WEEK 2 – ASSIGNMENT

Superset ID: 6390124

PL/SQL Exercises:-

Schema to be Created:

```
CREATE TABLE Customers (
    CustomerID NUMBER PRIMARY KEY,
    Name VARCHAR2(100),
    DOB DATE,
    Balance NUMBER,
    LastModified DATE,
    IsVIP CHAR(1)
);

CREATE TABLE Accounts (
    AccountID NUMBER PRIMARY KEY,
    CustomerID NUMBER,
    AccountType VARCHAR2(20),
    Balance NUMBER,
    LastModified DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

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
);

INSERT INTO Customers VALUES (1, 'John Doe', TO_DATE('1960-05-15','YYYY-MM-
DD'), 12000, SYSDATE, NULL);
INSERT INTO Customers VALUES (2, 'Jane Smith', TO_DATE('1988-07-20','YYYY-MM-
DD'), 8000, SYSDATE, NULL);

INSERT INTO Accounts VALUES (1, 1, 'Savings', 1000, SYSDATE);
INSERT INTO Accounts VALUES (2, 2, 'Checking', 1500, SYSDATE);

INSERT INTO Loans VALUES (1, 1, 5000, 5, SYSDATE, SYSDATE + 20); -- Due in
next 30 days

INSERT INTO Employees VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR',
TO_DATE('2015-06-15','YYYY-MM-DD'));
```

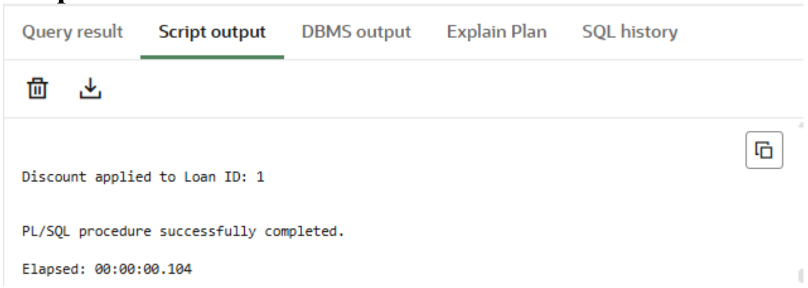
```
INSERT INTO Employees VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT',
TO_DATE('2017-03-20','YYYY-MM-DD'));

COMMIT;
```

Exercise 1: Control Structures

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

```
BEGIN
  FOR rec IN (
    SELECT l.LoanID, l.InterestRate, c.DOB
    FROM Loans l
    JOIN Customers c ON l.CustomerID = c.CustomerID
  ) LOOP
    -- Calculate age
    IF TRUNC(MONTHS_BETWEEN(SYSDATE, rec.DOB) / 12) > 60 THEN
      -- Apply 1% discount
      UPDATE Loans
      SET InterestRate = rec.InterestRate - 1
      WHERE LoanID = rec.LoanID;

      DBMS_OUTPUT.PUT_LINE('Discount applied to Loan ID: ' || rec.LoanID);
    END IF;
  END LOOP;
END;
/
```

Output:

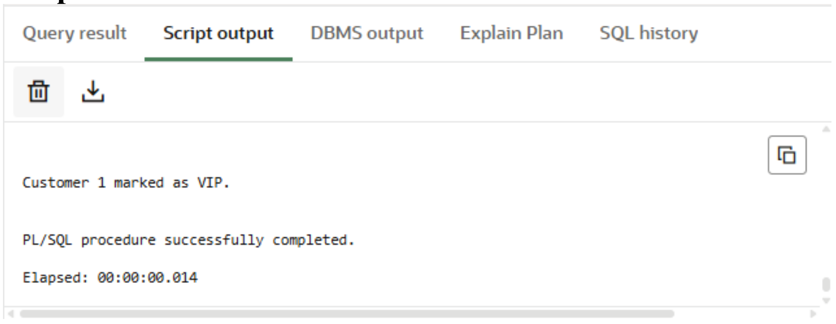


```
Discount applied to Loan ID: 1

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.104
```

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

```
BEGIN
  FOR rec IN (
    SELECT CustomerID, Balance
    FROM Customers
  ) LOOP
    IF rec.Balance > 10000 THEN
      UPDATE Customers
      SET IsVIP = 'Y'
      WHERE CustomerID = rec.CustomerID;

      DBMS_OUTPUT.PUT_LINE('Customer ' || rec.CustomerID || ' marked as 
VIP.');
    END IF;
  END LOOP;
END;
/
```

Output:

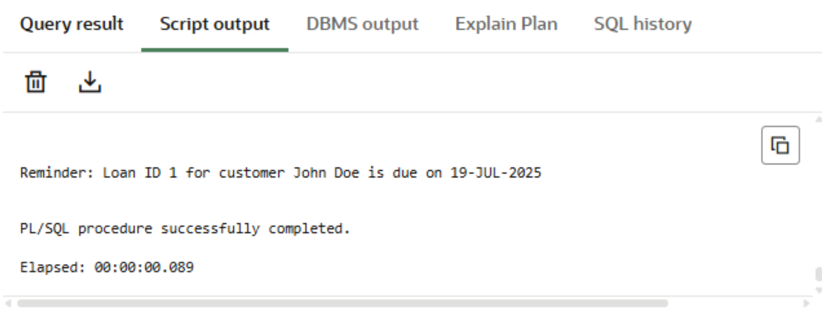


```
Customer 1 marked as VIP.


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.014
```

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.

```
BEGIN
  FOR rec IN (
    SELECT c.Name, l.LoanID, l.EndDate
    FROM Loans l
    JOIN Customers c ON l.CustomerID = c.CustomerID
    WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || rec.LoanID ||
                         ' for customer ' || rec.Name ||
                         ' is due on ' || TO_CHAR(rec.EndDate, 'DD-MON-
YYYY'));
  END LOOP;
END;
/
```

Output:



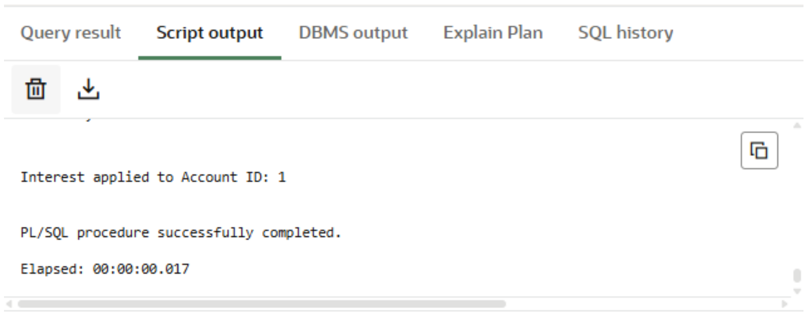
```
Reminder: Loan ID 1 for customer John Doe is due on 19-JUL-2025


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.089
```

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.

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
  FOR rec IN (
    SELECT AccountID, Balance
    FROM Accounts
    WHERE AccountType = 'Savings'
  ) LOOP
    UPDATE Accounts
    SET Balance = rec.Balance + (rec.Balance * 0.01)
    WHERE AccountID = rec.AccountID;

    DBMS_OUTPUT.PUT_LINE('Interest applied to Account ID: ' ||
rec.AccountID);
  END LOOP;
END;
/

BEGIN
  ProcessMonthlyInterest;
END;
/
```

Output:

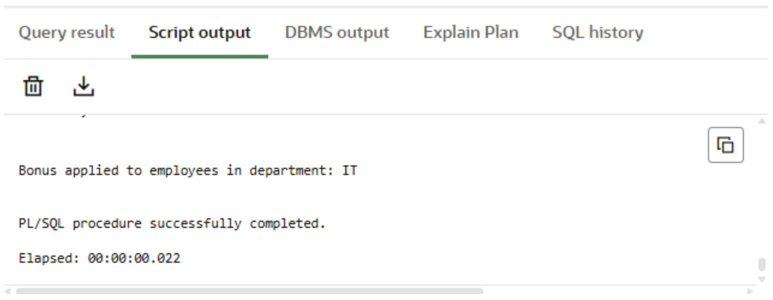


```
Interest applied to Account ID: 1

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.017
```

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.

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
  dept_name IN VARCHAR2,
  bonus_pct IN NUMBER
) IS
BEGIN
  UPDATE Employees
  SET Salary = Salary + (Salary * bonus_pct / 100)
  WHERE Department = dept_name;

  DBMS_OUTPUT.PUT_LINE('Bonus applied to employees in department: ' ||
dept_name);
END;
/

BEGIN
  UpdateEmployeeBonus('IT', 10);
END;
/
```

Output:

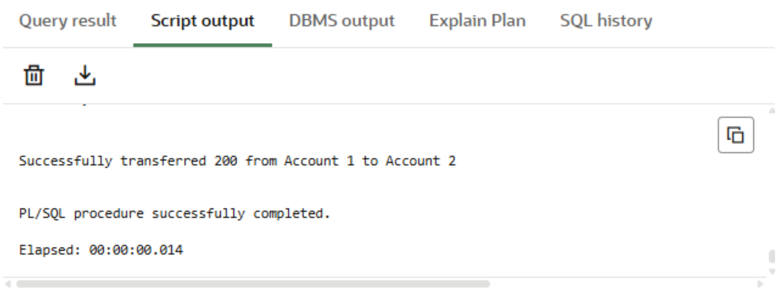


```
Bonus applied to employees in department: IT


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.022
```

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.

```
CREATE OR REPLACE PROCEDURE TransferFunds (
  from_acc IN NUMBER,
  to_acc IN NUMBER,
  amount IN NUMBER
) IS
  from_balance NUMBER;
BEGIN
  -- Get source account balance
  SELECT Balance INTO from_balance FROM Accounts WHERE AccountID = from_acc;

  IF from_balance < amount THEN
    RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance in source
account.');
  END IF;

  -- Deduct from source
  UPDATE Accounts
  SET Balance = Balance - amount
  WHERE AccountID = from_acc;

  -- Add to destination
  UPDATE Accounts
  SET Balance = Balance + amount
  WHERE AccountID = to_acc;

  DBMS_OUTPUT.PUT_LINE('Successfully transferred ' || amount ||
                       ' from Account ' || from_acc || ' to Account ' ||
to_acc);
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Transfer failed: ' || SQLERRM);
END;
/

BEGIN
  TransferFunds(1, 2, 200);
END;
/
```

Output:



```
Successfully transferred 200 from Account 1 to Account 2


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.014
```