Table: `Stadium`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| visit_date    | date    |
| people        | int     |
+---------------+---------+
visit_date is the primary key for this table.
Each row of this table contains the visit date and visit id to the stadium with the number o
No two rows will have the same visit_date, and as the id increases, the dates increase as we
```

Write an SQL query to display the records with three or more rows with **consecutive** `id`'s, and the number of people is greater than or equal to 100 for each.

Return the result table ordered by `visit_date` in **ascending order**.

The query result format is in the following example.

Example 1:**

```
Input:
Stadium table:
+------+------------+-----------+
| id   | visit_date | people    |
+------+------------+-----------+
| 1    | 2017-01-01 | 10        |
| 2    | 2017-01-02 | 109       |
| 3    | 2017-01-03 | 150       |
| 4    | 2017-01-04 | 99        |
| 5    | 2017-01-05 | 145       |
| 6    | 2017-01-06 | 1455      |
| 7    | 2017-01-07 | 199       |
| 8    | 2017-01-09 | 188       |
+------+------------+-----------+
Output:
+------+------------+-----------+
| id   | visit_date | people    |
+------+------------+-----------+
| 5    | 2017-01-05 | 145       |
| 6    | 2017-01-06 | 1455      |
| 7    | 2017-01-07 | 199       |
| 8    | 2017-01-09 | 188       |
```

```
+------+------------+-----------+
Explanation:
The four rows with ids 5, 6, 7, and 8 have consecutive ids and each of them has &gt;= 100 pe
The rows with ids 2 and 3 are not included because we need at least three consecutive ids.
```