5. Weather Table - Higher Temperature than Previous Day

Objective:

Find all the dates’ id values from the Weather table where the temperature was higher than the previous

day’s temperature.

Table Structure:

- id: Unique identifier for each record.
- recordDate: The date of the temperature record.
- temperature: The temperature on that date.

Approach:

- Use a self-join on the Weather table to compare each date with the previous day.
- Apply a filter to return rows where the temperature is greater than the previous day’s temperature.

SQL Query:

```
SELECT w1.id

FROM Weather w1

JOIN Weather w2

ON w1.recordDate = DATE_ADD(w2.recordDate, INTERVAL 1 DAY)

WHERE w1.temperature > w2.temperature;
```

Accepted

Runtime: 81 ms

• Case 1

Input

Weather =

```
| id | recordDate | temperature |
| -- | ---------- | ----------- |
| 1  | 2015-01-01 | 10          |
| 2  | 2015-01-02 | 25          |
| 3  | 2015-01-03 | 20          |
| 4  | 2015-01-04 | 30          |
```

Output

```
| id |
| -- |
| 2  |
| 4  |
```

Expected

```
| Id |
| -- |
| 2  |
| 4  |
```

6. Customers Who Visited but Made No Transactions

Objective:

Identify customers who visited the mall without making any transactions and count how many times they did so.

Tables Involved:

Visits

- visit_id (unique per visit)
- customer_id

Transactions

- transaction_id (unique per transaction)
- visit_id (foreign key from Visits)
- amount

Approach:

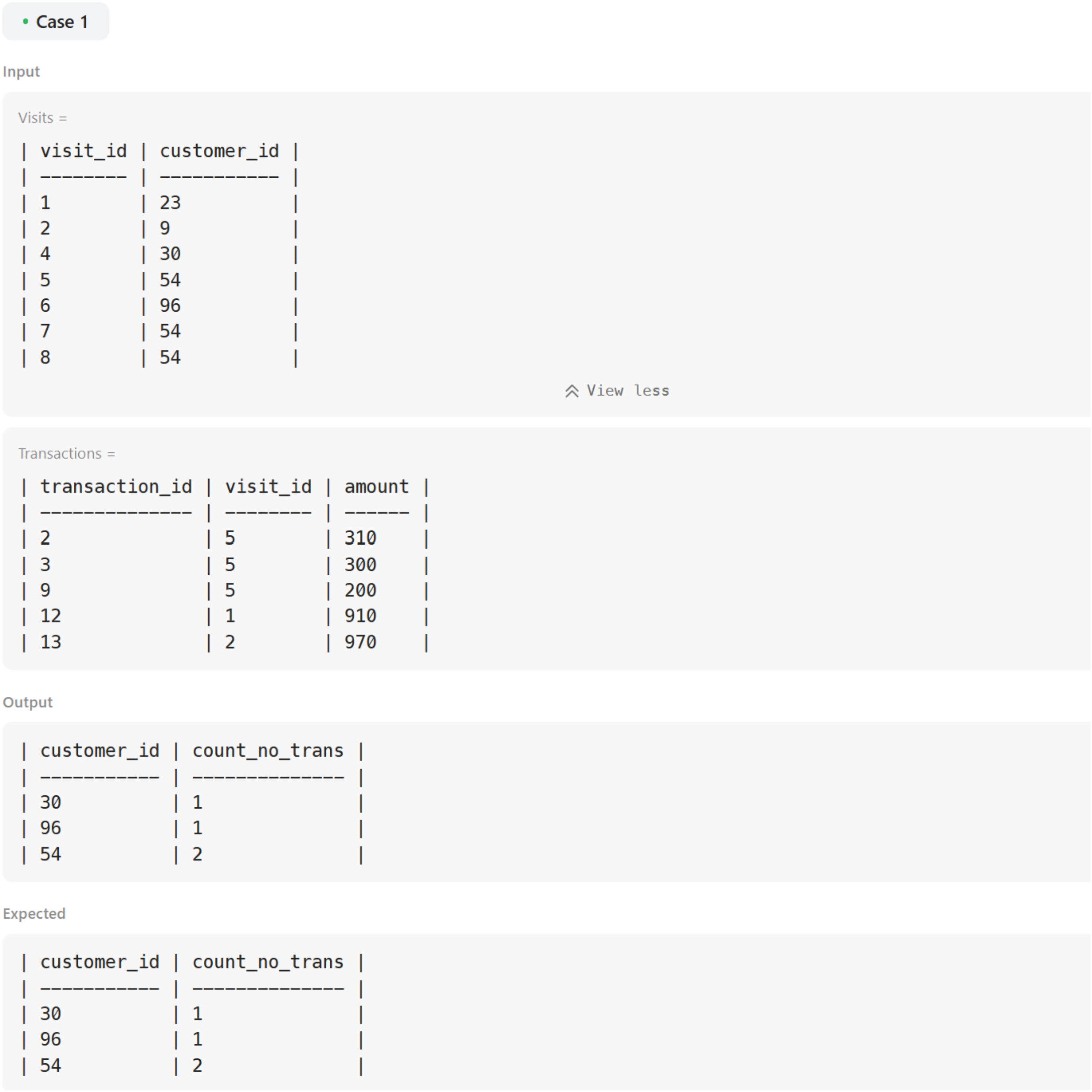
1. Left Join Visits with Transactions on visit_id to retain all visits, even those without a transaction.

2. Filter the result where transaction_id IS NULL to get only visits without transactions.

3. Group By customer_id to count how many such visits each customer made.

SQL Query:

```
SELECT v.customer_id, COUNT(*) AS count_no_trans
FROM Visits v
LEFT JOIN Transactions t
ON v.visit_id = t.visit_id
WHERE t.transaction_id IS NULL
GROUP BY v.customer_id;
```



• Case 1

Input

Visits =

```
| visit_id | customer_id |
| -------- | ----------- |
| 1        | 23          |
| 2        | 9           |
| 4        | 30          |
| 5        | 54          |
| 6        | 96          |
| 7        | 54          |
| 8        | 54          |
```

View less

Transactions =

```
| transaction_id | visit_id | amount |
| -------------- | -------- | ------ |
| 2              | 5        | 310    |
| 3              | 5        | 300    |
| 9              | 5        | 200    |
| 12             | 1        | 910    |
| 13             | 2        | 970    |
```

Output

```
| customer_id | count_no_trans |
| ----------- | -------------- |
| 30          | 1              |
| 96          | 1              |
| 54          | 2              |
```

Expected

```
| customer_id | count_no_trans |
| ----------- | -------------- |
| 30          | 1              |
| 96          | 1              |
| 54          | 2              |
```

7. Reporting Product Name, Year, and Price for Each Sale

Objective:

Retrieve a list of all product sales showing the product name, year of sale, and price per unit.

Tables Involved:

1. Sales

- sale_id (unique with year)
- product_id (foreign key)
- year
- quantity
- price (per unit)

2. Product

- product_id (primary key)
- product_name

Approach:

- Use an INNER JOIN between the Sales and Product tables on product_id.
- Select the product name, year, and price fields for each sale.

SQL Query:

```
SELECT
p.product_name,
s.year,
s.price
FROM Sales s
JOIN Product p
ON s.product_id = p.product_id;
```

- Case 1

Input

Sales =

```
| sale_id | product_id | year | quantity | price |
| ------- | ---------- | ---- | -------- | ----- |
| 1       | 100        | 2008 | 10       | 5000  |
| 2       | 100        | 2009 | 12       | 5000  |
| 7       | 200        | 2011 | 15       | 9000  |
```

Product =

```
| product_id | product_name |
| ---------- | ------------ |
| 100        | Nokia        |
| 200        | Apple        |
| 300        | Samsung      |
```

Output

```
| product_name | year | price |
| ------------ | ---- | ----- |
| Nokia        | 2009 | 5000  |
| Nokia        | 2008 | 5000  |
| Apple        | 2011 | 9000  |
```

Expected

```
| product_name | year | price |
| ------------ | ---- | ----- |
| Nokia        | 2009 | 5000  |
| Nokia        | 2008 | 5000  |
| Apple        | 2011 | 9000  |
```

8. Retrieving Unique ID for Each Employee

Objective:

Retrieve the unique ID for each employee. If an employee does not have a unique ID, show NULL instead.

- Tables Involved:
- Employees
- id (Primary key)
- name
- EmployeeUNI
- id (Primary key)
- unique_id

Approach:

- Use a LEFT JOIN between the Employees and EmployeeUNI tables on id.
- This ensures that all employees are included, and if there is no matching unique_id, it will show NULL.

SQL Query:

```
SELECT

e.id,

e.name,

eu.unique_id

FROM Employees e

LEFT JOIN EmployeeUNI eu

ON e.id = eu.id;
```

• Case 1

Input

Employees =

```
| id | name     |
| -- | -------- |
| 1  | Alice    |
| 7  | Bob      |
| 11 | Meir     |
| 90 | Winston  |
| 3  | Jonathan |
```

EmployeeUNI =

```
| id | unique_id |
| -- | --------- |
| 3  | 1         |
| 11 | 2         |
| 90 | 3         |
```

Output

```
| unique_id | name     |
| --------- | -------- |
| null      | Alice    |
| null      | Bob      |
| 2         | Meir     |
| 3         | Winston  |
| 1         | Jonathan |
```

Expected

```
| unique_id | name     |
| --------- | -------- |
| null      | Alice    |
| null      | Bob      |
| 2         | Meir     |
| 3         | Winston  |
| 1         | Jonathan |
```

9. Finding Invalid Tweets

Objective:

Identify the tweet IDs of tweets where the content length is strictly greater than 15 characters.

Table Involved:

Tweets

- tweet_id (Primary key)
- content (Text of the tweet)

Approach:

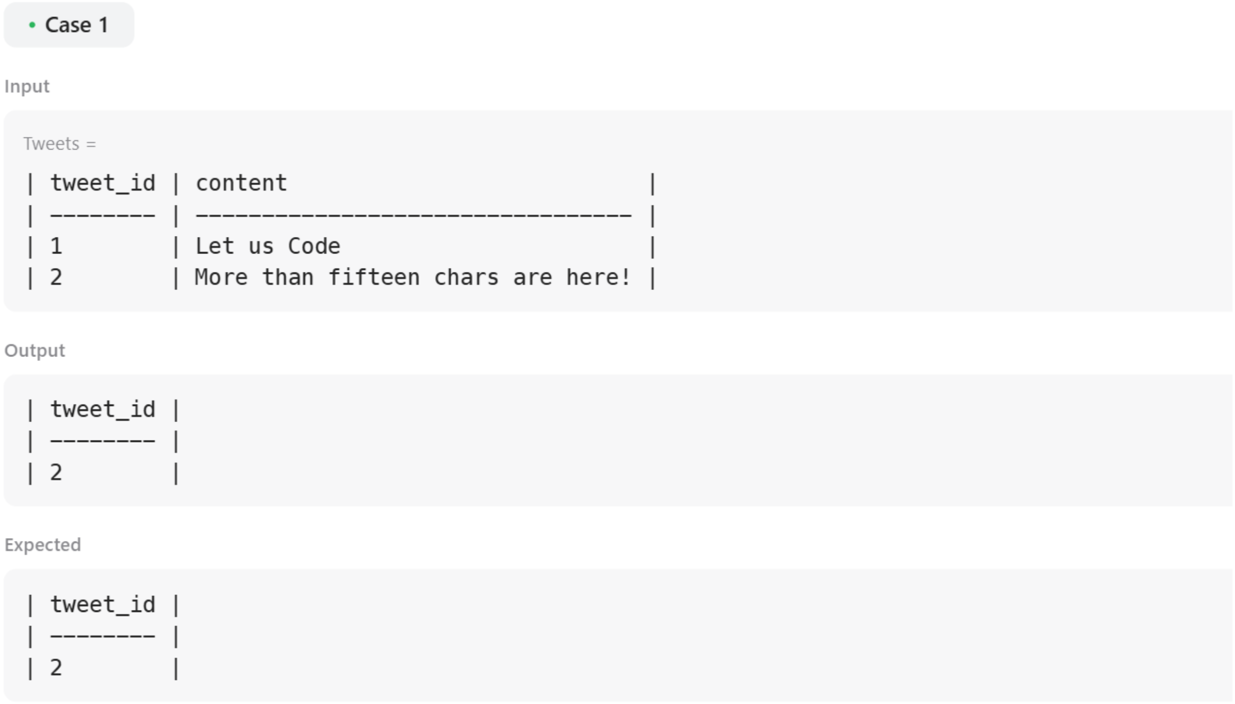
- Use the LENGTH() function to count the number of characters in the content column.
- Filter the tweets where the content length is greater than 15.

SQL Query:

```
SELECT tweet_id

FROM Tweets

WHERE LENGTH(csontent) > 15;
```



Input

Tweets =

```
| tweet_id | content                          |
| -------- | -------------------------------- |
| 1        | Let us Code                      |
| 2        | More than fifteen chars are here! |
```

Output

```
| tweet_id |
| -------- |
| 2        |
```

Expected

```
| tweet_id |
| -------- |
| 2        |
```