Amazon Electronics Products Analysis

By,

Sushmanth Sagala

Hadoop Project – UpX Academy

Project Information

- **Domain:** Retail
- **Technology use:** Hadoop, Hive
- **Dataset:** http://jmcauley.ucsd.edu/data/amazon/links.html
- The Amazon Product data set to be considered for this project is from Amazon, a well-known e-commerce giant. This is a very interesting project to get insights based analysis of product sales based on reviews and ratings.
- This dataset contains product reviews including 142.8 million reviews spanning May 1996 - July 2014. This data set includes reviews (ratings, text, helpfulness votes). From the various categories the Electronics category is chosen for the analysis. The size of reviews data set is 1.5 GB and ratings data set is 320 MB.

Data Description

- Review:
  - reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B
  - asin - ID of the product, e.g. 0000013714
  - reviewerName - name of the reviewer
  - helpful - helpfulness rating of the review, e.g. 2/3
  - reviewText - text of the review
  - overall - rating of the product
  - summary - summary of the review
  - unixReviewTime - time of the review (unix time)
  - reviewTime - time of the review (raw)
- Rating:
  - user - ID of the reviewer, e.g. A2SUAM1J3GNN3B
  - Item - ID of the product, e.g. 0000013714
  - Rating - rating of the product
  - Timestamp - time of the review (unix time)

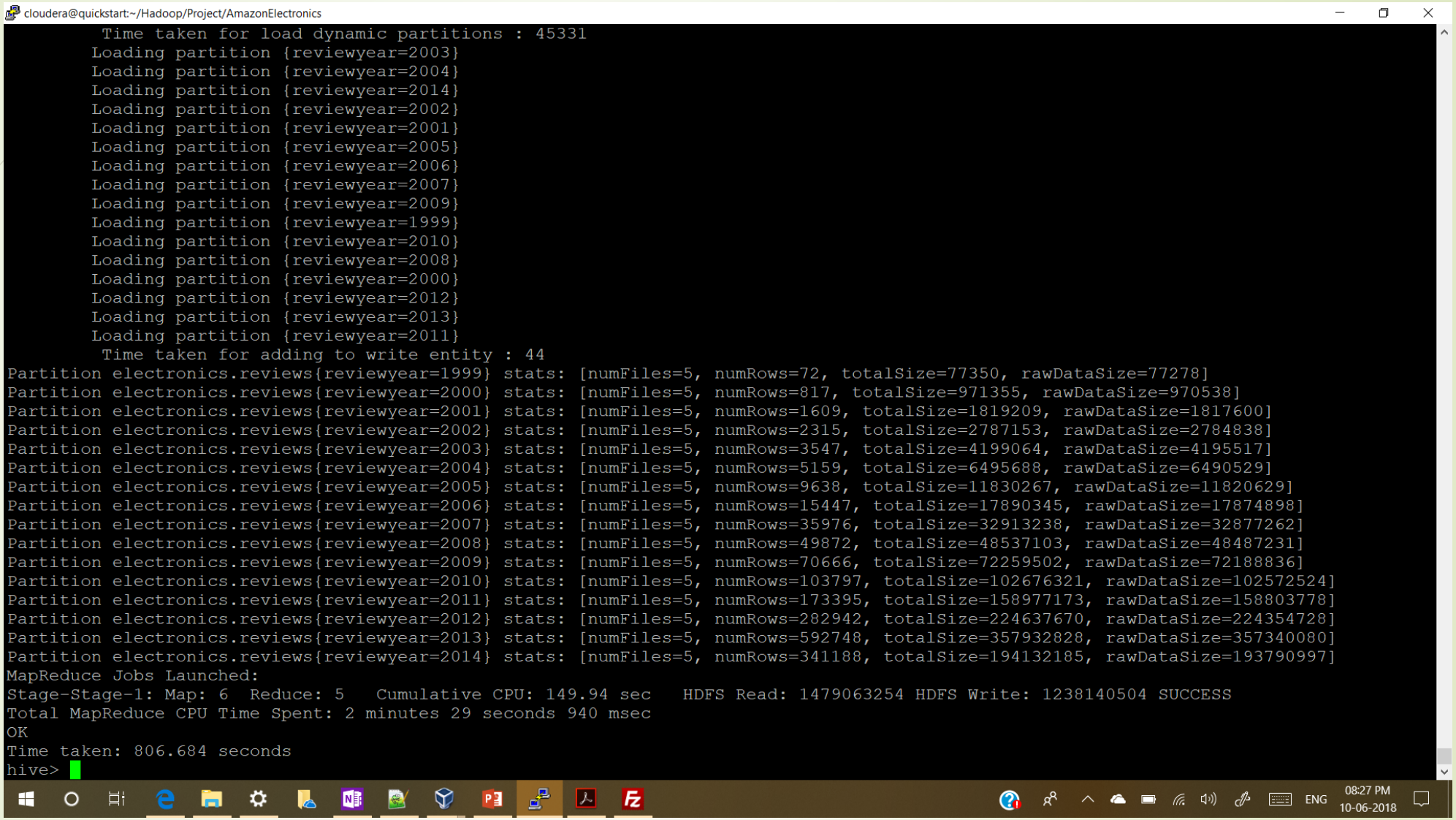
Business Questions

- **Hadoop / Hive**
  - Understand the Dataset
  - Top electronic item that was rated above 3.0 over the period of 8 years [2006-2014] (EG. Year 2006 - 100 items bought received all the rating above 3.0 - show top 1 item out of 100; sample data 2006 ,"ASIN", 'Review Count")
  - Worst review a product received in a given year between 2006 to 2014 (Eg. 2006, ASIN, Review Count (Least number of reviews and below 2.0 rating)
  - Maximum number of reviews given by same user in a year
  - Least Helpful reviews per year per products (helpful percentage between 1% and 30 %)
  - Most Helpful reviews Per year Per Products (Helpful percentage >75%)
  - Growth of review comments on products/year
  - Visualization of results observe

Q1. Understand the Dataset

Reviews dataset:

- Reviews dataset brought to HDFS as a Hive table using org.apache.hive.hcatalog.data.JsonSerDe library to load Json data.
  - CREATE TABLE amazon_reviews (reviewerID string, asin string, reviewerName string, helpful array<int>, reviewText string, overall float, summary string, unixReviewTime int, reviewTime string)ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe' STORED AS TEXTFILE;
  - LOAD DATA LOCAL INPATH 'data/Reviews/Electronics_5.json' OVERWRITE INTO TABLE amazon_reviews;
- Partitioned reviews data based on review year and clustered the data based on review rating.
  - CREATE TABLE reviews (reviewerID string, asin string, reviewerName string, helpful array<int>, reviewText string, overall float, summary string, unixReviewTime int, reviewTime string) PARTITIONED BY (reviewYear int) CLUSTERED BY (overall) INTO 5 BUCKETS ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
  - INSERT INTO TABLE reviews PARTITION(reviewYear) SELECT reviewerID,asin,reviewerName,helpful,reviewText,overall,summary,unixReviewTime,reviewTime,year(from_unixtime(unix_timestamp(reviewTime, 'MM dd, yyyy'))) FROM amazon_reviews;

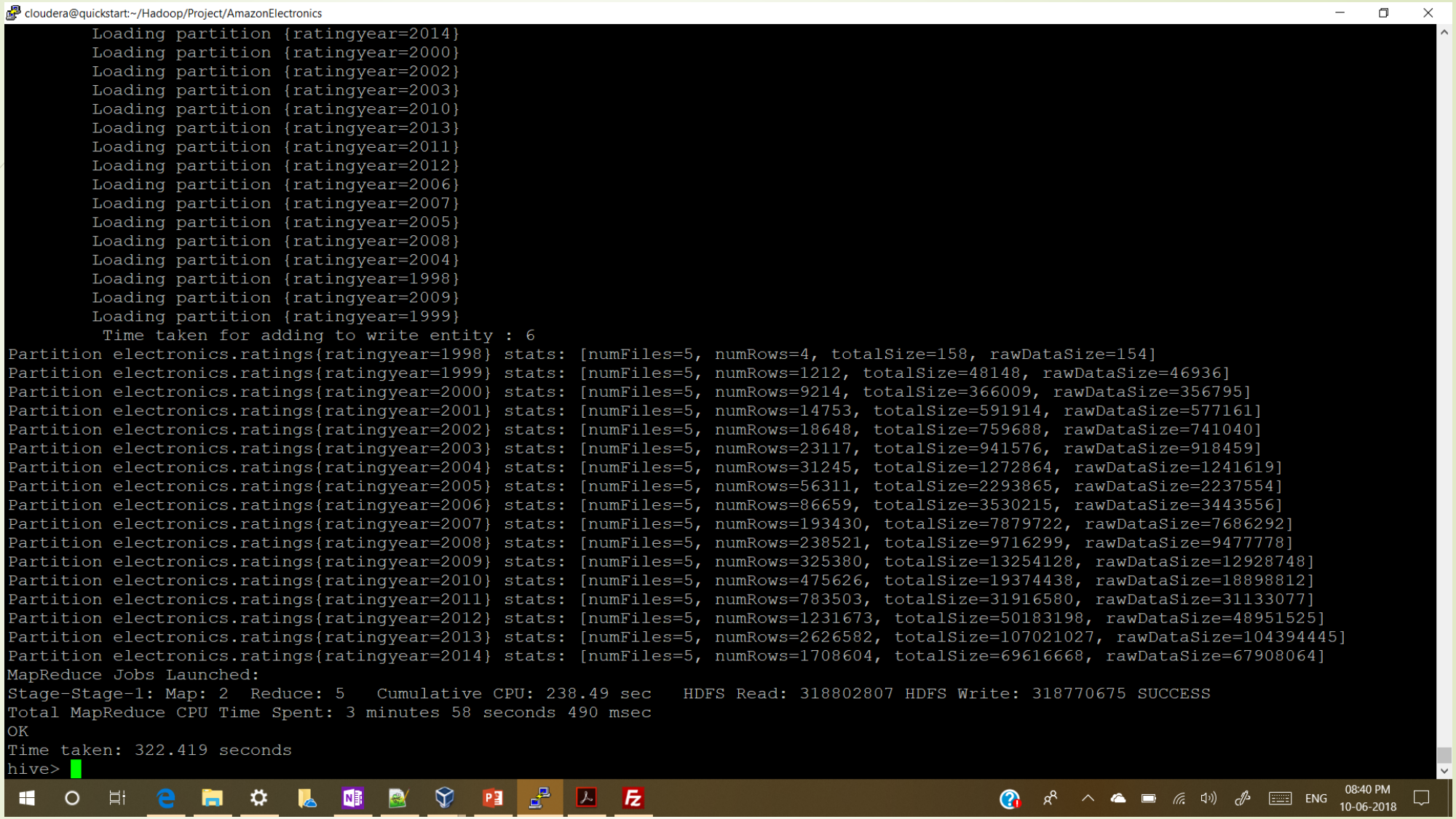


Sample output screenshot

Q1. Understand the Dataset

Ratings dataset:

- Ratings dataset brought to HDFS as a Hive table.
  - CREATE TABLE amazon_ratings (user string, item string, rating float, timestamp int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
  - LOAD DATA LOCAL INPATH 'data/Ratings/ratings_Electronics.csv' OVERWRITE INTO TABLE amazon_ratings;
- Partitioned rating data based on rating year and clustered the data based on rating.
  - CREATE TABLE ratings (user string, item string, rating float, timestamp int) PARTITIONED BY (ratingYear int) CLUSTERED BY (rating) INTO 5 BUCKETS ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
  - INSERT INTO TABLE ratings PARTITION(ratingYear) SELECT user,item,rating,timestamp,year(date_add(from_unixtime(timestamp),1)) FROM amazon_ratings;



Sample output screenshot

Business Questions Answered

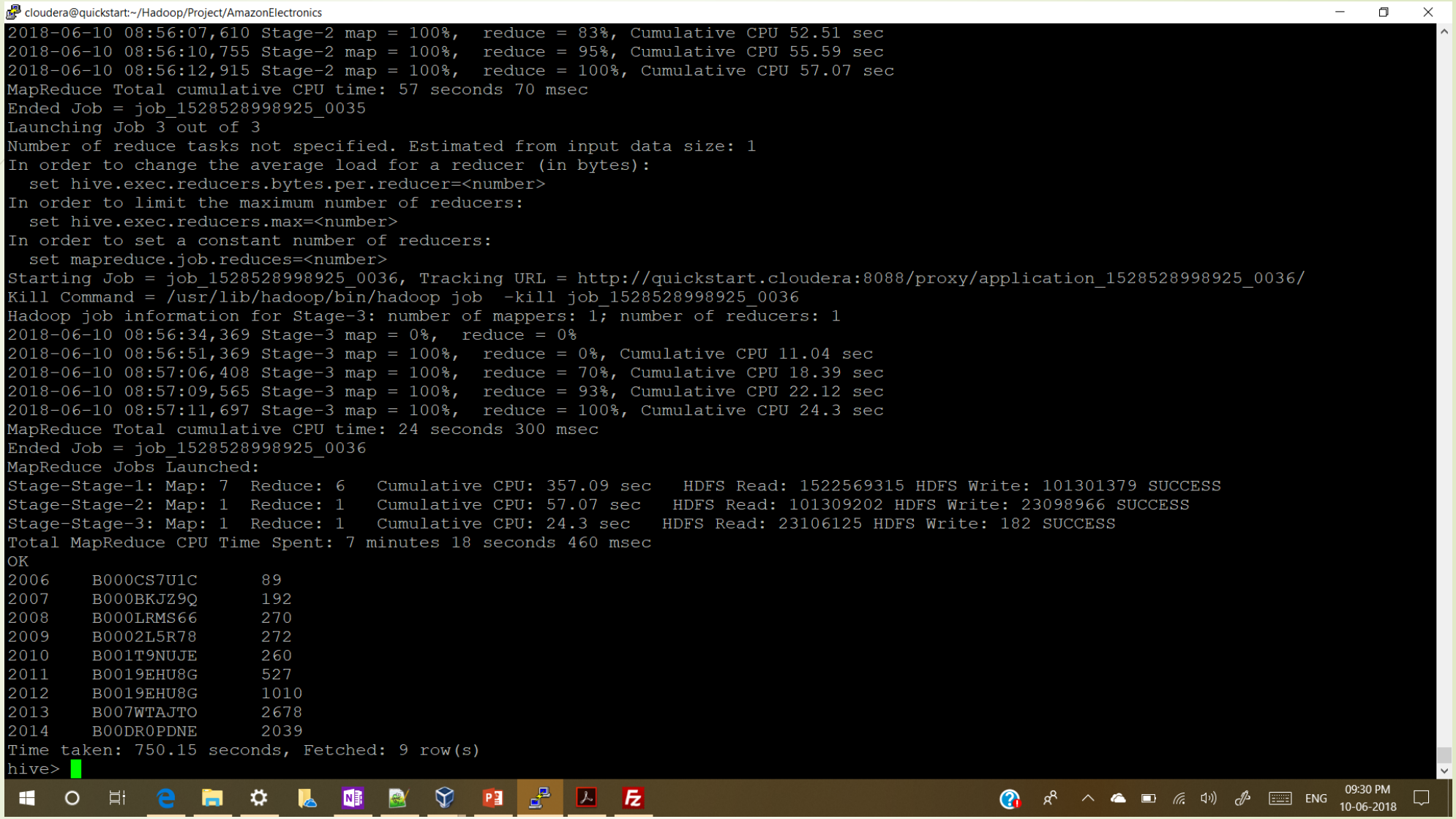
- Q2. Top electronic item that was rated above 3.0 over the period of 8 years [2006-2014] (EG. Year 2006 - 100 items bought received all the rating above 3.0 - show top 1 item out of 100; sample data 2006 ,"ASIN", 'Review Count")
  - To determine the Top electronics item rated above 3.0 against each year [2006-2014].
  - Used outer join as all reviewers will be present in ratings dataset. Right outer join is used as ratings is bigger table.
  - Grouped all the Items against each year with avg rating greater than 3.0 and choosing the item with highest review count against each year [2006-2014].

```
SELECT r.ratingYear,r.item,r.review_count

FROM  (SELECT rt.ratingYear,rt.item,rt.review_count,ROW_NUMBER() OVER(PARTITION BY rt.ratingYear ORDER BY rt.review_count DESC) AS item_rank

FROM (SELECT rat.ratingYear,rat.item,AVG(rat.rating) AS avg_rating,COUNT(rev.reviewerID) AS review_count

FROM (SELECT reviewYear,asin,reviewerID FROM reviews WHERE reviewYear >= 2006 AND reviewYear <= 2014) rev

LEFT OUTER JOIN (SELECT ratingYear,item,user,rating FROM ratings WHERE ratingYear >= 2006 AND ratingYear <= 2014) rat

ON rev.asin = rat.item AND rev.reviewerID = rat.user

GROUP BY rat.ratingYear,rat.item ) rt

WHERE avg_rating > CAST(3.0 AS FLOAT) ) r

WHERE r.item_rank = 1;
```

Business Questions Answered

- Q3. Worst review a product received in a given year between 2006 to 2014 (Eg. 2006, ASIN, Review Count (Least number of reviews and below 2.0 rating)
  - To determine the worst electronics item rated below 2.0 against each year [2006-2014].
  - Used outer join as all reviewers will be present in ratings dataset. Right outer join is used as ratings is bigger table.
  - Grouped all the Items against each year with avg rating lesser than 2.0 and choosing the item with least review count against each year [2006-2014].

```
SELECT r.ratingYear,r.item,r.review_count

FROM  (SELECT rt.ratingYear,rt.item,rt.review_count,ROW_NUMBER() OVER(PARTITION BY rt.ratingYear
ORDER BY rt.review_count ASC) AS item_rank

FROM (SELECT rat.ratingYear,rat.item,AVG(rat.rating) AS avg_rating,COUNT(rev.reviewerID) AS
review_count

FROM (SELECT reviewYear,asin,reviewerID FROM reviews WHERE reviewYear >= 2006 AND reviewYear <=
2014) rev

RIGHT OUTER JOIN (SELECT ratingYear,item,user,rating FROM ratings WHERE ratingYear >= 2006 AND
ratingYear <= 2014) rat

ON rev.asin = rat.item AND rev.reviewerID = rat.user

GROUP BY rat.ratingYear,rat.item ) rt

WHERE avg_rating < CAST(2.0 AS FLOAT) ) r

WHERE r.item_rank = 1;
```

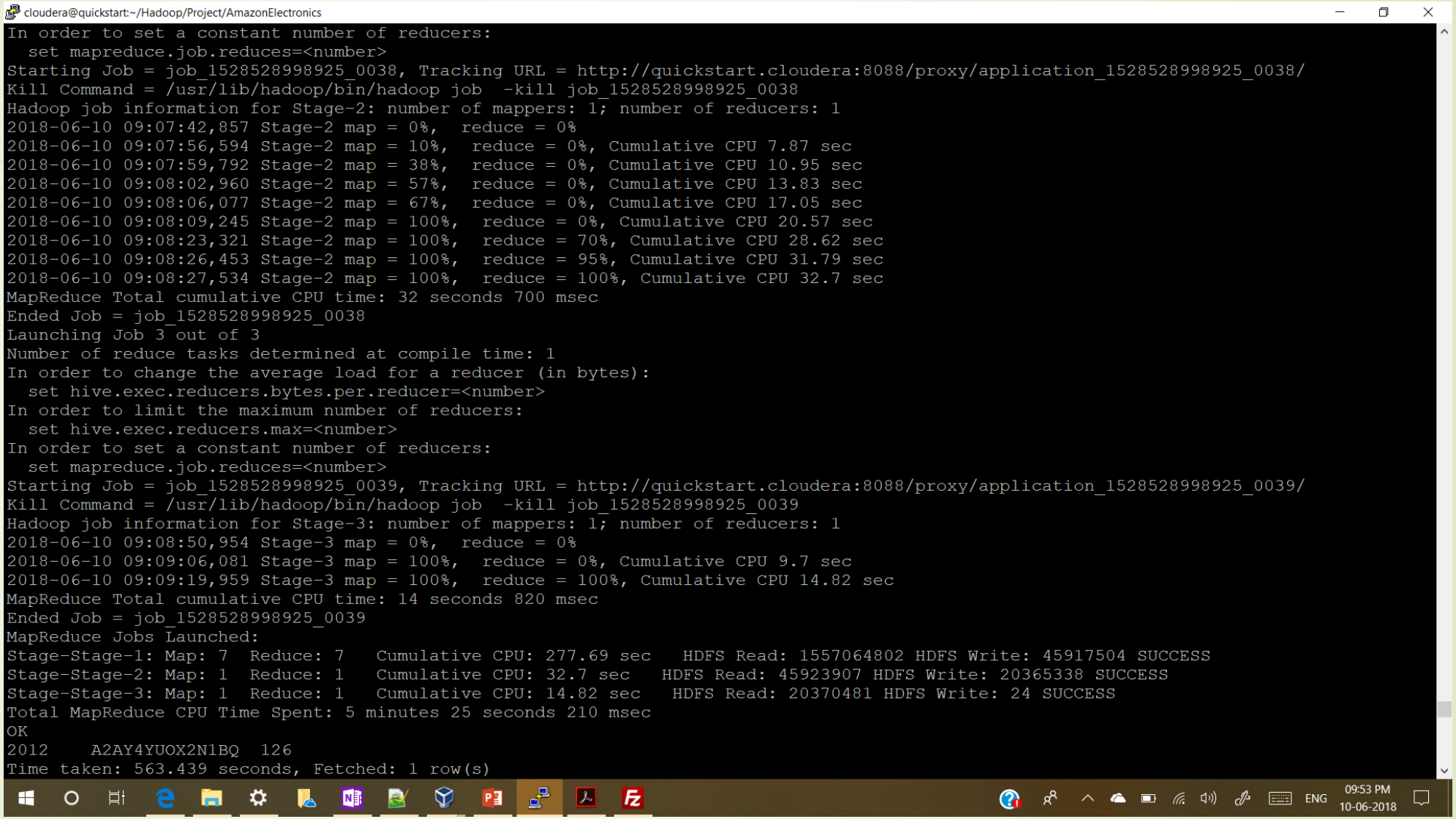


Sample output screenshot

Business Questions Answered

- Q4. Maximum number of reviews given by same user in a year
  - To determine the reviewer by whom maximum number of reviews given in a year.
  - Used outer join as all reviewers will be present in ratings dataset. Right outer join is used as ratings is bigger table.
  - Grouped all the reviewers against each year to find the highest reviews given.

```
SELECT rev.reviewYear,rev.reviewerID,COUNT(rev.reviewerID) AS review_count
FROM reviews rev
RIGHT OUTER JOIN ratings rat
ON rev.asin = rat.item AND rev.reviewerID = rat.user
GROUP BY rev.reviewYear,rev.reviewerID
ORDER BY review_count DESC LIMIT 1;
```



Sample output screenshot

Business Questions Answered

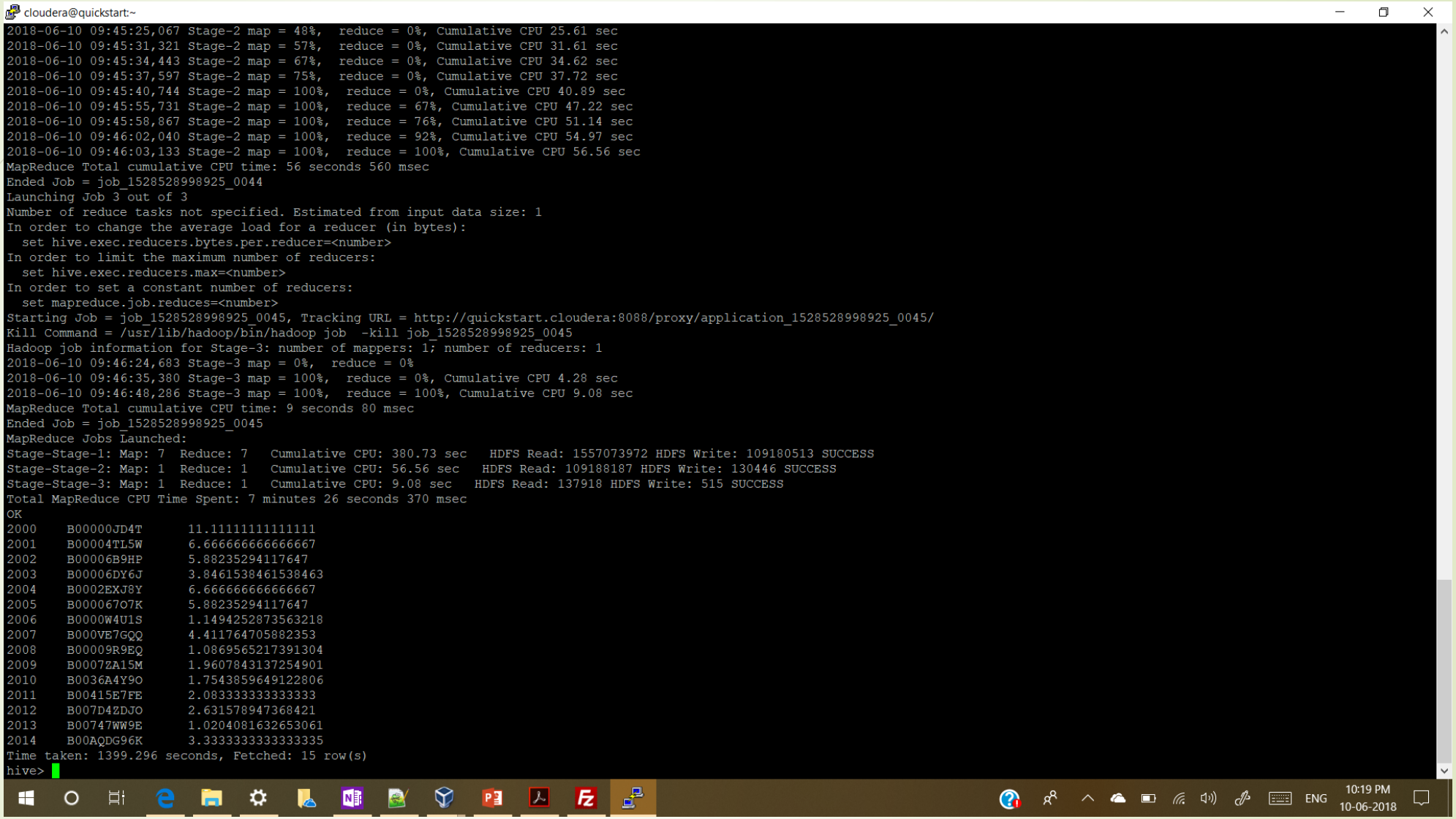
- Q5. Least Helpful reviews per year per products (helpful percentage between 1% and 30 %)
  - To determine the Item per year with least helpful reviews within 1% and 30%.
  - Used outer join as all reviewers will be present in ratings dataset. Right outer join is used as ratings is bigger table.
  - Grouped all the Items against each year with avg helpful percentage between 1% and 30%, choosing the item with least helpful reviews against each year.

```
SELECT r.ratingYear,r.item,r.avg_helpful_perc

FROM  (SELECT rt.ratingYear,rt.item,rt.avg_helpful_perc,ROW_NUMBER() OVER(PARTITION BY rt.ratingYear ORDER BY rt.avg_helpful_perc ASC) AS helpful_perc_rank

FROM (SELECT rat.ratingYear,rat.item,AVG((helpful[0] / helpful[1]) * 100) AS avg_helpful_perc

FROM reviews rev

RIGHT OUTER JOIN ratings rat

ON rev.asin = rat.item AND rev.reviewerID = rat.user

GROUP BY rat.ratingYear,rat.item ) rt

WHERE rt.avg_helpful_perc >= CAST(1.0 AS FLOAT) AND avg_helpful_perc <= CAST(30.0 AS FLOAT) ) r

WHERE r.helpful_perc_rank = 1;
```

Business Questions Answered

- Q6. Most Helpful reviews Per year Per Products (Helpful percentage >75%)
  - To determine the Item per year with most helpful reviews above 75%.
  - Used outer join as all reviewers will be present in ratings dataset. Right outer join is used as ratings is bigger table.
  - Grouped all the Items against each year with avg helpful percentage above 75%, choosing the item with most helpful reviews against each year.

```
SELECT r.ratingYear,r.item,r.avg_helpful_perc

FROM  (SELECT rt.ratingYear,rt.item,rt.avg_helpful_perc,ROW_NUMBER() OVER(PARTITION BY rt.ratingYear ORDER BY rt.avg_helpful_perc DESC) AS helpful_perc_rank

FROM (SELECT rat.ratingYear,rat.item,AVG((helpful[0] / helpful[1]) * 100) AS avg_helpful_perc

FROM reviews rev

RIGHT OUTER JOIN ratings rat

ON rev.asin = rat.item AND rev.reviewerID = rat.user

GROUP BY rat.ratingYear,rat.item ) rt

WHERE rt.avg_helpful_perc > CAST(75.0 AS FLOAT) ) r

WHERE r.helpful_perc_rank = 1;
```

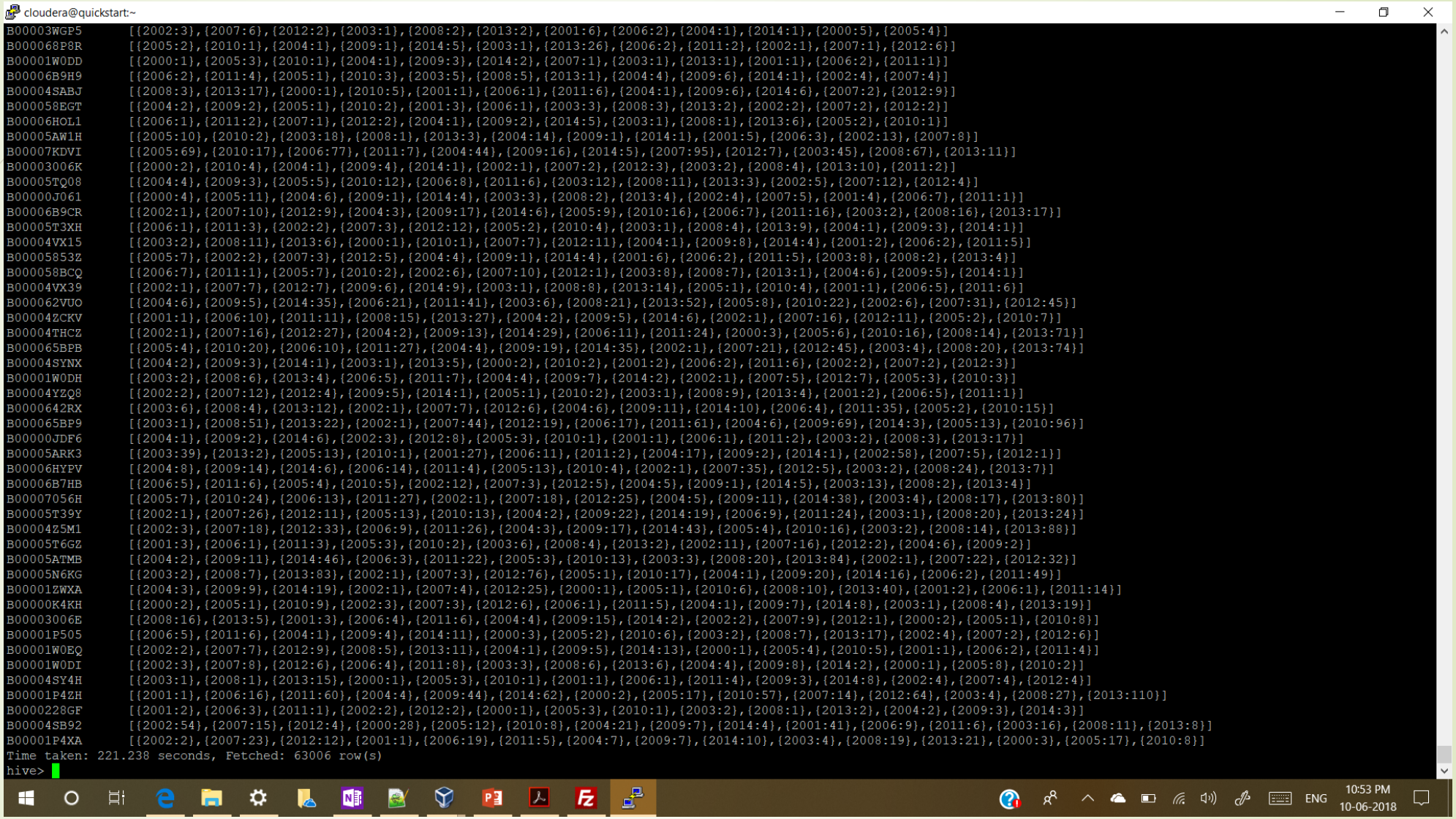


Sample output screenshot

Business Questions Answered

- Q7. Growth of review comments on products/year
  - To determine the growth of the review comments against each product every year.
  - Transposed review count of each year against every product using Map function and group by.
  - Grouped each product with a map of review counts against each year.

```
SELECT r.asin,collect_list(r.year_count_map) AS Year_count
FROM  (SELECT rev.asin,map(rev.reviewYear,COUNT(rev.reviewerID)) AS year_count_map
FROM reviews rev
GROUP BY rev.reviewYear,rev.asin) r
GROUP BY r.asin;
```



Sample output screenshot

Business Questions Answered

- Q8. Visualization of results observe
  - Reviews data is a subset of ratings dataset with additional details of review comments, helpful, summary, etc,.
  - Reviews adoption improved drastically from year 2007-2008.
  - Ratings adoption improved drastically from year 2006-2007.
  - Number of reviewer's comments per year started to multiple from year 2007, but in 2014 there was a drop in number unique reviewers.
  - Number of users per year started to multiple from year 2004, but in 2014 there was drop in number unique users.
  - As a trend number items reviewed each year is directly proportional to the number of users, same trend can be seen even for the drop in number of reviews during year 2014.

Conclusion

- Amazon Electronics Products data loaded and analysed successfully.
- Created an Database in Hive and loaded Reviews/Rating data as hive tables.
- Partitioned the Hive tables to answer the business questions using Hive queries.
- Code: https://github.com/ssushmanth/AmazonElectronicsAnalysis-Hive