**Otimização e Indexação – Análise e Melhorias**

**1. Índices sugeridos**

Com base nas consultas anteriores, é recomendado criar os seguintes **índices**:

-- Otimiza consultas por data (usadas em filtros, CTEs, tendências)

CREATE INDEX idx\_pedidos\_data\_pedido ON pedidos(data\_pedido);

-- Otimiza JOIN entre pedidos e clientes

CREATE INDEX idx\_pedidos\_id\_cliente ON pedidos(id\_cliente);

-- Otimiza JOIN entre itens\_pedido e pedidos

CREATE INDEX idx\_itens\_pedido\_id\_pedido ON itens\_pedido(id\_pedido);

-- Otimiza JOIN entre itens\_pedido e produtos

CREATE INDEX idx\_itens\_pedido\_id\_produto ON itens\_pedido(id\_produto);

**2. Justificativas por Tipo de Consulta:**

* **Top 5 Produtos mais Rentáveis:** se beneficia do uso de índices em id\_produto e data\_pedido, além da otimização de agregações com SUM().
* **Tendência de Vendas por Mês:** o uso de TO\_CHAR(data\_pedido, 'YYYY-MM') pode ser substituído por DATE\_TRUNC('month', data\_pedido) para melhorar a performance e permitir o uso de índices.
* **Clientes Inativos:** uso de LEFT JOIN e agregação por MAX(data\_pedido) é sensível a índices em id\_cliente e data\_pedido.
* **Detecção de Anomalias:** o cálculo de SUM(quantidade \* preco\_unitario) por pedido pode ser acelerado com índices em id\_pedido e agrupamentos bem estruturados.

**3. Outras Otimizações Sugeridas:**

* Evitar o uso de funções como ROUND, TO\_CHAR ou EXTRACT na cláusula WHERE, para não prejudicar a utilização de índices.
* Criar VIEWs para resultados recorrentes (ex: produtos mais vendidos, análise de tendências), facilitando reutilização e cache pelo PostgreSQL.
* Considerar materialização de CTEs pesadas, se reutilizadas em várias partes da consulta.

**Conclusão:**  
A criação de índices nos campos mais usados em filtros, JOINs e agrupamentos torna o banco mais eficiente e escalável. Otimizações adicionais, como simplificação de funções e uso de VIEWs, garantem um ambiente preparado para grandes volumes de dados e análises em tempo real.