ASSIGNMENT-HIVE

In the zip provided, you will find two files named movies.json and persona.json. They contain a small movie database in the JSON format. Here is an example record from movies.json (both files actually have one record per line, newlines are added here for readability):

```
{

"id": "movie:14",
"title": "Se7en",
"year": 1995,
"genre": "Crime",
"summary": " Two detectives, a rookie and a veteran, hunt a
serial killer who uses the seven
deadly sins as his modus operandi.",
"country": "USA",
"director":    {
"id": "artist:31",
"last_name": "Fincher",
"first_name": "David",
"year_of_birth": "1962"
},
"actors": [
     {
     "id": "artist:18",
     "role": "Doe"
     },
     {
     "id": "artist:22",
     "role": "Somerset"
     },
     {
     "id": "artist:32",
     "role": "Mills"
     }
     ]
}
```

And here is an example record from persona.json:

```
{
"id": "artist:18",
"last_name": "Spacey",
"first_name": "Kevin",
"year_of_birth": "1959"
}
```

As you can see, movies.json contains the full names and years of birth of movie directors, but only the identifiers of actors. The full names and years of birth of all artists, as well as their identifier, are listed in persona.json.

Unfortunately, Hive doesn't support JSON natively. Using the following Serializer/Deserializer (SerDe), create two tables with similar structures as the JSON structure that you will populate with the data from the JSON files:

https://github.com/rcongiu/Hive-JSON-Serde

Hint: you can keep the column names from the JSON files if it makes things easier.

1. Connect to the Hadoop cluster and copy the two files movies.json and persona.json to the HDFS. Run Hive. We want to load the two files into two relations named movies and artists, respectively. How can we load a JSON file into a relation? Start with the easier case: persona.json. We want a query with four fields: id, firstName, lastName, and yearOfBirth. All fields will be of type varchar. Load movies.json into a table with the following fields (all fields except director and actors are of type varchar):

   ```
   id, title, year, genre, summary, country,
   director: (id, lastName, firstName, yearOfBirth),
   actors: {(id, role)}
   ```

2. We want to get rid of the descriptions of the movies to simplify the output of the next questions. How can we modify the **movies** relation to remove descriptions?

3. Create a query that groups the titles of American movies by year. Your results should look like the following:

   ```
   [...]
   (1988,{(Rain Man),(Die Hard)})
   (1990,{(The Godfather: Part III),(Die Hard 2),(The
   Silence of the Lambs),(King of New York)})
   (1992,{(Unforgiven),(Bad Lieutenant),(Reservoir
   Dogs)})
   (1994,{(Pulp Fiction)})
   [...]
   ```

4. Write a query that groups the titles of American movies by director. Your results should look like this:

   ```
   ((artist:1,Coppola,Sofia,1971),{(Marie Antoinette)})
   ```

```
((artist:3,Hitchcock,Alfred,1899),{(Psycho),(North by
Northwest),(Rear Window),(Marnie),(The
Birds),(Vertigo)})
((artist:4,Scott,Ridley,1937),{(Alien),(Gladiator),(Bl
ade Runner)})
((artist:6,Cameron,James,1954),{(The
Terminator),(Titanic)})
[...]
```

5. Write a query that contains (movieId, actorId, role) tuples. Each movie will appear in as many tuples as there are actors listed for that movie. Should work only for American movies. Your results should look like the following:

```
(movie:1,artist:15,John Ferguson)
(movie:1,artist:16,Madeleine Elster)
(movie:2,artist:5,Ripley)
(movie:3,artist:109,Rose DeWitt Bukater)
[...]
```

6. Write a query that associates the ID of an American movie to the full description of each of its actors. Your results should look like the following:

```
(movie:67,artist:2,MarieAntoinet
te,Dunnst,Kirsten,)
(movie:2,artist:5,Ripley,Weaver,
Sigourney,1949)        (movie:5,
artist:  11,  Sean  Archer/Castor
Troy, Travolta , John, 1954)
(movie:17,artist:11,VincentVega,
Travolta,John,1954)
[...]
```

7. Write a query that lists, for each personality, the list of American movies which he directed and in which he played. The full name of the artist as well as their year of birth should be made available. You should get, for instance:

```
[...]
(artist:20,Eastwood,Clint,1930,artist:20,
{(movie:8,Unforgiven,artist:2
,William Munny),
(movie:63,Million Dollar
Baby,artist:20,Frankie Dunn),
(movie:26,AbsolutePower,artist
:20,Luther Whitney)},
{(movie:26,Absolute Power,artist:20),
(movie:63,Million Dollar Baby,artist:20),
(movie:8,Unforgiven,artist:20)})
[...]
```

8. Filter UDF We now want to list the title and director of all movies for which their director's first name and last name start with the same letter. To do so, you will need to write a Filter User-Defined Function (Filter UDF).
   a. Create a project in Eclipse with one class for your Filter UDF. What JAR(s) will you have to add? Which version? Where do you get them? What will you do to upload and register the JAR to hadoop?
   b. How do you make it so that Hive finds your JAR? How do you refer to your Filter UDF to use it?

   You should get the following output:

```
(Jaws,(artist:45,Spielberg,Steven,1946))
(The Lost World: Jurassic
Park,(artist:45,Spielberg,Steven,1946))
(Sound and Fury,(artist:138,Chabrol,Claude,1930))
```

9.
10. Eval UDF. We now want to list the title and the IMDB URL of the three movies found in the previous question. To do so, you will write an Eval User-Defined Function (Eval UDF) that queries IMDB's search function for a movie. For the movie "The Matrix Reloaded", for instance, it will access the following webpage:

    http://www.imdb.com/find?q=The+Matrix+Reloaded

    Warning: make sure that you add a call to *Thread.sleep(500)* after you access this URL to ensure that you don't query the IMDB servers too often, which could result in *your* IP getting blacklisted!

IMDB movie identifiers start with tt, followed by 7 digits. You can look for the first occurrence of such a string in the page (using a regular expression, for instance), and consider that it is the movie identifier of the first result. You can then construct the URL of the movie by using this identifier. For instance, for "The Matrix Reloaded", the corresponding IMDB URL will be:

http://www.imdb.com/title/tt0234215/

Your objective will be to obtain the following output:

```
(Jaws,http://www.imdb.com/title/tt0073195/)
(The Lost World: Jurassic
Park,http://www.imdb.com/title/tt0119567/)
(Sound and Fury,http://www.imdb.com/title/tt0240912/)
```

11. Load UDF. The file desc.txt from the archive contains the same information as the persona.json file, in a different format:

```
id=artist:1
last-name=Coppola
first-name=Sofia
year-of-birth=1971
--
id=artist:2
last-name=Dunst
first-name=Kirsten
year-of-birth=null
--
id=artist:3
last-name=Hitchcock
first-name=Alfred
year-of-birth=1899
...
```

Write a Load UDF that loads the file into a new table. Make sure that this relation contains the same data as artists.