

PR_05.1 Dani Gayol Rodríguez

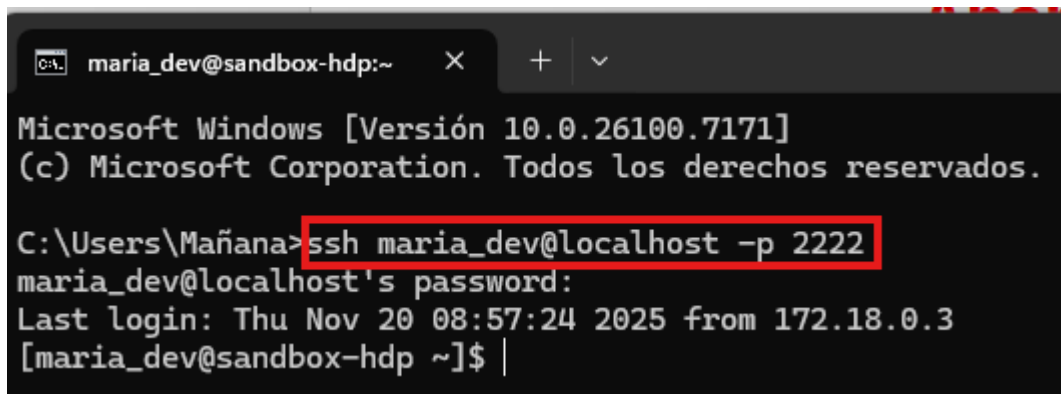
PR_05.1 Dani Gayol Rodríguez.....	1
Apartado A.....	1
1.) En la carpeta del usuario maria_dev en HDFS crea una subcarpeta llamada sqoop donde guardaremos los archivos de esta práctica.	2
2.) Importa con Sqoop las tres tablas que creamos en MySQL en la práctica anterior. 2	
3.) Importa, dejándola en un solo archivo, todos los datos de los empleados adjuntándoles a cada uno toda la información de su departamento.	3
4.) Muestra en HDFS la ubicación y contenido de los ficheros resultantes.	4
Apartado B.....	4
1.) En la carpeta del usuario maria_dev en HDFS crea una subcarpeta llamada movielens donde guardaremos los archivos u.data, u.user y u.item.	5
2.) Utilizando PIG, al archivo u.user quítale la última columna con el código postal. Guarda el resultado en el archivo u.user2.	5
3.) Utilizando PIG, del archivo u.item quédate solamente con las dos primeras columnas (id y título). Posteriormente de la columna título extrae el año y guárdala en una nueva columna (anio). En la columna título ha de quedar exclusivamente el título sin el año. Guarda el resultado en un archivo llamado u.item2	6
Apartado C	7
1.) Crea en tu servidor MySQL de AWS una nueva base de datos llamada movielens. .8	
2.) En dicha base de datos crea tres tablas (usuarios, votos y películas) con la estructura adecuada para almacenar los ficheros u.data, u.user2 y u.item2. Utiliza sentencias CREATE TABLE. Crea los índices y relaciones entre las tres tablas.	8
3.) Utilizando SQOOP, exporta los tres ficheros de HDFS a sus tablas.	8
4.) Comprueba con sentencias SELECT que los ficheros se importaron correctamente.....	9
Apartado D	10
Realiza las siguientes consultas utilizando DBEaver	11
1.) Top 10 películas más votadas de todos los tiempos (número de votos, no el valor de este).....	11
2.) Películas con nota media ≥ 4.5 y al menos 100 valoraciones.....	11

3.) Usuarios que han dado más de 300 valoraciones y su nota media.....	11
4.) Año con más películas votadas (por número total de votos).....	12
5.) Las 5 películas más "polarizadas" (mayor desviación estándar, con valoraciones muy extremas) con al menos 50 votos. (Investiga qué función de SQL de da la desviación estándar).....	12
6.) Usuarios cuya nota media es menor que la nota media global.	13
7.) Películas que han recibido al menos una valoración de 1 y una de 5 (las más divididas).	13
8.) Top 10 usuarios más activos en 1997 (por número de valoraciones ese año)...	14
9.) Películas estrenadas después de 1995 con mejor nota media que "Toy Story (1995).	14
10.) Usuarios que han valorado todas las películas estrenadas en 1993.....	15
11.) Evolución mensual del número de valoraciones en 1998.	15
12.) Las 5 películas con mayor aumento de popularidad (comparar 1997 vs 1998).	16
13.) Usuarios que han valorado más películas que la media de su género.	16
14.) Películas que nadie ha valorado con 3 estrellas (solo 1,2,4,5).	17
15.) Ranking de días de la semana con más actividad (lunes, martes...).	18
16.) Usuarios que han dado su primera y última valoración con diferencia > 6 meses.	18
17.) Las 10 películas con mayor ratio 5-estrellas / total valoraciones.	19
18.) UPDATE: Aumenta en 1 año la edad de todos los usuarios (simulación de paso del tiempo).	19
19.) INSERT: Añade una nueva película ficticia estrenada hoy.	20
20.) DELETE: Elimina todas las valoraciones anteriores a 1997.	21

Apartado A

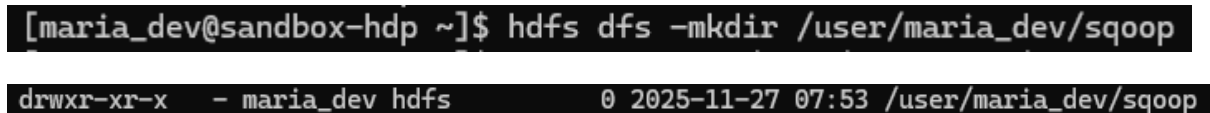
1.) En la carpeta del usuario maria_dev en HDFS crea una subcarpeta llamada sqoop donde guardaremos los archivos de esta práctica.

Para crear esta subcarpeta, primero tenemos que arrancar la máquina virtual que ya tenemos de anteriores prácticas, una vez arrancada ya nos podemos conectar a ella mediante “ssh” y crear la subcarpeta



```
C:\Users\Mañana> ssh maria_dev@localhost -p 2222
maria_dev@localhost's password:
Last login: Thu Nov 20 08:57:24 2025 from 172.18.0.3
[maria_dev@sandbox-hdp ~]$
```

Una vez conectados, creamos la subcarpeta con “mkdir”

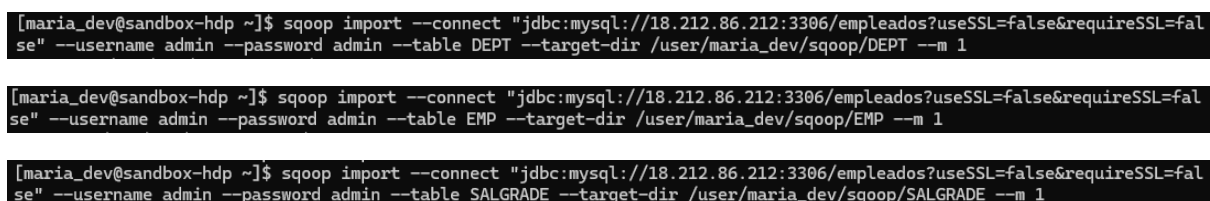


```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -mkdir /user/maria_dev/sqoop
drwxr-xr-x  - maria_dev hdfs          0 2025-11-27 07:53 /user/maria_dev/sqoop
```

2.) Importa con Sqoop las tres tablas que creamos en MySQL en la práctica anterior.

Ahora para importar las tablas creadas anteriormente, tenemos que lanzar el laboratorio que tenemos en AWS, una vez arrancado, nos vamos a “EC2” y dentro de ahí aparece nuestra instancia con el “MySQL”

Ahora para importarlas, utilizamos este comando para cada tabla:



```
[maria_dev@sandbox-hdp ~]$ sqoop import --connect "jdbc:mysql://18.212.86.212:3306/empleados?useSSL=false&requireSSL=false" --username admin --password admin --table DEPT --target-dir /user/maria_dev/sqoop/DEPT --m 1

[maria_dev@sandbox-hdp ~]$ sqoop import --connect "jdbc:mysql://18.212.86.212:3306/empleados?useSSL=false&requireSSL=false" --username admin --password admin --table EMP --target-dir /user/maria_dev/sqoop/EMP --m 1

[maria_dev@sandbox-hdp ~]$ sqoop import --connect "jdbc:mysql://18.212.86.212:3306/empleados?useSSL=false&requireSSL=false" --username admin --password admin --table SALGRADE --target-dir /user/maria_dev/sqoop/SALGRADE --m 1
```

Después de importarlas, vamos a comprobar que se importaron correctamente las tablas y los datos de cada tabla

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls /user/maria_dev/sqoop/
Found 3 items
drwxr-xr-x - maria_dev hdfs      0 2025-11-27 08:20 /user/maria_dev/sqoop/DEPT
drwxr-xr-x - maria_dev hdfs      0 2025-11-27 08:24 /user/maria_dev/sqoop/EMP
drwxr-xr-x - maria_dev hdfs      0 2025-11-27 08:26 /user/maria_dev/sqoop/SALGRADE
```

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -cat /user/maria_dev/sqoop/DEPT/pa*
10,ACCOUNTING,NEW YORK
20,RESEARCH,DALLAS
30,SALES,CHICAGO
40,OPERATIONS,BOSTON
```

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -cat /user/maria_dev/sqoop/EMP/pa*
null,20,SMITH,7369,1980-12-17,CLERK,7902,800.00
300.00,30,ALLEN,7499,1981-02-20,SALESMAN,7698,1600.00
500.00,30,WARD,7521,1981-02-22,SALESMAN,7698,1250.00
null,20,JONES,7566,1981-04-02,MANAGER,7839,2975.00
1400.00,30,MARTIN,7654,1981-10-28,SALESMAN,7698,1250.00
null,30,BLAKE,7698,1981-05-01,MANAGER,7839,2850.00
null,10,CLARK,7782,1981-06-09,MANAGER,7839,2450.00
null,20,SCOTT,7788,1982-12-09,ANALYST,7566,3000.00
null,10,KING,7839,1981-11-17,PRESIDENT,null,5000.00
0.00,30,TURNER,7844,1981-10-08,SALESMAN,7698,1500.00
null,20,ADAMS,7876,1983-01-12,CLERK,7788,1100.00
null,30,JAMES,7900,1981-12-03,CLERK,7698,950.00
null,20,FORD,7902,1981-12-03,ANALYST,7566,3000.00
null,10,MILLER,7934,1982-01-23,CLERK,7782,1300.00
```

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -cat /user/maria_dev/sqoop/SALGRADE/pa*
1,1200.00,700.00
2,1400.00,1201.00
3,2000.00,1401.00
4,3000.00,2001.00
5,9999.00,3001.00
```

3.) Importa, dejándola en un solo archivo, todos los datos de los empleados adjuntándoles a cada uno toda la información de su departamento.

Para hacer esto, es parecido al apartado anterior, el único cambio que hay que hacer es sustituir el “table” por “query” y poner ahí lo necesario para importarla como nos piden

```
[maria_dev@sandbox-hdp ~]$ sqoop import --connect "jdbc:mysql://18.212.86.212:3306/empleados?useSSL=false&requireSSL=false" --username admin --password admin --query "SELECT e.*, d.deptno AS dept_id, d.dname AS dept_nombre, d.loc AS dept_loc FROM EMP e JOIN DEPT d ON e.deptno = d.deptno WHERE \$CONDITIONS" --target-dir /user/maria_dev/sqoop/EMP_DEPT --m 1
```

Vamos a comprobar que todo se importó correctamente usando el comando “ls” y el comando “cat”

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls /user/maria_dev/sqoop/
Found 4 items
drwxr-xr-x - maria_dev hdfs      0 2025-11-27 08:20 /user/maria_dev/sqoop/DEPT
drwxr-xr-x - maria_dev hdfs      0 2025-11-27 08:24 /user/maria_dev/sqoop/EMP
drwxr-xr-x - maria_dev hdfs      0 2025-11-27 08:42 /user/maria_dev/sqoop/EMP_DEPT
drwxr-xr-x - maria_dev hdfs      0 2025-11-27 08:26 /user/maria_dev/sqoop/SALGRADE
```

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -cat /user/maria_dev/sqoop/EMP_DEPT/pa*
7782,CLARK,MANAGER,7839,1981-06-09,2450.00,null,10,10,ACCOUNTING,NEW YORK
7839,KING,PRESIDENT,null,1981-11-17,5000.00,null,10,10,ACCOUNTING,NEW YORK
7934,MILLER,CLERK,7782,1982-01-23,1300.00,null,10,10,ACCOUNTING,NEW YORK
7369,SMITH,CLERK,7902,1980-12-17,800.00,null,20,20,RESEARCH,DALLAS
7566,JONES,MANAGER,7839,1981-04-02,2975.00,null,20,20,RESEARCH,DALLAS
7788,SCOTT,ANALYST,7566,1982-12-09,3000.00,null,20,20,RESEARCH,DALLAS
7876,ADAMS,CLERK,7788,1983-01-12,1100.00,null,20,20,RESEARCH,DALLAS
7902,FORD,ANALYST,7566,1981-12-03,3000.00,null,20,20,RESEARCH,DALLAS
7499,ALLEN,SALESMAN,7698,1981-02-20,1600.00,300.00,30,30,SALES,CHICAGO
7521,WARD,SALESMAN,7698,1981-02-22,1250.00,500.00,30,30,SALES,CHICAGO
7654,MARTIN,SALESMAN,7698,1981-10-28,1250.00,1400.00,30,30,SALES,CHICAGO
7698,BLAKE,MANAGER,7839,1981-05-01,2850.00,null,30,30,SALES,CHICAGO
7844,TURNER,SALESMAN,7698,1981-10-08,1500.00,0.00,30,30,SALES,CHICAGO
7900,JAMES,CLERK,7698,1981-12-03,950.00,null,30,30,SALES,CHICAGO
```

4.) Muestra en HDFS la ubicación y contenido de los ficheros resultantes.

Ahora como ya hice anteriormente, comprobamos que todo esté bien

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls /user/maria_dev/sqoop/
Found 4 items
drwxr-xr-x - maria_dev hdfs      0 2025-11-27 08:20 /user/maria_dev/sqoop/DEPT
drwxr-xr-x - maria_dev hdfs      0 2025-11-27 08:24 /user/maria_dev/sqoop/EMP
drwxr-xr-x - maria_dev hdfs      0 2025-11-27 08:42 /user/maria_dev/sqoop/EMP_DEPT
drwxr-xr-x - maria_dev hdfs      0 2025-11-27 08:26 /user/maria_dev/sqoop/SALGRADE
```

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls /user/maria_dev/sqoop/DEPT
Found 2 items
-rw-r--r-- 1 maria_dev hdfs      0 2025-11-27 08:20 /user/maria_dev/sqoop/DEPT/_SUCCESS
-rw-r--r-- 1 maria_dev hdfs    80 2025-11-27 08:20 /user/maria_dev/sqoop/DEPT/part-m-00000
```

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -cat /user/maria_dev/sqoop/DEPT/pa*
10,ACCOUNTING,NEW YORK
20,RESEARCH,DALLAS
30,SALES,CHICAGO
40,OPERATIONS,BOSTON
```

Y las demás comprobaciones ya las hice en los ejercicios anteriores

Apartado B

1.) En la carpeta del usuario maria_dev en HDFS crea una subcarpeta llamada movielens donde guardaremos los archivos u.data, u.user y u.item.

Como los archivos de “u.user”, “u.data” y “u.item” ya los teníamos importados en el HDFS de anteriores prácticas, solo tenemos que crear la subcarpeta “movielens” y copiarle los archivos ahí

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -mkdir /user/maria_dev/movielens
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls /user/maria_dev/
Found 24 items
drwx----- - maria_dev hdfs      0 2025-11-12 10:38 /user/maria_dev/.Trash
drwx----- - maria_dev hdfs      0 2025-11-27 08:42 /user/maria_dev/.staging
drwxr-xr-x - maria_dev hdfs      0 2025-11-11 09:16 /user/maria_dev/datos
-rw-r--r-- 1 maria_dev hdfs 2226045 2025-11-17 11:14 /user/maria_dev/el_quijote.txt
drwxr-xr-x - maria_dev hdfs      0 2025-11-19 10:11 /user/maria_dev/maria_dev
drwxr-xr-x - maria_dev hdfs      0 2025-12-01 07:43 /user/maria_dev/movielens
drwxr-xr-x - maria_dev hdfs      0 2025-11-13 09:15 /user/maria_dev/pig_usuarios
drwxr-xr-x - maria_dev hdfs      0 2025-11-13 09:10 /user/maria_dev/practica_pig
```

Una vez creado la subcarpeta, vamos a pegar los archivos ahí

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -put u.data /user/maria_dev/movielens
[maria_dev@sandbox-hdp ~]$ hdfs dfs -put u.user /user/maria_dev/movielens
[maria_dev@sandbox-hdp ~]$ hdfs dfs -put u.item /user/maria_dev/movielens
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls /user/maria_dev/movielens/
Found 3 items
-rw-r--r-- 1 maria_dev hdfs 1979173 2025-12-01 07:44 /user/maria_dev/movielens/u.data
-rw-r--r-- 1 maria_dev hdfs 236344 2025-12-01 07:45 /user/maria_dev/movielens/u.item
-rw-r--r-- 1 maria_dev hdfs 22628 2025-12-01 07:44 /user/maria_dev/movielens/u.user
```

2.) Utilizando PIG, al archivo u.user quítale la última columna con el código postal. Guarda el resultado en el archivo u.user2.

Ahora utilizando “pig” vamos a cargar los datos del archivo para quitarle la última columna

```
grunt> users = LOAD '/user/maria_dev/movielens/u.user' USING PigStorage(',') AS (user_id:int, age:int, gender:chararray, occupation:chararray, zip:chararray);
```

Una vez cargados los datos, vamos a eliminar la última columna

```
grunt> users2 = FOREACH users GENERATE user_id, age, gender, occupation;
```

Finalmente, vamos a guardar el resultado como “u.user2”

```
grunt> STORE users2 INTO '/user/maria_dev/movielens/u.user2' USING PigStorage('|');
```

Ahora vamos a comprobar que todo se realizó correctamente

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls /user/maria_dev/movielens/
Found 4 items
-rw-r--r-- 1 maria_dev hdfs 1979173 2025-12-01 07:44 /user/maria_dev/movielens/u.data
-rw-r--r-- 1 maria_dev hdfs 236344 2025-12-01 07:45 /user/maria_dev/movielens/u.item
-rw-r--r-- 1 maria_dev hdfs 22628 2025-12-01 07:44 /user/maria_dev/movielens/u.user
drwxr-xr-x - maria_dev hdfs 0 2025-12-01 07:52 /user/maria_dev/movielens/u.user2
[maria_dev@sandbox-hdp ~]$ hdfs dfs -cat /user/maria_dev/movielens/u.user2/par*
1|24|M|technician
2|53|F|other
3|23|M|writer
4|24|M|technician
5|33|F|other
6|42|M|executive
7|57|M|administrator
8|36|M|administrator
9|29|M|student
10|53|M|lawyer
11|39|F|other
```

3.) Utilizando PIG, del archivo u.item quédate solamente con las dos primeras columnas (id y título). Posteriormente de la columna título extrae el año y guárdala en una nueva columna (anio). En la columna título ha de quedar exclusivamente el título sin el año. Guarda el resultado en un archivo llamado u.item2

Vamos a hacer algo parecido que el ejercicio anterior, primero abrimos “pig” y cargamos los datos

```
grunt> items = LOAD '/user/maria_dev/movielens/u.item' USING PigStorage('|') AS (movie_id:int, title:chararray, rest:chararray);
```

Una vez cargados los datos, vamos a extraer el año

```
grunt> items2 = FOREACH items GENERATE movie_id, REPLACE(title, '\\s*\\{[0-9]{4}\\}', '') AS title_clean, REGEX_EXTRACT(title, '.*\\{[0-9]{4}\\}', 1) AS anio;
```

Y ahora vamos a guardar el resultado como “u.item2”

```
grunt> STORE items2 INTO '/user/maria_dev/movielens/u.item2' USING PigStorage('|');
```

Finalmente, comprobamos que todo funciona correctamente

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls /user/maria_dev/movielens/
Found 5 items
-rw-r--r-- 1 maria_dev hdfs 1979173 2025-12-01 07:44 /user/maria_dev/movielens/u.data
-rw-r--r-- 1 maria_dev hdfs 236344 2025-12-01 07:45 /user/maria_dev/movielens/u.item
drwxr-xr-x - maria_dev hdfs 0 2025-12-01 08:07 /user/maria_dev/movielens/u.item2
-rw-r--r-- 1 maria_dev hdfs 22628 2025-12-01 07:44 /user/maria_dev/movielens/u.user
drwxr-xr-x - maria_dev hdfs 0 2025-12-01 07:52 /user/maria_dev/movielens/u.user2
[maria_dev@sandbox-hdp ~]$ hdfs dfs -cat /user/maria_dev/movielens/u.item2/par*
1|Toy Story|1995
2|GoldenEye|1995
3|Four Rooms|1995
4|Get Shorty|1995
5|Copycat|1995
6|Shanghai Triad (Yao a yao yao dao waipo qiao)|1995
7|Twelve Monkeys|1995
8|Babe|1995
9|Dead Man Walking|1995
10|Richard III|1995
```


Apartado C

1.) Crea en tu servidor MySQL de AWS una nueva base de datos llamada movielens.

Para hacer este ejercicio, tenemos que arrancar el servidor MySQL de AWS y conectarnos a el

```
C:\Users\Mañana>ssh -i "C:\Users\Mañana\Downloads\ubuntu.pem" ubuntu@54.86.196.253|
```

Una vez conectados, vamos a crear la base de datos “movielens”

```
ubuntu@ip-172-31-21-121:~$ mysql -h 54.86.196.253 -u root -p
```

```
mysql> CREATE DATABASE movielens;  
Query OK, 1 row affected (0.01 sec)  
  
mysql> USE movielens;  
Database changed
```

2.) En dicha base de datos crea tres tablas (usuarios, votos y películas) con la estructura adecuada para almacenar los ficheros u.data, u.user2 y u.item2. Utiliza sentencias CREATE TABLE. Crea los índices y relaciones entre las tres tablas.

Ahora vamos a crear las tres tablas

```
mysql> CREATE TABLE usuarios (user_id INT PRIMARY KEY, age INT, gender VARCHAR(10), occupation VARCHAR(50));  
Query OK, 0 rows affected (0.05 sec)  
  
mysql> CREATE TABLE peliculas (movie_id INT PRIMARY KEY, title VARCHAR(255), anio CHAR(4));  
Query OK, 0 rows affected (0.04 sec)  
  
mysql> CREATE TABLE votos (user_id INT, movie_id INT, rating INT, fecha BIGINT, PRIMARY KEY (user_id, movie_id, fecha),  
FOREIGN KEY (user_id) REFERENCES usuarios(user_id), FOREIGN KEY (movie_id) REFERENCES peliculas(movie_id));  
Query OK, 0 rows affected (0.06 sec)
```

3.) Utilizando SQOOP, exporta los tres ficheros de HDFS a sus tablas.

Una vez creadas las tablas, vamos a exportar los ficheros usando sqoop, para ello, tenemos que hacer lo siguiente

```
[maria_dev@sandbox-hdp ~]$ sqoop export --connect jdbc:mysql://54.86.196.253:3306/movielens?useSSL=false --username admin --password admin --table usuarios --export-dir /user/maria_dev/movielens/u.user2/part-v000-o000-r-00000 --input-fields-terminated-by '|' --m 1
```

```
[maria_dev@sandbox-hdp ~]$ sqoop export --connect jdbc:mysql://54.86.196.253:3306/movielens?useSSL=false --username admin --password admin --table peliculas --export-dir /user/maria_dev/movielens/u.item2/part-v000-o000-r-00000 --input-fields-terminated-by '|' --m 1
```

```
[maria_dev@sandbox-hdp ~]$ sqoop export --connect jdbc:mysql://54.86.196.253:3306/movielens?useSSL=false --username admin --password admin --table votos --export-dir /user/maria_dev/movielens/u.data --input-fields-terminated-by '\t' --m 1
```

4.) Comprueba con sentencias SELECT que los ficheros se importaron correctamente.

Finalmente, vamos a comprobar que se importó todo correctamente

```
mysql> SELECT * FROM usuarios LIMIT 10;
```

user_id	age	gender	occupation
1	24	M	technician
2	53	F	other
3	23	M	writer
4	24	M	technician
5	33	F	other
6	42	M	executive
7	57	M	administrator
8	36	M	administrator
9	29	M	student
10	53	M	lawyer

```
10 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM peliculas LIMIT 10;
```

movie_id	title	anio
1	Toy Story	1995
2	GoldenEye	1995
3	Four Rooms	1995
4	Get Shorty	1995
5	Copycat	1995
6	Shanghai Triad (Yao a yao yao dao waipo qiao)	1995
7	Twelve Monkeys	1995
8	Babe	1995
9	Dead Man Walking	1995
10	Richard III	1995

```
10 rows in set (0.01 sec)
```

```
mysql> SELECT * FROM votos LIMIT 10;
```

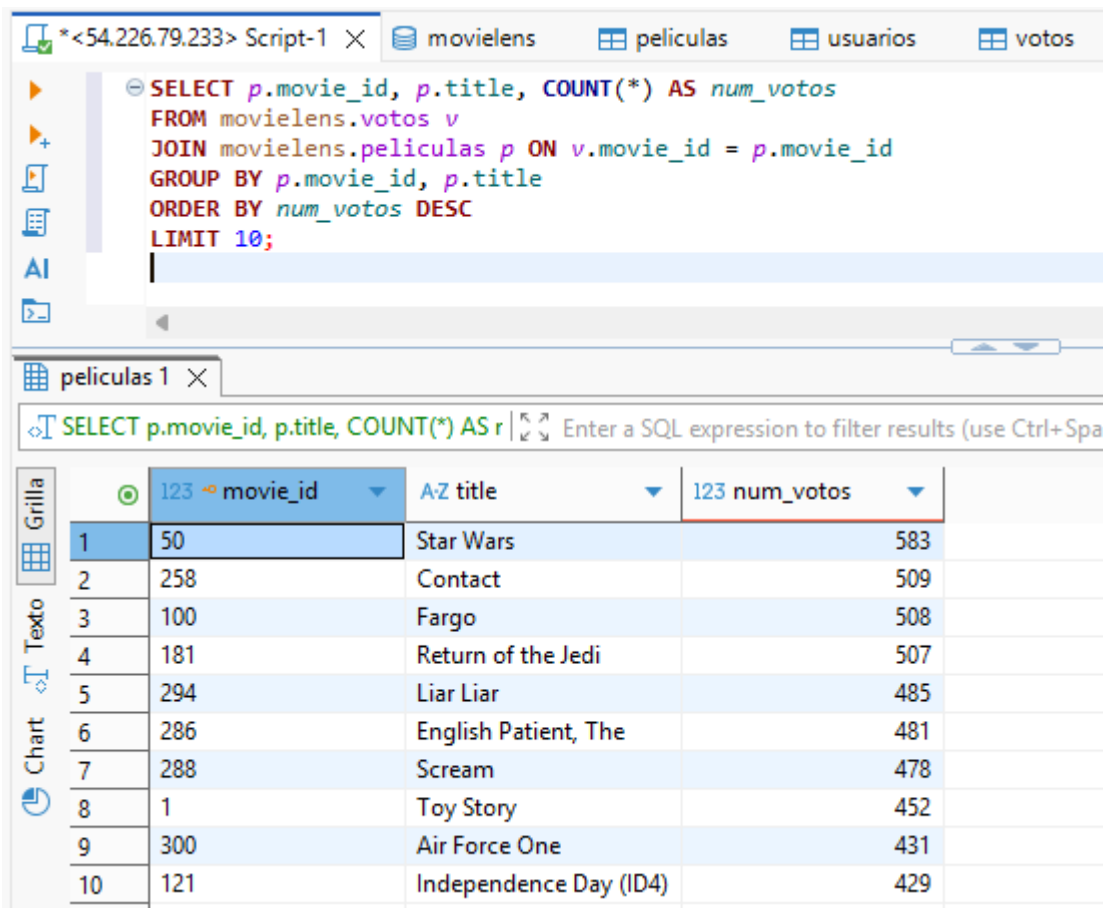
user_id	movie_id	rating	fecha
1	1	5	874965758
1	2	3	876893171
1	3	4	878542960
1	4	3	876893119
1	5	3	889751712
1	6	5	887431973
1	7	4	875071561
1	8	1	875072484
1	9	5	878543541
1	10	3	875693118

```
10 rows in set (0.00 sec)
```

Apartado D

Realiza las siguientes consultas utilizando DBeaver

1.) Top 10 películas más votadas de todos los tiempos (número de votos, no el valor de este).



The screenshot shows the DBeaver interface with a SQL query executed against the 'movielens' database. The query selects the top 10 movies by vote count. The results are displayed in a table with columns for rank, movie ID, title, and vote count.

```
SELECT p.movie_id, p.title, COUNT(*) AS num_votos
FROM movielens.votos v
JOIN movielens.peliculas p ON v.movie_id = p.movie_id
GROUP BY p.movie_id, p.title
ORDER BY num_votos DESC
LIMIT 10;
```

	123 movie_id	AZ title	123 num_votos
1	50	Star Wars	583
2	258	Contact	509
3	100	Fargo	508
4	181	Return of the Jedi	507
5	294	Liar Liar	485
6	286	English Patient, The	481
7	288	Scream	478
8	1	Toy Story	452
9	300	Air Force One	431
10	121	Independence Day (ID4)	429

2.) Películas con nota media ≥ 4.5 y al menos 100 valoraciones.

Script-1

```

SELECT p.movie_id, p.title,
       AVG(v.rating) AS nota_media,
       COUNT(*) AS num_votos
FROM movielens.votos v
JOIN movielens.peliculas p USING (movie_id)
GROUP BY p.movie_id, p.title
HAVING AVG(v.rating) >= 4.5 AND COUNT(*) >= 100
ORDER BY nota_media DESC, num_votos DESC;

```

películas 1

SELECT p.movie_id, p.title, AVG(v.rating) AS nota_media, COUNT(*) AS num_votos

Grilla	123 movie_id	A-Z title	123 nota_media	123 num_votos

3.) Usuarios que han dado más de 300 valoraciones y su nota media.

Script-1

```

SELECT u.user_id, u.age, u.gender, u.occupation,
       COUNT(*) AS num_valoraciones,
       ROUND(AVG(v.rating),2) AS nota_media
FROM movielens.votos v
JOIN movielens.usuarios u USING (user_id)
GROUP BY u.user_id, u.age, u.gender, u.occupation
HAVING COUNT(*) > 300
ORDER BY num_valoraciones DESC;

```

usuarios 1

SELECT u.user_id, u.age, u.gender, u.occupation, COUNT(*) AS num_valoraciones, ROUND(AVG(v.rating),2) AS nota_media

Grilla	123 user_id	123 age	A-Z gender	A-Z occupation	123 num_valoraciones	123 nota_media
1	405	22	F	healthcare	737	1,83
2	655	50	F	healthcare	685	2,91
3	13	47	M	educator	636	3,1
4	450	35	F	educator	540	3,86
5	276	21	M	student	518	3,47
6	416	20	F	student	493	3,85
7	537	36	M	engineer	490	2,87
8	303	19	M	student	484	3,37
9	234	60	M	retired	480	3,12
10	393	19	M	student	448	3,34

4.) Año con más películas votadas (por número total de votos).

The screenshot shows a SQL IDE with a query window and a results window. The query is:

```
SELECT YEAR(v.fecha) AS anio_voto, COUNT(*) AS num_votos
FROM movielens.votos v
GROUP BY YEAR(v.fecha)
ORDER BY num_votos DESC;
```

The results window shows a single row:

	123 anio_voto	123 num_votos
1	[NULL]	100.000

5.) Las 5 películas más "polarizadas" (mayor desviación estándar, con valoraciones muy extremas) con al menos 50 votos. (Investiga qué función de SQL de da la desviación estándar).

The screenshot shows a SQL IDE with a query window and a results window. The query is:

```
SELECT p.movie_id, p.title,
COUNT(*) AS num_votos,
ROUND(STDDEV_SAMP(v.rating),4) AS desv_estandar,
ROUND(AVG(v.rating),3) AS media
FROM movielens.votos v
JOIN movielens.peliculas p USING (movie_id)
GROUP BY p.movie_id, p.title
HAVING COUNT(*) >= 50
ORDER BY desv_estandar DESC
LIMIT 5;
```

The results window shows the following data:

	123 movie_id	A-Z title	123 num_votos	123 desv_estandar	123 media
1	1.065	Koyaanisqatsi	53	1,3675	3,491
2	898	Postman, The	58	1,3513	3,224
3	53	Natural Born Killers	128	1,3272	2,953
4	640	Cook the Thief His Wife & Her Lover, The	82	1,3238	3,024
5	219	Nightmare on Elm Street, A	111	1,3134	3,171

6.) Usuarios cuya nota media es menor que la nota media global.

Script-1

movielens

peliculas

usuarios

votos

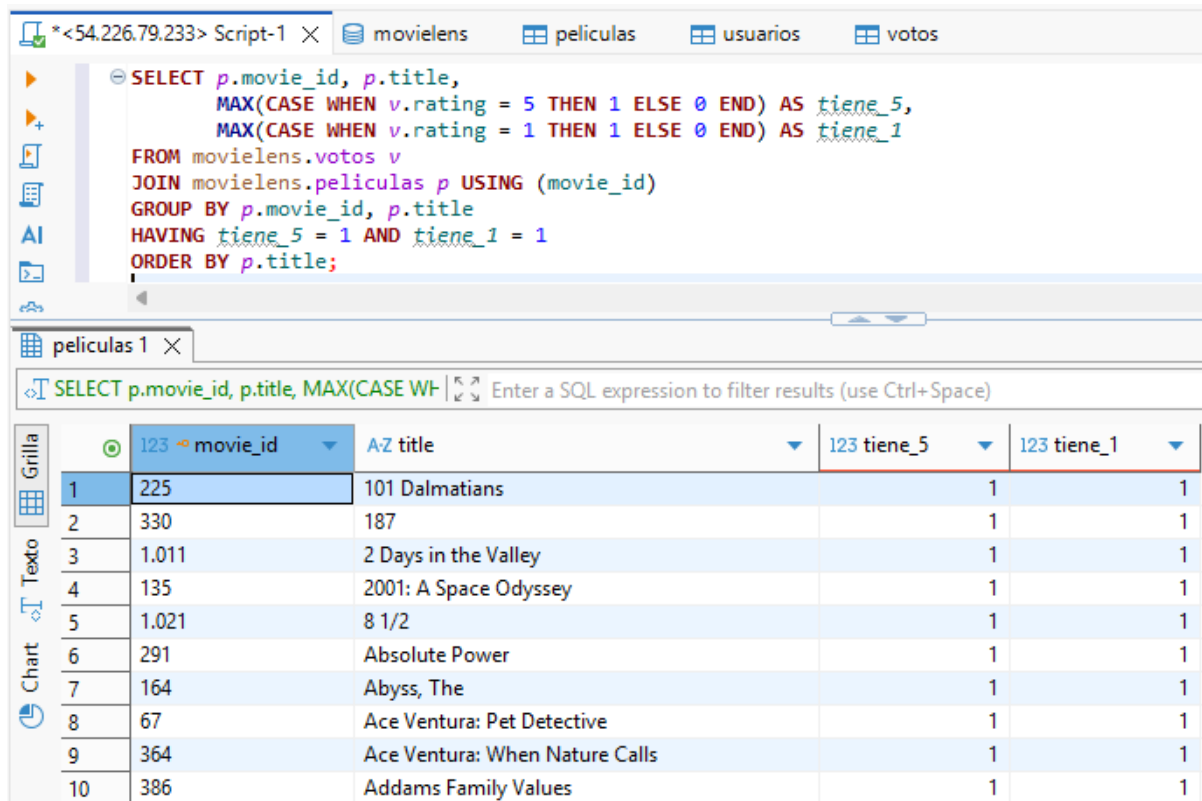
```
WITH global_avg AS (  
  SELECT AVG(rating) AS nota_global FROM movielens.votos  
)  
SELECT u.user_id, u.age, u.gender, u.occupation,  
       ROUND(AVG(v.rating),3) AS nota_media_usuario  
FROM movielens.votos v  
JOIN movielens.usuarios u USING (user_id)  
GROUP BY u.user_id, u.age, u.gender, u.occupation  
HAVING AVG(v.rating) < (SELECT nota_global FROM global_avg)  
ORDER BY nota_media_usuario ASC;
```

usuarios 1

WITH global_avg AS (SELECT AVG(rating) | Enter a SQL expression to filter results (use Ctrl+Space)

	123 user_id	123 age	AZ gender	AZ occupation	123 nota_media_usuario
1	181	26	M	executive	1,492
2	405	22	F	healthcare	1,834
3	445	21	M	writer	1,985
4	685	32	F	librarian	2,05
5	774	30	M	student	2,058
6	724	31	M	executive	2,165
7	206	14	F	student	2,172
8	865	25	M	artist	2,288
9	626	23	M	scientist	2,344
10	609	13	F	student	2,393

7.) Películas que han recibido al menos una valoración de 1 y una de 5 (las más divididas).



```

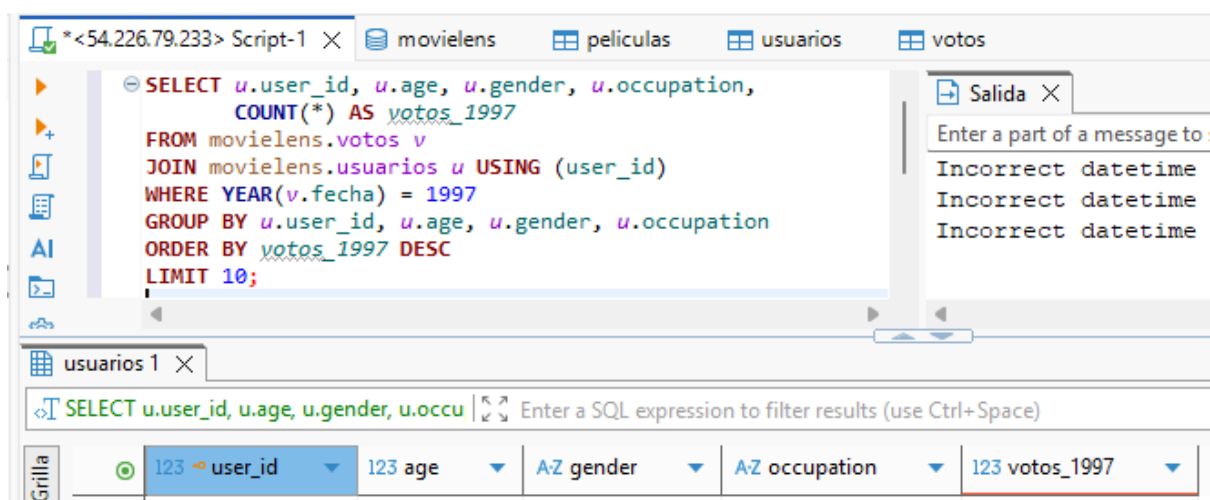
SELECT p.movie_id, p.title,
       MAX(CASE WHEN v.rating = 5 THEN 1 ELSE 0 END) AS tiene_5,
       MAX(CASE WHEN v.rating = 1 THEN 1 ELSE 0 END) AS tiene_1
FROM movielens.votos v
JOIN movielens.peliculas p USING (movie_id)
GROUP BY p.movie_id, p.title
HAVING tiene_5 = 1 AND tiene_1 = 1
ORDER BY p.title;

```

	123 movie_id	AZ title	123 tiene_5	123 tiene_1
1	225	101 Dalmatians	1	1
2	330	187	1	1
3	1.011	2 Days in the Valley	1	1
4	135	2001: A Space Odyssey	1	1
5	1.021	8 1/2	1	1
6	291	Absolute Power	1	1
7	164	Abyss, The	1	1
8	67	Ace Ventura: Pet Detective	1	1
9	364	Ace Ventura: When Nature Calls	1	1
10	386	Addams Family Values	1	1

8.) Top 10 usuarios más activos en 1997 (por número de valoraciones ese año).

Me sale un error ya que yo utilice un “bigint” para la fecha en vez de un “timestamp”



```

SELECT u.user_id, u.age, u.gender, u.occupation,
       COUNT(*) AS votos_1997
FROM movielens.votos v
JOIN movielens.usuarios u USING (user_id)
WHERE YEAR(v.fecha) = 1997
GROUP BY u.user_id, u.age, u.gender, u.occupation
ORDER BY votos_1997 DESC
LIMIT 10;

```

Salida X

Enter a part of a message to:

Incorrect datetime
Incorrect datetime
Incorrect datetime

	123 user_id	123 age	AZ gender	AZ occupation	123 votos_1997
--	-------------	---------	-----------	---------------	----------------

9.) Películas estrenadas después de 1995 con mejor nota media que "Toy Story (1995).

*<54.226.79.233> Script-1 X movielens peliculas usuarios votos

```
WITH toy AS (  
    SELECT AVG(v.rating) AS nota_toy  
    FROM movielens.votos v  
    JOIN movielens.peliculas p USING (movie_id)  
    WHERE p.title LIKE 'Toy Story%'  
)  
SELECT p.movie_id, p.title, p.anio,  
       ROUND(AVG(v.rating),3) AS nota_media, COUNT(*) AS num_votos  
FROM movielens.peliculas p  
JOIN movielens.votos v USING (movie_id)  
WHERE p.anio > 1995  
GROUP BY p.movie_id, p.title, p.anio  
HAVING AVG(v.rating) > (SELECT nota_toy FROM toy)  
ORDER BY nota_media DESC;
```

peliculas 1 X

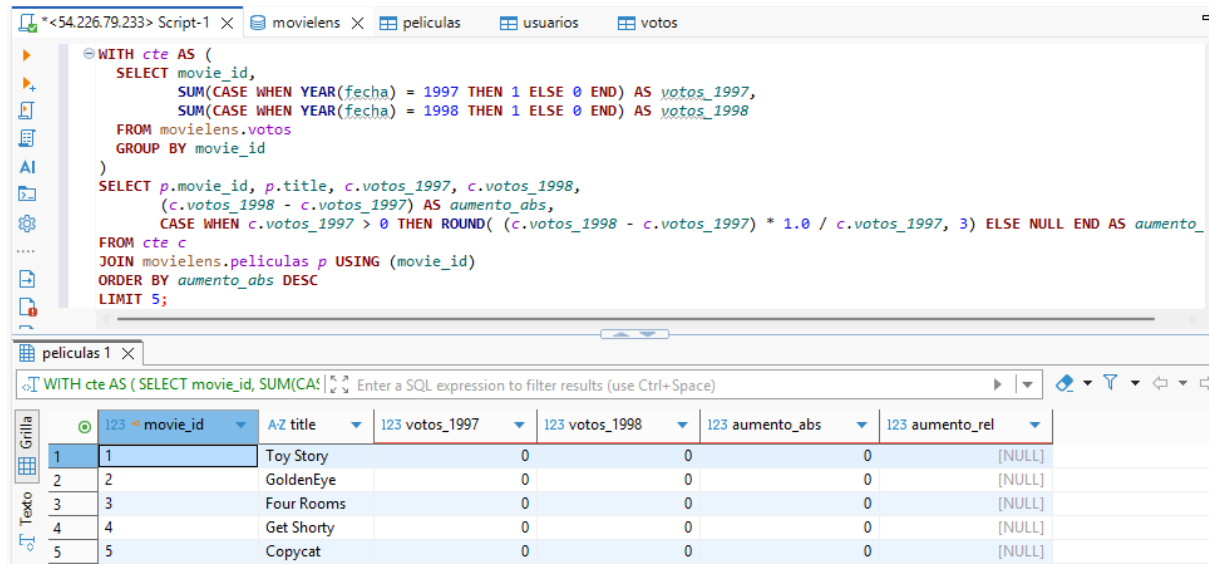
WITH toy AS (SELECT AVG(v.rating) AS n Enter a SQL expression to filter results (use Ctrl+Space)

	123 movie_id	AZ title	AZ anio	123 nota_media	123 num_votos
1	1.653	Entertaining Angels: The Dorothy Day Story	1996	5	1
2	1.500	Santa with Muscles	1996	5	2
3	1.293	Star Kid	1997	5	3
4	1.201	Marlene Dietrich: Shadow and Light	1996	5	1
5	1.189	Prefontaine	1997	5	3
6	1.642	Some Mother's Son	1996	4,5	2
7	1.594	Everest	1998	4,5	2
8	1.398	Anna	1996	4,5	2
9	114	Wallace & Gromit: The Best of Aardman Animation	1996	4,448	67
10	1.639	Bitter Sugar (Azucar Amarqo)	1996	4,333	3

10.) Usuarios que han valorado todas las películas estrenadas en 1993.

11.) Evolución mensual del número de valoraciones en 1998.

12.) Las 5 películas con mayor aumento de popularidad (comparar 1997 vs 1998).



```
WITH cte AS (
    SELECT movie_id,
           SUM(CASE WHEN YEAR(fecha) = 1997 THEN 1 ELSE 0 END) AS votos_1997,
           SUM(CASE WHEN YEAR(fecha) = 1998 THEN 1 ELSE 0 END) AS votos_1998
    FROM movielens.votos
    GROUP BY movie_id
)
SELECT p.movie_id, p.title, c.votos_1997, c.votos_1998,
       (c.votos_1998 - c.votos_1997) AS aumento_abs,
       CASE WHEN c.votos_1997 > 0 THEN ROUND( (c.votos_1998 - c.votos_1997) * 1.0 / c.votos_1997, 3) ELSE NULL END AS aumento_rel
FROM cte c
JOIN movielens.peliculas p USING (movie_id)
ORDER BY aumento_abs DESC
LIMIT 5;
```

	123 movie_id	AZ title	123 votos_1997	123 votos_1998	123 aumento_abs	123 aumento_rel
1	1	Toy Story	0	0	0	[NULL]
2	2	GoldenEye	0	0	0	[NULL]
3	3	Four Rooms	0	0	0	[NULL]
4	4	Get Shorty	0	0	0	[NULL]
5	5	Copycat	0	0	0	[NULL]

13.) Usuarios que han valorado más películas que la media de su género.

*<54.226.79.233> Script-1 X movielens películas usuarios votos

```
WITH per_user AS (
  SELECT user_id, gender, COUNT(DISTINCT movie_id) AS cnt_peliculas
  FROM movielens.votos v
  JOIN movielens.usuarios u USING (user_id)
  GROUP BY user_id, gender
),
avg_per_gender AS (
  SELECT gender, AVG(cnt_peliculas) AS avg_por_gender
  FROM per_user
  GROUP BY gender
)
SELECT p.user_id, u.age, p.gender, p.cnt_peliculas, ROUND(a.avg_por_gender,3) AS media_gender
FROM per_user p
JOIN avg_per_gender a USING (gender)
JOIN movielens.usuarios u ON u.user_id = p.user_id
WHERE p.cnt_peliculas > a.avg_por_gender
ORDER BY p.gender, p.cnt_peliculas DESC;
```

votos(+) 1 X

WITH per_user AS (SELECT user_id, gende | Enter a SQL expression to filter results (use Ctrl+Space)

	123 user_id	123 age	A-Z gender	123 cnt_peliculas	123 media_gender
1	405 F	22	F	737	94,286
2	655 F	50	F	685	94,286
3	450 F	35	F	540	94,286
4	416 F	20	F	493	94,286
5	417 F	27	F	365	94,286
6	796 F	32	F	358	94,286
7	269 F	31	F	323	94,286
8	642 F	18	F	318	94,286
9	151 F	38	F	307	94,286
10	457 F	33	F	277	94,286

14.) Películas que nadie ha valorado con 3 estrellas (solo 1,2,4,5).

Script-1

```

SELECT p.movie_id, p.title,
       COUNT(CASE WHEN v.rating = 3 THEN 1 END) AS num_3
FROM movielens.peliculas p
LEFT JOIN movielens.votos v USING (movie_id)
GROUP BY p.movie_id, p.title
HAVING SUM(CASE WHEN v.rating = 3 THEN 1 ELSE 0 END) = 0
ORDER BY p.title;

```

películas 1

SELECT p.movie_id, p.title, COUNT(CASE WHEN v.rating = 3 THEN 1 END) AS num_3

	123 movie_id	A-Z title	123 num_3
1	314	3 Ninjas: High Noon At Mega Mountain	0
2	1.664	8 Heads in a Duffel Bag	0
3	1.443	8 Seconds	0
4	1.536	Aiqing wansui	0
5	1.585	American Dream	0
6	437	Amityville 1992: It's About Time	0
7	438	Amityville 3-D	0
8	442	Amityville Curse, The	0
9	439	Amityville: A New Generation	0
10	858	Amityville: Dollhouse	0

15.) Ranking de días de la semana con más actividad (lunes, martes...).

Otra vez el error por el formato de la fecha

Script-1

```

SELECT DAYNAME(fecha) AS dia_semana, COUNT(*) AS num_votos
FROM movielens.votos
GROUP BY DAYNAME(fecha)
ORDER BY num_votos DESC;

```

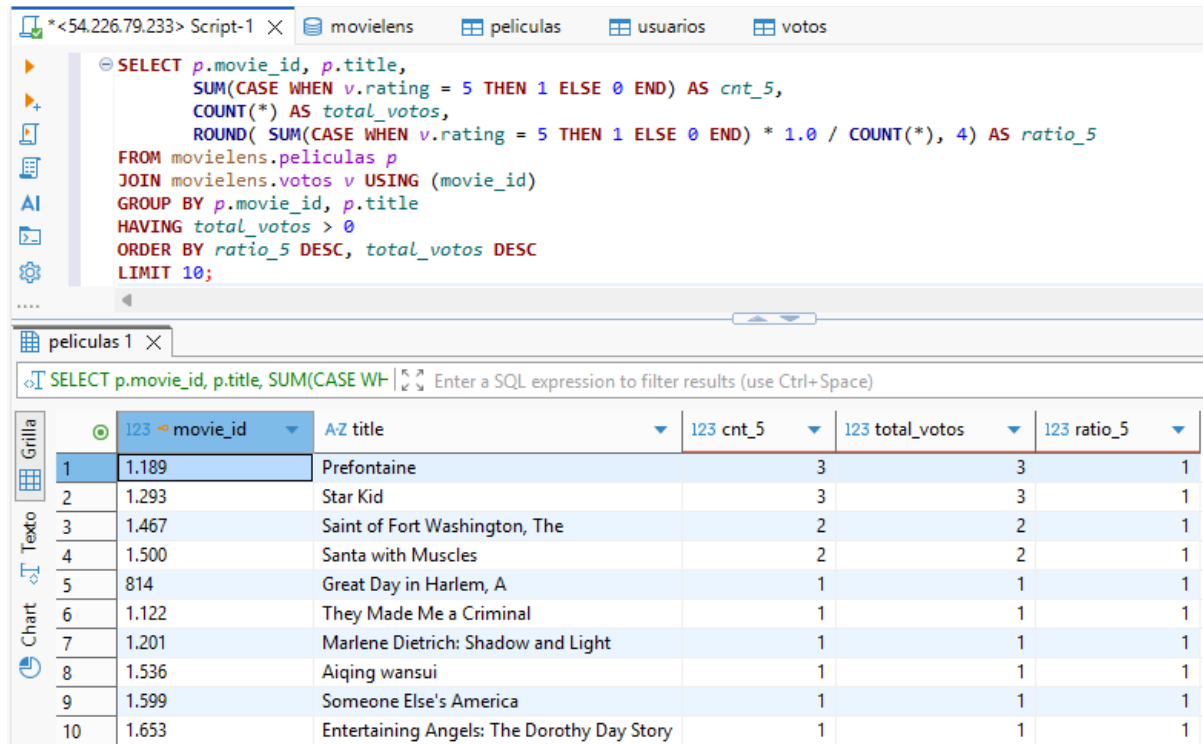
Resultados 1

SELECT DAYNAME(fecha) AS dia_semana, COUNT(*) AS num_votos

	A-Z dia_semana	123 num_votos
1	[NULL]	100.000

16.) Usuarios que han dado su primera y última valoración con diferencia > 6 meses.

17.) Las 10 películas con mayor ratio 5-estrellas / total valoraciones.



The screenshot shows a SQL IDE interface with a script editor and a results pane. The script editor contains a SQL query that calculates the ratio of 5-star ratings to total ratings for movies. The results pane shows the top 10 movies based on this ratio.

```
SELECT p.movie_id, p.title,
       SUM(CASE WHEN v.rating = 5 THEN 1 ELSE 0 END) AS cnt_5,
       COUNT(*) AS total_votos,
       ROUND( SUM(CASE WHEN v.rating = 5 THEN 1 ELSE 0 END) * 1.0 / COUNT(*), 4) AS ratio_5
FROM movielens.peliculas p
JOIN movielens.votos v USING (movie_id)
GROUP BY p.movie_id, p.title
HAVING total_votos > 0
ORDER BY ratio_5 DESC, total_votos DESC
LIMIT 10;
```

	123 movie_id	A-Z title	123 cnt_5	123 total_votos	123 ratio_5
1	1.189	Prefontaine	3	3	1
2	1.293	Star Kid	3	3	1
3	1.467	Saint of Fort Washington, The	2	2	1
4	1.500	Santa with Muscles	2	2	1
5	814	Great Day in Harlem, A	1	1	1
6	1.122	They Made Me a Criminal	1	1	1
7	1.201	Marlene Dietrich: Shadow and Light	1	1	1
8	1.536	Aiqing wansui	1	1	1
9	1.599	Someone Else's America	1	1	1
10	1.653	Entertaining Angels: The Dorothy Day Story	1	1	1

18.) UPDATE: Aumenta en 1 año la edad de todos los usuarios (simulación de paso del tiempo).

Para esto vamos a obtener primero la edad antes de modificarla para luego comparar

*<54.226.79.233> Script-1 X movielens peliculas usuarios votos

```
SELECT user_id, age FROM movielens.usuarios ORDER BY user_id LIMIT 10;
```

usuarios 1 X

SELECT user_id, age FROM movielens.usu Enter a SQL expression to filter results (use Ctrl+Space)

	123 user_id	123 age
1	1	24
2	2	53
3	3	23
4	4	24
5	5	33
6	6	42
7	7	57
8	8	36
9	9	29
10	10	53

Ahora vamos a aumentarles 1 año

*<54.226.79.233> Script-1 X movielens peliculas usuarios votos

```
UPDATE movielens.usuarios
SET age = age + 1;

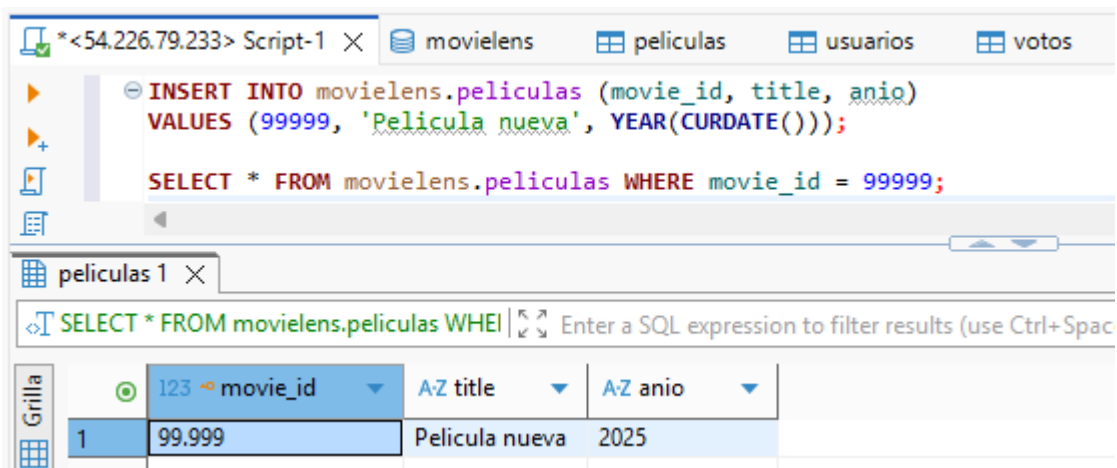
SELECT user_id, age FROM movielens.usuarios ORDER BY user_id LIMIT 10;
```

usuarios 1 X

SELECT user_id, age FROM movielens.usu Enter a SQL expression to filter results (use Ctrl+Space)

	123 user_id	123 age
1	1	25
2	2	54
3	3	24
4	4	25
5	5	34
6	6	43
7	7	58
8	8	37
9	9	30
10	10	54

19.) INSERT: Añade una nueva película ficticia estrenada hoy.



The screenshot shows a SQL client window with a script editor and a results pane. The script editor contains the following SQL code:

```
INSERT INTO movielens.peliculas (movie_id, title, anio)
VALUES (99999, 'Película nueva', YEAR(CURDATE()));

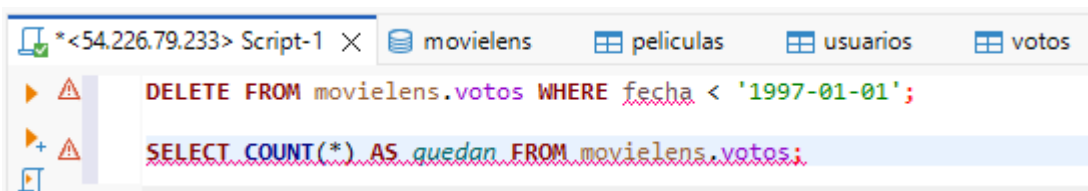
SELECT * FROM movielens.peliculas WHERE movie_id = 99999;
```

The results pane shows a table with the following data:

Grilla	123 movie_id	A-Z title	A-Z anio
1	99.999	Película nueva	2025

20.) DELETE: Elimina todas las valoraciones anteriores a 1997.

Este ejercicio me sale otra vez el error del formato de la “fecha” ya que yo no lo asigne con un “timestamp”, pero en case de que funcionara, el comando sería el siguiente



The screenshot shows a SQL client window with a script editor. The script editor contains the following SQL code:

```
DELETE FROM movielens.votos WHERE fecha < '1997-01-01';

SELECT COUNT(*) AS quedan FROM movielens.votos;
```