create database Company;

use Company;

create table Employees (

EmployeeID int primary key, FirstName varchar(50), LastName varchar(50),

DepartmentID int, Salary decimal(10,2), JoiningDate date, Position varchar(20));

create table departments (

DepartmentID int primary key, DepartmentName varchar(50),ManagerID int,

foreign key (ManagerID) REFERENCES Employees(EmployeeID));

create table Projects (

ProjectID int primary key,

projectName Varchar(100),

StartDate Date,

EndDate Date,

Budget decimal(15,2));

CREATE TABLE EmployeeProjects (

EmployeeID INT,

ProjectID int,

HoursAllocated int,

primary key (EmployeeID, ProjectID),

foreign key (EmployeeID) references Employees(EmployeeID),

foreign key (ProjectID) references Projects(ProjectID));

INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID, Salary, JoiningDate, Position) VALUES

(1, 'John', 'Doe', 101, 75000.00, '2020-01-15', 'Manager'),

(2, 'Jane', 'Smith', 102, 60000.00, '2019-06-10', 'Analyst'),

(3, 'Alice', 'Johnson', 103, 55000.00, '2021-03-01', 'Developer'),

(4, 'Bob', 'Brown', 101, 80000.00, '2018-11-20', 'Manager'),

(5, 'Carol', 'Davis', 104, 72000.00, '2022-04-12', 'Consultant');

INSERT INTO Departments (DepartmentID, DepartmentName, ManagerID) VALUES

(101, 'Human Resources', 1),

(102, 'Finance', 2),

(103, 'IT', 3),

(104, 'Marketing', 5);

INSERT INTO Projects (ProjectID, ProjectName, StartDate, EndDate, Budget) VALUES

(201, 'Project Alpha', '2023-01-01', '2023-12-31', 500000.00),

(202, 'Project Beta', '2022-03-15', '2023-03-14', 300000.00),

(203, 'Project Gamma', '2024-01-10', '2024-10-20', 400000.00);

INSERT INTO EmployeeProjects (EmployeeID, ProjectID, HoursAllocated) VALUES

(1, 201, 1500),

(2, 202, 1200),

(3, 203, 1300),

(4, 201, 1400),

(5, 202, 1100);

select \* from Employees;

select \* from EmployeeProjects;

select \* from Departments;

select \* from Projects;

#Write a query to retrieve all columns from the Employees table

select \* from Employees;

#Retrieve only the FirstName and LastName of all employees in the Employees table.

select FirstName, LastName from Employees;

#Find all employees with a salary greater than 60000.

select FirstName, LastName, Salary from Employees

where Salary > 60000;

#Write a query to find employees who joined the company after January 1, 2020

select \* from Employees

where JoiningDate > '2020-01-01';

#Use ORDER BY to sort employees by Salary in descending order.

select \* from Employees

order by Salary Desc;

#Display distinct departments from the Departments table.

SELECT DISTINCT DepartmentName

FROM Departments;

#Find the count of employees in the Employees table.

select count(EmployeeID) from Employees;

#Use LIKE to find employees whose LastName starts with 'S'.

SELECT \*

FROM employees

WHERE LastName LIKE 'S%';

#Retrieve employees who don’t belong to any department.

SELECT \*

FROM employees

WHERE DepartmentID IS NULL;

### There are no employee in the dataset who don't belong to any department.

#Calculate the average salary of all employees.

select Avg(Salary) as Avg\_Sal from Employees;

#Retrieve all departments where the number of employees is more than 1.

SELECT DepartmentName, COUNT(EmployeeID) AS employee\_count

FROM Departments

JOIN Employees ON Departments.DepartmentID = Employees.DepartmentID

GROUP BY DepartmentName

HAVING COUNT(EmployeeID) > 1;

#Write a query to retrieve employees who earn between 40000 and 70000.

SELECT \*

FROM Employees

WHERE Salary BETWEEN 40000 AND 70000;

#Show the total number of employees working in each department

SELECT DepartmentName, COUNT(EmployeeID) AS total\_employees

FROM Departments

JOIN Employees ON Departments.DepartmentID = Employees.DepartmentID

GROUP BY DepartmentName;

#Use JOIN to display employees along with their department names.

select FirstName, DepartmentName from

Employees join Departments on Departments.DepartmentID = Employees.DepartmentID;

#Write a query to find employees who work in the 'IT' department.

select FirstName, DepartmentName from

Employees join Departments on Departments.DepartmentID = Employees.DepartmentID

where DepartmentName = "IT";

#Find all projects where the budget is less than 400000.

select projectName, Budget from Projects

where Budget < 400000;

#Display employees and the number of projects they are working on.

SELECT Employees.EmployeeID, Employees.FirstName, COUNT(EmployeeProjects.ProjectID) AS ProjectCount

FROM Employees

LEFT JOIN EmployeeProjects ON Employees.EmployeeID = EmployeeProjects.EmployeeID

GROUP BY Employees.EmployeeID, Employees.FirstName;

#Retrieve a list of employees who work on multiple projects.

SELECT Employees.EmployeeID, Employees.FirstName, COUNT(EmployeeProjects.ProjectID) AS ProjectCount

FROM Employees

LEFT JOIN EmployeeProjects ON Employees.EmployeeID = EmployeeProjects.EmployeeID

GROUP BY Employees.EmployeeID, Employees.FirstName

having ProjectCount>1;

#Show departments with no assigned manager.

SELECT DepartmentID, DepartmentName

FROM departments

WHERE ManagerID IS NULL;

#Write a query to list employees, their departments, and their projects.

SELECT Employees.EmployeeID, Employees.FirstName, Employees.LastName, DepartmentName, projectName

FROM Employees

JOIN departments ON Employees.DepartmentID = Departments.DepartmentID

LEFT JOIN EmployeeProjects ON Employees.EmployeeID = EmployeeProjects.EmployeeID

LEFT JOIN Projects ON EmployeeProjects.ProjectID = Projects.ProjectID;

#Find employees with salaries higher than the average salary.

select FirstName, Salary

FROM Employees

having Salary > (SELECT AVG(Salary) FROM Employees);

### OR ###

SELECT Employees.FirstName, Employees.Salary, avgS.avgSalary AS AverageSalary

FROM Employees

CROSS JOIN (SELECT AVG(Salary) AS avgSalary FROM Employees) avgS

WHERE Employees.Salary > avgS.avgSalary;

#Write a query to calculate the total salary by department.

SELECT d.DepartmentName, SUM(e.Salary) AS TotalSalary

FROM Employees e

JOIN departments d ON e.DepartmentID = d.DepartmentID

GROUP BY d.DepartmentName;

#Display employees and their joining dates, showing the most recent hires first.

select FirstName ,JoiningDate from Employees

order by JoiningDate DESC;

#Write a query to find the maximum, minimum, and average budget of projects.

SELECT projectName, Budget,

MAX(Budget) OVER () AS MaxBudget,

MIN(Budget) OVER () AS MinBudget,

AVG(Budget) OVER () AS AvgBudget

FROM Projects;

#Create a view to show employees and their department names.

create view View1 as

select e.EmployeeID , e.FirstName , d.DepartmentName

from Employees e join Departments d on e.DepartmentID = d.DepartmentID;

select \* from View1;

#Retrieve the top 2 highest-paid employees.

select EmployeeID, FirstName, Salary

from Employees

Order by Salary DESC

limit 2 ;

#Display the number of projects per department.

SELECT d.DepartmentName, COUNT(p.ProjectID) AS NumberOfProjects

FROM Departments d

LEFT JOIN Employees e ON d.DepartmentID = e.DepartmentID

LEFT JOIN EmployeeProjects ep ON e.EmployeeID = ep.EmployeeID

LEFT JOIN Projects p ON ep.ProjectID = p.ProjectID

GROUP BY d.DepartmentName;

#Write a query to get the employee who spends the most hours on a project.

SELECT ep.EmployeeID, e.FirstName, e.LastName, SUM(ep.HoursAllocated) AS TotalHours

FROM EmployeeProjects ep

JOIN Employees e ON ep.EmployeeID = e.EmployeeID

GROUP BY ep.EmployeeID, e.FirstName, e.LastName

ORDER BY TotalHours DESC

LIMIT 1;

#Identify departments where the average employee salary is above 60000.

SELECT e.EmployeeID, e.FirstName, e.Salary, d.DepartmentName

FROM Employees e

JOIN Departments d ON e.DepartmentID = d.DepartmentID

JOIN (

SELECT DepartmentID

FROM Employees

GROUP BY DepartmentID

HAVING AVG(Salary) > 60000

) AS high\_avg\_depts ON d.DepartmentID = high\_avg\_depts.DepartmentID;

### OR ###

SELECT e.FirstName, d.DepartmentName, e.Salary , e.EmployeeID

FROM Employees e

JOIN Departments d ON e.DepartmentID = d.DepartmentID

WHERE e.DepartmentID IN (

SELECT DepartmentID

FROM Employees

GROUP BY DepartmentID

HAVING AVG(Salary) > 60000

);

#select employees who work on all projects

SELECT e.EmployeeID, e.FirstName, e.LastName

FROM Employees e

JOIN EmployeeProjects ep ON e.EmployeeID = ep.EmployeeID

GROUP BY e.EmployeeID, e.FirstName, e.LastName

HAVING COUNT(DISTINCT ep.ProjectID) = (SELECT COUNT(\*) FROM Projects);

#Find departments that have the highest budget allocation for their projects.

SELECT Departments.DepartmentName, Employees.EmployeeID, Projects.ProjectID, Projects.Budget

FROM Departments

JOIN Employees ON Departments.DepartmentID = Employees.DepartmentID

JOIN EmployeeProjects ON Employees.EmployeeID = EmployeeProjects.EmployeeID

JOIN Projects ON EmployeeProjects.ProjectID = Projects.ProjectID

order by Budget Desc

limit 1;

#Write a query to retrieve employees who work on the highest number of projects.

WITH ProjectCounts AS (

SELECT EmployeeID, COUNT(ProjectID) AS ProjectCount

FROM EmployeeProjects

GROUP BY EmployeeID

)

SELECT e.EmployeeID, e.FirstName, e.LastName, pc.ProjectCount

FROM Employees e

JOIN ProjectCounts pc ON e.EmployeeID = pc.EmployeeID

WHERE pc.ProjectCount = (SELECT MAX(ProjectCount) FROM ProjectCounts);

#Find projects where the total allocated hours exceed 2500.

select \* from EmployeeProjects;

select Projects.ProjectName , sum(HoursAllocated) as Total\_Hrs from Projects join EmployeeProjects on

Projects.ProjectID = EmployeeProjects.ProjectID

GROUP BY Projects.ProjectID, Projects.ProjectName

HAVING SUM(EmployeeProjects.HoursAllocated) > 2500;

#Calculate the percentage of employees in each department.

SELECT d.DepartmentName, COUNT(e.EmployeeID) AS EmployeeCount,

(COUNT(e.EmployeeID) \* 100.0 / (SELECT COUNT(\*) FROM Employees)) AS PercentageOfEmployees

FROM Departments d

JOIN Employees e ON d.DepartmentID = e.DepartmentID

GROUP BY d.DepartmentID, d.DepartmentName;

#Retrieve employees who have been in the company for more than 5 years.

SELECT EmployeeID, FirstName, LastName, JoiningDate

FROM Employees

WHERE DATEDIFF(CURDATE(), JoiningDate) > 365 \* 5;

#Write a query to find overlapping projects (where start and end dates overlap).

SELECT p1.ProjectID AS Project1\_ID, p1.ProjectName AS Project1\_Name,

p2.ProjectID AS Project2\_ID, p2.ProjectName AS Project2\_Name

FROM Projects p1

JOIN Projects p2 ON p1.ProjectID != p2.ProjectID

WHERE p1.StartDate < p2.EndDate

AND p1.EndDate > p2.StartDate;

#Write a query to list the department with the most employees.

select d.DepartmentName , Count(e.EmployeeID) as EmployeeCount from

Employees e join Departments d on e.DepartmentID = d.DepartmentID

group by d.DepartmentName

ORDER BY EmployeeCount DESC

Limit 1 ;

#Use a subquery to find employees whose salaries are above the average salary for their department.

select EmployeeID, FirstName, LastName , Salary from Employees

where Salary >(select Avg(Salary) from Employees);

#Find all departments with budgets that exceed the average department budget.

select Departments.DepartmentName , sum(Projects.Budget) as Total\_Department\_Budget from Departments

join Employees on Departments.DepartmentID = Employees.DepartmentID

join EmployeeProjects on EmployeeProjects.EmployeeID = Employees.EmployeeID

Join Projects on EmployeeProjects.ProjectID = Projects.ProjectID

GROUP BY Departments.DepartmentID, Departments.DepartmentName

HAVING SUM(Projects.Budget) > (SELECT AVG(Budget) FROM Projects);

#Create a stored procedure to calculate the total salary expense by department.

DELIMITER $$

CREATE PROCEDURE CalculateSalaryExpenseByDepartment()

BEGIN

select Departments.DepartmentName, Sum(Employees.Salary) from Departments

join Employees on Departments.DepartmentID = Employees.DepartmentID

group by Departments.DepartmentID;

end $$

DELIMITER ;

CALL CalculateSalaryExpenseByDepartment();

#Write a trigger to prevent any updates to the EmployeeProjects table if the HoursAllocated exceeds 40.

DELIMITER $$

CREATE TRIGGER PreventHoursAllocationExceeding40

BEFORE UPDATE ON EmployeeProjects

FOR EACH ROW

BEGIN

IF NEW.HoursAllocated > 40 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot update HoursAllocated. The value cannot exceed 40.';

END IF;

END $$

DELIMITER ;

UPDATE EmployeeProjects

SET HoursAllocated = 40

WHERE EmployeeID = 1 AND ProjectID = 101;

SELECT \* FROM EmployeeProjects;

#Write a query to retrieve the average salary, excluding the top 10% of earners.

WITH RankedEmployees AS (

SELECT Salary, ROW\_NUMBER() OVER (ORDER BY Salary DESC) AS RowNum, COUNT(\*) OVER () AS TotalCount

FROM Employees

)

SELECT AVG(Salary) AS AvgSalaryExcludingTop10

FROM RankedEmployees

WHERE RowNum > TotalCount \* 0.1;

#Write a query to retrieve departments with more than 1 employees, sorted by the average salary of the department.

SELECT d.DepartmentName, COUNT(e.EmployeeID) AS EmployeeCount, AVG(e.Salary) AS AvgSalary

FROM Departments d

JOIN Employees e ON d.DepartmentID = e.DepartmentID

GROUP BY d.DepartmentName

HAVING COUNT(e.EmployeeID) > 1

ORDER BY AvgSalary DESC;

DELIMITER $$

#Create a function that returns the tenure (in years) of an employee based on their joining date.

CREATE FUNCTION CalculateTenure(JoiningDate DATE)

RETURNS DECIMAL(5,2)

DETERMINISTIC

BEGIN

DECLARE tenure DECIMAL(5,2);

-- Calculate the difference in years between the current date and the joining date

SET tenure = TIMESTAMPDIFF(YEAR, JoiningDate, CURDATE());

-- Adjust the tenure to account for any remaining months (not a full year)

IF MONTH(CURDATE()) < MONTH(JoiningDate) OR (MONTH(CURDATE()) = MONTH(JoiningDate) AND DAY(CURDATE()) < DAY(JoiningDate)) THEN

SET tenure = tenure - 1;

END IF;

RETURN tenure;

END $$

DELIMITER ;

SELECT EmployeeID, FirstName, LastName, CalculateTenure(JoiningDate) AS TenureYears

FROM Employees;

#Write a query to calculate the difference between the maximum and minimum salaries in each department.

SELECT d.DepartmentName,

MAX(e.Salary) - MIN(e.Salary) AS SalaryDifference

FROM Departments d

JOIN Employees e ON d.DepartmentID = e.DepartmentID

GROUP BY d.DepartmentName;

#Write a CTE (Common Table Expression) to list employees with their cumulative hours on projects.

WITH CumulativeHours AS (

SELECT e.EmployeeID,

e.FirstName,

e.LastName,

ep.ProjectID,

ep.HoursAllocated,

SUM(ep.HoursAllocated) OVER (PARTITION BY e.EmployeeID ORDER BY ep.ProjectID) AS CumulativeHours

FROM Employees e

JOIN EmployeeProjects ep ON e.EmployeeID = ep.EmployeeID

)

SELECT EmployeeID, FirstName, LastName, ProjectID, CumulativeHours

FROM CumulativeHours;