Integrantes:

551694 - Diogo Fagioli Bombonatti
550711 - Gabriel Galdino da Silva
550531 - Luis Fernando Menezes Zampar
89162 - Murilo Nogueira

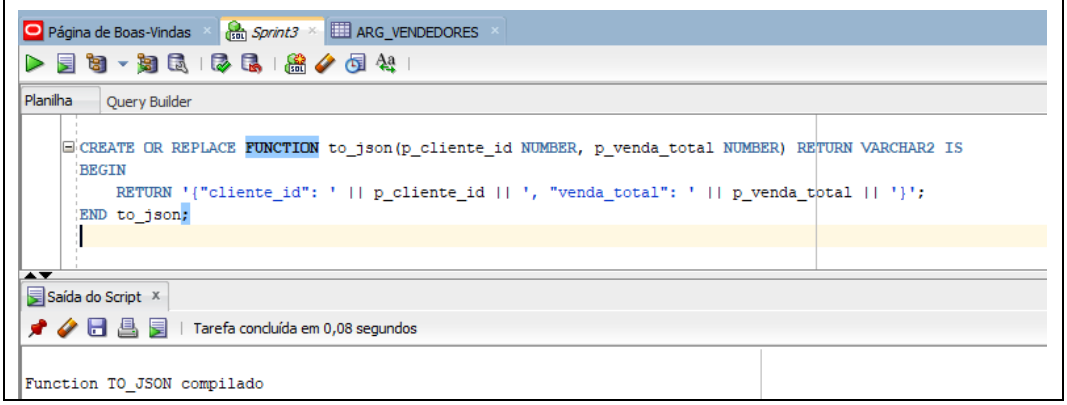
1. **Criar 02 procedimentos (30 pontos)**
   Cada procedimento deve tratar três exceções diferentes em cada procedimento:

   - **Primeiro procedimento** - Exibição de dados relacionais em formato JSON com tratamento de exceções:

     Este procedimento fará o JOIN de duas tabelas (por exemplo, arg_clientes e arg_vendas), transformará os dados em formato JSON e incluirá tratamento para três exceções: NO_DATA_FOUND, TOO_MANY_ROWS e OTHERS.

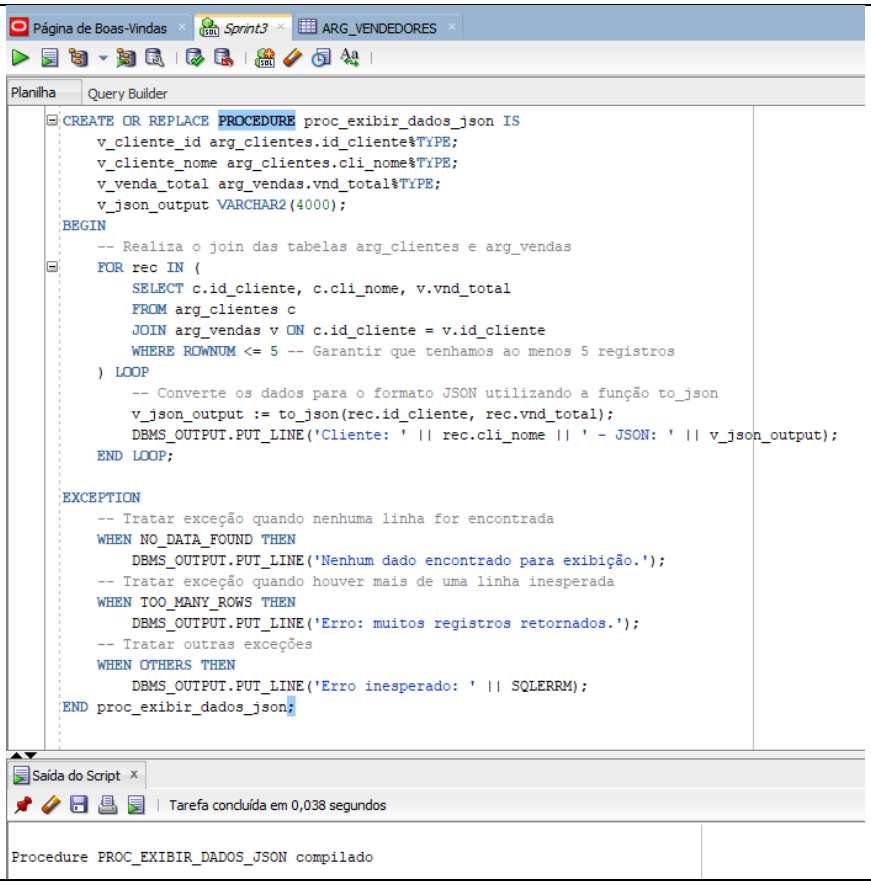
   - **Função para transformar os dados em JSON:**

```
CREATE OR REPLACE FUNCTION to_json(p_cliente_id NUMBER, p_venda_total
NUMBER) RETURN VARCHAR2 IS
BEGIN
    RETURN '{"cliente_id": ' || p_cliente_id || ', "venda_total": ' ||
p_venda_total || '}';
END to_json;
```



- **Procedimento: proc_exibir_dados_json**

```
CREATE OR REPLACE PROCEDURE proc_exibir_dados_json IS
  v_cliente_id arg_clientes.id_cliente%TYPE;
  v_cliente_nome arg_clientes.cli_nome%TYPE;
  v_venda_total arg_vendas.vnd_total%TYPE;
  v_json_output VARCHAR2(4000);
BEGIN
  -- Realiza o join das tabelas arg_clientes e arg_vendas
  FOR rec IN (
    SELECT c.id_cliente, c.cli_nome, v.vnd_total
    FROM arg_clientes c
    JOIN arg_vendas v ON c.id_cliente = v.id_cliente
    WHERE ROWNUM <= 5 -- Garantir que tenhamos ao menos 5 registros
  ) LOOP
    -- Converte os dados para o formato JSON utilizando a função to_json
    v_json_output := to_json(rec.id_cliente, rec.vnd_total);
    DBMS_OUTPUT.PUT_LINE('Cliente: ' || rec.cli_nome || ' - JSON: ' || v_json_output);
  END LOOP;

EXCEPTION
  -- Tratar exceção quando nenhuma linha for encontrada
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Nenhum dado encontrado para exibição.');
  -- Tratar exceção quando houver mais de uma linha inesperada
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Erro: muitos registros retornados.');
  -- Tratar outras exceções
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Erro inesperado: ' || SQLERRM);
END proc_exibir_dados_json;
```

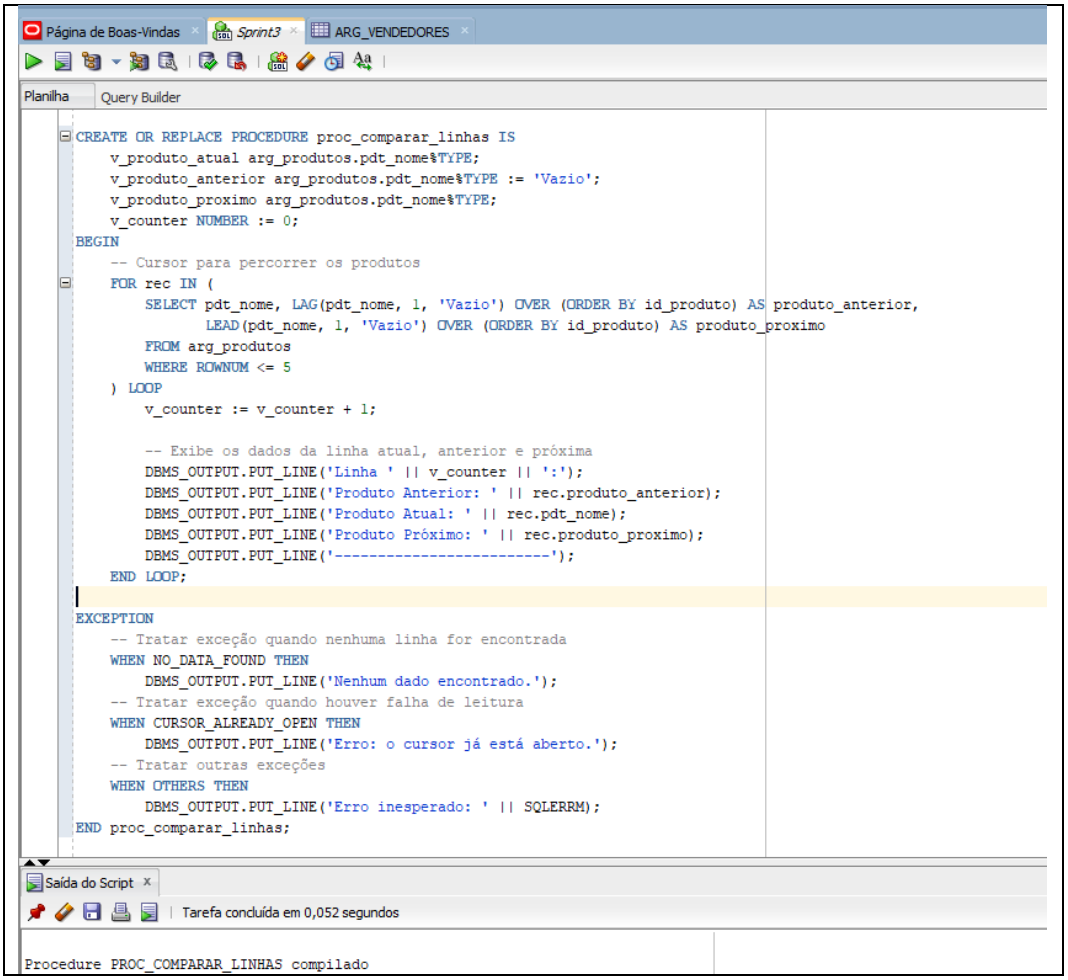


- **Segundo procedimento - Comparação de valor entre a linha anterior, atual e próxima**

  Este procedimento lerá os dados da tabela **arg_produtos**, exibindo o nome do produto atual, o anterior e o próximo. Ele tratará três exceções diferentes.

- **Procedimento: proc_comparar_linhas:**

```
CREATE OR REPLACE PROCEDURE proc_comparar_linhas IS
    v_produto_atual arg_produtos.pdt_nome%TYPE;
    v_produto_anterior arg_produtos.pdt_nome%TYPE := 'Vazio';
    v_produto_proximo arg_produtos.pdt_nome%TYPE;
    v_counter NUMBER := 0;
BEGIN
    -- Cursor para percorrer os produtos
    FOR rec IN (
        SELECT pdt_nome, LAG(pdt_nome, 1, 'Vazio') OVER (ORDER BY
id_produto) AS produto_anterior,
               LEAD(pdt_nome, 1, 'Vazio') OVER (ORDER BY id_produto)
AS produto_proximo
        FROM arg_produtos
        WHERE ROWNUM <= 5
    ) LOOP
        v_counter := v_counter + 1;

        -- Exibe os dados da linha atual, anterior e próxima
        DBMS_OUTPUT.PUT_LINE('Linha ' || v_counter || ':');
        DBMS_OUTPUT.PUT_LINE('Produto Anterior: ' ||
rec.produto_anterior);
        DBMS_OUTPUT.PUT_LINE('Produto Atual: ' || rec.pdt_nome);
        DBMS_OUTPUT.PUT_LINE('Produto Próximo: ' ||
rec.produto_proximo);
        DBMS_OUTPUT.PUT_LINE('-------------------------');
    END LOOP;

EXCEPTION
     -- Tratar exceção quando nenhuma linha for encontrada
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nenhum dado encontrado.');
    -- Tratar exceção quando houver falha de leitura
    WHEN CURSOR_ALREADY_OPEN THEN
        DBMS_OUTPUT.PUT_LINE('Erro: o cursor já está aberto.');
    -- Tratar outras exceções
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erro inesperado: ' || SQLERRM);
END proc_comparar_linhas;
```



- **Explicação das exceções tratadas:**
    - NO_DATA_FOUND: Tratada quando a consulta não retorna nenhum dado.
    - TOO_MANY_ROWS: Tratada quando uma consulta que deveria retornar uma única linha retorna várias.
    - OTHERS: Tratada para qualquer outra exceção não identificada especificamente.

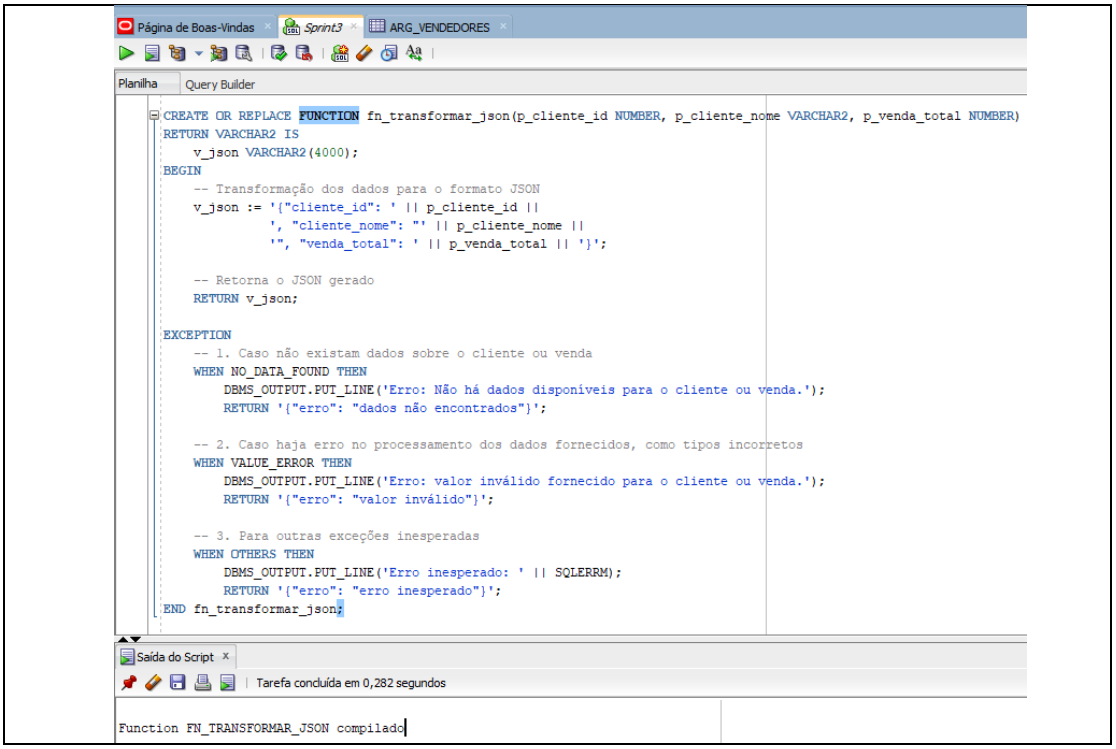
2. **Desenvolver duas funções (30 pontos):**

- **Função para Transformar Dados em Formato JSON**

  No projeto ArgosIA, a transformação de dados para JSON é uma operação central ao lidar com a recomendação de produtos e perfis de clientes. As exceções tratadas devem refletir possíveis erros que podem ocorrer ao gerar ou exibir recomendações.

- **Função: fn_transformar_json:**

```
CREATE OR REPLACE FUNCTION fn_transformar_json(p_cliente_id NUMBER,
p_cliente_nome VARCHAR2, p_venda_total NUMBER)
RETURN VARCHAR2 IS
  v_json VARCHAR2(4000);
BEGIN
  -- Transformação dos dados para o formato JSON
  v_json := '{"cliente_id": ' || p_cliente_id ||
        ', "cliente_nome": "' || p_cliente_nome ||
        '", "venda_total": ' || p_venda_total || '}';

  -- Retorna o JSON gerado
  RETURN v_json;

EXCEPTION
  -- 1. Caso não existam dados sobre o cliente ou venda
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Erro: Não há dados disponíveis para o cliente ou
venda.');
    RETURN '{"erro": "dados não encontrados"}';

  -- 2. Caso haja erro no processamento dos dados fornecidos, como tipos incorretos
  WHEN VALUE_ERROR THEN
    DBMS_OUTPUT.PUT_LINE('Erro: valor inválido fornecido para o cliente ou
venda.');
    RETURN '{"erro": "valor inválido"}';

  -- 3. Para outras exceções inesperadas
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Erro inesperado: ' || SQLERRM);
    RETURN '{"erro": "erro inesperado"}';
END fn_transformar_json;
```



- **Exceções tratadas:**
  - **NO_DATA_FOUND:** Caso nenhum dado seja encontrado para o cliente ou venda, uma situação que pode ocorrer se tentarmos gerar recomendações para um cliente sem histórico.

  - **VALUE_ERROR:** Caso ocorra um erro no tipo de dados fornecidos, por exemplo, fornecendo um nome numérico onde se espera um texto.

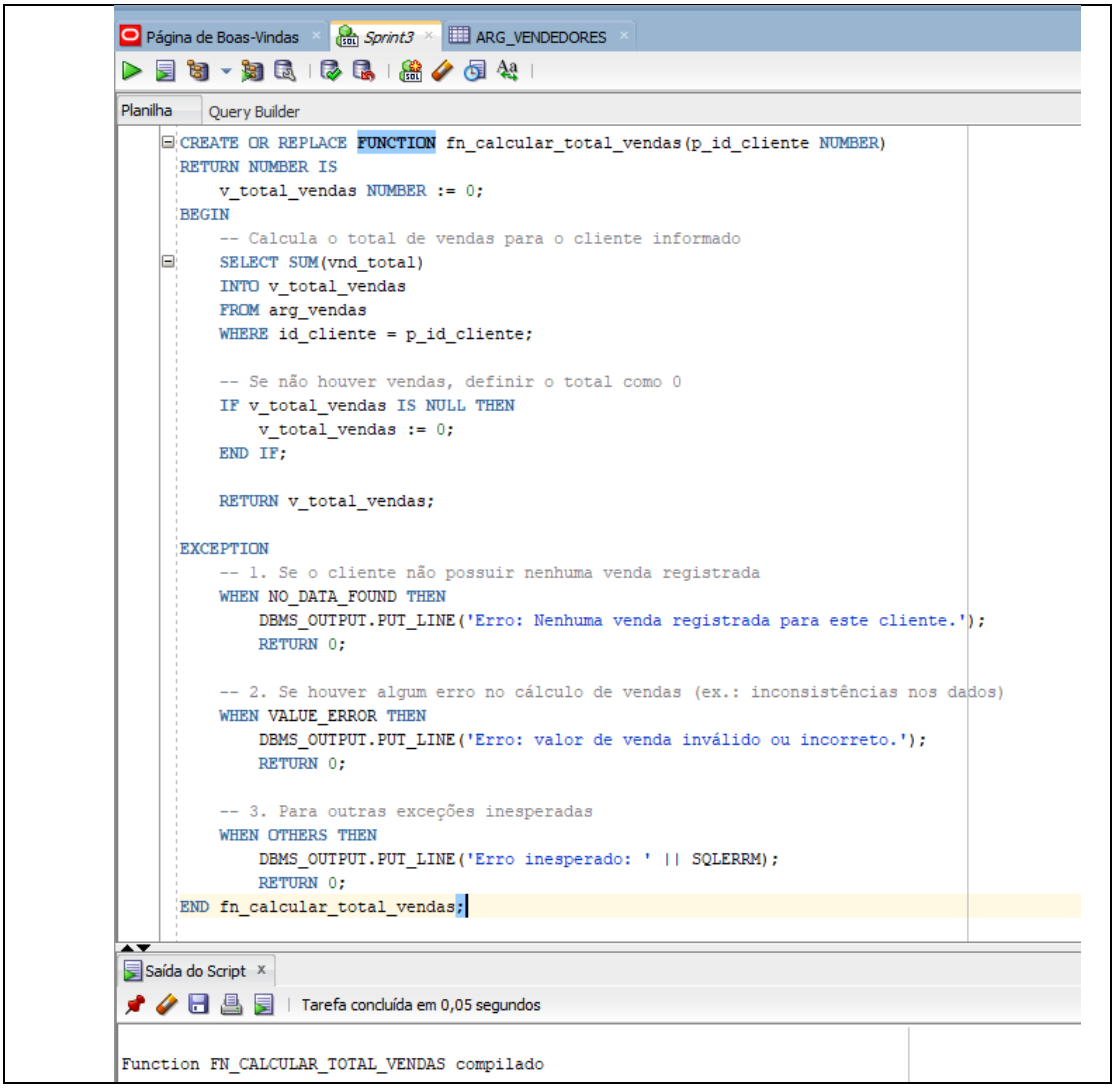
  - **OTHERS:** Tratamento genérico para erros inesperados, como problemas no sistema ou falha de banco de dados.

- **Função para Calcular o Total de Vendas:**

  No contexto de ArgosIA, o cálculo do total de vendas é importante para identificar o comportamento de compra de um cliente e seus remetentes. Vamos ajustar a função para refletir o uso prático no sistema e tratar exceções relacionadas ao comportamento de compra e suas peculiaridades.

- **Função: fn_calcular_total_vendas:**

```
CREATE OR REPLACE FUNCTION fn_calcular_total_vendas(p_id_cliente NUMBER)
RETURN NUMBER IS
  v_total_vendas NUMBER := 0;
BEGIN
  -- Calcula o total de vendas para o cliente informado
  SELECT SUM(vnd_total)
  INTO v_total_vendas
  FROM arg_vendas
  WHERE id_cliente = p_id_cliente;

  -- Se não houver vendas, definir o total como 0
  IF v_total_vendas IS NULL THEN
    v_total_vendas := 0;
  END IF;

  RETURN v_total_vendas;

EXCEPTION
  -- 1. Se o cliente não possuir nenhuma venda registrada
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Erro: Nenhuma venda registrada para este cliente.');
    RETURN 0;

  -- 2. Se houver algum erro no cálculo de vendas (ex.: inconsistências nos dados)
  WHEN VALUE_ERROR THEN
    DBMS_OUTPUT.PUT_LINE('Erro: valor de venda inválido ou incorreto.');
    RETURN 0;

  -- 3. Para outras exceções inesperadas
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Erro inesperado: ' || SQLERRM);
    RETURN 0;
END fn_calcular_total_vendas;
```



Exceções tratadas:

- **NO_DATA_FOUND:** Caso o cliente não tenha vendas registradas, uma situação comum para novos clientes ou usuários com pouca atividade.

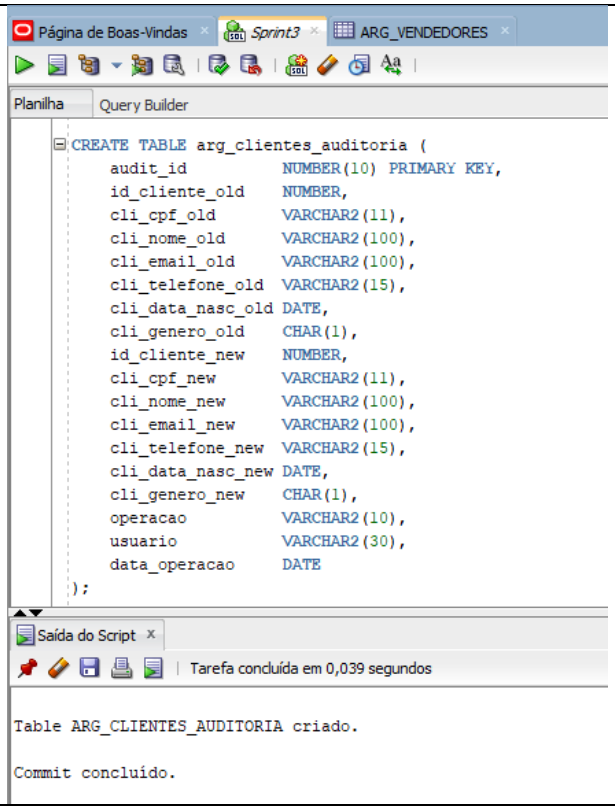
- **VALUE_ERROR:** Tratamento para erros no cálculo dos valores de vendas, que podem ocorrer devido a dados inconsistentes ou corrompidos.

- **OTHERS:** Tratamento genérico para qualquer outra falha inesperada, como erros de sistema ou falhas de banco de dados.

3. Gatilho:

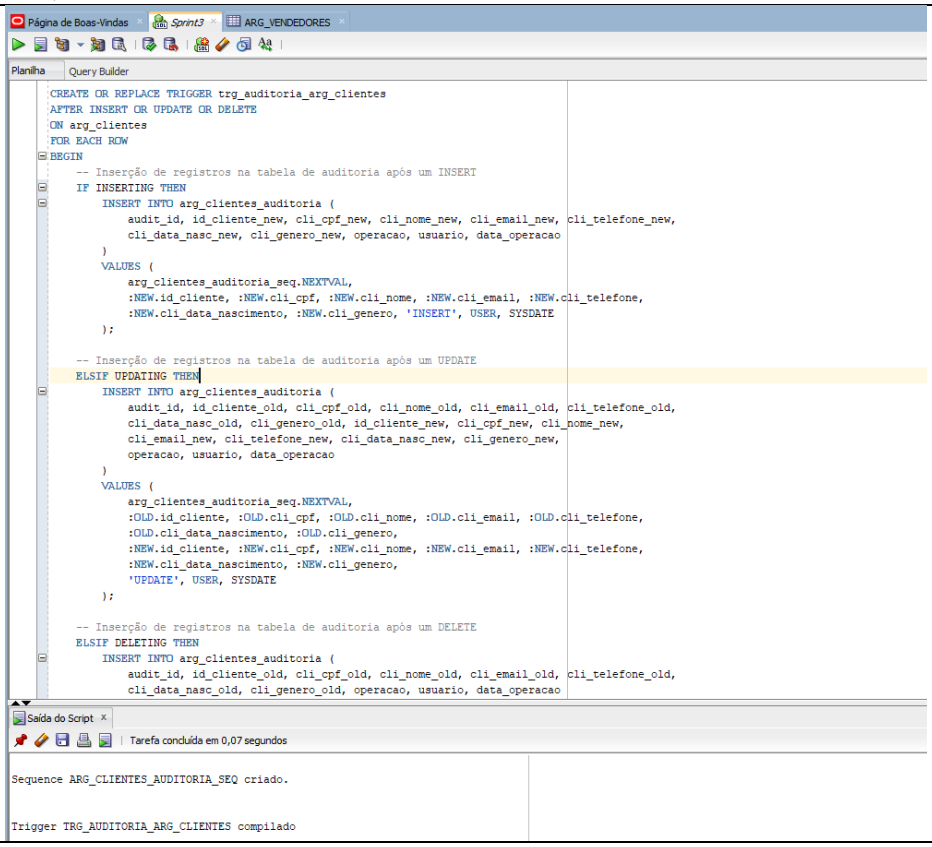
- **Criação da Tabela de Auditoria:**
  Essa tabela armazenará as informações necessárias para auditoria, incluindo os valores antigos (OLD), novos (NEW), o nome do usuário, a operação realizada e a data da operação.

```
CREATE TABLE arg_clientes_auditoria (
  audit_id        NUMBER(10) PRIMARY KEY,
  id_cliente_old   NUMBER,
  cli_cpf_old     VARCHAR2(11),
  cli_nome_old     VARCHAR2(100),
  cli_email_old    VARCHAR2(100),
  cli_telefone_old  VARCHAR2(15),
  cli_data_nasc_old DATE,
  cli_genero_old   CHAR(1),
  id_cliente_new   NUMBER,
  cli_cpf_new     VARCHAR2(11),
  cli_nome_new     VARCHAR2(100),
  cli_email_new    VARCHAR2(100),
  cli_telefone_new  VARCHAR2(15),
  cli_data_nasc_new DATE,
  cli_genero_new   CHAR(1),
  operacao        VARCHAR2(10),
  usuario         VARCHAR2(30),
  data_operacao    DATE
);
```



- **Criação de uma sequência para audit_id:**

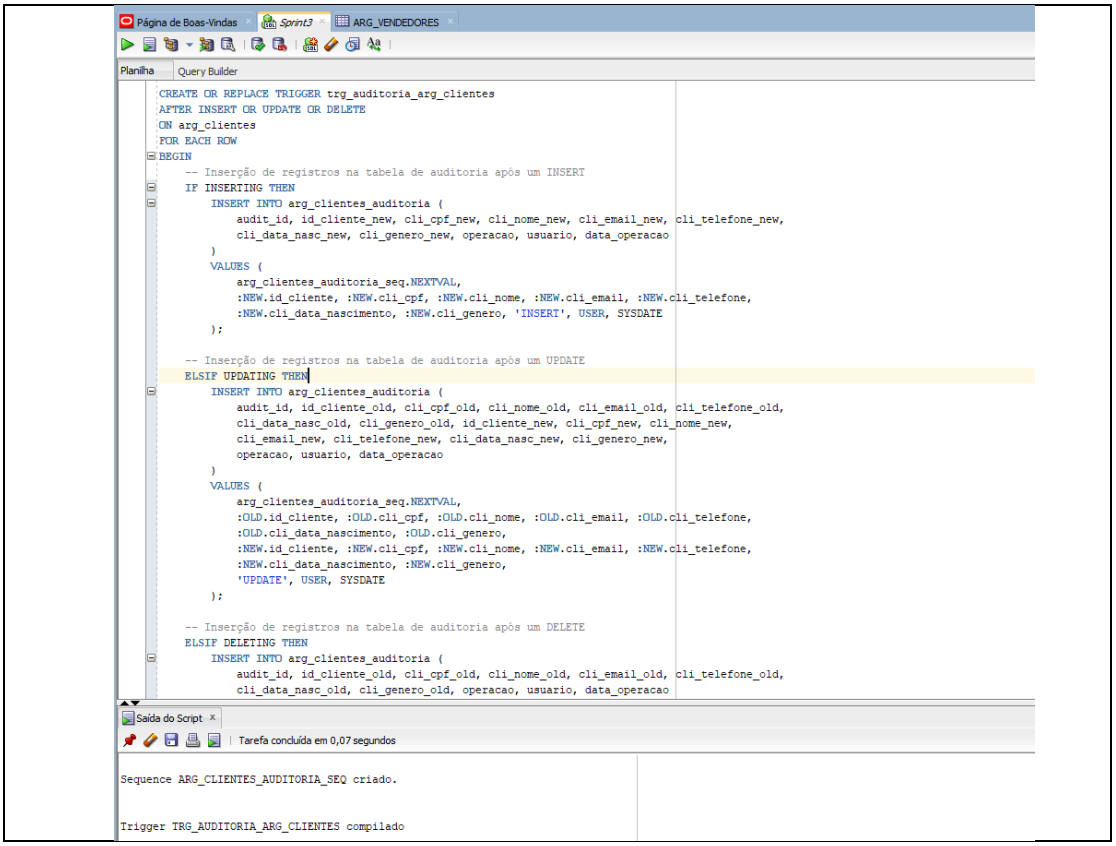
```
CREATE SEQUENCE arg_clientes_auditoria_seq
START WITH 1
INCREMENT BY 1
NOCACHE;
```



- **Gatilho de Auditoria:**
  *O gatilho será disparado para **INSERT**, **UPDATE**, e **DELETE** na tabela **arg_clientes**. Ele vai capturar as informações anteriores e novas, assim como o tipo de operação realizada e o nome do usuário que fez a alteração, utilizando a função USER para identificar o responsável pela operação.*

```
CREATE OR REPLACE TRIGGER trg_auditoria_arg_clientes
AFTER INSERT OR UPDATE OR DELETE
ON arg_clientes
FOR EACH ROW
BEGIN
  -- Inserção de registros na tabela de auditoria após um INSERT
  IF INSERTING THEN
    INSERT INTO arg_clientes_auditoria (
      audit_id, id_cliente_new, cli_cpf_new, cli_nome_new, cli_email_new,
cli_telefone_new,
      cli_data_nasc_new, cli_genero_new, operacao, usuario, data_operacao
    )
    VALUES (
      arg_clientes_auditoria_seq.NEXTVAL,
      :NEW.id_cliente, :NEW.cli_cpf, :NEW.cli_nome, :NEW.cli_email,
:NEW.cli_telefone,
```

```
        :NEW.cli_data_nascimento, :NEW.cli_genero, 'INSERT', USER, SYSDATE
      );

  -- Inserção de registros na tabela de auditoria após um UPDATE
  ELSIF UPDATING THEN
    INSERT INTO arg_clientes_auditoria (
      audit_id, id_cliente_old, cli_cpf_old, cli_nome_old, cli_email_old,
cli_telefone_old,
      cli_data_nasc_old, cli_genero_old, id_cliente_new, cli_cpf_new, cli_nome_new,
      cli_email_new, cli_telefone_new, cli_data_nasc_new, cli_genero_new,
      operacao, usuario, data_operacao
    )
    VALUES (
      arg_clientes_auditoria_seq.NEXTVAL,
      :OLD.id_cliente, :OLD.cli_cpf, :OLD.cli_nome, :OLD.cli_email,
:OLD.cli_telefone,
      :OLD.cli_data_nascimento, :OLD.cli_genero,
      :NEW.id_cliente, :NEW.cli_cpf, :NEW.cli_nome, :NEW.cli_email,
:NEW.cli_telefone,
      :NEW.cli_data_nascimento, :NEW.cli_genero,
      'UPDATE', USER, SYSDATE
    );

  -- Inserção de registros na tabela de auditoria após um DELETE
  ELSIF DELETING THEN
    INSERT INTO arg_clientes_auditoria (
      audit_id, id_cliente_old, cli_cpf_old, cli_nome_old, cli_email_old,
cli_telefone_old,
      cli_data_nasc_old, cli_genero_old, operacao, usuario, data_operacao
    )
    VALUES (
      arg_clientes_auditoria_seq.NEXTVAL,
      :OLD.id_cliente, :OLD.cli_cpf, :OLD.cli_nome, :OLD.cli_email,
:OLD.cli_telefone,
      :OLD.cli_data_nascimento, :OLD.cli_genero,
      'DELETE', USER, SYSDATE
    );
  END IF;
END trg_auditoria_arg_clientes;
```



- **Explicação do Gatilho:**

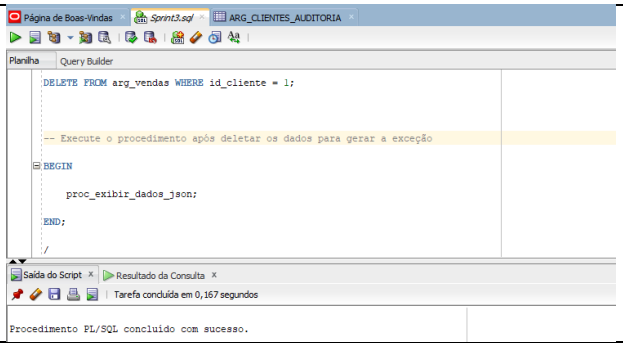
  - **INSERT:** Quando um novo registro é inserido na tabela arg_clientes, o gatilho armazena as informações do novo cliente na tabela de auditoria, registrando a operação como "INSERT".

  - **UPDATE:** Quando ocorre uma atualização na tabela arg_clientes, o gatilho salva tanto os dados antigos quanto os novos na tabela de auditoria, registrando a operação como "UPDATE".

  - **DELETE:** Quando um registro é excluído, o gatilho armazena os dados antigos do cliente na tabela de auditoria, registrando a operação como "DELETE".

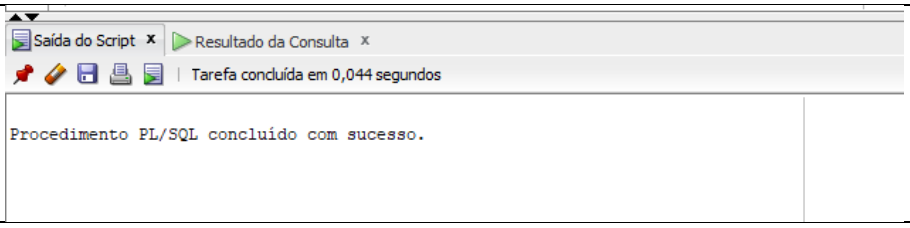
- **Comprovações dos Procedimentos:**
- **Procedimento 1: proc_exibir_dados_json:**

```
-- Execute o procedimento com os dados normais
BEGIN
  proc_exibir_dados_json;
END;
/

-- Para testar exceção "NO_DATA_FOUND", insira a seguinte condição no
procedimento ou altere a consulta para não retornar dados
DELETE FROM arg_vendas WHERE id_cliente = 1;

-- Execute o procedimento após deletar os dados para gerar a exceção
BEGIN
  proc_exibir_dados_json;
END;
/
```



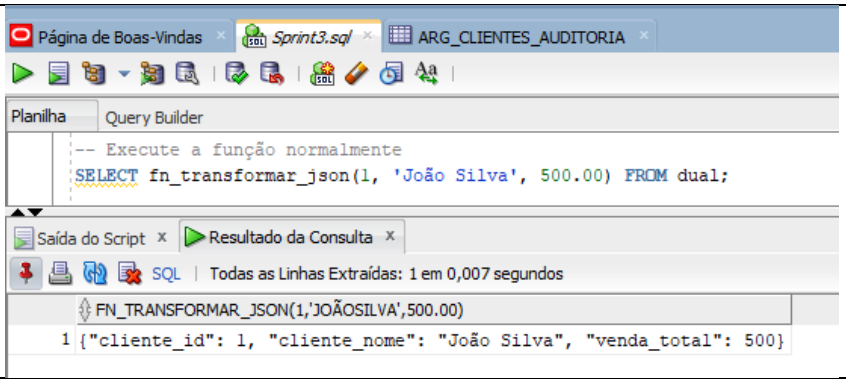
- **Procedimento 2: proc_comparar_linhas:**

```
-- Execute o procedimento normalmente
BEGIN
  proc_comparar_linhas;
END;
/

-- Para testar exceção "NO_DATA_FOUND", limpe temporariamente a tabela
DELETE FROM arg_produtos WHERE id_produto < 100;

-- Execute o procedimento para gerar a exceção "NO_DATA_FOUND"
BEGIN
  proc_comparar_linhas;
END;
/
```



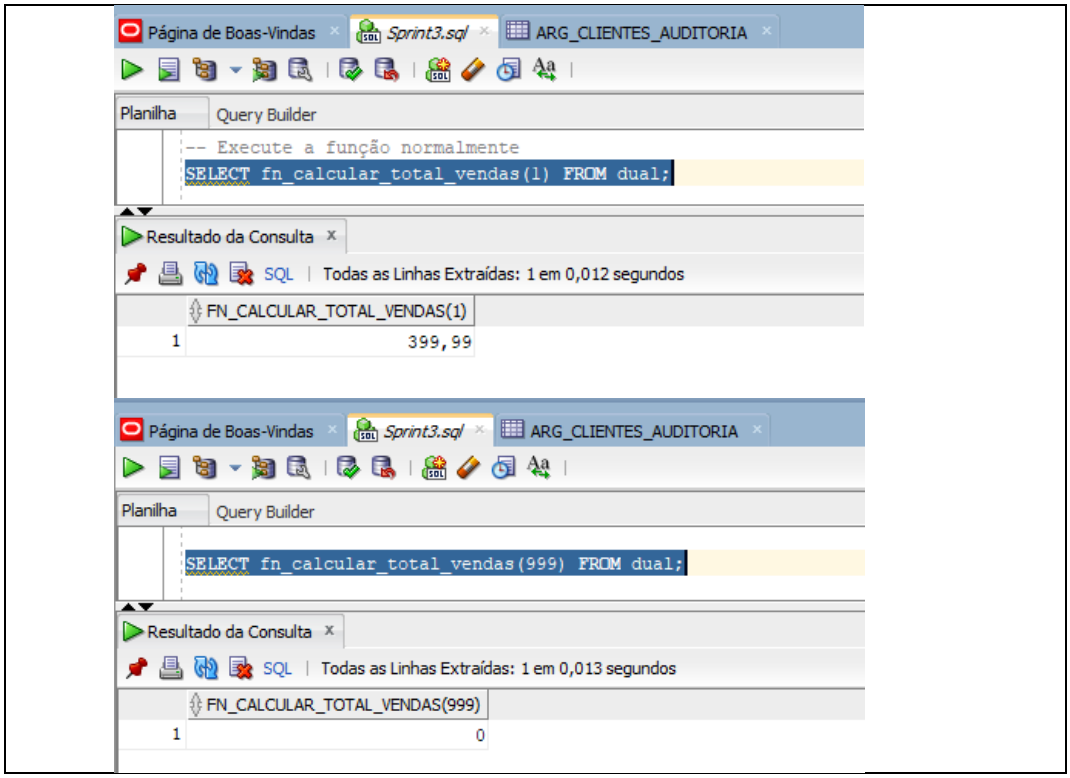
- **Função 1: fn_transformar_json:**

```
SELECT fn_transformar_json(1, 'João Silva', 500.00) FROM dual;
```



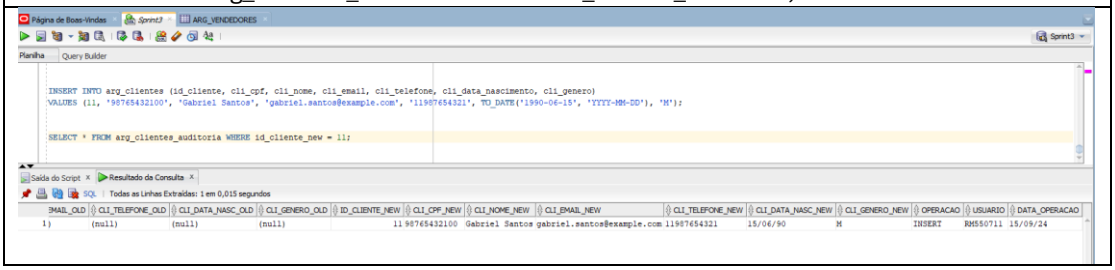
- **Função 2: fn_calcular_total_vendas:**

```
SELECT fn_calcular_total_vendas(1) FROM dual;
SELECT fn_calcular_total_vendas(999) FROM dual;
```



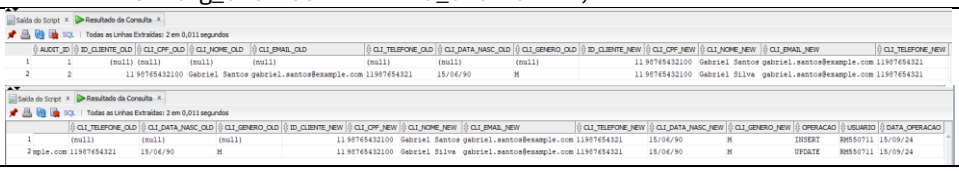
- **Comprovação do Gatilho rodando (Insert):**

```
INSERT INTO arg_clientes (id_cliente, cli_cpf, cli_nome, cli_email, cli_telefone,
cli_data_nascimento, cli_genero)
VALUES (11, '98765432100', 'Gabriel Santos', 'gabriel.santos@example.com', '11987654321',
TO_DATE('1990-06-15', 'YYYY-MM-DD'), 'M');


SELECT * FROM arg_clientes_auditoria WHERE id_cliente_new = 11;
```



- **Comprovação do Gatilho rodando (**UPDATE e DELETE**):**

```
UPDATE arg_clientes
SET cli_nome = 'Gabriel Silva'
WHERE id_cliente = 11;

-- Executa um DELETE para testar o gatilho
DELETE FROM arg_clientes WHERE id_cliente = 11;
```



Saída do Script | Resultado da Consulta

Todas as Linhas Extraídas: 2 em 0,011 segundos

AUDIT_ID	ID_CLIENTE_OLD	CLI_CPF_OLD	CLI_NOME_OLD	CLI_EMAIL_OLD	CLI_TELEFONE_OLD	CLI_DATA_NASC_OLD	CLI_GENERO_OLD	ID_CLIENTE_NEW	CLI_CPF_NEW	CLI_NOME_NEW	CLI_EMAIL_NEW	CLI_TELEFONE_NEW
1	1	(null)	(null)	(null)	(null)	(null)	(null)	11	98765432100	Gabriel Santos	gabriel.santos@example.com	11987654321
2	2	11	98765432100	Gabriel Santos	gabriel.santos@example.com	11987654321	15/06/90	M	11	98765432100	Gabriel Silva	gabriel.santos@example.com	11987654321

Saída do Script | Resultado da Consulta

Todas as Linhas Extraídas: 2 em 0,011 segundos

	CLI_TELEFONE_OLD	CLI_DATA_NASC_OLD	CLI_GENERO_OLD	ID_CLIENTE_NEW	CLI_CPF_NEW	CLI_NOME_NEW	CLI_EMAIL_NEW	CLI_TELEFONE_NEW	CLI_DATA_NASC_NEW	CLI_GENERO_NEW	OPERACAO	USUARIO	DATA_OPERACAO
1	(null)	(null)	(null)	11	98765432100	Gabriel Santos	gabriel.santos@example.com	11987654321	15/06/90	M	INSERT	RM550711	15/09/24
2	mple.com 11987654321	15/06/90	M	11	98765432100	Gabriel Silva	gabriel.santos@example.com	11987654321	15/06/90	M	UPDATE	RM550711	15/09/24