```sql
-- Original: For users who have tweeted about both candidates, which candidate have they tweeted about more, and the total likes and retweets of these tweets
WITH trumpCountTable(user_id, total_trump_tweets) AS (
    SELECT users.user_id, COUNT(tweets.tweet_id) AS total_trump_tweets, SUM(tweets.likes) AS total_likes, SUM(tweets.retweet_count) AS total_retweets
    FROM tweets INNER JOIN users
    ON tweets.user_id = users.user_id
    WHERE tweets.tweet_about = 'Trump'
    GROUP BY users.user_id
),
    bidenCountTable(user_id, total_biden_tweets) AS (
    SELECT users.user_id, COUNT(tweets.tweet_id) AS Total_Biden_Tweets, SUM(tweets.likes) AS total_likes, SUM(tweets.retweet_count) AS total_retweets
    FROM tweets INNER JOIN users
    ON tweets.user_id = users.user_id
    WHERE tweets.tweet_about = 'Biden'
    GROUP BY users.user_id
)
SELECT users.user_id, users.user_name,
       CASE WHEN trumpCountTable.total_trump_tweets > bidenCountTable.total_biden_tweets THEN 'Trump'
            WHEN trumpCountTable.total_trump_tweets < bidenCountTable.total_biden_tweets THEN 'Biden'
            ELSE 'Both'
       END AS most_tweeted_about,
       CASE WHEN trumpCountTable.total_trump_tweets > bidenCountTable.total_biden_tweets THEN trumpCountTable.total_likes
            WHEN trumpCountTable.total_trump_tweets < bidenCountTable.total_biden_tweets THEN bidenCountTable.total_likes
            ELSE trumpCountTable.total_likes + bidenCountTable.total_likes
```

```
24          END AS total_likes,
25          CASE WHEN trumpCountTable.total_trump_tweets
     > bidenCountTable.total_biden_tweets THEN
    trumpCountTable.total_retweets
26              WHEN trumpCountTable.total_trump_tweets
     < bidenCountTable.total_biden_tweets THEN
    bidenCountTable.total_retweets
27              ELSE trumpCountTable.total_retweets +
    bidenCountTable.total_retweets
28          END AS total_retweets,
29      users.user_followers_count, user_join_date
30  FROM users INNER JOIN trumpCountTable
31  ON users.user_id = trumpCountTable.user_id
32  INNER JOIN bidenCountTable
33  ON users.user_id = bidenCountTable.user_id;
34
35  -- Inefficient
36  WITH trumpCountTable(user_id, total_trump_tweets,
    total_likes, total_retweets) AS (
37      SELECT
38          users.user_id,
39          COUNT(tweets.tweet_id) AS
    total_trump_tweets,
40          SUM(tweets.likes) + 0 AS total_likes, --
    Adding 0 for redundancy
41          SUM(tweets.retweet_count + 0) AS
    total_retweets -- Adding redundant calculations
42      FROM tweets
43      INNER JOIN users
44      ON tweets.user_id = users.user_id
45      WHERE tweets.tweet_about = 'Trump'
46          AND tweets.tweet_id IS NOT NULL --
    Pointless filter
47      GROUP BY users.user_id
48      HAVING COUNT(tweets.tweet_id) >= 0 -- Useless
    HAVING clause
49  ),
50  bidenCountTable(user_id, total_biden_tweets,
    total_likes, total_retweets) AS (
51      SELECT
52          users.user_id,
```

```
53          COUNT(tweets.tweet_id) AS
   total_biden_tweets,
54          SUM(tweets.likes) + 0 AS total_likes, --
   Redundant addition
55          SUM(tweets.retweet_count + 0) AS
   total_retweets -- Unnecessary operation
56      FROM tweets
57      INNER JOIN users
58      ON tweets.user_id = users.user_id
59      WHERE tweets.tweet_about = 'Biden'
60          AND tweets.tweet_id IS NOT NULL --
   Pointless condition
61      GROUP BY users.user_id
62      HAVING COUNT(tweets.tweet_id) >= 0 --
   Unnecessary HAVING clause
63 )
64 SELECT
65      users.user_id,
66      users.user_name,
67      CASE
68          WHEN trumpCountTable.total_trump_tweets >
   bidenCountTable.total_biden_tweets THEN 'Trump'
69          WHEN trumpCountTable.total_trump_tweets <
   bidenCountTable.total_biden_tweets THEN 'Biden'
70          ELSE 'Both'
71      END AS most_tweeted_about,
72      CASE
73          WHEN trumpCountTable.total_trump_tweets >
   bidenCountTable.total_biden_tweets THEN
   trumpCountTable.total_likes
74          WHEN trumpCountTable.total_trump_tweets <
   bidenCountTable.total_biden_tweets THEN
   bidenCountTable.total_likes
75          ELSE trumpCountTable.total_likes +
   bidenCountTable.total_likes + 0 -- Extra addition
76      END AS total_likes,
77      CASE
78          WHEN trumpCountTable.total_trump_tweets >
   bidenCountTable.total_biden_tweets THEN
   trumpCountTable.total_retweets
79          WHEN trumpCountTable.total_trump_tweets <
```

```
 79 bidenCountTable.total_biden_tweets THEN
    bidenCountTable.total_retweets
 80         ELSE trumpCountTable.total_retweets +
    bidenCountTable.total_retweets + 0 -- Extra
    addition
 81     END AS total_retweets,
 82     users.user_followers_count,
 83     users.user_join_date,
 84     (SELECT COUNT(*) FROM tweets WHERE tweets.
    user_id = users.user_id) AS extra_subquery_count
    -- Unnecessary correlated subquery
 85 FROM users
 86 LEFT JOIN trumpCountTable -- Using LEFT JOIN
    instead of INNER JOIN
 87 ON users.user_id = trumpCountTable.user_id
 88 RIGHT JOIN bidenCountTable -- Using RIGHT JOIN
    unnecessarily
 89 ON users.user_id = bidenCountTable.user_id
 90 FULL OUTER JOIN tweets t -- Adding an extra,
    unnecessary join to the tweets table
 91 ON t.user_id = users.user_id
 92 WHERE users.user_id IS NOT NULL -- Pointless
    condition
 93 ORDER BY
 94     users.user_id,
 95     trumpCountTable.total_trump_tweets +
    bidenCountTable.total_biden_tweets, -- Unnecessary
     computation in ORDER BY
 96     users.user_followers_count + 0 -- Extra
    operation
 97 LIMIT 100000; -- Arbitrarily high limit
 98
 99
100 -- Original
101 WITH WeeklyEngagement AS (
102    SELECT
103        DATE_TRUNC('week', t.created_at) AS
    tweet_week,
104        t.tweet_about AS candidate,
105        COUNT(t.tweet_id) AS weekly_tweet_count,
106        SUM(t.likes + t.retweet_count) AS
```

```
106 weekly_engagement
107    FROM tweets t
108    GROUP BY DATE_TRUNC('week', t.created_at), t.
    tweet_about
109 ),
110 EventDays AS (
111    SELECT
112        t.created_at::DATE AS event_date,
113        t.tweet_about AS candidate,
114        COUNT(t.tweet_id) AS event_tweet_count,
115        SUM(t.likes + t.retweet_count) AS
    event_engagement
116    FROM tweets t
117    WHERE t.created_at::DATE IN ('2020-10-22', '
    2020-11-03')  -- Example event dates
118    GROUP BY t.created_at::DATE, t.tweet_about
119 )
120 SELECT
121    w.tweet_week,
122    w.candidate,
123    w.weekly_tweet_count,
124    w.weekly_engagement,
125    e.event_date,
126    e.event_tweet_count,
127    e.event_engagement
128 FROM WeeklyEngagement w
129 LEFT JOIN EventDays e ON DATE_TRUNC('week', e.
    event_date) = w.tweet_week AND e.candidate = w.
    candidate
130 ORDER BY w.candidate, w.tweet_week;
131
132 -- Inefficient
133 WITH WeeklyEngagement AS (
134    SELECT
135        DATE_TRUNC('week', t.created_at) AS
    tweet_week,
136        t.tweet_about AS candidate,
137        COUNT(DISTINCT t.tweet_id) AS
    weekly_tweet_count, -- Adding DISTINCT
    unnecessarily
138        SUM(t.likes + t.retweet_count + COALESCE(0
```

```
138 , t.retweet_count)) AS weekly_engagement -- More
    redundant operations
139    FROM tweets t
140    LEFT JOIN tweets t2 ON t.tweet_id = t2.tweet_id
     -- Self join for no reason
141    WHERE t.tweet_id IS NOT NULL
142        AND t.likes >= 0 -- Unnecessary condition
143    GROUP BY DATE_TRUNC('week', t.created_at), t.
    tweet_about, t.likes -- Grouping by unnecessary
    columns
144    HAVING COUNT(t.tweet_id) >= 0 -- Pointless
    HAVING clause
145 ),
146 EventDays AS (
147    SELECT
148        t.created_at::DATE AS event_date,
149        t.tweet_about AS candidate,
150        (SELECT COUNT(*) FROM tweets WHERE
    created_at = t.created_at) AS event_tweet_count,
    -- Nested correlated subquery
151        SUM(t.likes + t.retweet_count + 0) AS
    event_engagement
152    FROM tweets t
153    WHERE t.created_at::DATE IN (
154        SELECT DISTINCT created_at::DATE
155        FROM tweets
156        WHERE created_at::DATE IN ('2020-10-22', '
    2020-11-03')
157    )
158    GROUP BY t.created_at, t.created_at::DATE, t.
    tweet_about, t.likes -- Grouping by unnecessary
    columns
159    HAVING SUM(t.retweet_count) IS NOT NULL --
    Unnecessary HAVING condition
160 )
161 SELECT
162    w.tweet_week,
163    w.candidate,
164    w.weekly_tweet_count,
165    w.weekly_engagement,
166    e.event_date,
```

```
167    e.event_tweet_count,
168    e.event_engagement,
169    COALESCE(w.weekly_engagement, 0) + COALESCE(e.
    event_engagement, 0) AS combined_engagement --
    Redundant calculation
170 FROM WeeklyEngagement w
171 FULL OUTER JOIN EventDays e -- Using FULL OUTER
    JOIN when LEFT JOIN suffices
172    ON DATE_TRUNC('week', e.event_date) = w.
    tweet_week
173    AND e.candidate = w.candidate
174    AND 1=1 -- Adding a redundant condition
175 WHERE
176    1=1 -- Completely unnecessary
177    AND (w.candidate IS NOT NULL OR e.candidate IS
    NOT NULL) -- Adding redundant filters
178 ORDER BY
179    w.candidate,
180    w.tweet_week,
181    w.weekly_tweet_count,
182    COALESCE(e.event_date, '2000-01-01') -- Sorting
     with additional computed column
183 LIMIT 100000; -- Arbitrarily large LIMIT
184
185
186 -- Original
187 WITH UserEngagement AS (
188    SELECT
189        u.user_id,
190        u.user_name,
191        t.tweet_about AS candidate,
192        u.user_followers_count,
193        COUNT(t.tweet_id) AS total_tweets,
194        SUM(t.likes + t.retweet_count) AS
    total_engagement,
195        CASE
196            WHEN u.user_followers_count = 0 THEN 0
197            ELSE CAST(SUM(t.likes + t.retweet_count
    ) AS FLOAT) / u.user_followers_count
198        END AS follower_to_engagement_ratio
199    FROM users u
```

```
200     JOIN tweets t ON u.user_id = t.user_id
201     WHERE t.tweet_about IN ('Biden', 'Trump')
202     GROUP BY u.user_id, u.user_name, t.tweet_about
    , u.user_followers_count
203 )
204 SELECT
205     user_id,
206     user_name,
207     candidate,
208     user_followers_count,
209     total_tweets,
210     total_engagement,
211     follower_to_engagement_ratio
212 FROM UserEngagement
213 ORDER BY candidate, follower_to_engagement_ratio
    DESC
214 LIMIT 10;
215
216 -- Inefficient
217 WITH UserEngagement AS (
218     SELECT
219         u.user_id,
220         u.user_name,
221         t.tweet_about AS candidate,
222         u.user_followers_count,
223         COUNT(t.tweet_id) AS total_tweets,
224         SUM(t.likes + t.retweet_count) AS
    total_engagement,
225         CASE
226             WHEN u.user_followers_count = 0 THEN 0
227             ELSE CAST(SUM(t.likes + t.
    retweet_count) AS FLOAT) / u.user_followers_count
228         END AS follower_to_engagement_ratio
229     FROM users u
230     JOIN tweets t ON u.user_id = t.user_id
231     WHERE t.tweet_about IN ('Biden', 'Trump')
232     GROUP BY u.user_id, u.user_name, t.tweet_about
    , u.user_followers_count
233 ),
234 DuplicatedUserEngagement AS (
235     SELECT * FROM UserEngagement
```

```
236     UNION ALL
237     SELECT * FROM UserEngagement
238     UNION ALL
239     SELECT * FROM UserEngagement
240     UNION ALL
241     SELECT * FROM UserEngagement
242 ),
243 ExtendedUserEngagement AS (
244     SELECT
245         user_id,
246         user_name,
247         candidate,
248         user_followers_count,
249         total_tweets,
250         total_engagement,
251         follower_to_engagement_ratio,
252         ROW_NUMBER() OVER (PARTITION BY user_id
    ORDER BY follower_to_engagement_ratio DESC) AS
    row_num,
253         DENSE_RANK() OVER (PARTITION BY candidate
    ORDER BY total_engagement DESC) AS
    dense_rank_engagement
254     FROM DuplicatedUserEngagement
255 ),
256 EngagementWithDate AS (
257     SELECT
258         user_id,
259         user_name,
260         candidate,
261         user_followers_count,
262         total_tweets,
263         total_engagement,
264         follower_to_engagement_ratio,
265         CURRENT_DATE AS query_date
266     FROM ExtendedUserEngagement
267 ),
268 RedundantAggregations AS (
269     SELECT
270         user_id,
271         user_name,
272         candidate,
```

```
273         user_followers_count,
274         SUM(total_tweets) AS
    redundant_total_tweets,
275         SUM(total_engagement) AS
    redundant_total_engagement,
276         AVG(follower_to_engagement_ratio) AS
    redundant_follower_to_engagement_ratio
277     FROM EngagementWithDate
278     GROUP BY user_id, user_name, candidate,
    user_followers_count
279 ),
280 RepeatedEngagements AS (
281     SELECT * FROM RedundantAggregations
282     UNION ALL
283     SELECT * FROM RedundantAggregations
284     UNION ALL
285     SELECT * FROM RedundantAggregations
286 ),
287 UnnecessaryJoin AS (
288     SELECT
289         re.user_id,
290         re.user_name,
291         re.candidate,
292         re.user_followers_count,
293         re.redundant_total_tweets,
294         re.redundant_total_engagement,
295         re.redundant_follower_to_engagement_ratio,
296         uw.query_date
297     FROM RepeatedEngagements re
298     LEFT JOIN EngagementWithDate uw ON re.user_id
     = uw.user_id
299 ),
300 ExcessiveFilters AS (
301     SELECT *
302     FROM UnnecessaryJoin
303     WHERE user_followers_count > 1000
304     AND candidate = 'Trump'
305     AND user_name LIKE '%a%'
306     AND user_id NOT IN (SELECT user_id FROM
    UnnecessaryJoin WHERE candidate = 'Biden')
307 ),
```

```
308 DoubleSort AS (
309     SELECT * FROM ExcessiveFilters
310     ORDER BY
    redundant_follower_to_engagement_ratio DESC,
    user_id ASC
311 ),
312 FinalSortAndLimit AS (
313     SELECT * FROM DoubleSort
314     ORDER BY candidate DESC,
    redundant_follower_to_engagement_ratio DESC,
    user_id ASC
315     LIMIT 1000
316 )
317 SELECT
318     user_id,
319     user_name,
320     candidate,
321     user_followers_count,
322     redundant_total_tweets AS total_tweets,
323     redundant_total_engagement AS total_engagement
    ,
324     redundant_follower_to_engagement_ratio AS
    follower_to_engagement_ratio
325 FROM FinalSortAndLimit
326 ORDER BY candidate,
    redundant_follower_to_engagement_ratio DESC;
327
328
329 -- Original
330 WITH DailyCandidateVolume AS (
331    SELECT
332        created_at::DATE AS tweet_date,
333        tweet_about AS candidate,
334        COUNT(tweet_id) AS tweet_count,
335        SUM(likes + retweet_count) AS
    total_engagement
336    FROM tweets
337    WHERE tweet_about IN ('Biden', 'Trump')
338    GROUP BY created_at::DATE, tweet_about
339 ),
340 HighVolumeDays AS (
```

```
341    SELECT
342        candidate,
343        tweet_date,
344        tweet_count,
345        total_engagement,
346        RANK() OVER(PARTITION BY candidate ORDER BY
     tweet_count DESC) AS daily_rank
347    FROM DailyCandidateVolume
348 )
349 SELECT
350    tweet_date,
351    candidate,
352    tweet_count,
353    total_engagement
354 FROM HighVolumeDays
355 WHERE daily_rank <= 5
356 ORDER BY candidate, tweet_date;
357
358 -- Inefficient
359 WITH DailyCandidateVolume AS (
360    SELECT
361        created_at::DATE AS tweet_date,
362        tweet_about AS candidate,
363        COUNT(tweet_id) AS tweet_count,
364        SUM(likes + retweet_count) AS
    total_engagement
365    FROM tweets
366    WHERE tweet_about IN ('Biden', 'Trump')
367    GROUP BY created_at::DATE, tweet_about
368 ),
369 DuplicatedVolume AS (
370    SELECT * FROM DailyCandidateVolume
371    UNION ALL
372    SELECT * FROM DailyCandidateVolume
373    UNION ALL
374    SELECT * FROM DailyCandidateVolume
375    UNION ALL
376    SELECT * FROM DailyCandidateVolume
377 ),
378 HighVolumeDays AS (
379    SELECT
```

```
380         candidate,
381         tweet_date,
382         tweet_count,
383         total_engagement,
384         RANK() OVER(PARTITION BY candidate ORDER BY
     tweet_count DESC) AS daily_rank
385     FROM DuplicatedVolume
386 ),
387 FilteredHighVolumeDays AS (
388     SELECT *
389     FROM HighVolumeDays
390     WHERE tweet_count > 0  -- Redundant filter
391 ),
392 RepeatedVolumeFilter AS (
393     SELECT * FROM FilteredHighVolumeDays
394     UNION ALL
395     SELECT * FROM FilteredHighVolumeDays
396 ),
397 UnnecessaryJoin AS (
398     SELECT
399         vhd.candidate,
400         vhd.tweet_date,
401         vhd.tweet_count,
402         vhd.total_engagement,
403         vhd.daily_rank,
404         vhd.tweet_date AS extra_date
405     FROM RepeatedVolumeFilter vhd
406     LEFT JOIN tweets ts ON vhd.candidate = ts.
    tweet_about
407 ),
408 RedundantRanking AS (
409     SELECT
410         candidate,
411         tweet_date,
412         tweet_count,
413         total_engagement,
414         daily_rank,
415         ROW_NUMBER() OVER (PARTITION BY candidate
    ORDER BY total_engagement DESC) AS redundant_rank
416     FROM UnnecessaryJoin
417 ),
```

```
418 ExcessiveFilters AS (
419    SELECT *
420    FROM RedundantRanking
421    WHERE redundant_rank <= 5
422    AND tweet_count > 0  -- Redundant filter again
423    AND candidate IN ('Biden', 'Trump')  --
    Redundant filter
424 ),
425 FinalSort AS (
426    SELECT * FROM ExcessiveFilters
427    ORDER BY candidate, tweet_date DESC
428 ),
429 FinalOutput AS (
430    SELECT
431        tweet_date,
432        candidate,
433        tweet_count,
434        total_engagement
435    FROM FinalSort
436    WHERE daily_rank <= 5  -- Filtering again for
    the same condition
437 )
438 SELECT
439    tweet_date,
440    candidate,
441    tweet_count,
442    total_engagement
443 FROM FinalOutput
444 ORDER BY candidate, tweet_date;
445
446
447
448
449
450
451
452
453
```