with vars as (

SELECT

UNIX\_MILLIS("{QUERY\_DATE}") AS currentTs,

TIMESTAMP("{START\_TIME}") AS startTime,

TIMESTAMP("{END\_TIME}") AS endTime,

{MIN\_SCORE\_THRESHOLD} AS tweetEmbeddingsMinClusterScore,

{HALF\_LIFE} AS halfLife,

TIMESTAMP("{NO\_OLDER\_TWEETS\_THAN\_DATE}") AS noOlderTweetsThanDate

),

-- Get raw fav events

raw\_favs AS (

SELECT event.favorite.user\_id AS userId, event.favorite.tweet\_id AS tweetId, event.favorite.event\_time\_ms AS tsMillis, 1 AS favOrUnfav

FROM `twttr-bql-timeline-prod.timeline\_service\_favorites.timeline\_service\_favorites`, vars

WHERE (DATE(\_PARTITIONTIME) = DATE(vars.startTime) OR DATE(\_PARTITIONTIME) = DATE(vars.endTime)) AND

TIMESTAMP\_MILLIS(event.favorite.event\_time\_ms) >= vars.startTime

AND TIMESTAMP\_MILLIS(event.favorite.event\_time\_ms) <= vars.endTime

AND event.favorite IS NOT NULL

),

-- Get raw unfav events

raw\_unfavs AS (

SELECT event.unfavorite.user\_id AS userId, event.unfavorite.tweet\_id AS tweetId, event.unfavorite.event\_time\_ms AS tsMillis, -1 AS favOrUnfav

FROM `twttr-bql-timeline-prod.timeline\_service\_favorites.timeline\_service\_favorites`, vars

WHERE (DATE(\_PARTITIONTIME) = DATE(vars.startTime) OR DATE(\_PARTITIONTIME) = DATE(vars.endTime)) AND

TIMESTAMP\_MILLIS(event.favorite.event\_time\_ms) >= vars.startTime

AND TIMESTAMP\_MILLIS(event.favorite.event\_time\_ms) <= vars.endTime

AND event.unfavorite IS NOT NULL

),

-- Union fav and unfav events

favs\_unioned AS (

SELECT \* FROM raw\_favs

UNION ALL

SELECT \* FROM raw\_unfavs

),

-- Group by user and tweetId

user\_tweet\_fav\_pairs AS (

SELECT userId, tweetId, ARRAY\_AGG(STRUCT(favOrUnfav, tsMillis) ORDER BY tsMillis DESC LIMIT 1) as details, count(\*) as cnt

FROM favs\_unioned

GROUP BY userId, tweetId

),

-- Remove unfav events

tweet\_raw\_favs\_table AS (

SELECT userId, tweetId, CAST(dt.tsMillis AS FLOAT64) AS tsMillis

FROM user\_tweet\_fav\_pairs CROSS JOIN UNNEST(details) as dt

WHERE cnt < 3 AND dt.favOrUnfav = 1 -- cnt < 3 to remove crazy fav/unfav users

),

-- Get tweetIds that are eligible for tweet embeddings

tweet\_favs\_table AS (

SELECT userId, tweet\_raw\_favs\_table.tweetId, tsMillis

FROM tweet\_raw\_favs\_table, vars

JOIN (

SELECT tweetId, COUNT(DISTINCT(userId)) AS favCount

FROM tweet\_raw\_favs\_table

GROUP BY tweetId

HAVING favCount >= 8 --we only generate tweet embeddings for tweets with >= 8 favs

) eligible\_tweets USING(tweetId)

-- Apply tweet age filter here

WHERE timestamp\_millis((1288834974657 + ((tweet\_raw\_favs\_table.tweetId & 9223372036850581504) >> 22))) >= vars.noOlderTweetsThanDate

),

-- Read consumer embeddings

consumer\_embeddings AS (

{CONSUMER\_EMBEDDINGS\_SQL}

),

-- Update tweet cluster scores based on fav events

tweet\_cluster\_scores AS (

SELECT tweetId,

STRUCT(

clusterId,

CASE vars.halfLife

-- halfLife = -1 means there is no half life/decay and we directly take the sum as the score

WHEN -1 THEN SUM(clusterNormalizedLogFavScore)

ELSE SUM(clusterNormalizedLogFavScore \* POW(0.5, (currentTs - tsMillis) / vars.halfLife))

END AS clusterNormalizedLogFavScore,

COUNT(\*) AS favCount)

AS clusterIdToScores

FROM tweet\_favs\_table, vars

JOIN consumer\_embeddings USING(userId)

GROUP BY tweetId, clusterId, vars.halfLife

),

-- Generate tweet embeddings

tweet\_embeddings\_with\_top\_clusters AS (

SELECT tweetId, ARRAY\_AGG(

clusterIdToScores

ORDER BY clusterIdToScores.clusterNormalizedLogFavScore DESC

LIMIT {TWEET\_EMBEDDING\_LENGTH}

) AS clusterIdToScores

FROM tweet\_cluster\_scores

GROUP BY tweetId

)

-- Return (tweetId, clusterId, tweetScore) pairs where tweetScore > tweetEmbeddingsMinClusterScore

SELECT tweetId,

clusterId,

clusterNormalizedLogFavScore AS tweetScore, clusterIdToScores

FROM tweet\_embeddings\_with\_top\_clusters, UNNEST(clusterIdToScores) AS clusterIdToScores, vars

WHERE clusterIdToScores.clusterNormalizedLogFavScore > vars.tweetEmbeddingsMinClusterScore