QUESTION-1

Create database company_db;

USE company_db;

CREATE TABLE employees(

Employee_id INT primary key,

First_name VARCHAR(50),

Last_name VARCHAR(50),

Department VARCHAR (50),

Salary INT,

Hire_date DATE);

QUESTION-2

INSERT INTO employee

(“employee_id”, “first_name”, “last_name”, “department”, “salary”, “hire_date”)

VALUES

(101, “amit”, “sharma” , “hr” ,50000, “2020-01-15”),

(102, “riya”, “Kapoor” , “sales” , 75000, “2019-03-22”),

(103, “raj” , “mehta” , “it” ,90,000 , “2018-07-11”),

(104, “ neha” , “verma” , “it” , 85000 , “2021-09-01),

(105, “arjun”, “singh”, “ finance”, 60000, “2022-02-10”);

to check

SELECT * FROM EMPLOYEE;

QUESTION -3

Select * from employees

ORDER BY salary ASC;

QUESTION -4

Select * from employees

ORDER BY department ASC , salary DESC;

QUESTION -5

Select*from employees

Where department= “IT”

Order by hire_date DESC;

QUESTION-6

CREATE TABLE sales (

Sale_id INTprimary key,

Customer_name VARCHAR (50),

Amount INT,

Sale_date DATE

);

INSERT INTO sales

(sale_id,customer_name,amount,sale_date)

Values

(1, “ADITI” , 1500, “2024-08-01”),

(2, “ROHAN” , 2200, “2024-08-03”),

(3, “ADITI” , 3500 , “2024-09-05”),

(4, “MEENA”, 2700, “2024-09-15”),

(5, “ROHAN” , 4500, “2024-09-25”);

QUESTION-7

Select * from sales

ORDER BY amount DESC;

QUESTION-8

Select *from sales

WHERE customer_name= “ADITI”;

QUESTION -9

PRIMARY KEY	FOREIGN KEY
• Uniquely identifies each Record in a table	* Refers to the primary key of another table
• Cannot have duplicate Values	* can have duplicate values
• Cannot be null	* can be null
• Ensures entity integrity	* Ensures referential integrity
• One per table	* can be multiple in a table

QUESTION -10

A **constraint** tells the database *what is allowed and what is not allowed* when inserting, updating, or deleting data.

Example idea:

“This column must never be empty” or
“Every student ID must be unique”

These rules are enforced **automatically by the database**.

Why Are Constraints Used?

Constraints are used to:

1. **Maintain data integrity**
   Prevent invalid or incorrect data from entering the database.
2. **Ensure consistency**
   Make sure relationships between tables stay correct.
3. **Reduce errors**
   Catch mistakes at the database level instead of relying only on application code.
4. **Enforce business rules**
   Example: an age must be greater than 0, or a username must be unique.

Common Types of SQL Constraints

1. NOT NULL

Ensures a column cannot have a NULL value.

name VARCHAR(50) NOT NULL

📌 Use when a value is required.

---

2. UNIQUE

Ensures all values in a column are different.

email VARCHAR(100) UNIQUE

📌 Use for usernames, emails, IDs, etc.

---

3. PRIMARY KEY

A combination of NOT NULL + UNIQUE.
Uniquely identifies each row in a table.

id INT PRIMARY KEY

📌 Every table should have one primary key.

---

4. FOREIGN KEY

Links one table to another and enforces relationships.

FOREIGN KEY (student_id) REFERENCES students(id)

📌 Prevents invalid references between tables.

---

5. CHECK

Ensures values meet a specific condition.

age INT CHECK (age >= 18)

📌 Use for rules like ranges or conditions.

---

6. DEFAULT

Sets a default value if none is provided.

status VARCHAR(20) DEFAULT 'active'

📌 Useful for optional values.

Simple Example

```
CREATE TABLE Students (
    id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    age INT CHECK (age >= 5)
);
```

This table ensures:

- Every student has a unique ID
- Name cannot be empty
- Email must be unique
- Age must be 5 or older