**-- SCENARIO 1**

DELETE FROM CUSTOMERS WHERE CUSTOMERID = 3;

SELECT \* FROM CUSTOMERS;

SET SERVEROUTPUT ON;

CREATE OR REPLACE FUNCTION CALCULATEAGE(

P\_DOB IN DATE

) RETURN NUMBER IS

V\_AGE NUMBER;

BEGIN

V\_AGE := TRUNC((SYSDATE - P\_DOB) / 365);

RETURN V\_AGE;

END CALCULATEAGE;

/

SET SERVEROUTPUT ON;

DECLARE

CURSOR CURSOR\_CUST IS SELECT CUSTOMERID, DOB FROM CUSTOMERS;

V\_CUSTOMERID CUSTOMERS.CUSTOMERID%TYPE;

V\_DOB CUSTOMERS.DOB%TYPE;

V\_AGE NUMBER;

BEGIN

OPEN CURSOR\_CUST;

LOOP

FETCH CURSOR\_CUST INTO V\_CUSTOMERID, V\_DOB;

EXIT WHEN CURSOR\_CUST%NOTFOUND;

V\_AGE := CALCULATEAGE(V\_DOB);

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER ID : ' || V\_CUSTOMERID || ' AGE : ' || V\_AGE);

END LOOP;

CLOSE CURSOR\_CUST;

END;

/

**-- SCENARIO 2**

SELECT \* FROM LOANS;

SET SERVEROUTPUT ON;

CREATE OR REPLACE FUNCTION CALCULATEMONTHLYINSTALLMENT(

P\_LOAN\_AMOUNT IN NUMBER,

P\_INTEREST\_RATE IN NUMBER,

P\_LOAN\_DURATION\_YEARS IN NUMBER

) RETURN NUMBER IS

V\_MONTHLY\_RATE NUMBER;

V\_NUM\_PAYMENTS NUMBER;

V\_MONTHLY\_INSTALLMENT NUMBER;

BEGIN

V\_MONTHLY\_RATE := P\_INTEREST\_RATE / 12 / 100;

V\_NUM\_PAYMENTS := P\_LOAN\_DURATION\_YEARS \* 12;

IF V\_MONTHLY\_RATE = 0 THEN

V\_MONTHLY\_INSTALLMENT := P\_LOAN\_AMOUNT / V\_NUM\_PAYMENTS;

ELSE

V\_MONTHLY\_INSTALLMENT := P\_LOAN\_AMOUNT \* V\_MONTHLY\_RATE / (1 - POWER(1 + V\_MONTHLY\_RATE, -V\_NUM\_PAYMENTS));

END IF;

RETURN V\_MONTHLY\_INSTALLMENT;

END CALCULATEMONTHLYINSTALLMENT;

/

SET SERVEROUTPUT ON;

DECLARE

CURSOR LOAN\_CUR IS SELECT \* FROM LOANS;

V\_DATA LOANS%ROWTYPE;

V\_DURATION NUMBER;

V\_MONTHLYINSTALLMENT NUMBER;

BEGIN

OPEN LOAN\_CUR;

LOOP

FETCH LOAN\_CUR INTO V\_DATA;

EXIT WHEN LOAN\_CUR%NOTFOUND;

V\_DURATION := TRUNC((V\_DATA.ENDDATE - V\_DATA.STARTDATE)/365);

V\_MONTHLYINSTALLMENT := TRUNC(CALCULATEMONTHLYINSTALLMENT(V\_DATA.LOANAMOUNT, V\_DATA.INTERESTRATE, V\_DURATION),2);

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER ID : ' || V\_DATA.CUSTOMERID || ' MONTHLY INSTALLAMENT : ' || V\_MONTHLYINSTALLMENT);

END LOOP;

CLOSE LOAN\_CUR;

END;

/

**-- SCENARIO 3**

SELECT \* FROM ACCOUNTS;

SET SERVEROUTPUT ON;

CREATE OR REPLACE FUNCTION HASSUFFICIENTBALANCE(

P\_ACCOUNT\_ID 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\_ACCOUNT\_ID;

RETURN V\_BALANCE >= P\_AMOUNT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Error checking balance: ' || SQLERRM);

END HASSUFFICIENTBALANCE;

/

SET SERVEROUTPUT ON;

DECLARE

V\_ACCOUNTID ACCOUNTS.ACCOUNTID%TYPE := &ACCOUNTID;

V\_AMOUNT NUMBER := &AMOUNT;

V\_HAS BOOLEAN;

BEGIN

V\_HAS := HASSUFFICIENTBALANCE(V\_ACCOUNTID, V\_AMOUNT);

IF V\_HAS = TRUE THEN DBMS\_OUTPUT.PUT\_LINE(V\_ACCOUNTID || ' HAS SUFFICIENT AMOUNT');

ELSE DBMS\_OUTPUT.PUT\_LINE(V\_ACCOUNTID || ' DOES NOT HAVE SUFFICIENT AMOUNT');

END IF;

END;

/