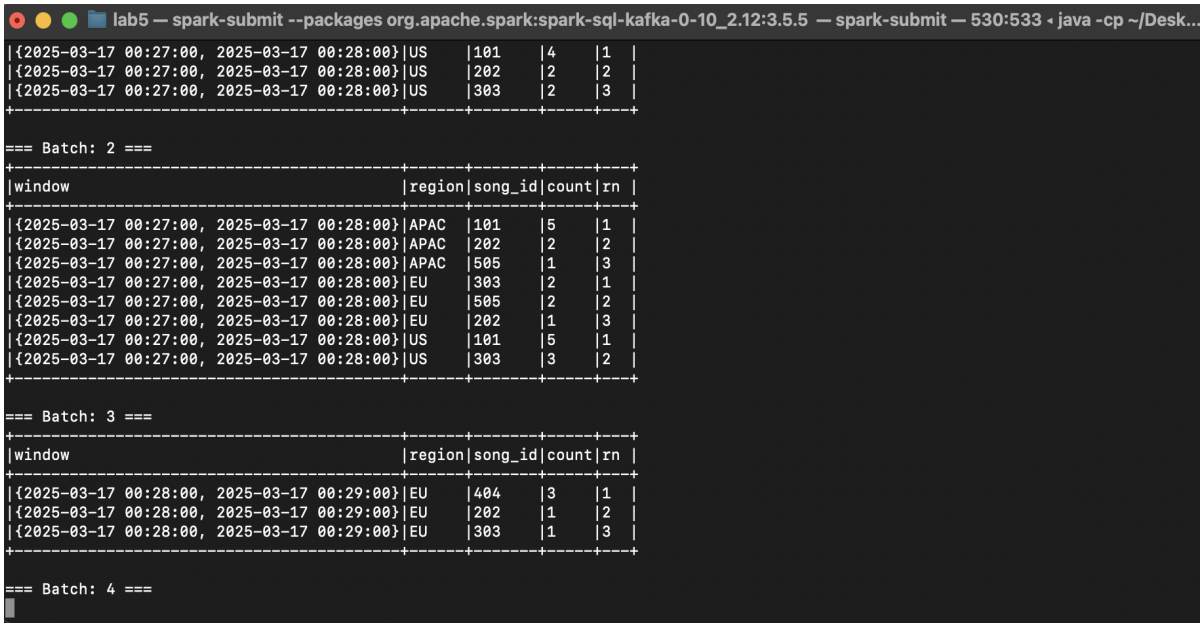
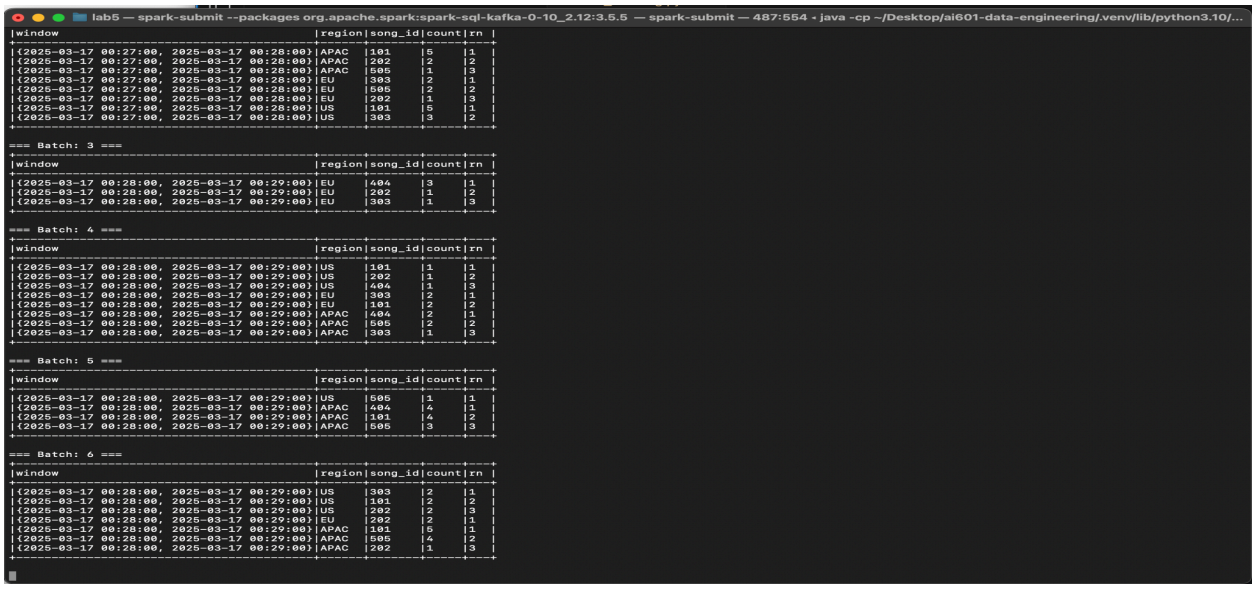
Varying the Time Window:

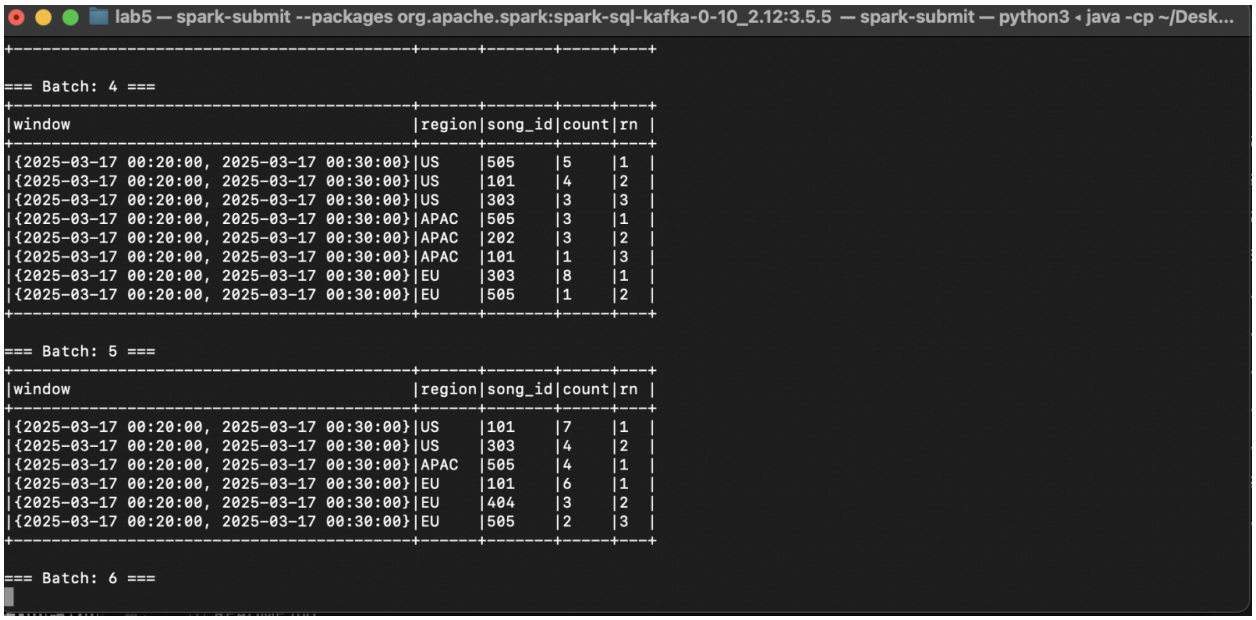
- **1-minute Window:**
  - "Now Trending" results will update very frequently
  - Latency is very low, as new changes are being detected almost instantaneously.
  - Due to low latency this could lead to a less stable "Now Trending" list.
- **10-minute Window:**
  - "Now Trending" results will update less frequently, reflecting a longer period of activity.
  - Latency is higher, as it takes longer for the list to be updated.
  - The list is more stable.
  - This provides a more reliable "Now Trending" list, but it might not be up to date all the time.
- **Trade-offs:**
  - **Latency vs. Stability:** Shorter windows provide lower latency but also less stability, while longer windows offer higher stability but also increased latency.



Now trending list with 1 minute window



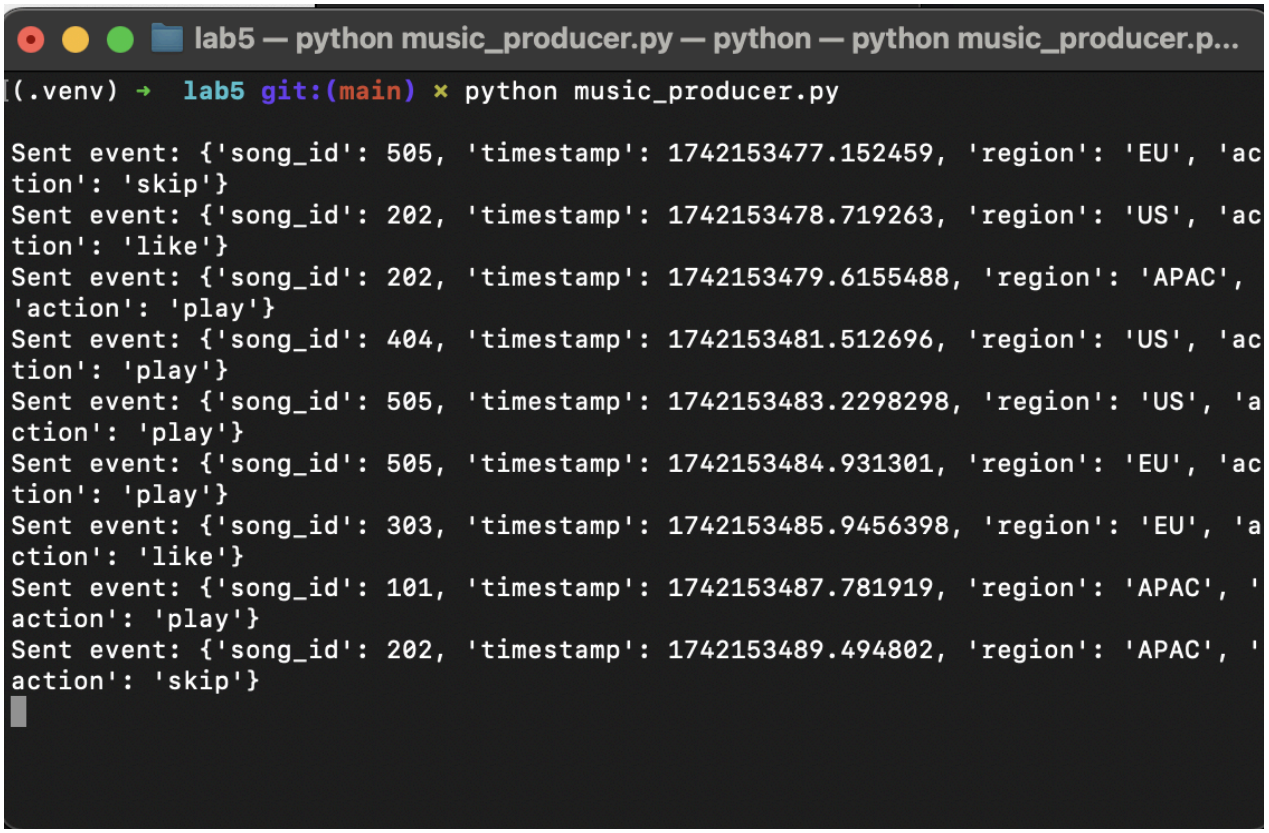
Now trending list with 1 minute window

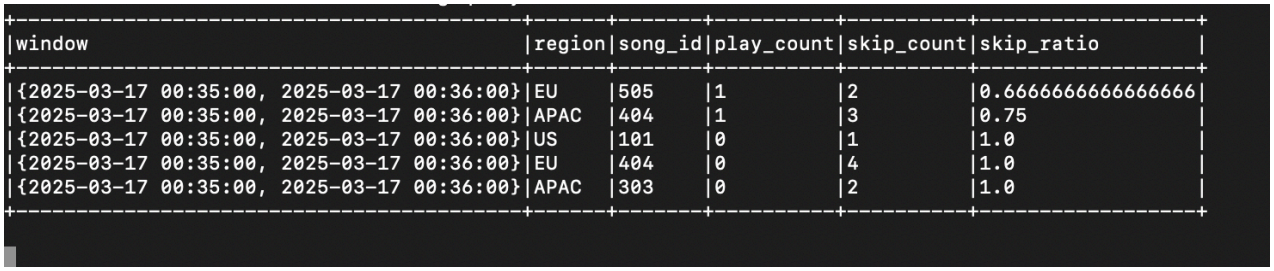


Now trending list with 10 minute window

Add Skip/Like Actions:

- The provided code now includes "like" actions in the aggregation.
- We also calculate the skip ratio now.



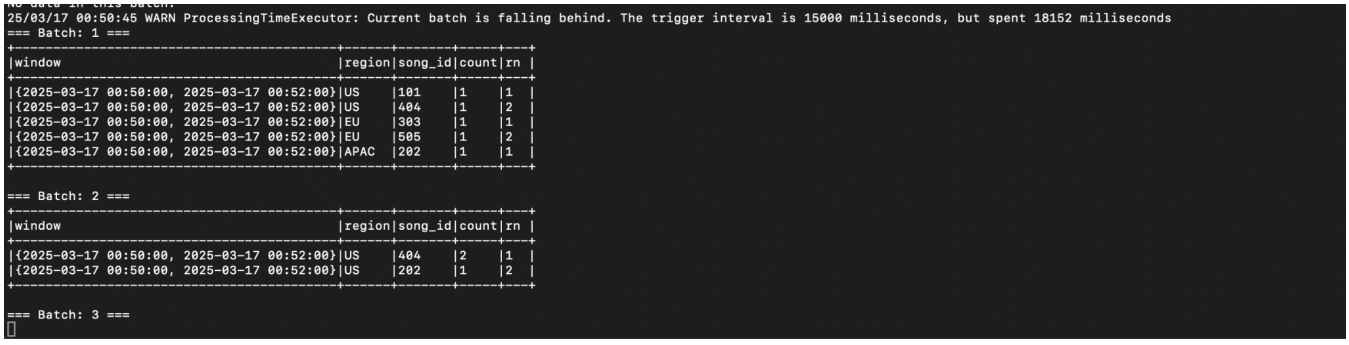


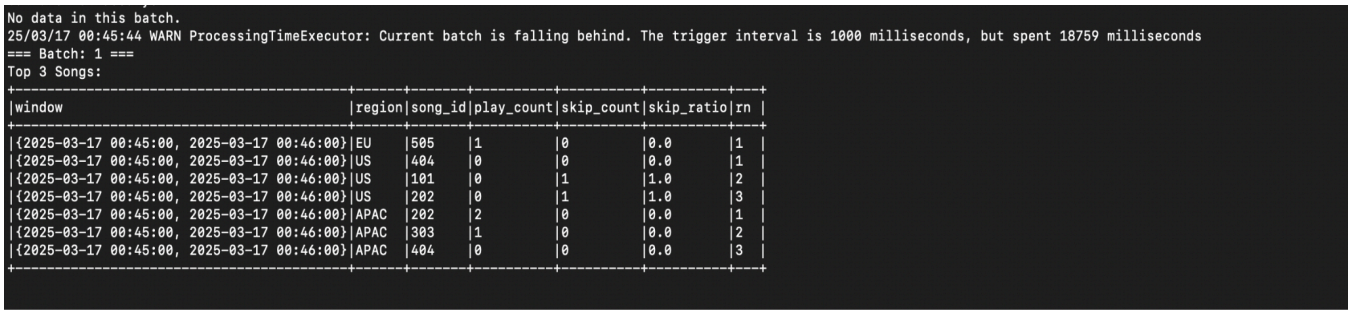
window	region	song_id	play_count	skip_count	skip_ratio
{2025-03-17 00:35:00, 2025-03-17 00:36:00}	EU	505	1	2	0.6666666666666666
{2025-03-17 00:35:00, 2025-03-17 00:36:00}	APAC	404	1	3	0.75
{2025-03-17 00:35:00, 2025-03-17 00:36:00}	US	101	0	1	1.0
{2025-03-17 00:35:00, 2025-03-17 00:36:00}	EU	404	0	4	1.0
{2025-03-17 00:35:00, 2025-03-17 00:36:00}	APAC	303	0	2	1.0

Skip Ratio and Skip/Like Actions Implementation

Set a Different Micro-Batch Interval:

- **1-second Trigger:**
  - PySpark processes new data every second, trying to provide very frequent updates.
  - CPU usage will be higher, as Spark is constantly checking for and processing new data.
  - Results appear faster.
- **10-15 second Trigger:**
  - PySpark tries to process data every 10-15 seconds based on the trigger.
  - CPU usage will be lower than a 1 second trigger.
  - Results will appear every 10-15 seconds depending on how faster it can process data.
- **Performance vs. Overhead:**
  - Shorter trigger intervals provide lower latency and provide faster updates but increase CPU overhead.
  - Longer intervals reduce CPU usage but introduce more latency.
  - The optimal choice depends on the application's needs. Real-time updates may require a shorter interval despite the overhead, while resource efficiency favors a longer one. We can decide on the tradeoff based on the task at hand.





Trigger intervals of 1 second and 15 seconds.

* Notice the warning that Current batch is falling behind because the window is much shorter than the processing time required.