1270. All People Report to the Given Manager.

Table**:** Employee

Column_Name	Type
Employee_id	int
Employee_name	varchar
manager_id	int

employee id is the primary key for this table. Each row of this table indicates that the employee with ID employee_id and name employee name reports his work to his/her direct manager with manager_id.

The head of the company is the employee with employee_id =1.

Write an SQL query to find employee_id of all employees that directly or indirectly report their work to the head of the company.

The indirect relation between managers will not exceed three managers as the company is small.

Return the result table in any order

The query result format is in the following example.

Example 1:

Input:

Employee table:

Employee_id	Employee_name	manager_id
1	Boss	1
3	Alice	3
2	Bob	1
4	Daniel	2
7	Luis	4
8	Jhon	3
9	Angela	8
77	Robert	1

Output:

Employee_id
2
77
4
7

ANSWER:

SELECT e1. employee_id

FROM Employees e1

LEFT JOIN Employees e2

ON e1. manager_id = e2. employee_id

LEFT JOIN Employees e3

ON e2. manager_id = e3. employee_id

WHERE e3. manager_id = 1

AND e1. employee _id <> e1. manager _id;

(**Hint**: we can do self-join, left-join

2 Bob 1 | 1 Boss 1 | 1 Boss 1

4 Daniel 2 | 2 Bob 1 | 1 Boss 1

7 Luis 4 | 4 Daniel 2 | 2 Bob 1

77 Robert 1 | 1 Boss 1 | 1 Boss 1).

1853. Convert to Date Format.

Table: Days

Column_Name	Type
Day	date

Day is the primary key for this table.

Write an SQL query to convert each date in Days into a string formatted as "day name, month name day, year".

Return the result table in any order.

The query result format is in the following example.

Example 1:

Example 1:

Input:

Days table:

day
2022-04-12
2021-08-09
2020-06-26

Output:

day
Tuesday, April 12, 2022
Monday, August 9, 2021
Friday, June 26, 2020

ANSWER:

SELECT To_char (day, 'DAY, MONTH DD, YYYY')

FROM Days;

(**Hint**: we can use To_char to convert date to different formats).

1699. Number of calls between two persons.

Table: Calls

Column_Name	Type
from_id	int
to_id	int
duration	int

This table does not have a primary key, it may contain duplicates.

This table contains the duration of a phone call between from_id and to_id.

from id ! = to id.

Write an SQL query to report the number of calls and the total call duration between each pair of distinct persons (person1, person2) where person1< person2.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:

Calls Table:

from_id	**to_id**	**duration**
1	2	59
2	1	11
1	3	20
3	4	100
3	4	200
3	4	200
4	3	499

Output:

Person1	Person2	Call_count	Total_duration
1	2	2	70
1	3	1	20
3	4	4	999

ANSWER:

```
SELECT person1, person2, count (*) AS Call_count, SUM (duration) AS Total_duration

FROM

(SELECT CASE WHEN from_id < to_id THEN from_id

                ELSE to_id END AS person1,

                CASE WHEN from_id < to_id THEN to_id

                ELSE from_id END AS person2, duration from Calls)

GROUP BY person1, person2;
```

OR

SELECT from_id, to_id, SUM (duration)

FROM

(SELECT from_id, to_id, duration from Calls

UNION ALL

SELECT to_id, from_id, duration from Calls)

WHERE from_id<to_id

GROUP BY from_id, to_id;

(**Hint**: we can do it by using CASE and UNION ALL and GROUP BY to group the records).

2066. Account Balance.

Table: Transactions

Column_Name	**Type**
Account_id	int
day	date
type	ENUM
amount	int

(Account_id, day) is the primary key for this table.

Each row contains information about one transaction, including the transaction type, the day it occurred on, and the amount.

type is an ENUM of the type ('Deposit', 'Withdraw').

Write an SQL query to report the balance of each user after each transaction. You may assume that the balance of each account before any transaction is and that the balance will never be below at any moment.

Return the result table in ascending order by Account_id, then by day in case of a tie

The query result format is in the following example.

Example 1:

Input:

Transactions Table:

Account_id	day	type	amount
1	2021-11-07	Deposit	2000
1	2021-11-09	Withdraw	1000
1	2021-11-11	Deposit	3000
2	2021-12-07	Deposit	7000
2	2021-12-12	Withdraw	7000

Output:

Account_id	day	amount
1	2021-11-07	2000
1	2021-11-09	1000
1	2021-11-11	4000
2	2021-12-07	7000
2	2021-12-12	0

ANSWER:

```
SELECT Account_id, day, SUM (CASE WHEN type = 'Deposit' THEN amount
                                  ELSE -amount
                             END)
OVER (PARTITION BY Account_id
      ORDER BY day) AS balance
FROM Transactions
ORDER BY Account_id, day;
```

1596. The Most Frequently Ordered Products for Each Customer.

Table: Customers.

Column_Name	Type
Customer_id	int
name	varchar

Customer_id is the primary key 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 primary key for this table.

This table contains information about the orders made by Customer_id.

No customer will order the same product more than once in a single day.

Table: Products.

Column_Name	Type
product_id	int
Product_name	varchar
Price	int

product_id is the primary key for this table.

This table contains information about the products.

Write an SQL query to find the most frequently ordered product(s) for each customer.

1# Write your MySQL query statement below

2#The result table should have the product id and product name for each customer id who ordered at least one order.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:

Customers Table:

Customer_id	name
1	Alice
2	Bob
3	Tom
4	Jerry
5	John

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	3
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:

Customer_id	Product_id	Product_name
1	2	Mouse
2	1	Keyboard
2	2	mouse

ANSWER:

```
SELECT c. customer_id, c. product_id, p. product_name
FROM (SELECT *, MAX (num_ordered) OVER (PARTITION BY customer_id
                                        ORDER BY num_ordered DESC) AS most_frequent
      FROM (SELECT customer_id, product_id, COUNT (*) AS num_ordered
            FROM Orders
            GROUP BY customer_id, product_id)) AS c
LEFT JOIN Products p
ON c. product_id = p. product_id
WHERE c.num_ordered = c. most_frequent;
```

1173. Immediate Food Delivery 1.

Table: Delivery.

Column_Name	Type
Delivery_id	Int
Customer_id	Int
Order_date	Date
Customer_pref_delivery_date	date

Delivery_id is the primary key 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.

Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal places.

The query 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	5	2019-08-02	2019-08-02
3	1	2019-08-11	2019-08-11
4	3	2019-08-24	2019-08-26
5	4	2019-08-21	2019-08-22
6	2	2019-08-11	2019-08-13

Output:

Immediate_percentage
33.33

ANSWER:

```
SELECT ROUND (SUM (CASE WHEN Order_date = Customer_pref_delivery_date THEN 1
                        ELSE 0
                   END)/ COUNT (*) * 100,2) AS Immediate_percentage
FROM Delivery;
```

1831. Maximum Transaction Each Day.

Table: Transactions

Column_Name	Type
Transaction_id	Int
Day	Datetime
amount	int

transaction_id is the primary key for this table. Each row contains information about one transaction.

Write an SQL query to report the IDs of the transactions with the maximum amount on their respective day. If in one day there are multiple such transactions, return all of them.

Return the result table ordered by transaction id in ascending order.

The query result format is in the following example.

Example 1:

Input:

Transaction table:

Transaction_id	Day	amount
8	2021-4-3 15:57:28	57
9	2021-4-28 08:47:25	21
1	2021-4-29 13:28:30	58
5	2021-4-28 16:39:59	40
6	2021-4-29 23:39:29	58

output:

Transaction_id
1
5
6
8

ANSWER:

```
SELECT transaction_id
FROM (SELECT *, MAX (amount) OVER (PARTITION BY DAY
ORDER BY amount DESC) AS maximum_amount
FROM (SELECT transaction_id, TO_CHAR (day, 'YYYY-MM-DD') AS DAY, amount
        FROM Transactions))
WHERE amount = maximum_amount
ORDER BY transaction_id;
```

OR

```
SELECT transaction_id
FROM (SELECT *, MAX (amount) OVER (PARTITION BY TO_CHAR (day, 'YYYY-MM-DD')
                ORDER BY amount DESC) AS maximum_amount
        FROM Transactions)
WHERE amount = maximum_amount
ORDER BY transaction_id;
```

1068. Product Sales Analysis I.

Table: Sales

Column_Name	Type
Sale_id	Int
Product_id	Int
Year	Int
Quantity	Int
price	int

(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

Column_Name	Type
Product_id	Int
Product_name	varchar

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 reports the product_name, year, and price for each sale id in the Sales table.

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	100	2008	10	5000
2	100	2009	12	5000
7	200	2011	15	9000

Product table:

Product_id	Product_name
100	Nokia
200	Apple
300	Samsung

Output:

Product_id	Total_ quantity
100	22
200	15

ANSWER:

SELECT p. Product_name, s. year

FROM Sales

LEFT JOIN Product p

ON s. product_id = p. product_id;

1069. Product Sales Analysis II.

Table: Sales

Column_Name	Type
Sale_id	Int
Product_id	Int
Year	Int
Quantity	Int
price	int

(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.

Write an SQL query that reports the total quantity sold for every product id.

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	100	2008	10	5000
2	100	2009	12	5000
7	200	2011	15	9000

Product table:

Product_id	Product_name
100	Nokia
200	Apple
300	Samsung

Output:

Product_id	Total_ quantity
100	22
200	15

ANSWER:

```
SELECT product_id, SUM (quantity) AS Total_ quantity
FROM Sales
GROUP BY product_id;
```

1468. Calculate Salaries.

Table: Salaries.

Column_Name	Type
Company_id	Int
Employee_id	Int
Employee_name	Varchar
salary	int

(Company_id, employee_id) is the primary key for this table.

This table contains the company id, the id, the name, and the salary for an employee.

Write an SQL query to find the salaries of the employees after applying taxes.

Round the salary to the nearest integer.

The tax rate is calculated for each company based on the following criteria:

- 0s If the max salary of any employee in the company is less than $1000.
- 24% If the max salary of any employee in the company is in the range. [1000, 100001 inclusive.
- 49% If the max salary of any employee in the company is greater than $10000.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:

Salaries table:

Company_id	Employee_id	Employee_name	salary
1	1	Tony	2000
1	2	Pronub	21300
1	3	Tyrrox	10800
2	1	Pam	300
2	7	Bassem	450
2	9	Hermione	700
3	7	Bocaben	100
3	2	Ognjen	2200
3	13	Nyancat	3300
3	2	Morninngcat	7777

output:

Company_id	Employee_id	Employee_name	salary
1	1	Tony	1020
1	2	Pronub	10863
1	3	Tyrrox	5508
2	1	Pam	300
2	7	Bassem	450
2	9	Hermione	700
3	7	Bocaben	76
3	2	Ognjen	1672

ANSWER:

```
SELECT   Company_id, employee_id, Employee_name,
ROUND (CASE WHEN MAX (salary) OVER (PARTITION BY Company_id) < 1000 THEN salary
            WHEN MAX (salary) OVER (PARTITION BY Company_id)
                                    BETWEEN 1000 AND 10000 THEN 0.76*salary
            ELSE 0.51*salary
            END, 0) AS salary
FROM Salaries;
```

534. Game Play Analysis III

Table: Activity

Column_Name	Type
Player_id	Int
Device_id	Int
Event_date	Date
Games_played	int

(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.

Write an SQL query to report for each player and date, how many games played so far by the player. That is, the total number of games played by the player until that date. Check the example for clarity.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:

Activity Table:

Player_id	Device_id	Event_date	Games_played
1	2	2016-03-01	5
1	2	2016-05-02	6
1	3	2016-06-25	1
3	1	2016-03-02	0
3	4	2016-07-03	5

Output:

Player_id	Event_date	Games_played_so_far
1	2016-03-01	5
1	2016-05-02	11
1	2016-06-25	12
3	2016-03-02	0
3	2016-07-03	5

ANSWER:

```
SELECT Player_id, Event_date, SUM(Games_played) OVER (PARTITION BY Player_id
                                                      ORDER BY Event_date) AS Games_played_so_far
FROM Activity;
```

1398. Customers Who Bought Products A and B but Not C.

Table: Customers

Column_Name	Type
Customer_id	Int
Customer_name	varchar

customer_id is the primary key for this table.

Customer_name is the name of the customer.

Table: orders

Column_Name	Type
Order_id	Int
Customer_id	int
Product_name	varchar

order id is the primary key for this table.

customer id is the id of the customer who bought the product "Product_name".

Write an SQL query to report the customer_id and Customer_name of customers who bought products "A", "B" but did not buy the product "C" since we want to recommend them to purchase this product.

Return the result table ordered by customer id.

The query result format is in the following example.

Example 1:

Input:

Customers Table:

Customer_id	Customer_name
1	Daniel
2	Diana
3	Elizabeth
4	Jhon

orders Table:

Order_id	Customer_id	Product_name
10	1	A
20	1	B
30	1	D
40	1	C
50	2	A
60	3	A
70	3	B
80	3	D
90	4	C

Output:

Customer_id	Customer_name
3	Elizabeth

ANSWER:

```
SELECT *FROM Customers
WHERE customer_id IN (SELECT customer_id
                      FROM Orders
                      GROUP BY customer_id
                      HAVING SUM (CASE WHEN Product_name = 'A' THEN 1
                                  ELSE 0 END) > 0
                      AND SUM (CASE WHEN Product_name = 'B' THEN 1
                                  ELSE 0 END) > 0
                      AND SUM (CASE WHEN Product_name = 'C' THEN 1 ELSE 0 END) = 0)
ORDER BY customer_id;
```

511. Game Play Analysis I

Table: Activity

Column_Name	Type
Player_id	Int
Device_id	Int
Event_date	Date
Games_played	int

(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.

Write an SQL query to report the first login date for each player.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:

Activity Table:

Player_id	Device_id	Event_date	Games_played
1	2	2016-03-01	5
1	2	2016-05-02	6
1	3	2016-06-25	1
3	1	2016-03-02	0
3	4	2016-07-03	5

Player_id	First_login
1	2016-03-01
2	2016-06-25
3	2016-03-02

ANSWER:

SELECT Player_id, MIN Event_date AS First_login

FROM Activity

GROUP BY Player_id;

1789. Primary Department for Each Employee

Table: Employee

Column_Name	Type
Employee_id	Int
Department_id	Int
Primary_flag	varchar

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

employee_id is the id of the employee.

Department_id is the id of the department to which the employee belongs.

Primary_flag is an ENUM (category) of type ('Y', 'N'). If the flag is 'Y', the department is the primary department for the employee. If the flag is 'N', the department is not the primary.

Employees can belong to multiple departments. When the employee joins other departments, they need to decide which department is their primary department. Note that when an employee belongs to only one department, their primary column is 'N'.

Write a solution to report all the employees with their primary department. For employees who belong to one department, report their only department.

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

The result format is in the following example.

Example 1:

Input:

Employee Table:

Employee_id	Department_id	Primary_flag
1	1	N
2	1	Y
2	2	N
3	3	N
4	2	N
4	3	Y
4	4	N

Output:

Employee_id	Department_id
1	1
2	1
3	3
4	3

Explanation:

- The Primary department for employee 1 is 1.

- The Primary department for employee 2 is 1.

- The Primary department for employee 3 is 3.

- The Primary department for employee 4 is 3.

ANSWER:

```
SELECT employee_id, CASE WHEN COUNT (Department_id) = 1 THEN Department_id
        WHEN COUNT (Department_id > 1 THEN SUM (Primary_flag = 'Y') *Department_id)
        END AS Department_id
FROM Employee
GROUP BY employee_id;
```

1327. List the Products Ordered in a Period

Table: Products

Column_Name	Type
Product_id	Int
Product_name	Varchar
product_category	varchar

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

This table contains data about the company's products.

Table: Orders

Column_Name	Type
Product_id	Int
order_date	date
unit	Int

This table may have duplicate rows.

product_id is a foreign key (reference column) to the Products table.

unit is the number of products ordered in order_date.

Write a solution to get the names of products that have at least 100 units ordered in **February 2020** and their amount.

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

The result format is in the following example.

Example 1:

Input:

Products table:

Product_id	Product_name	product_category
1	Leetcode solutions	Book
2	Jewels of stringology	Book
3	HP	Laptop
4	Lenovo	Laptop
5	Leetcode Kit	T-shirt

Orders table:

product_id	order_date	unit
1	2020-02-05	60
1	2020-02-10	70
2	2020-01-18	30
2	2020-02-11	80
3	2020-02-17	2
3	2020-02-24	3
4	2020-03-01	20
4	2020-03-04	30
4	2020-03-04	60
5	2020-02-25	50
5	2020-02-27	50
5	2020-03-01	50

Output:

Product_name	unit
Leetcode Solutions	130
Leetcode Kit	100

Explanation:

Products with product_id = 1 is ordered in February a total of (60 + 70) = 130.

Products with product_id = 2 is ordered in February a total of 80.

Products with product_id = 3 is ordered in February a total of (2 + 3) = 5.

Products with product_id = 4 was not ordered in February 2020.

Products with product_id = 5 is ordered in February a total of (50 + 50) = 100.

ANSWER:

```
SELECT p. Product_name, o. units_feb AS unit
FROM (SELECT product_id, SUM (units) AS units_feb
        FROM Orders
        WHERE YEAR (order_date) = 2020 AND MONTH (order_date) = 2
        GROUP BY product_id)  AS o
LEFT JOIN Products p ON o. product_id = p. product_id
WHERE o. units_feb >= 100;
```

607. Sales Person.

Table: **SalesPerson.**

Column_Name	Type
sales_id	Int
Name	Varchar
Salary	Int
Commission_rate	Int
Hire_date	date

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

Each row of this table indicates the name and the ID of a salesperson alongside their salary, commission rate, and hire date.

Table: Company

Column_Name	Type
Com_id	Int
Name	Varchar
City	varchar

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

Each row of this table indicates the name and the ID of a company and the city in which the company is located.

Table: Orders

Column_Name	Type
order_id	Int
order_date	Date
com_id sales_id	int
amount	Int

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

com_id is a foreign key (reference column) to com_id from the Company table.

sales_id is a foreign key (reference column) to sales_id from the SalesPerson table.

Each row of this table contains information about one order. This includes the ID of the company, the ID of the salesperson, the date of the order, and the amount paid.

Write a solution to find the names of all the salespersons who did not have any orders related to the company with the name **"RED"**.

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

The result format is in the following example.

Example 1:

Input:

SalesPerson table:

sales_id	name	Salary	Commission_rate	Hire_date
1	John	100000	6	4/1/2006
2	Amy	12000	5	5/1/2010
3	Mark	65000	12	12/25/2008
4	Pam	25000	25	1/1/2005
5	Alex	5000	10	2/3/2007

Company table:

com_id	name	city
1	RED	Boston
2	ORANGE	New York
3	YELLOW	Boston
4	GREEN	Austin

Orders table:

order_id	order_date	com_id	sales_id	amount
1	1/1/2014	3	4	100000
2	2/1/2014	4	5	5000
3	3/1/2014	1	1	50000
4	4/1/2014	1	4	25000

Output:

name
Amy
Mark
Alex

Explanation:

According to orders 3 and 4 in the Orders table, it is easy to tell that only salesperson John and Pam have sales to company RED, so we report all the other names in the table salesperson.

ANSWER:

```
SELECT name
FROM SalesPerson
WHERE sales_id NOT IN (SELECT o. sales_id
                       FROM Orders o
                       LEFT JOIN Company c ON o.com_id = c.com_id
                       WHERE c.name LIKE 'RED');
```

1715. Count Apples and Oranges

Table: Boxes.

Column_Name	Type
Box_id	int
Chest_id	Int
Apple_account	Int
Orange_account	Int

box id is the primary key for this table. chest id is a foreign key of the chests table. This table contains information about the boxes and the number of oranges and apples they have. Each box may include a chest, which also can contain oranges and apples.

Table: Chests

Column_Name	**Type**
Chest_id	Int
Apple_account	Int
Orange_account	int

chest id is the primary key for this table. This table contains information about the chests and the corresponding number of oranges and apples they have.

Write an SQL query to count the number of apples and oranges in all the boxes. If a box contains a chest, you should also include the number of apples and oranges it has.

The query result format is in the following example.

Example 1:

Input:

Boxes table:

Box_id	**Chest_id**	**Apple_account**	**Orange_account**
2	Null	6	15
18	14	4	15
19	3	8	4
12	2	19	20
20	6	12	9
8	6	9	9
3	14	16	7

chests table:

Chest_id	**Apple_account**	**Orange_account**
6	5	6
14	20	10
2	8	8
3	19	4
16	19	19

Output:

Apple_account	Orange_account
151	123

ANSWER:

```
SELECT SUM (CASE WHEN b. chest_id IS NULL THEN b.apple_count
                ELSE b.apple_count + c.apple_count
                END) AS apple_count, SUM (CASE WHEN b. chest_id IS NULL
                                                THEN b.orange_count
                                            ELSE b.orange_count + c.orange_count
                                            END) AS orange_count
FROM Boxes b
LEFT JOIN Chests c
ON b. chest_id = c.chest_id;
```

2041. Accepted Candidates From the Interviews.

Table: Candidates

Column_Name	Type
Candidate_id	Int
Name	Varchar
Years_of_exp	Int
Interview_id	int

candidate_id is the primary key column for this table. Each row of this table indicates the name of a candidate, their number of years of experience, and their interview ID.

Table: Rounds.

Column_Name	Type
Interview_id	Int
Round_id	Int
score	int

(interview_id, round_id) is the primary key column for this table.Each row of this table indicates the score of one round of an interview.

Write an SQL query to report the IDs of the candidates who have at least two years of experience and the sum of the score of their interview rounds is strictly greater than 15.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:

Candidates table:

Candidate_id	Name	Years_of_exp	Interview_id
11	Atticus	1	101
9	Ruben	6	104
6	Aliza	10	109
8	Alfredo	0	107

Rounds table:

Interview_id	Round_id	score
109	3	4
101	2	8
109	4	1
107	1	3
104	3	6
109	1	4
104	4	7
104	1	2
109	2	1
104	2	7
107	2	3
101	1	8

Output:

Candidate_id
9

ANSWER:

```
SELECT candidate_id
FROM Candidates AS c
LEFT JOIN Rounds AS r ON c.interview_id = r.interview_id
WHERE years_of_exp >= 2
GROUP BY c.interview_id
HAVING sum(score) > 15;
```

1934. Confirmation Rate.

Table: Signups

Column_Name	Type
User_id	Int
Time_stamp	datetime

user_id is the primary key for this table.

Each row contains information about the signup time for the user with ID user_id.

Table: Confirmations.

Column_Name	Type
User_id	Int
Time_stamp	Datetime
section	ENUM

(user_id, time_stamp) is the primary key for this table.

user_id is a foreign key with a reference to the Signups table.

action is an ENUM of the type ('confirmed', 'timeout')

Each row of this table indicates that the user with ID user_id requested a confirmation message at time_stamp and that confirmation message was either confirmed ('confirmed') or expired without confirming ('timeout').

The confirmation rate of a user is the number of 'confirmed' messages divided by the total number of requested confirmation messages. The confirmation rate of a user that did not request any confirmation messages is 0. Round the confirmation rate to two decimal places.

Write an SQL query to find the confirmation rate of each user.Return the result table in any order.The query result format is in the following example:

Example 1:

Input:

Signups table:

User_id	Time_stamp
3	2020-03-21 10:16:13
7	2020-01-04 13:57:59
2	2020-07-29 23:09:44
6	2020-12-09 10:39:37

Confirmations table:

User_id	Time_stamp	action
3	2021-01-06 03:30:46	Timeout
3	2021-07-14 14:00:00	Timeout
7	2021-06-12 11:57:29	Confirmed
7	2021-06-13 12:58:28	Confirmed
7	2021-06-14 13:59:27	Confirmed
2	2021-01-22 00:00:00	Confirmed
2	2021-02-28 23:59:59	Timeout

Result table

User_id	confirmation_rate
6	0.00
3	0.00
7	1.00
2	0.50

Explanation:

User 6 did not request any confirmation messages. The confirmation rate is 0.

User 3 made 2 requests and both timed out. The confirmation rate is 0.

User 7 made 3 requests and all were confirmed. The confirmation rate is 1.

User 2 made 2 requests where one was confirmed and the other timed out. The confirmation rate is 1 / 2 = 0.5.

ANSWER:

```
SELECT A.USER_ID,ROUND (SUM(CASE WHEN action='confirmed' THEN 1
                                 ELSE 0 END)/COUNT (*),2) AS confirmation_rate
FROM Signups A
LEFT JOIN Confirmations B
ON A.user_id =B.user_id
GROUP BY A.user_id;
```

1867. Orders With Maximum Quantity Above Average

Table: OrdersDetails.

Column_Name	Type
order_id	Int
product_id	Int
quantity	Int

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

A single order is represented as multiple rows, one row for each product in the order.

Each row of this table contains the quantity ordered of the product product_id in the order order_id.

You are running an e-commerce site that is looking for **imbalanced orders**. An **imbalanced order** is one whose **maximum** quantity is **strictly greater** than the **average** quantity of **every order (including itself)**.

The **average** quantity of an order is calculated as (total quantity of all products in the order) / (number of different products in the order).

The **maximum** quantity of an order is the highest quantity of any single product in the order.

Write a solution to find the order_id of all **imbalanced orders**.

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

The result format is in the following example.

Example 1:

Input:

OrdersDetails table:

order_id	product_id	quantity
1	1	12
1	2	10
1	3	15
2	1	8
2	4	4
2	5	6
3	3	5
3	4	18
4	5	2
4	6	8
5	7	9
5	8	9
3	9	20
2	9	4

Output:

order_id
1
3

Explanation:

The average quantity of each order is:

- order_id=1: (12+10+15)/3 = 12.3333333

- order_id=2: (8+4+6+4)/4 = 5.5

- order_id=3: (5+18+20)/3 = 14.333333

- order_id=4: (2+8)/2 = 5

- order_id=5: (9+9)/2 = 9

The maximum quantity of each order is:

- order_id=1: max(12, 10, 15) = 15

- order_id=2: max(8, 4, 6, 4) = 8

- order_id=3: max(5, 18, 20) = 20

- order_id=4: max(2, 8) = 8

- order_id=5: max(9, 9) = 9

Orders 1 and 3 are imbalanced because they have a maximum quantity that exceeds the average quantity of every order.

ANSWER:

```
SELECT order_id
FROM OrdersDetails
GROUP BY order_id
HAVING MAX(quantity) > (SELECT MAX(avg_quantity)
                        FROM (SELECT AVG(quantity) as avg_quantity
                              FROM OrdersDetails
                              GROUP BY order_id) t);
```

1148. Article views I

Table: Views

Column Name	Type
article_id	int
author_id	int
viewer_id	int
view_date	date

There is no primary key for this table, it may have duplicate rows.

Each row of this table indicates that some viewer viewed an article (written by some author) on some date.

Note that equal author_id and viewer_id indicate the same person.

Write an SQL query to find all the authors that viewed at least one of their own articles.

Return the result table sorted by id in ascending order.

The query result format is in the following example.

Example 1:

Input:

Views Table:

article_id	author_id	viewer_id	view_date
1	3	5	2019-08-01
1	3	6	2019-08-02
2	7	7	2019-08-01
2	7	6	2019-08-02
4	7	1	2019-07-22
3	4	4	2019-07-21
3	4	4	2019-07-21

Input:

Id
4
7

ANSWER:

```
SELECT DISTINCT author_id AS id

FROM Views

WHERE author_id = viewer_id

ORDER BY id;
```

1082. Sales analysis I

Table: Product

Column name	Type
product_id	int
product_name	varchar
unit_price	int

product_id is the primary key of this table.

Table: Sales

Column Name	Type
seller_id	int
product_id	int
buyer_id	int
sales_date	date
quantity	int
price	int

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

product_id is a foreign key to Product table.

Write an SQL query that reports the best seller by total sales price, If there is a tie, report them all.

The query result format is in the following example:

Product table:

Product_id	Product_name	Unit_price
1	S8	1000
2	G4	800
3	iphone	1400

Sales table:

seller_id	product_id	buyer_id	sale_date	quantity	price
1	1	1	2019-01-21	2	2000
1	2	2	2019-02-17	1	800
2	2	3	2019-06-02	1	800
3	3	4	2019-05-13	2	2800

Result table:

seller_id
1
3

ANSWER:

```
SELECT seller_id
FROM Sales
GROUP BY seller_id
HAVING SUM(price) = (SELECT SUM price
                     FROM Sales
                     GROUP BY seller_id
                     ORDER BY SUM(price) DESC
                     LIMIT 1);
```

1511. Customer Order Frequency

Table: Customers

Column Name	Type
customer_id	int
name	varchar
country	varchar

customer_id is the primary key for this table.
This table contains information of the customers in the company.

Table: Product

Column Name	Type
product_id	int
description	varchar
price	int

product_id is the primary key for this table.
This table contains information of the products in the company.
price is the product cost.

Table: Orders

Column Name	Type
order_id	int
customer_id	int
product_id	int
order_date	date
quantity	int

order_id is the primary key for this table.
This table contains information on customer orders.
customer_id is the id of the customer who bought "quantity" products with id "product_id".
Order_date is the date in format ('YYYY-MM-DD') when the order was shipped.

Write an SQL query to report the customer_id and customer_name of customers who have spent at least $100 in each month of June and July 2020.

Return the result table in any order.

The query result format is in the following example.

Customers:

Customer_ID	Name	Country
1	Winston	USA
2	Jonathan	Peru
3	Moustafa	Egypt

Product:

Product_ID	Description	Price
10	LC Phone	300
20	LC T-Shirt	10
30	LC Book	45
40	LC Keychain	2

Orders:

Order_ID	Customer_ID	Product_ID	Order_date	Quantity
1	1	10	2020-06-10	1
2	1	20	2020-07-01	1
3	1	30	2020-07-08	2
4	2	10	2020-06-15	2
5	2	40	2020-07-01	10
6	3	20	2020-06-24	2
7	3	30	2020-06-25	2

9	3	30	2020-05-08	3

Result Table:

Customer_ID	Name
1	Winston

Winston spent $300 (300 * 1) in June and $100 (10 * 1 + 45 * 2) in July 2020.
Jonathan spent $600 (300 * 2) in June and $20 (2 * 10) in July 2020.
Moustafa spent $110 (10 * 2 + 45 * 2) in June and $0 in July 2020.

ANSWER:

```
WITH cte AS

(SELECT o. customer_id, YEAR(o.order_date) AS

year, MONTH(o.order_date) AS month,

SUM(o.quantity * p.price) AS spend

FROM Orders o

LEFT JOIN Product p

ON o.product_id = p.product_id

WHERE YEAR(o.order_date) = 2020

AND MONTH(o.order_date) IN (6,7)

GROUP BY o. customer_id, YEAR(o.order_date),

MONTH(o.order_date)),

cte2 AS

SELECT customer_id

FROM cte

WHERE spend >= 100

GROUP BY customer_id

HAVING COUNT(Month = 2)


SELECT c.customer_id, Cu.name

FROM cte2 c

LEFT JOIN Customers Cu

ON c.customer_id = Cu.customer_id
```

1729. Find Followers Count

Column Name	Type
User_id	int
Follower_id	int

(user_id, follower_id) is the primary key for this table.

This table contains the IDs of a user and a follower in a social media app where the follower follows the user.

Write an SQL query that will, for each user, return the number of followers.

Return the result table ordered by user_id.

The query result format is in the following example:

Followers Table:

User_ID	Followers_count
0	1
1	0
2	0
2	1

Result Table:

User_ID	Followers_count
0	1
1	1
2	2

Explanation

The followers of 0 are {1}

The followers of 1 are {0}

The followers of 2 are {0,1}

ANSWER:

SELECT user_id, COUNT(DISTINCT follower_id AS

followers_count

FROM Followers

GROUP BY user_id

ORDER BY;

1633. Percentage of Users Attended a Contest

USER:

Column Name	Type
User_id	int
User_name	varchar

user_id is the primary key for this table.
Each row of this table contains the name and the id of a user.

REGISTER:

Column Name	Type
Contest_id	int
User_id	int

(contest_id, user_id) is the primary key for this table.
Each row of this table contains the id of a user and the contest they registered into.

Write an SQL query to find the percentage of the users registered in each contest rounded to **two decimals**.

Return the result table ordered by percentage in **descending order**. In case of a tie, order it by contest_id in **ascending order**.

The query result format is in the following example.

Example 1:

Input:
Users table:

User_ID	User_Name
6	Alice
2	Bob
7	Alex

Register table:

Contest_ID	User_ID
215	6
209	2
208	2
210	6

208	6
209	7
209	6
215	7
208	7
210	2
207	2
210	7

OUTPUT:

Contest_ID	Percentage
208	100
209	100
210	100
215	66.67
207	33.33

Explanation:
All the users registered in contests 208, 209, and 210. The percentage is 100% and we sort them in the answer table by contest_id in ascending order.
Alice and Alex registered in contest 215 and the percentage is ((2/3) * 100) = 66.67%
Bob registered in contest 207 and the percentage is ((1/3) * 100) = 33.33%

ANSWER:

```
SELECT contest_id, ROUND COUNT(DISTINCT user_id

/ (SELECT COUNT(user_id) FROM Users) * 100,2

AS percentage

FROM Register

GROUP BY contest_id

ORDER BY percentage DESC, contest_id
```