**01: Control Structures**

SCENARIO 1 :

**CODE :**

DECLARE

v\_found BOOLEAN := FALSE;

BEGIN

FOR cust IN (

SELECT l.LoanID, l.InterestRate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12 > 60

) LOOP

UPDATE Loans

SET InterestRate = cust.InterestRate - (cust.InterestRate \* 0.01)

WHERE LoanID = cust.LoanID;

v\_found := TRUE;

END LOOP;

IF NOT v\_found THEN

DBMS\_OUTPUT.PUT\_LINE('No customer found above age 60.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Interest rate updated for eligible senior citizens.');

END IF;

COMMIT;

END;

/

ScreenShot and Output :



SCENARIO 2 :

**CODE** :

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

DECLARE

v\_found BOOLEAN := FALSE;

BEGIN

FOR cust IN (

SELECT CustomerID

FROM Customers

WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

v\_found := TRUE;

END LOOP;

IF NOT v\_found THEN

DBMS\_OUTPUT.PUT\_LINE('No customer eligible for VIP promotion.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('VIP status updated for eligible customers.');

END IF;

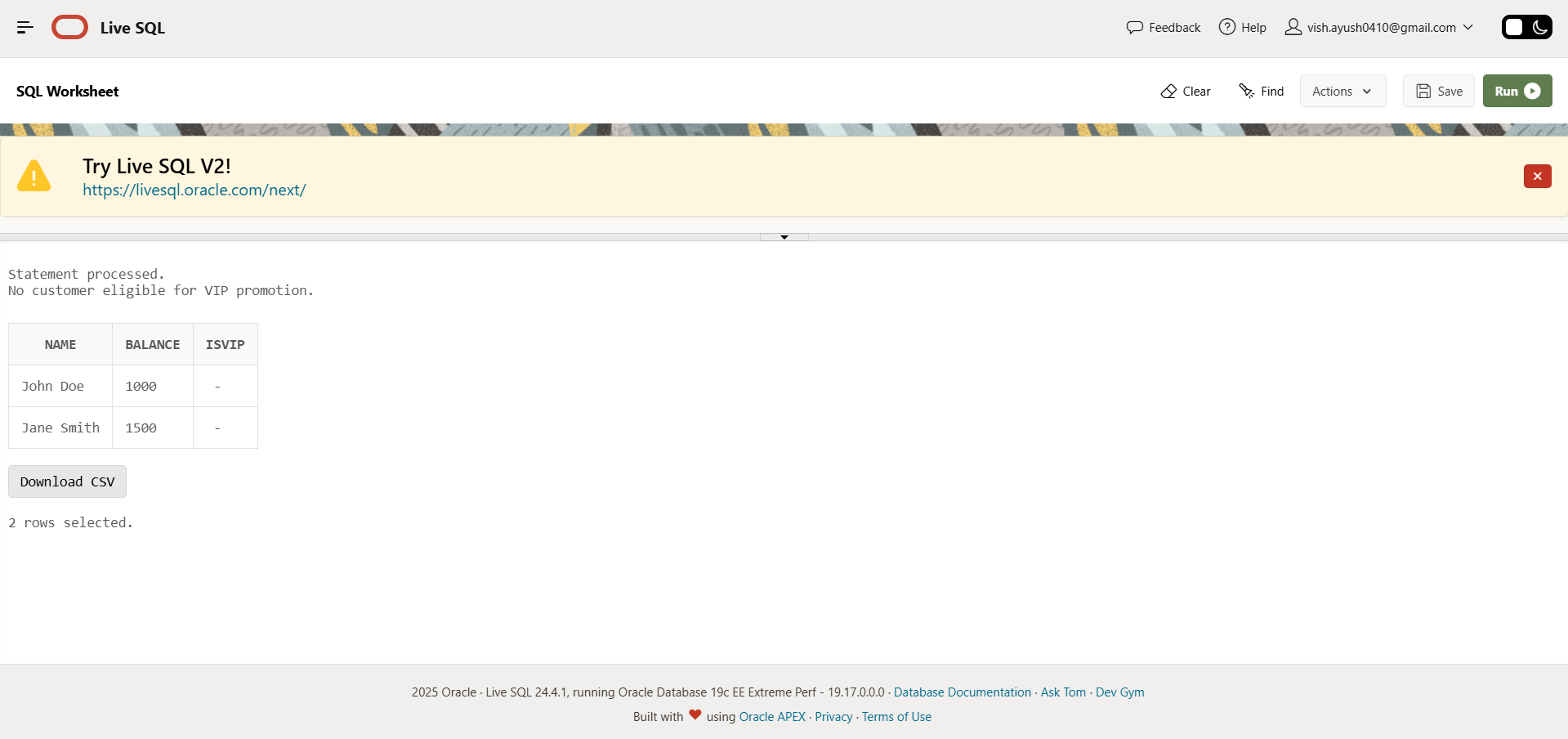
COMMIT;

END;

/

SELECT Name, Balance, IsVIP FROM Customers;

ScreenShot and Output :



**SCENARIO 3 :**

**CODE:**

DECLARE

v\_found BOOLEAN := FALSE;

BEGIN

FOR rec IN (

SELECT l.LoanID, c.Name, l.EndDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

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

' for customer ' || rec.Name || ' is due on ' || TO\_CHAR(rec.EndDate, 'DD-MON-YYYY'));

v\_found := TRUE;

END LOOP;

IF NOT v\_found THEN

DBMS\_OUTPUT.PUT\_LINE('No loans due in the next 30 days.');

END IF;

END;

/

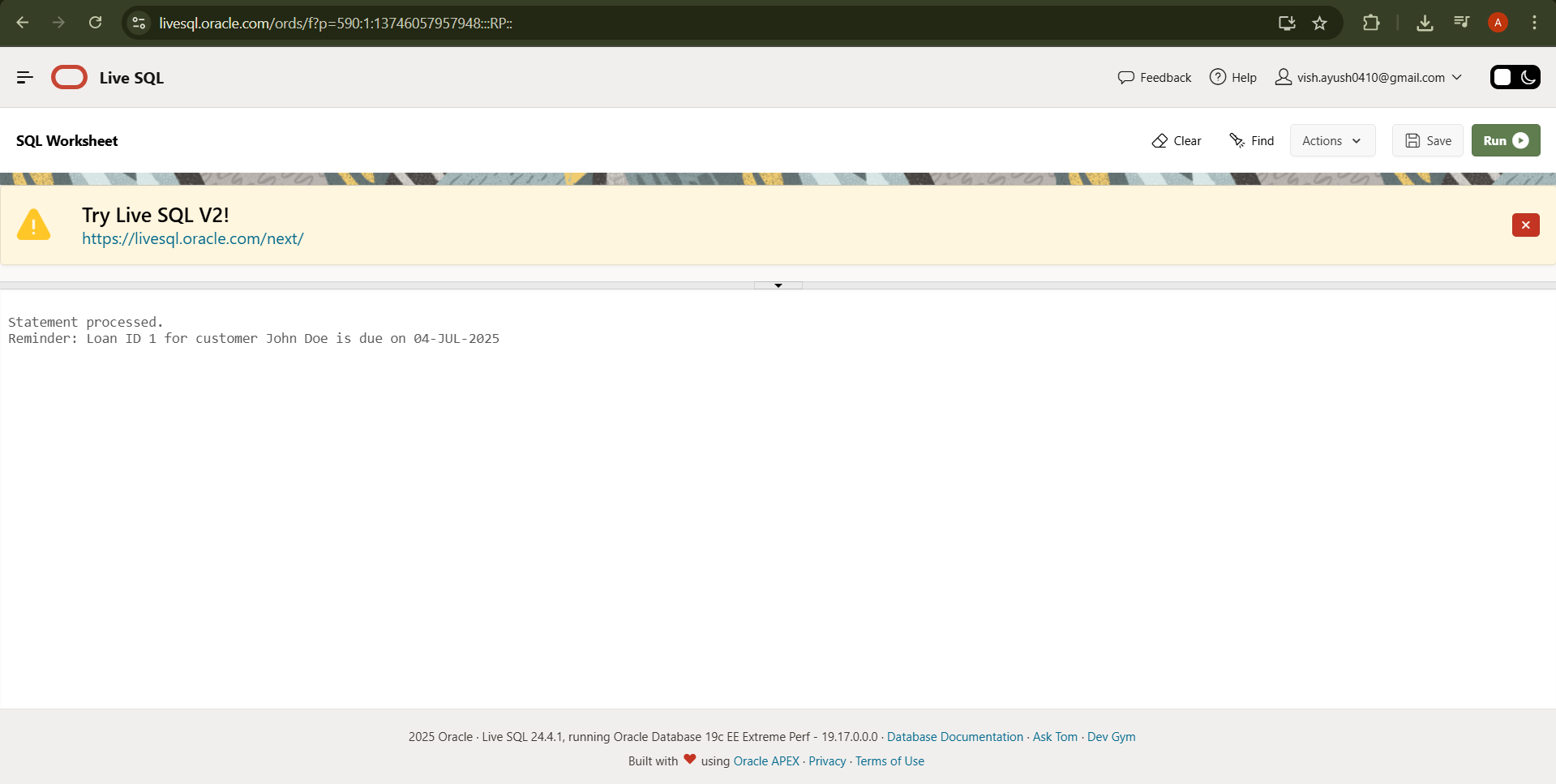
FOR TESTING :

-- Adjust loan to test the reminder

UPDATE Loans SET EndDate = SYSDATE + 5 WHERE LoanID = 1;

COMMIT;

Output and Screenshot :



**SCENARIO 4 :**

**CODE :**

DECLARE

v\_found BOOLEAN := FALSE;

v\_interest NUMBER;

BEGIN

FOR acc IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

v\_interest := acc.Balance \* 0.0033; -- 0.33% interest

UPDATE Accounts

SET Balance = Balance + v\_interest,

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest added to Account ID ' || acc.AccountID ||

'. New balance: ' || TO\_CHAR(acc.Balance + v\_interest, '9999.99'));

v\_found := TRUE;

END LOOP;

IF NOT v\_found THEN

DBMS\_OUTPUT.PUT\_LINE('No savings accounts found for interest calculation.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Interest calculation completed for all savings accounts.');

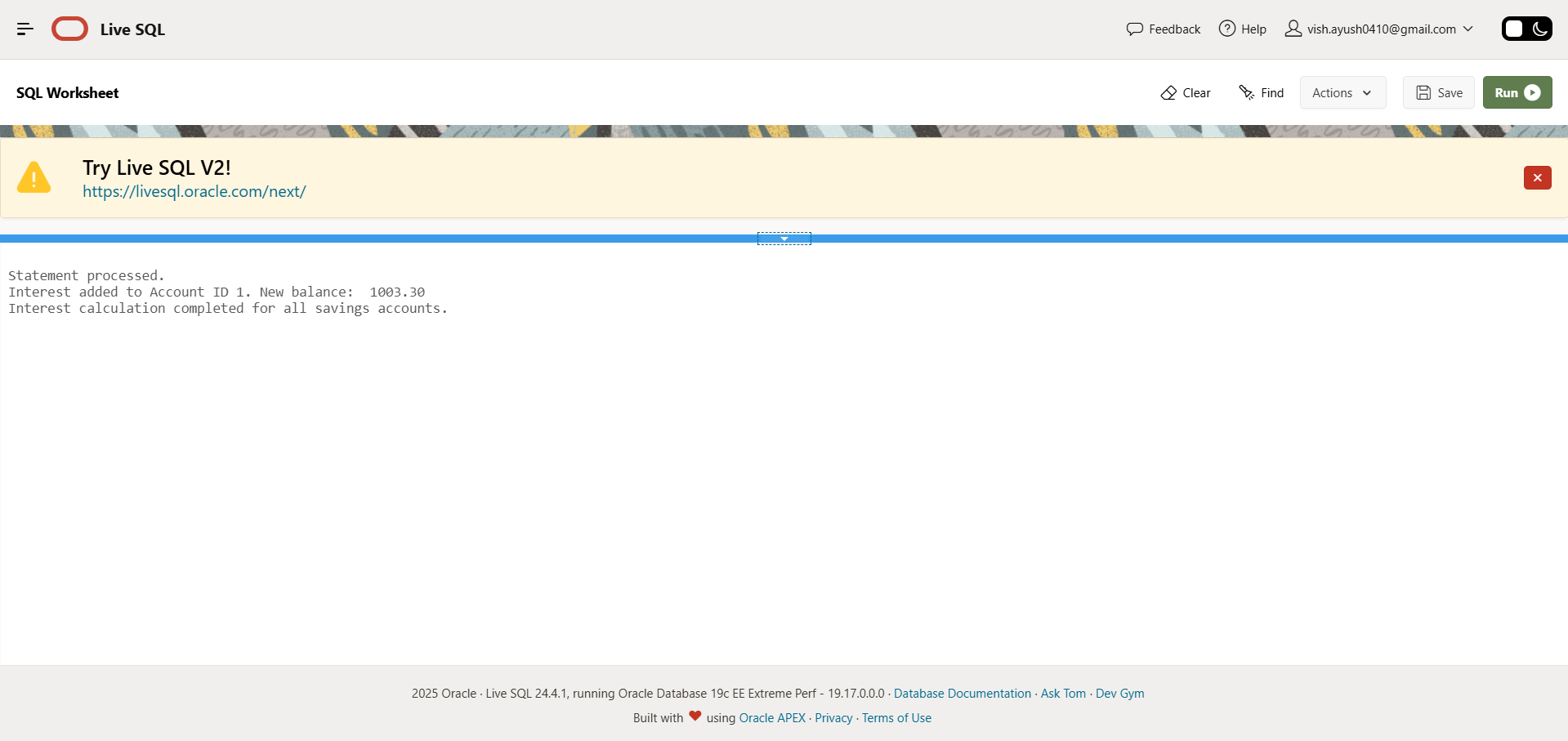
END IF;

COMMIT;

END;

/

Output and ScreenShot :



**02 : Stored Procedures**

**SCENARIO 1 :**

**CODE :**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

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

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Updated Account ID ' || acc.AccountID ||

' | New Balance: ' || TO\_CHAR(acc.Balance \* 1.01, '9999.99'));

END LOOP;

COMMIT;

END;

/

TO RUN PROCEDURE :

BEGIN

ProcessMonthlyInterest;

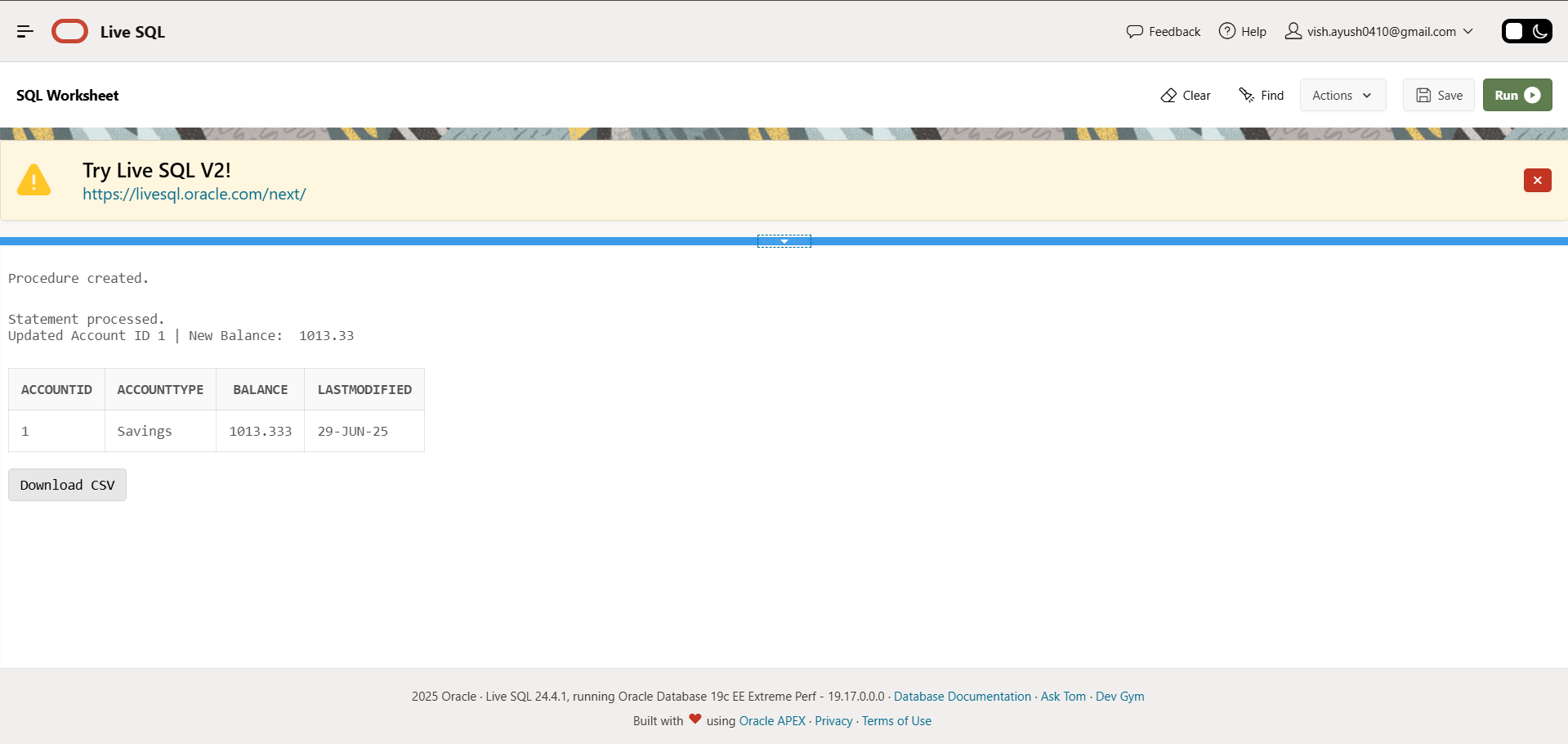
END;

/

TO VERIFY RESULTS :

SELECT AccountID, AccountType, Balance, LastModified FROM Accounts WHERE AccountType = 'Savings';

Output and ScreenShot :



**SCENARIO 2:**

**CODE :**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER

) IS

BEGIN

FOR emp IN (

SELECT EmployeeID, Salary

FROM Employees

WHERE Department = p\_department

) LOOP

UPDATE Employees

SET Salary = Salary + (emp.Salary \* p\_bonus\_pct)

WHERE EmployeeID = emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Updated Employee ID ' || emp.EmployeeID ||

' | New Salary: ' || TO\_CHAR(emp.Salary \* (1 + p\_bonus\_pct), '999999.99'));

END LOOP;

COMMIT;

END;

/

BEGIN

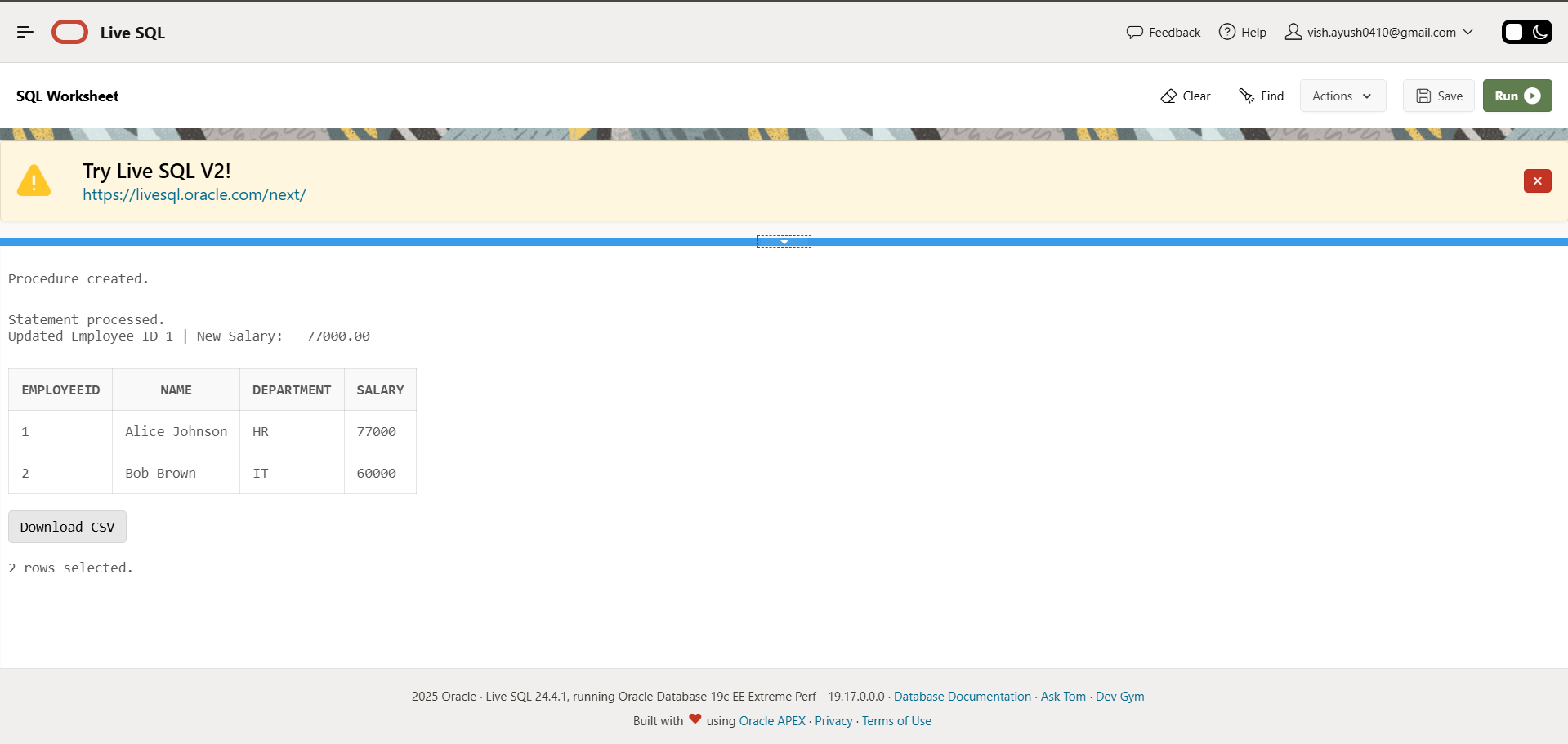
UpdateEmployeeBonus('HR', 0.10); -- 10% bonus to HR department

END;

/

SELECT EmployeeID, Name, Department, Salary FROM Employees;

Output and ScreenShot :



SCENARIO 3 :

**CODE** :

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account\_id IN NUMBER,

p\_to\_account\_id IN NUMBER,

p\_amount IN NUMBER

) IS

v\_source\_balance NUMBER;

BEGIN

-- Fetch the source account balance

SELECT Balance INTO v\_source\_balance

FROM Accounts

WHERE AccountID = p\_from\_account\_id

FOR UPDATE;

-- Check if sufficient balance

IF v\_source\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in account ID ' || p\_from\_account\_id);

ELSE

-- Deduct from source account

UPDATE Accounts

SET Balance = Balance - p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_from\_account\_id;

-- Add to destination account

UPDATE Accounts

SET Balance = Balance + p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_to\_account\_id;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || p\_amount || ' from account ' ||

p\_from\_account\_id || ' to account ' || p\_to\_account\_id);

COMMIT;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: One or both account IDs not found.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: ' || SQLERRM);

ROLLBACK;

END;

/

BEGIN

TransferFunds(1, 2, 500); -- Example: Transfer ₹500 from Account 1 to Account 2

END;

/

SELECT AccountID, Balance FROM Accounts WHERE AccountID IN (1, 2);

Output and ScreenShot :

