Solution – Exercise II

Hive, HiveQL

Solution

Prerequisites:

- Setup Google Cloud SDK
- Start VM instance
- Pull docker container `marcelmittelstaedt/hive_base:latest`
- Start docker container: `docker run -dit --name hive_base_container -p 8088:8088 -p 9870:9870 -p 9864:9864 marcelmittelstaedt/hive_base:latest`
- Get into docker container
- Start Hadoop and Hive Shell:
  - `start-all.sh`
  - `hive`

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/name_basics/** for file name.basics.tsv

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

3. Put TSV file to HDFS:

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


Solution

Exercise 1-4:

4. Create Hive Table `name_basics`:

```
hive > CREATE EXTERNAL TABLE IF NOT EXISTS name_basics(
    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' ST
ORED AS TEXTFILE LOCATION '/user/hadoop/imdb/name_basics'
TBLPROPERTIES ('skip.header.line.count'='1');
```

Solution

Exercise 5:

a) How many movies and how many TV series are within the IMDB dataset?

```
hive >  SELECT m.title_type, count(*)
        FROM title_basics m GROUP BY m.title_type;

tvMovie 120813
movie 532594
tvEpisode 4366605
tvSeries 172627
[…]

Time taken: 36.261 seconds, Fetched: 10 row(s)
```

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

```
hive >  SELECT * FROM name_basics n
        WHERE n.birth_year = ( SELECT MAX(birth_year) FROM name_basics);
```

Solution

Exercise 5:

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

```
hive >  SELECT * FROM name_basics n
        WHERE n.birth_year = ( SELECT MAX(birth_year) FROM name_basics);
nm10913258 Shea Lightfoot 2019 NULL actor NULL
Time taken: 66.858 seconds, Fetched: 1 row(s)
```

Well, thats actually a bug within IMDB data:

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 2010
- have an average rating equal or better than 8,1
- 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 title_basics m JOIN title_ratings r on (m.tconst = r.tconst)
          WHERE r.average_rating >= 8.1 and m.start_year >= 2010 and m.title_type = 'movie'
          and r.num_votes > 100000
          ORDER BY r.average_rating desc, r.num_votes DESC;

tt7286456 Joker 2019 9.0 240216
tt5813916 Dag II 2016 9.0 101524
tt1375666 Inception 2010 8.8 1880162
tt0816692 Interstellar 2014 8.6 1336209
tt4154756 Avengers: Infinity War 2018 8.5 714459
tt1675434 Intouchables 2011 8.5 692986
tt2582802 Whiplash 2014 8.5 635438
tt4154796 Avengers: Endgame 2019 8.5 575421
[…]
```


Solution

Exercise 5:

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

```
hive >    SELECT count(*)
          FROM title_basics m JOIN title_ratings r on (m.tconst = r.tconst)
          WHERE r.average_rating >= 8.1 and m.start_year >= 2010 and m.title_type = 'movie'
          and r.num_votes > 100000;

39
```

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 title_basics m JOIN title_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;

1995 8
2014 6
2009 6
2000 6
1998 5
2004 5
[…]
```