Functions

1. DECLARE

tlahir DATE;

umur NUMBER;

BEGIN

tlahir := TO\_DATE('&indate', 'DD-MM-YYYY');

umur := FLOOR((SYSDATE - tlahir) / 365);

DBMS\_OUTPUT.PUT\_LINE('Your age is ' || umur || ' years old');

END;

/

1. DECLARE @n INT = 360, -- total # of payments

@ir FLOAT = 0.05, -- annual interest rate (enter as a decimal, e.g., 5% as 0.05)

@pv FLOAT = 225000.00, -- present value (original loan amount)

@npy INT = 12, -- # of periods per year

@beg\_dt DATE = GETDATE(); -- the date of the first payment

WITH cte\_n1 (n) AS (SELECT 1 FROM (VALUES (1),(1),(1),(1),(1),(1),(1),(1),(1),(1)) n (n)),

cte\_n2 (n) AS (SELECT 1 FROM cte\_n1 a CROSS JOIN cte\_n1 b),

cte\_Tally (n) AS (

SELECT TOP (@n) ROW\_NUMBER() OVER (ORDER BY a.n)

FROM cte\_n2 a CROSS JOIN cte\_n2 b

ORDER BY a.n

)

SELECT pmt\_num = t.n,

pd.payment\_date,

beg\_balance = CONVERT(DECIMAL(19, 2), pv.beg\_balance),

scheduled\_pmt = CONVERT(DECIMAL(19, 2), pmt.pmt\_calc),

amt\_to\_interest = CONVERT(DECIMAL(19, 2), ipmt.ipmt),

amt\_to\_principal = CONVERT(DECIMAL(19, 2), ppmt.ppmt),

end\_balance = CONVERT(DECIMAL(19, 2), pv.end\_balance)

FROM cte\_Tally t

CROSS APPLY (

VALUES (

@pv / (POWER(1 + (@ir / @npy), @n) - 1) \*

((@ir / @npy) \* POWER(1 + (@ir / @npy), @n))

)

) pmt (pmt\_calc)

CROSS APPLY (

VALUES (

ABS(-@pv \* POWER(1 + (@ir / @npy), t.n - 1) +

pmt.pmt\_calc \* (POWER(1 + (@ir / @npy), t.n - 1) - 1) / (@ir / @npy)),

ABS(-@pv \* POWER(1 + (@ir / @npy), t.n) +

pmt.pmt\_calc \* (POWER(1 + (@ir / @npy), t.n) - 1) / (@ir / @npy))

)

) pv (beg\_balance, end\_balance)

CROSS APPLY (

VALUES (pv.beg\_balance \* (@ir / @npy))

) ipmt (ipmt)

CROSS APPLY (

VALUES (pmt.pmt\_calc - ipmt.ipmt)

) ppmt (ppmt)

CROSS APPLY (

VALUES (

CASE

WHEN @npy <= 12 THEN DATEADD(MONTH, (12 / @npy) \* (t.n - 1), @beg\_dt)

WHEN @npy = 26 THEN DATEADD(WEEK, 2 \* (t.n - 1), @beg\_dt)

ELSE DATEADD(DAY, (365 / @npy) \* (t.n - 1), @beg\_dt)

END

)

) pd (payment\_date);

1. CREATE OR REPLACE FUNCTION HasSufficientBalance(

p\_account\_id NUMBER,

p\_transaction\_amount NUMBER

) RETURN BOOLEAN IS

v\_account\_balance NUMBER;

BEGIN

SELECT balance INTO v\_account\_balance

FROM accounts

WHERE account\_id = p\_account\_id;

IF v\_account\_balance >= p\_transaction\_amount THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

END HasSufficientBalance;

/