**UNIVERSIDADE VILA VELHA**

**CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

Diogo Francis Belshoff, Gustavo Suter Gonçalves, Lucas Damasceno Bernardes

BANCO DE DADOS II

Projeto 2º Bimestre

VILA VELHA

2024

Diogo Francis Belshoff, Gustavo Suter Gonçalves, Lucas Damasceno Bernardes

BANCO DE DADOS II

Projeto 2º Bimestre

Trabalho apresentado no curso de Graduação em Ciência da Computação da Universidade Vila Velha.

Orientador: Prof. Jean-Rémi Bourguet

VILA VELHA

2024

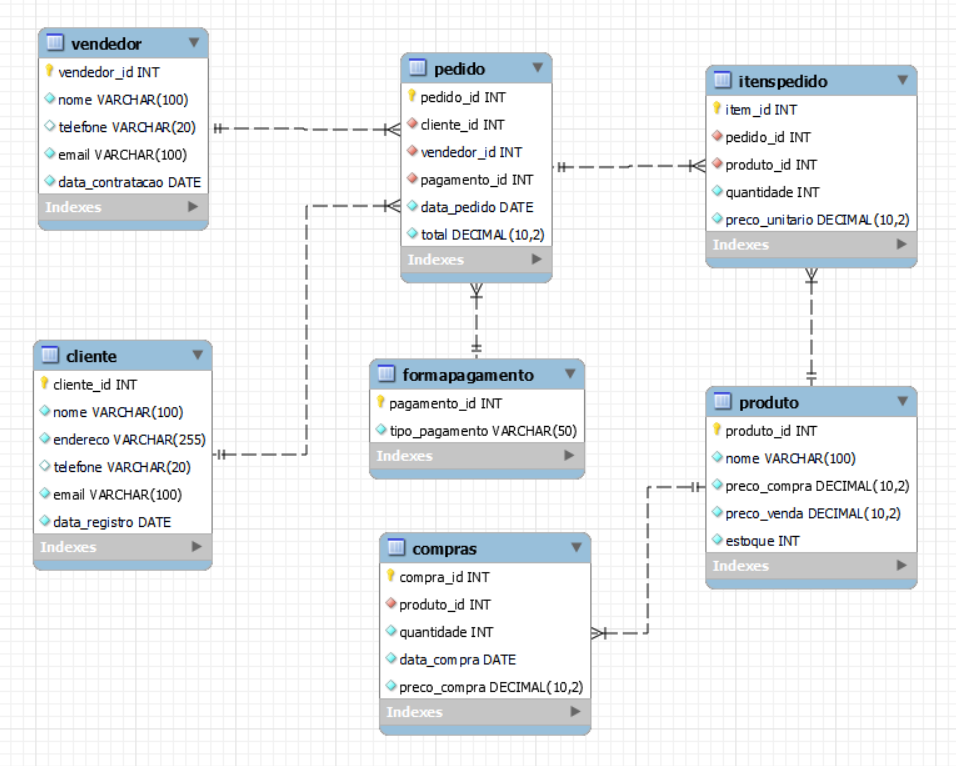
-- DIOGO FRANCIS BELSHOFF

-- GUSTAVO SUTER GONÇALVES

-- LUCAS DAMASCENO BERNARDES

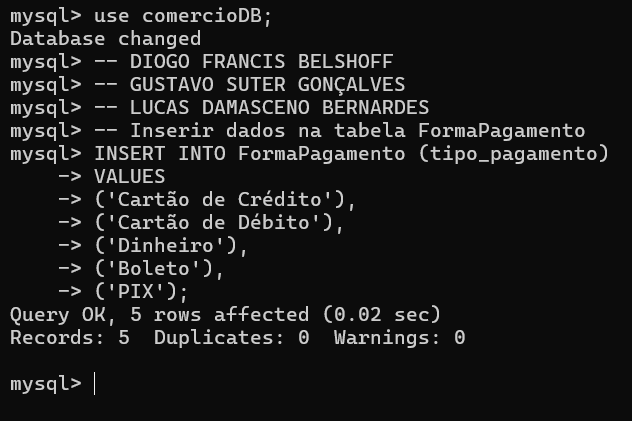
**1 - ARQUITETURA DO BANCO DE DADOS**

Diagrama:



**2 - POPULAR O BANCO DE DADOS**

Populando a tabela FORMAPAGAMENTO:



Para inserir dados nas tabelas CLIENTE, VENDEDOR E PRODUTO foi utilizado a API MOCKAROO para gerar os dados em arquivo .csv e um Script python para fazer a importação dos dados.

Foram gerados CLIENTE (1000 registros), VENDEDOR(20 registros) e PRODUTO(1000 registros)

Script de importação:

*import* mysql.connector

*from* datetime *import* datetime

# *Conectar ao banco de dados MySQL*

db\_connection = mysql.connector.connect(

    host="localhost",

    user="root",

    password="11534728",

    database="comercioDB"

)

cursor = db\_connection.cursor()

# *Ler os dados do arquivo CSV*

*with* open('MOCK\_DATA\_CLIENTE.CSV', 'r') *as* csv\_file:

    next(csv\_file)  # *Pule o cabeçalho*

*for* line *in* csv\_file:

        data = line.strip().split(',')

        # *Converter a data para o formato esperado pelo MySQL (YYYY-MM-DD), pois gerei no formato (dd.mm.yyyy)*

        data[5] = datetime.strptime(data[5], '%d.%m.%Y').strftime('%Y-%m-%d')

        cursor.execute("INSERT INTO Cliente (cliente\_id, nome, endereco, telefone, email, data\_registro) VALUES (%s, %s, %s, %s, %s, %s)", data)

# *Ler os dados do arquivo CSV*

*with* open('MOCK\_DATA\_VENDEDOR.CSV', 'r') *as* csv\_file:

    next(csv\_file)  # *Pule o cabeçalho*

*for* line *in* csv\_file:

        data = line.strip().split(',')

        # *Converter a data para o formato esperado pelo MySQL (YYYY-MM-DD), pois gerei no formato (dd.mm.yyyy)*

        data[4] = datetime.strptime(data[4], '%d.%m.%Y').strftime('%Y-%m-%d')

        cursor.execute("INSERT INTO Vendedor (vendedor\_id, nome, telefone, email, data\_contratacao) VALUES (%s, %s, %s, %s, %s)", data)

# *Ler os dados do arquivo CSV*

*with* open('MOCK\_DATA\_PRODUTO.CSV', 'r') *as* csv\_file:

    next(csv\_file)  # *Pule o cabeçalho*

*for* line *in* csv\_file:

        data = line.strip().split(';')

        cursor.execute("INSERT INTO Produto (produto\_id, nome, preco\_compra, preco\_venda, estoque) VALUES (%s, %s, %s, %s, %s)", data)

# *Confirmar as transações*

db\_connection.commit()

# *Fechar a conexão*

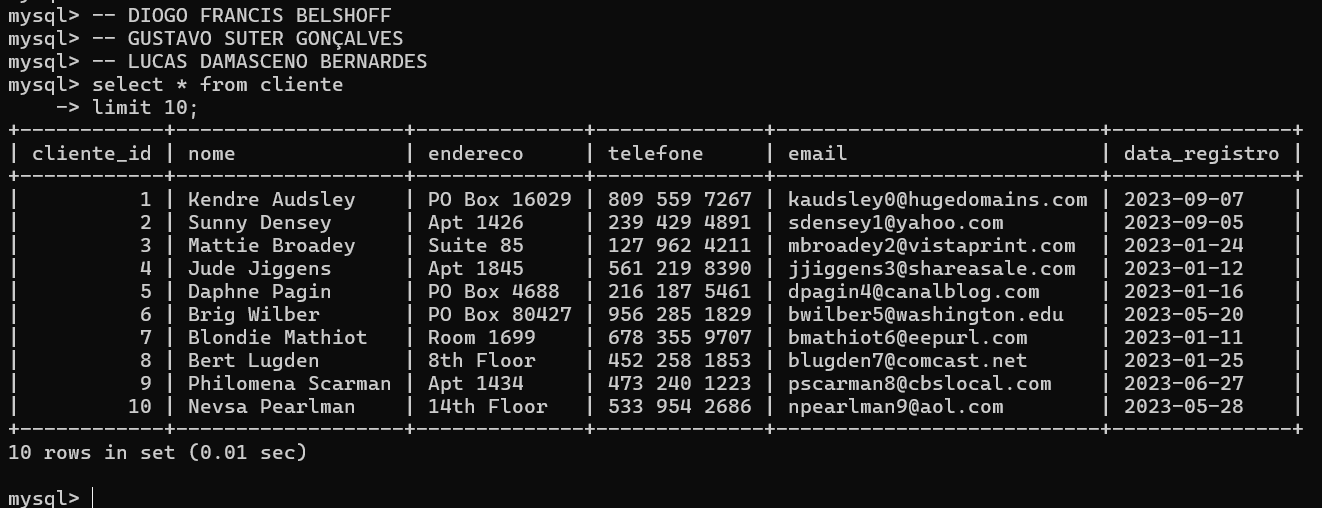
cursor.close()

db\_connection.close()

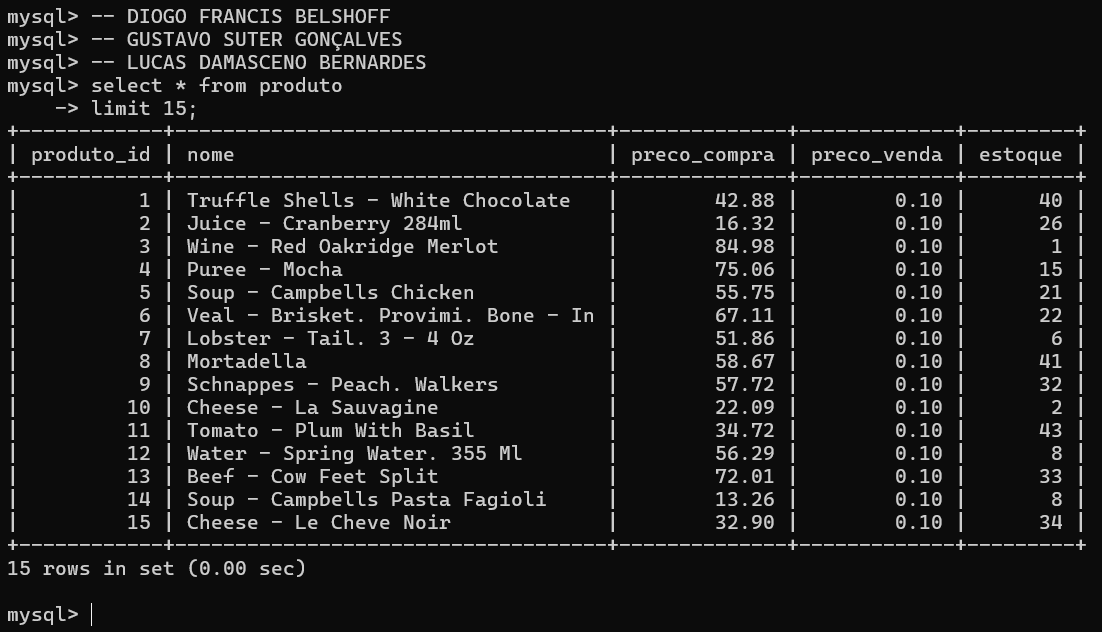
Execução do script:



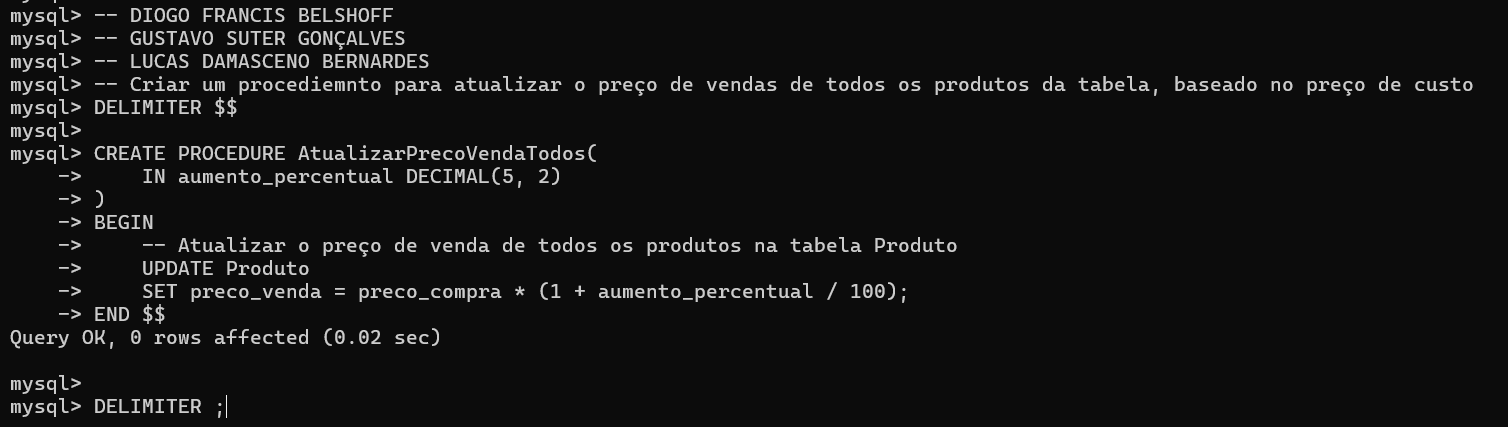
Checando a inserção de dados:



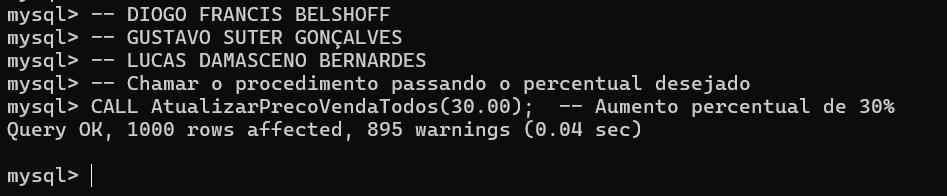




Depois de populado a tabela produtos, vamos criar um procedimento para aumentar o preço de vendas em todos os produtos com uma margem de 30%, em relação ao preço de custo:



Executar o procedimento:



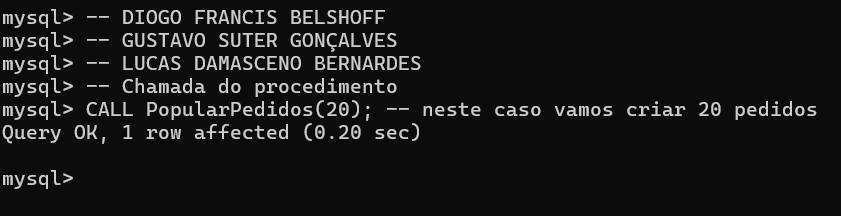
Checagem:



Agora, vamos criar um procedimento para popular as tabelas PEDIDO e ITENSPEDIDO com dados das tabelas acima:



Procedimento criado, vamos criar 20 pedidos:



Checagem:

Tabela PEDIDO populada:

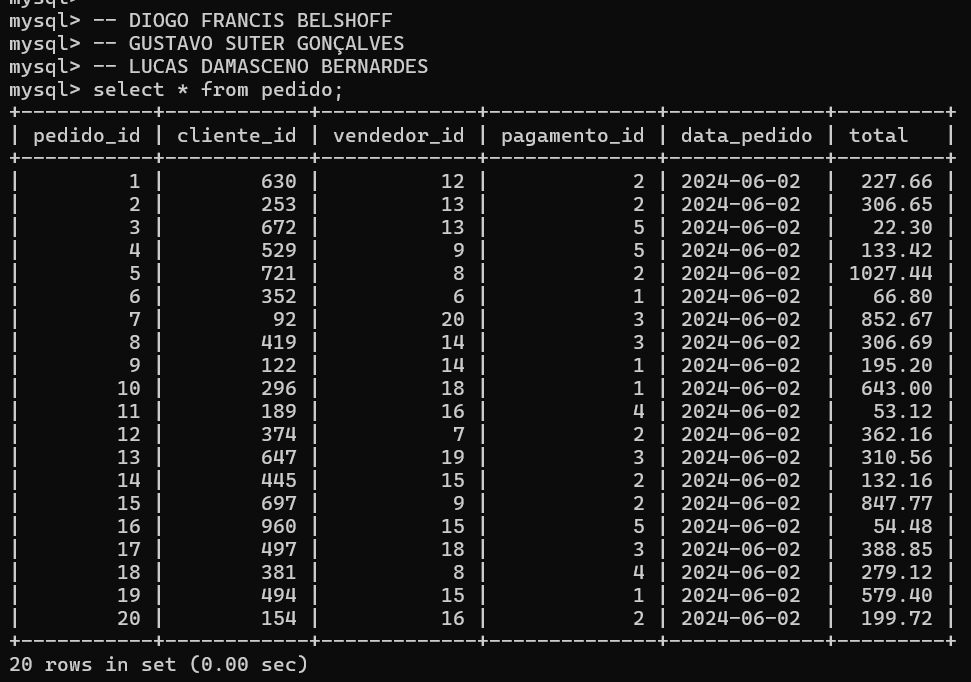
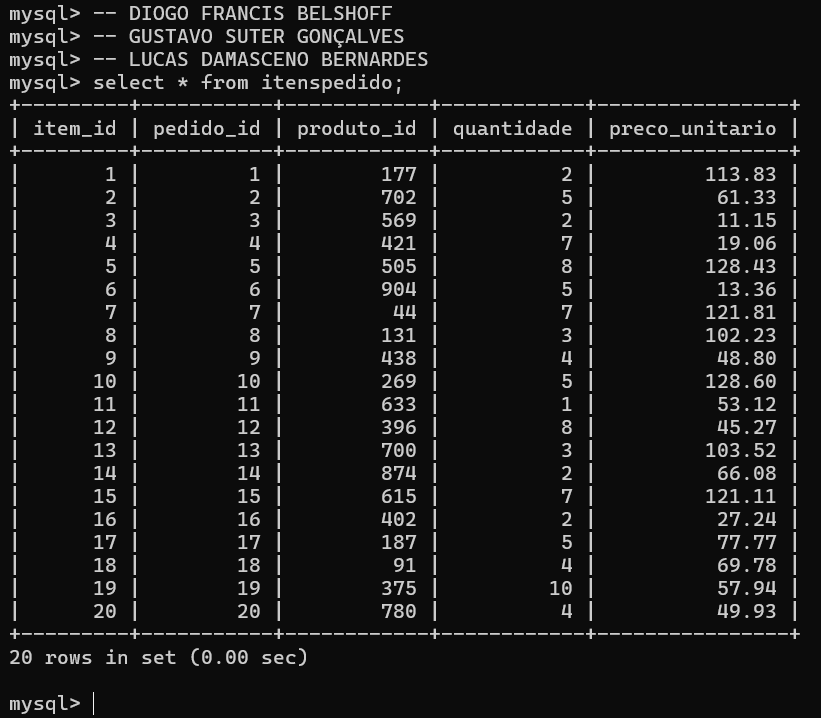
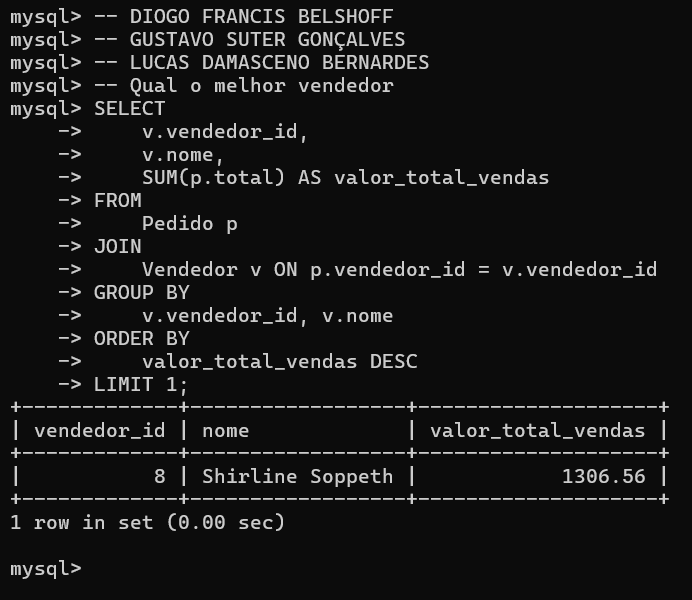


Tabela ITENSPEDIDO populada:

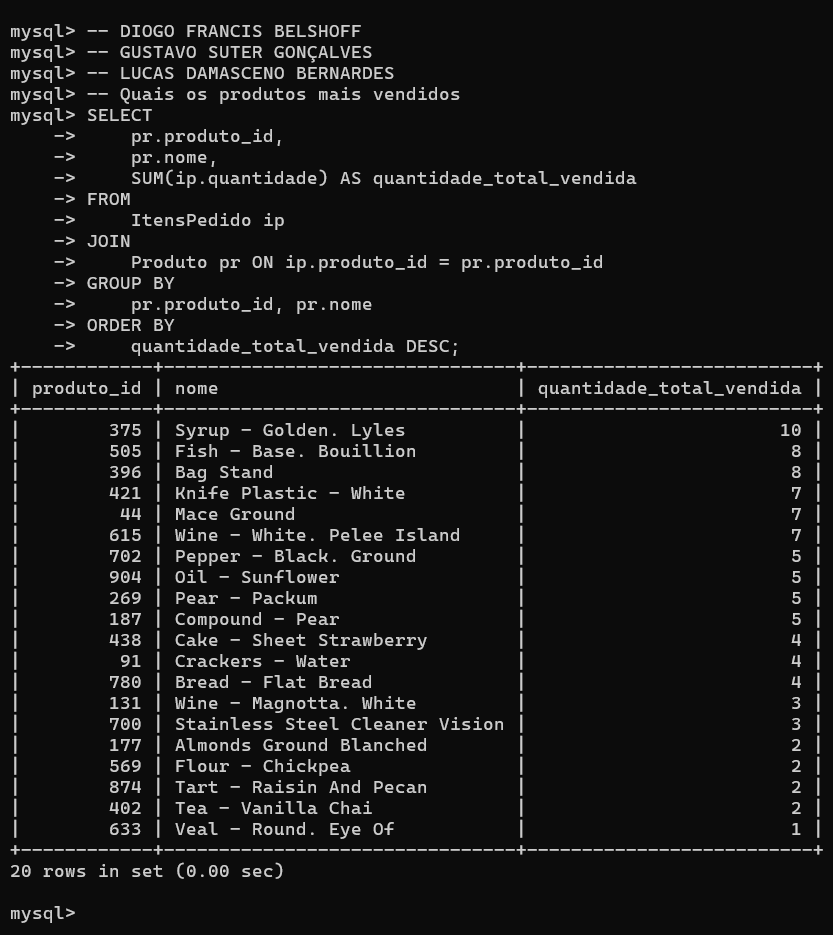


**3 – CONSULTAS**

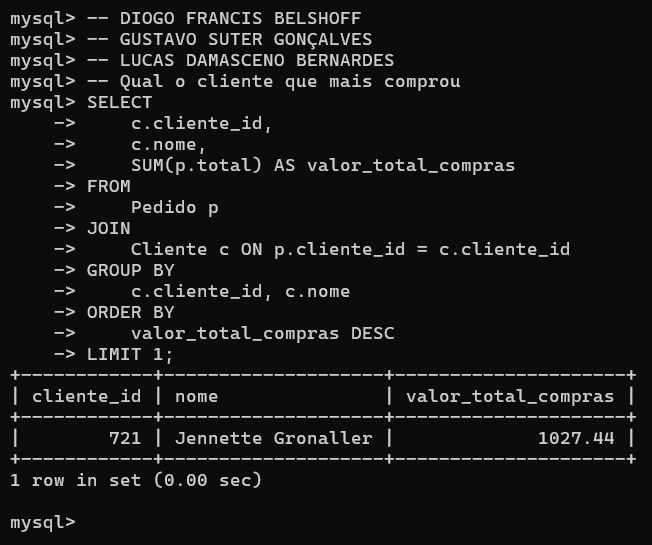
1ª consulta: Qual o melhor vendedor



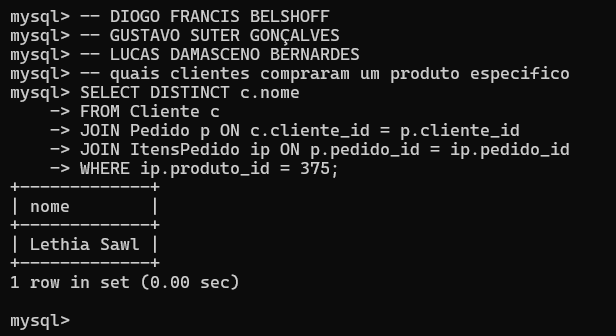
2ª consulta: Quais os produtos mais vendidos



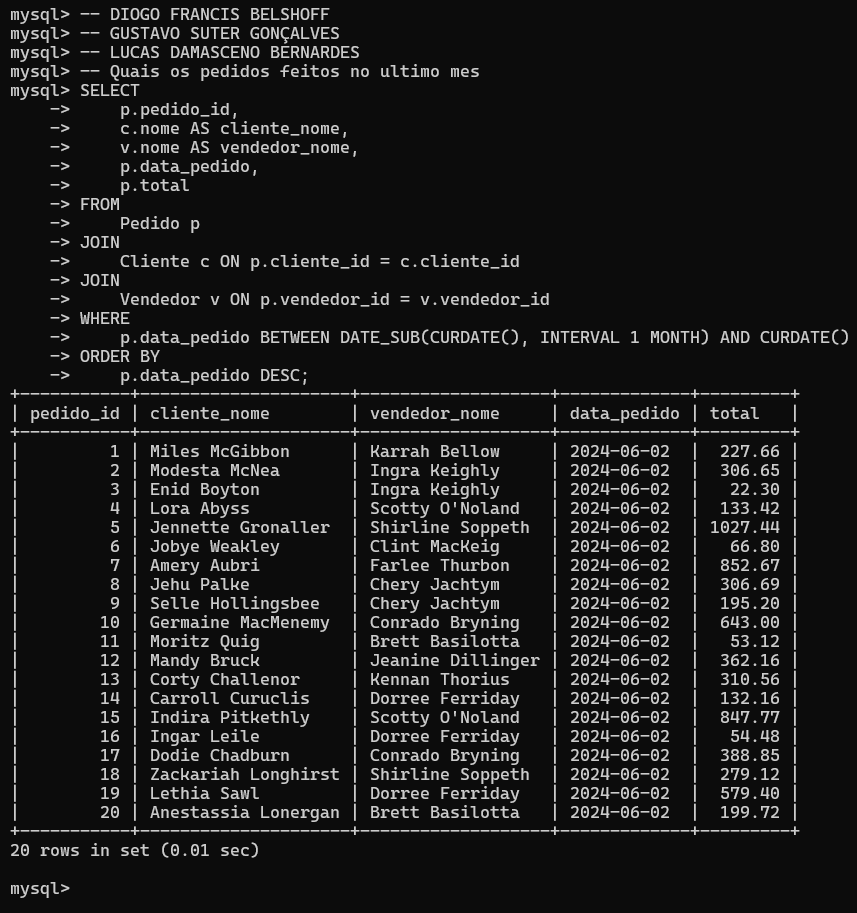
3ª consulta: Qual o cliente que mais comprou:



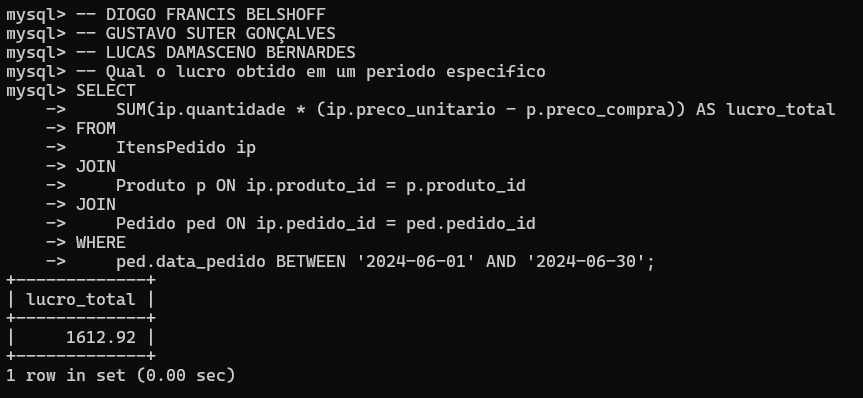
4ª consulta: Quais clientes compraram um determinado produto



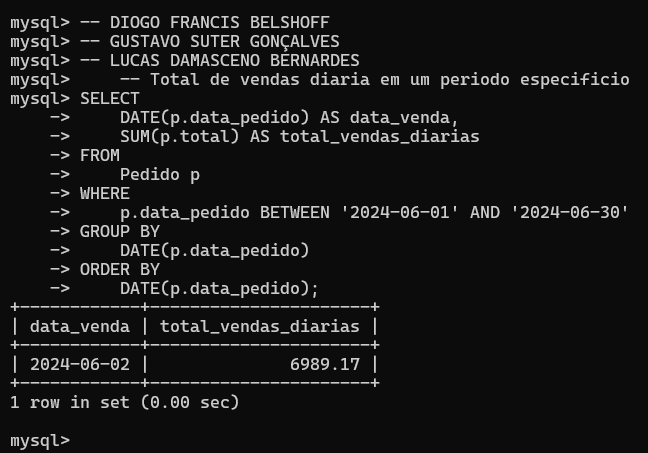
5ª consulta: Quais os pedidos feitos no ultimo mês



6ª consulta: Qual o lucro obtido em um período especifico

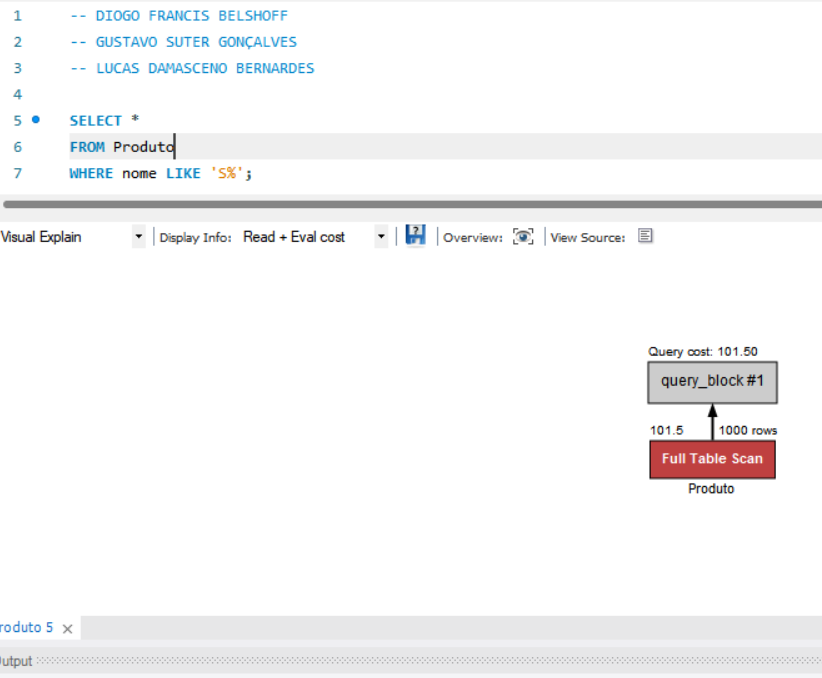


7ª consulta: Qual o total de vendas diária em um período especifico

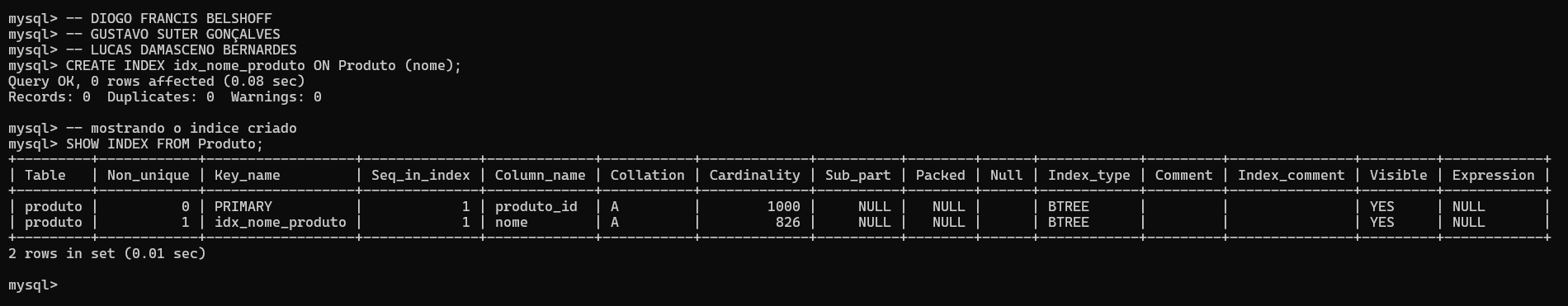


**4 – OTIMIZAR**

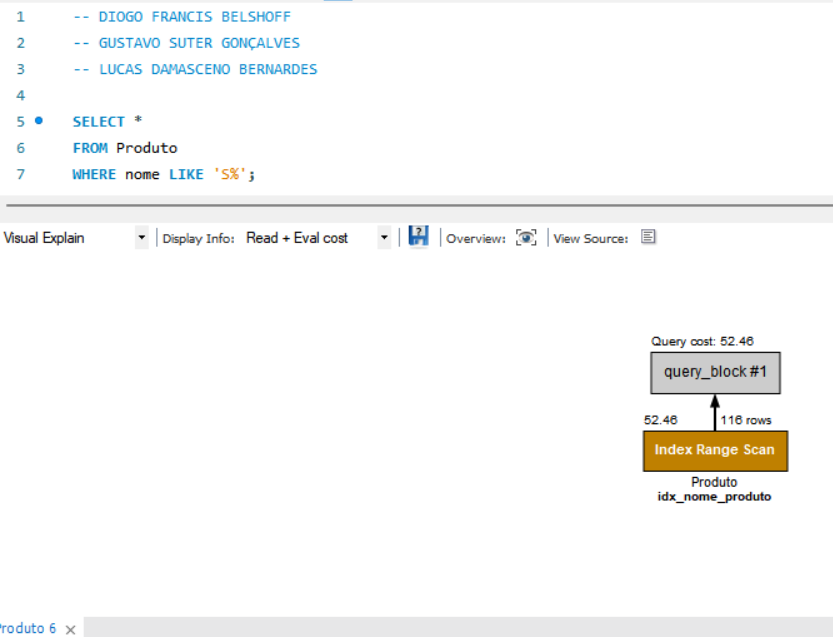
Vamos fazer uma consulta que filtre todos os produtos cuja descrição comece com a letra ‘S’ e observar o custo desta consulta:



Agora vamos adicionar um índice à coluna NOME da tabela PRODUTO e mostra-lo:



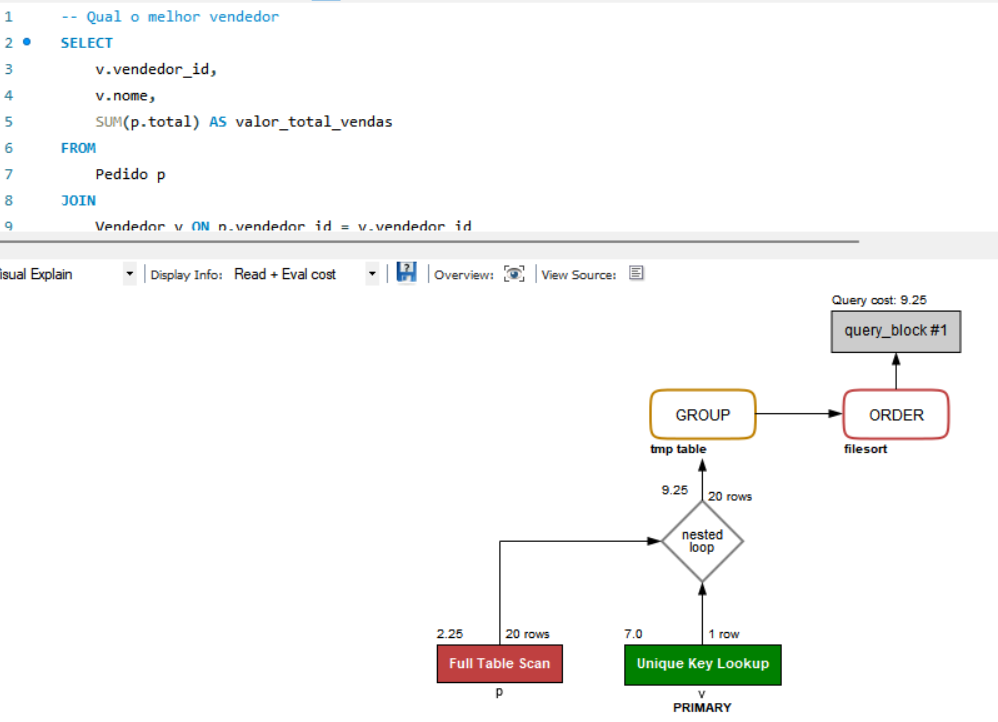
Agora vamos refazer a consulta e checar o custo:



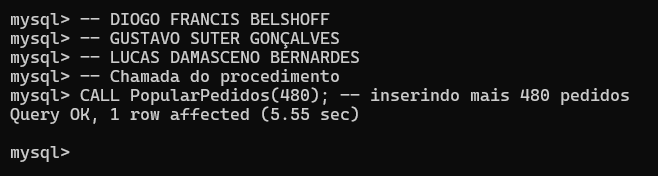
Podemos observar uma melhora significativa, devido ao uso do índice.

Agora vamos otimizar a consulta 1

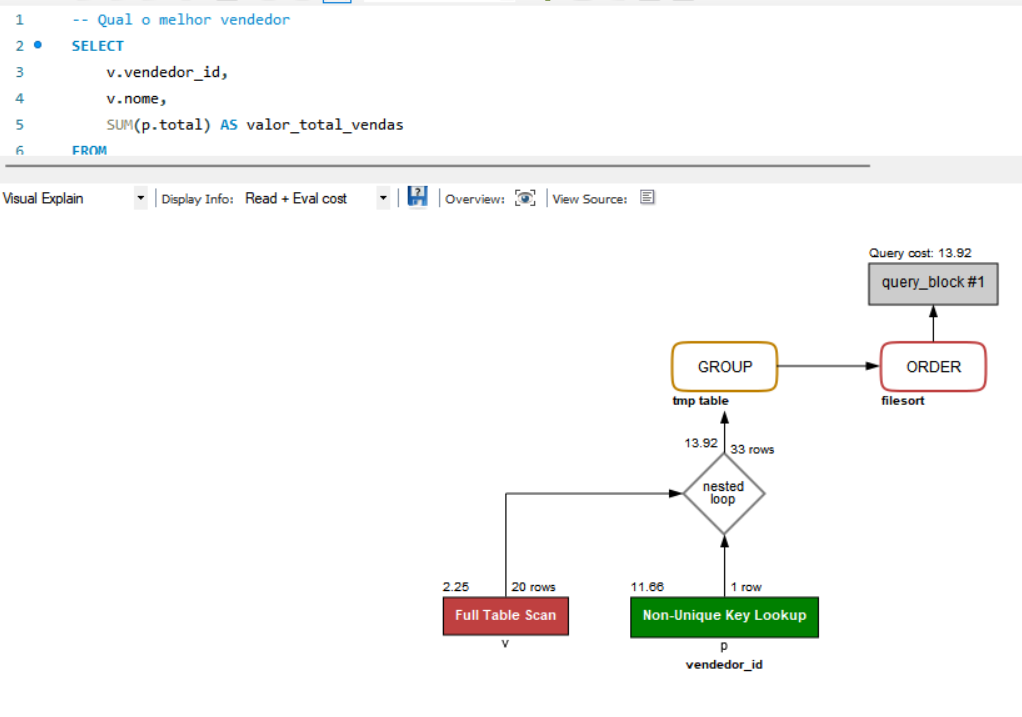
Vamos observar o custo e a execução da consulta com apenas 20 pedidos:



Agora vamos inserir mais 480 pedidos:

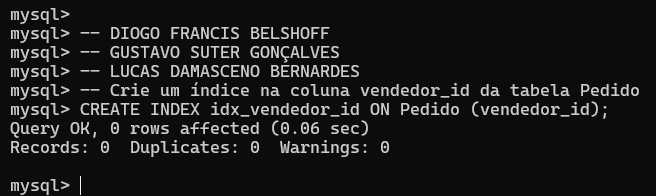


Agora vamos checar novamente o custo e a execução da consulta:

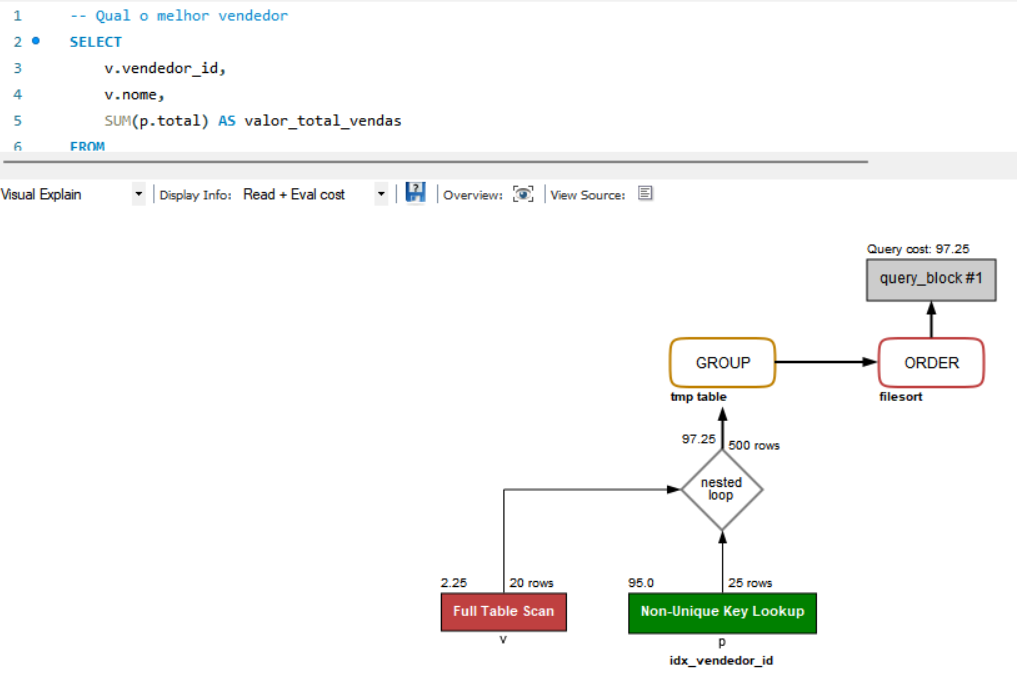


Podemos observar que o custo aumentou um pouco, e o plano de execução também mudou com o aumento da quantidade de registros na tabela PEDIDO.

Vamos criar um índice na coluna VENDEDOR\_ID da tabela PEDIDO, a fim de observar se teremos uma melhora na execução:



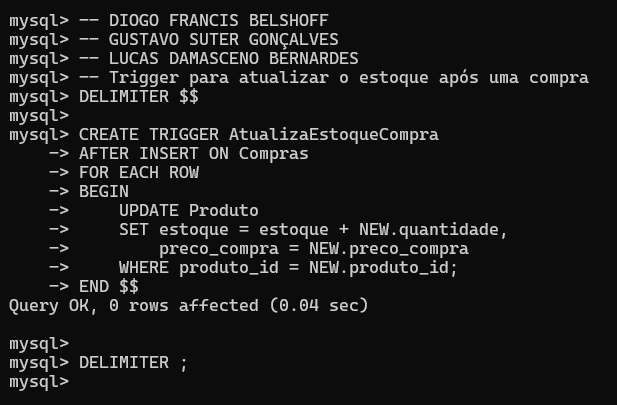
Agora vamos refazer a consulta:



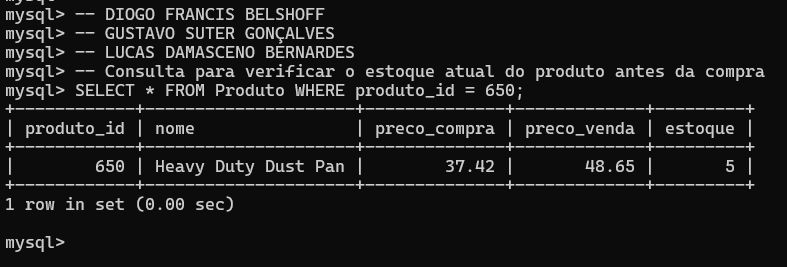
Podemos observar que a inserção do índice piorou a performance da consulta 1.

**5 – ESTRUTURAS AVANÇADAS**

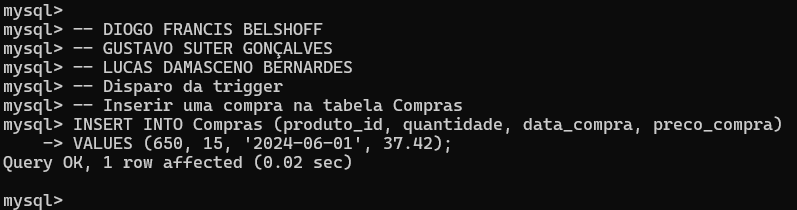
Inserir uma trigger para atualizar o estoque após uma compra:



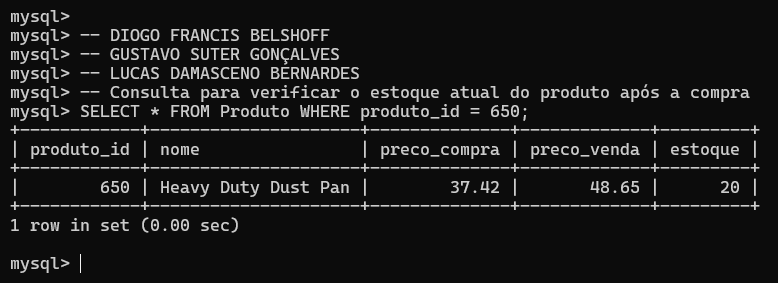
Checar o estoque do produto 650 antes da compra:



Inserir uma compra de 15 unidades do produto 650:

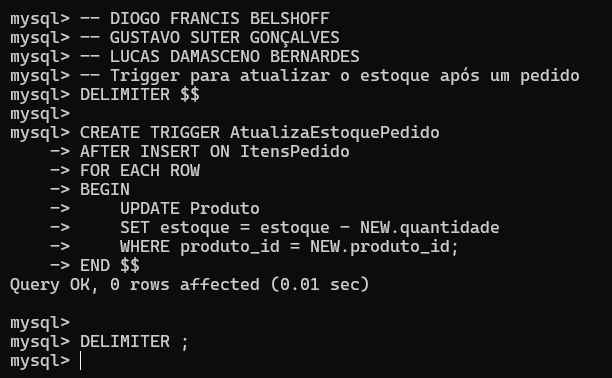


Checar se o estoque atualizou

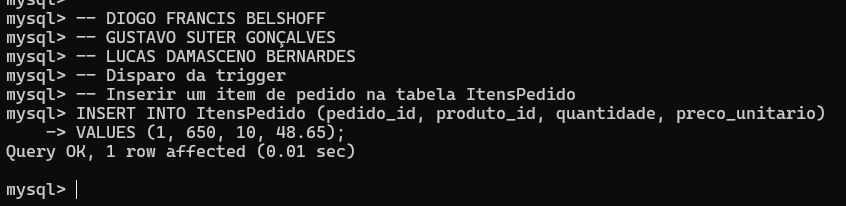


Ok, tudo certo.

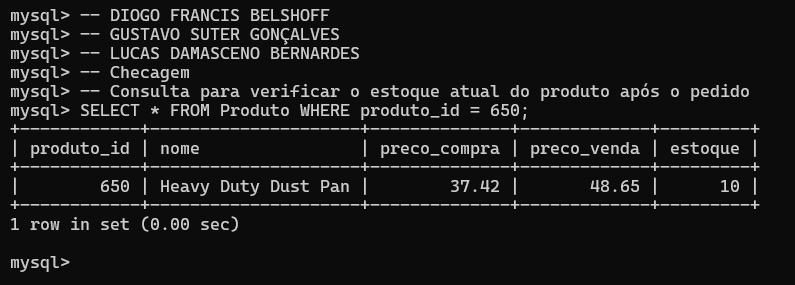
Inserir uma trigger para atualizar o estoque após uma venda



Inserir uma venda de 10 unidades do produto 650



Checar se o estoque atualizou



Ok, tudo certo.

Script:

-- Criar o banco de dados

CREATE DATABASE ComercioDB;

-- Selecionar o banco de dados criado

USE ComercioDB;

-- Tabela Cliente

CREATE TABLE Cliente (

cliente\_id INT AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(100) NOT NULL,

endereco VARCHAR(255) NOT NULL,

telefone VARCHAR(20),

email VARCHAR(100) NOT NULL,

data\_registro DATE NOT NULL

);

-- Tabela Vendedor

CREATE TABLE Vendedor (

vendedor\_id INT AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(100) NOT NULL,

telefone VARCHAR(20),

email VARCHAR(100) NOT NULL,

data\_contratacao DATE NOT NULL

);

-- Tabela Produto

CREATE TABLE Produto (

produto\_id INT AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(100) NOT NULL,

preco\_compra DECIMAL(10, 2) NOT NULL,

preco\_venda DECIMAL(10, 2) NOT NULL,

estoque INT NOT NULL DEFAULT 0

);

-- Tabela FormaPagamento

CREATE TABLE FormaPagamento (

pagamento\_id INT AUTO\_INCREMENT PRIMARY KEY,

tipo\_pagamento VARCHAR(50) NOT NULL

);

-- Tabela Pedido

CREATE TABLE Pedido (

pedido\_id INT AUTO\_INCREMENT PRIMARY KEY,

cliente\_id INT NOT NULL,

vendedor\_id INT NOT NULL,

pagamento\_id INT NOT NULL,

data\_pedido DATE NOT NULL,

total DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (cliente\_id) REFERENCES Cliente(cliente\_id),

FOREIGN KEY (vendedor\_id) REFERENCES Vendedor(vendedor\_id),

FOREIGN KEY (pagamento\_id) REFERENCES FormaPagamento(pagamento\_id)

);

-- Tabela ItensPedido

CREATE TABLE ItensPedido (

item\_id INT AUTO\_INCREMENT PRIMARY KEY,

pedido\_id INT NOT NULL,

produto\_id INT NOT NULL,

quantidade INT NOT NULL,

preco\_unitario DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (pedido\_id) REFERENCES Pedido(pedido\_id),

FOREIGN KEY (produto\_id) REFERENCES Produto(produto\_id)

);

-- Tabela Compras

CREATE TABLE Compras (

compra\_id INT AUTO\_INCREMENT PRIMARY KEY,

produto\_id INT NOT NULL,

quantidade INT NOT NULL,

data\_compra DATE NOT NULL,

preco\_compra DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (produto\_id) REFERENCES Produto(produto\_id)

);

-- POPULAR O BANCO DE DADOS

-- Inserir dados na tabela FormaPagamento

INSERT INTO FormaPagamento (tipo\_pagamento)

VALUES

('Cartão de Crédito'),

('Cartão de Débito'),

('Dinheiro'),

('Boleto'),

('PIX');

-- Criar um procediemnto para atualizar o preço de vendas de todos os produtos da tabela, baseado no preço de custo

DELIMITER $$

CREATE PROCEDURE AtualizarPrecoVendaTodos(

IN aumento\_percentual DECIMAL(5, 2)

)

BEGIN

-- Atualizar o preço de venda de todos os produtos na tabela Produto

UPDATE Produto

SET preco\_venda = preco\_compra \* (1 + aumento\_percentual / 100);

END $$

DELIMITER ;

-- Chamar o procedimento passando o percentual desejado

CALL AtualizarPrecoVendaTodos(30.00); -- Aumento percentual de 30%

-- Procedimento para popular as tabelas PEDIDO e ITENSPEDIDO

DELIMITER $$

CREATE PROCEDURE PopularPedidos(

IN num\_pedidos INT

)

BEGIN

DECLARE cliente\_id\_val INT;

DECLARE vendedor\_id\_val INT;

DECLARE produto\_id\_val INT;

DECLARE pedido\_id\_val INT;

DECLARE quantidade\_val INT;

DECLARE preco\_unitario\_val DECIMAL(10, 2);

DECLARE forma\_pagamento\_val INT;

DECLARE contador INT DEFAULT 0;

DECLARE total\_pedido DECIMAL(10, 2);

-- Cursor para percorrer os produtos

DECLARE cur\_produto CURSOR FOR

SELECT produto\_id FROM Produto;

-- Abrir o cursor

OPEN cur\_produto;

-- Loop para criar pedidos

pedido\_loop: LOOP

IF contador >= num\_pedidos THEN

LEAVE pedido\_loop;

END IF;

-- Selecionar um cliente aleatório

SET cliente\_id\_val = (SELECT cliente\_id FROM Cliente ORDER BY RAND() LIMIT 1);

-- Selecionar um vendedor aleatório

SET vendedor\_id\_val = (SELECT vendedor\_id FROM Vendedor ORDER BY RAND() LIMIT 1);

-- Selecionar um produto aleatório

SET produto\_id\_val = (SELECT produto\_id FROM Produto ORDER BY RAND() LIMIT 1);

-- Gerar uma quantidade vendida aleatória entre 1 e 10

SET quantidade\_val = FLOOR(RAND() \* 10) + 1;

-- Gerar uma forma de pagamento aleatória entre 1 e 5

SET forma\_pagamento\_val = FLOOR(RAND() \* 5) + 1;

-- Obter o preço unitário do produto

SET preco\_unitario\_val = (SELECT preco\_venda FROM Produto WHERE produto\_id = produto\_id\_val);

-- Calcular o total do pedido

SET total\_pedido = quantidade\_val \* preco\_unitario\_val;

-- Inserir o pedido na tabela Pedido

INSERT INTO Pedido (cliente\_id, vendedor\_id, pagamento\_id, data\_pedido, total)

VALUES (cliente\_id\_val, vendedor\_id\_val, forma\_pagamento\_val, CURRENT\_DATE(), total\_pedido);

-- Obter o ID do pedido inserido

SET pedido\_id\_val = LAST\_INSERT\_ID();

-- Inserir os itens do pedido na tabela ItensPedido

INSERT INTO ItensPedido (pedido\_id, produto\_id, quantidade, preco\_unitario)

VALUES (pedido\_id\_val, produto\_id\_val, quantidade\_val, preco\_unitario\_val);

SET contador = contador + 1;

END LOOP pedido\_loop;

-- Fechar o cursor

CLOSE cur\_produto;

END $$

DELIMITER ;

-- Chamada do procedimento

CALL PopularPedidos(); -- Cria a qtd de pedidos passada como paramentro

-- CONSULTAS

-- Qual o melhor vendedor

SELECT

v.vendedor\_id,

v.nome,

SUM(p.total) AS valor\_total\_vendas

FROM

Pedido p

JOIN

Vendedor v ON p.vendedor\_id = v.vendedor\_id

GROUP BY

v.vendedor\_id, v.nome

ORDER BY

valor\_total\_vendas DESC

LIMIT 1;

-- Quais os produtos mais vendidos

SELECT

pr.produto\_id,

pr.nome,

SUM(ip.quantidade) AS quantidade\_total\_vendida

FROM

ItensPedido ip

JOIN

Produto pr ON ip.produto\_id = pr.produto\_id

GROUP BY

pr.produto\_id, pr.nome

ORDER BY

quantidade\_total\_vendida DESC;

-- Qual o cliente que mais comprou

SELECT

c.cliente\_id,

c.nome,

SUM(p.total) AS valor\_total\_compras

FROM

Pedido p

JOIN

Cliente c ON p.cliente\_id = c.cliente\_id

GROUP BY

c.cliente\_id, c.nome

ORDER BY

valor\_total\_compras DESC

LIMIT 1;

-- quais clientes compraram um produto especifico

SELECT DISTINCT c.nome

FROM Cliente c

JOIN Pedido p ON c.cliente\_id = p.cliente\_id

JOIN ItensPedido ip ON p.pedido\_id = ip.pedido\_id

WHERE ip.produto\_id = 375;

-- Quais os pedidos feitos no ultimo mes

SELECT

p.pedido\_id,

c.nome AS cliente\_nome,

v.nome AS vendedor\_nome,

p.data\_pedido,

p.total

FROM

Pedido p

JOIN

Cliente c ON p.cliente\_id = c.cliente\_id

JOIN

Vendedor v ON p.vendedor\_id = v.vendedor\_id

WHERE

p.data\_pedido BETWEEN DATE\_SUB(CURDATE(), INTERVAL 1 MONTH) AND CURDATE()

ORDER BY

p.data\_pedido DESC;

-- Qual o lucro obtido em um periodo especifico

SELECT

SUM(ip.quantidade \* (ip.preco\_unitario - p.preco\_compra)) AS lucro\_total

FROM

ItensPedido ip

JOIN

Produto p ON ip.produto\_id = p.produto\_id

JOIN

Pedido ped ON ip.pedido\_id = ped.pedido\_id

WHERE

ped.data\_pedido BETWEEN '2024-06-01' AND '2024-06-30';

-- Total de vendas diaria em um periodo especificio

SELECT

DATE(p.data\_pedido) AS data\_venda,

SUM(p.total) AS total\_vendas\_diarias

FROM

Pedido p

WHERE

p.data\_pedido BETWEEN '2024-06-01' AND '2024-06-30'

GROUP BY

DATE(p.data\_pedido)

ORDER BY

DATE(p.data\_pedido);

-- Otimizações

-- Filtrar todos os produtos que comecem com a letra s

SELECT \*

FROM Produto

WHERE nome LIKE 'S%';

-- Criar o indice

CREATE INDEX idx\_nome\_produto ON Produto (nome);

-- Mostrar o indice

SHOW INDEX FROM Produto;

-- Crie um índice na coluna vendedor\_id da tabela Pedido

CREATE INDEX idx\_vendedor\_id ON Pedido (vendedor\_id);

-- ESTRUTURAS AVANÇADAS

-- Trigger para atualizar o estoque após uma compra

DELIMITER $$

CREATE TRIGGER AtualizaEstoqueCompra

AFTER INSERT ON Compras

FOR EACH ROW

BEGIN

UPDATE Produto

SET estoque = estoque + NEW.quantidade,

preco\_compra = NEW.preco\_compra

WHERE produto\_id = NEW.produto\_id;

END $$

DELIMITER ;

-- Consulta para verificar o estoque atual do produto antes da compra

SELECT \* FROM Produto WHERE produto\_id = 650;

-- Disparo da trigger

-- Inserir uma compra na tabela Compras

INSERT INTO Compras (produto\_id, quantidade, data\_compra, preco\_compra)

VALUES (650, 15, '2024-06-01', 37.42);

-- Checagem

-- Consulta para verificar o estoque atual do produto após a compra

SELECT \* FROM Produto WHERE produto\_id = 650;

-- Trigger para atualizar o estoque após um pedido

DELIMITER $$

CREATE TRIGGER AtualizaEstoquePedido

AFTER INSERT ON ItensPedido

FOR EACH ROW

BEGIN

UPDATE Produto

SET estoque = estoque - NEW.quantidade

WHERE produto\_id = NEW.produto\_id;

END $$

DELIMITER ;

-- Consulta para verificar o estoque atual do produto antes do pedido

SELECT \* FROM Produto WHERE produto\_id = 650;

-- Disparo da trigger

-- Inserir um item de pedido na tabela ItensPedido

INSERT INTO ItensPedido (pedido\_id, produto\_id, quantidade, preco\_unitario)

VALUES (1, 650, 10, 48.65);

-- Checagem

-- Consulta para verificar o estoque atual do produto após o pedido

SELECT \* FROM Produto WHERE produto\_id = 650;