WITH vars AS (

SELECT

TIMESTAMP('{START\_TIME}') AS start\_time,

TIMESTAMP('{END\_TIME}') AS end\_time,

UNIX\_MILLIS('{END\_TIME}') AS currentTs,

{HALFLIFE} AS halfLife,

{TWEET\_SAMPLE\_RATE} AS tweet\_sample\_rate,

{ENG\_SAMPLE\_RATE} AS eng\_user\_sample\_rate,

{MIN\_TWEET\_FAVS} AS min\_tweet\_favs,

{MIN\_TWEET\_IMPS} AS min\_tweet\_imps,

{MAX\_USER\_LOG\_N\_IMPS} AS max\_user\_log\_n\_imps,

{MAX\_USER\_LOG\_N\_FAVS} AS max\_user\_log\_n\_favs,

{MAX\_USER\_FTR} AS max\_user\_ftr,

{MAX\_TWEET\_FTR} AS max\_tweet\_ftr,

700 AS MAX\_EXPONENT, -- this is the maximum exponent one can have in bigquery

),

-- step 1: get impressions and favs

impressions AS (

SELECT

userIdentifier.userId AS user\_id,

item.tweetInfo.actionTweetId AS tweet\_id,

item.tweetInfo.actionTweetAuthorInfo.authorId AS author\_id,

TRUE AS impressed,

MIN(eventMetadata.sourceTimestampMs) AS minTsMilli

FROM twttr-bql-unified-prod.unified\_user\_actions.streaming\_unified\_user\_actions, vars

WHERE

actionType = "ClientTweetLingerImpression"

AND DATE(dateHour) BETWEEN DATE(vars.start\_time) AND DATE(vars.end\_time)

AND TIMESTAMP\_MILLIS(eventMetadata.sourceTimestampMs) BETWEEN vars.start\_time AND vars.end\_time

AND MOD(ABS(farm\_fingerprint(item.tweetInfo.actionTweetId || '')), vars.tweet\_sample\_rate) = 0

AND MOD(ABS(farm\_fingerprint(userIdentifier.userId || '')), vars.eng\_user\_sample\_rate) = 0

-- Apply tweet age filter here

AND timestamp\_millis((1288834974657 +

((item.tweetInfo.actionTweetId & 9223372036850581504) >> 22))) >= (vars.start\_time)

GROUP BY 1, 2, 3

),

favs AS (

SELECT

userIdentifier.userId AS user\_id,

item.tweetInfo.actionTweetId AS tweet\_id,

item.tweetInfo.actionTweetAuthorInfo.authorId AS author\_id,

MIN(eventMetadata.sourceTimestampMs) AS minTsMilli,

-- get last action, and make sure that it's a fav

ARRAY\_AGG(actionType ORDER BY eventMetadata.sourceTimestampMs DESC LIMIT 1)[OFFSET(0)] = "ServerTweetFav" AS favorited,

FROM `twttr-bql-unified-prod.unified\_user\_actions\_engagements.streaming\_unified\_user\_actions\_engagements`, vars

WHERE

actionType IN ("ServerTweetFav", "ServerTweetUnfav")

AND DATE(dateHour) BETWEEN DATE(vars.start\_time) AND DATE(vars.end\_time)

AND TIMESTAMP\_MILLIS(eventMetadata.sourceTimestampMs) BETWEEN vars.start\_time AND vars.end\_time

AND MOD(ABS(farm\_fingerprint(item.tweetInfo.actionTweetId || '')), vars.tweet\_sample\_rate) = 0

AND MOD(ABS(farm\_fingerprint(userIdentifier.userId || '')), vars.eng\_user\_sample\_rate) = 0

-- Apply tweet age filter here

AND timestamp\_millis((1288834974657 +

((item.tweetInfo.actionTweetId & 9223372036850581504) >> 22))) >= (vars.start\_time)

GROUP BY 1, 2, 3

HAVING favorited

),

eng\_data AS (

SELECT

user\_id, tweet\_id, author\_id, impressions.minTsMilli, favorited, impressed

FROM impressions

LEFT JOIN favs USING(user\_id, tweet\_id, author\_id)

),

eligible\_tweets AS (

SELECT

tweet\_id,

author\_id,

COUNTIF(favorited) num\_favs,

COUNTIF(impressed) num\_imps,

COUNTIF(favorited) \* 1.0 / COUNTIF(impressed) AS tweet\_ftr,

ANY\_VALUE(vars.min\_tweet\_favs) min\_tweet\_favs,

ANY\_VALUE(vars.min\_tweet\_imps) min\_tweet\_imps,

ANY\_VALUE(vars.max\_tweet\_ftr) max\_tweet\_ftr,

FROM eng\_data, vars

GROUP BY 1, 2

HAVING num\_favs >= min\_tweet\_favs -- this is an aggressive filter to make the workflow efficient

AND num\_imps >= min\_tweet\_imps

AND tweet\_ftr <= max\_tweet\_ftr -- filter to combat spam

),

eligible\_users AS (

SELECT

user\_id,

CAST(LOG10(COUNTIF(impressed) + 1) AS INT64) log\_n\_imps,

CAST(LOG10(COUNTIF(favorited) + 1) AS INT64) log\_n\_favs,

ANY\_VALUE(vars.max\_user\_log\_n\_imps) max\_user\_log\_n\_imps,

ANY\_VALUE(vars.max\_user\_log\_n\_favs) max\_user\_log\_n\_favs,

ANY\_VALUE(vars.max\_user\_ftr) max\_user\_ftr,

COUNTIF(favorited) \* 1.0 / COUNTIF(impressed) user\_ftr

from eng\_data, vars

GROUP BY 1

HAVING

log\_n\_imps < max\_user\_log\_n\_imps

AND log\_n\_favs < max\_user\_log\_n\_favs

AND user\_ftr < max\_user\_ftr

),

eligible\_eng\_data AS (

SELECT

user\_id,

eng\_data.author\_id,

tweet\_id,

minTsMilli,

favorited,

impressed

FROM eng\_data

INNER JOIN eligible\_tweets USING(tweet\_id)

INNER JOIN eligible\_users USING(user\_id)

),

follow\_graph AS (

SELECT userId, neighbor

FROM `twttr-bq-cassowary-prod.user.user\_user\_normalized\_graph` user\_user\_graph, unnest(user\_user\_graph.neighbors) as neighbor

WHERE DATE(\_PARTITIONTIME) =

( -- Get latest partition time

SELECT MAX(DATE(\_PARTITIONTIME)) latest\_partition

FROM `twttr-bq-cassowary-prod.user.user\_user\_normalized\_graph`, vars

WHERE Date(\_PARTITIONTIME) BETWEEN

DATE\_SUB(Date(vars.end\_time),

INTERVAL 14 DAY) AND DATE(vars.end\_time)

)

AND neighbor.isFollowed is True

),

extended\_eligible\_eng\_data AS (

SELECT

user\_id,

tweet\_id,

minTsMilli,

favorited,

impressed,

neighbor.neighborId is NULL as is\_oon\_eng

FROM eligible\_eng\_data left JOIN follow\_graph ON (follow\_graph.userId = eligible\_eng\_data.user\_id AND follow\_graph.neighbor.neighborId = eligible\_eng\_data.author\_id)

),

-- step 2: merge with iikf

iikf AS (

SELECT

userId AS user\_id,

clusterIdToScore.key AS clusterId,

clusterIdToScore.value.favScore AS favScore,

clusterIdToScore.value.favScoreClusterNormalizedOnly AS favScoreClusterNormalizedOnly,

clusterIdToScore.value.favScoreProducerNormalizedOnly AS favScoreProducerNormalizedOnly,

clusterIdToScore.value.logFavScore AS logFavScore,

clusterIdToScore.value.logfavScoreClusterNormalizedOnly AS logfavScoreClusterNormalizedOnly, -- probably no need for cluster normalization anymore

ROW\_NUMBER() OVER (PARTITION BY userId ORDER BY clusterIdToScore.value.logFavScore DESC) AS uii\_cluster\_rank\_logfavscore,

ROW\_NUMBER() OVER (PARTITION BY userId ORDER BY clusterIdToScore.value.logfavScoreClusterNormalizedOnly DESC) AS uii\_cluster\_rank\_logfavscoreclusternormalized,

FROM `twttr-bq-cassowary-prod.user.simclusters\_v2\_user\_to\_interested\_in\_20M\_145K\_2020`, UNNEST(clusterIdToScores) clusterIdToScore, vars

WHERE DATE(\_PARTITIONTIME) =

(-- Get latest partition time

SELECT MAX(DATE(\_PARTITIONTIME)) latest\_partition

FROM `twttr-bq-cassowary-prod.user.simclusters\_v2\_user\_to\_interested\_in\_20M\_145K\_2020`

WHERE Date(\_PARTITIONTIME) BETWEEN

DATE\_SUB(Date(vars.end\_time),

INTERVAL 14 DAY) AND DATE(vars.end\_time)

)

AND MOD(ABS(farm\_fingerprint(userId || '')), vars.eng\_user\_sample\_rate) = 0

AND clusterIdToScore.value.logFavScore != 0

),

eng\_w\_uii AS (

SELECT

T\_IMP\_FAV.user\_id,

T\_IMP\_FAV.tweet\_id,

T\_IMP\_FAV.impressed,

T\_IMP\_FAV.favorited,

T\_IMP\_FAV.minTsMilli,

T\_IMP\_FAV.is\_oon\_eng,

IIKF.clusterId,

IIKF.logFavScore,

IIKF.logfavScoreClusterNormalizedOnly,

IIKF.uii\_cluster\_rank\_logfavscore,

IIKF.uii\_cluster\_rank\_logfavscoreclusternormalized,

FROM extended\_eligible\_eng\_data T\_IMP\_FAV, vars

INNER JOIN iikf

ON T\_IMP\_FAV.user\_id = IIKF.user\_id

WHERE

T\_IMP\_FAV.impressed

),

-- step 3: Calculate tweet embedding

tweet\_cluster\_agg AS (

SELECT

tweet\_id,

clusterId,

SUM(IF(impressed, logFavScore, 0)) denom\_logFavScore,

SUM(IF(favorited, logFavScore, 0)) nom\_logFavScore,

COUNTIF(impressed) n\_imps,

COUNTIF(favorited) n\_favs,

COUNTIF(impressed AND uii\_cluster\_rank\_logfavscore <= 5) n\_imps\_at\_5,

COUNTIF(favorited AND uii\_cluster\_rank\_logfavscore <= 5) n\_favs\_at\_5,

COUNTIF(favorited AND uii\_cluster\_rank\_logfavscore <= 5 AND is\_oon\_eng) n\_oon\_favs\_at\_5,

COUNTIF(impressed AND uii\_cluster\_rank\_logfavscore <= 5 AND is\_oon\_eng) n\_oon\_imps\_at\_5,

SUM(IF(favorited AND uii\_cluster\_rank\_logfavscore <= 5, 1, 0) \* POW(0.5, (currentTs - minTsMilli) / vars.halfLife)) AS decayed\_n\_favs\_at\_5,

SUM(IF(impressed AND uii\_cluster\_rank\_logfavscore <= 5, 1, 0) \* POW(0.5, (currentTs - minTsMilli) / vars.halfLife)) AS decayed\_n\_imps\_at\_5,

SUM(IF(favorited, logfavScoreClusterNormalizedOnly, 0) \* POW(0.5, (currentTs - minTsMilli) / vars.halfLife)) AS dec\_sum\_logfavScoreClusterNormalizedOnly,

MIN(minTsMilli) minTsMilli,

FROM eng\_w\_uii, vars

GROUP BY 1, 2

),

tweet\_cluster\_intermediate AS (

SELECT

tweet\_id,

clusterId,

minTsMilli,

n\_imps,

n\_favs,

n\_favs\_at\_5,

n\_imps\_at\_5,

n\_oon\_favs\_at\_5,

n\_oon\_imps\_at\_5,

decayed\_n\_favs\_at\_5,

decayed\_n\_imps\_at\_5,

denom\_logFavScore,

nom\_logFavScore,

dec\_sum\_logfavScoreClusterNormalizedOnly,

SAFE\_DIVIDE(n\_favs\_at\_5, n\_imps\_at\_5) AS ftr\_at\_5,

SAFE\_DIVIDE(n\_oon\_favs\_at\_5, n\_oon\_imps\_at\_5) AS ftr\_oon\_at\_5,

row\_number() OVER (PARTITION BY tweet\_id ORDER BY nom\_logFavScore DESC) cluster\_nom\_logFavScore\_ranking,

row\_number() OVER (PARTITION BY tweet\_id ORDER BY dec\_sum\_logfavScoreClusterNormalizedOnly DESC) cluster\_decSumLogFavClusterNormalized\_ranking,

FROM tweet\_cluster\_agg

),

tweet\_e AS (

SELECT

tweet\_id,

MIN(minTsMilli) first\_serve\_millis,

DATE(TIMESTAMP\_MILLIS(MIN(minTsMilli))) date\_first\_serve,

ARRAY\_AGG(STRUCT(

clusterId,

-- the division by MAX\_EXPONENT is to avoid overflow operation

ftr\_at\_5 \* (2 / (1+EXP(-1\* (decayed\_n\_favs\_at\_5/1000))) - 1) \* IF(cluster\_decSumLogFavClusterNormalized\_ranking > MAX\_EXPONENT, 0, 1.0/(POW(1.1, cluster\_decSumLogFavClusterNormalized\_ranking-1))) AS ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay\_1\_1

) ORDER BY ftr\_at\_5 \* (2 / (1+EXP(-1\* (decayed\_n\_favs\_at\_5/1000))) - 1) \* IF(cluster\_decSumLogFavClusterNormalized\_ranking > MAX\_EXPONENT, 0, 1.0/(POW(1.1, cluster\_decSumLogFavClusterNormalized\_ranking-1))) DESC LIMIT {TWEET\_EMBEDDING\_LENGTH}) ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay\_1\_1\_embedding,

ARRAY\_AGG(STRUCT(

clusterId,

-- the division by MAX\_EXPONENT is to avoid overflow operation

ftr\_at\_5 \* (2 / (1+EXP(-1\* (decayed\_n\_favs\_at\_5/10000))) - 1) \* IF(cluster\_decSumLogFavClusterNormalized\_ranking > MAX\_EXPONENT, 0, 1.0/(POW(1.1, cluster\_decSumLogFavClusterNormalized\_ranking-1))) AS ftrat5\_decayed\_pop\_bias\_10000\_rank\_decay\_1\_1

) ORDER BY ftr\_at\_5 \* (2 / (1+EXP(-1\* (decayed\_n\_favs\_at\_5/1000))) - 1) \* IF(cluster\_decSumLogFavClusterNormalized\_ranking > MAX\_EXPONENT, 0, 1.0/(POW(1.1, cluster\_decSumLogFavClusterNormalized\_ranking-1))) DESC LIMIT {TWEET\_EMBEDDING\_LENGTH}) ftrat5\_decayed\_pop\_bias\_10000\_rank\_decay\_1\_1\_embedding,

ARRAY\_AGG(STRUCT(

clusterId,

-- the division by MAX\_EXPONENT is to avoid overflow operation

ftr\_oon\_at\_5 \* (2 / (1+EXP(-1\* (decayed\_n\_favs\_at\_5/1000))) - 1) \* IF(cluster\_nom\_logFavScore\_ranking > MAX\_EXPONENT, 0, 1.0/(POW(1.1, cluster\_nom\_logFavScore\_ranking-1))) AS oon\_ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay

) ORDER BY ftr\_oon\_at\_5 \* (2 / (1+EXP(-1\* (decayed\_n\_favs\_at\_5/1000))) - 1) \* IF(cluster\_nom\_logFavScore\_ranking > MAX\_EXPONENT, 0, 1.0/(POW(1.1, cluster\_nom\_logFavScore\_ranking-1))) DESC LIMIT {TWEET\_EMBEDDING\_LENGTH}) oon\_ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay\_embedding,

ARRAY\_AGG(STRUCT(

clusterId,

dec\_sum\_logfavScoreClusterNormalizedOnly

) ORDER BY dec\_sum\_logfavScoreClusterNormalizedOnly DESC LIMIT {TWEET\_EMBEDDING\_LENGTH}) dec\_sum\_logfavScoreClusterNormalizedOnly\_embedding,

FROM tweet\_cluster\_intermediate, vars

GROUP BY 1

),

tweet\_e\_unnest AS (

SELECT

tweet\_id AS tweetId,

clusterToScores.clusterId AS clusterId,

clusterToScores.{SCORE\_KEY} tweetScore

FROM tweet\_e, UNNEST({SCORE\_COLUMN}) clusterToScores

WHERE clusterToScores.{SCORE\_KEY} IS NOT NULL

AND clusterToScores.{SCORE\_KEY} > 0

)

SELECT

tweetId,

clusterId,

tweetScore

FROM tweet\_e\_unnest