City-Dataset:https://docs.google.com/spreadsheets/d/1dk9kRwcMxj5USuJqxtITD05S-aOUD6fzNzVW41dcpgc/edit?usp=sharing

Q1. Query all columns for all American cities in the CITY table with populations larger than 100000.
The CountryCode for America is USA.
The CITY table is described as follows:

CITY

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

Q2. Query the NAME field for all American cities in the CITY table with populations larger than 120000.
The CountryCode for America is USA.
The CITY table is described as follows:

CITY

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

Q3. Query all columns (attributes) for every row in the CITY table.
The CITY table is described as follows:

CITY

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

Q4. Query all columns for a city in CITY with the ID 1661.
The CITY table is described as follows:

CITY

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

Q5. Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.
The CITY table is described as follows:

CITY

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

Q6. Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.
The CITY table is described as follows:

CITY

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

station-table:https://docs.google.com/spreadsheets/d/1sHPhE7walQD5mL7ppFNqybyoOJY3E51N0cWYzhp2UH4/edit?usp=sharing

Q7. Query a list of CITY and STATE from the STATION table.
The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.

Q8. Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.
The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude

Q9. Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.
The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.

For example, if there are three records in the table with CITY values 'New York', 'New York', 'Bengalaru', there are 2 different city names: 'New York' and 'Bengalaru'. The query returns , because total number of records - number of unique city names = 3-2 =1

Q10. Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.
The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.
Sample Input
For example, CITY has four entries: DEF, ABC, PQRS and WXY.
Sample Output

ABC 3
PQRS 4

Hint -
When ordered alphabetically, the CITY names are listed as ABC, DEF, PQRS, and WXY, with lengths and. The longest name is PQRS, but there are options for shortest named city. Choose ABC, because it comes first alphabetically.
Note
You can write two separate queries to get the desired output. It need not be a single query.

Q11. Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.
Input Format
The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.

Q12. Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION. Your result cannot contain duplicates.
Input Format
The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.

Q13. Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.
Input Format
The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.

Q14. Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.

Input Format
The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.

Q15. Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.
Input Format
The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.

Q16. Query the list of CITY names from STATION that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.

Input Format
The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.

Q17.

Table: Product

Column Name	Type
product_id	int
product_name	varchar
unit_price	int

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

Table: Sales

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

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

product_id is a foreign key to the Product table.
Each row of this table contains some information about one sale.

Write an SQL query that reports the products that were only sold in the first quarter of 2019. That is, between 2019-01-01 and 2019-03-31 inclusive.
Return the result table in any order.
The query result format is in the following example.

Input:
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

Output:

product_id	product_name
1	S8

Explanation:
The product with id 1 was only sold in the spring of 2019.
The product with id 2 was sold in the spring of 2019 but was also sold after the spring of 2019.
The product with id 3 was sold after spring 2019.
We return only product 1 as it is the product that was only sold in the spring of 2019.

Q18.

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.

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

Output:

id
4
7

Q19.

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 immediately; 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.

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

Explanation: The orders with delivery id 2 and 3 are immediate while the others are scheduled.

Q20.

Table: Ads

Column Name	Type
ad_id	int
user_id	int
action	enum

(ad_id, user_id) is the primary key for this table.
Each row of this table contains the ID of an Ad, the ID of a user, and the action taken by this user regarding this Ad.
The action column is an ENUM type of ('Clicked', 'Viewed', 'Ignored').

A company is running Ads and wants to calculate the performance of each Ad.
Performance of the Ad is measured using Click-Through Rate (CTR) where:

$$CTR = \begin{cases} 0, & \text{if Ad total clicks + Ad total views = 0} \\ \frac{\text{Ad total clicks}}{\text{Ad total clicks + Ad total views}} \times 100, & \text{otherwise} \end{cases}$$

Write an SQL query to find the ctr of each Ad. Round ctr to two decimal points.
Return the result table ordered by ctr in descending order and by ad_id in ascending order in case of a tie.
The query result format is in the following example.

Input:
Ads table:

ad_id	user_id	action
1	1	Clicked
2	2	Clicked
3	3	Viewed
5	5	Ignored
1	7	Ignored
2	7	Viewed
3	5	Clicked
1	4	Viewed
2	11	Viewed
1	2	Clicked

Output:

ad_id	ctr
1	66.67
3	50
2	33.33
5	0

Explanation:
for ad_id = 1, ctr = (2/(2+1)) * 100 = 66.67
for ad_id = 2, ctr = (1/(1+2)) * 100 = 33.33
for ad_id = 3, ctr = (1/(1+1)) * 100 = 50.00
for ad_id = 5, ctr = 0.00, Note that ad_id = 5 has no clicks or views.
Note that we do not care about Ignored Ads.

Q21.

Table: Employee

Column Name	Type
employee_id	int
team_id	int

employee_id is the primary key for this table.
Each row of this table contains the ID of each employee and their respective team.

Write an SQL query to find the team size of each of the employees.
Return result table in any order.
The query result format is in the following example.

Input:
Employee Table:

employee_id	team_id
1	8
2	8
3	8
4	7
5	9
6	9

Output:

employee_id	team_size
1	3
2	3
3	3
4	1
5	2
6	2

Explanation:
Employees with Id 1,2,3 are part of a team with team_id = 8.
An employee with Id 4 is part of a team with team_id = 7.
Employees with Id 5,6 are part of a team with team_id = 9.

Q22.

Table: Countries

Column Name	Type
country_id	int
country_name	varchar

country_id is the primary key for this table.
Each row of this table contains the ID and the name of one country.

Table: Weather

Column Name	Type
country_id	int
weather_state	int
day	date

(country_id, day) is the primary key for this table.
Each row of this table indicates the weather state in a country for one day.

Write an SQL query to find the type of weather in each country for November 2019.
The type of weather is:

- Cold if the average weather_state is less than or equal 15,
- Hot if the average weather_state is greater than or equal to 25, and
- Warm otherwise.

Return result table in any order.
The query result format is in the following example.

Input:
Countries table:

country_id	country_name
2	USA
3	Australia
7	Peru
5	China
8	Morocco
9	Spain

Weather table:

country_id	weather_state	day
2	15	2019-11-01
2	12	2019-10-28

2	12	2019-10-27
3	-2	2019-11-10
3	0	2019-11-11
3	3	2019-11-12
5	16	2019-11-07
5	18	2019-11-09
5	21	2019-11-23
7	25	2019-11-28
7	22	2019-12-01
7	20	2019-12-02
8	25	2019-11-05
8	27	2019-11-15
8	31	2019-11-25
9	7	2019-10-23
9	3	2019-12-23

Output:

country_name	weather_type
USA	Cold
Australia	Cold
Peru	Hot
Morocco	Hot
China	Warm

Explanation:
Average weather_state in the USA in November is (15) / 1 = 15 so the weather type is Cold.
Average weather_state in Australia in November is (-2 + 0 + 3) / 3 = 0.333 so the weather type is Cold.
Average weather_state in Peru in November is (25) / 1 = 25 so the weather type is Hot.
The average weather_state in China in November is (16 + 18 + 21) / 3 = 18.333 so the weather type is warm.
Average weather_state in Morocco in November is (25 + 27 + 31) / 3 = 27.667 so the weather type is Hot.
We know nothing about the average weather_state in Spain in November so we do not include it in the result table.

Q23.

Table: Prices

Column Name	Type
product_id	int
start_date	date
end_date	date
price	int

(product_id, start_date, end_date) is the primary key for this table.
Each row of this table indicates the price of the product_id in the period from start_date to end_date.
For each product_id there will be no two overlapping periods. That means there will be no two intersecting periods for the same product_id.

Table: UnitsSold

Column Name	Type
product_id	int
purchase_date	date
units	int

There is no primary key for this table, it may contain duplicates.
Each row of this table indicates the date, units, and product_id of each product sold.

Write an SQL query to find the average selling price for each product. average_price should be rounded to 2 decimal places.
Return the result table in any order.
The query result format is in the following example.

Input:
Prices table:

product_id	start_date	end_date	price
1	2019-02-17	2019-02-28	5
1	2019-03-01	2019-03-22	20
2	2019-02-01	2019-02-20	15
2	2019-02-21	2019-03-31	30

UnitsSold table:

product_id	purchase_date	units
1	2019-02-25	100
1	2019-03-01	15
2	2019-02-10	200
2	2019-03-22	30

Output:

product_id	average_price
1	6.96
2	16.96

Explanation:
Average selling price = Total Price of Product / Number of products sold.
Average selling price for product 1 = ((100 * 5) + (15 * 20)) / 115 = 6.96
Average selling price for product 2 = ((200 * 15) + (30 * 30)) / 230 = 16.96

Q24.

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.

Input:
Activity table:

player_id	device_id	event_date	games_played
1	2	2016-03-01	5
1	2	2016-05-02	6
2	3	2017-06-25	1
3	1	2016-03-02	0
3	4	2018-07-03	5

Output:

player_id	first_login
1	2016-03-01
2	2017-06-25
3	2016-03-02

Q25.

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 device that is first logged in for each player.
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	2016-05-02	6
2	3	2017-06-25	1
3	1	2016-03-02	0
3	4	2018-07-03	5

Output:

player_id	device_id
1	2
2	3
3	1

Q26.

Table: Products

Column Name	Type
product_id	int
product_name	varchar
product_category	varchar

product_id is the primary key 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

There is no primary key for this table. It may have duplicate rows.
product_id is a foreign key to the Products table.
unit is the number of products ordered in order_date.

Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount.
Return result table in any order.
The query result format is in the following example.

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.

Q27.

Table: Users

Column Name	Type
user_id	int
name	varchar
mail	varchar

user_id is the primary key for this table.
This table contains information of the users signed up in a website. Some emails are invalid.

Write an SQL query to find the users who have valid emails.
A valid e-mail has a prefix name and a domain where:

- The prefix name is a string that may contain letters (upper or lower case), digits, underscore '_', period '.', and/or dash '-'. The prefix name must start with a letter.
- The domain is '@leetcode.com'.

Return the result table in any order.
The query result format is in the following example.

Input:
Users table:

user_id	name	mail
1	Winston	winston@leetcode.com
2	Jonathan	jonathanisgreat
3	Annabelle	bella-@leetcode.com
4	Sally	sally.come@leetcode.com
5	Marwan	quarz#2020@leetcode.com
6	David	david69@gmail.com
7	Shapiro	.shapo@leetcode.com

Output:

user_id	name	mail
1	Winston	winston@leetcode.com
3	Annabelle	bella-@leetcode.com
4	Sally	sally.come@leetcode.com

Explanation:
The mail of user 2 does not have a domain.
The mail of user 5 has the # sign which is not allowed.
The mail of user 6 does not have the leetcode domain.
The mail of user 7 starts with a period.

Q28.

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 about the customers in the company.

Table: Product

Column Name	Type
customer_id	int
name	varchar
country	varchar

product_id is the primary key for this table.
This table contains information on 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.

Input:
Customers table:

customer_id	name	country
1	Winston	USA
2	Jonathan	Peru
3	Moustafa	Egypt

Product table:

product_id	description	price
10	LC Phone	300
20	LC T-Shirt	10
30	LC Book	45
40	LC Keychain	2

Orders table:

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

Output:

customer_id	name
1	Winston

Explanation:
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.

Q29.

Table: TVProgram

Column Name	Type
program_date	date
content_id	int
channel	varchar

(program_date, content_id) is the primary key for this table.
This table contains information about the programs on the TV.
content_id is the id of the program in some channel on the TV.

Table: Content

Column Name	Type
content_id	varchar
title	varchar
Kids_content	enum
content_type	varchar

content_id is the primary key for this table.
Kids_content is an enum that takes one of the values ('Y', 'N') where:

'Y' means content for kids, otherwise 'N' is not content for kids.
content_type is the category of the content as movies, series, etc.

Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020.
Return the result table in any order.
The query result format is in the following example.

Input:
TVProgram table:

program_date	content_id	channel
2020-06-10 08:00	1	LC-Channel
2020-05-11 12:00	2	LC-Channel
2020-05-12 12:00	3	LC-Channel
2020-05-13 14:00	4	Disney Ch
2020-06-18 14:00	4	Disney Ch
2020-07-15 16:00	5	Disney Ch

Content table:

content_id	title	Kids_content	content_type
1	Leetcode Movie	N	Movies
2	Alg. for Kids	Y	Series
3	Database Sols	N	Series
4	Aladdin	Y	Movies
5	Cinderella	Y	Movies

Output:

title
Aladdin

Explanation:
"Leetcode Movie" is not a content for kids.
"Alg. for Kids" is not a movie.
"Database Sols" is not a movie
"Alladin" is a movie, content for kids and was streamed in June 2020.
"Cinderella" was not streamed in June 2020.

Q30.

Table: NPV

Column Name	Type
id	int
year	int
npv	int

(id, year) is the primary key of this table.
The table has information about the id and the year of each inventory and the corresponding net present value.

Table: Queries

Column Name	Type
id	int
year	int

(id, year) is the primary key of this table.
The table has information about the id and the year of each inventory query.

Write an SQL query to find the npv of each query of the Queries table.
Return the result table in any order.
The query result format is in the following example.

Input:
NPV table:

id	year	npv
1	2018	100
7	2020	30
13	2019	40
1	2019	113
2	2008	121
3	2009	12
11	2020	99
7	2019	0

Queries table:

id	year
1	2019
2	2008
3	2009
7	2018
7	2019
7	2020
13	2019

Output:

id	year	npv
1	2019	113
2	2008	121
3	2009	12
7	2018	0
7	2019	0
7	2020	30
13	2019	40

Explanation:
The npv value of (7, 2018) is not present in the NPV table, we consider it 0.
The npv values of all other queries can be found in the NPV table.

Q31.

Table: NPV

Column Name	Type
id	int
year	int
npv	int

(id, year) is the primary key of this table.
The table has information about the id and the year of each inventory and the corresponding net present value.

Table: Queries

Column Name	Type
id	int
year	int

(id, year) is the primary key of this table.
The table has information about the id and the year of each inventory query.

Write an SQL query to find the npv of each query of the Queries table.
Return the result table in any order.
The query result format is in the following example.

Input:
NPV table:

id	year	npv
1	2018	100
7	2020	30
13	2019	40
1	2019	113
2	2008	121
3	2009	12
11	2020	99
7	2019	0

Queries table:

id	year
1	2019
2	2008
3	2009
7	2018
7	2019
7	2020
13	2019

Output:

id	year	npv
1	2019	113
2	2008	121
3	2009	12
7	2018	0
7	2019	0
7	2020	30
13	2019	40

Explanation:

The npv value of (7, 2018) is not present in the NPV table, we consider it 0.
The npv values of all other queries can be found in the NPV table.

Q32.

Table: Employees

Column Name	Type
id	int
name	varchar

id is the primary key for this table.
Each row of this table contains the id and the name of an employee in a company.

Table: EmployeeUNI

Column Name	Type
id	int
unique_id	int

(id, unique_id) is the primary key for this table.
Each row of this table contains the id and the corresponding unique id of an employee in the company.

Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just show null.
Return the result table in any order.
The query result format is in the following example.

Input:
Employees table:

id	name
1	Alice
7	Bob
11	Meir
90	Winston
3	Jonathan

EmployeeUNI table:

id	unique_id
3	1
11	2
90	3

Output:

unique_id	name
null	Alice
null	Bob
2	Meir
3	Winston
1	Jonathan

Explanation:
Alice and Bob do not have a unique ID, We will show null instead.
The unique ID of Meir is 2.
The unique ID of Winston is 3.
The unique ID of Jonathan is 1.

Q33.

Table: Users

Column Name	Type
id	int
name	varchar

id is the primary key for this table.
name is the name of the user.

Table: Rides

Column Name	Type
id	int
user_id	int
distance	int

id is the primary key for this table.
user_id is the id of the user who travelled the distance "distance".

Write an SQL query to report the distance travelled by each user.
Return the result table ordered by travelled_distance in descending order, if two or more users travelled the same distance, order them by their name in ascending order.
The query result format is in the following example.

Input:
Users table:

id	name
1	Alice
2	Bob
3	Alex
4	Donald
7	Lee

13	Jonathan
19	Elvis

Rides table:

id	user_id	distance
1	1	120
2	2	317
3	3	222
4	7	100
5	13	312
6	19	50
7	7	120
8	19	400
9	7	230

Output:

name	travelled_distance
Elvis	450
Lee	450
Bob	317
Jonathan	312
Alex	222
Alice	120
Donald	0

Explanation:
Elvis and Lee travelled 450 miles, Elvis is the top traveller as his name is alphabetically smaller than Lee.
Bob, Jonathan, Alex, and Alice have only one ride and we just order them by the total distances of the ride.
Donald did not have any rides, the distance travelled by him is 0.

Q34.

Table: Products

Column Name	Type
product_id	int
product_name	varchar
product_category	varchar

product_id is the primary key 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

There is no primary key for this table. It may have duplicate rows.
product_id is a foreign key to the Products table.
unit is the number of products ordered in order_date.

Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount.
Return result table in any order.
The query result format is in the following example.

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

Q35.

Table: Movies

Column Name	Type
movie_id	int
title	varchar

movie_id is the primary key for this table.
The title is the name of the movie.

Table: Users

Column Name	Type
user_id	int
name	varchar

user_id is the primary key for this table.

Table: MovieRating

Column Name	Type
movie_id	int
user_id	int
rating	int
created_at	date

(movie_id, user_id) is the primary key for this table.
This table contains the rating of a movie by a user in their review.
created_at is the user's review date.

Write an SQL query to:

- Find the name of the user who has rated the greatest number of movies. In case of a tie, return the lexicographically smaller user name.
- Find the movie name with the highest average rating in February 2020. In case of a tie, return the lexicographically smaller movie name.

The query result format is in the following example.

Input:

Movies table:

movie_id	title
1	Avengers
2	Frozen 2
3	Joker

Users table:

user_id	name
1	Daniel
2	Monica
3	Maria
4	James

MovieRating table:

movie_id	user_id	rating	created_at
1	1	3	2020-01-12
1	2	4	2020-02-11
1	3	2	2020-02-12
1	4	1	2020-01-01
2	1	5	2020-02-17
2	2	2	2020-02-01
2	3	2	2020-03-01
3	1	3	2020-02-22
3	2	4	2020-02-25

Output:

results
Daniel
Frozen 2

Explanation:
Daniel and Monica have rated 3 movies ("Avengers", "Frozen 2" and "Joker") but Daniel is smaller lexicographically.
Frozen 2 and Joker have a rating average of 3.5 in February but Frozen 2 is smaller lexicographically.

Q36.

Table: Users

Column Name	Type
id	int
name	varchar

id is the primary key for this table.
name is the name of the user.

Table: Rides

Column Name	Type
id	int
user_id	int
distance	int

id is the primary key for this table.
user_id is the id of the user who travelled the distance "distance".

Write an SQL query to report the distance travelled by each user.
Return the result table ordered by travelled_distance in descending order, if two or more users travelled the same distance, order them by their name in ascending order.
The query result format is in the following example.

Input:
Users table:

id	name
1	Alice
2	Bob
3	Alex
4	Donald
7	Lee
13	Jonathan
19	Elvis

Rides table:

id	user_id	distance
1	1	120
2	2	317
3	3	222
4	7	100
5	13	312
6	19	50

7	7	120
8	19	400
9	7	230

Output:

name	travelled_distance
Elvis	450
Lee	450
Bob	317
Jonathan	312
Alex	222
Alice	120
Donald	0

Explanation:
Elvis and Lee travelled 450 miles, Elvis is the top traveller as his name is alphabetically smaller than Lee.
Bob, Jonathan, Alex, and Alice have only one ride and we just order them by the total distances of the ride.
Donald did not have any rides, the distance travelled by him is 0.

Q37.

Table: Employees

Column Name	Type
id	int
name	varchar

id is the primary key for this table.
Each row of this table contains the id and the name of an employee in a company.

Table: EmployeeUNI

Column Name	Type
id	int
unique_id	int

(id, unique_id) is the primary key for this table.
Each row of this table contains the id and the corresponding unique id of an employee in the company.

Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just show null.
Return the result table in any order.

The query result format is in the following example.

Input:
Employees table:

id	name
1	Alice
7	Bob
11	Meir
90	Winston
3	Jonathan

EmployeeUNI table:

id	unique_id
3	1
11	2
90	3

Output:

unique_id	name
null	Alice
null	Bob
2	Meir
3	Winston
1	Jonathan

Explanation:
Alice and Bob do not have a unique ID, We will show null instead.
The unique ID of Meir is 2.
The unique ID of Winston is 3.
The unique ID of Jonathan is 1.

Q38.

Table: Departments

Column Name	Type
id	int
name	varchar

id is the primary key of this table.
The table has information about the id of each department of a university.

Table: Students

Column Name	Type
id	int
name	varchar
department_id	int

id is the primary key of this table.
The table has information about the id of each student at a university and the id of the department he/she studies at.
Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exist.
Return the result table in any order.
The query result format is in the following example.

Input:
Departments table:

id	name
1	Electrical Engineering
7	Computer Engineering
13	Business Administration

Students table:

id	name	department_id
23	Alice	1
1	Bob	7
5	Jennifer	13
2	John	14
4	Jasmine	77
3	Steve	74
6	Luis	1
8	Jonathan	7
7	Daiana	33
11	Madelynn	1

Output:

id	name
2	John
7	Daiana
4	Jasmine
3	Steve

Explanation:
John, Daiana, Steve, and Jasmine are enrolled in departments 14, 33, 74, and 77 respectively.
Department 14, 33, 74, and 77 do not exist in the Departments table.

Q39.

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.

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

Explanation:
Users 1 and 2 had 2 calls and the total duration is 70 (59 + 11).
Users 1 and 3 had 1 call and the total duration is 20.
Users 3 and 4 had 4 calls and the total duration is 999 (100 + 200 + 200 + 499).

Q40.

Table: Prices

Column Name	Type
product_id	int
start_date	date
end_date	date
price	int

(product_id, start_date, end_date) is the primary key for this table.
Each row of this table indicates the price of the product_id in the period from start_date to end_date.
For each product_id there will be no two overlapping periods. That means there will be no two intersecting periods for the same product_id.

Table: UnitsSold

Column Name	Type
product_id	int
purchase_date	date
units	int

There is no primary key for this table, it may contain duplicates.
Each row of this table indicates the date, units, and product_id of each product sold.

Write an SQL query to find the average selling price for each product. average_price should be rounded to 2 decimal places.
Return the result table in any order.
The query result format is in the following example.

Input:
Prices table:

product_id	start_date	end_date	price
1	2019-02-17	2019-02-28	5
1	2019-03-01	2019-03-22	20
2	2019-02-01	2019-02-20	15
2	2019-02-21	2019-03-31	30

UnitsSold table:

product_id	purchase_date	units
1	2019-02-25	100
1	2019-03-01	15
2	2019-02-10	200
2	2019-03-22	30

Output:

product_id	average_price
1	6.96
2	16.96

Explanation:
Average selling price = Total Price of Product / Number of products sold.
Average selling price for product 1 = ((100 * 5) + (15 * 20)) / 115 = 6.96
Average selling price for product 2 = ((200 * 15) + (30 * 30)) / 230 = 16.96

Q41.

Table: Warehouse

Column Name	Type
name	varchar
product_id	int
units	int

(name, product_id) is the primary key for this table.
Each row of this table contains the information of the products in each warehouse.

Table: Products

Column Name	Type
product_id	int
product_name	varchar
Width	int
Length	int
Height	int

product_id is the primary key for this table.
Each row of this table contains information about the product dimensions (Width, Length, and Height) in feets of each product.

Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse.
Return the result table in any order.
The query result format is in the following example.

Input:
Warehouse table:

name	product_id	units
LCHouse1	1	1
LCHouse1	2	10
LCHouse1	3	5
LCHouse2	1	2
LCHouse2	2	2
LCHouse3	4	1

Products table:

product_id	product_name	Width	Length	Height
1	LC-TV	5	50	40
2	LC-KeyChain	5	5	5
3	LC-Phone	2	10	10
4	LC-T-Shirt	4	10	20

Output:

warehouse_name	volume
LCHouse1	12250
LCHouse2	20250
LCHouse3	800

Q42.

Table: Sales

Column Name	Type
sale_date	date
fruit	enum
sold_num	int

(sale_date, fruit) is the primary key for this table.
This table contains the sales of "apples" and "oranges" sold each day.

Write an SQL query to report the difference between the number of apples and oranges sold each day.
Return the result table ordered by sale_date.
The query result format is in the following example.

Input:
Sales table:

sale_date	fruit	sold_num
2020-05-01	apples	10
2020-05-01	oranges	8
2020-05-02	apples	15
2020-05-02	oranges	15
2020-05-03	apples	20
2020-05-03	oranges	0
2020-05-04	apples	15
2020-05-04	oranges	16

Output:

sale_date	diff
2020-05-01	2
2020-05-02	0
2020-05-03	20
2020-05-04	-1

Explanation:
Day 2020-05-01, 10 apples and 8 oranges were sold (Difference 10 - 8 = 2).
Day 2020-05-02, 15 apples and 15 oranges were sold (Difference 15 - 15 = 0).
Day 2020-05-03, 20 apples and 0 oranges were sold (Difference 20 - 0 = 20).
Day 2020-05-04, 15 apples and 16 oranges were sold (Difference 15 - 16 = -1).

Q43.

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 fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of players.
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	2016-03-02	0
3	4	2018-07-03	5

Output:

fraction
0.33

Explanation:
Only the player with id 1 logged back in after the first day he had logged in so the answer is 1/3 = 0.33

Q44.

Table: Employee

Column Name	Type
id	int
name	varchar
department	varchar
managerId	int

id is the primary key column for this table.
Each row of this table indicates the name of an employee, their department, and the id of their manager.
If managerId is null, then the employee does not have a manager.
No employee will be the manager of themself.

Write an SQL query to report the managers with at least five direct reports.
Return the result table in any order.
The query result format is in the following example.

Input:
Employee table:

id	name	department	managerId
101	John	A	None
102	Dan	A	101
103	James	A	101
104	Amy	A	101
105	Anne	A	101
106	Ron	B	101

Output:

name
John

Q45.

Table: Student

Column Name	Type
student_id	int
student_name	varchar
gender	varchar
dept_id	int

student_id is the primary key column for this table.
dept_id is a foreign key to dept_id in the Department tables.
Each row of this table indicates the name of a student, their gender, and the id of their department.

Table: Department

Column Name	Type
dept_id	int
dept_name	varchar

dept_id is the primary key column for this table.
Each row of this table contains the id and the name of a department.

Write an SQL query to report the respective department name and number of students majoring in each department for all departments in the Department table (even ones with no current students). Return the result table ordered by student_number in descending order. In case of a tie, order them by dept_name alphabetically.
The query result format is in the following example.

Input:
Student table:

student_id	student_name	gender	dept_id
1	Jack	M	1
2	Jane	F	1
3	Mark	M	2

Department table:

dept_id	dept_name
1	Engineering
2	Science
3	Law

Output:

dept_name	student_number
Engineering	2
Science	1
Law	0

Q46.

Table: Customer

Column Name	Type
customer_id	int
product_key	int

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

Table: Product

Column Name	Type
product_key	int

product_key is the primary key column for this table.

Write an SQL query to report the customer ids from the Customer table that bought all the products in the Product table.
Return the result table in any order.
The query result format is in the following example.

Input:
Customer table:

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

Product table:

product_key
5
6

Output:

customer_id
1
3

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

Q47.

Table: Project

Column Name	Type
project_id	int
employee_id	int

(project_id, employee_id) is the primary key of this table.
employee_id is a foreign key to the Employee table.
Each row of this table indicates that the employee with employee_id is working on the project with project_id.

Table: Employee

Column Name	Type
employee_id	int
name	varchar
experience_years	int

employee_id is the primary key of this table.
Each row of this table contains information about one employee.

Write an SQL query that reports the most experienced employees in each project. In case of a tie, report all employees with the maximum number of experience years.
Return the result table in any order.
The query result format is in the following example.

Input:
Project table:

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

Employee table:

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

Output:

project_id	employee_id
1	1
1	3
2	1

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

Q48.

Table: Books

Column Name	Type
book_id	int
name	varchar
available_from	date

book_id is the primary key of this table.

Table: Orders

Column Name	Type
order_id	int
book_id	int
quantity	int
dispatch_date	date

order_id is the primary key of this table.
book_id is a foreign key to the Books table.

Write an SQL query that reports the books that have sold less than 10 copies in the last year, excluding books that have been available for less than one month from today. Assume today is 2019-06-23.
Return the result table in any order.
The query result format is in the following example.

Input:

Books table:

book_id	name	available_from
1	"Kalila And Demna"	2010-01-01
2	"28 Letters"	2012-05-12
3	"The Hobbit"	2019-06-10
4	"13 Reasons Why"	2019-06-01
5	"The Hunger Games"	2008-09-21

Q49.

Table: Enrollments

Column Name	Type
student_id	int
course_id	int
grade	int

(student_id, course_id) is the primary key of this table.

Write a SQL query to find the highest grade with its corresponding course for each student. In case of a tie, you should find the course with the smallest course_id.
Return the result table ordered by student_id in ascending order.
The query result format is in the following example.

Input:
Enrollments table:

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

Output:

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

Q50.

Table: Teams

Column Name	Type
team_id	int
team_name	varchar

team_id is the primary key of this table.
Each row of this table represents a single football team.

Table: Matches

Column Name	Type
match_id	int
host_team	int
guest_team	int
host_goals	int
guest_goals	int

match_id is the primary key of this table.
Each row is a record of a finished match between two different teams.
Teams host_team and guest_team are represented by their IDs in the Teams table (team_id), and they scored host_goals and guest_goals goals, respectively.

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

Q51.

Column Name	Type
name	varchar
continent	varchar
area	int
population	int
gdp	int

name is the primary key column for this table.
Each row of this table gives information about the name of a country, the continent to which it belongs, its area, the population, and its GDP value.
A country is big if:

- it has an area of at least three million (i.e., 3000000 km2), or
- it has a population of at least twenty-five million (i.e., 25000000).

Write an SQL query to report the name, population, and area of the big countries.
Return the result table in any order.
The query result format is in the following example.

Input:
World table:

name	continent	area	population	gdp
Afghanistan	Asia	652230	25500100	20343000000
Albania	Europe	28748	2831741	12960000000
Algeria	Africa	2381741	37100000	188681000000
Andorra	Europe	468	78115	3712000000
Angola	Africa	1246700	20609294	100990000000

Output:

name	population	area
Afghanistan	25500100	652230
Algeria	37100000	2381741

Q52.

Table: Customer

Column Name	Type
id	int
name	varchar
referee_id	int

id is the primary key column for this table.
Each row of this table indicates the id of a customer, their name, and the id of the customer who referred them.

Write an SQL query to report the names of the customer that are not referred by the customer with id = 2.
Return the result table in any order.
The query result format is in the following example.

Input:
Customer table:

id	name	referee_id
1	Will	null
2	Jane	null
3	Alex	2
4	Bill	null
5	Zack	1
6	Mark	2

Output:

name
Will
Jane
Bill
Zack

Q53.

Table: Customers

Column Name	Type
id	int
name	varchar

id is the primary key column for this table.
Each row of this table indicates the ID and name of a customer.

Table: Orders

Column Name	Type
id	int
customerId	int

id is the primary key column for this table.
customerId is a foreign key of the ID from the Customers table.
Each row of this table indicates the ID of an order and the ID of the customer who ordered it.

Write an SQL query to report all customers who never order anything.
Return the result table in any order.
The query result format is in the following example.

Input:
Customers table:

id	name
1	Joe
2	Henry
3	Sam
4	Max

Orders table:

id	customerId
1	3
2	1

Output:

Customers
Henry
Max

Q54.

Table: Employee

Column Name	Type
employee_id	int
team_id	int

employee_id is the primary key for this table.
Each row of this table contains the ID of each employee and their respective team.

Write an SQL query to find the team size of each of the employees.
Return result table in any order.
The query result format is in the following example.

Input:
Employee Table:

employee_id	team_id
1	8
2	8
3	8
4	7
5	9
6	9

Output:

employee_id	team_size
1	3
2	3
3	3
4	1
5	2
6	2

Explanation:
Employees with Id 1,2,3 are part of a team with team_id = 8.
Employee with Id 4 is part of a team with team_id = 7.
Employees with Id 5,6 are part of a team with team_id = 9.

Q55

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-yyyyyyy' where xxx is the country code (3 characters) and yyyyyyy 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, caller 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

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.

Q56.

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 device that is first logged in for each player.
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	2016-05-02	6
2	3	2017-06-25	1
3	1	2016-03-02	0
3	4	2018-07-03	5

Output:

player_id	device_id
1	2
2	3
3	1

Q57.

Table: Orders

Column Name	Type
order_number	int
customer_number	int

order_number is the primary key for this table.
This table contains information about the order ID and the customer ID.

Write an SQL query to find the customer_number for the customer who has placed the largest number of orders.
The test cases are generated so that exactly one customer will have placed more orders than any other customer.
The query result format is in the following example.

Input:
Orders table:

order_number	customer_numbe
1	1
2	2
3	3
4	3

Output:

customer_number
3

Explanation:
The customer with number 3 has two orders, which is greater than either customer 1 or 2 because each of them only has one order.
So the result is customer_number 3.

Follow up: What if more than one customer has the largest number of orders, can you find all the customer_number in this case?

Q58.

Table: Cinema

Column Name	Type
seat_id	int
free	bool

seat_id is an auto-increment primary key column for this table.
Each row of this table indicates whether the ith seat is free or not. 1 means free while 0 means occupied.

Write an SQL query to report all the consecutive available seats in the cinema.
Return the result table ordered by seat_id in ascending order.
The test cases are generated so that more than two seats are consecutively available.
The query result format is in the following example.

Input:
Cinema table:

seat_id	free
1	1
2	0
3	1
4	1
5	1

Output:

seat_id
3
4
5

Q59.

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 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 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	int
sales_id	int
amount	int

order_id is the primary key column for this table.
com_id is a foreign key to com_id from the Company table.
sales_id is a foreign key 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 an SQL query to report 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 query result format is in the following example.

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	10000
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.

Q60.

Table: Triangle

Column Name	Type
x	int
y	int
z	int

(x, y, z) is the primary key column for this table.
Each row of this table contains the lengths of three line segments.

Write an SQL query to report for every three line segments whether they can form a triangle.
Return the result table in any order.
The query result format is in the following example.

Input:
Triangle table:

x	y	z
13	15	30
10	20	15

Output:

x	y	z	triangle
13	15	30	No
10	20	15	Yes

Q61.

Table: Point

Column Name	Type
x	int

x is the primary key column for this table.
Each row of this table indicates the position of a point on the X-axis.

Write an SQL query to report the shortest distance between any two points from the Point table.
The query result format is in the following example.

Input:
Point table:

x
-1
0
2

Output:

shortest
1

Explanation:

The shortest distance is between points -1 and 0 which is |(-1) - 0| = 1.

Follow up: How could you optimise your query if the Point table is ordered in ascending order?

Q62.

Table: ActorDirector

Column Name	Type
actor_id	int
director_id	int
timestamp	int

timestamp is the primary key column for this table.

Write a SQL query for a report that provides the pairs (actor_id, director_id) where the actor has cooperated with the director at least three times.
Return the result table in any order.
The query result format is in the following example.

Input:
ActorDirector table:

actor_id	director_id	timestamp
1	1	0
1	1	1
1	1	2
1	2	3
1	2	4
2	1	5
2	1	6

Output:

actor_id	director_id
1	1

Explanation:
The only pair is (1, 1) where they cooperated exactly 3 times.

Q63.

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 the 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.

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_name	year	price
Nokia	2008	5000
Nokia	2009	5000
Apple	2011	9000

Explanation:
From sale_id = 1, we can conclude that Nokia was sold for 5000 in the year 2008.
From sale_id = 2, we can conclude that Nokia was sold for 5000 in the year 2009.
From sale_id = 7, we can conclude that Apple was sold for 9000 in the year 2011.

Q64.

Table: Project

Column Name	Type
project_id	int
employee_id	int

(project_id, employee_id) is the primary key of this table.
employee_id is a foreign key to the Employee table.
Each row of this table indicates that the employee with employee_id is working on the project with project_id.

Table: Employee

Column Name	Type
employee_id	int
name	varchar
experience_years	int

employee_id is the primary key of this table.
Each row of this table contains information about one employee.

Write an SQL query that reports the average experience years of all the employees for each project, rounded to 2 digits.
Return the result table in any order.
The query result format is in the following example.

Input:
Project table:

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

Employee table:

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

Output:

project_id	average_years
1	2
2	2.5

Explanation:
The average experience years for the first project is (3 + 2 + 1) / 3 = 2.00 and for the second project is (3 + 2) / 2 = 2.50

Q65.

Table: Product

Column Name	Type
product_id	int
product_name	varchar
unit_price	int

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

Table: Sales

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

This table has no primary key, it can have repeated rows.
product_id is a foreign key to the Product table.
Each row of this table contains some information about one sale.

Write an SQL query that reports the best seller by total sales price, If there is a tie, report them all.
Return the result table in any order.
The query result format is in the following example.

Input:
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

Output:

seller_id
1
3

Explanation: Both sellers with id 1 and 3 sold products with the most total price of 2800.

Q66.

Table: Product

Column Name	Type
product_id	int
product_name	varchar
unit_price	int

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

Table: Sales

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

This table has no primary key, it can have repeated rows.
product_id is a foreign key to the Product table.
Each row of this table contains some information about one sale.

Write an SQL query that reports the buyers who have bought S8 but not iPhone. Note that S8 and iPhone are products present in the Product table.
Return the result table in any order.
The query result format is in the following example.

Input:
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	1	3	2019-06-02	1	800
3	3	3	2019-05-13	2	2800

Output:

buyer_id
1

Explanation:
The buyer with id 1 bought an S8 but did not buy an iPhone. The buyer with id 3 bought both.

Orders table:

order_id	book_id	quantity	dispatch_date
1	1	2	2018-07-26
2	1	1	2018-11-05
3	3	8	2019-06-11
4	4	6	2019-06-05
5	4	5	2019-06-20
6	5	9	2009-02-02
7	5	8	2010-04-13

Output:

book_id	name
1	"Kalila And Demna"
2	"28 Letters"
5	"The Hunger Games"

Q67.

Table: Customer

Column Name	Type
customer_id	int
name	varchar
visited_on	date
amount	int

(customer_id, visited_on) is the primary key for this table.
This table contains data about customer transactions in a restaurant.
visited_on is the date on which the customer with ID (customer_id) has visited the restaurant.
amount is the total paid by a customer.

You are the restaurant owner and you want to analyse a possible expansion (there will be at least one customer every day).
Write an SQL query to compute the moving average of how much the customer paid in a seven days window (i.e., current day + 6 days before). average_amount should be rounded to two decimal places.
Return result table ordered by visited_on in ascending order.
The query result format is in the following example.

Input:
Customer table:

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

Output:

visited_on	amount	average_amount
2019-01-07	860	122.86

2019-01-08	840	120
2019-01-09	840	120
2019-01-10	1000	142.86

Explanation:
1st moving average from 2019-01-01 to 2019-01-07 has an average_amount of (100 + 110 + 120 + 130 + 110 + 140 + 150)/7 = 122.86
2nd moving average from 2019-01-02 to 2019-01-08 has an average_amount of (110 + 120 + 130 + 110 + 140 + 150 + 80)/7 = 120
3rd moving average from 2019-01-03 to 2019-01-09 has an average_amount of (120 + 130 + 110 + 140 + 150 + 80 + 110)/7 = 120
4th moving average from 2019-01-04 to 2019-01-10 has an average_amount of (130 + 110 + 140 + 150 + 80 + 110 + 130 + 150)/7 = 142.86

Q68.

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.

Q69.

Table: Logs

Column Name	Type
log_id	int

log_id is the primary key for this table.
Each row of this table contains the ID in a log Table.

Write an SQL query to find the start and end number of continuous ranges in the table Logs.
Return the result table ordered by start_id.
The query result format is in the following example.

Input:
Logs table:

log_id
1
2
3
7
8
10

Output:

start_id	end_id
1	3
7	8
10	10

Explanation:
The result table should contain all ranges in table Logs.
From 1 to 3 is contained in the table.
From 4 to 6 is missing in the table
From 7 to 8 is contained in the table.
Number 9 is missing from the table.
Number 10 is contained in the table.

Q70.

Table: Students

Column Name	Type
student_id	int
student_name	varchar

student_id is the primary key for this table.
Each row of this table contains the ID and the name of one student in the school.

Table: Subjects

Column Name	Type
subject_name	varchar

subject_name is the primary key for this table.
Each row of this table contains the name of one subject in the school.

Table: Examinations

Column Name	Type
student_id	int
subject_name	varchar

There is no primary key for this table. It may contain duplicates.
Each student from the Students table takes every course from the Subjects table.
Each row of this table indicates that a student with ID student_id attended the exam of subject_name.

Write an SQL query to find the number of times each student attended each exam.
Return the result table ordered by student_id and subject_name.
The query result format is in the following example.

Input:
Students table:

student_id	student_name
1	Alice
2	Bob
13	John
6	Alex

Subjects table:

subject_name
Math
Physics
Programming

Examinations table:

student_id	subject_name
1	Math
1	Physics
1	Programming
2	Programming
1	Physics
1	Math
13	Math
13	Programming
13	Physics
2	Math
1	Math

Output:

student_id	student_name	subject_name	attended_exams
1	Alice	Math	3
1	Alice	Physics	2
1	Alice	Programming	1
2	Bob	Math	1
2	Bob	Physics	0

2	Bob	Programming	1
6	Alex	Math	0
6	Alex	Physics	0
6	Alex	Programming	0
13	John	Math	1
13	John	Physics	1
13	John	Programming	1

Explanation:
The result table should contain all students and all subjects.
Alice attended the Math exam 3 times, the Physics exam 2 times, and the Programming exam 1 time.
Bob attended the Math exam 1 time, the Programming exam 1 time, and did not attend the Physics exam.
Alex did not attend any exams.
John attended the Math exam 1 time, the Physics exam 1 time, and the Programming exam 1 time.

Q71.

Table: Employees

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.

Input:
Employees 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

Explanation:
The head of the company is the employee with employee_id 1.
The employees with employee_id 2 and 77 report their work directly to the head of the company.
The employee with employee_id 4 reports their work indirectly to the head of the company 4 --> 2 --> 1.
The employee with employee_id 7 reports their work indirectly to the head of the company 7 --> 4 --> 2 --> 1.
The employees with employee_id 3, 8, and 9 do not report their work to the head of the company directly or indirectly.

Q72.

Table: Transactions

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

id is the primary key of this table.
The table has information about incoming transactions.
The state column is an enum of type ["approved", "declined"].

Write an SQL query to find for each month and country, the number of transactions and their total amount, the number of approved transactions and their total amount.
Return the result table in any order.
The query result format is in the following example.

Input:
Transactions table:

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

Output:

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

Q73.

Table: Actions

Column Name	Type
user_id	int
post_id	int
action_date	date
action	enum
extra	varchar

There is no primary key for this table, it may have duplicate rows.
The action column is an ENUM type of ('view', 'like', 'reaction', 'comment', 'report', 'share').
The extra column has optional information about the action, such as a reason for the report or a type of reaction.

Table: Removals

Column Name	Type
post_id	int
remove_date	date

post_id is the primary key of this table.
Each row in this table indicates that some post was removed due to being reported or as a result of an admin review.

Write an SQL query to find the average daily percentage of posts that got removed after being reported as spam, rounded to 2 decimal places.
The query result format is in the following example.

Input:
Actions table:

user_id	post_id	action_date	action	extra
1	1	2019-07-01	view	null
1	1	2019-07-01	like	null
1	1	2019-07-01	share	null
2	2	2019-07-04	view	null
2	2	2019-07-04	report	spam
3	4	2019-07-04	view	null
3	4	2019-07-04	report	spam
4	3	2019-07-02	view	null
4	3	2019-07-02	report	spam

5	2	2019-07-03	view	null
5	2	2019-07-03	report	racism
5	5	2019-07-03	view	null
5	5	2019-07-03	report	racism

Removals table:

post_id	remove_date
2	2019-07-20
3	2019-07-18

Output:

average_daily_percent
75

Explanation:
The percentage for 2019-07-04 is 50% because only one post of two spam reported posts were removed.
The percentage for 2019-07-02 is 100% because one post was reported as spam and it was removed.
The other days had no spam reports so the average is (50 + 100) / 2 = 75%
Note that the output is only one number and that we do not care about the remove dates.

Q74.

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 fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of players.
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	2016-03-02	0
3	4	2018-07-03	5

Output:

fraction
0.33

Explanation:
Only the player with id 1 logged back in after the first day he had logged in so the answer is 1/3 = 0.33

Q75.

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 fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of players.
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	2016-03-02	0
3	4	2018-07-03	5

Output:

fraction
0.33

Explanation:
Only the player with id 1 logged back in after the first day he had logged in so the answer is 1/3 = 0.33

Q76.

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:

- 0% 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, 10000] 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.

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	Nyan Cat	3300
3	15	Morning Cat	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
3	13	Nyan Cat	2508
3	15	Morning Cat	5911

Explanation:
For company 1, Max salary is 21300. Employees in company 1 have taxes = 49%
For company 2, Max salary is 700. Employees in company 2 have taxes = 0%
For company 3, Max salary is 7777. Employees in company 3 have taxes = 24%
The salary after taxes = salary - (taxes percentage / 100) * salary
For example, Salary for Morning Cat (3, 15) after taxes = 7777 - 7777 * (24 / 100) = 7777 - 1866.48 = 5910.52, which is rounded to 5911.

Q77.

Table Variables:

Column Name	Type
name	varchar
value	int

name is the primary key for this table.
This table contains the stored variables and their values.

Table Expressions:

Column Name	Type
left_operand	varchar
operator	enum
right_operand	varchar

(left_operand, operator, right_operand) is the primary key for this table.
This table contains a boolean expression that should be evaluated.
operator is an enum that takes one of the values ('<', '>', '=')
The values of left_operand and right_operand are guaranteed to be in the Variables table.

Write an SQL query to evaluate the boolean expressions in Expressions table.
Return the result table in any order.
The query result format is in the following example.

Input:
Variables table:

name	value
x	66
y	77

Expressions table:

left_operand	operator	right_operand
x	>	y
x	<	y
x	=	y
y	>	x
y	<	x
x	=	x

Output:

left_operand	operator	right_operand	value
x	>	y	false
x	<	y	true
x	=	y	false
y	>	x	true
y	<	x	false
x	=	x	true

Explanation:
As shown, you need to find the value of each boolean expression in the table using the variables table.

Q78.

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-yyyyyyy' where xxx is the country code (3 characters) and yyyyyyy 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

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.

Q79.

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

Q80.

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

Q81.

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

Q82.

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

Q83.

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

Q84.

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.

Q85.

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

Q86.

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

Q87.
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

Q88

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.

Q89.

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-yyyyyyy' where xxx is the country code (3 characters) and yyyyyyy 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.

Q90.

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.

Q91.

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.

Q92.

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

Q93.

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

Q94.

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.

Q95.

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.

Q96.

You're given two tables on Spotify users' streaming data. songs_history table contains the historical streaming data and songs_weekly table contains the current week's streaming data.
Write a query to output the user id, song id, and cumulative count of song plays as of 4 August 2022 sorted in descending order.

Hint- Use group by

Definitions:

- song_weekly table currently holds data from 1 August 2022 to 7 August 2022.

- songs_history table currently holds data up to to 31 July 2022. The output should include the historical data in this table.

Assumption:

- There may be a new user or song in the songs_weekly table not present in the songs_history table.

songs_history Table:

Column Name	Type
history_id	integer
user_id	integer
song_id	integer
song_plays	integer

songs_history Example Input:

history_id	user_id	song_id	song_plays
10011	777	1238	11
12452	695	4520	1

song_plays: Refers to the historical count of streaming or song plays by the user.

songs_weekly Table:

Column Name	Type
user_id	integer
song_id	integer
listen_time	datetime

songs_weekly Example Input:

user_id	song_id	listen_time
777	1238	08/01/2022 12:00:00
695	4520	08/04/2022 08:00:00

125	9630	08/04/2022 16:00:00
695	9852	08/07/2022 12:00:00

Example Output:

user_id	song_id	song_plays
777	1238	12
695	4520	2
125	9630	1

Q97.

New TikTok users sign up with their emails, so each signup requires a text confirmation to activate the new user's account.
Write a query to find the confirmation rate of users who confirmed their signups with text messages. Round the result to 2 decimal places.

Hint- Use Joins

Assumptions:

- A user may fail to confirm several times with text. Once the signup is confirmed for a user, they will not be able to initiate the signup again.
- A user may not initiate the signup confirmation process at all.

`emails` Table:

Column Name	Type
email_id	integer
user_id	integer
signup_date	datetime

`emails` Example Input:

email_id	user_id	signup_date
125	7771	06/14/2022 00:00:00

236	6950	07/01/2022 00:00:00
433	1052	07/09/2022 00:00:00

texts Table:

Column Name	Type
text_id	integer
email_id	integer
signup_action	varchar

texts Example Input:

text_id	email_id	signup_action
6878	125	Confirmed
6920	236	Not Confirmed
6994	236	Confirmed

Example Output:

confirm_rate
0.67

Q98.

The table below contains information about tweets over a given period of time. Calculate the 3-day rolling average of tweets published by each user for each date that a tweet was posted. Output the user id, tweet date, and rolling averages rounded to 2 decimal places.
Hint- Use Count and group by

Important Assumptions:

- Rows in this table are *consecutive* and ordered by date.
- Each row represents a different day
- A day that does not correspond to a row in this table is not counted. The most recent day is the next row above the current row.

Note: Rolling average is a metric that helps us analyze data points by creating a series of averages based on different subsets of a dataset. It is also known as a moving average, running average, moving mean, or rolling mean.

`tweets` Table:

Column Name	Type
tweet_id	integer
user_id	integer
tweet_date	timestamp

`tweets` Example Input:

tweet_id	user_id	tweet_date
214252	111	06/01/2022 12:00:00
739252	111	06/01/2022 12:00:00
846402	111	06/02/2022 12:00:00
241425	254	06/02/2022 12:00:00
137374	111	06/04/2022 12:00:00

Example Output:

user_id	tweet_date	rolling_avg_3days
111	06/01/2022 12:00:00	2.00
111	06/02/2022 12:00:00	1.50
111	06/04/2022 12:00:00	1.33
254	06/02/2022 12:00:00	1.00

Q99.

Assume you are given the tables below containing information on Snapchat users, their ages, and their time spent sending and opening snaps. Write a query to obtain a breakdown of the time spent sending vs. opening snaps (as a percentage of total time spent on these activities) for each age group.

Hint- Use join and case

Output the age bucket and percentage of sending and opening snaps. Round the percentage to 2 decimal places.

Notes:

- You should calculate these percentages:
    - time sending / (time sending + time opening)
    - time opening / (time sending + time opening)
- To avoid integer division in percentages, multiply by 100.0 and not 100.

`activities` Table:

Column Name	Type
activity_id	integer
user_id	integer
activity_type	string ('send', 'open', 'chat')
time_spent	float
activity_date	datetime

`activities` Example Input:

activity_id	user_id	activity_type	time_spent	activity_date
7274	123	open	4.50	06/22/2022 12:00:00
2425	123	send	3.50	06/22/2022 12:00:00
1413	456	send	5.67	06/23/2022 12:00:00
1414	789	chat	11.00	06/25/2022 12:00:00
2536	456	open	3.00	06/25/2022 12:00:00

`age_breakdown` Table:

Column Name	Type
user_id	integer
age_bucket	string ('21-25', '26-30', '31-25')

`age_breakdown` Example Input:

user_id	age_bucket
123	31-35
456	26-30
789	21-25

Example Output:

age_bucket	send_perc	open_perc
26-30	65.40	34.60
31-35	43.75	56.25

Q100 .

The LinkedIn Creator team is looking for power creators who use their personal profile as a company or influencer page. This means that if someone's Linkedin page has more followers than all the companies they work for, we can safely assume that person is a Power Creator. Keep in mind that if a person works at multiple companies, we should take into account the company with the most followers.

Level - Medium
Hint- Use join and group by

Write a query to return the IDs of these LinkedIn power creators in ascending order.

Assumptions:

- A person can work at multiple companies.
- In the case of multiple companies, use the one with largest follower base.

`personal_profiles` Table:

Column Name	Type
profile_id	integer
name	string
followers	integer

`personal_profiles` Example Input:

profile_id	name	followers
1	Nick Singh	92,000
2	Zach Wilson	199,000
3	Daliana Liu	171,000
4	Ravit Jain	107,000
5	Vin Vashishta	139,000
6	Susan Wojcicki	39,000

`employee_company` Table:

Column Name	Type
personal_profile_id	integer
company_id	integer

`employee_company` Example Input:

personal_profile_id	company_id
1	4
1	9
2	2
3	1
4	3
5	6
6	5

`company_pages` Table:

Column Name	Type
company_id	integer

name	string
followers	integer

company_pages Example Input:

company_id	name	followers
1	The Data Science Podcast	8,000
2	Airbnb	700,000
3	The Ravit Show	6,000
4	DataLemur	200
5	YouTube	1,6000,000
6	DataScience.Vin	4,500
9	Ace The Data Science Interview	4479

Example Output:

profile_id
1
3
4
5

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:

Column	Type
ID	Integer
Name	String
Marks	Integer

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

ID	Name	Marks
1	Ashley	81
2	Samantha	75
4	Julia	76
3	Belvet	84

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:

Column	Type
A	Integer
B	Integer
C	Integer

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

A	B	C
20	20	23
20	20	20
20	21	22
13	14	30

Sample Output
Isosceles
Equilateral
Scalene
Not A Triangle

Explanation
Values in the tuple(20,20,23) form an Isosceles triangle, because A ≡ B.
Values in the tuple(20,20,20) form an Equilateral triangle, because A ≡ B ≡ C . Values in the tuple(20,21,22) form a Scalene triangle, because A ≠ B ≠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:

Column	Type
ID	Integer
Name	String
Salary	Integer

Note: Salary is per month.
Constraints
1000<salary < 10^5
Sample Input

ID	Name	Salary
1	Kristeen	1420
2	Ashley	2006
3	Julia	2210
4	Maria	3000

Sample Output
2061

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

ID	Name	Salary
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 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.

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:

Column	Type
Name	String
Occupation	String

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

Sample Input

An OCCUPATIONS table that contains the following records:

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

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:

Column	Type
Name	String
Occupation	String

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

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

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.

Column	Type
N	Integer
P	Integer

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:

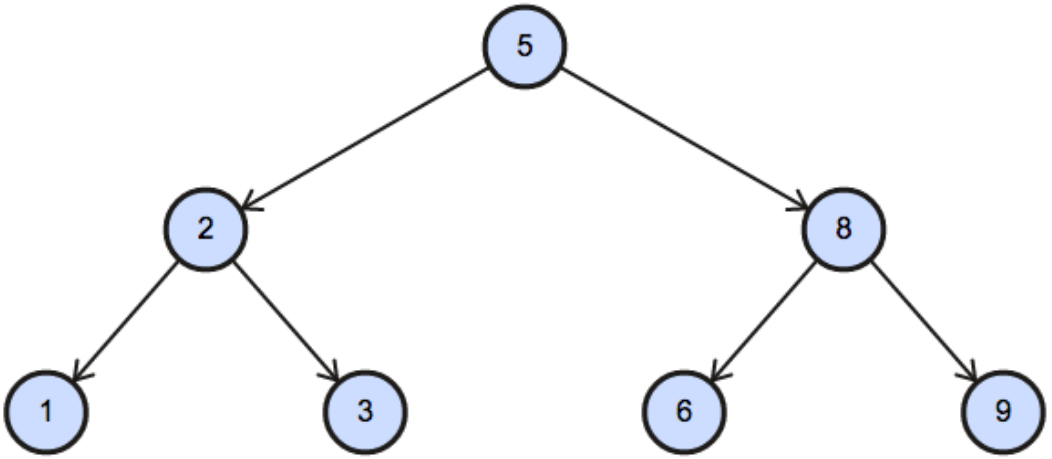
- Root: If node is root node.
- Leaf: If node is leaf node.
- Inner: If node is neither root nor leaf node.

Sample Input

N	P
1	2
3	2
6	8
9	8
2	5
8	5
5	null

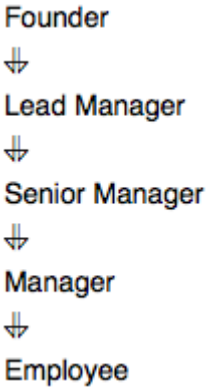
Sample Output
1 Leaf
2 Inner
3 Leaf
5 Root
6 Leaf
8 Inner
9 Leaf

Explanation
The Binary Tree below illustrates the sample:



Q111 .

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

Column	Type
company_code	String
founder	String

- Lead_Manager: The lead_manager_code is the code of the lead manager, and the company_code is the code of the working company.

Column	Type
lead_manager_code	String
company_code	String

- 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 working company.

Column	Type
senior_manager_code	String
lead_manager_code	String
company_code	String

- 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 working company.

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

Sample Input

Company Table:

company_code	founder
C1	Monika
C2	Samantha

Lead_Manager Table:

lead_manager_code	company_code
LM1	C1
LM2	C2

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 ≤ 1000) and then do the same thing to create Level M, and then filter by M ≤ L and then group by L having count(case when L/M = truc(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 ≤ 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

Column	Type
ID	Integer
Name	String
Marks	Integer

The STUDENTS table is described as follows:
The Name column only contains uppercase (A-Z) and lowercase (a-z) letters.
Sample Input

ID	Name	Marks
1	Ashley	81
2	Samantha	75
4	Julia	76
3	Belvet	84

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:

Column	Type
A	Integer
B	Integer
C	Integer

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

A	B	C
20	20	23
20	20	20
20	21	22
13	14	30

Sample Output
Isosceles
Equilateral
Scalene
Not A Triangle
Explanation
Values in the tuple(20,20,23) form an Isosceles triangle, because A ≡ B.
Values in the tuple(20,20,20) form an Equilateral triangle, because A ≡ B ≡ C . Values in the tuple(20,21,22) form a Scalene triangle, because A ≠ B ≠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-yyyyyyy' where xxx is the country code (3 characters) and yyyyyyy 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:

Column	*Type*
ID	*Integer*
Name	*String*
Salary	*Integer*

Note: Salary is per month.
Constraints
1000<salary < 10^5
Sample Input

ID	*Name*	*Salary*
1	*Kristeen*	*1420*
2	*Ashley*	*2006*
3	*Julia*	*2210*
4	*Maria*	*3000*

Sample Output
2061

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

ID	Name	Salary
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:

Column	Type
Name	String
Occupation	String

Occupation will only contain one of the following values: Doctor, Professor, Singer or Actor.
Sample Input
An OCCUPATIONS table that contains the following records:

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

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:

Column	Type
Name	String
Occupation	String

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

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

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.

Column	*Type*
N	*Integer*
P	*Integer*

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:

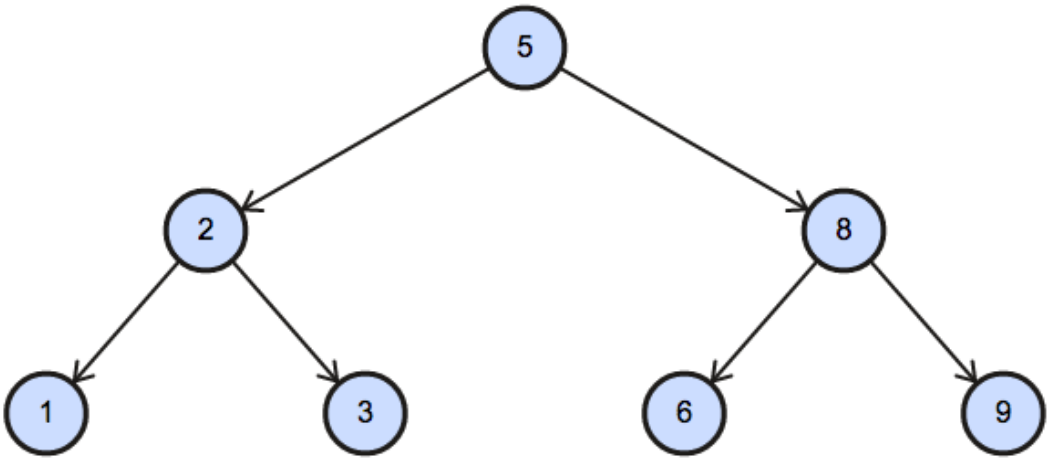
- Root: If node is root node.
- Leaf: If node is leaf node.
- Inner: If node is neither root nor leaf node.

Sample Input

N	*P*
1	*2*
3	*2*
6	*8*
9	*8*
2	*5*
8	*5*
5	*null*

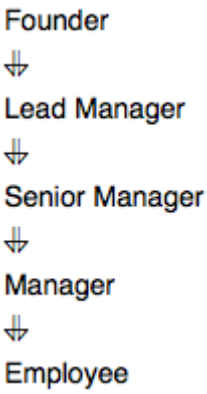
Sample Output
1 Leaf
2 Inner
3 Leaf
5 Root
6 Leaf
8 Inner
9 Leaf

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 company.

Column	Type
company_code	String
founder	String

- Lead_Manager: The lead_manager_code is the code of the lead manager, and the company_code is the code of the working company.

Column	Type
lead_manager_code	String
company_code	String

- 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 working company.

Column	Type
senior_manager_code	String
lead_manager_code	String
company_code	String

- 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 working company.

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

Sample Input

Company Table:

company_code	founder
C1	Monika
C2	Samantha

Lead_Manager Table:

lead_manager_code	company_code
LM1	C1
LM2	C2

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.

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 ≤ Y1.
Level - Medium
Source - Hackerrank
Hint - Use group by and having clause .
Sample Input

X	Y
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).

Column	Type
ID	Integer
Name	String

Students

Column	Type
ID	Integer
Friend_ID	Integer

Friends

Column	Type
ID	Integer
Salary	Float

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

ID	Friend_ID
1	2
2	3
3	4
4	1

Friends

ID	Name
1	Ashley
2	Samantha
3	Julia
4	Scarlet

Students

ID	Salary
1	15.20
2	10.06
3	11.55
4	12.12

Packages

Sample Output
Samantha
Julia
Scarlet

Explanation
See the following table:

ID	1	2	3	4
Name	Ashley	Samantha	Julia	Scarlet
Salary	15.20	10.06	11.55	12.12
Friend ID	2	3	4	1
Friend Salary	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 score of the challenge for the difficulty level.

Column	Type
difficulty_level	Integer
score	Integer

- 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 to, and score is the score of the submission.

Column	Type
submission_id	Integer
hacker_id	Integer
challenge_id	Integer
score	Integer

Sample Input

Hackers Table:

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

Difficulty Table:

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

Challenges Table:

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

:

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

Column	*Type*
Task_ID	*Integer*
Start_Date	*Date*
End_Date	*Date*

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

Task_ID	Start_Date	End_Date
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
transaction_id	integer
user_id	integer
spend	decimal
transaction_date	timestamp

`user_transactions` Example Input:

transaction_id	user_id	spend	transaction_date
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

1

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 = Ad Revenue / 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
user_id	integer
product_id	integer
quantity	integer
purchase_date	datetime

`purchases`Example Input:

user_id	product_id	quantity	purchase_date
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%.