create table icc\_world\_cup

(

Team\_1 Varchar(20),

Team\_2 Varchar(20),

Winner Varchar(20)

);

INSERT INTO icc\_world\_cup values('India','SL','India');

INSERT INTO icc\_world\_cup values('SL','Aus','Aus');

INSERT INTO icc\_world\_cup values('SA','Eng','Eng');

INSERT INTO icc\_world\_cup values('Eng','NZ','NZ');

INSERT INTO icc\_world\_cup values('Aus','India','India');

CREATE VIEW view\_name AS

with cte as (

select team\_1 as team\_name,case when team\_1 = winner then 1 else 0 end win\_cnt from icc\_world\_cup

union all

select team\_2 as team\_name,case when team\_2 = winner then 1 else 0 end win\_cnt from icc\_world\_cup )

select team\_name,count(\*) as match\_played , sum(win\_cnt) as No\_of\_wins,count(\*)-sum(win\_cnt) as No\_of\_losses from cte

group by 1

with cte as (

select team\_1 as team\_name, case when team\_1 = winner then 1 else 0 end as win\_cnt from icc\_world\_cup

union all

select team\_2 as team\_name, case when team\_2 = winner then 1 else 0 end as win\_cnt from icc\_world\_cup)

select team\_name, count(\*) as match\_played, sum(win\_cnt) No\_of\_wins, count(\*) - sum(win\_cnt) as No\_of\_losses from cte

group by 1

order by 3 desc

select team\_1 from icc\_world\_cup

create table team (team\_name varchar);

insert into team values('India'),('Pakistan'),('Bangladesh'),('Sri\_lanka')

select t1.team\_name, t2.team\_name from team t1 join

team t2 on t1.team\_name <> t2.team\_name

CREATE TABLE Products (

Order\_date date,

Sales int );

INSERT INTO Products(Order\_date,Sales)

VALUES

('2021-01-01',20), ('2021-01-02',32), ('2021-02-08',45), ('2021-02-04',31),

('2021-03-21',33), ('2021-03-06',19), ('2021-04-07',21), ('2021-04-22',10)

select extract( year from order\_date) as year,to\_char(order\_date,'mon') as month,sum(sales) from Products

group by 1,2

select to\_char(order\_date,'day') as day\_name, to\_char(order\_date,'mon' ) as name\_month ,

extract(month from order\_date) as month\_number from products

CREATE TABLE Applications (

candidate\_id int,

skills varchar);

INSERT INTO Applications(candidate\_id,skills)

VALUES

(101, 'Power BI'), (101, 'Python'), (101, 'SQL'), (102, 'Tableau'), (102, 'SQL'),

(108, 'Python'), (108, 'SQL'), (108, 'Power BI'), (104, 'Python'), (104, 'Excel')

select \* from Applications

select candidate\_id , count(skills) as skill\_count from Applications

where skills in ('Power BI','Python','SQL')

group by 1

having count(skills) = 3

order by 2 desc,1

CREATE TABLE Employee (

EmpID int NOT NULL,

EmpName Varchar,

Gender Char,

Salary int,

City Char(20) )

--- first run the above code then below code

INSERT INTO Employee

VALUES (1, 'Arjun', 'M', 75000, 'Pune'),

(2, 'Ekadanta', 'M', 125000, 'Bangalore'),

(3, 'Lalita', 'F', 150000 , 'Mathura'),

(4, 'Madhav', 'M', 250000 , 'Delhi'),

(5, 'Visakha', 'F', 120000 , 'Mathura')

CREATE TABLE EmployeeDetail (

EmpID int NOT NULL,

Project Varchar,

EmpPosition Char(20),

DOJ date )

--- first run the above code then below code

INSERT INTO EmployeeDetail

VALUES (1, 'P1', 'Executive', '26-01-2019'),

(2, 'P2', 'Executive', '04-05-2020'),

(3, 'P1', 'Lead', '21-10-2021'),

(4, 'P3', 'Manager', '29-11-2019'),

(5, 'P2', 'Manager', '01-08-2020')

select \* from EmployeeDetail Employee

select \* from Employee

----------------------------------------------------------------------------------------------------

'''\*Q1(a): Find the list of employees whose salary ranges between 2L to 3L.'''

select \* from Employee

where salary between 200000 AND 300000

--------------------------------------------------------------------------------------

'''Q1(b): Write a query to retrieve the list of employees from the same city.'''

select e.\* from Employee e join Employee em on e.city = em.city

and e.empid != em.empid

--------------------------------------------------------------------------------------------------

'''Q1(c): Query to find the null values in the Employee table. '''

select \* from Employee

where empid is null

----------------------------------------------------------------------------------

'''Q2(a): Query to find the cumulative sum of employee’s salary.'''

select empid, empname, salary, sum(salary) over (order by empid) as cumulativesum from employee

---------------------------------------------------------------

'''Q2(b): What’s the male and female employees ratio.'''

SELECT (count(\*) filter (where Gender ='M')\* 100.0/count(\*)) as maleratio,

(count(\*) filter (where Gender ='F')\* 100.0/count(\*)) as femaleratio

from Employee

--------------------------------------------------------------------------------

'''Q2(c): Write a query to fetch 50% records from the Employee table.'''

SELECT \* FROM Employee

WHERE EmpID <= (SELECT COUNT(EmpID)/2 from Employee)

-------------------------------------------------------------------------------------------------------

'''Q3: Query to fetch the employee’s salary but replace the LAST 2 digits with ‘XX’ '''

Select salary,concat(left(cast(salary as text),length(cast(salary as text))-2),'XX') as masked\_salary from employee

-----------------------------------------------------------------------------------------------------------------------------------------

'''Q4: Write a query to fetch even and odd rows from Employee table.'''

SELECT \* FROM

(SELECT \*, ROW\_NUMBER() OVER(ORDER BY EmpId) AS

RowNumber

FROM Employee) AS Emp

WHERE Emp.RowNumber % 2 = 0

select \* from (select \*,row\_number() over(order by empid) as rownumber from employee) as emp

where emp.rownumber % 2 = 1

----------------------------------------------------------------------------------------------------------------------------

'''Find Nth highest salary from employee table with and without using the TOP/LIMIT keywords.'''

select \* from employee

where salary = (select max(salary) from employee)

select salary from employee e1

where 2 - 1 = (select count(distinct(salary)) from employee e2

where e2.salary > e1.salary)

----------------------------------------------------------------------------------------------------------------------------------------------

'''Q7(a): Write a query to find and remove duplicate records from a table.'''

select empid, empname,gender,salary,city, count(\*) as duplicatecount from employee

group by empid, empname,gender,salary,city

HAVING COUNT(\*) > 1

delete from employee

where empid in (select empid from employee group by empid having count(\*) > 1);

---------------------------------------------------------------------------------------------------------------------

'''Q7(b): Query to retrieve the list of employees working in same project.'''

with cte as(

select e.empid,e.empname,ed.project from employee e join employeedetail ed on e.empid = ed.empid)

select c1.empid,c1.empname,c2.empname, c2.project from cte c1 join cte c2 on c1.project = c2.project

and c1.empid != c2.empid and c1.empid < c2.empid

--------------------------------------------------------------------------------------------------------------------------------

'''Q8: Show the employee with the highest salary for each project'''

with cte as(

select e.empid,e.empname, e.salary, ed.project , row\_number() over(partition by ed.project order by e.salary) as ranksalary

from employee e join employeedetail ed on e.empid = ed.empid)

select empid,empname,project,salary from cte

where ranksalary = 1

--------------------------------------------------------------------------------------------------------

'''Q9: Query to find the total count of employees joined each year'''

select extract(year from doj) as dojyear ,count(\*) from employeedetail

group by 1

order by 1

------------------------------------------------------------------------------------------------

'''Q10: Create 3 groups based on salary col, salary less than 1L is low, between 1 -2L is medium and above 2L is High'''

select salary, case when salary > 200000 then 'high'

when salary >=100000 and salary <= 200000 then 'medium' else 'low' end as salarystatus

from employee

-------------------------------------------------------------------------------------------------------------

'''Query to pivot the data in the Employee table and retrieve the total

salary for each city.

The result should display the EmpID, EmpName, and separate columns for each city

(Mathura, Pune, Delhi), containing the corresponding total salary.'''

select empid,empname,

sum(case when city = 'Mathura' then salary end ) as mathura,

sum(case when city = 'Pune' then salary end ) as Pune,

sum(case when city = 'Delhi' then salary end ) as Delhi

from employee

group by 1,2

select city, sum(salary) filter (where city = 'Mathura') from employee

group by 1

order by 2 asc

limit 1

-----------------------------------------------------------------------------

------Rough-------------

'''Q8: Show the employee with the highest salary for each project'''

with cte as (

select e1.empid,e1.empname,ed.project,e1.salary,row\_number() over(partition by ed.project order by salary) maxsalary

from employee e1 join employeedetail ed on e1.empid = ed.empid)

select \* from cte

where maxsalary = 1

select team\_1,max(length(team\_1)) from icc\_world\_cup

group by 1

select \* from view\_name

select ed.empposition , count(\*) from employee e1 join employeedetail ed on e1.empid = ed.empid

group by 1

ALTER TABLE [table\_name] MODIFY

[column\_name] [new\_data\_type];

------------------------------------------------------------------------------------------------------------------------

--------------------------------HackerRank-------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------

'''Query all columns for all American cities in the CITY table with populations larger

than 100000. The CountryCode for America is USA.'''

select \* from city

where population > 100000 and countrycode = 'USA'

------------------------------------------------------------------------------------------------------------

'''Query the NAME field for all American cities in the CITY table with populations larger than 120000.

The CountryCode for America is USA.'''

select name from city where population > 120000 and countrycode = 'USA'

-------------------------------------------------------------------------------------------------------------

'''Query all columns (attributes) for every row in the CITY table.'''

select \* from city

--------------------------------------------------------------------------------------------------------------

'''Query all columns for a city in CITY with the ID 1661.'''

select \* from city where id = 1661

-------------------------------------------------------------------------------------------------------------

'''Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.'''

select \* from city where COUNTRYCODE = 'jpn'

--------------------------------------------------------------------------------------------------------------

'''Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.'''

select name from city where COUNTRYCODE = 'jpn'

--------------------------------------------------------------------------------------------------------------

'''Query a list of CITY and STATE from the STATION table.'''

select city, state from station

----------------------------------------------------------------------------------------------------------

'''Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any

order, but exclude duplicates from the answer.'''

select distinct(city) from station where id % 2 = 0

------------------------------------------------------------------------------------------------------

'''ind the difference between the total number of CITY entries in the table and the number of distinct

CITY entries in the table.'''

select count(city) - count(distinct(city)) from station

------------------------------------------------------------------------------------------------------------

'''Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths

(i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one

that comes first when ordered alphabetically.'''

(select city,length(city) as city\_len from station

where length(city)=(select min(length(city)) from station) order by city limit 1)

union

(select city,length(city) as city\_len from station

where length(city)=(select max(length(city)) from station order by city limit 1));

-------------------------------------------------------------------------------------------------------------------

'''Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION.

Your result cannot contain duplicates.'''

SELECT DISTINCT CITY FROM STATION WHERE LEFT(CITY,1) IN ('a','e','i','o','u');-----mysql

SELECT DISTINCT CITY FROM STATION WHERE LOWER(CITY) SIMILAR TO '[aeiou]%' ----------postgresql

--------------------------------------------------------------------------------------------------------------

'''Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result

cannot contain duplicates.'''

SELECT DISTINCT CITY FROM STATION WHERE right(CITY,1) IN ('a','e','i','o','u');-----mysql

-------------------------------------------------------------------------------------------------

'''Query the list of CITY names from STATION which have vowels (i.e., a, e, i, o, and u) as both their

first and last characters. Your result cannot contain duplicates.'''

SELECT DISTINCT CITY FROM STATION WHERE (LEFT(CITY,1) IN ('a','e','i','o','u'))

and (right(CITY,1) IN ('a','e','i','o','u'))

-----------------------------------------------------------------------------------------------

'''Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.'''

SELECT DISTINCT CITY FROM STATION WHERE LEFT(CITY,1) NOT IN('a','e','i','o','u','A','E','I','O','U') ;

-----------------------------------------------------------------------------------------------------

'''Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.'''

select distinct city from station where right(city,1) not in ('a','e','i','o','u')

----------------------------------------------------------------------------------------------------------------------------

'''Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels.

Your result cannot contain duplicates.'''

(select distinct city from station where

LEFT(CITY,1) NOT IN('a','e','i','o','u','A','E','I','O','U') )

union

(select distinct city from station where

right(CITY,1) NOT IN('a','e','i','o','u','A','E','I','O','U'))

----------------------------------------------------------------------------------------------------------------------------

'''Query the list of CITY names from STATION that do not start with vowels and do not end

with vowels. Your result cannot contain duplicates.'''

select distinct city from station where

(LEFT(CITY,1) not in ('a','e','i','o','u','A','E','I','O','U'))

and

(right(CITY,1) not in ('a','e','i','o','u','A','E','I','O','U'))

----------------------------------------------------------------------------------------------------------------------------

'''Query the Name of any student in STUDENTS who scored higher than Marks. Order your output by the last three characters of

each name. If two or more students both have names ending in the same last three characters (i.e.: Bobby, Robby, etc.),

secondary sort them by ascending ID.'''

select name from students

where marks > 75

order by right(name,3),id

----------------------------------------------------------------------------------------------------------------------------

'''Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in

alphabetical order.'''

select name from employee order by name

----------------------------------------------------------------------------------------------------------------------------

'''Write a query that prints a list of employee names (i.e.: the name attribute) for employees in Employee having a salary

greater than per month who have been employees for less than months. Sort your result by ascending employee\_id.'''

select name from employee here salary > 2000 and months < 10

Write a query identifying the type of each record in the TRIANGLES table using its three side lengths. Output one of the following statements for each record in the table:

----------------------------------------------------------------------------------------------------------------------------

'''Equilateral: Its a triangle with sides of equal length.

Isosceles: Its a triangle with sides of equal length.

Scalene: Its a triangle with sides of differing lengths.

Not A Triangle: The given values of A, B, and C dont form a triangle.'''

select

case when A+B <= C OR B+C <= A OR A+C <= B then 'Not A Triangle'

when A = B AND B = C and A=C then 'Equilateral'

when A = B AND B != C OR A = C AND B != C OR B = C AND A != B then 'Isosceles'

ELSE 'Scalene'

end

from triangles

----------------------------------------------------------------------------------------------------------------------------

'''Query an alphabetically ordered list of all names in OCCUPATIONS, immediately followed by the first letter of each

profession as a parenthetical (i.e.: enclosed in parentheses). For example: AnActorName(A), ADoctorName(D),

AProfessorName(P), and ASingerName(S).

Query the number of ocurrences of each occupation in OCCUPATIONS. Sort the occurrences in ascending order, and

output them in the following format:'''

SELECT CONCAT(Name, CASE WHEN Occupation = 'Doctor' THEN '(D)'

WHEN Occupation = 'Actor' THEN '(A)' WHEN Occupation = 'Professor' THEN '(P)' ELSE '(S)' END)

FROM OCCUPATIONS ORDER BY Name; SELECT CONCAT('There are a total of ', count(),' ', LOWER(Occupation),'s.')

FROM OCCUPATIONS GROUP BY Occupation ORDER BY COUNT(),Occupation ASC;

----------------------------------------------------------------------------------------------------------------------------

'''You are given a table, BST, containing two columns: N and P, where N represents the value of a node in Binary Tree,

and P is the parent of N.'''

select N,

case

when P is null then 'Root'

when N in (select distinct P from BST where P is not null) then 'Inner'

else 'Leaf'

end from BST order by N

----------------------------------------------------------------------------------------------------------------------------

'''Ambers conglomerate corporation just acquired some new companies. Each of the companies'''

select c.company\_code,c.founder, count(distinct(e.lead\_manager\_code)),count(distinct(e.senior\_manager\_code)),

count(distinct(e.manager\_code)),count(distinct(e.employee\_code))

from Employee e left join company c on c.company\_code = e.company\_code

group by 1,2

order by 1

----------------------------------------------------------------------------------------------------------------------------

'''Query a count of the number of cities in CITY having a Population larger than .'''

select count(\*) from city where population > 100000

----------------------------------------------------------------------------------------------------------------------------

'''Query the total population of all cities in CITY where District is California.'''

select sum(population) from city where District = 'California'

----------------------------------------------------------------------------------------------------------------------------

'''Query the average population for all cities in CITY, rounded down to the nearest integer.'''

SELECT ROUND(AVG(population)) FROM city

----------------------------------------------------------------------------------------------------------------------------

'''Query the sum of the populations for all Japanese cities in CITY. The COUNTRYCODE for Japan is JPN.'''

select sum(population) from city where countrycode = 'JPN'

----------------------------------------------------------------------------------------------------------------------------

'''Query the difference between the maximum and minimum populations in CITY.'''

SELECT (MAX(population) - MIN(population)) FROM city;

----------------------------------------------------------------------------------------------------------------------------

'''Samantha was tasked with calculating the average monthly salaries for all employees in the EMPLOYEES table, but did not

realize her keyboards key was broken until after completing the calculation. She wants your help finding the

difference between her miscalculation (using salaries with any zeros removed), and the actual average salary.

Write a query calculating the amount of error (i.e.: average monthly salaries), and round it up to the next integer.

Input Format'''

SELECT ROUND(AVG(salary)) - ROUND(AVG(REPLACE(salary, '0', ''))) FROM employees;----- mysql

select round(avg(salary)) - round(avg(cast(replace(cast(salary as text),'0','') as integer))) as salary\_difference from

employee--------- postgresql

----------------------------------------------------------------------------------------------------------------------------

'''We define an employees total earnings to be their monthly worked, and the maximum total earnings to be the maximum total

earnings for any employee in the Employee table. Write a query to find the maximum total earnings for all employees

as well as the total number of employees who have maximum total earnings. Then print these values as space-separated

integers.'''

select Max(Earnings), Count(empname) from

(Select (Salary\*empid) as Earnings, empname from Employee ) as emp

Group by earnings order by earnings desc limit 1 -------------------- pos

----------------------------------------------------------------------------------------------------------------------------

'''Query the following two values from the STATION table:

The sum of all values in LAT\_N rounded to a scale of decimal places.

The sum of all values in LONG\_W rounded to a scale of decimal places.'''

select round(sum(lat\_n),2) , round(sum(long\_w),2) from station

----------------------------------------------------------------------------------------------------------------------------

'''Query the sum of Northern Latitudes (LAT\_N) from STATION having values greater than and less than . Truncate your answer

to decimal places.'''

select round(sum(lat\_n),4) from station where lat\_n > 38.7880 and lat\_n < 137.2345

----------------------------------------------------------------------------------------------------------------------------

'''Query the greatest value of the Northern Latitudes (LAT\_N) from STATION that is less than .

Truncate your answer to decimal places.'''

select round(max(lat\_n),4) from station where lat\_n < 137.2345

----------------------------------------------------------------------------------------------------------------------------

'''Query the Western Longitude (LONG\_W) for the largest Northern Latitude (LAT\_N) in STATION that is less than . Round your

answer to decimal places..'''

select round(long\_w,4) from station where lat\_n < 137.2345 order by lat\_n desc limit 1

----------------------------------------------------------------------------------------------------------------------------

'''Query the smallest Northern Latitude (LAT\_N) from STATION that is greater than . Round your answer to decimal places.'''

select round(lat\_n,4) from station where lat\_n > 38.7780 order by lat\_n limit 1

----------------------------------------------------------------------------------------------------------------------------

'''Query the Western Longitude (LONG\_W)where the smallest Northern Latitude (LAT\_N) in STATION is greater than.Round your answer to decimal places.'''

select round(long\_w,4) from station where lat\_n > 38.7780 order by lat\_n limit 1

----------------------------------------------------------------------------------------------------------------------------

'''Consider and to be two points on a 2D plane.

happens to equal the minimum value in Northern Latitude (LAT\_N in STATION).

happens to equal the minimum value in Western Longitude (LONG\_W in STATION).

happens to equal the maximum value in Northern Latitude (LAT\_N in STATION).

happens to equal the maximum value in Western Longitude (LONG\_W in STATION).

Query the Manhattan Distance between points and and round it to a scale of decimal places.'''

SELECT ROUND((MAX(LAT\_N)-MIN(LAT\_N))+ (MAX(LONG\_W)-MIN(LONG\_W)), 4) FROM STATION

----------------------------------------------------------------------------------------------------------------------------

'''Consider and to be two points on a 2D plane where are the respective minimum and maximum values of Northern Latitude

(LAT\_N) and are the respective minimum and maximum values of Western Longitude (LONG\_W) in STATION.

Query the Euclidean Distance between points and and format your answer to display decimal digits.'''

select round(sqrt(pow(max(lat\_n)-min(lat\_n),2) + pow(max(long\_w)-min(long\_w),2)),4) from station;

----------------------------------------------------------------------------------------------------------------------------

'''A median is defined as a number separating the higher half of a data set from the lower half. Query the median of the Northern

Latitudes (LAT\_N) from STATION and round your answer to decimal places.'''

WITH cte AS ( SELECT RANK() OVER (ORDER BY LAT\_N) AS rnk, LAT\_N FROM STATION )

SELECT ROUND(LAT\_N, 4) FROM cte WHERE rnk = CEILING((SELECT COUNT(LAT\_N) FROM STATION) / 2)

----------------------------------------------------------------------------------------------------------------------------

'''P(R) represents a pattern drawn by Julia in R rows. The following pattern represents P(20)'''

with RECURSIVE cte as( select 20 as rownumber

union all

select rownumber - 1 from cte where rownumber > 0)

select repeat(' \* ',rownumber) as pattern from cte

----------------------------------------------------------------------------------------------------------------------------

'''P(R) represents a pattern drawn by Julia in R rows. The following pattern represents P(20):'''

with RECURSIVE cte as( select 1 as rownumber

union all

select rownumber + 1 from cte where rownumber < 20)

select repeat(' \* ',rownumber) as pattern from cte

----------------------------------------------------------------------------------------------------------------------------

'''Write a query to print all prime numbers less than or equal to . Print your result on a single line, and use the ampersand ()

character as your separator (instead of a space).'''

WITH RECURSIVE cte(n) AS ( SELECT 2 UNION ALL SELECT n+1 from cte where n<1000 )

SELECT GROUP\_CONCAT(cte.n SEPARATOR '&') FROM cte WHERE NOT EXISTS (

select n FROM cte AS c WHERE c.n > 1 AND c.n <= SQRT(cte.n) AND cte.n % c.n = 0 )

----------------------------------------------------------------------------------------------------------------------------

'''Given the CITY and COUNTRY tables, query the sum of the populations of all cities where the CONTINENT is 'Asia'.'''

select sum(city.population) from city join country on city.countrycode = country.code where continent = 'Asia';

----------------------------------------------------------------------------------------------------------------------------

'''Given the CITY and COUNTRY tables, query the names of all cities where the CONTINENT is 'Africa'.'''

select city.name from city join country on city.countrycode = country.code

where continent = 'Africa'

----------------------------------------------------------------------------------------------------------------------------

'''Given the CITY and COUNTRY tables, query the names of all the continents (COUNTRY.Continent) and their respective average

city populations (CITY.Population) rounded down to the nearest integer.'''

SELECT COUNTRY.Continent, FLOOR(AVG(CITY.Population)) FROM COUNTRY

INNER JOIN CITY ON COUNTRY.Code = CITY.CountryCode GROUP BY COUNTRY.Continent;

----------------------------------------------------------------------------------------------------------------------------

'''You are given two tables: Students and Grades. Students contains three columns ID, Name and Marks.'''

SELECT IF(Grades.Grade > 7, Students.Name, NULL), Grades.Grade, Students.Marks FROM Students

LEFT JOIN Grades ON Students.Marks BETWEEN Grades.Min\_Mark AND Grades.Max\_Mark ORDER BY Grades.Grade DESC, Students.Name;

----------------------------------------------------------------------------------------------------------------------------

'''Julia just finished conducting a coding contest, and she needs your help assembling the leaderboard! Write a query to print

the respective hacker\_id and name of hackers who achieved full scores for more than one challenge. Order your output in

descending order by the total number of challenges in which the hacker earned a full score. If more than one hacker received

full scores in same number of challenges, then sort them by ascending hacker\_id.'''

select h.hacker\_id, h.name from Hackers h

join Submissions s ON h.hacker\_id = s.hacker\_id

join Challenges c ON s.challenge\_id = c.challenge\_id

join Difficulty d ON d.difficulty\_level = c.difficulty\_level

where d.score = s.score group by hacker\_id, name having count(c.challenge\_id) > 1

order by count(c.challenge\_id) desc, h.hacker\_id asc

----------------------------------------------------------------------------------------------------------------------------

'''ou did such a great job helping Julia with her last coding contest challenge that she wants you to work on this one, too!'''

SELECT s.hacker\_id, MAX(h.name) AS hacker\_name, SUM(s.max\_score) AS total\_max\_score FROM

( SELECT hacker\_id, challenge\_id, MAX(score) AS max\_score FROM Submissions GROUP BY hacker\_id, challenge\_id ) AS s

LEFT JOIN Hackers h ON s.hacker\_id = h.hacker\_id GROUP BY s.hacker\_id HAVING SUM(s.max\_score) > 0

ORDER BY total\_max\_score DESC, s.hacker\_id;

----------------------------------------------------------------------------------------------------------------------------

'''You are given three tables: Students, Friends and Packages. Students contains two columns: ID and Name. Friends contains

two columns: ID and Friend\_ID (ID of the ONLY best friend). Packages contains two columns: ID and Salary

(offered salary in $ thousands per month).

'''

SELECT s1.name AS name1 FROM Friends f JOIN Students s1 ON s1.id = f.id

JOIN Packages p1 ON p1.id = f.id

JOIN Packages p2 ON p2.id = f.friend\_id AND p2.salary > p1.salary ORDER BY p2.salary