

Large Scale Data Management
Athens University of Economics and Business
M.Sc. in Data Science

Programming Project II
Today's Date: February 21th, 2025
Due Date: March 28th, 2025

PREAMBLE

In this project, you will use the Apache Spark framework and the Apache Cassandra NoSQL database. The scope is to create a Structured Streaming Spark process that consumes Kafka messages and uses Cassandra as a sink to persist information. You are encouraged to use the attached Vagrantfile that will provide you with Apache Spark (installed in the VM), Apache Kafka (running as a container and accessible through port 29092) and Apache Cassandra (running as a container and accessible through port 9042).

PART I:

In this part you will create using Python and Kafka a stream of movies rated by a group of people. You should create with the use of the Python Faker library a list of names (at least 10) and use a list of movies (see attached file 'movies.csv') to periodically generate a movie for each of them (at most per minute). In addition to the Faker produced names you should also include your own name in the list (hard-coded). Your Python script should produce a Kafka stream with these movies. The information to be included in the stream comprises the name of the person watching the movie, the name of the movie, the current time, and a random movie rating (1-10).

You should consult file 'python-kafka-example.py' that will help you develop your own script.

PART II:

For the second part of this project you will develop a pyspark script that receives, processes and persists the messages produced by the first part of the project. In particular, for each message in the stream you should persist a Cassandra row that comprises the name of the person, the time this person rated the movie, and all the details of the movie as they appear in the 'netflix.csv' file. The persistence should occur at a configurable interval specified through a variable in your script. As a default value you should use 30 seconds.

You should consult file 'cassandra-spark-streaming-example.py' that will help you develop your own script.

When implementing your solution you should take extra care to:

1. Utilize Spark's caching capabilities.
2. Carefully select your Cassandra data model so that you can perform aggregations for a particular person and hour. For example your schema should be optimized for executing a query to find the average runtime (or rating, or the names, etc.) of the movies that a particular person rated during "2025-02-03 10:00".

You are encouraged to also consult file 'console-spark-streaming-example.py' that does not involve Cassandra and will help you test your approach while developing it.

EXECUTION:

You may execute your Python script through the vm produced by the Vagrantfile by executing:

```
$ python3 examples/python-kafka-example.py
```

You may execute your Apache Spark script through the vm produced by the Vagrantfile by executing:

```
$ spark-submit --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/cassandra-spark-streaming-example.py
```

If there are errors when downloading dependencies you should re-execute the script. If Cassandra connection error appears the script is designed to re-attempt a connection, so these errors can most likely be ignored.

Alternatively, you can start a pyspark session by executing:

```
$ pyspark --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
```

Finally, with regards to Cassandra below you can find the necessary commands to create a table:

```
cqlsh> CREATE KEYSPACE spotify WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 1};
cqlsh> create table spotify.records(
id int primary key,
name text,
song text);
```

COOPERATION:

You do not have the option of forming a team in this project. However, discussions on most aspects of the project in the classroom and the classroom's forum are encouraged.

REPORTING:

The final typed project report (brief report) must consist of:

1. The python script for the first part.
2. The pyspark script for the second part.
3. Details about your Cassandra data model.
4. A sample of persisted lines (around 50) of your Cassandra table.
5. 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.

USING DOCKER DIRECTLY

If you have trouble setting up the virtual machine, you could instead use docker¹ directly to set up the necessary utilities.

You should use docker-compose files included in the programming assignment files. Using the docker terminal change directory (cd) to the folder where these files are and execute:

¹<https://docs.docker.com/desktop/setup/install/windows-install/>

```
$ docker-compose -f docker-compose-kafka.yml up -d
$ docker-compose -f docker-compose-cassandra.yml up -d
$ docker-compose -f docker-compose-spark.yml up -d
```

Then, you should use the terminal of the last container to execute spark jobs.