05/14/2022

608. 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 **ordered** by `id` **in ascending order**.

MySQL

```
SELECT DISTINCT t1.id,(

CASE

    WHEN t1.p_id IS NULL THEN 'Root'

    WHEN t1.p_id IS NOT NULL AND t2.id IS NULL THEN 'Leaf'

    ELSE 'Inner'

    END
```

) AS Type

FROM Tree t1

LEFT JOIN Tree t2

ON t1.id = t2.p_id

ORDER BY t1.id ASC

176. 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`.

MySQL

SELECT MAX(salary) as SecondHighestSalary

FROM Employee

WHERE salary < (SELECT MAX(salary) FROM Employee)

#Solution 2

```
SELECT (
    SELECT DISTINCT Salary
    FROM EMPLOYEE
    ORDER BY Salary DESC
    LIMIT 1 OFFSET 1
) AS SecondHighestSalary
```

5/19/2022

1393. Capital Gain/Loss

Table: `Stocks`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| stock_name    | varchar |
| operation     | enum    |
| operation_day | int     |
| price         | int     |
+---------------+---------+
(stock_name, operation_day) is the primary key for this table.
The operation column is an ENUM of type ('Sell', 'Buy')
Each row of this table indicates that the stock which has stock_name had an operation
on the day operation_day with the price.
It is guaranteed that each 'Sell' operation for a stock has a corresponding 'Buy'
operation in a previous day. It is also guaranteed that each 'Buy' operation for a
stock has a corresponding 'Sell' operation in an upcoming day.
```

Write an SQL query to report the **Capital gain/loss** for each stock.

The **Capital gain/loss** of a stock is the total gain or loss after buying and selling the stock one or many times.

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

The query result format is in the following example.

Example 1:

```
Input:
Stocks table:
+---------------+-----------+---------------+--------+
| stock_name    | operation | operation_day | price  |
+---------------+-----------+---------------+--------+
| Leetcode      | Buy       | 1             | 1000   |
| Corona Masks  | Buy       | 2             | 10     |
| Leetcode      | Sell      | 5             | 9000   |
| Handbags      | Buy       | 17            | 30000  |
| Corona Masks  | Sell      | 3             | 1010   |
| Corona Masks  | Buy       | 4             | 1000   |
| Corona Masks  | Sell      | 5             | 500    |
| Corona Masks  | Buy       | 6             | 1000   |
| Handbags      | Sell      | 29            | 7000   |
| Corona Masks  | Sell      | 10            | 10000  |
+---------------+-----------+---------------+--------+
Output:
+---------------+-------------------+
| stock_name    | capital_gain_loss |
+---------------+-------------------+
| Corona Masks  | 9500              |
| Leetcode      | 8000              |

| Handbags      | -23000            |

+---------------+-------------------+
```

Explanation:

```
Leetcode stock was bought at day 1 for 1000$ and was sold at day 5 for 9000$. Capital
gain = 9000 - 1000 = 8000$.

Handbags stock was bought at day 17 for 30000$ and was sold at day 29 for 7000$.
Capital loss = 7000 - 30000 = -23000$.

Corona Masks stock was bought at day 1 for 10$ and was sold at day 3 for 1010$. It
was bought again at day 4 for 1000$ and was sold at day 5 for 500$. At last, it was
bought at day 6 for 1000$ and was sold at day 10 for 10000$. Capital gain/loss is the
sum of capital gains/losses for each ('Buy' --> 'Sell') operation = (1010 - 10) +
(500 - 1000) + (10000 - 1000) = 1000 - 500 + 9000 = 9500$.
```

MySQL

```
SELECT stock_name, SUM(
  CASE
    WHEN operation = 'Buy' THEN -price
    ELSE price
  END
) AS capital_gain_loss
FROM Stocks
GROUP BY stock_name;
```

1158. Market Analysis I

Table: `Users`

```
+----------------+---------+

| Column Name    | Type    |

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

| user_id        | int     |

| join_date      | date    |
| favorite_brand | varchar |

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

user_id is the primary key of this table.

This table has the info of the users of an online shopping website where users can
sell and buy items.
```

Table: `Orders`

```
+---------------+---------+

| Column Name   | Type    |

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

| order_id      | int     |

| order_date    | date    |

| item_id       | int     |

| buyer_id      | int     |

| seller_id     | int     |

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

order_id is the primary key of this table.

item_id is a foreign key to the Items table.

buyer_id and seller_id are foreign keys to the Users table.
```

Table: `Items`

```
+---------------+---------+

| Column Name   | Type    |

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

| item_id       | int     |

| item_brand    | varchar |

+---------------+---------+
item_id is the primary key of this table.
```

Write an SQL query to find for each user, the join date and the number of orders they made as a buyer in `2019`.

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

The query result format is in the following example.

Example 1:

```
Input:
Users table:
+---------+------------+----------------+
| user_id | join_date  | favorite_brand |
+---------+------------+----------------+
| 1       | 2018-01-01 | Lenovo         |
| 2       | 2018-02-09 | Samsung        |
| 3       | 2018-01-19 | LG             |
| 4       | 2018-05-21 | HP             |
+---------+------------+----------------+
Orders table:
+----------+------------+---------+----------+-----------+
| order_id | order_date | item_id | buyer_id | seller_id |
+----------+------------+---------+----------+-----------+
| 1        | 2019-08-01 | 4       | 1        | 2         |
| 2        | 2018-08-02 | 2       | 1        | 3         |
| 3        | 2019-08-03 | 3       | 2        | 3         |
| 4        | 2018-08-04 | 1       | 4        | 2         |
| 5        | 2018-08-04 | 1       | 3        | 4         |
| 6        | 2019-08-05 | 2       | 2        | 4         |
+----------+------------+---------+----------+-----------+
Items table:
+---------+------------+
| item_id | item_brand |
+---------+------------+
| 1       | Samsung    |
| 2       | Lenovo     |
| 3       | LG         |
| 4       | HP         |
+---------+------------+
Output:
+-----------+------------+----------------+
| buyer_id  | join_date  | orders_in_2019 |
+-----------+------------+----------------+
| 1         | 2018-01-01 | 1              |
| 2         | 2018-02-09 | 2              |
| 3         | 2018-01-19 | 0              |
| 4         | 2018-05-21 | 0              |
+-----------+------------+----------------+
```

MySQL

```
SELECT u.user_id AS buyer_id, u.join_date, ifnull(COUNT(buyer_id),0) as orders_in_2019
FROM Users u
LEFT JOIN Orders O
ON u.user_id = o.buyer_id AND YEAR(o.order_date) = '2019'
GROUP BY u.user_id
```