Exercício 1: Contagem de tarefas por prioridade em cada projeto

```
SELECT
    p.nome AS projeto,
    SUM(CASE WHEN t.prioridade = 'Baixa' THEN 1 ELSE 0 END) AS baixa,
    SUM(CASE WHEN t.prioridade = 'Normal' THEN 1 ELSE 0 END) AS normal,
    SUM(CASE WHEN t.prioridade = 'Alta' THEN 1 ELSE 0 END) AS alta,
    SUM(CASE WHEN t.prioridade = 'Urgente' THEN 1 ELSE 0 END) AS urgente
FROM
    Projetos p
LEFT JOIN
    Tarefas t ON p.id = t.id_projeto
GROUP BY
    p.id, p.nome
ORDER BY
    p.nome;
```

Exercício 2: Responsáveis com 3 ou mais tarefas

```
SELECT
    r.nome,
    COUNT(t.id) AS quantidade_tarefas
FROM
    Responsaveis r
JOIN
    Tarefas t ON r.id = t.id_responsavel_tarefa
GROUP BY
    r.id, r.nome
HAVING
    COUNT(t.id) >= 3
ORDER BY
    quantidade_tarefas DESC;
```

Exercício 3: Duração percebida média por status do projeto

```
SELECT
    p.status,
    ROUND(AVG(DATEDIFF(t.data_prevista_entrega, p.data_inicio)), 2) AS
duracao_percebida_media
FROM
    Projetos p
JOIN
    Tarefas t ON p.id = t.id_projeto
WHERE
    t.data_prevista_entrega IS NOT NULL
    AND p.data_inicio IS NOT NULL
GROUP BY
    p.status;
```

Exercício 4: Cargos dos responsáveis por tarefas em projetos em andamento

```
SELECT
    p.nome AS projeto,
    r.cargo,
    COUNT(t.id) AS quantidade_tarefas
FROM
    Projetos p
JOIN
    Tarefas t ON p.id = t.id_projeto
JOIN
    Responsaveis r ON t.id_responsavel_tarefa = r.id
WHERE
    p.status = 'Em Andamento'
GROUP BY
    p.nome, r.cargo
HAVING
    COUNT(t.id) > 0
ORDER BY
    p.nome, r.cargo;
```

Exercício 5: Top 5 projetos com mais tarefas em atraso

```
SELECT
    p.nome AS projeto,
    COUNT(t.id) AS tarefas_atrasadas
FROM
    Projetos p
JOIN
    Tarefas t ON p.id = t.id_projeto
WHERE
    t.status != 'Concluída'
    AND t.data_prevista_entrega < CURDATE()
GROUP BY
    p.id, p.nome
ORDER BY
    tarefas_atrasadas DESC
LIMIT 5;
```

Exercício 6: Tarefas urgentes/altas não concluídas

```
SELECT
    t.titulo,
    t.prioridade,
    t.status,
    p.nome AS nome_projeto,
    r.nome AS responsavel_tarefa
FROM
    Tarefas t
JOIN
    Projetos p ON t.id_projeto = p.id
JOIN
    Responsaveis r ON t.id_responsavel_tarefa = r.id
WHERE
    t.prioridade IN ('Alta', 'Urgente')
    AND t.status != 'Concluída'
ORDER BY
    t.prioridade DESC, p.nome;
```

Exercício 7: Projetos com tarefas pendentes

```
SELECT
   p.nome AS nome_projeto,
   COUNT(t.id) AS tarefas_pendentes
FROM
   Projetos p
JOIN
   Tarefas t ON p.id = t.id_projeto
WHERE
   t.status = 'Pendente'
GROUP BY
   p.id, p.nome
HAVING
   COUNT(t.id) >= 1
ORDER BY
   p.nome;
```

- WHERE → Filtra LINHAS
- HAVING → Filtra GRUPOS

Dicas Essenciais para sua Prova de Banco de Dados

Aqui estão as melhores dicas para você se sair bem na prova, organizadas por tópicos:

1. Dicas sobre Consultas SQL

Consultas Básicas

- SELECT: Sempre comece identificando quais colunas você precisa
- FROM: Certifique-se de incluir todas as tabelas necessárias
- WHERE: Use para filtrar linhas individuais antes do agrupamento

Junções (JOINs)

- INNER JOIN: Retorna apenas registros que existem em ambas as tabelas
- LEFT JOIN: Mantém todos os registros da tabela à esquerda
- Lembre-se: ON especifica a condição de junção

Agregação

- GROUP BY: Agrupa resultados por valores de coluna
- HAVING: Filtra grupos após a agregação (diferente de WHERE)

- Funções úteis: COUNT(), SUM(), AVG(), MAX(), MIN()

2. Dicas sobre o Uso de HAVING vs WHERE

Característica	WHERE	HAVING
Momento do Filtro	Antes da agregação	Depois da agregação
Funções de Agregação	Não pode usar	Pode usar
Velocidade	Mais rápido (filtra primeiro)	Mais lento (filtra depois)
Uso com GROUP BY	Opcional	Requer GROUP BY

Regra prática: Se o filtro envolver COUNT, SUM, AVG etc., use HAVING.

3. Dicas de Desempenho

1. Índices: São essenciais para consultas rápidas
    - Colunas frequentemente usadas em WHERE e JOINs devem ser indexadas
2. EXISTS vs IN: Para subconsultas, EXISTS geralmente é mais eficiente
3. LIMIT: Use para evitar retornar muitos dados desnecessariamente

4. Dicas para a Prova

1. Leia com atenção: Entenda exatamente o que cada questão pede
2. Comece pelo simples: Resolva primeiro as consultas básicas
3. Teste mentalmente: Imagine o resultado antes de escrever
4. Verifique junções: Certifique-se de não esquecer nenhuma condição de JOIN
5. Nomenclatura: Use aliases claros para tabelas (ex: p para Projetos)

5. Padrões Recomendados

1. Nomes de tabelas: No plural (Clientes, Pedidos)
2. Chaves primárias: Sempre definidas
3. Chaves estrangeiras: Sempre com constraints
4. Tipos de dados: Escolha os mais adequados (DATE para datas, etc.)