There are 5 join strategies in Spark

1. Broadcast Join
2. Shuffle Join
3. Sort Merge Join
4. Cartesian Join
5. Broadcast Nested Loop Join

Broadcast join will copy dataset B, which is the smaller dataset, to do a local join. This can be used when there is a smaller table involved and can be kept in memory. However, in this scenario, I am assuming dataset B is not small and must be constantly updated. In this case, I will not choose a broadcast join as there could be a case where dataset B can no longer be kept in memory.

Shuffle join would work best if the data is uniform. This means that there must be an equal distribution of 'geographical_location_oid' in dataset A that match to dataset B. However, this is an assumption on the data and I would require more information on this before making this join.

Sort merge join can be used because both datasets can be sorted by the join key, 'geographical_location_oid'. The keys will be matched in order of the sort which makes it scalable for larger datasets as well. And I do not need to worry about data shape as this is not affected by skew data.

Cartesian joins works by finding every combination of join per row between 2 tables. This was not considered due to scalability with a large dataset such as dataset A as this will result in high compute.

Broadcast nested loop join will loops through all rows in dataset B which will be broadcasted for every row found in dataset A. However, this is not scalable as it will take a lot of compute time and cost when dataset A becomes too large which is expected due to the data source and data context.