**Week2**

**PL/SQL programming**

**Exercise 1: Control Structures**

**Query:**

**Code:**

**Schema.sql**

CREATE TABLE customer$ (

  customer\_id    NUMBER PRIMARY KEY,

  customer\_name  VARCHAR2(100),

  age            NUMBER,

  balance        NUMBER,

  is\_vip         VARCHAR2(5)

);

CREATE TABLE loan$ (

  loan\_id        NUMBER PRIMARY KEY,

  customer\_id    NUMBER REFERENCES customer$(customer\_id),

  interest\_rate  NUMBER,

  due\_date       DATE

);

**Seed\_data.sql**

INSERT INTO customer$ VALUES (1, 'Alice Johnson', 65, 12000, 'FALSE');

INSERT INTO customer$ VALUES (2, 'Bob Smith', 45, 8000, 'FALSE');

INSERT INTO customer$ VALUES (3, 'Carol Lee', 70, 15000, 'FALSE');

INSERT INTO customer$ VALUES (4, 'David Kim', 30, 9500, 'FALSE');

-- Insert into loan$ table

INSERT INTO loan$ VALUES (101, 1, 8.5, TO\_DATE('2025-07-06', 'YYYY-MM-DD'));

INSERT INTO loan$ VALUES (102, 2, 7.0, TO\_DATE('2025-08-10', 'YYYY-MM-DD'));

INSERT INTO loan$ VALUES (103, 3, 9.0, TO\_DATE('2025-07-16', 'YYYY-MM-DD'));

INSERT INTO loan$ VALUES (104, 4, 6.5, TO\_DATE('2025-07-21', 'YYYY-MM-DD'));

COMMIT;

**Scenario 1: Apply 1% Discount for Age > 60 Query for scenario 1:**

BEGIN

-- Apply 1% discount for senior customers

FOR c IN (SELECT customer\_id FROM customer$ WHERE age > 60) LOOP

UPDATE loan$

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = c.customer\_id;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('✅ Scenario 1: Interest discount applied.');

DBMS\_OUTPUT.PUT\_LINE('📋 Updated loan$ Table:');

DBMS\_OUTPUT.PUT\_LINE('LoanID | CustomerID | Interest | Due Date');

DBMS\_OUTPUT.PUT\_LINE('-------+------------+----------+------------');

FOR l IN (

SELECT loan\_id, customer\_id, interest\_rate, TO\_CHAR(due\_date, 'DD-Mon-YYYY') AS due\_date FROM loan$

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

LPAD(l.loan\_id, 6) || ' | ' ||

LPAD(l.customer\_id, 10) || ' | ' ||

LPAD(l.interest\_rate || '%', 8) || ' | ' ||

l.due\_date

);

END LOOP;

END;

/

**OUTPUT:**



**Scenario 2: Promote to VIP if Balance > $10,000:**

**BEGIN**

**FOR c IN (SELECT customer\_id FROM customer$ WHERE balance > 10000) LOOP**

**UPDATE customer$**

**SET is\_vip = 'TRUE'**

**WHERE customer\_id = c.customer\_id;**

**END LOOP;**

**COMMIT;**

**DBMS\_OUTPUT.PUT\_LINE('✅ Scenario 2: VIP status updated.');**

**DBMS\_OUTPUT.PUT\_LINE('📋 Updated customer$ Table:');**

**DBMS\_OUTPUT.PUT\_LINE('ID | Name | Age | Balance | IsVIP');**

**DBMS\_OUTPUT.PUT\_LINE('---+----------------+-----+----------+-------');**

**FOR c IN (**

**SELECT customer\_id, customer\_name, age, balance, is\_vip FROM customer$**

**) LOOP**

**DBMS\_OUTPUT.PUT\_LINE(**

**LPAD(c.customer\_id, 2) || ' | ' ||**

**RPAD(c.customer\_name, 14) || ' | ' ||**

**LPAD(c.age, 3) || ' | ' ||**

**LPAD(TO\_CHAR(c.balance, '99999.99'), 8) || ' | ' ||**

**RPAD(c.is\_vip, 5)**

**);**

**END LOOP;**

**END;**

**/**

**OUTPUT:**



**Scenario 3: Print Reminders for Loans Due in 30 Days**

BEGIN

DBMS\_OUTPUT.PUT\_LINE('✅ Scenario 3: Loan Reminders (Due in 30 Days)');

DBMS\_OUTPUT.PUT\_LINE('LoanID | Customer Name | Due Date');

DBMS\_OUTPUT.PUT\_LINE('-------+------------------+------------');

FOR l IN (

SELECT loan\_id, customer\_id, TO\_CHAR(due\_date, 'DD-Mon-YYYY') AS due\_date

FROM loan$

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DECLARE

cust\_name customer$.customer\_name%TYPE;

BEGIN

SELECT customer\_name INTO cust\_name FROM customer$ WHERE customer\_id = l.customer\_id;

DBMS\_OUTPUT.PUT\_LINE(

LPAD(l.loan\_id, 6) || ' | ' ||

RPAD(cust\_name, 16) || ' | ' ||

l.due\_date

);

END;

END LOOP;

END;

/  
**OUTPUT:**



**Exercise 3: Stored Procedures**

**Table creation:**

-- Create savings accounts table

CREATE TABLE account$ (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

balance NUMBER,

account\_type VARCHAR2(20) -- 'SAVINGS' or 'CURRENT'

);

-- Create employee table

CREATE TABLE employee$ (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER

);

**INSERT VALUES:**

-- Insert accounts

INSERT INTO account$ VALUES (1, 'Alice', 10000, 'SAVINGS');

INSERT INTO account$ VALUES (2, 'Bob', 8000, 'SAVINGS');

INSERT INTO account$ VALUES (3, 'Carol', 5000, 'CURRENT');

INSERT INTO account$ VALUES (4, 'David', 15000, 'SAVINGS');

-- Insert employees

INSERT INTO employee$ VALUES (101, 'John', 'HR', 40000);

INSERT INTO employee$ VALUES (102, 'Sara', 'HR', 42000);

INSERT INTO employee$ VALUES (103, 'Mike', 'IT', 50000);

INSERT INTO employee$ VALUES (104, 'Nina', 'IT', 55000);

COMMIT;

**Scenario 1: ProcessMonthlyInterest for SAVINGS Accounts**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

-- Apply 1% interest to savings accounts

UPDATE account$

SET balance = balance + (balance \* 0.01)

WHERE account\_type = 'SAVINGS';

COMMIT;

-- Print updated account$ table

DBMS\_OUTPUT.PUT\_LINE('✅ ProcessMonthlyInterest: Applied 1% interest to savings accounts.');

DBMS\_OUTPUT.PUT\_LINE('ID | Name | Balance | Type');

DBMS\_OUTPUT.PUT\_LINE('---+--------+-----------+--------');

FOR acc IN (

SELECT account\_id, customer\_name, balance, account\_type FROM account$

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

LPAD(acc.account\_id, 2) || ' | ' ||

RPAD(acc.customer\_name, 6) || ' | ' ||

LPAD(TO\_CHAR(acc.balance, '99999.99'), 9) || ' | ' ||

acc.account\_type

);

END LOOP;

END;

/

**Scenario 2: UpdateEmployeeBonus Based on Department & Bonus %**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER -- e.g., 0.10 for 10%

) IS

BEGIN

-- Apply bonus to matching department

UPDATE employee$

SET salary = salary + (salary \* p\_bonus\_pct)

WHERE department = p\_department;

COMMIT;

-- Print updated employee$ table

DBMS\_OUTPUT.PUT\_LINE('✅ UpdateEmployeeBonus: Applied bonus to ' || p\_department || ' department.');

DBMS\_OUTPUT.PUT\_LINE('ID | Name | Dept | Salary');

DBMS\_OUTPUT.PUT\_LINE('----+-------+--------+--------');

FOR emp IN (

SELECT emp\_id, emp\_name, department, salary FROM employee$

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

LPAD(emp.emp\_id, 3) || ' | ' ||

RPAD(emp.emp\_name, 6) || ' | ' ||

RPAD(emp.department, 6) || ' | ' ||

emp.salary

);

END LOOP;

END;

/

**Scenario 3: TransferFunds from One Account to Another**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Check if source has enough balance

SELECT balance INTO v\_balance FROM account$ WHERE account\_id = p\_from\_account;

IF v\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('❌ Transfer failed: Insufficient balance.');

RETURN;

END IF;

-- Perform transfer

UPDATE account$ SET balance = balance - p\_amount WHERE account\_id = p\_from\_account;

UPDATE account$ SET balance = balance + p\_amount WHERE account\_id = p\_to\_account;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('✅ TransferFunds: Transferred ' || p\_amount || ' from Account ' || p\_from\_account || ' to Account ' || p\_to\_account);

-- Print updated accounts

DBMS\_OUTPUT.PUT\_LINE('ID | Name | Balance | Type');

DBMS\_OUTPUT.PUT\_LINE('---+--------+-----------+--------');

FOR acc IN (

SELECT account\_id, customer\_name, balance, account\_type FROM account$

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

LPAD(acc.account\_id, 2) || ' | ' ||

RPAD(acc.customer\_name, 6) || ' | ' ||

LPAD(TO\_CHAR(acc.balance, '99999.99'), 9) || ' | ' ||

acc.account\_type

);

END LOOP;

END;

/

**call\_scenarios.sql (To Test Everything):**

BEGIN

ProcessMonthlyInterest;

DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------');

UpdateEmployeeBonus('HR', 0.10);

DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------');

TransferFunds(1, 2, 1000);

END;

/

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.