

Tarea evaluable

3.1

Big data aplicado



Índice

APARTADO 1	2
1 - Importación de datos:	2
2 - Querys:	5
2.1 Número de goles que ha marcado Lionel Messi (sin contar autogoles).	5
2.2 Listado de los 5 partidos más recientes que ha jugado la selección española.	6
2.3 Número de goles que ha marcado España en toda su historia. Esta información debe extraerse de results, ya que goalscorers no contiene todos los goles.	7
2.4 Listado de los 5 máximos goleadores con la selección española (sin contar autogoles).	8
2.5 Listado de los jugadores españoles que han marcado algún gol de penalti en alguna Eurocopa (UEFA Euro), ordenados alfabéticamente.	9
2.6 Listado de los 5 máximos goleadores de las fases finales de los Mundiales (FIFA World Cup) (sin contar autogoles).	10
APARTADO 2	11
1 - Importación de datos:	11
2 - Querys:	15
2.1 ¿Cuál de las plataformas tiene más películas en su colección? Muestra la plataforma y el número de películas.	15
2.2 ¿Cuál de las plataformas tiene más películas en su colección? Muestra la plataforma y el número de películas.	16
2.3 ¿Cuál de las plataformas tiene más películas en su colección? Muestra la plataforma y el número de películas.	17
2.4 ¿Cuál de las plataformas tiene más películas en su colección? Muestra la plataforma y el número de películas.	18

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 se puede encontrar en Kaggle, donde podrás encontrar todos los detalles. De los tres archivos que componen el dataset, nos interesan sólo dos:

- **results.csv**, que contiene la información de todos los partidos disputados, incluyendo los equipos, el marcador, el campeonato y la 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 sido en propia puerta o de penalti.

1 - Importación de datos:

A través del comando `wget` he descargado los datos con el link ofrecido en los apuntes.

```
cloudera@quickstart.cloudera ~/tasca_3/apartado_1 (2.112s)
wget https://raw.githubusercontent.com/tnavarrete-iedib/bigdata-24-25/refs/heads/main/results.csv
--2024-12-19 13:50:04-- https://raw.githubusercontent.com/tnavarrete-iedib/bigdata-24-25/refs/heads/main/results.csv
Resolving raw.githubusercontent.com... 185.199.110.133, 185.199.109.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3655083 (3.5M) [text/plain]
Saving to: 'results.csv'

100%[=====] 3,655,083 2.14M/s in 1.6s

2024-12-19 13:50:05 (2.14 MB/s) - 'results.csv' saved [3655083/3655083]

cloudera@quickstart.cloudera ~/tasca_3/apartado_1 (1.622s)
wget https://raw.githubusercontent.com/tnavarrete-iedib/bigdata-24-25/refs/heads/main/goalscorers.csv
--2024-12-19 13:50:11-- https://raw.githubusercontent.com/tnavarrete-iedib/bigdata-24-25/refs/heads/main/goalscorers.csv
Resolving raw.githubusercontent.com... 185.199.109.133, 185.199.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3077437 (2.9M) [text/plain]
Saving to: 'goalscorers.csv'

100%[=====] 3,077,437 2.46M/s in 1.2s

2024-12-19 13:50:13 (2.46 MB/s) - 'goalscorers.csv' saved [3077437/3077437]

cloudera@quickstart.cloudera ~/tasca_3/apartado_1 (0.129s)
ls
goalscorers.csv  results.csv

cloudera@quickstart.cloudera ~/tasca_3/apartado_1
```

Después de importar los datos creamos la BBDD y las tablas que contendrán los archivos. El siguiente paso ya será cargar los datos en la BBDD de Hive.

The screenshot shows the Hive web interface. At the top, there's a search bar and a toolbar with icons for saving, refreshing, and settings. The main area displays a query with two lines: `1 CREATE DATABASE soccer;` and `2 USE soccer;`. Below the query, a status bar indicates the execution took 8.49s on the 'default' database in 'text' format. A green checkmark and the word 'Success.' confirm the query was executed successfully. At the bottom, the 'Query History' section shows the query was executed 'a few seconds ago'.

```
1 CREATE DATABASE soccer;
2 USE soccer;
```

8.49s default text

✓ Success.

Query History Saved Queries

a few seconds ago ✓ CREATE DATABASE soccer

The screenshot shows the Hive web interface with a more complex query. The query creates a table named 'soccer.results' with columns: 'date' (DATE), 'home_team' (STRING), 'away_team' (STRING), 'home_score' (INT), 'away_score' (INT), 'tournament' (STRING), 'city' (STRING), 'country' (STRING), and 'neutral' (BOOLEAN). It also specifies 'ROW FORMAT DELIMITED', 'FIELDS TERMINATED BY '\t'', and 'TBLPROPERTIES ("skip.header.line.count"="1")'. The execution time is 0s. A green checkmark and 'Success.' confirm the query was executed successfully. The 'Query History' section at the bottom shows the query was executed 'a few seconds ago'.


```
1 CREATE TABLE soccer.results(
2     date DATE,
3     home_team STRING,
4     away_team STRING,
5     home_score INT,
6     away_score INT,
7     tournament STRING,
8     city STRING,
9     country STRING,
10    neutral BOOLEAN
11 )
12 ROW FORMAT DELIMITED
13 FIELDS TERMINATED BY '\t'
14 TBLPROPERTIES ("skip.header.line.count"="1");
```

0s default text

✓ Success.

Query History Saved Queries

a few seconds ago ✓ CREATE TABLE soccer.results(date DATE, home_team STRING, away_team STRING,

 Hive

Add a name...

Add a description...

0s default text

```
1 CREATE TABLE soccer.goalscorers(  
2     date STRING,  
3     home_team STRING,  
4     away_team STRING,  
5     team STRING,  
6     scorer STRING,  
7     minute INT,  
8     own_goal BOOLEAN,  
9     penalty BOOLEAN  
10 )  
11 ROW FORMAT DELIMITED  
12 FIELDS TERMINATED BY '\t'  
13 TBLPROPERTIES ("skip.header.line.count"="1");
```


Success.

Query History

Saved Queries

a few seconds ago

CREATE TABLE soccer.goalscorers(
date STRING, home team STRING.

 Hive

Add a name...

Add a description...

0s default text

```
1 LOAD DATA LOCAL INPATH "/home/cloudera/tarea_3/apartado_1/results.csv"  
2 LOAD DATA LOCAL INPATH "/home/cloudera/tarea_3/apartado_1/goalscorers"
```

Success.

Query History

Saved Queries

a few seconds ago

LOAD DATA LOCAL INPATH "/home/cloudera/tarea_3
/apartado_1/results.csv" INTO TABLE

```

CREATE TABLE soccer.results (
    date DATE,
    home_team STRING,
    away_team STRING,
    home_score INT,
    away_score INT,
    tournament STRING,
    city STRING,
    country STRING,
    neutral BOOLEAN
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
TBLPROPERTIES ("skip.header.line.count"="1");
CREATE TABLE soccer.goalscorers (
    date STRING,
    home_team STRING,
    away_team STRING,
    team STRING,
    scorer STRING,
    minute INT,
    own_goal BOOLEAN,
    penalty BOOLEAN
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
TBLPROPERTIES ("skip.header.line.count"="1");

```


```
LOAD DATA LOCAL INPATH
"/home/cloudera/tarea_3/apartado_1/results.csv" INTO TABLE
soccer.results;

LOAD DATA LOCAL INPATH
"/home/cloudera/tarea_3/apartado_1/goalscorers.csv" INTO TABLE
soccer.goalscorers;
```

2 - Querys:

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

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



The screenshot shows the Hive web interface. At the top, there's a header with the Hive logo, a refresh button, and fields for "Add a name..." and "Add a description...". Below the header, there's a toolbar with icons for saving, copying, and settings. The main area displays a SQL query with line numbers 1, 2, and 3. Below the query, there's a "Query History" tab, a "Saved Queries" tab, and a "Results (1)" tab. The "Results (1)" tab is active, showing a table with one column, "num_goles", and one row with the value 55.

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

num_goles
55

2.2 Listado de los 5 partidos más recientes que ha jugado la selección española.

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

2m, 59s default text ?

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

Query History Saved Queries Results (5)

	results.date	results.home_team	results.away_team	results.home_score	results.away_score	results.tournament	results.city	ri
1	2024-11-18	Spain	Switzerland	3	2	UEFA Nations League	Santa Cruz de Tenerife	S
2	2024-11-15	Denmark	Spain	1	2	UEFA Nations League	Copenhagen	D
3	2024-10-15	Spain	Serbia	3	0	UEFA Nations League	Cordoba	S
4	2024-10-12	Spain	Denmark	1	0	UEFA Nations League	Murcia	S
5	2024-09-08	Switzerland	Spain	1	4	UEFA Nations League	Geneva	S

2.3 Número de goles que ha marcado España en toda su historia. Esta información debe extraerse de results, ya que goalscorers no contiene todos los goles.

```
SELECT

SUM(CASE WHEN home_team = 'Spain' THEN home_score ELSE 0 END) +

SUM(CASE WHEN away_team = 'Spain' THEN away_score ELSE 0 END) AS

total_goles

FROM soccer.results;
```



The screenshot shows the Hive web interface. At the top, there's a header with the Hive logo and options to 'Add a name...' and 'Add a description...'. Below this is a text area containing a SQL query to calculate the total goals scored by Spain. The query is as follows:

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

Below the query editor, there's a navigation bar with 'Query History', 'Saved Queries', and 'Results (1)'. The 'Results (1)' tab is active, showing a table with the following data:

total_goles	
1	1553

2.4 Listado de los 5 máximos goleadores con la selección española (sin contar autogoles).

```
SELECT scorer, COUNT(*) AS goles
FROM soccer.goalscorers
WHERE team = 'Spain' AND own_goal = FALSE
GROUP BY scorer
ORDER BY goles DESC
LIMIT 5;
```

The screenshot shows the Hive web interface. At the top, there's a header with the Hive logo and options to 'Add a name...' and 'Add a description...'. Below this, a toolbar contains icons for saving, opening, and other actions. The main area displays the SQL query being executed, with a status bar indicating '5m, 34s' and 'default' text. The query is as follows:



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


Below the query editor, there's a section for 'Query History', 'Saved Queries', and 'Results (5)'. The 'Results (5)' tab is active, showing a table with the following data:

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







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

```
SELECT DISTINCT g.scorer AS jugador
FROM soccer.goalscorers g
JOIN soccer.results r
  ON g.date = r.date AND g.home_team = r.home_team AND
  g.away_team = r.away_team
WHERE g.team = 'Spain'
      AND g.penalty = TRUE
      AND r.tournament = 'UEFA Euro'
ORDER BY jugador ASC;
```

 **Hive**  *Add a name...* *Add a description...*

6m, 45s  de

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

Query History   Saved Queries   Results (4)  

g.scorer

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

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

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

7m, 33

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

Query History  

Saved Queries  

Results (5)  

	g.scorer	goles
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 del entregable 2 trabajamos con el conjunto de datos de películas y series de la plataforma Amazon Prime, publicado en Kaggle por OctopusTeam. Además de Amazon Prime, OctopusTeam también publica el conjunto de datos de las otras principales plataformas de streaming:

- Netflix
- Apple TV+
- Amazon Prime
- Hulu
- HBO Max

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

Solo se deben tener en cuenta aquellas series o películas que están registradas en IMDB (tienen un imdbId). Puedes eliminar las otras filas directamente de los archivos de datos, antes de hacer la carga.

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

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

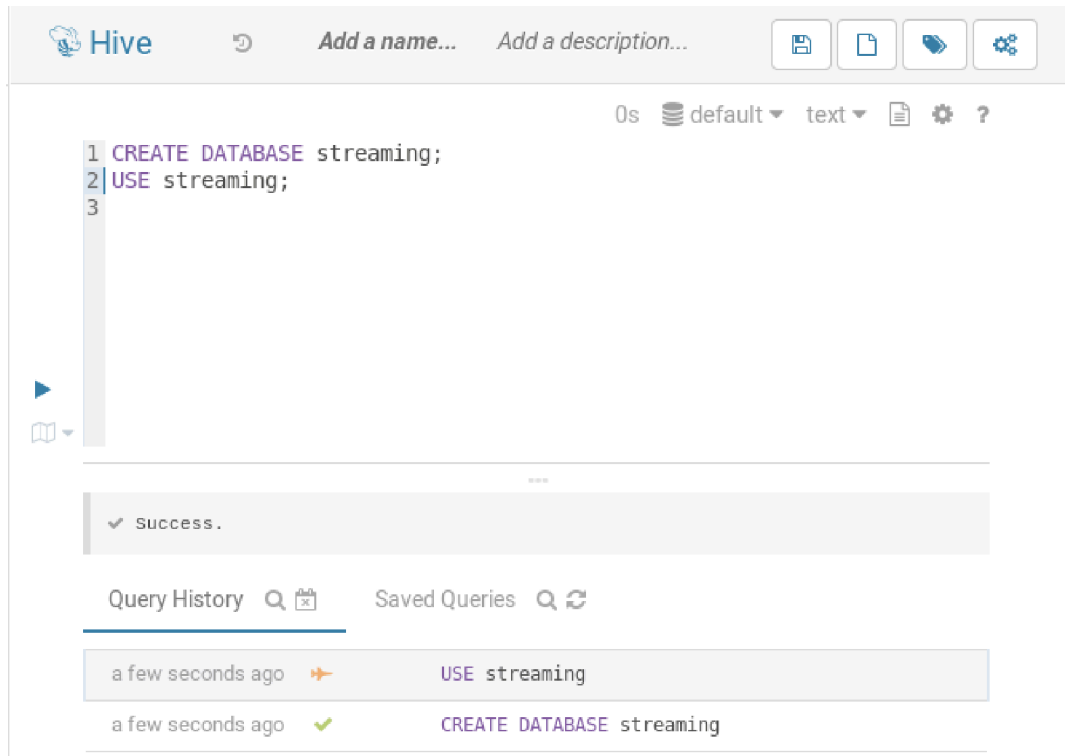
1 - Importación de datos:

He decidido descargar los archivos csv en mi ordenador local, formatearlos y después de esto ya están listos para trabajar con ellos. Para enviarlos a la máquina virtual he usado el comando `scp` apuntando al directorio que me interesa

```
~/Downloads (4.937s)
scp -o HostKeyAlgorithms=+ssh-rsa netflix.csv cloudera@192.168.100.103:/home/cloudera/tarea_3/apartado_2
cloudera@192.168.100.103's password:
netflix.csv 100% 4693KB 12.7

~/Downloads
scp -o HostKeyAlgorithms=+ssh-rsa apple.csv cloudera@192.168.100.103:/home/cloudera/tarea_3/apartado_2
```

Una vez hecho esto para cada archivo podemos crear la base de datos, las tablas y empezar a cargar los datos en Hive.

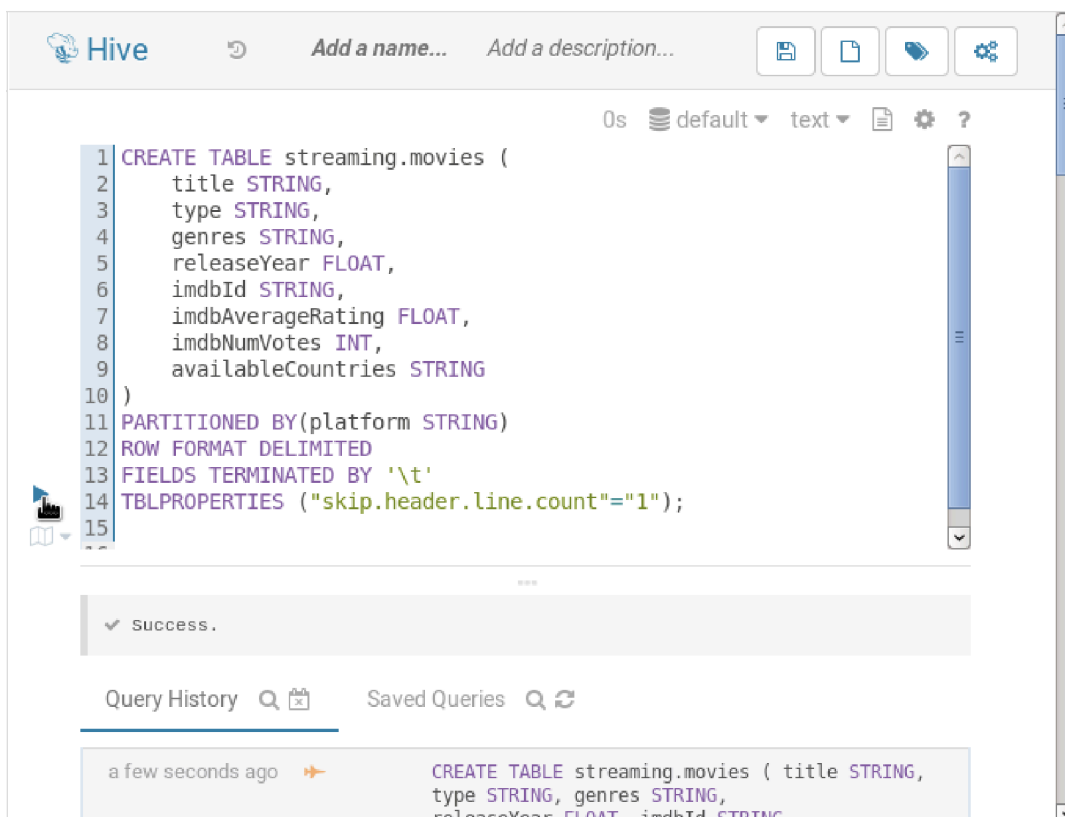


The screenshot shows the Hive web interface. At the top, there's a header with the Hive logo, a refresh button, and fields for "Add a name..." and "Add a description...". Below the header, there are icons for saving, opening, and other actions. The main area displays a SQL query:

```
1 CREATE DATABASE streaming;
2 USE streaming;
3
```

Below the query, there's a success message: "✓ Success." At the bottom, there's a "Query History" section showing a list of queries:

Time	Status	Query
a few seconds ago	✗	USE streaming
a few seconds ago	✓	CREATE DATABASE streaming



The screenshot shows the Hive web interface. At the top, there's a header with the Hive logo, a refresh button, and fields for "Add a name..." and "Add a description...". Below the header, there are icons for saving, opening, and other actions. The main area displays a SQL query:

```
1 CREATE TABLE streaming.movies (
2   title STRING,
3   type STRING,
4   genres STRING,
5   releaseYear FLOAT,
6   imdbId STRING,
7   imdbAverageRating FLOAT,
8   imdbNumVotes INT,
9   availableCountries STRING
10 )
11 PARTITIONED BY(platform STRING)
12 ROW FORMAT DELIMITED
13 FIELDS TERMINATED BY '\t'
14 TBLPROPERTIES ("skip.header.line.count"="1");
15
```

Below the query, there's a success message: "✓ Success." At the bottom, there's a "Query History" section showing a list of queries:

Time	Status	Query
a few seconds ago	✗	CREATE TABLE streaming.movies (title STRING, type STRING, genres STRING, releaseYear FLOAT, imdbId STRING,

```
1 LOAD DATA LOCAL INPATH 'home/cloudera/tarea_3/apartado_2/amazon.csv'
2 INTO TABLE streaming.movies
3 PARTITION(platform = 'amazon_prime');
4
5 LOAD DATA LOCAL INPATH 'home/cloudera/tarea_3/apartado_2/apple.csv'
6 INTO TABLE streaming.movies
7 PARTITION(platform = 'apple_tv');
8
9 LOAD DATA LOCAL INPATH 'home/cloudera/tarea_3/apartado_2/hbo.csv'
10 INTO TABLE streaming.movies
11 PARTITION(platform = 'hbo_max');
12
13 LOAD DATA LOCAL INPATH 'home/cloudera/tarea_3/apartado_2/hulu.csv'
14 INTO TABLE streaming.movies
15 PARTITION(platform = 'hulu');
```

✓ Success.

Query History 🔍 📅 Saved Queries 🔍 🔄

a few seconds ago ✓ LOAD DATA LOCAL INPATH 'home/cloudera/tarea_3/apartado_2/hulu.csv'
INTO TABLE streaming.movies PARTITION(platform = 'hulu')

```
CREATE DATABASE streaming;

USE streaming;

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");
```

```
LOAD DATA LOCAL INPATH
  '/home/cloudera/tarea_3/apartado_2/amazon.csv' INTO TABLE
  streaming.movies

PARTITION(platform = 'amazon_prime');

LOAD DATA LOCAL INPATH
  '/home/cloudera/tarea_3/apartado_2/apple.csv' INTO TABLE
  streaming.movies

PARTITION(platform = 'apple_tv');

LOAD DATA LOCAL INPATH
  '/home/cloudera/tarea_3/apartado_2/hbo.csv' INTO TABLE
  streaming.movies

PARTITION(platform = 'hbo_max');

LOAD DATA LOCAL INPATH
  '/home/cloudera/tarea_3/apartado_2/hulu.csv' INTO TABLE
  streaming.movies

PARTITION(platform = 'hulu');

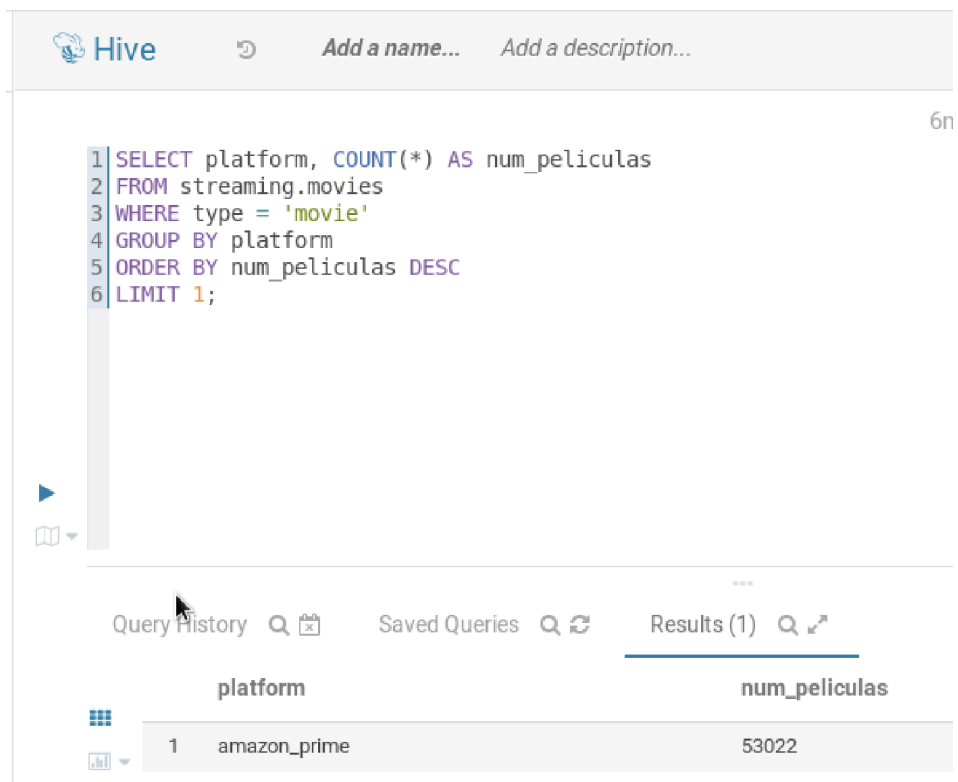
LOAD DATA LOCAL INPATH
  '/home/cloudera/tarea_3/apartado_2/netflix.csv' INTO TABLE
  streaming.movies

PARTITION(platform = 'netflix');
```


2 - Querys:

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

```
SELECT platform, COUNT(*) AS num_peliculas  
FROM streaming.movies  
WHERE type = 'movie'  
GROUP BY platform  
ORDER BY num_peliculas DESC  
LIMIT 1;
```



The screenshot shows the Hive web interface. At the top, there's a header with the Hive logo and options to 'Add a name...' and 'Add a description...'. Below the header, the query editor contains the following SQL code:

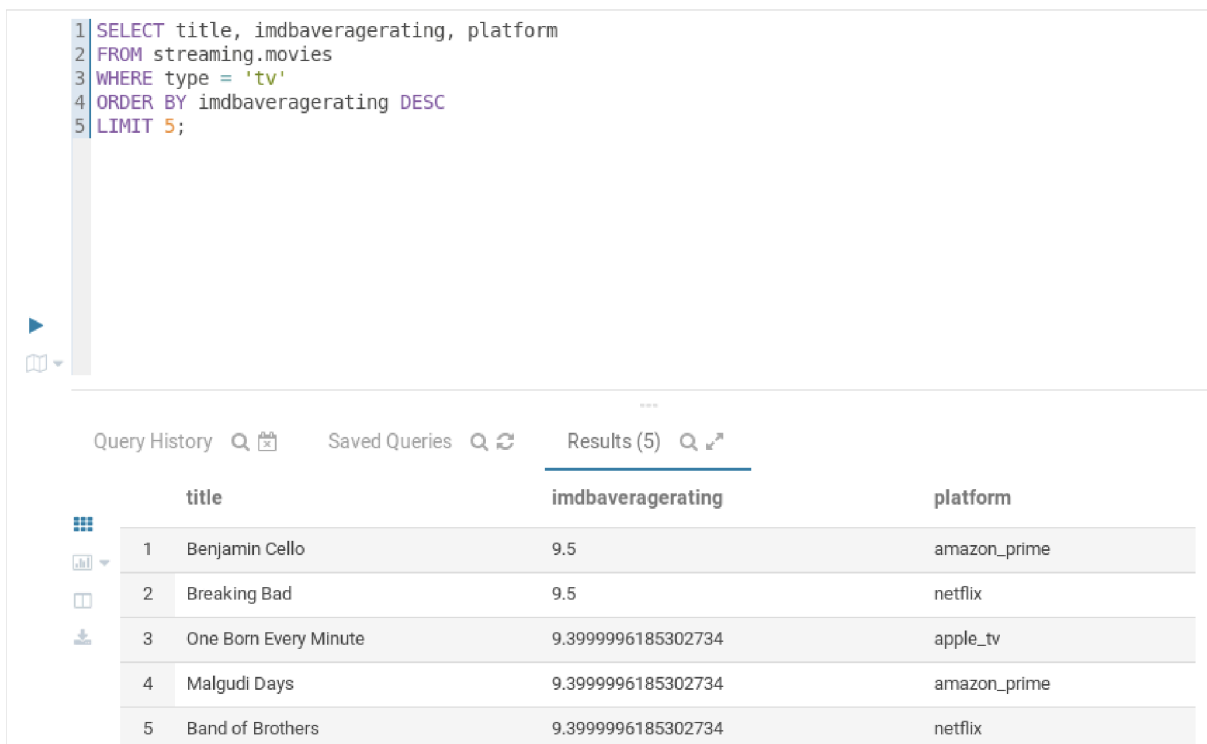
```
1 SELECT platform, COUNT(*) AS num_peliculas  
2 FROM streaming.movies  
3 WHERE type = 'movie'  
4 GROUP BY platform  
5 ORDER BY num_peliculas DESC  
6 LIMIT 1;
```

Below the query editor, there are tabs for 'Query History', 'Saved Queries', and 'Results (1)'. The 'Results (1)' tab is active, displaying a table with two columns: 'platform' and 'num_peliculas'. The table contains one row of data:

	platform	num_peliculas
1	amazon_prime	53022

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

```
SELECT title, imdbaveragerating, platform
FROM streaming.movies
WHERE type = 'tv'
ORDER BY imdbaveragerating DESC
LIMIT 5;
```



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

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

Below the editor, the results are displayed in a table with 5 rows. The table has three columns: title, imdbaveragerating, and platform. The results are sorted by imdbaveragerating in descending order.

	title	imdbaveragerating	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

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

```
SELECT platform, SUM(imdbNumVotes) AS total_votos
FROM streaming.movies
WHERE genres LIKE '%Sci-Fi%'
      OR genres LIKE '%Science Fiction%'
      AND type = 'tv'
GROUP BY platform
ORDER BY total_votos DESC;
```

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

Query History

Saved Queries

Results (5)

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

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

```
SELECT releaseyear, COUNT(*) AS num_peliculas
FROM streaming.movies
WHERE type = 'movie'
GROUP BY releaseYear
ORDER BY num_peliculas DESC
LIMIT 5;
```

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



Query History

Saved Queries

Results (5)

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