WITH

cluster\_top\_tweets AS (

{CLUSTER\_TOP\_TWEETS\_SQL}

),

flatten\_cluster\_top\_tweets AS (

SELECT

clusterId,

tweet.tweetId,

tweet.tweetScore,

FROM cluster\_top\_tweets, UNNEST(topKTweetsForClusterKey) AS tweet

),

--- There might be delay or skip for the fav-based dataset.

--- This query retrieved the dateHour of the latest partition available.

latest\_fav\_cluster\_to\_tweet AS (

SELECT

MAX(dateHour) AS latestTimestamp

FROM

`twttr-bq-cassowary-prod.user.simclusters\_fav\_based\_cluster\_to\_tweet\_index`

WHERE

TIMESTAMP(dateHour) >= TIMESTAMP("{START\_TIME}")

AND TIMESTAMP(dateHour) <= TIMESTAMP("{END\_TIME}")

),

flatten\_fav\_cluster\_top\_tweets AS (

SELECT

clusterId.clusterId AS clusterId,

tweet.key AS tweetId

FROM

`twttr-bq-cassowary-prod.user.simclusters\_fav\_based\_cluster\_to\_tweet\_index`,

UNNEST(topKTweetsWithScores.topTweetsByFavClusterNormalizedScore) AS tweet,

latest\_fav\_cluster\_to\_tweet

WHERE

dateHour=latest\_fav\_cluster\_to\_tweet.latestTimestamp

),

flatten\_cluster\_top\_tweets\_intersection AS (

SELECT

clusterId,

flatten\_cluster\_top\_tweets.tweetId,

flatten\_cluster\_top\_tweets.tweetScore

FROM

flatten\_cluster\_top\_tweets

INNER JOIN

flatten\_fav\_cluster\_top\_tweets

USING(clusterId, tweetId)

),

processed\_cluster\_top\_tweets AS (

SELECT

clusterId,

ARRAY\_AGG(STRUCT(tweetId, tweetScore) ORDER BY tweetScore LIMIT {CLUSTER\_TOP\_K\_TWEETS}) AS topKTweetsForClusterKey

FROM flatten\_cluster\_top\_tweets\_intersection

GROUP BY clusterId

)

SELECT \*

FROM processed\_cluster\_top\_tweets