**Exercise 1: Control Structures**

**Scenario 1**

Code ->

BEGIN

FOR cust\_rec IN (SELECT CustomerID, DOB FROM Customers) LOOP

IF MONTHS\_BETWEEN(SYSDATE, cust\_rec.DOB) / 12 > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust\_rec.CustomerID;

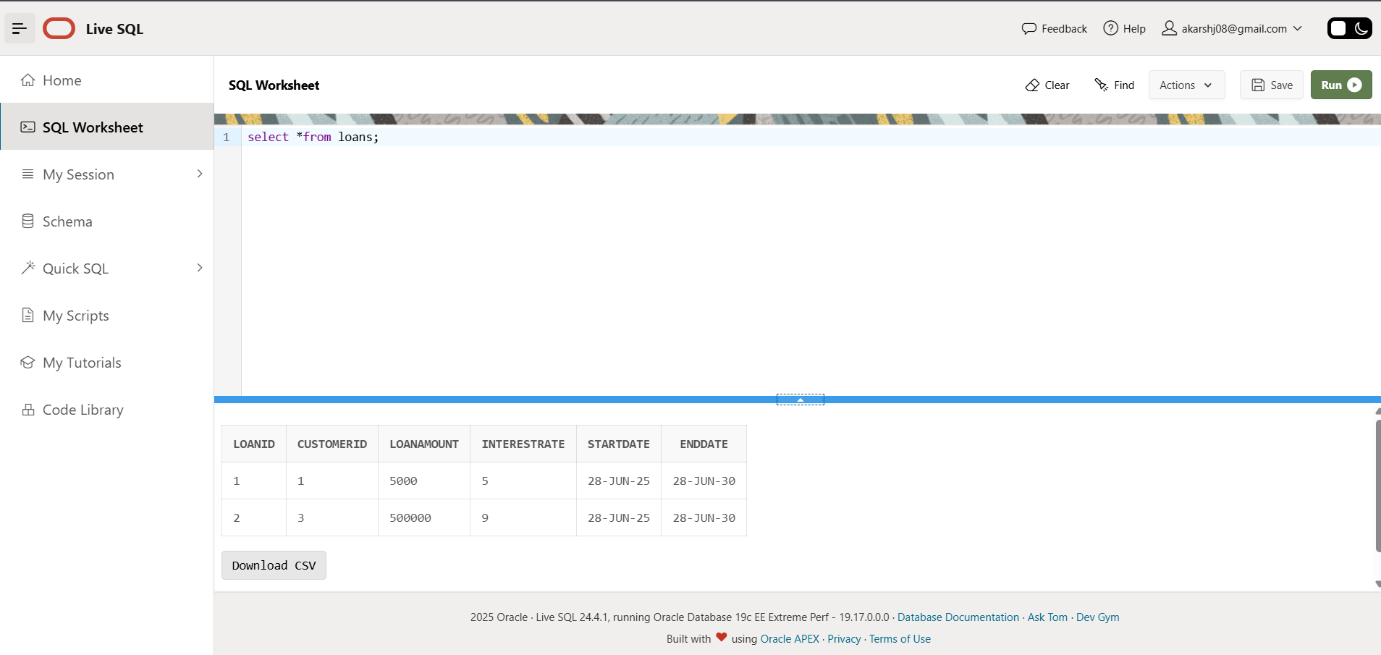
END IF;

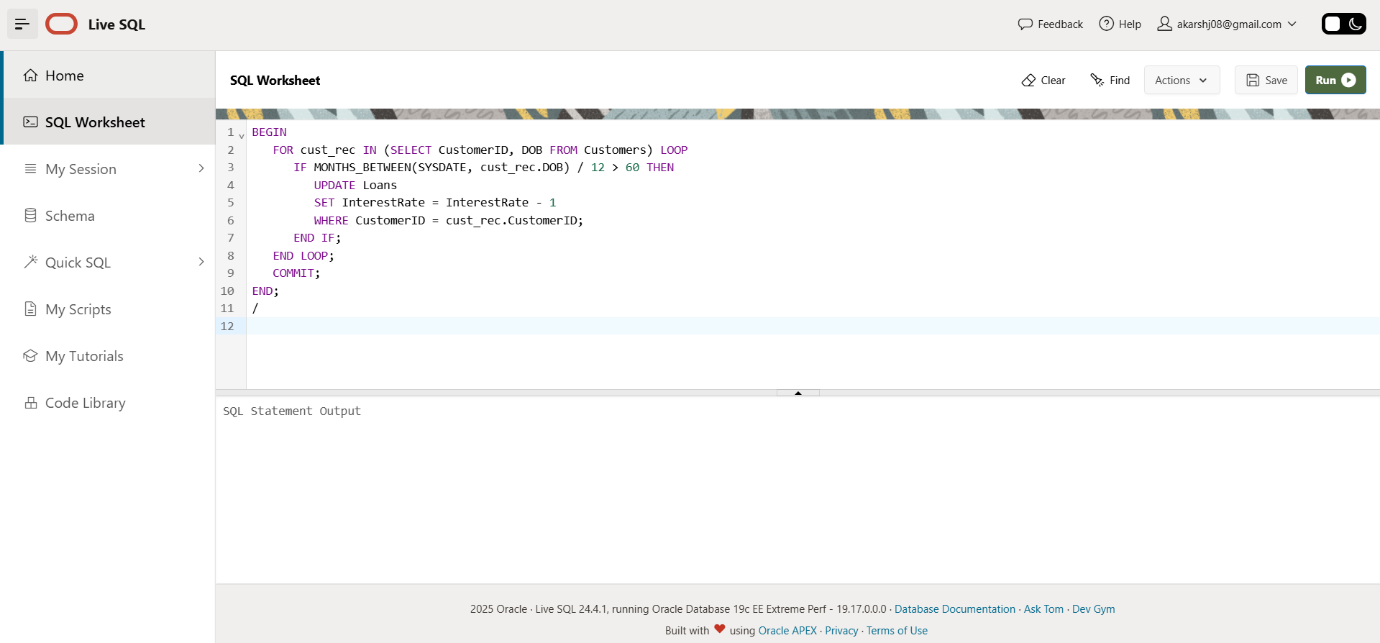
END LOOP;

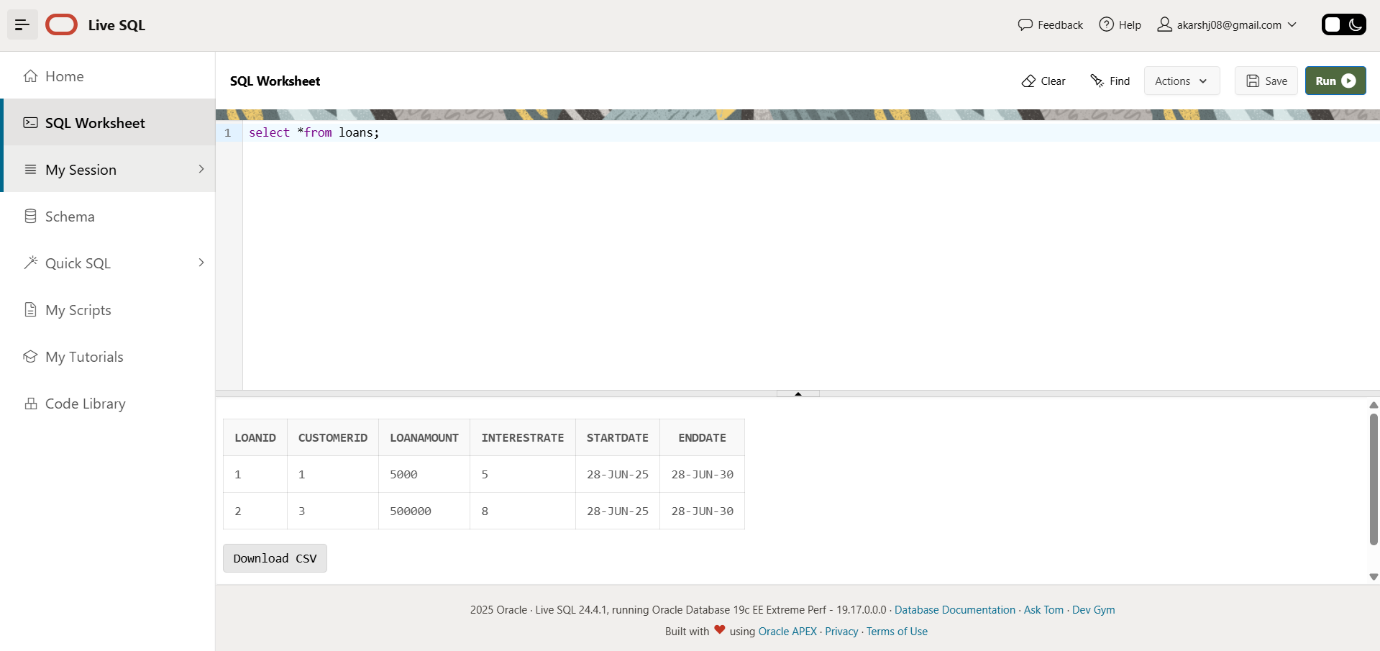
COMMIT;

END;

Output ->







Scenario 2

Code ->

BEGIN

FOR cust\_rec IN (SELECT CustomerID, Balance FROM Customers) LOOP

IF cust\_rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = cust\_rec.CustomerID;

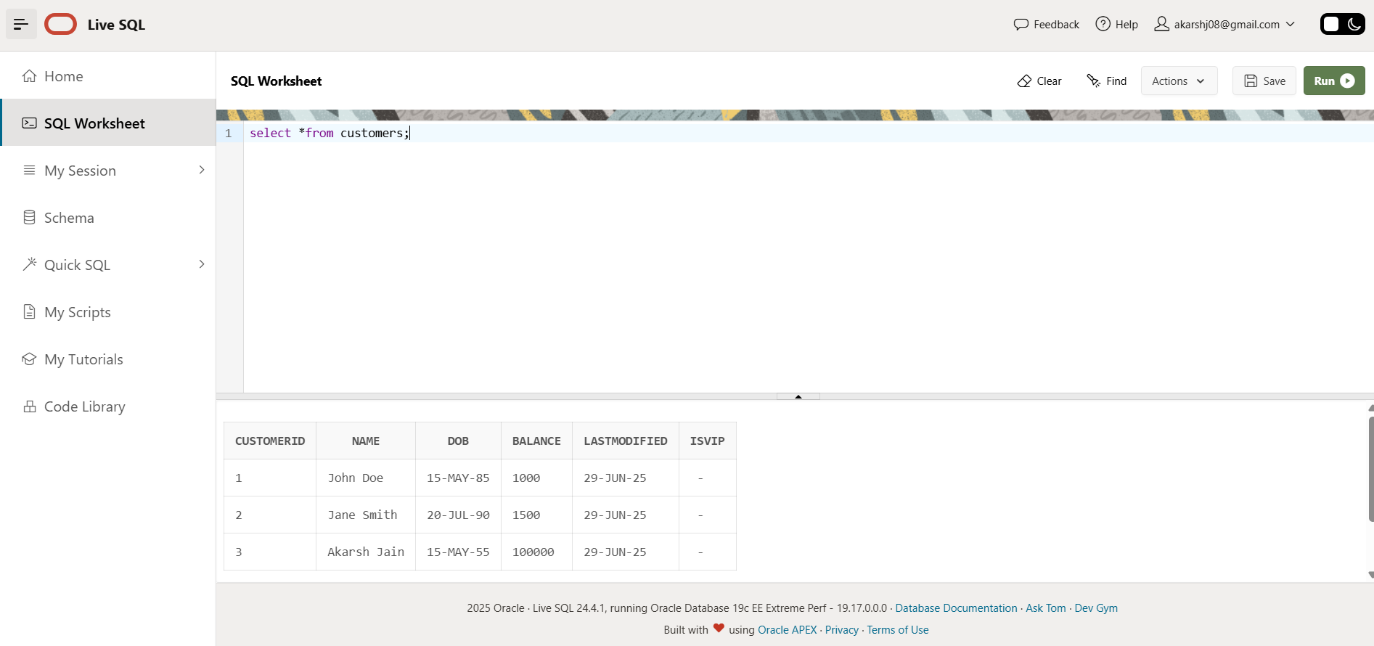
END IF;

END LOOP;

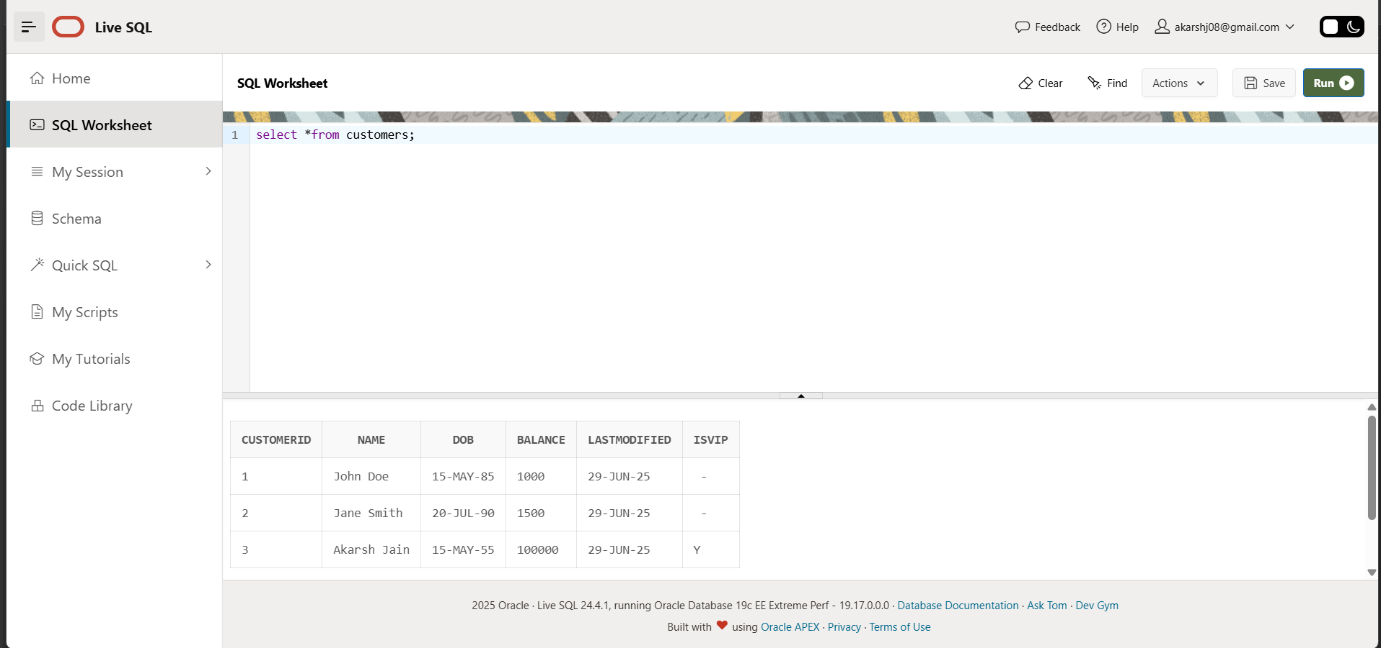
COMMIT;

END;

Output ->







Scenario 3

Code ->

SET SERVEROUTPUT ON;

BEGIN

FOR loan\_rec IN (

SELECT L.LoanID, C.Name, L.EndDate

FROM Loans L

JOIN Customers C ON L.CustomerID = C.CustomerID

WHERE L.EndDate <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.LoanID ||

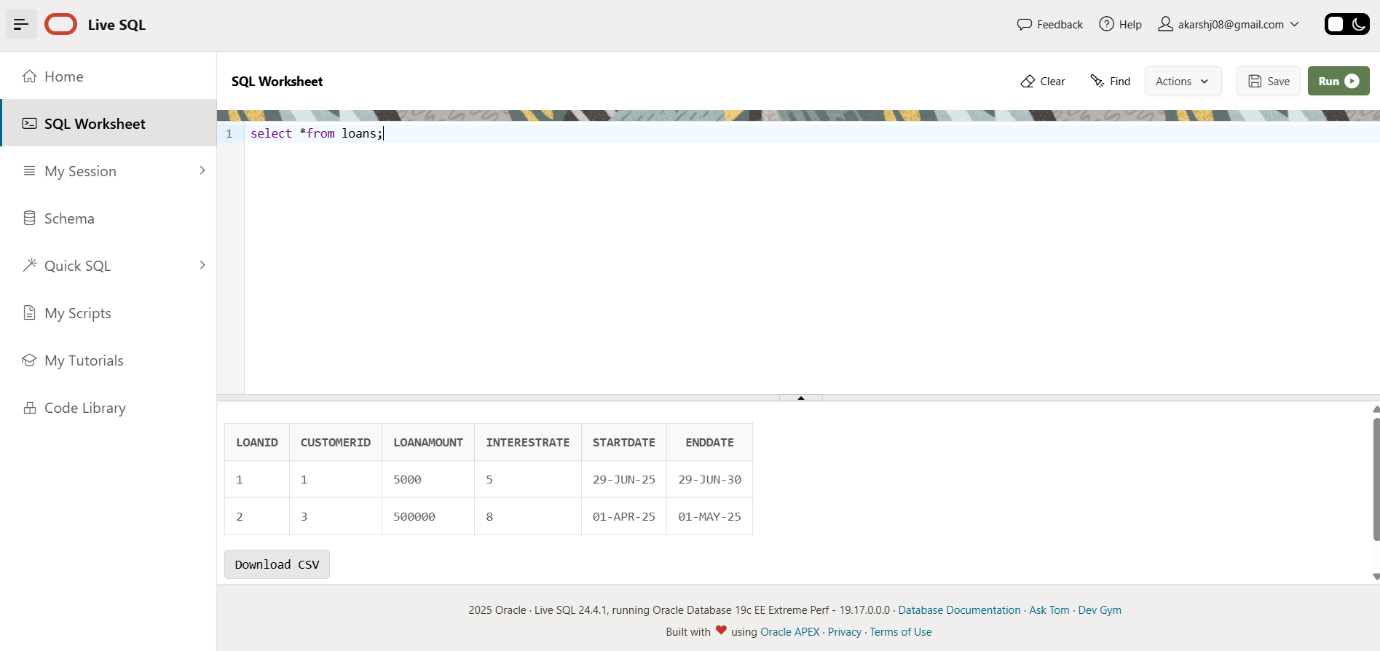
' for ' || loan\_rec.Name ||

' is due on ' || TO\_CHAR(loan\_rec.EndDate, 'YYYY-MM-DD'));

END LOOP;

END;

Output ->





Exercise 3: Stored Procedures

Scenario 1

Code ->

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc\_rec IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

SET Balance = Balance + (acc\_rec.Balance \* 0.01),

LastModified = SYSDATE

WHERE AccountID = acc\_rec.AccountID;

END LOOP;

COMMIT;

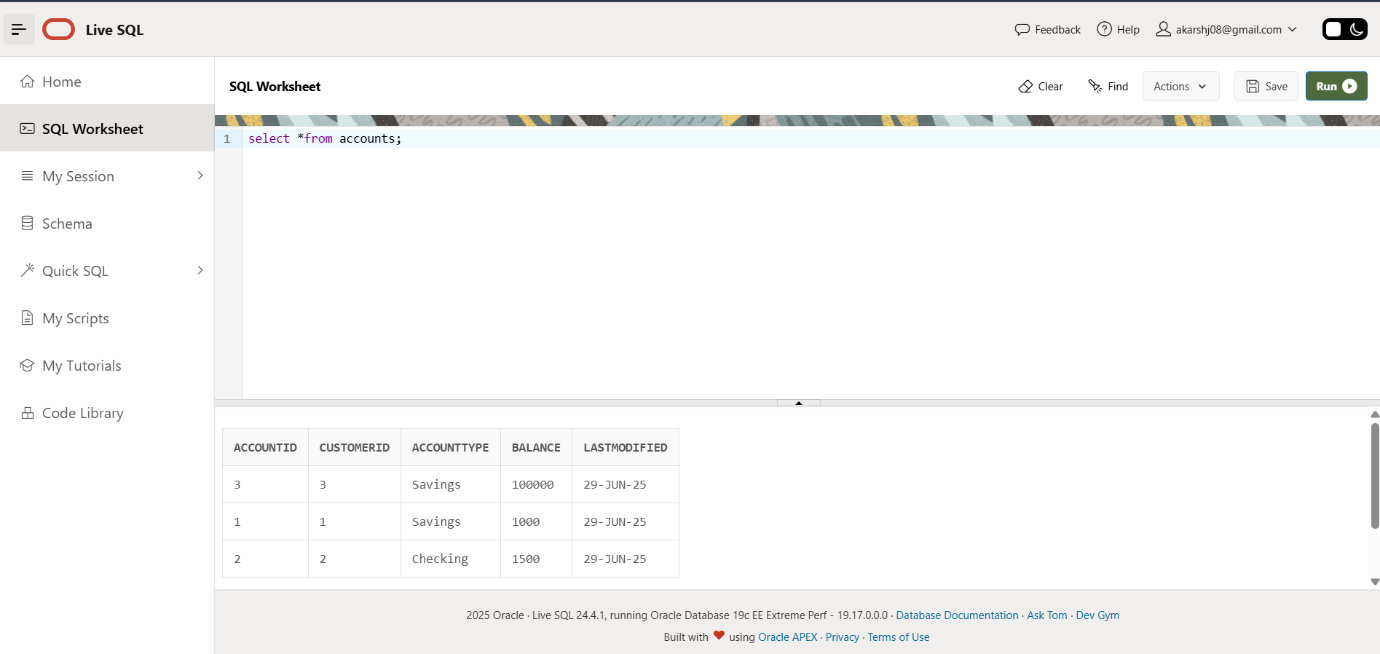
END;

BEGIN

ProcessMonthlyInterest ();

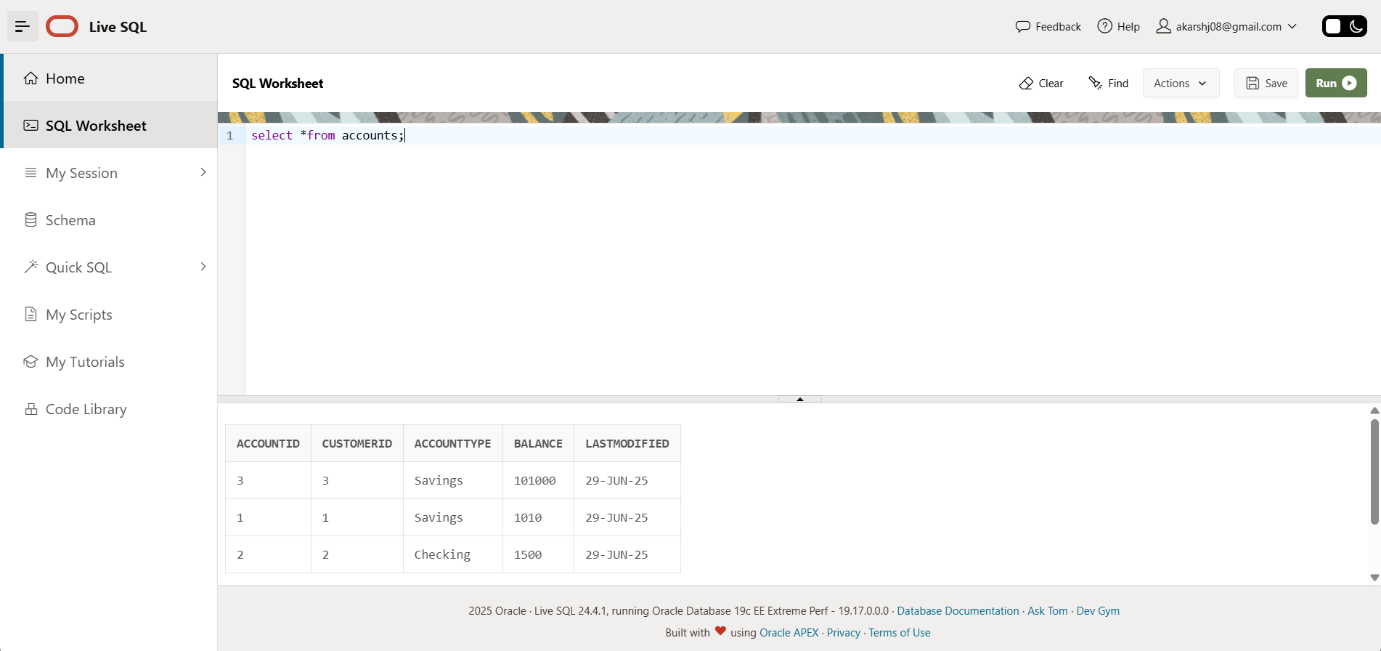
END;

Output ->









Scenario 2

Code ->

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_Department IN VARCHAR2,

p\_BonusPercent IN NUMBER

) AS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* p\_BonusPercent / 100)

WHERE Department = p\_Department;

COMMIT;

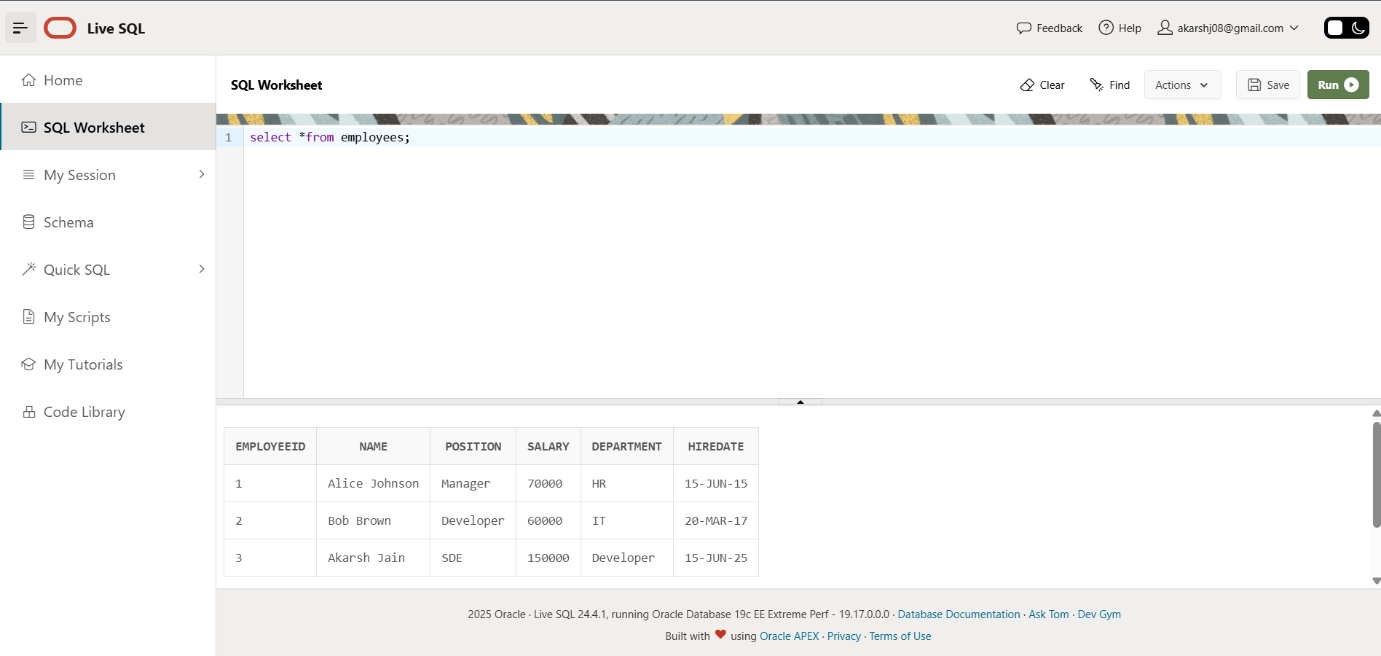
END;

BEGIN

UpdateEmployeeBonus('Developer', 10);

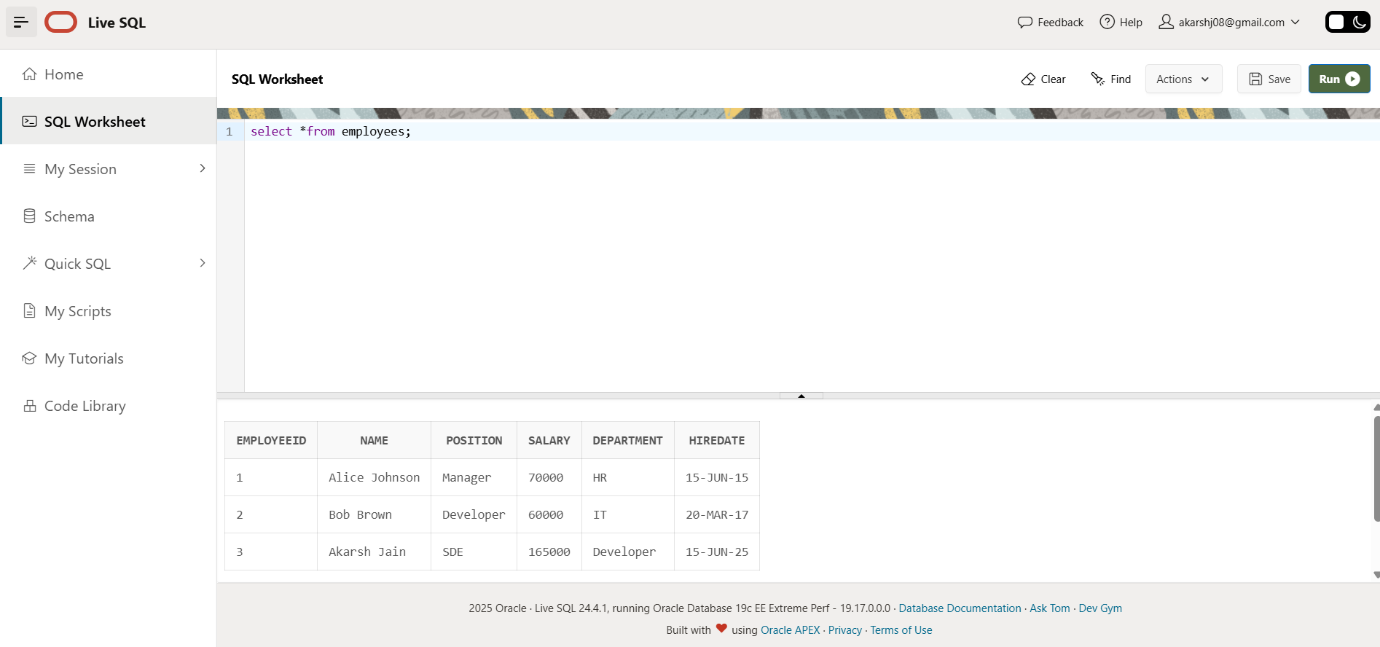
END;

Output ->









Scenario 3

Code ->

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

) AS

v\_FromBalance NUMBER;

BEGIN

SELECT Balance INTO v\_FromBalance

FROM Accounts

WHERE AccountID = p\_FromAccountID

FOR UPDATE;

IF v\_FromBalance < p\_Amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance in source account.');

END IF;

UPDATE Accounts

SET Balance = Balance - p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_FromAccountID;

UPDATE Accounts

SET Balance = Balance + p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_ToAccountID;

COMMIT;

END;

BEGIN

TransferFunds(1, 2, 500);

END;

Output ->

