**Cognizant Java FSE – (Deep Skilling)**

**(WEEK-2)**

**MODULE 1: PL/SQL programming**

**Submitted by**

Name: S Subhashri

Roll No: 111522102152

Email: 22102152@rmd.ac.in

College: RMD ENGINEERING COLLEGE

Batch: Java FSE – 2026

Superset ID: 6397078

* **PL/SQL Exercises:**

**Exercise 1: Control Structures**

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* 1. Question: 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.

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* 1. Question: 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.

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

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

**Solution:**

**Scenario 1: Apply 1% interest discount for customers above 60 years**

DECLARE

CURSOR cust\_cur IS

SELECT customer\_id, age, loan\_interest

FROM customers;

BEGIN

FOR cust\_rec IN cust\_cur LOOP

IF cust\_rec.age > 60 THEN

UPDATE customers

SET loan\_interest = loan\_interest - 1

WHERE customer\_id = cust\_rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Discount applied to customer ID: ' || cust\_rec.customer\_id);

ELSE

DBMS\_OUTPUT.PUT\_LINE('No discount for customer ID: ' || cust\_rec.customer\_id);

END IF;

END LOOP;

COMMIT;

END;

/

**Explanation:**

* We loop through all customers.
* If a customer is over 60, we reduce their LoanInterestRate by 1%.
* UPDATE is used inside the loop to modify the specific customer.
* COMMIT saves the changes.

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 2: Set IsVIP = TRUE for customers with balance > 10,000**

DECLARE

CURSOR cust\_cur IS

SELECT customer\_id, balance

FROM customers;

BEGIN

FOR cust\_rec IN cust\_cur LOOP

IF cust\_rec.balance > 10000 THEN

UPDATE customers

SET is\_vip = 'TRUE'

WHERE customer\_id = cust\_rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || cust\_rec.customer\_id || ' promoted to VIP.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || cust\_rec.customer\_id || ' not eligible for VIP.');

END IF;

END LOOP;

COMMIT;

END;

/

**Output**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Scenario 3: Print reminder messages for loans due in 30 days**

DECLARE

CURSOR loan\_cur IS

SELECT l.loan\_id, l.due\_date, l.amount, c.name

FROM loans l

JOIN customers c ON l.customer\_id = c.customer\_id

WHERE l.due\_date BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR loan\_rec IN loan\_cur LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || loan\_rec.name ||

', your loan (ID: ' || loan\_rec.loan\_id ||

') of $' || loan\_rec.amount ||

' is due on ' || TO\_CHAR(loan\_rec.due\_date, 'DD-Mon-YYYY') || '.');

END LOOP;

END;

/

**OUTPUT**

**A screenshot of a computer

AI-generated content may be incorrect.**

**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.

**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.

**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.

**Solution:**

**Scenario 1: ProcessMonthlyInterest Procedure**

CREATE TABLE savings\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER(10,2)

);

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest processed for all savings accounts.');

END;

/

SET SERVEROUTPUT ON;

EXEC ProcessMonthlyInterest;

**OUTPUT**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Scenario 2: UpdateEmployeeBonus Procedure**

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

balance NUMBER(10,2),

is\_vip VARCHAR2(5)

);

INSERT INTO customers VALUES (1, 'Ravi', 12000, 'FALSE');

INSERT INTO customers VALUES (2, 'Meena', 9500, 'FALSE');

INSERT INTO customers VALUES (3, 'Anil', 15000, 'FALSE');

COMMIT;

SET SERVEROUTPUT ON;

DECLARE

CURSOR cust\_cur IS

SELECT customer\_id, balance

FROM customers;

BEGIN

FOR cust\_rec IN cust\_cur LOOP

IF cust\_rec.balance > 10000 THEN

UPDATE customers

SET is\_vip = 'TRUE'

WHERE customer\_id = cust\_rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || cust\_rec.customer\_id || ' promoted to VIP.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || cust\_rec.customer\_id || ' not eligible for VIP.');

END IF;

END LOOP;

COMMIT;

END;

/

**Output**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Scenario 3: TransferFunds Procedure**

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

due\_date DATE,

amount NUMBER(10,2)

);

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100)

);

INSERT INTO customers VALUES (1, 'Ravi');

INSERT INTO customers VALUES (2, 'Meena');

INSERT INTO customers VALUES (3, 'Anil');

INSERT INTO loans VALUES (101, 1, SYSDATE + 10, 50000);

INSERT INTO loans VALUES (102, 2, SYSDATE + 40, 30000);

INSERT INTO loans VALUES (103, 3, SYSDATE + 5, 25000);

COMMIT;

SET SERVEROUTPUT ON;

DECLARE

CURSOR loan\_cur IS

SELECT l.loan\_id, l.due\_date, l.amount, c.name

FROM loans l

JOIN customers c ON l.customer\_id = c.customer\_id

WHERE l.due\_date BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR loan\_rec IN loan\_cur LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || loan\_rec.name ||

', your loan (ID: ' || loan\_rec.loan\_id ||

') of $' || loan\_rec.amount ||

' is due on ' || TO\_CHAR(loan\_rec.due\_date, 'DD-Mon-YYYY') || '.');

END LOOP;

END;

/

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.