

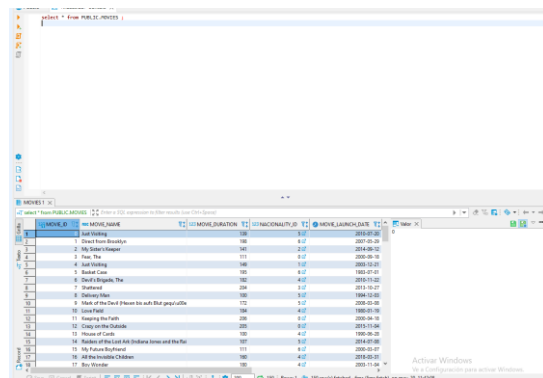
## 1.1 Devuelve todas las películas. Selección sin restricciones

La sentencia **SELECT** es la única que permite consultar información de forma que puede devolver un registro completo de datos o un solo valor. Se debe considerar que es la única función que permite obtener cualquier tipo de información desde las tablas de una base de datos.

El símbolo **\*** representa todos los registros (filas) así como todas las columnas de una determinada tabla.

### 1.1.1 Caso1 :Listar todos los registros de una tabla

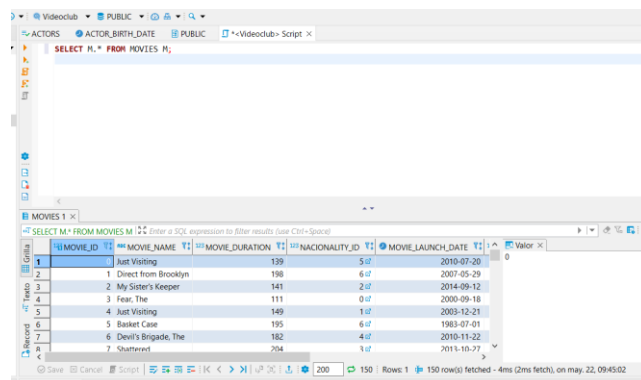
```
select * from PUBLIC.MOVIES ;
```



MOVIE_ID	MOVIE_NAME	MOVIE_DURATION	NACIONALITY_ID	MOVIE_LAUNCH_DATE
1	Just Visiting	139	5	2010-07-26
2	Direct from Brooklyn	139	6	2007-09-29
3	My Sister's Keeper	141	2	2014-09-12
4	Fear, The	111	0	2000-09-18
5	Just Visiting	149	1	2003-12-21
6	Devil's Brigade, The	195	6	1983-07-01
7	Chatterbox	204	1	2011-10-27
8	Chatterbox	204	1	2011-10-27
9	Mark of the Cross	172	5	2008-05-08
10	Just Visiting	139	5	2010-07-26
11	Keeping the Faith	209	0	2006-04-19
12	They Call It the Heart	200	0	2012-11-04
13	House of Cards	100	4	1986-09-29
14	House of Cards	100	4	1986-09-29
15	My Future Husband	131	0	2005-05-07
16	My Future Husband	131	0	2005-05-07
17	My Future Husband	131	0	2005-05-07

### 1.1.2 Caso 2: USO DE ALIAS DE TABLA.

```
SELECT M.* FROM MOVIES M;
```



MOVIE_ID	MOVIE_NAME	MOVIE_DURATION	NACIONALITY_ID	MOVIE_LAUNCH_DATE
1	Just Visiting	139	5	2010-07-26
2	Direct from Brooklyn	139	6	2007-09-29
3	My Sister's Keeper	141	2	2014-09-12
4	Fear, The	111	0	2000-09-18
5	Just Visiting	149	1	2003-12-21
6	Devil's Brigade, The	195	6	1983-07-01
7	Chatterbox	204	1	2011-10-27

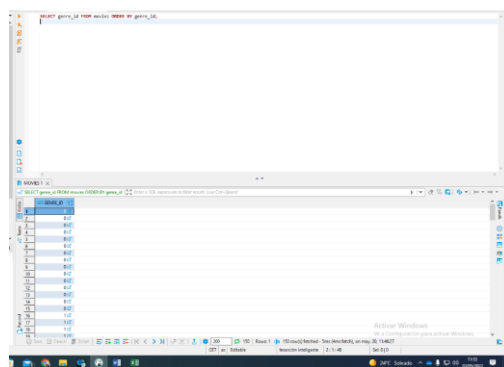
## 1.2 Devuelve todos los géneros de las películas, selección de datos limitando las columnas de una tabla:

Este tipo de consulta especifica las columnas que se deben mostrar del total de las presentes en una tabla mediante el nombre de la columna de la tabla antes del **FROM**.

### 1.2.1 Géneros ordenados.

**ORDER BY**, este tipo de consulta permite ordenar de forma ascendente (defecto) o descendentes los valores de una determinada columna o columnas de una tabla. En nuestro caso ordenamos por genero .

```
SELECT genre_id FROM movies ORDER BY genre_id;
```



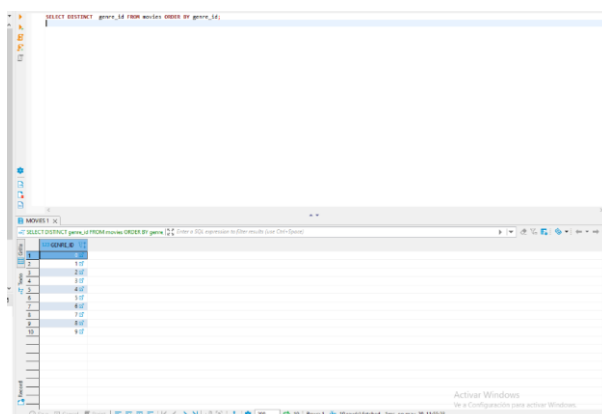
The screenshot shows a SQL query window with the query: `SELECT genre_id FROM movies ORDER BY genre_id;` The results pane displays a list of genre IDs: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100.

### 1.2.2 Géneros ordenados sin repetición. Restringir los datos repetidos.

La sintaxis **SELECT DISTINCT** nombre\_columna **FROM** nombre\_tabla;

#### 1.2.2.1 Solo los códigos de género.

```
SELECT DISTINCT genre_id FROM movies ORDER BY genre_id;
```



The screenshot shows a SQL query window with the query: `SELECT DISTINCT genre_id FROM movies ORDER BY genre_id;` The results pane displays a list of distinct genre IDs: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100.

### 1.2.3 Códigos de género con su nombre y editando los encabezados.

Mediante le uso de as lo logramos:

```
SELECT DISTINCT GENRE_ID AS ID_GÈNERO, GENRE_NAME AS GÈNERO FROM GENRE;
```

SQL Query: `SELECT DISTINCT GENRE_ID AS ID_GÉNERO, GENRE_NAME AS GÉNERO FROM GENRE;`

ID_GÉNERO	GÉNERO
1	Crime
2	Documentary
3	Mystery
4	Drama
5	Comedy
6	Horror
7	Animation
8	Thriller
9	Fantasy
10	War

#### 1.2.4 Géneros ordenados por id y género, nombre de película.

**ORDER BY** , este tipo de consulta permite ordenar de forma ascendente (defecto) o descendentes los valores de una determinada columna o columnas de una tabla. En nuestro caso ordenamos primero por género y luego por película (parece lo más lógico dentro de un videoclub).

**INNER JOIN**, permite mostrar la información cruzada entre dos o más tablas para diferentes procesos. Todo proviene de llave foránea de la tabla, ya que es la que permite acceder a la otra tabla y así combinarla.

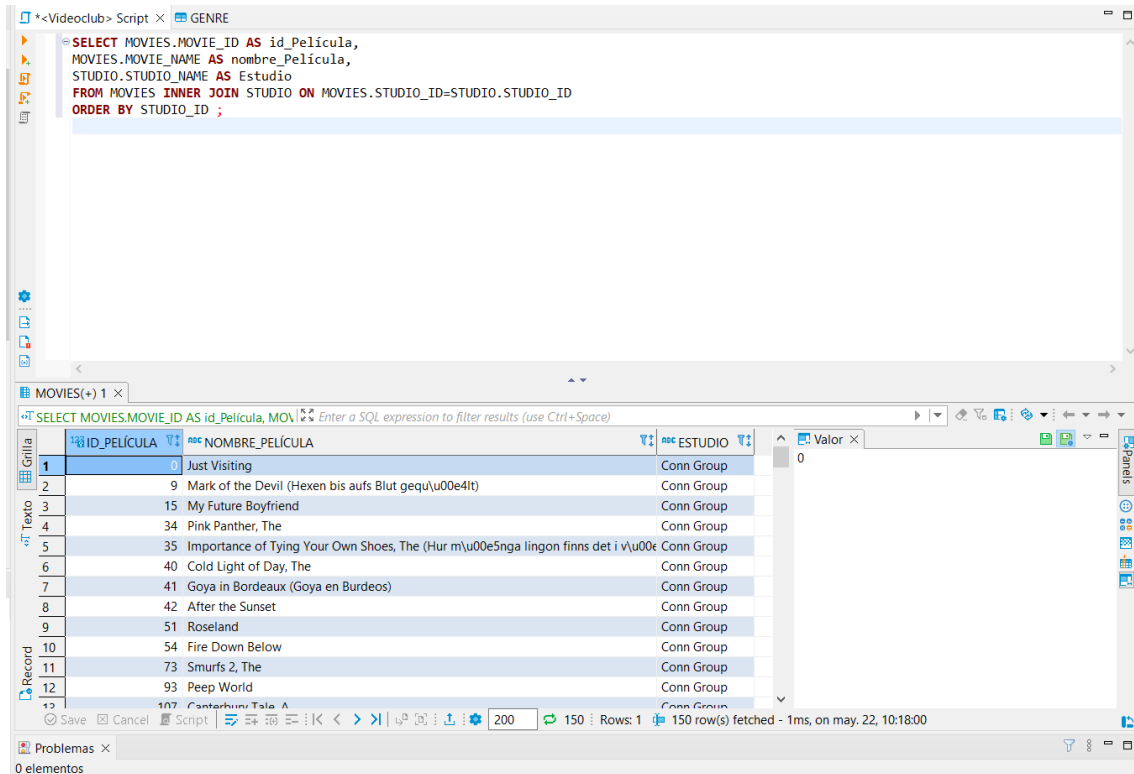
```
SELECT MOVIES.MOVIE_ID AS ID_Película,
MOVIES.MOVIE_NAME as Nombre_Películas ,
GENRE.GENRE_ID as Género_id,
GENRE_NAME as nombre_Género
FROM MOVIES
INNER JOIN GENRE ON MOVIES.GENRE_ID =GENRE.GENRE_ID
ORDER BY GENRE_ID,MOVIE_ID ;
```

SQL Query: `SELECT MOVIES.MOVIE_ID AS ID_Película, MOVIES.MOVIE_NAME as Nombre_Películas , GENRE.GENRE_ID as Género_id, GENRE_NAME as nombre_Género FROM MOVIES INNER JOIN GENRE ON MOVIES.GENRE_ID =GENRE.GENRE_ID ORDER BY GENRE_ID,MOVIE_ID ;`

ID_PELÍCULA	NOMBRE_PELÍCULAS	GÉNERO_ID	NOMBRE_GÉNERO
1	Keeping the Faith	0	Crime
2	Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark)	0	Crime
3	All the Invisible Children	0	Crime
4	Memory Lane	0	Crime
5	Goya in Bordeaux (Goya en Burdeos)	0	Crime
6	Knightsiders	0	Crime
7	My Brother Tom	0	Crime
8	Descent Part 2: The	0	Crime
9	Shao Lin (Shao Lin si)	0	Crime
10	Mars	0	Crime
11	CRGB	0	Crime
12	Green Up 2	0	Crime
13	Maximum Overdrive (Maximum Overdrive)	0	Crime

### 1.3 Devuelve el nombre de todas las películas y el nombre del estudio que las ha realizado.

```
SELECT MOVIES.MOVIE_ID AS id_Película,  
MOVIES.MOVIE_NAME AS nombre_Película,  
STUDIO.STUDIO_NAME AS Estudio  
FROM MOVIES INNER JOIN STUDIO ON MOVIES.STUDIO_ID=STUDIO.STUDIO_ID  
ORDER BY STUDIO_ID ;
```



The screenshot shows a database IDE window titled "Videoclub" with a script editor and a results grid. The script editor contains the following SQL query:

```
SELECT MOVIES.MOVIE_ID AS id_Película,  
MOVIES.MOVIE_NAME AS nombre_Película,  
STUDIO.STUDIO_NAME AS Estudio  
FROM MOVIES INNER JOIN STUDIO ON MOVIES.STUDIO_ID=STUDIO.STUDIO_ID  
ORDER BY STUDIO_ID ;
```

The results grid displays the following data:

ID_PELÍCULA	NOMBRE_PELÍCULA	ESTUDIO
1	Just Visiting	Conn Group
2	9 Mark of the Devil (Hexen bis aufs Blut gequ\u00e4lt)	Conn Group
3	15 My Future Boyfriend	Conn Group
4	34 Pink Panther, The	Conn Group
5	35 Importance of Tying Your Own Shoes, The (Hur m\u00e5nga l\u00e4ngon finns det i \u00e4lskan)	Conn Group
6	40 Cold Light of Day, The	Conn Group
7	41 Goya in Bordeaux (Goya en Burdeos)	Conn Group
8	42 After the Sunset	Conn Group
9	51 Roseland	Conn Group
10	54 Fire Down Below	Conn Group
11	73 Smurfs 2, The	Conn Group
12	93 Peep World	Conn Group
13	107 Canterbury Tale, A	Conn Group

The status bar at the bottom indicates "Rows: 1" and "150 row(s) fetched - 1ms, on may. 22, 10:18:00".

1.4 Devuelve el nombre y la edad de todos los directores menores o iguales de 50 años.

1.4.1 Sin filtrar por edad , primero calculamos la edad de todos (sin contar años bisiestos).

Utilizamos la función [DATEDIFF](#)

Example

Return the difference between two date values, in years:

```
SELECT DATEDIFF(year, '2017/08/25', '2011/08/25') AS DateDiff;
```

Try It Yourself >

Definition and Usage

The DATEDIFF() function returns the difference between two dates.

Syntax

```
DATEDIFF(interval, date1, date2)
```

Parameter Values

Parameter	Description
interval	Required. The part to return. Can be one of the following values: <ul style="list-style-type: none"><li>year, yyyy, yy = Year</li><li>quarter, qq, q = Quarter</li><li>month, mm, m = month</li><li>dayofyear = Day of the year</li><li>day, dy, y = Day</li><li>week, ww, wk = Week</li><li>weekday, dw, w = Weekday</li><li>hour, hh = hour</li><li>minute, mi, n = Minute</li><li>second, ss, s = Second</li><li>millisecond, ms = Millisecond</li></ul>
date1, date2	Required. The two dates to calculate the difference between

Technical Details

Ilustración 1 DATEDIFF()

Utilizamos la función [FLOOR](#)

SQL Server FLOOR() Function

[< Previous](#)

[SQL Server Functions >](#)

Example

Return the largest integer value that is equal to or less than 25.75:

```
SELECT FLOOR(25.75) AS FloorValue;
```

Try It Yourself >

Definition and Usage

The FLOOR() function returns the largest integer value that is smaller than or equal to a number.

Tip: Also look at the [CEILING\(\)](#) and [ROUND\(\)](#) functions.

Syntax

```
FLOOR(number)
```

Parameter Values

Parameter	Description
number	Required. A numeric value

Ilustración 2 FLOOR().

## Utilizamos la función [IFNULL\(\)](#)

### MySQL IFNULL() Function

[< Previous](#)[MySQL Functions](#)

#### Example

Return the specified value IF the expression is NULL, otherwise return the expression:

```
SELECT IFNULL(NULL, "idSchools.com");
```

[Try it Yourself »](#)

#### Definition and Usage

The IFNULL() function returns a specified value if the expression is NULL.

If the expression is NOT NULL, this function returns the expression.

#### Syntax

```
IFNULL(expression, alt_value)
```

#### Parameter Values

Parameter	Description
expression	Required. The expression to test whether is NULL.
alt_value	Required. The value to return if expression is NULL.

#### Technical Details

**Works in:** From MySQL 4.0

#### More Examples

##### Example

Return the specified value IF the expression is NULL, otherwise return the expression:

```
SELECT IFNULL("Hello", "idSchools.com");
```

## Utilizamos la función [NOW\(\)](#).

### MySQL NOW() Function

[< Previous](#)[MySQL Functions](#)

#### Example

Return current date and time:

```
SELECT NOW();
```

[Try it Yourself »](#)

#### Definition and Usage

The NOW() function returns the current date and time.

**Note:** The date and time is returned as "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS.uuuuuu (numeric).

#### Syntax

```
NOW()
```

#### Technical Details

**Works in:** From MySQL 4.0

#### More Examples

##### Example

Return current date and time + 1:

```
SELECT NOW() + 1;
```

[Try it Yourself »](#)

```
SELECT ACTORS.ACTOR_ID AS Id_Actor,
ACTORS.ACTOR_NAME AS nombre_Actor,
FLOOR(DATEDIFF(IFNULL(ACTORS.ACTOR_DEAD_DATE,NOW()),ACTORS.ACTOR_BIRTH_DATE)/
365) AS edad
FROM ACTORS ;
```

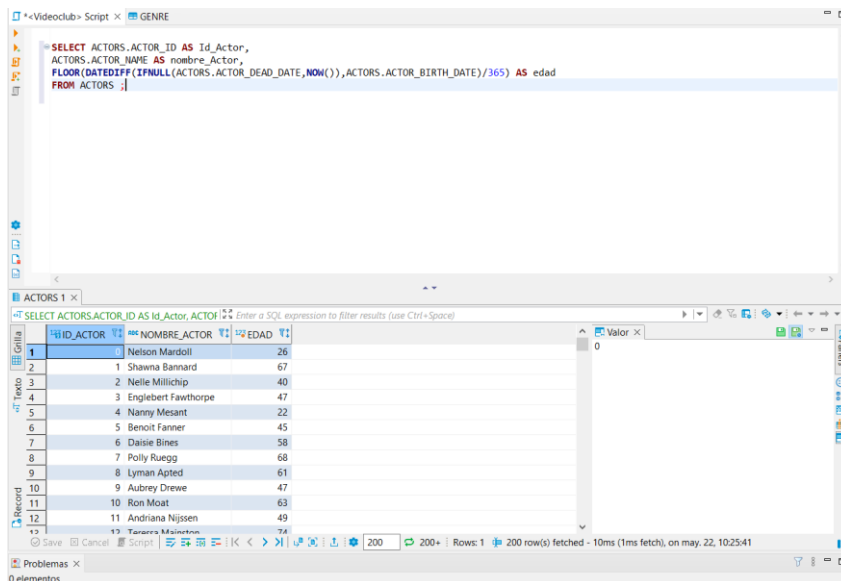


Ilustración 3 SIN FILTRAR POR EDAD

## 1.4.2 Filtrando por menores de 50 años.

Añadimos una sentencia [WHERE](#).

### SQL WHERE Clause

[← Previous](#)

#### The SQL WHERE Clause

The **WHERE** clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

#### WHERE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

**Note:** The **WHERE** clause is not only used in **SELECT** statements, it is also used in **UPDATE**, **DELETE**, etc.!

```
SELECT ACTORS.ACTOR_ID AS Id_Actor,
ACTORS.ACTOR_NAME AS nombre_Actor,
FLOOR(DATEDIFF(IFNULL(ACTORS.ACTOR_DEAD_DATE, NOW()), ACTORS.ACTOR_BIRTH_DATE)/
365) AS edad
FROM ACTORS
WHERE
FLOOR(DATEDIFF(IFNULL(ACTORS.ACTOR_DEAD_DATE, NOW()), ACTORS.ACTOR_BIRTH_DATE)/
365)<=50;
```

DBeaver 22.0.4 - «Videoclub» Script

Archivo Editar Navegar Buscar Editor SQL Base de Datos Ventana Ayuda

Navegador de Bases de Datos Proyectos «Videoclub» Script GENRE

Ingrese parte del nombre de un objeto aquí

- Videoclub
  - PUBLIC
    - INFORMATION\_SCHEMA
    - SYSTEM\_JOBS
    - PUBLIC
      - Tablas
        - ACTORS
        - AWARDS
        - DIRECTORS
        - GENRE
      - Columnas
        - GENRE\_ID (INTEGER)
        - GENRE\_NAME (VARCHAR(100))
      - Claves
      - Columnas de clave externa
      - Indíces
      - Referencias
      - Triggers
    - MEMBERS
    - MEMBERS\_MOVIE\_RENTAL
    - MOVIES
      - Columnas
        - MOVIE\_ID (INTEGER)
        - MOVIE\_NAME (VARCHAR(100))
        - MOVIE\_DURATION (INTEGER)
        - NACIONALITY\_ID (INTEGER)
        - MOVIE\_LAUNCH\_DATE (DATE)

Project - General DataSource

Name

- Bookmarks
- ER Diagrams
- Scripts

```
SELECT ACTORS.ACTOR_ID AS Id_Actor,
ACTORS.ACTOR_NAME AS nombre_Actor,
FLOOR(DATEDIFF(TFNULL(ACTORS.ACTOR_DEAD_DATE, NOW()), ACTORS.ACTOR_BIRTH_DATE) / 365) AS edad
FROM ACTORS
WHERE FLOOR(DATEDIFF(TFNULL(ACTORS.ACTOR_DEAD_DATE, NOW()), ACTORS.ACTOR_BIRTH_DATE) / 365) <= 50;
```

ACTORS 1

```
SELECT ACTORS.ACTOR_ID AS Id_Actor, ACTORS.ACTOR_NAME AS nombre_Actor, FLOOR(DATEDIFF(TFNULL(ACTORS.ACTOR_DEAD_DATE, NOW()), ACTORS.ACTOR_BIRTH_DATE) / 365) AS edad
```

ID_ACTOR	NOMBRE_ACTOR	EDAD
1	Nelson Mardoll	26
2	Nelle Millichip	40
3	Englebert Fawthorpe	47
4	Nanny Mesant	22
5	Benoit Fanner	45
6	Aubrey Drewe	47
7	Andriana Nijssen	49
8	Ermano Stud	35
9	Abbie Woolforde	48
10	Natty Quig	48
11	Augustin Forcer	32
12	Fulton Mortelli	47
13	Linnaea Amshar	46

Save Cancel Script SQL K > > > 200 134 Rows: 1 134 row(s) fetched - 16ms (1ms fetch), on may. 22, 10:28:17

Problemas 0 elementos

Descripción	Recursos	Ruta	Ubicación	Tipo
-------------	----------	------	-----------	------

CET es Editable Inserción inteligente 7:1:275 Set 0:0



## 1.5 Devuelve el nombre y la edad de todos los actores menores de 50 años que hayan fallecido.

Usamos el comando IS NOT NULL , que nos sirve para evaluar los campos no vacios, es decir aquellos que tengan fecha de muerte, claramente están muertos.

```
SELECT ACTORS.ACTOR_ID AS Id_Actor,  
ACTORS.ACTOR_NAME AS nombre_Actor,  
FLOOR(DATEDIFF(ACTORS.ACTOR_DEAD_DATE,ACTORS.ACTOR_BIRTH_DATE)/365) AS edad  
FROM ACTORS  
WHERE  
ACTOR_DEAD_DATE IS NOT NULL  
AND FLOOR(DATEDIFF(ACTORS.ACTOR_DEAD_DATE,ACTORS.ACTOR_BIRTH_DATE)/365)<=50  
ORDER BY edad;
```

ACTORS 1 x

```
SELECT ACTORS.ACTOR_ID AS Id_Actor,  
ACTORS.ACTOR_NAME AS nombre_Actor,  
FLOOR(DATEDIFF(ACTORS.ACTOR_DEAD_DATE,ACTORS.ACTOR_BIRTH_DATE)/365) AS edad  
FROM ACTORS  
WHERE  
ACTOR_DEAD_DATE IS NOT NULL  
AND FLOOR(DATEDIFF(ACTORS.ACTOR_DEAD_DATE,ACTORS.ACTOR_BIRTH_DATE)/365)<=50  
ORDER BY edad;
```

ID	ID_ACTOR	NOMBRE_ACTOR	EDAD
1	169	Easter Moon	20
2	183	Chloe Redmayne	20
3	136	Sean Wressell	22
4	240	Benjamin Renfrew	22
5	134	Jilli MacCaffery	27
6	57	Ebenezer Brandli	28
7	92	Aura Donativo	29
8	61	Harriette Mucker	31
9	109	Daniel MacWilliams	33
10	108	Daniel MacWilliams	33
11	107	Daniel MacWilliams	33
12	106	Daniel MacWilliams	33
13	105	Daniel MacWilliams	33
14	104	Daniel MacWilliams	33
15	103	Daniel MacWilliams	33

Problemas x

Compruebo 1-> id=170; 1984-1963=21 pero como ese año no llego a su cumpleaños (-1)  
=muerto a los 20.(vaya putada)

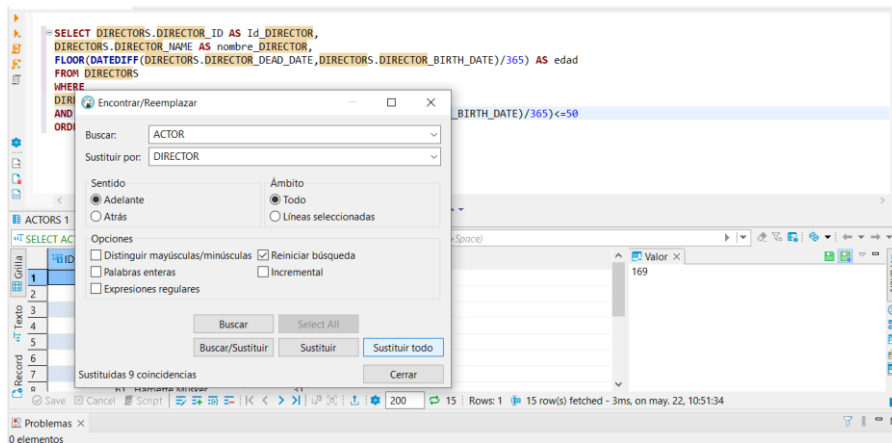
Devuelve el nombre y la edad de todos los actores menores de 50 años que hayan fallecido

	asc ACTOR_NAME	AGE
1	Ebenezer Brandli	39
2	Virgil Yakubowicz	45
3	Aura Donativo	41
4	Sean Wressell	45
5	Chloe Redmayne	40
6	Benjamin Renfrew	38

No coincide con la captura de campus dual donde solo 6 están muertos y con menos de 50 años

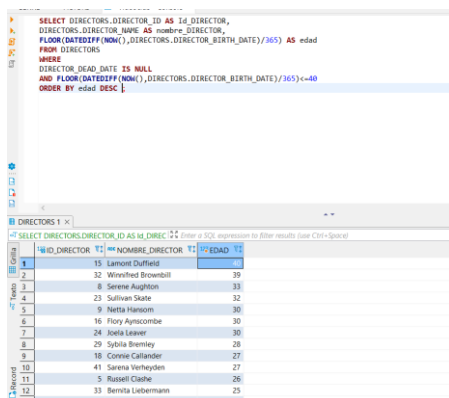
## 1.6 Devuelve el nombre de todos los directores menores o iguales de 40 años que estén vivos.

Basandonos en la consulta anterior mediante buscar y remaplaar ctrl+F reemplazamos ACTORS por DIRECTORS (tabla) ACTOR por DIRECTOR (campos).



```
SELECT DIRECTORS.DIRECTOR_ID AS Id_DIRECTOR,  
DIRECTORS.DIRECTOR_NAME AS nombre_DIRECTOR,  
FLOOR(DATEDIFF(NOW(), DIRECTORS.DIRECTOR_BIRTH_DATE)/365) AS edad  
FROM DIRECTORS  
WHERE  
DIRECTOR_DEAD_DATE IS NULL  
AND FLOOR(DATEDIFF(NOW(), DIRECTORS.DIRECTOR_BIRTH_DATE)/365) <= 40  
ORDER BY edad;
```

Como debemos buscar los vivos, la fecha de muerte debe estar vacía (is null), calculamos la edad con NOW() en lugar de la fecha de la muerte y mostramos los menores o iguales de ese calculo a 40 años, por último ordenamos por edad y así poder hacer una comprobación por limite( los de 40 años), 16 directores vivos menores de 40 años (campus dual 17).



16	15	Lamont Duffield	1981-12-08	[NULL]	Mekon
----	----	-----------------	------------	--------	-------

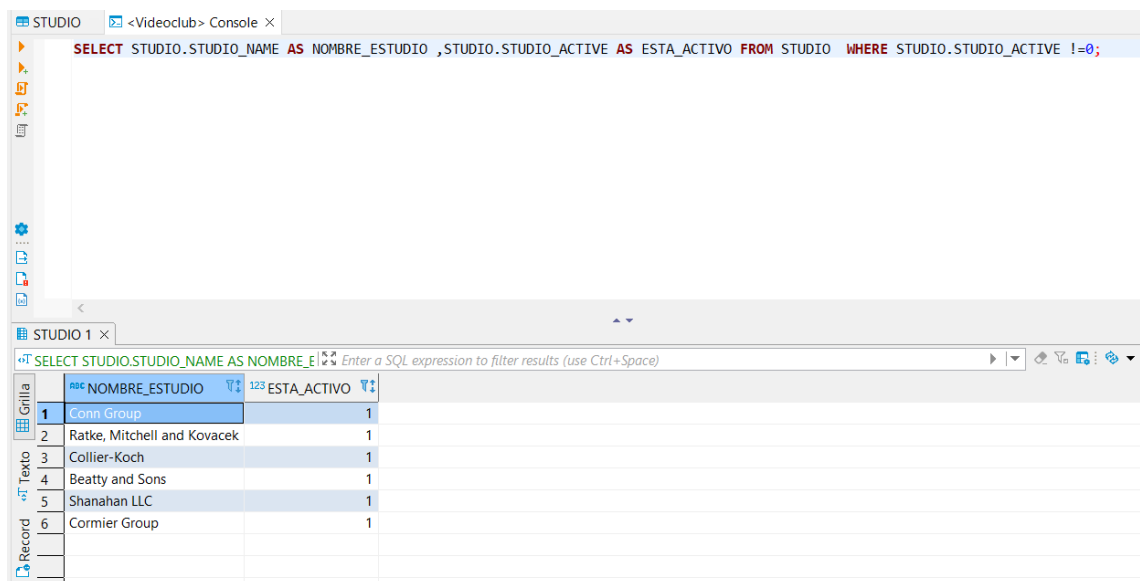
2022-1981=41 pero como aun no es agosto no llegó su cumpleaños -1 =40 años vivo.

La única diferencia con el campus dual que encuentra 17 concordancias y ahora a día de hoy son 16 es que Etti Byron ya cumplió en enero de 2022, 41 años:

13	12	Etti Byron	1981-01-10	[NULL]	Orlando
----	----	------------	------------	--------	---------

1.7 Devuelve la lista de todos los estudios de grabación que estén activos.

```
SELECT STUDIO.STUDIO_NAME AS NOMBRE_ESTUDIO ,STUDIO.STUDIO_ACTIVE AS  
ESTA_ACTIVO FROM STUDIO WHERE STUDIO.STUDIO_ACTIVE !=0;
```



The screenshot shows a database console window with the following SQL query and its results:

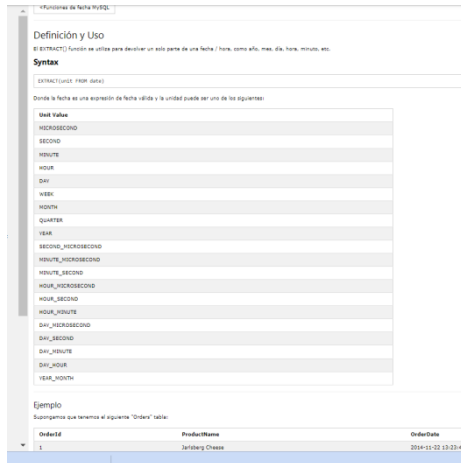
```
SELECT STUDIO.STUDIO_NAME AS NOMBRE_ESTUDIO ,STUDIO.STUDIO_ACTIVE AS  
ESTA_ACTIVO FROM STUDIO WHERE STUDIO.STUDIO_ACTIVE !=0;
```

	NOMBRE_ESTUDIO	ESTA_ACTIVO
1	Conn Group	1
2	Ratke, Mitchell and Kovacek	1
3	Collier-Koch	1
4	Beatty and Sons	1
5	Shanahan LLC	1
6	Cormier Group	1

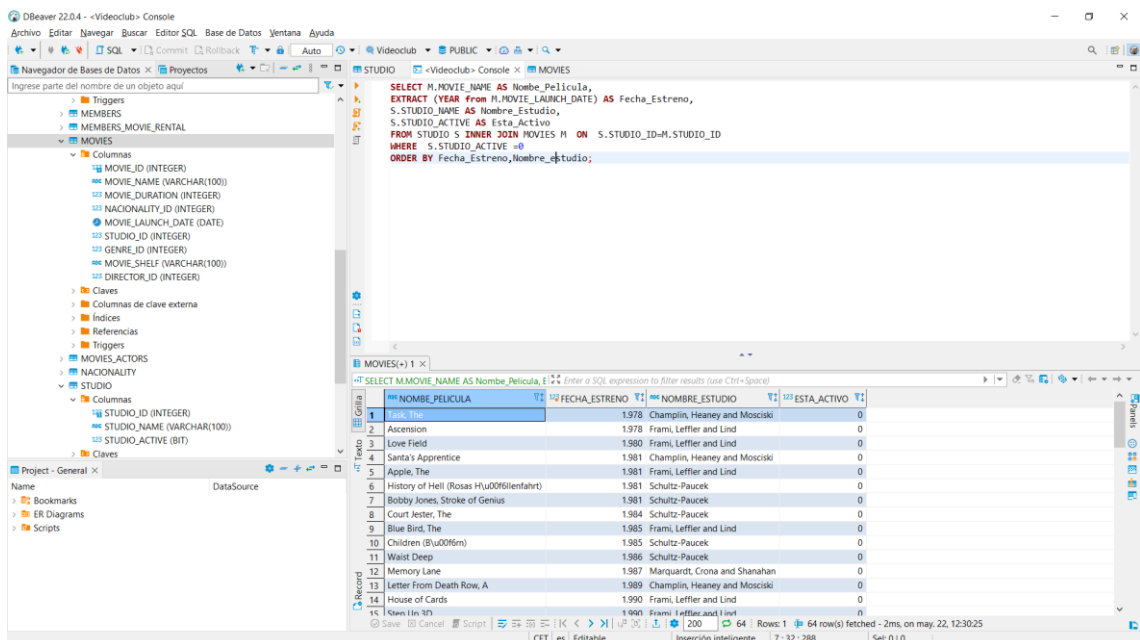
Se podría cambiar 1 por activo en el select???

## 1.8 Devuelve el nombre y el año de todas las películas que han sido producidas por un estudio que actualmente no esté activo

Utilizamos la función [Extract\(unit from date\)](#).



```
SELECT M.MOVIE_NAME AS Nombre_Pelicula,  
EXTRACT (YEAR from M.MOVIE_LAUNCH_DATE) AS Fecha_Estreno,  
S.STUDIO_NAME AS Nombre_Estudio,  
S.STUDIO_ACTIVE AS Esta_Activo  
FROM STUDIO S INNER JOIN MOVIES M ON S.STUDIO_ID=M.STUDIO_ID  
WHERE S.STUDIO_ACTIVE =0  
ORDER BY Fecha_Estreno,Nombre_estudio;
```



## 1.9 Devuelve una lista de los 20 últimos miembros en anotarse al videoclub.

### 1.9.1 Usando la sentencia Limit.

Usamos [LIMIT](#).

The following SQL statement shows the equivalent example for MySQL:

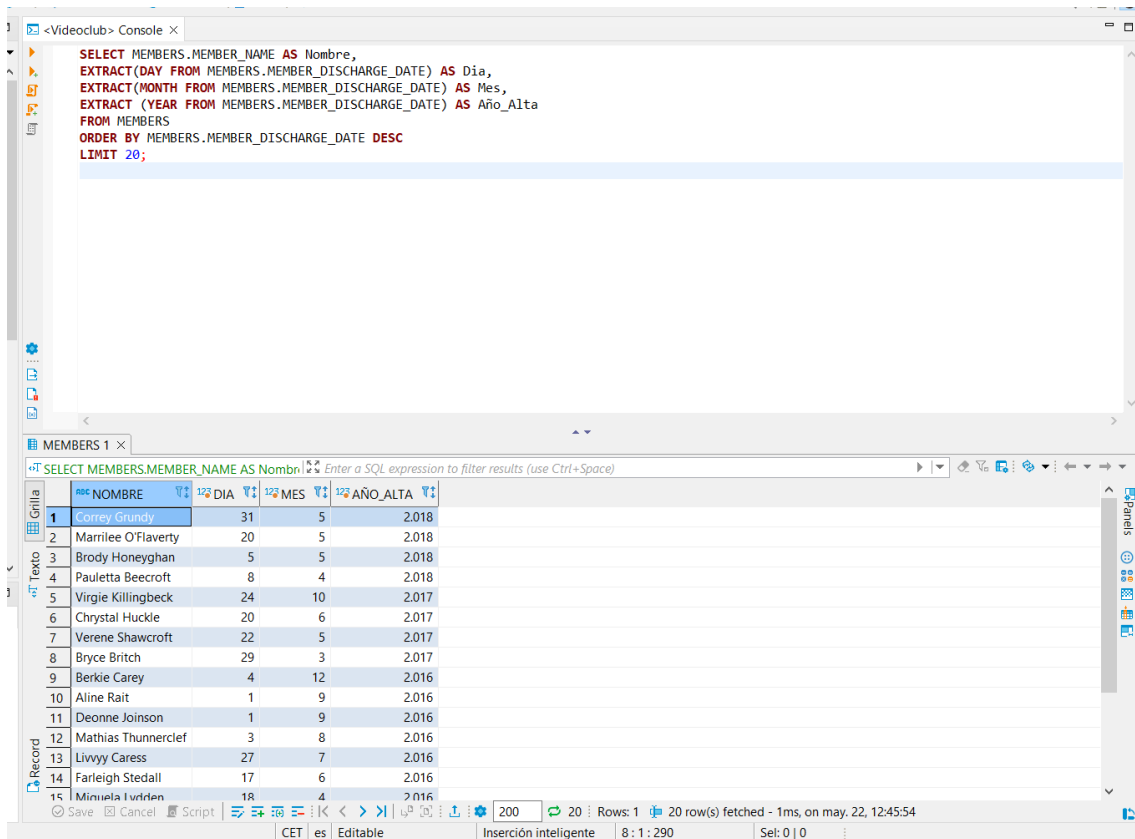
#### Example

```
SELECT * FROM Customers
WHERE Country='Germany'
LIMIT 3;
```

[Try it Yourself »](#)

Además de los nombres comprobamos que sea así separando las fecha en día, mes y año.

```
SELECT MEMBERS.MEMBER_NAME AS Nombre,
EXTRACT(DAY FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Dia,
EXTRACT(MONTH FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Mes,
EXTRACT (YEAR FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Año_Alta
FROM MEMBERS
ORDER BY MEMBERS.MEMBER_DISCHARGE_DATE DESC
LIMIT 20;
```



The screenshot shows a database console window with the following SQL query and its results:

```
SELECT MEMBERS.MEMBER_NAME AS Nombre,
EXTRACT(DAY FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Dia,
EXTRACT(MONTH FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Mes,
EXTRACT (YEAR FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Año_Alta
FROM MEMBERS
ORDER BY MEMBERS.MEMBER_DISCHARGE_DATE DESC
LIMIT 20;
```

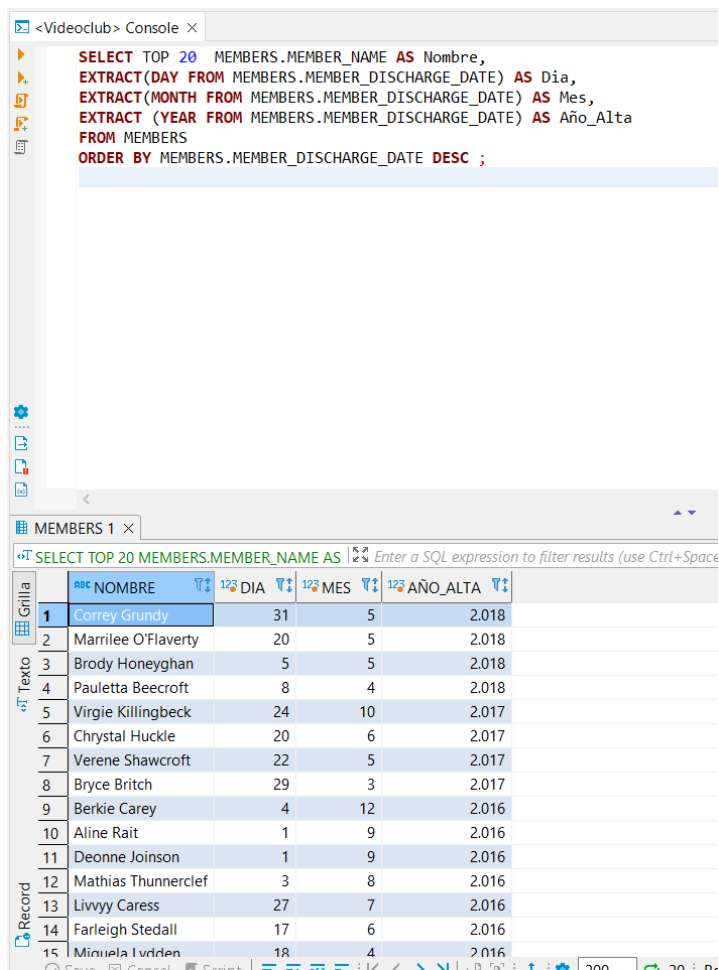
	NOMBRE	DIA	MES	AÑO_ALTA
1	Correy Grundy	31	5	2.018
2	Marrilee O'Flaverty	20	5	2.018
3	Brody Honeyghan	5	5	2.018
4	Pauletta Beecroft	8	4	2.018
5	Virgie Killingbeck	24	10	2.017
6	Chrystal Huckle	20	6	2.017
7	Verene Shawcroft	22	5	2.017
8	Bryce Britch	29	3	2.017
9	Berkie Carey	4	12	2.016
10	Aline Rait	1	9	2.016
11	Deonne Joinson	1	9	2.016
12	Mathias Thunnerclaf	3	8	2.016
13	Livvy Caress	27	7	2.016
14	Farleigh Stedall	17	6	2.016
15	Minuela Luriden	18	4	2.016

The console also shows a status bar at the bottom indicating: Rows: 1, 20 row(s) fetched - 1ms, on may. 22, 12:45:54.

## 1.9.2 Utilizando TOP.

Utilizando la clausula TOP esta aconsejada para tablas largas ¿??? Que diferencia con LIMIT???

```
SELECT TOP 20 MEMBERS.MEMBER_NAME AS Nombre,  
EXTRACT(DAY FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Dia,  
EXTRACT(MONTH FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Mes,  
EXTRACT (YEAR FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Año_Alta  
FROM MEMBERS  
ORDER BY MEMBERS.MEMBER_DISCHARGE_DATE DESC ;
```



The screenshot shows a database console window titled "<Videoclub> Console". The SQL query entered is:

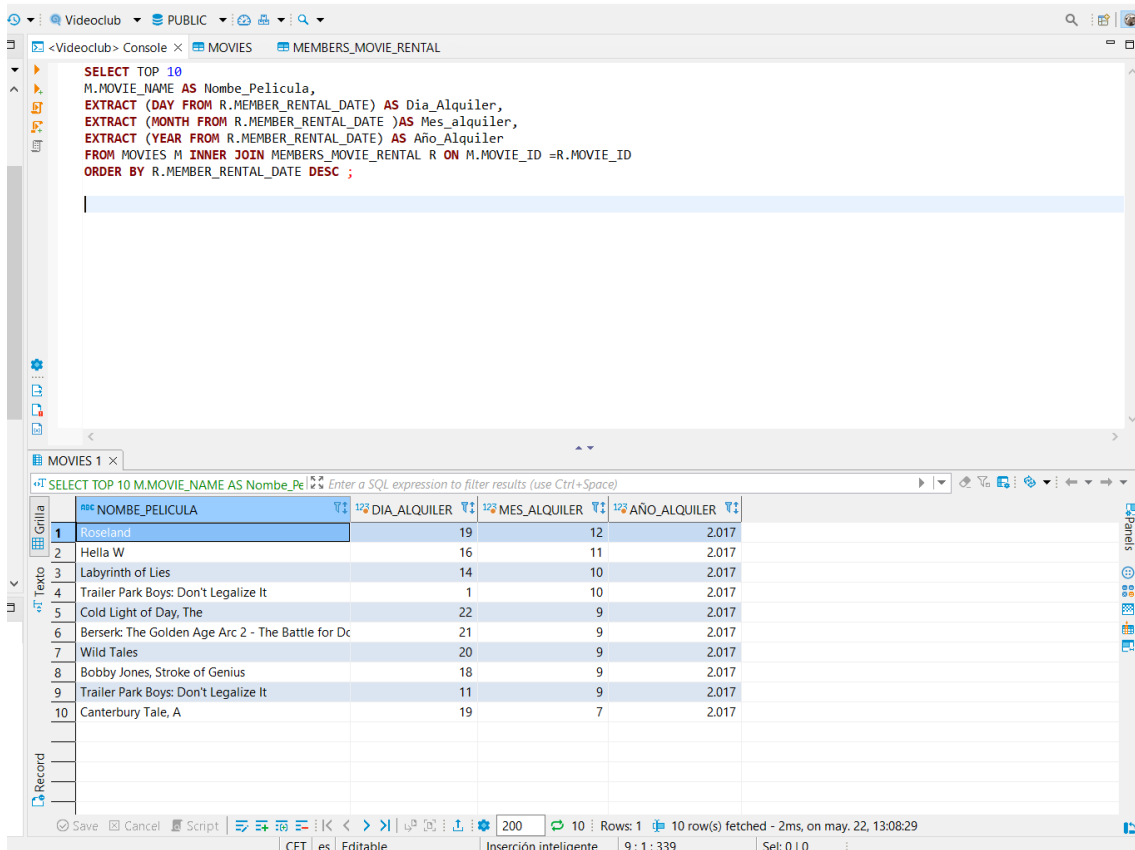
```
SELECT TOP 20 MEMBERS.MEMBER_NAME AS Nombre,  
EXTRACT(DAY FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Dia,  
EXTRACT(MONTH FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Mes,  
EXTRACT (YEAR FROM MEMBERS.MEMBER_DISCHARGE_DATE) AS Año_Alta  
FROM MEMBERS  
ORDER BY MEMBERS.MEMBER_DISCHARGE_DATE DESC ;
```

Below the query, the results are displayed in a table grid. The table has 5 columns: NOMBRE, DIA, MES, and AÑO\_ALTA. The results are ordered by the discharge date in descending order.

	Asc NOMBRE	123 DIA	123 MES	123 AÑO_ALTA
1	Correy Grundy	31	5	2.018
2	Marrilee O'Flaverty	20	5	2.018
3	Brody Honeyghan	5	5	2.018
4	Pauletta Beecroft	8	4	2.018
5	Virgie Killingbeck	24	10	2.017
6	Chrystal Huckle	20	6	2.017
7	Verene Shawcroft	22	5	2.017
8	Bryce Britch	29	3	2.017
9	Berkie Carey	4	12	2.016
10	Aline Rait	1	9	2.016
11	Deonne Joinson	1	9	2.016
12	Mathias Thunnerclaf	3	8	2.016
13	Livvy Caress	27	7	2.016
14	Farleigh Stedall	17	6	2.016
15	Minuela Ivdrden	18	4	2.016

1.10 Devuelve una lista de las últimas 10 películas que se han alquilado.

```
SELECT TOP 10
M.MOVIE_NAME AS Nombre_Pelicula,
EXTRACT (DAY FROM R.MEMBER_RENTAL_DATE) AS Dia_Alquiler,
EXTRACT (MONTH FROM R.MEMBER_RENTAL_DATE )AS Mes_alquiler,
EXTRACT (YEAR FROM R.MEMBER_RENTAL_DATE) AS Año_Alquiler
FROM MOVIES M INNER JOIN MEMBERS_MOVIE_RENTAL R ON M.MOVIE_ID =R.MOVIE_ID
ORDER BY R.MEMBER_RENTAL_DATE DESC ;
```



MOVIES 1 x

SELECT TOP 10 M.MOVIE\_NAME AS Nombre\_Pel... Enter a SQL expression to filter results (use Ctrl+Space)

	MOVIE_NAME	DIA_ALQUILER	MES_ALQUILER	AÑO_ALQUILER
1	Roseland	19	12	2.017
2	Hella W	16	11	2.017
3	Labyrinth of Lies	14	10	2.017
4	Trailer Park Boys: Don't Legalize It	1	10	2.017
5	Cold Light of Day, The	22	9	2.017
6	Berserk: The Golden Age Arc 2 - The Battle for D...	21	9	2.017
7	Wild Tales	20	9	2.017
8	Bobby Jones, Stroke of Genius	18	9	2.017
9	Trailer Park Boys: Don't Legalize It	11	9	2.017
10	Canterbury Tale, A	19	7	2.017

Save Cancel Script 200 10 Rows: 1 10 row(s) fetched - 2ms, on may.22, 13:08:29

CET es Editable Inserción inteligente 9 : 1 : 339 Sel: 0 | 0

```
SELECT
M.MOVIE_NAME AS Nombre_Pelicula,
EXTRACT (DAY FROM R.MEMBER_RENTAL_DATE) AS Dia_Alquiler,
EXTRACT (MONTH FROM R.MEMBER_RENTAL_DATE )AS Mes_alquiler,
EXTRACT (YEAR FROM R.MEMBER_RENTAL_DATE) AS Año_Alquiler
FROM MOVIES M INNER JOIN MEMBERS_MOVIE_RENTAL R ON M.MOVIE_ID =R.MOVIE_ID
ORDER BY R.MEMBER_RENTAL_DATE DESC
LIMIT 10;
```

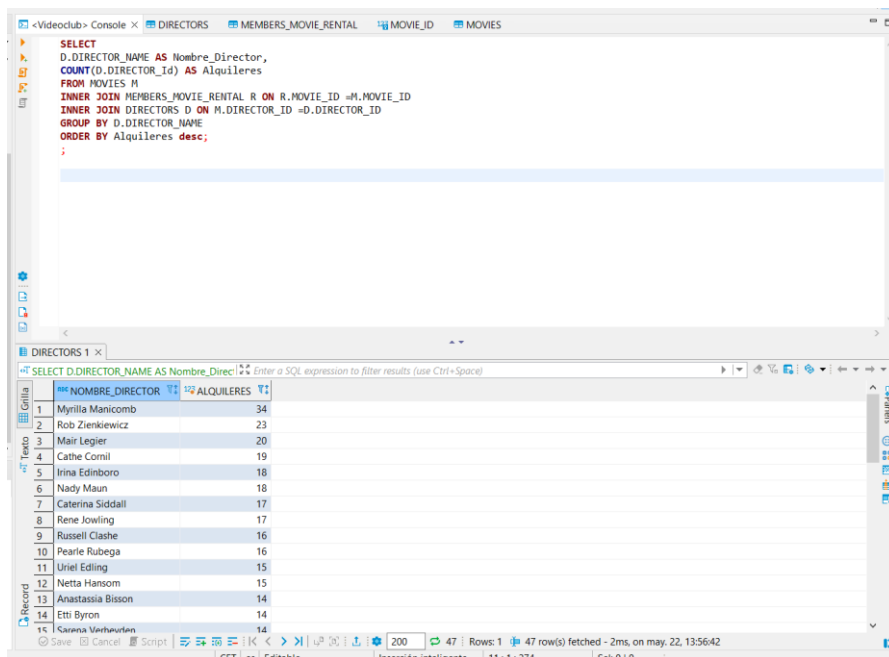
Produce la misma salida.

## 1.11 Indica cual es el nombre del director del que más películas se han alquilado

### SELECT

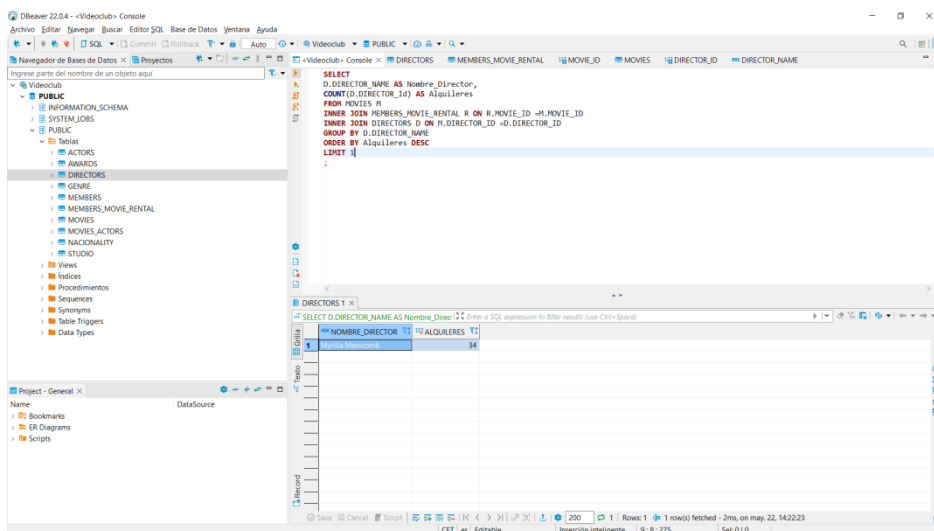
```
D.DIRECTOR_NAME AS Nombre_Director,  
COUNT(D.DIRECTOR_Id) AS Alquileres  
FROM MOVIES M  
INNER JOIN MEMBERS_MOVIE_RENTAL R ON R.MOVIE_ID =M.MOVIE_ID  
INNER JOIN DIRECTORS D ON M.DIRECTOR_ID =D.DIRECTOR_ID  
GROUP BY D.DIRECTOR_NAME  
ORDER BY Alquileres desc;
```

Añadiendo Limit 1 , ¿???No coincide con el campus Se comprueba con excel



```
SELECT  
D.DIRECTOR_NAME AS Nombre_Director,  
COUNT(D.DIRECTOR_Id) AS Alquileres  
FROM MOVIES M  
INNER JOIN MEMBERS_MOVIE_RENTAL R ON R.MOVIE_ID =M.MOVIE_ID  
INNER JOIN DIRECTORS D ON M.DIRECTOR_ID =D.DIRECTOR_ID  
GROUP BY D.DIRECTOR_NAME  
ORDER BY Alquileres desc;  
;
```

	NOMBRE_DIRECTOR	ALQUILERES
1	Myrilla Manicomb	34
2	Rob Zienkiewicz	23
3	Mair Legier	20
4	Cathe Cornil	19
5	Irina Edinboro	18
6	Nady Maun	18
7	Caterina Siddall	17
8	Rene Jowling	17
9	Russell Clashe	16
10	Pearlie Rubega	16
11	Uriel Edling	15
12	Netta Hansom	15
13	Anastassia Bisson	14
14	Etti Byron	14
15	Karena Vanhaurien	14



```
SELECT  
D.DIRECTOR_NAME AS Nombre_Director,  
COUNT(D.DIRECTOR_Id) AS Alquileres  
FROM MOVIES M  
INNER JOIN MEMBERS_MOVIE_RENTAL R ON R.MOVIE_ID =M.MOVIE_ID  
INNER JOIN DIRECTORS D ON M.DIRECTOR_ID =D.DIRECTOR_ID  
GROUP BY D.DIRECTOR_NAME  
ORDER BY Alquileres DESC  
LIMIT 1  
;
```

	NOMBRE_DIRECTOR	ALQUILERES
1	Myrilla Manicomb	34



1.12 Indica cuantos premios han ganado cada uno de los estudios con las películas que han creado.

No entiendo la tabla de premios

1.13 Indica cuántas películas ha realizado cada director antes de cumplir 41 años.

Utilizamos la función [DATEADD](#) para calcular la fecha a los 41 cumpleaños, no comprobamos si está muerto por consistencia de Base de datos.

## SQL Server DATEADD() Function

< Previous

< SQL Server Functions

### Example

Add one year to a date, then return the date:

```
SELECT DATEADD(year, 1, '2017/08/25') AS DateAdd;
```

Try it Yourself >

### Definition and Usage

The DATEADD() function adds a time/date interval to a date and then returns the date.

### Syntax

```
DATEADD(interval, number, date)
```

### Parameter Values

## SELECT

```
D.DIRECTOR_NAME AS Nombre_Director,
--DATEADD(YEAR,41,D.DIRECTOR_BIRTH_DATE) AS Fecha_41_Cumpleaños,
COUNT(M.MOVIE_NAME) AS n_Peliculas
--M.MOVIE_LAUNCH_DATE AS Lanzamiento
FROM DIRECTORS D
INNER JOIN MOVIES M ON D.DIRECTOR_ID =M.DIRECTOR_ID
WHERE M.MOVIE_LAUNCH_DATE < DATEADD(YEAR,41,D.DIRECTOR_BIRTH_DATE)
GROUP BY D.DIRECTOR_NAME
ORDER BY N_Peliculas DESC ;
```

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL query being executed. The bottom pane shows the results of the query in a table format.

DIRECTOR_NAME	n_PELICULAS
Uwehl Edling	6
Nady Maun	6
Netta Hansom	5
Elle Byrn	5
Anastasia Bisson	4
Lamont Duffield	4
Violetta Cotlard	4
Irma Edinboro	3
Myrtila Manicomb	3
Joella Leaver	3
Jourdan Bothen	3
Sanna Verheyden	3
Russell Clache	3
Clarence Farnow	1

## 1.14 Indica la edad media de los directores vivos.

La función [AVG](#) calcula la media de los datos .

### SQL Server AVG() Function

[< Previous](#)[< SQL Server Functions](#)

#### Example

Return the average value for the "Price" column in the "Products" table:

```
SELECT AVG(Price) AS AveragePrice FROM Products;
```

[Try it Yourself »](#)

### Definition and Usage

The AVG() function returns the average value of an expression.

**Note:** NULL values are ignored.

### Syntax

```
AVG(expression)
```

```
SELECT
AVG(DATEDIFF( now(),DIRECTORS.DIRECTOR_BIRTH_DATE)/365) AS media_Edad_Vivos
FROM DIRECTORS
WHERE DIRECTOR_DEAD_DATE IS NULL
;
```



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the following SQL query:

```
SELECT
AVG(DATEDIFF( now(),DIRECTORS.DIRECTOR_BIRTH_DATE)/365) AS media_Edad_Vivos
FROM DIRECTORS
WHERE DIRECTOR_DEAD_DATE IS NULL
;
```

The bottom pane shows the results of the query in a grid format:

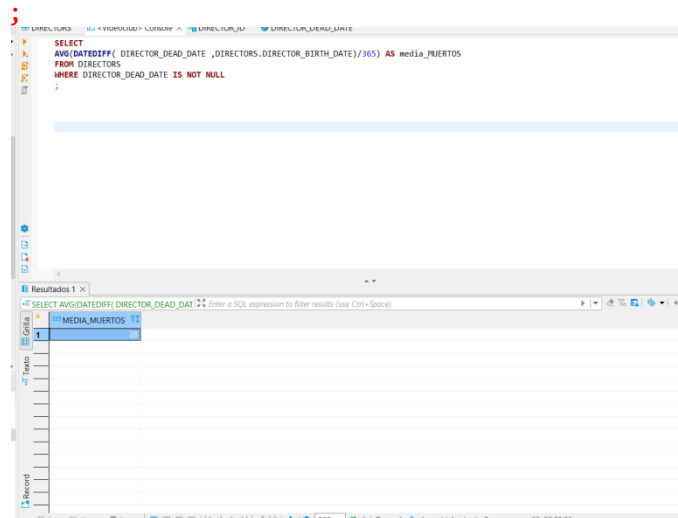
	media_Edad_Vivos
1	46

Tiene sentido con el resultado bootcamp 46 el año pasado

### 1.15 Indica la edad media de los actores que han fallecido.

Solo cambiamos la condición de where por IS NOT NULL (si la fecha de muerte no es nula viene a decir que están muertos). 6 muertos

```
SELECT
AVG(DATEDIFF( DIRECTOR_DEAD_DATE ,DIRECTORS.DIRECTOR_BIRTH_DATE)/365) AS
media_MUERTOS
FROM DIRECTORS
WHERE DIRECTOR_DEAD_DATE IS NOT NULL
;
```



No coincide con el BOOTCAM 32

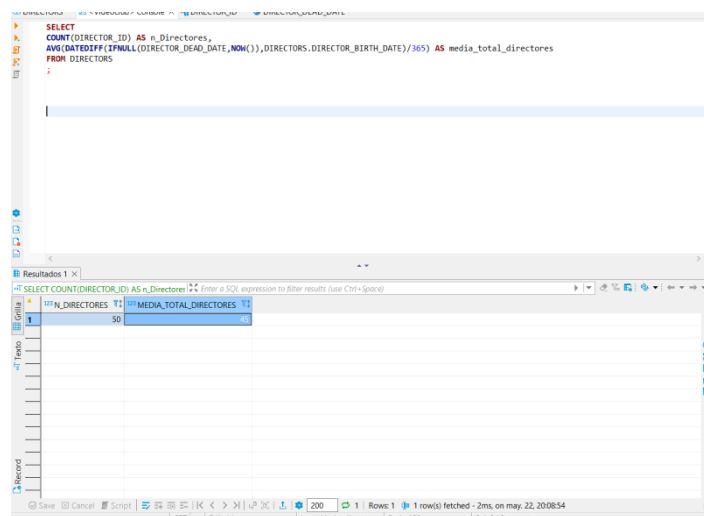
#### 1.15.1 Comprobación Manual

Resultado 44 vivos con edad media 47 y 6 muertos con edad media de 28

La media ponderada será  $(44 \times 47 + 6 \times 28) / 50 = (2068 + 168) \text{ años} / 50 \text{ actores} = 44,72 \text{ aprox } 45$

#### 1.15.2 Comprobación Manual

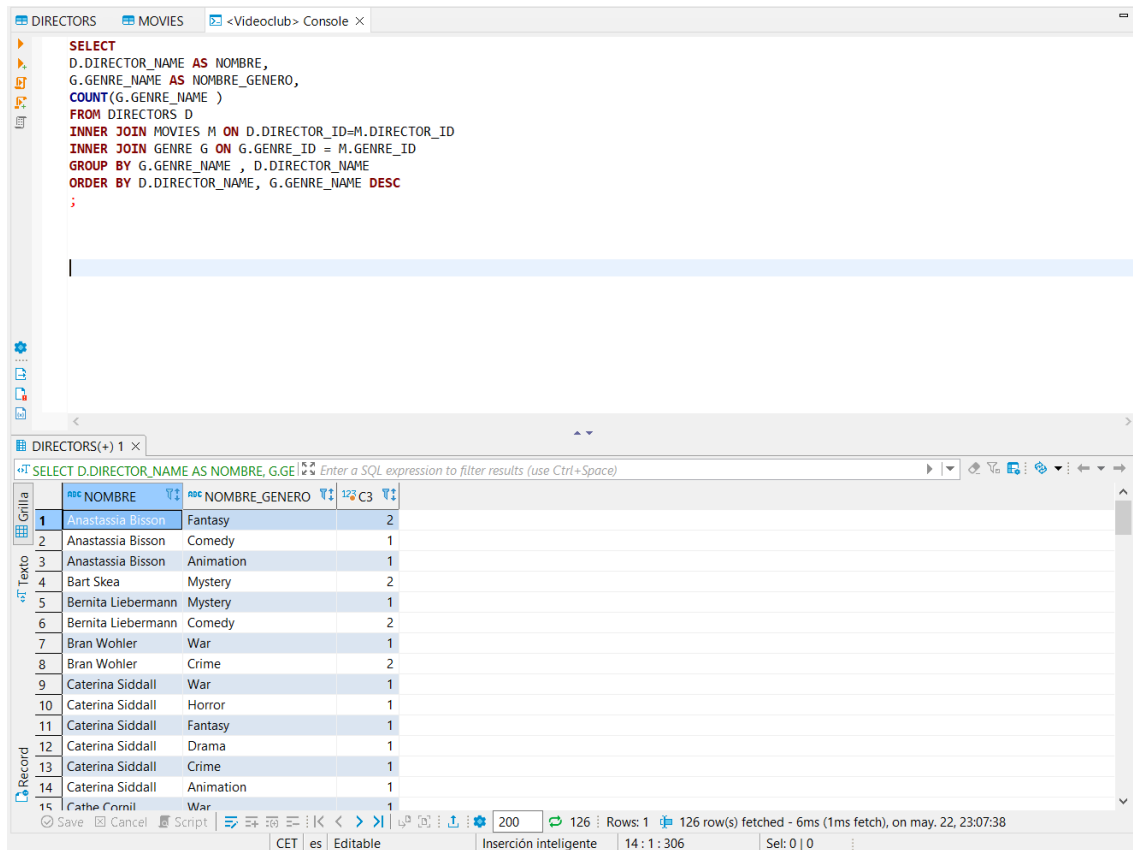
```
SELECT
COUNT(DIRECTOR_ID) AS n_Directores,
AVG(DATEDIFF(IFNULL(DIRECTOR_DEAD_DATE, NOW()), DIRECTORS.DIRECTOR_BIRTH_DATE)/
365) AS media_total_directores
FROM DIRECTORS
;
```



## 1.16 Indica cuál es el género favorito de cada uno de los directores cuando dirigen una película

Falta filtra por la primera fila de cada director.

```
SELECT
D.DIRECTOR_NAME AS NOMBRE,
G.GENRE_NAME AS NOMBRE_GENERO,
COUNT(G.GENRE_NAME )
FROM DIRECTORS D
INNER JOIN MOVIES M ON D.DIRECTOR_ID=M.DIRECTOR_ID
INNER JOIN GENRE G ON G.GENRE_ID = M.GENRE_ID
GROUP BY G.GENRE_NAME , D.DIRECTOR_NAME
ORDER BY D.DIRECTOR_NAME, G.GENRE_NAME DESC
;
```



The screenshot shows a database management tool interface. The top pane displays the following SQL query:

```
SELECT
D.DIRECTOR_NAME AS NOMBRE,
G.GENRE_NAME AS NOMBRE_GENERO,
COUNT(G.GENRE_NAME )
FROM DIRECTORS D
INNER JOIN MOVIES M ON D.DIRECTOR_ID=M.DIRECTOR_ID
INNER JOIN GENRE G ON G.GENRE_ID = M.GENRE_ID
GROUP BY G.GENRE_NAME , D.DIRECTOR_NAME
ORDER BY D.DIRECTOR_NAME, G.GENRE_NAME DESC
;
```

The bottom pane shows the results of the query in a table. The table has three columns: NOMBRE, NOMBRE\_GENERO, and COUNT. The data is sorted by director name and then by genre name in descending order of count.

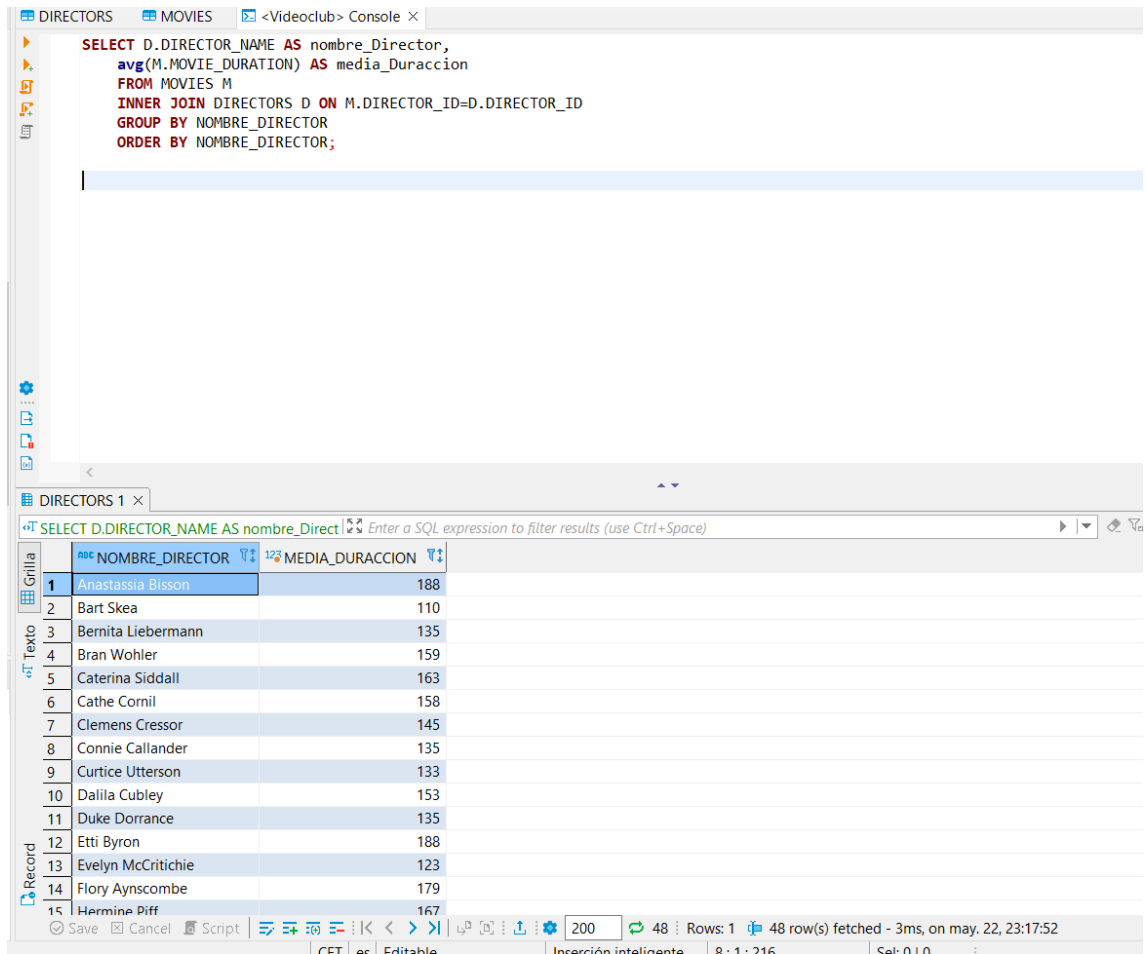
	NOMBRE	NOMBRE_GENERO	COUNT
1	Anastassia Bisson	Fantasy	2
2	Anastassia Bisson	Comedy	1
3	Anastassia Bisson	Animation	1
4	Bart Skea	Mystery	2
5	Bernita Liebermann	Mystery	1
6	Bernita Liebermann	Comedy	2
7	Bran Wohler	War	1
8	Bran Wohler	Crime	2
9	Caterina Siddall	War	1
10	Caterina Siddall	Horror	1
11	Caterina Siddall	Fantasy	1
12	Caterina Siddall	Drama	1
13	Caterina Siddall	Crime	1
14	Caterina Siddall	Animation	1
15	Cathe Cornil	War	1

The status bar at the bottom indicates that 126 rows were fetched, and the query was executed on May 22, 23:07:38.

1.17 Indica cuál es la nacionalidad favorita de cada uno de los estudios en la producción de las películas

1.18 Indica cuál es la media de duración de las películas de cada director.

```
SELECT D.DIRECTOR_NAME AS nombre_Director,  
       avg(M.MOVIE_DURATION) AS media_Duracion  
FROM MOVIES M  
INNER JOIN DIRECTORS D ON M.DIRECTOR_ID=D.DIRECTOR_ID  
GROUP BY NOMBRE_DIRECTOR  
ORDER BY NOMBRE_DIRECTOR;
```



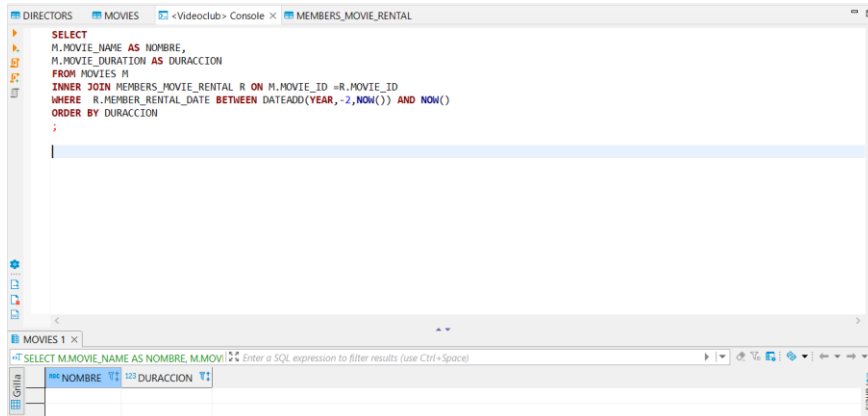
The screenshot shows a database management tool interface. The top pane displays the SQL query used to calculate the average movie duration for each director. The bottom pane shows the results of the query in a table format. The table has two columns: NOMBRE\_DIRECTOR and MEDIA\_DURACION. The results are sorted by director name.

	NOMBRE_DIRECTOR	MEDIA_DURACION
1	Anastassia Bisson	188
2	Bart Skea	110
3	Bernita Liebermann	135
4	Bran Wohler	159
5	Caterina Siddall	163
6	Cathe Cornil	158
7	Clemens Cressor	145
8	Connie Callander	135
9	Curtice Utterson	133
10	Dalila Cubley	153
11	Duke Dorrance	135
12	Etti Byron	188
13	Evelyn McCritchie	123
14	Flory Aynscombe	179
15	Hermine Piff	167

1.19 Indica cuál es la el nombre y la duración mínima de las películas que han sido alquiladas en los últimos 2 años por los miembros del videoclub

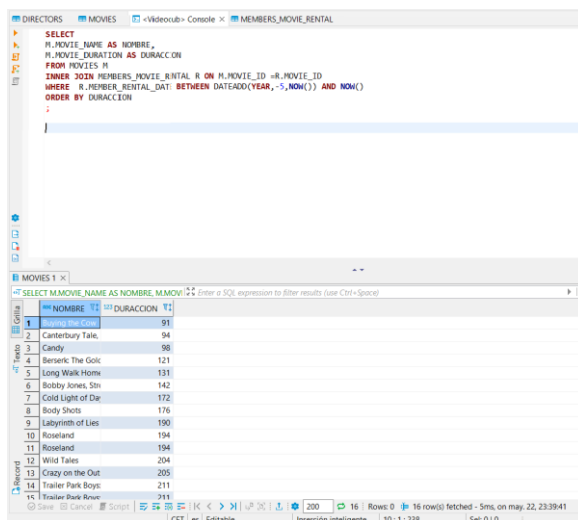
En los últimos dos años el resultado es 0.

La clausula BETWEEN debe estar ordenada de menor a mayor fecha



Encontramos que el ultimo año con resultados es hace 5 años (2017)

```
SELECT
M.MOVIE_NAME AS NOMBRE,
M.MOVIE_DURATION AS DURACCION
FROM MOVIES M
INNER JOIN MEMBERS_MOVIE_RENTAL R ON M.MOVIE_ID =R.MOVIE_ID
WHERE R.MEMBER_RENTAL_DATE BETWEEN DATEADD(YEAR,-5,NOW()) AND NOW()
ORDER BY DURACCION
;
```



Otra manera seria con la fecha literal:

```
SELECT
M.MOVIE_NAME AS NOMBRE,
M.MOVIE_DURATION AS DURACCION
FROM MOVIES M
INNER JOIN MEMBERS_MOVIE_RENTAL R ON M.MOVIE_ID =R.MOVIE_ID
WHERE R.MEMBER_RENTAL_DATE >='2017-01-01'
ORDER BY DURACCION
;
```

1.20 Indica cuál fue la primera película que alquilaron los miembros del videoclub cuyos teléfonos tengan como último dígito el ID de alguna nacionalidad



- 1.21 Indica el número de premios a los que estuvo nominado un actor, pero que no ha conseguido (Si una película está nominada a un premio, su actor también lo está)
- 1.22 Indica cuantos actores y directores hicieron películas para los estudios no activos
- 1.23 Indica el nombre, ciudad, y teléfono de todos los miembros del videoclub que hayan alquilado películas que hayan sido nominadas a más de 150 premios y ganaran menos de 50
- 1.24 Indica el número de películas que hayan hecho los directores durante las décadas de los 60, 70 y 80 que contengan la palabra "The" en cualquier parte del título
- 1.25 Indica si hay alguna coincidencia de nacimiento de ciudad (y si las hay, indicarlás) entre los miembros del videoclub y los directores.
- 1.26 Comprueba si hay errores en la BD entre las películas y directores (un director muerto en el 76 no puede dirigir una película en el 88)
- 1.27 Usando como condición la sentencia anterior, modifica la fecha de defunción a un año más tarde del estreno de la película (mediante sentencia SQL)