Project: Data Analysis of Movie Genres and Directors by Country

Introduction

The objective of this project is to analyze the movie genres and directors across different countries to identify the top-rated movie genres for each country and the most successful directors in the corresponding region and sector. The major tasks involved in this analysis are determining the most popular movie genres for each country using Hive and determining the most successful director for each genre in each country using Hive. The helper tasks include data cleaning, data transformation, and data aggregation to extract the necessary data fields for analysis and group the movies by country, genre, and director. The final task involves integrating the two outputs from the major tasks to identify the most successful directors for the most popular movie genres in each country.

Here are some highlights of our project:

- Successfully installing and running Hive on local machine: We started our project with zero knowledge of Hive, but we were able to overcome various installation challenges and get it up and running on our local machine. This was a significant accomplishment for us and allowed us to dive deeper into analyzing the movie data.
- Writing a Java script to collect data from public API: To collect the country information that maps country code to country name, e.g., tt0002333 Germany, we wrote a Java program using multi-thread technique to collect the most up-to-date data from a public API provided by OMDb.
- Finding a stable method to upload large files to S3: We had a lot of trouble uploading large files to S3 at the beginning of our project, and this wasted a lot of time. But after some research and experimentation, we found a stable method that allowed us to upload large files without errors. This was a huge time saver and allowed us to focus on other aspects of the project.
- Analyzing large and interesting data: In our task, we are analyzing around 2GB movie data since 2010 to find out the best genre and director in each country, which is interesting and useful. In our analysis, we found some interesting data points that were unexpected. For example, we found that documentaries were one of the most popular genres in most countries around the world. This was surprising to us and made us think about the value of authenticity and realism in storytelling. It was also a reminder that sometimes the truth is fancier than fiction.

Overall, this project will provide insights into the most successful movie genres and directors across different countries, which can be useful for filmmakers, producers, and investors in the movie industry.

Data

There are 2 sources of our data:

A. IMDB.com

Most of our data comes from https://www.imdb.com/interfaces/, which contains various information related to movies and TV shows. It is stored in TSV format and consists of multiple files. Each file has a different number of records, from 1 million to 35 million, and in a total of approximately 109 million records. Each file has a different number of attributes, with the lowest number of 3, the highest 9. Properties include title, region, type, year, rating, director, etc. The dataset generally spans from the early 1900s to the present day. Some titles may have information dating back to the late 1800s.

B. OMDb API

There is no country information for each movie in the IMDB data however, so we have to pull the country tag of each movie from https://www.omdbapi.com/. We wrote a **Java** program to

I. get the data from the OMDb API and
II. performs data cleaning on the data to normalize country names so that different variations of the same country are represented consistently.

To speed up the program we started **200** multiple threads to download country data from the API, and the final results are stored in a tsv file called *title.country.tsv*.

(Source code: GetCountry.java)

Also, because a single tsv file is large (800 MB), the upload to s3 bucket is very unstable and it almost always failed due to Connection was closed before we received a valid response from endpoint URL: xxx; even worse, we had to restart the upload from scratch every time it failed, which is a huge waste of time. To resolve this problem, we split the original big file into multiple smaller sub-files by line and upload them one by one, there are 2 benefits:

1. Uploading a smaller file is much easier and more stable that uploading a large one
2. Even though the upload for few small files filed, we just need to re-upload those failed ones instead of re-do the whole thing for all.

Shell command to split file and upload to s3:

```
split -l 1000000 -d -a 4 name.basics.tsv ./test/name.basics.tsv.

for file in ./test/*; do
    if [[ $file == .* ]]; then
        continue
    fi
    aws s3 cp "$file" s3://liuni-6240/movie-data/
done
```

There are 5 tables in our dataset:

- Basics: the basic information of movies, such as: name, type, genres, etc.
  Source file: *title.basics.tsv*
- Crew: the crew of each movie, including directors and writers
  Source file: *title.crew.ts*
  NOTE: the directors and writers here are stored as name id instead of human name.
- Ratings: the rating of each movie, including average rating and number of votes

Source file: *title.ratings.ts*

- Countries: the country tag of each movie
  Source file: *title.country.ts*
- Names: the information of each person, such as name, birthday, etc.
  Source file: *name.basics.tsv*

Below is a diagram for the relation between tables in our dataset, the column with a green arrow pointed to means the same object can be tracked between 2 tables using this column; for example, if we want to know the directors of a movie in table *basics*, we can use its title id to retrieve table *crew* to find that out.

To help get a better view of our dataset, below are data snippets for each table:

```
select * from basics limit 10;
```

	titleid	titletype	primarytitle	originaltitle	isadult	startyear	endyear	runtimeminutes	genres
1	tt0000001	short	Carmencita	Carmencita	0	1894	<null>	1	["Documentary","Short"]
2	tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	<null>	5	["Animation","Short"]
3	tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	0	1892	<null>	4	["Animation","Comedy","Romance"]
4	tt0000004	short	Un bon bock	Un bon bock	0	1892	<null>	12	["Animation","Short"]
5	tt0000005	short	Blacksmith Scene	Blacksmith Scene	0	1893	<null>	1	["Comedy","Short"]
6	tt0000006	short	Chinese Opium Den	Chinese Opium Den	0	1894	<null>	1	["Short"]
7	tt0000007	short	Corbett and Courtney Before the Kinetograph	Corbett and Courtney Before the Kinetograph	0	1894	<null>	1	["Short","Sport"]
8	tt0000008	short	Edison Kinetoscopic Record of a Sneeze	Edison Kinetoscopic Record of a Sneeze	0	1894	<null>	1	["Documentary","Short"]
9	tt0000009	movie	Miss Jerry	Miss Jerry	0	1894	<null>	45	["Romance"]
10	tt0000010	short	Leaving the Factory	La sortie de l'usine Lumière à Lyon	0	1895	<null>	1	["Documentary","Short"]

```
select * from crew limit 10;
```

	titleid	directors	writers
1	tt0000001	["nm0005690"]	<null>
2	tt0000002	["nm0721526"]	<null>
3	tt0000003	["nm0721526"]	<null>
4	tt0000004	["nm0721526"]	<null>
5	tt0000005	["nm0005690"]	<null>
6	tt0000006	["nm0005690"]	<null>
7	tt0000007	["nm0005690","nm0374658"]	<null>
8	tt0000008	["nm0005690"]	<null>
9	tt0000009	["nm0085156"]	["nm0085156"]
10	tt0000010	["nm0525910"]	<null>

```
select * from ratings limit 10;
```

	titleid	rating	numvotes
1	tt0000001	5.7	1966
2	tt0000002	5.8	263
3	tt0000003	6.5	1803
4	tt0000004	5.6	179
5	tt0000005	6.2	2603
6	tt0000006	5.1	178
7	tt0000007	5.4	817
8	tt0000008	5.4	2100
9	tt0000009	5.3	204
10	tt0000010	6.9	7094

```
select * from countries limit 10;
```

	titleid	country
1	tt0001628	["United States"]
2	tt0002431	["France"]
3	tt0001051	["Spain"]
4	tt0002031	["United States"]
5	tt0002001	["United States"]
6	tt0002089	["United Kingdom"]
7	tt0002514	["Sweden"]
8	tt0002211	["Denmark"]
9	tt0001184	["Spain"]
10	tt0002333	["Germany"]

```
select * from names limit 10;
```

	nameid	primaryname	birthyear	deathyear	primaryprofession	knownfortitles
1	nm0000001	Fred Astaire	1899	1987	["soundtrack","actor","miscellaneous"]	["tt0045537","tt0050419","tt0072308","tt0053137"]
2	nm0000002	Lauren Bacall	1924	2014	["actress","soundtrack"]	["tt0117057","tt0075213","tt0037382","tt0038355"]
3	nm0000003	Brigitte Bardot	1934	<null>	["actress","soundtrack","music_department"]	["tt0057345","tt0056404","tt0054452","tt0049189"]
4	nm0000004	John Belushi	1949	1982	["actor","soundtrack","writer"]	["tt0080455","tt0078723","tt0072562","tt0077975"]
5	nm0000005	Ingmar Bergman	1918	2007	["writer","director","actor"]	["tt0050976","tt0083922","tt0050986","tt0060827"]
6	nm0000006	Ingrid Bergman	1915	1982	["actress","soundtrack","producer"]	["tt0036855","tt0038787","tt0034583","tt0038109"]
7	nm0000007	Humphrey Bogart	1899	1957	["actor","soundtrack","producer"]	["tt0042593","tt0043265","tt0037382","tt0034583"]
8	nm0000008	Marlon Brando	1924	2004	["actor","soundtrack","director"]	["tt0078788","tt0047296","tt0070849","tt0068646"]
9	nm0000009	Richard Burton	1925	1984	["actor","soundtrack","producer"]	["tt0061184","tt0057877","tt0087803","tt0059749"]
10	nm0000010	James Cagney	1899	1986	["actor","soundtrack","director"]	["tt0031867","tt0029870","tt0035575","tt0042041"]

Technical Discussion

Hive task1: Determine the most popular movie genres (top 3) for each country after year 2010

Solution

The main idea is to analyze movie data and calculate the weighted average rating for each genre within each country. The solution involves joining relevant tables, filtering and transforming the data, and then computing the required metrics. The design pattern used here is a series of Common Table Expressions (CTEs) that enable breaking down the complex query into smaller, more manageable parts. This approach allows for step-by-step data processing, which includes joining tables, filtering rows, exploding genres and countries, calculating the weighted average rating, and finally ranking the genres within each country.

Hive pseudo code (Source code: top_genre.hql)

```
Create a table named 'top_genre' with columns: country, genre, weightedAveRating, and ranking.
Create a temporary view named 'top_genre_temp' with the following steps:
    Create a CTE (Common Table Expression) called 'full_table' with the following steps:
        Join 'basics', 'countries', and 'ratings' tables on 'titleId' column.
        Filter the joined table with the specified conditions
            startYear >= 2010
            titleType = 'movie'
            non-null values
    Create a CTE called 'exploded_table' with the following steps:
        Explode 'genres' and 'country' columns from the 'full_table' to create new rows for each
        genre and country.
    Create a CTE called 'weighted_avg_scores' with the following steps:
        Group the results by country and genre.
        Calculate the weighted average rating for each country and genre combination.
    Create a CTE called 'ranked_scores' with the following steps:
        Rank the weighted average rating within each country.
        Use the RANK() function to calculate the ranking and partition by country.
    Select the final columns for the 'top_genre_temp' view
        Columns: country, genre, weightedAveRating, and ranking from the 'ranked_scores' CTE.
        Order the results by country and weightedAveRating in descending order.
Insert the data from the 'top_genre_temp' view into the 'top_genre' table.
Drop the temporary view 'top_genre_temp'.
```

Hive task2: Determine the most successful director for each genre for each country after year 2010

Solution

The main idea is to identify the best director for each genre within each country, based on the weighted rating. It applies the same strategy as the previous task, involves joining relevant tables, filtering and transforming the data, and then computing the required metrics using Hive.

Similar to the previous task, a series of Common Table Expressions (CTEs) is employed to break down the complex Hive query into smaller, more manageable parts. This approach allows for step-by-step data processing in Hive, which includes joining tables, filtering rows, exploding countries, genres, and directors

using LATERAL VIEW EXPLODE, calculating the weighted rating, and finally ranking the directors within each country and genre using window functions.

Hive pseudo code (Source code: best_director.hql)

```
Create a table named 'best_director' with columns: country, genre, director_name, and weighted_rating.
Create a temporary view named 'best_director_tmp' with the following steps:
    Create a CTE (Common Table Expression) called 'all_info' with the following steps:
        Join 'basics', 'countries', 'crew', and 'ratings' tables on 'titleId' column.
        Filter the joined table with the specified conditions (titleType = 'movie', startYear >= 2010, and non-null values).
    Create a CTE called 'exploded_all_info' with the following steps:
        Explode 'countries', 'genres', and 'directors' columns from the 'all_info' to create new rows for each country,
        genre, and director.
        Filter the exploded_all_info to exclude rows with genre 'movie'.
    Create a CTE called 'weighted_rating_agg' with the following steps:
        Group the results by country, genre, and director.
        Calculate the weighted rating for each country, genre, and director combination.
    Create a CTE called 'director_rank' with the following steps:
        Rank the weighted rating within each country and genre.
        Use the RANK() function to calculate the ranking and partition by country and genre.
    Select the final columns for 'best_director_tmp' view from the 'director_rank' CTE
        Columns: country, genre, director_name, and weighted_rating
        Join with the 'names' table on 'nameId'.
        Filter the results to include only the top-ranked director in each country and genre combination.
        Order the results by country and genre.
Insert the data from the 'best_director_tmp' view into the 'best_director' table.
Drop the temporary view 'best_director_tmp'.
```

Hive helper task: Integrates the two outputs from the two major tasks

Solution

It aims to find the top 3 best directors for each genre in each country, based on the ranking. The script first combines the data from the 'top_genre' and 'best_director' tables achieved from the above two Hive tasks, then it applies window functions, DENSE_RANK() and ROW_NUMBER(), to rank and filter the top 3 results within each country. Finally, the results are inserted into the 'best_director_for_genres' table.

Hive pseudo code (Source code: final_result.hql)

```
Create a table named 'best_director_for_genres' with columns: country, genre, ranking, and best_directors (an array of strings).
Create a view named 'result_temp' with the following steps:
    Create a CTE (Common Table Expression) called 'combined_data' with the following steps:
        Join 'top_genre' and 'best_director' tables on country and genre columns.
        Group the joined table by country, genre, and ranking.
        Collect the list of best directors using COLLECT_LIST function.
    Create a CTE called 'ranked_data' with the following steps:
        Apply DENSE_RANK() window function to rank the data within each country based on the original ranking.
        Apply ROW_NUMBER() window function to generate row numbers within each country based on the original ranking.
        Include the best_directors column in the output.
    Select the final columns for 'result_temp' view from the 'ranked_data' CTE
        Columns: country, genre, ranking, and best_directors.
        Filter the results to include only the top 3 rows within each country.
        Order the results by country and ranking.
Insert the data from the 'result_temp' view into the 'best_director_for_genres' table.
Drop the view 'result_temp'.
```

Pig helper task: Write alternative Pig program

Solution

The main idea of the solution is to find the top genres and directors for each country based on the weighted average rating of their movies. The script employs Pig Latin and the design pattern used here is a series of transformations on the input data (filtering, joining, grouping, and flattening) to derive the desired output. The results are then combined and sorted to produce the top genres and directors for each country. The final output is stored in a specified directory.

Pig pseudo code (Source code: best_director_genre.pig)

```
Load and preprocess data
    Load data from TSV files into variables
    Filter and join tables to create a single dataset with all necessary information
Calculate weighted average ratings for genres and directors
    Generate a table with flattened genres, countries, and directors
    Group data by country and genre to calculate weighted average rating for each genre
    Group data by country, genre, and director to calculate the weighted average rating for each director
Find the top genres and directors
    Find the top genres for each country based on the weighted average ratings
    Find the top directors for each country and genre based on the weighted average ratings
    Create a bag of directors for each country and genre
Combine and sort results
    Join the top genres and director bags to create a combined result
    Group the result by country and select the top 3 rows for each group
Store the final output
    Store the final result into a specified output directory
```

Results

Cluster setup: 1 master (m5.xlarge) + 10 core nodes (m4.large)

Name	Step type	Status	Log files	Start time (UTC-07:00)	Elapsed time
final result	Hive script	Completed	controller | syslog | stderr | stdout	April 22, 2023 at 14:40	44 seconds
best director	Hive script	Completed	controller | syslog | stderr | stdout	April 22, 2023 at 14:33	2 minutes, 4 seconds
top genre	Hive script	Completed	controller | syslog | stderr | stdout	April 22, 2023 at 14:31	1 minute, 28 seconds
load data	Hive script	Completed	controller | syslog | stderr | stdout	April 22, 2023 at 14:30	50 seconds
pig	Pig script	Completed	controller | syslog | stderr | stdout	April 22, 2023 at 14:23	3 minutes, 42 seconds

As the results below shows, the pig script is a bit faster than the total runtime hive script above (3 min 42 sec v.s. 5 min 6 sec), to get rid of the impact introduced by startup of EMR step, we integrate all hive scripts into one (see: best_director_genre.hql) and ran both pig and hive again, below is the result:

Name	Step type	Status	Log files	Start time (UTC-07:00)	Elapsed time
hive	Hive script	Completed	controller | syslog | stderr | stdout	April 22, 2023 at 17:22	4 minutes
pig	Pig script	Completed	controller | syslog | stderr | stdout	April 22, 2023 at 17:18	3 minutes, 36 seconds

Detailed performance related files are put in zip file, whose name is Hive/Pig.log.

So, after removing the impact from starting the EMR step, the total runtime of hive script goes down from 5 mins to 4 mins, which is a large improvement.

Below are our findings based on the output:

- The Pig script runs faster than Hive and consumes less lines of code, this is because Pig Latin is a high-level data processing language that is designed to be easy to learn and use for data analysis and processing tasks, therefore it is easier to write the Pig script than Hive to achieve the same goal.
- Although Hive runs slower than Pig, it is easier for us to check the results than Hive, because the results are stored in a table and queryable, meaning we can write query to analyze the results instead of having to reading it through.
- Documentary is one of the top genres in most countries (128 out of 228 countries), which is really interesting because we thought it might be comedy or family movie, but seems people enjoy unvarnished depictions of reality.

  We put the output result in best_genre_director_output.txt.

Setup Challenges

Challenge 1

- Problem: No such method error

  We met this when we tried to start hive interface by running command hive, below is the error detail:

```
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Exception in thread "main" java.lang.NoSuchMethodError: com.google.common.base.Preconditions.checkArgument(ZLjava/lang/String;Ljava/lang/Object;)V
        at org.apache.hadoop.conf.Configuration.set(Configuration.java:1357)
        at org.apache.hadoop.conf.Configuration.set(Configuration.java:1338)
        at org.apache.hadoop.mapred.JobConf.setJar(JobConf.java:536)
        at org.apache.hadoop.mapred.JobConf.setJarByClass(JobConf.java:554)
        at org.apache.hadoop.mapred.JobConf.<init>(JobConf.java:448)
        at org.apache.hadoop.hive.conf.HiveConf.initialize(HiveConf.java:5141)
        at org.apache.hadoop.hive.conf.HiveConf.<init>(HiveConf.java:5099)
        at org.apache.hadoop.hive.common.LogUtils.initHiveLog4jCommon(LogUtils.java:97)
        at org.apache.hadoop.hive.common.LogUtils.initHiveLog4j(LogUtils.java:81)
        at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:699)
        at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:683)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:323)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:236)
```

- Solution:

  According to the error message, the method *com.google.common.base.Preconditions.checkArgument* belongs to guava.jar, then we checked the guava.jar in both Hadoop and hive directory and found that they are using different versions of guava. So, we upgraded the guava in hive by replacing it with the one from Hadoop which has higher version number. After doing that the problem got resolved.

Challenge 2

- Problem: User * is not allowed to impersonate anonymous

  We met this issue when we tried to connect to hive server using beeline, after typing in username and password, it suddenly told us that we are not allowed to impersonate anonymous and exited.

- Solution:

  That's because by default hive tries to execute operations as the calling user, therefore we added the below lines to hive config file conf/hive-site.xml (or modified the value if this property has already been there in your config file), to ask hive to execute operations as the hiveserver2 process user, then get rid of this error was resolved.

```
<property>
  <name>hive.server2.enable.doAs</name>
  <value>false</value>
  <description>
    Setting this property to true will have HiveServer2 execute
    Hive operations as the user making the calls to it.
  </description>
</property>
```

Challenge 3

- Problem: The UI of Hive interface is not easy to use

  The user interface of hive in command line tool is good but not user friendly. To be more detailed:

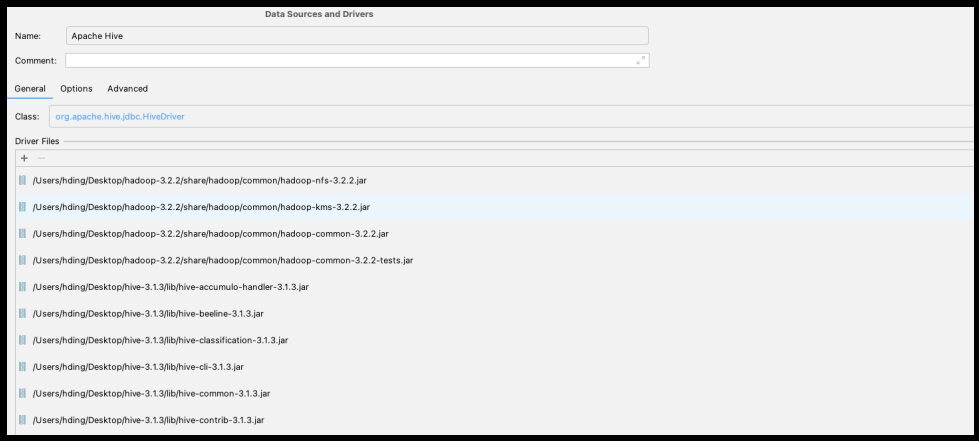
  1. It is not easy to retrieve historical queries, we have to press up key until found the target query
  2. It is not easy to modify a query with multiple lines.
  3. It is not easy to check the output because they are not well formatted, below is an example:

```
hive> select * from basics limit 10;
OK
tt0000001       short   Carmencita      Carmencita      0       1894    NULL    1       ["Documentary","Short"]
tt0000002       short   Le clown et ses chiens  Le clown et ses chiens  0       1892    NULL    5       ["Animation","Short"]
tt0000003       short   Pauvre Pierrot  Pauvre Pierrot  0       1892    NULL    4       ["Animation","Comedy","Romance"]
tt0000004       short   Un bon bock     Un bon bock     0       1892    NULL    12      ["Animation","Short"]
tt0000005       short   Blacksmith Scene        Blacksmith Scene        0       1893    NULL    1       ["Comedy","Short"]
tt0000006       short   Chinese Opium Den       Chinese Opium Den       0       1894    NULL    1       ["Short"]
tt0000007       short   Corbett and Courtney Before the Kinetograph     Corbett and Courtney Before the Kinetograph     0       1894    NULL    1       ["Short","Sport"]
tt0000008       short   Edison Kinetoscopic Record of a Sneeze  Edison Kinetoscopic Record of a Sneeze  0       1894    NULL    1       ["Documentary","Short"]
tt0000009       movie   Miss Jerry      Miss Jerry      0       1894    NULL    45      ["Romance"]
tt0000010       short   Leaving the Factory     La sortie de l'usine Lumière à Lyon     0       1895    NULL    1       ["Documentary","Short"]
Time taken: 0.085 seconds, Fetched: 10 row(s)
```

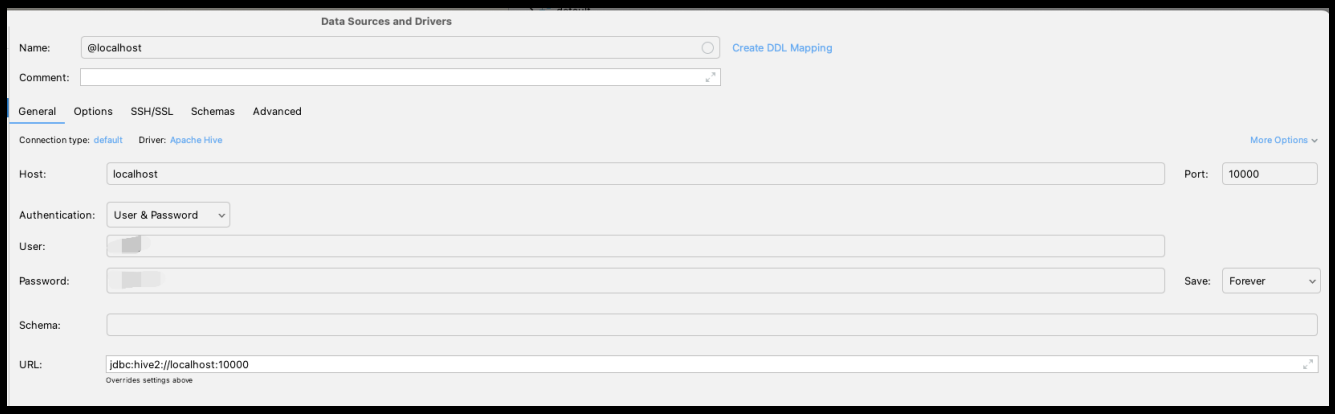
- Solution

  We can use the database tool of IDEA to connect to hive server on local machine, which is very handy and easy to use. There are few points that you should pay attention to:

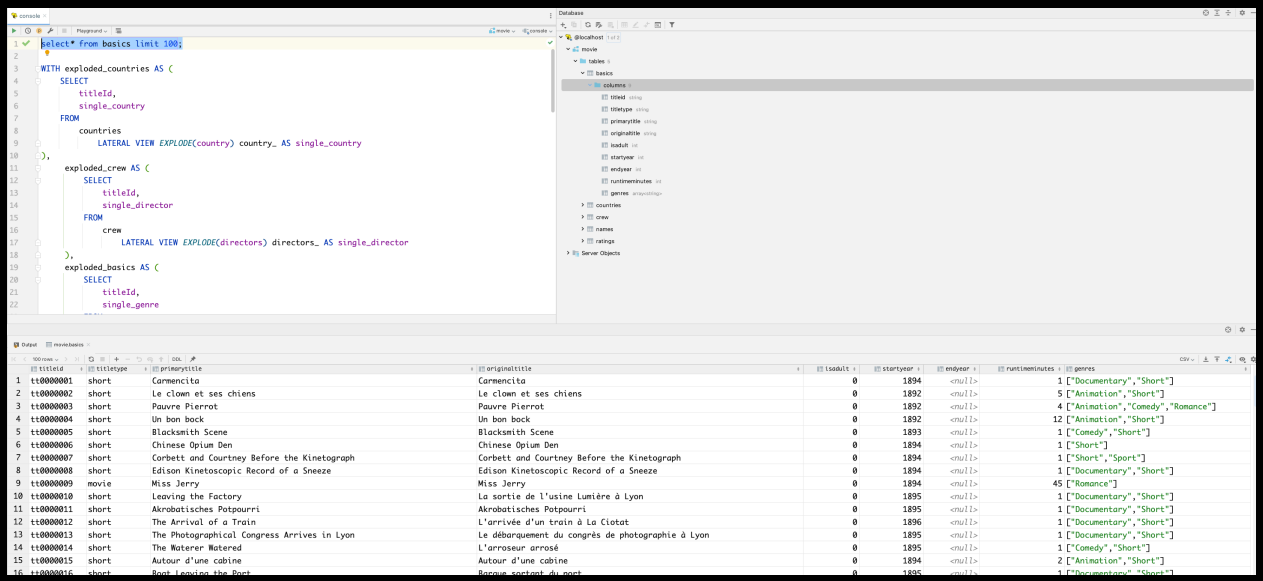
  First, remember to not use the default driver, remove it and import all the jars in Hadoop/share and hive/lib into IDEA and use the jdbc.HiveDriver.



Second, the username and password is the one of your operating system.



Below is an example of the UI of IDEA console if you set up all correctly.



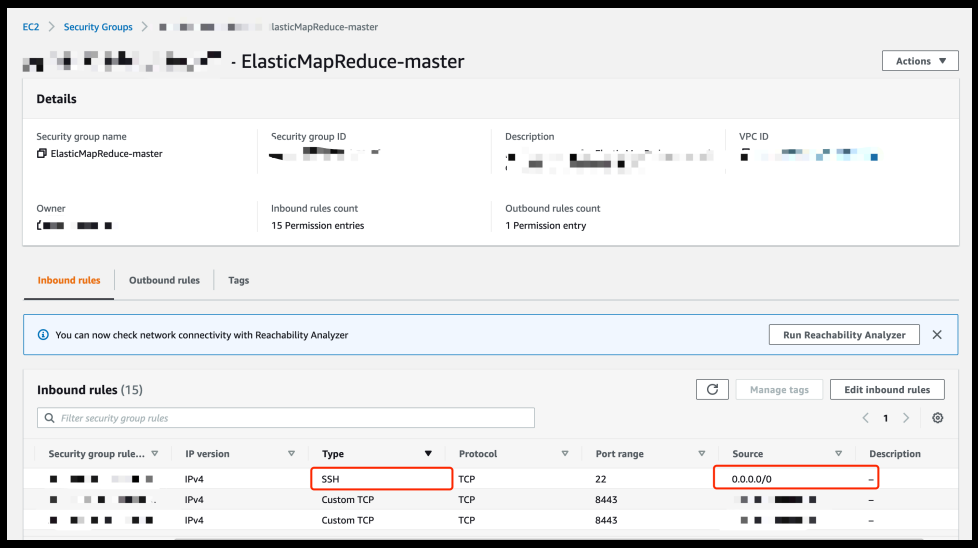
Challenge 4

- Problem: Timeout when connecting to EMR cluster using ssh

When we tried to connect to EMR cluster to verify the output, we met operation timeout exception.

- Solution:

  We added a new rule to the EC2 security groups on AWS console, set the type to SSH and source to 0.0.0.0/0 so that all IP addressed can connect to the cluster with the private key pair used to create the cluster.



Conclusion

Overall, in this project, we analyzed the movie genres and directors across different countries to identify the top-rated genres for each country and the most successful directors in the corresponding region and sector, based on a public dataset from IMDB.

The project involved various tasks, including determining the most popular genres for each country and the most successful director for each genre in each country using Hive. The team overcame various challenges, including installing and running Hive on their local machine, collecting country information using a Java program, and uploading large files to S3.

Our analysis revealed that the documentary genre is the most popular globally. In addition, we identified the top directors for each genre. This information can be valuable for filmmakers, producers, and investors in the movie industry looking to gain a deeper understanding of their respective fields.

Throughout the project, Qiansha and Ni contributed significantly to various tasks and reports. Qiansha took charge of data collection and implemented data analysis tools, while Ni led the report writing and conducted experiments on AWS EMR. We both had hands-on experience with all aspects of the project and gained valuable insights, making it a wonderful learning experience.

There are still some improvement spaces for our program, the Hive queries can sometimes be slow, especially when dealing with large datasets. Therefore, optimizing queries by using appropriate indexing and partitioning techniques can improve performance.