**SQL assignment:**

**Aggregate Functions**

Question: Find the average salary of employees in the employees table.

* select avg(salary) from employees;

Question: Count the number of employees in each department.

* select dept\_id, count(\*) from employees group by dept\_id

Question: Retrieve employee names and their corresponding department names.

* select emp\_name, dept\_name from employees

Question: Retrieve the names of employees who have a salary higher than the average

salary.

* select emp\_name from employees where salary > (select avg(salary) from employees);

**Complex JOINs**

Question: Retrieve the names of employees and their managers (use self-join).

* select e.emp\_name, m.manager\_name from employees as e left join Managers as m on e.manager\_id=m.manager\_id;

Question: Retrieve the salaries of employees along with the average salary of their

department.

* Select e.emp\_id, e.salary, d,avg\_sal from employees e join (select dept\_id,avg(salary) as avg\_sal from employees group by dept\_id) on e.dept\_id = d.dept\_id

**CTEs (Common Table Expressions)**

Question: Use a CTE to find the second highest salary in the employees table.



**Recursive Queries**

Question: Using a recursive CTE, retrieve the hierarchy of employees in the organization.

**Advanced Aggregation**

Question: Find the top three departments with the highest total salaries.

-> select dept\_id, sum(salary) as tot\_sal from employees group by dept\_id order bt tot\_sal desc limit 3

**HardComplex JOINs**

Question: Retrieve the names of employees and their managers (use self-join).

* Select e.emp\_name , m.manager\_name from employee e left join manager m on e.manager\_id=m.manager\_id

Question: Retrieve the salaries of employees along with the average salary of their

department.

* Select e.emp\_id, e.salary, d,avg\_sal from employees e join (select dept\_id,avg(salary) as avg\_sal from employees group by dept\_id) on e.dept\_id = d.dept\_id

1. Create an ER diagram for a university database that includes entities such as Students,

Courses, Instructors, and Departments. Define the relationships among these entities.Hint:

Identify primary and foreign keys, and consider relationships like "enrolls," "teaches," and

"belongs to."

create table dept (dept\_id int primary key, dept\_name varchar(50))

create table student (student\_id int primary key,student\_name varchar(50) dept\_id int,foreign key (dept\_id) references dept(dept\_id))

create table course (course\_id int primary key,course\_name varchar(50),dept\_id int,foreign key (dept\_id) references dept(dept\_id))

create table instructor (instructor\_id int primary key,instructor\_name varchar(50), dept\_id int, foreign key (dept\_id) references dept(dept\_id))

2. Convert the ER diagram you created into a relational schema. Clearly define tables,

primary keys, and foreign keys.Hint: Break down the entities into tables and show how they

relate through keys.

->

3. Given a table with the following structure, normalize it to 3NF:

Employee (EmployeeID, EmployeeName, DepartmentID, DepartmentName, ProjectID,

ProjectName, HoursWorked)Hint: Start by removing partial and transitive dependencies.

->

create table employee( empid int primary key, emp\_name varchar(50), dept\_id int,foreign key (dept\_id) references department(dept\_id));

create table dept(dept\_id int primary key,dept\_name varchar(50));

create table proj (proj\_id int primary key,proj\_name varchar(50));

create table emp\_proj(emp\_id int,proj\_id int,hours decimal,primary key (emp\_id, proj\_id), foreign key (emp\_id) references emp(emp\_id),foreign key (proj\_id) references proj(proj\_id));

4. Design a simple transaction in SQL to transfer money between two accounts. Ensure that

your transaction adheres to the ACID properties.Hint: Use BEGIN TRANSACTION,

COMMIT, and ROLLBACK.

->

5. Discuss different types of indexes and their use cases.

->

**MAIN QUESTION:**

Design a small database for a library system, including entities such as Books, Members,

Loans, and Authors. Provide the ER diagram, relational schema, and sample SQL queries to

manage the system.Hint: Include queries for checking out books, returning books, and

querying overdue loans.

->

--creating tables

create table authors(author\_id int primary key,author\_name varchar(50))

create table books(book\_id int primary key,book\_title varchar(50),author\_id int, foreign key (author\_id) references authors(author\_id))

create table members(member\_id int primary key, member\_name varchar(50))

create table loans(loan\_id int primary key,book\_id int,member\_id int,loan\_date date, return\_date date, foreign key (book\_id) references books(book\_id),

foreign key (member\_id) references members(member\_id))