Projeto Lógico de Banco de Dados – ECommerce

Este projeto lógico implementa o cenário de ecommerce, conforme especificado na narrativa anterior, com refinamentos para suportar clientes PJ e PF, múltiplas formas de pagamento e o controle de entregas com status e código de rastreio.

1. Estrutura das Tabelas:
   Cliente: Suporte a Pessoa Física (PF) e Pessoa Jurídica (PJ).
   Pedido: Associado a múltiplos pagamentos e entrega.
   Pagamento: Múltiplas formas de pagamento para um pedido.
   Entrega: Controla o status da entrega e o código de rastreio.

2. Relacionamentos:
   Cliente pode ser PJ ou PF, mas não ambos.
   Pedido tem uma ou mais formas de pagamento.
   Entrega tem status e código de rastreio.

Script SQL para Criação do Esquema

```sql
Tabela Cliente (com suporte a PJ e PF)
CREATE TABLE Cliente (
    ClienteID INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    TipoCliente ENUM('PJ', 'PF') NOT NULL,
    CPF_CNPJ VARCHAR(14) UNIQUE NOT NULL,   Dependendo se é PJ ou PF
    Telefone VARCHAR(20),
    Endereco VARCHAR(255)
);

Tabela Veículo
CREATE TABLE Veiculo (
    VeiculoID INT PRIMARY KEY AUTO_INCREMENT,
    Placa VARCHAR(10) UNIQUE NOT NULL,
    Modelo VARCHAR(100),
    Marca VARCHAR(50),
    Ano INT,
    ClienteID INT,
    FOREIGN KEY (ClienteID) REFERENCES Cliente(ClienteID)
);

Tabela Pedido
CREATE TABLE Pedido (
    PedidoID INT PRIMARY KEY AUTO_INCREMENT,
    ClienteID INT,
```

```
    DataPedido DATE NOT NULL,
    ValorTotal DECIMAL(10, 2),
    Status ENUM('Pendente', 'Concluído', 'Cancelado') NOT NULL,
    DataConclusao DATE,
    FOREIGN KEY (ClienteID) REFERENCES Cliente(ClienteID)
);

 Tabela Pagamento (para suportar múltiplos pagamentos)
CREATE TABLE Pagamento (
    PagamentoID INT PRIMARY KEY AUTO_INCREMENT,
    PedidoID INT,
    ValorPago DECIMAL(10, 2) NOT NULL,
    MetodoPagamento ENUM('Cartão de Crédito', 'Boleto', 'Pix') NOT NULL,
    StatusPagamento ENUM('Aprovado', 'Pendente', 'Cancelado') NOT NULL,
    DataPagamento DATE,
    FOREIGN KEY (PedidoID) REFERENCES Pedido(PedidoID)
);

 Tabela Entrega
CREATE TABLE Entrega (
    EntregaID INT PRIMARY KEY AUTO_INCREMENT,
    PedidoID INT,
    StatusEntrega ENUM('Em trânsito', 'Entregue', 'Cancelada') NOT NULL,
    CodigoRastreio VARCHAR(50),
    DataEntrega DATE,
    FOREIGN KEY (PedidoID) REFERENCES Pedido(PedidoID)
);

 Tabela Produto
CREATE TABLE Produto (
    ProdutoID INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(100) NOT NULL,
    Descricao TEXT,
    Preco DECIMAL(10, 2) NOT NULL,
    QuantidadeEmEstoque INT,
    CategoriaID INT,
    FOREIGN KEY (CategoriaID) REFERENCES Categoria(CategoriaID)
);

 Tabela Categoria
CREATE TABLE Categoria (
    CategoriaID INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(50) NOT NULL
);

 Tabela ItemPedido (N:N entre Pedido e Produto)
CREATE TABLE ItemPedido (
    ItemPedidoID INT PRIMARY KEY AUTO_INCREMENT,
```

```
    PedidoID INT,
    ProdutoID INT,
    Quantidade INT NOT NULL,
    PrecoUnitario DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (PedidoID) REFERENCES Pedido(PedidoID),
    FOREIGN KEY (ProdutoID) REFERENCES Produto(ProdutoID)
);

 Tabela Fornecedor
CREATE TABLE Fornecedor (
    FornecedorID INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(100) NOT NULL,
    CNPJ VARCHAR(14) UNIQUE NOT NULL,
    Telefone VARCHAR(20),
    Endereco VARCHAR(255)
);

 Tabela ProdutoFornecedor (N:N entre Produto e Fornecedor)
CREATE TABLE ProdutoFornecedor (
    ProdutoID INT,
    FornecedorID INT,
    PRIMARY KEY (ProdutoID, FornecedorID),
    FOREIGN KEY (ProdutoID) REFERENCES Produto(ProdutoID),
    FOREIGN KEY (FornecedorID) REFERENCES Fornecedor(FornecedorID)
);
```

 3. Inserção de Dados para Teste

```sql
 Inserindo Clientes
INSERT INTO Cliente (Nome, Email, TipoCliente, CPF_CNPJ, Telefone, Endereco)
VALUES ('Maria Silva', 'maria@exemplo.com', 'PF', '12345678901', '11999999999', 'Rua A, 123'),
       ('Empresa XYZ', 'contato@xyz.com.br', 'PJ', '98765432000189', '1133334444', 'Avenida B, 456');

 Inserindo Produtos
INSERT INTO Produto (Nome, Descricao, Preco, QuantidadeEmEstoque, CategoriaID)
VALUES ('Notebook', 'Notebook Dell i7', 3500.00, 10, 1),
       ('Mouse', 'Mouse Óptico', 50.00, 100, 2);

 Inserindo Pedido
INSERT INTO Pedido (ClienteID, DataPedido, ValorTotal, Status)
VALUES (1, '20240910', 3550.00, 'Pendente');

 Inserindo Pagamentos
```

```
INSERT INTO Pagamento (PedidoID, ValorPago, MetodoPagamento, StatusPagamento,
DataPagamento)
VALUES (1, 3000.00, 'Cartão de Crédito', 'Aprovado', '20240910'),
       (1, 550.00, 'Pix', 'Aprovado', '20240910');

 Inserindo Entrega
INSERT INTO Entrega (PedidoID, StatusEntrega, CodigoRastreio, DataEntrega)
VALUES (1, 'Em trânsito', 'BR1234567890', NULL);

 Inserindo Fornecedor
INSERT INTO Fornecedor (Nome, CNPJ, Telefone, Endereco)
VALUES ('Fornecedor ABC', '11122233000144', '1122223344', 'Rua C, 789');

 Relacionando Produto com Fornecedor
INSERT INTO ProdutoFornecedor (ProdutoID, FornecedorID)
VALUES (1, 1), (2, 1);
```

```
 4. Consultas SQL para Testes

 1. Recuperação Simples com SELECT
```sql
 Listar todos os clientes
SELECT  FROM Cliente;
```

 2. Filtros com WHERE
```sql
 Buscar pedidos pendentes
SELECT  FROM Pedido WHERE Status = 'Pendente';
```

 3. Expressão para Atributo Derivado
```sql
 Calcular o valor total dos pedidos por cliente
SELECT Cliente.Nome, SUM(Pedido.ValorTotal) AS TotalGasto
FROM Cliente
JOIN Pedido ON Cliente.ClienteID = Pedido.ClienteID
GROUP BY Cliente.Nome;
```

 4. Ordenação com ORDER BY
```sql
 Listar produtos ordenados por preço
SELECT Nome, Preco FROM Produto ORDER BY Preco DESC;
```

 5. Condições de Filtros com HAVING
```

```sql
 Mostrar clientes com valor total de pedidos superior a R$ 1000
SELECT Cliente.Nome, SUM(Pedido.ValorTotal) AS TotalGasto
FROM Cliente
JOIN Pedido ON Cliente.ClienteID = Pedido.ClienteID
GROUP BY Cliente.Nome
HAVING TotalGasto > 1000;
```

6. Junções entre Tabelas
```sql
 Relação de produtos e seus fornecedores
SELECT Produto.Nome AS Produto, Fornecedor.Nome AS Fornecedor
FROM Produto
JOIN ProdutoFornecedor ON Produto.ProdutoID = ProdutoFornecedor.ProdutoID
JOIN Fornecedor ON ProdutoFornecedor.FornecedorID = Fornecedor.FornecedorID;
```

7. Quantidade de Pedidos por Cliente
```sql
 Quantos pedidos foram feitos por cada cliente
SELECT Cliente.Nome, COUNT(Pedido.PedidoID) AS TotalPedidos
FROM Cliente
JOIN Pedido ON Cliente.ClienteID = Pedido.ClienteID
GROUP BY Cliente.Nome;
```