



# **UNIVERSIDAD DE CARTAGENA**

## **INGENIERIA DE SOFTWARE**

**BASE DE DATOS I**

**EXAMEN FINAL**

**Nombre:**

**Kevin Antonio González**

**Código del estudiante:**

**7502420046**

**Docente:**

**JHON CARLOS ARRIETAS**

**NOVIEMBRE 2025**

**Aplicación asociada al proyecto:  
Sistema AppCine – CRUD con MySQL**

**ENLACE GitHub : <https://github.com/kevin250313/parcial-bases-datos-kevin.git>**

**Enlace Video:** <https://youtu.be/koY1l-rhL98>

## Metodo Insert:

```
| xyz |  
+-----+  
16 rows in set (0.07 sec)  
  
mysql> use 7502420046_18_appcine_v2;  
Database changed  
mysql> show tables ;  
+-----+  
| Tables_in_7502420046_18_appcine_v2 |  
+-----+  
| adaptaciones  
| adaptaciones_comics  
| adaptaciones_series  
| adaptaciones_videojuegos  
| calificaciones  
| cines  
| clientes  
| dias_festivos  
| festivos  
| funciones  
| funciones_idioma  
| peliculas  
| peliculas_detalle  
| precuelas  
| promociones  
| remakes  
| reservas  
| salas  
| secuelas  
+-----+  
19 rows in set (0.05 sec)  
  
mysql> USE 7502420046_18_appcine_v2;  
Database changed  
mysql>  
mysql> -- INSERT: nueva pelicula de prueba  
mysql> INSERT INTO peliculas (titulo, genero, duracion, clasificacion, idioma_original, pais_origen, anio_produccion)  
    -> VALUES ('Prueba Pelicula Entrega', 'Drama', 95, '+13', 'Español', 'Colombia', 2025);  
Query OK, 1 row affected (0.03 sec)  
  
mysql>
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree, which includes the 'appmuseo' schema containing tables like 'adaptaciones', 'articulos', 'calificaciones', 'cargos', 'cines', 'clientes', 'descuento', 'detalle\_venta', and 'dias\_festivos'. The main area is a 'SQL File 3' editor with the following content:

```
script_unidad3_appmuseo_v2 categoria categoria categoria articulo data_7502420046
14     nombre VARCHAR(200),
15     correo VARCHAR(150),
16     telefono VARCHAR(30)
17 );
18
19 • CREATE TABLE reservas (
20     id INT AUTO_INCREMENT PRIMARY KEY,
21     id_cliente INT,
22     id_pelicula INT,
23     fecha_reserva DATE,
24     cantidad INT,
25     FOREIGN KEY (id_cliente) REFERENCES clientes(id),
26     FOREIGN KEY (id_pelicula) REFERENCES peliculas(id)
27 )
28
29
30 -- INSERT: nueva pelicula de prueba
31 • INSERT INTO peliculas (titulo, genero, duracion, clasificacion, idioma_original, pais_origen, anio_produccion)
32 VALUES ('Prueba Pelicula Entrega', 'Drama', 95, '+13', 'Espanol', 'Colombia', 2025)
33
```

A red box highlights the last two lines of the script, which insert a new movie record into the 'peliculas' table. The bottom right corner of the interface shows the execution time as 0.015 sec.

## Explicación técnica

**Operación:** INSERT (CREATE)

**Objetivo:** Añadir una nueva película de prueba al catálogo (peliculas).

**Campos insertados:** titulo, genero, duracion, clasificacion, idioma\_original, pais\_origen, anio\_produccion.

**Resultado esperado:** Se crea un nuevo registro en la tabla peliculas.

MySQL devuelve Query OK, 1 row affected y el auto\_increment genera pelicula\_id.

## 2) Cómo ejecutar en MySQL Workbench

1. Abre Workbench y conéctate a tu servidor.
2. **File → Open SQL Script...** → selecciona sql/crud\_demo.sql (o pega sólo el bloque INSERT en una nueva pestaña).
3. Selecciona la línea o el bloque (Ctrl+A) y presiona  (Execute).
4. Observa la sección **Action Output** (abajo) — ahí aparecerá el mensaje Query OK, 1 row affected u errores si los hay.

---

## 3) Cómo ejecutar en CLI (Command Line Client) y guardar salida

Abre CMD / PowerShell y ejecuta:

```
mysql -u root -p 7502420046_18_appcine_v2 -e "INSERT INTO peliculas  
(titulo, genero, duracion, clasificacion, idioma_original, pais_origen,  
anio_produccion) VALUES ('Prueba Pelicula Entrega', 'Drama', 95, '+13',  
'Español', 'Colombia', 2025);" > docs/output_insert.txt 2>&1
```

- Te pedirá la contraseña de MySQL.
- El archivo docs/output\_insert.txt contendrá la salida (o el error).

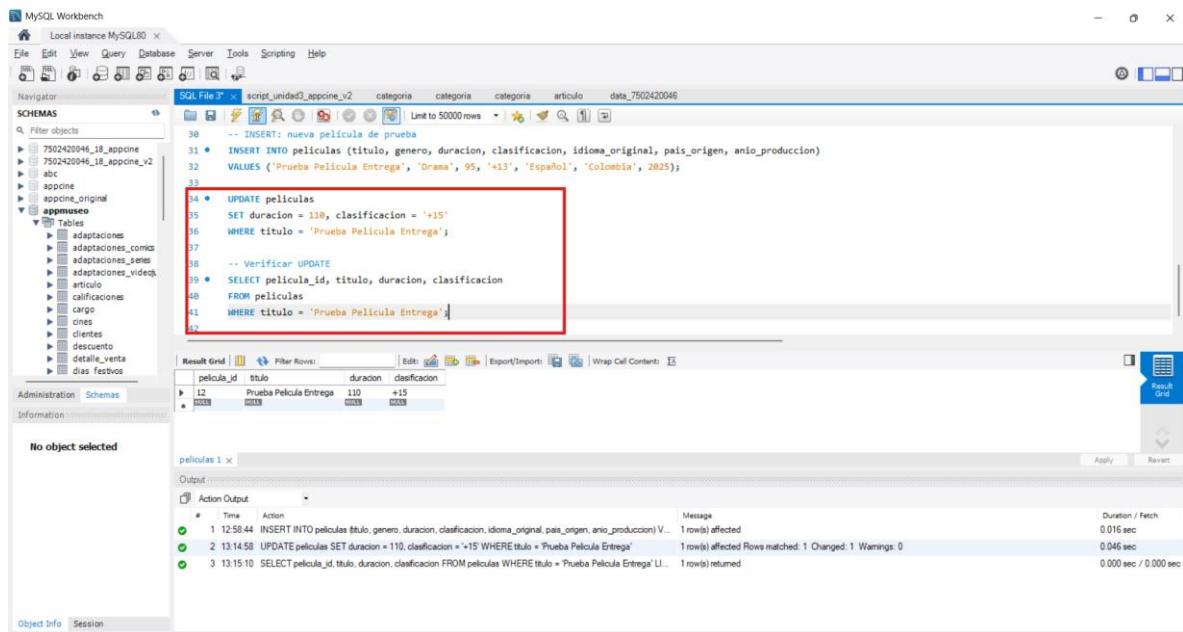
## Librerías / Instalación / Importaciones (texto listo)

Pega esto en la sección correspondiente del informe:

### Librerías / herramientas

- MySQL Server 8.0 → motor de BD.
- MySQL Workbench → ejecución y capturas.
- Cliente CLI (incluido en MySQL).

## Metodo Update :



The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with several tables listed under the 'appmeo' schema. The main area contains a SQL editor window with the following code:

```
-- INSERT: nueva película de prueba
INSERT INTO peliculas (titulo, genero, duracion, clasificacion, idioma_original, pais_origen, anio_produccion)
VALUES ('Prueba Película Entrega', 'Drama', 95, '+13', 'Español', 'Colombia', 2025);

-- UPDATE películas
UPDATE peliculas
SET duracion = 110, clasificacion = '+15'
WHERE titulo = 'Prueba Película Entrega';

-- Verificar UPDATE
SELECT pelicula_id, titulo, duracion, clasificacion
FROM peliculas
WHERE titulo = 'Prueba Película Entrega';
```

Below the SQL editor is a 'Result Grid' showing the updated data:

pelicula_id	titulo	duracion	clasificacion
12	Prueba Película Entrega	110	+15

The 'Output' pane at the bottom displays the execution log:

#	Time	Action	Message	Duration / Fetch
1	12:58:44	INSERT INTO peliculas(titulo, genero, duracion, clasificacion, idioma_original, pais_origen, anio_produccion) VALUES ('Prueba Película Entrega', 'Drama', 95, '+13', 'Español', 'Colombia', 2025);	1 row(s) affected	0.016 sec
2	13:14:58	UPDATE peliculas SET duracion = 110, clasificacion = '+15' WHERE titulo = 'Prueba Película Entrega'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.046 sec
3	13:15:10	SELECT pelicula_id, titulo, duracion, clasificacion FROM peliculas WHERE titulo = 'Prueba Película Entrega' \U00a7	1 row(s) returned	0.000 sec / 0.000 sec

```

mysql> USE 7502420046_18_appcine_v2;
Database changed
mysql>
mysql> -- UPDATE: cambiar duración y clasificación de la película de prueba
mysql> UPDATE peliculas
    -> SET duracion = 110, clasificacion = '+15'
    -> WHERE titulo = 'Prueba Película Entrega';
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> -- Verificar UPDATE
mysql> SELECT pelicula_id, titulo, duracion, clasificacion
    -> FROM peliculas
    -> WHERE titulo = 'Prueba Película Entrega';
+-----+-----+-----+-----+
| pelicula_id | titulo           | duracion | clasificacion |
+-----+-----+-----+-----+
|          12 | Prueba Película Entrega |      110 | +15           |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> |

```

## 2) Explicación técnica para el informe

**Operación:** UPDATE (MODIFY)

**Objetivo:** Modificar los datos de la película de prueba creada en el paso INSERT, cambiando duracion y clasificacion.

**Sentencia:**

UPDATE peliculas

SET duracion = 110, clasificacion = '+15'

WHERE titulo = 'Prueba Película Entrega';

**Resultado esperado:** MySQL devuelve Query OK, X rows affected (X normalmente = 1 si la fila existe). Debes verificar con un SELECT posterior que los valores se actualizaron correctamente.

**Consideraciones:** Siempre usar WHERE para evitar actualizar todas las filas. Para operaciones críticas, envolver en transacción si la BD lo requiere.

## 3) Ejecutar en MySQL Workbench (GUI)

1. Abre Workbench y conecta al servidor.

2. **File → Open SQL Script...** → abre sql/crud\_demo.sql (o sql/update\_demo.sql si lo separaste).
  3. Selecciona el bloque (Ctrl+A) o solo la sección UPDATE y SELECT.
  4. Presiona el botón  (Execute).
  5. Observa en **Action Output**: Query OK, 1 row affected y luego el resultado del SELECT en la pestaña de resultados.
- 

#### 4) Ejecutar en CLI y guardar salida (texto)

Abre **CMD / PowerShell** y copia/pega EXACTO:

```
cd "C:\Users\LENOVO\Documents\GitHub\parcial-bases-datos-kevin"
```

```
# Ejecutar UPDATE y guardar salida (redirección)
```

```
mysql -u root -p 7502420046_18_appcine_v2 -e "UPDATE peliculas SET duracion = 110, clasificacion = '+15' WHERE titulo = 'Prueba Pelicula Entrega'; SELECT pelicula_id, titulo, duracion, clasificacion FROM peliculas WHERE titulo = 'Prueba Pelicula Entrega';" > docs/output_update.txt 2>&1
```

- Te pedirá la contraseña.
- docs/output\_update.txt contendrá la salida. Sube ese archivo al repo como evidencia si prefieres texto sobre imagen.

## Metodo DELETE:

```

mysql>
mysql> DELETE FROM peliculas
    -> WHERE pelicula_id = 5;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails ('7502420046_18_appcine_v2`.`adaptaciones_comics`, CONSTRAINT `adaptaciones_comics_ibfk_1` FOREIGN KEY (`pelicula_id`) REFERENCES `peliculas` (`pelicula_id`))'
mysql> -- COMMIT para aplicar cambios (o ROLLBACK para deshacer si no es correcto)
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql> -- (C) Verificación posterior (SELECT debe devolver 0 filas)
mysql> SELECT pelicula_id, titulo FROM peliculas WHERE pelicula_id = 5;
+-----+-----+
| pelicula_id | titulo |
+-----+-----+
|      5 | Viaje Estelar |
+-----+-----+
1 row in set (0.00 sec)

mysql>

```

## Explicación técnica para el informe

### Operación: DELETE (ELIMINAR)

**Objetivo:** Eliminar registros de la tabla `peliculas` cumpliendo una condición.

### Sentencias usadas:

- `DELETE FROM peliculas WHERE pelicula_id = 5;` — elimina la fila con id 5.
- `DELETE FROM peliculas WHERE duracion < 90;` — ejemplo por regla de negocio.

**Comportamiento esperado:** MySQL devuelve `Query OK, X rows affected`. Si  $X = 0$ , no existía registro que cumpla la condición.

### Buenas prácticas:

- Usar siempre WHERE para no borrar toda la tabla.
- Para operaciones potencialmente destructivas, usar transacciones (START TRANSACTION; ... COMMIT;) y probar con SELECT antes de COMMIT.
- Hacer backup o exportar la tabla si la operación afecta muchos registros.

**Erros comunes y soluciones:**

- 0 rows affected → verificar la existencia con SELECT.
- Violaciones de integridad (FK) al borrar una fila padre que tiene hijos → puede requerir ON DELETE CASCADE o borrar primero los hijos.
- Falta de permisos → GRANT DELETE ON ... TO 'user'@'host';

Pegalo exactamente en la sección DELETE del informe.

---

## C — Cómo ejecutar en MySQL Workbench

Abrir MySQL Workbench y conectarte a tu instancia local.

1. File → Open SQL Script... → selecciona el archivo que contiene el DELETE (sql/crud\_demo.sql o sql/delete\_demo.sql) o pega el bloque DELETE en una nueva pestaña.
2. Selecciona la sentencia DELETE (o todo el bloque con transacción).
3. Presiona el botón **Execute**.
4. Observa en la sección **Action Output**: verás Query OK, X row(s) affected.
5. Ejecuta la verificación con:
6. SELECT pelicula\_id, titulo FROM peliculas WHERE pelicula\_id = 5;  
→ debería devolver 0 filas si fue borrada.

**7. Captura de pantalla:** toma una imagen donde se vea:

- la sentencia DELETE en el editor,
  - el resultado/Action Output con Query OK, X row(s) affected,
  - y si quieras la verificación SELECT mostrando 0 filas.
- 

## D — Cómo ejecutar en Command Line Client

Abre CMD y ejecuta (copia exacto; te pedirá la contraseña):

```
cd "C:\Users\LENOVO\Documents\GitHub\parcial-bases-datos-kevin"
```

```
# Ejecutar DELETE + verificación y guardar toda la salida en  
docs/output_delete.txt
```

```
mysql -u root -p 7502420046_18_appcine_v2 -e "DELETE FROM peliculas  
WHERE pelicula_id = 5; SELECT pelicula_id, titulo FROM peliculas  
WHERE pelicula_id = 5;" > docs/output_delete.txt 2>&1
```

- docs/output\_delete.txt contendrá la respuesta de MySQL (ej. Query OK, 1 row affected y la tabla de verificación).
- Si usaste transacción, ejecuta:

```
mysql -u root -p 7502420046_18_appcine_v2 -e "START TRANSACTION;  
DELETE FROM peliculas WHERE pelicula_id=5; SELECT ROW_COUNT()  
AS filas_afectadas; ROLLBACK;" > docs/output_delete.txt 2>&1
```

(Eso muestra filas\_afectadas y hace ROLLBACK para no perder datos si estás probando.)

## Consultas CAE

### CONSULTA 1: Listar películas ordenadas alfabéticamente

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help' are visible. The 'Query' tab is active. Below the navigation is a toolbar with icons for file operations, search, and refresh. The main area has tabs for 'SQL File 3' and 'script\_unidad3\_apoline\_v2'. The SQL pane contains the following code:

```
118 • SELECT id, titulo, genero
119   FROM peliculas
120  ORDER BY titulo ASC;
```

The Result Grid pane displays the results of the query:

id	titulo	genero
17	Avatar	Sci-Fi
15	Braveheart	History
21	Coco	Animation
25	Deadpool	Action
23	Dune	Sci-Fi
8	Fight Club	Drama
5	Forrest Gump	Drama
11	Gladiator	Action
6	Inception	Sci-Fi
9	Interstellar	Sci-Fi
20	Joker	Crime
24	Jurassic Park	Adventure

The Output pane at the bottom shows the execution log:

Action	Time	Message	Duration / Fetch
20	13:53:42	SELECT id, titulo, genero FROM peliculas WHERE titulo LIKE '%man%' LIMIT 0, 50000	0.000 sec / 0.000 sec
21	13:53:54	SELECT id, titulo, genero FROM peliculas WHERE id = 1 LIMIT 0, 50000	0 row(s) returned
22	14:05:12	INSERT INTO peliculas(titulo, genero) VALUES ('The Shawshank Redemption', 'Drama'), ('The Godfather', 'C...')	25 rows(s) affected Records: 25 Duplicates: 0 Warnings: 0
23	14:05:36	select * from peliculas LIMIT 0, 50000	25 row(s) returned
24	14:07:06	SELECT id, titulo, genero FROM peliculas ORDER BY titulo ASC LIMIT 0, 50000	25 row(s) returned

ORDER BY ordena texto usando orden lexicográfico ASCII.

ASC significa ascendente (A → Z).

#### Objetivo / Por qué se usa

Mostrar el catálogo completo de películas y permitir que la aplicación

despliegue la lista en orden alfabético. Esto es útil para interfaces de tipo “catálogo” o “buscar”.

### **Qué columnas devuelve y por qué**

- id — identificador único (PK) de la película; necesario para enlaces o acciones (ver, editar, reservar).
- titulo — nombre de la película (clave informativa).
- genero — clasificación por género para filtros.

### **Características SQL utilizadas**

- ORDER BY para ordenar resultados.
- No usa WHERE — devuelve todas las filas.

### **Salida esperada**

Tabla con todas las películas, ordenadas por titulo. Si la tabla está vacía, devuelve 0 filas.

### **Errores / casos a revisar**

- 0 rows returned → la tabla está vacía.
- Si el orden debe ser por fecha o popularidad, cambiar ORDER BY.

### **Consulta 2 — Buscar películas por género**

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' is selected. The left sidebar displays the 'Schemas' tree, which includes several databases like '7502420046\_18\_appline' and '7502420046\_18\_appline\_v2'. The main area contains a SQL editor window with the following query:

```

129
130 • SELECT id, titulo, genero
131   FROM peliculas
132  WHERE genero = 'Action'
133
134
135
136

```

The result grid shows the following data:

id	titulo	genero
3	The Dark Knight	Action
11	Gladiator	Action
18	The Avengers	Action
25	Deadpool	Action
26	...	...

Below the result grid, the 'Output' pane shows the execution log:

Time	Action	Message	Duration / Fetch
22 14:05:12	INSERT INTO peliculas(titulo, genero) VALUES ('The Shawshank Redemption', 'Drama'), ('The Godfather', 'Cr...')	25 row(s) affected Records: 25 Duplicates: 0 Warnings: 0	0.031 sec
23 14:05:36	select * From peliculas LIMIT 0, 50000	25 row(s) returned	0.000 sec / 0.000 sec
24 14:07:06	SELECT id, titulo, genero FROM peliculas ORDER BY titulo ASC LIMIT 0, 50000	25 row(s) returned	0.000 sec / 0.000 sec
25 14:10:26	SELECT id, titulo, genero FROM peliculas WHERE genero = 'Acción' LIMIT 0, 50000	0 row(s) returned	0.000 sec / 0.000 sec
26 14:11:20	SELECT id, titulo, genero FROM peliculas WHERE genero = 'Action' LIMIT 0, 50000	4 row(s) returned	0.000 sec / 0.000 sec

## Objetivo / Por qué se usa

Filtrar películas por género para ofrecer secciones temáticas (ej. acción, comedia) en la app o para análisis estadístico.

## Qué columnas devuelve y por qué

- id y titulo para identificar y mostrar la película.
- genero para confirmar el filtro.

## Características SQL utilizadas

- WHERE para filtrar por igualdad de texto. Es sensible al contenido exacto (acentos, mayúsculas según collation).

## Salida esperada

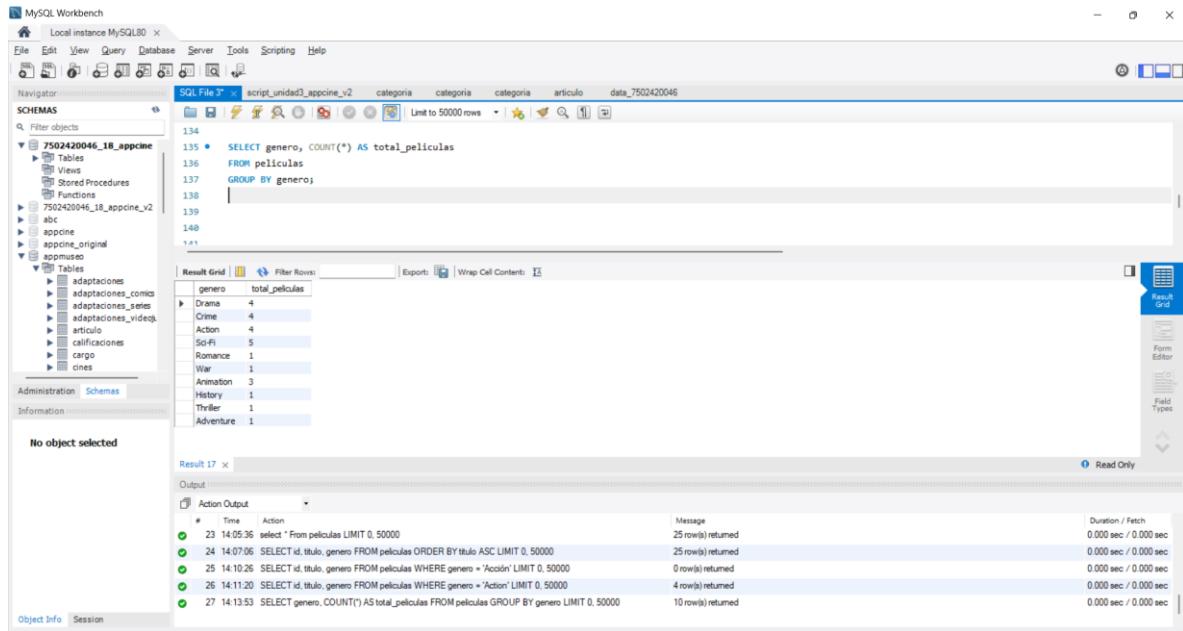
Todas las películas cuyo genero sea exactamente Acción. Si no hay coincidencias, devuelve 0 filas.

## Errores / casos a revisar

- Error de coincidencia por mayúsculas/minúsculas o tildes → usar WHERE genero LIKE '%Accion%' COLLATE utf8mb4\_general\_ci o normalizar valores.

- Si varios géneros están juntos (ej. "Acción, Aventura"), usar LIKE '%Acción%'

## Consulta 3 — Contar cuántas películas hay por género



The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```

134
135 •  SELECT genero, COUNT(*) AS total_peliculas
136   FROM peliculas
137  GROUP BY genero;
138
139
140
141

```

The result grid displays the following data:

genero	total_peliculas
Drama	4
Crime	4
Action	4
Sci-Fi	5
Romance	1
War	1
Animation	3
History	1
Thriller	1
Adventure	1

The output pane shows the following log entries:

Action	Time	Action	Message	Duration / Fetch
23	14:05:36	select * From peliculas LIMIT 0, 50000	25 row(s) returned	0.000 sec / 0.000 sec
24	14:07:06	SELECT id, titulo, genero FROM peliculas ORDER BY titulo ASC LIMIT 0, 50000	25 row(s) returned	0.000 sec / 0.000 sec
25	14:10:26	SELECT id, titulo, genero FROM peliculas WHERE genero = 'Acción' LIMIT 0, 50000	0 row(s) returned	0.000 sec / 0.000 sec
26	14:11:20	SELECT id, titulo, genero FROM peliculas WHERE genero = 'Acción' LIMIT 0, 50000	4 row(s) returned	0.000 sec / 0.000 sec
27	14:13:53	SELECT genero, COUNT(*) AS total_peliculas FROM peliculas GROUP BY genero LIMIT 0, 50000	10 row(s) returned	0.000 sec / 0.000 sec

### Objetivo / Por qué se usa

Obtener un resumen agregado por género; útil para reportes, dashboards y decisiones (p. ej., ver qué géneros predominan).

### Qué columnas devuelve y por qué

- genero — la categoría.
- total\_peliculas — número de películas en cada género.

### Características SQL utilizadas

- GROUP BY para agregar por género.
- COUNT(\*) función de agregación.

### Salida esperada

Una fila por cada género presente y su conteo. Si no hay datos, no devuelve filas.

## Errores / casos a revisar

- Valores inconsistentes en genero (variaciones en texto) inflan el número de grupos. Normalizar géneros al insertar/actualizar o usar LOWER() para agrupar sin distinción de mayúsculas.

## Consulta 4 — Buscar películas cuyo título contiene una palabra

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with several tables under the '7502420046\_18\_appcine' database. The main area shows a query editor with the following SQL code:

```
SELECT id, titulo
FROM peliculas
WHERE titulo LIKE '%the%';
```

The result grid below the query shows 20 rows of movie titles starting with 'the'. The execution history at the bottom lists several previous queries, all of which completed successfully (0.000 sec / 0.000 sec).

## Objetivo / Por qué se usa

Búsqueda parcial por título (matching). Útil para buscadores dentro de la app donde el usuario escribe parte del título.

## Qué columnas devuelve y por qué

- id, titulo, genero para identificar y mostrar coincidencias.

## Características SQL utilizadas

- LIKE '%...%' para búsqueda de subcadenas.
- LEFT/RIGHT wildcard % permite que la palabra aparezca en cualquier parte.

## Salida esperada

Todas las filas con titulo que contenga man (case-insensitive depende de la collation). Si necesitas buscar con acentos o mayúsculas, ajustar collation o usar LOWER(titulo) LIKE '%the%'

## Errores / casos a revisar

- Performance: LIKE '%xxx%' no usa índices; en tablas grandes puede ser lento. Alternativa: índices fulltext o soluciones de búsqueda (no permitidas si solo SQL).
- Falsos positivos si la subcadena aparece en otra palabra.

## Consulta 5 — Consultar película por ID específico

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the schema structure of the database, including tables like 'adaptaciones', 'ad适应aciones\_comics', 'adaptaciones\_series', 'adaptaciones\_videos', 'articulo', 'calificaciones', 'cargo', and 'cines'. The main area shows a SQL editor window with the following query:

```
145 •  SELECT id, titulo, genero
146   FROM peliculas
147  WHERE id = 1
148
149
150
151
152
```

The result grid shows one row of data:

	id	titulo	genero
1	1	The Shawshank Redemption	Drama

Below the result grid is a 'Result Output' panel showing the execution timeline:

#	Time	Action	Message	Duration / Fetch
29	14:16:11	SELECT id, titulo, genero FROM peliculas WHERE titulo LIKE "%lamo%" LIMIT 0, 50000	0 row(s) returned	0.000 sec / 0.000 sec
30	14:16:49	SELECT id, titulo FROM peliculas WHERE titulo LIKE "%the%" LIMIT 0, 50000	8 row(s) returned	0.000 sec / 0.000 sec
31	14:18:50	SELECT id, titulo, genero FROM peliculas WHERE id = 1 LIMIT 0, 50000	1 row(s) returned	0.000 sec / 0.000 sec
32	14:19:01	SELECT id, titulo, genero FROM peliculas WHERE id = 1 LIMIT 0, 50000	1 row(s) returned	0.000 sec / 0.000 sec
33	14:19:09	SELECT id, titulo, genero FROM peliculas WHERE id = 1 LIMIT 0, 50000	1 row(s) returned	0.000 sec / 0.000 sec

## Objetivo / Por qué se usa

Obtener la ficha completa o los datos básicos de una película conocida por su id (ej. al ver detalles en la app).

## **Qué columnas devuelve y por qué**

- id, titulo, genero — datos clave para la vista de detalle.

## **Características SQL utilizadas**

- WHERE id = para buscar por PK (operación rápida que usa índice primario).

## **Salida esperada**

La fila con id = 1 si existe; si no, 0 filas.

## **Errores / casos a revisar**

- 0 rows returned → id no existe; verificar con SELECT MAX(id) FROM peliculas; para ver rango.
- Asegúrate de que id es numérico (no usar comillas en CLI salvo que quieras forzar conversión).