Sellers With No Sales

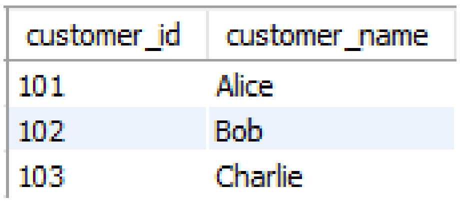
Problem Statement:

Write a query to report the names of all sellers who did not make any sales in `2020`.

- Return the result table ordered by `seller_name` in ascending order.

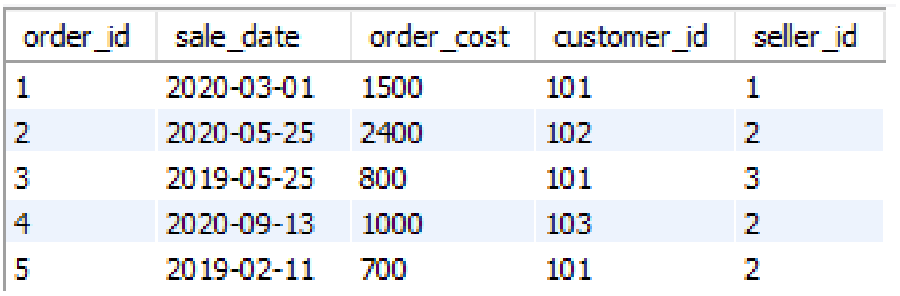
Sample Input:

Table: Customer



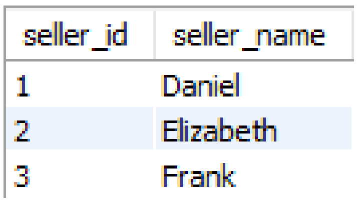
customer_id	customer_name
101	Alice
102	Bob
103	Charlie

Table: Orders



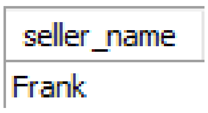
order_id	sale_date	order_cost	customer_id	seller_id
1	2020-03-01	1500	101	1
2	2020-05-25	2400	102	2
3	2019-05-25	800	101	3
4	2020-09-13	1000	103	2
5	2019-02-11	700	101	2

Table: Seller



seller_id	seller_name
1	Daniel
2	Elizabeth
3	Frank

Sample output:



seller_name
Frank

Explanation:

- Daniel made 1 sale in March 2020.

- Elizabeth made 2 sales in 2020 and 1 sale in 2019.
- Frank made 1 sale in 2019 but no sales in 2020.

Ans :

```
SELECT seller_name
FROM Seller
WHERE seller_id NOT IN (
    SELECT DISTINCT seller_id
    FROM Orders
    WHERE YEAR(sale_date) = '2020'
)
ORDER BY seller_name ASC;
```

Qn2 : Customers With Positive Revenue

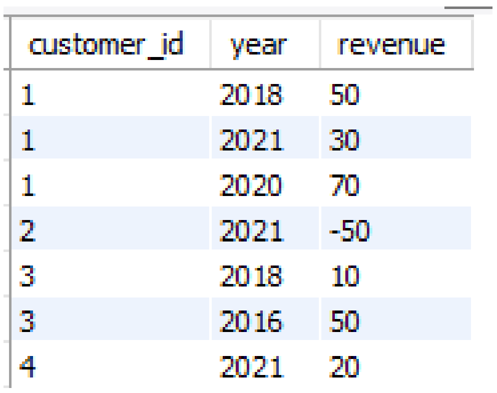
Problem Statement:

Write a query to report the customers with postive revenue in the year 2021.

- Return the result table ordered by `customer_id` in ascending order.

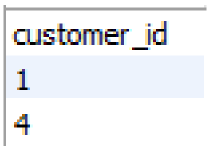
Sample Input:

Table: Customers



customer_id	year	revenue
1	2018	50
1	2021	30
1	2020	70
2	2021	-50
3	2018	10
3	2016	50
4	2021	20

Sample output:



customer_id
1
4

Explanation:

- Customer 1 has revenue equal to 30 in the year 2021.
- Customer 2 has revenue equal to -50 in the year 2021.
- Customer 3 has no revenue in the year 2021.
- Customer 4 has revenue equal to 20 in the year 2021.
- Thus only customers 1 and 4 have positive revenue in the year 2021.

```
Ans : SELECT customer_id
FROM Customers
WHERE year = '2021'
GROUP BY customer_id
HAVING SUM(revenue) >= 0
ORDER BY customer_id
```

Qn 3: Candidate Winning

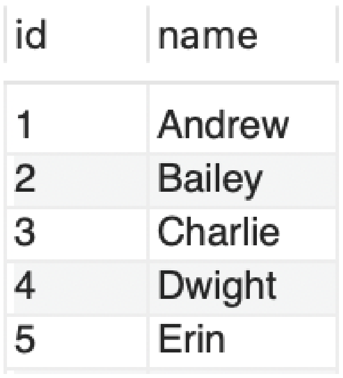
Problem Description:

Write a query to report the name of the winning candidate (i.e., the candidate who got the largest number of votes).

- The test cases are generated so that **exactly one candidate** wins the election.

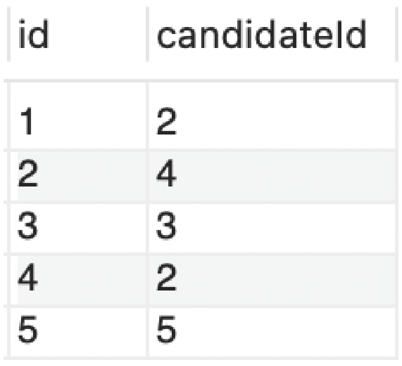
Sample Input:

Table: **Candidate**



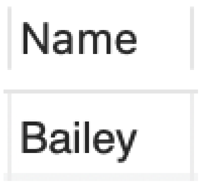
id	name
1	Andrew
2	Bailey
3	Charlie
4	Dwight
5	Erin

Table: **Vote**



id	candidateId
1	2
2	4
3	3
4	2
5	5

Sample Output:



Name
Bailey

Sample Explanation:

- Candidate Bailey has 2 votes. Candidates Charlie, Dwight, and Erin have 1 vote each.
- The winner is Candidate Bailey.

```
Ans: SELECT name AS 'Name'
FROM Candidate JOIN
(SELECT candidateId
FROM Vote
GROUP BY candidateId
ORDER BY COUNT(*) DESC
LIMIT 1) AS winner
WHERE Candidate.id = winner.candidateId;
```

Qn 4: Biggest Window Between Visits

Problem Statement:

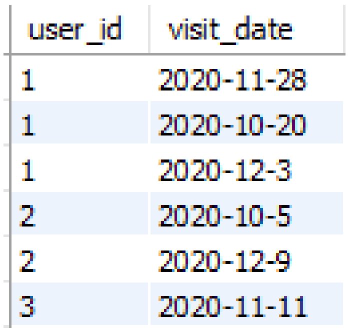
Assume today's date is `'2021-1-1'`.

Write a query 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` in ascending order.

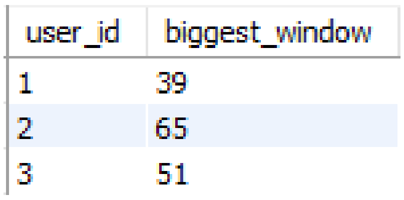
Sample Input:

Table: uservisits



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

Sample 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 the dates 2020-11-11 and 2021-1-1 with a total of 51 days.

```
Ans : SELECT user_id, MAX(diff) AS biggest_window
FROM
(
SELECT user_id,
DATEDIFF(LEAD(visit_date, 1, '2021-01-01') OVER(PARTITION BY use
r_id ORDER BY visit_date), visit_date) AS diff
FROM uservisits
)a
GROUP BY user_id
ORDER BY user_id;
```

Qn 5: Needed hours

Problem Description:

In a company, each employee must work a certain number of hours every month. Employees work in sessions.

The number of hours an employee worked can be calculated from the sum of the number of minutes the employee worked in all of their sessions.

The number of minutes in each session is rounded up.

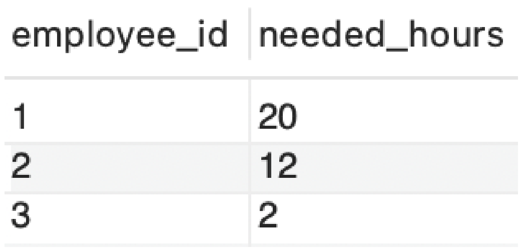
- For example, if the employee worked for 51 minutes and 2 seconds in a session, we consider it 52 minutes.
    - We use **CEIL()** function to return the smallest integer value that is bigger than or equal to a number.

Write a query to report the IDs of the employees that will be deducted. In other words, report the IDs of the employees that did not work the needed hours.

- Order the output by **employee_id** in ascending order.

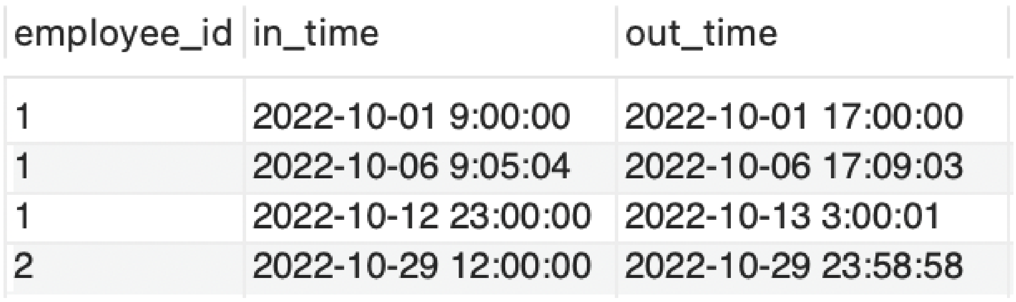
Sample Input:

Table: employees



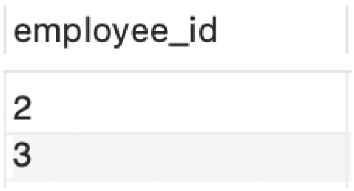
employee_id	needed_hours
1	20
2	12
3	2

Table: logs



employee_id	in_time	out_time
1	2022-10-01 9:00:00	2022-10-01 17:00:00
1	2022-10-06 9:05:04	2022-10-06 17:09:03
1	2022-10-12 23:00:00	2022-10-13 3:00:01
2	2022-10-29 12:00:00	2022-10-29 23:58:58

Sample Output:



employee_id
2
3

Sample Explanation:

- **Employee 1**:
    - Worked for three sessions:
        - On 2022-10-01, they worked for 8 hours.
        - On 2022-10-06, they worked for 8 hours and 4 minutes.
        - On 2022-10-12, they worked for 4 hours and 1 minute. Note that they worked through midnight.
        - Employee 1 worked a total of 20 hours and 5 minutes across sessions and will not be deducted.
- **Employee 2**:
    - Worked for one session:
        - On 2022-10-29, they worked for 11 hours and 59 minutes.
        - Employee 2 did not work their hours and will be deducted.
- **Employee 3**:
    - Did not work in any session.
        - Employee 3 did not work their hours and will be deducted.

Ans : SELECT employee_id
FROM
(
SELECT t.employee_id, t.needed_hours,
ROUND(SUM(t.time_minutes) / 60, 2) AS worked_hours
FROM
(
SELECT e.employee_id, e.needed_hours,
CEIL(TIMESTAMPDIFF(MINUTE, l.in_time, l.out_time)) AS time_minutes
FROM employees e
INNER JOIN logs l
USING(employee_id)
) t
GROUP BY t.employee_id,t.needed_hours
) t1
WHERE worked_hours < needed_hours
ORDER BY employee_id;