Trabalho Prático - Etapa 2

Alunos: Gabriel Henrique Santos, Gabriela Ribeiro, Júlia Ribeiro, Larissa Rocha, Vitor Firmino.

Descrição do problema

Para um aplicativo de delivery alimentício, os requisitos de dados são fundamentais para garantir que o sistema opere de maneira eficiente e proporcione uma experiência otimizada para usuários, restaurantes, entregadores e a própria plataforma. Esses requisitos abrangem desde informações sobre clientes e restaurantes até o gerenciamento de pedidos, pagamentos e avaliações.

O sistema deve armazenar dados essenciais dos clientes, que utilizam a plataforma para realizar seus pedidos. Cada cliente terá um identificador único, nome de usuário e senha. Os clientes podem ser pessoas físicas (possuindo nome completo, endereço, número de telefone, e-mail e cadastro de pessoa física) ou jurídicas (possuindo cadastro de pessoa jurídica, razão social, endereço e telefone), permitindo maior flexibilidade no cadastro. O histórico de pedidos, incluindo detalhes sobre compras anteriores e preferências alimentares, deve ser salvo para que o aplicativo possa oferecer sugestões personalizadas, facilitando pedidos futuros e melhorando a experiência do usuário.

Além disso, o sistema deve gerenciar as informações dos restaurantes cadastrados, que são tratados como pessoas jurídicas. Cada restaurante terá um identificador único e tipos de culinária oferecidos. Essas informações permitem que os usuários conheçam melhor as opções disponíveis e façam escolhas informadas ao selecionar onde realizar suas compras.

Os produtos oferecidos pelos restaurantes devem ser detalhados no sistema. Cada produto terá um identificador único, nome, descrição, preço, informações de disponibilidade e uma URL da imagem, permitindo aos clientes visualizar o que estão comprando e tomar decisões mais informadas.

Os pedidos realizados pelos clientes são essenciais para rastrear operações de entrega. Cada pedido será identificado de forma única e incluirá o identificador do cliente e do restaurante, a data e hora do pedido, o estado atual (como em preparação, a caminho ou entregue) e o valor total. Além disso, os itens do pedido serão registrados, detalhando cada produto, suas quantidades e preços, garantindo transparência nas transações.

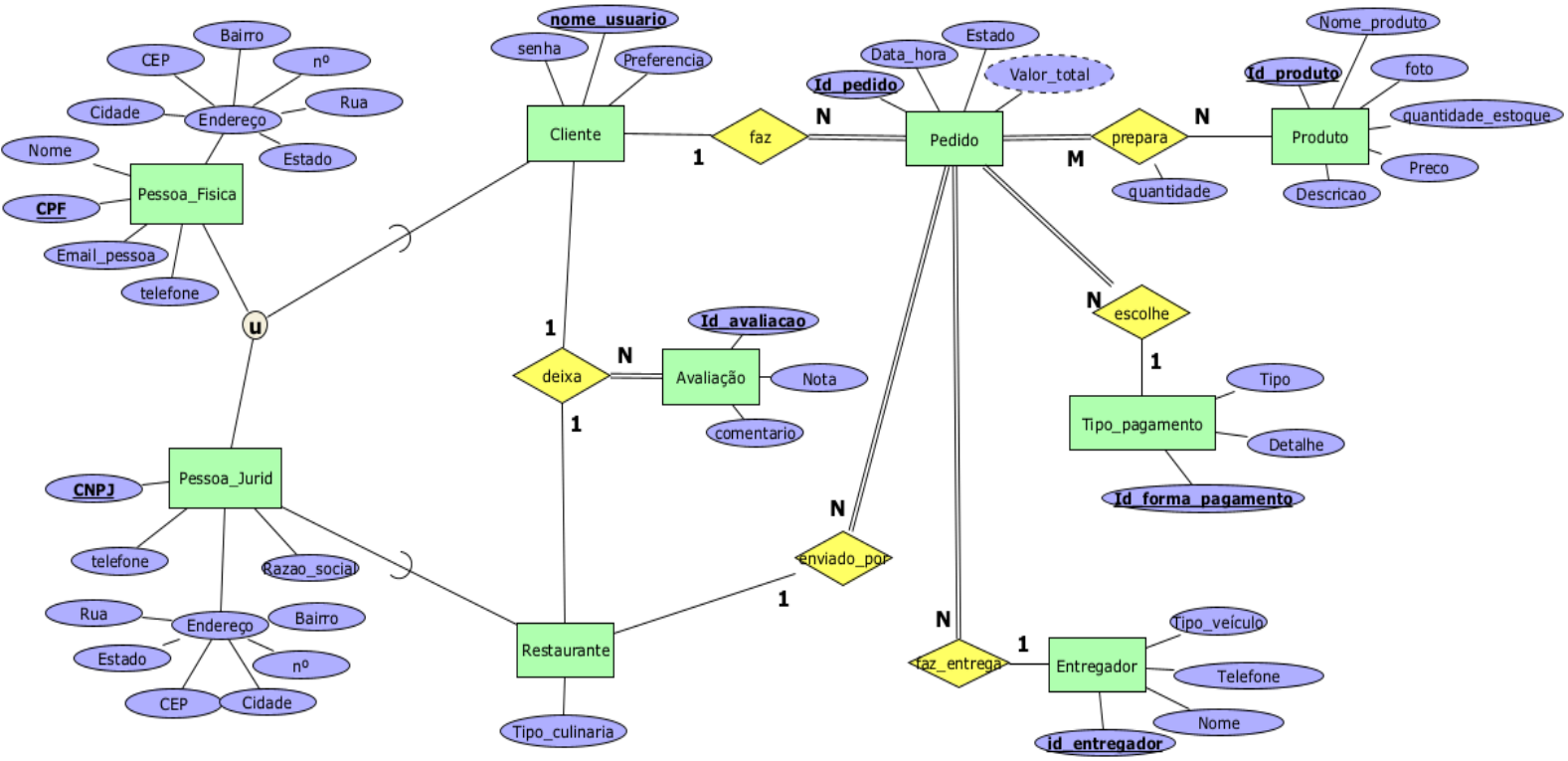
O sistema também deve armazenar informações sobre as formas de pagamento disponíveis. Cada método de pagamento terá um identificador único, tipo (como cartão de crédito, débito ou dinheiro) e detalhes adicionais, se necessário. A diversidade nas opções de pagamento é crucial para atender às preferências dos clientes e facilitar o processo de compra.

Informações sobre os entregadores responsáveis pelas entregas de pedidos são igualmente importantes, incluindo um identificador único para o entregador, nome e telefone do contato do entregador, além do tipo de veículo utilizado. Esses dados garantem que as entregas sejam realizadas de forma eficiente e dentro do prazo.

As avaliações feitas pelos clientes sobre os restaurantes também desempenham um papel vital no ecossistema do delivery. Cada avaliação terá um identificador único, o identificador do cliente que avaliou, o identificador do restaurante, a nota dada (de 1 a 5 estrelas) e um comentário opcional. Essas avaliações ajudam a construir a reputação dos restaurantes na plataforma, influenciando as decisões de outros clientes e promovendo a qualidade dos serviços.

Em resumo, esses requisitos de dados são essenciais para o funcionamento eficiente de um aplicativo de delivery alimentício, assegurando uma gestão eficaz de pedidos, envios, pagamentos e avaliações, e proporcionando uma experiência positiva a todos os usuários envolvidos no processo.

Diagrama ER:



Modelo para o dicionário de dados

Tipo Entidade	Pessoa_Jurid		
Descrição	Representa entidade de atributos em comum de pessoa física e pessoa jurídica		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**

CEP	Código de Endereçamento Postal	Texto (8)	N
Bairro	Divisão geográfica dentro de uma cidade	Texto(20)	N
Cidade	Nome da localidade onde reside.	Texto(20)	N
Rua	Refere-se à via pública onde reside.	Texto(50)	N
Número	Número que identifica uma localização específica	Texto(10)	N
Estado	Unidade federativa onde a cidade está localizada.	Texto(20)	N
Telefone_pessoa	Número de telefone da pessoa	Texto(11)	N
CNPJ	Cadastro de pessoa Jurídica	Texto (11dígitos)	N
Razao_social	Nome formal da empresa	Texto (80)	N

Tipo Entidade	Pessoa_Fisica		
Descrição	Representa entidade de atributos em comum de pessoa física e pessoa jurídica		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
CEP	Código de Endereçamento Postal	Texto (8)	N
Bairro	Divisão geográfica dentro de uma cidade	Texto(20)	N
Cidade	Nome da localidade onde reside.	Texto(20)	N
Rua	Refere-se à via pública onde reside.	Texto(50)	N
Número	Número que identifica uma localização específica	Texto(10)	N
Estado	Unidade federativa onde a cidade está localizada.	Texto(20)	N
Telefone_pessoa	Número de telefone da pessoa	Texto(11)	N
CPF	Cadastro de pessoa Física	Texto (11dígitos)	N
Nome	Nome completo da pessoa	Texto (80)	N
Email_pessoa	Email da pessoa	Texto (50)	N

Tipo Entidade	Cliente		
Descrição	Conjunto de clientes que utilizam a plataforma de delivery		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Nome_usuario	Nome de usuário do cliente	Texto (30)	N
Senha	Senha para login	Texto(30)	N
Preferencia	Preferência alimentar do cliente	Texto(100)	S

Tipo Entidade	Restaurante		
Descrição	Conjunto de restaurantes cadastrados na plataforma		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Tipo_culinaria	Tipos de culinária oferecidos	Texto (50)	N

Tipo Entidade	Produto		
Descrição	Conjunto de produtos oferecidos pelos restaurantes		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Id_produto	Identificador único do produto	Inteiro (4)positivo	N
Nome_produto	Nome do produto	Texto (80)	N
Descricao	Descrição do produto	Texto (200)	S
Preco	Preço do produto	Real (8,2) positivo	N
Quantidade_em_estoque	Quantidade em estoque	Inteiro (4) positivo	N
Foto	URL da imagem do produto	Texto (200)	S

Tipo Entidade	Pedido		
Descrição	Conjunto de pedidos realizados pelos cliente		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Id_pedido	Identificador único do pedido	Inteiro (4) positivo	N
Data_hora	Data e hora do pedido	Data/hora	N
Estado	Estado atual do pedido (1 para recebendo pagamento, 2 para preparando pedido, 3 para saiu para entrega, 4 para foi entregue)	Inteiro positivo (1)	N
Valor_total	Valor total do pedido (atributo derivado calculado pela soma de todos os produtos do pedido vezes a quantidade que cada produto individual foi pedido)	Real (8,2)positivo	N

Tipo Entidade	Tipo_Pagamento
Descrição	Conjunto de métodos de pagamento disponíveis

Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Id_forma_pagamento	Identificador único da forma de pagamento	Inteiro positivo(4)	N
Tipo	Tipo de pagamento (1 para dinheiro,2 para cartão ou 3 para pix)	inteiro positivo (1)	N
Detalhes	Detalhes adicionais sobre o pagamento	Texto (100)	S

Tipo Entidade	Entregador
Descrição	Informações sobre as entregas de pedidos

Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Id_entregador	Identificador único do entregador	Inteiro (4) positivo	N
Nome_contato	Nome do entregador	Texto (80)	N
Telefone_contato	Telefone de contato do entregador	Texto (11)	N
Tipo_veículo	Tipo de veículo do entregador(1 para moto, 2 para carro, 3 para bicicleta)	inteiro positivo (1)	N

Tipo Entidade	Avaliação
Descrição	Avaliações feitas por clientes sobre restaurante

Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Id_avaliacao	Identificador único da avaliação	Inteiro (4) positivo	N
Nota	Nota dada ao restaurante	Inteiro (1)	N
Comentario	Comentário adicional sobre a avaliação	Texto (200)	S

Tipo Relacionamento	Faz
Descrição	Relacionamento entre o pedido e o cliente que o realiza. Cada cliente pode realizar inúmeros pedidos, mas cada pedido é feito por apenas um cliente.

Tipo Relacionamento	Envia
Descrição	Relacionamento que associa os pedidos aos restaurantes. Cada pedido está ligado a um restaurante, mas um restaurante pode receber vários pedidos.

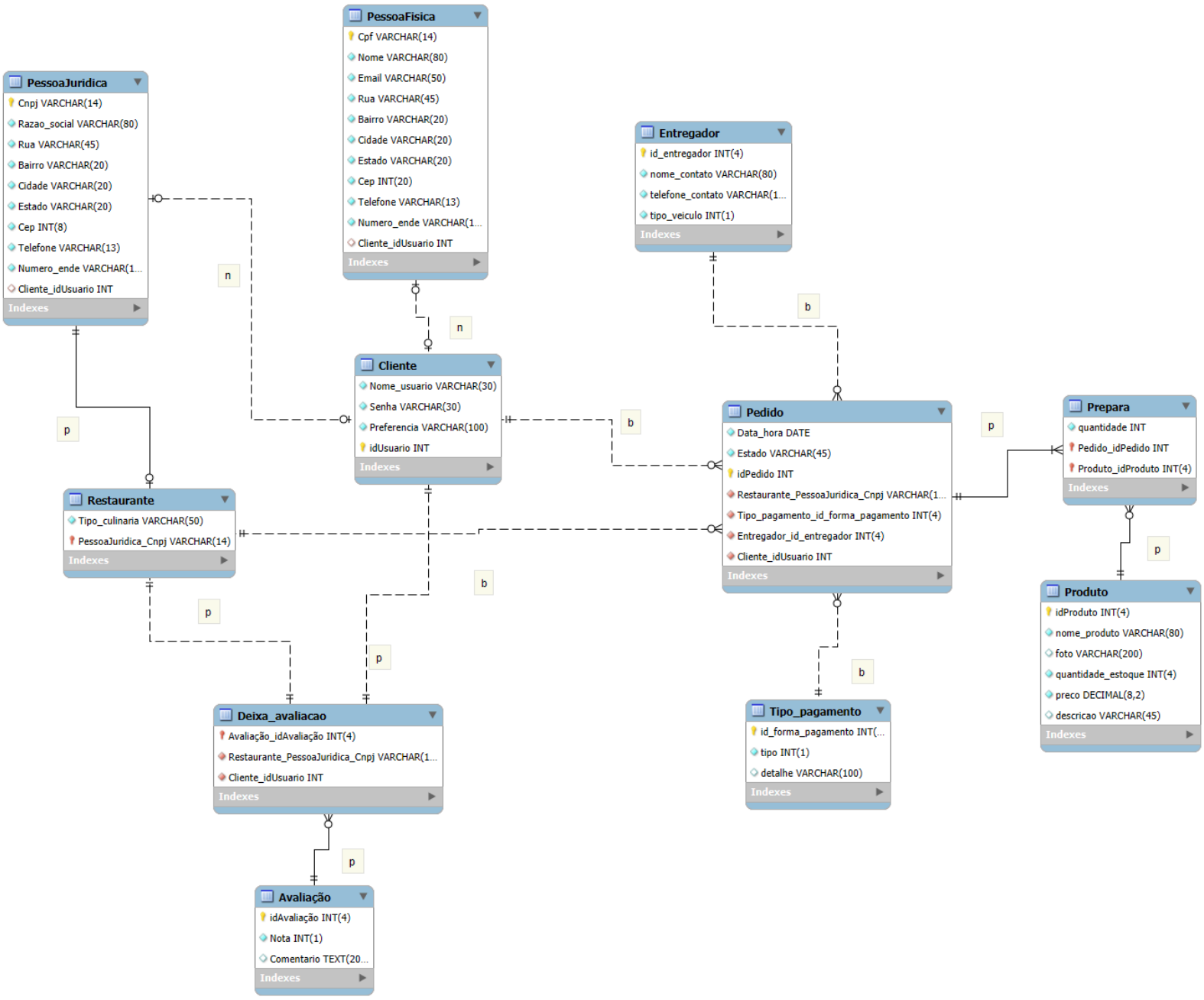
Tipo Relacionamento	Escolhe
Descrição	Nesse relacionamento o cliente pode escolher uma forma de pagamento para pagar seu pedido. Cada pedido pode ter apenas uma forma de pagamento associada (como cartão de crédito, débito ou dinheiro). Mas uma forma de pagamento pode estar presente em vários pedidos.

Tipo Relacionamento	Faz_entrega
Descrição	Relacionamento em que o entregador faz a entrega do pedido. Cada entregador pode entregar diversos pedidos, mas um pedido só pode ser entregue por um entregador.

Tipo Relacionamento	Deixa
Descrição	Nesse relacionamento o cliente pode fazer uma avaliação ao restaurante em que ele fez o pedido. Cada cliente pode fazer várias avaliações, mas cada avaliação está associada a um único cliente e a um único restaurante.

Tipo Relacionamento	Prepara		
Descrição	Este relacionamento conecta os pedidos aos produtos que os compõem. Cada pedido pode conter diversos produtos, e cada produto pode estar presente em vários pedidos diferentes.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Quantidade	Quantidade do produto	Inteiro (4) positivo	N

Diagrama Relacional



Dicionário de dados – Relacional

Tipo Entidade	Pessoa_Jurid
Descrição	Representa entidade de atributos em comum de pessoa física e pessoa jurídica
Atributos	
Nome	**Descrição**
CEP	Código de Endereçamento Postal
Bairro	Divisão geográfica dentro de uma cidade
Cidade	Nome da localidade onde reside.
Rua	Refere-se à via pública onde reside.
Número	Número que identifica uma localização específica
Estado	Unidade federativa onde a cidade está localizada.
Telefone_pessoa	Número de telefone da pessoa
CNPJ	Cadastro de pessoa Jurídica
Razao_social	Nome formal da empresa
Nome_usuario	Nome de usuário do cliente.
idCliente	id do cliente

Tipo Entidade	Pessoa_Fisica
Descrição	Representa entidade de atributos em comum de pessoa física e pessoa jurídica
Atributos	
Nome	**Descrição**
CEP	Código de Endereçamento Postal
Bairro	Divisão geográfica dentro de uma cidade
Cidade	Nome da localidade onde reside.
Rua	Refere-se à via pública onde reside.
Número	Número que identifica uma localização específica
Estado	Unidade federativa onde a cidade está localizada.
Telefone_pessoa	Número de telefone da pessoa
CPF	Cadastro de pessoa Física
Nome	Nome completo da pessoa
Nome_usuario	Nome de usuário do cliente.
idCliente	id do cliente
Email_pessoa	Email da pessoa

Tabela	Cliente
Descrição	Informações sobre clientes cadastrados.
Atributos	
Nome	**Descrição**
Senha	Senha de acesso do cliente.

Nome_usuario	Nome de usuário do cliente.
Preferencia	Preferências alimentares ou gerais.
idCliente	id do cliente

Tabela	Restaurante
Descrição	Informações sobre os restaurantes cadastrados.
Atributos	
Nome	**Descrição**
Tipo_culinaria	Tipo de culinária oferecida.
PessoaJuridica_cnpj	Chave Estrangeira Pessoa Juridica

Tabela	Deixa_avaliacao
Descrição	Avaliações que podem ser deixadas por clientes
Atributos	
Nome	**Descrição**
Avaliacao_idavaliacao	Chave estrangeira Avaliação
Restaurante_PessoaJuridica_cnpj	Restaurante a ser avaliado.
Cliente_Nome_usuario	Cliente que está avaliando.
Cliente_idUsuario	Id do cliente q está avaliando.

Tabela	Avaliacao
Descrição	Informações sobre avaliações feitas por clientes.
Atributos	
Nome	**Descrição**
ID_Avaliacao	Código de identificação da avaliação.
Nota	Nota atribuída ao restaurante.
Comentario	Comentário sobre a experiência.

Tabela	Pedido
Descrição	Conjunto de pedidos feitos.
Atributos	
Nome	**Descrição**
ID_Pedido	Código de identificação do pedido.
Data_hora	Data e hora em que o pedido foi feito.
Estado	Estado atual do pedido.
Restaurante_PessoaJuridica_Cnpj	Restaurante responsável pelo pedido.
Tipo_pagamento_id_forma_pagamento	Forma de pagamento.
Entregador_id_entregador	Entregador responsável pela entrega do pedido.
Cliente_Nome_usuario	Nome de usuário do cliente.
Cliente_idUsuario	Código de identificação de usuário do cliente.

Tabela	Tipo Pagamento
Descrição	Conjunto de métodos de pagamento disponíveis.
Atributos	

Nome	**Descrição**
ID_forma_pagamento	Código de identificação da forma de pagamento
Tipo	Tipo de pagamento.
Detalhes	Detalhes adicionais sobre o pagamento.

Tabela	Entregador
Descrição	Entregadores responsáveis pela entrega dos pedidos.
Atributos	
Nome	**Descrição**
ID_Entregador	Código de identificação do entregador.
Nome_contato	Nome do entregador.
Telefone_contato	Telefone de contato do entregador.
Tipo_veiculo	Tipo de veiculo do entregador.

Tabela	Produto
Descrição	Conjunto de produtos oferecidos pelo restaurante.
Atributos	
Nome	**Descrição**
ID_Produto	Código de identificação do produto.
Nome_produto	Nome do produto.
Descricao	Descrição do produto
Preco	Preço do produto
Quantidade_estoque	Quantidade do produto em estoque.
Foto	Foto do produto.

Tabela	Prepara
Descrição	Informações sobre o preparo do pedido.
Atributos	
Nome	**Descrição**
Quantidade	Quantidade de produtos.
Pedido_idPedido	Código de identificação do pedido que está sendo preparado.
Produto_idProduto	Código de identificação do produto que foi pedido.

```
-- MySQL Workbench Forward Engineering

-- DROP DATABASE mydb;
```

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE
,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';



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

-- Schema mydb

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



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

-- Schema mydb

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

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;



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

-- Table `mydb`.`Cliente`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Cliente` (

  `Nome_usuario` VARCHAR(30) NOT NULL,

  `Senha` VARCHAR(30) NOT NULL,

  `Preferencia` VARCHAR(100) NOT NULL,

  `idUsuario` INT NOT NULL,

  PRIMARY KEY (`idUsuario`),

  UNIQUE INDEX `Nome_usuario_UNIQUE` (`Nome_usuario` ASC))

ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`PessoaJuridica`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`PessoaJuridica` (
  `Cnpj` VARCHAR(14) NOT NULL,
  `Razao_social` VARCHAR(80) NOT NULL,
  `Rua` VARCHAR(45) NOT NULL,
  `Bairro` VARCHAR(20) NOT NULL,
  `Cidade` VARCHAR(20) NOT NULL,
  `Estado` VARCHAR(20) NOT NULL,
  `Cep` INT(8) NOT NULL,
  `Telefone` VARCHAR(13) NOT NULL,
  `Numero_ende` VARCHAR(10) NOT NULL,
  `Cliente_idUsuario` INT NULL,
  PRIMARY KEY (`Cnpj`),
  INDEX `fk_PessoaJuridica_Cliente1_idx` (`Cliente_idUsuario` ASC),
  UNIQUE INDEX `Cliente_idUsuario_UNIQUE` (`Cliente_idUsuario` ASC),
  CONSTRAINT `fk_PessoaJuridica_Cliente1`
    FOREIGN KEY (`Cliente_idUsuario`)
    REFERENCES `mydb`.`Cliente` (`idUsuario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`PessoaFisica`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`PessoaFisica` (
  `Cpf` VARCHAR(14) NOT NULL,
  `Nome` VARCHAR(80) NOT NULL,
  `Email` VARCHAR(50) NOT NULL,
  `Rua` VARCHAR(45) NOT NULL,
  `Bairro` VARCHAR(20) NOT NULL,
  `Cidade` VARCHAR(20) NOT NULL,
  `Estado` VARCHAR(20) NOT NULL,
  `Cep` INT(20) NOT NULL,
  `Telefone` VARCHAR(13) NOT NULL,
  `Numero_ende` VARCHAR(10) NOT NULL,
  `Cliente_idUsuario` INT NULL,
  PRIMARY KEY (`Cpf`),
  INDEX `fk_PessoaFisica_Cliente1_idx` (`Cliente_idUsuario` ASC),
  UNIQUE INDEX `Cliente_idUsuario_UNIQUE` (`Cliente_idUsuario` ASC),
  CONSTRAINT `fk_PessoaFisica_Cliente1`
    FOREIGN KEY (`Cliente_idUsuario`)
    REFERENCES `mydb`.`Cliente` (`idUsuario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Restaurante`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Restaurante` (
```

```
  `Tipo_culinaria` VARCHAR(50) NOT NULL,
  `PessoaJuridica_Cnpj` VARCHAR(14) NOT NULL,
  PRIMARY KEY (`PessoaJuridica_Cnpj`),
  CONSTRAINT `fk_Restaurante_PessoaJuridica1`
    FOREIGN KEY (`PessoaJuridica_Cnpj`)
    REFERENCES `mydb`.`PessoaJuridica` (`Cnpj`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Avaliação`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Avaliação` (
  `idAvaliação` INT(4) NOT NULL,
  `Nota` INT(1) NOT NULL,
  `Comentario` TEXT(200) NULL,
  PRIMARY KEY (`idAvaliação`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Deixa_avaliacao`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Deixa_avaliacao` (
  `Avaliação_idAvaliação` INT(4) NOT NULL,
  `Restaurante_PessoaJuridica_Cnpj` VARCHAR(14) NOT NULL,
```

```
  `Cliente_idUsuario` INT NOT NULL,
  INDEX `fk_Deixa_avaliacao_Avaliação1_idx` (`Avaliação_idAvaliação` ASC),
  PRIMARY KEY (`Avaliação_idAvaliação`),
  INDEX `fk_Deixa_avaliacao_Restaurante1_idx` (`Restaurante_PessoaJuridica_Cnpj` ASC),
  INDEX `fk_Deixa_avaliacao_Cliente1_idx` (`Cliente_idUsuario` ASC),
  CONSTRAINT `fk_Deixa_avaliacao_Avaliação1`
    FOREIGN KEY (`Avaliação_idAvaliação`)
    REFERENCES `mydb`.`Avaliação` (`idAvaliação`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Deixa_avaliacao_Restaurante1`
    FOREIGN KEY (`Restaurante_PessoaJuridica_Cnpj`)
    REFERENCES `mydb`.`Restaurante` (`PessoaJuridica_Cnpj`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Deixa_avaliacao_Cliente1`
    FOREIGN KEY (`Cliente_idUsuario`)
    REFERENCES `mydb`.`Cliente` (`idUsuario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Tipo_pagamento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Tipo_pagamento` (
  `id_forma_pagamento` INT(4) NOT NULL,
```

```
  `tipo` INT(1) NOT NULL DEFAULT 1,
  `detalhe` VARCHAR(100) NULL,
  PRIMARY KEY (`id_forma_pagamento`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Produto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Produto` (
  `idProduto` INT(4) NOT NULL,
  `nome_produto` VARCHAR(80) NOT NULL,
  `foto` VARCHAR(200) NULL,
  `quantidade_estoque` INT(4) NOT NULL,
  `preco` DECIMAL(8,2) NOT NULL,
  `descricao` VARCHAR(45) NULL,
  PRIMARY KEY (`idProduto`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Entregador`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Entregador` (
  `id_entregador` INT(4) NOT NULL,
  `nome_contato` VARCHAR(80) NOT NULL,
  `telefone_contato` VARCHAR(11) NOT NULL,
  `tipo_veiculo` INT(1) NOT NULL,
```

```
  PRIMARY KEY (`id_entregador`))
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`Pedido`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Pedido` (
  `Data_hora` DATE NOT NULL,
  `Estado` VARCHAR(45) NOT NULL,
  `idPedido` INT NOT NULL,
  `Restaurante_PessoaJuridica_Cnpj` VARCHAR(14) NOT NULL,
  `Tipo_pagamento_id_forma_pagamento` INT(4) NOT NULL,
  `Entregador_id_entregador` INT(4) NOT NULL,
  `Cliente_idUsuario` INT NOT NULL,
  PRIMARY KEY (`idPedido`),
  INDEX `fk_Pedido_Restaurante1_idx` (`Restaurante_PessoaJuridica_Cnpj` ASC),
  INDEX `fk_Pedido_Tipo_pagamento1_idx` (`Tipo_pagamento_id_forma_pagamento` ASC),
  INDEX `fk_Pedido_Entregador1_idx` (`Entregador_id_entregador` ASC),
  INDEX `fk_Pedido_Cliente1_idx` (`Cliente_idUsuario` ASC),
  CONSTRAINT `fk_Pedido_Restaurante1`
    FOREIGN KEY (`Restaurante_PessoaJuridica_Cnpj`)
    REFERENCES `mydb`.`Restaurante` (`PessoaJuridica_Cnpj`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pedido_Tipo_pagamento1`
    FOREIGN KEY (`Tipo_pagamento_id_forma_pagamento`)
    REFERENCES `mydb`.`Tipo_pagamento` (`id_forma_pagamento`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pedido_Entregador1`
    FOREIGN KEY (`Entregador_id_entregador`)
    REFERENCES `mydb`.`Entregador` (`id_entregador`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pedido_Cliente1`
    FOREIGN KEY (`Cliente_idUsuario`)
    REFERENCES `mydb`.`Cliente` (`idUsuario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Prepara`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Prepara` (
  `quantidade` INT NOT NULL,
  `Pedido_idPedido` INT NOT NULL,
  `Produto_idProduto` INT(4) NOT NULL,
  INDEX `fk_Prepara_Pedido1_idx` (`Pedido_idPedido` ASC),
  PRIMARY KEY (`Produto_idProduto`, `Pedido_idPedido`),
  CONSTRAINT `fk_Prepara_Pedido1`
    FOREIGN KEY (`Pedido_idPedido`)
    REFERENCES `mydb`.`Pedido` (`idPedido`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Prepara_Produto1`
    FOREIGN KEY (`Produto_idProduto`)
    REFERENCES `mydb`.`Produto` (`idProduto`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;



ALTER TABLE `Deixa_avaliacao` DROP FOREIGN KEY `fk_Deixa_avaliacao_Avaliação1`;
ALTER TABLE `Deixa_avaliacao` DROP FOREIGN KEY
`fk_Deixa_avaliacao_Restaurante1`;
ALTER TABLE `Deixa_avaliacao` DROP FOREIGN KEY `fk_Deixa_avaliacao_Cliente1`;

ALTER TABLE `Prepara` DROP FOREIGN KEY `fk_Prepara_Pedido1`;
ALTER TABLE `Prepara` DROP FOREIGN KEY `fk_Prepara_Produto1`;

ALTER TABLE `Restaurante` DROP FOREIGN KEY `fk_Restaurante_PessoaJuridica1`;

ALTER TABLE `Deixa_avaliacao`
  ADD CONSTRAINT `fk_Deixa_avaliacao_Avaliação1`
  FOREIGN KEY (`Avaliação_idAvaliação`)
  REFERENCES `mydb`.`Avaliação` (`idAvaliação`)
  ON DELETE CASCADE
```

```
  ON UPDATE NO ACTION;


ALTER TABLE `Deixa_avaliacao`
  ADD CONSTRAINT `fk_Deixa_avaliacao_Restaurante1`
  FOREIGN KEY (`Restaurante_PessoaJuridica_Cnpj`)
  REFERENCES `mydb`.`Restaurante` (`PessoaJuridica_Cnpj`)
  ON DELETE CASCADE
  ON UPDATE NO ACTION;


ALTER TABLE `Deixa_avaliacao`
  ADD CONSTRAINT `fk_Deixa_avaliacao_Cliente1`
  FOREIGN KEY (`Cliente_idUsuario`)
  REFERENCES `mydb`.`Cliente` (`idUsuario`)
  ON DELETE CASCADE
  ON UPDATE NO ACTION;


ALTER TABLE `Prepara`
  ADD CONSTRAINT `fk_Prepara_Pedido1`
  FOREIGN KEY (`Pedido_idPedido`)
  REFERENCES `mydb`.`Pedido` (`idPedido`)
  ON DELETE CASCADE
  ON UPDATE NO ACTION;


ALTER TABLE `Prepara`
  ADD CONSTRAINT `fk_Prepara_Produto1`
  FOREIGN KEY (`Produto_idProduto`)
  REFERENCES `mydb`.`Produto` (`idProduto`)
  ON DELETE CASCADE
```

```
  ON UPDATE NO ACTION;


  ALTER TABLE `Restaurante`
  ADD CONSTRAINT `fk_Restaurante_PessoaJuridica1`
  FOREIGN KEY (`PessoaJuridica_Cnpj`)
  REFERENCES `mydb`.`PessoaJuridica` (`Cnpj`)
  ON DELETE CASCADE
  ON UPDATE NO ACTION;


-- -----------------------------------------------------
-- Exemplos de ALTER TABLE e DROP TABLE
-- -----------------------------------------------------

-- Exemplo 1: ALTER TABLE - Adicionando uma coluna
ALTER TABLE mydb.Cliente
ADD COLUMN Data_nascimento DATE AFTER Preferencia;

-- Exemplo 2: ALTER TABLE - Modificando uma coluna
ALTER TABLE mydb.PessoaFisica
MODIFY COLUMN Telefone VARCHAR(15) NOT NULL;

-- Exemplo 3: ALTER TABLE - Removendo uma coluna
ALTER TABLE mydb.Cliente
DROP COLUMN Data_nascimento;

-- Exemplo 4: Criando uma tabela extra chamada 'Funcionarios'
```

```
CREATE TABLE IF NOT EXISTS mydb.Funcionarios (
  idFuncionario INT NOT NULL,
  nomeFuncionario VARCHAR(80) NOT NULL,
  cargo VARCHAR(50) NOT NULL,
  salario DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (idFuncionario)
) ENGINE = InnoDB;

-- Exemplo 5: Excluindo a tabela 'Funcionarios'
DROP TABLE IF EXISTS mydb.Funcionarios;

-- Inserção de dados na tabela Cliente
INSERT INTO `mydb`.`Cliente` (`Nome_usuario`, `Senha`, `Preferencia`, `idUsuario`)
VALUES
('user1', 'senha1', 'vegetariano', 1),
('user2', 'senha2', 'gluten-free', 2),
('user3', 'senha3', 'vegan', 3),
('user4', 'senha4', 'sem lactose', 4),
('user5', 'senha5', 'sem carne', 5);

-- Inserção de dados na tabela PessoaJuridica
INSERT INTO `mydb`.`PessoaJuridica` (`Cnpj`, `Razao_social`, `Rua`, `Bairro`, `Cidade`,
`Estado`, `Cep`, `Telefone`, `Numero_ende`, `Cliente_idUsuario`) VALUES
('12345678000199', 'Restaurante A', 'Rua A', 'Bairro A', 'Cidade A', 'Estado A', 12345678,
'1234567890', '10', 1),
('98765432000188', 'Restaurante B', 'Rua B', 'Bairro B', 'Cidade B', 'Estado B', 23456789,
'2345678901', '20', 2),
('11223344000177', 'Restaurante C', 'Rua C', 'Bairro C', 'Cidade C', 'Estado C', 34567890,
'3456789012', '30', 3),
('22334455000166', 'Restaurante D', 'Rua D', 'Bairro D', 'Cidade D', 'Estado D', 45678901,
'4567890123', '40', 4),
```

('33445566000155', 'Restaurante E', 'Rua E', 'Bairro E', 'Cidade E', 'Estado E', 56789012, '5678901234', '50', 5);

-- Inserção de dados na tabela PessoaFisica

INSERT INTO `mydb`.`PessoaFisica` (`Cpf`, `Nome`, `Email`, `Rua`, `Bairro`, `Cidade`, `Estado`, `Cep`, `Telefone`, `Numero_ende`, `Cliente_idUsuario`) VALUES

('12345678901', 'Pessoa A', 'pessoaA@email.com', 'Rua 1', 'Bairro 1', 'Cidade 1', 'Estado 1', 12345678, '1234567890', '10', 1),

('23456789012', 'Pessoa B', 'pessoaB@email.com', 'Rua 2', 'Bairro 2', 'Cidade 2', 'Estado 2', 23456789, '2345678901', '20', 2),

('34567890123', 'Pessoa C', 'pessoaC@email.com', 'Rua 3', 'Bairro 3', 'Cidade 3', 'Estado 3', 34567890, '3456789012', '30', 3),

('45678901234', 'Pessoa D', 'pessoaD@email.com', 'Rua 4', 'Bairro 4', 'Cidade 4', 'Estado 4', 45678901, '4567890123', '40', 4),

('56789012345', 'Pessoa E', 'pessoaE@email.com', 'Rua 5', 'Bairro 5', 'Cidade 5', 'Estado 5', 56789012, '5678901234', '50', 5);

-- Inserção de dados na tabela Restaurante

INSERT INTO `mydb`.`Restaurante` (`Tipo_culinaria`, `PessoaJuridica_Cnpj`) VALUES

('Italiana', '12345678000199'),

('Japonesa', '98765432000188'),

('Brasileira', '11223344000177'),

('Mexicana', '22334455000166'),

('Francesa', '33445566000155');

-- Inserção de dados na tabela Avaliação

INSERT INTO `mydb`.`Avaliação` (`idAvaliação`, `Nota`, `Comentario`) VALUES

(4, 2, 'Comida razoável, mas o ambiente deixou a desejar.'),

(6, 5, NULL),

(1, 5, 'Excelente!'),

(2, 4, 'Muito bom, mas pode melhorar.'),

(3, 3, 'Bom, porém o atendimento foi demorado.'),

(5, 1, 'Muito ruim, não gostei da comida.');

-- Inserção de dados na tabela Deixa_avaliacao

INSERT INTO `mydb`.`Deixa_avaliacao` (`Avaliação_idAvaliação`, `Restaurante_PessoaJuridica_Cnpj`, `Cliente_idUsuario`) VALUES

(2, '98765432000188', 2),

(5, '33445566000155', 5),

(1, '12345678000199', 1),

(3, '11223344000177', 3),

(4, '22334455000166', 4);

-- Inserção de dados na tabela Tipo_pagamento

INSERT INTO `mydb`.`Tipo_pagamento` (`id_forma_pagamento`, `tipo`, `detalhe`) VALUES

(1, 1, 'Cartão de crédito'),

(2, 2, 'Débito'),

(3, 1, 'Dinheiro'),

(4, 2, 'Pix'),

(5, 1, 'Criptomoeda');

-- Inserção de dados na tabela Produto

INSERT INTO `mydb`.`Produto` (`idProduto`, `nome_produto`, `foto`, `quantidade_estoque`, `preco`, `descricao`) VALUES

(2, 'Sushi', 'sushi.jpg', 200, 50.00, 'Sushi variado'),

(1, 'Pizza', 'pizza.jpg', 100, 30.00, 'Pizza de mussarela'),

(3, 'Feijoada', 'feijoada.jpg', 50, 40.00, 'Feijoada completa'),

(4, 'Taco', 'taco.jpg', 120, 25.00, 'Taco de carne'),

(5, 'Croissant', 'croissant.jpg', 80, 15.00, 'Croissant de chocolate');

-- Inserção de dados na tabela Entregador

INSERT INTO `mydb`.`Entregador` (`id_entregador`, `nome_contato`, `telefone_contato`, `tipo_veiculo`) VALUES

(1, 'João', '9876543210', 1),

(2, 'Maria', '9876543211', 2),

(3, 'Pedro', '9876543212', 1),

(4, 'Ana', '9876543213', 2),

(5, 'Lucas', '9876543214', 1);

-- Inserção de dados na tabela Pedido

INSERT INTO `mydb`.`Pedido` (`Data_hora`, `Estado`, `idPedido`, `Restaurante_PessoaJuridica_Cnpj`, `Tipo_pagamento_id_forma_pagamento`, `Entregador_id_entregador`, `Cliente_idUsuario`) VALUES

('2024-12-26', 'Finalizado', 2, '98765432000188', 2, 2, 2),

('2024-12-27', 'Em entrega', 3, '11223344000177', 3, 3, 3),

('2024-12-27', 'Em entrega', 6, '11223344000177', 3, 3, 5),

('2024-12-28', 'Em preparo', 7, '22334455000166', 4, 4, 5),

('2024-12-29', 'Cancelado', 8, '33445566000155', 5, 5, 5),

('2024-12-29', 'Cancelado', 5, '33445566000155', 5, 5, 5),

('2024-12-25', 'Em preparo', 1, '12345678000199', 1, 1, 1),

('2024-12-28', 'Em preparo', 4, '22334455000166', 4, 4, 4);

-- Inserção de dados na tabela Prepara

INSERT INTO `mydb`.`Prepara` (`quantidade`, `Pedido_idPedido`, `Produto_idProduto`) VALUES

(2, 1, 1),

(1, 3, 3),

(1, 1, 4),

(3, 2, 2),

```
(5, 4, 4),
(4, 5, 5);

-- Exemplos de Modificação de Dados (UPDATE)

-- 1. Alterando a Preferência de um Cliente
UPDATE `mydb`.`Cliente`
SET `Preferencia` = 'sem glúten'
WHERE `idUsuario` = 1;

-- 2. Alterando o Nome de um Produto
UPDATE `mydb`.`Produto`
SET `nome_produto` = 'Pizza Margherita'
WHERE `idProduto` = 1;

-- 3. Alterando o Estado de um Pedido
UPDATE `mydb`.`Pedido`
SET `Estado` = 'Concluído'
WHERE `idPedido` = 1;

-- 4. Modificando a Nota de Avaliação
UPDATE `mydb`.`Avaliação`
SET `Nota` = 4
WHERE `idAvaliação` = 1;

-- 5. UPDATE Aninhado (Alterando Dados em Duas Tabelas)
UPDATE `mydb`.`Restaurante` r
JOIN `mydb`.`PessoaJuridica` p ON r.PessoaJuridica_Cnpj = p.Cnpj
```

```
SET r.Tipo_culinaria = 'Mediterrânea', p.Cidade = 'Nova Cidade'

WHERE p.Cnpj = '12345678000199';


-- Exemplos de Exclusão de Dados (DELETE)


-- 1. Exclui da tabela deixa_avaliação

DELETE FROM `mydb`.`Deixa_avaliacao`

WHERE `Cliente_idUsuario` = 5;


-- 2. Excluindo um Produto

DELETE FROM `mydb`.`Produto`

WHERE `idProduto` = 2;


-- 3. Excluindo uma Avaliação

DELETE FROM `mydb`.`Avaliação`

WHERE `idAvaliação` = 4;


-- 4. Excluindo um Pedido

DELETE FROM `mydb`.`Pedido`

WHERE `idPedido` = 3;


-- 5. DELETE Aninhado (Excluindo das tabelas deixa_avaliação e pedido usuarios e tem id=2)

DELETE p, da

FROM `mydb`.`Pedido` p

JOIN `mydb`.`Deixa_avaliacao` da ON p.idPedido = da.Avaliação_idAvaliação

JOIN `mydb`.`Cliente` c ON p.Cliente_idUsuario = c.idUsuario

WHERE c.idUsuario = 2;
```

```
-- F1: Consulta com INNER JOIN

-- Recupera os nomes dos clientes e os tipos de culinária dos restaurantes onde o cliente deixou uma avaliação

SELECT c.Nome_usuario, r.Tipo_culinaria

FROM `mydb`.`Cliente` c

INNER JOIN `mydb`.`Deixa_avaliacao` da ON c.idUsuario = da.Cliente_idUsuario

INNER JOIN `mydb`.`Restaurante` r ON da.Restaurante_PessoaJuridica_Cnpj = r.PessoaJuridica_Cnpj;


-- F2: Consulta com OUTER JOIN

-- Recupera todos os clientes e, caso existam, os pedidos feitos por eles. Caso o cliente não tenha feito nenhum pedido, ele ainda será listado.

SELECT c.Nome_usuario, p.idPedido

FROM `mydb`.`Cliente` c

LEFT OUTER JOIN `mydb`.`Pedido` p ON c.idUsuario = p.Cliente_idUsuario;


-- F3: Consulta com ORDER BY

-- Recupera todos os pedidos ordenados pela data em ordem crescente e, em caso de empate, ordena por idPedido em ordem decrescente

SELECT idPedido, Data_hora, Estado

FROM `mydb`.`Pedido`

ORDER BY Data_hora ASC, idPedido DESC;


-- F4: Consulta com GROUP BY

-- Recupera a quantidade de pedidos feitos por cada cliente

SELECT Cliente_idUsuario, COUNT(*) AS quantidade_pedidos

FROM `mydb`.`Pedido`

GROUP BY Cliente_idUsuario;


-- F5: Consulta com HAVING
```

```
-- Recupera os clientes que fizeram mais de 5 pedidos
SELECT Cliente_idUsuario, COUNT(*) AS quantidade_pedidos
FROM `mydb`.`Pedido`
GROUP BY Cliente_idUsuario
HAVING COUNT(*) > 3;

-- F6: Consulta com UNION
-- Recupera os CNPJs de restaurantes e os CPFs de pessoas físicas que são clientes
SELECT Cnpj AS identificador FROM `mydb`.`PessoaJuridica`
UNION
SELECT Cpf AS identificador FROM `mydb`.`PessoaFisica`;

-- F7: Consulta com IN
-- Recupera os pedidos realizados por clientes com idUsuario igual a 1, 2 ou 3
SELECT idPedido, Data_hora, Estado
FROM `mydb`.`Pedido`
WHERE Cliente_idUsuario IN (1, 2, 3);

-- F8: Consulta com LIKE
-- Recupera os nomes de produtos que contêm a palavra "Pizza" no nome
SELECT nome_produto
FROM `mydb`.`Produto`
WHERE nome_produto LIKE '%Pizza%';

-- F9: Consulta com IS NULL
-- Recupera todas as avaliações que têm comentários nulos
SELECT idAvaliação, Nota, Comentario
FROM `mydb`.`Avaliação`
```

```
WHERE Comentario IS NULL;


-- F10: Consulta com ANY

-- Recupera todos os pedidos cujo valor total (baseado na quantidade e preço de cada produto) seja maior que o preço de qualquer pedido feito pelo cliente de idUsuario = 1


SELECT p.idPedido, p.Data_hora

FROM `mydb`.`Pedido` p

JOIN `mydb`.`Prepara` pr ON p.idPedido = pr.Pedido_idPedido

JOIN `mydb`.`Produto` prod ON pr.Produto_idProduto = prod.idProduto

GROUP BY p.idPedido

HAVING SUM(pr.quantidade * prod.preco) > ANY (

   -- Subconsulta para obter os pedidos do cliente 1 e calcular o valor total desses pedidos

   SELECT SUM(pr2.quantidade * prod2.preco)

   FROM `mydb`.`Pedido` p2

   JOIN `mydb`.`Prepara` pr2 ON p2.idPedido = pr2.Pedido_idPedido

   JOIN `mydb`.`Produto` prod2 ON pr2.Produto_idProduto = prod2.idProduto

   WHERE p2.Cliente_idUsuario = 1

   GROUP BY p2.idPedido

);


-- F11: Consulta com ALL

-- Recupera todos os pedidos cujo valor total (baseado na quantidade e preço de cada produto) seja maior que o valor de todos os pedidos feitos pelo cliente de idUsuario = 1


SELECT p.idPedido, p.Data_hora

FROM `mydb`.`Pedido` p

JOIN `mydb`.`Prepara` pr ON p.idPedido = pr.Pedido_idPedido

JOIN `mydb`.`Produto` prod ON pr.Produto_idProduto = prod.idProduto
```

```
GROUP BY p.idPedido

HAVING SUM(pr.quantidade * prod.preco) > ALL (

    -- Subconsulta para obter os pedidos do cliente 1 e calcular o valor total desses pedidos

    SELECT SUM(pr2.quantidade * prod2.preco)

    FROM `mydb`.`Pedido` p2

    JOIN `mydb`.`Prepara` pr2 ON p2.idPedido = pr2.Pedido_idPedido

    JOIN `mydb`.`Produto` prod2 ON pr2.Produto_idProduto = prod2.idProduto

    WHERE p2.Cliente_idUsuario = 1

    GROUP BY p2.idPedido

);




-- F12: Consulta com EXISTS

-- Recupera os restaurantes para os quais há pedidos feitos por clientes, ou seja, restaurantes que têm pedidos realizados

SELECT DISTINCT r.Tipo_culinaria

FROM `mydb`.`Restaurante` r

WHERE EXISTS (

    SELECT 1

    FROM `mydb`.`Pedido` p

    WHERE p.Restaurante_PessoaJuridica_Cnpj = r.PessoaJuridica_Cnpj

);



-- F13: Consulta adicional com operadores AND, OR, NOT e BETWEEN

-- Recupera os pedidos feitos por clientes cujo estado do pedido seja 'Em andamento' ou 'Finalizado', e que foram feitos entre 1 de janeiro de 2024 e 30 de junho de 2024.

SELECT idPedido, Data_hora, Estado

FROM `mydb`.`Pedido`
```

```
WHERE (Estado = 'Em andamento' OR Estado = 'Finalizado')

AND Data_hora BETWEEN '2024-01-01' AND '2024-06-30';


-- F14: Consulta adicional com operadores NOT e LIKE

-- Recupera todos os produtos cujo nome não contenha a palavra 'bacon'

SELECT nome_produto

FROM `mydb`.`Produto`

WHERE nome_produto NOT LIKE '%bacon%';


-- F15: Consulta para calcular o valor total de cada pedido

SELECT p.idPedido, SUM(pr.quantidade * prod.preco) AS Valor_Total

FROM `mydb`.`Pedido` p

JOIN `mydb`.`Prepara` pr ON p.idPedido = pr.Pedido_idPedido

JOIN `mydb`.`Produto` prod ON pr.Produto_idProduto = prod.idProduto

GROUP BY p.idPedido;



-- Procedimentos/Funções


DELIMITER $$

-- Este procedimento verifica se um produto específico tem estoque suficiente, e em seguida retorna uma mensagem com base na quantidade disponível.

CREATE PROCEDURE VerificarEstoqueProduto(IN p_idProduto INT, IN p_quantidade INT)

BEGIN

    DECLARE v_estoque INT;


    -- Obtém a quantidade em estoque do produto

    SELECT quantidade_estoque INTO v_estoque

    FROM Produto
```

```
    WHERE idProduto = p_idProduto;

    -- Verifica se há estoque suficiente
    IF v_estoque >= p_quantidade THEN
        SELECT 'Estoque suficiente' AS Resultado;
    ELSE
        SELECT 'Estoque insuficiente' AS Resultado;
    END IF;
END$$

DELIMITER ;

CALL VerificarEstoqueProduto(1, 10);

DELIMITER $$

-- Este procedimento utiliza o CASE WHEN para categorizar um pedido como "baixo valor", "médio valor" ou "alto valor", com base no total calculado dos produtos em um pedido.
CREATE PROCEDURE CategorizarPedidoPorValor(IN p_idPedido INT)
BEGIN
    DECLARE v_total DECIMAL(10,2);

    -- Calcula o valor total do pedido
    SELECT SUM(P.preco * Pr.quantidade)
    INTO v_total
    FROM Produto P
    JOIN Prepara Pr ON P.idProduto = Pr.Produto_idProduto
    WHERE Pr.Pedido_idPedido = p_idPedido;
```

```
    -- Classifica o pedido com base no valor total
    SELECT CASE
        WHEN v_total < 50 THEN 'baixo valor'
        WHEN v_total BETWEEN 50 AND 200 THEN 'médio valor'
        ELSE 'alto valor'
    END AS Categoria;
END$$

DELIMITER ;
CALL CategorizarPedidoPorValor(1);



DELIMITER $$

-- Criação da nova função de valor total do pedido
CREATE FUNCTION CalcularValorTotalPedido(p_idPedido INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE v_total DECIMAL(10,2);

    -- Calcula o valor total do pedido somando preço * quantidade dos produtos
    SELECT SUM(P.preco * Pr.quantidade)
    INTO v_total
    FROM Produto P
    JOIN Prepara Pr ON P.idProduto = Pr.Produto_idProduto
    WHERE Pr.Pedido_idPedido = p_idPedido;
```

```
    RETURN v_total;
END$$

DELIMITER ;

SELECT CalcularValorTotalPedido(1) AS Total_Pedido;



 -- triggers

CREATE TABLE IF NOT EXISTS Notificacoes (
    idNotificacao INT AUTO_INCREMENT PRIMARY KEY,
    idPedido INT NOT NULL,
    idUsuario INT NOT NULL,
    Mensagem TEXT,
    DataEnvio TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- trigger de alteracao
DELIMITER $$

CREATE TRIGGER AfterUpdatePedidoNotificacao
AFTER UPDATE ON Pedido
FOR EACH ROW
BEGIN
    DECLARE mensagem TEXT;
```

```
    -- Verifica se o estado do pedido foi alterado
    IF OLD.Estado <> NEW.Estado THEN
        -- Define a mensagem de notificação com base no novo estado
        CASE NEW.Estado
            WHEN 'Em preparo' THEN
                SET mensagem = CONCAT('Seu pedido #', NEW.idPedido, ' está em preparo.');
            WHEN 'Em entrega' THEN
                SET mensagem = CONCAT('Seu pedido #', NEW.idPedido, ' saiu para entrega.');
            WHEN 'Finalizado' THEN
                SET mensagem = CONCAT('Seu pedido #', NEW.idPedido, ' foi entregue.
Obrigado!');
            WHEN 'Cancelado' THEN
                SET mensagem = CONCAT('Seu pedido #', NEW.idPedido, ' foi cancelado.');
            ELSE
                SET mensagem = CONCAT('O estado do seu pedido #', NEW.idPedido, ' foi alterado
para: ', NEW.Estado);
        END CASE;

        -- Insere a notificação na tabela Notificacoes
        INSERT INTO Notificacoes (idPedido, idUsuario, Mensagem)
        VALUES (NEW.idPedido, NEW.Cliente_idUsuario, mensagem);
    END IF;
END$$

DELIMITER ;

UPDATE Pedido
SET Estado = 'Finalizado'
```

```
WHERE idPedido = 1;

SELECT * FROM Notificacoes;



-- DROP TRIGGER BeforeDeleteProduto;

DELIMITER $$

CREATE TRIGGER BeforeDeleteProduto
BEFORE DELETE ON Produto
FOR EACH ROW
BEGIN
    DECLARE pedidosAtivos INT;

    -- Verifica se o produto está em pedidos ativos (não finalizados ou cancelados)
    SELECT COUNT(*) INTO pedidosAtivos
    FROM Prepara
    WHERE Produto_idProduto = OLD.idProduto
      AND Pedido_idPedido IN (
          SELECT idPedido
          FROM Pedido
          WHERE Estado NOT IN ('Finalizado', 'Cancelado')
      );

    -- Se houver pedidos ativos com esse produto, impede a exclusão
    IF pedidosAtivos > 0 THEN
        SIGNAL SQLSTATE '45000'
```

```
        SET MESSAGE_TEXT = 'Ele está em pedidos ativos.';
    END IF;
END$$

DELIMITER ;



DELETE FROM `mydb`.`Produto`
WHERE `idProduto` = 3;

-- terceiro trigger de inserção
CREATE TABLE IF NOT EXISTS HistoricoPreco (
    idHistorico INT AUTO_INCREMENT PRIMARY KEY,
    idProduto INT NOT NULL,
    PrecoAntigo DECIMAL(8, 2) NOT NULL,
    PrecoNovo DECIMAL(8, 2) NOT NULL,
    DataAlteracao TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

DELIMITER $$

CREATE TRIGGER AfterUpdatePrecoProduto
AFTER UPDATE ON Produto
FOR EACH ROW
BEGIN
    -- Verifica se o preço do produto foi alterado
    IF OLD.preco <> NEW.preco THEN
        -- Insere um registro na tabela de histórico de preços
```

```
        INSERT INTO HistoricoPreco (idProduto, PrecoAntigo, PrecoNovo)
        VALUES (OLD.idProduto, OLD.preco, NEW.preco);
    END IF;
END$$

DELIMITER ;

UPDATE Produto
SET preco = 35.00
WHERE idProduto = 1;

SELECT * FROM HistoricoPreco;

-- View 1: Usuários e suas preferências

CREATE VIEW mydb.vw_clientes_preferencias AS
SELECT Nome_usuario, Preferencia
FROM mydb.Cliente;


SELECT * FROM vw_clientes_preferencias WHERE Preferencia = 'vegan';

-- View 2: Avaliações detalhadas
CREATE VIEW mydb.vw_avaliacoes_detalhadas AS
SELECT
    c.Nome_usuario AS Cliente,
    r.Tipo_culinaria AS Culinaria,
    a.Nota,
```

```
    a.Comentario
FROM mydb.Deixa_avaliacao da
INNER JOIN mydb.Avaliação a ON da.Avaliação_idAvaliação = a.idAvaliação
INNER JOIN mydb.Cliente c ON da.Cliente_idUsuario = c.idUsuario
INNER JOIN mydb.Restaurante r ON da.Restaurante_PessoaJuridica_Cnpj =
r.PessoaJuridica_Cnpj;

SELECT * FROM vw_avaliacoes_detalhadas WHERE Nota >= 4;

-- View 3: Produtos com estoque baixo
CREATE VIEW mydb.vw_produtos_estoque_baixo AS
SELECT idProduto, nome_produto, quantidade_estoque
FROM mydb.Produto
WHERE quantidade_estoque < 80;

-- DROP VIEW vw_produtos_estoque_baixo;

SELECT * FROM vw_produtos_estoque_baixo;

-- 1. Criação de Usuários
CREATE USER 'usuario1'@'localhost' IDENTIFIED BY 'senha123';
CREATE USER 'usuario2'@'%' IDENTIFIED BY 'senha456';

GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;

-- 2. Concessão de Permissões (GRANT)
-- Concede permissão de leitura (SELECT) em todas as tabelas do banco 'mydb' para 'usuario1'
```

```
GRANT SELECT ON mydb.* TO 'usuario1'@'localhost';


-- Concede permissão de leitura, inserção e atualização (SELECT, INSERT, UPDATE) na tabela 'Cliente' para 'usuario2'

GRANT SELECT, INSERT, UPDATE ON mydb.Cliente TO 'usuario2'@'%';


-- 3. Revogação de Permissões (REVOKE)

-- Revoga permissão de leitura (SELECT) em todas as tabelas do banco 'mydb' para 'usuario1'

REVOKE SELECT ON mydb.* FROM 'usuario1'@'localhost';


-- Revoga permissão de inserção (INSERT) na tabela 'Cliente' para 'usuario2'

REVOKE INSERT ON mydb.Cliente FROM 'usuario2'@'%';


-- 5. Verificando Permissões

-- Mostra as permissões concedidas ao 'usuario1'

SHOW GRANTS FOR 'usuario1'@'localhost';


-- Mostra as permissões concedidas ao 'usuario2'

SHOW GRANTS FOR 'usuario2'@'%';
```