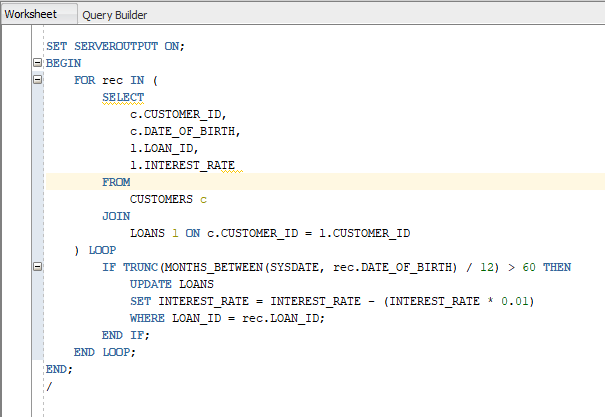
**Exercise 1: Control Structures**

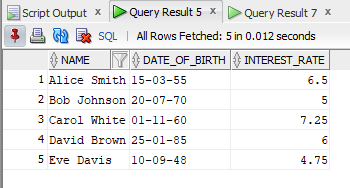
**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* + **Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

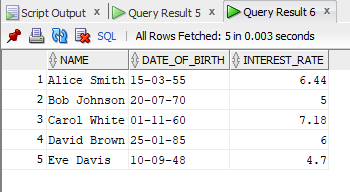


Output

Before

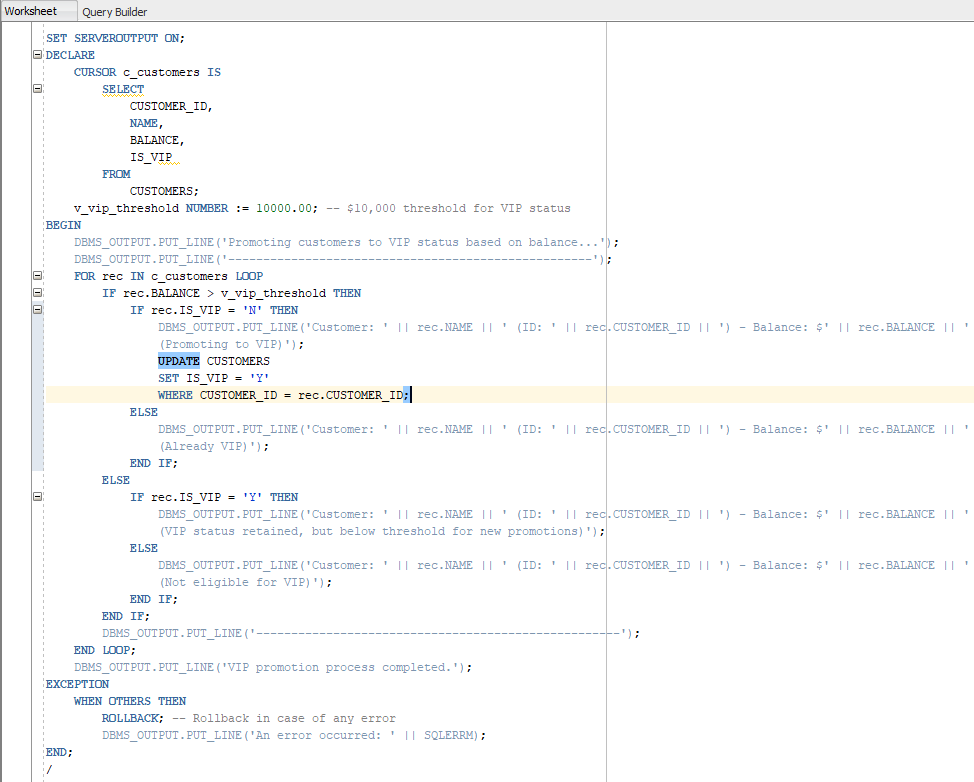


After



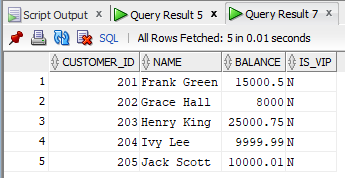
**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* + **Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

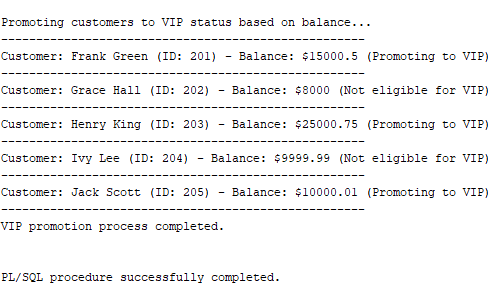


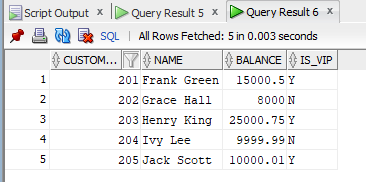
Output

Before



After



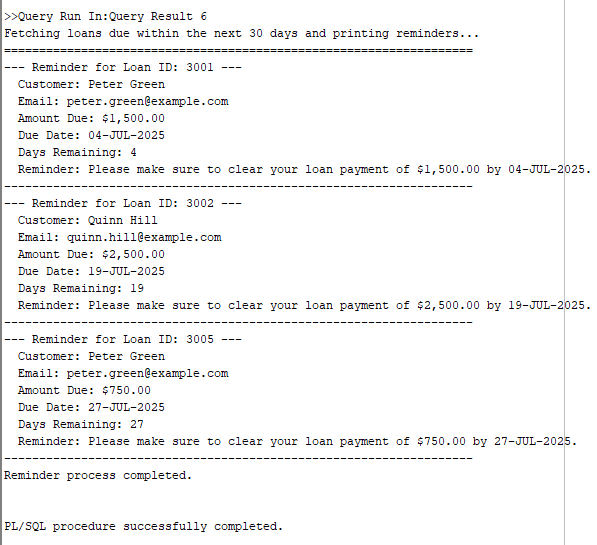


**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* + **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.



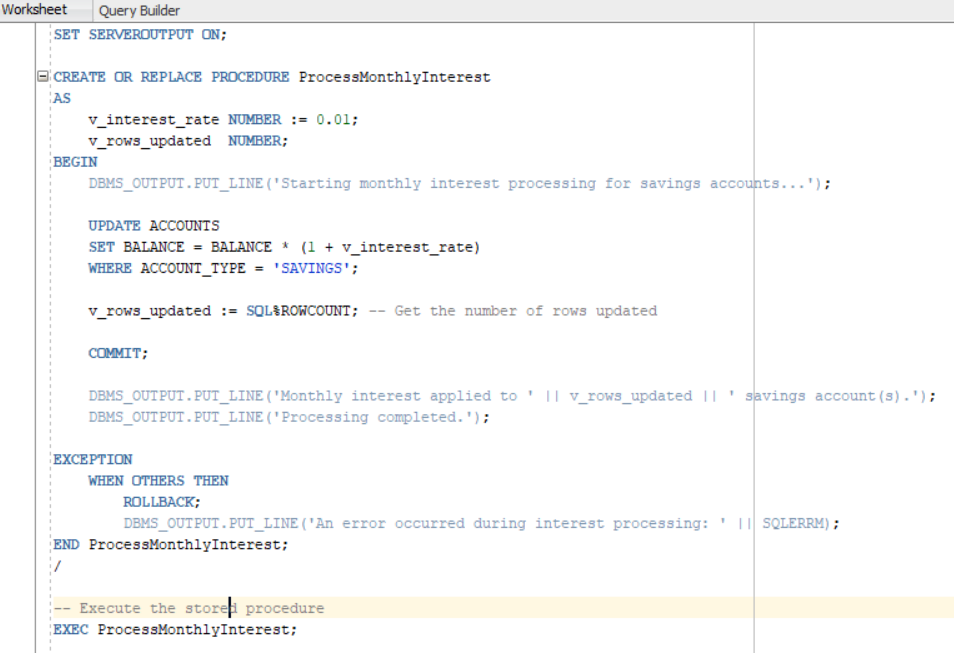
Output



**Exercise 3: Stored Procedures**

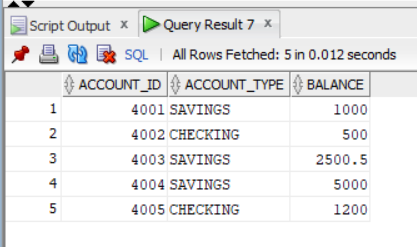
**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

* + **Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

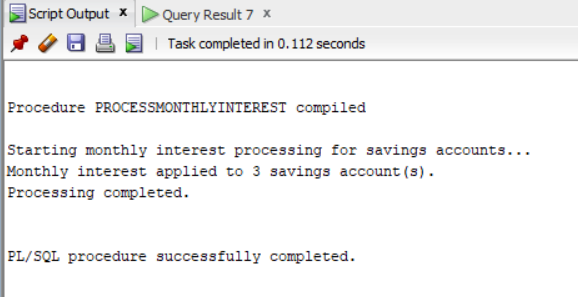


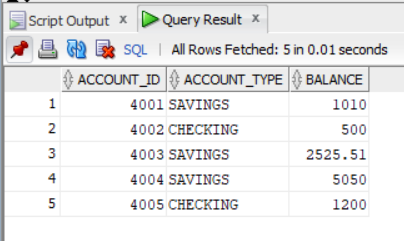
Output

Before



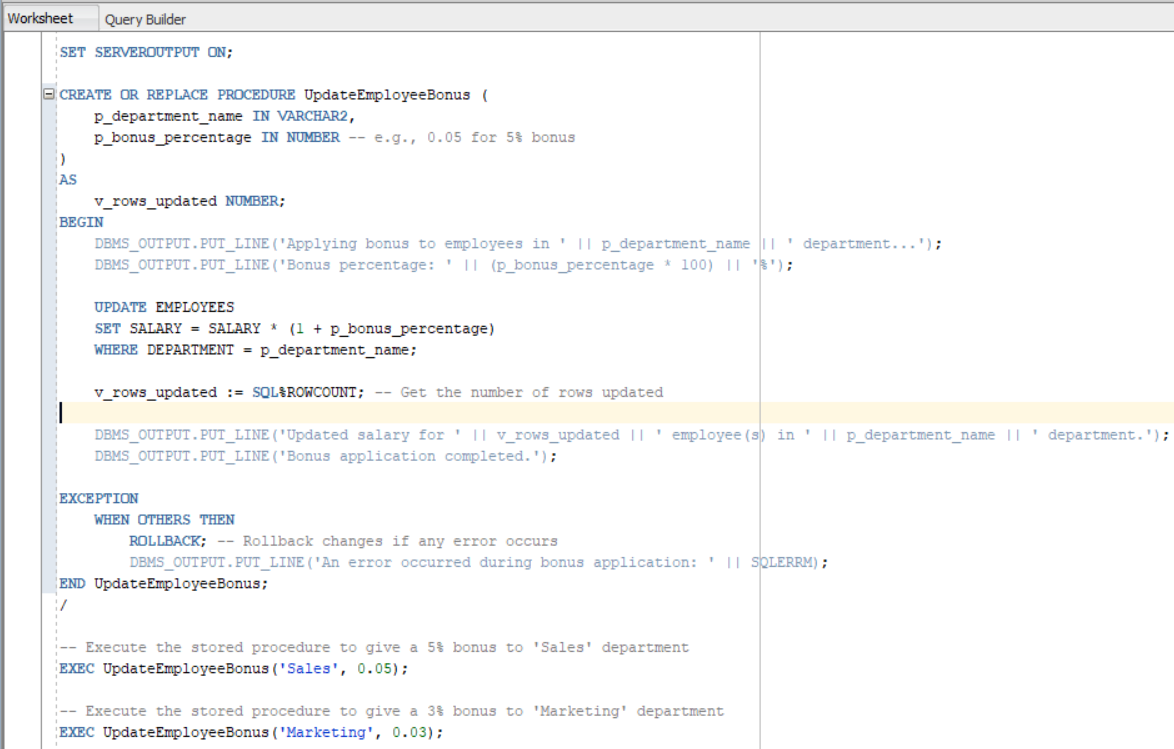
After





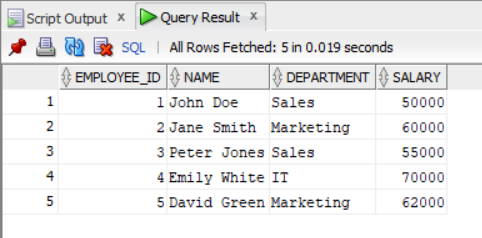
**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

* + **Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

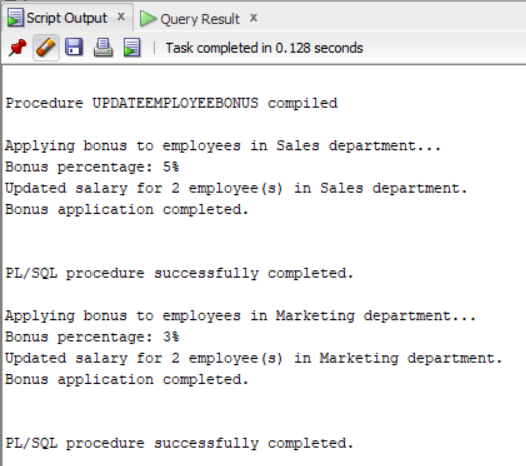


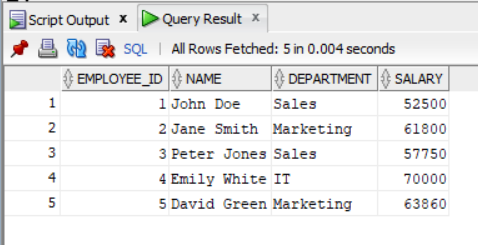
Output

Before



After





**Scenario 3:** Customers should be able to transfer funds between their accounts.

* + **Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

Code

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_source\_account\_number IN VARCHAR2,

p\_destination\_account\_number IN VARCHAR2,

p\_amount IN NUMBER

)

AS

v\_source\_balance NUMBER;

e\_insufficient\_funds EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_insufficient\_funds, -20001); -- Custom error number

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Attempting to transfer $' || p\_amount || ' from ' || p\_source\_account\_number || ' to ' || p\_destination\_account\_number || '...');

-- 1. Check if source and destination accounts exist

-- (Adding this check for robustness, though not explicitly asked)

DECLARE

v\_source\_exists NUMBER;

v\_dest\_exists NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_source\_exists FROM BANK\_ACCOUNTS WHERE ACCOUNT\_NUMBER = p\_source\_account\_number;

SELECT COUNT(\*) INTO v\_dest\_exists FROM BANK\_ACCOUNTS WHERE ACCOUNT\_NUMBER = p\_destination\_account\_number;

IF v\_source\_exists = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Source account ' || p\_source\_account\_number || ' does not exist.');

END IF;

IF v\_dest\_exists = 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Destination account ' || p\_destination\_account\_number || ' does not exist.');

END IF;

END;

-- 2. Get current balance of source account

SELECT BALANCE

INTO v\_source\_balance

FROM BANK\_ACCOUNTS

WHERE ACCOUNT\_NUMBER = p\_source\_account\_number

FOR UPDATE OF BALANCE; -- Lock the row to prevent concurrent updates

-- 3. Check for sufficient balance

IF v\_source\_balance < p\_amount THEN

RAISE e\_insufficient\_funds; -- Raise custom exception

END IF;

-- 4. Deduct from source account

UPDATE BANK\_ACCOUNTS

SET BALANCE = BALANCE - p\_amount

WHERE ACCOUNT\_NUMBER = p\_source\_account\_number;

-- 5. Add to destination account

UPDATE BANK\_ACCOUNTS

SET BALANCE = BALANCE + p\_amount

WHERE ACCOUNT\_NUMBER = p\_destination\_account\_number;

COMMIT; -- Commit the transaction

DBMS\_OUTPUT.PUT\_LINE('Transfer of $' || p\_amount || ' completed successfully.');

EXCEPTION

WHEN e\_insufficient\_funds THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient funds in account ' || p\_source\_account\_number || '. Current balance: $' || v\_source\_balance || '. Amount requested: $' || p\_amount || '.');

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: One of the accounts was not found. (This should be caught by explicit checks above now).');

WHEN OTHERS THEN

ROLLBACK; -- Rollback all changes in case of any other error

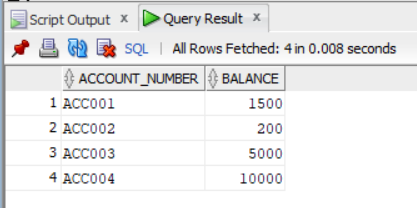
DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred during transfer: ' || SQLERRM);

END TransferFunds;

/

Output

Before



After

