Data Analysis and Visualization Using Hive and Tableau

PROJECT II

Project Members:

Neelam Kumari (8808 6618)

Rajat Koujalagi (6138 2944)

Group Member Contribution:

Neelam Kumari: Worked on enron dataset and documentation

Rajat Koujalagi: Worked on Netflix dataset and visualization

Data Sets Used:

1. Enron dataset
2. Netflix dataset

DATA PROCESSING

NETFLIX DATA ANALYSIS:

Loading Data:

CREATE EXTERNAL TABLE IF NOT EXISTS titles(mi INT, yearOfRelease INT, title STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION 's3n://spring-2014-ds/movie_dataset/movie_titles/';

CREATE EXTERNAL TABLE IF NOT EXISTS ratings(mid INT, customer_id INT, rating INT,date STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION 's3n://spring-2014-ds/movie_dataset/movie_ratings/';

Hypothesis 1: As shown in Fig1, Internet was not prevalent during early 20th century and not many movies were released. As the time passed by the higher number of movies were released and also internet became more prevalent amongst people which enabled them to rate the movies. As shown in Fig2, in late 20th century and early 21st century the average rating again goes down, which portrays that as more number of movies are made the quality went down.

```
SELECT netflix_join.yearofrelease, avg(netflix_join.rating) AS rating
FROM (   SELECT *  FROM titles JOIN ratings ON (titles.mi = ratings.mid) ) netflix_join
GROUP BY netflix_join.yearofrelease;
```

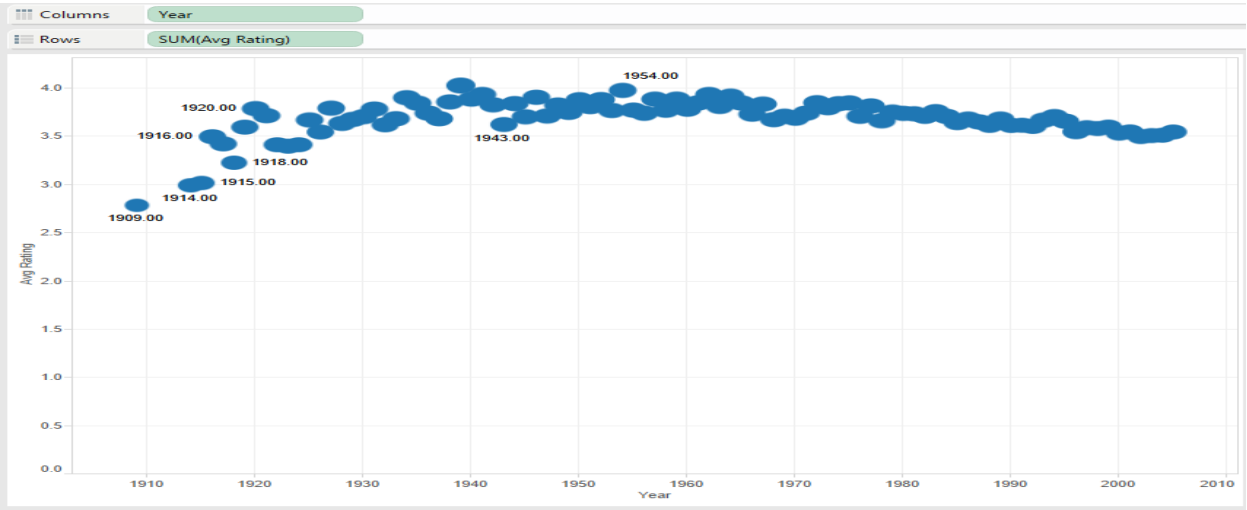

Fig1: Average Rating of the movies from 20th till 21st century

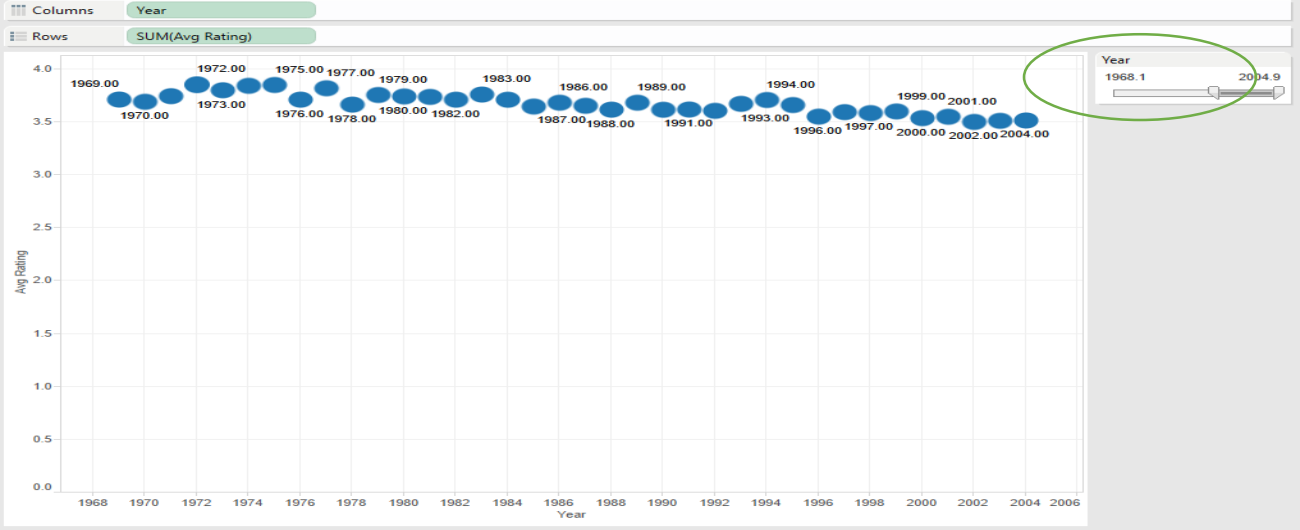

Fig2: Average Rating of the movies with selected range encircled above.

Hypothesis 2: With the passage of time from 20th century to present, the entertainment market has proved to be huge business market as more and more people watch movies as a break from their busy work life. Hence we see the drastic rise in number of movies released yearwise in Fig3, in the most recent times.

SELECT yearofrelease, count(*) FROM titles GROUP BY yearOfRelease;

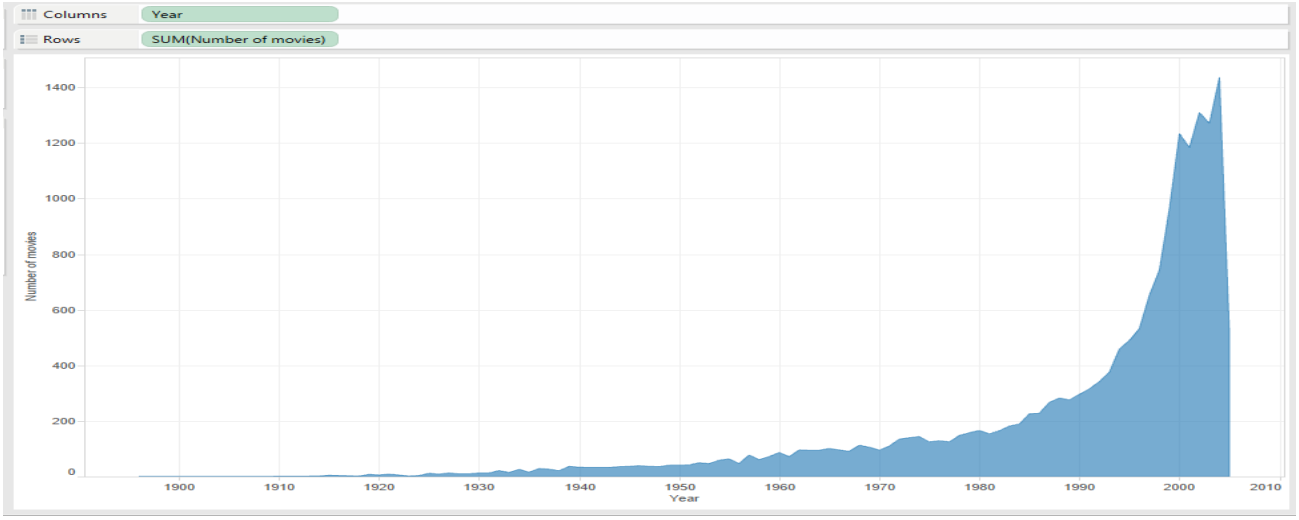

Fig 3: Number of Movies Per year

Hypothesis 3: Higher the number of internet users higher would be the number of votes for movies per year. Huge chunk of population rely on the average movie ratings in order to decide whether or not to watch the newly released movie. Hence as the number of votes increases the average rating would be more and more accurate. From Fig4 we can see that number of people voting has increased drastically from 1990 to 2010 which is because of high internet usage.

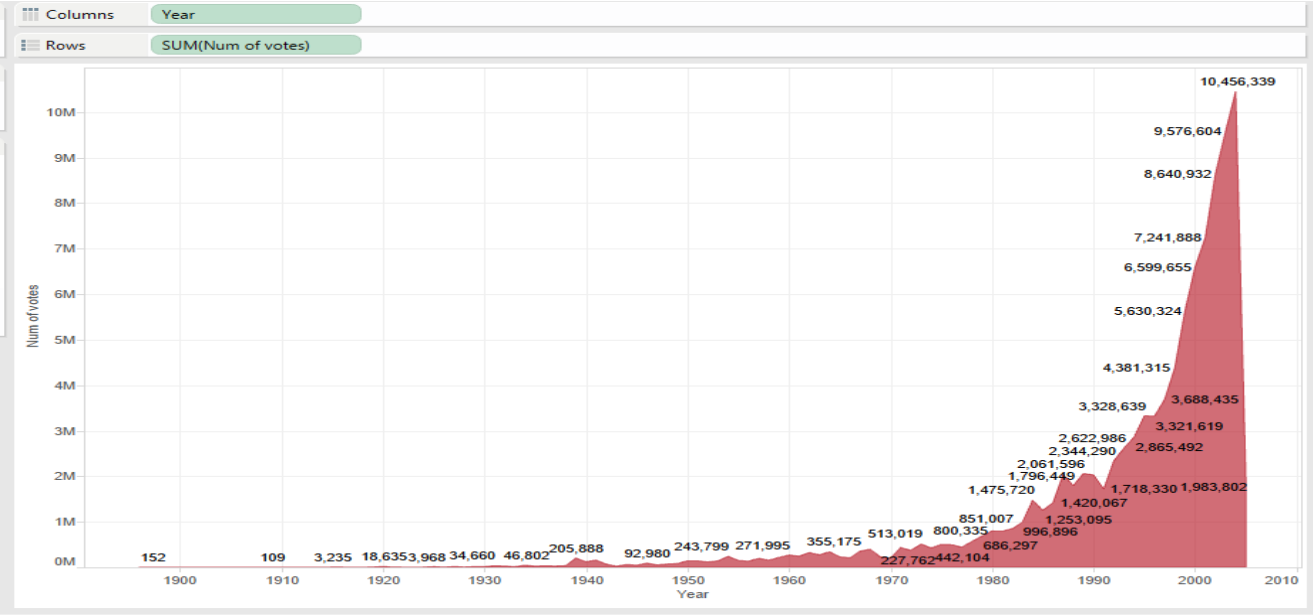

Fig4: Number of votes per year

Other interesting Queries:

3. **TOP 10 movies between 2000 till 2010**

```
SELECT result.title, result.rating  FROM (SELECT title_ratings_join.title AS title,
avg(title_ratings_join.rating) AS rating FROM ( SELECT *
  FROM titles JOIN ratings ON (titles.mi = ratings.mid)   ) title_ratings_join
WHERE title_ratings_join.yearOfRelease >=2000 AND title_ratings_join.yearOfRelease <2010
GROUP BY title_ratings_join.title ) result ORDER BY result.rating DESC LIMIT 10;
```

4. **MOVIES released before 1970 but rated in 2000's**

```
select distinct(title), yearOfRelease from titles join ratings on titles.mi=ratings.mid
where titles.yearOfRelease<1970 and ratings.date like '2%';
```

5. **Movies which have average rating above 3 with the 1,00,000 votes**

```
SELECT titles.title from titles join ratings on (titles.mi=ratings.mid) where ratings.rating > 3 group
by titles.title having count(ratings.customer_id) > 100000;
```

6. **Top 10 movies throughout the whole period**

```
SELECT result.title, result.rating FROM (    SELECT title_ratings_join.title AS
title,avg(title_ratings_join.rating) AS rating   FROM (    SELECT *  FROM titles JOIN ratings ON
(titles.mi = ratings.mid)) title_ratings_join    GROUP BY title_ratings_join.title ) result    ORDER
BY result.rating DESC LIMIT 10;
```

ENRON DATA ANALYSIS:

Loading Data Set

```
CREATE TABLE enron(id string, times string,fromheader String, toheader Array<String>,cc
Array<String>,Subject String, context String)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
COLLECTION ITEMS TERMINATED BY ','
LOCATION 's3n://spring-2014-ds/enron_dataset/';
```

Hypothesis 1: The number of email communication approaches when scandal occurred.

```
insert overwrite directory 's3n://hivedsbucket/result'
select regexp_extract(times,'([a-zA-Z]{3}, )([1-3]?[0-9] )([a-zA-Z]{3} )([0-9]{4} )(.*?)',4),
regexp_extract(times,'([a-zA-Z]{3}, )([1-3]?[0-9] )([a-zA-Z]{3} )([0-9]{4} )(.*?)',3), count(distinct(id)) as
total_mail  from enron_to
group by regexp_extract(times,'([a-zA-Z]{3}, )([1-3]?[0-9] )([a-zA-Z]{3} )([0-9]{4} )(.*?)',4),
regexp_extract(times,'([a-zA-Z]{3}, )([1-3]?[0-9] )([a-zA-Z]{3} )([0-9]{4} )(.*?)',3);
```

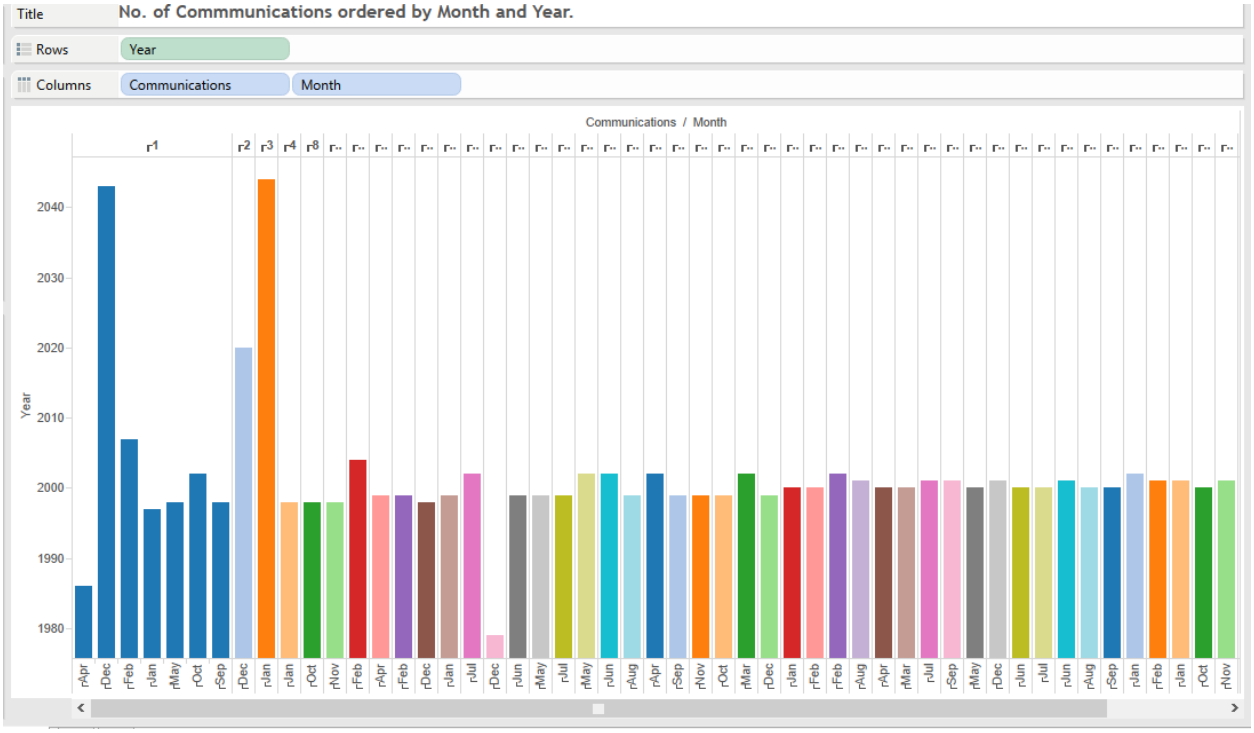

Fig 5: Number of email communication

Hypothesis 2: Popular subject in the conversations

```
insert overwrite directory 's3n://hivedsbucket/result'
select rtrim(ltrim(regexp_replace(subject,'RE:|FW:',''))), count(*) from enron
```

group by rtrim(ltrim(regexp_replace(subject,'RE:|FW:','')));

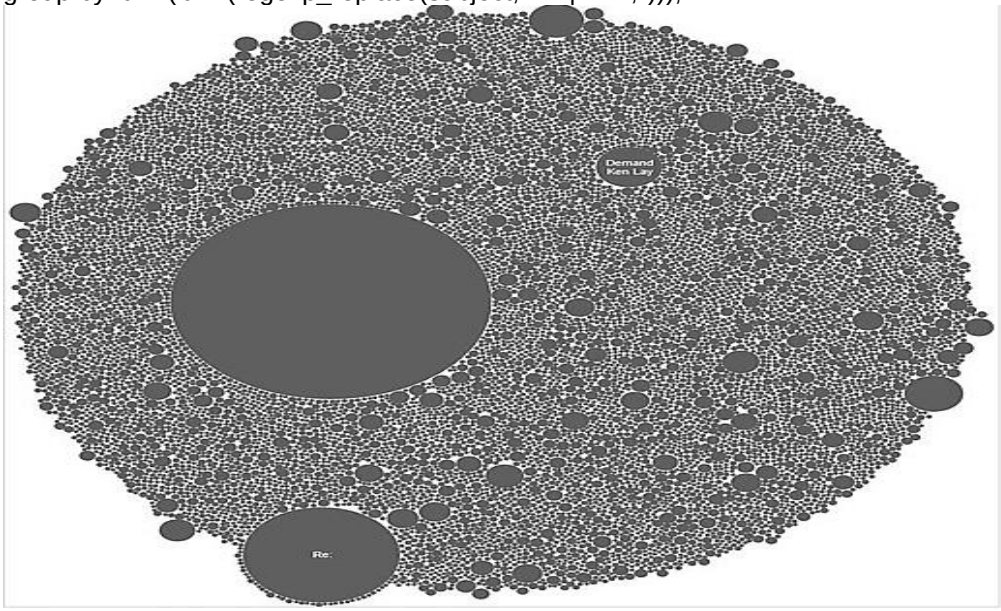

Fig 6: Maximum emails from person to person

Hypothesis 3: Maximum email from person to person

```
SELECT s1.fromheader, s1.toheader, count(*) as num
   FROM enron s1
   GROUP BY s1.fromheader, s1.toheader
   ORDER BY num DESC
   LIMIT 5;
```

pete.davis@enron.com ["pete.davis@enron.com"] 9141
vince.kaminski@enron.com ["vkaminski@aol.com"] 4308
enron.announcements@enron.com ["all.worldwide@enron.com"] 2206
enron.announcements@enron.com ["all.houston@enron.com"] 1701
kay.mann@enron.com ["suzanne.adams@enron.com"] 1528

The result portrays that Pete Davis has sent maximum number of emails to himself.

Hypothesis 4: **Count number of threads for each enron employee**

```
select count(fromheader), count(subject) from enron where lower(subject) like 're%' or lower(subject) like
'fw%' group by fromheader, subject;
```

LESSONS LEARNT:

1. Set operations such as INTERSECTION, UNION and EXCEPT are not present which makes querying a bit difficult.
2. HIVE version installed on AWS did not support JOINS such as right join
3. Sub queries could not be used with the FROM and WHERE clauses, hence making it mandatory to use EXISTS in order to perform any queries.
   The Version of Hive available on AWS had problems with use of joins, specially self joins.