1. **Todas as obras cadastradas ordenadas por data de publicação:**

SELECT * FROM obras ORDER BY data_publicacao;

Este comando seleciona todos os registros da tabela obras e os organiza pela coluna data_publicacao em ordem crescente (padrão do ORDER BY). Isso responde à necessidade de exibir as obras em sequência cronológica

2. **Contagem das obras no acervo:**

SELECT COUNT(*) AS total_obras FROM obras;

O COUNT(*) conta o número total de registros na tabela obras, ou seja, quantas obras existem no acervo. O alias AS total_obras nomeia o resultado para facilitar a leitura.

3. **Datas de empréstimos e quantidade emprestada por data:**

SELECT data_emprestimo, COUNT(*) AS qtd_emprestimos

FROM emprestimos

GROUP BY data_emprestimo;

O GROUP BY agrupa os registros por data de empréstimo (data_emprestimo). Para cada grupo, o COUNT(*) calcula a quantidade de empréstimos feitos naquela data. Isso atende à pergunta de quantos empréstimos foram feitos por data.

4. **Empréstimos realizados pela recepcionista Alice Meire entre 8h e 9h:**

SELECT * FROM emprestimos

WHERE funcionario_id = (SELECT id FROM funcionarios WHERE nome = 'Alice Meire')

AND HOUR(horario_emprestimo) BETWEEN 8 AND 9;

Aqui, o subquery (SELECT id FROM funcionarios WHERE nome = 'Alice Meire') obtém o id da funcionária. A função HOUR() extrai a hora do campo horario_emprestimo, e a cláusula BETWEEN 8 AND 9 filtra os registros desse intervalo.

5. **Devoluções entre 29/03/2012 e 02/02/2013:**

SELECT * FROM devolucoes

WHERE data_devolucao BETWEEN '2012-03-29' AND '2013-02-02';

A cláusula BETWEEN é usada para filtrar os registros cuja data_devolucao está no intervalo especificado. É uma forma direta e eficiente para esse tipo de consulta.

6. **Reservas com status 'reservado' feitas a partir de 18/08/2011:**

SELECT * FROM reservas

WHERE data_reserva >= '2011-08-18' AND status = 'reservado';

- A condição data_reserva >= '2011-08-18' garante que só sejam consideradas reservas feitas a partir dessa data. Já a condição status = 'reservado' filtra registros com o status desejado.

7. **Devoluções antes de 29/03/2012:**

SELECT * FROM devolucoes

WHERE data_devolucao < '2012-03-29';

O filtro data_devolucao < '2012-03-29' exclui registros com data igual ou posterior a 29/03/2012. Responde exatamente ao que foi pedido.

8. **Obras diferentes de 'O Conde de Monte Cristo' e 'Filhos e Amantes':**

SELECT * FROM obras

WHERE titulo NOT IN ('O Conde de Monte Cristo', 'Filhos e Amantes');

O operador NOT IN exclui registros cujo título está na lista especificada. Assim, retorna apenas as obras diferentes das mencionadas.

9. **Quantidade de obras do gênero 'Ficção':**

SELECT COUNT(*) AS total_ficcao

FROM obras

WHERE genero = 'Ficção';

O filtro WHERE genero = 'Ficção' considera apenas as obras do gênero Ficção. O COUNT(*) contabiliza o total dessas obras.

10. **Livro com maior quantidade em estoque, incluindo editora e autor:**

```
SELECT titulo, editora, autor, MAX(quantidade_estoque) AS max_estoque
FROM obras;
```

A função MAX() retorna o maior valor da coluna quantidade_estoque, indicando o livro com o maior número de exemplares. As colunas titulo, editora e autor complementam os detalhes do livro.

11. **Livro com menor quantidade em estoque e vezes emprestado:**

```
SELECT o.titulo, COUNT(e.id) AS qtd_emprestimos, o.quantidade_estoque
FROM obras o
LEFT JOIN emprestimos e ON o.id = e.obra_id
GROUP BY o.id
ORDER BY o.quantidade_estoque ASC
LIMIT 1;
```

A consulta retorna o livro com o menor estoque (ordenando por quantidade_estoque de forma crescente) e o número de vezes que foi emprestado.

Usa o LEFT JOIN para garantir que todos os livros sejam considerados, mesmo aqueles que não foram emprestados.

O COUNT(e.id) conta quantos empréstimos foram feitos para cada livro. O LIMIT 1 retorna apenas o livro com menor estoque.

12. **Total de empréstimos feitos por cada funcionário ativo:**

```
SELECT f.nome, COUNT(e.id) AS total_emprestimos
FROM funcionarios f
JOIN emprestimos e ON f.id = e.funcionario_id
WHERE f.ativo = 1
GROUP BY f.nome;
```

Conta quantos empréstimos cada funcionário ativo fez, filtrando apenas os funcionários com ativo = 1.

O JOIN une as tabelas funcionarios e emprestimos, e o COUNT(e.id) conta os empréstimos feitos por cada funcionário.

13. **Obras com maior número de publicações agrupadas por gênero:**

```
SELECT genero, COUNT(*) AS total_publicacoes
FROM obras
GROUP BY genero
ORDER BY total_publicacoes DESC;
```

Conta quantas obras foram publicadas por gênero.

O GROUP BY agrupa as obras por genero e o ORDER BY organiza de forma decrescente para mostrar primeiro os gêneros com mais publicações.

14. **Alterar o gênero da obra 'Discurso do Método' para Político:**

```
UPDATE obras
SET genero = 'Político'
WHERE titulo = 'Discurso do Método';
```

Altera o gênero da obra "Discurso do Método" para "Político".

O UPDATE é usado para modificar dados, e o WHERE garante que a alteração ocorra apenas para o título específico.

15. **Alterar o bairro de Alberto Roberto para Perdizes:**

```
UPDATE usuarios
SET bairro = 'Perdizes'
WHERE nome = 'Alberto Roberto';
```

Atualiza o bairro do usuário Alberto Roberto para "Perdizes". O UPDATE modifica a tabela usuarios e o WHERE filtra para alterar apenas o registro do usuário específico.

16. **Alterar o estoque para obras desaparecidas:**

```
UPDATE obras
```

SET quantidade_estoque = quantidade_estoque - 1

WHERE titulo IN ('Filho Nativo', 'Vidas Secas', 'Dom Casmurro');

Reduz o estoque em 1 para as obras "Filho Nativo", "Vidas Secas" e "Dom Casmurro".

A operação é realizada usando o UPDATE, e o WHERE com IN seleciona as obras específicas.

17. **Inserir novos usuários na base:**

INSERT INTO usuarios (id, nome, endereco, cidade, telefone, cep, cpf)

VALUES

(31, 'Alfredo Tenttoni', 'Rua Amazonas 58', 'Pirai', '6549-5421', '02170-251', '294.264.875-32'),

(32, 'Cindy Crall', 'Rua Ipiranga 123', 'Vila Cristal', '5846-6577', '02182-637', '122.147.655-49'),

(33, 'Rubens Pardo', 'Avenida dos Monges 51', 'Campo Grande', '5184-8978', '52412-365', '654.586.472-98'),

(34, 'Carlos Pracidelli', 'Travessa dos Irmãos 48', 'Cotia', '8945-7986', '23124-005', '341.251.651-75'),

(35, 'Ernesto Coimbra', 'Avenida Ampére 414', 'Jardim Elvira', '5844-2654', '05728-368', '193.107.214-35');

Insere múltiplos registros de novos usuários na tabela usuarios. Cada valor dentro do VALUES corresponde aos dados de um novo usuário.

18. **Verificar duplicidade de usuários no banco de dados:**

SELECT nome, COUNT(*) AS qtd_duplicados

FROM usuarios

GROUP BY nome, cpf

HAVING qtd_duplicados > 1;

Identifica usuários com o mesmo nome e CPF, contando quantas vezes o mesmo par aparece.

O HAVING filtra para mostrar apenas os registros que aparecem mais de uma vez (duplicados).

19. **Excluir usuários duplicados:**

```
DELETE FROM usuarios
WHERE id NOT IN (
  SELECT MIN(id)
  FROM usuarios
  GROUP BY nome, cpf
);
```

Exclui registros duplicados de usuários, mantendo apenas o registro com o id mínimo para cada nome e CPF.

O DELETE é usado para remover dados, e a subconsulta encontra o id mínimo de cada grupo de duplicados.

20. **Adicionar o valor individual de cada obra:**

```
ALTER TABLE obras ADD COLUMN valor_livro DECIMAL(10,2);
```

Adiciona uma nova coluna valor_livro à tabela obras para armazenar o preço de cada livro.

O tipo DECIMAL(10,2) é usado para armazenar valores monetários com até 10 dígitos no total e 2 casas decimais.

21. **Alterar o campo Multa_Atraso para VARCHAR(3):**

```
ALTER TABLE emprestimos
MODIFY multa_atraso VARCHAR(3);
```

O comando ALTER TABLE é utilizado para modificar a estrutura da tabela emprestimos.

A coluna multa_atraso tem seu tipo alterado para VARCHAR(3), que permite armazenar valores textuais de até 3 caracteres (como 'Sim' e 'Não').

Isso é feito para armazenar a informação de forma mais legível, ao invés de usar valores numéricos (como 0 ou 1)

22. **Atualizar registros de Multa_Atraso (0 → Não, 1 → Sim):**

```
UPDATE emprestimos
SET multa_atraso = CASE
  WHEN multa_atraso = '0' THEN 'Não'
  WHEN multa_atraso = '1' THEN 'Sim'
END;
```

O UPDATE modifica os registros existentes na tabela emprestimos.

A função CASE é utilizada para alterar os valores da coluna multa_atraso, trocando os números 0 e 1 por 'Não' e 'Sim', respectivamente.

Isso melhora a legibilidade dos dados, substituindo números por palavras compreensíveis

23. **Excluir o campo Valor_Livro:**

```
ALTER TABLE obras DROP COLUMN valor_livro;
```

O comando ALTER TABLE é utilizado para remover a coluna valor_livro da tabela obras.

O DROP COLUMN é responsável por excluir essa coluna, o que pode ser necessário quando ela não é mais necessária ou precisa ser substituída por outro tipo de dado.

24. **Lista de livros entregues com atraso e funcionários responsáveis:**

```
SELECT livros.titulo, funcionarios.nome AS funcionario
FROM emprestimos
JOIN livros ON emprestimos.livro_id = livros.id
JOIN funcionarios ON emprestimos.funcionario_id = funcionarios.id
WHERE emprestimos.multa_atraso = 'Sim';
```

A consulta busca livros que foram devolvidos com atraso, ou seja, com a coluna multa_atraso marcada como 'Sim'.

Usa o JOIN para unir as tabelas emprestimos, livros e funcionarios, obtendo o título do livro e o nome do funcionário responsável pelo empréstimo.

O filtro WHERE é aplicado para pegar apenas os registros de livros com atraso.

25. **Livros devolvidos por autores estrangeiros e valor total:**

```
SELECT obras.titulo, obras.autor, SUM(obras.valor_livro) AS total
FROM obras
JOIN devolucoes ON obras.id = devolucoes.obra_id
WHERE obras.autor_nacionalidade != 'Brasileiro'
GROUP BY obras.id;
```

Essa consulta calcula o valor total das obras devolvidas por autores estrangeiros.

A tabela obras é unida com a tabela devolucoes para identificar quais obras foram devolvidas.

O filtro WHERE obras.autor_nacionalidade != 'Brasileiro' limita o resultado apenas às obras de autores não brasileiros.

A função SUM calcula o valor total de cada obra devolvida, agrupando por obras.id.

26. **Usuários que fizeram empréstimos com entrega em 21/08/2011:**

```
SELECT usuarios.nome
FROM emprestimos
JOIN usuarios ON emprestimos.usuario_id = usuarios.id
WHERE emprestimos.data_entrega = '2011-08-21';
```

Retorna a lista de usuários que fizeram empréstimos com a data de entrega especificada (21/08/2011).

O JOIN é usado para combinar as tabelas emprestimos e usuarios, conectando as informações do usuário com os empréstimos realizados.

A cláusula WHERE garante que apenas empréstimos com data de entrega igual a 21/08/2011 sejam retornados.

27. **Obras publicadas antes de 04/03/2013, quantidade e valor unitário:**

```
SELECT titulo, quantidade_estoque, valor_livro
FROM obras
WHERE data_publicacao < '2013-03-04';
```

Essa consulta seleciona obras que foram publicadas antes de 04/03/2013.

A cláusula WHERE filtra as obras com base na data de publicação.

O resultado retorna o título, a quantidade em estoque e o valor unitário das obras

28. **Funcionários separados por ativos/inativos, cargos e salários:**

```
SELECT nome, ativo, cargo, salario
FROM funcionarios
ORDER BY ativo DESC, cargo;
```

Retorna uma lista de funcionários, incluindo seu status de ativo/inativo, cargo e salário.

O ORDER BY organiza os funcionários primeiro pelos status ativo (em ordem decrescente) e depois por cargo, garantindo que funcionários ativos apareçam primeiro.

29. **Livros com autores, editoras e quantidade em estoque:**

```
SELECT titulo, autor, editora, quantidade_estoque
FROM obras;
```

Seleciona os títulos dos livros, seus autores, editoras e a quantidade de estoque disponível.

O SELECT extrai as colunas diretamente da tabela obras, sem necessidade de joins ou filtros.

30. **Funcionários com idade entre 30 e 40 anos e seus departamentos:**

```
SELECT nome, departamento
FROM funcionarios
```

WHERE TIMESTAMPDIFF(YEAR, data_nascimento, CURDATE()) BETWEEN 30 AND 40;

A consulta busca funcionários com idades entre 30 e 40 anos.

A função TIMESTAMPDIFF calcula a diferença em anos entre a data de nascimento do funcionário e a data atual (CURDATE()), e a cláusula BETWEEN filtra os resultados para idades dentro do intervalo especificado.

31. **Criação de uma visão com nome do livro, autor e valor:**

CREATE VIEW livros_valor AS

SELECT titulo, autor, valor_livro

FROM obras;

A instrução CREATE VIEW cria uma visão chamada livros_valor.

A visão é uma tabela virtual que exibe os dados do título, autor e valor do livro, retirados da tabela obras.

A vantagem de uma visão é que ela permite consultas simplificadas a partir de um conjunto de dados, sem a necessidade de reescrever a consulta toda vez.

32. **Livros com valor maior que R$ 90,00:**

SELECT id, titulo

FROM obras

WHERE valor_livro > 90.00;

A consulta retorna os livros da tabela obras que possuem um valor superior a R$ 90,00.

A cláusula WHERE é usada para filtrar os livros com valor superior a 90. O SELECT escolhe apenas as colunas id e titulo para exibir os resultados

33. **Atualizar salário do auxiliar financeiro com aumento de 12%:**

UPDATE funcionarios

SET salario = salario * 1.12

WHERE cargo = 'Auxiliar Financeiro';

A consulta usa o UPDATE para modificar o salário dos funcionários com o cargo "Auxiliar Financeiro".

O novo salário é calculado multiplicando o salário atual por 1.12 (um aumento de 12%).

A cláusula WHERE restringe a atualização apenas aos funcionários com o cargo de "Auxiliar Financeiro".

34. **Atualizar data de demissão de Alice Meire para o último dia do mês atual:**

```
UPDATE funcionarios

SET data_demissao = LAST_DAY(CURDATE())

WHERE nome = 'Alice Meire';
```

O comando UPDATE altera o registro da funcionária Alice Meire na tabela funcionarios.

A função LAST_DAY(CURDATE()) calcula o último dia do mês atual e atribui esse valor ao campo data_demissao.

A cláusula WHERE filtra para garantir que a atualização ocorra apenas para a funcionária com o nome "Alice Meire".

35. **Obras com a letra "C", por gênero e publicadas entre 2011 e 2013:**

```
SELECT titulo, genero, data_publicacao

FROM obras

WHERE titulo LIKE '%C%' AND data_publicacao BETWEEN '2011-01-01' AND '2013-12-31'

ORDER BY genero;
```

A consulta retorna obras cujo título contém a letra "C" (usando o operador LIKE com o padrão %C%).

O filtro BETWEEN é usado para limitar a busca às obras publicadas entre 2011 e 2013.

A cláusula ORDER BY organiza os resultados por gênero.

36. **Funcionários que não emprestaram nenhum livro:**

```
SELECT id, nome, departamento
FROM funcionarios
WHERE id NOT IN (SELECT DISTINCT funcionario_id FROM emprestimos)
ORDER BY nome;
```

O comando retorna a lista de funcionários que não realizaram nenhum empréstimo de livros.

A subconsulta SELECT DISTINCT funcionario_id FROM emprestimos encontra os IDs dos funcionários que emprestaram livros. O NOT IN exclui esses funcionários da lista de resultados.

A cláusula ORDER BY organiza a lista por nome.

37. **Quantidade de logradouros agrupados por CEP:**

```
SELECT cep, COUNT(endereco) AS qtd_logradouros
FROM usuarios
GROUP BY cep;
```

A consulta agrupa os registros de usuarios pelo código postal (cep) e conta o número de endereços (COUNT(endereco)) associados a cada CEP.

O GROUP BY cep organiza os dados por CEP, e o COUNT retorna a quantidade de endereços para cada um.

38. **Quantidade de endereços agrupados por usuário:**

```
SELECT usuario_id, COUNT(endereco) AS qtd_enderecos
FROM usuarios
GROUP BY usuario_id;
```

Retorna a quantidade de endereços por usuário, agrupando os registros pela coluna usuario_id.

O COUNT(endereco) calcula quantos endereços estão associados a cada usuário, e o GROUP BY usuario_id organiza os resultados por usuário

39. **Obras reservadas em 18/08/2011 às 15:00 com responsável:**

```
SELECT obras.titulo, usuarios.nome AS responsavel
FROM reservas
JOIN obras ON reservas.obra_id = obras.id
JOIN usuarios ON reservas.usuario_id = usuarios.id
WHERE data_reserva = '2011-08-18' AND hora_reserva = '15:00:00';
```

A consulta retorna os títulos das obras reservadas em uma data específica (18/08/2011 às 15:00) e os responsáveis por essas reservas.

Usa o JOIN para combinar as tabelas reservas, obras e usuarios, relacionando as obras reservadas com os usuários que fizeram a reserva.

A cláusula WHERE filtra pela data e hora específicas.

40. **Livros emprestados por Emily Mall e Whitney Cinse com valores:**

```
SELECT usuarios.nome, obras.titulo, obras.valor_livro
FROM emprestimos
JOIN usuarios ON emprestimos.usuario_id = usuarios.id
JOIN obras ON emprestimos.obra_id = obras.id
WHERE usuarios.nome IN ('Emily Mall', 'Whitney Cinse');
```

Retorna os livros emprestados por Emily Mall e Whitney Cinse, junto com os valores dos livros.

A consulta faz o JOIN entre as tabelas emprestimos, usuarios e obras para obter as informações necessárias.

O filtro WHERE seleciona apenas os registros dos usuários mencionados.

41. **Primeira pessoa a reservar, pegar emprestado e devolver um livro:**

```
SELECT usuarios.nome, MIN(reservas.data_reserva) AS primeira_reserva,
    MIN(emprestimos.data_emprestimo) AS primeiro_emprestimo,
    MIN(devolucoes.data_devolucao) AS primeira_devolucao
FROM usuarios
```

LEFT JOIN reservas ON usuarios.id = reservas.usuario_id

LEFT JOIN emprestimos ON usuarios.id = emprestimos.usuario_id

LEFT JOIN devolucoes ON usuarios.id = devolucoes.usuario_id

GROUP BY usuarios.nome

ORDER BY primeira_reserva ASC, primeiro_emprestimo ASC, primeira_devolucao ASC

LIMIT 1;

A consulta busca a primeira pessoa que fez uma reserva, pegou emprestado e devolveu um livro.

Usa LEFT JOIN para garantir que todos os usuários sejam considerados, mesmo que não tenham feito todas as ações (reserva, empréstimo e devolução).

A função MIN é usada para encontrar a primeira data de cada ação.

A cláusula GROUP BY agrupa os resultados por usuário, e o ORDER BY organiza os usuários pela data de reserva, empréstimo e devolução.

O LIMIT 1 retorna apenas o primeiro resultado com base na ordem das datas.

42. **Quantidade de obras por editora:**

SELECT editora, COUNT(*) AS total_obras

FROM obras

GROUP BY editora;

A consulta agrupa as obras por editora e conta quantas obras pertencem a cada editora, usando COUNT(*).

O GROUP BY organiza os resultados por editora, e o COUNT(*) retorna a quantidade total de obras de cada editora.

43. **Livros não devolvidos, dias de atraso, multa total e usuário responsável:**

SELECT usuarios.nome, obras.titulo, DATEDIFF(CURDATE(), emprestimos.data_prevista_devolucao) AS dias_atraso,

DATEDIFF(CURDATE(), emprestimos.data_prevista_devolucao) * 5 AS multa_total

```
FROM emprestimos

JOIN usuarios ON emprestimos.usuario_id = usuarios.id

JOIN obras ON emprestimos.obra_id = obras.id

WHERE emprestimos.data_devolucao IS NULL AND DATEDIFF(CURDATE(),
emprestimos.data_prevista_devolucao) > 0;
```

A consulta retorna os livros que não foram devolvidos, com o número de dias de atraso e a multa total.

A função DATEDIFF calcula a diferença entre a data atual (CURDATE()) e a data prevista de devolução.

A multa é calculada multiplicando o número de dias de atraso por 5 (presumivelmente 5 é o valor da multa diária).

O filtro WHERE garante que apenas os empréstimos não devolvidos e com atraso sejam retornados.

44. **Usuários que moram em avenidas, CPF e logradouro ordenados:**

```
SELECT nome, cpf, endereco

FROM usuarios

WHERE endereco LIKE 'Avenida%'

ORDER BY cpf DESC;
```

A consulta retorna os usuários que moram em avenidas, filtrando com o LIKE 'Avenida%', que seleciona endereços que começam com "Avenida".

A cláusula ORDER BY cpf DESC organiza os resultados de forma decrescente pelo CPF.

45. **Livros emprestados mais de uma vez entre 2011 e 2013, agrupados:**

```
SELECT obras.titulo, COUNT(emprestimos.id) AS total_emprestimos

FROM emprestimos

JOIN obras ON emprestimos.obra_id = obras.id

WHERE emprestimos.data_emprestimo BETWEEN '2011-01-01' AND '2013-12-31'

GROUP BY obras.titulo
```

HAVING total_emprestimos > 1;

A consulta retorna os livros que foram emprestados mais de uma vez entre 2011 e 2013.

O JOIN entre emprestimos e obras permite associar os empréstimos aos livros.

A cláusula HAVING filtra os resultados para mostrar apenas os livros com mais de um empréstimo durante o período.

46. **Valor médio dos livros e os abaixo da média:**

SELECT titulo, valor_livro

FROM obras

WHERE valor_livro < (SELECT AVG(valor_livro) FROM obras);

A consulta retorna os livros cujo valor é inferior à média dos valores de todos os livros.

A subconsulta (SELECT AVG(valor_livro) FROM obras) calcula a média dos valores dos livros.

O WHERE filtra os livros que têm um valor abaixo dessa média.

47. **Média salarial e funcionários que ganham acima da média por departamento:**

SELECT departamento, nome, salario

FROM funcionarios

WHERE salario > (SELECT AVG(salario) FROM funcionarios WHERE departamento = funcionarios.departamento);

A consulta retorna os funcionários que ganham mais do que a média salarial do seu departamento.

A subconsulta calcula a média salarial dentro de cada departamento, e a cláusula WHERE filtra os funcionários com salários acima dessa média.

48. **Usuários com cadastro na biblioteca que nunca levaram livros (em maiúsculas):**

SELECT UPPER(nome) AS nome

FROM usuarios

WHERE id NOT IN (SELECT DISTINCT usuario_id FROM emprestimos);

A consulta retorna os usuários que nunca pegaram livros emprestados, usando a subconsulta SELECT DISTINCT usuario_id FROM emprestimos para identificar aqueles que já realizaram empréstimos.

A função UPPER transforma os nomes dos usuários para maiúsculas antes de exibir.

49. **Usuários que pegaram mais de 3 livros, ordenados pelo CEP:**

SELECT usuarios.nome, obras.titulo, usuarios.cep

FROM emprestimos

JOIN usuarios ON emprestimos.usuario_id = usuarios.id

JOIN obras ON emprestimos.obra_id = obras.id

GROUP BY usuarios.id, obras.id

HAVING COUNT(emprestimos.id) > 3

ORDER BY usuarios.cep ASC;

A consulta retorna os usuários que pegaram mais de 3 livros emprestados, juntamente com os títulos dos livros e seus CEPs.

A cláusula HAVING COUNT(emprestimos.id) > 3 filtra os usuários que fizeram mais de 3 empréstimos.

O ORDER BY organiza os resultados pelo CEP de forma ascendente.

50. **Análise de estoque de livros reservados/emprestados e disponíveis por gênero:**

SELECT genero, SUM(quantidade_estoque) AS disponiveis,

SUM(quantidade_estoque - quantidade_reservada) AS emprestados

FROM obras

GROUP BY genero;

A consulta calcula a quantidade de livros disponíveis e emprestados por gênero.

A função SUM(quantidade_estoque) retorna o total de livros disponíveis em estoque para cada gênero.

A expressão quantidade_estoque - quantidade_reservada calcula quantos livros foram emprestados (livros que estão no estoque, mas já foram reservados).

O GROUP BY genero agrupa os dados por gênero de livro, mostrando as somas para cada um.

51. **Horário de maior e menor movimento:**

```
SELECT HOUR(data_hora) AS hora, COUNT(*) AS total_movimentacoes
FROM (
  SELECT data_hora FROM emprestimos
  UNION ALL
  SELECT data_hora FROM devolucoes
  UNION ALL
  SELECT data_hora FROM reservas
) AS movimentacoes
GROUP BY HOUR(data_hora)
ORDER BY total_movimentacoes DESC;
```

A consulta analisa o horário de maior e menor movimento de transações no sistema (empréstimos, devoluções e reservas).

O UNION ALL junta as três tabelas de movimentações (empréstimos, devoluções e reservas), sem remover duplicatas.

A função HOUR(data_hora) extrai a hora da data de cada transação, e COUNT(*) conta quantas transações ocorreram por hora.

O GROUP BY HOUR(data_hora) agrupa as movimentações por hora, e o ORDER BY total_movimentacoes DESC ordena por quantidade de movimentações em ordem decrescente.

52. **3 autores mais lidos em 2012/2013 e 2 menos lidos:**

```
SELECT autor, COUNT(emprestimos.id) AS total_lidos
FROM emprestimos
```

```
JOIN obras ON emprestimos.obra_id = obras.id

WHERE emprestimos.data_emprestimo BETWEEN '2012-01-01' AND '2013-12-31'

GROUP BY autor

ORDER BY total_lidos DESC

LIMIT 3;
```

A consulta retorna os 3 autores mais lidos entre 2012 e 2013.

O JOIN entre emprestimos e obras associa os empréstimos aos seus respectivos livros.

A cláusula WHERE filtra os empréstimos realizados no período de 2012 a 2013.

O GROUP BY autor agrupa os resultados por autor, e COUNT(emprestimos.id) conta o número de livros emprestados de cada autor.

O ORDER BY total_lidos DESC ordena os autores pela quantidade de livros emprestados e o LIMIT 3 retorna apenas os três mais lidos.

53. **Tabela de livros por funcionário, total de empréstimos e devoluções:**

```
SELECT funcionarios.nome, COUNT(emprestimos.id) AS total_emprestimos,

    COUNT(devolucoes.id) AS total_devolucoes

FROM funcionarios

LEFT JOIN emprestimos ON funcionarios.id = emprestimos.funcionario_id

LEFT JOIN devolucoes ON funcionarios.id = devolucoes.funcionario_id

GROUP BY funcionarios.nome;
```

A consulta retorna o total de empréstimos e devoluções realizados por cada funcionário.

O LEFT JOIN entre funcionarios, emprestimos e devolucoes garante que todos os funcionários sejam listados, mesmo que não tenham realizado empréstimos ou devoluções.

O COUNT(emprestimos.id) conta quantos empréstimos cada funcionário fez, e o COUNT(devolucoes.id) conta as devoluções.

O GROUP BY funcionarios.nome agrupa os resultados por funcionário

54. **Usuários que possuem mesmo endereço que editoras:**

```
SELECT usuarios.nome, usuarios.telefone, editoras.nome AS editora
FROM usuarios
JOIN editoras ON usuarios.endereco = editoras.endereco;
```

A consulta retorna os usuários que têm o mesmo endereço que as editoras.

O JOIN entre usuarios e editoras usa o campo endereco para encontrar as correspondências entre usuários e editoras que moram no mesmo local.

O resultado inclui o nome do usuário, o telefone e o nome da editora.

55. **Visão com livros e preços da editora Leya:**

```
CREATE VIEW livros_leya AS
SELECT titulo, valor_livro
FROM obras
WHERE editora = 'Leya';
```

A consulta cria uma **view** (uma tabela virtual) chamada livros_leya, que exibe o título e o valor do livro apenas para os livros da editora Leya.

A cláusula WHERE filtra os livros da editora Leya.

56. **Atualizar preços de livros da editora Saraiva com aumento de 16%:**

```
UPDATE obras
SET valor_livro = valor_livro * 1.16
WHERE editora = 'Saraiva';
```

A consulta atualiza o preço dos livros da editora Saraiva, aplicando um aumento de 16%.

O SET valor_livro = valor_livro * 1.16 multiplica o valor do livro atual por 1,16 para aplicar o aumento de 16%.

A cláusula WHERE garante que apenas os livros da editora Saraiva sejam atualizados

57. **5 obras com menos e mais publicações, com detalhes:**

```
SELECT titulo, autor, editora, quantidade_publicacoes

FROM obras

ORDER BY quantidade_publicacoes ASC

LIMIT 5

UNION ALL

SELECT titulo, autor, editora, quantidade_publicacoes

FROM obras

ORDER BY quantidade_publicacoes DESC

LIMIT 5;
```

A consulta retorna as 5 obras com o menor número de publicações e as 5 com o maior número de publicações.

O primeiro SELECT ordena os livros pela quantidade de publicações em ordem crescente (ASC) e limita a 5 resultados.

O segundo SELECT faz a mesma coisa, mas ordena em ordem decrescente (DESC).

O UNION ALL combina os dois resultados em um único conjunto

58. **Usuários com CPF iniciando por 193, CPF mascarado:**

```
SELECT nome, CONCAT('193.***.***-', SUBSTRING(cpf, -2)) AS cpf_mascarado

FROM usuarios

WHERE cpf LIKE '193%';
```

A consulta retorna o nome e o CPF mascarado dos usuários cujos CPFs começam com '193'.

O LIKE '193%' filtra os CPFs que começam com '193'.

A função CONCAT combina uma string de formato '193..-' com os dois últimos dígitos do CPF, que são extraídos usando a função SUBSTRING(cpf, -2).