Ejercicios de Apache HiveQL con Movielens

1) Gestión de Tablas y Formato de Datos (CREATE TABLE, EXTERNAL, ROW FORMAT)

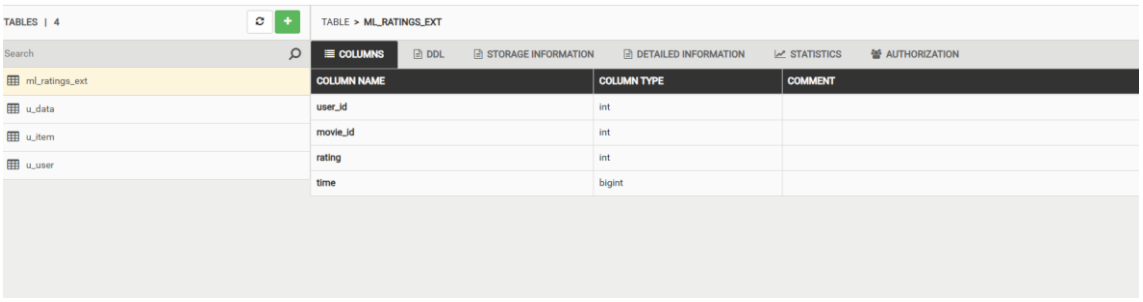
1) Carga de Datos Delimitados (Básico):

   i. Crea una Tabla Externa llamada ml_ratings_ext para el archivo u.data.
   ii. Asegúrate de especificar que los campos están delimitados por el carácter tabulador (\t) y que las filas terminan en salto de línea.
   iii. Define las cuatro columnas: user_id , movie_id , rating (INT) y timestamp (BIGINT).

Usamos este script

```
CREATE EXTERNAL TABLE ml_ratings_ext (
    user_id INT,
    movie_id INT,
    rating INT,
    time BIGINT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
LOCATION '/user/maria_dev/movielens/u.data';
```

Resultado:



2) Definición de Esquema y Comentarios:

   i. Crea una Tabla Gestionada (Managed) llamada ml_users_managed para el archivo u.user .
   ii. Especifica la delimitadora barra vertical (|).
   iii. Añade un COMMENT a la tabla y un COMMENT a la columna occupation explicando su uso.

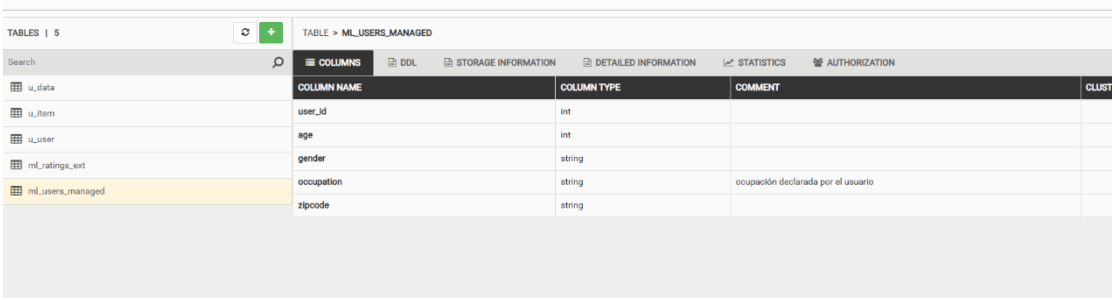
Usamos este script

```
USE movielens2;

DROP TABLE IF EXISTS ml_users_managed;

CREATE TABLE ml_users_managed (
    user_id INT,
    age INT,
    gender STRING,
    occupation STRING COMMENT 'ocupación declarada por el usuario',
    zipcode STRING
)
COMMENT 'tabla gestionada con información de usuarios de MovieLens'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n';
```

Resultado:



3) Simulación de Pérdida de Datos (DROP TABLE):

   i. Ejecuta DROP TABLE ml_ratings_ext; . ¿Se eliminaron los datos de HDFS? Explica por qué.

"ml_ratings_ext" es una tabla externa. Hive solo elimina el metadato, pero no borra los archivos del directorio HDFS indicado en LOCATION.

   ii. Ejecuta DROP TABLE ml_users_managed; . ¿Se eliminaron los datos de HDFS? Explica por qué.

"ml_users_managed" es una tabla gestionada (MANAGED). En este caso, Hive elimina tanto el metadato como los archivos físicos asociados al almacén interno de la base de datos.

2) Tipos de Datos Complejos (ARRAY, MAP, STRUCT)

1) Uso de STRUCT (Información Personal):

   i. Crea una nueva tabla ml_user_info basada en u_user .

```
USE movielens2;

DROP TABLE IF EXISTS ml_user_info;
```

```
CREATE TABLE ml_user_info (
    user_id INT,
    personal_data STRUCT<
        age: INT,
        gender: STRING
    >,
    occupation STRING,
    zipcode STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n';
```

ii. Combina las columnas age (INT) y gender (STRING) en una sola columna de tipo STRUCT llamada personal_data.

```
INSERT INTO ml_user_info
SELECT
    user_id,
    named_struct('age', age, 'gender', gender),
    occupation,
    zipcode
FROM u_user;
```

iii. Escribe una consulta para seleccionar el campo gender de esta nueva columna y filtra por todas las mujeres ('F').

```
SELECT user_id, personal_data.gender FROM ml_user_info
   WHERE personal_data.gender = 'F';
```

user_id	gender
2	F
5	F
11	F
12	F
15	F
18	F
20	F
23	F
24	F
27	F
32	F
34	F
35	F
36	F

2) Uso de ARRAY (Géneros de Película):

   i. El archivo u.item tiene 19 columnas binarias para géneros. Simula que ya han sido procesadas en una columna de tipo ARRAY llamada genres_list (ARRAY<STRING>).
   ii. Escribe una consulta que devuelva el tamaño (size()) de esta lista de géneros para cada película.

```
SELECT
    movie_id,
    movie_title,
    size(genres_list) AS num_genres
FROM ml_items_array;
```

   iii. Muestra el segundo género (índice 1) de la película con movie_id = 50.

```
SELECT
    movie_id,
    genres_list[1] AS second_genre
FROM ml_items_array
WHERE movie_id = 50;
```

3) Uso de MAP (Puntuación de Usuarios por Ocupación):

   i. Simula que tienes una tabla que resume la puntuación promedio (FLOAT) de los usuarios en función de su ocupación (STRING), almacenada en un MAP<STRING, FLOAT> llamado avg_rating_by_occupation .
   ii. Utiliza la función map_keys() para listar todas las ocupaciones diferentes que existen en el MAP .

```
SELECT
    user_id,
    map_keys(avg_rating_by_occupation) AS occupations
FROM ml_user_ratings_map;
```

3) Consultas Básicas y JOINS

1) Agregación Básica:

   i. Calcula el promedio de todas las calificaciones (rating) que existen en la tabla ml_ratings_ext .

```
SELECT AVG(rating) AS avg_rating
FROM ml_ratings_ext;
```

avg_rating
3.52986

2) Filtrado y Agrupación:

   i. Agrupa los usuarios por gender y occupation (de ml_users_managed).
   ii. Calcula el número de usuarios que hay en cada combinación (e.g., 'M', 'programmer' , 'F', 'student').
   iii. Ordena el resultado de forma descendente por el conteo.

```
SELECT
    gender,
    occupation,
    COUNT(*) AS num_users
FROM ml_users_managed
GROUP BY gender, occupation
ORDER BY num_users DESC;
```

gender	occupation	num_users
M	student	136
M	other	69
M	educator	69
M	engineer	65
F	student	60
M	programmer	60
M	administrator	43
F	administrator	36
F	other	36
M	executive	29
F	librarian	29
M	scientist	28
M	technician	26
M	writer	26
F	educator	26
M	librarian	22
F	writer	19
M	entertainment	16

3) INNER JOIN:

   i. Utiliza un INNER JOIN entre ml_ratings_ext y ml_users_managed sobre user_id .
   ii. Calcula la calificación promedio por occupation .

```
SELECT
    u.occupation,
    AVG(r.rating) AS avg_rating
```

```
FROM ml_ratings_ext r
INNER JOIN ml_users_managed u
    ON r.user_id = u.user_id
GROUP BY u.occupation
ORDER BY avg_rating DESC;
```

u.occupation	avg_rating
none	3.779134295227525
lawyer	3.7353159851301116
doctor	3.688888888888889
educator	3.6706206312221985
artist	3.653379549393414
administrator	3.6356464768017114
scientist	3.611273080660836
salesman	3.582943925233645
programmer	3.5682604794257147
librarian	3.560781338896264
other	3.5523773797242804
engineer	3.541406727828746
technician	3.5322304620650313
student	3.5151432345038027
marketing	3.4856410256410255

4) LEFT OUTER JOIN:

   i. Crea una tabla ficticia unrated_users que contenga solo 5 user_id s de la tabla ml_users_managed que no hayan calificado ninguna película.

```
DROP TABLE IF EXISTS unrated_users;

CREATE TABLE unrated_users AS
SELECT u.user_id
FROM ml_users_managed u
LEFT JOIN ml_ratings_ext r
    ON u.user_id = r.user_id
WHERE r.user_id IS NULL
LIMIT 5;
```

   ii. Realiza un LEFT OUTER JOIN desde ml_users_managed a ml_ratings_ext .

iii. Filtra el resultado para encontrar aquellos usuarios que no tienen calificaciones (donde el rating es NULL después del join).

```
SELECT
    u.user_id,
    u.gender,
    u.occupation
FROM ml_users_managed u
LEFT OUTER JOIN ml_ratings_ext r
    ON u.user_id = r.user_id
WHERE r.rating IS NULL;
```

u.user_id	u.gender	u.occupation

5) LEFT SEMI JOIN (Simulación de IN/EXISTS):

i. Usando ml_ratings_ext y ml_users_managed , lista todos los user_id que tienen una ocupación de 'programmer' y que han calificado al menos una película. (Debe usarse LEFT SEMI JOIN , no un INNER JOIN simple).

```
SELECT u.user_id
FROM ml_users_managed u
LEFT SEMI JOIN ml_ratings_ext r
    ON u.user_id = r.user_id
WHERE u.occupation = 'programmer';
```

u.user_id
17
29
45
53
55
58
82
102