Trigger report query working

Store_Activity (store_id, day, start_time, end_time)
Store_Status (store_id, status, timestamp_utc)
Store_Time_Zone (store_id, timezone)

- First I used a query to get all unique store_id from all tables.

```
WITH all_store_ids AS (
        SELECT store_id FROM public."Store_Status"
        UNION
        SELECT store_id FROM public."Store_Activity"
        UNION
        SELECT store_id FROM public."Store_Time_Zone"
    )
```

- Next I converted all the local starting and end time to UST by joining the Store_Actvity and Store_Time_Zone. Also handled cases like if no data is available convert time to ‘America/Chicago’ and also handles case that if Store_Activity is available for a store_id, the default start time is 00:00:00 and end time is 23:59:59.

```
store_activity_utc AS (
        SELECT DISTINCT
          asi.store_id,
          gs.day AS day,
          COALESCE(
            (sa.start_time_local AT TIME ZONE COALESCE(stz.timezone_str,
'America/Chicago'))::time,
            '00:00:00'::time
          ) AS start_time_utc,
          COALESCE(
            (sa.end_time_local AT TIME ZONE COALESCE(stz.timezone_str,
'America/Chicago'))::time,
            '23:59:59'::time
          ) AS end_time_utc
        FROM
          all_store_ids asi
        CROSS JOIN
          generate_series(0, 6) AS gs(day)
        LEFT JOIN
          public."Store_Activity" sa ON asi.store_id = sa.store_id AND gs.day =
sa.day
        LEFT JOIN
          public."Store_Time_Zone" stz ON asi.store_id = stz.store_id
    )
```

- Now I calculated the max timestamp. I have considered this timestamp to be the current timestamp and use this timestamp to generate the report.

```
max_timestamp AS (
        SELECT TIMESTAMP '2023-01-18 18:13:22.47922' AS target_timestamp
    ),
```

- Now I have filtered store status to get only the timestamp which is within the week of current timestamp. This gives me all Store_Status for a week.

```
filtered_store_status AS (
        SELECT *
        FROM public."Store_Status"
        WHERE timestamp_utc > (SELECT target_timestamp FROM
max_timestamp)
    )
```

- Now I calculate all the valid timestamp ie all the timestamp converted to day and check with Status_Activity and map them to their day.

```
valid_timestamps AS (
        SELECT DISTINCT ON (ss.store_id, sa.day)
          ss.store_id,
          ss.timestamp_utc AS valid_timestamp,
          (EXTRACT(ISODOW FROM ss.timestamp_utc) - 1) AS day, -- Adjust day
of week
          ss.status,
          sa.start_time_utc,
          sa.end_time_utc
        FROM
          filtered_store_status ss
        JOIN
          store_activity_utc sa ON ss.store_id = sa.store_id
        WHERE
          ss.timestamp_utc::time BETWEEN sa.start_time_utc AND
sa.end_time_utc
        ORDER BY
          ss.store_id, sa.day, ss.timestamp_utc -- Ensure uniqueness
    )
```

- Now we calculate the total uptime and downtime for a day. Uptime for a day is sum of difference between start and end time of day. So I pre calculated the total uptime for each day.

```
uptime_downtime_per_day AS (
        SELECT
            tu.store_id,
            tu.day,
            tu.start_time_utc,
            tu.end_time_utc,
            COALESCE(tup.total_uptime_hours, 0) AS uptime_hours
        FROM
            (
                SELECT DISTINCT store_id, day, start_time_utc, end_time_utc
                FROM store_activity_utc
            ) AS tu
        LEFT JOIN
            total_uptime_per_day tup ON tu.store_id = tup.store_id AND tu.day =
tup.day
    ),
```

- Now I calculated the count of all the active and inactive status for a day.

```
    status_counts AS (
        SELECT
            sa.store_id,
            sa.day,
            COALESCE(SUM(CASE WHEN vt.status IS NOT NULL THEN 1 ELSE 0
END), 0) AS total_entries,
            COALESCE(SUM(CASE WHEN vt.status = 'inactive' THEN 1 ELSE 0
END), 0) AS cnt_inactive

        FROM
            store_activity_utc sa
        LEFT JOIN
            valid_timestamps vt ON sa.store_id = vt.store_id AND sa.day = vt.day AND
sa.start_time_utc = vt.start_time_utc AND sa.end_time_utc = vt.end_time_utc
        GROUP BY
            sa.store_id, sa.day
    ),
```

- Now I also add the last status and timestamp of each day. Last status will be helpful in determining the uptime and downtime last hour.

```
uptime_downtime_per_day AS (
        SELECT
            sa.store_id,
            sa.day,
```

```
            LAST_VALUE(vt.status) OVER (PARTITION BY sa.store_id, sa.day
ORDER BY vt.valid_timestamp DESC) AS last_status,
            LAST_VALUE(vt.valid_timestamp) OVER (PARTITION BY sa.store_id,
sa.day ORDER BY vt.valid_timestamp DESC) AS last_status_timestamp
        FROM
            store_activity_utc sa
        LEFT JOIN
            valid_timestamps vt ON sa.store_id = vt.store_id AND sa.day = vt.day AND
sa.start_time_utc = vt.start_time_utc AND sa.end_time_utc = vt.end_time_utc
    ),
```

- Now we start the calculation of uptime last hour. I will just see the last_status of each day for that store_id. If last status is ‘inactive or NULL’ , uptime is 0 and downtime is 60 min else uptime is 60 and downtime is 0 mins.

```
uptime_last_hour AS (
        SELECT
            asi.store_id,
            CASE
                WHEN lspd.last_status IS NULL THEN 0 -- No data for last hour,
downtime is 60, uptime is 0
                WHEN lspd.last_status = 'active' THEN 60 -- Last status was active,
uptime is 60, downtime is 0
                ELSE 0 -- Last status was inactive, uptime is 0, downtime is 60
            END AS uptime_last_hour,
            CASE
                WHEN lspd.last_status IS NULL THEN 60
                ELSE 0
            END AS downtime_last_hour
        FROM
            all_store_ids asi
        LEFT JOIN
            last_status_per_day lspd ON asi.store_id = lspd.store_id AND
EXTRACT(DOW FROM CURRENT_DATE)::int = lspd.day -- Adjust for zero-based
indexing of days
    ),
```

- Uptime last day is just the difference between count of inactive status of day and total uptime of that day. Downtime is just the the count of inactive status of that day.

```
uptime_last_day AS (
    SELECT
```

```
    sc.store_id,
    sc.day,
    COALESCE(udp.total_uptime_hours, 0) AS uptime_hours,
    CASE
    WHEN sc.total_entries = 0 THEN
      0 -- Assuming 24 hours in a day since there are no entries
    WHEN sc.cnt_inactive = sc.total_entries THEN
      24 -- All entries are inactive, downtime is 24 hours
    ELSE
      24 - COALESCE(udp.total_uptime_hours, 0) -- Subtract uptime hours from
24 hours
    END AS downtime_last_day_hours
  FROM
    status_counts sc
  LEFT JOIN
    total_uptime_per_day udp ON sc.store_id = udp.store_id AND sc.day =
udp.day
  ),
```

- Now finally we calculate the uptime last week. Its just the sum of all the uptime for all days and Sum of all the downtime for each day is downtime of the week.

```
total_uptime_per_week AS (
  SELECT
    store_id,
    SUM(total_uptime_hours) AS total_uptime_week
  FROM
    total_uptime_per_day
  GROUP BY
    store_id
  ),
  uptime_downtime_per_week AS (
  SELECT
    asi.store_id,
    COALESCE(SUM(CASE WHEN vt.status = 'active' THEN
COALESCE(udp.total_uptime_hours, 0) ELSE (total_uptime_week -
COALESCE(udp.total_uptime_hours, 0)) END), 0) AS total_uptime_week,
    COALESCE(SUM(CASE WHEN vt.status = 'inactive' THEN
COALESCE(udp.total_uptime_hours, 0) ELSE 0 END), 0) AS total_downtime_week
  FROM
    all_store_ids asi
  LEFT JOIN
    total_uptime_per_week tupw ON asi.store_id = tupw.store_id
  LEFT JOIN
    valid_timestamps vt ON asi.store_id = vt.store_id
```

```
LEFT JOIN
    total_uptime_per_day udp ON asi.store_id = udp.store_id
GROUP BY
    asi.store_id, total_uptime_week
)
```