

LAB FILE

Database Management Systems
(DBMS) Academic Year: 2024-2025

Name: Aditya Mhaismale
PRN Number: 2124UCSM1070
Branch: Cyber Security
Subject: DBMS

Index

Sr.No.	Practical Name
1	Install and set up MySQL. Create a database and table for employee details.
2	Create a table for storing student information and perform basic operations.
3	Create a table with various data types and perform queries.
4	Create a table with constraints like Primary Key, Foreign Key, and Unique.
5	Create a table with integrity constraints like NOT NULL, CHECK, and DEFAULT.
6	Use DDL and DML commands to create tables and manipulate data.
7	Create a Sales table and use aggregate functions to summarize sales data.
8	Perform JOIN operations on Customers and Orders tables.

Practical 1

Aim:

Install and set up MySQL. Create a database and a table to store employee details. Perform basic operations like INSERT and DELETE.

Code:

-- Step 1: Create a database

```
CREATE DATABASE EmployeeDB;
```

-- Step 2: Use the database

```
USE EmployeeDB;
```

-- Step 3: Create a table for employee details

```
CREATE TABLE Employees (  
    EmployeeID INT PRIMARY KEY,  
    Name VARCHAR(100),  
    Department VARCHAR(100)  
);
```

-- Step 4: Insert data into the table

```
INSERT INTO Employees (EmployeeID, Name, Department)  
VALUES (1, 'Aditya', 'HR'),  
       (2, 'Vishnu', 'Finance');
```

-- Step 5: Delete a record from the table

```
DELETE FROM Employees WHERE EmployeeID = 2;
```

-- Step 6: View the table

```
SELECT * FROM Employees;
```

Screenshots of Output:

The screenshot displays the MySQL Workbench interface. The left sidebar contains navigation panels for Management, Instance, and Performance. The central pane shows a SQL query editor with the following code:

```
6 Name VARCHAR(100),
7 Department VARCHAR(100)
8 );
9
10 • INSERT INTO Employees (EmployeeID, Name, Department)
11 VALUES (1, 'Aditya', 'HR'),
12 (2, 'Vishnu', 'Finance');
13
14 • DELETE FROM Employees WHERE EmployeeID = 2;
15
16 -- Step 6: View the table
17 • SELECT * FROM Employees;
18
```

Below the query editor, the 'Result Grid' shows the output of the last query:

EmployeeID	Name	Department
1	Aditya	HR

The bottom pane shows the 'Action Output' log, which records the execution of the SQL statements:

#	Time	Action	Message	Duration / Fetch
1	21:34:42	CREATE DATABASE EmployeeDB	1 row(s) affected	0.016 sec
2	21:34:44	USE EmployeeDB	0 row(s) affected	0.000 sec
3	21:34:46	CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, Name VARCHAR(100), Department VA...	0 row(s) affected	0.031 sec
4	21:34:50	INSERT INTO Employees (EmployeeID, Name, Department) VALUES (1, 'Aditya', 'HR'), (2, 'Vishnu', 'Finance')	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.047 sec
5	21:34:54	DELETE FROM Employees WHERE EmployeeID = 2	1 row(s) affected	0.015 sec
6	21:34:59	SELECT * FROM Employees LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Practical 2

Aim:

Create a table for storing student information. Insert sample data and perform basic operations: INSERT, UPDATE, DELETE, and SELECT.

Code:

-- Step 1: Create a table for student information

```
CREATE TABLE Students (  
    StudentID INT PRIMARY KEY,  
    Name VARCHAR(100),  
    Age INT,  
    Grade CHAR(2)  
);
```

-- Step 2: Insert sample data

```
INSERT INTO Students (StudentID, Name, Age, Grade)  
VALUES (1, 'Aditya', 18, 'A'),  
      (2, 'Urmi', 17, 'B');
```

-- Step 3: Update a record

```
UPDATE Students SET Grade = 'A+' WHERE StudentID = 1;
```

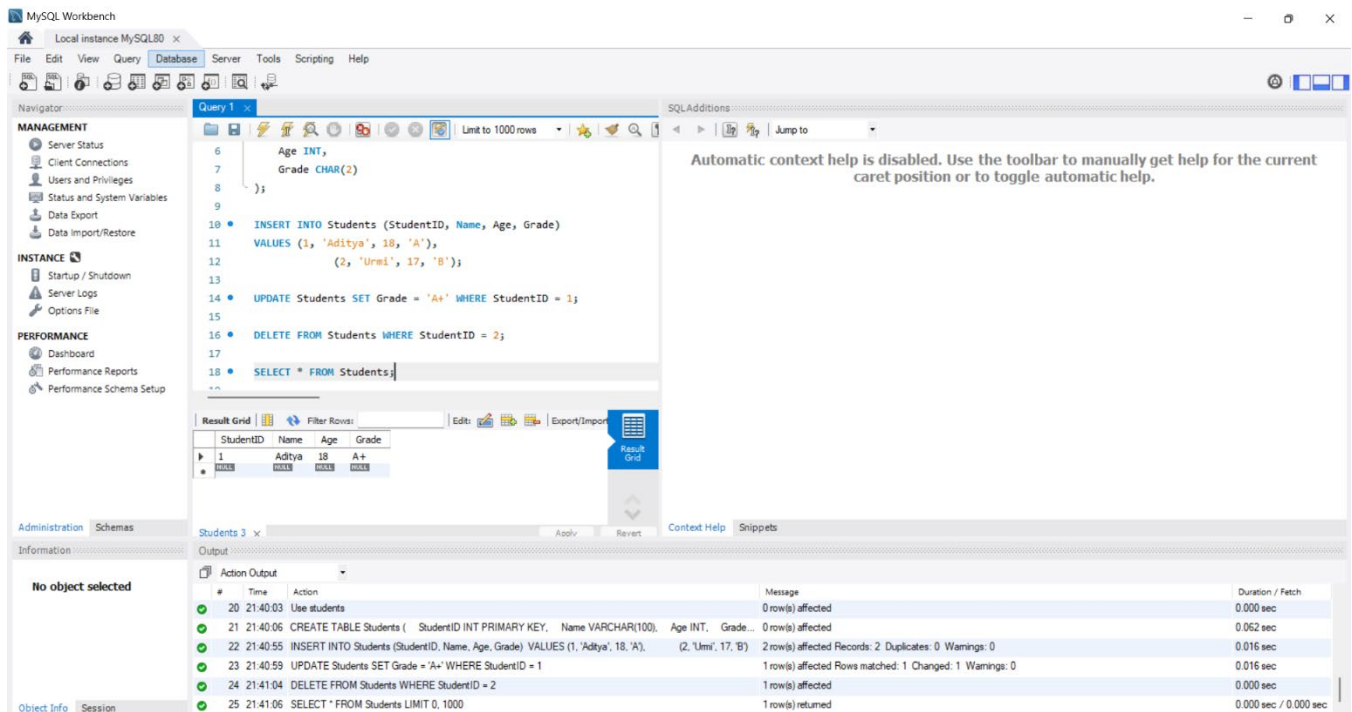
-- Step 4: Delete a record

```
DELETE FROM Students WHERE StudentID = 2;
```

-- Step 5: Retrieve all records

```
SELECT * FROM Students;
```

Screenshots of Output:



The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'MANAGEMENT' and 'PERFORMANCE' tabs. The main window is divided into three panes: the top pane shows the SQL editor with the following queries:

```
6 Age INT,  
7 Grade CHAR(2)  
8 );  
9  
10 INSERT INTO Students (StudentID, Name, Age, Grade)  
11 VALUES (1, 'Aditya', 18, 'A'),  
12 (2, 'Urmi', 17, 'B');  
13  
14 UPDATE Students SET Grade = 'A+' WHERE StudentID = 1;  
15  
16 DELETE FROM Students WHERE StudentID = 2;  
17  
18 SELECT * FROM Students;
```

The middle pane shows the 'Result Grid' with the following data:

StudentID	Name	Age	Grade
1	Aditya	18	A+

The bottom pane shows the 'Output' tab with the following execution results:

#	Time	Action	Message	Duration / Fetch
20	21:40:03	Use students	0 row(s) affected	0.000 sec
21	21:40:06	CREATE TABLE Students (StudentID INT PRIMARY KEY, Name VARCHAR(100), Age INT, Grade...	0 row(s) affected	0.062 sec
22	21:40:55	INSERT INTO Students (StudentID, Name, Age, Grade) VALUES (1, 'Aditya', 18, 'A'), (2, 'Urmi', 17, 'B');	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.016 sec
23	21:40:59	UPDATE Students SET Grade = 'A+' WHERE StudentID = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
24	21:41:04	DELETE FROM Students WHERE StudentID = 2	1 row(s) affected	0.000 sec
25	21:41:06	SELECT * FROM Students LIMIT 0, 1000	1 row(s) returned	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 a table with various data types

```
CREATE TABLE EmployeeDetails (  
    EmployeeID INT PRIMARY KEY,  
    Name VARCHAR(100),  
    Salary DECIMAL(10, 2),  
    JoiningDate DATE,  
    ActiveStatus BOOLEAN  
);
```

-- Step 2: Insert sample data

```
INSERT INTO EmployeeDetails (EmployeeID, Name, Salary, JoiningDate,  
ActiveStatus)  
VALUES (1, 'Michael Jordan', 50000.00, '2023-01-15', TRUE),  
      (2, 'Aditya', 60000.00, '2022-06-20', FALSE);
```

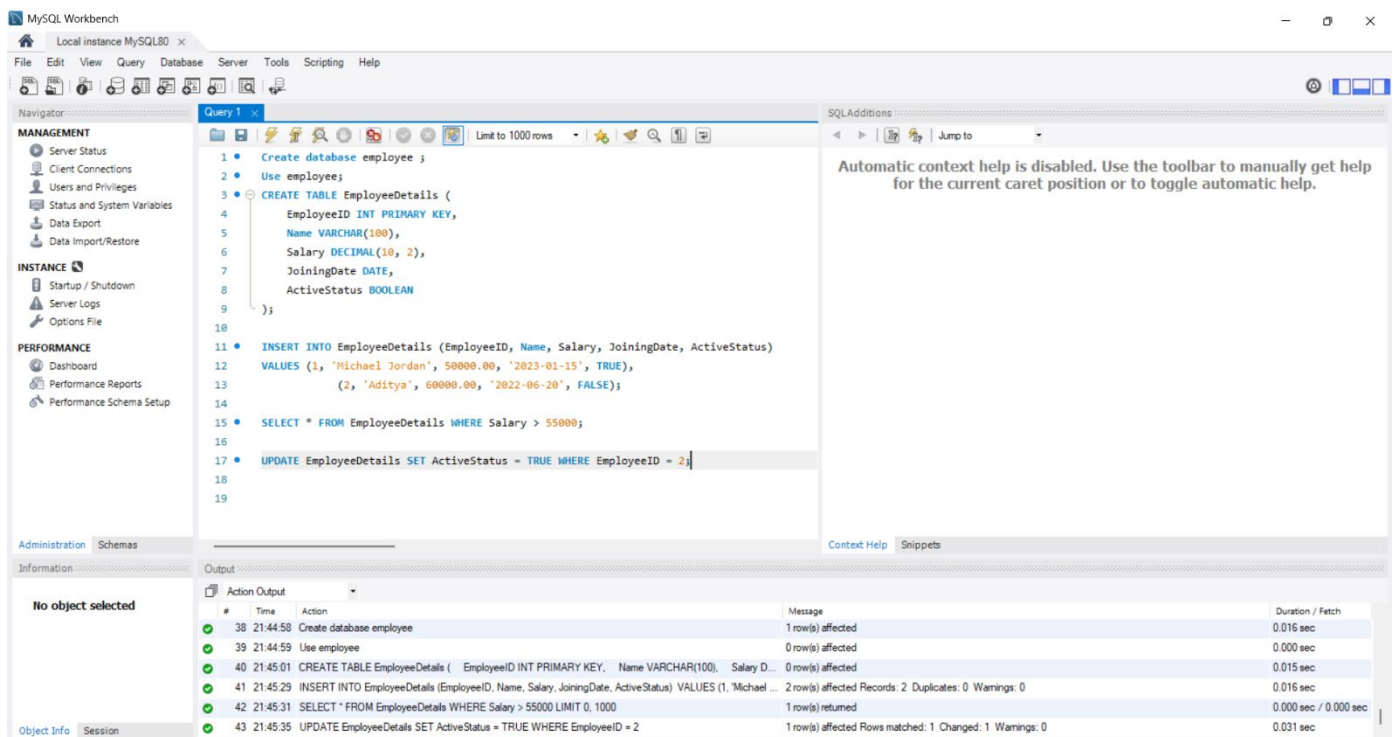
-- Step 3: Retrieve employees with salary > 55000

```
SELECT * FROM EmployeeDetails WHERE Salary > 55000;
```

-- Step 4: Update ActiveStatus

```
UPDATE EmployeeDetails SET ActiveStatus = TRUE WHERE EmployeeID = 2;
```

Screenshots of Output:



The screenshot displays the MySQL Workbench interface. The central pane shows a SQL script with the following queries:

```
1 • Create database employee ;
2 • Use employee;
3 • CREATE TABLE EmployeeDetails (
4     EmployeeID INT PRIMARY KEY,
5     Name VARCHAR(100),
6     Salary DECIMAL(10, 2),
7     JoiningDate DATE,
8     ActiveStatus BOOLEAN
9 );
10
11 • INSERT INTO EmployeeDetails (EmployeeID, Name, Salary, JoiningDate, ActiveStatus)
12 VALUES (1, 'Michael Jordan', 50000.00, '2023-01-15', TRUE),
13         (2, 'Aditya', 60000.00, '2022-06-20', FALSE);
14
15 • SELECT * FROM EmployeeDetails WHERE Salary > 55000;
16
17 • UPDATE EmployeeDetails SET ActiveStatus = TRUE WHERE EmployeeID = 2;
18
19
```

The right pane shows a message: "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 "Action Output" table, which lists the execution results of the queries:

#	Time	Action	Message	Duration / Fetch
38	21:44:58	Create database employee	1 row(s) affected	0.016 sec
39	21:44:59	Use employee	0 row(s) affected	0.000 sec
40	21:45:01	CREATE TABLE EmployeeDetails (EmployeeID INT PRIMARY KEY, Name VARCHAR(100), Salary D...	0 row(s) affected	0.015 sec
41	21:45:29	INSERT INTO EmployeeDetails (EmployeeID, Name, Salary, JoiningDate, ActiveStatus) VALUES (1, 'Michael ...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.016 sec
42	21:45:31	SELECT * FROM EmployeeDetails WHERE Salary > 55000 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
43	21:45:35	UPDATE EmployeeDetails SET ActiveStatus = TRUE WHERE EmployeeID = 2	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.031 sec

Practical 4

Aim:

Create a table to store employee information with constraints like Primary Key, Foreign Key, and Unique. Insert valid and invalid data to test the constraints.

Code:

-- Step 1: Create a department table

```
CREATE TABLE Departments (  
    DepartmentID INT PRIMARY KEY,  
    DepartmentName VARCHAR(100) UNIQUE  
);
```

-- Step 2: Create an employee table with foreign key

```
CREATE TABLE Employees (  
    EmployeeID INT PRIMARY KEY,  
    Name VARCHAR(100),  
    DepartmentID INT,  
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)  
);
```

-- Step 3: Insert valid data

```
INSERT INTO Departments (DepartmentID, DepartmentName) VALUES (1, 'HR'), (2, 'Finance');
```

```
INSERT INTO Employees (EmployeeID, Name, DepartmentID) VALUES (1, 'Aditya', 1);
```

-- Step 4: Insert invalid data (test constraints)

-- This will fail due to UNIQUE constraint

```
INSERT INTO Departments (DepartmentID, DepartmentName) VALUES (3, 'HR');
```

-- This will fail due to FOREIGN KEY constraint

```
INSERT INTO Employees (EmployeeID, Name, DepartmentID) VALUES (2, 'Ram', 5);
```

Screenshots of Output:

The screenshot displays the MySQL Workbench interface. The 'Query' tab is active, showing a series of SQL statements. The 'Output' pane at the bottom shows the execution results of these queries. The first three queries (lines 1-14) are successful: creating a database, using it, and creating two tables with constraints. The fourth query (line 15) successfully inserts two rows into the Departments table. The fifth query (line 16) successfully inserts one row into the Employees table. The sixth query (line 17) fails with a 'Duplicate entry' error for the 'HR' department name, which is expected due to the UNIQUE constraint. The seventh query (line 18) fails with a 'Foreign key constraint fails' error because the DepartmentID '5' does not exist in the Departments table. The eighth query (line 19) is not executed due to the previous error.

```
1 Create database employee ;
2 Use employee;
3 CREATE TABLE Departments (
4     DepartmentID INT PRIMARY KEY,
5     DepartmentName VARCHAR(100) UNIQUE
6 );
7
8 CREATE TABLE Employees (
9     EmployeeID INT PRIMARY KEY,
10    Name VARCHAR(100),
11    DepartmentID INT,
12    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
13 );
14
15 INSERT INTO Departments (DepartmentID, DepartmentName) VALUES (1, 'HR'), (2, 'Finance');
16 INSERT INTO Employees (EmployeeID, Name, DepartmentID) VALUES (1, 'Aditya', 1);
17 INSERT INTO Departments (DepartmentID, DepartmentName) VALUES (3, 'HR');
18 INSERT INTO Employees (EmployeeID, Name, DepartmentID) VALUES (2, 'Ram', 5);
19
```

#	Time	Action	Message	Duration / Fetch
6	21:50:41	CREATE TABLE Departments (DepartmentID INT PRIMARY KEY, DepartmentName VARCHAR(100) UN...	0 row(s) affected	0.016 sec
7	21:50:43	CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, Name VARCHAR(100), DepartmentID L...	0 row(s) affected	0.031 sec
8	21:50:46	INSERT INTO Departments (DepartmentID, DepartmentName) VALUES (1, 'HR'), (2, 'Finance')	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
9	21:50:47	INSERT INTO Employees (EmployeeID, Name, DepartmentID) VALUES (1, 'Aditya', 1)	1 row(s) affected	0.016 sec
10	21:50:48	INSERT INTO Departments (DepartmentID, DepartmentName) VALUES (3, 'HR')	Error Code: 1062. Duplicate entry 'HR' for key 'departments.DepartmentName'	0.015 sec
11	21:50:49	INSERT INTO Employees (EmployeeID, Name, DepartmentID) VALUES (2, 'Ram', 5)	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('employee`.`employees`.`CON...	0.015 sec

Practical 5

Aim:

Create a table for Customer details with integrity constraints like NOT NULL, CHECK, and DEFAULT. Insert valid and invalid data to test these constraints.

Code:

-- Step 1: Create a table with constraints

```
CREATE TABLE Customers (  
    CustomerID INT PRIMARY KEY,  
    Name VARCHAR(100) NOT NULL,  
    Age INT CHECK (Age >= 18),  
    MembershipType VARCHAR(50) DEFAULT 'Basic'  
);
```

-- Step 2: Insert valid data

```
INSERT INTO Customers (CustomerID, Name, Age) VALUES (1, 'Aditya Mhaismale',  
25);
```

-- Step 3: Insert invalid data (test constraints)

-- This will fail due to NOT NULL constraint

```
INSERT INTO Customers (CustomerID, Age) VALUES (2, 30);
```

-- This will fail due to CHECK constraint

INSERT INTO Customers (CustomerID, Name, Age) VALUES (3, 'John doe', 17);

Screenshots of Output:

The screenshot displays the MySQL Workbench interface. The central pane shows a SQL script with the following queries:

```
1 create database shop;
2 use shop;
3 CREATE TABLE Customers (
4     CustomerID INT PRIMARY KEY,
5     Name VARCHAR(100) NOT NULL,
6     Age INT CHECK (Age >= 18),
7     MembershipType VARCHAR(50) DEFAULT 'Basic'
8 );
9
10 INSERT INTO Customers (CustomerID, Name, Age) VALUES (1, 'Aditya Mhaismale', 25);
11 INSERT INTO Customers (CustomerID, Age) VALUES (2, 30);
12 INSERT INTO Customers (CustomerID, Name, Age) VALUES (3, 'John doe', 17);
13
```

The bottom pane shows the 'Action Output' tab with the following results:

#	Time	Action	Message	Duration / Fetch
20	21:55:00	create database shop	1 row(s) affected	0.016 sec
21	21:55:09	use shop	0 row(s) affected	0.000 sec
22	21:55:12	CREATE TABLE Customers (CustomerID INT PRIMARY KEY, Name VARCHAR(100) NOT NULL, Ag...	0 row(s) affected	0.031 sec
23	21:55:40	INSERT INTO Customers (CustomerID, Name, Age) VALUES (1, 'Aditya Mhaismale', 25)	1 row(s) affected	0.000 sec
24	21:55:41	INSERT INTO Customers (CustomerID, Age) VALUES (2, 30)	Error Code: 1364 Field 'Name' doesn't have a default value	0.000 sec
25	21:55:44	INSERT INTO Customers (CustomerID, Name, Age) VALUES (3, 'John doe', 17)	Error Code: 3819 Check constraint 'customers_chk_1' is violated.	0.000 sec

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:

-- Step 1: Create a table

```
CREATE TABLE Products (  
    ProductID INT PRIMARY KEY,  
    ProductName VARCHAR(100),  
    Price DECIMAL(10, 2)  
);
```

-- Step 2: Insert data

```
INSERT INTO Products (ProductID, ProductName, Price)  
VALUES (1, 'Laptop', 1000.00), (2, 'Smartphone', 500.00);
```

-- Step 3: Update data

```
UPDATE Products SET Price = 1200.00 WHERE ProductID = 1;
```

-- Step 4: Delete data

```
DELETE FROM Products WHERE ProductID = 2;
```

-- Step 5: Retrieve data

SELECT * FROM Products;

Screenshots of Output:

The screenshot displays the MySQL Workbench interface. The central query editor contains the following SQL script:

```
3 CREATE TABLE Products (  
4   ProductID INT PRIMARY KEY,  
5   ProductName VARCHAR(100),  
6   Price DECIMAL(10, 2)  
7 );  
8  
9 INSERT INTO Products (ProductID, ProductName, Price)  
10 VALUES (1, 'Laptop', 1000.00), (2, 'Smartphone', 500.00);  
11  
12  
13 UPDATE Products SET Price = 1200.00 WHERE ProductID = 1;  
14 DELETE FROM Products WHERE ProductID = 2;  
15 SELECT * FROM Products;
```

Below the query editor, the 'Result Grid' shows the output of the final query:

ProductID	ProductName	Price
1	Laptop	1200.00

The bottom panel shows the 'Output' tab with the 'Action Output' view. It lists the execution steps and their results:

#	Time	Action	Message	Duration / Fetch
35	21:58:23	use shop	0 row(s) affected	0.000 sec
36	21:58:25	CREATE TABLE Products (ProductID INT PRIMARY KEY, ProductName VARCHAR(100), Price DEC...	0 row(s) affected	0.032 sec
37	21:58:27	INSERT INTO Products (ProductID, ProductName, Price) VALUES (1, 'Laptop', 1000.00), (2, 'Smartphone', 5...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
38	21:58:29	UPDATE Products SET Price = 1200.00 WHERE ProductID = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
39	21:58:30	DELETE FROM Products WHERE ProductID = 2	1 row(s) affected	0.000 sec
40	21:58:31	SELECT * FROM Products LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

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:

-- Step 1: Create a Sales table

```
CREATE TABLE Sales (  
    SaleID INT PRIMARY KEY,  
    ProductName VARCHAR(100),  
    Quantity INT,  
    Price DECIMAL(10, 2)  
);
```

-- Step 2: Insert data

```
INSERT INTO Sales (SaleID, ProductName, Quantity, Price)  
VALUES (1, 'Laptop', 5, 1000.00),  
      (2, 'Smartphone', 10, 500.00);
```

-- Step 3: Use aggregate functions

```
SELECT COUNT(*) AS TotalSales FROM Sales;  
  
SELECT SUM(Quantity * Price) AS TotalRevenue FROM Sales;
```

SELECT AVG(Price) AS AveragePrice FROM Sales;

SELECT MIN(Price) AS MinimumPrice FROM Sales;

SELECT MAX(Price) AS MaximumPrice FROM Sales;

Screenshots of Output:

The screenshot displays the MySQL Workbench interface. The central query editor contains the following SQL code:

```
8 Price DECIMAL(10, 2)
9 );
10
11
12 INSERT INTO Sales (SaleID, ProductName, Quantity, Price)
13 VALUES (1, 'Laptop', 5, 1000.00),
14 (2, 'Smartphone', 10, 500.00);
15
16 SELECT COUNT(*) AS TotalSales FROM Sales;
17 SELECT SUM(Quantity * Price) AS TotalRevenue FROM Sales;
18 SELECT AVG(Price) AS AveragePrice FROM Sales;
19 SELECT MIN(Price) AS MinimumPrice FROM Sales;
20 SELECT MAX(Price) AS MaximumPrice FROM Sales;
```

The 'Result Grid' at the bottom shows the output for the 'AveragePrice' query, displaying a single value: 750.000000.

The 'Output' pane at the bottom shows the execution log for the queries:

#	Time	Action	Message	Duration / Fetch
46	22:00:27	SELECT COUNT(*) AS TotalSales FROM Sales LIMIT 0.1000	1 row(s) returned	0.000 sec / 0.000 sec
47	22:00:29	SELECT SUM(Quantity * Price) AS TotalRevenue FROM Sales LIMIT 0.1000	1 row(s) returned	0.000 sec / 0.000 sec
48	22:00:32	SELECT AVG(Price) AS AveragePrice FROM Sales LIMIT 0.1000	1 row(s) returned	0.000 sec / 0.000 sec
49	22:00:34	SELECT MIN(Price) AS MinimumPrice FROM Sales LIMIT 0.1000	1 row(s) returned	0.015 sec / 0.000 sec
50	22:00:36	SELECT MAX(Price) AS MaximumPrice FROM Sales LIMIT 0.1000	1 row(s) returned	0.000 sec / 0.000 sec
51	22:00:39	SELECT AVG(Price) AS AveragePrice FROM Sales LIMIT 0.1000	1 row(s) returned	0.000 sec / 0.000 sec

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:

-- Step 1: Create Customers and Orders tables

```
CREATE TABLE Customers (
```

```
    CustomerID INT PRIMARY KEY,
```

```
    Name VARCHAR(100)
```

```
);
```

```
CREATE TABLE Orders (
```

```
    OrderID INT PRIMARY KEY,
```

```
    CustomerID INT,
```

```
    OrderAmount DECIMAL(10, 2),
```

```
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
```

```
);
```

-- Step 2: Insert data

```
INSERT INTO Customers (CustomerID, Name) VALUES (1, 'Aditya'), (2, 'Bob');
```

```
INSERT INTO Orders (OrderID, CustomerID, OrderAmount) VALUES (101, 1, 500.00), (102, 2, 1000.00);
```

-- Step 3: Perform JOIN operations

-- INNER JOIN

```
SELECT c.Name, o.OrderAmount
```

```
FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID;
```

-- LEFT JOIN

```
SELECT c.Name, o.OrderAmount
```

```
FROM Customers c LEFT JOIN Orders o ON c.CustomerID = o.CustomerID;
```

-- RIGHT JOIN

```
SELECT c.Name, o.OrderAmount
```

```
FROM Customers c RIGHT JOIN Orders o ON c.CustomerID = o.CustomerID;
```

Screenshots of Output:

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'MANAGEMENT' and 'PERFORMANCE' tabs. The main window shows a SQL query editor with the following queries:

```
15
16 INSERT INTO Customers (CustomerID, Name) VALUES (1, 'Aditya'), (2, 'Bob');
17 INSERT INTO Orders (OrderID, CustomerID, OrderAmount) VALUES (101, 1, 500.00), (102, 2, 1000.00);
18
19 SELECT c.Name, o.OrderAmount
20 FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID;
21
22 SELECT c.Name, o.OrderAmount
23 FROM Customers c LEFT JOIN Orders o ON c.CustomerID = o.CustomerID;
24
25 SELECT c.Name, o.OrderAmount
26 FROM Customers c RIGHT JOIN Orders o ON c.CustomerID = o.CustomerID;
27
```

The 'Result Grid' shows the output of the first query (INNER JOIN):

Name	OrderAmount
Aditya	500.00
Bob	1000.00

The 'Action Output' tab at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
53	22:02:31	CREATE TABLE Orders (OrderID INT PRIMARY KEY, CustomerID INT, OrderAmount DECIMAL(10, 2);	0 row(s) affected	0.047 sec
54	22:02:50	INSERT INTO Customers (CustomerID, Name) VALUES (1, 'Aditya'), (2, 'Bob');	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.015 sec
55	22:03:00	INSERT INTO Orders (OrderID, CustomerID, OrderAmount) VALUES (101, 1, 500.00), (102, 2, 1000.00);	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.016 sec
56	22:03:13	SELECT c.Name, o.OrderAmount FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID;	2 row(s) returned	0.000 sec / 0.000 sec
57	22:03:22	SELECT c.Name, o.OrderAmount FROM Customers c LEFT JOIN Orders o ON c.CustomerID = o.CustomerID;	2 row(s) returned	0.000 sec / 0.000 sec
58	22:03:29	SELECT c.Name, o.OrderAmount FROM Customers c RIGHT JOIN Orders o ON c.CustomerID = o.CustomerID;	2 row(s) returned	0.000 sec / 0.000 sec