

QUESTION BANK DOCKET

**Table of Contents**

[1 Source 2](#_Toc253285289)

[2 Category 3](#_Toc965934774)

[3 Tech Area 3](#_Toc613746458)

[4 Complexity 3](#_Toc904143571)

[5 Question 3](#_Toc1426514880)

[6 Answer 3](#_Toc441207452)

[7 References 3](#_Toc1838997797)

# Source

Enter the name of the source where you got the question from

[HackerRank]

# Category

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Tech Area

SQL

# Complexity

Complex

# Question

Write a query to print all *prime numbers* less than or equal to 1000. Print your result on a single line, and use the ampersand (&) character as your separator (instead of a space).

For example, the output for all prime numbers <=10 would be:

2&3&5&7

# Answer

**Steps 1:**First we will DECLARE a variable I with initial value 2.

**Query:**

DECLARE @I INT=2

**Step 2:**Then we will DECLARE a variable PRIME with an initial value of 0 (this will set the value of PRIME).

**Query:**

DECLARE @PRIME INT=0

**Step 3:** Table Definition

We will create a temporary table variable that will hold prime numbers (using DECLARE and TABLE keywords).

**Query:**

DECLARE @OUTPUT TABLE (NUM INT)

**Step 4:**Now we will use nested while loop, same as we write a program for prime numbers.

**Query:**

DECLARE @I INT=2

DECLARE @PRIME INT=0

DECLARE @OUTPUT TABLE (NUM INT)

WHILE @I<=1000

BEGIN

DECLARE @J INT = @I-1

SET @PRIME=1

WHILE @J>1

BEGIN

IF @I % @J=0

BEGIN

SET @PRIME=0

END

SET @J=@J-1

END

IF @PRIME =1

BEGIN

INSERT @OUTPUT VALUES (@I)

END

SET @I=@I+1

END

SELECT \* FROM @OUTPUT

# Source

Enter the name of the source where you got the question from

[Hacker Rank]

# Category

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Tech Area

SQL

# Complexity

Simple

# Question

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.  
The **STATION** table is described as follows:



where **LAT\_N** is the northern latitude and **LONG\_W** is the western longitude.

# Answer

**SELECT** **DISTINCT** CITY

**FROM** STATION

**WHERE** ID % **2** = **0**;

# Source

Enter the name of the source where you got the question from

[Hacker Rank]

# Category

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Tech Area

SQL

# Complexity

Medium

# Question

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**: It's a triangle with 3  sides of equal length.
* **Isosceles**: It's a triangle with 2 sides of equal length.
* **Scalene**: It's a triangle with 3  sides of differing lengths.
* **Not A Triangle**: The given values of *A*, *B*, and *C* don't form a triangle.

**Input Format**

The **TRIANGLES** table is described as follows:



Each row in the table denotes the lengths of each of a triangle's three sides.

**Sample Input**



**Sample Output**

Isosceles

Equilateral

Scalene

Not A Triangle

# Answer

**SELECT**

**CASE**

**WHEN** (A + B <= **C**)

**OR** (B + **C** <= A)

**OR** (A + **C** <= B) **THEN** 'Not A Triangle'

**WHEN** (A = B)

**AND** (B = **C**) **THEN** 'Equilateral'

**WHEN** ((A = B) &(A != **C**))

**OR** ((B = **C**) &(B != A))

**OR** ((A = **C**) &(A != B)) **THEN** 'Isosceles'

**WHEN** (A != B)

**AND** (B != **C**)

**AND** (A != **C**) **THEN** 'Scalene'

**END** **AS** Triangle\_Type

**FROM**

TRIANGLES;

# Source

Enter the name of the source where you got the question from

[Hacker Rank]

# Category

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Tech Area

SQL

# Complexity

Simple

# Question

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

The **CITY** table is described as follows :  


# Answer

**SELECT** \* **FROM** CITY

**WHERE** COUNTRYCODE = "JPN";

# Source

Enter the name of the source where you got the question from

[Hacker Rank]

# Category

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Tech Area

SQL

# Complexity

Medium

# Question

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*.



Write a query to find the node type of *Binary Tree* ordered by the value of the node. Output one of the following for each node:

* *Root*: If node is root node.
* *Leaf*: If node is leaf node.
* *Inner*: If node is neither root nor leaf node.

**Sample Input**



**Sample Output**

1 Leaf

2 Inner

3 Leaf

5 Root

6 Leaf

8 Inner

9 Leaf

**Explanation**

The *Binary Tree* below illustrates the sample:



# Answer

**SELECT** N,

**CASE**

**WHEN** P **IS** **NULL** **THEN** 'Root'

**WHEN** (**SELECT** **COUNT**(\*) **FROM** BST **WHERE** B.N=P)>**0** **THEN** 'Inner'

**ELSE** 'Leaf'

**END** **AS** PLACE

**FROM** BST B

**ORDER** **BY** N;

# Source

Enter the name of the source where you got the question from

[Hacker Rank]

# Category

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Tech Area

SQL

# Complexity

Simple

# Question

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

The **CITY** table is described as follows:  


# Answer

**SELECT** \* **FROM** CITY

**WHERE** ID = **1661**;

# Source

Enter the name of the source where you got the question from

[Hacker Rank]

# Category

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Tech Area

SQL

# Complexity

Medium

# Question

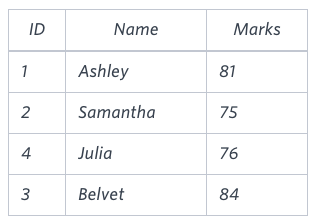
Query the *Name* of any student in **STUDENTS** who scored higher than 75  *Marks*. Order your output by the *last three characters* of each name. If two or more students both have names ending in the same last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending *ID*.

**Input Format**

The **STUDENT’S** table is described as follows: 

The *Name* column only contains uppercase (A-Z) and lowercase (a-z) letters.

**Sample Input**



# Answer

**SELECT** NAME

**FROM** STUDENTS

**WHERE** MARKS > **75**

**ORDER** **BY** **RIGHT**(NAME,**3**) **ASC**, ID **ASC**;

# Source

Enter the name of the source where you got the question from

[Hacker Rank]

# Category

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Tech Area

SQL

# Complexity

Simple

# Question

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

**Input Format**

The **CITY** table is described as follows:



# Answer

**SELECT** **SUM** (POPULATION)

**FROM** CITY

**WHERE** DISTRICT = 'California';

# Source

Enter the name of the source where you got the question from

[Hacker Rank]

# Category

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Tech Area

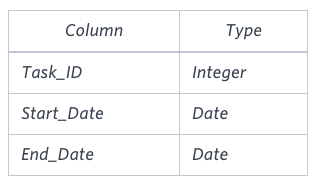
SQL

# Complexity

Complex

# Question

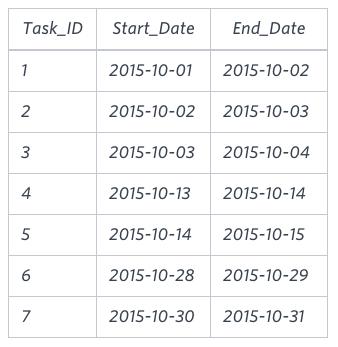
You are given a table, *Projects*, containing three columns: *Task\_ID*, *Start\_Date* and *End\_Date*. It is guaranteed that the difference between the *End\_Date* and the *Start\_Date* is equal to *1* day for each row in the table.



If the *End\_Date* of the tasks are consecutive, then they are part of the same project. Samantha is interested in finding the total number of different projects completed.

Write a query to output the start and end dates of projects listed by the number of days it took to complete the project in ascending order. If there is more than one project that have the same number of completion days, then order by the start date of the project.

**Sample Input**



**Sample Output**

2015-10-28 2015-10-29

2015-10-30 2015-10-31

2015-10-13 2015-10-15

2015-10-01 2015-10-04

# Answer

**SELECT** s.Proj\_Start\_Date, **min**(e.Proj\_End\_Date) **as** Real\_Proj\_End\_Date

**FROM**

(**SELECT** Start\_Date **as** Proj\_Start\_Date **FROM** Projects **WHERE** Start\_Date **NOT** **IN** (**SELECT** End\_Date **FROM** Projects)) s,

(**SELECT** End\_Date **as** Proj\_End\_Date **FROM** Projects **WHERE** End\_Date **NOT** **IN** (**SELECT** Start\_Date **FROM** Projects)) e

**WHERE** s.Proj\_Start\_Date < e.Proj\_End\_Date

**GROUP** **BY** s.Proj\_Start\_Date

**ORDER** **BY** DATEDIFF(**min**(e.Proj\_End\_Date), s.Proj\_Start\_Date) **ASC**, s.Proj\_Start\_Date **ASC**;

# Source

Enter the name of the source where you got the question from

[HackerRank]

# Category

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Tech Area

SQL

# Complexity

Simple

# Question

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

**Input Format**

The **CITY** table is described as follows:



# Answer

**SELECT** **MAX**(POPULATION) - **MIN**(POPULATION)

**FROM** CITY



QUESTION BANK DOCKET

**Table of Contents**

[1 Source 2](#_Toc253285289)

[2 Category 3](#_Toc965934774)

[3 Tech Area 3](#_Toc613746458)

[4 Complexity 3](#_Toc904143571)

[5 Question 3](#_Toc1426514880)

[6 Answer 3](#_Toc441207452)

[7 References 3](#_Toc1838997797)

# Source

Coderbyte

# Category

SQL(Programming)

# Tech Area

Structured Query Language(SQL)

# Complexity

Medium

# Question

CODERBYTE SQL EMPLOYEE SALARIES CHALLENGE

Your table: maintable\_M3LJZ

MySQL version: 5.5.56-log

+-------+---------------+------------+------------------+---------------+

| ID | Name | DivisionID | ManagerID | Salary |

+-------+---------------+------------+------------------+---------------+

| 356 | Daniel Smith | 100 | 133 | 40000 |

| 122 | Arnold Sully | 101 | null | 60000 |

| 467 | Lisa Roberts | 100 | null | 80000 |

| 112 | Mary Dial | 105 | 467 | 65000 |

| 775 | Dennis Front | 103 | null | 90000 |

| 111 | Larry Weis | 104 | 35534 | 75000 |

| 222 | Mark Red | 102 | 133 | 86000 |

| 577 | Robert Night | 105 | 12353 | 76000 |

| 133 | Susan Wall | 105 | 577 | 110000 |

+-------+---------------+------------+------------------+---------------+

Problem Statement

In this MySQL challenge, your query should return the information for the employee with

the third highest salary. Write a query that will find this employee and return that row,

but then replace the DivisionID column with the corresponding DivisionName from the table

cb\_companydivisions. You should also replace the ManagerID column with the ManagerName if

the ID exists in the table and is not NULL.

Your output should look like the following table.

+-------+---------------+--------------+--------------------+---------------+

| ID | Name | DivisionName | ManagerName | Salary |

+-------+---------------+--------------+--------------------+---------------+

| 222 | Mark Red | Sales | Susan Wall | 86000 |

+-------+---------------+--------------+--------------------+---------------+

# Answer

|  |
| --- |
| SELECT |
|  | maintable\_M3LJZ.ID, |
|  | maintable\_M3LJZ.Name, |
|  | cb\_companydivisions.DivisionName AS DivisionName, |
|  | t.Name AS ManagerName, |
|  | maintable\_M3LJZ.Salary |
|  | FROM maintable\_M3LJZ |
|  |  |
|  | JOIN cb\_companydivisions ON cb\_companydivisions.id = maintable\_M3LJZ.DivisionID |
|  |  |
|  | JOIN |
|  | ( |
|  | SELECT |
|  | Name, |
|  | ID |
|  | FROM |
|  | maintable\_M3LJZ |
|  | )AS t ON t.ID = maintable\_M3LJZ.ManagerID |
|  |  |
|  | WHERE Salary = ( |
|  | SELECT |
|  | MIN(Salary) |
|  | FROM ( |
|  | SELECT |
|  | \* |
|  | FROM |
|  | maintable\_M3LJZ |
|  | ORDER BY salary DESC |
|  | LIMIT 3 |
|  | ) AS x |
|  | ) |

# References

[CoderByte-Challenges-Solutions/sql\_employee\_salaries.sql at master · Hardanish-Singh/CoderByte-Challenges-Solutions (github.com)](https://github.com/Hardanish-Singh/CoderByte-Challenges-Solutions/blob/master/SQL/sql_employee_salaries.sql)

# Source

Coderbyte

# Category

SQL(Programming)

# Tech Area

Structured Query Language(SQL)

# Complexity

Medium

# Question

CODERBYTE SQL MEMBER COUNT CHALLENGE

Your table: maintable\_L99ON

MySQL version: 5.5.56-log

+-------+---------------+----------+--------------------+---------------+-----+

| ID | FirstName | LastName | ReportsTo | Position | Age |

+-------+---------------+----------+--------------------+---------------+-----+

| 1 | Daniel | Smith | Bob Boss | Engineer | 25 |

| 2 | Mike | White | Bob Boss | Contractor | 22 |

| 3 | Jenny | Richards | null | CEO | 45 |

| 4 | Robert | Black | Daniel Smith | Sales | 22 |

| 5 | Noah | Fritz | Jenny Richards | Assistant | 30 |

| 6 | David | S | Jenny Richards | Director | 32 |

| 7 | Ashley | Wells | David S | Assistant | 25 |

| 8 | Ashley | Johnson | null | Intern | 25 |

+-------+---------------+----------+--------------------+---------------+-----+

Problem Statement

In this MySQL challenge, your query should return the names of the people who are reported

to (excluding null values) the number of members that report to them, and the average the

number of members that report to them, and the average age of those members as an integer

The rows should be ordered by the names in alphabetical order. Your output should look

like the following table.

+-----------------+------------+------------+

| ReportsTo | Members | Average Age|

+-----------------+------------+------------+

| Bob Boss | 2 | 24 |

| Daniel Smith | 1 | 22 |

| David S | 1 | 25 |

| Jenny Richards | 2 | 31 |

+-----------------+------------+------------+

# Answer

|  |
| --- |
| SELECT |
|  | ReportsTo, |
|  | COUNT(ReportsTo) AS Members, |
|  | CEIL(AVG(AGE)) AS `Average Age` |
|  | FROM |
|  | maintable\_L99ON |
|  | WHERE ReportsTo IS NOT NULL |
|  | GROUP BY ReportsTo |
|  | ORDER BY ReportsTo |

# References

[CoderByte-Challenges-Solutions/sql\_member\_count.sql at master · Hardanish-Singh/CoderByte-Challenges-Solutions (github.com)](https://github.com/Hardanish-Singh/CoderByte-Challenges-Solutions/blob/master/SQL/sql_member_count.sql)

# Source

Coderbyte

# Category

SQL(Programming)

# Tech Area

Structured Query Language(SQL)

# Complexity

Medium

# Question

CODERBYTE SQL USER LOGINS CHALLENGE

Your table: maintable\_O9AAP

MySQL version: 5.5.56-log

+---------------+---------------+

| ID | DateJoined |

+---------------+---------------+

| 2343434 | 2017-01-06 |

| 6343544 | 2017-01-12 |

| 355445 | 2017-01-16 |

| 2322324 | 2017-01-25 |

| 122323 | 2017-02-05 |

| 454343 | 2017-02-07 |

| 2342342 | 2017-02-21 |

| 1322323 | 2017-03-05 |

| 5341124 | 2017-03-07 |

| 13663442 | 2017-03-14 |

| 26323242 | 2017-03-16 |

| 12263545 | 2017-03-25 |

| 23423423 | 2017-03-25 |

| 34452352 | 2017-03-25 |

| 235534534| 2017-03-25 |

| 22224233 | 2017-03-26 |

| 3353235 | 2017-04-05 |

| 56243434 | 2017-04-14 |

| 33463266 | 2017-04-21 |

| 6634653 | 2017-05-07 |

| 2342324 | 2017-05-14 |

| 2366436 | 2017-05-16 |

| 1123124 | 2017-05-25 |

| 3534534 | 2017-05-25 |

| 46763543 | 2017-05-25 |

| 33463255 | 2017-05-25 |

+---------------+---------------+

Problem Statement

In this MySQL challenge, the table provided shows all new users signing up on a specific

date in the format YYYY-MM-DD. Your query should output the change from one month to the

next. Because the first month has no preceding month, your output should skip that row.

Your output should look like the following table.

+-----------------+---------------------+

| Month | MonthToMonthChange |

+-----------------+---------------------+

| February | -1 |

| March | 6 |

| April | -6 |

| May | 4 |

+-----------------+---------------------+

# Answer

|  |
| --- |
|  |
|  | (SOLUTION ONE WITHOUT LAG FUNCTION) |
|  |  |
|  | SELECT |
|  | MONTHNAME(STR\_TO\_DATE(MONTH(DateJoined), '%m')) AS Month, |
|  | date1 - previous AS MonthToMonthChange |
|  | FROM |
|  | ( |
|  | SELECT |
|  | date1, |
|  | DateJoined, |
|  | @prev previous, |
|  | @prev := date1 AS prev |
|  | FROM |
|  | ( |
|  | SELECT |
|  | COUNT(DateJoined) AS date1, |
|  | DateJoined, |
|  | ( SELECT @prev := '' ) r |
|  | FROM |
|  | maintable\_O9AAP |
|  | GROUP BY MONTH(DateJoined) |
|  | ORDER BY DateJoined |
|  | )AS SubQuery1 |
|  | ) AS SubQuery2 |
|  | WHERE previous > 0 |

# References

[CoderByte-Challenges-Solutions/sql\_user\_logins.sql at master · Hardanish-Singh/CoderByte-Challenges-Solutions (github.com)](https://github.com/Hardanish-Singh/CoderByte-Challenges-Solutions/blob/master/SQL/sql_user_logins.sql)

# Source

Coderbyte

# Category

SQL(Programming)

# Tech Area

Structured Query Language(SQL)

# Complexity

Medium

# Question

CODERBYTE SQL VENDOR JOIN CHALLENGE

In this MySQL challenge, your query should return the vendor information along with

the values from the table cb\_vendorinformation. You should combine the values of the

two tables based on the GroupID column. The final query should only print out the

GroupID, CompanyName, and final count of all rows that are grouped into each company

name under a column titled Count. The output table should be then sorted by the

Count column and then sorted by GroupID so that a higher number appears first.

Your output should look like the following table.

+---------+---------------------+-------+

| GroupID | CompanyName | Count |

+---------+---------------------+-------+

| 27| Machinx | 1|

| 5| WaterBus Enterprise | 1|

| 36| Johnson and Sons | 2|

| 35| Shipping & Co. | 3|

| 6| Alloy LLC | 3|

| 40| FireConsulting | 5|

| 39| News Corp. | 6|

+---------+---------------------+-------+

# Answer

|  |
| --- |
| SELECT |
|  | maintable\_V47CA.GroupID, |
|  | cb\_vendorinformation.CompanyName, |
|  | COUNT(maintable\_V47CA.GroupID) AS `Count` |
|  | FROM |
|  | maintable\_V47CA |
|  | JOIN cb\_vendorinformation ON cb\_vendorinformation.GroupID = maintable\_V47CA.GroupID |
|  | GROUP BY |
|  | maintable\_V47CA.GroupID, |
|  | cb\_vendorinformation.CompanyName |
|  | ORDER BY |
|  | COUNT(maintable\_V47CA.GroupID), |
|  | maintable\_V47CA.GroupID |

# References

[CoderByte-Challenges-Solutions/sql\_vendor\_join.sql at master · Hardanish-Singh/CoderByte-Challenges-Solutions (github.com)](https://github.com/Hardanish-Singh/CoderByte-Challenges-Solutions/blob/master/SQL/sql_vendor_join.sql)

# Source

Coderbyte

# Category

SQL(Programming)

# Tech Area

Structured Query Language(SQL)

# Complexity

Medium

# Question

SQL Vendor Sort

your query should return the vendor information along with the values from the table cb\_vendorinformation.

You should combine the values of the two tables based on the GroupID column.

The final query should consolidate the rows to be grouped by FirstName,

and a Count column should be added at the end that adds up the number of times that person shows up in the table.

The output table should be sorted by the Count column in ascending order and then sorted by CompanyName in alphabetical order.

Your output should look like the

+---------+------------+-----------+------------+--------------+---------------------+--------+

| GroupID | FirstName | LastName | Job | ExternalID | CompanyNane | Count |

+---------+------------+-----------+------------+--------------+---------------------+--------+

| 6 | Larry | Weis | Contractor | 4219 | Alloy LLC | 2 |

+---------+------------+-----------+------------+--------------+---------------------+--------+

| 39 | Mary | Dial | Manager | 1860 | News Corp. | 2 |

+---------+------------+-----------+------------+--------------+---------------------+--------+

| 5 | Dennis | S | Contractor | 24122 | WaterBus Enterprise | 2 |

+---------+------------+-----------+------------+--------------+---------------------+--------+

| 39 | Lisa | Roberts | Manager | 1860 | News Corp. | 3 |

+---------+------------+-----------+------------+--------------+---------------------+--------+

| 35 | Daniel | Knolle | Manager | 39765 | Shipping & Co. | 6 |

+---------+------------+-----------+------------+--------------+---------------------+--------+

| 35 | Arnold | Sully | Manager | 48507 | Shipping & Co. | 6 |

+---------+------------+-----------+------------+--------------+---------------------+--------+

# Answer

|  |
| --- |
| SELECT |
|  | \* |
|  | FROM |
|  | ( |
|  | SELECT |
|  | main\_table.GroupID, |
|  | main\_table.FirstName, |
|  | main\_table.LastName, |
|  | main\_table.Job, |
|  | main\_table.ExternalID, |
|  | cb\_vendorinformation.CompanyName, |
|  | ( SELECT COUNT(\*) FROM maintable\_J22DJ AS sub\_table WHERE main\_table.FirstName = sub\_table.FirstName ) AS Count |
|  | FROM |
|  | maintable\_J22DJ AS main\_table |
|  | JOIN cb\_vendorinformation ON cb\_vendorinformation.GroupID = main\_table.GroupID |
|  | GROUP by FirstName, LastName |
|  | ) AS SubQuery |
|  | ORDER BY SubQuery.COUNT, SubQuery.CompanyName, SubQuery.FirstName DESC |

# References

[CoderByte-Challenges-Solutions/sql\_vendor\_sort.sql at master · Hardanish-Singh/CoderByte-Challenges-Solutions (github.com)](https://github.com/Hardanish-Singh/CoderByte-Challenges-Solutions/blob/master/SQL/sql_vendor_sort.sql)

# Source

Coderbyte

# Category

SQL(Programming)

# Tech Area

Structured Query Language(SQL)

# Complexity

Medium

# Question

SQL COMPANY BOSS

Your table: maintable\_WD9RV

MySQL version: 5.5.56-log

In this MySQL challenge, your query should return all the people who report to either Jenny Richards

or have a NULL value in ReportsTo. The rows should be ordered by Age. Your query should also add a

column at the end with a title of Boss Title where the value is either None or CEO.

+-----------+----------+----------------+---------------+-----+-----------+

| FirstName | LastName | ReportsTo | Position | Age | BossTitle |

+-----------+----------+----------------+---------------+-----+-----------+

| Ashley | Johnson | null | Intern | 25 | None |

+-----------+----------+----------------+---------------+-----+-----------+

| Noah | Fritz | Jenny Richards | Assistant | 30 | CEO |

+-----------+----------+----------------+---------------+-----+-----------+

| David | S | Jenny Richards | Director | 32 | CEO |

+-----------+----------+----------------+---------------+-----+-----------+

| Jenny | Richards | null | CEO | 45 | None |

+-----------+----------+----------------+---------------+-----+-----------+

# Answer

|  |
| --- |
| SELECT |
|  | FirstName, |
|  | LastName, |
|  | ReportsTo, |
|  | Position, |
|  | Age, |
|  | CASE WHEN ReportsTo = 'Jenny Richards' THEN 'CEO' ELSE 'None' END AS BossTitle |
|  | FROM |
|  | maintable\_WD9RV |
|  | WHERE ReportsTo = 'Jenny Richards' OR ReportsTo IS NULL |
|  | ORDER BY Age; |

# References

[CoderByte-Challenges-Solutions/SQL\_Company\_Boss.sql at master · Hardanish-Singh/CoderByte-Challenges-Solutions (github.com)](https://github.com/Hardanish-Singh/CoderByte-Challenges-Solutions/blob/master/SQL/SQL_Company_Boss.sql)

# Source

Coderbyte

# Category

SQL(Programming)

# Tech Area

Structured Query Language(SQL)

# Complexity

Easy

# Question

SQL BASIC SUM

Your table: maintable\_OT6J7

MySQL version: 5.5.56-log

In this MySQL challenge,

your query should return LastName & the sum of Age from your table for all users with a LastName = Smith

The column title of the summed ages should be SumAge. Your output should look like the following table.

+----------+---------+

| Lastname | SumAge |

+----------+---------+

| Smith | 47 |

+----------+---------+

# Answer

|  |
| --- |
| SELECT |
|  | LastName, |
|  | SUM(Age) AS SumAge |
|  | FROM |
|  | maintable\_OT6J7 |
|  | WHERE LastName = 'Smith'; |

# References

[CoderByte-Challenges-Solutions/sql\_basic\_sum.sql at master · Hardanish-Singh/CoderByte-Challenges-Solutions (github.com)](https://github.com/Hardanish-Singh/CoderByte-Challenges-Solutions/blob/master/SQL/sql_basic_sum.sql)

# Source

LeetCode

# Category

SQL(Programming)

# Tech Area

Structured Query Language(SQL)

# Complexity

Easy

# Question

Rank Scores

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| id | int |

| score | decimal |

+-------------+---------+

id is the primary key for this table.

Each row of this table contains the score of a game. Score is a floating point value with two decimal places.

Write an SQL query to rank the scores. The ranking should be calculated according to the following rules:

The scores should be ranked from the highest to the lowest.

If there is a tie between two scores, both should have the same ranking.

After a tie, the next ranking number should be the next consecutive integer value. In other words, there should be no holes between ranks.

Return the result table ordered by score in descending order.

The query result format is in the following example.

# Answer

SELECT Score, @rank := @rank + (@prev <> (@prev := Score)) Rank FROM Scores, (SELECT @rank := 0, @prev := -1) init ORDER BY Score desc

# References

[Official Solution - Rank Scores - LeetCode](https://leetcode.com/problems/rank-scores/solutions/2665416/official-solution/)

# Source

LeedCode

# Category

SQL(Programming)

# Tech Area

Structured Query Language(SQL)

# Complexity

Hard

# Question

Trips and Users

+-------------+----------+

| Column Name | Type |

+-------------+----------+

| id | int |

| client\_id | int |

| driver\_id | int |

| city\_id | int |

| status | enum |

| request\_at | date |

+-------------+----------+

id is the primary key for this table.

The table holds all taxi trips. Each trip has a unique id, while client\_id and driver\_id are foreign keys to the users\_id at the Users table.

Status is an ENUM type of ('completed', 'cancelled\_by\_driver', 'cancelled\_by\_client').

+-------------+----------+

| Column Name | Type |

+-------------+----------+

| users\_id | int |

| banned | enum |

| role | enum |

+-------------+----------+

users\_id is the primary key for this table.

The table holds all users. Each user has a unique users\_id, and role is an ENUM type of ('client', 'driver', 'partner').

banned is an ENUM type of ('Yes', 'No').

The cancellation rate is computed by dividing the number of canceled (by client or driver) requests with unbanned users by the total number of requests with unbanned users on that day.

Write a SQL query to find the cancellation rate of requests with unbanned users (both client and driver must not be banned) each day between "2013-10-01" and "2013-10-03". Round Cancellation Rate to two decimal points.

Return the result table in any order.

The query result format is in the following example.

# Answer

SELECT request\_at Day, ROUND(COUNT(IF(status LIKE 'cancelled%', 1, NULL))/ COUNT(request\_at),2) "Cancellation Rate"

FROM

Trips

WHERE

client\_id NOT IN

(SELECT users\_id FROM Users WHERE banned = 'Yes' AND role = 'client')

AND

driver\_id NOT IN

(SELECT users\_id FROM Users WHERE banned = 'Yes' AND role = 'driver')

AND

request\_at BETWEEN "2013-10-01" AND "2013-10-03"

GROUP BY request\_at;

# References

[Trips and Users - LeetCode](https://leetcode.com/problems/trips-and-users/)

# Source

LeedCode

# Category

SQL(Programming)

# Tech Area

Structured Query Language(SQL)

# Complexity

Hard

# Question

Department top Three Salaries

+--------------+---------+

| Column Name | Type |

+--------------+---------+

| id | int |

| name | varchar |

| salary | int |

| departmentId | int |

+--------------+---------+

id is the primary key column for this table.

departmentId is a foreign key of the ID from the Department table.

Each row of this table indicates the ID, name, and salary of an employee. It also contains the ID of their department.

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| id | int |

| name | varchar |

+-------------+---------+

id is the primary key column for this table.

Each row of this table indicates the ID of a department and its name.

A company's executives are interested in seeing who earns the most money in each of the company's departments. A high earner in a department is an employee who has a salary in the top three unique salaries for that department.

Write an SQL query to find the employees who are high earners in each of the departments.

Return the result table in any order.

The query result format is in the following example.

# Answer

select d.name Department, e.name Employee, e.salary Salary

from employee e inner join department d

on (e.departmentid = d.id)

where (select count(distinct b.Salary) from Employee as b

where b.DepartmentId = e.DepartmentId

and b.Salary > e.Salary) < 3;

# References

[Department Top Three Salaries - LeetCode](https://leetcode.com/problems/department-top-three-salaries/)



QUESTION BANK DOCKET

**Table of Contents**

[1 Source 3](#_Toc253285289)

[2 Category 3](#_Toc965934774)

[3 Tech Area 3](#_Toc613746458)

[4 Complexity 3](#_Toc904143571)

[5 Question 3](#_Toc1426514880)

[6 Answer 4](#_Toc441207452)

[7 References 4](#_Toc1838997797)

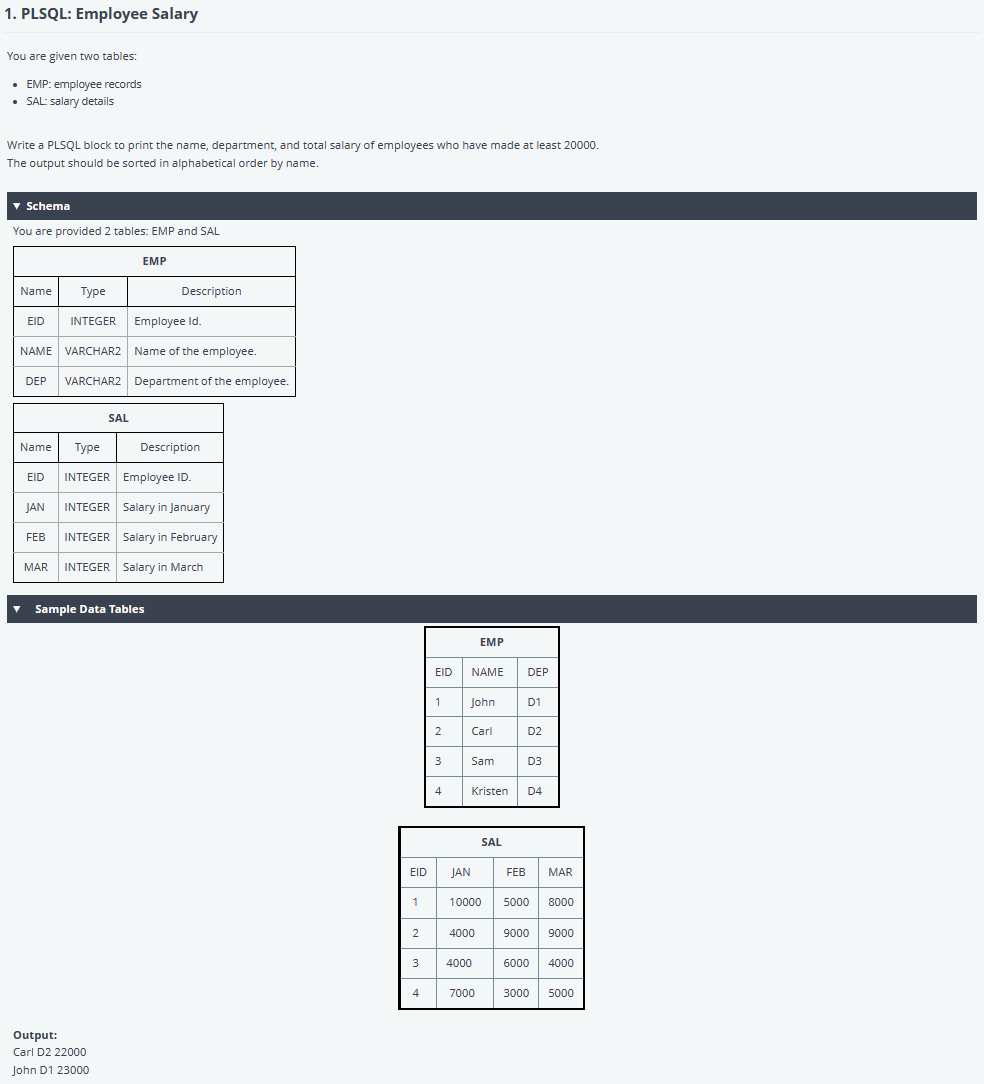
**1). Source:- HackerRank**

**2). Category:-DataBase**

**3). Tech Area : SQL**

**4). Complexity** **: Simple**

**5). Question**



**6). Answer**

**7).References :**https://www.hackerrank.com/work/tests/1512434/questions

**1). Source:- HackerRank**

**2). Category:-DataBase**

**3). Tech Area : SQL**

**4). Complexity : Simple**

**5). Question**



**6). Answer**

**7).References:** https://www.hackerrank.com/x/tests/all/1512434/questions/380501/view

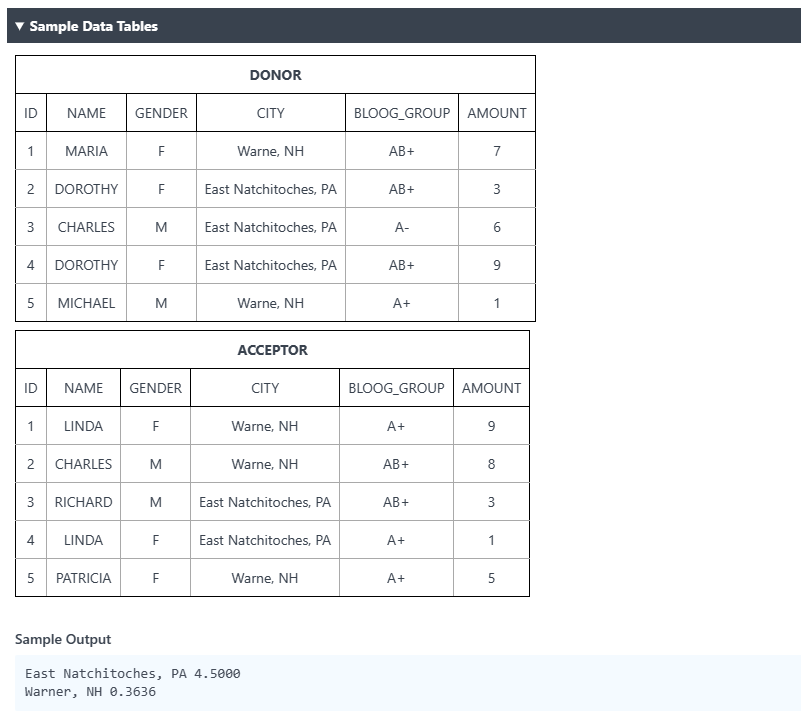
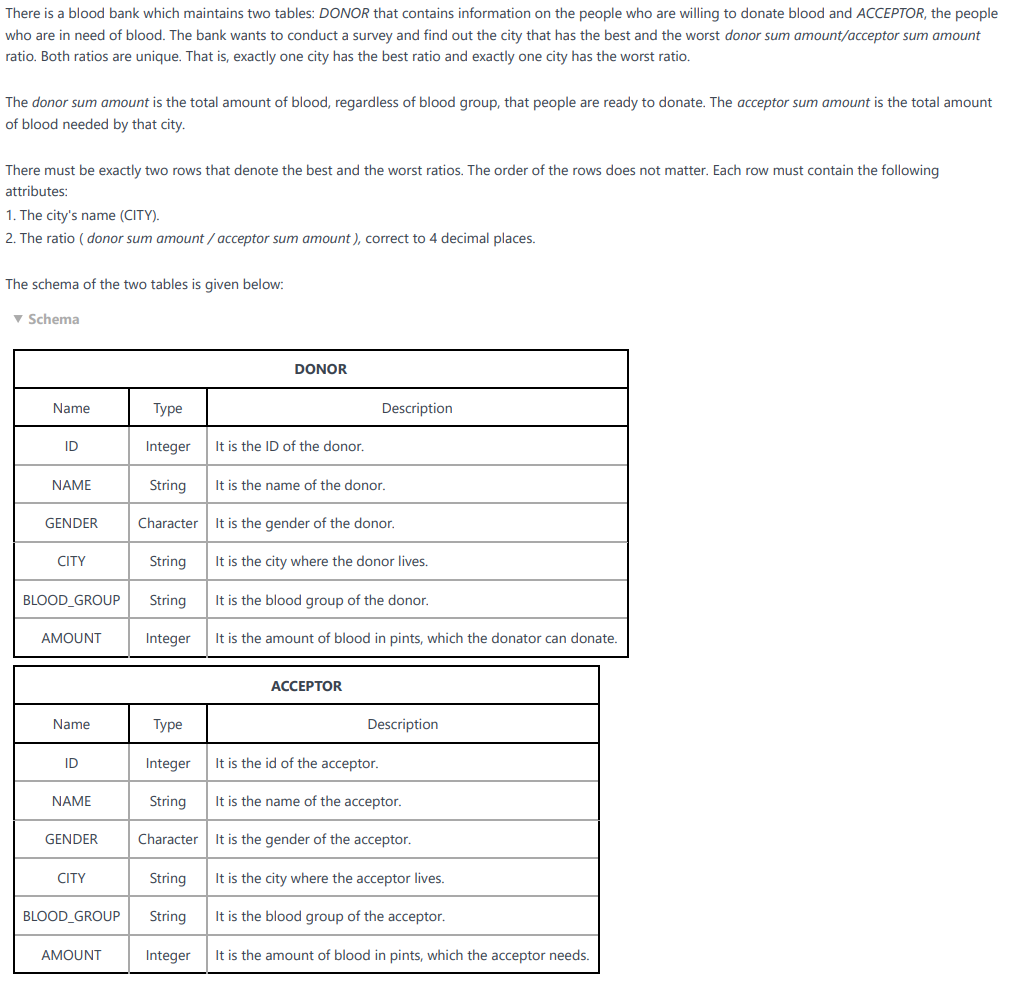
**1). Source:- HackerRank**

**2). Category:-DataBase**

**3). Tech Area : SQL**

**4). Complexity : Intermediate**

**5). Question**



6). Answer

7).References :-https://www.hackerrank.com/x/tests/all/1512434/questions/816565/view

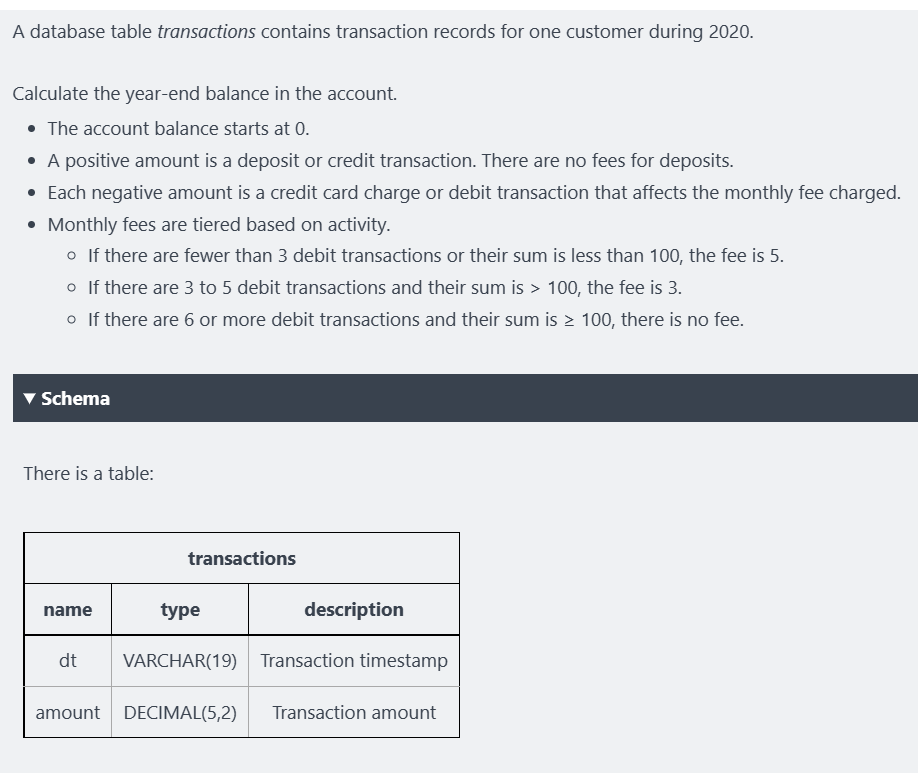
**1). Source:- HackerRank**

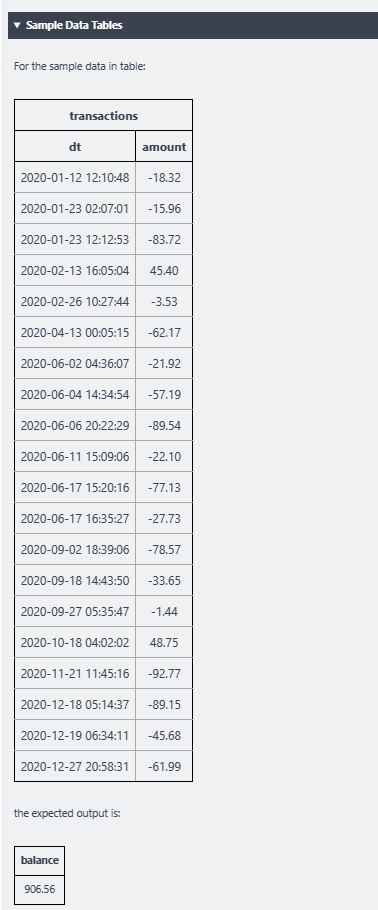
**2). Category:-DataBase**

**3). Tech Area : SQL**

**4). Complexity : Simple**

**5). Question**





**6). Answer**

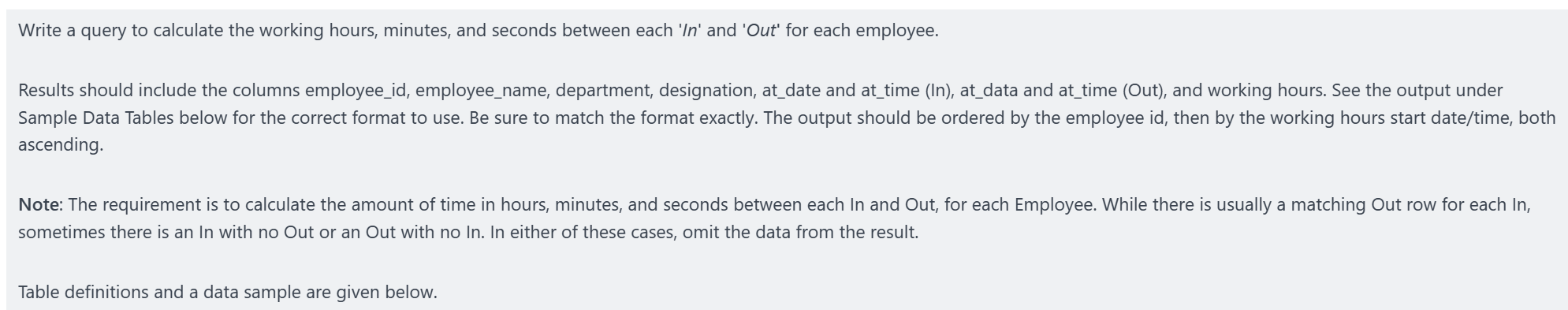
**7).References :** https://www.hackerrank.com/x/tests/all/1512434/questions/1238841/view

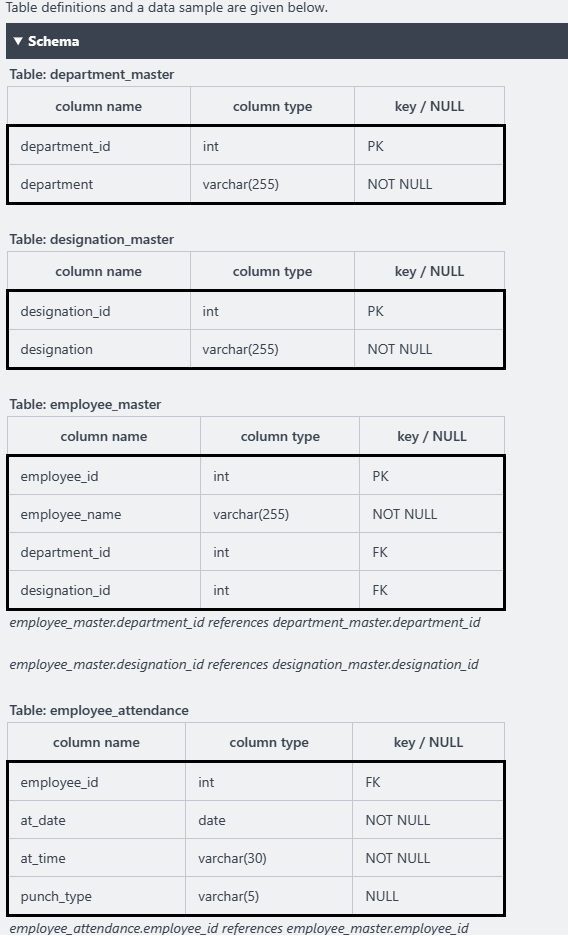
**1). Source:- HackerRank**

**2). Category:-DataBase**

**3). Tech Area : SQL**

**4). Complexity : HARD**

**5). Question**



**6). Answer**

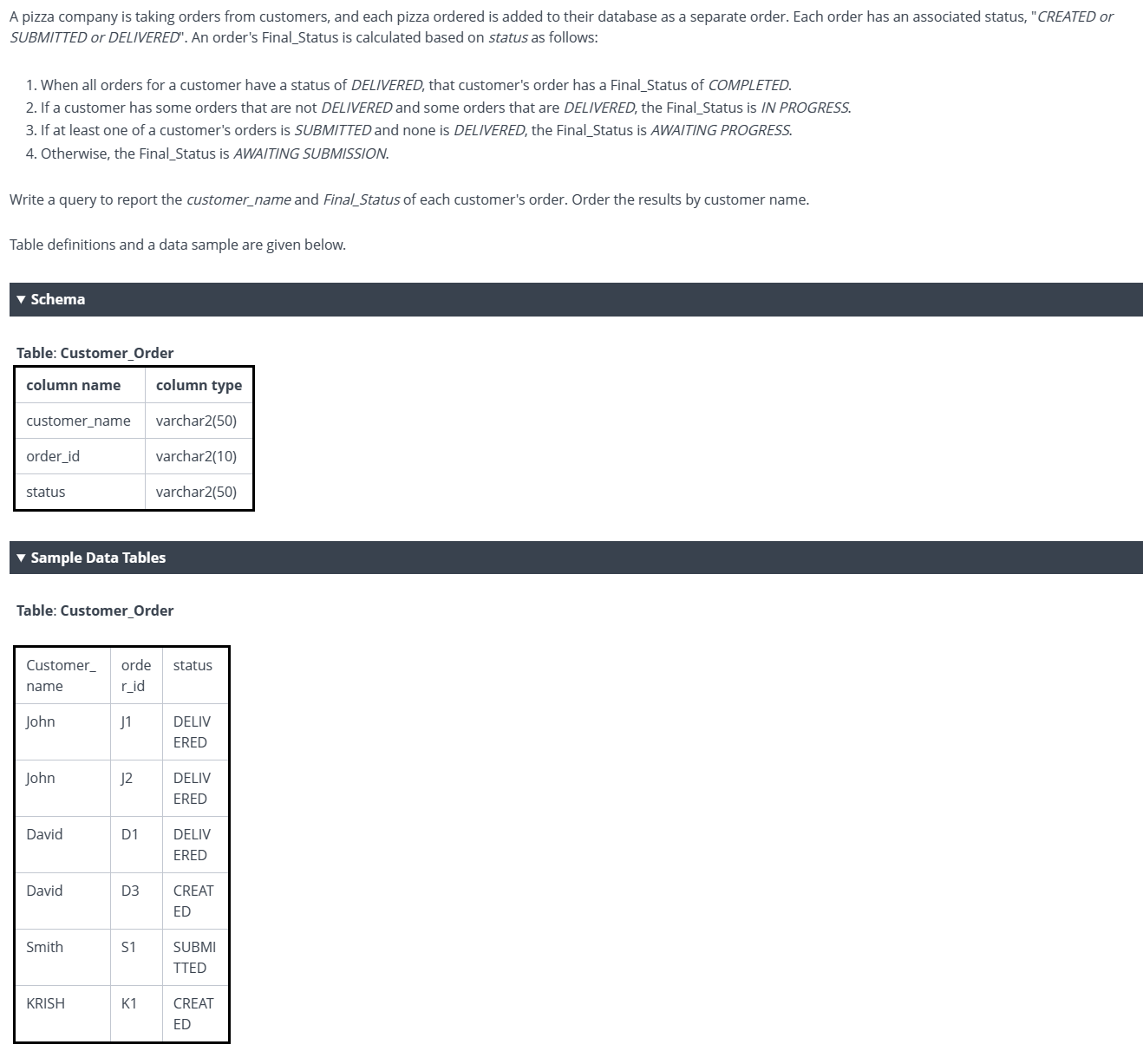
**7).References :** https://www.hackerrank.com/x/tests/all/1512434/questions/1010714/view

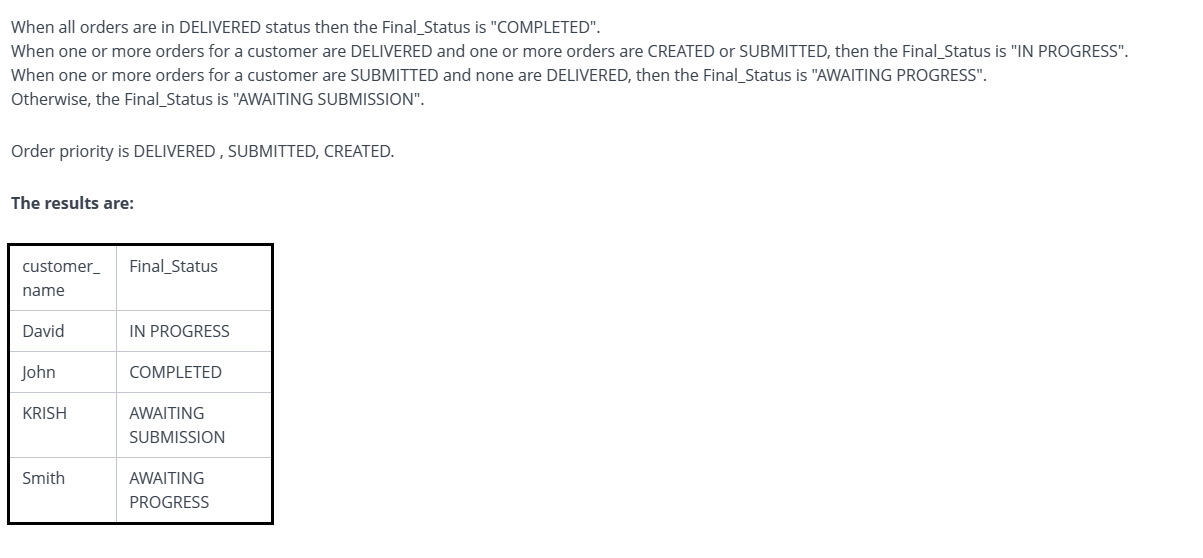
**1). Source:- HackerRank**

**2). Category:-DataBase**

**3). Tech Area : SQL**

**4). Complexity : Hard**

**5). Question**

**6). Answer**

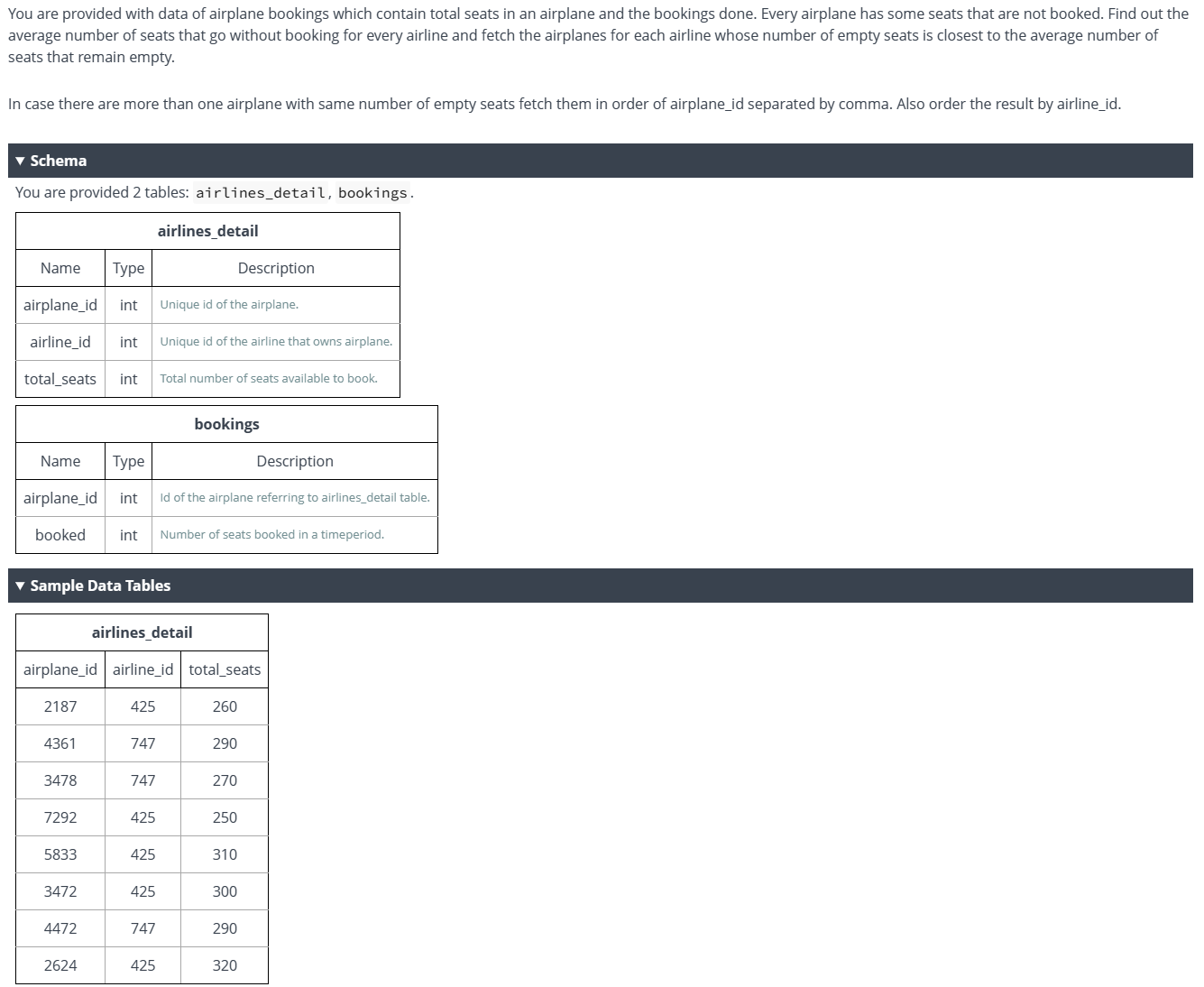
**7). References** : https://www.hackerrank.com/x/tests/all/1512434/questions/1010954/view

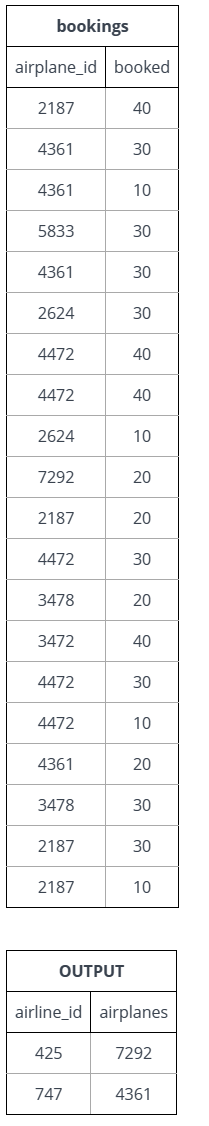
**1). Source:- HackerRank**

**2). Category:-DataBase**

**3). Tech Area : SQL**

**4). Complexity : Hard**

**5). Question**



**6). Answer**

**7).References :** https://www.hackerrank.com/x/tests/all/1512434/questions/1157208/view

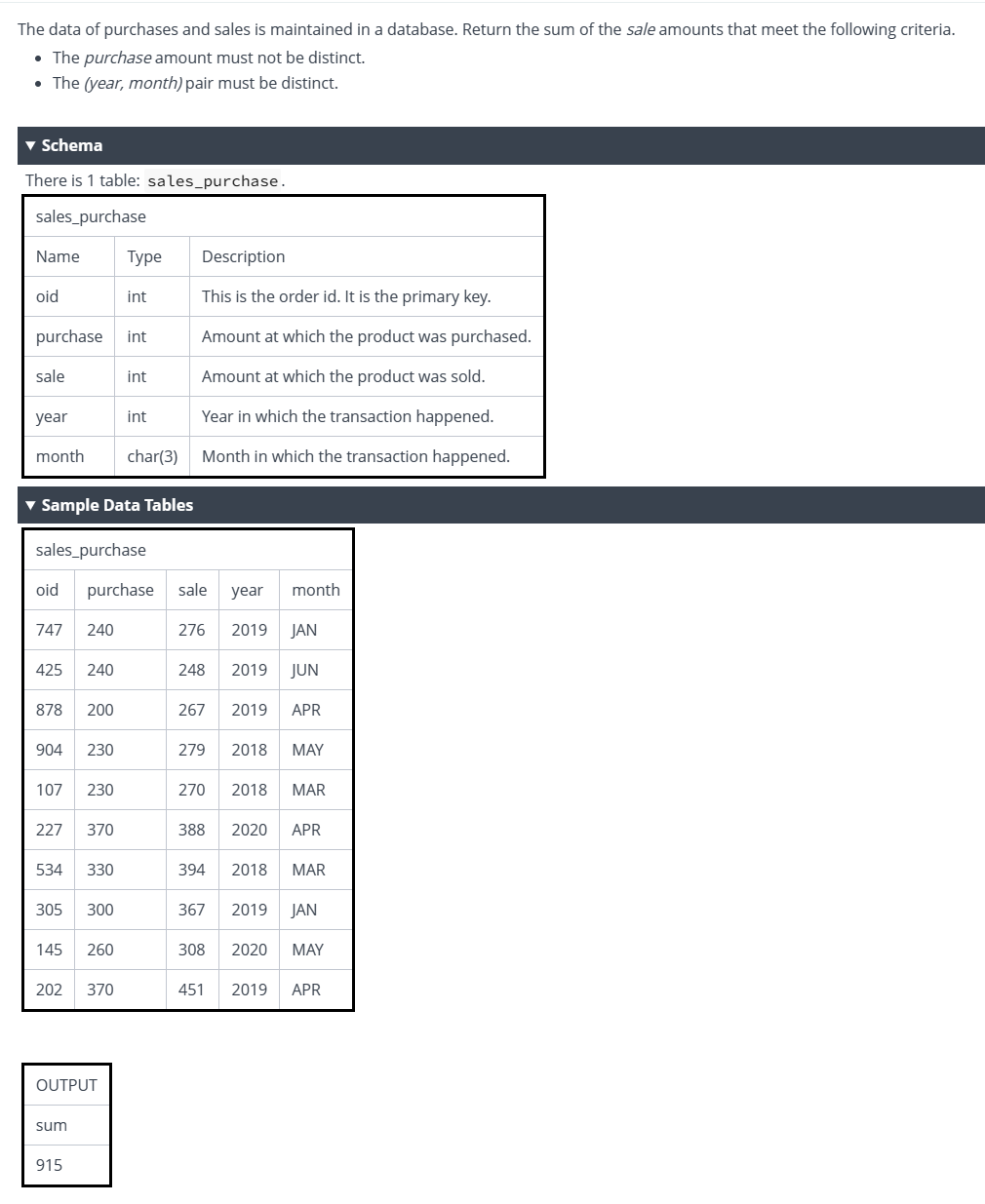
**1). Source:- HackerRank**

**2). Category:-DataBase**

**3). Tech Area : SQL**

**4). Complexity : Hard**

**5). Question**



**6). Answer**

**7).References :** https://www.hackerrank.com/x/tests/all/1512434/questions/1137362/view

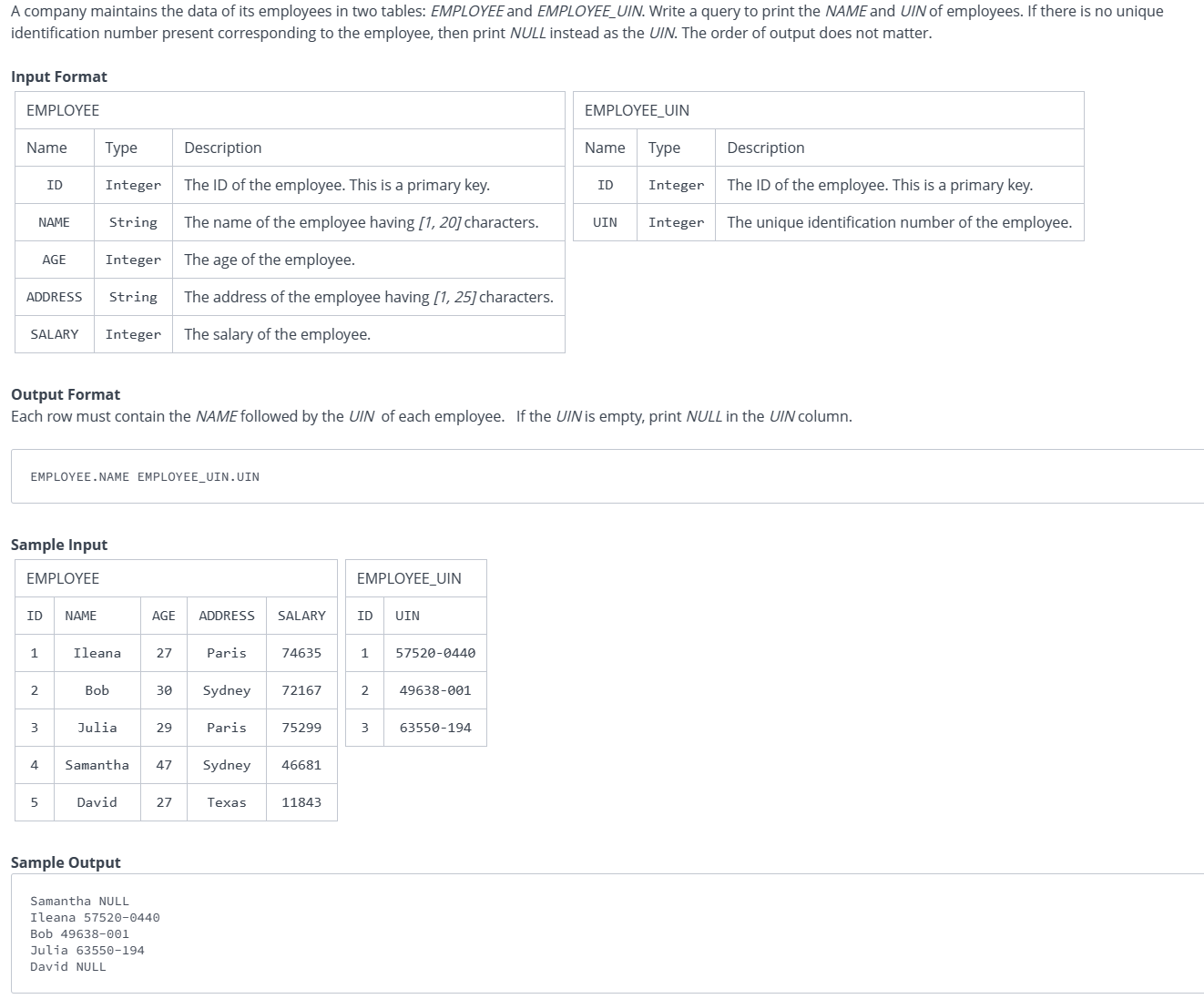
**1). Source:- HackerRank**

**2). Category:-DataBase**

**3). Tech Area : SQL**

**4). Complexity : Intermediate**

**5). Question**



**6). Answer**

**7).References :** https://www.hackerrank.com/x/tests/all/1512434/questions/380499/view

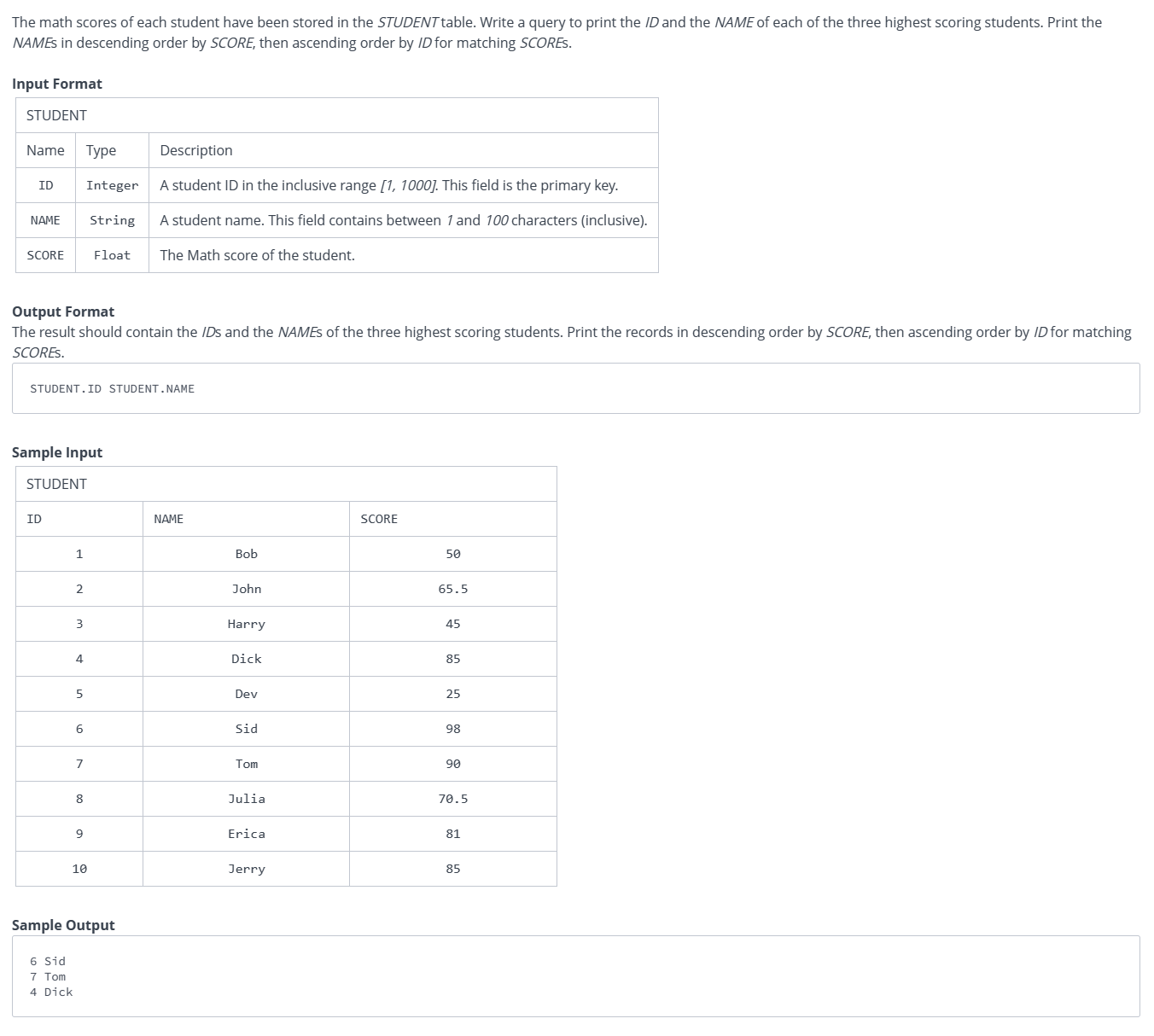
**1). Source:- HackerRank**

**2). Category:-DataBase**

**3). Tech Area : SQL**

**4). Complexity : Simple**

**5). Question**



**6). Answer**

**7).References :** https://www.hackerrank.com/x/tests/all/1512434/questions/380496/view



QUESTION BANK DOCKET

**Table of Contents**

[1 Source 3](#_Toc124585200)

[2 Tech 3](#_Toc124585201)

[3 Focus Area 3](#_Toc124585202)

[4 Complexity 3](#_Toc124585203)

[5 Question 3](#_Toc124585204)

[6 Answer 3](#_Toc124585205)

[7 References 3](#_Toc124585206)

# Source

LEET CODE

# Tech

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Focus Area

SQL

# Complexity

**Simple** / Medium / Complex

# Question

Q.1) Combine Two Tables::

Table: Person

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| person Id | int |

| lastName | varchar |

| firstName | varchar |

+-------------+---------+

Person Id is the primary key column for this table.

This table contains information about the ID of some persons and their first and last names.

Table: Address

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| address Id | int |

| person Id | int |

| city | varchar |

| state | varchar |

+-------------+---------+

Address Id is the primary key column for this table.

Each row of this table contains information about the city and state of one person with ID = Person Id.

Write an SQL query to report the first name, last name, city, and state of each person in the Person table. If the address of a person Id is not present in the Address table, report null instead.

Return the result table in **Any order.**

The query result format is in the following example.

# Answer

SOL :

# Write your MySQL query statement below

select FirstName, LastName, City, State

from Person left join Address

on Person.PersonId = Address.PersonId;

# References

URL : <https://leetcode.com/problems/combine-two-tables/>

# Source

LEET CODE

# Tech

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Focus Area

**SQL**

# Complexity

**Simple** / Medium / Complex

# Question

2 Q : Second Highest Salary

Table: Employee

+-------------+------+

| Column Name | Type |

+-------------+------+

| id | int |

| salary | int |

+-------------+------+

id is the primary key column for this table.

Each row of this table contains information about the salary of an employee.

Write an SQL query to report the second highest salary from the Employee table. If there is no second highest salary, the query should report null.

The query result format is in the following example.

# Answer

# Write your MySQL query statement below

SELECT

    IFNULL(

      (SELECT DISTINCT Salary

       FROM Employee

       ORDER BY Salary DESC

        LIMIT 1 OFFSET 1),

    NULL) AS SecondHighestSalary

# References

URL : https://leetcode.com/problems/second-highest-salary/

# Source

LEET CODE

# Tech

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Focus Area

**SQL**

# Complexity

Simple **/ Medium /** Complex

# Question

Nth Highest Salary

Table: Employee

+-------------+------+

| Column Name | Type |

+-------------+------+

| id | int |

| salary | int |

+-------------+------+

id is the primary key column for this table.

Each row of this table contains information about the salary of an employee.

Write an SQL query to report the nth highest salary from the Employee table. If there is no nth highest salary, the query should report null.

The query result format is in the following example.

# Answer

SOL :

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

SET N = N-1;

  RETURN (

      SELECT DISTINCT(salary) from Employee order by salary DESC

      LIMIT 1 OFFSET N

  );

END

# References

URL : https://leetcode.com/problems/nth-highest-salary

Source

LEEPCODE

# Tech

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Focus Area

**SQL**

# Complexity

Simple **/ Medium** / Complex

# Question

5 Q : Delete Duplicate Emails

URL : https://leetcode.com/problems/delete-duplicate-emails/

Table: Person

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| id | int |

| email | varchar |

+-------------+---------+

id is the primary key column for this table.

Each row of this table contains an email. The emails will not contain uppercase letters.

Write an SQL query to **delete** all the duplicate emails, keeping only one unique email with the smallest id. Note that you are supposed to write a DELETE statement and not a SELECT one.

After running your script, the answer shown is the Person table. The driver will first compile and run your piece of code and then show the Person table. The final order of the Person table **does not matter**.

The query result format is in the following example.

# Answer

SOL :

DELETE p1 FROM Person p1,

    Person p2

WHERE

    p1.Email = p2.Email AND p1.Id > p2.Id

# References

URL : https://leetcode.com/problems/delete-duplicate-emails/

# Source

LEET CODE

# Tech

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Focus Area

**SQL**

# Complexity

Simple / Medium / **Complex**

# Question

6Q : Find Customer Referee

Table: Customer

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| id | int |

| name | varchar |

| referee\_id | int |

+-------------+---------+

id is the primary key column for this table.

Each row of this table indicates the id of a customer, their name, and the id of the customer who referred them.

Write an SQL query to report the names of the customer that are **not referred by** the customer with id = 2.

Return the result table in **any order**.

The query result format is in the following example.

# Answer

SOL :

SELECT name FROM customer WHERE referee\_id <> 2 OR referee\_id IS NULL;

# References

URL : <https://leetcode.com/problems/find-customer-referee/>

# Source

LEET CODE

# Tech

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Focus Area

**SQL**

# Complexity

Simple / Medium / **Complex**

# Question

7 Q : Human Traffic of Stadium

Table: Stadium

+---------------+---------+

| Column Name | Type |

+---------------+---------+

| id | int |

| visit\_date | date |

| people | int |

+---------------+---------+

visit\_date is the primary key for this table.

Each row of this table contains the visit date and visit id to the stadium with the number of people during the visit.

No two rows will have the same visit\_date, and as the id increases, the dates increase as well.

Write an SQL query to display the records with three or more rows with **consecutive** id's, and the number of people is greater than or equal to 100 for each.

Return the result table ordered by visit\_date in **ascending order**.

The query result format is in the following example.

# Answer

select distinct t1.\*

from stadium t1, stadium t2, stadium t3

where t1.people >= 100 and t2.people >= 100 and t3.people >= 100

and

(

    (t1.id - t2.id = 1 and t1.id - t3.id = 2 and t2.id - t3.id =1)  -- t1, t2, t3

    or

    (t2.id - t1.id = 1 and t2.id - t3.id = 2 and t1.id - t3.id =1) -- t2, t1, t3

    or

    (t3.id - t2.id = 1 and t2.id - t1.id =1 and t3.id - t1.id = 2) -- t3, t2, t1

)

order by t1.id

;

# References

URL : <https://leetcode.com/problems/human-traffic-of-stadium/>

# Source

LEET CODE

# Tech

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Focus Area

**SQL**

# Complexity

Simple / Medium / **Complex**

# Question

8 Q : Tree Node

URL : <https://leetcode.com/problems/tree-node/>

Table: Tree

+-------------+------+

| Column Name | Type |

+-------------+------+

| id | int |

| p\_id | int |

+-------------+------+

id is the primary key column for this table.

Each row of this table contains information about the id of a node and the id of its parent node in a tree.

The given structure is always a valid tree.

Each node in the tree can be one of three types:

* **"Leaf"**: if the node is a leaf node.
* **"Root"**: if the node is the root of the tree.
* **"Inner"**: If the node is neither a leaf node nor a root node.

Write an SQL query to report the type of each node in the tree.

Return the result table in **any order**.

The query result format is in the following example.

# Answer

SOL :

SELECT

id, 'Root' AS Type

FROM

tree

WHERE

p\_id IS NULL

UNION

SELECT

id, 'Leaf' AS Type

FROM

tree

WHERE

id NOT IN (SELECT DISTINCT

p\_id

FROM

tree

WHERE

p\_id IS NOT NULL)

AND p\_id IS NOT NULL

UNION

SELECT

id, 'Inner' AS Type

FROM

tree

WHERE

id IN (SELECT DISTINCT

p\_id

FROM

tree

WHERE

p\_id IS NOT NULL)

AND p\_id IS NOT NULL

ORDER BY id;

# References

<https://leetcode.com/problems/tree-node/>

# Source

LEET CODE

# Tech

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Focus Area

**SQL**

# Complexity

Simple / Medium / **Complex**

# Question

9 Q : Human Traffic of Stadium

URL : <https://leetcode.com/problems/human-traffic-of-stadium/>

Table: Stadium

+---------------+---------+

| Column Name | Type |

+---------------+---------+

| id | int |

| visit\_date | date |

| people | int |

+---------------+---------+

visit\_date is the primary key for this table.

Each row of this table contains the visit date and visit id to the stadium with the number of people during the visit.

No two rows will have the same visit\_date, and as the id increases, the dates increase as well.

Write an SQL query to display the records with three or more rows with **consecutive** id's, and the number of people is greater than or equal to 100 for each.

Return the result table ordered by visit\_date in **ascending order**.

The query result format is in the following example.

# Answer

SOL :

select distinct t1.\*

from stadium t1, stadium t2, stadium t3

where t1.people >= 100 and t2.people >= 100 and t3.people >= 100

and

(

    (t1.id - t2.id = 1 and t1.id - t3.id = 2 and t2.id - t3.id =1)  -- t1, t2, t3

    or

    (t2.id - t1.id = 1 and t2.id - t3.id = 2 and t1.id - t3.id =1) -- t2, t1, t3

    or

    (t3.id - t2.id = 1 and t2.id - t1.id =1 and t3.id - t1.id = 2) -- t3, t2, t1

)

order by t1.id

;

# References

URL : <https://leetcode.com/problems/human-traffic-of-stadium/>

# Source

LEET CODE

# Tech

Programming, Java, .NET, **SQL**, Angular

(Mark one of the above as Bold and Underline)

# Focus Area

**SQL**

# Complexity

Simple / Medium **/ Complex**

# Question

10 Q : Sales Analysis

Table: Product

+--------------+---------+

| Column Name | Type |

+--------------+---------+

| product\_id | int |

| product\_name | varchar |

| unit\_price | int |

+--------------+---------+

product\_id is the primary key of this table.

Each row of this table indicates the name and the price of each product.

Table: Sales

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| seller\_id | int |

| product\_id | int |

| buyer\_id | int |

| sale\_date | date |

| quantity | int |

| price | int |

+-------------+---------+

This table has no primary key, it can have repeated rows.

product\_id is a foreign key to the Product table.

Each row of this table contains some information about one sale.

Write an SQL query that reports the **products** that were **only** sold in the first quarter of 2019. That is, between 2019-01-01 and 2019-03-31 inclusive.

Return the result table in **any order**.

The query result format is in the following example.

# Answer

SOL :

select product\_id,product\_name

from product p join sales s

using(product\_id)

group by product\_id

having sum(sale\_date<"2019-01-01")=0

and sum(sale\_date>"2019-03-31")=0

# References

**SQL Coding Questions – GeeksforGreeks**

**Category: SQL Query Programming**

**Tech Area: SQL**

1. Delete duplicate data from table only first data remains constant.

|  |  |  |
| --- | --- | --- |
| Id | Name | Salary |
| 1 | Harpreet | 20000 |
| 2 | Ravi | 30000 |
| 3 | Vinay | 10000 |
| 4 | Ravi | 30000 |
| 5 | Harpreet | 20000 |
| 6 | Vinay | 10000 |
| 7 | Rajeev | 40000 |
| 8 | Vinay | 10000 |
| 9 | Ravi | 30000 |
| 10 | Sanjay | 50000 |

Answers : DELETE M1

From managers M1, managers M2

Where M2.Name = M1.Name AND M1.Id>M2.IdDE

Output:

|  |  |  |
| --- | --- | --- |
| Id | Name | Salary |
| 1 | Harpreet | 20000 |
| 2 | Ravi | 30000 |
| 3 | Vinay | 10000 |
|  |  |  |
| 7 | Rajeev | 40000 |
| 10 | Sanjay | 50000 |

1. Find the Employees who hired in the Last n months.

Finding the Employees who have been hire in the last n months. Here we get desire output by using TIMESTAMPDIFF() MySQL function.

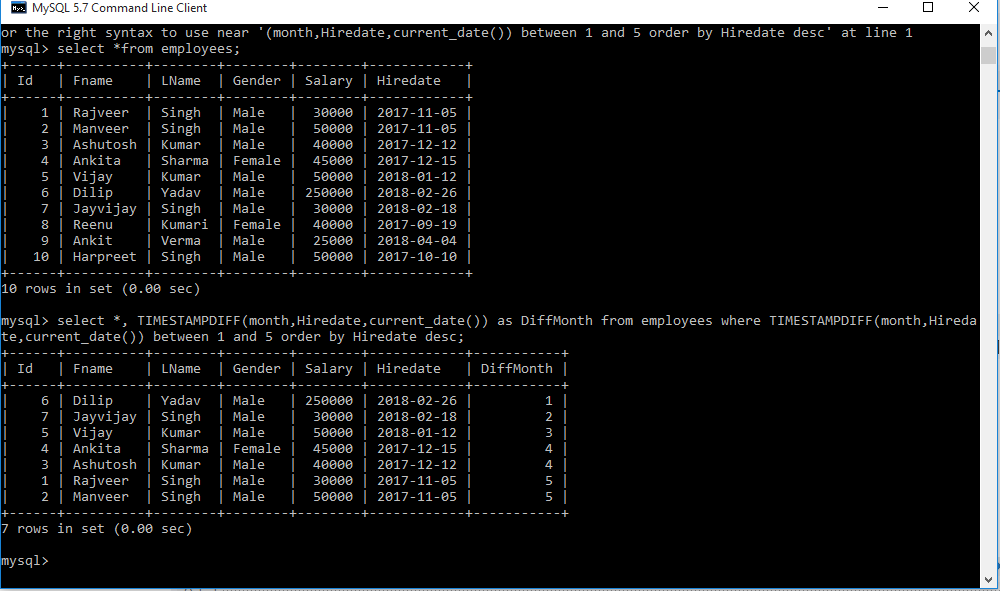
| ID | FName | LName | Gender | Salary | Hiredate |
| --- | --- | --- | --- | --- | --- |
| 1 | Rajveer | Singh | Male | 30000 | 2017/11/05 |
| 2 | Manveer | Singh | Male | 50000 | 2017/11/05 |
| 3 | Ashutosh | Kumar | Male | 40000 | 2017/12/12 |
| 4 | Ankita | Sharma | Female | 45000 | 2017/12/15 |
| 5 | Vijay | Kumar | Male | 50000 | 2018/01/12 |
| 6 | Dilip | Yadav | Male | 25000 | 2018/02/26 |
| 7 | Jayvijay | Singh | Male | 30000 | 2018/02/18 |
| 8 | Reenu | Kumari | Female | 40000 | 2017/09/19 |
| 9 | Ankit | Verma | Male | 25000 | 2018/04/04 |
| 10 | Harpreet | Singh | Male | 50000 | 2017/10/10 |

Answer: Select \*,TIMESTAMPDIFF (month, Hiredate, current\_date()) as DiffMonth

From employees Where TIMESTAMPDIFF (month, Hiredate, current\_date())

Between 1 and 5 Order by Hiredate desc;

**Note –** Here in query 1 and 5 are indicates 1 to n months.which show the Employees who have hired last 1 to 5 months. In this query DiffMonth is a extra column for our understanding which show the Nth months.

Output : 

1. Find the Employees who hired in the Last n days.

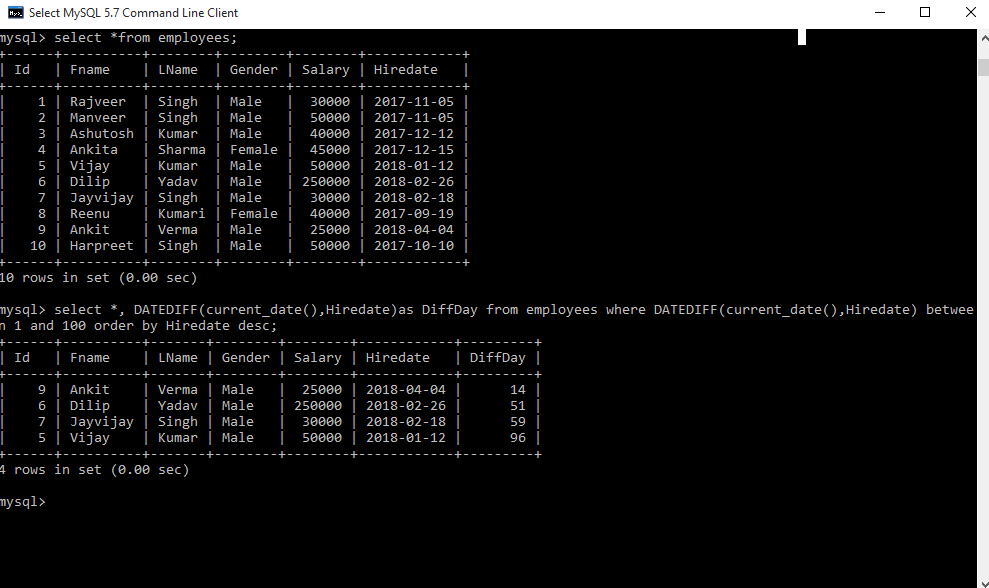
Finding the Employees who have been hire in the last n days. Here we get desire output by using DATEDIFF() MySQL function.

| ID | FName | LName | Gender | Salary | Hiredate |
| --- | --- | --- | --- | --- | --- |
| 1 | Rajveer | Singh | Male | 30000 | 2017/11/05 |
| 2 | Manveer | Singh | Male | 50000 | 2017/11/05 |
| 3 | Ashutosh | Kumar | Male | 40000 | 2017/12/12 |
| 4 | Ankita | Sharma | Female | 45000 | 2017/12/15 |
| 5 | Vijay | Kumar | Male | 50000 | 2018/01/12 |
| 6 | Dilip | Yadav | Male | 25000 | 2018/02/26 |
| 7 | Jayvijay | Singh | Male | 30000 | 2018/02/18 |
| 8 | Reenu | Kumari | Female | 40000 | 2017/09/19 |
| 9 | Ankit | Verma | Male | 25000 | 2018/04/04 |
| 10 | Harpreet | Singh | Male | 50000 | 2017/10/10 |

Answer: Select \*, DATEDIFF (current\_date(), Hiredate) as DiffDay

From Employees Where DATEDIFF (current\_date(), Hiredate) between 1 and 100 order by Hiredate desc;

**Note –** Here in query 1 and 100 are indicates 1 to n days.which show the Employees who have hired last 1 to 100 days. In this query DiffDay is a extra column for our understanding which show the Nth days.



1. Find the Employees who hired in the Last n years.

Finding the Employees who have been hire in the last n years. Here we get desire output by using TIMESTAMPDIFF() mysql function.

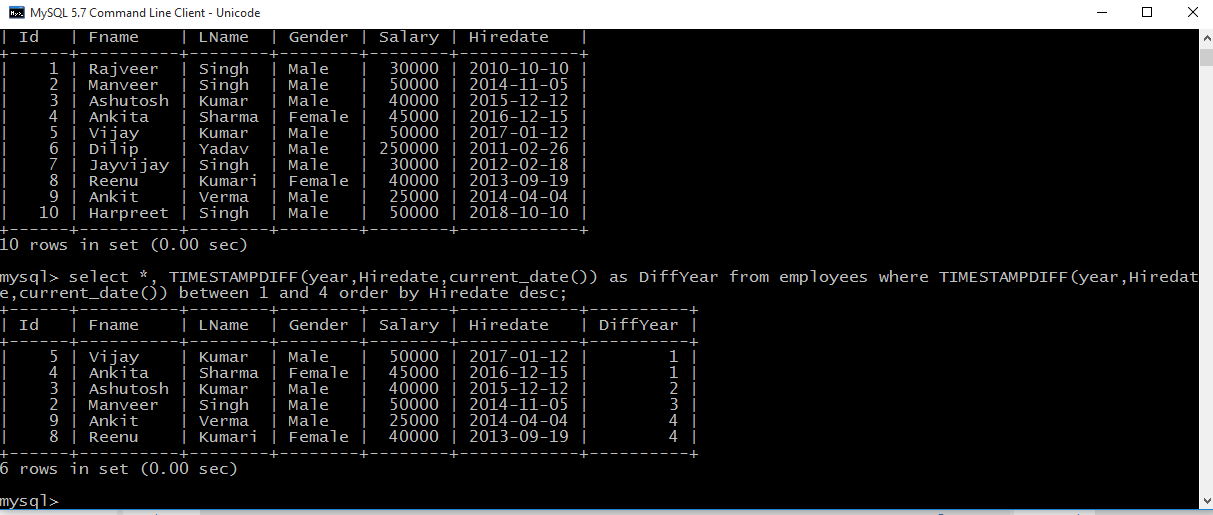
| ID | FName | LName | Gender | Salary | Hiredate |
| --- | --- | --- | --- | --- | --- |
| 1 | Rajveer | Singh | Male | 30000 | 2010/11/05 |
| 2 | Manveer | Singh | Male | 50000 | 2017/11/05 |
| 3 | Ashutosh | Kumar | Male | 40000 | 2015/12/12 |
| 4 | Ankita | Sharma | Female | 45000 | 2016/12/15 |
| 5 | Vijay | Kumar | Male | 50000 | 2017/01/12 |
| 6 | Dilip | Yadav | Male | 25000 | 2011/02/26 |
| 7 | Jayvijay | Singh | Male | 30000 | 2012/02/18 |
| 8 | Reenu | Kumari | Female | 40000 | 2013/09/19 |
| 9 | Ankit | Verma | Male | 25000 | 2017/04/04 |
| 10 | Harpreet | Singh | Male | 50000 | 2017/10/10 |

Answer: Select \*, TIMESTAMPDIFF (year, Hiredate, current\_date()) as DiffYear

From employeesWhere TIMESTAMPDIFF (year, Hiredate, current\_date()) between Hiredate desc;

**Note –** Here in query 1 and 4 are indicates 1 to n years.which show the Employees who have hired last 1 to 4 years. In this query DiffYear is a extra column for our understanding which show the Nth years.

Output:



1. Select all names that start with a given letter.

Here we get desire output by using three different query

| ID | FName | LName | Gender | Salary | Hiredate |
| --- | --- | --- | --- | --- | --- |
| 1 | Rajveer | Singh | Male | 30000 | 2010/11/05 |
| 2 | Manveer | Singh | Male | 50000 | 2017/11/05 |
| 3 | Ashutosh | Kumar | Male | 40000 | 2015/12/12 |
| 4 | Ankita | Sharma | Female | 45000 | 2016/12/15 |
| 5 | Vijay | Kumar | Male | 50000 | 2017/01/12 |
| 6 | Dilip | Yadav | Male | 25000 | 2011/02/26 |
| 7 | Jayvijay | Singh | Male | 30000 | 2012/02/18 |
| 8 | Reenu | Kumari | Female | 40000 | 2013/09/19 |
| 9 | Ankit | Verma | Male | 25000 | 2017/04/04 |
| 10 | Harpreet | Singh | Male | 50000 | 2017/10/10 |

Answer: Select \* From employees Where Fname like 'A%';

Select \*From employees Where left(FName, 1)='A';

Select \*From employees Where substring(FName, 1, 1)='A';

**Note –** Here every query will give same output and the list of Employees who’s FName start with letter A.

6.  
1. Student Table

|  |  |  |
| --- | --- | --- |
| EnrollNo | StudentName | Address |
| 1000 | geek1 | geeksquiz1 |
| 1001 | geek2 | geeksquiz2 |
| 1002 | geek3 | geeksquiz3 |

2. Course ID Enroll

|  |  |
| --- | --- |
| CourseID | EnrollNo |
| 1 | 1000 |
| 2 | 1000 |
| 3 | 1000 |
| 1 | 1002 |
| 2 | 1003 |

join query that shows the names of students enrolled in different courseIDs.

Answer:SELECT StudentCourse.CourseID, Student.StudentName FROM StudentCourse

INNER JOIN Student ON StudentCourse.EnrollNo = Student.EnrollNo ORDER

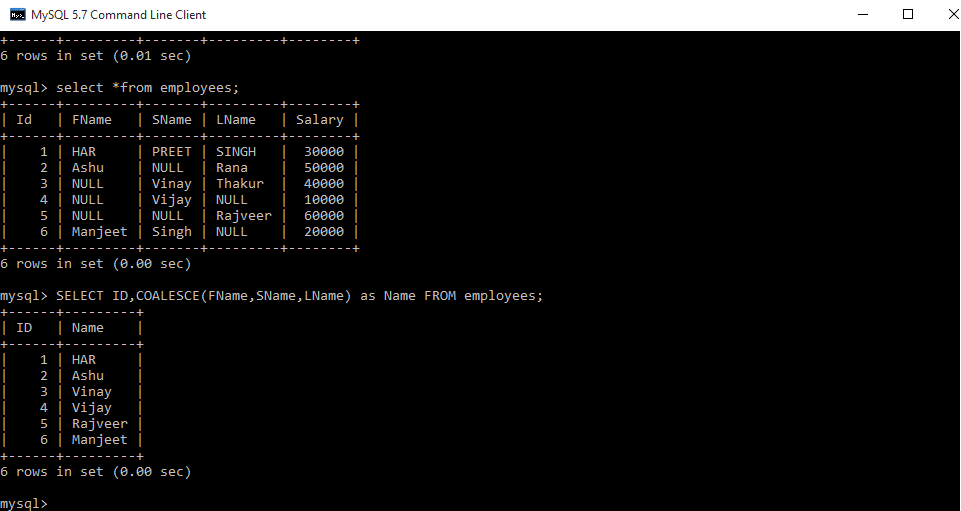
BY StudentCourse.CourseID;  
Output:

|  |  |
| --- | --- |
| CourseID | StudentName |
| 1 | geek1 |
| 1 | geek3 |
| 2 | geek1 |
| 3 | geek1 |

1. **Find the Name of Employees where First Name, Second Name, and Last Name is given in the table. Some Name is missing such as First Name, Second Name and maybe Last Name. Here we will use**[COALESCE()](https://www.geeksforgeeks.org/sql-general-functions-nvl-nvl2-decode-coalesce-nullif-lnnvl-nanvl/)**function which will return first Non Null values.**

| ID | FName | SName | LName | Salary |
| --- | --- | --- | --- | --- |
| 1 | Har | preet | Singh | 30000 |
| 2 | Ashu | NULL | Rana | 50000 |
| 3 | NULL | Vinay | Thakur | 40000 |
| 4 | NULL | Vinay | NULL | 10000 |
| 5 | NULL | NULL | Rajveer | 60000 |
| 6 | Manjeet | Singh | NULL | 60000 |

Answer: SELECT ID, COALESCE(FName, SName, LName) as Name FROM employees;



1. **There is a table where only one row is fully repeated. Write a Query to find the Repeated row**

| **Name** | **Section** |
| --- | --- |
| abc | CS1 |
| bcd | CS2 |
| abc | CS1 |

Answer: SELECT name,section FROM tbl GROUP BY name, sectionHAVING COUNT(\*) >1

1. **Consider the following Employee table. How many rows are there in the result of the following query? ID   salary   DeptName** 1    10000      EC 2    40000      EC 3    30000      CS 4    40000      ME 5    50000      ME 6    60000      ME 7    70000      CS **How many rows are there in the result of the following query?**

Answer: SELECT E.ID FROM  Employee E WHERE  EXISTS  (SELECT E2.salary FROM

Employee E2 WHERE E2.DeptName = 'CS'AND   E.salary > E2.salary)

1. **Write a trigger to update Emp table such that, If an updation is done in Dep table then salary of all employees of that department should be incremented by some amount (updation)** Assuming Table name are Dept and Emp, trigger can be written as follows:   
   Answer: CREATE OR REPLACE TRIGGER update\_trig

AFTER UPDATE ON Dept

FOR EACH ROW

DECLARE

CURSOR emp\_cur IS SELECT \* FROM Emp;

BEGIN

FOR i IN emp\_cur LOOP

IF i.dept\_no = :NEW.dept\_no THEN

DBMS\_OUTPUT.PUT\_LINE(i.emp\_no);  --  for printing those

UPDATE Emp                      -- emp number which are

SET sal = i.sal + 100           -- updated

WHERE emp\_no = i.emp\_no;

END IF;

END LOOP;

END;

10.**Name the Employee who has the third-highest salary using sub queries.**

Answer: SELECT Emp1.Name FROM Employee Emp1 WHERE 2 = (SELECT

COUNT(DISTINCT(Emp2.Salary))FROM Employee Emp2 WHERE Emp2.Salary > Emp1.Salary)