Optimizing PySpark Joins: How I Reduced Execution Time from 11s to 1.9s Using AQE & Broadcast Join!

📊 Step 1: Understanding the Dataset & Skewness

📌 Dataset Overview

I worked with two datasets:

1️⃣ Orders Dataset (~1M+ rows)

- Contains **highly skewed** `customer_id` values (some IDs have **500K+ records**).
- Columns: `customer_id`, `order_id`, `amount`.

2️⃣ Customers Dataset (~10K rows)

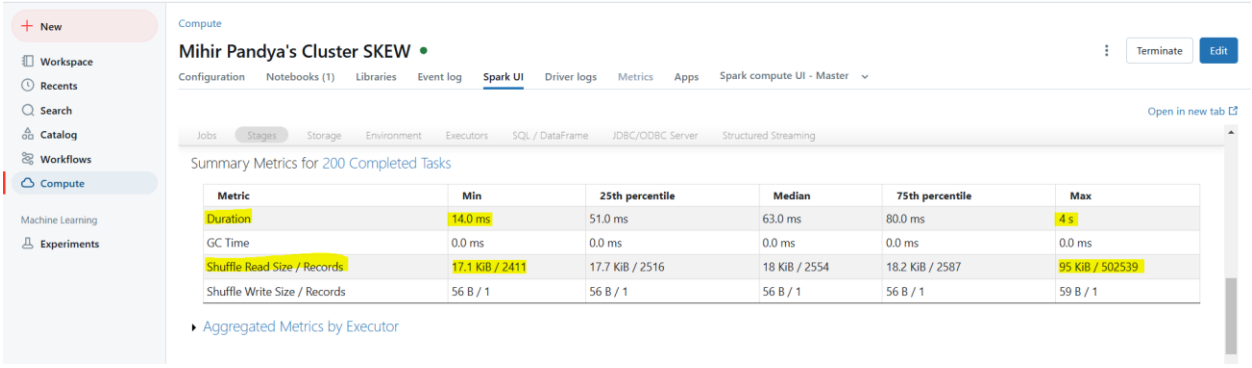
- Smaller in size, **ideal for broadcasting**.
- Columns: `customer_id`, `customer_name`.

📌 Identifying Data Skew

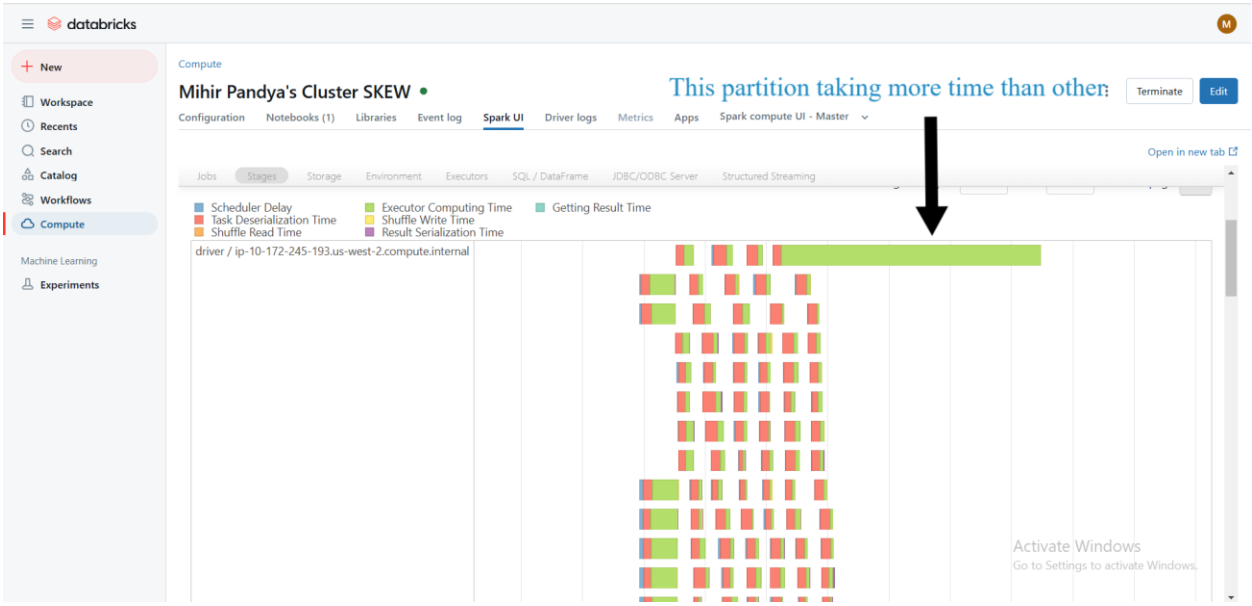
To check for skewness, I ran the following query:

🔍 Observations from Spark UI

- Duration of some task was 14ms and while for some was 4s. This shows the skewness is present.
- Most of the task had shuffle read size/Records - 17.1Kib/2411 while a few were having 95KiB/502539. This also shows skewness.



- **Event Timeline Analysis**: The highlighted partition is taking more time than others



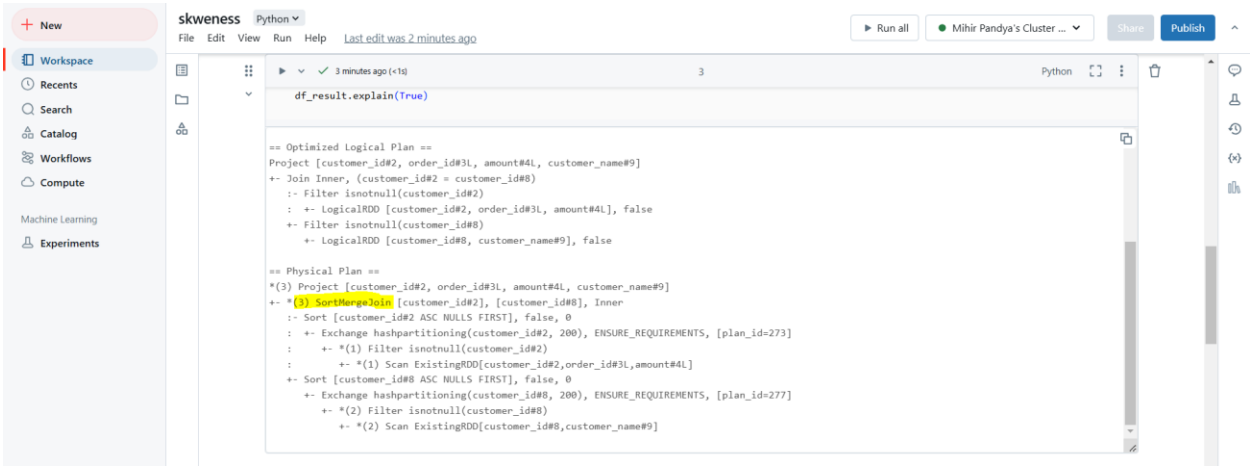
🔬 Step 2: Deep Dive into Optimizations

1️⃣ Without AQE & No Optimizations (11.26s) – Slowest Approach

- **Join Type:** Sort-Merge Join
- **Issues Identified:**
  - Expensive **shuffle operations** due to hash partitioning.

- Uneven task distribution due to **skewed customer_id**.
- **Some tasks took 4s**, while others took **0.14s**

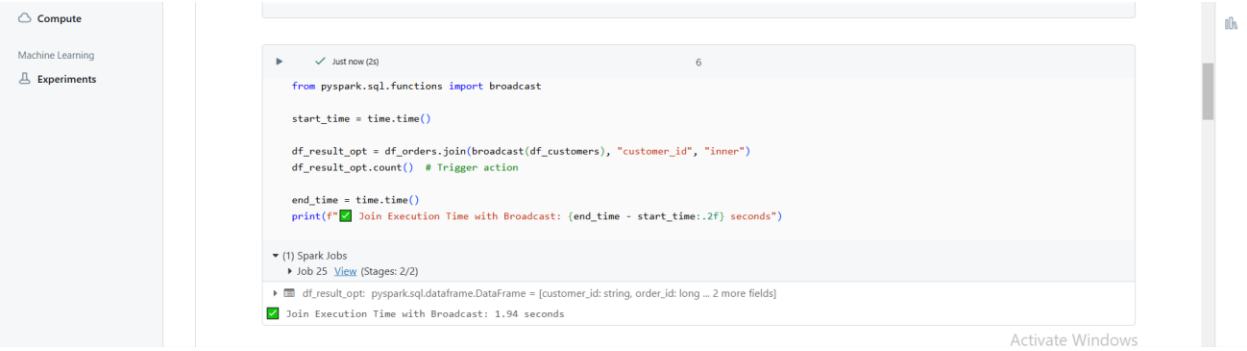
📌 Logical & Physical Plan:



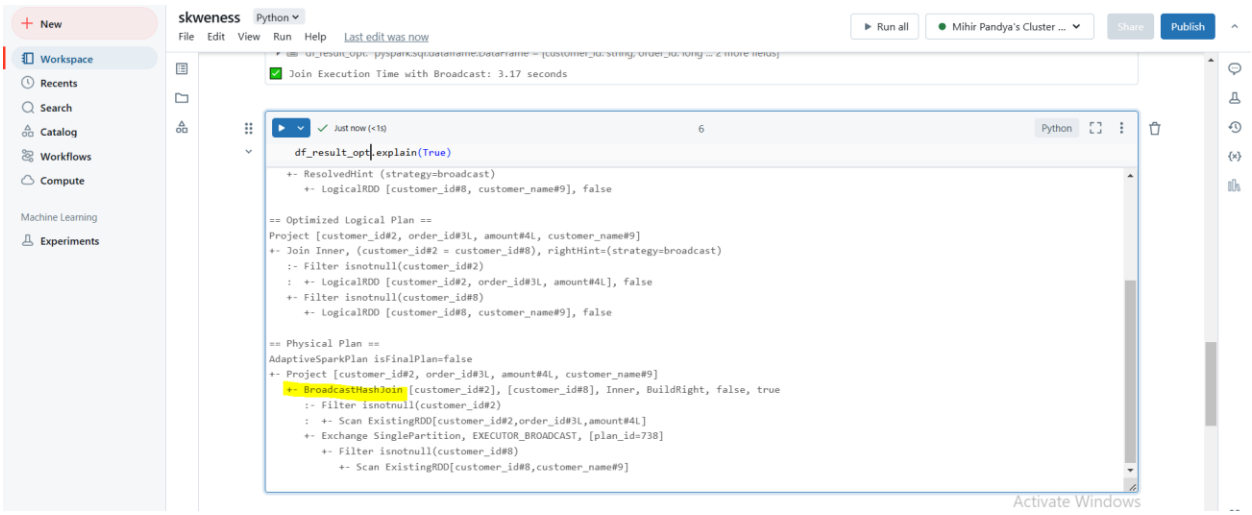
2️⃣ Without AQE + Broadcast Join (1.94s) – Best Performance

- **Join Type: Broadcast Hash Join**
- **Why It's Faster:**
  - **Avoids shuffle** by broadcasting `df_customers` to all nodes.
  - **Reduces data movement**, improving execution time by **5x**.
  - Spark automatically chooses **Broadcast Hash Join** when explicitly applied.

📌 Code Implementation:



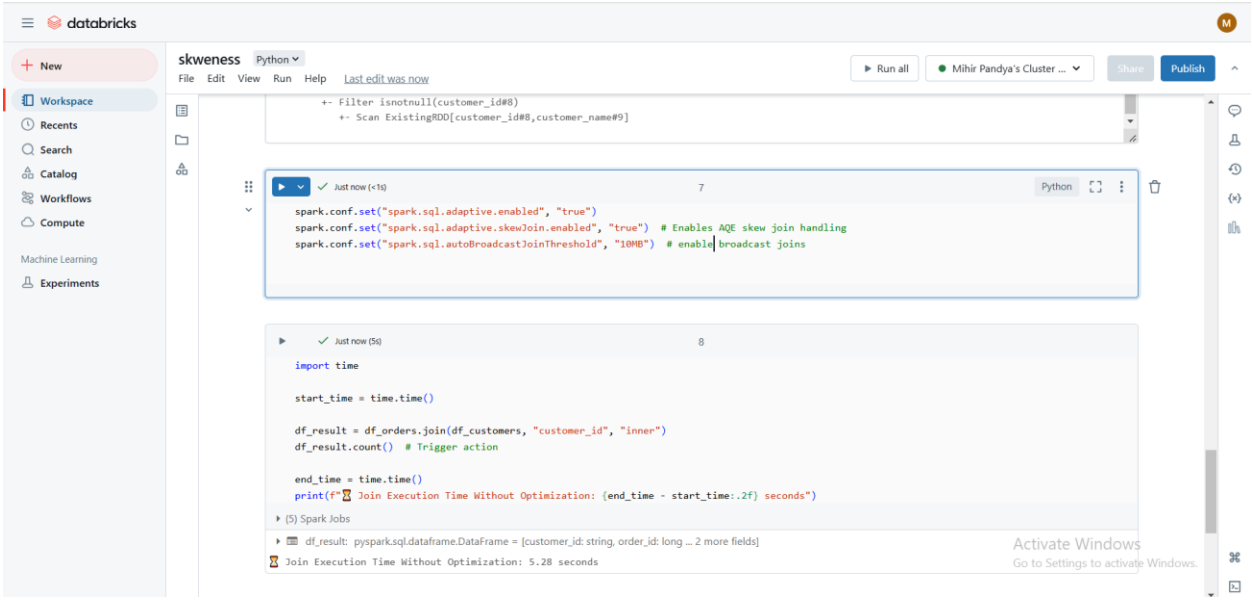
📌 Logical & Physical Plan:



3️⃣ With AQE Enabled (5.28s) – Improved, but Not the Best

- **Join Type:** Sort-Merge Join (default)
- **AQE dynamically optimized shuffle partitions**, but **did not choose Broadcast Join automatically**.
- Execution time **improved from 11.26s → 5.28s**, but still **slower than explicit Broadcast Join**.

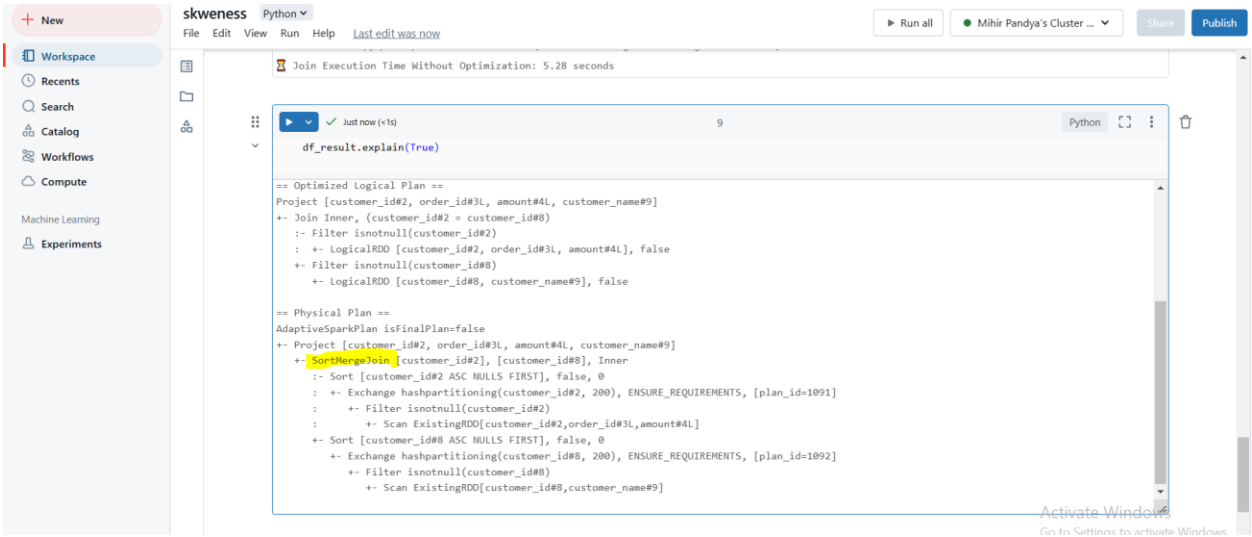
📌 Code Implementation:



📌 Logical & Physical Plan:

- **AQE dynamically adjusts shuffle partitions.**

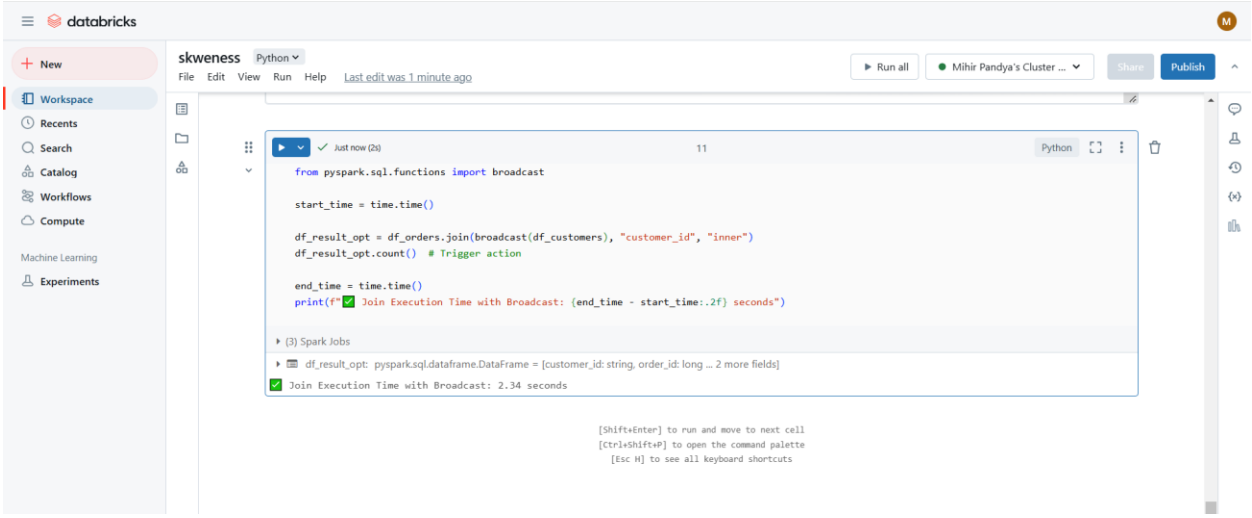
- **Still uses Sort-Merge Join** (not the most optimal).



🔥 With AQE + Manual Broadcast Join (2.34s) – Balanced Approach

- **Join Type: Broadcast Hash Join**
- **Combines the benefits of AQE + Broadcast Join:**
  - ✅ **AQE optimizes shuffle partitions.**
  - ✅ **Manual Broadcast Join ensures efficient lookup.**
- **Execution time: Improved to 2.34s** (near the best case).

📌 Code Implementation:



🔑 Key Takeaways & Recommendations

✅ **Broadcast Join is a game-changer** when joining a **small** table (~10K rows) with a **large** one (~1M+ rows).
✅ **AQE helps optimize shuffle partitions,** but doesn't always **automatically pick the best join strategy**.
✅ **Skewed joins can create performance bottlenecks.** Identifying skewness using **groupBy analysis & Spark UI event timeline** is crucial.
✅ **For best performance:**

- **If a small dataset is involved, manually apply Broadcast Join.**
- **For highly skewed datasets, AQE + Broadcast Join gives the best balance.**