PLSQL_Exercises

1. Control Structures

```
CREATE TABLE Customers (
   CustomerID NUMBER PRIMARY KEY,
   Name VARCHAR2(50),
   Age NUMBER,
   Balance NUMBER,
   IsVIP CHAR(1) DEFAULT 'N'
);

CREATE TABLE Loans (
   LoanID NUMBER PRIMARY KEY,
   CustomerID NUMBER,
   InterestRate NUMBER,
   DueDate DATE,
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

-- Insert sample customers
INSERT INTO Customers VALUES (1, 'John Smith', 65, 15000, 'N');
INSERT INTO Customers VALUES (2, 'Alice Green', 45, 8000, 'N');
INSERT INTO Customers VALUES (3, 'Bob White', 70, 12000, 'N');
INSERT INTO Customers VALUES (4, 'Charlie Brown', 30, 5000, 'N');

-- Insert sample loans
INSERT INTO Loans VALUES (101, 1, 10.5, SYSDATE + 15);
INSERT INTO Loans VALUES (102, 2, 9.5, SYSDATE + 45);
INSERT INTO Loans VALUES (103, 3, 8.5, SYSDATE + 10);
INSERT INTO Loans VALUES (104, 4, 11.0, SYSDATE + 5);

-- Enable formatted output
SET LINESIZE 100
SET PAGESIZE 50

-- Set column width for display
COLUMN Name FORMAT A20
COLUMN IsVIP FORMAT A5
COLUMN Balance FORMAT 999999.99
COLUMN LoanID FORMAT 9999
COLUMN InterestRate FORMAT 99.99
COLUMN DueDate FORMAT A12
```

```
SQL> select * from customers;

CUSTOMERID NAME                          AGE    BALANCE ISVIP
---------- -------------------- ---------- ---------- -----
         1 John Smith                    65   15000.00 N
         2 Alice Green                   45    8000.00 N
         3 Bob White                     70   12000.00 N
         4 Charlie Brown                 30    5000.00 N
```

```
SQL> select * from loans;

LOANID CUSTOMERID INTERESTRATE DUEDATE
------ ---------- ------------ ------------
   101          1        10.50 16-JUL-25
   102          2         9.50 15-AUG-25
   103          3         8.50 11-JUL-25
   104          4        11.00 06-JUL-25
```

Scenario 1: Apply 1% Discount on Interest for Age > 60

```
SQL> BEGIN
  2      FOR loan_rec IN (
  3          SELECT l.LoanID, l.InterestRate
  4          FROM Loans l
  5          JOIN Customers c ON l.CustomerID = c.CustomerID
  6          WHERE c.Age > 60
  7      )
  8      LOOP
  9          UPDATE Loans
 10          SET InterestRate = loan_rec.InterestRate - 1
 11          WHERE LoanID = loan_rec.LoanID;
 12      END LOOP;
 13
 14      COMMIT;
 15  END;
 16  /

PL/SQL procedure successfully completed.
```

Scenario 2: Set IsVIP Flag for Balance > 10000

```
SQL> BEGIN
  2      FOR cust_rec IN (
  3          SELECT CustomerID FROM Customers WHERE Balance > 10000
  4      )
  5      LOOP
  6          UPDATE Customers
  7          SET IsVIP = 'Y'
  8          WHERE CustomerID = cust_rec.CustomerID;
  9      END LOOP;
 10
 11      COMMIT;
 12  END;
 13  /

PL/SQL procedure successfully completed.
```

Scenario 3: Send Reminder for Loans Due in Next 30 Days

```
SQL> DECLARE
  2      v_name VARCHAR2(50);
  3  BEGIN
  4      FOR loan_rec IN (
  5          SELECT l.LoanID, c.Name, l.DueDate
  6          FROM Loans l
  7          JOIN Customers c ON l.CustomerID = c.CustomerID
  8          WHERE l.DueDate <= SYSDATE + 30
  9      )
 10      LOOP
 11          DBMS_OUTPUT.PUT_LINE('Reminder: ' || loan_rec.Name ||
 12                               ', your loan (ID: ' || loan_rec.LoanID ||
 13                               ') is due on ' || TO_CHAR(loan_rec.DueDate, 'DD-Mon-YYYY'));
 14      END LOOP;
 15  END;
 16  /
Reminder: John Smith, your loan (ID: 101) is due on 16-Jul-2025
Reminder: Bob White, your loan (ID: 103) is due on 11-Jul-2025
Reminder: Charlie Brown, your loan (ID: 104) is due on 06-Jul-2025

PL/SQL procedure successfully completed.
```

2. Stored Procedures

```
-- Table for Accounts
CREATE TABLE Accounts (
   AccountID NUMBER PRIMARY KEY,
   CustomerName VARCHAR2(100),
   AccountType VARCHAR2(20), -- e.g., 'Savings', 'Current'
   Balance NUMBER
);
```

```
-- Table for Employees
CREATE TABLE Employees (
   EmpID NUMBER PRIMARY KEY,
   EmpName VARCHAR2(100),
   Department VARCHAR2(50),
   Salary NUMBER,
   PerformanceScore NUMBER
);

-- Insert into Accounts
INSERT INTO Accounts VALUES (1, 'Alice', 'Savings', 10000);
INSERT INTO Accounts VALUES (2, 'Bob', 'Current', 5000);
INSERT INTO Accounts VALUES (3, 'Charlie', 'Savings', 20000);
INSERT INTO Accounts VALUES (4, 'David', 'Savings', 15000);

-- Insert into Employees
INSERT INTO Employees VALUES (101, 'John', 'Sales', 40000, 85);
INSERT INTO Employees VALUES (102, 'Jane', 'HR', 35000, 78);
INSERT INTO Employees VALUES (103, 'Smith', 'Sales', 45000, 90);
INSERT INTO Employees VALUES (104, 'Emily', 'HR', 38000, 82);

COLUMN CustomerName FORMAT A20
COLUMN AccountType FORMAT A10
COLUMN Balance FORMAT 99999999.99

COLUMN EmpName FORMAT A20
COLUMN Department FORMAT A15
COLUMN Salary FORMAT 999999.99
```

```
SQL> select * from accounts;

 ACCOUNTID CUSTOMERNAME          ACCOUNTTYP      BALANCE
---------- -------------------- ---------- ------------
         1 Alice                Savings       10000.00
         2 Bob                  Current        5000.00
         3 Charlie              Savings       20000.00
         4 David                Savings       15000.00
```

```
SQL> select * from employees;

     EMPID EMPNAME              DEPARTMENT         SALARY PERFORMANCESCORE
---------- -------------------- -------------- ---------- ----------------
       101 John                 Sales             40000.00               85
       102 Jane                 HR                35000.00               78
       103 Smith                Sales             45000.00               90
       104 Emily                HR                38000.00               82
```

Scenario 1: ProcessMonthlyInterest

```
SQL> CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
  2  BEGIN
  3      UPDATE Accounts
  4      SET Balance = Balance + (Balance * 0.01)
  5      WHERE AccountType = 'Savings';
  6
  7      COMMIT;
  8  END;
  9  /

Procedure created.
```

Scenario 2: UpdateEmployeeBonus

```
SQL> CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(
  2      dept IN VARCHAR2,
  3      bonus_percent IN NUMBER
  4  ) IS
  5  BEGIN
  6      UPDATE Employees
  7      SET Salary = Salary + (Salary * bonus_percent / 100)
  8      WHERE Department = dept;
  9
 10      COMMIT;
 11  END;
 12  /

Procedure created.
```

Scenario 3: TransferFunds

```
SQL> CREATE OR REPLACE PROCEDURE TransferFunds(
  2      from_account IN NUMBER,
  3      to_account IN NUMBER,
  4      amount IN NUMBER
  5  ) IS
  6      from_balance NUMBER;
  7  BEGIN
  8      SELECT Balance INTO from_balance FROM Accounts WHERE AccountID = from_account FOR UPDATE;
  9
 10      IF from_balance < amount THEN
 11          RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds in source account.');
 12      ELSE
 13          UPDATE Accounts SET Balance = Balance - amount WHERE AccountID = from_account;
 14          UPDATE Accounts SET Balance = Balance + amount WHERE AccountID = to_account;
 15      END IF;
 16
 17      COMMIT;
 18  END;
 19  /

Procedure created.
```