Week-2

(PLSQL)

Exercise 1 – Control Structures

**Creating Tables-**

CREATE TABLE CUSTOMERS (

CUSTOMERID NUMBER PRIMARY KEY,

NAME VARCHAR2(100),

DOB DATE,

BALANCE NUMBER,

LASTMODIFIED DATE

);

CREATE TABLE ACCOUNTS (

ACCOUNTID NUMBER PRIMARY KEY,

CUSTOMERID NUMBER,

ACCOUNTTYPE VARCHAR2(20),

BALANCE NUMBER,

LASTMODIFIED DATE,

FOREIGN KEY ( CUSTOMERID )

REFERENCES CUSTOMERS ( CUSTOMERID )

);

CREATE TABLE TRANSACTIONS (

TRANSACTIONID NUMBER PRIMARY KEY,

ACCOUNTID NUMBER,

TRANSACTIONDATE DATE,

AMOUNT NUMBER,

TRANSACTIONTYPE VARCHAR2(10),

FOREIGN KEY ( ACCOUNTID )

REFERENCES ACCOUNTS ( ACCOUNTID )

);

CREATE TABLE LOANS (

LOANID NUMBER PRIMARY KEY,

CUSTOMERID NUMBER,

LOANAMOUNT NUMBER,

INTERESTRATE NUMBER,

STARTDATE DATE,

ENDDATE DATE,

FOREIGN KEY ( CUSTOMERID )

REFERENCES CUSTOMERS ( CUSTOMERID )

);

CREATE TABLE EMPLOYEES (

EMPLOYEEID NUMBER PRIMARY KEY,

NAME VARCHAR2(100),

POSITION VARCHAR2(50),

SALARY NUMBER,

DEPARTMENT VARCHAR2(50),

HIREDATE DATE

);

**Inserting into CUSTOMERS**

INSERT INTO CUSTOMERS (CUSTOMERID, NAME, DOB, BALANCE, LASTMODIFIED)

VALUES (1, 'Ravi Verma', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

INSERT INTO CUSTOMERS (CUSTOMERID, NAME, DOB, BALANCE, LASTMODIFIED)

VALUES (2, 'Sneha Sharma', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

**Inserting into ACCOUNTS**

INSERT INTO ACCOUNTS (ACCOUNTID, CUSTOMERID, ACCOUNTTYPE, BALANCE, LASTMODIFIED)

VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO ACCOUNTS (ACCOUNTID, CUSTOMERID, ACCOUNTTYPE, BALANCE, LASTMODIFIED)

VALUES (2, 2, 'Checking', 1500, SYSDATE);

**Inserting into TRANSACTIONS**

INSERT INTO TRANSACTIONS (TRANSACTIONID, ACCOUNTID, TRANSACTIONDATE, AMOUNT, TRANSACTIONTYPE)

VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO TRANSACTIONS (TRANSACTIONID, ACCOUNTID, TRANSACTIONDATE, AMOUNT, TRANSACTIONTYPE)

VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

**Inserting into LOANS**

INSERT INTO LOANS (LOANID, CUSTOMERID, LOANAMOUNT, INTERESTRATE, STARTDATE, ENDDATE)

VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

**Inserting into EMPLOYEES**

INSERT INTO EMPLOYEES (EMPLOYEEID, NAME, POSITION, SALARY, DEPARTMENT, HIREDATE)

VALUES (1, 'Amit Joshi', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO EMPLOYEES (EMPLOYEEID, NAME, POSITION, SALARY, DEPARTMENT, HIREDATE)

VALUES (2, 'Neha Kapoor', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

ALTER TABLE CUSTOMERS

ADD ISVIP CHAR(10)

CHECK (ISVIP IN ('TRUE', 'FALSE'));

Scenerio-1

SELECT \* FROM CUSTOMERS;

SELECT \* FROM LOANS;

SET SERVEROUTPUT ON;

DECLARE

CURSOR CUSTOMER\_CURSOR IS

SELECT CUSTOMERID, EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM DOB) AS AGE

FROM CUSTOMERS;

VAR\_CUSTOMER\_ID CUSTOMERS.CUSTOMERID%TYPE;

VAR\_AGE NUMBER;

BEGIN

FOR CUSTOMER\_RECORD IN CUSTOMER\_CURSOR LOOP

VAR\_CUSTOMER\_ID := CUSTOMER\_RECORD.CUSTOMERID;

VAR\_AGE := CUSTOMER\_RECORD.AGE;

IF VAR\_AGE > 60 THEN

UPDATE LOANS

SET INTERESTRATE = INTERESTRATE - 1

WHERE CUSTOMERID = VAR\_CUSTOMER\_ID;

ELSE

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER WITH CUSTOMER ID : ' || VAR\_CUSTOMER\_ID || ' IS OF AGE : ' || VAR\_AGE);

DBMS\_OUTPUT.PUT\_LINE('NO CHANGE IN LOAN');

END IF;

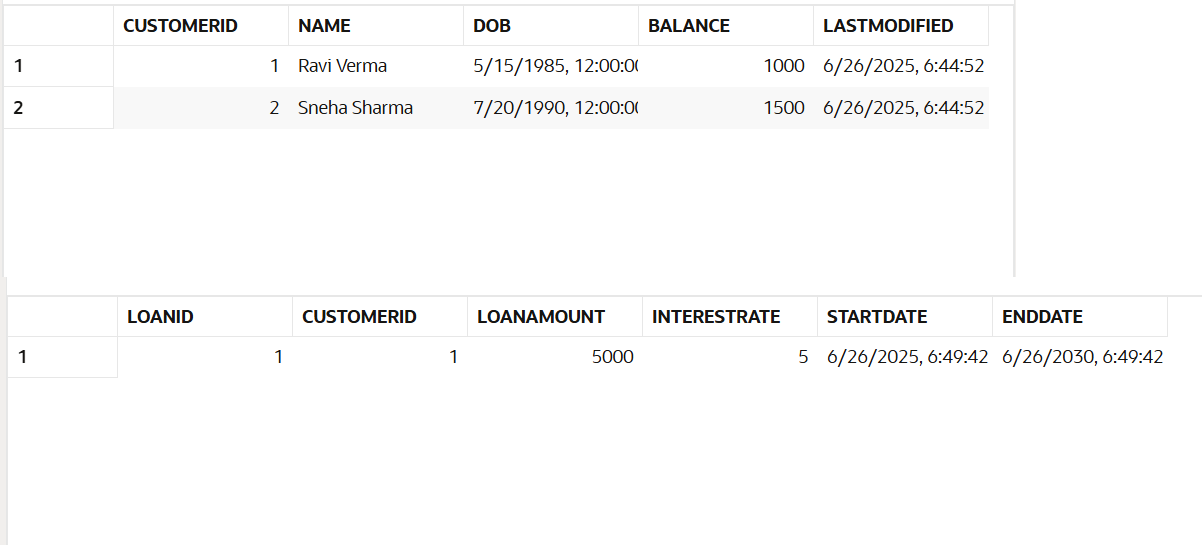
END LOOP;

COMMIT;

END;

/

SELECT \* FROM LOANS;



Scenerio-2

DESC CUSTOMERS;

ALTER TABLE CUSTOMERS ADD ISVIP CHAR(10) CONSTRAINT CHK1 CHECK(ISVIP IN ('TRUE','FALSE')) ;

SELECT \* FROM CUSTOMERS;

SET SERVEROUTPUT ON;

DECLARE

CURSOR CUSTOMER\_CURSOR IS

SELECT CUSTOMERID, BALANCE

FROM CUSTOMERS;

VAR\_CUSTOMER\_ID CUSTOMERS.CUSTOMERID%TYPE;

VAR\_BALANCE CUSTOMERS.BALANCE%TYPE;

BEGIN

FOR CUSTOMER\_RECORD IN CUSTOMER\_CURSOR LOOP

VAR\_CUSTOMER\_ID := CUSTOMER\_RECORD.CUSTOMERID;

VAR\_BALANCE := CUSTOMER\_RECORD.BALANCE;

IF VAR\_BALANCE > 10000 THEN

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER ID : ' || VAR\_CUSTOMER\_ID || ' HAS BALANCE GREATER THAN 10000');

UPDATE CUSTOMERS

SET ISVIP = 'TRUE'

WHERE CUSTOMERID = VAR\_CUSTOMER\_ID;

ELSE

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER ID : ' || VAR\_CUSTOMER\_ID || ' HAS BALANCE LESSER THAN 10000');

UPDATE CUSTOMERS

SET ISVIP = 'FALSE'

WHERE CUSTOMERID = VAR\_CUSTOMER\_ID;

END IF;

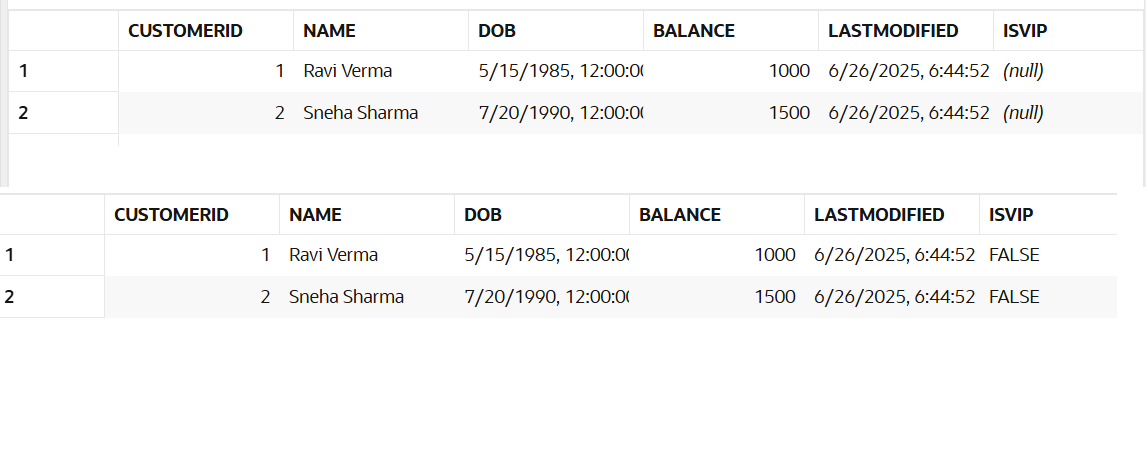
END LOOP;

COMMIT;

END;

/

SELECT \* FROM CUSTOMERS;



Scenerio-3

SET SERVEROUTPUT ON;

DECLARE

CURSOR CUR\_LOANS IS

SELECT L.LOANID, L.CUSTOMERID, C.NAME, L.ENDDATE

FROM LOANS L

JOIN CUSTOMERS C ON L.CUSTOMERID = C.CUSTOMERID

WHERE L.ENDDATE BETWEEN SYSDATE AND SYSDATE + 30;

V\_LOAN\_ID LOANS.LOANID%TYPE;

V\_CUSTOMER\_ID LOANS.CUSTOMERID%TYPE;

V\_CUSTOMER\_NAME CUSTOMERS.NAME%TYPE;

V\_END\_DATE LOANS.ENDDATE%TYPE;

V\_FOUND BOOLEAN := FALSE;

BEGIN

OPEN CUR\_LOANS;

LOOP

FETCH CUR\_LOANS INTO V\_LOAN\_ID, V\_CUSTOMER\_ID, V\_CUSTOMER\_NAME, V\_END\_DATE;

EXIT WHEN CUR\_LOANS%NOTFOUND;

V\_FOUND := TRUE;

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ' || V\_LOAN\_ID || ' for customer ' || V\_CUSTOMER\_NAME || ' (ID: ' || V\_CUSTOMER\_ID || ') is due on ' || TO\_CHAR(V\_END\_DATE, 'YYYY-MM-DD'));

END LOOP;

CLOSE CUR\_LOANS;

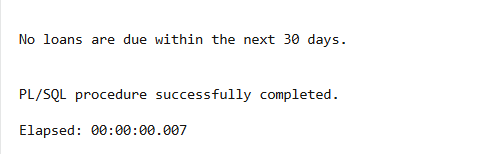
IF NOT V\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No loans are due within the next 30 days.');

END IF;

END;

/



Exercise 3-Stored Procedure

**Enabling DBMS\_OUTPUT**

SET SERVEROUTPUT ON;

**Creating Tables**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE,

IsVIP CHAR(10) CHECK (IsVIP IN ('TRUE', 'FALSE'))

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

**Inserting into Customers**

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified, IsVIP)

VALUES (1, 'Aman Verma', TO\_DATE('1958-08-15', 'YYYY-MM-DD'), 25000, SYSDATE, NULL);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified, IsVIP)

VALUES (2, 'Priya Nair', TO\_DATE('2002-03-12', 'YYYY-MM-DD'), 9500, SYSDATE, NULL);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified, IsVIP)

VALUES (3, 'Rahul Singh', TO\_DATE('1960-11-05', 'YYYY-MM-DD'), 12000, SYSDATE, NULL);

**Inserting into Accounts**

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 25000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Savings', 9500, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (3, 3, 'Savings', 12000, SYSDATE);

**Inserting into Transactions**

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (1, 1, SYSDATE, 5000, 'Deposit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (2, 2, SYSDATE, 3000, 'Withdrawal');

**Inserting into Loans**

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 400000, 8, SYSDATE, SYSDATE + 15);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (2, 2, 150000, 10, SYSDATE, ADD\_MONTHS(SYSDATE, 36));

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (3, 3, 250000, 7.5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

**Inserting into Employees**

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (1, 'Neha Sharma', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (2, 'Rohit Mehta', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

COMMIT;

**Process Monthly Interest**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest added to savings accounts.');

END;

/

BEGIN ProcessMonthlyInterest; END;



**Update Employee Bonus**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE Employees

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

HireDate = SYSDATE

WHERE Department = p\_department;

COMMIT;

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

END;

/

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

/

SELECT \* FROM EMPLOYEES;



**Transfer Funds Between Accounts**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) AS

v\_from\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_from\_balance

FROM Accounts

WHERE AccountID = p\_from\_account

FOR UPDATE;

IF v\_from\_balance >= p\_amount THEN

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('₹' || p\_amount || ' transferred from Account ' || p\_from\_account || ' to Account ' || p\_to\_account);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient funds in Account ' || p\_from\_account || '. Transfer failed.');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

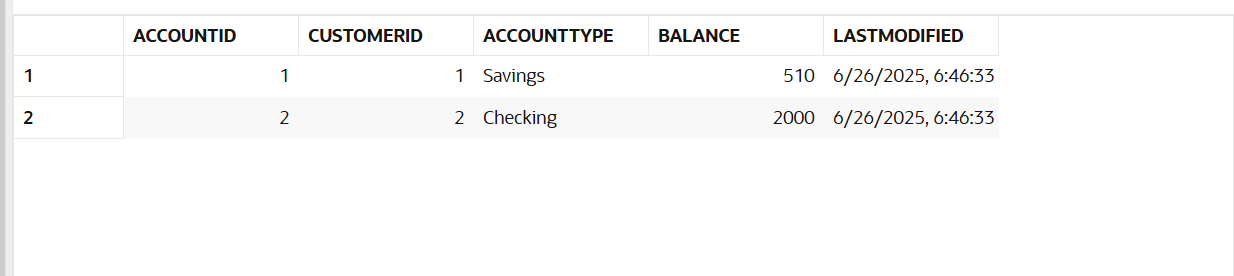
DBMS\_OUTPUT.PUT\_LINE('Invalid Account ID.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

END;

/



**For calling-**

**BEGIN TransferFunds(1, 2, 500); END;**