Proyecto 1 SQL – Gestión de Base de Datos Empresarial

La empresa TechNova S.A. desea crear una base de datos para gestionar la información de sus empleados y departamentos.
Realiza las siguientes operaciones en SQL utilizando los distintos tipos de lenguajes (DDL, DML, DQL, DCL y TCL).
Al finalizar cada bloque, verifica los resultados con consultas adecuadas.

1) DDL – Definición de datos

- Crea una base de datos llamada empresa_technova.
- Crea una tabla empleados con los campos:
  - id (INT, PRIMARY KEY)
  - nombre (VARCHAR(50))
  - edad (INT)
  - salario (DECIMAL(10,2))
- Añade una nueva columna direccion (VARCHAR(100)) a empleados.
- Crea una tabla departamentos con los campos:
  - id_dep (INT, PRIMARY KEY)
  - nombre (VARCHAR(50))
  - id_empleado (INT, FOREIGN KEY que referencia a empleados(id)).

2) DML – Manipulación de datos

- Inserta **tres empleados** y **dos departamentos**.
- Modifica el salario de un empleado.
- Elimina al empleado cuyo salario sea inferior a 1500.

3) DQL – Consulta de datos

- Muestra todos los empleados.
- Muestra el **nombre y salario** de los empleados **mayores de 30 años**, ordenados por salario descendente.
- Muestra el **número de empleados agrupados por edad**.
- Muestra la **edad y el salario medio** de los empleados con salario medio > 1800 (usa GROUP BY y HAVING).

- Realiza una JOIN para mostrar el **nombre del empleado y el nombre de su departamento**.

4) DCL – Control de acceso (básico)

- Crea un usuario usuario1.
- Concede a usuario1 **SELECT e INSERT** sobre la tabla empleados.
- Revoca el permiso de **INSERT** a usuario1 (manteniendo SELECT).

5) TCL – Control de transacciones

- Inicia una transacción.
- Inserta un nuevo empleado.
- Crea un **SAVEPOINT**.
- Actualiza el salario del nuevo empleado y **revierte los cambios hasta el SAVEPOINT**.
- Ejecuta **COMMIT** para confirmar los cambios finales.

6) Vistas (Views)

- Crea una vista llamada vista_empleados_activos que muestre: id, nombre, edad, salario y **nombre del departamento** (usa JOIN entre empleados y departamentos).
  La vista debe **excluir empleados con salario < 1500**.
- Crea otra vista vista_resumen_salarios que muestre por edad: edad, total_empleados y salario_medio (usa COUNT(*) y AVG(salario)).

Realiza consultas sobre ambas vistas:

- Todos los empleados activos ordenados por salario descendente (desde vista_empleados_activos).
- Edades con salario medio > 2000 desde vista_resumen_salarios.

7) Roles y permisos

Crea los siguientes roles:

- rol_consulta con permisos de lectura.
- rol_editor_empleados con permisos de SELECT e INSERT/UPDATE en empleados.

Concede permisos a los roles:

- A rol_consulta:
  - SELECT sobre todas las vistas creadas.

    - SELECT sobre las tablas empleados y departamentos.
- A rol_editor_empleados:
    - SELECT, INSERT, UPDATE sobre empleados.
    - (**No conceder DELETE**).

Asigna roles a usuarios:

- Asigna rol_consulta a usuario1.
- Crea usuario2 y asígnale rol_editor_empleados.

Prueba de permisos:

- Con usuario1, verifica que **puede consultar** vista_empleados_activos pero **no puede insertar** en empleados.
- Con usuario2, verifica que **puede insertar/actualizar** en empleados y **consultar las vistas**.

Revocaciones y limpieza:

- Revoca rol_editor_empleados de usuario2.
- Revoca rol_consulta de usuario1.
- Revoca de los roles cualquier permiso que concediste sobre las vistas y tablas.
- Elimina los roles con DROP ROLE.

8) Triggers

Crea mecanismos automáticos para auditar cambios y controlar reglas de negocio.

1. **Tabla de auditoría de salarios**
    - Crea una tabla empleados_salario_log con los campos:
        - id_log (INT, PRIMARY KEY, autoincremental si tu SGBD lo permite)
        - id_empleado (INT)
        - salario_anterior (DECIMAL(10,2))
        - salario_nuevo (DECIMAL(10,2))
        - fecha_cambio (DATETIME o TIMESTAMP)
        - usuario_bd (VARCHAR(50)) – para guardar el usuario de BD que hizo el cambio (si tu SGBD lo permite con funciones del tipo CURRENT_USER).

2. **Trigger de auditoría en UPDATE de salario**
    - Crea un **TRIGGER AFTER UPDATE** sobre la tabla empleados que:
        - Se dispare **solo cuando el salario cambie**.
        - Inserte un registro en empleados_salario_log con:
            - el id del empleado,
            - el salario anterior,
            - el salario nuevo,
            - la fecha/hora del cambio,
            - el usuario de BD que realizó la operación.
    - Realiza un UPDATE de salario sobre algún empleado y verifica que el trigger ha insertado el registro en la tabla de log.
3. **Trigger de control de salario mínimo en INSERT**
    - Crea un **TRIGGER BEFORE INSERT** sobre empleados que:
        - Verifique que el salario del nuevo empleado sea **como mínimo 1000**.
        - Si el salario es menor de 1000, realiza una de estas dos acciones (elige una, según tu SGBD):
            - Ajusta el salario automáticamente a 1000, o
            - Lanza un error que impida la inserción.
    - Prueba el trigger intentando insertar un empleado con salario 800 y verifica el comportamiento.
4. **Trigger en DELETE**
    - Crea un **TRIGGER BEFORE DELETE** en empleados que:
        - Inserte en una tabla empleados_borrados (que debes crear) los datos básicos del empleado eliminado (id, nombre, edad, salario, fecha_borrado).
    - Elimina un empleado y comprueba que aparece en empleados_borrados.

9) Índices

Mejora el rendimiento de las consultas creando índices adecuados.

1. **Índice por edad en empleados**
    - Crea un índice llamado idx_empleados_edad sobre la columna edad de la tabla empleados.

- Realiza una consulta que agrupe por edad y observa (si tu SGBD lo permite) el plan de ejecución antes y después de crear el índice.

2. **Índice en salario para consultas de rango**
   - Crea un índice idx_empleados_salario sobre la columna salario.
   - Ejecuta varias consultas que filtren por rangos de salario (por ejemplo, WHERE salario BETWEEN 1500 AND 2500) y observa la mejora.
3. **Índice único en nombre de departamento**
   - Crea un **índice único** idx_departamentos_nombre_unique sobre la columna nombre de la tabla departamentos para evitar nombres de departamento duplicados.
   - Intenta insertar un departamento con un nombre ya existente y comprueba que el SGBD no lo permite.
4. **Índice compuesto**
   - Crea un índice compuesto idx_empleados_edad_salario sobre (edad, salario).
   - Realiza una consulta que filtre por edad y ordene por salario y comprueba el uso del índice (si tu SGBD permite ver el plan de ejecución).

10) Copias de seguridad y restauración (Backup & Recovery)

Define y documenta cómo harías las copias de seguridad de la base de datos empresa_technova.

Restauración de la base de datos desde un backup

- Escribe el comando que utilizarías para **restaurar** la base de datos empresa_technova a partir del fichero de backup generado.
- Explica brevemente en qué situaciones usarías esta restauración

Rúbrica de Evaluación – Proyecto SQL TechNova (Sobre 10 puntos)

Criterio	Descripción	Puntuación
Definición de datos (DDL)	Creación correcta de la base de datos, tablas, claves primarias y foráneas, y columna adicional.	**1.5 / 10**
Manipulación de datos (DML)	Inserción, actualización y eliminación de registros sin errores.	**0.7 / 10**
Consultas (DQL)	Consultas bien construidas: filtros, ordenaciones, agrupamientos, condiciones y uso de JOIN.	**1.5 / 10**
Control de acceso (DCL)	Creación de usuarios, concesión y revocación de permisos correctamente aplicadas.	**0.7 / 10**
Control de transacciones (TCL)	Uso de START TRANSACTION, SAVEPOINT, ROLLBACK y COMMIT con coherencia.	**0.7 / 10**
Vistas (Views)	Creación, uso y consultas sobre vistas válidas.	**0.7 / 10**
Roles y permisos avanzados	Creación y asignación de roles, privilegios y revocaciones finales.	**0.7 / 10**
Triggers	Triggers funcionales y verificados (auditoría, validación, etc.).	**0.7 / 10**
Índices	Creación de índices adecuados y justificados; comprobación de su uso.	**0.7 / 10**
Backup y restauración	Comandos correctos de backup, backup de esquema y restauración.	**0.7 / 10**
Claridad y documentación del script	Código limpio, organizado y comentado por secciones.	**0.7 / 10**

Total: 10 puntos