**PL/SQL programming**

**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:**

CREATE TABLE loan\_customers (

customer\_id NUMBER,

age NUMBER,

loan\_interest\_rate NUMBER

);

-- Insert data into loan\_customers

INSERT INTO loan\_customers VALUES (101, 65, 7.5);

INSERT INTO loan\_customers VALUES (102, 45, 8.0);

INSERT INTO loan\_customers VALUES (103, 70, 9.2);

INSERT INTO loan\_customers VALUES (104, 60, 6.5);

INSERT INTO loan\_customers VALUES (105, 75, 7.0);

-- Show initial data

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Initial Current Loan Interest Rates');

FOR c IN (SELECT \* FROM loan\_customers) LOOP

DBMS\_OUTPUT.PUT\_LINE('ID: ' || c.customer\_id || ', Age: ' || c.age || ', Rate: ' || c.loan\_interest\_rate);

END LOOP;

END;

/

-- Apply 1% discount for age > 60

BEGIN

FOR c IN (SELECT \* FROM loan\_customers WHERE age > 60) LOOP

UPDATE loan\_customers

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

WHERE customer\_id = c.customer\_id;

END LOOP;

COMMIT;

END;

/

-- Show updated data

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Updated Current Loan Interest Rates');

FOR c IN (SELECT \* FROM loan\_customers) LOOP

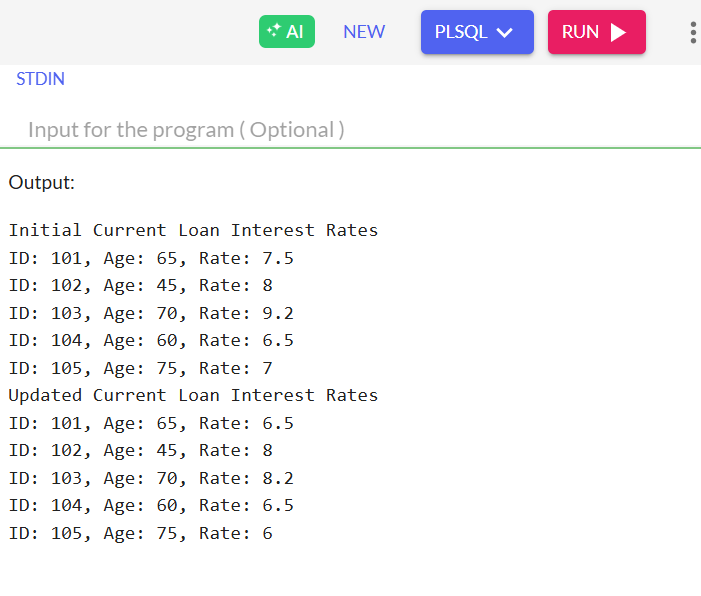
DBMS\_OUTPUT.PUT\_LINE('ID: ' || c.customer\_id || ', Age: ' || c.age || ', Rate: ' || c.loan\_interest\_rate);

END LOOP;

END;

/

**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:**

--Create customer table

CREATE TABLE customers (

customer\_id NUMBER,

name VARCHAR2(50),

balance NUMBER,

isvip CHAR(1) -- 'T' for VIP, 'F' for regular

);

-- Insert some customers

INSERT INTO customers VALUES (1, 'Ausha', 9600, 'F');

INSERT INTO customers VALUES (2, 'Bargav', 12100, 'F');

INSERT INTO customers VALUES (3, 'Harish', 10500, 'F');

INSERT INTO customers VALUES (4, 'Madhu', 8010, 'F');

INSERT INTO customers VALUES (5, 'Shivani', 15200, 'F');

-- PL/SQL block to promote VIPs

BEGIN

FOR rec IN (SELECT \* FROM customers WHERE balance > 10000) LOOP

UPDATE customers

SET isvip = 'T'

WHERE customer\_id = rec.customer\_id;

END LOOP;

COMMIT;

END;

/

--Display updated table

BEGIN

DBMS\_OUTPUT.PUT\_LINE('VIP Status Updated');

FOR rec IN (SELECT \* FROM customers) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Customer ID: ' || rec.customer\_id ||

', Name: ' || rec.name ||

', Balance: ' || rec.balance ||

', IsVIP: ' || rec.isvip

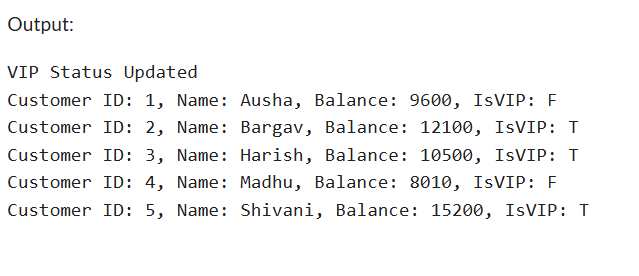
);

END LOOP;

END;

/

**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:**

-- Create table named loan\_customers

CREATE TABLE loan\_customers (

customer\_id NUMBER,

name VARCHAR2(50),

loan\_due\_date DATE

);

-- Insert sample data into the table

INSERT INTO loan\_customers VALUES (1, 'Ausha', SYSDATE + 10);

INSERT INTO loan\_customers VALUES (2, 'Bargav', SYSDATE + 40);

INSERT INTO loan\_customers VALUES (3, 'Harish', SYSDATE + 25);

INSERT INTO loan\_customers VALUES (4, 'Madhu', SYSDATE - 5);

INSERT INTO loan\_customers VALUES (5, 'Shivani', SYSDATE + 30);

-- PL/SQL block to print reminders

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Loan Due Reminders (Next 30 Days)');

FOR rec IN (

SELECT customer\_id, name, loan\_due\_date

FROM loan\_customers

WHERE loan\_due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder Message: Customer "' || rec.name || '" (ID: ' || rec.customer\_id ||

') has a loan due on ' || TO\_CHAR(rec.loan\_due\_date, 'DD-Mon-YYYY')

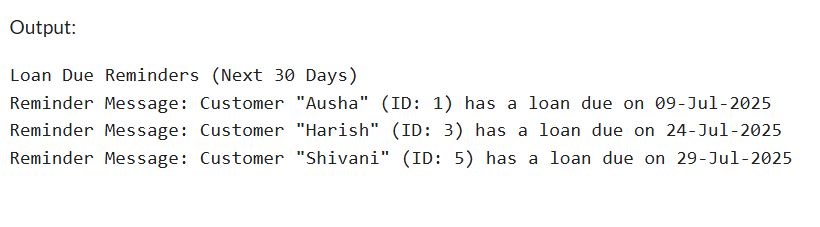
);

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:**

CREATE TABLE savings\_accounts (

account\_id NUMBER,

account\_holder VARCHAR2(50),

balance NUMBER

);

--Insert sample data

INSERT INTO savings\_accounts VALUES (101, 'Ausha', 5200);

INSERT INTO savings\_accounts VALUES (102, 'Bargav', 10000.9);

INSERT INTO savings\_accounts VALUES (103, 'Harish', 7500);

INSERT INTO savings\_accounts VALUES (104, 'Madhu', 1200);

INSERT INTO savings\_accounts VALUES (105, 'Shivani', 6050.86);

--Create the stored procedure ProcessMonthlyInterest

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (SELECT account\_id, balance FROM savings\_accounts) LOOP

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_id = acc.account\_id;

END LOOP;

COMMIT;

END;

/

--Execute the procedure

BEGIN

ProcessMonthlyInterest;

END;

/

--Display updated balances

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Updated Savings Account Balances');

FOR acc IN (SELECT \* FROM savings\_accounts) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Account ID: ' || acc.account\_id ||

', Holder: ' || acc.account\_holder ||

', Balance: ' || TO\_CHAR(acc.balance, '99999.99')

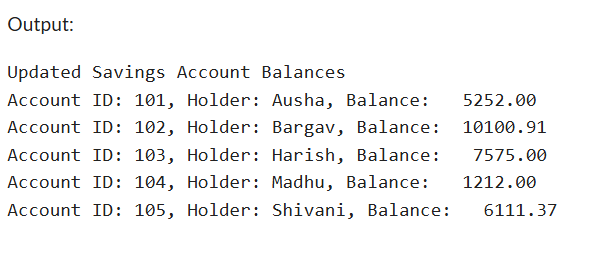
);

END LOOP;

END;

/

**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:**

CREATE TABLE employees (

emp\_id NUMBER,

name VARCHAR2(50),

department VARCHAR2(30),

salary NUMBER

);

--Insert sample employee data

INSERT INTO employees VALUES (1, 'Nikhitha', 'Finance', 80000);

INSERT INTO employees VALUES (2, 'Esha', 'IT', 68000);

INSERT INTO employees VALUES (3, 'Charan', 'Finance', 55000);

INSERT INTO employees VALUES (4, 'Deepika', 'HR', 75000);

INSERT INTO employees VALUES (5, 'Bhuesh', 'IT', 62000);

--Create the stored procedure to apply bonus

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;

END;

/

--Call the procedure with bonus for 'IT' department

BEGIN

UpdateEmployeeBonus('IT', 10); -- Apply 10% bonus to IT employees passed as a Parameter

END;

/

--Display updated salaries

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Updated Employee Salaries');

FOR rec IN (SELECT \* FROM employees) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'ID: ' || rec.emp\_id ||

', Name: ' || rec.name ||

', Dept: ' || rec.department ||

', Salary:' || TO\_CHAR(rec.salary,'999999.99')

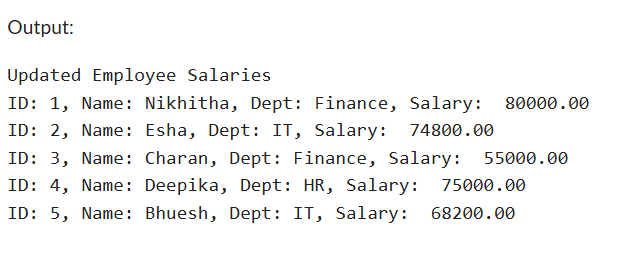
);

END LOOP;

END;

/

**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:**

**CREATE TABLE bank\_accounts (**

**account\_id NUMBER PRIMARY KEY,**

**account\_holder VARCHAR2(50),**

**balance NUMBER**

**);**

**INSERT INTO bank\_accounts VALUES (1001, 'Arun', 8080.30);**

**INSERT INTO bank\_accounts VALUES (1002, 'Navya', 6087.98);**

**INSERT INTO bank\_accounts VALUES (1003, 'Spoorthi', 4900.00);**

**--Create the TransferFunds stored procedure**

**CREATE OR REPLACE PROCEDURE TransferFunds (**

**from\_account IN NUMBER,**

**to\_account IN NUMBER,**

**amount IN NUMBER**

**) IS**

**from\_balance NUMBER;**

**BEGIN**

**-- Get the balance of the source account**

**SELECT balance INTO from\_balance**

**FROM bank\_accounts**

**WHERE account\_id = from\_account;**

**-- Check if sufficient balance exists**

**IF from\_balance < amount THEN**

**DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance');**

**ELSE**

**-- Deduct from source**

**UPDATE bank\_accounts**

**SET balance = balance - amount**

**WHERE account\_id = from\_account;**

**-- Add to destination**

**UPDATE bank\_accounts**

**SET balance = balance + amount**

**WHERE account\_id = to\_account;**

**COMMIT;**

**DBMS\_OUTPUT.PUT\_LINE('Transfer successful: ' || amount ||**

**' transferred from Account ' || from\_account ||**

**' to Account ' || to\_account);**

**END IF;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('Transfer failed: One or both account IDs are invalid.');**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.PUT\_LINE('Transfer failed due to unexpected error: ' || SQLERRM);**

**END;**

**/**

**--Execute the procedure**

**BEGIN**

**TransferFunds(1001, 1002, 2000); -- Transfer ₹2000**

**END;**

**/**

**--Display updated account balances**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('Updated Account Balances');**

**FOR acc IN (SELECT \* FROM bank\_accounts) LOOP**

**DBMS\_OUTPUT.PUT\_LINE(**

**'Account ID: ' || acc.account\_id ||**

**', Holder: ' || acc.account\_holder ||**

**', Balance:' || TO\_CHAR(acc.balance, '999999.99')**

**);**

**END LOOP;**

**END;**

**/**

**OUTPUT:**

