2238. Number of Times a Driver was a Passenger

Column Name	Type
ride_id	int
driver_id	int
Passenger_id	int

ride_id is the primary key for this table. Each row of this table contains the ID of the driver and the ID of the passenger that rode in ride_id.

Note that driver_id != passenger_id.

Write an SQL query to report the ID of each driver and the number of times they were a passenger.

Return the result table in any order.

The query result format is in the following example.

Example 1.

Input:

Rides table :

ride_id	driver_id	Passenger_id
1	7	1
2	7	2
3	11	1
4	11	7
5	11	7
6	11	3

Output:

driver_id	cnt
7	2
11	0

Answer:

With cte AS

(SELECT passenger_id, COUNT(*) AS num_of_times

FROM Rides

GROUP BY passenger_id)

(Hint:left join with common table expression)

SELECT DISTINCT r.driver_id, CASE WHER c.num_of _times

IS NOT NULL THEN c.num_of_times

ELSE 0 END AS cnt

FROM Rides r

LEFT JOIN cte c

ON r.driver_id=c.passenger_id

1355. Activity Participants

Table: Friends

Column Name	Type
id	int
name	varchar
activity	varchar

id is the id of the friend and primary key for this table.

Name is the name of the friend.

Activity is the name of the activity which the friend takes part in.

Table: Activities

Column Name	Type
id	int
name	varchar

Id is the primary key for this table. Name is the name of the activity.

Write an SQL query to find the names of all the activities with neither the maximum nor the minimum number of participants.

Each activity in the **Activities** table is performed by any person in the table Friends.

Return the result table in any order.

The query result format is in the following example.

Example 1.

Input:

Friends table:

id	name	activity
1	Jonathan D	Eating
2	Jade W	Singing
3	Victor J	Singing
4	Elvis Q	Eating
5	Daniel A	Eating
6	Bob B	Horse Riding

Activities table:

id	**Name**
1	Eating
2	Singing
3	Horse Riding

Output:

activity
Singing

Answer:

```
WITH cte AS
(SELECT activity, COUNT(id) AS num_part
FROM Friends
GROUP BY activity),

Cte2 AS
(SELECT a.name, CASE WHEN c.num_part IS NOT NULL THEN c.num_part
ELSE 0 END AS frequency
FROM Activities a
LEFT JOIN cte c
ON a.name=c.activity)

SELECT name AS activity
FROM cte2
WHERE frequency <> (SELECT MAX(frequency) FROM cte2)
AND frequency <> (SELECT MIN (frequency) FROM cte2)
```

1709. Biggest Window Between Visits

Table: UserVisits

Column Name	Type
User_id	int
Visit_date	date

This table does not have a primary key.

This table contains logs of the dates that users visited a certain retailer.

Assume today's date is '2021-1-1'.

Write an SQL query that will,for each user_id,find out the largest window of days between each visit and the one right after it(or today if you are considering the last visit)

Return the result table ordered by user_id.

The query result format is in the following example.

Example 1.

Input: UserVisits table:

user_id	visit_date
1	2020-11-28
1	2020-10-20
1	2020-12-3
2	2020-10-5
2	2020-12-9
3	2020-11-11

Output: Result table:

user_id	biggest_window
1	39
2	65
3	51

For the first user, the windows in question are between dates:

- 2020-10-20 and 2020-11-28 with a total of 39 days.
- 2020-11-28 and 2020-12-3 with a total of 5 days.
- 2020-12-3 and 2021-1-1 with a total of 29 days.

Making the biggest window the one with 39 days.

For the second user, the windows in question are between dates:

- 2020-10-5 and 2020-12-9 with a total of 65 days.
- 2020-12-9 and 2021-1-1 with a total of 23 days.

Making the biggest window the one with 65 days.

For the third user, the only window in question is between dates 2020-11-11 and 2021-1-1 with a total of 51 days.

Answer:

```
with next_day AS(
SELECT *,
 IFNULL(LEAD(visit_date ,1) OVER(PARTITION BY user_id ORDER BY visit_date),'2021-1-1') AS NEXT_DATE
 FROM UserVisits)
SELECT user_id, MAX(DATEDIFF(NEXT_DATE,visit_date)) AS biggest_window
FROM next_day
GROUP BY user_id;


-- You don't have to find rank to find the Max, you can simply group by and use the MAX function. -- Import you use Rank where multiple row can have same rank. Remember this


SELECT USER_ID, MAX(VISIT_WINDOW) AS biggest_window
FROM (SELECT  USER_ID,
   DATEDIFF(LEAD(VISIT_DATE,1,'2021-1-1') OVER(PARTITION BY USER_ID ORDER BY VISIT_DATE),VISIT_DATE) AS VISIT_WINDOW
FROM USERVISITS
ORDER BY 1,2) AS A
GROUP BY 1
```

1077. Project Employees III

Table: Project

Column Name	Type
project_id	int
employee_id	int

(project_id, employee_id) is the primary key (combination of columns with unique values) of this table.

employee_id is a foreign key (reference column) to Employee table.

Each row of this table indicates that the employee with employee_id is working on the project with project_id.

Table: Employee

Column Name	Type
employee_id	int
name	varchar
experience_years	int

employee_id is the primary key (column with unique values) of this table.

Each row of this table contains information about one employee.

Write a solution to report the most experienced employees in each project. In case of a tie, report all employees with the maximum number of experience years.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input: Project table:

project_id	employee_id
1	1
1	2
1	3
2	1
2	4

Employee table:

employee_id	name	experience_years
1	Khaled	3
2	Ali	2
3	John	3
4	Doe	2

Output:

project_id	employee_id
1	1
1	3
2	1

Explanation: Both employees with id 1 and 3 have the most experience among the employees of the first project. For the second project, the employee with id 1 has the most experience.

Answer:

```
WITH
  T AS (
    SELECT
      *,
      RANK() OVER (
        PARTITION BY project_id
        ORDER BY experience_years DESC
      ) AS rk
    FROM
      Project
      JOIN Employee USING (employee_id)
  )
SELECT project_id, employee_id
FROM T
WHERE rk = 1;
```

1204. Last Person to Fit in the Bus

Column Name	Type
person_id	int
person_name	varchar
weight	int
turn	int

person_id column contains unique values.

This table has the information about all people waiting for a bus.

The person_id and turn columns will contain all numbers from 1 to n, where n is the number of rows in the table.

turn determines the order of which the people will board the bus, where turn=1 denotes the first person to board and turn=n denotes the last person to board.

weight is the weight of the person in kilograms.

There is a queue of people waiting to board a bus. However, the bus has a weight limit of 1000 kilograms, so there may be some people who cannot board.

Write a solution to find the person_name of the last person that can fit on the bus without exceeding the weight limit. The test cases are generated such that the first person does not exceed the weight limit.

Note that *only one* person can board the bus at any given turn.

The result format is in the following example.

Example 1:

Input:

Queue table:

Person_id	person_name	weight	turn
5	Alice	250	1
4	Bob	175	5
3	Alex	350	2

6	John Cena	400	3
1	Winston	500	6
2	Marie	200	4

Output:

person_name
John Cena

Explanation: The folowing table is ordered by the turn for simplicity.

Turn	ID	Weight	Name	Total Weight	
1	5	Alice	250	250	
2	3	Alex	350	600	
3	6	John Cena	400	1000	**(last person to board)**
4	2	Marie	200	1200	
5	4	Bob	175		**(cannot board)**
6	1	Winston	500		

Answer:

```
WITH
  T AS (
    SELECT
      person_name,
      SUM(weight) OVER (ORDER BY turn) AS s
    FROM Queue
  )
SELECT person_name
FROM T
WHERE s <= 1000
ORDER BY s DESC
LIMIT 1;
```

1112. Highest Grade For Each Student

Column Name	Type
student_id	int
course_id	int
grade	int

(student_id, course_id) is the primary key (combination of columns with unique values) of this table.

grade is never NULL.

Write a solution to find the highest grade with its corresponding course for each student. In case of a tie, you should find the course with the smallest course_id.

Return the result table ordered by student_id in ascending order.

The result format is in the following example.

Example 1:

Input:

Enrollments table:

student_id	course_id	grade
2	2	95
2	3	95
1	1	90
1	2	99
3	1	80
3	2	75
3	3	82

Output:

student_id	course_id	grade
1	2	99
2	2	95
3	3	82

Answer:

Subquery

We can first query the highest grade of each student, and then query the minimum course number corresponding to the highest grade of each student.

```
SELECT student_id, MIN(course_id) AS course_id, grade
FROM Enrollments
WHERE
  (student_id, grade) IN (
    SELECT student_id, MAX(grade) AS grade
    FROM Enrollments
    GROUP BY 1
  )
GROUP BY 1
ORDER BY 1;
```

608. Tree Node

Table: Tree

Column Name	Type
id	int
p_id	int

id is the column with unique values 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:

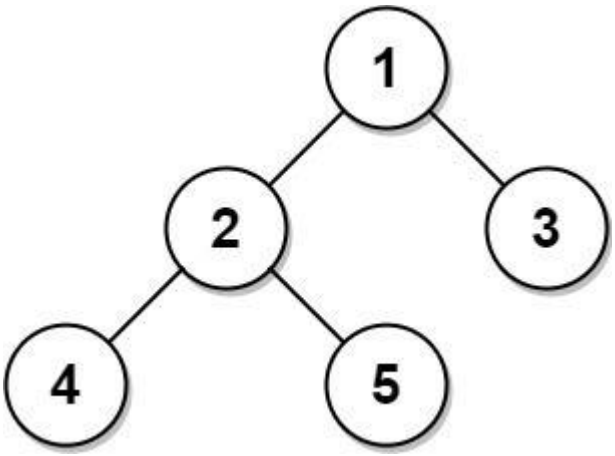
- "Leaf": if the node is a leaf node.
- "Root": if the node is the root of the tree.
- "Inner": If the node is neither a leaf node nor a root node.

Write a solution to report the type of each node in the tree.

Return the result table in any order.

The result format is in the following example.

Example 1:



Input:

Tree table:

id	P_id
1	Null
2	1
3	1
4	2
5	2

Output:

id	type
1	Root
2	Inner
3	Leaf
4	Leaf
5	Leaf

Explanation:

Node 1 is the root node because its parent node is null and it has child nodes 2 and 3.

Node 2 is an inner node because it has parent node 1 and child node 4 and 5.

Nodes 3, 4, and 5 are leaf nodes because they have parent nodes and they do not have child nodes.

Example 2:

Input:

Tree table:

id	P_id
1	null

Output:

id	type
1	Root

Explanation: If there is only one node on the tree, you only need to output its root attributes.

Answer:

```
SELECT
    id,
    CASE
        WHEN p_id IS NULL THEN 'Root'
        WHEN id IN (SELECT p_id FROM Tree) THEN 'Inner'
        ELSE 'Leaf'
    END AS type
FROM Tree;
```

1321. Restaurant Growth

Table: Customer

Column Name	Type
customer_id	int
name	varchar
visited_on	date
amount	int

In SQL,(customer_id, visited_on) is the primary key for this table.

This table contains data about customer transactions in a restaurant.

visited_on is the date on which the customer with ID (customer_id) has visited the restaurant.

amount is the total paid by a customer.

You are the restaurant owner and you want to analyze a possible expansion (there will be at least one customer every day).

Compute the moving average of how much the customer paid in a seven days window (i.e., current day + 6 days before). average_amount should be rounded to two decimal places.

Return the result table ordered by visited_on in ascending order.

The result format is in the following example**.**

Example 1:

Input:

Customer table:

customer_id	name	visited_on	amount
1	Jhon	2019-01-01	100
2	Daniel	2019-01-02	110
3	Jade	2019-01-03	120
4	Khaled	2019-01-04	130
5	Winston	2019-01-05	110
6	Elvis	2019-01-06	140
7	Anna	2019-01-07	150
8	Maria	2019-01-08	80
9	Jaze	2019-01-09	110
1	Jhon	2019-01-10	130
3	Jade	2019-01-10	150

Output:

visited_on	amount	average_amount
2019-01-07	860	122.86
2019-01-08	840	120
2019-01-09	840	120
2019-01-10	1000	142.86

Answer:

```
SELECT
    a.visited_on,
    SUM(b.amount) AS amount,
    ROUND(SUM(b.amount) / 7, 2) AS average_amount
FROM
    (SELECT DISTINCT visited_on FROM customer) AS a
    JOIN customer AS b ON DATEDIFF(a.visited_on, b.visited_on) BETWEEN 0 AND 6
WHERE a.visited_on >= (SELECT MIN(visited_on) FROM customer) + 6
GROUP BY 1
ORDER BY 1;
```

1532. The Most Recent Three Orders

Table: Customers

Column Name	Type
customer_id	int
name	varchar

customer_id is the column with unique values for this table.

This table contains information about customers.

Table: Orders

Column Name	Type
order_id	int
order_date	date
customer_id	int
cost	int

order_id is the column with unique values for this table.

This table contains information about the orders made by customer_id.

Each customer has one order per day.

Write a solution to find the most recent three orders of each user. If a user ordered less than three orders, return all of their orders.

Return the result table ordered by customer_name in ascending order and in case of a tie by the customer_id in ascending order. If there is still a tie, order them
by order_date in descending order.

The result format is in the following example.

Example 1:

Input:

Customers table:

customer_id	name
1	Winston
2	Jonathan
3	Annabelle
4	Marwan
5	Khaled

Orders table:

order_id	order_date	customer_id	cost
1	2020-07-31	1	30
2	2020-07-30	2	40
3	2020-07-31	3	70
4	2020-07-29	4	100
5	2020-06-10	1	1010
6	2020-08-01	2	102
7	2020-08-01	3	111

8	2020-08-03	1	99
9	2020-08-07	2	32
10	2020-07-15	1	2

Output:

customer_name	customer_id	order_id	order_date
Annabelle	3	7	2020-08-01
Annabelle	3	3	2020-07-31
Jonathan	2	9	2020-08-07
Jonathan	2	6	2020-08-01
Jonathan	2	2	2020-07-30
Marwan	4	4	2020-07-29
Winston	1	8	2020-08-03
Winston	1	1	2020-07-31
Winston	1	10	2020-07-15

Explanation:

Winston has 4 orders, we discard the order of "2020-06-10" because it is the oldest order.

Annabelle has only 2 orders, we return them.

Jonathan has exactly 3 orders.

Marwan ordered only one time.

We sort the result table by customer_name in ascending order, by customer_id in ascending order, and

Answer:

```
WITH
   T AS ( SELECT  *,ROW_NUMBER() OVER (
            PARTITION BY customer_id
            ORDER BY order_date DESC
         ) AS rk
      FROM Orders
         JOIN Customers USING (customer_id)   )
SELECT name AS customer_name, customer_id, order_id, order_date
FROM T WHERE rk <= 3
ORDER BY 1, 2, 4 DESC;
```

2112. The Airport With the Most Traffic

Table: Flights

Column Name	Type
departure_airport	int
arrival_airport	int
flights_count	int

(departure_airport, arrival_airport) is the primary key column (combination of columns with unique values) for this table.

Each row of this table indicates that there were flights_count flights that departed from departure_airport and arrival

Write a solution to report the ID of the airport with the most traffic. The airport with the most traffic is the airport that has the largest total number of flights that either departed from or arrived at the airport. If there is more than one airport with the most traffic, report them all.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:

Flights table:

departure_airport	arrival_airport	flights_count
1	2	4
2	1	5
2	4	5

Output:

| airport_id |

2

Explanation:

Airport 1 was engaged with 9 flights (4 departures, 5 arrivals).

Airport 2 was engaged with 14 flights (10 departures, 4 arrivals).

Airport 4 was engaged with 5 flights (5 arrivals).

The airport with the most traffic is airport 2.

Answer:

```
WITH
  T AS (
    SELECT * FROM Flights
    UNION
    SELECT arrival_airport, departure_airport, flights_count FROM Flights
  ),
  P AS (
    SELECT departure_airport, SUM(flights_count) AS cnt
    FROM T
    GROUP BY 1
  )
SELECT departure_airport AS airport_id
FROM P
WHERE cnt = (SELECT MAX(cnt) FROM P);
```

626. Exchange Seats

Table: Seat

Column Name	Type
id	int
student	varchar

id is the primary key (unique value) column for this table.

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

The ID sequence always starts from 1 and increments continuously.

Write a solution to swap the seat id of every two consecutive students. If the number of students is odd, the id of the last student is not swapped.

Return the result table ordered by id in ascending order.

Example 1:

Input: Seat table:

id	student
1	Abbot
2	Doris
3	Emerson
4	Green
5	Jeames

Output:

id	student
1	Doris
2	Abbot
3	Green
4	Emerson
5	Jeames

Explanation:

Note that if the number of students is odd, there is no need to change the last one's seat.

Answer:

```
SELECT
    id + (
        CASE
            WHEN id % 2 = 1
            AND id != (SELECT MAX(id) FROM Seat) THEN 1
            WHEN id % 2 = 0 THEN -1
            ELSE 0
        END
    ) AS id,
    student
FROM Seat
ORDER BY 1;
```

1164. Product Price at a Given Date

Table: Products

Column Name	Type
product_id	int
new_price	int
change_date	date

(product_id, change_date) is the primary key (combination of columns with unique values) of this table.

Each row of this table indicates that the price of some product was changed to a new price at some date.

Write a solution to find the prices of all products on 2019-08-16. Assume the price of all products before any change is 10.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input: Products table:

product_id	new_price	change_date
1	20	2019-08-14
2	50	2019-08-14
1	30	2019-08-15
1	35	2019-08-16
2	65	2019-08-17
3	20	2019-08-18

Output:

product_id	price
2	50
1	35
3	10

Answer:

```
WITH
  T AS (SELECT DISTINCT product_id FROM Products),
  P AS (
    SELECT product_id, new_price AS price
    FROM Products
    WHERE
      (product_id, change_date) IN (
        SELECT product_id, MAX(change_date) AS change_date
        FROM Products
        WHERE change_date <= '2019-08-16'
        GROUP BY 1
      )
  )
SELECT product_id, IFNULL(price, 10) AS price
FROM
  T
  LEFT JOIN P USING (product_id);
```

1045. Customers Who Bought All Products

Table: Customer

Column Name	Type
customer_id	int
product_key	int

This table may contain duplicates rows.

customer_id is not NULL.

product_key is a foreign key (reference column) to Product table.

Table: Product

Column Name	Type
product_key	int

product_key is the primary key (column with unique values) for this table.

Write a solution 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 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:

product_key
5
6

Output:

customer_id
1
3

Explanation:

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

Answer:

```
SELECT customer_id
FROM Customer
GROUP BY customer_id
HAVING COUNT(DISTINCT product_key) = (SELECT COUNT(distinct product_key)
FROM Product);
```

1193. Monthly Transactions I

Table: Transactions

Column Name	Type
id	int
country	varchar
state	enum
amount	int
trans_date	date

id is the primary key of this table.

The table has information about incoming transactions.

The state column is an enum of type ["approved", "declined"].

Write an SQL query to find for each month and country, the number of transactions and their total amount, the number of approved transactions and their total amount.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input: Transactions table:

id	country	state	amount	trans_date
121	US	approved	1000	2018-12-18
122	US	declined	2000	2018-12-19
123	US	approved	2000	2019-01-01
124	US	approved	2000	2019-01-07

Output:

month	country	trans_count	approved_count	trans_total_amount	approved_total_amount
2018-12	US	2	1	3000	1000
2019-01	US	1	1	2000	2000
2019-01	DE	1	1	2000	2000

Answer:

```
SELECT
    DATE_FORMAT(trans_date, '%Y-%m') AS month,
    country,
    COUNT(1) AS trans_count,
    SUM(state = 'approved') AS approved_count,
    SUM(amount) AS trans_total_amount,
    SUM(IF(state = 'approved', amount, 0)) AS approved_total_amount
FROM Transactions
GROUP BY 1, 2;
```

1158. Market Analysis I

Table: Users

Column Name	Type
user_id	int
join_date	date
favorite_brand	varchar

user_id is the primary key (column with unique values) 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 (column with unique values) of this table.

item_id is a foreign key (reference column) 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 (column with unique values) of this table.

Write a solution 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 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	2019-08-03	1	4	2
5	2019-08-03	1	3	4
6	2019-08-03	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

Answer:

```
SELECT
    u.user_id AS buyer_id,
    u.join_date,
    COUNT(order_id) AS orders_in_2019
FROM
    Users AS u
    LEFT JOIN Orders AS o ON u.user_id = o.buyer_id AND YEAR(order_date) = 2019
GROUP BY user_id;
```

1174. Immediate Food Delivery II

Table: Delivery

Column Name	Type
delivery_id	int
customer_id	int
order_date	date
customer_pref_delivery_date	date

delivery_id is the column of unique values of this table.

The table holds information about food delivery to customers that make orders at some date and specify a preferred delivery date (on the same order date or after it).

If the customer's preferred delivery date is the same as the order date, then the order is called immediate; otherwise, it is called scheduled.

The first order of a customer is the order with the earliest order date that the customer made. It is guaranteed that a customer has precisely one first order.

Write a solution to find the percentage of immediate orders in the first orders of all customers, rounded to 2 decimal places.

The result format is in the following example.

Example 1:

Input: Delivery table:

delivery_id	customer_id	order_date	customer_pref_delivery_date
1	1	2019-08-01	2019-08-02
2	2	2019-08-02	2019-08-02
3	1	2019-08-11	2019-08-12
4	3	2019-08-24	2019-08-24
5	3	2019-08-21	2019-08-22
6	2	2019-08-11	2019-08-13
7	4	2019-08-09	2019-08-09

Output:

immediate_percentage
50.00

Answer:

```
select
round(100*sum(case when b.min_order_date = b.min_delivery_date then 1 else 0 end)/count(*), 2)
as immediate_percentage
from (
  select min(order_date) as min_order_date, min(customer_pref_delivery_date) as min_delivery_date
  from delivery
  group by customer_id
) b;
```

178. Rank Scores

Table: Scores

Column Name	Type
id	int
score	decimal

id is the primary key (column with unique values) 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 a solution to find the rank of 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 result format is in the following example.

Example 1:

Input: Scores table:

id	score
1	3.50
2	3.65
3	4.00
4	3.85
5	4.00
6	3.65

Output:

score	rank
4.00	1
4.00	1
3.85	2
3.65	3
3.65	3
3.50	4

Answer:

SELECT score,DENSE_RANK() OVER(ORDER BY Score DESC)

AS ‘rank’

FROM Scores

ORDER BY score DESC

1126. Active Businesses

Table: Events

Column Name	Type
business_id	int
event_type	varchar
occurences	int

(business_id, event_type) is the primary key (combination of columns with unique values) of this table.

Each row in the table logs the info that an event of some type occurred at some business for a number of times.

The average activity for a particular event_type is the average occurences across all companies that have this event**.**

An active business is a business that has more than one event_type such that their occurences is strictly greater than the average activity for that event.

Write a solution to find all active businesses.Return the result table in any order.

The result format is in the following example.

Example 1:

Input: Events table:

business_id	event_type	occurences
1	reviews	7
3	reviews	3
1	ads	11
2	ads	7
3	ads	6
1	page views	3
2	page views	12

Output:

business_id
1

Answer:

```
SELECT business_id
FROM
    EVENTS AS t1
    JOIN (
        SELECT
            event_type,
            AVG(occurences) AS occurences
        FROM EVENTS
        GROUP BY event_type
    ) AS t2
        ON t1.event_type = t2.event_type
WHERE t1.occurences > t2.occurences
GROUP BY business_id
HAVING COUNT(1) > 1;
```

176. Second Highest Salary

Table: Employee

Column Name	Type
id	int
salary	int

id is the primary key (column with unique values) for this table.

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

Write a solution to find the second highest distinct salary from the Employee table. If there is no second highest salary, return null (return None in Pandas).The result format is in the following example.

Example 1:

Input: Employee table:

id	salary
1	100
2	200
3	300

Output:

SecondHighestSalary
200

Example 2:

Input: Employee table:

id	salary
1	100

Output:

SecondHighestSalary
null

Answer:

```
WITH
  RankedEmployees AS (
    SELECT *, DENSE_RANK() OVER(ORDER BY salary DESC) AS `rank`
    FROM Employee
  )
SELECT MAX(salary) AS SecondHighestSalary
FROM RankedEmployees
WHERE `rank` = 2;
```

2175. The Change in Global Rankings

Table: TeamPoints

Column Name	Type
team_id	int
name	varchar
points	int

team_id is the primary key for this table.

Each row of this table contains the ID of a national team, the name of the country it represents, and the point

Table: PointsChange

Column Name	Type
team_id	int
points_change	int

team_id is the primary key for this table.

Each row of this table contains the ID of a national team and the change in its points in the global rankings. points_change can be:

- 0: indicates no change in points.- positive: indicates an increase in points.

- negative: indicates a decrease in points. Each team_id that appears in TeamPoints will also appear in this table.

The global ranking of a national team is its rank after sorting all the teams by their points in descending order. If two teams have the same points, we break the tie by sorting them by their name in lexicographical order.

The points of each national team should be updated based on its corresponding points_change value.

Write an SQL query to calculate the change in the global rankings after updating each team's points.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input: TeamPoints table:

team_id	name	points
3	Algeria	1431
1	Senegal	2132
2	New Zealand	1402
4	Croatia	1817

PointsChange table:

team_id	points_change
3	399
2	0
4	13
1	-22

Output:

team_id	name	rank_diff
1	Senegal	0
4	Croatia	-1
3	Algeria	1
2	New Zealand	0

Explanation:

The global rankings were as follows:

team_id	name	points	rank
1	Senegal	2132	1
4	Croatia	1817	2
3	Algeria	1431	3
2	New Zealand	1402	4

After updating the points of each team, the rankings became the following:

team_id	name	points	rank
1	Senegal	2110	1
3	Algeria	1830	2
4	Croatia	1830	3
2	New Zealand	1402	4

Since after updating the points Algeria and Croatia have the same points, they are ranked according to their lexicographic order. Senegal lost 22 points but their rank did not change. Croatia gained 13 points but their rank decreased by one. Algeria gained 399 points and their rank increased by one. New Zealand did not gain or lose points and their rank did not change.

Answer:

```
select team_id, name, cast(old_rk as signed)-cast(new_rk as signed) rank_diff
from (
    select t.team_id, name,
    rank() over (order by points desc, name) old_rk,
    rank() over (order by points_change+points desc, name) new_rk
    from TeamPoints t join PointsChange p on t.team_id=p.team_id) t;
```

1549. The Most Recent Orders for Each Product

Table: Customers

Column Name	Type
customer_id	int
name	varchar

customer_id is the column with unique values for this table.

This table contains information about the customers.

Table: Orders

Column Name	Type
order_id	int
order_date	date
customer_id	int
product_id	int

order_id is the column with unique values for this table.

This table contains information about the orders made by customer_id.

There will be no product ordered by the same user more than once in one day.

Table: Products

Column Name	Type
product_id	int
product_name	varchar
price	int

product_id is the column with unique values for this table.

This table contains information about the Products.

Write a solution to find the most recent order(s) of each product.

Return the result table ordered by product_name in ascending order and in case of a tie by the product_id in ascending order. If there still a tie, order them by order_id in ascending order.

The result format is in the following example.

Example 1:

Input: Customers table:

customer_id	name
1	Winston
2	Jonathan
3	Annabelle
4	Marwan
5	Khaled

Orders table:

order_id	order_date	customer_id	product_id
1	2020-07-31	1	1
2	2020-07-30	2	2
3	2020-08-29	3	3
4	2020-07-29	4	1
5	2020-06-10	1	2
6	2020-08-01	2	1
7	2020-08-01	3	1
8	2020-08-03	1	2
9	2020-08-07	2	3
10	2020-07-15	1	2

Products table:

product_id	product_name	price
1	keyboard	120
2	mouse	80
3	screen	600
4	hard disk	450

Output:

product_name	product_id	order_id	order_date
keyboard	1	6	2020-08-01
keyboard	1	7	2020-08-01
mouse	2	8	2020-08-03
screen	3	3	2020-08-29

Explanation:

keyboard's most recent order is in 2020-08-01, it was ordered two times this day.

mouse's most recent order is in 2020-08-03, it was ordered only once this day.

screen's most recent order is in 2020-08-29, it was ordered only once this day.

The hard disk was never ordered and we do not include it in the result table.

Answer:

```
WITH
  T AS (
    SELECT
      *,
      RANK() OVER (
        PARTITION BY product_id
        ORDER BY order_date DESC
      ) AS rk
    FROM
      Orders
      JOIN Products USING (product_id)
  )
SELECT product_name, product_id, order_id, order_date
FROM T
WHERE rk = 1
ORDER BY 1, 2, 3;
```