Q1:- Create a New Database and Table for Employees.
Task : Create a new database named and Create a table named with the following Columns:

Solution :- CREATE DATABASE company_db;

```
CREATE TABLE employees (
employee_id INT AUTO_INCREMENT PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
department VARCHAR(50),
salary INT,
hire_date DATE
);
```

Q2 :- Insert Data into Employees Table.
Task :- Insert the following sample records into the employees table.

Solution :- CREATE TABLE employees (
```
employee_id INT AUTO_INCREMENT PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
department VARCHAR(50),
salary INT,
hire_date DATE
);

INSERT INTO employees VALUES
(101, 'Amit', 'Sharma', 'HR', 50000, "2020-01-15"),
(102, 'Riya', 'Kapoor', 'Sales', 75000, "2019-03-22"),
(103, 'Raj', 'Mehta', 'IT', 90000, "2018-07-11"),
(104, 'Nehna', 'Verma', 'IT', 85000, "2021-09-01"),
(105, 'Arjun', 'Singh', 'Finance', 60000, "2022-02-10");

SELECT * FROM employees;
```

Q3 :- Display All Employee Records Sorted by Salary (Lowest to Highest).

Solution :- SELECT * FROM employees
```
ORDER BY salary ASC;
```

Q4 :- Show Employees Sorted by Department (A–Z) and Salary (High → Low).

Solution :- SELECT

employee_id, first_Name, last_Name, department, salary
FROM employees
ORDER BY Department ASC, Salary DESC;

Q5 :- List All Employees in the IT Department, Ordered by Hire Date (Newest First).

Solution :- SELECT * FROM employees
WHERE department = 'IT'
ORDER BY hire_date DESC;

Q6 :- Create and Populate a Sales Table.
Task :- Create a table to track sales data:

Solution :- CREATE TABLE Sales (
sale_id INT AUTO_INCREMENT PRIMARY KEY,
customer_name VARCHAR(50),
amount INT,
sale_date DATE
);

INSERT INTO Sales (customer_name, amount, sale_date) VALUES
('Aditi', 1500, "2024-08-01"),
('Rohan', 2200, "2024-08-03"),
('Aditi', 3500, "2024-09-05"),
('Meena', 2700, "2024-09-15"),
('Rohan', 4500, "2024-09-25");

SELECT * FROM Sales;

Q7 :- Display All Sales Records Sorted by Amount (Highest → Lowest).

Solution :- SELECT * FROM Sales
ORDER BY amount DESC;

Q8 :- Show All Sales Made by Customer “Aditi”.

Solution :- SELECT * FROM Sales
WHERE customer_name = 'Aditi';

Q9 :- What is the Difference Between a Primary Key and a Foreign Key?

Solution :-

Feature	**Primary Key (PK)**	**Foreign Key (FK)**

Purpose	To uniquely identify a specific record in the table.	To link two tables together and enforce referential integrity.
Uniqueness	Must be unique. No two rows can have the same PK.	Can be duplicated. Multiple rows can share the same FK value (e.g., many orders from one customer).
Null Values	Never. A Primary Key cannot be NULL.	Allowed. A Foreign Key can be NULL (unless explicitly restricted).
Count	Only one Primary Key is allowed per table.	Multiple Foreign Keys are allowed in a single table.
Indexing	Automatically indexed by the database for speed.	Not automatically indexed (usually requires manual indexing for performance).

Summary

- Use a **Primary Key** when you need to ensure every row is unique.
- Use a **Foreign Key** when you need to connect that row to a record in a different table.

Q10 :- What Are Constraints in SQL and Why Are They Used?

Solution :- In SQL, **constraints** are rules applied to columns in a table. They are used to limit the type of data that can go into a table.
Think of them as "gatekeepers" or "quality control" for your database. If you try to enter data that violates a constraint, the database will reject the action (giving you an error) to protect the data's integrity.

Why Are They Used?

1. **Data Integrity:** They ensure that the data is accurate and reliable.
2. **Consistency:** They prevent invalid data from being entered (e.g., stopping someone from entering `-500` as a salary).
3. **Uniqueness:** They ensure specific records don't have duplicates (e.g., two users cannot have the same email address).

Common SQL Constraints

1. PRIMARY KEY

Uniquely identifies each record in a table. It cannot be `NULL` and must be unique.

- **Use case:** Student ID, Order ID, Social Security Number.
- *Example:* No two students can have the same Student ID.

2. FOREIGN KEY

Links two tables together. It ensures that the data in one table matches a valid record in another table.

- **Use case:** An "Orders" table has a `CustomerID` that points to the "Customers" table.
- *Example:* You cannot add an order for a customer who doesn't exist in the database.

3. NOT NULL

Ensures that a column cannot have a `NULL` (empty) value.

- **Use case:** Usernames or passwords in a login system.
- *Example:* A user must provide a password; this field cannot be left blank.

5. CHECK

Ensures that the values in a column satisfy a specific condition.

- **Use case:** Age limits or price validation.
- *Example:* `CHECK (Age >= 18)` ensures no one under 18 is added to the table.

6. DEFAULT

Sets a default value for a column if no value is specified.

- **Use case:** Setting a default country or status.
- *Example:* If a user doesn't select a country, the database automatically saves it as "USA".