✅ Parte Teórica

1. ¿Qué mecanismo podrías utilizar para registrar automáticamente cada vez que se realice un nuevo envío?

✅ **c) Trigger AFTER INSERT sobre la tabla Envios**
🔹 *Se ejecuta automáticamente después de insertar un nuevo registro.*

2. ¿Qué opción es más adecuada para calcular el total de envíos por zona?

✅ **d) Un procedimiento almacenado, ya que modifica registros**
🔹 *Aunque el cálculo en sí no modifica registros, un procedimiento puede realizar lógica compleja, aceptar parámetros, y devolver resultados personalizados. Entonces, si se desea más control, un procedimiento es una mejor opción que una vista.*

🔍 Explicación detallada:

❌ ¿Por qué NO una función (UDF)?

- Las **funciones** en MySQL están diseñadas para **devolver un valor escalar o único resultado**.
- **No permiten ejecutar instrucciones SQL complejas como INSERT, UPDATE o DELETE**, ni mostrar un conjunto de resultados con múltiples filas (como una tabla).
- Están pensadas para ser usadas **dentro de una consulta**, por ejemplo en un `SELECT`, como `SELECT meses_en_empresa(...)`.

✅ ¿Por qué sí un procedimiento almacenado?

- **Los procedimientos almacenados** pueden:
    - Ejecutar múltiples sentencias SQL.
    - **Devolver conjuntos de resultados (SELECT)** como una tabla.
    - Aceptar **parámetros de entrada o salida** (por ejemplo, una zona específica).
    - Incluir **lógica condicional y de control de flujo** (IF, CASE, loops).
- Esto lo hace **ideal** para realizar consultas como:

- Agrupar envíos por zona.
- Aplicar filtros condicionales.
- Generar reportes personalizados.

📝 **Ejemplo realista:** Si querés obtener los totales por zona **y además** enviar el resultado a otra tabla, o tomar decisiones adicionales con esos datos, **solo un procedimiento puede hacerlo.**

3. ¿Qué ventajas aporta una CTE al realizar consultas complejas sobre envíos agrupados por estado?

✅ **b) Permite escribir consultas recursivas y estructuradas**

4. ¿Qué efecto tendría un trigger BEFORE INSERT sobre la tabla Envios?

✅ **b) Validar o modificar datos antes de que se inserten**

5. ¿Qué función cumple la cláusula WITH CHECK OPTION en una vista de paquetes en tránsito?

✅ **b) Asegura que los datos insertados cumplan con la condición de la vista**

6. ¿Qué tipo de índice convendría utilizar sobre la columna paquete_id en la tabla Envios para búsquedas eficientes y evitar duplicaciones?

✅ **c) Índice único (UNIQUE)**

🛠️ Parte Práctica

Supongamos estas estructuras básicas:

```
CREATE DATABASE EmpresaServicios;
USE EmpresaServicios;

CREATE TABLE Clientes (
    cliente_id INT PRIMARY KEY,
    nombre VARCHAR(50),
```

```
    apellido VARCHAR(50)
);

CREATE TABLE Empleados (
    empleado_id INT PRIMARY KEY,
    nombre VARCHAR(50),
    cargo VARCHAR(50),
    fecha_ingreso DATE
);

CREATE TABLE Proyectos (
    proyecto_id INT PRIMARY KEY,
    nombre VARCHAR(100),
    cliente_id INT,
    facturacion DECIMAL(10,2),
    FOREIGN KEY (cliente_id) REFERENCES Clientes(cliente_id)
);

CREATE TABLE Tareas (
    tarea_id INT PRIMARY KEY,
    proyecto_id INT,
    descripcion TEXT,
    FOREIGN KEY (proyecto_id) REFERENCES Proyectos(proyecto_id)
);

CREATE TABLE auditoria_proyectos (
    id_auditoria INT AUTO_INCREMENT PRIMARY KEY,
    proyecto_id INT,
    fecha_modificacion DATETIME
);
```

---

7. Procedimiento para total facturado + vista ClientesFrecuentes

sql
CopiarEditar

```
-- CTE dentro de vista
WITH TotalFacturado AS (
    SELECT
        c.cliente_id,
        c.nombre,
        c.apellido,
```

```
        SUM(p.facturacion) AS total_facturado
    FROM Clientes c
    JOIN Proyectos p ON c.cliente_id = p.cliente_id
    GROUP BY c.cliente_id
)
CREATE VIEW ClientesFrecuentes AS
SELECT *
FROM TotalFacturado
WHERE total_facturado > 10000;
```

Explicación

- El JOIN nos permite **combinar información de ambas tablas**:
    - Cliente (nombre y apellido)
    - Total facturado (suma de todos sus proyectos)

⚠️ Si no usáramos JOIN, **no podríamos ver el nombre del cliente** junto con el total facturado.

---

8. Trigger AFTER UPDATE en Proyectos

```
DELIMITER $$

CREATE TRIGGER trg_auditoria_proyectos
AFTER UPDATE ON Proyectos
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_proyectos (proyecto_id,
fecha_modificacion)
    VALUES (NEW.proyecto_id, NOW());
END $$

DELIMITER ;
```

---

9. Crear índice sobre proyecto_id en Tareas

sql
CopiarEditar

```
CREATE INDEX idx_proyecto_id ON Tareas(proyecto_id);
```

---

10. Función + vista de antigüedad en meses

Función

sql
CopiarEditar

```
DELIMITER $$

CREATE FUNCTION meses_en_empresa(fecha_ingreso DATE)
RETURNS INT
DETERMINISTIC
BEGIN
    RETURN PERIOD_DIFF(DATE_FORMAT(CURDATE(), '%Y%m'),
DATE_FORMAT(fecha_ingreso, '%Y%m'));
END $$

DELIMITER ;
```

Vista

```
CREATE VIEW EmpleadosConAntiguedad AS
SELECT
    nombre,
    cargo,
    meses_en_empresa(fecha_ingreso) AS antiguedad_meses
FROM Empleados
WHERE meses_en_empresa(fecha_ingreso) > 24;
```