



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

Abstract.....	2
Preface.....	3
Table of contents.....	4
List of figures.....	5
1 Introduction.....	7
2 CHAPTER 1: REQUIREMENT ANALYSIS.....	8
2.1 Requirement analysis.....	8
2.1.1 Functional requirements.....	8
2.1.2 Data Requirements	9
3 CHAPTER2:CONCEPTUALDESIGN.....	10
3.1 Conceptual Design.....	10
3.1.1 ER Diagram	10
3.1.2 UML MODEL	11
4 CHAPTER 3:IMPLEMETATION.....	12
4.1 Normalization.....	12
4.2 MySQL Implementation	12
4.2.1 Creating the schema (COMPUTERS).....	12
4.2.2 Table Creation	13
4.2.3 Inserting data into Tables.....	19
4.2.4 Update data from the Tables	24
4.2.5 Delete data from the Tables.....	26
5 CHAPTER 4: TRANSACTIONS.....	27
5.1 Simple queries.....	27
5.2 Complex queries.....	32
6 CHAPTER 5: TUNING.....	37

List of figures

FIGURE 1 ER MODEL	10
FIGURE 2 UML MODEL	11
FIGURE 3 1ST NORMAL FORM	12
FIGURE 4 DATABASE CREATION	12
FIGURE 5 CREATING LAPTOP TABLE	13
FIGURE 6 WARRANTY TABLE	13
FIGURE 7 CUSTOMER TABLE	14
FIGURE 8 PAYMENT TABLE	14
FIGURE 9 REVIEWS TABLE	15
FIGURE 10 SELLER TABLE	15
FIGURE 11 :SELLER_DETAILS TABLE	16
FIGURE 12 ORDERS TABLE	16
FIGURE 13 INVOICE TABLE	17
FIGURE 14 SHIPPING TABLE	17
FIGURE 15 SHIPPING_DETAILS TABLE	18
FIGURE 16 INSERTING INTO THE LAPTOP TABLE	19
FIGURE 17 INSERTING INTO THE WARRANTY TABLE	19
FIGURE 18 INSERTING INTO THE CUSTOMER TABLE	20
FIGURE 19 INSERTING INTO THE PAYMENT TABLE	20
FIGURE 20 INSERTING INTO THE REVIEWS TABLE	21
FIGURE 21 INSERTING INTO THE SELLER AND SELLER_DETAILS TABLE	21
FIGURE 22 INSERTING INTO THE ORERS TABLE	22
FIGURE 23 INSERTING INTO THE SHIPPING AND SHIPPING_DETAILS TABLE	22
FIGURE 24 INSERTING INTO THE INVOICE TABLE	23
FIGURE 25:UPDATING DATA FROM TABLES-1	24
FIGURE 26:UPDATING DATA FROM TABLES-2	24
FIGURE 27:UPDATING DATA FROM TABLES-3	25
FIGURE 28:UPDATING DATA FROM TABLES-4	25
FIGURE 29:DELETE DATA FROM TABLES-1	26
FIGURE 30:DELETE DATA FROM TABLES-2	26
FIGURE 31:RETRIEVING ALL DATA FROM LAPTOP TABLE	27
FIGURE 32: RETRIEVING SELECTED DATA FROM PAYMENT TABLE	27
FIGURE 33:SELECT OPERATION OF SHIPPING DETAILS TABLE	28
FIGURE 34:CARTESIAN PRODUCT OF CUSTOMER AND ORDERS	28
FIGURE 35:CREATING USER VIEW	29
FIGURE 36:RENAME OPERATION	29
FIGURE 37:USING AGGREGATION FUNCTIONS	30
FIGURE 38:ORDER BY OPERATION	30
FIGURE 39:LIKE OPERATION	31
FIGURE 40:UNION	32
FIGURE 41:INTERSECTION	32
FIGURE 42:SET DIFFERENCE	33
FIGURE 43:DIVISION	33
FIGURE 44:NATURAL JOIN	34
FIGURE 45:INNER JOIN	34
FIGURE 46:LEFT OUTER JOIN	35
FIGURE 47 RIGHT OUTER JOIN	35
FIGURE 48:FULL OUTR JOIN	36
FIGURE 49:OUTER UNION	36
FIGURE 50:BEFORE TUNING DIFFERENCE OPERATION	37
FIGURE 51:AFTER TUNING DIFFERENCE OPERATION	37
FIGURE 52: BEFORE TUNING UNION OPERATION	38
FIGURE 53: AFTER TUNING UNION OPERATION	38
FIGURE 54 BEFORE TUNING INTERSECTION OPERATION	39

FIGURE 55 AFTER TUNING INTERSECTION OPERATION	39
FIGURE 56 BEFORE TUNING INNER JOIN OPERATION	40
FIGURE 57 AFTER TUNING INNER JOIN OPERATION	40
FIGURE 58 BEFORE TUNING SELECT	41
FIGURE 59 AFTER TUNING SELECT	41
FIGURE 60 BEFORE TUNING RIGHT OUTER JOIN	42
FIGURE 61 AFTER TUNING RIGHT OUTER JOIN	42
FIGURE 62 BEFORE TUNING NATURAL JOIN.....	43
FIGURE 63 AFTER TUNING NATURAL JOIN	43
FIGURE 64 BEFORE TUNING NESTED QUERY01.....	44
FIGURE 65 AFTER TUNING NESTED QUERY01	44
FIGURE 66 BEFORE TUNING NESTED QUERY02.....	45
FIGURE 67 AFTER TUNING NESTED QUERY02	45
FIGURE 68 BEFORE TUNING NESTED QUERY03.....	46
FIGURE 69 AFTER TUNING NESTED QUERY03	46

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: REQUIREMENT 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

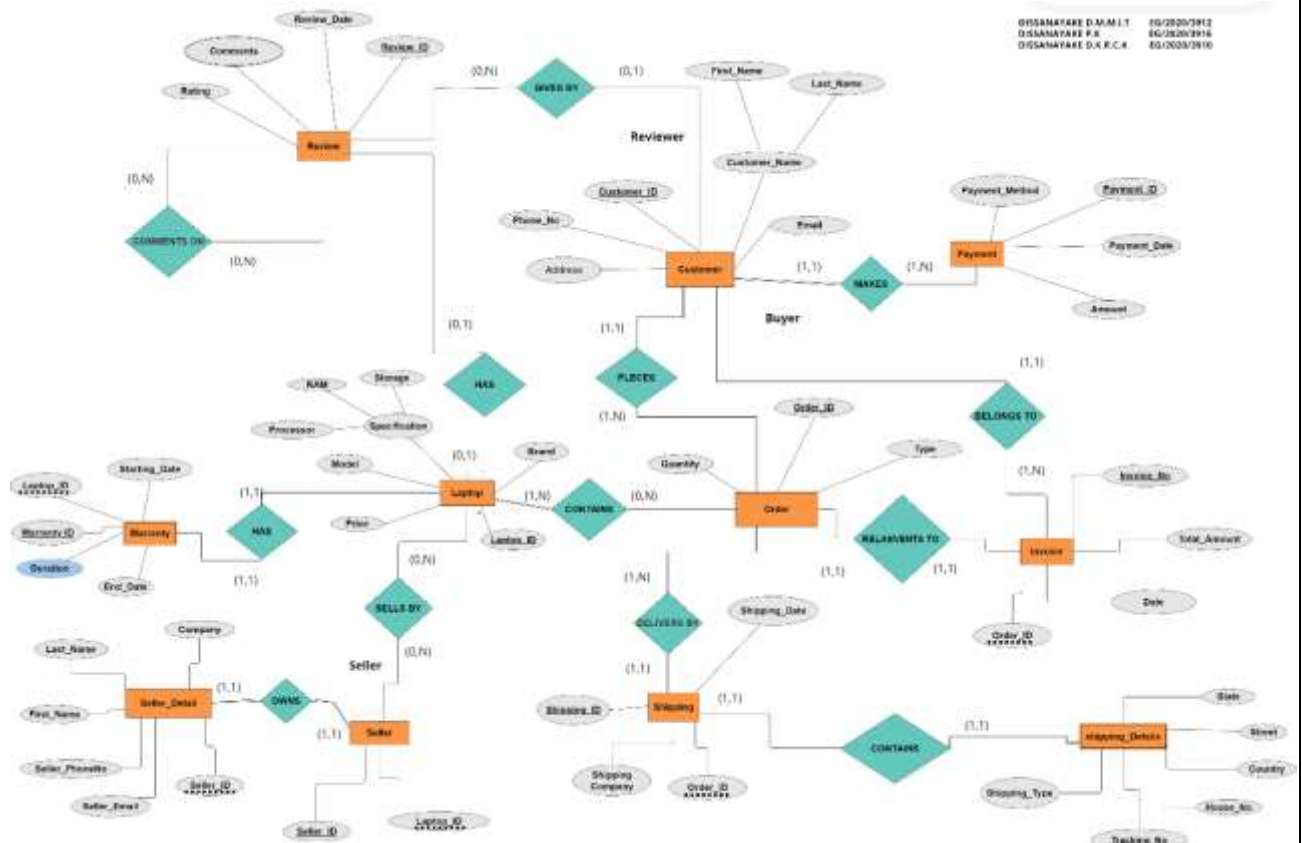


FIGURE 1 ER MODEL

3.1.2 UML MODEL

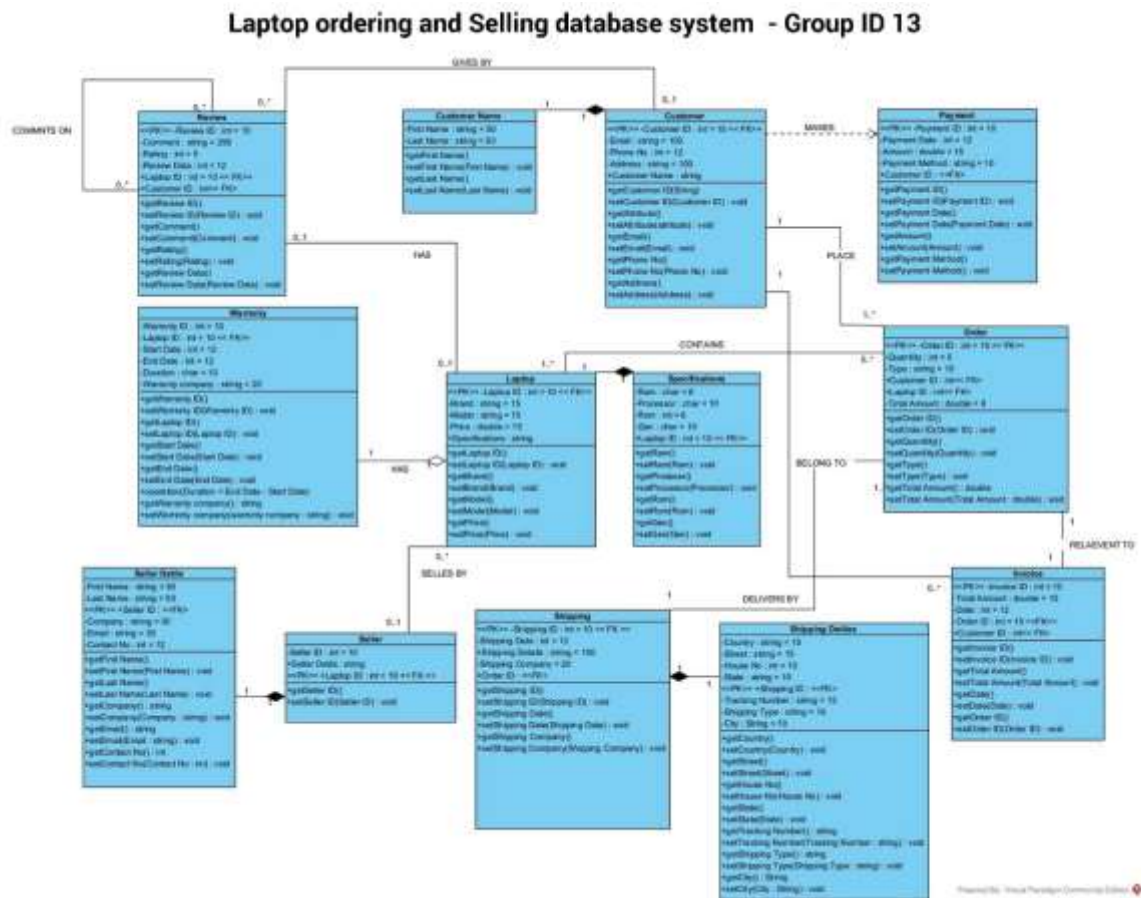


FIGURE 2 UML MODEL

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

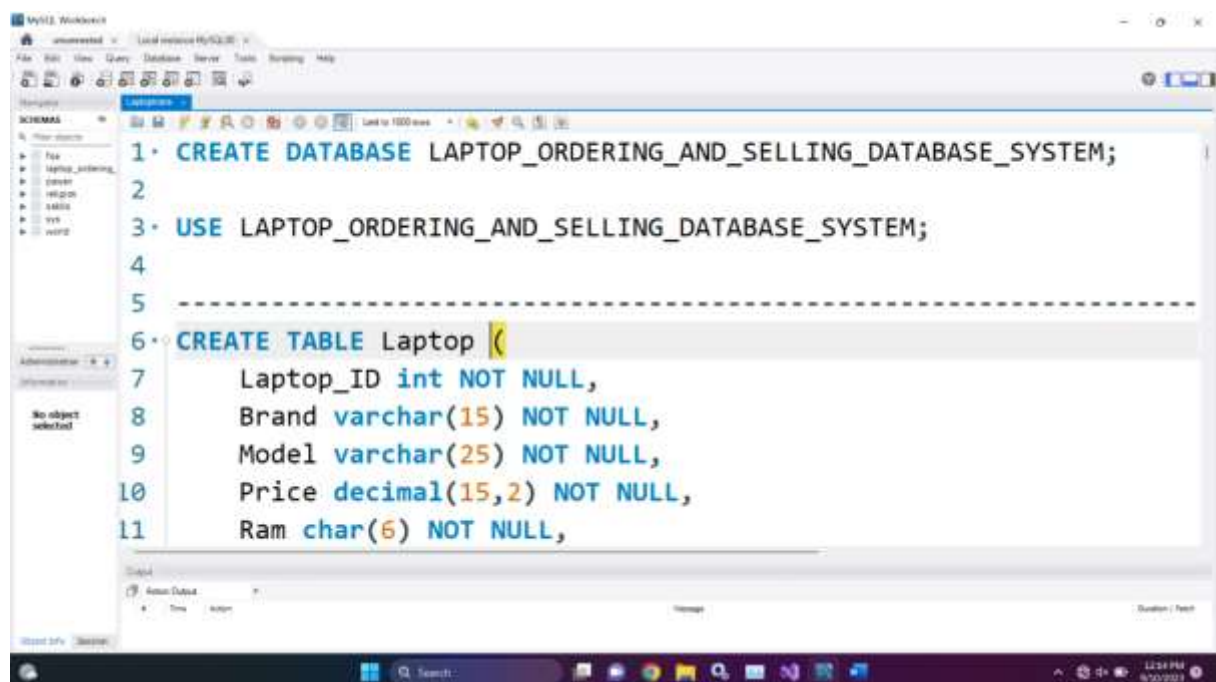
FIGURE 3 1ST NORMAL FORM

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

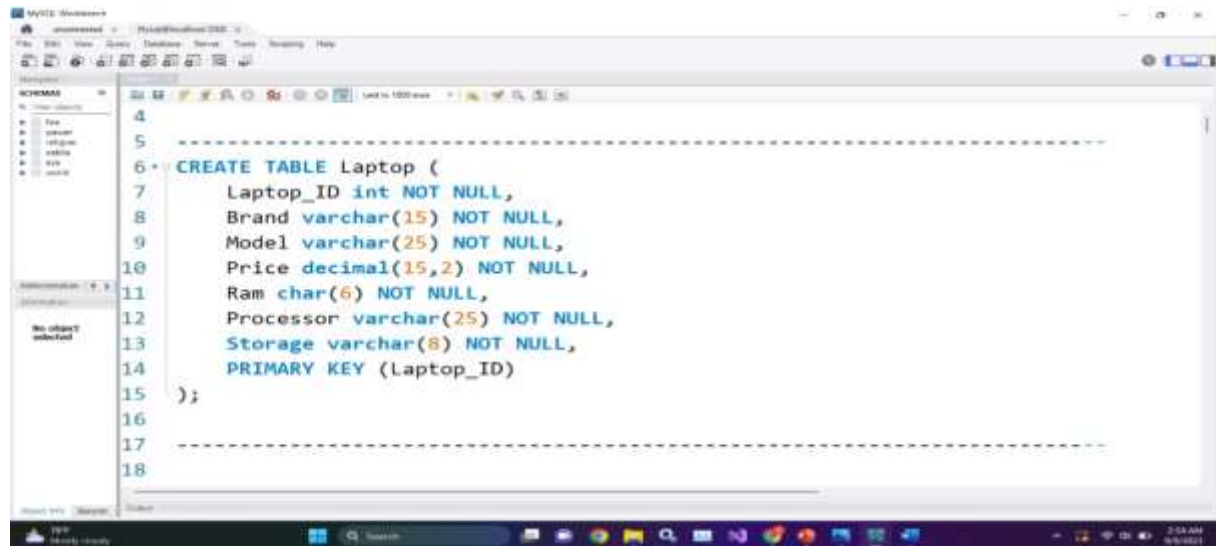
4.2.1 Creating the schema (COMPUTERS)



```
1• CREATE DATABASE LAPTOP_ORDERING_AND_SELLING_DATABASE_SYSTEM;
2
3• USE LAPTOP_ORDERING_AND_SELLING_DATABASE_SYSTEM;
4
5
6• CREATE TABLE Laptop (
7    Laptop_ID int NOT NULL,
8    Brand varchar(15) NOT NULL,
9    Model varchar(25) NOT NULL,
10   Price decimal(15,2) NOT NULL,
11   Ram char(6) NOT NULL,
```

FIGURE 4 DATABASE CREATION

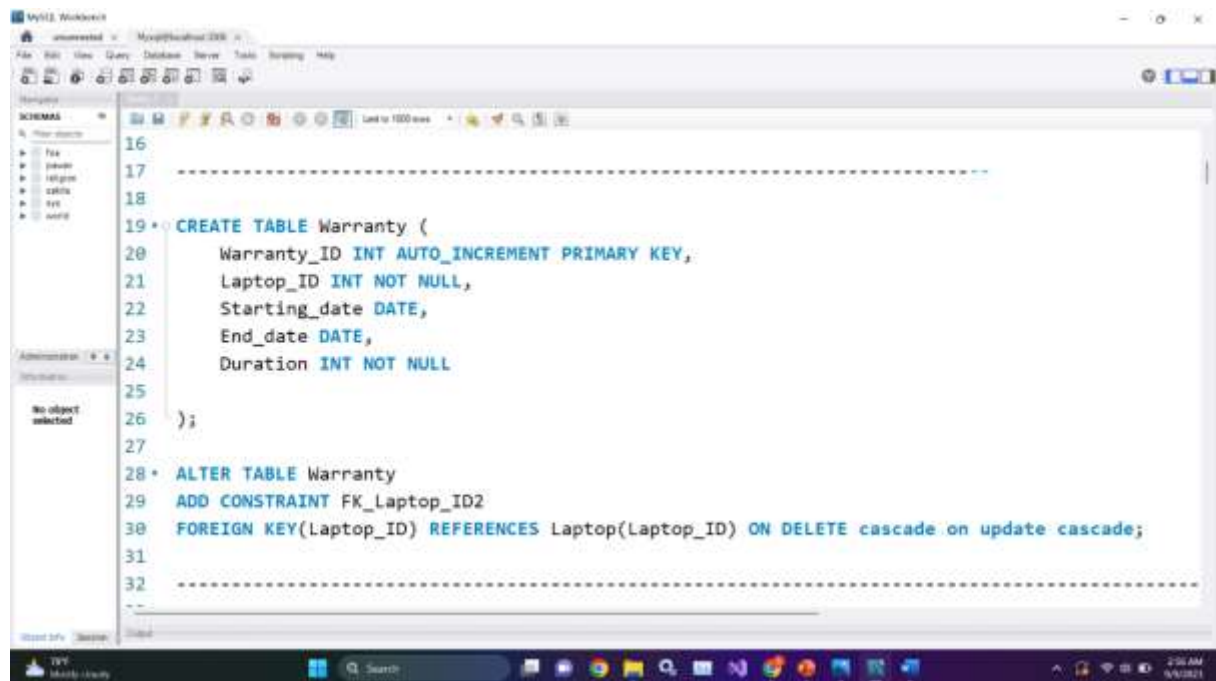
4.2.2 Table Creation



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a list of databases: 'Test', 'information', 'mysql', 'performance_schema', 'sys', and 'world'. The 'Test' database is selected. The main editor window displays the following SQL code for creating the 'Laptop' table:

```
4  
5 -----  
6 * CREATE TABLE Laptop (  
7     Laptop_ID int NOT NULL,  
8     Brand varchar(15) NOT NULL,  
9     Model varchar(25) NOT NULL,  
10    Price decimal(15,2) NOT NULL,  
11    Ram char(6) NOT NULL,  
12    Processor varchar(25) NOT NULL,  
13    Storage varchar(8) NOT NULL,  
14    PRIMARY KEY (Laptop_ID)  
15 );  
16  
17 -----  
18
```

FIGURE 5 CREATING LAPTOP TABLE



The screenshot shows the MySQL Workbench interface. The main editor window displays the following SQL code for creating the 'Warranty' table and adding a foreign key constraint:

```
16  
17 -----  
18  
19 * CREATE TABLE Warranty (  
20     Warranty_ID INT AUTO_INCREMENT PRIMARY KEY,  
21     Laptop_ID INT NOT NULL,  
22     Starting_date DATE,  
23     End_date DATE,  
24     Duration INT NOT NULL  
25 );  
26  
27  
28 * ALTER TABLE Warranty  
29 ADD CONSTRAINT FK_Laptop_ID2  
30 FOREIGN KEY(Laptop_ID) REFERENCES Laptop(Laptop_ID) ON DELETE cascade on update cascade;  
31  
32 -----
```

FIGURE 6 WARRANTY TABLE

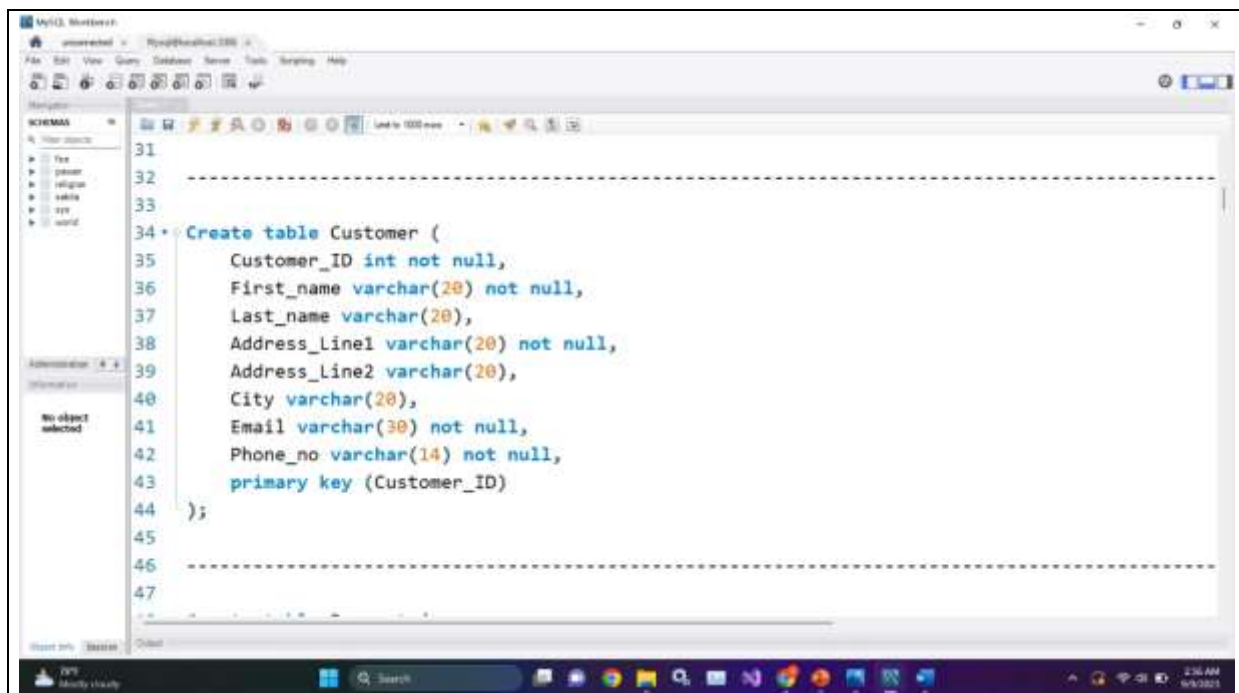


FIGURE 7 CUSTOMER TABLE

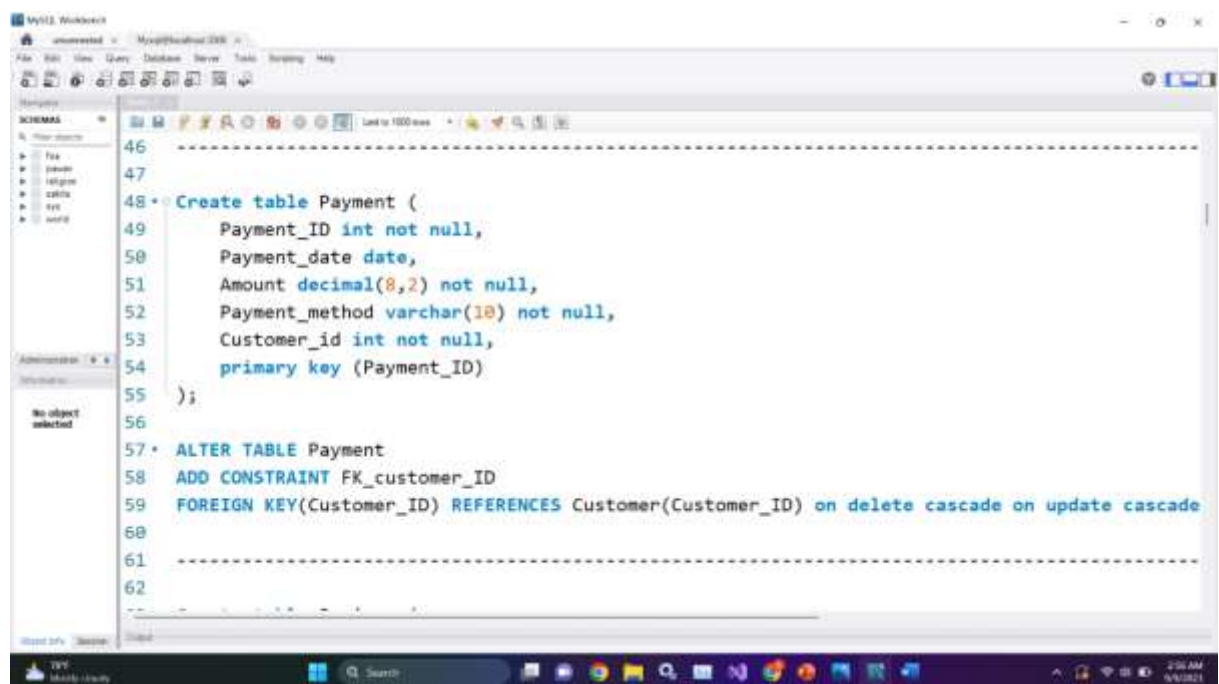


FIGURE 8 PAYMENT TABLE

The screenshot shows the MySQL Workbench interface with a SQL script editor. The script defines the 'Reviews' table with columns: Review_ID (int, not null, primary key), Comments (varchar(256)), Rating (int), Review_date (date), Laptop_ID (int, not null, foreign key to Laptop), and Customer_ID (int, not null, foreign key to customer). The script includes the following SQL statements:

```

62
63 * Create table Reviews (
64     Review_ID int not null,
65     Comments varchar(256),
66     Rating int,
67     Review_date date,
68     Laptop_ID int not null,
69     Customer_ID int not null,
70
71     primary key (Review_ID)
72 );
73
74 * ALTER TABLE reviews
75 ADD CONSTRAINT fk_laptopid
76 FOREIGN KEY (Laptop_id) REFERENCES Laptop (Laptop_id) ON DELETE cascade on update cascade,
77 ADD CONSTRAINT fk_customer
78 FOREIGN KEY (customer_id)REFERENCES customer(customer_id) ON delete cascade on update cascade;
79

```

FIGURE 9 REVIEWS TABLE

The screenshot shows the MySQL Workbench interface with a SQL script editor. The script defines the 'Seller' table with columns: laptop_ID (int, not null, foreign key to Laptop), seller_ID (int, not null, primary key), and a unique constraint on Seller_ID. The script includes the following SQL statements:

```

76 FOREIGN KEY (Laptop_id) REFERENCES Laptop (Laptop_id) ON DELETE cascade on update cascade,
77 ADD CONSTRAINT fk_customer
78 FOREIGN KEY (customer_id)REFERENCES customer(customer_id) ON delete cascade on update cascade;
79
80 -----
81
82 * Create table Seller (
83     laptop_ID int not null,
84     seller_ID int not null,
85     primary key (laptop_ID)
86 );
87
88 * ALTER TABLE Seller
89 ADD CONSTRAINT fk_seller
90 FOREIGN KEY (Laptop_id) REFERENCES Laptop (Laptop_id) ON DELETE cascade on update cascade;
91 * alter table Seller add constraint uniq unique(Seller_ID);
92 -----
93

```

FIGURE 10 SELLER TABLE

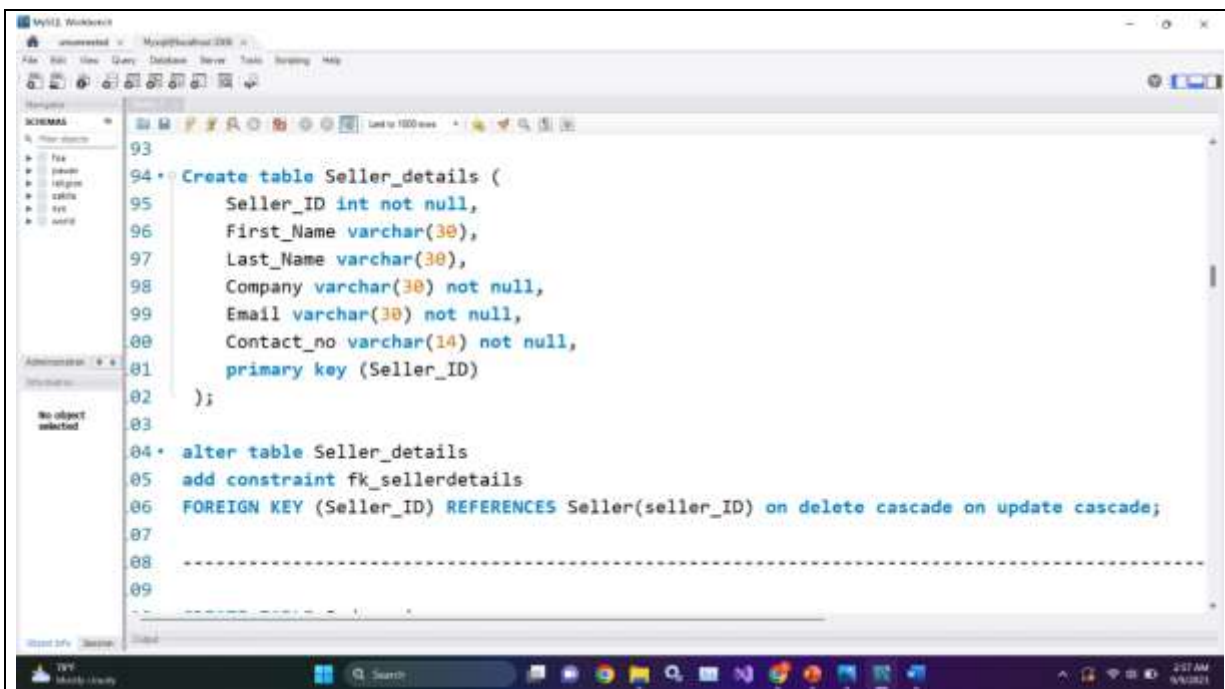


FIGURE 11 :SELLER_DETAILS TABLE

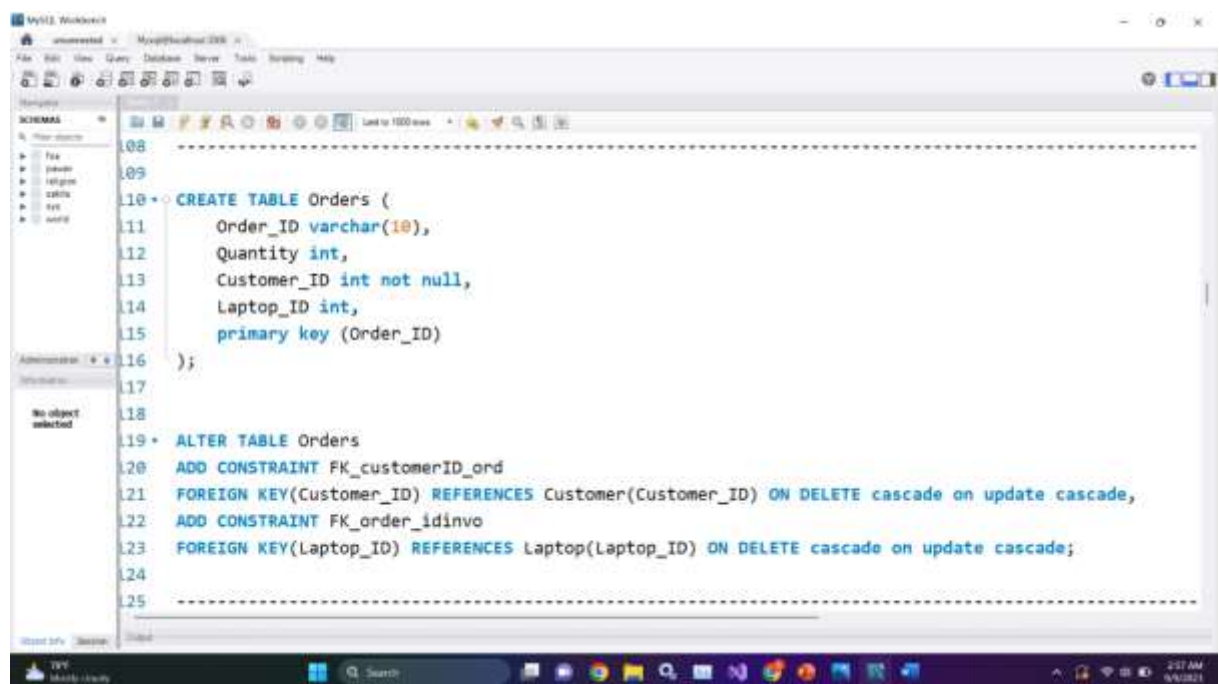


FIGURE 12 ORDERS TABLE

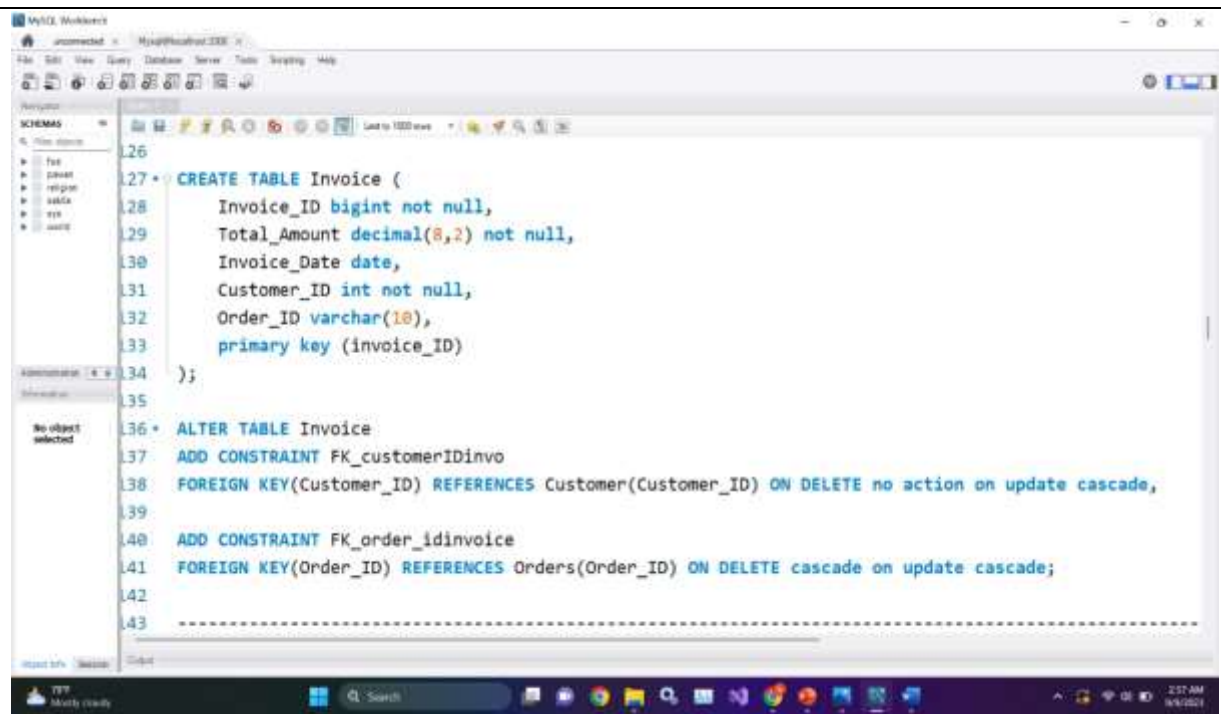


FIGURE 13 INVOICE TABLE

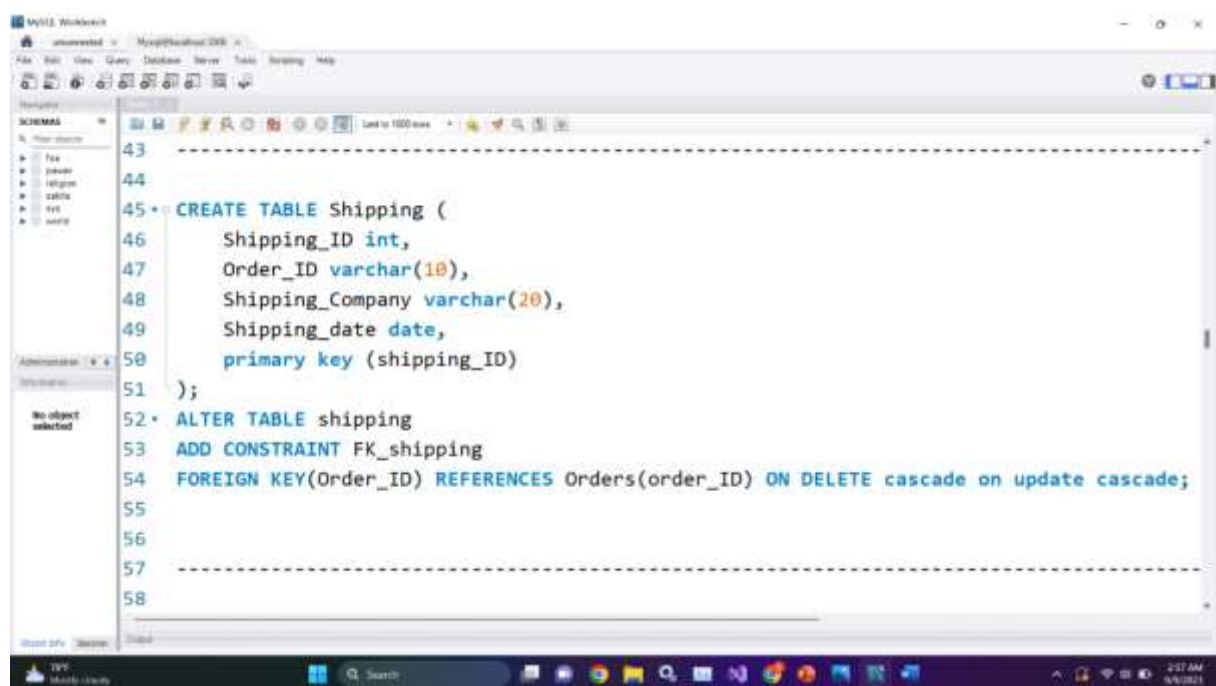


FIGURE 14 SHIPPING TABLE

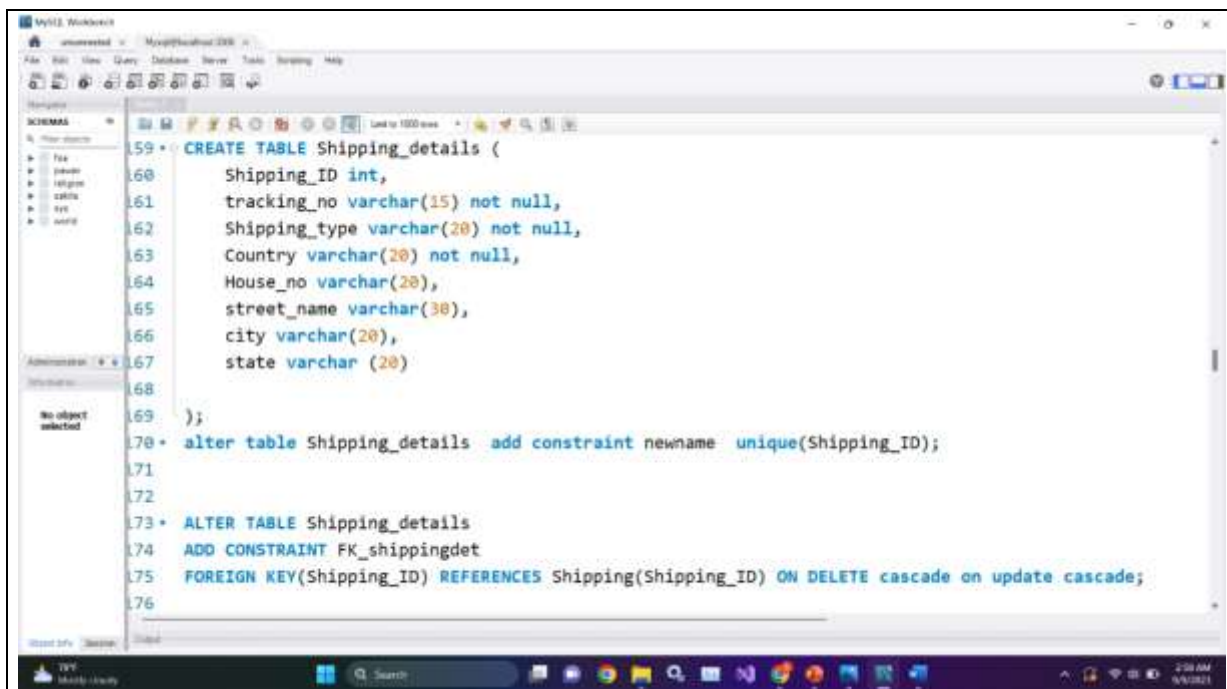
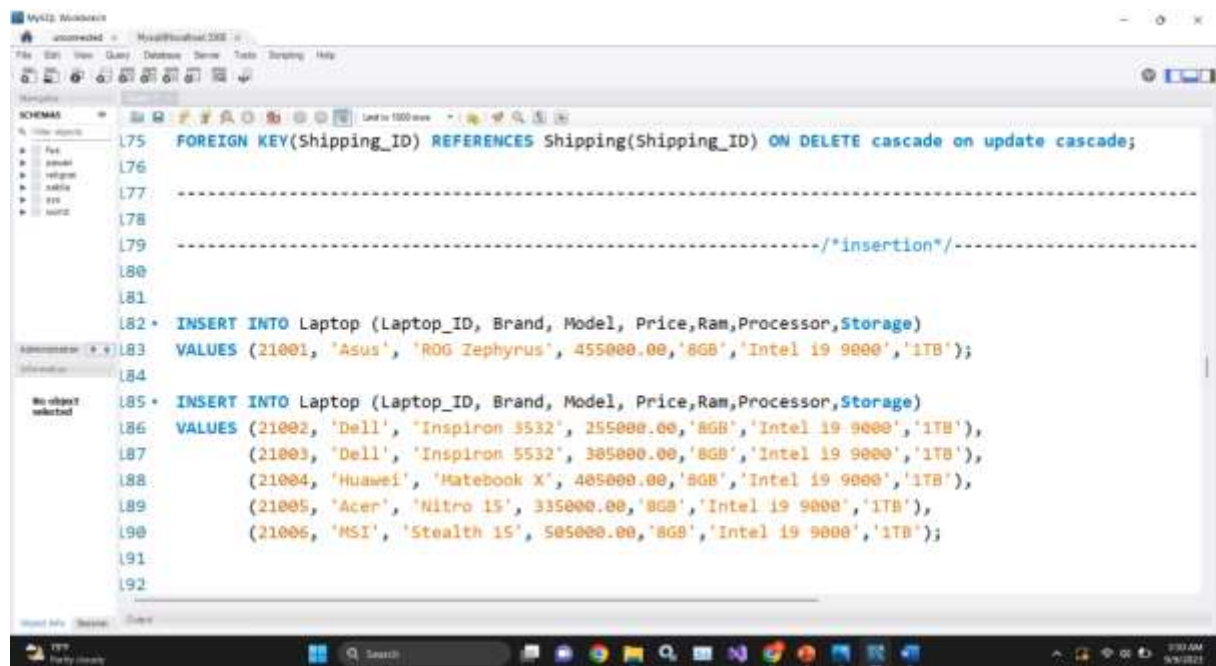


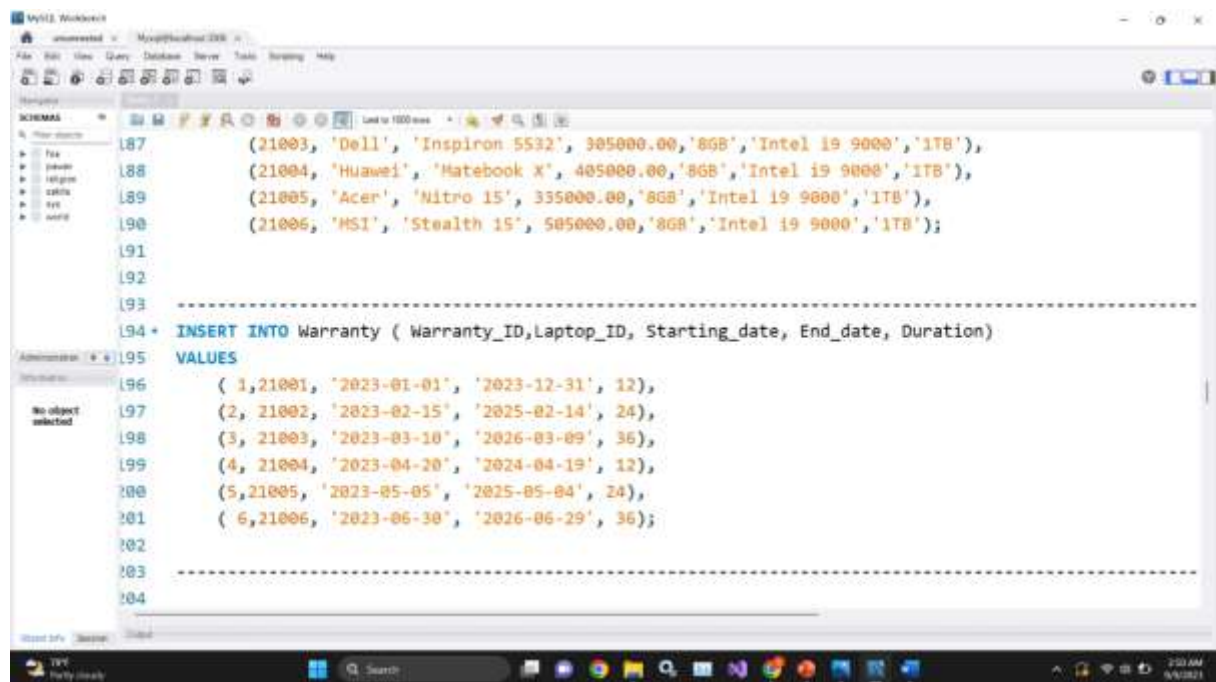
FIGURE 15 SHIPPING_DETAILS TABLE

4.2.3 Inserting data into Tables



The screenshot shows the MySQL Workbench interface with a SQL editor. The code defines a foreign key for the Laptop table and inserts six rows of data. The foreign key constraint is: `FOREIGN KEY(Shipping_ID) REFERENCES Shipping(Shipping_ID) ON DELETE cascade on update cascade;`. The insert statements are: `INSERT INTO Laptop (Laptop_ID, Brand, Model, Price, Ram, Processor, Storage) VALUES (21001, 'Asus', 'ROG Zephyrus', 455000.00, '8GB', 'Intel i9 9000', '1TB');` and `INSERT INTO Laptop (Laptop_ID, Brand, Model, Price, Ram, Processor, Storage) VALUES (21002, 'Dell', 'Inspiron 5532', 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



The screenshot shows the MySQL Workbench interface with a SQL editor. The code inserts six rows of data into the Warranty table. The insert statement is: `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

The screenshot shows the MySQL Workbench interface with a SQL editor containing several INSERT statements for the Customer table. The left sidebar shows the 'SCHEMAS' list with 'mydb' selected. The main editor area displays the following SQL code:

```

202
203
204
205 * INSERT INTO Customer (Customer_ID, First_name, Last_name, Address_line1, Address_line2, City, Email, Phone_no)
206 VALUES (1001, 'John', 'Doe', '123 Main St', 'Apt 4B', 'New York', 'john.doe@gmail.com', 94714567890);
207
208 * INSERT INTO Customer
209 VALUES (1002, 'Jane', 'Smith', '456 Elm St', NULL, 'Los Angeles', 'jane.smith@gmail.com', 94787654321);
210
211 * INSERT INTO Customer
212 VALUES (1003, 'Robert', 'Johnson', '789 Oak St', 'Suite 12', 'Chicago', 'robert.johnson@gmail.com', 94772891234);
213
214 * INSERT INTO Customer
215 VALUES (1004, 'Emily', 'Brown', '101 Pine St', NULL, 'San Francisco', 'emily.brown@gmail.com', 94726789012);
216
217 * INSERT INTO Customer
218 VALUES (1005, 'Michael', 'Wilson', '222 Birch St', 'Apt 3C', 'Boston', 'michael.wilson@gmail.com', 94775678901);
219
220 * INSERT INTO Customer
221 VALUES (1006, 'Sarah', 'Davis', '555 Maple St', NULL, 'Seattle', 'sarah.davis@gmail.com', 94709012345);
222
223

```

FIGURE 18 INSERTING INTO THE CUSTOMER TABLE

The screenshot shows the MySQL Workbench interface with a SQL editor containing two INSERT statements. The left sidebar shows the 'SCHEMAS' list with 'mydb' selected. The main editor area displays the following SQL code:

```

223
224
225 * INSERT INTO Payment (Payment_ID, Payment_date, Amount, Payment_method, Customer_id)
226 VALUES
227 (8001, '2023-09-04', 456600.00, 'Credit', 1001),
228 (8002, '2023-09-05', 355575.00, 'Cash', 1002),
229 (8003, '2023-09-06', 255120.00, 'Debit', 1003),
230 (8004, '2023-09-07', 175550.00, 'Credit', 1004),
231 (8005, '2023-09-08', 195000.00, 'Cash', 1005),
232 (8006, '2023-09-09', 600000.00, 'Debit', 1006);
233
234
235
236 * INSERT INTO Reviews (Review_ID, Comments, Rating, Review_date, Laptop_ID, Customer_ID)
237 VALUES
238 (101, 'Great laptop', 5, '2023-09-01', 21001, 1001),
239 (103, 'Excellent build quality', 4, '2023-09-03', 21003, 1003),

```

FIGURE 19 INSERTING INTO THE PAYMENT TABLE

```

32      (8886, '2023-09-09', 600000.00, 'Debit', 1006);
33
34  -----
35
36  • INSERT INTO Reviews (Review_ID, Comments, Rating, Review_date, Laptop_ID, Customer_ID)
37  VALUES
38      (101, 'Great laptop', 5, '2023-09-01', 21001, 1001),
39      (103, 'Excellent build quality', 4, '2023-09-03', 21003, 1003),
40      (104, 'Fast and powerful', 5, '2023-09-04', 21004, 1004),
41      (105, 'Poor battery life', 2, '2023-09-05', 21005, 1005),
42      (106, 'Good value for the price', 4, '2023-09-06', 21006, 1006);
43
44  -----
45  • INSERT INTO Seller (Laptop_ID,Seller_ID)
46  VALUES
47      (21001, 21)

```

FIGURE 20 INSERTING INTO THE REVIEWS TABLE

```

250  -----
251
252  • INSERT INTO Seller (Laptop_ID,Seller_ID)
253  VALUES
254      (21001,21),
255      (21003,22),
256      (21004,23),
257      (21005,24),
258      (21006,25);
259
260  • Insert into seller_details (Seller_ID, First_Name,Last_Name,Company,Email,contact_no)
261  values
262      (21, 'Adam', 'Surt', 'ASD Inc.', 'Adam@yahoo.com', 987654321),
263      (22, 'Chalvi', 'Puri', 'CJB Corporation', 'Charlie@gmail.com', 94786754321),
264      (23, 'Sumoh', 'Himal', 'SHP Ltd.', 'sumoh@hotmail.com', 9478654321),
265      (24, 'Surbh', 'Jade', 'HPE Industries', 'surbh@gmail.com', 88775446544);
266
267  -----

```

Time	Action	Message	Duration / Rows
14: 04:23:44	DELETE FROM Order WHERE (Order_ID = 300000)	1 record affected	0.002 sec
15: 04:23:44	DELETE FROM Invoice WHERE Invoice_ID = 3000	0 record affected	0.000 sec
16: 04:23:44	DELETE FROM Shipping WHERE Shipping_ID = 3000	0 record affected	0.000 sec
17: 04:23:44	DELETE FROM Shipping_details WHERE Shipping_ID = 3000	0 record affected	0.000 sec

FIGURE 21 INSERTING INTO THE SELLER AND SELLER_DETAILS TABLE

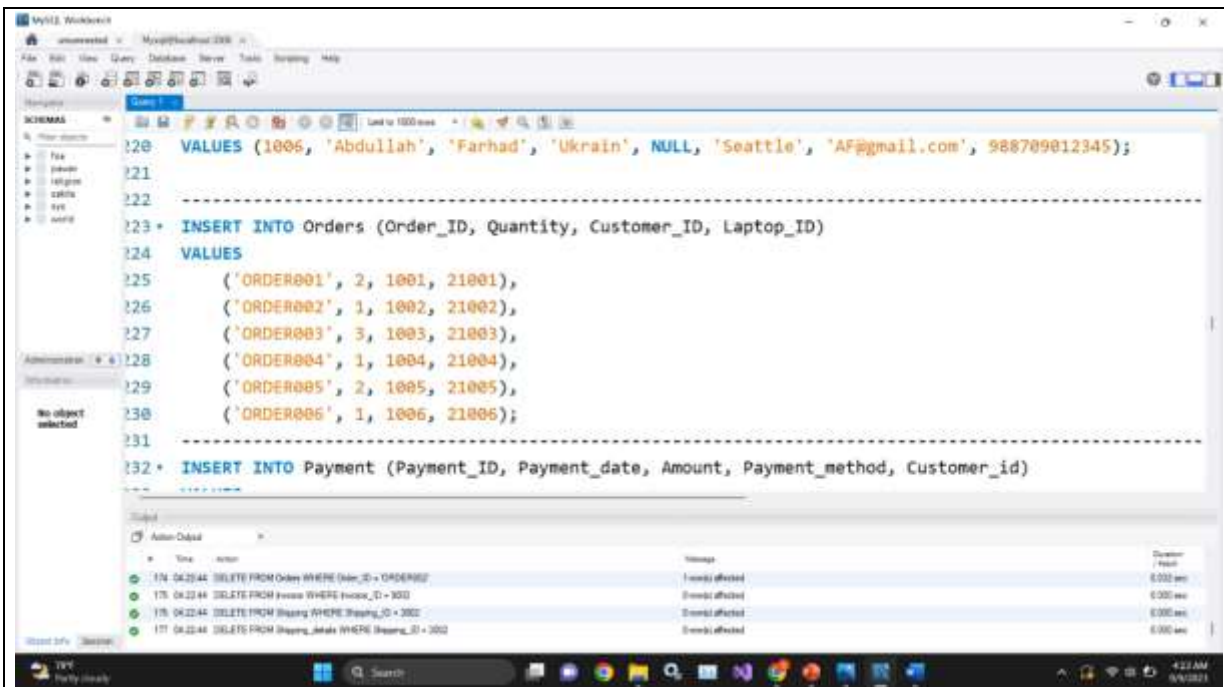


FIGURE 22 INSERTING INTO THE ORERS TABLE

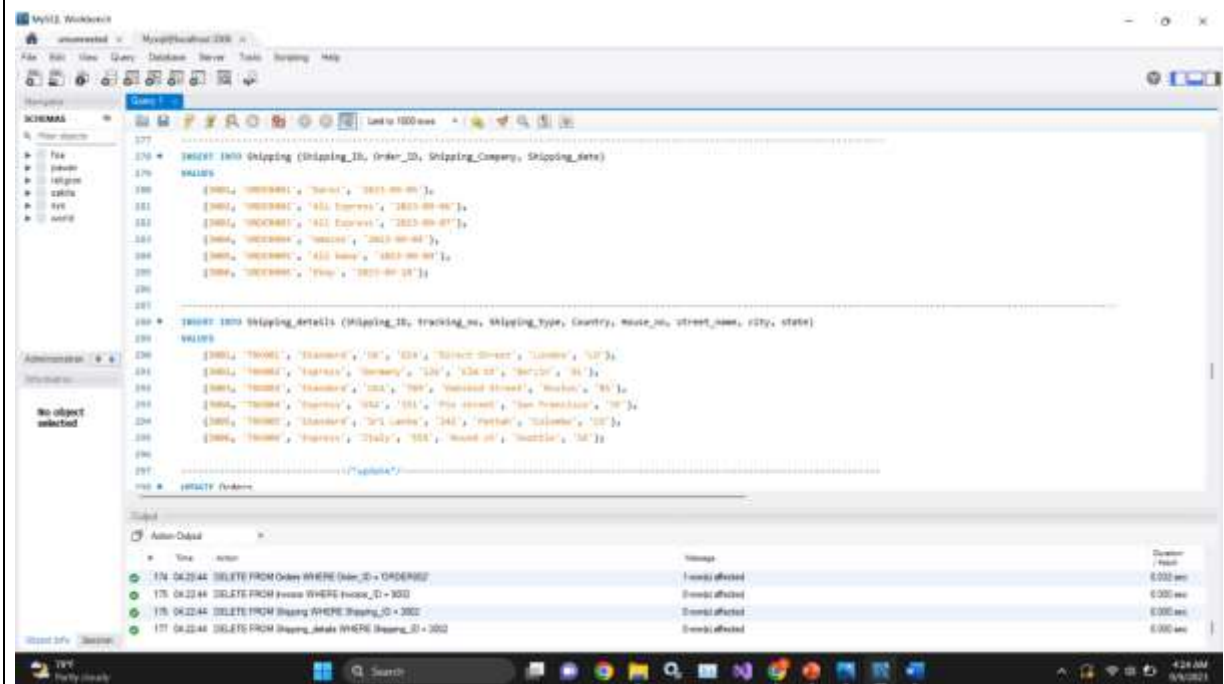


FIGURE 23 INSERTING INTO THE SHIPPING AND SHIPPING_DETAILS TABLE

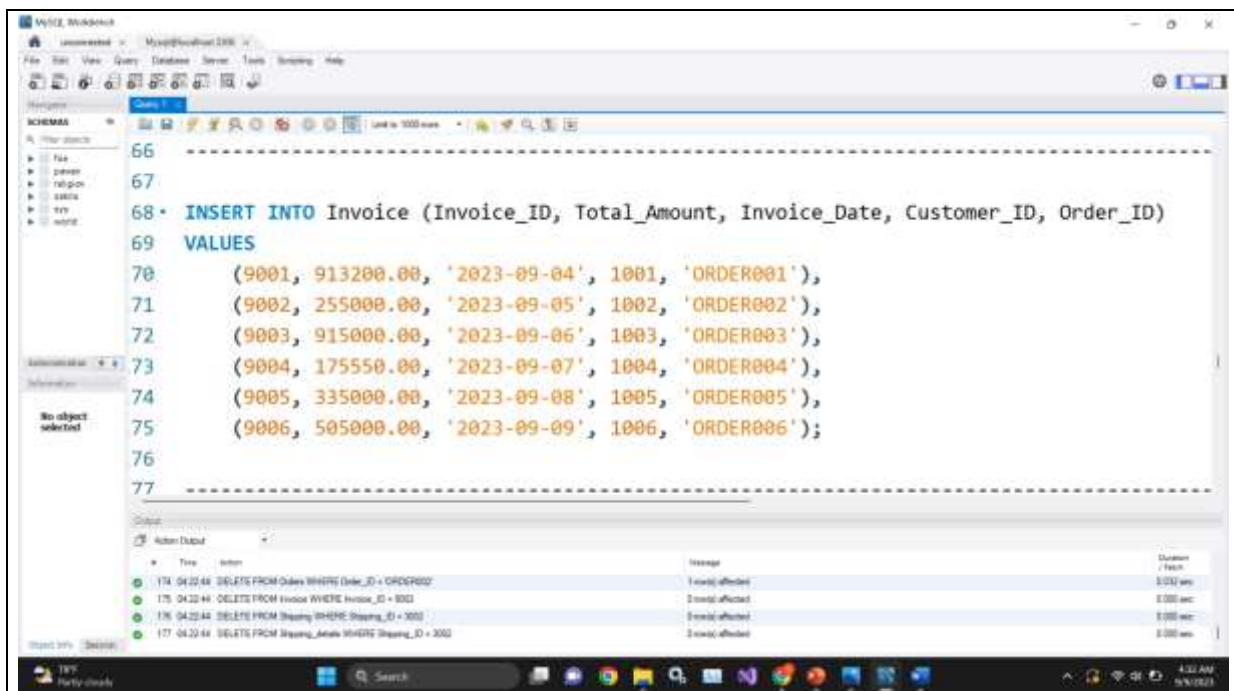


FIGURE 24 INSERTING INTO THE INVOICE TABLE

4.2.4 Update data from the Tables

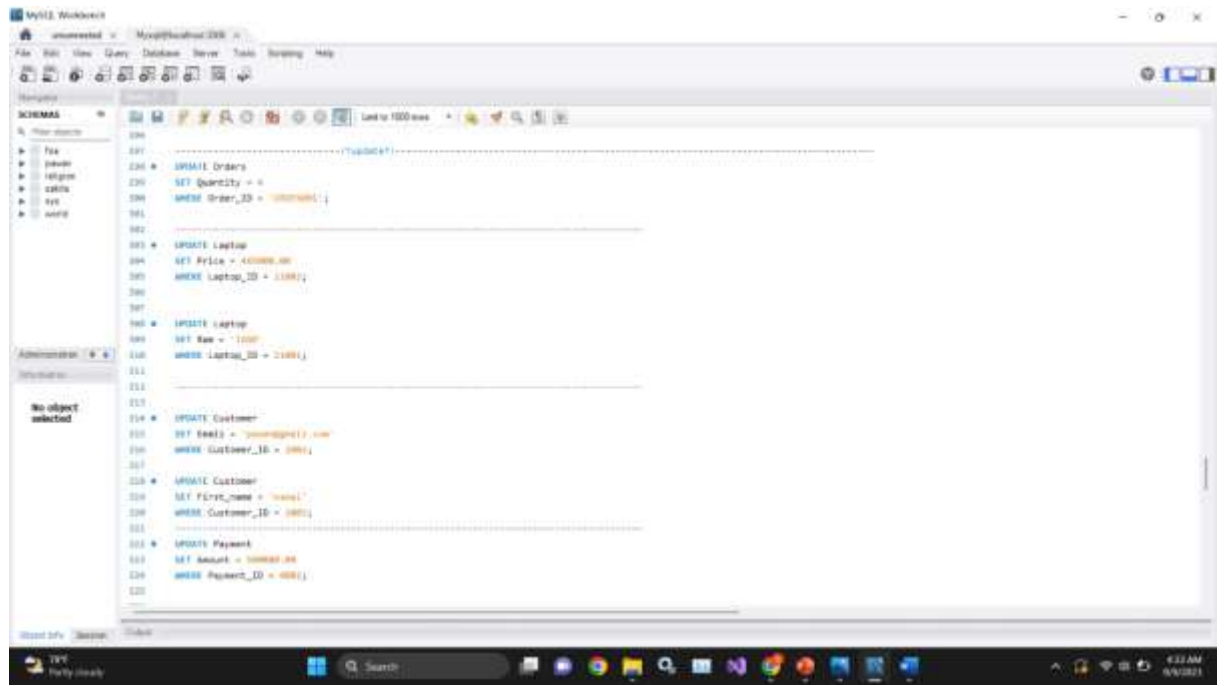


FIGURE 25:UPDATING DATA FROM TABLES-1

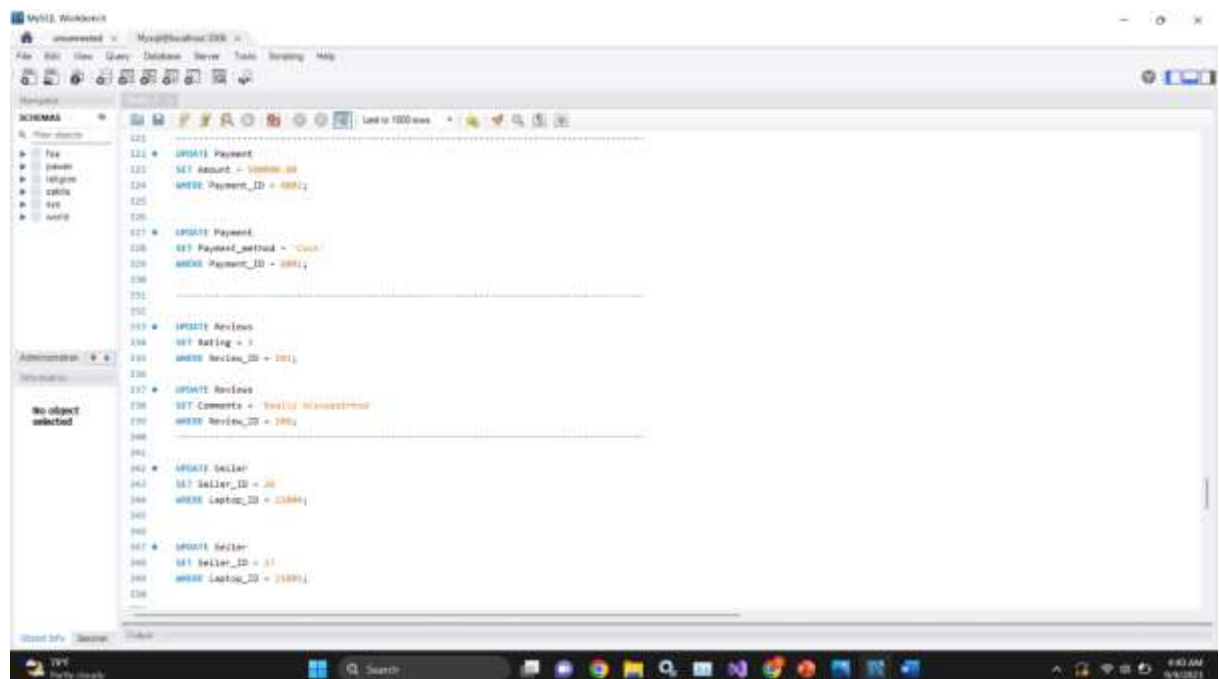


FIGURE 26:UPDATING DATA FROM TABLES-2

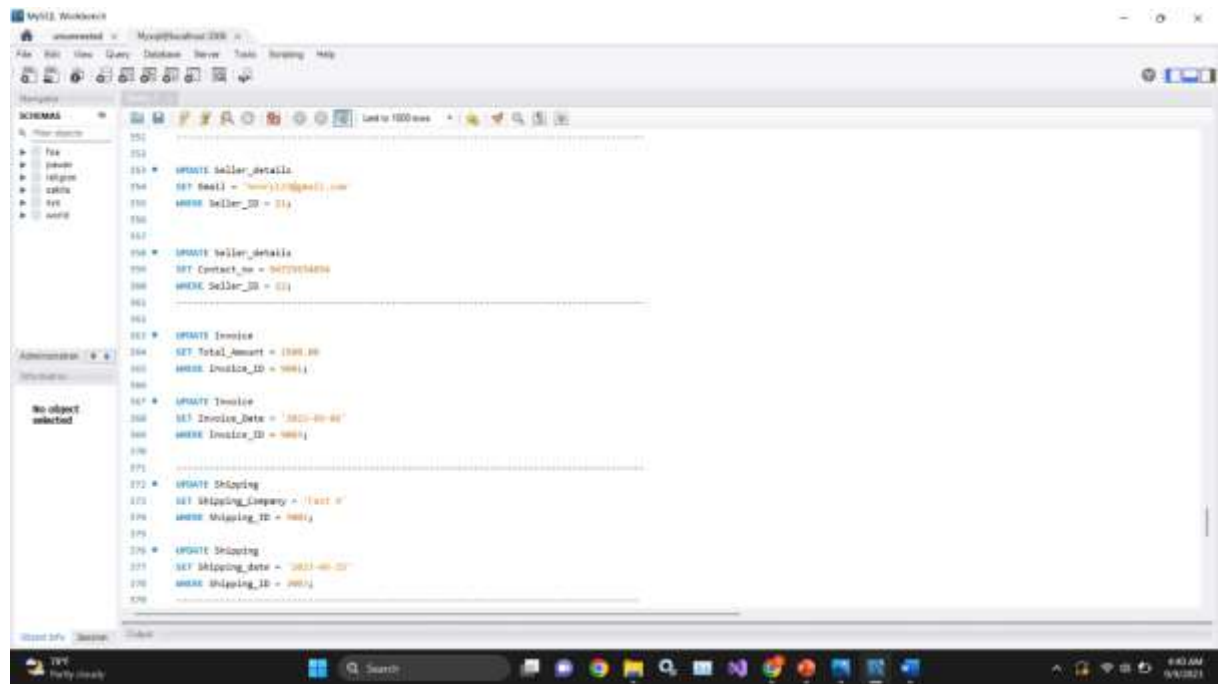


FIGURE 27:UPDATING DATA FROM TABLES-3

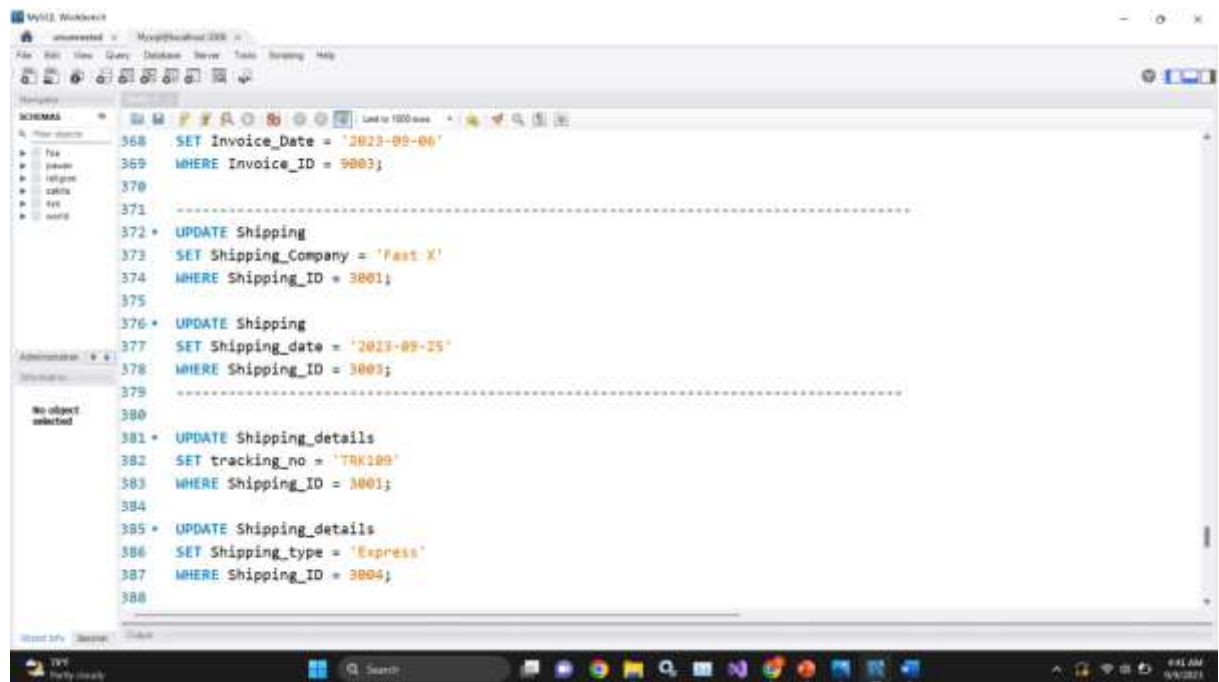


FIGURE 28:UPDATING DATA FROM TABLES-4

4.2.5 Delete data from the Tables

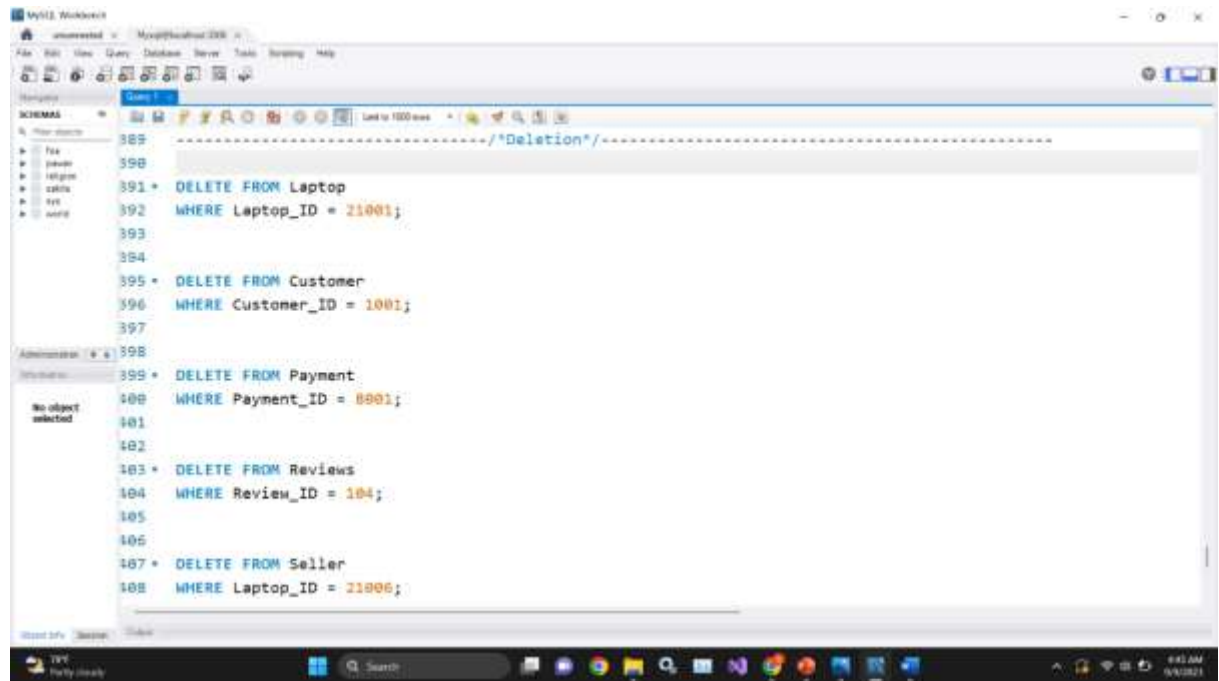


FIGURE 29:DELETE DATA FROM TABLES-1

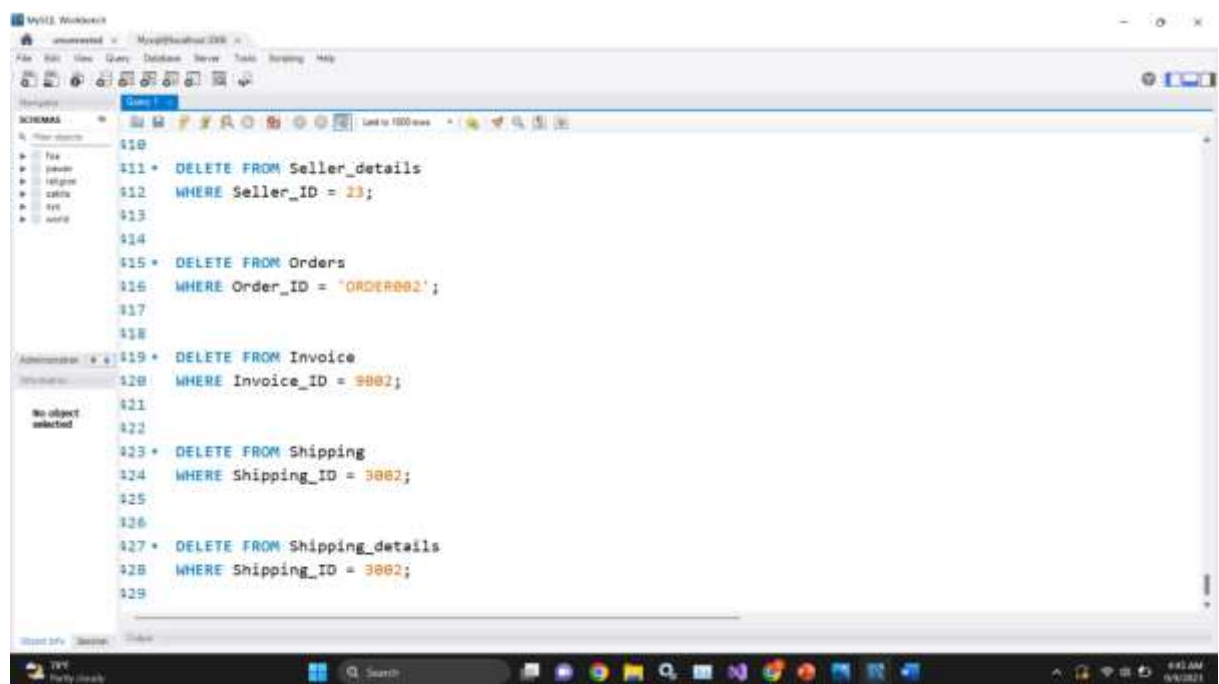
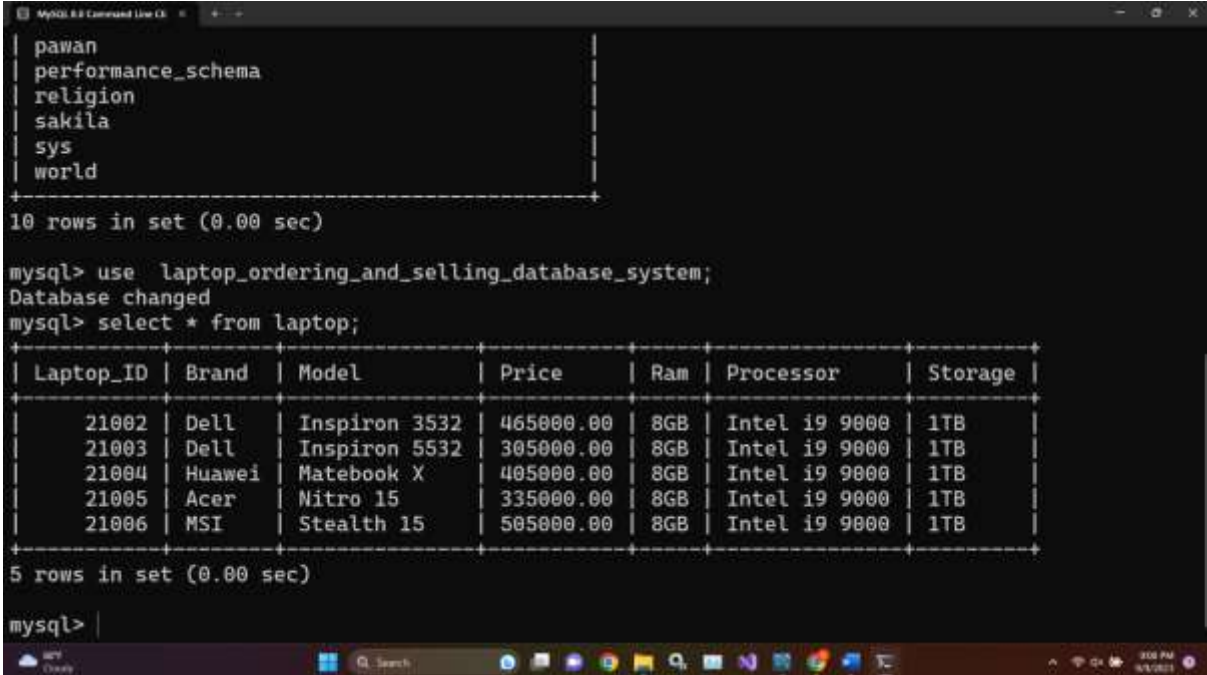


FIGURE 30:DELETE DATA FROM TABLES-2

5 CHAPTER 4: TRANSACTIONS

5.1 Simple queries



```
mysql> show databases;
+-----+
| pawan          |
| performance_schema |
| religion        |
| sakila          |
| sys             |
| world           |
+-----+
10 rows in set (0.00 sec)

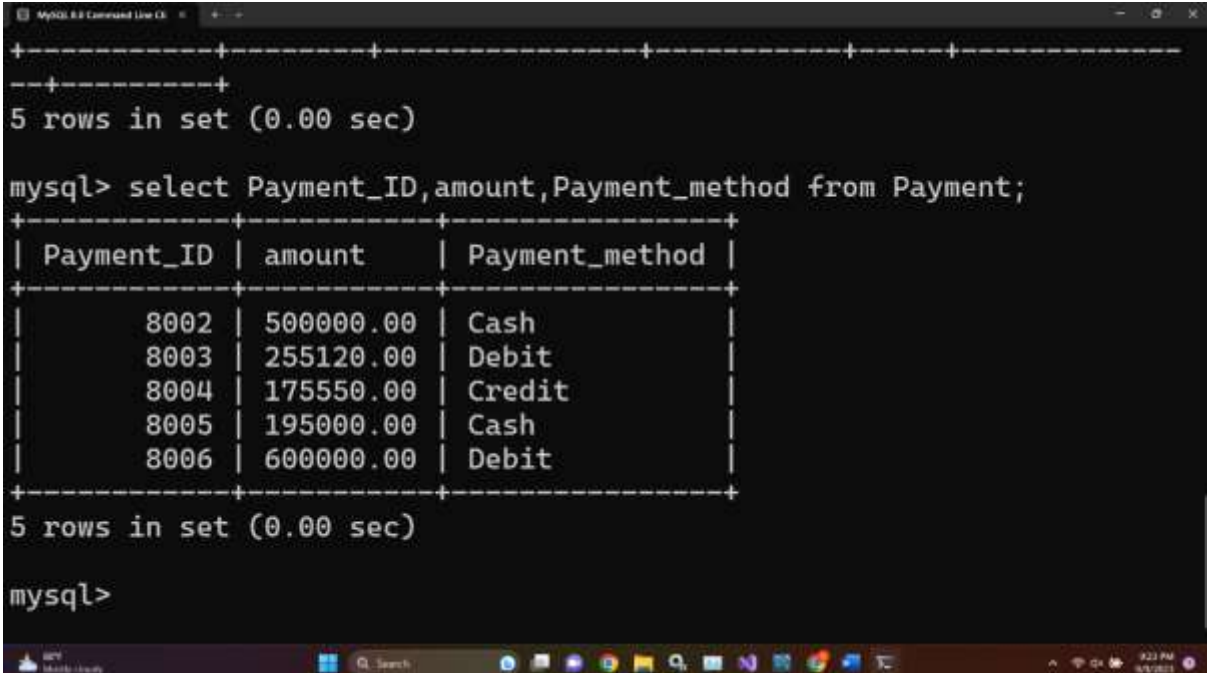
mysql> use laptop_ordering_and_selling_database_system;
Database changed
mysql> select * from laptop;
```

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



```
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	CD
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

MySQL 8.0 Command Line Client

Empty set (0.00 sec)

mysql> SELECT * FROM CUSTOMER CROSS JOIN ORDERS;

Customer_ID	First_name	Last_name	Address_Line1	Address_Line2	City	State	Phone_no	Order_ID	Quantity	Customer_ID	LastLog_ID	
1	John	Deo	111 Main Street	Apt 08	New York		johnd@example.com	100-123-4567	ORD0000	1	1	21000
1	John	Deo	111 Main Street	Apt 08	New York		johnd@example.com	100-123-4567	ORD0000	2	6	21000
1	John	Deo	111 Main Street	Apt 08	New York		johnd@example.com	100-123-4567	ORD0000	1	6	21000
1	John	Deo	111 Main Street	Apt 08	New York		johnd@example.com	100-123-4567	ORD0000	2	6	21000
1	John	Deo	111 Main Street	Apt 08	New York		johnd@example.com	100-123-4567	ORD0000	1	6	21000
2	John	Smith	008 Elm St		Los Angeles		johnd@example.com	323-987-6543	ORD0000	2	6	21000
2	John	Smith	008 Elm St		Los Angeles		johnd@example.com	323-987-6543	ORD0000	1	6	21000
2	John	Smith	008 Elm St		Los Angeles		johnd@example.com	323-987-6543	ORD0000	2	6	21000
2	John	Smith	008 Elm St		Los Angeles		johnd@example.com	323-987-6543	ORD0000	1	6	21000
2	John	Smith	008 Elm St		Los Angeles		johnd@example.com	323-987-6543	ORD0000	2	6	21000
3	Bob	Johnson	789 Oak St		Chicago		bobj@example.com	312-567-8901	ORD0000	1	6	21000
3	Bob	Johnson	789 Oak St		Chicago		bobj@example.com	312-567-8901	ORD0000	2	6	21000
3	Bob	Johnson	789 Oak St		Chicago		bobj@example.com	312-567-8901	ORD0000	1	6	21000
3	Bob	Johnson	789 Oak St		Chicago		bobj@example.com	312-567-8901	ORD0000	2	6	21000
3	Bob	Johnson	789 Oak St		Chicago		bobj@example.com	312-567-8901	ORD0000	1	6	21000
4	Allie	Brown	111 Elm St	Apt 2C	San Francisco		allieb@example.com	555-789-0123	ORD0000	1	6	21000
4	Allie	Brown	111 Elm St	Apt 2C	San Francisco		allieb@example.com	555-789-0123	ORD0000	2	6	21000
4	Allie	Brown	111 Elm St	Apt 2C	San Francisco		allieb@example.com	555-789-0123	ORD0000	1	6	21000
4	Allie	Brown	111 Elm St	Apt 2C	San Francisco		allieb@example.com	555-789-0123	ORD0000	2	6	21000
4	Allie	Brown	111 Elm St	Apt 2C	San Francisco		allieb@example.com	555-789-0123	ORD0000	1	6	21000
5	Eva	Williams	555 Pine St		Houston		evaw@example.com	281-234-5678	ORD0000	1	6	21000
5	Eva	Williams	555 Pine St		Houston		evaw@example.com	281-234-5678	ORD0000	2	6	21000
5	Eva	Williams	555 Pine St		Houston		evaw@example.com	281-234-5678	ORD0000	1	6	21000
5	Eva	Williams	555 Pine St		Houston		evaw@example.com	281-234-5678	ORD0000	2	6	21000
5	Eva	Williams	555 Pine St		Houston		evaw@example.com	281-234-5678	ORD0000	1	6	21000
6	David	Lee	888 Oak St	Apt 3D	Miami		davidl@example.com	305-975-5432	ORD0000	1	6	21000
6	David	Lee	888 Oak St	Apt 3D	Miami		davidl@example.com	305-975-5432	ORD0000	2	6	21000
6	David	Lee	888 Oak St	Apt 3D	Miami		davidl@example.com	305-975-5432	ORD0000	1	6	21000
6	David	Lee	888 Oak St	Apt 3D	Miami		davidl@example.com	305-975-5432	ORD0000	2	6	21000
6	David	Lee	888 Oak St	Apt 3D	Miami		davidl@example.com	305-975-5432	ORD0000	1	6	21000

26 rows (0.00 sec)

mysql>

80% MySQL Workbench

MySQL Workbench

Search

1:40 AM 8/13/2021

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

```
MySQL 8.0 Command Line CL
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 8.0 Command Line CL
mysql> SELECT brand,model,price from Laptop order by price asc;
+-----+-----+-----+
| brand | model      | price |
+-----+-----+-----+
| Dell  | Inspiron 5532 | 385000.00 |
| 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

```
MySQL 8.0 Command Line CL
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

```
MySQL 8.0 Command Line CL
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 |
+-----+
| Arisna |
| NY |
| San Francisco |
| London |
| Seattle |
| Boston |
| Colombo |
+-----+
7 rows in set (0.00 sec)

mysql>
```

FIGURE 40:UNION

```
MySQL 8.0 Command Line CL
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 | 485000.00 | 2023-02-15 | 2025-02-14 | 24 |
| 21003 | Dell | Inspiron 5532 | 380000.00 | 2023-03-10 | 2026-03-09 | 36 |
| 21004 | Huawei | Matebook X | 480000.00 | 2023-04-20 | 2024-04-19 | 12 |
| 21005 | Acer | Nitro 15 | 335000.00 | 2023-05-05 | 2025-05-04 | 24 |
| 21006 | MSI | Stealth 15 | 505000.00 | 2023-06-30 | 2026-06-29 | 36 |
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

FIGURE 41:INTERSECTION


```
MySQL 8.0 Command Line CL - + - -
| 21006 | RSI | 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 Marranty AS M 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 | 2023-04-10 | 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

```
MySQL 8.0 Command Line CL - + - -
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | TR0001 | Standard | 01A | 700 | Dakota Street | Boston | 01 | 1 | Bob | Johnson | 700 Oak St. | 011 | bob@example.com | 555-123-4567 |
| 2 | TR0001 | Express | 01B | 150 | Pine Street | San Francisco | 01 | 2 | Alice | Brown | 555 Elm St. | 010 | alice@example.com | 555-789-0123 |
| 3 | TR0001 | Standard | 01C | 200 | 1st Avenue | Portland | 01 | 3 | Eve | Williams | 800 Elm St. | 011 | eve@example.com | 555-234-5678 |
| 4 | TR0001 | Express | 01D | 100 | Grand St. | Seattle | 01 | 4 | David | Lee | 900 Oak St. | 015 | david@example.com | 555-456-7890 |
| 5 | TR0001 | Standard | 01E | 180 | Main St. | Portland | 01 | 5 | John | Doe | 123 Main Street | 016 | john@example.com | 555-123-4567 |
| 6 | TR0001 | Standard | 01F | 120 | Main St. | Portland | 01 | 6 | Jane | Smith | 456 Elm St. | 011 | jane@example.com | 555-123-4567 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> SELECT DISTINCT M.Laptop_ID FROM Marranty AS M WHERE NOT EXISTS ( SELECT L.Laptop_ID FROM Laptop AS L WHERE NOT EXISTS ( SELECT M.Warranty_ID FROM Marranty AS M WHERE M.Laptop_ID = L.Laptop_ID AND M.Warranty_ID = M.Laptop_ID );
Empty set (0.00 sec)

mysql> |
```

FIGURE 43:DIVISION

```

mysql> CREATE VIEW NaturalJoin_Shipping_Customer AS SELECT S1.Shipping_ID, S1.tracking_no, S1.Shipping_type, S1.Country, S1.House_no, S1.street_name, S1.city AS Shipping_City, S1.state, C.Customer_ID, C.First_name, C.Last_name, C.Address_Line1, C.Address_Line2, C.Email, C.Phone_no FROM Shipping_Details AS S1 NATURAL JOIN Customer AS C;
[SHOW 104 (0.041) Table 'NaturalJoin_Shipping_Customer' already exists]
mysql> CREATE VIEW NaturalJoin_Shipping_Customer AS SELECT S1.Shipping_ID, S1.tracking_no, S1.Shipping_type, S1.Country, S1.House_no, S1.street_name, S1.city AS Shipping_City, S1.state, C.Customer_ID, C.First_name, C.Last_name, C.Address_Line1, C.Address_Line2, C.Email, C.Phone_no FROM Shipping_Details AS S1 NATURAL JOIN Customer AS C;
[SHOW 104 (0.041) Table 'NaturalJoin_Shipping_Customer' already exists]
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> CREATE VIEW NL_Shipping_Customer AS SELECT S1.Shipping_ID, S1.tracking_no, S1.Shipping_type, S1.Country, S1.House_no, S1.street_name, S1.city AS Shipping_City, S1.state, C.Customer_ID, C.First_name, C.Last_name, C.Address_Line1, C.Address_Line2, C.Email, C.Phone_no FROM Shipping_Details AS S1 NATURAL JOIN Customer AS C;
[Query OK, 0 rows affected (0.46 sec)]
mysql> select * from NaturalJoin_Shipping_Customer;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Shipping_ID | tracking_no | Shipping_type | Country | House_no | street_name | Shipping_City | state | Customer_ID | First_name | Last_name | Address_Line1 | Address_Line2 | Email | Phone_no |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 10000 | Express | USA | 101 | Rio street | San Francisco | SF | 1 | Alice | Brown | 121 Elm St | Apt 2C | alie@brownie.com | 555-555-4121 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
[ row 10 out of 10 rows ]
mysql>

```

FIGURE 44:NATURAL JOIN

```

mysql> CREATE VIEW NL_Shipping_Customer AS SELECT S1.Shipping_ID, S1.tracking_no, S1.Shipping_type, S1.Country, S1.House_no, S1.street_name, S1.city AS Shipping_City, S1.state, C.Customer_ID, C.First_name, C.Last_name, C.Address_Line1, C.Address_Line2, C.Email, C.Phone_no FROM Shipping_Details AS S1 INNER JOIN Customer AS C ON S1.Shipping_ID = C.Customer_ID;
[Query OK, 0 rows affected (0.47 sec)]
mysql> select * from NL_Shipping_Customer;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Shipping_ID | tracking_no | Shipping_type | Country | House_no | street_name | Shipping_City | state | Customer_ID | First_name | Last_name | Address_Line1 | Address_Line2 | Email | Phone_no |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 10000 | Standard | USA | 101 | Subvill Street | Austin | TX | 1 | Bob | Johnson | 789 Oak St | Bldg | bob@sample.com | 555-555-9890 | |
| 4 | 10000 | Express | USA | 101 | Rio street | San Francisco | SF | 4 | Alice | Brown | 121 Elm St | Apt 2C | alie@brownie.com | 555-555-4121 |
| 5 | 10000 | Standard | USA | 242 | Latha | Petaluma | California | CA | 5 | Ann | Williams | 325 Pine St. | Bldg | ann@sample.com | 555-555-1878 |
| 6 | 10000 | Express | Brazil | 101 | Road 10 | Seattle | WA | 6 | David | Lee | 456 Oak St | Apt 10 | david@sample.com | 555-555-5522 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
[ row 10 out of 10 rows ]
mysql>

```

FIGURE 45:INNER JOIN

MySQL 8.0 Command Line Client

```
mysql> CREATE VIEW V1 AS SELECT S1.Shipping_ID, S2.Tracking_no, S2.Shipping_type, S2.Country, S2.House_no, S2.Street_name, S2.City AS Shipping_City, S2.State, C.Customer_ID, C.First_name, C.Last_name, C.Address_Line1, C.Address_Line2, C.Email, C.Phone_no FROM Shipping_Details S1 LEFT JOIN Customer AS C ON S1.Shipping_ID = C.Customer_ID;
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> select * from V1;
```

Shipping_ID	Tracking_no	Shipping_type	Country	House_no	Street_name	Shipping_City	State	Customer_ID	First_name	Last_name	Address_Line1	Address_Line2	Email	Phone_no
1	100001	Standard	USA	789	Oakfield Street	Boston	MA	1	Bob	Johnson	789 Oak St	N/A	bob@sample.com	185-547-0980
2	100002	Express	USA	123	Pine Street	San Francisco	CA	2	Alice	Smith	123 Elm St	Apt. 2C	alice@sample.com	155-290-4212
3	100003	Standard	Kenya	232	Petrich	Columbo	CE	3	Eva	Williams	345 Pine St	N/A	eva@sample.com	188-290-5678
4	100004	Express	Italy	555	Round st	Seattle	WA	4	David	Lee	888 Oak St	Apt. 3D	david@sample.com	192-875-0412

0 rows in set (0.00 sec)

```
mysql>
```

FIGURE 46:LEFT OUTER JOIN

MySQL 8.0 Command Line Client

```
mysql> select * from V1;
```

Shipping_ID	Tracking_no	Shipping_type	Country	House_no	Street_name	Shipping_City	State	Customer_ID	First_name	Last_name	Address_Line1	Address_Line2	Email	Phone_no
1	100001	Standard	USA	789	Oakfield Street	Boston	MA	1	Bob	Johnson	789 Oak St	N/A	bob@sample.com	185-547-0980
2	100002	Express	USA	123	Pine Street	San Francisco	CA	2	Alice	Smith	123 Elm St	Apt. 2C	alice@sample.com	155-290-4212
3	100003	Standard	Kenya	232	Petrich	Columbo	CE	3	Eva	Williams	345 Pine St	N/A	eva@sample.com	188-290-5678
4	100004	Express	Italy	555	Round st	Seattle	WA	4	David	Lee	888 Oak St	Apt. 3D	david@sample.com	192-875-0412

0 rows in set (0.00 sec)

```
mysql>
```

FIGURE 47 RIGHT OUTER JOIN

```
MySQL 8.0 Command Line CL
> use db; set (0.00 sec)

mysql> CREATE VIEW F1 AS select * from LOG.Shipping_Customers UNION select * from REG.Shipping_Customers;
Query OK, 0 rows affected (0.38 sec)

mysql> select * from F1;
```

Shipping_ID	Tracking_no	Shipping_type	Country	House_no	Street_name	Shipping_City	state	Customer_ID	First_name	Last_name	Address_Line1	Address_Line2	Email	Phone_no
1	TR0001	Standard	USA	789	Nashville Street	Boston	MA	1	Bob	Jackson	789 Oak St	N/A	bob@example.com	555-987-0000
2	TR0001	Express	USA	123	Pine Street	San Francisco	CA	2	Alice	Brown	321 Elm St	Apt 2C	alice@example.com	555-789-0123
3	TR0001	Standard	USA	234	Pertham	California	CA	3	Eve	Williams	555 Elm St	N/A	eve@example.com	555-234-5678
4	TR0001	Express	Italy	122	Rond al	Sancti	IN	4	David	Lee	234 Oak St	Apt 2C	david@example.com	555-456-7890
N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	5	John	Kim	123 Main Street	Apt 4B	john@example.com	555-123-4567
N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	6	Jane	Smith	456 Elm St	N/A	jane@example.com	555-987-0543

```
> use db; set (0.00 sec)

mysql>
```

FIGURE 48:FULL OUTR JOIN

```
MySQL 8.0 Command Line CL
> use db; set (0.00 sec)

mysql> CREATE VIEW F2 AS SELECT ID,Shipping_ID, ID,Tracking_no, ID,Shipping_type, ID,Country, ID,House_no, ID,Street_name, ID,City AS Shipping_City, ID,State, C,Customer_ID, C,First_name, C>Last_name, C,Address_Line1, C,Address_Line2, C,Email, C,Phone_no FROM Shipping_Details AS S LEFT JOIN REG_Customer AS C;
```

```
Query OK, 0 rows affected (0.00 sec)

mysql> select * from F2;
```

Shipping_ID	Tracking_no	Shipping_type	Country	House_no	Street_name	Shipping_City	state	Customer_ID	First_name	Last_name	Address_Line1	Address_Line2	Email	Phone_no
1	TR0001	Express	USA	123	Pine Street	San Francisco	CA	2	Alice	Brown	321 Elm St	Apt 2C	alice@example.com	555-789-0123

```
> use db; set (0.00 sec)

mysql>
```

FIGURE 49:OUTER UNION

6 CHAPTER 5: TUNING

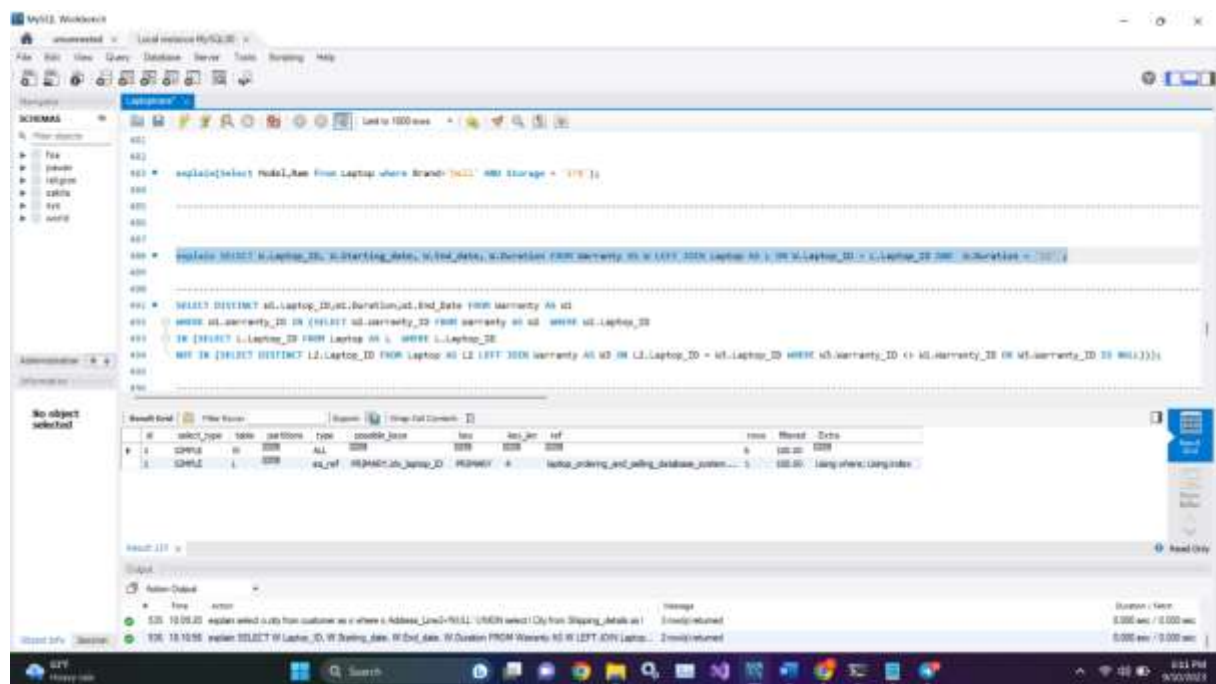


FIGURE 50:BEFORE TUNING DIFFERENCE OPERATION

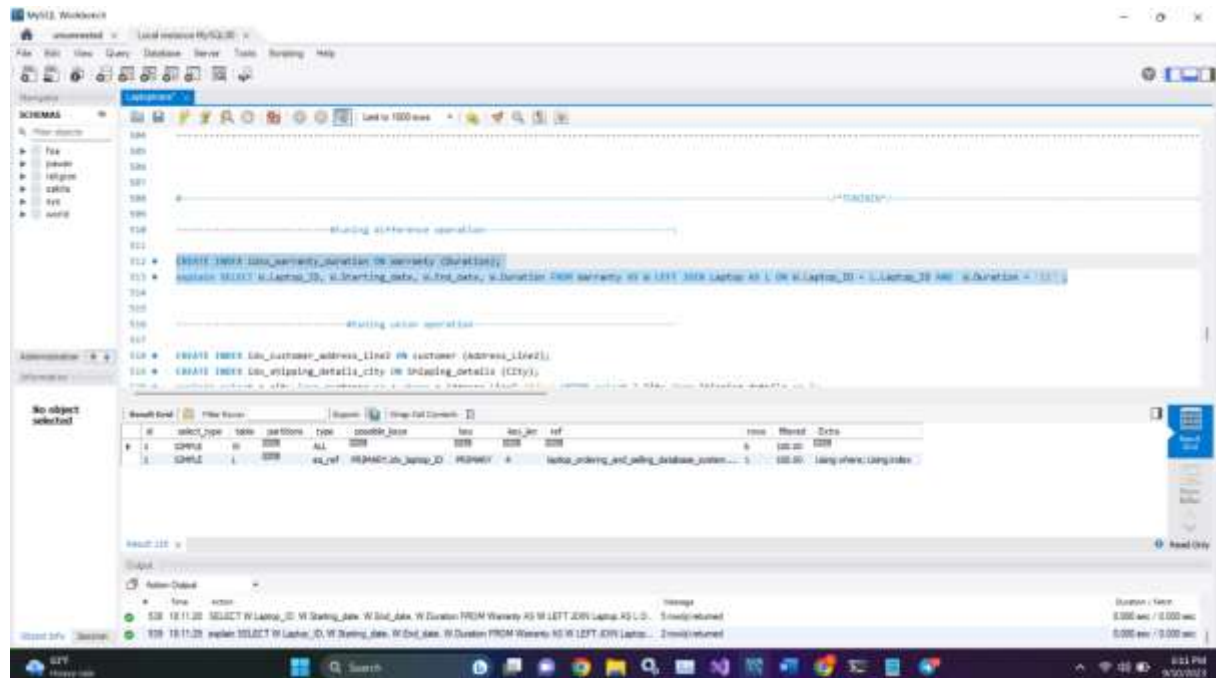


FIGURE 51:AFTER TUNING DIFFERENCE OPERATION

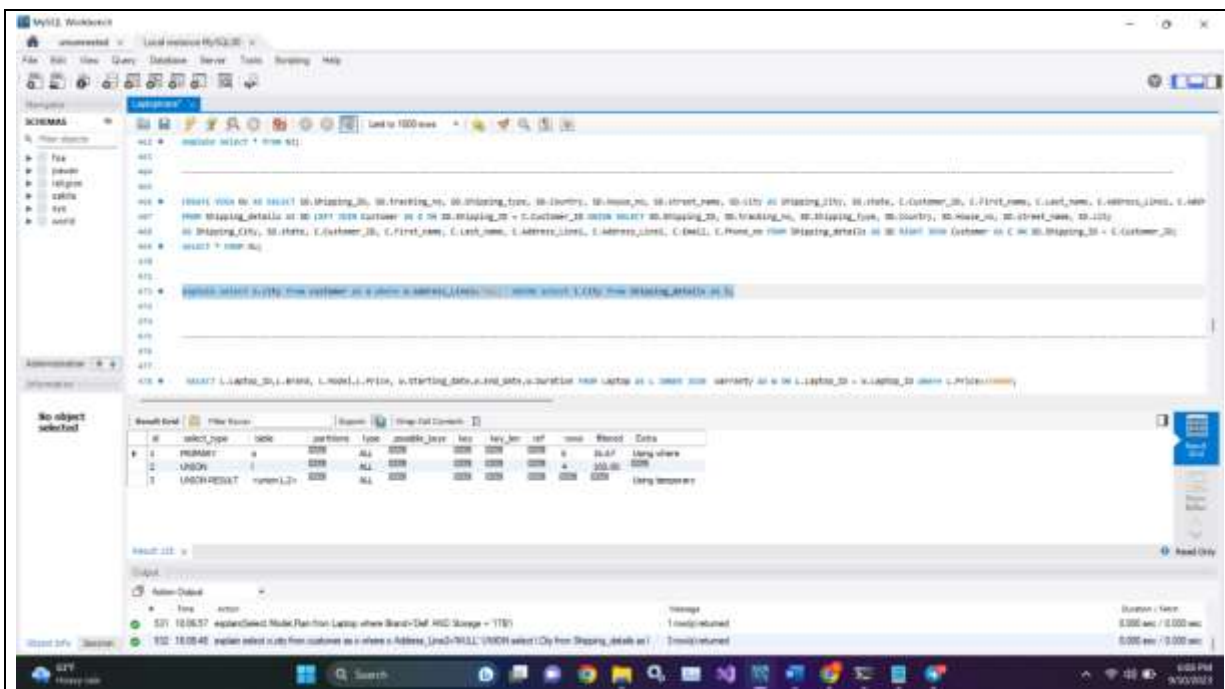


FIGURE 52: BEFORE TUNING UNION OPERATION

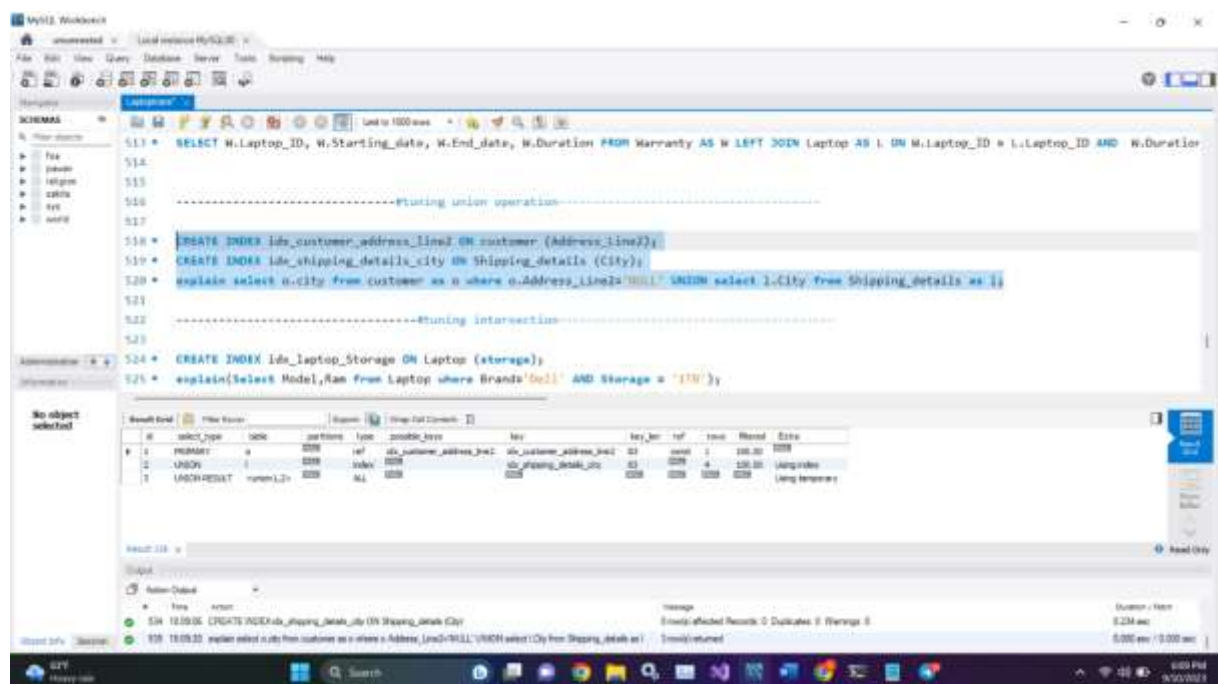


FIGURE 53: AFTER TUNING UNION OPERATION

- 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 tuned properly by using an index.

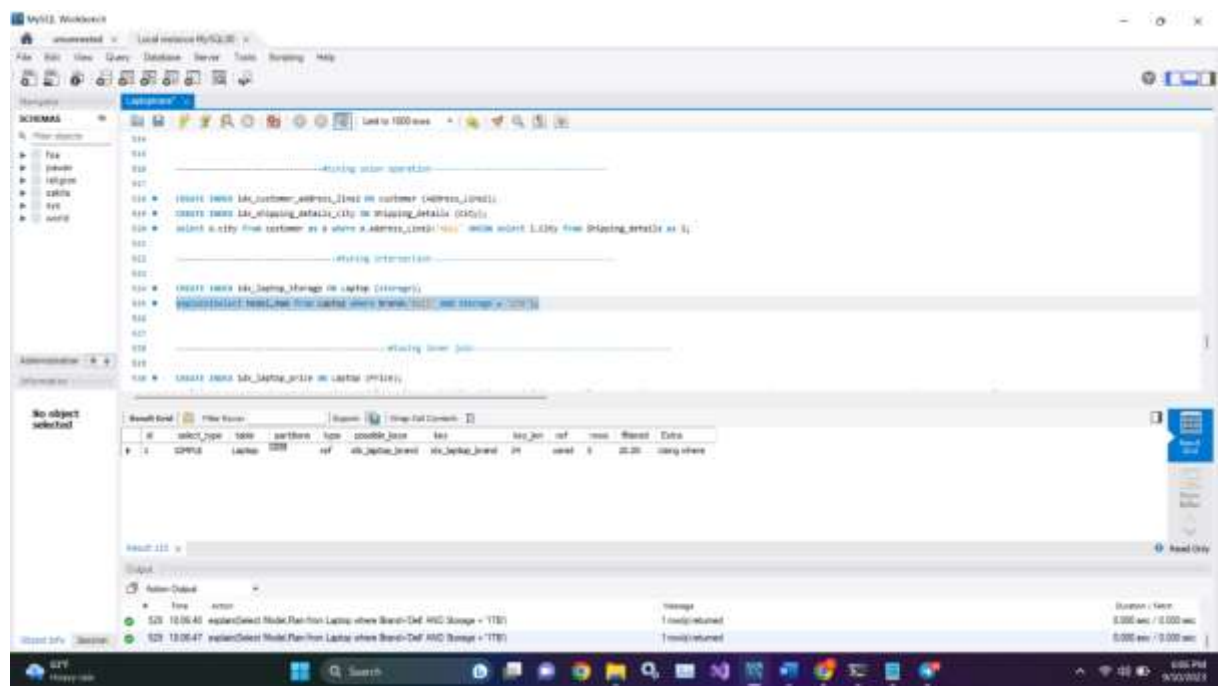


FIGURE 54 BEFORE TUNING INTERSECTION OPERATION

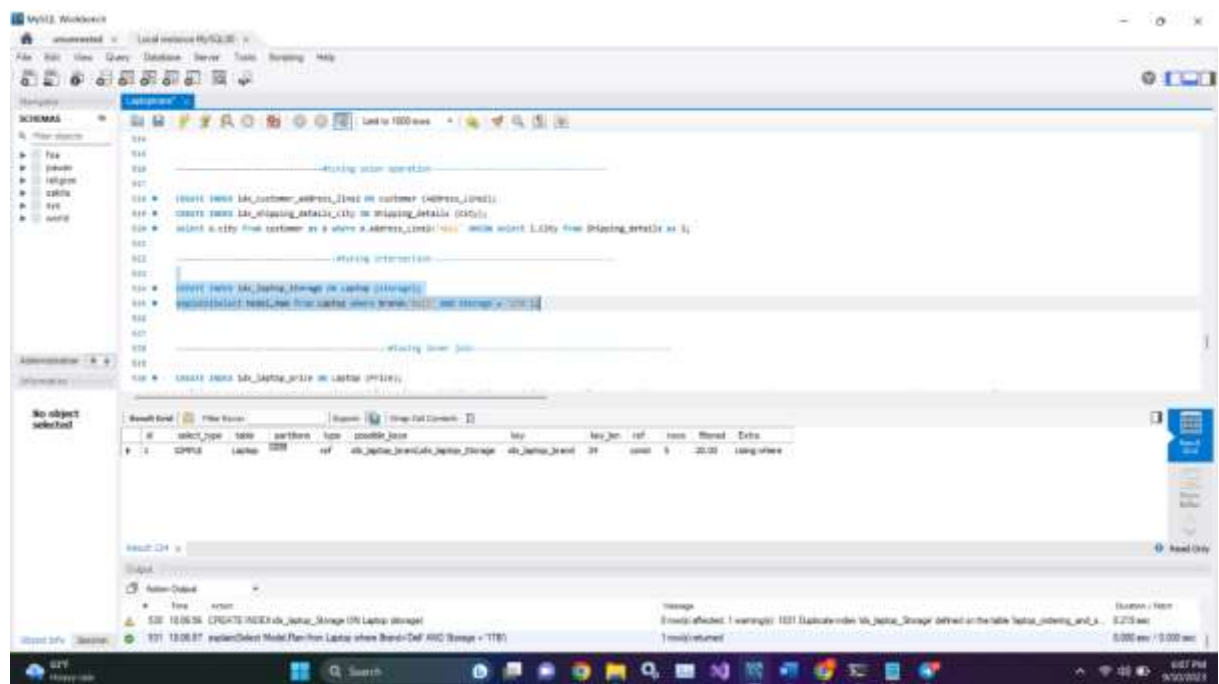


FIGURE 55 AFTER TUNING INTERSECTION OPERATION

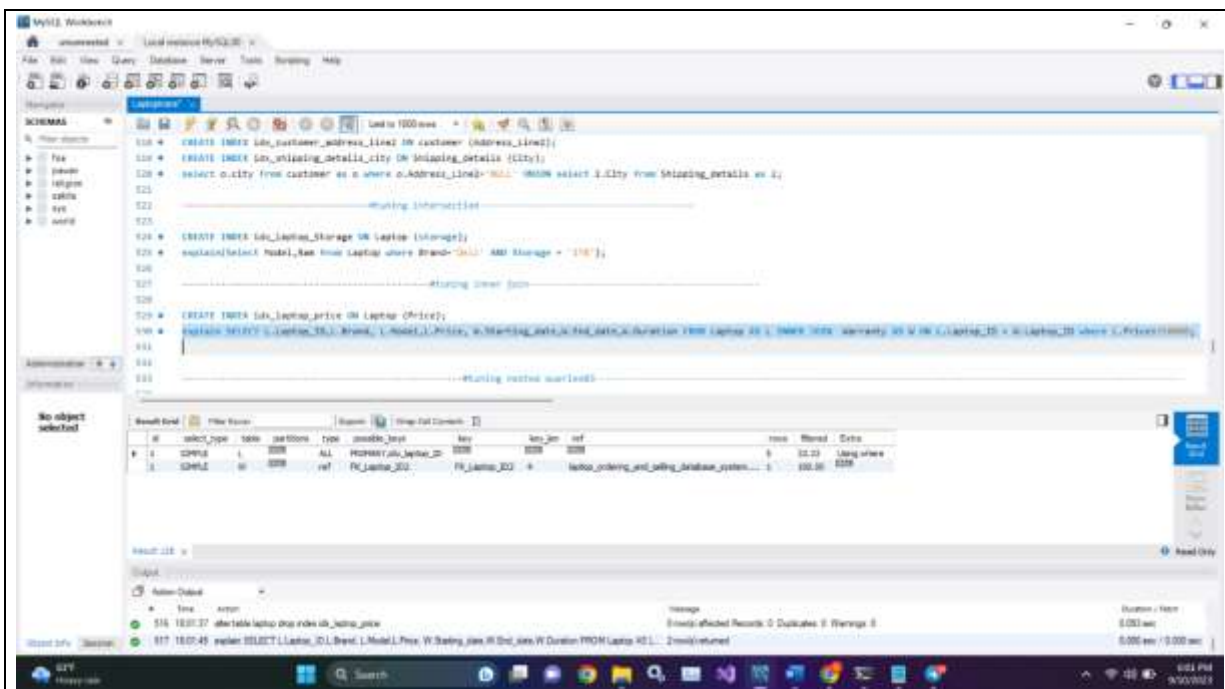


FIGURE 56 BEFORE TUNING INNER JOIN OPERATION

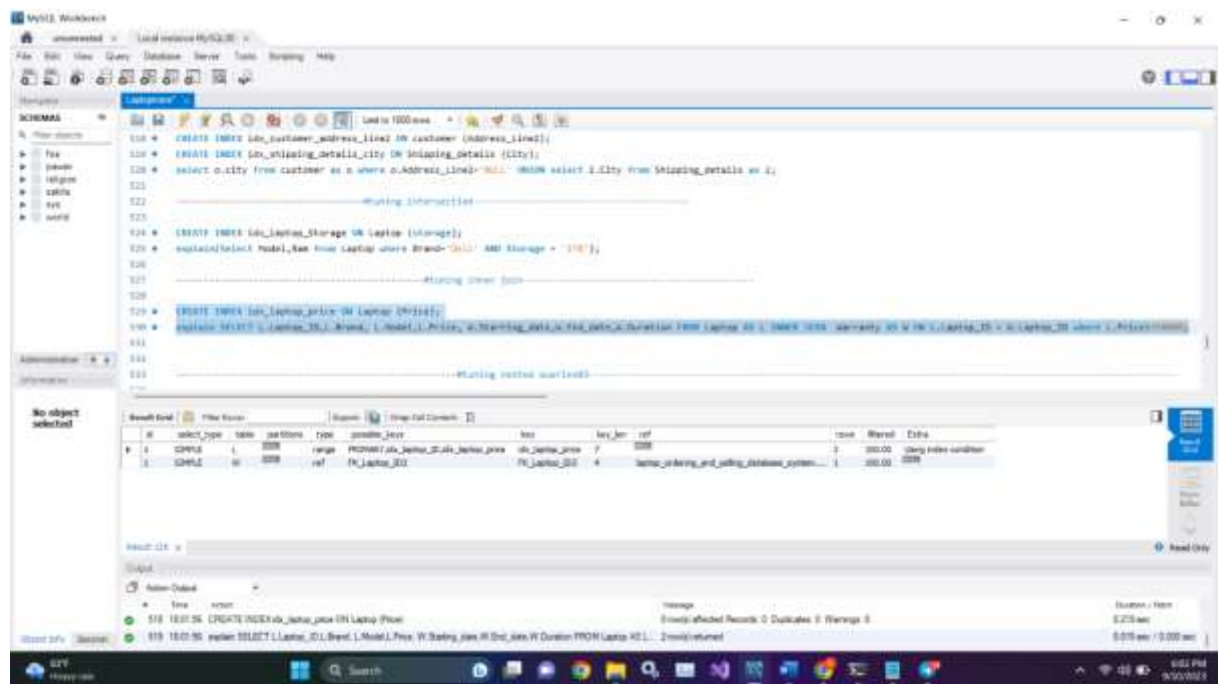


FIGURE 57 AFTER TUNING INNER JOIN OPERATION

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

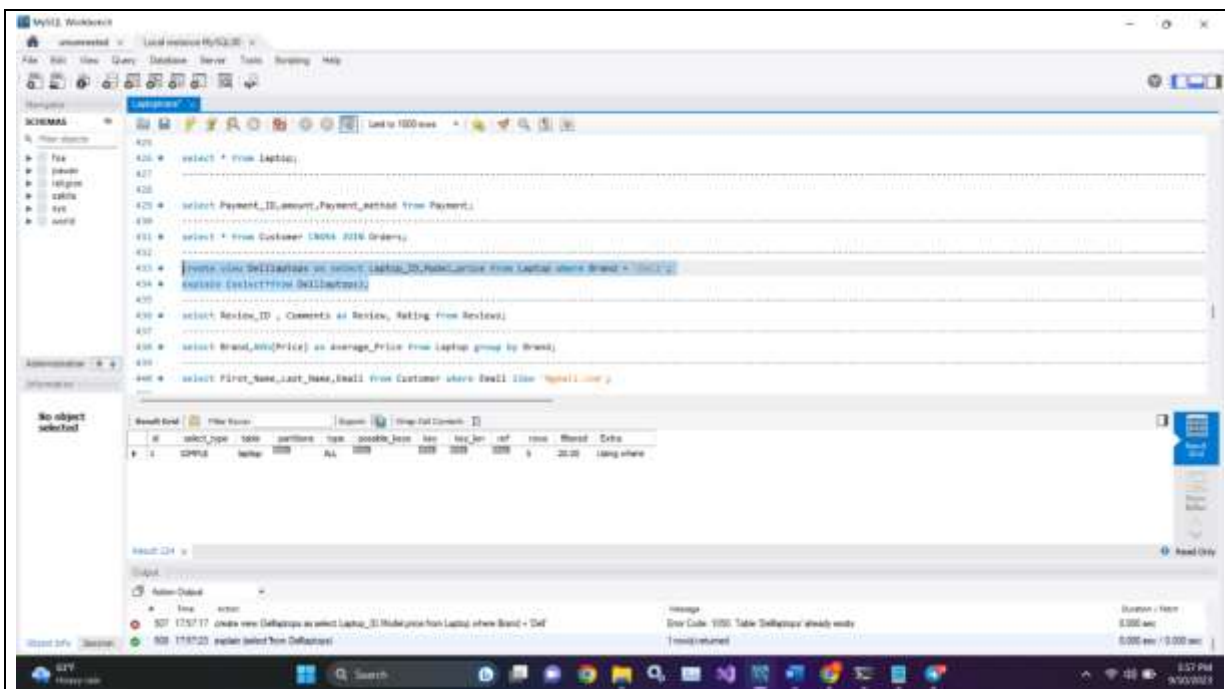


FIGURE 58 BEFORE TUNING SELECT

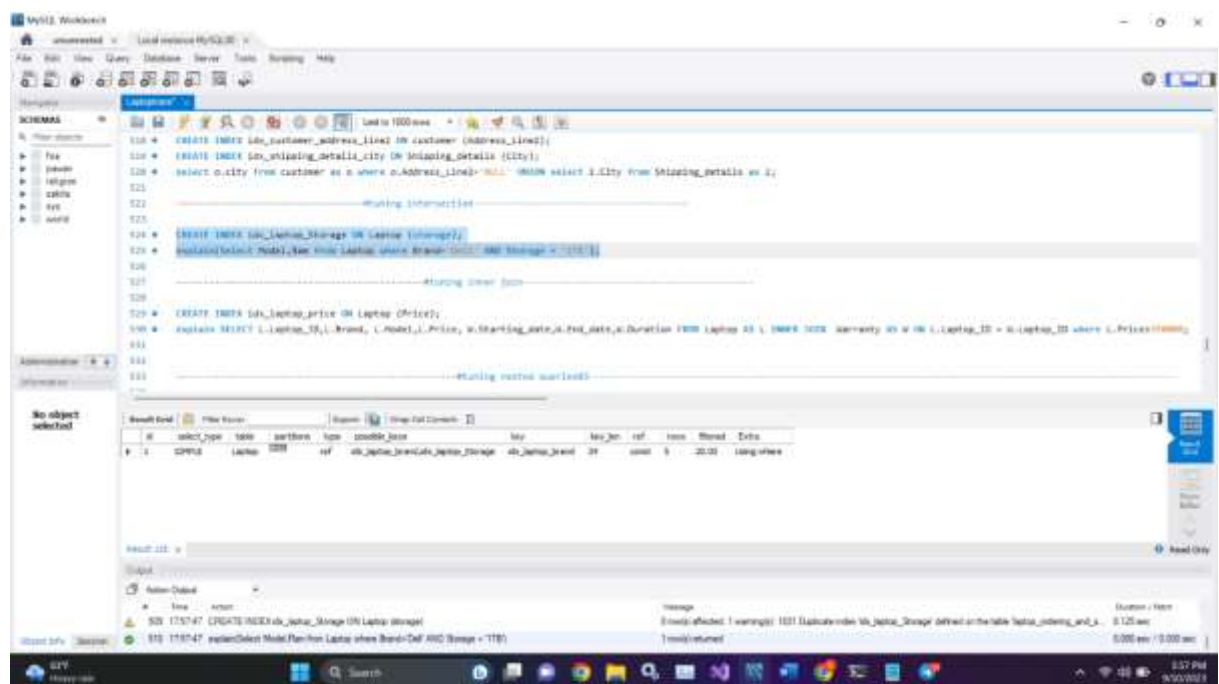


FIGURE 59 AFTER TUNING SELECT

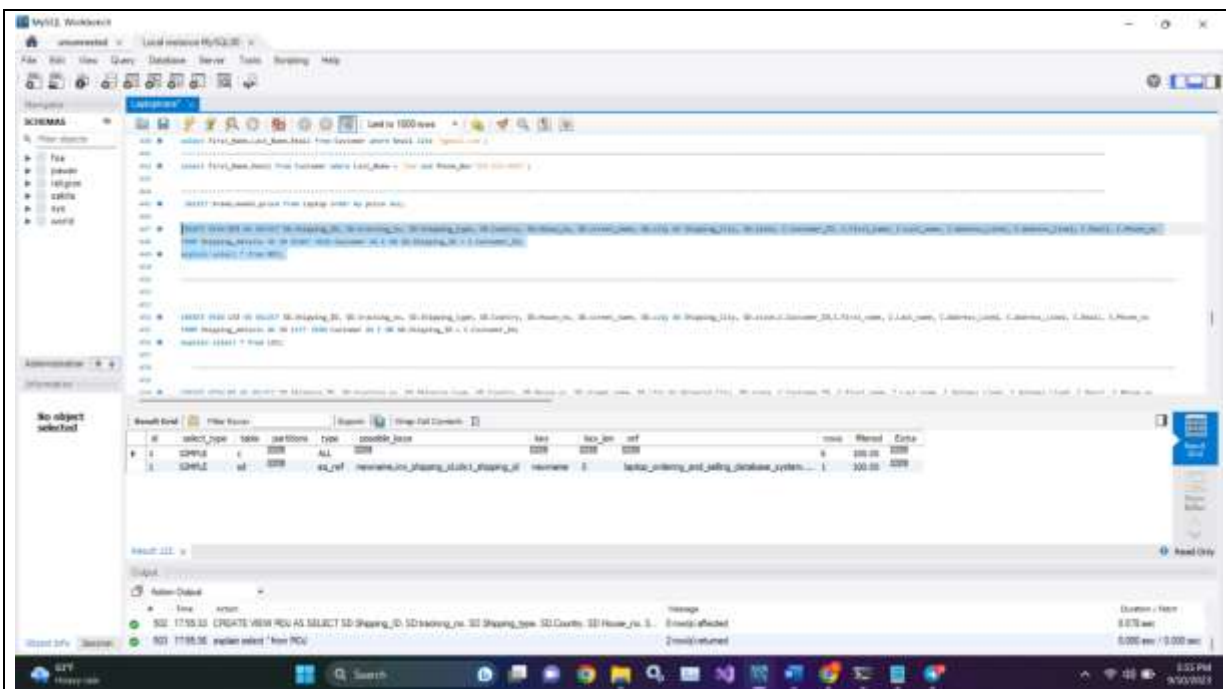


FIGURE 60 BEFORE TUNING RIGHT OUTER JOIN

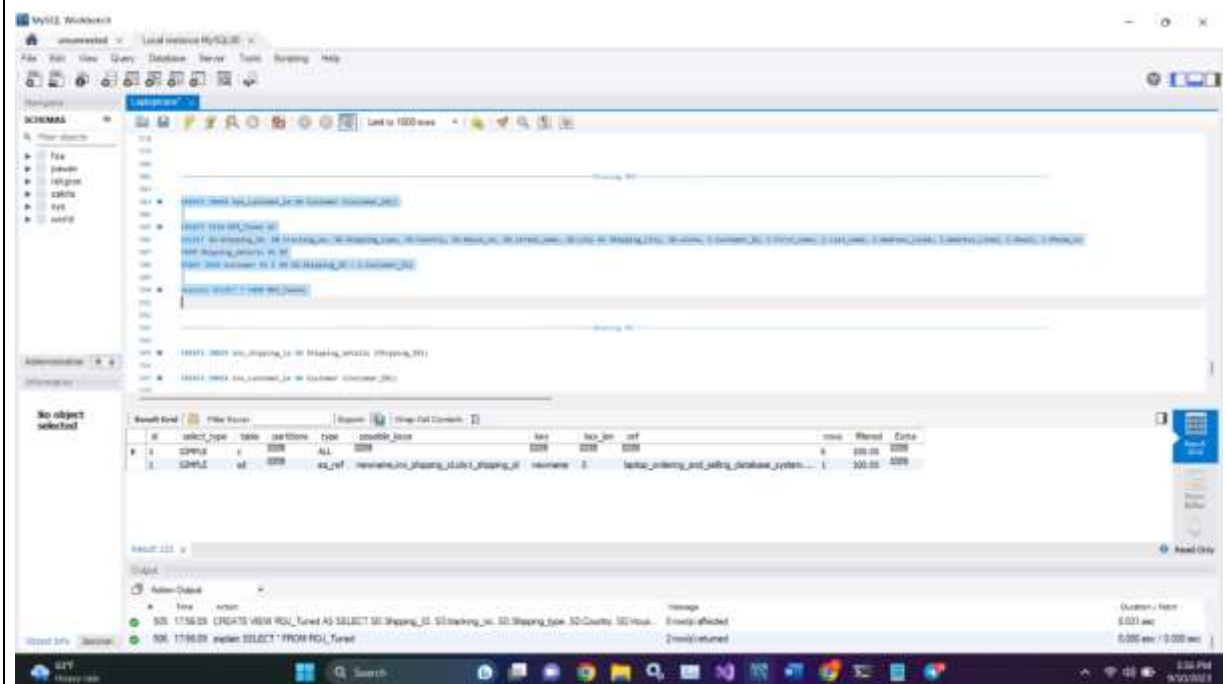


FIGURE 61 AFTER TUNING RIGHT OUTER JOIN

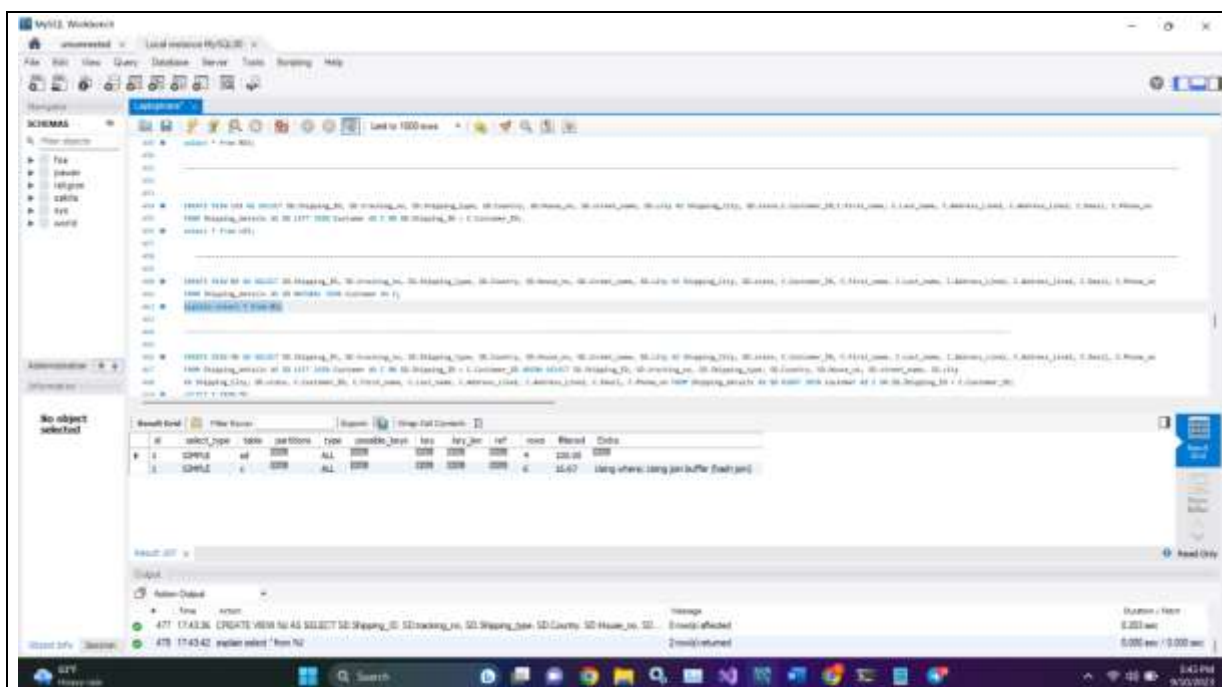


FIGURE 62 BEFORE TUNING NATURAL JOIN

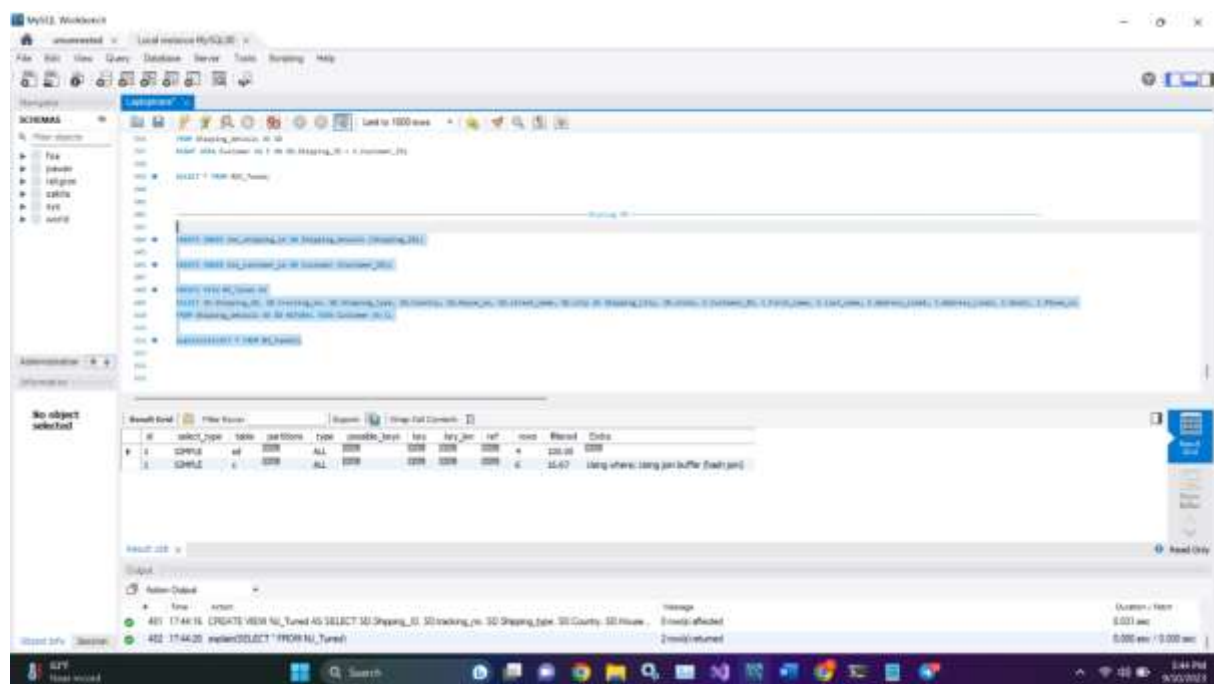


FIGURE 63 AFTER TUNING NATURAL JOIN

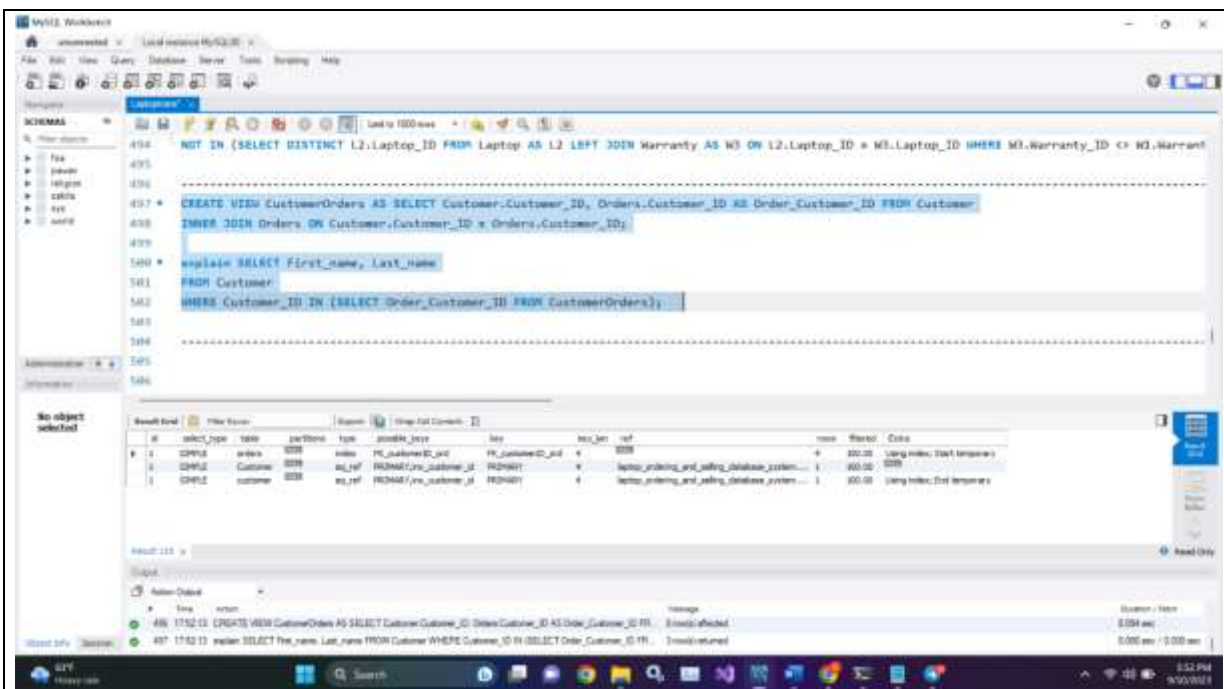


FIGURE 64 BEFORE TUNING NESTED QUERY01

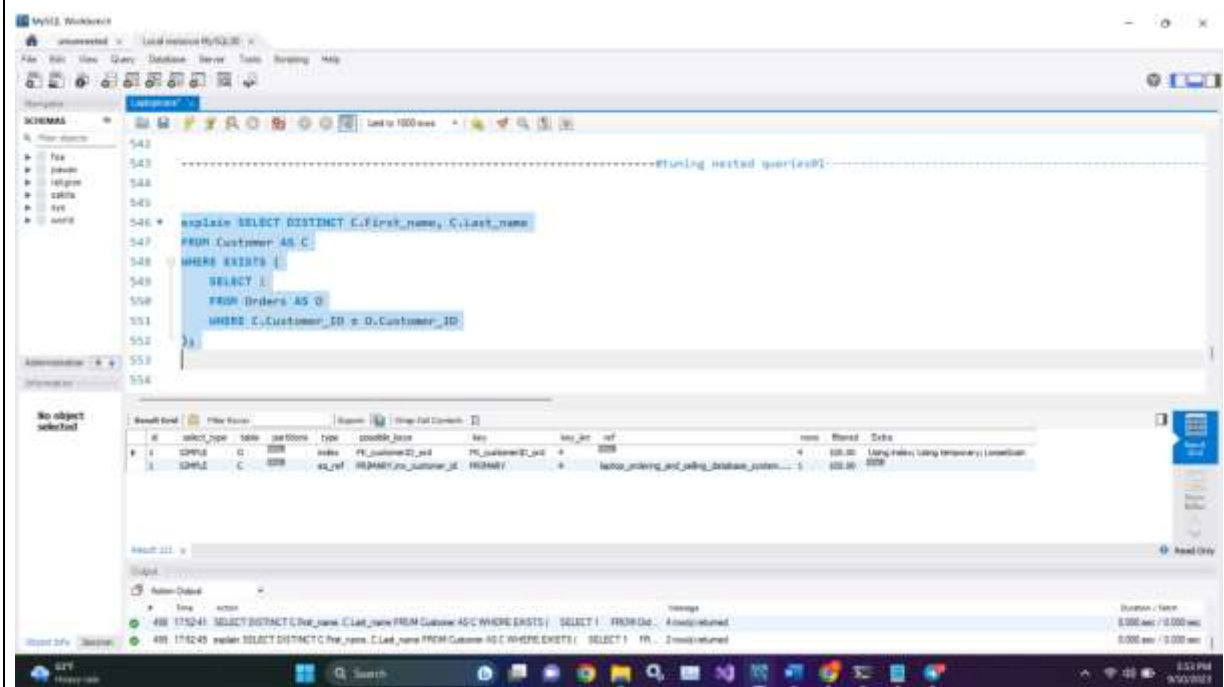


FIGURE 65 AFTER TUNING NESTED QUERY01

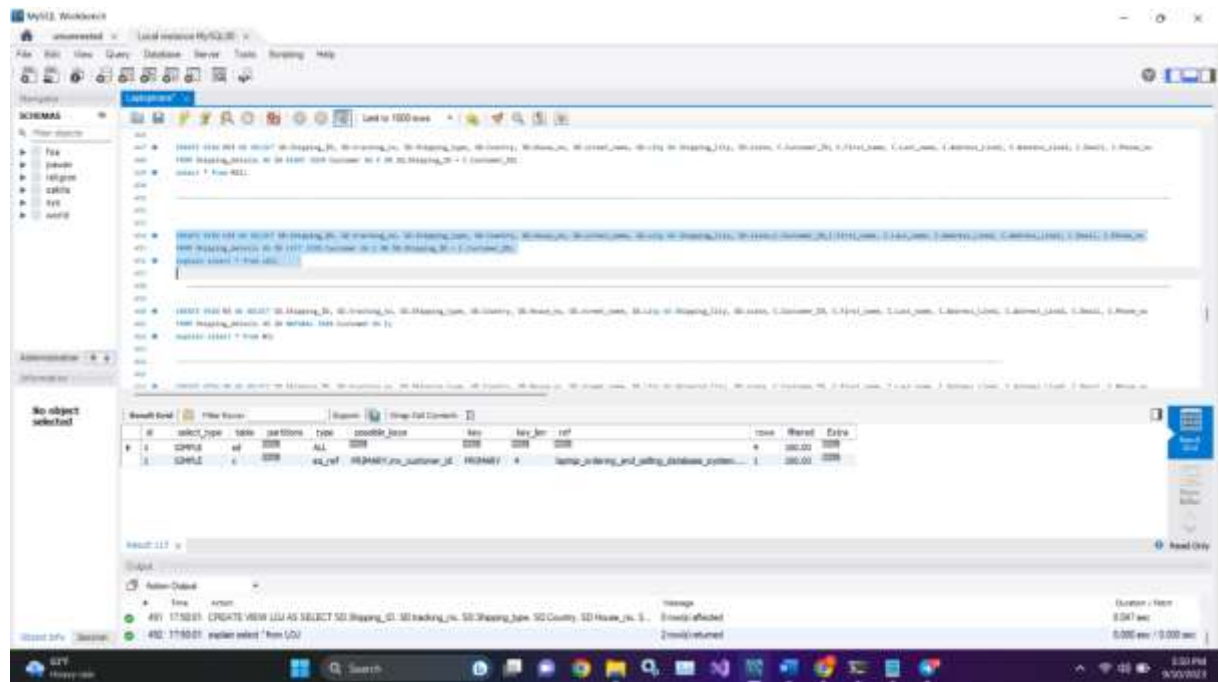


FIGURE 66 BEFORE TUNING NESTED QUERY02

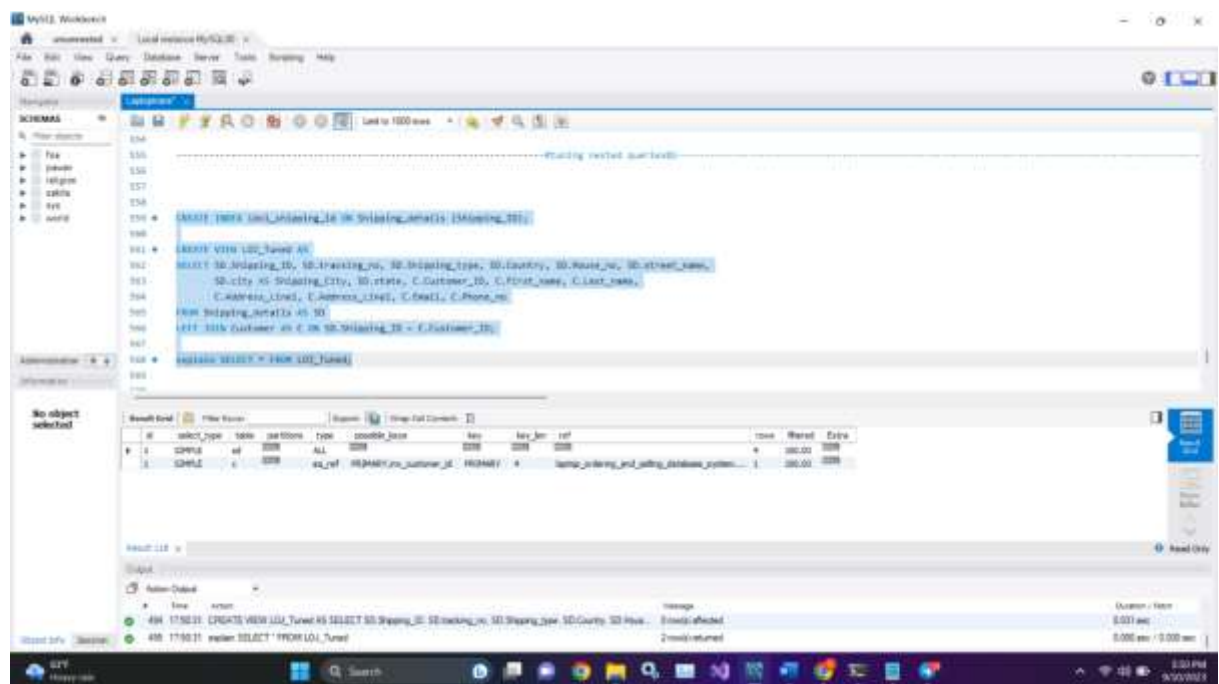


FIGURE 67 AFTER TUNING NESTED QUERY02

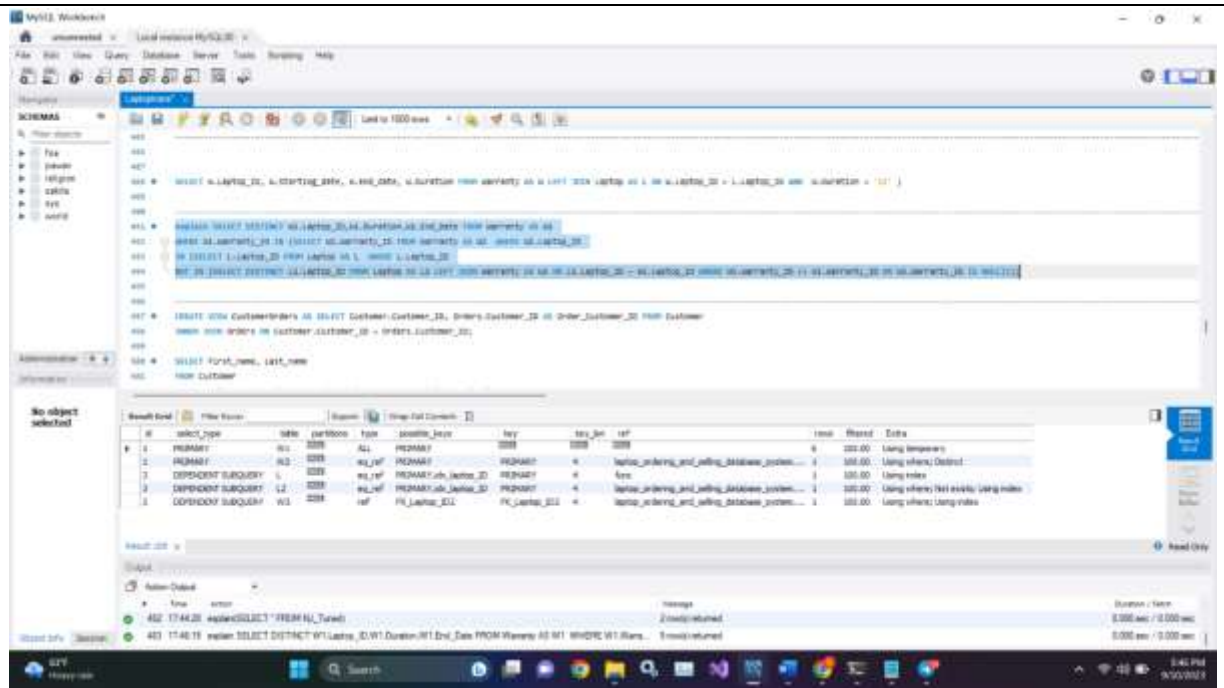


FIGURE 68 BEFORE TUNING NESTED QUERY03

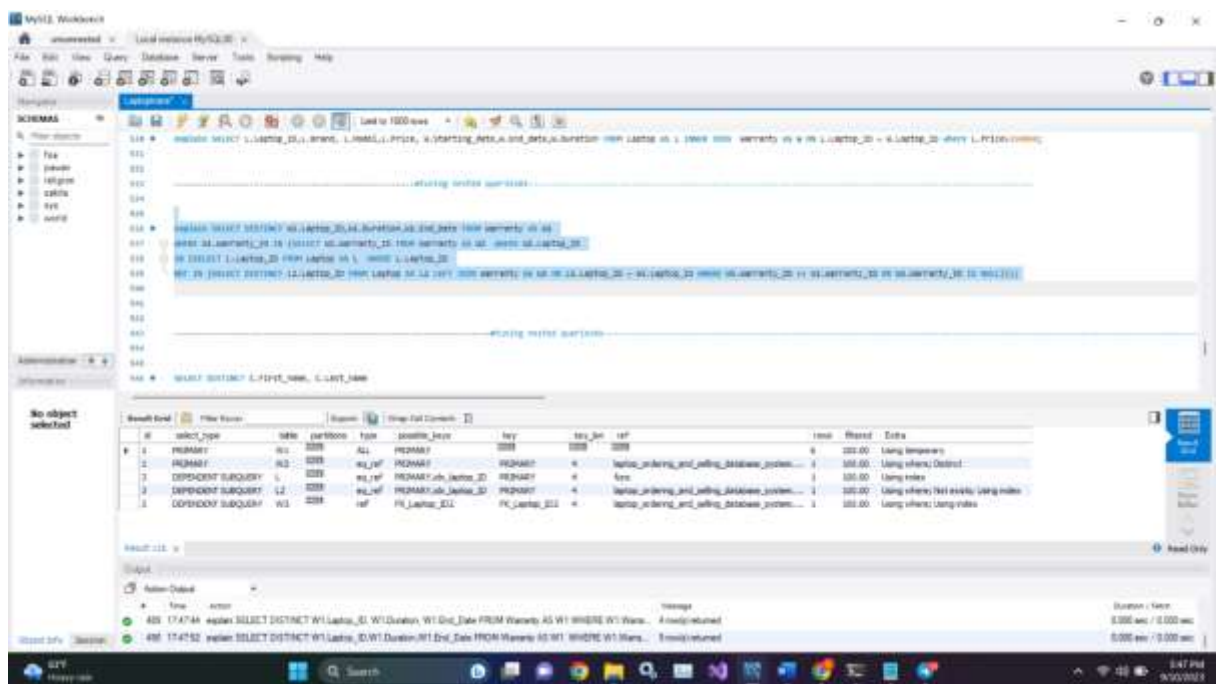


FIGURE 69 AFTER TUNING NESTED QUERY03

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