Programming Assignment № 2

Evangelia Panourgia, Athens University of Economics and Business University 27/03/2025

Part I

The python script for the first part.

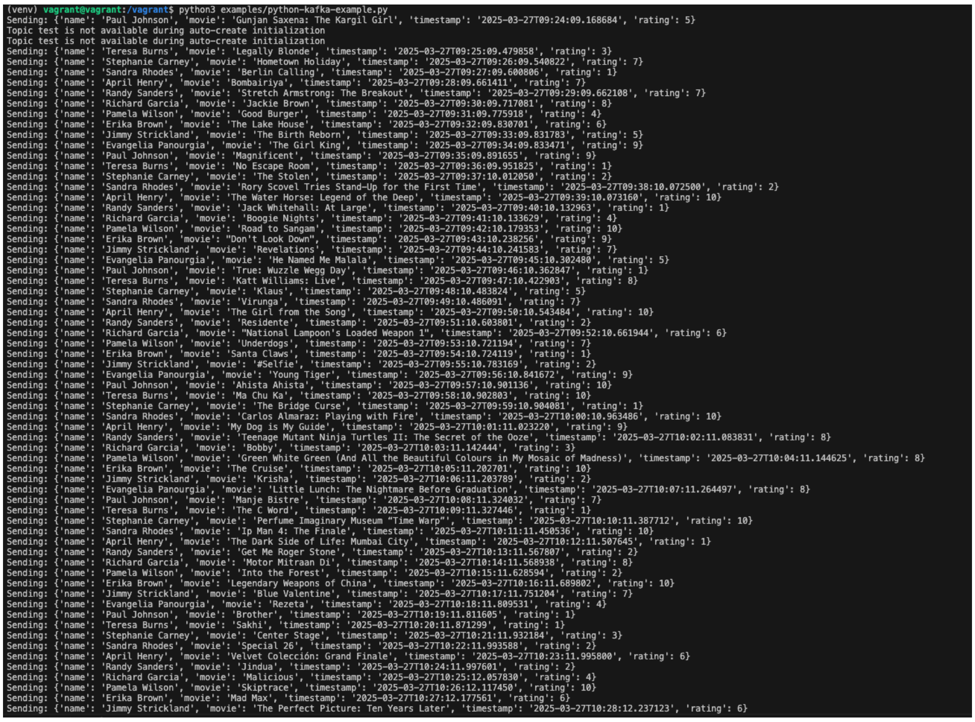
The provided Python script is an asynchronous Kafka producer designed to simulate real-time movie rating events. It uses the `Faker` library to generate synthetic user names and reads actual movie titles from a CSV file using `pandas`. Each generated user rates a randomly selected movie with a random rating between 1 and 10, along with a current timestamp. These events are serialized to JSON and sent to a Kafka topic named `"test"` using the `aiokafka` library. The process runs continuously, with each user producing one rating per minute, making it ideal for streaming applications such as those built with Apache Spark Structured Streaming. Additionally, the script appends a specific user name, “Evangelia Panourgia”, to ensure consistent personal data inclusion for testing or filtering.

Listing 1: python-kafka-example

```
"""
Async Kafka Producer: Fake Movie Ratings Generator

Overview:
---------
This script continuously generates **fake movie rating events** and sends them to a Kafka topic ("test").
It simulates users rating random movies with timestamps, mimicking real-time data.

Components:
-----------
1. Faker: Generates fake human names.
2. Pandas: Loads real movie titles from a CSV file.
3. aiokafka: Asynchronous Kafka producer library.
4. Kafka: Acts as the message broker for event streaming.

Use Case:
---------
Run this script to simulate live rating data for a Spark Structured Streaming application to consume and process.
"""
```

```python
import json
import asyncio
import random
from datetime import datetime

import pandas as pd
from faker import Faker
from aiokafka import AIOKafkaProducer

# Load movie titles
movies_df = pd.read_csv(’data/movies.csv’, header=None, names=["movie_title"])
movie_titles = movies_df[’movie_title’].dropna().tolist()

# Generate names
fake = Faker()
names = [fake.name() for _ in range(10)]
names.append("Evangelia Panourgia") # add my own name

# Kafka settings
KAFKA_TOPIC = "test"
KAFKA_BOOTSTRAP_SERVERS = "localhost:29092"

# JSON serializer
def serializer(data):
    return json.dumps(data).encode()

# Producer logic
async def produce():
    producer = AIOKafkaProducer(
        bootstrap_servers=KAFKA_BOOTSTRAP_SERVERS,
        value_serializer=serializer,
        compression_type="gzip"
    )

    await producer.start()
    try:
        while True:
            for name in names:
                movie = random.choice(movie_titles)
                rating = random.randint(1, 10)
                timestamp = datetime.now().isoformat()

                message = {
                    "name": name,
                    "movie": movie,
                    "timestamp": timestamp,
                    "rating": rating
                }

                print(f"Sending: {message}")
                await producer.send(KAFKA_TOPIC, message)
                await asyncio.sleep(60) # One message per user per minute
    finally:
        await producer.stop()

# Run it (no deprecation warning)
if __name__ == "__main__":
    loop = asyncio.new_event_loop()
    asyncio.set_event_loop(loop)
    loop.run_until_complete(produce())
```

The image below depicts the output of kafka prodicer runing in the first terminal.

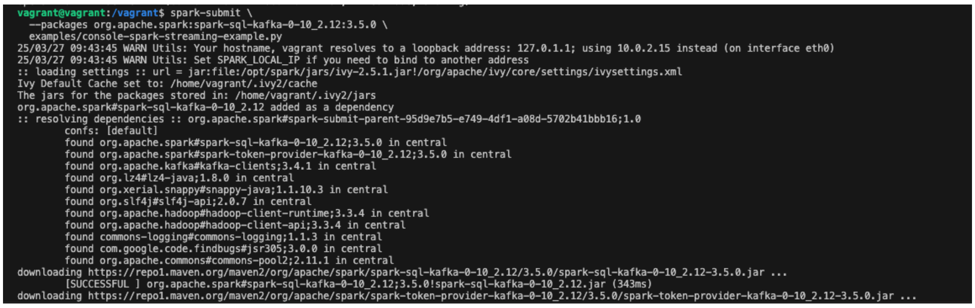


```
(venv) vagrant@vagrant:/vagrant$ python3 examples/python-kafka-example.py
Sending: {'name': 'Paul Johnson', 'movie': 'Gunjan Saxena: The Kargil Girl', 'timestamp': '2025-03-27T09:24:09.168684', 'rating': 5}
Topic test is not available during auto-create initialization
Topic test is not available during auto-create initialization
Sending: {'name': 'Teresa Burns', 'movie': 'Legally Blonde', 'timestamp': '2025-03-27T09:25:09.479858', 'rating': 3}
Sending: {'name': 'Stephanie Carney', 'movie': 'Hometown Holiday', 'timestamp': '2025-03-27T09:26:09.540822', 'rating': 7}
Sending: {'name': 'Sandra Rhodes', 'movie': 'Berlin Calling', 'timestamp': '2025-03-27T09:27:09.600806', 'rating': 1}
Sending: {'name': 'April Henry', 'movie': 'Bombairiya', 'timestamp': '2025-03-27T09:28:09.661411', 'rating': 7}
Sending: {'name': 'Randy Sanders', 'movie': 'Stretch Armstrong: The Breakout', 'timestamp': '2025-03-27T09:29:09.662108', 'rating': 7}
Sending: {'name': 'Richard Garcia', 'movie': 'Jackie Brown', 'timestamp': '2025-03-27T09:30:09.717081', 'rating': 8}
Sending: {'name': 'Pamela Wilson', 'movie': 'Good Burger', 'timestamp': '2025-03-27T09:31:09.775918', 'rating': 4}
Sending: {'name': 'Erika Brown', 'movie': 'The Lake House', 'timestamp': '2025-03-27T09:32:09.830701', 'rating': 6}
Sending: {'name': 'Jimmy Strickland', 'movie': 'The Birth Reborn', 'timestamp': '2025-03-27T09:33:09.831783', 'rating': 5}
Sending: {'name': 'Evangelia Panourgia', 'movie': 'The Girl King', 'timestamp': '2025-03-27T09:34:09.833471', 'rating': 9}
Sending: {'name': 'Paul Johnson', 'movie': 'Magnificent', 'timestamp': '2025-03-27T09:35:09.891655', 'rating': 9}
Sending: {'name': 'Teresa Burns', 'movie': 'No Escape Room', 'timestamp': '2025-03-27T09:36:09.951825', 'rating': 1}
Sending: {'name': 'Stephanie Carney', 'movie': 'The Stolen', 'timestamp': '2025-03-27T09:37:10.012050', 'rating': 2}
Sending: {'name': 'Sandra Rhodes', 'movie': 'Rory Scovel Tries Stand-Up for the First Time', 'timestamp': '2025-03-27T09:38:10.072500', 'rating': 2}
Sending: {'name': 'April Henry', 'movie': 'The Water Horse: Legend of the Deep', 'timestamp': '2025-03-27T09:39:10.073160', 'rating': 10}
Sending: {'name': 'Randy Sanders', 'movie': 'Jack Whitehall: At Large', 'timestamp': '2025-03-27T09:40:10.132963', 'rating': 1}
Sending: {'name': 'Richard Garcia', 'movie': 'Boogie Nights', 'timestamp': '2025-03-27T09:41:10.133629', 'rating': 4}
Sending: {'name': 'Pamela Wilson', 'movie': 'Road to Sangam', 'timestamp': '2025-03-27T09:42:10.179353', 'rating': 10}
Sending: {'name': 'Erika Brown', 'movie': "Don't Look Down", 'timestamp': '2025-03-27T09:43:10.238256', 'rating': 9}
Sending: {'name': 'Jimmy Strickland', 'movie': 'Revelations', 'timestamp': '2025-03-27T09:44:10.241583', 'rating': 7}
Sending: {'name': 'Evangelia Panourgia', 'movie': 'He Named Me Malala', 'timestamp': '2025-03-27T09:45:10.302480', 'rating': 5}
Sending: {'name': 'Paul Johnson', 'movie': 'True: Wuzzle Wegg Day', 'timestamp': '2025-03-27T09:46:10.362847', 'rating': 1}
Sending: {'name': 'Teresa Burns', 'movie': 'Katt Williams: Live', 'timestamp': '2025-03-27T09:47:10.422903', 'rating': 8}
Sending: {'name': 'Stephanie Carney', 'movie': 'Klaus', 'timestamp': '2025-03-27T09:48:10.483824', 'rating': 5}
Sending: {'name': 'Sandra Rhodes', 'movie': 'Virunga', 'timestamp': '2025-03-27T09:49:10.486091', 'rating': 7}
Sending: {'name': 'April Henry', 'movie': 'The Girl from the Song', 'timestamp': '2025-03-27T09:50:10.543484', 'rating': 10}
Sending: {'name': 'Randy Sanders', 'movie': 'Residente', 'timestamp': '2025-03-27T09:51:10.603801', 'rating': 2}
Sending: {'name': 'Richard Garcia', 'movie': "National Lampoon's Loaded Weapon 1", 'timestamp': '2025-03-27T09:52:10.661944', 'rating': 6}
Sending: {'name': 'Pamela Wilson', 'movie': 'Underdogs', 'timestamp': '2025-03-27T09:53:10.721194', 'rating': 7}
Sending: {'name': 'Erika Brown', 'movie': 'Santa Claws', 'timestamp': '2025-03-27T09:54:10.724119', 'rating': 1}
Sending: {'name': 'Jimmy Strickland', 'movie': '#Selfie', 'timestamp': '2025-03-27T09:55:10.783169', 'rating': 2}
Sending: {'name': 'Evangelia Panourgia', 'movie': 'Young Tiger', 'timestamp': '2025-03-27T09:56:10.841672', 'rating': 9}
Sending: {'name': 'Paul Johnson', 'movie': 'Ahista Ahista', 'timestamp': '2025-03-27T09:57:10.901136', 'rating': 10}
Sending: {'name': 'Teresa Burns', 'movie': 'Ma Chu Ka', 'timestamp': '2025-03-27T09:58:10.902803', 'rating': 10}
Sending: {'name': 'Stephanie Carney', 'movie': 'The Bridge Curse', 'timestamp': '2025-03-27T09:59:10.904081', 'rating': 1}
Sending: {'name': 'Sandra Rhodes', 'movie': 'Carlos Almaraz: Playing with Fire', 'timestamp': '2025-03-27T10:00:10.963486', 'rating': 10}
Sending: {'name': 'April Henry', 'movie': 'My Dog is My Guide', 'timestamp': '2025-03-27T10:01:11.023220', 'rating': 9}
Sending: {'name': 'Randy Sanders', 'movie': 'Teenage Mutant Ninja Turtles II: The Secret of the Ooze', 'timestamp': '2025-03-27T10:02:11.083831', 'rating': 8}
Sending: {'name': 'Richard Garcia', 'movie': 'Bobby', 'timestamp': '2025-03-27T10:03:11.142444', 'rating': 3}
Sending: {'name': 'Pamela Wilson', 'movie': 'Green White Green (And All the Beautiful Colours in My Mosaic of Madness)', 'timestamp': '2025-03-27T10:04:11.144625', 'rating': 8}
Sending: {'name': 'Erika Brown', 'movie': 'The Cruise', 'timestamp': '2025-03-27T10:05:11.202701', 'rating': 10}
Sending: {'name': 'Jimmy Strickland', 'movie': 'Krisha', 'timestamp': '2025-03-27T10:06:11.203789', 'rating': 2}
Sending: {'name': 'Evangelia Panourgia', 'movie': 'Little Lunch: The Nightmare Before Graduation', 'timestamp': '2025-03-27T10:07:11.264497', 'rating': 8}
Sending: {'name': 'Paul Johnson', 'movie': 'Manje Bistre', 'timestamp': '2025-03-27T10:08:11.324032', 'rating': 7}
Sending: {'name': 'Teresa Burns', 'movie': 'The C Word', 'timestamp': '2025-03-27T10:09:11.327446', 'rating': 1}
Sending: {'name': 'Stephanie Carney', 'movie': 'Perfume Imaginary Museum “Time Warp”', 'timestamp': '2025-03-27T10:10:11.387712', 'rating': 10}
Sending: {'name': 'Sandra Rhodes', 'movie': 'Ip Man 4: The Finale', 'timestamp': '2025-03-27T10:11:11.450536', 'rating': 10}
Sending: {'name': 'April Henry', 'movie': 'The Dark Side of Life: Mumbai City', 'timestamp': '2025-03-27T10:12:11.507645', 'rating': 1}
Sending: {'name': 'Randy Sanders', 'movie': 'Get Me Roger Stone', 'timestamp': '2025-03-27T10:13:11.567807', 'rating': 2}
Sending: {'name': 'Richard Garcia', 'movie': 'Motor Mitraan Di', 'timestamp': '2025-03-27T10:14:11.568938', 'rating': 8}
Sending: {'name': 'Pamela Wilson', 'movie': 'Into the Forest', 'timestamp': '2025-03-27T10:15:11.628594', 'rating': 2}
Sending: {'name': 'Erika Brown', 'movie': 'Legendary Weapons of China', 'timestamp': '2025-03-27T10:16:11.689802', 'rating': 10}
Sending: {'name': 'Jimmy Strickland', 'movie': 'Blue Valentine', 'timestamp': '2025-03-27T10:17:11.751204', 'rating': 7}
Sending: {'name': 'Evangelia Panourgia', 'movie': 'Rezeta', 'timestamp': '2025-03-27T10:18:11.809531', 'rating': 4}
Sending: {'name': 'Paul Johnson', 'movie': 'Brother', 'timestamp': '2025-03-27T10:19:11.811605', 'rating': 1}
Sending: {'name': 'Teresa Burns', 'movie': 'Sakhi', 'timestamp': '2025-03-27T10:20:11.871299', 'rating': 1}
Sending: {'name': 'Stephanie Carney', 'movie': 'Center Stage', 'timestamp': '2025-03-27T10:21:11.932184', 'rating': 3}
Sending: {'name': 'Sandra Rhodes', 'movie': 'Special 26', 'timestamp': '2025-03-27T10:22:11.993588', 'rating': 2}
Sending: {'name': 'April Henry', 'movie': 'Velvet Colección: Grand Finale', 'timestamp': '2025-03-27T10:23:11.995800', 'rating': 6}
Sending: {'name': 'Randy Sanders', 'movie': 'Jindua', 'timestamp': '2025-03-27T10:24:11.997601', 'rating': 2}
Sending: {'name': 'Richard Garcia', 'movie': 'Malicious', 'timestamp': '2025-03-27T10:25:12.057830', 'rating': 4}
Sending: {'name': 'Pamela Wilson', 'movie': 'Skiptrace', 'timestamp': '2025-03-27T10:26:12.117450', 'rating': 10}
Sending: {'name': 'Erika Brown', 'movie': 'Mad Max', 'timestamp': '2025-03-27T10:27:12.177561', 'rating': 6}
Sending: {'name': 'Jimmy Strickland', 'movie': 'The Perfect Picture: Ten Years Later', 'timestamp': '2025-03-27T10:28:12.237123', 'rating': 6}
```

Figure 1: Terminal 1: Kafka Producer.

An additional modification was applied to the console script located in the `examples` directory, specifically to the file named `console-spark-streaming-example.py`. This adjustment was necessary to tailor the script to the project's requirements and ensure compatibility with the streaming architecture.

The image below depicts the console output helping us to debug the kafka producer real-time data. It runs in another terminal.



```
vagrant@vagrant:/vagrant$ spark-submit \
  --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.0 \
  examples/console-spark-streaming-example.py
25/03/27 09:43:45 WARN Utils: Your hostname, vagrant resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface eth0)
25/03/27 09:43:45 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
:: loading settings :: url = jar:file:/opt/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/vagrant/.ivy2/cache
The jars for the packages stored in: /home/vagrant/.ivy2/jars
org.apache.spark#spark-sql-kafka-0-10_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-95d9e7b5-e749-4df1-a08d-5702b41bbb16;1.0
        confs: [default]
        found org.apache.spark#spark-sql-kafka-0-10_2.12;3.5.0 in central
        found org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.5.0 in central
        found org.apache.kafka#kafka-clients;3.4.1 in central
        found org.lz4#lz4-java;1.8.0 in central
        found org.xerial.snappy#snappy-java;1.1.10.3 in central
        found org.slf4j#slf4j-api;2.0.7 in central
        found org.apache.hadoop#hadoop-client-runtime;3.3.4 in central
        found org.apache.hadoop#hadoop-client-api;3.3.4 in central
        found commons-logging#commons-logging;1.1.3 in central
        found com.google.code.findbugs#jsr305;3.0.0 in central
        found org.apache.commons#commons-pool2;2.11.1 in central
downloading https://repo1.maven.org/maven2/org/apache/spark/spark-sql-kafka-0-10_2.12/3.5.0/spark-sql-kafka-0-10_2.12-3.5.0.jar ...
        [SUCCESSFUL ] org.apache.spark#spark-sql-kafka-0-10_2.12;3.5.0!spark-sql-kafka-0-10_2.12.jar (343ms)
downloading https://repo1.maven.org/maven2/org/apache/spark/spark-token-provider-kafka-0-10_2.12/3.5.0/spark-token-provider-kafka-0-10_2.12-3.5.0.jar ...
```

Figure 2: Terminal 2: Console for Kafka Producer.

Part 2

The pyspark script for the second part.

This PySpark application establishes a real-time data pipeline that reads movie rating events from a Kafka topic, enriches them with metadata from a static Netflix dataset, and writes the processed results into a Cassandra database. The script defines two schemas: one for parsing JSON-encoded rating messages received from Kafka, and another for reading static metadata from a CSV file. It converts timestamps to proper `TimestampType`, and groups data by hourly buckets for time-based aggregations. Ratings are joined with metadata on the movie title to create an enriched DataFrame, which is then written to Cassandra using the `foreachBatch` method in micro-batch mode. This architecture enables structured streaming analysis and time-windowed queries for individual users and movies.

Listing 2: cassandra-spark-streaming-example.py.

```python
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, TimestampType
from pyspark.sql.functions import from_json, col, to_timestamp, date_format

# Schema for Kafka messages (movie ratings)
ratingSchema = StructType([
    StructField("name", StringType(), False),
    StructField("movie", StringType(), False),
    StructField("timestamp", StringType(), False),
    StructField("rating", IntegerType(), False)
])

# Schema for Netflix CSV
netflixSchema = StructType([
    StructField("show_id", StringType(), True),
    StructField("title", StringType(), False),
    StructField("director", StringType(), True),
    StructField("country", StringType(), True),
    StructField("release_year", StringType(), True),
    StructField("rating", StringType(), True),
    StructField("duration", StringType(), True)
])

# Initialize Spark session
spark = (
    SparkSession.builder
    .appName("MovieRatingStreamer")
    .config("spark.jars.packages", "org.apache.spark:spark-sql-kafka-0-10↩
       _2.12:3.5.0,com.datastax.spark:spark-cassandra-connector_2↩
       .12:3.4.1")
    .config("spark.cassandra.connection.host", "localhost")
    .getOrCreate()
)

spark.sparkContext.setLogLevel("ERROR")

# Read Netflix CSV with renamed rating column to avoid conflict with "↩
   rating" in Kafka schema
netflix_df = (
    spark.read.schema(netflixSchema)
         .option("header", True)
         .csv("data/netflix.csv")
         .withColumnRenamed("rating", "rating_category")  # Rename avoids ↩
            conflict
         .cache()
)

# Read streaming data from Kafka
df = (
    spark.readStream.format("kafka")
         .option("kafka.bootstrap.servers", "localhost:29092")
         .option("subscribe", "test")
         .option("startingOffsets", "latest")
         .load()
)

# Parse JSON from Kafka messages
ratings_df = (
    df.selectExpr("CAST(value AS STRING)")
      .select(from_json(col("value"), ratingSchema).alias("data"))
      .select("data.*")
      .withColumn("timestamp", to_timestamp("timestamp"))  # Convert ↩
         string timestamp to TimestampType
      .withColumn("hour_bucket", date_format("timestamp", "yyyy-MM-dd HH↩
         :00"))  # Grouping by hour
)

# Join ratings with static Netflix metadata
enriched_df = (
    ratings_df.join(netflix_df, ratings_df.movie == netflix_df.title, "left")
              .drop("title")
)

# Optional: Print schema for debugging (can be removed later)
enriched_df.printSchema()

# Define Cassandra write logic
def writeToCassandra(writeDF, _):
    (
        writeDF.select(
            "name", "movie", "timestamp", "rating", "hour_bucket",
            "show_id", "director", "country", "release_year", "rating_category", "duration"
        )
        .write
        .format("org.apache.spark.sql.cassandra")
        .mode('append')
        .options(table="movie_ratings", keyspace="netflix_ks")
        .save()
    )

# Write to Cassandra in a streaming loop
result = None
while result is None:
    try:
        result = (
            enriched_df.writeStream
                .foreachBatch(writeToCassandra)
                .outputMode("update")
                .option("checkpointLocation", "/tmp/checkpoints/movie")
                .trigger(processingTime="30 seconds")
                .start()
                .awaitTermination()
        )
    except Exception as e:
        print(f"Streaming error: {e}")
```

Details about your Cassandra data model.

The Cassandra data model was carefully designed to optimize query performance for the specific requirement of aggregating movie ratings by user and hour. The primary key is composed of a composite partition key `(name, hour_bucket)`, ensuring that all ratings provided by a specific user within a given hour are stored in the same partition. This design eliminates the need for inefficient filtering operations and allows for fast, targeted queries such as retrieving all movies rated by a user during a specific hour or calculating the average rating and duration of those movies. The clustering columns `(timestamp, movie)` further organize the data chronologically within each partition, enabling efficient range queries and ordered retrieval. Crucially, the fields `rating` and `duration` are stored as integers, allowing native support for aggregation functions like `AVG()`, which would not be possible with text fields. Overall, this schema supports the core analytical tasks required by the application while adhering to Cassandra's best practices for scalability and performance.

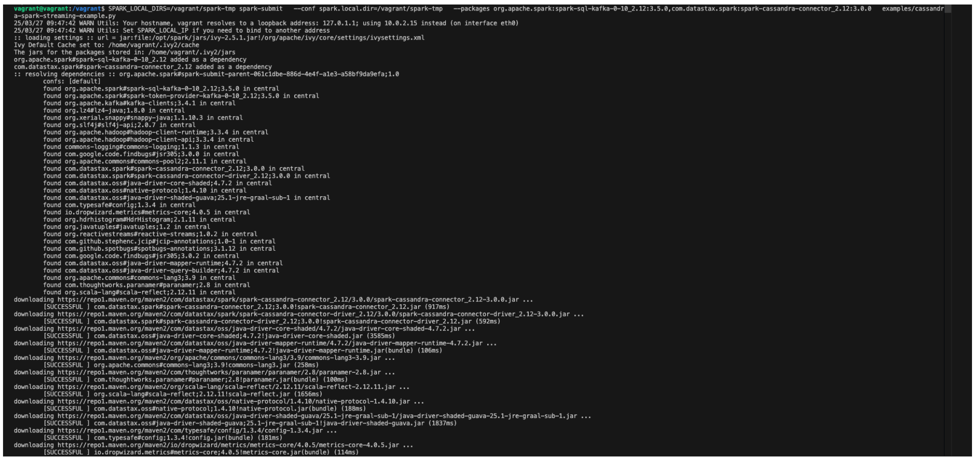
Listing 3: Cassandra schema for movie ratings

```sql
CREATE KEYSPACE IF NOT EXISTS netflix_ks
WITH replication = {’class’: ’SimpleStrategy’, ’replication_factor’: 1};

CREATE TABLE IF NOT EXISTS netflix_ks.movie_ratings (
    name text,
    hour_bucket text,
    timestamp timestamp,
    movie text,
    rating int,
    show_id text,
    director text,
    country text,
    release_year int,         -- changed from text : int
    rating_category text,
    duration int,             -- changed from text : int
    PRIMARY KEY ((name, hour_bucket), timestamp, movie)
);
```

A sample of persisted lines (around 50) of your Cassandra table.

To inspect the data stored in the `movie_ratings` table of the `netflix_ks` keyspace, we can use the `cqlsh` shell interface provided by Apache Cassandra. The following query retrieves approximately 50 rows that were previously persisted by the streaming pipeline:

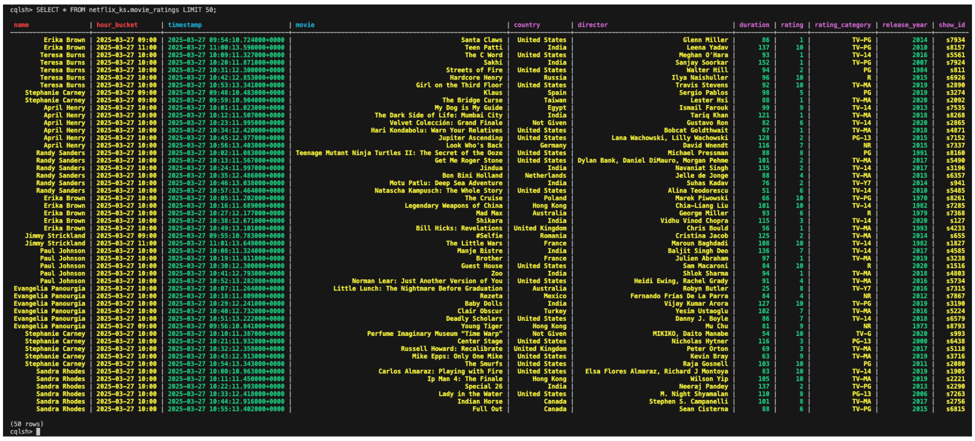
```sql
SELECT * FROM netflix_ks.movie_ratings LIMIT 50;
```



```
vagrant@vagrant:/vagrant$ SPARK_LOCAL_DIRS=/vagrant/spark-tmp spark-submit   --conf spark.local.dir=/vagrant/spark-tmp   --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.0,com.datastax.spark:spark-cassandra-connector_2.12:3.0.0   examples/cassandr
a-spark-streaming-example.py
25/03/27 09:47:42 WARN Utils: Your hostname, vagrant resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface eth0)
25/03/27 09:47:42 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
:: loading settings :: url = jar:file:/opt/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/vagrant/.ivy2/cache
The jars for the packages stored in: /home/vagrant/.ivy2/jars
org.apache.spark#spark-sql-kafka-0-10_2.12 added as a dependency
com.datastax.spark#spark-cassandra-connector_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-061c1dbe-886d-4e4f-a1e3-a58bf9da9efa;1.0
        confs: [default]
        found org.apache.spark#spark-sql-kafka-0-10_2.12;3.5.0 in central
        found org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.5.0 in central
        found org.apache.kafka#kafka-clients;3.4.1 in central
        found org.lz4#lz4-java;1.8.0 in central
        found org.xerial.snappy#snappy-java;1.1.10.3 in central
        found org.slf4j#slf4j-api;2.0.7 in central
        found org.apache.hadoop#hadoop-client-runtime;3.3.4 in central
        found org.apache.hadoop#hadoop-client-api;3.3.4 in central
        found commons-logging#commons-logging;1.1.3 in central
        found com.google.code.findbugs#jsr305;3.0.0 in central
        found org.apache.commons#commons-pool2;2.11.1 in central
        found com.datastax.spark#spark-cassandra-connector_2.12;3.0.0 in central
        found com.datastax.spark#spark-cassandra-connector-driver_2.12;3.0.0 in central
        found com.datastax.oss#java-driver-core-shaded;4.7.2 in central
        found com.datastax.oss#native-protocol;1.4.10 in central
        found com.datastax.oss#java-driver-shaded-guava;25.1-jre-graal-sub-1 in central
        found com.typesafe#config;1.3.4 in central
        found io.dropwizard.metrics#metrics-core;4.0.5 in central
        found org.hdrhistogram#HdrHistogram;2.1.11 in central
        found org.javatuples#javatuples;1.2 in central
        found org.reactivestreams#reactive-streams;1.0.2 in central
        found com.github.stephenc.jcip#jcip-annotations;1.0-1 in central
        found com.github.spotbugs#spotbugs-annotations;3.1.12 in central
        found com.google.code.findbugs#jsr305;3.0.2 in central
        found com.datastax.oss#java-driver-mapper-runtime;4.7.2 in central
        found com.datastax.oss#java-driver-query-builder;4.7.2 in central
        found org.apache.commons#commons-lang3;3.9 in central
        found com.thoughtworks.paranamer#paranamer;2.8 in central
        found org.scala-lang#scala-reflect;2.12.11 in central
downloading https://repo1.maven.org/maven2/com/datastax/spark/spark-cassandra-connector_2.12/3.0.0/spark-cassandra-connector_2.12-3.0.0.jar ...
        [SUCCESSFUL ] com.datastax.spark#spark-cassandra-connector_2.12;3.0.0!spark-cassandra-connector_2.12.jar (917ms)
downloading https://repo1.maven.org/maven2/com/datastax/spark/spark-cassandra-connector-driver_2.12/3.0.0/spark-cassandra-connector-driver_2.12-3.0.0.jar ...
        [SUCCESSFUL ] com.datastax.spark#spark-cassandra-connector-driver_2.12;3.0.0!spark-cassandra-connector-driver_2.12.jar (592ms)
downloading https://repo1.maven.org/maven2/com/datastax/oss/java-driver-core-shaded/4.7.2/java-driver-core-shaded-4.7.2.jar ...
        [SUCCESSFUL ] com.datastax.oss#java-driver-core-shaded;4.7.2!java-driver-core-shaded.jar (3585ms)
downloading https://repo1.maven.org/maven2/com/datastax/oss/java-driver-mapper-runtime/4.7.2/java-driver-mapper-runtime-4.7.2.jar ...
        [SUCCESSFUL ] com.datastax.oss#java-driver-mapper-runtime;4.7.2!java-driver-mapper-runtime.jar(bundle) (106ms)
downloading https://repo1.maven.org/maven2/org/apache/commons/commons-lang3/3.9/commons-lang3-3.9.jar ...
        [SUCCESSFUL ] org.apache.commons#commons-lang3;3.9!commons-lang3.jar (258ms)
downloading https://repo1.maven.org/maven2/com/thoughtworks/paranamer/paranamer/2.8/paranamer-2.8.jar ...
        [SUCCESSFUL ] com.thoughtworks.paranamer#paranamer;2.8!paranamer.jar(bundle) (100ms)
downloading https://repo1.maven.org/maven2/org/scala-lang/scala-reflect/2.12.11/scala-reflect-2.12.11.jar ...
        [SUCCESSFUL ] org.scala-lang#scala-reflect;2.12.11!scala-reflect.jar (1656ms)
downloading https://repo1.maven.org/maven2/com/datastax/oss/native-protocol/1.4.10/native-protocol-1.4.10.jar ...
        [SUCCESSFUL ] com.datastax.oss#native-protocol;1.4.10!native-protocol.jar(bundle) (188ms)
downloading https://repo1.maven.org/maven2/com/datastax/oss/java-driver-shaded-guava/25.1-jre-graal-sub-1/java-driver-shaded-guava-25.1-jre-graal-sub-1.jar ...
        [SUCCESSFUL ] com.datastax.oss#java-driver-shaded-guava;25.1-jre-graal-sub-1!java-driver-shaded-guava.jar (1837ms)
downloading https://repo1.maven.org/maven2/com/typesafe/config/1.3.4/config-1.3.4.jar ...
        [SUCCESSFUL ] com.typesafe#config;1.3.4!config.jar(bundle) (181ms)
downloading https://repo1.maven.org/maven2/io/dropwizard/metrics/metrics-core/4.0.5/metrics-core-4.0.5.jar ...
        [SUCCESSFUL ] io.dropwizard.metrics#metrics-core;4.0.5!metrics-core.jar(bundle) (114ms)
```

Figure 3: Terminal 3: Spark pre-process, enrich with metadata..

This query provides a representative snapshot of the data ingested into Cassandra, including key fields such as `name`, `movie`, `timestamp`, `rating`, and `duration`. Analyzing these records ensures data integrity, validates the correctness of the ingestion pipeline, and confirms that the schema design supports the desired querying capabilities.

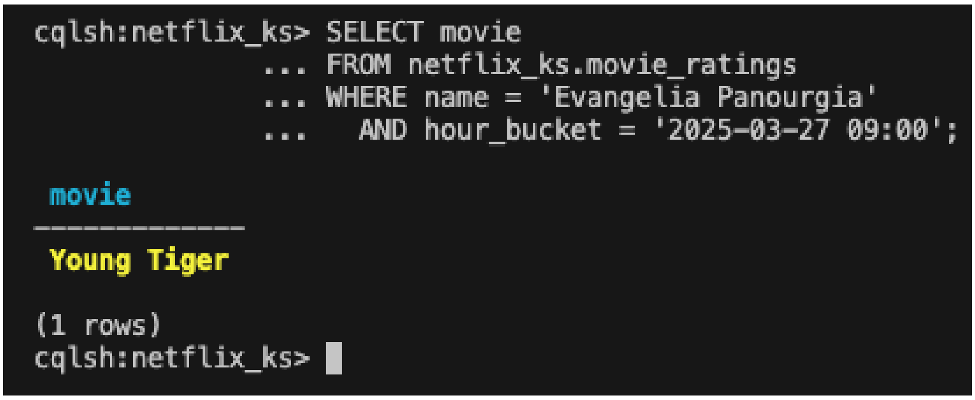


```
cqlsh> SELECT * FROM netflix_ks.movie_ratings LIMIT 50;

 name                | hour_bucket      | timestamp                       | movie                                                   | country        | director                                 | duration | rating | rating_category | release_year | show_id
---------------------+------------------+---------------------------------+---------------------------------------------------------+----------------+------------------------------------------+----------+--------+-----------------+--------------+---------
         Erika Brown | 2025-03-27 09:00 | 2025-03-27 09:54:10.724000+0000 |                                             Santa Claws |  United States |                             Glenn Miller |       86 |      1 |           TV-PG |         2014 |   s7934
         Erika Brown | 2025-03-27 11:00 | 2025-03-27 11:00:13.590000+0000 |                                              Teen Patti |          India |                              Leena Yadav |      137 |     10 |           TV-PG |         2010 |   s8157
        Teresa Burns | 2025-03-27 10:00 | 2025-03-27 10:09:11.327000+0000 |                                              The C Word |  United States |                            Meghan O'Hara |       93 |      1 |           TV-14 |         2016 |   s5561
        Teresa Burns | 2025-03-27 10:00 | 2025-03-27 10:20:11.871000+0000 |                                                   Sakhi |          India |                           Sanjay Soorkar |      152 |      1 |           TV-PG |         2007 |   s7924
        Teresa Burns | 2025-03-27 10:00 | 2025-03-27 10:31:12.300000+0000 |                                         Streets of Fire |  United States |                              Walter Hill |       94 |      2 |              PG |         1984 |    s811
        Teresa Burns | 2025-03-27 10:00 | 2025-03-27 10:42:12.853000+0000 |                                          Hardcore Henry |         Russia |                          Ilya Naishuller |       96 |     10 |               R |         2015 |   s6926
        Teresa Burns | 2025-03-27 10:00 | 2025-03-27 10:53:13.341000+0000 |                                 Girl on the Third Floor |  United States |                           Travis Stevens |       92 |     10 |           TV-MA |         2019 |   s2890
    Stephanie Carney | 2025-03-27 09:00 | 2025-03-27 09:48:10.483000+0000 |                                                   Klaus |          Spain |                            Sergio Pablos |       98 |      5 |              PG |         2019 |   s3274
    Stephanie Carney | 2025-03-27 09:00 | 2025-03-27 09:59:10.904000+0000 |                                        The Bridge Curse |         Taiwan |                               Lester Hsi |       88 |      1 |           TV-MA |         2020 |   s2092
         April Henry | 2025-03-27 10:00 | 2025-03-27 10:01:11.023000+0000 |                                      My Dog is My Guide |          Egypt |                            Ismail Farouk |       99 |      9 |           TV-14 |         2013 |   s7535
         April Henry | 2025-03-27 10:00 | 2025-03-27 10:12:11.507000+0000 |                      The Dark Side of Life: Mumbai City |          India |                               Tariq Khan |      121 |      1 |           TV-MA |         2018 |   s8268
         April Henry | 2025-03-27 10:00 | 2025-03-27 10:23:11.995000+0000 |                          Velvet Colección: Grand Finale |      Not Given |                              Gustavo Ron |       82 |      6 |           TV-14 |         2020 |   s2865
         April Henry | 2025-03-27 10:00 | 2025-03-27 10:34:12.420000+0000 |                     Hari Kondabolu: Warn Your Relatives |  United States |                        Bobcat Goldthwait |       67 |      1 |           TV-MA |         2018 |   s4871
         April Henry | 2025-03-27 10:00 | 2025-03-27 10:45:12.977000+0000 |                                       Jupiter Ascending |  United States |          Lana Wachowski, Lilly Wachowski |      128 |      2 |           PG-13 |         2015 |   s7152
         April Henry | 2025-03-27 10:00 | 2025-03-27 10:56:13.403000+0000 |                                         Look Who's Back |        Germany |                             David Wnendt |      116 |      7 |              NR |         2015 |   s7337
       Randy Sanders | 2025-03-27 10:00 | 2025-03-27 10:02:11.083000+0000 | Teenage Mutant Ninja Turtles II: The Secret of the Ooze |  United States |                         Michael Pressman |       88 |      8 |              PG |         1991 |   s8160
       Randy Sanders | 2025-03-27 10:00 | 2025-03-27 10:13:11.567000+0000 |                                      Get Me Roger Stone |  United States | Dylan Bank, Daniel DiMauro, Morgan Pehme |      101 |      2 |           TV-MA |         2017 |   s5490
       Randy Sanders | 2025-03-27 10:00 | 2025-03-27 10:24:11.997000+0000 |                                                  Jindua |          India |                           Navaniat Singh |      135 |      2 |           TV-14 |         2017 |   s3196
       Randy Sanders | 2025-03-27 10:00 | 2025-03-27 10:35:12.486000+0000 |                                        Bon Bini Holland |    Netherlands |                           Jelle de Jonge |       88 |      4 |           TV-MA |         2015 |   s6357
       Randy Sanders | 2025-03-27 10:00 | 2025-03-27 10:46:13.038000+0000 |                          Motu Patlu: Deep Sea Adventure |          India |                              Suhas Kadav |       76 |      2 |           TV-Y7 |         2014 |    s941
       Randy Sanders | 2025-03-27 10:00 | 2025-03-27 10:57:13.464000+0000 |                      Natascha Kampusch: The Whole Story |  United States |                         Alina Teodorescu |       51 |      6 |           TV-14 |         2010 |   s5485
         Erika Brown | 2025-03-27 10:00 | 2025-03-27 10:05:11.202000+0000 |                                              The Cruise |         Poland |                           Marek Piwowski |       66 |     10 |           TV-PG |         1970 |   s8261
         Erika Brown | 2025-03-27 10:00 | 2025-03-27 10:16:11.689000+0000 |                              Legendary Weapons of China |      Hong Kong |                           Chia-Liang Liu |      101 |     10 |           TV-14 |         1982 |   s7285
         Erika Brown | 2025-03-27 10:00 | 2025-03-27 10:27:12.177000+0000 |                                                 Mad Max |      Australia |                            George Miller |       93 |      6 |               R |         1979 |   s7368
         Erika Brown | 2025-03-27 10:00 | 2025-03-27 10:38:12.671000+0000 |                                                 Shikara |          India |                       Vidhu Vinod Chopra |      115 |      3 |           TV-14 |         2020 |    s127
         Erika Brown | 2025-03-27 10:00 | 2025-03-27 10:49:13.101000+0000 |                                 Bill Hicks: Revelations | United Kingdom |                              Chris Bould |       56 |      1 |           TV-MA |         1993 |   s4233
    Jimmy Strickland | 2025-03-27 09:00 | 2025-03-27 09:55:10.783000+0000 |                                                 #Selfie |        Romania |                           Cristina Jacob |      125 |      2 |           TV-MA |         2014 |    s655
    Jimmy Strickland | 2025-03-27 11:00 | 2025-03-27 11:01:13.649000+0000 |                                         The Little Wars |         France |                          Maroun Baghdadi |      108 |     10 |           TV-14 |         1982 |   s1827
        Paul Johnson | 2025-03-27 10:00 | 2025-03-27 10:08:11.324000+0000 |                                            Manje Bistre |          India |                         Baljit Singh Deo |      136 |      7 |           TV-14 |         2017 |   s4585
        Paul Johnson | 2025-03-27 10:00 | 2025-03-27 10:19:11.811000+0000 |                                                 Brother |         France |                           Julien Abraham |       97 |      1 |           TV-MA |         2019 |   s3238
        Paul Johnson | 2025-03-27 10:00 | 2025-03-27 10:30:12.300000+0000 |                                             Guest House |  United States |                             Sam Macaroni |       84 |     10 |               R |         2020 |   s1516
        Paul Johnson | 2025-03-27 10:00 | 2025-03-27 10:41:12.793000+0000 |                                                     Zoo |          India |                             Shlok Sharma |       94 |      1 |           TV-MA |         2018 |   s4803
        Paul Johnson | 2025-03-27 10:00 | 2025-03-27 10:52:13.282000+0000 |                 Norman Lear: Just Another Version of You |  United States |                Heidi Ewing, Rachel Grady |       91 |      4 |           TV-MA |         2016 |   s5734
 Evangelia Panourgia | 2025-03-27 10:00 | 2025-03-27 10:07:11.264000+0000 |           Little Lunch: The Nightmare Before Graduation |      Australia |                             Robyn Butler |       25 |      8 |           TV-Y7 |         2016 |   s7315
 Evangelia Panourgia | 2025-03-27 10:00 | 2025-03-27 10:18:11.809000+0000 |                                                  Rezeta |         Mexico |               Fernando Frías De La Parra |       84 |      4 |              NR |         2012 |   s7867
 Evangelia Panourgia | 2025-03-27 10:00 | 2025-03-27 10:29:12.241000+0000 |                                              Baby Dolls |          India |                        Vijay Kumar Arora |      127 |     10 |           TV-PG |         2019 |   s3190
 Evangelia Panourgia | 2025-03-27 10:00 | 2025-03-27 10:40:12.732000+0000 |                                            Clair Obscur |         Turkey |                           Yesim Ustaoglu |      102 |      7 |           TV-MA |         2016 |   s5224
 Evangelia Panourgia | 2025-03-27 10:00 | 2025-03-27 10:51:13.222000+0000 |                                         Deadly Scholars |  United States |                           Danny J. Boyle |       86 |      7 |           TV-14 |         2018 |   s6579
 Evangelia Panourgia | 2025-03-27 09:00 | 2025-03-27 09:56:10.841000+0000 |                                             Young Tiger |      Hong Kong |                                   Mu Chu |       81 |      9 |              NR |         1973 |   s8793
    Stephanie Carney | 2025-03-27 10:00 | 2025-03-27 10:10:11.387000+0000 |                     Perfume Imaginary Museum “Time Warp” |      Not Given |                     MIKIKO, Daito Manabe |       54 |     10 |            TV-G |         2020 |    s993
    Stephanie Carney | 2025-03-27 10:00 | 2025-03-27 10:21:11.932000+0000 |                                            Center Stage |  United States |                          Nicholas Hytner |      116 |      3 |           PG-13 |         2000 |   s6438
    Stephanie Carney | 2025-03-27 10:00 | 2025-03-27 10:32:12.358000+0000 |                             Russell Howard: Recalibrate | United Kingdom |                              Peter Orton |       69 |      3 |           TV-MA |         2017 |   s5118
    Stephanie Carney | 2025-03-27 10:00 | 2025-03-27 10:43:12.913000+0000 |                                Mike Epps: Only One Mike |  United States |                               Kevin Bray |       63 |      9 |           TV-MA |         2019 |   s3716
    Stephanie Carney | 2025-03-27 10:00 | 2025-03-27 10:54:13.343000+0000 |                                              The Smurfs |  United States |                             Raja Gosnell |      103 |     10 |              PG |         2011 |   s2080
       Sandra Rhodes | 2025-03-27 10:00 | 2025-03-27 10:00:10.963000+0000 |                       Carlos Almaraz: Playing with Fire |  United States |    Elsa Flores Almaraz, Richard J Montoya |       83 |     10 |           TV-14 |         2019 |   s1905
       Sandra Rhodes | 2025-03-27 10:00 | 2025-03-27 10:11:11.450000+0000 |                                    Ip Man 4: The Finale |      Hong Kong |                               Wilson Yip |      105 |     10 |           TV-MA |         2019 |   s2221
       Sandra Rhodes | 2025-03-27 10:00 | 2025-03-27 10:22:11.993000+0000 |                                              Special 26 |          India |                            Neeraj Pandey |      137 |      2 |           TV-PG |         2013 |   s2290
       Sandra Rhodes | 2025-03-27 10:00 | 2025-03-27 10:33:12.418000+0000 |                                       Lady in the Water |  United States |                       M. Night Shyamalan |      110 |      8 |           PG-13 |         2006 |   s7263
       Sandra Rhodes | 2025-03-27 10:00 | 2025-03-27 10:44:12.916000+0000 |                                            Indian Horse |         Canada |                    Stephen S. Campanelli |      101 |      8 |           TV-MA |         2017 |   s2756
       Sandra Rhodes | 2025-03-27 10:00 | 2025-03-27 10:55:13.402000+0000 |                                                Full Out |         Canada |                            Sean Cisterna |       88 |      6 |           TV-PG |         2015 |   s6815

(50 rows)
cqlsh>
```

Figure 4: A sample output of the Cassandra table with 50 persisted entries.

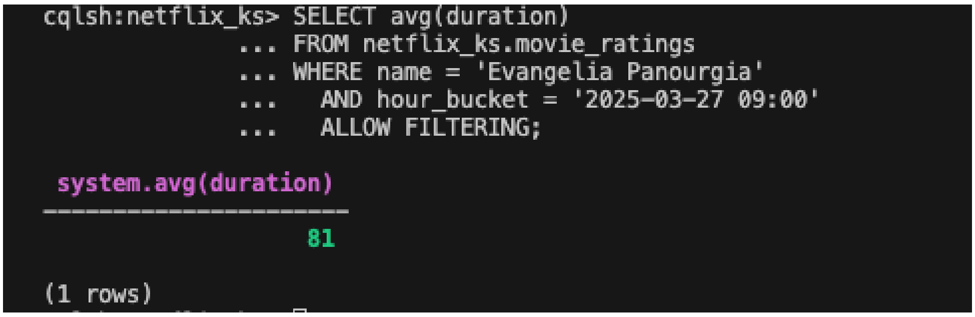
Two CQL queries and their results in your database about your own name and a particular hour that generate the average runtime of the movies that you've rated during this hour, and the names of the movies, respectively.

To support time-based analytics in our Cassandra data model, we designed the schema to partition data by `name` and `hour_bucket`, allowing efficient access to all ratings a specific user made during a given hour. The first query retrieves the titles of movies rated by the user *Evangelia Panourgia* during the hour of `2025-03-27 09:00` using a direct lookup on the partition key. The second query, which includes the `ALLOW FILTERING` clause, calculates the average duration of those rated movies within the same hour. While Cassandra does not natively support aggregations without filtering, our schema minimizes performance issues by narrowing queries to a specific partition, ensuring practical execution even with the filtering clause.



```
cqlsh:netflix_ks> SELECT movie
              ... FROM netflix_ks.movie_ratings
              ... WHERE name = 'Evangelia Panourgia'
              ...   AND hour_bucket = '2025-03-27 09:00';

 movie
-------------
 Young Tiger

(1 rows)
cqlsh:netflix_ks> 
```

Figure 5: Names of the movies you rated during a particular hour SELECT movie.



```
cqlsh:netflix_ks> SELECT avg(duration)
              ... FROM netflix_ks.movie_ratings
              ... WHERE name = 'Evangelia Panourgia'
              ...   AND hour_bucket = '2025-03-27 09:00'
              ...   ALLOW FILTERING;

 system.avg(duration)
----------------------
                   81

(1 rows)
```

Figure 6: Average duration of the movies you rated during the same hour SELECT avg(duration).

Hosted on GitHub Repository - Deploy Instructions.

The complete implementation of this project is hosted on GitHub and is publicly accessible at `https://github.com/e-panourgia/large-data-kafka-cassandra`. The repository contains all necessary components, including Kafka producers, Spark Structured Streaming jobs, and Cassandra schemas. To execute the project, clone the repository using `git clone https://github.com/e-panourgia/large-data-kafka-cassandra`, then follow the instructions in the README file to set up Docker containers, start Kafka and Cassandra services, and run the streaming application using `spark-submit`. Ensure that you have installed Docker, Docker Compose, and Spark locally or are running within the provided Vagrant virtual machine for a consistent and reproducible development environment.

Important note, this implementation was designed to run on mac m4 pc.

Acknowledgements

I would like to sincerely thank **Professor Liakos Panayiotis** for his invaluable guidance, support, and dedication throughout the entire teaching process. His insightful lectures and continuous encouragement greatly contributed to the successful completion of this project.