Uern – Universidade Estadual do RN

Curso de Sistemas para internet

Componente Curricular: Banco de Dados

Nome do Discente: Francisco das Chagas Fernandes de Queiroz Filho

Relatório do Banco de Dados - Pizzaria

Sumário

1. Introdução

Este relatório apresenta o desenvolvimento de um banco de dados para uma pizzaria, com foco na organização das informações de pedidos, clientes, pizzas e ingredientes. O objetivo é proporcionar um sistema eficiente para gerenciar pedidos e manter a integridade dos dados de forma automatizada.

Para isso, foram criadas tabelas relacionais seguindo boas práticas de modelagem de banco de dados, além de procedimentos armazenados e gatilhos para automatizar algumas operações. A seguir, serão detalhadas todas as etapas da criação desse banco.

2. Descrição do Sistema

O sistema da pizzaria gerencia pedidos realizados pelos clientes, armazenando informações sobre pizzas e ingredientes utilizados. As principais entidades envolvidas são:

- Clientes: Contêm informações sobre os clientes cadastrados.
- Pizzas: Define os tipos de pizzas disponíveis e seus preços.
- Ingredientes: Lista os ingredientes cadastrados.
- Pedidos: Registra os pedidos realizados pelos clientes, com status de andamento.
- Itens do Pedido: Define quais pizzas estão inclusas em cada pedido.
- Pizzas e Ingredientes: Relaciona quais ingredientes fazem parte de cada pizza.

3. Estrutura do Banco de Dados

Para representar o funcionamento da pizzaria, foram criadas seis tabelas principais:

- **CLIENTES**: Armazena informações como nome, telefone e endereço.
- **PIZZAS**: Contém os dados das pizzas oferecidas, incluindo nome e preço.
- **INGREDIENTES**: Lista os ingredientes disponíveis.
- **PIZZAS_INGREDIENTES**: Relaciona cada pizza com seus respectivos ingredientes.
- **PEDIDOS**: Registra os pedidos realizados pelos clientes.
- **ITENS_PEDIDOS**: Guarda os detalhes dos pedidos, incluindo as pizzas e quantidades solicitadas.

A estrutura foi planejada para manter a integridade dos dados e evitar duplicações ou inconsistências.

4. Procedimentos Armazenados

Foram criados dois procedimentos armazenados para automatizar consultas:

- SP_PERIODO_PEDIDO(ID_CLIENTE): Retorna os pedidos pendentes de um cliente.

- SP_LISTAR_PEDIDOS_CLIENTE(ID_CLIENTE): Lista todos os pedidos de um cliente, incluindo detalhes das pizzas e quantidades.

Exemplo de Execução:

```
CALL SP_PERIODO_PEDIDO(1);
CALL SP_LISTAR_PEDIDOS_CLIENTE(1);
```

5. Gatilhos

Além dos procedimentos armazenados, foram criados gatilhos para automatizar algumas operações e evitar problemas no gerenciamento dos pedidos.

- **GT_PENDENTE**: Define automaticamente o status do pedido como "pendente" e registra a data do pedido.
- **GT_NAO_EXCLUSAO**: Impede que clientes com pedidos pendentes sejam excluídos do banco de dados.

Esses gatilhos garantem que o sistema funcione de maneira mais segura e confiável, reduzindo a necessidade de intervenções manuais.

Exemplo de Gatilho:

```
DELIMITER $$
CREATE TRIGGER GT_PENDENTE
AFTER INSERT ON PEDIDOS
FOR EACH ROW
BEGIN
  UPDATE PEDIDOS
  SET STATUS = 'pendente', DATA_PEDIDO = NOW()
  WHERE ID = NEW.ID;
END $$
DELIMITER ;
```

6. Scripts e Prints resultants

1. Scripts

```
/*CRIANDO O BD*/
create database          PIZZARIA;

/*SELECIONA O BD*/
use PIZZARIA;

/*CRIA A TABELA PIZZAS*/
create table PIZZAS(
        ID int primary key auto_increment,
   NOME varchar(100) not null unique,
   PRECO decimal(6,2) not null
);

/*CRIA A TABELA INGREDIENTES*/
create table INGREDIENTES(
        ID int primary key auto_increment,
   NOME varchar(100) not null
);

/*CRIA A TABELA QUE RELACIONA PIZZAS E INGREDIENTES*/
create table PIZZAS_INGREDIENTES(
        ID_PIZZA int not null,
   ID_INGREDIENTE int not null,
   primary key(ID_PIZZA, ID_INGREDIENTE),
   foreign key  (ID_PIZZA) references PIZZAS(ID) on delete cascade,
```

```
   foreign key (ID_INGREDIENTE) references INGREDIENTES(ID) on delete cascade
);

/*CRIAR UMA TABELA DE CLIENTES*/
create table CLIENTES(
ID int primary key auto_increment,
NOME varchar (100) not null,
TELEFONE varchar (20) not null unique,
ENDERECO text not null
);

/* CRIAR A TABELA DE PEDIDOS*/
create table PEDIDOS(
        ID int primary key auto_increment,
   ID_CLIENTE int not null,
   DATA_PEDIDO timestamp,
   status enum ('pendente', 'preparando', 'entregue', 'cancelado') default 'pendente',
   foreign key (ID_CLIENTE) references CLIENTES(ID) on delete cascade
);

/*CRIAR A TABELA DE ITENS DO PEDIDO*/
create table ITENS_PEDIDOS(
        ID int primary key auto_increment,
   ID_PEDIDO int not null,
   ID_PIZZA int not null,
```

```
    QTD int default 1,
    foreign key(ID_PEDIDO) references PEDIDOS(ID) on delete cascade,
    foreign key(ID_PIZZA) references PIZZAS(ID) on delete cascade
);



desc PIZZAS;
desc INGREDIENTES;
desc PIZZAS_INGREDIENTES;
desc CLIENTES;
desc PEDIDOS;
desc ITENS_PEDIDOS;



/*INSERINDO INGREDIENTES*/
insert into INGREDIENTES (NOME)
values
            ('queijo'),
    ('tomate'),
    ('manjericão'),
    ('calabresa');



/*INSERIR PIZZAS*/
INSERT INTO PIZZAS (NOME, PRECO)
```

```
SELECT 'Marguerita', 40.00

WHERE NOT EXISTS (SELECT 1 FROM PIZZAS WHERE NOME = 'Marguerita');


INSERT INTO PIZZAS (NOME, PRECO)

SELECT 'Calabresa', 45.00

WHERE NOT EXISTS (SELECT 1 FROM PIZZAS WHERE NOME = 'Calabresa');



/*RELACINAR PIZZAS AO INGREDIENTE*/

insert into PIZZAS_INGREDIENTES(ID_PIZZA,ID_INGREDIENTE) values
(1,1),(1,2),(1,3), -- PIZZA MARGERITA

(2,1),(2,4);-- PIZZA CALABRESA


/*INSERIR UM CLIENTE*/

insert into CLIENTES (NOME, TELEFONE, ENDERECO) values ('Joao',
'84997058669', 'Rua das laranjas');

insert into CLIENTES (NOME, TELEFONE, ENDERECO) values
('Maria','85998674785', 'Rua das bananeiras');


/*INSERINDO PEDIDO*/

insert into PEDIDOS(ID_CLIENTE) values(1),(11);


/*INSERINDO PEDIDO COM PIZZAS*/

insert into ITENS_PEDIDOS(ID_PEDIDO,ID_PIZZA, QTD) values (1,1,2),(2,2,1);


-- VER TODOS OS DADOS INSERIDOS
```

```
SELECT * FROM PIZZAS;

SELECT * FROM INGREDIENTES;

SELECT * FROM PIZZAS_INGREDIENTES;

SELECT * FROM CLIENTES;

SELECT * FROM PEDIDOS;

SELECT * FROM ITENS_PEDIDOS;


/*AS FUNÇÕES DE AGREGAÇÃO*/

select count(*) from PIZZAS; -- numero total registros

select sum(ID_PEDIDO) from ITENS_PEDIDOS; --  somaitens coluna id_pedido

select avg(PRECO) from PIZZAS; -- CALCULANDO A MEDIA DE PRECO DAS PIZZAS

select min(PRECO) AS PRECOMINIMO, max(PRECO) AS PRECOMAXIMO FROM
PIZZAS; -- RETORNA A PIZZA MAIS BARATA E MAIS CARA


/*CONSULTAS COM JOIN*/

/*INNER JOIN*/

SELECT -- LISTA DADOS DOS CLIENTES DOS PEDIDOS FEITOS

   PEDIDOS.ID AS Pedido,

   CLIENTES.NOME AS Cliente,

   CLIENTES.TELEFONE,

   PEDIDOS.DATA_PEDIDO,

   PEDIDOS.STATUS

FROM PEDIDOS

INNER JOIN CLIENTES ON PEDIDOS.ID_CLIENTE = CLIENTES.ID;
```

```
/*INNER JOIN*/

SELECT -- LISTA AS PIZZAS E SEUS INGREDIENTES

   PIZZAS.ID AS ID_PIZZA,

   PIZZAS.NOME AS PIZZA_NOME,

   INGREDIENTES.ID AS ID_INGREDIENTE,

   INGREDIENTES.NOME AS INGREDIENTE_NOME

FROM PIZZAS

INNER JOIN PIZZAS_INGREDIENTES ON PIZZAS.ID =
PIZZAS_INGREDIENTES.ID_PIZZA

INNER JOIN INGREDIENTES ON PIZZAS_INGREDIENTES.ID_INGREDIENTE =
INGREDIENTES.ID;



/*LEFT JOIN*/

SELECT /*Listar todos os clientes e seus pedidos (inclusive clientes sem pedidos)*/

        PEDIDOS.ID AS PEDIDO,

   coalesce(CLIENTES.ID, 'SEM CLIENTE') AS CLIENTE,

   coalesce(CLIENTES.TELEFONE, 'TELEFONE NÃO INFORMADO') AS TELEFONE,

   PEDIDOS.DATA_PEDIDO,

   PEDIDOS.STATUS

FROM PEDIDOS

LEFT JOIN CLIENTES ON CLIENTES.ID = PEDIDOS.ID_CLIENTE;


/*RIGHT JOIN*/

SELECT /*Listar todos os clientes e seus pedidos (inclusive clientes sem pedidos)*/
```

```
        PEDIDOS.ID AS PEDIDO,
    coalesce(CLIENTES.ID, 'SEM CLIENTE') AS CLIENTE,
    coalesce(CLIENTES.TELEFONE, 'TELEFONE NÃO INFORMADO') AS TELEFONE,
    PEDIDOS.DATA_PEDIDO,
    PEDIDOS.STATUS
FROM PEDIDOS
RIGHT JOIN CLIENTES ON CLIENTES.ID = PEDIDOS.ID_CLIENTE;


/*CRIAÇÃO DAS VIEWS*/
CREATE VIEW PIZZAS_E_INGREDIENTES AS
        SELECT -- LISTA AS PIZZAS E SEUS INGREDIENTES
                PIZZAS.ID AS ID_PIZZA,
                PIZZAS.NOME AS PIZZA_NOME,
                INGREDIENTES.ID AS ID_INGREDIENTE,
                INGREDIENTES.NOME AS INGREDIENTE_NOME
        FROM PIZZAS
        INNER JOIN PIZZAS_INGREDIENTES ON PIZZAS.ID =
PIZZAS_INGREDIENTES.ID_PIZZA
        INNER JOIN INGREDIENTES ON PIZZAS_INGREDIENTES.ID_INGREDIENTE =
INGREDIENTES.ID;


CREATE VIEW CLIENTES_PEDIDOS AS
        SELECT -- LISTA DADOS DOS CLIENTES DOS PEDIDOS FEITOS
    PEDIDOS.ID AS Pedido,
    CLIENTES.NOME AS Cliente,
```

```
   CLIENTES.TELEFONE,
   PEDIDOS.DATA_PEDIDO,
   PEDIDOS.STATUS
FROM PEDIDOS
INNER JOIN CLIENTES ON PEDIDOS.ID_CLIENTE = CLIENTES.ID;

CREATE VIEW MEDIA_PRECO AS
        select avg(PRECO) from PIZZAS; -- CALCULANDO A MEDIA DE PRECO DAS
PIZZAS;



/*CONSULTANDO AS VIEWS*/
SELECT * FROM PIZZAS_E_INGREDIENTES;
SELECT * FROM CLIENTES_PEDIDOS;
SELECT * FROM MEDIA_PRECO;

/*CRIANDO PROCEDIMENTOS*/
-- 1°
DELIMITER $$
CREATE PROCEDURE SP_PERIODO_PEDIDO(IN ID_CLIENTE INT)
/*retorna todos os pedidos com status pendente*/
BEGIN
   SELECT * FROM PEDIDOS
   WHERE ID_CLIENTE = ID_CLIENTE
   AND STATUS = 'pendente';
```

```
END $$

DELIMITER ;

-- 2°

DELIMITER $$

CREATE PROCEDURE SP_LISTAR_PEDIDOS_CLIENTE(IN ID_CLIENTE INT)

/*Retorna todos os pedidos de um cliente específico, informando os detalhes dos pedidos feitos por ele.*/

BEGIN

  SELECT

    PEDIDOS.ID AS ID_PEDIDO,

    PEDIDOS.DATA_PEDIDO,

    PEDIDOS.STATUS,

    ITENS_PEDIDOS.ID_PIZZA,

    PIZZAS.NOME AS PIZZA,

    ITENS_PEDIDOS.QTD AS QUANTIDADE

  FROM PEDIDOS

  INNER JOIN ITENS_PEDIDOS ON PEDIDOS.ID = ITENS_PEDIDOS.ID_PEDIDO

  INNER JOIN PIZZAS ON ITENS_PEDIDOS.ID_PIZZA = PIZZAS.ID

  WHERE PEDIDOS.ID_CLIENTE = ID_CLIENTE;

END $$

DELIMITER ;


/*EXECUTANDO O PROCEDIMENTO*/

CALL SP_PERIODO_PEDIDO(1);

CALL SP_LISTAR_PEDIDOS_CLIENTE(1);
```

```
/*CRIANDO GATILHOS*/
/*1° Define automaticamente o status "pendente" e a data do pedido.l.*/
DELIMITER $$

CREATE TRIGGER GT_PENDENTE
AFTER INSERT ON PEDIDOS
FOR EACH ROW
BEGIN
   UPDATE PEDIDOS
   SET STATUS = 'pendente', DATA_PEDIDO = NOW()
   WHERE ID = NEW.ID;
END $$

DELIMITER ;
/*2° Impede a exclusão de clientes com pedidos pendentes. */
DELIMITER $$

CREATE TRIGGER GT_NAO_EXCLUSAO
BEFORE DELETE ON CLIENTES
FOR EACH ROW
BEGIN
   DECLARE PEDIDOS_PENDENTES INT;
```

```
    -- Conta os pedidos pendentes do cliente que será excluído
    SELECT COUNT(*) INTO PEDIDOS_PENDENTES
    FROM PEDIDOS
    WHERE ID_CLIENTE = OLD.ID AND STATUS = 'pendente';

    -- Se houver pedidos pendentes, impede a exclusão
    IF PEDIDOS_PENDENTES > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Erro: Cliente possui pedidos pendentes e não pode ser excluído.';
    END IF;
END $$

DELIMITER ;

DELETE FROM CLIENTES WHERE ID = 1;
commit;
```

2. Prints resultantes

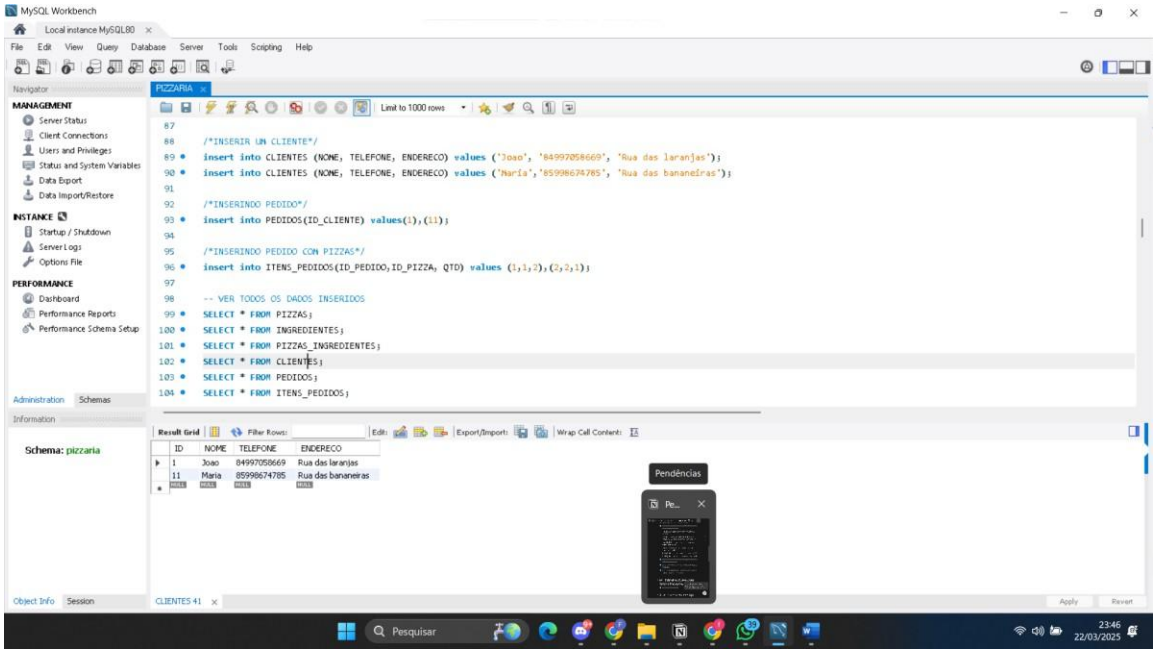

Fig 1 select clients

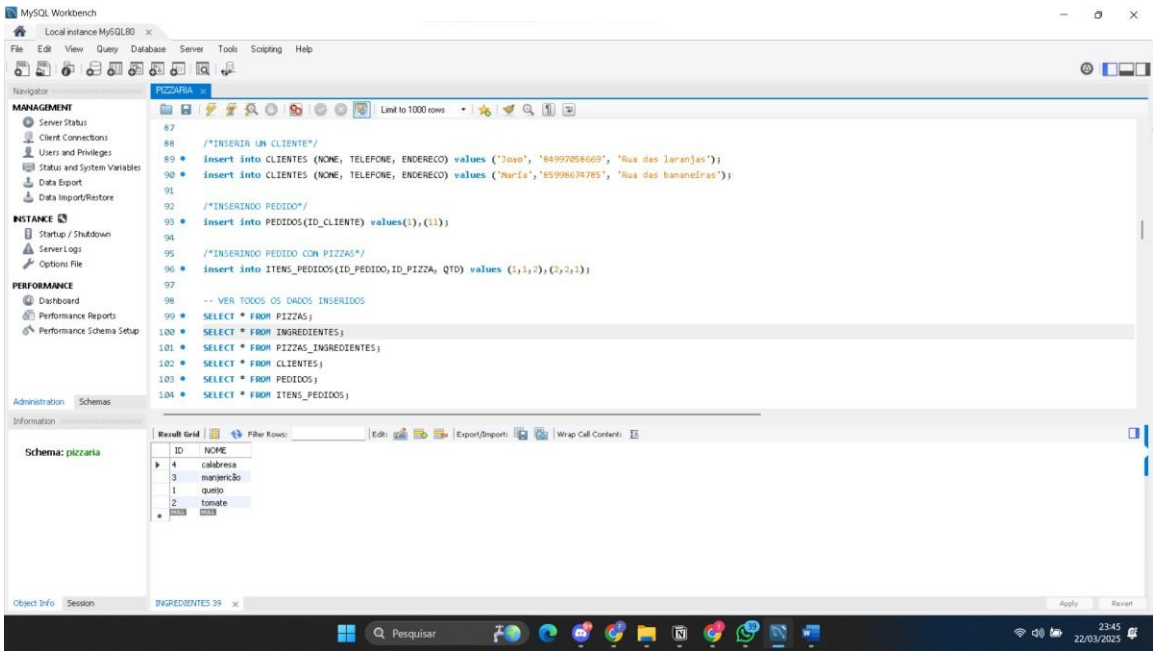

Fig 2 select ingredientes

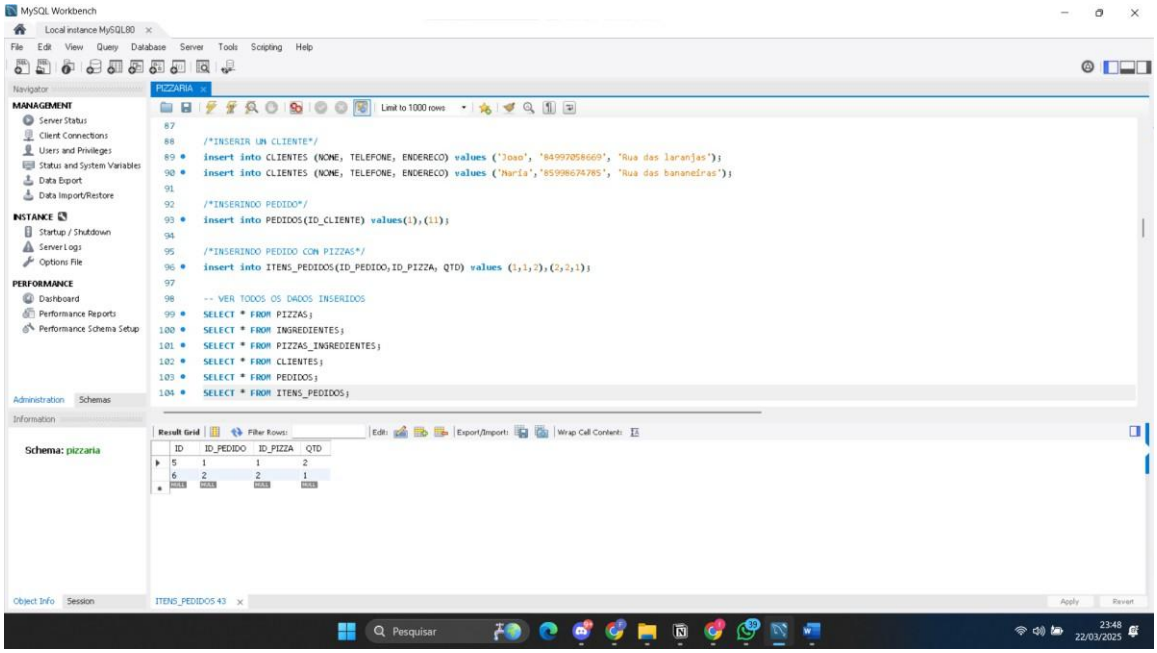

Fig 3 select items pedidos

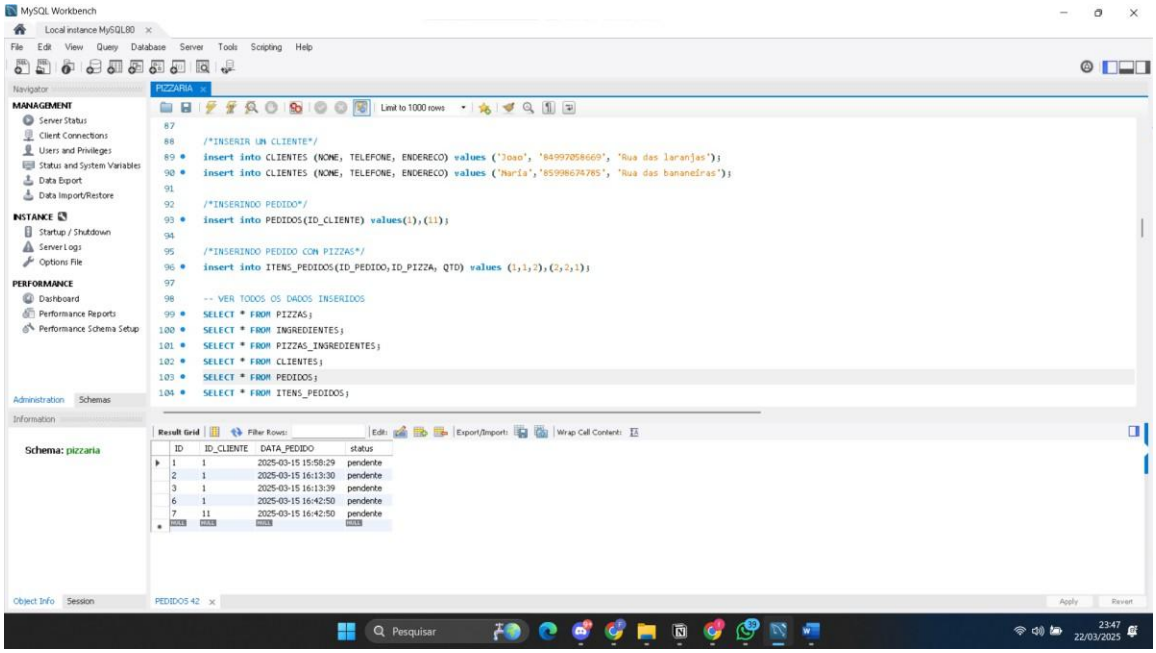

Fig 4 select pedidos

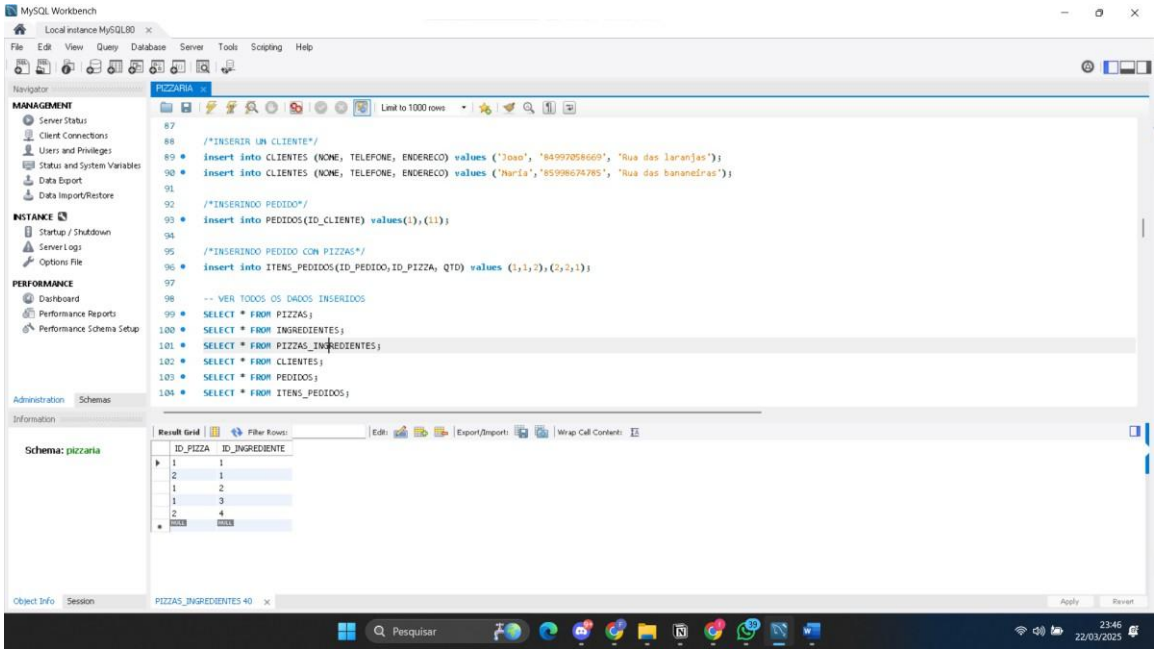

Fig 5 select pizzas ingredients

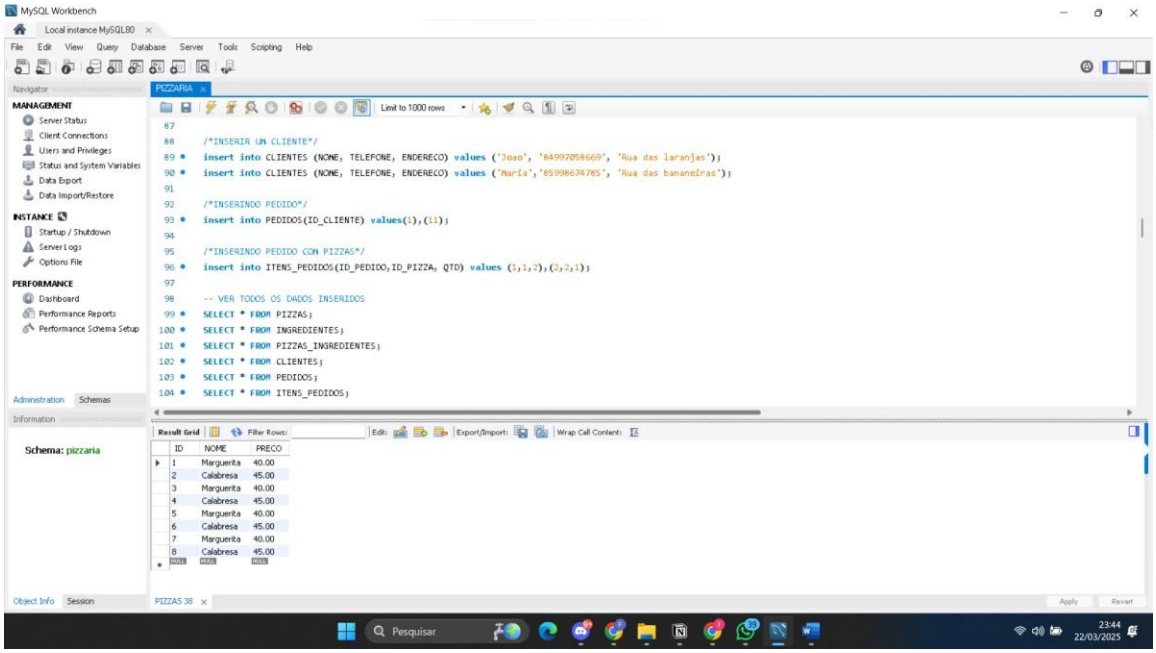

Fig 6 select pizzas

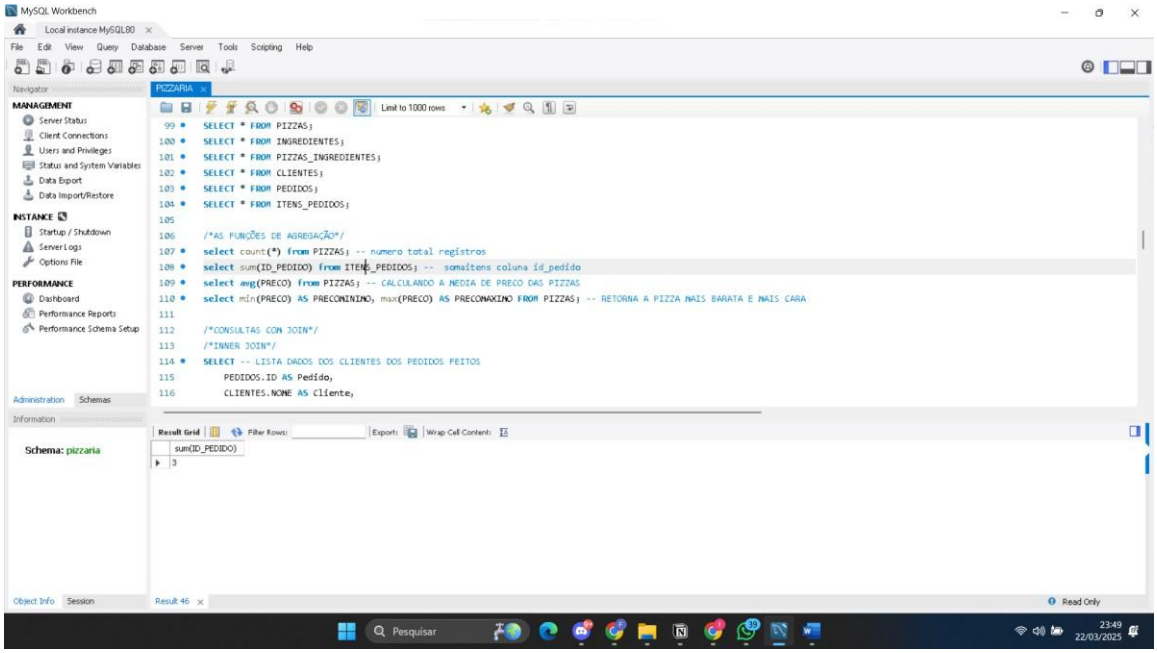

Fig 7 sun

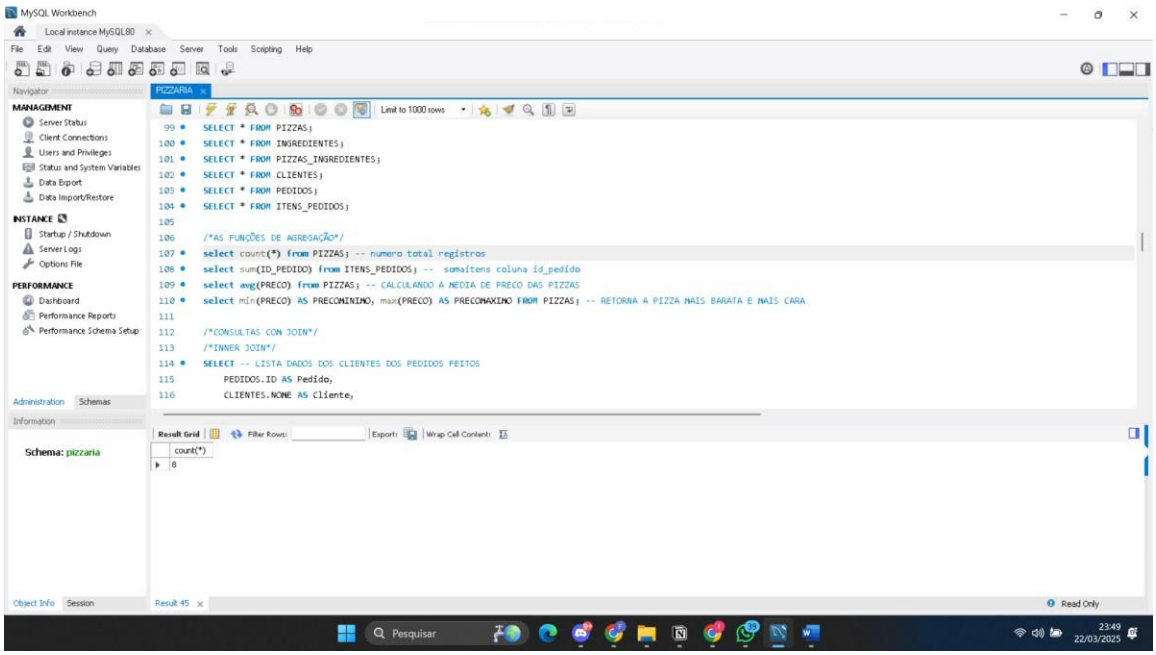

Fig 8 count

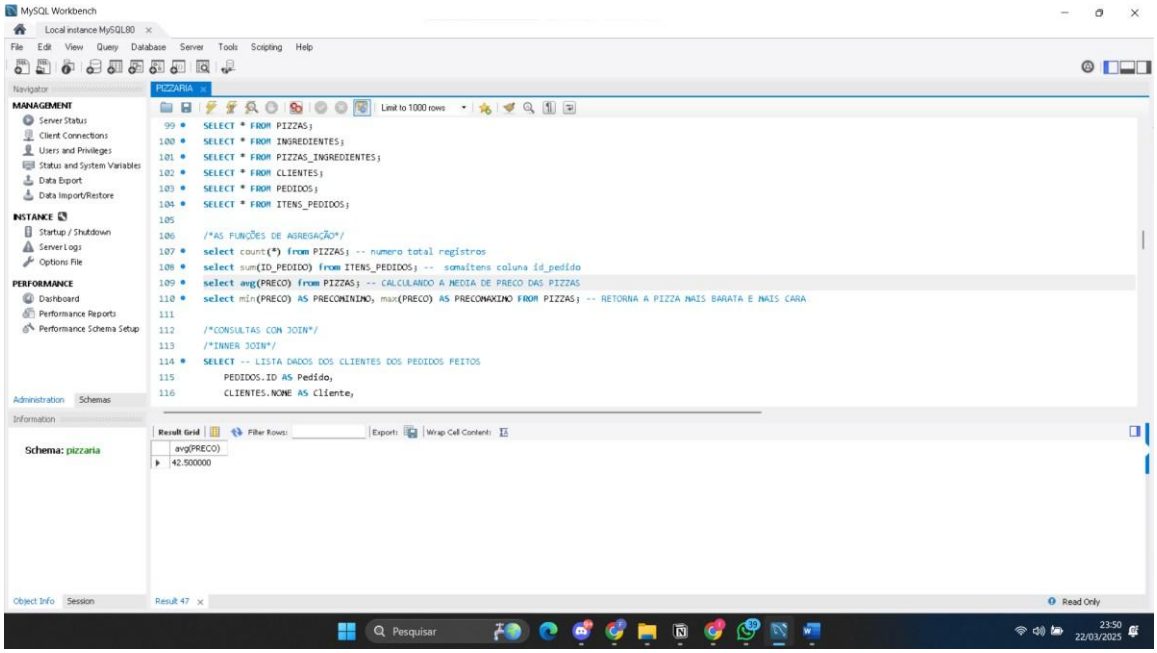

Fig 9 avg

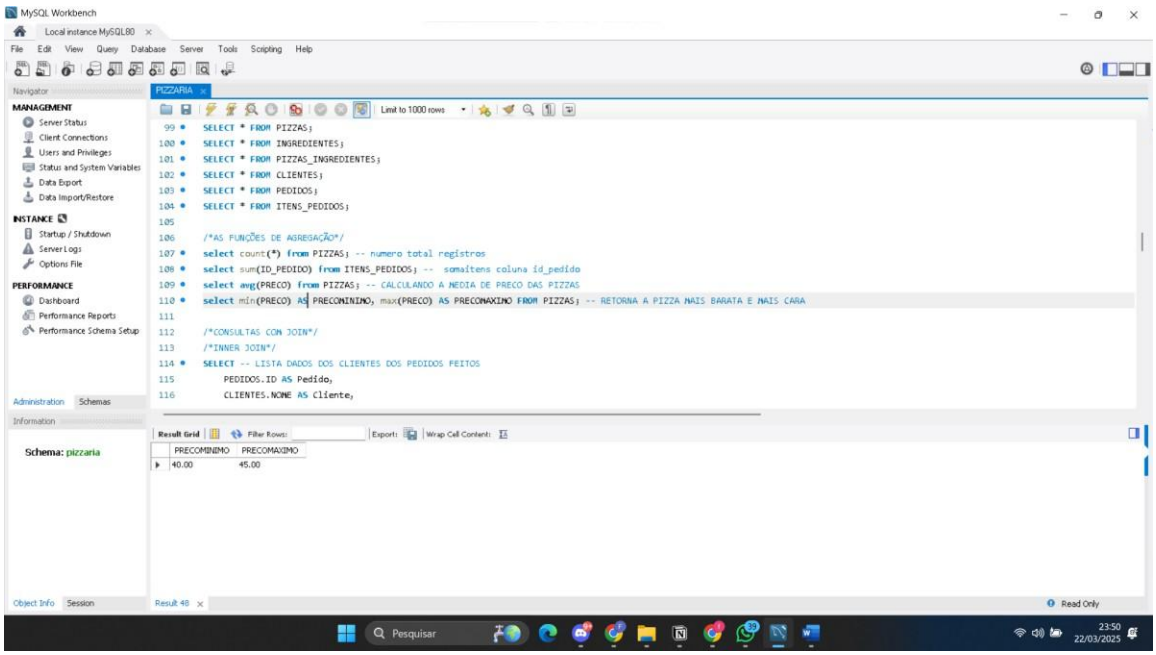

Fig 10 Min e Máx

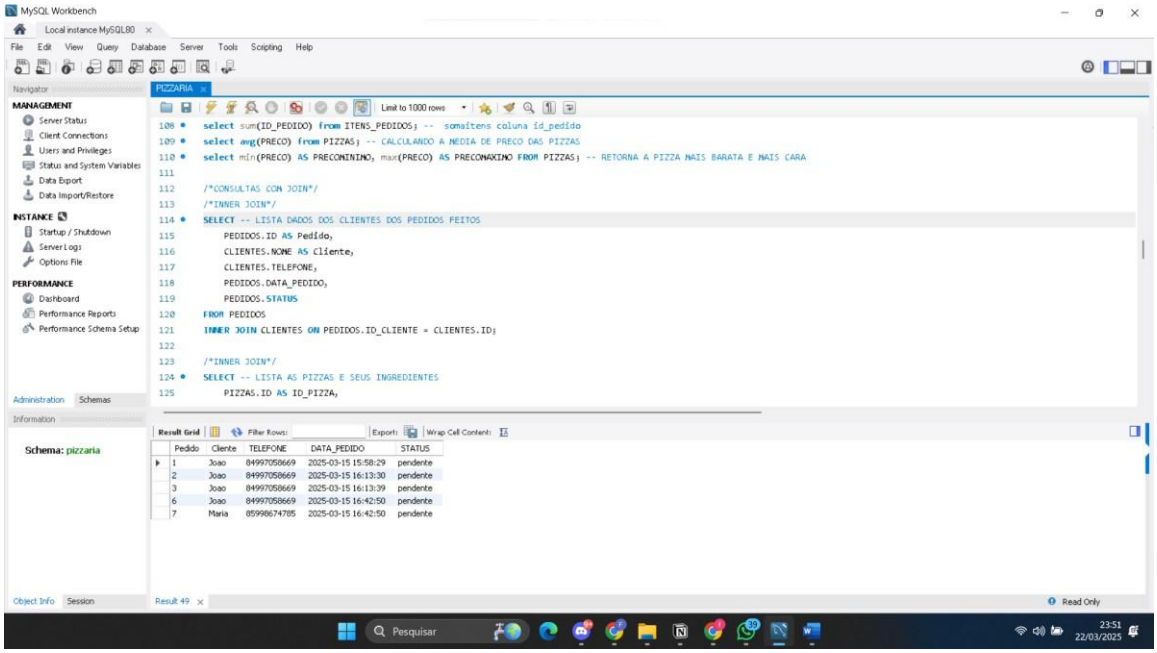

Fig 11 inner Join

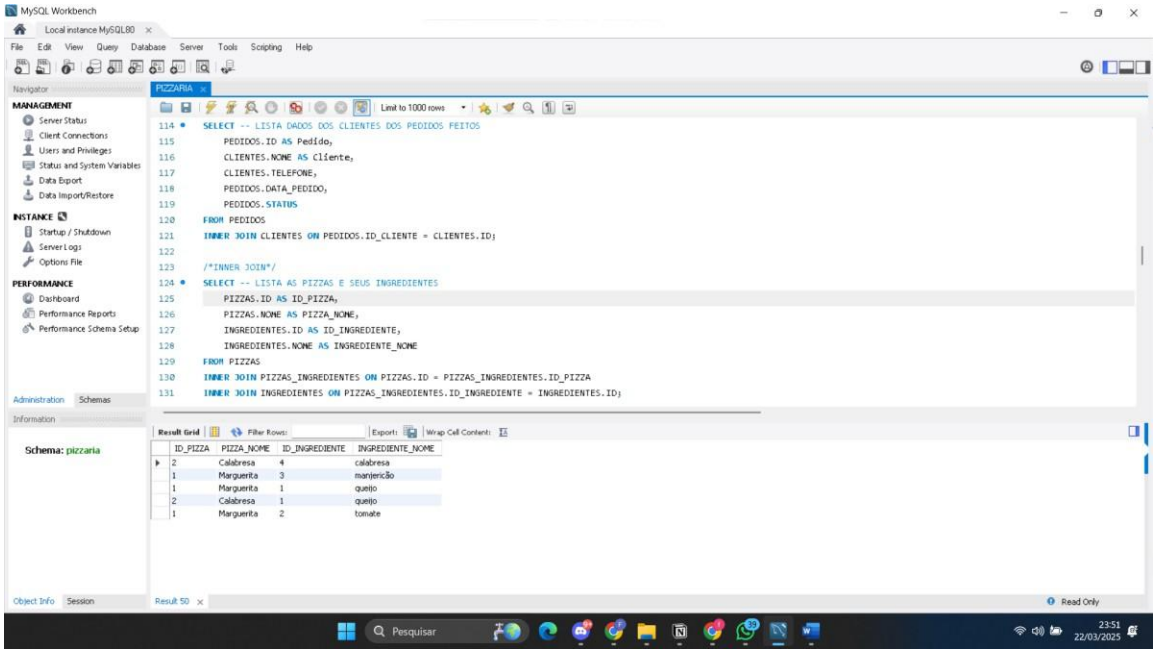

Fig 12 inner join

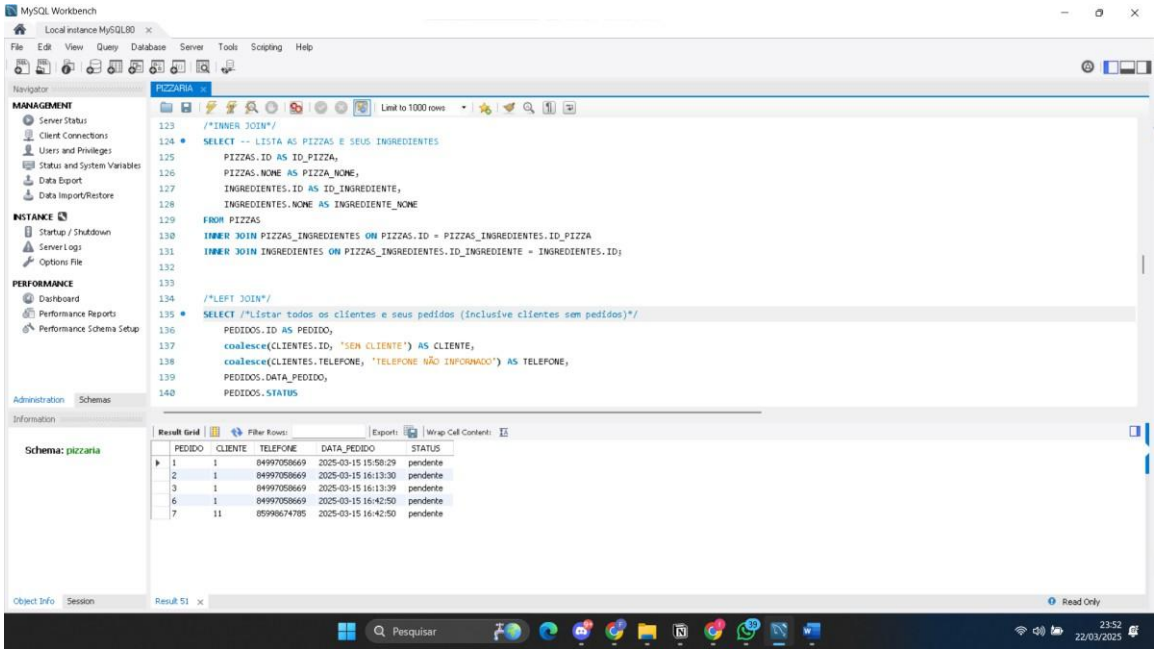

Fig 13 left join

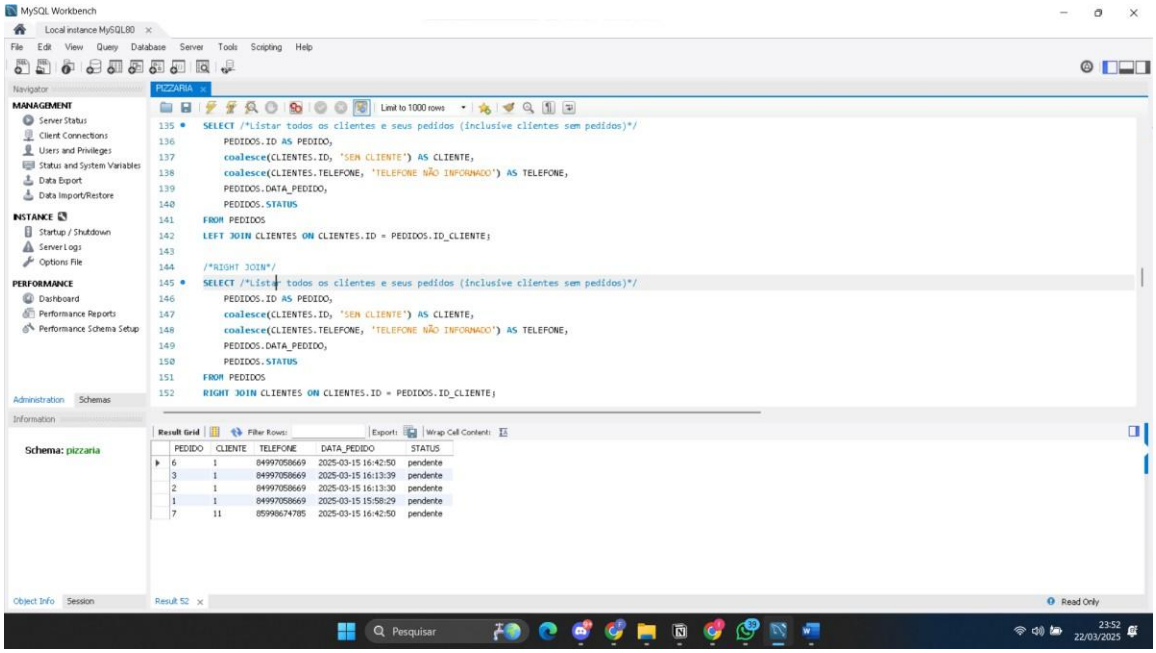

Fig 14 right join

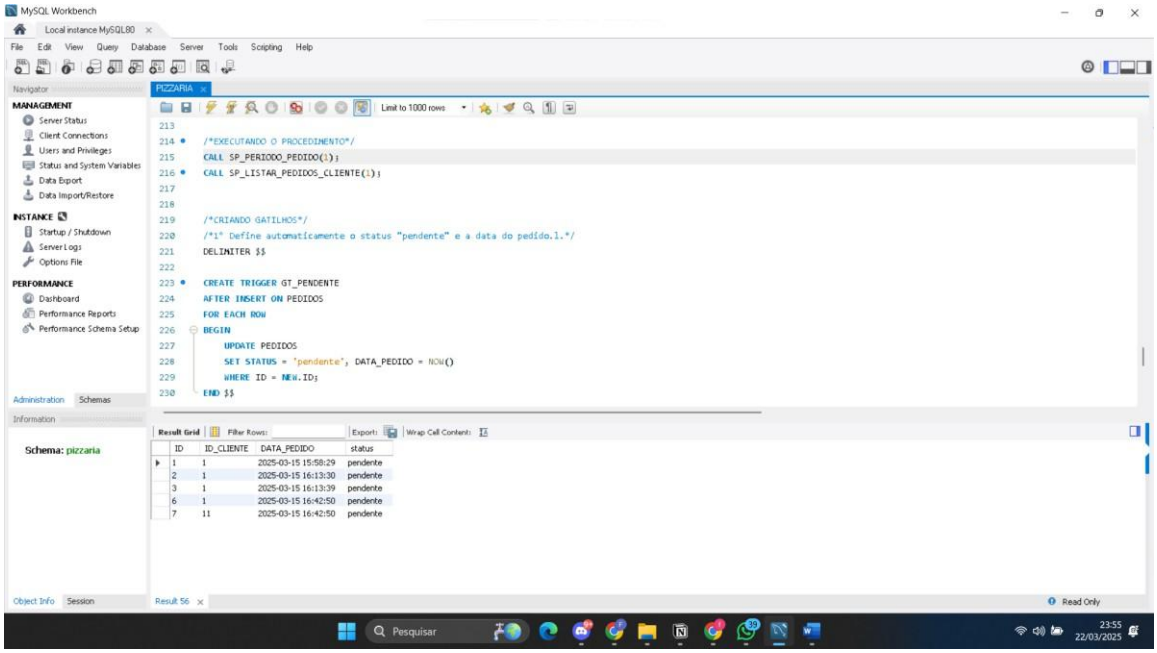

Fig 15 procedimento

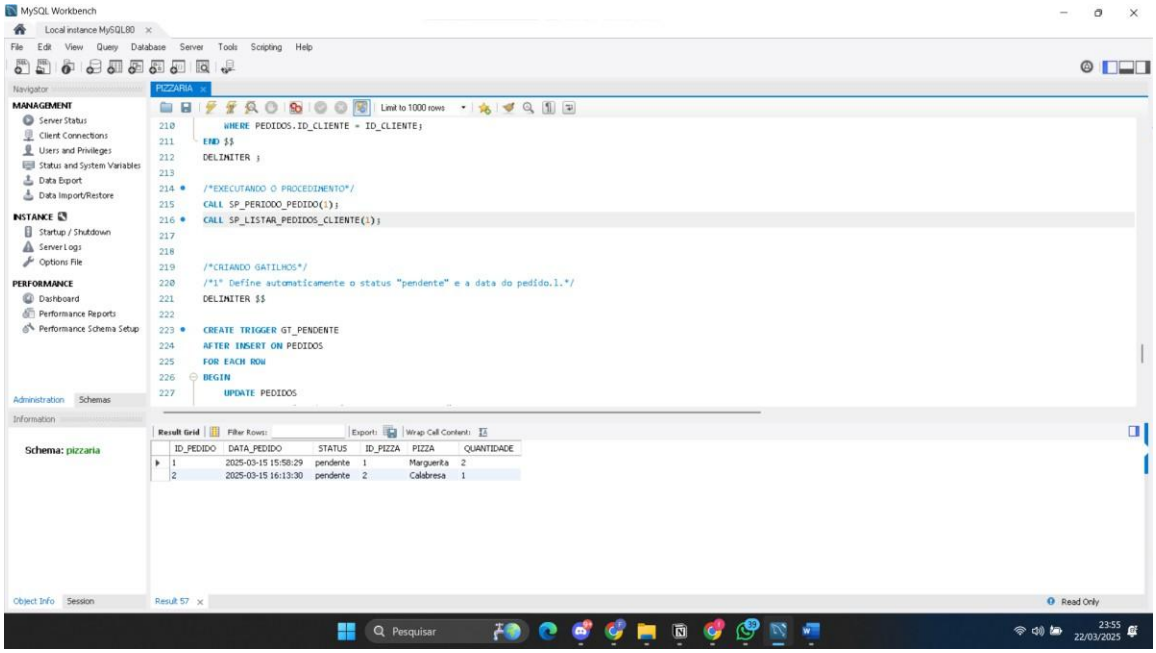

Fig 16 procedimento

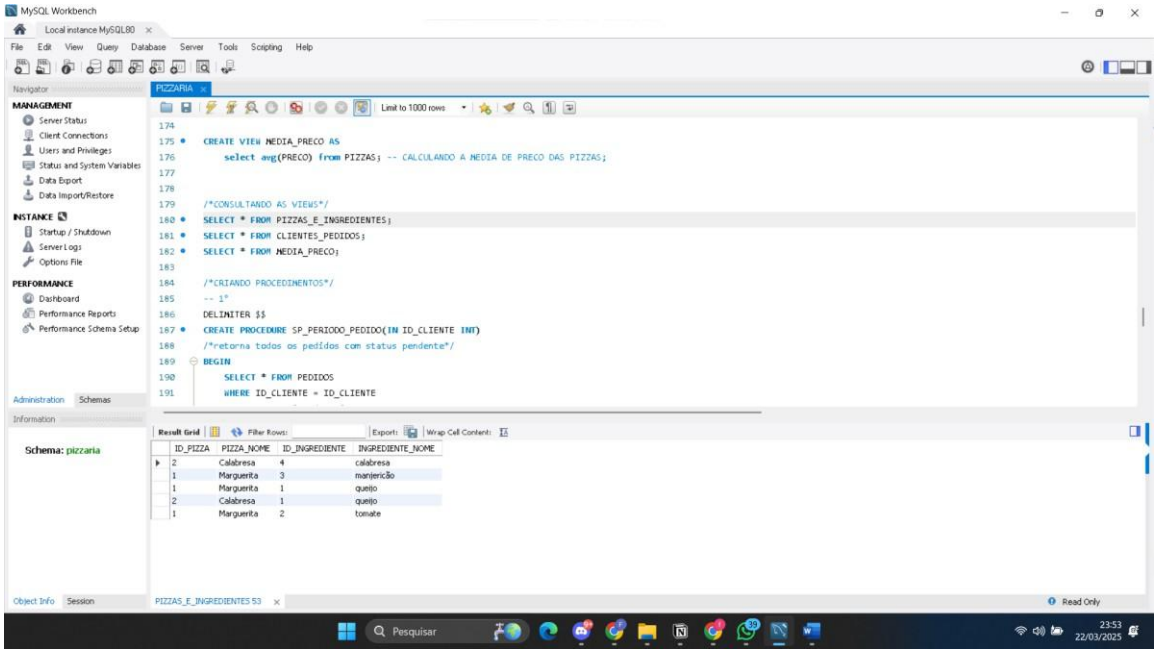

Fig 17 views

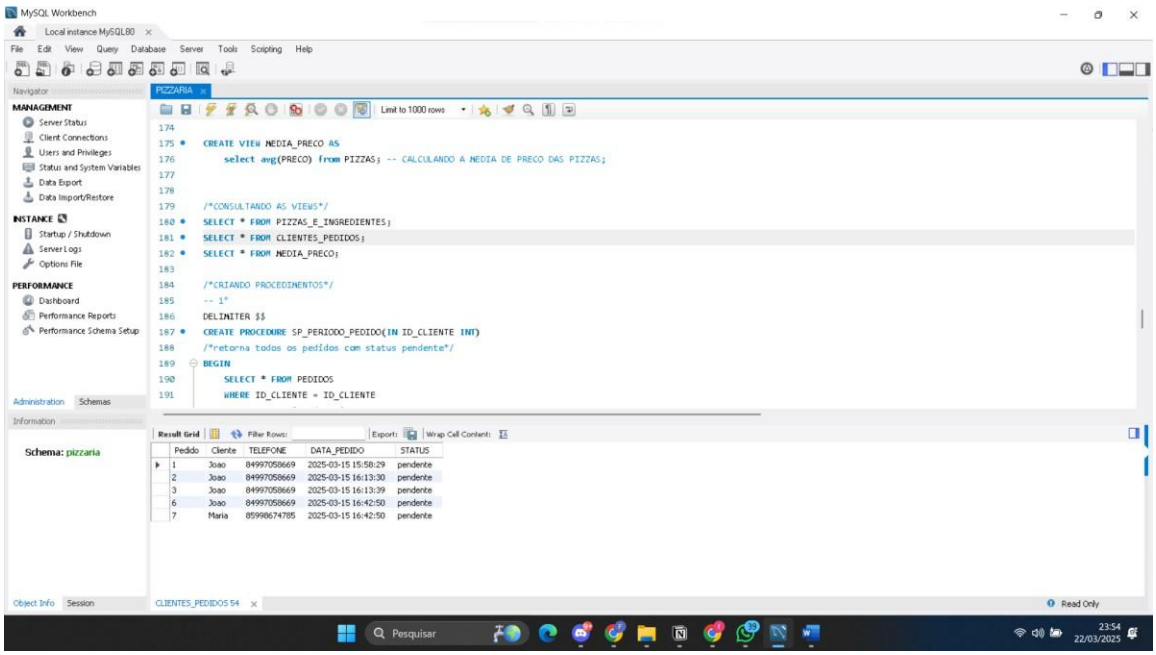

Fig 18 views

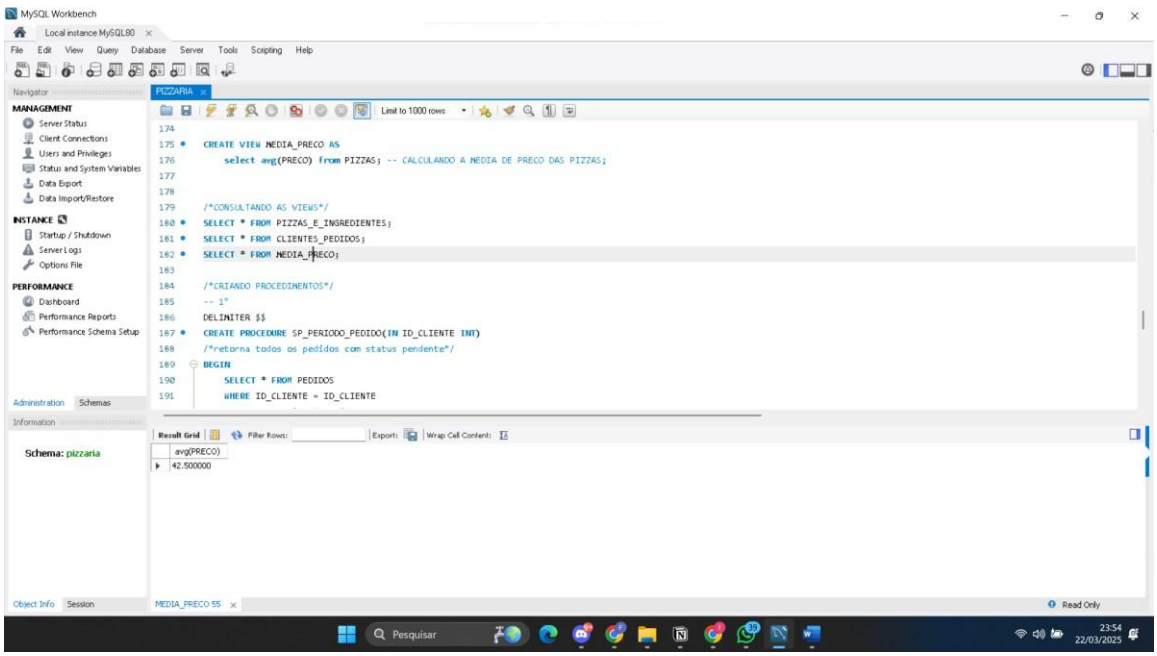

Fig 19 views

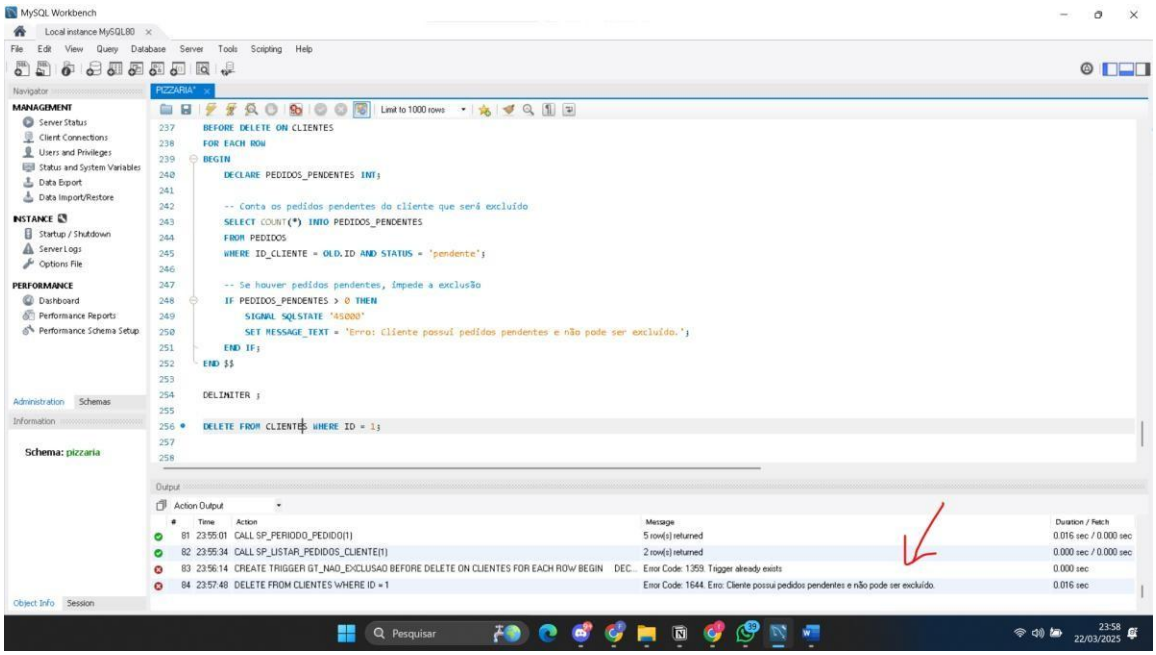

Fig 20 teste gatilho

7. Considerações Finais

O banco de dados desenvolvido atende às necessidades da pizzaria, permitindo a gestão eficiente de pedidos, clientes e produtos. Com o uso de procedimentos armazenados e gatilhos, foi possível automatizar operações críticas, garantindo maior segurança e integridade dos dados.

Futuras melhorias podem incluir a implementação de relatórios detalhados, otimização de consultas e criação de um sistema web para acesso aos dados.

8. Acesso Github:

https://github.com/Franciscofernandes01/Banco_pizzaria

9. Referências

ABNT NBR 6023:2018. Informação e documentação - Referências - Elaboração. Rio de Janeiro: ABNT, 2018.