Scenario 1: CalculateAge - Calculate the Age of Customers

CREATE OR REPLACE FUNCTION CalculateAge(p\_DOB IN DATE)

RETURN NUMBER

IS

v\_Age NUMBER;

BEGIN

v\_Age := FLOOR(MONTHS\_BETWEEN(SYSDATE, p\_DOB) / 12);

RETURN v\_Age;

EXCEPTION

WHEN OTHERS THEN

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

RETURN NULL;

END;

/

Scenario 2: CalculateMonthlyInstallment - Compute the Monthly Installment for a Loan

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment(

p\_LoanAmount IN NUMBER,

p\_InterestRate IN NUMBER,

p\_LoanDurationYears IN NUMBER

)

RETURN NUMBER

IS

v\_MonthlyInterestRate NUMBER;

v\_NumberOfPayments NUMBER;

v\_MonthlyInstallment NUMBER;

BEGIN

v\_MonthlyInterestRate := p\_InterestRate / 1200; -- Annual rate to monthly rate (assuming interest rate is given in percentage)

v\_NumberOfPayments := p\_LoanDurationYears \* 12; -- Total number of monthly payments

v\_MonthlyInstallment := p\_LoanAmount \* v\_MonthlyInterestRate /

(1 - POWER(1 + v\_MonthlyInterestRate, -v\_NumberOfPayments));

RETURN v\_MonthlyInstallment;

EXCEPTION

WHEN OTHERS THEN

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

RETURN NULL;

END;

/

Scenario 3: HasSufficientBalance - Check if a Customer Has Sufficient Balance

CREATE OR REPLACE FUNCTION HasSufficientBalance(

p\_AccountID IN Accounts.AccountID%TYPE,

p\_Amount IN NUMBER

)

RETURN BOOLEAN

IS

v\_Balance Accounts.Balance%TYPE;

BEGIN

SELECT Balance INTO v\_Balance FROM Accounts WHERE AccountID = p\_AccountID;

IF v\_Balance >= p\_Amount THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Account not found');

RETURN FALSE;

WHEN OTHERS THEN

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

RETURN FALSE;

END;

/