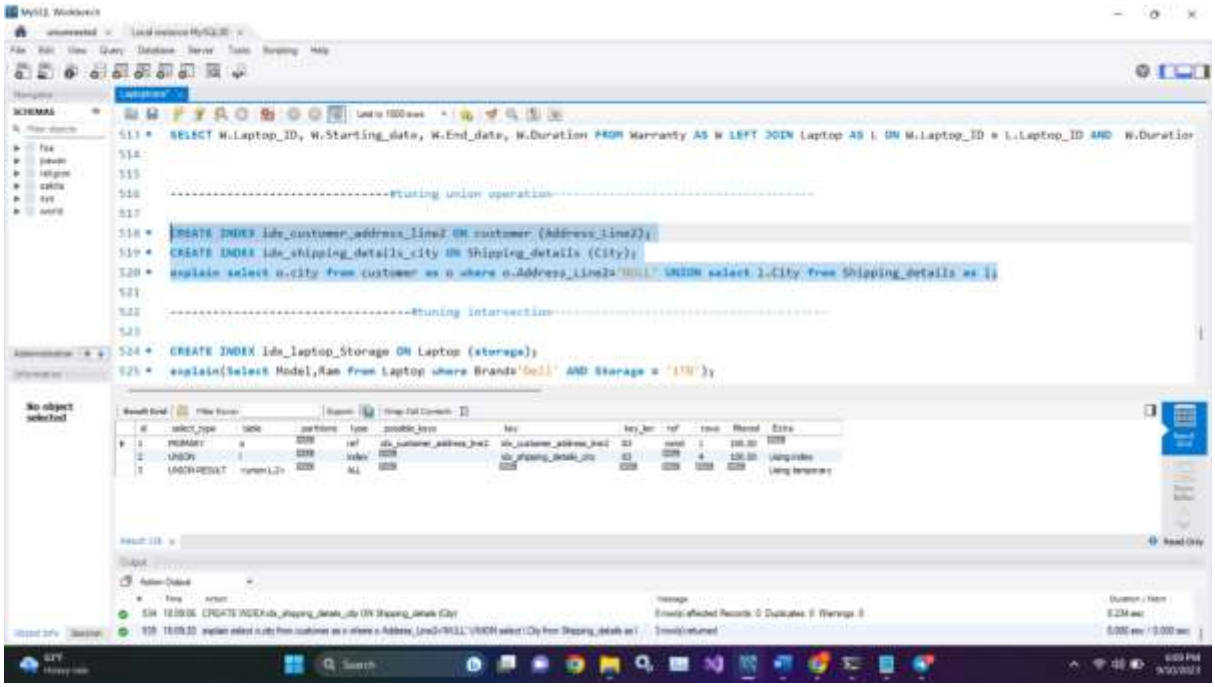


FIGURE 53: AFTER TUNING UNION OPERATION

- When considering the above two figures, it can be seen that number of accessing rows have been reduced after the tuning process. So it can be considered that above query was tunned properly by using an index.

=-==-==

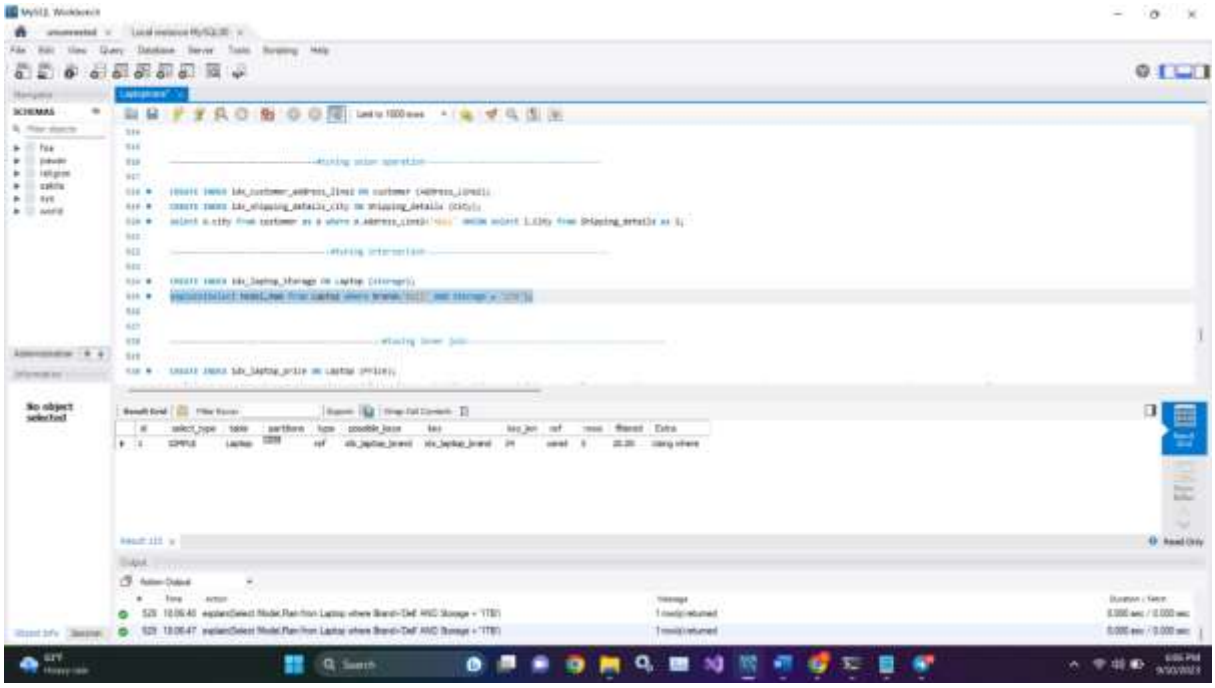


FIGURE 54 BEFORE TUNING INTERSECTION OPERATION

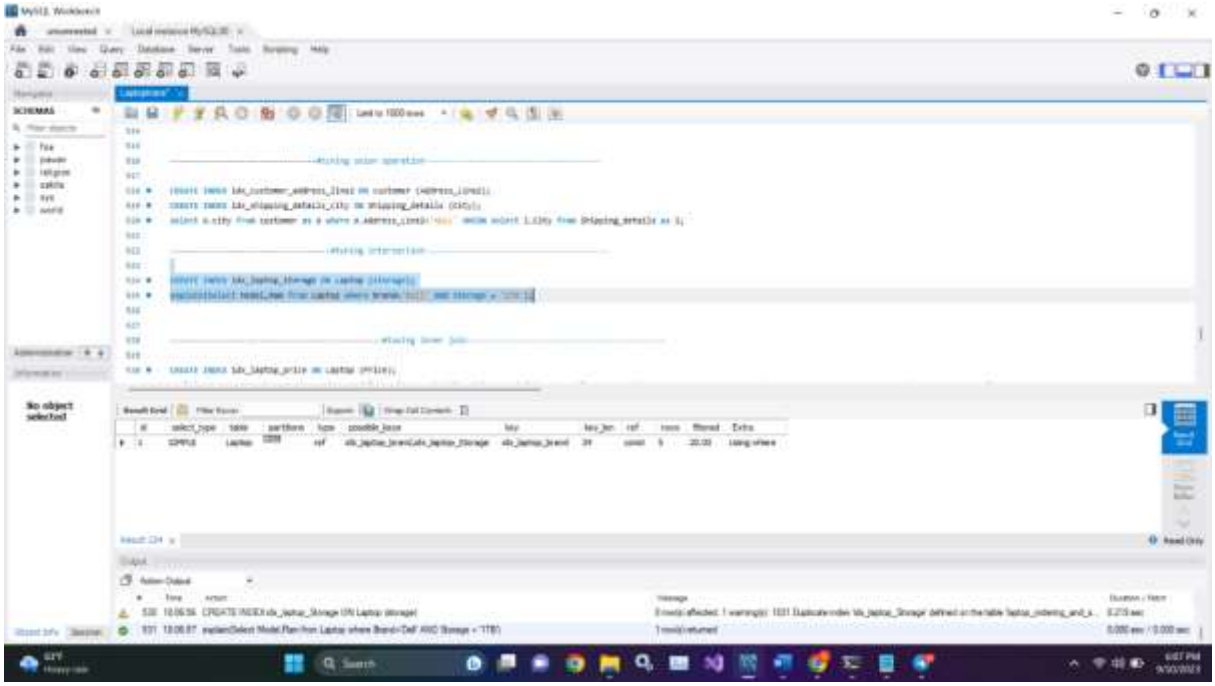


FIGURE 55 AFTER TUNING INTERSECTION OPERATION

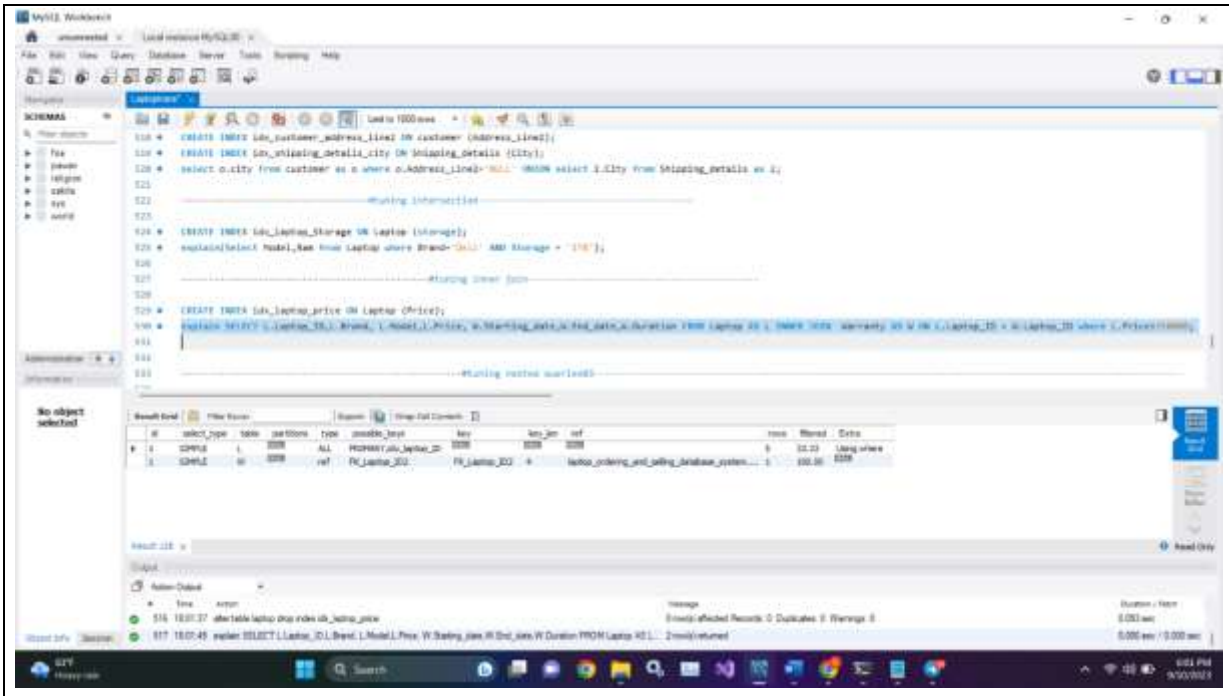


FIGURE 56 BEFORE TUNING INNER JOIN OPERATION

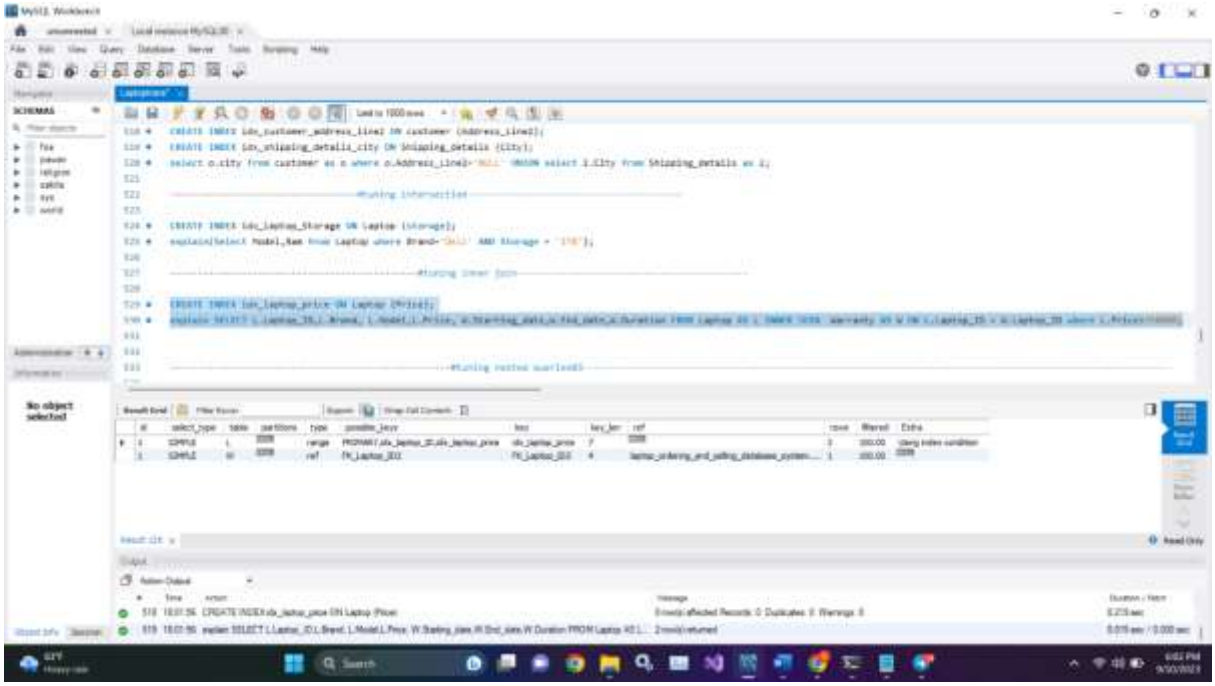


FIGURE 57 AFTER TUNING INNER JOIN OPERATION

- Here also the number of accessed rows have been reduced after the tuning process. That means data were filtered properly by using the index.

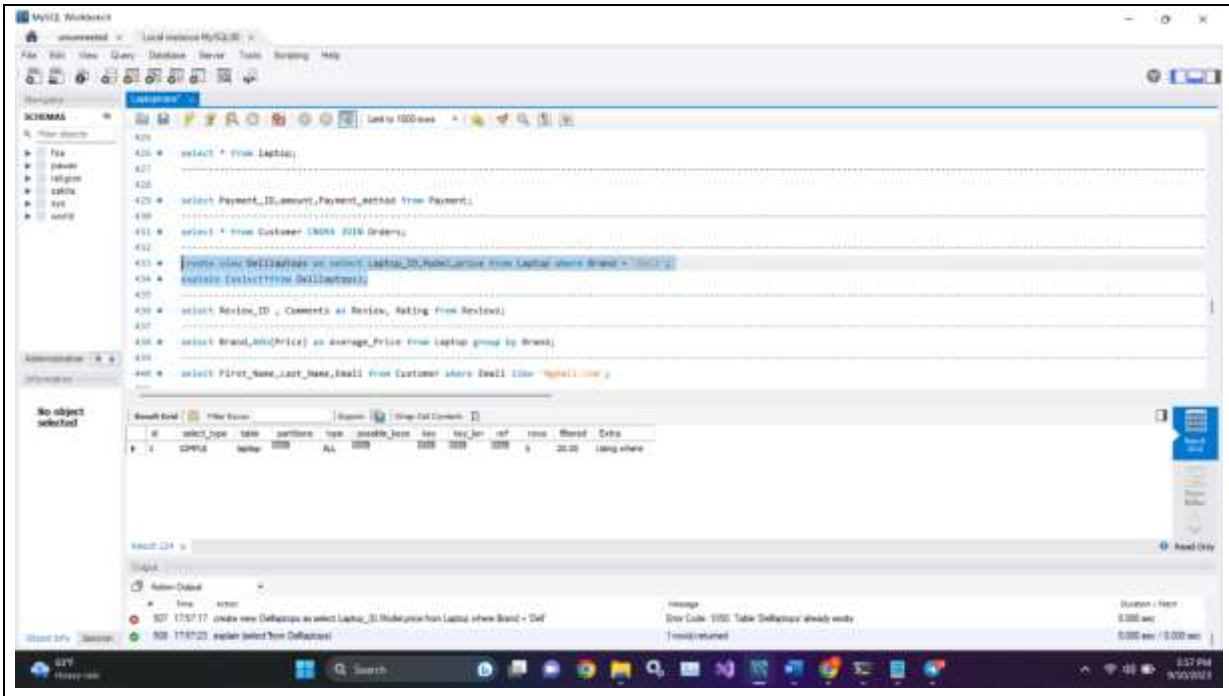


FIGURE 58 BEFORE TUNING SELECT

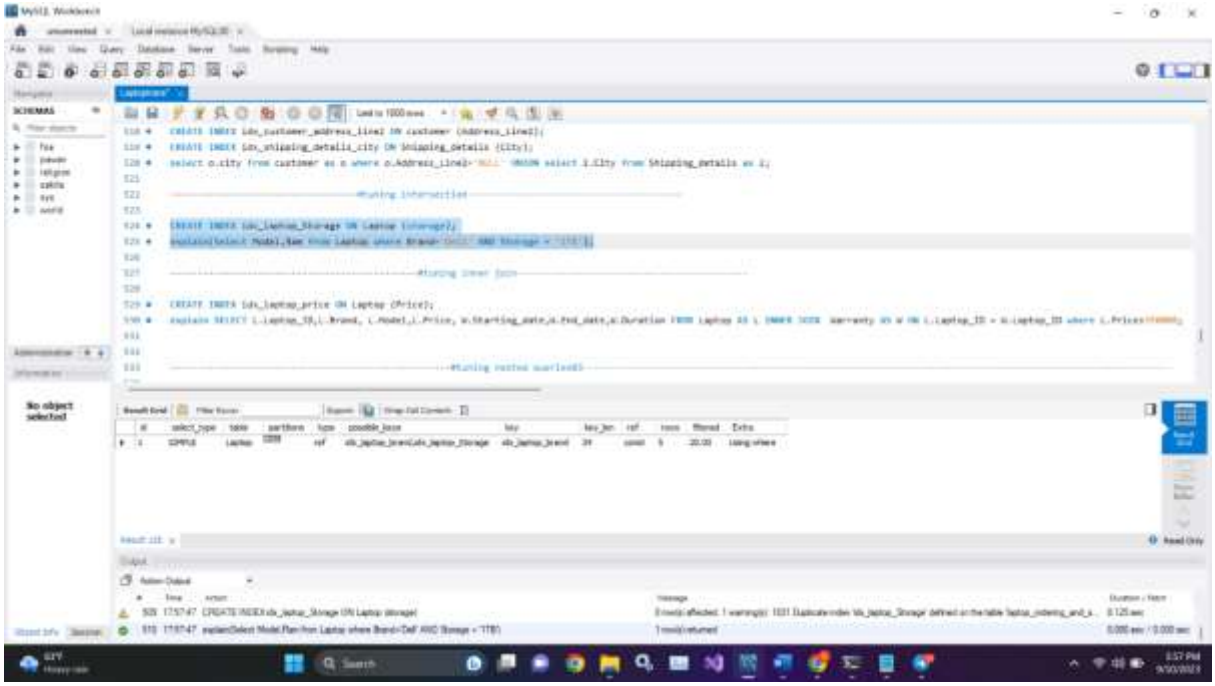


FIGURE 59 AFTER TUNING SELECT

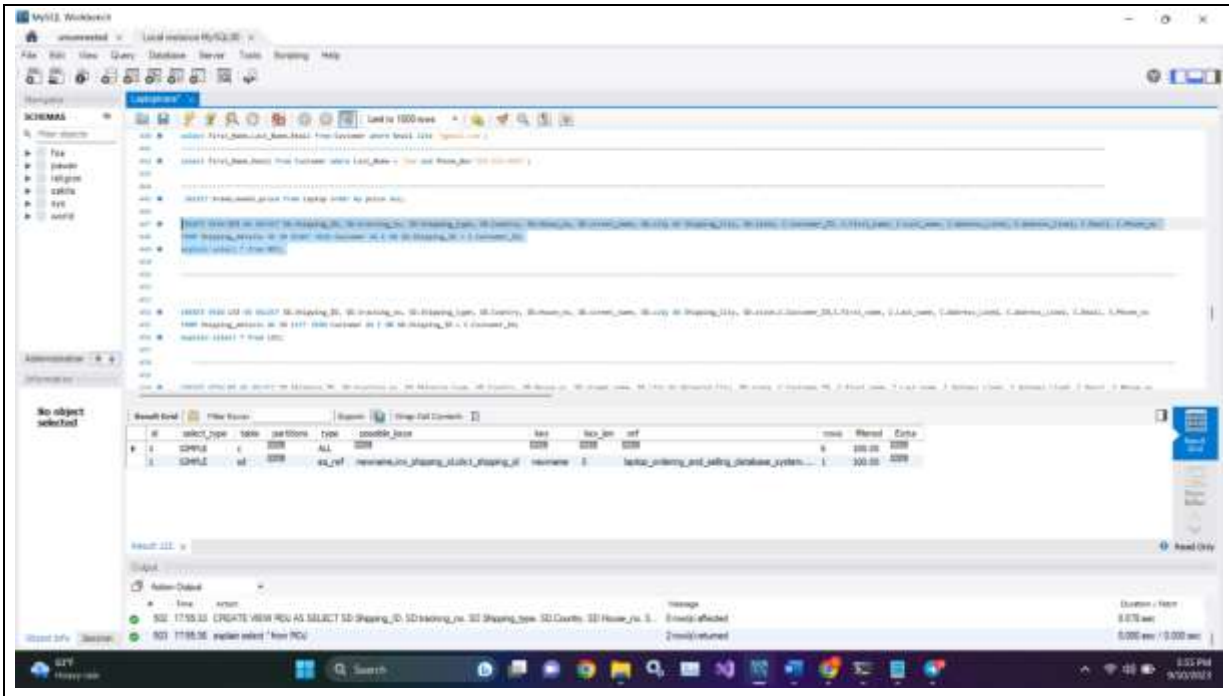


FIGURE 60 BEFORE TUNING RIGHT OUTER JOIN

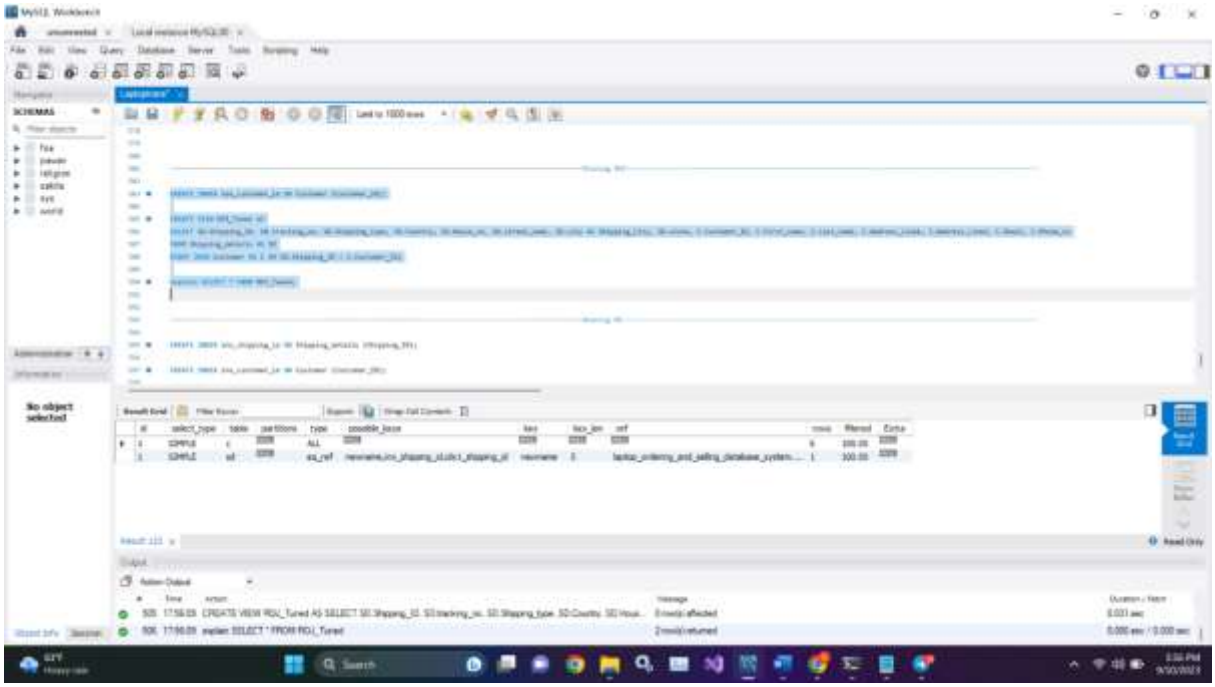


FIGURE 61 AFTER TUNING RIGHT OUTER JOIN

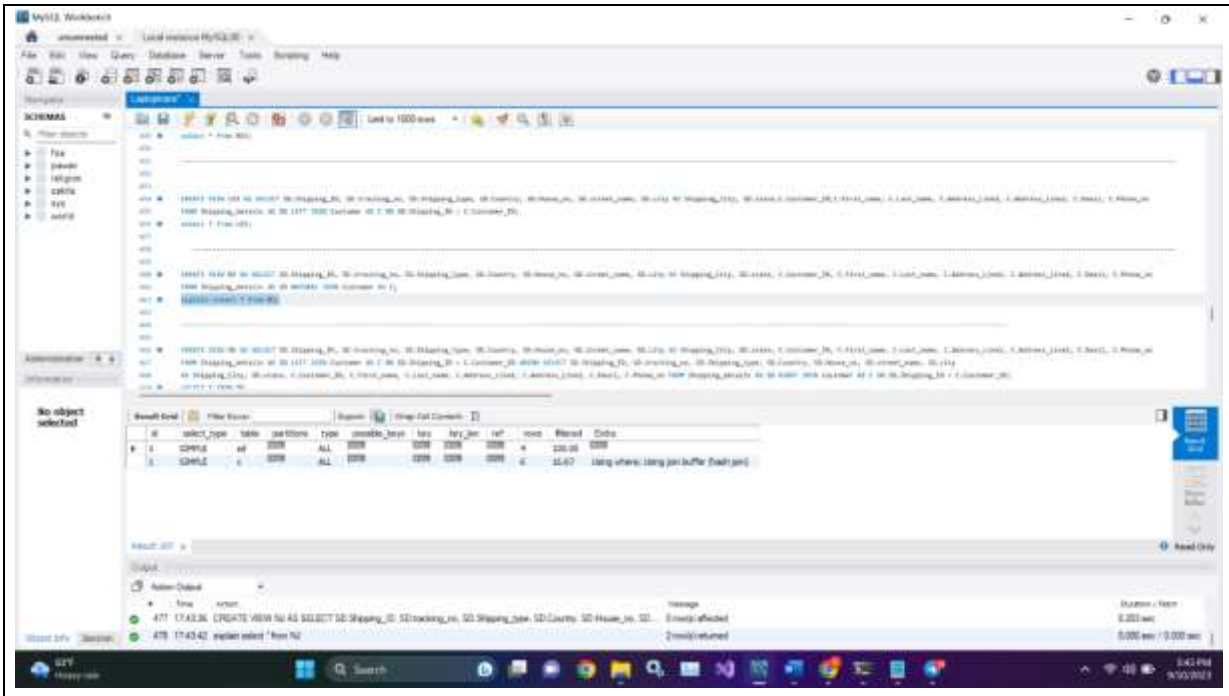


FIGURE 62 BEFORE TUNING NATURAL JOIN

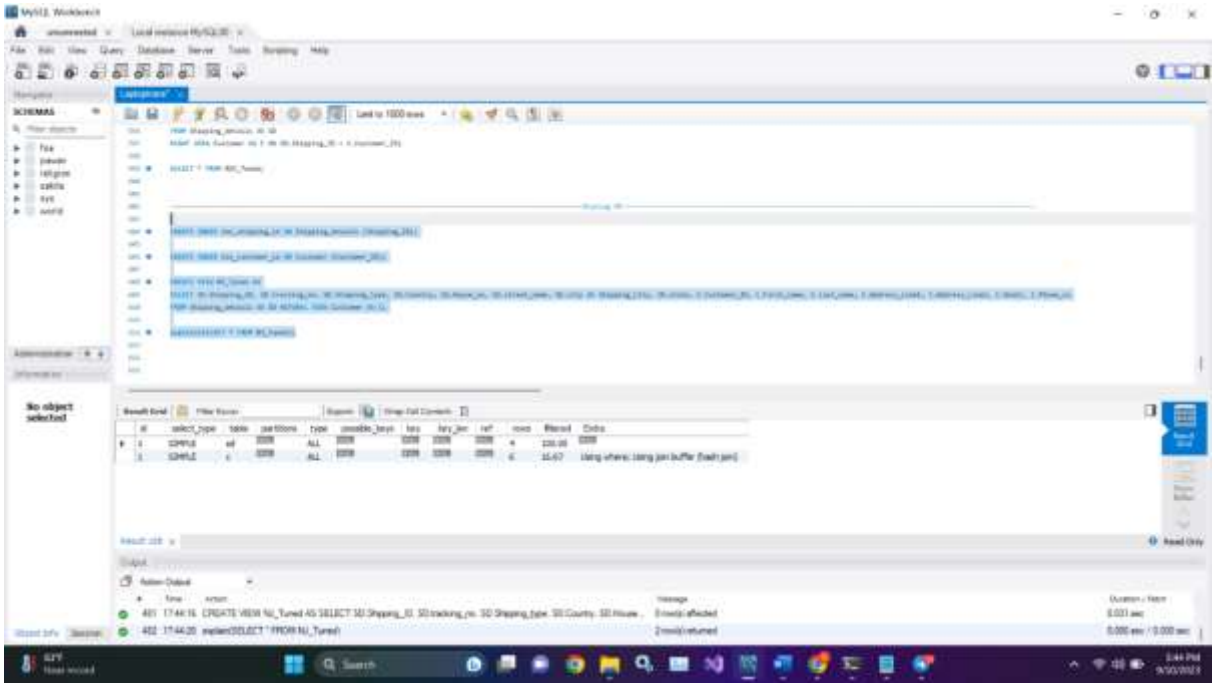


FIGURE 63 AFTER TUNING NATURAL JOIN

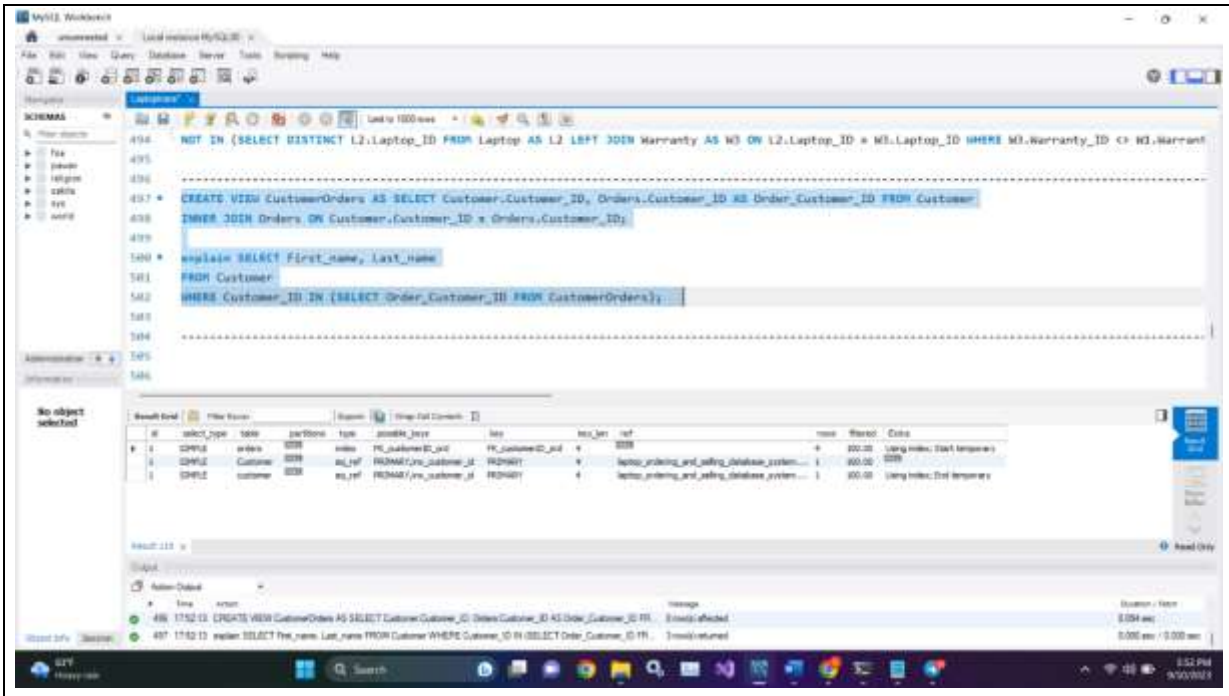


FIGURE 64 BEFORE TUNING NESTED QUERY01

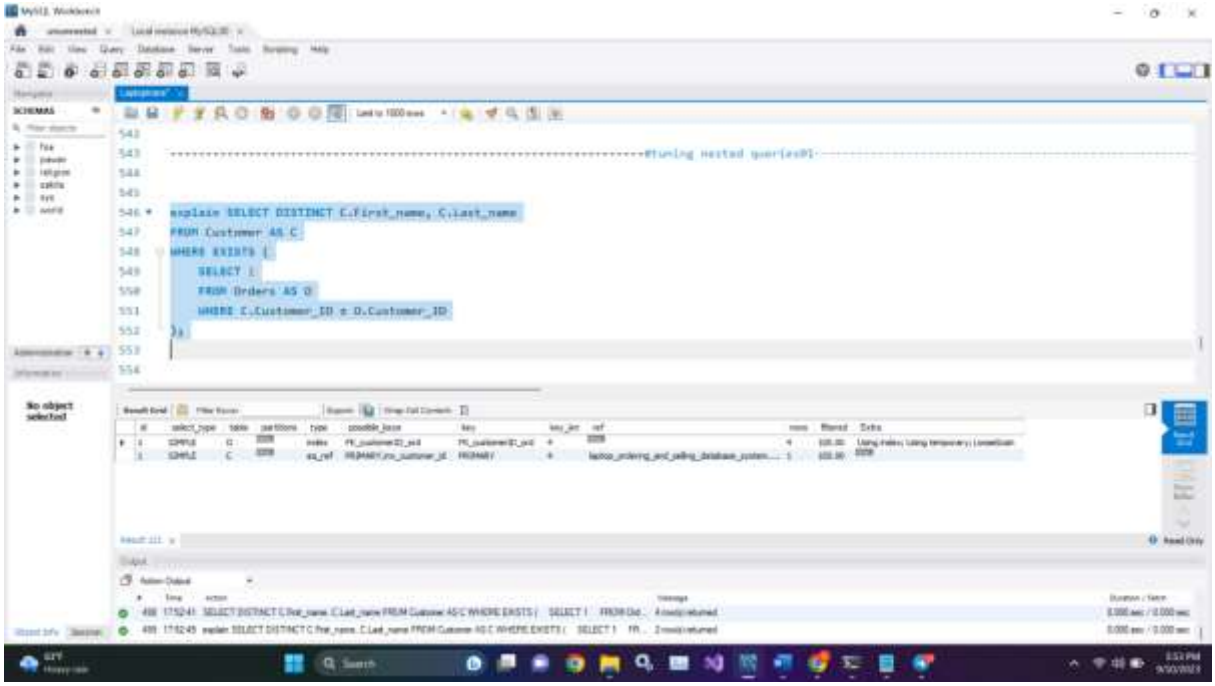


FIGURE 65 AFTER TUNING NESTED QUERY01

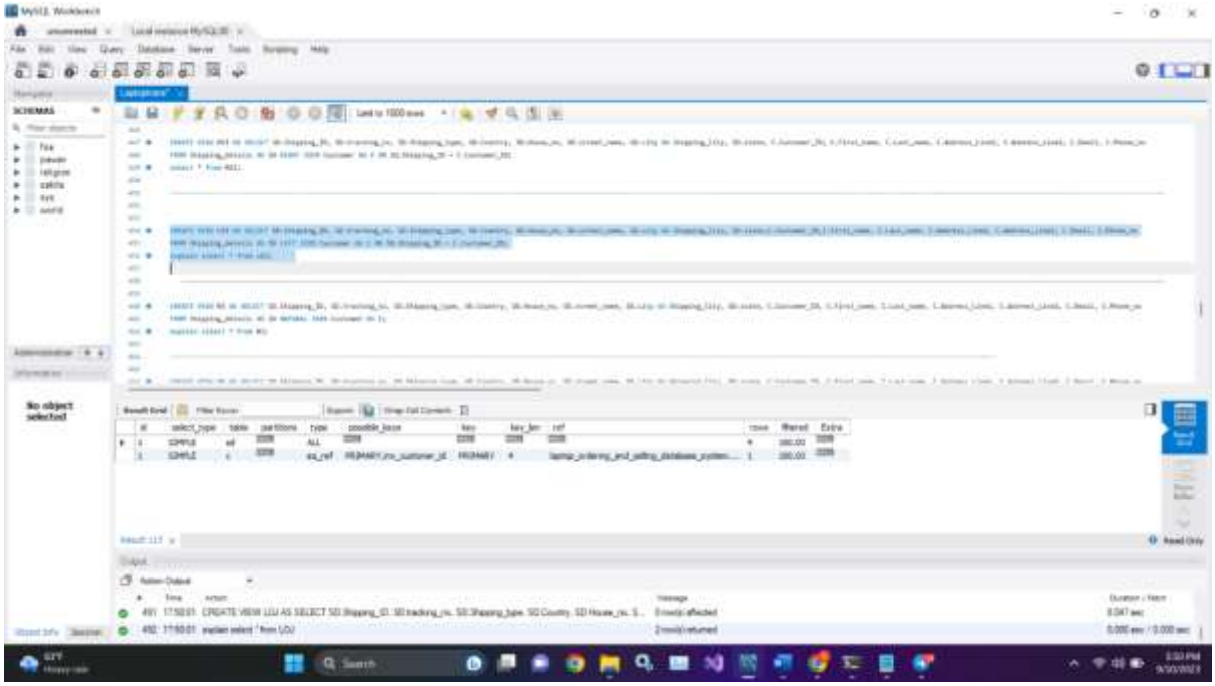


FIGURE 66 BEFORE TUNING NESTED QUERY02

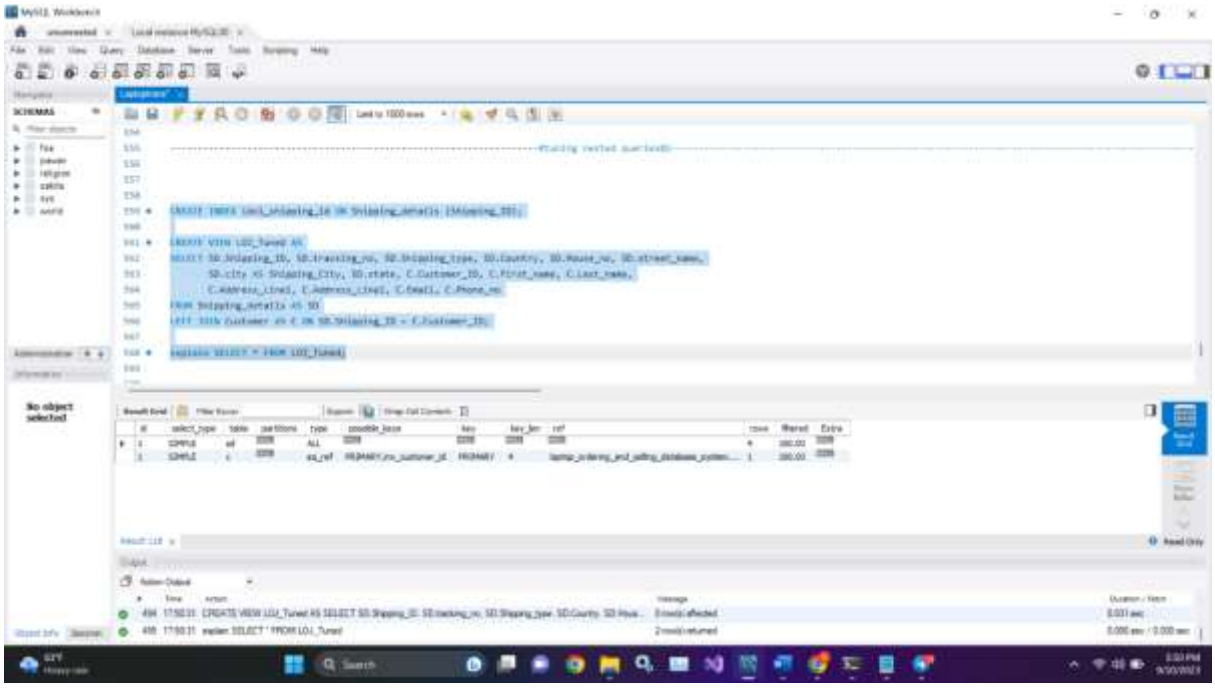


FIGURE 67 AFTER TUNING NESTED QUERY02

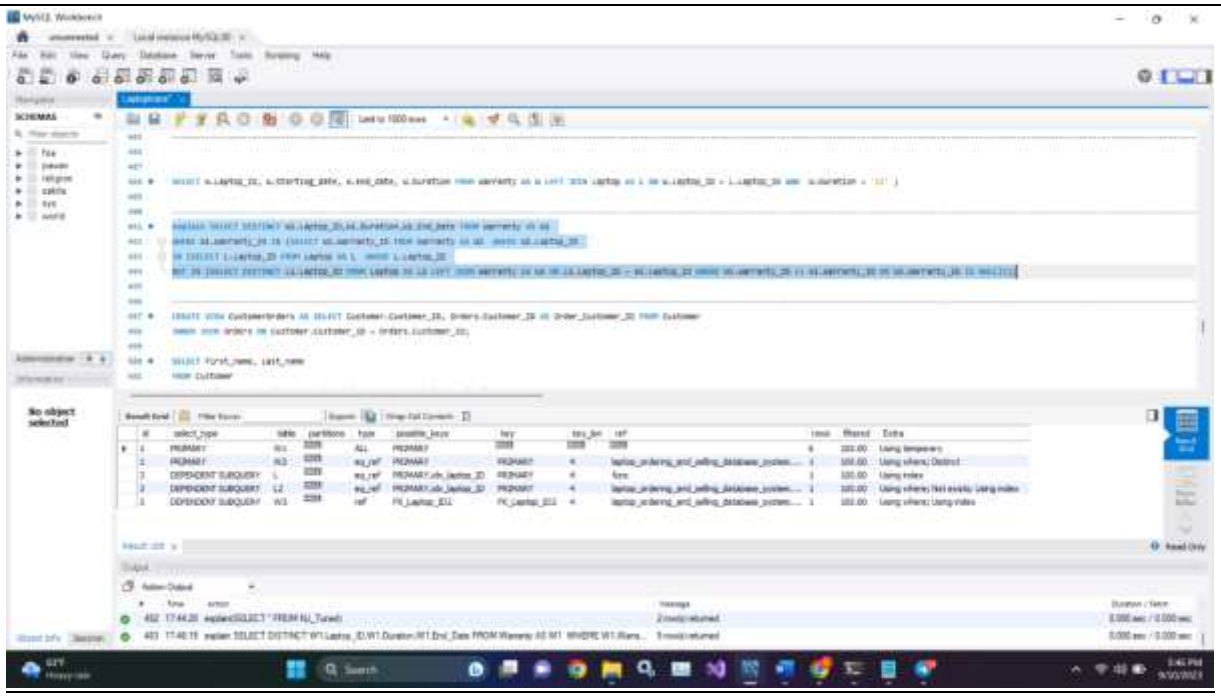


FIGURE 68 BEFORE TUNING NESTED QUERY03

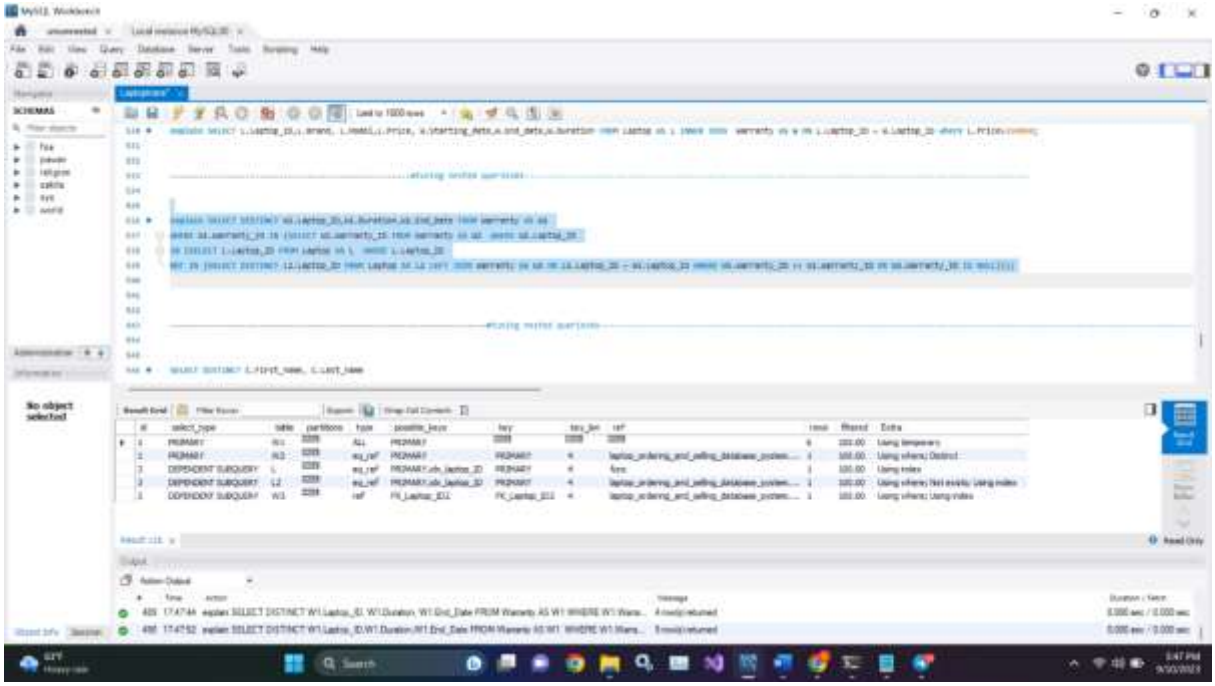


FIGURE 69 AFTER TUNING NESTED QUERY03

However, in other queries mentioned above, have same number of accessing rows before tunning process and the after the tunning process. Therefore, it can conclude that those queries were already optimized.