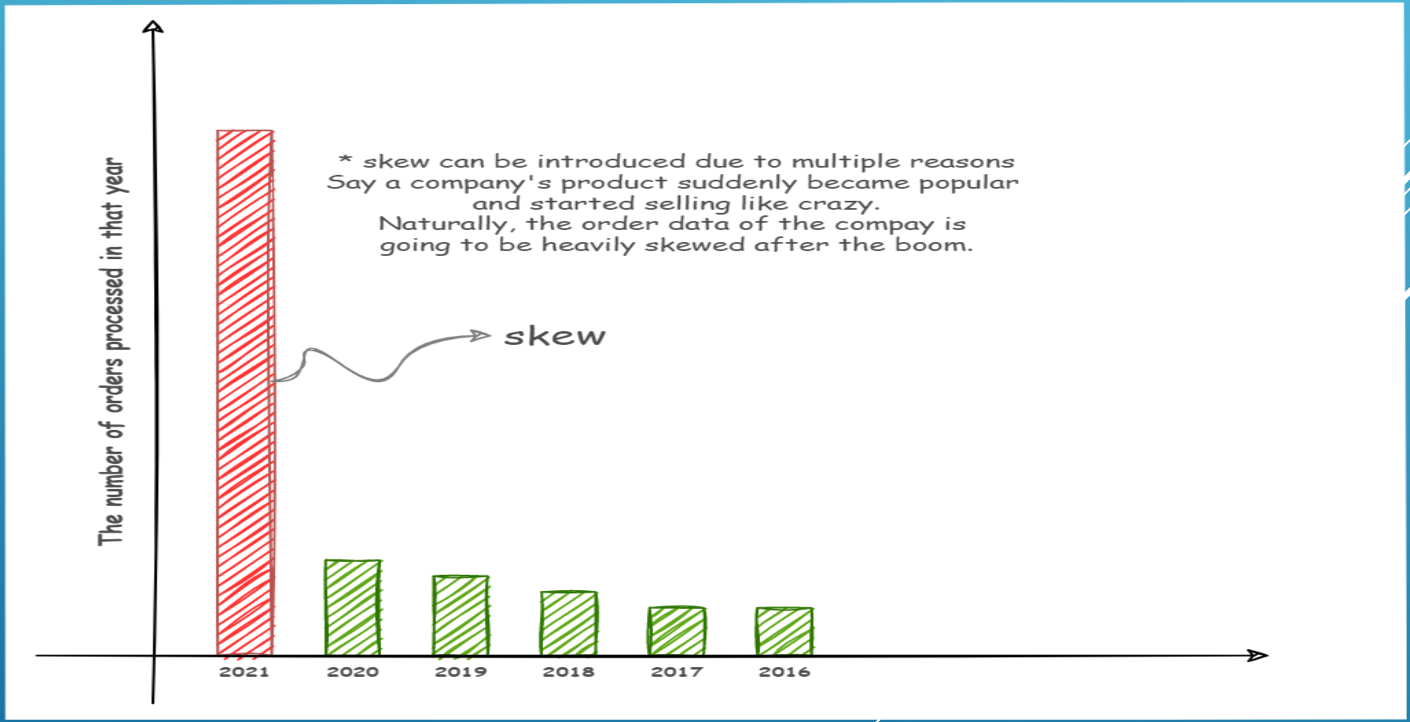
DATA SKEW

Data skew in Spark is when the data is unevenly distributed across the partitions. This can lead to performance problems, as some tasks will take much longer to complete than others.

There are a few things that can cause data skew in Spark, including:

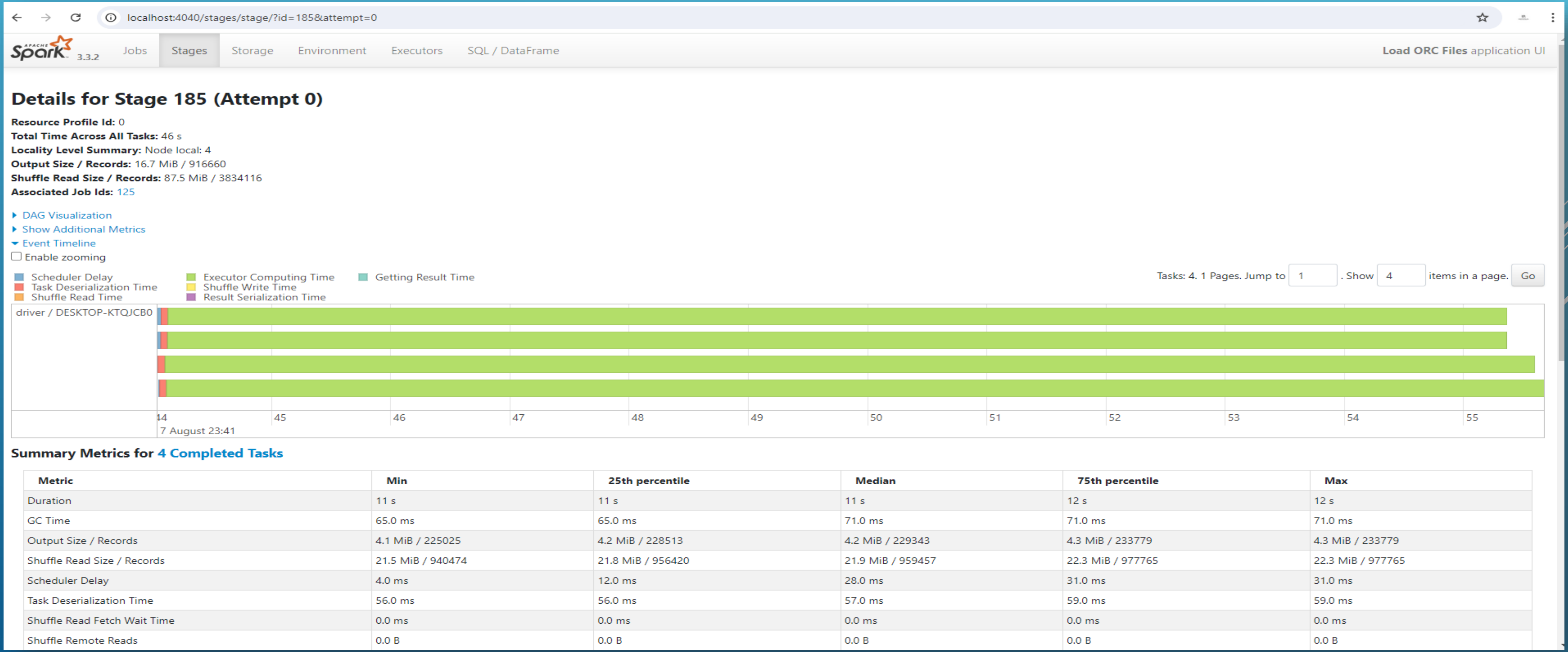
5(S) Basic Problems

Skew: Data in each partition is imbalanced.
Spill: File was written to disk memory due to insufficient RAM.
Shuffle: Data is moved between Spark executors during the run.
Storage: Too tiny file stored, file scanning and schema related.
Serialization: Segments of code have been distributed across the cluster.



How Can We Monitor Performance?

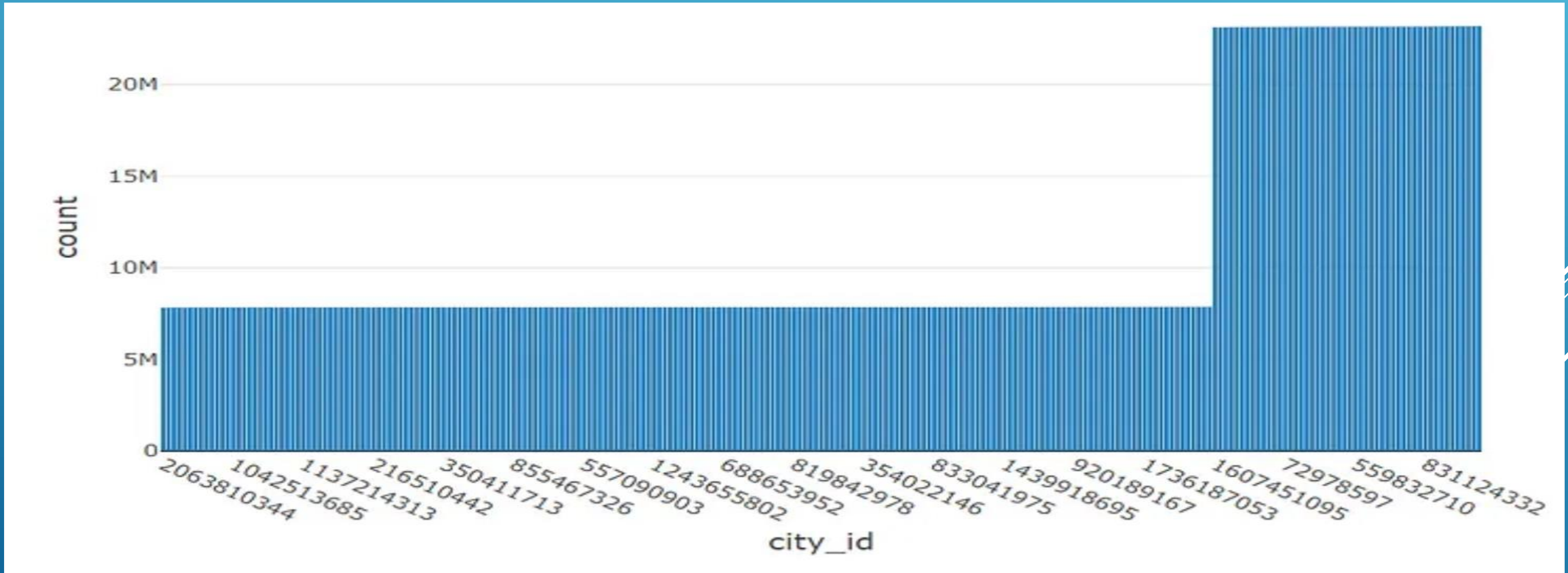
use “**Spark UI**” in menu as showing in the following picture.



Metric	Min	25th percentile	Median	75th percentile	Max
Duration	11 s	11 s	11 s	12 s	12 s
GC Time	65.0 ms	65.0 ms	71.0 ms	71.0 ms	71.0 ms
Output Size / Records	4.1 MiB / 225025	4.2 MiB / 228513	4.2 MiB / 229343	4.3 MiB / 233779	4.3 MiB / 233779
Shuffle Read Size / Records	21.5 MiB / 940474	21.8 MiB / 956420	21.9 MiB / 959457	22.3 MiB / 977765	22.3 MiB / 977765
Scheduler Delay	4.0 ms	12.0 ms	28.0 ms	31.0 ms	31.0 ms
Task Deserialization Time	56.0 ms	56.0 ms	57.0 ms	59.0 ms	59.0 ms
Shuffle Read Fetch Wait Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	0.0 ms
Shuffle Remote Reads	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B

Example (Skew)

The first data set is very skewed as per the count of the number of transactions (*transaction dataset*) by “**city_id**” which will be our **joining key.**



Skewness in join key “city_id” on transaction data

Stages

The last stage of the Spark job in this case is our performance testing write (**noop**). Then we will go to **SparkUI -> Stages** and look for our Stages ID, which is 7 for this job.

Configuration Notebooks (1) Libraries Event log Spark UI Driver Logs Metrics Apps Spark cluster UI - Master

Jobs Stages Storage Environment Executors SQL JDBC/ODBC Server Structured Streaming

Page: 1 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	3925523546683993925	Step B: How skewed? executeCollect at SmallTableCache.scala:62 +details	2022/02/27 15:22:35	1.2 min	8/8				
1	3925523546683993925	Step B: How skewed? collectResult at OutputAggregator.scala:194 +details	2022/02/27 15:23:48	3.8 min	104/104	2.5 GiB			1472.2 KiB
3	3925523546683993925	Step B: How skewed? collectResult at OutputAggregator.scala:194 +details	2022/02/27 15:27:39	42 s	2/2			1472.2 KiB	
4	3925523546683993925	Step C: Establish a baseline executeCollect at SmallTableCache.scala:62 +details	2022/02/27 15:29:54	44 s	1/1				
5	3925523546683993925	Step C: Establish a baseline save at command-799887080289471:13 +details	2022/02/27 15:30:38	17 s	1/1	37.0 MiB			78.1 MiB
6	3925523546683993925	Step C: Establish a baseline save at command-799887080289471:13 +details	2022/02/27 15:30:38	17 min	825/825	102.7 GiB			148.0 GiB
7	3925523546683993925	Step C: Establish a baseline save at command-799887080289471:13 +details	2022/02/27 15:47:48	4.2 min	832/832			148.0 GiB	

SparkUI -> Stages and click on description of selected stages id

Event Timeline

Scroll down a bit on pause on “**Event Timeline**”. You will spot the very long taking task time which means you are facing skewness in data!

Event Timeline — Baseline

Event Timeline

Then, scroll down a bit further. You will see "Summary Metrics".
"Shuffle Read Size" shows the amount of shuffle data across partitions. It is calculated into simple descriptive statistics.
And you can spot that the amount of data across partitions is very skewed!
Min to median populations is 0.0 M/0 records while 75th percentile to max is 435 MB to 2.6 GB !!

Summary Metrics for 832 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	2.0 ms	4.0 ms	9.0 ms	13 s	1.9 min
GC Time	0.0 ms	0.0 ms	0.0 ms	96.0 ms	5 s
Spill (memory)	0.0 B	0.0 B	0.0 B	544 MiB	5 GiB
Spill (disk)	0.0 B	0.0 B	0.0 B	238.8 MiB	2.4 GiB
Shuffle Read Size / Records	0.0 B / 0	0.0 B / 0	0.0 B / 0	435.1 MiB / 7829525	2.6 GiB / 46295875

Skew !

Showing 1 to 5 of 5 entries

Summary Metrics — Baseline

Aggregated Metrics by Executor

The last section on this page is “**Aggregated Metrics by Executor**”, which should stay below the “Summary Metrics”.
In this table we will look at 2 columns:
- **“Spill (Memory)” =** size of the de-serialized form of the spill data in memory
- **“Spill (Disk)” =** size of the serialized form of the data on disk.

Aggregated Metrics by Executor

Show 20 entries Search:

Executor ID	Logs	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Blacklisted	Shuffle Read Size / Records	Spill (Memory)	Spill (Disk)
2	stdout stderr	10.1.2.7:35059	14 min	108	0	0	108	false	18 GiB / 328071600	29.1 GiB	13.7 GiB
3	stdout stderr	10.1.2.15:40195	15 min	107	0	0	107	false	17.1 GiB / 311989561	28 GiB	13.3 GiB
4	stdout stderr	10.1.2.16:40947	15 min	170	0	0	170	false	18.8 GiB / 343022717	31.8 GiB	15 GiB
5	stdout stderr	10.1.2.8:37683	15 min	78	0	0	78	false	19.3 GiB / 349694107	34.1 GiB	16.1 GiB
6	stdout stderr	10.1.2.5:38935	14 min	58	0	0	58	false	18 GiB / 326998807	30.1 GiB	14.1 GiB
7	stdout stderr	10.1.2.6:44559	16 min	63	0	0	63	false	21.1 GiB / 381033702	37.7 GiB	17.7 GiB
8	stdout stderr	10.1.2.17:44473	14 min	129	0	0	129	false	18.1 GiB / 328033162	28.8 GiB	13.6 GiB
9	stdout stderr	10.1.2.18:43225	14 min	119	0	0	119	false	17.6 GiB / 319571721	29.6 GiB	14 GiB

Aggregated Metrics by Executor

Solution to Data Skew

Some strategies we can implement to make skew join faster.

- Skew hint (Databricks-specific)
- Salted skewed column: Split large partitions into smaller ones
- Adaptive Query Execution (Spark 3.x)
- Increase execution memory

In the first part, I will introduce only AQE and execution memory tuning

- “Spill (Memory)” = size of the de-serialized form of the spill data in memory
- “Spill (Disk)” = size of the serialized form of the data on disk.

Adaptive Query Execution (Spark 3.x)

To enable AQE in Spark just set configurations using the following scripts, and keep the remaining the same.

With a high level, this takes **19.01 minutes**! Its faster by 3 minutes.

spark.conf.set("spark.sql.adaptive.enables", true)
spark.conf.set("spark.sql.adaptive.skewedJoin.enabled", true)

Scala

```scala
sc.setJobDescription("Step E: Join with AQE")

// Enable AQE and the adaptive Skew Join
spark.conf.set("spark.sql.adaptive.enabled", true)
spark.conf.set("spark.sql.adaptive.skewedJoin.enabled", true)

// The default is 64 MB, but in this case we want to maintain 128m partitions
spark.conf.set("spark.sql.adaptive.advisoryPartitionSizeInBytes", "128m")

val ctyDF = spark.read.format("delta").load(ctyPath)   // Load the cities table

val trxDF = spark
  .read.format("delta").load(trxPath)                  // Load the transactions table
  //.hint("skew", "city_id")                           // Not required with AQE's spark.sql.adaptive.skewedJoin

trxDF
  .join(ctyDF, ctyDF("city_id") === trxDF("city_id")) // Join by city_id
  .write.format("noop").mode("overwrite").save()       // Execute a noop write to test
```

AQE setting

▸ (3) Spark Jobs
▸ ctyDF: org.apache.spark.sql.DataFrame = [city_id: integer, city: string ... 3 more fields]
▸ trxDF: org.apache.spark.sql.DataFrame = [transacted_at: timestamp, trx_id: string ... 4 more fields]

```
ctyDF: org.apache.spark.sql.DataFrame = [city_id: int, city: string ... 3 more fields]
trxDF: org.apache.spark.sql.DataFrame = [transacted_at: timestamp, trx_id: string ... 4 more fields]
Command took 19.01 minutes -- by wasurat.s
```

Join with AQE setting

Event Timeline

To enable AQE in Spark just set configurations using the following scripts, and keep the remaining the same.

spark.conf.set("spark.sql.adaptive.enables", true)
spark.conf.set("spark.sql.adaptive.skewedJoin.enabled", true)

Event Timeline with AQE