In this lab, we are experimenting with two config parameters to benchmark the performance of running queries:

1. maxPartitionBytes: Max number of bytes each parallel task can process at a given time
2. shufflePartitions: Number of partitions created after shuffling

Small cluster: 2g per core

Parameters:

1. MaxPartitionBytes: 48 MB

- Partitions created: 1,434

Predicate pushdown:

- Runtime: 17 sec

- Input rows: 2.05 billion (61.6 GB)

After predicate pushdown:

- Output rows: 681 MM

- Predicate pushdown with order date timestamp and writing to disk took 5 min

– Select Orderdate column and distinct

Summary Metrics for 98 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	1 s	2 s	2 s	3 s	3 s
GC Time	12.0 ms	36.0 ms	42.0 ms	49.0 ms	0.3 s
Input Size / Records	20.7 MiB / 16000000	24.5 MiB / 20625000	25.8 MiB / 21000000	26.1 MiB / 21250000	26.7 MiB / 23000000
Shuffle Write Size / Records	2.6 KiB / 121	2.6 KiB / 121	2.7 KiB / 121	2.7 KiB / 121	2.7 KiB / 121

- Query execution plan showed a shuffle and hash aggregate.

- AQE was applied, and the shuffle partitions were coalesced from 20 partitions to 1 to improve performance

- OrderDateTimestamp with and without CAST
- Predicate pushdown without CAST runs in 2 sec
- Filter with CAST processes all partitions, in our case 1,434 (maxPartitionBytes = 48 MB)
- Executors had to process each partition to convert date from timestamp to date and then apply filter

Summary Metrics for 1434 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	16.0 ms	0.6 s	1 s	1 s	4 s
GC Time	0.0 ms	18.0 ms	45.0 ms	68.0 ms	0.4 s
Input Size / Records	256 KiB / 0	16.4 MiB / 528856	46.2 MiB / 1500000	57.6 MiB / 1875000	118.1 MiB / 3850240

Explode is not a wide transformation as DAG visualization does not have an exchange stage.

Summary Metrics for 269 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	0.9 s	4 s	5 s	5 s	9 s
GC Time	33.0 ms	0.2 s	0.2 s	0.3 s	2 s
Input Size / Records	15 MiB / 875000	127 MiB / 7500000	129.7 MiB / 7625000	140.9 MiB / 8250000	153.2 MiB / 9000000

Observations:

1. In the small cluster with 2 workers 8gb and 4 cores each, if the maxPartitionBytes parameter is decreased, it increases the number of partitions each core has to process. For example, I tested with 2 values of maxPartitionBytes
    a. 48 MB, which created 1K+ memory partition,s which slows the performance of compute-intensive tasks like scanning through all partitions or when query pruning does not take effect when transformations like CAST are used on columns.
    b. 100 MB which created 881 partitions still operations like scanning through all files when query pruning was not applied, was slow but slightly better than 48 MB maxPartitionBytes

Summary

1. maxPartitionBytes dramatically improves the performance of file scans when file size is not too small example 48 MB which overwhelms the workers with I/O costs for reading and maintaining metadata and when the file size is not too large.
2. There is no one-size-fits-all recommendation and it is always better to test a few scenarios before finalizing the maxPartitionBytes parameter.

Note: I couldn't test with medium cluster as I had a problem with my Azure subscription as I created multiple subscriptions and after waiting for a few hours when I was finally able to create a new subscription, it didn't allow me to use that subscription to set up the Databricks workspace. Apologies for the last minute notice and I will work on getting it foxed before the next assignment.