PES UNIVERSITY
100 feet Ring Road, BSK 3rd Stage
Bengaluru 560085
Department of Computer Science and Engineering

Department of Computer Science and Engineering
B. Tech. CSE - 6th Semester
Jan – May 2025

UE22CS343BB3
DATABASE TECHNOLOGIES (DBT)

PROJECT REPORT
on

Twitch Stream Analysis

Submitted by : Team #: _____

Mohul YP	PES1UG22CS360	6F	Sai Mohananshu	PES1UG22CS501	6I
Rupak R	PES1UG22CS488	6H			

Class of Prof. Nagasundari

Table of Contents

Introduction

This project is a real-time data engineering pipeline designed to collect, process, and analyze live streaming data from the Twitch platform. By integrating key big data technologies—Apache Kafka, Apache Spark, and MySQL—the system demonstrates an end-to-end solution for handling high-velocity streaming data.

The pipeline begins with a Kafka producer that interfaces with the Twitch API to fetch live stream metadata, such as streamer names, viewer counts, and game categories. This data is published to a Kafka topic based on dynamic criteria like viewer thresholds. Apache Spark then consumes this stream using Structured Streaming to process the data in real time. During processing, Spark performs various analytical tasks such as identifying trending games and top streamers.

The final stage involves storing the processed results in a MySQL database, enabling persistent storage and easy access for reporting or dashboarding. This architecture not only supports real-time analytics but also showcases the scalability and modularity of modern data processing frameworks.

By simulating a real-world use case, this project serves as a foundational template for anyone looking to build streaming applications for live data insights and decision-making.

Installation of Software

To run the Twitch data pipeline project, you'll need to install the following software:

1. Python 3.8+ – for running the scripts.
2. Apache Kafka – for real-time message streaming.
3. Apache Spark (with PySpark) – for stream processing.
4. MySQL Server – for storing processed data.
5. Twitch API credentials – for accessing Twitch data.

Input Data

https://api.twitch.tv/helix/streams

Source

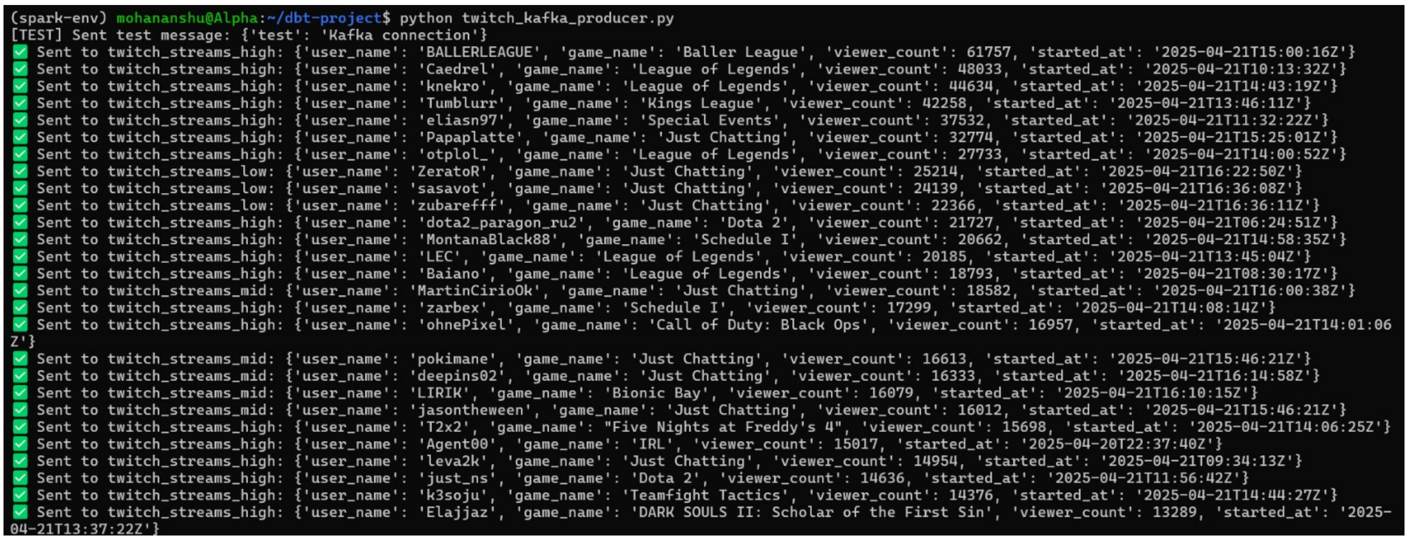
Twitch API Endpoint
This API endpoint returns data about the top currently active live streams on Twitch.

Description

The endpoint provides JSON-formatted metadata for each stream, including:

- Streamer's username
- Game title
- Viewer count
- Language
- Stream start time

Input streaming data from the API



```
(spark-env) mohananshu@Alpha:~/dbt-project$ python twitch_kafka_producer.py
[TEST] Sent test message: {'test': 'Kafka connection'}
✅ Sent to twitch_streams_high: {'user_name': 'BALLERLEAGUE', 'game_name': 'Baller League', 'viewer_count': 61757, 'started_at': '2025-04-21T15:00:16Z'}
✅ Sent to twitch_streams_high: {'user_name': 'Caedrel', 'game_name': 'League of Legends', 'viewer_count': 48033, 'started_at': '2025-04-21T10:13:32Z'}
✅ Sent to twitch_streams_high: {'user_name': 'knekro', 'game_name': 'League of Legends', 'viewer_count': 44634, 'started_at': '2025-04-21T14:43:19Z'}
✅ Sent to twitch_streams_high: {'user_name': 'Tumblurr', 'game_name': 'Kings League', 'viewer_count': 42258, 'started_at': '2025-04-21T13:46:11Z'}
✅ Sent to twitch_streams_high: {'user_name': 'eliasn97', 'game_name': 'Special Events', 'viewer_count': 37532, 'started_at': '2025-04-21T11:32:22Z'}
✅ Sent to twitch_streams_high: {'user_name': 'Papaplatte', 'game_name': 'Just Chatting', 'viewer_count': 32774, 'started_at': '2025-04-21T15:25:01Z'}
✅ Sent to twitch_streams_high: {'user_name': 'otplol_', 'game_name': 'League of Legends', 'viewer_count': 27733, 'started_at': '2025-04-21T14:00:52Z'}
✅ Sent to twitch_streams_low: {'user_name': 'ZeratoR', 'game_name': 'Just Chatting', 'viewer_count': 25214, 'started_at': '2025-04-21T16:22:50Z'}
✅ Sent to twitch_streams_low: {'user_name': 'sasavot', 'game_name': 'Just Chatting', 'viewer_count': 24139, 'started_at': '2025-04-21T16:36:08Z'}
✅ Sent to twitch_streams_low: {'user_name': 'zubarefff', 'game_name': 'Just Chatting', 'viewer_count': 22366, 'started_at': '2025-04-21T16:36:11Z'}
✅ Sent to twitch_streams_high: {'user_name': 'dota2_paragon_ru2', 'game_name': 'Dota 2', 'viewer_count': 21727, 'started_at': '2025-04-21T06:24:51Z'}
✅ Sent to twitch_streams_high: {'user_name': 'MontanaBlack88', 'game_name': 'Schedule I', 'viewer_count': 20662, 'started_at': '2025-04-21T14:58:35Z'}
✅ Sent to twitch_streams_high: {'user_name': 'LEC', 'game_name': 'League of Legends', 'viewer_count': 20185, 'started_at': '2025-04-21T13:45:04Z'}
✅ Sent to twitch_streams_high: {'user_name': 'Baiano', 'game_name': 'League of Legends', 'viewer_count': 18793, 'started_at': '2025-04-21T08:30:17Z'}
✅ Sent to twitch_streams_mid: {'user_name': 'MartinCirioOk', 'game_name': 'Just Chatting', 'viewer_count': 18582, 'started_at': '2025-04-21T16:00:38Z'}
✅ Sent to twitch_streams_high: {'user_name': 'zarbex', 'game_name': 'Schedule I', 'viewer_count': 17299, 'started_at': '2025-04-21T14:08:14Z'}
✅ Sent to twitch_streams_high: {'user_name': 'ohnePixel', 'game_name': 'Call of Duty: Black Ops', 'viewer_count': 16957, 'started_at': '2025-04-21T14:01:06
Z'}
✅ Sent to twitch_streams_mid: {'user_name': 'pokimane', 'game_name': 'Just Chatting', 'viewer_count': 16613, 'started_at': '2025-04-21T15:46:21Z'}
✅ Sent to twitch_streams_mid: {'user_name': 'deepins02', 'game_name': 'Just Chatting', 'viewer_count': 16333, 'started_at': '2025-04-21T16:14:58Z'}
✅ Sent to twitch_streams_mid: {'user_name': 'LIRIK', 'game_name': 'Bionic Bay', 'viewer_count': 16079, 'started_at': '2025-04-21T16:10:15Z'}
✅ Sent to twitch_streams_mid: {'user_name': 'jasontheween', 'game_name': 'Just Chatting', 'viewer_count': 16012, 'started_at': '2025-04-21T15:46:21Z'}
✅ Sent to twitch_streams_high: {'user_name': 'T2x2', 'game_name': "Five Nights at Freddy's 4", 'viewer_count': 15698, 'started_at': '2025-04-21T14:06:25Z'}
✅ Sent to twitch_streams_high: {'user_name': 'Agent00', 'game_name': 'IRL', 'viewer_count': 15017, 'started_at': '2025-04-20T22:37:40Z'}
✅ Sent to twitch_streams_high: {'user_name': 'leva2k', 'game_name': 'Just Chatting', 'viewer_count': 14954, 'started_at': '2025-04-21T09:34:13Z'}
✅ Sent to twitch_streams_high: {'user_name': 'just_ns', 'game_name': 'Dota 2', 'viewer_count': 14636, 'started_at': '2025-04-21T11:56:42Z'}
✅ Sent to twitch_streams_high: {'user_name': 'k3soju', 'game_name': 'Teamfight Tactics', 'viewer_count': 14376, 'started_at': '2025-04-21T14:44:27Z'}
✅ Sent to twitch_streams_high: {'user_name': 'Elajjaz', 'game_name': 'DARK SOULS II: Scholar of the First Sin', 'viewer_count': 13289, 'started_at': '2025-
04-21T13:37:22Z'}
```

This data is refreshed in real-time and serves as the raw input for the Kafka producer, forming the basis for further streaming and batch analytics in the project.

Streaming Mode Experiment

Description

This experiment involves using Apache Spark's Structured Streaming to process real-time Twitch stream data as it flows in via Apache Kafka. The script (spark_streaming.py) subscribes to a Kafka topic, processes stream metadata like viewer count and game category, and prints analytical summaries such as the most popular games and streamers.

Windows

If running on Windows:

- Ensure Kafka and Zookeeper are properly configured and running.
- Spark should be installed with Hadoop binaries compatible with Windows.
- Set environment variables for JAVA_HOME, SPARK_HOME, and PYSPARK_PYTHON.
- Use WSL (Windows Subsystem for Linux) if issues persist with native Spark on Windows.

Results

- Real-time logs show aggregated metrics (top 5 games and streamers).
- Processed data can be stored in MySQL (via batch_processing.py).
- Confirms that Spark is capable of handling real-time data feeds and performing meaningful analytics on-the-fly.

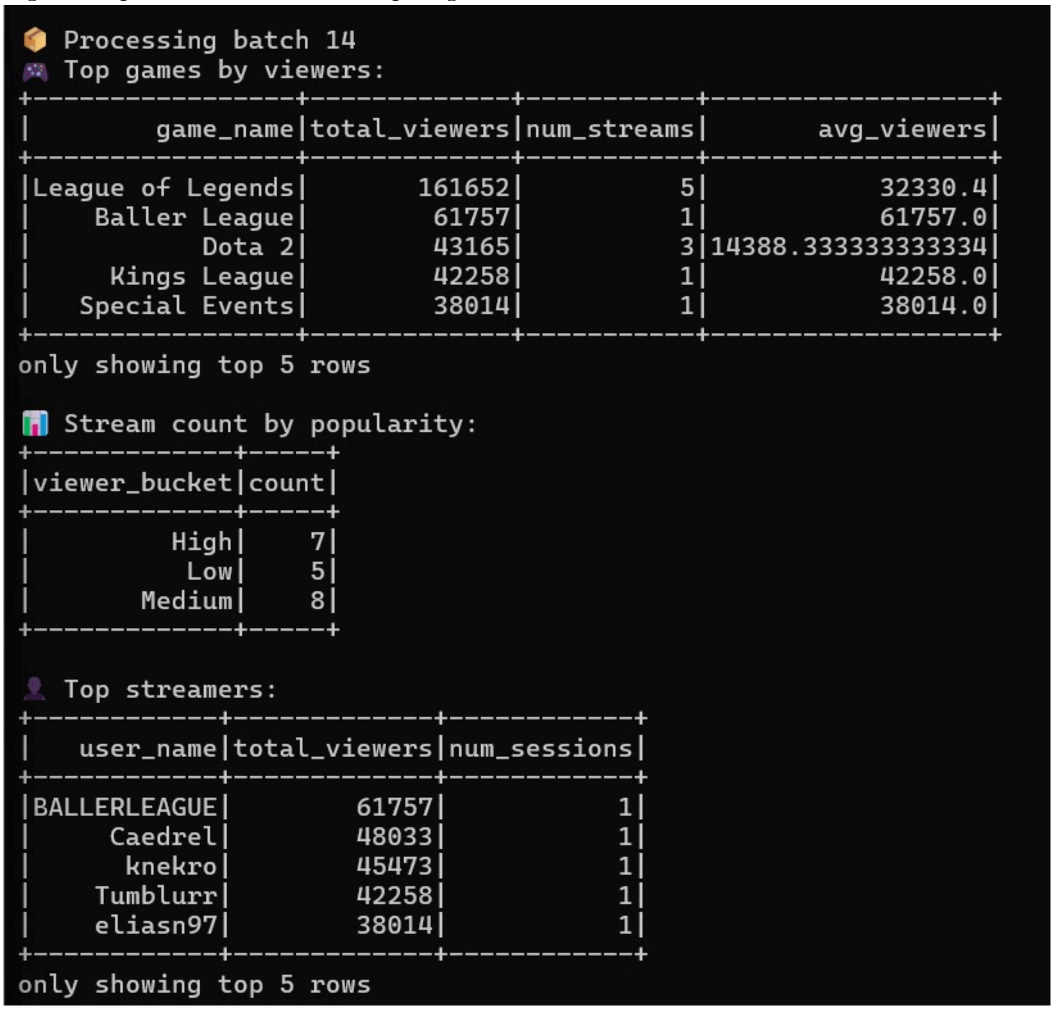
We have set up 3 topics listening to the streaming data from the API.
The topics are categorized as high, mid, low based on the runtime duration of the stream.

The reasoning behind this is that we will be able to use analytics to compare the viewership and retain performance of various streamers.

This will let companies that want to sponsor or pay for advertisements identify streams which will give them a maximum return on investment.

outputs for streaming data : (each batch is basically a window)

topic : high (streams with a high uptime)

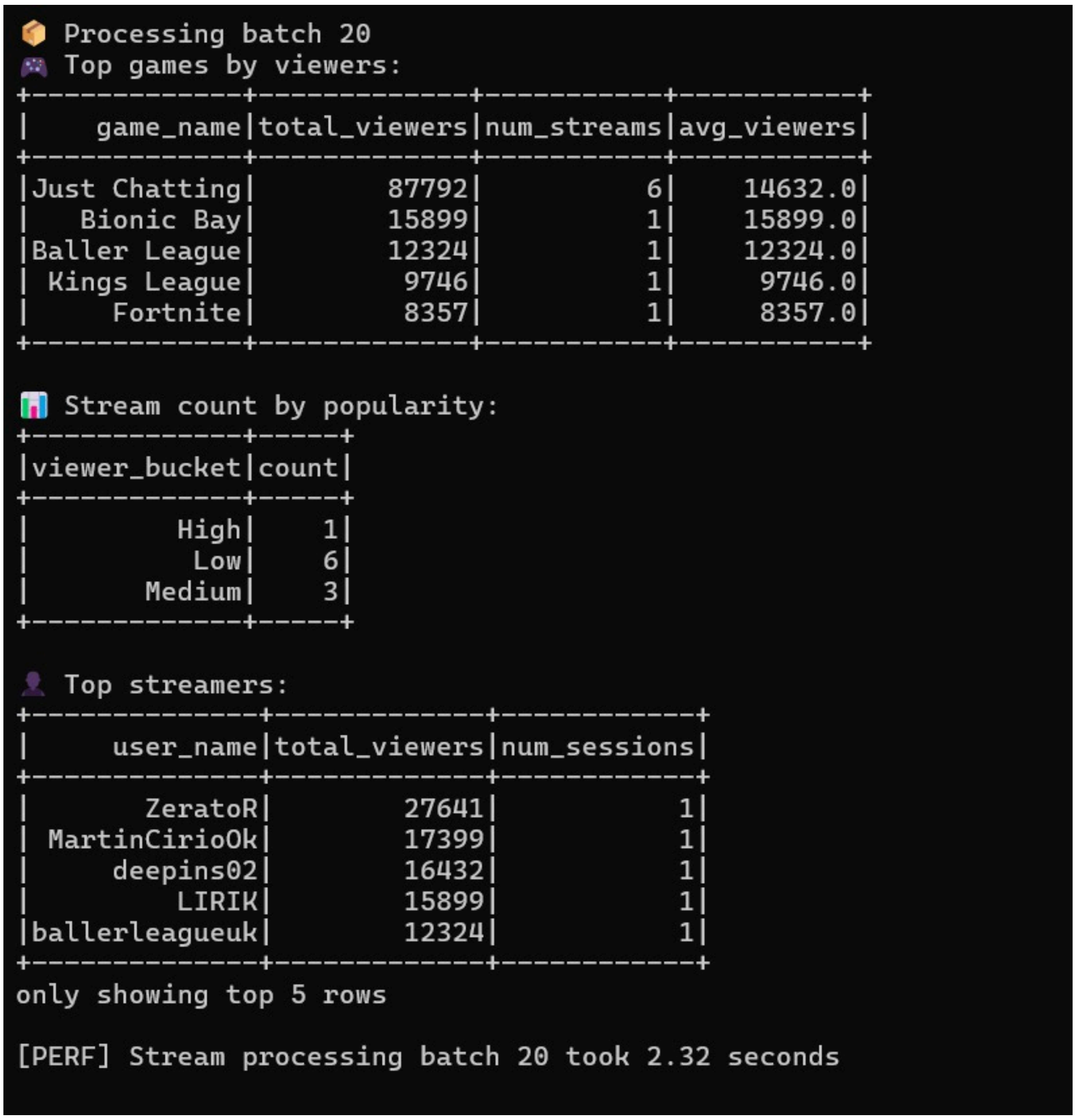


```
📦 Processing batch 14
🎮 Top games by viewers:
+-----------------+-------------+-----------+------------------+
|        game_name|total_viewers|num_streams|       avg_viewers|
+-----------------+-------------+-----------+------------------+
|League of Legends|       161652|          5|           32330.4|
|    Baller League|        61757|          1|           61757.0|
|           Dota 2|        43165|          3|14388.333333333334|
|     Kings League|        42258|          1|           42258.0|
|   Special Events|        38014|          1|           38014.0|
+-----------------+-------------+-----------+------------------+
only showing top 5 rows

📊 Stream count by popularity:
+-------------+-----+
|viewer_bucket|count|
+-------------+-----+
|         High|    7|
|          Low|    5|
|       Medium|    8|
+-------------+-----+

👤 Top streamers:
+------------+-------------+------------+
|   user_name|total_viewers|num_sessions|
+------------+-------------+------------+
|BALLERLEAGUE|        61757|           1|
|     Caedrel|        48033|           1|
|      knekro|        45473|           1|
|    Tumblurr|        42258|           1|
|    eliasn97|        38014|           1|
+------------+-------------+------------+
only showing top 5 rows
```

topic :mid (streams with a medium uptime)



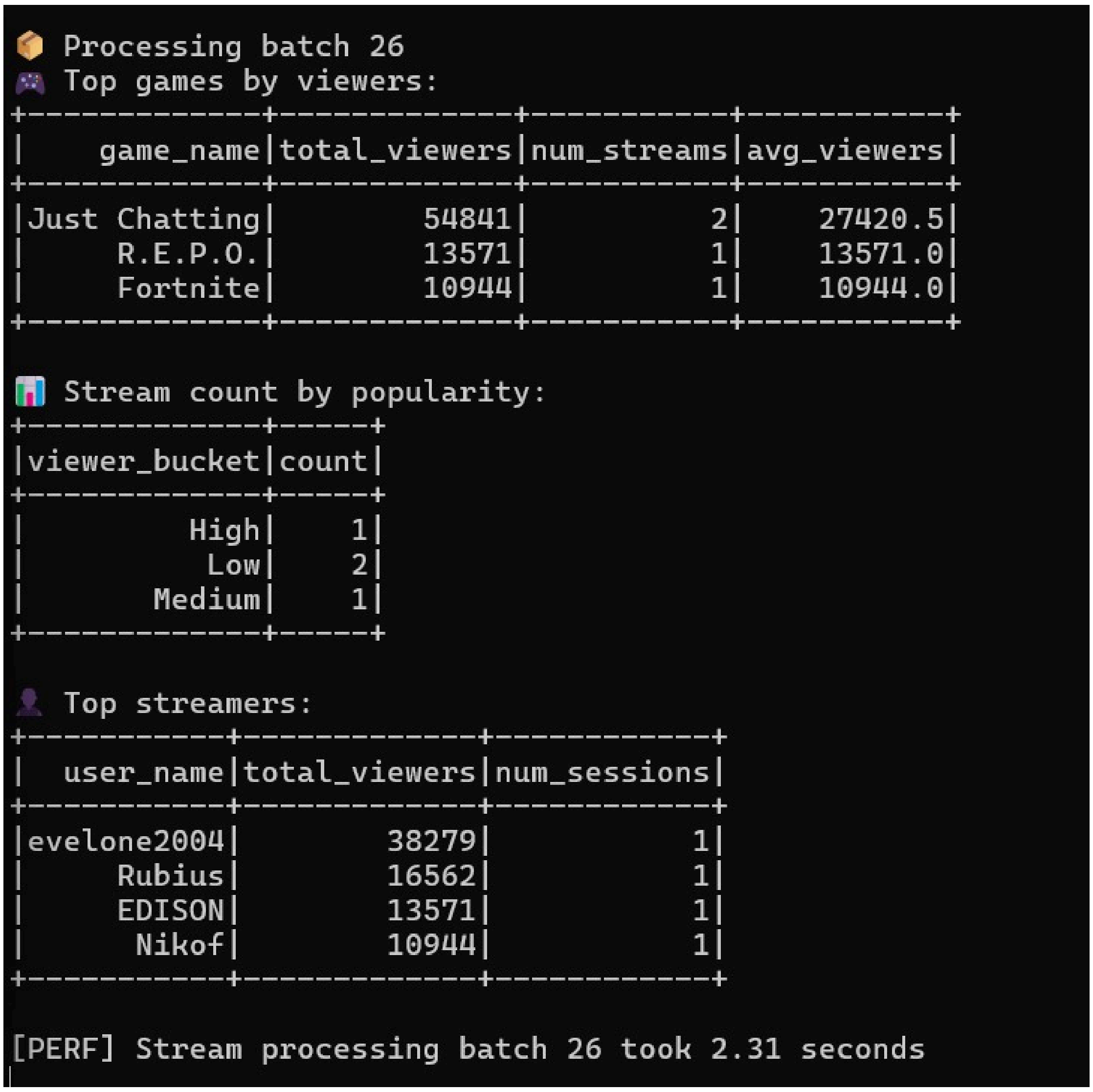
```
📦 Processing batch 20
🎮 Top games by viewers:
+-------------+-------------+-----------+-----------+
|    game_name|total_viewers|num_streams|avg_viewers|
+-------------+-------------+-----------+-----------+
|Just Chatting|        87792|          6|    14632.0|
|   Bionic Bay|        15899|          1|    15899.0|
|Baller League|        12324|          1|    12324.0|
| Kings League|         9746|          1|     9746.0|
|     Fortnite|         8357|          1|     8357.0|
+-------------+-------------+-----------+-----------+

📊 Stream count by popularity:
+-------------+-----+
|viewer_bucket|count|
+-------------+-----+
|         High|    1|
|          Low|    6|
|       Medium|    3|
+-------------+-----+

👤 Top streamers:
+--------------+-------------+------------+
|     user_name|total_viewers|num_sessions|
+--------------+-------------+------------+
|       ZeratoR|        27641|           1|
| MartinCirioOk|        17399|           1|
|     deepins02|        16432|           1|
|         LIRIK|        15899|           1|
|ballerleagueuk|        12324|           1|
+--------------+-------------+------------+
only showing top 5 rows

[PERF] Stream processing batch 20 took 2.32 seconds
```

topic : low (streams which have recently started)



```
📦 Processing batch 26
🎮 Top games by viewers:
+------------+-------------+-----------+-----------+
|   game_name|total_viewers|num_streams|avg_viewers|
+------------+-------------+-----------+-----------+
|Just Chatting|        54841|          2|    27420.5|
|     R.E.P.O.|        13571|          1|    13571.0|
|     Fortnite|        10944|          1|    10944.0|
+------------+-------------+-----------+-----------+

📊 Stream count by popularity:
+-------------+-----+
|viewer_bucket|count|
+-------------+-----+
|         High|    1|
|          Low|    2|
|       Medium|    1|
+-------------+-----+

👤 Top streamers:
+----------+-------------+------------+
|  user_name|total_viewers|num_sessions|
+----------+-------------+------------+
|evelone2004|        38279|           1|
|     Rubius|        16562|           1|
|     EDISON|        13571|           1|
|      Nikof|        10944|           1|
+----------+-------------+------------+

[PERF] Stream processing batch 26 took 2.31 seconds
```

Batch Mode Experiment

Description

In the batch mode experiment, historical Twitch stream data is processed using Apache Spark in batch mode instead of streaming. The script batch_processing.py reads a predefined dataset (simulated or from Kafka), processes it to extract key insights like top games, viewer distribution, and active streamers, and stores the results in a MySQL database. This simulates offline processing for trend analysis and reporting.

Data Size

The experiment can process any reasonable volume of previously streamed data. For this project, a dataset containing several hundred records was used to mimic an hour's worth of stream activity, offering a compact yet representative workload.

Results

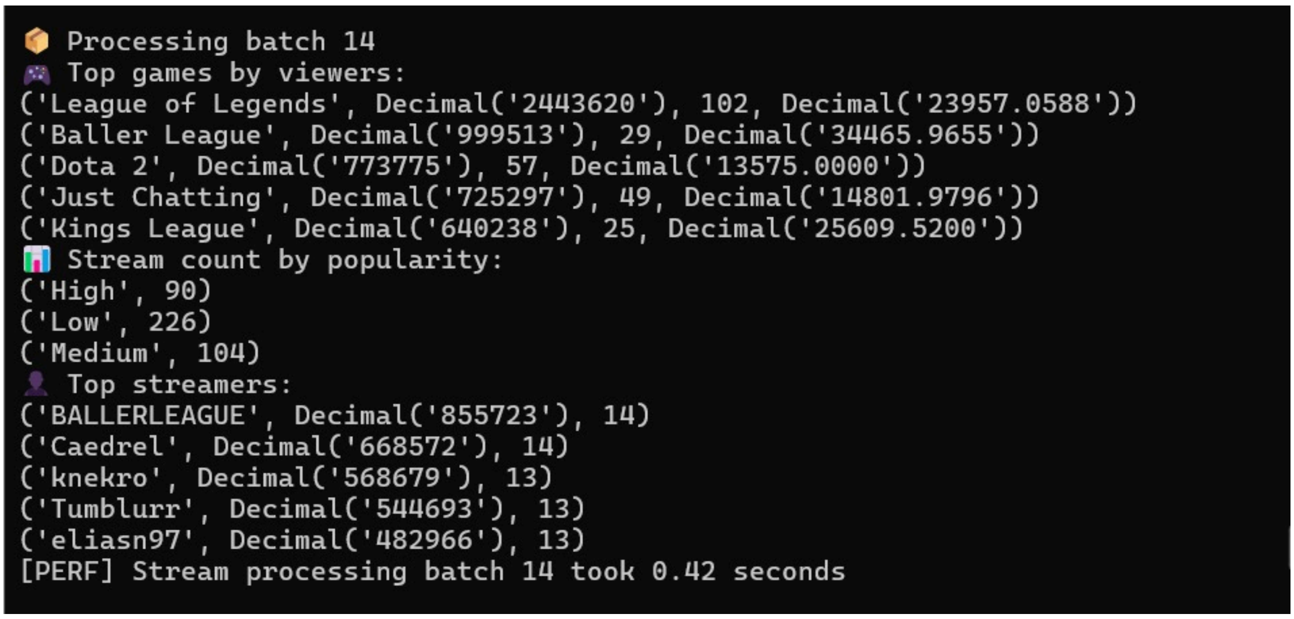
- Successfully extracted and stored analytics in MySQL.
- Queries run on MySQL tables showed correct aggregation of stream data.
- Compared to streaming mode, batch mode allowed more complex transformations and aggregations due to relaxed latency constraints.

We have performed batch processing in two aspects :

1) continuous ingestion in MySQL and performing queries similar to those performed on the spark streaming set.

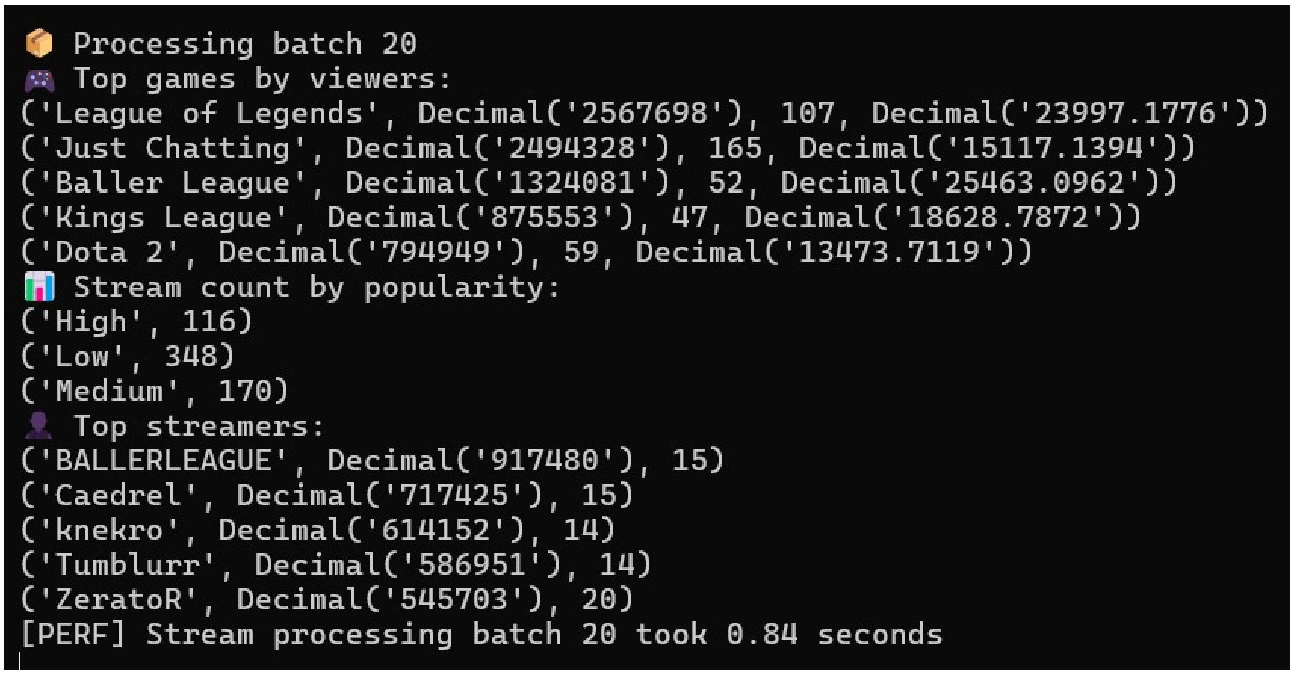
   This lets us incrementally monitor changing values while comparing stream vs batch processing.

topic : high (streams with a high uptime)



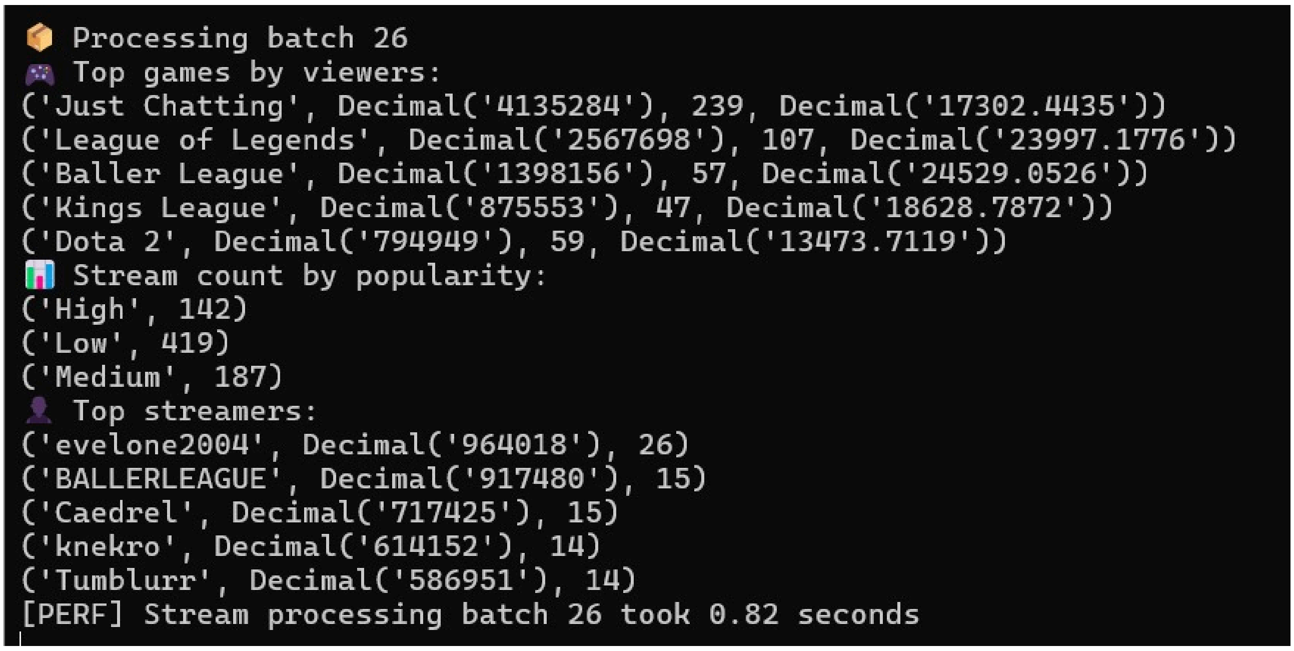
```
📦 Processing batch 14
🎮 Top games by viewers:
('League of Legends', Decimal('2443620'), 102, Decimal('23957.0588'))
('Baller League', Decimal('999513'), 29, Decimal('34465.9655'))
('Dota 2', Decimal('773775'), 57, Decimal('13575.0000'))
('Just Chatting', Decimal('725297'), 49, Decimal('14801.9796'))
('Kings League', Decimal('640238'), 25, Decimal('25609.5200'))
📊 Stream count by popularity:
('High', 90)
('Low', 226)
('Medium', 104)
👤 Top streamers:
('BALLERLEAGUE', Decimal('855723'), 14)
('Caedrel', Decimal('668572'), 14)
('knekro', Decimal('568679'), 13)
('Tumblurr', Decimal('544693'), 13)
('eliasn97', Decimal('482966'), 13)
[PERF] Stream processing batch 14 took 0.42 seconds
```

topic :mid (streams with a medium uptime)



```
📦 Processing batch 20
🎮 Top games by viewers:
('League of Legends', Decimal('2567698'), 107, Decimal('23997.1776'))
('Just Chatting', Decimal('2494328'), 165, Decimal('15117.1394'))
('Baller League', Decimal('1324081'), 52, Decimal('25463.0962'))
('Kings League', Decimal('875553'), 47, Decimal('18628.7872'))
('Dota 2', Decimal('794949'), 59, Decimal('13473.7119'))
📊 Stream count by popularity:
('High', 116)
('Low', 348)
('Medium', 170)
👤 Top streamers:
('BALLERLEAGUE', Decimal('917480'), 15)
('Caedrel', Decimal('717425'), 15)
('knekro', Decimal('614152'), 14)
('Tumblurr', Decimal('586951'), 14)
('ZeratoR', Decimal('545703'), 20)
[PERF] Stream processing batch 20 took 0.84 seconds
```

topic : low (streams which have recently started)



```
📦 Processing batch 26
🎮 Top games by viewers:
('Just Chatting', Decimal('4135284'), 239, Decimal('17302.4435'))
('League of Legends', Decimal('2567698'), 107, Decimal('23997.1776'))
('Baller League', Decimal('1398156'), 57, Decimal('24529.0526'))
('Kings League', Decimal('875553'), 47, Decimal('18628.7872'))
('Dota 2', Decimal('794949'), 59, Decimal('13473.7119'))
📊 Stream count by popularity:
('High', 142)
('Low', 419)
('Medium', 187)
👤 Top streamers:
('evelone2004', Decimal('964018'), 26)
('BALLERLEAGUE', Decimal('917480'), 15)
('Caedrel', Decimal('717425'), 15)
('knekro', Decimal('614152'), 14)
('Tumblurr', Decimal('586951'), 14)
[PERF] Stream processing batch 26 took 0.82 seconds
```

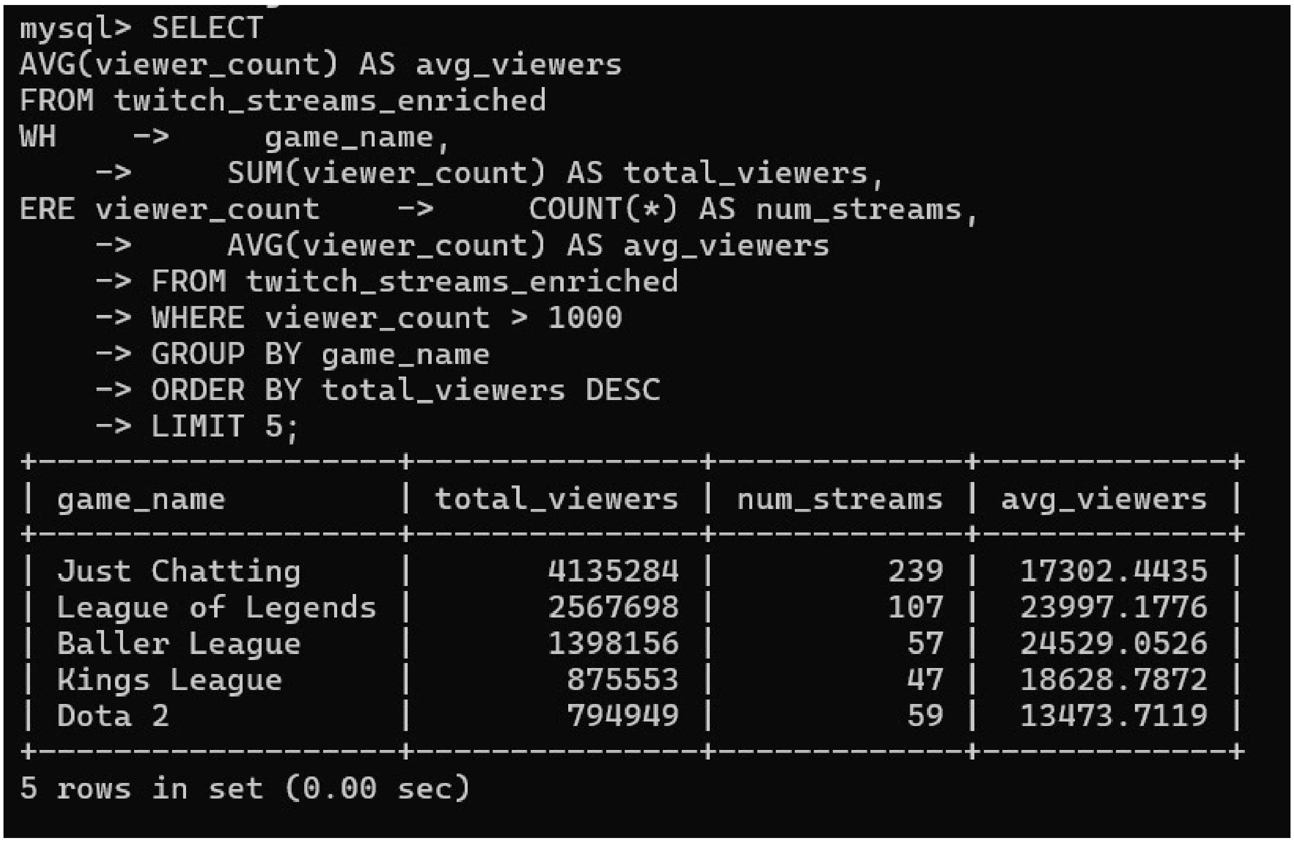
You might notice the top streamers not changing much while the other two categories (viewrship per game and popularity buckets) having a noticeable difference in values.

 This lets us infer that those certain streamers have a very high retention rate towards their viewers and for prolonged periods of time (as most of the streamers listed in the most viewed table are from streams with high uptimes).

This is a very valuable metric for companies who want to display ads to the widest audience possible with a maximum return on their investment.

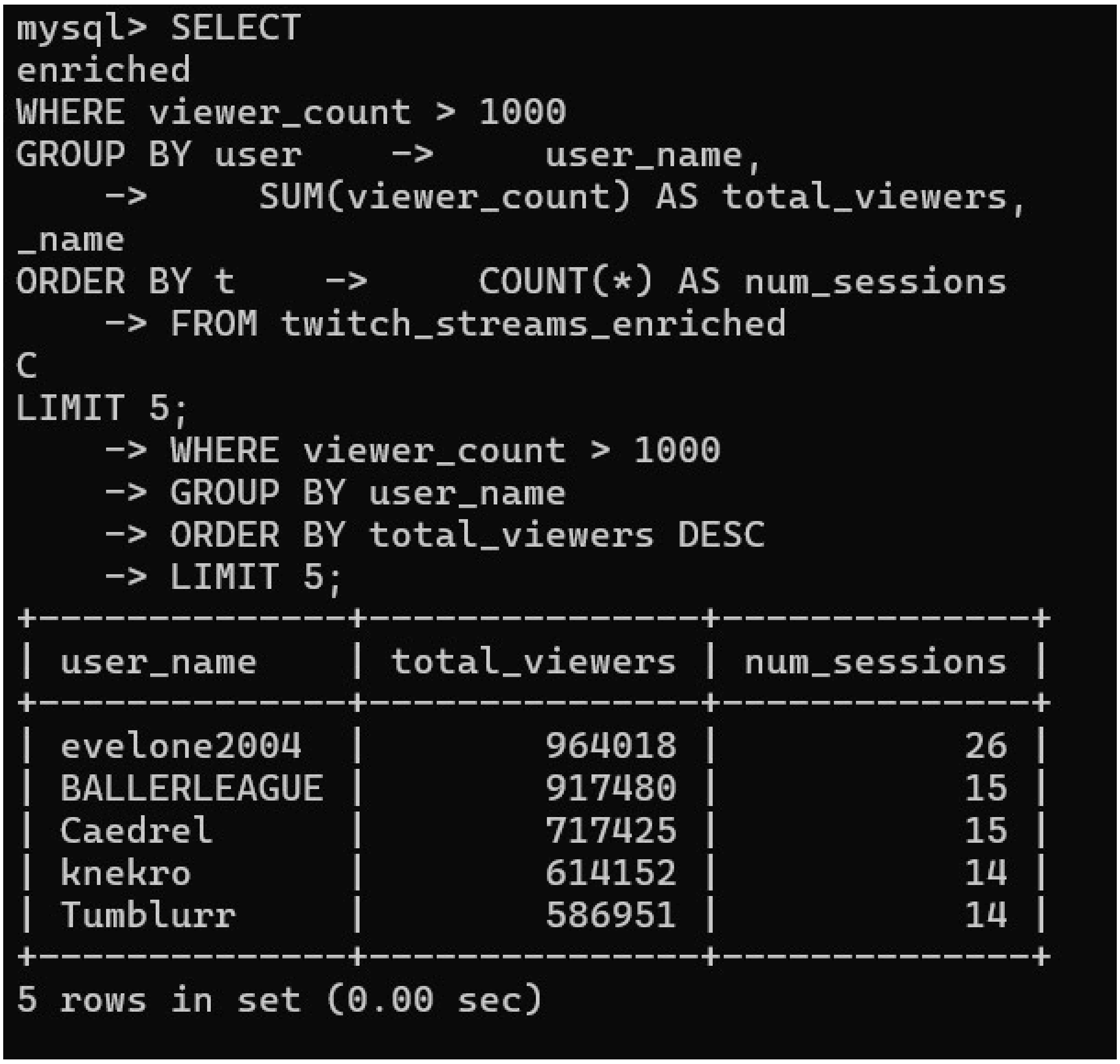
2) Batch Processing on the entire dataset :

Top Games by Total Viewers



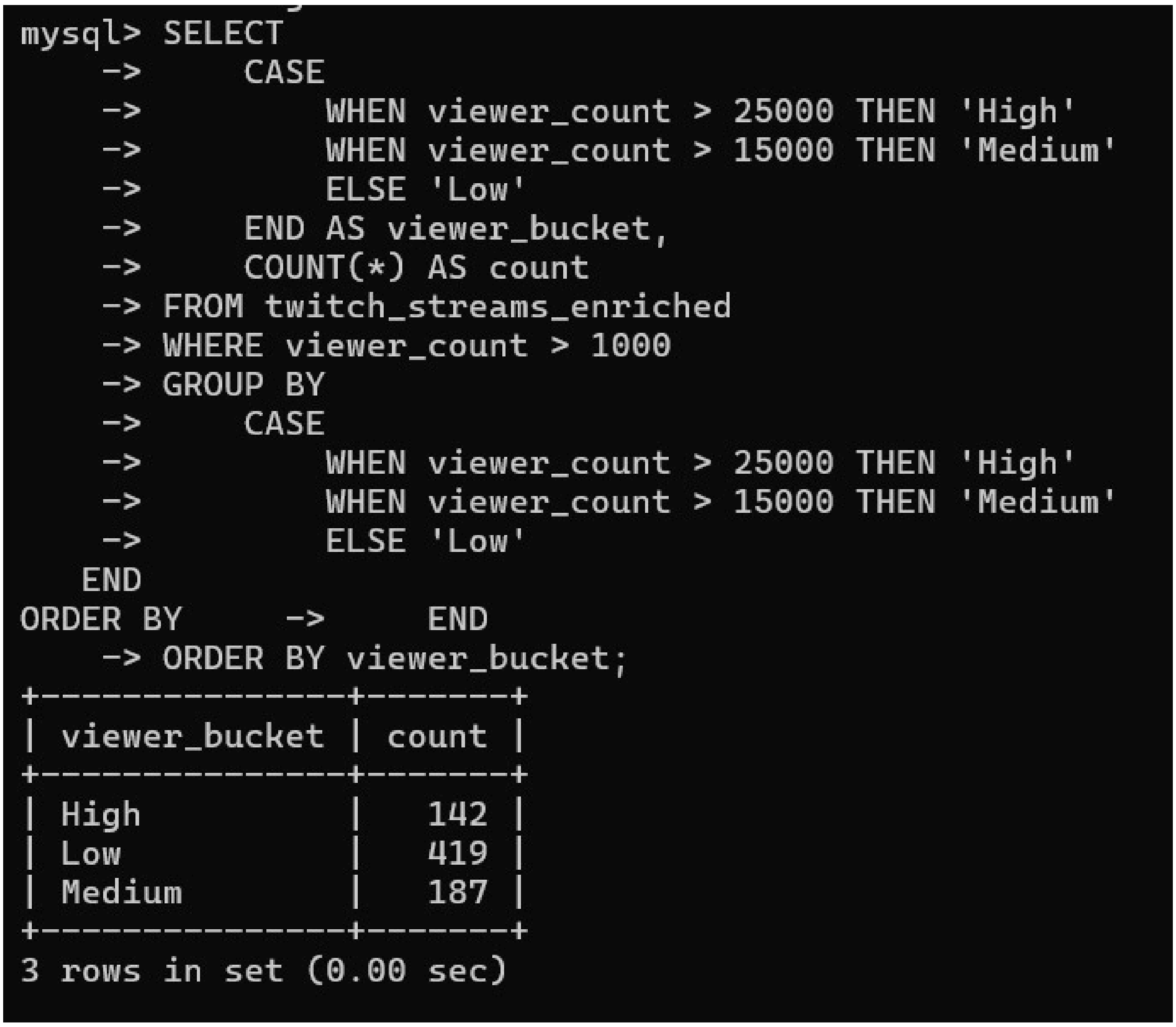
```
mysql> SELECT
AVG(viewer_count) AS avg_viewers
FROM twitch_streams_enriched
WH    ->     game_name,
    ->     SUM(viewer_count) AS total_viewers,
ERE viewer_count    ->     COUNT(*) AS num_streams,
    ->     AVG(viewer_count) AS avg_viewers
    -> FROM twitch_streams_enriched
    -> WHERE viewer_count > 1000
    -> GROUP BY game_name
    -> ORDER BY total_viewers DESC
    -> LIMIT 5;
+-------------------+---------------+-------------+-------------+
| game_name         | total_viewers | num_streams | avg_viewers |
+-------------------+---------------+-------------+-------------+
| Just Chatting     |       4135284 |         239 |  17302.4435 |
| League of Legends |       2567698 |         107 |  23997.1776 |
| Baller League     |       1398156 |          57 |  24529.0526 |
| Kings League      |        875553 |          47 |  18628.7872 |
| Dota 2            |        794949 |          59 |  13473.7119 |
+-------------------+---------------+-------------+-------------+
5 rows in set (0.00 sec)
```

Top Streamers by Total Viewers



```
mysql> SELECT
enriched
WHERE viewer_count > 1000
GROUP BY user    ->     user_name,
    ->     SUM(viewer_count) AS total_viewers,
_name
ORDER BY t    ->     COUNT(*) AS num_sessions
    -> FROM twitch_streams_enriched
C
LIMIT 5;
    -> WHERE viewer_count > 1000
    -> GROUP BY user_name
    -> ORDER BY total_viewers DESC
    -> LIMIT 5;
+-------------+---------------+--------------+
| user_name   | total_viewers | num_sessions |
+-------------+---------------+--------------+
| evelone2004 |        964018 |           26 |
| BALLERLEAGUE|        917480 |           15 |
| Caedrel     |        717425 |           15 |
| knekro      |        614152 |           14 |
| Tumblurr    |        586951 |           14 |
+-------------+---------------+--------------+
5 rows in set (0.00 sec)
```

Stream Count by Popularity Bucket



```
mysql> SELECT
    ->     CASE
    ->         WHEN viewer_count > 25000 THEN 'High'
    ->         WHEN viewer_count > 15000 THEN 'Medium'
    ->         ELSE 'Low'
    ->     END AS viewer_bucket,
    ->     COUNT(*) AS count
    -> FROM twitch_streams_enriched
    -> WHERE viewer_count > 1000
    -> GROUP BY
    ->     CASE
    ->         WHEN viewer_count > 25000 THEN 'High'
    ->         WHEN viewer_count > 15000 THEN 'Medium'
    ->         ELSE 'Low'
   END
ORDER BY     ->     END
    -> ORDER BY viewer_bucket;
+---------------+-------+
| viewer_bucket | count |
+---------------+-------+
| High          |   142 |
| Low           |   419 |
| Medium        |   187 |
+---------------+-------+
3 rows in set (0.00 sec)
```

Comparison of Streaming & Batch Modes

Results and Discussion

The batch processing is likely faster (as seen from the screenshots) for the following reasons:

- Database Optimization: MySQL is highly optimized for query operations like GROUP BY, SUM, and ORDER BY. These operations can be faster in a mature database system compared to Spark's in-memory processing, especially for moderate data volumes.
- Query Execution: In the batch processing implementation, the SQL queries are run on data that's already been stored in MySQL. This leverages MySQL's indexing and query optimization capabilities, while Spark's in-memory processing doesn't have these optimizations for smaller datasets.
- Resource Allocation: The stream processing is doing all computation within Spark, which might be configured with limited resources in your development environment, while MySQL might be better optimized for your specific hardware.
- Processing Overhead: Stream processing typically includes additional overhead for maintaining the streaming context and state management, which can slow down processing compared to batch operations.
- Data Consistency: Batch processing can leverage more efficient bulk operations since it's working with a complete dataset at once, rather than continuously processing incoming data streams.
- Execution Model : Batch processing can fully utilize parallelization across the entire dataset Streaming processing often has sequential dependencies that limit parallelization

Conclusion

Streaming mode is best suited for real-time use cases where latency is critical, such as monitoring live stream popularity or triggering alerts when viewership spikes. It provides immediate insights but has limitations in processing depth due to time constraints.

Batch mode, on the other hand, excels at processing large datasets with deeper transformations and aggregations. It's ideal for retrospective analytics, periodic reporting, and generating business intelligence insights.

In this project, both modes complement each other—streaming for immediacy, batch for depth—demonstrating how hybrid architectures can be powerful in real-world data systems.

References

- Apache Kafka
  - Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A distributed messaging system for log processing. Proceedings of the NetDB, 11, 1-7.
  - Apache Kafka Documentation. https://kafka.apache.org/documentation/
- Apache Spark
  - Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. HotCloud, 10(10-10), 95.
- Spark Structured Streaming Documentation.
  - https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html
- MySQL
  - Schwartz, B., Zaitsev, P., & Tkachenko, V. (2012). High Performance MySQL: Optimization, Backups, and Replication. O'Reilly Media.
  - MySQL Documentation. https://dev.mysql.com/doc/
- Python

  - McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media.
  - Python Documentation. https://docs.python.org/3/
- Streaming Data Architecture
  - Kreps, J. (2014). I Heart Logs: Event Data, Stream Processing, and Data Integration. O'Reilly Media.
  - Kleppmann, M. (2017). Designing Data-Intensive Applications. O'Reilly Media.
- Real-time Analytics
  - Psaltis, A. (2017). Streaming Data: Understanding the Real-Time Pipeline. Manning Publications.
  - Dunning, T., & Friedman, E. (2014). Time Series Databases: New Ways to Store and Access Data. O'Reilly Media.
- Twitch API Documentation. https://dev.twitch.tv/docs/api/
  - Wang, R., & Zhu, K. (2016). How Application Programming Interfaces Enable Digital Innovation—The Case of Twitch. International Conference on Information Systems.
- Data Transformation Techniques
  - White, T. (2015). Hadoop: The Definitive Guide. O'Reilly Media.
  - Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). Learning Spark: Lightning-Fast Big Data Analysis. O'Reilly Media.
- Stream Classification Methods
  - Bifet, A., & Kirkby, R. (2009). Data stream mining: A practical approach. The University of Waikato.
- Performance Evaluation
  - Shukla, A., & Simmhan, Y. (2018). Benchmarking distributed stream processing platforms for IoT applications. Performance Evaluation, 127, 1-26.
- Big Data Processing
  - Mayer-Schönberger, V., & Cukier, K. (2013). Big Data: A Revolution That Will Transform How We Live, Work, and Think. Houghton Mifflin Harcourt.
  - Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.