Week 4: Stream processing for Beat Blast

Analysis & Report

Tejashwini Saravanan

School of Business, Government, and Economics

ISM6362 Big Data and Cloud-Based Tools

Professor: Anthony Kwiatkowski

19 july 2025, Saturday

Summer 2025

Objective :

This assignment involved simulating user interaction events on a music platform (BeatBlast) and building a real-time data processing pipeline using Apache Spark Structured Streaming. The goal was to simulate JSON event streams, apply windowed aggregations, and write the output in a partitioned format suitable for long-term analytics.

1. Data Simulation Script

A Python script was used to simulate songPlay, songLike, appOpen, and songSkip events. The script randomly selected values for songId, userId, sessionId, country, and platform. It generated JSON files every second to simulate a real-time event stream.

{

"eventType": "songPlay",

"eventTimestamp": "2025-07-18T22:56:32.197Z",

"songId": "song_001",

"sessionId": "sess_2",

"userId": "user_3",

"platform": "android",

"country": "US"

}

2. Structured Streaming Pipeline (PySpark)

- **Used SparkSession to read streaming data from the simulated input folder (/content/stream_input/).**
- **Defined an explicit schema for the JSON input.**
- **Added a processingTimestamp column and converted the event timestamp to TimestampType.**
- **Wrote output as Parquet files partitioned by:**

- **year**
- **month**
- **day**
- **country**

3. Aggregation Queries

A. Popular Songs

- **Grouped by 5-minute tumbling windows and songId.**
- **Counted the number of songPlay events.**

Sample Output:

Window Start	Window End	songId	play_count
2025-07-18 22:50:00	2025-07-18 23:00:00	song_001	12
2025-07-18 22:50:00	2025-07-18 23:00:00	song_002	13

B. Active User Sessions

- **Counted distinct sessionIds from appOpen events in a 10-minute sliding window (every 5 minutes).**

Sample Output:

Window Start	Window End	sessionId	activity
2025-07-18 22:40:00	2025-07-18 22:50:00	sess_1	1
2025-07-18 22:50:00	2025-07-18 23:00:00	sess_3	15

4. Partitioned Storage Structure

Output files were saved to /content/beatblast_datalake/song_plays/ with the following structure:

sql

CopyEdit

song_plays/

year=2025/

month=07/

day=18/

country=US/

country=IN/

country=GB/

country=CA/

5. Justifications & Conceptual Questions

Q: Why did you choose a 10-minute watermark?

To allow for late-arriving data within a realistic buffer window, while avoiding holding too much data in memory.

Q: Why these partition columns and order?

year/month/day/country allows easy querying by time and geography. Analytics teams often filter by date and country.

Q: Would you use the same partitioning for songLike?

Yes, likely the same it helps unify datasets for joins and time-based queries.

Q: Why use Structured Streaming over DStreams?

Structured Streaming provides a declarative API, exactly-once guarantees, better SQL support, and easier integration with modern data lakes.

Q: How would you handle very late data?

Use longer watermarks, enable append-only + reprocessing jobs, or configure event-time joins with allowed lateness.

Conclusion

This assignment demonstrated end-to-end stream processing using Apache Spark in Google Colab. I simulated real-time JSON events, processed them with watermarks and time windows, and saved them in a partitioned structure ideal for analytics.