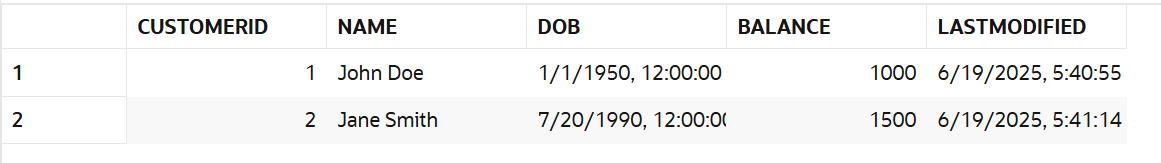
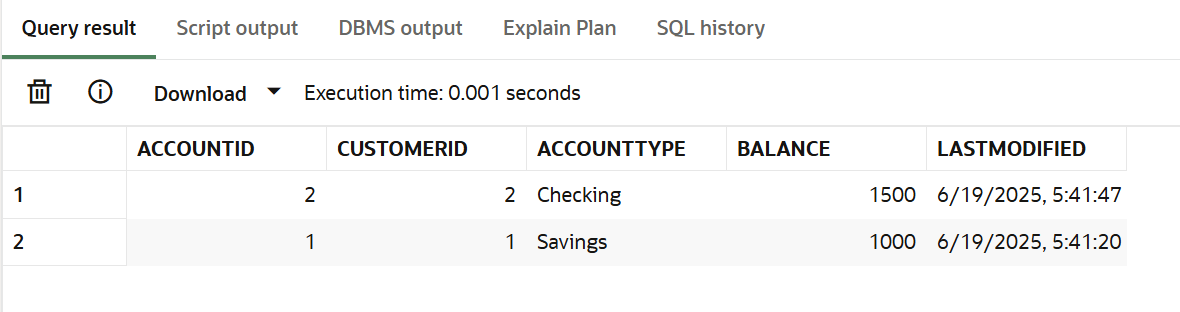
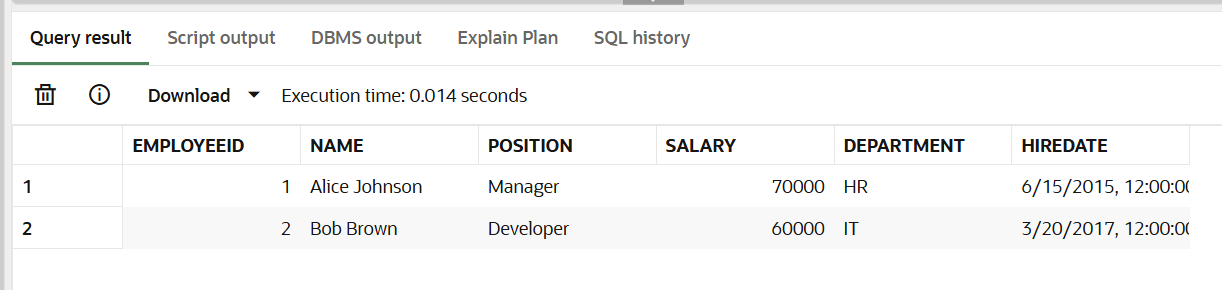
**TABLES TO BE CREATED**  
  
CUSTOMERS TABLE  


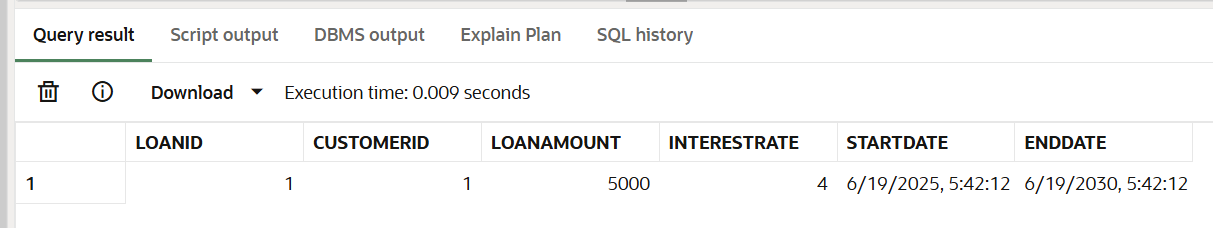
ACCOUNTS TABLE



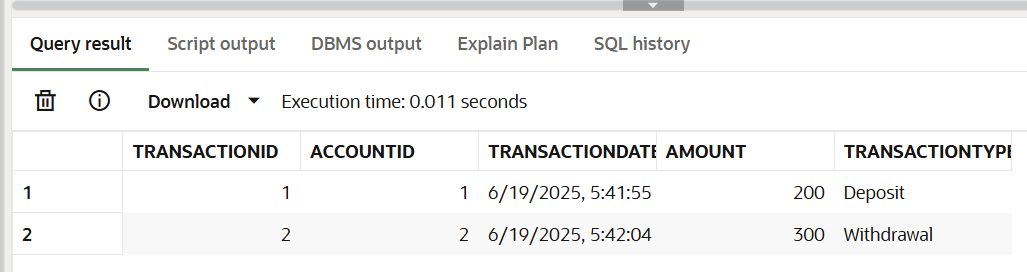
EMPLOYEES TABLE



LOANS TABLE



TRANSACTIONS TABLE



**Exercise-1 Control Structures**

**Question-1**  
Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

**PL/SQL Block:**

BEGIN

    FOR rec IN (

        SELECT l.LoanID, c.DOB

        FROM Loans l

        JOIN Customers c ON l.CustomerID = c.CustomerID

    ) LOOP

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

            UPDATE Loans

            SET InterestRate = InterestRate - 1

            WHERE LoanID = rec.LoanID;

        END IF;

    END LOOP;

    COMMIT;

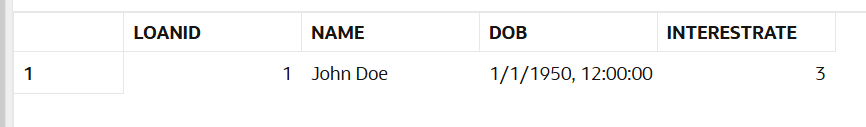
END;

SELECT l.LoanID, c.Name, c.DOB, l.InterestRate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID;

**OUTPUT:**



**Question-2**

Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

**PL/SQL block:**

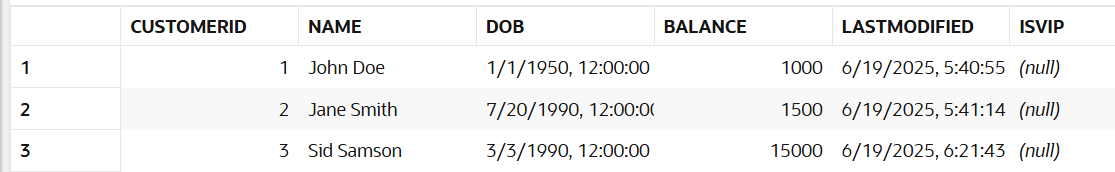
Since there is no IsVIP column in the existing CUSTOMERS table, add it.

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

The below is the loop to make the customer’s IS VIP flag to ‘TRUE’ if their balance>10000. As there is no such customer in the existing table, we will now a row in the table to verify the working of the PL/SQL block.

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (3, 'Sid Samson', TO\_DATE('1990-03-03', 'YYYY-MM-DD'), 15000, SYSDATE);



BEGIN

    FOR rec IN (SELECT CustomerID, Balance FROM Customers) LOOP

        IF rec.Balance > 10000 THEN

            UPDATE Customers

            SET IsVIP = 'TRUE'

            WHERE CustomerID = rec.CustomerID;

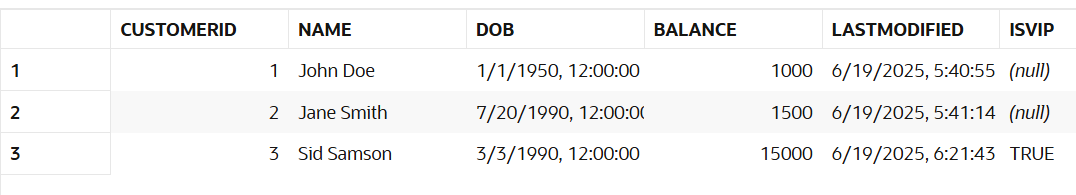
        END IF;

    END LOOP;

    COMMIT;

END;

**OUTPUT:**

After execution of the loop on the CUSTOMER table with a new row.  


**Question-3**

Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**PL/SQL Block:**

Add a row in the LOANS table with an end date due 15 days from now (19-06-2025).

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (2, 2, 8000, 6, SYSDATE, SYSDATE + 15);

PL/SQL block:  
BEGIN

    FOR rec IN (

        SELECT c.Name, l.EndDate

        FROM Loans l

        JOIN Customers c ON l.CustomerID = c.CustomerID

        WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

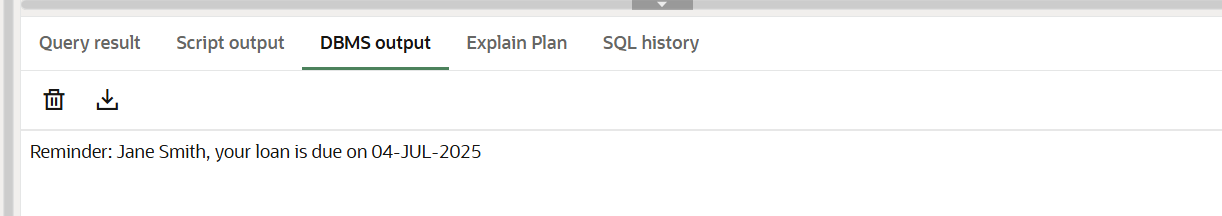
    ) LOOP

        DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || rec.Name || ', your loan is due on ' || TO\_CHAR(rec.EndDate, 'DD-MON-YYYY'));

    END LOOP;

END;

**OUTPUT:**



**Exercise-3 Storage Procedures**

**Question-1:**

Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

**Procedure:**CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    FOR rec IN (

        SELECT AccountID, Balance

        FROM Accounts

        WHERE AccountType = 'Savings'

    ) LOOP

        UPDATE Accounts

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

            LastModified = SYSDATE

        WHERE AccountID = rec.AccountID;

    END LOOP;

    COMMIT;

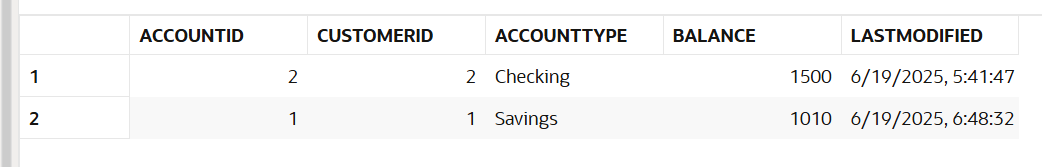
END;

BEGIN

    ProcessMonthlyInterest;

END;

select \* from ACCOUNTS;

**Output:  
**

**Question-2:**

Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**Procedure:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

    p\_department IN VARCHAR2,

    p\_bonus\_pct IN NUMBER  -- e.g., 10 for 10%

) IS

BEGIN

    UPDATE Employees

    SET Salary = Salary + (Salary \* (p\_bonus\_pct / 100))

    WHERE Department = p\_department;

    COMMIT;

END;

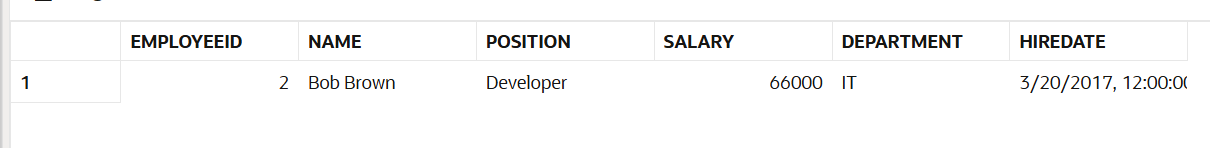
BEGIN

    UpdateEmployeeBonus('IT', 10);

END;

SELECT \* FROM Employees WHERE Department = 'IT';

**Output:**



**Question-3:**Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer. **Procedure:**

CREATE OR REPLACE PROCEDURE TransferFunds(

    p\_from\_account IN NUMBER,

    p\_to\_account IN NUMBER,

    p\_amount IN NUMBER

) IS

    v\_balance NUMBER;

BEGIN

    -- Get balance of source account

    SELECT Balance INTO v\_balance

    FROM Accounts

    WHERE AccountID = p\_from\_account

    FOR UPDATE;

    IF v\_balance < p\_amount THEN

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

    END IF;

    -- Deduct from source account

    UPDATE Accounts

    SET Balance = Balance - p\_amount,

        LastModified = SYSDATE

    WHERE AccountID = p\_from\_account;

    -- Add to destination account

    UPDATE Accounts

    SET Balance = Balance + p\_amount,

        LastModified = SYSDATE

    WHERE AccountID = p\_to\_account;

    COMMIT;

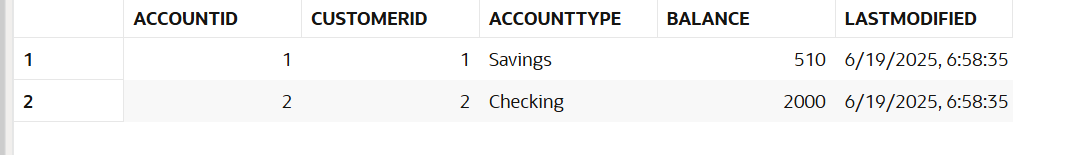
END;

BEGIN

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

END;

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

**Output:**  
****