sql_script_1

March 2, 2022

0.0.1 614. Second Degree Follower(Median)

Table: Follow

+		+-		+
	Column Name		V -	١
+-		-+-		+
-	followee	-	varchar	-
-	follower		varchar	1
+-		-+-		-+

(followee, follower) is the primary key column for this table. Each row of this table indicates that the user follower follows the user followee on a social network. There will not be a user following themself.

A second-degree follower is a user who:

follows at least one user, and is followed by at least one user. Write an SQL query to report the second-degree users and the number of their followers.

Return the result table ordered by follower in alphabetical order.

The query result format is in the following example.

Example 1:

Input: Follow table:

+		-+-		+
1	followee	ı	follower	
+-		-+-		+
	Alice	1	Bob	
	Bob		Cena	
	Bob		Donald	
	Donald		Edward	
+-		-+-		+

Output:

+-		+-		+
1	follower	1	num	
+-		+-		+
Ι	Bob	ı	2	ı

```
| Donald | 1 |
```

Explanation: User Bob has 2 followers. Bob is a second-degree follower because he follows Alice, so we include him in the result table. User Donald has 1 follower. Donald is a second-degree follower because he follows Bob, so we include him in the result table. User Alice has 1 follower. Alice is not a second-degree follower because she does not follow anyone, so we don not include her in the result table.

Solution

```
select
   followee as follower, count(1) as num
from
   Follow
where
   followee in (select follower from Follow)
group by followee
order by followee
```

0.0.2 615. Average Salary: Departments VS Company(Hard)

Table: Salary

id is the primary key column for this table. Each row of this table indicates the salary of an employee in one month. employee_id is a foreign key from the Employee table.

Table: Employee

employee_id is the primary key column for this table. Each row of this table indicates the department of an employee.

Write an SQL query to report the comparison result (higher/lower/same) of the average salary of employees in a department to the company's average salary.

Return the result table in any order.

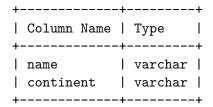
The query result format is in the following example.

Solution

```
select
    SUBSTR(pay date, 1, 7) as pay month
    , department_id
    , case when avg sum depy>avg sum then 'higher'
        when avg_sum_depy=avg_sum then 'same'
        else 'lower' end as comparison
from
    select
        a.pay_date
        , b.department_id
        , avg(a.amount) over (partition by a.pay_date) as avg_sum
        , avg(a.amount) over (partition by a.pay_date, b.department_id) as avg_sum_depy
    from
        Salary a
    join
        Employee b
    on
        a.employee id = b.employee id
group by pay_month, department_id, comparison
order by pay_month DESC
```

0.0.3 618. Students Report By Geography(Hard)

Table: Student



There is no primary key for this table. It may contain duplicate rows. Each row of this table indicates the name of a student and the continent they came from.

A school has students from Asia, Europe, and America.

Write an SQL query to pivot the continent column in the Student table so that each name is sorted alphabetically and displayed underneath its corresponding continent. The output headers should be America, Asia, and Europe, respectively.

The test cases are generated so that the student number from America is not less than either Asia or Europe.

The query result format is in the following example.

Example 1:

Input: Student table:

+-		-+-		-+
1	name	1	00110110110	1
+		+-		+
	Jane		America	
	Pascal	1	Europe	
	Xi	1	Asia	
	Jack	1	America	1
+-		-+-		-+

Output:

+-		+-		+-		+
	America		Asia	١	Europe	1
+-		+-		+-		+
1	Jack		Xi		Pascal	1
1	Jane		null		null	
+-		+-		+-		+

Follow up: If it is unknown which continent has the most students, could you write a query to generate the student report?

Solution: This one is hard, what is group by, why should you add "MAX" after group

0.0.4 1045. Customers Who Bought All Products(Median)

Table: Customer

group by currank

There is no primary key for this table. It may contain duplicates. product_key is a foreign key to Product table.

Table: Product

İ	Column N	ame		+
	product_		int	⊤ +

product_key is the primary key column for this table.

Write an SQL query to report the customer ids from the Customer table that bought all the products in the Product table.

Return the result table in any order.

The query result format is in the following example.

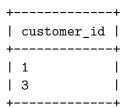
Example 1:

Input: Customer table:

+	++
customer_id	product_key
+	++
1	5
2	6
3	5
3	6
1	6
+	++

Product table:

Output:



Explanation: The customers who bought all the products (5 and 6) are customers with IDs 1 and 3.

Solution:

```
select
    customer_id
from
(
    select * from
        Customer c
    where product_key in (select product_key from Product)
    group by customer_id, product_key
) cc
group by customer_id
having count(1) = (select count(1) from Product)
```

0.0.5 1070. Product Sales Analysis III(Median)

Table: Sales

(sale_id, year) is the primary key of this table. product_id is a foreign key to Product table. Each row of this table shows a sale on the product product_id in a certain year. Note that the price is per unit.

Table: Product

product_id is the primary key of this table. Each row of this table indicates the product name of each product.

Write an SQL query that selects the product id, year, quantity, and price for the first year of every product sold.

Return the resulting table in any order.

The query result format is in the following example.

Example 1:

Input: Sales table:

sale_id	+ product_id +	year	quantity	price
1 2	100	2008 2009 2011	10 12	5000 5000 9000

Product table:

Output:

product_id	first_year	•		•		-+ -+
100 200	2008		10 15	•	5000 9000	 -

Solution:

```
select product_id, year as first_year, quantity, price
from
(
select *, rank() over(partition by product_id order by year) as rn
from Sales
) a
where rn = 1
```

0.0.6 1045. Customers Who Bought All Products(Median)

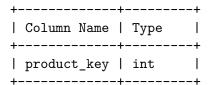
Table: Customer

```
+-----+
| Column Name | Type |
+-----+
| customer_id | int |
| product_key | int |
```

There is no primary key for this table. It may contain duplicates.

product_key is a foreign key to Product table.

Table: Product



product_key is the primary key column for this table. Write an SQL query to report the customer ids from the Customer table that bought all the products in the Product table.

Return the result table in any order.

The query result format is in the following example.

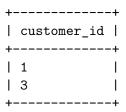
Example 1:

Input: Customer table:

+	++
customer_id	product_key
+	++
1	5
2	6
3	5
3	6
1 1	l 6 I
+	+

Product table:

Output:



Explanation: The customers who bought all the products (5 and 6) are customers with IDs 1 and 3.

Solution:

```
select
    customer_id
from
(
    select * from
        Customer c
    where product_key in (select product_key from Product)
    group by customer_id, product_key
) cc
group by customer_id
having count(1) = (select count(1) from Product)
```

0.0.7 1097. Game Play Analysis V(Hard)

+	-+-		L
Column Name	 	Туре	 -
•	'	'	'
player_id		int	
device_id	Ι	int	
event date	i	date	ı
event_date	ı	uate	
games_played	1	int	
+	-+-		۲

(player_id, event_date) is the primary key of this table. This table shows the activity of players of some games. Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on someday using some device.

The install date of a player is the first login day of that player.

We define day one retention of some date x to be the number of players whose install date is x and they logged back in on the day right after x, divided by the number of players whose install date is x, rounded to 2 decimal places.

Write an SQL query to report for each install date, the number of players that installed the game on that day, and the day one retention.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input: Activity table:

+		+-		+-		-+-		+
			_		_		games_played	1
+	 1			Ċ	2016-03-01	Ċ	 5	-+
i	1	İ	2	İ	2016-03-02	i	6	İ

```
    | 2
    | 3
    | 2017-06-25 | 1
    |

    | 3
    | 1
    | 2016-03-01 | 0
    |

    | 3
    | 4
    | 2016-07-03 | 5
    |
```

Output:

İ	install_dt		installs		Day1_retention	
İ	2016-03-01 2017-06-25	İ	2	1	0.50 0.00	
+		+-		+-		+

Explanation: Player 1 and 3 installed the game on 2016-03-01 but only player 1 logged back in on 2016-03-02 so the day 1 retention of 2016-03-01 is 1/2=0.50 Player 2 installed the game on 2017-06-25 but didn't log back in on 2017-06-26 so the day 1 retention of 2017-06-25 is 0/1=0.00

Solution:

```
# Write your MySQL query statement below

select
   install_dt
   , sum(case when datediff_num = 0 then 1 else 0 end) as installs
   , round(sum(case when datediff_num = 1 then 1 else 0 end) / sum(case when datediff_num = 0

from
(
   select
   *
    , min(event_date) over(partition by player_id) as install_dt
    , datediff(event_date, min(event_date) over(partition by player_id)) as datediff_num
   from
    Activity
) t1
group by install_dt
```