Name: Sanket Jadhav

PRN: 2124UCSM1126

Dept: Cyber Security

Practical No.: - 1

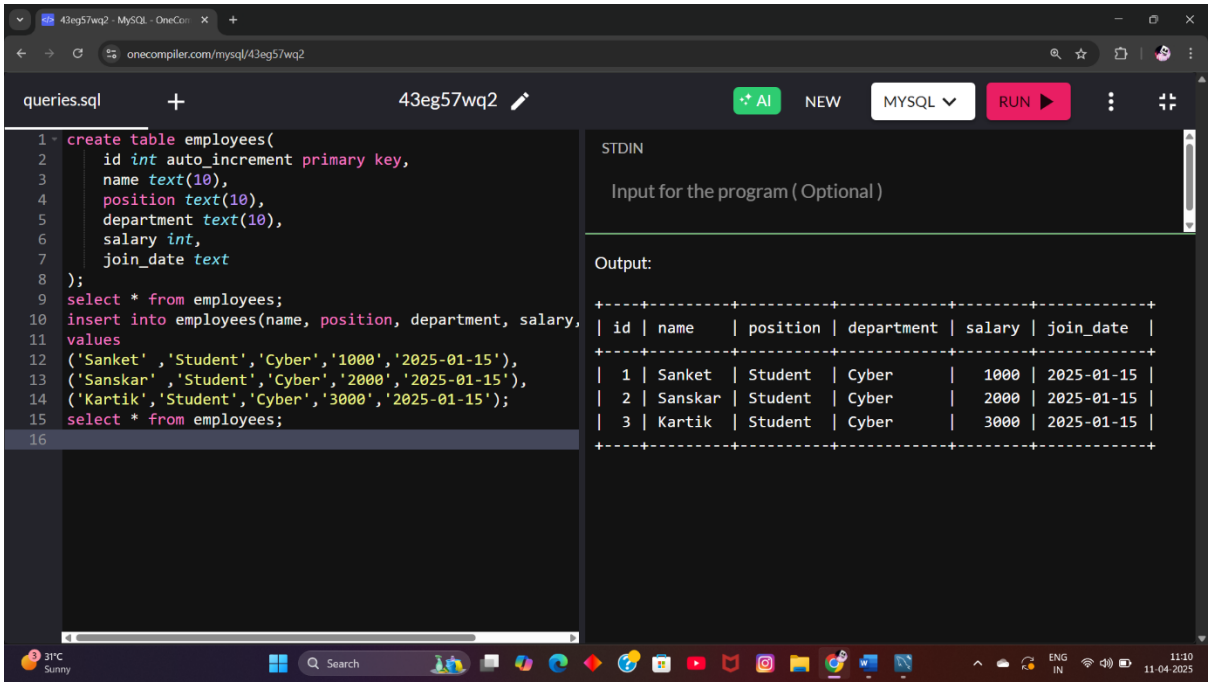
Install and set up MySQL.

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: -



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: -

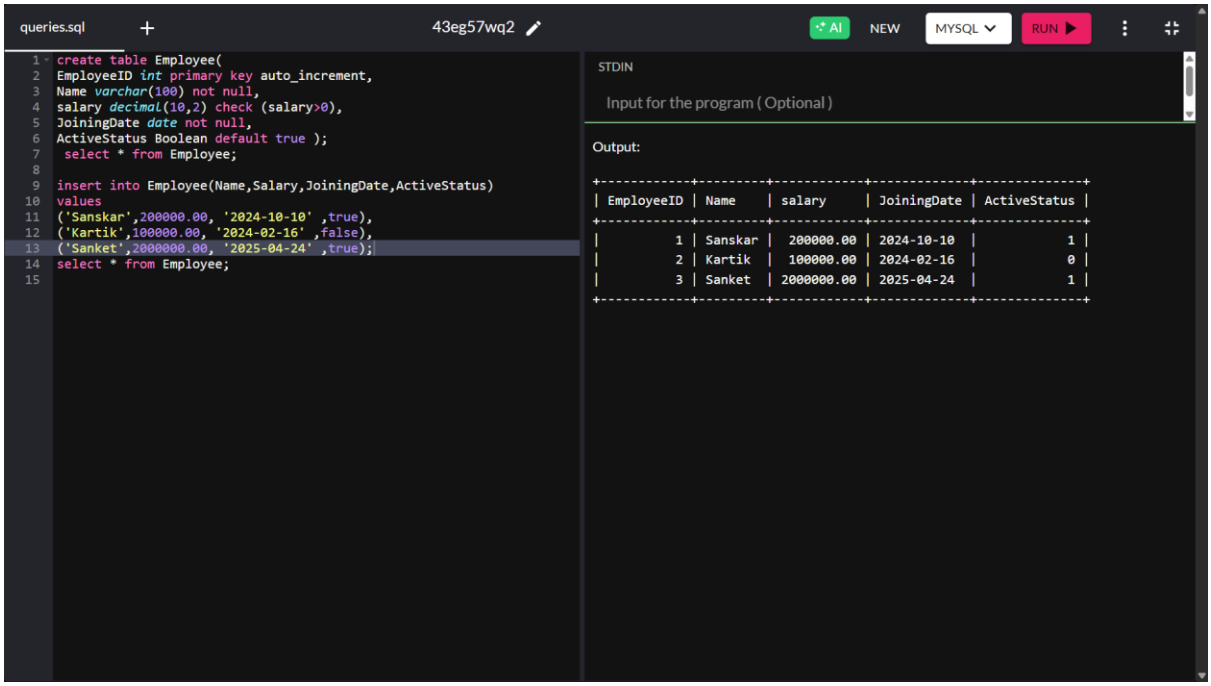
```
create table Employee(
EmployeeID int primary key auto_increment,
Name varchar(100) not null,
salary decimal(10,2) check (salary>0),
JoiningDate date not null,
ActiveStatus Boolean default true );
 select * from Employee;


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: -



Practical No.: - 3

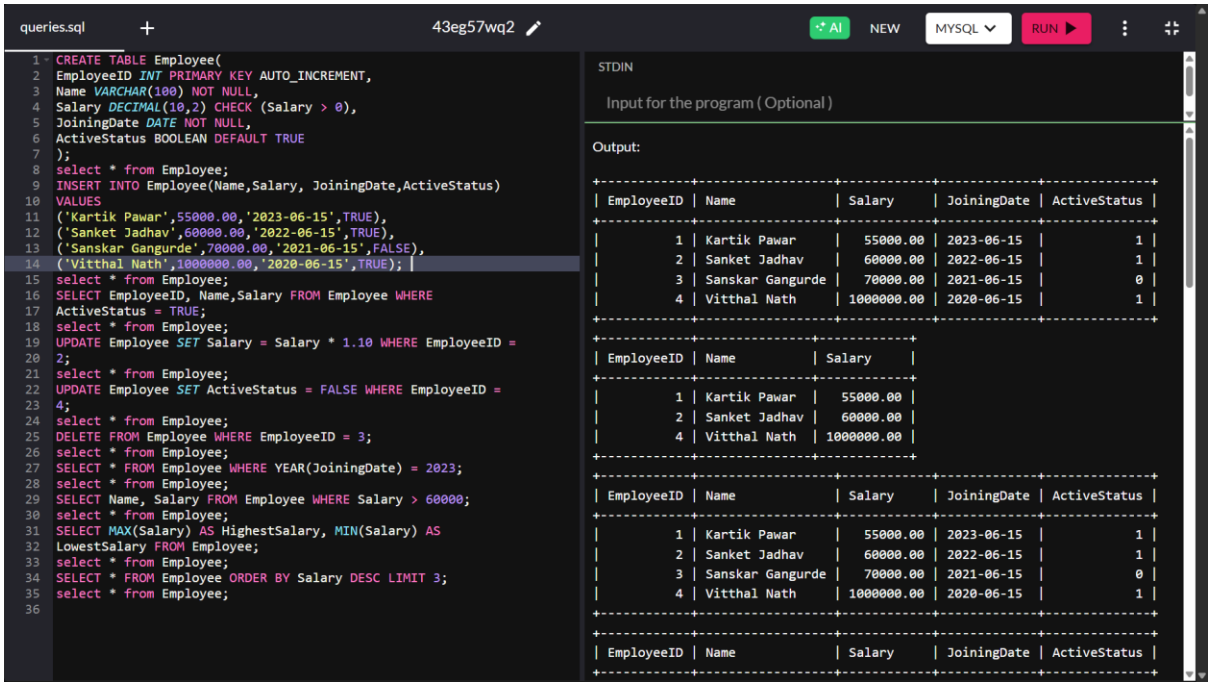
Aim: - Create a table with columns for EmployeeID, Name, Salary, JoiningDate, and ActiveStatus using different data types. Insert sample data and perform queries to manipulate and retrieve data.

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: -



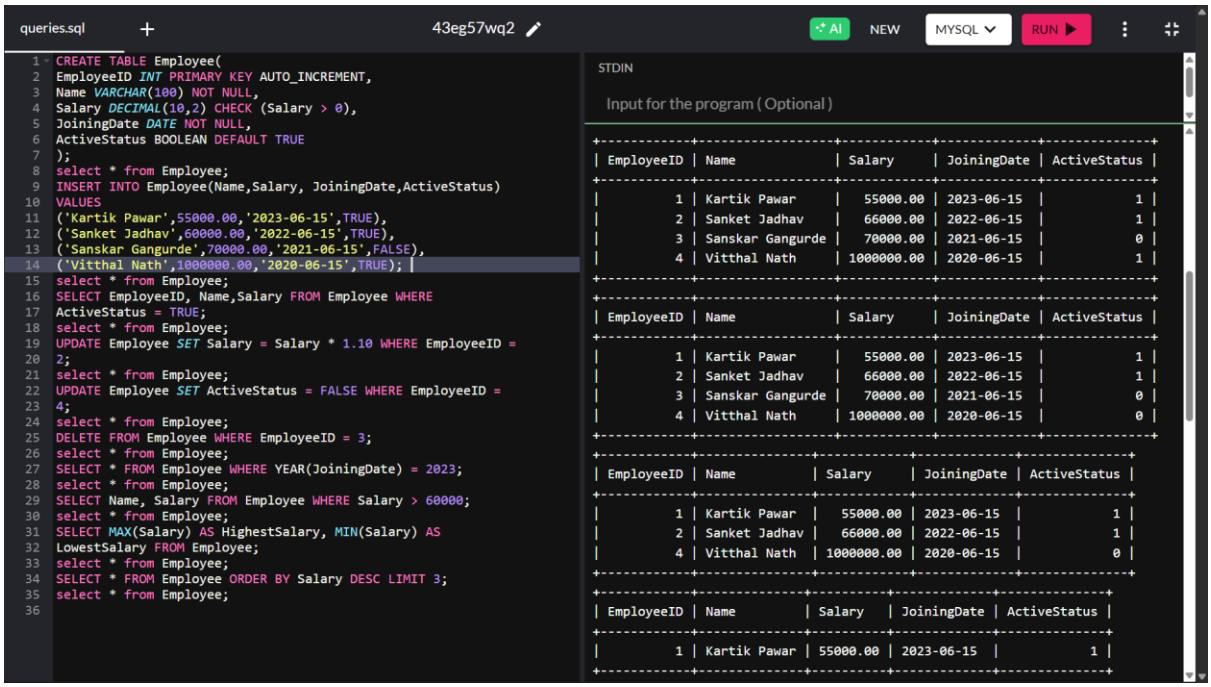
```
queries.sql                                  43eg57wq2                    AI  NEW  MYSQL  RUN

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;
```

STDIN

Input for the program (Optional)

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    | 1000000.00 | 2020-06-15  |            1 |
+------------+-----------------+------------+-------------+--------------+
+------------+---------------+------------+
| EmployeeID | Name          | Salary     |
+------------+---------------+------------+
|          1 | Kartik Pawar  |   55000.00 |
|          2 | Sanket Jadhav |   60000.00 |
|          4 | Vitthal Nath  | 1000000.00 |
+------------+---------------+------------+
+------------+-----------------+------------+-------------+--------------+
| 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    | 1000000.00 | 2020-06-15  |            1 |
+------------+-----------------+------------+-------------+--------------+
+------------+-----------------+------------+-------------+--------------+
| EmployeeID | Name            | Salary     | JoiningDate | ActiveStatus |
+------------+-----------------+------------+-------------+--------------+
```



```
queries.sql                                  43eg57wq2                    AI  NEW  MYSQL  RUN

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;
```

STDIN

Input for the program (Optional)

```
+------------+-----------------+------------+-------------+--------------+
| 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    | 1000000.00 | 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    | 1000000.00 | 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 |
|          4 | Vitthal Nath  | 1000000.00 | 2020-06-15  |            0 |
+------------+---------------+------------+-------------+--------------+
+------------+--------------+----------+-------------+--------------+
| EmployeeID | Name         | Salary   | JoiningDate | ActiveStatus |
+------------+--------------+----------+-------------+--------------+
|          1 | Kartik Pawar | 55000.00 | 2023-06-15  |            1 |
+------------+--------------+----------+-------------+--------------+
```

queries.sql 43eg57wq2 AI NEW MYSQL RUN

```sql
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;
```

STDIN

Input for the program (Optional)

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	1000000.00	2020-06-15	0

Name	Salary
Sanket Jadhav	66000.00
Vitthal Nath	1000000.00

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	1000000.00	2020-06-15	0

HighestSalary	LowestSalary
1000000.00	55000.00

EmployeeID	Name	Salary	JoiningDate	ActiveStatus

queries.sql 43eg57wq2 AI NEW MYSQL RUN

```sql
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;
```

STDIN

Input for the program (Optional)

4	Vitthal Nath	1000000.00	2020-06-15	0

HighestSalary	LowestSalary
1000000.00	55000.00

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	1000000.00	2020-06-15	0

EmployeeID	Name	Salary	JoiningDate	ActiveStatus
4	Vitthal Nath	1000000.00	2020-06-15	0
2	Sanket Jadhav	66000.00	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	66000.00	2022-06-15	1
4	Vitthal Nath	1000000.00	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

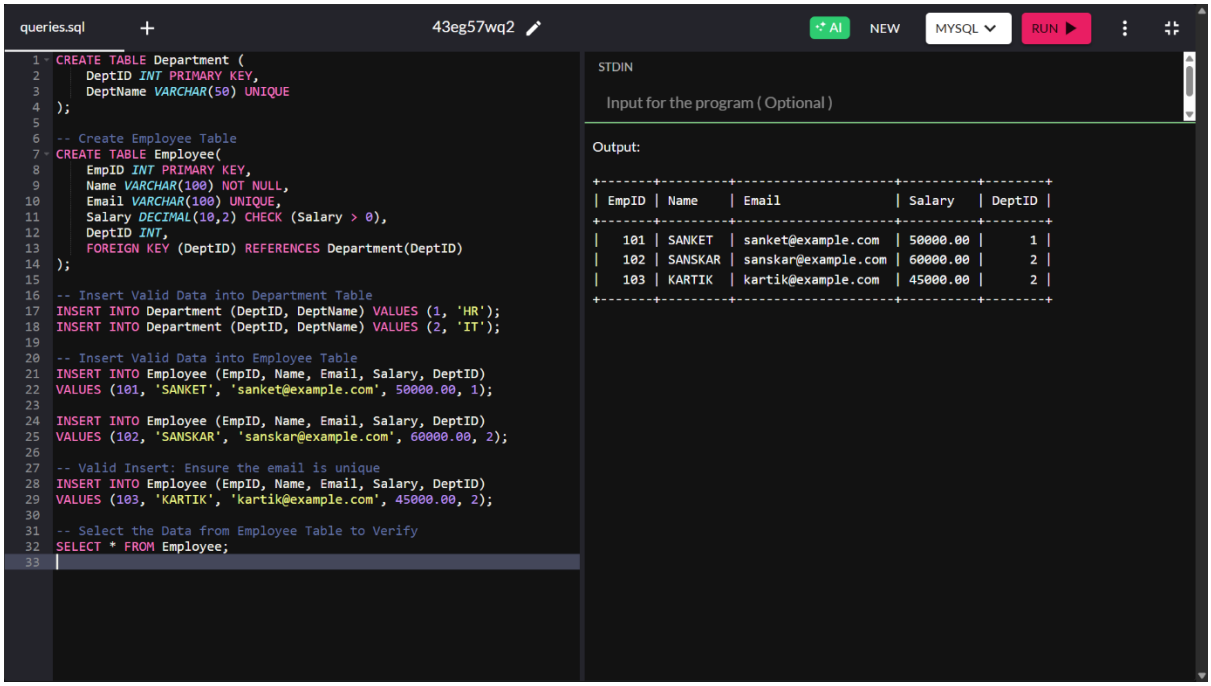
INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID)

VALUES (103, 'KARTIK', 'kartik@example.com', 45000.00, 2);

-- Select the Data from Employee Table to Verify

SELECT * FROM Employee;

Screenshot of Output: -



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: -

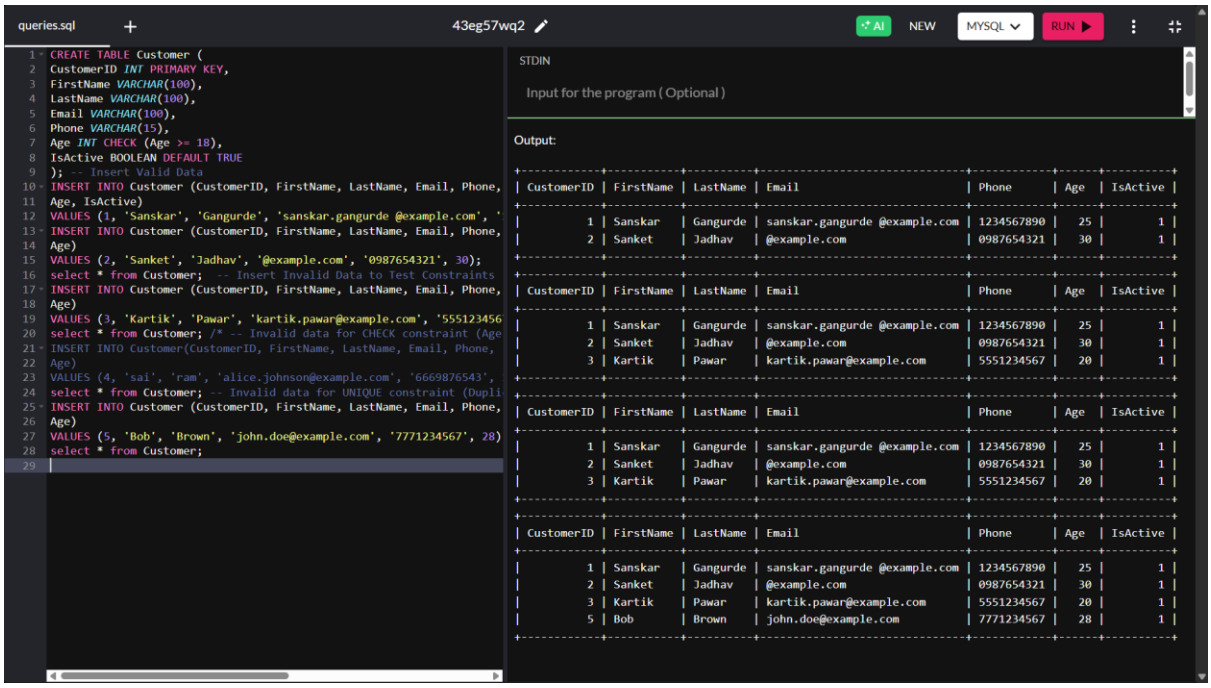
CREATE TABLE Customer (

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: -



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),

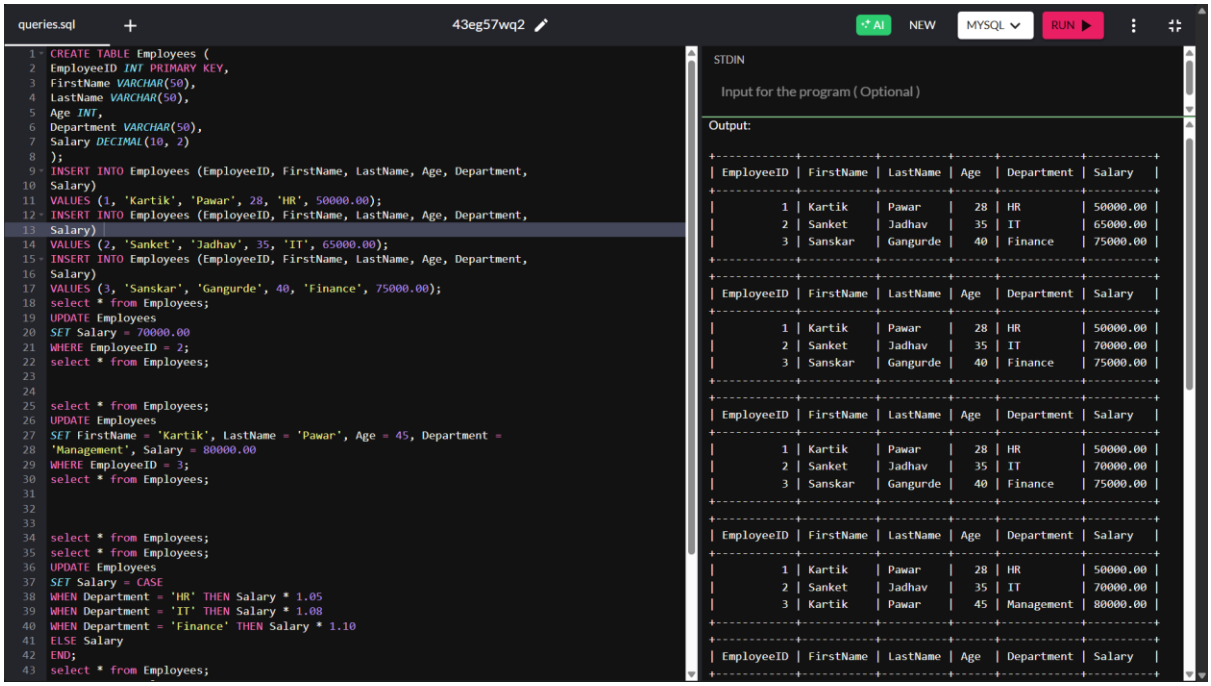
Salary DECIMAL(10, 2)

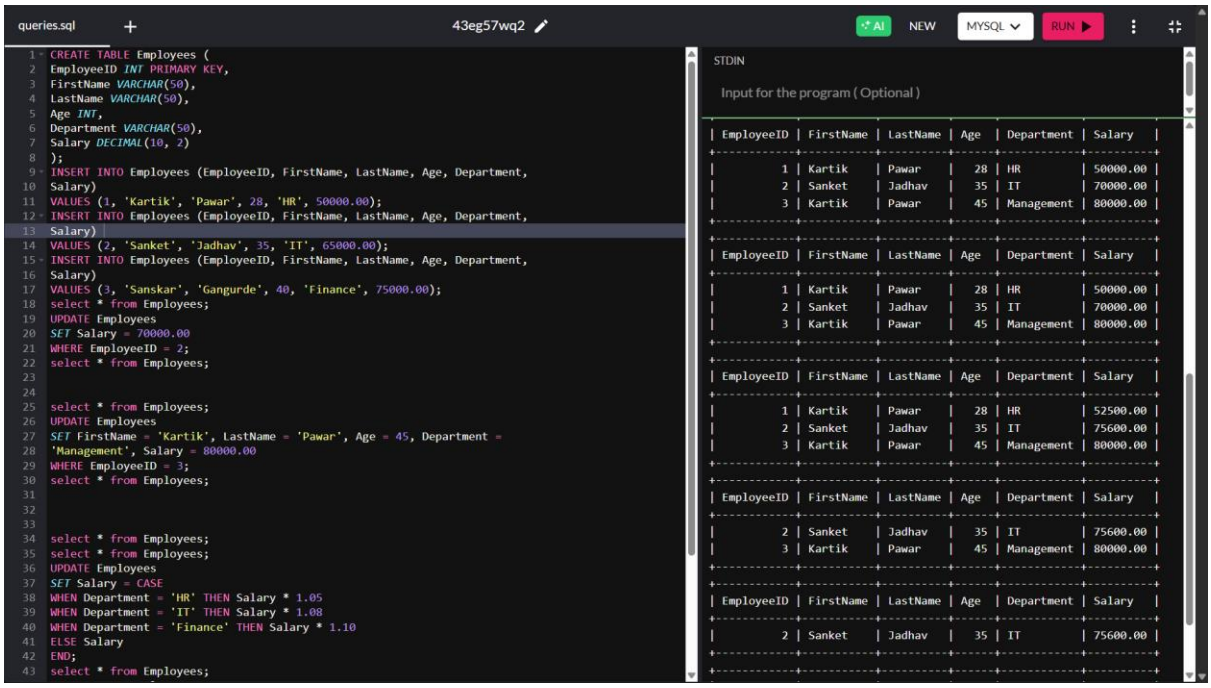
);

```
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: -





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
```

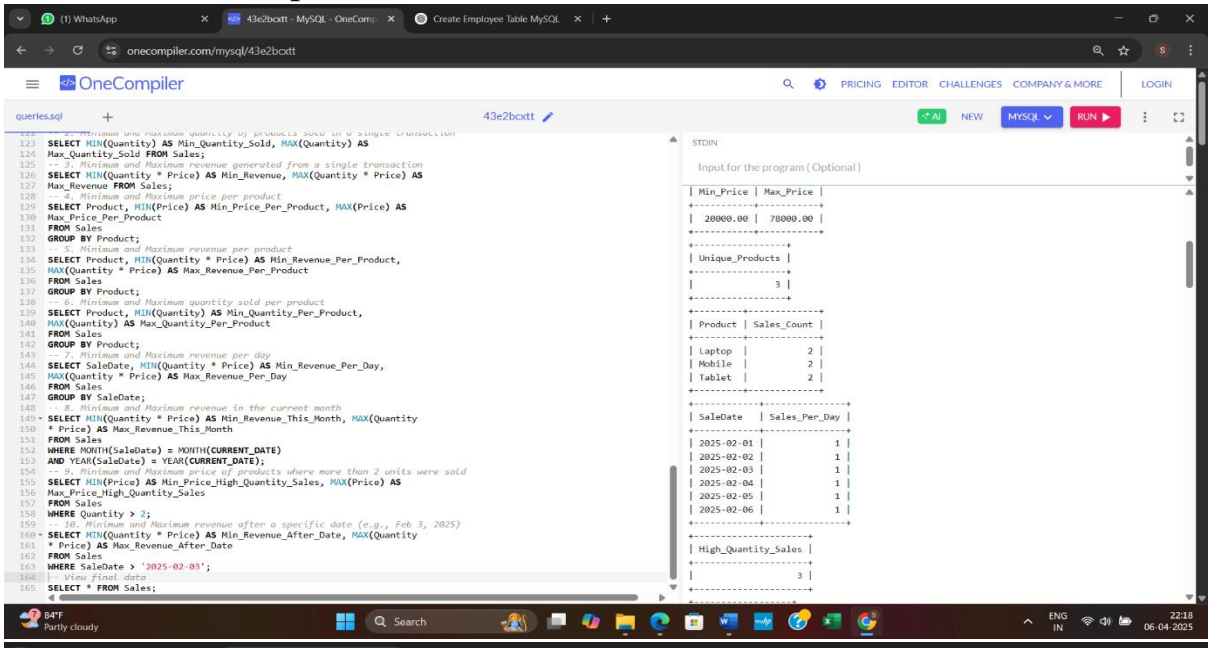
```
GROUP BY Product;

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;
```

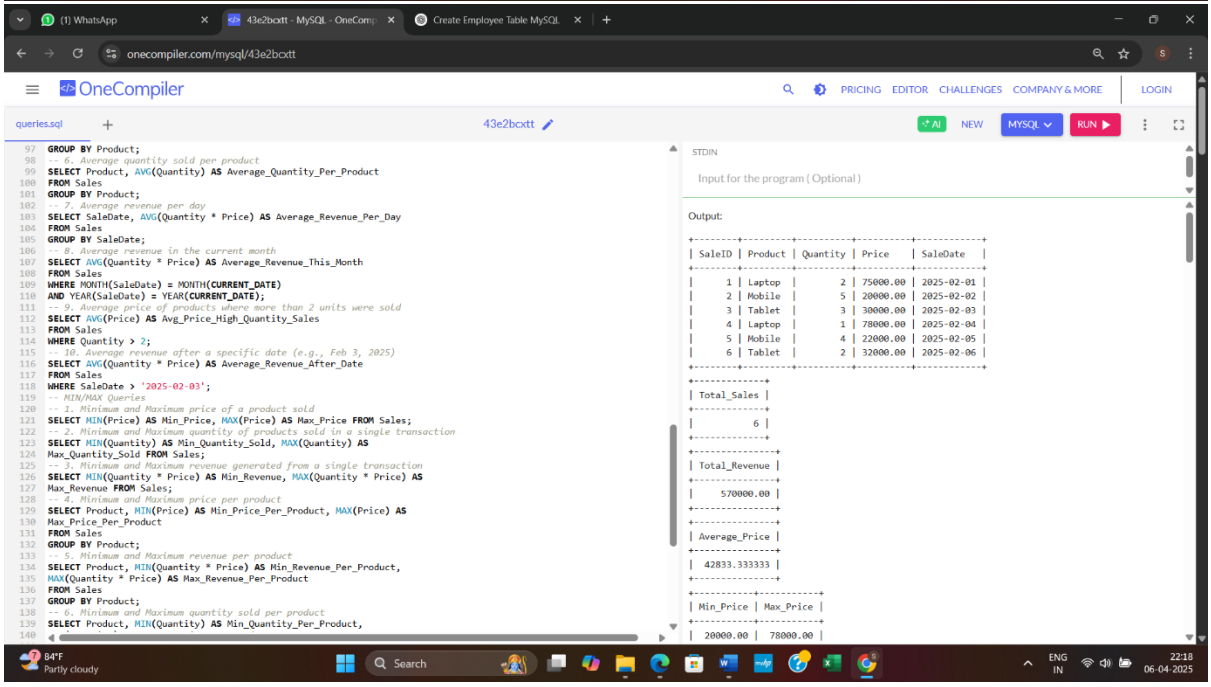
```
SELECT AVG(Quantity * Price) AS Average_Revenue_Per_Transaction
FROM Sales;

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
```

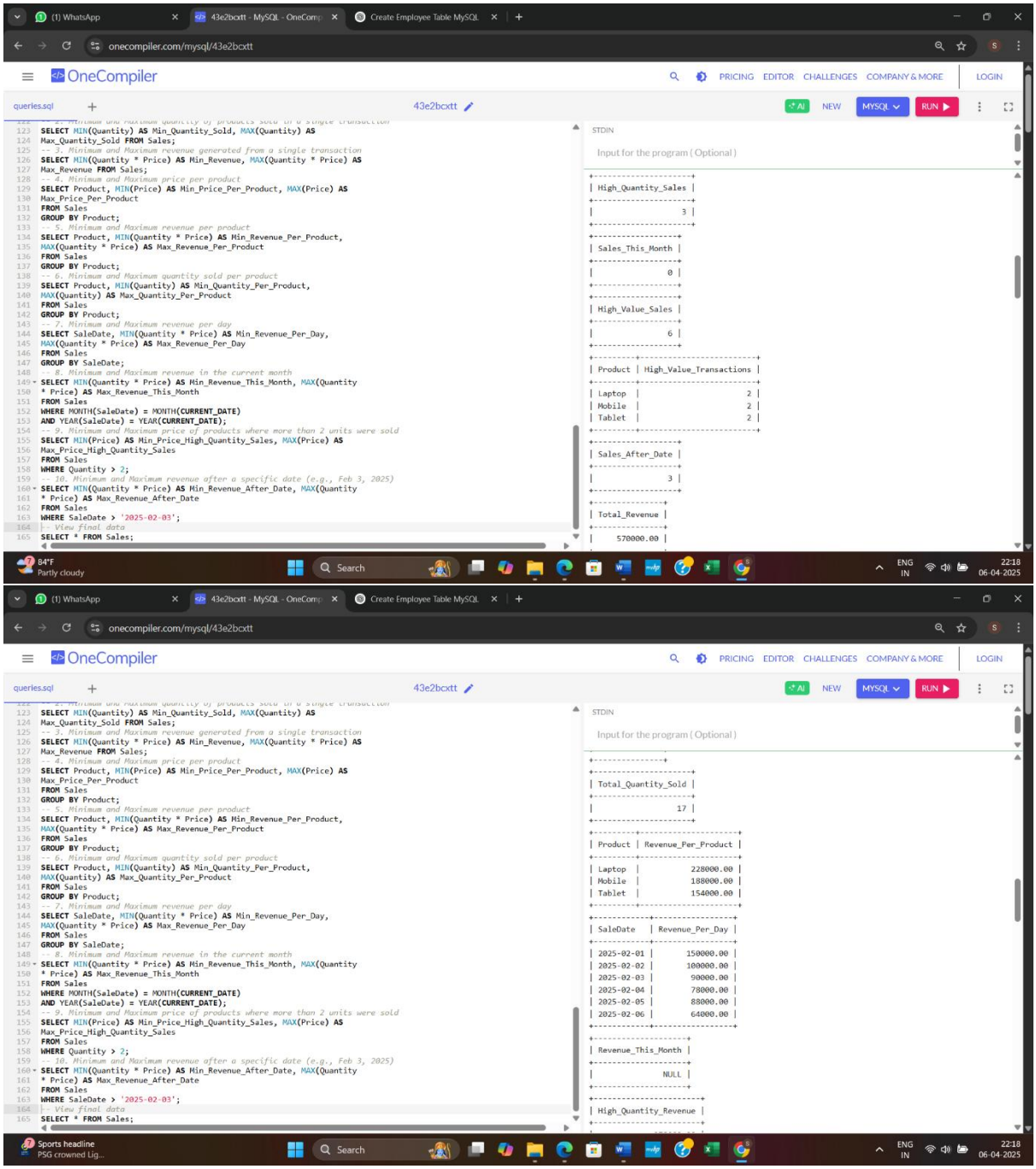
```
WHERE Quantity > 2;

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;
```

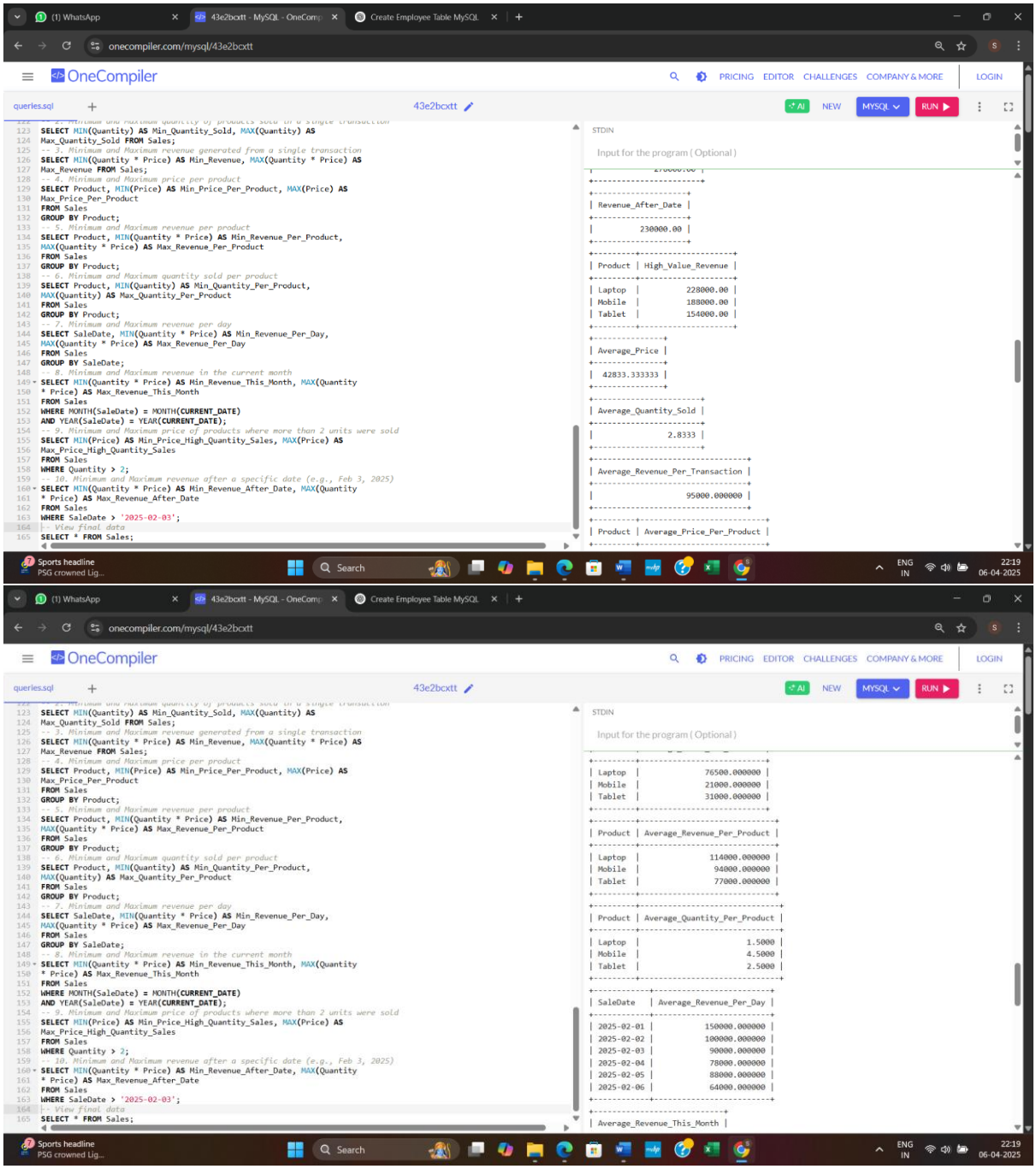
Screenshot of Output:





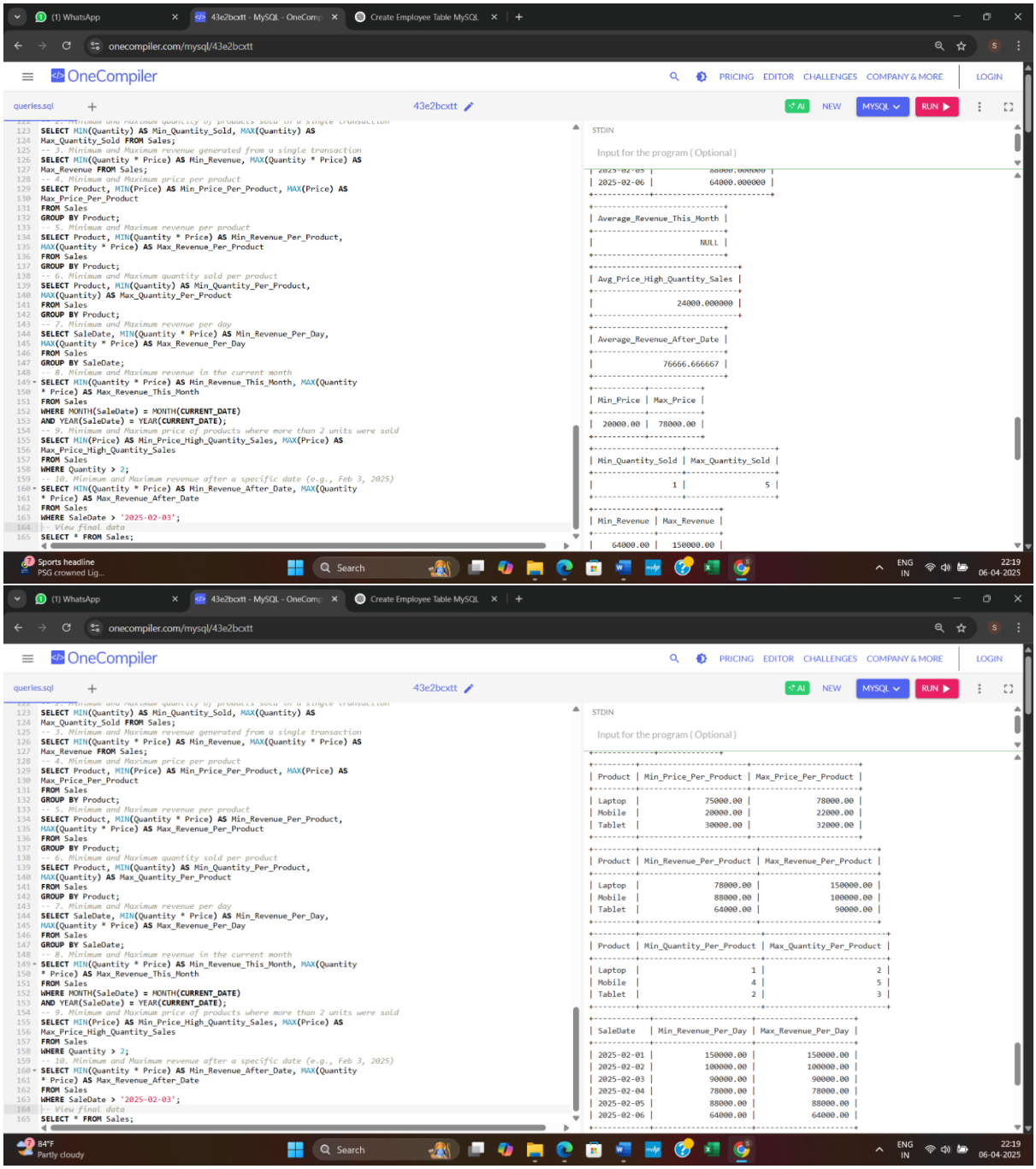

onecompiler.com/mysql/43e2bcxtt
OneCompiler
PRICING EDITOR CHALLENGES COMPANY & MORE LOGIN
queries.sql
43e2bcxtt
NEW
MYSQL
RUN
STDIN
Input for the program (Optional)
SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS Max_Quantity_Sold FROM Sales;
-- 3. Minimum and Maximum revenue generated from a single transaction
SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS Max_Revenue FROM Sales;
-- 4. 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;
-- 5. 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;
-- 6. 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;
-- 7. 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;
-- 8. 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);
-- 9. 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;
-- 10. 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';
-- View final data
SELECT * FROM Sales;
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
84°F
Partly cloudy
Search
ENG
IN
22:18
06-04-2025

onecompiler.com/mysql/43e2bcxtt
OneCompiler
PRICING EDITOR CHALLENGES COMPANY & MORE LOGIN
queries.sql
43e2bcxtt
NEW
MYSQL
RUN
STDIN
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...
Search
ENG
IN
22:18
06-04-2025



```
SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS
Max_Quantity_Sold FROM Sales;
-- 3. Minimum and Maximum revenue generated from a single transaction
SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS
Max_Revenue FROM Sales;
-- 4. 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;
-- 5. 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;
-- 6. 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;
-- 7. 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;
-- 8. 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);
-- 9. 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;
-- 10. 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';
-- View final data
SELECT * FROM Sales;
```

```
+---------------------+
| Revenue_After_Date |
+---------------------+
|          230000.00 |
+---------------------+

+---------+---------------------+
| Product | High_Value_Revenue |
+---------+---------------------+
| Laptop  |          228000.00 |
| Mobile  |          188000.00 |
| Tablet  |          154000.00 |
+---------+---------------------+

+---------------+
| Average_Price |
+---------------+
|  42833.333333 |
+---------------+

+-----------------------+
| Average_Quantity_Sold |
+-----------------------+
|                2.8333 |
+-----------------------+

+---------------------------------+
| Average_Revenue_Per_Transaction |
+---------------------------------+
|                95000.000000 |
+---------------------------------+

+---------+-----------------------------+
| Product | Average_Price_Per_Product |
+---------+-----------------------------+
| Laptop  |                 76500.000000 |
| Mobile  |                 21000.000000 |
| Tablet  |                 31000.000000 |
+---------+-----------------------------+

+---------+-------------------------------+
| Product | Average_Revenue_Per_Product |
+---------+-------------------------------+
| Laptop  |                 114000.000000 |
| Mobile  |                  94000.000000 |
| Tablet  |                  77000.000000 |
+---------+-------------------------------+

+---------+-------------------------------+
| Product | Average_Quantity_Per_Product |
+---------+-------------------------------+
| Laptop  |                        1.5000 |
| Mobile  |                        4.5000 |
| Tablet  |                        2.5000 |
+---------+-------------------------------+

+------------+--------------------------+
| SaleDate   | Average_Revenue_Per_Day |
+------------+--------------------------+
| 2025-02-01 |           150000.000000 |
| 2025-02-02 |           100000.000000 |
| 2025-02-03 |            90000.000000 |
| 2025-02-04 |            78000.000000 |
| 2025-02-05 |            88000.000000 |
| 2025-02-06 |            64000.000000 |
+------------+--------------------------+

+-----------------------------+
| Average_Revenue_This_Month |
+-----------------------------+
```



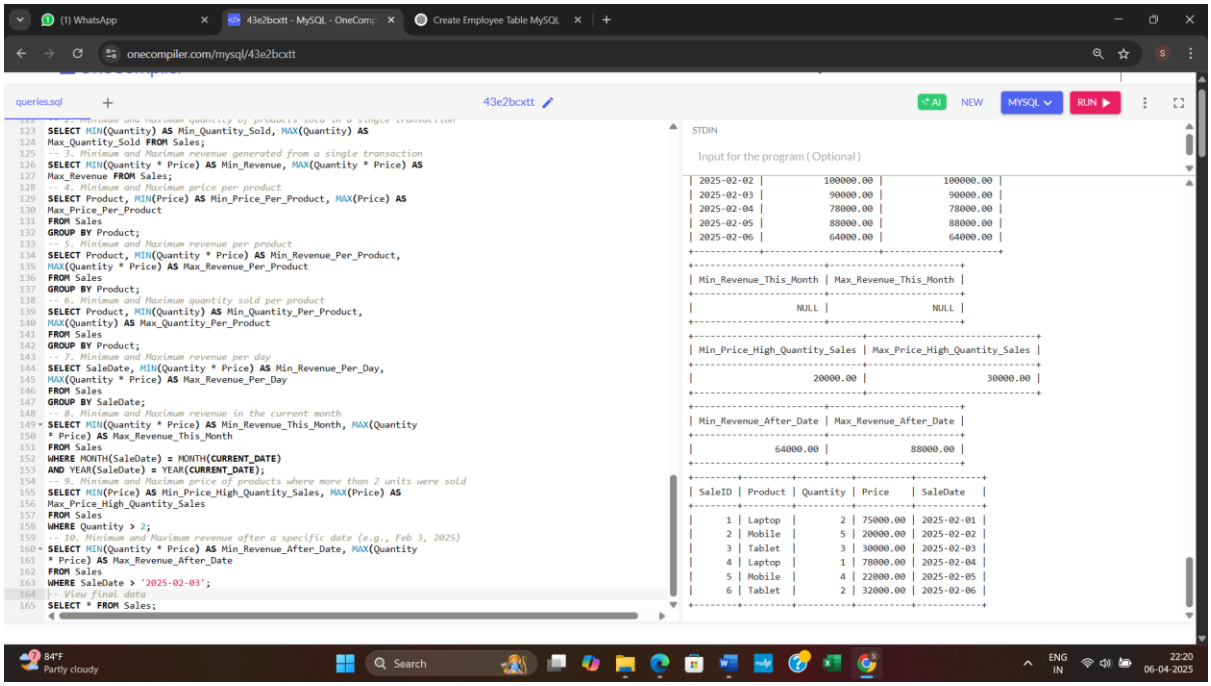
```
SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS
Max_Quantity_Sold FROM Sales;
-- 3. Minimum and Maximum revenue generated from a single transaction
SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS
Max_Revenue FROM Sales;
-- 4. Minimum and Maximum price per product
SELECT Product, MIN(Price) AS Min_Price_Per_Product, MAX(Price) AS
Max_Price_Per_Product
FROM Sales
GROUP BY Product;
-- 5. 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;
-- 6. 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;
-- 7. 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;
-- 8. 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);
-- 9. 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;
-- 10. 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';
-- View final data
SELECT * FROM Sales;
```

```
| 2025-02-06 |            64000.000000 |
+------------+--------------------------+

+-----------------------------+
| Average_Revenue_This_Month  |
+-----------------------------+
|                        NULL |
+-----------------------------+

+-------------------------------+
| Avg_Price_High_Quantity_Sales |
+-------------------------------+
|                  24000.000000 |
+-------------------------------+

+-----------------------------+
| Average_Revenue_After_Date  |
+-----------------------------+
|                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 |
+-------------+-------------+
```



```
SELECT MIN(Quantity) AS Min_Quantity_Sold, MAX(Quantity) AS
Max_Quantity_Sold FROM Sales;
-- 3. Minimum and Maximum revenue generated from a single transaction
SELECT MIN(Quantity * Price) AS Min_Revenue, MAX(Quantity * Price) AS
Max_Revenue FROM Sales;
-- 4. 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;
-- 5. 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;
-- 6. 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;
-- 7. 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;
-- 8. 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);
-- 9. 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;
-- 10. 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';
-- View final data
SELECT * FROM Sales;
```


```
+---------+-------------------------+-------------------------+
| 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  |                  78000.00 |                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 |
+------------+-------------------------+-------------------------+
```



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

AI NEW MYSQL RUN

```sql
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;
```

STDIN

Input for the program (Optional)

Output:

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

queries.sql

43eg57wq2

AI NEW MYSQL RUN

```sql
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;
```

STDIN

Input for the program (Optional)

```
|           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 | Sanskar       |      102 | 2024-01-02 |
|           4 | David         |      103 | 2024-01-03 |
+-------------+---------------+----------+------------+
+-------------+---------------+----------+------------+
| customer_id | customer_name | order_id | order_date |
+-------------+---------------+----------+------------+
|           1 | Snaket        |      101 | 2024-01-01 |
|           2 | Sanskar       |      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 | Sanskar       |      102 | 2024-01-02 |
|           4 | David         |      103 | 2024-01-03 |
+-------------+---------------+----------+------------+
```