Solution – Exercise 02

MapReduce in Java, Hive, HiveQL

Solution

Prerequisites:

- Download, Install and Setup Hadoop and YARN (previous lecture)
- Download, Install and Setup Apache Hive
- Start HDFS, YARN and Hive CLI

Solution

Exercise 1-4:

1. Download and unzip https://datasets.imdbws.com/name.basics.tsv.gz

```
wget https://datasets.imdbws.com/name.basics.tsv.gz
gunzip name.basics.tsv.gz
```

2. Create HDFS directory **/user/hadoop/imdb/actors/**names.tsv for file name.basics.tsv

```
hadoop fs -mkdir /user/hadoop/imdb/actors/
```

3. Create HDFS directory **/user/hadoop/imdb/actors/**names.tsv for file name.basics.tsv

```
hadoop fs -put name-basics.tsv /user/hadoop/imdb/actors/names.tsv
```

Solution

Exercise 1-4:

4. Create Hive Table `imdb_actors`:

```
hive >    CREATE EXTERNAL TABLE IF NOT EXISTS imdb_actors(
          nconst STRING,
          primary_name STRING,
          birth_year INT,
          death_year STRING,
          primary_profession STRING,
          known_for_titles STRING
          ) COMMENT 'IMDb Actors' ROW FORMAT DELIMITED FIELDS TERMINATED BY '
\t' STORED AS TEXTFILE LOCATION '/user/hadoop/imdb/actors';
```

Solution

Exercise 5:

a) How many movies are within the IMDB dataset?

```
hive >     SELECT count(*) FROM imdb_movies m WHERE m.title_type = 'movie'

499.052
```

b) Who is the oldest actor/writer/… within the dataset?

```
hive >    SELECT * FROM imdb_actors a
          WHERE a.birth_year = (SELECT MIN(birth_year) FROM imdb_actors )
```

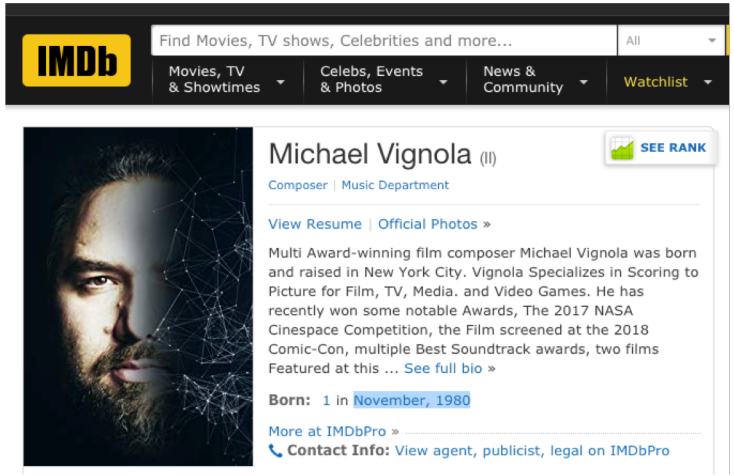
Solution

Exercise 5:

b) Who is the oldest actor/writer/… within the dataset?

a.nconst	a.primary_name	a.birth_year	a.death_year	a.primary_profession	a.known_for_titles
nm8572003	Michael Vignola	1	[NULL]	composer,music_department	tt6417824,tt4600298,tt4099244,tt6998038

Well, thats atually a bug within IMDB data:



Solution

Exercise 5:

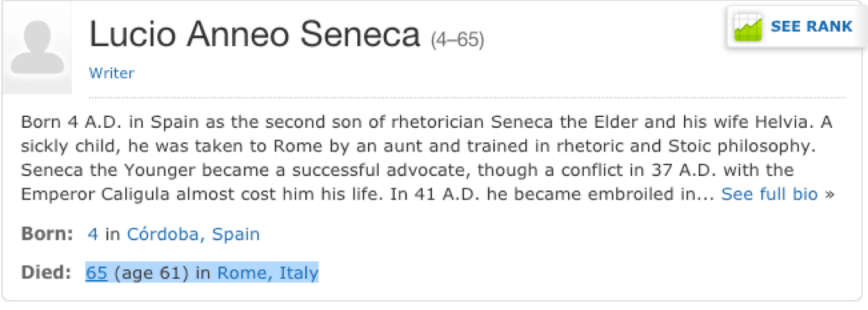
b) Who is the oldest actor/writer/… within the dataset?

Better go with:

```
hive >      SELECT * FROM imdb_actors a
            WHERE a.birth_year =
            (SELECT MIN(birth_year) FROM imdb_actors WHERE  birth_year > 1)
```

Lucio Seneca seems to be the oldest (without trash data)

a.nconst	a.primary_name	a.bi	a.death_year	
nm0194670	Célina Cély	4	[NULL]	ac
nm0784172	Lucio Anneo Seneca	4	0065	wr



Solution

Exercise 5:

c) Create a list (`m.tconst, m.original_title, m.start_year, r.average_rating, r.num_votes`) of movies which are:
- equal or newer than year 2000
- have an average rating better than 8
- have been voted more than 100.000 times

```
hive >      SELECT m.tconst, m.original_title, m.start_year, r.average_rating, r.num_votes
            FROM imdb_movies m JOIN imdb_ratings r on (m.tconst = r.tconst)
            WHERE r.average_rating > 8 and m.start_year >= 2000 and m.title_type = 'movie'
            and r.num_votes > 100000
            ORDER BY r.average_rating desc, r.num_votes DESC
```

m.tconst	m.original_title	m.start_year	r.average_rating	r.num_votes
tt0468569	The Dark Knight	2.008	9,0	1.969.110
tt0167260	The Lord of the Rings: The Return of the King	2.003	8,9	1.424.076
tt1375666	Inception	2.010	8,8	1.749.822
tt0120737	The Lord of the Rings: The Fellowship of the Ring	2.001	8,8	1.440.978
tt0167261	The Lord of the Rings: The Two Towers	2.002	8,7	1.287.434
tt0816692	Interstellar	2.014	8,6	1.213.141

Solution

Exercise 5:

d) How many movies are in list of c)?

```
hive >      SELECT count(*)
            FROM imdb_movies m JOIN imdb_ratings r on (m.tconst = r.tconst)
            WHERE r.average_rating > 8 and m.start_year >= 2000 and m.title_type = 'movie'
            and r.num_votes > 100000

86
```

Solution

Exercise 5:

e) We want to know which years have been great for cinema.
Create a list with one row per year and a related count of movies which:
- have an average rating better than 8
- have been voted more than 100.000 times
ordered descending by count of movies.

```
hive >      SELECT m.start_year, count(*)
            FROM imdb_movies m JOIN imdb_ratings r on (m.tconst = r.tconst)
            WHERE r.average_rating > 8 and m.title_type = 'movie'
            and r.num_votes > 100000
            GROUP BY m.start_year
            ORDER BY count(*) DESC
```

123 m.start_year	123 _c1
1.995	8
2.014	7
2.009	6
2.001	6
2.004	6
2.011	5
2.010	5
2.002	5
2.000	5
1.999	5
1.998	5
2.016	5
1.994	5
1.988	5