Olinda 24 de Fevereiro de 2024
Edson Henrique Ferreira da Silva
Desafio parte 02

Consultas SQL

1. Quantos pedidos foram feitos?

```
SELECT COUNT(DISTINCT order_id) AS total_pedidos
FROM orders;
```

2. Qual é o produto mais vendido em termos de quantidade?

```
SELECT product_id, product_name, SUM(quantity) AS total_vendido
FROM order_items
JOIN products ON order_items.product_id = products.product_id
GROUP BY product_id, product_name
ORDER BY total_vendido DESC
LIMIT 1;
```

3. Qual cliente gastou mais dinheiro em compras?

```
SELECT customer_id, SUM(prince * quantity) AS total_gasto
FROM orders
JOIN order_items ON orders.order_id = order_items.order_id
JOIN products ON order_items.product_id = products.product_id
GROUP BY customer_id
ORDER BY total_gasto DESC
LIMIT 1;
```

4. Quais são os produtos que foram vendidos em todos os pedidos?

```
SELECT product_id, product_name
FROM products
WHERE product_id IN (
   SELECT DISTINCT product_id
   FROM order_items
   GROUP BY order_id
   HAVING COUNT(DISTINCT order_id) = (SELECT COUNT(DISTINCT order_id) FROM
orders)
);
```

5. Qual o dia da semana com o maior número de pedidos?

```
SELECT DAYNAME(order_date) AS dia_semana, COUNT(order_id) AS total_pedidos
FROM orders
GROUP BY dia_semana
ORDER BY total_pedidos DESC
LIMIT 1;
```

Agora, para a segunda parte do desafio, criamos um script SQL que transforma os dados brutos do banco de dados de vendas no formato adequado para o armazenamento de um Data Lake. Supondo que você quer extrair dados para análises futuras, pode-se criar uma tabela agregada com as informações relevantes.

Script SQL para Transformação de Dados:

```
-- Crie uma tabela para armazenar dados agregados

CREATE TABLE data_lake_sales AS

SELECT

    o.order_id,

    o.customer_id,

    o.order_date,

    p.product_id,

    p.product_name,

    p.product_type,

    oi.quantity,

    p.prince,

    oi.quantity * p.prince AS total_item_price,

    SUM(oi.quantity * p.prince) OVER (PARTITION BY o.order_id) AS total_order_price

FROM orders o

JOIN order_items oi ON o.order_id = oi.order_id

JOIN products p ON oi.product_id = p.product_id;
```

Este script cria uma tabela chamada `data_lake_sales` que contém informações agregadas sobre cada pedido, incluindo o valor total do pedido. Certifique-se de adaptar o script de acordo com suas necessidades específicas e a estrutura real do seu banco de dados.

Considerações de Desempenho:

- Utilize índices nas colunas que são usadas em cláusulas WHERE e JOINs para melhorar o desempenho.
- Avalie o plano de execução da consulta e ajuste conforme necessário.
- Considere a criação de índices em colunas usadas frequentemente em operações de junção.

Lembre-se de que essas são sugestões iniciais, e você pode precisar adaptar conforme a estrutura real do seu banco de dados e requisitos específicos. Documente as decisões tomadas e qualquer otimização feita para garantir a compreensão futura. Se tiver mais perguntas ou precisar de ajustes, estou à disposição.

Algumas melhorias e considerações adicionais:

1. Indexação:

- Certifique-se de que as colunas utilizadas em cláusulas WHERE e JOINs estejam indexadas para otimizar o desempenho, principalmente `order_id`, `product_id`, e `customer_id`.

-- Exemplo de criação de índices

CREATE INDEX idx_orders_order_id ON orders(order_id);

CREATE INDEX idx_order_items_order_id ON order_items(order_id);

CREATE INDEX idx_order_items_product_id ON order_items(product_id);

CREATE INDEX idx_products_product_id ON products(product_id);

2. Uso de JOINs Explícitos:

- Utilize JOINs explícitos para melhorar a legibilidade e manutenção do código.

SELECT

o.order_id,

o.customer_id,

o.order_date,

p.product_id,

p.product_name,

p.product_type,

oi.quantity,

p.prince,

oi.quantity * p.prince AS total_item_price,

SUM(oi.quantity * p.prince) OVER (PARTITION BY o.order_id) AS total_order_price

FROM orders o

JOIN order_items oi ON o.order_id = oi.order_id

JOIN products p ON oi.product_id = p.product_id;

3. Ajustes na Agregação:

- Se necessário, ajuste a lógica de agregação de acordo com os requisitos específicos da análise que a equipe de ciência de dados pretende realizar.

4. Comentários:

- Adicione comentários explicativos para cada parte do script, destacando a lógica e a razão por trás de cada operação.

```
-- Este script cria uma tabela agregada para armazenar dados para análises futuras.
-- Ele inclui informações sobre cada pedido, produto e quantidades.
-- Certifique-se de adaptar conforme necessário.
CREATE TABLE data_lake_sales AS
SELECT
    o.order_id,
    o.customer_id,
    o.order_date,
    p.product_id,
    p.product_name,
    p.product_type,
    oi.quantity,
    p.prince,
    oi.quantity * p.prince AS total_item_price,
    SUM(oi.quantity * p.prince) OVER (PARTITION BY o.order_id) AS total_order_price
FROM orders o
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id;
```

Estas são apenas sugestões para aprimorar a legibilidade e o desempenho do script. Podendo se adaptar conforme necessário, considerando as características específicas do banco de dados e requisitos da equipe de ciência de dados.