23_Histogram of Tweets Easy - Solution

Source - https://datalemur.com/questions/sql-histogram-tweets

Running Notes

- write a query to obtain a histogram of tweets posted per user in 2022
    - tweet count per user as the bucket
    - number of Twitter users who fall into that bucket

  eg:

tweet_bucket (tweet count per user in 2022)	users_num (the number of users who fall into this bucket)
1	10
2	4

```
WITH reference_table AS (SELECT user_id,count(user_id) count_of_
FROM tweets
WHERE EXTRACT(YEAR from tweet_date)='2022'
GROUP BY user_id)

SELECT count_of_posts_by_user_in_2022 as tweet_bucket,
count(count_of_posts_by_user_in_2022) as users_num
FROM reference_table
GROUP BY count_of_posts_by_user_in_2022
```

okay, so before 13.01.2025 I didnt have much idea about query optimization, but yesterday I spent some time reading about query optimization when I was reading the tutorial of SQL Order of Execution

Now, I realize the problems in my query -

Using
`EXTRACT(YEAR FROM tweet_date)` in the `WHERE` clause can make the query non-SARGable. This is because applying a function (such as `EXTRACT`) to a column prevents the database from using an index on that column efficiently. Instead of being able to directly use an index to filter rows, the database has to compute the result of the `EXTRACT` function for every row in the table.

Why is this a problem?

When you apply a function to a column, it typically results in a full table scan (or a full scan of the relevant partition) because the database has to process each row to evaluate the function. Indexes are usually built to allow quick lookups without transforming the indexed column's values.

A query is not SARGABLE if it uses operators or functions that prevent index usage or require full table scans. For example, using negation (NOT), wildcard (LIKE), or arithmetic (+, -, *, /) operators on indexed columns can make a query not SARGABLE.

💡 A query is SARGABLE if it uses operators and functions that can take advantage of indexes. For example, using equality (=), inequality (<>, !=), range (BETWEEN), or membership (IN) operators on indexed columns can make a query SARGABLE.

How to make it SARGable

You can rewrite the query to avoid applying a function to the column, such as by using a range condition that achieves the same result.

FINAL CODE:

```
WITH reference_table AS (SELECT user_id,count(user_id) count_of_
FROM tweets
-- standard format of date - YYYY-MM-DD
WHERE tweet_date BETWEEN '2022-01-01' AND '2022-12-31'
GROUP BY user_id)

SELECT count_of_posts_by_user_in_2022 as tweet_bucket,
count(count_of_posts_by_user_in_2022) as users_num
FROM reference_table
GROUP BY count_of_posts_by_user_in_2022
```

Now, the rewritten query using `WHERE tweet_date BETWEEN '2022-01-01' AND '2022-12-31'` is **SARGable** (Search ARGument-able), meaning it allows the database to utilize an index on the `tweet_date` column if one exists.

Written By,

Harshee Pitroda