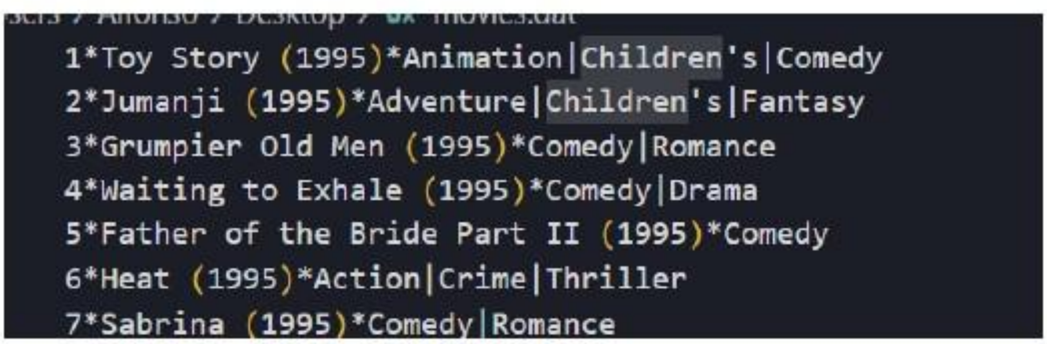
INTRODUCCIÓN

Consultas con HIVE

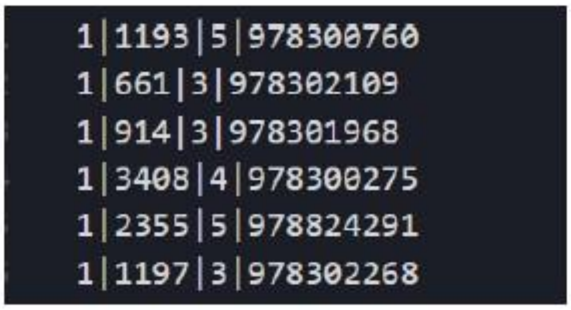
- Trabajaremos con los dos archivos que vienen con la tarea (***movies.dat*** *-películas con sus géneros- y* ***ratings.dat*** *-con un millón de votaciones-*) con la siguiente estructura:

movies.dat - id, película y sus géneros



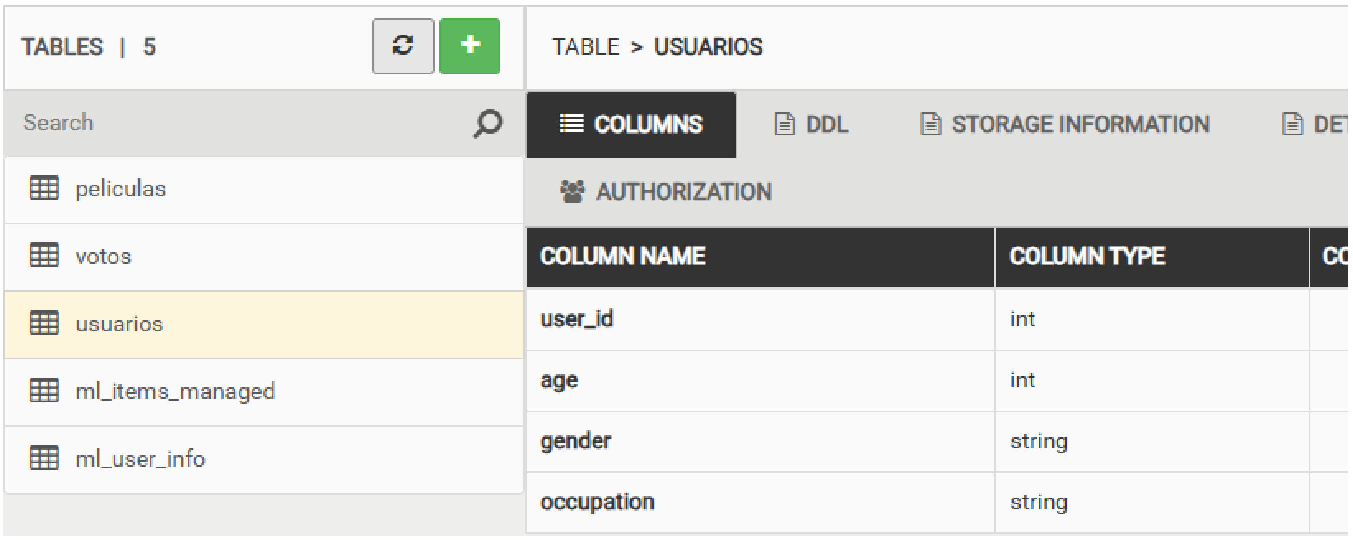
```
1*Toy Story (1995)*Animation|Children's|Comedy
2*Jumanji (1995)*Adventure|Children's|Fantasy
3*Grumpier Old Men (1995)*Comedy|Romance
4*Waiting to Exhale (1995)*Comedy|Drama
5*Father of the Bride Part II (1995)*Comedy
6*Heat (1995)*Action|Crime|Thriller
7*Sabrina (1995)*Comedy|Romance
```

rating.dat - id,usuario,voto,timestamp

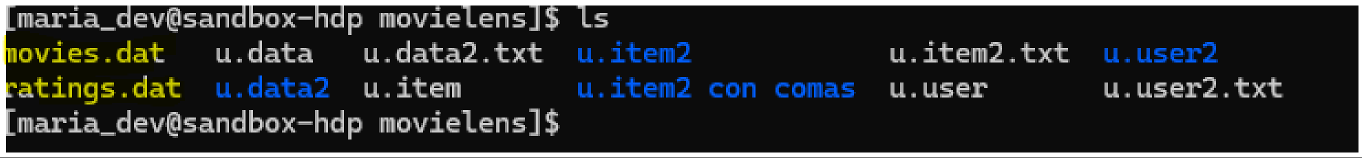


```
1|1193|5|978300760
1|661|3|978302109
1|914|3|978301968
1|3408|4|978300275
1|2355|5|978824291
1|1197|3|978302268
```

- Para cada una de las consultas, mostrar su código y una captura con la salida de las mismas.



COLUMN NAME	COLUMN TYPE	CO
user_id	int	
age	int	
gender	string	
occupation	string	



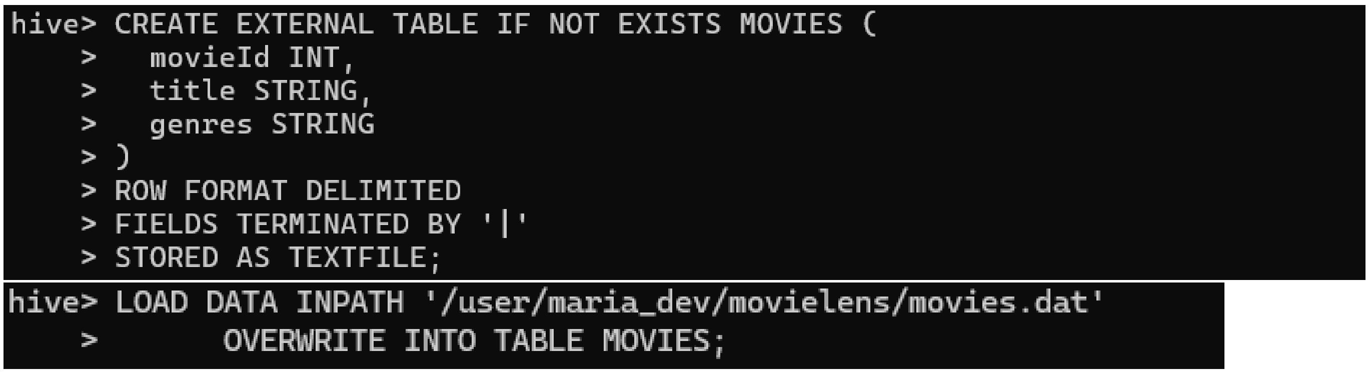
```
[maria_dev@sandbox-hdp movielens]$ ls
movies.dat   u.data   u.data2.txt  u.item2           u.item2.txt  u.user2
ratings.dat  u.data2  u.item       u.item2 con comas u.user       u.user2.txt
[maria_dev@sandbox-hdp movielens]$
```

CONTENIDO

APARTADO A

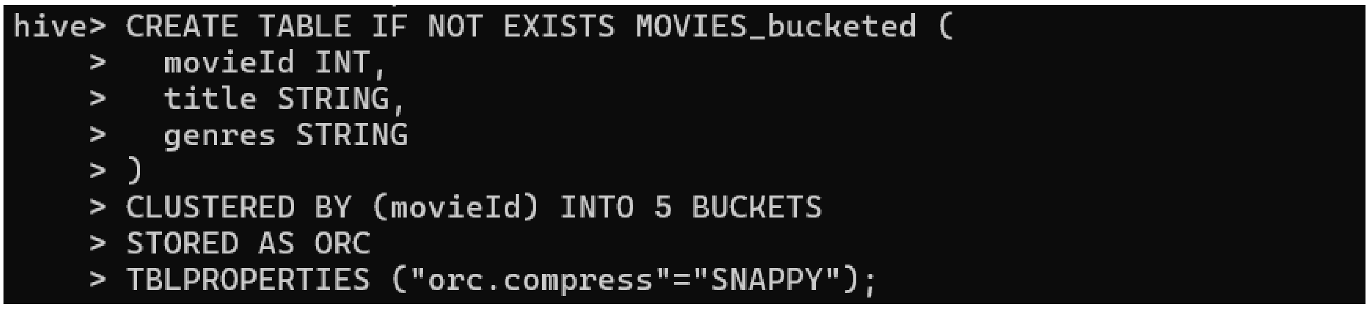
1. Crea las tablas HIVE adecuadas para cargar los datos de cada uno de los archivos. Para mayor eficiencia, crearemos ambas tablas con 5 *buckets* sobre el **id** de la película. Tener especial cuidado en el campo que almacene el género de las películas.

Vamos a crear sobre la base de datos movielens las Tablas MOVIES Y RATINGS

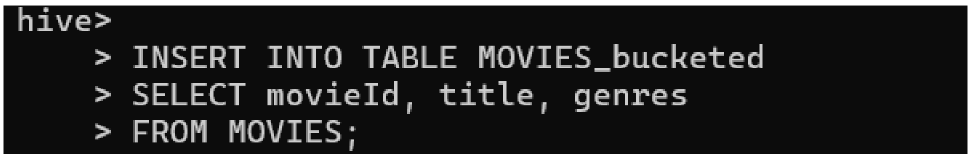


```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS MOVIES (
    >   movieId INT,
    >   title STRING,
    >   genres STRING
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '|'
    > STORED AS TEXTFILE;
hive> LOAD DATA INPATH '/user/maria_dev/movielens/movies.dat'
    >       OVERWRITE INTO TABLE MOVIES;
```

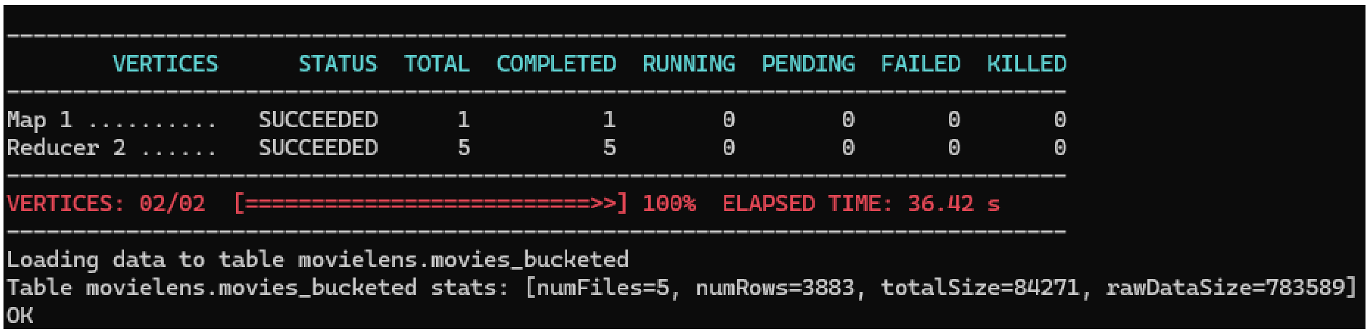
Buket para movies



```
hive> CREATE TABLE IF NOT EXISTS MOVIES_bucketed (
    >   movieId INT,
    >   title STRING,
    >   genres STRING
    > )
    > CLUSTERED BY (movieId) INTO 5 BUCKETS
    > STORED AS ORC
    > TBLPROPERTIES ("orc.compress"="SNAPPY");
```

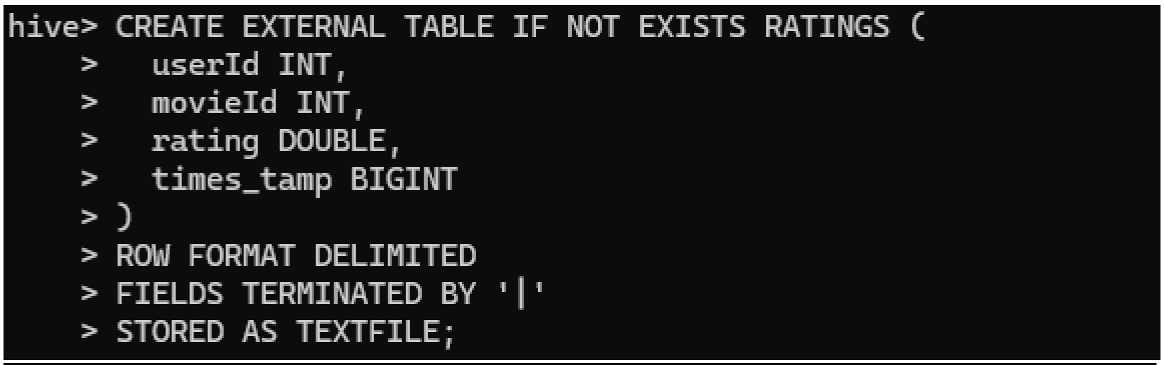


```
hive>
    > INSERT INTO TABLE MOVIES_bucketed
    > SELECT movieId, title, genres
    > FROM MOVIES;
```

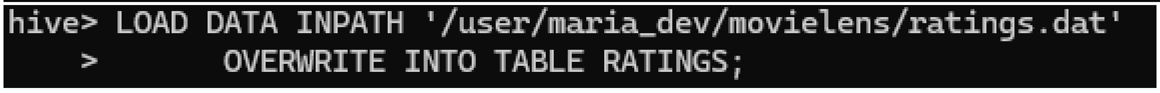


```
--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      5          5        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 36.42 s
--------------------------------------------------------------------------------
Loading data to table movielens.movies_bucketed
Table movielens.movies_bucketed stats: [numFiles=5, numRows=3883, totalSize=84271, rawDataSize=783589]
OK
```

Creacion de tabla ratings

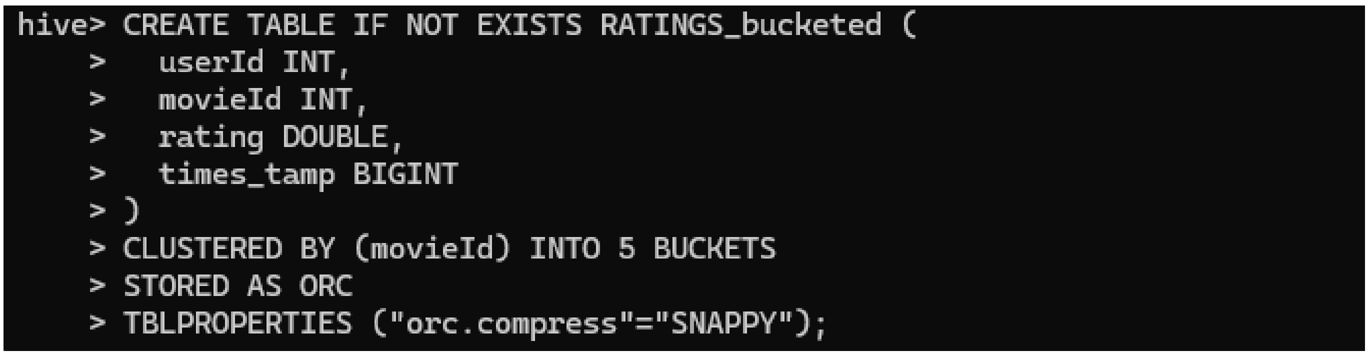


```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS RATINGS (
    >   userId INT,
    >   movieId INT,
    >   rating DOUBLE,
    >   times_tamp BIGINT
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '|'
    > STORED AS TEXTFILE;
```

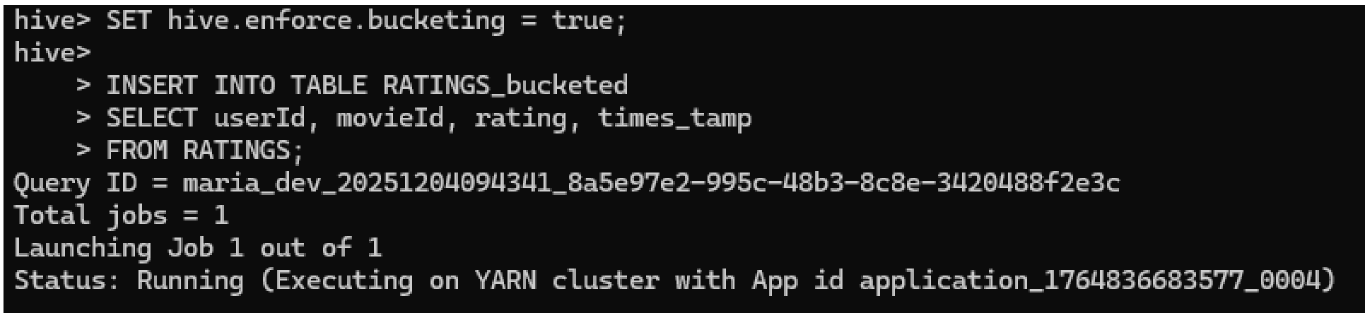


```
hive> LOAD DATA INPATH '/user/maria_dev/movielens/ratings.dat'
    >       OVERWRITE INTO TABLE RATINGS;
```

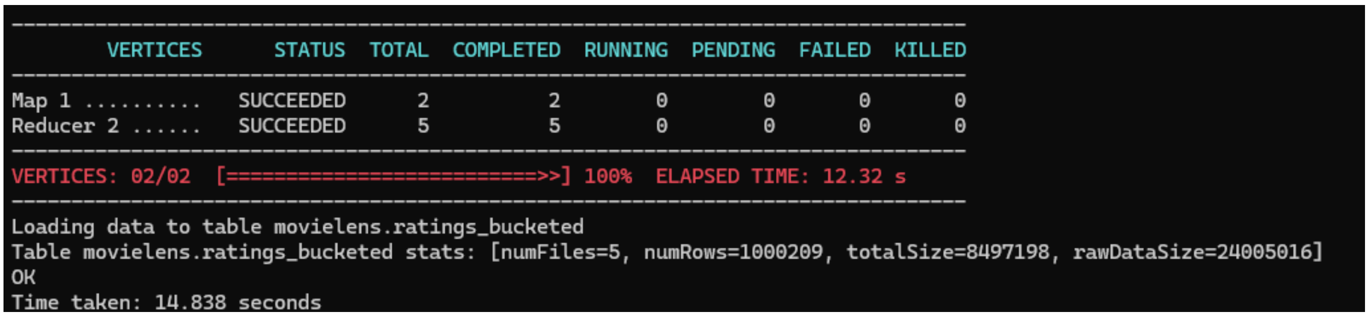
Buket para ratings



```
hive> CREATE TABLE IF NOT EXISTS RATINGS_bucketed (
    >   userId INT,
    >   movieId INT,
    >   rating DOUBLE,
    >   times_tamp BIGINT
    > )
    > CLUSTERED BY (movieId) INTO 5 BUCKETS
    > STORED AS ORC
    > TBLPROPERTIES ("orc.compress"="SNAPPY");
```

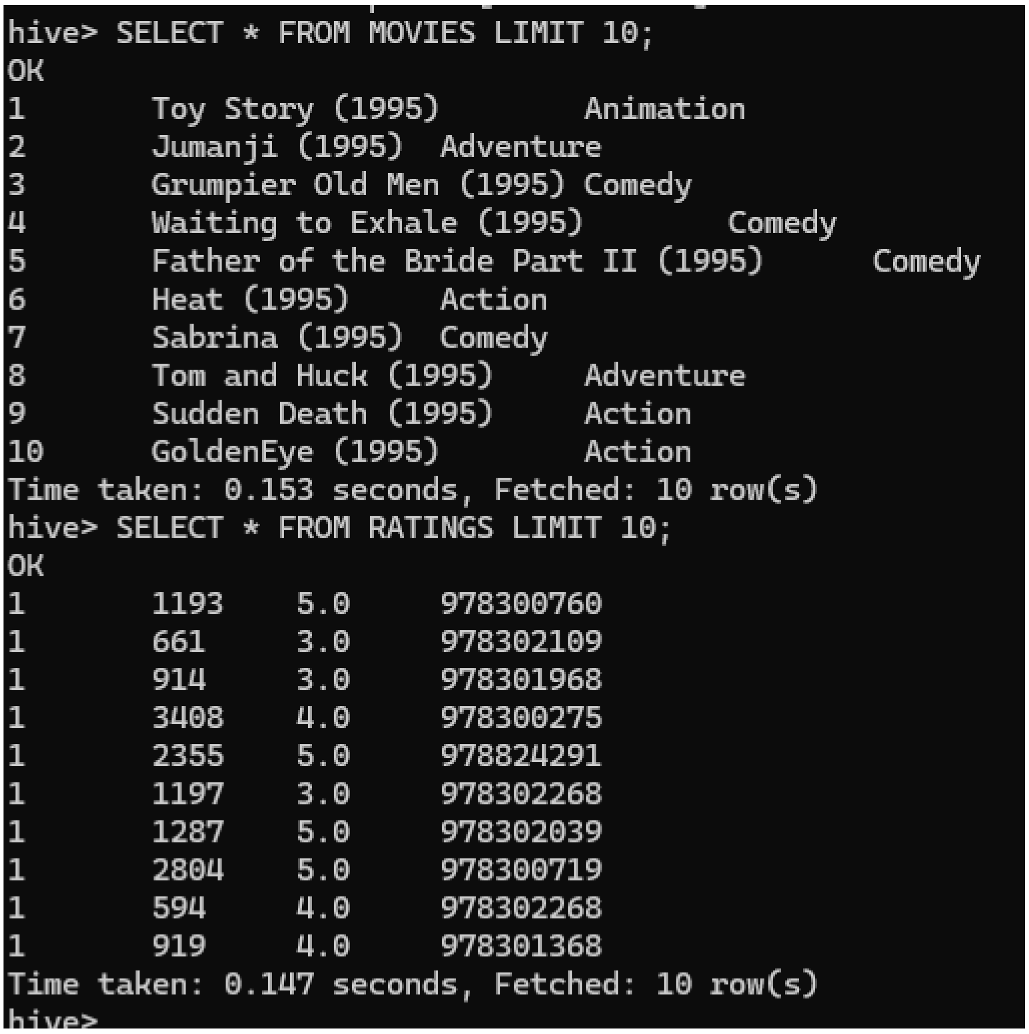


```
hive> SET hive.enforce.bucketing = true;
hive>
    > INSERT INTO TABLE RATINGS_bucketed
    > SELECT userId, movieId, rating, times_tamp
    > FROM RATINGS;
Query ID = maria_dev_20251204094341_8a5e97e2-995c-48b3-8c8e-3420488f2e3c
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1764836683577_0004)
```



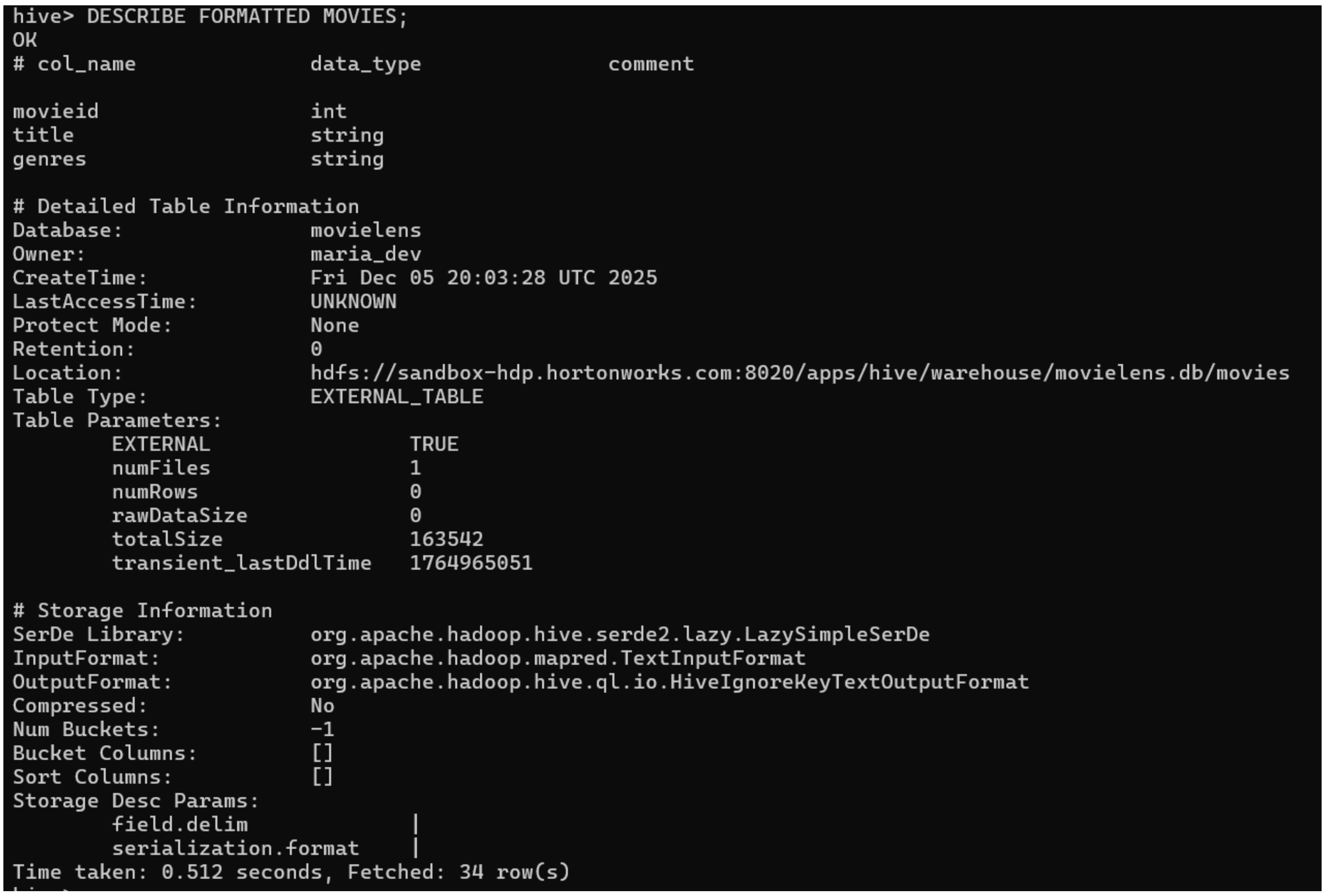
```
----------------------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      2          2        0        0       0       0
Reducer 2 ......   SUCCEEDED      5          5        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 12.32 s
----------------------------------------------------------------------------------------------
Loading data to table movielens.ratings_bucketed
Table movielens.ratings_bucketed stats: [numFiles=5, numRows=1000209, totalSize=8497198, rawDataSize=24005016]
OK
Time taken: 14.838 seconds
```

2. Mostrar las consultas de creación y carga de las tablas con una captura que muestre los datos cargados.

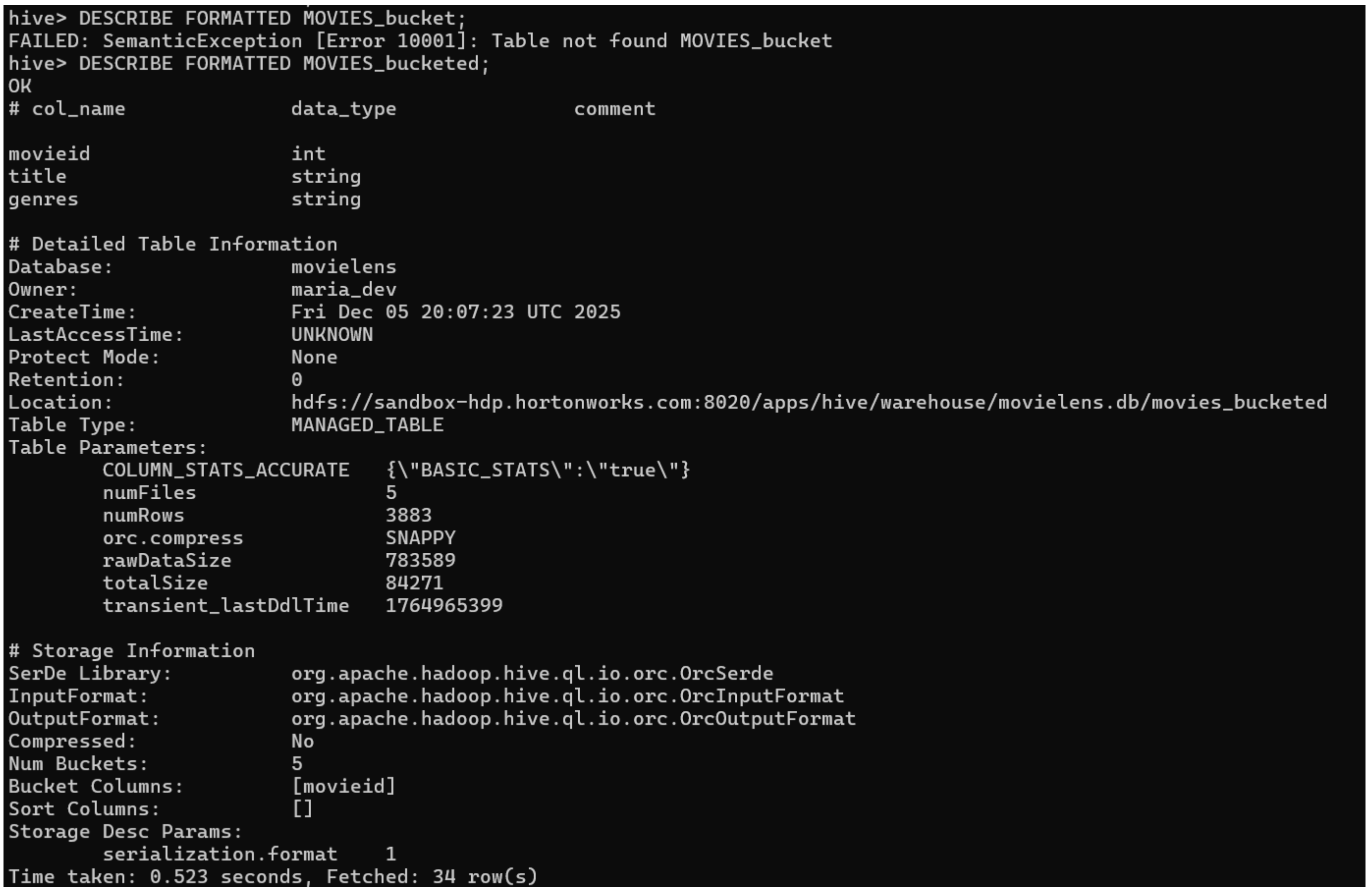


```
hive> SELECT * FROM MOVIES LIMIT 10;
OK
1       Toy Story (1995)        Animation
2       Jumanji (1995)  Adventure
3       Grumpier Old Men (1995) Comedy
4       Waiting to Exhale (1995)        Comedy
5       Father of the Bride Part II (1995)      Comedy
6       Heat (1995)     Action
7       Sabrina (1995)  Comedy
8       Tom and Huck (1995)     Adventure
9       Sudden Death (1995)     Action
10      GoldenEye (1995)        Action
Time taken: 0.153 seconds, Fetched: 10 row(s)
hive> SELECT * FROM RATINGS LIMIT 10;
OK
1       1193    5.0     978300760
1       661     3.0     978302109
1       914     3.0     978301968
1       3408    4.0     978300275
1       2355    5.0     978824291
1       1197    3.0     978302268
1       1287    5.0     978302039
1       2804    5.0     978300719
1       594     4.0     978302268
1       919     4.0     978301368
Time taken: 0.147 seconds, Fetched: 10 row(s)
hive>
```

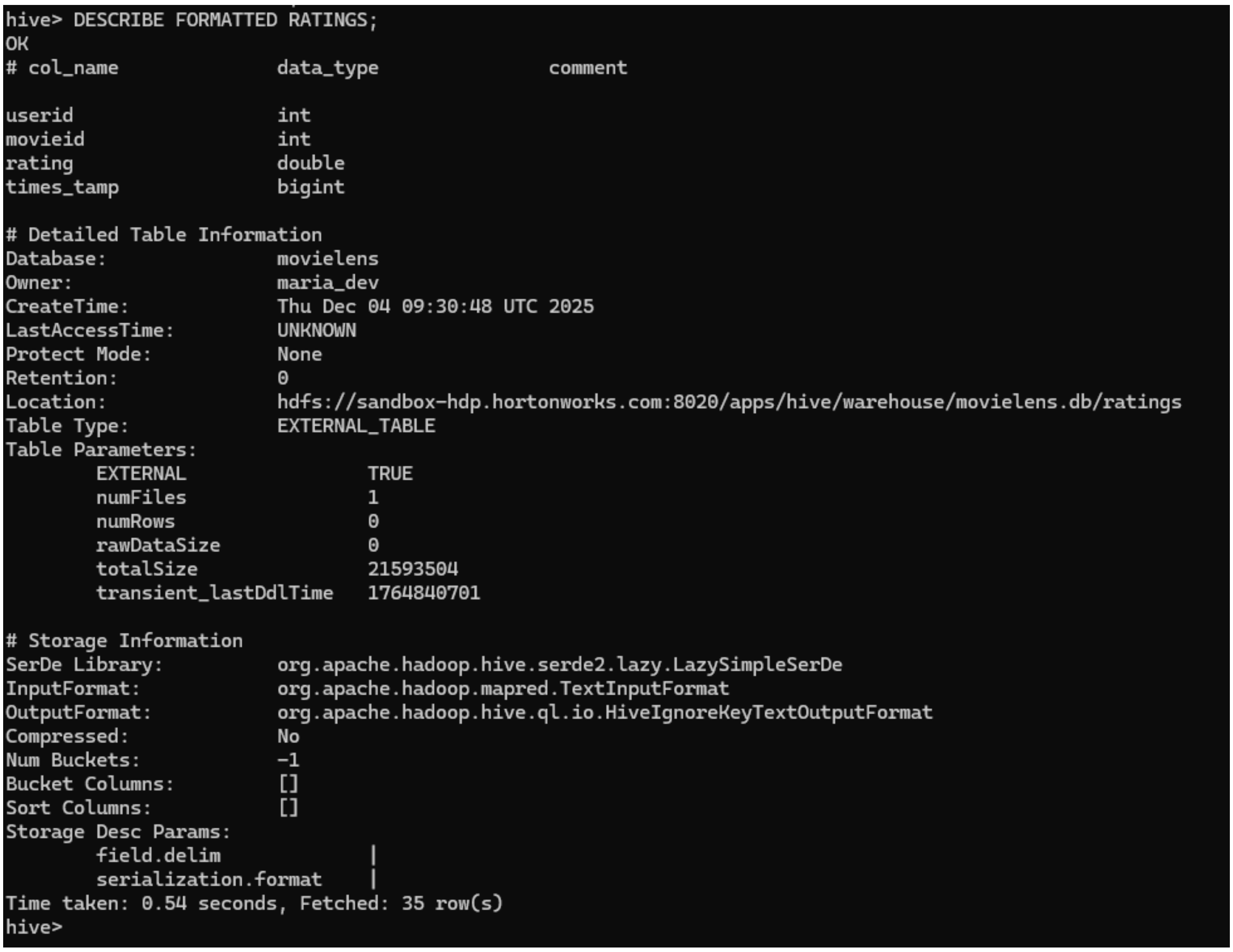
3. Mostrar una captura de las tablas en el *warehouse* de HIVE donde se vean los *buckets.*

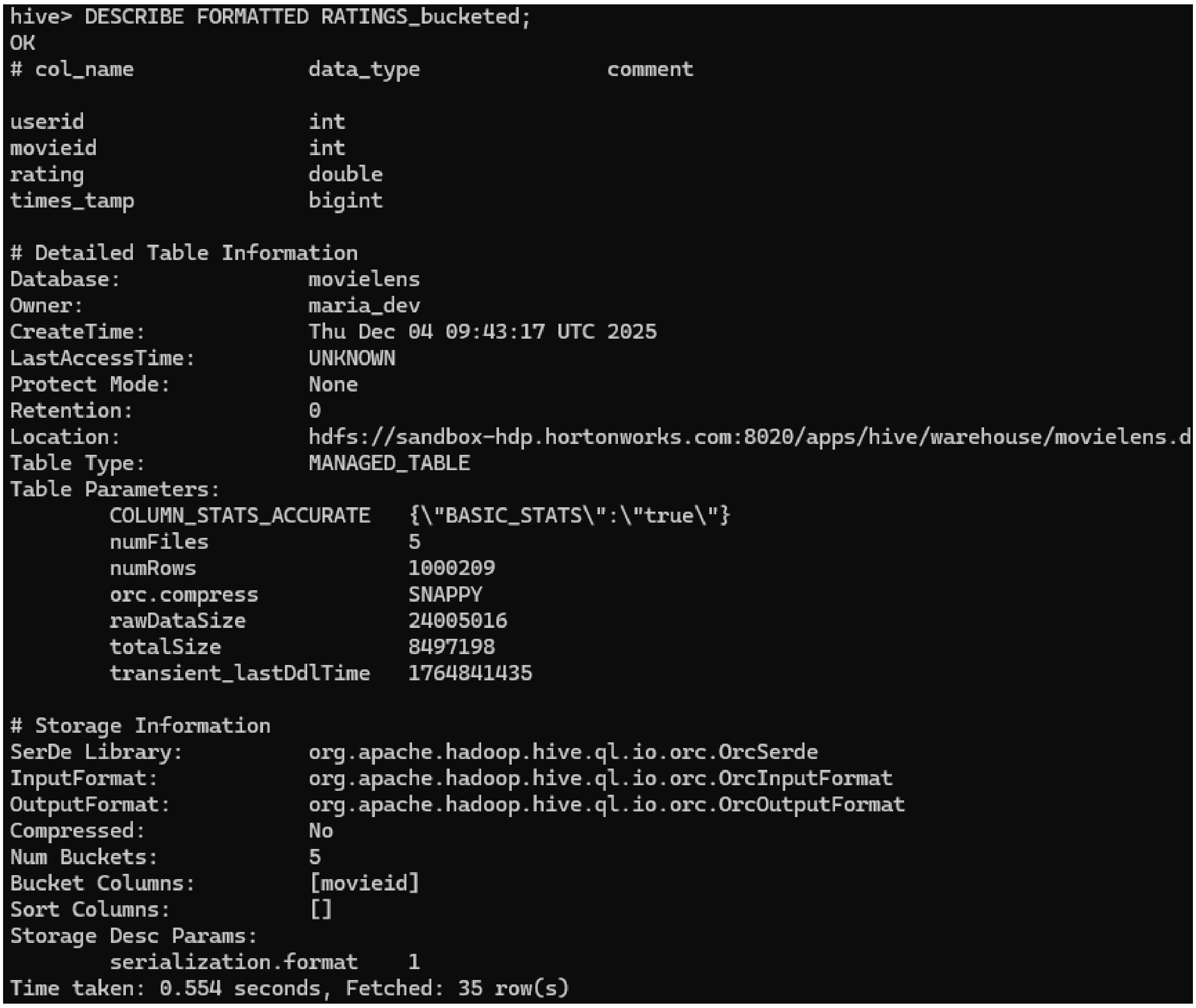


```
hive> DESCRIBE FORMATTED MOVIES;
OK
# col_name              data_type               comment

movieid                 int
title                   string
genres                  string

# Detailed Table Information
Database:               movielens
Owner:                  maria_dev
CreateTime:             Fri Dec 05 20:03:28 UTC 2025
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/movielens.db/movies
Table Type:             EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                TRUE
        numFiles                1
        numRows                 0
        rawDataSize             0
        totalSize               163542
        transient_lastDdlTime   1764965051

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             |
        serialization.format    |
Time taken: 0.512 seconds, Fetched: 34 row(s)
```



```
hive> DESCRIBE FORMATTED MOVIES_bucket;
FAILED: SemanticException [Error 10001]: Table not found MOVIES_bucket
hive> DESCRIBE FORMATTED MOVIES_bucketed;
OK
# col_name              data_type               comment

movieid                 int
title                   string
genres                  string

# Detailed Table Information
Database:               movielens
Owner:                  maria_dev
CreateTime:             Fri Dec 05 20:07:23 UTC 2025
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/movielens.db/movies_bucketed
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        numFiles                5
        numRows                 3883
        orc.compress            SNAPPY
        rawDataSize             783589
        totalSize               84271
        transient_lastDdlTime   1764965399

# Storage Information
SerDe Library:          org.apache.hadoop.hive.ql.io.orc.OrcSerde
InputFormat:            org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
Compressed:             No
Num Buckets:            5
Bucket Columns:         [movieid]
Sort Columns:           []
Storage Desc Params:
        serialization.format    1
Time taken: 0.523 seconds, Fetched: 34 row(s)
```

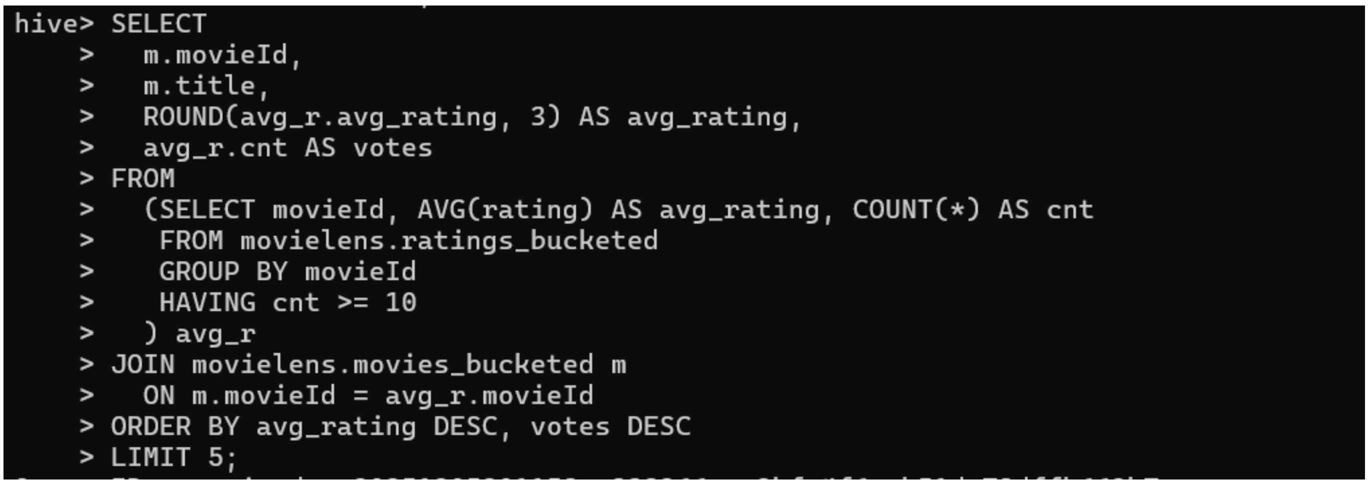

```
hive> DESCRIBE FORMATTED RATINGS;
OK
# col_name              data_type               comment

userid                  int
movieid                 int
rating                  double
times_tamp              bigint

# Detailed Table Information
Database:               movielens
Owner:                  maria_dev
CreateTime:             Thu Dec 04 09:30:48 UTC 2025
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/movielens.db/ratings
Table Type:             EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                TRUE
        numFiles                1
        numRows                 0
        rawDataSize             0
        totalSize               21593504
        transient_lastDdlTime   1764840701

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             |
        serialization.format    |
Time taken: 0.54 seconds, Fetched: 35 row(s)
hive>
```



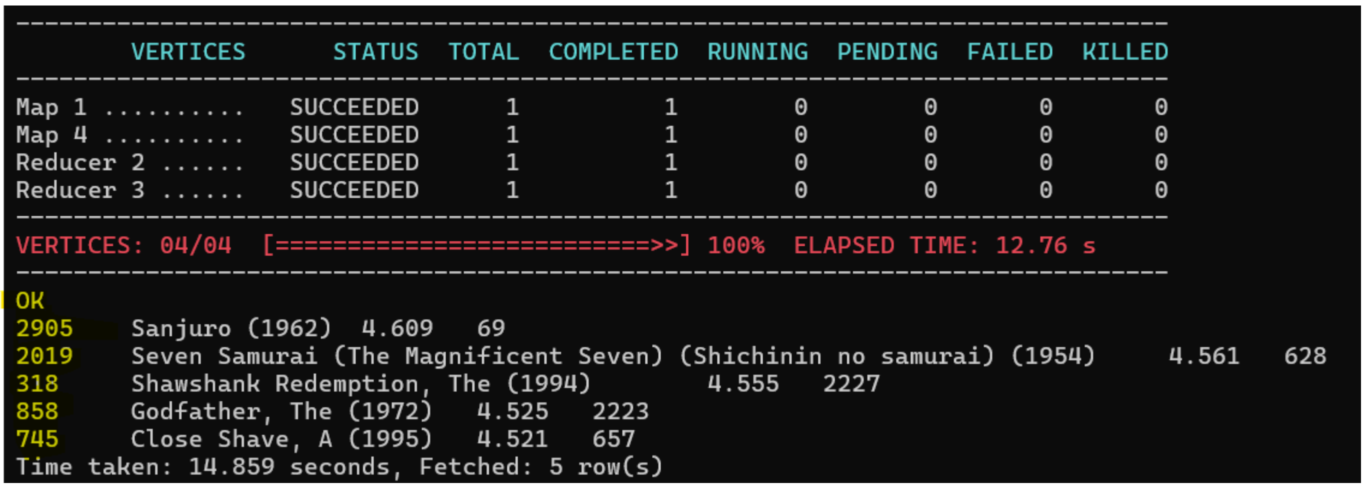
```
hive> DESCRIBE FORMATTED RATINGS_bucketed;
OK
# col_name              data_type               comment

userid                  int
movieid                 int
rating                  double
times_tamp              bigint

# Detailed Table Information
Database:               movielens
Owner:                  maria_dev
CreateTime:             Thu Dec 04 09:43:17 UTC 2025
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/movielens.d
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        numFiles                5
        numRows                 1000209
        orc.compress            SNAPPY
        rawDataSize             24005016
        totalSize               8497198
        transient_lastDdlTime   1764841435

# Storage Information
SerDe Library:          org.apache.hadoop.hive.ql.io.orc.OrcSerde
InputFormat:            org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
Compressed:             No
Num Buckets:            5
Bucket Columns:         [movieid]
Sort Columns:           []
Storage Desc Params:
        serialization.format    1
Time taken: 0.554 seconds, Fetched: 35 row(s)
```

CONTENIDO
APARTADO B

1. Mediante una consulta en HIVE, encontrar las cinco películas (código, título y media de votos) mejor valoradas, que hayan sido votadas al menos por 10 usuarios.



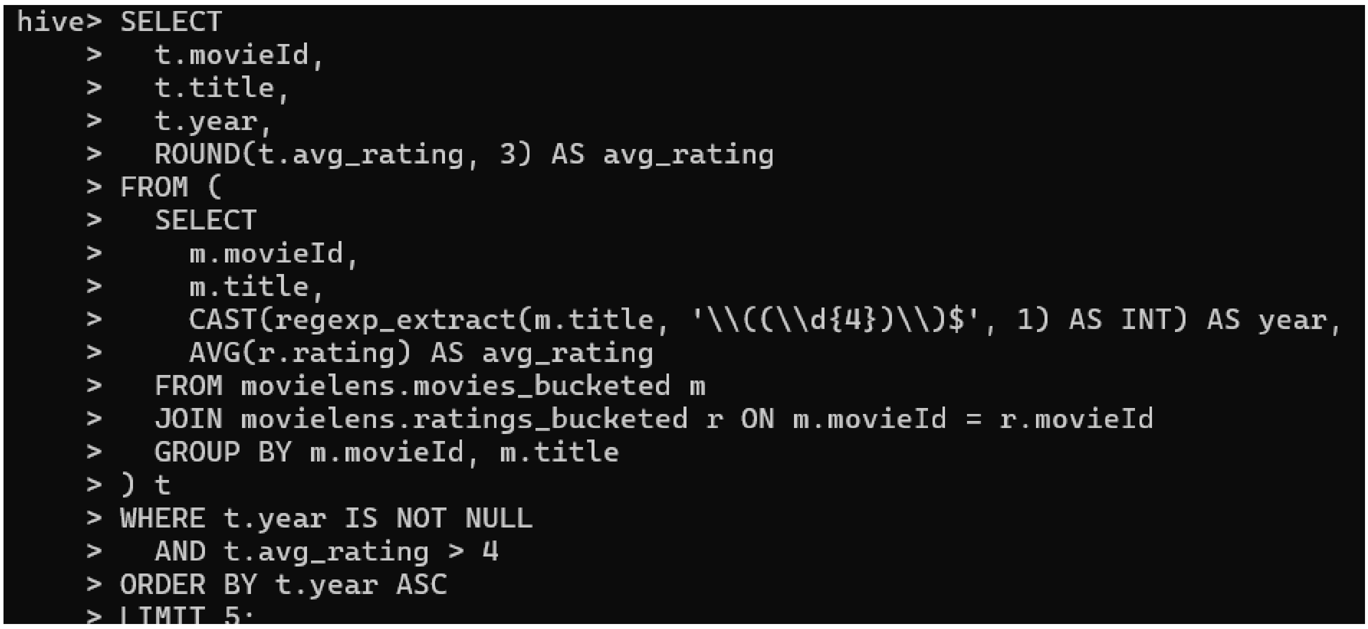
```
hive> SELECT
    >   m.movieId,
    >   m.title,
    >   ROUND(avg_r.avg_rating, 3) AS avg_rating,
    >   avg_r.cnt AS votes
    > FROM
    >   (SELECT movieId, AVG(rating) AS avg_rating, COUNT(*) AS cnt
    >    FROM movielens.ratings_bucketed
    >    GROUP BY movieId
    >    HAVING cnt >= 10
    >   ) avg_r
    > JOIN movielens.movies_bucketed m
    >   ON m.movieId = avg_r.movieId
    > ORDER BY avg_rating DESC, votes DESC
    > LIMIT 5;
```



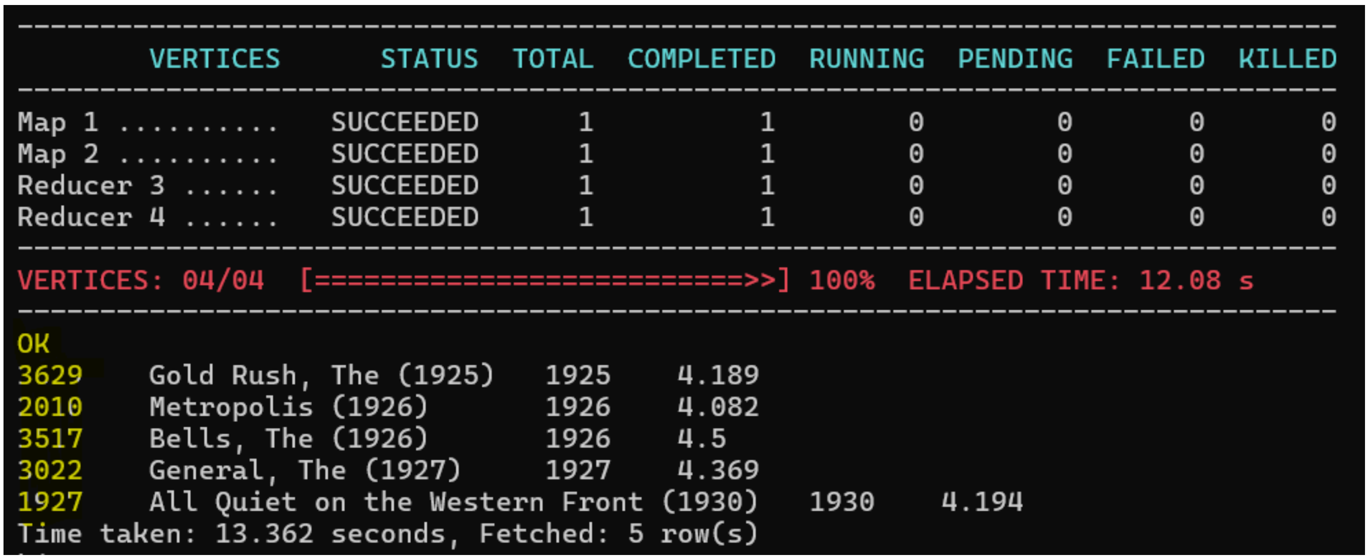
```
----------------------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Map 4 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 04/04  [==========================>>] 100%  ELAPSED TIME: 12.76 s
----------------------------------------------------------------------------------------------
OK
2905    Sanjuro (1962)  4.609   69
2019    Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)     4.561   628
318     Shawshank Redemption, The (1994)        4.555   2227
858     Godfather, The (1972)   4.525   2223
745     Close Shave, A (1995)   4.521   657
Time taken: 14.859 seconds, Fetched: 5 row(s)
```

CONTENIDO
APARTADO C

1. Encontrar las cinco películas más antiguas con una valoración media por encima de 4 puntos.



```
hive> SELECT
    >   t.movieId,
    >   t.title,
    >   t.year,
    >   ROUND(t.avg_rating, 3) AS avg_rating
    > FROM (
    >   SELECT
    >     m.movieId,
    >     m.title,
    >     CAST(regexp_extract(m.title, '\\((\\d{4})\\)$', 1) AS INT) AS year,
    >     AVG(r.rating) AS avg_rating
    >   FROM movielens.movies_bucketed m
    >   JOIN movielens.ratings_bucketed r ON m.movieId = r.movieId
    >   GROUP BY m.movieId, m.title
    > ) t
    > WHERE t.year IS NOT NULL
    >   AND t.avg_rating > 4
    > ORDER BY t.year ASC
    > LIMIT 5;
```



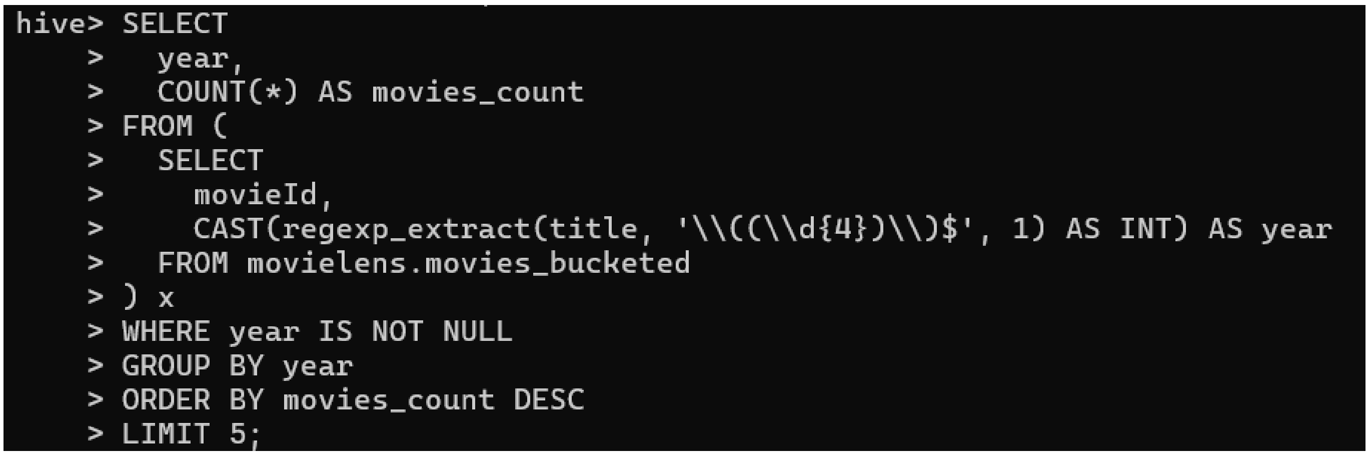
```
----------------------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Map 2 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 4 ......   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 04/04  [==========================>>] 100%  ELAPSED TIME: 12.08 s
----------------------------------------------------------------------------------------------
OK
3629    Gold Rush, The (1925)   1925    4.189
2010    Metropolis (1926)       1926    4.082
3517    Bells, The (1926)       1926    4.5
3022    General, The (1927)     1927    4.369
1927    All Quiet on the Western Front (1930)   1930    4.194
Time taken: 13.362 seconds, Fetched: 5 row(s)
```

CONTENIDO

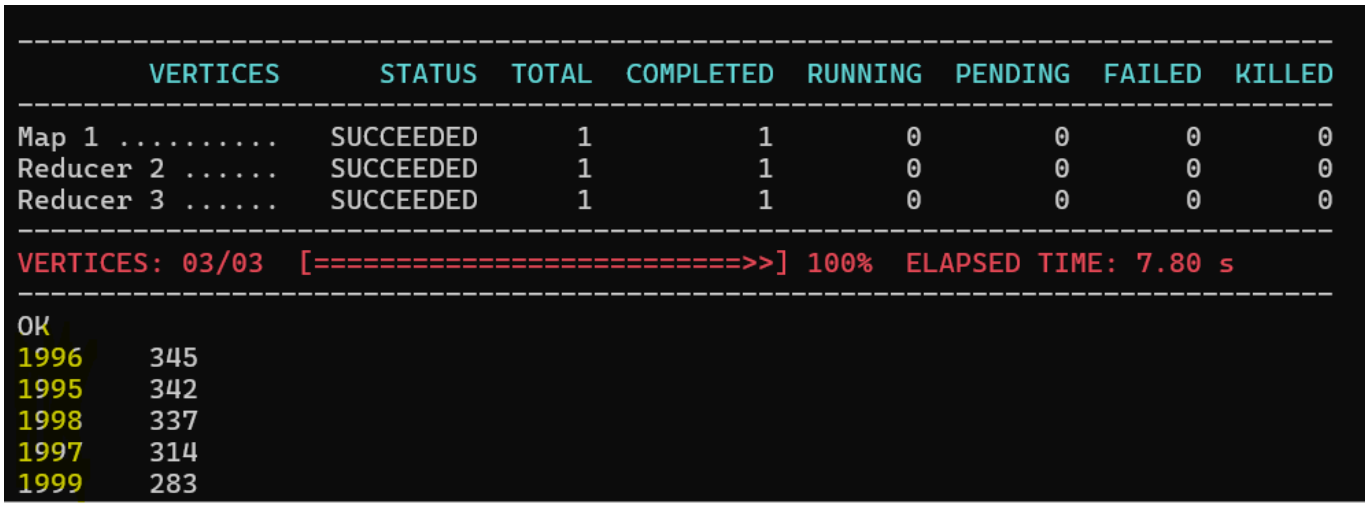
APARTADO D

Investigando las funciones que nos ofrece HIVE para el manejo de cadenas….

1. Muestra los cinco años en que se editaron más películas indicando el número de ellas para cada año.



```
hive> SELECT
    >   year,
    >   COUNT(*) AS movies_count
    > FROM (
    >   SELECT
    >     movieId,
    >     CAST(regexp_extract(title, '\\((\\d{4})\\)$', 1) AS INT) AS year
    >   FROM movielens.movies_bucketed
    > ) x
    > WHERE year IS NOT NULL
    > GROUP BY year
    > ORDER BY movies_count DESC
    > LIMIT 5;
```



```
--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 7.80 s
--------------------------------------------------------------------------------
OK
1996    345
1995    342
1998    337
1997    314
1999    283
```

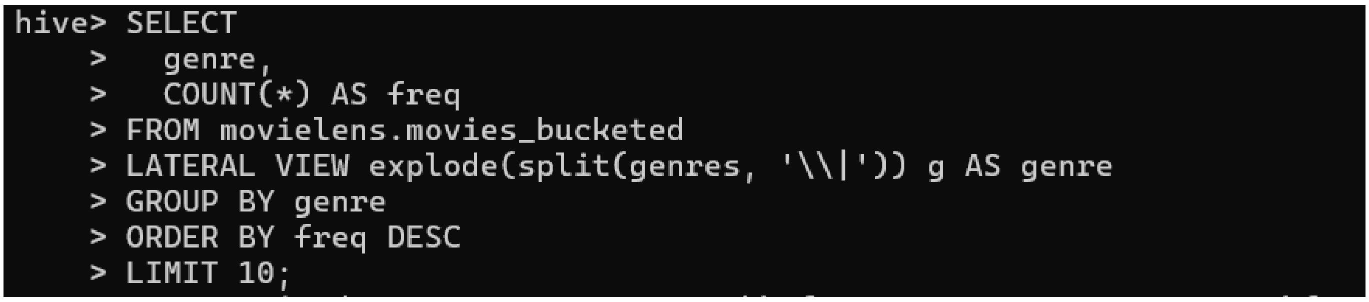
CONTENIDO

APARTADO E

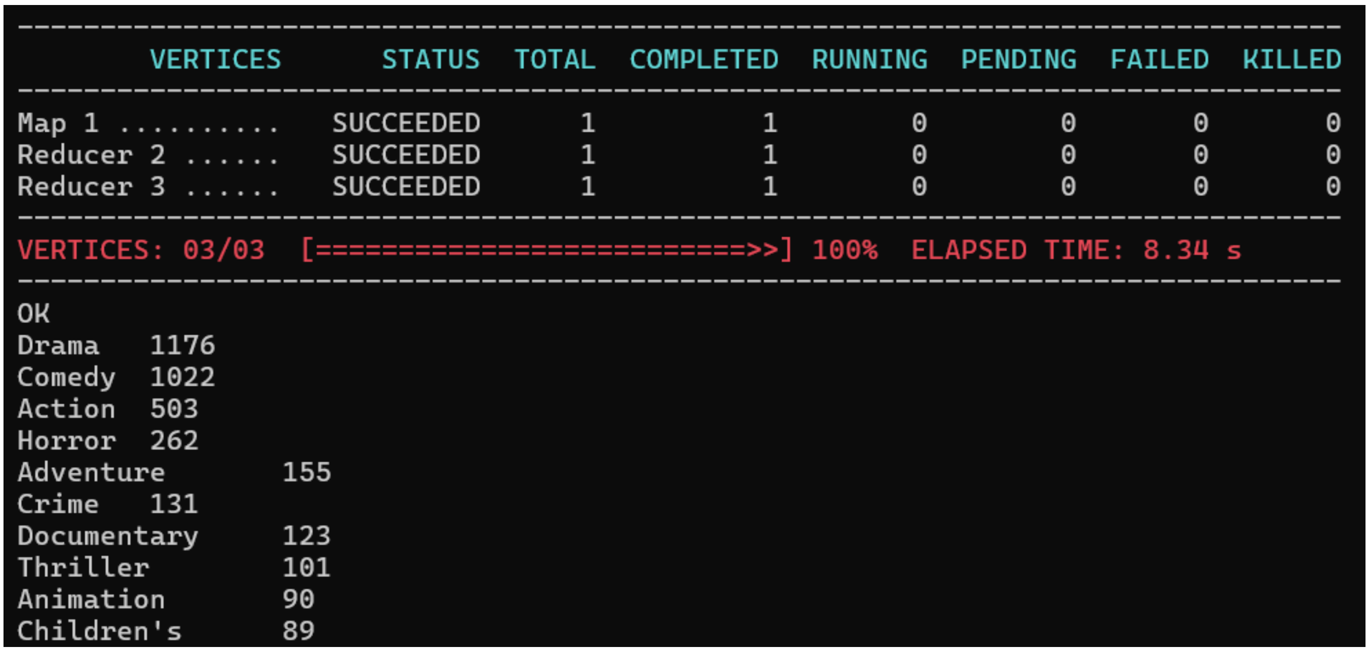
Consultando el enlace de abajo:

https://www.bigdatainrealworld.com/what-is-the-difference-between-explode-and-lateralview-explode-in-hive/

1. Muestra los diez géneros de películas más frecuentes.

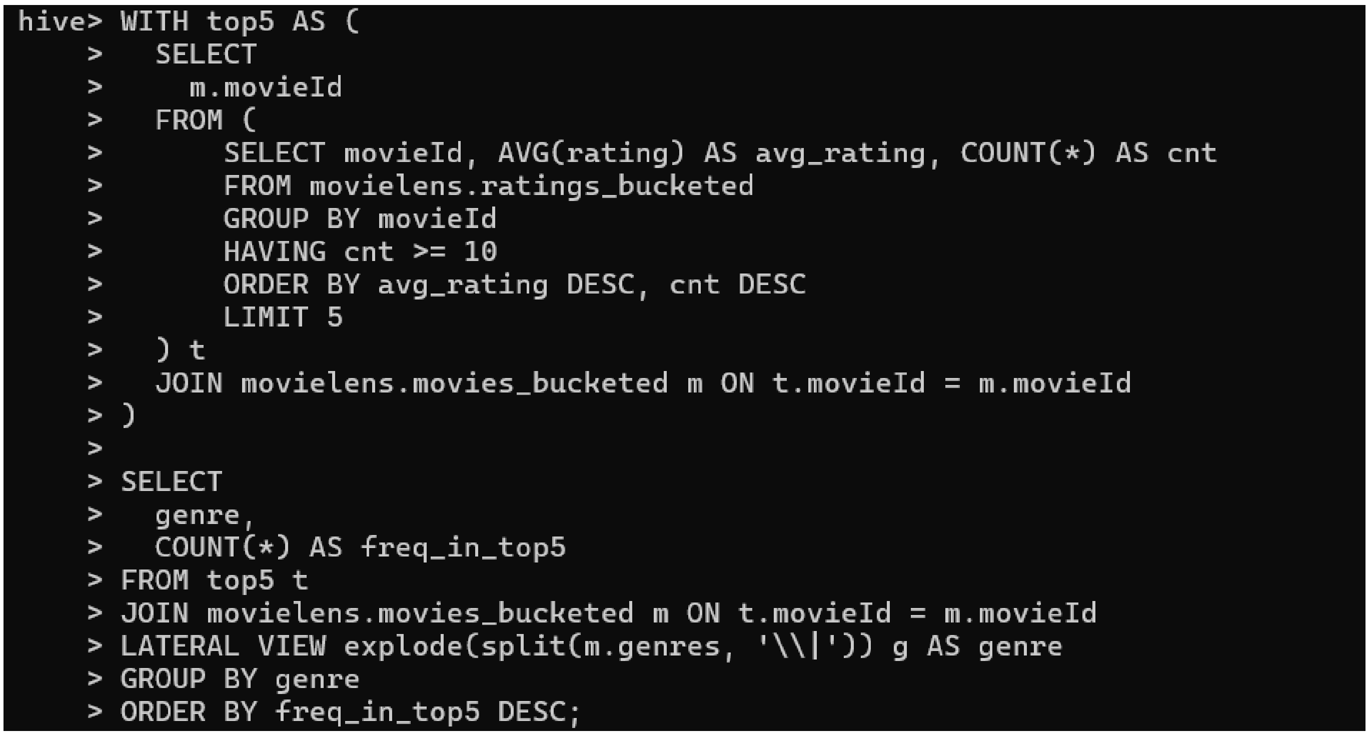


```
hive> SELECT
    >   genre,
    >   COUNT(*) AS freq
    > FROM movielens.movies_bucketed
    > LATERAL VIEW explode(split(genres, '\\|')) g AS genre
    > GROUP BY genre
    > ORDER BY freq DESC
    > LIMIT 10;
```



```
----------------------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 8.34 s
----------------------------------------------------------------------------------------------
OK
Drama   1176
Comedy  1022
Action  503
Horror  262
Adventure       155
Crime   131
Documentary     123
Thriller        101
Animation       90
Children's      89
```

2. Partiendo de la consulta del Apartado B de arriba. ¿Qué géneros son los más frecuentes en dichas películas?



```
hive> WITH top5 AS (
    >   SELECT
    >     m.movieId
    >   FROM (
    >       SELECT movieId, AVG(rating) AS avg_rating, COUNT(*) AS cnt
    >       FROM movielens.ratings_bucketed
    >       GROUP BY movieId
    >       HAVING cnt >= 10
    >       ORDER BY avg_rating DESC, cnt DESC
    >       LIMIT 5
    >   ) t
    >   JOIN movielens.movies_bucketed m ON t.movieId = m.movieId
    > )
    >
    > SELECT
    >   genre,
    >   COUNT(*) AS freq_in_top5
    > FROM top5 t
    > JOIN movielens.movies_bucketed m ON t.movieId = m.movieId
    > LATERAL VIEW explode(split(m.genres, '\\|')) g AS genre
    > GROUP BY genre
    > ORDER BY freq_in_top5 DESC;
```