SPOTIFY TOP MUSIC CHARTS BIG DATA ANALYSIS

HADOOP

ABSTRACT

Analyzing the Spotify top music charts dataset using Hadoop ecosystem and generated reports by utilizing Hadoop framework such as Map Reduce, PIG, and HIVE

Vignesh Kumar Baskar

NUID: 002196442

INFO 7250 – Engineering Big Data Systems

FINAL PROJECT REPORT

- **Vignesh Kumar Baskar (002196442)**

Spotify Top Music Charts Analysis

TABLE OF CONTENTS:

CONTENT	PAGE
OVERVIEW	1
DATASET DETAILS	1
EXPLORATORY DATA ANALYSIS	2
LOADING DATA IN HADOOP	5
MAP REDUCE ANALYSIS 1. Top performing Music titles on different streaming regions 2. Total count of TOP200 music categories on the different streaming regions 3. Total count of the VIRAL music categories based on streaming regions 4. Top Music Artist for every streaming region 5. Most streamed Music Title for every year [2017 - 2021] 6. Most trending Music Title for every year [2017 - 2021] 7. Top 10 streaming regions along with their streaming count numbers 8. Top 20 global music artists based on the total streams 9. Average streaming numbers for every region 10. Distinct Spotify Artists in the top chart list 11. Sum of total streaming numbers for every region	6
PIG ANALYSIS 1. Distinct trend patterns among the music titles 2. Finding count of trend patterns based on the streaming regions 3. Total number of artists in every streaming region 4. Top ranked title in the United States on 1st June 2021 5. Getting URL of the “Shape of You” music title for United States region	47
HIVE ANALYSIS 1. Count of top 200 music titles based on the streaming regions 2. Count of viral music hits based on the streaming regions 3. Top 10 streaming countries based on streams 4. Top 20 global music artists based on streams 5. Distinct count values of Regions, Artists, and Titles 6. Top 5 regions with high average streaming numbers 7. Total streaming numbers based for every year (2017-2021)	51

Overview:

- Spotify is an audio streaming platform and one of the largest music streaming service providers over 456 million monthly active users and available in 184 countries

Dataset Details:

- This is a complete dataset of all the Top 200 and Viral 50 music charts published globally by Spotify
- Spotify publishes a new chart every 2-3 days
- This dataset consists of records from 1st Jan 2017 to 31st Dec 2021
- Spotify charts dataset consists of music titles of several categories along with their required attributes such as ranking, date, artist, region, and streams
- The size of the dataset is 3.48 GB

Dataset Link:

- For my INFO 7250 final project, I have taken Spotify top music charts dataset from Kaggle
- Link: https://doi.org/10.34740/KAGGLE/DS/1265407

Exploratory Data Analysis:

- I have performed Exploratory Data Analysis (EDA) using python to analyze the Spotify dataset using visual techniques and to discover the trends and data patterns
- I have attached the entire EDA as a file in my final project report

Dataset Statistics:

Dataset statistics	
Number of variables	9
Number of observations	26173514
Missing cells	5851639
Missing cells (%)	2.5%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	13.1 GiB
Average record size in memory	538.1 B

Variable types	
Categorical	6
Numeric	2
URL	1

Column Variables:

- TITLE

title
Categorical

HIGH CARDINALITY

Distinct	164806
Distinct (%)	0.6%
Missing	11
Missing (%)	< 0.1%
Memory size	1.9 GiB

Shape of You	68965
Believer	60172
Someone You Loved	57595
Dance Monkey	56850
Blinding Lights	56058
Other values (164801)	25873863

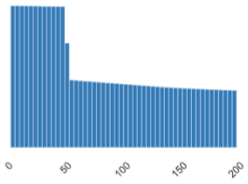
- RANK

rank
Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION

Distinct	200
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	80.92313722

Minimum	1
Maximum	200
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	199.7 MiB



- DATE

date
Categorical

HIGH CARDINALITY

Distinct	1826
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	1.6 GiB

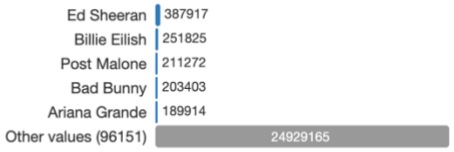
2020-12-24	17022
2021-07-16	17009
2021-05-21	16997
2021-11-26	16974
2021-06-11	16944
Other values (1821)	26088568

- ARTIST

artist
Categorical

HIGH CARDINALITY

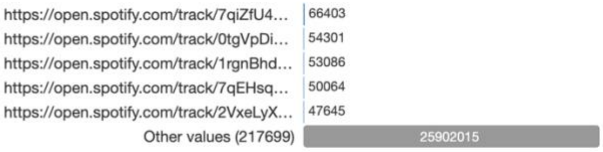
Distinct	96156
Distinct (%)	0.4%
Missing	18
Missing (%)	< 0.1%
Memory size	1.8 GiB



- URL

url
URL

Distinct	217704
Distinct (%)	0.8%
Missing	0
Missing (%)	0.0%
Memory size	2.7 GiB



- REGION

region
Categorical

HIGH CARDINALITY

Distinct	70
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	1.6 GiB

- CHART

chart
Categorical

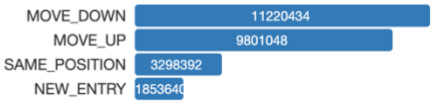
HIGH CORRELATION

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	1.5 GiB

- TREND

trend
Categorical

Distinct	4
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	1.6 GiB



- STREAMS

streams
Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	788013
Distinct (%)	3.9%
Missing	5851610
Missing (%)	22.4%
Infinite	0
Infinite (%)	0.0%
Mean	55261.31438

Minimum	1001
Maximum	19749704
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	199.7 MiB

SAMPLE DATA:

title	rank	date	artist	url	region	chart	trend
Chantaje (feat. Maluma)	1	2017-01-01	Shakira	https://open.spotify.com/track/6mICuAdrwEjh6Y6lroV2Kg	Argentina	top200	SAME
Vente Pa' Ca (feat. Maluma)	2	2017-01-01	Ricky Martin	https://open.spotify.com/track/7DM4BPaS7uofFul3ywMe46	Argentina	top200	MOVE
Reggaetón Lento (Bailemos)	3	2017-01-01	CNCO	https://open.spotify.com/track/3AEZUABDXNtecAOSC1qTfo	Argentina	top200	MOVE
Safari	4	2017-01-01	J Balvin, Pharrell Williams, BIA, Sky	https://open.spotify.com/track/6rQSrBHf7HlZjtcMZ4S4bO	Argentina	top200	SAME
Shaky Shaky	5	2017-01-01	Daddy Yankee	https://open.spotify.com/track/58IL315gMSTD37DOZPJ2hf	Argentina	top200	MOVE
Traicionera	6	2017-01-01	Sebastian Yatra	https://open.spotify.com/track/5J1c3M4EldCfNxXwrwt8mT	Argentina	top200	MOVE
Cuando Se Pone a Bailar	7	2017-01-01	Rombai	https://open.spotify.com/track/1MpKZi1zTXpERKwxmOu1PH	Argentina	top200	MOVE
Otra vez (feat. J Balvin)	8	2017-01-01	Zion & Lennox	https://open.spotify.com/track/3QwBODjSEzelZyVjxPOHdq	Argentina	top200	MOVE
La Bicicleta	9	2017-01-01	Carlos Vives, Shakira	https://open.spotify.com/track/0sXvAOmXgjR2QUqLK1MltU	Argentina	top200	MOVE
Dile Que Tu Me Quieres	10	2017-01-01	Ozuna	https://open.spotify.com/track/20ZAJdsKB5IGbGj4ilRt2o	Argentina	top200	MOVE

Loaded the Spotify dataset in Hadoop

Displaying list of files in the Project Hadoop folder

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -ls /Project/
2022-12-14 15:48:00,261 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rwxrwxrwx   1 vignesh supergroup  246297603 2022-12-11 04:01 /Project/RankStreams.csv
-rwxrwxrwx   1 vignesh supergroup 3495541597 2022-12-04 16:24 /Project/spotify.csv
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

Viewing the file in Hadoop

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -head /Project/spotify.csv
2022-12-14 15:48:42,883 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Chantaje (feat. Maluma),1,2017-01-01,Shakira,https://open.spotify.com/track/6mICuAdrwEjh6Y6lroV2Kg,Argentina,top200,SAME_POSITION,253019
Vente Pa' Ca (feat. Maluma),2,2017-01-01,Ricky Martin,https://open.spotify.com/track/7DM4BPaS7uofFul3ywMe46,Argentina,top200,MOVE_UP,223988.0
Reggaetón Lento (Bailemos),3,2017-01-01,CNCO,https://open.spotify.com/track/3AEZUABDXNtecAOSC1qTfo,Argentina,top200,MOVE_DOWN,210943.0
Safari,4,2017-01-01,J Balvin Pharrell Williams BIA Sky,https://open.spotify.com/track/6rQSrBHf7HlZjtcMZ4S4bO,Argentina,top200,SAME_POSITION,173865.0
Shaky Shaky,5,2017-01-01,Daddy Yankee,https://open.spotify.com/track/58IL315gMSTD37DOZPJ2hf,Argentina,top200,MOVE_UP,153956.0
Traicionera,6,2017-01-01,Sebastian Yatra,https://open.spotify.com/track/5J1c3M4EldCfNxXwrwt8mT,Argentina,top200,MOVE_DOWN,151140.0
Cuando Se Pone a Bailar,7,2017-01-01,Rombai,https://open.spotify.com/track/1MpKZi1zTXpERKwxmOu1PH,Argentina,top200,MOVE_DOWN,148369.0
Otra vez (feat. J Balvin),8,2017-01-01,Zion & Lennox,https://open.spo
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

Displaying summary of the Spotify dataset file

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -du -s /Project/spotify.csv
2022-12-14 15:50:33,562 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
3495541597  3495541597  /Project/spotify.csv
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

Saved the Map Reduce output files in the ProjectResults folder in Hadoop

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -ls /ProjectResults/
2022-12-14 15:57:26,593 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 16 items
drwxrwxrwx   - vignesh supergroup          0 2022-12-06 21:07 /ProjectResults/AvgStreamingNoByRegions
drwxrwxrwx   - vignesh supergroup          0 2022-12-09 21:42 /ProjectResults/DistinctArtists
drwxrwxrwx   - vignesh supergroup          0 2022-11-27 16:53 /ProjectResults/MostStreamedMusicTitlesByYear
drwxrwxrwx   - vignesh supergroup          0 2022-12-02 01:49 /ProjectResults/StreamsByArtist
drwxrwxrwx   - vignesh supergroup          0 2022-11-27 17:42 /ProjectResults/StreamsByRegion
drwxrwxrwx   - vignesh supergroup          0 2022-11-27 13:18 /ProjectResults/SumOfTotalStreamsByRegions
drwxrwxrwx   - vignesh supergroup          0 2022-11-27 17:42 /ProjectResults/Top10StreamingRegions
drwxrwxrwx   - vignesh supergroup          0 2022-11-24 04:37 /ProjectResults/Top200CategoryCount
drwxrwxrwx   - vignesh supergroup          0 2022-12-02 01:49 /ProjectResults/Top20GlobalArtists
drwxrwxrwx   - vignesh supergroup          0 2022-11-25 18:14 /ProjectResults/TopMusicArtistByRegion
drwxrwxrwx   - vignesh supergroup          0 2022-12-11 21:50 /ProjectResults/TopMusicArtists
drwxrwxrwx   - vignesh supergroup          0 2022-12-11 21:33 /ProjectResults/TopMusicTitles
drwxrwxrwx   - vignesh supergroup          0 2022-11-25 18:19 /ProjectResults/TopMusicTitlesByRegion
drwxrwxrwx   - vignesh supergroup          0 2022-11-27 13:21 /ProjectResults/TotalStreamsBasedOnArtist
drwxrwxrwx   - vignesh supergroup          0 2022-11-25 18:13 /ProjectResults/TrendingMusicTitlesEachYear
drwxrwxrwx   - vignesh supergroup          0 2022-11-24 07:34 /ProjectResults/ViralHitCounts
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

Big Data Analysis:

Map Reduce:

1. **Top performing Music titles on different streaming regions**

Code:

```
package com.mycompany.finalproject_spotifycharts;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 *
 * @author vignesh
 */

public class TopMusicTitles {

   public static class Spotify_Mapper extends Mapper<Object, Text, Text, Text> {

      @Override
      public void map (Object key, Text value, Context context) throws IOException, InterruptedException {

         String[] tokens = value.toString().split(",");

         Text region = new Text();
         region.set(tokens[5]);

         Text title = new Text(tokens[0]);

         context.write(region, title);

      }
   }
```

```
    public static class Spotify_Reducer extends Reducer<Text, Text, Text, Text> {

        private Text topMusicTitle = new Text();

        @Override
        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

            // Storing the title and it's count as a key value pairs in HashMap
            HashMap<String, Integer> titleCounts = new HashMap();

            int totalTitles = 0;

            for(Text val: values) {

                String title = val.toString();
                int count = 0;

                // checking for the same title
                if(titleCounts.containsKey(title)){
                    // get the count of the respective title stored in the Hash Map variable
                    count = titleCounts.get(title);
                }

                titleCounts.put(title, count + 1);
                totalTitles += 1;

            }

            // Calling the function to get the top counted title
            String topTitle = getTopTitlesByRegion(titleCounts);

            topMusicTitle.set(topTitle);

            context.write(key, topMusicTitle);

        }

        // Getting the Top Titles for every region
        private String getTopTitlesByRegion(Map<String, Integer> titleCounts) {

            String topTitle = "";

            int maxCount = 0;

            // entrySet returns a set view of the hash map
            for (Map.Entry<String, Integer> titleCount : titleCounts.entrySet()) {
```

```
            // checking if count value > maxCount value
            if (titleCount.getValue() > maxCount) {
                maxCount = titleCount.getValue();
                topTitle = titleCount.getKey();
            }

        }

        return topTitle;

    }

}

// Adding Combiner function to summarize the map output records with the same key
public static class Spotify_Combiner extends Reducer<Text, Text, Text, Text> {

    private Text topMusicTitle = new Text();

    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

        // Storing the title and it's count as a key value pairs in HashMap
        HashMap<String, Integer> titleCounts = new HashMap();

        int totalTitles = 0;

        for(Text val: values) {

            String title = val.toString();
            int count = 0;

            // checking for the same title
            if(titleCounts.containsKey(title)){
                // get the count of the respective title stored in the Hash Map variable
                count = titleCounts.get(title);
            }

            titleCounts.put(title, count + 1);
            totalTitles += 1;

        }

        // Calling the function to get the top counted title
        String topTitle = getTopTitlesByRegion(titleCounts);
```

```
        topMusicTitle.set(topTitle);

        context.write(key, topMusicTitle);

    }

    // Getting the Top Titles for every region
    private String getTopTitlesByRegion(Map<String, Integer> titleCounts) {

        String topTitle = "";

        int maxCount = 0;

        // entrySet returns a set view of the hash map
        for (Map.Entry<String, Integer> titleCount : titleCounts.entrySet()) {

            // checking if count value > maxCount value
            if (titleCount.getValue() > maxCount) {
                maxCount = titleCount.getValue();
                topTitle = titleCount.getKey();
            }

        }

        return topTitle;

    }

}


public static void main (String[] args) throws Exception  {

    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Top Music Charts based on Region");
    job.setJarByClass(TopMusicTitles.class);

    job.setMapperClass(Spotify_Mapper.class);
    job.setReducerClass(Spotify_Reducer.class);
    job.setCombinerClass(Spotify_Combiner.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
```

```
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true)?0:1);

    }

}
```

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /ProjectResults/TopMusicTitles/part-r-00000
2022-12-14 16:04:27,674 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Andorra Closer
Argentina       Me Gusta
Australia       Happier
Austria Ohne mein Team
Belgium Shape of You
Bolivia Me Rehúso
Brazil  Só Você
Bulgaria        Lucid Dreams
Canada  Falling
Chile   Me Rehúso
Colombia        Culpables
Costa Rica      Shape of You
Czech Republic  Happier
Denmark Happier
Dominican Republic      La Difícil
Ecuador Culpables
Egypt   Ya Sattar
```

File contents

```
Andorra  Closer
Argentina Me Gusta
Australia  Happier
Austria  Ohne mein Team
Belgium  Shape of You
Bolivia  Me Rehúso
Brazil  Só Você
Bulgaria  Lucid Dreams
```

2. **Total count of the TOP200 music categories on the different streaming regions**

Code:

```
package com.mycompany.finalproject_spotifycharts;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

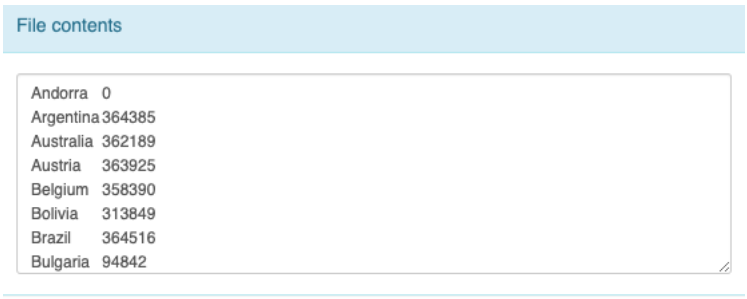
```
/**
 *
 * @author vignesh
 */
public class Top200CatergoryCount {

   public static class Spotify_Mapper extends Mapper<Object, Text, Text, Text> {

      @Override
      public void map (Object key, Text value, Context context) throws IOException, InterruptedException {

         String[] tokens = value.toString().split(",");
         Text region = new Text();
         region.set(tokens[5]);

         // passing the region as key and category as values
         context.write(region, new Text(tokens[6]));
      }
   }

   public static class Spotify_Reducer extends Reducer<Text, Text, Text, IntWritable> {

      private IntWritable sumObj = new IntWritable();

      @Override
      public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

         int sum = 0;

         for(Text val: values) {

            // checking if the category is top200 and then adds the sum varibale by 1
            if(val.toString().equals("top200")){
               sum += 1;
            }

         }

         sumObj.set(sum);

         context.write(key, sumObj);

      }

   }
```

```
    public static void main (String[] args) throws Exception  {

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Top200 category counts based on Region");
        job.setJarByClass(Top200CatergoryCount.class);

        job.setMapperClass(Spotify_Mapper.class);
        job.setReducerClass(Spotify_Reducer.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true)?0:1);

    }

}
```

Output:

The entire output file has been attached in the final report submission.

```
(base) vigneshs@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /ProjectResults/Top200CategoryCount/part-r-00000
2022-12-14 16:08:12,375 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Andorra 0
Argentina       364385
Australia       362189
Austria 363925
Belgium 358390
Bolivia 313849
Brazil  364516
Bulgaria        94842
Canada  361390
Chile   358185
Colombia        358384
Costa Rica      358384
Czech Republic  357276
Denmark 358389
Dominican Republic      342318
Ecuador 358384
Egypt   123769
El Salvador     302106
Estonia 108007
Finland 358192
```



3. Total count of the VIRAL music categories based on streaming regions

Code:

```
package com.mycompany.finalproject_spotifycharts;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 *
 * @author vignesh
 */
public class ViralMusicHitCounts {

   public static class Spotify_Mapper extends Mapper<Object, Text, Text, Text> {

      @Override
      public void map (Object key, Text value, Context context) throws IOException, InterruptedException {

         String[] tokens = value.toString().split(",");
         Text region = new Text();
         region.set(tokens[5]);

         // passing the region as key and category as values
         context.write(region, new Text(tokens[6]));
      }
   }

   public static class Spotify_Reducer extends Reducer<Text, Text, Text, IntWritable> {
```

```
    private IntWritable sumObj = new IntWritable();

    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

        int sum = 0;

        for(Text val: values) {

            // checking if the category is viral50 and then adds the sum varibale by 1
            if(val.toString().equals("viral50")){
                sum += 1;
            }

        }

        sumObj.set(sum);

        context.write(key, sumObj);

    }
  }

  public static void main (String[] args) throws Exception  {

    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Viral Hits category counts based on Region");
    job.setJarByClass(ViralMusicHitCounts.class);

    job.setMapperClass(Spotify_Mapper.class);
    job.setReducerClass(Spotify_Reducer.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true)?0:1);

  }

}
```

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /ProjectResults/ViralHitCounts/part-r-00000
2022-12-14 16:10:26,138 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Andorra 79592
Argentina       90923
Australia       90914
Austria 90668
Belgium 90929
Bolivia 90664
Brazil  89923
Bulgaria        90665
Canada  90889
Chile   90915
Colombia        90902
Costa Rica      90918
Czech Republic  90725
Denmark 90930
Dominican Republic      90823
Ecuador 90905
Egypt   56700
```

File contents

```
Andorra 79592
Argentina 90923
Australia 90914
Austria 90668
Belgium 90929
Bolivia 90664
Brazil 89923
Bulgaria 90665
```

4. **Top Music Artist for every streaming region**

Code:

```
package com.mycompany.finalproject_spotifycharts;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 *
 * @author vignesh
 */
```

```
public class TopMusicArtists {

   public static class Spotify_Mapper extends Mapper<Object, Text, Text, Text> {

      @Override
      public void map (Object key, Text value, Context context) throws IOException, InterruptedException {

         String[] tokens = value.toString().split(",");
         Text region = new Text();
         region.set(tokens[5]);

         context.write(region, new Text(tokens[3]));
      }
   }

   public static class Spotify_Reducer extends Reducer<Text, Text, Text, Text> {

      private Text topMusicArtist = new Text();

      @Override
      public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

         // Storing the artist and their count as a key value pairs in HashMap
         HashMap<String, Integer> artistCounts = new HashMap();

         int totalArtists = 0;

         for(Text val: values) {

            String artist = val.toString();
            int count = 0;

            // checking for the same artist
            if(artistCounts.containsKey(artist)){
               // get the count of the respective artist stored in the Hash Map variable
               count = artistCounts.get(artist);
            }

            artistCounts.put(artist, count + 1);
            totalArtists += 1;

         }

         // Calling the function to get the top counted artist
         String topArtist = getTopArtistByRegion(artistCounts);

         topMusicArtist.set(topArtist);
```

```
        context.write(key, topMusicArtist);

    }

    // Getting the Top Artists for every region
    private String getTopArtistByRegion(Map<String, Integer> artistCounts) {

        String topArtist = "";

        int maxCount = 0;

        // entrySet returns a set view of the hash map
        for (Map.Entry<String, Integer> artistCount : artistCounts.entrySet()) {

            // checking if count value > maxCount value
            if (artistCount.getValue() > maxCount) {
                maxCount = artistCount.getValue();
                topArtist = artistCount.getKey();
            }

        }

        return topArtist;

    }

}

// Adding Combiner function to summarize the map output records with the same key
public static class Spotify_Combiner extends Reducer<Text, Text, Text, Text> {

    private Text topMusicArtist = new Text();

    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

        // Storing the artist and their count as a key value pairs in HashMap
        HashMap<String, Integer> artistCounts = new HashMap();

        int totalArtists = 0;

        for(Text val: values) {

            String artist = val.toString();
            int count = 0;
```

```
            // checking for the same artist
            if(artistCounts.containsKey(artist)){
               // get the count of the respective artist stored in the Hash Map variable
               count = artistCounts.get(artist);
            }

            artistCounts.put(artist, count + 1);
            totalArtists += 1;

         }

         // Calling the function to get the top counted artist
         String topArtist = getTopArtistByRegion(artistCounts);

         topMusicArtist.set(topArtist);

         context.write(key, topMusicArtist);

      }

      // Getting the Top Artists for every region
      private String getTopArtistByRegion(Map<String, Integer> artistCounts) {

         String topArtist = "";

         int maxCount = 0;

         // entrySet returns a set view of the hash map
         for (Map.Entry<String, Integer> artistCount : artistCounts.entrySet()) {

            // checking if count value > maxCount value
            if (artistCount.getValue() > maxCount) {
               maxCount = artistCount.getValue();
               topArtist = artistCount.getKey();
            }

         }

         return topArtist;

      }

   }


   public static void main (String[] args) throws Exception  {

      Configuration conf = new Configuration();
```

```
    Job job = Job.getInstance(conf, "Top Music Artists based on Region");
    job.setJarByClass(TopMusicArtists.class);

    job.setMapperClass(Spotify_Mapper.class);
    job.setReducerClass(Spotify_Reducer.class);
    job.setCombinerClass(Spotify_Combiner.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true)?0:1);

  }

}
```

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /ProjectResults/TopMusicArtists/part-r-00000
2022-12-14 16:12:41,479 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Andorra Rauw Alejandro
Argentina       Paulo Londra
Australia       Ed Sheeran
Austria Ed Sheeran
Belgium Ed Sheeran
Bolivia Bad Bunny
Brazil  Henrique & Juliano
Bulgaria        Pop Smoke
Canada  Post Malone
Chile   Bad Bunny
Colombia        Bad Bunny
Costa Rica      Bad Bunny
Czech Republic  Viktor Sheen
Denmark Ed Sheeran
Dominican Republic      Bad Bunny
Ecuador Bad Bunny
Egypt   Wegz
El Salvador     Bad Bunny
Estonia Billie Eilish
Finland JVG
France  Ninho
Germany Apache 207
```

File contents

```
Andorra  Rauw Alejandro
Argentina Paulo Londra
Australia  Ed Sheeran
Austria  Ed Sheeran
Belgium  Ed Sheeran
Bolivia  Bad Bunny
Brazil  Henrique & Juliano
Bulgaria  Pop Smoke
```

Ed Sheeran is one of the Top Artists in many regions.

5. **Most streamed Music Title for every year [2017 - 2021]**

Code:

```
package com.mycompany.finalproject_spotifycharts;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 *
 * @author vignesh
 */

public class MostStreamedMusicTitles {


   public static class Spotify_Mapper extends Mapper<Object, Text, Text, Text> {


      @Override
      public void map (Object key, Text value, Context context) throws IOException, InterruptedException {

         String[] tokens = value.toString().split(",", -1);

         Text year = new Text();

         String[] date = tokens[2].split("-");
         year.set(date[0]);

         // Passing the year as a key and combination of streams and title as value to the reducer
         context.write(year, new Text(tokens[8] + ";" + tokens[0]));

      }
   }
```

```
public static class Spotify_Reducer extends Reducer<Text, Text, Text, Text> {

    private Text mostStreamedMusicTitle = new Text();

    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

        double totalStreams = 0;
        double streams;

        HashMap<String, Double> titleStreams = new HashMap();

        for(Text val: values) {

            String[] records = val.toString().split(";",-1);

            String title = records[1];

            // if there's empty title then assigning the value of the stream to be 0
            if(records[0].equals("")){
                streams = 0;
            }else{
                streams = Double.parseDouble(records[0]);
            }

            double count = 0;

            // checking for the same title
            if(titleStreams.containsKey(title)){
                // get the count of the respective title stored in the Hash Map variable
                count = titleStreams.get(title);
            }

            titleStreams.put(title, count + streams);
            totalStreams += count;

        }

        // Calling the function to get the top counted artist
        String topStreamed = getMostStreamedMusic(titleStreams);

        mostStreamedMusicTitle.set(topStreamed);

        context.write(key, mostStreamedMusicTitle);

    }
```

```
    // Getting the most streamed music titles for every year
    private String getMostStreamedMusic(Map<String, Double> titleStreams) {

        String mostStreamedTitle = "";

        double maxCount = 0;

        // entrySet returns a set view of the hash map
        for (Map.Entry<String, Double> titleStream : titleStreams.entrySet()) {

            // checking if count value > maxCount value
            if (titleStream.getValue() > maxCount) {
                maxCount = titleStream.getValue();
                mostStreamedTitle = titleStream.getKey();
            }

        }

        return mostStreamedTitle;

    }

}

public static void main (String[] args) throws Exception  {

    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Most streamed music titles every year");
    job.setJarByClass(MostStreamedMusicTitles.class);

    job.setMapperClass(Spotify_Mapper.class);
    job.setReducerClass(Spotify_Reducer.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true)?0:1);

}


}
```

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /ProjectResults/MostStreamedMusicTitlesByYear/part-r-00000
2022-12-14 16:26:20,640 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2017    Shape of You
2018    God's Plan
2019    Señorita
2020    Blinding Lights
2021    STAY (with Justin Bieber)
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

File contents

2017	Shape of You
2018	God's Plan
2019	Señorita
2020	Blinding Lights
2021	STAY (with Justin Bieber)

6. **Most trending Music Title for every year [2017 - 2021]**

Code:

```
package com.mycompany.finalproject_spotifycharts;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 *
 * @author vignesh
 */
public class TrendingMusicTitles {

    public static class Spotify_Mapper extends Mapper<Object, Text, Text, Text> {

        @Override
```

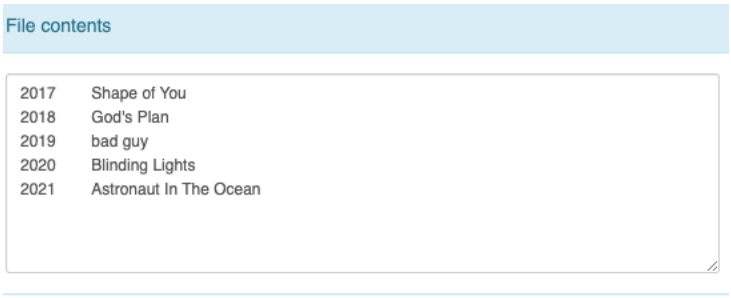
```
    public void map (Object key, Text value, Context context) throws IOException, InterruptedException {

        String[] tokens = value.toString().split(",");

        Text year = new Text();

        String[] date = tokens[2].split("-");
        year.set(date[0]);

        Text title = new Text(tokens[0]);

        context.write(year, title);

    }
}

public static class Spotify_Reducer extends Reducer<Text, Text, Text, Text> {

    private Text topMusicTitle = new Text();

    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

        HashMap<String, Integer> titleCounts = new HashMap();

        int totalTitles = 0;

        for(Text val: values) {

            String title = val.toString();
            int count = 0;

            if(titleCounts.containsKey(title)){
                count = titleCounts.get(title);
            }

            titleCounts.put(title, count + 1);
            totalTitles += 1;

        }

        String topTitle = getTopArtistByRegion(titleCounts);

        topMusicTitle.set(topTitle);

        context.write(key, topMusicTitle);
```

```
    }

    private String getTopArtistByRegion(Map<String, Integer> titleCounts) {

        String topTitle = "";

        int maxCount = 0;

        for (Map.Entry<String, Integer> titleCount : titleCounts.entrySet()) {
            if (titleCount.getValue() > maxCount) {
                maxCount = titleCount.getValue();
                topTitle = titleCount.getKey();
            }

        }

        return topTitle;

    }

}


public static void main (String[] args) throws Exception  {

    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Trending music titles every year");
    job.setJarByClass(TrendingMusicTitles.class);

    job.setMapperClass(Spotify_Mapper.class);
    job.setReducerClass(Spotify_Reducer.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true)?0:1);

}

}
```

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /ProjectResults/TrendingMusicTitlesEachYear/part-r-00000
2022-12-14 16:28:45,618 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2017    Shape of You
2018    God's Plan
2019    bad guy
2020    Blinding Lights
2021    Astronaut In The Ocean
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```



7. **Top 10 streaming regions along with their streaming count numbers**

Code:

package com.mycompany.finalproject_spotifycharts;

import java.io.IOException;
import java.util.Comparator;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.TreeMap;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

*/**
 **
 * @author vignesh
 /

```
public class Top10StreamingRegions {

   // Mapper of the Job1
   public static class Spotify_Mapper extends Mapper<Object, Text, Text, Text> {

      @Override
      public void map (Object key, Text value, Context context) throws IOException, InterruptedException {

         String[] tokens = value.toString().split(",", -1);
         Text region = new Text();
         region.set(tokens[5]);

         // passing the region as a key and streams as a value
         context.write(region, new Text(new Text(tokens[8])));

      }

   }

   // Reducer of the Job1
   public static class Spotify_Reducer extends Reducer<Text, Text, Text, DoubleWritable> {

      private DoubleWritable sumObj = new DoubleWritable();

      @Override
      public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

         double sum = 0;

         for(Text val: values) {

            // if there's empty streams then assigning the value of sum to be 0
            if(val.toString().equals("")){
               sum += 0;
            }
            else{
               double temp = Double.parseDouble(val.toString());
               sum += temp;
            }

         }

         sumObj.set(sum);

         context.write(key, sumObj);
```

```
    }

}

// Mapper of the Job2
public static class Spotify_TopNMapper extends Mapper<Object, Text, Text, DoubleWritable> {

    private int n;
    private TreeMap<Double, String> stream_list;

    // Called once at the beginning of the task
    @Override
    public void setup(Context context) {
        n = 10;

        // local list for sorting stream values
        stream_list = new TreeMap<Double, String>();

    }

    // Called once for each key/value pair in the input split
    @Override
    public void map(Object key, Text value, Context context) {

        // Spiltting the input rows by tab space since hadoop stores stores the key value output as same
        String[] line = value.toString().split("\t");

        // adds the total streams as key, and region as values for sorting based on total stream counts
        stream_list.put(Double.valueOf(line[1]), line[0]);

        // checks if the local stream_list contains more than 10 elements
        if (stream_list.size() > n){
            // removing the first element with the smallest streaming count value
            stream_list.remove(stream_list.firstKey());
        }

    }

    // Called once at the end of the task
    @Override
    public void cleanup(Context context) throws IOException, InterruptedException {

        // write the topN streaming regions before proceeding with TopNReducer

        for (Map.Entry<Double, String> entry : stream_list.entrySet()) {

            String region = entry.getValue();
            Double stream = entry.getKey();
```

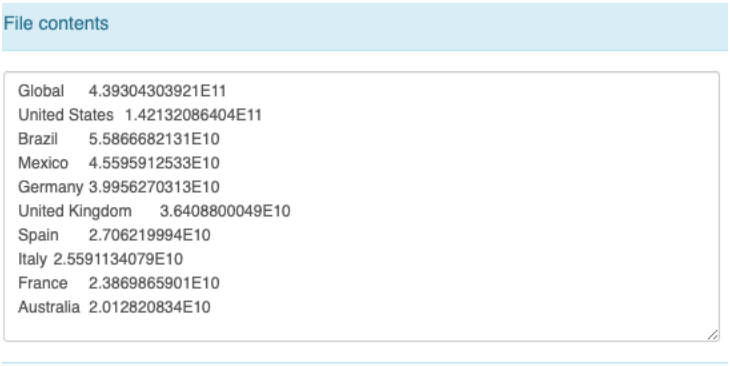
```
            // passes the region as a key and total streams as a value
            context.write(new Text(region), new DoubleWritable(stream));
        }

    }

}


// Reducer of the Job2 {Input: Region, TotalStreams }
public static class Spotify_TopNReducer extends Reducer<Text, DoubleWritable, Text, DoubleWritable> {

    private int n;
    private TreeMap<Double, String> stream_list;

    // Called once at the start of the task
    @Override
    public void setup(Context context) {
        n = 10;

        // list with regions gloabally sorted by their streaming counts
        stream_list = new TreeMap<Double, String>();
    }

    // This method is called once for each key
    @Override
    public void reduce(Text key, Iterable<DoubleWritable> values, Context context) {

        double streamcount = 0;

        // get the streams for each regions
        for(DoubleWritable value : values){
            streamcount = value.get();
        }

        // adds the stream count as key and the region as value for sorting based on the stream count
        stream_list.put(streamcount, key.toString());

        // checks if the global stream_list contains more than 10 elements
        if (stream_list.size() > n){

            // removing the first element with the smallest streaming count value
            stream_list.remove(stream_list.firstKey());
        }

    }
```

```
    // Called once at the end of the task
    @Override
    public void cleanup(Context context) throws IOException, InterruptedException
    {

        // HashMap for sorting the stream count in reverse order
        LinkedHashMap<String, Double> streams_list_sorted = new LinkedHashMap<>();

        stream_list.entrySet()
                .stream()
                .sorted(Map.Entry.comparingByKey(Comparator.reverseOrder()))
                .forEachOrdered(x -> streams_list_sorted.put( x.getValue(), x.getKey()));


        // the output will be sorted by the reverse streaming count

        for (Map.Entry< String, Double> entry : streams_list_sorted.entrySet()) {

            String region = entry.getKey();
            Double stream = entry.getValue();
            context.write( new Text(region), new DoubleWritable(stream));

        }

    }

}


// Main method
public static void main (String[] args) throws Exception  {

    Configuration conf = new Configuration();

    // Job1
    Job job1 = Job.getInstance(conf, "Sum of total streams based on regions");
    job1.setJarByClass(Top10StreamingRegions.class);

    job1.setMapperClass(Spotify_Mapper.class);
    job1.setReducerClass(Spotify_Reducer.class);

    job1.setMapOutputKeyClass(Text.class);
    job1.setMapOutputValueClass(Text.class);

    job1.setOutputKeyClass(Text.class);
    job1.setOutputValueClass(DoubleWritable.class);

    FileInputFormat.addInputPath(job1, new Path(args[0]));
```

```
        FileOutputFormat.setOutputPath(job1, new Path(args[1]));


        boolean job1_success = job1.waitForCompletion(true);

        if(job1_success){

            // Job2
            Job job2 = Job.getInstance(conf, "Top 10 Streaming Regions");
            job2.setJarByClass(Top10StreamingRegions.class);

            job2.setMapperClass(Spotify_TopNMapper.class);
            job2.setReducerClass(Spotify_TopNReducer.class);

            job2.setMapOutputKeyClass(Text.class);
            job2.setMapOutputValueClass(DoubleWritable.class);

            job2.setOutputKeyClass(DoubleWritable.class);
            job2.setOutputValueClass(Text.class);

            FileInputFormat.addInputPath(job2, new Path(args[2]));
            FileOutputFormat.setOutputPath(job2, new Path(args[3]));

            job2.waitForCompletion(true);

        }
    }
}
```

Output:

The entire output file has been attached in the final report submission.



After Global region, ***United States*** has the most streaming counts.

8. **Top 20 global music artists based on the total streams**

Code:

```
package com.mycompany.finalproject_spotifycharts;

import java.io.IOException;
import java.util.Comparator;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.TreeMap;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 *
 * @author vignesh
 */

public class Top20GlobalArtists {

   // Mapper of Job1
   public static class Spotify_Mapper extends Mapper<Object, Text, Text, Text> {

      @Override
      public void map (Object key, Text value, Context context) throws IOException, InterruptedException {
```

```
        String[] tokens = value.toString().split(",", -1);
        Text artist = new Text();
        artist.set(tokens[3]);

        // passing the artist as a key and streams as a value
        context.write(artist, new Text(new Text(tokens[8])));

    }

}

// Reducer of Job1
public static class Spotify_Reducer extends Reducer<Text, Text, Text, DoubleWritable> {

    private DoubleWritable sumObj = new DoubleWritable();

    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

        double sum = 0;

        for(Text val: values) {

            // if there's empty stream value then assigning the value of sum to be 0
            if(val.toString().equals("")){
                sum += 0;
            }
            else{
                double temp = Double.parseDouble(val.toString());
                sum += temp;
            }

        }

        sumObj.set(sum);

        context.write(key, sumObj);

    }

}

// Mapper of Job2
public static class Spotify_Top20Mapper extends Mapper<Object, Text, Text, DoubleWritable> {

    private int n;
```

```
    private TreeMap<Double, String> stream_list;

    // Called once at the beginning of the task
    @Override
    public void setup(Context context) {
        n = 20;
        // local list for sorting stream values
        stream_list = new TreeMap<Double, String>();

    }

    // Called once for each key/value pair in the input split
    @Override
    public void map(Object key, Text value, Context context) {

        // Spiltting the input rows by tab space since hadoop stores stores the key value output as same
        String[] line = value.toString().split("\t");

        // adds the total streams as key, and artist as values for sorting based on total stream counts
        stream_list.put(Double.valueOf(line[1]), line[0]);


        // checks if the local stream_list contains more than 20 elements
        if (stream_list.size() > n){
            // removing the first element with the smallest streaming count value
            stream_list.remove(stream_list.firstKey());
        }

    }

    // Called once at the end of the task
    @Override
    public void cleanup(Context context) throws IOException, InterruptedException {

        // write the topN streaming artists before proceeding with TopNReducer

        for (Map.Entry<Double, String> entry : stream_list.entrySet()) {

            String artist = entry.getValue();
            Double stream = entry.getKey();

            // passes the artist as a key and total streams as a value
            context.write(new Text(artist), new DoubleWritable(stream));
        }

    }

}
```

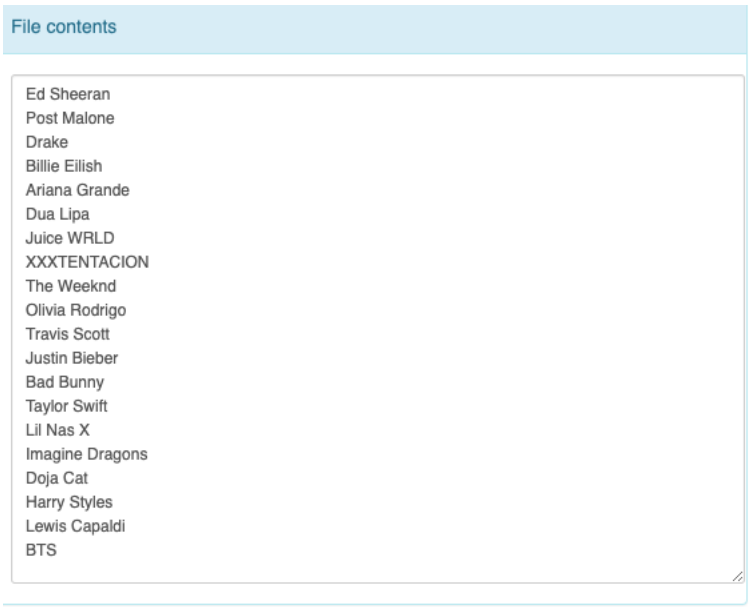
```
// Reducer of Job2 {Input: Artist, TotalStreams }
public static class Spotify_Top20Reducer extends Reducer<Text, DoubleWritable, NullWritable, Text> {

   private int n;
   private TreeMap<Double, String> stream_list;


   // Called once at the start of the task
   @Override
   public void setup(Reducer.Context context) {
      n = 20;

      // list with artists gloabally sorted by their streaming counts
      stream_list = new TreeMap<Double, String>();
   }

   // This method is called once for each key
   @Override
   public void reduce(Text key, Iterable<DoubleWritable> values, Context context) {

      double streamcount = 0;

      // get the streams for each artist
      for(DoubleWritable value : values){
         streamcount = value.get();
      }

      // adds the stream count as key and the artist as value for sorting based on the stream count
      stream_list.put(streamcount, key.toString());

      // checks if the global stream_list contains more than 20 elements
      if (stream_list.size() > n){

         // removing the first element with the smallest streaming count value
         stream_list.remove(stream_list.firstKey());
      }

   }

   // Called once at the end of the task
   @Override
   public void cleanup(Context context) throws IOException, InterruptedException
   {

      // HashMap for sorting the stream count in reverse order
      LinkedHashMap<String, Double> streams_list_sorted = new LinkedHashMap<>();
```

```
        stream_list.entrySet()
                .stream()
                .sorted(Map.Entry.comparingByKey(Comparator.reverseOrder()))
                .forEachOrdered(x -> streams_list_sorted.put( x.getValue(), x.getKey()));

        // the output will be sorted by the reverse streaming count
        for (Map.Entry< String, Double> entry : streams_list_sorted.entrySet()) {

            String artist = entry.getKey();
            Double stream = entry.getValue();

            // the output will be consist only the top 20 artists
            context.write( NullWritable.get(), new Text(artist));

        }

    }

}


// Main method
public static void main (String[] args) throws Exception  {

    Configuration conf = new Configuration();

    // Job1
    Job job1 = Job.getInstance(conf, "Sum of total streams based on regions");
    job1.setJarByClass(Top10StreamingRegions.class);

    job1.setMapperClass(Spotify_Mapper.class);
    job1.setReducerClass(Spotify_Reducer.class);

    job1.setMapOutputKeyClass(Text.class);
    job1.setMapOutputValueClass(Text.class);

    job1.setOutputKeyClass(Text.class);
    job1.setOutputValueClass(DoubleWritable.class);

    FileInputFormat.addInputPath(job1, new Path(args[0]));
    FileOutputFormat.setOutputPath(job1, new Path(args[1]));


    boolean job1_success = job1.waitForCompletion(true);

    if(job1_success){

        // Job2
```

```
        Job job2 = Job.getInstance(conf, "Top 20 Global Music Artists based on streams");
        job2.setJarByClass(Top10StreamingRegions.class);

        job2.setMapperClass(Spotify_Top20Mapper.class);
        job2.setReducerClass(Spotify_Top20Reducer.class);

        job2.setMapOutputKeyClass(Text.class);
        job2.setMapOutputValueClass(DoubleWritable.class);

        job2.setOutputKeyClass(DoubleWritable.class);
        job2.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(job2, new Path(args[2]));
        FileOutputFormat.setOutputPath(job2, new Path(args[3]));

        job2.waitForCompletion(true);

    }

  }

}
```

Output:

The entire output file has been attached in the final report submission.



9. Average streaming numbers for every region

Code:

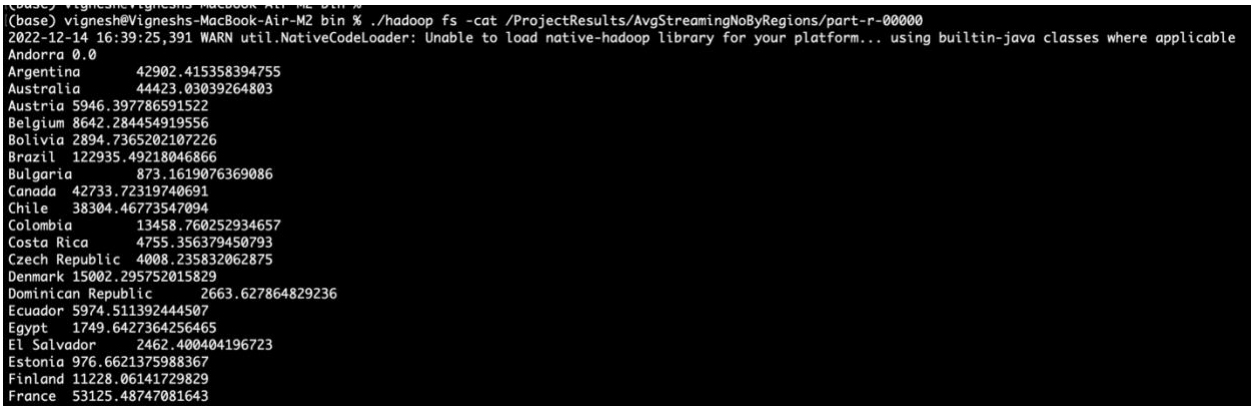
```
package com.mycompany.finalproject_spotifycharts;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 *
 * @author vignesh
 */

public class AverageStreamingNumbers {


    public static class Spotify_Mapper extends Mapper<Object, Text, Text, Text> {

        @Override
```

```
    public void map (Object key, Text value, Context context) throws IOException, InterruptedException {

        String[] tokens = value.toString().split(",", -1);
        Text region = new Text();
        region.set(tokens[5]);

        // passing the region as key and streams as a value
        context.write(region, new Text(new Text(tokens[8])));

    }

}

public static class Spotify_Reducer extends Reducer<Text, Text, Text, DoubleWritable> {

    private DoubleWritable avgObj = new DoubleWritable();

    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

        double sum = 0;
        int count = 0;

        for(Text val: values) {

            // is stream value is empty then adding the value as 0
            if(val.toString().equals("")){
                sum += 0;
            }
            else{
                double temp = Double.parseDouble(val.toString());
                sum += temp;
            }

            count += 1;

        }

        double avg = sum/count;
        avgObj.set(avg);

        context.write(key, avgObj);

    }

}
```

```
    // An average is not an associative operation
    // but if the count is output from the reducer with the sum of salary
    // these two values can be multiplied to preserve the sum for the final reduce phase


    public static void main (String[] args) throws Exception  {

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Average Streaming Numbers based on Regions");
        job.setJarByClass(AverageStreamingNumbers.class);

        job.setMapperClass(Spotify_Mapper.class);
        job.setReducerClass(Spotify_Reducer.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DoubleWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true)?0:1);

    }

}
```

Output:

The entire output file has been attached in the final report submission.



10. Distinct Spotify Artists in the top chart list

Code:

```
package com.mycompany.finalproject_spotifycharts;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 *
 * @author vignesh
 */
public class DistinctArtists {

    public static class Spotify_Mapper extends Mapper<Object, Text, Text, IntWritable> {

        private static final IntWritable one = new IntWritable(1);

        @Override
        public void map (Object key, Text value, Context context) throws IOException, InterruptedException {

            String[] tokens = value.toString().split(",", -1);
            Text artist = new Text();
            artist.set(tokens[3]);
```

```
            context.write(artist, one);
        }

    }

    public static class Spotify_Reducer extends Reducer<Text, IntWritable, Text, NullWritable> {


        @Override
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {

            int sum = 0;

            for(IntWritable val: values) {

                sum += val.get();
                if(sum == 1){
                    context.write(key, NullWritable.get());
                }

            }
        }
    }

    public static void main (String[] args) throws Exception  {

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Spotify Artists");
        job.setJarByClass(DistinctArtists.class);

        job.setMapperClass(Spotify_Mapper.class);
        job.setReducerClass(Spotify_Reducer.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(NullWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true)?0:1);

    }

}
```

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -head /ProjectResults/DistinctArtists/part-r-00000
2022-12-14 16:41:19,678 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

!!!
!!! Lea Lea
"""Elena Of Avalor"" Cast"
"""Weird Al"" Yankovic"
"""聲夢傳奇""學員"
"@LMK ~ LoveMusic""Kindness"
"Alexis y Fido Franco ""El Gorilla"""
"Andrew Lloyd Webber ""Cats"" 1981 Original London Cast"
"Andrius Mamontovas Džordana Butkutė Jūratė Miliauskaitė Vaidas Vinks Egidijus Sipavičius Saulius ""Samas"" Urbonavičius Laimontas ""Lainius"" Dinius Tomas Augulis"
"Anuel AA Cardi B Black Jonas Point Secreto ""El Famoso Biberon"" Liro Shaq"
"Arturo ""Zambo"" Cavero Oscar Avilés"
"Arturo ""Zambo"" Cavero"
"Atomic Otro Way J Alvarez Secreto ""El Famoso Biberon"""
"Baby Ranks Daddy Yankee Tonny Tun Tun Wisin & Yandel Héctor ""El Father"""
"Bad Bunny Héctor ""El Father"""
"Banda Selectiva de Angel Romero ""El Tacuache"""
"Beatrice Silver ""AG"" Orissaar"
"Black Jonas Point Secreto ""El Famoso Biberon"" Poeta Callejero Quimico Ultra Mega Albert06 El Veterano"
"Bobby ""Blue"" Bland"
"Bobby ""Boris"" Pickett The Crypt-Kickers"
"Bobby ""Boris"" Pickett"
"Bobby Cruz Ricardo ""Richie"" Ray"
"Boy
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

File contents

```
!!!
!!! Lea Lea
"""Elena Of Avalor"" Cast"
"""Weird Al"" Yankovic"
"""聲夢傳奇""學員"
"@LMK ~ LoveMusic""Kindness"
"Alexis y Fido Franco ""El Gorilla"""
"Andrew Lloyd Webber ""Cats"" 1981 Original London Cast"
"Andrius Mamontovas Džordana Butkutė Jūratė Miliauskaitė Vaidas Vinks Egidijus
Sipavičius Saulius ""Samas"" Urbonavičius Laimontas ""Lainius"" Dinius Tomas Augulis"
"Anuel AA Cardi B Black Jonas Point Secreto ""El Famoso Biberon"" Liro Shaq"
```

11. Sum of total streaming numbers for every region

Code:

package com.mycompany.finalproject_spotifycharts;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

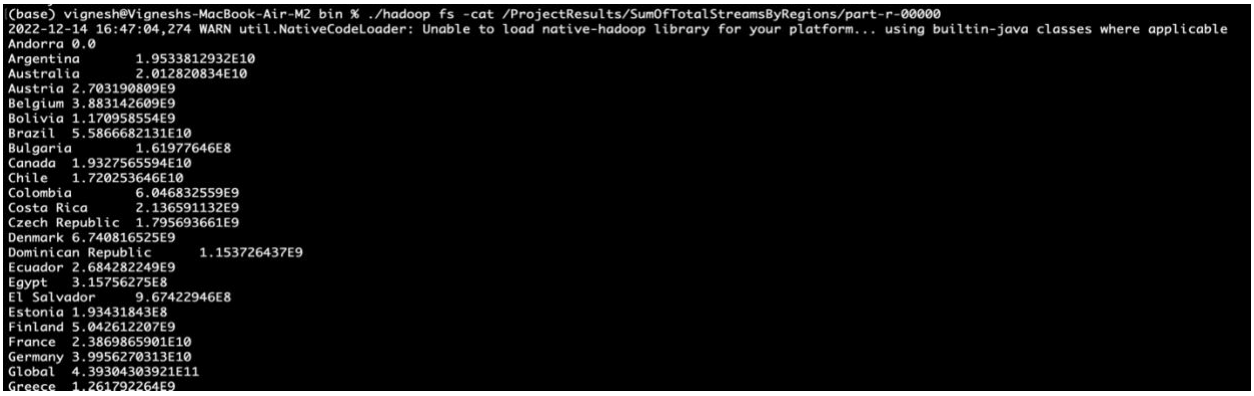
*/**

```
 *
 * @author vignesh
 */
public class SumOfStreamsByRegion {

   public static class Spotify_Mapper extends Mapper<Object, Text, Text, Text> {

      @Override
      public void map (Object key, Text value, Context context) throws IOException, InterruptedException {

         String[] tokens = value.toString().split(",", -1);
         Text region = new Text();
         region.set(tokens[5]);

         // passing the region as key and streams as value
         context.write(region, new Text(new Text(tokens[8])));

      }
   }

   public static class Spotify_Reducer extends Reducer<Text, Text, Text, DoubleWritable> {

      private DoubleWritable sumObj = new DoubleWritable();

      @Override
      public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

         double sum = 0;

         for(Text val: values) {

            // if the stream value is empty then adding the value as 0
            if(val.toString().equals("")){
               sum += 0;
            }
            else{
               double temp = Double.parseDouble(val.toString());
               sum += temp;
            }

         }

         sumObj.set(sum);
         context.write(key, sumObj);
      }

   }
```

```
    public static void main (String[] args) throws Exception  {

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Sum of total streams based on regions");
        job.setJarByClass(SumOfStreamsByRegion.class);

        job.setMapperClass(Spotify_Mapper.class);
        job.setReducerClass(Spotify_Reducer.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DoubleWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true)?0:1);

    }
}
```

Output:

The entire output file has been attached in the final report submission.



PIG Analysis:

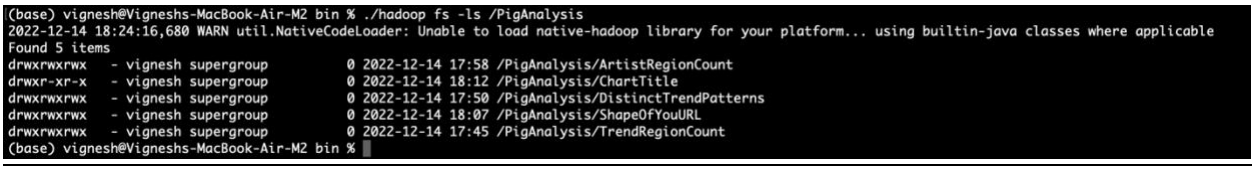
Running PIG in Map Reduce

pig -x mapreduce

Loading the Spotify dataset from Hadoop

```
spotify_charts = LOAD '/Project/spotify.csv'
                 USING PigStorage(',')
                 AS (title:chararray, rank:int, date:chararray, artist:chararray, url:chararray,
                 region:chararray, chart:chararray, trend:chararray, streams:int);
```

Stored all the Pig output files in the Hadoop PigAnalysis folder



Analysis:

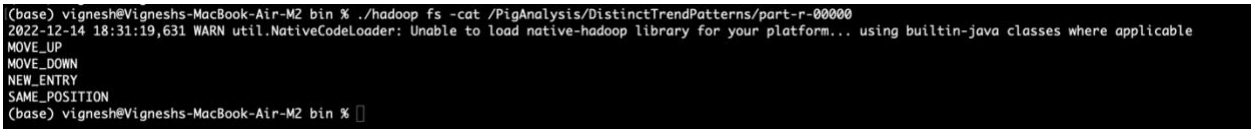
1. **Distinct trend patterns among the music titles**

Query:

```
title_trends = FOREACH spotify_charts GENERATE trend;
distinct_trends = DISTINCT title_trends;
DUMP distinct_trends;
STORE distinct_trends into '/PigAnalysis/DistinctTrendPatterns';
```

Output:

The entire output file has been attached in the final report submission.



File contents

```
MOVE_UP
MOVE_DOWN
NEW_ENTRY
SAME_POSITION
```

2. **Finding count of trend patterns based on the streaming regions**

Query:

trend_region = FOREACH spotify_charts GENERATE region, trend;
group_trend_region = GROUP trend_region BY (region, trend);
DESCRIBE group_trend_region;
order_trend_region = ORDER group_trend_region BY group;
ILLUSTRATE order_trend_region;
trend_count_region = FOREACH order_trend_region GENERATE group, COUNT(trend_region) as TrendPatternCounts;
DUMP trend_count_region;
STORE trend_count_region into '/PigAnalysis/TrendRegionCount';

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /PigAnalysis/TrendRegionCount/part-r-00000
2022-12-14 18:43:58,075 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
(Andorra,MOVE_DOWN)     26157
(Andorra,MOVE_UP)       27083
(Andorra,NEW_ENTRY)     13492
(Andorra,SAME_POSITION) 12860
(Argentina,MOVE_DOWN)   192261
(Argentina,MOVE_UP)     164750
(Argentina,NEW_ENTRY)   27652
(Argentina,SAME_POSITION)       70645
(Australia,MOVE_DOWN)   193299
(Australia,MOVE_UP)     167338
(Australia,NEW_ENTRY)   29099
(Australia,SAME_POSITION)       63367
(Austria,MOVE_DOWN)     199124
(Austria,MOVE_UP)       173195
(Austria,NEW_ENTRY)     37004
(Austria,SAME_POSITION) 45270
(Belgium,MOVE_DOWN)     193015
(Belgium,MOVE_UP)       172409
(Belgium,NEW_ENTRY)     37511
(Belgium,SAME_POSITION) 46384
```

File contents

```
(Andorra,MOVE_DOWN)     26157
(Andorra,MOVE_UP)       27083
(Andorra,NEW_ENTRY)     13492
(Andorra,SAME_POSITION) 12860
(Argentina,MOVE_DOWN)   192261
(Argentina,MOVE_UP)     164750
(Argentina,NEW_ENTRY)   27652
(Argentina,SAME_POSITION)       70645
(Australia,MOVE_DOWN)   193299
(Australia,MOVE_UP)     167338
(Australia,NEW_ENTRY)   29099
(Australia,SAME_POSITION)       63367
```

3. **Total number of artists in every streaming region**

Query:

artist_region = FOREACH spotify_charts GENERATE artist, region;
group_artist_region = GROUP artist_region BY region;
DESCRIBE group_artist_region;
artist_count_region = FOREACH group_artist_region GENERATE group, COUNT(artist_region) as TotalArtists;
artist_count_region_order = ORDER artist_count_region BY TotalArtists DESC;
ILLUSTRATE artist_count_region_order;
DUMP artist_count_region_order;
STORE artist_count_region_order into '/PigAnalysis/ArtistRegionCount';

Output:

The entire output file has been attached in the final report submission.

4. **Top ranked title in the United States on 1st June 2021**

Query:

title_filter = FILTER spotify_charts BY region == 'United States' AND date == '2021-06-01' and rank == 1;
DESCRIBE title_filter;
chart_title = FOREACH title_filter GENERATE title;
DUMP chart_title;
STORE chart_title into '/PigAnalysis/ChartTitle';

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /PigAnalysis/ChartTitle/part-m-00000
2022-12-14 18:47:18,077 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Castaways
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

File contents

Castaways

5. **Getting URL of the "Shape of You" music title for United States region**

Query:

song_filter = FILTER spotify_charts BY region == 'United States' AND title == 'Shape of You';
DESCRIBE song_filter;
song_url = FOREACH song_filter GENERATE url;
url_shapeofyou = DISTINCT song_url;
DUMP url_shapeofyou;
STORE url_shapeofyou into '/PigAnalysis/ShapeOfYouURL';

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /PigAnalysis/ShapeOfYouURL/part-r-00000
2022-12-14 18:48:39,865 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
https://open.spotify.com/track/0FE9t6xYkqWXU2ahLh6D8X
https://open.spotify.com/track/7qiZfU4dY1lWllzX7mPBI3
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

File contents

https://open.spotify.com/track/0FE9t6xYkqWXU2ahLh6D8X
https://open.spotify.com/track/7qiZfU4dY1lWllzX7mPBI3

HIVE Analysis:

Created Spotify database and loaded the Spotify dataset into table

CREATE DATABASE spotify;
SHOW databases;
USE spotify;

CREATE TABLE IF NOT EXISTS spotify.charts (
title string,
rank int,
chartdate date,
artist string,
url string,
region string,
chart string,
trend string,
streams int)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

LOAD DATA INPATH 'Project/spotify.csv' INTO TABLE spotify.charts;

SHOW tables;

DESCRIBE spotify.charts;

```
hive> SHOW databases;
OK
default
spotify
Time taken: 0.105 seconds, Fetched: 2 row(s)
hive> USE spotify;
OK
Time taken: 0.07 seconds
hive> SHOW tables;
OK
charts
Time taken: 0.13 seconds, Fetched: 1 row(s)
hive> DESCRIBE spotify.charts;
OK
title                   string
rank                    int
chartdate               date
artist                  string
url                     string
region                  string
chart                   string
trend                   string
streams                 int
Time taken: 0.639 seconds, Fetched: 9 row(s)
hive>
```

Analysis:

1. **Count of top 200 music titles based on the streaming regions**

Query:

INSERT OVERWRITE DIRECTORY '/HiveAnalysis/Top200CountsByRegion'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
SELECT region, count(title) FROM charts where chart = 'top200' GROUP BY region ORDER BY region;

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /HiveAnalysis/Top200CountsByRegion/000000_0
2022-12-14 21:39:49,298 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Argentina       364385
Australia       362189
Austria 363925
Belgium 358390
Bolivia 313849
Brazil  364516
Bulgaria        94842
Canada  361390
Chile   358185
Colombia        358384
Costa Rica      358384
Czech Republic  357276
Denmark 358389
Dominican Republic      342318
Ecuador 358384
Egypt   123769
```

File contents

```
Argentina 364385
Australia 362189
Austria 363925
Belgium 358390
Bolivia 313849
Brazil 364516
Bulgaria 94842
Canada 361390
```

2. **Count of viral music hits based on the streaming regions**

Query:

INSERT OVERWRITE DIRECTORY '/HiveAnalysis/ViralCountsByRegion'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
SELECT region, count(title) FROM charts where chart = 'viral50' GROUP BY region ORDER BY region;

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /HiveAnalysis/ViralCountsByRegion/000000_0
2022-12-14 21:41:01,890 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Andorra 79592
Argentina       90923
Australia       90914
Austria 90668
Belgium 90929
Bolivia 90664
Brazil  89923
Bulgaria        90665
Canada  90889
Chile   90915
Colombia        90902
Costa Rica      90918
Czech Republic  90725
Denmark 90930
```

File contents

```
Andorra  79592
Argentina 90923
Australia  90914
Austria   90668
Belgium   90929
Bolivia   90664
Brazil    89923
Bulgaria  90665
```

3. **Top 10 streaming countries based on streams**

Query:

INSERT OVERWRITE DIRECTORY '/HiveAnalysis/Top10StreamingRegions'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
SELECT region, sum(streams) as total FROM charts GROUP BY region ORDER BY total DESC LIMIT 10;

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /HiveAnalysis/Top10StreamingRegions/000000_0
2022-12-14 21:41:55,762 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Global  439304303921
United States   142132086404
Brazil  55866682131
Mexico  45595912533
Germany 39956270313
United Kingdom  36408800049
Spain   27062199940
Italy   25591134079
France  23869865901
Australia       20128208340
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

File contents

```
Global 439304303921
United States 142132086404
Brazil 55866682131
Mexico 45595912533
Germany 39956270313
United Kingdom 36408800049
Spain 27062199940
Italy 25591134079
France 23869865901
Australia 20128208340
```

4. **Top 20 global music artists based on streams**

Query:

INSERT OVERWRITE DIRECTORY '/HiveAnalysis/Top20GlobalArtists'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
SELECT artist FROM (select artist, sum(streams) AS total FROM charts GROUP BY artist) temp ORDER BY temp.total DESC LIMIT 20;

Output:

The entire output file has been attached in the final report submission.

File contents

Ed Sheeran
Post Malone
Drake
Billie Eilish
Ariana Grande
Dua Lipa
Juice WRLD
XXXTENTACION
The Weeknd
Olivia Rodrigo
Travis Scott
Justin Bieber
Bad Bunny
Taylor Swift
Lil Nas X
Imagine Dragons
Doja Cat
Harry Styles
Lewis Capaldi
BTS

5. **Distinct count values of Regions, Artists, and Titles**

Query:

INSERT OVERWRITE DIRECTORY '/HiveAnalysis/CountOfRegionsArtistsTitles'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
SELECT count(distinct region) AS TotalRegions, count(distinct artist) AS TotalArtists, count(distinct title) AS TotalMusicTitles FROM charts;

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /HiveAnalysis/CountOfRegionsArtistsTitles/000000_0
2022-12-14 21:44:09,577 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
70      96156   164722
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

File contents

70 96156 164722

6. **Top 5 regions with high average streaming numbers**

Query:

INSERT OVERWRITE DIRECTORY '/HiveAnalysis/Top5HighAvgStreamRegions'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
SELECT region, round(avg(streams),2) AS total FROM charts GROUP BY region ORDER BY total DESC LIMIT 5;

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /HiveAnalysis/Top5HighAvgStreamRegions/000000_0
2022-12-14 21:45:09,568 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Global  1212937.98
United States   390275.48
Brazil  153262.63
Mexico  127240.97
Germany 111789.56
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

File contents

```
Global  1212937.98
United States   390275.48
Brazil  153262.63
Mexico  127240.97
Germany 111789.56
```

7. **Total streaming numbers based for every year (2017-2021)**

Query:

INSERT OVERWRITE DIRECTORY '/HiveAnalysis/TotalStreamsEveryYear'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
SELECT date_format(chartdate,'yyyy'), sum(streams) FROM charts GROUP BY date_format(chartdate,'yyyy')
ORDER BY date_format(chartdate,'yyyy');

Output:

The entire output file has been attached in the final report submission.

```
(base) vignesh@Vigneshs-MacBook-Air-M2 bin % ./hadoop fs -cat /HiveAnalysis/TotalStreamsEveryYear/000000_0
2022-12-14 21:46:33,809 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2017    174455871758
2018    205228560775
2019    233465561067
2020    253395685784
2021    256469446291
(base) vignesh@Vigneshs-MacBook-Air-M2 bin %
```

File contents

```
2017    174455871758
2018    205228560775
2019    233465561067
2020    253395685784
2021    256469446291
```