



CURSO DE ESPECIALIZACIÓN EN INTELIGENCIA ARTIFICIAL Y BIGDATA

BIGDATA APLICADO

TAREA EVALUABLE 3.1

Autor: Carlos Sánchez Recio.
FECHA

Índice

Apartado 1		1
1	Número de goles que ha marcado Lionel Messi (sin contar autogoles). . . .	3
2	Listado de los 5 partidos más recientes jugados por la selección española. .	3
3	Número de goles que ha marcado a España en toda su historia. Esta información debe sacarse de resultas, ya que <i>goalscorers</i> no contiene todos los goles.	3
4	Listado de los 5 máximos goleadores con la selección española (sin contar autogoles).	4
5	Listado de los jugadores españoles que han marcado algún gol de penalti en alguna Eurocopa (UEFA Euro), ordenados alfabéticamente.	4
6	Listado de los 5 máximos goleadores de las fases finales de los mundiales (FIFA World Cup) (sin contar autogoles).	5
Apartado 2		6
1	¿Cuál de las plataformas tiene más películas en su colección? Muestra la plataforma y el número de películas.	8
2	¿Cuáles son las 5 series con mejor valoración en IMDB (<i>imdbAverageRating</i>)? Para cada serie, muestra el título, la valoración y la plataforma en la que se encuentra.	9
3	¿Cuál es el total de votos en IMDB (<i>imdbNumVotes</i>) de todas las series del género de ciencia ficción en cada una de las plataformas? Para cada plataforma, muestra la plataforma y número de votos, ordenados de mayor a menor número de votos.	9
4	¿Cuáles son los 5 años en los que se han lanzado más películas? Para cada año, muestra el año y número de películas, ordenados de mayor a menor número de películas.	10

Apartado 1

Vamos a trabajar con el dataset de resultados de todos los partidos de fútbol disputados entre selecciones nacionales desde 1872 hasta la actualidad, que podéis encontrar en Kaggle, donde podrás encontrar todos los detalles.

De los tres ficheros de los que consta el dataset, nos interesan sólo dos:

- `results.csv`, que contiene la información de todos los partidos disputados, incluyendo equipos, marcador, campeonato y sede.
- `goalscorers.csv`, que contiene la información de todos los goles marcados en estos partidos. Para cada gol, se indica el partido (fecha y equipos), el equipo y jugador que marca el gol, el minuto y dos flags que indican si ha estado en propia portería o de penalti.

Una vez obtenidos los archivos necesarios (en mi caso desde los repositorios `results.csv` y `goalscorers.csv`) se pueden subir o no al sistema de archivos distribuidos de Hadoop (como se muestra en la imagen) o sencillamente dejarlos en el sistema local.

```

carlos ~ cloudera@quickstart:~/block3$ ssh -v -oHostKeyAlgorithms+=ssh-rsa cloudera@192.168.1.117 - 114x50
[cloudera@quickstart block3]$ wget https://raw.githubusercontent.com/tnavarrete-iedib/bigdata-24-25/refs/heads/main/results.csv
--2024-12-05 09:44:13-- https://raw.githubusercontent.com/tnavarrete-iedib/bigdata-24-25/refs/heads/main/results.csv
Resolving raw.githubusercontent.com... 185.199.108.133, 185.199.111.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com[185.199.108.133]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3655083 (3.5M) [text/plain]
Saving to: 'results.csv'

100%[=====] 3,655,083 13.0M/s in 0.3s

2024-12-05 09:44:14 (13.0 MB/s) - 'results.csv' saved [3655083/3655083]

[cloudera@quickstart block3]$ wget https://raw.githubusercontent.com/tnavarrete-iedib/bigdata-24-25/refs/heads/main/goalscorers.csv
--2024-12-05 09:44:26-- https://raw.githubusercontent.com/tnavarrete-iedib/bigdata-24-25/refs/heads/main/goalscorers.csv
Resolving raw.githubusercontent.com... 185.199.110.133, 185.199.111.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com[185.199.110.133]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3077437 (2.9M) [text/plain]
Saving to: 'goalscorers.csv'

100%[=====] 3,077,437 10.7M/s in 0.3s

2024-12-05 09:44:27 (10.7 MB/s) - 'goalscorers.csv' saved [3077437/3077437]

[cloudera@quickstart block3]$ hdfs
goalscorers.csv results.csv
[cloudera@quickstart block3]$ hdfs dfs -mkdir soccer
[cloudera@quickstart block3]$ hdfs dfs -ls
Found 4 items
drwxr-xr-x - cloudera cloudera 0 2024-11-21 08:47 prime
drwxr-xr-x - cloudera cloudera 0 2024-11-21 06:43 quijote
drwxr-xr-x - cloudera cloudera 0 2024-12-05 09:50 soccer
drwxr-xr-x - cloudera cloudera 0 2024-11-21 16:10 top_10_series_since_2020
[cloudera@quickstart block3]$ hdfs dfs -put results.csv soccer
[cloudera@quickstart block3]$ hdfs dfs -put goalscorers.csv soccer
[cloudera@quickstart block3]$ hdfs dfs -ls
Found 4 items
drwxr-xr-x - cloudera cloudera 0 2024-11-21 08:47 prime
drwxr-xr-x - cloudera cloudera 0 2024-11-21 06:43 quijote
drwxr-xr-x - cloudera cloudera 0 2024-12-05 09:51 soccer
drwxr-xr-x - cloudera cloudera 0 2024-11-21 16:10 top_10_series_since_2020
[cloudera@quickstart block3]$ hdfs dfs -ls soccer
Found 2 items
-rw-r--r-- 1 cloudera cloudera 3077437 2024-12-05 09:51 soccer/goalscorers.csv
-rw-r--r-- 1 cloudera cloudera 3655083 2024-12-05 09:50 soccer/results.csv
[cloudera@quickstart block3]$

```

Figura 1. 1: Obtención de los archivos CSV necesarios y subir a HDFS.

Antes de cargar los datos en Hive, se necesitan crear tanto la base de datos como las tablas que contendrán los datos de los archivos. Para ello se ejecutan las sentencias siguientes¹:

```

1 CREATE DATABASE soccer;
2 USE soccer;
3
4 -- Create table for results
5 -- I'll avoid the NOT NULL but should be used

```

¹Se puede ver la ejecución de éstas en la *Figura 2*

```

6 CREATE TABLE soccer.results (
7     date DATE,
8     home_team STRING,
9     away_team STRING,
10    home_score INT,
11    away_score INT,
12    tournament STRING,
13    city STRING,
14    country STRING,
15    neutral BOOLEAN
16 )
17 ROW FORMAT DELIMITED
18 FIELDS TERMINATED BY '\t'
19 TBLPROPERTIES ("skip.header.line.count"="1");
20
21 -- Create table for goals
22 -- Again, avoiding NOT NULL but should be there
23 CREATE TABLE soccer.goalscorers (
24     date STRING,
25     home_team STRING,
26     away_team STRING,
27     team STRING,
28     scorer STRING,
29     minute INT,
30     own_goal BOOLEAN,
31     penalty BOOLEAN
32 )
33 ROW FORMAT DELIMITED
34 FIELDS TERMINATED BY '\t'
35 TBLPROPERTIES ("skip.header.line.count"="1");
36
37 -- Load data into tables
38 -- I did use the LOCAL way because I wasn't able to do it with HDFS
39 LOAD DATA LOCAL INPATH "/home/cloudera/block3/soccer/results.csv"
40 INTO TABLE soccer.results;
41 LOAD DATA LOCAL INPATH "/home/cloudera/block3/soccer/goalscorers.csv"
42 INTO TABLE soccer.goalscorers;

```

Tras este paso se tienen que cargar los datos en Hive, en mi caso mediante la interfaz web que proporciona el servicio:

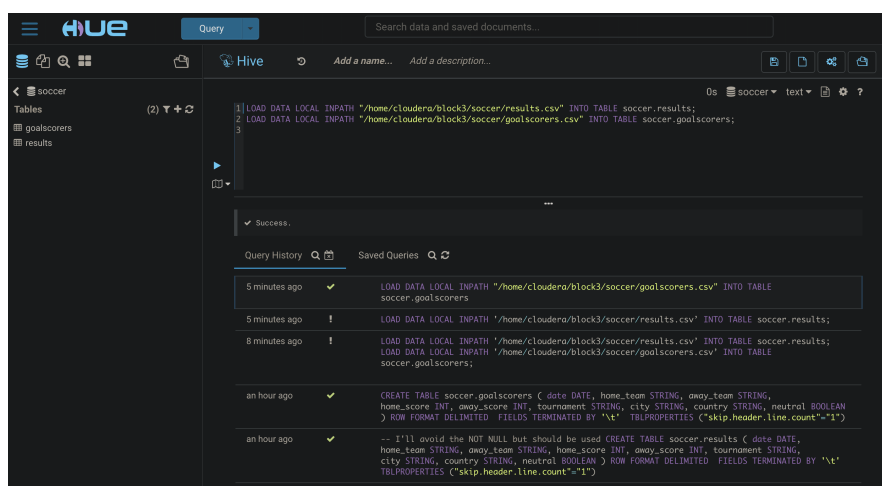


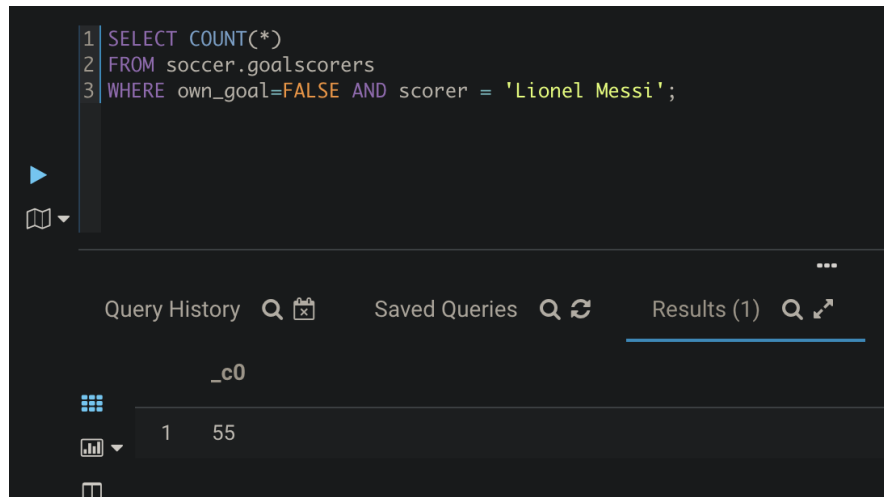
Figura 1. 2: Cargar los datos en Hive.

Como se ve en esta última imagen al final he usado el método con archivos locales ya que no pude conseguir hacerlo desde HDFS a pesar de que estuvieran subidos al servicio distribuido (no afecta en el resultado final).

Como analistas de datos nos han pedido una serie de preguntas que debemos responder utilizando Apache Hive. Son estas:

1 Número de goles que ha marcado Lionel Messi (sin contar autogoles).

```
1 SELECT COUNT(*)
2 FROM soccer.goalscorers
3 WHERE own_goal=FALSE AND scorer = 'Lionel Messi';
```



2 Listado de los 5 partidos más recientes jugados por la selección española.

```
1 SELECT *
2 FROM soccer.results
3 WHERE home_team = 'Spain' OR away_team = 'Spain'
4 ORDER BY date DESC
5 LIMIT 5;
```

The screenshot shows a Hive query execution interface. The query is: `SELECT * FROM soccer.results WHERE home_team = 'Spain' OR away_team = 'Spain' ORDER BY date DESC LIMIT 5;`. The interface includes a 'Query History' tab, a 'Saved Queries' tab, and a 'Results (5)' tab. The 'Results (5)' tab is active, showing a table with 5 rows of match data.

	results.date	results.home_team	results.away_team	results.home_score	results.away_score	results.tournament
1	2024-11-18	Spain	Switzerland	3	2	UEFA Nations League
2	2024-11-15	Denmark	Spain	1	2	UEFA Nations League
3	2024-10-15	Spain	Serbia	3	0	UEFA Nations League
4	2024-10-12	Spain	Denmark	1	0	UEFA Nations League
5	2024-09-08	Switzerland	Spain	1	4	UEFA Nations League

3 Número de goles que ha marcado a España en toda su historia. Esta información debe sacarse de *results*, ya que *goalscorers* no contiene todos los goles.

```
1 SELECT
2     SUM(CASE WHEN home_team = 'Spain' THEN home_score ELSE 0 END +
3         CASE WHEN away_team = 'Spain' THEN away_score ELSE 0 END)
4 FROM
5     soccer.results;
```

```

1 SELECT
2     SUM(CASE WHEN home_team = 'Spain' THEN home_score ELSE 0 END +
3         CASE WHEN away_team = 'Spain' THEN away_score ELSE 0 END)
4 FROM
5     soccer.results;

```

Query History Saved Queries Results (1)

	_c0
1	1553

- 4 Listado de los 5 máximos goleadores con la selección española (sin contar auto-goles).

```

1 SELECT scorer, COUNT(*) AS num_goals
2 FROM soccer.goalscorers
3 WHERE team = 'Spain' AND own_goal = FALSE
4 GROUP BY scorer
5 ORDER BY num_goals DESC
6 LIMIT 5;

```

```

1 SELECT
2     scorer,
3     COUNT(*) AS num_goals
4 FROM
5     soccer.goalscorers
6 WHERE
7     team = 'Spain' AND own_goal = FALSE
8 GROUP BY
9     scorer
10 ORDER BY
11     num_goals DESC
12 LIMIT 5;

```

Query History Saved Queries Results (5)

	scorer	num_goals
1	David Villa	41
2	Raúl	32
3	Álvaro Morata	29
4	Fernando Torres	28
5	Fernando Hierro	25

- 5 Listado de los jugadores españoles que han marcado algún gol de penalti en alguna Eurocopa (UEFA Euro), ordenados alfabéticamente.

```

1 SELECT g.scorer
2 FROM soccer.goalscorers AS g
3 INNER JOIN soccer.results AS r ON r.date = g.date
4 WHERE g.penalty=TRUE AND g.team='Spain' AND r.tournament = 'UEFA Euro'
5 GROUP BY g.scorer
6 ORDER BY g.scorer DESC;

```

```
1 SELECT g.scorer
2 FROM soccer.goalscorers AS g
3 INNER JOIN soccer.results AS r ON r.date = g.date
4 WHERE g.penalty=TRUE AND g.team='Spain' AND r.tournament = 'UEFA Euro'
5 GROUP BY g.scorer
6 ORDER BY g.scorer DESC;
```

Query History Saved Queries Results (4)

	g.scorer
1	Daniel Ruiz
2	Francisco José Carrasco
3	Gaizka Mendieta
4	Xabi Alonso

6 Listado de los 5 máximos goleadores de las fases finales de los mundiales (FIFA World Cup) (sin contar autogoles).

```
1 SELECT g.scorer, COUNT(*) AS goals
2 FROM soccer.goalscorers AS g
3 INNER JOIN soccer.results AS r ON r.date = g.date
4 WHERE g.own_goal=FALSE AND r.tournament = 'FIFA World Cup'
5 GROUP BY g.scorer
6 ORDER BY goals DESC
7 LIMIT 5;
```

```
1 SELECT g.scorer, COUNT(*) AS goals
2 FROM soccer.goalscorers AS g
3 INNER JOIN soccer.results AS r ON r.date = g.date
4 WHERE g.own_goal=FALSE AND r.tournament = 'FIFA World Cup'
5 GROUP BY g.scorer
6 ORDER BY goals DESC
7 LIMIT 5;
```

Query History Saved Queries Results (5)

	g.scorer	goals
1	Just Fontaine	60
2	Gerd Müller	49
3	Helmut Rahn	44
4	Uwe Seeler	44
5	Miroslav Klose	43

Apartado 2

En la tarea de la entrega 2 trabajamos con el dataset de películas y series de la plataforma Amazon Prime, publicado en Kaggle por OctopusTeam. Además de Amazon Prime, OctopusTeam también publica el dataset de las otras principales plataformas de streaming:

- [Netflix](#)
- [Apple TV+](#)
- [Amazon Prime](#)
- [Hulu](#)
- [HBO Max](#)

Todos los datasets tienen la misma estructura.

También puedes encontrar los archivos en GitHub del curso (actualizados hasta el 22/11/2024), donde ya se han empleado tabuladores como separadores de campo: [Netflix](#), [Apple TV+](#), [Amazon Prime](#), [Hulu](#) y [HBO Max](#).

Tienes que descargar el archivo de cada plataforma e importar los datos en una tabla Hive, utilizando 5 particiones estáticas, una para cada plataforma.

Sólo deben tenerse en cuenta aquellas series o películas que están registradas en IMDB (tienen un imdbId). Puedes quitar las otras filas directamente de los archivos de datos, antes de realizar la carga.

Si una serie o película está disponible en varias plataformas, podemos considerarlas como series o películas distintas.

Recuerda que el campo type nos indica si se trata de una película (movie) o una serie (tv).

Para el formateo de los datos, he utilizado conjuntamente los enlaces proporcionados en el enunciado al repositorio de Toni, además de un pequeño script en un proyecto con **poetry** ([enlace al repositorio](#)) en mi repositorio personal para el curso, en el que formateo los archivos y elimino las finas con valores NaN, manteniendo el formato interesado para esta tarea (usando tabulaciones como separador de columnas). Desde el shell del servidor, obtengo estos archivos mediante el comando **wget** y los enlaces a mi repositorio.

```
[cloudera@quickstart ~]$  
[cloudera@quickstart ~]$  
[cloudera@quickstart ~]$ tree ./block3/streaming/  
./block3/streaming/  
|-- amazon.csv  
|-- apple.csv  
|-- hbo.csv  
|-- hulu.csv  
`-- netflix.csv  
  
0 directories, 5 files  
[cloudera@quickstart ~]$
```

Figura 2. 3: Archivos para la segunda parte de la actividad.

Una vez los datos están en el sistema de archivos local, se repite el mismo proceso que en el apartado anterior para la creación de estructuras de tablas, añadiendo el particionamiento con la instrucción `PARTITIONED` para generar una partición por cada plataforma:

```

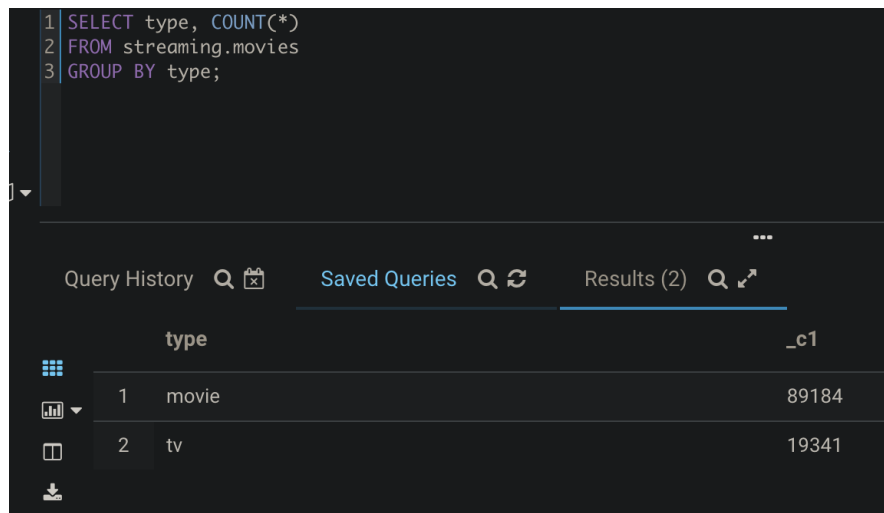
1 CREATE DATABASE streaming;
2 USE streaming;
3
4 -- Create the table with partitioning
5 CREATE TABLE streaming.movies (
6     title STRING,
7     type STRING,
8     genres STRING,
9     releaseYear FLOAT,
10    imdbId STRING,
11    imdbAverageRating FLOAT,
12    imdbNumVotes INT,
13    availableCountries STRING
14 )
15 PARTITIONED BY(platform STRING)
16 ROW FORMAT DELIMITED
17 FIELDS TERMINATED BY '\t'
18 TBLPROPERTIES ("skip.header.line.count"="1");
19
20 -- Load the data
21 LOAD DATA LOCAL INPATH '/home/cloudera/block3/streaming/amazon.csv'
22 INTO TABLE streaming.movies
23 PARTITION(platform = 'amazon_prime');
24 LOAD DATA LOCAL INPATH '/home/cloudera/block3/streaming/apple.csv'
25 INTO TABLE streaming.movies
26 PARTITION(platform = 'apple_tv');
27 LOAD DATA LOCAL INPATH '/home/cloudera/block3/streaming/hbo.csv'
28 INTO TABLE streaming.movies
29 PARTITION(platform = 'hbo_max');
30 LOAD DATA LOCAL INPATH '/home/cloudera/block3/streaming/hulu.csv'
31 INTO TABLE streaming.movies
32 PARTITION(platform = 'hulu');
33 LOAD DATA LOCAL INPATH '/home/cloudera/block3/streaming/netflix.csv'
34 INTO TABLE streaming.movies
35 PARTITION(platform = 'netflix');

```

a minute ago	✓	LOAD DATA LOCAL INPATH '/home/cloudera/block3/streaming/netflix.csv' INTO TABLE streaming.movies PARTITION(platform = 'netflix')
2 minutes ago	✓	LOAD DATA LOCAL INPATH '/home/cloudera/block3/streaming/hulu.csv' INTO TABLE streaming.movies PARTITION(platform = 'hulu')
4 minutes ago	✓	LOAD DATA LOCAL INPATH '/home/cloudera/block3/streaming/hbo.csv' INTO TABLE streaming.movies PARTITION(platform = 'hbo_max')
4 minutes ago	✓	LOAD DATA LOCAL INPATH '/home/cloudera/block3/streaming/apple.csv' INTO TABLE streaming.movies PARTITION(platform = 'apple_tv')
6 minutes ago	✓	LOAD DATA LOCAL INPATH '/home/cloudera/block3/streaming/amazon.csv' INTO TABLE streaming.movies PARTITION(platform = 'amazon_prime')
9 minutes ago	✓	CREATE TABLE streaming.movies (title STRING, type STRING, genres STRING, releaseYear FLOAT, imdbId STRING, imdbAverageRating FLOAT, imdbNumVotes INT, availableCountries STRING) PARTITIONED BY(platform STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' TBLPROPERTIES ("skip.header.line.count"="1")
an hour ago	✓	USE streaming
an hour ago	✓	CREATE DATABASE streaming

Antes de realizar las consultas que se piden para esta parte de la tarea, ejecuté la siguiente consulta para obtener los posibles tipos para el campo 'type' ya que será necesario para las consultas siguientes.

```
1 SELECT type, COUNT(*)
2 FROM streaming.movies
3 GROUP BY type;
```



The screenshot shows a SQL query editor with the following query:

```
1 SELECT type, COUNT(*)
2 FROM streaming.movies
3 GROUP BY type;
```

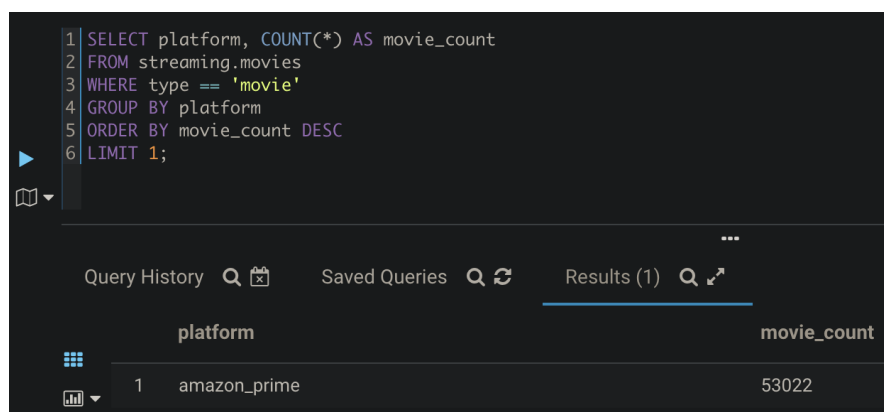
Below the query editor, the results are displayed in a table with two columns: 'type' and '_c1'.

type	_c1
1 movie	89184
2 tv	19341

Tienes que responder a las siguientes consultas:

- 1 ¿Cuál de las plataformas tiene más películas en su colección? Muestra la plataforma y el número de películas.

```
1 SELECT platform, COUNT(*) AS movie_count
2 FROM streaming.movies
3 WHERE type == 'movie'
4 GROUP BY platform
5 ORDER BY movie_count DESC
6 LIMIT 1;
```



The screenshot shows a SQL query editor with the following query:

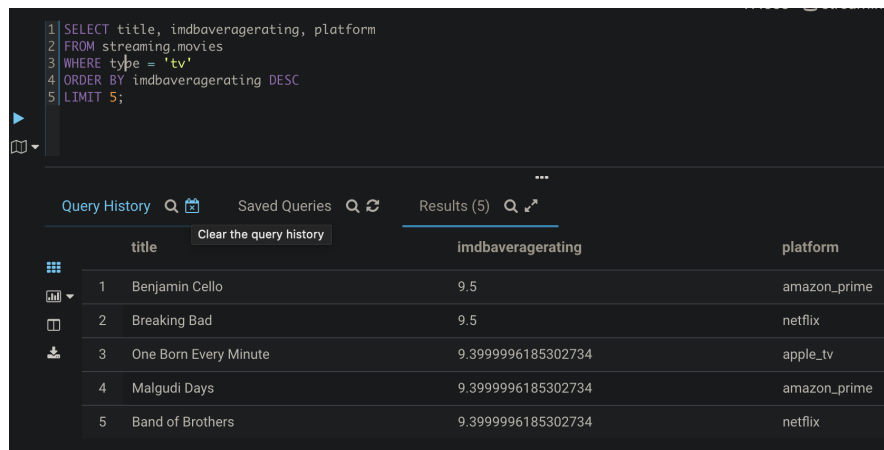
```
1 SELECT platform, COUNT(*) AS movie_count
2 FROM streaming.movies
3 WHERE type == 'movie'
4 GROUP BY platform
5 ORDER BY movie_count DESC
6 LIMIT 1;
```

Below the query editor, the results are displayed in a table with two columns: 'platform' and 'movie_count'.

platform	movie_count
1 amazon_prime	53022

- 2 ¿Cuáles son las 5 series con mejor valoración en IMDB (imdbAverageRating)? Para cada serie, muestra el título, la valoración y la plataforma en la que se encuentra.

```
1 SELECT title, imdbaverageating, platform
2 FROM streaming.movies
3 WHERE type = 'tv'
4 ORDER BY imdbaverageating DESC
5 LIMIT 5;
```



The screenshot shows a SQL query editor with the following query:

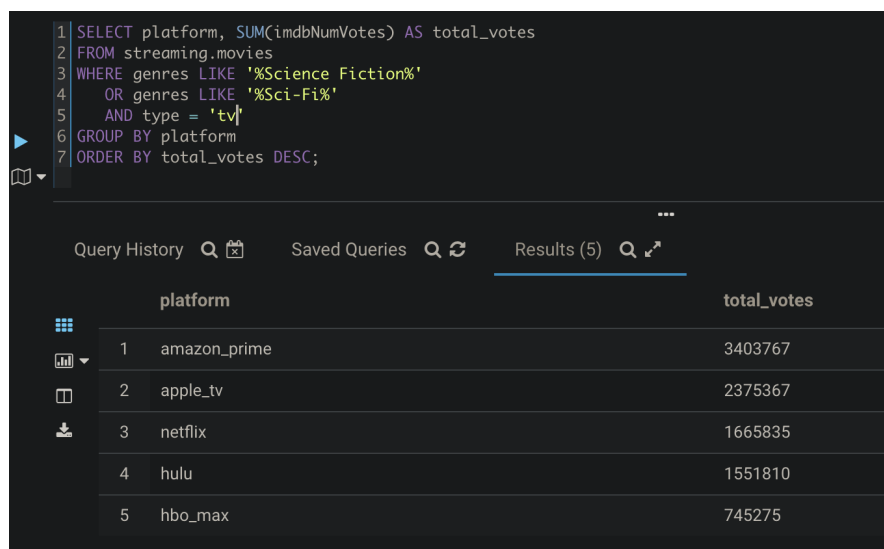
```
1 SELECT title, imdbaverageating, platform
2 FROM streaming.movies
3 WHERE type = 'tv'
4 ORDER BY imdbaverageating DESC
5 LIMIT 5;
```

The results are displayed in a table with 5 rows and 3 columns: title, imdbaverageating, and platform.

	title	imdbaverageating	platform
1	Benjamin Cello	9.5	amazon_prime
2	Breaking Bad	9.5	netflix
3	One Born Every Minute	9.3999996185302734	apple_tv
4	Malgudi Days	9.3999996185302734	amazon_prime
5	Band of Brothers	9.3999996185302734	netflix

- 3 ¿Cuál es el total de votos en IMDB (imdbNumVotes) de todas las series del género de ciencia ficción en cada una de las plataformas? Para cada plataforma, muestra la plataforma y número de votos, ordenados de mayor a menor número de votos.

```
1 SELECT platform, SUM(imdbNumVotes) AS total_votes
2 FROM streaming.movies
3 WHERE genres LIKE '%Science Fiction%'
4 OR genres LIKE '%Sci-Fi%'
5 AND type = 'tv'
6 GROUP BY platform
7 ORDER BY total_votes DESC;
```



The screenshot shows a SQL query editor with the following query:

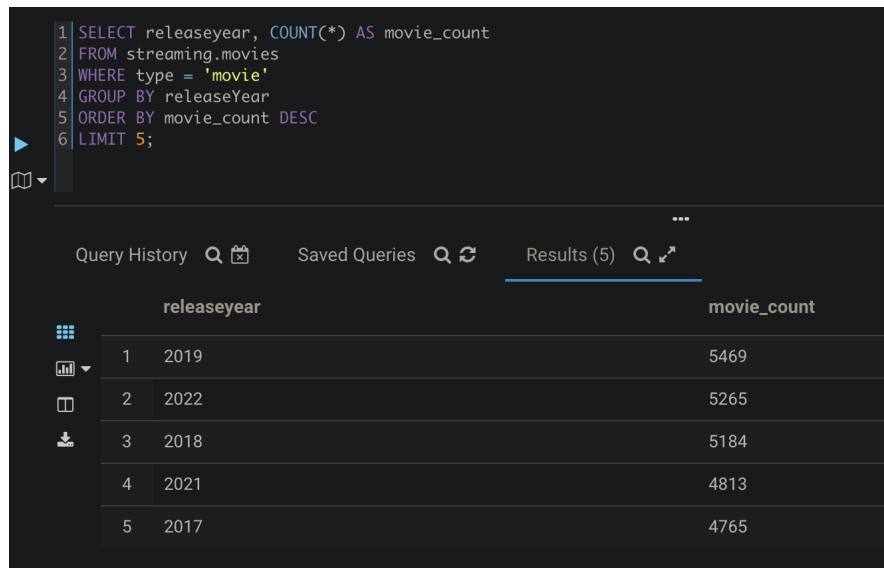
```
1 SELECT platform, SUM(imdbNumVotes) AS total_votes
2 FROM streaming.movies
3 WHERE genres LIKE '%Science Fiction%'
4 OR genres LIKE '%Sci-Fi%'
5 AND type = 'tv'
6 GROUP BY platform
7 ORDER BY total_votes DESC;
```

The results are displayed in a table with 5 rows and 2 columns: platform and total_votes.

	platform	total_votes
1	amazon_prime	3403767
2	apple_tv	2375367
3	netflix	1665835
4	hulu	1551810
5	hbo_max	745275

- 4 ¿Cuáles son los 5 años en los que se han lanzado más películas? Para cada año, muestra el año y número de películas, ordenados de mayor a menor número de películas.

```
1 SELECT releaseyear, COUNT(*) AS movie_count
2 FROM streaming.movies
3 WHERE type = 'movie'
4 GROUP BY releaseYear
5 ORDER BY movie_count DESC
6 LIMIT 5;
```



The screenshot shows a SQL query editor with a dark theme. The query is as follows:

```
1 SELECT releaseyear, COUNT(*) AS movie_count
2 FROM streaming.movies
3 WHERE type = 'movie'
4 GROUP BY releaseYear
5 ORDER BY movie_count DESC
6 LIMIT 5;
```

Below the query editor, there is a tab labeled "Results (5)". The results are displayed in a table with two columns: "releaseyear" and "movie_count". The table contains five rows of data, ordered by the number of movies in descending order.

	releaseyear	movie_count
1	2019	5469
2	2022	5265
3	2018	5184
4	2021	4813
5	2017	4765