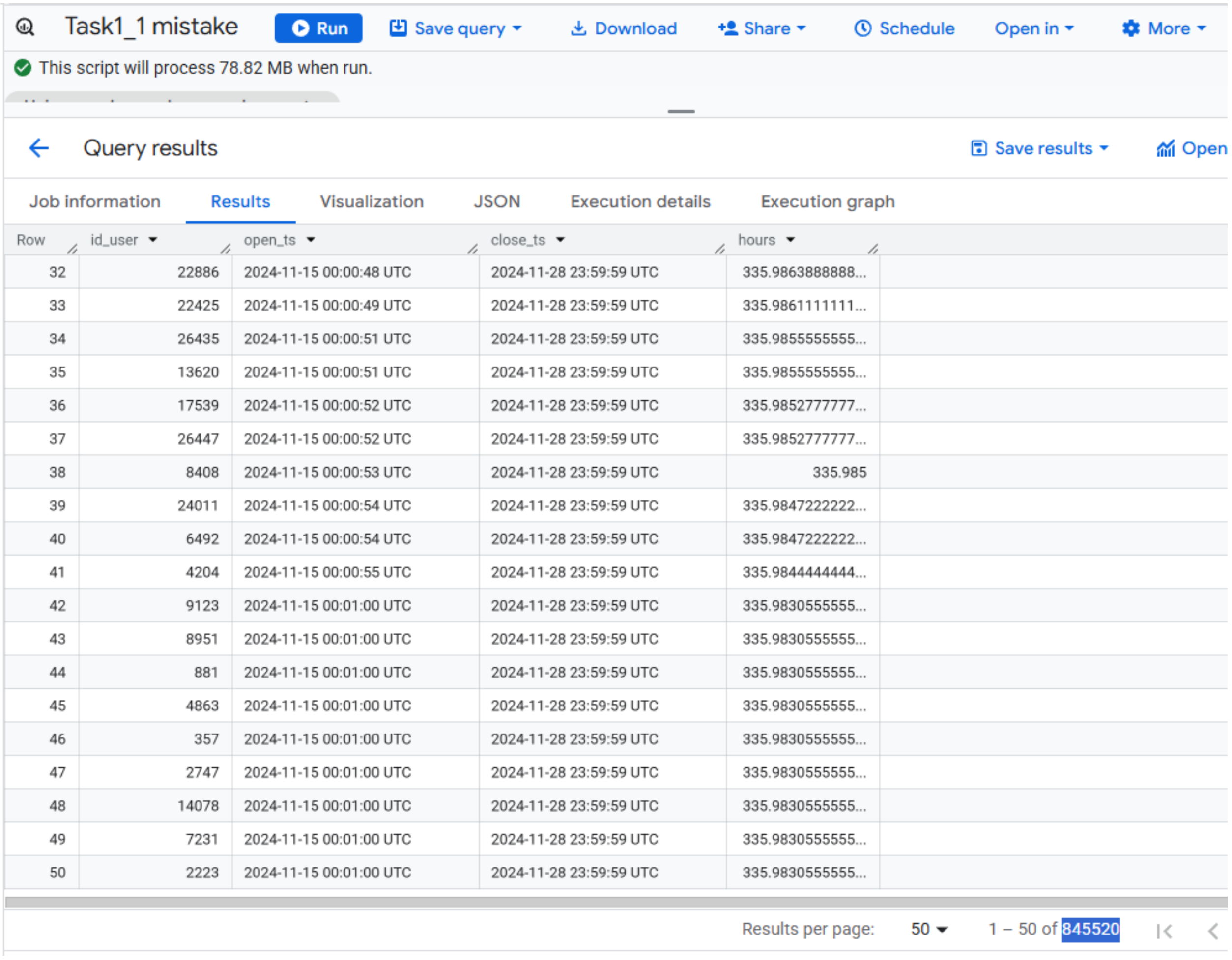
**At first, the task seemed quite straightforward — to calculate the number of hours spent, which is exactly what I initially did:**

```
-- select required columns and assign a table name (base)
WITH base AS (
    SELECT
        id_user,
        action,
        TIMESTAMP(timestamp_action) AS ts
    FROM `analytics.user_sessions_events`
),

-- current moment = maximum timestamp in the data
max_ts AS (
    SELECT MAX(ts) AS max_time FROM base
),

-- create open → close pairs
paired AS (
    SELECT
        b.id_user,
        b.ts AS open_time,
        LEAD(b.ts) OVER (PARTITION BY b.id_user ORDER BY b.ts) AS next_time,
        LEAD(b.action) OVER (PARTITION BY b.id_user ORDER BY b.ts) AS next_action
    FROM base b
    WHERE b.action = 'open'
),

-- if next_action != 'close' — close the session at max_ts
sessions AS (
    SELECT
        p.id_user,
        p.open_time,
        CASE
            WHEN p.next_action = 'close' THEN p.next_time
            ELSE m.max_time
        END AS close_time
    FROM paired p
    CROSS JOIN max_ts m
),

-- generate all days in the range.
-- generation is used instead of existing dates because there may be days
-- when users did not visit the site at all.
-- The task explicitly requires the last 10 days.
days AS (
    SELECT
        day
    FROM (
        SELECT
            DATE_SUB((SELECT DATE(max_time) FROM max_ts), INTERVAL offset DAY) AS day
        FROM UNNEST(GENERATE_ARRAY(0, 9)) AS offset
    )
),

-- split each session by days
session_days AS (
    SELECT
        s.id_user,
        d.day,
        -- start within the day
        GREATEST(s.open_time, TIMESTAMP(d.day)) AS start_ts,
        -- end within the day
        LEAST(s.close_time, TIMESTAMP(DATE_ADD(d.day, INTERVAL 1 DAY))) AS end_ts
    FROM sessions s
    JOIN days d
        ON DATE(s.open_time) <= d.day
       AND DATE(s.close_time) >= d.day
),

-- calculate duration (hours)
durations AS (
    SELECT
        id_user,
        day,
        TIMESTAMP_DIFF(end_ts, start_ts, SECOND) / 3600.0 AS hours_online
    FROM session_days
    WHERE end_ts > start_ts   -- filter zero intervals
)

-- final result
SELECT
    id_user,
    day,
    SUM(hours_online) AS total_hours_online
FROM durations
GROUP BY id_user, day
ORDER BY id_user, day;

WITH base AS (
  SELECT id_user, action, TIMESTAMP(timestamp_action) AS ts
  FROM `analytics.user_sessions_events`
),

-- batch version of pairing (LEAD on OPEN) — temporarily for validation
paired AS (
  SELECT
    b.id_user,
    b.ts AS open_ts,
    LEAD(b.ts) OVER (PARTITION BY b.id_user ORDER BY b.ts) AS next_ts,
    LEAD(b.action) OVER (PARTITION BY b.id_user ORDER BY b.ts) AS next_action
  FROM base b
  WHERE b.action = 'open'
),

sessions AS (
  SELECT
    id_user,
    open_ts,
    CASE
      WHEN next_action = 'close' THEN next_ts
      ELSE (SELECT MAX(TIMESTAMP(timestamp_action))
            FROM `analytics.user_sessions_events`)
    END AS close_ts
  FROM paired
)

SELECT
  id_user,
  open_ts,
  close_ts,
  TIMESTAMP_DIFF(close_ts, open_ts, SECOND) / 3600.0 AS hours
FROM sessions
WHERE TIMESTAMP_DIFF(close_ts, open_ts, HOUR) > 24
ORDER BY hours DESC;
```

**However, the result was, to put it mildly, far from ideal.**
**Seeing 336 hours per day is clearly impossible, yet there were many such cases in the output.**

**Therefore, I decided to conduct a short EDA to understand why this happened. See the file "EDA".**



This script will process 78.82 MB when run.

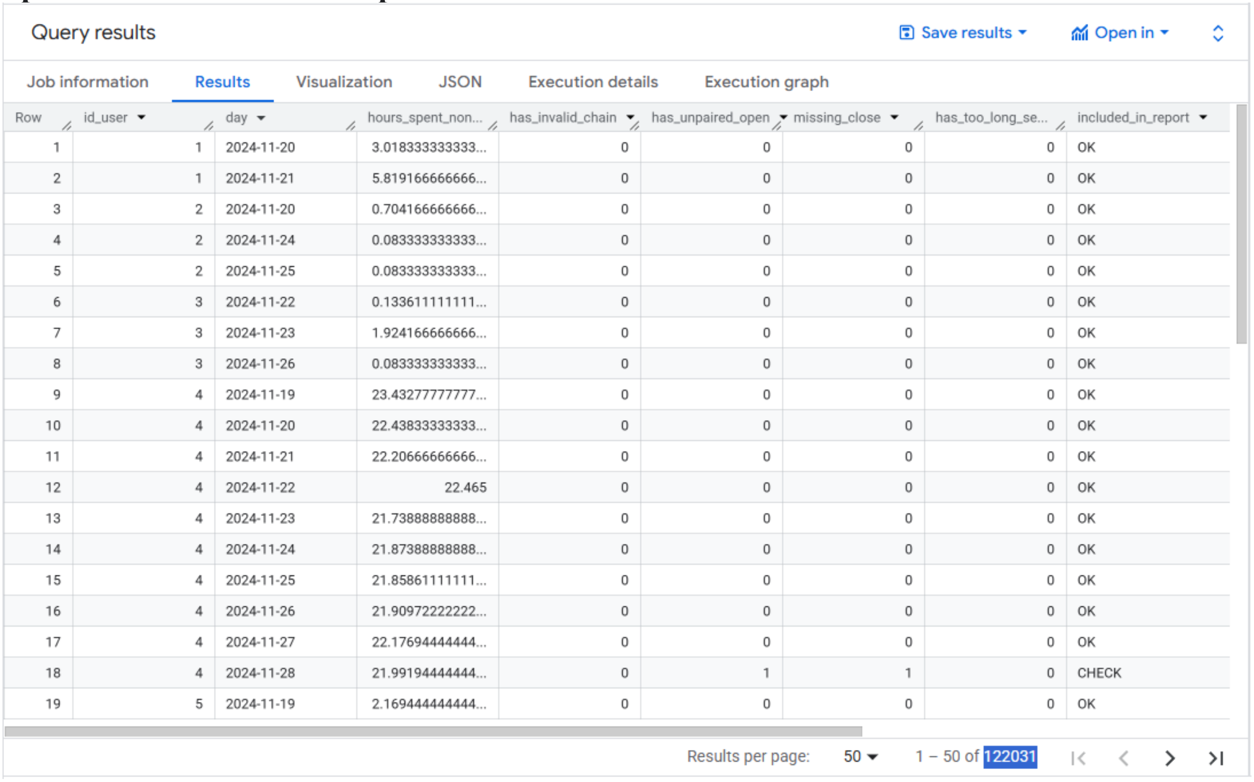
| Row | id_user | open_ts | close_ts | hours |
|---|---|---|---|---|
| 32 | 22886 | 2024-11-15 00:00:48 UTC | 2024-11-28 23:59:59 UTC | 335.9863888888... |
| 33 | 22425 | 2024-11-15 00:00:49 UTC | 2024-11-28 23:59:59 UTC | 335.9861111111... |
| 34 | 26435 | 2024-11-15 00:00:51 UTC | 2024-11-28 23:59:59 UTC | 335.9855555555... |
| 35 | 13620 | 2024-11-15 00:00:51 UTC | 2024-11-28 23:59:59 UTC | 335.9855555555... |
| 36 | 17539 | 2024-11-15 00:00:52 UTC | 2024-11-28 23:59:59 UTC | 335.9852777777... |
| 37 | 26447 | 2024-11-15 00:00:52 UTC | 2024-11-28 23:59:59 UTC | 335.9852777777... |
| 38 | 8408 | 2024-11-15 00:00:53 UTC | 2024-11-28 23:59:59 UTC | 335.985 |
| 39 | 24011 | 2024-11-15 00:00:54 UTC | 2024-11-28 23:59:59 UTC | 335.9847222222... |
| 40 | 6492 | 2024-11-15 00:00:54 UTC | 2024-11-28 23:59:59 UTC | 335.9847222222... |
| 41 | 4204 | 2024-11-15 00:00:55 UTC | 2024-11-28 23:59:59 UTC | 335.9844444444... |
| 42 | 9123 | 2024-11-15 00:01:00 UTC | 2024-11-28 23:59:59 UTC | 335.9830555555... |
| 43 | 8951 | 2024-11-15 00:01:00 UTC | 2024-11-28 23:59:59 UTC | 335.9830555555... |
| 44 | 881 | 2024-11-15 00:01:00 UTC | 2024-11-28 23:59:59 UTC | 335.9830555555... |
| 45 | 4863 | 2024-11-15 00:01:00 UTC | 2024-11-28 23:59:59 UTC | 335.9830555555... |
| 46 | 357 | 2024-11-15 00:01:00 UTC | 2024-11-28 23:59:59 UTC | 335.9830555555... |
| 47 | 2747 | 2024-11-15 00:01:00 UTC | 2024-11-28 23:59:59 UTC | 335.9830555555... |
| 48 | 14078 | 2024-11-15 00:01:00 UTC | 2024-11-28 23:59:59 UTC | 335.9830555555... |
| 49 | 7231 | 2024-11-15 00:01:00 UTC | 2024-11-28 23:59:59 UTC | 335.9830555555... |
| 50 | 2223 | 2024-11-15 00:01:00 UTC | 2024-11-28 23:59:59 UTC | 335.9830555555... |

Results per page: 50   1 – 50 of 845520

**The next step was to flag the data and keep the ability to filter out irrelevant records.**

```sql
-- base table with duplicates removed
WITH cleaned AS (
  SELECT DISTINCT
    id_user,
    action,
    TIMESTAMP(timestamp_action) AS ts
  FROM `analytics.user_sessions_events`
),

-- compute previous event using LAG
with_prev AS (
  SELECT
    id_user,
    action,
    ts,
    LAG(action) OVER (PARTITION BY id_user ORDER BY ts) AS prev_action,
    LAG(ts)     OVER (PARTITION BY id_user ORDER BY ts) AS prev_ts
  FROM cleaned
),

-- mark logically invalid event chains
marked AS (
  SELECT
    id_user,
    action,
    ts AS event_ts,
    prev_action,
    prev_ts,
    CASE
      WHEN prev_action IS NULL AND action = 'close' THEN 1   -- close without prior open
      WHEN prev_action = 'open'  AND action = 'open'  THEN 1 -- two consecutive opens
      WHEN prev_action = 'close' AND action = 'close' THEN 1 -- two consecutive closes
      ELSE 0
    END AS is_invalid_chain
  FROM with_prev
),

-- attach next event using LEAD
paired AS (
  SELECT
    id_user,
    action,
    event_ts,
    is_invalid_chain,
    LEAD(action) OVER (PARTITION BY id_user ORDER BY event_ts) AS next_action,
    LEAD(event_ts) OVER (PARTITION BY id_user ORDER BY event_ts) AS next_ts
  FROM marked
),

-- build all sessions including invalid or missing close events
sessions AS (
  SELECT
    id_user,
    event_ts AS session_start,
    COALESCE(next_ts, (SELECT MAX(ts) FROM cleaned)) AS session_end,
    is_invalid_chain,
    CASE WHEN action = 'open' AND next_action = 'close' THEN 0 ELSE 1 END AS is_unpaired_open,
    CASE WHEN next_ts IS NULL THEN 1 ELSE 0 END AS missing_close,
    CASE
      WHEN next_ts IS NOT NULL
       AND TIMESTAMP_DIFF(next_ts, event_ts, HOUR) > 24
      THEN 1 ELSE 0
    END AS too_long,
    TIMESTAMP_DIFF(
      COALESCE(next_ts, (SELECT MAX(ts) FROM cleaned)),
      event_ts,
      SECOND
    ) AS duration_seconds
  FROM paired
  WHERE action = 'open'
),

-- split sessions across calendar days
expanded AS (
  SELECT
    id_user,
    is_invalid_chain,
    is_unpaired_open,
    missing_close,
    too_long,
    day,
    CASE
      WHEN DATE(session_start) = DATE(session_end)
        THEN TIMESTAMP_DIFF(session_end, session_start, SECOND)
      WHEN day = DATE(session_start)
        THEN TIMESTAMP_DIFF(
          TIMESTAMP_TRUNC(session_start, DAY) + INTERVAL 1 DAY,
          session_start,
          SECOND
        )
      WHEN day = DATE(session_end)
        THEN TIMESTAMP_DIFF(
          session_end,
          TIMESTAMP_TRUNC(session_end, DAY),
          SECOND
        )
      ELSE 24 * 3600
    END AS seconds_in_day
  FROM sessions,
  UNNEST(GENERATE_DATE_ARRAY(DATE(session_start), DATE(session_end))) AS day
),

-- define the last 10-day window
max_dates AS (
  SELECT MAX(DATE(ts)) AS max_date FROM cleaned
),

date_window AS (
  SELECT
    DATE_SUB(max_date, INTERVAL 9 DAY) AS start_day,
    max_date AS end_day
  FROM max_dates
)

-- final aggregated result with data quality flags
SELECT
  e.id_user,
  e.day,
  SUM(e.seconds_in_day) / 3600.0 AS hours_spent_non_rounded,
  MAX(CAST(e.is_invalid_chain AS INT64)) AS has_invalid_chain,
  MAX(CAST(e.is_unpaired_open AS INT64)) AS has_unpaired_open,
  MAX(CAST(e.missing_close AS INT64))    AS missing_close,
  MAX(CAST(e.too_long AS INT64))         AS has_too_long_session,
  CASE
    WHEN MAX(CAST(e.is_invalid_chain AS INT64)) = 0
     AND MAX(CAST(e.is_unpaired_open AS INT64)) = 0
     AND MAX(CAST(e.missing_close AS INT64))    = 0
     AND MAX(CAST(e.too_long AS INT64))         = 0
    THEN 'OK'
    ELSE 'CHECK'
  END AS included_in_report
FROM expanded e
JOIN date_window w
  ON e.day BETWEEN w.start_day AND w.end_day
GROUP BY e.id_user, e.day
ORDER BY e.id_user, e.day;
```

**The result turned out to be much more realistic.**
**The end user can independently filter the data depending on further conditions and goals.**
**Special attention should be paid to the last column marked "CHECK".**



Query results

Save results | Open in

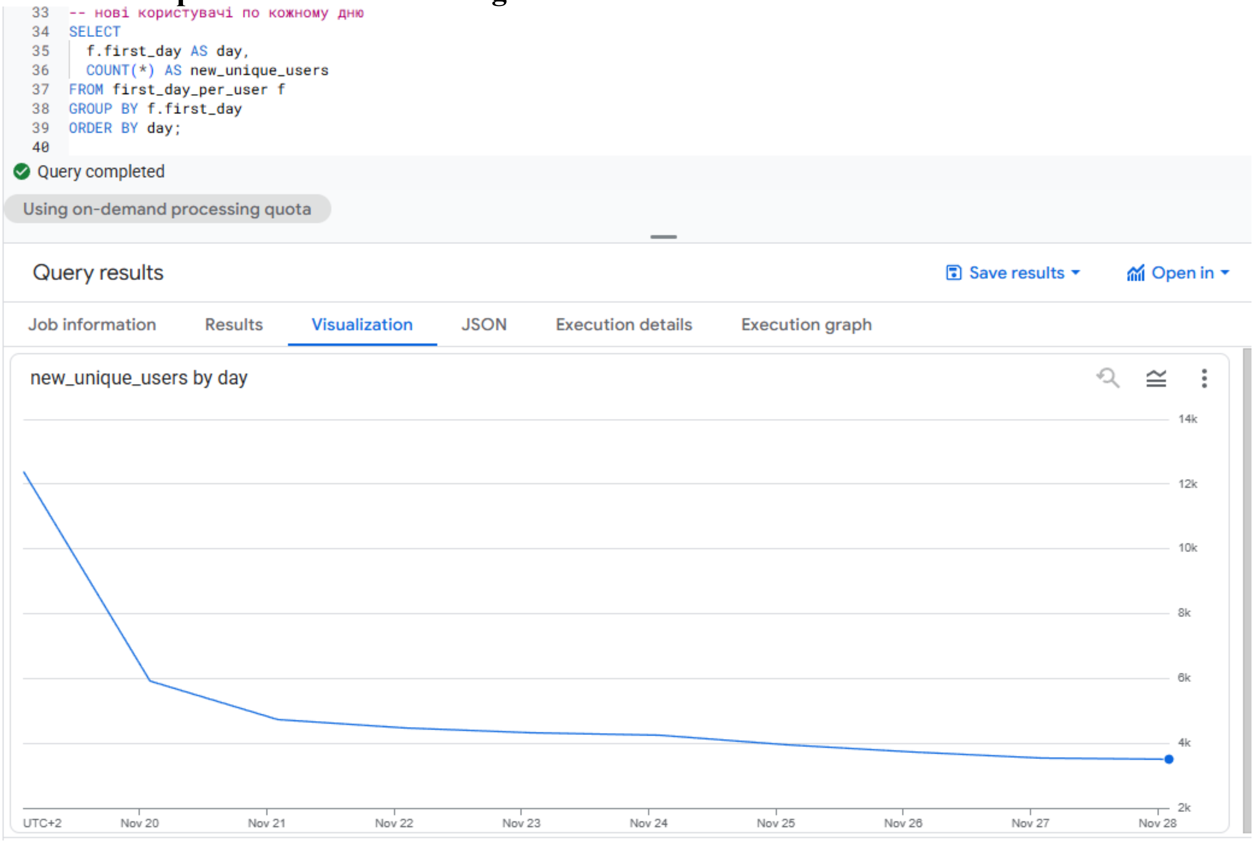
Job information | Results | Visualization | JSON | Execution details | Execution graph

| Row | id_user | day | hours_spent_non... | has_invalid_chain | has_unpaired_open | missing_close | has_too_long_se... | included_in_report |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2024-11-20 | 3.018333333333... | 0 | 0 | 0 | 0 | OK |
| 2 | 1 | 2024-11-21 | 5.819166666666... | 0 | 0 | 0 | 0 | OK |
| 3 | 2 | 2024-11-20 | 0.704166666666... | 0 | 0 | 0 | 0 | OK |
| 4 | 2 | 2024-11-24 | 0.083333333333... | 0 | 0 | 0 | 0 | OK |
| 5 | 2 | 2024-11-25 | 0.083333333333... | 0 | 0 | 0 | 0 | OK |
| 6 | 3 | 2024-11-22 | 0.133611111111... | 0 | 0 | 0 | 0 | OK |
| 7 | 3 | 2024-11-23 | 1.924166666666... | 0 | 0 | 0 | 0 | OK |
| 8 | 3 | 2024-11-26 | 0.083333333333... | 0 | 0 | 0 | 0 | OK |
| 9 | 4 | 2024-11-19 | 23.43277777777... | 0 | 0 | 0 | 0 | OK |
| 10 | 4 | 2024-11-20 | 22.43833333333... | 0 | 0 | 0 | 0 | OK |
| 11 | 4 | 2024-11-21 | 22.20666666666... | 0 | 0 | 0 | 0 | OK |
| 12 | 4 | 2024-11-22 | 22.465 | 0 | 0 | 0 | 0 | OK |
| 13 | 4 | 2024-11-23 | 21.73888888888... | 0 | 0 | 0 | 0 | OK |
| 14 | 4 | 2024-11-24 | 21.87388888888... | 0 | 0 | 0 | 0 | OK |
| 15 | 4 | 2024-11-25 | 21.85861111111... | 0 | 0 | 0 | 0 | OK |
| 16 | 4 | 2024-11-26 | 21.90972222222... | 0 | 0 | 0 | 0 | OK |
| 17 | 4 | 2024-11-27 | 22.17694444444... | 0 | 0 | 0 | 0 | OK |
| 18 | 4 | 2024-11-28 | 21.99194444444... | 0 | 1 | 1 | 0 | CHECK |
| 19 | 5 | 2024-11-19 | 2.169444444444... | 0 | 0 | 0 | 0 | OK |

Results per page: 50    1 – 50 of 122031

**A significant reduction in the number of records (approximately 10x) can be explained by the fact that users appear many times in the original dataset, while in the final output we aggregate data per user per day.**

We can now proceed with visualizing the obtained results.



```
-- нові користувачі по кожному дню
SELECT
  f.first_day AS day,
  COUNT(*) AS new_unique_users
FROM first_day_per_user f
GROUP BY f.first_day
ORDER BY day;

```

Query completed

Using on-demand processing quota