**Exercise 4: Functions**

**Scenario 1:** Calculate the age of customers for eligibility checks.

**Question:** Write a function CalculateAge that takes a customer's date of birth as input and returns their age in years.

**Query**:

CREATE OR REPLACE FUNCTION CalculateAge (p\_dob DATE)

RETURN NUMBER

IS

v\_age NUMBER;

BEGIN

v\_age := TRUNC(MONTHS\_BETWEEN(SYSDATE, p\_dob) / 12);

RETURN v\_age;

END CalculateAge;

**Function Calling:**

**Query:**

SELECT CustomerID, Name, DOB, CalculateAge(DOB) AS Age

FROM Customers;

**Scenario 2:** The bank needs to compute the monthly installment for a loan.

**Question:** Write a function **CalculateMonthlyInstallment** that takes the loan amount, interest rate, and loan duration in years as input and returns the monthly installment amount.

**Query:**

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment (

p\_loan\_amount NUMBER,

p\_annual\_interest\_rate NUMBER,

p\_duration\_years NUMBER

) RETURN NUMBER

IS

v\_monthly\_interest\_rate NUMBER;

v\_total\_payments NUMBER;

v\_monthly\_installment NUMBER;

BEGIN

v\_monthly\_interest\_rate := p\_annual\_interest\_rate / 12 / 100;

v\_total\_payments := p\_duration\_years \* 12;

v\_monthly\_installment := p\_loan\_amount \* v\_monthly\_interest\_rate \* POWER(1 + v\_monthly\_interest\_rate, v\_total\_payments) / (POWER(1 + v\_monthly\_interest\_rate, v\_total\_payments) - 1);

RETURN v\_monthly\_installment;

END CalculateMonthlyInstallment;

**Function Calling:**

**Query:**

DECLARE

v\_monthly\_installment NUMBER;

BEGIN

v\_monthly\_installment := CalculateMonthlyInstallment(5000, 5, 5);

DBMS\_OUTPUT.PUT\_LINE('Monthly Installment: ' || v\_monthly\_installment);

END;

**Scenario 3:** Check if a customer has sufficient balance before making a transaction.

**Question:** Write a function **HasSufficientBalance** that takes an account ID and an amount as input and returns a boolean indicating whether the account has at least the specified amount.

**Query:**

CREATE OR REPLACE FUNCTION HasSufficientBalance (

p\_account\_id NUMBER,

p\_amount NUMBER

) RETURN BOOLEAN

IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_account\_id;

IF v\_balance >= p\_amount THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'An error occurred while checking the balance.');

END HasSufficientBalance;

**Function Calling:**

**Query:**

DECLARE

v\_result BOOLEAN;

BEGIN

v\_result := HasSufficientBalance(1, 500);

IF v\_result THEN

DBMS\_OUTPUT.PUT\_LINE('Account 1 has sufficient balance.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Account 1 does not have sufficient balance .');

END IF;

v\_result := HasSufficientBalance(2, 2000);

IF v\_result THEN

DBMS\_OUTPUT.PUT\_LINE('Account 2 has sufficient balance.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Account 2 does not have sufficient balance.');

END IF;

END;