INSTITUTO FEDERAL DA PARAÍBA - IFPB
Unidade Acadêmica de Informação e Comunicação
CST em Sistemas para Internet

Disciplina: Banco de Dados II – 2023.1

Professores: Damires e Thiago

Grupo: _____

Roteiro para Projeto de Banco de Dados Relacional (Escopo e início)

**** Monte sua equipe: 2 integrantes**

1. **Descreva o escopo de aplicação que precisa de um banco de dados (seu projeto).**
    a. Descrição geral das regras do negócio e do que se espera da aplicação/banco de dados

Título -> Sistema para aplicativo de Delivery.

Contexto -> O RAPFOOD é um aplicativo de delivery que tem como objetivo controlar o processo de um aplicativo de delivery tradicional, desde o cadastro dos clientes, estabelecimentos e entregadores até a realização dos pedidos.

O sistema a ser criado, será responsável pelo cadastramento e gerenciamento dos pedidos. Para que o cliente comece a usar os serviços disponíveis na plataforma, é necessário que o cliente realize o cadastro, informando seus dados pessoais, como CPF, pelo menos 1 telefone,nome, email, endereço completo, com número, bairro,rua, cidade e estado.

Para que o estabelecimento comece a receber pedidos, é necessário que ele faça o cadastro no aplicativo, precisando do seu CNPJ, nome fantasia, telefone, e endereço completo, com número,rua, bairro, cidade e estado.

E também, para que o entregador retire os pedidos, é obrigatório o registro dele na plataforma, precisando do seu CPF, nome, conta bancária (número da conta e a agência), e tipo do veículo.

Os cliente podem realizar vários pedidos no aplicativo, porém cada pedido é realizado por apenas um cliente, pois cada pedido é único. Esse pedido é identificado por meio de um código identificador, e ainda possui a data e forma de pagamento,e o valor total do pedido.

Cada pedido contém um ou vários itens na qual é identificado pelo seu código e também possui a descrição do produto.

O estabelecimento é responsável por fornecer um ou vários pedidos, e cada pedido é fornecido por apenas um estabelecimento. Após o pedido ser fornecido pelo estabelecimento, o entregador retira um ou vários pedidos. Depois que ele coleta esse ou esses pedidos, é gerado a sua rota.

b. Requisitos Funcionais da aplicação

Requisitos funcionais para clientes :

1 - Cadastrar cliente.

2 - Realizar pedido.

3 - Adicionar itens no pedido.

4 - Adicionar novo número telefônico.

5 - Apresentar uma lista dos estabelecimentos, o produto e o valor dele.

6 - Resgatar as informações do pedido que o cliente fez, que está incluso o código do pedido, data do pedido, nome do cliente, forma de pagamento, nome do estabelecimento associado ao pedido e o nome do entregador.

7 - Resgatar a disponibilidade de produtos nos estabelecimentos juntamente com sua quantidade e sua descrição.

8 - Resgatar o valor total de gastos de um cliente.

Requisitos funcionais para entregadores:

1 - Cadastrar entregador.

2 - Resgatar quantos pedidos estão atrelados ao cpf dele em um determinado período de tempo.

Requisitos funcionais para estabelecimentos:

1 - Cadastrar estabelecimento.

2 - Adicionar itens no estoque.

3 - Resgatar o valor total de faturamento em um mês específico.

Requisitos funcionais *para equipe de gestão do aplicativo:*

1 - Cadastrar produto

2 - Consultar quantidade de vendas dos dias 13 e 14 de novembro.

3 - Obter uma lista de todos os clientes atrelados aos seus números telefônicos.

4 - Resgatar o produto mais vendido do aplicativo junto com a quantidade de vendas que ele está incluso.

5 - Resgatar os 15 estabelecimentos que mais venderam, junto com a quantidade de cada um .

6 - Consulta para resgatar os Estados que possuem menos de 5 clientes.

7 - Consulta para identificar as formas de pagamento mais usadas e suas quantidades para cada pedido feito.

8 - Consulta para identificar CPFs que nunca fizeram pedidos.

9 - Consulta para identificar os nomes juntamente com os CPFs dos clientes que fizeram compras no mês de Outubro.

10 - Consulta para resgatar os produtos que estão em estoque, mas não estão em nenhum pedido.

11 - Consultar os clientes que moram na cidade de João Pessoa no Estado da Paraíba.

c. Requisitos de Dados

Cliente

Estabelecimento

Entregador

Produto

Pedido

Itens do pedido

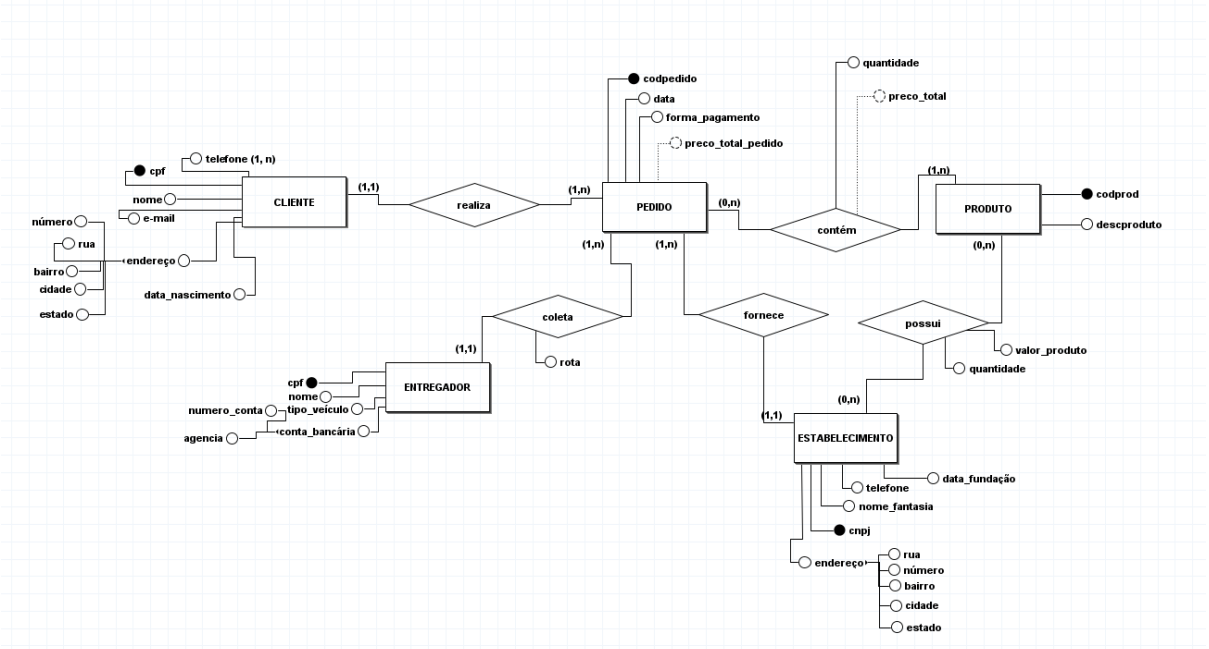
Estoque

Telefones dos clientes

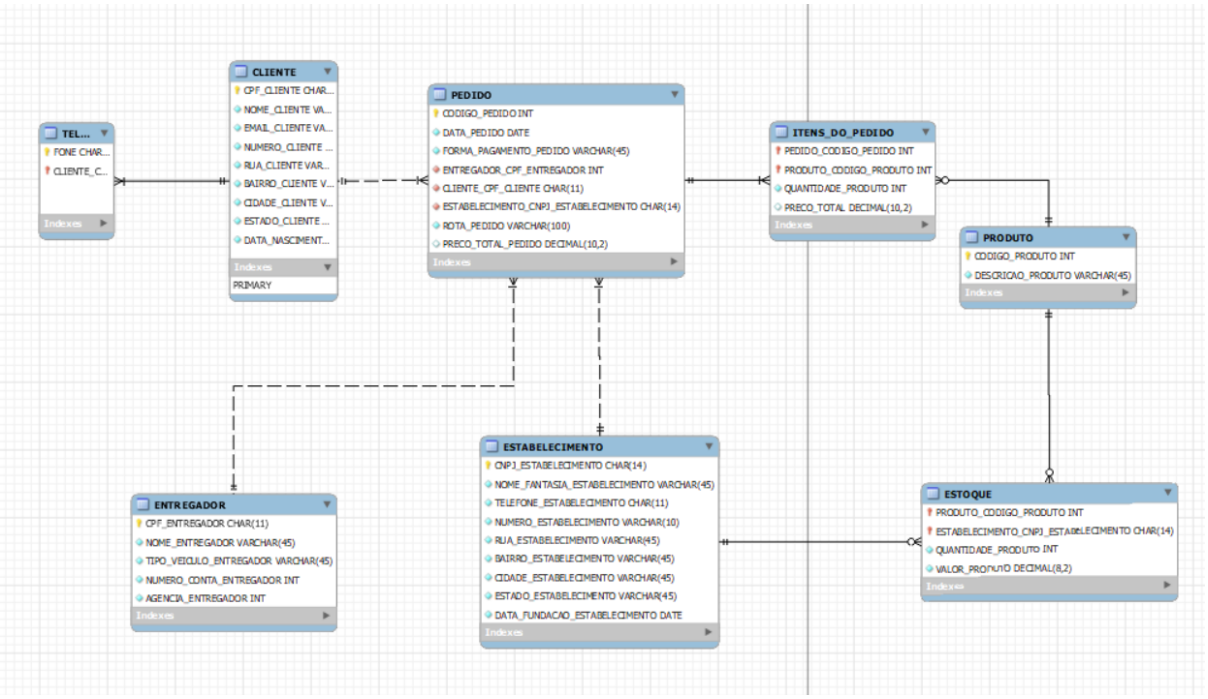
** Veja o exemplo de descrição de aplicação mostrado/anexado.

2. Diagrama Entidade-Relacionamento em nível Conceitual (versão atualizada)

Especifique uma versão inicial do seu DER conceitual com as entidades, relacionamentos e principais atributos. As entidades e relacionamentos devem ser especificados conforme escopo e regras de negócio descritas na seção anterior.



3. Diagrama do nível lógico (Atualizado)



4. Script (OBSERVAÇÃO IMPORTANTE) - >

1 PASSO: CRIAÇÃO DAS TABELAS

2 PASSO : CRIAÇÃO DAS TRIGGERS

3 PASSO : POVOAMENTO DAS TABELAS

4 PASSO : DEMAIS COMANDOS

/* 2. Implementação do projeto de BDR no SGBD PostgreSQL

**** Todas as operações devem apresentar seu enunciado e sua solução.**

**** Os comandos (todos) devem fazer sentido à aplicação e a seus requisitos. */**

-- a. Criação e uso de objetos básicos (12,0):

-- AVISO MUITO IMPORTANTE !!!!!

-- 01 PASSO : CRIAÇÃO DAS TABELAS

-- 02 PASSO : CRIAÇÃO DOS TRIGGERS

-- 03 PASOO : POVOAMENTO DAS TABELAS

-- 04 : DEMAIS COMANDOS

-- IMPORTANTE !!!!!!!!!!!!!!!!!!!!!!!!!!!!

-- Criação da tabela cliente

```
-----------------------------------------------------------------------------------------------------

CREATE TABLE cliente (
    cpf_cliente CHAR(11) PRIMARY KEY,
    nome_cliente VARCHAR(45) NOT NULL,
    email_cliente VARCHAR(45) NOT NULL,
    numero_cliente VARCHAR(10) NOT NULL,
    bairro_cliente VARCHAR(45) NOT NULL,
    cidade_cliente VARCHAR(45) NOT NULL,
    estado_cliente CHAR(2) NOT NULL,
    data_nascimento DATE NOT NULL
);

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

-- Criação da tabela estabelecimento

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

CREATE TABLE estabelecimento (
    cnpj_estabelecimento CHAR(14) PRIMARY KEY,
```

```
    nome_fantasia_estabelecimento VARCHAR(45) NOT NULL,
    telefone_estabelecimento CHAR(11) NOT NULL,
    numero_estabelecimento VARCHAR(10) NOT NULL,
    rua_estabelecimento VARCHAR(45) NOT NULL,
    bairro_estabelecimento VARCHAR(45) NOT NULL,
    cidade_estabelecimento VARCHAR(45) NOT NULL,
    estado_estabelecimento VARCHAR(45) NOT NULL,
    data_fundacao_estabelecimento DATE NOT NULL
);

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

-- Criação da tabela produto

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

CREATE TABLE produto (
    codigo_produto INT PRIMARY KEY,
    descricao_produto VARCHAR(45) NOT NULL
);
ALTER TABLE produto
ADD CONSTRAINT unique_descricao_produto UNIQUE (descricao_produto);
```

```
-- Criação da tabela entregador


CREATE TABLE entregador (
    cpf_entregador CHAR(11) PRIMARY KEY,
    nome_entregador VARCHAR(45) NOT NULL,
    tipo_veiculo VARCHAR(45) NOT NULL,
    numero_conta INT NOT NULL UNIQUE,
    agencia INT NOT NULL
);


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


-- Criação da tabela telefone do cliente


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


CREATE TABLE telefone (
    numero_telefone CHAR(11) ,
    cpf_cliente CHAR(11) ,
    PRIMARY KEY (numero_telefone, cpf_cliente),
    FOREIGN KEY (cpf_cliente) REFERENCES cliente(cpf_cliente)
```

```
);

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

-- Criação da tabela estoque

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

CREATE TABLE estoque (
    cod_produto INT,
    cnpj_estabelecimento CHAR(14),
    quantidade INT NOT NULL,
    valor_produto DECIMAL(8,2) not null,
    PRIMARY KEY (cod_produto, cnpj_estabelecimento),
    FOREIGN KEY (cod_produto) REFERENCES produto(codigo_produto),
    FOREIGN KEY (cnpj_estabelecimento) REFERENCES
      estabelecimento(cnpj_estabelecimento)
);

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

-- Criação da tabela pedido
```

```
-----------------------------------------------------------------------------------------------------

CREATE TABLE pedido (

    codigo_pedido INT PRIMARY KEY,

    data_pedido DATE NOT NULL,

    forma_pagamento VARCHAR(45) NOT NULL,

    cpf_entregador CHAR(11) NOT NULL,

    cpf_cliente CHAR(11) NOT NULL,

    cnpj_estabelecimento CHAR(14) NOT NULL,

    rota_pedido VARCHAR(45),

    FOREIGN KEY (cpf_entregador) REFERENCES entregador(cpf_entregador),

    FOREIGN KEY (cpf_cliente) REFERENCES cliente(cpf_cliente),

    FOREIGN KEY (cnpj_estabelecimento) REFERENCES
      estabelecimento(cnpj_estabelecimento)

);

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

-- Criação da tabela Itens do Pedido
```

```
-----------------------------------------------------------------------------------------------------

CREATE TABLE itens_do_pedido (
    codigo_pedido INT,
    codigo_produto INT,
    quantidade INT NOT NULL, -- Trigger before insert on itens_pedido
    preco_total numeric(10,2),
    PRIMARY KEY (codigo_pedido, codigo_produto),
    FOREIGN KEY (codigo_pedido) REFERENCES pedido(codigo_pedido),
    FOREIGN KEY (codigo_produto) REFERENCES produto(codigo_produto)
);

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

-- Popular as tabelas ---

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

-- Cliente

INSERT INTO cliente (cpf_cliente, nome_cliente, email_cliente, numero_cliente,
    bairro_cliente, cidade_cliente, estado_cliente, data_nascimento)
```

VALUES

('11111111111', 'João Silva', 'joao@example.com', '123456789', 'Centro', 'São Paulo', 'SP', '1990-05-15'),

('22222222222', 'Maria Oliveira', 'maria@example.com', '987654321', 'Jardim', 'Rio de Janeiro', 'RJ', '1985-08-22'),

('33333333333', 'Carlos Santos', 'carlos@example.com', '654321987', 'Vila', 'Belo Horizonte', 'MG', '1988-12-03'),

('44444444444', 'Ana Pereira', 'ana@example.com', '876543210', 'Parque', 'Porto Alegre', 'RS', '1995-02-28'),

('55555555555', 'Pedro Souza', 'pedro@example.com', '234567890', 'Liberdade', 'Recife', 'PE', '1980-11-10'),

('66666666666', 'Fernanda Lima', 'fernanda@example.com', '678901234', 'Copacabana', 'Fortaleza', 'CE', '1992-07-07'),

('77777777777', 'Ricardo Oliveira', 'ricardo@example.com', '345678901', 'Centro', 'Salvador', 'BA', '1983-04-18'),

('88888888888', 'Larissa Silva', 'larissa@example.com', '789012345', 'Barra', 'Curitiba', 'PR', '1998-09-25'),

('99999999999', 'Gabriel Santos', 'gabriel@example.com', '567890123', 'Santo Antônio', 'Manaus', 'AM', '1987-06-05'),

('10101010101', 'Juliana Lima', 'juliana@example.com', '901234567', 'Jardins', 'Brasília', 'DF', '1993-03-15'),

('23452421004', 'Amanda Rocha', 'amanda@example.com', '905678901', 'Vila Madalena', 'São Paulo', 'SP', '1988-09-28'),

('89012345005', 'Pedro Oliveira', 'pedro@example.com', '906789012', 'Ipanema', 'Rio de Janeiro', 'RJ', '1992-04-14'),

('56789012306', 'Isabela Martins', 'isabela@example.com', '907890123', 'Boa Viagem', 'Recife', 'PE', '1980-12-01'),

('12345678907', 'Renato Costa', 'renato@example.com', '908901234', 'Centro', 'Belo Horizonte', 'MG', '1995-07-18'),

('34567890108', 'Carolina Silva', 'carolina@example.com', '909012345', 'Itaim Bibi', 'São Paulo', 'SP', '1987-02-25'),

('67890123409', 'Matheus Santos', 'matheus@example.com', '910123456', 'Leblon', 'Rio de Janeiro', 'RJ', '1998-11-03'),

('89012345610', 'Ana Souza', 'ana@example.com', '911234567', 'Ponta Negra', 'Natal', 'RN', '1983-06-09'),

('01234567811', 'Lucas Oliveira', 'lucas@example.com', '912345678', 'Barra', 'Salvador', 'BA', '1991-01-22'),

('23456789012', 'Beatriz Lima', 'beatriz@example.com', '913456789', 'Jardim Europa', 'São Paulo', 'SP', '1986-08-07'),

('45678901213', 'Gustavo Pereira', 'gustavo@example.com', '914567890', 'Lapa', 'Rio de Janeiro', 'RJ', '1994-03-31'),

('67890123414', 'Julia Costa', 'julia@example.com', '915678901', 'Piedade', 'Recife', 'PE', '1989-10-15'),

('89012345615', 'Roberto Martins', 'roberto@example.com', '916789012', 'Centro', 'Belo Horizonte', 'MG', '1997-05-04'),

('12345678916', 'Fernanda Oliveira', 'fernanda@example.com', '917890123', 'Boa Viagem', 'Recife', 'PE', '1984-12-20'),

('34567890117', 'Ricardo Santos', 'ricardo@example.com', '918901234', 'Ipanema', 'Rio de Janeiro', 'RJ', '1993-07-03'),

('56789012318', 'Mariana Lima', 'mariana@example.com', '919012345', 'Vila Madalena', 'São Paulo', 'SP', '1981-02-16'),

('78901234519', 'Gabriel Silva', 'gabriel@example.com', '920123456', 'Barra', 'Salvador', 'BA', '1996-09-09'),

('01234567820', 'Larissa Souza', 'larissa@example.com', '921234567', 'Leblon', 'Rio de Janeiro', 'RJ', '1982-04-26'),

('88888888899', 'Larissa Silva', 'larissa@example.com', '789012345', 'Barra', 'Curitiba', 'PR', '1998-09-25'),

('21341598765', 'João Silva', 'joao.silva@example.com', '123456789', 'Mangabeira', 'João Pessoa', 'PB', '2000-01-01'),

```
('21341512345', 'Maria Oliveira', 'maria.oliveira@example.com', '234567890',
   'Bancários', 'João Pessoa', 'PB', '1995-05-15'),

('21341567890', 'Pedro Santos', 'pedro.santos@example.com', '345678901',
   'Altiplano', 'João Pessoa', 'PB', '1987-11-30'),

('21341543210', 'Ana Pereira', 'ana.pereira@example.com', '456789012',
   'Mandacaru', 'João Pessoa', 'PB', '1999-07-22'),

('21341565432', 'Lucas Souza', 'lucas.souza@example.com', '567890123',
   'Manaíra', 'João Pessoa', 'PB', '1982-03-10'),

('21341587654', 'Juliana Costa', 'juliana.costa@example.com', '678901234', 'Torre',
   'João Pessoa', 'PB', '1990-09-05'),

('21341534567', 'Rafael Lima', 'rafael.lima@example.com', '789012345', 'Tambaú',
   'João Pessoa', 'PB', '1985-04-18'),

('21341598701', 'Carolina Fernandes', 'carolina.fernandes@example.com',
   '890123456', 'Cristo Redentor', 'João Pessoa', 'PB', '1997-12-03'),

('21341501234', 'Diego Oliveira', 'diego.oliveira@example.com', '901234567',
   'Jaguaribe', 'João Pessoa', 'PB', '1993-08-15'),

('21341587698', 'Fernanda Santos', 'fernanda.santos@example.com', '012345678',
   'Centro', 'João Pessoa', 'PB', '1988-06-27');



select * from cliente;



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



-- Estabelecimento



-----------------------------------------------------------------------------------------------------
```

```
INSERT INTO estabelecimento (cnpj_estabelecimento,
    nome_fantasia_estabelecimento, telefone_estabelecimento,
    numero_estabelecimento, rua_estabelecimento, bairro_estabelecimento,
    cidade_estabelecimento, estado_estabelecimento,
    data_fundacao_estabelecimento)

VALUES

 ('11111111111111', 'Bom Gosto Empório', '1122334455', '123', 'Rua das Delícias',
    'Centro', 'São Paulo', 'SP', '2000-01-15'),

 ('22222222222222', 'Sabor Natural Market', '2233445566', '456', 'Avenida dos
    Sabores', 'Jardim', 'Rio de Janeiro', 'RJ', '1995-08-22'),

 ('33333333333333', 'Aromas da Culinária', '3344556677', '789', 'Rua do Sabor',
    'Vila', 'Belo Horizonte', 'MG', '2005-12-03'),

 ('44444444444444', 'Central de Sabores', '4455667788', '101', 'Avenida da
    Degustação', 'Parque', 'Porto Alegre', 'RS', '2010-02-28'),

 ('55555555555555', 'Gastronomia Express', '5566778899', '202', 'Rua dos Sabores',
    'Liberdade', 'Recife', 'PE', '1992-11-10'),

 ('66666666666666', 'Paladar Gourmet', '6677889900', '303', 'Avenida do Paladar',
    'Copacabana', 'Fortaleza', 'CE', '2015-07-07'),

 ('77777777777777', 'Bistrô do Sabor', '7788990011', '404', 'Rua do Saboroso',
    'Centro', 'Salvador', 'BA', '2008-04-18'),

 ('88888888888888', 'Taste Haven', '8899001122', '505', 'Avenida dos Aromas',
    'Barra', 'Curitiba', 'PR', '2012-09-25'),

 ('99999999999999', 'Sabores da Terra', '9900112233', '606', 'Rua dos Sabores
    Locais', 'Santo Antônio', 'Manaus', 'AM', '2018-06-05'),

 ('10101010101010', 'Gastronomia Plural', '1011122233', '707', 'Avenida das
    Especiarias', 'Jardins', 'Brasília', 'DF', '2003-03-15');

insert into estabelecimento values

 ('10101010101011', 'Cantinho Delicia', '1011122232', '708', 'Avenida das
    Especiarias', 'Jardins', 'Paraíba', 'PB', '2007-03-15');

INSERT INTO estabelecimento VALUES
```

('98765432101234', 'Sabor do Oriente', '3030301234', '101', 'Avenida das Especiarias', 'Centro', 'Recife', 'PE', '2014-09-10'),

('87654321098765', 'Delícias do Sudeste', '4040402345', '202', 'Rua dos Sabores', 'Vila', 'Salvador', 'BA', '2013-08-12'),

('76543210987654', 'Aroma da Montanha', '5050503456', '303', 'Avenida do Sabor Montanhês', 'Copacabana', 'São Paulo', 'SP', '2009-02-28'),

('65432109876543', 'Prazeres Gastronômicos', '6060604567', '404', 'Rua dos Sabores Marítimos', 'Barra', 'Belo Horizonte', 'MG', '2011-11-10'),

('54321098765432', 'Sabores da Natureza', '7070705678', '505', 'Avenida das Ervas', 'Jardins', 'Curitiba', 'PR', '2017-07-07'),

('43210987654321', 'Gastronomia Local', '8080806789', '606', 'Rua das Variedades Gastronômicas', 'Centro', 'Porto Alegre', 'RS', '2010-04-18'),

('32109876543210', 'Sabor Brasileiro', '9090907890', '707', 'Avenida dos Sabores Locais', 'Santo Antônio', 'Brasília', 'DF', '2015-09-25'),

('21098765432109', 'Delícias Regionais', '1010108901', '808', 'Rua do Sabor Variado', 'Liberdade', 'Manaus', 'AM', '2008-06-05'),

('10987654321098', 'Aromas do Cerrado', '1212129012', '909', 'Avenida dos Temperos', 'Sertão', 'Cuiabá', 'MT', '2012-03-15');

select * from estabelecimento;

-- Produto

```
INSERT INTO produto (codigo_produto, descricao_produto)

VALUES

  (1, 'Lasanha'),

  (2, 'Strogonoff de Frango'),

  (3, 'Pizza Margherita'),

  (4, 'Hambúrguer Gourmet'),

  (5, 'Sushi Variado'),

  (6, 'Salada Caesar'),

  (7, 'Camarão ao Molho Branco'),

  (8, 'Massa Carbonara'),

  (9, 'Frango ao Curry'),

  (10, 'Tiramisu'),

  (11, 'Fejoada');



select * from produto;



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



-- Entregador



-----------------------------------------------------------------------------------------------------
```

```
INSERT INTO entregador (cpf_entregador, nome_entregador, tipo_veiculo,
    numero_conta, agencia)

VALUES

 ('11111111112', 'Carlos Silva', 'Motocicleta', 123456789, 1010),

 ('22222222223', 'Mariana Oliveira', 'Bicicleta', 234567890, 2020),

 ('33333333334', 'Lucas Santos', 'Carro', 345678901, 3030),

 ('44444444445', 'Camila Pereira', 'Van', 456789012, 4040),

 ('55555555556', 'Pedro Souza', 'Motocicleta', 567890123, 5050),

 ('66666666667', 'Fernanda Lima', 'Bicicleta', 678901234, 6060),

 ('77777777778', 'Ricardo Oliveira', 'Carro', 789012345, 7070),

 ('88888888889', 'Larissa Silva', 'Van', 890123456, 8080),

 ('99999999990', 'Gabriel Santos', 'Motocicleta', 901234567, 9090),

 ('10101010111', 'Juliana Lima', 'Carro', 112233445, 1010);



select * from entregador;



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



-- Telefone dos clientes



-----------------------------------------------------------------------------------------------------------
```

```
INSERT INTO telefone (numero_telefone, cpf_cliente)

VALUES

  ('11122223333', '11111111111'),

  ('22233334444', '22222222222'),

  ('33344445555', '33333333333'),

  ('44455556666', '44444444444'),

  ('55566667777', '55555555555'),

  ('66677778888', '66666666666'),

  ('77788889999', '77777777777'),

  ('88899990000', '88888888888'),

  ('99900001111', '99999999999'),

  ('00011112222', '10101010101');


select * from telefone;


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


-- Estoque


-----------------------------------------------------------------------------------------------
```

```
-- Bom Gosto Empório

INSERT INTO estoque (cod_produto, cnpj_estabelecimento, quantidade,
    valor_produto)

VALUES

  (1, '11111111111111', 50, 19.99),

  (2, '11111111111111', 30, 15.99),

  (3, '11111111111111', 20, 22.99);

INSERT INTO estoque (cod_produto, cnpj_estabelecimento, quantidade,
    valor_produto)

VALUES

  (4, '11111111111111', 10, 25.99),

  (6, '11111111111111', 25, 20.99),

  (5, '11111111111111', 100, 25.99),

  (7, '11111111111111', 25, 20.99),

  (8, '11111111111111', 100, 25.99),

  (9, '11111111111111', 25, 20.99);




-- Sabor Natural Market

INSERT INTO estoque (cod_produto, cnpj_estabelecimento, quantidade,
    valor_produto)

VALUES
```

```
  (4, '22222222222222', 40, 12.99),

  (5, '22222222222222', 25, 29.99),

  (6, '22222222222222', 35, 18.99),

  (1, '22222222222222', 40, 12.99),

  (2, '22222222222222', 25, 29.99),

  (3, '22222222222222', 35, 18.99),

  (7, '22222222222222', 40, 12.99),

  (8, '22222222222222', 25, 29.99),

  (9, '22222222222222', 35, 18.99);




-- Aromas da Culinária

INSERT INTO estoque (cod_produto, cnpj_estabelecimento, quantidade,
    valor_produto)

VALUES

  (7, '33333333333333', 60, 24.99),

  (8, '33333333333333', 15, 32.99),

  (9, '33333333333333', 40, 27.99),

  (1, '33333333333333', 60, 24.99),

  (2, '33333333333333', 100, 32.99),

  (3, '33333333333333', 40, 27.99),

  (4, '33333333333333', 60, 24.99),
```

```
  (5, '33333333333333', 100, 32.99),

  (6, '33333333333333', 40, 27.99);




-- Central de Sabores

INSERT INTO estoque (cod_produto, cnpj_estabelecimento, quantidade,
      valor_produto)

VALUES

  (10, '44444444444444', 30, 19.99),

  (1, '44444444444444', 25, 15.99),

  (2, '44444444444444', 15, 22.99),

  (3, '44444444444444', 100, 19.99),

  (4, '44444444444444', 100, 15.99),

  (5, '44444444444444', 100, 22.99),

   (6, '44444444444444', 90, 19.99),

  (7, '44444444444444', 60, 15.99),

  (8, '44444444444444', 80, 22.99);




-- Gastronomia Express

INSERT INTO estoque (cod_produto, cnpj_estabelecimento, quantidade,
      valor_produto)

VALUES
```

```
  (3, '55555555555555', 50, 22.99),

  (4, '55555555555555', 20, 12.99),

  (5, '55555555555555', 30, 29.99),

  (1, '55555555555555', 50, 22.99),

  (2, '55555555555555', 20, 12.99),

  (6, '55555555555555', 30, 29.99),

  (7, '55555555555555', 50, 22.99),

  (8, '55555555555555', 20, 12.99),

  (9, '55555555555555', 30, 29.99);



-- Paladar Gourmet

INSERT INTO estoque (cod_produto, cnpj_estabelecimento, quantidade,
    valor_produto)

VALUES

  (6, '66666666666666', 45, 18.99),

  (7, '66666666666666', 10, 24.99),

  (8, '66666666666666', 25, 32.99),

  (9, '66666666666666', 100, 18.99),

  (10, '66666666666666', 100, 24.99),

  (1, '66666666666666', 100, 32.99),

  (2, '66666666666666', 45, 18.99),
```

```
  (3, '66666666666666', 10, 24.99),

  (4, '66666666666666', 25, 32.99);




-- Bistrô do Sabor

INSERT INTO estoque (cod_produto, cnpj_estabelecimento, quantidade,
    valor_produto)

VALUES

  (9, '77777777777777', 35, 27.99),

  (10, '77777777777777', 15, 19.99),

  (1, '77777777777777', 20, 15.99),

  (2, '77777777777777', 100, 27.99),

  (3, '77777777777777', 100, 19.99),

  (4, '77777777777777', 100, 15.99),

  (5, '77777777777777', 100, 27.99),

  (6, '77777777777777', 100, 19.99),

  (7, '77777777777777', 100, 15.99);




-- Taste Haven

INSERT INTO estoque (cod_produto, cnpj_estabelecimento, quantidade,
    valor_produto)

VALUES
```

```
  (2, '88888888888888', 40, 22.99),

  (3, '88888888888888', 25, 12.99),

  (4, '88888888888888', 30, 29.99),

  (5, '88888888888888', 100, 20.99),

  (6, '88888888888888', 100, 10.99),

  (7, '88888888888888', 100, 20.99);



-- Sabores da Terra

INSERT INTO estoque (cod_produto, cnpj_estabelecimento, quantidade,
    valor_produto)

VALUES

  (5, '99999999999999', 55, 29.99),

  (6, '99999999999999', 18, 18.99),

  (7, '99999999999999', 22, 24.99),

  (8, '99999999999999', 100, 29.99),

  (9, '99999999999999', 100, 18.99),

  (10, '99999999999999', 100, 24.99);



-- Gastronomia Plural

INSERT INTO estoque (cod_produto, cnpj_estabelecimento, quantidade,
    valor_produto)

VALUES
```

```
  (8, '10101010101010', 30, 32.99),

  (9, '10101010101010', 25, 27.99),

  (10, '10101010101010', 15, 19.99),

  (11,'10101010101010',20,24.99),

  (1, '10101010101010', 30, 16.89),

  (2, '10101010101010', 25, 15.68),

  (3, '10101010101010', 15, 20.12),

  (4,'10101010101010',20,69.89);




select * from estoque;



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



-- Pedido



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



INSERT INTO pedido (codigo_pedido, data_pedido, forma_pagamento,
    cpf_entregador, cpf_cliente, cnpj_estabelecimento, rota_pedido)

VALUES

  (1, '2023-11-13', 'Cartão de Crédito', '11111111112', '11111111111', '11111111111111',
    NULL),
```

(2, '2023-11-13', 'Pix', '22222222223', '22222222222', '22222222222222', NULL),

(3, '2023-11-13', 'Dinheiro', '33333333334', '33333333333', '33333333333333', NULL),

(4, '2023-11-13', 'Cartão de Débito', '44444444445', '44444444444', '44444444444444', NULL),

(5, '2023-11-13', 'Pix', '55555555556', '55555555555', '55555555555555', NULL),

(6, '2023-11-13', 'Dinheiro', '66666666667', '66666666666', '66666666666666', NULL),

(7, '2023-11-13', 'Cartão de Crédito', '77777777778', '77777777777', '77777777777777', NULL),

(8, '2023-11-13', 'Pix', '88888888889', '88888888888', '88888888888888', NULL),

(9, '2023-11-13', 'Dinheiro', '99999999990', '99999999999', '99999999999999', NULL),

(10, '2023-11-13', 'Cartão de Débito', '10101010111', '10101010101', '10101010101010', NULL);

INSERT INTO pedido (codigo_pedido, data_pedido, forma_pagamento, cpf_entregador, cpf_cliente, cnpj_estabelecimento, rota_pedido)

VALUES (11, '2023-11-14', 'Cartão de Débito', '10101010111', '10101010101', '10101010101010', NULL);

INSERT INTO pedido (codigo_pedido, data_pedido, forma_pagamento, cpf_entregador, cpf_cliente, cnpj_estabelecimento, rota_pedido)

VALUES (12, '2023-11-14', 'Cartão de Débito', '10101010111', '22222222222', '10101010101010', NULL);

INSERT INTO pedido (codigo_pedido, data_pedido, forma_pagamento, cpf_entregador, cpf_cliente, cnpj_estabelecimento, rota_pedido)

VALUES (13, '2023-11-14', 'Cartão de Débito', '10101010111', '99999999999', '10101010101010', NULL);

INSERT INTO pedido (codigo_pedido, data_pedido, forma_pagamento, cpf_entregador, cpf_cliente, cnpj_estabelecimento, rota_pedido)

```
VALUES (14, '2023-11-14', 'Cartão de Débito', '10101010111', '44444444444',
   '10101010101010', NULL);

INSERT INTO pedido (codigo_pedido, data_pedido, forma_pagamento,
   cpf_entregador, cpf_cliente, cnpj_estabelecimento, rota_pedido)

VALUES (15, '2023-11-14', 'Cartão de Débito', '10101010111', '77777777777',
   '55555555555555', NULL);

INSERT INTO pedido (codigo_pedido, data_pedido, forma_pagamento,
   cpf_entregador, cpf_cliente, cnpj_estabelecimento, rota_pedido)

VALUES

(16, '2023-11-13', 'Cartão de Crédito', '11111111112', '11111111111', '11111111111111',
   NULL),

(17, '2023-10-15', 'Pix', '55555555556', '55555555555', '55555555555555', NULL),

(18, '2023-10-20', 'Dinheiro', '44444444445', '44444444444', '44444444444444',
   NULL),

(19, '2023-09-25', 'Cartão de Débito', '66666666667', '66666666666',
   '66666666666666', NULL),

(20, '2023-09-30', 'Pix', '22222222223', '22222222222', '22222222222222', NULL),

(21, '2023-11-13', 'Dinheiro', '77777777778', '77777777777', '77777777777777',
   NULL),

(22, '2023-11-13', 'Cartão de Crédito', '10101010111', '10101010101',
   '10101010101010', NULL),

(23, '2023-10-15', 'Dinheiro', '88888888889', '88888888888', '88888888888888',
   NULL),

(24, '2023-09-25', 'Pix', '33333333334', '33333333333', '33333333333333', NULL),

(25, '2023-09-30', 'Cartão de Débito', '99999999990', '99999999999',
   '99999999999999', NULL),

(26, '2023-09-30', 'Cartão de Débito', '99999999990', '99999999999',
   '99999999999999',null),
```

```
  (27, '2023-09-30', 'Cartão de Débito', '99999999990', '99999999999',
     '99999999999999',null),

  (28, '2023-09-30', 'Cartão de Débito', '99999999990', '11111111111',
     '99999999999999',null);




select * from pedido;



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



-- Itens_do_Pedido



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



INSERT INTO itens_do_pedido (codigo_pedido, codigo_produto, quantidade,
     preco_total)

VALUES

  (1, 1, 1, NULL),

  (2, 2, 1, NULL),

  (3, 3, 1, NULL),

  (4, 4, 1, NULL),

  (5, 5, 1, NULL),
```

```
  (6, 6, 1, NULL),

  (7, 7, 1, NULL),

  (1, 2, 10, NULL),

  (1, 4, 1, NULL);

insert into itens_do_pedido values(1,8,2);




select * from itens_do_pedido;



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



/* 1 consulta com uma tabela usando operadores básicos de filtro (e.g., IN,

between, is null, etc). */



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



-- CONSULTA 01



-----------------------------------------------------------------------------------------------------
```

/*1 - Com o objetivo de impulsionar a arrecadação de fundos, o aplicativo lançou uma promoção relâmpago que

teve duração entre os dias 13 e 14 de novembro de 2023. Agora, buscamos obter informações sobre a

quantidade de vendas realizadas durante esse período específico, visando avaliar o sucesso e o impacto

da promoção. */

-- Consulta para obter a quantidade de vendas do dia 13 e 14 de novembro de 2023.

```
SELECT count(*)

FROM pedido

WHERE data_pedido between '2023-11-13' and '2023-11-14';
```

/* 3 consultas com inner JOIN na cláusula FROM (pode ser self join, caso o

domínio indique esse uso). */

-- CONSULTA 02

/* 1 - A equipe de marketing está planejando uma campanha promocional exclusiva para todos os clientes

cadastrados no serviço de delivery. Para viabilizar essa iniciativa, é essencial coletar os números de

telefone e os respectivos nomes associados a eles. Essas informações serão utilizadas para enviar SMS

personalizados, proporcionando uma experiência mais direta e personalizada aos clientes durante a

campanha. */

-- Consulta para obter nome e telefone de todos os clientes.

```
SELECT cliente.nome_cliente, telefone.numero_telefone
FROM cliente
INNER JOIN telefone ON cliente.cpf_cliente = telefone.cpf_cliente;
```

-- CONSULTA 03

/* 2 - Sempre é buscado aprimorar constantemente o aplicativo de delivery, e para isso,

é analisado de perto os pedidos dos clientes. Descobriu que o 'Produto Mais Pedido' é uma informação

valiosa. Destaca-se esse item na página inicial, sugerimos ativamente aos usuários e criamos promoções

em torno dele. Essa abordagem otimiza nosso estoque e atende às preferências dos clientes,

proporcionando uma experiência gastronômica excepcional. */

```
-- Consulta para obter a descrição e a quantidade do produto mais pedido.

SELECT

   produto.descricao_produto,

   COUNT(itens_do_pedido.codigo_produto) AS total_pedidos

FROM produto

INNER JOIN itens_do_pedido ON produto.codigo_produto =
    itens_do_pedido.codigo_produto

GROUP BY produto.descricao_produto

ORDER BY total_pedidos DESC

LIMIT 1;

;
```

```
-- CONSULTA 04

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

/* 3 - Uma das principais funcionalidades do nosso aplicativo é apresentar de
     forma clara e acessível

o preço e a descrição de cada produto, juntamente com informações sobre o
     estabelecimento.

Isso possibilita aos clientes a comparação de preços entre diferentes
     restaurantes, permitindo

escolhas mais informadas e a opção de fazer pedidos em estabelecimentos
     específicos de acordo

com suas preferências. */


-- Consulta para obter o nome do estabelecimento, a descrição do produto e o
     valor.


SELECT

   est.nome_fantasia_estabelecimento,

   pr.descricao_produto,

   e.valor_produto

FROM

   estoque e

INNER JOIN estabelecimento est ON e.cnpj_estabelecimento =
     est.cnpj_estabelecimento
```

```
INNER JOIN produto pr ON e.cod_produto = pr.codigo_produto;

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

-- /* 1 consulta com left/right/full outer join na cláusula FROM */

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

-- CONSULTA 05

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

/* 1 - O aplicativo está atualmente promovendo uma campanha exclusiva,
     especialmente direcionada aos nossos

valorizados estabelecimentos parceiros. Nesta iniciativa, os 15 restaurantes que
     se destacarem com o

maior volume de vendas serão recompensados com incentivos especiais
     oferecidos pela própria plataforma

(Lembrando que podem existir estabelecimentos que não tenham vendido nada) */

-- Consulta para resgatar os 15 estabelecimentos que mais venderam.
```

```
SELECT

   estabelecimento.nome_fantasia_estabelecimento,

   NULLIF(COUNT(pedido.codigo_pedido), 0) AS total_pedidos

FROM estabelecimento

LEFT JOIN pedido ON estabelecimento.cnpj_estabelecimento =
    pedido.cnpj_estabelecimento

GROUP BY
    estabelecimento.nome_fantasia_estabelecimento,estabelecimento.cnpj_estabe
    lecimento

ORDER BY total_pedidos DESC NULLS LAST

limit 15;
```

-- /* 2 consultas usando Group By (e possivelmente o having) */

-- CONSULTA 06

/* 1 - Foi conduzido um levantamento dos estados com menor concentração de clientes, identificados por meio

de uma análise criteriosa dos dados. A consulta realizada destacou os estados (estado_cliente) nos quais a

presença de clientes é inferior a 5. Esta análise revela oportunidades estratégicas para a implementação

de ações direcionadas, visando estimular o crescimento e o engajamento dos clientes nestas regiões

específicas. A partir desses insights, serão aplicadas estratégias personalizadas para promover um aumento

no consumo e fortalecer a presença da nossa base de clientes nesses estados, contribuindo assim para o

crescimento geral do nosso negócio. */

-- Consulta para resgatar os Estados que possuem menos de 5 clientes.

```
SELECT estado_cliente, COUNT(*) AS qc

FROM cliente

GROUP BY estado_cliente

HAVING COUNT(*) < 5;
```

-- CONSULTA 07

/* 2 - Com o propósito de aprimorar o sistema de pagamentos do aplicativo, o departamento financeiro

identificou um ponto crucial no quesito pagamentos. A necessidade é descobrir qual forma de pagamento

é mais utilizada pelos clientes e quantas vezes cada método foi empregado. Essa análise detalhada será

fundamental para otimizar e aperfeiçoar a experiência do usuário em transações financeiras, contribuindo

para a eficiência e eficácia do sistema de pagamentos. */

-- Consulta para identificar as formas de pagamento mais usadas e suas quantidades.

```
SELECT forma_pagamento, COUNT(*) AS quantidade_pedidos
FROM pedido
GROUP BY forma_pagamento
ORDER BY quantidade_pedidos DESC;
```

-- /* 1 consulta usando alguma operação de conjunto (union, except ou intersect) */

-- CONSULTA 08

/* 1 - O setor de inteligência de negócios identificou clientes cadastrados que ainda não fizeram pedidos

no aplicativo. A coleta dos CPFs visa segmentá-los estrategicamente, possibilitando campanhas direcionadas

para reativar o engajamento. Para incentivar a primeira compra, está sendo planejado uma promoção exclusiva

para esse grupo. Além disso,essa análise profunda permitirá estratégias mais eficazes para atraí-los de

volta, alinhando-se ao objetivo de aprimorar a experiência do usuário e maximizar o potencial de cada

cliente. */

-- Consulta para identificar CPFs de clientes que nunca fizeram pedidos.

```
SELECT cpf_cliente

FROM cliente

EXCEPT
```

```
SELECT cpf_cliente

FROM pedido;
```

```
-- /* 2 consultas que usem subqueries */
```

```
-- CONSULTA 09
```

```
/* 1- Com a proximidade da Black Friday, marcada para o dia 24 de novembro, esta
     consulta torna-se

extremamente valiosa. Ao identificar os clientes que realizaram compras no mês
     de outubro, terão

a oportunidade de direcionar as melhores promoções e ofertas exclusivas a esse
     grupo. Essa estratégia visa

maximizar o impacto das promoções, proporcionando uma experiência mais
     vantajosa aos clientes que já

demonstraram interesse no nosso serviço durante o mês anterior. */
```

```
-- Consulta para identificar os nomes juntamente com os CPFs dos clientes que
     fizeram compras no mês de

-- Outubro



SELECT nome_cliente,cpf_cliente

FROM cliente

WHERE cpf_cliente IN (SELECT  cliente.cpf_cliente FROM pedido

                                        where cliente.cpf_cliente = pedido.cpf_cliente

                                        and extract(month from data_pedido)= 10) ;



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



-- CONSULTA 10



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



/* 2 - O departamento de inteligência artificial detectou, por meio de análises
     internas, que determinados

produtos não estão alcançando o volume de vendas esperado. Esta consulta é
     especialmente valiosa para

identificar com precisão quais são esses produtos. Ao localizar itens que estão
     disponíveis em estoque, mas

não foram solicitados em nenhum pedido, torna-se possível desenvolver
     estratégias promocionais direcionadas.
```

Essa abordagem visa impulsionar as vendas desses produtos específicos, abrindo oportunidades para promoções

especiais ou campanhas de marketing focalizadas. A implementação dessas ações busca otimizar o desempenho

comercial e garantir uma gestão mais eficiente do inventário. */

-- Consulta para resgatar os produtos que estão em estoque, mas não estão em nenhum pedido.

```
SELECT codigo_produto, descricao_produto

FROM produto

WHERE codigo_produto IN (

   SELECT DISTINCT cod_produto

   FROM estoque

)

AND codigo_produto NOT IN (

   SELECT DISTINCT codigo_produto

   FROM itens_do_pedido

);
```

-- Visões (12,0):

/* 1 visão que permita inserção */

-- ViEW 01

/* A criação da view get_clientes_joão_pessoa é uma estratégia voltada para uma promoção de um prato regional no

aplicativo, que ocorrerá no dia do aniversário de João Pessoa. Essa visão exclusiva permite a inserção

simplificada de novos clientes que residem nessa cidade, facilitando a participação na promoção para os

residentes de João Pessoa. Ao filtrar os clientes com base no estado e cidade, a view proporciona uma

abordagem eficaz para garantir a elegibilidade na promoção, contribuindo para uma experiência mais

direcionada e personalizada aos usuários durante o evento comemorativo. */

-- View responsável por resgatar todas as pessoas que moram na cidade de João Pessoa.

```
CREATE OR REPLACE VIEW get_clientes_joao_pessoa AS

SELECT *

FROM cliente

WHERE estado_cliente = 'PB' AND cidade_cliente = 'João Pessoa'

with check option;



select * from get_clientes_joao_pessoa;



insert into get_clientes_joao_pessoa values('22334400998','Caroline
    Ribeiro','caroline@example.com','21',

'Cristo','João Pessoa','PB','1995-12-03');

insert into get_clientes_joao_pessoa values('22334400988','Caroline
    Rib','caroline@example.com','21',

'Cristo','João Pessoa','PB','1995-12-03');

insert into get_clientes_joao_pessoa values('22334400991','Caroline
    Rib','caroline@example.com','21',

'Cristo','João Pessoa','PB','1995-12-03');



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



/* 2 visões robustas (e.g., com vários joins) com justificativa semântica, de acordo
    com os

requisitos da aplicação. */
```

-- VIEW 02

/* 1 - A criação do get_pedido_simplificado com a apresentação de todos os registros foi pensada para

otimizar a clareza das informações, especialmente durante análises globais, relatórios abrangentes e

administração de sistemas. Essa abordagem evita a inclusão de dados potencialmente confusos, como CNPJ

e CPF, proporcionando uma visão mais direta e compreensível. Além disso, em consideração à segurança dos

dados, a decisão de não divulgar informações mais sensíveis contribui para garantir a privacidade e a

integridade das informações contidas na visão. Essa escolha estratégica visa equilibrar a necessidade

de transparência nas análises gerais com a proteção de dados sensíveis. */

-- View responsável por resgatar informações do pedido de uma forma mais clara e objetiva.

CREATE OR REPLACE VIEW get_pedido_simplificado AS

```
SELECT
    p.codigo_pedido,
    p.data_pedido,
    c.nome_cliente,
    p.forma_pagamento,
    e.nome_fantasia_estabelecimento as nome_estabelecimento,
    en.nome_entregador
FROM
    pedido p
    JOIN cliente c ON p.cpf_cliente = c.cpf_cliente
    JOIN estabelecimento e ON p.cnpj_estabelecimento = e.cnpj_estabelecimento
     JOIN entregador en on p.cpf_entregador = en.cpf_entregador;

select * from get_pedido_simplificado;

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

-- ViEW 03

-----------------------------------------------------------------------------------------------
```

```
/* 2 - A criação da view get_disponibilidade_produtos é altamente valiosa, pois
     permite aos clientes

visualizarem de maneira fácil e eficiente quais produtos estão disponíveis em
     cada estabelecimento. Essa

funcionalidade simplifica a experiência do cliente junto com a navegação no
     aplicativo, agiliza a tomada

de decisões e proporciona transparência na oferta de produtos. */



-- View responsável por resgatar a disponibilidade de produtos em cada
     estabelecimento.



CREATE OR REPLACE VIEW get_disponibilidade_produtos AS

SELECT

   e.nome_fantasia_estabelecimento,

   pr.descricao_produto,

   estoque.quantidade

FROM

   estabelecimento e

   JOIN estoque ON e.cnpj_estabelecimento = estoque.cnpj_estabelecimento

   JOIN produto pr ON estoque.cod_produto = pr.codigo_produto;



select * from get_disponibilidade_produtos;



-----------------------------------------------------------------------------------------------
```

-- Índices (12,0)

-- Índice 01

/* 1 - A criação de um índice nas colunas estado_cliente e cidade_cliente visa otimizar a busca na visão

get_clientes_joao_pessoa (VIEW 01). Essa decisão é estratégica, considerando a alta frequência de consultas

nessa tabela em comparação com a baixa frequência de alterações nos dados. A implementação desse índice

busca significativamente melhorar o desempenho das consultas, reduzindo o tempo necessário para recuperar

informações específicas de clientes da cidade de João Pessoa no estado da Paraíba. Essa otimização é

ótima para garantir uma resposta eficiente em operações de leitura. E nessa situação em especial, já que

é uma consulta simultânea é bem melhor usar um indíce composto, pois além de ser mais eficiente na hora

da busca, reduz o espaço ocupado pelos indíces, já que é mais compacto do que dois indíces separados. */

-- Criação do indíce composto nas colunas estado_cliente e cidade_cliente na tabela cliente, com o

-- propósito de facilitar a busca na get_clientes_joao_pessoa. (VIEW 01)

CREATE INDEX index_estado_cidade_cliente

ON cliente (estado_cliente, cidade_cliente);

-- Índice 02

/* 2 - A criação de um índice na coluna data_pedido, como proposto pelo índice index_data_pedido, é altamente

recomendada para otimizar a consulta que busca a quantidade de vendas entre os dias 13 e 14 de novembro

de 2023 (CONSULTA 01). Este índice facilitará a recuperação eficiente dos registros dentro desse intervalo

de datas, melhorando significativamente o desempenho da consulta. Ao permitir uma busca rápida e direta dos

dados desejados, o índice contribuirá para uma resposta mais ágil e eficiente, otimizando a análise do

sucesso e impacto da promoção relâmpago durante esse período específico. */

-- Criação do índice normal na coluna data_pedido da tabela pedido, com o objetivo de ser mais ágil a busca

--pelos pedidos feitos nas datas 13 e 14 de novembro de 2023. (CONSULTA 01)

```
CREATE INDEX index_data_pedido ON pedido (data_pedido);
```

-- Índice 03

/* 3 - Na iniciativa promocional direcionada aos estabelecimentos parceiros, a consulta visa identificar os 15

restaurantes que se destacaram com o maior volume de vendas. Para otimizar essa análise, a criação de

um índice na coluna nome_fantasia_estabelecimento da tabela "estabelecimento" é altamente recomendada.

Este índice contribui significativamente para a eficiência do processo de agrupamento (GROUP BY), onde

a contagem de pedidos por estabelecimento é realizada. Ao facilitar a localização e organização rápida

das linhas da tabela, o índice não apenas otimiza a operação de agrupamento, mas também melhora a

performance geral da consulta. Dessa forma, a presença desse índice proporciona ganhos substanciais

em termos de tempo de processamento e eficiência, tornando-o uma escolha estratégica para consultas

que envolvem a identificação dos estabelecimentos mais destacados em volume de vendas. */

-- Criação do índice normal da coluna nome_fantasia_estabelecimento da tabela estabelecimento, com a

-- finalidade de otimizar a eficiência do group by, na qual a contagem de pedidos por estabelecimento é

-- feita. (CONSULTA 05)

```
CREATE INDEX index_nome_fantasia_estabelecimento ON
    estabelecimento(nome_fantasia_estabelecimento);
```

-- Reescrita de consultas (6,0)

-- Identificar 2 exemplos de consultas dentro do contexto da aplicação (questão 2.a) que

-- possam e devam ser melhoradas. Reescrevê-las. Justificar a reescrita.

-- Aprimoração 1 da Consulta 10

/* 1 - A utilização de JOINs em substituição a subconsultas, especialmente aquelas envolvendo IN/NOT IN,

é uma escolha mais eficiente em termos de desempenho. Os otimizadores de consultas dos SGBDs são

projetados para lidar de maneira eficaz com operações de junção. Além disso, JOINs proporcionam uma

semântica mais clara e um código mais legível, facilitando a compreensão e manutenção de outras pessoas. */

-- Consulta para resgatar os produtos que estão em estoque, mas não estão em nenhum pedido.

```
SELECT p.codigo_produto, p.descricao_produto --

FROM produto p

JOIN estoque e ON p.codigo_produto = e.cod_produto

LEFT JOIN itens_do_pedido i ON p.codigo_produto = i.codigo_produto

WHERE i.codigo_produto IS NULL;
```

-- Aprimoração 2 da Consulta 09

/* 2 - A otimização da consulta foi alcançada ao substituir a subconsulta com IN por uma junção

utilizando o JOIN. Essa abordagem elimina a necessidade de avaliação repetida da subconsulta,sem precisar

iterar sobre cada linha externa e, para cada uma,executasse a subconsulta, o que resulta em um melhor

desempenho. Além disso, a utilização do JOIN expressa de forma mais clara a relação entre as tabelas cliente

e pedido, e retira a necessidade de avaliar separadamente cada linha externa. Além disso, o uso de

abreviações com o nome cliente e pedido, deixam o codigo mais limpo e de fácil entendimento. Essa modificação

torna o código mais legível, facilitando a compreensão e manutenção, enquanto a criação do índice na coluna

data_pedido (índice 02) ajuda a otimizar a execução da consulta, garantindo eficiência em ambientes com grande

volume de dados. */

-- Consulta para identificar os nomes juntamente com os CPFs dos clientes que fizeram compras no mês de

-- Outubro

```
SELECT c.nome_cliente, c.cpf_cliente

FROM cliente c

JOIN pedido p ON c.cpf_cliente = p.cpf_cliente

WHERE extract(month from p.data_pedido) = 10;
```

```
-- e. Funções e procedures armazenadas (14,0):
```

```
-- 1 função que use SUM, MAX, MIN, AVG ou COUNT

-- 2 funções e 1 procedure com justificativa semântica, conforme os requisitos da
     aplicação

-- ** Pelo menos uma função ou procedure deve ter tratamento de exceção

-- ** As funções desta seção não são as mesmas das funções de triggers
```

-- Função 01 que use SUM, MAX, MIN, AVG ou COUNT

/* 1 - A introdução da funcionalidade que exibe o total de gastos do cliente no aplicativo é uma adição

valiosa, permitindo que os usuários tenham uma compreensão imediata de seus gastos acumulados. Essa

funcionalidade utiliza a função calcularValorPorCliente, garantindo precisão nos cálculos e proporcionando

uma experiência simplificada. Agora, os clientes podem facilmente monitorar quanto gastaram ao longo do

tempo, contribuindo para uma visão clara de seus hábitos de compra. */

-- Função responsável por retornar o valor total de gastos de um determinado cliente.

```
CREATE OR REPLACE FUNCTION calcularValorPorCliente(cpf_clienteP CHAR(11))
RETURNS NUMERIC AS
$$
DECLARE
  total_compras NUMERIC;
BEGIN
```

```
  SELECT COALESCE(SUM(preco_total), 0) INTO total_compras

  FROM itens_do_pedido -- Da tabela itens_do_pedido

  WHERE codigo_pedido IN (SELECT codigo_pedido FROM pedido WHERE
    cpf_cliente = cpf_clienteP);

  -- Enquanto o codigo do pedido da tabela itens_pedido, esteja na tabela pedido,
    e o cpf

  -- seja o parâmetro que foi passado

  RETURN total_compras;

END;

$$

LANGUAGE plpgsql;



-- OBSERVAÇÃO MUITO IMPORTANTE: Essa função só irá funcionar, quando a
     trigger 3 for criada,

-- e for feita uma inserção na tabela itens_do_pedido .



-- Passo 01 : select * from calcularvalorporcliente('11111111111');



-- Passo 02 após criação da trigger 3 : insert into itens_do_pedido
     values(1,7,10,null);



-- Passo 03 : select * from calcularvalorporcliente('11111111111');



-- Esse insert so poderá ser feitos após a criação da trigger !!!!!!!!!!!!!!
```

select * from itens_do_pedido;

-- 2 funções e 1 procedure com justificativa semântica, conforme os requisitos da aplicação

----

-- Função 01 com justificativa semântica, conforme os requisitos da aplicação

----

/* 1 - A função contarPedidosPorEntregador é eficiente, pois proporciona ao entregador uma visão detalhada e

transparente do número de entregas que ele realizou em um período específico. Ao receber informações

precisas sobre sua atividade de entrega, o entregador pode avaliar seu desempenho e planejar sua agenda

```
com base nas demandas passadas. Essa eficiência contribui não apenas para a
    gestão pessoal do entregador,

mas também para a otimização geral das operações de entrega do
    estabelecimento, permitindo uma alocação

mais eficaz de recursos e uma melhor compreensão do fluxo de trabalho ao longo
    do tempo. */



-- Criação da função para contar quantos pedidos um entregador tem em um
    período de tempo



CREATE OR REPLACE FUNCTION contarPedidosPorEntregador(

   cpf_entregadorP CHAR(11),

   data_inicio DATE,

   data_fim DATE

)

RETURNS INT AS $$

DECLARE

   total_pedidos INT;

BEGIN

   SELECT COUNT(*)

   INTO total_pedidos

   FROM pedido

   WHERE cpf_entregador = cpf_entregadorP

      AND data_pedido >= data_inicio
```

```
        AND data_pedido <= data_fim;


    RETURN total_pedidos;

END;

$$ LANGUAGE plpgsql;


select * from contarPedidosPorEntregador('11111111112','2023-09-01','2023-11-15');


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


-- Função 02 com justificativa semântica, conforme os requisitos da aplicação


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


/* 2 - A função calcularFaturamentoMensalEstabelecimento retorna o valor total
      faturado por um

estabelecimento em um determinado mês e ano. Essa métrica fornece uma visão
      mensal do desempenho

financeiro, possibilitando que a administração avalie a eficácia das estratégias de
      vendas e tome

decisões informadas para melhorar o desempenho do negócio. */


-- Criação da função para calcular o faturamento mensal de um determinado
      estabelecimento.
```

```
CREATE OR REPLACE FUNCTION
    calcularFaturamentoMensalEstabelecimento(cnpj_estabelecimentoP
    CHAR(14),

                                                    mes INTEGER, ano
    INTEGER)

RETURNS NUMERIC AS

$$

DECLARE

  valor_faturado NUMERIC;

BEGIN

  SELECT

    COALESCE(SUM(ip.preco_total), 0) INTO valor_faturado

  FROM

    itens_do_pedido ip

  JOIN

    pedido p ON ip.codigo_pedido = p.codigo_pedido

  WHERE

    p.cnpj_estabelecimento = cnpj_estabelecimentoP

    AND EXTRACT(MONTH FROM p.data_pedido) = mes

    AND EXTRACT(YEAR FROM p.data_pedido) = ano;



  RETURN valor_faturado;
```

```
END;

$$

LANGUAGE plpgsql;



-- OBSERVAÇÃO MUITO IMPORTANTE: Essa função só irá funcionar, quando a
    trigger 3 for criada,

-- e for feita uma inserção na tabela itens_do_pedido .



/* Passo 1 após criação da trigger 3 : insert into pedido values(200, '2023-09-30',

                                                                'Cartão de
    Débito',

                                                               '99999999990',

                                                              '11111111111',


    '99999999999999',null);  */



-- Passo 02          : select * from
    calcularFaturamentoMensalEstabelecimento('99999999999999',09,2023);



-- Passo 03 : insert into itens_do_pedido values (200,5,1);



-- Passo 04 : select * from
    calcularFaturamentoMensalEstabelecimento('99999999999999',09,2023);
```

-- Esses 2 insert so poderão ser feitos após a criação da trigger !!!!!!!!!!!!!!!

-- Procedure 01 com justificativa semântica, conforme os requisitos da aplicação

----

/* 1 - A utilização da procedure, apesar da integridade referencial já garantida pelo SGBD, justifica-se pela

necessidade de apresentar mensagens de erro personalizadas e compreensíveis aos usuários finais.

Isso aprimora significativamente a experiência do usuário, evitando mensagens técnicas do banco de dados e

oferecendo explicações mais acessíveis sobre possíveis problemas. Então essa escolha contribui para uma

interação mais amigável e compreensível com o sistema por parte dos usuários. */

-- Procedure responsável por inserir um novo pedido, tratando execeções e mostrando ao usuário de uma maneira

-- mais humanizada, ao invés de aparecer algo técnico.

```
CREATE OR REPLACE PROCEDURE inserirPedido(

    IN codigo_pedidoP INT,

    IN data_pedidoP DATE,

    IN forma_pagamentoP VARCHAR(45),

    IN cpf_entregadorP CHAR(11),

    IN cpf_clienteP CHAR(11),

    IN cnpj_estabelecimentoP CHAR(14)

)



AS $$

BEGIN

    -- Verifica se o entregador existe

    IF NOT EXISTS (SELECT 1 FROM entregador WHERE cpf_entregador =
      cpf_entregadorP) THEN

            raise notice 'Entregador com CPF % não encontrado.', cpf_entregadorP;

        RAISE EXCEPTION '';

    END IF;



    -- Verifica se o cliente existe

    IF NOT EXISTS (SELECT 1 FROM cliente WHERE cpf_cliente = cpf_clienteP)
      THEN

            raise notice 'Cliente com CPF % não encontrado.', cpf_clienteP;

        RAISE EXCEPTION '';
```

```
    END IF;



    -- Verifica se o estabelecimento existe

    IF NOT EXISTS (SELECT 1 FROM estabelecimento WHERE
      cnpj_estabelecimento = cnpj_estabelecimentoP) THEN

            raise notice 'Estabelecimento com CNPJ % não encontrado.',
     cnpj_estabelecimentoP;

       RAISE EXCEPTION '';

    END IF;



    -- Insere o pedido

    INSERT INTO pedido (codigo_pedido, data_pedido, forma_pagamento,
      cpf_entregador, cpf_cliente, cnpj_estabelecimento)

    VALUES (codigo_pedidoP, data_pedidoP, forma_pagamentoP, cpf_entregadorP,
      cpf_clienteP, cnpj_estabelecimentoP);



    raise notice 'Pedido inserido com sucesso.';



EXCEPTION

    WHEN raise_exception then

            raise notice 'Tente novamente com dados válidos';

END;

$$
```

```
LANGUAGE 'plpgsql';



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



-- Teste 01 - Entregador com CPF XXXXXXXXXXX não encontrado. ( Exceção )



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



CALL inserirPedido(

    1, -- Código do pedido

    '2023-11-18', -- Data do pedido

    'Cartão de Crédito', -- Forma de pagamento

    '22222222211', -- CPF do entregador

    '11111111111', -- CPF do cliente

    '01234567890999' -- CNPJ do estabelecimento

);



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



-- Teste 02 - Cliente com CPF XXXXXXXXXXX não encontrado. ( Exceção )
```

CALL inserirPedido(

1, -- Código do pedido

'2023-11-18', -- Data do pedido

'Cartão de Crédito', -- Forma de pagamento

'99999999990', -- CPF do entregador

'84209472819', -- CPF do cliente

'01234567890999' -- CNPJ do estabelecimento

);

-- Teste 03 - Estabelecimento com CNPJ XXXXXXXXXXXXXX não encontrado. (Exceção)

CALL inserirPedido(

1, -- Código do pedido

'2023-11-18', -- Data do pedido

'Cartão de Crédito', -- Forma de pagamento

```
    '99999999990', -- CPF do entregador
    '23456789012', -- CPF do cliente
    '01234567891234' -- CNPJ do estabelecimento
);

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

-- Teste 04 - Pedido inserido com sucesso. ( Sem exceções )

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

CALL inserirPedido(
    76, -- Código do pedido
    '2023-11-18', -- Data do pedido
    'Cartão de Crédito', -- Forma de pagamento
    '99999999990', -- CPF do entregador
    '23456789012', -- CPF do cliente
    '98765432101234' -- CNPJ do estabelecimento
);

-----------------------------------------------------------------------------------------------------
```

-- Triggers (9,0)

/* 3 diferentes triggers com justificativa semântica, de acordo com os requisitos da

aplicação. */

-- 01 Trigger

/* A integração da trigger verificar_estoque e da função associada verificarEstoque é de extrema importância,

pois estabelece verificações cruciais antes da inserção de um novo item no pedido. Inicialmente, o sistema

verifica se o produto está disponível no estoque associado ao estabelecimento do pedido, lançando uma

exceção se não for encontrado. Posteriormente, realiza uma segunda verificação para garantir que a quantidade

solicitada não exceda a disponibilidade em estoque. Ao superar essas etapas, a função efetua o desconto da

quantidade no estoque e calcula o valor total do item vezes a quantidade que o cliente solicitou, proporcionando

um controle preciso e eficiente do estoque, reforçando a integridade dos dados e prevenindo transações inválidas

no sistema. Essa abordagem é essencial para garantir a consistência e confiabilidade das informações, tornando-a

uma ferramenta valiosa para o gerenciamento de pedidos e estoques no contexto do sistema. */

-- Trigger responsável por manter o controle do estoque, evitando se não tiver produtos suficientes, ou se

-- o estabelecimento não possuir o produto específico .

```
CREATE TRIGGER verificar_estoque

BEFORE INSERT ON itens_do_pedido

FOR EACH ROW execute procedure verificarEstoque();
```

```
CREATE OR REPLACE FUNCTION verificarEstoque() RETURNS TRIGGER AS

$$

DECLARE
```

```
    qtd_em_estoque INT;
    precoD NUMERIC(10,2);
    cnpjD CHAR(14);
BEGIN
    -- Obtemos o cnpj_estabelecimento do pedido associado ao novo item
    SELECT pedido.cnpj_estabelecimento
    INTO cnpjD
    FROM pedido
    WHERE pedido.codigo_pedido = NEW.codigo_pedido;

    -- Verifica a quantidade em estoque antes de inserir um novo item no pedido
    SELECT estoque.quantidade
    INTO qtd_em_estoque
    FROM estoque
    WHERE estoque.cnpj_estabelecimento = cnpjD
      AND estoque.cod_produto = NEW.codigo_produto;
     if qtd_em_estoque is null then
            raise exception 'Não existe esse produto no estabelecimento do pedido
    escolhido';
            return null;
     end  if;
```

```
    SELECT estoque.valor_produto

    INTO precoD

    FROM estoque

    WHERE estoque.cnpj_estabelecimento = cnpjD

      AND estoque.cod_produto = NEW.codigo_produto;



    -- Verifica se há estoque suficiente para o novo item no pedido

    IF qtd_em_estoque < NEW.quantidade THEN

       RAISE NOTICE 'Estoque insuficiente para o produto no estabelecimento ';

       RETURN NULL;

    ELSE

       UPDATE estoque

       SET quantidade = quantidade - NEW.quantidade

       WHERE cod_produto = NEW.codigo_produto AND cnpj_estabelecimento =
      cnpjD;



       NEW.preco_total = NEW.quantidade * precoD;



            raise notice 'Quantidade inserida com sucesso, junto com seu preço
      total';

       RETURN NEW;

    END IF;

END;
```

```
$$

LANGUAGE plpgsql;

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

-- Teste 01 - ERROR:  Não existe esse produto no estabelecimento do pedido
      escolhido ( Exceção )

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

insert into itens_do_pedido values (8,8,10,null);

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

-- Teste 02 - ERROR:  Estoque insuficiente para o produto no estabelecimento  (
      Exceção )

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

insert into itens_do_pedido values (8,4,10000,null);

-----------------------------------------------------------------------------------------------------
```

-- Teste 03 - Quantidade inserida com sucesso, junto com seu preço total (Sem exceção)

---

insert into itens_do_pedido values (3,4,10,null);

-- 02 Trigger

/* 2 - A implementação desta trigger simplifica o processo de pedidos para os clientes, ao mesmo

tempo em que estabelece uma prática de segurança eficaz. Quando um cliente deseja indicar um novo endereço

durante um pedido, é necessário realizar a alteração diretamente em seu cadastro. Essa abordagem não apenas

garante a consistência dos dados, mas também reforça a segurança do sistema, vinculando qualquer modificação

de endereço à autenticação e autorização do cliente. Em última análise, essa funcionalidade não só alivia

os clientes da necessidade de inserir repetidamente seu endereço, mas também promove a segurança e adesão

às regras de negócio, uma vez que a atualização do endereço é obrigatoriamente realizada através do cadastro. */

-- Trigger responsável por resgatar todo o endereço do cliente que solicitou o pedido, fazer a concatenação

-- e inserir na rota do pedido .

```
create trigger adicionaRota before insert on pedido

for each row execute procedure adiciona_Rota();
```

```
create or replace function adiciona_Rota () returns trigger as

$$

declare

registro record;

begin

select numero_cliente,bairro_cliente,cidade_cliente,estado_cliente into registro
    from
```

```
cliente where cpf_cliente = new.cpf_cliente;



NEW.rota_pedido = format('%s %s %s %s', registro.numero_cliente,
    registro.bairro_cliente,

                                            registro.cidade_cliente,
    registro.estado_cliente);



return new;

end;

$$

language 'plpgsql';



insert into pedido values (891, '2023-09-30', 'Cartão de Débito', '99999999990',
    '11111111111', '99999999999999',null);

insert into itens_do_pedido values(891,5,1,null);

select * from pedido where codigo_pedido = 891;




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



-- 03 Trigger



-----------------------------------------------------------------------------------------------------
```

/* 1- Essa trigger é uma solução eficaz para automatizar a atualização dinâmica do valor total do pedido,

garantindo a integridade e consistência dos dados. Ao verificar e criar a coluna "preco_total_pedido"

apenas quando necessário, ela demonstra uma abordagem flexível e adaptável a mudanças na estrutura da

tabela "pedido". E toda vez que acontece um nova inserção na tabela itens_do_pedido, essa trigger é

responsável por atualizar o valor total na tabela pedido */

-- Trigger responsável por criar uma coluna chamada valor total do pedido, se não existir, e toda vez

-- que for adicionado um novo produto no pedido, ela é responsável por atualizar o valor total do pedido

---

```
CREATE TRIGGER atualiza_preco_total_pedido

AFTER INSERT ON itens_do_pedido

FOR EACH ROW

EXECUTE FUNCTION atualizar_preco_total_pedido_function();
```

```
-----------------------------------------------------------------------------------------------------

CREATE OR REPLACE FUNCTION atualizar_preco_total_pedido_function()

RETURNS TRIGGER AS $$

DECLARE

   total_pedido numeric(10,2);

BEGIN

   -- Verifica se a coluna preco_total_pedido existe na tabela pedido

   IF NOT EXISTS (

      SELECT 1

      FROM information_schema.columns

      WHERE table_name = 'pedido' AND column_name = 'preco_total_pedido'

   ) THEN

      -- Cria a coluna preco_total_pedido se não existir

      EXECUTE 'ALTER TABLE pedido ADD COLUMN preco_total_pedido
     numeric(10,2) DEFAULT 0';

   END IF;



   -- Soma os valores da tabela itens_do_pedido para o pedido específico

   SELECT COALESCE(SUM(preco_total), 0)

   INTO total_pedido

   FROM itens_do_pedido
```

```
    WHERE codigo_pedido = NEW.codigo_pedido;


    -- Atualiza ou insere na coluna preco_total_pedido na tabela pedido
    UPDATE pedido
    SET preco_total_pedido = total_pedido
    WHERE codigo_pedido = NEW.codigo_pedido;


    RETURN NEW;
END;
$$ LANGUAGE plpgsql;


insert into itens_do_pedido values(1,9,5,null);




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


-- EXTRAS


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


-- TESTES
```

-- Apagar em ordem tirando as dependências das tabelas :

```
delete from telefone;

delete from estoque;

delete from itens_do_pedido;

delete  from pedido;

delete from cliente;

delete from estabelecimento;

delete from entregador;

delete from produto;
```

```
select * from cliente;

select * from estabelecimento;

select * from entregador;

select * from telefone;

select * from produto;

select * from itens_do_pedido;

select * from pedido;
```