

Q 101.

Table: UserActivity

Column Name	Type
username	varchar
activity	varchar
startDate	Date
endDate	Date

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

This table contains information about the activity performed by each user in a period of time.

A person with a username performed an activity from startDate to endDate.

Write an SQL query to show the second most recent activity of each user.

If the user only has one activity, return that one. A user cannot perform more than one activity at the same time.

Return the result table in any order.

The query result format is in the following example.

Input:

UserActivity table:

username	activity	startDate	endDate
Alice	Travel	2020-02-12	2020-02-20
Alice	Dancing	2020-02-21	2020-02-23
Alice	Travel	2020-02-24	2020-02-28
Bob	Travel	2020-02-11	2020-02-18

Output:

username	activity	startDate	endDate
Alice	Dancing	2020-02-21	2020-02-23
Bob	Travel	2020-02-11	2020-02-18

Explanation:

The most recent activity of Alice is Travel from 2020-02-24 to 2020-02-28, before that she was dancing from 2020-02-21 to 2020-02-23.

Bob only has one record, we just take that one.

Q102.

Table: UserActivity

Column Name	Type
username	varchar
activity	varchar
startDate	Date
endDate	Date

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

This table contains information about the activity performed by each user in a period of time.

A person with a username performed an activity from startDate to endDate.

Write an SQL query to show the second most recent activity of each user.

If the user only has one activity, return that one. A user cannot perform more than one activity at the same time.

Return the result table in any order.

The query result format is in the following example.

Input:

UserActivity table:

username	activity	startDate	endDate
Alice	Travel	2020-02-12	2020-02-20
Alice	Dancing	2020-02-21	2020-02-23
Alice	Travel	2020-02-24	2020-02-28
Bob	Travel	2020-02-11	2020-02-18

Output:

username	activity	startDate	endDate
Alice	Dancing	2020-02-21	2020-02-23
Bob	Travel	2020-02-11	2020-02-18

Explanation:

The most recent activity of Alice is Travel from 2020-02-24 to 2020-02-28, before that she was dancing from 2020-02-21 to 2020-02-23.

Bob only has one record, we just take that one.

Q103.

Query the Name of any student in STUDENTS who scored higher than 75 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.

Input Format

The STUDENTS table is described as follows:

<i>Column</i>	<i>Type</i>
<i>ID</i>	<i>Integer</i>
<i>Name</i>	<i>String</i>
<i>Marks</i>	<i>Integer</i>

The Name column only contains uppercase (A-Z) and lowercase (a-z) letters.

Sample Input

<i>ID</i>	<i>Name</i>	<i>Marks</i>
<i>1</i>	<i>Ashley</i>	<i>81</i>
<i>2</i>	<i>Samantha</i>	<i>75</i>
<i>4</i>	<i>Julia</i>	<i>76</i>
<i>3</i>	<i>Belvet</i>	<i>84</i>

Sample Output

Ashley
Julia
Belvet

Explanation

Only Ashley, Julia, and Belvet have Marks > 75 . If you look at the last three characters of each of their names, there are no duplicates and 'ley' < 'lia' < 'vet'.

Q104.

Write a query that prints a list of employee names (i.e.: the name attribute) for employees in Employee having a salary greater than \$2000 per month who have been employees for less than 10 months. Sort your result by ascending employee_id.

Input Format

The Employee table containing employee data for a company is described as follows:

Column	Type
employee_id	Integer
name	String
months	Integer
salary	Integer

where employee_id is an employee's ID number, name is their name, months is the total number of months they've been working for the company, and salary is the their monthly salary.

Sample Input

employee_id	name	months	salary
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

Sample Output

Angela

Michael

Todd

Joe

Explanation

Angela has been an employee for 1 month and earns \$3443 per month.

Michael has been an employee for 6 months and earns \$2017 per month.

Todd has been an employee for 5 months and earns \$3396 per month.

Joe has been an employee for 9 months and earns \$3573 per month.

We order our output by ascending employee_id.

Q105

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

Output one of the following statements for each record in the table:

- Equilateral: It's a triangle with sides of equal length.
- Isosceles: It's a triangle with sides of equal length.
- Scalene: It's a triangle with sides of differing lengths.
- Not A Triangle: The given values of A, B, and C don't form a triangle.

Input Format

The TRIANGLES table is described as follows:

<i>Column</i>	<i>Type</i>
<i>A</i>	<i>Integer</i>
<i>B</i>	<i>Integer</i>
<i>C</i>	<i>Integer</i>

Each row in the table denotes the lengths of each of a triangle's three sides.

Sample Input

<i>A</i>	<i>B</i>	<i>C</i>
<i>20</i>	<i>20</i>	<i>23</i>
<i>20</i>	<i>20</i>	<i>20</i>
<i>20</i>	<i>21</i>	<i>22</i>
<i>13</i>	<i>14</i>	<i>30</i>

Sample Output

Isosceles

Equilateral

Scalene

Not A Triangle

Explanation

Values in the tuple(20,20,23) form an Isosceles triangle, because $A \equiv B$.

Values in the tuple(20,20,20) form an Equilateral triangle, because $A \equiv B \equiv C$. Values in the tuple(20,21,22) form a Scalene triangle, because $A \neq B \neq C$.

Values in the tuple (13,14,30) cannot form a triangle because the combined value of sides A and B is not larger than that of side C .

Q106.

Samantha was tasked with calculating the average monthly salaries for all employees in the EMPLOYEES table, but did not realise her keyboard's 0 key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeros removed), and the actual average salary.

Write a query calculating the amount of error (i.e.: actual - miscalculated average monthly salaries), and round it up to the next integer.

Input Format

The EMPLOYEES table is described as follows:

<i>Column</i>	<i>Type</i>
<i>ID</i>	<i>Integer</i>
<i>Name</i>	<i>String</i>
<i>Salary</i>	<i>Integer</i>

Note: Salary is per month.

Constraints

$1000 < \text{salary} < 10^5$

Sample Input

<i>ID</i>	<i>Name</i>	<i>Salary</i>
<i>1</i>	<i>Kristeen</i>	<i>1420</i>
<i>2</i>	<i>Ashley</i>	<i>2006</i>
<i>3</i>	<i>Julia</i>	<i>2210</i>
<i>4</i>	<i>Maria</i>	<i>3000</i>

Sample Output

2061

Explanation

The table below shows the salaries without zeros as they were entered by Samantha:

<i>ID</i>	<i>Name</i>	<i>Salary</i>
1	Kristeen	142
2	Ashley	26
3	Julia	221
4	Maria	3

Samantha computes an average salary of 98.00 . The actual average salary is 2159.00. The resulting error between the two calculations is $2159.00 - 98.00 = 2061.00$. Since it is equal to the integer 2061, it does not get rounded up.

Q107.

We define an employee's total earnings to be their monthly salary * months 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 2 space-separated integers.

Level - Easy

Hint - Use Aggregation functions

Input Format

The Employee table containing employee data for a company is described as follows:

Column	Type
employee_id	Integer
name	String
months	Integer
salary	Integer

where employee_id is an employee's ID number, name is their name, months is the total number of months they've been working for the company, and salary is their monthly salary.

Sample Input

employee_id	name	months	salary
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

Sample Output
69952 1

Explanation:

The table and earnings data is depicted in the following diagram:

employee_id	name	months	salary	earnings
12228	Rose	15	1968	29520
33645	Angela	1	3443	3443
45692	Frank	17	1608	27336
56118	Patrick	7	1345	9415
59725	Lisa	11	2330	25630
74197	Kimberly	16	4372	69952
78454	Bonnie	8	1771	14168
83565	Michael	6	2017	12102
98607	Todd	5	3396	16980
99989	Joe	9	3573	32157

The maximum earnings value is 69952. The only employee with earnings= 69952 is Kimberly, so we print the maximum earnings value (69952) and a count of the number of employees who have earned \$69952 (which is 1) as two space-separated values.

Q108.

Generate the following two result sets:

1. 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). For example: AnActorName(A), ADoctorName(D), AProfessorName(P), and ASingerName(S).

Query the number of occurrences of each occupation in OCCUPATIONS. Sort the occurrences in ascending order, and output them in the following format:

Level - Medium

There are a total of [occupation_count] [occupation]s.

2. 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.

Input Format

The OCCUPATIONS table is described as follows:

<i>Column</i>	<i>Type</i>
<i>Name</i>	<i>String</i>
<i>Occupation</i>	<i>String</i>

Occupation will only contain one of the following values: Doctor, Professor, Singer or Actor.

Sample Input

An OCCUPATIONS table that contains the following records:

<i>Name</i>	<i>Occupation</i>
<i>Samantha</i>	<i>Doctor</i>
<i>Julia</i>	<i>Actor</i>
<i>Maria</i>	<i>Actor</i>
<i>Meera</i>	<i>Singer</i>
<i>Ashely</i>	<i>Professor</i>
<i>Ketty</i>	<i>Professor</i>
<i>Christeen</i>	<i>Professor</i>
<i>Jane</i>	<i>Actor</i>
<i>Jenny</i>	<i>Doctor</i>
<i>Priya</i>	<i>Singer</i>

Sample Output

Ashely(P)

Christeen(P)

Jane(A)

Jenny(D)

Julia(A)

Ketty(P)

Maria(A)

Meera(S)

Priya(S)

Samantha(D)

There are a total of 2 doctors.

There are a total of 2 singers.

There are a total of 3 actors.

There are a total of 3 professors.

Hint -

The results of the first query are formatted to the problem description's specifications.

The results of the second query are ascendingly ordered first by number of names corresponding to each profession (2<= 2<=3<=3), and then alphabetically by profession (doctor <= singer , and actor <= professor).

Q109 .

Pivot 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.

Input Format

The OCCUPATIONS table is described as follows:

<i>Column</i>	<i>Type</i>
<i>Name</i>	<i>String</i>
<i>Occupation</i>	<i>String</i>

Occupation will only contain one of the following values: Doctor, Professor, Singer or Actor.

Sample Input

<i>Name</i>	<i>Occupation</i>
<i>Samantha</i>	<i>Doctor</i>
<i>Julia</i>	<i>Actor</i>
<i>Maria</i>	<i>Actor</i>
<i>Meera</i>	<i>Singer</i>
<i>Ashely</i>	<i>Professor</i>
<i>Ketty</i>	<i>Professor</i>
<i>Christeen</i>	<i>Professor</i>
<i>Jane</i>	<i>Actor</i>
<i>Jenny</i>	<i>Doctor</i>
<i>Priya</i>	<i>Singer</i>

Sample Output

Jenny Ashley Meera Jane
Samantha Christeen Priya Julia
NULL Ketty NULL Maria

Hint -

The first column is an alphabetically ordered list of Doctor names.

The second column is an alphabetically ordered list of Professor names.

The third column is an alphabetically ordered list of Singer names.

The fourth column is an alphabetically ordered list of Actor names.

The empty cell data for columns with less than the maximum number of names per occupation (in this case, the Professor and Actor columns) are filled with NULL values.

Q110.

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.

<i>Column</i>	<i>Type</i>
<i>N</i>	<i>Integer</i>
<i>P</i>	<i>Integer</i>

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.

Sample Input

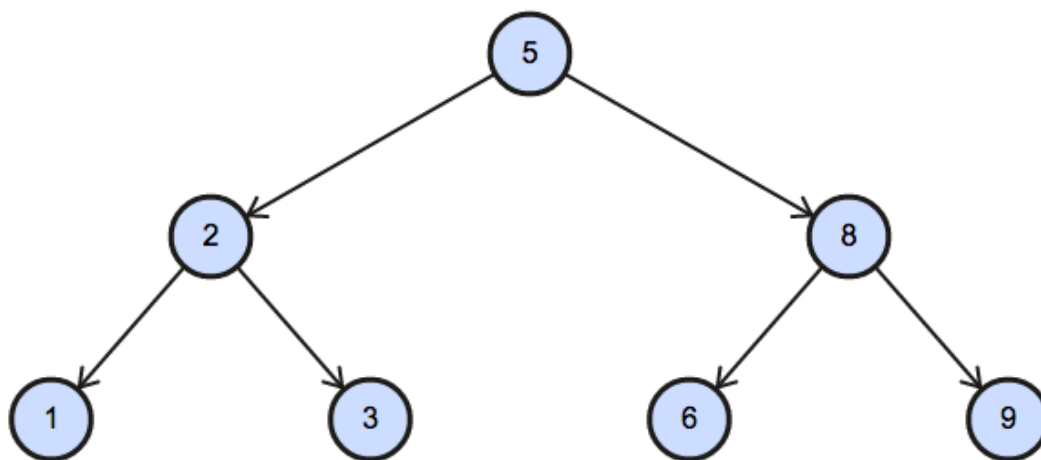
<i>N</i>	<i>P</i>
<i>1</i>	<i>2</i>
<i>3</i>	<i>2</i>
<i>6</i>	<i>8</i>
<i>9</i>	<i>8</i>
<i>2</i>	<i>5</i>
<i>8</i>	<i>5</i>
<i>5</i>	<i>null</i>

Sample Output

1 Leaf
2 Inner
3 Leaf
5 Root
6 Leaf
8 Inner
9 Leaf

Explanation

The Binary Tree below illustrates the sample:



Q111 .

Amber's conglomerate corporation just acquired some new companies. Each of the companies

Founder



Lead Manager



Senior Manager



Manager



Employee

follows this hierarchy:

Given the table schemas below, write a query to print the company_code, founder name, total number of lead managers, total number of senior managers, total number of managers, and total number of employees. Order your output by ascending company_code.

Level - Medium

Note:

- The tables may contain duplicate records.

- The company_code is string, so the sorting should not be numeric. For example, if the company_codes are C_1, C_2, and C_10, then the ascending company_codes will be C_1, C_10, and C_2.

Input Format

The following tables contain company data:

- Company: The company_code is the code of the company and founder is the founder of the

Column	Type
company_code	String
founder	String

company.

- Lead_Manager: The lead_manager_code is the code of the lead manager, and the

Column	Type
lead_manager_code	String
company_code	String

company_code is the code of the working company.

- Senior_Manager: The senior_manager_code is the code of the senior manager, the lead_manager_code is the code of its lead manager, and the company_code is the code of the

Column	Type
senior_manager_code	String
lead_manager_code	String
company_code	String

working company.

- Manager: The manager_code is the code of the manager, the senior_manager_code is the code of its senior manager, the lead_manager_code is the code of its lead manager, and the company_code is the code of the working company.

Column	Type
manager_code	String
senior_manager_code	String
lead_manager_code	String
company_code	String

- Employee: The employee_code is the code of the employee, the manager_code is the code of its manager, the senior_manager_code is the code of its senior manager, the

lead_manager_code is the code of its lead manager, and the company_code is the code of the

Column	Type
employee_code	String
manager_code	String
senior_manager_code	String
lead_manager_code	String
company_code	String

working company.

Sample Input

company_code	founder
C1	Monika
C2	Samantha

Company Table:

lead_manager_code	company_code
LM1	C1
LM2	C2

Lead_Manager Table:

Senior_Manager Table:

senior_manager_code	lead_manager_code	company_code
SM1	LM1	C1
SM2	LM1	C1
SM3	LM2	C2

Manager Table:

manager_code	senior_manager_code	lead_manager_code	company_code
M1	SM1	LM1	C1
M2	SM3	LM2	C2
M3	SM3	LM2	C2

Employee Table:

employee_code	manager_code	senior_manager_code	lead_manager_code	company_code
E1	M1	SM1	LM1	C1
E2	M1	SM1	LM1	C1
E3	M2	SM3	LM2	C2
E4	M3	SM3	LM2	C2

Sample Output

C1 Monika 1 2 1 2

C2 Samantha 1 1 2 2

Hint -

In company C1, the only lead manager is LM1. There are two senior managers, SM1 and SM2, under LM1. There is one manager, M1, under senior manager SM1. There are two employees, E1 and E2, under manager M1.

In company C2, the only lead manager is LM2. There is one senior manager, SM3, under LM2. There are two managers, M2 and M3, under senior manager SM3. There is one employee, E3, under manager M2, and another employee, E4, under manager, M3.

Q112.

Write a query to print all prime numbers less than or equal to 1000. Print your result on a single line, and use the ampersand (&) character as your separator (instead of a space).

For example, the output for all prime numbers ≤ 10 would be: 2&3&5&7

Hint - Firstly, select L Prime_Number from (select Level L from Dual connect Level \leq 1000) and then do the same thing to create Level M, and then filter by $M \leq L$ and then group by L having count(case when L/M = trunc(L/M) then 'Y' end) = 2 order by L

Q113.

P(R) represents a pattern drawn by Julia in R rows. The following pattern represents P(5):

```
*
* *
* * *
* * * *
* * * * *
```

Write a query to print the pattern P(20).

Level - Easy

Source - Hackerrank

Hint - Use SYS_CONNECT_BY_PATH(NULL, '*') FROM DUAL

Q114.

P(R) represents a pattern drawn by Julia in R rows. The following pattern represents P(5):

```
* * * * *
* * * *
* * *
* *
*
```

Write a query to print the pattern P(20).

Level - Easy

Hint - Use SYS_CONNECT_BY_PATH(NULL, '*') FROM DUAL

Q116. You are given a table, Functions, containing two columns: X and Y.

Column	Type
X	Integer
Y	Integer

Two pairs (X1, Y1) and (X2, Y2) are said to be symmetric pairs if $X1 = Y2$ and $X2 = Y1$.

Write a query to output all such symmetric pairs in ascending order by the value of X. List the rows such that $X1 \leq Y1$.

.

Sample Input

X	Y
20	20
20	20
20	21
23	22
22	23
21	20

Sample Output

20 20

20 21

22 23

Q115.

Query the Name of any student in STUDENTS who scored higher than 75 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.

Level - Easy

Hint - Use Like

Input Format

<i>Column</i>	<i>Type</i>
<i>ID</i>	<i>Integer</i>
<i>Name</i>	<i>String</i>
<i>Marks</i>	<i>Integer</i>

The STUDENTS table is described as follows:

The Name column only contains uppercase (A-Z) and lowercase (a-z) letters.

Sample Input

<i>ID</i>	<i>Name</i>	<i>Marks</i>
<i>1</i>	<i>Ashley</i>	<i>81</i>
<i>2</i>	<i>Samantha</i>	<i>75</i>
<i>4</i>	<i>Julia</i>	<i>76</i>
<i>3</i>	<i>Belvet</i>	<i>84</i>

Sample Output

Ashley

Julia

Belvet

Explanation

Only Ashley, Julia, and Belvet have Marks > 75 . If you look at the last three characters of each of their names, there are no duplicates and 'ley' < 'lia' < 'vet'.

Q116.

Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in alphabetical order.

Level - Easy

Hint - Use ORDER BY

Input Format

The Employee table containing employee data for a company is described as follows:

Column	Type
employee_id	Integer
name	String
months	Integer
salary	Integer

where employee_id is an employee's ID number, name is their name, months is the total number of months they've been working for the company, and salary is their monthly salary.

Sample Input

employee_id	name	months	salary
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

Sample Output

Angela

Bonnie

Frank

Joe

Kimberly

Lisa

Michael

Patrick

Rose

Todd

Q117. Write a query that prints a list of employee names (i.e.: the name attribute) for employees in Employee having a salary greater than \$2000 per month who have been employees for less than 10 months. Sort your result by ascending employee_id.

Level - Easy

Hint - Use Ascending

Input Format

The Employee table containing employee data for a company is described as follows:

Column	Type
employee_id	Integer
name	String
months	Integer
salary	Integer

where employee_id is an employee's ID number, name is their name, months is the total number of months they've been working for the company, and salary is the their monthly salary.

Sample Input

employee_id	name	months	salary
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

Sample Output

Angela

Michael

Todd

Joe

Explanation

Angela has been an employee for 1 month and earns \$3443 per month.

Michael has been an employee for 6 months and earns \$2017 per month.

Todd has been an employee for 5 months and earns \$3396 per month.
Joe has been an employee for 9 months and earns \$3573 per month.
We order our output by ascending employee_id.

Q118. Write a query identifying the type of each record in the TRIANGLES table using its three side lengths. Output one of the following statements for each record in the table:

- Equilateral: It's a triangle with sides of equal length.
- Isosceles: It's a triangle with sides of equal length.
- Scalene: It's a triangle with sides of differing lengths.
- Not A Triangle: The given values of A, B, and C don't form a triangle.

Level - Easy

Hint - Use predefined functions for calculation.

Input Format

The TRIANGLES table is described as follows:

<i>Column</i>	<i>Type</i>
<i>A</i>	<i>Integer</i>
<i>B</i>	<i>Integer</i>
<i>C</i>	<i>Integer</i>

Each row in the table denotes the lengths of each of a triangle's three sides.

Sample Input

<i>A</i>	<i>B</i>	<i>C</i>
<i>20</i>	<i>20</i>	<i>23</i>
<i>20</i>	<i>20</i>	<i>20</i>
<i>20</i>	<i>21</i>	<i>22</i>
<i>13</i>	<i>14</i>	<i>30</i>

Sample Output

Isosceles

Equilateral

Scalene

Not A Triangle

Explanation

Values in the tuple(20,20,23) form an Isosceles triangle, because $A \equiv B$.

Values in the tuple(20,20,20) form an Equilateral triangle, because $A \equiv B \equiv C$. Values in the tuple(20,21,22) form a Scalene triangle, because $A \neq B \neq C$.

Values in the tuple (13,14,30) cannot form a triangle because the combined value of sides A and B is not larger than that of side C.

Q119. Assume you are given the table below containing information on user transactions for particular products. Write a query to obtain the year-on-year growth rate for the total spend of each product for each year.

Output the year (in ascending order) partitioned by product id, current year's spend, previous year's spend and year-on-year growth rate (percentage rounded to 2 decimal places).

Level - Hard

Hint - Use extract function

user_transactions Table:

Column Name	Type
transaction_id	integer
product_id	integer
spend	decimal
transaction_date	datetime

user_transactions Example Input:

transaction_id	product_id	spend	transaction_date
1341	123424	1500.60	12/31/2019 12:00:00
1423	123424	1000.20	12/31/2020 12:00:00
1623	123424	1246.44	12/31/2021 12:00:00
1322	123424	2145.32	12/31/2022 12:00:00

Example Output:

y	product_id	curr_year_spend	prev_year_spend	yoy_rate
2	123424	1500.60		
2	123424	1000.20	1500.60	-33.35
2	123424	1246.44	1000.20	24.62
2	123424	2145.32	1246.44	72.12

Q120. Amazon wants to maximise the number of items it can stock in a 500,000 square feet warehouse. It wants to stock as many prime items as possible, and afterwards use the remaining square footage to stock the most number of non-prime items.

Write a SQL query to find the number of prime and non-prime items that can be stored in the 500,000 square feet warehouse. Output the item type and number of items to be stocked.

Hint - create a table containing a summary of the necessary fields such as item type ('prime_eligible', 'not_prime'), SUM of square footage, and COUNT of items grouped by the item type.

inventory table:

Column Name	Type
item_id	integer
item_type	string
item_category	string

square_footage	decimal
----------------	---------

inventory Example Input:

item_id	item_type	item_category	square_footage
1374	prime_eligible	mini refrigerator	68.00
4245	not_prime	standing lamp	26.40
2452	prime_eligible	television	85.00
3255	not_prime	side table	22.60
1672	prime_eligible	laptop	8.50

Example Output:

item_type	item_count
prime_eligible	9285
not_prime	6

Q121. Assume you have the table below containing information on Facebook user actions. Write a query to obtain the active user retention in July 2022. Output the month (in numerical format 1, 2, 3) and the number of monthly active users (MAUs).

Hint: An active user is a user who has user action ("sign-in", "like", or "comment") in the current month and last month.

Hint- Use generic correlated subquery

user_actions Table:

Column Name	Type
user_id	integer
event_id	integer
event_type	string ("sign-in", "like", "comment")
event_date	datetime

user_actionsExample Input:

user_id	event_id	event_type	event_date
445	7765	sign-in	05/31/2022 12:00:00
742	6458	sign-in	06/03/2022 12:00:00
445	3634	like	06/05/2022 12:00:00
742	1374	comment	06/05/2022 12:00:00
648	3124	like	06/18/2022 12:00:00

Example Output for June 2022:

month	monthly_active_users
6	1

Q122. Google's marketing team is making a Superbowl commercial and needs a simple statistic to put on their TV ad: the median number of searches a person made last year. However, at Google scale, querying the 2 trillion searches is too costly. Luckily, you have access to the summary table which tells you the number of searches made last year and how many Google users fall into that bucket.

Write a query to report the median of searches made by a user. Round the median to one decimal point.

Hint- Write a subquery or common table expression (CTE) to generate a series of data (that's keyword for column) starting at the first search and ending at some point with an optional incremental value.

search_frequency Table:

Column Name	Type
searches	integer
num_users	integer

search_frequency Example Input:

searches	num_users
1	2
2	2
3	3
4	1

Example Output:

median
2.5

Q123. Write a query to update the Facebook advertiser's status using the daily_pay table. Advertiser is a two-column table containing the user id and their payment status based on the last payment and daily_pay table has current information about their payment. Only advertisers who paid will show up in this table.

Output the user id and current payment status sorted by the user id.

Hint- Query the daily_pay table and check through the advertisers in this table. .

advertiser Table:

Column Name	Type
user_id	string
status	string

advertiser Example Input:

user_id	status
bing	NEW
yahoo	NEW
alibaba	EXISTING

daily_pay Table:

Column Name	Type
user_id	string
paid	decimal

daily_pay Example Input:

user_id	paid
yahoo	45.00
alibaba	100.00
target	13.00

Definition of advertiser status:

- New: users registered and made their first payment.
- Existing: users who paid previously and recently made a current payment.
- Churn: users who paid previously, but have yet to make any recent payment.
- Resurrect: users who did not pay recently but may have made a previous payment and have made payment again recently.

Example Output:

user_id	new_status
bing	CHURN
yahoo	EXISTING
alibaba	EXISTING

Bing's updated status is CHURN because no payment was made in the daily_pay table whereas Yahoo which made a payment is updated as EXISTING.

The dataset you are querying against may have different input & output - this is just an example!

Read this before proceeding to solve the question

For better understanding of the advertiser's status, we're sharing with you a table of possible transitions based on the payment status.

#	Start	End	Condition
1	NEW	EXISTING	Paid on day T
2	NEW	CHURN	No pay on day T
3	EXISTING	EXISTING	Paid on day T
4	EXISTING	CHURN	No pay on day T
5	CHURN	RESURRECT	Paid on day T
6	CHURN	CHURN	No pay on day T
7	RESURRECT	EXISTING	Paid on day T
8	RESURRECT	CHURN	No pay on day T

1. Row 2, 4, 6, 8: As long as the user has not paid on day T, the end status is updated to CHURN regardless of the previous status.
2. Row 1, 3, 5, 7: When the user paid on day T, the end status is updated to either EXISTING or RESURRECT, depending on their previous state. RESURRECT is only possible when the previous state is CHURN. When the previous state is anything else, the status is updated to EXISTING.

Q124. Amazon Web Services (AWS) is powered by fleets of servers. Senior management has requested data-driven solutions to optimise server usage.

Write a query that calculates the total time that the fleet of servers was running. The output should be in units of full days.

Level - Hard

Hint-

1. Calculate individual uptimes
2. Sum those up to obtain the uptime of the whole fleet, keeping in mind that the result must be output in units of full days

Assumptions:

- Each server might start and stop several times.
- The total time in which the server fleet is running can be calculated as the sum of each server's uptime.

server_utilization Table:

Column Name	Type
server_id	integer
status_time	timestamp
session_status	string

server_utilization Example Input:

server_id	status_time	session_status
1	08/02/2022 10:00:00	start
1	08/04/2022 10:00:00	stop
2	08/17/2022 10:00:00	start
2	08/24/2022 10:00:00	stop

Example Output:

total_uptime_days
21

Q125. Sometimes, payment transactions are repeated by accident; it could be due to user error, API failure or a retry error that causes a credit card to be charged twice.

Using the transactions table, identify any payments made at the same merchant with the same credit card for the same amount within 10 minutes of each other. Count such repeated payments.

Level - Hard

Hint- Use Partition and order by

Assumptions:

- The first transaction of such payments should not be counted as a repeated payment. This means, if there are two transactions performed by a merchant with the same credit card and for the same amount within 10 minutes, there will only be 1 repeated payment.

transactions Table:

Column Name	Type
transaction_id	integer
merchant_id	integer
credit_card_id	integer
amount	integer
transaction_timestamp	datetime

transactions Example Input:

transaction_id	merchant_id	credit_card_id	amount	transaction_timestamp
1	101	1	100	09/25/2022 12:00:00
2	101	1	100	09/25/2022 12:08:00
3	101	1	100	09/25/2022 12:28:00
4	102	2	300	09/25/2022 12:00:00
6	102	2	400	09/25/2022 14:00:00

Example Output:

payment_count
1

Q126. DoorDash's Growth Team is trying to make sure new users (those who are making orders in their first 14 days) have a great experience on all their orders in their 2 weeks on the platform.

Unfortunately, many deliveries are being messed up because:

- the orders are being completed incorrectly (missing items, wrong order, etc.)
- the orders aren't being received (wrong address, wrong drop off spot)
- the orders are being delivered late (the actual delivery time is 30 minutes later than when the order was placed). Note that the estimated_delivery_timestamp is automatically set to 30 minutes after the order_timestamp.

Hint- Use Where Clause and joins

Write a query to find the bad experience rate in the first 14 days for new users who signed up in June 2022. Output the percentage of bad experience rounded to 2 decimal places.

orders Table:

Column Name	Type
order_id	integer
customer_id	integer
trip_id	integer
status	string ('completed successfully', 'completed incorrectly', 'never received')
order_timestamp	timestamp

orders Example Input:

order_id	customer_id	trip_id	status	order_timestamp
727424	8472	100463	completed successfully	06/05/2022 09:12:00
242513	2341	100482	completed incorrectly	06/05/2022 14:40:00
141367	1314	100362	completed incorrectly	06/07/2022 15:03:00
582193	5421	100657	never_received	07/07/2022 15:22:00
253613	1314	100213	completed successfully	06/12/2022 13:43:00

trips Table:

Column Name	Type
dasher_id	integer
trip_id	integer
estimated_delivery_timestamp	timestamp
actual_delivery_timestamp	timestamp

trips Example Input:

dasher_id	trip_id	estimated_delivery_timestamp	actual_delivery_timestamp
101	100463	06/05/2022 09:42:00	06/05/2022 09:38:00
102	100482	06/05/2022 15:10:00	06/05/2022 15:46:00
101	100362	06/07/2022 15:33:00	06/07/2022 16:45:00
102	100657	07/07/2022 15:52:00	-
103	100213	06/12/2022 14:13:00	06/12/2022 14:10:00

customers Table:

Column Name	Type
customer_id	integer
signup_timestamp	timestamp

customers Example Input:

customer_id	signup_timestamp
8472	05/30/2022 00:00:00
2341	06/01/2022 00:00:00
1314	06/03/2022 00:00:00
1435	06/05/2022 00:00:00
5421	06/07/2022 00:00:00

Example Output:

bad_experience_pct
75.00

Q127.

Table: Scores

Column Name	Type
player_name	varchar
gender	varchar
day	date
score_points	int

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

A competition is held between the female team and the male team.

Each row of this table indicates that a player_name and with gender has scored score_point in someday.

Gender is 'F' if the player is in the female team and 'M' if the player is in the male team.

Write an SQL query to find the total score for each gender on each day.

Return the result table ordered by gender and day in ascending order.

The query result format is in the following example.

Input:

Scores table:

player_name	gender	day	score_points
Aron	F	2020-01-01	17
Alice	F	2020-01-07	23
Bajrang	M	2020-01-07	7
Khali	M	2019-12-25	11
Slaman	M	2019-12-30	13
Joe	M	2019-12-31	3
Jose	M	2019-12-18	2
Priya	F	2019-12-31	23
Priyanka	F	2019-12-30	17

Output:

gender	day	total
F	2019-12-30	17
F	2019-12-31	40
F	2020-01-01	57
F	2020-01-07	80
M	2019-12-18	2
M	2019-12-25	13

M	2019-12-30	26
M	2019-12-31	29
M	2020-01-07	36

Explanation:

For the female team:

The first day is 2019-12-30, Priyanka scored 17 points and the total score for the team is 17.

The second day is 2019-12-31, Priya scored 23 points and the total score for the team is 40.

The third day is 2020-01-01, Aron scored 17 points and the total score for the team is 57.

The fourth day is 2020-01-07, Alice scored 23 points and the total score for the team is 80.

For the male team:

The first day is 2019-12-18, Jose scored 2 points and the total score for the team is 2.

The second day is 2019-12-25, Khali scored 11 points and the total score for the team is 13.

The third day is 2019-12-30, Slaman scored 13 points and the total score for the team is 26.

The fourth day is 2019-12-31, Joe scored 3 points and the total score for the team is 29.

The fifth day is 2020-01-07, Bajrang scored 7 points and the total score for the team is 36.

Q128.

Table Person:

Column Name	Type
id	int
name	varchar
phone_number	varchar

id is the primary key for this table.

Each row of this table contains the name of a person and their phone number.

Phone number will be in the form 'xxx-yyyyyy' where xxx is the country code (3 characters) and yyyyyy is the phone number (7 characters) where x and y are digits. Both can contain leading zeros.

Table Country:

Column Name	Type
name	varchar
country_code	varchar

country_code is the primary key for this table.

Each row of this table contains the country name and its code. country_code will be in the form 'xxx' where x is digits.

Table Calls:

Column Name	Type
caller_id	int
callee_id	int
duration	int

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

Each row of this table contains the caller id, callee id and the duration of the call in minutes. caller_id != callee_id

A telecommunications company wants to invest in new countries. The company intends to invest in the countries where the average call duration of the calls in this country is strictly greater than the global average call duration.

Write an SQL query to find the countries where this company can invest.

Return the result table in any order.

The query result format is in the following example.

Input:

Person table:

id	name	phone_number
3	Jonathan	051-1234567
12	Elvis	051-7654321
1	Moncef	212-1234567
2	Maroua	212-6523651
7	Meir	972-1234567
9	Rachel	972-0011100

Country table:

name	country_code
Peru	51
Israel	972
Morocco	212
Germany	49
Ethiopia	251
Ethiopia	251

Calls table:

caller_id	callee_id	duration
1	9	33
2	9	4

1	2	59
3	12	102
3	12	330
12	3	5
7	9	13
7	1	3
9	7	1
1	7	7

Output:

country
Peru

Explanation:

The average call duration for Peru is $(102 + 102 + 330 + 330 + 5 + 5) / 6 = 145.666667$

The average call duration for Israel is $(33 + 4 + 13 + 13 + 3 + 1 + 1 + 7) / 8 = 9.37500$

The average call duration for Morocco is $(33 + 4 + 59 + 59 + 3 + 7) / 6 = 27.5000$

Global call duration average = $(2 * (33 + 4 + 59 + 102 + 330 + 5 + 13 + 3 + 1 + 7)) / 20 = 55.70000$

Since Peru is the only country where the average call duration is greater than the global average, it is the only recommended country.

Q129.

Table: Numbers

Column Name	Type
num	int
frequency	int

num is the primary key for this table.

Each row of this table shows the frequency of a number in the database.

The median is the value separating the higher half from the lower half of a data sample.

Write an SQL query to report the median of all the numbers in the database after decompressing the Numbers table. Round the median to one decimal point.

The query result format is in the following example.

Input:

Numbers table:

num	frequency
0	7
1	1
2	3
3	1

Output:

median
0

Explanation:

If we decompose the Numbers table, we will get [0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 3], so the median is $(0 + 0) / 2 = 0$.

Q130.

Table: Salary

Column Name	Type
id	int
employee_id	int
amount	int
pay_date	date

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

Column Name	Type
employee_id	int
department_id	int

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.

Input:

Salary table:

id	employee_id	amount	pay_date
1	1	9000	2017/03/31
2	2	6000	2017/03/31
3	3	10000	2017/03/31
4	1	7000	2017/02/28
5	2	6000	2017/02/28
6	3	8000	2017/02/28

Employee table:

employee_id	department_id
1	1
2	2
3	2

Output:

pay_month	department_id	comparison
2017-02	1	same
2017-03	1	higher
2017-02	2	same
2017-03	2	lower

Explanation:

In March, the company's average salary is $(9000+6000+10000)/3 = 8333.33...$

The average salary for department '1' is 9000, which is the salary of employee_id '1' since there is only one employee in this department. So the comparison result is 'higher' since $9000 > 8333.33$ obviously.

The average salary of department '2' is $(6000 + 10000)/2 = 8000$, which is the average of employee_id '2' and '3'. So the comparison result is 'lower' since $8000 < 8333.33$.

With the same formula for the average salary comparison in February, the result is 'same' since both the departments '1' and '2' have the same average salary with the company, which is 7000.

Q131.

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.

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.

Input:

Activity table:

player_id	device_id	event_date	games_played
1	2	2016-03-01	5
1	2	2016-03-02	6
2	3	2017-06-25	1
3	1	2016-03-01	0
3	4	2016-07-03	5

Output:

install_dt	installs	Day1_retention
2016-03-01	2	0.5
2017-06-25	1	0

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 $0 / 1 = 0.00$

Q132.

Table: Players

Column Name	Type
player_id	int
group_id	int

player_id is the primary key of this table.
Each row of this table indicates the group of each player.

Table: Matches

Column Name	Type
match_id	int
first_player	int
second_player	int
first_score	int
second_score	int

match_id is the primary key of this table.
Each row is a record of a match, first_player and second_player contain the player_id of each match.
first_score and second_score contain the number of points of the first_player and second_player respectively.
You may assume that, in each match, players belong to the same group.

The winner in each group is the player who scored the maximum total points within the group. In the case of a tie, the lowest player_id wins.
Write an SQL query to find the winner in each group.
Return the result table in any order.
The query result format is in the following example.

Input:

Players table:

player_id	group_id
15	1
25	1
30	1
45	1
10	2
35	2

50	2
20	3
40	3

Matches table:

match_id	first_player	second_player	first_score	second_score
1	15	45	3	0
2	30	25	1	2
3	30	15	2	0
4	40	20	5	2
5	35	50	1	1

Output:

group_id	player_id
1	15
2	35
3	40

Q133.

Table: Student

Column Name	Type
student_id	int
student_name	varchar

student_id is the primary key for this table.
student_name is the name of the student.

Table: Exam

Column Name	Type
exam_id	int
student_id	int
score	int

(exam_id, student_id) is the primary key for this table.

Each row of this table indicates that the student with student_id had a score points in the exam with id exam_id.

A quiet student is the one who took at least one exam and did not score the high or the low score.

Write an SQL query to report the students (student_id, student_name) being quiet in all exams. Do not return the student who has never taken any exam.

Return the result table ordered by student_id.

The query result format is in the following example.

Input:

Student table:

student_id	student_name
1	Daniel
2	Jade
3	Stella
4	Jonathan
5	Will

Exam table:

exam_id	student_id	score
10	1	70
10	2	80
10	3	90
20	1	80
30	1	70
30	3	80
30	4	90
40	1	60
40	2	70
40	4	80

Output:

student_id	student_name
2	Jade

Explanation:

For exam 1: Student 1 and 3 hold the lowest and high scores respectively.

For exam 2: Student 1 holds both the highest and lowest score.

For exam 3 and 4: Student 1 and 4 hold the lowest and high scores respectively.

Students 2 and 5 have never got the highest or lowest in any of the exams.

Since student 5 is not taking any exam, he is excluded from the result.

So, we only return the information of Student 2.

Q134.

Table: Student

Column Name	Type
student_id	int
student_name	varchar

student_id is the primary key for this table.
student_name is the name of the student.

Table: Exam

Column Name	Type
exam_id	int
student_id	int
score	int

(exam_id, student_id) is the primary key for this table.
Each row of this table indicates that the student with student_id had a score points in the exam with id exam_id.

A quiet student is the one who took at least one exam and did not score the high or the low score.
Write an SQL query to report the students (student_id, student_name) being quiet in all exams. Do not return the student who has never taken any exam.
Return the result table ordered by student_id.
The query result format is in the following example.

Input:

Student table:

student_id	student_name
1	Daniel
2	Jade
3	Stella
4	Jonathan
5	Will

Exam table:

exam_id	student_id	score
10	1	70
10	2	80
10	3	90
20	1	80

30	1	70
30	3	80
30	4	90
40	1	60
40	2	70
40	4	80

Output:

student_id	student_name
2	Jade

Explanation:

For exam 1: Student 1 and 3 hold the lowest and high scores respectively.

For exam 2: Student 1 holds both the highest and lowest score.

For exam 3 and 4: Student 1 and 4 hold the lowest and high scores respectively.

Students 2 and 5 have never got the highest or lowest in any of the exams.

Since student 5 is not taking any exam, he is excluded from the result.

So, we only return the information of Student 2.

Q135.

Table: UserActivity

Column Name	Type
username	varchar
activity	varchar
startDate	Date
endDate	Date

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

This table contains information about the activity performed by each user in a period of time.

A person with a username performed an activity from startDate to endDate.

Write an SQL query to show the second most recent activity of each user.

If the user only has one activity, return that one. A user cannot perform more than one activity at the same time.

Return the result table in any order.

The query result format is in the following example.

Input:

UserActivity table:

username	activity	startDate	endDate
Alice	Travel	2020-02-12	2020-02-20

Alice	Dancing	2020-02-21	2020-02-23
Alice	Travel	2020-02-24	2020-02-28
Bob	Travel	2020-02-11	2020-02-18

Output:

username	activity	startDate	endDate
Alice	Dancing	2020-02-21	2020-02-23
Bob	Travel	2020-02-11	2020-02-18

Explanation:

The most recent activity of Alice is Travel from 2020-02-24 to 2020-02-28, before that she was dancing from 2020-02-21 to 2020-02-23.

Bob only has one record, we just take that one.

Q136.

Table: UserActivity

Column Name	Type
username	varchar
activity	varchar
startDate	Date
endDate	Date

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

This table contains information about the activity performed by each user in a period of time.

A person with a username performed an activity from startDate to endDate.

Write an SQL query to show the second most recent activity of each user.

If the user only has one activity, return that one. A user cannot perform more than one activity at the same time.

Return the result table in any order.

The query result format is in the following example.

Input:

UserActivity table:

username	activity	startDate	endDate
Alice	Travel	2020-02-12	2020-02-20
Alice	Dancing	2020-02-21	2020-02-23
Alice	Travel	2020-02-24	2020-02-28
Bob	Travel	2020-02-11	2020-02-18

Output:

username	activity	startDate	endDate
Alice	Dancing	2020-02-21	2020-02-23
Bob	Travel	2020-02-11	2020-02-18

Explanation:

The most recent activity of Alice is Travel from 2020-02-24 to 2020-02-28, before that she was dancing from 2020-02-21 to 2020-02-23.

Bob only has one record, we just take that one.

Q137.

Samantha was tasked with calculating the average monthly salaries for all employees in the EMPLOYEES table, but did not realise her keyboard's 0 key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeros removed), and the actual average salary.

Write a query calculating the amount of error (i.e.: actual - miscalculated average monthly salaries), and round it up to the next integer.

Input Format

The EMPLOYEES table is described as follows:

<i>Column</i>	<i>Type</i>
<i>ID</i>	<i>Integer</i>
<i>Name</i>	<i>String</i>
<i>Salary</i>	<i>Integer</i>

Note: Salary is per month.

Constraints

$1000 < \text{salary} < 10^5$

Sample Input

<i>ID</i>	<i>Name</i>	<i>Salary</i>
<i>1</i>	<i>Kristeen</i>	<i>1420</i>
<i>2</i>	<i>Ashley</i>	<i>2006</i>
<i>3</i>	<i>Julia</i>	<i>2210</i>
<i>4</i>	<i>Maria</i>	<i>3000</i>

Sample Output
2061

Explanation

The table below shows the salaries without zeros as they were entered by Samantha:

<i>ID</i>	<i>Name</i>	<i>Salary</i>
1	Kristeen	142
2	Ashley	26
3	Julia	221
4	Maria	3

Samantha computes an average salary of 98.00 . The actual average salary is 2159.00.

The resulting error between the two calculations is $2159.00 - 98.00 = 2061.00$. Since it is equal to the integer 2061, it does not get rounded up.

Q138.

We define an employee's total earnings to be their monthly salary * months 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 2 space-separated integers.

Level - Easy

Hint - Use Aggregation functions

Input Format

The Employee table containing employee data for a company is described as follows:

Column	Type
employee_id	Integer
name	String
months	Integer
salary	Integer

where employee_id is an employee's ID number, name is their name, months is the total number of months they've been working for the company, and salary is the their monthly salary.

Sample Input

employee_id	name	months	salary
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

Sample Output
69952 1

Explanation:

The table and earnings data is depicted in the following diagram:

employee_id	name	months	salary	earnings
12228	Rose	15	1968	29520
33645	Angela	1	3443	3443
45692	Frank	17	1608	27336
56118	Patrick	7	1345	9415
59725	Lisa	11	2330	25630
74197	Kimberly	16	4372	69952
78454	Bonnie	8	1771	14168
83565	Michael	6	2017	12102
98607	Todd	5	3396	16980
99989	Joe	9	3573	32157

The maximum earnings value is 69952. The only employee with earnings= 69952 is Kimberly, so we print the maximum earnings value (69952) and a count of the number of employees who have earned \$69952 (which is 1) as two space-separated values.

Q139.

Generate the following two result sets:

1. 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). For example: AnActorName(A), ADoctorName(D), AProfessorName(P), and ASingerName(S).

Query the number of occurrences of each occupation in OCCUPATIONS. Sort the occurrences in ascending order, and output them in the following format:

Level - Medium

There are a total of [occupation_count] [occupation]s.

2. 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.

Input Format

The OCCUPATIONS table is described as follows:

<i>Column</i>	<i>Type</i>
<i>Name</i>	<i>String</i>
<i>Occupation</i>	<i>String</i>

Occupation will only contain one of the following values: Doctor, Professor, Singer or Actor.

Sample Input

An OCCUPATIONS table that contains the following records:

<i>Name</i>	<i>Occupation</i>
<i>Samantha</i>	<i>Doctor</i>
<i>Julia</i>	<i>Actor</i>
<i>Maria</i>	<i>Actor</i>
<i>Meera</i>	<i>Singer</i>
<i>Ashely</i>	<i>Professor</i>
<i>Ketty</i>	<i>Professor</i>
<i>Christeen</i>	<i>Professor</i>
<i>Jane</i>	<i>Actor</i>
<i>Jenny</i>	<i>Doctor</i>
<i>Priya</i>	<i>Singer</i>

Sample Output

Ashely(P)

Christeen(P)

Jane(A)

Jenny(D)

Julia(A)

Ketty(P)

Maria(A)

Meera(S)

Priya(S)

Samantha(D)

There are a total of 2 doctors.

There are a total of 2 singers.

There are a total of 3 actors.

There are a total of 3 professors.

Hint -

The results of the first query are formatted to the problem description's specifications.

The results of the second query are ascendingly ordered first by number of names corresponding to each profession (2<= 2<=3<=3), and then alphabetically by profession (doctor <= singer , and actor <= professor).

Q140 .

Pivot 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.

Input Format

The OCCUPATIONS table is described as follows:

<i>Column</i>	<i>Type</i>
<i>Name</i>	<i>String</i>
<i>Occupation</i>	<i>String</i>

Occupation will only contain one of the following values: Doctor, Professor, Singer or Actor.

Sample Input

<i>Name</i>	<i>Occupation</i>
<i>Samantha</i>	<i>Doctor</i>
<i>Julia</i>	<i>Actor</i>
<i>Maria</i>	<i>Actor</i>
<i>Meera</i>	<i>Singer</i>
<i>Ashely</i>	<i>Professor</i>
<i>Ketty</i>	<i>Professor</i>
<i>Christeen</i>	<i>Professor</i>
<i>Jane</i>	<i>Actor</i>
<i>Jenny</i>	<i>Doctor</i>
<i>Priya</i>	<i>Singer</i>

Sample Output

Jenny Ashley Meera Jane
Samantha Christeen Priya Julia
NULL Ketty NULL Maria

Hint -

The first column is an alphabetically ordered list of Doctor names.

The second column is an alphabetically ordered list of Professor names.

The third column is an alphabetically ordered list of Singer names.

The fourth column is an alphabetically ordered list of Actor names.

The empty cell data for columns with less than the maximum number of names per occupation (in this case, the Professor and Actor columns) are filled with NULL values.

Q141.

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.

<i>Column</i>	<i>Type</i>
<i>N</i>	<i>Integer</i>
<i>P</i>	<i>Integer</i>

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.

Sample Input

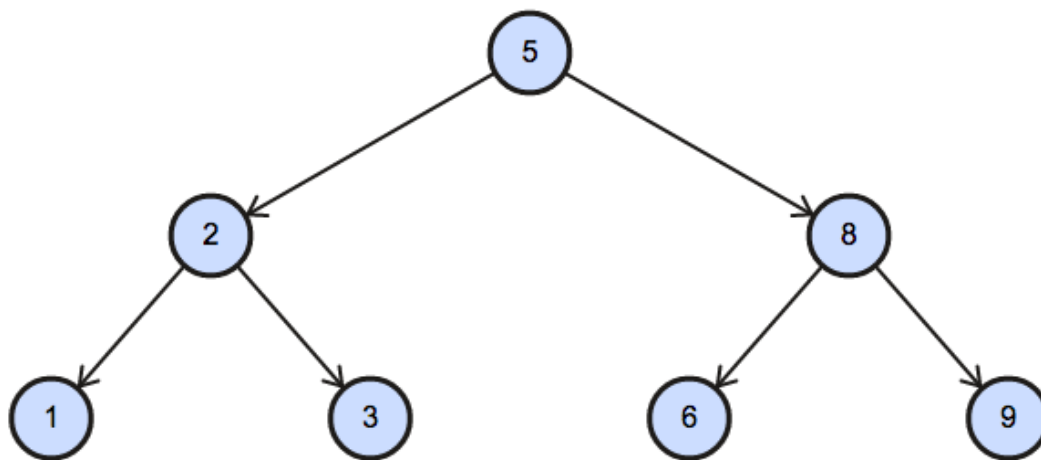
<i>N</i>	<i>P</i>
<i>1</i>	<i>2</i>
<i>3</i>	<i>2</i>
<i>6</i>	<i>8</i>
<i>9</i>	<i>8</i>
<i>2</i>	<i>5</i>
<i>8</i>	<i>5</i>
<i>5</i>	<i>null</i>

Sample Output

1 Leaf
2 Inner
3 Leaf
5 Root
6 Leaf
8 Inner
9 Leaf

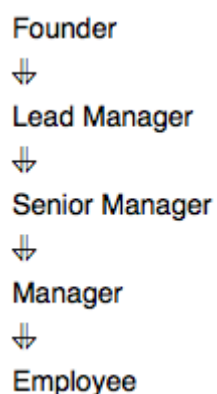
Explanation

The Binary Tree below illustrates the sample:



Q142 .

Amber's conglomerate corporation just acquired some new companies. Each of the companies



follows this hierarchy:

Given the table schemas below, write a query to print the company_code, founder name, total number of lead managers, total number of senior managers, total number of managers, and total number of employees. Order your output by ascending company_code.

Level - Medium

Note:

- The tables may contain duplicate records.

- The company_code is string, so the sorting should not be numeric. For example, if the company_codes are C_1, C_2, and C_10, then the ascending company_codes will be C_1, C_10, and C_2.

Input Format

The following tables contain company data:

- Company: The company_code is the code of the company and founder is the founder of the

Column	Type
company_code	String
founder	String

company.

- Lead_Manager: The lead_manager_code is the code of the lead manager, and the

Column	Type
lead_manager_code	String
company_code	String

company_code is the code of the working company.

- Senior_Manager: The senior_manager_code is the code of the senior manager, the lead_manager_code is the code of its lead manager, and the company_code is the code of the

Column	Type
senior_manager_code	String
lead_manager_code	String
company_code	String

working company.

- Manager: The manager_code is the code of the manager, the senior_manager_code is the code of its senior manager, the lead_manager_code is the code of its lead manager, and the company_code is the code of the working company.

Column	Type
manager_code	String
senior_manager_code	String
lead_manager_code	String
company_code	String

- Employee: The employee_code is the code of the employee, the manager_code is the code of its manager, the senior_manager_code is the code of its senior manager, the

lead_manager_code is the code of its lead manager, and the company_code is the code of the

Column	Type
employee_code	String
manager_code	String
senior_manager_code	String
lead_manager_code	String
company_code	String

working company.

Sample Input

company_code	founder
C1	Monika
C2	Samantha

Company Table:

lead_manager_code	company_code
LM1	C1
LM2	C2

Lead_Manager Table:

Senior_Manager Table:

senior_manager_code	lead_manager_code	company_code
SM1	LM1	C1
SM2	LM1	C1
SM3	LM2	C2

Manager Table:

manager_code	senior_manager_code	lead_manager_code	company_code
M1	SM1	LM1	C1
M2	SM3	LM2	C2
M3	SM3	LM2	C2

Employee Table:

employee_code	manager_code	senior_manager_code	lead_manager_code	company_code
E1	M1	SM1	LM1	C1
E2	M1	SM1	LM1	C1
E3	M2	SM3	LM2	C2
E4	M3	SM3	LM2	C2

Sample Output

C1 Monika 1 2 1 2

C2 Samantha 1 1 2 2

Hint -

In company C1, the only lead manager is LM1. There are two senior managers, SM1 and SM2, under LM1. There is one manager, M1, under senior manager SM1. There are two employees, E1 and E2, under manager M1.

In company C2, the only lead manager is LM2. There is one senior manager, SM3, under LM2. There are two managers, M2 and M3, under senior manager SM3. There is one employee, E3, under manager M2, and another employee, E4, under manager, M3.

Q143 .

You are given a table, Functions, containing two columns: X and Y.

<i>Column</i>	<i>Type</i>
<i>X</i>	<i>Integer</i>
<i>Y</i>	<i>Integer</i>

Two pairs (X1, Y1) and (X2, Y2) are said to be symmetric pairs if $X1 = Y2$ and $X2 = Y1$.

Write a query to output all such symmetric pairs in ascending order by the value of X. List the rows such that $X1 \leq Y1$.

Level - Medium

Source - Hackerrank

Hint - Use group by and having clause .

Sample Input

<i>X</i>	<i>Y</i>
20	20
20	20
20	21
23	22
22	23
21	20

Sample Output

20 20

20 21

22 23

Q144 .

You are given three tables: Students, Friends and Packages. Students contains two columns: ID and Name. Friends contains two columns: ID and Friend_ID (ID of the ONLY best friend). Packages contain two columns: ID and Salary (offered salary in \$ thousands per month).

<i>Column</i>	<i>Type</i>
<i>ID</i>	<i>Integer</i>
<i>Name</i>	<i>String</i>

Students

<i>Column</i>	<i>Type</i>
<i>ID</i>	<i>Integer</i>
<i>Friend_ID</i>	<i>Integer</i>

Friends

<i>Column</i>	<i>Type</i>
<i>ID</i>	<i>Integer</i>
<i>Salary</i>	<i>Float</i>

Packages

Write a query to output the names of those students whose best friends got offered a higher salary than them. Names must be ordered by the salary amount offered to the best friends. It is guaranteed that no two students get the same salary offer.

Sample Input

<i>ID</i>	<i>Friend_ID</i>
1	2
2	3
3	4
4	1

Friends

<i>ID</i>	<i>Name</i>
1	Ashley
2	Samantha
3	Julia
4	Scarlet

Students

<i>ID</i>	<i>Salary</i>
1	15.20
2	10.06
3	11.55
4	12.12

Packages

Sample Output

Samantha

Julia

Scarlet

Explanation

See the following table:

<i>ID</i>	1	2	3	4
<i>Name</i>	Ashley	Samantha	Julia	Scarlet
<i>Salary</i>	15.20	10.06	11.55	12.12
<i>Friend ID</i>	2	3	4	1
<i>Friend Salary</i>	10.06	11.55	12.12	15.20

Now,

- Samantha's best friend got offered a higher salary than her at 11.55
- Julia's best friend got offered a higher salary than her at 12.12
- Scarlet's best friend got offered a higher salary than her at 15.2
- Ashley's best friend did NOT get offered a higher salary than her

The name output, when ordered by the salary offered to their friends, will be:

- Samantha
- Julia
- Scarlet

Q145.

Julia just finished conducting a coding contest, and she needs your help assembling the leaderboard! Write a query to print the respective hacker_id and name of hackers who achieved full scores for more than one challenge. Order your output in descending order by the total number of challenges in which the hacker earned a full score. If more than one hacker received full scores in the same number of challenges, then sort them by ascending hacker_id.

Level - Medium

Hint - Use group by and having clause and order by .

Input Format

The following tables contain contest data:

- Hackers: The hacker_id is the id of the hacker, and name is the name of the hacker.

Column	Type
hacker_id	Integer
name	String

- Difficulty: The difficult_level is the level of difficulty of the challenge, and score is the

Column	Type
difficulty_level	Integer
score	Integer

score of the challenge for the difficulty level.

- Challenges: The challenge_id is the id of the challenge, the hacker_id is the id of the hacker who created the challenge, and difficulty_level is the level of difficulty of the challenge.

Column	Type
challenge_id	Integer
hacker_id	Integer
difficulty_level	Integer

- Submissions: The submission_id is the id of the submission, hacker_id is the id of the hacker who made the submission, challenge_id is the id of the challenge that the submission belongs

Column	Type
submission_id	Integer
hacker_id	Integer
challenge_id	Integer
score	Integer

to, and score is the score of the submission.

Sample Input

hacker_id	name
5580	Rose
8439	Angela
27205	Frank
52243	Patrick
52348	Lisa
57645	Kimberly
77726	Bonnie
83082	Michael
86870	Todd
90411	Joe

Hackers Table:

difficulty_level	score
1	20
2	30
3	40
4	60
5	80
6	100
7	120

Difficulty Table:

challenge_id	hacker_id	difficulty_level
4810	77726	4
21089	27205	1
36566	5580	7
66730	52243	6
71055	52243	2

Challenges Table:

:

submission_id	hacker_id	challenge_id	score
68628	77726	36566	30
65300	77726	21089	10
40326	52243	36566	77
8941	27205	4810	4
83554	77726	66730	30
43353	52243	66730	0
55385	52348	71055	20
39784	27205	71055	23
94613	86870	71055	30
45788	52348	36566	0
93058	86870	36566	30
7344	8439	66730	92
2721	8439	4810	36
523	5580	71055	4
49105	52348	66730	0
55877	57645	66730	80
38355	27205	66730	35
3924	8439	36566	80
97397	90411	66730	100
84162	83082	4810	40
97431	90411	71055	30

Submissions Table

Sample Output

90411 Joe

Explanation

Hacker 86870 got a score of 30 for challenge 71055 with a difficulty level of 2, so 86870 earned a full score for this challenge.

Hacker 90411 got a score of 30 for challenge 71055 with a difficulty level of 2, so 90411 earned a full score for this challenge.

Hacker 90411 got a score of 100 for challenge 66730 with a difficulty level of 6, so 90411 earned a full score for this challenge.

Only hacker 90411 managed to earn a full score for more than one challenge, so we print their hacker_id and name as 2 space-separated values.

Q146.

You are given a table, Projects, containing three columns: Task_ID, Start_Date and End_Date. It is guaranteed that the difference between the End_Date and the Start_Date is equal to 1 day for each row in the table.

Level - Medium

Hint - Use Advance join

<i>Column</i>	<i>Type</i>
<i>Task_ID</i>	<i>Integer</i>
<i>Start_Date</i>	<i>Date</i>
<i>End_Date</i>	<i>Date</i>

If the End_Date of the tasks are consecutive, then they are part of the same project. Samantha is interested in finding the total number of different projects completed.

Write a query to output the start and end dates of projects listed by the number of days it took to complete the project in ascending order. If there is more than one project that have the same number of completion days, then order by the start date of the project.

Sample Input

<i>Task_ID</i>	<i>Start_Date</i>	<i>End_Date</i>
1	2015-10-01	2015-10-02
2	2015-10-02	2015-10-03
3	2015-10-03	2015-10-04
4	2015-10-13	2015-10-14
5	2015-10-14	2015-10-15
6	2015-10-28	2015-10-29
7	2015-10-30	2015-10-31

Sample Output

2015-10-28 2015-10-29

2015-10-30 2015-10-31

2015-10-13 2015-10-15

2015-10-01 2015-10-04

Explanation

The example describes following four projects:

- Project 1: Tasks 1, 2 and 3 are completed on consecutive days, so these are part of the project. Thus the start date of project is 2015-10-01 and end date is 2015-10-04, so it took 3 days to complete the project.
- Project 2: Tasks 4 and 5 are completed on consecutive days, so these are part of the project. Thus, the start date of project is 2015-10-13 and end date is 2015-10-15, so it took 2 days to complete the project.
- Project 3: Only task 6 is part of the project. Thus, the start date of project is 2015-10-28 and end date is 2015-10-29, so it took 1 day to complete the project.
- Project 4: Only task 7 is part of the project. Thus, the start date of project is 2015-10-30 and end date is 2015-10-31, so it took 1 day to complete the project.

Q147.

In an effort to identify high-value customers, Amazon asked for your help to obtain data about users who go on shopping sprees. A shopping spree occurs when a user makes purchases on 3 or more consecutive days.

List the user IDs who have gone on at least 1 shopping spree in ascending order.

transactions Table:

Column Name	Type
user_id	integer
amount	float
transaction_date	timestamp

transactions Example Input:

user_id	amount	transaction_date
1	9.99	08/01/2022 10:00:00
1	55	08/17/2022 10:00:00
2	149.5	08/05/2022 10:00:00
2	4.89	08/06/2022 10:00:00
2	34	08/07/2022 10:00:00

Example Output:

user_id
2

Q148 .

You are given a table of PayPal payments showing the payer, the recipient, and the amount paid. A two-way unique relationship is established when two people send money back and forth. Write a query to find the number of two-way unique relationships in this data.

Assumption:

- A payer can send money to the same recipient multiple times.

payments Table:

Column Name	Type
payer_id	integer
recipient_id	integer
amount	integer

payments Example Input:

payer_id	recipient_id	amount
101	201	30
201	101	10
101	301	20
301	101	80
201	301	70

Example Output:

unique_relationships
2

Q149. Assume you are given the table below on user transactions. Write a query to obtain the list of customers whose first transaction was valued at \$50 or more. Output the number of users.

Clarification:

- Use the `transaction_date` field to determine which transaction should be labeled as the first for each user.
- Use a specific function (we can't give too much away!) to account for scenarios where a user had multiple transactions on the same day, and one of those was the first.

`user_transactions` Table:

Column Name	Type
<code>transaction_id</code>	integer
<code>user_id</code>	integer
<code>spend</code>	decimal
<code>transaction_date</code>	timestamp

`user_transactions` Example Input:

<code>transaction_id</code>	<code>user_id</code>	<code>spend</code>	<code>transaction_date</code>
759274	111	49.50	02/03/2022 00:00:00
850371	111	51.00	03/15/2022 00:00:00
615348	145	36.30	03/22/2022 00:00:00
137424	156	151.00	04/04/2022 00:00:00
248475	156	87.00	04/16/2022 00:00:00

Example Output:

users

Q150.

Assume you are given the table below containing measurement values obtained from a sensor over several days. Measurements are taken several times within a given day.

Write a query to obtain the sum of the odd-numbered and even-numbered measurements on a particular day, in two different columns.

Note that the 1st, 3rd, 5th measurements within a day are considered odd-numbered measurements and the 2nd, 4th, 6th measurements are even-numbered measurements.

measurements Table:

Column Name	Type
measurement_id	integer
measurement_value	decimal
measurement_time	datetime

measurements Example Input:

measurement_id	measurement_value	measurement_time
131233	1109.51	07/10/2022 09:00:00
135211	1662.74	07/10/2022 11:00:00
523542	1246.24	07/10/2022 13:15:00
143562	1124.50	07/11/2022 15:00:00
346462	1234.14	07/11/2022 16:45:00

Example Output:

measurement_day	odd_sum	even_sum
07/10/2022 00:00:00	2355.75	1662.74

07/11/2022 00:00:00	1124.50	1234.14
---------------------	---------	---------

Q151.

In an effort to identify high-value customers, Amazon asked for your help to obtain data about users who go on shopping sprees. A shopping spree occurs when a user makes purchases on 3 or more consecutive days.

List the user IDs who have gone on at least 1 shopping spree in ascending order.

Level - Medium

Hint - Use self join

transactions Table:

Column Name	Type
user_id	integer
amount	float
transaction_date	timestamp

transactions Example Input:

user_id	amount	transaction_date
1	9.99	08/01/2022 10:00:00
1	55	08/17/2022 10:00:00
2	149.5	08/05/2022 10:00:00
2	4.89	08/06/2022 10:00:00
2	34	08/07/2022 10:00:00

Example Output:

user_id
2

Q152.

The Airbnb Booking Recommendations team is trying to understand the "substitutability" of two rentals and whether one rental is a good substitute for another. They want you to write a query to find the unique combination of two Airbnb rentals with the same exact amenities offered. Output the count of the unique combination of Airbnb rentals.

Level - Medium

Hint - Use unique statement

Assumptions:

- If property 1 has a kitchen and pool, and property 2 has a kitchen and pool too, it is a good substitute and represents a unique matching rental.
- If property 3 has a kitchen, pool and fireplace, and property 4 only has a pool and fireplace, then it is not a good substitute.

rental_amenities Table:

Column Name	Type
rental_id	integer
amenity	string

rental_amenities Example Input:

rental_id	amenity
123	pool
123	kitchen
234	hot tub
234	fireplace
345	kitchen

345	pool
456	pool

Example Output:

matching_airbnb
1

Q153.

Google marketing managers are analysing the performance of various advertising accounts over the last month. They need your help to gather the relevant data.

Write a query to calculate the return on ad spend (ROAS) for each advertiser across all ad campaigns. Round your answer to 2 decimal places, and order your output by the advertiser_id.

Level - Medium

Hint: ROAS = Ad Revenue / Ad Spend

ad_campaigns Table:

Column Name	Type
campaign_id	integer
spend	integer
revenue	float
advertiser_id	integer

ad_campaigns Example Input:

campaign_id	spend	revenue	advertiser_id
1	5000	7500	3
2	1000	900	1
3	3000	12000	2
4	500	2000	4
5	100	400	4

Example Output:

advertiser_id	ROAS
1	0.9
2	4
3	1.5
4	4

Q154.

Your team at Accenture is helping a Fortune 500 client revamp their compensation and benefits program. The first step in this analysis is to manually review employees who are potentially overpaid or underpaid.

An employee is considered to be potentially overpaid if they earn more than 2 times the average salary for people with the same title. Similarly, an employee might be underpaid if they earn less than half of the average for their title. We'll refer to employees who are both underpaid and overpaid as compensation outliers for the purposes of this problem.

Write a query that shows the following data for each compensation outlier: employee ID, salary, and whether they are potentially overpaid or potentially underpaid (refer to Example Output below).

Hint: $ROAS = \text{Ad Revenue} / \text{Ad Spend}$

employee_pay Table:

Column Name	Type
employee_id	integer
salary	integer
title	varchar

employee_pay Example Input:

employee_id	salary	title
101	80000	Data Analyst
102	90000	Data Analyst
103	100000	Data Analyst
104	30000	Data Analyst

105	120000	Data Scientist
106	100000	Data Scientist
107	80000	Data Scientist
108	310000	Data Scientist

Example Output:

employee_id	salary	status
104	30000	Underpaid
108	310000	Overpaid

Q155.

You are given a table of PayPal payments showing the payer, the recipient, and the amount paid. A two-way unique relationship is established when two people send money back and forth. Write a query to find the number of two-way unique relationships in this data.

Assumption:

- A payer can send money to the same recipient multiple times.

Hint- Use the INTERSECT set operator.

payments Table:

Column Name	Type
payer_id	integer
recipient_id	integer
amount	integer

payments Example Input:

payer_id	recipient_id	amount
101	201	30
201	101	10
101	301	20
301	101	80
201	301	70

Example Output:

unique_relationships
2

Q156.

Assume you are given the table below containing information on user purchases. Write a query to obtain the number of users who purchased the same product on two or more different days. Output the number of unique users.

PS. On 26 Oct 2022, we expanded the `purchases` data set, thus the official output may vary from before.

Hint- Count the distinct number of dates formatted into the `DATE` format in the `COUNT(DISTINCT)`.

purchases Table:

Column Name	Type
<code>user_id</code>	integer
<code>product_id</code>	integer
<code>quantity</code>	integer
<code>purchase_date</code>	datetime

purchasesExample Input:

<code>user_id</code>	<code>product_id</code>	<code>quantity</code>	<code>purchase_date</code>
536	3223	6	01/11/2022 12:33:44

827	3585	35	02/20/2022 14:05:26
536	3223	5	03/02/2022 09:33:28
536	1435	10	03/02/2022 08:40:00
827	2452	45	04/09/2022 00:00:00

Example Output:

repeat_purchasers
1

Q157.

Say you have access to all the transactions for a given merchant account. Write a query to print the cumulative balance of the merchant account at the end of each day, with the total balance reset back to zero at the end of the month. Output the transaction date and cumulative balance.

Hint-You should use CASE.

transactions Table:

Column Name	Type
transaction_id	integer
type	string ('deposit', 'withdrawal')
amount	decimal
transaction_date	timestamp

transactions Example Input:

transaction_id	type	amount	transaction_date
19153	deposit	65.90	07/10/2022 10:00:00
53151	deposit	178.55	07/08/2022 10:00:00

29776	withdrawal	25.90	07/08/2022 10:00:00
16461	withdrawal	45.99	07/08/2022 10:00:00
77134	deposit	32.60	07/10/2022 10:00:00

Example Output:

transaction_date	balance
07/08/2022 12:00:00	106.66
07/10/2022 12:00:00	205.16

Q158.

Assume you are given the table below containing information on Amazon customers and their spend on products belonging to various categories. Identify the top two highest-grossing products within each category in 2022. Output the category, product, and total spend.

Hint- Use where ,and, group by .

product_spend Table:

Column Name	Type
category	string
product	string
user_id	integer
spend	decimal
transaction_date	timestamp

product_spend Example Input:

category	product	user_id	spend	transaction_date
----------	---------	---------	-------	------------------

appliance	refrigerator	165	246.00	12/26/2021 12:00:00
appliance	refrigerator	123	299.99	03/02/2022 12:00:00
appliance	washing machine	123	219.80	03/02/2022 12:00:00
electronics	vacuum	178	152.00	04/05/2022 12:00:00
electronics	wireless headset	156	249.90	07/08/2022 12:00:00
electronics	vacuum	145	189.00	07/15/2022 12:00:00

Example Output:

category	product	total_spend
appliance	refrigerator	299.99
appliance	washing machine	219.80
electronics	vacuum	341.00
electronics	wireless headset	249.90

Q159.

Facebook is analysing its user signup data for June 2022. Write a query to generate the churn rate by week in June 2022. Output the week number (1, 2, 3, 4, ...) and the corresponding churn rate rounded to 2 decimal places.

For example, week number 1 represents the dates from 30 May to 5 Jun, and week 2 is from 6 Jun to 12 Jun.

Hint- Use Extract.

Assumptions:

- If the last_login date is within 28 days of the signup_date, the user can be considered churned.
- If the last_login is more than 28 days after the signup date, the user didn't churn.

users Table:

Column Name	Type
user_id	integer

signup_date	datetime
last_login	datetime

users Example Input:

user_id	signup_date	last_login
1001	06/01/2022 12:00:00	07/05/2022 12:00:00
1002	06/03/2022 12:00:00	06/15/2022 12:00:00
1004	06/02/2022 12:00:00	06/15/2022 12:00:00
1006	06/15/2022 12:00:00	06/27/2022 12:00:00
1012	06/16/2022 12:00:00	07/22/2022 12:00:00

Example Output:

signup_week	churn_rate
1	66.67
3	50.00

User ids 1001, 1002, and 1004 signed up in the first week of June 2022. Out of the 3 users, 1002 and 1004's last login is within 28 days from the signup date, hence they are churned users.

To calculate the churn rate, we take churned users divided by total users signup in the week. Hence 2 users / 3 users = 66.67%.