Program Logic:

The Map Reduce program here should result in 31 files for each day of the month.
The output file will have key,value pairs.
The key here will be a composite key - a combination of a song id + hour at which it is streamed.
The value here will be the total count of how many times a song id was streamed at certain hour of the day.

Based on the output files generated from running the Map Reduce program , I inferred I can arrive at the desired result with below steps:

1. For predicting trending songs for a certain day (nth), I should take into account the output file from the previous day(n-1 th).
2. For hours ranging from 12 pm - 12 am on a certain day, the weight-age assigned for the songs should be more as these
hours can give the actual latest trend.
3. If a song has been played less than 5 times in a certain hour on a day, I should not be assigning any weightage to such a song.
4. From the output file of Map Reduce on certain day, certain operations need to be done to arrive at prediction of top 100 songs on its next day.
5. Hence, for a single song id played at different hours, I need the sum of total number of times it was played.
This means I need to Group the records on SongID -> Get the count of times it was played in all the hours of (n-1)th day.
6. I would then sort the records in Descending order of the sum count (this already has included my assigned weights based on different hours)
7. I would then pick the top 100 records -> This will be my trending song list for nth day.

I have followed below approach:

a) Download the MR output raw file for required dates on your machine from AWS s3 bucket for further processing. Save it as .txt file

b) The file showed records like:

```
 Allegiant&quot:18     :1
 Don&#039:11     :5
 Don&#039:12     :2
 Don&#039:14     :2
 Don&#039:16     :3
 Don&#039:17     :1
 Don&#039:19     :1
```

c) On Notepad++ , Replace '\t' with ''
For e.g. - earlier file looked like - Allegiant&quot:18 :1
Now it looks like - Allegiant&quot:18:1

d) On Notepad++ , Replace ':' with '\t'
For e.g. - earlier file looked like - Don&#039:4:2
Now it looks like - Don&#039   4   2

e) Load this data into a MySQL table for e.g. -

```
create table Txtdata (

songid varchar(100),
hour int,
songcount int
);

LOAD DATA LOCAL INFILE 'C:/Users/Deepika
Chauhan/Desktop/SaavnProjectOutput/part-00025.txt' INTO TABLE txtdata
LINES TERMINATED BY '\n';
```

f) Check if you have all the records in your table.

```
select * from txtdata
```

g) Now, group by songid and take the count for each group. Dump this data into another table because you need to process it further.

```
create table txtdata1 (

songid varchar(100),
songcount int);

INSERT INTO txtdata1 (songid, songcount) SELECT txtdata.songid ,
SUM(txtdata.songcount) from txtdata GROUP BY txtdata.songid
```

h) Check if you have all the data grouped by song id and the respective count using:

```
select * from txtdata1
```

i) Now, order this data in descending order based on count , and return 100 Distinct records -> songids

```
select DISTINCT(songid) from txtdata1 ORDER BY songcount DESC LIMIT 100
```

j) After you are done, export the result to a text file, using export operation on MySQL.

k) Drop the tables created and create them newly when you are working on next file.

```
drop table txtdata;
drop table txtdata1;
```

l) Repeat setps - e) to k) for all 7 days - 25th to 31st of December.

POINT TO NOTE::

For output files generated using Sample data, I could combine steps- g , h , i into a single operation as:

```
select songid , SUM(songcount) as temp from txtdata group by songid ORDER
BY temp DESC LIMIT 100;
```

However, the output files generated from actual dataset were extremely huge and this query did not work.
To solve this issue, I came up with alternative - dump the resultant from group by - count in temporary table
and then apply further operations - sort and limit.

Running of Jar file:

The Jar file created from Java Project was run on AWS EC2 instance.
Steps followed from :

1) How to run Java program on EC2 document in the Map Reduce session.
2) The input and output data to be extracted from s3 in AWS using project guidelines.

Please Note: Kindly open all .txt files in Notepad++