TABELAS:

USE SEUCARDAPIO\_VICTOR

GO

create table Cardapio\_sc (

IdCardapio\_sc INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

Almoco\_sc VARCHAR(100) NOT NULL,

Janta\_sc VARCHAR(100) NOT NULL,

Bebidas\_sc VARCHAR(25) NOT NULL,

DiasDaSemana\_sc VARCHAR(25) NOT NULL

);

\*\*//Adcionar após ter criado a coluna: \\\*\*

Alter table Cardapio\_sc

ADD CONSTRAINT CK\_DiasDaSemana CHECK (DiasDaSemana\_sc IN ('domingo','segunda-feira', 'terça-feira', 'quarta-feira', 'quinta-feira', 'sexta-feira', 'sábado'));

ALTER TABLE Cardapio\_sc

ADD CONSTRAINT CK\_Drinks CHECK (Bebidas\_sc IN (‘Refrigerante’, ‘Agua’, ‘Suco’,’Nao’));

\*\*//Colocar gatilho para pessoa escolher Coca-Cola, Fanta ou guaraná ou deixar na logica.\\\*\*

)

CREATE TABLE Clientes\_sc (

IdCliente\_sc INT IDENTITY(1, 1) PRIMARY KEY,

Nome\_sc VARCHAR(50) NOT NULL,

Idade\_sc INT NOT NULL,

Peso\_sc DECIMAL(4,2) NOT NULL,

TipoAlimentacao\_sc VARCHAR(20) NOT NULL

);

\*\*//usado para que a chave estrangeira funciona em uma coluna que não é primary key\\\*\*

ALTER TABLE Clientes\_sc

ADD CONSTRAINT NomeUnico UNIQUE (TipoAlimentacao\_sc); NÃO ADCIONAR

\*\*//Adcionar após ter criado a coluna: \\\*\*

Alter table Clientes\_sc

ADD CONSTRAINT CK\_ALIMENTACAO\_ CHECK (TipoAlimentacao\_sc IN (‘vegetariano’, ‘vegano’, ’como de tudo’));

CREATE TABLE Pedidos\_sc(

IdPedidos\_sc INT IDENTITY(1, 1) PRIMARY KEY,

IdCliente\_sc INT NOT NULL,

Contato\_sc VARCHAR(25) NOT NULL,

CodigoPostal\_sc INT NOT NULL,

Endereco\_sc VARCHAR(100) NOT NULL,

Alimentacao\_sc VARCHAR(20) NOT NULL,

HabitualAlmoco\_sc INT NOT NULL,

HabitualJanta\_sc INT NOT NULL,

DataPedido\_sc DATE NOT NULL,

PedidoRealizado\_sc VARCHAR(5) NOT NULL,

PedidoSuspenso\_sc VARCHAR(5) NOT NULL,

Entregador\_sc VARCHAR(50) NOT NULL,

FOREIGN KEY (IdCliente\_sc) REFERENCES Clientes\_sc(IdCliente\_sc),

FOREIGN KEY (HabitualAlmoco\_sc) REFERENCES Cardapio\_sc(IdCardapio\_sc),

FOREIGN KEY (HabitualJanta\_sc) REFERENCES Cardapio\_sc(IdCardapio\_sc)

);

Alter table Pedidos\_sc

ADD CONSTRAINT CK\_ALIMENTACAO\_sc CHECK (PedidoRealizado\_sc IN (‘Sim’, ‘Nao’));

Alter table Pedidos\_sc

ADD CONSTRAINT CK\_ALIMENTACAO CHECK (PedidoSuspenso\_sc IN (‘Sim’, ‘Nao’));

CREATE TABLE Entregas\_sc (

IdEntregas\_sc INT IDENTITY PRIMARY KEY,

IdPedidos\_sc INT NOT NULL,

Forncedor\_sc VARCHAR(30) NOT NULL,

CNPJforn\_sc NUMERIC(20) NOT NULL,

MotoBoy\_sc VARCHAR(50) NOT NULL,

ContatoEntregador CHAR(15) NOT NULL,

PreçoFretes\_sc DECIMAL(4, 2) NOT NULL,

DiasdaSemana\_sc VARCHAR(25) NOT NULL,

Realizados\_sc VARCHAR(5) NOT NULL,

Suspensos\_sc VARCHAR(5) NOT NULL,

FOREIGN KEY (IdPedidos\_sc) REFERENCES Pedidos\_sc(IdPedidos\_sc),

);

Alter table Entregas\_sc

ADD CONSTRAINT CK\_ALIMENT\_ CHECK (Realizados\_sc IN (‘Sim’, ‘Nao’));

Alter table Entregas\_sc

ADD CONSTRAINT CK\_ALIMENT CHECK (Suspensos\_sc IN (‘Sim’, ‘Nao’));

\*\*//usado para que a chave estrangeira funciona em uma coluna que não é primary key\\\*\*

ALTER TABLE Cardapio\_sc

ADD CONSTRAINT DiasdaSemana\_sc UNIQUE (DiasdaSemana\_sc); NÃO ADCIONAR

PROCEDURE> cliente,cardapio,entregas:

\*\*//procedure criada para Buscar todos clientes\\\*\* USEI

CREATE PROCEDURE [dbo].[controleClientes]

AS

BEGIN

SELECT

a.Nome\_sc,

b.Contato\_sc,

b.CodigoPostal\_sc,

b.Endereco\_sc,

b.HabitualAlmoco\_sc,

b.HabitualJanta\_sc

FROM

Clientes\_sc a

INNER JOIN

Pedidos\_sc b ON a.IdCliente\_sc = b.IdCliente\_sc

END;

GO

\*\*//procedure criada para Buscar clientes separadamente por ID\\\*\*

CREATE PROCEDURE controleClientesUN USEI

@IdNome\_sc INT

AS

BEGIN

SELECT

a.Nome\_sc,

b.Contato\_sc,

b.CodigoPostal\_sc,

b.Endereco\_sc,

b.HabitualAlmoco\_sc,

b.HabitualJanta\_sc

FROM

Clientes\_sc a

INNER JOIN

Pedidos\_sc b ON a.IdCliente\_sc = b.IdCliente\_sc

Where a.IdCliente\_sc = @IdNome\_sc;

END;

GO

//\*\*Procedure para mostrar os relatorios por dia\*\*\\ USEI

CREATE PROCEDURE RelatorioPorDia

@DataPedido DATE

AS

BEGIN

SELECT

c.IdCliente\_sc,

c.Nome\_sc,

c.Idade\_sc,

c.Peso\_sc,

c.TipoAlimentacao\_sc,

p.IdPedidos\_sc,

p.Contato\_sc,

p.CodigoPostal\_sc,

p.Endereco\_sc,

p.Alimentacao\_sc,

p.HabitualAlmoco\_sc,

p.HabitualJanta\_sc,

p.DataPedido\_sc,

p.PedidoRealizado\_sc,

p.PedidoSuspenso\_sc,

p.Entregador\_sc,

caA.Almoco\_sc AS AlmocoMaisPedido,

caJ.Janta\_sc AS JantaPedido,

e.IdEntregas\_sc,

e.Forncedor\_sc,

e.CNPJforn\_sc,

e.MotoBoy\_sc AS EntregadorEntrega,

e.ContatoEntregador,

e.PreçoFretes\_sc,

e.DiasdaSemana\_sc,

e.Realizados\_sc,

e.Suspensos\_sc

FROM Clientes\_sc c

LEFT JOIN Pedidos\_sc p ON c.IdCliente\_sc = p.IdCliente\_sc

LEFT JOIN Cardapio\_sc caA ON p.HabitualAlmoco\_sc = caA.IdCardapio\_sc

LEFT JOIN Cardapio\_sc caJ ON p.HabitualJanta\_sc = caJ.IdCardapio\_sc

LEFT JOIN Entregas\_sc e ON p.IdPedidos\_sc = e.IdPedidos\_sc

WHERE p.DataPedido\_sc = @DataPedido

ORDER BY c.IdCliente\_sc, p.IdPedidos\_sc;

END;

GO

//\*\*Procedure para achar pelo tipo de alimentação\*\*\\ USEI

CREATE PROCEDURE RelatorioPorTipoAlimentacao

@TipoAlimentacao VARCHAR(20)

AS

BEGIN

SELECT

c.IdCliente\_sc,

c.Nome\_sc,

c.Idade\_sc,

c.Peso\_sc,

c.TipoAlimentacao\_sc,

p.IdPedidos\_sc,

p.Contato\_sc,

p.CodigoPostal\_sc,

p.Endereco\_sc,

p.Alimentacao\_sc,

p.HabitualAlmoco\_sc,

p.HabitualJanta\_sc,

p.DataPedido\_sc,

p.PedidoRealizado\_sc,

p.PedidoSuspenso\_sc,

p.Entregador\_sc,

caA.Almoco\_sc AS AlmocoMaisPedido,

caJ.Janta\_sc AS JantaPedido,

e.IdEntregas\_sc,

e.Forncedor\_sc,

e.CNPJforn\_sc,

e.MotoBoy\_sc AS EntregadorEntrega,

e.ContatoEntregador,

e.PreçoFretes\_sc,

e.DiasdaSemana\_sc,

e.Realizados\_sc,

e.Suspensos\_sc

FROM Clientes\_sc c

LEFT JOIN Pedidos\_sc p ON c.IdCliente\_sc = p.IdCliente\_sc

LEFT JOIN Cardapio\_sc caA ON p.HabitualAlmoco\_sc = caA.IdCardapio\_sc

LEFT JOIN Cardapio\_sc caJ ON p.HabitualJanta\_sc = caJ.IdCardapio\_sc

LEFT JOIN Entregas\_sc e ON p.IdPedidos\_sc = e.IdPedidos\_sc

WHERE c.TipoAlimentacao\_sc = @TipoAlimentacao

ORDER BY c.IdCliente\_sc, p.IdPedidos\_sc;

END;

GO

Para funcionar todas as tabelas devem ser preenchidas

TRIGGERS(gatilhos):

DESCOBRIR ONDE ESTA O GATILHO

SELECT

name AS trigger\_name,

object\_name(parent\_id) AS table\_name,

type\_desc AS trigger\_type,

create\_date,

modify\_date

FROM

sys.triggers

WHERE

type = 'TR' -- Filtrar por gatilhos DML (INSERT, UPDATE, DELETE)

ORDER BY

table\_name,

trigger\_name;

TRIGGER PARA INSERIR, ATUALIZAR E DELETAR DE UMA TABELA APENAS:

Usei em PEDIDOS, ENTREGAS, USEI

CREATE TRIGGER trg\_Pedidosid / trg\_Entregasid

ON Clientes\_sc

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted)

BEGIN

INSERT INTO Pedidos\_sc (IdCliente\_sc)

SELECT IdCliente\_sc FROM inserted

END

IF EXISTS (SELECt \* FROM deleted)

BEGIN

UPDATE Pedidos\_sc

SET IdCliente\_sc = i.IdCliente\_sc

FROM Pedidos\_sc p

JOIN inserted i ON p.IdCliente\_sc = i.IdCliente\_sc

WHERE EXISTS (SELECT 1 FROM deleted d WHERE d.IdCliente\_sc = p.IdCliente\_sc);

END

IF EXISTS (SELECT \* FROM deleted)

BEGIN

DELETE FROM Pedidos\_sc

WHERE IdCliente\_sc IN (SELECT IdCliente\_sc FROM deleted);

END

END;

CREATE TRIGGER trg\_Entregasid USEI

ON Pedidos\_sc

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted)

BEGIN

INSERT INTO Entregas\_sc (IdPedidos\_sc)

SELECT IdPedidos\_sc FROM inserted

END

IF EXISTS (SELECt \* FROM deleted)

BEGIN

UPDATE Entregas\_sc

SET IdPedidos\_sc = i.IdPedidos\_sc

FROM Pedidos\_sc p

JOIN inserted i ON p.IdPedidos\_sc = i.IdPedidos\_sc

WHERE EXISTS (SELECT 1 FROM deleted d WHERE d.IdPedidos\_sc = p.IdPedidos\_sc);

END

IF EXISTS (SELECT \* FROM deleted)

BEGIN

DELETE FROM Pedidos\_sc

WHERE IdPedidos\_sc IN (SELECT IdPedidos\_sc FROM deleted);

END

END;

//\*\*Trigger para atualizar Alimentacao\_sc em Pedidos\_sc\*\*\\

//\*\*Trigger para atualizar ENGTREGADOR\_sc\*(PEDIDOS) em eNTREGAS\_sc\*\*\\

CREATE TRIGGER TR\_PreencheAlimentacaoPedidos / USEI TR\_PreencheEntregadorEntregas

ON Pedidos\_sc

AFTER INSERT

AS

BEGIN

UPDATE p

SET p.Alimentacao\_sc = c.TipoAlimentacao\_sc

FROM Pedidos\_sc p

INNER JOIN inserted i ON p.IdPedidos\_sc = i.IdPedidos\_sc

INNER JOIN Clientes\_sc c ON p.IdCliente\_sc = c.IdCliente\_sc

END;

GO

CREATE TRIGGER TR\_PreencheMotoBoy USEI

ON Pedidos\_sc

AFTER INSERT

AS

BEGIN

UPDATE e

SET e.MotoBoy\_sc = p.Entregador\_sc

FROM Entregas\_sc e

INNER JOIN inserted i ON e.MotoBoy\_sc = i.Entregador\_sc

INNER JOIN Pedidos\_sc p ON e.MotoBoy\_sc = p.Entregador\_sc

END;

GO

//\*\*Trigger para atualizar Entregador\_sc em Pedidos\_sc ao inserir ou atualizar Entregas\_sc\*\*\\ NÃO DEU CERTO

CREATE TRIGGER TR\_AtualizaEntregadorPedidos

ON Entregas\_sc

AFTER INSERT, UPDATE

AS

BEGIN

UPDATE p

SET p.Entregador\_sc = e.Entregador\_sc

FROM Pedidos\_sc p

INNER JOIN inserted i ON p.IdPedidos\_sc = i.IdPedidos\_sc

INNER JOIN Entregas\_sc e ON e.IdPedidos\_sc = p.IdPedidos\_sc

END;

GO

-----------------------------

//\*\*Trigger para atualizar o HabitualAlmoco\_sc do cliente\*\*\\

CREATE TRIGGER TR\_AtualizaHabitualAlmoco USEI

ON Pedidos\_sc

AFTER INSERT

AS

BEGIN

UPDATE p

SET HabitualAlmoco\_sc = (

SELECT TOP 1 HabitualAlmoco\_sc

FROM Pedidos\_sc

WHERE IdCliente\_sc = p.IdCliente\_sc

GROUP BY HabitualAlmoco\_sc

ORDER BY COUNT(\*) DESC

)

FROM Pedidos\_sc p

INNER JOIN inserted i ON p.IdPedidos\_sc = i.IdPedidos\_sc

END;

GO

//\*\*Trigger para atualizar o HabitualJanta\_sc do cliente\*\*\\

CREATE TRIGGER TR\_AtualizaHabitualJanta USEI

ON Pedidos\_sc

AFTER INSERT

AS

BEGIN

UPDATE p

SET HabitualJanta\_sc = (

SELECT TOP 1 HabitualJanta\_sc

FROM Pedidos\_sc

WHERE IdCliente\_sc = p.IdCliente\_sc

GROUP BY HabitualJanta\_sc

ORDER BY COUNT(\*) DESC

)

FROM Pedidos\_sc p

INNER JOIN inserted i ON p.IdPedidos\_sc = i.IdPedidos\_sc

END;

GO

CREATE TRIGGER TR\_AtualizaEntregas\_sc

ON Pedidos\_sc

AFTER INSERT, UPDATE

AS

BEGIN

UPDATE Entregas\_sc

SET Entregador\_sc(entregas) = i.Entregador\_sc(pedidos)

FROM Entregador\_sc e (entregas)

INNER JOIN inserted i ON e.entregador\_sc = i.Entregador\_sc -- Assumindo uma chave primária para o relacionamento

WHERE i.Entregador\_sc IS NOT NULL; -- Opcional: evita atualizar com NULLs

END;