

Centro Universitário UNISATC

Engenharia de Software 3a fase – Banco de Dados II – Prof. Jorge Luiz da Silva

TRABALHO FINAL COM BASE EM METODOLOGIAS ATIVAS DE APRENDIZAGEM

Projeto de banco de dados para um sistema de Atendimento Doceria

Ana Beatriz Meller - @AnaBea07

Daniela Cardoso - @DanielaMF

Emely Pickler - @emelypickler

Julia Meller – @juliameller

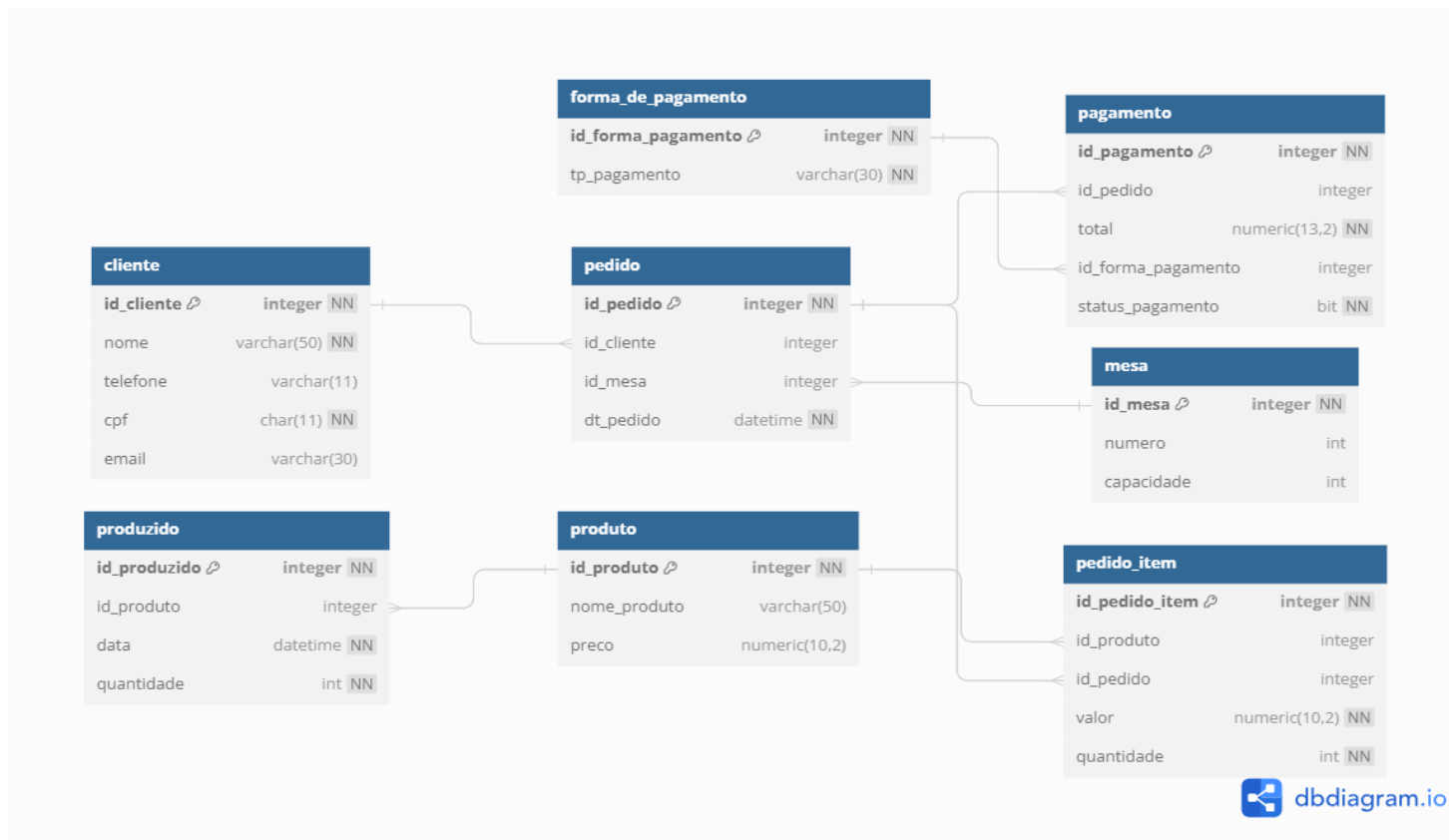
Thiago Larangeira - @thiagolarangeiras

Criciúma, 13/11/2023

URL do projeto no GitHub

<https://github.com/juliameller/trabalhobd2>

Modelo ER Físico



Dicionário de Dados

Tabela	Cliente					
Descrição	Tabela responsável por armazenar os dados dos clientes da doceria					
Atributos						
Nome da Coluna	Tipo do Dado	Valor min e max	Nulidade	PK	FK	Descrição
id_cliente	int	1 – sem limite	NOT NULL	X		Identificador único do cliente
nome	varchar(50)	10 - 50	NOT NULL			Nome do cliente
telefone	varchar(11)	1 - 11				Telefone do cliente
cpf	char(11)	1 - 11	NOT NULL			CPF do cliente
email	varchar(30)	10 - 30				Email do cliente

Tabela	Produto					
Descrição	Tabela responsável por armazenar os dados dos produtos da doceria					
Atributos						
Nome da Coluna	Tipo do Dado	Valor min e max	Nulidade	PK	FK	Descrição
id_produto	int	1 – sem limite	NOT NULL	X		Identificador único do produto
nome_produto	varchar(50)	10 - 50				Nome do produto
preco	numeric(10,2)	10				Preço do produto

Tabela	Mesa					
Descrição	Tabela responsável por armazenar os dados das mesas da doceria					
Atributos						
Nome da Coluna	Tipo do Dado	Valor min e max	Nulidade	PK	FK	Descrição
id_mesa	int	1 – sem limite	NOT NULL	X		Identificador único da mesa
numero	int	2 – sem limite				Número da mesa
capacidade	int	3 – sem limite				Capacidade da mesa

Tabela	Pedido					
Descrição	Tabela responsável por armazenar os dados dos pedidos da doceria					
Atributos						
Nome da Coluna	Tipo do Dado	Valor min e max	Nulidade	PK	FK	Descrição
id_pedido	int	1 – sem limite	NOT NULL	X		Identificador único do pedido
id_cliente	int	1 – sem limite			X	Identificador único do cliente
id_mesa	int	1 – sem limite			X	Identificador único da mesa
dt_pedido	datetime	1753-01-01 00:00:00.000 - 9999-12-31	NOT NULL			Dia e Hora do pedido
Índice (Não Possui Índice)		23:59:59.997				

Tabela	Pedido_item					
Descrição	Tabela responsável por armazenar os dados dos pedidos da doceria					
Atributos						
Nome da Coluna	Tipo do Dado	Valor min e max	Nulidade	PK	FK	Descrição
id_pedido_item	int	1 – sem limite	NOT NULL	X		Identificador único do item do pedido
id_pedido	int	1 – sem limite			X	Identificador único do pedido
id_produto	int	1 – sem limite			X	Identificador único do produto
valor	numeric(10,2)	10	NOT NULL			Valor do item do pedido
quantidade	int	1 – sem limite	NOT NULL			Quantidade do item do pedido
Índice						
Nome do índice	Clustered	NonClustered	Unique	Colunas		
idx_pedido_item_id_pedido	X		X	id_pedido		
idx_pedido_item_id_pedido2		X		id_pedido INCLUDE: valor, quantidade		

Tabela	Pagamento					
Descrição	Tabela responsável por armazenar os dados dos pagamentos da doceria					
Atributos						
Nome da Coluna	Tipo do Dado	Valor min e max	Nulidade	PK	FK	Descrição
id_pagamento	int	1 – sem limite	NOT NULL	X		Identificador único do item do pedido
id_pedido	int	1 – sem limite			X	Identificador único do pedido
total	numeric(13,2)	13	NOT NULL			Total do pagamento
id_forma_pagamento	int	1 – sem limite			X	Identificador único da forma de pagamento
status_pagamento	bit	pago = 1 ou em aberto = 0	NOT NULL			Status do pagamento
Índice						
Nome do índice	Clustered	NonClustered	Unique	Colunas		
idx_pagamento_id_pedido	X			id_pedido		
idx_pagamento_id_pedido_status_pagamento		X		id_pedido, status_pagamento INCLUDE: total		

Tabela	Forma_de_pagamento					
Descrição	Tabela responsável por armazenar os dados das formas de pagamentos da doceria					
Atributos						
Nome da Coluna	Tipo do Dado	Valor min e max	Nulidade	PK	FK	Descrição
id_forma_pagamento	int	1 – sem limite	NOT NULL	X		Identificador único da forma de pagamento
tp_pagamento	varchar(30)	1 - 30	NOT NULL			Tipo de pagamento

Tabela	Produzido					
Descrição	Tabela responsável por armazenar os dados do que foi produzido da doceria					
Atributos						
Nome da Coluna	Tipo do Dado	Valor min e max	Nulidade	PK	FK	Descrição
id_produzido	int	1 – sem limite	NOT NULL	X		Identificador único do item do pedido
id_produto	int	1 – sem limite			X	Identificador único do produto
data	datetime	1753-01-01 00:00:00.000 - 9999-12-31 23:59:59.997	NOT NULL			Data da produção
quantidade	int	1 – sem limite	NOT NULL			Quantidade do que foi produzido

Script dos comandos DDL para criação do Banco de dados

```
CREATE TABLE [cliente] (
    [id_cliente] integer PRIMARY KEY NOT NULL IDENTITY(1, 1),
    [nome] varchar(50) NOT NULL,
    [telefone] varchar(11),
    [cpf] char(11) NOT NULL,
    [email] varchar(30)
)
GO
```

```
CREATE TABLE [produto] (
    [id_produto] integer PRIMARY KEY NOT NULL IDENTITY(1, 1),
    [nome_produto] varchar(50) NOT NULL,
```

```
[preco] numeric(10,2)
)
GO
```

```
CREATE TABLE [mesa] (
    [id_mesa] integer PRIMARY KEY NOT NULL IDENTITY(1, 1),
    [numero] int,
    [capacidade] int
)
GO
```

```
CREATE TABLE [pedido] (
    [id_pedido] integer PRIMARY KEY NOT NULL IDENTITY(1, 1),
    [id_cliente] integer NOT NULL,
    [id_mesa] integer,
    [dt_pedido] datetime NOT NULL
)
GO
```

```
CREATE TABLE [pedido_item] (
    [id_pedido_item] integer PRIMARY KEY NOT NULL IDENTITY(1, 1),
    [id_produto] integer NOT NULL,
```

```
[id_pedido] integer NOT NULL,  
[valor] numeric(10,2) NOT NULL,  
[quantidade] int NOT NULL  
)  
GO
```

```
CREATE TABLE [pagamento] (  
[id_pagamento] integer PRIMARY KEY NOT NULL IDENTITY(1, 1),  
[id_pedido] integer NOT NULL,  
[total] numeric(13,2) NOT NULL,  
[id_forma_pagamento] integer NOT NULL,  
[status_pagamento] bit NOT NULL -- pago = 1 ou em aberto = 0  
)  
GO
```

```
CREATE TABLE [forma_de_pagamento] (  
[id_forma_pagamento] integer PRIMARY KEY NOT NULL IDENTITY(1, 1),  
[tp_pagamento] varchar(30) NOT NULL  
)  
GO
```

```
CREATE TABLE [produzido] (  

```

```
[id_produzido] integer PRIMARY KEY NOT NULL IDENTITY(1, 1),  
[id_produto] integer NOT NULL,  
[data] datetime NOT NULL,  
[quantidade] int NOT NULL  
)  
GO
```

```
ALTER TABLE [pedido] ADD CONSTRAINT [fk_pedido_cliente] FOREIGN KEY ([id_cliente]) REFERENCES [cliente] ([id_cliente])  
GO
```

```
ALTER TABLE [pedido] ADD CONSTRAINT [fk_pedido_mesa] FOREIGN KEY ([id_mesa]) REFERENCES [mesa] ([id_mesa])  
GO
```

```
ALTER TABLE [pedido_item] ADD CONSTRAINT [fk_pedido_item_produto] FOREIGN KEY ([id_produto]) REFERENCES [produto] ([id_produto])  
GO
```

```
ALTER TABLE [pedido_item] ADD CONSTRAINT [fk_pedido_item_pedido] FOREIGN KEY ([id_pedido]) REFERENCES [pedido] ([id_pedido])  
GO
```

```
ALTER TABLE [pagamento] ADD CONSTRAINT [fk_pagamento_pedido] FOREIGN KEY ([id_pedido]) REFERENCES [pedido] ([id_pedido])  
GO
```



```
ALTER TABLE [pagamento] ADD CONSTRAINT [fk_pagamento_forma_pagamento] FOREIGN KEY ([id_forma_pagamento]) REFERENCES [forma_de_pagamento] ([id_forma_pagamento])
```

```
GO
```

```
ALTER TABLE [produzido] ADD CONSTRAINT [fk_produzido_produto] FOREIGN KEY ([id_produto]) REFERENCES [produto] ([id_produto])
```

```
GO
```

Script que popula as tabelas do Banco de dados

```
INSERT INTO [cliente] ([nome], [telefone], [cpf], [email])
```

```
VALUES
```

```
('Maria Silva', '48999998888', '12345678901', 'maria.silva@gmail.com'),
```

```
('João Souza', '48888887777', '09876543210', 'joao.souza@hotmail.com'),
```

```
('Ana Pereira', '48777776666', '12312312312', 'ana.pereira@gmail.com'),
```

```
('Carlos Lima', '48666665555', '32132132132', 'carlos.lima@hotmail.com'),
```

```
('Fernanda Oliveira', '48555554444', '45645645645', 'fernanda.oliveira@gmail.com'),
```

```
('Pedro Almeida', '48444443333', '65465465465', 'pedro.almeida@hotmail.com'),
```

```
('Mariana Costa', '48333332222', '78978978978', 'mariana.costa@gmail.com'),
```

```
('Lucas Santos', '48222221111', '98798798798', 'lucas.santos@hotmail.com'),
```

```
('Juliana Rocha', '48111110000', '11122233344', 'juliana.rocha@gmail.com'),
```

```
('Gabriel Torres', '4800009999', '55566677788', 'gabriel.torres@hotmail.com'),
```

```
('Bruno Martins', '4887654321', '12309876543', 'bruno.martins@gmail.com'),
```

```
('Larissa Mendes', '4876543210', '32109876543', 'larissa.mendes@hotmail.com'),
```

```
('Rafael Dias', '4865432109', '45609876543', 'rafael.dias@gmail.com'),
```

('Camila Silva', '4854321098', '65409876543', 'camila.silva@hotmail.com'),
('Rodrigo Santos', '4843210987', '78909876543', 'rodrigo.santos@gmail.com'),
('Beatriz Almeida', '4832109876', '98709876543', 'beatriz.almeida@hotmail.com'),
('Marcelo Carvalho', '4821098765', '11133355577', 'marcelo.carvalho@gmail.com'),
('Renata Costa', '4810987654', '22244466688', 'renata.costa@hotmail.com'),
('Ana Maria', '11987654321', '12345678901', 'ana@example.com'),
('Carlos Silva', '21987654321', '23456789012', 'carlos@example.com'),
('Beatriz Sousa', '31987654321', '34567890123', 'beatriz@example.com'),
('Daniel Costa', '41987654321', '45678901234', 'daniel@example.com'),
('Eliana Torres', '51987654321', '56789012345', 'eliana@example.com'),
('Fernando Almeida', '61987654321', '67890123456', 'fernando@example.com'),
('Gabriela Lima', '71987654321', '78901234567', 'gabriela@example.com'),
('Henrique Oliveira', '81987654321', '89012345678', 'henrique@example.com'),
('Isabel Martins', '91987654321', '90123456789', 'isabel@example.com'),
('João Pereira', '10198765432', '01234567890', 'joao@example.com');

INSERT INTO [produto] ([nome_produto], [preco])

VALUES

('Bolo de Chocolate', 10.00),
('Torta de Morango', 15.50),
('Pão de Mel', 3.50),

```
('Brigadeiro', 2.00),  
( 'Pudim de Leite', 7.00),  
( 'Cookie de Chocolate', 8.50),  
( 'Trufa de Chocolate', 2.50),  
( 'Brownie', 7.00),  
( 'Cheesecake', 13.50),  
( 'Bolo de Cenoura', 9.00);
```

```
INSERT INTO [mesa] ([numero], [capacidade])
```

```
VALUES
```

```
(1, 4),  
(2, 2),  
(3, 6),  
(4, 4),  
(5, 2),  
(6, 8),  
(7, 4),  
(8, 2),  
(9, 6),  
(10, 4);
```

```
INSERT INTO [pedido] ([id_cliente], [id_mesa], [dt_pedido]) VALUES
```

(1, 2, '2024-06-19'),
(3, 2, '2024-06-19'),
(13, 6, '2024-06-19'),
(1, 2, '2024-06-19'),
(16, 3, '2024-06-19'),
(14, 2, '2024-06-19'),
(3, 1, '2024-06-19'),
(15, 10, '2024-06-19'),
(12, 3, '2024-06-19'),
(20, 3, '2024-06-20'),
(18, 8, '2024-06-20'),
(7, 6, '2024-06-20'),
(10, 5, '2024-06-20'),
(21, 7, '2024-06-20'),
(20, 6, '2024-06-20'),
(4, 10, '2024-06-20'),
(17, 5, '2024-06-20'),
(14, 2, '2024-06-20'),
(22, 3, '2024-06-20'),
(16, 6, '2024-06-21'),
(5, 1, '2024-06-21'),
(27, 1, '2024-06-21'),

(14, 9, '2024-06-21'),
(3, 6, '2024-06-21'),
(16, 7, '2024-06-21'),
(7, 4, '2024-06-21'),
(21, 6, '2024-06-21'),
(26, 1, '2024-06-21'),
(26, 7, '2024-06-21'),
(12, 8, '2024-06-22'),
(27, 5, '2024-06-22'),
(17, 10, '2024-06-22'),
(17, 5, '2024-06-22'),
(17, 9, '2024-06-22'),
(13, 9, '2024-06-22'),
(21, 5, '2024-06-22'),
(9, 1, '2024-06-22'),
(2, 6, '2024-06-22'),
(22, 6, '2024-06-22'),
(4, 3, '2024-06-23'),
(20, 2, '2024-06-23'),
(2, 6, '2024-06-23'),
(17, 10, '2024-06-23'),
(11, 10, '2024-06-23'),

```
(17, 1, '2024-06-23'),  
(22, 5, '2024-06-23'),  
(12, 4, '2024-06-23'),  
(21, 2, '2024-06-23'),  
(2, 7, '2024-06-23');
```

```
INSERT INTO [pedido_item] ([id_produto], [id_pedido], [valor], [quantidade]) VALUES
```

```
(1, 1, 10.0, 2),  
(6, 1, 8.5, 14),  
(10, 2, 9.0, 7),  
(5, 2, 7.0, 18),  
(7, 3, 2.5, 5),  
(6, 3, 8.5, 1),  
(5, 3, 7.0, 4),  
(7, 4, 2.5, 9),  
(8, 4, 7.0, 1),  
(9, 4, 13.5, 4),  
(1, 4, 10.0, 5),  
(7, 5, 2.5, 3),  
(5, 5, 7.0, 1),  
(1, 6, 10.0, 1),
```

(3, 7, 3.5, 17),

(7, 7, 2.5, 1),

(10, 7, 9.0, 2),

(1, 8, 10.0, 1),

(9, 9, 13.5, 1),

(4, 9, 2.0, 14),

(1, 9, 10.0, 1),

(3, 10, 3.5, 3),

(10, 10, 9.0, 6),

(4, 10, 2.0, 15),

(8, 10, 7.0, 7),

(10, 11, 9.0, 7),

(2, 11, 15.5, 9),

(9, 11, 13.5, 1),

(4, 11, 2.0, 13),

(5, 12, 7.0, 12),

(10, 12, 9.0, 7),

(1, 12, 10.0, 9),

(1, 13, 10.0, 3),

(8, 13, 7.0, 5),

(7, 13, 2.5, 5),

(4, 13, 2.0, 9),

(7, 14, 2.5, 6),
(8, 14, 7.0, 3),
(4, 15, 2.0, 1),
(1, 16, 10.0, 3),
(7, 16, 2.5, 5),
(3, 16, 3.5, 4),
(5, 17, 7.0, 4),
(8, 17, 7.0, 4),
(3, 17, 3.5, 8),
(7, 17, 2.5, 1),
(9, 17, 13.5, 1),
(8, 18, 7.0, 1),
(2, 18, 15.5, 1),
(10, 18, 9.0, 2),
(6, 18, 8.5, 5),
(6, 19, 8.5, 6),
(9, 19, 13.5, 4),
(7, 20, 2.5, 1),
(6, 20, 8.5, 4),
(5, 21, 7.0, 6),
(2, 21, 15.5, 6),
(1, 22, 10.0, 22),

(2, 22, 15.5, 4),
(1, 23, 10.0, 2),
(2, 23, 15.5, 1),
(9, 23, 13.5, 5),
(10, 23, 9.0, 10),
(8, 24, 7.0, 9),
(9, 24, 13.5, 4),
(10, 25, 9.0, 11),
(2, 25, 15.5, 3),
(3, 26, 3.5, 5),
(10, 26, 9.0, 1),
(1, 26, 10.0, 4),
(5, 26, 7.0, 10),
(8, 26, 7.0, 7),
(3, 27, 3.5, 9),
(4, 28, 2.0, 30),
(7, 28, 2.5, 1),
(2, 28, 15.5, 1),
(3, 28, 3.5, 4),
(7, 29, 2.5, 11),
(4, 29, 2.0, 4),
(3, 29, 3.5, 2),

(6, 29, 8.5, 2),
(7, 30, 2.5, 9),
(10, 31, 9.0, 23),
(5, 31, 7.0, 5),
(4, 31, 2.0, 36),
(8, 31, 7.0, 6),
(6, 31, 8.5, 3),
(2, 32, 15.5, 4),
(1, 32, 10.0, 3),
(3, 32, 3.5, 2),
(6, 32, 8.5, 13),
(6, 33, 8.5, 6),
(3, 33, 3.5, 23),
(4, 34, 2.0, 1),
(9, 34, 13.5, 1),
(2, 34, 15.5, 23),
(2, 35, 15.5, 1),
(7, 35, 2.5, 17),
(10, 35, 9.0, 1),
(4, 35, 2.0, 1),
(8, 36, 7.0, 7),
(5, 37, 7.0, 22),

(9, 37, 13.5, 7),

(3, 38, 3.5, 3),

(8, 38, 7.0, 5),

(5, 39, 7.0, 2),

(5, 40, 7.0, 1),

(1, 40, 10.0, 25),

(2, 41, 15.5, 13),

(3, 42, 3.5, 1),

(10, 42, 9.0, 5),

(2, 42, 15.5, 6),

(6, 42, 8.5, 9),

(1, 42, 10.0, 17),

(3, 43, 3.5, 6),

(2, 43, 15.5, 1),

(8, 43, 7.0, 12),

(8, 44, 7.0, 3),

(7, 44, 2.5, 17),

(1, 45, 10.0, 1),

(5, 46, 7.0, 8),

(3, 47, 3.5, 13),

(4, 48, 2.0, 16),

(1, 48, 10.0, 1),

```
(1, 49, 10.0, 1),
```

```
(4, 49, 2.0, 11),
```

```
(3, 49, 3.5, 1),
```

```
(6, 49, 8.5, 4);
```

```
INSERT INTO [forma_de_pagamento] ([tp_pagamento])
```

```
VALUES
```

```
('Cartão de Crédito'),
```

```
('Dinheiro'),
```

```
('Cartão de Débito'),
```

```
('Pix'),
```

```
('Vale Refeição'),
```

```
('Transferência Bancária'),
```

```
('Boleto'),
```

```
('Crédito Loja');
```

```
INSERT INTO [pagamento] ([id_pedido], [total], [id_forma_pagamento], [status_pagamento]) VALUES
```

```
(1, 139.00, 6, 1),
```

```
(2, 189.00, 5, 1),
```

```
(3, 49.00, 5, 1),
```

(4, 173.50, 4, 1),

(5, 14.50, 3, 1),

(6, 10.00, 2, 1),

(7, 80.00, 5, 1),

(8, 10.00, 7, 1),

(9, 51.50, 7, 1),

(10, 143.50, 7, 1),

(11, 242.00, 4, 1),

(12, 237.00, 5, 1),

(13, 95.50, 7, 1),

(14, 36.00, 7, 1),

(15, 2.00, 6, 1),

(16, 56.50, 6, 1),

(17, 100.00, 4, 1),

(18, 83.00, 2, 1),

(19, 105.00, 6, 1),

(20, 36.50, 1, 1),

(21, 135.00, 4, 1),

(22, 282.00, 6, 1),

(23, 193.00, 7, 1),

(24, 117.00, 3, 1),

(25, 145.50, 4, 1),

(26, 185.50, 1, 1),
(27, 31.50, 2, 1),
(28, 92.00, 4, 1),
(29, 59.50, 2, 1),
(30, 22.50, 6, 1),
(31, 381.50, 6, 1),
(32, 209.50, 1, 1),
(33, 131.50, 4, 1),
(34, 372.00, 1, 1),
(35, 69.00, 1, 1),
(36, 49.00, 1, 1),
(37, 248.50, 1, 1),
(38, 45.50, 3, 1),
(39, 14.00, 7, 1),
(40, 257.00, 6, 1),
(41, 201.50, 2, 1),
(42, 388.00, 7, 1),
(43, 120.50, 7, 1),
(44, 63.50, 6, 1),
(45, 10.00, 3, 1),
(46, 56.00, 5, 1),
(47, 45.50, 4, 1),

(48, 42.00, 3, 1),

(49, 69.50, 5, 1);

INSERT INTO [produzido] ([id_produto], [data], [quantidade])

VALUES

-- Dia 19 - qua

(1, '2024-06-19', 13), (2, '2024-06-19', 10), (3, '2024-06-19', 20),

(4, '2024-06-19', 30), (5, '2024-06-19', 23), (6, '2024-06-19', 15),

(7, '2024-06-19', 18), (8, '2024-06-19', 10), (9, '2024-06-19', 6),

(10, '2024-06-19', 19),

-- Dia 20 - qui

(1, '2024-06-20', 18), (2, '2024-06-20', 10), (3, '2024-06-20', 24),

(4, '2024-06-20', 32), (5, '2024-06-20', 22), (6, '2024-06-20', 17),

(7, '2024-06-20', 18), (8, '2024-06-20', 13), (9, '2024-06-20', 8),

(10, '2024-06-20', 17),

-- Dia 21 - sex

(1, '2024-06-21', 28), (2, '2024-06-21', 15), (3, '2024-06-21', 26),

(4, '2024-06-21', 35), (5, '2024-06-21', 28), (6, '2024-06-21', 20),

(7, '2024-06-21', 22), (8, '2024-06-21', 17), (9, '2024-06-21', 10),

(10, '2024-06-21', 22),

-- Dia 22 - sab

```
(1, '2024-06-22', 30), (2, '2024-06-22', 28), (3, '2024-06-22', 28),  
(4, '2024-06-22', 38), (5, '2024-06-22', 30), (6, '2024-06-22', 22),  
(7, '2024-06-22', 24), (8, '2024-06-22', 18), (9, '2024-06-22', 12),  
(10, '2024-06-22', 24),
```

-- Dia 23 - dom

```
(1, '2024-06-23', 22), (2, '2024-06-23', 20), (3, '2024-06-23', 22),  
(4, '2024-06-23', 32), (5, '2024-06-23', 24), (6, '2024-06-23', 18),  
(7, '2024-06-23', 20), (8, '2024-06-23', 15), (9, '2024-06-23', 9),  
(10, '2024-06-23', 20);
```

Principais consultas mapeadas baseadas em regras de negócio

-- Perguntas de negócio:

--1. Qual o ticket médio do dia 19/06/2024 a 23/06/2024 ?

```
SELECT dbo.calcular_ticket_medio('2024-06-19', '2024-06-23') AS TicketMedioJunho;
```

--2. Quais foram os dias e a quantidade de produtos do mês de junho 2024 que tiveram sobra na produção?

```
WITH vendas_por_dia AS (
```

```
    SELECT
```

```
        pi.id_produto,
```



```
        nome_produto,
        p.dt_pedido,
        SUM(pi.quantidade) AS total_vendido
FROM pedido_item pi
JOIN pedido p ON pi.id_pedido = p.id_pedido
        INNER JOIN produto pr ON pi.id_produto = pr.id_produto
GROUP BY pi.id_produto, p.dt_pedido, nome_produto
),
sobra_produtos AS (
    SELECT
        p.id_produto,
        pr.nome_produto,
        p.data,
        p.quantidade AS quantidade_produzida,
        COALESCE(vpd.total_vendido, 0) AS quantidade_vendida,
        (p.quantidade - COALESCE(vpd.total_vendido, 0)) AS sobra
    FROM produzido p
    LEFT JOIN vendas_por_dia vpd ON p.id_produto = vpd.id_produto AND p.data = vpd.dt_pedido
        INNER JOIN produto pr ON p.id_produto = pr.id_produto
)
SELECT
    data,
```

```
id_produto,  
    nome_produto,  
    quantidade_produzida,  
    sobra  
FROM sobra_produtos  
WHERE sobra > 0  
ORDER BY id_produto;
```

--3. Qual foi o produto mais consumido? e o menos consumido?

```
WITH soma AS (  
    SELECT  
        id_produto,  
        SUM(quantidade) AS qt  
    FROM pedido_item  
    GROUP BY id_produto  
) , rankado AS(  
    SELECT  
        id_produto,  
        qt,  
        DENSE_RANK() OVER(ORDER BY qt) as rank_menos,  
        DENSE_RANK() OVER(ORDER BY qt desc) as rank_mais  
    FROM soma
```

```

)
SELECT
    ra.id_produto,
    pr.nome_produto,
    ra.qt AS quantidade_vendida,
    CASE rank_mais
        WHEN 1 THEN 'Mais vendida'
        ELSE 'Menos vendida'
    END AS tipo
FROM
    ranqueado ra
INNER JOIN produto pr ON pr.id_produto = ra.id_produto
WHERE rank_menos = 1 or rank_mais = 1;

```

--4. Qual a forma de pagamento mais utilizada? quantidade de pedidos que usaram essa forma, valor total?

```

WITH total AS (
    SELECT
        pa.id_forma_pagamento,
        (
            SELECT tp_pagamento
            FROM forma_de_pagamento
            WHERE id_forma_pagamento = pa.id_forma_pagamento
        )
    FROM pedido pa
)
SELECT
    total.id_forma_pagamento,
    total.tp_pagamento,
    COUNT(*) AS quantidade_pedidos,
    SUM(pa.valor) AS valor_total
FROM total
GROUP BY total.id_forma_pagamento, total.tp_pagamento;

```

```

    ) AS tp_forma_pagamento,
    COUNT(*) AS qtde_pedidos,
    SUM(pa.total) AS valor_total,
    DENSE_RANK() OVER(ORDER BY COUNT(*) desc) as rank
FROM
    pagamento pa
GROUP BY id_forma_pagamento
)
SELECT
    id_forma_pagamento,
    tp_forma_pagamento,
    qtde_pedidos,
    valor_total
FROM total
WHERE rank = 1;

```

Trigger, Procedure e Function

-- TRIGGER: Atualizar Total de Pagamento

```

CREATE OR ALTER TRIGGER atualizar_total_pagamento ON pedido_item
AFTER INSERT AS

```

BEGIN

IF (ROWCOUNT_BIG() = 0)

RETURN;

DECLARE

@count_pedido INT,

@total_atualizado DECIMAL(10,2),

@id_pedido INT,

@valor DECIMAL(10,2),

@quantidade INT

SELECT

@id_pedido = id_pedido, @valor = valor, @quantidade = quantidade

FROM inserted

-- Calcular o total do novo pedido

SELECT @total_atualizado = @valor * @quantidade

-- Verificar se o pagamento já existe para o pedido

SELECT @count_pedido = COUNT(id_pedido)

FROM pagamento

WHERE id_pedido = @id_pedido;

```
-- Se o pagamento existir, atualizar o total

IF @count_pedido > 0

    BEGIN

        UPDATE pagamento

        SET total = @total_atualizado + total

        WHERE id_pedido = @id_pedido;

    END

-- Se não existir, inserir um novo registro de pagamento

ELSE

    BEGIN

        INSERT INTO pagamento (id_pedido, total, status_pagamento)

        VALUES (@id_pedido, @total_atualizado, 0);

    END

END

GO


-- PROCEDURE: Registrar Produção Diária

CREATE OR ALTER PROCEDURE registrar_producao_diaria(@id_produto INT, @data DATE, @quantidade INT) AS

BEGIN

    DECLARE @producao_existente INT;

    -- Verificar se já existe um registro para esse produto na data especificada
```

```
        SELECT @producao_existente = quantidade
FROM produzido
WHERE id_produto = @id_produto AND @data = data

-- Se existir, atualizar a quantidade
IF @producao_existente > 0
    BEGIN
        UPDATE produzido
        SET quantidade = @producao_existente + @quantidade
        WHERE id_produto = @id_produto AND @data = data
    END
ELSE
    BEGIN
        INSERT INTO produzido (id_produto, data, quantidade)
        VALUES (@id_produto, @data, @quantidade)
    END
END
GO
```

-- FUNCTION: Calcular Ticket Médio

```
CREATE OR ALTER FUNCTION calcular_ticket_medio(@data_inicio DATE, @data_fim DATE) RETURNS DECIMAL(10,2) AS
BEGIN
```

```
DECLARE @ticket_medio DECIMAL(10,2);

SELECT @ticket_medio = AVG(total)

FROM pedido p

JOIN pagamento pm ON p.id_pedido = pm.id_pedido

WHERE p.dt_pedido BETWEEN @data_inicio AND @data_fim AND pm.status_pagamento = 1; -- Considera apenas os pedidos pagos

RETURN @ticket_medio

END

GO
```