INSTITUTO INFNET

ESCOLA SUPERIOR DE TECNOLOGIA

GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

Lista de Exercícios de Revisão SQL 2

60 exercícios

Prof: Leonardo Glória

Aluno: Richard de Jesus Cabral Alves.

Exercícios

PostgreSQL — Lista de 60 Exercícios

Objetivo: praticar o ecossistema criado no `postgres-labs` cobrindo DDL, DML e consultas variadas.

Como entregar

- Crie um arquivo `respostas.sql` com as soluções numeradas (1 a 60) em comentários acima de cada query/comando.
- Quando a tarefa pedir *explicação curta*, escreva em comentário SQL (`--`).
- Para exercícios de `VIEW/MATERIALIZED VIEW`, coloque o `DROP ... IF EXISTS` antes, para facilitar a correção.

A) Aquecimento: DDL/DML e Modelagem (1–10)

1. **Criar tabela** `comercio.cupons` com colunas: `codigo TEXT PK`, `percentual NUMERIC(5,2) CHECK (percentual BETWEEN 0 AND 100)`, `expira_em DATE`. Insira 3 cupons (um expirado ontem, dois válidos). *Pergunta direta:* por que usar `NUMERIC` em vez de `REAL` para dinheiro? (responda em 1 linha)

2. **Adicionar coluna** `cupom_codigo TEXT` em `comercio.pedidos`, com FK opcional para `comercio.cupons(codigo)`; em `ON DELETE` ponha `SET NULL`.

3. **Criar tipo ENUM** `ufo.severidade` com valores `('BAIXA','MEDIA','ALTA')` e adicionar coluna `severidade ufo.severidade` em `ufo.avistamentos` com default `MEDIA`.

4. **Crie índice BTREE** em `comercio.pedidos(criado_em)` e explique em 1 frase quando BTREE é melhor que GIN.

5. **Inserir dados**: crie 5 `comercio.clientes` e 5 `comercio.pedidos` (misture `ABERTO/PAGO/ENVIADO`). Insira itens em 3 pedidos.

6. **UPDATE com regra**: aumente `preco` em 10% apenas dos `comercio.produtos` cuja `atributos->>'switch'` = `'brown'` **ou** que tenham tag `'gamer'`.

7. **DELETE seguro**: remova todos os `comercio.pagamentos` com `valor = 0`. Antes, mostre-os com um `SELECT` usando as mesmas condições (sem apagar ainda).

8. **ALTER TABLE**: torne `org.funcionarios.email` obrigatório (`NOT NULL`) criando a coluna se não existir; se já existir com nulos, corrija usando um email sintético `'<nome>@ex.com'` antes do `ALTER`.

9. **Pergunta direta:** qual a diferença entre `DELETE FROM` e `TRUNCATE`? Cite 2 diferenças práticas em 2 linhas.

10. **Gerar chaves**: ajuste `comercio.produtos` para usar `GENERATED ALWAYS AS IDENTITY` (se ainda estiver `SERIAL`). Explique em 1 linha a vantagem sobre `SERIAL`.

B) Consultas básicas e filtros (11–20)

11. Liste `id, nome, preco` de `comercio.produtos` com `preco BETWEEN 100 AND 400` **e** (`nome ILIKE '%sql%'` **ou** tem tag `'office'`). Ordene por `preco DESC`, pegue só 5.

12. Liste clientes sem email (`IS NULL`). *Pergunta direta:* por que `WHERE email = NULL` não funciona? Reescreva corretamente.

13. Mostre `pedidos` com `status` em (`ABERTO`,`PAGO`) **e** `criado_em` nos últimos 60 dias. Mostre `id, status, criado_em`.

14. Busque `produtos` cujo `atributos` contenha **todas** as chaves `switch` **e** `layout`. Dica: operadores JSONB `?` e `?&`/`@>`.

15. Traga `produtos` sem nenhuma tag (array JSON vazio **ou** `NULL`).

16. Liste `itens_pedido` calculando `subtotal = quantidade*preco_unit`. Filtre apenas `subtotal >= 300`. Ordene por `subtotal DESC`.

17. *Pergunta direta:* explique `COALESCE` com um exemplo em 1 linha usando `cupom_codigo`.

18. Liste `produtos` cujo nome **não** contenha as palavras `('Mouse','Headset')` (case-insensitive).

19. Mostre os 10 clientes criados mais recentemente (considerando `created_at`).

20. Liste `funcionarios` do `org` contratados *antes* de 2024-01-01 com `skills`

contendo `'java'` **ou** `'sql'` (ARRAY).

C) JOINs e agregações (21–30)

21. Para cada `pedido`, traga `id, cliente, status, total_itens` (soma de `quantidade*preco_unit`). Inclua pedidos sem itens (soma 0).

22. Para cada `cliente`, traga `qtd_pedidos` e `valor_total` (soma global). Ordene por `valor_total DESC`. Mostre top 5.

23. Liste `produtos` e o total vendido de cada um. Mostre só os que venderam **algo** (> 0).

24. Descubra quais `categorias` **nunca** tiveram produtos vendidos (zero vendas).

25. Para cada `categoria`, mostre o **ticket médio por pedido** (use join via `produto_categorias`).

26. *Pergunta direta:* diferença entre `INNER JOIN` e `LEFT JOIN` em 1 linha com mini-exemplo.

27. Quantos pedidos com cupom válido (não expirado)? Considere `cupom_codigo` no pedido e compare com `cupons.expira_em`.

28. Média de preço por `tag` (explodindo `tags`). Mostre `tag` e `avg(preco)`.

29. Percentual de pedidos por `status` (use `count(*)` e proporção sobre o total). Arredonde para 2 casas.

30. Top 3 clientes por **valor total pago** (considere `comercio.pagamentos.valor` somados por pedido).

D) Operadores de conjunto (31–35)

31. `UNION`: IDs de produtos que são `('Teclado Mecânico','Mouse Vertical')` **ou** têm a tag `'office'` — sem duplicar.

32. `UNION ALL`: repita o exercício anterior com `UNION ALL` e **conte** o total de linhas (mostre que duplica quando cai nas duas regras).
Pergunta direta: em 1 linha, diferença de `UNION` vs `UNION ALL`.

33. `INTERSECT`: SKUs que atendam **ambas** as condições: nome ILIKE `%sql%` **e**

tenham tag `'estudo'`.

34. `EXCEPT`: produtos com tag `'gamer'` **exceto** os com `preco < 400`.

35. Combine `clientes` criados no último mês **UNION** clientes que fizeram pedidos acima de 1000 em qualquer data (IDs únicos).

E) Subqueries e EXISTS (36–40)

36. Clientes **com** pedidos: use `EXISTS`. Liste `id, nome`.

37. Clientes **sem** pedidos: use `NOT EXISTS` (anti-join).

38. Produtos com preço **acima** da média de `comercio.produtos` (subquery escalar).

39. Para cada `pedido`, mostre `id` e um campo `tem_headset_gamer BOOLEAN` se **existe** item cujo produto tenha tag `'gamer'` **e** nome ILIKE `%headset%`.

40. *Pergunta direta:* cite 2 vantagens de `EXISTS/NOT EXISTS` sobre `IN/NOT IN` em subqueries correlacionadas (2 linhas).

F) Datas e faixas (41–45)

41. Pedidos **entre** `now() - interval '90 days'` e `now()`. Inclua bordas.

42. Agrupe faturamento **mensal** (por `date_trunc('month', criado_em)`) mostrando `mes`, `valor_total`, `qtd_pedidos`.

43. Gere uma série de dias do último mês (`generate_series`) e faça `LEFT JOIN` com vendas diárias para mostrar **dias sem vendas** como zero.

44. *Pergunta direta:* `TIMESTAMP WITH TIME ZONE` vs `TIMESTAMP WITHOUT TIME ZONE` — quando escolher um ou outro? (2 linhas)

45. Extraia `dow` (dia da semana) e mostre qual dia tem maior número de pedidos (0–6).

G) CTE (WITH) e janelamento (46–50)

46. Use `WITH` para calcular `somas_por_pedido` e depois agregue por `status` (soma total dos pedidos por status).

47. `WITH` para montar uma "tabela" de parâmetros (`limite NUMERIC := 500`) e reutilizar em duas consultas no mesmo bloco.

48. Window function: para cada `categoria`, calcule o **ranking** de produtos por quantidade vendida (`dense_rank()` desc). Mostre top 3 por categoria.

49. Window function: calcule a **média móvel** de 7 dias do faturamento diário.

50. *CTE recursiva simples*: gere números de 1 a 10 e una ao número de pedidos criados nesses 10 últimos dias (mapa `dia_n -> qtd_pedidos`).

H) Arrays e JSONB (51–55)

51. Arrays: liste `funcionarios` que tenham **todas** as skills em `{'java','sql'}`. Dica: `@>` em arrays.

52. Arrays: exploda `skills` com `unnest` e conte quantos funcionários possuem cada skill (ranking desc).

53. JSONB: traga `produtos` cujo `atributos` contenha `{"surround": true}` e `{"layout":"abnt2"}` ao mesmo tempo. Use `@>`.

54. JSONB: produza colunas derivadas `dpi_int` (cast de `atributos->>'dpi'`) e `tem_rgb BOOLEAN` (`tags @> '["rgb"]'`). Filtre `dpi_int >= 8000 OR tem_rgb`.

55. *Pergunta direta:* JSON vs JSONB — cite 2 diferenças práticas, especialmente para indexação e performance (2 linhas).

I) Views e Materialized Views (56–60)

56. VIEW atualizável `comercio.vw_pedidos_abertos` (somente `status='ABERTO'`) com `WITH CHECK OPTION`. Teste um `UPDATE` que viole a condição (deve falhar).

57. VIEW `comercio.vw_itens_enriquecidos` trazendo item + nome do

produto + nome do cliente + status do pedido. (Somente SELECT; sem agregação.)

58. VIEW `comercio.vw_vendas_por_mes` produzindo `mes, valor_total, qtd_pedidos`. Valide com 2 queries usando meses diferentes.

59. MATERIALIZED VIEW `comercio.mv_top_produtos` com `id_produto, nome, qtd_vendida`. Crie **índice único** necessário para `REFRESH CONCURRENTLY`. Faça um `REFRESH` após inserir novos itens e comprove a mudança.

60. *Pergunta direta (views):* em 2 linhas, quando preferir **MV** em vez de **VIEW**? Cite um trade-off importante.

Dicas gerais

- Para segurança de correção, prefixe todas as suas views/MVs com `DROP VIEW/MATERIALIZED VIEW IF EXISTS ...;`
- Em JSONB use índices GIN (`jsonb_path_ops` ou padrão). Em arrays, filtros com `@>` podem usar GIN em `TEXT[]`.
- `EXPLAIN ANALYZE` é seu amigo para comparar consulta direta vs MV.

28.