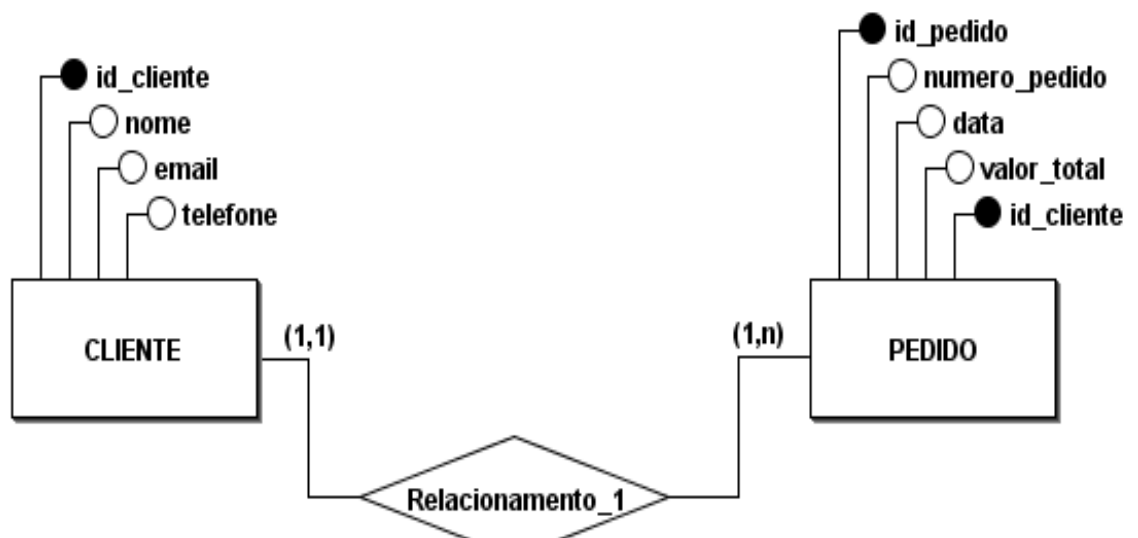


PLANO DE SITUAÇÃO DE APRENDIZAGEM

Modelagem das classes Cliente e Pedido.

A relação entre elas é de um-para-muitos (1:N), onde um cliente pode ter vários pedidos.



Criação do schema e tabelas no MySQL.

Script SQL criado, testado e funcionando.

```
1  -- Cria o banco de dados se ele não existir
2  • CREATE DATABASE IF NOT EXISTS loja_dart;
3  -- Seleciona o banco de dados para uso
4  • USE loja_dart;
5  -- Tabela de Clientes
6  • CREATE TABLE IF NOT EXISTS clientes (
7      id INT AUTO_INCREMENT PRIMARY KEY,
8      nome VARCHAR(255) NOT NULL,
9      email VARCHAR(255) UNIQUE NOT NULL,
10     telefone VARCHAR(20)
11 );
12 -- Tabela de Pedidos
13 • CREATE TABLE IF NOT EXISTS pedidos (
14     id INT AUTO_INCREMENT PRIMARY KEY,
15     data DATE NOT NULL,
16     valor_total DECIMAL(10, 2) NOT NULL,
17     id_cliente INT,
18     FOREIGN KEY (id_cliente) REFERENCES clientes(id)
19 );
```

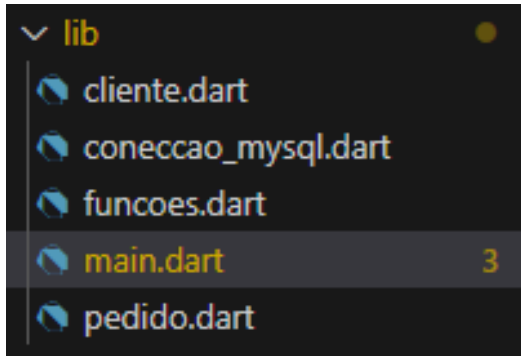
Output

Action Output

#	Time	Action	Message
✓ 1	18:23:32	create database loja_dart	1 row(s) affected
✓ 2	18:23:55	use loja_dart	0 row(s) affected
✓ 3	18:26:32	CREATE TABLE IF NOT EXISTS clientes (id INT AUTO_INCREMENT PRIMARY KEY, ...	0 row(s) affected
✓ 4	18:26:32	CREATE TABLE IF NOT EXISTS pedidos (id INT AUTO_INCREMENT PRIMARY KEY, ...	0 row(s) affected

ESTRUTURA DO PROJETO:

Para facilitar a execucao do exercicio , criei pastas para agrupar as funcoes, conneccao com base de dados e as classes.



Implementação da conexão em Dart e funções de INSERT.

Coneccao Dart > Mysql

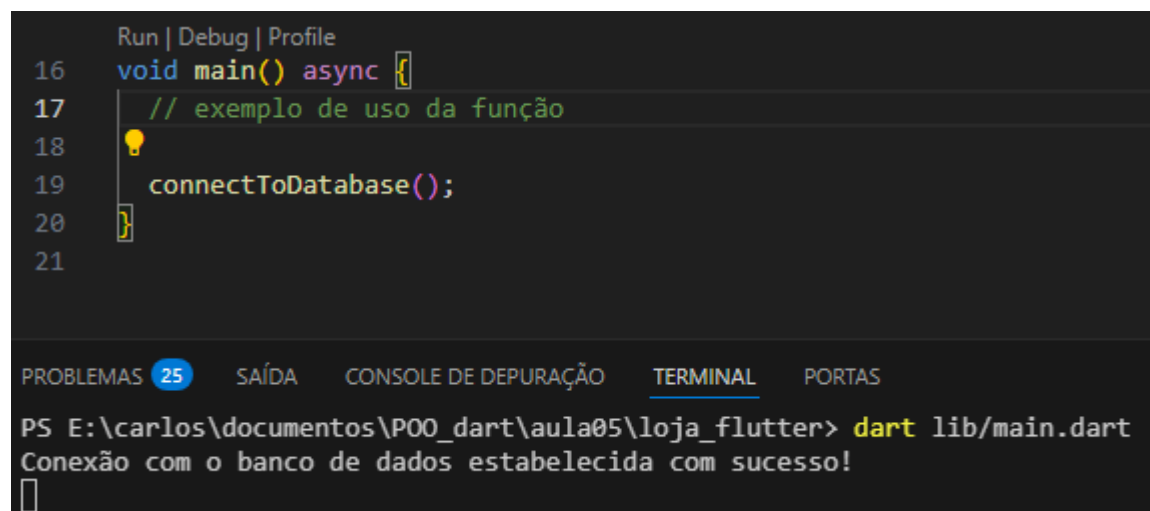
```
// Importa o pacote mysql_client para conexão com MySQL
import 'package:mysql_client/mysql_client.dart';

// Configuração de conexão com o banco de dados
const String _dbHost = '127.0.0.1'; // localhost
const int _dbPort = 3306;
const String _dbUser = 'carlos';
const String _dbPassword = 'Ktmsx-350f';
const String _dbDatabase = 'loja_dart';

/// Função para conectar ao banco de dados MySQL
Future<MySQLConnection?> connectToDatabase() async {
  try {
    final conn = await MySQLConnection.createConnection(
      host: _dbHost,
      port: _dbPort,
      userName: _dbUser,
      databaseName: _dbDatabase,
      password: _dbPassword,
      //secure: false, // <- Desativa SSL
    );
  }
}
```

```
    await conn.connect();  
    print( 'Conexão com o banco de dados estabelecida com sucesso!');  
    return conn;  
  } catch (erro) {  
    print('Erro ao conectar ao banco de dados: $erro');  
    return null;  
  }  
}
```

Resultado da funcao connectToDatabase no terminal:



The screenshot shows an IDE with a Dart file. The code defines a `main` function that calls `connectToDatabase`. The terminal at the bottom shows the command `dart lib/main.dart` being executed, resulting in the output `Conexão com o banco de dados estabelecida com sucesso!`.

```
Run | Debug | Profile  
16 void main() async {  
17   // exemplo de uso da função  
18     
19   connectToDatabase();  
20 }  
21  
  
PROBLEMAS 25 SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS  
PS E:\carlos\documentos\P00_dart\aula05\loja_flutter> dart lib/main.dart  
Conexão com o banco de dados estabelecida com sucesso!  
█
```

INSERT Clientes

Codigo funcao:

```
// Função principal que insere um cliente
Future<void> inserirCliente(Cliente cliente) async {
  final conn = await connectToDatabase();

  if (conn == null) {
    print('Não foi possível inserir o cliente. Conexão falhou.');
```

```
    return;
  }

  try {
    await conn.execute(
      'INSERT INTO clientes (nome, email, telefone) VALUES (:nome,
:email, :telefone)',
      {
        'nome': cliente.nome,
        'email': cliente.email,
        'telefone': cliente.telefone,
      },
    );
    print('Cliente ${cliente.nome} inserido com sucesso!');
```

```
  } catch (e) {
    print('Erro ao inserir cliente: $e');
  } finally {
    // A conexão é fechada aqui, independentemente do resultado
    await conn.close();
    print('Conexão com o banco de dados fechada.');
```

```
  }
}
```

Saida terminal resultado inserirCliente:

```
Run | Debug | Profile
62 void main() async {
63   // Exemplo de uso da função
64   final novoCliente = Cliente(nome: 'Joana da Silva', email: 'joana@teste.com', telefone: '11999998888');
65   await inserirCliente(novoCliente);
66 }

PROBLEMAS 6 SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

PS E:\carlos\documentos\P00_dart\aula05\loja_flutter> dart lib/main.dart
Conexão com o banco de dados estabelecida com sucesso!
Cliente Joana da Silva inserido com sucesso!
Conexão com o banco de dados fechada.
PS E:\carlos\documentos\P00_dart\aula05\loja_flutter> █
```

Inserir Pedido:

Codigo da funcao:

```
// Função principal que insere um pedido
Future<void> inserirPedido(Pedido pedido) async {
  final conn = await connectToDatabase();

  if (conn == null) {
    print('Não foi possível inserir o pedido. Conexão falhou.');
```

```
    return;
  }

  try {
    await conn.execute(
      'INSERT INTO pedidos (data, valor_total, id_cliente) VALUES
(:data, :valor_total, :id_cliente)',
      {
        'data': pedido.data,
        'valor_total': pedido.valorTotal,
        'id_cliente': pedido.idCliente,
      },
    );
  }
```

```

        print('Pedido para o cliente com ID ${pedido.idCliente} inserido
com sucesso!');
    } catch (e) {
        print('Erro ao inserir pedido: $e');
    } finally {
        await conn.close();
        print('Conexão com o banco de dados encerrada.');
    }
}

```

Saida resultado inserirPedido:

```

Run | Debug | Profile
62 void main() async {
63     // Exemplo de uso da função inserirPedido
64     Pedido pedido = Pedido(
65         data: DateTime.now(),
66         valorTotal: 150.0,
67         idCliente: 1,
68     ); // Pedido
69
70     await inserirPedido(pedido);
71 }

```

PROBLEMAS 7 SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

```

• PS E:\carlos\documentos\P00_dart\aula05\loja_flutter> dart lib/main.dart
Conexão com o banco de dados estabelecida com sucesso!
Pedido para o cliente com ID 1 inserido com sucesso!
Conexão com o banco de dados encerrada.
❖ PS E:\carlos\documentos\P00_dart\aula05\loja_flutter> 

```

SELECT + ORDER BY

Funcao listarClientesOrdenadosPorNome, codigo:

```
//listar clientes ordenados por nome
Future<void> listarClientesOrdenadosPorNome() async {
  final conn = await connectToDatabase();

  if (conn == null) {
    print('Não foi possível listar os clientes. Conexão falhou.');
```

```
    return;
  }

  try {
    final results = await conn.execute(
      'SELECT id, nome, email, telefone FROM clientes ORDER BY nome',
    );

    print('\n--- Lista de Clientes ---');
    for (var row in results.rows) {
      final id = row.assoc()['id'];
      final nome = row.assoc()['nome'];
      final email = row.assoc()['email'];
      final telefone = row.assoc()['telefone'];

      print('ID: $id, Nome: $nome, Email: $email, Telefone: $telefone');
```

```
    }

  } catch (e) {
    print('Erro ao listar clientes: $e');
  } finally {
    await conn.close();
    print('Conexão com o banco de dados encerrada.');
```

```
  }
}
```


Saida terminal resultado da funcao:

```
Run | Debug | Profile
22 void main() async {
23     // Função lista clientes ordenados por nome
24     listarClientesOrdenadosPorNome();
25 }
26
```

PROBLEMAS 30 SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

PS E:\carlos\documentos\P00_dart\aula05\loja_flutter> dart lib/main.dart
Conexão com o banco de dados estabelecida com sucesso!

• --- Lista de Clientes ---
ID: 4, Nome: Anna da Silva, Email: annaa@exemplo.com, Telefone: 11999996666
ID: 2, Nome: Joana da Silva, Email: joana@teste.com, Telefone: 11999998888
ID: 1, Nome: João da Silva, Email: joao@teste.com, Telefone: 11999998888
ID: 3, Nome: Maria Oliveira, Email: maria@exemplo.com, Telefone: 11999997777
Conexão com o banco de dados encerrada.
❖ PS E:\carlos\documentos\P00_dart\aula05\loja_flutter> █

SELECT + JOIN+ ORDER BY

Funcao listarPedidosComClientes, utiliza 'join' para apresentar o nome do cliente na lista de pedidos ordenado por data decendente.

```
/// Lista todos os pedidos, mostrando o nome do cliente ao invés do ID.
Future<void> listarPedidosComClientes() async {
    final conn = await connectToDatabase();

    if (conn == null) {
        print('Não foi possível listar os pedidos. Conexão falhou.');
```

```

try {
    // A consulta usa JOIN para combinar as informações das duas
tabelas
    final results = await conn.execute(
        'SELECT p.id, p.data, p.valor_total, c.nome '
        'FROM pedidos p '
        'JOIN clientes c ON p.id_cliente = c.id '
        'ORDER BY p.data DESC',
    );

    // Itera sobre os resultados e exibe de forma formatada
    for (var row in results.rows) {
        final pedidoId = row.assoc()['id'];
        final data = row.assoc()['data'].toString().substring(0, 10);
        final valorTotal =
double.parse(row.assoc()['valor_total'].toString()).toStringAsFixed(2);
        final clienteNome = row.assoc()['nome'];

        print('Id: $pedidoId, Data: $data, Valor Total: R\$ $valorTotal,
Cliente: $clienteNome'); // Exibe os detalhes do pedido
    }

} catch (e) {
    print('Erro ao consultar pedidos: $e');
} finally {
    await conn.close();
    print('Conexão com o banco de dados encerrada.');
```

Resultado funcao listarPedidosComClientes no terminal

```
Run | Debug | Profile
18 void main() async {
19     // Função principal que lista pedidos com clientes
20     listarPedidosComClientes();
21 }
22
```

PROBLEMAS 25 SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

PS E:\carlos\documentos\P00_dart\aula05\loja_flutter> dart lib/main.dart
Conexão com o banco de dados estabelecida com sucesso!
Id: 1, Data: 2025-08-16, Valor Total: R\$ 150.00, Cliente: João da Silva
Id: 2, Data: 2025-08-16, Valor Total: R\$ 180.00, Cliente: Joana da Silva
Id: 3, Data: 2025-08-15, Valor Total: R\$ 110.00, Cliente: João da Silva
Id: 4, Data: 2025-08-14, Valor Total: R\$ 170.00, Cliente: João da Silva
Id: 6, Data: 2025-08-14, Valor Total: R\$ 70.00, Cliente: Joana da Silva
Id: 5, Data: 2025-08-13, Valor Total: R\$ 170.00, Cliente: Joana da Silva
Conexão com o banco de dados encerrada.
PS E:\carlos\documentos\P00_dart\aula05\loja_flutter> █

SELECT + COUNT + SUM + LEFT JOIN + GROUP BY

Codigo da funcao resumoPedidoPorCliente com total gasto:

```
/// Resumo de pedidos por cliente, mostrando o total gasto e a
quantidade de pedidos.
Future<void> resumoPedidosPorCliente() async {
    final conn = await connectToDatabase();

    if (conn == null) {
        print('Não foi possível gerar o resumo. Conexão falhou.');
```

return;

```
    }

    try {
        final results = await conn.execute(
            'SELECT c.nome, COUNT(p.id) AS total_pedidos, SUM(p.valor_total)
AS total_gasto '
            'FROM clientes c '
            'LEFT JOIN pedidos p ON c.id = p.id_cliente '
            'GROUP BY c.id '
            'ORDER BY total_gasto DESC',
```

```

);

print('\n--- Resumo de Pedidos por Cliente ---');
print('Nome do Cliente      | Total Pedidos | Total Gasto');
print('-----');

// Itera sobre os resultados
for (var row in results.rows) {
    final clienteNome = row.assoc()['nome'];
    final totalPedidos = row.assoc()['total_pedidos'];
    final totalGasto = row.assoc()['total_gasto'];

    // O valor gasto pode ser null se o cliente não tiver pedidos.
    final valorFormatado = totalGasto != null
        ? 'R\$
${double.parse(totalGasto.toString()).toStringAsFixed(2)}'
        : 'R\$ 0.00';

    print('${clienteNome.toString().padRight(20)} |
${totalPedidos.toString().padRight(13)} | ${valorFormatado}');
}

} catch (e) {
    print('Erro ao gerar o resumo de pedidos: $e');
} finally {
    await conn.close();
    print('Conexão com o banco de dados encerrada.');
}
}

```

Resultado da funcao resumoPedidoPorCliente com total no terminal:

```
15 |
    Run | Debug | Profile
16 | void main() async {
17 |     // Exemplo de uso da função listarPedidosComClientes
18 |     //funcao listarPedidosComClientes realiza join entre pedidos e clientes
19 |     // e exibe os pedidos com o nome do cliente
20 |
21 |     resumoPedidosPorCliente();
22 | }
23 |
```

PROBLEMAS 25 SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

Conexão com o banco de dados estabelecida com sucesso!

--- Resumo de Pedidos por Cliente ---

Nome do Cliente	Total Pedidos	Total Gasto
João da Silva	3	R\$ 430.00
Joana da Silva	3	R\$ 420.00
Maria Oliveira	0	R\$ 0.00
Anna da Silva	0	R\$ 0.00

Conexão com o banco de dados encerrada.

PS E:\carlos\documentos\P00_dart\aula05\loja_flutter>