Q1.

DECLARE

-- Define a record type to hold transaction details

TYPE TransactionRecord IS RECORD (

CustomerID NUMBER,

CustomerName VARCHAR2(100),

AccountID NUMBER,

TransactionID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10)

);

-- Define a cursor to fetch transactions for the current month

CURSOR TransactionsCursor IS

SELECT c.CustomerID, c.Name AS CustomerName, t.AccountID, t.TransactionID,

t.TransactionDate, t.Amount, t.TransactionType

FROM Customers c

JOIN Accounts a ON c.CustomerID = a.CustomerID

JOIN Transactions t ON a.AccountID = t.AccountID

WHERE t.TransactionDate >= TRUNC(SYSDATE, 'MM') AND t.TransactionDate < ADD\_MONTHS(TRUNC(SYSDATE, 'MM'), 1)

ORDER BY c.CustomerID, t.TransactionDate;

-- Declare a record variable to hold each fetched transaction

transaction\_rec TransactionRecord;

-- Variables to track the current customer and account

current\_customer\_id NUMBER := NULL;

current\_account\_id NUMBER := NULL;

BEGIN

-- Open the cursor

OPEN TransactionsCursor;

-- Fetch transactions one by one

LOOP

FETCH TransactionsCursor INTO transaction\_rec;

EXIT WHEN TransactionsCursor%NOTFOUND;

-- Check if we moved to a new customer

IF current\_customer\_id IS NULL OR current\_customer\_id != transaction\_rec.CustomerID THEN

-- Print a header for the new customer's statement

IF current\_customer\_id IS NOT NULL THEN

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

END IF;

DBMS\_OUTPUT.PUT\_LINE('Customer ID: ' || transaction\_rec.CustomerID);

DBMS\_OUTPUT.PUT\_LINE('Customer Name: ' || transaction\_rec.CustomerName);

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

current\_customer\_id := transaction\_rec.CustomerID;

current\_account\_id := NULL; -- Reset account ID for new customer

END IF;

-- Check if we moved to a new account

IF current\_account\_id IS NULL OR current\_account\_id != transaction\_rec.AccountID THEN

DBMS\_OUTPUT.PUT\_LINE('Account ID: ' || transaction\_rec.AccountID);

current\_account\_id := transaction\_rec.AccountID;

END IF;

-- Print the transaction details

DBMS\_OUTPUT.PUT\_LINE('Transaction ID: ' || transaction\_rec.TransactionID);

DBMS\_OUTPUT.PUT\_LINE('Transaction Date: ' || TO\_CHAR(transaction\_rec.TransactionDate, 'YYYY-MM-DD'));

DBMS\_OUTPUT.PUT\_LINE('Transaction Type: ' || transaction\_rec.TransactionType);

DBMS\_OUTPUT.PUT\_LINE('Amount: ' || transaction\_rec.Amount);

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

END LOOP;

-- Close the cursor

CLOSE TransactionsCursor;

END;

/

Q2.

DECLARE

-- Define a constant for the annual maintenance fee

annual\_fee CONSTANT NUMBER := 50;

-- Define a record type to hold account details

TYPE AccountRecord IS RECORD (

AccountID NUMBER,

Balance NUMBER

);

-- Define a cursor to fetch all accounts

CURSOR AccountsCursor IS

SELECT AccountID, Balance

FROM Accounts

FOR UPDATE OF Balance;

-- Declare a record variable to hold each fetched account

account\_rec AccountRecord;

BEGIN

-- Open the cursor

OPEN AccountsCursor;

-- Fetch accounts one by one

LOOP

FETCH AccountsCursor INTO account\_rec;

EXIT WHEN AccountsCursor%NOTFOUND;

-- Apply the annual fee to the account balance

UPDATE Accounts

SET Balance = Balance - annual\_fee

WHERE CURRENT OF AccountsCursor;

END LOOP;

-- Close the cursor

CLOSE AccountsCursor;

-- Commit the transaction to make changes permanent

COMMIT;

-- Print a message indicating the process is complete

DBMS\_OUTPUT.PUT\_LINE('Annual maintenance fee applied to all accounts.');

END;

/

Q3.

DECLARE

-- Define a new interest rate increase

interest\_rate\_increase CONSTANT NUMBER := 1;

-- Define a record type to hold loan details

TYPE LoanRecord IS RECORD (

LoanID NUMBER,

InterestRate NUMBER

);

-- Define a cursor to fetch all loans

CURSOR LoansCursor IS

SELECT LoanID, InterestRate

FROM Loans

FOR UPDATE OF InterestRate;

-- Declare a record variable to hold each fetched loan

loan\_rec LoanRecord;

BEGIN

-- Open the cursor

OPEN LoansCursor;

-- Fetch loans one by one

LOOP

FETCH LoansCursor INTO loan\_rec;

EXIT WHEN LoansCursor%NOTFOUND;

-- Update the interest rate for the loan

UPDATE Loans

SET InterestRate = loan\_rec.InterestRate + interest\_rate\_increase

WHERE CURRENT OF LoansCursor;

END LOOP;

-- Close the cursor

CLOSE LoansCursor;

-- Commit the transaction to make changes permanent

COMMIT;

-- Print a message indicating the process is complete

DBMS\_OUTPUT.PUT\_LINE('Interest rates updated for all loans.');

END;

/