**(Healthcare Data Analysis) Documentation**

**Objective:**

I builded a Spark application that will process a daily CSV file from a HDFS folder and perform certain transformations on it, and then store the transformed data in a Cassandra table.

**Tools Used:**

1. Python3 – Microsoft VScode
2. Databricks
3. DataStax Astra (Cassandra DB)

**Files Attached:**

1. Spark\_Ass3.pdf – This file
2. Stage\_healthcare\_analysis– Pyspark File that creates stage tables for daily load.
3. Target\_healthcare\_analysis– Pyspark File that creates final target tables for daily load.

**Process and File Descriptions:**

**Step 1:**

So first created csv dataset using a python mock\_data\_generator file and uploaded that to gcs bucket input folder. Then I created a spark job that takes the daily file from the healthcare\_analysis bucket and input folder. I made sure that there is authentication between databricks and GCP cloud storage, by placing the keys in the dbfs location.

A computer screen with text on it

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step 2:**

I made sure to include data quality checks so that the data is in the correct format.

A screenshot of a computer

Description automatically generated

**Step 3:**

I then went ahead and performed the necessary transformations/queries on it, post which I followed the documentation for Datastax AstraDB to generate a connection between databricks and the Cassandra DB. This involved downloading a ‘secure bundle’ as well as the key/token which was then placed in the DBFS location.

A screenshot of a computer screen

Description automatically generated

**Step 4:**

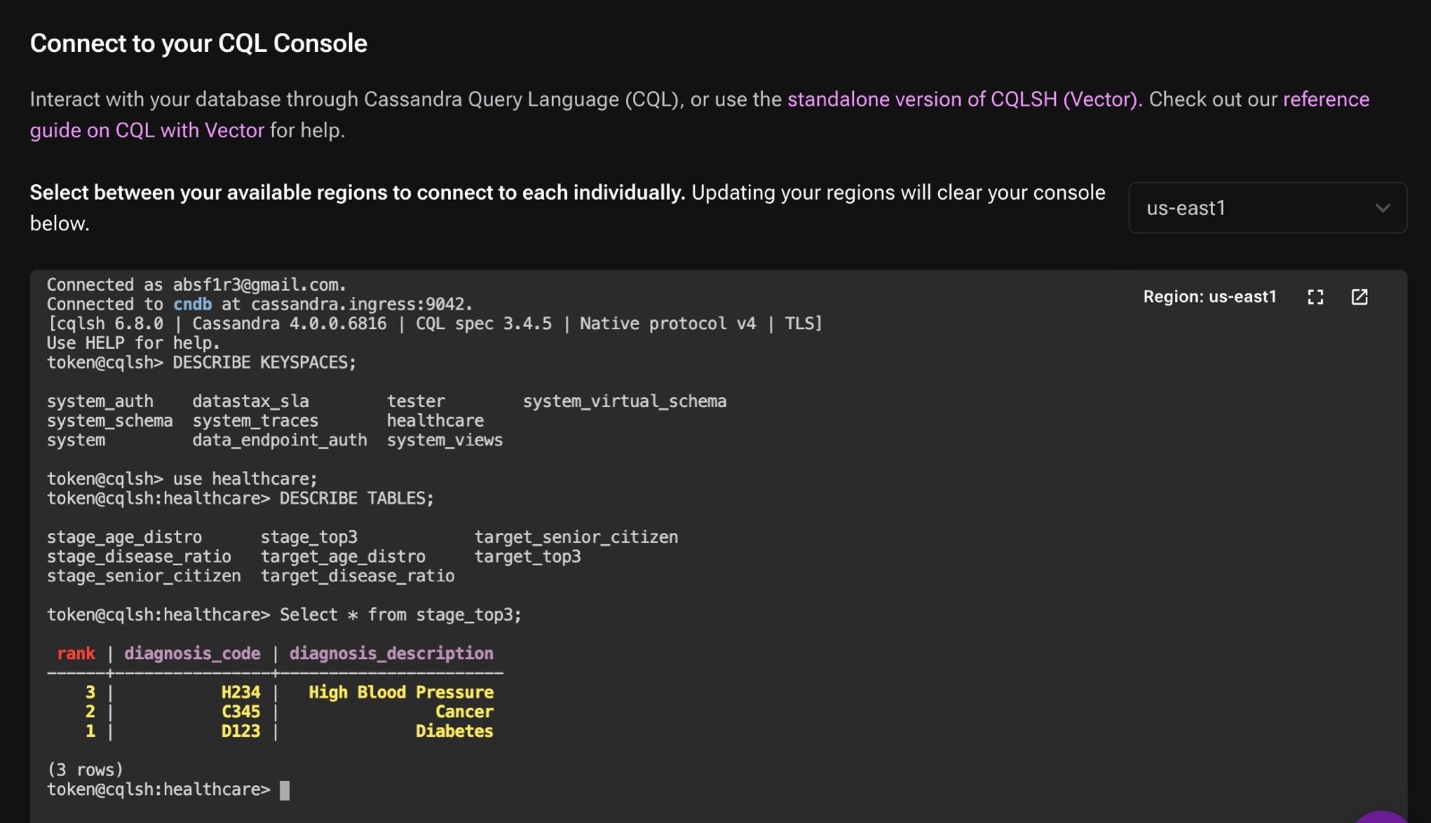
After that using CQL I then checked if there was any table (respective table) in the keyspace. If not I created a new table and then pushed the data into it. If there was an existing table then I truncated all the data and loaded the new data. (This is like forming a daily staging table)

A screen shot of a computer program

Description automatically generated

**Step 5:**

I also checked using the CQL UI from datastax to see if data is present in the tables. I created stage tables for each of the scenarios and also made sure to archive the input files.



**Step 6:**

In another script I created target tables for each of the stage tables where data is inserted in the ‘upsert’ mode. The idea is for the target tables, data is moved from the respective stage table🡪 if no target table exists then a new target table is created and data from the stage table is pushed , else if a target table already exists then upsert is peformed. I made sure to select appropriate keys to match for each of those tables.

A screen shot of a computer screen

Description automatically generated

**Step 7:**

I then created a healthcare\_processing workflow for the two scripts using databricks. The second job is triggered only when the staging process is finished. This way there is a dependency between the two jobs. I also made sure to implement notifications for any failures.  
  
A screenshot of a computer

Description automatically generated

**Challenges:**

1. It was not possible to get the Cassandra-spark connector to work (Lots of time spent on trying to get the correct jar/jdbc drivers) and had to resort to the ‘session\_execute’ method of loading data row by row.