```
-- ============================================================
-- Lista_Consultas.sql
-- Banco: plataforma_cursos
--(PostgreSQL)
-- ============================================================

-- Conectar no banco no psql, se necessário:
-- \c plataforma_cursos

-- 1) Listar nome, cidade, uf de todos os alunos.
SELECT nome, cidade, uf FROM aluno;

-- 2) Exibir titulo, nivel, preco de todos os cursos publicados.
SELECT titulo, nivel, preco FROM curso WHERE publicado = TRUE;

-- 3) Mostrar nome, especialidade de instrutores com e-mail não nulo.
SELECT nome, especialidade FROM instrutor WHERE email IS NOT NULL;

-- 4) Listar título das aulas do curso "PostgreSQL do Zero" ordenadas por ordem.
SELECT a.titulo
FROM aula a
JOIN curso c ON c.id_curso = a.id_curso
WHERE c.titulo = 'PostgreSQL do Zero'
ORDER BY a.ordem;

-- 5) Listar nome dos alunos e data_cadastro em ordem crescente de data.
SELECT nome, data_cadastro FROM aluno ORDER BY data_cadastro ASC;

-- 6) Cursos com preco < 200.
SELECT id_curso, titulo, preco FROM curso WHERE preco < 200;

-- 7) Alunos da UF = 'DF' ou 'GO'.
```

```
SELECT id_aluno, nome, uf FROM aluno WHERE uf IN ('DF','GO');


-- 8) Matrículas ativas a partir de 2025-03-01.

SELECT * FROM matricula WHERE status = 'ativa' AND data_matricula >= DATE '2025-03-01';


-- 9) Pagamentos método pix e status aprovado.

SELECT * FROM pagamento WHERE metodo = 'pix' AND status = 'aprovado';


-- 10) Aulas com duração entre 40 e 50 min (inclusive).

SELECT * FROM aula WHERE duracao_min BETWEEN 40 AND 50;


-- 11) Cursos ordenados por nivel e depois preco desc.

SELECT id_curso, titulo, nivel, preco

FROM curso

ORDER BY nivel, preco DESC;


-- 12) Alunos ordenados por cidade, exibindo uf como "Estado".

SELECT nome, cidade, uf AS "Estado" FROM aluno ORDER BY cidade;


-- 13) Instrutores ordenados por nome exibindo especialidade como "Área".

SELECT nome, especialidade AS "Área" FROM instrutor ORDER BY nome;


-- 14) Aulas de "DevOps na Prática" ordenadas por duracao_min desc.

SELECT a.titulo, a.duracao_min

FROM aula a

JOIN curso c ON c.id_curso = a.id_curso

WHERE c.titulo = 'DevOps na Prática'

ORDER BY a.duracao_min DESC;


-- 15) Exibir titulo e preco*0.9 como preco_com_desconto (10%).

SELECT titulo, ROUND(preco * 0.9, 2) AS preco_com_desconto FROM curso;
```

```sql
-- 16) Contar quantos cursos existem por nivel.
SELECT nivel, COUNT(*) AS qtde_cursos
FROM curso
GROUP BY nivel
ORDER BY nivel;

-- 17) Média de duracao_min das aulas por id_curso.
SELECT id_curso, AVG(duracao_min)::NUMERIC(10,2) AS media_duracao
FROM aula
GROUP BY id_curso
ORDER BY id_curso;

-- 18) Soma de valor de pagamentos aprovados por metodo.
SELECT metodo, SUM(valor) AS total
FROM pagamento
WHERE status = 'aprovado'
GROUP BY metodo
ORDER BY total DESC;

-- 19) Quantidade de alunos por uf.
SELECT uf, COUNT(*) AS qtde
FROM aluno
GROUP BY uf
ORDER BY qtde DESC;

-- 20) Preço médio por categoria.
SELECT cat.nome AS categoria, AVG(c.preco)::NUMERIC(10,2) AS preco_medio
FROM curso c
JOIN categoria cat ON cat.id_categoria = c.id_categoria
GROUP BY cat.nome
ORDER BY preco_medio DESC;
```

```sql
-- 21) Cursos por instrutor com total >= 2 (HAVING).
SELECT i.nome AS instrutor, COUNT(*) AS total_cursos
FROM curso c
JOIN instrutor i ON i.id_instrutor = c.id_instrutor
GROUP BY i.nome
HAVING COUNT(*) >= 2
ORDER BY total_cursos DESC, i.nome;

-- 22) Pagamentos aprovados por curso com soma > 500.
SELECT c.titulo, SUM(p.valor) AS total_aprovado
FROM pagamento p
JOIN matricula m ON m.id_matricula = p.id_matricula
JOIN curso c ON c.id_curso = m.id_curso
WHERE p.status = 'aprovado'
GROUP BY c.titulo
HAVING SUM(p.valor) > 500
ORDER BY total_aprovado DESC;

-- 23) Aulas por curso com média de duração > 40 min.
SELECT c.titulo, AVG(a.duracao_min)::NUMERIC(10,2) AS media
FROM aula a
JOIN curso c ON c.id_curso = a.id_curso
GROUP BY c.titulo
HAVING AVG(a.duracao_min) > 40
ORDER BY media DESC;

-- 24) Matrículas ativas por curso com count >= 2.
SELECT c.titulo, COUNT(*) AS ativas
FROM matricula m
JOIN curso c ON c.id_curso = m.id_curso
WHERE m.status = 'ativa'
GROUP BY c.titulo
```

```
HAVING COUNT(*) >= 2
ORDER BY ativas DESC, c.titulo;

-- 25) Alunos por mês de cadastro.
SELECT DATE_TRUNC('month', data_cadastro)::DATE AS mes, COUNT(*) AS qtd
FROM aluno
GROUP BY 1
ORDER BY 1;

-- 26) Curso e nome do instrutor (JOIN).
SELECT c.titulo, i.nome AS instrutor
FROM curso c
JOIN instrutor i ON i.id_instrutor = c.id_instrutor
ORDER BY c.titulo;

-- 27) Nome do aluno, titulo do curso, status da matrícula.
SELECT a.nome AS aluno, c.titulo AS curso, m.status
FROM matricula m
JOIN aluno a ON a.id_aluno = m.id_aluno
JOIN curso c ON c.id_curso = m.id_curso
ORDER BY a.nome, c.titulo;

-- 28) Pagamentos (valor, método, status) com nome do aluno e titulo do curso.
SELECT a.nome AS aluno, c.titulo AS curso, p.valor, p.metodo, p.status
FROM pagamento p
JOIN matricula m ON m.id_matricula = p.id_matricula
JOIN aluno a ON a.id_aluno = m.id_aluno
JOIN curso c ON c.id_curso = m.id_curso
ORDER BY p.data_pagamento DESC;

-- 29) Aulas (titulo) com titulo do curso correspondente.
SELECT c.titulo AS curso, a.ordem, a.titulo AS aula
```

```
FROM aula a
JOIN curso c ON c.id_curso = a.id_curso
ORDER BY c.titulo, a.ordem;

-- 30) Avaliações: nome do aluno, titulo do curso, nota.
SELECT a.nome AS aluno, c.titulo AS curso, av.nota
FROM avaliacao av
JOIN aluno a ON a.id_aluno = av.id_aluno
JOIN curso c ON c.id_curso = av.id_curso
ORDER BY av.data_avaliacao DESC;

-- 31) Cursos com preco > média de todos os cursos.
SELECT id_curso, titulo, preco
FROM curso
WHERE preco > (SELECT AVG(preco) FROM curso)
ORDER BY preco DESC;

-- 32) Alunos sem nenhuma avaliação.
SELECT a.id_aluno, a.nome
FROM aluno a
WHERE NOT EXISTS (
  SELECT 1 FROM avaliacao av WHERE av.id_aluno = a.id_aluno
)
ORDER BY a.nome;

-- 33) Instrutores que não possuem cursos de nivel 'Avançado'.
SELECT i.id_instrutor, i.nome
FROM instrutor i
WHERE NOT EXISTS (
  SELECT 1 FROM curso c
  WHERE c.id_instrutor = i.id_instrutor
    AND c.nivel = 'Avançado'
```

```
)
ORDER BY i.nome;

-- 34) Cursos com mais de 2 aulas.
SELECT c.id_curso, c.titulo
FROM curso c
WHERE (SELECT COUNT(*) FROM aula a WHERE a.id_curso = c.id_curso) > 2
ORDER BY c.titulo;

-- 35) Alunos com valor total pago (aprovado) acima de 400.
SELECT a.id_aluno, a.nome, SUM(p.valor) AS total_pago
FROM pagamento p
JOIN matricula m ON m.id_matricula = p.id_matricula
JOIN aluno a ON a.id_aluno = m.id_aluno
WHERE p.status = 'aprovado'
GROUP BY a.id_aluno, a.nome
HAVING SUM(p.valor) > 400
ORDER BY total_pago DESC;

-- 36) Matrículas realizadas no mesmo mês da data_publicacao do curso.
SELECT a.nome AS aluno, c.titulo AS curso, m.data_matricula, c.data_publicacao
FROM matricula m
JOIN curso c ON c.id_curso = m.id_curso
JOIN aluno a ON a.id_aluno = m.id_aluno
WHERE DATE_TRUNC('month', m.data_matricula) = DATE_TRUNC('month',
c.data_publicacao)
ORDER BY m.data_matricula;

-- 37) Para cada nivel, curso mais caro (window ROW_NUMBER).
WITH ranked AS (
  SELECT nivel, id_curso, titulo, preco,
       ROW_NUMBER() OVER (PARTITION BY nivel ORDER BY preco DESC) AS rk
  FROM curso
```

```
)
SELECT nivel, id_curso, titulo, preco
FROM ranked
WHERE rk = 1
ORDER BY nivel;

-- 38) Top 3 cursos por soma de pagamentos aprovados (window).
WITH totais AS (
  SELECT c.id_curso, c.titulo, SUM(p.valor) AS total
  FROM pagamento p
  JOIN matricula m ON m.id_matricula = p.id_matricula
  JOIN curso c ON c.id_curso = m.id_curso
  WHERE p.status = 'aprovado'
  GROUP BY c.id_curso, c.titulo
)
SELECT id_curso, titulo, total
FROM totais
ORDER BY total DESC
LIMIT 3;

-- 39) Diferença (dias) entre data_matricula e primeiro pagamento por matrícula.
WITH primeiro_pag AS (
  SELECT id_matricula, MIN(data_pagamento) AS primeira_data
  FROM pagamento
  GROUP BY id_matricula
)
SELECT m.id_matricula, a.nome AS aluno, c.titulo AS curso,
       m.data_matricula, pp.primeira_data,
       (pp.primeira_data - m.data_matricula) AS dias_ate_pagamento
FROM matricula m
LEFT JOIN primeiro_pag pp ON pp.id_matricula = m.id_matricula
JOIN aluno a ON a.id_aluno = m.id_aluno
```

```
JOIN curso c ON c.id_curso = m.id_curso
ORDER BY m.id_matricula;

-- 40) Nota média por curso e ranking (DENSE_RANK).
WITH medias AS (
  SELECT c.id_curso, c.titulo, AVG(av.nota)::NUMERIC(10,2) AS media
  FROM curso c
  LEFT JOIN avaliacao av ON av.id_curso = c.id_curso
  GROUP BY c.id_curso, c.titulo
)
SELECT id_curso, titulo, media,
       DENSE_RANK() OVER (ORDER BY media DESC NULLS LAST) AS posicao
FROM medias
ORDER BY posicao, titulo;

-- ============================================================
-- Lista_Consultas_Adicional_sql
-- Banco: plataforma_cursos
-- (41–90) com comandos SQL prontos (PostgreSQL)
-- ============================================================
-- \c plataforma_cursos

-- 41) Cursos publicados com preço >= 300, ordenados desc por preço.
SELECT id_curso, titulo, preco
FROM curso
WHERE publicado = TRUE AND preco >= 300
ORDER BY preco DESC, titulo;

-- 42) Alunos cujo nome começa com 'A' ou 'M'.
SELECT id_aluno, nome
FROM aluno
WHERE nome ILIKE 'A%%' OR nome ILIKE 'M%%'
```

```
ORDER BY nome;

-- 43) E-mails de alunos em formato minúsculo e domínio extraído.
SELECT email, LOWER(email) AS email_lower,
       SPLIT_PART(email, '@', 2) AS dominio
FROM aluno;

-- 44) Cursos com carga_horas entre 20 e 30 (inclusive).
SELECT id_curso, titulo, carga_horas
FROM curso
WHERE carga_horas BETWEEN 20 AND 30
ORDER BY carga_horas, titulo;

-- 45) Total de matrículas por status.
SELECT status, COUNT(*) AS qtde
FROM matricula
GROUP BY status
ORDER BY qtde DESC;

-- 46) Valor total pago (aprovado) por UF do aluno.
SELECT a.uf, SUM(p.valor) AS total_uf
FROM pagamento p
JOIN matricula m ON m.id_matricula = p.id_matricula
JOIN aluno a ON a.id_aluno = m.id_aluno
WHERE p.status = 'aprovado'
GROUP BY a.uf
ORDER BY total_uf DESC NULLS LAST;

-- 47) Cursos sem nenhuma aula (LEFT JOIN e IS NULL).
SELECT c.id_curso, c.titulo
FROM curso c
LEFT JOIN aula a ON a.id_curso = c.id_curso
```

```
WHERE a.id_aula IS NULL
ORDER BY c.titulo;

-- 48) Cursos com pelo menos uma avaliação nota 5.
SELECT DISTINCT c.id_curso, c.titulo
FROM curso c
JOIN avaliacao av ON av.id_curso = c.id_curso
WHERE av.nota = 5
ORDER BY c.titulo;

-- 49) Média de notas por curso (inclua cursos sem avaliação).
SELECT c.titulo, AVG(av.nota)::NUMERIC(10,2) AS media_nota
FROM curso c
LEFT JOIN avaliacao av ON av.id_curso = c.id_curso
GROUP BY c.titulo
ORDER BY media_nota DESC NULLS LAST, c.titulo;

-- 50) Ranking de alunos por total de pagamentos aprovados (SUM OVER).
WITH totais AS (
  SELECT a.id_aluno, a.nome, COALESCE(SUM(p.valor) FILTER (WHERE
p.status='aprovado'),0) AS total
  FROM aluno a
  LEFT JOIN matricula m ON m.id_aluno = a.id_aluno
  LEFT JOIN pagamento p ON p.id_matricula = m.id_matricula
  GROUP BY a.id_aluno, a.nome
)
SELECT id_aluno, nome, total,
      RANK() OVER (ORDER BY total DESC) AS posicao
FROM totais
ORDER BY posicao, nome;

-- 51) Top 5 alunos por quantidade de matrículas.
SELECT a.id_aluno, a.nome, COUNT(m.id_matricula) AS qtd
```

```
FROM aluno a
LEFT JOIN matricula m ON m.id_aluno = a.id_aluno
GROUP BY a.id_aluno, a.nome
ORDER BY qtd DESC, a.nome
LIMIT 5;

-- 52) Cursos cujo título contém a palavra 'PostgreSQL' (ILIKE).
SELECT id_curso, titulo
FROM curso
WHERE titulo ILIKE '%%PostgreSQL%%'
ORDER BY titulo;

-- 53) Diferença entre preco e preco_pago médio por curso.
SELECT c.titulo,
    c.preco,
    AVG(m.preco_pago)::NUMERIC(10,2) AS preco_pago_medio,
    (c.preco - AVG(m.preco_pago))::NUMERIC(10,2) AS diferenca_media
FROM curso c
JOIN matricula m ON m.id_curso = c.id_curso
GROUP BY c.id_curso, c.titulo, c.preco
ORDER BY diferenca_media DESC;

-- 54) Número de aulas por curso, ordenado decrescente.
SELECT c.titulo, COUNT(a.id_aula) AS aulas
FROM curso c
LEFT JOIN aula a ON a.id_curso = c.id_curso
GROUP BY c.titulo
ORDER BY aulas DESC, c.titulo;

-- 55) Cursos e seu instrutor, mostrando também a categoria.
SELECT c.titulo, i.nome AS instrutor, cat.nome AS categoria
FROM curso c
```

```
JOIN instrutor i ON i.id_instrutor = c.id_instrutor
JOIN categoria cat ON cat.id_categoria = c.id_categoria
ORDER BY cat.nome, c.titulo;

-- 56) Alunos com matrículas 'trancada' ou 'concluida'.
SELECT DISTINCT a.id_aluno, a.nome
FROM aluno a
JOIN matricula m ON m.id_aluno = a.id_aluno
WHERE m.status IN ('trancada', 'concluida')
ORDER BY a.nome;

-- 57) Cursos publicados em 2025-03 (mês/ano da data_publicacao).
SELECT id_curso, titulo, data_publicacao
FROM curso
WHERE DATE_TRUNC('month', data_publicacao) = DATE '2025-03-01'
ORDER BY titulo;

-- 58) Percentual de cursos por nivel (proporção).
WITH tot AS (SELECT COUNT(*) AS n FROM curso)
SELECT nivel,
    COUNT(*) AS qtd,
    ROUND(100.0 * COUNT(*) / (SELECT n FROM tot), 2) AS percentual
FROM curso
GROUP BY nivel
ORDER BY qtd DESC;

-- 59) Ultima matrícula por aluno (usando MAX(data_matricula)).
SELECT a.id_aluno, a.nome, MAX(m.data_matricula) AS ultima_matricula
FROM aluno a
LEFT JOIN matricula m ON m.id_aluno = a.id_aluno
GROUP BY a.id_aluno, a.nome
ORDER BY ultima_matricula DESC NULLS LAST;
```

```sql
-- 60) Cursos com preço abaixo da mediana (aproximação via percentil_cont).
WITH stats AS (
  SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY preco) AS mediana
FROM curso
)
SELECT c.id_curso, c.titulo, c.preco
FROM curso c, stats
WHERE c.preco < stats.mediana
ORDER BY c.preco;

-- 61) Pagamentos por método com contagem de aprovados e pendentes (FILTER).
SELECT metodo,
    COUNT(*) FILTER (WHERE status='aprovado') AS aprovados,
    COUNT(*) FILTER (WHERE status='pendente') AS pendentes,
    COUNT(*) FILTER (WHERE status='estornado') AS estornados
FROM pagamento
GROUP BY metodo
ORDER BY metodo;

-- 62) Alunos que possuem pelo menos 2 matrículas ativas.
SELECT a.id_aluno, a.nome, COUNT(*) AS qtd_ativas
FROM aluno a
JOIN matricula m ON m.id_aluno = a.id_aluno AND m.status='ativa'
GROUP BY a.id_aluno, a.nome
HAVING COUNT(*) >= 2
ORDER BY qtd_ativas DESC, a.nome;

-- 63) Cursos cujo instrutor é de nome que contém 'a' (case-insensitive).
SELECT c.titulo, i.nome AS instrutor
FROM curso c
JOIN instrutor i ON i.id_instrutor = c.id_instrutor
WHERE i.nome ILIKE '%%a%%'
```

```
ORDER BY i.nome, c.titulo;

-- 64) Duração total (min) de aulas por curso.
SELECT c.titulo, COALESCE(SUM(a.duracao_min),0) AS duracao_total
FROM curso c
LEFT JOIN aula a ON a.id_curso = c.id_curso
GROUP BY c.titulo
ORDER BY duracao_total DESC, c.titulo;

-- 65) Cursos sem nenhuma matrícula.
SELECT c.id_curso, c.titulo
FROM curso c
LEFT JOIN matricula m ON m.id_curso = c.id_curso
WHERE m.id_matricula IS NULL
ORDER BY c.titulo;

-- 66) Média de preços por categoria e nível.
SELECT cat.nome AS categoria, c.nivel, AVG(c.preco)::NUMERIC(10,2) AS preco_medio
FROM curso c
JOIN categoria cat ON cat.id_categoria = c.id_categoria
GROUP BY cat.nome, c.nivel
ORDER BY cat.nome, c.nivel;

-- 67) Primeira aula (menor ordem) de cada curso.
WITH primeiras AS (
  SELECT id_curso, MIN(ordem) AS min_ordem
  FROM aula
  GROUP BY id_curso
)
SELECT c.titulo, a.titulo AS primeira_aula, a.ordem
FROM primeiras p
JOIN aula a ON a.id_curso = p.id_curso AND a.ordem = p.min_ordem
```

```
JOIN curso c ON c.id_curso = a.id_curso
ORDER BY c.titulo;

-- 68) Última aula (maior ordem) de cada curso (window function).
SELECT c.titulo, a.titulo AS ultima_aula, a.ordem
FROM (
  SELECT a.*, ROW_NUMBER() OVER (PARTITION BY id_curso ORDER BY ordem DESC) AS rn
  FROM aula a
) a
JOIN curso c ON c.id_curso = a.id_curso
WHERE a.rn = 1
ORDER BY c.titulo;

-- 69) Cursos com avaliações média >= 4.
SELECT c.titulo, AVG(av.nota)::NUMERIC(10,2) AS media
FROM curso c
JOIN avaliacao av ON av.id_curso = c.id_curso
GROUP BY c.titulo
HAVING AVG(av.nota) >= 4
ORDER BY media DESC, c.titulo;

-- 70) Alunos e quantidade de cursos avaliados.
SELECT a.nome, COUNT(av.id_avaliacao) AS qtde_avaliacoes
FROM aluno a
LEFT JOIN avaliacao av ON av.id_aluno = a.id_aluno
GROUP BY a.nome
ORDER BY qtde_avaliacoes DESC, a.nome;

-- 71) Cursos com preço acima de todos os cursos do nível 'Iniciante' (ALL).
SELECT id_curso, titulo, preco
FROM curso
WHERE preco > ALL (SELECT preco FROM curso WHERE nivel = 'Iniciante')
```

```
ORDER BY preco DESC;

-- 72) Cursos com preço menor do que algum curso de 'Avançado' (ANY).
SELECT id_curso, titulo, preco
FROM curso
WHERE preco < ANY (SELECT preco FROM curso WHERE nivel = 'Avançado')
ORDER BY preco;

-- 73) Alunos sem matrícula (NOT EXISTS).
SELECT a.id_aluno, a.nome
FROM aluno a
WHERE NOT EXISTS (
  SELECT 1 FROM matricula m WHERE m.id_aluno = a.id_aluno
)
ORDER BY a.nome;

-- 74) Pagamentos por mês (ano-mês) e total aprovado.
SELECT TO_CHAR(data_pagamento, 'YYYY-MM') AS ano_mes,
      SUM(valor) FILTER (WHERE status='aprovado') AS total_aprovado
FROM pagamento
GROUP BY 1
ORDER BY 1;

-- 75) Cursos com pelo menos 1 matrícula 'concluida'.
SELECT DISTINCT c.id_curso, c.titulo
FROM curso c
JOIN matricula m ON m.id_curso = c.id_curso
WHERE m.status = 'concluida'
ORDER BY c.titulo;

-- 76) Percentual de matrículas por status (em relação ao total).
WITH tot AS (SELECT COUNT(*) AS n FROM matricula)
```

```
SELECT status,
    COUNT(*) AS qtd,
    ROUND(100.0 * COUNT(*) / (SELECT n FROM tot), 2) AS percentual
FROM matricula
GROUP BY status
ORDER BY qtd DESC;

-- 77) Cursos e receita total (pagamentos aprovados) ordenada decrescente.
SELECT c.titulo, COALESCE(SUM(p.valor) FILTER (WHERE p.status='aprovado'),0) AS receita
FROM curso c
LEFT JOIN matricula m ON m.id_curso = c.id_curso
LEFT JOIN pagamento p ON p.id_matricula = m.id_matricula
GROUP BY c.titulo
ORDER BY receita DESC, c.titulo;

-- 78) Intervalo (dias) entre data_publicacao do curso e primeira matrícula dele.
WITH primeira_m AS (
  SELECT id_curso, MIN(data_matricula) AS primeira
  FROM matricula
  GROUP BY id_curso
)
SELECT c.titulo, c.data_publicacao, pm.primeira,
    (pm.primeira - c.data_publicacao) AS dias_ate_primeira_matricula
FROM curso c
LEFT JOIN primeira_m pm ON pm.id_curso = c.id_curso
ORDER BY dias_ate_primeira_matricula;

-- 79) Nota média por instrutor (média das notas dos cursos daquele instrutor).
SELECT i.nome AS instrutor, AVG(av.nota)::NUMERIC(10,2) AS media_instrutor
FROM instrutor i
JOIN curso c ON c.id_instrutor = i.id_instrutor
JOIN avaliacao av ON av.id_curso = c.id_curso
```

```
GROUP BY i.nome
ORDER BY media_instrutor DESC NULLS LAST, i.nome;

-- 80) Cursos cujo total de aulas > média de aulas por curso.
WITH por_curso AS (
  SELECT id_curso, COUNT(*) AS aulas FROM aula GROUP BY id_curso
), media AS (
  SELECT AVG(aulas) AS m FROM por_curso
)
SELECT c.titulo, pc.aulas
FROM por_curso pc
JOIN curso c ON c.id_curso = pc.id_curso
JOIN media ON TRUE
WHERE pc.aulas > media.m
ORDER BY pc.aulas DESC;

-- 81) Alunos da região Sudeste (SP, RJ, MG, ES).
SELECT id_aluno, nome, uf
FROM aluno
WHERE uf IN ('SP','RJ','MG','ES')
ORDER BY nome;

-- 82) Títulos de cursos com INITCAP (formatação de título).
SELECT INITCAP(titulo) AS titulo_formatado
FROM curso
ORDER BY titulo_formatado;

-- 83) Cursos com desconto médio (preco - preco_pago médio) > 50.
SELECT c.titulo, (c.preco - AVG(m.preco_pago))::NUMERIC(10,2) AS desconto_medio
FROM curso c
JOIN matricula m ON m.id_curso = c.id_curso
GROUP BY c.titulo, c.preco
```

```
HAVING (c.preco - AVG(m.preco_pago)) > 50
ORDER BY desconto_medio DESC;

-- 84) Alunos com pagamentos estornados (listar aluno e quantidade).
SELECT a.nome, COUNT(*) AS estornos
FROM pagamento p
JOIN matricula m ON m.id_matricula = p.id_matricula
JOIN aluno a ON a.id_aluno = m.id_aluno
WHERE p.status = 'estornado'
GROUP BY a.nome
ORDER BY estornos DESC, a.nome;

-- 85) Cursos com maior ticket médio (SUM(valor aprovado) / qtd matrículas).
WITH base AS (
  SELECT c.id_curso, c.titulo,
         SUM(p.valor) FILTER (WHERE p.status='aprovado') AS receita,
         COUNT(DISTINCT m.id_matricula) AS qtd_mat
  FROM curso c
  LEFT JOIN matricula m ON m.id_curso = c.id_curso
  LEFT JOIN pagamento p ON p.id_matricula = m.id_matricula
  GROUP BY c.id_curso, c.titulo
)
SELECT id_curso, titulo,
       (receita / NULLIF(qtd_mat,0))::NUMERIC(10,2) AS ticket_medio
FROM base
ORDER BY ticket_medio DESC NULLS LAST, titulo;

-- 86) Cursos publicados entre 2025-02-01 e 2025-03-31.
SELECT id_curso, titulo, data_publicacao
FROM curso
WHERE data_publicacao BETWEEN DATE '2025-02-01' AND DATE '2025-03-31'
ORDER BY data_publicacao, titulo;
```

```sql
-- 87) Aulas com duração acima da média geral de duração.
WITH media AS (SELECT AVG(duracao_min) AS m FROM aula)
SELECT a.id_aula, a.titulo, a.duracao_min
FROM aula a, media
WHERE a.duracao_min > media.m
ORDER BY a.duracao_min DESC;

-- 88) Cursos cujo instrutor não tem e-mail cadastrado (LEFT JOIN + IS NULL).
SELECT c.titulo, i.nome AS instrutor
FROM curso c
LEFT JOIN instrutor i ON i.id_instrutor = c.id_instrutor
WHERE i.email IS NULL
ORDER BY c.titulo;

-- 89) Alunos com nome e e-mail concatenados.
SELECT nome || ' <' || email || '>' AS contato
FROM aluno
ORDER BY nome;

-- 90) Distribuição de notas (1 a 5) por curso (pivot simples com FILTER).
SELECT c.titulo,
    COUNT(*) FILTER (WHERE av.nota=1) AS n1,
    COUNT(*) FILTER (WHERE av.nota=2) AS n2,
    COUNT(*) FILTER (WHERE av.nota=3) AS n3,
    COUNT(*) FILTER (WHERE av.nota=4) AS n4,
    COUNT(*) FILTER (WHERE av.nota=5) AS n5
FROM curso c
LEFT JOIN avaliacao av ON av.id_curso = c.id_curso
GROUP BY c.titulo
ORDER BY c.titulo;
```