**Contributors**

**Member 1**

* Name : Kavin Kumar Arumugam
* Class Id : 1

**Member 2**

* Name : Alper Erel
* Class Id : 8

**Member 3**

* Name : Jayden Tran
* Class Id : 27

**Table of Contents:**

[1. Objectives](https://github.com/kaphc/CSEE5590_Big_Data_Programming/wiki/LAB-1#objectives)  
[2. Approaches/Methods](https://github.com/kaphc/CSEE5590_Big_Data_Programming/wiki/LAB-1#approachesmethods)  
[3. Datasets](https://github.com/kaphc/CSEE5590_Big_Data_Programming/wiki/LAB-1#datasets)  
[4. Code Screenshots](https://github.com/kaphc/CSEE5590_Big_Data_Programming/wiki/LAB-1#code-screenshots)  
[5. Results/Output Screenshots](https://github.com/kaphc/CSEE5590_Big_Data_Programming/wiki/LAB-1#resultsoutput-screenshots)  
[6. Conclusion](https://github.com/kaphc/CSEE5590_Big_Data_Programming/wiki/LAB-1#conclusion)

**Objectives:**

* **Question-1: Use Case: Hadoop Map-Reduce Algorithm:**

**Finding Facebook common friends:** Facebook has an archive of friends (Consider that if "A" is a friend of "B", that means "B" is a friend of "A"). Assume disk space is not a constraint because they execute potentially thousands of requests everyday. They want to do the calculations beforehand so that the processing time for the future requests will be significantly decreased. One of the most common processing demand is the feature that shows number of mutual friends. When you look at somebody's profile, you see the list for the mutual friends. We will be using the Map-Reduce algorithm so that we will have the data for mutual friends at the beginning of each day and have the results ready. Afterwards, it's just a rapid search.

* **Question-2: Use Case: Use Map-Reduce algorithm to analyze You Tube data set given.**

**Task 1:** Determine the top 5 categories that has the maximum number of videos.  
**Task 2:** Determine the top 10 high-rated videos on You tube.

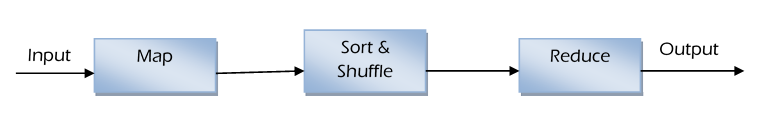
* **Question-3: Use Case: Hive and Solr**

**1. Hive Usecase**  
a. Create a new Hive Table and also include Complex Data Types.  
b. Use built-in functions from Hive.  
c. Perform 10 queries on the dataset.  
**2. Solr Usecase**  
a. Create a new Solr Collection.  
b. Perform 10 queries on the created collection.  
c. Record the execution time for the created 10 queries.

**Approaches/Methods:**

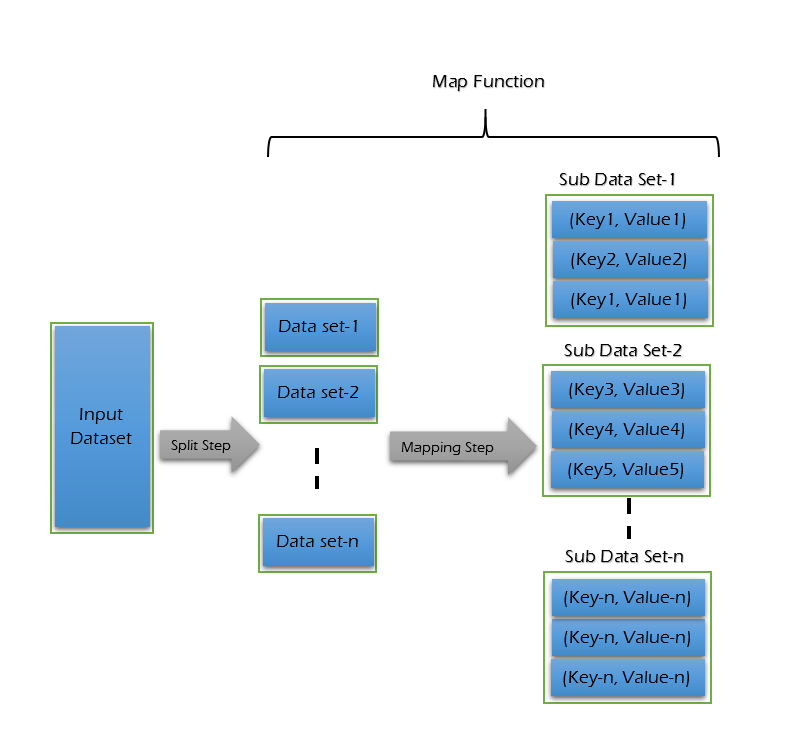
* **Question 1 & 2:**

Map-Reduce algorithm is very advantageous when it comes to process extensive numbers of lines of data. It splits input task into littler and reasonable sub-assignments to manage them in parallel. Map-Reduce algorithm basically sends the processing node to where the data stands.

**Map-Reduce Process Consists of 3 Stages:**  


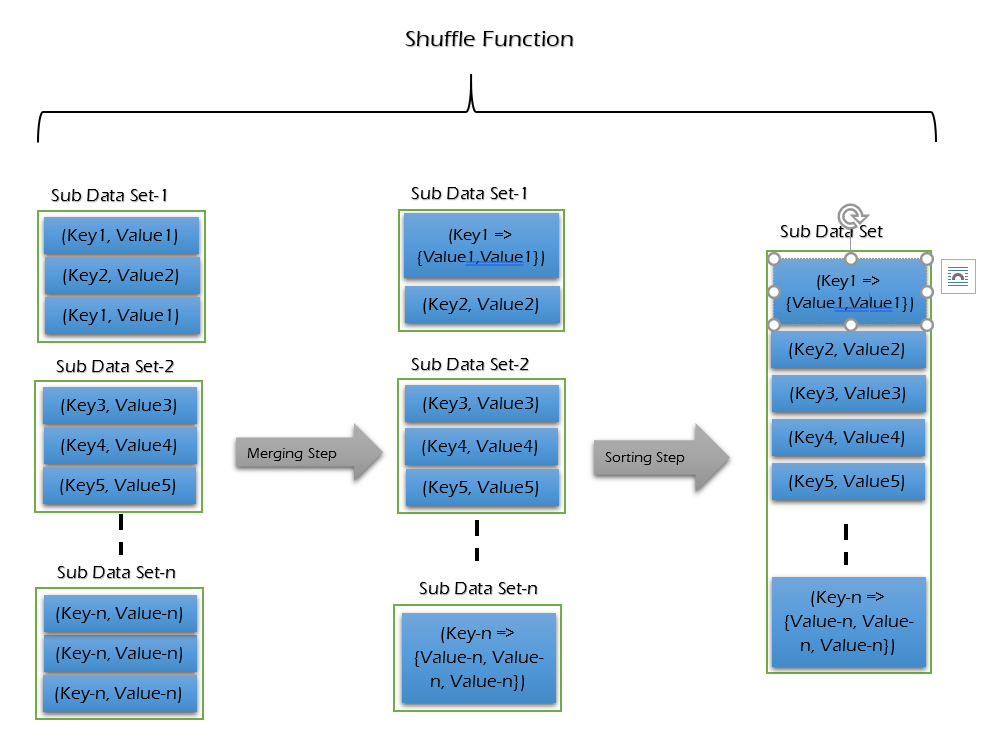
**1) Map Stage**  
Map Function is the initial phase in MapReduce Algorithm. At Map stage will take a shot at key and value sets as ianput.  
A list of data is given to mapper class called mapper **Splitting** - Takes input dataset and divide the input dataset into small groups.  
**Mapping** - Takes the splitted dataset and perform required computation or action on each of them.

**OUTPUT:** set of key and value pairs as <Key, Value>.



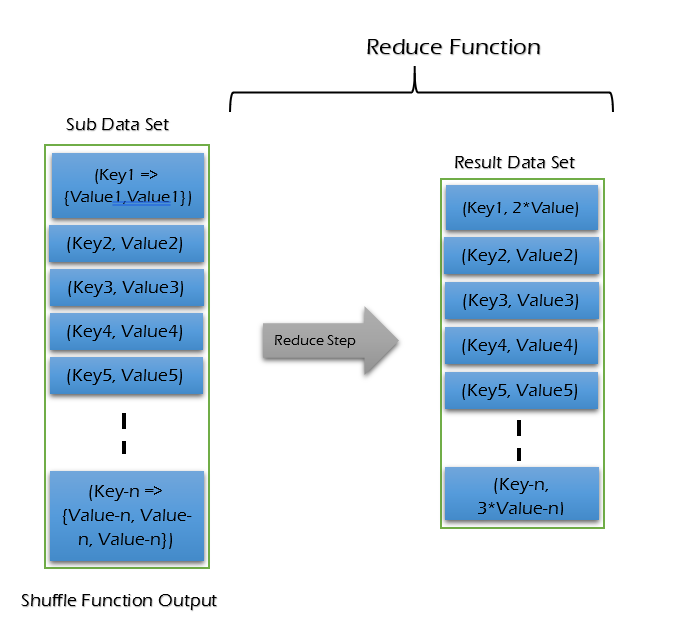
**2) Sort & Shuffle Stage**  
Shuffle and Sort is the second phase in MapReduce Algorithm.  
This Shuffle and Sort is also called as “Combine Function”  
The output from the mapper class is taken as input and sort and shuffle them.  
**Merging** - Find and merge all key and value pairs which all have same key.  
**Sorting** - Sort all of the key and value pairs by keys.

**OUTPUT:** group of key and value pairs as <Key, List>



**3) Reduce Stage**  
Reducer is the final stage in MapReduce Algorithm.  
Takes list of sorted pairs of <Key, List>  
After finishing the reducer part the cluster collects the data and send the data back to hadoop server.

**OUTPUT:** Result as <Key, Value>



* **Question 3:**

1. Hive Usecase

* Create a new table in hive.
* Use built-in hive functions.
* Create 10 queries

1. Solr Usecase

* Create a new Solr Collection.
* Make 10 queries.
* Record execution time for the queries.

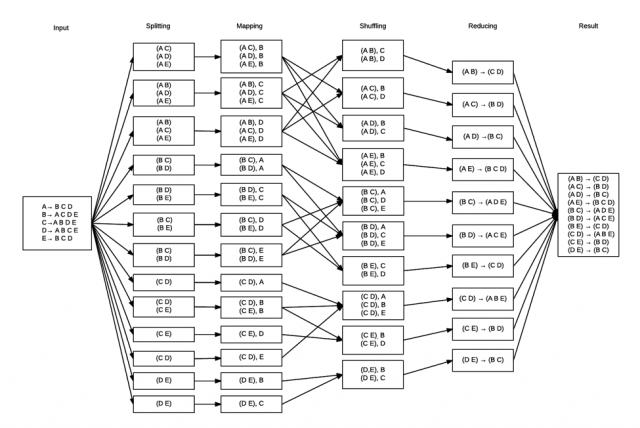
**Datasets:**

* For Question-1: [Facebook Dataset-1](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/data/question-1/facebook_mutual_friends_dataset_1.txt), [Facebook Dataset - 2](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/data/question-1/facebook_mutual_friends_2.txt)
* For Question-2: [Youtube Dataset-before pre-processing](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/data/question-2/youtubedata.csv), [Youtube Dataset-after pre-processing](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/data/question-2/youtubedata.txt)
* For Question-3: [Zomato Dataset](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/data/question-3/zomato.csv)

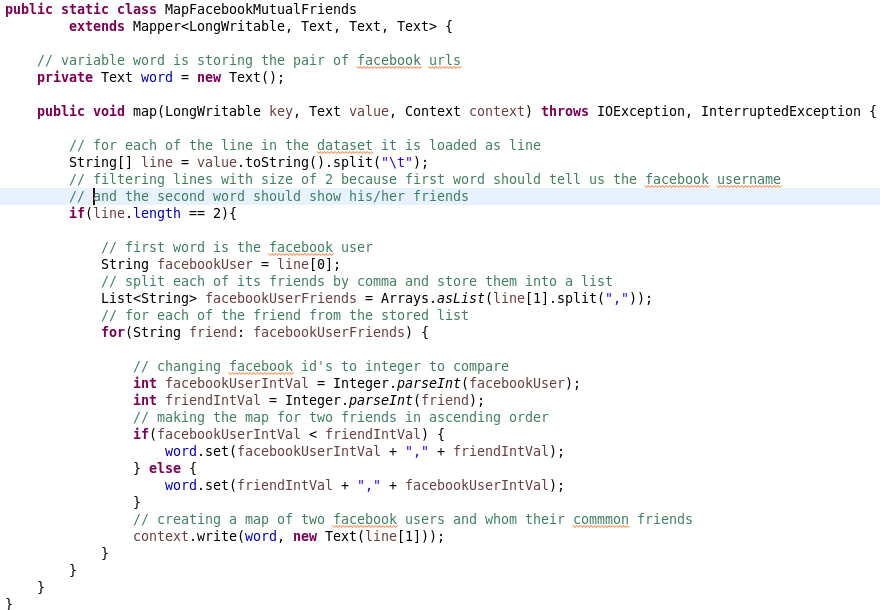
**Code Screenshots:**

* **Question-1:**

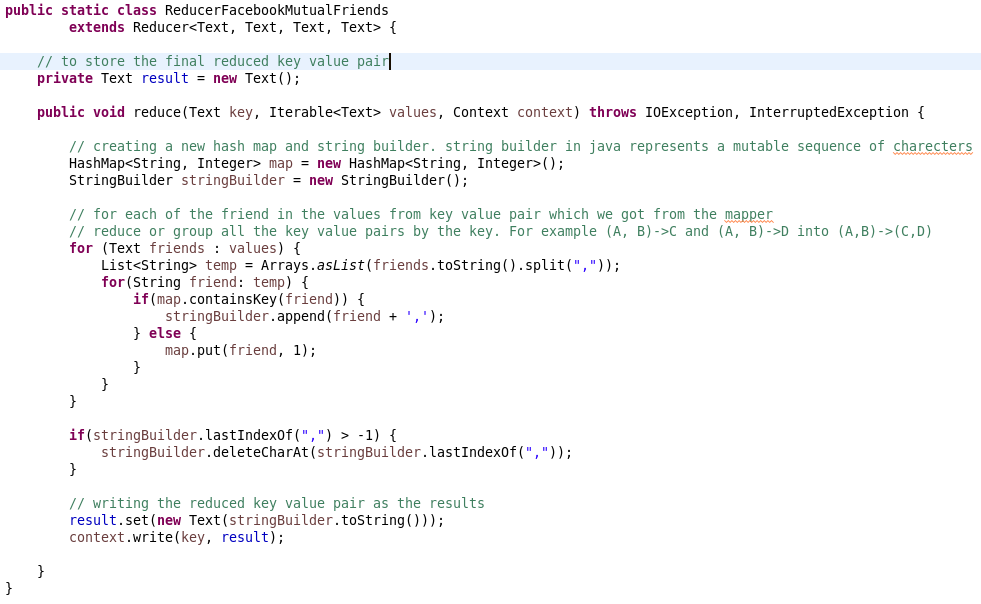
**Map Reduce Diagram:**



**1. Mapper (To see the code ->**[**click here**](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/code/question-1/FacebookMutualFriends.java)**)**



**2. Reducer (To see the code ->**[**click here**](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/code/question-1/FacebookMutualFriends.java)**)**



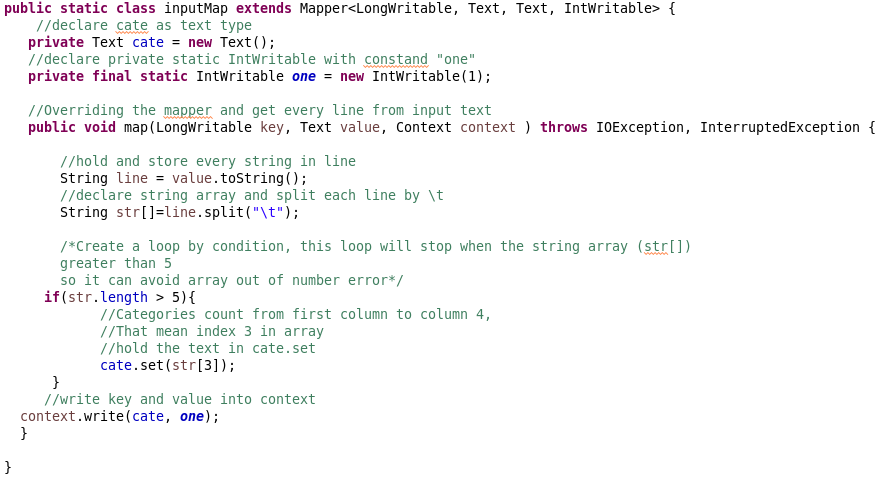
**3. Main Class (To see the code ->**[**click here**](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/code/question-1/FacebookMutualFriends.java)**)**



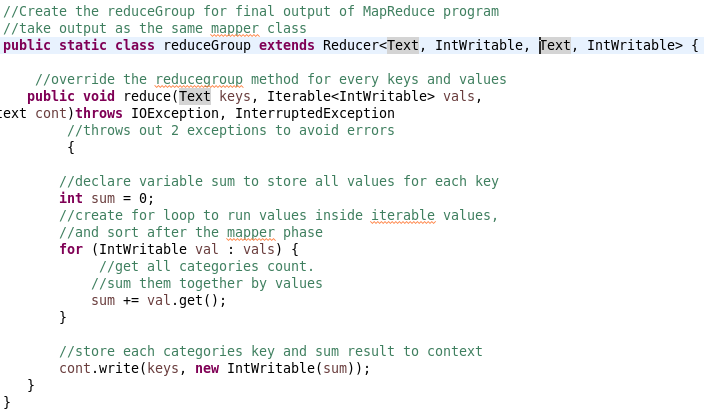
* **Question-2:**

**Part-1**

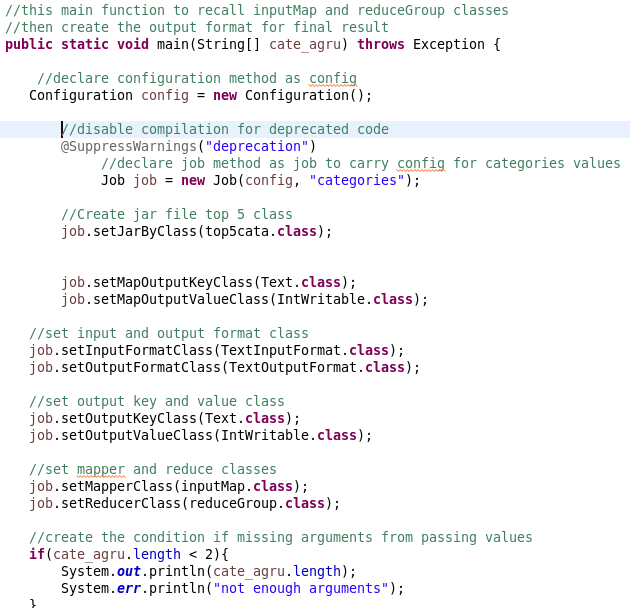
**1. Mapper (To see the code ->**[**click here**](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/code/question-2/top5cata.java)**)**



**2. Reducer (To see the code ->**[**click here**](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/code/question-2/top5cata.java)**)**

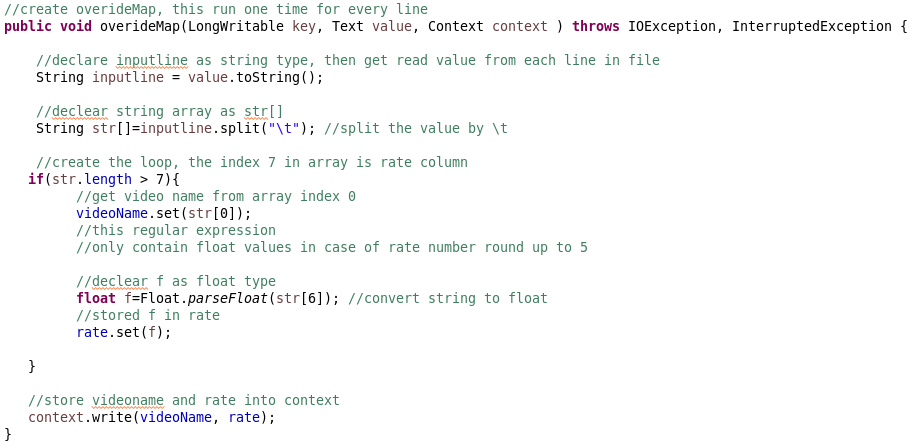


**3. Main Class (To see the code ->**[**click here**](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/code/question-2/top5cata.java)**)**

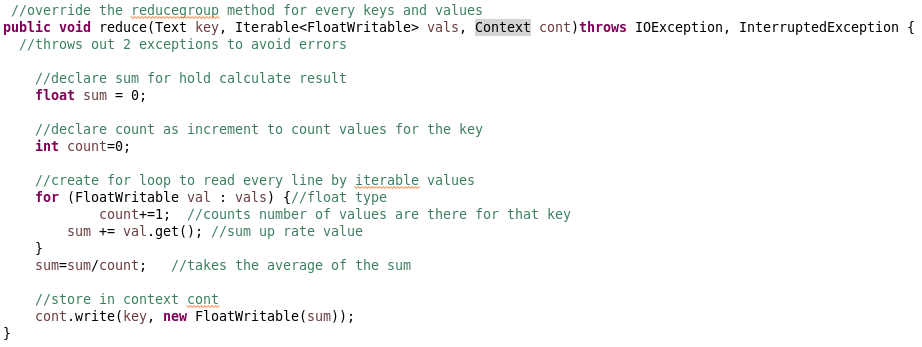


**Part-2**

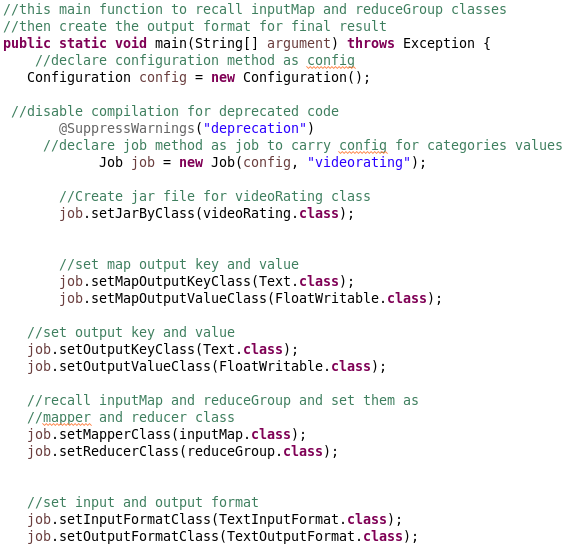
**1. Mapper (To see the code ->**[**click here**](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/code/question-2/videoRating.java)**)**



**2. Reducer (To see the code ->**[**click here**](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/code/question-2/videoRating.java)**)**



**3. Main Class (To see the code ->**[**click here**](https://github.com/kaphc/CSEE5590_Big_Data_Programming/blob/master/LAB-1/Source/code/question-2/videoRating.java)**)**

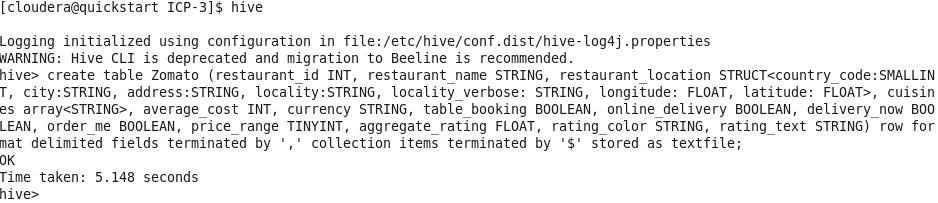


* **Question-3:**

**Part-1**

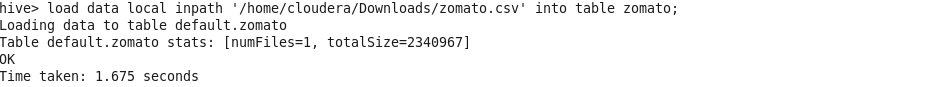
**Start Hive and Create a new table called Zomato.**

**Command: create table Zomato (restaurant\_id INT, restaurant\_name STRING, restaurant\_location STRUCT<country\_code:SMALLINT, city:STRING, address:STRING, locality:STRING, locality\_verbose: STRING, longitude: FLOAT, latitude: FLOAT>, cuisines array, average\_cost INT, currency STRING, table\_booking BOOLEAN, online\_delivery BOOLEAN, delivery\_now BOOLEAN, order\_me BOOLEAN, price\_range TINYINT, aggregate\_rating FLOAT, rating\_color STRING, rating\_text STRING) row format delimited fields terminated by ',' collection items terminated by '$' stored as textfile;**



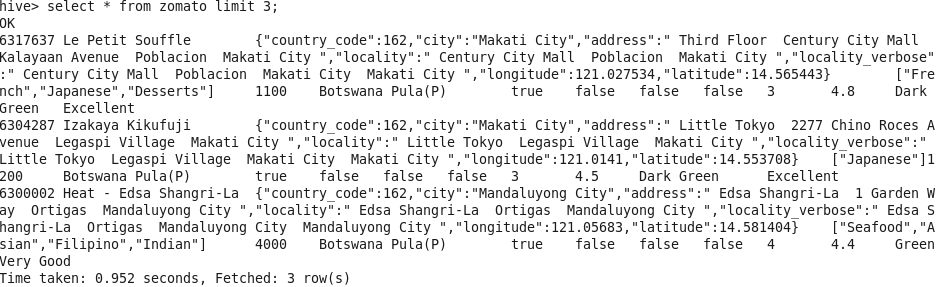
**Load the csv file into the created table.**

**Command: load data local inpath '/home/cloudera/Downloads/zomato.csv' into table zomato;**



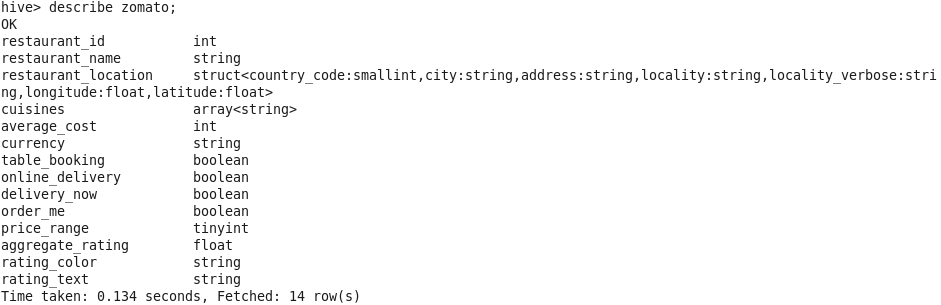
**To verify the data insertion.**

**Command: select \* from zomato limit 3;**



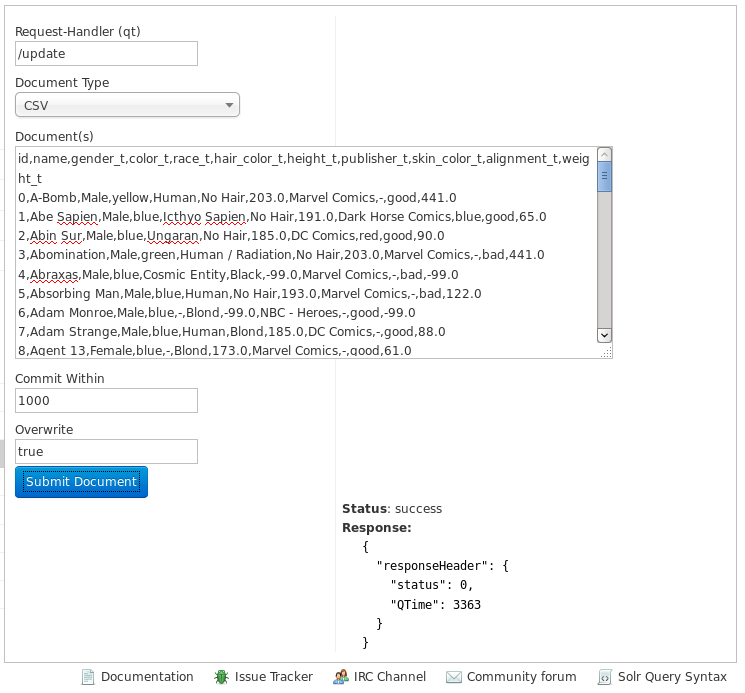
**To describe the created table.**

**Command: select restaurant\_location.city, count(\*) from zomato group by restaurant\_location.city;**



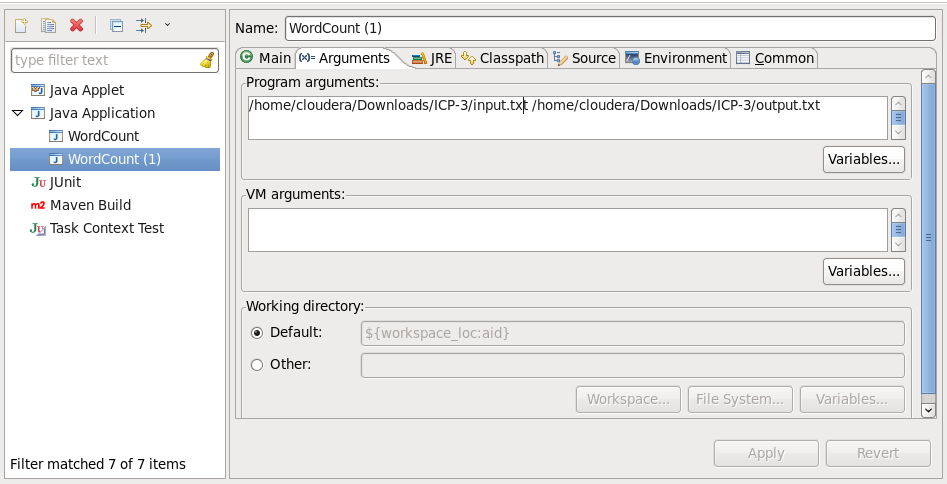
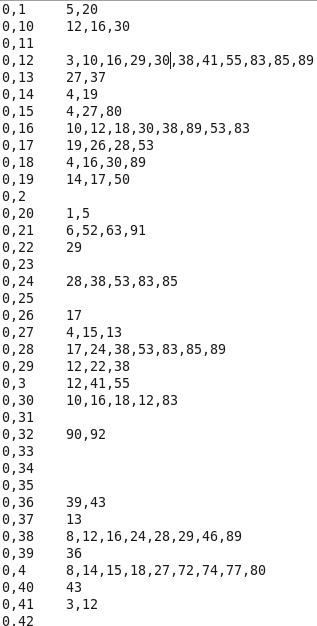
**Part-2**

**Data Insertion in solr.**



**Results/Output Screenshots:**

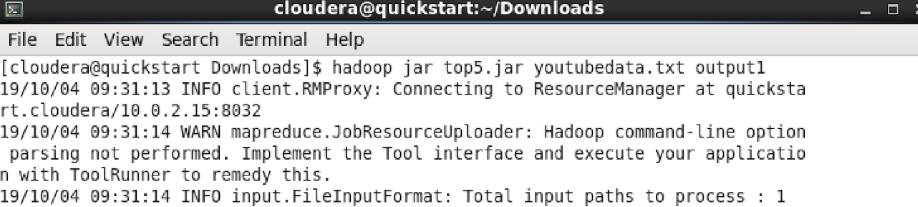
* **Question-1:**

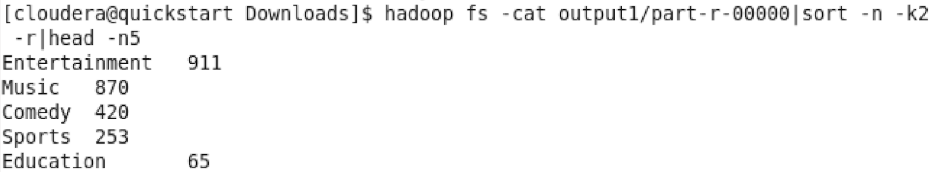
* **Question-2:**

**Part-1**

**Command: Hadoop jar top5.jar youtubedata.txt output1**

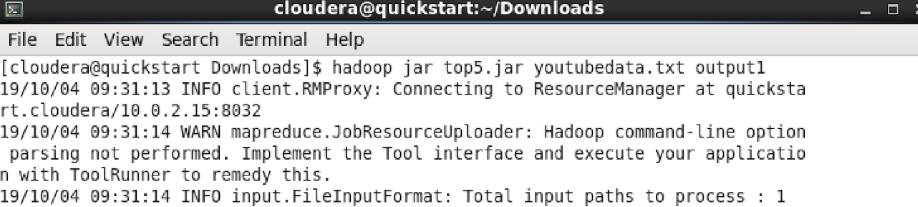


**Command: Hadoop fs -cat /output1/part-r-00000|sort -n -k2 -r|head -n**

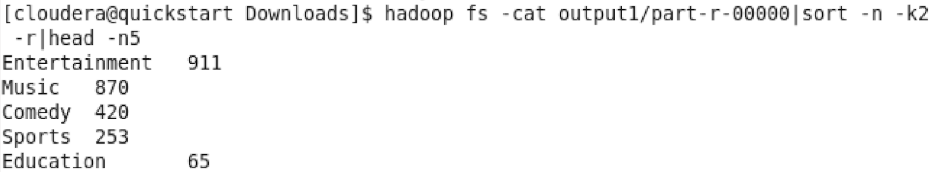


**Part-2**

**Command: Hadoop jar top10.jar youtubedata.txt output2**



**Command: Hadoop fs -cat output2/part-r-00000 |sort -n -k2 -r|head -n10**



**Explaination**

**\*\*Hadoop fs -cat /output1/part-r-00000 \*\***

* fs: specify an operation related to Hadoop
* -cat: use to view the content
* /output1/part-r-00000 : directory to the output file stored

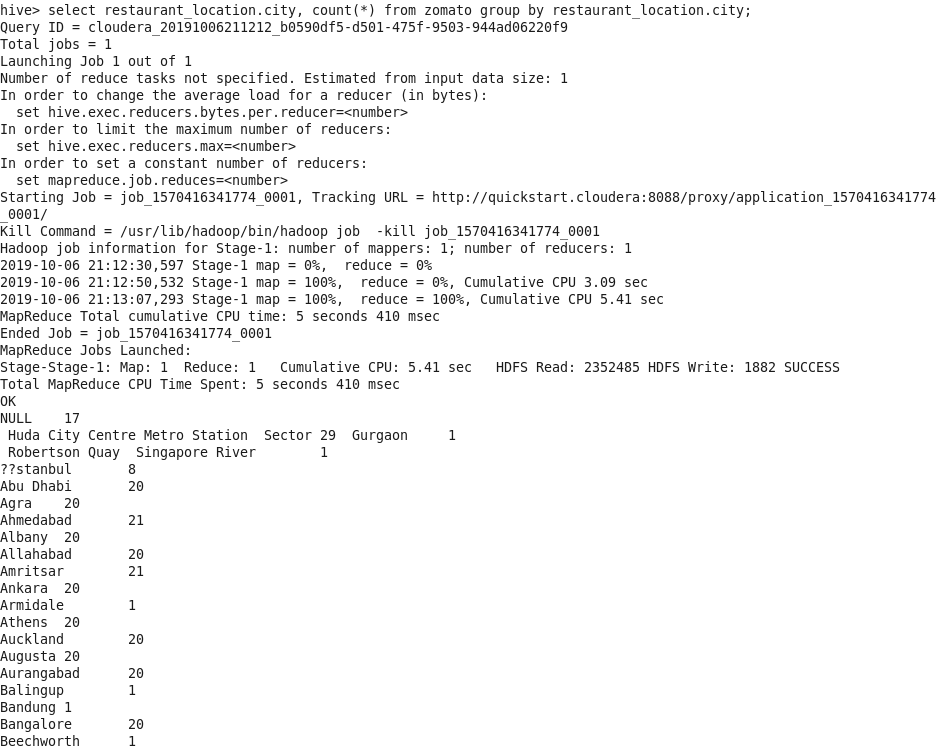
**\*\*sort -n -k2 -r|head -n \*\***

* sort: sort the data
* -n: sort by numerically
* -k2: second column
* -r : recursive operation
* head -n5 : take the first 5 values after sorting.
* **Question-3:**

**Part-1**

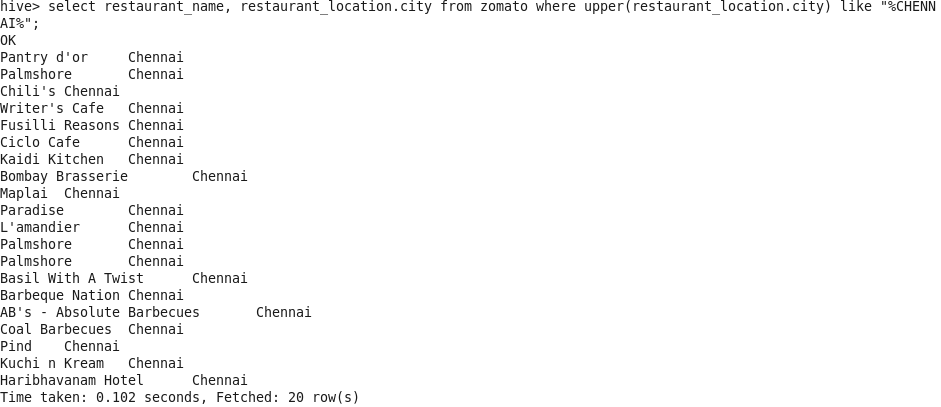
**Query-1: Find number of restaurants in each city.**

**Command: select restaurant\_location.city, count(\*) from zomato group by restaurant\_location.city;**



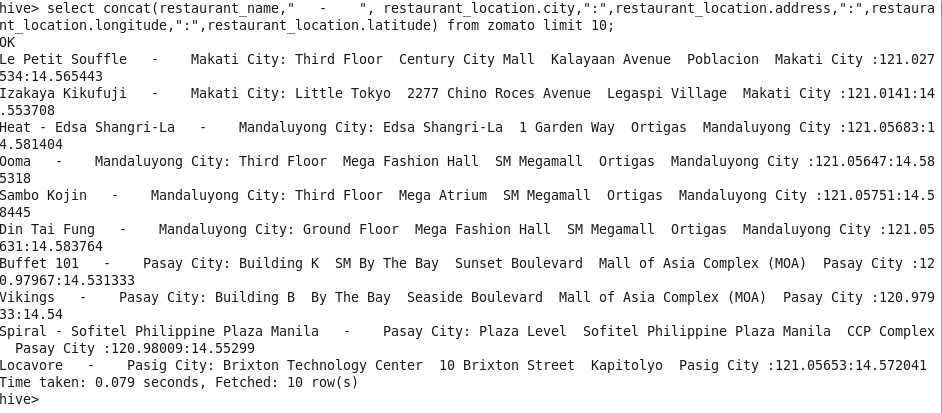
**Query-2: Print restaurant names from chennai.**

**Command: select restaurant\_name, restaurant\_location.city from zomato where upper(restaurant\_location.city) like "%CHENNAI%";**



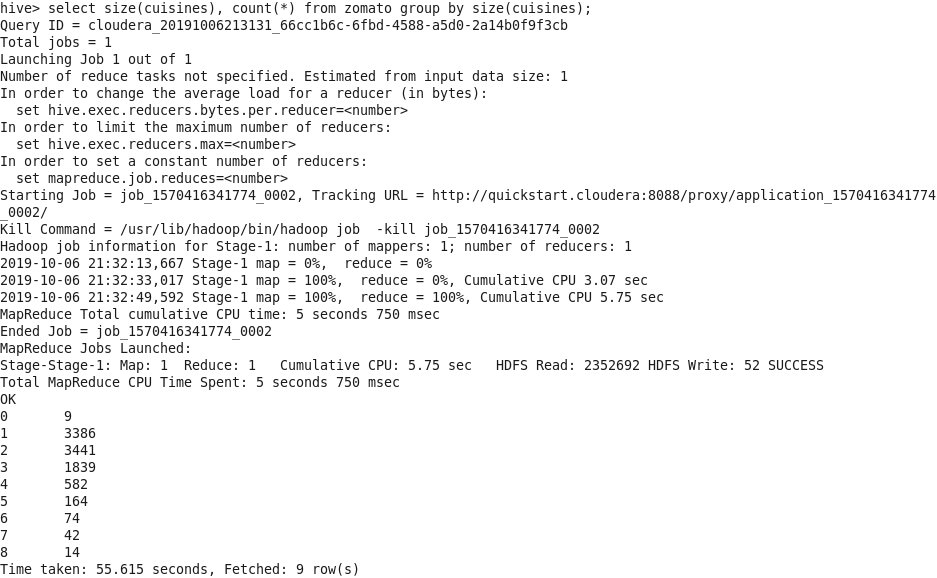
**Query-3: Concat restaurant name with address, latitude, and longitude.**

**Command: select concat(restaurant\_name," - ", restaurant\_location.city,":",restaurant\_location.address,":",restaurant\_location.longitude,":",restaurant\_ ocation.latitude) from zomato limit 10;**



**Query-4: To print size of cuisines with number of restaurants with that cuisines.**

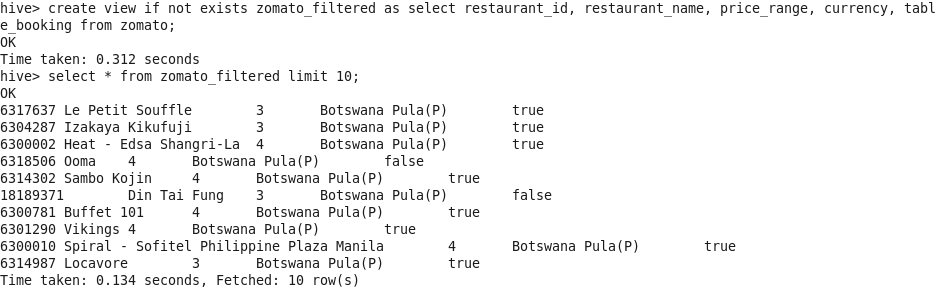
**Command: select size(cuisines), count(\*) from zomato group by size(cuisines);**



**Query-5: Creating a new views in a database can hide personal information and users who have access to the view will not be able to view other hidden messages.**

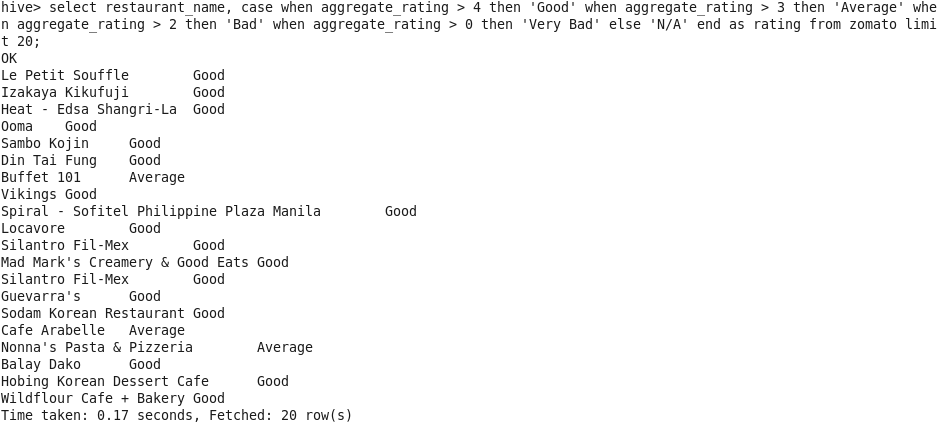
**Command: create view if not exists zomato\_filtered as select restaurant\_id, restaurant\_name, price\_range, currency, table\_booking from zomato;**

**select \* from zomato\_filtered limit 10;**



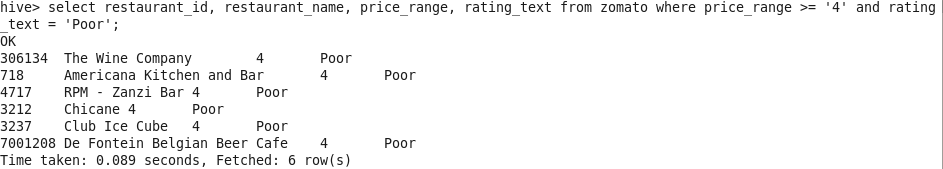
**Query-6: Categorize the restaurants by their rating as "Good", "Average", "Bad", and "Very Bad".**

**Command: select restaurant\_name, case when aggregate\_rating > 4 then 'Good' when aggregate\_rating > 3 then 'Average' when aggregate\_rating > 2 then 'Bad' when aggregate\_rating > 0 then 'Very Bad' else 'N/A' end as rating from zomato limit 20;**



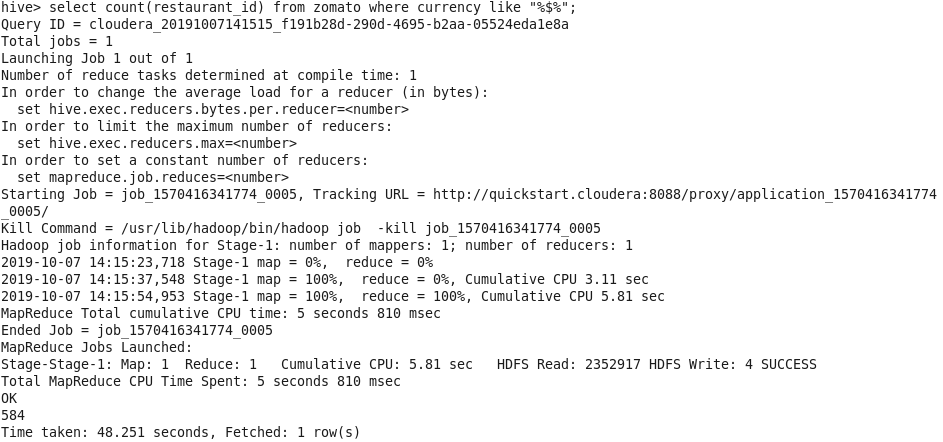
**Query-7: To print restaurants with highest prices and bad rating.**

**Command: select restaurant\_id, restaurant\_name, price\_range, rating\_text from zomato where price\_range >= '4' and rating\_text = 'Poor';**



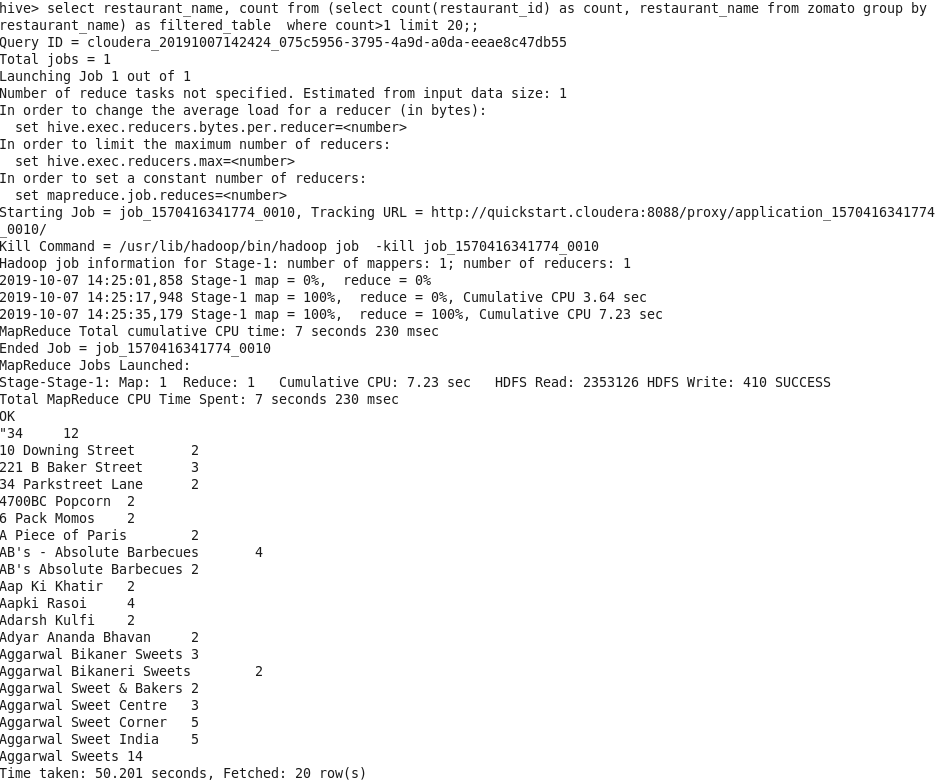
**Query-8: To count number of restaurants using dollars as their currency.**

**Command: select count(restaurant\_id) from zomato where currency like "%$%";**



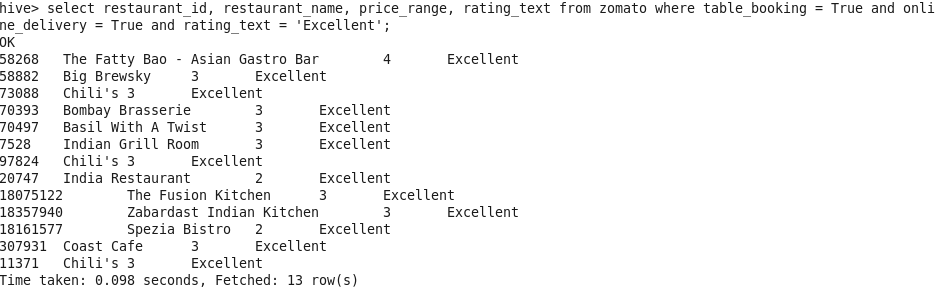
**Query-9: To find restaurants with two or more locations.**

**Command: select restaurant\_name, count from (select count(restaurant\_id) as count, restaurant\_name from zomato group by restaurant\_name) as filtered\_table where count>1 limit 20;**



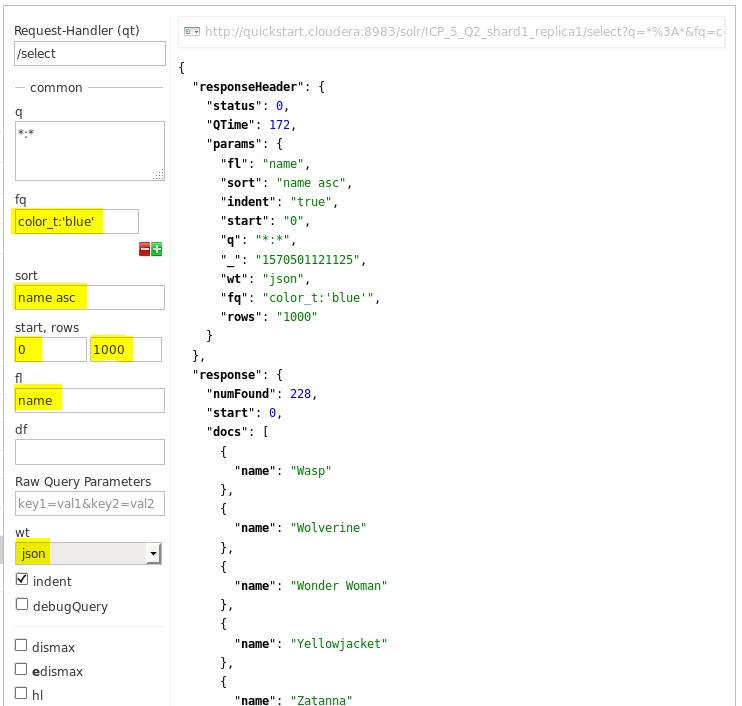
**Query-10: To print restaurant with best customer satisfaction.**

**Command: select restaurant\_id, restaurant\_name, price\_range, rating\_text from zomato where table\_booking = True and online\_delivery = True and delivery\_now = True and rating\_text = 'Excellent';**

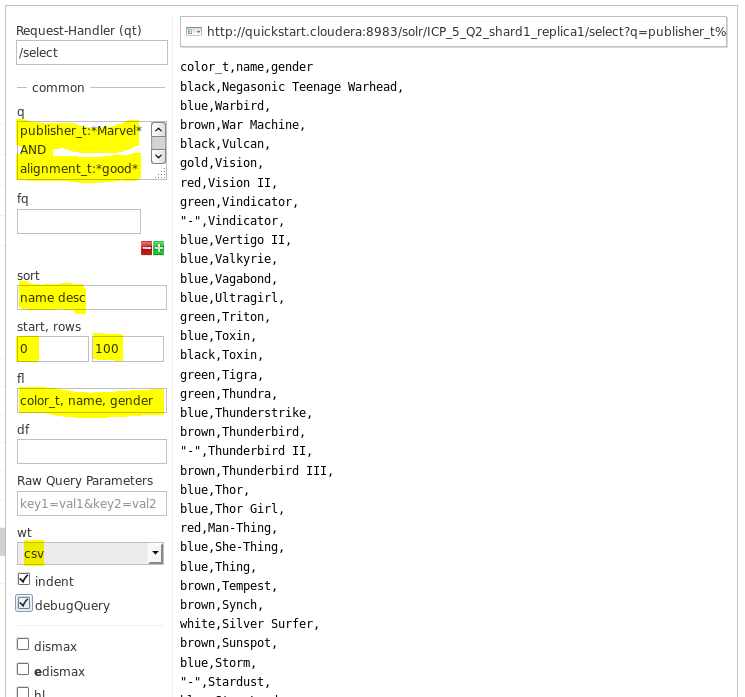


**Part-2**

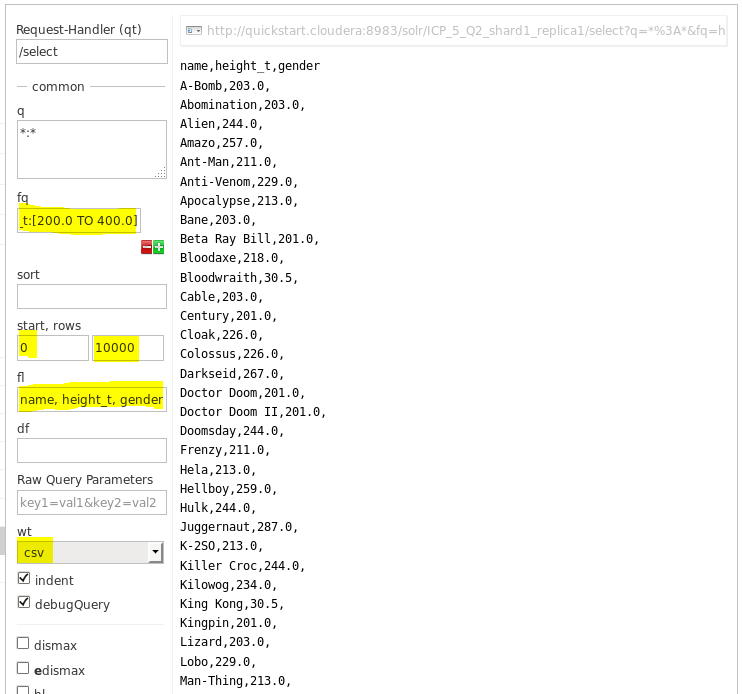
**Query-1: To print names where color is blue from top 1000 records and print them in asc order of name and in json.**



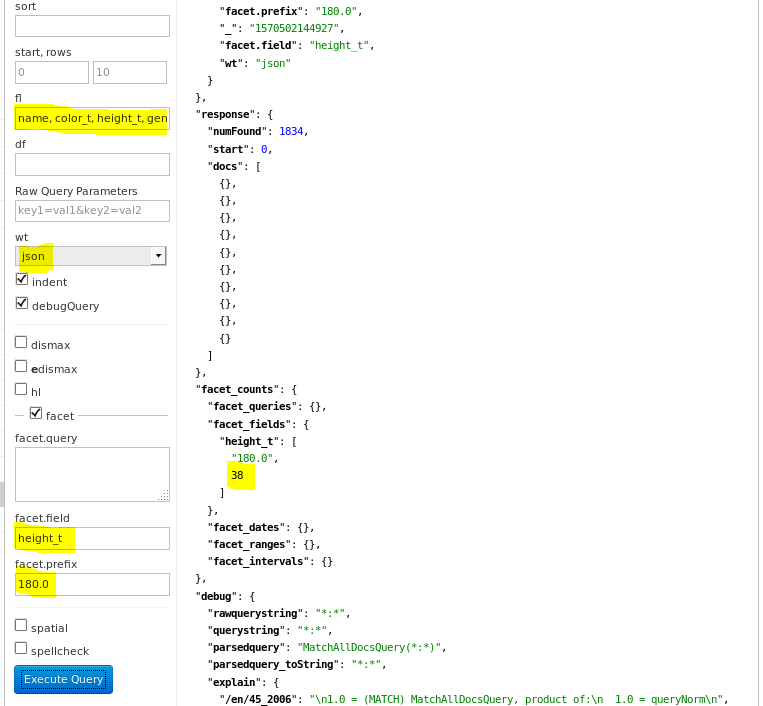
**Query-2: Print name, gender and color from the table and print them as csv, in desc order of name and select from top 100 records.**



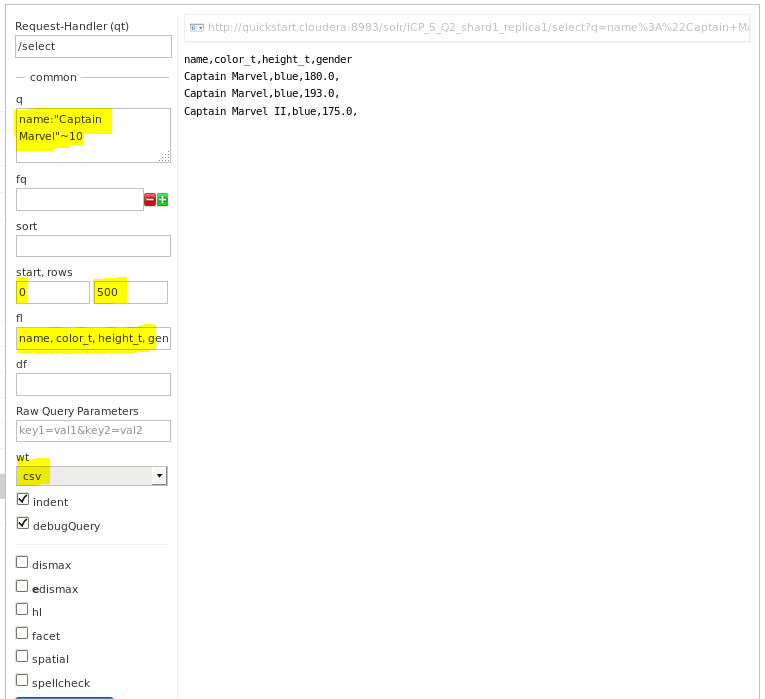
**Query-3: Select name, height and gender as csv from top 10000 records where height is between 200 and 400**



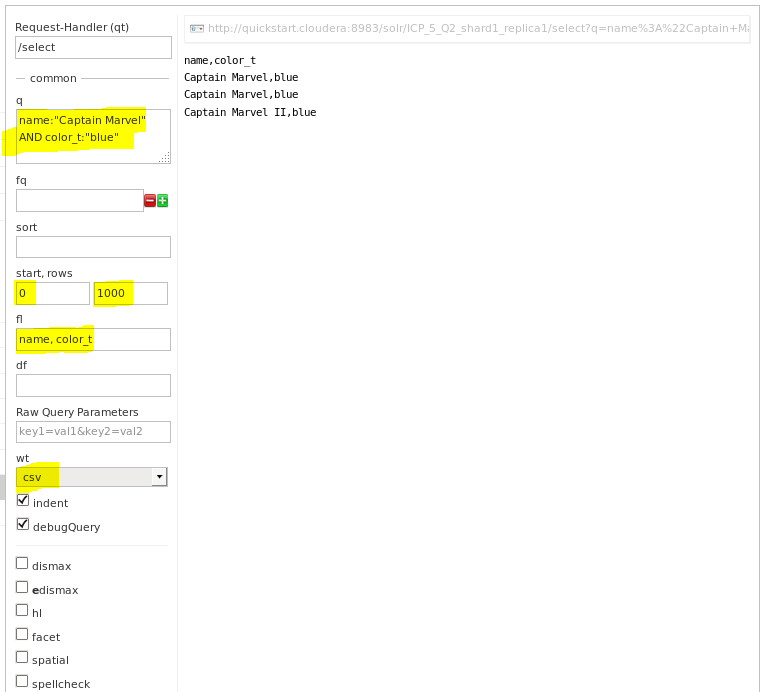
**Query-4: To print name, color, height and gender as json where height is exactly equal to 180.0.**



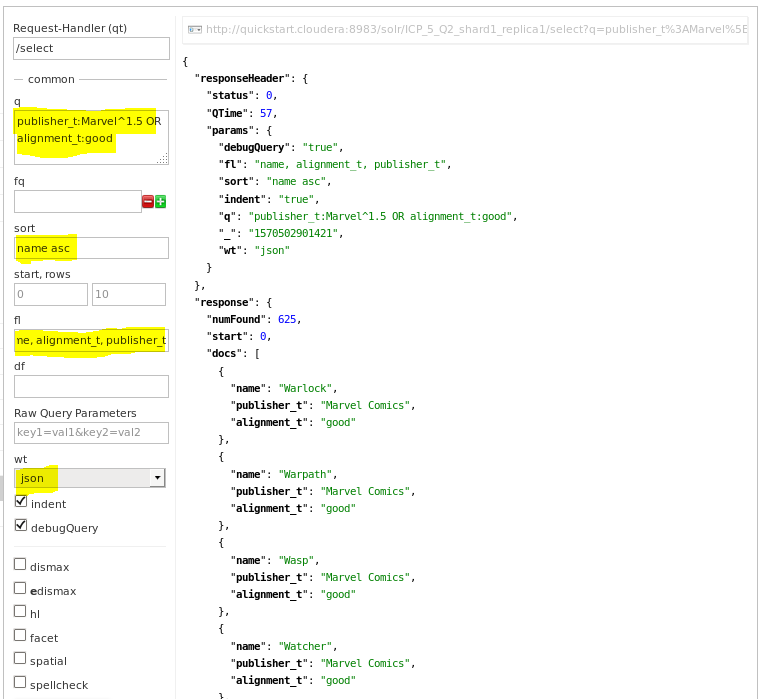
**Query-5: To do proximity search on words "Captain" and "Marvel" and print name, color from top 500 records.**



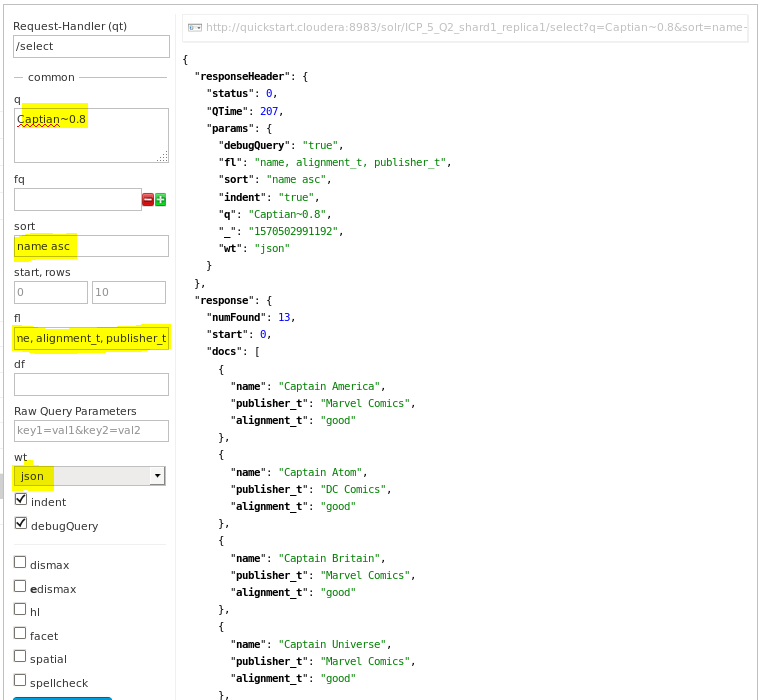
**Query-6: To print all the records from 0 to 1000 where name is "Captain Marvel" and color is "blue" (using AND operator). Print as csv.**



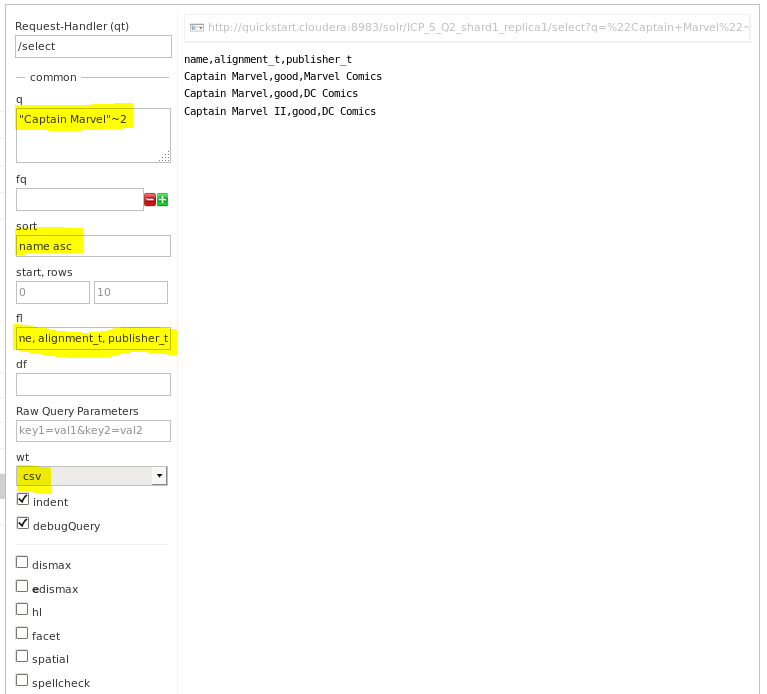
**Query-7: Boost queries: To find publisher as Marvel and alignment is good. Print in asc order of name and in json.**



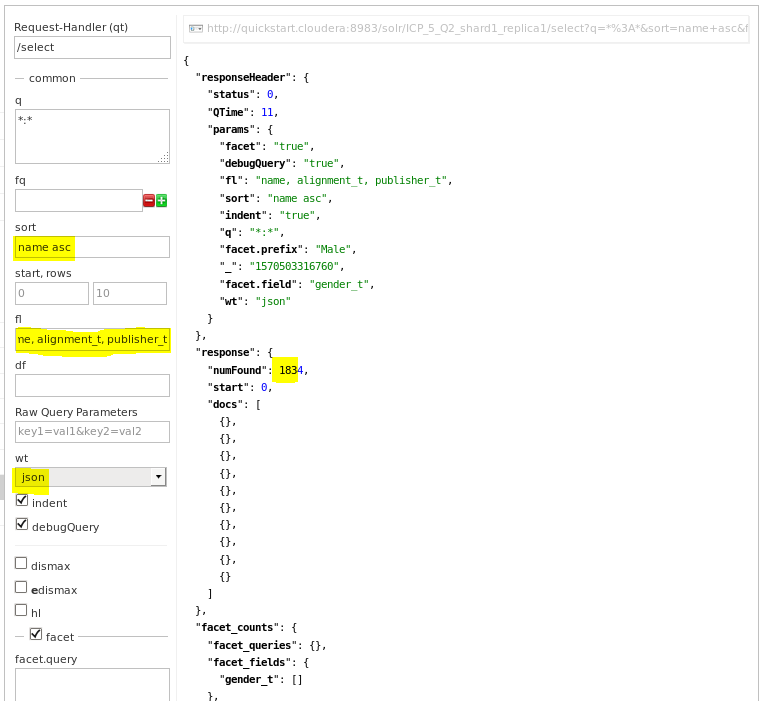
**Query-8: To do fuzzy search on the word "Captian" and print in asc of name and in json.**



**Query-9: To do proximity search on the words "Captain" and "Marvel" with distance as 2. Print in csv format and asc order of name.**



**Query-10: To find number of Males.**


**Conclusion**

In this Lab Assignment we were able to revise ICP-1 to ICP-7. This lab assignment made us to understand Hive, Solr and MapReduce concepts.