Functions

--Scenario 1

CREATE OR REPLACE FUNCTION CalculateAge (

p\_dob IN DATE

) RETURN NUMBER

IS

v\_age NUMBER;

BEGIN

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

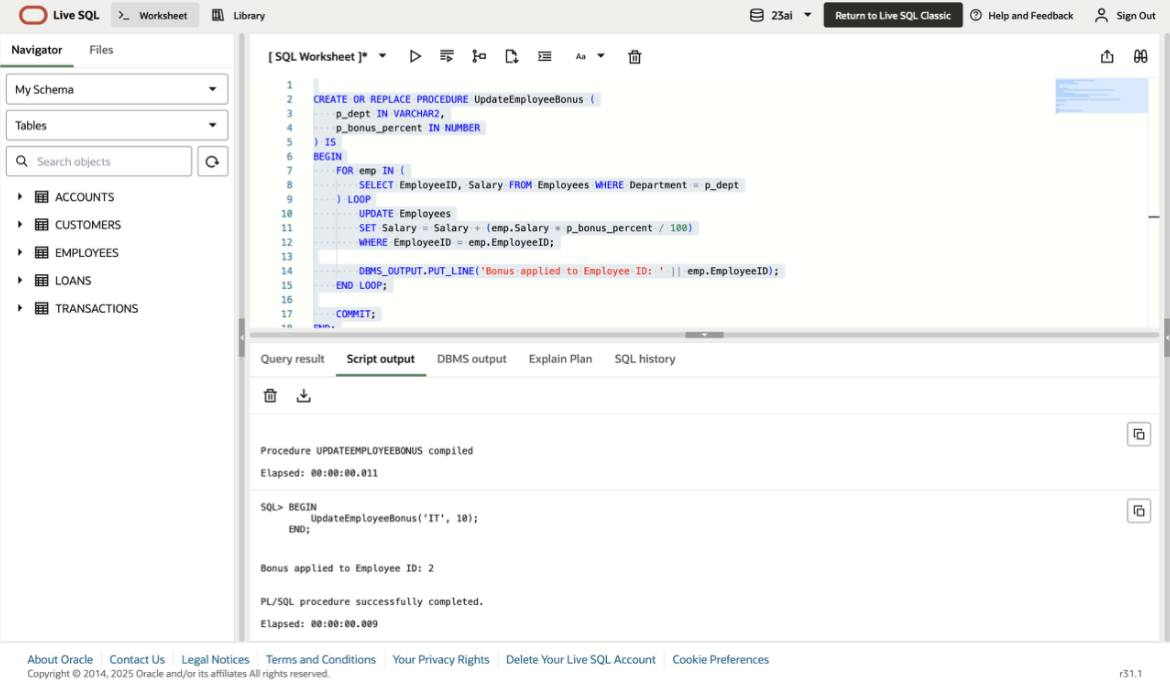
RETURN v\_age;

END;

/

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

FROM Customers;



--Scenario 2

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment (

p\_loan\_amount IN NUMBER,

p\_interest\_rate IN NUMBER,

p\_years IN NUMBER

) RETURN NUMBER

IS

v\_emi NUMBER;

r NUMBER := p\_interest\_rate / 12 / 100;

n NUMBER := p\_years \* 12;

BEGIN

IF r = 0 THEN

v\_emi := p\_loan\_amount / n;

ELSE

v\_emi := p\_loan\_amount \* r \* POWER(1 + r, n) / (POWER(1 + r, n) - 1);

END IF;

RETURN ROUND(v\_emi, 2);

END;

/

SELECT LoanID, CustomerID,

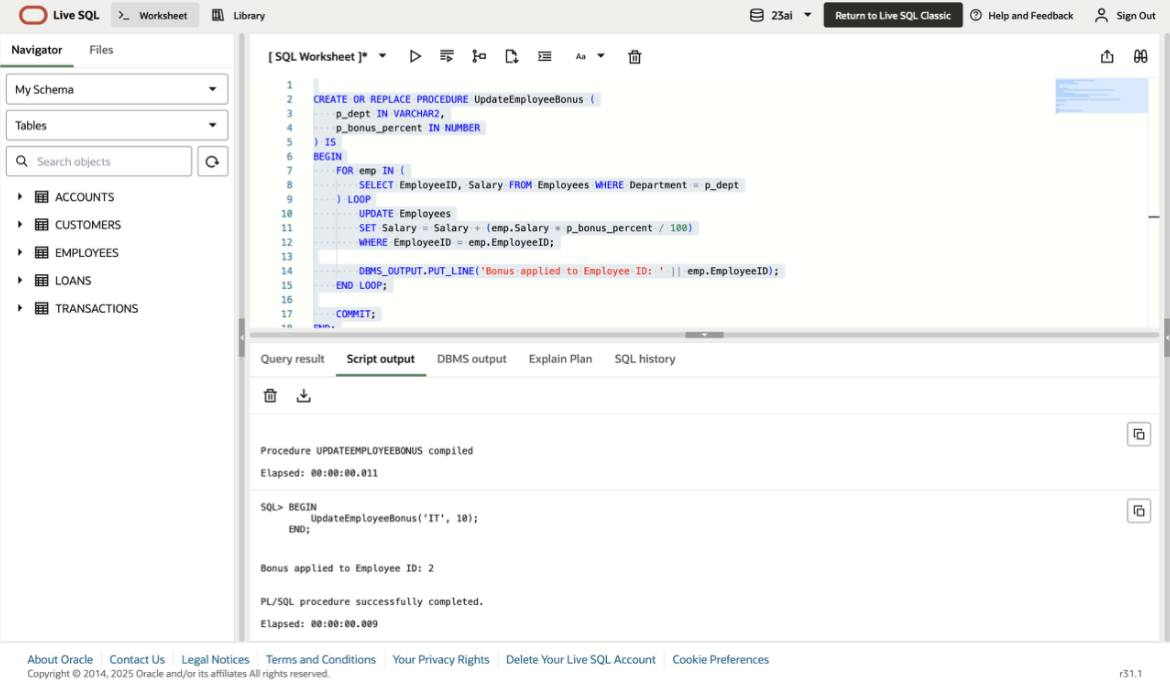
LoanAmount,

InterestRate,

ROUND(MONTHS\_BETWEEN(EndDate, StartDate)/12) AS Years,

CalculateMonthlyInstallment(LoanAmount, InterestRate, ROUND(MONTHS\_BETWEEN(EndDate, StartDate)/12)) AS EMI

FROM Loans;



--scenario 3

CREATE OR REPLACE FUNCTION HasSufficientBalance (

p\_account\_id IN NUMBER,

p\_amount IN NUMBER

) RETURN BOOLEAN

IS

v\_balance NUMBER;

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;

END;

/

DECLARE

v\_result BOOLEAN;

BEGIN

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

IF v\_result THEN

DBMS\_OUTPUT.PUT\_LINE('Sufficient balance.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance.');

END IF;

END;

/

SELECT \* FRom ACCOUNTS;

