1. Consultas SQL mais avançadas

- **Subconsultas (Subqueries)**: Aprender a fazer consultas dentro de outras consultas. Útil para cenários complexos de seleção de dados.
- **Agregação de dados**: Uso de funções como COUNT, SUM, AVG, MIN, MAX para obter informações agregadas.
- **GROUP BY e HAVING**: Agrupamento de dados com base em critérios específicos e filtragem de grupos com HAVING.
- **Ordenação e Limitação**: Uso de ORDER BY e LIMIT para controlar a ordem dos resultados e o número de linhas retornadas.

2. Transações no Banco de Dados

- **Transações**: Estudo de como agrupar múltiplas operações de banco de dados em uma única transação. Aprenda sobre BEGIN TRANSACTION, COMMIT, e ROLLBACK.
- **Controle de concorrência**: Entender como diferentes transações simultâneas são gerenciadas e evitar problemas como deadlock e inconsistência de dados.

3. Normalização e Desnormalização

- **Normalização**: Aprender sobre as formas normais (1NF, 2NF, 3NF, etc.) para organizar as tabelas de modo eficiente, evitando redundância e problemas de integridade.
- **Desnormalização**: Entender quando desnormalizar pode ser útil para melhorar o desempenho em cenários específicos, como sistemas de leitura intensiva.

4. Índices

- **Criação de índices**: Aprender como e quando criar índices em colunas para melhorar o desempenho das consultas.
- **Índices compostos**: Entendimento de índices que cobrem mais de uma coluna.
- **Impacto dos índices**: Saber equilibrar o uso de índices para melhorar a performance de leitura sem prejudicar as operações de escrita.

5. Stored Procedures e Triggers

- **Stored Procedures**: Estudo de procedimentos armazenados no banco de dados que podem ser chamados para executar uma sequência de comandos SQL.
- **Triggers**: Criar gatilhos que disparam automaticamente ações no banco de dados em resposta a eventos como INSERT, UPDATE, ou DELETE.

6. Backup e Recuperação

- **Backup de Banco de Dados**: Técnicas de backup para garantir que seus dados possam ser restaurados em caso de falha.
- **Recuperação**: Como restaurar um banco de dados a partir de backups ou corrigir dados corrompidos.

7. ORM (Object-Relational Mapping)

- **Introdução ao ORM**: Explore como integrar um ORM (como SQLAlchemy ou Django ORM) no Python para mapear classes diretamente para tabelas do banco de dados, facilitando o trabalho com bancos de dados de forma mais orientada a objetos.
- **Vantagens e desvantagens**: Entender os benefícios e limitações do uso de ORMs em comparação com o SQL manual.

8. Performance e Otimização

- **Análise de desempenho**: Aprender a usar o EXPLAIN ou EXPLAIN ANALYZE para entender o desempenho de suas consultas.
- **Cache de consultas**: Técnicas para cachear resultados de consultas frequentes e melhorar a performance geral.
- **Particionamento de tabelas**: Divisão de tabelas grandes em partes menores para otimizar consultas e operações de manutenção.

9. Modelagem de Banco de Dados

- **Modelagem conceitual e lógica**: Criação de diagramas ER (Entidade-Relacionamento) para planejar o design do banco de dados.
- **Modelagem física**: Implementação de modelos lógicos em esquemas físicos, considerando desempenho e integridade.

10. SQL Avançado: CTEs, Views e Funções

- **Common Table Expressions (CTEs)**: Uso de expressões de tabela comuns para simplificar consultas complexas.
- **Views**: Criação e uso de views (visões) como consultas armazenadas reutilizáveis.
- **Funções definidas pelo usuário**: Criação de funções SQL personalizadas que podem ser usadas em suas consultas.

11. NoSQL e Bancos de Dados Não Relacionais

- **Introdução ao NoSQL**: Explore bancos de dados não relacionais, como MongoDB, Redis, Cassandra, e compreenda em quais situações eles são mais adequados do que bancos de dados relacionais.
- **Diferenças entre SQL e NoSQL**: Compreender a diferença entre a modelagem de dados relacional e não relacional.