PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

PROJETO DE BANCO DE DADOS

2ª PARTE

Panificadora Pandoka

Equipe:

Rafael Galafassi

Marcos Vinicius

Vinicius Costa

Vinicius Viana

Curitiba

Novembro de 2023

Sumário

1 Domínio de Aplicação para o Banco de Dados

1.1 Identificação do projeto

Panificadora Pandoka

Pandoka

1.2 Tema do Projeto

A área de atuação selecionada para este projeto é uma panificadora, um estabelecimento no setor de comércio alimentício, especializado na produção e comercialização de uma ampla gama de produtos de panificação, incluindo pães, bolos, doces, salgados e itens correlatos.

O objetivo central do projeto é desenvolver e implementar um sistema abrangente de gestão para a panificadora. Esse sistema abrangerá áreas cruciais como o ponto de vendas, controle de estoque e a gestão operacional, visando aprimorar a eficiência e a organização de todas as operações do estabelecimento.

1.3 Usuários do sistema

- Ambiente Externo - Clientes:

Os principais usuários neste contexto são os clientes, representando o ambiente externo do sistema. Eles interagem com a panificadora como consumidores finais, buscando uma variedade de produtos de panificação para atender às suas necessidades alimentares e preferências pessoais. Os clientes desempenham um papel vital no sistema, influenciando as demandas e as preferências de compra que direcionam as operações e a oferta de produtos.

- Ambiente Interno - Lojistas:

Os lojistas, representando o ambiente interno do sistema, são os funcionários e gestores diretamente envolvidos nas operações da panificadora. Eles desempenham funções-chave no ambiente interno, como a gestão do estoque, atendimento ao cliente, preparação dos produtos, administração financeira e supervisão das operações diárias. São responsáveis por garantir a eficiência operacional e a satisfação do cliente, contribuindo diretamente para o sucesso e a produtividade da panificadora.

1.4 Funcionalidade 1 do Projeto

- Sistema de Pedidos de Produtos para Clientes:

A funcionalidade proposta visa facilitar e aprimorar a experiência do cliente ao possibilitar a implementação de um sistema de pedidos de produtos. Utilizando os dados armazenados no banco de dados (BD), os usuários terão a capacidade de selecionar, visualizar e adicionar produtos a um carrinho de compras. Essa funcionalidade permitirá que os clientes realizem um ou vários pedidos de forma conveniente e eficiente.

Catálogo de Produtos: Disponibilizar um catálogo completo de produtos disponíveis na panificadora, com detalhes, e informações relevantes.

Adição ao Carrinho de Compras: Permitir aos clientes adicionar produtos desejados ao carrinho de compras, possibilitando a visualização e a gestão dos itens selecionados.

Gestão de Pedidos: Após a seleção dos produtos, viabilizar o processo de finalização do pedido, permitindo revisão, e confirmação antes do envio.

Integração com BD: Utilizar o banco de dados para armazenar informações sobre os produtos selecionados, detalhes do pedido e histórico de compras dos clientes.

1.5 Funcionalidade 2 do Projeto

- Controle de Catálogo de Produtos para Funcionários:

Essa funcionalidade proposta visa permitir que os funcionários, responsáveis pela administração e gestão interna da panificadora, tenham a capacidade de realizar alterações no catálogo de produtos. Isso inclui a adição, remoção ou modificação de itens disponíveis para venda no sistema.

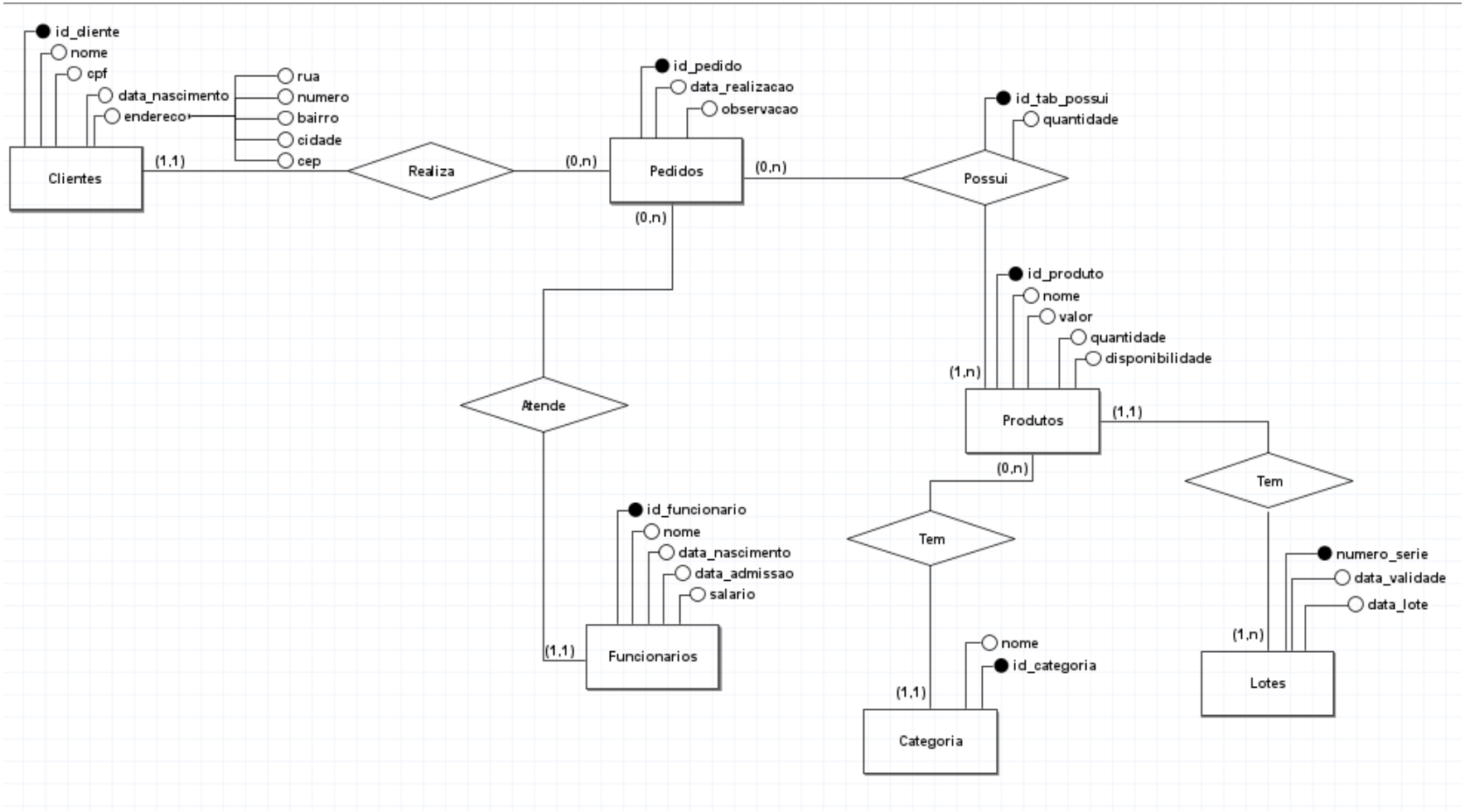
Acesso e Permissões: Conceder acesso restrito e controlado aos funcionários autorizados, com permissões específicas para realizar alterações no catálogo de produtos.

Interface de Gerenciamento: Desenvolver uma interface intuitiva e amigável para permitir a fácil modificação do catálogo de produtos. Isso pode envolver um painel de controle ou um sistema de administração no qual os funcionários possam adicionar, editar ou remover produtos.

Registro de Atividades: Registrar e rastrear todas as alterações realizadas no catálogo de produtos, incluindo informações sobre quem fez a modificação, quando e qual foi a alteração efetuada. Isso é fundamental para a transparência e para rastrear eventuais problemas ou discrepâncias.

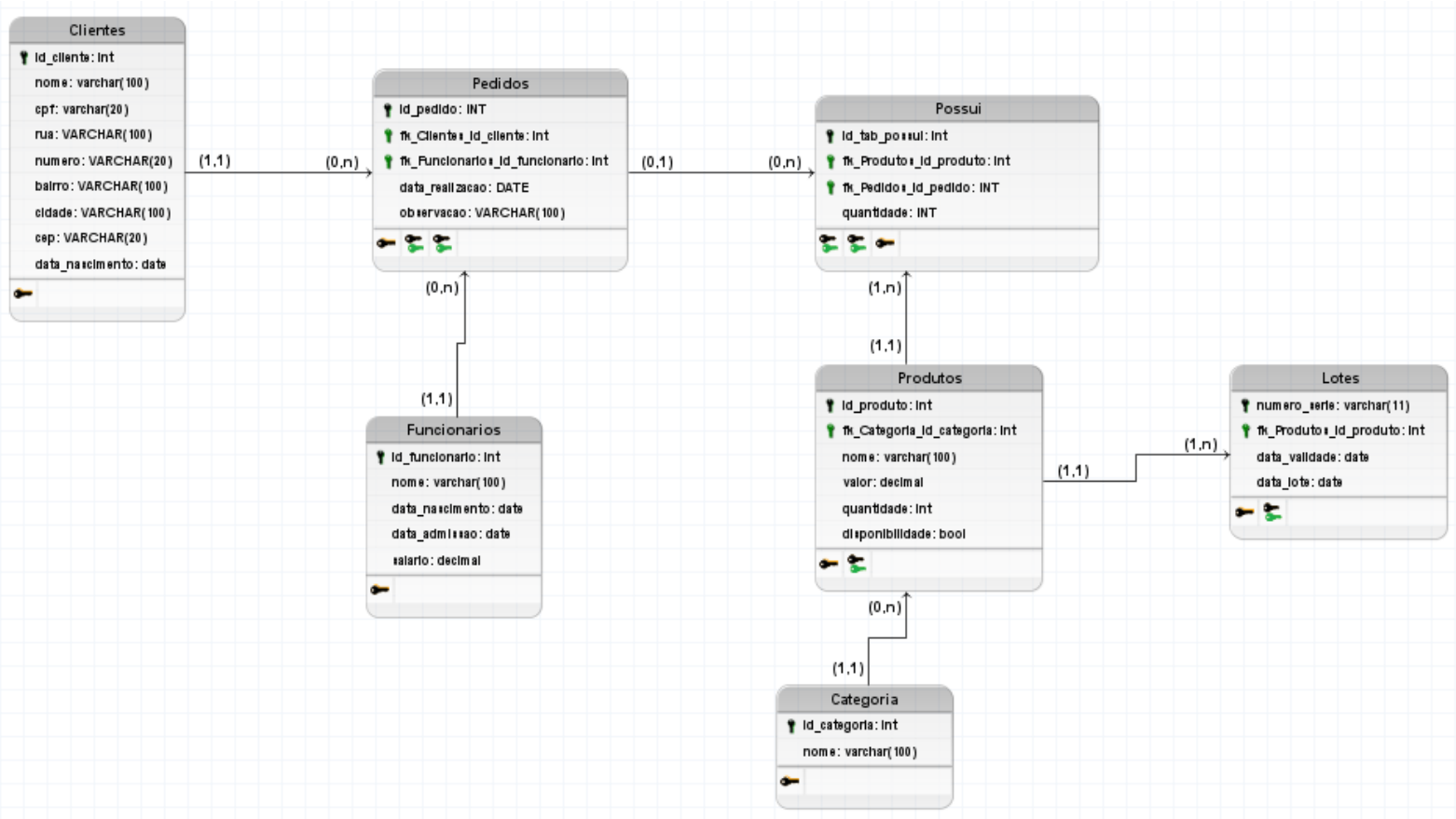
2 Modelo Conceitual

Imagem de boa resolução do Modelo Conceitual (**Modelo Entidade-Relacionamento - MER**) na **3FN**



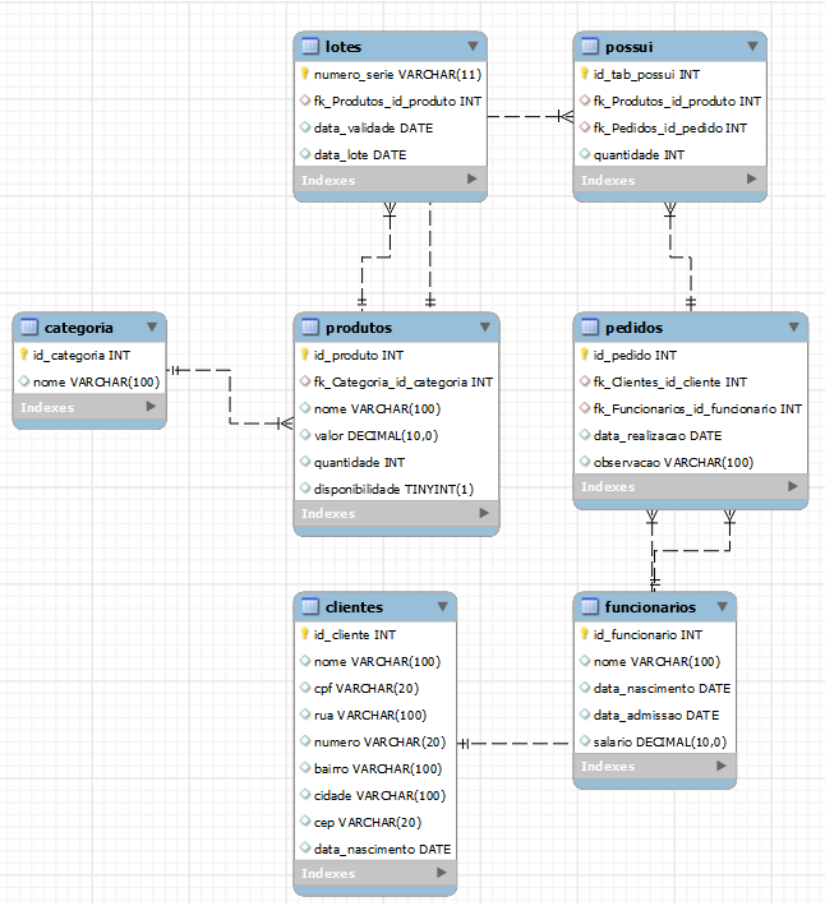
3 Modelo Lógico (3FN)

Imagem de boa resolução do Modelo Lógico (**Modelo Relacional - MR**) na **3FN**, feito com o **brModelo**.



4 Modelo Lógico (3FN)

Imagem de boa resolução do Modelo Lógico (**Modelo Relacional - MR**) na **3FN**, feito com o **ENGENHARIA REVERSA do MySQL Workbench**.



5 Modelo Físico (3FN)

5.1 SQL para criação de tabelas e restrições

```
/* Lógico_1: */

CREATE TABLE Clientes (
    id_cliente int PRIMARY KEY,
    nome varchar(100),
    cpf varchar(11),
    rua varchar(100),
    numero int,
    bairro varchar(100),
    cidade varchar(100),
    cep varchar(11),
    data_nascimento date
);

CREATE TABLE Pedidos (
    id_pedido int PRIMARY KEY,
    data_realizacao date,
    observacao varchar(100),
    fk_Clientes_id_cliente int,
    fk_Funcionarios_id_funcionario int
);

CREATE TABLE Produtos (
    id_produto int PRIMARY KEY,
    nome varchar(100),
    valor decimal,
    quantidade int,
    disponibilidade bool,
```

```
    fk_Categoria_id_categoria int
);

CREATE TABLE Funcionarios (
    id_funcionario int PRIMARY KEY,
    nome varchar(100),
    data_nascimento date,
    data_admissao date,
    salario decimal
);

CREATE TABLE Categoria (
    id_categoria int PRIMARY KEY,
    nome varchar(100)
);

CREATE TABLE Lotes (
    numero_serie varchar(11) PRIMARY KEY,
    data_validade date,
    data_lote date,
    fk_Produtos_id_produto int
);

CREATE TABLE Possui (
    fk_Produtos_id_produto int,
    fk_Pedidos_id_pedido int,
    quantidade int
);

ALTER TABLE Pedidos ADD CONSTRAINT FK_Pedidos_2
```

```
    FOREIGN KEY (fk_Clientes_id_cliente)
    REFERENCES Clientes (id_cliente)
    ON DELETE CASCADE;

ALTER TABLE Pedidos ADD CONSTRAINT FK_Pedidos_3
    FOREIGN KEY (fk_Funcionarios_id_funcionario)
    REFERENCES Funcionarios (id_funcionario)
    ON DELETE CASCADE;

ALTER TABLE Produtos ADD CONSTRAINT FK_Produtos_2
    FOREIGN KEY (fk_Categoria_id_categoria)
    REFERENCES Categoria (id_categoria)
    ON DELETE CASCADE;

ALTER TABLE Lotes ADD CONSTRAINT FK_Lotes_2
    FOREIGN KEY (fk_Produtos_id_produto)
    REFERENCES Produtos (id_produto)
    ON DELETE RESTRICT;

ALTER TABLE Possui ADD CONSTRAINT FK_Possui_1
    FOREIGN KEY (fk_Produtos_id_produto)
    REFERENCES Produtos (id_produto)
    ON DELETE RESTRICT;

ALTER TABLE Possui ADD CONSTRAINT FK_Possui_2
    FOREIGN KEY (fk_Pedidos_id_pedido)
    REFERENCES Pedidos (id_pedido)
    ON DELETE SET NULL;
```

5.2 SQL para inserção de pelo menos 15 registros para cada tabela

Importante: Indicar em imagens de boa resolução, o resultado de cada comando de INSERT

Insert 01 (Tabela Clientes): Colocar SQL e o que faz!

id_cliente	nome	cpf	rua	numero	bairro	cidade	cep	data_nascimento
1	Enrico Ricardo Bento da Conceição	44733799225	Estrada Transamazônica, s/n	214	Sucunduri	Apuí	69275970	2001-07-17
2	Lorenzo Mário Novaes	63330296291	Estrada Transamazônica, s/n	829	Sucunduri	Apuí	69275970	2001-09-10
3	Julio Cláudio Oliver Souza	21587939290	Avenida 13 de Novembro 850	872	Centro	Apuí	69265970	2001-01-13
4	Lorenzo Sérgio da Luz	09927098227	Estrada Transamazônica, s/n	853	Sucunduri	Apuí	69275970	2001-03-10
5	Francisco Diego Enzo Baptista	70623119285	Estrada Transamazônica, s/n	985	Sucunduri	Apuí	69275970	2001-08-25
6	Erick Fernando Pinto	09057788209	Estrada Transamazônica, s/n	668	Sucunduri	Apuí	69275970	2001-01-14
7	Thales Márcio Murilo Araújo	90897248260	Avenida 13 de Novembro 850	563	Centro	Apuí	69265970	2001-05-07
8	Matheus Augusto Porto	07081493246	Estrada Transamazônica, s/n	475	Sucunduri	Apuí	69275970	2001-01-04
9	Geraldo Francisco Luís Nogueira	08570690207	Estrada Transamazônica, s/n	419	Sucunduri	Apuí	69275970	2001-04-14
10	Marcelo Alexandre Márcio Martins	71001629205	Estrada Transamazônica, s/n	722	Sucunduri	Apuí	69275970	2001-02-03
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Insert 02 (Tabela Funcionários): Colocar SQL e o que faz!

id_funcionario	nome	data_nascimento	data_admissao	salario
1	Tomás Lorenzo Lopes	2000-01-16	2023-09-24	3500
2	Sebastião Thales Assunção	2000-04-05	2023-09-24	3200
3	Julio Rafael Martin Rodrigues	2000-04-12	2023-09-24	3400
4	Filipe Márcio Davi Nogueira	2000-04-26	2023-09-24	3600
5	Matheus Isaac Daniel Oliveira	2000-04-11	2023-09-24	3300
6	Pedro Henrique Juan Castro	2000-02-19	2023-09-24	3700
7	Bryan Diogo Araújo	2000-02-08	2023-09-24	3100
8	Mateus Sebastião Theo Sales	2000-06-13	2023-09-24	3800
9	Lucas Osvaldo Osvaldo Silveira	2000-08-27	2023-09-24	4000
10	Davi Henry Kevin da Paz	2000-01-09	2023-09-24	3900
NULL	NULL	NULL	NULL	NULL

Insert 03 (Tabela Categorias): Colocar SQL e o que faz!

id_categoria	nome
3	Bolachas
2	Bolos
4	Doces
1	Pães
5	Salgados
NULL	NULL

Insert 04 (Tabela Produtos): Colocar SQL e o que faz!

id_produto	nome	valor	data_validade	quantidade_estoque	disponibilidade	fk_Categoria_id_categoria
1	Pão Francês	2	2024-10-10	100	1	1
2	Pão de Forma Integral	2	2024-10-12	80	1	1
3	Bolo de Cenoura	8	2024-10-15	20	1	2
4	Bolo de Chocolate	8	2024-10-15	15	1	2
5	Biscoito de Chocolate	4	2024-11-01	50	1	3
6	Biscoito de Coco	4	2024-11-01	60	1	3
7	Doce de Leite	5	2024-10-30	30	1	4
8	Brigadeiro	3	2024-10-28	40	1	4
9	Salgado de Queijo	3	2024-10-25	25	1	5
10	Salgado de Presunto	3	2024-10-25	30	1	5
11	Croissant	2	2024-10-20	20	1	1
12	Rosquinha	2	2024-10-25	40	1	3
13	Pão de Queijo	3	2024-10-15	30	1	5
14	Torta de Limão	10	2024-10-22	10	1	2
15	Torta de Morango	10	2024-10-22	12	1	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Insert 05 (Tabela Pedidos): Colocar SQL e o que faz!

id_pedido	data_realizacao	observacao	fk_Clientes_id_cliente	fk_Funcionarios_id_funcionario
1	2023-09-24	Entregar na Rua A	1	1
2	2023-09-25	Entregar na Rua B	2	2
3	2023-09-26	Entregar na Rua C	3	3
4	2023-09-27	Entregar na Rua D	4	4
5	2023-09-28	Entregar na Rua E	5	5
6	2023-09-29	Entregar na Rua F	6	6
7	2023-09-30	Entregar na Rua G	7	7
8	2023-10-01	Entregar na Rua H	8	8
9	2023-10-02	Entregar na Rua I	9	9
10	2023-10-03	Entregar na Rua J	10	10
11	2023-10-08	Irei Buscar	10	10
12	2023-04-01	Vou pegar	10	10
13	2023-12-22	Entregar na Rua J	1	8
NULL	NULL	NULL	NULL	NULL

Insert 06 (Tabela Lotes): Colocar SQL e o que faz!

	numero_de_serie	data_lote	data_validade	fk_Produto_id_produto
▶	LOTE2023001	2023-01-15	2023-10-10	1
	LOTE2023002	2023-02-20	2023-10-12	2
	LOTE2023003	2023-03-10	2023-10-15	3
	LOTE2023004	2023-04-05	2023-10-15	4
	LOTE2023005	2023-05-12	2023-11-01	5
	LOTE2023006	2022-05-12	2023-11-01	5
	LOTE2023007	2023-02-15	2023-11-01	5
	LOTE2023008	2023-01-19	2023-11-01	5
	LOTE2023009	2023-07-20	2023-11-01	5
	LOTE2023010	2023-08-28	2023-11-01	5
	LOTE2023011	2023-11-30	2023-11-01	5
	LOTE2023012	2023-12-23	2023-11-01	5
	LOTE2023013	2023-04-09	2023-11-01	5
	LOTE2023014	2023-12-12	2023-11-01	5
	LOTE2023015	2023-05-12	2023-11-01	5
*	NULL	NULL	NULL	NULL

6 Consultas e Programação

Para consultas com referências entre tabelas, usar apenas JOIN.

6.1 SQL para 3 consultas com AGREGAÇÃO de recuperação de dados

Consulta 1:

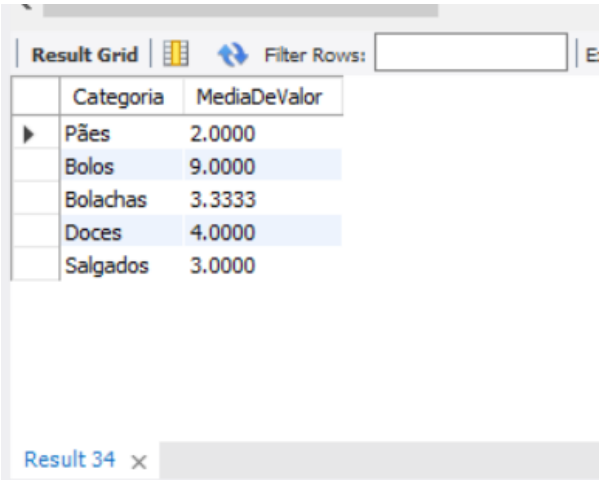
a) Descreva por qual razão a consulta é importante para o usuário.

Essa consulta é valiosa para os usuários que desejam entender a média de preços dos produtos em cada categoria. Ela calcula a média dos valores dos produtos em estoque para cada categoria de produto. Essas informações são úteis para determinar os preços médios de produtos em categorias específicas, permitindo que os usuários ajustem suas estratégias de precificação, identifiquem categorias de alto valor ou baixo valor e tomem decisões informadas sobre gerenciamento de estoque e promoções. Essa consulta ajuda os usuários a otimizar suas estratégias de preços e estoque com base em dados concretos

b) Código SQL

```
SELECT
  C.nome AS Categoria,
  AVG(P.valor) AS MediaDeValor
FROM
  Categoria AS C
JOIN
  Produtos AS P ON C.id_categoria = P.fk_Categoria_id_categoria
GROUP BY
  C.id_categoria;
```

c) Resultado da consulta.



Consulta 2:

a) Descreva por qual razão a consulta é importante para o usuário.

Esta consulta é relevante para os usuários que desejam entender quantos pedidos cada cliente fez. Ela calcula o total de pedidos feitos por cada cliente, permitindo que as empresas identifiquem seus clientes mais ativos e avaliem o envolvimento dos clientes com seus produtos ou serviços. Isso é valioso para estratégias de marketing direcionadas e para o gerenciamento de relacionamento com os clientes.

b) Código SQL.

```
SELECT
  C.id_cliente,
  C.nome AS NomeCliente,
  COUNT(P.id_pedido) AS TotalPedidos
FROM
  Clientes AS C
LEFT JOIN
  Pedidos AS P ON C.id_cliente = P.fk_Clientes_id_cliente
GROUP BY
  C.id_cliente;
```

c) Resultado da consulta.

Result Grid | Filter Rows: | Export:

id_cliente	NomeCliente	TotalPedid
1	Enrico Ricardo Bento da Conceição	2
2	Lorenzo Mário Novaes	1
3	Julio Cláudio Oliver Souza	1
4	Lorenzo Sérgio da Luz	1
5	Francisco Diego Enzo Baptista	1
6	Erick Fernando Pinto	1
7	Thales Márcio Murilo Araújo	1
8	Matheus Augusto Porto	1
9	Geraldo Francisco Luís Nogueira	1
10	Marcelo Alexandre Márcio Martins	3

Result 35

<u>Consulta 3</u>:

a) Descreva por qual razão a consulta é importante para o usuário.

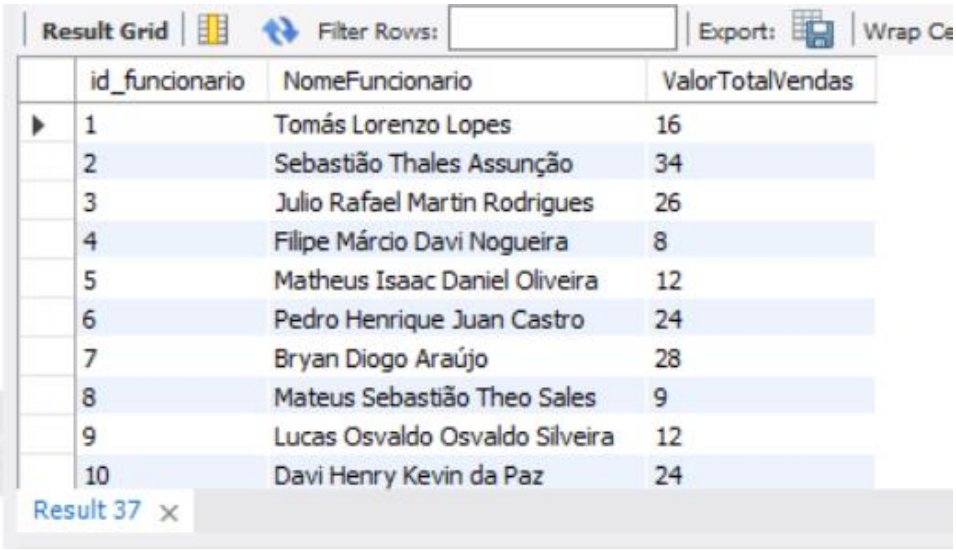
Essa consulta é importante para os usuários que desejam avaliar o desempenho de seus funcionários em termos de vendas. Ela calcula o valor total de vendas gerado por cada funcionário, ajudando as empresas a reconhecer seus principais vendedores e entender como cada funcionário contribui para as receitas da empresa. Isso é fundamental para incentivos de desempenho, distribuição de comissões e gerenciamento eficaz da equipe de vendas.

b) Código SQL.

```
SELECT
   F.id_funcionario,
   F.nome AS NomeFuncionario,
   SUM(IP.quantidade * Pr.valor) AS ValorTotalVendas
FROM
   Funcionarios AS F
 JOIN
   Pedidos AS P ON F.id_funcionario = P.fk_Funcionarios_id_funcionario
JOIN
```

```
    ItensPedido AS IP ON P.id_pedido = IP.fk_Pedidos_id_pedido
JOIN
    Produtos AS Pr ON IP.fk_Produtos_id_produto = Pr.id_produto
GROUP BY
    F.id_funcionario;
```

c) Resultado da consulta.



Result Grid | Filter Rows: | Export: | Wrap Ce

id_funcionario	NomeFuncionario	ValorTotalVendas
1	Tomás Lorenzo Lopes	16
2	Sebastião Thales Assunção	34
3	Julio Rafael Martin Rodrigues	26
4	Filipe Márcio Davi Nogueira	8
5	Matheus Isaac Daniel Oliveira	12
6	Pedro Henrique Juan Castro	24
7	Bryan Diogo Araújo	28
8	Mateus Sebastião Theo Sales	9
9	Lucas Osvaldo Osvaldo Silveira	12
10	Davi Henry Kevin da Paz	24

Result 37

Consulta 4:

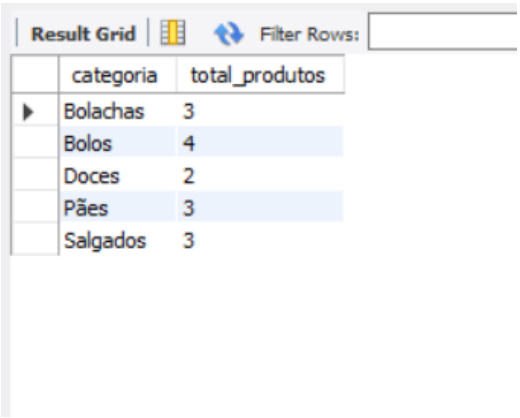
A) Descreva por qual razão a consulta é importante para o usuário.

A consulta SQL é essencial para o usuário acompanhar e analisar o estoque da loja por categoria. Permite avaliar o desempenho relativo das categorias, identificar produtos em excesso ou escassez, e guiar decisões estratégicas, como ajustes no mix de produtos e promoções. É uma ferramenta valiosa para a gestão de inventário, fornecendo insights para otimizar o estoque e melhorar as estratégias de vendas.

B) Código SQL.

```
SELECT
   c.nome AS categoria,
   COUNT(p.id_produto) AS total_produtos
FROM
   Produtos p
JOIN
   Categoria c ON p.fk_Categoria_id_categoria = c.id_categoria

GROUP BY
   c.nome;
```

C) Resultado da consulta.



categoria	total_produtos
Bolachas	3
Bolos	4
Doces	2
Pães	3
Salgados	3

6.2 SQL para 3 consultas com IR (PK + FK) de recuperação de dados

Consulta 1:

a) Descreva por qual razão a consulta é importante para o usuário.

Essa consulta é útil para os usuários que desejam obter um panorama detalhado dos produtos em estoque, suas categorias e os lotes associados. Isso permite um controle minucioso do estoque, rastreamento da origem dos produtos por meio dos lotes e uma visão geral das categorias de produtos disponíveis. Os usuários podem tomar decisões informadas sobre a gestão de estoque, rastrear produtos com base em lotes e categorias e verificar a disponibilidade e informações detalhadas de produtos específicos.

b) Código SQL.

```
SELECT
  P.id_produto,
  P.nome AS nome_produto,
  P.valor,
  P.quantidade_estoque,
  CASE
    WHEN P.disponibilidade = 1 THEN 'Disponível'
    ELSE 'Não Disponível'
  END AS disponibilidade,
  C.id_categoria,
  C.nome AS nome_categoria,
  L.numero_de_serie,
  L.data_validade,
  L.data_lote
FROM
  Produtos AS P
JOIN
  Categoria AS C ON P.fk_Categoria_id_categoria = C.id_categoria
JOIN
  Lotes AS L ON L.fk_Produto_id_produto = P.id_produto;
```

c) Resultado da consulta.

```
419     L.data_lote
420 FROM
421     Produtos AS P
422 JOIN
423     Categoria AS C ON P.fk_Categoria_id_categoria = C.id_categoria
424 JOIN
425     Lotes AS L ON L.fk_Produto_id_produto = P.id_produto;
426
427 • select * from Lotes;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_produto	nome_produto	valor	quantidade_estoque	disponibilidade	id_categoria	nome_categoria
5	Biscoito de Chocolate	4	50	1	3	Bolachas
6	Biscoito de Coco	4	60	1	3	Bolachas
12	Rosquinha	2	40	1	3	Bolachas
3	Bolo de Cenoura	8	20	1	2	Bolos
4	Bolo de Chocolate	8	15	1	2	Bolos
14	Torta de Limão	10	10	1	2	Bolos
15	Torta de Morango	10	12	1	2	Bolos
7	Doce de Leite	5	30	1	4	Doces
8	Brigadeiro	3	40	1	4	Doces

Result 31 | Read Only

<u>Consulta 2</u>:

a) Descreva por qual razão a consulta é importante para o usuário.

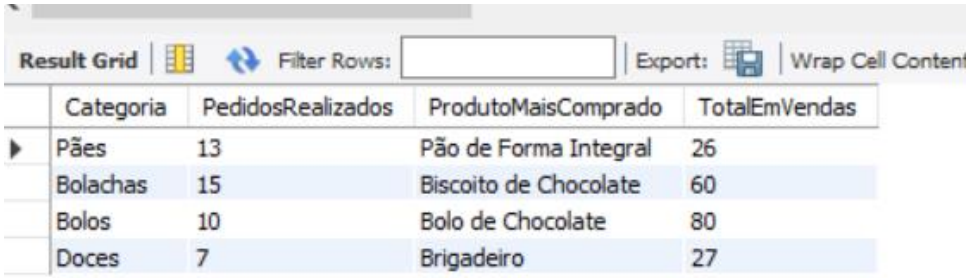
Esta consulta recupera informações sobre categorias de produtos, a quantidade total de pedidos realizados por categoria, o produto mais comprado em cada categoria e o total em vendas para cada categoria. A consulta usa várias subconsultas e chaves estrangeiras para obter essas informações.

b) Código SQL.

```
SELECT
  C.nome AS Categoria,
  SUM(I.quantidade) AS PedidosRealizados,
  (SELECT P.nome
   FROM Produtos AS P
   JOIN ItensPedido AS IP ON P.id_produto = IP.fk_Produtos_id_produto
   WHERE P.fk_Categoria_id_categoria = C.id_categoria
```

```
    GROUP BY P.id_produto
    ORDER BY SUM(IP.quantidade) DESC
    LIMIT 1) AS ProdutoMaisComprado,
    (SELECT SUM(P.valor * IP.quantidade)
     FROM Produtos AS P
     JOIN ItensPedido AS IP ON P.id_produto = IP.fk_Produtos_id_produto
     WHERE P.fk_Categoria_id_categoria = C.id_categoria
     ) AS TotalEmVendas
FROM
   Produtos AS P
JOIN
   ItensPedido AS I ON P.id_produto = I.fk_Produtos_id_produto
JOIN
   Categoria AS C ON P.fk_Categoria_id_categoria = C.id_categoria
GROUP BY P.fk_Categoria_id_categoria;
```

c) Resultado da consulta (totalEmVendas = quantidade total do produto vendido.)



Categoria	PedidosRealizados	ProdutoMaisComprado	TotalEmVendas
Pães	13	Pão de Forma Integral	26
Bolachas	15	Biscoito de Chocolate	60
Bolos	10	Bolo de Chocolate	80
Doces	7	Brigadeiro	27

<u>Consulta 3</u>:

a) Descreva por qual razão a consulta é importante para o usuário.

A consulta fornece insights cruciais que beneficiam tanto a gestão interna quanto o atendimento ao cliente, contribuindo para a eficiência operacional e tomada de decisões informadas na empresa.

b) Código SQL.

```
SELECT
    c.nome AS nome_cliente,
    c.cpf,
    p.id_pedido,
    ip.quantidade AS quantidade_produto,
    pr.nome AS nome_produto,
    CONCAT('R$ ', FORMAT(pr.valor, 2)) AS valor_unitario
FROM
    Clientes c
JOIN
    Pedidos p ON c.id_cliente = p.fk_Clientes_id_cliente
JOIN
    ItensPedido ip ON p.id_pedido = ip.fk_Pedidos_id_pedido
JOIN
    Produtos pr ON ip.fk_Produtos_id_produto = pr.id_produto;
```

c) Resultado da consulta.

nome_cliente	cpf	id_pedido	quantidade_produto	nome_produto	valor_unitario
Enrico Ricardo Bento da Conceição	44733799225	1	2	Pão Francês	R$ 2.00
Enrico Ricardo Bento da Conceição	44733799225	1	3	Biscoito de Chocolate	R$ 4.00
Lorenzo Mário Novaes	63330296291	2	1	Pão de Forma Integral	R$ 2.00
Lorenzo Mário Novaes	63330296291	2	4	Bolo de Chocolate	R$ 8.00
Julio Cláudio Oliver Souza	21587939290	3	2	Bolo de Cenoura	R$ 8.00
Julio Cláudio Oliver Souza	21587939290	3	2	Doce de Leite	R$ 5.00
Lorenzo Sérgio da Luz	09927098227	4	2	Pão Francês	R$ 2.00
Lorenzo Sérgio da Luz	09927098227	4	1	Biscoito de Coco	R$ 4.00
Francisco Diego Enzo Baptista	70623119285	5	3	Pão de Forma Integral	R$ 2.00
Francisco Diego Enzo Baptista	70623119285	5	2	Brigadeiro	R$ 3.00
Erick Fernando Pinto	09057788209	6	1	Bolo de Cenoura	R$ 8.00
Erick Fernando Pinto	09057788209	6	4	Biscoito de Chocolate	R$ 4.00
Thales Márcio Murilo Araújo	90897248260	7	2	Bolo de Chocolate	R$ 8.00

Result 9

6.3 1 Stored procedure

SP 1:

a) Descreva o que a *Stored Produceure* (SP) faz (valores de INPUT e de OUTPUT).

Usamos uma consulta SQL para encontrar o nome do cliente com base no ID do pedido passado como parâmetro de entrada.
O resultado é armazenado no parâmetro de saída **nome_cliente**

b) Descreva por qual razão a SP é importante para o usuário.
Ela é importante para facilitar a forma como o usuario ira relacionar o pedido com o cliente

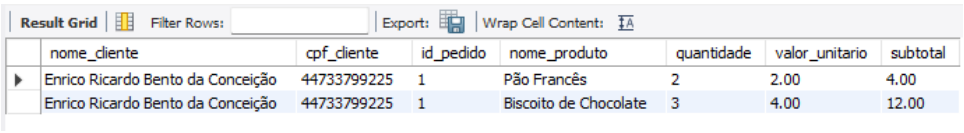
c) Código SQL

```
-- Consulta 02 --

DELIMITER //


CREATE PROCEDURE ObterDetalhesPedido (IN pedido_id INT)

BEGIN

  -- Declaração de variáveis

  DECLARE total_pedido DECIMAL(10, 2);


  -- Tabela temporária para armazenar detalhes do pedido

  CREATE TEMPORARY TABLE TempDetalhesPedido (

    nome_cliente VARCHAR(100),

    cpf_cliente VARCHAR(11),

    id_pedido INT,

    nome_produto VARCHAR(100),

    quantidade INT,

    valor_unitario DECIMAL(10, 2),

    subtotal DECIMAL(10, 2)

  );


  -- Popula a tabela temporária com os detalhes do pedido

  INSERT INTO TempDetalhesPedido
```

```
SELECT
    c.nome AS nome_cliente,
    c.cpf AS cpf_cliente,
    p.id_pedido,
    pr.nome AS nome_produto,
    ip.quantidade,
    pr.valor AS valor_unitario,
    ip.quantidade * pr.valor AS subtotal
FROM
    Clientes c
JOIN
    Pedidos p ON c.id_cliente = p.fk_Clientes_id_cliente
JOIN
    ItensPedido ip ON p.id_pedido = ip.fk_Pedidos_id_pedido
JOIN
    Produtos pr ON ip.fk_Produtos_id_produto = pr.id_produto
WHERE
    p.id_pedido = pedido_id;

-- Calcula o total do pedido
SELECT SUM(subtotal) INTO total_pedido FROM TempDetalhesPedido;

-- Seleciona todos os detalhes do pedido
SELECT * FROM TempDetalhesPedido;

-- Retorna o total do pedido
SELECT total_pedido AS total_pedido;

-- Drop da tabela temporária
DROP TEMPORARY TABLE IF EXISTS TempDetalhesPedido;
```

```
END //


DELIMITER ;


SET @pedido_id = 1; -- Use a variable that matches the parameter name

CALL ObterDetalhesPedido(@pedido_id);

SELECT @pedido_id;
```

d) Resultado da execução da SP



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

nome_cliente	cpf_cliente	id_pedido	nome_produto	quantidade	valor_unitario	subtotal
Enrico Ricardo Bento da Conceição	44733799225	1	Pão Francês	2	2.00	4.00
Enrico Ricardo Bento da Conceição	44733799225	1	Biscoito de Chocolate	3	4.00	12.00

6.4 1 Trigger

Trigger 1:

e) Descreva o que o *Trigger* faz para as operações de INSERT / UPDATE / DELETE.

Triggers em bancos de dados são acionados automaticamente em resposta a operações como INSERT, UPDATE e DELETE em tabelas. Eles permitem automatizar ações, validar dados e manter a integridade do banco de dados em momentos específicos. Os triggers de INSERT são acionados após a inserção de dados, os de UPDATE após a atualização e os de DELETE após a exclusão. Eles são usados para melhorar a automação e a confiabilidade das operações de banco de dados.

f) Descreva por qual razão o *Trigger* é importante para o usuário.
Neste trigger, chamado **AtualizarEstoquePedido**, ele é executado após a inserção de um novo registro na tabela **ItensPedido**. Ele primeiro obtém o ID do produto e a quantidade do pedido inserido e, em seguida, atualiza a quantidade de estoque do produto subtraindo a quantidade do pedido. Dessa forma, o estoque do produto é automaticamente atualizado sempre que um pedido é inserido.

g) Código SQL

```
DELIMITER //


CREATE TRIGGER AtualizarEstoquePedido
```

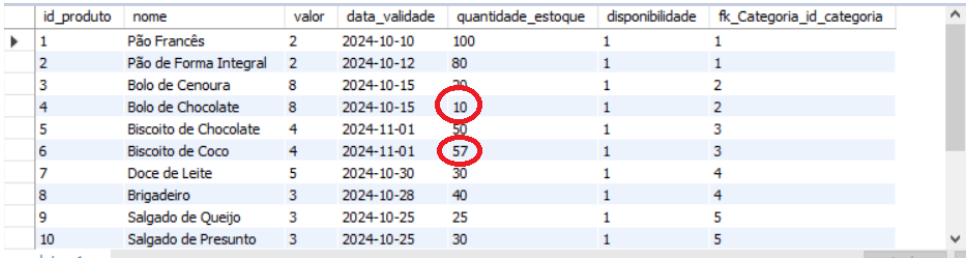
```
AFTER INSERT ON ItensPedido

FOR EACH ROW

BEGIN

   DECLARE produto_id INT;

   DECLARE quantidade_pedido INT;


   -- Obter o ID do produto e a quantidade do pedido inserido

   SELECT fk_Produtos_id_produto, quantidade INTO produto_id, quantidade_pedido

   FROM ItensPedido

   WHERE Id_itensPedido = NEW.Id_itensPedido;



   UPDATE Produtos

   SET quantidade_estoque = quantidade_estoque - quantidade_pedido

   WHERE id_produto = produto_id;

END;

//


DELIMITER ;
```

h) Resultado da execução do *Trigger*

Antes da execuçao da trigger:

id_produto	nome	valor	data_validade	quantidade_estoque	disponibilidade	fk_Categoria_id_categoria
1	Pão Francês	2	2024-10-10	100	1	1
2	Pão de Forma Integral	2	2024-10-12	80	1	1
3	Bolo de Cenoura	8	2024-10-15	20	1	2
4	Bolo de Chocolate	8	2024-10-15	15	1	2
5	Biscoito de Chocolate	4	2024-11-01	70	1	3
6	Biscoito de Coco	4	2024-11-01	60	1	3
7	Doce de Leite	5	2024-10-30	30	1	4
8	Brigadeiro	3	2024-10-28	40	1	4
9	Salgado de Queijo	3	2024-10-25	25	1	5
10	Salgado de Presunto	3	2024-10-25	30	1	5

Depois da execuçao da trigger:



id_produto	nome	valor	data_validade	quantidade_estoque	disponibilidade	fk_Categoria_id_categoria
1	Pão Francês	2	2024-10-10	100	1	1
2	Pão de Forma Integral	2	2024-10-12	80	1	1
3	Bolo de Cenoura	8	2024-10-15	20	1	2
4	Bolo de Chocolate	8	2024-10-15	10	1	2
5	Biscoito de Chocolate	4	2024-11-01	50	1	3
6	Biscoito de Coco	4	2024-11-01	57	1	3
7	Doce de Leite	5	2024-10-30	30	1	4
8	Brigadeiro	3	2024-10-28	40	1	4
9	Salgado de Queijo	3	2024-10-25	25	1	5
10	Salgado de Presunto	3	2024-10-25	30	1	5

Insert utilizado para ativaçao da trigger:

INSERT INTO Pedidos (id_pedido, data_realizacao, observacao, fk_Clientes_id_cliente, fk_Funcionarios_id_funcionario)

VALUES (14, '2023-06-26', 'Entregar na Rua Bananas', 2, 10);

-- Itens do Pedido 7

INSERT INTO ItensPedido (Id_itensPedido, fk_Produtos_id_produto, fk_Pedidos_id_pedido, quantidade)

VALUES (26, 4, 14, 5), (25, 6, 14, 3);