**SQL Questions – Part 2**

1. [**https://app.interviewquery.com/questions/subscription-overlap**](https://app.interviewquery.com/questions/subscription-overlap)

Given a table of product subscriptions with a subscription start date and end date for each user, write a query that returns true or false whether or not each user has a subscription date range that overlaps with any other user.

**Subscriptions table**

| user\_id | start\_date | end\_date |
| --- | --- | --- |
| 1 | 2019-01-01 | 2019-01-31 |
| 2 | 2019-01-15 | 2019-01-17 |
| 3 | 2019-01-29 | 2019-02-04 |
| 4 | 2019-02-05 | 2019-02-10 |

Output

| user\_id | overlap |
| --- | --- |
| 1 | True |
| 2 | True |
| 3 | True |
| 4 | False |

with

overlapping as

(select s1.user\_id first\_user ,s2.user\_id AS second\_user, s1.start\_date,s1.end\_date,s2.start\_date,s2.end\_date

from subscriptions s1

CROSS join subscriptions s2

WHERE s1.user\_id != s2.user\_id

and ((s2.start\_date<s1.end\_date AND s2.start\_date > s1.start\_date) OR (s2.end\_date < s1.end\_date AND s2.end\_date>s1.start\_date))

),

distinct\_users AS

(select first\_user From overlapping

union

select second\_user from overlapping

)

Select user\_id,case when user\_id = first\_user then true else false end AS overlap

FROM subscriptions LEFT JOIN distinct\_users ON user\_id = first\_user

OR

SELECT s1.user\_id, MAX( CASE

WHEN (s1.start\_date > s2.start\_date AND s1.start\_date > s2.end\_date) THEN 0

ELSE 1

END) overlap

FROM **subscriptions**s1, **subscriptions**s2

WHERE s1.user\_id <> s2.user\_id

GROUP BY s1.user\_id

1. [**https://app.interviewquery.com/questions/random-sql-sample**](https://app.interviewquery.com/questions/random-sql-sample)

<https://beginnersbook.com/2018/11/sql-select-random-rows-from-table/>

**big\_table**

| column | type |
| --- | --- |
| id | int |
| name | varchar |

Let's say we have a table with an id and name field. The table holds over 100 million rows and we want to sample a random row in the table without throttling the database.

Write a query to randomly sample a row from this table.

SELECT TOP 1 \*

FROM table\_name

ORDER BY NEWID();

1. [**https://app.interviewquery.com/questions/fradulent-upvotes**](https://app.interviewquery.com/questions/fradulent-upvotes)

**users** table

| **column** | **type** |
| --- | --- |
| id | integer |
| created\_at | datetime |
| username | string |

**comments** table

| **column** | **type** |
| --- | --- |
| id | integer |
| created\_at | datetime |
| post\_id | integer |
| user\_id | integer |
| is\_deleted | boolean |

**comment\_votes** table

| **column** | **type** |
| --- | --- |
| id | integer |
| created\_at | datetime |
| user\_id | integer |
| comment\_id | integer |
| is\_upvote | boolean |

 We're given three tables representing a forum of users and their comments on posts.

We want to figure out if users are creating multiple accounts to upvote their own comments.

1. What kind of metrics could we use to figure this out?

2. Write a query that could display the percentage of users on our forum that would be acting fradulently in this manner.

1. [**https://app.interviewquery.com/questions/upsell-transactions**](https://app.interviewquery.com/questions/upsell-transactions)

transactions` table

| column | type |
| --- | --- |
| id | integer |
| user\_id | integer |
| created\_at | datetime |
| product\_id | integer |
| quantity | integer |

We're given a table of product purchases. Each row in the table represents an individual user product purchase.

Write a query to get the number of customers that were upsold by purchasing additional products.

Note that if the customer purchased two things on the same day that does not count as an upsell as they were purchased within a similar timeframe.

SELECT COUNT(DISTINCT t1.userid)

FROM transactions t1 INNER JOIN transactions t2

ON (t1.userid=t2.userid AND t1.id<>t2.id)

AND cast(t1.createdat as date)<>cast(t2.createdat as date)

AND t1.productid<>t2.product\_id

OR

SELECT COUNT(DISTINCT t.userid)

FROM

(SELECT id, userid, createdat, productid, quantity,

DENSE\_RANK() OVER ( PARTITION by userid, productid ORDER BY cast(t1.createdat as date)) AS rnk

FROM transactions) t WHERE rnk > 1

1. [**https://www.interviewquery.com/questions/first-touch-attribution**](https://www.interviewquery.com/questions/first-touch-attribution)

***attribution`***table

| **column** | **type** |
| --- | --- |
| session\_id | integer |
| channel | string |
| conversion | boolean |

***`user\_sessions`***table

| **column** | **type** |
| --- | --- |
| session\_id | integer |
| created\_at | datetime |
| user\_id | integer |

 The schema above is for a retail online shopping company consisting of two tables, attribution and user\_sessions.

* The attribution table logs a session visit for each row.
* If conversion is true, then the user converted to buying on that session.
* The channel column represents which advertising platform the user was attributed to for that specific session.
* Lastly the `user\_sessions` table maps many to one session visits back to one user.

First touch attribution is defined as the channel to which the converted user was associated with when they first discovered the website.

Calculate the first touch attribution for each user\_id that converted.

Example output:

| user\_id | channel |
| --- | --- |
| 123 | facebook |
| 145 | google |
| 153 | facebook |
| 172 | organic |
| 173 | email |

SELECT a.user\_id, a.channel FROM

(SELECT us.user\_id, att.channel,

RANK() OVER (PARTITION BY us.user\_id ORDER BY created\_at ASC) as rnk

FROM attribution att

LEFT JOIN user\_sessions us

ON att.session\_id = us.session\_id

WHERE att.conversion = True) a

WHERE rnk = 1

<https://stackoverflow.com/questions/64627144/does-where-clause-run-first-and-then-the-analytical-function-or-how-does-the-exe/64627384#64627384>

1. **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 UPPER(SUBSTRING(city,1,1)) NOT IN ('A','E','I','O','U');

1. **Query the list of *CITY* names ending with vowels (a, e, i, o, u) from** STATION**. Your result *cannot* contain duplicates.**

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

1. **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')

1. **Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.**

SELECT COUNT(\*) - COUNT(DISTINCT(CITY)) FROM STATION;

1. **Write a query identifying the type of each record in the TRIANGLES table using its three side lengths.**

**SQL Server:**

SELECT

CASE

WHEN A + B <= C OR A + C <= B OR B + C <= A THEN 'Not A Triangle'

WHEN A = B AND B = C THEN 'Equilateral'

WHEN A = B OR A = C OR B = C THEN 'Isosceles'

ELSE 'Scalene'

END

FROM TRIANGLES;

1. **Query the greatest value of the Northern Latitudes (LAT\_N) from STATION that is less than 137.2345. Truncate your answer to 4 decimal places.**

SELECT ROUND(MAX(LAT\_N), 4) FROM STATION WHERE LAT\_N < 137.2345;

1. **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)

order by city

1. **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 TOP 1 CITY, LEN(CITY) FROM Station ORDER BY LEN(CITY),CITY;

SELECT TOP 1 CITY, LEN(CITY) FROM Station ORDER BY LEN(CITY) DESC,CITY;

1. **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 substring(name,len(name)-2, 3) , id asc;

OR

SELECT name FROM Students WHERE marks > 75 ORDER BY RIGHT(name, 3), ID ASC;

1. **Generate the following two result sets:**
2. 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.
3. Query the number of ocurrences of each occupation in OCCUPATIONS. Sort the occurrences in ascending order, and output them in the following format:

There are a total of [occupation\_count] [occupation]s. where [occupation\_count] is the number of occurrences of an occupation in OCCUPATIONS and [occupation] is the lowercase occupation name. If more than one Occupation has the same [occupation\_count], they should be ordered alphabetically.

Note: There will be at least two entries in the table for each type of occupation

select concat(Name,'(',Substring(Occupation,1,1),')') as Name

from occupations

Order by Name;

select concat('There are a total of ',count(occupation),' ',lower(occupation),'s. ') as total

from occupations

group by occupation

order by count(occupation), occupation;

**OR**

**With case statement if occupations are known**

SELECT concat(NAME, CASE

WHEN occupation = "Doctor" THEN "(D)"

WHEN occupation = "Professor"

THEN "(P)" WHEN occupation = "Singer"

THEN "(S)" WHEN occupation = "Actor" THEN "(A)"

END )

FROM OCCUPATIONS

ORDER BY NAME;

select concat('There are a total of ',count(occupation),' ',lower(occupation),'s. ') as total

from occupations

group by occupation

order by count(occupation), occupation;

1. [**Pivot**](https://en.wikipedia.org/wiki/Pivot_table) **the *Occupation* column in** OCCUPATIONS **so that each *Name* is sorted alphabetically and displayed underneath its corresponding *Occupation*. The output column headers should be *Doctor*, *Professor*, *Singer*, and *Actor*, respectively.**

Note: **Print** NULL **when there are no more names corresponding to an occupation.**

<https://www.hackerrank.com/challenges/occupations/problem>

1. **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.**

<https://www.hackerrank.com/challenges/binary-search-tree-1/problem>

SELECT N, IF(P IS NULL,'Root',

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

'Leaf'))

FROM BST AS B ORDER BY N;

1. **https://www.hackerrank.com/challenges/print-prime-numbers/forum**
2. **https://www.hackerrank.com/challenges/the-company/problem**

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 company c

inner join employee e on e.company\_code = c.company\_code

group by c.company\_code,c.founder

order by c.company\_code;

1. **We define an employee's *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.**

[**https://www.hackerrank.com/challenges/earnings-of-employees/problem**](https://www.hackerrank.com/challenges/earnings-of-employees/problem)

**ORACLE**

SELECT \* FROM

(SELECT MAX(MONTHS\*SALARY), COUNT(EMPLOYEE\_ID)

FROM EMPLOYEE e GROUP BY MONTHS\*SALARY ORDER BY MONTHS\*SALARY DESC) emp

WHERE ROWNUM = 1;

**SQL SERVER**

SELECT TOP 1 mx, cnt FROM

(SELECT MAX(MONTHS\*SALARY) as mx, COUNT(EMPLOYEE\_ID) as cnt

FROM EMPLOYEE e

GROUP BY MONTHS\*SALARY ) emp

ORDER BY mx DESC

**Analytical Function solution:**

SELECT max\_sal, count(max\_sal\_count)FROM

(SELECT MONTHS\*SALARY AS max\_sal,

DENSE\_RANK() OVER (ORDER BY MONTHS\*SALARY DESC) AS max\_sal\_count

FROM EMPLOYEE) emp

WHERE max\_sal\_count = 1

GROUP BY max\_sal;

1. [**https://www.hackerrank.com/challenges/the-blunder/problem**](https://www.hackerrank.com/challenges/the-blunder/problem)

SELECT CAST(CEILING((AVG(CAST(Salary AS Float)) - AVG(CAST(REPLACE(Salary, 0, '')AS Float)))) AS INT)

FROM EMPLOYEES;

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=(SELECT MIN(LAT\_N)

FROM Station

WHERE LAT\_N>38.7780);

1. [**https://www.hackerrank.com/challenges/weather-observation-station-20/problem**](https://www.hackerrank.com/challenges/weather-observation-station-20/problem)

**PERCENTILE\_DISC FUNCION :**

<https://docs.microsoft.com/en-us/sql/t-sql/functions/percentile-disc-transact-sql?view=sql-server-ver15>

PERCENTILE\_DISC ( numeric\_literal ) WITHIN GROUP ( ORDER BY order\_by\_expression [ ASC | DESC ] )

OVER ( [ <partition\_by\_clause> ] )

SELECT DISTINCT CAST( PERCENTILE\_DISC(0.5) WITHIN GROUP (ORDER BY LAT\_N) OVER() AS DECIMAL(10,4) )

FROM STATION

1. **https://www.hackerrank.com/challenges/the-report/problem**

SELECT CASE WHEN GRADE<8 THEN NULL

ELSE Name

END, Grade, Marks

FROM Students JOIN Grades

ON Students.Marks BETWEEN Grades.MIN\_MARK AND Grades.MAX\_MARK

ORDER BY Grade desc, Name

1. **https://www.hackerrank.com/challenges/harry-potter-and-wands/problem**

SELECT id, age, coins\_needed, power

FROM

(

SELECT W.id, WP.age, W.coins\_needed, W.power,

ROW\_NUMBER() OVER

(

PARTITION BY W.code,W.power

ORDER BY W.coins\_needed, W.power DESC

) AS RowNumber

FROM Wands W WITH (NOLOCK)

INNER JOIN Wands\_Property WP WITH (NOLOCK) ON W.code = WP.code

WHERE WP.is\_evil = 0

)

AS Wand\_Data

WHERE RowNumber = 1

ORDER BY power DESC, age DESC

OR

SELECT w.id, p.age, w.coins\_needed, w.power

FROM Wands w JOIN Wands\_Property p

ON (w.code = p.code)

where p.is\_evil = 0

AND w.coins\_needed = (SELECT MIN(coins\_needed) FROM Wands w1 JOIN Wands\_Property p1

ON (w1.code = p1.code)

WHERE w1.power = w.power

AND p1.age = p.age)

ORDER BY w.power desc, p.age DESC

1. [**https://www.hackerrank.com/challenges/challenges/problem**](https://www.hackerrank.com/challenges/challenges/problem)

SELECT c.hacker\_id, h.name ,count(c.hacker\_id) c\_count

FROM Hackers h

inner join Challenges c on c.hacker\_id = h.hacker\_id

GROUP BY c.hacker\_id, h.name

HAVING

count(c.hacker\_id) = (SELECT MAX(temp1.cnt)

FROM (SELECT COUNT(hacker\_id) cnt

FROM Challenges

GROUP BY hacker\_id) temp1)

OR count(c.hacker\_id) IN (SELECT t.cnt

FROM ( SELECT COUNT(\*) cnt

FROM challenges

GROUP BY hacker\_id) t

GROUP BY t.cnt

HAVING COUNT(t.cnt) = 1)

order by c\_count DESC, c.hacker\_id;

1. [**https://www.interviewquery.com/questions/empty-neighborhoods**](https://www.interviewquery.com/questions/empty-neighborhoods)

SELECT \* FROM neighborhoods n LEFT JOIN users u

ON n.id = u.id

WHERE u.id IS NULL;

1. **https://www.interviewquery.com/questions/employee-project-budgets**

**projects**

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

+->| **id** | **int** |

| | title | **varchar** |

| | start\_date | **date** |

| | end\_date | **date** |

| | budget | **int** |

| +---------------+---------+

|

| **employees\_projects**

| +---------------+---------+

+--| project\_id | int |

| employee\_id | **int** |

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

Write a query to get the top five most expensive projects by budget to employee count ratio?

SELECT budget, COUNT(employee\_id) AS employeeCount, budget/COUNT(employee\_id) AS budgetRatio

FROM (

SELECT employee\_project.\*, projects .\* ,

RANK() OVER (ORDER BY budget ) as rnk

FROM employee\_project INNER JOIN projects ON id = project\_id

) r

WHERE rnk<=5

GROUP BY id

1. [**https://www.interviewquery.com/questions/closest-sat-scores**](https://www.interviewquery.com/questions/closest-sat-scores)

scores table

| column | type |
| --- | --- |
| id | integer |
| student | varchar |
| score | integer |

Given a table of students and their SAT test scores, write a query to return the two students with the closest test scores with the score difference. If there are multiple students with the same minimum score difference, select the student name combination that is higher in the alphabet.

Input

| id | student | score |
| --- | --- | --- |
| 1 | Jack | 1700 |
| 2 | Alice | 2010 |
| 3 | Miles | 2200 |
| 4 | Scott | 2100 |

Output

| one\_student | other\_student | score\_diff |
| --- | --- | --- |
| Alice | Scott | 90 |

SELECT s1.student, s2.student FROM

( SELECT s1.student, s2.student, DENSE\_RANK() OVER ( ORDER BY ABS(s1.score – s2.score), s1.student, s2.student) AS rnk FROM student s1 INNER JOIN student s2 WHERE s1.id<s2.id ) scoreComp

WHERE rnk = 1

OR

SELECT \* FROM

(SELECT student, LEAD(student, 1) OVER (ORDER BY score, student) AS nextStudent, score - LEAD(score, 1) OVER (ORDER BY score, student) AS scoreDiff FROM scores ORDER BY scoreDiff ) base

WHERE scorediff >=0

LIMIT 1

1. [**https://www.interviewquery.com/questions/third-purchase**](https://www.interviewquery.com/questions/third-purchase)

transactions` table

| column | type |
| --- | --- |
| id | integer |
| user\_id | integer |
| created\_at | datetime |
| product\_id | integer |
| quantity | integer |

Given the revenue transactions table above, write a query that finds the third purchase of every user.

SELECT id, user\_id, created\_at, product\_id, quantity FROM (

SELECT \*, RANK() OVER (PARTITION BY user\_id, ORDER BY created\_at) AS rnk\_user

FROM transactions) user

WHERE rnk\_user = 3