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

*Programming Project II*
Today's Date: February $23^{th}$, 2024
Due Date: March $22^{th}$, 2024

## 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 persit 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 songs listened 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 songs (see attached file 'spotify-songs.csv') to periodically generate a song 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 songs. The information to be included in the stream comprises the name of the person listening to the song, the name of the song, and the current time.

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 listened to the song, and all the details of the song as they appear in the 'spotify-songs.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 danceability (or tempo, or the names, etc.) of the songs that a particular person listened to during "2024-02-03 10:00".

You are encouraged to also consult file 'console-spark-streaming-example.py' that does not involve Cassandra and will help you test you 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 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 danceability of the songs that you've listened during this hour, and the names of the songs, respectively.