

NAME:JANHAVI PAWAR

PRN:2124UCSF1074

Dept:Cyber Security

DBMS PRACTICAL

Practical:1

Aim: Create a database and a table to store employee details. Perform basic operations like INSERT, UPDATE, and DELETE using SELECT queries.

Code:

```
-- Create the database if it doesn't exist
```

```
CREATE DATABASE IF NOT EXISTS EmployeeDB;
```

```
-- Use the created database
```

```
USE EmployeeDB;
```

```
-- Create the Employee table
```

```
CREATE TABLE IF NOT EXISTS Employee (
    EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Position VARCHAR(100),
    Salary DECIMAL(10, 2) CHECK (Salary > 0),
    JoiningDate DATE NOT NULL,
    ActiveStatus BOOLEAN DEFAULT TRUE
);
```

```
-- Insert employee data into the Employee table
```

```
INSERT INTO Employee (Name, Position, Salary, JoiningDate, ActiveStatus)
```

```
VALUES
```

```
('Janhavi Pawar', 'Manager', 75000.00, '2023-01-01', TRUE),  
('Mansi Chalak', 'Developer', 60000.00, '2022-05-10', TRUE),  
('Gauri Dani', 'Tester', 50000.00, '2021-08-15', TRUE),  
('Sanjivani Mogare', 'Designer', 45000.00, '2020-03-20', FALSE);
```

```
-- Select all data from the Employee table
```

```
SELECT * FROM Employee;
```

The screenshot shows the MySQL Workbench interface. The 'Query 1' tab contains the following SQL code:

```
20 ('Janhavi Pawar', 'Manager', 75000.00, '2023-01-01', TRUE),  
21 ('Mansi Chalak', 'Developer', 60000.00, '2022-05-10', TRUE),  
22 ('Gauri Dani', 'Tester', 50000.00, '2021-08-15', TRUE),  
23 ('Sanjivani Mogare', 'Designer', 45000.00, '2020-03-20', FALSE);  
24  
25 -- Select all data from the Employee table  
26 • SELECT * FROM Employee;  
--
```

The 'Result Grid' shows the data from the Employee table:

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Janhavi Pawar	55000.00	2023-06-15	1
2	Mansi Chalak	87120.61	2023-09-25	1
4	Tamanna Shalih	63000.00	2020-07-05	1
5	NULL	NULL	NULL	NULL

The 'Output' tab shows the history of database actions:

Action	Time	Message	Duration / Fetch
CREATE DATABASE IF NOT EXISTS EmployeeDB	2023-03-04 20:03:04	1 row(s) affected, 1 warning(s): 1007 Can't create database 'employees'; database exists	0.000 sec
USE EmployeeDB	2023-03-08 20:03:08	0 row(s) affected	0.000 sec
CREATE TABLE IF NOT EXISTS Employee (EmployeeID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(50), Position VARCHAR(50), Salary DECIMAL(10,2), JoiningDate DATE, ActiveStatus BOOLEAN)	2023-03-16 20:03:16	3 row(s) affected, 1 warning(s): 1050 Table 'employee' already exists	0.000 sec
INSERT INTO Employee (Name, Position, Salary, JoiningDate, ActiveStatus) VALUES ('Janhavi Pawar', 'Manager', 75000.00, '2023-01-01', TRUE)	2023-03-24 20:03:24	Error Code: 1054. Unknown column 'Position' in field list	0.000 sec
SELECT * FROM Employee LIMIT 0, 1000	2023-03-37 20:03:37	3 row(s) returned	0.000 sec / 0.000 sec

```
-- Select only active employees
```

```
SELECT EmployeeID, Name, Position, Salary FROM Employee WHERE ActiveStatus = TRUE;
```

```
-- Update salary for the employee with EmployeeID 2 (Mansi)
```

```
UPDATE Employee
```

```
SET Salary = Salary * 1.10
```

```
WHERE EmployeeID = 2;
```

```
-- Update the ActiveStatus for the employee with EmployeeID 4 (Sanjivani)
```

```
UPDATE Employee
```

```
SET ActiveStatus = TRUE
```

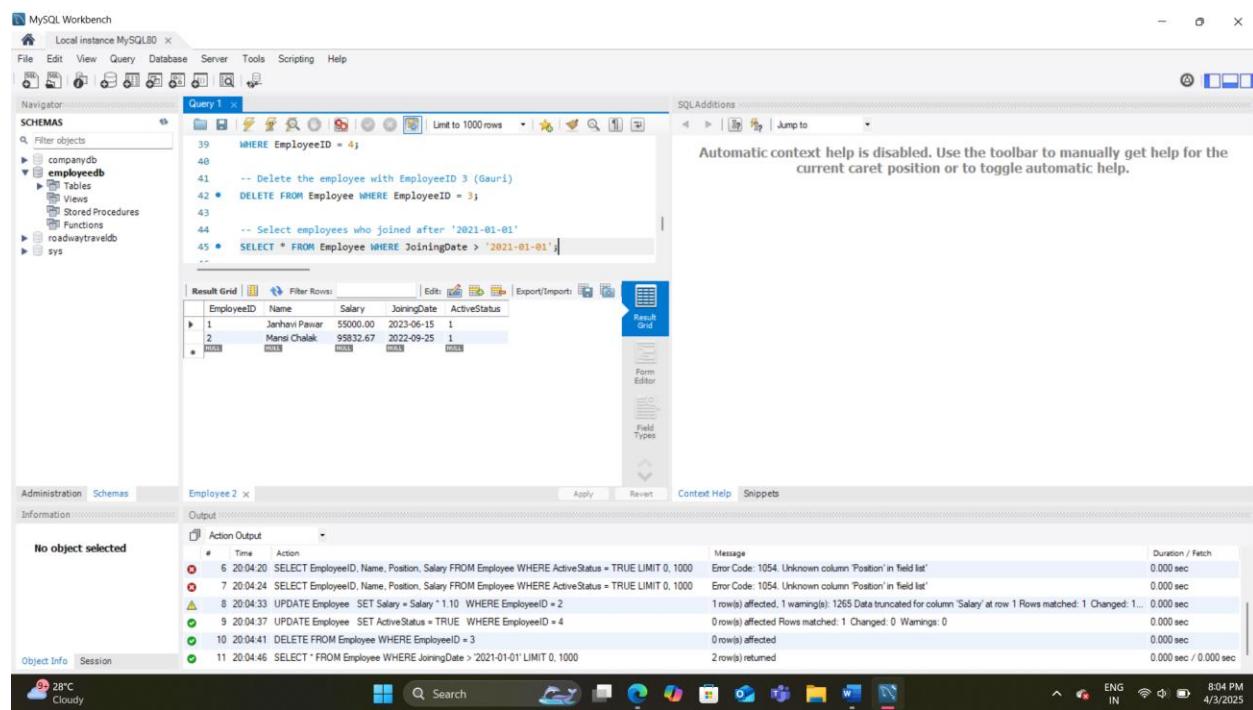
```
WHERE EmployeeID = 4;
```

```
-- Delete the employee with EmployeeID 3 (Gauri)
```

```
DELETE FROM Employee WHERE EmployeeID = 3;
```

```
-- Select employees who joined after '2021-01-01'
```

```
SELECT * FROM Employee WHERE JoiningDate > '2021-01-01';
```



The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query editor contains the following script:

```
39 WHERE EmployeeID = 4;
40
41 -- Delete the employee with EmployeeID 3 (Gauri)
42 DELETE FROM Employee WHERE EmployeeID = 3;
43
44 -- Select employees who joined after '2021-01-01'
45 SELECT * FROM Employee WHERE JoiningDate > '2021-01-01';
46
```

The results grid displays the following data:

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Janhavi Pawar	55000.00	2023-06-15	1
2	Mansi Chalak	95832.67	2022-09-25	1
3				

The status bar at the bottom shows the following information: 28°C Cloudy, Search, File Explorer, Task View, Taskbar, 8:04 PM, ENG IN, 4/3/2025.

```
-- Select names and salaries for employees with a salary greater than 60000
```

```
SELECT Name, Salary FROM Employee WHERE Salary > 60000;
```

```

42 • DELETE FROM Employee WHERE EmployeeID = 3
43
44 -- Select employees who joined after '2021-01-01'
45 • SELECT * FROM Employee WHERE JoiningDate > '2021-01-01';
46
47 -- Select names and salaries for employees with a salary greater than 60000
48 • SELECT Name, Salary FROM Employee WHERE Salary > 60000;
49

```

Name	Salary
Mansi Chalak	95832.67
Tananna Shahn	63000.00

Action Output

Time	Action	Message	Duration / Fetch
7 20:04:24	SELECT EmployeeID, Name, Position, Salary FROM Employee WHERE ActiveStatus = TRUE LIMIT 0, 1000	Error Code: 1054. Unknown column 'Position' in field list'	0.000 sec
8 20:04:33	UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID = 2	1row(s) affected. 1 warning(s): 1265 Data truncated for column 'Salary' at row 1 Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
9 20:04:37	UPDATE Employee SET ActiveStatus = TRUE WHERE EmployeeID = 4	0row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec
10 20:04:41	DELETE FROM Employee WHERE EmployeeID = 3	0row(s) affected	0.000 sec
11 20:04:46	SELECT * FROM Employee WHERE JoiningDate > '2021-01-01' LIMIT 0, 1000	2row(s) returned	0.000 sec / 0.000 sec
12 20:05:16	SELECT Name, Salary FROM Employee WHERE Salary > 60000 LIMIT 0, 1000	2row(s) returned	0.000 sec / 0.000 sec

-- Get the highest and lowest salary in the Employee table

SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS LowestSalary FROM Employee;

```

45 • * FROM Employee WHERE JoiningDate > '2021-01-01';
46
47 ct names and salaries for employees with a salary greater than 60000
48 • Name, Salary FROM Employee WHERE Salary > 60000;
49
50 the highest and lowest salary in the Employee table
51 • MAX(Salary) AS HighestSalary, MIN(Salary) AS LowestSalary FROM Employee;
52

```

HighestSalary	LowestSalary
95832.67	55000.00

Action Output

Time	Action	Message	Duration / Fetch
8 20:04:33	UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID = 2	1row(s) affected. 1 warning(s): 1265 Data truncated for column 'Salary' at row 1 Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
9 20:04:37	UPDATE Employee SET ActiveStatus = TRUE WHERE EmployeeID = 4	0row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec
10 20:04:41	DELETE FROM Employee WHERE EmployeeID = 3	0row(s) affected	0.000 sec
11 20:04:46	SELECT * FROM Employee WHERE JoiningDate > '2021-01-01' LIMIT 0, 1000	2row(s) returned	0.000 sec / 0.000 sec
12 20:05:16	SELECT Name, Salary FROM Employee WHERE Salary > 60000 LIMIT 0, 1000	2row(s) returned	0.000 sec / 0.000 sec
13 20:05:46	SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS LowestSalary FROM Employee LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec

-- Select the top 3 employees by salary, ordered in descending order

```
SELECT * FROM Employee ORDER BY Salary DESC LIMIT 3;
```

The screenshot shows the MySQL Workbench interface. The 'Query 1' tab contains the following SQL code and its execution results:

```
48 *   Name, Salary FROM Employee WHERE Salary > 60000;
49
50   the highest and lowest salary in the Employee table
51 *   MAX(Salary) AS HighestSalary, MIN(Salary) AS LowestSalary FROM Employee;
52
53   ct the top 3 employees by salary, ordered in descending order
54 *   * FROM Employee ORDER BY Salary DESC LIMIT 3;
```

The 'Result Grid' shows the following data:

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
2	Mansi Chalak	95832.67	2022-09-25	1
4	Tamanna Shalh	63000.00	2020-07-05	1
1	Janhavi Pawar	55000.00	2023-06-15	1
	NULL	NULL	NULL	NULL

The 'Output' tab displays the execution log:

#	Time	Action	Message	Duration / Fetch
3	20:04:37	UPDATE Employee SET ActiveStatus = TRUE WHERE EmployeeID = 4	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec
10	20:04:41	DELETE FROM Employee WHERE EmployeeID = 3	0 row(s) affected	0.000 sec
11	20:04:46	SELECT * FROM Employee WHERE JoiningDate > 2021-01-01 LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
12	20:05:16	SELECT Name, Salary FROM Employee WHERE Salary > 60000 LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
13	20:05:46	SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS LowestSalary FROM Employee LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
14	20:06:12	SELECT * FROM Employee ORDER BY Salary DESC LIMIT 3	3 row(s) returned	0.000 sec / 0.000 sec

Practical 2:

Aim: Design an ER diagram for a Roadway Travel Management System with entities like Customer, Travel Route, and Booking. Create tables and perform operations such as bookings and route assignments.

Code:

```
-- Create Database
```

```
CREATE DATABASE IF NOT EXISTS RoadwayTravelDB;
```

```
USE RoadwayTravelDB;
```

```
-- Create Customer Table
```

```
CREATE TABLE Customer (
```

```
    CustomerID INT PRIMARY KEY AUTO_INCREMENT,
```

```
    Name VARCHAR(100) NOT NULL,
```

```
Email VARCHAR(100),  
PhoneNumber VARCHAR(15),  
Address VARCHAR(255)  
);  
-- Create Travel Route Table  
CREATE TABLE TravelRoute (  
    RouteID INT PRIMARY KEY AUTO_INCREMENT,  
    StartLocation VARCHAR(100),  
    EndLocation VARCHAR(100),  
    Distance DECIMAL(5, 2),  
    Price DECIMAL(10, 2)  
);  
-- Create Booking Table  
CREATE TABLE Booking (  
    BookingID INT PRIMARY KEY AUTO_INCREMENT,  
    CustomerID INT,  
    RouteID INT,  
    BookingDate DATE NOT NULL,  
    SeatNumber INT,  
    TotalPrice DECIMAL(10, 2),  
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),  
    FOREIGN KEY (RouteID) REFERENCES TravelRoute(RouteID)  
);  
-- Insert Sample Data into Customer Table  
INSERT INTO Customer (Name, Email, PhoneNumber, Address) VALUES  
('Janhavi Pawar', 'janhavipawar@example.com', '1234567890', '123 Main St, City A'),
```

```
('Mansi Chalak', 'mansichalak@example.com', '9876543210', '456 Elm St, City B');
```

-- Insert Sample Data into TravelRoute Table

```
INSERT INTO TravelRoute (StartLocation, EndLocation, Distance, Price) VALUES
```

```
('City A', 'City B', 150.00, 50.00),
```

```
('City B', 'City C', 200.00, 70.00),
```

```
('City C', 'City D', 300.00, 90.00);
```

-- Insert Booking for a Customer

```
INSERT INTO Booking (CustomerID, RouteID, BookingDate, SeatNumber, TotalPrice)
```

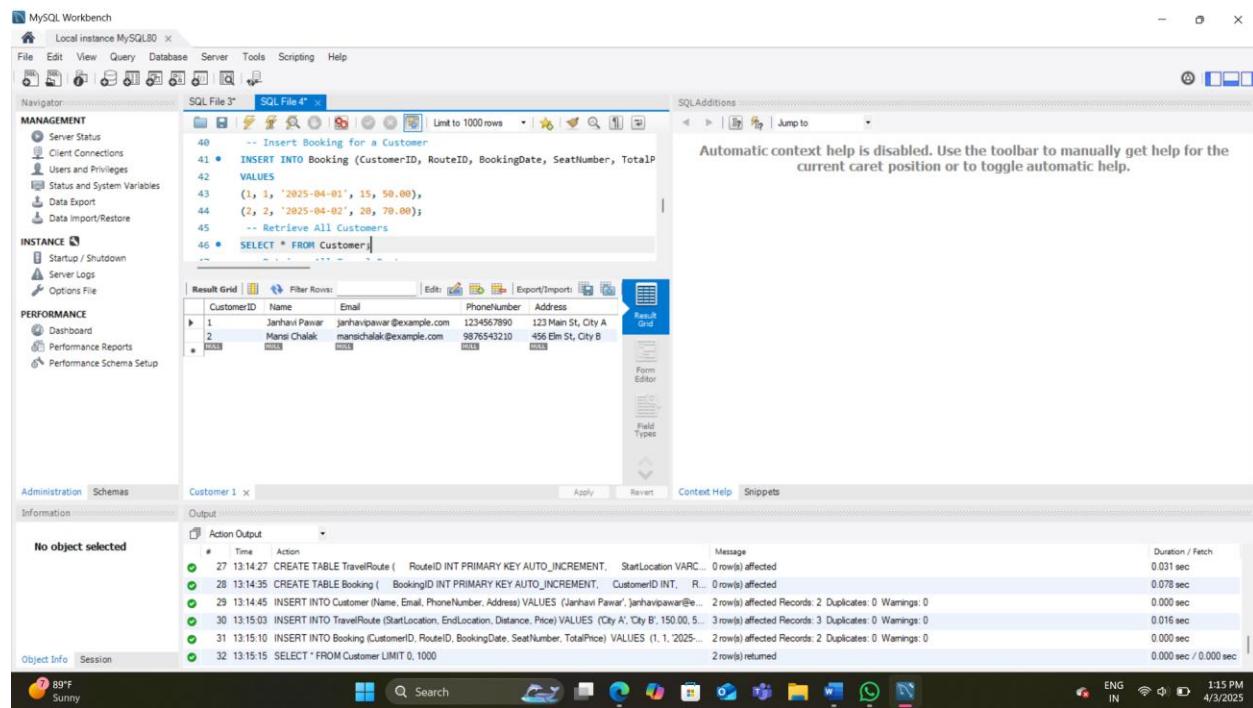
```
VALUES
```

```
(1, 1, '2025-04-01', 15, 50.00),
```

```
(2, 2, '2025-04-02', 20, 70.00);
```

-- Retrieve All Customers

```
SELECT * FROM Customer;
```



CustomerID	Name	Email	PhoneNumber	Address
1	Janhavi Pawar	janhavipawar@example.com	1234567890	123 Main St, City A
2	Mansi Chalak	mansichalak@example.com	9876543210	456 Elm St, City B

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Time	Action	Message	Duration / Fetch
27 13:14:27	CREATE TABLE TravelRoute (RouteID INT PRIMARY KEY AUTO_INCREMENT, StartLocation VARCHAR(255))	0 row(s) affected	0.031 sec
28 13:14:35	CREATE TABLE Booking (BookingID INT PRIMARY KEY AUTO_INCREMENT, CustomerID INT, RouteID INT, BookingDate DATE, SeatNumber INT, TotalPrice DECIMAL(10,2))	0 row(s) affected	0.078 sec
29 13:14:45	INSERT INTO Customer (Name, Email, PhoneNumber, Address) VALUES ('Janhavi Pawar', 'janhavipawar@example.com', '1234567890', '123 Main St, City A')	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
30 13:15:03	INSERT INTO TravelRoute (StartLocation, EndLocation, Distance, Price) VALUES ('City A', 'City B', 150.00, 50.00)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.016 sec
31 13:15:10	INSERT INTO Booking (CustomerID, RouteID, BookingDate, SeatNumber, TotalPrice) VALUES (1, 1, '2025-04-01', 15, 50.00)	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
32 13:15:15	SELECT * FROM Customer LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

-- Retrieve All Travel Routes

```
SELECT * FROM TravelRoute;
```

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: MANAGEMENT, INSTANCE, PERFORMANCE, Administration, Schemas, Information

SQL File 3* SQL File 4*

```

43 (1, 1, '2025-04-01', 15, 50.00),
44 (2, 2, '2025-04-02', 20, 70.00),
45 -- Retrieve All Customers
46 • SELECT * FROM Customer;
47 -- Retrieve All Travel Routes
48 • SELECT * FROM TravelRoute;
49 -- Retrieve All Bookings

```

Result Grid: RouteID, StartLocation, EndLocation, Distance, Price

RouteID	StartLocation	EndLocation	Distance	Price
1	City A	City B	150.00	50.00
2	City B	City C	200.00	70.00
3	City C	City D	300.00	90.00

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas

TravelRoute 2

Output: Action Output

Time	Action	Message	Duration / Fetch
28 13:14:35	CREATE TABLE Booking (BookingID INT PRIMARY KEY AUTO_INCREMENT, CustomerID INT, R...	0 row(s) affected	0.076 sec
29 13:14:45	INSERT INTO Customer (Name, Email, PhoneNumber, Address) VALUES ('Janhavi Pawar', 'janhavipawar@...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.008 sec
30 13:15:03	INSERT INTO TravelRoute (StartLocation, EndLocation, Distance, Price) VALUES ('City A', 'City B', 150.00, 5...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.016 sec
31 13:15:10	INSERT INTO Booking (CustomerID, RouteID, BookingDate, SeatNumber, TotalPrice) VALUES (1, 1, '2025-04-01', 15, 50.00)	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
32 13:15:15	SELECT * FROM Customer LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
33 13:16:00	SELECT * FROM TravelRoute LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

89°F Sunny 11:16 PM 4/3/2025

-- Retrieve All Bookings

SELECT * FROM Booking;

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: MANAGEMENT, INSTANCE, PERFORMANCE, Administration, Schemas, Information

SQL File 3* SQL File 4*

```

46 • SELECT * FROM Customer;
47 -- Retrieve All Travel Routes
48 • SELECT * FROM TravelRoute;
49 -- Retrieve All Bookings
50 • SELECT * FROM Booking;
51 -- Retrieve All Bookings for a Specific Customer (e.g., CustomerID = 1)
52 • SELECT b.BookingID, c.Name AS CustomerName, r.StartLocation, r.EndLocatio

```

Result Grid: BookingID, CustomerID, RouteID, BookingDate, SeatNumber, TotalPrice

BookingID	CustomerID	RouteID	BookingDate	SeatNumber	TotalPrice
1	1	1	2025-04-01	15	50.00
2	2	2	2025-04-02	20	70.00

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas

Booking 3

Output: Action Output

Time	Action	Message	Duration / Fetch
29 13:14:45	INSERT INTO Customer (Name, Email, PhoneNumber, Address) VALUES ('Janhavi Pawar', 'janhavipawar@...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.008 sec
30 13:15:03	INSERT INTO TravelRoute (StartLocation, EndLocation, Distance, Price) VALUES ('City A', 'City B', 150.00, 5...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.016 sec
31 13:15:10	INSERT INTO Booking (CustomerID, RouteID, BookingDate, SeatNumber, TotalPrice) VALUES (1, 1, '2025-04-01', 15, 50.00)	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
32 13:15:15	SELECT * FROM Customer LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
33 13:16:00	SELECT * FROM TravelRoute LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
34 13:16:18	SELECT * FROM Booking LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

89°F Sunny 11:16 PM 4/3/2025

-- Retrieve All Bookings for a Specific Customer (e.g., CustomerID = 1)

SELECT b.BookingID, c.Name AS CustomerName, r.StartLocation, r.EndLocation,

```

b.BookingDate, b.SeatNumber, b.TotalPrice
FROM Booking b
JOIN Customer c ON b.CustomerID = c.CustomerID
JOIN TravelRoute r ON b.RouteID = r.RouteID
WHERE c.CustomerID = 1;

```

```

52 • SELECT b.BookingID, c.Name AS CustomerName, r.StartLocation, r.EndLocation
53   b.BookingDate, b.SeatNumber, b.TotalPrice
54   FROM Booking b
55   JOIN Customer c ON b.CustomerID = c.CustomerID
56   JOIN TravelRoute r ON b.RouteID = r.RouteID
57   WHERE c.CustomerID = 1;
58   -- Update Booking Information (e.g., Update Seat Number for a specific Booking)

```

BookingID	CustomerName	StartLocation	EndLocation	BookingDate	SeatNumber	TotalPrice
1	Janhavi Pawar	City A	City B	2025-04-01	15	50.00

Action Output

#	Time	Action	Message	Duration / Fetch
30	13:15:03	INSERT INTO TravelRoute (StartLocation, EndLocation, Distance, Price) VALUES ('City A', 'City B', 150.00, 50.00)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.016 sec
31	13:15:10	INSERT INTO Booking (CustomerID, RouteID, BookingDate, SeatNumber, TotalPrice) VALUES (1, 1, '2025-04-01', 15, 50.00)	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
32	13:15:15	SELECT * FROM Customer LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
33	13:16:00	SELECT * FROM TravelRoute LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
34	13:16:10	SELECT * FROM Booking LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
35	13:16:50	SELECT b.BookingID, c.Name AS CustomerName, r.StartLocation, r.EndLocation, b.BookingDate, b.SeatNumber, b.TotalPrice FROM Booking b JOIN Customer c ON b.CustomerID = c.CustomerID JOIN TravelRoute r ON b.RouteID = r.RouteID WHERE c.CustomerID = 1;	1 row(s) returned	0.000 sec / 0.000 sec

-- Update Booking Information (e.g., Update Seat Number for a specific Booking)

UPDATE Booking

SET SeatNumber = 25, TotalPrice = 70.00

WHERE BookingID = 1;

-- Delete a Booking

DELETE FROM Booking WHERE BookingID = 2;

-- Display Available Routes for a Customer

SELECT r.RouteID, r.StartLocation, r.EndLocation, r.Price

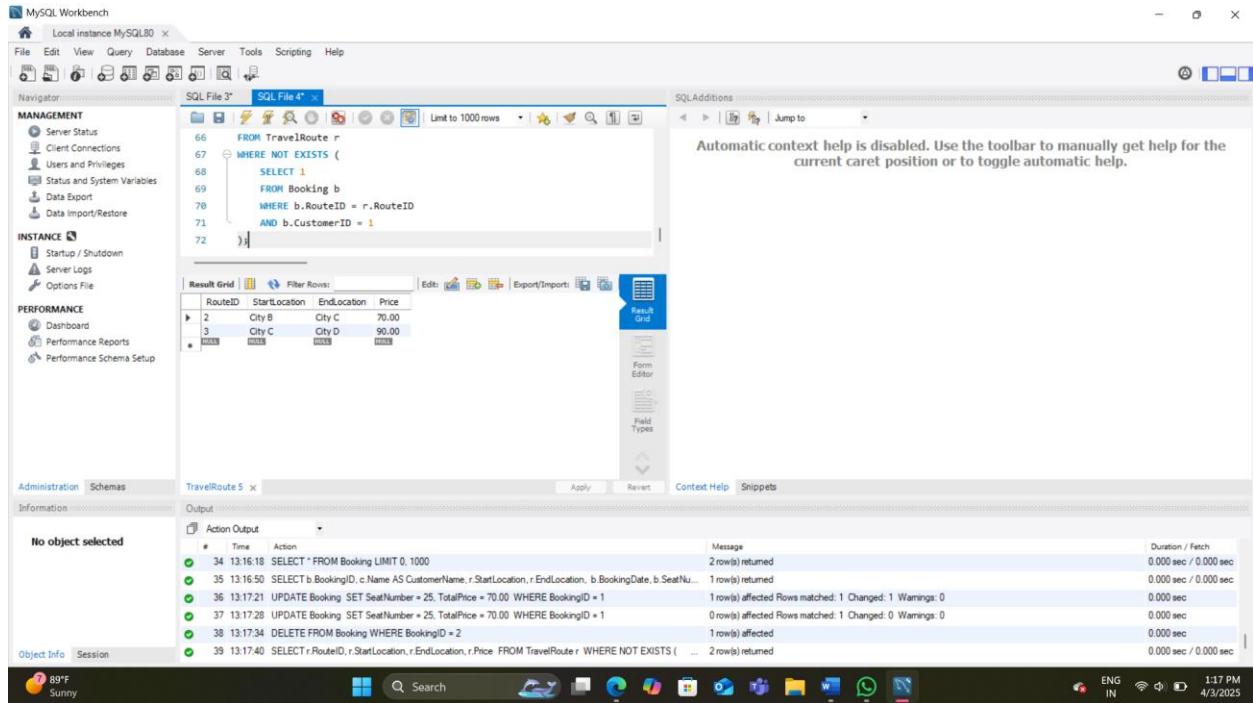
FROM TravelRoute r

WHERE NOT EXISTS (

```

SELECT 1
FROM Booking b
WHERE b.RouteID = r.RouteID
AND b.CustomerID = 1
);

```



MySQL Workbench - Local instance MySQL80

SQL Editor 4:

```

66  FROM TravelRoute r
67  WHERE NOT EXISTS (
68      SELECT 1
69      FROM Booking b
70      WHERE b.RouteID = r.RouteID
71      AND b.CustomerID = 1
72  );

```

Result Grid:

RouteID	StartLocation	EndLocation	Price
2	City B	City C	70.00
3	City C	City D	90.00

Output:

Action	Time	Message	Duration / Fetch
34	13:16:18	SELECT * FROM Booking LIMIT 0, 1000	0.000 sec / 0.000 sec
35	13:16:50	SELECT b.BookingID, c.Name AS CustomerName, r.StartLocation, r.EndLocation, b.BookingDate, b.SeatNumber, b.TotalPrice, b.ActiveStatus FROM Booking b JOIN Customer c ON b.CustomerID = c.CustomerID JOIN TravelRoute r ON b.RouteID = r.RouteID WHERE b.CustomerID = 1	0.000 sec / 0.000 sec
36	13:17:21	UPDATE Booking SET SeatNumber = 25, TotalPrice = 70.00 WHERE BookingID = 1	0.000 sec
37	13:17:28	UPDATE Booking SET SeatNumber = 25, TotalPrice = 70.00 WHERE BookingID = 1	0.000 sec
38	13:17:34	DELETE FROM Booking WHERE BookingID = 2	0.000 sec
39	13:17:40	SELECT r.RouteID, r.StartLocation, r.EndLocation, r.Price FROM TravelRoute r WHERE NOT EXISTS (0.000 sec / 0.000 sec

Practical 3:

Aim: Create a table with columns for EmployeeID, Name, Salary, JoiningDate, and ActiveStatus using different data types. Insert sample data and perform queries to manipulate and retrieve data.

Code:

-- Step 1: Create the Database and Use It

```
CREATE DATABASE EmployeeDB;
```

```
USE EmployeeDB;
```

-- Step 2: Create the Employee Table

```
CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Salary DECIMAL(10,2) CHECK (Salary > 0),
    JoiningDate DATE NOT NULL,
    ActiveStatus BOOLEAN DEFAULT TRUE
);
```

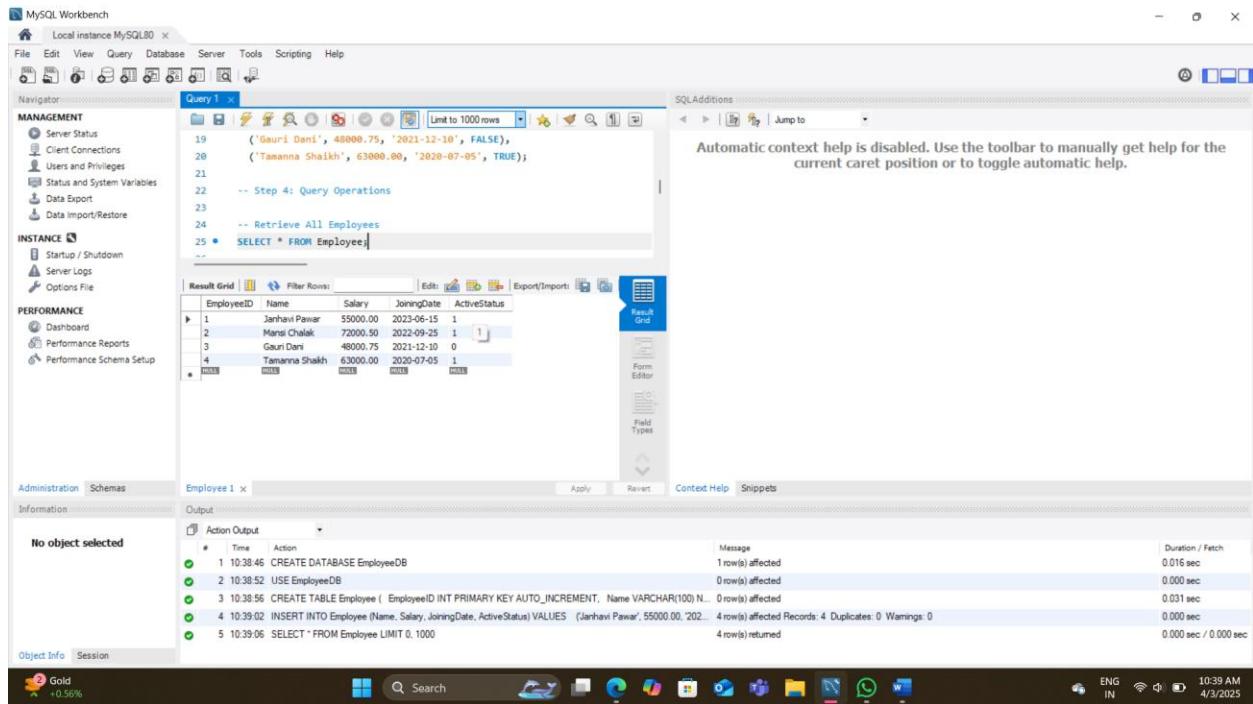
-- Step 3: Insert Sample Data (Updated with New Names)

```
INSERT INTO Employee (Name, Salary, JoiningDate, ActiveStatus)
VALUES
    ('Janhavi Pawar', 55000.00, '2023-06-15', TRUE),
    ('Mansi Chalak', 72000.50, '2022-09-25', TRUE),
    ('Gauri Dani', 48000.75, '2021-12-10', FALSE),
    ('Tamanna Shaikh', 63000.00, '2020-07-05', TRUE);
```

-- Step 4: Query Operations

-- Retrieve All Employees

```
SELECT * FROM Employee;
```



```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator: Local instance MySQL80
MANAGEMENT
  Server Status
  Client Connections
  Users and Privileges
  Status and System Variables
  Data Export
  Data Import/Restore
INSTANCE
  Startup / Shutdown
  Server Logs
  Options File
PERFORMANCE
  Dashboard
  Performance Reports
  Performance Schema Setup

Query 1
19  ('Gauri Dani', 48000.75, '2021-12-10', FALSE),
20  ('Tannanna Shaikh', 63000.00, '2020-07-05', TRUE);
21
22  -- Step 4: Query Operations
23
24  -- Retrieve All Employees
25  SELECT * FROM Employee;
26

Result Grid | Filter Rows: | Edit: | Export/Import: |
EmployeeID Name Salary JoiningDate ActiveStatus
1 Janhavi Pawar 55000.00 2023-06-15 1
2 Mansi Chalak 72000.50 2023-09-25 1
3 Gauri Dani 48000.75 2021-12-10 0
4 Tannanna Shaikh 63000.00 2020-07-05 1

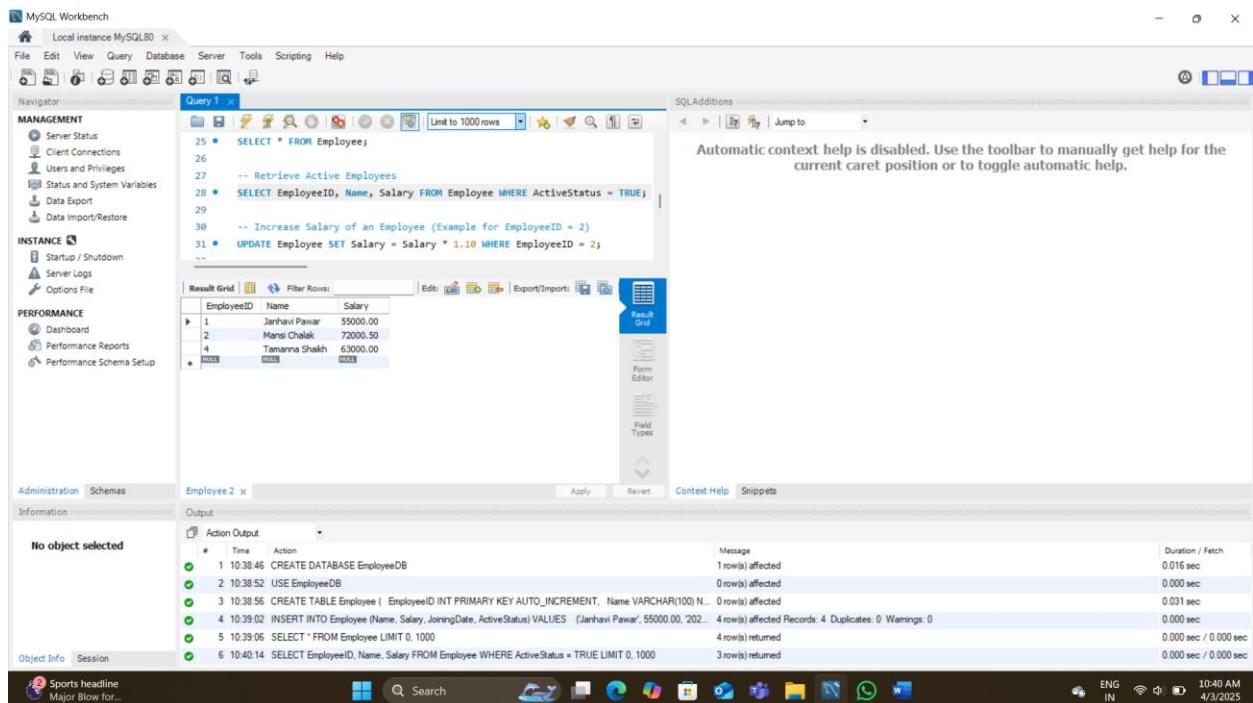
Administration Schemas Employee 1 x
Information Output
No object selected
Action Output
# Time Action Message Duration / Fetch
1 10:38:46 CREATE DATABASE EmployeeDB 1 row(s) affected 0.016 sec
2 10:38:52 USE EmployeeDB 0 row(s) affected 0.000 sec
3 10:38:56 CREATE TABLE Employee ( EmployeeID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(100) N... 0 row(s) affected 0.031 sec
4 10:39:02 INSERT INTO Employee (Name, Salary, JoiningDate, ActiveStatus) VALUES ('Janhavi Pawar', 55000.00, '2023-06-15', 1) 4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0 0.000 sec
5 10:39:06 SELECT * FROM Employee LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec

Object Info Session
0 Gold +0.56%
10:39 AM 4/3/2025

```

-- Retrieve Active Employees

SELECT EmployeeID, Name, Salary FROM Employee WHERE ActiveStatus = TRUE;



```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator: Local instance MySQL80
MANAGEMENT
  Server Status
  Client Connections
  Users and Privileges
  Status and System Variables
  Data Export
  Data Import/Restore
INSTANCE
  Startup / Shutdown
  Server Logs
  Options File
PERFORMANCE
  Dashboard
  Performance Reports
  Performance Schema Setup

Query 1
25  SELECT * FROM Employee;
26
27  -- Retrieve Active Employees
28  SELECT EmployeeID, Name, Salary FROM Employee WHERE ActiveStatus = TRUE;
29
30  -- Increase Salary of an Employee (Example for EmployeeID = 2)
31  UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID = 2;
32

Result Grid | Filter Rows: | Edit: | Export/Import: |
EmployeeID Name Salary
1 Janhavi Pawar 55000.00
2 Mansi Chalak 72000.50
4 Tannanna Shaikh 63000.00

Administration Schemas Employee 2 x
Information Output
No object selected
Action Output
# Time Action Message Duration / Fetch
1 10:38:46 CREATE DATABASE EmployeeDB 1 row(s) affected 0.016 sec
2 10:38:52 USE EmployeeDB 0 row(s) affected 0.000 sec
3 10:38:56 CREATE TABLE Employee ( EmployeeID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(100) N... 0 row(s) affected 0.031 sec
4 10:39:02 INSERT INTO Employee (Name, Salary, JoiningDate, ActiveStatus) VALUES ('Janhavi Pawar', 55000.00, '2023-06-15', 1) 4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0 0.000 sec
5 10:39:06 SELECT * FROM Employee LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
6 10:40:14 UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID = 2; 4 row(s) returned 0.000 sec / 0.000 sec

Object Info Session
Sports headline Major Blow for...
10:40 AM 4/3/2025

```

-- Increase Salary of an Employee (Example for EmployeeID = 2)

UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID = 2;

-- Change Active Status of an Employee (Example for EmployeeID = 4)

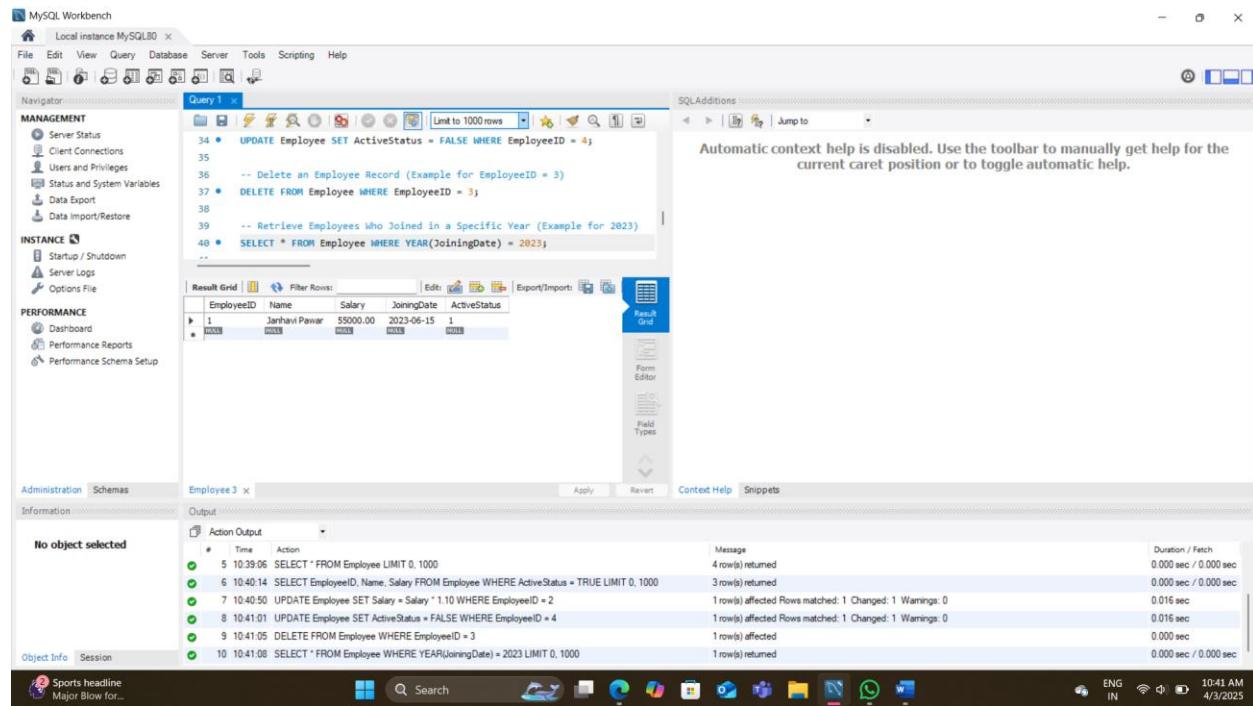
```
UPDATE Employee SET ActiveStatus = FALSE WHERE EmployeeID = 4;
```

-- Delete an Employee Record (Example for EmployeeID = 3)

```
DELETE FROM Employee WHERE EmployeeID = 3;
```

-- Retrieve Employees Who Joined in a Specific Year (Example for 2023)

```
SELECT * FROM Employee WHERE YEAR(JoiningDate) = 2023;
```

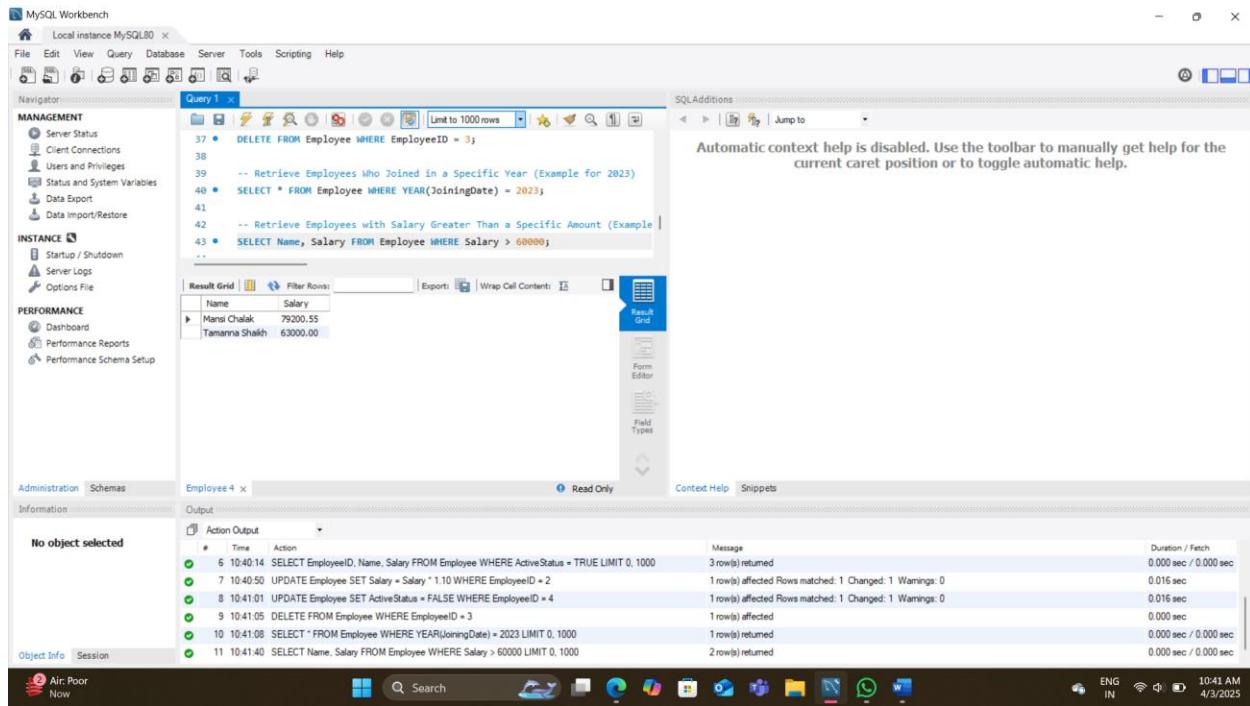


EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Janhavi Pawar	55000.00	2023-06-15	1

Action	Time	Message	Duration / Fetch
5	10:39:06	4 row(s) returned	0.000 sec / 0.000 sec
6	10:40:14	3 row(s) returned	0.000 sec / 0.000 sec
7	10:40:50	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
8	10:41:01	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
9	10:41:05	1 row(s) affected	0.000 sec
10	10:41:08	4 row(s) returned	0.000 sec / 0.000 sec

-- Retrieve Employees with Salary Greater Than a Specific Amount (Example for 60000)

```
SELECT Name, Salary FROM Employee WHERE Salary > 60000;
```



MySQL Workbench - Local instance MySQL80

Query 1

```

37 • DELETE FROM Employee WHERE EmployeeID = 3
38
39 -- Retrieve Employees Who Joined in a Specific Year (Example for 2023)
40 • SELECT * FROM Employee WHERE YEAR(JoiningDate) = 2023;
41
42 -- Retrieve Employees with Salary Greater Than a Specific Amount (Example
43 • SELECT Name, Salary FROM Employee WHERE Salary > 60000;
44

```

Result Grid

Name	Salary
Mani Chalak	79200.55
Tanmaya Shalih	63000.00

Employee 4

Read Only

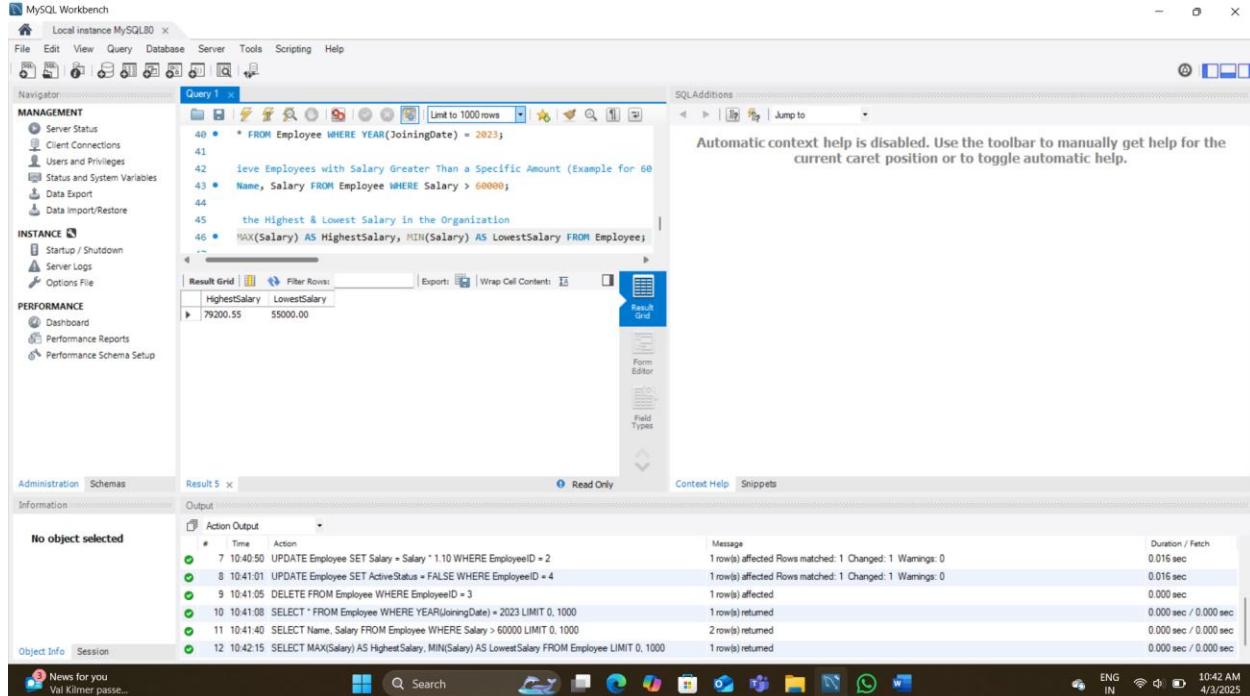
Action Output

Time	Action	Message	Duration / Fetch
6 10:40:14	SELECT EmployeeID, Name, Salary FROM Employee WHERE ActiveStatus = TRUE LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
7 10:40:50	UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID = 2	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
8 10:41:01	UPDATE Employee SET ActiveStatus = FALSE WHERE EmployeeID = 4	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
9 10:41:05	DELETE FROM Employee WHERE EmployeeID = 3	1 row(s) affected	0.000 sec
10 10:41:08	SELECT * FROM Employee WHERE YEAR(JoiningDate) = 2023 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
11 10:41:40	SELECT Name, Salary FROM Employee WHERE Salary > 60000 LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

10:41 AM 4/3/2025

-- Find the Highest & Lowest Salary in the Organization

SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS LowestSalary FROM Employee;



MySQL Workbench - Local instance MySQL80

Query 1

```

40 • * FROM Employee WHERE YEAR(JoiningDate) = 2023;
41
42 -- Retrieve Employees with Salary Greater Than a Specific Amount (Example for 2023)
43 • SELECT Name, Salary FROM Employee WHERE Salary > 60000;
44
45 -- Find the Highest & Lowest Salary in the Organization
46 • MAX(Salary) AS HighestSalary, MIN(Salary) AS LowestSalary FROM Employee;

```

Result Grid

HighestSalary	LowestSalary
79200.55	55000.00

Result 5

Read Only

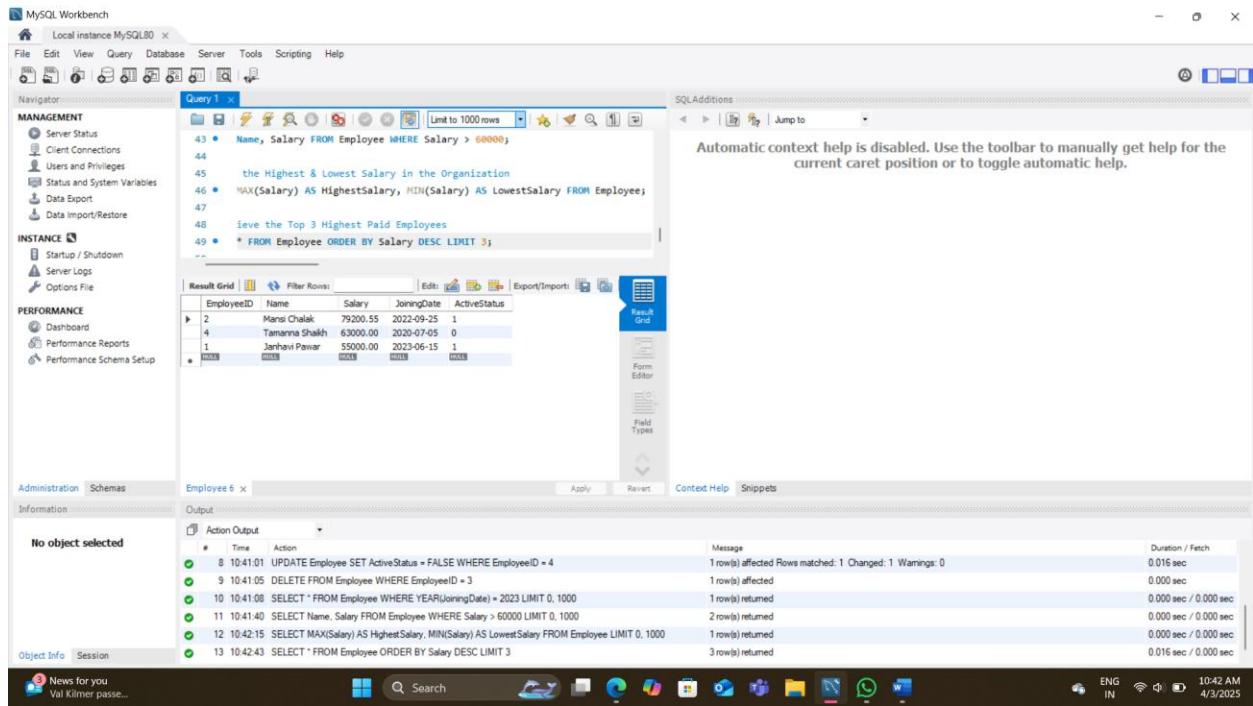
Action Output

Time	Action	Message	Duration / Fetch
7 10:40:50	UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID = 2	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
8 10:41:01	UPDATE Employee SET ActiveStatus = FALSE WHERE EmployeeID = 4	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
9 10:41:05	DELETE FROM Employee WHERE EmployeeID = 3	1 row(s) affected	0.000 sec
10 10:41:08	SELECT * FROM Employee WHERE YEAR(JoiningDate) = 2023 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
11 10:41:40	SELECT Name, Salary FROM Employee WHERE Salary > 60000 LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
12 10:42:15	SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS LowestSalary FROM Employee LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

10:42 AM 4/3/2025

-- Retrieve the Top 3 Highest Paid Employees

SELECT * FROM Employee ORDER BY Salary DESC LIMIT 3;



Practical 4:

Aim: Create a table to store employee information with constraints like Primary Key, Foreign Key, and Unique.

Code:

-- Drop existing tables if they exist

DROP TABLE IF EXISTS Employee;

DROP TABLE IF EXISTS Department;

-- Create the Department Table

CREATE TABLE Department (

DeptID INT PRIMARY KEY,

DeptName VARCHAR(50) UNIQUE

);

```
-- Create the Employee Table
```

```
CREATE TABLE Employee (
    EmpID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Salary DECIMAL(10,2) CHECK (Salary > 0),
    DeptID INT REFERENCES Department(DeptID)
);
```

```
-- Insert Data into Department Table
```

```
INSERT INTO Department (DeptID, DeptName) VALUES (1, 'HR');
INSERT INTO Department (DeptID, DeptName) VALUES (2, 'IT');
```

```
-- Insert Data into Employee Table
```

```
INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (101, 'Janhavi',
'janhavi@example.com', 50000.00, 1);
INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (102, 'Mansi',
'mansi@example.com', 60000.00, 2);
INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (103, 'Gauri',
'gauri@example.com', 55000.00, 1);
INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (104, 'Sanjivani',
'sanjivani@example.com', 65000.00, 2);
```

```
-- Select all data from Department Table
```

```
SELECT * FROM Department;
```

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

companydb

Tables Views Stored Procedures Functions

customerdb employeedb roadwaytraveldb salesdb sys

SQL File 2* x

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```

25 • INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (101, 'Janhavi', 50000.00, 1)
26 • INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (102, 'Mansi', 60000.00, 2)
27 • INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (103, 'Gauri', 55000.00, 1)
28 • INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (104, 'Sanjivani', 65000.00, 2)
29
30 -- Select all data from Department Table
31 • SELECT * FROM Department;
32

```

Result Grid | Filter Rows: | Edit | Export/Import: |

DeptID	DeptName
1	HR
2	IT
3	Marketing
4	Finance

Administration Schemas Department 1 x

Information Output

Action Output

#	Time	Action	Message	Duration / Fetch
13	10:14:06	INSERT INTO Department (DeptID, DeptName) VALUES (2, 'IT')	1 row(s) affected	0.016 sec
14	10:14:16	INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (101, 'Janhavi', 'janhavi@example.com', 50000.00, 1)	1 row(s) affected	0.006 sec
15	10:14:20	INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (102, 'Mansi', 'mansi@example.com', 60000.00, 2)	1 row(s) affected	0.016 sec
16	10:14:23	INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (103, 'Gauri', 'gauri@example.com', 55000.00, 1)	1 row(s) affected	0.015 sec
17	10:14:25	INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (104, 'Sanjivani', 'sanjivani@example.com', 65000.00, 2)	1 row(s) affected	0.015 sec
18	10:14:30	SELECT * FROM Department LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

80°F Haze 10:14 AM 4/4/2025 ENG IN

-- Select all data from Employee Table

SELECT * FROM Employee;

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

companydb

Tables Views Stored Procedures Functions

customerdb employeedb roadwaytraveldb salesdb sys

SQL File 2* x

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```

28 • INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (104, 'Sanjivani', 65000.00, 2)
29
30 -- Select all data from Department Table
31 • SELECT * FROM Department;
32
33 -- Select all data from Employee Table
34 • SELECT * FROM Employee;
35

```

Result Grid | Filter Rows: | Edit | Export/Import: |

EmpID	Name	Email	Salary	DeptID
101	Janhavi	janhavi@example.com	50000.00	1
102	Mansi	mansi@example.com	60000.00	2
103	Gauri	gauri@example.com	55000.00	1
104	Sanjivani	sanjivani@example.com	65000.00	2
105	HR	HR@example.com	55000.00	3

Administration Schemas Employee 2 x

Information Output

Action Output

#	Time	Action	Message	Duration / Fetch
14	10:14:16	INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (101, 'Janhavi', 'janhavi@example.com', 50000.00, 1)	1 row(s) affected	0.000 sec
15	10:14:20	INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (102, 'Mansi', 'mansi@example.com', 60000.00, 2)	1 row(s) affected	0.016 sec
16	10:14:23	INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (103, 'Gauri', 'gauri@example.com', 55000.00, 1)	1 row(s) affected	0.015 sec
17	10:14:25	INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (104, 'Sanjivani', 'sanjivani@example.com', 65000.00, 2)	1 row(s) affected	0.015 sec
18	10:14:30	SELECT * FROM Department LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
19	10:14:54	SELECT * FROM Employee LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

80°F Haze 10:14 AM 4/4/2025 ENG IN

Practical 5:

Aim: To Test constraints like PRIMARY KEY, UNIQUE, and CHECK by inserting invalid data into the Employee table.

Code:

```
-- Create the Database
```

```
CREATE DATABASE CustomerDB;
```

```
-- Use the Database
```

```
USE CustomerDB;
```

```
-- Create the Customer Table
```

```
CREATE TABLE Customer (
```

```
    CustomerID INT PRIMARY KEY,
```

```
    FirstName VARCHAR(100) NOT NULL,
```

```
    LastName VARCHAR(100) NOT NULL,
```

```
    Email VARCHAR(100) UNIQUE,
```

```
    Phone VARCHAR(15),
```

```
    Age INT CHECK (Age >= 18),
```

```
    IsActive BOOLEAN DEFAULT TRUE
```

```
);
```

```
-- Insert Valid Data
```

```
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age, IsActive)
```

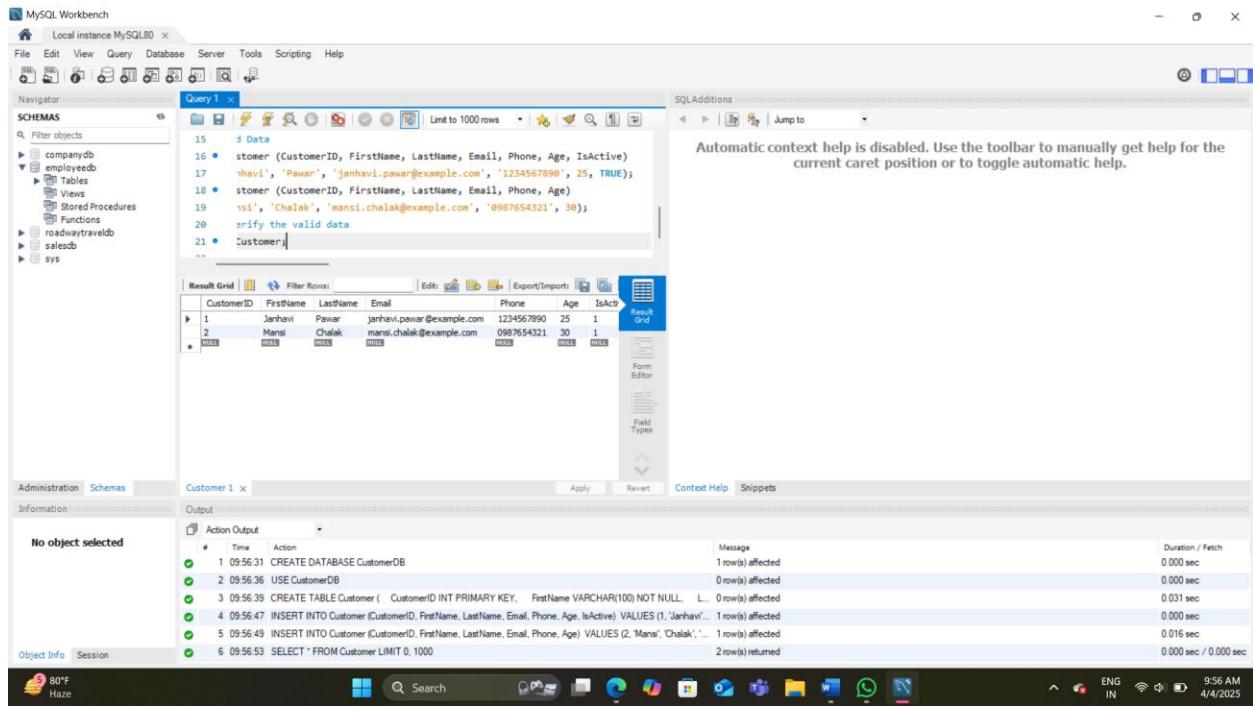
```
VALUES (1, 'Janhavi', 'Pawar', 'janhavi.pawar@example.com', '1234567890', 25, TRUE);
```

```
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age)
```

```
VALUES (2, 'Mansi', 'Chalak', 'mansi.chalak@example.com', '0987654321', 30);
```

```
-- Select to verify the valid data
```

```
SELECT * FROM Customer;
```



Practical 6:

Aim: Use DDL commands to create tables and DML commands to insert, update, and delete data. Write SELECT queries to retrieve and verify data changes.

Code:

-- Create the Database

```
CREATE DATABASE EmployeeDB;
```

-- Use the Database

```
USE EmployeeDB;
```

-- Create the Employees Table

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Age INT,
    Department VARCHAR(50),
    Salary DECIMAL(10, 2)
);
```

-- Insert Data into Employees Table

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department, Salary)
VALUES (1, 'Janhavi', 'Pawar', 28, 'HR', 50000.00);
```

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department, Salary)
VALUES (2, 'Mansi', 'Chalak', 35, 'IT', 65000.00);
```

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department, Salary)
VALUES (3, 'Gauri', 'Dani', 40, 'Finance', 75000.00);
```

-- Updates (DML Commands)

-- 1. Update a single column (e.g., update salary for EmployeeID 2)

```
UPDATE Employees
SET Salary = 70000.00
WHERE EmployeeID = 2;
```

-- 2. Update multiple columns for a specific row (e.g., update name and salary for EmployeeID 2)

UPDATE Employees

SET FirstName = 'Janhavi', LastName = 'Williams', Salary = 75000.00

WHERE EmployeeID = 2;

-- 3. Update entire tuple (all columns for EmployeeID 3)

UPDATE Employees

SET FirstName = 'Mansi', LastName = 'Brown', Age = 45, Department = 'Management', Salary = 80000.00

WHERE EmployeeID = 3;

-- 4. Update with a condition (e.g., increase salary by 10% for all employees in HR)

UPDATE Employees

SET Salary = Salary * 1.10

WHERE Department = 'HR';

-- 5. Update with a subquery (e.g., increase salary for Employee with highest salary)

UPDATE Employees

SET Salary = Salary + 5000

WHERE Salary = (SELECT MAX(Salary) FROM Employees);

-- 6. Update using a CASE statement (e.g., increase salary based on department)

UPDATE Employees

SET Salary = CASE

WHEN Department = 'HR' THEN Salary * 1.05

WHEN Department = 'IT' THEN Salary * 1.08

```
WHEN Department = 'Finance' THEN Salary * 1.10
```

```
ELSE Salary
```

```
END;
```

```
-- Delete Data from the Table (DML Command)
```

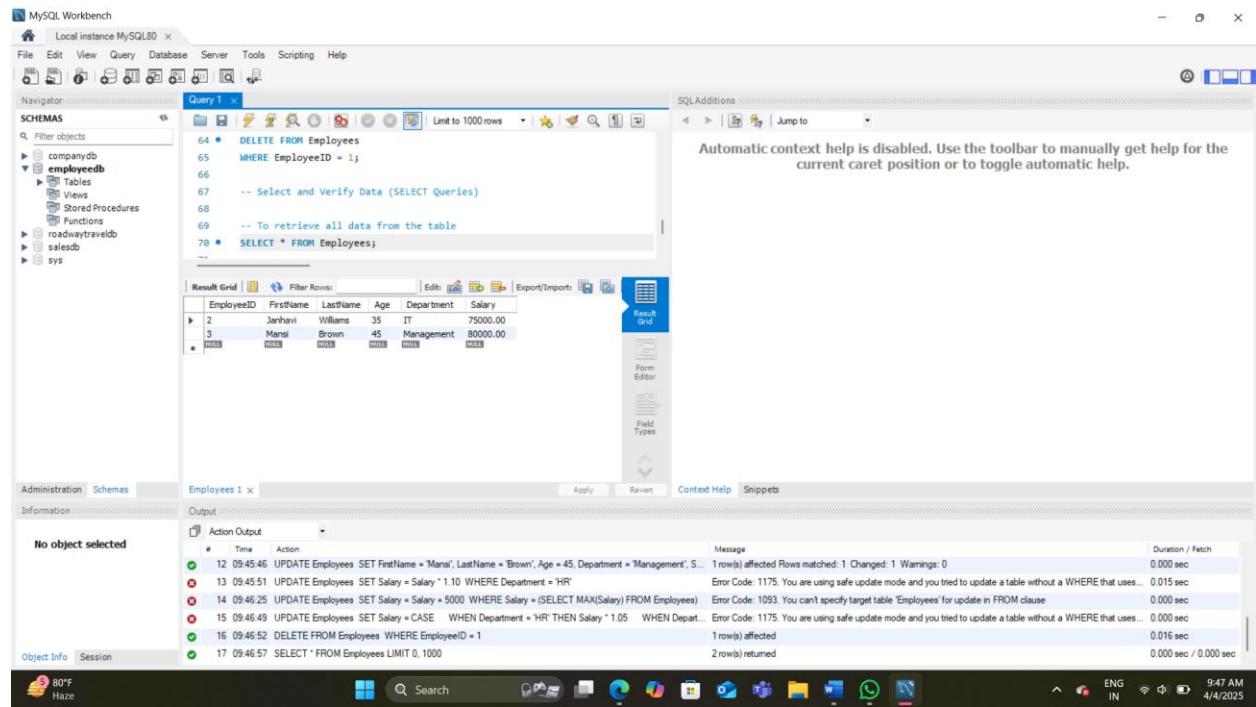
```
DELETE FROM Employees
```

```
WHERE EmployeeID = 1;
```

```
-- Select and Verify Data (SELECT Queries)
```

```
-- To retrieve all data from the table
```

```
SELECT * FROM Employees;
```



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the following SQL code:

```
64 • DELETE FROM Employees
65 WHERE EmployeeID = 1;
66
67 -- Select and Verify Data (SELECT Queries)
68
69 -- To retrieve all data from the table
70 • SELECT * FROM Employees;
```
- Result Grid:** Displays the results of the SELECT query, showing two rows of employee data:

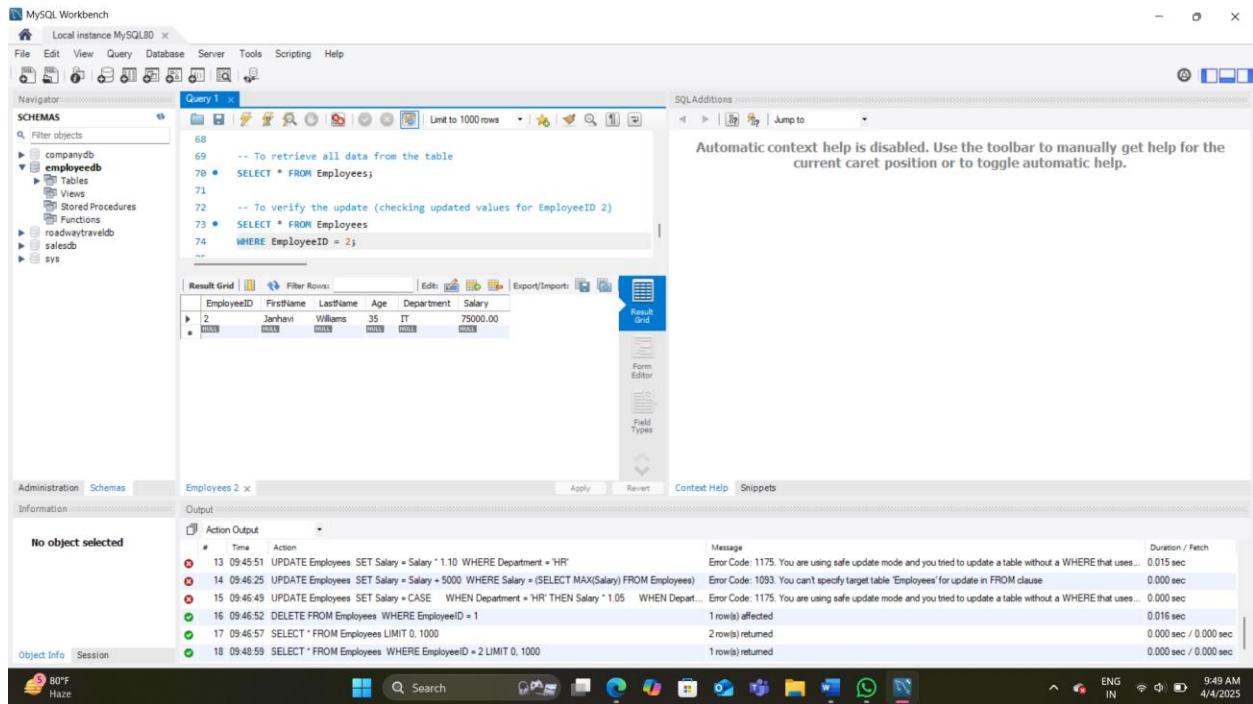
EmployeeID	FirstName	LastName	Age	Department	Salary
2	Janavi	Williams	35	IT	75000.00
3	Mansi	Brown	45	Management	80000.00
- Output Window:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
12	09:45:46	UPDATE Employees SET FirstName = 'Mansi', LastName = 'Brown', Age = 45, Department = 'Management', Salary = 80000.00 WHERE EmployeeID = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
13	09:45:51	UPDATE Employees SET Salary = Salary * 1.10 WHERE Department = 'HR'	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a column that is foreign to the table. This is dangerous and could cause data to be updated that you did not intend.	0.015 sec
14	09:46:25	UPDATE Employees SET Salary = Salary + 5000 WHERE Salary = (SELECT MAX(Salary) FROM Employees)	Error Code: 1093. You can't specify target table 'Employees' for update in FROM clause	0.000 sec
15	09:46:49	UPDATE Employees SET Salary = CASE WHEN Department = 'HR' THEN Salary * 1.05 WHEN Department = 'Management' THEN Salary * 1.10 ELSE Salary END	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a column that is foreign to the table. This is dangerous and could cause data to be updated that you did not intend.	0.000 sec
16	09:46:52	DELETE FROM Employees WHERE EmployeeID = 1	1 row(s) affected	0.016 sec
17	09:46:57	SELECT * FROM Employees LIMIT 0,1000	2 row(s) returned	0.000 sec / 0.000 sec

```
-- To verify the update (checking updated values for EmployeeID 2)
```

```
SELECT * FROM Employees
```

```
WHERE EmployeeID = 2;
```



MySQL Workbench - Local instance MySQL80

Query 1

```

68 -- To retrieve all data from the table
70 • SELECT * FROM Employees;
71
72 -- To verify the update (checking updated values for EmployeeID 2)
73 • SELECT * FROM Employees
74 WHERE EmployeeID = 2;
75

```

Result Grid

EmployeeID	FirstName	LastName	Age	Department	Salary
2	Janhavi	Williams	35	IT	75000.00

Administration Schemas

Employees 2

Action Output

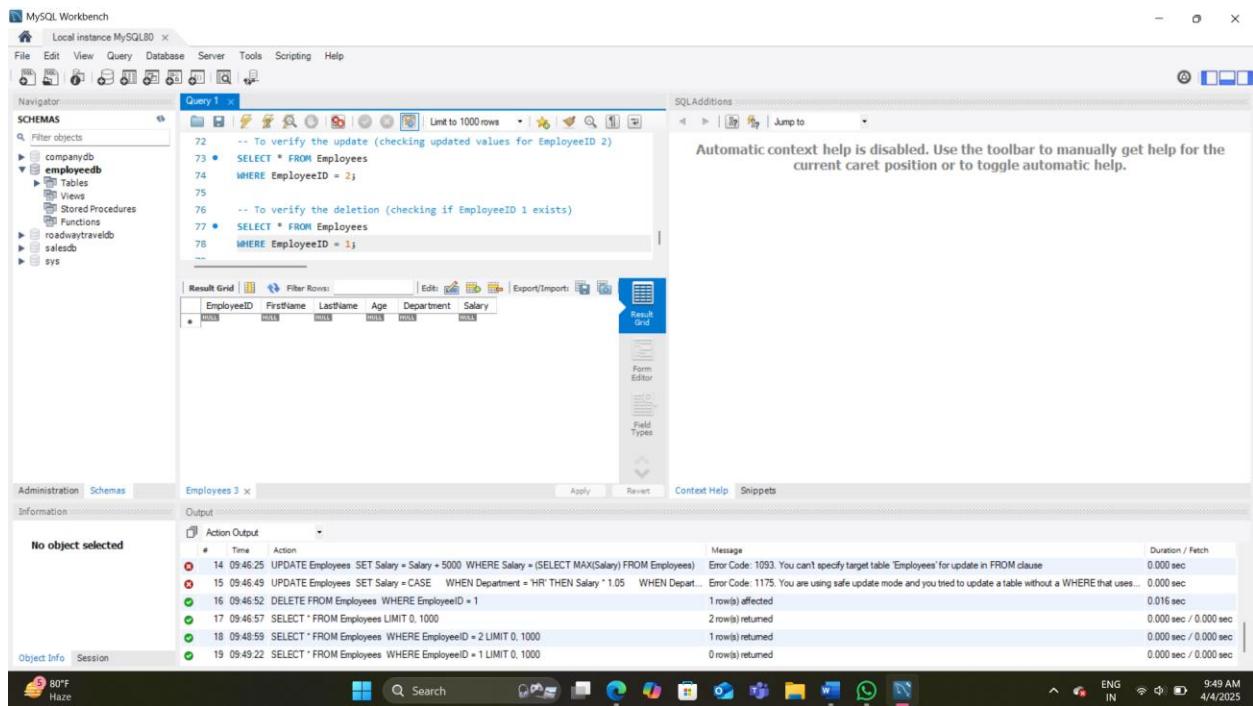
Time	Action	Message	Duration / Fetch
13:09:45.51	UPDATE Employees SET Salary = Salary + 1.10 WHERE Department = 'HR'	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses...	0.015 sec
14:09:46.25	UPDATE Employees SET Salary = Salary + 5000 WHERE Salary = (SELECT MAX(Salary) FROM Employees)	Error Code: 1093. You can't specify target table 'Employees' for update in FROM clause	0.000 sec
15:09:46.49	UPDATE Employees SET Salary = CASE WHEN Department = 'HR' THEN Salary + 1.05 WHEN Depart...	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses...	0.000 sec
16:09:46.52	DELETE FROM Employees WHERE EmployeeID = 1	1row(s) affected	0.016 sec
17:09:46.57	SELECT * FROM Employees LIMIT 0, 1000	2row(s) returned	0.000 sec / 0.000 sec
18:09:48.59	SELECT * FROM Employees WHERE EmployeeID = 2 LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec

80°F Haze 9:49 AM 4/4/2025

-- To verify the deletion (checking if EmployeeID 1 exists)

SELECT * FROM Employees

WHERE EmployeeID = 1;



MySQL Workbench - Local instance MySQL80

Query 1

```

72 -- To verify the update (checking updated values for EmployeeID 2)
73 • SELECT * FROM Employees
74 WHERE EmployeeID = 2;
75
76 -- To verify the deletion (checking if EmployeeID 1 exists)
77 • SELECT * FROM Employees
78 WHERE EmployeeID = 1;
79

```

Result Grid

EmployeeID	FirstName	LastName	Age	Department	Salary
1					

Administration Schemas

Employees 3

Action Output

Time	Action	Message	Duration / Fetch
14:09:46.25	UPDATE Employees SET Salary = Salary + 5000 WHERE Salary = (SELECT MAX(Salary) FROM Employees)	Error Code: 1093. You can't specify target table 'Employees' for update in FROM clause	0.000 sec
15:09:46.49	UPDATE Employees SET Salary = CASE WHEN Department = 'HR' THEN Salary + 1.05 WHEN Depart...	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses...	0.000 sec
16:09:46.52	DELETE FROM Employees WHERE EmployeeID = 1	1row(s) affected	0.016 sec
17:09:46.57	SELECT * FROM Employees LIMIT 0, 1000	2row(s) returned	0.000 sec / 0.000 sec
18:09:48.59	SELECT * FROM Employees WHERE EmployeeID = 2 LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
19:09:49.22	SELECT * FROM Employees WHERE EmployeeID = 1 LIMIT 0, 1000	0row(s) returned	0.000 sec / 0.000 sec

80°F Haze 9:49 AM 4/4/2025

Practical 7:

Aim: Create a Sales table and use aggregate functions like COUNT, SUM, AVG, MIN, and MAX to summarize sales data and calculate statistics.

Code:

```
-- Create the Database
```

```
CREATE DATABASE SalesDB;
```

```
-- Use the Database
```

```
USE SalesDB;
```

```
-- Create the Sales Table
```

```
CREATE TABLE Sales (
    SaleID INT PRIMARY KEY AUTO_INCREMENT,
    Product VARCHAR(50),
    Quantity INT,
    Price DECIMAL(10,2),
    SaleDate DATE
);
```

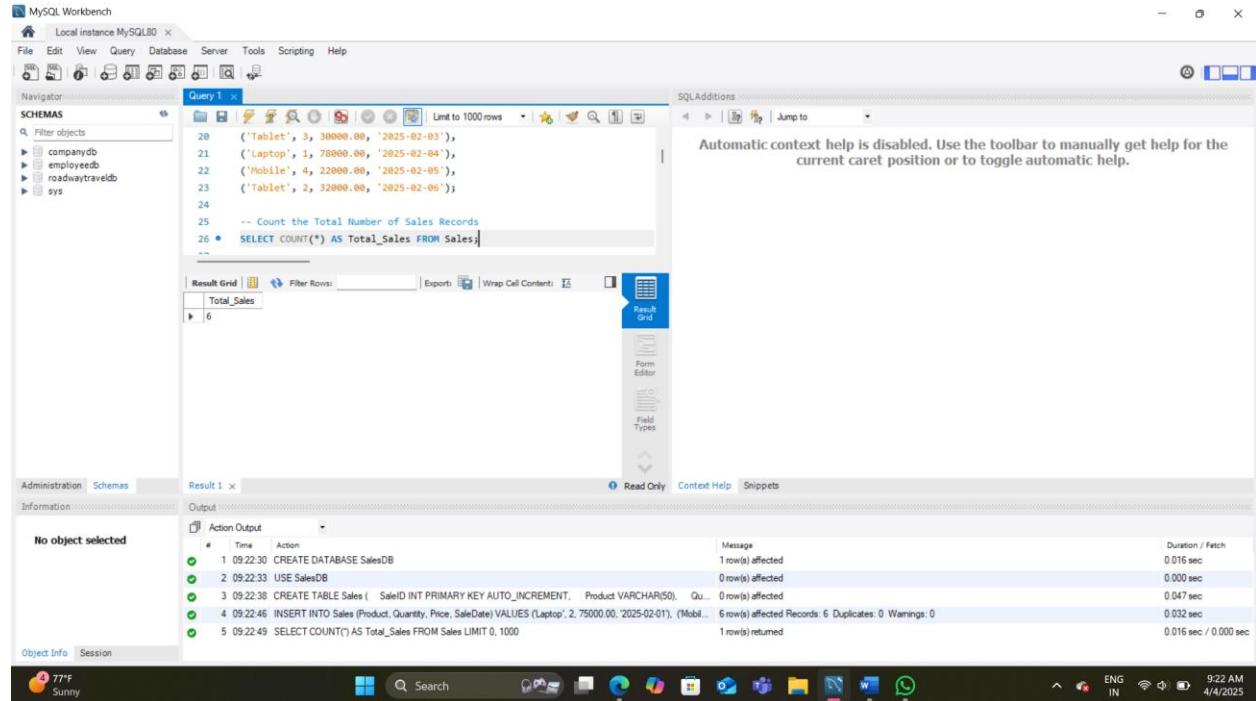
```
-- Insert Sample Data into Sales Table
```

```
INSERT INTO Sales (Product, Quantity, Price, SaleDate) VALUES
('Laptop', 2, 75000.00, '2025-02-01'),
('Mobile', 5, 20000.00, '2025-02-02'),
('Tablet', 3, 30000.00, '2025-02-03'),
('Laptop', 1, 78000.00, '2025-02-04'),
('Mobile', 4, 22000.00, '2025-02-05'),
```

```
('Tablet', 2, 32000.00, '2025-02-06');
```

-- Count the Total Number of Sales Records

```
SELECT COUNT(*) AS Total_Sales FROM Sales;
```



The screenshot shows the MySQL Workbench interface. The 'Query 1' tab contains the following SQL code:

```
20 ('Tablet', 2, 30000.00, '2025-02-03'),  
21 ('Laptop', 1, 78000.00, '2025-02-04'),  
22 ('Mobile', 4, 22000.00, '2025-02-05'),  
23 ('Tablet', 2, 32000.00, '2025-02-06');  
24  
25 -- Count the Total Number of Sales Records  
26 • SELECT COUNT(*) AS Total_Sales;
```

The 'Result Grid' shows the output of the query:

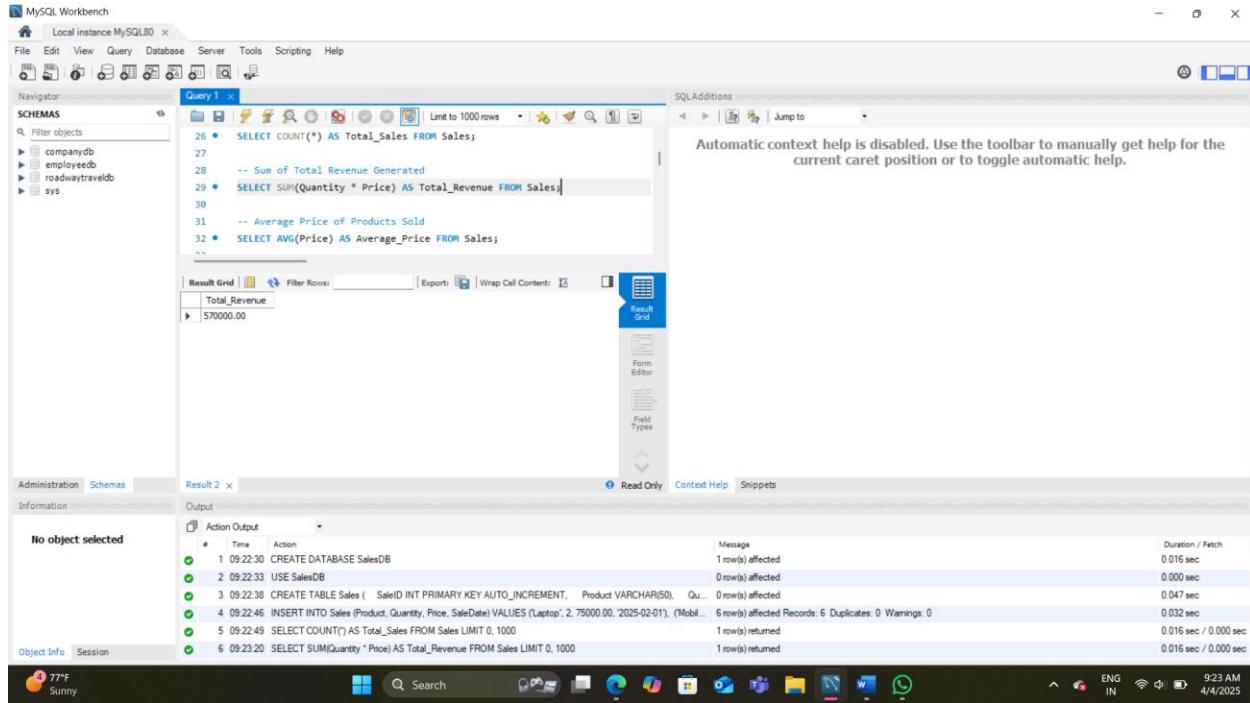
Total_Sales
6

The 'Information' tab shows the 'Action Output' log:

Action	Time	Message	Duration / Fetch
1	09:22:30	CREATE DATABASE SalesDB	0.016 sec
2	09:22:33	USE SalesDB	0.000 sec
3	09:22:38	CREATE TABLE Sales (SaleID INT PRIMARY KEY AUTO_INCREMENT, Product VARCHAR(50), Quantity INT, Price DECIMAL(10,2), SaleDate DATE)	0.047 sec
4	09:22:46	INSERT INTO Sales (Product, Quantity, Price, SaleDate) VALUES ('Laptop', 2, 75000.00, '2025-02-01'), ('Mobile', 4, 22000.00, '2025-02-05'), ('Tablet', 2, 30000.00, '2025-02-03'), ('Tablet', 2, 32000.00, '2025-02-06')	0.032 sec
5	09:22:49	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	0.016 sec / 0.000 sec

-- Sum of Total Revenue Generated

```
SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales;
```



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1

```

26 • SELECT COUNT(*) AS Total_Sales FROM Sales;
27
28 -- Sum of Total Revenue Generated
29 • SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales;
30
31 -- Average Price of Products Sold
32 • SELECT AVG(Price) AS Average_Price FROM Sales;
33

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid

Total_Revenue
570000.00

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas Result 2 x Read Only Context Help Snippets

Information Output

Action Output

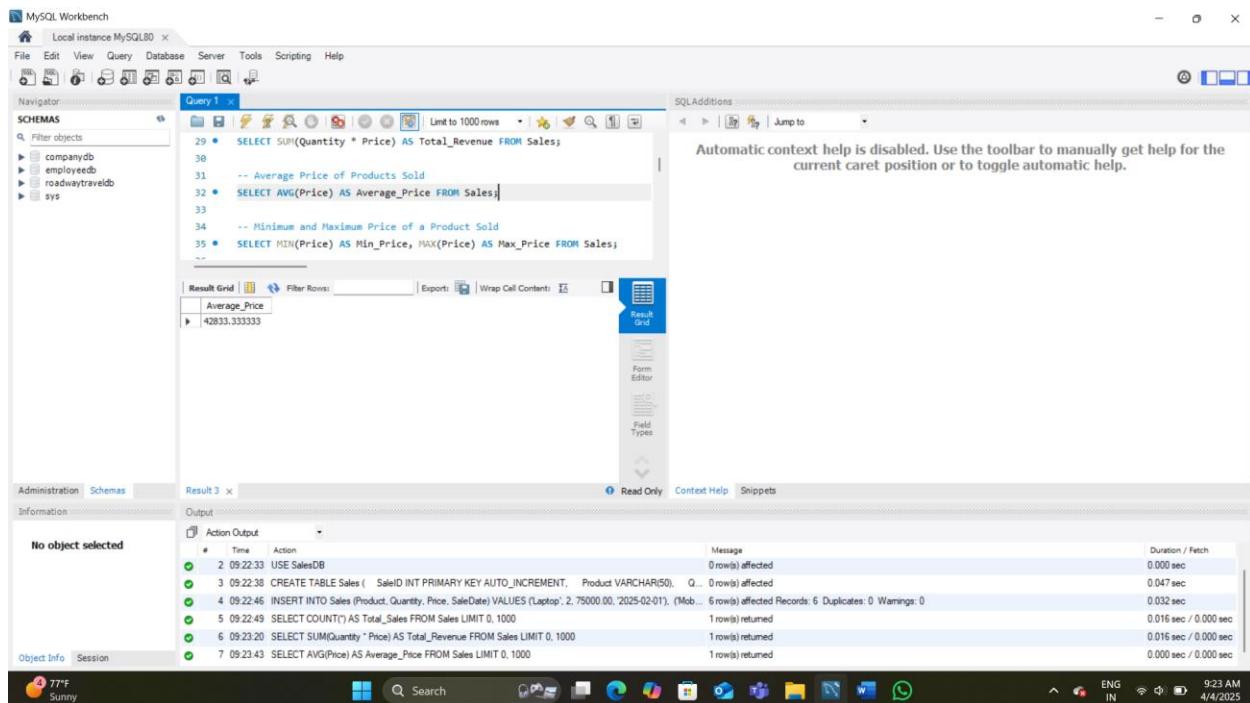
Time	Action	Message	Duration / Fetch
1 09:22:30	CREATE DATABASE SalesDB	1 row(s) affected	0.016 sec
2 09:22:33	USE SalesDB	0 row(s) affected	0.006 sec
3 09:22:38	CREATE TABLE Sales (SaleID INT PRIMARY KEY AUTO_INCREMENT, Product VARCHAR(50), Qu...	0 row(s) affected	0.047 sec
4 09:22:46	INSERT INTO Sales (Product, Quantity, Price, SaleDate) VALUES ('Laptop', 2, 75000.00, '2025-02-01'), ('Mobi...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.032 sec
5 09:22:49	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
6 09:23:20	SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

Object Info Session

77°F Sunny 9:23 AM 4/4/2025

-- Average Price of Products Sold

SELECT AVG(Price) AS Average_Price FROM Sales;



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1

```

29 • SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales;
30
31 -- Average Price of Products Sold
32 • SELECT AVG(Price) AS Average_Price FROM Sales;
33
34 -- Minimum and Maximum Price of a Product Sold
35 • SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales;
36

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid

Average_Price
42833.33333

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas Result 3 x Read Only Context Help Snippets

Information Output

Action Output

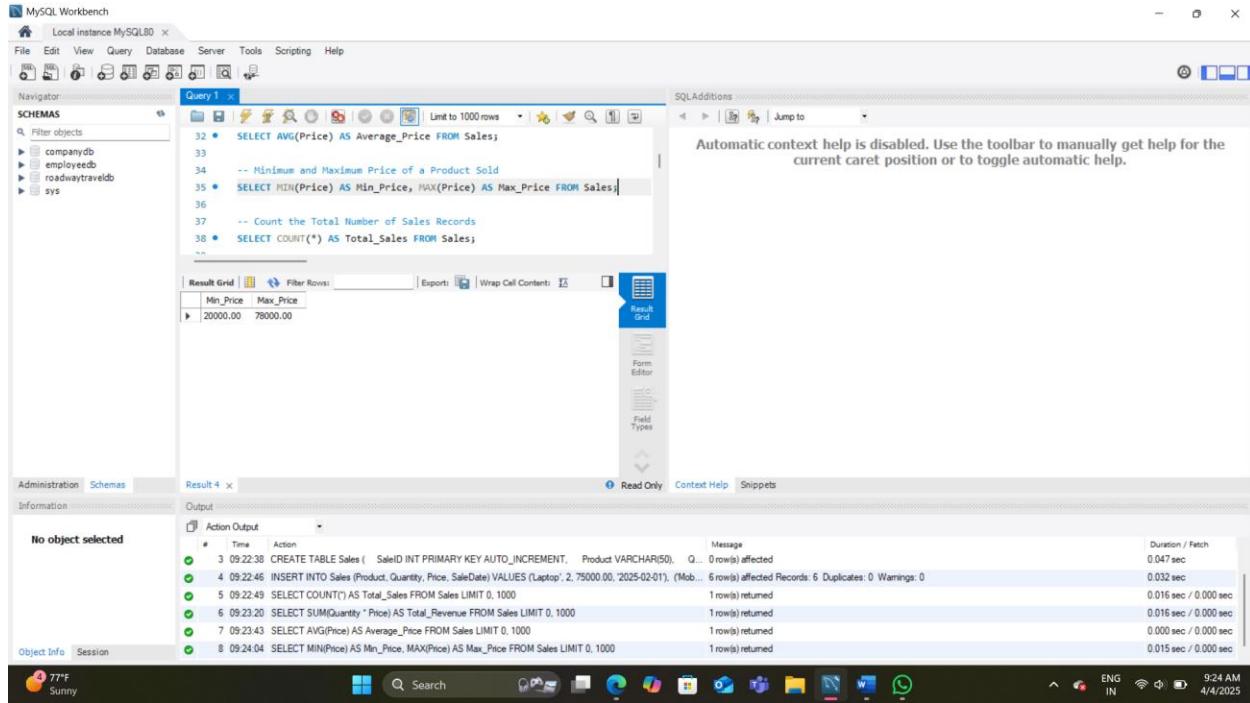
Time	Action	Message	Duration / Fetch
2 09:22:33	USE SalesDB	0 row(s) affected	0.000 sec
3 09:22:38	CREATE TABLE Sales (SaleID INT PRIMARY KEY AUTO_INCREMENT, Product VARCHAR(50), Qu...	0 row(s) affected	0.047 sec
4 09:22:46	INSERT INTO Sales (Product, Quantity, Price, SaleDate) VALUES ('Laptop', 2, 75000.00, '2025-02-01'), ('Mobi...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.032 sec
5 09:22:49	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
6 09:23:20	SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
7 09:23:43	SELECT AVG(Price) AS Average_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

77°F Sunny 9:23 AM 4/4/2025

-- Minimum and Maximum Price of a Product Sold

SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales;



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1

```

32 • SELECT AVG(Price) AS Average_Price FROM Sales;
33
34 -- Minimum and Maximum Price of a Product Sold
35 • SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales;
36
37 -- Count the Total Number of Sales Records
38 • SELECT COUNT(*) AS Total_Sales FROM Sales;

```

Result Grid | Filter Rows: Export | Wrap Cell Contents | Result Grid | Form Editor | Field Types

Min_Price	Max_Price
20000.00	78000.00

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas Result 4

Information Output

Action Output

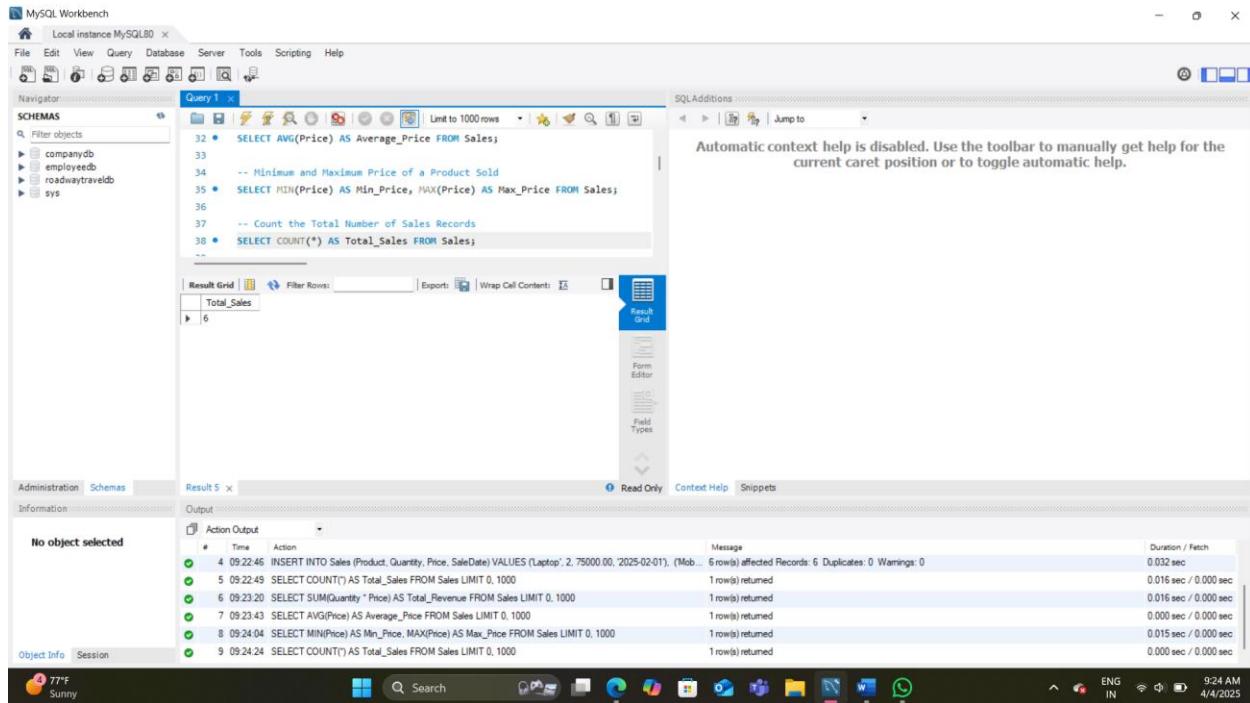
#	Time	Action	Message	Duration / Fetch
3	09:22:38	CREATE TABLE Sales (0 rows affected	0.047 sec
4	09:22:46	INSERT INTO Sales (Product, Quantity, Price, SaleDate) VALUES ('Laptop', 2, 75000.00, 2025-02-01);	(Mobile...) 6 rows affected Records: 6 Duplicates: 0 Warnings: 0	0.037 sec
5	09:22:49	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
6	09:23:20	SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
7	09:23:43	SELECT AVG(Price) AS Average_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
8	09:24:04	SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec

Object Info Session

77°F Sunny 9:24 AM 4/4/2025

-- Count the Total Number of Sales Records

SELECT COUNT(*) AS Total_Sales FROM Sales;



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1

```

32 • SELECT AVG(Price) AS Average_Price FROM Sales;
33
34 -- Minimum and Maximum Price of a Product Sold
35 • SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales;
36
37 -- Count the Total Number of Sales Records
38 • SELECT COUNT(*) AS Total_Sales FROM Sales;

```

Result Grid | Filter Rows: Export | Wrap Cell Contents | Result Grid | Form Editor | Field Types

Total_Sales
6

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas Result 5

Information Output

Action Output

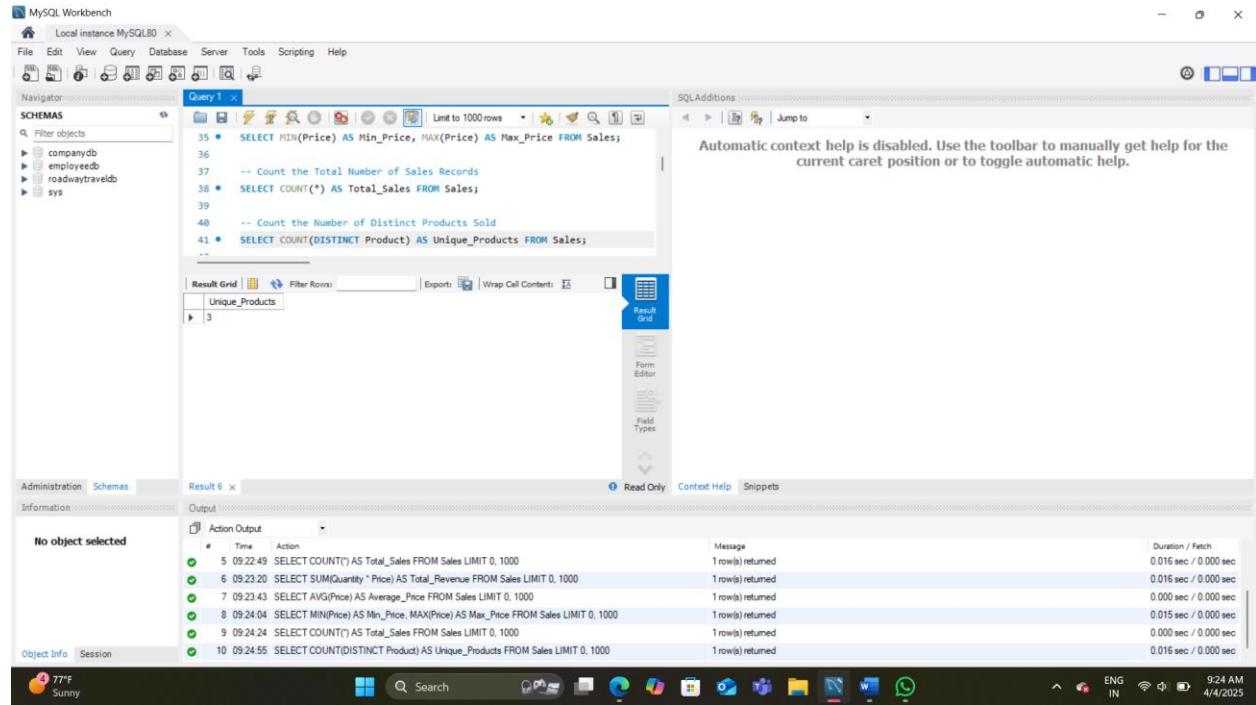
#	Time	Action	Message	Duration / Fetch
4	09:22:46	INSERT INTO Sales (Product, Quantity, Price, SaleDate) VALUES ('Laptop', 2, 75000.00, 2025-02-01);	(Mobile...) 6 rows affected Records: 6 Duplicates: 0 Warnings: 0	0.032 sec
5	09:22:49	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
6	09:23:20	SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
7	09:23:43	SELECT AVG(Price) AS Average_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
8	09:24:04	SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
9	09:24:24	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

77°F Sunny 9:24 AM 4/4/2025

-- Count the Number of Distinct Products Sold

```
SELECT COUNT(DISTINCT Product) AS Unique_Products FROM Sales;
```



The screenshot shows the MySQL Workbench interface. The 'Query 1' tab contains the following SQL code:

```
35 •  SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales;
36
37 -- Count the Total Number of Sales Records
38 •  SELECT COUNT(*) AS Total_Sales FROM Sales;
39
40 -- Count the Number of Distinct Products Sold
41 •  SELECT COUNT(DISTINCT Product) AS Unique_Products FROM Sales;
42
```

The 'Result Grid' shows the output of the last query:

Unique_Products
3

The 'Result 6' tab shows the execution history:

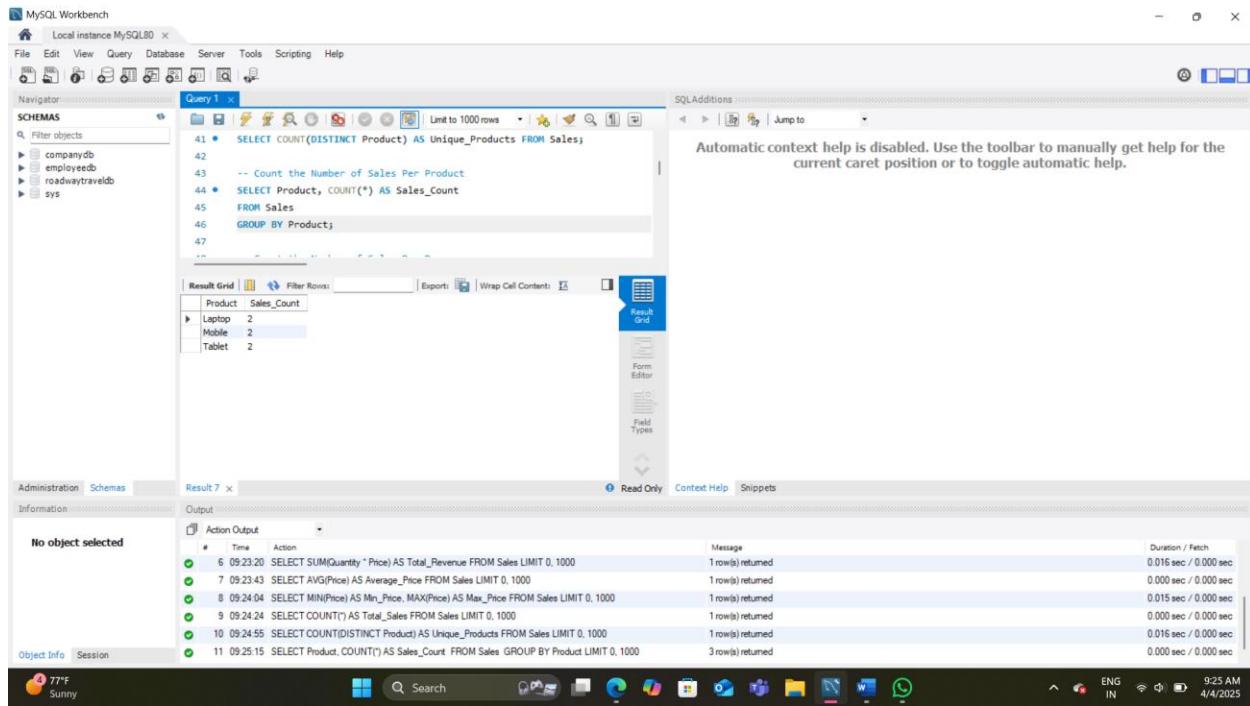
Action	Time	Message	Duration / Fetch
5	09:22:49	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	1row(s) returned 0.016 sec / 0.000 sec
6	09:23:20	SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1row(s) returned 0.016 sec / 0.000 sec
7	09:23:43	SELECT AVG(Price) AS Average_Price FROM Sales LIMIT 0, 1000	1row(s) returned 0.000 sec / 0.000 sec
8	09:24:04	SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales LIMIT 0, 1000	1row(s) returned 0.015 sec / 0.000 sec
9	09:24:24	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	1row(s) returned 0.000 sec / 0.000 sec
10	09:24:55	SELECT COUNT(DISTINCT Product) AS Unique_Products FROM Sales LIMIT 0, 1000	1row(s) returned 0.016 sec / 0.000 sec

-- Count the Number of Sales Per Product

```
SELECT Product, COUNT(*) AS Sales_Count
```

```
FROM Sales
```

```
GROUP BY Product;
```



MySQL Workbench - Local instance MySQL80

Query 1

```

41 • SELECT COUNT(DISTINCT Product) AS Unique_Products FROM Sales;
42
43 -- Count the Number of Sales Per Product
44 • SELECT Product, COUNT(*) AS Sales_Count
45 FROM Sales
46 GROUP BY Product;
47

```

Result Grid

Product	Sales_Count
Laptop	2
Mobile	2
Tablet	2

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas

Result 7

Action Output

Action	Time	Message	Duration / Fetch
6 09:23:20	SELECT SUM(Quality * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1row(s) returned	0.016 sec / 0.000 sec
7 09:23:43	SELECT AVG(Price) AS Average_Price FROM Sales LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
8 09:24:04	SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales LIMIT 0, 1000	1row(s) returned	0.015 sec / 0.000 sec
9 09:24:24	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
10 09:24:55	SELECT COUNT(DISTINCT Product) AS Unique_Products FROM Sales LIMIT 0, 1000	1row(s) returned	0.016 sec / 0.000 sec
11 09:25:15	SELECT Product, COUNT(*) AS Sales_Count FROM Sales GROUP BY Product LIMIT 0, 1000	3row(s) returned	0.000 sec / 0.000 sec

Object Info Session

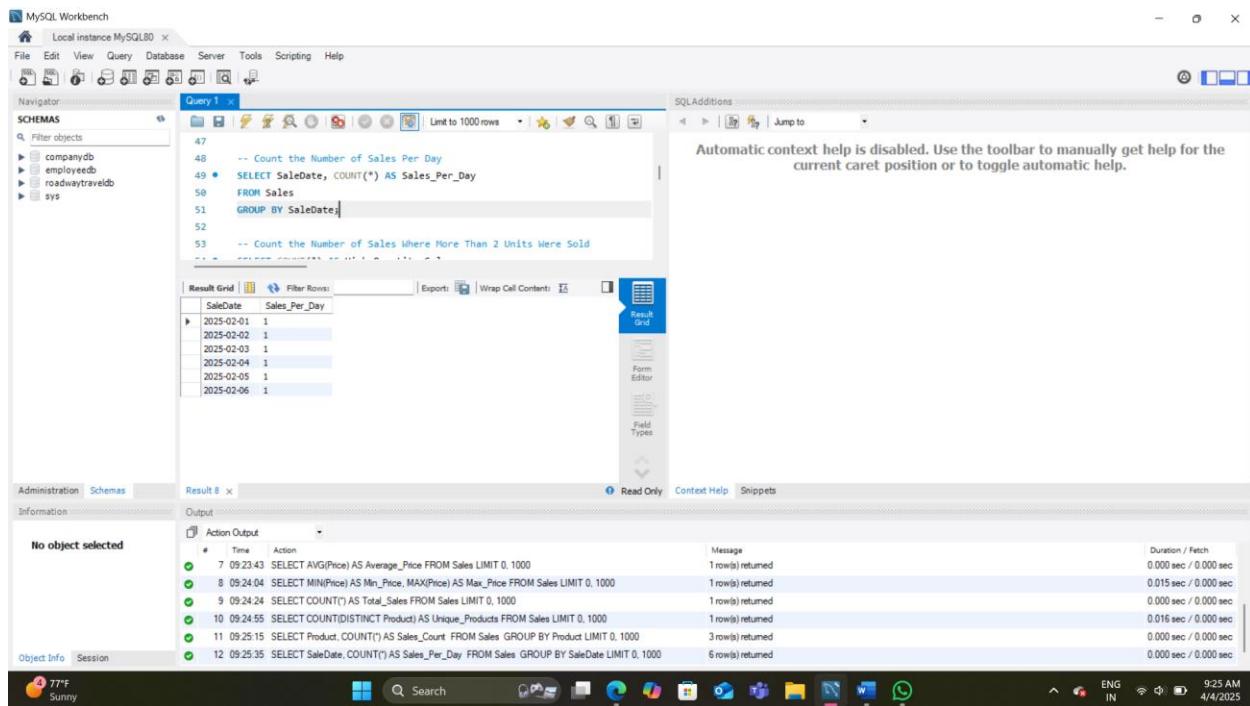
9:25 AM 4/4/2025

-- Count the Number of Sales Per Day

SELECT SaleDate, COUNT(*) AS Sales_Per_Day

FROM Sales

GROUP BY SaleDate;



MySQL Workbench - Local instance MySQL80

Query 1

```

47
48 -- Count the Number of Sales Per Day
49 • SELECT SaleDate, COUNT(*) AS Sales_Per_Day
50 FROM Sales
51 GROUP BY SaleDate;
52
53 -- Count the Number of Sales Where More Than 2 Units Were Sold

```

Result Grid

SaleDate	Sales_Per_Day
2025-02-01	1
2025-02-02	1
2025-02-03	1
2025-02-04	1
2025-02-05	1
2025-02-06	1

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas

Result 8

Action Output

Action	Time	Message	Duration / Fetch
7 09:23:43	SELECT AVG(Price) AS Average_Price FROM Sales LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
8 09:24:04	SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales LIMIT 0, 1000	1row(s) returned	0.015 sec / 0.000 sec
9 09:24:24	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
10 09:24:55	SELECT COUNT(DISTINCT Product) AS Unique_Products FROM Sales LIMIT 0, 1000	1row(s) returned	0.016 sec / 0.000 sec
11 09:25:15	SELECT Product, COUNT(*) AS Sales_Count FROM Sales GROUP BY Product LIMIT 0, 1000	3row(s) returned	0.000 sec / 0.000 sec
12 09:25:35	SELECT SaleDate, COUNT(*) AS Sales_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6row(s) returned	0.000 sec / 0.000 sec

Object Info Session

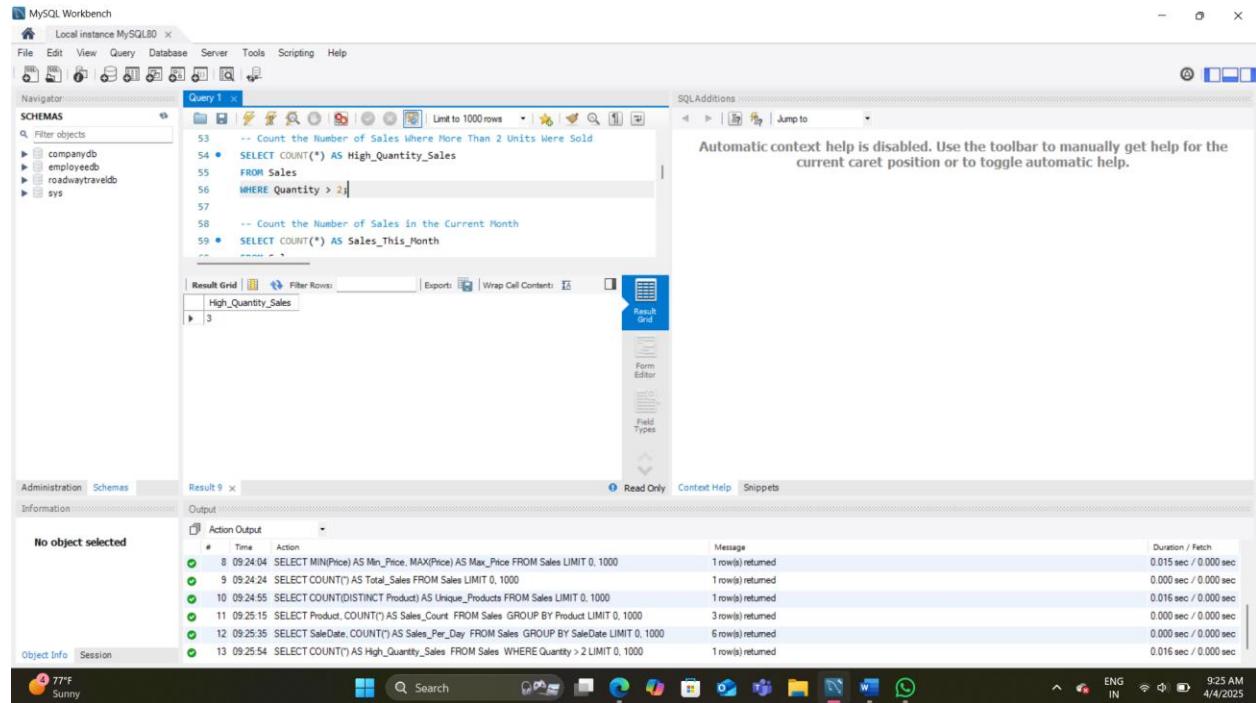
9:25 AM 4/4/2025

-- Count the Number of Sales Where More Than 2 Units Were Sold

```
SELECT COUNT(*) AS High_Quantity_Sales
```

```
FROM Sales
```

```
WHERE Quantity > 2;
```



The screenshot shows the MySQL Workbench interface. The 'Query 1' tab contains the following SQL code:

```
-- Count the Number of Sales Where More Than 2 Units Were Sold
SELECT COUNT(*) AS High_Quantity_Sales
FROM Sales
WHERE Quantity > 2;
```

The 'Result Grid' shows the output:

High_Quantity_Sales
3

The 'Information' tab shows the 'Action Output' log:

Action	Time	Message	Duration / Fetch
8 09:24:04	SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
9 09:24:24	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
10 09:24:55	SELECT COUNT(DISTINCT Product) AS Unique_Products FROM Sales LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
11 09:25:15	SELECT Product, COUNT(*) AS Sales_Count FROM Sales GROUP BY Product LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
12 09:25:35	SELECT SaleDate, COUNT(*) AS Sales_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
13 09:25:54	SELECT COUNT(*) AS High_Quantity_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

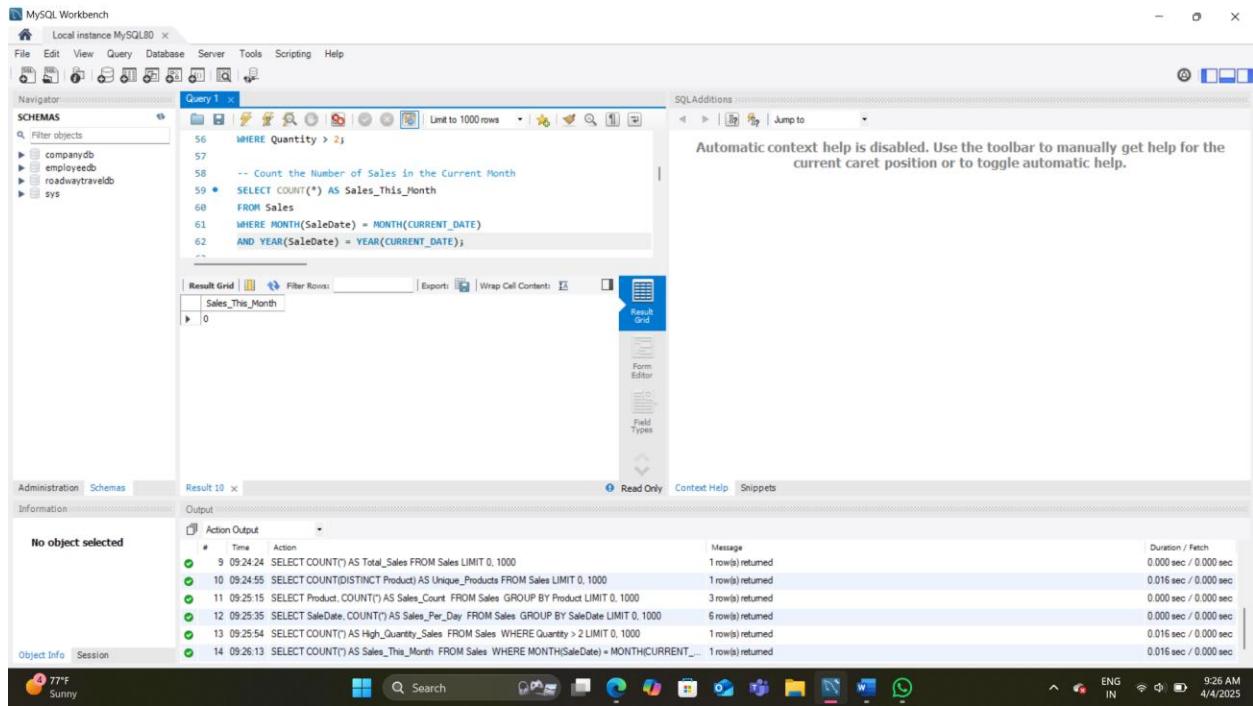
-- Count the Number of Sales in the Current Month

```
SELECT COUNT(*) AS Sales_This_Month
```

```
FROM Sales
```

```
WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
```

```
AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
```



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1:

```

56 WHERE Quantity > 2;
57
58 -- Count the Number of Sales in the Current Month
59 • SELECT COUNT(*) AS Sales_This_Month
60 FROM Sales
61 WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
62 AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
63

```

Result Grid | Filter Rows: Export: Wrap Cell Contents: Result Grid

Result 10:

Sales_This_Month
0

Action Output:

#	Time	Action	Message	Duration / Fetch
9	09-24-24	SELECT COUNT(*) AS Total_Sales FROM Sales LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
10	09-24-55	SELECT COUNT(DISTINCT Product) AS Unique_Products FROM Sales LIMIT 0, 1000	1row(s) returned	0.016 sec / 0.000 sec
11	09-25-15	SELECT Product.COUNT() AS Sales_Count FROM Sales GROUP BY Product LIMIT 0, 1000	3row(s) returned	0.000 sec / 0.000 sec
12	09-25-35	SELECT SaleDate.COUNT() AS Sales_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6row(s) returned	0.000 sec / 0.000 sec
13	09-25-54	SELECT COUNT(*) AS High_Quantity_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1row(s) returned	0.016 sec / 0.000 sec
14	09-26-13	SELECT COUNT(*) AS Sales_This_Month FROM Sales WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE) AND YEAR(SaleDate) = YEAR(CURRENT_DATE);	1row(s) returned	0.016 sec / 0.000 sec

Object Info Session

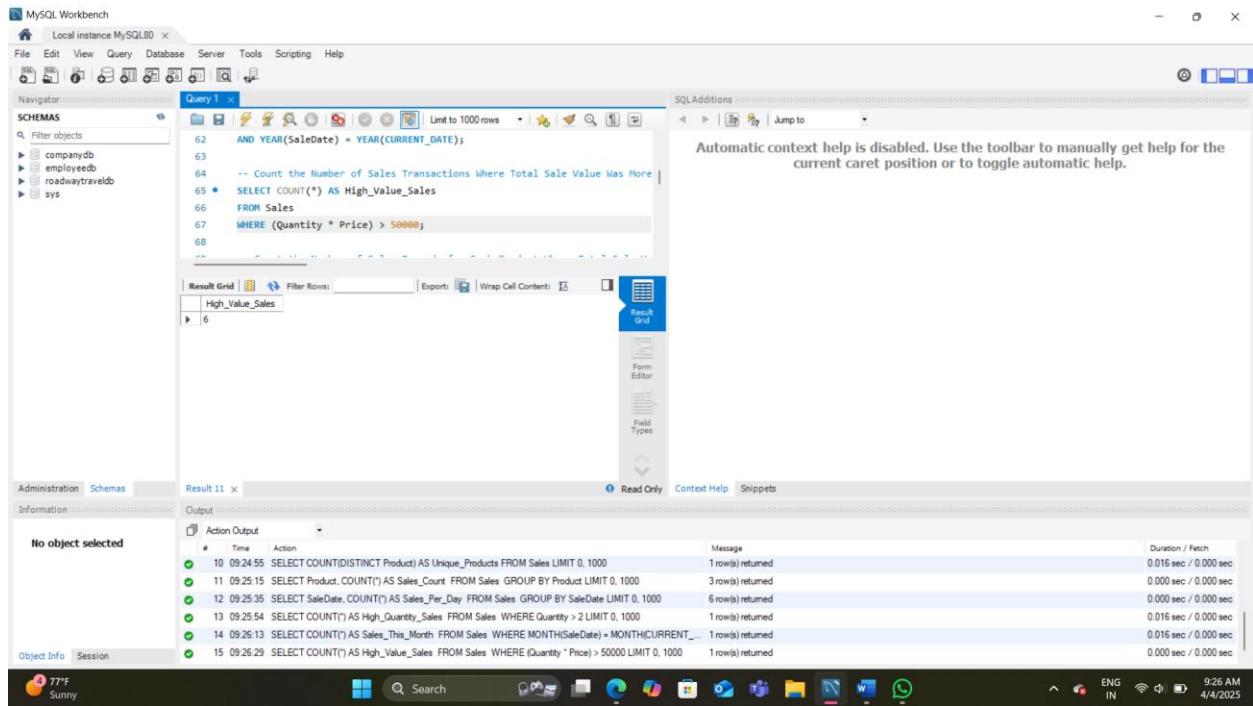
77°F Sunny 9:26 AM 4/4/2025

-- Count the Number of Sales Transactions Where Total Sale Value Was More Than ₹50,000

SELECT COUNT(*) AS High_Value_Sales

FROM Sales

WHERE (Quantity * Price) > 50000;



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1:

```

62 AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
63
64 -- Count the Number of Sales Transactions Where Total Sale Value Was More
65 • SELECT COUNT(*) AS High_Value_Sales
66 FROM Sales
67 WHERE (Quantity * Price) > 50000;
68

```

Result Grid | Filter Rows: Export: Wrap Cell Contents: Result Grid

Result 11:

High_Value_Sales
6

Action Output:

#	Time	Action	Message	Duration / Fetch
10	09-24-55	SELECT COUNT(DISTINCT Product) AS Unique_Products FROM Sales LIMIT 0, 1000	1row(s) returned	0.016 sec / 0.000 sec
11	09-25-15	SELECT Product.COUNT() AS Sales_Count FROM Sales GROUP BY Product LIMIT 0, 1000	3row(s) returned	0.000 sec / 0.000 sec
12	09-25-35	SELECT SaleDate.COUNT() AS Sales_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6row(s) returned	0.000 sec / 0.000 sec
13	09-25-54	SELECT COUNT(*) AS High_Quantity_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1row(s) returned	0.016 sec / 0.000 sec
14	09-26-13	SELECT COUNT(*) AS Sales_This_Month FROM Sales WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE) AND YEAR(SaleDate) = YEAR(CURRENT_DATE);	1row(s) returned	0.016 sec / 0.000 sec
15	09-26-29	SELECT COUNT(*) AS High_Value_Sales FROM Sales WHERE (Quantity * Price) > 50000 LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec

Object Info Session

77°F Sunny 9:26 AM 4/4/2025

-- Count the Number of Sales Records for Each Product Where Total Sale Value Is Greater Than ₹40,000

```
SELECT Product, COUNT(*) AS High_Value_Transactions
```

```
FROM Sales
```

```
WHERE (Quantity * Price) > 40000
```

```
GROUP BY Product;
```

The screenshot shows the MySQL Workbench interface. In the top query editor, a SQL script is run:

```
68 -- Count the Number of Sales Records for Each Product Where Total Sale Value Is Greater Than ₹40,000
69
70 • SELECT Product, COUNT(*) AS High_Value_Transactions
71 FROM Sales
72 WHERE (Quantity * Price) > 40000
73 GROUP BY Product;
74
```

The results are displayed in a table:

Product	High_Value_Transactions
Laptop	2
Mobile	2
Tablet	2

Below the results, the 'Information' pane shows the 'Action Output' history:

Action	Time	Message	Duration / Fetch
11 09:25:15	SELECT Product, COUNT(*) AS Sales_Count FROM Sales GROUP BY Product LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
12 09:25:35	SELECT SaleDate, COUNT(*) AS Sales_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
13 09:25:54	SELECT COUNT(*) AS High_Value_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
14 09:26:13	SELECT COUNT(*) AS Sales_Per_Month FROM Sales WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)	1 row(s) returned	0.016 sec / 0.000 sec
15 09:26:29	SELECT COUNT(*) AS High_Value_Sales FROM Sales WHERE (Quantity * Price) > 50000 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
16 09:26:46	SELECT Product, COUNT(*) AS High_Value_Transactions FROM Sales WHERE (Quantity * Price) > 40000 ...	3 row(s) returned	0.000 sec / 0.000 sec

-- Count the Number of Sales Made After a Specific Date (e.g., Feb 3, 2025)

```
SELECT COUNT(*) AS Sales_After_Date
```

```
FROM Sales
```

```
WHERE SaleDate > '2025-02-03';
```

MySQL Workbench - Local instance MySQL80

Query 1:

```

74  -- Count the Number of Sales Made After a Specific Date (e.g., Feb 3, 2025)
75  SELECT COUNT(*) AS Sales_After_Date
76  FROM Sales
77  WHERE SaleDate > '2025-02-03'
78
79
80  -- Sum of Total Revenue Generated

```

Result Grid:

Sales_After_Date
3

Output:

Action	Time	Action	Message	Duration / Fetch
12	09:25:35	SELECT SaleDate, COUNT(*) AS Sales_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
13	09:25:54	SELECT COUNT(*) AS High_Quantity_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
14	09:26:13	SELECT COUNT(*) AS Sales_This_Month FROM Sales WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)	1 row(s) returned	0.016 sec / 0.000 sec
15	09:26:29	SELECT COUNT(*) AS High_Value_Sales FROM Sales WHERE (Quantity * Price) > 50000 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
16	09:26:46	SELECT Product, COUNT(*) AS High_Value_Transactions FROM Sales WHERE (Quantity * Price) > 40000 ...	3 row(s) returned	0.000 sec / 0.000 sec
17	09:27:09	SELECT COUNT(*) AS Sales_After_Date FROM Sales WHERE SaleDate > '2025-02-03' LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec

9:27 AM 4/4/2025

-- Sum of Total Revenue Generated

SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales;

MySQL Workbench - Local instance MySQL80

Query 1:

```

77  FROM Sales
78  WHERE SaleDate > '2025-02-03'
79
80  -- Sum of Total Revenue Generated
81  SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales;
82
83  -- Sum of Total Quantity of Products Sold

```

Result Grid:

Total_Revenue
570000.00

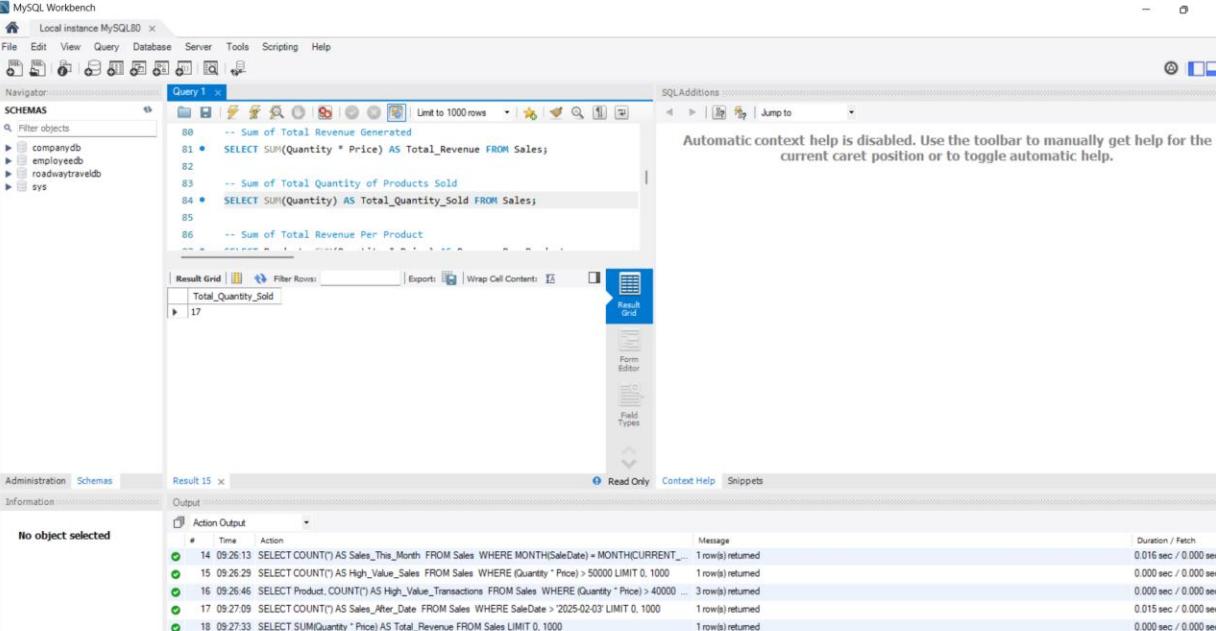
Output:

Action	Time	Action	Message	Duration / Fetch
13	09:25:54	SELECT COUNT(*) AS High_Quantity_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
14	09:26:13	SELECT COUNT(*) AS Sales_This_Month FROM Sales WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)	1 row(s) returned	0.016 sec / 0.000 sec
15	09:26:29	SELECT COUNT(*) AS High_Value_Sales FROM Sales WHERE (Quantity * Price) > 50000 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
16	09:26:46	SELECT Product, COUNT(*) AS High_Value_Transactions FROM Sales WHERE (Quantity * Price) > 40000 ...	3 row(s) returned	0.000 sec / 0.000 sec
17	09:27:09	SELECT COUNT(*) AS Sales_After_Date FROM Sales WHERE SaleDate > '2025-02-03' LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
18	09:27:33	SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

9:27 AM 4/4/2025

-- Sum of Total Quantity of Products Sold

SELECT SUM(Quantity) AS Total_Quantity_Sold FROM Sales;



MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

companydb
employeedb
roadwaytraveldb
sys

Query 1

80 -- Sum of Total Revenue Generated
81 • SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales;
82
83 -- Sum of Total Quantity of Products Sold
84 • SELECT SUM(Quantity) AS Total_Quantity_Sold FROM Sales;
85
86 -- Sum of Total Revenue Per Product

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Total_Quantity_Sold

17

Result 15

Output

No object selected

Action Output

#	Time	Action	Message	Duration / Fetch
14	09:26:13	SELECT COUNT(*) AS Sales_This_Month FROM Sales WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)	1 row(s) returned	0.016 sec / 0.000 sec
15	09:26:29	SELECT COUNT(*) AS High_Value_Sales FROM Sales WHERE (Quantity * Price) > 50000 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
16	09:26:46	SELECT Product, COUNT(*) AS High_Value_Transactions FROM Sales WHERE (Quantity * Price) > 40000	3 row(s) returned	0.000 sec / 0.000 sec
17	09:27:09	SELECT COUNT(*) AS Sales_After_Date FROM Sales WHERE SaleDate > '2025-02-03' LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
18	09:27:33	SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
19	09:27:53	SELECT SUM(Quantity) AS Total_Quantity_Sold FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

77°F Sunny

ENG IN

9:27 AM 4/4/2024

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

-- Sum of Total Revenue Per Product

```
SELECT Product, SUM(Quantity * Price) AS Revenue_Per_Product
```

FROM Sales

GROUP BY Product;

The screenshot shows the MySQL Workbench interface. The top navigation bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar displays the Navigator and SCHEMAS (companydb, employeedb, roadwaytraveldb, sys). The main area contains a Query Editor with the following SQL code:

```
83 -- Sum of Total Quantity of Products Sold
84 • SELECT SUM(Quantity) AS Total_Quantity_Sold FROM Sales;
85
86 -- Sum of Total Revenue Per Product
87 • SELECT Product, SUM(Quantity * Price) AS Revenue_Per_Product
88 FROM Sales
89 GROUP BY Product;
```

The Result Grid shows the output of the second query:

Product	Revenue_Per_Product
Laptop	228000.00
Mobile	188000.00
Tablet	154000.00

The bottom section shows a history of recent actions:

Action	Time	Message	Duration / Fetch
15	09:26:29	SELECT COUNT(*) AS High_Value_Sales FROM Sales WHERE (Quantity * Price) > 50000 LIMIT 0, 1000	1 row(s) returned 0.000 sec / 0.000 sec
16	09:26:46	SELECT Product, COUNT(*) AS High_Value_Transactions FROM Sales WHERE (Quantity * Price) > 40000	3 row(s) returned 0.000 sec / 0.000 sec
17	09:27:09	SELECT COUNT(*) AS Sales_After_Date FROM Sales WHERE SaleDate > 2025-02-03 LIMIT 0, 1000	1 row(s) returned 0.015 sec / 0.000 sec
18	09:27:33	SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1 row(s) returned 0.000 sec / 0.000 sec
19	09:27:53	SELECT SUM(Quantity) AS Total_Quantity_Sold FROM Sales LIMIT 0, 1000	1 row(s) returned 0.000 sec / 0.000 sec
20	09:28:14	SELECT Product, SUM(Quantity * Price) AS Revenue_Per_Product FROM Sales GROUP BY Product LIMIT 0, 3	3 row(s) returned 0.000 sec / 0.000 sec

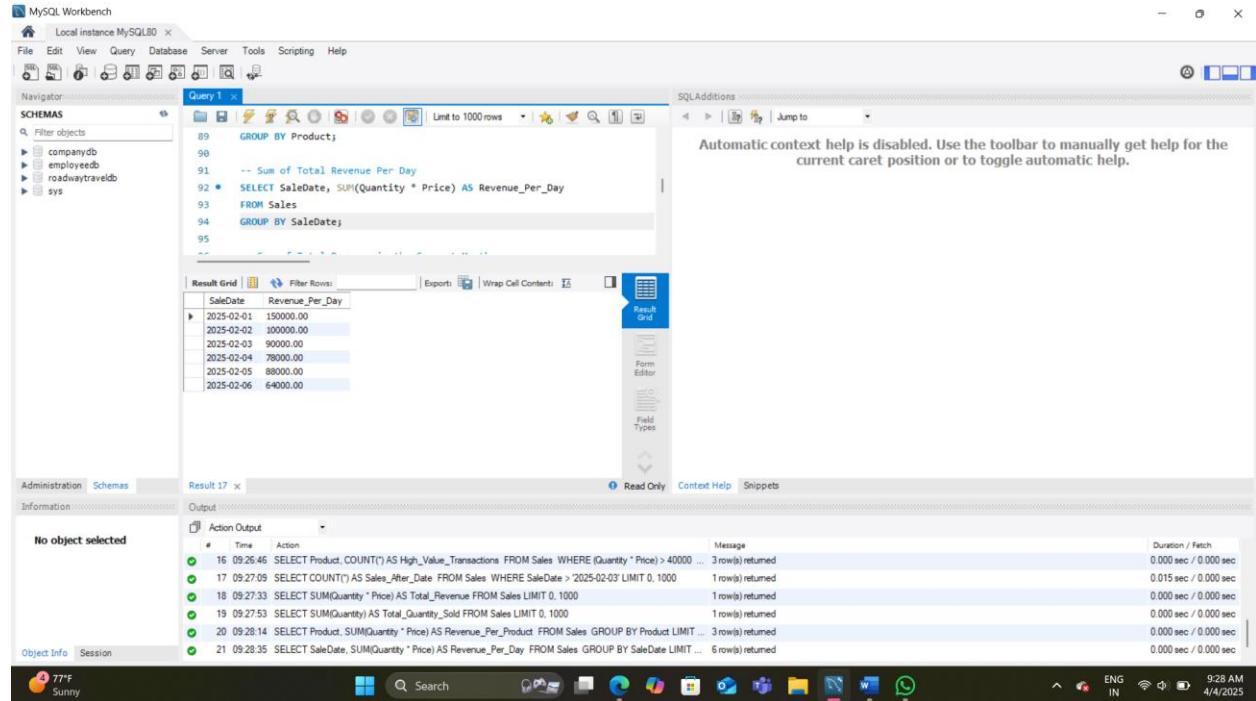
The status bar at the bottom shows the system tray with icons for battery, signal, and date/time (9:28 AM, 4/4/2022).

-- Sum of Total Revenue Per Day

```
SELECT SaleDate, SUM(Quantity * Price) AS Revenue_Per_Day
```

```
FROM Sales
```

```
GROUP BY SaleDate;
```



The screenshot shows the MySQL Workbench interface. The 'Query 1' tab contains the following SQL code:

```
89 GROUP BY Product;
90
91 -- Sum of Total Revenue Per Day
92 • SELECT SaleDate, SUM(Quantity * Price) AS Revenue_Per_Day
93 FROM Sales
94 GROUP BY SaleDate;
95
```

The 'Result Grid' shows the following data:

SaleDate	Revenue_Per_Day
2025-02-01	150000.00
2025-02-02	300000.00
2025-02-03	90000.00
2025-02-04	78000.00
2025-02-05	88000.00
2025-02-06	64000.00

The 'Information' tab shows the following session history:

Action	Time	Action	Message	Duration / Fetch
16	09:26:45	SELECT Product, COUNT() AS High_Value_Transactions FROM Sales WHERE (Quantity * Price) > 40000 ...	3 row(s) returned	0.000 sec / 0.000 sec
17	09:27:09	SELECT COUNT(*) AS Sales_After_Day FROM Sales WHERE SaleDate > '2025-02-03' LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
18	09:27:33	SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
19	09:27:53	SELECT SUM(Quantity) AS Total_Quantity_Sold FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
20	09:28:14	SELECT Product, SUM(Quantity * Price) AS Revenue_Per_Product FROM Sales GROUP BY Product LIMIT ...	3 row(s) returned	0.000 sec / 0.000 sec
21	09:28:35	SELECT SaleDate, SUM(Quantity * Price) AS Revenue_Per_Day FROM Sales GROUP BY SaleDate LIMIT ...	6 row(s) returned	0.000 sec / 0.000 sec

-- Sum of Total Revenue in the Current Month

```
SELECT SUM(Quantity * Price) AS Revenue_This_Month
```

```
FROM Sales
```

```
WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
```

```
AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
```

The screenshot shows the MySQL Workbench interface. The top navigation bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar displays the Navigator, SCHEMAS (with companydb, employeedb, roadwaystraveldb, and sys selected), and a toolbar with various icons. The main area contains a Query Editor with the following SQL code:

```
95
96  -- Sum of Total Revenue in the Current Month
97  •  SELECT SUM(Quantity * Price) AS Revenue_This_Month
98  FROM Sales
99  WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
100 AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
101
```

The Result Grid shows a single row with the value 'Revenue_This_Month'. The bottom section shows the History tab with the following log entries:

#	Time	Action	Message	Duration / Fetch
17	09:27:09	SELECT COUNT(*) AS Sales_Hter_Date FROM Sales WHERE SaleDate > '2025-02-03' LIMIT 0, 1000	1 rows(s) returned	0.015 sec / 0.000 sec
18	09:27:33	SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1 rows(s) returned	0.000 sec / 0.000 sec
19	09:27:53	SELECT SUM(Quantity) AS Total_Quantity_Sold FROM Sales LIMIT 0, 1000	1 rows(s) returned	0.000 sec / 0.000 sec
20	09:28:14	SELECT Product, SUM(Quantity * Price) AS Revenue_Per_Product FROM Sales GROUP BY Product LIMIT 0, 1000	3 rows(s) returned	0.000 sec / 0.000 sec
21	09:28:35	SELECT SaleDate, SUM(Quantity * Price) AS Revenue_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
22	09:29:00	SELECT SUM(Quantity * Price) AS Revenue_The_Month FROM Sales WHERE MONTH(SaleDate) = MON...	1 rows(s) returned	0.000 sec / 0.000 sec

The bottom status bar shows the system tray with icons for battery, signal, and network, and the date/time '4/4/2023 9:29 AM'.

-- Sum of Revenue for Sales Where Quantity Sold Is Greater Than 2

```
SELECT SUM(Quantity * Price) AS High_Quantity_Revenue
```

FROM Sales

WHERE Quantity > 2;

The screenshot shows the MySQL Workbench interface. The top window is titled 'Local instance MySQL80' and contains a query editor with the following SQL code:

```
101
102  -- Sum of Revenue for Sales Where Quantity Sold Is Greater Than 2
103  SELECT SUM(Quantity * Price) AS High_Quantity_Revenue
104  FROM Sales
105  WHERE Quantity > 2;
106
107  -- Sum of Total Revenue Generated After a Specific Date (e.g., Feb 3, 202
108
109  SELECT SUM(Quantity * Price) AS Total_Revenue
110  FROM Sales
111  WHERE SaleDate > '2023-02-03' AND SaleDate < '2023-02-04';
```

The results grid shows a single row with the value 278000.00. The bottom window is titled 'Result 19' and shows a history of executed queries:

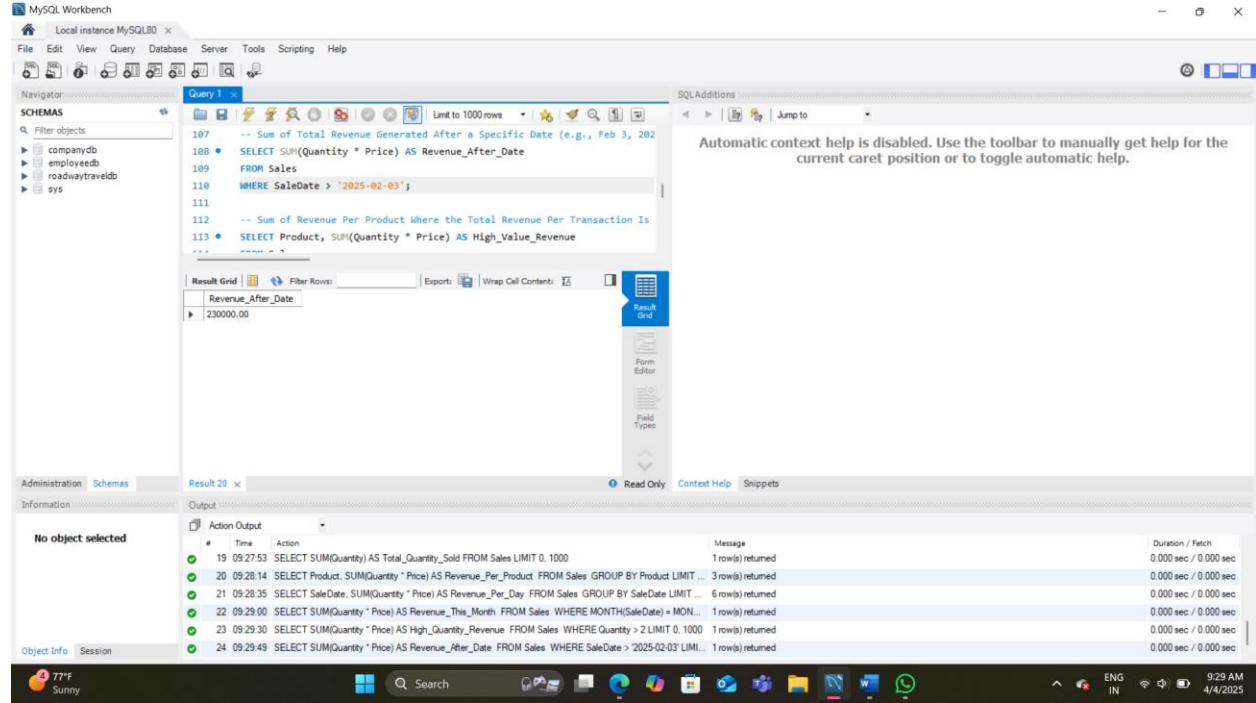
#	Time	Action	Message	Duration / Fetch
18	09:27:33	SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
19	09:27:53	SELECT SUM(Quantity) AS Total_Quantity_Sold FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
20	09:28:14	SELECT Product, SUM(Quantity * Price) AS Revenue_Per_Product, FROM Sales GROUP BY Product LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
21	09:28:35	SELECT SaleDate, SUM(Quantity * Price) AS Revenue_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
22	09:29:00	SELECT SUM(Quantity * Price) AS Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = MONTH(CURDATE())	1 row(s) returned	0.000 sec / 0.000 sec
23	09:29:30	SELECT SUM(Quantity * Price) AS High_Quantity_Revenue FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

-- Sum of Total Revenue Generated After a Specific Date (e.g., Feb 3, 2025)

```
SELECT SUM(Quantity * Price) AS Revenue_After_Date
```

```
FROM Sales
```

```
WHERE SaleDate > '2025-02-03';
```



The screenshot shows the MySQL Workbench interface. The 'Query 1' tab contains the following SQL code:

```
-- Sum of Total Revenue Generated After a Specific Date (e.g., Feb 3, 2025)
SELECT SUM(Quantity * Price) AS Revenue_After_Date
FROM Sales
WHERE SaleDate > '2025-02-03';

-- Sum of Revenue Per Product Where the Total Revenue Per Transaction Is
SELECT Product, SUM(Quantity * Price) AS High_Value_Revenue
```

The 'Result Grid' shows the output of the first query:

Revenue_After_Date
230000.00

The 'Information' tab shows the execution history:

Action	Time	Action	Message	Duration / Fetch
19	09:27:53	SELECT SUM(Quantity) AS Total_Quantity_Sold FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
20	09:28:14	SELECT Product, SUM(Quantity * Price) AS Revenue_Per_Product FROM Sales GROUP BY Product LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
21	09:28:35	SELECT SaleDate, SUM(Quantity * Price) AS Revenue_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
22	09:29:00	SELECT SUM(Quantity * Price) AS Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = MON...	1 row(s) returned	0.000 sec / 0.000 sec
23	09:29:30	SELECT SUM(Quantity * Price) AS High_Quantity_Revenue FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
24	09:29:49	SELECT SUM(Quantity * Price) AS Revenue_After_Date FROM Sales WHERE SaleDate > '2025-02-03' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

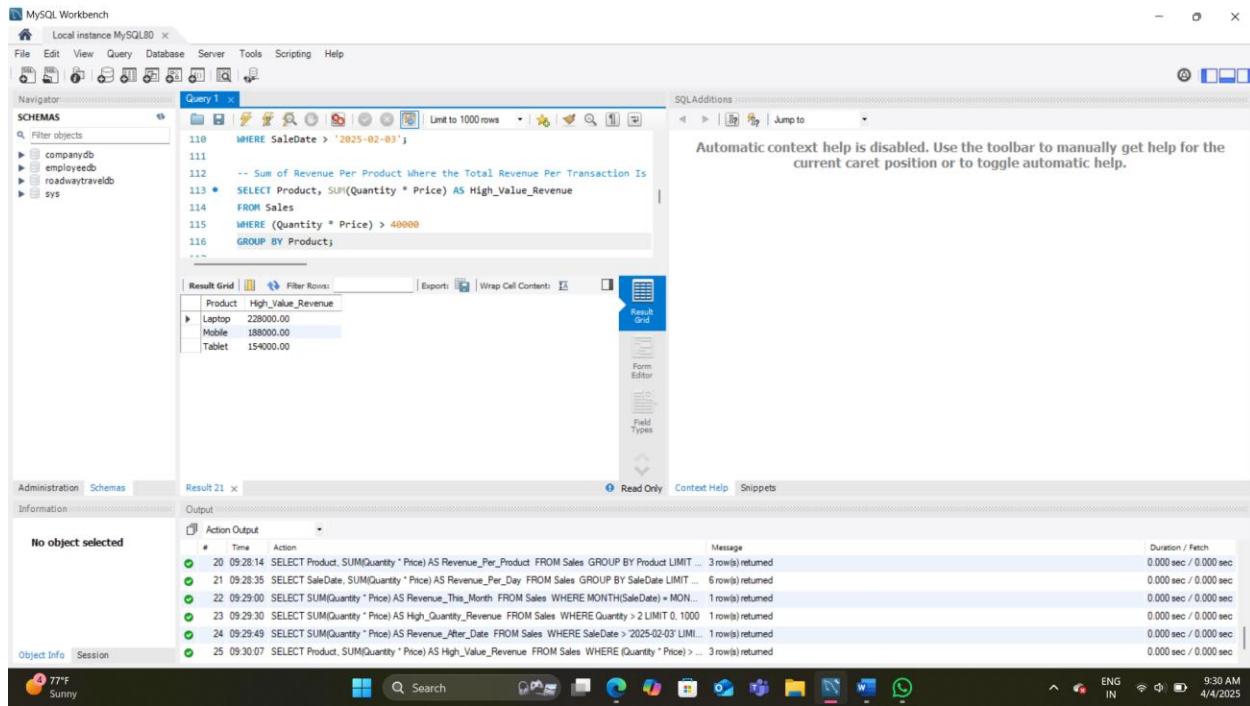
-- Sum of Revenue Per Product Where the Total Revenue Per Transaction Is Greater Than ₹40,000

```
SELECT Product, SUM(Quantity * Price) AS High_Value_Revenue
```

```
FROM Sales
```

```
WHERE (Quantity * Price) > 40000
```

```
GROUP BY Product;
```



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1

```

110 WHERE SaleDate > '2025-02-03';
111
112 -- Sum of Revenue Per Product where the Total Revenue Per Transaction Is
113 *   SELECT Product, SUM(Quantity * Price) AS High_Value_Revenue
114   FROM Sales
115   WHERE (Quantity * Price) > 40000
116   GROUP BY Product;
117

```

Result Grid | Filter Rows: Export | Wrap Cell Content: |

Product	High_Value_Revenue
Laptop	228000.00
Mobile	188000.00
Tablet	154000.00

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas Result 21 x Read Only Context Help Snippets

Information Output

Action Output

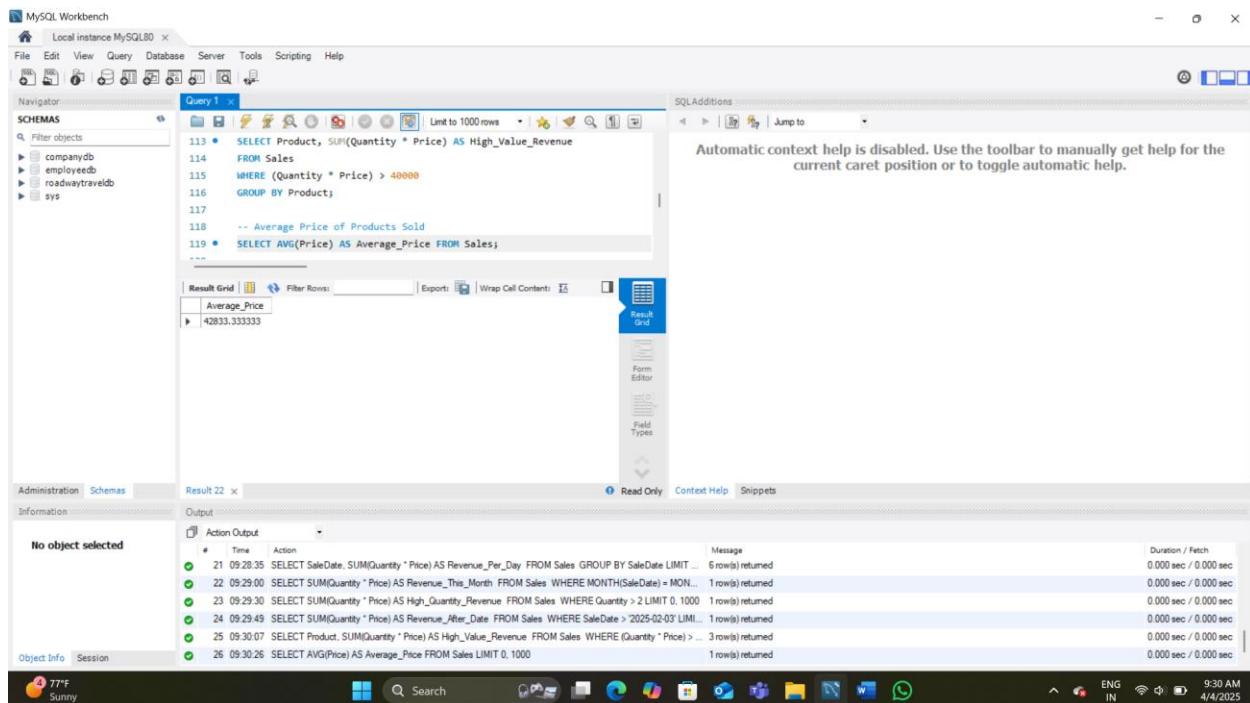
#	Time	Action	Message	Duration / Fetch
20	09:28:14	SELECT Product, SUM(Quantity * Price) AS Revenue_Per_Product FROM Sales GROUP BY Product LIMIT ...	3 row(s) returned	0.000 sec / 0.000 sec
21	09:28:35	SELECT SaleDate, SUM(Quantity * Price) AS Revenue_Per_Day FROM Sales GROUP BY SaleDate LIMIT ...	6 row(s) returned	0.000 sec / 0.000 sec
22	09:29:00	SELECT SUM(Quantity * Price) AS Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = MON...	1 row(s) returned	0.000 sec / 0.000 sec
23	09:29:30	SELECT SUM(Quantity * Price) AS High_Quantity_Revenue FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
24	09:29:49	SELECT SUM(Quantity * Price) AS Revenue_After_Date FROM Sales WHERE SaleDate > '2025-02-03' LIMIT ...	1 row(s) returned	0.000 sec / 0.000 sec
25	09:30:07	SELECT Product, SUM(Quantity * Price) AS High_Value_Revenue FROM Sales WHERE (Quantity * Price) > ...	3 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

77°F Sunny 9:30 AM 4/4/2025

-- Average Price of Products Sold

SELECT AVG(Price) AS Average_Price FROM Sales;



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1

```

113 *   SELECT Product, SUM(Quantity * Price) AS High_Value_Revenue
114   FROM Sales
115   WHERE (Quantity * Price) > 40000
116   GROUP BY Product;
117
118   -- Average Price of Products Sold
119 *   SELECT AVG(Price) AS Average_Price FROM Sales;
120

```

Result Grid | Filter Rows: Export | Wrap Cell Content: |

Average_Price
42833.333333

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas Result 22 x Read Only Context Help Snippets

Information Output

Action Output

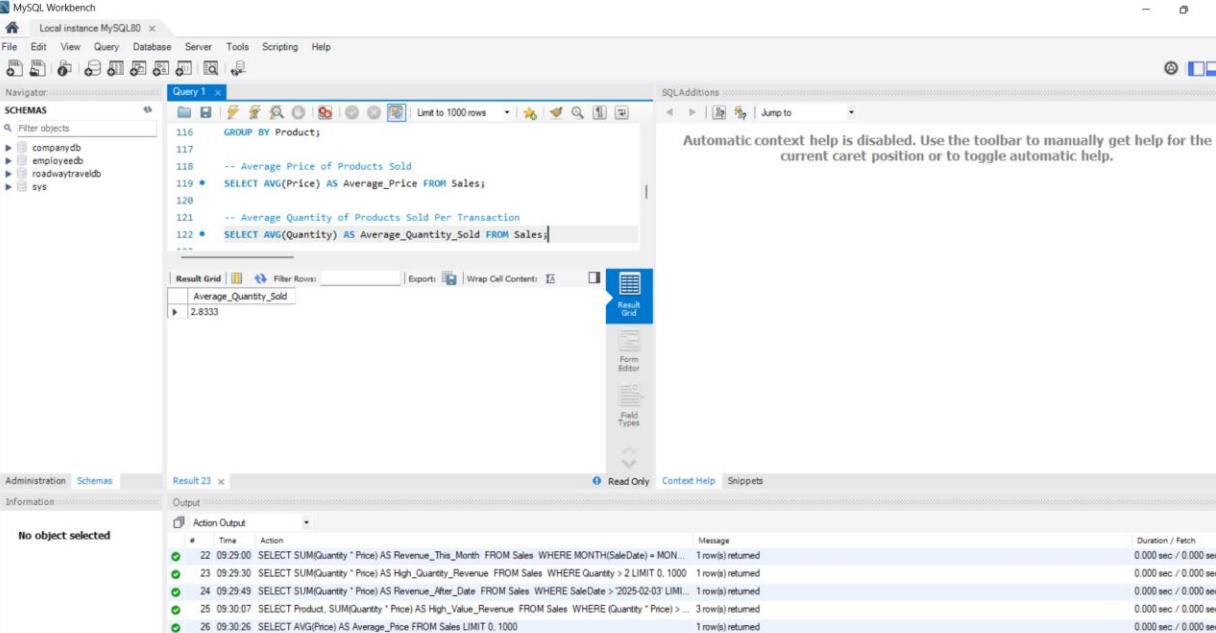
#	Time	Action	Message	Duration / Fetch
21	09:28:35	SELECT SaleDate, SUM(Quantity * Price) AS Revenue_Per_Day FROM Sales GROUP BY SaleDate LIMIT ...	6 row(s) returned	0.000 sec / 0.000 sec
22	09:29:00	SELECT SUM(Quantity * Price) AS Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = MON...	1 row(s) returned	0.000 sec / 0.000 sec
23	09:29:30	SELECT SUM(Quantity * Price) AS High_Quantity_Revenue FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
24	09:29:49	SELECT SUM(Quantity * Price) AS Revenue_After_Date FROM Sales WHERE SaleDate > '2025-02-03' LIMIT ...	1 row(s) returned	0.000 sec / 0.000 sec
25	09:30:07	SELECT Product, SUM(Quantity * Price) AS High_Value_Revenue FROM Sales WHERE (Quantity * Price) > ...	3 row(s) returned	0.000 sec / 0.000 sec
26	09:30:26	SELECT AVG(Price) AS Average_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

77°F Sunny 9:30 AM 4/4/2025

-- Average Quantity of Products Sold Per Transaction

SELECT AVG(Quantity) AS Average_Quantity_Sold FROM Sales;



MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- companydb
- employeeedb
- roadwaytraveldb
- sys

Query 1 ×

```
116 GROUP BY Product;
117
118 -- Average Price of Products Sold
119 • SELECT AVG(Price) AS Average_Price FROM Sales;
120
121 -- Average Quantity of Products Sold Per Transaction
122 • SELECT AVG(Quantity) AS Average_Quantity_Sold FROM Sales;
123
```

Result Grid | Filter Rows | Export | Wrap Cell Content |

Average_Quantity_Sold
2,833

Result Grid | Form Editor | Field Types

Administration Schemas

Information

No object selected

Object Info Session

Result 23 ×

Output

Action Output

#	Time	Action	Message	Duration / Fetch
22	09:29:00	SELECT SUM(Quantity * Price) AS Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = MONTH(NOW())	1 row(s) returned	0.000 sec / 0.000 sec
23	09:29:30	SELECT SUM(Quantity * Price) AS High_Quantity_Revenue FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
24	09:29:49	SELECT SUM(Quantity * Price) AS Revenue_After_Date FROM Sales WHERE SaleDate > '2025-02-03' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
25	09:30:07	SELECT Product, SUM(Quantity * Price) AS High_Value_Revenue FROM Sales WHERE (Quantity * Price) > 10000	3 row(s) returned	0.000 sec / 0.000 sec
26	09:30:26	SELECT AVG(Price) AS Average_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
27	09:30:43	SELECT AVG(Quantity) AS Average_Quantity_Sold FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

77°F Sunny

Search

9:30 AM 4/4/2024

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

-- Average Revenue Per Transaction

```
SELECT AVG(Quantity * Price) AS Average_Revenue_Per_Transaction
```

FROM Sales;

The screenshot shows the MySQL Workbench interface. The top navigation bar includes 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help'. The 'Query' tab is selected, showing a toolbar with various icons for database management. The 'Navigator' pane on the left lists 'SCHEMAS' with 'companydb', 'employeedb', 'roadwaytraveldb', and 'sys' selected. The main area displays a query editor with the following SQL code:

```
122 •  SELECT AVG(Quantity) AS Average_Quantity_Sold FROM Sales;
123
124 -- Average Revenue Per Transaction
125 •  SELECT AVG(Quantity * Price) AS Average_Revenue_Per_Transaction
126   FROM Sales;
127
128 -- Average Price Per Product
```

The results grid shows the output of the second query:

Average_Revenue_Per_Transaction
95000.000000

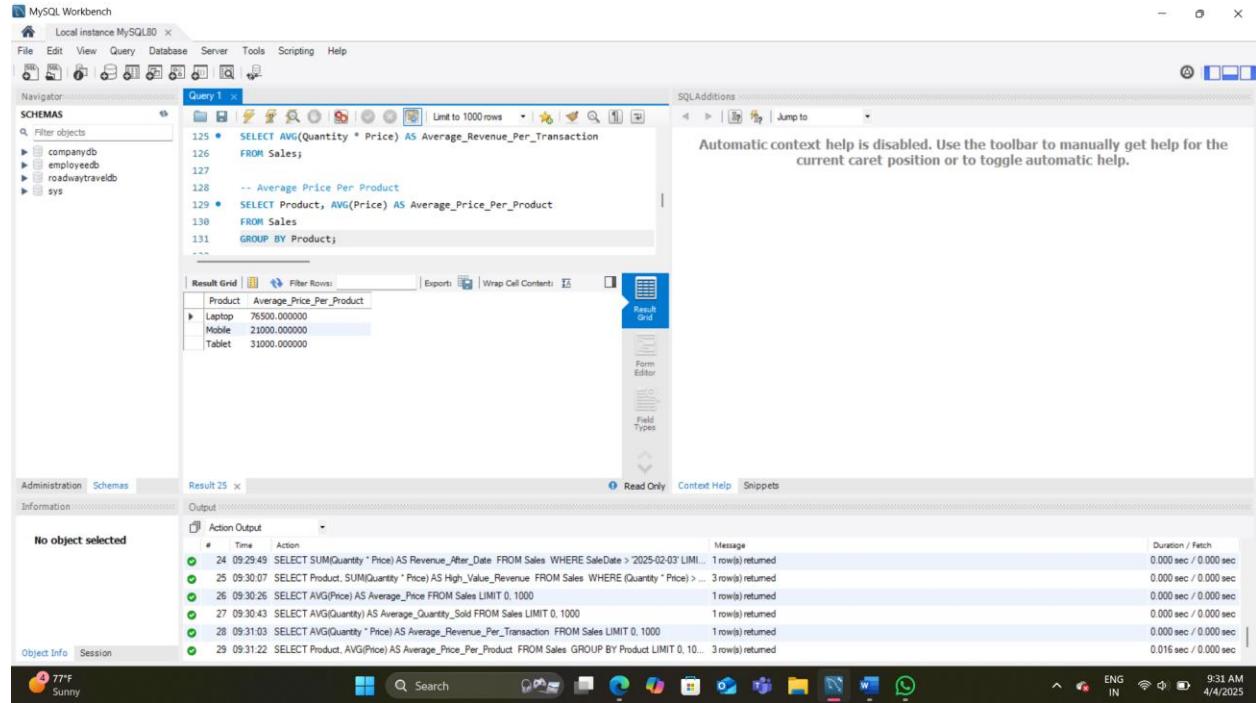
On the right, a message states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." The bottom pane shows the 'Result Grid' tab selected, and the status bar indicates the duration and fetch time as 0.000 sec / 0.000 sec.

-- Average Price Per Product

```
SELECT Product, AVG(Price) AS Average_Price_Per_Product
```

```
FROM Sales
```

```
GROUP BY Product;
```



The screenshot shows the MySQL Workbench interface. The 'Query' tab contains the following SQL code:

```
125 •  SELECT AVG(Quantity * Price) AS Average_Revenue_Per_Transaction
126  FROM Sales;
127
128 -- Average Price Per Product
129 •  SELECT Product, AVG(Price) AS Average_Price_Per_Product
130  FROM Sales
131  GROUP BY Product;
132
```

The 'Result Grid' tab displays the results:

Product	Average_Price_Per_Product
Laptop	76500.000000
Mobile	21000.000000
Tablet	31000.000000

The 'Output' tab shows the execution history:

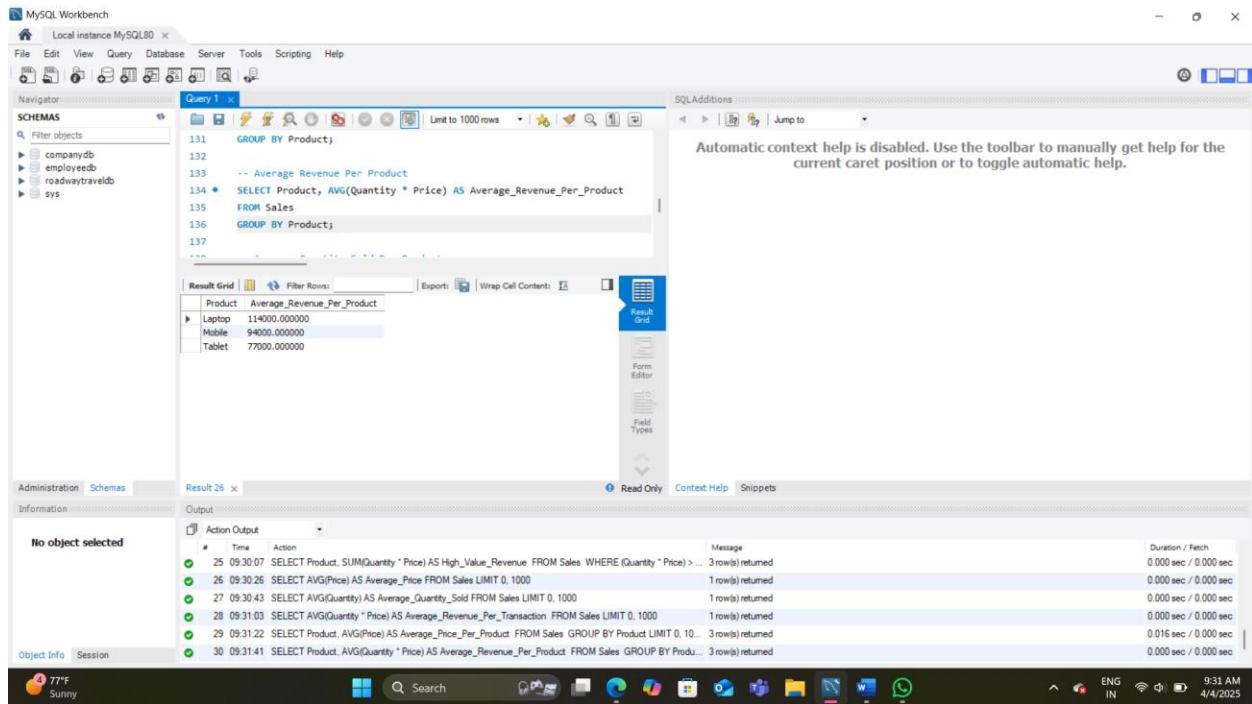
Action	Time	Action	Message	Duration / Fetch
24	09:29:49	SELECT SUM(Quantity * Price) AS Revenue_After_Date FROM Sales WHERE SaleDate > '2025-02-03' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
25	09:30:07	SELECT Product, SUM(Quantity * Price) AS High_Value_Revenue FROM Sales WHERE (Quantity * Price) > ...	3 row(s) returned	0.000 sec / 0.000 sec
26	09:30:26	SELECT AVG(Price) AS Average_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
27	09:30:43	SELECT AVG(Quantity) AS Average_Quantity_Sold FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
28	09:31:03	SELECT AVG(Quantity * Price) AS Average_Revenue_Per_Transaction FROM Sales LIMIT 0, 10...	1 row(s) returned	0.000 sec / 0.000 sec
29	09:31:22	SELECT Product, AVG(Price) AS Average_Price_Per_Product FROM Sales GROUP BY Product LIMIT 0, 10...	3 row(s) returned	0.016 sec / 0.000 sec

-- Average Revenue Per Product

```
SELECT Product, AVG(Quantity * Price) AS Average_Revenue_Per_Product
```

```
FROM Sales
```

```
GROUP BY Product;
```



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1

```

131 GROUP BY Product;
132
133 -- Average Revenue Per Product
134 • SELECT Product, AVG(Quantity * Price) AS Average_Revenue_Per_Product
135 FROM Sales
136 GROUP BY Product;
137

```

Result Grid | Filter Rows: Export: Wrap Cell Contents: Result Grid Form Editor Field Types

Product	Average_Revenue_Per_Product
Laptop	114000.000000
Mobile	94000.000000
Tablet	77000.000000

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas Result 26 x Read Only Context Help Snippets

Information Output

Action Output

Action	Time	Message	Duration / Fetch
25	09:30:07	SELECT Product, SUM(Quantity * Price) AS High_Value_Revenue FROM Sales WHERE (Quantity * Price) > ... 3 row(s) returned	0.000 sec / 0.000 sec
26	09:30:26	SELECT AVG(Price) AS Average_Price FROM Sales LIMIT 0, 1000	0.000 sec / 0.000 sec
27	09:30:43	SELECT AVG(Quantity) AS Average_Quantity_Sold FROM Sales LIMIT 0, 1000	0.000 sec / 0.000 sec
28	09:31:03	SELECT AVG(Quantity * Price) AS Average_Revenue_Per_Transaction FROM Sales LIMIT 0, 1000	0.000 sec / 0.000 sec
29	09:31:22	SELECT Product, AVG(Price) AS Average_Price_Per_Product FROM Sales GROUP BY Product LIMIT 0, 10... 3 row(s) returned	0.016 sec / 0.000 sec
30	09:31:41	SELECT Product, AVG(Quantity * Price) AS Average_Revenue_Per_Product FROM Sales GROUP BY Product... 3 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

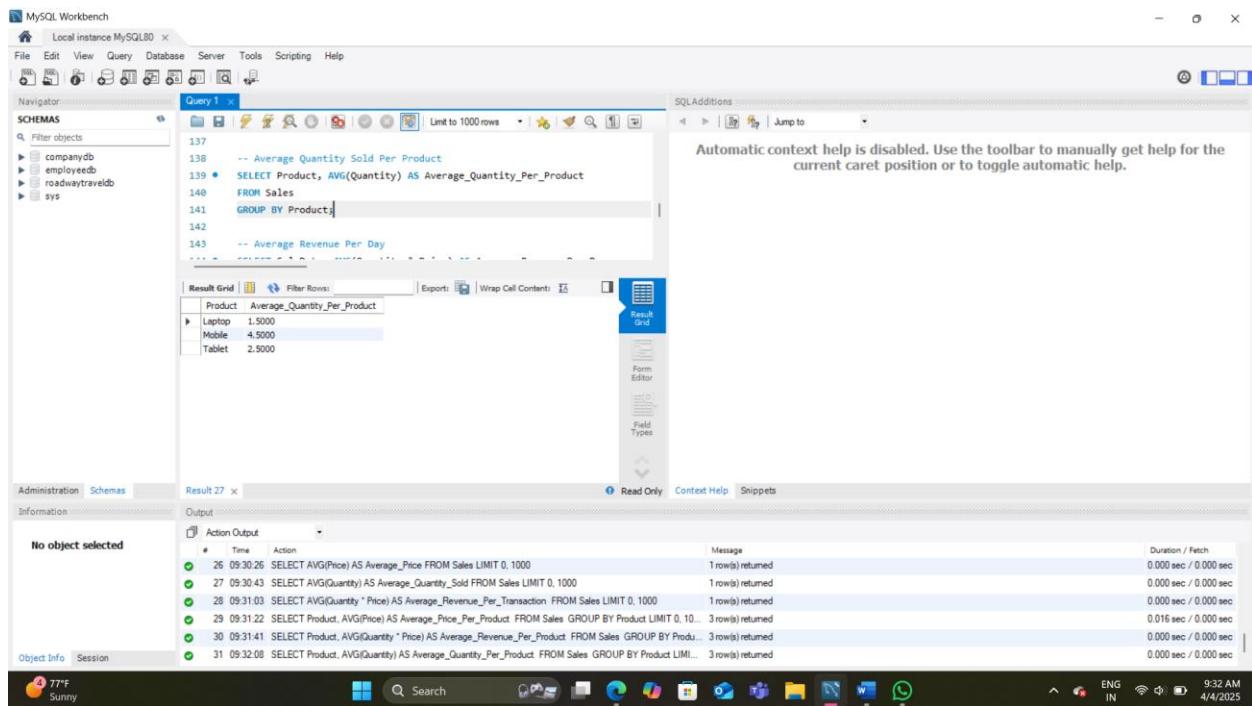
97°F Sunny 9:31 AM 4/4/2025

-- Average Quantity Sold Per Product

SELECT Product, AVG(Quantity) AS Average_Quantity_Per_Product

FROM Sales

GROUP BY Product;



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1

```

137
138 -- Average Quantity Sold Per Product
139 • SELECT Product, AVG(Quantity) AS Average_Quantity_Per_Product
140 FROM Sales
141 GROUP BY Product;
142
143 -- Average Revenue Per Day

```

Result Grid | Filter Rows: Export: Wrap Cell Contents: Result Grid Form Editor Field Types

Product	Average_Quantity_Per_Product
Laptop	1.5000
Mobile	4.5000
Tablet	2.5000

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas Result 27 x Read Only Context Help Snippets

Information Output

Action Output

Action	Time	Message	Duration / Fetch
26	09:30:26	SELECT AVG(Price) AS Average_Price FROM Sales LIMIT 0, 1000	1 row(s) returned 0.000 sec / 0.000 sec
27	09:30:43	SELECT AVG(Quantity) AS Average_Quantity_Sold FROM Sales LIMIT 0, 1000	1 row(s) returned 0.000 sec / 0.000 sec
28	09:31:03	SELECT AVG(Quantity * Price) AS Average_Revenue_Per_Transaction FROM Sales LIMIT 0, 1000	1 row(s) returned 0.000 sec / 0.000 sec
29	09:31:22	SELECT Product, AVG(Price) AS Average_Price_Per_Product FROM Sales GROUP BY Product LIMIT 0, 10... 3 row(s) returned	0.016 sec / 0.000 sec
30	09:31:41	SELECT Product, AVG(Quantity * Price) AS Average_Revenue_Per_Product FROM Sales GROUP BY Product... 3 row(s) returned	0.000 sec / 0.000 sec
31	09:32:08	SELECT Product, AVG(Quantity) AS Average_Quantity_Per_Product FROM Sales GROUP BY Product LIMIT 0, 10... 3 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

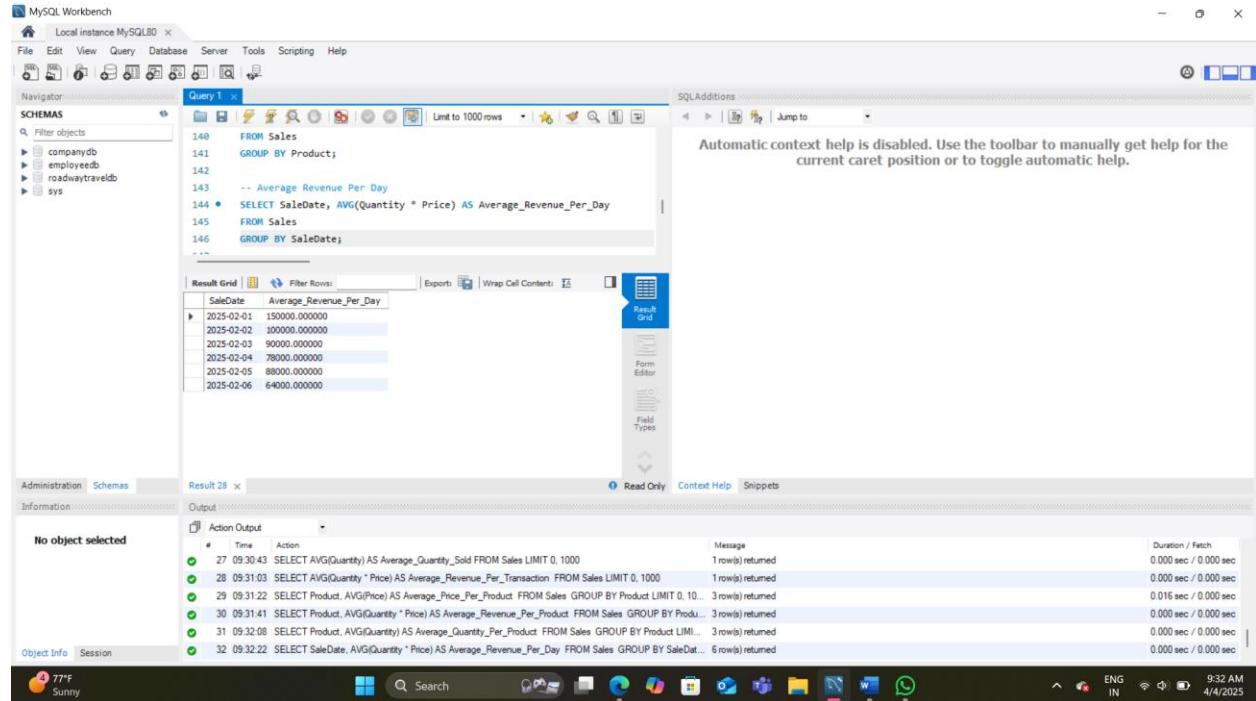
97°F Sunny 9:32 AM 4/4/2025

-- Average Revenue Per Day

```
SELECT SaleDate, AVG(Quantity * Price) AS Average_Revenue_Per_Day
```

```
FROM Sales
```

```
GROUP BY SaleDate;
```



The screenshot shows the MySQL Workbench interface. The 'Query 1' tab contains the following SQL code:

```
140  FROM Sales
141  GROUP BY Product;
142
143  -- Average Revenue Per Day
144  •  SELECT SaleDate, AVG(Quantity * Price) AS Average_Revenue_Per_Day
145  FROM Sales
146  GROUP BY SaleDate;
```

The 'Result Grid' shows the output of the query:

SaleDate	Average_Revenue_Per_Day
2025-02-01	150000.000000
2025-02-02	300000.000000
2025-02-03	90000.000000
2025-02-04	78000.000000
2025-02-05	88000.000000
2025-02-06	64000.000000

The 'Information' tab shows the execution history:

Action	Time	Action	Message	Duration / Fetch
27	09:30:43	SELECT AVG(Quantity) AS Average_Quantity_Sold FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
28	09:31:03	SELECT AVG(Quantity * Price) AS Average_Revenue_Per_Transaction FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
29	09:31:22	SELECT Product, AVG(Price) AS Average_Price_Per_Product FROM Sales GROUP BY Product LIMIT 0, 1000	3 row(s) returned	0.016 sec / 0.000 sec
30	09:31:41	SELECT Product, AVG(Quantity * Price) AS Average_Revenue_Per_Product FROM Sales GROUP BY Product LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
31	09:32:08	SELECT Product, AVG(Quantity) AS Average_Quantity_Per_Product FROM Sales GROUP BY Product LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
32	09:32:22	SELECT SaleDate, AVG(Quantity * Price) AS Average_Revenue_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

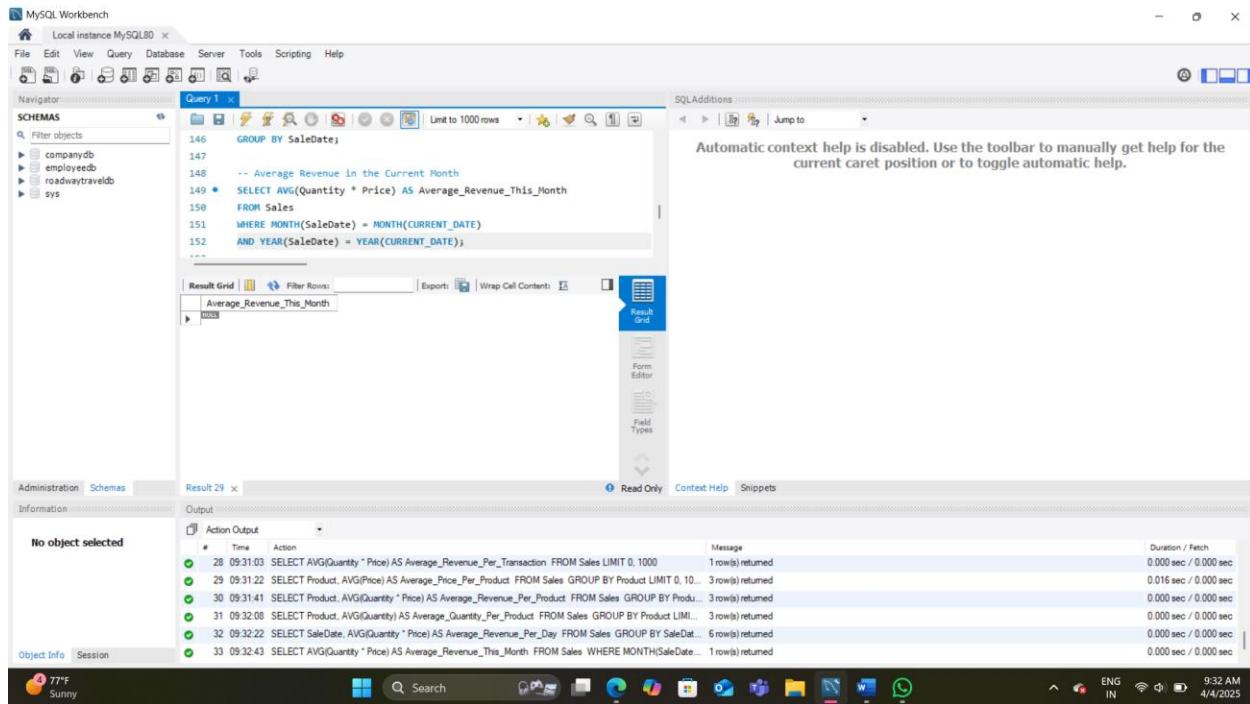
-- Average Revenue in the Current Month

```
SELECT AVG(Quantity * Price) AS Average_Revenue_This_Month
```

```
FROM Sales
```

```
WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
```

```
AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
```



MySQL Workbench - Local instance MySQL80

Query 1

```

146 GROUP BY SaleDate;
147
148 -- Average Revenue in the Current Month
149 * SELECT AVG(Quantity * Price) AS Average_Revenue_This_Month
150 FROM Sales
151 WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
152 AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
153

```

Result Grid | Filter Rows | Export | Wrap Cell Contents | Result Grid | Form Editor | Field Types

Average_Revenue_This_Month

Result 29

Action Output

Action	Time	Message	Duration / Fetch
28	09:31:03	SELECT AVG(Quantity * Price) AS Average_Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE) AND YEAR(SaleDate) = YEAR(CURRENT_DATE);	0.000 sec / 0.000 sec
29	09:31:22	SELECT Product, AVG(Price) AS Average_Price_Per_Product FROM Sales GROUP BY Product LIMIT 0, 10;	0.016 sec / 0.000 sec
30	09:31:41	SELECT Product, AVG(Quantity * Price) AS Average_Revenue_Per_Product FROM Sales GROUP BY Product;	0.000 sec / 0.000 sec
31	09:32:08	SELECT Product, AVG(Quantity) AS Average_Quantity_Per_Product FROM Sales GROUP BY Product LIMIT 0, 10;	0.000 sec / 0.000 sec
32	09:32:22	SELECT SaleDate, AVG(Quantity * Price) AS Average_Revenue_Per_Day FROM Sales GROUP BY SaleDate;	0.000 sec / 0.000 sec
33	09:32:43	SELECT AVG(Quantity * Price) AS Average_Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE) AND YEAR(SaleDate) = YEAR(CURRENT_DATE);	0.000 sec / 0.000 sec

Object Info | Session

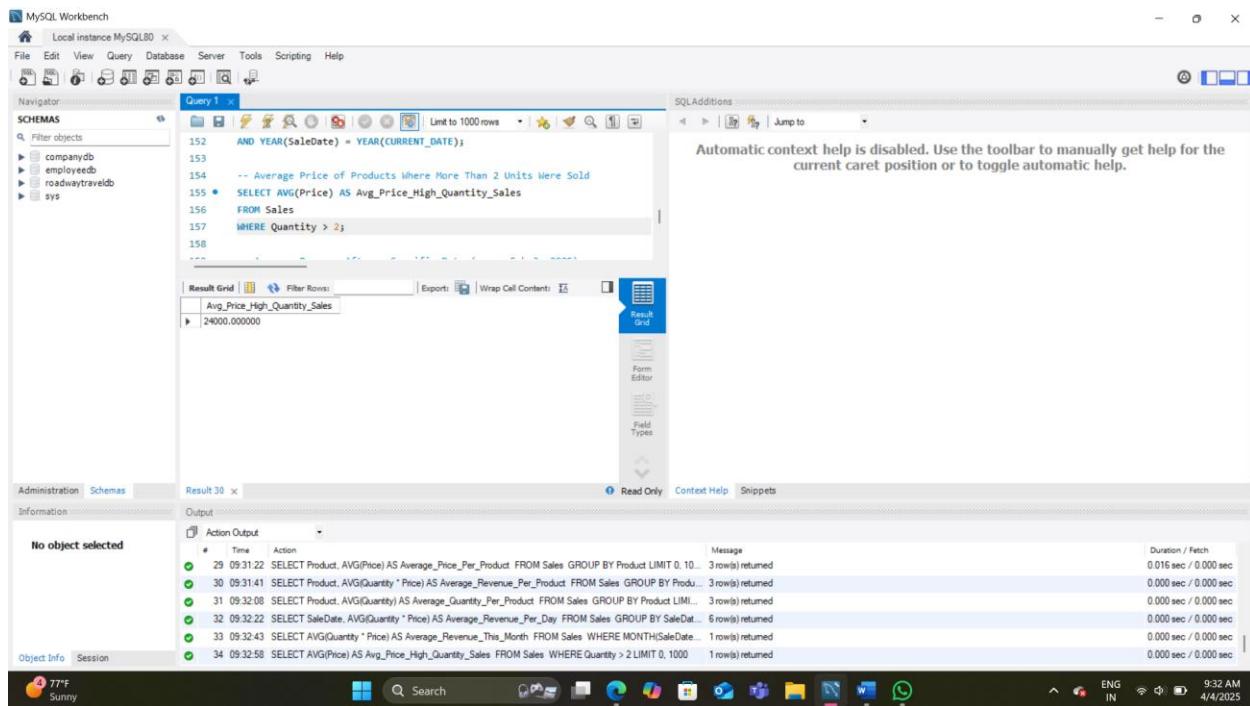
9:32 AM 4/4/2025

-- Average Price of Products Where More Than 2 Units Were Sold

SELECT AVG(Price) AS Avg_Price_High_Quantity_Sales

FROM Sales

WHERE Quantity > 2;



MySQL Workbench - Local instance MySQL80

Query 1

```

152 AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
153
154 -- Average Price of Products Where More Than 2 Units Were Sold
155 * SELECT AVG(Price) AS Avg_Price_High_Quantity_Sales
156 FROM Sales
157 WHERE Quantity > 2;
158

```

Result Grid | Filter Rows | Export | Wrap Cell Contents | Result Grid | Form Editor | Field Types

Avg_Price_High_Quantity_Sales

Result 30

Action Output

Action	Time	Message	Duration / Fetch
29	09:31:22	SELECT Product, AVG(Price) AS Average_Price_Per_Product FROM Sales GROUP BY Product LIMIT 0, 10;	0.016 sec / 0.000 sec
30	09:31:41	SELECT Product, AVG(Quantity * Price) AS Average_Revenue_Per_Product FROM Sales GROUP BY Product;	0.000 sec / 0.000 sec
31	09:32:08	SELECT Product, AVG(Quantity) AS Average_Quantity_Per_Product FROM Sales GROUP BY Product LIMIT 0, 10;	0.000 sec / 0.000 sec
32	09:32:22	SELECT SaleDate, AVG(Quantity * Price) AS Average_Revenue_Per_Day FROM Sales GROUP BY SaleDate;	0.000 sec / 0.000 sec
33	09:32:43	SELECT AVG(Quantity * Price) AS Average_Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE) AND YEAR(SaleDate) = YEAR(CURRENT_DATE);	0.000 sec / 0.000 sec
34	09:32:58	SELECT AVG(Price) AS Avg_Price_High_Quantity_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000;	0.000 sec / 0.000 sec

Object Info | Session

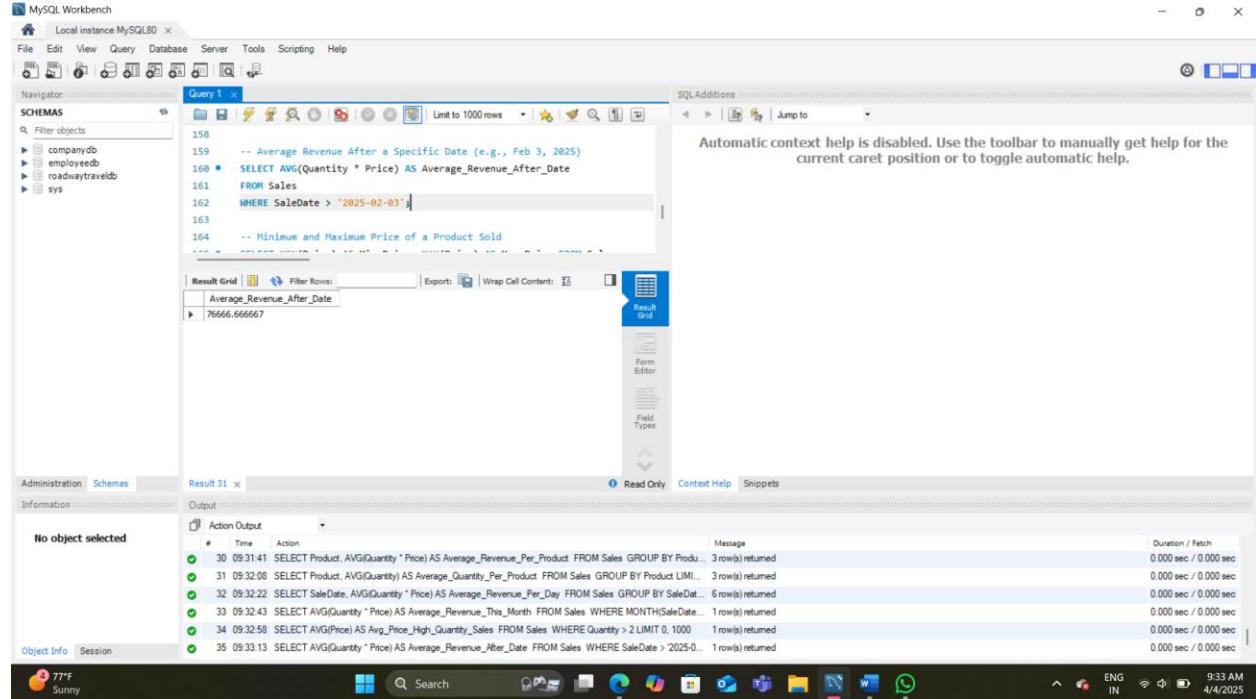
9:32 AM 4/4/2025

-- Average Revenue After a Specific Date (e.g., Feb 3, 2025)

```
SELECT AVG(Quantity * Price) AS Average_Revenue_After_Date
```

```
FROM Sales
```

```
WHERE SaleDate > '2025-02-03';
```



The screenshot shows the MySQL Workbench interface. The 'Query 1' tab contains the SQL code for calculating average revenue after a specific date. The 'Result Grid' shows the result of the query: 76666.666667. The 'Information' tab shows the execution history for the session, listing various SELECT statements with their execution times, message counts, and duration.

Action	Time	Action	Message	Duration / Fetch
30	09:31:41	SELECT Product, AVG(Quantity * Price) AS Average_Revenue_Per_Product FROM Sales GROUP BY Product	3 row(s) returned	0.000 sec / 0.000 sec
31	09:32:08	SELECT Product, AVG(Quantity) AS Average_Quantity_Per_Product FROM Sales GROUP BY Product LIMIT 1	3 row(s) returned	0.000 sec / 0.000 sec
32	09:32:22	SELECT SaleDate, AVG(Quantity * Price) AS Average_Revenue_Per_Day FROM Sales GROUP BY SaleDate	6 row(s) returned	0.000 sec / 0.000 sec
33	09:32:43	SELECT AVG(Quantity * Price) AS Average_Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = 2	1 row(s) returned	0.000 sec / 0.000 sec
34	09:32:58	SELECT AVG(Price) AS Avg_Price_High_Quantity_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
35	09:33:13	SELECT AVG(Quantity * Price) AS Average_Revenue_After_Date FROM Sales WHERE SaleDate > '2025-02-03'	1 row(s) returned	0.000 sec / 0.000 sec

-- Minimum and Maximum Price of a Product Sold

```
SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales;
```

```

161 FROM Sales
162 WHERE SaleDate > '2025-02-03'
163
164 -- Minimum and Maximum Price of a Product Sold
165 • SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales
166
167 -- Minimum and Maximum Quantity of Products Sold in a Single Transaction
168
169
170

```

Min_Price	Max_Price
20000.00	78000.00

Action Output

Time	Action	Message	Duration / Fetch
31 09:32:08	SELECT Product, AVG(Quantity) AS Average_Quantity_Per_Product FROM Sales GROUP BY Product LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
32 09:32:22	SELECT SaleDate, AVG(Quantity * Price) AS Average_Revenue_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
33 09:32:43	SELECT AVG(Quantity * Price) AS Average_Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = 2	1 row(s) returned	0.000 sec / 0.000 sec
34 09:32:58	SELECT AVG(Price) AS Avg_Price_High_Quantity_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
35 09:33:13	SELECT AVG(Quantity * Price) AS Average_Revenue_After_Date FROM Sales WHERE SaleDate > '2025-02-03' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
36 09:33:30	SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

-- Minimum and Maximum Quantity of Products Sold in a Single Transaction

SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold
FROM Sales;

```

164 -- Minimum and Maximum Price of a Product Sold
165 • SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales
166
167 -- Minimum and Maximum Quantity of Products Sold in a Single Transaction
168 • SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold
169 FROM Sales
170

```

Min_Quantity_Sold	Max_Quantity_Sold
1	5

Action Output

Time	Action	Message	Duration / Fetch
32 09:32:22	SELECT SaleDate, AVG(Quantity * Price) AS Average_Revenue_Per_Day FROM Sales GROUP BY SaleDate LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
33 09:32:43	SELECT AVG(Quantity * Price) AS Average_Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = 2	1 row(s) returned	0.000 sec / 0.000 sec
34 09:32:58	SELECT AVG(Price) AS Avg_Price_High_Quantity_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
35 09:33:13	SELECT AVG(Quantity * Price) AS Average_Revenue_After_Date FROM Sales WHERE SaleDate > '2025-02-03' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
36 09:33:30	SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
37 09:33:45	SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

-- Minimum and Maximum Revenue Generated from a Single Transaction

```
SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_Revenue
FROM Sales;
```

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- companydb
- employeedb
- roadwaytraveldb
- sys

Query 1

```
167 -- Minimum and Maximum Quantity of Products Sold in a Single Transaction
168 •  SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_
169   FROM Sales;
170
171 -- Minimum and Maximum Revenue Generated from a Single Transaction
172 •  SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_
173   FROM Sales;
174
```

Result Grid

Min_Revenue	Max_Revenue
64000.00	150000.00

Result 34

Output

Action Output

#	Time	Action	Message	Duration / Fetch
33	09:32:43	SELECT AVG(Quantity * Price) AS Average_Revenue_This_Month FROM Sales WHERE MONTH(SaleDate) = 9	1 row(s) returned	0.000 sec / 0.000 sec
34	09:32:58	SELECT AVG(Price) AS Avg_Price_High_Quantity_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
35	09:33:13	SELECT AVG(Quantity * Price) AS Average_Revenue_After_Date FROM Sales WHERE SaleDate > '2025-01-01' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
36	09:33:30	SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
37	09:33:45	SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
38	09:34:15	SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_Revenue FROM Sales LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

77°F Sunny

Search

9:34 AM 4/4/2025

-- Minimum and Maximum Price Per Product

```
SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS
Max_Price_Per_Product
FROM Sales
GROUP BY Product;
```

```

173 FROM Sales
174
175 -- Minimum and Maximum Price Per Product
176 • SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS Max_Price_Per_Product
177 FROM Sales
178 GROUP BY Product;
179

```

Product	Min_Price_Per_Product	Max_Price_Per_Product
Laptop	75000.00	78000.00
Mobile	20000.00	22000.00
Tablet	30000.00	32000.00

Result 35 x

Action Output

Action	Time	Message	Duration / Fetch
34	09:32:58	SELECT AVG(Price) AS Avg_Price_High_Quantity_Sales FROM Sales WHERE Quantity > 2 LIMIT 0, 1000	1row(s) returned 0.000 sec / 0.000 sec
35	09:33:13	SELECT AVG(Quantity * Price) AS Average_Revenue_After_Date FROM Sales WHERE SaleDate > '2025-01-01'	1row(s) returned 0.000 sec / 0.000 sec
36	09:33:30	SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales LIMIT 0, 1000	1row(s) returned 0.000 sec / 0.000 sec
37	09:33:45	SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold FROM Sales LIMIT 0, 1000	1row(s) returned 0.000 sec / 0.000 sec
38	09:34:15	SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_Revenue FROM Sales LIMIT 0, 1000	1row(s) returned 0.000 sec / 0.000 sec
39	09:34:35	SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS Max_Price_Per_Product FROM Sales	3row(s) returned 0.000 sec / 0.000 sec

77°F Sunny

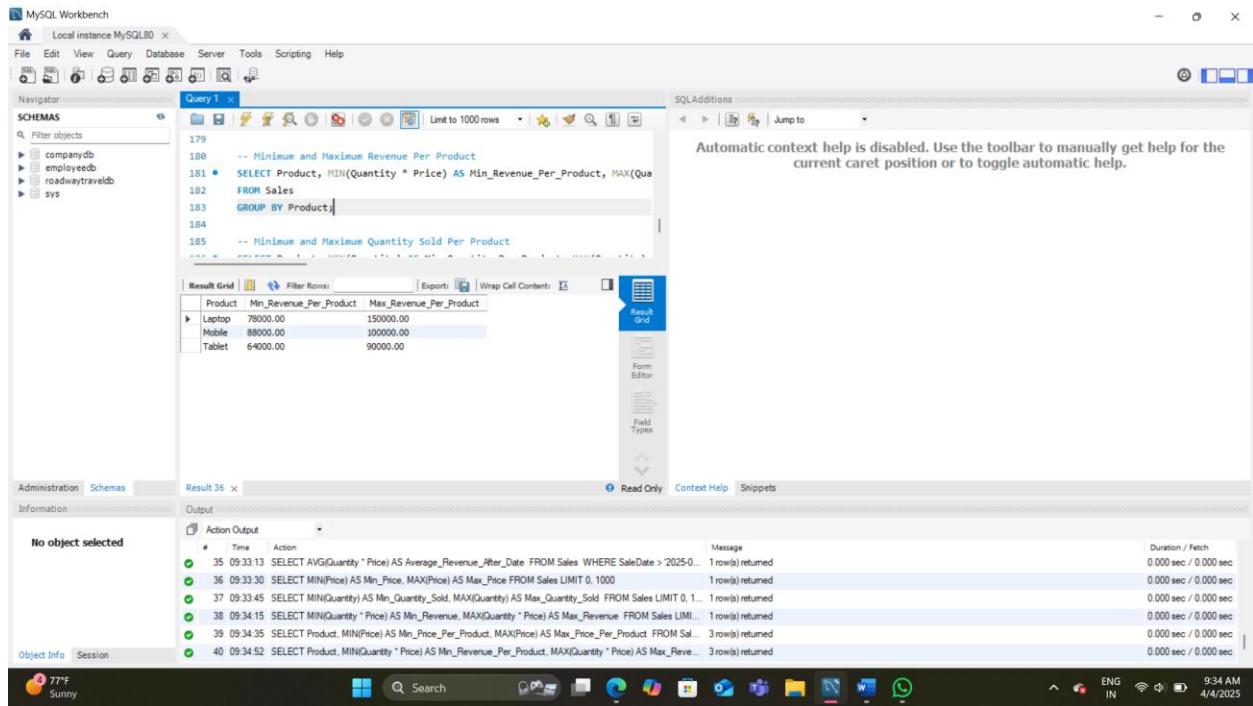
ENG IN 9:34 AM 4/4/2025

-- Minimum and Maximum Revenue Per Product

```

SELECT Product, MIN(Quantity * Price) AS Min_Revenue_Per_Product, MAX(Quantity * Price) AS Max_Revenue_Per_Product
FROM Sales
GROUP BY Product;

```



```

179
180  -- Minimum and Maximum Revenue Per Product
181  •  SELECT Product, MIN(Quantity * Price) AS Min_Revenue_Per_Product, MAX(Qua
182  FROM Sales
183  GROUP BY Product;
184
185  -- Minimum and Maximum Quantity Sold Per Product

```

Product	Min_Revenue_Per_Product	Max_Revenue_Per_Product
Laptop	78000.00	150000.00
Mobile	88000.00	100000.00
Tablet	64000.00	90000.00

-- Minimum and Maximum Quantity Sold Per Product

SELECT Product, MIN(Quantity) AS Min_Quantity_Per_Product, MAX(Quantity) AS Max_Quantity_Per_Product

FROM Sales

GROUP BY Product;

MySQL Workbench - Local instance MySQL80

Query 1

```

182 FROM Sales
183 GROUP BY Product;
184
185 -- Minimum and Maximum Quantity Sold Per Product
186 • SELECT Product, MIN(Quantity) AS Min_Quantity_Per_Product, MAX(Quantity)
187 FROM Sales
188 GROUP BY Product;
189

```

Result Grid

Product	Min_Quantity_Per_Product	Max_Quantity_Per_Product
Laptop	1	2
Mobile	4	5
Tablet	2	3

Administration Schemas Result 37

Action Output

Action	Time	Message	Duration / Fetch
36	09:33:30	SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales LIMIT 0, 1000	0.000 sec / 0.000 sec
37	09:33:45	SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold FROM Sales LIMIT 0, 1000	0.000 sec / 0.000 sec
38	09:34:15	SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_Revenue FROM Sales LIMIT 0, 1000	0.000 sec / 0.000 sec
39	09:34:35	SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS Max_Price_Per_Product FROM Sales GROUP BY Product	0.000 sec / 0.000 sec
40	09:34:52	SELECT Product, MIN(Quantity * Price) AS Min_Revenue_Per_Product, MAX(Quantity * Price) AS Max_Revenue_Per_Product FROM Sales GROUP BY Product	0.000 sec / 0.000 sec
41	09:35:08	SELECT Product, MIN(Quantity_Per_Product), MAX(Quantity_Per_Product) FROM Sales GROUP BY Product	0.000 sec / 0.000 sec

Object Info Session

77°F Sunny

-- Minimum and Maximum Revenue Per Day

```

SELECT SaleDate, MIN(Quantity * Price) AS Min_Revenue_Per_Day, MAX(Quantity * Price)
AS Max_Revenue_Per_Day
FROM Sales
GROUP BY SaleDate;

```

MySQL Workbench - Local instance MySQL80

Navigator: SCHEMAS

Query 1

```

188 GROUP BY Product;
189
190 -- Minimum and Maximum Revenue Per Day
191 • SELECT SaleDate, MIN(Quantity * Price) AS Min_Revenue_Per_Day, MAX(Quantity * Price) AS Max_Revenue_Per_Day
192 FROM Sales
193 GROUP BY SaleDate;
194
  
```

Result Grid

SaleDate	Min_Revenue_Per_Day	Max_Revenue_Per_Day
2025-02-01	150000.00	150000.00
2025-02-02	100000.00	100000.00
2025-02-03	90000.00	90000.00
2025-02-04	78000.00	78000.00
2025-02-05	88000.00	88000.00
2025-02-06	64000.00	64000.00

Administration Schemas Result 38

Information Output

Action Output

Time	Action	Message	Duration / Fetch
37 09:33:45	SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold FROM Sales LIMIT 0, 1...	1 row(s) returned	0.000 sec / 0.000 sec
38 09:34:15	SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_Revenue FROM Sales LIMIT 0, 1...	1 row(s) returned	0.000 sec / 0.000 sec
39 09:34:35	SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS Max_Price_Per_Prod...	3 row(s) returned	0.000 sec / 0.000 sec
40 09:34:52	SELECT Product, MIN(Quantity * Price) AS Min_Revenue_Per_Product, MAX(Quantity * Price) AS Max_Reven...	3 row(s) returned	0.000 sec / 0.000 sec
41 09:35:08	SELECT Product, MIN(Quantity) AS Min_Quantity_Per_Product, MAX(Quantity) AS Max_Quantity_Per_Prod...	3 row(s) returned	0.000 sec / 0.000 sec
42 09:35:25	SELECT SaleDate, MIN(Quantity * Price) AS Min_Revenue_Per_Day, MAX(Quantity * Price) AS Max_Reven...	6 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

77°F Sunny 9:35 AM 4/4/2025

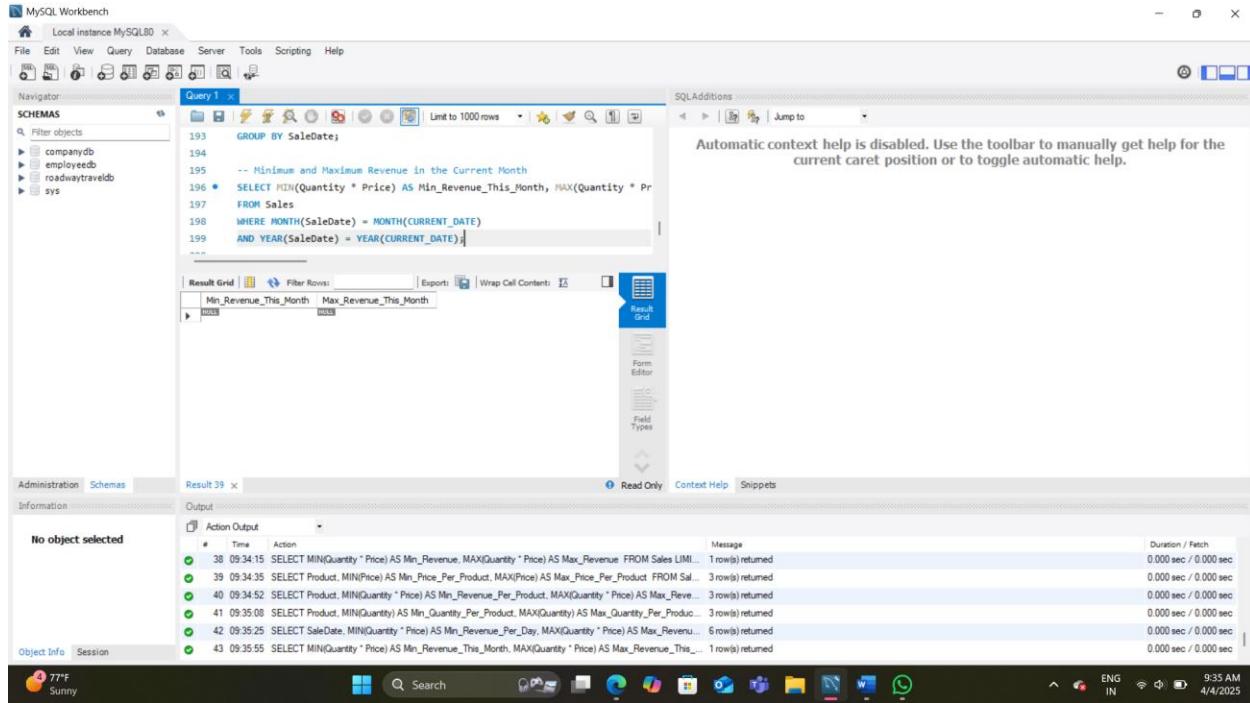
-- Minimum and Maximum Revenue in the Current Month

SELECT MIN(Quantity * Price) AS Min_Revenue_This_Month, MAX(Quantity * Price) AS Max_Revenue_This_Month

FROM Sales

WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)

AND YEAR(SaleDate) = YEAR(CURRENT_DATE);



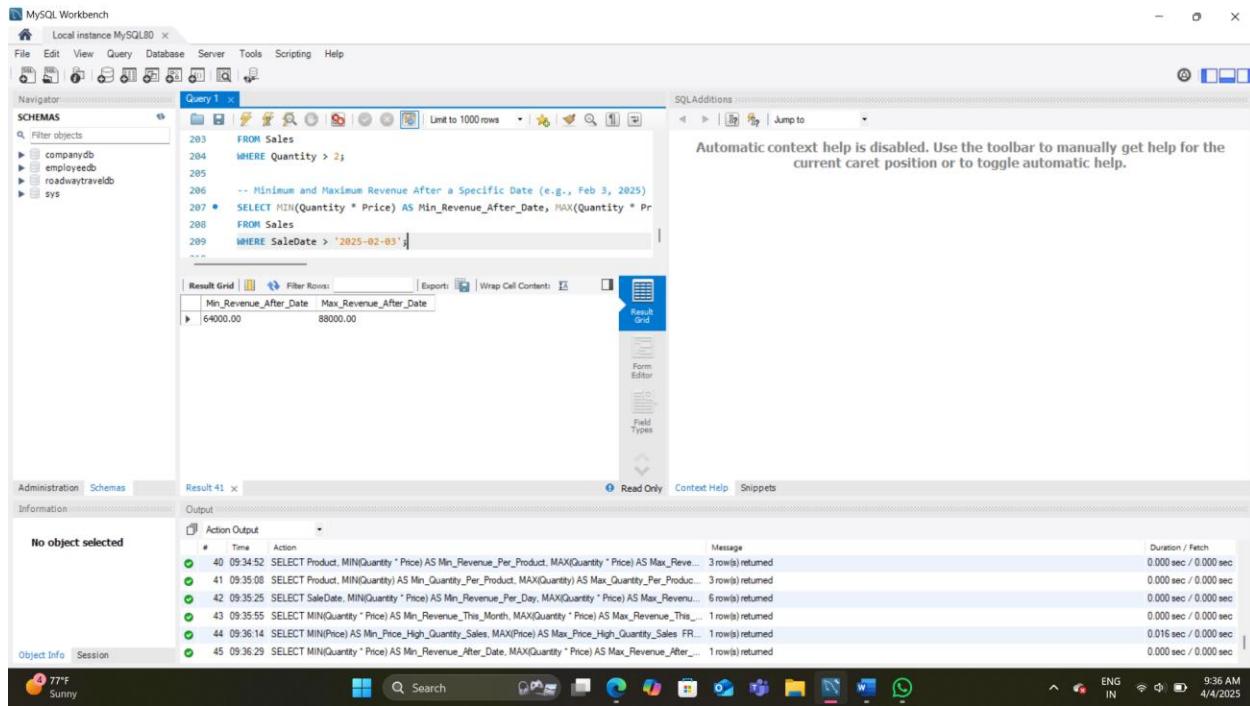
-- Minimum and Maximum Price of Products Where More Than 2 Units Were Sold

```
SELECT MIN(Price) AS Min_Price_High_Quantity_Sales, MAX(Price) AS
Max_Price_High_Quantity_Sales
FROM Sales
WHERE Quantity > 2;
```

The screenshot shows the MySQL Workbench interface. In the top-left, the 'Query 1' tab is active, displaying a SQL query to find the minimum and maximum price of products sold in February 2025. The results are shown in a table with two rows: 'Min_Price_High_Quantity_Sales' and 'Max_Price_High_Quantity_Sales', both with a value of 20000.00. The bottom section, 'Result 40', shows the execution log with 16 entries, all of which completed in 0.000 sec. The system tray at the bottom indicates it's 9:36 AM on 4/4/2025, the weather is 77°F and sunny, and the user is connected to the 'ENG' network.

-- Minimum and Maximum Revenue After a Specific Date (e.g., Feb 3, 2025)

```
SELECT MIN(Quantity * Price) AS Min_Revenue_After_Date, MAX(Quantity * Price) AS
Max_Revenue_After_Date
FROM Sales
WHERE SaleDate > '2025-02-03';
```



Practical 8:

Aim: Given Customers and Orders tables, write SQL queries to perform INNER JOIN, LEFT JOIN, and RIGHT JOIN to retrieve combined data for customer orders.

Code:

-- Create the Database

CREATE DATABASE CompanyDB;

-- Use the Database

USE CompanyDB;

-- Create the Customers Table

CREATE TABLE Customers (

customer_id INT PRIMARY KEY,

customer_name VARCHAR(100) NOT NULL

```
);

-- Create the Orders Table

CREATE TABLE Orders (
    order_id INT PRIMARY KEY,
    order_date DATE NOT NULL,
    customer_id INT,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);

-- Insert Data into Customers Table

INSERT INTO Customers (customer_id, customer_name) VALUES
(1, 'Janhavi'),
(2, 'Mansi'),
(3, 'Gauri'),
(4, 'Sanjivani');

-- Insert Data into Orders Table

INSERT INTO Orders (order_id, order_date, customer_id) VALUES
(101, '2024-01-01', 1),
(102, '2024-01-02', 2),
(103, '2024-01-03', 4);

-- Select all data from Customers

SELECT * FROM Customers;
```

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

Query 1:

```

23 -- Insert Data Into Orders Table
24 • INSERT INTO Orders (order_id, order_date, customer_id) VALUES
25 (101, '2024-01-01', 1),
26 (102, '2024-01-02', 2),
27 (103, '2024-01-03', 4);
28 -- Select all data from Customers
29 • SELECT * FROM Customers
  
```

Result Grid:

customer_id	customer_name
1	Janhavi
2	Mansi
3	Gauri
4	Sangjivani
101	NULL

Administration Schemas Customers 1 x

Information Output

Action Output:

#	Time	Action	Message	Duration / Fetch
2	10:02:01	USE CompanyDB	0 row(s) affected	0.000 sec
3	10:02:04	CREATE TABLE Customers (customer_id INT PRIMARY KEY, customer_name VARCHAR(100) NOT NULL)	0 row(s) affected	0.031 sec
4	10:02:07	CREATE TABLE Orders (order_id INT PRIMARY KEY, order_date DATE NOT NULL, customer_id INT)	0 row(s) affected	0.031 sec
5	10:02:11	INSERT INTO Customers (customer_id, customer_name) VALUES (1, 'Janhavi'), (2, 'Mansi'), (3, 'Gauri'), (4, 'Sangjivani')	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.000 sec
6	10:02:15	INSERT INTO Orders (order_id, order_date, customer_id) VALUES (101, '2024-01-01', 1), (102, '2024-01-02', 2), (103, '2024-01-03', 4)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
7	10:02:19	SELECT * FROM Customers LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

80°F Haze 10:02 AM 4/4/2025 ENG IN

-- Select all data from Orders

SELECT * FROM Orders;

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

Query 1:

```

29 • SELECT * FROM Customers
30 -- Select all data from Orders
31 • SELECT * FROM Orders;
32 -- INNER JOIN: Show customers and their orders (only those who have placed orders)
33 • SELECT
34   c.customer_id,
35   c.customer_name,
  
```

Result Grid:

order_id	order_date	customer_id
101	2024-01-01	1
102	2024-01-02	2
103	2024-01-03	4
1005	NULL	NULL

Administration Schemas Orders 2 x

Information Output

Action Output:

#	Time	Action	Message	Duration / Fetch
3	10:02:04	CREATE TABLE Customers (customer_id INT PRIMARY KEY, customer_name VARCHAR(100) NOT NULL)	0 row(s) affected	0.031 sec
4	10:02:07	CREATE TABLE Orders (order_id INT PRIMARY KEY, order_date DATE NOT NULL, customer_id INT)	0 row(s) affected	0.031 sec
5	10:02:11	INSERT INTO Customers (customer_id, customer_name) VALUES (1, 'Janhavi'), (2, 'Mansi'), (3, 'Gauri'), (4, 'Sangjivani')	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.000 sec
6	10:02:15	INSERT INTO Orders (order_id, order_date, customer_id) VALUES (101, '2024-01-01', 1), (102, '2024-01-02', 2), (103, '2024-01-03', 4)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
7	10:02:19	SELECT * FROM Customers LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
8	10:02:47	SELECT * FROM Orders LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

80°F Haze 10:02 AM 4/4/2025 ENG IN

-- INNER JOIN: Show customers and their orders (only those who have placed orders)

SELECT

```

c.customer_id,
c.customer_name,
o.order_id,
o.order_date
FROM
Customers c
INNER JOIN
Orders o
ON
c.customer_id = o.customer_id;

```

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

companydb

Tables

Views

Stored Procedures

Functions

customerdb

employeedb

roadwaytraveldb

salesdb

sys

Query 1

```

38  FROM
39    Customers c
40  INNER JOIN
41    Orders o
42  ON
43    c.customer_id = o.customer_id
44  -- LEFT JOIN: Show all customers with their orders (if any)

```

Result Grid

customer_id	customer_name	order_id	order_date
1	Janhavi	101	2024-01-01
2	Mansi	102	2024-01-02
4	Sargivari	103	2024-01-03

SQL Additions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result 3

Read Only

Context Help Snippets

Action Output

#	Time	Action	Message	Duration / Fetch
4	10:02:07	CREATE TABLE Orders (order_id INT PRIMARY KEY, order_date DATE NOT NULL, customer_id ...)	0 row(s) affected	0.031 sec
5	10:02:11	INSERT INTO Customers (customer_id, customer_name) VALUES (1, 'Janhavi'), (2, 'Mansi'), (3, 'Gauri'), (4, 'Sa...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.000 sec
6	10:02:15	INSERT INTO Orders (order_id, order_date, customer_id) VALUES (101, '2024-01-01', 1), (102, '2024-01-02', 2), (103, '2024-01-03', 4)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
7	10:02:19	SELECT * FROM Customers LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
8	10:02:47	SELECT * FROM Orders LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
9	10:03:21	SELECT c.customer_id, c.customer_name, o.order_id, o.order_date FROM Customers c ...	3 row(s) returned	0.000 sec / 0.000 sec

Information

No object selected

Object Info Session

80°F Haze

Search

ENG IN 10:03 AM 4/4/2025

-- LEFT JOIN: Show all customers with their orders (if any)

```

SELECT
c.customer_id,
c.customer_name,
o.order_id,

```

```

o.order_date
FROM
Customers c
LEFT JOIN
Orders o
ON
c.customer_id = o.customer_id;

```

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

companydb

Tables

Views

Stored Procedures

Functions

customerdb

employeesdb

roadwaytraveldb

salesdb

sys

Query 1

```

49   o.order_date
50   FROM
51   Customers c
52   LEFT JOIN
53   Orders o
54   ON
55   c.customer_id = o.customer_id;

```

Result Grid

customer_id	customer_name	order_id	order_date
1	Janhvi	101	2024-01-01
2	Mansi	102	2024-01-02
3	Gauri	103	2024-01-03
4	Sanghani	103	2024-01-03

Result 4

Read Only

Context Help Snippets

Action Output

Action	Message	Duration / Fetch
5 10:02:11 INSERT INTO Customers (customer_id, customer_name) VALUES (1, 'Janhvi'), (2, 'Mansi'), (3, 'Gauri'), (4, 'Sa...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.000 sec
6 10:02:15 INSERT INTO Orders (order_id, order_date, customer_id) VALUES (101, 2024-01-01, 1), (102, 2024-01-02, 2...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
7 10:02:19 SELECT * FROM Customers LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
8 10:02:47 SELECT * FROM Orders LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
9 10:03:21 SELECT c.customer_id, c.customer_name, o.order_id, o.order_date FROM Customers c ...	3 row(s) returned	0.000 sec / 0.000 sec
10 10:03:58 SELECT c.customer_id, c.customer_name, o.order_id, o.order_date FROM Customers c ...	4 row(s) returned	0.000 sec / 0.000 sec

80°F Haze

ENG IN 10:03 AM 4/4/2025

-- RIGHT JOIN: Show all orders with customer details (if any)

```

SELECT
c.customer_id,
c.customer_name,
o.order_id,
o.order_date
FROM
Customers c

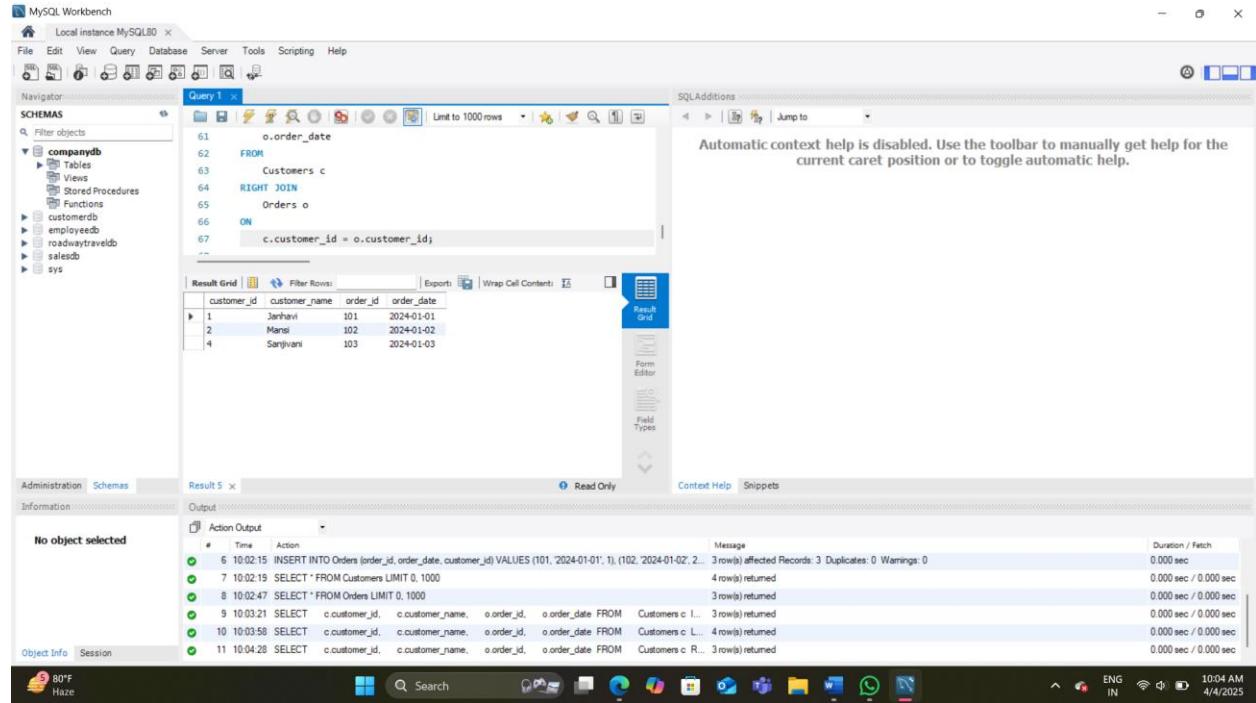
```

RIGHT JOIN

Orders o

ON

c.customer_id = o.customer_id;



The screenshot shows the MySQL Workbench interface with a query editor and a history panel.

Query Editor (Query 1):

```
61     o.order_date
62   FROM
63     Customers c
64   RIGHT JOIN
65     Orders o
66   ON
67     c.customer_id = o.customer_id;
```

Result Grid:

customer_id	customer_name	order_id	order_date
1	Janhvi	101	2024-01-01
2	Mansi	102	2024-01-02
4	Sanjivani	103	2024-01-03

Output Panel (Action Output):

#	Time	Action	Message	Duration / Fetch
6	10:02:15	INSERT INTO Orders (order_id, order_date, customer_id) VALUES (101, '2024-01-01', 1), (102, '2024-01-02', 2), (103, '2024-01-03', 4)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
7	10:02:19	SELECT * FROM Customers LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
8	10:02:47	SELECT * FROM Orders LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
9	10:03:21	SELECT c.customer_id, c.customer_name, o.order_id, o.order_date FROM Customers c ...	3 row(s) returned	0.000 sec / 0.000 sec
10	10:03:58	SELECT c.customer_id, c.customer_name, o.order_id, o.order_date FROM Customers c ...	4 row(s) returned	0.000 sec / 0.000 sec
11	10:04:28	SELECT c.customer_id, c.customer_name, o.order_id, o.order_date FROM Customers c ...	3 row(s) returned	0.000 sec / 0.000 sec