Consultas con HIVE

1) 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.

Script utilizado para crear la base y las tablas:

```
Creamos la base de datos (el nombre es mío)
CREATE DATABASE IF NOT EXISTS filmThoughts;
USE filmThoughts;

Configuramos ciertas variables
SET hive.enforce.bucketing = true;
SET hive.exec.dynamic.partition.mode = nonstrict;

Creamos la tabla movies
DROP TABLE IF EXISTS movies;
CREATE TABLE movies (
    movie_id INT,
    title STRING,
    genres ARRAY<STRING>
)
CLUSTERED BY (movie_id) INTO 5 BUCKETS La dividimos en 5 buckets
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '*'
COLLECTION ITEMS TERMINATED BY '|' Para el campo genres
STORED AS TEXTFILE
LOCATION '/user/maria_dev/tarea_6_3/movies.dat'

Creamos la tabla ratings
DROP TABLE IF EXISTS ratings;
CREATE TABLE ratings (
    movie_id INT,
    user_id INT,
    rating INT,
    ts BIGINT
)
CLUSTERED BY (movie_id) INTO 5 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|' /* La dividimos en 5 buckets */
STORED AS TEXTFILE
LOCATION '/user/maria_dev/tarea_6_3/ratings.dat';
```

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

Consulta de movies

movies.movie_id	movies.title	movies.genres
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"]
8	Tom and Huck (1995)	["Adventure","Children's"]
9	Sudden Death (1995)	["Action"]
10	GoldenEye (1995)	["Action","Adventure","Thriller"]

Consulta de ratings

ratings.movie_id	ratings.user_id	ratings.rating	ratings.ts
1	1193	5	978300760
1	661	3	978302109
1	914	3	978301968
1	3408	4	978300275
1	2355	5	978824291
1	1197	3	978302268
1	1287	5	978302039
1	2804	5	978300719
1	594	4	978302268
1	919	4	978301368

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

Para movies:

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

movie_id                int
title                   string
genres                  array<string>

# Detailed Table Information
Database:               filmthoughts
Owner:                  maria_dev
CreateTime:             Tue Dec 09 10:16:25 UTC 2025
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev/tarea_6_3/movies.dat
Table Type:             EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                TRUE
        numFiles                1
        totalSize               163666
        transient_lastDdlTime   1765275385

# 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:            5
Bucket Columns:         [movie_id]
Sort Columns:           []
Storage Desc Params:
        colelction.delim        |
        field.delim             *
        serialization.format    *
Time taken: 1.199 seconds, Fetched: 33 row(s)
```

Para ratings:

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

movie_id                int
user_id                 int
rating                  int
ts                      bigint

# Detailed Table Information
Database:               filmthoughts
Owner:                  maria_dev
CreateTime:             Tue Dec 09 10:16:26 UTC 2025
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev/tarea_6_3/ratings.dat
Table Type:             EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                TRUE
        numFiles                1
        totalSize               21593504
        transient_lastDdlTime   1765275386

# 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:            5
Bucket Columns:         [movie_id]
Sort Columns:           []
Storage Desc Params:
        field.delim             |
        serialization.format    |
Time taken: 0.599 seconds, Fetched: 33 row(s)
```

2) 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.

```
SELECT
    m.movie_id,
    m.title,
    AVG(r.rating) AS avg_rating
FROM
    movies m
JOIN
    ratings r
ON
    m.movie_id = r.movie_id
GROUP BY
    m.movie_id, m.title
HAVING
    COUNT(r.user_id) >= 10
ORDER BY
    avg_rating DESC
LIMIT 5;
```

m.movie_id	m.title	avg_rating
283	New Jersey Drive (1995)	4.962962962962963
2339	I'll Be Home For Christmas (1998)	4.956521739130435
3324	Drowning Mona (2000)	4.904761904761905
3902	Goya in Bordeaux (Goya en Bodeos) (1999)	4.890909090909091
446	Farewell My Concubine (1993)	4.8431372549019605

3) APARTADO C

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

```
SELECT
    m.movie_id,
    m.title,
      CAST(REGEXP_EXTRACT(m.title, '\\((\\d{4})\\)$', 1) AS INT) AS year,
    AVG(r.rating) AS avg_rating
FROM
    movies m
JOIN
    ratings r
ON
    m.movie_id = r.movie_id
GROUP BY
    m.movie_id, m.title
HAVING
    avg_rating >= 4
ORDER BY
```

```
    year ASC
LIMIT 5;
```

m.movie_id	m.title	year	avg_rating
1490	B	null	4.1
3231	Saphead, The (1920)	1920	4.124183006535947
3310	Kid, The (1921)	1921	4.121212121212121
3629	Gold Rush, The (1925)	1925	4.161290322580645
957	Scarlet Letter, The (1926)	1926	4.271341463414634

4) APARTADO D

1) Investigando las funciones que nos ofrece HIVE para el manejo de cadenas... Muestra los cinco años en que se editaron más películas indicando el número de ellas para cada año.

```
SELECT
    CAST(REGEXP_EXTRACT(title, '\\((\\d{4})\\)$', 1) AS INT) AS year,
    COUNT(*) AS num_movies
FROM
    movies
GROUP BY
    CAST(REGEXP_EXTRACT(title, '\\((\\d{4})\\)$', 1) AS INT)
ORDER BY
    num_movies DESC
LIMIT 5;
```

year	num_movies
1996	345
1995	342
1998	337
1997	314
1999	283

5) APARTADO E

1) Muestra los diez géneros de películas más frecuentes.

```
SELECT
    genre,
```

```
    COUNT(*) AS num_movies
FROM movies
LATERAL VIEW explode(genres) g AS genre
GROUP BY genre
ORDER BY num_movies DESC
LIMIT 10;
```

genre	num_movies
Drama	1603
Comedy	1198
Action	503
Thriller	492
Romance	471
Horror	343
Adventure	283
Sci-Fi	276
Children's	251
Crime	211

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

```
SELECT
    g.genre,
    COUNT(*) AS freq
FROM (
    SELECT m.movie_id, m.genres
    FROM movies m
    JOIN ratings r
      ON m.movie_id = r.movie_id
    GROUP BY m.movie_id, m.genres
    HAVING COUNT(r.user_id) >= 10
) m
LATERAL VIEW explode(m.genres) g AS genre
GROUP BY g.genre
ORDER BY freq DESC;
```

g.genre	freq
Drama	1603
Comedy	1198
Action	503
Thriller	492
Romance	471
Horror	343
Adventure	283
Sci-Fi	276
Children's	251
Crime	211
War	142
Documentary	127
Musical	114
Mystery	106
Animation	105
Western	68
Fantasy	68
Film-Noir	44
A	2