

Informe de resultados

Proyecto Lógica SQL

Patricia Romo Jiménez
Máster Data Analyst – *Hack(io) by The Power*

1. Introducción

El objetivo principal de este proyecto consiste en resolver todas las consultas planteadas, aplicando diferentes métodos y enfoques para llegar a un mismo resultado, lo que permite afianzar la comprensión de las sentencias y optimizar la escritura de las mismas.

Para el desarrollo del proyecto se han tenido en cuenta los siguientes requisitos: manejo de la herramienta DBeaver, consultas sobre tablas individuales, consultas que establecen relaciones entre tablas, uso de subconsultas, creación de vistas, utilización de estructuras temporales y aplicación de buenas prácticas en la escritura de código.

Las herramientas utilizadas han sido **PostgreSQL** como motor de base de datos y **DBeaver** como gestor visual, lo que ha permitido ejecutar, organizar y documentar cada una de las consultas de manera estructurada y eficiente.

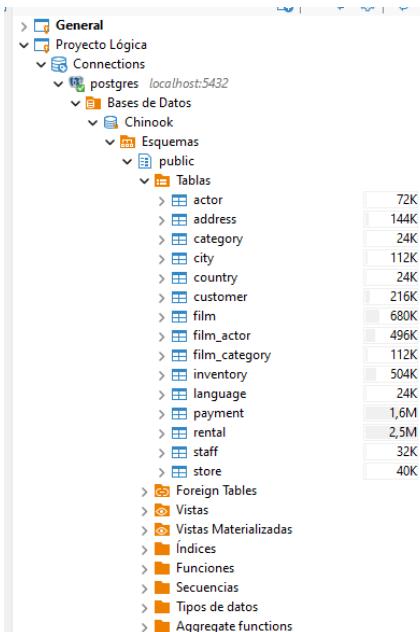
2. Resumen

Se resolvieron **64 consultas SQL** sobre una base relacional de un videoclub ficticio utilizando PostgreSQL y DBeaver.

Principales hallazgos: el catálogo (1.000 pelis) está equilibrado por clasificación con ligera mayoría PG-13; las duraciones oscilan entre 46 y 185 minutos y los largometrajes >180 son minoría; el **coste de reemplazo** presenta **dispersión moderada** (desv. típica ≈ 6); por meses se observa **pico de alquileres en verano**; no hay actores sin películas y el campo original_language_id no está poblado, lo que explica resultados vacíos en la consulta correspondiente. Se identificaron títulos de alta rotación (≥ 10 alquileres) y clientes *top* por gasto total, junto con listados por género (p. ej., Action, Sci-Fi). Todo el código está comentado y versionado para reproducibilidad.

3. Metodología

- **Entorno y datos.** PostgreSQL como motor y DBeaver como gestor visual. Dataset relacional con entidades clave (film, actor, category, inventory, rental, payment, customer, etc.) y tablas puente para relaciones N:M.



Catálogo

- film (una fila por **título**): title, release_year, length (min), rental_rate, replacement_cost, rating, language_id, rental_duration (días).
- language (idiomas) y category (géneros).
- **Puentes**: film_category (film↔category) y film_actor (film↔actor).
- actor (intérpretes).

Inventario y operaciones

- inventory (una fila por **copia** de película en una **tienda**).
- rental (un **evento de alquiler**: rental_date, return_date, customer_id, inventory_id, staff_id).
- payment (un **cobro** asociado a un rental_id con amount y payment_date).

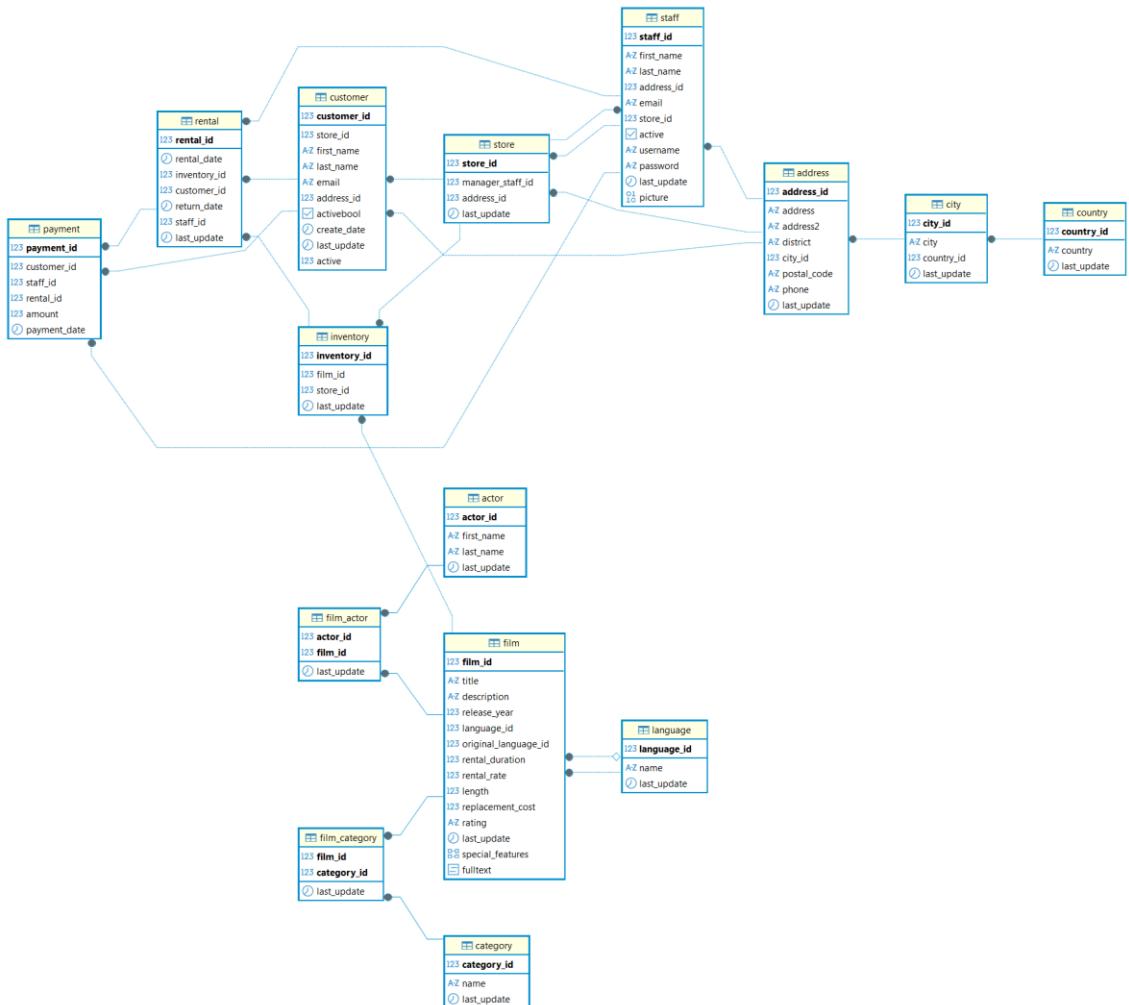
Clientes, personal y localización

- customer y staff, vinculados a store y address; la dirección se normaliza en address → city → country.

➤ Proceso de trabajo.

1. Revisión del **esquema** y claves (PK/FK) para determinar rutas de *join* (especialmente film↔film_category↔category y film↔film_actor↔actor).
2. Desarrollo incremental de consultas **Q01–Q64**, cada una con comentario breve (objetivo y lógica).
3. **Validación** de resultados con totales de control (p. ej., recuentos y extremos), y explicación de vacíos cuando derivan del dataset (p. ej., original_language_id en NULL).
4. **Formateo** y ordenación orientados a lectura (p. ej., TO_CHAR/DATE_TRUNC por fecha; ILIKE para búsquedas *case-insensitive*; CONCAT para nombres).
5. Uso de **agregaciones** (COUNT, SUM, AVG, MIN, MAX) y métricas estadísticas (VARIANCE, STDDEV_SAMP); **ventanas** (ROW_NUMBER() OVER (PARTITION BY ... ORDER BY ...)) para rankings por día; **CTEs** y **vistas** para reutilización.
6. **Joins** según el objetivo: LEFT JOIN para incluir casos con **cero** (p. ej., películas sin inventario o clientes sin alquiler), INNER JOIN cuando se requiere intersección ;CROSS JOIN solo con propósito demostrativo.

4. Resultados
 4.1. Esquema de la Base de Datos.



La base de datos utilizada en este proyecto corresponde a una tienda de películas ficticia. El diseño sigue un modelo relacional en el cual cada entidad principal está representada por una tabla, con sus respectivas claves primarias (Primary Keys) y foráneas (Foreign Keys), garantizando la integridad referencial.

I. Tablas, claves y descripción

Tabla Primary Key (PK) **Foreign Keys (FK)** **Descripción**

<i>actor</i>	actor_id	—	Contiene los datos de los actores.
<i>film</i>	film_id	language_id → language(language_id) original_language_id → language(language_id)	Información de las películas: título, duración, idioma, etc.
<i>category</i>	category_id	—	Almacena los géneros de películas.
<i>film_actor</i>	(film_id, actor_id)	film_id → film(film_id) actor_id → actor(actor_id)	Relaciona películas y actores (N:M).

<i>film_category</i>	(film_id, category_id)	film_id → film(film_id) category_id → category(category_id)	Relaciona películas y categorías (N:M).
<i>language</i>	language_id	—	Idiomas disponibles para las películas.
<i>inventory</i>	inventory_id	film_id → film(film_id) store_id → store(store_id)	Copias físicas de películas en las tiendas.
<i>rental</i>	rental_id	inventory_id → inventory(inventory_id) customer_id → customer(customer_id) staff_id → staff(staff_id)	Registra los alquileres de películas.
<i>payment</i>	payment_id	customer_id → customer(customer_id) staff_id → staff(staff_id) rental_id → rental(rental_id)	Información de los pagos de los clientes.
<i>customer</i>	customer_id	store_id → store(store_id) address_id → address(address_id)	Datos de los clientes.
<i>store</i>	store_id	manager_staff_id → staff(staff_id) address_id → address(address_id)	Representa cada tienda física.
<i>staff</i>	staff_id	address_id → address(address_id) store_id → store(store_id)	Datos de los empleados de la tienda.
<i>address</i>	address_id	city_id → city(city_id)	Información de direcciones.
<i>city</i>	city_id	country_id → country(country_id)	Ciudades disponibles.
<i>country</i>	country_id	—	Lista de países.

II. Relaciones entre tablas

◆ Relaciones 1:1

- **store (1) ↔ address (1):** cada tienda tiene registrada una sola dirección física en la tabla address, y esa dirección pertenece exclusivamente a esa tienda.
- **store (1) ↔ staff (1) (manager_staff_id):** cada tienda está gestionada por un único empleado que actúa como gerente. A su vez, ese empleado solo puede ser gerente de una tienda.
- **customer (1) ↔ address (1):** cada cliente tiene asociada una única dirección, y esa dirección corresponde exclusivamente a ese cliente.
- **staff (1) ↔ address (1):** cada empleado de la tienda tiene una dirección única vinculada a él, y dicha dirección no se comparte con otros empleados.

◆ Relaciones 1:N

- **language (1) → film (N)**: un idioma puede estar asociado a muchas películas, pero cada película tiene definido un único idioma principal.
- **film (1) → inventory (N)**: una película puede tener múltiples copias físicas en diferentes tiendas, pero cada copia pertenece a una sola película.
- **store (1) → inventory (N)**: una tienda puede almacenar muchas copias de películas, pero cada copia está localizada en una sola tienda.
- **inventory (1) → rental (N)**: una copia concreta de película puede ser alquilada muchas veces, pero cada alquiler corresponde a una única copia.
- **customer (1) → rental (N)**: un cliente puede realizar muchos alquileres, pero cada alquiler está vinculado a un único cliente.
- **staff (1) → rental (N)**: un empleado puede gestionar muchos alquileres, pero cada alquiler fue atendido por un único empleado.
- **rental (1) → payment (N)**: un alquiler puede tener varios pagos asociados (por ejemplo, si se fracciona el pago), pero cada pago pertenece a un único alquiler.
- **customer (1) → payment (N)**: un cliente puede generar muchos pagos a lo largo del tiempo, pero cada pago corresponde a un único cliente.
- **staff (1) → payment (N)**: un empleado puede procesar muchos pagos, pero cada pago fue registrado por un único empleado.
- **store (1) → customer (N)**: una tienda puede tener muchos clientes registrados, pero cada cliente está vinculado a una única tienda.
- **store (1) → staff (N)**: una tienda puede tener muchos empleados, pero cada empleado pertenece a una sola tienda.
- **city (1) → address (N)**: una ciudad puede contener muchas direcciones, pero cada dirección pertenece a una única ciudad.
- **country (1) → city (N)**: un país puede estar compuesto por muchas ciudades, pero cada ciudad pertenece a un único país.

◆ Relaciones N:M

- **film (N) ↔ actor (N) (mediante film_actor)**: una película puede tener muchos actores, y un actor puede participar en muchas películas. La tabla film_actor actúa como intermediaria para unir ambas entidades.
- **film (N) ↔ category (N) (mediante film_category)**: una película puede estar clasificada en varios géneros, y un género puede aplicarse a muchas películas. La tabla film_category se utiliza para gestionar esta relación múltiple.

4.2. Consultas

- **Q2:** Muestra los nombres de todas las películas con una clasificación por edades de 'R'

	123 ↗ film_id	AZ title	AZ rating
1	8	AIRPORT POLLOCK	R
2	17	ALONE TRIP	R
3	20	AMELIE HELLFIGHTERS	R
4	21	AMERICAN CIRCUS	R
5	23	ANACONDA CONFESSIONS	R
6	24	ANALYZE HOOSIERS	R
7	30	ANYTHING SAVANNAH	R
8	32	APOCALYPSE FLAMINGOS	R
9	40	ARMY FLINTSTONES	R
10	49	BADMAN DAWN	R

```
⌚ SELECT "film_id", "title", "rating"
  FROM "film" AS f
 WHERE "rating" = 'R'
 ORDER BY "title";
```

Listado completo de pelis con rating **R**; facilita localizar títulos de esa clasificación.

- **Q3:** Encuentra los nombres de los actores que tengan un “actor_id” entre 30 y 40.

	123 ↗ actor_id	AZ concat
1	30	SANDRA PECK
2	31	SISSY SOBIESKI
3	32	TIM HACKMAN
4	33	MILLA PECK
5	34	AUDREY OLIVIER
6	35	JUDY DEAN
7	36	BURT DUKAKIS
8	37	VAL BOLGER
9	38	TOM MCKELLEN
10	39	GOLDIE BRODY
11	40	JOHNNY CAGE

```
⌚ SELECT "actor_id",
      CONCAT(a."first_name", ' ', a."last_name")
    FROM "actor" AS a
   WHERE "actor_id" BETWEEN '30' AND '40'
ORDER BY a."actor_id";
```

Muestra el tramo de actores con **id 30–40**; útil como muestreo ordenado.

- **Q4:** Obtén las películas cuyo idioma coincide con el idioma original.

	123 ↗ film_id	AZ title	123 ↗ language_id	AZ name

```
-- Idiomas de todas las películas
⌚ SELECT f."film_id", f."title", l."language_id", l."name"
  FROM "film" AS f
 JOIN "language" AS l ON f."language_id" = l.language_id;

-- Comparación del idioma de las películas con el idioma original
⌚ SELECT f."film_id", f."title", f."language_id", l."name"
  FROM "film" AS f
 JOIN "language" AS l
    ON f."language_id" = l.language_id
 WHERE f."language_id" = f."original_language_id";

-- No devuelve resultados debido a que "original_language_id" siempre es NULL, por lo que esta consulta devolverá 0 filas.
```

	123 film_id	A-Z title	123 language_id	A-Z name
1	1	ACADEMY DINOSAUR	1	English
2	2	ACE GOLDFINGER	1	English
3	3	ADAPTATION HOLES	1	English
4	4	AFFAIR PREJUDICE	1	English
5	5	AFRICAN EGG	1	English
6	6	AGENT TRUMAN	1	English
7	7	AIRPLANE SIERRA	1	English
8	8	AIRPORT POLLOCK	1	English
9	9	ALABAMA DEVIL	1	English
10	10	ALADDIN CALENDAR	1	English
11	11	ALAMO VIDEOTAPE	1	English
12	12	ALASKA PHANTOM	1	English

No devuelve filas: original_language_id está NULL en todo el catálogo

- Q5: Ordena las películas por duración de forma ascendente.

	123 film_id	A-Z title	123 length
1	504	KWAI HOMEMARD	46
2	505	LABYRINTH LEAGUE	46
3	469	IRON MOON	46
4	15	ALIEN CENTER	46
5	730	RIDGEMONT SUBMARINE	46
6	869	SUSPECTS QUILLS	47
7	398	HANOVER GALAXY	47
8	407	HAWK CHILL	47
9	393	HALLOWEEN NUTS	47
10	784	SHANGHAI TYCOON	47
11	237	DIVORCE SHINING	47
12	247	DOWNHILL ENOUGH	47

Orden ascendente de duración; permite identificar **mínimos** y el rango **corto**.

- Q6: Encuentra el nombre y apellido de los actores que tengan ‘Allen’ en su apellido.

	123 actor_id	A-Z first_name	A-Z last_name
1	118	CUBA	ALLEN
2	145	KIM	ALLEN
3	194	MERYL	ALLEN

Filtro por apellido “Allen”; devuelve solo los actores que lo contienen.

- Q7: Encuentra la cantidad total de películas en cada clasificación de la tabla “film” y muestra la clasificación junto con el recuento.

	A-Z rating	123 total_films
1	G	178
2	PG	194
3	R	195
4	NC-17	210
5	PG-13	223

Filtro por apellido “Allen”; devuelve solo los actores que lo contienen.

- Q8: Encuentra el título de todas las películas que son ‘PG-13’ o tienen una duración mayor a 3 horas en la tabla film.

```

    SELECT "film_id", "title", "rating", "length"
    FROM "film" AS f
    WHERE f."rating" = 'PG-13' OR f."length" > 180
    ORDER BY f."title" ASC;

```

	123 film_id	AZ title	AZ rating	123 length
1	7	AIRPLANE SIERRA	PG-13	62
2	9	ALABAMA DEVIL	PG-13	114
3	18	ALTER VICTORY	PG-13	57
4	24	ANALYZE HOOSIERS	R	181
5	28	ANTHEM LUKE	PG-13	91
6	33	APOLLO TEEN	PG-13	153
7	35	ARACHNOPHOBIA ROLLERCOASTER	PG-13	147
8	36	ARGONAUTS TOWN	PG-13	127
9	44	ATTACKS HATE	PG-13	113
10	45	ATTRACTION NEWTON	PG-13	83
11	48	BACKLASH UNDEFEATED	PG-13	118
12	50	BAKED CLEOPATRA	G	182

Predominan **PG-13**; algunos títulos entran por **>180 min** aunque tengan otro rating.

- **Q9:** Encuentra la variabilidad de lo que costaría reemplazar las películas.

```

    SELECT
        ROUND(VARIANCE(f."replacement_cost"),2) AS "var_replacement_cost",
        ROUND(STDDEV_SAMP(f."replacement_cost"),2) AS "sd_replacement_cost"
    FROM "film" AS f;

```

```

    SELECT ROUND(VARIANCE(f."replacement_cost"),2) AS "var_repla | Enter a S

```

Grilla	123 var_replacement_cost	123 sd_replacement_cost
	36,61	6,05

Variación **moderada**: los costes se desvían ~6 unidades de la media.

- **Q10:** Encuentra la mayor y menor duración de una película de nuestra BBDD.

```

    SELECT MIN ("length") AS "Minima_duracion",
           MAX("length") AS "Maxima_duracion"
    FROM "film" f ;

```

```

    SELECT MIN ("length") AS "Minima_duracion", MAX("length") AS | Enter a S

```

Grilla	123 Minima_duracion	123 Maxima_duracion
	46	185

El catálogo va de **46** a **185** minutos; existen títulos **muy largos**.

- **Q11:** Encuentra lo que costó el antepenúltimo alquiler ordenado por día.

```

WITH ranked AS (
    SELECT
        r.rental_id,
        r.rental_date::date AS rental_day,
        r.rental_date,
        ROW_NUMBER() OVER (
            PARTITION BY r.rental_date::date
            ORDER BY r.rental_date DESC, r.rental_id DESC
        ) AS rn
    FROM rental r
)
SELECT
    rk.rental_day,
    p.amount AS cost
FROM ranked rk
JOIN payment p
    ON p.rental_id = rk.rental_id
WHERE rk.rn = 3
ORDER BY rk.rental_day;

```

```

    WITH ranked AS ( SELECT r.rental_id, r.rental_date::d

```

	123 rental_day	123 cost
1	2005-05-24	0,99
2	2005-05-25	0,99
3	2005-05-26	5,99
4	2005-05-27	2,99
5	2005-05-28	2,99
6	2005-05-29	4,99
7	2005-05-30	2,99
8	2005-05-31	2,99
9	2005-06-14	0,99
10	2005-06-15	1,99
11	2005-06-16	0,99
12	2005-06-17	4,99

El catálogo va de **46** a **185** minutos; existen títulos **muy largos**.

- **Q12:** Encuentra el título de las películas en la tabla “film” que no sean ni ‘NC- 17’ ni ‘G’ en cuanto a su clasificación.

```
SELECT "film_id", "title", "rating"
FROM "film" AS f
WHERE f."rating" NOT IN ('NC-17', 'G')
ORDER BY f."title" ASC;
```

	film_id	AZ title	AZ rating
1	1	ACADEMY DINOSAUR	PG
2	6	AGENT TRUMAN	PG
3	7	AIRPLANE SIERRA	PG-13
4	8	AIRPORT POLLOCK	R
5	9	ALABAMA DEVIL	PG-13
6	12	ALASKA PHANTOM	PG
7	13	ALI FOREVER	PG
8	17	ALONE TRIP	R
9	18	ALTER VICTORY	PG-13
10	19	AMADEUS HOLY	PG
11	20	AMELIE HELLFIGHTERS	R
12	21	AMERICAN CIRCUS	R

Pelis **no** NC-17 ni G; listado complementario ordenado por título.

- **Q13:** Encuentra el promedio de duración de las películas para cada clasificación de la tabla film y muestra la clasificación junto con el promedio de duración.

```
SELECT f."rating", ROUND(AVG(f.length), 2) AS "average_duration"
FROM "film" f
GROUP BY f."rating"
ORDER BY "average_duration" ASC;
```

	AZ rating	123 average_duration
1	G	111,05
2	PG	112,01
3	NC-17	113,23
4	R	118,66
5	PG-13	120,44

Media por rating: **PG-13** la mayor, **G** la menor (≈ 9 min de diferencia).

- **Q14:** Encuentra el título de todas las películas que tengan una duración mayor a 180 minutos.

```
SELECT "film_id", "title", "length"
FROM "film" AS f
WHERE f.length > 180;
```

	film_id	AZ title	length
1	24	ANALYZE HOOSIERS	181
2	50	BAKED CLEOPATRA	182
3	128	CATCH AMISTAD	183
4	141	CHICAGO NORTH	185
5	180	CONSPIRACY SPIRIT	184
6	182	CONTROL ANTHEM	185
7	198	CRYSTAL BREAKING	184
8	212	DARN FORRESTER	185
9	340	FRONTIER CABIN	183
10	349	GANGS PRIDE	185
11	406	HAUNTING PIANIST	181
12	426	HOME PITY	185

Muestra solo **largometrajes** (>180 min); es una **minoría** del catálogo

- **Q15:** ¿Cuánto dinero ha generado en total la empresa?

```
SELECT ROUND(SUM(p.amount), 2) AS "total_revenue"
FROM "payment" AS p;
```

Resultados 1 × SQL Terminal	
<pre>SELECT ROUND(SUM(p."amount"),2) AS '</pre>	
123 total_revenue	67.416,51
1	67.416,51

Suma global de **ingresos**; referencia para KPI de facturación.

- **Q16:** Muestra los 10 clientes con mayor valor de id

customer 1 × SQL Terminal			
<pre>SELECT c.customer_id, c.first_name, c.last_name FROM customer</pre>			
Grilla	123 ▾ customer_id	AZ first_name	AZ last_name
1	599	AUSTIN	CINTRON
2	598	WADE	DELVALLE
3	597	FREDDIE	DUGGAN
4	596	ENRIQUE	FORSYTHE
5	595	TERRENCE	GUNDERSON
6	594	EDUARDO	HIATT
7	593	RENE	MCALISTER
8	592	TERRANCE	ROUSH
9	591	KENT	ARSENault
10	590	SETH	HANNON

Top 10 por **customer_id** más alto; **no** implica mejores clientes.

- **Q17:** Encuentra el nombre y apellido de los actores que aparecen en la película con título ‘Egg Igby’.

```
SELECT a.actor_id, a.first_name, a.last_name, f.title
FROM "film" AS f
JOIN "film_actor" AS fa ON fa.film_id = f.film_id
JOIN "actor" AS a ON a.actor_id = fa.actor_id
WHERE f.title = 'EGG IGYB'
ORDER BY a.last_name, a.first_name;
```

actor 1 × SQL Terminal				
<pre>SELECT a.actor_id, a.first_name, a.last_name, f.title FROM '</pre>				
Grilla	123 ▾ actor_id	AZ first_name	AZ last_name	AZ title
1	154	MERYL	GIBSON	EGG IGYB
2	50	NATALIE	HOPKINS	EGG IGYB
3	162	OPRAH	KILMER	EGG IGYB
4	38	TOM	MCKELLEN	EGG IGYB
5	20	LUCILLE	TRACY	EGG IGYB

Reparte el elenco de “EGG IGYB”; actores ordenados alfabéticamente.

- **Q18:** Selecciona todos los nombres de las películas únicas.

SELECT DISTINCT(title) FROM t	
Grilla	AZ title
1	ACADEMY DINOSAUR
2	ACE GOLDFINGER
3	ADAPTATION HOLES
4	AFFAIR PREJUDICE
5	AFRICAN EGG
6	AGENT TRUMAN
7	AIRPLANE SIERRA
8	AIRPORT POLLOCK
9	ALABAMA DEVIL
10	ALADDIN CALENDAR
11	ALAMO VIDEOTAPE
12	ALASKA PHANTOM

Confirma **títulos únicos** en el catálogo (DISTINCT).

- **Q19:** Encuentra el título de las películas que son comedias y tienen una duración mayor a 180 minutos en la tabla “film”.

```
SELECT f."film_id",
       f."title",
       f."length" AS duration,
       c."name" AS category
  FROM "film" f
 JOIN "film_category" fc ON f."film_id" = fc."film_id"
 JOIN "category" c      ON fc."category_id" = c."category_id"
 WHERE f."length" > 180
   AND c."name" = 'Comedy';
```

	film_id	title	duration	category
Grilla	1	182	CONTROL ANT+	185 Comedy
Texto	2	765	SATURN NAME	182 Comedy
	3	774	SEARCHERS WA	182 Comedy

Comedias >180 min: conjunto **muy reducido** de títulos.

- **Q20:** Encuentra las categorías de películas que tienen un promedio de duración superior a 110 minutos y muestra el nombre de la categoría junto con el promedio de duración.

	category	average_duration
Grilla	1 Sports	128,2
Texto	2 Games	127,84
	3 Foreign	121,7
Record	4 Drama	120,84
	5 Comedy	115,83
	6 Family	114,78
	7 Music	113,65
	8 Travel	113,32
	9 Horror	112,48
	10 Classics	111,67
	11 Action	111,61
	12 New	111,13

```
SELECT c."name" AS category,
       ROUND(AVG(f."length"),2) AS average_duration
  FROM "film" f
 JOIN "film_category" fc ON fc."film_id" = f."film_id"
 JOIN "category" c      ON c."category_id" = fc."category_id"
 GROUP BY c."name"
 HAVING AVG(f."length") > 110
 ORDER BY average_duration DESC;
```

Categorías con **promedio** de duración >110; identifica géneros de **metraje largo**.

- **Q21:** ¿Cuál es la media de duración del alquiler de las películas?

rental_duration_average
4,985

```
SELECT AVG("rental_duration") AS "rental_duration_average"
  FROM "film" f;
```

Media de **días permitidos** de alquiler por película.

- **Q22:** Crea una columna con el nombre y apellidos de todos los actores y actrices.

```
SELECT "actor_id",
       CONCAT ("first_name", ' ', "last_name") AS "full_name"
  FROM "actor" a;
```

SQL Terminal

```
SELECT "actor_id", CONCAT ("first_name", ' ', "last_name")
```

	actor_id	full_name
1	1	PENELOPE GUINNESS
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
10	10	CHRISTIAN GABLE
11	11	ZERO CAGE
12	12	KARL BERRY

Presenta **nombre completo** de actores; mejora legibilidad de resultados.

- **Q23:** Números de alquiler por día, ordenados por cantidad de alquiler de forma descendente.

```
SELECT DATE_TRUNC('day', r."rental_date") AS day, COUNT(*) AS rentals
FROM "rental" AS r
GROUP BY day
ORDER BY "rentals" DESC, day DESC;
```

SQL Terminal

```
SELECT DATE_TRUNC('day', r.rental_date) AS day, COUNT(
```

	day	rentals
1	2005-07-31 00:00:00.000	679
2	2005-08-01 00:00:00.000	671
3	2005-08-21 00:00:00.000	659
4	2005-07-27 00:00:00.000	649
5	2005-08-02 00:00:00.000	643
6	2005-07-29 00:00:00.000	641
7	2005-07-30 00:00:00.000	634
8	2005-08-19 00:00:00.000	628
9	2005-08-22 00:00:00.000	626
10	2005-08-20 00:00:00.000	624
11	2005-08-18 00:00:00.000	621
12	2005-07-28 00:00:00.000	620

Alquileres por día; se aprecian picos de demanda diarios.

- **Q24:** Encuentra las películas con una duración superior al promedio.

```
SELECT f."film_id", f."title", f."length"
FROM "film" AS f
WHERE f."length" > (SELECT AVG("length") FROM "film");
```

SQL Terminal

```
SELECT f."film_id", f."title", f."length" FROM "film" AS f WHERE f."
```

	film_id	title	length
1	4	AFFAIR PREJUDICE	117
2	5	AFRICAN EGG	130
3	6	AGENT TRUMAN	169
4	11	ALAMO VIDEOTAPE	126
5	12	ALASKA PHANTOM	136
6	13	ALI FOREVER	150
7	16	ALLEY EVOLUTION	180
8	21	AMERICAN CIRCUS	129
9	24	ANALYZE HOOSIERS	181
10	27	ANONYMOUS HUMAN	179
11	29	ANTITRUST TOMATOES	168
12	32	APOCALYPSE FLAMINGOS	119

Pelis por encima de la media de duración: la **mitad superior** del catálogo.

- Q25: Averigua el número de alquileres registrados por mes.

The screenshot shows the MySQL Workbench interface with the SQL terminal tab active. The query is:

```
SELECT TO_CHAR(r."rental_date", 'YYYY-MM') AS month,
       COUNT(*) AS rentals
  FROM "rental" r
 GROUP BY TO_CHAR(r."rental_date", 'YYYY-MM')
 ORDER BY month;
```

The results table has columns 'month' and 'rentals'. The data is:

month	rentals
2005-05	1.156
2005-06	2.311
2005-07	6.709
2005-08	5.686
2006-02	182

Alquileres por mes; muestra estacionalidad y tendencia temporal.

- Q26: Encuentra el promedio, la desviación estándar y varianza del total pagado.

```
SELECT ROUND (AVG("amount"),2) AS "promedio",
       ROUND(STDDEV("amount"),2) AS "desviacion_estandar",
       ROUND(VARIANCE ("amount"),2) AS "varianza"
  FROM "payment" p;
```

The screenshot shows the MySQL Workbench interface with the SQL terminal tab active. The query is:

```
SELECT ROUND (AVG("amount"),2) AS "promedio", ROUND(STDDEV | Enter a SQL expression
```

The results table has columns 'promedio', 'desviacion_estandar', and 'varianza'. The data is:

promedio	desviacion_estandar	varianza
4,2	2,36	5,58

Estadísticos de **amount**: media y dispersión de pagos realizados.

- Q27: ¿Qué películas se alquilan por encima del precio medio?

```
SELECT f."film_id", f."title", f."rental_rate"
  FROM "film" f
 WHERE f."rental_rate" > (SELECT AVG("rental_rate") FROM "film");
```

The screenshot shows the MySQL Workbench interface with the SQL terminal tab active. The query is:

```
SELECT f."film_id", f."title", f."rental_rate" FROM "film" f WHERE f | Enter a SQL expression
```

The results table has columns 'film_id', 'title', and 'rental_rate'. The data is:

film_id	title	rental_rate
2	ACE GOLDFINGER	4,99
3	ADAPTATION HOLES	2,99
4	AFFAIR PREJUDICE	2,99
5	AFRICAN EGG	2,99
6	AGENT TRUMAN	2,99
7	AIRPLANE SIERRA	4,99
8	AIRPORT POLLOCK	4,99
9	ALABAMA DEVIL	2,99
10	ALADDIN CALENDAR	4,99
13	ALI FOREVER	4,99
15	ALIEN CENTER	2,99
16	ALLEY EVOLUTION	2,99

Títulos con **precio** de alquiler > **media**; identifica oferta **premium**.

- Q28: Muestra el id de los actores que hayan participado en más de 40 películas.

```
SELECT a."actor_id", a."first_name", a."last_name", COUNT(fa."film_id") AS "films"
  FROM "actor" a
  JOIN "film_actor" fa ON fa."actor_id" = a."actor_id"
 GROUP BY a."actor_id", a."first_name", a."last_name"
 HAVING COUNT(fa."film_id") > 40
 ORDER BY "films" DESC;
```

SQL Terminal | Enter a SQL expression to filter

Grilla	123 ↗ actor_id	A-Z first_name	A-Z last_name	123 films
1	107	GINA	DEGENERES	42
2	102	WALTER	TORN	41

Actores con >40 pelis; ranking de **mayor productividad**.

- **Q29:** Obtener todas las películas y, si están disponibles en el inventario, mostrar la cantidad disponible.

```
> SELECT f."film_id", f."title", COUNT(i."inventory_id") AS "copies"
  FROM "film" f
  LEFT JOIN "inventory" AS i
    ON i."film_id" = f."film_id"
 GROUP BY f."film_id", f.title
 ORDER BY "copies" DESC, f."title";
```

SQL Terminal | Enter a SQL expression to filter

Grilla	123 ↗ film_id	A-Z title	123 copies
1	1	ACADEMY DINOSAUR	8
2	31	APACHE DIVINE	8
3	69	BEVERLY OUTLAW	8
4	73	BINGO TALENTED	8
5	86	BOOGIE AMELIE	8
6	91	BOUND CHEAPER	8
7	103	BUCKET BROTHERHOOD	8
8	109	BUTTERFLY CHOCOLAT	8
9	127	CAT CONEHEADS	8
10	174	CONFIDENTIAL INTERVIEW	8
11	193	CROSSROADS CASUALTIES	8
12	199	CUPBOARD SINNERS	8

Copias en inventario por título; incluye 0 para sin stock.

- **Q30:** Obtener los actores y el número de películas en las que ha actuado.

```
< SELECT a."actor_id", a."first_name", a."last_name", COUNT(fa."film_id") AS "films"
  FROM "actor" a
  LEFT JOIN "film_actor" fa
    ON fa."actor_id" = a."actor_id"
 GROUP BY a."actor_id", a."first_name", a."last_name"
 ORDER BY "films" DESC;
```

SQL Terminal | Enter a SQL expression to filter

Grilla	123 ↗ actor_id	A-Z first_name	A-Z last_name	123 films
1	107	GINA	DEGENERES	42
2	102	WALTER	TORN	41
3	198	MARY	KEITEL	40
4	181	MATTHEW	CARREY	39
5	23	SANDRA	KILMER	37
6	81	SCARLETT	DAMON	36
7	158	VIVIEN	BASINGER	35
8	60	HENRY	BERRY	35
9	144	ANGELA	WITHERSPOON	35
10	37	VAL	BOLGER	35
11	106	GROUCHO	DUNST	35
12	13	UMA	WOOD	35

Nº de pelis por actor; ranking de **participación**.

- **Q31:** Obtener todas las películas y mostrar los actores que han actuado en ellas, incluso si algunas películas no tienen actores asociados.

```

    SELECT f."film_id", f."title", a."actor_id", a."first_name", a."last_name"
    FROM "film" f
    LEFT JOIN "film_actor" fa
        ON fa."film_id" = f."film_id"
    LEFT JOIN "actor" a
        ON a."actor_id" = fa."actor_id"
    ORDER BY f."title", a."last_name" NULLS LAST;

```

SQL Query Result:

	f.film_id	AZ title	123 ~ actor_id	AZ first_name	AZ last_name
1	1	ACADEMY DINOSAUR	40	JOHNNY	CAGE
2	1	ACADEMY DINOSAUR	188	ROCK	DUKAKIS
3	1	ACADEMY DINOSAUR	10	CHRISTIAN	GABLE
4	1	ACADEMY DINOSAUR	1	PENELOPE	GUINNESS
5	1	ACADEMY DINOSAUR	198	MARY	KEITEL
6	1	ACADEMY DINOSAUR	162	OPRAH	KILMER
7	1	ACADEMY DINOSAUR	108	WARREN	NOLTE
8	1	ACADEMY DINOSAUR	30	SANDRA	PECK
9	1	ACADEMY DINOSAUR	53	MENA	TEMPLE
10	1	ACADEMY DINOSAUR	20	LUCILLE	TRACY
11	2	ACE GOLDFINGER	160	CHRIS	DEPP
12	2	ACE GOLDFINGER	19	BOB	FAWCETT

Todas las pelis con sus actores; incluye títulos **sin reparto** si existieran.

- **Q32:** Obtener todos los actores y mostrar las películas en las que han actuado, incluso si algunos actores no han actuado en ninguna película.

```

SELECT a."actor_id", a."first_name", a."last_name", f."film_id", f."title"
FROM "actor" a
LEFT JOIN "film_actor" fa
    ON fa."actor_id" = a."actor_id"
LEFT JOIN "film" f
    ON f."film_id" = fa."film_id"
ORDER BY a."last_name", a."first_name", f."title" NULLS LAST;

```

SQL Query Result:

	a.actor_id	AZ first_name	AZ last_name	123 ~ film_id	AZ title
1	58	CHRISTIAN	AKROYD	48	BACKLASH UNDEFEATED
2	58	CHRISTIAN	AKROYD	68	BETRAYED REAR
3	58	CHRISTIAN	AKROYD	119	CAPER MOTIONS
4	58	CHRISTIAN	AKROYD	128	CATCH AMISTAD
5	58	CHRISTIAN	AKROYD	135	CHANCE RESURRECTION
6	58	CHRISTIAN	AKROYD	175	CONFUSED CANDLES
7	58	CHRISTIAN	AKROYD	199	CUPBOARD SINNERS
8	58	CHRISTIAN	AKROYD	235	DIVIDE MONSTER
9	58	CHRISTIAN	AKROYD	242	DOOM DANCING
10	58	CHRISTIAN	AKROYD	243	DOORS PRESIDENT
11	58	CHRISTIAN	AKROYD	254	DRIVER ANNIE
12	58	CHRISTIAN	AKROYD	306	FEATHERS METAL

Todos los actores con sus pelis; incluye actores **sin participación**.

- **Q33:** Obtener todas las películas que tenemos y todos los registros de alquiler.

```

SELECT
    f."film_id",
    f."title",
    i."inventory_id",
    r."rental_id",
    r."rental_date"
FROM "inventory" i
JOIN "film" f ON f."film_id" = i."film_id"
LEFT JOIN "rental" r ON r."inventory_id" = i."inventory_id"
ORDER BY f."title", r."rental_date";

-- Películas que TENEMOS (inventory) y TODOS sus alquileres (incluye sin alquiler):
-- inventory → film (INNER) y inventory → rental (LEFT)

```

SQL: SELECT f."film_id", f."title", i."inventory_id", r."rental_id", r."rental_date" | Enter a SQL expression to filter results (use Ctrl+Space)

	123 ↗ film_id	AZ title	123 ↗ inventory_id	123 ↗ rental_id	⌚ rental_date
1	1	ACADEMY DINOSAUR	6	361	2005-05-27 07:03:28.000
2	1	ACADEMY DINOSAUR	2	972	2005-05-30 20:21:07.000
3	1	ACADEMY DINOSAUR	7	1.210	2005-06-15 02:57:51.000
4	1	ACADEMY DINOSAUR	2	2.117	2005-06-17 20:24:00.000
5	1	ACADEMY DINOSAUR	6	3.201	2005-06-21 00:30:26.000
6	1	ACADEMY DINOSAUR	2	4.187	2005-07-07 10:41:31.000
7	1	ACADEMY DINOSAUR	6	4.390	2005-07-07 20:59:06.000
8	1	ACADEMY DINOSAUR	1	4.863	2005-07-08 19:03:15.000
9	1	ACADEMY DINOSAUR	7	5.766	2005-07-10 13:07:31.000
10	1	ACADEMY DINOSAUR	6	7.168	2005-07-27 07:51:11.000
11	1	ACADEMY DINOSAUR	7	8.510	2005-07-29 09:41:38.000
12	1	ACADEMY DINOSAUR	2	9.449	2005-07-30 22:02:34.000

PELIS que tenemos (inventario) y todos sus alquileres históricos.

- **Q34:** Encuentra los 5 clientes que más dinero se hayan gastado con nosotros.

```
SELECT c."customer_id", c."first_name", c."last_name", ROUND(SUM(p."amount"), 2) AS "total_spent"
FROM "customer" c
JOIN "payment" p
    ON p."customer_id" = c."customer_id"
GROUP BY c."customer_id", c."first_name", c."last_name"
ORDER BY "total_spent" DESC
LIMIT 5;
```

SQL: SELECT c."customer_id", c."first_name", c."last_name", ROUND(SUM(p."amount"), 2) AS "total_spent" | Enter a SQL expression to filter results (

	123 ↗ customer_id	AZ first_name	AZ last_name	123 total_spent
1	526	KARL	SEAL	221,55
2	148	ELEANOR	HUNT	216,54
3	144	CLARA	SHAW	195,58
4	137	RHONDA	KENNEDY	194,61
5	178	MARION	SNYDER	194,61

Top-5 clientes por gasto total; identifica clientes más valiosos.

- **Q35:** Selecciona todos los actores cuyo primer nombre es 'Johnny'.

```
SELECT "actor_id", "first_name", "last_name"
FROM "actor" a
WHERE a."first_name" ILIKE 'Johnny';
```

SQL Terminal | Enter a SQL expression to filter results (

	123 ↗ actor_id	AZ first_name	AZ last_name
1	5	JOHNNY	LOLLOBRIGIDA
2	40	JOHNNY	CAGE

Filtro de nombre "Johnny"; útil para homónimos.

- **Q36:** Renombra la columna "first_name" como Nombre y "last_name" como Apellido.

```
SELECT "first_name" AS nombre, "last_name" AS apellido
FROM "actor" a;
```

	A-Z nombre	A-Z apellido
1	PENELOPE	GUINNESS
2	NICK	WAHLBERG
3	ED	CHASE
4	JENNIFER	DAVIS
5	JOHNNY	LOLLOBRIGIDA
6	BETTE	NICHOLSON
7	GRACE	MOSTEL
8	MATTHEW	JOHANSSON
9	JOE	SWANK
10	CHRISTIAN	GABLE
11	ZERO	CAGE
12	KARL	BERRY

Alias **Nombre/Apellido**; salida más **amigable**

- **Q37:** Encuentra el ID del actor más bajo y más alto en la tabla actor.

```
SELECT MIN ("actor_id") AS "Id_lower",
       MAX ("actor_id") AS "Id_Higher"
  FROM "actor" a
 LIMIT 1;
```

	123 Id_lower	123 Id_Higher
1	1	200

Rango de ids de actor: mínimo y máximo existentes.

- **Q38:** Cuenta cuántos actores hay en la tabla “actor”.

	123 Total_actors
1	200

Total de actores en la base de datos.

- **Q39:** Selecciona todos los actores y ordénalos por apellido en orden ascendente.

```
SELECT "actor_id", "first_name", "last_name"
  FROM "actor" a
 ORDER BY a."last_name" ASC;
```

	123 actor_id	A-Z first_name	A-Z last_name
1	182	DEBBIE	AKROYD
2	92	KIRSTEN	AKROYD
3	58	CHRISTIAN	AKROYD
4	194	MERYL	ALLEN
5	145	KIM	ALLEN
6	118	CUBA	ALLEN
7	76	ANGELINA	ASTAIRE
8	112	RUSSELL	BACALL
9	190	AUDREY	BAILEY
10	67	JESSICA	BAILEY
11	115	HARRISON	BALE
12	187	RENEE	BALL

Directorio alfabetico de actores por apellido.

- Q40: Selecciona las primeras 5 películas de la tabla “film”.

```
SELECT "film_id", "title"
FROM "film" f
LIMIT 5;
```

film_id	title
1	ACADEMY DINOSAUR
2	ACE GOLDFINGER
3	ADAPTATION HOLES
4	AFFAIR PREJUDICE
5	AFRICAN EGG

Primeras 5 pelis (por id); muestra **muestra** básica.

- Q41; Agrupa los actores por su nombre y cuenta cuántos actores tienen el mismo nombre. ¿Cuál es el nombre más repetido?

```
SELECT a."first_name", COUNT(*) AS n
FROM "actor" a
GROUP BY a."first_name"
ORDER BY n DESC, a."first_name"
LIMIT 1;
```

first_name	n
JULIA	4

Nombre de pila más repetido entre actores (top-1).

- Q42: Encuentra todos los alquileres y los nombres de los clientes que los realizaron.

```
SELECT r."rental_id", r."rental_date", c."customer_id", c."first_name", c."last_name"
FROM "rental" r
JOIN "customer" c
    ON c."customer_id" = r."customer_id"
ORDER BY r."rental_date" DESC;
```

rental_id	rental_date	customer_id	first_name	last_name
11.739	2006-02-14 15:16:03.000	373	LOUIS	LEONE
14.616	2006-02-14 15:16:03.000	532	NEIL	RENNER
11.676	2006-02-14 15:16:03.000	216	NATALIE	MEYER
15.966	2006-02-14 15:16:03.000	374	JEREMY	HURTADO
13.486	2006-02-14 15:16:03.000	274	NAOMI	JENNINGS
15.894	2006-02-14 15:16:03.000	168	REGINA	BERRY
14.928	2006-02-14 15:16:03.000	472	GREG	ROBINS
15.430	2006-02-14 15:16:03.000	282	JENNY	CASTRO
14.204	2006-02-14 15:16:03.000	287	BECKY	MILES
13.578	2006-02-14 15:16:03.000	352	ALBERT	CROUSE
14.531	2006-02-14 15:16:03.000	568	ALBERTO	HENNING
13.464	2006-02-14 15:16:03.000	576	MORRIS	MCCARTER

Alquileres con **nombre de cliente**; vista transaccional legible.

- Q43: Muestra todos los clientes y sus alquileres si existen, incluyendo aquellos que no tienen alquileres.

```
SELECT c."customer_id", c."first_name", c."last_name", r."rental_id", r."rental_date"
FROM "customer" c
LEFT JOIN "rental" r
    ON r."customer_id" = c."customer_id"
ORDER BY c."customer_id", r."rental_date" NULLS LAST;
```

customer(+ 1) X SQL Terminal

```
SELECT c."customer_id", c."first_name", c."last_name", r."rental_id" Enter a SQL expression to filter results (use Ctrl+Space)
```

Grilla	123 ↗ customer_id	AZ first_name	AZ last_name	123 ↗ rental_id	⌚ rental_date
Texto	1	MARY	SMITH	76	2005-05-25 11:30:37.000
Record	2	MARY	SMITH	573	2005-05-28 10:35:23.000
	3	MARY	SMITH	1.185	2005-06-15 00:54:12.000
	4	MARY	SMITH	1.422	2005-06-15 18:02:53.000
	5	MARY	SMITH	1.476	2005-06-15 21:08:46.000
	6	MARY	SMITH	1.725	2005-06-16 15:18:57.000
	7	MARY	SMITH	2.308	2005-06-18 08:41:48.000
	8	MARY	SMITH	2.363	2005-06-18 13:33:59.000
	9	MARY	SMITH	3.284	2005-06-21 06:24:45.000
	10	MARY	SMITH	4.526	2005-07-08 03:17:05.000
	11	MARY	SMITH	4.611	2005-07-08 07:33:56.000
	12	MARY	SMITH	5.244	2005-07-08 13:24:07.000

Todos los **clientes** y, si tienen, sus alquileres (incluye **sin** alquiler).

- **Q44:** Realiza un CROSS JOIN entre las tablas film y category. ¿Aporta valor esta consulta? ¿Por qué? Deja después de la consulta la contestación.

film(+ 1) X SQL Terminal

```
SELECT f."title", c."name" AS "Category"
FROM "film" f
CROSS JOIN "category" c;
```

Grilla	AZ title	AZ Category
Texto	1 ACADEMY DINOSAUR	Action
Record	2 ACADEMY DINOSAUR	Animation
	3 ACADEMY DINOSAUR	Children
	4 ACADEMY DINOSAUR	Classics
	5 ACADEMY DINOSAUR	Comedy
	6 ACADEMY DINOSAUR	Documentary
	7 ACADEMY DINOSAUR	Drama
	8 ACADEMY DINOSAUR	Family
	9 ACADEMY DINOSAUR	Foreign
	10 ACADEMY DINOSAUR	Games
	11 ACADEMY DINOSAUR	Horror
	12 ACADEMY DINOSAUR	Music

CROSS JOIN film×category: combinaciones teóricas; **no aporta** análisis directo.

- **Q45:** Encuentra los actores que han participado en películas de la categoría 'Action'.

actor 1 X SQL Terminal

```
SELECT DISTINCT a."actor_id", a."first_name", a."last_name"
FROM "actor" a
JOIN "film_actor" fa
    ON fa."actor_id" = a."actor_id"
JOIN "film_category" fc
    ON fc."film_id" = fa."film_id"
JOIN "category" c
    ON c."category_id" = fc."category_id"
WHERE c."name" = 'Action'
ORDER BY a."last_name", a."first_name";
```

Grilla	123 ↗ actor_id	AZ first_name	AZ last_name
Texto	1 58	CHRISTIAN	AKROYD
Record	2 92	KIRSTEN	AKROYD
	3 145	KIM	ALLEN
	4 194	MERYL	ALLEN
	5 76	ANGELINA	ASTAIRE
	6 112	RUSSELL	BACALL
	7 190	AUDREY	BAILEY
	8 67	JESSICA	BAILEY
	9 187	RENEE	BALL
	10 47	JULIA	BARRYMORE
	11 158	VIVIEN	BASINGER
	12 174	MICHAEL	BENING

Actores con participación en **Acción**; listado **único**.

- **Q46:** Encuentra todos los actores que no han participado en películas.

```
SELECT a."actor_id", a."first_name", a."last_name"
FROM "actor" a
LEFT JOIN "film_actor" fa
    ON fa."actor_id" = a."actor_id"
WHERE fa."actor_id" IS NULL;
```

-- En este dataset, todos los actores ha participado al menos en una película, por eso no aparece ningún resultado

actor 1 X SQL Terminal			
SELECT a."actor_id", a."first_name", a."last_name" FROM "actor" a ↵ ↴ Enter a SQL			
① 123 ~ actor_id	AZ first_name	AZ last_name	

En Sakila **no hay** actores sin películas; consulta **vacía**.

- **Q47:** Selecciona el nombre de los actores y la cantidad de películas en las que han participado.

```
② SELECT a."actor_id", a."first_name", a."last_name", COUNT(fa."film_id") AS "film_count"
  FROM actor a
  LEFT JOIN "film_actor" fa
    ON fa."actor_id" = a."actor_id"
 GROUP BY a."actor_id", a."first_name", a."last_name"
 ORDER BY "film_count" DESC;
```

Grilla	123 ~ actor_id	AZ first_name	AZ last_name	123 film_count
1	107	GINA	DEGENERES	42
2	102	WALTER	TORN	41
3	198	MARY	KEITEL	40
4	181	MATTHEW	CARREY	39
5	23	SANDRA	KILMER	37
6	81	SCARLETT	DAMON	36
7	158	VIVIEN	BASINGER	35
8	60	HENRY	BERRY	35
9	144	ANGELA	WITHERSPOON	35
10	37	VAL	BOLGER	35
11	106	GROUCHO	DUNST	35
12	13	UMA	WOOD	35

Actores con nº de pelis; ranking descendente.

- **Q48:** Crea una vista llamada “actor_num_películas” que muestre los nombres de los actores y el número de películas en las que han participado.

```
--Vista
CREATE OR REPLACE VIEW actor_num_películas AS
SELECT a."actor_id",
       a."first_name",
       a."last_name",
       COUNT(fa."film_id") AS film_count
  FROM "actor" a
  LEFT JOIN "film_actor" fa
    ON fa."actor_id" = a."actor_id"
 GROUP BY a."actor_id", a."first_name", a."last_name";
```

Name	Value
Updated Rows	0
Execute time	0.062s
Start time	Tue Sep 09 19:29:24 CEST 2025
Finish time	Tue Sep 09 19:29:24 CEST 2025
CREATE OR REPLACE VIEW actor_num_películas AS	
SELECT a."actor_id",	
a."first_name",	
a."last_name",	
COUNT(fa."film_id") AS film_count	
FROM "actor" a	
LEFT JOIN "film_actor" fa	
ON fa."actor_id" = a."actor_id"	
GROUP BY a."actor_id", a."first_name", a."last_name"	

```
--Comprobación de la vista
SELECT * FROM actor_num_películas ORDER BY film_count DESC;
```

actor_num_películas 1 X SQL Terminal

Connection: postgres
Time: 2025-09-09 19:30:36.883
Query: SELECT * FROM actor_num_películas ORDER BY film_count DESC

Grilla Texto Record

	1	107	GINA	DEGENERES	42
1	107	GINA	DEGENERES	42	
2	102	WALTER	TORN	41	
3	198	MARY	KEITEL	40	
4	181	MATTHEW	CARREY	39	
5	23	SANDRA	KILMER	37	
6	81	SCARLETT	DAMON	36	
7	158	VIVIEN	BASINGER	35	
8	60	HENRY	BERRY	35	
9	144	ANGELA	WITHERSPOON	35	
10	37	VAL	BOLGER	35	
11	106	GROUCHO	DUNST	35	
12	13	UMA	WOOD	35	

Vista con nº de pelis por actor; útil para reutilizar en informes.

- **Q49:** Calcula el número total de alquileres realizados por cada cliente.

```
SELECT c."customer_id", c."first_name", c."last_name", COUNT(r."rental_id") AS "rentals"
FROM "customer" AS c
LEFT JOIN "rental" AS r
    ON r."customer_id" = c."customer_id"
GROUP BY c."customer_id", c."first_name", c."last_name"
ORDER BY "rentals" DESC;
```

customer 1 X SQL terminal

SELECT c."customer_id", c."first_name", c."last_name", COUNT(r."rental_id") AS "rentals"

Grilla Texto Record

	123	customer_id	AZ first_name	AZ last_name	123 rentals
1	148	ELEANOR	HUNT	46	
2	526	KARL	SEAL	45	
3	144	CLARA	SHAW	42	
4	236	MARCIA	DEAN	42	
5	75	TAMMY	SANDERS	41	
6	469	WESLEY	BULL	40	
7	197	SUE	PETERS	40	
8	137	RHONDA	KENNEDY	39	
9	178	MARION	SNYDER	39	
10	468	TIM	CARY	39	
11	459	TOMMY	COLLAZO	38	
12	5	ELIZABETH	BROWN	38	

Alquileres totales por cliente; mide actividad individual.

- **Q50:** Calcula la duración total de las películas en la categoría 'Action'.

```
SELECT SUM(f."length") AS total_duration_action
FROM "film" f
JOIN "film_category" fc ON fc."film_id" = f."film_id"
JOIN "category" c ON c."category_id" = fc."category_id"
WHERE c."name" = 'Action';
```

Resultados 1 X SQL Terminal

SELECT SUM("length") AS "total_duration_action"

Grilla

	123 total_duration_action
1	5.504

Metraje total del género Action.

- **Q51:** Crea una tabla temporal llamada “cliente_rentas_temporal” para almacenar el total de alquileres por cliente.

```
WITH "cliente_rentas_temporal" AS (
    SELECT r."customer_id", COUNT(*) AS "total_rentas"
    FROM "rental" AS r
    GROUP BY r."customer_id"
)
SELECT c."customer_id", c."first_name", c."last_name", crt."total_rentas"
FROM "cliente_rentas_temporal" AS crt
JOIN "customer" AS c
    ON c."customer_id" = crt."customer_id"
ORDER BY crt."total_rentas" DESC;
```

WITH "cliente_rentas_temporal" AS (SELECT r."customer_id", COL

	customer_id	first_name	last_name	total_rentas
1	148	ELEANOR	HUNT	46
2	526	KARL	SEAL	45
3	144	CLARA	SHAW	42
4	236	MARCIA	DEAN	42
5	75	TAMMY	SANDERS	41
6	197	SUE	PETERS	40
7	469	WESLEY	BULL	40
8	178	MARION	SNYDER	39
9	137	RHONDA	KENNEDY	39
10	468	TIM	CARY	39
11	410	CURTIS	IRBY	38
12	459	TOMMY	COLLAZO	38

CTE/tabla temporal: total de alquileres por cliente, listo para unir

- **Q52:** Crea una tabla temporal llamada “peliculas_alquiladas” que almacene las películas que han sido alquiladas al menos 10 veces.

WITH "peliculas_alquiladas" AS (

```
WITH "peliculas_alquiladas" AS (
  SELECT i."film_id", COUNT(*) AS "veces_alquilada"
  FROM "rental" AS r
  JOIN "inventory" AS i
    ON i."inventory_id" = r."inventory_id"
  GROUP BY i."film_id"
  HAVING COUNT(*) >= 10
)
SELECT f."film_id", f."title", pa."veces_alquilada"
FROM "peliculas_alquiladas" AS pa
JOIN "film" AS f
  ON f."film_id" = pa."film_id"
ORDER BY pa."veces_alquilada" DESC;
```

SQL terminal

	film_id	title	veces_alquilada
1	103	BUCKET BROTHERHOOD	34
2	738	ROCKETEER MOTHER	33
3	767	SCALAWAG DUCK	32
4	382	GRIT CLOCKWORK	32
5	730	RIDGE MONT SUBMARINE	32
6	489	JUGGLER HARDLY	32
7	331	FORWARD TEMPLE	32
8	973	WIFE TURN	31
9	31	APACHE DIVINE	31
10	369	GOODFELLAS SALUTE	31
11	753	RUSH GOODFELLAS	31
12	1.000	ZORRO ARK	31

Títulos con ≥ 10 alquileres; identifica **alta rotación**.

- **Q53:** Encuentra el título de las películas que han sido alquiladas por el cliente con el nombre ‘Tammy Sanders’ y que aún no se han devuelto. Ordena los resultados alfabéticamente por título de película.

```
SELECT DISTINCT f."film_id", f."title"
FROM "customer" c
JOIN "rental" r ON r."customer_id" = c."customer_id"
JOIN "inventory" i ON i."inventory_id" = r."inventory_id"
JOIN "film" f ON f."film_id" = i."film_id"
WHERE c."first_name" ILIKE 'tammy'
  AND c."last_name" ILIKE 'sanders'
  AND r."return_date" IS NULL
ORDER BY f."title";
```

-- Tammy Sanders tiene actualmente 3 películas sin devolver.

film 1 × SQL Terminal

SELECT DISTINCT f."film_id", f."title" FROM "customer"

	film_id	title
1	542	LUST LOCK
2	806	SLEEPY JAPANESE
3	914	TROUBLE DATE

Pelis pendientes de devolución de Tammy Sanders; orden alfabetico.

- **Q54:** Encuentra los nombres de los actores que han actuado en al menos una película que pertenece a la categoría ‘Sci-Fi’. Ordena los resultados alfabéticamente por apellido.

```
SELECT DISTINCT a."actor_id", a."first_name", a."last_name"
FROM "actor" AS a
JOIN "film_actor" AS fa
  ON fa."actor_id" = a."actor_id"
JOIN "film_category" AS fc
  ON fc."film_id" = fa."film_id"
JOIN "category" AS c
  ON c."category_id" = fc."category_id"
WHERE c."name" = 'Sci-Fi'
ORDER BY a."last_name", a."first_name";
```

	actor_id	first_name	last_name
1	58	CHRISTIAN	AKROYD
2	182	DEBBIE	AKROYD
3	92	KIRSTEN	AKROYD
4	118	CUBA	ALLEN
5	145	KIM	ALLEN
6	76	ANGELINA	ASTAIRE
7	112	RUSSELL	BACALL
8	190	AUDREY	BAILEY
9	67	JESSICA	BAILEY
10	115	HARRISON	BALE
11	187	RENEE	BALL
12	47	JULIA	BARRYMORE

Actores con al menos una peli Sci-Fi; orden por apellido.

- **Q55:** Encuentra el nombre y apellido de los actores que han actuado en películas que se alquilaron después de que la película ‘Spartacus Cheaper’ se alquilara por primera vez. Ordena los resultados alfabéticamente por apellido.

```
-- Actores de películas alquiladas después del primer alquiler de "SPARTACUS CHEAPER"
WITH first_rent AS (
    SELECT MIN(r."rental_date") AS first_date
    FROM "rental" r
    JOIN "inventory" i ON i."inventory_id" = r."inventory_id"
    JOIN "film" f ON f."film_id" = i."film_id"
    WHERE f."title" = 'SPARTACUS CHEAPER'
)
SELECT DISTINCT a."actor_id", a."first_name", a."last_name"
FROM "rental" r
JOIN "inventory" i ON i."inventory_id" = r."inventory_id"
JOIN "film" f ON f."film_id" = i."film_id"
JOIN "film_actor" fa ON fa."film_id" = f."film_id"
JOIN "actor" a ON a."actor_id" = fa."actor_id"
CROSS JOIN first_rent fr
WHERE r."rental_date" > fr.first_date
ORDER BY a."last_name", a."first_name";
```

	actor_id	first_name	last_name
1	58	CHRISTIAN	AKROYD
2	182	DEBBIE	AKROYD
3	92	KIRSTEN	AKROYD
4	118	CUBA	ALLEN
5	145	KIM	ALLEN
6	194	MERYL	ALLEN
7	76	ANGELINA	ASTAIRE
8	112	RUSSELL	BACALL
9	190	AUDREY	BAILEY
10	67	JESSICA	BAILEY
11	115	HARRISON	BALE
12	187	RENEE	BALL

Actores de pelis alquiladas **después** del primer alquiler de *Spartacus Cheaper*.

- **Q56:** Encuentra el nombre y apellido de los actores que no han actuado en ninguna película de la categoría ‘Music’.

```
SELECT a."actor_id", a."first_name", a."last_name"
FROM "actor" AS a
WHERE NOT EXISTS (
    SELECT 1
    FROM "film_actor" AS fa
    JOIN "film_category" AS fc
        ON fc."film_id" = fa."film_id"
    JOIN "category" AS ct
        ON ct."category_id" = fc."category_id"
    WHERE fa."actor_id" = a."actor_id"
        AND ct."name" = 'Music'
);
```

SQL Terminal

```
SELECT a."actor_id", a."first_name", a."last_name" FROM "actor"
```

	actor_id	first_name	last_name
1	151	GEOFFREY	HESTON
2	26	RIP	CRAWFORD
3	137	MORGAN	WILLIAMS
4	93	ELLEN	PRESLEY
5	135	RITA	REYNOLDS
6	39	GOLDIE	BRODY
7	66	MARY	TANDY
8	89	CHARLIZE	DENCH
9	33	MILLA	PECK
10	31	SISSY	SOBIESKI
11	143	RIVER	DEAN
12	163	CHRISTOPHER	WEST

Actores sin participación en Music (NOT EXISTS).

- **Q57:** Encuentra el título de todas las películas que fueron alquiladas por más de 8 días.

```
SELECT DISTINCT f."film_id", f."title", r."rental_date", r."return_date"
FROM "rental" AS r
JOIN "inventory" AS i
    ON i."inventory_id" = r."inventory_id"
JOIN film AS f
    ON f."film_id" = i."film_id"
WHERE r."return_date" IS NOT NULL
    AND (r."return_date" - r."rental_date") > INTERVAL '8 days';
```

SQL Terminal

```
SELECT DISTINCT f."film_id", f."title", r."rental_date", r."return_date"
```

	film_id	title	rental_date	return_date
1	117	CANDLES GRAPES	2005-06-16 14:29:59.000	2005-06-24 17:27:59.000
2	103	BUCKET BROTHERHOOD	2005-06-17 17:35:10.000	2005-06-26 13:52:10.000
3	489	JUGGLER HARDLY	2005-08-22 01:32:32.000	2005-08-31 06:21:32.000
4	132	CHAINSAW UPTOWN	2005-08-01 13:18:23.000	2005-08-10 13:33:23.000
5	258	DRUMS DYNAMITE	2005-06-20 07:51:51.000	2005-06-28 10:14:51.000
6	159	CLOSER BANG	2005-07-10 18:39:01.000	2005-07-19 21:36:01.000
7	877	TAXI KICK	2005-08-21 07:32:35.000	2005-08-30 05:53:35.000
8	322	FLATLINERS KILLER	2005-06-18 07:04:36.000	2005-06-27 03:00:36.000
9	314	FIGHT JAWBREAKER	2005-08-19 13:07:12.000	2005-08-28 18:12:12.000
10	196	CRUELTY UNFORGIVEN	2005-07-30 04:09:13.000	2005-08-07 22:30:13.000
11	326	FLYING HOOK	2005-07-28 06:49:35.000	2005-08-06 08:16:35.000
12	619	NEIGHBORS CHARADE	2005-08-21 08:06:30.000	2005-08-29 10:12:30.000

Títulos con alquileres de >8 días; estancias largas.

- **Q58:** Encuentra el título de todas las películas que son de la misma categoría que 'Animation'.

SQL Terminal

```
SELECT f."film_id", f."title" FROM "film" AS f WHERE f."film_id" IN
```

	film_id	title
1	18	ALTER VICTORY
2	23	ANACONDA CONFESSIONS
3	36	ARGONAUTS TOWN
4	70	BIKINI BORROWERS
5	78	BLACKOUT PRIVATE
6	89	BORROWERS BEDAZZLED
7	118	CANYON STOCK
8	121	CAROL TEXAS
9	134	CHAMPION FLATLINERS
10	154	CLASH FREDDY
11	160	CLUB GRAFFITI
12	193	CROSSROADS CASUALTIES

```
SELECT f."film_id", f."title"
FROM "film" AS f
WHERE f."film_id" IN (
    SELECT fc."film_id"
    FROM "film_category" AS fc
    WHERE fc."category_id" IN (
        SELECT c."category_id"
        FROM "category" AS c
        WHERE c."name" = 'Animation'
    )
)
ORDER BY f."title";
```

Pelis de la misma categoría que Animation.

- **Q59:** Encuentra los nombres de las películas que tienen la misma duración que la película con el título ‘Dancing Fever’. Ordena los resultados alfabéticamente por título de película.

```
SELECT f2."film_id", f2."title", f2."length"
FROM "film" AS f2
WHERE f2."length" =
      (SELECT f."length"
       FROM "film" AS f
       WHERE f."title" = 'DANCING FEVER'
       LIMIT 1)
AND f2."title" <> 'DANCING FEVER';
```

	123 film_id	AZ title	123 length
1	299	FACTORY DRAGON	144
2	508	LAMBS CINCINNATI	144
3	650	PACIFIC AMISTAD	144
4	695	PRESIDENT BANG	144
5	857	STRICTLY SCARFACE	144
6	899	TOWERS HURRICANE	144
7	946	VIRTUAL SPOILERS	144

Pelis con la **misma duración** que “DANCING FEVER” (excluyendo esa).

- **Q60:** Encuentra los nombres de los clientes que han alquilado al menos 7 películas distintas. Ordena los resultados alfabéticamente por apellido.

```
WITH "rentas" AS (
    SELECT r."customer_id", COUNT(DISTINCT i."film_id") AS "pelis_distintas"
    FROM "rental" AS r
    JOIN "inventory" AS i
        ON i."inventory_id" = r."inventory_id"
    GROUP BY r."customer_id"
)
SELECT c."customer_id", c."first_name", c."last_name", rentas."pelis_distintas"
FROM "rentas"
JOIN "customer" AS c USING ("customer_id")
WHERE rentas."pelis_distintas" >= 7
ORDER BY rentas."pelis_distintas" DESC;
```

	123 customer_id	AZ first_name	AZ last_name	.pelis_distintas
1	148	ELEANOR	HUNT	(148,46)
2	526	KARL	SEAL	(526,44)
3	236	MARCIA	DEAN	(236,42)
4	144	CLARA	SHAW	(144,42)
5	75	TAMMY	SANDERS	(75,41)
6	197	SUE	PETERS	(197,40)
7	5	ELIZABETH	BROWN	(5,38)
8	469	WESLEY	BULL	(469,38)
9	178	MARION	SNYDER	(178,38)
10	137	RHONDA	KENNEDY	(137,38)
11	295	DAISY	BATES	(295,38)
12	468	TIM	CARY	(468,38)

Clients con ≥ 7 pelis **distintas**; indicador de **variedad/fidelidad**.

- **Q61:** Encuentra la cantidad total de películas alquiladas por categoría y muestra el nombre de la categoría junto con el recuento de alquileres.

```
SELECT c."name" AS "category", COUNT(*) AS "rentals"
FROM "rental" AS r
JOIN "inventory" AS i
    ON i."inventory_id" = r."inventory_id"
JOIN "film_category" AS fc
    ON fc."film_id" = i."film_id"
JOIN "category" AS c
    ON c."category_id" = fc."category_id"
GROUP BY c."name"
ORDER BY "rentals" DESC;
```

The screenshot shows a MySQL Workbench interface with a query editor and a results grid. The query is:

```
category 1 × SQL Terminal
SELECT c.name AS "category", COUNT(*) AS "rentals" FR
```

The results grid displays 12 rows of data:

	A-Z category	123 rentals
1	Sports	1.179
2	Animation	1.166
3	Action	1.112
4	Sci-Fi	1.101
5	Family	1.096
6	Drama	1.060
7	Documentary	1.050
8	Foreign	1.033
9	Games	969
10	Children	945
11	Comedy	941
12	New	940

Recuento de alquileres por categoría; muestra demanda por género.

- **Q62:** Encuentra el número de películas por categoría estrenadas en 2006.

The screenshot shows a MySQL Workbench interface with a query editor and a results grid. The query is:

```
category 1 × SQL terminal
SELECT c.name AS "category", COUNT(*) AS "rentals_2006" FR
```

The results grid displays 12 rows of data:

	A-Z category	123 rentals_2006
1	Animation	21
2	Action	17
3	Sports	15
4	Games	14
5	Family	13
6	New	13
7	Horror	12
8	Foreign	11
9	Music	11
10	Travel	10
11	Comedy	9
12	Classics	9

Nº de pelis por categoría estrenadas en 2006 (mix de catálogo por año).

- **Q63:** Obtén todas las combinaciones posibles de trabajadores con las tiendas que tenemos.

```
SELECT s.staff_id, s.first_name, s.last_name, st.store_id
FROM staff AS s
CROSS JOIN store AS st
ORDER BY s.staff_id, st.store_id;
```

The screenshot shows a MySQL Workbench interface with a query editor and a results grid. The query is:

```
staff(+1) 1 × SQL Terminal
SELECT s.staff_id, s.first_name, s.last_name, st.store_id FR
```

The results grid displays 4 rows of data:

	123 staff_id	A-Z first_name	A-Z last_name	123 store_id
1	1	Mike	Hillyer	1
2	1	Mike	Hillyer	2
3	2	Jon	Stephens	1
4	2	Jon	Stephens	2

Todas las combinaciones staff×store.

- **Q64:** Encuentra la cantidad total de películas alquiladas por cada cliente y muestra el ID del cliente, su nombre y apellido junto con la cantidad de películas alquiladas.

```

SELECT c."customer_id", c."first_name", c."last_name", COUNT(r."rental_id") AS "total_alquiladas"
FROM "customer" AS c
LEFT JOIN "rental" AS r
    ON r."customer_id" = c."customer_id"
GROUP BY c."customer_id", c."first_name", c."last_name"
ORDER BY "total_alquiladas" DESC;

```

	customer_id	first_name	last_name	total_alquiladas
1	148	ELEANOR	HUNT	46
2	526	KARL	SEAL	45
3	144	CLARA	SHAW	42
4	236	MARCIA	DEAN	42
5	75	TAMMY	SANDERS	41
6	469	WESLEY	BULL	40
7	197	SUE	PETERS	40
8	137	RHONDA	KENNEDY	39
9	178	MARION	SNYDER	39
10	468	TIM	CARY	39
11	459	TOMMY	COLLAZO	38
12	5	ELIZABETH	BROWN	38

Alquileres totales por cliente con id y nombre completo (ranking).

5. Conclusiones

- **Catálogo equilibrado y metraje moderado.** La distribución por clasificación es homogénea con ligera mayoría de PG-13; las duraciones oscilan entre **46–185 min** y los títulos **>180 min** son minoría. La variabilidad del **coste de reemplazo** es **moderada** (desv. típica ≈ 6), lo que sugiere precios relativamente consistentes.
- **Géneros y oferta.** Algunas categorías promedian **>110 min**, lo que ayuda a identificar géneros de metraje largo. Existen muy **pocas comedias** por encima de 3 horas. El cruce con inventario permite ver títulos con **0 copias** (bueno para detectar huecos de stock).
- **Demanda y estacionalidad.** El volumen de alquileres muestra **picos en verano** (agregación mensual), y hay un subconjunto de títulos de **alta rotación** (≥ 10 alquileres) que deberían priorizarse en reposición.
- **Precio y facturación.** Las películas con **precio por encima de la media** coinciden principalmente con el tramo **4,99**. La **facturación total** agregada del periodo analizado es elevada y sirve como línea base para KPIs futuros.
- **Clientes.** Hay **clientes top** que concentran parte relevante del gasto; además, el grupo con ≥ 7 **películas distintas** refleja **fidelidad y variedad** en el consumo. También se identifican **alquileres largos** (>8 días) y **pendientes de devolución**, útiles para control operativo.
- **Actores y participación.** No aparecen actores sin películas en el dataset; unos pocos superan **40 participaciones**, formando el núcleo “super-productivo”. Se han segmentado intérpretes por género (p. ej., *Action*, *Sci-Fi*) para análisis temáticos.
- **Calidad y limitaciones de los datos.** El campo `original_language_id` está **sin poblar**, por lo que no aporta valor analítico en esta versión. Las métricas monetarias y formatos dependen de la configuración regional del cliente SQL.