```
-- Criação da tabela Pedidos
CREATE TABLE Pedidos (
  pedido_id NUMBER PRIMARY KEY,
  cliente_id NUMBER,
  fornecedor_id NUMBER,
  valor_total NUMBER,
  status VARCHAR2(50)
);

-- Criação da tabela Fornecedor
CREATE TABLE Fornecedor (
  fornecedor_id NUMBER PRIMARY KEY,
  nome VARCHAR2(50),
  status_pedido VARCHAR2(50)
);

-- Criação da tabela Cliente
CREATE TABLE Cliente (
  cliente_id NUMBER PRIMARY KEY,
  nome VARCHAR2(50),
  estoque_minimo NUMBER,
  canal_comunicacao VARCHAR2(50)
);

-- Criação da tabela Produtos
CREATE TABLE Produtos (
  produto_id NUMBER PRIMARY KEY,
  nome VARCHAR2(50),
  preco NUMBER,
  estoque_atual NUMBER
);

-- Criação da tabela Itens_pedido
CREATE TABLE Itens_pedido (
  item_pedido_id NUMBER PRIMARY KEY,
  pedido_id NUMBER,
  produto_id NUMBER,
  quantidade NUMBER,
  FOREIGN KEY (pedido_id) REFERENCES Pedidos(pedido_id),
  FOREIGN KEY (produto_id) REFERENCES Produtos(produto_id)
);

INSERT INTO Pedidos (pedido_id, cliente_id, fornecedor_id, valor_total,
status)
VALUES (1, 1, 1, 100.00, 'Em processamento');

INSERT INTO Fornecedor (fornecedor_id, nome, status_pedido)
VALUES (1, 'Fornecedor A', 'Em andamento');

INSERT INTO Cliente (cliente_id, nome, estoque_minimo, canal_comunicacao)
VALUES (1, 'Cliente A', 10, 'E-mail');

INSERT INTO Produtos (produto_id, nome, estoque_atual)
VALUES (1, 'Produto A', 20);

CREATE OR REPLACE TRIGGER calcular_valor_total
AFTER INSERT ON Itens_pedido
FOR EACH ROW
```

```
DECLARE
  total NUMBER;
BEGIN
  SELECT SUM(Produtos.preco * Itens_pedido.quantidade) INTO total
  FROM Produtos JOIN Itens_pedido ON Produtos.produto_id =
Itens_pedido.produto_id
  WHERE Itens_pedido.pedido_id = :new.pedido_id;

  UPDATE Pedidos
  SET valor_total = total
  WHERE pedido_id = :new.pedido_id;

EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Erro: ' || SQLCODE || ' : ' || SQLERRM);
END;
/

CREATE OR REPLACE TRIGGER processar_pedido
AFTER INSERT ON Pedidos
FOR EACH ROW
BEGIN
  UPDATE Pedidos
  SET status = 'Processado'
  WHERE pedido_id = :new.pedido_id;

EXCEPTION
  WHEN OTHERS THEN
    -- Aqui você pode lidar com o erro, talvez registrando-o em algum
lugar
    DBMS_OUTPUT.PUT_LINE('Erro: ' || SQLCODE || ' : ' || SQLERRM);
END;
/


-- Criação do procedimento para reabastecimento automático de produtos
com baixo estoque
CREATE OR REPLACE PROCEDURE reabastecer_produtos AS
BEGIN
  FOR c IN (SELECT * FROM Cliente) LOOP
    FOR p IN (SELECT * FROM Produtos WHERE estoque_atual <
c.estoque_minimo) LOOP
      DBMS_OUTPUT.PUT_LINE('Baixo estoque do produto: ' || p.nome);
    END LOOP;
  END LOOP;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Nenhum dado encontrado.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Erro: ' || SQLCODE || ' : ' || SQLERRM);
END;
/

-- Criação do procedimento para enviar notificações aos clientes
CREATE OR REPLACE PROCEDURE enviar_notificacoes AS
BEGIN
  FOR c IN (SELECT * FROM Cliente) LOOP
```

```
    FOR p IN (SELECT * FROM Pedidos WHERE cliente_id = c.cliente_id) LOOP
      -- Lógica para envio de notificações
      DBMS_OUTPUT.PUT_LINE('O Status do seu pedido é: ' || p.status);
    END LOOP;
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Nenhum dado encontrado.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Erro: ' || SQLCODE || ' : ' || SQLERRM);
END;
/

CREATE OR REPLACE TRIGGER notificar_cliente
AFTER INSERT ON Pedidos
FOR EACH ROW
BEGIN
  enviar_notificacoes;
EXCEPTION
  WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('Erro: ' || SQLCODE || ' : ' || SQLERRM);
END;
/
```