# Laboratorio 2

1. **¿Qué ventajas encontraron al encapsular lógica en funciones en lugar de repetir consultas SQL?**

   Encapsular la lógica en funciones nos permitió centralizar reglas de negocio como el cálculo de ocupación o la recomendación de actividades. Esto reduce la duplicación de código, facilita el mantenimiento y mejora la claridad general del sistema, ya que evita repetir consultas SQL complejas en distintos lugares.

2. **¿Qué criterios usaron para decidir cuándo implementar una función y cuándo una vista?**

   Utilizamos funciones cuando necesitábamos lógica dinámica, parámetros o condiciones específicas que varían según el contexto. Por ejemplo, al recomendar actividades o calcular porcentajes. En cambio, empleamos vistas para presentar datos estáticos o consolidados que pueden ser reutilizados por otras consultas o reportes, como los resúmenes de inscripciones.

3. **¿Qué limitaciones encontraron al trabajar con procedimientos almacenados en comparación con funciones?**

   Una limitación importante es que los procedimientos almacenados no pueden ser invocados directamente dentro de una consulta SQL, a diferencia de las funciones. Además, los procedimientos están orientados a ejecutar acciones complejas (como inserciones o validaciones múltiples), pero no devuelven resultados que puedan integrarse fácilmente en otras consultas.

4. **¿Creen que el trigger que implementaron garantiza la integridad de los datos en todos los escenarios posibles? Justifiquen su respuesta.**

   El trigger implementado garantiza que no se supere la capacidad máxima definida para cada actividad, lo cual protege una regla de integridad crítica del sistema. Sin embargo, en entornos con alta concurrencia o múltiples transacciones simultáneas, podrían surgir condiciones de carrera que no se resuelven solo con un trigger, por lo que sería necesario complementar con control transaccional más robusto.

5. **¿Cómo adaptarían su solución para que escale en una base de datos con millones de registros?**

   Para escalar eficientemente, implementaríamos índices en campos clave como id_actividad, id_persona y fecha, optimizaríamos subconsultas en funciones, y utilizaríamos paginación para resultados extensos. También consideraríamos particionar tablas como inscripcion por fecha, y revisaríamos el uso de vistas materiales si fuera necesario para mejorar el rendimiento.

6. **¿Qué escenarios podrían romper su lógica actual si no existiera el trigger?**

   Si no existiera el trigger de verificación de capacidad, podrían registrarse inscripciones que superen el límite de cupos definidos para una actividad. Esto invalidaría los datos mostrados en las vistas y funciones que dependen de un conteo exacto de inscritos, afectando decisiones basadas en disponibilidad o niveles de ocupación.

7. **¿Qué dificultades enfrentaron al definir funciones que devuelven conjuntos de resultados?**

   Tuvimos que asegurarnos de que los tipos y nombres de columnas especificados en RETURNS TABLE coincidieran con los seleccionados en el RETURN QUERY. También fue necesario controlar el uso correcto de JOIN y GROUP BY para evitar resultados vacíos o erróneos. La construcción de lógica condicional dentro de estas funciones también requirió una validación cuidadosa.

8. **¿Consideran que su diseño sería compatible con una arquitectura de microservicios? ¿Por qué sí o por qué no?**

   Sí, nuestro diseño es compatible, ya que encapsula la lógica de negocio dentro de funciones y procedimientos reutilizables. Esto facilita exponer dicha lógica como parte de APIs en un entorno de microservicios. Sin embargo, para una implementación completamente desacoplada, sería necesario diseñar servicios independientes por dominio y posiblemente descomponer el esquema actual en módulos más pequeños.

## 9. ¿Cómo reutilizarían las vistas que definieron en reportes o en otros sistemas?

Las vistas definidas se pueden usar directamente en sistemas de generación de reportes o herramientas de visualización como Power BI. También pueden ser consumidas por APIs para integrar los datos con otras plataformas. Esto permite mantener consistencia en los reportes, ya que encapsulan la lógica de presentación y resumen.

## 10. ¿Qué aprendieron sobre la separación entre la lógica de negocio y la lógica de persistencia al hacer este laboratorio?

Aprendimos que separar la lógica de negocio dentro de funciones, procedimientos y triggers permite mantener el código más ordenado, reutilizable y fácil de mantener. Esto evita mezclar la lógica directamente en las consultas o en la aplicación, lo que además facilita el trabajo colaborativo y la integración con otros sistemas.