

TALLER BASES DE DATOS
ING. WILLIAM ALEXANDER MATA LLANA PORRAS

TALLER MYSQL VS PostgreSQL

UNIVERSIDAD DE CUNDINAMARCA

FACULTAD DE INGENIERIA – NOCHE

LINEA DE PROFUNDIZACION III

GESTOR DE EL CONOCIMIENTO: Alexander Matallana

CHIA, CUNDINAMARCA 29 sep. 25

TALLER: Explorando las diferencias entre MySQL y PostgreSQL con Docker

Objetivo general

Reconocer las principales diferencias entre los motores de base de datos MySQL y PostgreSQL, ejecutarlos en contenedores Docker, y explorar sus comandos básicos en la terminal interactiva de PostgreSQL, complementando con el uso de un cliente gráfico (pgAdmin o DBeaver).

Objetivos específicos

- Comprender los conceptos generales de MySQL y PostgreSQL.
- Comparar su estructura, características, ventajas y comandos principales.
- Explorar los comandos básicos desde la terminal interactiva (psql).
- Conectarse a la base de datos con un cliente gráfico externo (pgAdmin o DBeaver).

Parte 1. Consulta guiada

Investiga los siguientes aspectos sobre MySQL y PostgreSQL, consúltalos en fuentes confiables (documentación oficial, blogs técnicos, etc.) y completa la siguiente tabla. Adjunta una imagen o captura de la fuente consultada o del entorno donde verificaste la información.

Aspecto a comparar	MySQL	PostgreSQL
Tipo de sistema (relacional / mixto)	MySQL es un sistema de administración de bases de datos relacionales que permite almacenar datos en forma de tablas con filas y columnas. Se trata de un sistema ampliamente conocido en el que se basan numerosas aplicaciones web, sitios web dinámicos y sistemas integrados. https://aws.amazon.com/es/compare/the-difference-between-mysql-vs-postgresql/?utm_source=chatgpt.com	PostgreSQL es un sistema de administración de bases de datos relacionales de objetos que ofrece más características. Aporta más flexibilidad en cuanto a tipos de datos, escalabilidad, simultaneidad e integridad de los datos. https://aws.amazon.com/es/compare/the-difference-between-mysql-vs-postgresql/?utm_source=chatgpt.com
Licencia	El software del servidor MySQL y las bibliotecas cliente utilizan una distribución con doble licencia. Se ofrecen bajo la GPL versión 2, o una licencia propietaria. También hay soporte técnico gratuito disponible en diferentes canales y foros de IRC https://en.wikipedia.org/wiki/MySQL?utm_source=chatgpt.com	PostgreSQL se publica bajo la <u>Licencia PostgreSQL</u> , una licencia de código abierto liberal, similar a las licencias BSD o MIT. Sistema de gestión de bases de datos PostgreSQL (también conocido como Postgres, anteriormente como Postgres95) https://www.postgresql.org/about/licence/#:~:text=PostgreSQL%20is%20released%20under%20the,the%20BSD%20or%20MIT%20licenses.&text=Permission%20to%20use%2C%20copy%2C%20modify,UPDATES%2C%20ENHANCEMENTS%2C%20OR%20MODIFICATIONS.
Enfoque principal	Rendimiento, simplicidad, uso web, facilidad de	Funcionalidad rica, integridad,

TALLER BASES DE DATOS
ING. WILLIAM ALEXANDER MATALLANA PORRAS

	configuración para aplicaciones con muchas consultas de lectura. https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/?utm_source=chatgpt.com/	características avanzadas (tipos de datos, extensibilidad), aplicaciones complejas. https://aws.amazon.com/es/comparar/the-difference-between-mysql-vs-postgresql/?utm_source=chatgpt.com
Tipos de datos admitidos	Tipos estándar disponibles	Más diverso, incluye matrices, hstore
Integridad referencial	Soporta claves foráneas, pero depende del motor de almacenamiento: InnoDB la soporta, otros motores pueden tener limitaciones	Muy fuerte soporte de integridad: claves foráneas, restricciones avanzadas, verificaciones (CHECK), vistas materializadas, triggers complejos https://kinsta.com/blog/postgresql-vs-mysql/?utm_source=chatgpt.com
Soporte de JSON y datos complejos	ha ofrecido la JSON_TABLE() función desde sus primeras versiones, lo que permite la conversión entre JSON y formatos relacionales. proporciona funciones integradas para validar JSON contra el esquema JSON. https://www.bytebase.com/blog/postgres-vs-mysql-json-support/?utm_source=chatgpt.com	Almacena datos en formato binario descompuesto. Si bien la entrada es ligeramente más lenta debido a la sobrecarga de conversión, su procesamiento es significativamente más rápido, ya que no requiere reanálisis. https://www.bytebase.com/blog/postgres-vs-mysql-json-support/?utm_source=chatgpt.com
Soporte para funciones y procedimientos	Procedimiento almacenado: Es un objeto que se crea con la sentencia CREATE PROCEDURE y se invoca con la sentencia CALL. Un procedimiento puede tener cero o muchos parámetros de entrada y cero o muchos parámetros de salida. Función almacenada: Es un objeto que se crea con la sentencia CREATE FUNCTION y se invoca con la sentencia SELECT o dentro de una expresión. Una función puede tener cero o muchos parámetros de entrada y siempre devuelve un valor, asociado al nombre de la función. https://josejuansanchez.org/bd/unidad-12-teoria/index.html#triggers-procedimientos-y-funciones-en-mysql	Crear una función o procedimiento es relativamente sencillo, dependiendo de la complejidad y del nivel de programación que tengamos. Actualmente se utiliza el comando CREATE FUNCTION para crear funciones y procedimientos. La diferencia está en el tipo de dato que tienen que devolver. Además podemos indicarle si necesita parámetros de entrada. https://www.todopostgresql.com/manejando-funciones-en-postgresql/
Nivel de cumplimiento del estándar SQL	MySQL puede operar en diferentes modos SQL y aplicarlos de forma distinta para cada cliente, según el valor de la <u>sql_mode</u> variable del sistema. Los administradores de bases de datos (DBA) pueden configurar el modo SQL global para que se ajuste a los requisitos operativos del servidor	PostgreSQL se adhiere estrechamente a los estándares SQL, lo que garantiza que sus consultas y comandos SQL sean consistentes con otros sistemas basados en SQL https://www.linkedin.com/p

	del sitio, y cada aplicación puede configurar su modo SQL de sesión según sus propios requisitos https://dev.mysql.com/doc/refman/8.4/en/compatibility.html	ulse/understanding-difference-between-sql-postgresql-which-naeem-shahzad- fc90e#:~:text=Cumpliment o%20de%20los%20est%C3%A1ndares%20SQL,principal%20a%20trav%C3%A9s%20de%20complementos.
Extensiones disponibles	MySQL Server admite algunas extensiones que probablemente no encontrará en otros SGBD SQL. Tenga en cuenta que, si las usa, es muy probable que su código no sea portable a otros servidores SQL. En algunos casos, puede escribir código que incluya extensiones MySQL, pero que siga siendo portable, utilizando comentarios como los siguientes: En este caso, MySQL Server analiza y ejecuta el código dentro del comentario como cualquier otra sentencia SQL, pero otros servidores SQL deberían ignorar las extensiones. Por ejemplo, MySQL Server reconoce la STRAIGHT_JOIN palabra clave en la siguiente sentencia, pero otros servidores no deberían: https://dev.mysql.com/doc/refman/8.4/en/extensions-to-ansi.html#:~:text=Los%20tipos%20de%20datos%20MEDIUMINT,%2C%20NULL%20%2C%20UNSIGNED%20y%20ZEROFILL%20	Gran ecosistema de extensiones: PostGIS (para GIS), extensiones de tiempo, búsqueda de texto, FDW (foreign data wrappers), etc https://www.liquibase.com/resources/guides/postgresql-vs-mysql?utm_source=chatgpt.com .
Uso recomendado	MySQL se usa principalmente para el desarrollo web, como motor de bases de datos para sitios y aplicaciones dinámicas, gestionando datos de clientes, productos y transacciones para plataformas de comercio electrónico y redes sociales. También es útil para centralizar registros, crear dashboards, y ejecutar consultas SQL complejas para informes y visualizaciones	PostgreSQL es ideal para aplicaciones transaccionales como OLTP, análisis de datos, y para almacenar datos complejos, incluyendo datos geoespaciales y no estructurados. También es una buena opción para aplicaciones web y móviles, y se beneficia de su uso en la nube por su escalabilidad. Sus puntos fuertes incluyen alto rendimiento, compatibilidad con estándares ACID para transacciones, y funciones para manejar datos complejos y concurrencia

Parte 2. Exploración práctica con Docker y PostgreSQL

Realiza la configuración de tu entorno Docker para levantar una instancia de PostgreSQL. Registra los pasos ejecutados y adjunta un pantallazo de la ejecución exitosa.

Espacio para evidencias:

Creación de la carpeta

1.
Verificamos la versión de Docker

TALLER BASES DE DATOS

ING. WILLIAM ALEXANDER MATALLANA PORRAS

docker pull postgres

```
C:\Users\yesen>docker --version
Docker version 28.4.0, build d8eb465
```

2. Descargamos la imagen de postgres

```
C:\Users\yesen>docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
ce1261c6d567: Pull complete
e1b18b5359f0: Pull complete
8551209c5a1e: Pull complete
8d0a13cb166d: Pull complete
8a9c24e23f88: Pull complete
5773151508cd: Pull complete
28b206cbbc14: Pull complete
ef07360e404d: Pull complete
168b3ade331e: Pull complete
30b4b10fcf1d: Pull complete
0a51ed68fa52: Pull complete
1d5017cf452d: Pull complete
180db792316f: Pull complete
6458b7f41c65: Pull complete
Digest: sha256:1d288494853e244e7a78d87b3526e650e5221c622f9768ecac9313d0874a9c39
Status: Downloaded newer image for postgres:latest
docker.io/library/postgres:latest

C:\Users\yesen>
```

3. Creamos el contenedor

```
C:\Users\yesen>docker run --name mi_postgres -e POSTGRES_PASSWORD=12345 -p 5432:5432 -d postgres
9d2664924b7525a4f82828871e1a55769482ec82f8a72be64b75156408c331b5

C:\Users\yesen>
```

Containers [Give feedback](#)

Container CPU usage 0.03% / 400% (4 CPUs available) Container memory usage 59.34MB / 3.64GB [Show charts](#)

Search Only show running containers

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	EjercicioBD	d3da93179c2a	mysql:latest	3315:3306	0%	5 days ago	Refresh Play Logs Delete
<input checked="" type="checkbox"/>	mi_postgres	9d2664924b75	postgres	5432:5432 ↗	0.04%	53 seconds ago	Refresh Play Logs Delete

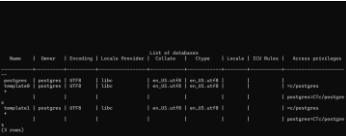
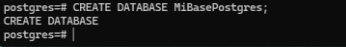
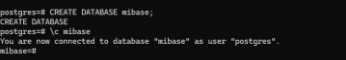
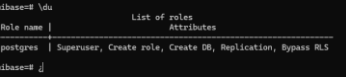
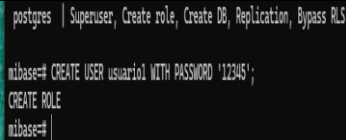
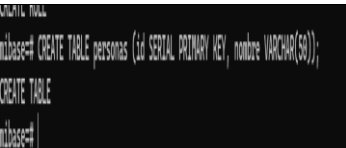

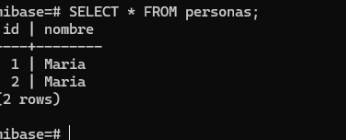
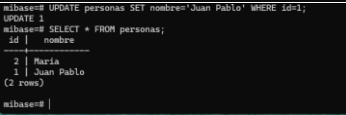
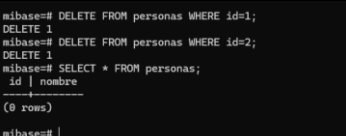
TALLER BASES DE DATOS
ING. WILLIAM ALEXANDER MATALLANA PORRAS

Ingresamos al contenedor

```
C:\Users\yesen>  
C:\Users\yesen>docker exec -it mi_postgres psql -U postgres  
psql (18.0 (Debian 18.0-1.pgdg13+3))  
Type "help" for help.  
  
postgres=# |
```

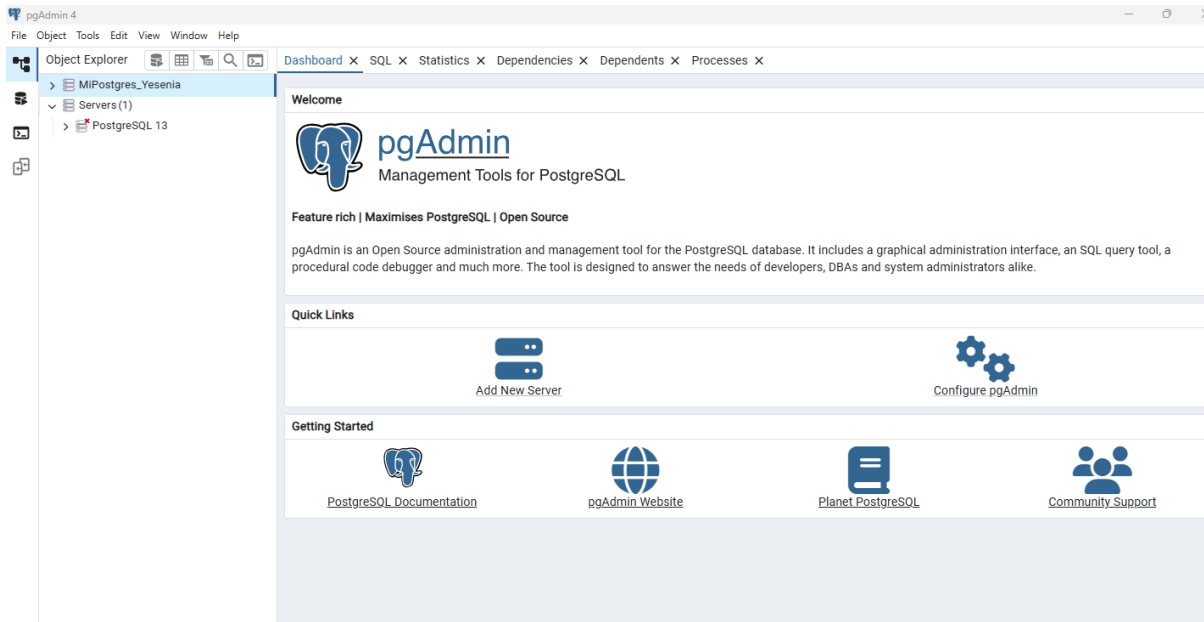
Parte 3. Exploración en la terminal interactiva (psql)

Accede al contenedor de PostgreSQL y explora la terminal interactiva. Ejecuta los comandoásicos para gestionar bases de datos, usuarios y tablas. Describe con tus palabras qué hace cada acción y adjunta la evidencia visual (pantallazo de la terminal).

Acción / Comando explorado	MSQL	PostgreSQL	Descripción o evidencia (pantallazo PostgreSQL)
Listar bases de datos	SELECT name FROM sys.databases;	l o \list	
Crear una base de datos	CREATE DATABASE MiBase;	CREATE DATABASE MiBasePostgres;	
Conectarse a una base específica	use Practica;	\c mibase	
Listar usuarios	SELECT name FROM sys.syslogins;	\du	
Crear un usuario nuevo	CREATE LOGIN usuario1 WITH PASSWORD='12345'; CREATE USER usuario1 FOR LOGIN usuario1;	CREATE USER usuario1 WITH PASSWORD '12345';	
Crear una tabla	CREATE TABLE Personas (id INT PRIMARY KEY, nombre NVARCHAR(50));	CREATE TABLE personas (id SERIAL PRIMARY KEY, nombre VARCHAR(50));	
Insertar datos en una tabla	INSERT INTO Personas (id, nombre) VALUES (1, 'Ana');	INSERT INTO personas (nombre) VALUES (Maria);	
Consultar registros	SELECT * FROM Personas;	SELECT * FROM personas;	
Actualizar registros	UPDATE Personas SET nombre='Maria' WHERE id=1;	UPDATE personas SET nombre='Mari a' WHERE id=1	
Eliminar registros	DELETE FROM Personas WHERE id=1;	DELETE FROM personas WHERE id=1;	

Parte 4. Conexión con cliente gráfico (pgAdmin o DBeaver)

Conéctate a la base de datos PostgreSQL utilizando pgAdmin o DBeaver. Realiza la conexión, crea una base de datos de prueba y explora sus tablas y registros. Anexa pantallazos de la conexión y describe brevemente el proceso.



Evidencia visual:

Parte 5. Actividades de aplicación

Realiza las siguientes tareas en tu entorno PostgreSQL y documenta los resultados con pantallazos o descripciones breves.

Modelo Relacional (versión reducida)

1. Autor

Campo	Tipo de Dato	Clave	Descripción
id_autor	INT	PK	Identificador del autor
nombre	VARCHAR(120)		Nombre completo del autor

TALLER BASES DE DATOS
ING. WILLIAM ALEXANDER MATA LLANA PORRAS

nacionalidad	VARCHAR(80)		Nacionalidad (opcional)

```
CREATE TABLE autor (id_autor SERIAL PRIMARY KEY, nombre VARCHAR(120) NOT NULL, nacionalidad VARCHAR(80));
```

```
mibase=# CREATE TABLE autor (id_autor SERIAL PRIMARY KEY, nombre VARCHAR(120) NOT NULL, nacionalidad VARCHAR(80));
CREATE TABLE
mibase=# CREATE TABLE libro (id_libro SERIAL PRIMARY KEY, titulo VARCHAR(200) NOT NULL, isbn VARCHAR(20) UNIQUE NOT NULL, anio_
publicacion INT, id_autor INT REFERENCES autor(id_autor), stock INT CHECK (stock >= 0));
CREATE TABLE
mibase=# SELECT * FROM autor;
 id_autor | nombre | nacionalidad
-----+-----+-----
(0 rows)
```

2. Libro

Campo	Tipo de Dato	Clave	Descripción
id_libro	INT	PK	Identificador del libro
titulo	VARCHAR(200)		Título del libro
isbn	VARCHAR(20)		Código ISBN (único)
anio_publicacion	INT		Año de publicación
id_autor	INT	FK	Autor del libro (1 autor por libro)
stock	INT		Unidades disponibles para préstamo (>=0)

```
CREATE TABLE libro (id_libro SERIAL PRIMARY KEY, titulo VARCHAR(200) NOT NULL, isbn VARCHAR(20) UNIQUE NOT NULL, anio_publicacion INT, id_autor INT NOT NULL, stock INT CHECK (stock >= 0), CONSTRAINT fk_libro_autor FOREIGN KEY (id_autor) REFERENCES autor(id_autor));
```

```
mibase=# CREATE TABLE libro (id_libro SERIAL PRIMARY KEY, titulo VARCHAR(200) NOT NULL, isbn VARCHAR(20) UNIQUE NOT NULL, anio_
publicacion INT, id_autor INT NOT NULL, stock INT CHECK (stock >= 0), CONSTRAINT fk_libro_autor FOREIGN KEY (id_autor) REFERENC
ES autor(id_autor));
CREATE TABLE
mibase=# SELECT * FROM libro;
 id_libro | titulo | isbn | anio_publicacion | id_autor | stock
-----+-----+-----+-----+-----+-----
(0 rows)

mibase=# |
```

3. Usuario

Campo	Tipo de Dato	Clave	Descripción
id_usuario	INT	PK	Identificador del usuario
documento	VARCHAR(30)		Documento de identidad (único)
nombre	VARCHAR(120)		Nombre completo
email	VARCHAR(120)		Correo electrónico (opcional)

TALLER BASES DE DATOS

ING. WILLIAM ALEXANDER MATALLANA PORRAS


```
CREATE TABLE usuario (id_usuario SERIAL PRIMARY KEY, documento VARCHAR(30) UNIQUE NOT NULL, nombre VARCHAR(120) NOT NULL, email VARCHAR(120));
```

```
(0 rows)

mibase=# CREATE TABLE usuario (id_usuario SERIAL PRIMARY KEY, documento VARCHAR(30) UNIQUE NOT NULL, nombre VARCHAR(120) NOT NULL, email VARCHAR(120));
CREATE TABLE
mibase=# SELECT * FROM usuario;
 id_usuario | documento | nombre | email
-----+-----+-----+-----
(0 rows)

mibase=# |
```

4. Prestamo

(Un registro por libro prestado a un usuario. Si un usuario lleva 2 libros, se crean 2 filas.)

Campo	Tipo de Dato	Clave	Descripción
id_prestamo	INT	PK	Identificador del préstamo
id_usuario	INT	FK	Usuario que realiza el préstamo
id_libro	INT	FK	Libro prestado
fecha_prestamo	DATE		Fecha del préstamo

fecha_devolucion_max	DATE		Fecha límite de devolución
fecha_devuelto	DATE		Fecha de devolución efectiva (nullable)
estado	VARCHAR(15)		ABIERTO / CERRADO

```
CREATE TABLE prestamo (id_prestamo SERIAL PRIMARY KEY, id_usuario INT NOT NULL, id_libro INT NOT NULL, fecha_prestamo DATE NOT NULL, fecha_devolucion_max DATE NOT NULL, fecha_devuelto DATE, estado VARCHAR(15) CHECK (estado IN ('ABIERTO','CERRADO')),CONSTRAINT fk_prestamo_usuario FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario), CONSTRAINT fk_prestamo_libro FOREIGN KEY (id_libro) REFERENCES libro(id_libro));
```

```
mibase=# CREATE TABLE prestamo (id_prestamo SERIAL PRIMARY KEY, id_usuario INT NOT NULL, id_libro INT NOT NULL, fecha_prestamo DATE NOT NULL, fecha_devolucion_max DATE NOT NULL, fecha_devuelto DATE, estado VARCHAR(15) CHECK (estado IN ('ABIERTO','CERRADO')),CONSTRAINT fk_prestamo_usuario FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario), CONSTRAINT fk_prestamo_libro FOREIGN KEY (id_libro) REFERENCES libro(id_libro));
CREATE TABLE
mibase=# SELECT * FROM prestamo;
 id_prestamo | id_usuario | id_libro | fecha_prestamo | fecha_devolucion_max | fecha_devuelto | estado
-----+-----+-----+-----+-----+-----+-----
(0 rows)

mibase=#
```

Relaciones

Autor — Libro

- Relación: Un autor tiene muchos libros.
- Cardinalidad: 1 a N.

Libro — Prestamo

- Relación: Un libro puede estar presente en muchos préstamos (una fila por unidad prestada).
- Cardinalidad: 1 a N.
- Regla de negocio sugerida: solo permitir préstamo si stock > 0 y al prestar disminuir stock en 1; al devolver, incrementar stock en 1.

Usuario — Prestamo

- Relación: Un usuario puede tener muchos préstamos.
- Cardinalidad: 1 a N.

Consultas a realizar

1. Libros con su autor

Columnas: titulo, nombre_autor, anio_publicacion.

```
SELECT l.titulo, a.nombre AS nombre_autor, l.anio_publicacion
FROM libro l
JOIN autor a ON l.id_autor = a.id_autor;
```

```
INSERT 0 2
mibase=# SELECT l.titulo, a.nombre AS nombre_autor, l.anio_publicacion
FROM libro l
JOIN autor a ON l.id_autor = a.id_autor;
          titulo          | nombre_autor | anio_publicacion
-----+-----+-----
 Cien años de soledad    | Gabriel García Márquez | 1967
 El amor en los tiempos del cólera | Gabriel García Márquez | 1985
 La casa de los espíritus | Isabel Allende | 1982
 Conversación en LaCatedral | Mario Vargas Llosa | 1969
(4 rows)
mibase=#
```

2. Préstamos abiertos con datos del usuario y del libro

Columnas: id_prestamo, usuario, titulo_libro, fecha_prestamo, fecha_devolucion_max, estado.}

```
mibase=# SELECT p.id_prestamo, u.nombre AS usuario, l.titulo AS titulo_libro,
p.fecha_prestamo, p.fecha_devolucion_max, p.estado
FROM prestamo p
JOIN usuario u ON p.id_usuario = u.id_usuario
JOIN libro l ON p.id_libro = l.id_libro
WHERE p.estado = 'ABIERTO';
 id_prestamo | usuario | titulo_libro | fecha_prestamo | fecha_devolucion_max | estado
-----+-----+-----+-----+-----+-----
          1 | Juan Pérez | Cien años de soledad | 2025-09-01 | 2025-09-15 | ABIERTO
(1 row)
mibase=#
```

3. Historial de préstamos de un usuario (por documento)

```
SELECT l.titulo AS titulo_libro, p.fecha_prestamo, p.fecha_devuelto, p.estado FROM prestamo
p JOIN usuario u ON p.id_usuario = u.id_usuario JOIN libro l ON p.id_libro = l.id_libro
WHERE u.documento = '12345';
```

Dado un documento, listar titulo_libro, fecha_prestamo, fecha_devuelto, estado.

```
mibase=# SELECT l.titulo AS titulo_libro, p.fecha_prestamo, p.fecha_devuelto, p.estado
FROM prestamo p
JOIN usuario u ON p.id_usuario = u.id_usuario
JOIN libro l ON p.id_libro = l.id_libro
WHERE u.documento = '12345';
      titulo_libro      | fecha_prestamo | fecha_devuelto | estado
-----+-----+-----+-----
 Cien años de soledad | 2025-09-01    |                | ABIERTO
(1 row)

mibase=# |
```

4. Top de autores por cantidad de libros registrados

Columnas: autor, cantidad_libros. Ordenar descendente.

```
SELECT a.nombre AS autor, COUNT(l.id_libro) AS cantidad_libros
FROM autor a
JOIN libro l ON a.id_autor = l.id_autor
GROUP BY a.nombre
ORDER BY cantidad_libros DESC;
```

```
mibase=# SELECT a.nombre AS autor, COUNT(l.id_libro) AS cantidad_libros
FROM autor a
JOIN libro l ON a.id_autor = l.id_autor
GROUP BY a.nombre
ORDER BY cantidad_libros DESC;
      autor      | cantidad_libros
-----+-----
 Gabriel García Márquez |          2
 Isabel Allende   |          1
 Mario Vargas Llosa |          1
(3 rows)

mibase=# |
```

5. Disponibilidad actual de cada libro

Columnas: titulo, stock. Filtrar libros con stock = 0 (sin unidades disponibles).

```
mibase=# SELECT titulo, stock
FROM libro
WHERE stock = 0;
      titulo      | stock
-----+-----
 Conversación en LaCatedral |      0
(1 row)

mibase=# |
```

6. **Libros prestados actualmente (no devueltos)**

Columnas: titulo, usuario, fecha_prestamo, fecha_devolucion_max.

(Pista: `prestamo.estado='ABIERTO'` o `fecha_devuelto IS NULL` según tu regla.)

```
SELECT l.titulo, u.nombre AS usuario, p.fecha_prestamo, p.fecha_devolucion_max
FROM prestamo p
JOIN libro l ON p.id_libro = l.id_libro
JOIN usuario u ON p.id_usuario = u.id_usuario
WHERE p.estado = 'ABIERTO' OR p.fecha_devuelto IS NULL;
```

```
mibase=# SELECT l.titulo, u.nombre AS usuario, p.fecha_prestamo, p.fecha_devolucion_max
FROM prestamo p
JOIN libro l ON p.id_libro = l.id_libro
JOIN usuario u ON p.id_usuario = u.id_usuario
WHERE p.estado = 'ABIERTO' OR p.fecha_devuelto IS NULL;
      titulo      | usuario | fecha_prestamo | fecha_devolucion_max
-----+-----+-----+-----
 Cien años de soledad | Juan Pérez | 2025-09-01 | 2025-09-15
 El amor en los tiempos del cólera | Ana Torres | 2025-08-20 | 2025-09-05
(2 rows)

mibase=# |
```

7. **Usuarios con cantidad de préstamos abiertos**

Columnas: usuario, prestamos_abiertos. (Agrupar por usuario.)

```
SELECT u.nombre AS usuario, COUNT(*) AS prestamos_abiertos
FROM prestamo p
JOIN usuario u ON p.id_usuario = u.id_usuario
WHERE p.estado = 'ABIERTO'
GROUP BY u.nombre;
```

```
mibase=# SELECT u.nombre AS usuario, COUNT(*) AS prestamos_abiertos
FROM prestamo p
JOIN usuario u ON p.id_usuario = u.id_usuario
WHERE p.estado = 'ABIERTO'
GROUP BY u.nombre;
  usuario  | prestamos_abiertos
-----+-----
 Juan Pérez | 1
(1 row)

mibase=# |
```

8. **Búsqueda por texto (título o autor)**

Dado un término, mostrar titulo, autor, anio_publicacion donde el término aparezca en título o nombre de autor (insensible a mayúsculas).

```
SELECT l.titulo, a.nombre AS autor, l.anio_publicacion
FROM libro l
JOIN autor a ON l.id_autor = a.id_autor
WHERE LOWER(l.titulo) LIKE LOWER('%amor%')
OR LOWER(a.nombre) LIKE LOWER('%amor%');
```

```
mibase=# SELECT l.titulo, a.nombre AS autor, l.anio_publicacion
FROM libro l
JOIN autor a ON l.id_autor = a.id_autor
WHERE LOWER(l.titulo) LIKE LOWER('%amor%')
      OR LOWER(a.nombre) LIKE LOWER('%amor%');
      titulo                | autor                | anio_publicacion
-----+-----+-----
El amor en los tiempos del cólera | Gabriel García Márquez | 1985
(1 row)

mibase=# |
```

9. Préstamos vencidos

Listar usuario, titulo, dias_atraso cuando CURRENT_DATE > fecha_devolucion_max y el préstamo siga abierto.

```
SELECT u.nombre AS usuario, l.titulo, CURRENT_DATE - p.fecha_devolucion_max AS
dias_atraso
FROM prestamo p
JOIN usuario u ON p.id_usuario = u.id_usuario
JOIN libro l ON p.id_libro = l.id_libro
WHERE CURRENT_DATE > p.fecha_devolucion_max
      AND p.estado = 'ABIERTO';
```

```
mibase=# SELECT u.nombre AS usuario, l.titulo, CURRENT_DATE - p.fecha_devolucion_max AS dias_atraso
FROM prestamo p
JOIN usuario u ON p.id_usuario = u.id_usuario
JOIN libro l ON p.id_libro = l.id_libro
WHERE CURRENT_DATE > p.fecha_devolucion_max
      AND p.estado = 'ABIERTO';
      usuario | titulo                | dias_atraso
-----+-----+-----
Juan Pérez | Cien años de soledad | 15
(1 row)

mibase=# |
```

10. Libros por década de publicación

Mostrar década (ej. 1990s, 2000s) y cantidad de libros. (Tip: agrupa por anio_publicacion/10.)

```
SELECT (anio_publicacion/10)*10 || 's' AS decada, COUNT(*) AS cantidad_libros
FROM libro
GROUP BY decada
ORDER BY decada;
```

TALLER BASES DE DATOS
ING. WILLIAM ALEXANDER MATALLANA PORRAS

```
mibase=# SELECT (anio_publicacion/10)*10 || 's' AS decada, COUNT(*) AS cantidad_libros
FROM libro
GROUP BY decada
ORDER BY decada;
 decada | cantidad_libros 
-----+-----
 1960s  |                2
 1980s  |                2
(2 rows)

mibase=#
```

Evidencias visuales: