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

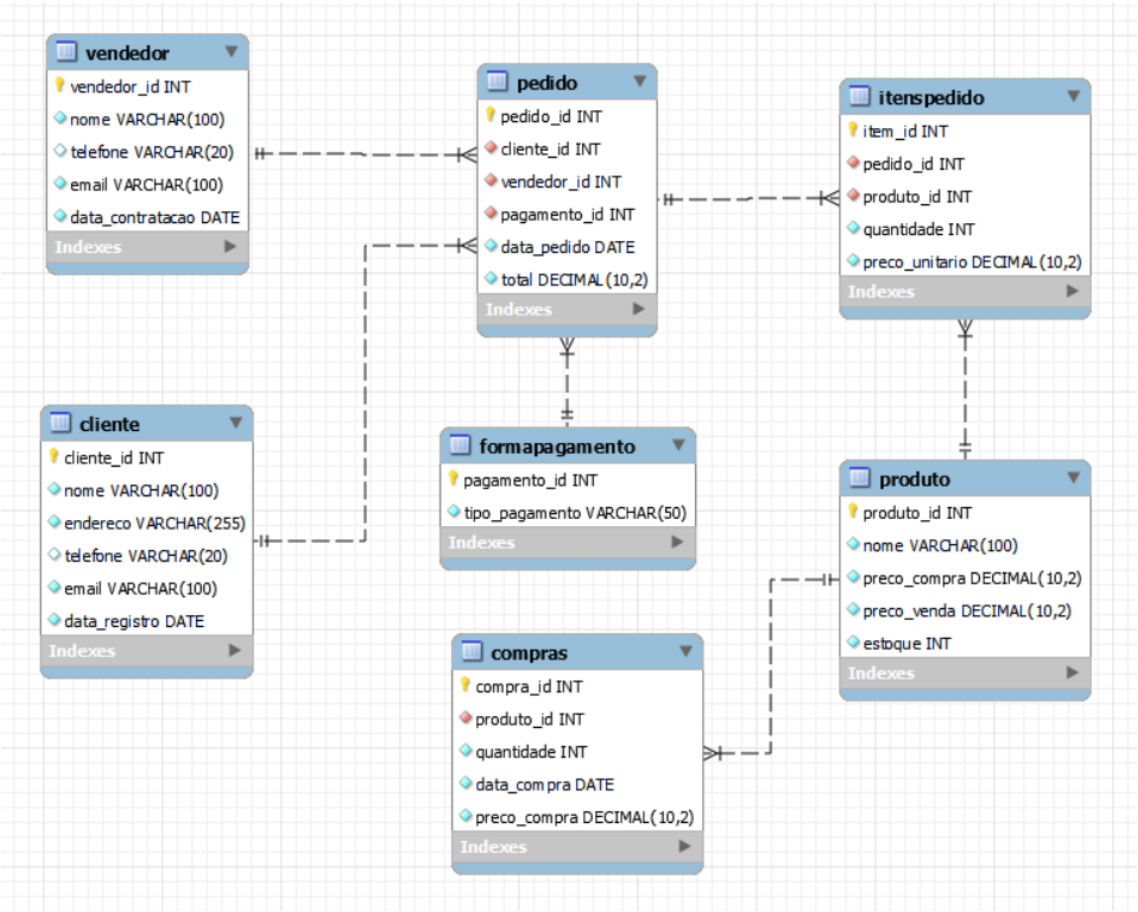
-- DIOGO FRANCIS BELSHOFF

-- GUSTAVO SUTER GONÇALVES

-- LUCAS DAMASCENO BERNARDES

1 - ARQUITETURA DO BANCO DE DADOS

Diagrama:



2 - POPULAR O BANCO DE DADOS

Populando a tabela FORMAPAGAMENTO:

```
mysql> use comercioDB;
Database changed
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Inserir dados na tabela FormaPagamento
mysql> INSERT INTO FormaPagamento (tipo_pagamento)
    -> VALUES
    -> ('Cartão de Crédito'),
    -> ('Cartão de Débito'),
    -> ('Dinheiro'),
    -> ('Boleto'),
    -> ('PIX');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
```

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:

```python
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:

```python
# script_import.py
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()


# DIOGO FRANCIS BELSHOFF
# GUSTAVO SUTER GONÇALVES
# LUCAS DAMASCENO BERNARDES
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL

PS C:\Users\Belshoff\Documents\UVV\2_PERIODO_2024_1\DESIG_E_DESENVOLVIMENTO_DE_BANCO_DE_DADOS_2\2_bimestre> python script_import.py
PS C:\Users\Belshoff\Documents\UVV\2_PERIODO_2024_1\DESIG_E_DESENVOLVIMENTO_DE_BANCO_DE_DADOS_2\2_bimestre>
```

Checando a inserção de dados:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> select * from cliente
    -> limit 10;
+------------+------------------+--------------+--------------+---------------------------+---------------+
| cliente_id | nome             | endereco     | telefone     | email                     | data_registro |
+------------+------------------+--------------+--------------+---------------------------+---------------+
|          1 | Kendre Audsley   | PO Box 16029 | 809 559 7267 | kaudsley0@hugedomains.com | 2023-09-07    |
|          2 | Sunny Densey     | Apt 1426     | 239 429 4891 | sdensey1@yahoo.com        | 2023-09-05    |
|          3 | Mattie Broadey   | Suite 85     | 127 962 4211 | mbroadey2@vistaprint.com  | 2023-01-24    |
|          4 | Jude Jiggens     | Apt 1845     | 561 219 8390 | jjiggens3@shareasale.com  | 2023-01-12    |
|          5 | Daphne Pagin     | PO Box 4688  | 216 187 5461 | dpagin4@canalblog.com     | 2023-01-16    |
|          6 | Brig Wilber      | PO Box 80427 | 956 285 1829 | bwilber5@washington.edu   | 2023-05-20    |
|          7 | Blondie Mathiot  | Room 1699    | 678 355 9707 | bmathiot6@eepurl.com      | 2023-01-11    |
|          8 | Bert Lugden      | 8th Floor    | 452 258 1853 | blugden7@comcast.net      | 2023-01-25    |
|          9 | Philomena Scarman | Apt 1434    | 473 240 1223 | pscarman8@cbslocal.com    | 2023-06-27    |
|         10 | Nevsa Pearlman   | 14th Floor   | 533 954 2686 | npearlman9@aol.com        | 2023-05-28    |
+------------+------------------+--------------+--------------+---------------------------+---------------+
10 rows in set (0.01 sec)

mysql>
```

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> select * from vendedor
    -> limit 5;
+-------------+-----------------+--------------+---------------------------+-----------------+
| vendedor_id | nome            | telefone     | email                     | data_contratacao |
+-------------+-----------------+--------------+---------------------------+-----------------+
|           1 | Daven Chatburn  | 131-691-1960 | dchatburn0@mit.edu        | 2023-07-05      |
|           2 | Barny Zecchinii | 892-209-9307 | bzecchinii1@technorati.com | 2023-12-08      |
|           3 | Bryana Normaville | 157-297-1205 | bnormaville2@hhs.gov    | 2023-09-27      |
|           4 | Ebenezer Foxen  | 409-369-4821 | efoxen3@bbb.org           | 2023-12-17      |
|           5 | Pietra Musson   | 982-613-1662 | pmusson4@bizjournals.com  | 2023-11-06      |
+-------------+-----------------+--------------+---------------------------+-----------------+
5 rows in set (0.01 sec)

mysql>
```

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> select * from produto
    -> limit 15;
+------------+-----------------------------------+--------------+-------------+---------+
| produto_id | nome                              | preco_compra | preco_venda | estoque |
+------------+-----------------------------------+--------------+-------------+---------+
|          1 | Truffle Shells - White Chocolate  |        42.88 |        0.10 |      40 |
|          2 | Juice - Cranberry 284ml           |        16.32 |        0.10 |      26 |
|          3 | Wine - Red Oakridge Merlot        |        84.98 |        0.10 |       1 |
|          4 | Puree - Mocha                     |        75.06 |        0.10 |      15 |
|          5 | Soup - Campbells Chicken          |        55.75 |        0.10 |      21 |
|          6 | Veal - Brisket. Provimi. Bone - In |        67.11 |        0.10 |      22 |
|          7 | Lobster - Tail. 3 - 4 Oz          |        51.86 |        0.10 |       6 |
|          8 | Mortadella                        |        58.67 |        0.10 |      41 |
|          9 | Schnappes - Peach. Walkers        |        57.72 |        0.10 |      32 |
|         10 | Cheese - La Sauvagine             |        22.09 |        0.10 |       2 |
|         11 | Tomato - Plum With Basil          |        34.72 |        0.10 |      43 |
|         12 | Water - Spring Water. 355 Ml      |        56.29 |        0.10 |       8 |
|         13 | Beef - Cow Feet Split             |        72.01 |        0.10 |      33 |
|         14 | Soup - Campbells Pasta Fagioli    |        13.26 |        0.10 |       8 |
|         15 | Cheese - Le Cheve Noir            |        32.90 |        0.10 |      34 |
+------------+-----------------------------------+--------------+-------------+---------+
15 rows in set (0.00 sec)

mysql>
```

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:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Criar um procediemnto para atualizar o preço de vendas de todos os produtos da tabela, baseado no preço de custo
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE AtualizarPrecoVendaTodos(
    ->     IN aumento_percentual DECIMAL(5, 2)
    -> )
    -> BEGIN
    ->     -- Atualizar o preço de venda de todos os produtos na tabela Produto
    ->     UPDATE Produto
    ->     SET preco_venda = preco_compra * (1 + aumento_percentual / 100);
    -> END $$
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
```

Executar o procedimento:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Chamar o procedimento passando o percentual desejado
mysql> CALL AtualizarPrecoVendaTodos(30.00);  -- Aumento percentual de 30%
Query OK, 1000 rows affected, 895 warnings (0.04 sec)

mysql>
```

Checagem:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> select * from produto
    -> limit 10;
+------------+-----------------------------------+--------------+-------------+---------+
| produto_id | nome                              | preco_compra | preco_venda | estoque |
+------------+-----------------------------------+--------------+-------------+---------+
|          1 | Truffle Shells - White Chocolate  |        42.88 |       55.74 |      40 |
|          2 | Juice - Cranberry 284ml           |        16.32 |       21.22 |      26 |
|          3 | Wine - Red Oakridge Merlot        |        84.98 |      110.47 |       1 |
|          4 | Puree - Mocha                     |        75.06 |       97.58 |      15 |
|          5 | Soup - Campbells Chicken          |        55.75 |       72.48 |      21 |
|          6 | Veal - Brisket. Provimi. Bone - In |        67.11 |       87.24 |      22 |
|          7 | Lobster - Tail. 3 - 4 Oz          |        51.86 |       67.42 |       6 |
|          8 | Mortadella                        |        58.67 |       76.27 |      41 |
|          9 | Schnappes - Peach. Walkers        |        57.72 |       75.04 |      32 |
|         10 | Cheese - La Sauvagine             |        22.09 |       28.72 |       2 |
+------------+-----------------------------------+--------------+-------------+---------+
10 rows in set (0.00 sec)

mysql>
```

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

```
mysql>
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Procedimento para popular as tabelas PEDIDO e ITENSPEDIDO
mysql> DELIMITER $$
mysql>
mysql> 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 $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
```

Procedimento criado, vamos criar 20 pedidos:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Chamada do procedimento
mysql> CALL PopularPedidos(20); -- neste caso vamos criar 20 pedidos
Query OK, 1 row affected (0.20 sec)

mysql>
```

Checagem:

Tabela PEDIDO populada:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> select * from pedido;
+-----------+------------+-------------+--------------+-------------+---------+
| pedido_id | cliente_id | vendedor_id | pagamento_id | data_pedido | total   |
+-----------+------------+-------------+--------------+-------------+---------+
|         1 |        630 |          12 |            2 | 2024-06-02  |  227.66 |
|         2 |        253 |          13 |            2 | 2024-06-02  |  306.65 |
|         3 |        672 |          13 |            5 | 2024-06-02  |   22.30 |
|         4 |        529 |           9 |            5 | 2024-06-02  |  133.42 |
|         5 |        721 |           8 |            2 | 2024-06-02  | 1027.44 |
|         6 |        352 |           6 |            1 | 2024-06-02  |   66.80 |
|         7 |         92 |          20 |            3 | 2024-06-02  |  852.67 |
|         8 |        419 |          14 |            3 | 2024-06-02  |  306.69 |
|         9 |        122 |          14 |            1 | 2024-06-02  |  195.20 |
|        10 |        296 |          18 |            1 | 2024-06-02  |  643.00 |
|        11 |        189 |          16 |            4 | 2024-06-02  |   53.12 |
|        12 |        374 |           7 |            2 | 2024-06-02  |  362.16 |
|        13 |        647 |          19 |            3 | 2024-06-02  |  310.56 |
|        14 |        445 |          15 |            2 | 2024-06-02  |  132.16 |
|        15 |        697 |           9 |            2 | 2024-06-02  |  847.77 |
|        16 |        960 |          15 |            5 | 2024-06-02  |   54.48 |
|        17 |        497 |          18 |            3 | 2024-06-02  |  388.85 |
|        18 |        381 |           8 |            4 | 2024-06-02  |  279.12 |
|        19 |        494 |          15 |            1 | 2024-06-02  |  579.40 |
|        20 |        154 |          16 |            2 | 2024-06-02  |  199.72 |
+-----------+------------+-------------+--------------+-------------+---------+
20 rows in set (0.00 sec)
```

Tabela ITENSPEDIDO populada:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> select * from itenspedido;
+---------+-----------+------------+------------+----------------+
| item_id | pedido_id | produto_id | quantidade | preco_unitario |
+---------+-----------+------------+------------+----------------+
|       1 |         1 |        177 |          2 |         113.83 |
|       2 |         2 |        702 |          5 |          61.33 |
|       3 |         3 |        569 |          2 |          11.15 |
|       4 |         4 |        421 |          7 |          19.06 |
|       5 |         5 |        505 |          8 |         128.43 |
|       6 |         6 |        904 |          5 |          13.36 |
|       7 |         7 |         44 |          7 |         121.81 |
|       8 |         8 |        131 |          3 |         102.23 |
|       9 |         9 |        438 |          4 |          48.80 |
|      10 |        10 |        269 |          5 |         128.60 |
|      11 |        11 |        633 |          1 |          53.12 |
|      12 |        12 |        396 |          8 |          45.27 |
|      13 |        13 |        700 |          3 |         103.52 |
|      14 |        14 |        874 |          2 |          66.08 |
|      15 |        15 |        615 |          7 |         121.11 |
|      16 |        16 |        402 |          2 |          27.24 |
|      17 |        17 |        187 |          5 |          77.77 |
|      18 |        18 |         91 |          4 |          69.78 |
|      19 |        19 |        375 |         10 |          57.94 |
|      20 |        20 |        780 |          4 |          49.93 |
+---------+-----------+------------+------------+----------------+
20 rows in set (0.00 sec)

mysql>
```

3 – CONSULTAS

1ª consulta: Qual o melhor vendedor

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Qual o melhor vendedor
mysql> 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;
+-------------+-----------------+--------------------+
| vendedor_id | nome            | valor_total_vendas |
+-------------+-----------------+--------------------+
|           8 | Shirline Soppeth |            1306.56 |
+-------------+-----------------+--------------------+
1 row in set (0.00 sec)

mysql>
```

2ª consulta: Quais os produtos mais vendidos

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Quais os produtos mais vendidos
mysql> 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;
+------------+--------------------------------+--------------------------+
| produto_id | nome                           | quantidade_total_vendida |
+------------+--------------------------------+--------------------------+
|        375 | Syrup - Golden. Lyles          |                       10 |
|        505 | Fish - Base. Bouillion         |                        8 |
|        396 | Bag Stand                      |                        8 |
|        421 | Knife Plastic - White          |                        7 |
|         44 | Mace Ground                    |                        7 |
|        615 | Wine - White. Pelee Island     |                        7 |
|        702 | Pepper - Black. Ground         |                        5 |
|        904 | Oil - Sunflower                |                        5 |
|        269 | Pear - Packum                  |                        5 |
|        187 | Compound - Pear                |                        5 |
|        438 | Cake - Sheet Strawberry        |                        4 |
|         91 | Crackers - Water               |                        4 |
|        780 | Bread - Flat Bread             |                        4 |
|        131 | Wine - Magnotta. White         |                        3 |
|        700 | Stainless Steel Cleaner Vision |                        3 |
|        177 | Almonds Ground Blanched        |                        2 |
|        569 | Flour - Chickpea               |                        2 |
|        874 | Tart - Raisin And Pecan        |                        2 |
|        402 | Tea - Vanilla Chai             |                        2 |
|        633 | Veal - Round. Eye Of           |                        1 |
+------------+--------------------------------+--------------------------+
20 rows in set (0.00 sec)

mysql>
```

3ª consulta: Qual o cliente que mais comprou:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Qual o cliente que mais comprou
mysql> 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;
+------------+-------------------+---------------------+
| cliente_id | nome              | valor_total_compras |
+------------+-------------------+---------------------+
|        721 | Jennette Gronaller |             1027.44 |
+------------+-------------------+---------------------+
1 row in set (0.00 sec)

mysql>
```

4ª consulta: Quais clientes compraram um determinado produto

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- quais clientes compraram um produto especifico
mysql> 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;
+-------------+
| nome        |
+-------------+
| Lethia Sawl |
+-------------+
1 row in set (0.00 sec)

mysql>
```

5ª consulta: Quais os pedidos feitos no ultimo mês

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Quais os pedidos feitos no ultimo mes
mysql> 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;
+-----------+--------------------+------------------+-------------+---------+
| pedido_id | cliente_nome       | vendedor_nome    | data_pedido | total   |
+-----------+--------------------+------------------+-------------+---------+
|         1 | Miles McGibbon     | Karrah Bellow    | 2024-06-02  |  227.66 |
|         2 | Modesta McNea      | Ingra Keighly    | 2024-06-02  |  306.65 |
|         3 | Enid Boyton        | Ingra Keighly    | 2024-06-02  |   22.30 |
|         4 | Lora Abyss         | Scotty O'Noland  | 2024-06-02  |  133.42 |
|         5 | Jennette Gronaller | Shirline Soppeth | 2024-06-02  | 1027.44 |
|         6 | Jobye Weakley      | Clint MacKeig    | 2024-06-02  |   66.80 |
|         7 | Amery Aubri        | Farlee Thurbon   | 2024-06-02  |  852.67 |
|         8 | Jehu Palke         | Chery Jachtym    | 2024-06-02  |  306.69 |
|         9 | Selle Hollingsbee  | Chery Jachtym    | 2024-06-02  |  195.20 |
|        10 | Germaine MacMenemy | Conrado Bryning  | 2024-06-02  |  643.00 |
|        11 | Moritz Quig        | Brett Basilotta  | 2024-06-02  |   53.12 |
|        12 | Mandy Bruck        | Jeanine Dillinger| 2024-06-02  |  362.16 |
|        13 | Corty Challenor    | Kennan Thorius   | 2024-06-02  |  310.56 |
|        14 | Carroll Curuclis   | Dorree Ferriday  | 2024-06-02  |  132.16 |
|        15 | Indira Pitkethly   | Scotty O'Noland  | 2024-06-02  |  847.77 |
|        16 | Ingar Leile        | Dorree Ferriday  | 2024-06-02  |   54.48 |
|        17 | Dodie Chadburn     | Conrado Bryning  | 2024-06-02  |  388.85 |
|        18 | Zackariah Longhirst| Shirline Soppeth | 2024-06-02  |  279.12 |
|        19 | Lethia Sawl        | Dorree Ferriday  | 2024-06-02  |  579.40 |
|        20 | Anestassia Lonergan| Brett Basilotta  | 2024-06-02  |  199.72 |
+-----------+--------------------+------------------+-------------+---------+
20 rows in set (0.01 sec)

mysql>
```

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

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Qual o lucro obtido em um periodo especifico
mysql> 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';
+-------------+
| lucro_total |
+-------------+
|     1612.92 |
+-------------+
1 row in set (0.00 sec)
```

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

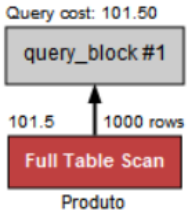
```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql>     -- Total de vendas diaria em um periodo especificio
mysql> SELECT
    ->     DATE(p.data_pedido) AS data_venda,
    ->     SUM(p.total) AS total_vendas_diarias
    -> FROM
    ->     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);
+------------+----------------------+
| data_venda | total_vendas_diarias |
+------------+----------------------+
| 2024-06-02 |              6989.17 |
+------------+----------------------+
1 row in set (0.00 sec)

mysql>
```

4 – OTIMIZAR

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

```
1    -- DIOGO FRANCIS BELSHOFF
2    -- GUSTAVO SUTER GONÇALVES
3    -- LUCAS DAMASCENO BERNARDES
4
5    SELECT *
6    FROM Produto
7    WHERE nome LIKE 'S%';
```

Visual Explain | Display Info: Read + Eval cost | Overview: | View Source:



roduto 5 ×

utput

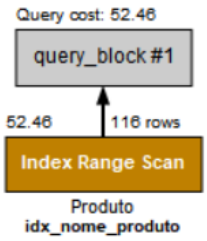
Agora vamos adicionar um índice à coluna NOME da tabela PRODUTO e mostra-lo:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> CREATE INDEX idx_nome_produto ON Produto (nome);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> -- mostrando o indice criado
mysql> SHOW INDEX FROM Produto;
+---------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table   | Non_unique | Key_name          | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+---------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| produto |          0 | PRIMARY           |            1 | produto_id  | A         |        1000 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| produto |          1 | idx_nome_produto  |            1 | nome        | A         |         826 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+---------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
2 rows in set (0.01 sec)

mysql>
```

Agora vamos refazer a consulta e checar o custo:

```
1     -- DIOGO FRANCIS BELSHOFF
2     -- GUSTAVO SUTER GONÇALVES
3     -- LUCAS DAMASCENO BERNARDES
4
5 •   SELECT *
6     FROM Produto
7     WHERE nome LIKE 'S%';
```

Visual Explain | Display Info: Read + Eval cost | Overview: | View Source:



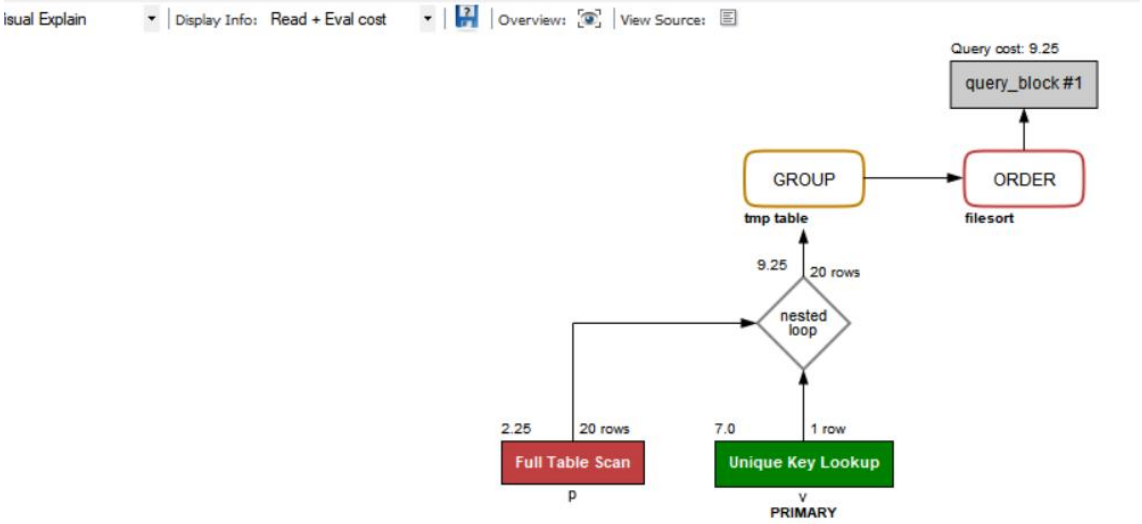
roduto 6 ×

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:

```
1    -- Qual o melhor vendedor
2 •  SELECT
3        v.vendedor_id,
4        v.nome,
5        SUM(p.total) AS valor_total_vendas
6    FROM
7        Pedido p
8    JOIN
9        Vendedor v ON p.vendedor_id = v.vendedor_id
```



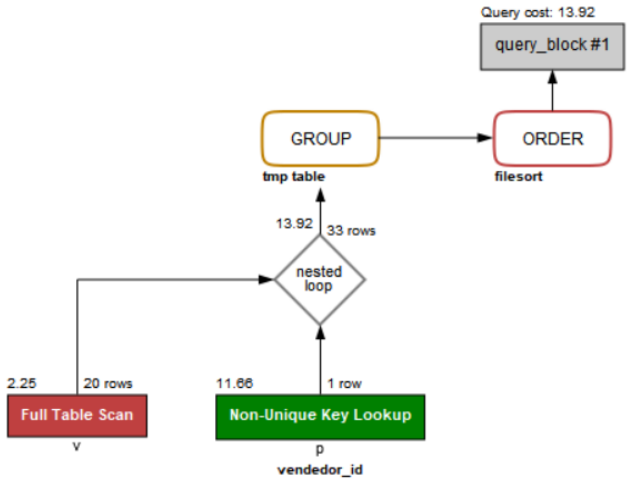
Agora vamos inserir mais 480 pedidos:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Chamada do procedimento
mysql> CALL PopularPedidos(480); -- inserindo mais 480 pedidos
Query OK, 1 row affected (5.55 sec)

mysql>
```

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

```
1     -- Qual o melhor vendedor
2 •   SELECT
3         v.vendedor_id,
4         v.nome,
5         SUM(p.total) AS valor_total_vendas
6     FROM
```

Visual Explain | Display Info: Read + Eval cost | Overview: | View Source:



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.

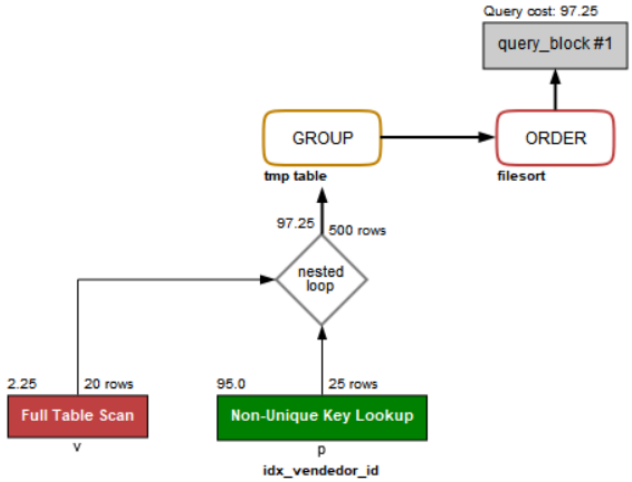
Vamos criar um índice na coluna VENDEDOR_ID da tabela PEDIDO, a fim de observar se teremos uma melhora na execução:

```
mysql>
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Crie um índice na coluna vendedor_id da tabela Pedido
mysql> CREATE INDEX idx_vendedor_id ON Pedido (vendedor_id);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Agora vamos refazer a consulta:

```sql
-- Qual o melhor vendedor
SELECT
    v.vendedor_id,
    v.nome,
    SUM(p.total) AS valor_total_vendas
FROM
```

Visual Explain | Display Info: Read + Eval cost | Overview: | View Source:



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:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Trigger para atualizar o estoque após uma compra
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER AtualizaEstoqueCompra
    -> AFTER INSERT ON Compras
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE Produto
    ->     SET estoque = estoque + NEW.quantidade,
    ->         preco_compra = NEW.preco_compra
    ->     WHERE produto_id = NEW.produto_id;
    -> END $$
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> DELIMITER ;
mysql>
```

Checar o estoque do produto 650 antes da compra:

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Consulta para verificar o estoque atual do produto antes da compra
mysql> SELECT * FROM Produto WHERE produto_id = 650;
+------------+--------------------+--------------+-------------+---------+
| produto_id | nome               | preco_compra | preco_venda | estoque |
+------------+--------------------+--------------+-------------+---------+
|        650 | Heavy Duty Dust Pan |        37.42 |       48.65 |       5 |
+------------+--------------------+--------------+-------------+---------+
1 row in set (0.00 sec)

mysql>
```

Inserir uma compra de 15 unidades do produto 650:

```
mysql>
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Disparo da trigger
mysql> -- Inserir uma compra na tabela Compras
mysql> INSERT INTO Compras (produto_id, quantidade, data_compra, preco_compra)
    -> VALUES (650, 15, '2024-06-01', 37.42);
Query OK, 1 row affected (0.02 sec)

mysql>
```

Checar se o estoque atualizou

```
mysql>
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Consulta para verificar o estoque atual do produto após a compra
mysql> SELECT * FROM Produto WHERE produto_id = 650;
+------------+---------------------+--------------+-------------+----------+
| produto_id | nome                | preco_compra | preco_venda | estoque |
+------------+---------------------+--------------+-------------+----------+
|        650 | Heavy Duty Dust Pan |        37.42 |       48.65 |      20 |
+------------+---------------------+--------------+-------------+----------+
1 row in set (0.00 sec)

mysql>
```

Ok, tudo certo.

Inserir uma trigger para atualizar o estoque após uma venda

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Trigger para atualizar o estoque após um pedido
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER AtualizaEstoquePedido
    -> AFTER INSERT ON ItensPedido
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE Produto
    ->     SET estoque = estoque - NEW.quantidade
    ->     WHERE produto_id = NEW.produto_id;
    -> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
```

Inserir uma venda de 10 unidades do produto 650

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Disparo da trigger
mysql> -- Inserir um item de pedido na tabela ItensPedido
mysql> INSERT INTO ItensPedido (pedido_id, produto_id, quantidade, preco_unitario)
    -> VALUES (1, 650, 10, 48.65);
Query OK, 1 row affected (0.01 sec)

mysql>
```

Checar se o estoque atualizou

```
mysql> -- DIOGO FRANCIS BELSHOFF
mysql> -- GUSTAVO SUTER GONÇALVES
mysql> -- LUCAS DAMASCENO BERNARDES
mysql> -- Checagem
mysql> -- Consulta para verificar o estoque atual do produto após o pedido
mysql> SELECT * FROM Produto WHERE produto_id = 650;
+------------+---------------------+--------------+-------------+---------+
| produto_id | nome                | preco_compra | preco_venda | estoque |
+------------+---------------------+--------------+-------------+---------+
|        650 | Heavy Duty Dust Pan |        37.42 |       48.65 |      10 |
+------------+---------------------+--------------+-------------+---------+
1 row in set (0.00 sec)

mysql>
```

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;
```

```sql
-- 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;
```