Set 2: Advanced Queries with Joins, CASE, Aggregation

1.

Write a query to list employees, their projects, and hours logged, with projects having more than 100 hours total.

select employee\_id,project\_id,sum(hours\_logged) as total\_logged\_hours from employee\_projects group by employee\_id,project\_id having sum(hours\_logged)>100;

or

select e.employee\_id,first\_name,last\_name,project\_id, hours\_logged from employees e join employee\_projects ep on e.employee\_id=ep.employee\_id group by e.employee\_id,first\_name,last\_name,project\_id,hours\_logged having sum(hours\_logged)>100;

1. Find employees with the most logged hours across all projects.

select employee\_id,sum(hours\_logged) as total\_hours\_logged from employee\_projects group by employee\_id order by total\_hours\_logged desc limit 1;

or

select e.employee\_id,first\_name,last\_name,sum(hours\_logged) as total\_hours\_logged from employees e join employee\_projects ep on e.employee\_id=ep.employee\_id group by e.employee\_id,first\_name,last\_name order by total\_hours\_logged desc limit 1;

1. Display department names and the total salary paid to employees in each department.

--select sum(salary) as total\_salary, department\_name from employees e join departments d on e.department\_id=d.department\_id group by department\_name;

1. Retrieve employees whose salaries are above the average salary across the company.

--select employee\_id,salary

from employees

where salary>(select avg(salary) from employees);

1. Write a query to display projects that started after 2023 and do not have any employees assigned yet.

--select p.project\_id,project\_name,employee\_id from projects p left join employee\_projects ep on p.project\_id=ep.project\_id where start\_date>'2023-01-01' and employee\_id is null;

1. Using a CASE statement , classify transactions as "High Value" if the amount is greater than 5,000; otherwise, classify them as "Low Value."

--select \* ,

case

when amount>5000

then 'high value'

else 'low value'

end as classification

from transactions;

1. Identify the project(s) with the most employees assigned.

--select project\_id,count(distinct employee\_id) as emp\_count from employee\_projects group by project\_id order by emp\_count desc limit 1;

Or

select count(employee\_id) as employee\_count\_below\_50000,department\_name,salary from employees e join departments d on e.department\_id=d.department\_id group by department\_name,salary having salary<50000;

1. Write a query to find the 5 employees with the highest salaries who have worked on the most projects.

--select first\_name,last\_name,count(project\_id) as count\_e,salary from employees e join employee\_projects ep on e.employee\_id=ep.employee\_id group by first\_name,last\_name,salary order by salary desc,count\_e desc limit 5;

1. Retrieve project names and the average hours logged per employee for each project.

--select project\_name,avg(hours\_logged) as avg\_hours from projects p join employee\_projects ep on p.project\_id=ep.project\_id group by project\_name;

1. Write a query to display transaction amounts for employees hired in 2022 or later.

====doubt because i have 3 records were there in employees table but getting only 2 records

--select amount,e.employee\_id,hire\_date from employees e join transactions t on e.employee\_id=t.employee\_id where hire\_date>='2022-01-01';

1. Identify employees who have not logged any hours on projects but have received salary transactions.

--select t.employee\_id,transaction\_id,amount from transactions t left join employee\_projects ep on t.employee\_id=ep.employee\_id where ep.hours\_logged is null;

1. Find departments with no employees using a LEFT JOIN .

// we didn’t get any records because employee\_id can’t be null.because employee\_id is PK

--select department\_name,d.department\_id,employee\_id from departments d left join employees e on d.department\_id=e.department\_id where employee\_id is null;

1. Retrieve the employees who have worked on more than one project and logged over 200 hours in total.

--select employee\_id from employee\_projects group by employee\_id having count(project\_id)>1 and sum(hours\_logged)>200;

Or

--select ep.employee\_id,concat(first\_name,' ',last\_name) as full\_name from employee\_projects ep join employees e on ep.employee\_id=e.employee\_id group by employee\_id having count(project\_id)>1 and sum(hours\_logged)>200;

1. Write a query to fetch projects where no hours have been logged by employees.

--select p.project\_id,project\_name from projects p left join employee\_projects ep on p.project\_id=ep.project\_id where employee\_id is null;

1. Display employees who have received both a bonus and a deduction in their transactions.

--select employee\_id from transactions

where transaction\_type in('bonus' ,'deduction')

group by employee\_id

having count(transaction\_id)=2;

Or

--select t.employee\_id,concat(first\_name,' ',last\_name) as full\_name from transactions t join employees e on t.employee\_id=e.employee\_id where transaction\_type in('bonus' ,'deduction') group by employee\_id having count(transaction\_id)=2;

1. Identify employees with missing or null last\_name values and classify them as "Data Issue."

--select employee\_id , case when last\_name='' or last\_name is null then 'data issue' else 'no issue' end as classification from employees;

1. Using a self join , display pairs of employees in the same department with different salaries.
2. Write a query to calculate the total hours logged for each employee and classify them into "High Performer" (> 100 hours) or "Normal Performer."

--select employee\_id,hours\_logged, case when sum(hours\_logged)>100 then 'high performer' else 'normal performer' end as classification from employee\_projects group by employee\_id,hours\_logged;

1. Find the latest transaction date for each employee who received a salary payment.

--select employee\_id, max(transaction\_date) as transaction\_date from transactions where transaction\_type='salary' group by employee\_id;