Saavn Trending – MapReduce Project

Overview:

The scope of this project is to implement a MapReduce program to analyze and find the top trending songs list from the log data. Here, the Hadoop MapReduce programming model by Java language is implemented. This helps to achieve processing of big data in better way and get desired output values.

Driver:

1. The class “TrendAnalyzer” is the Driver. It has the Main method. The Main method is the starting point of the program execution. It invokes the run() method and gets the result status. Based on the result status it terminates the program.

2. The run method implements the JOB CONFIGURATION. In this Method all the parameters are set. The following items are initialized here.

   1) Jar Class which acts as the Main(Driver) Class.
   2) Mapper Class and Mapper output Key and Value types – Both types are Text.
   3) Combiner Class.
   4) Reducer Class and Reducer output Key and Value types – Both types are Text.
   5) Number of Reducer tasks. Set to Maximum 1.
   6) Input and Output file types. Both types are Text.
   7) AWS S3 security credentials. To read the input file from S3.
   8) Input and Output file locations. Captured from command line arguments.
   9) After setting all parameters the MapReduce job is initiated.
   10) Renaming of the final output files. This is done once job successfully completed.

3. The above items are properly implemented with Exception Handling. And the run() method returns the result status.

Mapper:

The Map phase takes input from the Record Reader, processes it, and produces the output as another set of key-value pairs. The following logic is implemented in Mapper.

1. The TrendMapper”” class extends the Mapper class as per the rule.
2. The 2 input arguments are key and values. They are with type Object and Text respectively.
3. The output arguments too are key and values. They are with type Text and Text respectively. This is to achieve the desired Mapper output key value pairs.

4. The map() method is overridden as per the rule. It has the input key value types as mentioned above with one more argument context. The context is used to write the output of this method.
5. The business logic of this method is as follow:
   Reading the input text file line by line. Here each line is treated as one record. Then extracting the Song Id and Date for each record. Filtering is done based on the date range. Then the key value pairs are set with Data as Key and Song as Value.

6. At the end execution of this method for N times the intermediate ouput will be prepared with one key value pair for each line of the input file. This may contain duplicate keys.

Combiner:

The Combiner phase takes each key-value pair from the Map phase, processes it, and produces the output as key-value collection pairs. Here the plain combiner "TrendCombiner" is implemented without any additional logic (Only for Proof of Concept of usage the Combiner).

Reducer:

The Reducer phase takes each key-value collection pair from the Combiner phase, processes it, and passes the output as key-value pairs. Shuffling and sorting are done already. The following logic is implemented in Reducer.

1. The "TrendReducer" class extends the Reducer class as per the rule.
2. The 2 input arguments are key and values. They are with type Text and Text respectively.
3. The output arguments too are key and values. They are with type Text and Text respectively. This is to achieve the desired Final output key value pairs.
4. The reduce() method is overridden as per the rule. It has the input key value types as mentioned above with one more argument context. Usually the context is used to write the output of this method. But here customization is done by using instance of MultipleOutputs class to achieve writing output into multiple files.
5. The business logic of this method is as follow:

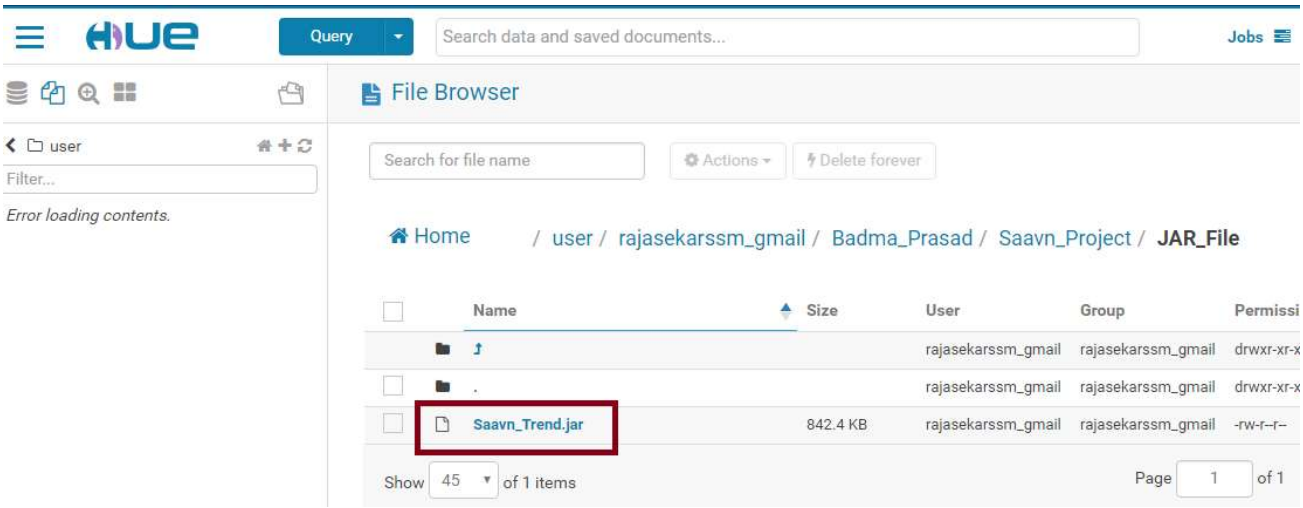
Reading the songs by date wise. Calculating count of songs per day and put them in a Map. Sorting in Max to Min count per date – For this operation a custom function is added which sorts the Map based on the value. Then Appending only first 100 songs into the output string. Writing the output to separate files per date with day number as file name.

Execution Steps:

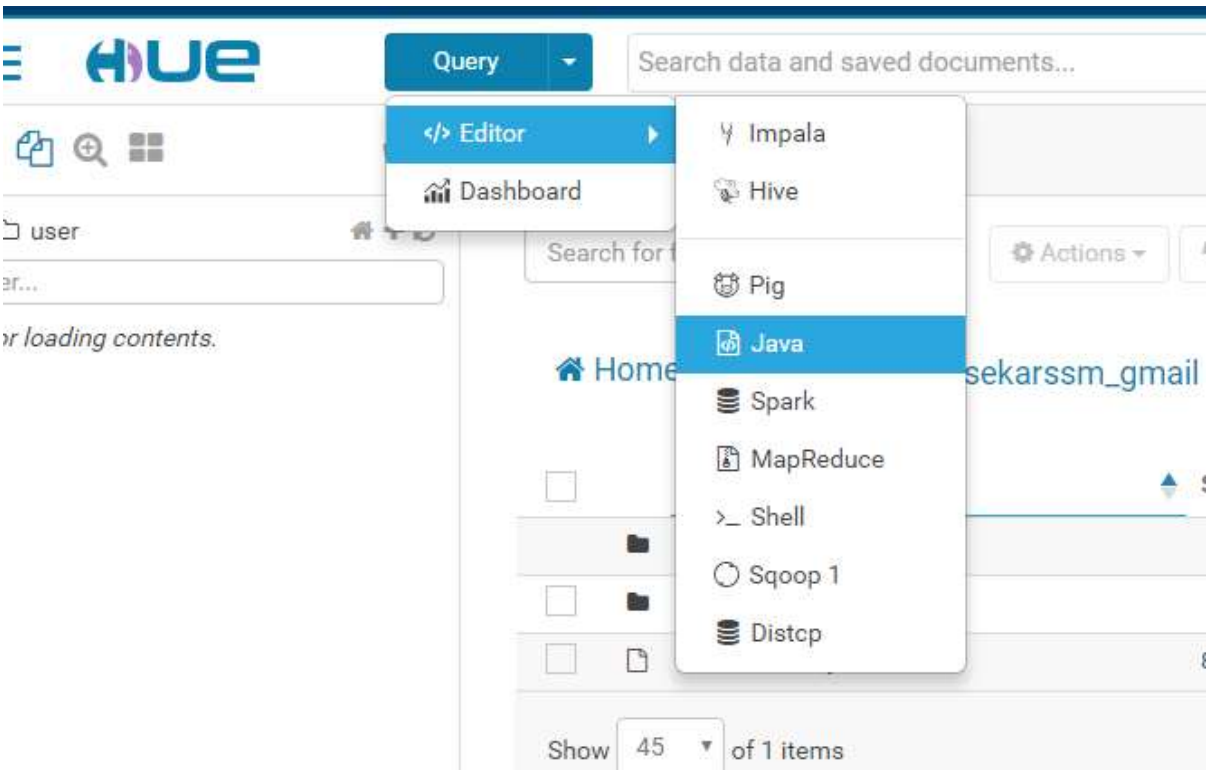
This program is designed to read the input file directly from AWS S3 using HUE interface. The output file is stored in local storage of a machine where the Cloudera cluster is running. It is important to pass the proper input and output arguments as said above when we run this program in command line.

Following steps illustrates how to run this program using HUE interface running on Cloudera cluster.

1) Upload the Jar to a specific folder location using HUE File Browser.



2) Open the item In the Menu Query → Editor → Java

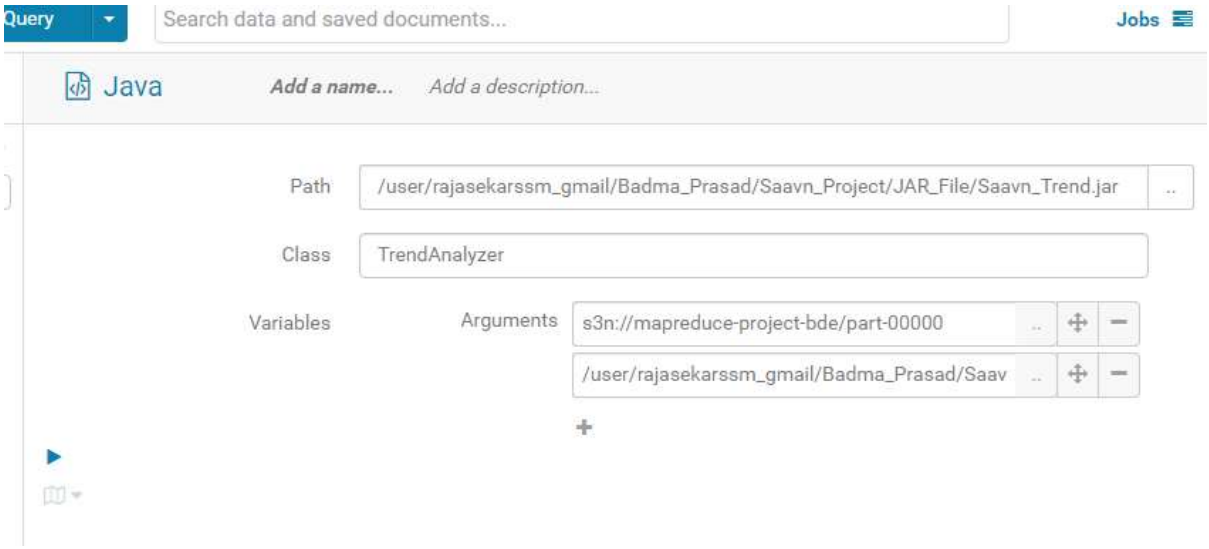


3) Choose the JAR file from the location where it uploaded in above step.
   [I.E]:
   /user/rajasekarssm_gmail/Badma_Prasad/Saavn_Project/JAR_File/**Saavn_Trend.jar**

   Enter the Class Name as **TrendAnalyzer**

   Add Variable Argument 1 as **s3n://mapreduce-project-bde/part-00000**

   Add Variable Argument 2 as
   /user/rajasekarssm_gmail/Badma_Prasad/Saavn_Project/output



4) Execute the program by clicking the execute button in the bottom.
5) Once the program started and successfully completed then the output files will be placed in the output folder mentioned above.

Please Note: The above execution steps are **Example only**. The folder names and other details can be different as per case.