1. Consultas Anidadas (Subqueries)

Teoría: Las consultas anidadas o subconsultas son consultas SQL dentro de otra consulta SQL principal. Permiten realizar operaciones complejas dividiéndolas en pasos lógicos, como encontrar datos en función de condiciones que dependen de otras tablas o cálculos previos.

Uso Práctico: Son útiles para seleccionar datos en base a un conjunto específico de resultados de otra consulta. Las subconsultas pueden aparecer en cláusulas como SELECT, WHERE, y FROM.

Ejemplo:
Supongamos que queremos obtener los nombres de las materias en las que los alumnos tienen la mínima cantidad de horas por semana.

sql
```
SELECT nombreMateria
FROM materias
WHERE horasPorSemana = (
  SELECT MIN(horasPorSemana)
  FROM materias
);
```

En este caso:
- La subconsulta (SELECT MIN(horasPorSemana) FROM materias) obtiene el valor mínimo de horasPorSemana.
- La consulta externa selecciona las materias que coinciden con ese valor mínimo.

---

2. Stored Procedures (Procedimientos Almacenados)

Teoría: Un Stored Procedure es un conjunto de instrucciones SQL almacenadas en el servidor de la base de datos, que se pueden ejecutar como una sola unidad. Estos procedimientos permiten ejecutar operaciones complejas en una base de datos, encapsulando lógica de negocios y mejorando la eficiencia.

Uso Práctico: Se usan para realizar operaciones repetitivas, como cálculos o actualizaciones en masa, y para facilitar el mantenimiento del código SQL en aplicaciones.

Ejemplo:
Creemos un procedimiento almacenado que muestre el total de alumnos inscritos en una materia específica, dado su id_materia.

sql
```
CREATE PROCEDURE contarAlumnosMateria (IN idMateria INT)
BEGIN
```

```
    SELECT COUNT(id_alumno) AS totalAlumnos
    FROM asignaciones
    WHERE id_materia = idMateria;
END;
```

Para ejecutar el procedimiento y obtener el número de alumnos inscritos en una materia con id_materia = 1:

sql
```
CALL contarAlumnosMateria(1);
```

---

3. Checks (Restricciones de Integridad)

Teoría: La restricción CHECK se utiliza para asegurar que los valores de una columna cumplan con una condición específica. Ayuda a mantener la integridad de los datos en la base de datos, asegurando que los datos ingresados cumplan ciertos criterios.

Uso Práctico: Se aplican para validar datos al insertarlos o actualizarlos. Por ejemplo, asegurarse de que el valor de horas semanales de una materia esté en un rango lógico.

Ejemplo:
Supongamos que queremos asegurar que las horas semanales (horasPorSemana) de una materia estén siempre entre 1 y 10.

sql
```
CREATE TABLE materias (
    id_materia INT PRIMARY KEY,
    nombreMateria VARCHAR(50),
    horasPorSemana INT,
    CHECK (horasPorSemana BETWEEN 1 AND 10)
);
```

En este caso:
- La restricción CHECK asegura que los valores de horasPorSemana siempre estarán dentro del rango especificado. Si se intenta insertar o actualizar un valor fuera de este rango, la operación fallará.

---

4. Triggers (Disparadores)

Teoría: Los triggers son rutinas que se ejecutan automáticamente en respuesta a eventos específicos en una tabla, como INSERT, UPDATE o DELETE. Permiten mantener la integridad de datos y automatizar ciertas tareas, como actualizaciones y auditorías.

Uso Práctico: Son útiles para realizar acciones automáticas, como registrar cambios en los datos o calcular valores derivados sin intervención manual.

Ejemplo:
Creamos un trigger que registre cada vez que un alumno se inscribe en una materia, almacenando el ID del alumno y la fecha en una tabla de registro_inscripciones.

sql
```
CREATE TRIGGER registrar_inscripcion
AFTER INSERT ON asignaciones
FOR EACH ROW
BEGIN
   INSERT INTO registro_inscripciones (id_alumno, fecha_inscripcion)
   VALUES (NEW.id_alumno, NEW.fechaInscripcion);
END;
```

En este caso:
- El trigger registrar_inscripcion se activa después de una inserción en la tabla asignaciones.
- Inserta en registro_inscripciones los datos del alumno y la fecha de inscripción.

---

5. Views (Vistas)

Teoría: Las vistas son consultas almacenadas que actúan como una tabla virtual. Permiten simplificar el acceso a datos complejos y abstraer el modelo de datos subyacente, proporcionando una capa de seguridad al limitar el acceso directo a los datos.

Uso Práctico: Facilitan la recuperación de datos frecuentes y la organización de datos para que sea más accesible y seguro, especialmente cuando se necesita exponer solo ciertos datos de una tabla.

Ejemplo:
Creamos una vista que muestra la información completa de los alumnos y sus materias.

sql
```
CREATE VIEW vista_alumnos_materias AS
SELECT alumnos.nombre, alumnos.apellido, materias.nombreMateria,
asignaciones.fechaInscripcion
FROM alumnos
JOIN asignaciones ON alumnos.id_alumno = asignaciones.id_alumno
```

JOIN materias ON asignaciones.id_materia = materias.id_materia;

En este caso:
- La vista vista_alumnos_materias contiene solo la información relevante para el seguimiento de alumnos y materias, omitiendo otros datos que puedan estar en las tablas originales.
- Ahora podemos consultar esta vista como una tabla normal para obtener información estructurada de alumnos y sus materias sin acceder directamente a cada tabla.

Resumen General:

Característica	Descripción	Ejemplo en Institución Educativa
Consultas Anidadas	Subconsultas dentro de consultas para condiciones complejas.	Materias con el mínimo de horas por semana.
Stored Procedures	Procedimientos almacenados para ejecutar operaciones repetitivas.	Contar alumnos inscritos en una materia.
Checks	Restricciones para validar los datos al insertarlos o actualizarlos.	Verificar que las horas de una materia estén entre 1 y 10.
Triggers	Disparadores que ejecutan acciones automáticas en respuesta a eventos.	Registrar cada inscripción de un alumno en una materia.
Views	Consultas almacenadas como tablas virtuales para simplificar acceso y mejorar seguridad.	Vista que muestra la información de alumnos junto con sus materias y fechas de inscripción.