**COGNIZANT JAVA FSE COURSE WEEK 2**

**NAME:SNEHA LAKSHMI S(4174601)**

PL/SQL EXERCISES

EXERCISE 1: Control Structures

CREATE TABLE CUSTOMERS (

  CustomerID NUMBER PRIMARY KEY,

  Name VARCHAR2(50),

  Age NUMBER,

  Balance NUMBER(10, 2),

  IsVIP CHAR(1) DEFAULT 'N'

);

CREATE TABLE LOANS (

  LoanID NUMBER PRIMARY KEY,

  CustomerID NUMBER,

  InterestRate NUMBER(5, 2),

  DueDate DATE

);

-- Insert some dummy data

INSERT INTO CUSTOMERS VALUES (1, 'Alice', 65, 15000, 'N');

INSERT INTO CUSTOMERS VALUES (2, 'Bob', 45, 5000, 'N');

INSERT INTO CUSTOMERS VALUES (3, 'Charlie', 70, 12000, 'N');

INSERT INTO LOANS VALUES (1, 1, 5.5, SYSDATE + 20);

INSERT INTO LOANS VALUES (2, 2, 6.0, SYSDATE + 40);

INSERT INTO LOANS VALUES (3, 3, 4.5, SYSDATE + 10);

COMMIT;

BEGIN

  FOR rec IN (SELECT CustomerID FROM CUSTOMERS WHERE Age > 60) LOOP

    UPDATE LOANS

    SET InterestRate = InterestRate - 1

    WHERE CustomerID = rec.CustomerID;

  END LOOP;

  COMMIT;

END;

/

BEGIN

  FOR rec IN (SELECT CustomerID FROM CUSTOMERS WHERE Balance > 10000) LOOP

    UPDATE CUSTOMERS

    SET IsVIP = 'Y'

    WHERE CustomerID = rec.CustomerID;

  END LOOP;

  COMMIT;

END;

/

BEGIN

  FOR rec IN (

    SELECT L.LoanID, C.Name, L.DueDate

    FROM LOANS L JOIN CUSTOMERS C

    ON L.CustomerID = C.CustomerID

    WHERE L.DueDate <= SYSDATE + 30

  ) LOOP

    DBMS\_OUTPUT.PUT\_LINE(

      'Reminder: Loan ID ' || rec.LoanID ||

      ' for customer ' || rec.Name ||

      ' is due on ' || TO\_CHAR(rec.DueDate, 'DD-MON-YYYY')

    );

  END LOOP;

END;

/

SELECT \* FROM CUSTOMERS;

SELECT \* FROM LOANS;

OUTPUT:  
Script Output

Table CUSTOMERS created.

Elapsed: 00:00:00.014

Table LOANS created.

Elapsed: 00:00:00.017

1 row inserted.

Elapsed: 00:00:00.013

1 row inserted.

Elapsed: 00:00:00.001

1 row inserted.

Elapsed: 00:00:00.001

1 row inserted.

Elapsed: 00:00:00.012

1 row inserted.

Elapsed: 00:00:00.001

1 row inserted.

Elapsed: 00:00:00.001

Commit complete.

Elapsed: 00:00:00.002

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.086

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.013

Reminder: Loan ID 1 for customer Alice is due on 19-JUL-2025

Reminder: Loan ID 3 for customer Charlie is due on 09-JUL-2025

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.019

LOANID CUSTOMERID INTERESTRATE DUEDATE

------ ---------- ------------ -------------------------

1 1 4.5 07/19/2025, 07:06:14 PM

2 2 6 08/08/2025, 07:06:14 PM

3 3 3.5 07/09/2025, 07:06:14 PM

Elapsed: 00:00:00.008

3 rows selected.

Query Result:

"CUSTOMERID" "NAME" "AGE" "BALANCE" "ISVIP"

1 "Alice" 65 15000 "Y"

2 "Bob" 45 5000 "N"

3 "Charlie" 70 12000 "Y"  
  
  
EXERCISE 2: Stored Procedures

CREATE TABLE ACCOUNTS (

  AccountID NUMBER PRIMARY KEY,

  CustomerID NUMBER,

  AccountType VARCHAR2(20), -- e.g., 'Savings', 'Checking'

  Balance NUMBER(10, 2)

);

-- Insert sample accounts

INSERT INTO ACCOUNTS VALUES (1, 1, 'Savings', 1000);

INSERT INTO ACCOUNTS VALUES (2, 1, 'Checking', 500);

INSERT INTO ACCOUNTS VALUES (3, 2, 'Savings', 2000);

INSERT INTO ACCOUNTS VALUES (4, 2, 'Checking', 1000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE ACCOUNTS

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Interest applied.');

END;

/

BEGIN

ProcessMonthlyInterest;

END;

/

SELECT \* FROM ACCOUNTS;

CREATE TABLE EMPLOYEES (

  EmployeeID NUMBER PRIMARY KEY,

  Name VARCHAR2(50),

  Department VARCHAR2(50),

  Salary NUMBER(10, 2)

);

-- Insert sample employees

INSERT INTO EMPLOYEES VALUES (1, 'Alice', 'Sales', 5000);

INSERT INTO EMPLOYEES VALUES (2, 'Bob', 'Sales', 5500);

INSERT INTO EMPLOYEES VALUES (3, 'Charlie', 'HR', 6000);

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

  p\_Department IN VARCHAR2,

  p\_BonusPct IN NUMBER

) IS

BEGIN

  UPDATE EMPLOYEES

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

  WHERE Department = p\_Department;

  COMMIT;

END;

/

BEGIN

  UpdateEmployeeBonus('Sales', 10);

END;

/

SELECT \* FROM EMPLOYEES;

CREATE OR REPLACE PROCEDURE TransferFunds(

  p\_SourceAccountID IN NUMBER,

  p\_TargetAccountID IN NUMBER,

  p\_Amount IN NUMBER

) IS

  v\_SourceBalance NUMBER;

BEGIN

  -- Get source balance

  SELECT Balance INTO v\_SourceBalance

  FROM ACCOUNTS

  WHERE AccountID = p\_SourceAccountID;

  -- Check if enough

  IF v\_SourceBalance >= p\_Amount THEN

    -- Subtract from source

    UPDATE ACCOUNTS

    SET Balance = Balance - p\_Amount

    WHERE AccountID = p\_SourceAccountID;

    -- Add to target

    UPDATE ACCOUNTS

    SET Balance = Balance + p\_Amount

    WHERE AccountID = p\_TargetAccountID;

    COMMIT;

  ELSE

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

  END IF;

END;

/

BEGIN

  TransferFunds(1, 2, 200);

END;

/

SELECT \* FROM ACCOUNTS;

OUTPUT:

SQL> COMMIT

Commit complete.  
  
Elapsed: 00:00:00.001

SQL> CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS  
BEGIN  
UPDATE ACCOUNTS  
SET Balance = Balance + (Balance \* 0.01)...

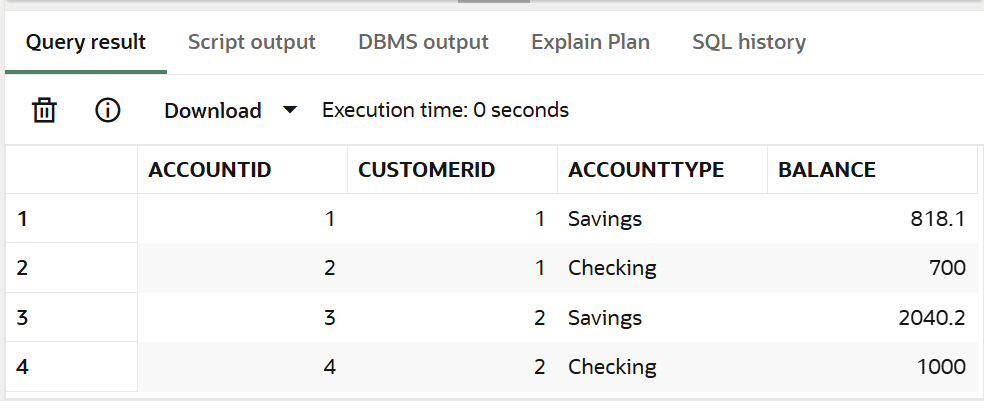
Show more...

Procedure PROCESSMONTHLYINTEREST compiled  
  
Elapsed: 00:00:00.004

SQL> BEGIN  
ProcessMonthlyInterest;  
END;

Interest applied.  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.008

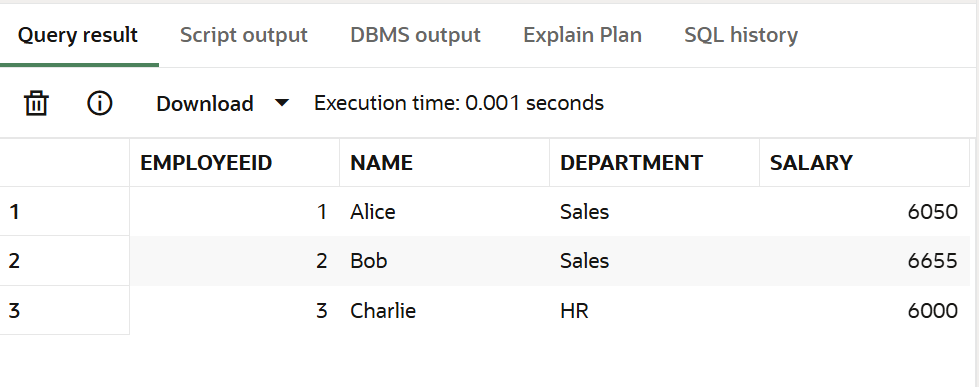
QUERY RESULT:



Procedure UPDATEEMPLOYEEBONUS compiled  
  
Elapsed: 00:00:00.004

SQL> BEGIN  
UpdateEmployeeBonus('Sales', 10);  
END;

PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.011



Procedure TRANSFERFUNDS compiled  
  
Elapsed: 00:00:00.004

SQL> BEGIN  
TransferFunds(1, 2, 200);  
END;

PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.012

