Find user with valid email id

Table: Users

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| user_id       | int     |
| name          | varchar |
| mail          | varchar |
+---------------+---------+
```

user_id is the primary key (column with unique values) for this table.
This table contains information of the users signed up in a website. Some e-mails are invalid.

Write a solution 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 result format is in the following example.

Example 1:

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.

Group sold product by date

Table Activities:

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| sell_date   | date    |
| product     | varchar |
+-------------+---------+
```

There is no primary key (column with unique values) for this table. It may contain duplicates.
Each row of this table contains the product name and the date it was sold in a market.

Write a solution to find for each date the number of different products sold and their names.

The sold products names for each date should be sorted lexicographically.

Return the result table ordered by sell_date.

The result format is in the following example.

Example 1:

Input:
Activities table:

```
+------------+------------+
| sell_date  | product    |
+------------+------------+
| 2020-05-30 | Headphone  |
| 2020-06-01 | Pencil     |
| 2020-06-02 | Mask       |
| 2020-05-30 | Basketball |
| 2020-06-01 | Bible      |
| 2020-06-02 | Mask       |
| 2020-05-30 | T-Shirt    |
+------------+------------+
```

Output:

```
+------------+----------+------------------------------+
| sell_date  | num_sold | products                     |
+------------+----------+------------------------------+
| 2020-05-30 | 3        | Basketball,Headphone,T-shirt |
| 2020-06-01 | 2        | Bible,Pencil                 |
| 2020-06-02 | 1        | Mask                         |
+------------+----------+------------------------------+
```

Explanation:
For 2020-05-30, Sold items were (Headphone, Basketball, T-shirt), we sort them lexicographically and separate them by a comma.
For 2020-06-01, Sold items were (Pencil, Bible), we sort them lexicographically and separate them by a comma.
For 2020-06-02, the Sold item is (Mask), we just return it.

Second highest salary

Table: Employee

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| id          | int  |
| salary      | int  |
+-------------+------+
```

id is the primary key (column with unique values) for this table.
Each row of this table contains information about the salary of an employee.

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

The result format is in the following example.

Example 1:

Input:
Employee table:

```
+----+--------+
| id | salary |
+----+--------+
| 1  | 100    |
| 2  | 200    |
| 3  | 300    |
+----+--------+
```

Output:

```
+---------------------+
| SecondHighestSalary |
+---------------------+
| 200                 |
+---------------------+
```

Example 2:

Input:
Employee table:

```
+----+--------+
| id | salary |
+----+--------+
| 1  | 100    |
+----+--------+
```

Output:

```
+---------------------+
| SecondHighestSalary |
+---------------------+
| null                |
+---------------------+
```

Delete 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 **delete** all duplicate emails, keeping only one unique email with the smallest id.

For SQL users, please note that you are supposed to write a DELETE statement and not a SELECT one.

For Pandas users, please note that you are supposed to modify Person in place.

After running your script, the answer shown is the Person table. The driver will first compile and run your piece of code and then show the Person table. The final order of the Person table **does not matter**.

The result format is in the following example.

Example 1:

Input:
Person table:

```
+----+------------------+
| id | email            |
+----+------------------+
| 1  | john@example.com |
| 2  | bob@example.com  |
| 3  | john@example.com |
+----+------------------+
```

Output:

```
+----+------------------+
| id | email            |
+----+------------------+
| 1  | john@example.com |
| 2  | bob@example.com  |
+----+------------------+
```

Explanation: john@example.com is repeated two times. We keep the row with the smallest Id = 1.

Patients with a condition

Table: Patients

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| patient_id   | int     |
| patient_name | varchar |
| conditions   | varchar |
+--------------+---------+
```

patient_id is the primary key (column with unique values) for this table.
'conditions' contains 0 or more code separated by spaces.
This table contains information of the patients in the hospital.

Write a solution to find the patient_id, patient_name, and conditions of the patients who have Type I Diabetes. Type I Diabetes always starts with DIAB1 prefix.

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

The result format is in the following example.

Example 1:

Input:
Patients table:

```
+------------+--------------+--------------+
| patient_id | patient_name | conditions   |
+------------+--------------+--------------+
| 1          | Daniel       | YFEV COUGH   |
| 2          | Alice        |              |
| 3          | Bob          | DIAB100 MYOP |
| 4          | George       | ACNE DIAB100 |
| 5          | Alain        | DIAB201      |
+------------+--------------+--------------+
```

Output:

```
+------------+--------------+--------------+
| patient_id | patient_name | conditions   |
+------------+--------------+--------------+
| 3          | Bob          | DIAB100 MYOP |
| 4          | George       | ACNE DIAB100 |
+------------+--------------+--------------+
```

Explanation: Bob and George both have a condition that starts with DIAB1.