Exercise 2:Stored Procedures

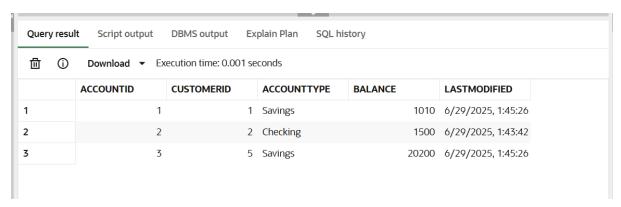
Scenario 1:

Accounts Table:

	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	LASTMODIFIED
1	1	1	Savings	1000	6/29/2025, 1:43:42
2	2	2	Checking	1500	6/29/2025, 1:43:42
3	3	5	Savings	20000	6/29/2025, 1:43:42

Output:

Accounts Table



Scenario 2:

Employee Table:

Query resul	lt Script output	DBMS output Ex	plain Plan SQL his	tory		
ů û	Download ▼ E	xecution time: 0.017 se	econds			
	EMPLOYEEID	NAME	POSITION	SALARY	DEPARTMENT	HIREDATE
1	1	Alice Johnson	Manager	70000	HR	6/15/2015, 12:00
2	2	Bob Brown	Developer	60000	IT	3/20/2017, 12:00

OUTPUT:

Employee Table:

	Script output	DBMS output Exp	olain Plan SQL his	tory		
1 0	Download ▼ Exe	ecution time: 0 second	s			
E	MPLOYEEID	NAME	POSITION	SALARY	DEPARTMENT	HIREDATE
1	1	Alice Johnson	Manager	70000	HR	6/15/2015, 12:00
2	2	Bob Brown	Developer	66000	IT	3/20/2017, 12:00

Scenario 3:

Accounts Table:

Query resul	Query result Script output DBMS output Explain Plan SQL history □ □ Download ▼ Execution time: 0.001 seconds						
	ACCOUNTID	CUSTOMERID		ACCOUNTTYPE	BALANCE		LASTMODIFIED
1	1		1	Savings		1010	6/29/2025, 1:45:26
2	2	2	2	Checking		1500	6/29/2025, 1:43:42
3	3	į	5	Savings		20200	6/29/2025, 1:45:26

OUTPUT:

Query result Script output DBMS output Explain Plan SQL history □ Ownload Execution time: 0.002 seconds						
	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	LASTMODIFIED	
1	1	1	Savings	810	6/29/2025, 1:45:26	
2	2	2	Checking	1700	6/29/2025, 1:43:42	
3	3	5	Savings	20200	6/29/2025, 1:45:26	

```
PL/SQL CODE:
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
BEGIN
  UPDATE Accounts
  SET Balance = Balance * 1.01,
    LastModified = SYSDATE
  WHERE AccountType = 'Savings';
  COMMIT;
END;
select * from ACCOUNTS;
select * from EMPLOYEES;
select * from ACCOUNTS;
-- 6. Stored Procedure: UpdateEmployeeBonus
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
  dept_name IN VARCHAR,
  bonus_percent IN NUMBER
) AS
BEGIN
  UPDATE Employees
  SET Salary = Salary + (Salary * bonus_percent / 100)
  WHERE Department = dept_name;
  COMMIT;
END;
```

```
-- 7. Stored Procedure: TransferFunds
CREATE OR REPLACE PROCEDURE TransferFunds (
  from_account IN NUMBER,
  to_account IN NUMBER,
  amount IN NUMBER
) AS
  insufficient_balance EXCEPTION;
  from_balance NUMBER;
BEGIN
  SELECT Balance INTO from_balance
  FROM Accounts
  WHERE AccountID = from_account
  FOR UPDATE;
  IF from_balance < amount THEN
    RAISE insufficient_balance;
  END IF;
  -- Deduct from source
  UPDATE Accounts
  SET Balance = Balance - amount,
    LastModified = SYSDATE
  WHERE AccountID = from_account;
  -- Add to destination
  UPDATE Accounts
  SET Balance = Balance + amount,
    LastModified = SYSDATE
  WHERE AccountID = to_account;
  -- Log Transactions
  INSERT INTO Transactions VALUES (Transactions_seq.NEXTVAL, from_account, SYSDATE,
amount, 'Debit');
  INSERT INTO Transactions VALUES (Transactions_seq.NEXTVAL, to_account, SYSDATE,
amount, 'Credit');
```

```
COMMIT;
EXCEPTION
  WHEN insufficient_balance THEN
    ROLLBACK;
    DBMS_OUTPUT_LINE('Transfer failed: Insufficient funds.');
  WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Transfer failed: ' || SQLERRM);
END;
-- 8. Test Procedure Calls
-- Run monthly interest calculation
EXEC ProcessMonthlyInterest;
-- Give 10% bonus to IT department
EXEC UpdateEmployeeBonus('IT', 10);
-- Transfer 200 from Account 1 to Account 2
EXEC TransferFunds(1, 2, 200);
```