Ejercicios para ejecutar comandos en MySQL

Luego Confeccionar el DER

EJERCICIO 1: Clínica Veterinaria "Dres Lucea"

Cliente: Dueño de una clínica veterinaria
Requiere: Registrar información de mascotas, sus dueños, visitas médicas y veterinarios.

Requisitos:

- Cada **mascota** tiene nombre, especie, raza, edad, y pertenece a un **dueño**.
- Cada **dueño** tiene nombre, apellido, teléfono y email (único).
- Cada **visita médica** incluye fecha, motivo, observaciones y está asociada a una **mascota** y a un **veterinario**.
- Cada **veterinario** tiene nombre, apellido, matrícula profesional y horario de atención.
- **Facturación:** Se debe registrar el costo de cada visita médica, el método de pago (efectivo, tarjeta, transferencia) y el estado del pago (pendiente, pagado).

EJERCICIO 2: Academia de Idiomas "Conocer el mundo"

Cliente: Coordinador de una academia de idiomas
Requiere: Llevar registro de alumnos, profesores, cursos y pagos incluyendo la facturación.

Requisitos:

- Cada **alumno** se registra con nombre, apellido, email y teléfono.
- Cada **curso** tiene idioma, nivel (básico, intermedio, avanzado), días y horario.
- Cada **profesor** enseña uno o más cursos y tiene CUIL, nombre y especialidad.
- Deben registrarse los **pagos de los alumnos**, con fecha, importe y forma de pago (efectivo, tarjeta, transferencia) y estado del pago (pendiente, pagado).
- **Facturación:** Generar un resumen de facturación mensual por alumno y por curso.

EJERCICIO 3: Biblioteca Popular "Argentina"

Cliente: Encargado de una biblioteca
Requiere: Gestionar socios, libros, préstamos, devoluciones y facturación.

Requisitos:

- Cada **socio** tiene nombre, apellido, dirección, email y teléfono.
- Cada **libro** tiene título, autor, editorial, año, y un código único.
- Un **préstamo** incluye fecha de préstamo, fecha de devolución (prevista y real) y está asociado a un **socio** y un **libro**.
- Se deben registrar múltiples préstamos por socio.
- **Facturación:** En caso de retraso en la devolución de un libro, se debe registrar una multa por día de demora, calculando el monto total a pagar y el estado del pago (pendiente, pagado).

EJERCICIO 4: Estudio de Danza "Contem&Drak"

Cliente: Propietaria de un estudio de danza
Requiere: Control de alumnos, clases, profesores y asistencias. Además, incorporar el módulo Facturación.

Requisitos:

- Cada **alumno** tiene nombre, edad, dirección y forma de pago.
- Cada **profesor** da clases en uno o más estilos (ballet, salsa, tango, etc.) y tiene un CUIL y teléfono.
- Cada **clase** tiene fecha, hora, estilo de danza, profesor y aula.
- Se debe llevar registro de **asistencia de los alumnos a cada clase**.
- **Facturación:** Registrar el costo por clase y los pagos realizados por los alumnos. Generar un reporte mensual por alumno y profesor.

EJERCICIO 5: Taller Mecánico "Spinola"

Cliente: Dueño de un taller mecánico
Requiere: Control de vehículos, clientes, servicios realizados, mecánicos, facturación.

Requisitos:

- Cada **cliente** tiene nombre, DNI, teléfono y dirección.
- Cada **vehículo** tiene marca, modelo, patente (única) y pertenece a un cliente.
- Cada **servicio** (reparación o mantenimiento) tiene fecha, descripción, costo y lo realiza un **mecánico**.

- Cada **mecánico** tiene nombre, apellido, especialidad y horario disponible.
- **Facturación:** Registrar el costo del servicio, el método de pago y el estado del pago (pendiente, pagado). Generar un resumen de facturación mensual por cliente y por mecánico.

Detalles a tener en cuenta para la inserción de datos y verificaciones previas

Paso 1: Verificación de existencia de datos clave

Antes de realizar cualquier inserción en las tablas que contienen **claves foráneas**, es fundamental que verifiques que **los registros en las tablas relacionadas existan**. De lo contrario, **las claves foráneas no se podrán insertar debido a que las restricciones no se cumplirían.**

Ejemplo:

Verificación de existencia de registros clave:

Para la tabla VisitaMedica, necesitamos verificar que existan registros en las tablas Mascota y Veterinario.

Código:

```
-- Verificar existencia de idMascota en la tabla Mascota

SELECT * FROM Mascota WHERE idMascota = 1;



-- Verificar existencia de idVeterinario en la tabla Veterinario

SELECT * FROM Veterinario WHERE idVeterinario = 1;
```

Para la tabla Facturacion, necesitamos verificar que el idVisita existe en la tabla VisitaMedica.

```
-- Verificar existencia de idVisita en la tabla VisitaMedica

SELECT * FROM VisitaMedica WHERE idVisita = 2;
```

Paso 2: Estructura de la carga de datos

En el contexto de bases de datos relacionales, una tupla es una fila de datos en una tabla, es decir, un conjunto de valores que forman un registro completo.

Se utilizan paréntesis, para cada grupo de datos que se carga en una tabla, y la separación entre ellos es con una coma (,):

Los paréntesis agrupan los valores como un conjunto que se insertará en una única fila de la tabla.

Las comas separan los valores, indicando que cada uno de ellos corresponde a una columna específica en el orden definido.

Paréntesis y coma ayudan al motor de SQL a interpretar cada conjunto de valores como una fila completa que debe insertarse en una tabla.

A su vez, cuando vamos a realizar más de una inserción, la separación de los paréntesis también es con una coma (,), hasta llegar a la última que lleva como siempre **;. Entonces, por ejemplo**

```
INSERT INTO Dueño (idDueño, Nombre, Apellido, Telefono, Email)

VALUES

(1, 'Marcos', 'Sánchez', '1156789900', 'marcos.sanchez@example.com'),

(2, 'Lucía', 'García', '1122334455', 'lucia.garcia@example.com');
```

Esto le dice al **motor SQL** que debe insertar las **filas completas** en una única instrucción.

La sintaxis con paréntesis y comas es estándar en SQL porque permite que el motor interprete los datos de forma estructurada y consistente. Cambiar el formato podría generar ambigüedad o errores en la interpretación de los datos.

Paso 3: Orden de inserción de datos

El orden de inserción debe seguir un **flujo lógico que respete las relaciones entre las tablas y sus claves foráneas.**

Insertar dueños (No tiene claves foráneas)

```
INSERT INTO Dueño VALUES (1, 'Marcos', 'Sánchez', '1156789900', 'marcos.sanchez@example.com');

INSERT INTO Dueño VALUES (2, 'Lucía', 'García', '1122334455', 'lucia.garcia@example.com');
```

Insertar mascotas (Referencia la tabla Dueño)

```
INSERT INTO Mascota VALUES (1, 'Firulais', 'Perro', 'Labrador', 5, 1);

INSERT INTO Mascota VALUES (2, 'Mimi', 'Gato', 'Siames', 3, 2);
```

Insertar veterinarios (No tiene claves foráneas)

```
INSERT INTO Veterinario VALUES (1, 'Dra. Ana', 'Pérez', 'MP12345', 'Lunes a Viernes 9-12');

INSERT INTO Veterinario VALUES (2, 'Dr. Juan', 'Gómez', 'MP67890', 'Lunes a Viernes 14-18');
```

Insertar visitas médicas (Referencia las tablas Mascota y Veterinario)

```
INSERT INTO VisitaMedica VALUES (1, 1, 1, '2025-05-04', 'Consulta general', 'Revisión anual');

INSERT INTO VisitaMedica VALUES (2, 2, 2, '2025-05-05', 'Vacunación', 'Primera dosis aplicada');
```

Insertar facturación (Referencia la tabla VisitaMedica)

```
INSERT INTO Facturacion VALUES (1, 1, 1500.00, 'Tarjeta', 'Pagado');

INSERT INTO Facturacion VALUES (2, 2, 800.00, 'Efectivo', 'Pendiente');
```

Paso 4: Verificación de integridad referencial

Una vez insertados los datos, es importante verificar que la integridad referencial se haya mantenido. Esto significa **que cada clave foránea** en las tablas secundarias **debe corresponder a una clave primaria en la tabla principal**.

Verificar integridad referencial de la tabla VisitaMedica:

```
-- Verificar que todas las visitas médicas tengan un idMascota válido
SELECT * FROM VisitaMedica vm
LEFT JOIN Mascota m ON vm.idMascota = m.idMascota
WHERE m.idMascota IS NULL;


-- Verificar que todas las visitas médicas tengan un idVeterinario válido
```

```
SELECT * FROM VisitaMedica vm
LEFT JOIN Veterinario v ON vm.idVeterinario = v.idVeterinario
WHERE v.idVeterinario IS NULL;
```

Verificar integridad referencial de la tabla Facturacion:

```
-- Verificar que todas las facturas tengan un idVisita válido
SELECT * FROM Facturacion f
LEFT JOIN VisitaMedica vm ON f.idVisita = vm.idVisita
WHERE vm.idVisita IS NULL;
```

Paso 5: Sugerencias adicionales para evitar errores

Desactivar temporalmente las restricciones de clave foránea (en entornos de prueba):

Cuando estamos trabajando en un entorno de desarrollo o prueba y necesitamos realizar inserciones sin que las claves foráneas generen errores, podemos **desactivar las restricciones de clave foránea temporalmente. No se recomienda esto en producción, ya que puede generar inconsistencias.**

```
-- Desactivar comprobación de claves foráneas
SET FOREIGN_KEY_CHECKS = 0;


-- Realizar inserciones


-- Activar comprobación de claves foráneas
SET FOREIGN_KEY_CHECKS = 1;
```

Validar datos antes de insertar:

Realizamos validaciones de los datos antes de hacer las inserciones. Esto incluye **asegurarte de que no se repitan registros con claves únicas** (como el email en la tabla Dueño) y **que las fechas sean correctas**.

Paso 6: Conclusión

Siguiendo este flujo de trabajo y asegurándonos que los datos estén validados antes de la inserción, podemos evitar muchos errores relacionados con claves foráneas y otros problemas comunes**. La clave es verificar siempre las relaciones entre las tablas antes de insertar datos en las tablas dependientes**.