Spark join strategies

Broadcast join

One of the datasets (usually the smaller dataset) will be broadcast to all the worker nodes where each worker node has a copy of the smaller dataset, allowing fast look up. The larger dataset is scanned, and matches are found using the broadcasted data. This is most effective when one of the datasets is small, typically a few MBs to a few GBs.

Pros:

- Eliminates expensive data shuffling across partitions.
- Faster execution for small reference datasets.

Cons:

- Limited by the memory available on each executor.
- Performance degrades if the smaller dataset is too large to fit in memory.

Sort-Merge join

Sort-Merge Join requires both datasets to be sorted on the join key before merging them efficiently. The sorting operation ensures that matches can be found without excessive reprocessing. This strategy is best suited for large datasets that do not fit entirely in memory.

Pros:

- Efficient for very large datasets where broadcasting is not feasible.
- Avoids excessive memory consumption.

Cons:

- Sorting both datasets can be expensive.
- Requires significant shuffle operations, increasing execution time.

Shuffle hash join

Shuffle Hash Join distributes both datasets across nodes based on the join key and builds a hash table on each partition to find matching rows. This method is useful when neither dataset fits in memory, but both are small enough to perform hashing efficiently.

Pros:

- Can handle medium-sized datasets more efficiently than Sort-Merge Join.

Cons:

- Requires shuffling, which can slow down performance.
- If data is skewed, one partition might receive significantly more data than others.

Cross join

A Cartesian join computes the Cartesian product of two datasets, meaning each row from one dataset is paired with every row from the other. The total number of output rows is the product of the row counts of both datasets (m * n).

Pros:

- Useful for specific cases like generating test datasets or exploratory data analysis.

Cons:

- Extremely expensive in terms of computation and memory.
- Lead to performance issues in large datasets

Join strategy for this project: Broadcast Join

Since the provided datasets are not very large, and Dataset B contains approximately 10,000 rows, it can easily fit into memory and be broadcasted. This eliminates the need for shuffling, thereby reducing execution time and improving efficiency.