**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 (date\_of\_birth IN DATE) RETURN NUMBER AS

age\_in\_years NUMBER;

BEGIN

SELECT FLOOR(MONTHS\_BETWEEN(SYSDATE, date\_of\_birth) / 12) INTO age\_in\_years

FROM dual;

RETURN age\_in\_years;

END CalculateAge;

**Explanation:**

We use the MONTHS\_BETWEEN function to calculate the difference in months between the current date (SYSDATE) and the customer's date of birth. By dividing the result by 12 and using the FLOOR function, we obtain the age of the customer in years. The calculated age is then returned as the output of the function.

**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 (loan\_amount IN NUMBER, interest\_rate IN NUMBER, loan\_duration\_years IN NUMBER) RETURN NUMBER AS

monthly\_interest\_rate NUMBER;

total\_months NUMBER := loan\_duration\_years \* 12;

monthly\_installment NUMBER;

BEGIN

monthly\_interest\_rate := interest\_rate / 12 / 100;

monthly\_installment := (loan\_amount \* monthly\_interest\_rate) / (1 - (1 + monthly\_interest\_rate)\*\*-total\_months);

RETURN monthly\_installment;

END CalculateMonthlyInstallment;

**Explanation:**

We calculate the monthly interest rate by dividing the annual interest rate by 12 (number of months in a year) and converting it to a decimal. We determine the total number of months for the loan by multiplying the loan duration in years by 12. The monthly installment amount is computed using the formula for the monthly loan payment based on the loan amount, monthly interest rate, and total number of months. The calculated monthly installment amount is then returned as the output of the function.

**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 (account\_id IN NUMBER, amount IN NUMBER) RETURN BOOLEAN AS

account\_balance NUMBER;

sufficient\_balance BOOLEAN := FALSE;

BEGIN

SELECT balance INTO account\_balance

FROM accounts

WHERE id = account\_id;

IF account\_balance >= amount THEN

sufficient\_balance := TRUE;

END IF;

RETURN sufficient\_balance;

END HasSufficientBalance;

**Explanation:**

We retrieve the balance associated with the provided account ID from the accounts table. The function checks if the account balance is greater than or equal to the specified amount. If the account has sufficient balance for the transaction, the function sets the boolean value to TRUE; otherwise, it remains FALSE. The function returns the boolean value indicating whether the account has enough balance for the transaction.