**Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

* + **Question:** 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.

DECLARE

CURSOR savings\_accounts IS

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings';

BEGIN

FOR acc IN savings\_accounts LOOP

UPDATE Accounts

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

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

END LOOP;

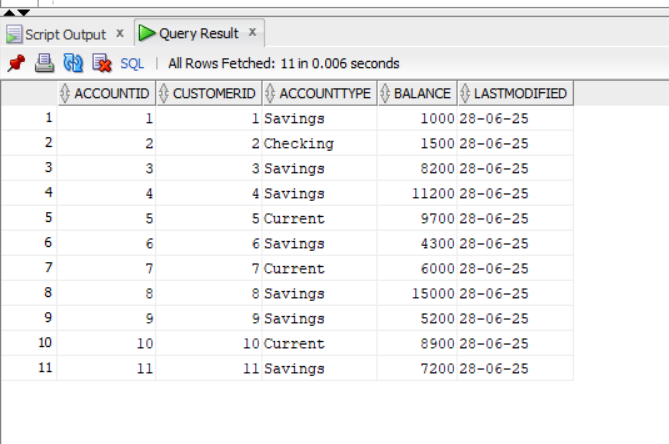
COMMIT;

END;

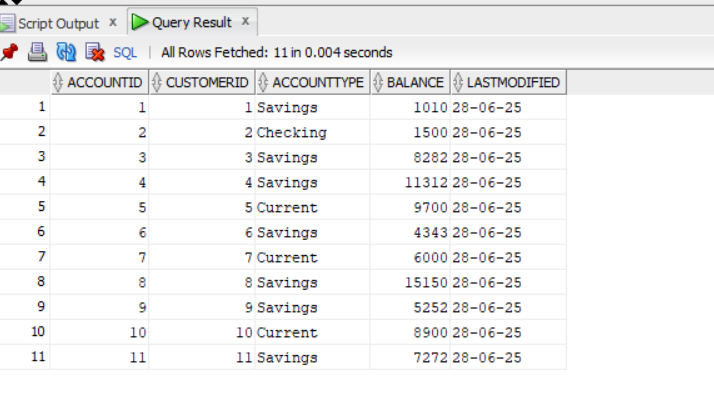
/

Select \* from Accounts;//to see the changes in the balance column in the table

BEFORE:



AFTER:



**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

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

DECLARE

v\_department Employees.Department%TYPE := 'IT'; -- Change as we needed

v\_bonus\_percent NUMBER := 10; -- For example: 10%(can add the bonus based on our requirements)

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* v\_bonus\_percent / 100)

WHERE Department = v\_department;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || v\_bonus\_percent || '% applied to department: ' || v\_department);

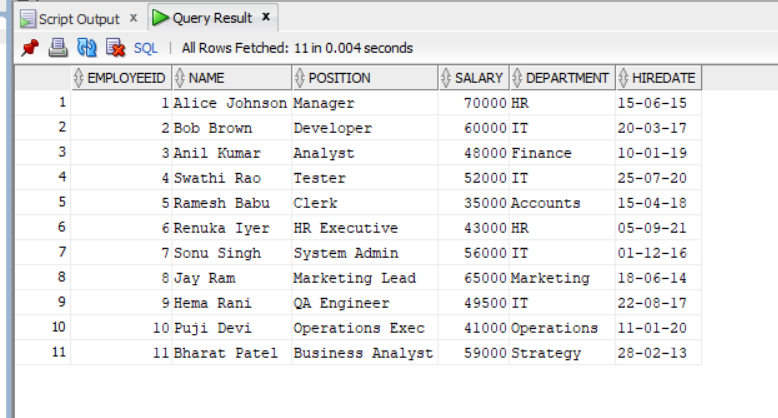
COMMIT;

END;

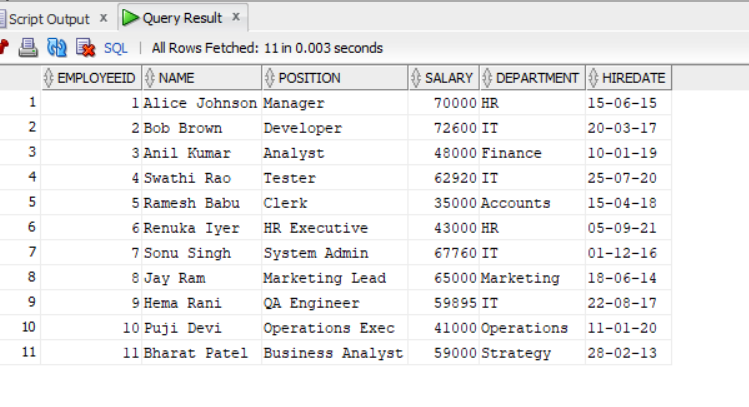
/

Select \* from Employees;//see the change in there salary after adding the bonus percentage

BEFORE:



AFTER:



**Scenario 3:** Customers should be able to transfer funds between their accounts.

* + **Question:** 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.

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_source\_account IN Accounts.AccountID%TYPE,

p\_target\_account IN Accounts.AccountID%TYPE,

p\_amount IN NUMBER

) AS

v\_source\_balance Accounts.Balance%TYPE;

BEGIN

-- Get current balance of source account

SELECT Balance

INTO v\_source\_balance

FROM Accounts

WHERE AccountID = p\_source\_account;

-- Check for sufficient funds

IF v\_source\_balance < p\_amount THEN

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

END IF;

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_source\_account;

-- Add to target

UPDATE Accounts

SET Balance = Balance + p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_target\_account;

COMMIT;

END;

/

OUTPUT:

