LISTA DE EXERCÍCIOS SQL - DDL E DML NO MYSQL WORKBENCH

PARTE 1: COMANDOS DDL (Data Definition Language)

Exercício 1: CREATE DATABASE e CREATE TABLE

Crie um banco de dados chamado sistema_escolar e implemente as seguintes tabelas:

```
CREATE DATABASE sistema_escolar;
USE sistema_escolar;

CREATE TABLE alunos (
  matricula INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100) NOT NULL,
  sobrenome VARCHAR(100) NOT NULL,
  data_nascimento DATE,
  cpf VARCHAR(14) UNIQUE,
  email VARCHAR(100),
  ativo BOOLEAN DEFAULT TRUE
);

CREATE TABLE cursos (
  id_curso INT AUTO_INCREMENT PRIMARY KEY,
  nome_curso VARCHAR(100) NOT NULL,
  carga_horaria INT,
  valor DECIMAL(10,2),
  nivel VARCHAR(50)
);

CREATE TABLE matriculas (
  id_matricula INT AUTO_INCREMENT PRIMARY KEY,
  aluno_matricula INT,
  curso_id INT,
  data_matricula DATE,
  status VARCHAR(20)
);
```

Exercício 2: ALTER TABLE - Adicionando Colunas

Altere as tabelas criadas adicionando novas colunas:

```
ALTER TABLE alunos ADD COLUMN telefone VARCHAR(20);
ALTER TABLE alunos ADD COLUMN endereco VARCHAR(200);
ALTER TABLE cursos ADD COLUMN professor_responsavel VARCHAR(100);
ALTER TABLE cursos ADD COLUMN turno VARCHAR(20);
```

Exercício 3: ALTER TABLE - Modificando Colunas

Modifique a estrutura das colunas existentes:

```
ALTER TABLE alunos MODIFY COLUMN nome VARCHAR(150) NOT NULL;
ALTER TABLE cursos MODIFY COLUMN valor DECIMAL(12,2);
ALTER TABLE alunos CHANGE COLUMN telefone telefone_celular VARCHAR(20);
```

Exercício 4: ALTER TABLE - Chaves Estrangeiras

Adicione constraints de chave estrangeira nas tabelas:

```
ALTER TABLE matriculas
ADD CONSTRAINT fk_aluno
FOREIGN KEY (aluno_matricula) REFERENCES alunos(matricula);

ALTER TABLE matriculas
ADD CONSTRAINT fk_curso
FOREIGN KEY (curso_id) REFERENCES cursos(id_curso);
```

Exercício 5: ALTER TABLE - Outros Constraints

Adicione diferentes tipos de constraints:

```
ALTER TABLE alunos ADD CONSTRAINT chk_email CHECK (email LIKE '%@%');
ALTER TABLE cursos ADD CONSTRAINT chk_carga CHECK (carga_horaria > 0);
ALTER TABLE cursos ADD CONSTRAINT unique_nome UNIQUE (nome_curso);
```

Exercício 6: DROP e TRUNCATE

Pratique comandos de remoção (tenha cuidado!):

```
-- Criar uma tabela temporária para teste
CREATE TABLE tabela_teste (
   id INT PRIMARY KEY,
   descricao VARCHAR(50)
);

-- Remover estrutura completa
DROP TABLE IF EXISTS tabela_teste;

-- Criar novamente e limpar dados mantendo estrutura
CREATE TABLE tabela_teste (id INT, descricao VARCHAR(50));
INSERT INTO tabela_teste VALUES (1, 'teste');
TRUNCATE TABLE tabela_teste;
```

PARTE 2: COMANDOS DML (Data Manipulation Language)

Exercício 7: INSERT - Inserção Simples

Insira múltiplos registros nas tabelas criadas:

```
INSERT INTO alunos (nome, sobrenome, data_nascimento, cpf, email, ativo)
VALUES ('João', 'Silva', '2000-05-15', '111.222.333-44', 'joao@email.com', TRUE);


INSERT INTO alunos VALUES
(NULL, 'Maria', 'Santos', '1999-08-20', '222.333.444-55', 'maria@email.com', TRUE,
'11987654321', 'Rua A, 123');

INSERT INTO cursos (nome_curso, carga_horaria, valor, nivel) VALUES
('Python Básico', 40, 500.00, 'Iniciante'),
('Banco de Dados', 60, 800.00, 'Intermediário'),
('Java Avançado', 80, 1200.00, 'Avançado'),
('Web Design', 50, 600.00, 'Iniciante');
```

Exercício 8: INSERT - Múltiplas Inserções

Insira pelo menos 10 alunos e 10 matrículas diferentes, ***use a criatividade e complete os 5 alunos faltantes***:

```
INSERT INTO alunos (nome, sobrenome, data_nascimento, cpf, email) VALUES
('Carlos', 'Oliveira', '2001-03-10', '333.444.555-66', 'carlos@email.com'),
('Ana', 'Costa', '1998-12-05', '444.555.666-77', 'ana@email.com'),
('Pedro', 'Almeida', '2002-07-22', '555.666.777-88', 'pedro@email.com'),
('Julia', 'Ferreira', '2000-11-30', '666.777.888-99', 'julia@email.com'),
('Lucas', 'Souza', '1999-04-18', '777.888.999-00', 'lucas@email.com');

INSERT INTO matriculas (aluno_matricula, curso_id, data_matricula, status)
VALUES
(1, 1, '2025-01-10', 'Ativo'),
(2, 2, '2025-01-15', 'Ativo'),
(3, 1, '2025-02-01', 'Ativo'),
(4, 3, '2025-02-05', 'Ativo'),
(5, 4, '2025-02-10', 'Ativo');
```

Exercício 9: SELECT - Consultas Básicas

Realize consultas simples e com filtros:

```
-- Selecionar todos os alunos
SELECT * FROM alunos;
```

```
-- Selecionar apenas nome e email dos alunos ativos
SELECT nome, sobrenome, email FROM alunos WHERE ativo = TRUE;

-- Selecionar cursos com carga horária maior que 50 horas
SELECT nome_curso, carga_horaria, valor FROM cursos WHERE carga_horaria > 50;

-- Selecionar alunos nascidos após o ano 2000
SELECT nome, sobrenome, data_nascimento FROM alunos WHERE data_nascimento > '2000-01-01';
```

Exercício 10: SELECT - Operadores e LIKE

Pratique diferentes operadores relacionais:

```
-- Cursos com valor entre 500 e 900 reais
SELECT * FROM cursos WHERE valor BETWEEN 500 AND 900;

-- Alunos cujo nome começa com 'A'
SELECT * FROM alunos WHERE nome LIKE 'A%';

-- Alunos cujo sobrenome termina com 'Silva'
SELECT * FROM alunos WHERE sobrenome LIKE '%Silva';

-- Cursos de nível Iniciante ou Intermediário
SELECT * FROM cursos WHERE nivel IN ('Iniciante', 'Intermediário');

-- Alunos sem telefone cadastrado
SELECT * FROM alunos WHERE telefone_celular IS NULL;
```

Exercício 11: SELECT - Ordenação e Limitação

Use ORDER BY, LIMIT e DISTINCT:

```
-- Listar alunos em ordem alfabética
SELECT * FROM alunos ORDER BY nome ASC, sobrenome ASC;

-- Listar cursos do mais caro para o mais barato
SELECT nome_curso, valor FROM cursos ORDER BY valor DESC;

-- Os 3 cursos mais caros
SELECT nome_curso, valor FROM cursos ORDER BY valor DESC LIMIT 3;
```

```
-- Listar níveis de curso disponíveis (sem repetição)
SELECT DISTINCT nivel FROM cursos;
```

Exercício 12: SELECT - Funções de Agregação

Pratique COUNT, SUM, AVG, MAX, MIN:

```
-- Contar total de alunos cadastrados
SELECT COUNT(*) AS total_alunos FROM alunos;

-- Contar alunos ativos
SELECT COUNT(*) AS alunos_ativos FROM alunos WHERE ativo = TRUE;

-- Calcular valor médio dos cursos
SELECT AVG(valor) AS media_preco FROM cursos;


-- Encontrar maior e menor carga horária
SELECT MAX(carga_horaria) AS maior_carga, MIN(carga_horaria) AS menor_carga
FROM cursos;

-- Somar carga horária total de todos os cursos
SELECT SUM(carga_horaria) AS carga_total FROM cursos;
```

Exercício 13: SELECT - GROUP BY e HAVING

Agrupe dados e filtre grupos:

```
-- Contar quantos cursos existem por nível
SELECT nivel, COUNT(*) AS quantidade FROM cursos GROUP BY nivel;

-- Valor médio dos cursos por nível
SELECT nivel, AVG(valor) AS valor_medio FROM cursos GROUP BY nivel;

-- Contar matrículas por status
SELECT status, COUNT(*) AS total FROM matriculas GROUP BY status;

-- Mostrar apenas níveis com mais de 1 curso
SELECT nivel, COUNT(*) AS quantidade
FROM cursos
GROUP BY nivel
```

```
HAVING COUNT(*) > 1;
```

Exercício 14: UPDATE - Atualizando Registros

Pratique alterações em dados existentes:

```
-- Atualizar telefone de um aluno específico
UPDATE alunos SET telefone_celular = '11999887766' WHERE matricula = 1;

-- Aumentar em 10% o valor de todos os cursos de nível Iniciante
UPDATE cursos SET valor = valor * 1.10 WHERE nivel = 'Iniciante';

-- Atualizar múltiplos campos
UPDATE alunos
SET endereco = 'Rua Nova, 456', email = 'novoemail@email.com'
WHERE matricula = 2;

-- Desativar alunos sem email cadastrado
UPDATE alunos SET ativo = FALSE WHERE email IS NULL;
```

Exercício 15: UPDATE com Subconsultas

Atualizações mais complexas:

```
-- Atualizar status das matrículas do curso mais caro
UPDATE matriculas
SET status = 'Prioridade'
WHERE curso_id = (SELECT id_curso FROM cursos ORDER BY valor DESC LIMIT 1);
```

Exercício 16: DELETE - Removendo Registros

Pratique exclusão de dados com cuidado:

```
-- Deletar alunos inativos
DELETE FROM alunos WHERE ativo = FALSE;

-- Deletar cursos com valor acima de 2000
DELETE FROM cursos WHERE valor > 2000;

-- Deletar matrículas com status 'Cancelado'
DELETE FROM matriculas WHERE status = 'Cancelado';
```

PARTE 3: EXERCÍCIOS COMBINADOS DDL + DML

Exercício 17: Projeto Completo - Sistema de Biblioteca

Crie do zero um sistema de biblioteca:

```
-- DDL: Criar estrutura
CREATE DATABASE biblioteca;
USE biblioteca;

CREATE TABLE autores (
  id_autor INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100) NOT NULL,
  nacionalidade VARCHAR(50)
);

CREATE TABLE livros (
  id_livro INT AUTO_INCREMENT PRIMARY KEY,
  titulo VARCHAR(200) NOT NULL,
  isbn VARCHAR(20) UNIQUE,
  ano_publicacao YEAR,
  categoria VARCHAR(50),
  autor_id INT,
  FOREIGN KEY (autor_id) REFERENCES autores(id_autor)
);

CREATE TABLE usuarios (
  id_usuario INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100) NOT NULL,
  cpf VARCHAR(14) UNIQUE,
  telefone VARCHAR(20),
  data_cadastro DATE DEFAULT (CURRENT_DATE)
);

CREATE TABLE emprestimos (
  id_emprestimo INT AUTO_INCREMENT PRIMARY KEY,
  livro_id INT,
  usuario_id INT,
  data_emprestimo DATE,
  data_devolucao_prevista DATE,
  data_devolucao_real DATE,
  status VARCHAR(20),
  FOREIGN KEY (livro_id) REFERENCES livros(id_livro),
  FOREIGN KEY (usuario_id) REFERENCES usuarios(id_usuario)
);

-- DML: Popular tabelas
INSERT INTO autores (nome, nacionalidade) VALUES
```

```
('Machado de Assis', 'Brasileira'),
('Clarice Lispector', 'Brasileira'),
('George Orwell', 'Britânica');

INSERT INTO livros (titulo, isbn, ano_publicacao, categoria, autor_id) VALUES
('Dom Casmurro', '978-8535908863', 1899, 'Romance', 1),
('A Hora da Estrela', '978-8520923948', 1977, 'Romance', 2),
('1984', '978-0451524935', 1949, 'Ficção', 3);

INSERT INTO usuarios (nome, cpf, telefone) VALUES
('José Silva', '123.456.789-00', '11987654321'),
('Maria Oliveira', '234.567.890-11', '11876543210');

INSERT INTO emprestimos (livro_id, usuario_id, data_emprestimo,
data_devolucao_prevista, status) VALUES
(1, 1, '2025-10-01', '2025-10-15', 'Em andamento'),
(2, 2, '2025-10-05', '2025-10-19', 'Em andamento');
```

Exercício 18: Consultas Complexas no Sistema de Biblioteca

Execute consultas avançadas:

```
-- Listar todos os empréstimos ativos
SELECT u.nome AS usuario, l.titulo AS livro, e.data_emprestimo,
e.data_devolucao_prevista
FROM emprestimos e
JOIN usuarios u ON e.usuario_id = u.id_usuario
JOIN livros l ON e.livro_id = l.id_livro
WHERE e.status = 'Em andamento';

-- Contar quantos livros cada autor tem cadastrado
SELECT a.nome, COUNT(l.id_livro) AS total_livros
FROM autores a
LEFT JOIN livros l ON a.id_autor = l.autor_id
GROUP BY a.nome;


-- Identificar empréstimos atrasados (data prevista < hoje)
SELECT u.nome, l.titulo, e.data_devolucao_prevista
FROM emprestimos e
JOIN usuarios u ON e.usuario_id = u.id_usuario
JOIN livros l ON e.livro_id = l.id_livro
WHERE e.data_devolucao_prevista < CURDATE() AND e.data_devolucao_real IS
NULL;
```