Ministry of Higher Education and Research
Higher School of Computer Science 08 May 1945 - Sidi Bel Abbes

Second Year Second Cycle - Artificial Intelligence and Data Science

LAB 07 : STREAM PROCESSING

Presented By : FELLAH Abdelnour.

Date: May 12, 2024

1 INTRODUCTION

Stream processing is a powerful paradigm for handling real-time data streams, enabling organizations to derive valuable insights and make informed decisions. In this lab, we explore the use of ksqlDB, a streaming SQL engine for Apache Kafka, to perform stream processing tasks.

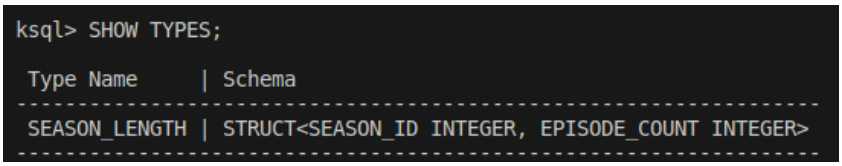
2 SETUP

```
# docker-compose up
# docker exec -it ksqldb-cli ksql http://ksqldb-server:8088
```

3 PART 01 : KSQLDB QUERIES

QUESTION 01 :

```
ksql> CREATE TYPE season_length AS STRUCT<season_id INT, episode_count INT>;
ksql> SHOW TYPES;
```



QUESTION 02 :

The appropriate collection type for titles and production_changes :

- titles : tables, they are mutable.
- production_changes : stream because they are immutable events.

QUESTION 03 :

```
ksql> CREATE STREAM productionChanges (
        rowkey VARCHAR KEY,title_id INT, change_type VARCHAR,
        before SEASON_LENGTH,after SEASON_LENGTH,created_at VARCHAR)
  WITH (
        kafka_topic='production_changes',
        value_format='json',
        partitions=4,
        TIMESTAMP='created_at',
        TIMESTAMP_FORMAT='yyyy-MM-dd HH:mm:ss'
```

```
  );

ksql> CREATE TABLE title (id INT PRIMARY KEY,title VARCHAR)
  WITH (KAFKA_TOPIC='titles',PARTITIONS='4',VALUE_FORMAT='JSON');
```

```
ksql> LIST STREAMS;

 Stream Name                   | Kafka Topic                   | Key Format | Value Format | Windowed
------------------------------------------------------------------------------------------------------
 KSQL_PROCESSING_LOG           | default_ksql_processing_log   | KAFKA      | JSON         | false
 PRODUCTIONCHANGES             | production_changes            | KAFKA      | JSON         | false
 SEASON_LENGTH_CHANGES         | season_length_changes         | KAFKA      | JSON         | false
 SEASON_LENGTH_CHANGES_ENRICHED | season_length_changes_enriched | KAFKA     | JSON         | false
------------------------------------------------------------------------------------------------------
ksql> LIST TABLES;

 Table Name                 | Kafka Topic                 | Key Format | Value Format | Windowed
------------------------------------------------------------------------------------------------
 SEASON_LENGTH_CHANGE_COUNTS | season_length_change_counts | KAFKA     | JSON         | true
 TITLE                      | titles                      | KAFKA      | JSON         | false
------------------------------------------------------------------------------------------------
ksql>
```

Question 04 :

```
ksql> INSERT INTO title (id, title) VALUES (1,'title1');
ksql> INSERT INTO productionChanges (rowkey, title_id,change_type,before,after,created_at)
  VALUES ('rowkey1',1,'season_length',STRUCT(season_id := 1,episode_count := 12),
  STRUCT(season_id := 1,episode_count := 8),'2021-02-08 11:30:00');
```

```
ksql> SELECT * FROM QUERYABLE_TITLE;
+------------------------------------------------+------------------------------------------------+
|ID                                              |TITLE                                           |
+------------------------------------------------+------------------------------------------------+
|1                                               |title1                                          |
Query terminated
ksql> SELECT * FROM productionChanges;
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|ROWKEY              |TITLE_ID            |CHANGE_TYPE         |BEFORE              |AFTER               |CREATED_AT          |
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|rowkey1             |1                   |season_length       |{SEASON_ID=1, EPISODE_CO|{SEASON_ID=1, EPISODE_CO|2021-02-08 11:30:00 |
|                    |                    |                    |UNT=12}             |UNT=8}              |                    |
|rowkey1             |1                   |season_length       |{SEASON_ID=1, EPISODE_CO|{SEASON_ID=1, EPISODE_CO|2021-02-08 11:30:00 |
|                    |                    |                    |UNT=12}             |UNT=8}              |                    |
Query Completed
Query terminated
ksql>
```

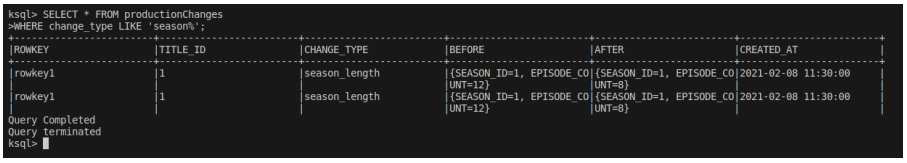
Question 05 :

```
ksql> SET 'auto.offset.reset' = 'earliest';
ksql> SELECT * FROM productionChanges
  WHERE created_at < '2023-04-14 à 12:00:00'
  EMIT CHANGES;
```

```
ksql> SELECT * FROM productionChanges
>WHERE created_at < '2023-04-14 à 12:00:00';
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|ROWKEY              |TITLE_ID            |CHANGE_TYPE         |BEFORE              |AFTER               |CREATED_AT          |
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|rowkey1             |1                   |season_length       |{SEASON_ID=1, EPISODE_CO|{SEASON_ID=1, EPISODE_CO|2021-02-08 11:30:00 |
|                    |                    |                    |UNT=12}             |UNT=8}              |                    |
|rowkey1             |1                   |season_length       |{SEASON_ID=1, EPISODE_CO|{SEASON_ID=1, EPISODE_CO|2021-02-08 11:30:00 |
|                    |                    |                    |UNT=12}             |UNT=8}              |                    |
Query Completed
Query terminated
ksql>
```

QUESTION 06 :

```
ksql> SELECT * FROM productionChanges
  WHERE change_type LIKE 'season%'
  EMIT CHANGES;
```



QUESTION 07 :

```
ksql> CREATE STREAM season_length_changes
  WITH (KAFKA_TOPIC='season_length_changes',PARTITIONS='4',VALUE_FORMAT='JSON',REPLICAS = '1')
  AS
        SELECT rowkey,title_id,created_at,
        IFNULL(after->season_id, before->season_id) as season_id,
        before->episode_count as old_episode_count,
        after->episode_count as new_episode_count
        FROM productionChanges
        WHERE change_type = 'season_length';
```

QUESTION 08 :

```
ksql> SELECT title
        FROM season_length_changes s
        INNER JOIN title t
        ON CAST(s.title_id AS INT) = t.id
        EMIT CHANGES ;
```

QUESTION 09 :

```
ksql> CREATE STREAM season_length_changes_enriched
  WITH (
        KAFKA_TOPIC='season_length_changes_enriched',
        PARTITIONS='4',
        VALUE_FORMAT='JSON',
        REPLICAS = '1',
        TIMESTAMP='created_at',
        TIMESTAMP_FORMAT='yyyy-MM-dd HH:mm:ss'
  ) AS
        SELECT rowkey,title_id,created_at,old_episode_count,new_episode_count,title,id
        FROM season_length_changes s
        INNER JOIN title t
        ON CAST(s.title_id AS INT) = t.id;
```

QUESTION 10 :

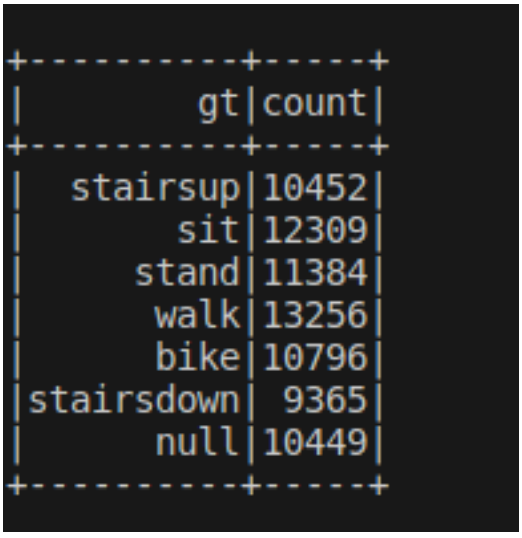
```
ksql> CREATE TABLE season_length_change_counts
  WITH (
        KAFKA_TOPIC='season_length_change_counts',
        PARTITIONS='4',
        VALUE_FORMAT='JSON',
        REPLICAS = '1'
  ) AS
        SELECT title_id,COUNT(*) AS change_count,LATEST_BY_OFFSET(new_episode_count) AS latest_episode_count
        FROM season_length_changes_enriched
        WINDOW TUMBLING (SIZE 1 HOUR)
        GROUP BY title_id;
```

4 PART 02 : STREAM PROCESSING WITH SPARK

PROGRAM :

```
from pyspark.sql import SparkSession
from time import sleep

spark = SparkSession.builder.appName("Chapitre4").getOrCreate()
static = spark.read.json("./data")

dataSchema = static.schema
static.printSchema()

streaming = spark.readStream.schema(dataSchema).option("maxFilesPerTrigger", 1).json("./da

activityCounts = streaming.groupBy("gt").count()

activityQuery = activityCounts.writeStream.queryName("activity_counts") \
   .format("memory").outputMode("complete") \
   .start()

spark.streams.active

for x in range(5):
   spark.sql("SELECT * FROM activity_counts").show()
   sleep(1)
```

OUTPUT :



```
+----------+-----+
|        gt|count|
+----------+-----+
|  stairsup|10452|
|       sit|12309|
|     stand|11384|
|      walk|13256|
|      bike|10796|
|stairsdown| 9365|
|      null|10449|
+----------+-----+
```

5 CONCLUSION

In this lab, we have explored stream processing using ksqlDB and Apache Kafka. We have learned how to set up a streaming data pipeline, write streaming SQL queries with ksqlDB, Through this process, we have gained valuable insights into the capabilities of stream processing and its potential applications.