Medium

Table: UserVisits

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| user_id     | int  |
| visit_date  | date |
+-------------+------+
```
This table does not have a primary key, it might contain duplicate rows.
This table contains logs of the dates that users visited a certain retailer.

Assume today's date is '2021-1-1'.

Write a solution that will, for each user_id, find out the largest window of days between each visit and the one right after it (or today if you are considering the last visit).

Return the result table ordered by user_id.

The query result format is in the following example.

Example 1:

Input:
UserVisits table:
```
+---------+------------+
| user_id | visit_date |
+---------+------------+
| 1       | 2020-11-28 |
| 1       | 2020-10-20 |
| 1       | 2020-12-3  |
| 2       | 2020-10-5  |
| 2       | 2020-12-9  |
| 3       | 2020-11-11 |
+---------+------------+
```
Output:
```
+---------+---------------+
| user_id | biggest_window|
+---------+---------------+
| 1       | 39            |
| 2       | 65            |
| 3       | 51            |
+---------+---------------+
```
Explanation:
For the first user, the windows in question are between dates:
   - 2020-10-20 and 2020-11-28 with a total of 39 days.
   - 2020-11-28 and 2020-12-3 with a total of 5 days.
   - 2020-12-3 and 2021-1-1 with a total of 29 days.
Making the biggest window the one with 39 days.
For the second user, the windows in question are between dates:
   - 2020-10-5 and 2020-12-9 with a total of 65 days.
   - 2020-12-9 and 2021-1-1 with a total of 23 days.
Making the biggest window the one with 65 days.
For the third user, the only window in question is between dates 2020-11-11 and 2021-1-1 with a total of 51 days.

```
# Write your MySQL query statement below
WITH CTE AS (SELECT *, LEAD(visit_date,1,'2021-01-01')OVER (PARTITION BY user_id order by
visit_date) AS L
FROM UserVisits)

SELECT user_id,MAX(DATEDIFF(L, visit_date)) AS biggest_window
FROM CTE
GROUP BY user_id
```