Técnica 1: Uso de Varios ResultSets (importes1.java)

Ventajas:

1. **Modularidad:** Cada consulta se realiza de manera independiente, lo que facilita la comprensión y el mantenimiento del código.
2. **Flexibilidad:** Permite realizar operaciones adicionales entre consultas, como cálculos intermedios o transformaciones de datos.
3. **Control:** Ofrece un mayor control sobre el flujo de datos y la lógica de negocio, permitiendo ajustes específicos en cada paso del proceso.

Inconvenientes:

1. **Rendimiento:** Realizar múltiples consultas puede ser ineficiente, especialmente si la base de datos es grande o si hay muchas relaciones entre tablas.
2. **Complejidad:** El código puede volverse complejo y difícil de seguir debido a la cantidad de consultas y ResultSets.
3. **Consistencia:** Existe el riesgo de inconsistencias si los datos cambian entre consultas, lo que puede llevar a resultados incorrectos.

Escenarios:

- **Adecuado:** Cuando se necesita realizar operaciones complejas o específicas entre consultas, o cuando se requiere un control detallado sobre cada paso del proceso.
- **No Adecuado:** En aplicaciones de alto rendimiento donde la eficiencia es crítica, o cuando se trabaja con grandes volúmenes de datos.

Técnica 2: Uso de una Única Consulta SQL (importes2.java)

Ventajas:

1. **Eficiencia:** Realizar una única consulta SQL reduce la sobrecarga de comunicación con la base de datos, mejorando el rendimiento.
2. **Simplicidad:** El código es más sencillo y fácil de mantener, ya que toda la lógica se maneja en una sola consulta.
3. **Consistencia:** Al realizar una única consulta, se garantiza que los datos no cambien entre operaciones, evitando inconsistencias.

Inconvenientes:

1. **Complejidad de la Consulta:** Las consultas SQL pueden volverse complejas y difíciles de leer, especialmente si se necesitan múltiples uniones y agregaciones.
2. **Limitaciones de SQL:** Algunas operaciones complejas pueden ser difíciles o imposibles de realizar directamente en SQL, lo que limita la flexibilidad.
3. **Depuración:** Depurar problemas en consultas SQL complejas puede ser más difícil que en código Java.

Escenarios:

- **Adecuado:** En aplicaciones donde el rendimiento es crítico y se necesita procesar grandes volúmenes de datos de manera eficiente.
- **No Adecuado:** Cuando se requiere realizar operaciones complejas o específicas que no se pueden manejar fácilmente en una única consulta SQL.