--- USO DA VIEW ---

A criação da View *vw_pedido_detalhado* tem como objetivo simplificar consultas complexas que envolvem informações dispersas em diversas tabelas. Ao invés de repetir joins e ocasionais em consultas frequentes, a visão é encapsulada, tornando as consultas mais legíveis e facilitando a manutenção do código. A visualização agrega informações relacionadas a pedidos, clientes, status e itens de pedido em uma estrutura mais simples.

--- USO DA FUNÇÃO ---

A criação da Função *calcular_preco_total_pedido* tem como objetivo encapsular a lógica de cálculo do preço total de um pedido. Isso promove a reutilização de código e fornece uma abstração útil para calcular o preço total em diferentes partes do sistema. A função pode ser chamada em diferentes partes do sistema sempre que o preço total de um pedido precise ser calculado. As alterações na lógica de cálculo podem ser feitas centralmente na função, garantindo consistência em todo o sistema.

--- USO DA STORED PROCEDURE ---

A criação do Stored Procedure *processar_pedido* tem como objetivo agrupar um conjunto de instruções SQL relacionadas ao processamento de um pedido. Isso facilita a manutenção e execução consistente de tarefas específicas relacionadas aos pedidos. Um procedimento armazenado agrupa a atualização do status do pedido e a inserção na tabela de vendas em uma única unidade lógica, assim facilita a manutenção ao centralizar o código relacionado aos pedidos.

--- USO DA TRIGGER ---

Uma trigger em um banco de dados é um conjunto de instruções que são automaticamente executadas (ou "acionadas") em resposta a um determinado evento em uma tabela ou exibição. Esses eventos podem ser a inserção, atualização ou exclusão de registros em uma tabela. As triggers são usadas para automatizar a execução de ações específicas, como a atualização de outras tabelas, validações de dados ou envio de notificações.

Exemplo de Trigger: Vamos supor que sempre que um novo cliente for adicionado à tabela cliente, queremos registrar essa adição em uma tabela de log chamada log_cliente.

```
DELIMITER //

CREATE TRIGGER after_insert_cliente

AFTER INSERT ON cliente FOR EACH ROW

BEGIN

   INSERT INTO log_cliente (acao, cliente_id, data_hora)

   VALUES ('Inserção de Cliente', NEW.id_cliente, NOW());

END;

//

DELIMITER ;
```

Neste exemplo:

'"after_insert_cliente"' é o nome da trigger.

'"AFTER INSERT ON cliente"' especifica que a trigger será acionada após a inserção de um novo registro na tabela cliente.

'"FOR EACH ROW"' indica que a trigger será executada para cada linha afetada pela instrução INSERT.

'"BEGIN...END"' contém as instruções que serão executadas quando a trigger for acionada.

'"NEW.id_cliente"' refere-se ao novo valor da coluna id_cliente no registro recém-inserido.

'"NOW()"' retorna a data e hora atuais.

Quando um novo registro é inserido na tabela cliente, a trigger after_insert_cliente é acionada automaticamente após a conclusão da operação de inserção. Ela executa as instruções dentro do bloco BEGIN...END, que, neste caso, inserem uma entrada na tabela log_cliente.