3 common Spark join strategies are Broadcast Hash Join, Shuffle Hash Join and Sort-Merge Join.

<u>Broadcast Hash Join (BHJ)</u>

Spark broadcasts (copies) a DataFrame containing a smaller dataset to all its worker nodes. Within each worker node, Spark joins the broadcasted DataFrame to the partitions of its resident DataFrame, that typical contains a significantly larger dataset.

This method requires minimal shuffling, which makes it highly efficient when the smaller DataFrame can fit in to the work nodes memory.

<u>Shuffle Hash Join (SHJ)</u>

2 DataFrames are shuffled on the join keys to align keys. Once the keys are aligned, a hash table is created to map the joins of the 2 DataFrames, and is stored in each partition of the smaller dataset. This hash table is then use to probe matches from the larger dataset.

This method is recommended when the 2 datasets are medium to large sized. Limitations for this method include:

- The 2 datasets must vary in size, i.e. one set must be smaller than the other
- Sufficient resources must be allocated, e.g. sufficient network bandwidth and memory to handle heavy shuffling

<u>Sort Merge Join (SMJ)</u>

Spark shuffles 2 DataFrames by join key and sorts the partition of both DataFrames. The sorted partitions are then merged to find matches.

This method is recommended for large DataFrames as it is more reliable and can easily scale. However, it consumes significant CPU and I/Ops due to sorting.

Based on the size difference between Dataset A and B (1M vs 100K), I have implemented BHJ strategy as is the most optimal strategy.