

MYSQL PRACTICE QUESTIONS

SUBMITTED BY:-

STUDENT NAME -: PALLAVIUPADHYAY

ENROLL NO-: 23BCT0008

SUBJECT -: DATABASE DESIGN (MYSQL)

1. Insert a new student with the following details: ID = 2, Name = 'Jane Smith', Age = 22, and Grade = 'A' into the student table.

QUERY-->

```
INSERT INTO student VALUE(2 , 'Jane Smith', 22, 'A' );
```

INPUT:

```
1  -- 1. Insert a new student with the following details: ID = 2, Name = 'Jane Smith', Age =22, and Grade = 'A' into the student table.
2
3  • CREATE DATABASE college;
4  • SHOW DATABASES;
5  • Use college;
6
7  • CREATE TABLE student(
8    ID int PRIMARY KEY NOT NULL,
9    S_Name varchar(50) NOT NULL,
10   Age INT NOT NULL,
11   grade char(3) NOT NULL);
12
13  • INSERT INTO student VALUE(2 , 'Jane Smith', 22, 'A' );
14  • SELECT* from student;
```

OUTPUT:

	ID	S_Name	Age	grade
▶	2	Jane Smith	22	A
●	NULL	NULL	NULL	NULL

2. Insert a record into an employees table with columns emp_id, emp_name, department, and salary. Insert emp_id = 101, emp_name = 'Alice', department = 'HR', and salary = 50000.

QUERY ->

```
25 • INSERT INTO employee VALUE(101 , 'Alice ', 'HR', 50000 );
```

INPUT:

```
16 -- 2 Insert a record into an employees table with columns emp_id, emp_name, department, and salary. Insert emp_id = 101, emp_name = 'Alice', department = 'HR', and salary = 50000.
17 • CREATE DATABASE company;
18 • USE company;
19 • CREATE table employee(
20     emp_id INT PRIMARY KEY NOT NULL,
21     emp_name VARCHAR(50) NOT NULL,
22     department VARCHAR(50) NOT NULL,
23     salary FLOAT NOT NULL);
24
25 • INSERT INTO employee VALUE(101 , 'Alice ', 'HR', 50000 );
26 • SELECT * from employee;
```

OUTPUT:

	emp_id	emp_name	department	salary
▶	101	Alice	HR	50000
•	NULL	NULL	NULL	NULL

-
3. Insert multiple records into the student table in a single query. Insert one student with ID = 3, Name = 'Mark', Age = 23, Grade = 'B' and another student with ID = 4, Name = 'Lisa', Age = 21, Grade = 'A'.

QUERY ->

```
32 • INSERT INTO student (ID, S_Name, Age, grade) VALUES ( 3, 'Mark', 23, 'B'), (4, 'Lisa', 21, 'A');
```

INPUT:

```

28 -- 3. Insert multiple records into the student table in a single query. Insert one student with
29 -- ID = 3, Name = 'Mark', Age = 23, Grade = 'B' and another student with ID = 4, Name
30 -- = 'Lisa', Age = 21, Grade = 'A'.
31 • use college;
32 • INSERT INTO student (ID, S_Name, Age, grade) VALUES ( 3, 'Mark', 23, 'B'), (4, 'Lisa', 21, 'A');
33 • SELECT * FROM student;

```

OUTPUT:

	ID	S_Name	Age	grade
▶	2	Jane Smith	22	A
	3	Mark	23	B
	4	Lisa	21	A
✱	NULL	NULL	NULL	NULL

-
4. Add a new book to the books table with BookID = 101, Title = 'MySQL Essentials', Author = 'John Doe', Price = 29.99.

QUERY ->

```

46 • INSERT INTO book VALUE(101, 'MySQL Essentials', 'John Doe', 29.99);

```

INPUT:

```

35 -- 4. Add anewbook to the books table with BookID = 101, Title = 'MySQL Essentials',
36 -- Author = 'John Doe', Price = 29.99.
37
38 • CREATE DATABASE library;
39 • USE library;
40 • show DATABASES;
41 • CREATE TABLE book(
42     BookID INT PRIMARY KEY NOT NULL,
43     Title VARCHAR(50) NOT NULL,
44     Author VARCHAR(50) NOT NULL,
45     Price FLOAT NOT NULL);
46 • INSERT INTO book VALUE(101, 'MySQL Essentials', 'John Doe', 29.99);
47 • SELECT * FROM book;
48

```

OUPUT:

BookID	Title	Author	Price
101	MySQL Essentials	John Doe	29.99
NULL	NULL	NULL	NULL

5. Retrieve all the students from the student table

QUERY ->

```
51 • SELECT * FROM student;
```

INPUT:

```
49      -- 5. Retrieve all the students from the student table
50 •    USE college;
51 •    SELECT * FROM student;
```

OUTPUT:

	ID	S_Name	Age	grade
▶	2	Jane Smith	22	A
	3	Mark	23	B
	4	Lisa	21	A
*	NULL	NULL	NULL	NULL

6. Get the names and majors of all students who are 21 years old or older from the student table

	ID	S_Name	Age	grade	major
	3	Mark	23	B	CS
	4	Lisa	21	A	CS
	6	Ava Hernandez	19	A	CS
	403	Ethan Clark	21	A	CE
	404	Mia Patel	0	C	CS
	405	Noah Walker	22	B	CE
	407	Lucas Evans	21	B	CS
	NULL	NULL	NULL	NULL	NULL

QUERY->

```
-- 6.  Get the names and majors of all students who are 21 years old or older from the student table

SELECT S_NAME , MAJOR
FROM STUDENT
WHERE AGE >=21;
```

OUTPUT:

	S_NAME	MAJOR
▶	Mark	CS
	Lisa	CS
	Ethan Clark	CE
	Noah Walker	CE
	Lucas Evans	CS

7. Select all books from the books table that cost more than 20 dollars

QUERY ->

```
-- 7. Select all books from the books table that cost more than 20 dollars  
USE library;  
SELECT * FROM book WHERE price > 20;
```

OUTPUT:

	BookID	Title	Author	Price
▶	101	MySQL Essentials	John Doe	29.99
●	NULL	NULL	NULL	NULL

8. Retrieve all customers who have placed at least one order from the orders table.
Display their CustomerID, CustomerName, and the TotalAmount of their first order
(earliest by date).

CUSTOMERID	CUSTOMERNAME
101	John Doe
102	Jane Smith
103	Alex Brown
104	Michael King
105	Sarah Lee
NULL	NULL

CUSTOMERS TABLE

orderid	customerid	orderdate	totalamount	orderstatus
2001	101	2023-01-01	150.00	Pending
2002	101	2023-02-10	200.00	Pending
2003	102	2023-01-15	300.00	Complete
2004	103	2023-03-05	400.00	Pending
2005	102	2023-02-20	100.00	Pending
2006	105	2024-01-10	50.00	Pending
2007	104	2023-11-25	250.00	Pending
NULL	NULL	NULL	NULL	NULL

ORDERS TABLE

QUERY->

- ```

SELECT c.customerid, c.customername, o.totalamount
FROM customers c
JOIN (
 SELECT customerid, MIN(orderdate) AS first_order_date
 FROM orders
 GROUP BY customerid
) AS first_order ON c.customerid = first_order.customerid
JOIN orders o ON o.customerid = first_order.customerid AND o.orderdate = first_order.first_order_date;

```

OUTPUT:

| customerid | customername | totalamount |
|------------|--------------|-------------|
| 101        | John Doe     | 150.00      |
| 102        | Jane Smith   | 300.00      |
| 103        | Alex Brown   | 400.00      |
| 104        | Michael King | 250.00      |
| 105        | Sarah Lee    | 50.00       |

---

9. List the names of employees who earn more than the average salary in the employees table.

QUERY:-

```

SELECT emp_name
FROM employee
WHERE salary > (SELECT AVG(salary) FROM employee);

```

OUTPUT:-

| emp_name      |
|---------------|
| Olivia Harris |
| Emma Turner   |

---

10. Retrieve all products from the products table that have never been ordered. Display the ProductID, ProductName, and Price.

| ProductID | ProductName         | Price | StockQuantity | orderdate  |
|-----------|---------------------|-------|---------------|------------|
| 1         | Laptop              | 75000 | 10            | 2024-10-01 |
| 2         | Smartphone          | 25000 | 50            | 2024-09-25 |
| 3         | Headphones          | 1500  | 200           | 2024-09-20 |
| 4         | Tablet              | 30000 | 3             | 2024-10-05 |
| 5         | Smartwatch          | 5000  | 1             | 2024-09-30 |
| 6         | Monitor             | 12000 | 2             | 2024-09-28 |
| 7         | Keyboard            | 1200  | 150           | 2024-09-22 |
| 8         | Mouse               | 800   | 300           | 2024-10-03 |
| 9         | Printer             | 8000  | 20            | 2024-10-07 |
| 10        | External Hard Drive | 4000  | 60            | 2024-10-02 |
| NULL      | NULL                | NULL  | NULL          | NULL       |

**Product table**

| orderid | customerid | orderdate  | totalamount | orderstatus | productid |
|---------|------------|------------|-------------|-------------|-----------|
| 2001    | 101        | 2023-01-01 | 150.00      | Complete    | 1         |
| 2002    | 101        | 2023-02-10 | 200.00      | Complete    | 2         |
| 2003    | 102        | 2023-01-15 | 300.00      | Complete    | 3         |
| 2004    | 103        | 2023-03-05 | 400.00      | Complete    | 4         |
| 2005    | 102        | 2023-02-20 | 100.00      | Pending     | 5         |
| 2006    | 105        | 2024-01-10 | 50.00       | Pending     | 6         |
| 2007    | 104        | 2023-11-25 | 250.00      | Complete    | 7         |
| 3001    | 105        | 2024-01-15 | 250.50      | Pending     | NULL      |
| NULL    | NULL       | NULL       | NULL        | NULL        | NULL      |

**orders table**

**Query->**

```
SELECT p.productid, p.productname, p.price
FROM product p
LEFT JOIN orders o ON p.productid = o.productid
WHERE o.productid IS NULL;
```

**Output:-**

| productid | productname         | price |
|-----------|---------------------|-------|
| 8         | Mouse               | 800   |
| 9         | Printer             | 8000  |
| 10        | External Hard Drive | 4000  |
| 11        | PHONE               | 17000 |

**11. For each CustomerID, show the total number of orders they have placed, the total amount spent, and the average amount spent per order**

| orderid | customerid | orderdate  | totalamount | orderstatus |
|---------|------------|------------|-------------|-------------|
| 2001    | 101        | 2023-01-01 | 150.00      | Pending     |
| 2002    | 101        | 2023-02-10 | 200.00      | Pending     |
| 2003    | 102        | 2023-01-15 | 300.00      | Complete    |
| 2004    | 103        | 2023-03-05 | 400.00      | Pending     |
| 2005    | 102        | 2023-02-20 | 100.00      | Pending     |
| 2006    | 105        | 2024-01-10 | 50.00       | Pending     |
| 2007    | 104        | 2023-11-25 | 250.00      | Pending     |
| NULL    | NULL       | NULL       | NULL        | NULL        |

**QUERY->**

```

SELECT customerid,
 COUNT(orderid) AS total_orders,
 SUM(totalamount) AS total_spent,
 AVG(totalamount) AS avg_spent_per_order
FROM orders
GROUP BY customerid;

```

OUTPUT:

| customerid | total_orders | total_spent | avg_spent_per_order |
|------------|--------------|-------------|---------------------|
| 101        | 2            | 350.00      | 175.000000          |
| 102        | 2            | 400.00      | 200.000000          |
| 103        | 1            | 400.00      | 400.000000          |
| 104        | 1            | 250.00      | 250.000000          |
| 105        | 1            | 50.00       | 50.000000           |

12. Update the age of the student whose ID = 1 to 21 in the student table.

|   | ID   | S_Name        | Age  | grade |
|---|------|---------------|------|-------|
| ▶ | 1    | Jane Smith    | 22   | A     |
|   | 3    | Mark          | 23   | B     |
|   | 4    | Lisa          | 21   | A     |
|   | 6    | Ava Hernandez | 19   | A     |
|   | 403  | Ethan Clark   | 21   | A     |
|   | 404  | Mia Patel     | 0    | C     |
|   | 405  | Noah Walker   | 22   | B     |
|   | 407  | Lucas Evans   | 21   | B     |
| * | NULL | NULL          | NULL | NULL  |

ANS->

```

-- 12. Update the age of the student whose ID = 1 to 21 in the student table.
UPDATE student
Set ID = 21
WHERE ID = 1;
SELECT * FROM student;

```

OUTPUT:



| ID   | S_Name        | Age  | grade |
|------|---------------|------|-------|
| 4    | Lisa          | 21   | A     |
| 6    | Ava Hernandez | 19   | A     |
| 21   | Jane Smith    | 22   | A     |
| 403  | Ethan Clark   | 21   | A     |
| 404  | Mia Patel     | 0    | C     |
| 405  | Noah Walker   | 22   | B     |
| 407  | Lucas Evans   | 21   | B     |
| NULL | NULL          | NULL | NULL  |

13. Increase the price of all books by 10% in the books table.

| BookID | Title                  | Author         | Price |
|--------|------------------------|----------------|-------|
| 101    | MySQL Essentials       | John Doe       | 29.99 |
| 301    | The Silent Forest      | aniel Moore    | 18.5  |
| 303    | "Journey to the Stars" | Michael Carter | 9.2   |
| 304    | "The Hidden Truth"     | Olivia Brown   | 14.95 |
| 306    | "Beneath the Waves"    | Emma Roberts   | 12.6  |
| 307    | "Shadows of the Night" | William Davis  | 13.85 |
| NULL   | NULL                   | NULL           | NULL  |

QUERY->

```
UPDATE BOOK
SET PRICE = PRICE * 1.10;
```

OUTPUT:-

| BookID | Title                  | Author         | Price  |
|--------|------------------------|----------------|--------|
| 101    | MySQL Essentials       | John Doe       | 32.989 |
| 301    | The Silent Forest      | aniel Moore    | 20.35  |
| 303    | "Journey to the Stars" | Michael Carter | 10.12  |
| 304    | "The Hidden Truth"     | Olivia Brown   | 16.445 |
| 306    | "Beneath the Waves"    | Emma Roberts   | 13.86  |
| 307    | "Shadows of the Night" | William Davis  | 15.235 |
| NULL   | NULL                   | NULL           | NULL   |

14. Increase the salary of employees who have been in the employees table for more than 5 years by 10%. Assume there's a column HireDate in the table, and only those hired before 2019-09-30 should get the raise.

| emp_id | emp_name       | department | salary | DEPARTMENTID | HIREDATE   |
|--------|----------------|------------|--------|--------------|------------|
| 5      | Olivia Harris  | IT         | 68200  | 10           | 2016-05-10 |
| 101    | Alice          | HR         | 55000  | 1            | 2018-02-14 |
| 201    | Alice Green    | HR         | 49500  | 4            | 2015-11-30 |
| 206    | William Carter | Sales      | 52800  | 20           | 2018-02-14 |
| 207    | Emma Turner    | Operations | 53000  | 15           | 2017-08-21 |
| NULL   | NULL           | NULL       | NULL   | NULL         | NULL       |

QUERY-:

```
UPDATE employee
SET salary = salary * 1.10
WHERE hiredate < '2019-09-30';
SELECT * from employee;
```

OUTPUT-:

| emp_id | emp_name       | department | salary | DEPARTMENTID | HIREDATE   |
|--------|----------------|------------|--------|--------------|------------|
| 5      | Olivia Harris  | IT         | 68200  | 10           | 2016-05-10 |
| 101    | Alice          | HR         | 55000  | 1            | 2018-02-14 |
| 201    | Alice Green    | HR         | 49500  | 4            | 2015-11-30 |
| 206    | William Carter | Sales      | 52800  | 20           | 2018-02-14 |
| 207    | Emma Turner    | Operations | 53000  | 15           | 2017-08-21 |
| NULL   | NULL           | NULL       | NULL   | NULL         | NULL       |

15. Change the OrderStatus to 'Completed' for all orders in the orders table that were placed before 2023-12-31 and whose TotalAmount is greater than 100

| orderid | customerid | orderdate  | totalamount | orderstatus |
|---------|------------|------------|-------------|-------------|
| 2001    | 101        | 2023-01-01 | 150.00      | Pending     |
| 2002    | 101        | 2023-02-10 | 200.00      | Pending     |
| 2003    | 102        | 2023-01-15 | 300.00      | Complete    |
| 2004    | 103        | 2023-03-05 | 400.00      | Pending     |
| 2005    | 102        | 2023-02-20 | 100.00      | Pending     |
| 2006    | 105        | 2024-01-10 | 50.00       | Pending     |
| 2007    | 104        | 2023-11-25 | 250.00      | Pending     |
| NULL    | NULL       | NULL       | NULL        | NULL        |

QUERY -:

```

SET SQL_SAFE_UPDATES = 0;
UPDATE orders
SET orderstatus = 'Complete'
WHERE orderdate < '2023-12-31' AND totalamount > 100;
SELECT * FROM ORDERS;

```

OUTPUT:-

| orderid | customerid | orderdate  | totalamount | orderstatus |
|---------|------------|------------|-------------|-------------|
| 2001    | 101        | 2023-01-01 | 150.00      | Complete    |
| 2002    | 101        | 2023-02-10 | 200.00      | Complete    |
| 2003    | 102        | 2023-01-15 | 300.00      | Complete    |
| 2004    | 103        | 2023-03-05 | 400.00      | Complete    |
| 2005    | 102        | 2023-02-20 | 100.00      | Pending     |
| 2006    | 105        | 2024-01-10 | 50.00       | Pending     |
| 2007    | 104        | 2023-11-25 | 250.00      | Complete    |
| NULL    | NULL       | NULL       | NULL        | NULL        |

16. Delete the student with ID = 1 from the student table.

|   | ID   | S_Name        | Age  | grade |
|---|------|---------------|------|-------|
| ▶ | 1    | Jane Smith    | 22   | A     |
|   | 3    | Mark          | 23   | B     |
|   | 4    | Lisa          | 21   | A     |
|   | 6    | Ava Hernandez | 19   | A     |
|   | 403  | Ethan Clark   | 21   | A     |
|   | 404  | Mia Patel     | 0    | C     |
|   | 405  | Noah Walker   | 22   | B     |
|   | 407  | Lucas Evans   | 21   | B     |
| * | NULL | NULL          | NULL | NULL  |

QUERY->

- ```

-- -- 16. Delete the student with ID = 1 from the student table.
• DELETE FROM student WHERE ID = 1;
• SELECT * FROM student;

```

OUTPUT:

	ID	S_Name	Age	grade
▶	3	Mark	23	B
	4	Lisa	21	A
	6	Ava Hernandez	19	A
	403	Ethan Clark	21	A
	404	Mia Patel	0	C
	405	Noah Walker	22	B
	407	Lucas Evans	21	B
•	NULL	NULL	NULL	NULL

17. Remove all books from the books table that are priced below 15 dollar

BookID	Title	Author	Price
101	MySQL Essentials	John Doe	32.989
301	The Silent Forest	aniel Moore	20.35
303	"Journey to the Stars"	Michael Carter	10.12
304	"The Hidden Truth"	Olivia Brown	16.445
306	"Beneath the Waves"	Emma Roberts	13.86
307	"Shadows of the Night"	William Davis	15.235
NULL	NULL	NULL	NULL

ANS ->

```
DELETE FROM BOOK
WHERE PRICE < 15;
SELECT * FROM BOOK;
```

OUTPUT:-

BookID	Title	Author	Price
101	MySQL Essentials	John Doe	32.989
301	The Silent Forest	aniel Moore	20.35
304	"The Hidden Truth"	Olivia Brown	16.445
307	"Shadows of the Night"	William Davis	15.235
NULL	NULL	NULL	NULL

18. Delete all customers who have not placed any orders in the orders table. Ensure no orders exist for those customers before deleting.

CUSTOMERID	CUSTOMERNAME
101	John Doe
102	Jane Smith
103	Alex Brown
104	Michael King
105	Sarah Lee
NULL	NULL

QUERY-:

```
DELETE FROM customers
WHERE customerid NOT IN (SELECT DISTINCT customerid FROM orders);
select * from customers;
```

OUTPUT-:

CUSTOMERID	CUSTOMERNAME
101	John Doe
102	Jane Smith
103	Alex Brown
104	Michael King
105	Sarah Lee
NULL	NULL

19. Remove all products from the products table where the StockQuantity is less than 5 and the product has not been ordered in the past 6 months

ProductID	ProductName	Price	StockQuantity	orderdate
1	Laptop	75000	10	2024-10-01
2	Smartphone	25000	50	2024-09-25
3	Headphones	1500	200	2024-09-20
4	Tablet	30000	3	2024-10-05
5	Smartwatch	5000	1	2024-09-30
6	Monitor	12000	2	2024-09-28
7	Keyboard	1200	150	2024-09-22
8	Mouse	800	300	2024-10-03
9	Printer	8000	20	2024-10-07
10	External Hard Drive	4000	60	2024-10-02
NULL	NULL	NULL	NULL	NULL

QUERY->

```
DELETE FROM product
WHERE stockquantity < 5
AND productid NOT IN (
    SELECT DISTINCT productid
    FROM orders
    WHERE orderdate >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
);
```

OUTPUT:-

ProductID	ProductName	Price	StockQuantity	orderdate
1	Laptop	75000	10	2024-10-01
2	Smartphone	25000	50	2024-09-25
3	Headphones	1500	200	2024-09-20
7	Keyboard	1200	150	2024-09-22
8	Mouse	800	300	2024-10-03
9	Printer	8000	20	2024-10-07
10	External Hard Drive	4000	60	2024-10-02
11	PHONE	17000	6	2024-10-01
NULL	NULL	NULL	NULL	NULL

20. Create a new table called courses with the following columns: CourseID, CourseName, and Credits.

QUERY ->

```

57 • CREATE TABLE course(
58     CourseID INT PRIMARY KEY NOT NULL,
59     CourseName VARCHAR(50) NOT NULL,
60     Credits INT NOT NULL);

```

INPUT:

```

55 -- 20. Create a new table called courses with the following columns: CourseID, CourseName, and Credits.
56 • USE college;
57 • CREATE TABLE course(
58     CourseID INT PRIMARY KEY NOT NULL,
59     CourseName VARCHAR(50) NOT NULL,
60     Credits INT NOT NULL);

```

OUTPUT:

	Field	Type	Null	Key	Default	Extra
►	CourseID	int	NO	PRI	NULL	
	CourseName	varchar(50)	NO		NULL	
	Credits	int	NO		NULL	

21. Insert two new courses into the courses table. One with CourseID = 201, CourseName = 'Database Management', and Credits = 3, and another with CourseID = 202, CourseName = 'Web Development', and Credits = 4.

ANS->

```

64 • INSERT INTO course (CourseID, CourseName,Credits) VALUES (201,'Database Management', 3), (202, 'Web Development', 4);

```

INPUT:

```

62 -- 21. Insert two new courses into the courses table. One with CourseID = 201, CourseName = 'Database Management', and Credits = 3, and another with CourseID = 202, CourseName = 'Web Development', and Credits = 4.
63
64 • INSERT INTO course (CourseID, CourseName,Credits) VALUES (201,'Database Management', 3), (202, 'Web Development', 4);
65 • SELECT * From course;

```

OUTPUT:

	CourseID	CourseName	Credits
►	201	Database Management	3
	202	Web Development	4
•	NULL	NULL	NULL

- 22.** Insert a new record into the orders table with OrderID = 3001, CustomerID = 105, OrderDate = '2024-01-15', and TotalAmount = 250.50. Ensure that the CustomerID exists in the customers table.

orderid	customerid	orderdate	totalamount	orderstatus
2001	101	2023-01-01	150.00	Pending
2002	101	2023-02-10	200.00	Pending
2003	102	2023-01-15	300.00	Complete
2004	103	2023-03-05	400.00	Pending
2005	102	2023-02-20	100.00	Pending
2006	105	2024-01-10	50.00	Pending
2007	104	2023-11-25	250.00	Pending
NULL	NULL	NULL	NULL	NULL

QUERY:-

```
INSERT INTO orderS (orderid, customerid, orderdate, totalamount, orderstatus)
VALUES (3001, 105, '2024-01-15', 250.50, 'Pending');
SELECT * FROM ORDERS;
```

OUTPUT:-

orderid	customerid	orderdate	totalamount	orderstatus
2001	101	2023-01-01	150.00	Complete
2002	101	2023-02-10	200.00	Complete
2003	102	2023-01-15	300.00	Complete
2004	103	2023-03-05	400.00	Complete
2005	102	2023-02-20	100.00	Pending
2006	105	2024-01-10	50.00	Pending
2007	104	2023-11-25	250.00	Complete
3001	105	2024-01-15	250.50	Pending

- 23.** the employee who earns the highest salary in each department. Assume there's a DepartmentID in the employees table

emp_id	emp_name	department	salary	DEPARTMENTID
5	Olivia Harris	IT	62000	10
101	Alice	HR	50000	1
201	Alice Green	HR	45000	4
206	William Carter	Sales	48000	20
207	Emma Turner	Operations	53000	15
NULL	NULL	NULL	NULL	NULL

QUERY:-


```

SELECT emp_id, emp_name, department, salary
FROM employee e
WHERE salary = (
    SELECT MAX(salary)
    FROM employee
    WHERE department = e.department
);

```

OUTPUT:-

emp_id	emp_name	department	salary
5	Olivia Harris	IT	62000
101	Alice	HR	50000
206	William Carter	Sales	48000
207	Emma Turner	Operations	53000
NULL	NULL	NULL	NULL

24. Display the CustomerID and TotalAmount for the top 5 largest orders placed in the orders table.

customerID	customerName	TotalAmount	orderStatus	orderdate
101	John Smith	150.75	completed	2024-10-01
102	Emily Johnson	99.5	Pending	2024-10-03
103	Robert Brown	200	completed	2024-10-05
104	Sarah Williams	350.25	Shipped	2024-10-02
NULL	NULL	NULL	NULL	NULL

QUERY->

```

SELECT customerID , totalamount FROM order_ ORDER BY totalamount DESC LIMIT 5;

```

OUTPUT:-

customerId	totalamount
104	350.25
103	200
101	150.75
102	99.5

*******COMPLETE*******
