Assumption:

1. geographical\_location\_oid is the joining key to the reference table.
2. The expected output schema, geographical\_location is bigint, may have referenced to geographical\_location\_oid.

A white rectangular box with black text

AI-generated content may be incorrect.

1. For the understanding of top\_x, it is top\_x by “geographical\_location”. If the requirement is as simple as just top 10, then limit(x) could be implemented on the filtered\_df.

Comments on scoring criteria:

1. Repo link : <https://github.com/RepoSheep/htx_q>
2. Scala-style report : [htx\_q/htx-de/htx-scala-style-report.xml](https://github.com/RepoSheep/htx_q/blob/main/htx-de/htx-scala-style-report.xml)
3. Run the following in terminal or cmd with scala installed:

*sbt run "/Users/alexchen/Documents/repo/htx/htx\_q/data/trans.parquet" "/Users/alexchen/Documents/repo/htx/htx\_q/data/ref\_table.parquet" "rank\_df\_alex" "2*"

sbt run “<trans\_dataset\_path>” whitespace “<reference\_dataset\_path>” whitespace “<output\_result\_path>” whitespace “<top\_x>”

output can be found [parquet](https://github.com/RepoSheep/htx_q/tree/main/htx-de/item_rank_df) , [csv](https://github.com/RepoSheep/htx_q/tree/main/htx-de/item_rank_df_csv)

1. Primarily work with Pyspark and SQL, spent too much time trying to setup a scala runnable environment. May require more time to learn about scala programming with RDD implementation.
2. –
3. 0
4. 1. Primary time for join will be O(N +M) given 2 dataframe  
   2.group by will be O(N) with shuffle

3. window (ranking) will be O(NlogK) since every geographical\_location\_oid needs to be scanned then each item in the location

4. filter for top\_x is 0(1)

For left outer join without using join/merge function:

Make the reference dataset as a dictionary of key value pair using *right = dict(ref.collect()*), then iterate through the trans df using a custom function to retrieve the geo\_location\_oid value from the dictionary by using  *right.get(key)*…

Highly inefficient and limited by the ref dataset size.

1. –
2. Have to perform EDA to confirm and understand which geo location is skewed. Then implement adding of salt in the column value for both datasets example 5 different salt permutations, then the ref table should also have 5 different records. Perform left join against the salted ref table.

1. pyspark solution: <https://github.com/RepoSheep/htx_q/blob/main/HTX%20Pyspark%20Solution.py>

The databricks platform used automatically uses broadcast. If broadcasting is explicitly required:  
joined\_df = trans.alias("trans").join(**broadcast**(ref\_table).alias("ref\_table"),col("trans.geographical\_location\_oid") == col("ref\_table.geographical\_location\_oid"),"left")

2. Data Architecture Design

Considering that streaming table is required, using delta table, specifically delta live table by databricks will serve the purpose.

We can use readStream function to read the parquet file, withWatermark method to specify the window of monitoring based on timestamp\_detected column in the trans table. Along with dropDuplicationWithinWatermark method on detection\_oid column.

Alternatively, write a writeback function against the table uising df.dropDuplicate(col(“detection\_oid”)) and runs on microbatch or batch schedule.

Recommendation: given the scenario, recommend to use **watermark** to account of late or errors upstream ingestion for more control implementation.

Watermark feature is native to spark streaming engine thus not only specific to snowflake or databricks. However, would recommend this setup in the Azure Environment.