**1. Analisando o cenário**

Temos três blocos principais de informação:

1. **Clientes** – Dados cadastrais (nome, telefone, endereço) para entregas.
2. **Pizzas (Cardápio)** – Sabores, tamanhos, preços.
3. **Pedidos** – Ligação entre cliente e as pizzas escolhidas, data/hora, valor total.

**2. Entidades, atributos e tipos de dados**

**Entidade: Cliente**

| **Atributo** | **Tipo de dado** | **Regra de negócio** |
| --- | --- | --- |
| ClienteID | INT (PK, IDENTITY) | Identificador único do cliente |
| Nome | VARCHAR(100) | Nome completo do cliente |
| Telefone | VARCHAR(15) | Único por cliente (DDD + número) |
| Logradouro | VARCHAR(150) | Rua/Avenida |
| Numero | VARCHAR(10) | Número da residência |
| Bairro | VARCHAR(50) | Bairro |
| Cidade | VARCHAR(50) | Cidade |
| UF | CHAR(2) | Estado |
| CEP | CHAR(9) | Formato 99999-999 |

**Entidade: Pizza**

| **Atributo** | **Tipo de dado** | **Regra de negócio** |
| --- | --- | --- |
| PizzaID | INT (PK, IDENTITY) | Identificador único |
| Sabor | VARCHAR(50) | Nome do sabor |
| Tamanho | CHAR(1) | P (Pequena), M (Média), G (Grande) |
| Preco | DECIMAL(10,2) | Preço da pizza |

**Entidade: Pedido**

| **Atributo** | **Tipo de dado** | **Regra de negócio** |
| --- | --- | --- |
| PedidoID | INT (PK, IDENTITY) | Identificador único |
| ClienteID | INT (FK) | Cliente que fez o pedido |
| DataHora | DATETIME | Data e hora do pedido |
| ValorTotal | DECIMAL(10,2) | Valor final calculado |

**Entidade: ItemPedido *(tabela de ligação Pedido ↔ Pizza)***

| **Atributo** | **Tipo de dado** | **Regra de negócio** |
| --- | --- | --- |
| ItemPedidoID | INT (PK, IDENTITY) | Identificador único |
| PedidoID | INT (FK) | Pedido relacionado |
| PizzaID | INT (FK) | Pizza escolhida |
| Quantidade | INT | Quantidade dessa pizza no pedido |
| ValorUnitario | DECIMAL(10,2) | Valor da pizza no momento do pedido |

1. Modelo Relacional:

Cliente (ClienteID PK, Nome, Telefone, Logradouro, Numero, Bairro, Cidade, UF, CEP)

Pizza (PizzaID PK, Sabor, Tamanho, Preco)

Pedido (PedidoID PK, ClienteID FK, DataHora, ValorTotal)

ItemPedido (ItemPedidoID PK, PedidoID FK, PizzaID FK, Quantidade, ValorUnitario)

Relações:

* **Cliente 1:N Pedido**
* **Pedido 1:N ItemPedido**
* **Pizza 1:N ItemPedido**

4. Script SQL Server

-- Criar banco de dados

CREATE DATABASE PizzariaSenai;

GO

USE PizzariaSenai;

GO

-- Tabela Cliente

CREATE TABLE Cliente (

ClienteID INT PRIMARY KEY IDENTITY(1,1),

Nome VARCHAR(100) NOT NULL,

Telefone VARCHAR(15) NOT NULL UNIQUE,

Logradouro VARCHAR(150) NOT NULL,

Numero VARCHAR(10) NOT NULL,

Bairro VARCHAR(50) NOT NULL,

Cidade VARCHAR(50) NOT NULL,

UF CHAR(2) NOT NULL,

CEP CHAR(9) NOT NULL

);

-- Tabela Pizza

CREATE TABLE Pizza (

PizzaID INT PRIMARY KEY IDENTITY(1,1),

Sabor VARCHAR(50) NOT NULL,

Tamanho CHAR(1) NOT NULL CHECK (Tamanho IN ('P','M','G')),

Preco DECIMAL(10,2) NOT NULL

);

-- Tabela Pedido

CREATE TABLE Pedido (

PedidoID INT PRIMARY KEY IDENTITY(1,1),

ClienteID INT NOT NULL,

DataHora DATETIME NOT NULL DEFAULT GETDATE(),

ValorTotal DECIMAL(10,2) NOT NULL,

FOREIGN KEY (ClienteID) REFERENCES Cliente(ClienteID)

);

-- Tabela ItemPedido

CREATE TABLE ItemPedido (

ItemPedidoID INT PRIMARY KEY IDENTITY(1,1),

PedidoID INT NOT NULL,

PizzaID INT NOT NULL,

Quantidade INT NOT NULL CHECK (Quantidade > 0),

ValorUnitario DECIMAL(10,2) NOT NULL,

FOREIGN KEY (PedidoID) REFERENCES Pedido(PedidoID),

FOREIGN KEY (PizzaID) REFERENCES Pizza(PizzaID)

);

-- =======================================

-- INSERÇÃO DE DADOS FICTÍCIOS

-- =======================================

-- Inserir clientes

INSERT INTO Cliente (Nome, Telefone, Logradouro, Numero, Bairro, Cidade, UF, CEP) VALUES

('João Silva', '11988887777', 'Rua das Flores', '123', 'Centro', 'São Paulo', 'SP', '01001-000'),

('Maria Oliveira', '11977776666', 'Avenida Brasil', '456', 'Jardim', 'São Paulo', 'SP', '02002-000'),

('Carlos Pereira', '11966665555', 'Rua Santos', '789', 'Bela Vista', 'São Paulo', 'SP', '03003-000');

-- Inserir pizzas

INSERT INTO Pizza (Sabor, Tamanho, Preco) VALUES

('Calabresa', 'G', 50.00),

('Mussarela', 'M', 35.00),

('Portuguesa', 'G', 55.00),

('Frango com Catupiry', 'M', 40.00);

-- Inserir pedidos

INSERT INTO Pedido (ClienteID, DataHora, ValorTotal) VALUES

(1, '2025-08-10 19:30:00', 120.00),

(2, '2025-08-10 20:15:00', 75.00),

(1, '2025-08-11 18:45:00', 90.00),

(3, '2025-08-11 21:00:00', 55.00);

-- Inserir itens dos pedidos

INSERT INTO ItemPedido (PedidoID, PizzaID, Quantidade, ValorUnitario) VALUES

-- Pedido 1 (João Silva)

(1, 1, 2, 50.00),

(1, 2, 1, 35.00),

-- Pedido 2 (Maria Oliveira)

(2, 3, 1, 55.00),

(2, 2, 1, 35.00),

-- Pedido 3 (João Silva)

(3, 4, 2, 40.00),

-- Pedido 4 (Carlos Pereira)

(4, 1, 1, 50.00);

-- =======================================

-- CONSULTAS SQL

-- =======================================

-- 1. Listar todos os pedidos de um cliente

SELECT

P.PedidoID,

P.DataHora,

P.ValorTotal

FROM Pedido P

INNER JOIN Cliente C ON P.ClienteID = C.ClienteID

WHERE C.Nome = 'João Silva';

-- 2. Calcular faturamento diário

SELECT

CAST(DataHora AS DATE) AS Dia,

SUM(ValorTotal) AS Faturamento

FROM Pedido

GROUP BY CAST(DataHora AS DATE)

ORDER BY Dia;

-- 3. Buscar pizza mais vendida

SELECT

Pi.Sabor,

SUM(IP.Quantidade) AS QuantidadeVendida

FROM ItemPedido IP

INNER JOIN Pizza Pi ON IP.PizzaID = Pi.PizzaID

GROUP BY Pi.Sabor

ORDER BY QuantidadeVendida DESC

OFFSET 0 ROWS FETCH NEXT 1 ROWS ONLY;  
  
  
Caso for necessário o endereço:  
  
  
**Como ficaria com a tabela Endereco**

**Modelo relacional atualizado**:

Cliente (ClienteID PK, Nome, Telefone)

Endereco (EnderecoID PK, ClienteID FK, Logradouro, Numero, Bairro, Cidade, UF, CEP)

Pizza (PizzaID PK, Sabor, Tamanho, Preco)

Pedido (PedidoID PK, ClienteID FK, EnderecoID FK, DataHora, ValorTotal)

ItemPedido (ItemPedidoID PK, PedidoID FK, PizzaID FK, Quantidade, ValorUnitario)  
  
CREATE TABLE Endereco (

EnderecoID INT PRIMARY KEY IDENTITY(1,1),

ClienteID INT NOT NULL,

Logradouro VARCHAR(150) NOT NULL,

Numero VARCHAR(10) NOT NULL,

Bairro VARCHAR(50) NOT NULL,

Cidade VARCHAR(50) NOT NULL,

UF CHAR(2) NOT NULL,

CEP CHAR(9) NOT NULL,

FOREIGN KEY (ClienteID) REFERENCES Cliente(ClienteID)

);

ALTER TABLE Pedido

ADD EnderecoID INT NOT NULL

FOREIGN KEY (EnderecoID) REFERENCES Endereco(EnderecoID);

import pyodbc

# CONFIGURAÇÃO DA CONEXÃO

server = 'localhost' # ou NOME-DO-SERVIDOR\SQLEXPRESS

database = 'PizzariaSenai'

username = 'sa' # seu usuário do SQL Server

password = '123456' # sua senha

driver = '{ODBC Driver 17 for SQL Server}'

conn = pyodbc.connect(

f'DRIVER={driver};SERVER={server};DATABASE={database};UID={username};PWD={password}'

)

cursor = conn.cursor()

# =========================

# FUNÇÕES CRUD

# =========================

def inserir\_cliente(nome, telefone, logradouro, numero, bairro, cidade, uf, cep):

cursor.execute("""

INSERT INTO Cliente (Nome, Telefone, Logradouro, Numero, Bairro, Cidade, UF, CEP)

VALUES (?, ?, ?, ?, ?, ?, ?, ?)

""", (nome, telefone, logradouro, numero, bairro, cidade, uf, cep))

conn.commit()

print("✅ Cliente inserido com sucesso.")

def listar\_clientes():

cursor.execute("SELECT \* FROM Cliente")

for row in cursor.fetchall():

print(row)

def atualizar\_cliente(cliente\_id, novo\_telefone):

cursor.execute("UPDATE Cliente SET Telefone = ? WHERE ClienteID = ?", (novo\_telefone, cliente\_id))

conn.commit()

print("✅ Cliente atualizado.")

def deletar\_cliente(cliente\_id):

cursor.execute("DELETE FROM Cliente WHERE ClienteID = ?", (cliente\_id,))

conn.commit()

print("✅ Cliente deletado.")

def inserir\_pizza(sabor, tamanho, preco):

cursor.execute("""

INSERT INTO Pizza (Sabor, Tamanho, Preco)

VALUES (?, ?, ?)

""", (sabor, tamanho, preco))

conn.commit()

print("✅ Pizza inserida.")

def inserir\_pedido(cliente\_id, itens):

# Calcula valor total

valor\_total = sum(qtd \* preco for (\_, qtd, preco) in itens)

cursor.execute("""

INSERT INTO Pedido (ClienteID, DataHora, ValorTotal)

VALUES (?, GETDATE(), ?)

""", (cliente\_id, valor\_total))

conn.commit()

pedido\_id = cursor.execute("SELECT SCOPE\_IDENTITY()").fetchone()[0]

for pizza\_id, qtd, preco in itens:

cursor.execute("""

INSERT INTO ItemPedido (PedidoID, PizzaID, Quantidade, ValorUnitario)

VALUES (?, ?, ?, ?)

""", (pedido\_id, pizza\_id, qtd, preco))

conn.commit()

print(f"✅ Pedido {pedido\_id} inserido com sucesso.")

# =========================

# RELATÓRIOS

# =========================

def listar\_pedidos\_cliente(cliente\_id):

cursor.execute("""

SELECT Pedido.PedidoID, Pedido.DataHora, Pedido.ValorTotal

FROM Pedido

WHERE Pedido.ClienteID = ?

""", (cliente\_id,))

for row in cursor.fetchall():

print(row)

def faturamento\_por\_dia():

cursor.execute("""

SELECT CAST(DataHora AS DATE) AS Dia, SUM(ValorTotal) AS Faturamento

FROM Pedido

GROUP BY CAST(DataHora AS DATE)

ORDER BY Dia

""")

for row in cursor.fetchall():

print(row)

def pizza\_mais\_vendida():

cursor.execute("""

SELECT TOP 1 Pizza.Sabor, SUM(ItemPedido.Quantidade) AS QuantidadeVendida

FROM ItemPedido

JOIN Pizza ON ItemPedido.PizzaID = Pizza.PizzaID

GROUP BY Pizza.Sabor

ORDER BY QuantidadeVendida DESC

""")

print("🍕 Pizza mais vendida:", cursor.fetchone())

# =========================

# EXEMPLO DE USO

# =========================

if \_\_name\_\_ == "\_\_main\_\_":

# Inserir cliente

inserir\_cliente("João Silva", "11988887777", "Rua das Flores", "123", "Centro", "São Paulo", "SP", "01001-000")

# Inserir pizzas

inserir\_pizza("Calabresa", "G", 50.00)

inserir\_pizza("Mussarela", "M", 35.00)

# Inserir pedido

inserir\_pedido(1, [(1, 1, 50.00), (2, 2, 35.00)]) # (PizzaID, Quantidade, Preço)

# Relatórios

listar\_pedidos\_cliente(1)

faturamento\_por\_dia()

pizza\_mais\_vendida()

**INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN** e **FULL JOIN** usando o banco da **PizzariaSenai** e já aproveitando para criar **views** no SQL Server para facilitar consultas no dia a dia.  
  
  
-- =======================================

-- INNER JOIN - Pedidos com informações completas

-- =======================================

CREATE VIEW vw\_Pedidos\_Completos AS

SELECT

P.PedidoID,

C.Nome AS Cliente,

C.Telefone,

Pi.Sabor AS Pizza,

IP.Quantidade,

IP.ValorUnitario,

P.DataHora,

P.ValorTotal

FROM Pedido P

INNER JOIN Cliente C ON P.ClienteID = C.ClienteID

INNER JOIN ItemPedido IP ON P.PedidoID = IP.PedidoID

INNER JOIN Pizza Pi ON IP.PizzaID = Pi.PizzaID;

GO

-- =======================================

-- LEFT JOIN - Lista todos os clientes e seus pedidos (mesmo sem pedido)

-- =======================================

CREATE VIEW vw\_Clientes\_Todos\_Pedidos AS

SELECT

C.ClienteID,

C.Nome,

P.PedidoID,

P.DataHora,

P.ValorTotal

FROM Cliente C

LEFT JOIN Pedido P ON C.ClienteID = P.ClienteID;

GO

-- =======================================

-- RIGHT JOIN - Lista todos os pedidos e seus clientes (mesmo se cliente não existir mais)

-- =======================================

CREATE VIEW vw\_Pedidos\_Todos\_Clientes AS

SELECT

C.Nome,

P.PedidoID,

P.DataHora,

P.ValorTotal

FROM Cliente C

RIGHT JOIN Pedido P ON C.ClienteID = P.ClienteID;

GO

-- =======================================

-- FULL JOIN - Lista todos os clientes e todos os pedidos (mostra NULL onde não houver relação)

-- =======================================

CREATE VIEW vw\_Clientes\_e\_Pedidos AS

SELECT

C.Nome,

P.PedidoID,

P.DataHora,

P.ValorTotal

FROM Cliente C

FULL JOIN Pedido P ON C.ClienteID = P.ClienteID;

GO

-- =======================================

-- VIEW com Faturamento Diário

-- =======================================

CREATE VIEW vw\_Faturamento\_Diario AS

SELECT

CAST(DataHora AS DATE) AS Dia,

SUM(ValorTotal) AS Faturamento

FROM Pedido

GROUP BY CAST(DataHora AS DATE);

GO

-- =======================================

-- VIEW Pizza mais vendida

-- =======================================

CREATE VIEW vw\_Pizza\_Mais\_Vendida AS

SELECT

Pi.Sabor,

SUM(IP.Quantidade) AS QuantidadeVendida

FROM ItemPedido IP

INNER JOIN Pizza Pi ON IP.PizzaID = Pi.PizzaIDGROUP BY Pi.Sabor;

Python – Criando e Consultando Views com JOINs

import pyodbc

# Conexão ao SQL Server

server = 'localhost'

database = 'PizzariaSenai'

username = 'sa'

password = '123456'

driver = '{ODBC Driver 17 for SQL Server}'

conn = pyodbc.connect(

f'DRIVER={driver};SERVER={server};DATABASE={database};UID={username};PWD={password}'

)

cursor = conn.cursor()

# =============================

# 1. Criando as Views com JOINs

# =============================

def criar\_views():

views\_sql = [

# INNER JOIN

"""

CREATE OR ALTER VIEW vw\_Pedidos\_Completos AS

SELECT

P.PedidoID,

C.Nome AS Cliente,

C.Telefone,

Pi.Sabor AS Pizza,

IP.Quantidade,

IP.ValorUnitario,

P.DataHora,

P.ValorTotal

FROM Pedido P

INNER JOIN Cliente C ON P.ClienteID = C.ClienteID

INNER JOIN ItemPedido IP ON P.PedidoID = IP.PedidoID

INNER JOIN Pizza Pi ON IP.PizzaID = Pi.PizzaID;

""",

# LEFT JOIN

"""

CREATE OR ALTER VIEW vw\_Clientes\_Todos\_Pedidos AS

SELECT

C.ClienteID,

C.Nome,

P.PedidoID,

P.DataHora,

P.ValorTotal

FROM Cliente C

LEFT JOIN Pedido P ON C.ClienteID = P.ClienteID;

""",

# RIGHT JOIN

"""

CREATE OR ALTER VIEW vw\_Pedidos\_Todos\_Clientes AS

SELECT

C.Nome,

P.PedidoID,

P.DataHora,

P.ValorTotal

FROM Cliente C

RIGHT JOIN Pedido P ON C.ClienteID = P.ClienteID;

""",

# FULL JOIN

"""

CREATE OR ALTER VIEW vw\_Clientes\_e\_Pedidos AS

SELECT

C.Nome,

P.PedidoID,

P.DataHora,

P.ValorTotal

FROM Cliente C

FULL JOIN Pedido P ON C.ClienteID = P.ClienteID;

""",

# Faturamento diário

"""

CREATE OR ALTER VIEW vw\_Faturamento\_Diario AS

SELECT

CAST(DataHora AS DATE) AS Dia,

SUM(ValorTotal) AS Faturamento

FROM Pedido

GROUP BY CAST(DataHora AS DATE);

""",

# Pizza mais vendida

"""

CREATE OR ALTER VIEW vw\_Pizza\_Mais\_Vendida AS

SELECT

Pi.Sabor,

SUM(IP.Quantidade) AS QuantidadeVendida

FROM ItemPedido IP

INNER JOIN Pizza Pi ON IP.PizzaID = Pi.PizzaID

GROUP BY Pi.Sabor;

"""

]

for sql in views\_sql:

cursor.execute(sql)

conn.commit()

print("✅ Views criadas/atualizadas com sucesso!")

# =============================

# 2. Consultando as Views

# =============================

def consultar\_view(nome\_view):

cursor.execute(f"SELECT \* FROM {nome\_view}")

colunas = [desc[0] for desc in cursor.description] # pega nome das colunas

resultados = cursor.fetchall()

print(f"\n📄 Resultados da view: {nome\_view}")

print(colunas)

for linha in resultados:

print(linha)

# =============================

# Execução de exemplo

# =============================

if \_\_name\_\_ == "\_\_main\_\_":

criar\_views()

consultar\_view("vw\_Pedidos\_Completos")

consultar\_view("vw\_Clientes\_Todos\_Pedidos")

consultar\_view("vw\_Faturamento\_Diario")

consultar\_view("vw\_Pizza\_Mais\_Vendida")

conn.close()

1. Cria o banco **PizzariaSenai** e as tabelas.
2. Insere dados fictícios (clientes, endereços, cardápio, pedidos, itens).
3. Cria **views** com INNER JOIN, LEFT JOIN, RIGHT JOIN e FULL JOIN.
4. Consulta e exibe os resultados das views.
5. Mantém o CRUD básico (inserir cliente, pizza, pedido).

O código será feito usando **pyodbc** para conectar ao SQL Server.

import pyodbc

# Configuração da conexão com SQL Server

server = 'localhost' # ou o nome do servidor

database = 'PizzariaSenai'

username = 'sa'

password = 'sua\_senha'

conn\_str = f'DRIVER={{SQL Server}};SERVER={server};DATABASE={database};UID={username};PWD={password}'

# Conectar

conn = pyodbc.connect(conn\_str, autocommit=True)

cursor = conn.cursor()

# 1 - Criar tabelas

cursor.execute("""

IF OBJECT\_ID('Clientes', 'U') IS NOT NULL DROP TABLE Clientes;

CREATE TABLE Clientes (

id\_cliente INT PRIMARY KEY IDENTITY(1,1),

nome NVARCHAR(100),

telefone NVARCHAR(15)

);

""")

cursor.execute("""

IF OBJECT\_ID('Enderecos', 'U') IS NOT NULL DROP TABLE Enderecos;

CREATE TABLE Enderecos (

id\_endereco INT PRIMARY KEY IDENTITY(1,1),

id\_cliente INT,

logradouro NVARCHAR(200),

numero NVARCHAR(10),

bairro NVARCHAR(50),

cidade NVARCHAR(50),

estado CHAR(2),

cep NVARCHAR(9),

FOREIGN KEY (id\_cliente) REFERENCES Clientes(id\_cliente)

);

""")

cursor.execute("""

IF OBJECT\_ID('Pizzas', 'U') IS NOT NULL DROP TABLE Pizzas;

CREATE TABLE Pizzas (

id\_pizza INT PRIMARY KEY IDENTITY(1,1),

sabor NVARCHAR(100),

preco DECIMAL(10,2)

);

""")

cursor.execute("""

IF OBJECT\_ID('Pedidos', 'U') IS NOT NULL DROP TABLE Pedidos;

CREATE TABLE Pedidos (

id\_pedido INT PRIMARY KEY IDENTITY(1,1),

id\_cliente INT,

data\_pedido DATETIME DEFAULT GETDATE(),

valor\_total DECIMAL(10,2),

FOREIGN KEY (id\_cliente) REFERENCES Clientes(id\_cliente)

);

""")

cursor.execute("""

IF OBJECT\_ID('ItensPedido', 'U') IS NOT NULL DROP TABLE ItensPedido;

CREATE TABLE ItensPedido (

id\_item INT PRIMARY KEY IDENTITY(1,1),

id\_pedido INT,

id\_pizza INT,

quantidade INT,

subtotal DECIMAL(10,2),

FOREIGN KEY (id\_pedido) REFERENCES Pedidos(id\_pedido),

FOREIGN KEY (id\_pizza) REFERENCES Pizzas(id\_pizza)

);

""")

# 2 - Inserir dados fictícios

cursor.execute("INSERT INTO Clientes (nome, telefone) VALUES ('João Silva', '11999999999')")

cursor.execute("INSERT INTO Clientes (nome, telefone) VALUES ('Maria Souza', '11888888888')")

cursor.execute("INSERT INTO Enderecos (id\_cliente, logradouro, numero, bairro, cidade, estado, cep) VALUES (1, 'Rua A', '100', 'Centro', 'São Paulo', 'SP', '01000-000')")

cursor.execute("INSERT INTO Enderecos (id\_cliente, logradouro, numero, bairro, cidade, estado, cep) VALUES (2, 'Av B', '200', 'Jardins', 'São Paulo', 'SP', '02000-000')")

cursor.execute("INSERT INTO Pizzas (sabor, preco) VALUES ('Calabresa', 40.00)")

cursor.execute("INSERT INTO Pizzas (sabor, preco) VALUES ('Mussarela', 35.00)")

cursor.execute("INSERT INTO Pizzas (sabor, preco) VALUES ('Portuguesa', 45.00)")

cursor.execute("INSERT INTO Pedidos (id\_cliente, valor\_total) VALUES (1, 80.00)")

cursor.execute("INSERT INTO Pedidos (id\_cliente, valor\_total) VALUES (2, 45.00)")

cursor.execute("INSERT INTO ItensPedido (id\_pedido, id\_pizza, quantidade, subtotal) VALUES (1, 1, 1, 40.00)")

cursor.execute("INSERT INTO ItensPedido (id\_pedido, id\_pizza, quantidade, subtotal) VALUES (1, 2, 1, 40.00)")

cursor.execute("INSERT INTO ItensPedido (id\_pedido, id\_pizza, quantidade, subtotal) VALUES (2, 3, 1, 45.00)")

# 3 - Criar Views com JOINs

cursor.execute("""

IF OBJECT\_ID('vw\_PedidosClientes', 'V') IS NOT NULL DROP VIEW vw\_PedidosClientes;

CREATE VIEW vw\_PedidosClientes AS

SELECT p.id\_pedido, c.nome, c.telefone, p.data\_pedido, p.valor\_total

FROM Pedidos p

INNER JOIN Clientes c ON p.id\_cliente = c.id\_cliente;

""")

cursor.execute("""

IF OBJECT\_ID('vw\_PedidosEnderecos', 'V') IS NOT NULL DROP VIEW vw\_PedidosEnderecos;

CREATE VIEW vw\_PedidosEnderecos AS

SELECT c.nome, e.logradouro, e.numero, e.bairro, e.cidade, e.estado, e.cep

FROM Clientes c

LEFT JOIN Enderecos e ON c.id\_cliente = e.id\_cliente;

""")

cursor.execute("""

IF OBJECT\_ID('vw\_TodosClientesPedidos', 'V') IS NOT NULL DROP VIEW vw\_TodosClientesPedidos;

CREATE VIEW vw\_TodosClientesPedidos AS

SELECT c.nome, p.id\_pedido, p.valor\_total

FROM Clientes c

RIGHT JOIN Pedidos p ON c.id\_cliente = p.id\_cliente;

""")

cursor.execute("""

IF OBJECT\_ID('vw\_ClientesFullJoin', 'V') IS NOT NULL DROP VIEW vw\_ClientesFullJoin;

CREATE VIEW vw\_ClientesFullJoin AS

SELECT c.nome, p.id\_pedido, p.valor\_total

FROM Clientes c

FULL JOIN Pedidos p ON c.id\_cliente = p.id\_cliente;

""")

# 4 - Consultar Views

def exibir\_view(nome\_view):

print(f"\nResultados da view {nome\_view}:")

for row in cursor.execute(f"SELECT \* FROM {nome\_view}"):

print(row)

exibir\_view("vw\_PedidosClientes")

exibir\_view("vw\_PedidosEnderecos")

exibir\_view("vw\_TodosClientesPedidos")

exibir\_view("vw\_ClientesFullJoin")

cursor.close()

conn.close()  
  
  
Esse script já **cria tudo, insere dados, gera as views, e exibe os resultados** no console.  
Se você quiser, posso **jogar esse código junto com o CRUD que já havíamos feito** para ter um sistema único e funcional.