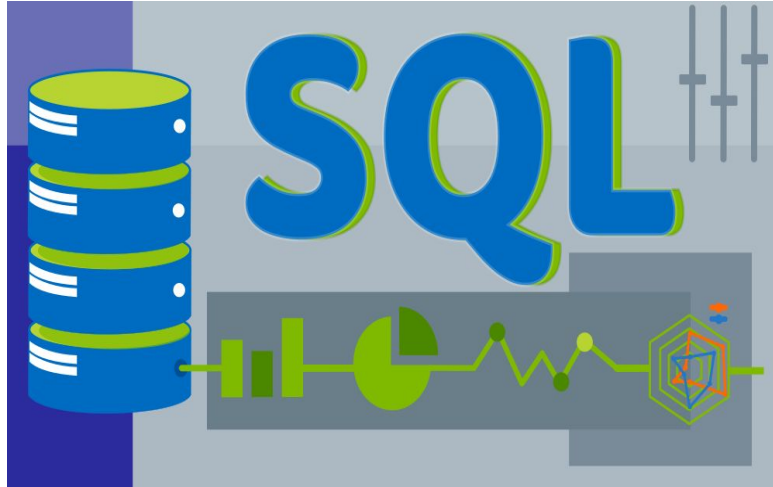


LENGUAJE DE DATOS



LENGUAJE DE DATOS



LENGUAJE SQL



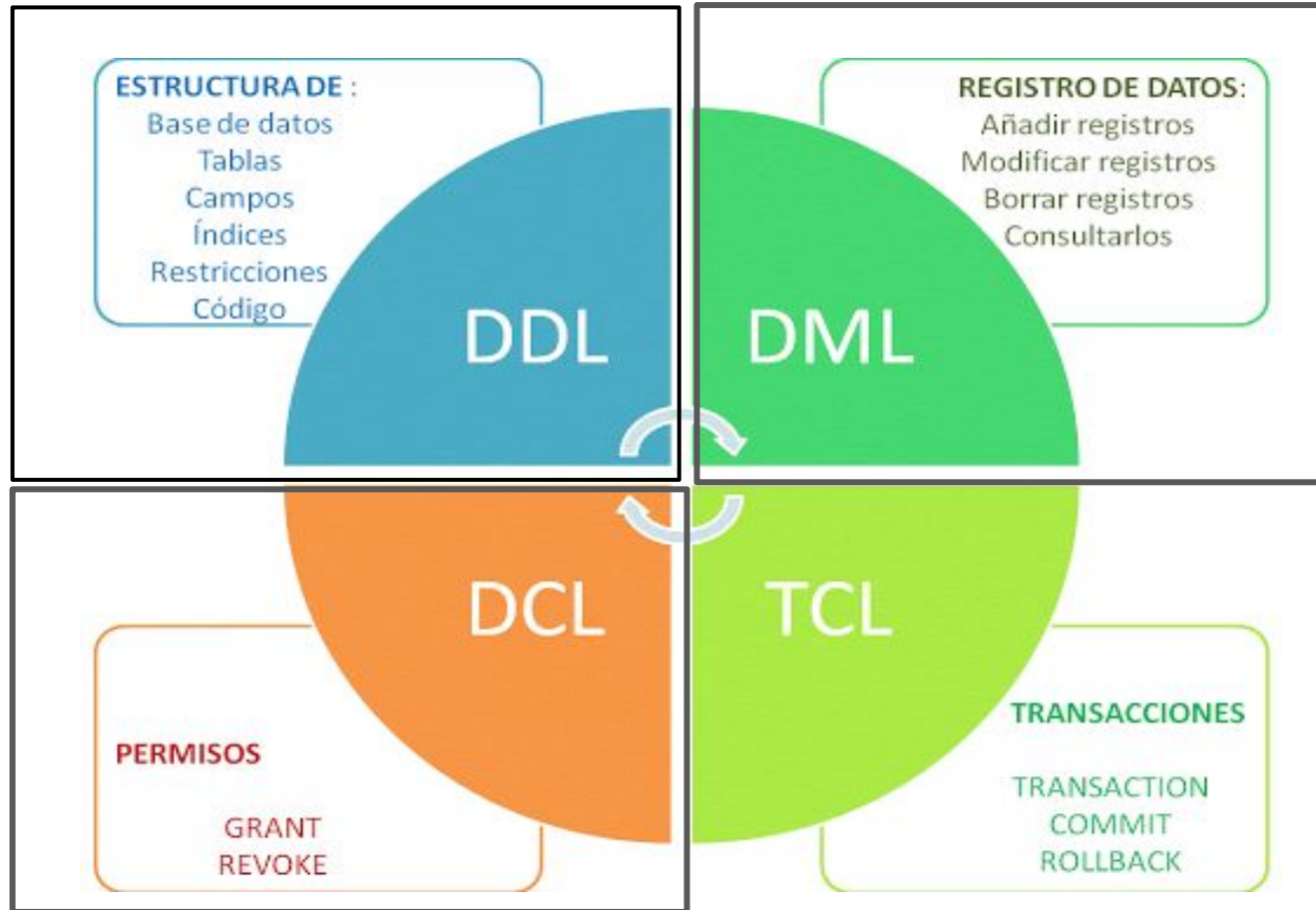
SISTEMA GESTOR DE BASES DE DATOS



ESTÁNDARES POR LOS QUE RIGEN LAS BASES DE DATOS

1. ARQUITECTURA ANSI
2. NORMA ISO /IEC 2700
3. LEY ORGÁNICA DE PROTECCIÓN DE DATOS

EL LENGUAJE SQL SE DIVIDE EN 4 GRUPOS (DDL Y DML, DCL son los que veremos)



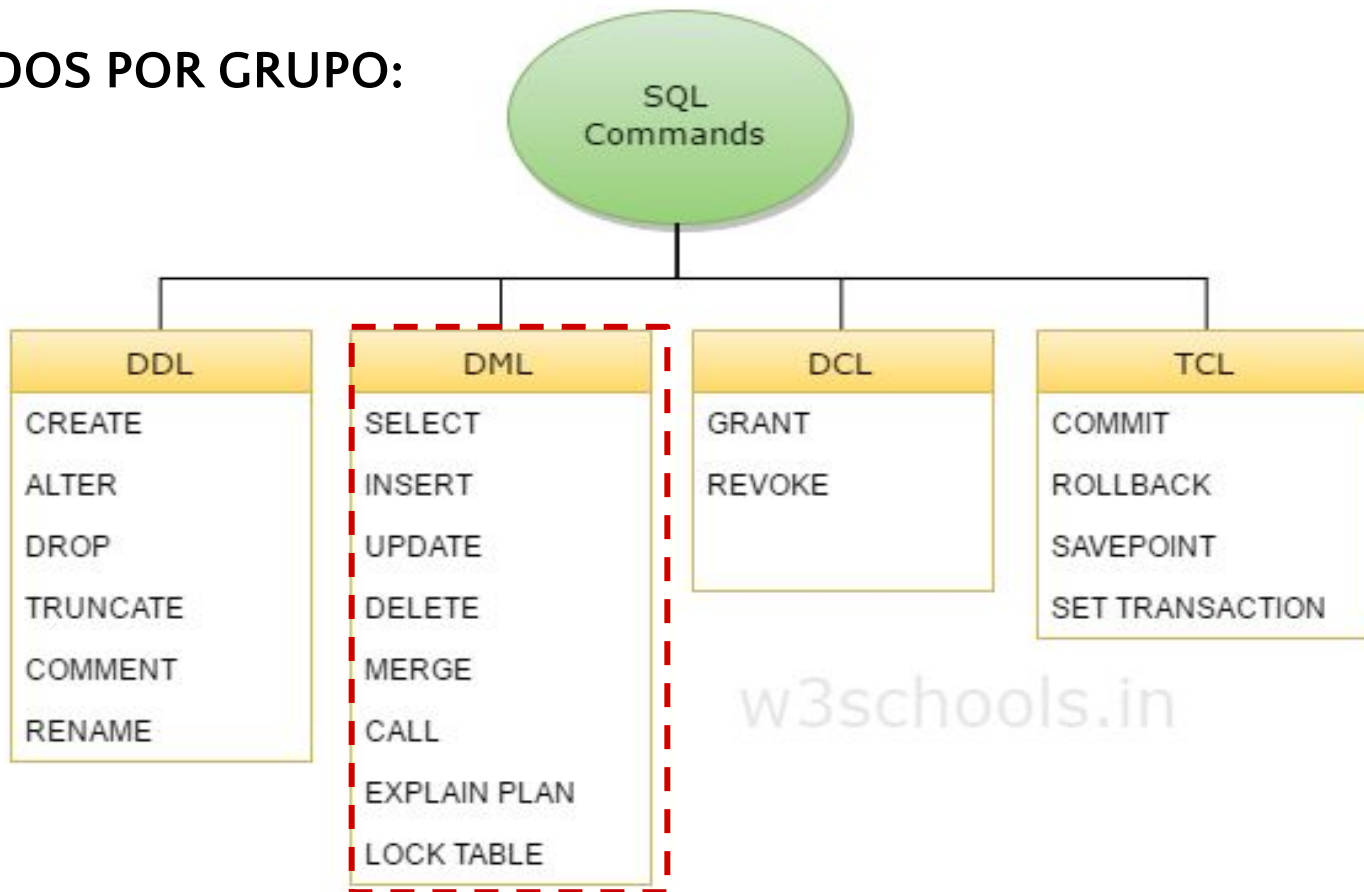


QUE ES EL LENGUAJE DE DEFINICIÓN DE DATOS:

Data Manipulation Language (DML) es un subconjunto de [SQL](#) (Structured Query Language) que se utiliza para gestionar y manipular datos en una base de datos relacional

Acción financiada por el SCE y por el Servicio Público de Empleo Estatal

COMANDOS POR GRUPO:





1. QUE ES UN GESTOR DE BASES DE DATOS

Los gestores de base de datos, conocidos también como Data Base Management System (DBMS), son sistemas informáticos que te ayudarán a crear y gestionar de manera eficaz tus bases de datos. La información que llega a tus manos debe ser almacenada adecuadamente en una base de datos

<https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/sistema-gestor-de-base-de-datos-sqbd/>



QUE ES UN SCRIPT:



Es un **programa**, o sea un conjunto de **comandos**, que se le da a un motor SQL para decirle lo que debe hacer y en qué orden debe hacerlo. ¿Cómo se escribe un script? Como un archivo de texto plano, o sea sin negritas, ni subrayados, nada de eso. Por ejemplo, puedes usar el Bloc de Notas para escribirlo.

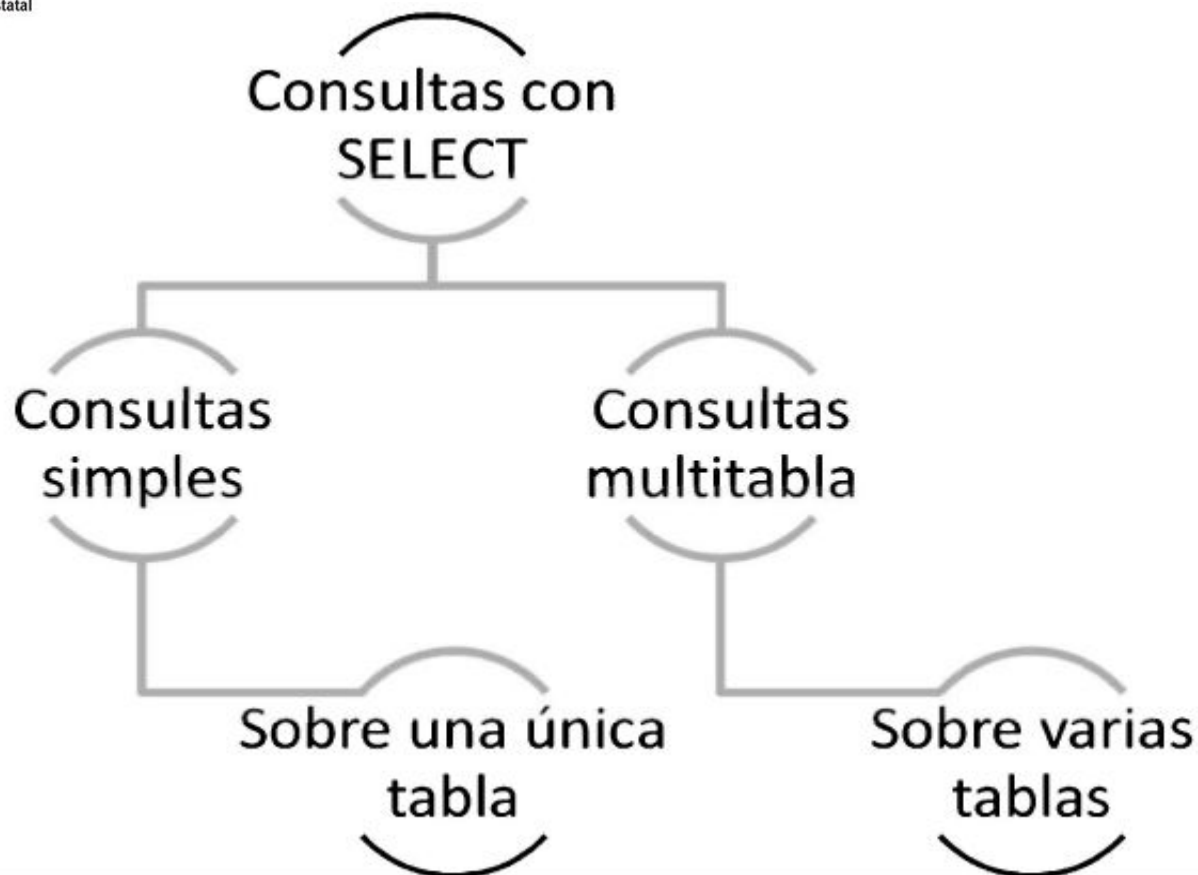
```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,  
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,  
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE=NO_AUTO_CREATE_INDEX;  
  
DROP SCHEMA IF EXISTS consult;  
CREATE SCHEMA consult;  
USE consult;  
  
CREATE TABLE address (  
    address_id INTEGER NOT NULL AUTO_INCREMENT,  
    line1 VARCHAR(50) NOT NULL,  
    line2 VARCHAR(50) NULL,  
    city VARCHAR(50) NOT NULL,  
    region VARCHAR(50) NOT NULL,  
    country VARCHAR(50) NOT NULL,  
    postal_code VARCHAR(50) NOT NULL,  
    CONSTRAINT address_pk PRIMARY KEY (address_id))
```




<https://www.w3schools.com/sql/>

Esta web nos ayuda a aprender a utilizar los comandos para poder trabajar con una base de datos.

Acción financiada por el SCE y por el Servicio Público de Empleo Estatal



Las consultas multitable:

Vistas las diferentes consultas que podemos realizar sobre una tabla, vamos a ver ahora los protocolos de consulta cuando queremos hacer una búsqueda basada en datos que están contenidos en diferentes tablas de la base de datos. Las operaciones del álgebra relacional que se van a utilizar en las consultas multitable son:

UNICO, EXCEPT, INTERSECT, CROSS JOIN, INNER JOIN, LEFT JOIN, RIGHT JOIN y FULL JOIN.

SIGNIFICADO S.Q.L:

Structured Query Language, en español lenguaje de consulta estructurada que interactúa con bases de datos relacionales.

Nota: Fue inventado por IBM





ADEMÁS DE CONSULTAS EL LENGUAJE NOS PERMITE:

- CREAR BASES DE DATOS DE CERO
- CREAR CAMPOS ETC.
- ELIMINAR Y MODIFICAR
- DEFINIR TIPOS DE DATOS, ESTABLECER RELACIONES

ESTÁNDAR SQL.

SQL, comenzó a convertirse en un lenguaje ESTÁNDAR en los diversos SGBD, logrando así que en 1986 sea estandarizado por ANSI, dando lugar a la primera versión estándar de este lenguaje: “SQL-86” o “SQL1”. ANSI SQL ha sufrido varias revisiones y agregados a lo largo del tiempo. Actualmente cuenta con la versión **SQL:2016**.



Lenguajes de definición de datos

DDL (Data Definition Language, en español Lenguaje de Definición de Datos): se utilizan para crear y modificar la estructura de una base de datos.

- CREATE
- ALTER
- DROP
- TRUNCATE

DML (Data Manipulation Language, en español Lenguaje de Manipulación de Datos): se utilizan para seleccionar, insertar, actualizar y borrar de manera definitiva, registros de una base de datos (consultas de selección y acción).

- SELECT
- INSERT
- UPDATE
- DELETE



Lenguajes de definición de datos

DCL (Data Control Language):

Proporcionan seguridad a la información en la base de datos.

- GRANT
- REVOKE

TCL (Transaction Control lenguaje):

Se preocupan de la gestión de cambios en los datos.

- COMMIT
- ROLLBACK
- SAVEPOINT



CLÁUSULAS

FROM
WHERE
GROUP BY
HAVING
ORDER BY

Instrucción SQL:

Una instrucción es la unión entre comandos, cláusulas, operadores y funciones de agregado, es decir:

Instrucción SQL = Comando + Cláusula + Operadores + Funciones de agregado

<https://www.w3schools.com/sql/>



Cláusulas SQL:

SELECT: Es una orden que realiza peticiones a las tablas trabajando con las demás que observamos aquí debajo.

FROM: especifica la tabla de la que se quieren obtener los registros.

WHERE(Donde): especifica las condiciones o criterios de los registros seleccionados.

GROUP BY: para agrupar los registros seleccionados en función de un campo.

HAVING(teniendo): especifica las condiciones o criterios que deben cumplir los grupos.

ORDER BY: ordena los registros seleccionados en función de un campo.



LOS OPERADORES

Comparación estándar:

Esta condición básica de búsqueda compara el valor de dos expresiones. Para comparar las expresiones utilizaremos los operadores de comparación.

Operadores de Comparación	
Operador	Uso
<	Menor que
>	Mayor que
≠	Distinto de
<=	Menor o igual que
>=	Mayor o igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo
In	Utilizado para especificar registros de una base de datos

OPERADORES LÓGICOS

Operadores Lógicos	
Operador	Uso
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

SINTAXIS DE LA INSTRUCCIÓN SELECT

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

```
SELECT * FROM table_name;
```

SELECT se utiliza para seleccionar datos de una base de datos.



UTILIZANDO SELECT

Forma 1: `SELECT * FROM sakila.actor;`

Forma 2: `SELECT * FROM ACTOR;`

```
1 • use sakila;
2 • SELECT * from actor
3
```

	actor_id	first_name	last_name	last_update
▶	1	PENELOPE	GUINNESS	2006-02-15 04:34:33
	2	NICK	WAHLBERG	2006-02-15 04:34:33
	3	ED	CHASE	2006-02-15 04:34:33
	4	JENNIFER	DAVIS	2006-02-15 04:34:33
	5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
	6	BETTE	NICHOLSON	2006-02-15 04:34:33
	7	GRACE	MOSTEL	2006-02-15 04:34:33
	8	MATTHEW	JOHANSSON	2006-02-15 04:34:33

```
1 • SELECT * FROM sakila.actor;
```

	actor_id	first_name	last_name	last_update
▶	1	PENELOPE	GUINNESS	2006-02-15 04:34:33
	2	NICK	WAHLBERG	2006-02-15 04:34:33
	3	ED	CHASE	2006-02-15 04:34:33
	4	JENNIFER	DAVIS	2006-02-15 04:34:33
	5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
	6	BETTE	NICHOLSON	2006-02-15 04:34:33
	7	GRACE	MOSTEL	2006-02-15 04:34:33
	8	MATTHEW	JOHANSSON	2006-02-15 04:34:33
	9	JOE	SWANK	2006-02-15 04:34:33
	10	CHRISTIAN	GABLE	2006-02-15 04:34:33

UTILIZANDO SELECT

```
SELECT actor_id, first_name FROM sakila.actor;
```

```
1 • SELECT actor_id, first_name FROM sakila.actor;
```

```
2
```

<

Result Grid



Filter Rows:

Edit:



Export/Import:

	actor_id	first_name
▶	1	PENELOPE
	2	NICK
	3	ED
	4	JENNIFER
	5	JOHNNY
	6	DESSA

UTILIZANDO SELECT AS

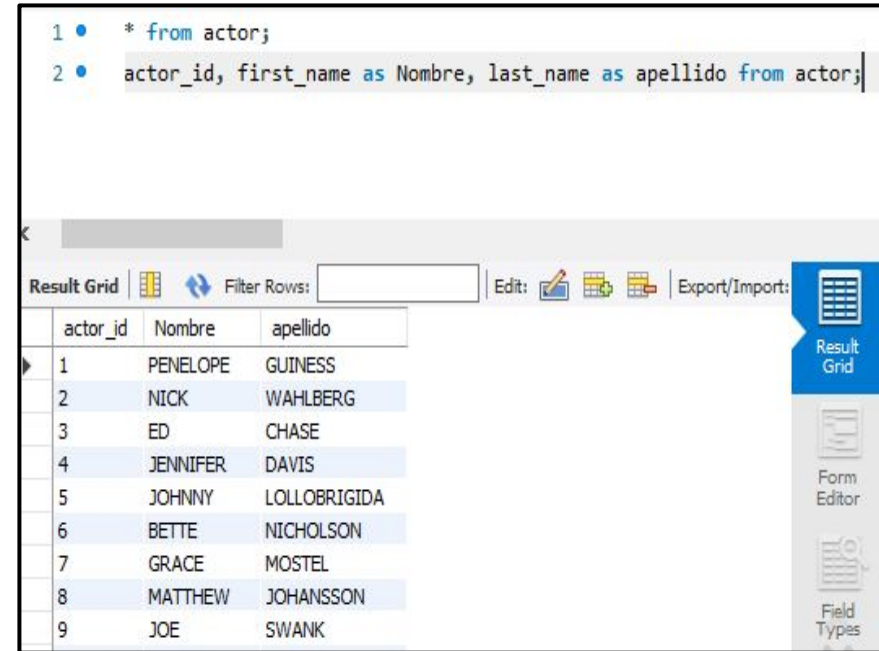
COMODÍN AS: El comando AS se utiliza para cambiar el nombre de una columna o tabla con un alias.

Solo existe un alias durante la duración de la consulta.

SELECT y AS, para también renombrar columnas durante la consulta

select * from actor; (Veamos cómo se escriben)

select actor_id, first_name as Nombre, last_name as apellido from actor;



The screenshot shows a SQL query editor with two lines of code:

```
1 • * from actor;  
2 • actor_id, first_name as Nombre, last_name as apellido from actor;
```

Below the query editor, there is a toolbar with options: Result Grid, Filter Rows, Edit, and Export/Import. The Result Grid is selected, displaying a table with the following data:

	actor_id	Nombre	apellido
1	1	PENELOPE	GUINESS
2	2	NICK	WAHLBERG
3	3	ED	CHASE
4	4	JENNIFER	DAVIS
5	5	JOHNNY	LOLLOBRIGIDA
6	6	BETTE	NICHOLSON
7	7	GRACE	MOSTEL
8	8	MATTHEW	JOHANSSON
9	9	JOE	SWANK

On the right side of the interface, there are buttons for Result Grid, Form Editor, and Field Types.

UTILIZANDO DISTINCT

DISTINCT. Devuelve solo valores no repetidos (diferentes) en el conjunto de resultados. descarta elementos repetidos

VEAMOS:

```
SELECT amount FROM payment;
```

```
select * from payment;
```

```
SELECT DISTINCT amount FROM payment;
```

ORDER BY

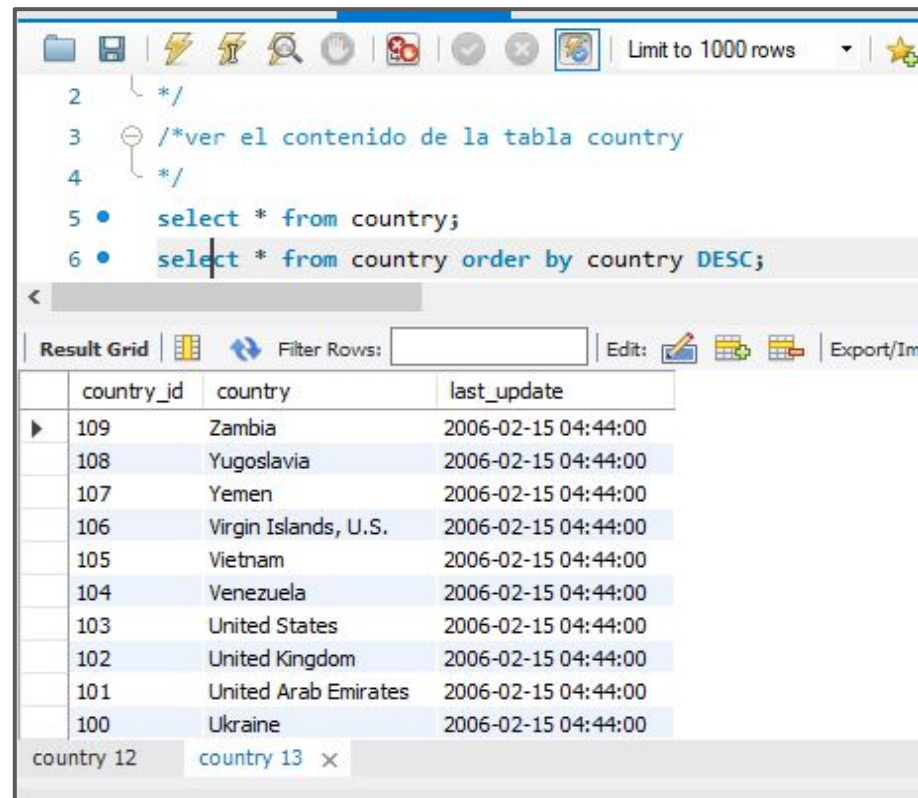
PODEMOS ORDENAR DE MANERA ASCENDENTE (ASC) O DESCENDENTE (DESC).

SINTAXIS:

SELECT columna
FROM tabla
ORDER BY
columna

Vamos a ordenar la tabla país de manera descendente.

```
select * from country order by country desc;
```



The screenshot shows a database query editor interface. The top toolbar includes icons for file operations, execution, and search. The query editor displays a SQL query with line numbers 2 through 6. The query is: `select * from country order by country DESC;`. Below the query editor, the 'Result Grid' tab is active, showing a table with 4 columns: `country_id`, `country`, and `last_update`. The table contains 12 rows of data, ordered by country name in descending order. The bottom status bar shows 'country 12' and 'country 13 x'.

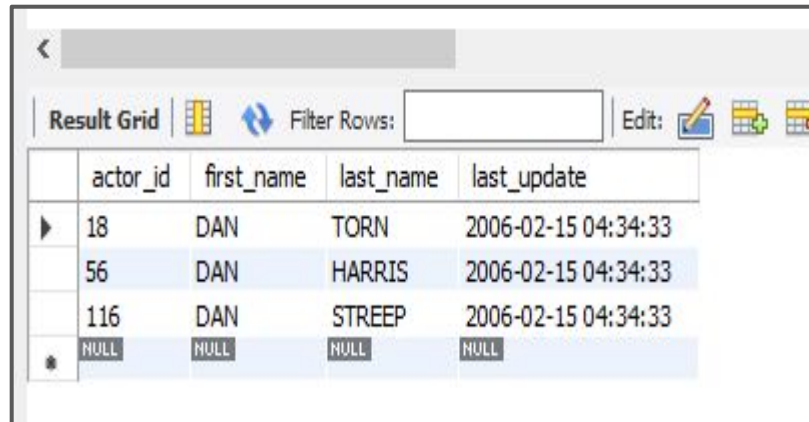
	country_id	country	last_update
▶	109	Zambia	2006-02-15 04:44:00
	108	Yugoslavia	2006-02-15 04:44:00
	107	Yemen	2006-02-15 04:44:00
	106	Virgin Islands, U.S.	2006-02-15 04:44:00
	105	Vietnam	2006-02-15 04:44:00
	104	Venezuela	2006-02-15 04:44:00
	103	United States	2006-02-15 04:44:00
	102	United Kingdom	2006-02-15 04:44:00
	101	United Arab Emirates	2006-02-15 04:44:00
	100	Ukraine	2006-02-15 04:44:00

CONSULTA CON CLÁUSULA CONDICIONAL WHERE

La cláusula WHERE se utiliza para seleccionar aquellos registros que cumplen una o más condiciones

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

```
select * from actor WHERE  
first_name="DAN";
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of a query, showing columns 'actor_id', 'first_name', 'last_name', and 'last_update'. The results are filtered to show only records where 'first_name' is 'DAN'. The first three rows are highlighted in blue, and the fourth row is highlighted in light blue. The first row has 'actor_id' 18, 'first_name' 'DAN', 'last_name' 'TORN', and 'last_update' '2006-02-15 04:34:33'. The second row has 'actor_id' 56, 'first_name' 'DAN', 'last_name' 'HARRIS', and 'last_update' '2006-02-15 04:34:33'. The third row has 'actor_id' 116, 'first_name' 'DAN', 'last_name' 'STREEP', and 'last_update' '2006-02-15 04:34:33'. The fourth row has 'actor_id' NULL, 'first_name' NULL, 'last_name' NULL, and 'last_update' NULL.






	actor_id	first_name	last_name	last_update
▶	18	DAN	TORN	2006-02-15 04:34:33
	56	DAN	HARRIS	2006-02-15 04:34:33
	116	DAN	STREEP	2006-02-15 04:34:33
*	NULL	NULL	NULL	NULL

CONSULTA CON CLÁUSULA CONDICIONAL WHERE

La cláusula WHERE se utiliza para seleccionar aquellos registros que cumplen una o más condiciones






```
select * from city where country_id = 44;
```

```
1 • select * from city where country_id = 44;
```

Result Grid				
Filter Rows: <input type="text"/>				
Edit:   				
Export/Import:  				
	city_id	city	country_id	last_update
▶	8	Adoni	44	2006-02-15 04:45:25
	9	Ahmadnagar	44	2006-02-15 04:45:25
	18	Allappuzha (Alleppey)	44	2006-02-15 04:45:25
	22	Ambattur	44	2006-02-15 04:45:25
	24	Amroha	44	2006-02-15 04:45:25
	51	Balurghat	44	2006-02-15 04:45:25
	71	Berhampore (Baharampur)	44	2006-02-15 04:45:25
	73	Bhavnagar	44	2006-02-15 04:45:25
	74	Bhilwara	44	2006-02-15 04:45:25

```
select * from customer where store_id=1;
```



```
1 • select * from customer where store_id=1;
```

Result Grid							
Filter Rows: <input type="text"/>							
Edit:   							
Export/Import:   Wrap Cell Co							
	customer_id	store_id	first_name	last_name	email	address_id	ac
▶	1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1
	2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1
	3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1
	5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1
	7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1
	10	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacustomer.org	14	1
	12	1	NANCY	THOMAS	NANCY.THOMAS@sakilacustomer.org	16	1
	15	1	HELEN	HARRIS	HELEN.HARRIS@sakilacustomer.org	19	1
	17	1	DONNA	THOMPSON	DONNA.THOMPSON@sakilacustomer.org	21	1





CONSULTA CON CLAUSULA WHERE

La cláusula WHERE se utiliza para seleccionar aquellos registros que cumplen una o más condiciones

```
1 • select * from inventory;
```

Result Grid				
Filter Rows: <input type="text"/>				
Edit:  				
	inventory_id	film_id	store_id	last_update
▶	1	1	1	2006-02-15 05:09:17
	2	1	1	2006-02-15 05:09:17
	3	1	1	2006-02-15 05:09:17
	4	1	1	2006-02-15 05:09:17
	5	1	2	2006-02-15 05:09:17
	6	1	2	2006-02-15 05:09:17
	7	1	2	2006-02-15 05:09:17
	8	1	2	2006-02-15 05:09:17
	9	2	2	2006-02-15 05:09:17
	10	2	2	2006-02-15 05:09:17

```
1 • select * from inventory where film_id >= 50;
```

Result Grid				
Filter Rows: <input type="text"/>				
Edit:    Export/Import: 				
	inventory_id	film_id	store_id	last_update
▶	222	50	1	2006-02-15 05:09:17
	223	50	1	2006-02-15 05:09:17
	224	50	1	2006-02-15 05:09:17
	225	50	2	2006-02-15 05:09:17
	226	50	2	2006-02-15 05:09:17
	227	51	1	2006-02-15 05:09:17
	228	51	1	2006-02-15 05:09:17
	229	51	2	2006-02-15 05:09:17
	230	51	2	2006-02-15 05:09:17
	231	51	2	2006-02-15 05:09:17

CONSULTA CON CLAUSULA WHERE

La cláusula WHERE se utiliza para seleccionar aquellos registros que cumplen una o más condiciones

[illegible][illegible]

CONSULTA CON CLÁUSULA CONDICIONAL WHERE

La cláusula WHERE se utiliza para seleccionar aquellos registros que cumplen una o más condiciones

```
1 • select * from language
```



Result Grid



Filter Rows:

Edit:



	language_id	name	last_update
▶	1	English	2006-02-15 05:02:19
	2	Italian	2006-02-15 05:02:19
	3	Japanese	2006-02-15 05:02:19
	4	Mandarin	2006-02-15 05:02:19
	5	French	2006-02-15 05:02:19
	6	German	2006-02-15 05:02:19
*	NULL	NULL	NULL

```
1 • select * from language where name <> "French"
```



Result Grid



Filter Rows:

Edit:



Export








	language_id	name	last_update
▶	1	English	2006-02-15 05:02:19
	2	Italian	2006-02-15 05:02:19
	3	Japanese	2006-02-15 05:02:19
	4	Mandarin	2006-02-15 05:02:19
	6	German	2006-02-15 05:02:19
*	NULL	NULL	NULL

CONSULTA CON WHERE CON OPERADORES AND, OR, NOT

El operador **AND** muestra un registro si todas las condiciones separadas por AND son VERDADERAS.

```
1 SELECT * FROM country where country= "Argentina" and country_id=6;
```

<

Result Grid   Filter Rows: Edit:    Export/Import:   W

	country_id	country	last_update
▶	6	Argentina	2006-02-15 04:44:00
*	NULL	NULL	NULL

CONSULTA CON WHERE CON OPERADORES AND, OR, NOT

CON EL OPERADOR OR EL CASO ES DIFERENTE El operador OR muestra un registro si alguna de las condiciones separadas por OR es VERDADERA

```
1 SELECT * FROM country where country= "Argentina" or country_id=4;
```

Result Grid Filter Rows: Edit: Export/Import: Wrap

	country_id	country	last_update
▶	4	Angola	2006-02-15 04:44:00
	6	Argentina	2006-02-15 04:44:00
*	NULL	NULL	NULL

```
1 SELECT * FROM country where country= "Argentina" or country="Angola";
```






Result Grid Filter Rows: Edit: Export/Import: Wrap Cel

	country_id	country	last_update
▶	4	Angola	2006-02-15 04:44:00
	6	Argentina	2006-02-15 04:44:00
*	NULL	NULL	NULL

CONSULTA CON WHERE CON OPERADORES AND, OR, NOT

El operador NOT muestra un registro si la condición (es) NO VERDADERA.

```
1  SELECT * FROM category where not name="Action"
```

Result Grid |   Filter Rows: | Edit:    | Ex

	category_id	name	last_update
▶	2	Animation	2006-02-15 04:46:27
	3	Children	2006-02-15 04:46:27
	4	Classics	2006-02-15 04:46:27
	5	Comedy	2006-02-15 04:46:27
	6	Documentary	2006-02-15 04:46:27
	7	Drama	2006-02-15 04:46:27
	8	Family	2006-02-15 04:46:27
	9	Foreign	2006-02-15 04:46:27
	10	Games	2006-02-15 04:46:27
	11	Horror	2006-02-15 04:46:27

CONSULTA CON WHERE CON OPERADORES AND, OR, NOT

```
1 • select * from rental where not staff_id = 1
2   and customer_id > 250
3   and inventory_id < 100 order by customer_id asc;
```

Result Grid |   Filter Rows: | Edit:    | Export/Import:   | Wrap

	rental_id	rental_date	inventory_id	customer_id	return_date	staff_id	last_up
▶	7168	2005-07-27 07:51:11	6	252	2005-08-01 04:08:11	2	2006-02
	5072	2005-07-09 05:01:58	94	254	2005-07-18 08:17:58	2	2006-02
	11928	2005-08-17 16:28:24	79	261	2005-08-23 17:50:24	2	2006-02
	2562	2005-06-19 03:15:05	51	265	2005-06-21 08:26:05	2	2006-02
	2699	2005-06-19 13:29:28	72	267	2005-06-24 11:15:28	2	2006-02
	8931	2005-07-30 02:30:07	16	268	2005-08-02 08:24:07	2	2006-02
	10310	2005-08-01 04:24:47	9	271	2005-08-04 05:36:47	2	2006-02
	4937	2005-07-08 22:29:59	33	273	2005-07-15 21:51:59	2	2006-02
	14483	2005-08-21 13:43:59	37	275	2005-08-28 16:38:59	2	2006-02
	13037	2005-08-19 08:53:57	65	275	2005-08-28 08:56:57	2	2006-02

CONSULTA CON WHERE CON OPERADORES: IN

El operador IN (en) le permite especificar varios valores en una cláusula WHERE.

El operador IN es una abreviatura de múltiples condiciones

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

```
1 • select * from customer where first_name = "MARY" and first_name = "PATRICIA";
```

Below the query, there is a "Result Grid" section. The header row contains the following columns: customer_id, store_id, first_name, last_name, email, address_id, active, create_date, last_update. The first row of data shows all NULL values.

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

```
1 • select * from customer where first_name = "MARY" OR first_name = "PATRICIA";
```

Below the query, there is a "Result Grid" section. The header row contains the following columns: customer_id, store_id, first_name, last_name, email, address_id, active. The first two rows of data are highlighted in blue:

customer_id	store_id	first_name	last_name	email	address_id	active
1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1
*	NULL	NULL	NULL	NULL	NULL	NULL

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

```
1 • select * from customer where first_name in ('MARY', 'PATRICIA');
```

Below the query, there is a "Result Grid" section. The header row contains the following columns: customer_id, store_id, first_name, last_name, email, address_id, active. The first two rows of data are highlighted in blue:

customer_id	store_id	first_name	last_name	email	address_id	active
1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1
*	NULL	NULL	NULL	NULL	NULL	NULL

CONSULTA CON WHERE CON OPERADORES: IN






```
1 • select * from film where special_features in ('Trailers', 'Deleted Scenes') and rating in ('G', 'PG13')
```

base_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost	rating	special_features	last_update
2006	1	NULL	6	2.99	130	22.99	G	Deleted Scenes	2006-02-15 05:03:42
2006	1	NULL	3	2.99	74	15.99	G	Trailers	2006-02-15 05:03:42
2006	1	NULL	5	0.99	99	10.99	G	Trailers	2006-02-15 05:03:42
2006	1	NULL	7	2.99	148	21.99	G	Trailers	2006-02-15 05:03:42
2006	1	NULL	4	0.99	63	29.99	G	Deleted Scenes	2006-02-15 05:03:42
2006	1	NULL	4	2.99	61	14.99	G	Trailers	2006-02-15 05:03:42
2006	1	NULL	3	4.99	179	16.99	G	Trailers	2006-02-15 05:03:42
		NULL							

CONSULTA CON WHERE CON OPERADORES: IN COMBINADO CON NOT.

El operador NOT muestra un registro si la condición (es) NO es VERDADERA.

```
1 • select * from category where name not in ('Action', 'Animation', 'Children');
```

Result Grid			
Filter Rows: <input type="text"/>			
Edit:   			
Export/Import:  			
Wrap Cell Content			
	category_id	name	last_update
▶	4	Classics	2006-02-15 04:46:27
	5	Comedy	2006-02-15 04:46:27
	6	Documentary	2006-02-15 04:46:27
	7	Drama	2006-02-15 04:46:27
	8	Family	2006-02-15 04:46:27
	9	Foreign	2006-02-15 04:46:27
	10	Games	2006-02-15 04:46:27
	11	Horror	2006-02-15 04:46:27
	12	Music	2006-02-15 04:46:27
	13	New	2006-02-15 04:46:27

PRÁCTICA N° 1

- ## 1. CONSULTAR TODAS LAS PELÍCULAS
- ## 2. CONSULTAR LOS TÍTULOS DE TODAS LAS PELÍCULAS (FILM)
- ## 3. CONSULTAR NOMBRE Y APELLIDO DE LA TABLA ACTOR

PRÁCTICA N° 2

- ## 1. CONSULTA STORE_ID, FIRST_NAME Y LAST_NAME DE LA TABLA CUSTOMER DE LA BASE DE DATOS SAKILA.
- ## 2. CAMBIA EL NOMBRE DE LAS COLUMNAS STORE_ID, FIRST_NAME Y LAST_NAME A TIENDA, NOMBRE Y APELLIDO RESPECTIVAMENTE.
- ## 3. ORDENA DE MANERA DESCENDENTE LA COLUMNA APELLIDO TABLA CUSTOMER
- ## 4. CONSULTA LA TABLA PAYMENT DE LA BASE DE DATOS SAKILA.
- ## 5. ¿CUÁL ES LA CANTIDAD MÁS BAJA Y MÁS ALTA DE LA COLUMNA AMOUNT?

PRÁCTICA N° 3

1. Consulta description, release_year de la tabla film de la base de datos sakila y filtra la información dónde title sea IMPACT ALADDIN

2. Consulta la tabla **payment** de la base de datos Sakila muestra la información donde **amount** sea mayor a 0.99.

PRÁCTICA N° 4

1. Consulta la tabla payment de la base de datos sakila filtra la información donde customer_id sea igual a 36, y amount sea mayor a 0.99 y staff_id sea igual a 1.

2. Consulta la tabla rental de la base de datos sakila filtra la información donde staff_id no sea 1, y customer_id sea mayor a 250 y inventory_id sea menor de 100.

PRÁCTICA N° 5

1: Consulta la tabla film_text de la base de datos sakila, filtra la información dónde title sea ZORRO ARK, VIRGIN DAISY, UNITED PILOT

2.Consulta la tabla city de la base de datos sakila filtra la información donde city sea Chiayi, Dongying, Fukuyama y Kilis.

#3.Consulta la tabla city de la base de datos sakila
#filtra la información donde city no sea Chiayi, Dongying, Fukuyama y Kilis.

```
select city as ciudad from city where city not in ("Chiayi",  
"Dongying", "Fukuyama", "Kilis");
```

SOLUCION PRACTICA 5

Solución con not: `select title from film_text where title in ("ZORRO ARK", "VIRGIN DAISY", "UNITED PILOT");`

```
26 • select title from film_text where title in ("ZORRO ARK", "VIRGIN DAISY", "UNITED PILOT");
```

```
27 # 3 Consulta la tabla city de la base de datos sakila
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title
UNITED PILOT
VIRGIN DAISY
ZORRO ARK

Solución: `select city as ciudad from city where city in ("Chiayi", "Dongying", "Fukuyama", "Kilis");;`

```
29 • select city as ciudad from city where city in ("Chiayi", "Dongying", "Fukuyama", "Kilis");
```

```
30
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ciudad
Chiayi
Dongying
Fukuyama
Kilis

#3.Consulta la tabla city de la base de datos sakila
#filtra la información donde city no sea Chiayi,
Dongying, Fukuyama y Kilis.

select city as ciudad from city where city not in ("Chiayi", "Dongying", "Fukuyama", "Kilis");

CONSULTA CON BETWEEN(entre):

El operador BETWEEN es inclusivo: se incluyen los valores inicial y final.

```
1 • select * from RENTAL where (customer_id between 300 and 350)
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

	rental_id	rental_date	inventory_id	customer_id	return_date	staff_id	last_update
▶	457	2005-05-27 19:52:29	2368	300	2005-06-02 17:17:29	2	2006-02-15 21:30:00
	780	2005-05-29 14:18:32	1919	300	2005-06-06 20:14:32	1	2006-02-15 21:30:00
	1111	2005-05-31 15:24:19	1137	300	2005-06-08 21:18:19	1	2006-02-15 21:30:00
	1381	2005-06-15 15:17:21	3262	300	2005-06-20 17:07:21	2	2006-02-15 21:30:00
	3177	2005-06-20 22:32:44	2546	300	2005-06-22 23:01:44	1	2006-02-15 21:30:00
	3775	2005-07-06 13:27:33	3696	300	2005-07-09 10:27:33	1	2006-02-15 21:30:00
	4030	2005-07-07 02:25:42	1734	300	2005-07-08 22:53:42	2	2006-02-15 21:30:00
	5562	2005-07-10 03:17:42	3604	300	2005-07-12 03:26:42	1	2006-02-15 21:30:00
	5705	2005-07-10 10:09:17	4030	300	2005-07-19 07:24:17	2	2006-02-15 21:30:00
	6111	2005-07-11 07:26:57	3504	300	2005-07-13 10:43:57	2	2006-02-15 21:30:00

RENTAL 61 x

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

```
1 • select * from RENTAL where (customer_id between 300 and 350) and staff_id = 1
```

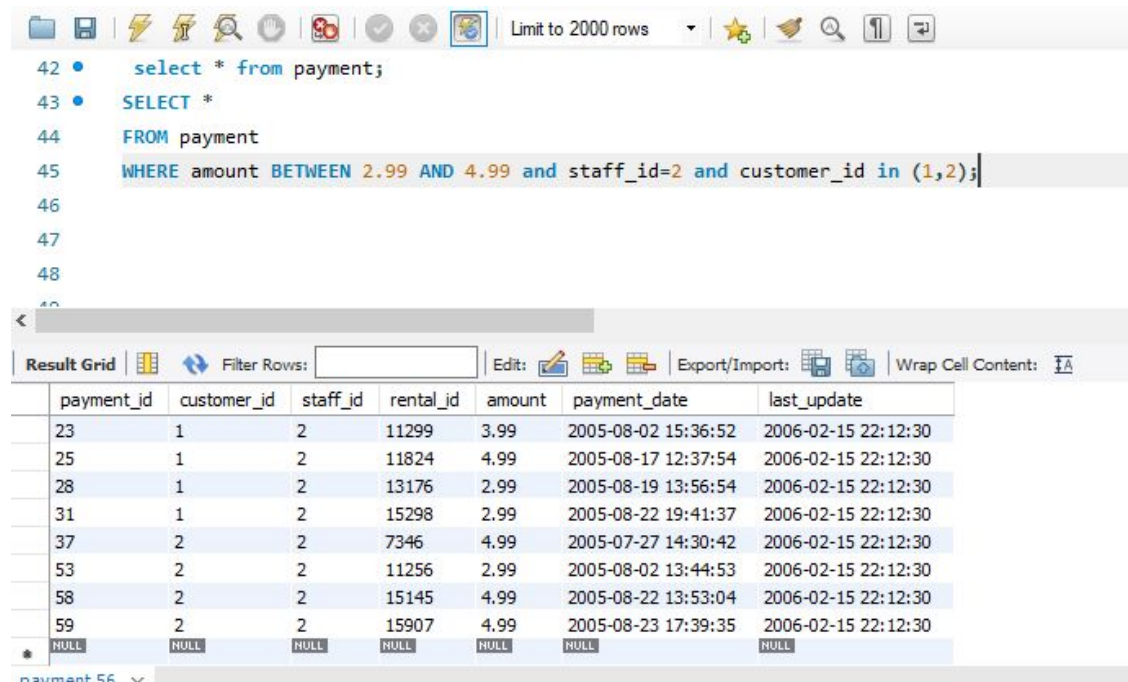
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

	rental_id	rental_date	inventory_id	customer_id	return_date	staff_id	last_update
▶	780	2005-05-29 14:18:32	1919	300	2005-06-06 20:14:32	1	2006-02-15 21:30:00
	1111	2005-05-31 15:24:19	1137	300	2005-06-08 21:18:19	1	2006-02-15 21:30:00
	3177	2005-06-20 22:32:44	2546	300	2005-06-22 23:01:44	1	2006-02-15 21:30:00
	3775	2005-07-06 13:27:33	3696	300	2005-07-09 10:27:33	1	2006-02-15 21:30:00
	5562	2005-07-10 03:17:42	3604	300	2005-07-12 03:26:42	1	2006-02-15 21:30:00
	6822	2005-07-12 18:23:39	1592	300	2005-07-19 21:06:39	1	2006-02-15 21:30:00
	8117	2005-07-28 19:20:16	1502	300	2005-08-05 23:55:16	1	2006-02-15 21:30:00
	8210	2005-07-28 23:31:05	3424	300	2005-08-06 17:36:05	1	2006-02-15 21:30:00
	9078	2005-07-30 08:01:00	955	300	2005-07-31 10:39:00	1	2006-02-15 21:30:00
	9127	2005-07-30 09:46:36	4114	300	2005-07-31 07:43:36	1	2006-02-15 21:30:00

RENTAL 65 x

PRÁCTICA N° 6

1. CONSULTA LA TABLA PAYMENT DE LA BASE DE DATOS SAKILA FILTRA LA INFORMACIÓN DONDE AMOUNT ESTÉ ENTRE 2.99 Y 4.99, STAFF_ID SEA IGUAL A 2 Y CUSTOMER_ID SEA 1 Y 2



The screenshot shows a database query editor with a toolbar at the top. The SQL query is as follows:

```
42 • select * from payment;
43 • SELECT *
44 FROM payment
45 WHERE amount BETWEEN 2.99 AND 4.99 and staff_id=2 and customer_id in (1,2);
46
47
48
49
```

Below the query editor, the results are displayed in a table. The table has 8 columns: payment_id, customer_id, staff_id, rental_id, amount, payment_date, and last_update. The results show 10 rows of data, including a row with NULL values.

payment_id	customer_id	staff_id	rental_id	amount	payment_date	last_update
23	1	2	11299	3.99	2005-08-02 15:36:52	2006-02-15 22:12:30
25	1	2	11824	4.99	2005-08-17 12:37:54	2006-02-15 22:12:30
28	1	2	13176	2.99	2005-08-19 13:56:54	2006-02-15 22:12:30
31	1	2	15298	2.99	2005-08-22 19:41:37	2006-02-15 22:12:30
37	2	2	7346	4.99	2005-07-27 14:30:42	2006-02-15 22:12:30
53	2	2	11256	2.99	2005-08-02 13:44:53	2006-02-15 22:12:30
58	2	2	15145	4.99	2005-08-22 13:53:04	2006-02-15 22:12:30
59	2	2	15907	4.99	2005-08-23 17:39:35	2006-02-15 22:12:30
NULL	NULL	NULL	NULL	NULL	NULL	NULL

At the bottom left, there is a status bar showing "payment 56" and a dropdown arrow.

CONSULTAS CON BETWEEN

VAMOS A COMBINAR BETWEEN Y OPERADORES AND, OR

PRÁCTICA N° 7

BUSCAR PELÍCULAS QUE TENGAN UNA DURACIÓN ENTRE (BETWEEN) 100 Y (AND) 120 MINUTOS O (OR) ENTRE (BETWEEN) 50 Y (AND) 70 MINUTOS TABLA FILM

SOLUCIÓN :

```
1 SELECT * FROM sakila.film
2 where length between 100 and 120 or length between 50 and 70
3
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: ☐

	release_year	language_id	original_lang	rental_dur	rental_rate	length	replacement_cost	rating	special_features
anorama of a Crocodile And a ...	2006	1	NULL	4	4.99	60	20.99	R	Trailers,Commentarie
haracter Study of a Secret Ag...	2006	1	NULL	4	2.99	119	17.99	G	Commentaries
of a Mad Cow And a Pioneer ...	2006	1	NULL	4	2.99	61	14.99	G	Trailers
of a Feminist And a Feminist ...	2006	1	NULL	4	2.99	63	13.99	G	Behind the Scenes
n of a Feminist And a Databas...	2006	1	NULL	6	0.99	67	19.99	PG-13	Trailers,Commentarie
of a Boat And a Man who mu...	2006	1	NULL	4	0.99	53	25.99	NC-17	Commentaries,Delete
pistle of a Woman And a Mad...	2006	1	NULL	3	0.99	52	17.99	NC-17	Commentaries,Delete
of a Frisbee And a Lumberjack...	2006	1	NULL	4	4.99	120	22.99	NC-17	Trailers,Commentarie
ter Study of a Woman And a	2006	1	NULL	3	0.99	61	26.99	NC-17	Trailers,Commentarie

film 2 x Apply Revert

CONSULTA CON LIKE

El operador LIKE se usa en una cláusula WHERE para buscar un patrón específico en una columna.

- %: El signo de porcentaje representa cero, uno o varios caracteres
- _ - El guión bajo representa un solo carácter

```
SELECT column1, column2, ...  
FROM table_name  
WHERE columnN LIKE pattern;
```

1 • `select * from actor where first_name like 'A%'`

Result Grid | Filter Rows: | Edit: | Export/Im

	actor_id	first_name	last_name	last_update
▶	29	ALEC	WAYNE	2006-02-15 04:34:33
	34	AUDREY	OLIVIER	2006-02-15 04:34:33
	49	ANNE	CRONYN	2006-02-15 04:34:33
	65	ANGELA	HUDSON	2006-02-15 04:34:33
	71	ADAM	GRANT	2006-02-15 04:34:33
	76	ANGELINA	ASTAIRE	2006-02-15 04:34:33
	125	ALBERT	NOLTE	2006-02-15 04:34:33
	132	ADAM	HOPPER	2006-02-15 04:34:33

1 • `select * from actor where first_name like 'A%' and last_name like 'C%'`

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co








	actor_id	first_name	last_name	last_update
▶	49	ANNE	CRONYN	2006-02-15 04:34:33
*	NULL	NULL	NULL	NULL

CONSULTA CON LIKE

El operador LIKE se usa en una cláusula WHERE para buscar un patrón específico en una columna.

- %: El signo de porcentaje representa cero, uno o varios caracteres
- _ El guión bajo representa un solo carácter

```
1 • select * from actor where first_name like '%A' and last_name like '%E'
```

Result Grid   Filter Rows: Edit:    Export/Import:   Wrap Cell Center

	actor_id	first_name	last_name	last_update
▶	47	JULIA	BARRYMORE	2006-02-15 04:34:33
	53	MENA	TEMPLE	2006-02-15 04:34:33
	76	ANGELINA	ASTAIRE	2006-02-15 04:34:33
	122	SALMA	NOLTE	2006-02-15 04:34:33
	178	LISA	MONROE	2006-02-15 04:34:33
	200	THORA	TEMPLE	2006-02-15 04:34:33
*	NULL	NULL	NULL	NULL

```
1 • select * from actor where first_name like '%AL'
```



Result Grid   Filter Rows: Edit:    Export/Imp

	actor_id	first_name	last_name	last_update
▶	37	VAL	BOLGER	2006-02-15 04:34:33
	165	AL	GARLAND	2006-02-15 04:34:33
*	NULL	NULL	NULL	NULL

CONSULTA CON LIKE

```
1 • select * from actor where first_name like 'A%E'
```

AQUÍ LE PEDIMOS QUE BUSQUE EN actor un nombre que comience por A y termine en E

<				
Result Grid				
Filter Rows: <input type="text"/>				
Edit: 				
Export/Import: 				
	actor_id	first_name	last_name	last_update
▶	49	ANNE	CRONYN	2006-02-15 04:34:33
✱	NULL	NULL	NULL	NULL

CONSULTA CON LIKE

PRÁCTICA N° 8

```
## 1. CONSULTA LA TABLA FILM DE LA BASE DE DATOS SAKILA FILTRA LA
INFORMACIÓN DONDE RELEASE_YEAR SEA IGUAL A 2006 Y TITLE EMPIECE CON ALI
## 2. PELÍCULAS QUE CUESTEN 0.99, 2.99 Y TENGAN UN RATING 'G' O 'R'
Y QUE HABLEN DE COCODRILOS (COCODRILE)
## 3. DIRECCIONES DE ONTARIO O DE PUNJAB O QUE SU CÓDIGO POSTAL
ACABE EN 5 O QUE SU TELÉFONO ACABE EN 5
##
```

CONSULTA CON LIKE

SOLUCIÓN PRÁCTICA N° 7

```
2. SELECT * FROM sakila.film where description like  
'%crocodile%' and rental_rate in (0.99,2.99) and rating in  
('R','G')
```

```
3. SELECT * FROM sakila.address  
where district='ontario' or district='punjab'  
or postal_code like '%5' or phone like '%5'
```

CONSULTA CON LIKE

SOLUCIÓN PRÁCTICA 8. ##1

```
1 • select * from sakila.film where release_year = 2006 and title like 'ALI%'
```

<						
Result Grid						
Filter Rows: <input type="text"/>						
Edit:						
Export/Import:						
Wrap Cell Content						
	film_id	title	description	release_year	language_id	original
▶	13	ALI FOREVER	A Action-Packed Drama of a Dentist And a Croc...	2006	1	NULL
	14	ALICE FANTASIA	A Emotional Drama of a A Shark And a Databas...	2006	1	NULL
	15	ALIEN CENTER	A Brilliant Drama of a Cat And a Mad Scientist w...	2006	1	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

Consultas con agrupamiento

La declaración **GROUP BY** agrupa las filas que tienen los mismos valores en filas de resumen, cómo "encontrar el número de clientes en cada país".

La instrucción **GROUP BY** se usa a menudo con funciones agregadas (**COUNT**, **MAX**, **MIN**, **SUM**, **AVG**) para agrupar el conjunto de resultados por una o más columnas.

FUNCIONES

La función COUNT() devuelve el número de filas que coinciden con un criterio específico.

SE UTILIZA CON SELECT Y WHERE

Consulta los apellidos de los actores, así como cuántos actores tienen ese apellido.

```
SELECT last_name, COUNT(*) as 'N° DE  
APELLIDOS '  
FROM actor  
GROUP BY last_name;
```

FUNCIONES

La función COUNT() devuelve el número de filas que coinciden con un criterio específico.

Consulta los apellidos de la tabla actores , así como cuántos actores tienen ese apellido.

use sakila;

```
SELECT last_name as 'Apellidos', COUNT(*) as 'Nº APELLIDOS'  
FROM actor  
GROUP BY last_name;
```

```
SELECT last_name as 'Apellidos',  
COUNT(*) as 'Nº APELLIDOS'  
FROM actor WHERE last_name  
="NEESON"  
GROUP BY last_name;
```

FUNCIONES

HAVING

Especifica una condición de búsqueda para un grupo o agregado. **HAVING** solo se puede utilizar con la instrucción **SELECT**. Normalmente, **HAVING** se usa con una cláusula **GROUP BY**. Cuando no se usa **GROUP BY**,

La cláusula **WHERE** se aplica primero a las filas individuales de las tablas u objetos con valores de tabla del panel Diagrama. Solo se agrupan las filas que cumplen las condiciones de la cláusula **WHERE**.

La cláusula **HAVING** se aplica a continuación a las filas del conjunto de resultados. Solo aparecen en el resultado de la consulta los grupos que cumplen las condiciones **HAVING**. Solo se puede aplicar una cláusula **HAVING** a las columnas que también aparecen en la cláusula **GROUP BY** o en una función de agregado.

FUNCIONES

Consulta en apellidos de la tabla actores la cantidad de actores que tienen el mismo apellido, mas de 3 veces o igual a 3

```
SELECT last_name, COUNT(*) as 'Last  
Name Count'  
FROM actor  
GROUP BY last_name  
HAVING COUNT(*) >=3;
```

CONSULTAS A LA BASE DE DATOS INSTITUTO

PRACTICA CON INSERT.

FORMA 1

```
INSERT INTO alumno (id,nombre ,apellido1 ,apellido2, fecha_nacimiento, es_repetidor, teléfono)  
VALUES (1, "Maria", "Sánchez","Pérez","1990/12/01", "No", "600335878" );  
select * from alumno;
```

#FORMA 2

```
INSERT INTO alumno VALUES (2, "Juan", "Sáez", "Vega", "1998-04-02", "No", "615668576");  
INSERT INTO alumno VALUES (3, "Tomás", "Ramírez", "Gea", "1988-01-03", "No", null);  
INSERT INTO alumno VALUES (4, "Lucía", "Sánchez", "Ortega", "1993-06-13", "Si", null);  
INSERT INTO alumno VALUES (5, "Francisco", "Martínez", "López", "1995-11-24", "No", 698597452);  
INSERT INTO alumno VALUES (6, "Irene", "Gutierrez", "Sánchez", "1991-03-28", "Si", null);  
INSERT INTO alumno VALUES (7, "Cristina", "Fernandez", "Ramirez", "1996-09-17", "No", "628962312");  
INSERT INTO alumno VALUES (8, "Antonio", "Carretero", "Ortega", "1994-05-20", "Si", 612858585);  
INSERT INTO alumno VALUES (9, "Manuel", "Domínguez", "Hernández", "1999-07-08", "No", null);  
INSERT INTO alumno VALUES (10, "Daniel ", "Moreno", "Ruiz", "1998-02-03", "No", null);
```

#REALIZAR LAS CONSULTAS

CONSULTAR TODO EL CONTENIDO DE LA BASE DE DATOS

```
SELECT * FROM ALUMNO ;
```

```
SELECT *
```

```
FROM ALUMNO ;
```

CONSULTAR EL NOMBRE DE TODOS LOS ALUMNOS

```
SELECT nombre
```

```
from Alumno;
```

#CONSULTAR EL NOMBRE Y EL PRIMER APELLIDO

```
SELECT NOMBRE, APELLIDO1 FROM ALUMNO;
```

#CONSULTAR EL NOMBRE Y EL SEGUNDO APELLIDO APELLIDO

```
SELECT NOMBRE, APELLIDO2 FROM ALUMNO;
```

```
SELECT NOMBRE, APELLIDO1, FECHA_NACIMIENTO FROM ALUMNO WHERE NOMBRE ="Lucía";
```

#CONSULTA CON LA FUNCIÓN CONCAT

```
SELECT * FROM ALUMNO;
```

```
SELECT CONCAT(nombre, apellido1,apellido2) AS NOMBRE_Y_APELLIDOS FROM ALUMNO;
```

#CONSULTA CON LA FUNCIÓN CONCAT AÑADIENDO SEPARACIÓN DE LOS REGISTROS UTILIZANDO CONCAT_WS

```
SELECT CONCAT_WS(' ',nombre, apellido1,apellido2) AS NOMBRE_Y_APELLIDOS FROM ALUMNO;
```

#CONSULTA EN UNA SOLA COLUMNA EL NOMBRE Y APELLIDOS DE LOS ALUMNO ADEMÁS ORDENA ALFABÉTICAMENTE

#DE FORMA ASCENDENTE POR EL NOMBRE

```
SELECT CONCAT_WS(' ',nombre, apellido1,apellido2) AS NOMBRE_Y_APELLIDOS FROM ALUMNO ORDER BY NOMBRE ASC;
```

#CONSULTAR EL NOMBRE Y APELLIDOS DE LOS ALUMNOS Y ORDENAR POR APELLIDO 1 Y APELLIDO 2

```
SELECT NOMBRE, APELLIDO1, APELLIDO2 FROM ALUMNO ORDER BY APELLIDO1, APELLIDO2;
```

#INSERTAR UN REGISTRO CON DONDE REPETIDOR SEA QUIZÁS

```
INSERT INTO alumno
```

```
VALUES (11, "Pedro ", "Martinez", "Rus", "1998-02-03", "Si", null);
```

```
SELECT * FROM ALUMNO;
```

#CONSULTAR LOS ALUMNOS REPETIDORES

```
SELECT NOMBRE, APELLIDO1, APELLIDO2, es_repetidor as Repite  
FROM ALUMNO WHERE es_repetidor="Si" order by nombre;
```

#CONSULTAR LOS ALUMNOS CON EL APELLIDO1 SANCHEZ

```
SELECT * FROM ALUMNO WHERE APELLIDO1="SANCHEZ" ;
```

#CONSULTAR LOS ALUMNOS QUE SE APELLIDAN SÁNCHEZ

```
SELECT * FROM ALUMNO WHERE APELLIDO1="SANCHEZ" OR APELLIDO2="SANCHEZ";
```

#CONSULTA EL NOMBRE Y EL APELLIDO DEL ALUMNO/A CON LA ID 6

```
SELECT NOMBRE, APELLIDO1, APELLIDO2 FROM ALUMNO WHERE ID=6;
```

#CONSULTAR LOS ALUMNOS/AS QUE HAN NACIDO EN 1994-05-20

```
SELECT * FROM ALUMNO WHERE fecha_nacimiento ="1994-05-20";
```

#CONSULTAR EL ALUMNADO NACIDO EN 1998

```
SELECT * FROM ALUMNO WHERE fecha_nacimiento between "1998/01/01" AND "1998/12/31" ORDER BY  
fecha_nacimiento ASC;
```

BUSCAR LOS ALUMNOS/AS QUE TIENEN DE PRIMER APELLIDO SAEZ, MORENO, FERNANDEZ

SELECT * FROM ALUMNO WHERE APELLIDO1 IN("SAEZ", "MORENO", "FERNANDEZ") ORDER BY APELLIDO1;

SELECT * FROM ALUMNO WHERE APELLIDO1="SAEZ" OR APELLIDO1="MORENO" OR
APELLIDO1="FERNÁNDEZ" ORDER BY APELLIDO1;

#BUSCAR LOS ALUMNO DONDE SU NOMBRE COMIENCEN POR D

SELECT * FROM ALUMNO WHERE NOMBRE LIKE "D%";

#CONSULTAR TODOS LOS ALUMNOS QUE EL PRIMER APELLIDO FINALICE EN Z

SELECT * FROM ALUMNO WHERE APELLIDO1 LIKE "%Z";

#CONSULTAR LOS ALUMNOS CONTENGAN LA LETRA I NOMBRE

SELECT * FROM ALUMNO WHERE NOMBRE LIKE "%I%";

#CONSULTAR LOS ALUMNOS CUYO NOMBRE TENGA 4 CARACTERES

GUION BAJO 4 CARACTERES CUATRO GUIONES

SELECT * FROM ALUMNO;

SELECT * FROM ALUMNO WHERE NOMBRE LIKE "____";

#TAREA

#CONSULTAR LOS ALUMNOS DE LA TABLA ALUMNO MOSTRANDO EL NOMBRE Y LOS DOS APELLIDOS EN UNA COLUMNA (CONCAT_SW)

```
SELECT NOMBRE, CONCAT_WS(' ', apellido1,apellido2) AS APELLIDOS FROM ALUMNO;
```

CONSULTA LOS ALUMNOS QUE NO SON REPETIDORES QUE SE VEA SU NOMBRE APELLIDOS Y NO_REPITEN

```
SELECT nombre as Nombre, CONCAT_WS(' ', apellido1,apellido2)  
as Apellidos, es_repetidor as No_repiten  
FROM ALUMNO WHERE es_repetidor="No" order by nombre asc;
```

CONSULTAR EL NUMERO DE TELEFONO DE LOS ALUMNOS QUE NO TIENEN TELEFONO

```
SELECT NOMBRE, teléfono FROM ALUMNO;
```

#FORMA INCORRECTA

```
SELECT NOMBRE, teléfono FROM ALUMNO WHERE teléfono= NULL;
```

#FORMA CORRECTA

```
SELECT NOMBRE, teléfono FROM ALUMNO WHERE teléfono IS null ORDER BY NOMBRE;
```

#CONSULTAR ALUMNOS QUE EL NOMBRE COMIENCE POR M

```
SELECT * FROM ALUMNO WHERE NOMBRE="M%";
```

```
SELECT * FROM ALUMNO WHERE NOMBRE LIKE "M%";
```

```
SELECT * FROM ALUMNO;
```


CONSULTAS DE AGRUPAMIENTO

DADAS LAS SIGUIENTES PROPUESTAS OBSERVA COMO ES LA SINTAXIS DE LAS FUNCIONES QUE SE EXPONEN EN LA SIGUIENTE DIAPOSITIVA. REALIZA LA CONSULTA CON LA PETICIÓN QUE SE PIDE

CONSULTAS CON AGRUPAMIENTO

COUNT: devuelve el número total de filas seleccionadas por la consulta.

MIN: devuelve el valor mínimo del campo que especifiquemos.

MAX: devuelve el valor máximo del campo que especifiquemos.

SUM: suma los valores del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.

AVG: devuelve el valor promedio del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.

SAKILA

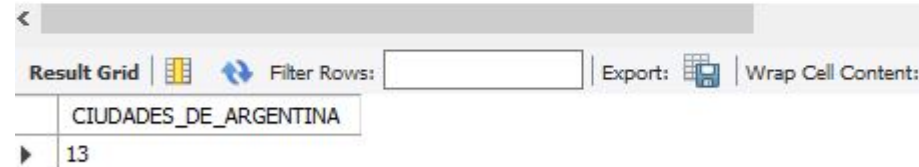
UTILIZANDO COUNT

```
SELECT COUNT(column_name)
FROM table name
WHERE condition;
```

¿Cuántas ciudades tiene el country 'ARGENTINA'?

```
SELECT * FROM CITY;
SELECT COUNT(*) AS CIUDADES_DE_ARGENTINA FROM CITY
WHERE COUNTRY_ID = 6;
```

```
1 • USE SAKILA;
2 • SELECT * FROM COUNTRY;
3 • SELECT * FROM CITY;
4 • SELECT COUNT(*) AS CIUDADES_DE_ARGENTINA FROM CITY
5   WHERE COUNTRY_ID = 6;
```



CIUDADES_DE_ARGENTINA
13

UTILIZANDO AVG

```
SELECT AVG(column_name)  
FROM table_name  
WHERE condition;
```

Calcular la media de duración de película(film)

```
SELECT AVG(LENGTH)AS  
PROMEDIO_DURACION  
FROM FILM;
```

SUM () Esta función nos permite sumar elementos de una columna

```
SELECT SUM(column_name)  
FROM table name  
WHERE condition;
```

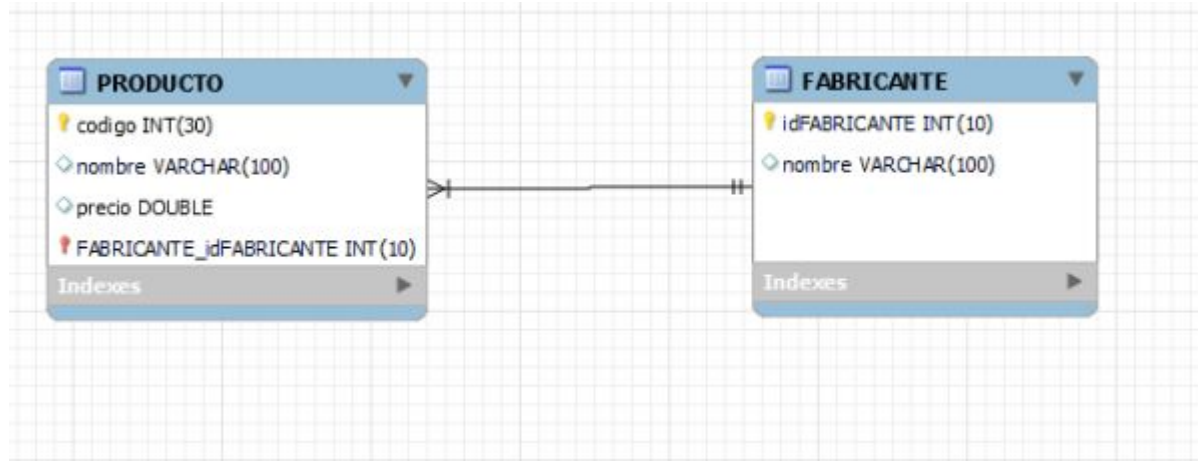
Suma el total de los importes del precio de las películas (rental_rate) de todas las películas.

```
SELECT * FROM FILM;  
SELECT SUM(RENTAL_RATE)  
SUMATORIA_TOTAL_PRECIO_D  
E_TOTAS_LAS_PELICULAS  
FROM FILM;
```

CONSULTAS
MULTITABLAS

BASE DE DATOS FERRETERIA

1: Dado el modelo de la base de datos FERRETERIA que consta de dos tablas con sus atributos :



INSERTA LOS DATOS A LAS TABLAS CORRESPONDIENTE:

```
INSERT TO fabricante VALUES (1, 'Asus');  
INSERT TO fabricante VALUES (2, 'Lenovo');  
INSERT TO fabricante VALUES (3, 'Hewlett-Packard');  
INSERT TO fabricante VALUES (4, 'Samsung');  
INSERT TO fabricante VALUES (5, 'Seagate');  
INSERT TO fabricante VALUES (6, 'Crucial');  
INSERT TO fabricante VALUES (7, 'Gigabyte');  
INSERT TO fabricante VALUES (8, 'Huawei');  
INSERT TO fabricante VALUES (9, 'Xiaomi');
```


INSERTA LOS DATOS A LAS TABLAS CORRESPONDIENTE:

```
INSERT TO producto VAL(1, 'Disco duro SATA3 1TB', 86.99, 5);
INSERT TO producto VAL(2, 'Memoria RAM DDR4 8GB', 120, 6);
INSERT TO producto VAL(3, 'Disco SSD 1 TB', 150.99, 4);
INSERT TO producto VAL(4, 'GeForce GTX 1050Ti', 185, 7);
INSERT TO producto VAL(5, 'GeForce GTX 1080 Xtreme', 755, 6);
INSERT TO producto VAL(6, 'Monitor 24 LED Full HD', 202, 1);
INSERT TO producto VAL(7, 'Monitor 27 LED Full HD', 245.99, 1);
INSERT TO producto VAL(8, 'Portátil Yoga 520', 559, 2);
INSERT TO producto VAL(9, 'Portátil Ideapd 320', 444, 2);
INSERT TO producto VAL(10, 'Impresora HP Deskjet 3720', 59.99, 3);
INSERT TO producto VAL(11, 'Impresora HP Laserjet Pro M26nw', 180, 3);
```

REALIZA LA ACTIVIDAD CONSULTAS BÁSICAS II
FERRETERIA

1. Visualiza una consulta que muestre el nombre del producto, precio y nombre de fabricante de todos los productos de la base de datos.
2. Visualiza una consulta con el nombre del producto, precio y nombre de fabricante de todos los productos de la base de datos. Ordena el resultado por el nombre del fabricante y en ascendente.
3. Visualiza una consulta con el código del producto, nombre del producto, código del fabricante y nombre del fabricante, de todos los productos de la base de datos.
4. Visualiza el nombre del producto, su precio y el nombre de su fabricante, del producto más barato.
5. Visualiza el nombre del producto, su precio y el nombre de su fabricante, del producto más caro.
6. Visualiza una consulta de todos los productos del fabricante Lenovo.

7. Visualiza una consulta de todos los productos del fabricante Crucial que tengan un precio mayor que 200€.
8. Visualiza una consulta con todos los productos de los fabricantes Asus, Hewlett-Packard y Seagate. Sin utilizar el operador IN.
9. Visualiza una consulta con todos los productos de los fabricantes Asus, Hewlett-Packard y Seagate. Utilizando el operador IN.
10. Visualiza una consulta con el nombre y el precio de todos los productos de los fabricantes y donde su nombre termine por la vocal e.
11. Visualiza una consulta con el nombre y el precio de todos los productos donde el nombre del fabricante contenga el carácter w en su nombre.
12. Visualiza una consulta con el nombre de producto, precio y nombre de fabricante, de todos los productos que tengan un precio mayor o igual a 180€. Ordene el resultado en primer lugar por el precio (en orden descendente) y en segundo lugar por el nombre (en orden ascendente)
13. Visualiza una consulta con el código y el nombre del fabricante, únicamente de aquellos fabricantes que tienen productos asociados en la base de datos.

CONSULTAS MULTITABLAS SIN JOIN

1. Visualiza una consulta que muestre el nombre del producto, nombre de fabricante de todos los productos de la base de datos.

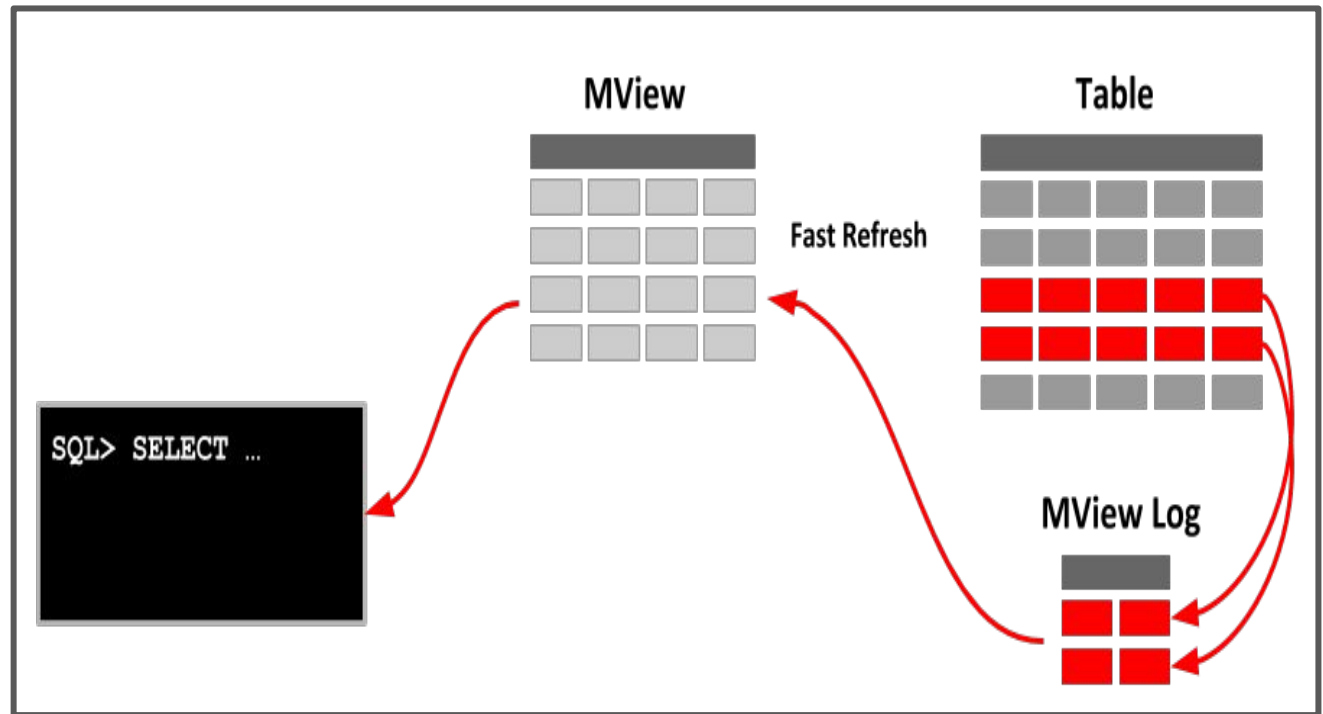
```
SELECT PRODUCTO.NOMBRE AS NOMBRE_PRODUCTO, PRODUCTO.PRECIO,  
FABRICANTE.NOMBRE AS NOMBRE_FABRICANTE FROM PRODUCTO, FABRICANTE  
WHERE FABRICANTE.CODIGO= PRODUCTO.FABRICANTE_CODIGO;
```

2. Visualiza una consulta con el nombre del producto, precio y nombre de fabricante de todos los productos de la base de datos. Ordena el resultado por el nombre del fabricante y en ascendente.
3. Visualiza una consulta con el código del producto, nombre del producto, código del fabricante y nombre del fabricante, de todos los productos de la base de datos.
4. Visualiza el nombre del producto, su precio y el nombre de su fabricante, del producto más barato.

Enlace : Presentación. CONSULTAS CON INNER JOIN, LEFT JOIN, RIGHT JOIN

https://docs.google.com/presentation/d/1RRyYDR2Esr7-Yq81amBDX-6uRnq4nfJB4MxMeB_wqRY/edit?usp=sharing

LAS VISTAS



SINTAXIS:

```
CREATE [OR REPLACE] VIEW nombre_vista [column_list]  
AS consulta_SELECT
```

- **OR REPLACE:** Reemplaza una vista existente en caso de coincidir en nombre.
- **nombre_vista:** Nombre de la vista a crear.
- **column_list:** Listado de columnas a crear.
- **consulta_SELECT:** Consulta SELECT que queremos realizar para obtener la información que contendrá la vista.

¿Qué son las vistas?

Una vista es un tipo de consulta (virtual) generada a partir de la ejecución de varias consultas sobre una o más tablas. Una vista posee la misma estructura de filas y columnas que cualquier otra tabla en MySQL, y se almacenan de la misma manera, como detalle a tener en cuenta no pueden existir dos con el mismo nombre.

Qué Ventajas tienen el uso de las vista

Privacidad de la información ya que podemos crear perfiles de usuarios con niveles de acceso permitiendo ocultar las tablas.

Optimizar la base de datos en cuanto a rendimiento, si tenemos un grupo de usuarios accediendo a la misma consulta sobre las mismas tablas pues lo más apropiado es tener una vista de dicha consulta.

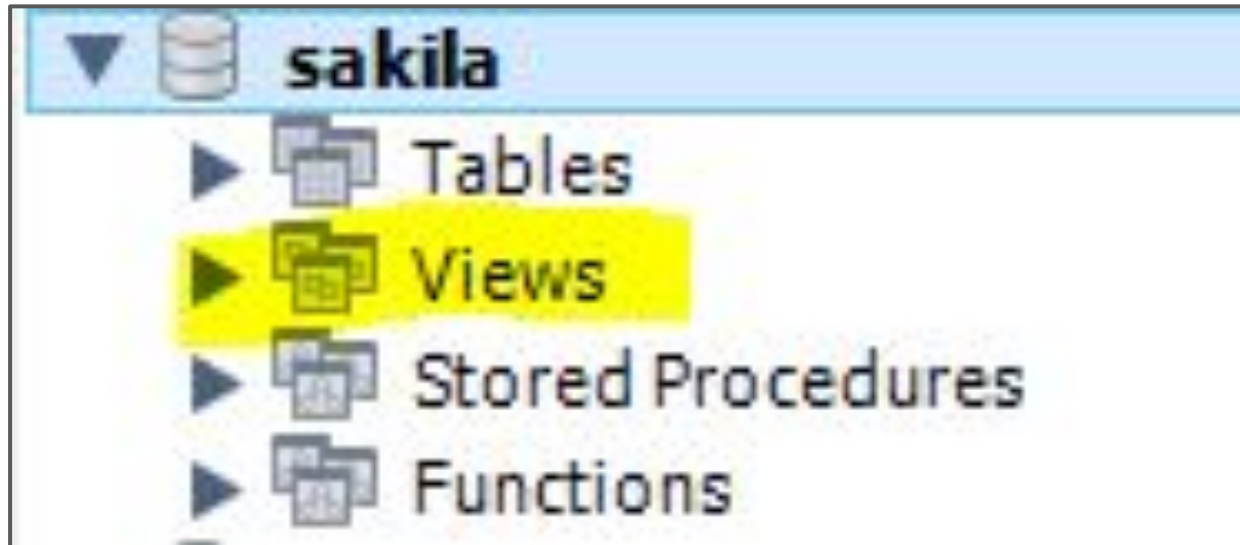
Cuanto más complejas sean las consultas que se deben ejecutar para obtener la vista, mayor será la ganancia de rendimiento.

También es una ventaja en términos de seguridad si no queremos que los usuarios puedan obtener datos de las tablas directamente, sino a través de la vista.

CREAR VISTAS

LAS VISTAS LAS PODEMOS CREAR DE DIFERENTES MANERAS TODAS MUY VÁLIDAS ESTAS PUEDEN SER SIMPLES O COMPLEJAS UNA VISTA ES UNA CONSULTA RECURRENTES.

EN WORKBENCH TENEMOS EN CADA BASE DE DATOS UN ÁREA PARA CREAR NUESTRAS VISTAS .



LAS VISTAS PUEDEN SER TODO LO COMPLEJO QUE SEA NECESARIO YA QUE LAS VISTAS SON CONSULTAS QUE SE REALIZAN DE FORMA HABITUAL Y ADEMÁS SE QUEDAN ALMACENADAS EN EL APARTADO View DE LA BASE DE DATOS. VAMOS A CREAR UNA VISTA.

SAKILA: Vamos a crear una vista que visualice los datos completos de mis empleados

```
CREATE VIEW DATOS_EMPLEADOS
```

```
#crear una vista con la siguiente consulta
```

```
#La vista se llamará datos_empleados
```

```
SELECT s.staff_id AS 'ID', CONCAT(s.first_name, s.last_name) AS Nombre,  
       a.address AS Direccion, a.postal_code AS Codigo_postal,  
       a.phone AS Tlf_contacto, sakila.city.city AS Ciudad,  
       sakila.country.country AS Pais,  
       s.store_id AS CodigoTienda  
FROM  
    (((sakila.staff s  
    JOIN sakila.address a ON ((s.address_id = a.address_id)))  
    JOIN sakila.city ON ((a.city_id = sakila.city.city_id)))  
    JOIN sakila.country ON ((sakila.city.country_id = sakila.country.country_id))));
```

OTRAS COSAS QUE PODEMOS REALIZAR SOBRE LAS VISTAS

#Consultar el listado de vistas disponibles

SHOW FULL TABLES

WHERE table_type = 'VIEW';

#RENOMBRAR UNA VISTA

#Renombrar una vista

RENAME TABLE `DATOS_EMPLEADOS` TO CONTACTO_EMPLEADOS;

#MODIFICAR UNA VISTA

ALTER VIEW nombre_vista [column_list]

AS consulta_SELECT

#MODIFICAR LA VISTA LISTAS

ALTER VIEW

LISTAS AS

SELECT first_name AS NOMBRE_ACTOR

FROM actor

ORDER BY first_name ASC;

ACTIVIDAD PRÁCTICA:

Consultar los nombres y direcciones de correo electrónico de todos los clientes canadienses

#CREAR UNA VISTA CON LA SIGUIENTE CONSULTA

```
select cust.first_name
       ,cust.last_name
       ,coalesce(cust.email, 'No Email Available') as 'Email'
from
customer cust
inner join
address as a
on cust.address_id = a.address_id
inner join
city
on a.city_id = city.city_id
inner join
country
on city.country_id = country.country_id
where
country.country = 'Canada';
```

UPDATE y ALTER

ACTUALIZAR CONTENIDO DE UNA TABLA

El comando **ALTER SQL** es una declaración **DDL** (lenguaje de definición de datos). ALTER se utiliza para actualizar la estructura de la tabla en la base de datos (como agregar, eliminar, modificar los atributos de las tablas en la base de datos).

Comando ACTUALIZAR:

El comando UPDATE SQL es una declaración **DML** (lenguaje de manipulación de datos). Se utiliza para manipular los datos de cualquier columna existente. Pero no se puede cambiar la definición de la tabla.

ALTER

- Cambiar el nombre a una tabla. ...
- Cambiar el propietario de una tabla o vista. ...
- Cambiar el nombre de una columna o campo
- Eliminar la restricción de una tabla.
- Modificar un campo con su tipo de dato VARCHAR. ...
- Modificar la codificación de compresión de una columna.
- Modificar el charset

SINTAXIS

UPDATE *table_name*
SET *column1 = value1, column2 = value2, ...*
WHERE *condition;*

UPDATE sakila.actor SET first_name='Maicol',last_name='Mora' WHERE actor_id=199;

MODIFICAR EL TIPO DE DATOS DE UN CAMPO

SINTAXIS

ALTER TABLE [tableName] MODIFY
[columnName] [newDataType];

SELECT * FROM ACTOR;
ALTER TABLE ACTOR MODIFY last_name VARCHAR(50);

```
#MODIFICAR EL TIPO DE DATOS DE UN CAMPO CON MODIFY, CHANGE  
SELECT * FROM ACTOR;  
ALTER TABLE ACTOR MODIFY last_name VARCHAR(50);  
ALTER TABLE actor CHANGE last_name last_name Varchar(52);
```

CAMBIAR UNA RESTRICCIÓN A UNA COLUMNA