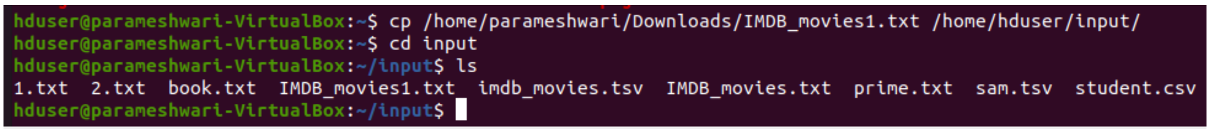
18CSC403 – PRACTICAL TECHNIQUES FOR BIG DATA PROCESSING FINAL PROJECT WORK – (28-11-2021)

- Parameshwari S (CB.SC.I5DAS18026)

MapReduce, Pig, Hive, Mongo and Cassandra queries

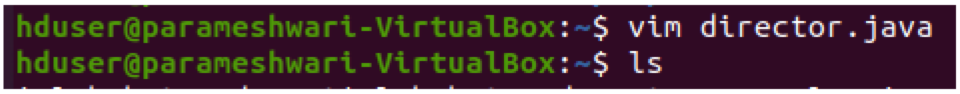
1. **Directors who released the more number of films. (MapReduce)**

Step 1 : Copying the data into hadoop

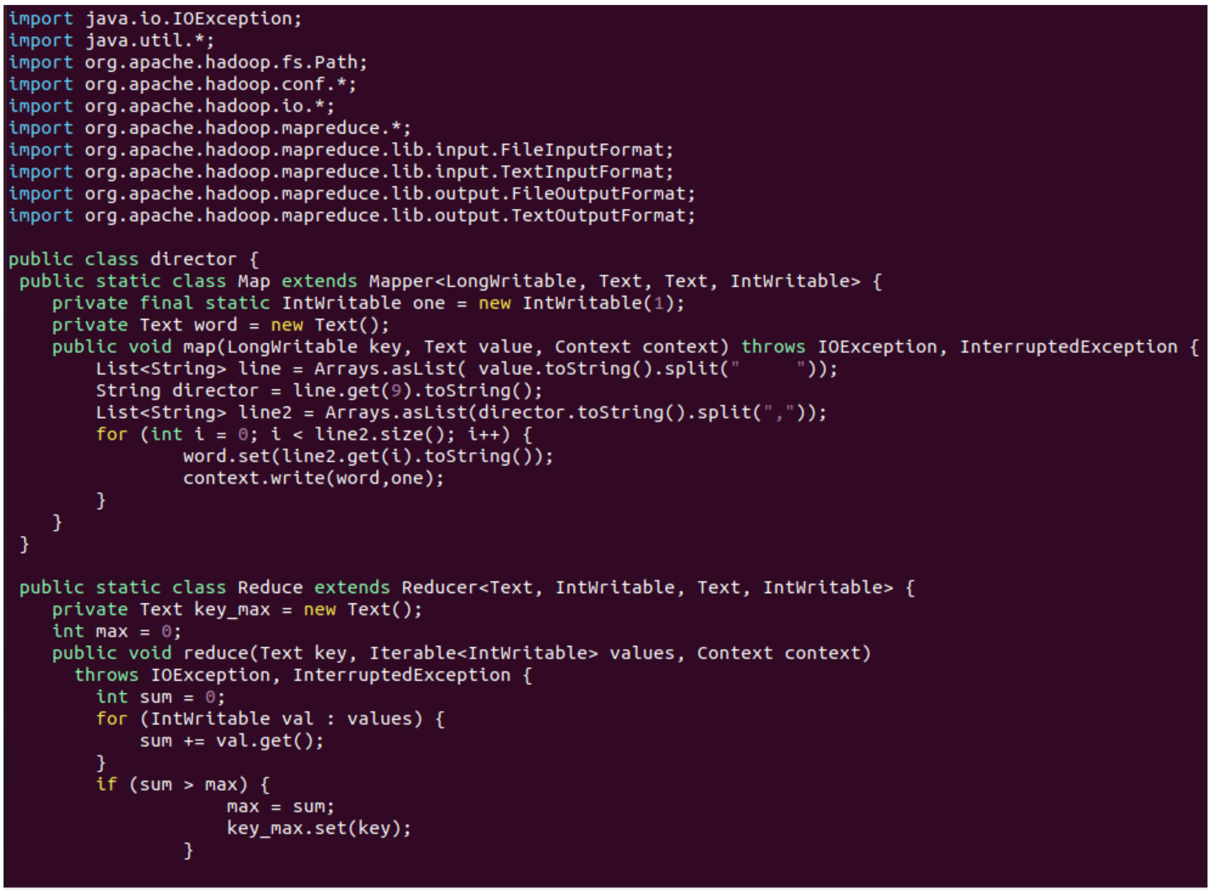


```
hduser@parameshwari-VirtualBox:~$ cp /home/parameshwari/Downloads/IMDB_movies1.txt /home/hduser/input/
hduser@parameshwari-VirtualBox:~$ cd input
hduser@parameshwari-VirtualBox:~/input$ ls
1.txt  2.txt  book.txt  IMDB_movies1.txt  imdb_movies.tsv  IMDB_movies.txt  prime.txt  sam.tsv  student.csv
hduser@parameshwari-VirtualBox:~/input$ 
```

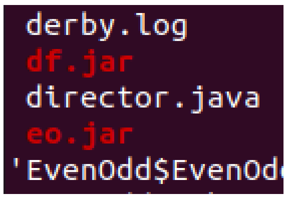
Step 2 : Creating the java program



```
hduser@parameshwari-VirtualBox:~$ vim director.java
hduser@parameshwari-VirtualBox:~$ ls
```

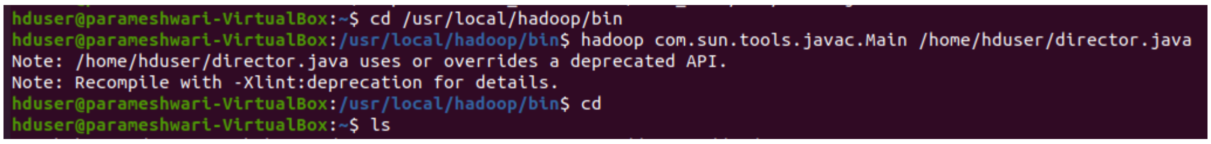


```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class director {
 public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        List<String> line = Arrays.asList( value.toString().split("     "));
        String director = line.get(9).toString();
        List<String> line2 = Arrays.asList(director.toString().split(","));
        for (int i = 0; i < line2.size(); i++) {
                word.set(line2.get(i).toString());
                context.write(word,one);
        }
    }
 }

 public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
    private Text key_max = new Text();
    int max = 0;
    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 > max) {
                    max = sum;
                    key_max.set(key);
                }
```

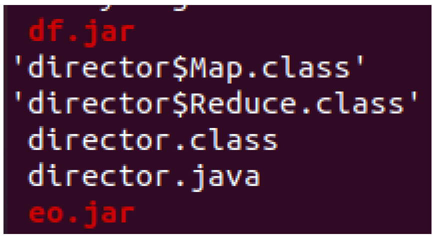


```
derby.log
df.jar
director.java
eo.jar
'EvenOdd$EvenOd
```

Step 3 : Create the class files

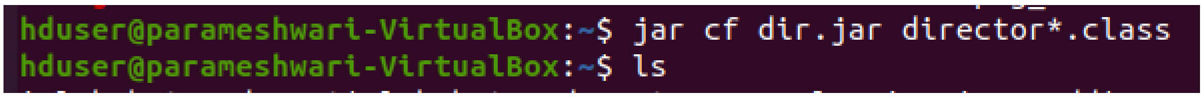


```
hduser@parameshwari-VirtualBox:~$ cd /usr/local/hadoop/bin
hduser@parameshwari-VirtualBox:/usr/local/hadoop/bin$ hadoop com.sun.tools.javac.Main /home/hduser/director.java
Note: /home/hduser/director.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
hduser@parameshwari-VirtualBox:/usr/local/hadoop/bin$ cd
hduser@parameshwari-VirtualBox:~$ ls
```

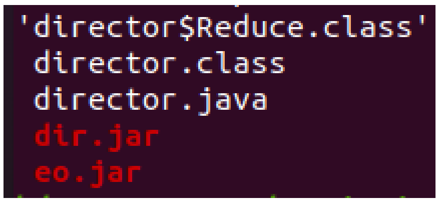


```
 df.jar
'director$Map.class'
'director$Reduce.class'
 director.class
 director.java
 eo.jar
```

Step 4 : Create the jar file

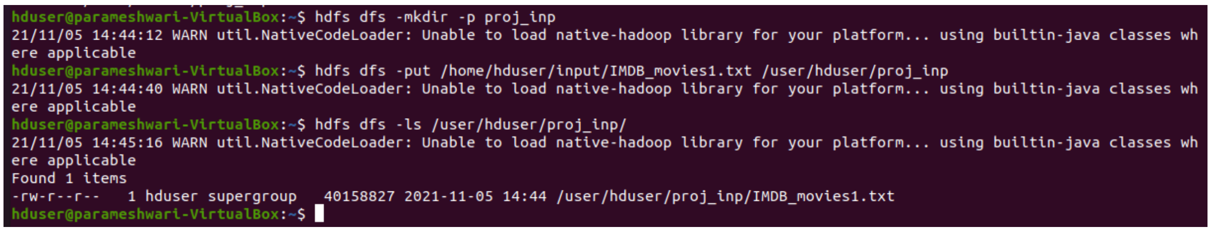


```
hduser@parameshwari-VirtualBox:~$ jar cf dir.jar director*.class
hduser@parameshwari-VirtualBox:~$ ls
```



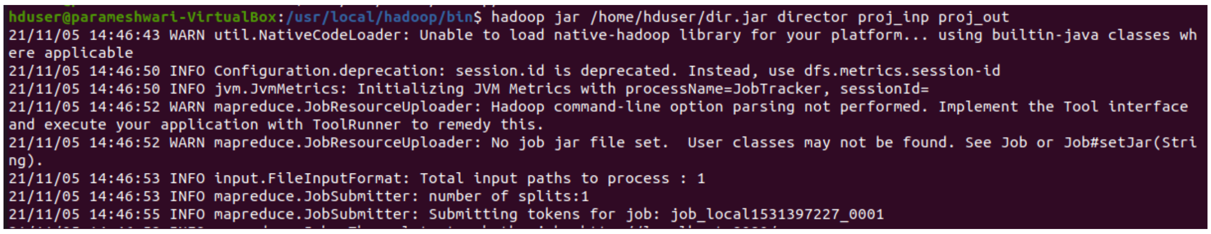
```
'director$Reduce.class'
 director.class
 director.java
 dir.jar
 eo.jar
```

Step 5 : Make an input directory – **input** in HDFS and add the text files to the directory

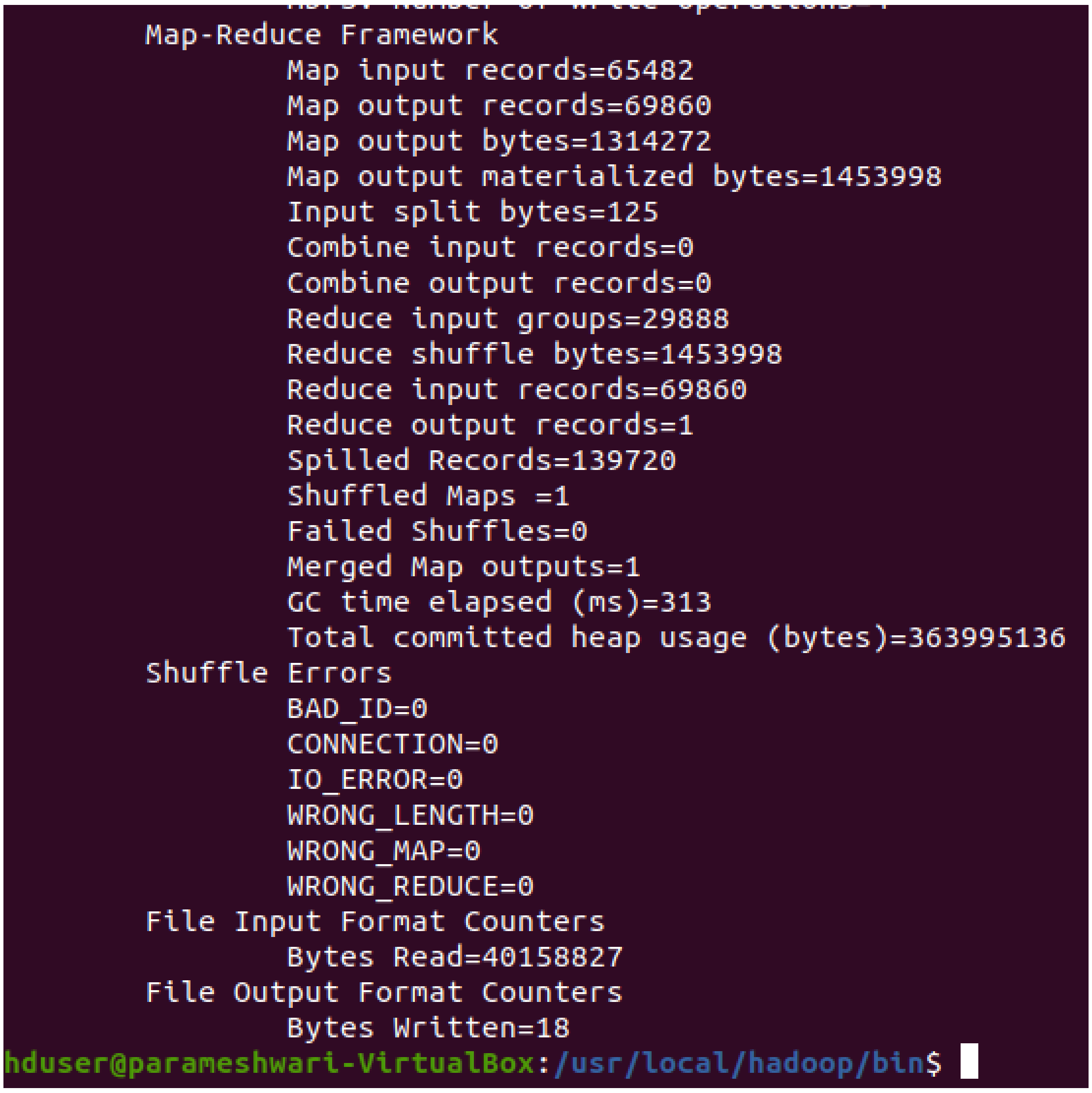


```
hduser@parameshwari-VirtualBox:~$ hdfs dfs -mkdir -p proj_inp
21/11/05 14:44:12 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
ere applicable
hduser@parameshwari-VirtualBox:~$ hdfs dfs -put /home/hduser/input/IMDB_movies1.txt /user/hduser/proj_inp
21/11/05 14:44:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
ere applicable
hduser@parameshwari-VirtualBox:~$ hdfs dfs -ls /user/hduser/proj_inp/
21/11/05 14:45:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
ere applicable
Found 1 items
-rw-r--r--   1 hduser supergroup   40158827 2021-11-05 14:44 /user/hduser/proj_inp/IMDB_movies1.txt
hduser@parameshwari-VirtualBox:~$ 
```

Step 6 : Run the program

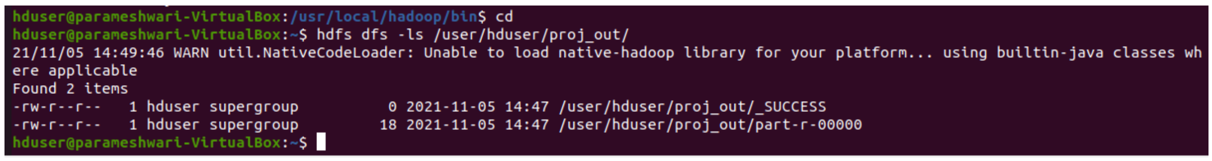


```
hduser@parameshwari-VirtualBox:/usr/local/hadoop/bin$ hadoop jar /home/hduser/dir.jar director proj_inp proj_out
21/11/05 14:46:43 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
ere applicable
21/11/05 14:46:50 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
21/11/05 14:46:50 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
21/11/05 14:46:52 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface
and execute your application with ToolRunner to remedy this.
21/11/05 14:46:52 WARN mapreduce.JobResourceUploader: No job jar file set.  User classes may not be found. See Job or Job#setJar(Stri
ng).
21/11/05 14:46:53 INFO input.FileInputFormat: Total input paths to process : 1
21/11/05 14:46:53 INFO mapreduce.JobSubmitter: number of splits:1
21/11/05 14:46:55 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1531397227_0001
```

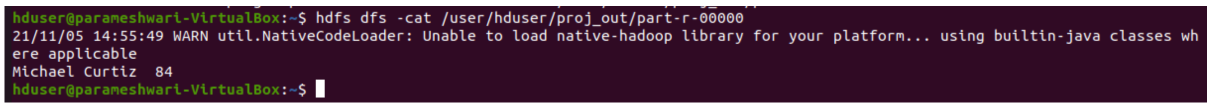


```
        Map-Reduce Framework
                Map input records=65482
                Map output records=69860
                Map output bytes=1314272
                Map output materialized bytes=1453998
                Input split bytes=125
                Combine input records=0
                Combine output records=0
                Reduce input groups=29888
                Reduce shuffle bytes=1453998
                Reduce input records=69860
                Reduce output records=1
                Spilled Records=139720
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=313
                Total committed heap usage (bytes)=363995136
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=40158827
        File Output Format Counters
                Bytes Written=18
hduser@parameshwari-VirtualBox:/usr/local/hadoop/bin$
```

Step 7 : Check the output



```
hduser@parameshwari-VirtualBox:/usr/local/hadoop/bin$ cd
hduser@parameshwari-VirtualBox:~$ hdfs dfs -ls /user/hduser/proj_out/
21/11/05 14:49:46 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
ere applicable
Found 2 items
-rw-r--r--   1 hduser supergroup          0 2021-11-05 14:47 /user/hduser/proj_out/_SUCCESS
-rw-r--r--   1 hduser supergroup         18 2021-11-05 14:47 /user/hduser/proj_out/part-r-00000
hduser@parameshwari-VirtualBox:~$
```



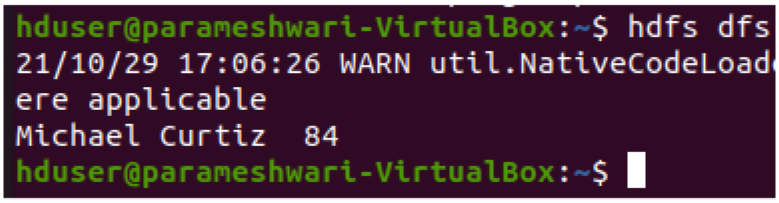
```
hduser@parameshwari-VirtualBox:~$ hdfs dfs -cat /user/hduser/proj_out/part-r-00000
21/11/05 14:55:49 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
ere applicable
Michael Curtiz  84
hduser@parameshwari-VirtualBox:~$
```

<u>CODE</u> –

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class director {
 public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
        List<String> line = Arrays.asList( value.toString().split("          "));
            String director = line.get(9).toString().;
            List<String> line2 = Arrays.asList(director.toString().split(","));
            for (int i = 0; i < line2.size(); i++) {
                    word.set(line2.get(i).toString());
                    context.write(word,one);
            }
    }
 }

 public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
   private Text key_max = new Text();
   int max = 0;
   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 > max) {
          max = sum;
          key_max.set(key);
       }
    }
  @Override
  protected void cleanup(Context context) throws IOException, InterruptedException {
     context.write(key_max, new IntWritable(max));
  }
 }

 public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = new Job(conf, "director");
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    job.setMapperClass(Map.class);
```

```
        job.setReducerClass(Reduce.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.waitForCompletion(true);
      }
     }
```

OUTPUT –

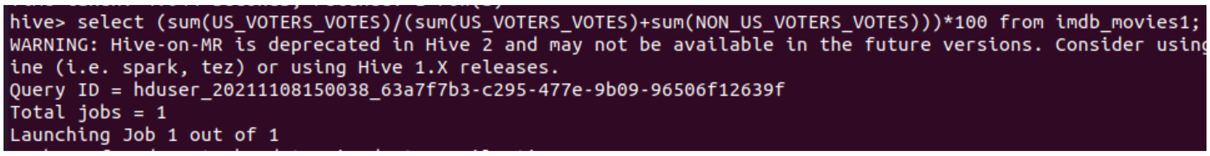


```
hduser@parameshwari-VirtualBox:~$ hdfs dfs
21/10/29 17:06:26 WARN util.NativeCodeLoad
ere applicable
Michael Curtiz  84
hduser@parameshwari-VirtualBox:~$
```

2. **Percentage of US voters.** **(hive)**

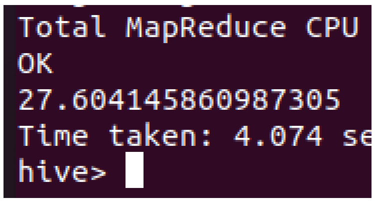
CODE –

```
select
(sum(US_VOTERS_VOTES)/(sum(US_VOTERS_VOTES)+sum(NON_US_VOTERS_VOTES)))*
100  from imdb_movies1;
```



```
hive> select (sum(US_VOTERS_VOTES)/(sum(US_VOTERS_VOTES)+sum(NON_US_VOTERS_VOTES)))*100 from imdb_movies1;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using
ine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20211108150038_63a7f7b3-c295-477e-9b09-96506f12639f
Total jobs = 1
Launching Job 1 out of 1
```

OUTPUT –

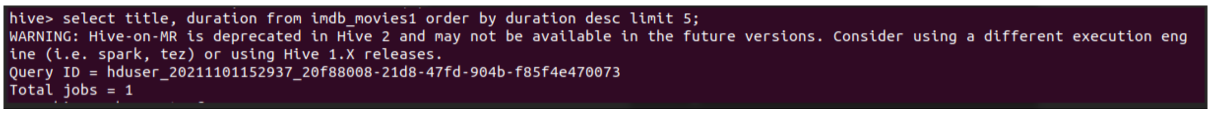


```
Total MapReduce CPU
OK
27.604145860987305
Time taken: 4.074 se
hive>
```

3. **Top 5 movies that has the longest duration.** **(hive)**

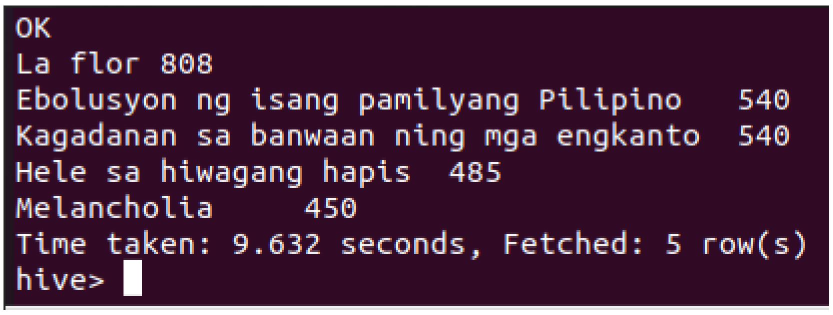
CODE –

```
select title, duration from imdb_movies1 order by duration desc limit 5;
```



```
hive> select title, duration from imdb_movies1 order by duration desc limit 5;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution eng
ine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20211101152937_20f88008-21d8-47fd-904b-f85f4e470073
Total jobs = 1
```

<u>OUTPUT</u> –

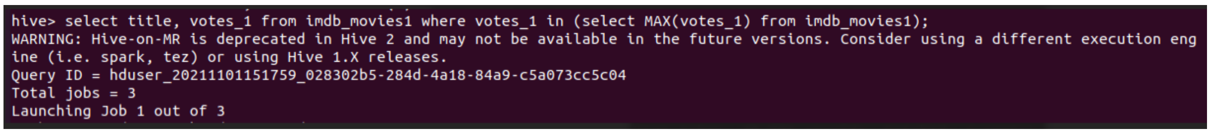


```
OK
La flor 808
Ebolusyon ng isang pamilyang Pilipino   540
Kagadanan sa banwaan ning mga engkanto  540
Hele sa hiwagang hapis  485
Melancholia     450
Time taken: 9.632 seconds, Fetched: 5 row(s)
hive>
```

4. **Movies that got the lowest voting. (hive)**

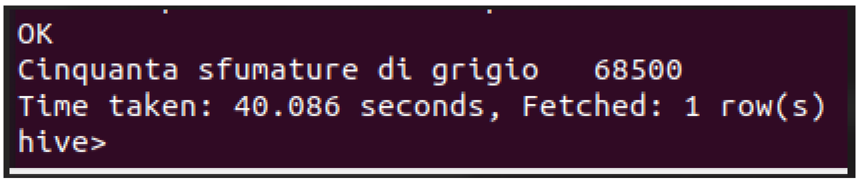
<u>CODE</u> –

select title, votes_1 from imdb_movies1 where votes_1 in (select MAX(votes_1) from imdb_movies1);



```
hive> select title, votes_1 from imdb_movies1 where votes_1 in (select MAX(votes_1) from imdb_movies1);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution eng
ine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20211101151759_028302b5-284d-4a18-84a9-c5a073cc5c04
Total jobs = 3
Launching Job 1 out of 3
```

<u>OUTPUT</u> –

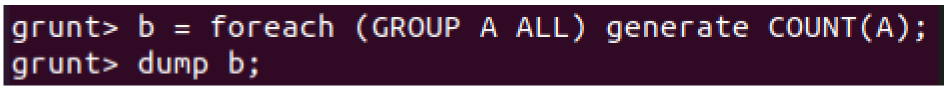


```
OK
Cinquanta sfumature di grigio   68500
Time taken: 40.086 seconds, Fetched: 1 row(s)
hive>
```

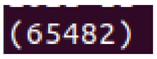
5. **Count of movies dubbed in english (Pig)**

<u>CODE</u> –

> b= for each (Group A ALL) generate COUNT (A);
> dump b;

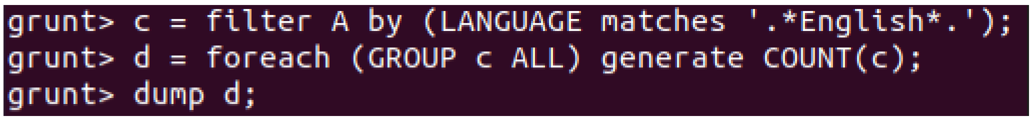


```
grunt> b = foreach (GROUP A ALL) generate COUNT(A);
grunt> dump b;
```



```
(65482)
```

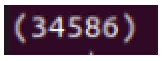
> c= filter A by (LANGUAGE matches '.*English*.');
>d= foreach (GROUP c ALL) generate COUNT(c);
> dump d;



```
grunt> c = filter A by (LANGUAGE matches '.*English*.');
grunt> d = foreach (GROUP c ALL) generate COUNT(c);
grunt> dump d;
```

OUTPUT –

>34586 english movies

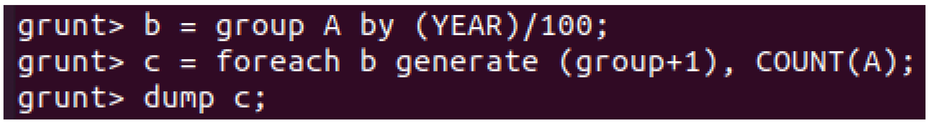


```
(34586)
```

6. **Count of movies in each century (Pig)**

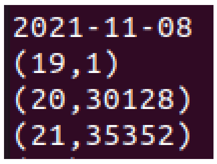
CODE –

> b= group A by (YEAR)/100;
> c= foreach b generate (group+1), COUNT(A);
> dump c;



```
grunt> b = group A by (YEAR)/100;
grunt> c = foreach b generate (group+1), COUNT(A);
grunt> dump c;
```

OUTPUT –

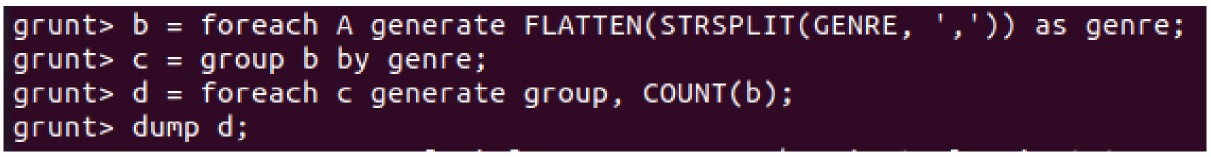


```
2021-11-08
(19,1)
(20,30128)
(21,35352)
```

7. **Number of movies in different genre (Pig)**

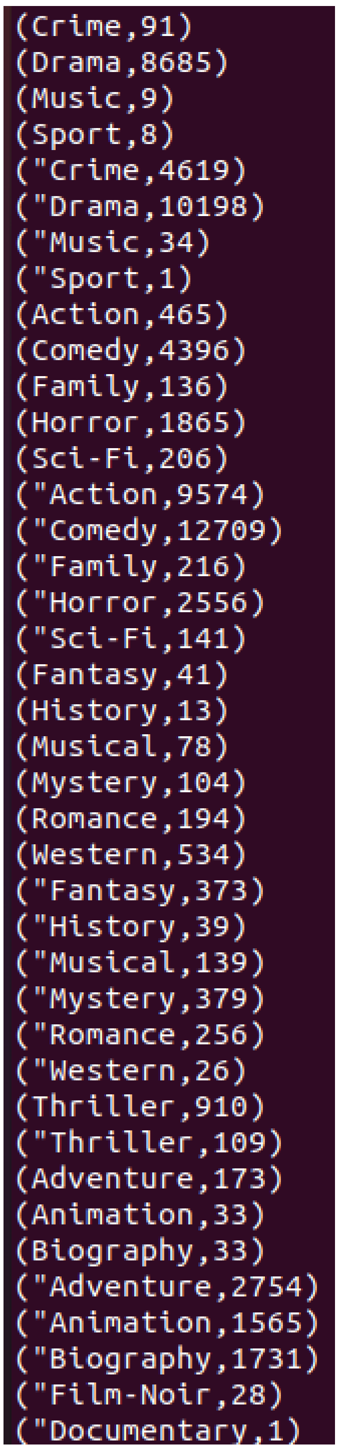
CODE –

> b=foreach A generate FLATTEN(STRSPLIT(GENRE, ',')) as genre;
> c= group b by genre;
> d= foreach c generate group, COUNT(b);
> dump d;



```
grunt> b = foreach A generate FLATTEN(STRSPLIT(GENRE, ',')) as genre;
grunt> c = group b by genre;
grunt> d = foreach c generate group, COUNT(b);
grunt> dump d;
```

<u>OUTPUT</u> –



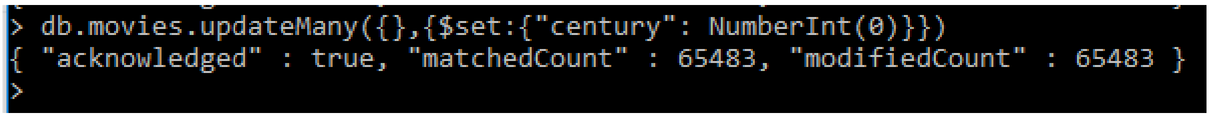
```
(Crime,91)
(Drama,8685)
(Music,9)
(Sport,8)
("Crime,4619)
("Drama,10198)
("Music,34)
("Sport,1)
(Action,465)
(Comedy,4396)
(Family,136)
(Horror,1865)
(Sci-Fi,206)
("Action,9574)
("Comedy,12709)
("Family,216)
("Horror,2556)
("Sci-Fi,141)
(Fantasy,41)
(History,13)
(Musical,78)
(Mystery,104)
(Romance,194)
(Western,534)
("Fantasy,373)
("History,39)
("Musical,139)
("Mystery,379)
("Romance,256)
("Western,26)
(Thriller,910)
("Thriller,109)
(Adventure,173)
(Animation,33)
(Biography,33)
("Adventure,2754)
("Animation,1565)
("Biography,1731)
("Film-Noir,28)
("Documentary,1)
```

8. **Adding a new column 'century' based on year (Mongo)**

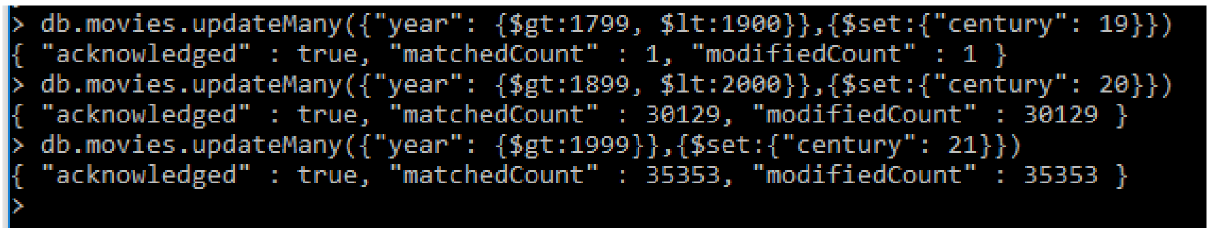
CODE –

- db.movies.updateMany({}, {$set: {“century”: NumberInt(0)}})
- db.movies.updateMany({“year”: {$gt:1799, $lt:1900}}, {$set: {“century”: 19}})
- db.movies.updateMany({“year”: {$gt:1899, $lt:2000}}, {$set: {“century”: 20}})
- db.movies.updateMany({“year”: {$gt:1999}}, {$set: {“century”: 21}})

OUTPUT –



```
> db.movies.updateMany({},{$set:{"century": NumberInt(0)}})
{ "acknowledged" : true, "matchedCount" : 65483, "modifiedCount" : 65483 }
>
```



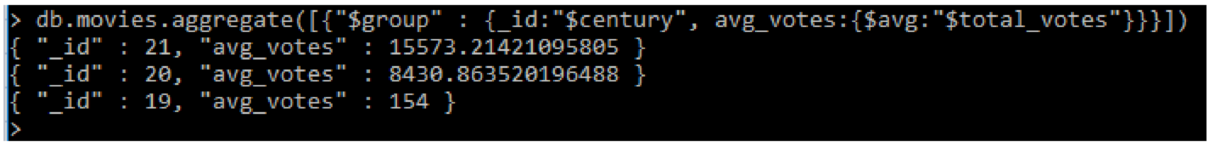
```
> db.movies.updateMany({"year": {$gt:1799, $lt:1900}},{$set:{"century": 19}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.movies.updateMany({"year": {$gt:1899, $lt:2000}},{$set:{"century": 20}})
{ "acknowledged" : true, "matchedCount" : 30129, "modifiedCount" : 30129 }
> db.movies.updateMany({"year": {$gt:1999}},{$set:{"century": 21}})
{ "acknowledged" : true, "matchedCount" : 35353, "modifiedCount" : 35353 }
>
```

9. **Average votes given for a movies in each century (Mongo)**

CODE –

db.movies.aggregate([{“$group”: {_id: “$century”, avg_votes: {$avg: “$total_votes”}}}])

OUTPUT –



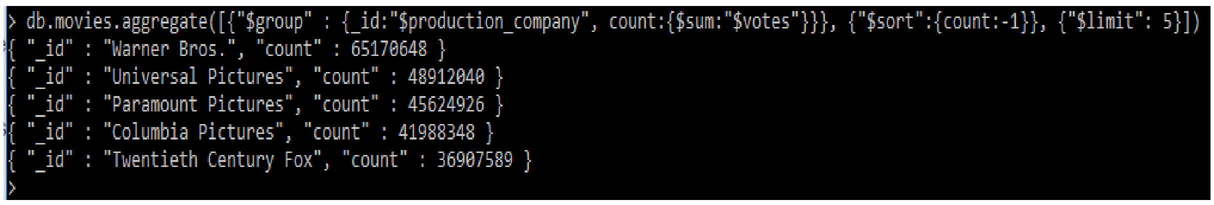
```
> db.movies.aggregate([{"$group" : {_id:"$century", avg_votes:{$avg:"$total_votes"}}}])
{ "_id" : 21, "avg_votes" : 15573.21421095805 }
{ "_id" : 20, "avg_votes" : 8430.863520196488 }
{ "_id" : 19, "avg_votes" : 154 }
>
```

10. **Top 5 best production companies (Mongo)**

CODE –

db.movies.aggregate([{“$group”: {_id: “$production_company”, count: {“$sum: “$votes”}}}, {“$sort”: {count: -1}}, {“$limit”: 5}])

OUTPUT –



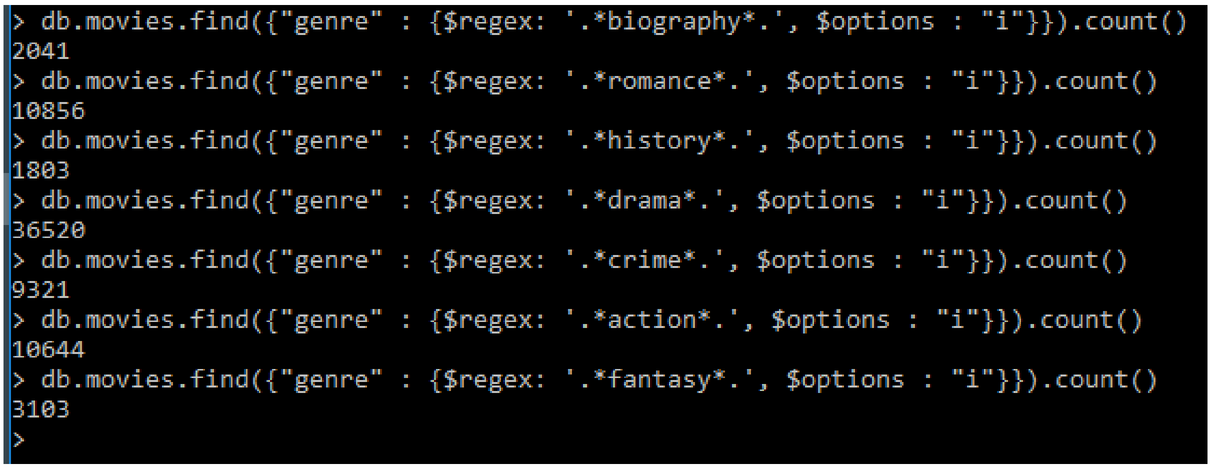
```
> db.movies.aggregate([{"$group" : {_id:"$production_company", count:{$sum:"$votes"}}}, {"$sort":{count:-1}}, {"$limit": 5}])
{ "_id" : "Warner Bros.", "count" : 65170648 }
{ "_id" : "Universal Pictures", "count" : 48912040 }
{ "_id" : "Paramount Pictures", "count" : 45624926 }
{ "_id" : "Columbia Pictures", "count" : 41988348 }
{ "_id" : "Twentieth Century Fox", "count" : 36907589 }
>
```

11. Count of movies in various genres (Mongo)

CODE –

- db.movies.find({“genre”: {$regex: ‘.*biography*.’, $option: “i”}}).count()
- db.movies.find({“genre”: {$regex: ‘.*romance*.’, $option: “i”}}).count()
- db.movies.find({“genre”: {$regex: ‘.*history*.’, $option: “i”}}).count()
- db.movies.find({“genre”: {$regex: ‘.*drama*.’, $option: “i”}}).count()
- db.movies.find({“genre”: {$regex: ‘.*crime*.’, $option: “i”}}).count()
- db.movies.find({“genre”: {$regex: ‘.*action*.’, $option: “i”}}).count()
- db.movies.find({“genre”: {$regex: ‘.*fantasy*.’, $option: “i”}}).count()

OUTPUT –



```
> db.movies.find({"genre" : {$regex: '.*biography*.', $options : "i"}}).count()
2041
> db.movies.find({"genre" : {$regex: '.*romance*.', $options : "i"}}).count()
10856
> db.movies.find({"genre" : {$regex: '.*history*.', $options : "i"}}).count()
1803
> db.movies.find({"genre" : {$regex: '.*drama*.', $options : "i"}}).count()
36520
> db.movies.find({"genre" : {$regex: '.*crime*.', $options : "i"}}).count()
9321
> db.movies.find({"genre" : {$regex: '.*action*.', $options : "i"}}).count()
10644
> db.movies.find({"genre" : {$regex: '.*fantasy*.', $options : "i"}}).count()
3103
>
```

12. Year that released most number of films in each century (Mongo)

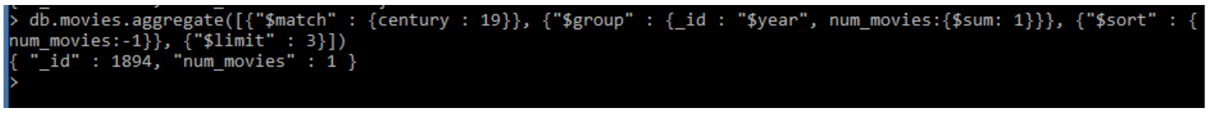
CODE –

- db.movies.aggregate([{“$match”: {century: 19}}, {“$group”: {_id: “$year”, num_movies: {$sum: 1}}}, {“$sort”: {num_movies: -1}}, {“$limit”: 3}])

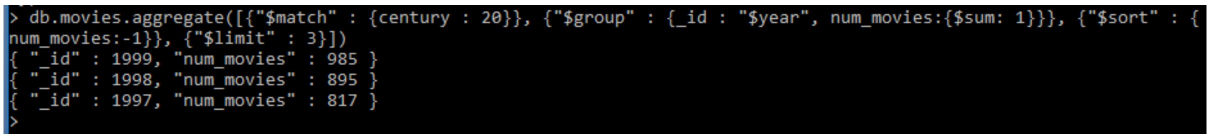
- db.movies.aggregate([{“$match”: {century: 20}}, {“$group”: {_id: “$year”, num_movies: {$sum: 1}}}, {“$sort”: {num_movies: -1}}, {“$limit”: 3}])

- db.movies.aggregate([{“$match”: {century: 21}}, {“$group”: {_id: “$year”, num_movies: {$sum: 1}}}, {“$sort”: {num_movies: -1}}, {“$limit”: 3}])

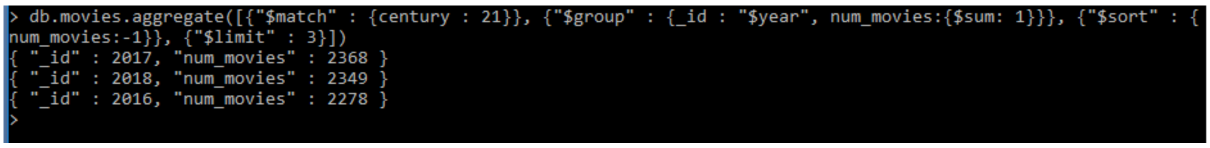
OUTPUT –



```
> db.movies.aggregate([{"$match" : {century : 19}}, {"$group" : {_id : "$year", num_movies:{$sum: 1}}}, {"$sort" : {
num_movies:-1}}, {"$limit" : 3}])
{ "_id" : 1894, "num_movies" : 1 }
>
```



```
> db.movies.aggregate([{"$match" : {century : 20}}, {"$group" : {_id : "$year", num_movies:{$sum: 1}}}, {"$sort" : {
num_movies:-1}}, {"$limit" : 3}])
{ "_id" : 1999, "num_movies" : 985 }
{ "_id" : 1998, "num_movies" : 895 }
{ "_id" : 1997, "num_movies" : 817 }
>
```



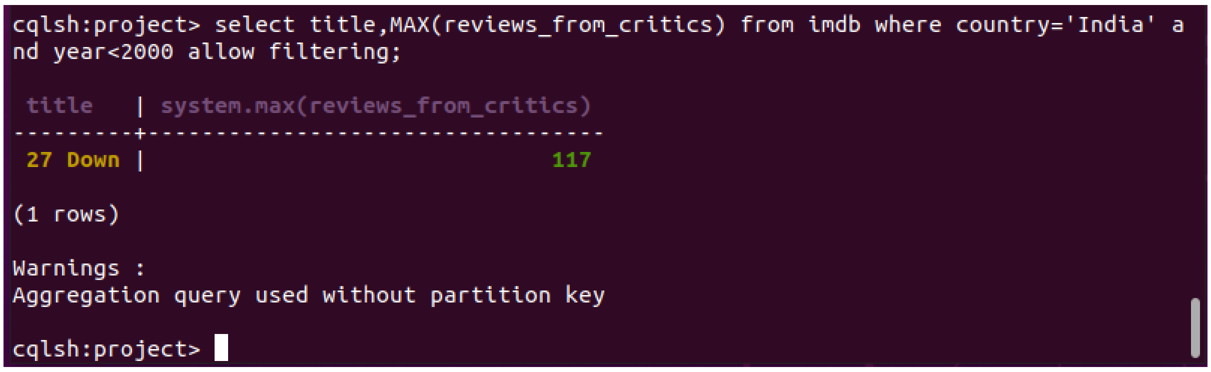
```
> db.movies.aggregate([{"$match" : {century : 21}}, {"$group" : {_id : "$year", num_movies:{$sum: 1}}}, {"$sort" : {
num_movies:-1}}, {"$limit" : 3}])
{ "_id" : 2017, "num_movies" : 2368 }
{ "_id" : 2018, "num_movies" : 2349 }
{ "_id" : 2016, "num_movies" : 2278 }
>
```

13. Movie that got maximum reviews from critics in India after year 2000 (Cassandra)

CODE –

select title, MAX(reviews_from_critics) from imdb where country=’India’ and year<2000 allow filtering;

OUTPUT –



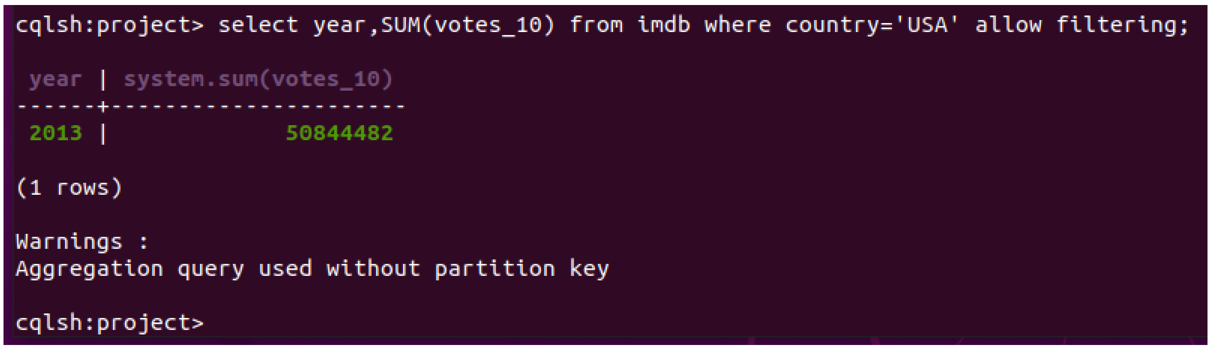
```
cqlsh:project> select title,MAX(reviews_from_critics) from imdb where country='India' a
nd year<2000 allow filtering;

 title   | system.max(reviews_from_critics)
---------+---------------------------------
 27 Down |                              117

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:project>
```

14. Year that released top rated movies in USA (Cassandra)

CODE –

select year, SUM(votes_10) from imdb where country=’USA’ allow filtering;

OUTPUT –



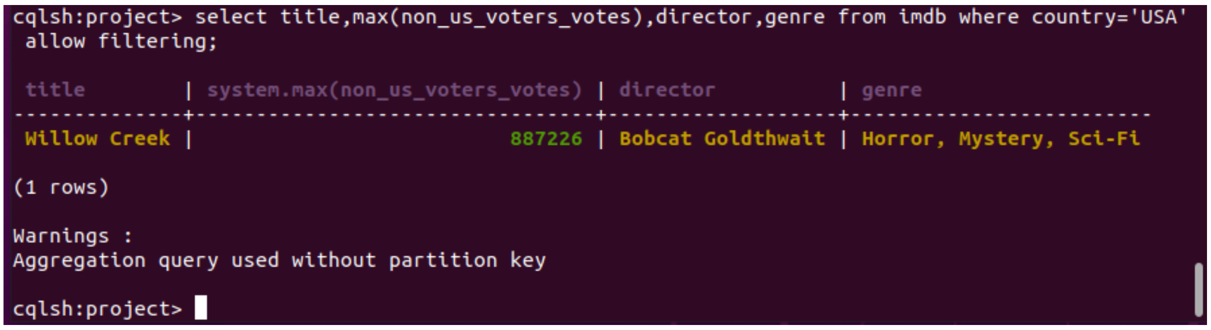
```
cqlsh:project> select year,SUM(votes_10) from imdb where country='USA' allow filtering;

 year | system.sum(votes_10)
------+----------------------
 2013 |             50844482

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:project>
```

15. Details of a movie which released in USA that got most votes from Non-USA people (Cassandra)

CODE –

select title, MAX(non_us_voters_votes), director, genre from imdb where country='USA' allow filtering;

OUTPUT –



```
cqlsh:project> select title,max(non_us_voters_votes),director,genre from imdb where country='USA'
 allow filtering;

 title        | system.max(non_us_voters_votes) | director          | genre
--------------+---------------------------------+-------------------+------------------------
 Willow Creek |                          887226 | Bobcat Goldthwait | Horror, Mystery, Sci-Fi

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:project>
```