CA675 Cloud Technologies - Assignment 2

IMDb Movies Data Analysis Using Hive, BigQuery, and Tableau Cloud

Group Number – 23

Member Details	Student ID	Email ID
Karthikeyan Pugazhandhi	22267182	karthikeyan.pugazhandhi2@mail.dcu.ie
Prateen Balaji Ravikumar	23268650	prateen.ravikumar2@mail.dcu.ie
Dev Anand Jayakumar	23261806	dev.jayakumar2@mail.dcu.ie
Rahul Shankar	23266483	rahul.shankar3@mail.dcu.ie

Git Repository Link – https://github.com/KarthikeyanPugazhandhi01/CloudAssingnment2.git

Video Walk through Link - https://drive.google.com/file/d/1lz595b12v8OipWABFFWf2bWL2UscB4D6/view?usp=drive_link

Important Links

GCP staging bucket – https://storage.cloud.google.com/mycloudassingnment2dcu1/
Tableau Report Link - https://eu-west-1a.online.tableau.com/t/tableaudcu/views/IMDBRatingAnalysis/Dashboard1

Introduction:

The entertainment industry, characterized by a diverse array of movies, TV shows, and the creative minds behind them, generates a wealth of data that holds the key to understanding audience preferences and industry dynamics. Our analysis embarks on an exploration of the IMDb dataset, a comprehensive repository of information that spans the realm of films, TV series, and the talented individuals who bring them to life.

The IMDb Dataset:

Our journey begins with the IMDb dataset, a treasure trove of insights into the cinematic and television landscape. Comprising six meticulously cleaned tables, each offering a unique perspective on the data, we have laid a robust foundation for our analysis. This dataset includes crucial details about movies, TV shows, and the individual profiles of actors, directors, and other professionals, setting the stage for a deep dive into the world of entertainment.

Our analytical journey is driven by a set of key objectives aimed at uncovering the intricacies of the IMDb dataset:

- Identifying trends in the most popular rated movies over time.
- Recognizing the most prolific and highly-rated individuals in the industry.

Empowering Decision-Making:

- The culmination of Hive's data cleaning, BigQuery's data processing, and Looker Studio's visualization capabilities positions our analysis as a valuable resource for decision-makers, content creators, and industry enthusiasts.

- This entire project is creating in Google Cloud Platform utilizing Technologies such as

Related Works:

"Collaborative Filtering for Movie Recommendations"

This work by Y. Koren et al. delves into collaborative filtering techniques for generating movie recommendations based on user behavior and preferences. By leveraging user-item interaction data, the study explores how collaborative filtering methods, such as matrix factorization, can effectively predict user preferences and improve the accuracy of movie recommendations. The research contributes valuable insights into the application of collaborative filtering algorithms in the context of movie recommendation systems.

"Mining and Summarizing Customer Reviews"

M. Hu et al. focus on sentiment analysis in the realm of movie reviews, aiming to mine and summarize customer sentiments. The study employs natural language processing techniques to extract sentiments from user reviews on platforms like IMDb. By understanding the sentiment patterns in reviews, the research contributes to a deeper comprehension of audience reactions towards movies and provides a foundation for sentiment-based analytics in the film industry.

“Temporal Patterns in Movie Ticket Sales"

J. Feng et al. explore temporal trends in movie ticket sales, investigating patterns that influence audience behavior over time. The research examines factors such as release timing, seasonality, and holidays to uncover insights into the temporal dynamics of movie preferences. By analyzing the temporal aspects of movie consumption, the study provides valuable considerations for film distributors and marketers in optimizing release strategies.

"The Value of Stars: Social Influence in the Movie Industry"

In this work by M. De Vany and W. D. Walls, the focus is on the influence of actors on the success of movies. The study employs network analysis techniques to understand the social dynamics within the movie industry. By analyzing collaboration networks and the impact of star actors, the research sheds light on the interconnectedness of individuals in the film industry and the role of actors in shaping the success of movies. The findings contribute to a deeper understanding of the social aspects influencing movie outcomes.

Technologies Used:

In this project we have used Google Cloud Platform to perform all the data cleaning, data processing, data aa analysis and Visualization of the data visualization

Technology	**Description**
Python	To pull the data from the IMDb website and store it in the GCP Bucket

Apache Hive	To store all the CSV data file into tables, pre-process the data and clean the dataset to use it to process.
BigQuery	Data Processing is done through the BigQuery to find the final analysis.
Tableau Cloud	To visualize the data

Description of the dataset:

- The IMDb datasets, accessible through https://datasets.imdbws.com/, constitute an extensive collection of structured data offering unparalleled insights into the global film and television industry. Providing a comprehensive view of movies, TV shows.

- This dataset consists of 7 Zip files which has detailed catalog of films and TV shows, providing essential information such as titles, release dates, genres, and production details.

- This is a large dataset of 6.32 GB. Which consists of 7 CSV files. Here is a summary of the information in the dataset:

- The **title_akas** table contains information about alternative titles for movies and TV shows.
- The **title_basics** table contains basic information about movies and TV shows, such as the title, release date, and genre.
- The **title_rankings** table contains information about the rankings of movies and TV shows, such as the IMDb rating and the Academy Award nominations.
- The **title_principals** table contains information about the people who worked on movies and TV shows, such as the actors, directors, and writers.
- The **title_ratings** table contains information about the ratings of movies and TV shows, such as the number of users who have rated the movie or TV show and the average rating.

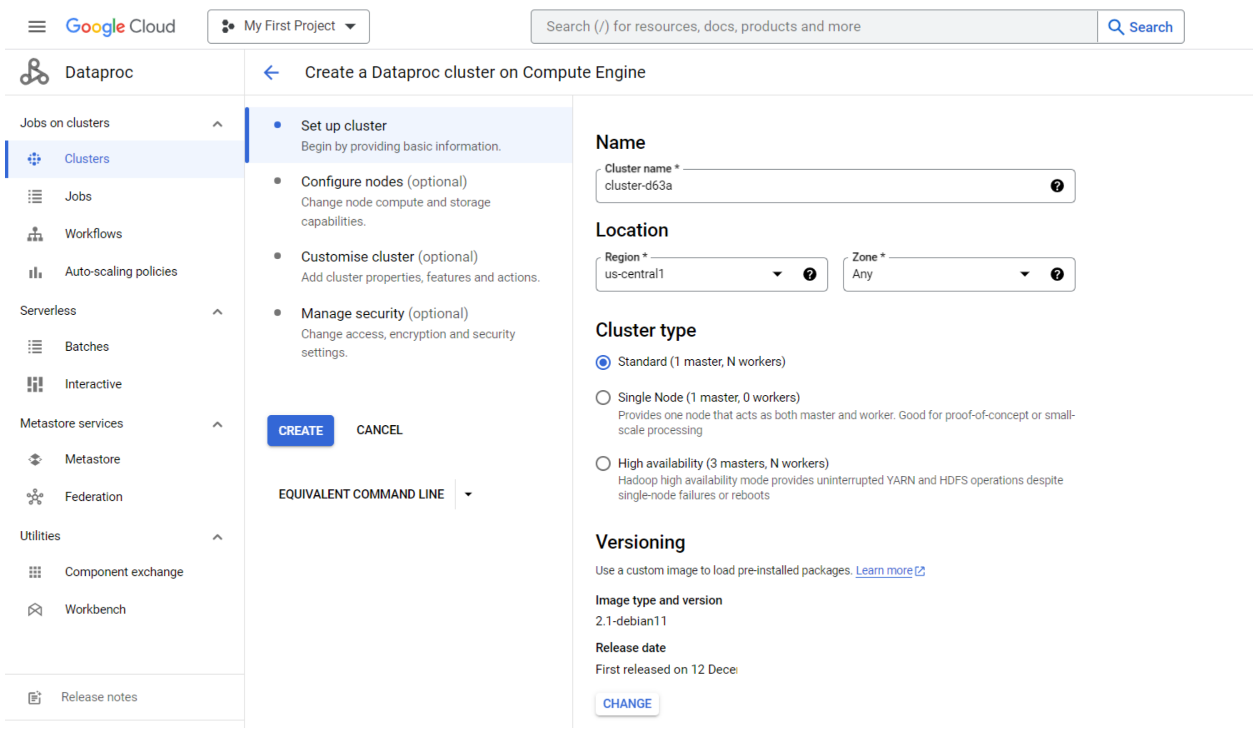
- The **title_cons** table contains information about the constraints on the data in the **title_basics** table.
- The **title_foreign_key** table contains information about the foreign keys in the **title_basics** table.

I. Creating cluster and GCP Bucket:

For this project, we've chosen Google Cloud Platform (GCP) and its Dataproc service to streamline the setup and management of our Hadoop framework. Dataproc handles the complexities of tasks such as provisioning, configuration, tuning, and scaling Hadoop clusters, making it an ideal solution for our needs.

We've created a GCP Dataproc cluster and installed Hadoop, and Hive on it. The cluster is accessible through the Dataproc tab in the GCP console.

1. We are creating cluster on compute engine as using Compute Engine gives you more control over the setup but also requires manual provisioning, configuration, and maintenance.



Create a Dataproc cluster on Compute Engine

- Set up cluster
  Begin by providing basic information.
- Configure nodes (optional)
  Change node compute and storage capabilities.
- Customise cluster (optional)
  Add cluster properties, features and actions.
- Manage security (optional)
  Change access, encryption and security settings.

Name

Cluster name * cluster-d63a

Location

Region * us-central1

Zone * Any

Cluster type

[✓ Standard (1 master, N workers)]

Single Node (1 master, 0 workers)
Provides one node that acts as both master and worker. Good for proof-of-concept or small-scale processing

High availability (3 masters, N workers)
Hadoop high availability mode provides uninterrupted YARN and HDFS operations despite single-node failures or reboots

Versioning

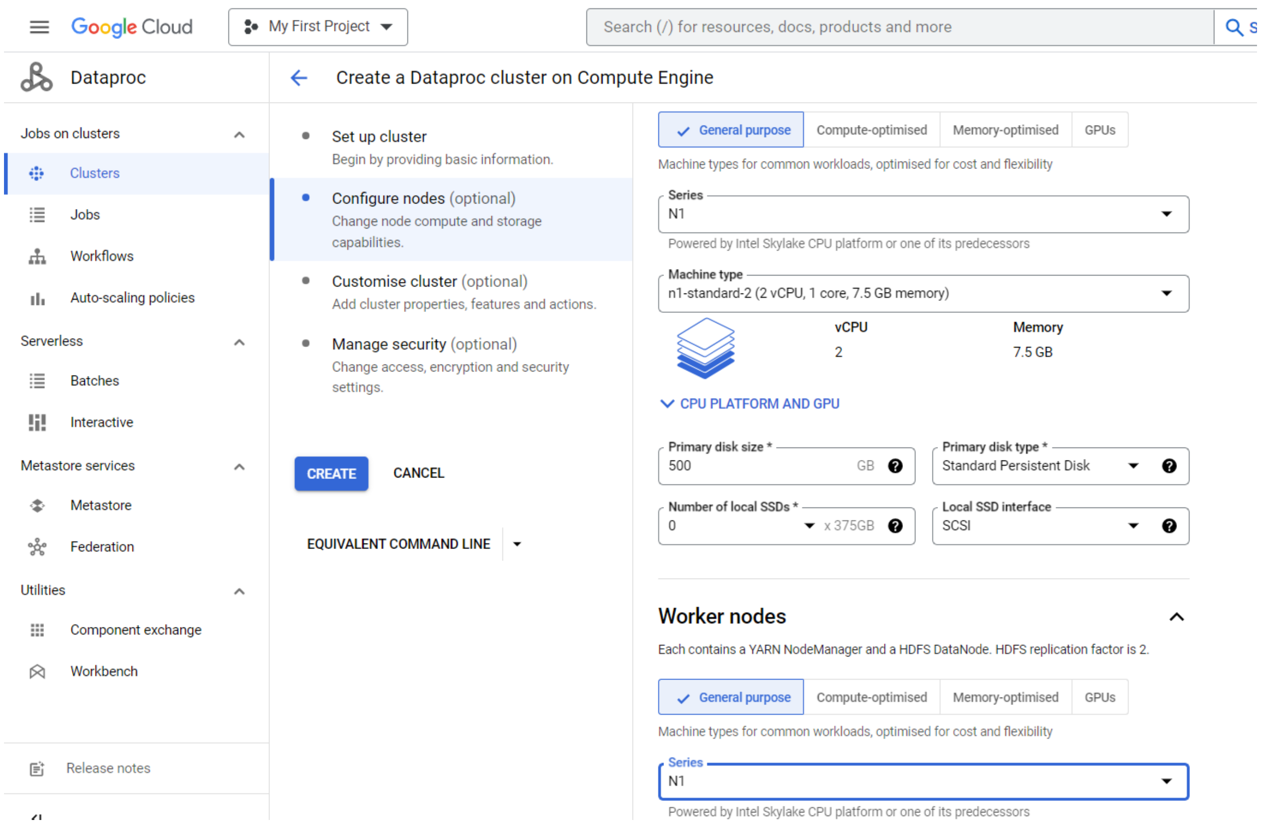
Use a custom image to load pre-installed packages. Learn more

Image type and version
2.1-debian11

Release date
First released on 12 Dece

2. Choose N1 Series in configuring node as the N1 series is well-suited for general-purpose workloads that don't require specialized hardware configurations. It offers a good balance of compute resources, making it suitable for a wide range of applications and use cases.

3. You can see that the configured cluster is provisioned when its status is in running status.


Create a Dataproc cluster on Compute Engine

- Set up cluster — Begin by providing basic information.
- Configure nodes (optional) — Change node compute and storage capabilities.
- Customise cluster (optional) — Add cluster properties, features and actions.
- Manage security (optional) — Change access, encryption and security settings.

CREATE CANCEL

EQUIVALENT COMMAND LINE

General purpose | Compute-optimised | Memory-optimised | GPUs

Machine types for common workloads, optimised for cost and flexibility

Series: N1

Powered by Intel Skylake CPU platform or one of its predecessors

Machine type: n1-standard-2 (2 vCPU, 1 core, 7.5 GB memory)

vCPU	Memory
2	7.5 GB

CPU PLATFORM AND GPU

Primary disk size *: 500 GB

Primary disk type *: Standard Persistent Disk

Number of local SSDs *: 0 x 375GB

Local SSD interface: SCSI

Worker nodes

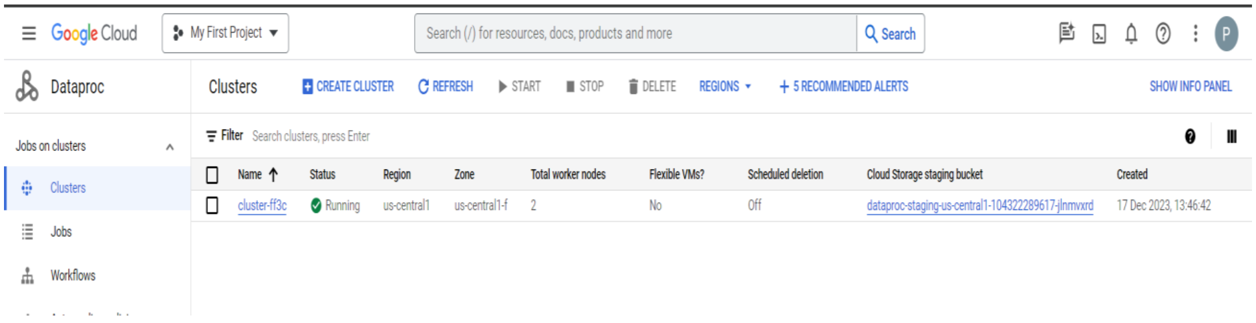
Each contains a YARN NodeManager and a HDFS DataNode. HDFS replication factor is 2.

General purpose | Compute-optimised | Memory-optimised | GPUs

Machine types for common workloads, optimised for cost and flexibility

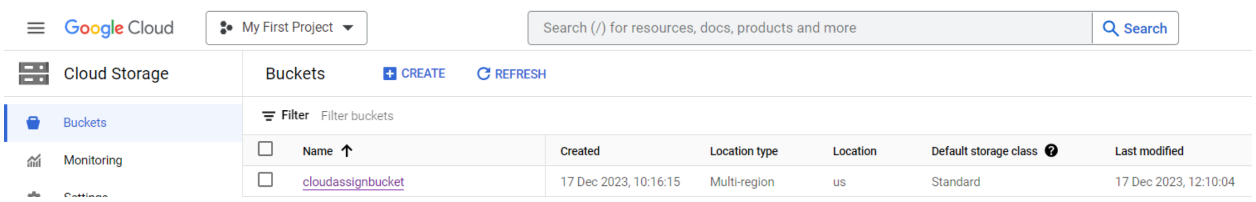
Series: N1

Powered by Intel Skylake CPU platform or one of its predecessors


Clusters

Name	Status	Region	Zone	Total worker nodes	Flexible VMs?	Scheduled deletion	Cloud Storage staging bucket	Created
cluster-ff3c	Running	us-central1	us-central1-f	2	No	Off	dataproc-staging-us-central1-104322289617-jlnmvxrd	17 Dec 2023, 13:46:42

4. Now we have created Data Proc Cluster and the next step is to create a bucket. Under the storage tab choose the bucket option and create a bucket as **cloudassignbucket.**


Buckets

Name	Created	Location type	Location	Default storage class	Last modified
cloudassignbucket	17 Dec 2023, 10:16:15	Multi-region	us	Standard	17 Dec 2023, 12:10:04

5. Now we are loading the dataset which is directly from the IMDb site to out bucket. This saves a lot of time and storage that is being in the local storage. We are using a python script to down the ZIP files from the IMDb site, unzip it and save it in the GCP Bucket. This py file is available in the git repo which has been submitted.

```
# Set your local directory to store downloaded and converted files
local_directory = 'local_files'
os.makedirs(local_directory, exist_ok=True)

# List of GZ files
gz_files = [
    'name.basics.tsv.gz',
    'title.akas.tsv.gz',
    'title.basics.tsv.gz',
    'title.crew.tsv.gz',
    'title.episode.tsv.gz',
    'title.principals.tsv.gz',
    'title.ratings.tsv.gz',
]

for gz_file in gz_files:
    # Set input and output file paths
    gz_file_path = os.path.join(local_directory, gz_file)
    tsv_file_path = os.path.join(local_directory, gz_file[:-3])  # Remove the '.gz' extension
    csv_file_path = os.path.join(local_directory, gz_file[:-6] + '.csv')

 # Remove '.tsv.gz' and add '.csv'
 # Unzip the GZ file
    with gzip.open(gz_file_path, 'rb') as gz_file_content:
        with open(tsv_file_path, 'wb') as tsv_file:
            tsv_file.write(gz_file_content.read())

 # Convert TSV to CSV using pandas
    df = pd.read_csv(tsv_file_path, delimiter='\t', encoding='utf-8')
    df.to_csv(csv_file_path, index=False)

    print(f"Conversion completed for {gz_file}. CSV file saved at {csv_file_path}")

# Clean up: Remove the intermediate TSV files
for gz_file in gz_files:
    tsv_file_path = os.path.join(local_directory, gz_file[:-3])
    os.remove(tsv_file_path)

print("Conversion and cleanup completed.")
```

6. Now all the downloaded files which is in GCP Bucket should be imported to Cloud Shell directory to use the files.

# Copy all the files to the SourceCodefolder

```
gsutil cp gs://mycloudassingnment2dcu1/name.basics.tsv
gs://mycloudassingnment2dcu1/SourceCodefolder/
gsutil cp gs://mycloudassingnment2dcu1/title.akas.tsv
gs://mycloudassingnment2dcu1/SourceCodefolder/
gsutil cp gs://mycloudassingnment2dcu1/title.basics.tsv
gs://mycloudassingnment2dcu1/SourceCodefolder/
gsutil cp gs://mycloudassingnment2dcu1/title.episode.tsv
gs://mycloudassingnment2dcu1/SourceCodefolder/
gsutil cp gs://mycloudassingnment2dcu1/title.principals.tsv
gs://mycloudassingnment2dcu1/SourceCodefolder/
gsutil cp gs://mycloudassingnment2dcu1/title.ratings.tsv
gs://mycloudassingnment2dcu1/SourceCodefolder/
gsutil cp gs://mycloudassingnment2dcu1/title.crew.tsv
gs://mycloudassingnment2dcu1/SourceCodefolder/
```

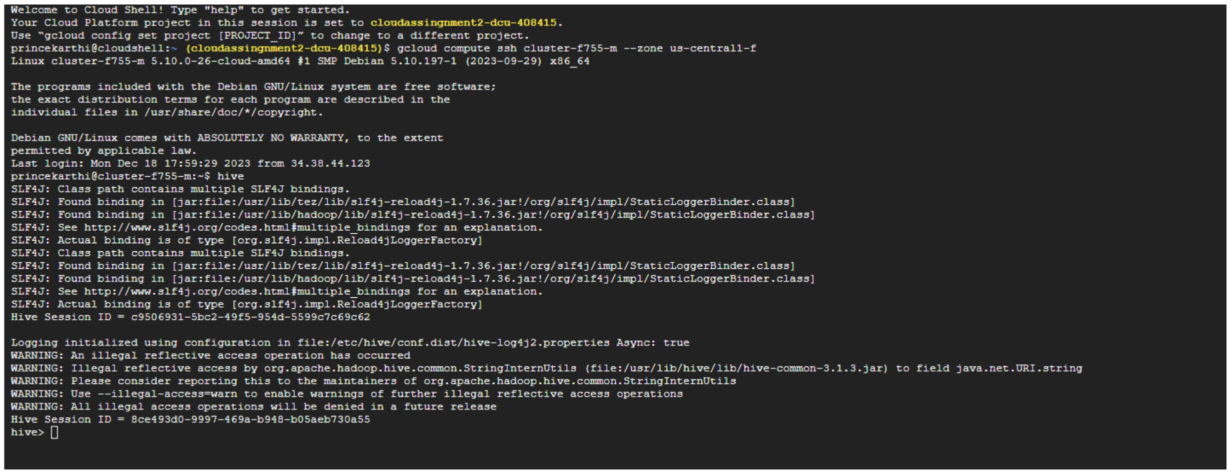
We have uploaded all the files in the Source code folder and the next step is loading all the CSV files into Hive tables.

II. Data Cleaning and Pre-Processing:

1. We are using Apache hive to load the data from the GCP bucket as tables and clean the data. Establish a connection with the Hive. Load the data into Hive Table the below Script. There are 7 CSV files so we are creating 7 different tables and loading the data into the tables.

# Getting hive instance and connecting to hive:

```
-- Comments: List compute instances and add SSH keys metadata
gcloud compute instances list
gcloud compute ssh cluster-f755-m --zone us-central1-f
```



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cloudassingnment2-dcu-408415.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
princekarthi@cloudshell:~ (cloudassingnment2-dcu-408415)$ gcloud compute ssh cluster-f755-m --zone us-central1-f
Linux cluster-f755-m 5.10.0-26-cloud-amd64 #1 SMP Debian 5.10.197-1 (2023-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Dec 18 17:59:29 2023 from 34.38.44.123
princekarthi@cluster-f755-m:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/tez/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Reload4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/tez/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Reload4jLoggerFactory]
Hive Session ID = c9506931-5bc2-49f5-954d-5599c7c69c62

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: true
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.hive.common.StringInternUtils (file:/usr/lib/hive/lib/hive-common-3.1.3.jar) to field java.net.URI.string
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.hive.common.StringInternUtils
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Hive Session ID = 8ce493d0-9997-469a-b948-b05aeb730a55
hive> 
```

2. Creating all the 6 table in hive using hive code as shown below

```sql
-- Comments: Drop tables if they exist
DROP TABLE IF EXISTS name_basics;
DROP TABLE IF EXISTS title_akas;
DROP TABLE IF EXISTS title_basics;
DROP TABLE IF EXISTS title_episode;
DROP TABLE IF EXISTS title_principals;
DROP TABLE IF EXISTS title_ratings;
DROP TABLE IF EXISTS title_crew;

-- Comments: Create tables for IMDb data
CREATE TABLE name_basics (
  nconst STRING,
  primaryName STRING,
  birthYear INT,
  deathYear INT,
  primaryProfession ARRAY<STRING>,
  knownForTitles ARRAY<STRING>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE;

-- (Repeat similar CREATE TABLE statements for other tables)

-- Comments: Load data into tables from Cloud Storage
LOAD DATA INPATH 'gs://mycloudassingnment2dcu1/SourceCodefolder/name.basics.tsv' INTO TABLE name_basics;

-- (Repeat similar LOAD DATA statements for other tables)
```

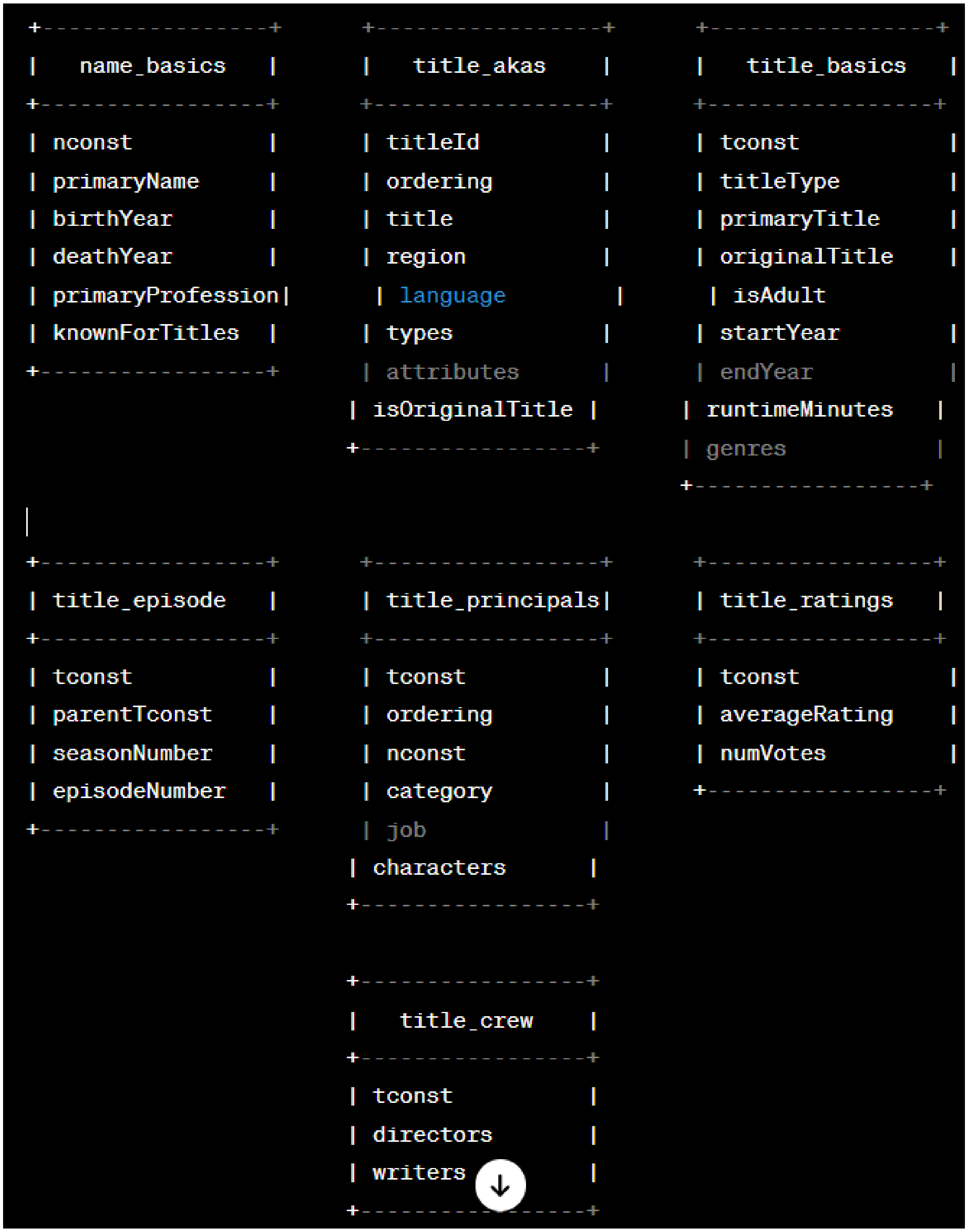
3. This code creates a table such as name_basics. So subsequently create another 6 tables to load the data from the CSV. Now load the data into the table using the below script.

```sql
-- Comments: Load data into tables from Cloud Storage
LOAD DATA INPATH 'gs://mycloudassingnment2dcu1/SourceCodefolder/name.basics.tsv' INTO TABLE name_basics;

-- (Repeat similar LOAD DATA statements for other tables)
```

4. Now all the data has been loaded into the hive tables, so now we are starting to clean the data.

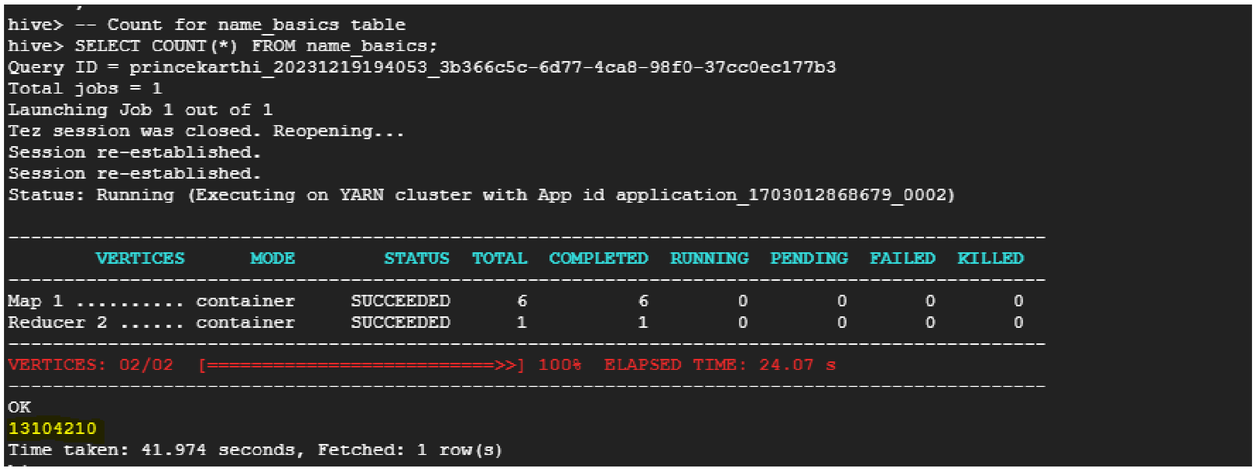
5. Below is the Entity diagram for 7 tables.



```
+-----------------+      +-----------------+      +-----------------+
|   name_basics   |      |   title_akas    |      |   title_basics  |
+-----------------+      +-----------------+      +-----------------+
| nconst          |      | titleId         |      | tconst          |
| primaryName     |      | ordering        |      | titleType       |
| birthYear       |      | title           |      | primaryTitle    |
| deathYear       |      | region          |      | originalTitle   |
| primaryProfession|      | language        |      | isAdult
| knownForTitles  |      | types           |      | startYear       |
+-----------------+      | attributes      |      | endYear         |
                        | isOriginalTitle |      | runtimeMinutes  |
                        +-----------------+      | genres          |
                                                 +-----------------+
|
+-----------------+      +-----------------+      +-----------------+
| title_episode   |      | title_principals|      | title_ratings   |
+-----------------+      +-----------------+      +-----------------+
| tconst          |      | tconst          |      | tconst          |
| parentTconst    |      | ordering        |      | averageRating   |
| seasonNumber    |      | nconst          |      | numVotes        |
| episodeNumber   |      | category        |      +-----------------+
+-----------------+      | job             |
                        | characters      |
                        +-----------------+

                        +-----------------+
                        |   title_crew    |
                        +-----------------+
                        | tconst          |
                        | directors       |
                        | writers         |
                        +-----------------+
```

6. To confirm the data, we look up counts in tables that indicate the size of the data. I'm displaying one table here, called basic, with 13104201 records. For the other six, we have the same or more data.

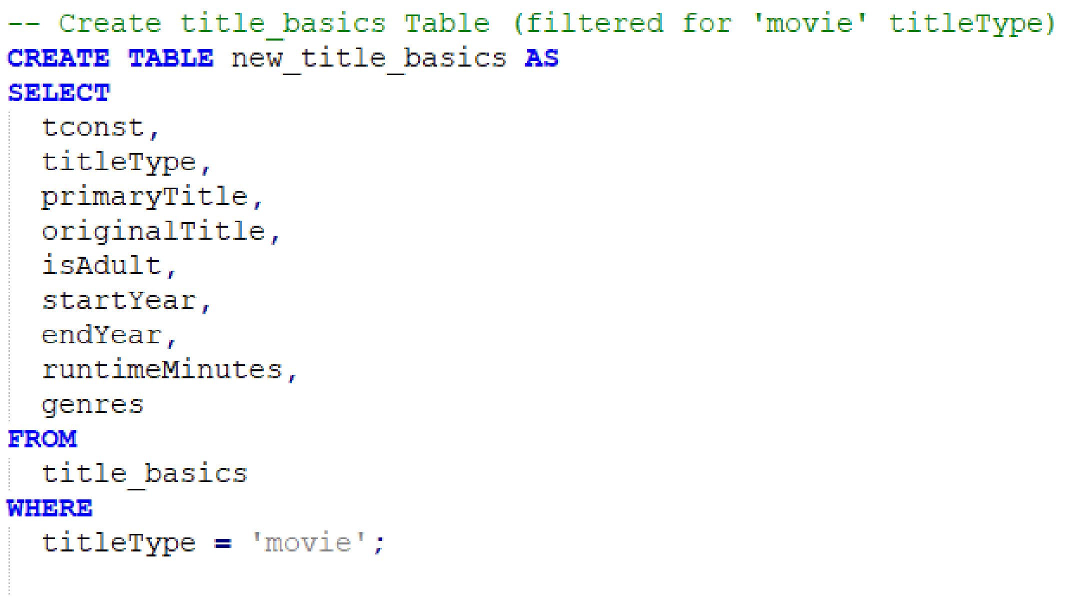


```
hive> -- Count for name_basics table
hive> SELECT COUNT(*) FROM name_basics;
Query ID = princekarthi_20231219194053_3b366c5c-6d77-4ca8-98f0-37cc0ec177b3
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1703012868679_0002)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 24.07 s
----------------------------------------------------------------------------------------------
OK
13104210
Time taken: 41.974 seconds, Fetched: 1 row(s)
```

1. Filtering title_basics for 'movie' titleType:

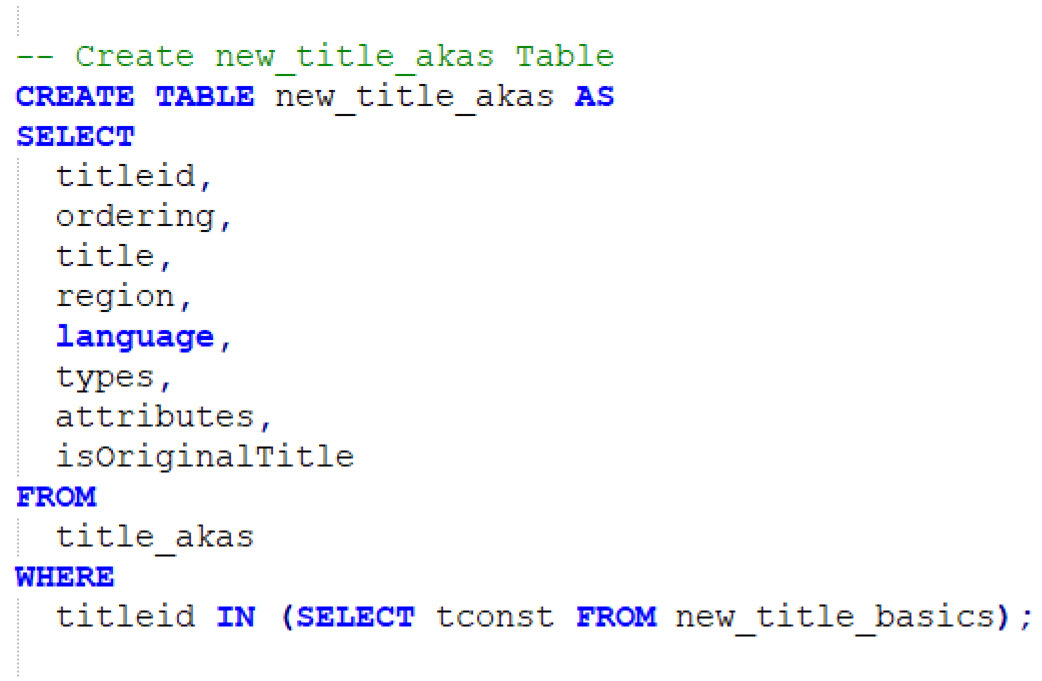
- To reduce the complexity and performance we are picking only movie data type and avoiding others like shorts, cartoon, shows, series etc. To perform this a new table new_title_basics is created by selecting only the rows from title_basics where the titleType is 'movie'.



```
-- Create title_basics Table (filtered for 'movie' titleType)
CREATE TABLE new_title_basics AS
SELECT
  tconst,
  titleType,
  primaryTitle,
  originalTitle,
  isAdult,
  startYear,
  endYear,
  runtimeMinutes,
  genres
FROM
  title_basics
WHERE
  titleType = 'movie';
```

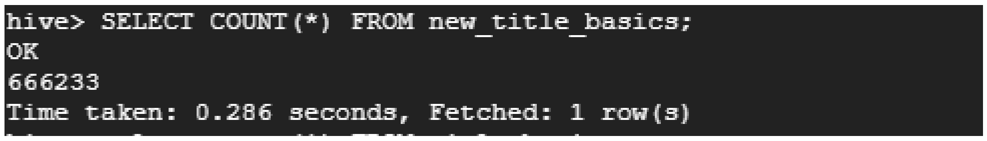
2. **Creating Filtered Tables for Related Data:**

- Several new tables (new_title_akas, new_title_principals, new_title_ratings, new_title_crew, new_name_basics) are created by selecting rows from corresponding original tables based on conditions related to the filtered movie data.



```
-- Create new_title_akas Table
CREATE TABLE new_title_akas AS
SELECT
  titleid,
  ordering,
  title,
  region,
  language,
  types,
  attributes,
  isOriginalTitle
FROM
  title_akas
WHERE
  titleid IN (SELECT tconst FROM new_title_basics);
```

- Here we have mentioned only 1 table but the whole code to create the filtered table is in git repository which has been shared.
- Let verify the count after filtration.

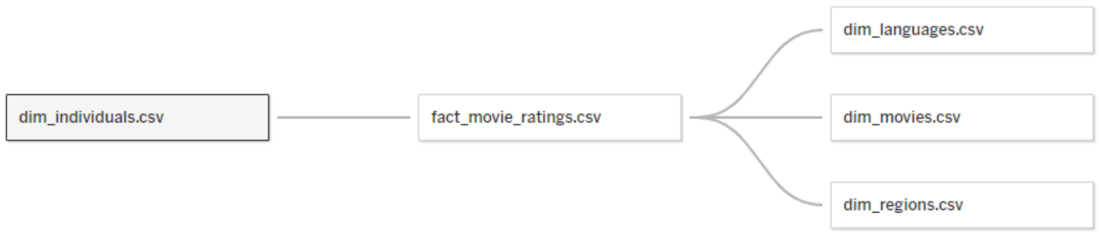


```
hive> SELECT COUNT(*) FROM new_title_basics;
OK
666233
Time taken: 0.286 seconds, Fetched: 1 row(s)
```

- As you see the values are reduced from 10408722 to 666233, that’s around 15 time small than original data

.

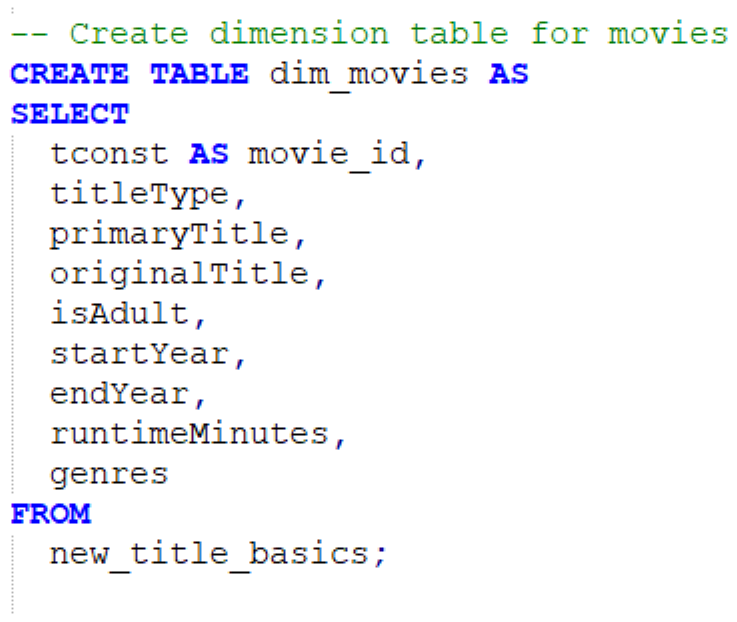
3. **Creating Fact Table and Dimension Tables:**



Now we are creating 1 fact and 4-dimension table, and we are following **star schema**.

- A fact table (fact_movie_ratings) is created by joining relevant data from the new title basics, title principals, and title ratings tables.
- Dimension tables **(dim_movies, dim_individuals, dim_regions, dim_languages)** are created by selecting specific columns from the new filtered tables.

```
-- Create the fact table
CREATE TABLE fact_movie_ratings AS
SELECT
  tb.tconst AS movie_id,
  tp.nconst AS person_id,
  tr.averageRating,
  tr.numVotes
FROM
  new_title_basics tb
  JOIN new_title_principals tp ON tb.tconst = tp.tconst
  JOIN new_title_ratings tr ON tb.tconst = tr.tconst;
```

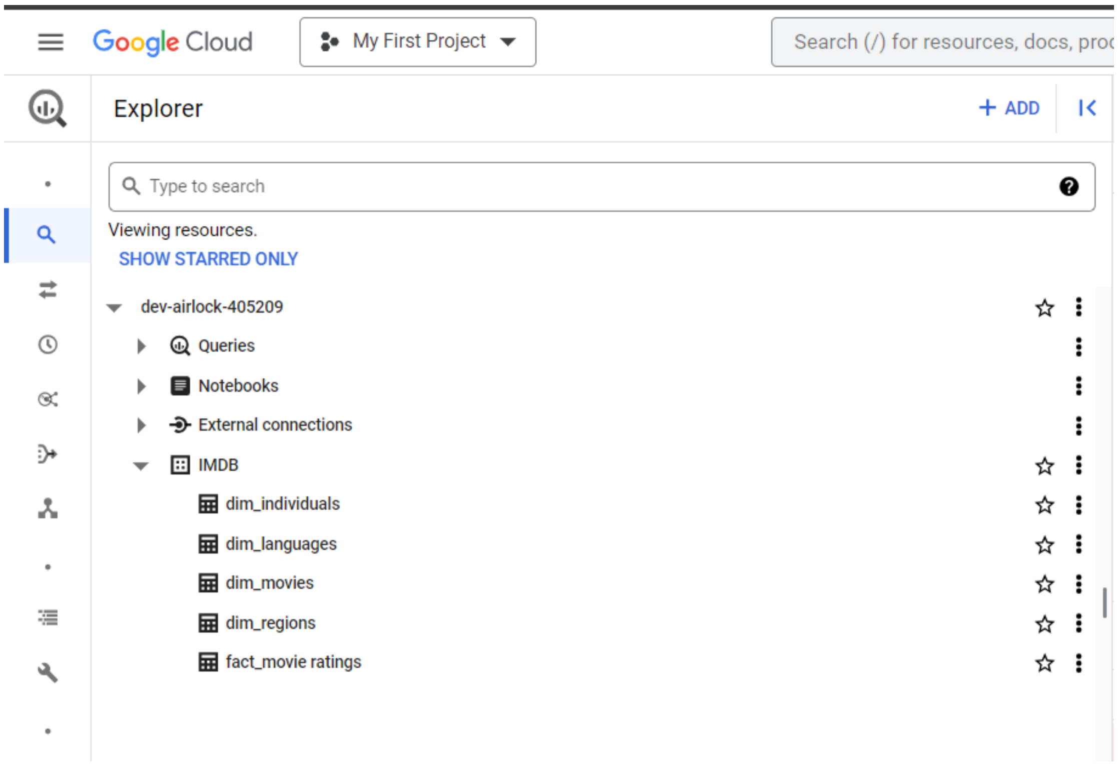


```sql
-- Create dimension table for movies
CREATE TABLE dim_movies AS
SELECT
  tconst AS movie_id,
  titleType,
  primaryTitle,
  originalTitle,
  isAdult,
  startYear,
  endYear,
  runtimeMinutes,
  genres
FROM
  new_title_basics;
```

- The creation of fact and dimension tables facilitates a structured and organized approach to data analysis, enhancing the efficiency and effectiveness of querying and reporting tasks.

7. Data Processing:

- Our data processing pipeline involves the creation of fact and dimension tables, filtering and structuring data to suit analytical needs. The structured data is then queried using BigQuery's SQL capabilities to extract valuable information. For instance, we identify the top 10 actors based on the number of votes they've received and unveil the top 10 highest-rated movies, allowing us to discern trends and patterns within the IMDb dataset.
- Now we are importing all the data from the hive table to Big query to process the data as it is compatible with Google Looker Studio for visualization.



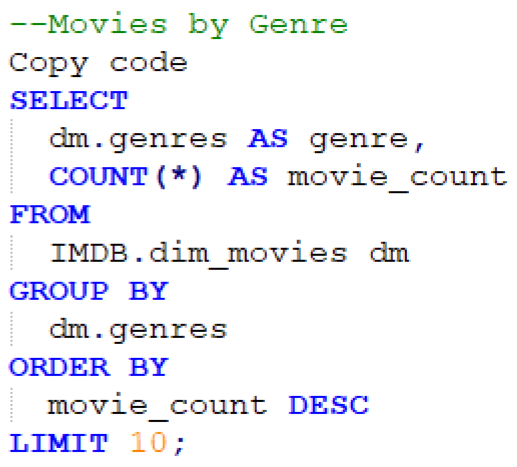
We have created a dataset in the BigQuery and imported all the data from the hive tables to IMDB dataset.

```
--Top 10 Actors by Number of Votes:
SELECT
  IMDB.dim_individuals.person_name AS actor_name,
  SUM(IMDB.fact_movie_ratings.numVotes) AS total_votes
FROM
  IMDB.fact_movie_ratings
  JOIN IMDB.dim_individuals ON IMDB.fact_movie_ratings.person_id = IMDB.dim_individuals.person_id
GROUP BY
  actor_name
ORDER BY
  total_votes DESC
LIMIT 10;
```

This query leverages the fact_movie_ratings and dim_individuals tables to aggregate the votes received by each actor, providing insights into their popularity.

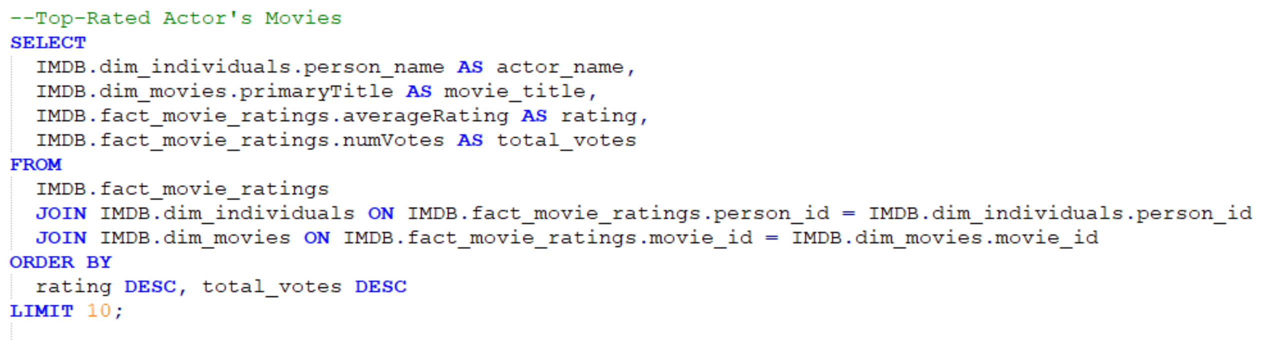
```
--Top 10 Highest-Rated Movies
SELECT
  dm.primaryTitle AS movie_title,
  fr.averageRating AS rating,
  fr.numVotes AS total_votes
FROM
  dev-airlock-405209.IMDB.fact_movie ratings
  JOIN dev-airlock-405209.IMDB.dim_movies dm ON fr.movie_id = dm.movie_id
ORDER BY
  rating DESC, total_votes DESC
LIMIT 10;
```

This query extracts information from the fact_movie_ratings and dim_movies tables to showcase the highest-rated movies, providing valuable insights into audience preferences.



```
--Movies by Genre
Copy code
SELECT
  dm.genres AS genre,
  COUNT(*) AS movie_count
FROM
  IMDB.dim_movies dm
GROUP BY
  dm.genres
ORDER BY
  movie_count DESC
LIMIT 10;
```

This query summarizes the distribution of movies across genres, aiding in the identification of popular genres within the IMDb dataset.

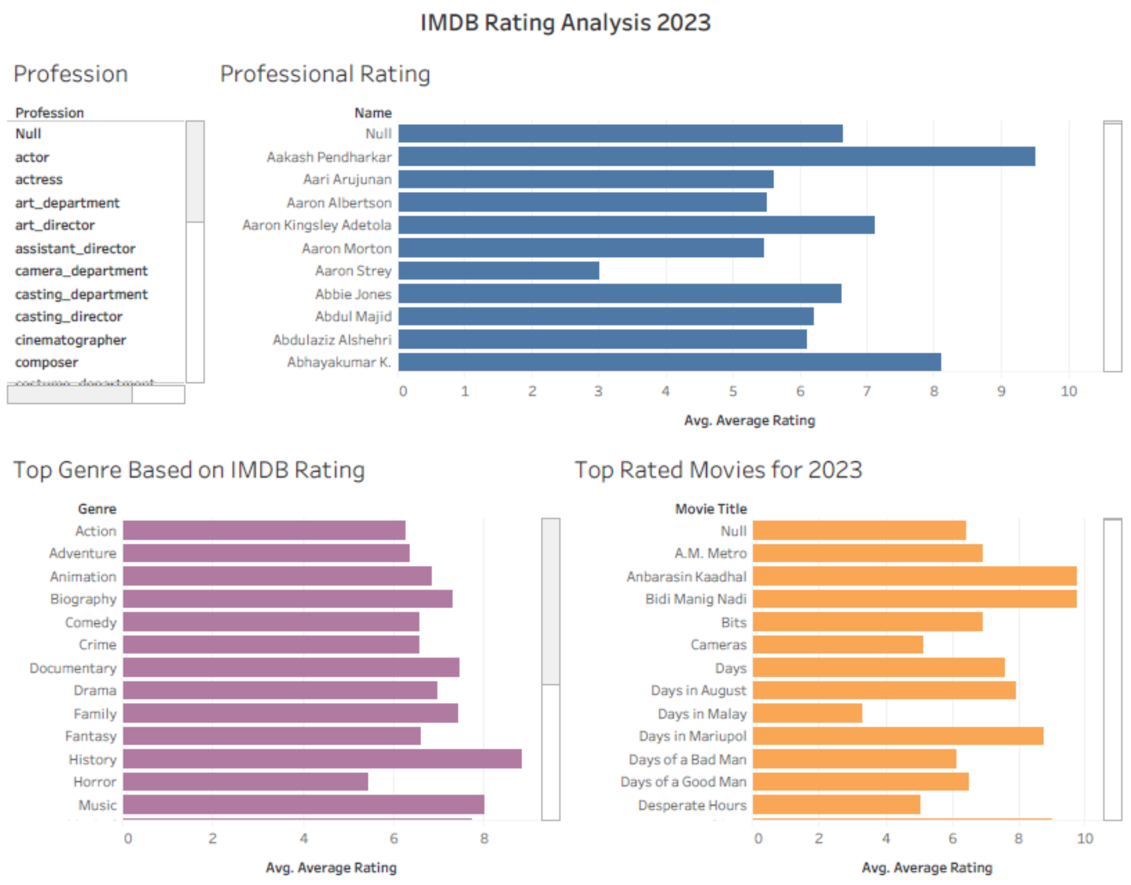


```
--Top-Rated Actor's Movies
SELECT
  IMDB.dim_individuals.person_name AS actor_name,
  IMDB.dim_movies.primaryTitle AS movie_title,
  IMDB.fact_movie_ratings.averageRating AS rating,
  IMDB.fact_movie_ratings.numVotes AS total_votes
FROM
  IMDB.fact_movie_ratings
  JOIN IMDB.dim_individuals ON IMDB.fact_movie_ratings.person_id = IMDB.dim_individuals.person_id
  JOIN IMDB.dim_movies ON IMDB.fact_movie_ratings.movie_id = IMDB.dim_movies.movie_id
ORDER BY
  rating DESC, total_votes DESC
LIMIT 10;
```

This query establishes connections between the fact_movie_ratings, dim_individuals, and dim_movies tables to reveal the top-rated movies associated with top-rated actors.

8. Visualisation :

We are using Google Looker Studio Pro which is a cloud visualization tool. Here we are connecting the BigQuery tables directly into the Looker and visualizing the final analytics.



Here the main filter is year where we are seeing only 2023 movie details. Based on that we can select the profession and their IMDB career rating based on the movies and there top Gener and there best movies.

Challenge:

Resource Management and Scalability in Our Cloud Data Processing Project: The handling of larger datasets in the cloud and the scaling of our project may provide scalability and resource management difficulties. It can be challenging to maximize cloud resource utilization while maintaining cost-effectiveness, particularly when managing fluctuating workloads and dynamic data volumes.

Lesson Learned:

Lesson: In our cloud data processing project, cost monitoring and dynamic resource scaling are essential.

The issue taught us the value of dynamic resource scalability and ongoing cost monitoring while utilizing cloud computing for our project's end-to-end data processing. Scalability capabilities in our cloud platforms, such elastic computing and auto-scaling, let us adjust to changing workloads.

Important Lessons for Our Project:

Auto-scaling methods: In order to automatically modify resources in response to workload needs, we should put auto-scaling methods into place. This guarantees cost reductions during off-peak hours and effective resource usage during peak times.

Tools for Cost Monitoring: To ensure ongoing cost monitoring, we need to make use of third-party solutions and cloud platform tools. By putting up notifications, we may be informed of unforeseen cost increases and respond promptly to maximize resource utilization.

Reserved Instances and Spot Instances: To save money and preserve resource provisioning flexibility, we should investigate options like reserved instances and spot instances.

Fine-Tune Configuration: It's critical to routinely examine and improve setups for data processing jobs. Making the most use of our cloud resources is ensured by optimizing workflows and algorithms.

Individual Contribution:

We divided our task into 4 parts,

Task	User
Gathering Source data and identifying the resources to be used and report preparation.	Rahul Shankar
Data loading and Cleaning	Dev Anand Jayakumar
Designing and creating Data warehouse in Big Query	Prateen Balaji Ravikumar
Dashboard Creating	Karthikeyan Pugazhandhi

1. **Gathering Source Data and Identifying Resources:**

In the initial phase, we selected the source data from the IMDB server, gathering comprehensive datasets relevant to our project. We carefully identified the necessary resources to access and extract data from the IMDB server, ensuring that we have the required permissions and connections established.

2. **Data Loading and Cleaning:**

Following the data gathering phase, we utilized Hive for data cleaning. The Hive platform allowed us to efficiently process and clean the IMDB data, addressing issues such as missing values, outliers, and inconsistencies. By leveraging Hive's capabilities, we ensured that our datasets were prepared for further analysis and reporting.

3. **Designing and Creating Data Warehouse in BigQuery:**

Subsequently, we transitioned to the design and creation of our data warehouse, utilizing BigQuery. This involved defining the schema, tables, and relationships within BigQuery to structure the data effectively. BigQuery's scalability and performance were instrumental in accommodating our data needs, providing a robust foundation for subsequent stages of the project.

4. **Dashboard Creating:**

For the reporting phase, we employed Tableau to create visually compelling dashboards. Using Tableau's intuitive interface, we designed dynamic and informative visualizations that showcase key insights derived from our cleaned and structured data. The Tableau dashboards serve as a powerful tool for stakeholders, enabling them to explore and understand trends, metrics, and patterns.

This updated description reflects the use of IMDB as the data source, Hive for data cleaning, and Tableau for reporting, providing a comprehensive overview of the data-driven workflow from source to reporting in our project.