**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.

**CODE**

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE customer\_data';

EXCEPTION

  WHEN OTHERS THEN

    NULL;

END;

/

CREATE TABLE customer\_data (

    customer\_id NUMBER PRIMARY KEY,

    name VARCHAR2(100),

    age NUMBER,

    loan\_interest\_rate NUMBER(5,2)

);

INSERT INTO customer\_data VALUES (1, 'Arun', 65, 9.5);

INSERT INTO customer\_data VALUES (2, 'Sneha', 45, 10.0);

INSERT INTO customer\_data VALUES (3, 'Vikram', 70, 8.5);

COMMIT;

BEGIN

    FOR customer\_rec IN (

        SELECT customer\_id

        FROM customer\_data

        WHERE age > 60

    ) LOOP

        UPDATE customer\_data

        SET loan\_interest\_rate = loan\_interest\_rate - 1

        WHERE customer\_id = customer\_rec.customer\_id;

    END LOOP;

    COMMIT;

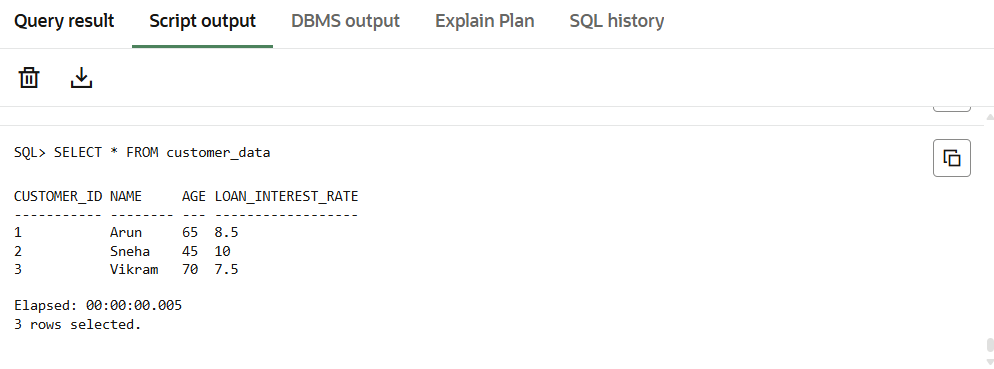
    DBMS\_OUTPUT.PUT\_LINE('Discount applied to senior citizens.');

END;

/

SELECT \* FROM customer\_data;

**OUTPUT**

****

**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.

**CODE**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE customer\_accounts';

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

/

CREATE TABLE customer\_accounts (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

balance NUMBER(12,2),

is\_vip VARCHAR2(5) DEFAULT 'FALSE'

);

INSERT INTO customer\_accounts VALUES (1, 'Arun', 9500, 'FALSE');

INSERT INTO customer\_accounts VALUES (2, 'Sneha', 12000, 'FALSE');

INSERT INTO customer\_accounts VALUES (3, 'Vikram', 15000, 'FALSE');

INSERT INTO customer\_accounts VALUES (4, 'Sara', 8000, 'FALSE');

INSERT INTO customer\_accounts VALUES (5, 'Ravi', 10500, 'FALSE');

COMMIT;

BEGIN

FOR customer\_rec IN (

SELECT customer\_id, balance

FROM customer\_accounts

) LOOP

IF customer\_rec.balance > 10000 THEN

UPDATE customer\_accounts

SET is\_vip = 'TRUE'

WHERE customer\_id = customer\_rec.customer\_id;

END IF;

END LOOP;

COMMIT;

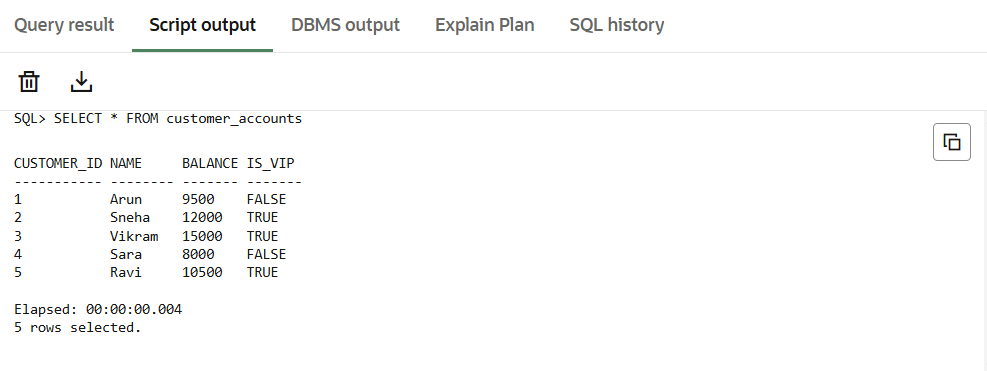
DBMS\_OUTPUT.PUT\_LINE('VIP status updated based on balance.');

END;

/

SELECT \* FROM customer\_accounts;

**OUTPUT**



**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.

**CODE**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE loan\_accounts';

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

/

CREATE TABLE loan\_accounts (

loan\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

due\_date DATE,

amount NUMBER(10,2)

);

INSERT INTO loan\_accounts VALUES (1, 'Arun', SYSDATE + 10, 5000);

INSERT INTO loan\_accounts VALUES (2, 'Sneha', SYSDATE + 5, 15000);

INSERT INTO loan\_accounts VALUES (3, 'Vikram', SYSDATE + 40, 20000);

INSERT INTO loan\_accounts VALUES (4, 'Sara', SYSDATE + 25, 8000);

INSERT INTO loan\_accounts VALUES (5, 'Ravi', SYSDATE - 5, 9000);

COMMIT;

BEGIN

FOR loan\_rec IN (

SELECT loan\_id, customer\_name, due\_date, amount

FROM loan\_accounts

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Loan ID ' || loan\_rec.loan\_id ||

' for ' || loan\_rec.customer\_name ||

' of amount $' || loan\_rec.amount ||

' is due on ' || TO\_CHAR(loan\_rec.due\_date, 'DD-Mon-YYYY')

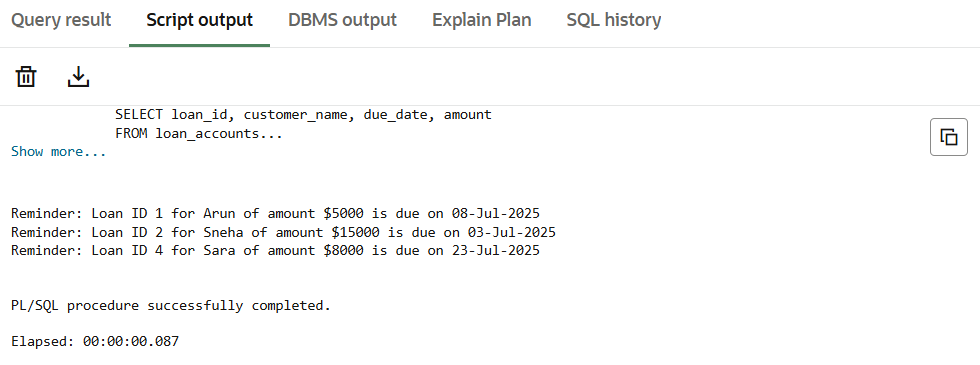
);

END LOOP;

END;

/

**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.

**CODE**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE savings\_accounts';

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

/

CREATE TABLE savings\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

balance NUMBER(12,2)

);

INSERT INTO savings\_accounts VALUES (1, 'Arun', 10000);

INSERT INTO savings\_accounts VALUES (2, 'Sneha', 15000);

INSERT INTO savings\_accounts VALUES (3, 'Vikram', 8000);

INSERT INTO savings\_accounts VALUES (4, 'Sara', 5000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc\_rec IN (

SELECT account\_id, balance FROM savings\_accounts

) LOOP

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_id = acc\_rec.account\_id;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest processed for all savings accounts.');

END;

/

BEGIN

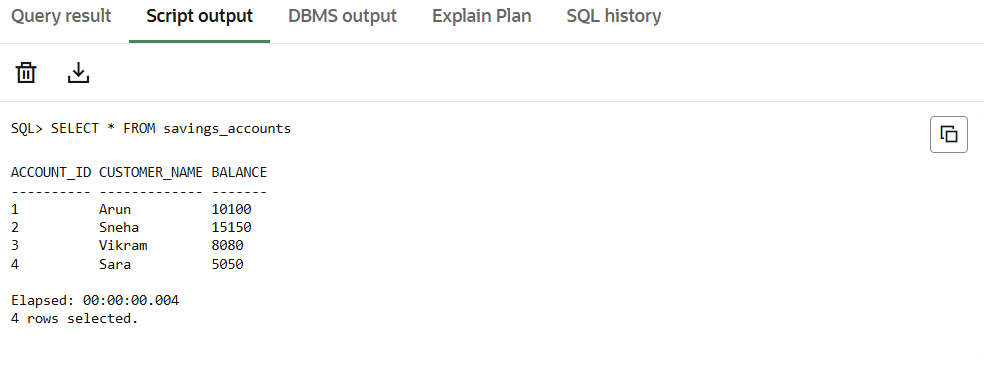
ProcessMonthlyInterest;

END;

/

SELECT \* FROM savings\_accounts;

**OUTPUT**

****

**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.

**CODE**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE employees';

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

/

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER(10,2)

);

INSERT INTO employees VALUES (1, 'Arun', 'IT', 50000);

INSERT INTO employees VALUES (2, 'Sneha', 'HR', 45000);

INSERT INTO employees VALUES (3, 'Vikram', 'IT', 55000);

INSERT INTO employees VALUES (4, 'Sara', 'Finance', 60000);

INSERT INTO employees VALUES (5, 'Ravi', 'HR', 47000);

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE employees

SET salary = salary + (salary \* bonus\_percent / 100)

WHERE department = dept\_name;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department: ' || dept\_name);

END;

/

BEGIN

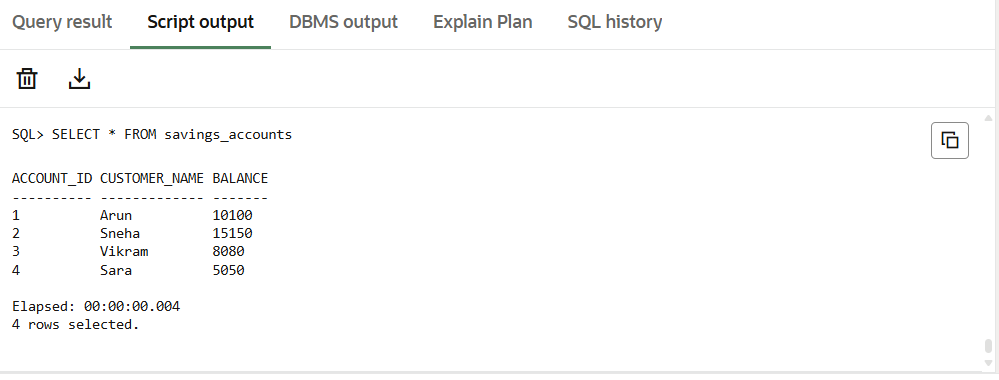
UpdateEmployeeBonus('HR', 10);

END;

/

SELECT \* FROM employees;

**OUTPUT**

****

**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**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE bank\_accounts';

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

/

CREATE TABLE bank\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

balance NUMBER(12,2)

);

INSERT INTO bank\_accounts VALUES (1, 'Arun', 10000);

INSERT INTO bank\_accounts VALUES (2, 'Sneha', 5000);

INSERT INTO bank\_accounts VALUES (3, 'Vikram', 2000);

COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_acc IN NUMBER,

to\_acc IN NUMBER,

amount IN NUMBER

) IS

from\_balance NUMBER;

BEGIN

SELECT balance INTO from\_balance

FROM bank\_accounts

WHERE account\_id = from\_acc;

IF from\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in source account.');

ELSE

UPDATE bank\_accounts

SET balance = balance - amount

WHERE account\_id = from\_acc;

UPDATE bank\_accounts

SET balance = balance + amount

WHERE account\_id = to\_acc;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || amount ||

' from account ' || from\_acc ||

' to account ' || to\_acc);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One of the account IDs does not exist.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/

BEGIN

TransferFunds(1, 2, 3000); -- valid transfer

TransferFunds(3, 2, 5000); -- insufficient balance

TransferFunds(99, 2, 100); -- non-existent account

END;

/

SELECT \* FROM bank\_accounts;

**OUTPUT**

