**Division (DID, dname, managerID)**

**Employee (empID, name, salary, *DID*)**

**Project (PID, pname, budget, *DID*)**

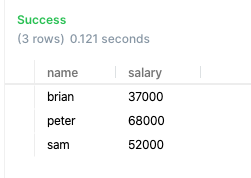
**Workon *(PID, EmpID*, hours)**

1. **List the name and salary of employees who work for division 3.**

SELECT name,salary

FROM employee

WHERE did=3;

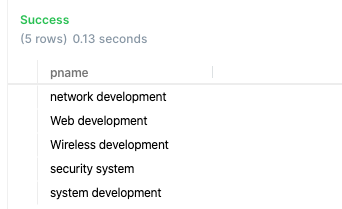


1. **List the name of project whose budget is between 5000-7000**

SELECT pname

FROM project

WHERE budget BETWEEN 5000 AND 7000;

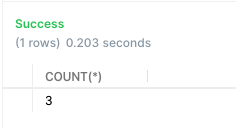


1. **List the total number of employees whose initial of name is 's'. (hint, using LIKE operator and wildcard character).**

SELECT COUNT(\*)

FROM employee

WHERE name LIKE 's%';



1. **List the total number of employees whose initial of name is NOT 's’ for each division, including division ID.**

SELECT did,count(\*)

FROM employee

WHERE name NOT LIKE 's%'

GROUP BY did;

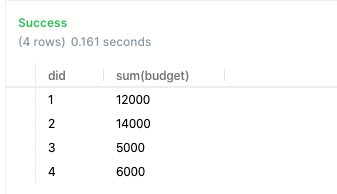


1. **List the total project budget for each division, including division ID.**

SELECT did, sum(budget)

FROM project

GROUP BY did;



1. **List the ID of the division that has two or more projects with budget over $6000.**

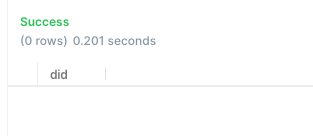
select did

FROM project

where budget>6000

GROUP BY did

HAVING count(\*)>=2;

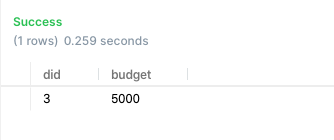


1. **List the ID of division that sponsors project "Web development", List the project budget too.**

SELECT did, budget

FROM project

WHERE pname = 'Web development';



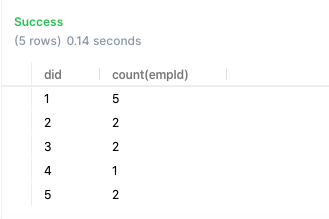
1. **List the total number of employees whose salary is above $40000 for each division, list division ID.**

SELECT did, count(empId)

FROM employee

WHERE salary > 40000

GROUP BY did;

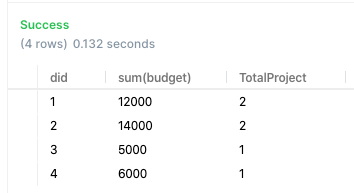


1. **List the total number of project and total budget for each division, show division ID.**

SELECT did, sum(budget), count (\*) AS TotalProject

FROM project

GROUP BY did;



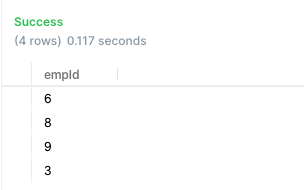
1. **List the ID of employee that worked on more than three projects.**

SELECT empId

FROM workon

GROUP BY empId

HAVING count(pid)>3;

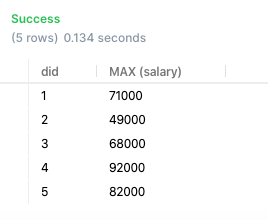


1. **List the ID of each division with its highest salary.**

SELECT did, MAX (salary)

FROM employee

GROUP BY did;



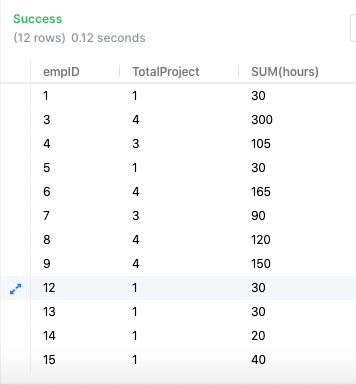
1. **List the total number of projects each employee works on, including employee's ID and total hours an employee spent on project.**

SELECT empID, count(pid) AS TotalProject, SUM(hours)

FROM workon

GROUP BY empID

ORDER BY empID;

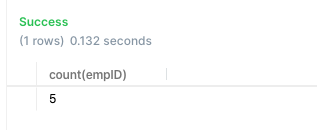


1. **List the total number of employees who work on project 1.**

SELECT count(empID)

FROM workon

WHERE pid = 1;



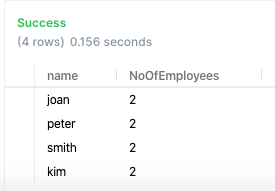
1. **List names that are shared by more than one employee and list the number of employees who share that name.**

SELECT name, count(empID) AS NoOfEmployees

FROM employee

GROUP BY name

HAVING count(empID)>1;



1. **Bonus question (1 point) List the total number of employee and total salary for each division, including division name (hint: use JOIN operation, read the text for join operation)**

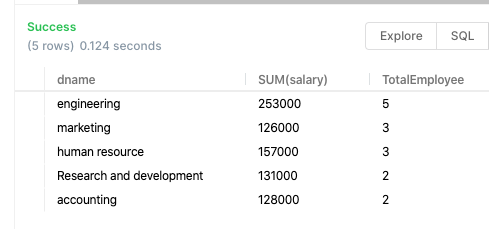
SELECT division.dname, SUM(salary), count(empID) AS TotalEmployee

FROM employee

JOIN division

ON employee.did=division.did

GROUP BY employee.did;



1. **List the name of division that sponsors project "Web development", List the project budget too.**

SELECT d.dname,p.budget

FROM project p, division d

WHERE p.did = d.did AND lower(p.pname) = 'web development';

OR

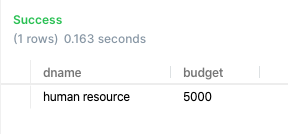
SELECT d.dname, p.budget

FROM project p

JOIN division d

ON p.did = d.did

WHERE lower(pname) = 'web development';



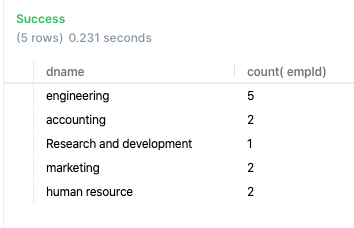
1. **List the total number of employees whose salary is above $40000 for each division, show division name.**

SELECT d.dname, COUNT(empId)

FROM employee e, division d

WHERE e.did = d.did AND e.salary >40000

GROUP BY d.dname;



1. **List the total number of project and total budget for each division, show division name.**

SELECT d.dname, COUNT(pid) AS NoOfProjects, SUM(budget) AS TotalBudget

FROM project p, division d

WHERE p.did = d.did

GROUP BY d.dname;

OR

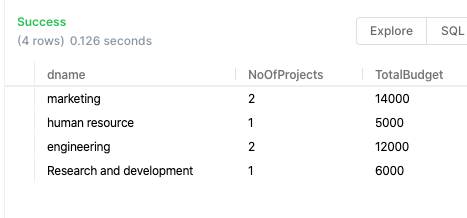
SELECT d.dname, COUNT(pid) NoOfProjects, SUM(budget) TotalBudget

FROM project p

JOIN division d

ON p.did = d.did

GROUP BY d.dname;



1. **For each project, list its name and total number of employees who work on that project.**

SELECT p.pname, COUNT(empid) AS TotalEmployees

FROM workon w, project p

WHERE w.pid = p.pid

GROUP BY p.pname;



1. **List the name and ID of employees that worked on more than one projects. (note: there are some employees who have same names).**

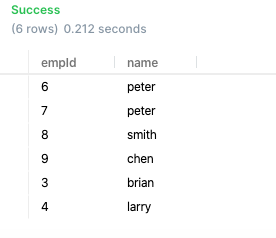
SELECT w.empId, e.name

FROM workon w, employee e

WHERE w.empId = e.empId

GROUP BY w.empID, e.name

HAVING COUNT(\*)>1;



1. **List the name of division that has more than 2 projects whose budget is over 3000.**

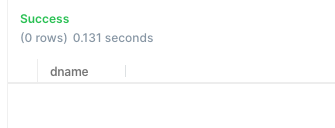
SELECT d.dname

FROM project p, division d

WHERE p.did = d.did AND p.budget>3000

GROUP BY d.dname

HAVING count(budget)>2;



1. **List the total number of project each employee works on, including employee's name (note:  there are some employees who have same names ).**

SELECT e.empid, e.name, count(pid) AS NoOfProjects

FROM workon w, employee e

WHERE w.empid=e.empid

GROUP BY e.empid, e.name

ORDER BY e.empid, e.name;



1. **List the total number of employees who work on project "Web development".  Also list the total working hours for this project.**

SELECT count(empId), sum(hours)

FROM workon w, project p

WHERE w.pid = p.pid AND lower(p.pname) = 'web development';

OR

SELECT count(empId), sum(hours)

FROM workon

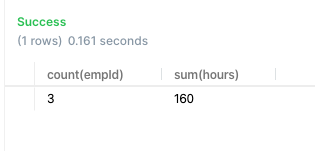
WHERE pid IN (

SELECT pid

FROM project

WHERE lower(pname) = 'web development'

);



1. **List the name of employee and the total number of projects the employee works on, as well as the total hours he/she spent on the project(s) (Hint: use left outer join so that employees who don't work on project will be also listed with zero project count).**

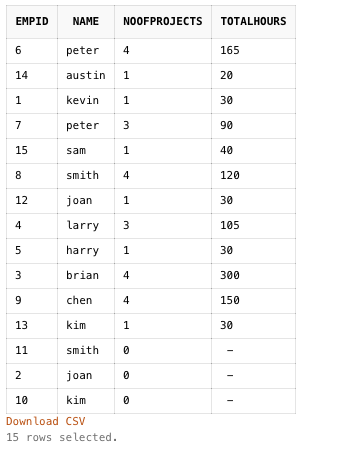
SELECT e.empid, e.name, count(pid) AS NoOfProjects, sum(hours) AS TotalHours

FROM employee e

LEFT JOIN workon w

ON e.empid = w.empid

GROUP BY e.empid, e.name;

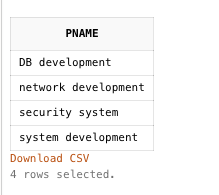


1. **List the name of project that 'chen' works on. (Hint: join three tables)**

SELECT p.pname

FROM project p ,employee e, workon w

WHERE e.empid = w.empid AND w.pid = p.pid AND lower(e.name) = 'chen';



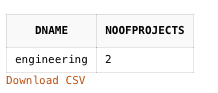
1. **List the total number of projects whose budget is above 1000 in 'engineering ' division (show division name).**

SELECT d.dname, count(pid) AS NoOfProjects

FROM project p, division d

WHERE d.did = p.did AND lower(dname) = 'engineering' AND p.budget>1000

GROUP BY d.dname;



1. **List the total number of project and total budget for each division, including division name (use left outer join because some division may not have any project).**

SELECT d.dname, sum(budget) AS TotalBudget, count(budget) AS NoOfProjects

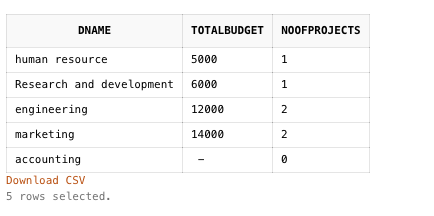
FROM division d

LEFT JOIN project p

ON d.did = p.did

GROUP BY d.dname

ORDER BY TotalBudget;

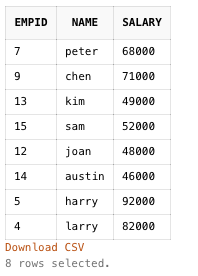


1. **List the name of employee and his/her salary, who work on any project and salary is over $45000. (note:  don't duplicate an employee in the list)**

SELECT DISTINCT e.empid, e.name, e.salary

FROM employee e, workon w

WHERE e.empid = w.empid AND e.salary > 45000;



1. **List the name of the employee who work on one or more projects with budget over $5000. (note:  don't duplicate an employee in the list)**

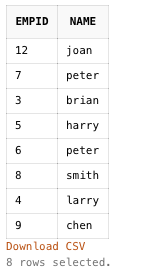
SELECT e.empid, e.name

FROM employee e, workon w, project p

WHERE e.empid = w.empid AND w.pid = p.pid AND p.budget>5000

GROUP BY e.empId, e.name

HAVING count(w.pid)>=1;

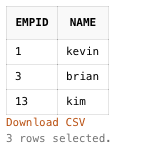


1. **List the name of employees who work on project "web development".**

select e.empid, e.name

FROM employee e, workon w, project p

WHERE e.empid = w.empid AND w.pid = p.pid AND lower(p.pname) = 'web development';



1. **List the name of employee whose salary is over company's average salary.**

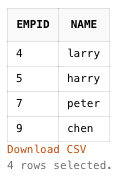
SELECT empid, name

FROM employee

WHERE salary > (

select avg(salary)

FROM employee);



1. **Create a view called "high\_load\_employee" that shows the ID and the name of employees whose total project hours is over 100. Also include employees' total hours, and total projects. Show the code of view create statement and use select statement to show the contents of the view.**

CREATE OR REPLACE VIEW high\_load\_employee

AS

SELECT e.empid, e.name, sum(hours) TotalHoursWorked,count(pid) NoOfProjects

FROM employee e, workon w

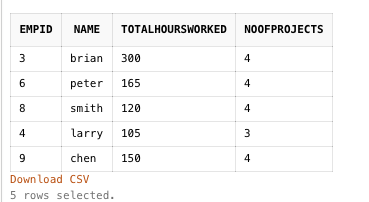
WHERE e.empid = w.empid

GROUP BY e.empid, e.name

HAVING sum(hours)>100;

SELECT \*

FROM high\_load\_employee;



1. **List the name of employee and the total number of projects the employee works on, as well as the total hours he/she spent on the project(s) (Hint: use Left Outer Join so that employees who don't work on project will be also listed with zero project count).**

SELECT e.empid, e.name, count(pid) TotalProjects, sum(hours) TotalHours

FROM employee e

LEFT JOIN workon w

ON e.empid = w.empid

GROUP BY e.empid, e.name;

OR

SELECT e.empid, e.name, count(pid) TotalProjects, sum(hours) TotalHours

FROM employee e, workon w

WHERE e.empid = w.empid (+)

GROUP BY e.empid, e.name;



1. **List the name of project that ‘Chen’ works on but 'Mary' does not work on.**

SELECT pname

FROM project

WHERE pid IN (

SELECT pid

FROM workon w, employee e

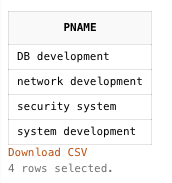
WHERE e.empid = w.empid AND lower(e.name) = 'chen')

AND pid NOT IN (

SELECT pid

FROM workon w, employee e

WHERE w.empid = e.empid AND lower(e.name) = 'mary');



1. **List the total number of projects whose budget is above the average project budget (hint: use subquery)**

SELECT count(pid) TotalProjects

FROM project

WHERE budget > (

SELECT avg(budget)

FROM project

);



1. **List the total number of projects 'chen' does not works on (must use subquery for this query, use "not in" in where clause).**

SELECT count(DISTINCT pid) TotalProjectsWithoutChen

FROM workon

WHERE pid NOT IN (

SELECT w.pid

FROM workon w, employee e

WHERE e.empid = w.empid AND lower(e.name) = 'chen'

);



1. **List the total number of projects that employee from 'accounting ' division work on, also list the total hours those employees spent on projects.**

SELECT count(DISTINCT pid) TotalProjects, sum(hours) TotalHours

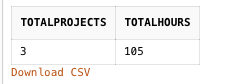
FROM workon

WHERE empid IN (

SELECT e.empid

FROM employee e, division d

WHERE e.did = d.did AND lower(d.dname) = 'accounting');



1. **Among all projects, list the name of the project that has more people working in (hint use a subquery at HAVING clause).**

SELECT pname

FROM project p, workon w

WHERE p.pid = w.pid

GROUP BY pname

HAVING count(w.empid) = (

SELECT max(count(empid))

FROM workon w

GROUP BY pid);



1. **List the total number of employees whose salary is over company's average salary (subquery) for each division.**

SELECT d.did, count(e.empid) TotalEmployees

FROM division d

LEFT JOIN employee e

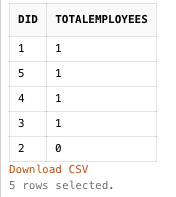
ON e.did = d.did

AND e.salary > (

SELECT avg(salary)

FROM employee)

GROUP BY d.did;



1. **List the name of manager (note a manager is an employee whose empid is in division table as managerID) who work on project (use subquery).**

SELECT d.managerid, e.name

FROM division d, employee e

WHERE d.managerid = e.empid AND d.managerid IN (

SELECT DISTINCT empid

FROM workon

);

**OR**

SELECT e.empid, e.name

FROM employee e

WHERE empid IN (

SELECT managerid

FROM division

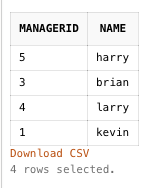
WHERE managerid IN (

SELECT DISTINCT empid

FROM workon

)

);



1. **List the name of employees who don't work on project "web development"  (You MUST use subquery strategy, not to use JOIN because JOIN operation does not work for this query, think about why).**

SELECT empid, name

FROM employee

WHERE empid NOT IN (

SELECT w.empid

FROM workon w, project p

WHERE w.pid = p.pid AND lower(p.pname) = 'web development'

);



1. **Create a view named as 'to\_be\_list' that has the name of employee and his/her salary, who don't work on any project and salary is over $55000. Also show employee's division name. Run select statement to show the contents of this view.**

CREATE OR REPLACE VIEW to\_be\_list

AS

SELECT e.empid, e.name, e.salary, d.dname

FROM employee e, division d

WHERE e.did = d.did AND e.salary > 55000 AND e.empid NOT IN (

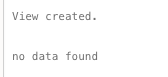
SELECT DISTINCT w.empid

FROM workon w

);

SELECT \*

FROM to\_be\_list;



1. **List the name of employee whose salary is higher than ‘Larry ‘.**

SELECT empid, name

FROM employee

WHERE salary > (

SELECT salary

FROM employee

WHERE lower(name) = 'larry'

);



1. **List name of project whose budget is SECOND lowest, list budget too. (hint, refer to the video lecture).**

SELECT pname, budget

FROM project

WHERE budget = (

SELECT min(budget)

FROM project

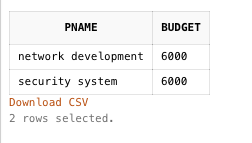
WHERE budget > (

SELECT min(budget)

FROM project

)

);



1. **List the name of project that 'chen' works on but not from chen's division.**

SELECT p.pname

FROM employee e, project p, workon w

WHERE e.empid = w.empid AND p.pid= w.pid AND lower(e.name) = 'chen' AND e.did <> p.did;

**OR**

SELECT p.pname

FROM employee e, project p, workon w

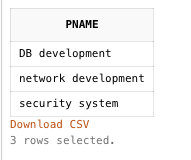
WHERE e.empid = w.empid AND p.pid= w.pid AND lower(e.name) = 'chen' AND p.did NOT IN (

SELECT did

FROM employee

where lower(name) = 'chen'

);



1. **List the name of employee who make highest salary in his/her division.**

SELECT e.did, e.empid, e.name

FROM employee e

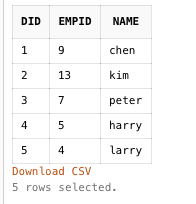
WHERE salary IN (

SELECT max(salary)

FROM employee e1

WHERE e1.did = e.did)

ORDER BY e.did;



1. **List the name of project sponsored by Chen's division. (hint/think: find a project whose DID equals to the DID of an employee whose name is Chen)**

SELECT pname

FROM project

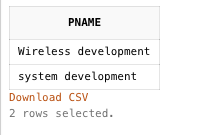
WHERE did = (

SELECT did

FROM employee

WHERE lower(name)='chen'

);



1. **List the name of employee who is working on the project whose budget is below the**divisional**average project budget (use correlated subquery).**

SELECT DISTINCT e.empid, e.name

FROM employee e, workon w

WHERE e.empid = w.empid and w.pid IN (

SELECT pid

FROM project p WHERE budget < (

SELECT avg(budget)

FROM project pp

WHERE pp.did = p.did)

)

ORDER BY e.empid

;



1. **List the name of project that some employee(s) who is/are working on it make less than divisional average salary (use correlated subquery).**

SELECT DISTINCT pname

FROM project p, workon w, employee e

WHERE p.pid = w.pid AND e.empid = w.empid AND salary < (

SELECT avg(salary)

FROM employee e1

WHERE e1.did = e.did

);



1. **List the total number of divisions that has 2 or more employees working on projects. For this query I built a framework of the code, you just need to fill in the right code in \_\_\_\_, and then run the code to get the result.**

select count(did)

from division

where did in (

select did

from workon w, project p

where w.pid = p.pid

group by did

having count(empid)>2

);



1. **List the total number of projects 'accounting' division manager works on. (Note, if an employee is a division's manager, his/her empID is IN the Division table)**

SELECT count(pid) TotalProjects

FROM workon w

WHERE empid = (

SELECT managerid

FROM division

WHERE lower(dname) = 'accounting'

);



1. **List the name of the employees (and his/her DID) who work on more projects than his/her**divisional**colleagues. (hint: co-related subquery, also use having, compare count() to count, use “ … having count (pid) >=ALL (select count (pid) …..)**

SELECT e.empid, e.name, did

FROM employee e, workon w

WHERE e.empid = w.empid

GROUP BY e.empid,e.name,did

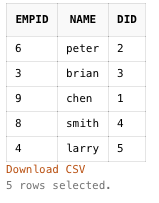
HAVING count(pid)>= ALL (

SELECT count(pid)

FROM workon ww, employee ee

WHERE ww.empid = ee.empid AND e.did = ee.did

GROUP BY ee.empid);



1. **List the name of the division that has more than one employee whose salary is greater than company's average salary (subquery, group by, having)**

SELECT dname

FROM division d, employee e

WHERE d.did = e.did AND salary> (

SELECT avg(salary)

FROM employee)

GROUP BY dname

HAVING count(empid) > 1;



1. **List the name of the division that has more than one employee whose salary is greater than the divisional average salary (corelated subquery, group by, having)**

SELECT dname

FROM division d, employee e

WHERE d.did = e.did AND salary > (

SELECT avg(salary)

FROM employee ee

WHERE e.did = ee.did)

GROUP BY dname

HAVING count(empid) >1;



1. **List the name of the employee that has the lowest salary in his division and list the total number of projects this employee is work on (use corelated subquery).**

SELECT e.empid, e.name, count(pid) TotalProjects

FROM employee e, workon w

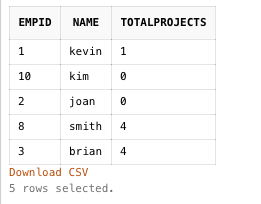
WHERE e.empid = w.empid (+) AND e.salary = (

SELECT min(salary)

FROM employee e1

WHERE e.did = e1.did)

GROUP BY e.empid, e.name;



1. **List the name of project Larry does not work on.**

SELECT DISTINCT pname

FROM project p, workon w

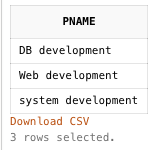
WHERE p.pid = w.pid AND p.pid NOT IN (

SELECT pid

FROM workon ww, employee e

WHERE ww.empid = e.empid AND lower(name) = 'larry'

);



1. **List the name of employee in Chen's division who works on a project that Chen does NOT work on.**

SELECT e.empid, e.name

FROM employee e, workon w

WHERE e.empid = w.empid AND did = (

SELECT did

FROM employee

WHERE lower(name) = 'chen'

)

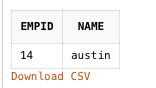
AND pid NOT IN (

SELECT ww.pid

FROM workon ww, employee ee

WHERE ww.empid = ee.empid AND lower(ee.name)='chen'

);



1. **List the name of divisions that sponsors project(s) Chen works on. (Namely, if there is a project 'chen' works on, find the name of the division that sponsors that project.)**

SELECT DISTINCT dname

FROM division d, project p, workon w, employee e

WHERE d.did = p.did AND p.pid = w.pid AND e.empid = w.empid AND lower(e.name) = 'chen';

OR

SELECT DISTINCT dname

FROM division d, project p

WHERE d.did = p.did AND pid IN (

SELECT pid

FROM workon w, employee e

WHERE w.empid = e.empid AND lower(e.name) = 'chen'

);



1. **List the name of division (d) that has employee who work on a project (p) not sponsored by this division. (hint in a corelated subquery where d.did <> p.did)**

SELECT DISTINCT dname

FROM division d, project p

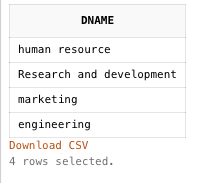
WHERE d.did = p.did AND pid IN (

SELECT w.pid

FROM project pp, workon w

WHERE pp.did <> p.did

);



1. **List the name of employee who work with Chen on some project(s).**

SELECT DISTINCT e.empid, e.name

FROM employee e, workon w

WHERE e.empid = w.empid AND w.pid IN (

SELECT w1.pid

FROM workon w1, employee e1

WHERE w1.empid = e1.empid AND lower(e1.name) = 'chen')

AND lower(name)<>'chen';



1. **Increase the salary of employees in engineering division by 10% if they work on more than 1 project.**

UPDATE employee

SET salary = salary \* 1.1

WHERE did = (

SELECT did

FROM division

WHERE lower(dname) = 'engineering'

)

AND empid IN (

SELECT e.empid

FROM employee e, workon w

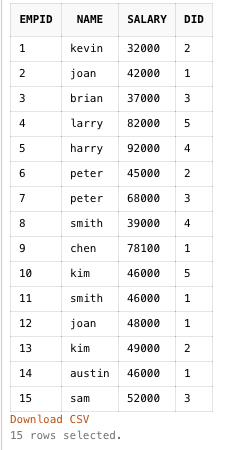
WHERE e.empid=w.empid

GROUP BY e.empid

HAVING count(pid)>1

);





NOTE: Employee Chen is updated from 71000 to 78100.

1. **Increase the budget of a project by 10% if it has more than two employees working on it.**

UPDATE project

SET budget = budget \*1.1

WHERE pid IN (

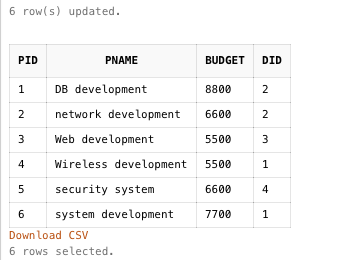
SELECT pid

FROM workon w

GROUP BY pid

HAVING count(empid)>2

);



NOTE: Budget for all six projects were changed since they all had more than two employees working in the same project

1. **Use CREATE statement to create a table Client (ClientName, phone). Note ClientName is primary key and you must define this primary key in CREATE statement. Show the statement.**

CREATE TABLE client (

ClientName VARCHAR (25),

phone VARCHAR (12),

CONSTRAINT client\_clientname\_pk PRIMARY KEY (ClientName)

);

--OR

CREATE TABLE client (

ClientName VARCHAR (25) PRIMARY KEY,

phone VARCHAR (12)

);

1. **Use INSERT statement to add two client records into Client table (make up your own data for clients). Show the INSERT statements and use select statement to show the table contents.**

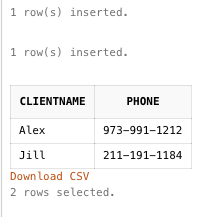
INSERT INTO client

VALUES ('Alex', '973-991-1212');

INSERT INTO client

VALUES ('Jill','211-191-1184');

SELECT \* FROM client;



1. **Use ALTER statements to add a foreign key***ClientName***into the Project table. So that table Client has a one-to-many relationship with table Project. Note, you need to use TWO ALTER statements, one for adding ClientName into Project table; one for adding foreign key constraint into Project table.  Show the ALTER statements.**

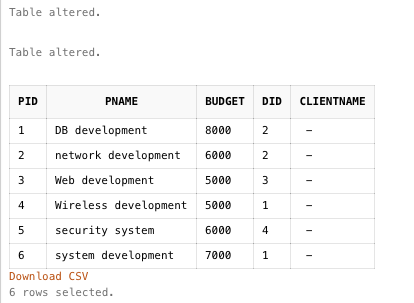
ALTER TABLE project

ADD ClientName VARCHAR (25);

ALTER TABLE project

ADD CONSTRAINT project\_clientname\_fk FOREIGN KEY (ClientName) REFERENCES client(ClientName);

SELECT \* from project;



1. **Use ALTER statement to Add an attribute Project\_Count into Employee table (data type to be integer, refer to the data type used for Workon table (hours)  in loadDB file).**

ALTER TABLE employee

ADD Project\_Count INTEGER;

1. **Use UPDATE statement to fill the value of Project\_count of each employee record in Employee table. Namely, add the count of total number of projects an employee works on into Project\_count in Employee table for each employee.**

UPDATE employee

SET Project\_Count = (

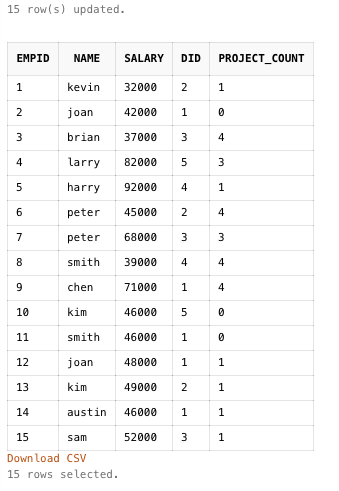
SELECT count(pid)

FROM workon w

WHERE empid=employee.empid

);

SELECT \* from employee;



1. **Create a table Promotion\_list (EMPID, Name, Salary, DivisionName).**

CREATE TABLE Promotion\_List (

EMPID INTEGER,

Name VARCHAR (30),

Salary FLOAT,

DivisionName VARCHAR (25),

CONSTRAINT promotion\_list\_empid\_pk PRIMARY KEY (EMPID)

);

1. **Load Promotion\_list with the information of employees who make less than company average and work on at least 2 projects. Show the code and result.**

INSERT INTO Promotion\_List (EMPID,Name,Salary,DivisionName)

SELECT e.empid,e.name,e.salary,d.dname

FROM employee e, division d, workon w

WHERE e.empid = w.empid AND e.did = d.did AND e.salary < (

SELECT avg(salary)

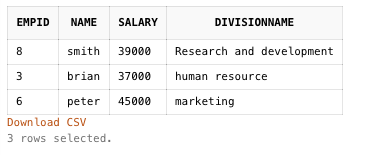
FROM employee

)

GROUP BY e.empid, e.name,e.salary,d.dname

HAVING count(pid)>=2;

SELECT \* FROM Promotion\_List;



1. **Increase the budget of a project by 5% if there is a manager working on it.**

UPDATE project p

SET budget = budget/1.05

WHERE EXISTS (

SELECT pname

FROM division d

WHERE d.did = p.did AND d.managerid IN (

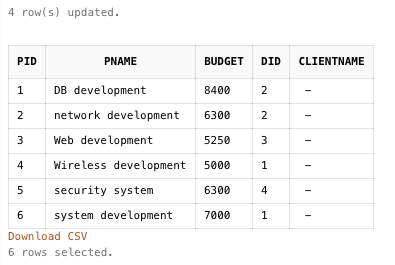
SELECT empid

FROM workon

)

);

SELECT \* FROM project;



1. **List the name of employee who work on a project sponsored by his/her own division. (corelated subquery)**

SELECT e.empid, e.name

FROM employee e, workon w

WHERE e.empid = w.empid AND w.pid IN (

SELECT pid

FROM project p, division d

WHERE p.did = d.did AND e.did = d.did

);



1. **List the name of project that has budget that is higher than ALL projects from 'marketing' division.**

SELECT pname

FROM project

WHERE budget > (

SELECT max(budget)

FROM project p, division d

WHERE p.did = d.did AND lower(d.dname)= 'marketing'

);



1. **List the name of project that has budget that are higher than ALL projects from 'chen's division.**

SELECT pname

FROM project

WHERE budget > (

SELECT max(budget)

FROM project p, employee e

WHERE p.did = e.did AND lower(e.name)= 'chen'

);



1. **List the name of employee who work on more projects than employee 'chen'.**

SELECT e.empid, e.name

FROM employee e, workon w

WHERE e.empid = w.empid

GROUP BY e.empid, e.name

HAVING count (w.pid) > (

SELECT count(w.pid)

FROM employee e, workon w

WHERE e.empid = w.empid AND lower(e.name) = 'chen'

);



1. **List The name of division that has employee(s) who work on other division's project (corelated subquery)**

SELECT DISTINCT dname

FROM division d, employee e

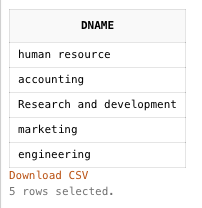
WHERE d.did =e.did AND e.empid IN (

SELECT empid

FROM workon w, project p

WHERE w.pid = p.pid AND e.did <> p.did

);



1. **List the name of employee who works ONLY with his/her divisional colleagues on project(s).**

SELECT e.empid, e.name

FROM employee e, workon w

WHERE e.empid = w.empid AND NOT EXISTS (

SELECT \*

FROM workon ww, project pp

WHERE ww.pid =pp.pid AND e.did <> pp.did AND w.empid NOT IN (

SELECT empid

FROM project

WHERE e.empid <> empid

)

);

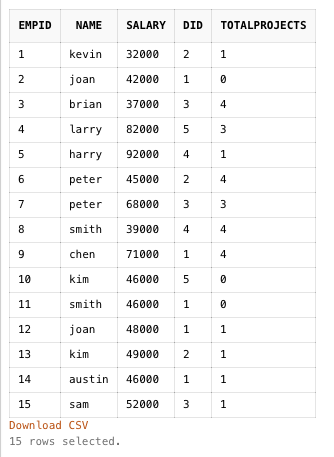


1. **run the following code and test the trigger . Namely,  insert a row in related table (show the INSERT statement) and show the content of the table before and after triggering event (***Assuming the you have created an attribute TotalProjects in Project table).*

create or replace trigger workon\_ai\_trigger  
after insert on workon  
for each row  
begin  
update employee

   set totalprojects = totalprojects + 1  
   where employee.empid = :new.empid;  
end ;

BEFORE TRIGGERING EVENT:

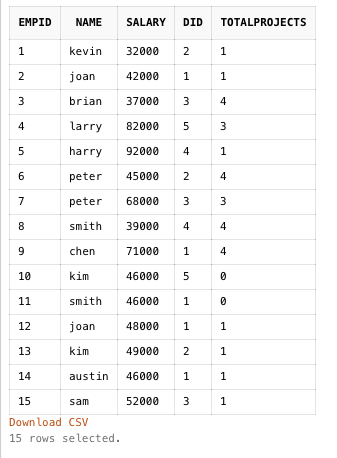


INSERTING ON WORKON TABLE:

INSERT INTO workon

VALUES (1,2,20);

AFTER TRIGGERING EVENT:



NOTE: Once record was created for joan (empid =2), the totalprojects was automatically updated to 1 from 0 with the help of trig

1. **List the name of project that has MOST of employees working on it. (hint: use group by and having with subquery)**

SELECT p.pname

FROM project p, workon w

WHERE p.pid = w.pid

GROUP BY p.pname

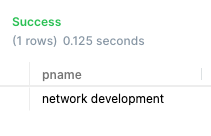
HAVING count(empid)>= ALL (

SELECT count(empid)

FROM workon

GROUP BY pid

);



1. **List the name of division that has more employees whose salary is above the divisional average salary than any other divisions.**

SELECT d.dname, count(empid)

FROM division d, employee e

WHERE d.did = e.did AND salary > (

SELECT avg(salary)

FROM employee ee

WHERE e.did = ee.did

)

GROUP BY d.dname

HAVING count(empid) >= ALL (

SELECT count(empid)

FROM employee e1

WHERE salary > (

SELECT avg(salary)

FROM employee e2

WHERE e1.did = e2.did

)

GROUP BY did

);

--OR

SELECT d.dname, count(empid)

FROM division d, employee e

WHERE d.did = e.did AND salary > (

SELECT avg(salary)

FROM employee ee

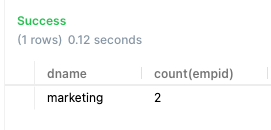
WHERE e.did = ee.did

)

GROUP BY d.dname

ORDER BY count(empid) DESC

LIMIT 1;



1. **Increase the salary of division manager by 1% if he/she works on other division's project (hint: need a co-related condition)**

UPDATE employee e

SET salary = salary \* 1.01

WHERE empid IN (

SELECT d.managerid

FROM division d, workon w, project p

WHERE d.managerid = w.empid AND p.pid = w.pid AND p.did <> e.did

);



1. **List the total number of employees from Chen's division who work with Chen on project development. (note if a Chen's divisional colleague works on more than one projects with Chen, this colleague should be only count once.)**

SELECT count(DISTINCT e.empid) Total\_divisional\_colleague\_working\_with\_Chen

FROM employee e, workon w

WHERE e.empid = w.empid AND e.did = (

SELECT did

FROM employee

WHERE lower(name) = 'chen'

)

AND w.pid IN (

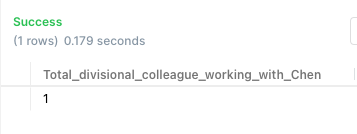
SELECT pid

FROM workon ww, employee ee

WHERE ee.empid = ww.empid AND lower(ee.name) = 'chen'

)

AND lower(e.name) <> 'chen';



1. **For each project, list the name of project, and (1) the total number of employee working on it, (2 )the total number of employees who work on  it but from other division (not the division that sponsors this project), and (3) the total number of employee who work on it and are from the same division that sponsors this project.**

From Project

Note, this query is to show that a (co-related) subquery can appear at the Select clause. The words total, insiders and outsiders are just the given titles/alias for columns of counts returned by subqueries .

SELECT pname, (

SELECT count(empid)

FROM workon w

where pp.pid = w.pid

) total, (

SELECT count(w.empid)

FROM workon w, project p, employee e

WHERE e.empid = w.empid AND p.pid = w.pid AND e.did <> p.did AND pp.pid = w.pid

) outsiders, (

SELECT count(w.empid)

FROM workon w, project p, employee e

WHERE e.empid = w.empid AND p.pid = w.pid AND e.did = p.did AND pp.pid = w.pid

) insiders

FROM project pp;

