

Name: Sanket Jadhav

PRN: 2124UCSM1126

Dept: Cyber Security

Practical No.: - 1

Install and set up MySQL.

Aim: - To create a database called company2 and define an employees table with appropriate attributes like id, name, position, department, salary, and join_date. Insert sample data and fetch records using SQL queries.

Code: -

```
create database company2;
use company2;
create table employees(
    id int auto_increment primary key,
    name varchar(50) not null,
    position varchar(50) not null,
    department varchar(50),
    salary decimal(10,2),
    join_date date
);
select * from employees;
insert into employees(name, position, department, salary, join_date)
values
('sanket','Student','Cyber','1000','2025-01-15'),
('Sanskart','Student','Cyber','2000','2025-01-15'),
('Kartik','Student','Cyber','3000','2025-01-15');
select * from employees;
```

Screenshot of Output: -

The screenshot shows a MySQL query editor on OneCompiler. The code creates a table 'employees' with columns id, name, position, department, salary, and join_date. It then inserts three rows of data: ('Sanket', 'Student', 'Cyber', '1000', '2025-01-15'), ('Sanskar', 'Student', 'Cyber', '2000', '2025-01-15'), and ('Kartik', 'Student', 'Cyber', '3000', '2025-01-15'). Finally, it selects all data from the employees table. The output displays the inserted data in a tabular format.

```
1+ create table employees(
2+     id int auto_increment primary key,
3+     name text(10),
4+     position text(10),
5+     department text(10),
6+     salary int,
7+     join_date text
8+ );
9 select * from employees;
10 insert into employees(name, position, department, salary,
11 values
12 ('Sanket', 'Student', 'Cyber', '1000', '2025-01-15'),
13 ('Sanskar', 'Student', 'Cyber', '2000', '2025-01-15'),
14 ('Kartik', 'Student', 'Cyber', '3000', '2025-01-15');
15 select * from employees;
16
```

Output:

id	name	position	department	salary	join_date
1	Sanket	Student	Cyber	1000	2025-01-15
2	Sanskar	Student	Cyber	2000	2025-01-15
3	Kartik	Student	Cyber	3000	2025-01-15

Practical No.: - 2

Aim: - To establish a database called Employee and make an Employee table with PRIMARY KEY, NOT NULL, CHECK, and DEFAULT constraints. The table will have employee information such as EmployeeID, Name, Salary, JoiningDate, and ActiveStatus. Insert data and retrieve it using SQL queries as well.

Code: -

```
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 );
select * from Employee;
```

```
insert into Employee(Name,Salary,JoiningDate,ActiveStatus)
values
```

```
('Sanskar',200000.00, '2024-10-10' ,true),
```

```
('Kartik',100000.00, '2024-02-16' ,false),  
('Sanket',2000000.00, '2025-04-24' ,true);  
select * from Employee;
```

Screenshot of Output: -

The screenshot shows a MySQL terminal window with the following details:

- Title Bar:** queries.sql + 43eg57wq2
- Toolbar:** AI, NEW, MYSQL, RUN
- Input Area:** Shows the SQL code:

```
1+ create table Employee(  
2 EmployeeID int primary key auto_increment,  
3 Name varchar(100) not null,  
4 salary decimal(10,2) check (salary>0),  
5 JoiningDate date not null,  
6 ActiveStatus Boolean default true );  
7 select * from Employee;  
8  
9 insert into Employee(Name,Salary,JoiningDate,ActiveStatus)  
10 values  
11 ('Sanskar',200000.00, '2024-10-10' ,true),  
12 ('Kartik',100000.00, '2024-02-16' ,false),  
13 ('Sanket',2000000.00, '2025-04-24' ,true);  
14 select * from Employee;  
15
```
- Output Area:** Shows the resulting table:

EmployeeID	Name	salary	JoiningDate	ActiveStatus
1	Sanskar	200000.00	2024-10-10	1
2	Kartik	100000.00	2024-02-16	0
3	Sanket	2000000.00	2025-04-24	1

Practical No.: - 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: -

```
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  
);  
select * from Employee;  
INSERT INTO Employee(Name,Salary,JoiningDate,ActiveStatus)  
VALUES  
('Kartik Pawar',55000.00,'2023-06-15',TRUE),
```

```
('Sanket Jadhav',60000.00,'2022-06-15',TRUE),  
('Sanskar Gangurde',70000.00,'2021-06-15',FALSE),  
('Vitthal Nath',1000000.00,'2020-06-15',TRUE);  
select * from Employee;  
SELECT EmployeeID, Name,Salary FROM Employee WHERE  
ActiveStatus = TRUE;  
select * from Employee;  
UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID =  
2;  
select * from Employee;  
UPDATE Employee SET ActiveStatus = FALSE WHERE EmployeeID =  
4;  
select * from Employee;  
DELETE FROM Employee WHERE EmployeeID = 3;  
select * from Employee;  
SELECT * FROM Employee WHERE YEAR(JoiningDate) = 2023;  
select * from Employee;  
SELECT Name, Salary FROM Employee WHERE Salary > 60000;  
select * from Employee;  
SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS  
LowestSalary FROM Employee;  
select * from Employee;  
SELECT * FROM Employee ORDER BY Salary DESC LIMIT 3;  
select * from Employee;
```

Screenshot of Output: -

The screenshot shows the MySQL Workbench interface with a script editor containing a SQL script and two result panes.

Script Content:

```
1 - CREATE TABLE Employee(
2 EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
3 Name VARCHAR(100) NOT NULL,
4 Salary DECIMAL(10,2) CHECK (Salary > 0),
5 JoiningDate DATE NOT NULL,
6 ActiveStatus BOOLEAN DEFAULT TRUE
7 );
8 select * from Employee;
9 INSERT INTO Employee(Name,Salary, JoiningDate,ActiveStatus)
10 VALUES
11 ('Kartik Pawar',55000.00,'2023-06-15',TRUE),
12 ('Sanket Jadhav',60000.00,'2022-06-15',TRUE),
13 ('Sanskar Gangurde',70000.00,'2021-06-15',FALSE),
14 ('Vitthal Nath',100000.00,'2020-06-15',TRUE);
15 select * from Employee;
16 SELECT EmployeeID, Name,Salary FROM Employee WHERE
17 ActiveStatus = TRUE;
18 select * from Employee;
19 UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID =
20 2;
21 select * from Employee;
22 UPDATE Employee SET ActiveStatus = FALSE WHERE EmployeeID =
23 4;
24 select * from Employee;
25 DELETE FROM Employee WHERE EmployeeID = 3;
26 select * from Employee;
27 SELECT * FROM Employee WHERE YEAR(JoiningDate) = 2023;
28 select * from Employee;
29 SELECT Name, Salary FROM Employee WHERE Salary > 60000;
30 select * from Employee;
31 SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS
LowestSalary FROM Employee;
32 select * from Employee;
33 SELECT * FROM Employee ORDER BY Salary DESC LIMIT 3;
34 select * from Employee;
35
36
```

Output:

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Kartik Pawar	55000.00	2023-06-15	1
2	Sanket Jadhav	60000.00	2022-06-15	1
3	Sanskar Gangurde	70000.00	2021-06-15	0
4	Vitthal Nath	100000.00	2020-06-15	1

EmployeeID	Name	Salary
1	Kartik Pawar	55000.00
2	Sanket Jadhav	60000.00
4	Vitthal Nath	100000.00

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Kartik Pawar	55000.00	2023-06-15	1
2	Sanket Jadhav	60000.00	2022-06-15	1
3	Sanskar Gangurde	70000.00	2021-06-15	0
4	Vitthal Nath	100000.00	2020-06-15	1

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

The screenshot shows the MySQL Workbench interface with a script editor containing the same SQL script and two result panes.

Script Content:

```
1 - CREATE TABLE Employee(
2 EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
3 Name VARCHAR(100) NOT NULL,
4 Salary DECIMAL(10,2) CHECK (Salary > 0),
5 JoiningDate DATE NOT NULL,
6 ActiveStatus BOOLEAN DEFAULT TRUE
7 );
8 select * from Employee;
9 INSERT INTO Employee(Name,Salary, JoiningDate,ActiveStatus)
10 VALUES
11 ('Kartik Pawar',55000.00,'2023-06-15',TRUE),
12 ('Sanket Jadhav',60000.00,'2022-06-15',TRUE),
13 ('Sanskar Gangurde',70000.00,'2021-06-15',FALSE),
14 ('Vitthal Nath',100000.00,'2020-06-15',TRUE);
15 select * from Employee;
16 SELECT EmployeeID, Name,Salary FROM Employee WHERE
17 ActiveStatus = TRUE;
18 select * from Employee;
19 UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID =
20 2;
21 select * from Employee;
22 UPDATE Employee SET ActiveStatus = FALSE WHERE EmployeeID =
23 4;
24 select * from Employee;
25 DELETE FROM Employee WHERE EmployeeID = 3;
26 select * from Employee;
27 SELECT * FROM Employee WHERE YEAR(JoiningDate) = 2023;
28 select * from Employee;
29 SELECT Name, Salary FROM Employee WHERE Salary > 60000;
30 select * from Employee;
31 SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS
LowestSalary FROM Employee;
32 select * from Employee;
33 SELECT * FROM Employee ORDER BY Salary DESC LIMIT 3;
34 select * from Employee;
35
36
```

Output:

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Kartik Pawar	55000.00	2023-06-15	1
2	Sanket Jadhav	66000.00	2022-06-15	1
3	Sanskar Gangurde	70000.00	2021-06-15	0
4	Vitthal Nath	100000.00	2020-06-15	1

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Kartik Pawar	55000.00	2023-06-15	1
2	Sanket Jadhav	66000.00	2022-06-15	1
3	Sanskar Gangurde	70000.00	2021-06-15	0
4	Vitthal Nath	100000.00	2020-06-15	0

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Kartik Pawar	55000.00	2023-06-15	1
2	Sanket Jadhav	66000.00	2022-06-15	1
4	Vitthal Nath	100000.00	2020-06-15	0

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

queries.sql + 43eg57wq2

```

1- CREATE TABLE Employee(
2 EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
3 Name VARCHAR(100) NOT NULL,
4 Salary DECIMAL(10,2) CHECK (Salary > 0),
5 JoiningDate DATE NOT NULL,
6 ActiveStatus BOOLEAN DEFAULT TRUE
7 );
8 select * from Employee;
9 INSERT INTO Employee(Name,Salary,JoiningDate,ActiveStatus)
VALUES
10 ('Kartik Pawar',55000.00,'2023-06-15',TRUE),
11 ('Sanket Jadhav',60000.00,'2022-06-15',TRUE),
12 ('Sanskars Gangurde',70000.00,'2021-06-15',FALSE),
13 ('Vitthal Nath',100000.00,'2020-06-15',TRUE);
14 select * from Employee;
15 SELECT EmployeeID, Name,Salary FROM Employee WHERE
16 ActiveStatus = TRUE;
17 select * from Employee;
18 UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID =
20 2;
19 select * from Employee;
20 UPDATE Employee SET ActiveStatus = FALSE WHERE EmployeeID =
23 4;
21 select * from Employee;
22 DELETE FROM Employee WHERE EmployeeID = 3;
23 select * from Employee;
24 SELECT * FROM Employee WHERE YEAR(JoiningDate) = 2023;
25 select * from Employee;
26 SELECT Name, Salary FROM Employee WHERE Salary > 60000;
27 select * from Employee;
28 SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS
29 LowestSalary FROM Employee;
30 select * from Employee;
31 SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS
32 LowestSalary FROM Employee;
33 select * from Employee;
34 SELECT * FROM Employee ORDER BY Salary DESC LIMIT 3;
35 select * from Employee;
36

```

STDIN
Input for the program (Optional)

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Kartik Pawar	55000.00	2023-06-15	1
2	Sanket Jadhav	60000.00	2022-06-15	1
4	Vitthal Nath	100000.00	2020-06-15	0

Name	Salary
Sanket Jadhav	60000.00
Vitthal Nath	100000.00

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Kartik Pawar	55000.00	2023-06-15	1
2	Sanket Jadhav	60000.00	2022-06-15	1
4	Vitthal Nath	100000.00	2020-06-15	0

HighestSalary	LowestSalary
100000.00	55000.00

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
4	Vitthal Nath	100000.00	2020-06-15	0

queries.sql + 43eg57wq2

```

1- CREATE TABLE Employee(
2 EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
3 Name VARCHAR(100) NOT NULL,
4 Salary DECIMAL(10,2) CHECK (Salary > 0),
5 JoiningDate DATE NOT NULL,
6 ActiveStatus BOOLEAN DEFAULT TRUE
7 );
8 select * from Employee;
9 INSERT INTO Employee(Name,Salary,JoiningDate,ActiveStatus)
VALUES
10 ('Kartik Pawar',55000.00,'2023-06-15',TRUE),
11 ('Sanket Jadhav',60000.00,'2022-06-15',TRUE),
12 ('Sanskars Gangurde',70000.00,'2021-06-15',FALSE),
13 ('Vitthal Nath',100000.00,'2020-06-15',TRUE);
14 select * from Employee;
15 SELECT EmployeeID, Name,Salary FROM Employee WHERE
16 ActiveStatus = TRUE;
17 ActiveStatus = TRUE;
18 select * from Employee;
19 UPDATE Employee SET Salary = Salary * 1.10 WHERE EmployeeID =
20 2;
21 select * from Employee;
22 UPDATE Employee SET ActiveStatus = FALSE WHERE EmployeeID =
23 4;
24 select * from Employee;
25 DELETE FROM Employee WHERE EmployeeID = 3;
26 select * from Employee;
27 SELECT * FROM Employee WHERE YEAR(JoiningDate) = 2023;
28 select * from Employee;
29 SELECT Name, Salary FROM Employee WHERE Salary > 60000;
30 select * from Employee;
31 SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS
32 LowestSalary FROM Employee;
33 select * from Employee;
34 SELECT * FROM Employee ORDER BY Salary DESC LIMIT 3;
35 select * from Employee;
36

```

STDIN
Input for the program (Optional)

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
4	Vitthal Nath	100000.00	2020-06-15	0

HighestSalary	LowestSalary
100000.00	55000.00

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Kartik Pawar	55000.00	2023-06-15	1
2	Sanket Jadhav	60000.00	2022-06-15	1
4	Vitthal Nath	100000.00	2020-06-15	0

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
4	Vitthal Nath	100000.00	2020-06-15	0
2	Sanket Jadhav	60000.00	2022-06-15	1
1	Kartik Pawar	55000.00	2023-06-15	1

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
1	Kartik Pawar	55000.00	2023-06-15	1
2	Sanket Jadhav	60000.00	2022-06-15	1
4	Vitthal Nath	100000.00	2020-06-15	0

Practical No.: - 4

Creating Employee Table with Constraints

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

Code: -

```
CREATE TABLE Department (
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(50) UNIQUE
);
```

-- Create 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,
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);
```

-- Insert Valid Data into Department Table

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

-- Insert Valid Data into Employee Table

```
INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID)
VALUES (101, 'SANKET', 'sanket@example.com', 50000.00, 1);
```

```
INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID)
VALUES (102, 'SANSKAR', 'sanskar@example.com', 60000.00, 2);
```

-- Valid Insert: Ensure the email is unique

```
INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID)
VALUES (103, 'KARTIK', 'kartik@example.com', 45000.00, 2);
```

-- Select the Data from Employee Table to Verify

```
SELECT * FROM Employee;
```

Screenshot of Output: -

The screenshot shows a MySQL terminal window with the following details:

- File:** queries.sql
- Session ID:** 43eg57wq2
- Environment:** AI, NEW, MYSQL, RUN
- Output:** The terminal displays the results of the SELECT query, which retrieves three rows of data from the Employee table.

EmpID	Name	Email	Salary	DeptID
101	SAMKET	sanket@example.com	50000.00	1
102	SANSKAR	sanskars@example.com	60000.00	2
103	KARTIK	kartik@example.com	45000.00	2

Practical No.: - 5

Testing Employee Constraints

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

Code: -

```
CREATE TABLE Customer (
CustomerID INT PRIMARY KEY,
```

```
FirstName VARCHAR(100),  
LastName VARCHAR(100),  
Email VARCHAR(100),  
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, 'Sanskar', 'Gangurde', 'sanskar.gangurde@example.com',  
'1234567890', 25, TRUE);  
  
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone,  
Age)  
VALUES (2, 'Sanket', 'Jadhav', '@example.com', '0987654321', 30);  
  
select * from Customer; -- Insert Invalid Data to Test Constraints -- Invalid data  
for NOT NULL constraint (FirstName is NULL)  
  
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone,  
Age)  
VALUES (3, 'Kartik', 'Pawar', 'kartik.pawar@example.com', '5551234567', 20);  
  
select * from Customer; /* -- Invalid data for CHECK constraint (Age less than  
18)  
  
INSERT INTO Customer(CustomerID, FirstName, LastName, Email, Phone,  
Age)  
VALUES (4, 'sai', 'ram', 'alice.johnson@example.com', '6669876543', 16);*/  
  
select * from Customer; -- Invalid data for UNIQUE constraint (Duplicate  
Email)  
  
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone,  
Age)  
VALUES (5, 'Bob', 'Brown', 'john.doe@example.com', '7771234567', 28);
```

```
select * from Customer;
```

Screenshot of Output: -

```
queries.sql + 43eg57wq2 AI NEW MYSQL RUN STDIN Input for the program (Optional) Output:
```

```
+-----+-----+-----+-----+-----+
| CustomerID | FirstName | LastName | Email           | Phone      | Age | IsActive |
+-----+-----+-----+-----+-----+
| 1 | Sanskar  | Gangurde | sanskar.gangurde@example.com | 1234567890 | 25 | 1 |
| 2 | Sanket   | Jadhav    | @example.com          | 0987654321 | 30 | 1 |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| CustomerID | FirstName | LastName | Email           | Phone      | Age | IsActive |
+-----+-----+-----+-----+-----+
| 1 | Sanskar  | Gangurde | sanskar.gangurde@example.com | 1234567890 | 25 | 1 |
| 2 | Sanket   | Jadhav    | @example.com          | 0987654321 | 30 | 1 |
| 3 | Kartik   | Pawar     | kartik.pawar@example.com | 5551234567  | 20 | 1 |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| CustomerID | FirstName | LastName | Email           | Phone      | Age | IsActive |
+-----+-----+-----+-----+-----+
| 1 | Sanskar  | Gangurde | sanskar.gangurde@example.com | 1234567890 | 25 | 1 |
| 2 | Sanket   | Jadhav    | @example.com          | 0987654321 | 30 | 1 |
| 3 | Kartik   | Pawar     | kartik.pawar@example.com | 5551234567  | 20 | 1 |
| 5 | Bob       | Brown     | john.doe@example.com   | 7771234567  | 28 | 1 |
+-----+-----+-----+-----+-----+
```

Practical No.: - 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 TABLE Employees (
EmployeeID INT PRIMARY KEY,
FirstName VARCHAR(50),
LastName VARCHAR(50),
Age INT,
Department VARCHAR(50),
Salary DECIMAL(10, 2)
);
```

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age,  
Department,  
Salary)
```

```
VALUES (1, 'Kartik', 'Pawar', 28, 'HR', 50000.00);
```

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age,  
Department,
```

```
Salary)
```

```
VALUES (2, 'Sanket', 'Jadhav', 35, 'IT', 65000.00);
```

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age,  
Department,
```

```
Salary)
```

```
VALUES (3, 'Sanskar', 'Gangurde', 40, 'Finance', 75000.00);
```

```
select * from Employees;
```

```
UPDATE Employees
```

```
SET Salary = 70000.00
```

```
WHERE EmployeeID = 2;
```

```
select * from Employees;
```

```
select * from Employees;
```

```
UPDATE Employees
```

```
SET FirstName = 'Kartik', LastName = 'Pawar', Age = 45, Department =
```

```
'Management', Salary = 80000.00
```

```
WHERE EmployeeID = 3;
```

```
select * from Employees;
```

```
select * from Employees;  
select * from Employees;  
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;  
select * from Employees;  
DELETE FROM Employees  
WHERE EmployeeID = 1;  
SELECT * FROM Employees;  
SELECT * FROM Employees  
WHERE EmployeeID = 2;  
SELECT * FROM Employees  
WHERE EmployeeID = 1;  
select * from Employees;
```

Screenshot of Output: -

queries.sql

```

1 - CREATE TABLE Employees (
2   EmployeeID INT PRIMARY KEY,
3   FirstName VARCHAR(50),
4   LastName VARCHAR(50),
5   Age INT,
6   Department VARCHAR(50),
7   Salary DECIMAL(10, 2)
8 );
9 - INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department,
10   Salary)
11 VALUES (1, 'Kartik', 'Pawar', 28, 'HR', 50000.00);
12 - INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department,
13   Salary)
14 VALUES (2, 'Sanket', 'JadHAV', 35, 'IT', 65000.00);
15 - INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department,
16   Salary)
17 VALUES (3, 'SanskAR', 'Gangurde', 40, 'Finance', 75000.00);
18 select * from Employees;
19 UPDATE Employees
20 SET Salary = 70000.00
21 WHERE EmployeeID = 2;
22 select * from Employees;
23
24
25 select * from Employees;
26 UPDATE Employees
27 SET FirstName = 'Kartik', LastName = 'Pawar', Age = 45, Department =
28 'Management', Salary = 80000.00
29 WHERE EmployeeID = 3;
30 select * from Employees;
31
32
33
34 select * from Employees;
35 select * from Employees;
36 UPDATE Employees
37 SET Salary = CASE
38 WHEN Department = 'HR' THEN Salary * 1.05
39 WHEN Department = 'IT' THEN Salary * 1.08
40 WHEN Department = 'Finance' THEN Salary * 1.10
41 ELSE Salary
42 END;
43 select * from Employees;

```

Output:

EmployeeID	FirstName	LastName	Age	Department	Salary
1	Kartik	Pawar	28	HR	50000.00
2	Sanket	JadHAV	35	IT	65000.00
3	SanskAR	Gangurde	40	Finance	75000.00

EmployeeID	FirstName	LastName	Age	Department	Salary
1	Kartik	Pawar	28	HR	50000.00
2	Sanket	JadHAV	35	IT	70000.00
3	SanskAR	Gangurde	40	Finance	75000.00

EmployeeID	FirstName	LastName	Age	Department	Salary
1	Kartik	Pawar	28	HR	50000.00
2	Sanket	JadHAV	35	IT	70000.00
3	Kartik	Pawar	45	Management	80000.00

EmployeeID	FirstName	LastName	Age	Department	Salary
1	Kartik	Pawar	28	HR	52500.00
2	Sanket	JadHAV	35	IT	75600.00
3	Kartik	Pawar	45	Management	80000.00

EmployeeID	FirstName	LastName	Age	Department	Salary
2	Sanket	JadHAV	35	IT	75600.00

queries.sql

```

1 - CREATE TABLE Employees (
2   EmployeeID INT PRIMARY KEY,
3   FirstName VARCHAR(50),
4   LastName VARCHAR(50),
5   Age INT,
6   Department VARCHAR(50),
7   Salary DECIMAL(10, 2)
8 );
9 - INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department,
10   Salary)
11 VALUES (1, 'Kartik', 'Pawar', 28, 'HR', 50000.00);
12 - INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department,
13   Salary)
14 VALUES (2, 'Sanket', 'JadHAV', 35, 'IT', 65000.00);
15 - INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department,
16   Salary)
17 VALUES (3, 'SanskAR', 'Gangurde', 40, 'Finance', 75000.00);
18 select * from Employees;
19 UPDATE Employees
20 SET Salary = 70000.00
21 WHERE EmployeeID = 2;
22 select * from Employees;
23
24
25 select * from Employees;
26 UPDATE Employees
27 SET FirstName = 'Kartik', LastName = 'Pawar', Age = 45, Department =
28 'Management', Salary = 80000.00
29 WHERE EmployeeID = 3;
30 select * from Employees;
31
32
33
34 select * from Employees;
35 select * from Employees;
36 UPDATE Employees
37 SET Salary = CASE
38 WHEN Department = 'HR' THEN Salary * 1.05
39 WHEN Department = 'IT' THEN Salary * 1.08
40 WHEN Department = 'Finance' THEN Salary * 1.10
41 ELSE Salary
42 END;
43 select * from Employees;

```

Output:

EmployeeID	FirstName	LastName	Age	Department	Salary
1	Kartik	Pawar	28	HR	50000.00
2	Sanket	JadHAV	35	IT	70000.00
3	Kartik	Pawar	45	Management	80000.00

EmployeeID	FirstName	LastName	Age	Department	Salary
1	Kartik	Pawar	28	HR	50000.00
2	Sanket	JadHAV	35	IT	70000.00
3	Kartik	Pawar	45	Management	80000.00

EmployeeID	FirstName	LastName	Age	Department	Salary
1	Kartik	Pawar	28	HR	52500.00
2	Sanket	JadHAV	35	IT	75600.00
3	Kartik	Pawar	45	Management	80000.00

EmployeeID	FirstName	LastName	Age	Department	Salary
2	Sanket	JadHAV	35	IT	75600.00

Practical No.: - 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 TABLE Sales (

SaleID INT PRIMARY KEY AUTO_INCREMENT,

Product VARCHAR(50),

```
Quantity INT,  
Price DECIMAL(10,2),  
SaleDate DATE  
);  
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’);  
SELECT * FROM Sales;
```

```
SELECT COUNT(*) AS Total_Sales FROM Sales;  
SELECT SUM(Quantity * Price) AS Total_Revenue FROM Sales;  
SELECT AVG(Price) AS Average_Price FROM Sales;  
SELECT MIN(Price) AS Min_Price, MAX(Price) AS Max_Price FROM Sales;  
SELECT COUNT(DISTINCT Product) AS Unique_Products FROM Sales;  
SELECT Product, COUNT(*) AS Sales_Count  
FROM Sales  
GROUP BY Product;  
SELECT SaleDate, COUNT(*) AS Sales_Per_Day  
FROM Sales  
GROUP BY SaleDate;
```

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

```
FROM Sales
WHERE Quantity > 2;

SELECT COUNT(*) AS Sales_This_Month
FROM Sales
WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
AND YEAR(SaleDate) = YEAR(CURRENT_DATE);

SELECT COUNT(*) AS High_Value_Sales
FROM Sales
WHERE (Quantity * Price) > 50000;

SELECT Product, COUNT(*) AS High_Value_Transactions
FROM Sales
WHERE (Quantity * Price) > 40000
GROUP BY Product;

SELECT COUNT(*) AS Sales_After_Date
FROM Sales
WHERE SaleDate > '2025-02-03';

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

SELECT SUM(Quantity) AS Total_Quantity_Sold FROM Sales;

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

GROUP BY Product;

```
SELECT SaleDate, SUM(Quantity * Price) AS Revenue_Per_Day  
FROM Sales  
GROUP BY SaleDate;
```

```
SELECT SUM(Quantity * Price) AS Revenue_This_Month  
FROM Sales  
WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)  
AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
```

```
SELECT SUM(Quantity * Price) AS High_Quantity_Revenue  
FROM Sales  
WHERE Quantity > 2;
```

```
SELECT SUM(Quantity * Price) AS Revenue_After_Date  
FROM Sales  
WHERE SaleDate > '2025-02-03';
```

```
SELECT Product, SUM(Quantity * Price) AS High_Value_Revenue  
FROM Sales  
WHERE (Quantity * Price) > 40000  
GROUP BY Product;
```

```
SELECT AVG(Price) AS Average_Price FROM Sales;
```

```
SELECT AVG(Quantity) AS Average_Quantity_Sold FROM Sales;
```

```
SELECT AVG(Quantity * Price) AS Average_Revenue_Per_Transaction  
FROM Sales;
```

```
SELECT Product, AVG(Price) AS Average_Price_Per_Product  
FROM Sales  
GROUP BY Product;
```

```
SELECT Product, AVG(Quantity * Price) AS Average_Revenue_Per_Product  
FROM Sales  
GROUP BY Product;
```

```
SELECT Product, AVG(Quantity) AS Average_Quantity_Per_Product  
FROM Sales  
GROUP BY Product;
```

```
SELECT SaleDate, AVG(Quantity * Price) AS Average_Revenue_Per_Day  
FROM Sales  
GROUP BY SaleDate;
```

```
SELECT AVG(Quantity * Price) AS Average_Revenue_This_Month  
FROM Sales  
WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)  
AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
```

```
SELECT AVG(Price) AS Avg_Price_High_Quantity_Sales  
FROM Sales
```

```
WHERE Quantity > 2;
```

```
SELECT AVG(Quantity * Price) AS Average_Revenue_After_Date  
FROM Sales  
WHERE SaleDate > '2025-02-03';
```

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

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

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

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

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

```
SELECT Product, MIN(Quantity) AS Min_Quantity_Per_Product,  
MAX(Quantity) AS Max_Quantity_Per_Product  
FROM Sales
```

GROUP BY Product;

SELECT SaleDate, MIN(Quantity * Price) AS Min_Revenue_Per_Day,

MAX(Quantity * Price) AS Max_Revenue_Per_Day

FROM Sales

GROUP BY SaleDate;

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);

SELECT MIN(Price) AS Min_Price_High_Quantity_Sales, MAX(Price) AS

Max_Price_High_Quantity_Sales

FROM Sales

WHERE Quantity > 2;

SELECT MIN(Quantity * Price) AS Min_Revenue_After_Date, MAX(Quantity

* Price) AS Max_Revenue_After_Date

FROM Sales

WHERE SaleDate > '2025-02-03';

SELECT * FROM Sales;

Screenshot of Output:

The screenshot shows two identical MySQL query execution environments side-by-side, demonstrating the results of the provided SQL queries.

Top Window (Screenshot 1):

```

123 -- 1. Minimum and Maximum quantity of products sold in a single transaction
124 SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold
125   FROM Sales
126 WHERE (Min(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_Revenue) FROM Sales
127   GROUP BY Product;
128   4. Minimum and Maximum price per product
129   SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS Max_Price_Per_Product
130   FROM Sales
131   GROUP BY Product;
132   5. Minimum and Maximum revenue per product
133   SELECT Product, MIN(Quantity * Price) AS Min_Revenue_Per_Product,
134   MAX(Quantity * Price) AS Max_Revenue_Per_Product
135   FROM Sales
136   GROUP BY Product;
137   6. Minimum and Maximum quantity sold per product
138   SELECT Product, MIN(Quantity) AS Min_Quantity_Per_Product,
139   MAX(Quantity) AS Max_Quantity_Per_Product
140   FROM Sales
141   GROUP BY Product;
142   7. Minimum and Maximum revenue per day
143   SELECT SaleDate, MIN(Quantity * Price) AS Min_Revenue_Per_Day,
144   MAX(Quantity * Price) AS Max_Revenue_Per_Day
145   FROM Sales
146   GROUP BY SaleDate;
147   8. Minimum and Maximum revenue in the current month
148   - SELECT MIN(Quantity * Price) AS Min_Revenue_This_Month, MAX(Quantity *
149   * Price) AS Max_Revenue_This_Month
150   FROM Sales
151   WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
152   AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
153   9. Minimum and Maximum price of products where more than 2 units were sold
154   SELECT MIN(Price) AS Min_Price_High_Quantity_Sales, MAX(Price) AS
155   Max_Price_High_Quantity_Sales
156   FROM Sales
157   WHERE Quantity > 2;
158   10. Average revenue after a specific date (e.g., Feb 3, 2025)
159   - SELECT MIN(Quantity * Price) AS Min_Revenue_After_Date, MAX(Quantity *
160   * Price) AS Max_Revenue_After_Date
161   FROM Sales
162   WHERE SaleDate > '2025-02-03';
163   11. View final data
164   SELECT * FROM Sales;
    
```

Bottom Window (Screenshot 2):

```

97 GROUP BY Product;
98 -- 6. Average quantity sold per product
99 SELECT Product, AVG(Quantity) AS Average_Quantity_Per_Product
100 FROM Sales
101 GROUP BY Product;
102 -- 7. Average revenue per day
103 SELECT SaleDate, AVG(Quantity * Price) AS Average_Revenue_Per_Day
104 FROM Sales
105 GROUP BY SaleDate;
106 -- 8. Average revenue in the current month
107 SELECT AVG(Quantity * Price) AS Average_Revenue_This_Month
108 FROM Sales
109 WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
110 AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
111 -- 9. Minimum and Maximum sales where more than 2 units were sold
112 SELECT AVG(Price) AS Avg_Price_High_Quantity_Sales
113 FROM Sales
114 WHERE Quantity > 2;
115 -- 10. Average revenue after a specific date (e.g., Feb 3, 2025)
116 SELECT MIN(Quantity * Price) AS Average_Revenue_After_Date
117 FROM Sales
118 WHERE SaleDate > '2025-02-03';
119 -- MIN/MAX Queries
120 -- 1. Minimum and Maximum price of a product sold
121 -- 2. Minimum and Maximum price of a product
122 -- 3. Minimum and Maximum quantity of products sold in a single transaction
123 SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS
124 Max_Quantity_Sold
125   FROM Sales
126 WHERE (Min(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS
127 Max_Revenue) FROM Sales
128   4. Minimum and Maximum price per product
129   SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS
130   Max_Price_Per_Product
131   FROM Sales
132   GROUP BY Product;
133   5. Minimum and Maximum revenue per product
134   SELECT Product, MIN(Quantity * Price) AS Min_Revenue_Per_Product,
135   MAX(Quantity * Price) AS Max_Revenue_Per_Product
136   FROM Sales
137   GROUP BY Product;
138   6. Minimum and Maximum quantity sold per product
139   SELECT Product, MIN(Quantity) AS Min_Quantity_Per_Product,
140   
```

The right side of both windows displays the input for the program (Optional) and the resulting data from the STDIN section. The data includes:

- Min_Price | Max_Price | 20000.00 | 78000.00
- Unique_Products | 3
- Product | Sales_Count | Laptop | 2 | Mobile | 2 | Tablet | 2
- 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
- High_Quantity_Sales | 3
- SaleID | Product | Quantity | Price | SaleDate | 1 | Laptop | 2 | 75000.00 | 2025-02-01 | 2 | Mobile | 5 | 20000.00 | 2025-02-02 | 3 | Tablet | 3 | 30000.00 | 2025-02-03 | 4 | Laptop | 1 | 78000.00 | 2025-02-04 | 5 | Mobile | 4 | 22000.00 | 2025-02-05 | 6 | Tablet | 2 | 32000.00 | 2025-02-06
- Total_Sales | 6
- Total_Revenue | 570000.00
- Average_Price | 42833.33333
- Min_Price | Max_Price | 20000.00 | 78000.00

queries.sql

```

-- 1. Minimum and Maximum quantity of products sold in a single transaction
123 SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS
124 Max_QuantitySold FROM Sales;
125 -- 2. Minimum and Maximum revenue generated from a single transaction
126 SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS
127 Max_Revenue FROM Sales;
128 -- 3. Minimum and Maximum price per product
129 SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS
130 Max_Price_Per_Product
131 FROM Sales;
132 GROUP BY Product;
133 -- 4. Minimum and Maximum revenue per product
134 SELECT Product, MIN(Quantity * Price) AS Min_Revenue_Per_Product,
135 MAX(Quantity * Price) AS Max_Revenue_Per_Product
136 FROM Sales;
137 GROUP BY Product;
138 -- 5. Minimum and Maximum quantity sold per product
139 SELECT Product, MIN(Quantity) AS Min_Quantity_Per_Product,
140 MAX(Quantity) AS Max_Quantity_Per_Product
141 FROM Sales;
142 GROUP BY Product;
143 -- 6. Minimum and Maximum revenue per day
144 SELECT SaleDate, MIN(Quantity * Price) AS Min_Revenue_Per_Day,
145 MAX(Quantity * Price) AS Max_Revenue_Per_Day
146 FROM Sales;
147 GROUP BY SaleDate;
148 -- 7. Minimum and Maximum revenue in the current month
149 -- 8. Minimum and Maximum price in the current month
150 -- 9. Minimum and Maximum revenue after a specific date (e.g., Feb 3, 2025)
151 -- 10. Minimum and Maximum price in the current month
152 WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
153 AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
154 -- 11. Minimum and Maximum price of products where more than 2 units were sold
155 SELECT MIN(Price) AS Min_Price_High_Quantity_Sales, MAX(Price) AS
156 Max_Price_High_Quantity_Sales
157 FROM Sales
158 WHERE Quantity > 2;
159 -- 12. Minimum and Maximum revenue after a specific date (e.g., Feb 3, 2025)
160 -- 13. Minimum and Maximum revenue after a specific date (e.g., Feb 3, 2025)
161 -- 14. Minimum and Maximum revenue per day
162 -- 15. Minimum and Maximum revenue per month
163 WHERE SaleDate > '2025-02-03';
164 -- View final data
165 SELECT * FROM Sales;

```

43e2bcxtt

STDRIN

Input for the program (Optional)

High_Quantity_Sales
3

Sales_This_Month
0

High_Value_Sales
6

Product High_Value_Transactions

Laptop 2
Mobile 2
Tablet 2

Sales_After_Date
3

Total_Revenue
570000.00

B4F Partly cloudy

22:18
06-04-2025

queries.sql

```

-- 1. Minimum and Maximum quantity of products sold in a single transaction
123 SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS
124 Max_QuantitySold FROM Sales;
125 -- 2. Minimum and Maximum revenue generated from a single transaction
126 SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS
127 Max_Revenue FROM Sales;
128 -- 3. Minimum and Maximum price per product
129 SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS
130 Max_Price_Per_Product
131 FROM Sales;
132 GROUP BY Product;
133 -- 4. Minimum and Maximum revenue per product
134 SELECT Product, MIN(Quantity * Price) AS Min_Revenue_Per_Product,
135 MAX(Quantity * Price) AS Max_Revenue_Per_Product
136 FROM Sales;
137 GROUP BY Product;
138 -- 5. Minimum and Maximum quantity sold per product
139 SELECT Product, MIN(Quantity) AS Min_Quantity_Per_Product,
140 MAX(Quantity) AS Max_Quantity_Per_Product
141 FROM Sales;
142 GROUP BY Product;
143 -- 6. Minimum and Maximum revenue per day
144 SELECT SaleDate, MIN(Quantity * Price) AS Min_Revenue_Per_Day,
145 MAX(Quantity * Price) AS Max_Revenue_Per_Day
146 FROM Sales;
147 GROUP BY SaleDate;
148 -- 7. Minimum and Maximum revenue in the current month
149 -- 8. Minimum and Maximum price in the current month
150 -- 9. Minimum and Maximum revenue after a specific date (e.g., Feb 3, 2025)
151 -- 10. Minimum and Maximum price in the current month
152 WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
153 AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
154 -- 11. Minimum and Maximum price of products where more than 2 units were sold
155 SELECT MIN(Price) AS Min_Price_High_Quantity_Sales, MAX(Price) AS
156 Max_Price_High_Quantity_Sales
157 FROM Sales
158 WHERE Quantity > 2;
159 -- 12. Minimum and Maximum revenue after a specific date (e.g., Feb 3, 2025)
160 -- 13. Minimum and Maximum revenue after a specific date (e.g., Feb 3, 2025)
161 -- 14. Minimum and Maximum revenue per day
162 -- 15. Minimum and Maximum revenue per month
163 WHERE SaleDate > '2025-02-03';
164 -- View final data
165 SELECT * FROM Sales;

```

43e2bcxtt

STDRIN

Input for the program (Optional)

Total_Quantity_Sold
17

Product Revenue_Per_Product

Laptop 228000.00
Mobile 188000.00
Tablet 154000.00

SaleDate Revenue_Per_Day

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

Revenue_This_Month
NULL

High_Quantity_Revenue

Sports headline
PSG crowned Lig...

22:18
06-04-2025

queries.sql

```

1-- 1. Minimum and maximum quantity of products sold in a single transaction
2SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold FROM Sales;
3-- 2. Minimum and Maximum revenue generated from a single transaction
4SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_Revenue FROM Sales;
5-- 3. Minimum and Maximum price per product
6SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS Max_Price_Per_Product
7FROM Sales;
8-- 4. Minimum and Maximum revenue per product
9SELECT Product, MIN(Quantity * Price) AS Min_Revenue_Per_Product,
10MAX(Quantity * Price) AS Max_Revenue_Per_Product
11FROM Sales;
12-- 5. Minimum and Maximum quantity sold per product
13SELECT Product, MIN(Quantity) AS Min_Quantity_Per_Product,
14MAX(Quantity) AS Max_Quantity_Per_Product
15FROM Sales;
16-- 6. Minimum and Maximum revenue per day
17SELECT SaleDate, MIN(Quantity * Price) AS Min_Revenue_Per_Day,
18MAX(Quantity * Price) AS Max_Revenue_Per_Day
19FROM Sales;
20-- 7. Minimum and Maximum revenue in the current month
21-- 8. Minimum and Maximum revenue after a specific date (e.g., Feb 3, 2025)
22-- 9. Minimum and Maximum price where more than 2 units were sold
23-- 10. Minimum and Maximum revenue after a specific date (e.g., Feb 3, 2025)
24-- 11. Minimum and Maximum revenue after a specific date (e.g., Feb 3, 2025)
25-- 12. Minimum and Maximum revenue per day
26-- 13. Minimum and Maximum quantity sold per day
27-- 14. Minimum and Maximum quantity sold per month
28-- 15. Minimum and Maximum revenue per month
29-- 16. Minimum and Maximum revenue per year
30-- 17. Minimum and Maximum price per year
31-- 18. Minimum and Maximum revenue per year
32-- 19. Minimum and Maximum revenue per year
33-- 20. Minimum and Maximum revenue per year
34-- 21. Minimum and Maximum revenue per year
35-- 22. Minimum and Maximum revenue per year
36-- 23. Minimum and Maximum revenue per year
37-- 24. Minimum and Maximum revenue per year
38-- 25. Minimum and Maximum revenue per year
39-- 26. Minimum and Maximum revenue per year
40-- 27. Minimum and Maximum revenue per year
41-- 28. Minimum and Maximum revenue per year
42-- 29. Minimum and Maximum revenue per year
43-- 30. Minimum and Maximum revenue per year
44-- 31. Minimum and Maximum revenue per year
45-- 32. Minimum and Maximum revenue per year
46-- 33. Minimum and Maximum revenue per year
47-- 34. Minimum and Maximum revenue per year
48-- 35. Minimum and Maximum revenue per year
49-- 36. Minimum and Maximum revenue per year
50-- 37. Minimum and Maximum revenue per year
51-- 38. Minimum and Maximum revenue per year
52-- 39. Minimum and Maximum revenue per year
53-- 40. Minimum and Maximum revenue per year
54-- 41. Minimum and Maximum revenue per year
55-- 42. Minimum and Maximum revenue per year
56-- 43. Minimum and Maximum revenue per year
57-- 44. Minimum and Maximum revenue per year
58-- 45. Minimum and Maximum revenue per year
59-- 46. Minimum and Maximum revenue per year
60-- 47. Minimum and Maximum revenue per year
61-- 48. Minimum and Maximum revenue per year
62-- 49. Minimum and Maximum revenue per year
63-- 50. Minimum and Maximum revenue per year
64-- 51. Minimum and Maximum revenue per year
65-- 52. Minimum and Maximum revenue per year
66-- 53. Minimum and Maximum revenue per year
67-- 54. Minimum and Maximum revenue per year
68-- 55. Minimum and Maximum revenue per year
69-- 56. Minimum and Maximum revenue per year
70-- 57. Minimum and Maximum revenue per year
71-- 58. Minimum and Maximum revenue per year
72-- 59. Minimum and Maximum revenue per year
73-- 60. Minimum and Maximum revenue per year
74-- 61. Minimum and Maximum revenue per year
75-- 62. Minimum and Maximum revenue per year
76-- 63. Minimum and Maximum revenue per year
77-- 64. Minimum and Maximum revenue per year
78-- 65. Minimum and Maximum revenue per year
79-- 66. Minimum and Maximum revenue per year
80-- 67. Minimum and Maximum revenue per year
81-- 68. Minimum and Maximum revenue per year
82-- 69. Minimum and Maximum revenue per year
83-- 70. Minimum and Maximum revenue per year
84-- 71. Minimum and Maximum revenue per year
85-- 72. Minimum and Maximum revenue per year
86-- 73. Minimum and Maximum revenue per year
87-- 74. Minimum and Maximum revenue per year
88-- 75. Minimum and Maximum revenue per year
89-- 76. Minimum and Maximum revenue per year
90-- 77. Minimum and Maximum revenue per year
91-- 78. Minimum and Maximum revenue per year
92-- 79. Minimum and Maximum revenue per year
93-- 80. Minimum and Maximum revenue per year
94-- 81. Minimum and Maximum revenue per year
95-- 82. Minimum and Maximum revenue per year
96-- 83. Minimum and Maximum revenue per year
97-- 84. Minimum and Maximum revenue per year
98-- 85. Minimum and Maximum revenue per year
99-- 86. Minimum and Maximum revenue per year
100-- 87. Minimum and Maximum revenue per year
101-- 88. Minimum and Maximum revenue per year
102-- 89. Minimum and Maximum revenue per year
103-- 90. Minimum and Maximum revenue per year
104-- 91. Minimum and Maximum revenue per year
105-- 92. Minimum and Maximum revenue per year
106-- 93. Minimum and Maximum revenue per year
107-- 94. Minimum and Maximum revenue per year
108-- 95. Minimum and Maximum revenue per year
109-- 96. Minimum and Maximum revenue per year
110-- 97. Minimum and Maximum revenue per year
111-- 98. Minimum and Maximum revenue per year
112-- 99. Minimum and Maximum revenue per year
113-- 100. Minimum and Maximum revenue per year
114-- 101. Minimum and Maximum revenue per year
115-- 102. Minimum and Maximum revenue per year
116-- 103. Minimum and Maximum revenue per year
117-- 104. Minimum and Maximum revenue per year
118-- 105. Minimum and Maximum revenue per year
119-- 106. Minimum and Maximum revenue per year
120-- 107. Minimum and Maximum revenue per year
121-- 108. Minimum and Maximum revenue per year
122-- 109. Minimum and Maximum revenue per year
123-- 110. Minimum and Maximum revenue per year
124-- 111. Minimum and Maximum revenue per year
125-- 112. Minimum and Maximum revenue per year
126-- 113. Minimum and Maximum revenue per year
127-- 114. Minimum and Maximum revenue per year
128-- 115. Minimum and Maximum revenue per year
129-- 116. Minimum and Maximum revenue per year
130-- 117. Minimum and Maximum revenue per year
131-- 118. Minimum and Maximum revenue per year
132-- 119. Minimum and Maximum revenue per year
133-- 120. Minimum and Maximum revenue per year
134-- 121. Minimum and Maximum revenue per year
135-- 122. Minimum and Maximum revenue per year
136-- 123. Minimum and Maximum revenue per year
137-- 124. Minimum and Maximum revenue per year
138-- 125. Minimum and Maximum revenue per year
139-- 126. Minimum and Maximum revenue per year
140-- 127. Minimum and Maximum revenue per year
141-- 128. Minimum and Maximum revenue per year
142-- 129. Minimum and Maximum revenue per year
143-- 130. Minimum and Maximum revenue per year
144-- 131. Minimum and Maximum revenue per year
145-- 132. Minimum and Maximum revenue per year
146-- 133. Minimum and Maximum revenue per year
147-- 134. Minimum and Maximum revenue per year
148-- 135. Minimum and Maximum revenue per year
149-- 136. Minimum and Maximum revenue per year
150-- 137. Minimum and Maximum revenue per year
151-- 138. Minimum and Maximum revenue per year
152-- 139. Minimum and Maximum revenue per year
153-- 140. Minimum and Maximum revenue per year
154-- 141. Minimum and Maximum revenue per year
155-- 142. Minimum and Maximum revenue per year
156-- 143. Minimum and Maximum revenue per year
157-- 144. Minimum and Maximum revenue per year
158-- 145. Minimum and Maximum revenue per year
159-- 146. Minimum and Maximum revenue per year
160-- 147. Minimum and Maximum revenue per year
161-- 148. Minimum and Maximum revenue per year
162-- 149. Minimum and Maximum revenue per year
163-- 150. Minimum and Maximum revenue per year
164-- 151. Minimum and Maximum revenue per year
165-- 152. Minimum and Maximum revenue per year

```

Input for the program (Optional)

Revenue_After_Date	230000.00
Product High_Value_Revenue	Laptop 228000.00
Product High_Value_Revenue	Mobile 188000.00
Product High_Value_Revenue	Tablet 154000.00
Average_Price	42833.33333
Average_Quantity_Sold	2.8333
Average_Revenue_Per_Transaction	95000.00000
Product Average_Price_Per_Product	Laptop 76500.000000
Product Average_Price_Per_Product	Mobile 21000.000000
Product Average_Price_Per_Product	Tablet 31000.000000
Product Average_Revenue_Per_Product	Laptop 114000.000000
Product Average_Revenue_Per_Product	Mobile 94000.000000
Product Average_Revenue_Per_Product	Tablet 77000.000000
Product Average_Quantity_Per_Product	Laptop 1.5000
Product Average_Quantity_Per_Product	Mobile 4.5000
Product Average_Quantity_Per_Product	Tablet 2.5000
SaleDate Average_Revenue_Per_Day	2025-02-01 150000.000000
SaleDate Average_Revenue_Per_Day	2025-02-02 180000.000000
SaleDate Average_Revenue_Per_Day	2025-02-03 90000.000000
SaleDate Average_Revenue_Per_Day	2025-02-04 78000.000000
SaleDate Average_Revenue_Per_Day	2025-02-05 88000.000000
SaleDate Average_Revenue_Per_Day	2025-02-06 64000.000000
Average_Revenue_This_Month	Average_Revenue_This_Month

OneCompiler

```

SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold FROM Sales;
-- 1. Minimum and Maximum quantity sold in a single transaction
SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_Revenue FROM Sales;
-- 2. Minimum and Maximum revenue generated from a single transaction
SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS Max_Price_Per_Product
FROM Sales
GROUP BY Product;
-- 3. Minimum and Maximum price 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;
-- 4. 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;
-- 5. 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;
-- 6. 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);
-- 7. Minimum and Maximum price of products where more than 2 units were sold
SELECT Product, MIN(Price) AS Min_Price_High_Quantity_Sales, MAX(Price) AS Max_Price_High_Quantity_Sales
FROM Sales
WHERE Quantity > 2;
-- 8. 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';
-- 9. View final data
SELECT * FROM Sales;

```

Input for the program (Optional)

2025-02-05	80000.000000
2025-02-06	64000.000000
	NULL
	Average_Revenue_This_Month
	24000.000000
	Avg_Price_High_Quantity_Sales
	76666.666667
	Min_Price Max_Price
	20000.00 78000.00
	Min_Quantity_Sold Max_Quantity_Sold
	1 5
	Min_Revenue Max_Revenue
	64000.00 150000.00

Sports headline
PSG crowned Lig... 22:19 06-04-2025

OneCompiler

```

SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold FROM Sales;
-- 1. Minimum and Maximum quantity sold in a single transaction
SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_Revenue FROM Sales;
-- 2. Minimum and Maximum revenue generated from a single transaction
SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS Max_Price_Per_Product
FROM Sales
GROUP BY Product;
-- 3. Minimum and Maximum price 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;
-- 4. 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;
-- 5. 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;
-- 6. 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);
-- 7. Minimum and Maximum price of products where more than 2 units were sold
SELECT Product, MIN(Price) AS Min_Price_High_Quantity_Sales, MAX(Price) AS Max_Price_High_Quantity_Sales
FROM Sales
WHERE Quantity > 2;
-- 8. 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';
-- 9. View final data
SELECT * FROM Sales;

```

Input for the program (Optional)

Product Min_Price_Per_Product Max_Price_Per_Product
Laptop 75000.00 78000.00
Mobile 20000.00 22000.00
Tablet 30000.00 32000.00
Product Min_Revenue_Per_Product Max_Revenue_Per_Product
Laptop 70000.00 150000.00
Mobile 88000.00 100000.00
Tablet 64000.00 90000.00
Product Min_Quantity_Per_Product Max_Quantity_Per_Product
Laptop 1 2
Mobile 4 5
Tablet 2 3
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

84°F Partly cloudy 22:19 06-04-2025

The screenshot shows a browser window with several tabs open. The active tab is titled '43e2bcxtt - MySQL - OneCompiler'. The page displays a SQL script named 'queries.sql' and its execution results.

```

123 SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold FROM Sales;
124 -- 1. Minimum and Maximum revenue generated from a single transaction
125 SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_Revenue FROM Sales;
126 -- 2. Minimum and Maximum price per product
127 SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS Max_Price_Per_Product
128 FROM Sales;
129 GROUP BY Product;
130 -- 3. Minimum and Maximum revenue per product
131 SELECT Product, MIN(Quantity * Price) AS Min_Revenue_Per_Product,
132 MAX(Quantity * Price) AS Max_Revenue_Per_Product
133 FROM Sales;
134 GROUP BY Product;
135 -- 4. Minimum and Maximum quantity sold per product
136 SELECT Product, MIN(Quantity) AS Min_Quantity_Per_Product,
137 MAX(Quantity) AS Max_Quantity_Per_Product
138 FROM Sales;
139 GROUP BY Product;
140 -- 5. Minimum and Maximum revenue per day
141 SELECT SaleDate, MIN(Quantity * Price) AS Min_Revenue_Per_Day,
142 MAX(Quantity * Price) AS Max_Revenue_Per_Day
143 FROM Sales;
144 GROUP BY SaleDate;
145 -- 6. Minimum and Maximum revenue in the current month
146 - SELECT MIN(Quantity * Price) AS Min_Revenue_This_Month, MAX(Quantity
147 * Price) AS Max_Revenue_This_Month
148 FROM Sales
149 WHERE MONTH(SaleDate) = MONTH(CURRENT_DATE)
150 AND YEAR(SaleDate) = YEAR(CURRENT_DATE);
151 -- 7. Minimum and Maximum price of products where more than 2 units were sold
152 SELECT MIN(Price) AS Min_Price_High_Quantity_Sales, MAX(Price) AS
153 Max_Price_High_Quantity_Sales
154 FROM Sales
155 WHERE Quantity > 2;
156 -- 8. Minimum and Maximum revenue after a specific date (e.g., Feb 3, 2025)
157 - SELECT MIN(Quantity * Price) AS Min_Revenue_After_Date, MAX(Quantity
158 * Price) AS Max_Revenue_After_Date
159 FROM Sales
160 WHERE SaleDate > '2025-02-03';
161 -- View final data
162 SELECT * FROM Sales;

```

The results of the query execution are shown in a table:

	Min_Revenue_This_Month	Max_Revenue_This_Month			
	NULL	NULL			
	Min_Price_High_Quantity_Sales	Max_Price_High_Quantity_Sales			
	20000.00	30000.00			
	Min_Revenue_After_Date	Max_Revenue_After_Date			
	64000.00	88000.00			
	SaleID	Product	Quantity	Price	SaleDate
1	Laptop	2	75000.00	2025-02-01	
2	Mobile	5	20000.00	2025-02-02	
3	Tablet	3	30000.00	2025-02-03	
4	Laptop	1	78000.00	2025-02-04	
5	Mobile	4	22000.00	2025-02-05	
6	Tablet	2	32000.00	2025-02-06	

Practical No.: - 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 TABLE Customers (
customer_id INT PRIMARY KEY,
customer_name VARCHAR(100) NOT NULL
);

```

```

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 INTO Customers (customer_id, customer_name) VALUES  
(1, 'Snaket'),  
(2, 'Sanskar'),  
(3, 'Kartik'),  
(4, 'David');  
  
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 * FROM Customers;  
  
SELECT * FROM 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;
```

```
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;
```

```
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;
```

Screenshot of Output: -

queries.sql + 43eg57wq2

```
16 INSERT INTO Orders (order_id, order_date, customer_id) VALUES
17 (101, '2024-01-01', 1),
18 (102, '2024-01-02', 2),
19 (103, '2024-01-03', 4);
20 SELECT * FROM Customers;
21 SELECT * FROM Orders;
22
23 SELECT
24 c.customer_id,
25 c.customer_name,
26 o.order_id,
27 o.order_date
28 FROM
29 Customers c
30 INNER JOIN
31 Orders o
32 ON
33 c.customer_id = o.customer_id;
34
35 SELECT
36 c.customer_id,
37 c.customer_name,
38 o.order_id,
39 o.order_date
40 FROM
41 Customers c
42 LEFT JOIN
43 Orders o
44 ON
45 c.customer_id = o.customer_id;
46
47 SELECT
48 c.customer_id,
49 c.customer_name,
50 o.order_id,
51 o.order_date
52 FROM
53 Customers c
54 RIGHT JOIN
55 Orders o
56 ON
57 c.customer_id = o.customer_id;
58
```

STDIN

Input for the program (Optional)

Output:

customer_id	customer_name
1	Snaket
2	Sanskars
3	Kartik
4	David

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

customer_id	customer_name	order_id	order_date
1	Snaket	101	2024-01-01
2	Sanskars	102	2024-01-02
3	Kartik	NULL	NULL
4	David	103	2024-01-03

customer_id	customer_name	order_id	order_date
1	Snaket	101	2024-01-01
2	Sanskars	102	2024-01-02
3	Kartik	NULL	NULL
4	David	103	2024-01-03

queries.sql + 43eg57wq2

```
16 INSERT INTO Orders (order_id, order_date, customer_id) VALUES
17 (101, '2024-01-01', 1),
18 (102, '2024-01-02', 2),
19 (103, '2024-01-03', 4);
20 SELECT * FROM Customers;
21 SELECT * FROM Orders;
22
23 SELECT
24 c.customer_id,
25 c.customer_name,
26 o.order_id,
27 o.order_date
28 FROM
29 Customers c
30 INNER JOIN
31 Orders o
32 ON
33 c.customer_id = o.customer_id;
34
35 SELECT
36 c.customer_id,
37 c.customer_name,
38 o.order_id,
39 o.order_date
40 FROM
41 Customers c
42 LEFT JOIN
43 Orders o
44 ON
45 c.customer_id = o.customer_id;
46
47 SELECT
48 c.customer_id,
49 c.customer_name,
50 o.order_id,
51 o.order_date
52 FROM
53 Customers c
54 RIGHT JOIN
55 Orders o
56 ON
57 c.customer_id = o.customer_id;
58
```

STDIN

Input for the program (Optional)

Output:

3	Kartik
4	David

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

customer_id	customer_name	order_id	order_date
1	Snaket	101	2024-01-01
2	Sanskars	102	2024-01-02
3	Kartik	NULL	NULL
4	David	103	2024-01-03

customer_id	customer_name	order_id	order_date
1	Snaket	101	2024-01-01
2	Sanskars	102	2024-01-02
3	Kartik	NULL	NULL
4	David	103	2024-01-03