**Exercise 1: Control Structures**

**Scenario 1:**

**For this scenario the customer table have to create:**

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

age NUMBER,

balance NUMBER(12,2),

is\_vip CHAR(1) DEFAULT 'N' -- 'Y' = Yes, 'N' = No

);

//pl/sql code

SET SERVEROUTPUT ON;

BEGIN

FOR cust\_rec IN (SELECT customer\_id FROM customers WHERE age > 60) LOOP

FOR loan\_rec IN (

SELECT loan\_id, interest\_rate FROM loans

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

) LOOP

UPDATE loans

SET interest\_rate = loan\_rec.interest\_rate - 1

WHERE loan\_id = loan\_rec.loan\_id;

DBMS\_OUTPUT.PUT\_LINE(

'Loan ID ' || loan\_rec.loan\_id ||

' for Customer ' || cust\_rec.customer\_id ||

' updated: ' || loan\_rec.interest\_rate || '% -> ' ||

(loan\_rec.interest\_rate - 1) || '%'

);

END LOOP;

END LOOP;

COMMIT;

END;

/

**Scenario 2:**

SET SERVEROUTPUT ON;

BEGIN

FOR cust\_rec IN (SELECT customer\_id, name, balance FROM customers WHERE balance > 10000) LOOP

UPDATE customers

SET is\_vip = 'Y'

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

DBMS\_OUTPUT.PUT\_LINE(

'Customer ' || cust\_rec.name ||

' promoted to VIP. Balance: $' || cust\_rec.balance

);

END LOOP;

COMMIT;

END;

/

**Scenaio 3:**

-- Loan Table

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER REFERENCES customers(customer\_id),

interest\_rate NUMBER(5,2),

due\_date DATE

);

//pl/sql code

SET SERVEROUTPUT ON;

BEGIN

FOR loan\_rec IN (

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

FROM loans l

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

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

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: ' || loan\_rec.name ||

' has a loan (ID: ' || loan\_rec.loan\_id ||

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

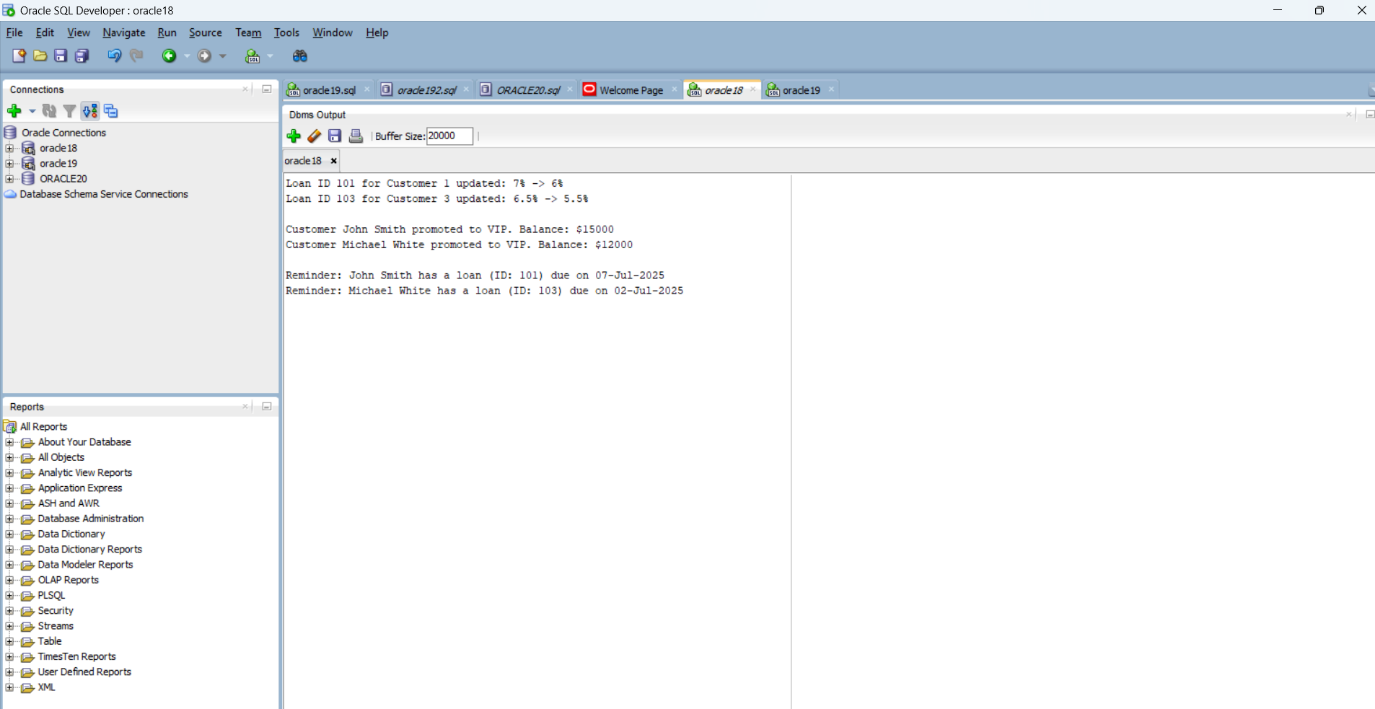
);

END LOOP;

END;

/

**Output:**

****

**Exercise 3: Stored Procedures**

# Tables created

-- Savings Account Table

CREATE TABLE savings\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER(12,2)

);

-- Employees Table

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department\_id NUMBER,

salary NUMBER(10,2)

);

-- Generic Accounts Table (for fund transfer)

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER(12,2)

);

**Scenario 1:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

v\_old\_balance NUMBER;

v\_new\_balance NUMBER;

BEGIN

FOR acc IN (SELECT account\_id, balance FROM savings\_accounts) LOOP

v\_old\_balance := acc.balance;

v\_new\_balance := acc.balance + (acc.balance \* 0.01);

UPDATE savings\_accounts

SET balance = v\_new\_balance

WHERE account\_id = acc.account\_id;

DBMS\_OUTPUT.PUT\_LINE('Account ' || acc.account\_id ||

' updated: ' || v\_old\_balance ||

' -> ' || v\_new\_balance);

END LOOP;

COMMIT;

END;

/

**Scenaio 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept\_id IN NUMBER,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

FOR emp\_rec IN (

SELECT emp\_id, name, salary

FROM employees

WHERE department\_id = p\_dept\_id

) LOOP

UPDATE employees

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

WHERE emp\_id = emp\_rec.emp\_id;

DBMS\_OUTPUT.PUT\_LINE('Employee ' || emp\_rec.name ||

' new salary: ' ||

(emp\_rec.salary + emp\_rec.salary \* p\_bonus\_percent / 100));

END LOOP;

COMMIT;

END;

/

**Scenaio 3:**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_source\_acc\_id IN NUMBER,

p\_target\_acc\_id IN NUMBER,

p\_amount IN NUMBER

) IS

v\_source\_balance NUMBER;

BEGIN

-- Lock and check source balance

SELECT balance INTO v\_source\_balance

FROM accounts

WHERE account\_id = p\_source\_acc\_id

FOR UPDATE;

IF v\_source\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: insufficient balance.');

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance in source account.');

END IF;

-- Deduct and update

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_source\_acc\_id;

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_target\_acc\_id;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || p\_amount ||

' from Account ' || p\_source\_acc\_id ||

' to Account ' || p\_target\_acc\_id);

COMMIT;

END;

/

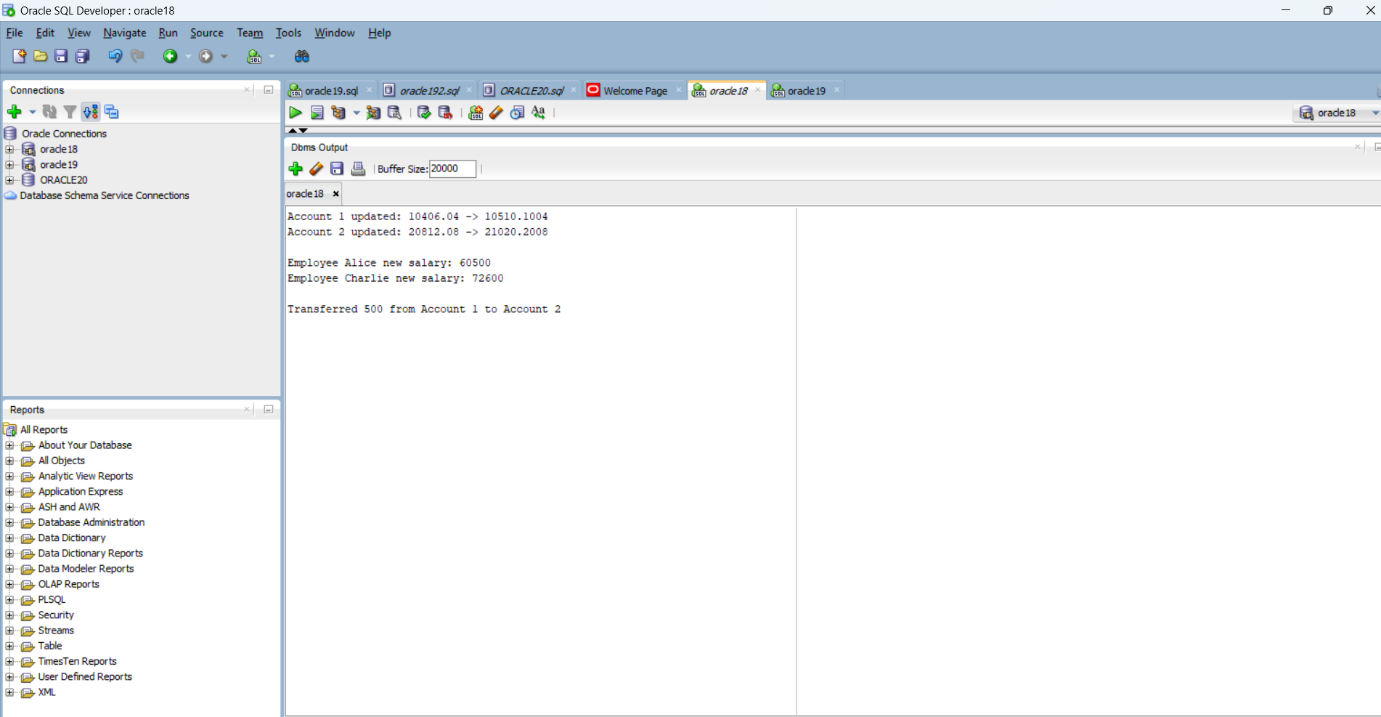
# to get output this line need to br ececuted;

**EXEC ProcessMonthlyInterest;**

**EXEC UpdateEmployeeBonus(101, 10);**

**EXEC TransferFunds(1, 2, 500);**

**OUTPUT:**

****