**Technical assessment for a Big Data developer — Extended**

**The purpose of the assessment is to test your ability to:**

* Import data to Hive
* Use Spark and Scala efficiently
* Integrate with other systems using Kafka

**What is expected?**

The application you will be writing stores and transforms data about residential mortgages based securities. You will import data to HIVE, do simple transformation on it and save it to a different database. Then you will publish a message to a Kafka consumer to notify that the process is done.

**Step 1:**

Write an application that accepts and file name of a ZIP file that includes a DAT file with Securities information. The application will extract this file and import its contents into a Hive table called security\_raw. The structure of the table can be found here:

[http://embs.com/public/html/PostProcessedFileFmt.htm#Sec](http://embs.com/public/html/PostProcessedFileFmt.htm" \l "Sec)

Please use the attached file: “GNM\_SEC.ZIP-2017-05-08\_23-58-17.ZIP” to test your application.

You can use any programming language to write this application.

**Step 2:**

Write a Spark application using Scala that will read security\_raw and create a new table called security\_normalised. This table should be in ORC format and it will be identical to security\_raw except of the following changes:

1. Drop the columns: PrefixId, InsSrc, UpdSrc
2. Create a new column called: NormalisationDate -> this will be the timestamp of the insertion time of this record to your new table (UTC).
3. Create a new boolean column called: IsNew. If the InsDt column of the record is in the last 12 months then the value will be true, otherwise false.
4. Aggregate the data in the security\_normalised table by Prefix, Product and issue year calculating the minimum, maximum and sum of the issue amount. Store the aggregated data in a table called prefix\_product\_issue\_year.
5. Generate a matrix of the sum of the Issue Amount column by Prefix and Product. Use the values for Prefix as columns and the Product values as the rows in the matrix. Output the result to either a Hive table or text file

e.g.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **AF** | **AR** | **JM** | **RM** | **Total** |
| **GNMIIARM** | 128898922.00 | 269109541.40 | 0.00 | 0.00 | 398008463.40 |
| **GNMII30MJM** | 0.00 | 0.00 | 548253269.00 | 0.00 | 548253269.00 |
| **GNMII15MJM** | 0.00 | 0.00 | 2515477.00 | 0.00 | 2515477.00 |
| **GNMIIREVMTG** | 0.00 | 0.00 | 0.00 | 1120611.00 | 1120611.00 |
| **Total** | 128898922.00 | 269109541.40 | 550768746.00 | 1120611.00 | 949897820.40 |

**Step 3:**

Create a Kafka topic called normalisation\_completed, then amend your Scala application so it sends a message when it finished creating security\_normalised.