1. Write a SQL query to create Employee & Department table with specific columns and constraints.(**Note**- Create Table, insert data as per your Choice)

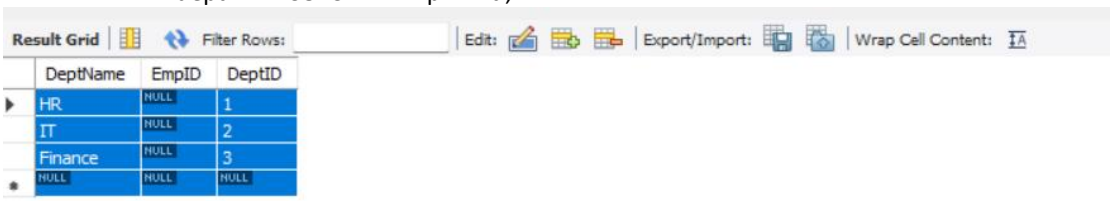
```
>>show databases;
use employeemanagementsystem;
show tables;
desc dept;
desc employee;
select * from dept;
select * from employee;
```

```
1  Alice   HR      30000
2  Bob     IT      45000
3  Charlie HR      40000
4  David   Finance 55000
5  Eva     IT      60000
6  Frank   Finance 50000
7  Grace   HR      35000
8  Hank    IT      70000
9  Ivy     Finance 45000
10 Jack    IT      40000
```

```
1 HR
2 IT
3 Finance
```

2. Write a SQL query to add any new column to an existing table.

```
>>ALTER TABLE dept ADD COLUMN EmpID int ;
```



3. Write a SQL query to insert multiple records (At least 5) into a table in a single operation.

```
>>INSERT INTO employee (EmpID, name, Dept, salary)
VALUES
 (11, 'Alice', 'IT', 50000),
 (12, 'Bob', 'HR', 60000),
 (13, 'Charlie', 'Finance', 30000),
 (14, 'Diana', 'IT', 100000),
 (15, 'Ethan', 'HR', 20000);
```

4. Write a SQL query to delete all records from a table.

```
>> delete  from employee where Name='frank';
```

5. Write a SQL query to update salary records from a table where id = 5

```
>>UPDATE employee
SET salary = 30000
WHERE EmpID = 5;
```

6. Write a query to grant SELECT privilege on the Emp table to user readonly_user

```
>>GRANT SELECT ON employeemanagementsystem.employeeTO 'readonly_user'@'localhost';
```

7. Write a query to revoke INSERT privilege on the Emp table from user temp_user.

```
>>REVOKE INSERT ON employeemanagementsystem.employeeFROM 'temp_user'@'localhost';
```

8. Write a query to insert a new employee into the Emp table and commit the transaction

```
>>INSERT INTO Employee (EmpID, Name, dept, Salary)
VALUES (16, 'Rahul Sharma', 2, 50000);

-- Commit the transaction to make the insertion permanent
COMMIT;
```

9. Write a query to update the Salary of an employee with EmpID = 101 and commit the transaction.

```
>>UPDATE Employee
SET Salary = 60000
WHERE EmpID = 10;

-- Commit the transaction
COMMIT;
```

10. Write a query to set a SAVEPOINT before updating the DeptID of an employee.

```
>>START TRANSACTION;

-- Set a savepoint before updating
SAVEPOINT before_dept_update;

-- Update the DeptID of the employee
UPDATE Employee
SET dept = 3
WHERE EmpID = 7;

-- Commit the transaction to make changes permanent
COMMIT;
```

11. Write a query to rollback to the previously created SAVEPOINT.

```
START TRANSACTION;

-- Set a savepoint
SAVEPOINT before_dept_update;

-- Make an update
```

```
UPDATE Employee
SET dept = 2
WHERE EmpID = 9;
-- Rollback to the savepoint if needed
ROLLBACK TO SAVEPOINT before_dept_update;

-- Commit the transaction to save changes permanently
COMMIT;
```

12. Write a query to truncate all data from the Emp table.

```
TRUNCATE TABLE Employee;
```

13. Write a query to list all employees whose salary is greater than 50,000

```
92 •  SELECT EmpID, Name, Salary
93    FROM Employee
94    WHERE Salary > 50000;
95
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

EmpID	Name	Salary
12	Bob	60000
14	Diana	100000
NULL	NULL	NULL

14. Write a query to increase the salary of all employees in Dept = 'IT' by 10%.

```
>>UPDATE Employee
SET Salary = Salary * 1.10
WHERE dept = 'IT';
```

15. Write a query to count the total number of employees in each department.

```
>>SELECT dept, COUNT(*) AS Total_Employees
FROM Employee
GROUP BY dept;
```

16. Write a query to list departments that have no employees. & create a view for those
salespeople who belong to the city of Mumbai.

```
>>drop  table employee;
drop table dept;

-- Create the new table for following operation.
CREATE TABLE Emp (
   EmpID INT PRIMARY KEY,
   EmpName VARCHAR(50),
   DeptID INT,
   Salary DECIMAL(10,2),
   Job VARCHAR(50),
   City VARCHAR(50),
   FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);
CREATE TABLE Department (
   DeptID INT PRIMARY KEY,
   DeptName VARCHAR(50)
);
```

```
INSERT INTO Department (DeptID, DeptName)
VALUES
(10, 'Sales'),
(20, 'HR'),
(30, 'IT'),
(40, 'Finance');

INSERT INTO Department (DeptID, DeptName)
VALUES
(10, 'Sales'),
(20, 'HR'),
(30, 'IT'),
(40, 'Finance');

INSERT INTO Emp (EmpID, EmpName, DeptID, Salary, Job, City)
VALUES
(101, 'Rahul', 10, 55000, 'Salesperson', 'Mumbai'),
(102, 'Priya', 10, 60000, 'Salesperson', 'Pune'),
(103, 'Amit', 20, 48000, 'HR Executive', 'Delhi'),
(104, 'Neha', 30, 75000, 'Developer', 'Mumbai');


SELECT D.DeptID, D.DeptName
FROM Department D
LEFT JOIN Emp E ON D.DeptID = E.DeptID
WHERE E.EmpID IS NULL;


CREATE VIEW Mumbai_Salespeople AS
SELECT EmpID, EmpName, DeptID, Salary, City
FROM Emp
WHERE City = 'Mumbai' AND Job = 'Salesperson';


SELECT * FROM Mumbai_Salespeople;
```

17. Write a query to display the highest salary among all employees.

```
>>SELECT EmpID, EmpName, Salary
FROM Emp
WHERE Salary = (SELECT MAX(Salary) FROM Emp);
```

18. Write a query to display the 2nd highest salary among all employees.

```
>>SELECT MAX(Salary) AS Second_Highest_Salary
FROM Emp
WHERE Salary < (SELECT MAX(Salary) FROM Emp);
```

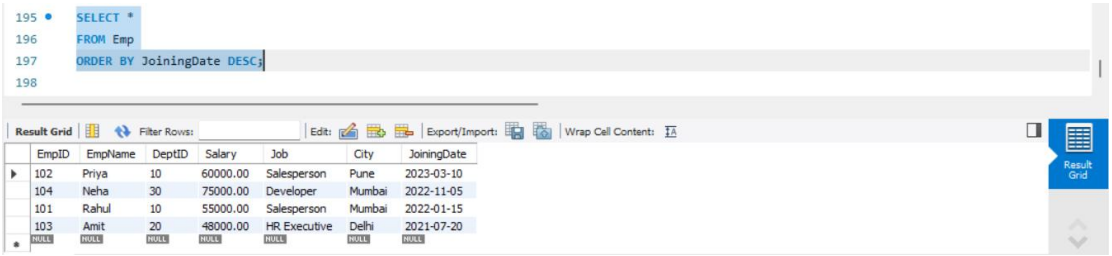
19. Write a query to show all employees ordered by JoiningDate in descending order

```
>>ALTER TABLE Department
RENAME COLUMN DeptName TO DepartmentName;

ALTER TABLE Emp
ADD JoiningDate DATE;

desc Emp;
select * from Emp;
```

```
UPDATE Emp
SET JoiningDate = CASE EmpID
   WHEN 101 THEN '2022-01-15'
   WHEN 102 THEN '2023-03-10'
   WHEN 103 THEN '2021-07-20'
   WHEN 104 THEN '2022-11-05'
END;
```



20. Write a query to rename the DeptName column to DepartmentName.

```
>>-- SQL Server / PostgreSQL
ALTER TABLE Department
RENAME COLUMN DeptName TO DepartmentName;
```