Table: Person

COLUMN NAME	TYPE
PERSONID	INT
LASTNAME	VARCHAR
FIRSTNAME	VARCHAR

This table contains information about the ID of some persons and their first and last names.

Table: ADDRESS

COLUMN NAME	TYPE
ADDRESSID	INT
PERSONID	INT
CITY	VARCHAR
STATE	VARCHAR

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

Each row of this table contains information about the city and state of one person with ID = PersonId.

Write a solution to report the first name, last name, city, and state of each person in the Person table. If the address of a personId is not present in the Address table, report null instead.

Return the result table in any order.

The result format is in the following example

Example 1:
Input:

Person table:

PERSONID	LASTNAME	FIRSTNAME
1	WANG	ALLEN
2	ALICE	BOB

Address table:

ADDRESSID	PERSONID	CITY	STATE
1	2	NEW YORK CITY	NEW YORK
2	3	LEETCODE	CALIFORNIA

Output:

FIRSTNAME	LASTNAME	CITY	STATE
ALLEN	WANG	NULL	NULL
BOB	ALICE	NEW YORK CITY	NEW YORK

Explanation:

There is no address in the address table for the pers onId = 1 so we return null in their city and state. addressId = 1 contains information about the address of personId = 2.

Solution: select

p.firstname,p.lastname,a.city,a.state from

person p left join address a on

p.personId=a.personId ;

2.Table: Employee

COLUMN NAME	TYPE
ID	INT
NAME	VARCHAR
SALARY	INT
MANAGERID	INT

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

Each row of this table indicates the ID of an employee, their name, salary, and the ID of their manager.

Write a solution to find the employees who earn more than their managers.

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

The result format is in the following example.

Example 1:

Input:

ID	NAME	SALARY	MANAGERID
1	JOE	70000	3
2	HENRY	80000	4
3	SAM	60000	NULL
4	MAX	90000	NULL

Output:

EMPLOYEEE

JOE

Explanation: Joe is the only employee who earns more than his manager.

SOLUTION:

Select e1.name as employee

From employee e1

Inner join employee e2

On e1.managerId=e2.id

Where e1.salary > e2.salary;

3.DUPLICATE EMAILS

Table: Person

COLUMN NAME	TYPE
ID	INT
EMAIL	VARCHAR

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

Each row of this table contains an email. The emails will not contain uppercase letters.

Write a solution to report all the duplicate emails. Note that it's guaranteed that the email field is not NULL.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:

Person table:

ID	EMAIL
1	a@b.com
2	c@d.com
3	a@b.com

OUTPUT:

EMAIL
a@b.com

Explanation: a@b.com is repeated two times.

SOLUTION:

```
SELECT EMAIL
FROM PERSON
GROUP BY EMAIL
HAVING COUNT(DISTINCT ID)>1;
```

4. CUSTOMER WHO NEVER ORDER

Table :CUSTOMER

COLUMN NAME	TYPE
ID	INT
NAME	VARCHAR

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

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

Table: Orders

. CUSTOMER WHO NEVER ORDER

COLUMN NAME	TYPE
ID	INT
CUSTOMERID	INT

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

customerId is a foreign key (reference columns) 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 a solution to find all customers who never order anything.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:

Customers table:

ID	NAME
1	JOE
2	HENRY
3	SAM
4	MAX

Orders table:

ID	CUSTOMERID
1	3
2	2

Output:

CUSTOMERS
HENRY
MAX

SOLUTION:

```
SELECT C.NAME AS CUSTOMERS
FROM CUSTOMERS C
WHERE C.ID NOT IN
(SELECT C.ID
FROM CUSTOMERS C
```

INNER JOIN ORDERS O

ON C.ID = O.customerID);

5.EMPLOYEE BONUS

TABLE :EMPLOYEE

COLUMN NAME	TYPE
EMPID	INT
NAME	VARCHAR
SUPERVISOR	INT
SALARY	INT

empId is the column with unique values for this table.

Each row of this table indicates the name and the ID of an employee in addition to their salary and the id of their manager**.**

Table: Bonus

COLUMN NAME	TYPE
EMPID	INT
BONUS	INT

empId is the column of unique values for this table.

empId is a foreign key (reference column) to empId from the Employee table.

Each row of this table contains the id of an employee and their respective bonus.

Write a solution to report the name and bonus amount of each employee with a bonus less than 1000.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input: Employee table:

EMPID	NAME	SUPERVISOR	SALARY
3	BRAD	NULL	4000
1	JOHN	3	1000
2	DAN	3	2000
4	THOMAS	3	4000

Bonus table:

EMPID	BONUS
2	500
4	2000

Output:

NAME	BONUS
BRAD	NULL
JOHN	NULL
DAN	500

SOLUTION:

```
SELECT E.NAME,B.BONUS
FROM EMPLOYEE E LEFT JOIN BONUS B
ON E.EMPID=B.EMPID
WHERE B.BONUS<1000
OR B.BONUS IS NULL;
```

6.FIND CUSTOMER REFREE

TABLE:CUSTOMER

COLUMN NAME	TYPE
ID	INT
NAME	VARCHAR
REFREE_ID	VARCHAR

In SQL, 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.

Find the names of the customer that are not referred by the customer with id = 2.

Return the result table in any order.

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

Example 1:

Input:

Customer table:

ID	NAME	REFREE_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

SOLUTION:

```
SELECT NAME
FROM CUSTOMER
WHERE REFREE_ID!=2
OR REFREE_ID IS NULL;
```

7. CUSTOMER PLACING THE LARGEST NUMBER OF ORDERS

Table: Orders

COLUMN_NAME	TYPE
ORDER_NUMBER	INT
CUSTOMER_NUMBER	INT

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

This table contains information about the order ID and the customer ID.

Write a solution 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 result format is in the following example.

Example 1:

Input: Orders table:

ORDER_NUMBER	CUSTOMER_NUMBER
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.

SOLUTION:

```
WITH CTE AS
(SELECT CUSTOMER_NUMBER ,COUNT(ORDER_NUMBER) AS NUMORD
FROM ORDERS
GROUP BY CUSTOMER_NUMBER)
SELECT CUSTOMER_NUMBER
```

FROM CTE

WHERE NUMORD=(SELECT MAX(NUMORD)FROM CTE);

8. BIG COUNTRIES

Table: World

COLUMN_NAME	TYPE
NAME	VARCHAR
CONTINENT	VARCHAR
AREA	INT
POPULATION	INT
GDP	BIGINT

name is the primary key (column with unique values) 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 km^2), or
- it has a population of at least twenty-five million (i.e., 25000000).

Write a solution to find the name, population, and area of the big countries.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:

World table:

NAME	CONTINENT	AREA	POPULATION	GDP
AFGANISTAN	ASIA	652230	25500100	20343000000
ALBIAN	EUROPE	28748	2831741	12960000000
ALGERIA	AFRICA	2381741	37100000	18868100000
ANDORRA	EUROPE	468	78115	371200000
ANGOLA	AFRICA	1246700	20609294	10099000000

Output:

NAME	POPULATION	AREA
AFGHANISTAN	25500100	652230
ALGERIA	37100000	2381741

SOLUTION:

```
SELECT NAME,POPULATION,AREA
FROM WORLD
WHERE AREA>=3000000
OR POPULATION>=25000000;
```

9. CLASSES MORE THAN 5 STUDENTS

Table: Courses

COLUMN_NAME	TYPE
STUDENT	VARCHAR
CLASS	VARCHAR

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

Each row of this table indicates the name of a student and the class in which they are enrolled.

Write a solution to find all the classes that have at least five students.

Return the result table in any order.

The result format is in the following example.

Explanation:

- Math has 6 students, so we include it.

- English has 1 student, so we do not include it.

- Biology has 1 student, so we do not include it.

- Computer has 1 student, so we do not include it

Example 1:

Input: Courses table

STUDENT	CLASS
A	MATH
B	ENGLISH
C	MATH
D	BIOLOGY
E	MATH
F	COMPUTER
G	MATH
H	MATH
I	MATH

Output:

CLASS
MATH

SOLUTION:

```
SELECT CLASS
FROM COURSES
GROUP BY CLASS
HAVING COUNT(STUDENT)>=5;
```

10.Triangle Judgement

Table: Triangle

COLUMN__NAME	TYPE
X	INT
Y	INT
Z	INT

In SQL, (x, y, z) is the primary key column for this table.

Each row of this table contains the lengths of three line segments.

Report for every three line segments whether they can form a triangle.

Return the result table in any order.

The result format is in the following example.

Example 1:

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

SOLUTION:

```
SELECT *,CASE WHEN  X+Y>Z  AND Y+Z>X  AND X+Z>Y
THEN  'YES'
ELSE 'NO' END AS TRIANGLE
FROM TRIANGLE;
```

11. Biggest Single Number

Table: MyNumbers

COLUMN_NAME	TYPE
NUM	INT

This table may contain duplicates (In other words, there is no primary key for this table in SQL).

Each row of this table contains an integer.

A single number is a number that appeared only once in the MyNumbers table.

Find the largest single number. If there is no single number, report null.

The result format is in the following example.

Example 1:

Input: MyNumbers table

NUM
8
8
3
3
1
4
5
6

Output:

NUM
6

Explanation: The single numbers are 1, 4, 5, and 6.

Since 6 is the largest single number, we return it.

Example 2:

Input: MyNumbers table

NUM
8
8
7
7
3
3
3

Output:

NUM
NULL

Explanation: There are no single numbers in the input table so we return null.

SOLUTION:

```
WITH CTE AS

(SELECT NUM FROM MYNUMBERS
```

GROUP BY NUM

HAVING COUNT(NUM)=1)

SELECT CASE WHEN COUNT(*)>0 THEN MAX(NUM)

ELSE NULL END AS NUM

FROM CTE;

12. NOT BORING MOVIES

Table: Cinema

COLUMN_NAME	TYPE
ID	INT
MOVIE	VARCHAR
DESCRIPTION	VARCHAR
RATING	FLOAT

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

Each row contains information about the name of a movie, its genre, and its rating. rating is a 2 decimal places float in the range [0, 10]

Write a solution to report the movies with an odd-numbered ID and a description that is not "boring".

Return the result table ordered by rating in descending order.

The result format is in the following example.

Example 1:Input: Cinema table

ID	MOVIE	DESCRIPTION	RATING
1	WAR	GREAT 3D	8.9
2	SCIENCE	FICTION	8.5
3	IRISH	BORING	6.2
4	ICE SONG	FANTACY	8.6
5	HOUSE CARD	INTERESTING	9.1

Output:

ID	MOVIE	DESCRIPTION	RATING
5	HOUSECARD	INTERESTING	9.1
1	WAR	GREAT 3D	8.9

Explanation:

We have three movies with odd-numbered IDs: 1, 3, and 5. The movie with ID = 3 is boring so we do not include it in the answer

SOLUTION:

```
SELECT  *
FROM CINEMA
WHERE MOD(ID,2)!=0
AND DESCRIPTION !='boring'
ORDER BY RATING DESC;
```

13. Swap Salary

Table: Salary

COLUMN_NAME	TYPE
ID	INT
NAME	VARCHAR
SEX	ENUM
SALARY	INT

id is the primary key (column with unique values) for this table.The sex column is ENUM (category) value of type ('m', 'f').

The table contains information about an employee.

Write a solution to swap all 'f' and 'm' values (i.e., change all 'f' values to 'm' and vice versa) with a single update statement and no intermediate temporary tables.

Note that you must write a single update statement, do not write any select statement for this problem.

The result format is in the following example.

Example 1:

Input: Salary table

ID	NAME	SEX	SALARY
1	A	m	2500
2	B	f	1500

3	**C**	**m**	**5500**
4	**D**	**f**	**500**

Output:

ID	NAME	SEX	SALARY
1	**A**	**f**	**2500**
2	**B**	**m**	**1500**
3	**C**	**f**	**5500**
4	**D**	**m**	**500**

Explanation:

(1, A) and (3, C) were changed from 'm' to 'f'.

(2, B) and (4, D) were changed from 'f' to 'm'.

SOLUTION:

```
UPDATE SALARY
SET SEX = CASE WHEN SEX ='f' THEN 'm'
WHEN SEX='m' THEN 'f' END;
```

14. Actors and Directors Who Cooperated At Least Three Times

Table: ActorDirector

COLUMN_NAME	TYPE
ACTOR_ID	INT
DIRECTOR_ID	INT
TIMESTAMP	INT

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

Write a solution to find all 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 result format is in the following example.

Example 1:

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.

SOLUTION:

```
SELECT ACTOR_ID,DIRECTOR_ID

FROM ACTORDIRECTOR

GROUP BY ACTOR_ID ,DIRECTOR_ID

HAVING COUNT(TIMESTAMP)>=3;
```

15.QUERIES QUALITY AND PERCENTAGE

Table: Queries

COLUMN_NAME	TYPE
QUERY NAME	VARCHAR
RESULT	VARVHAR
POSITION	INT
RATING	INT

This table may have duplicate rows.

This table contains information collected from some queries on a database.

The position column has a value from **1** to **500**.

The rating column has a value from **1** to **5**. Query with rating less than 3 is a poor query.

We define query quality as:

The average of the ratio between query rating and its position.

We also define poor query percentage as:

The percentage of all queries with rating less than 3.

Write a solution to find each query_name,
the quality and poor_query_percentage.

Both quality and poor_query_percentage should be rounded to 2 decimal places.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:

QUERYNAME	RESULT	POSITION	RATING
DOG	GOLDENRETRIEVER	1	5
DOG	GERMANSHEPHERD	2	5
DOG	MULE	100	1
CAT	SHIRAZI	5	2
CAT	SIMASE	3	3
CAT	SPYNX	7	4

Output:

QUERY NAME	QUALITY	POOR_QUERY_PERCENTAGE
DOG	2.50	33.33
CAT	0.66	33.33

Explanation:

Dog queries quality is ((5 / 1) + (5 / 2) + (1 / 200)) / 3 = 2.50

Dog queries poor_ query_percentage is (1 / 3) * 100 = 33.33

Cat queries quality equals ((2 / 5) + (3 / 3) + (4 / 7)) / 3 = 0.66

Cat queries poor_ query_percentage is (1 / 3) * 100 = 33.33

SOLUTION:

```
WITH CTE AS
(SELECT QUERY_NAME,RATING/POSITION AS RATIO
CASE WHEN RATING <3 THEN 1
ELSE 0 END AS QUALITY_BINARY
FROM QUERIES
SELECT QUER_NAME, ROUND(AVG(RATIO),2) AS
QUALITY, ROUND((SUM(QUALITY_BINARY)/COUNT(*))
*100,2) AS POOR_QUERY_PERCENTAGE
FROM CTE
GROUP BY QUERY_NAME;
```

16. AVREAGE SELLING PRICE

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 (combination of columns with unique values) 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
UNIT	INT

This table may contain duplicate rows.

Each row of this table indicates the date, units, and product_id of each product sold.

Write a solution to find the average selling price for each product. average_price should be rounded to 2 decimal places. If a product does not have any sold units, its average selling price is assumed to be 0.

Return the result table in any order.

The result format is in the following example.

Example 1:

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	15
2	2019-02-01	2019/02/22	20
2	2019-02-21	2019/03/31	15

UnitsSold table:

PRODUCT_ID	PURCHASED_DATE	UNITS
1	2019-02-25	100
1	2019-03-01	15
2	2019-02-10	200

2	**2019-03-22**	**30**

OUPUT:

PRODUCT_ID	AVREAGE_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

SOLUTION:

```
SELECT P.PRODUCT_ID,SUM(P.PRICE*U.UNITS)/
SUM(U.UNITS),2) AS AVERAGE_PRICE
FROM PRICES P
INNER JOIN UNITSOLD U
ON P.PRODUCT_ID =U.PRODUCT_ID
WHERE U.PURCHASE_DATE BETWEEN P.START_DATE AND
P.END_DATE
GROUP BY P.PRODUCT_ID;
```

17.WEATHER TYPE IN EACH COUNTRY

TABLE: COUNTRIES

COLUMN_NAME	TYPE
COUNTRY_ID	INT
COUNTRY_NAME	VARCHAR

country_id is the primary key (column with unique values) 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
DAY	DATE
WEATHER_STATE	INT

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

Write a solution 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 the result table in any order.
The result format is in the following example.

Example 1:

INPUT: Countries table

COUNTRY_ID	COUNTRY_NAME
2	USA
3	AUSTRLIA
7	PERU
5	CHINA
8	MOROCCO
9	SPAIN

INPUT: 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 USA in November is (15) / 1 = 15 so weather type is Cold.
Average weather_state in Austraila in November is (-2 + 0 + 3) / 3 = 0.333 so weather type is Cold.
Average weather_state in Peru in November is (25) / 1 = 25 so the weather type is Hot.
Average weather_state in China in November is (16 + 18 + 21) / 3 = 18.333 so weather type is Warm.
Average weather_state in Morocco in November is (25 + 27 + 31) / 3 = 27.667 so 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.

Solutions:

```
SELECT country_name,
       CASE
        WHEN AVG(weather_state) <= 15 THEN 'Cold'
      WHEN AVG(weather_state) >= 25 THEN 'Hot'
         ELSE 'Warm'
    END AS weather_type
FROM Weather AS w
   JOIN Countries USING (country_id)
WHERE DATE_FORMAT(day, '%Y-%m') = '2019-11'
GROUP BY 1;
```

18.FIND THE TEAM SIZE

Table: Employee

COLUMN_NAME	TYPE
EMPLOYEE_ID	INT
TEAM_ID	INT

employee_id is the primary key (column with unique values) for this table. Each row of this table contains the ID of each employee and their respective team.

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

Example 1:
Input: Employee Table

EMPLOYEE_ID	TEAM_ID
1	8
2	8
3	8
4	7
5	9
6	9

Output:

EMPLOYEE_ID	TEAM_ID
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.

SOLUTION:

```
WITH
   T AS (
      SELECT team_id, COUNT(1) AS team_size
      FROM Employee
      GROUP BY 1
   )
SELECT employee_id, team_size
FROM
   Employee
   JOIN T USING (team_id);
```

19.STUDENT AND EXAMINATION

Table: Students

COLUMN_NAME	TYPE
STUDENT_ID	INT
SUBJECT_NAME	VARCHAR

student_id is the primary key (column with unique values) 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 (column with unique values) 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_NAM	VARCHAR

There is no primary key (column with unique values) 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 a solution to find the number of times each student attended each exam.
Return the result table ordered by student_id and subject_name.
The result format is in the following example.

Example 1:
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_EXAM
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.

SOLUTION:

```
SELECT student_id, student_name, subject_name, COUNT(e.student_id)
AS attended_exams
FROM  Students JOIN Subject  LEFT JOIN Examinations AS
USING(student_id, subject_name)
GROUP BY 1, 3
ORDER BY 1, 3;
```

20.STUDENT WITH INVALID DEPARMENT

Table: Departments

COLUMN_NAME	TYPE
ID	INT
NAME	VARCHAR

In SQL, 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

In SQL, 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.

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 result format is in the following example.

Example 1:
Input: Departments table

ID	NAME
1	ELECTRICAL ENGINNERING
7	COMPUTER ENGINEERIG
13	BUSSINESS 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	LEVIS	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.

Solutions:

```
SELECT id, name
FROM Students
WHERE department_id NOT IN (SELECT id FROM Departments);
```

21.CONSECUTIVE AVAILABLE SEATS

Table: Cinema

COLUMN_NAME	TYPE
SEAT_ID	IN
FREE	BOOL

seat_id is an auto-increment column for this table.
Each row of this table indicates whether the i^{th} seat is free or not. 1 means free while 0 means occupied.

Find 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 result format is in the following example.

Example 1:
Input: Cinema table

SEAT_ID	FREE
1	1
2	0
3	1
4	1
5	1

Output:

SEAT_ID
3
4
5

SOLUTIONS:

```
WITH
  T AS (
    SELECT
      *,
      SUM(free = 1) OVER (
        ORDER BY seat_id
        ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
      ) AS cnt
    FROM Cinema
  )
SELECT seat_id
FROM T
WHERE free = 1 AND cnt > 1
ORDER BY 1;
```

22.SHORTEST DISTANCE IN A LINE

Table: Point

COLUMN_NAME	TYPE
X	INT

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

Find the shortest distance between any two points from the Point table.
The result format is in the following example.

Example 1:
Input: Point table

X
-1
0
2

Output:

SHORTEST
1

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

SOLUTION:

```
SELECT MIN(p2.x - p1.x) AS shortest
FROM
   Point AS p1
   JOIN Point AS p2 ON p1.x < p2.x;
```

23.REPORTED POSTS

Table: Actions

COLUMN_NAME	TYPE
USER_ID	INT
POST_ID	INT
ACTION_DATE	DATE
ACTION	ENUM
EXTRA	VARCHAR

This table may have duplicate rows.
The action column is an ENUM (category) 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.
extra is never NULL.

Write a solution to report the number of posts reported yesterday for each report reason. Assume today is 2019-07-05.
Return the result table in any order.
The result format is in the following example.

Example 1:
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	4	2019-07-04	VIEW	NULL
2	4	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

4	2	2019-07-04	VIEW	NULL
3	2	2019-07-04	REPORT	RACISM
5	5	2019-07-04	VIEW	NULL
5	5	2019-07-04	REPORT	RACISM

Output:

REPORT_REASON	REPORT_COUNT
SPAM	1
RACISM	2

Explanation: Note that we only care about report reasons with non-zero number of reports.

SOLUTION:

```
SELECT extra AS report_reason, COUNT(DISTINCT post_id) AS
report_count
FROM Actions
WHERE action_date = '2019-07-04' AND action = 'report'
GROUP BY 1;
```