**Exercise 1: Control Structures**

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

CREATE TABLE Bank(

    Customer\_id INT,

    Customer\_name VARCHAR(20),

    Customer\_age INT

);

INSERT INTO Bank(Customer\_id, Customer\_name, Customer\_age) VALUES (101, 'Zandeep', 20);

INSERT INTO Bank(Customer\_id, Customer\_name, Customer\_age) VALUES (102, 'Yashwanth Saran', 67);

INSERT INTO Bank(Customer\_id, Customer\_name, Customer\_age) VALUES  (103, 'Jothi Ramakumar', 61);

INSERT INTO Bank(Customer\_id, Customer\_name, Customer\_age) VALUES (104, 'Rama Kumar', 64);

DECLARE

v\_id Bank.Customer\_id%TYPE;

v\_name Bank.Customer\_name%TYPE;

v\_age Bank.Customer\_age%TYPE;

-- DECLARE THE CURSOR FOR DISPLAYING MULITPLE ROWS

CURSOR Bank\_cursor IS SELECT Customer\_id, Customer\_name, Customer\_age FROM Bank;

BEGIN

    FOR record IN Bank\_cursor LOOP

    IF  record.Customer\_age > 60 THEN

    DBMS\_OUTPUT.PUT\_LINE('CUSTOMER ID :' || record.Customer\_id );

    DBMS\_OUTPUT.PUT\_LINE('CUSTOMER NAME :' || record.Customer\_name );

    DBMS\_OUTPUT.PUT\_LINE('CUSTOMER AGE :' || record.Customer\_age );

    DBMS\_OUTPUT.PUT\_LINE('Eligible for apply a 1% discount to their current loan interest rates');

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

    ELSE

     DBMS\_OUTPUT.PUT\_LINE('CUSTOMER ID :' || record.Customer\_id );

    DBMS\_OUTPUT.PUT\_LINE('CUSTOMER NAME :' || record.Customer\_name );

    DBMS\_OUTPUT.PUT\_LINE('CUSTOMER AGE :' || record.Customer\_age );

    DBMS\_OUTPUT.PUT\_LINE(' Not Eligible for apply a 1% discount to their current loan interest rates');

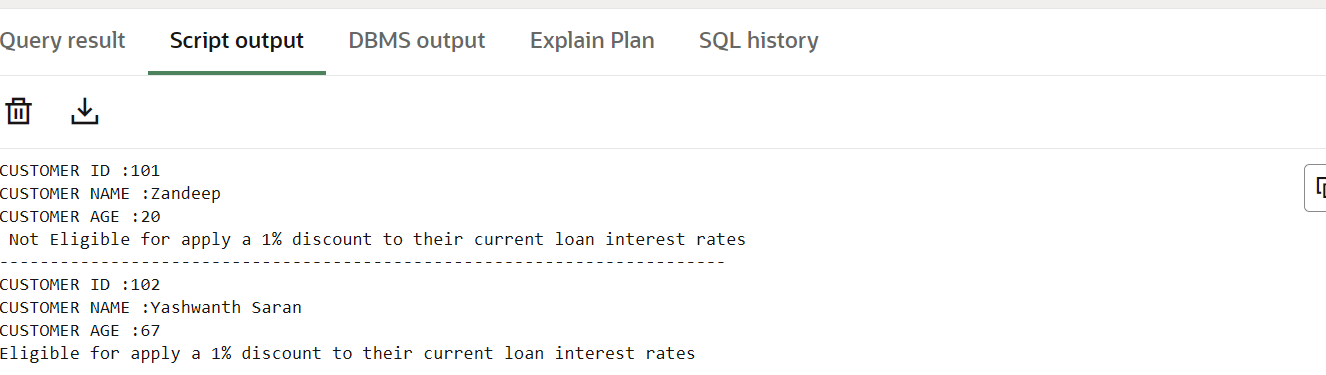
    DBMS\_OUTPUT.PUT\_LINE('------------------------------------------------------------------------');

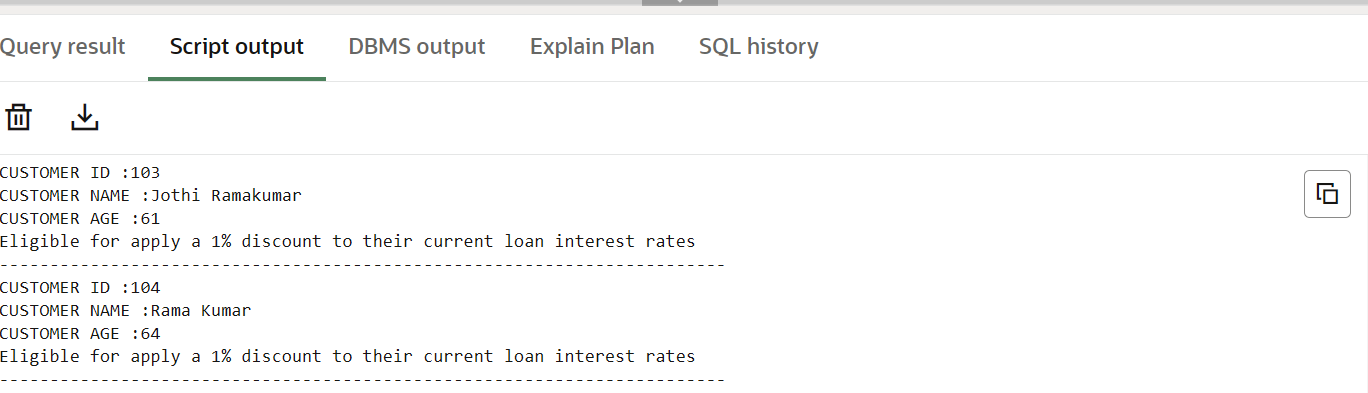
    END IF;

    END LOOP;

END;

**OUTPUT:**





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

CREATE TABLE Bank (

    Customer\_id INT,

    Customer\_name VARCHAR2(20),

    Customer\_balance INT

);

INSERT INTO Bank(Customer\_id, Customer\_name, Customer\_balance) VALUES (101, 'Zandeep', 9000);

INSERT INTO Bank(Customer\_id, Customer\_name, Customer\_balance) VALUES (102, 'Yashwanth Saran', 20000);

INSERT INTO Bank(Customer\_id, Customer\_name, Customer\_balance) VALUES (103, 'Jothi Ramakumar', 30000);

INSERT INTO Bank(Customer\_id, Customer\_name, Customer\_balance) VALUES (104, 'Rama Kumar', 40000);

COMMIT;

DECLARE

    v\_id Bank.Customer\_id%TYPE;

    v\_name Bank.Customer\_name%TYPE;

    v\_balance Bank.Customer\_balance%TYPE;

    VIP BOOLEAN := FALSE;

    -- Cursor

    CURSOR Bank\_cursor IS

        SELECT Customer\_id, Customer\_name, Customer\_balance FROM Bank;

BEGIN

    FOR record IN Bank\_cursor LOOP

        IF record.Customer\_balance > 10000 THEN

            VIP := TRUE;

            DBMS\_OUTPUT.PUT\_LINE('CUSTOMER NAME (VIP CUSTOMER): ' || record.Customer\_name);

            DBMS\_OUTPUT.PUT\_LINE('CUSTOMER CURRENT BALANCE IS : $' || record.Customer\_balance);

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

        ELSE

            DBMS\_OUTPUT.PUT\_LINE('CUSTOMER NAME (NOT VIP CUSTOMER): ' || record.Customer\_name);

            DBMS\_OUTPUT.PUT\_LINE('CUSTOMER CURRENT BALANCE IS : $' || record.Customer\_balance);

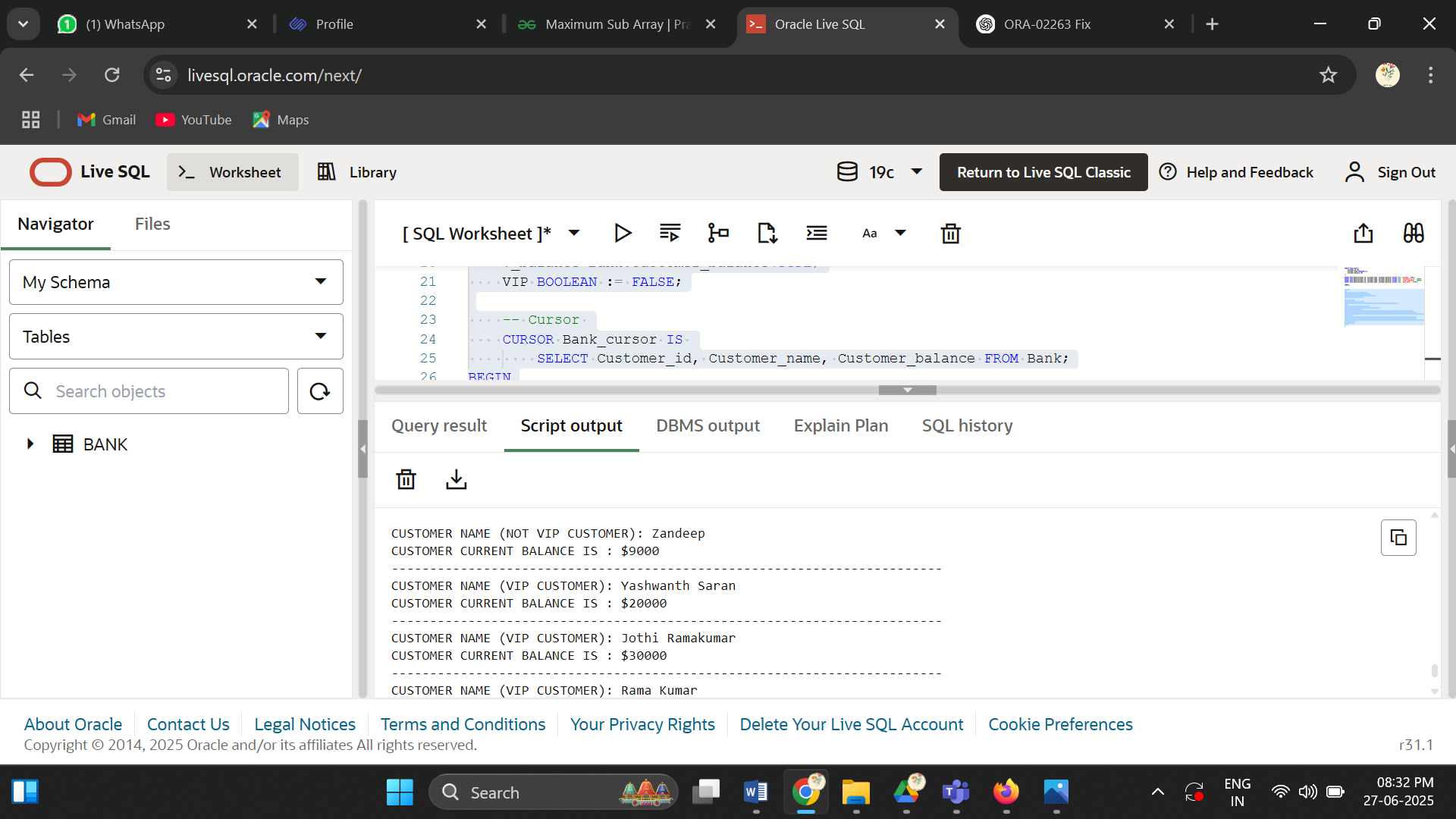
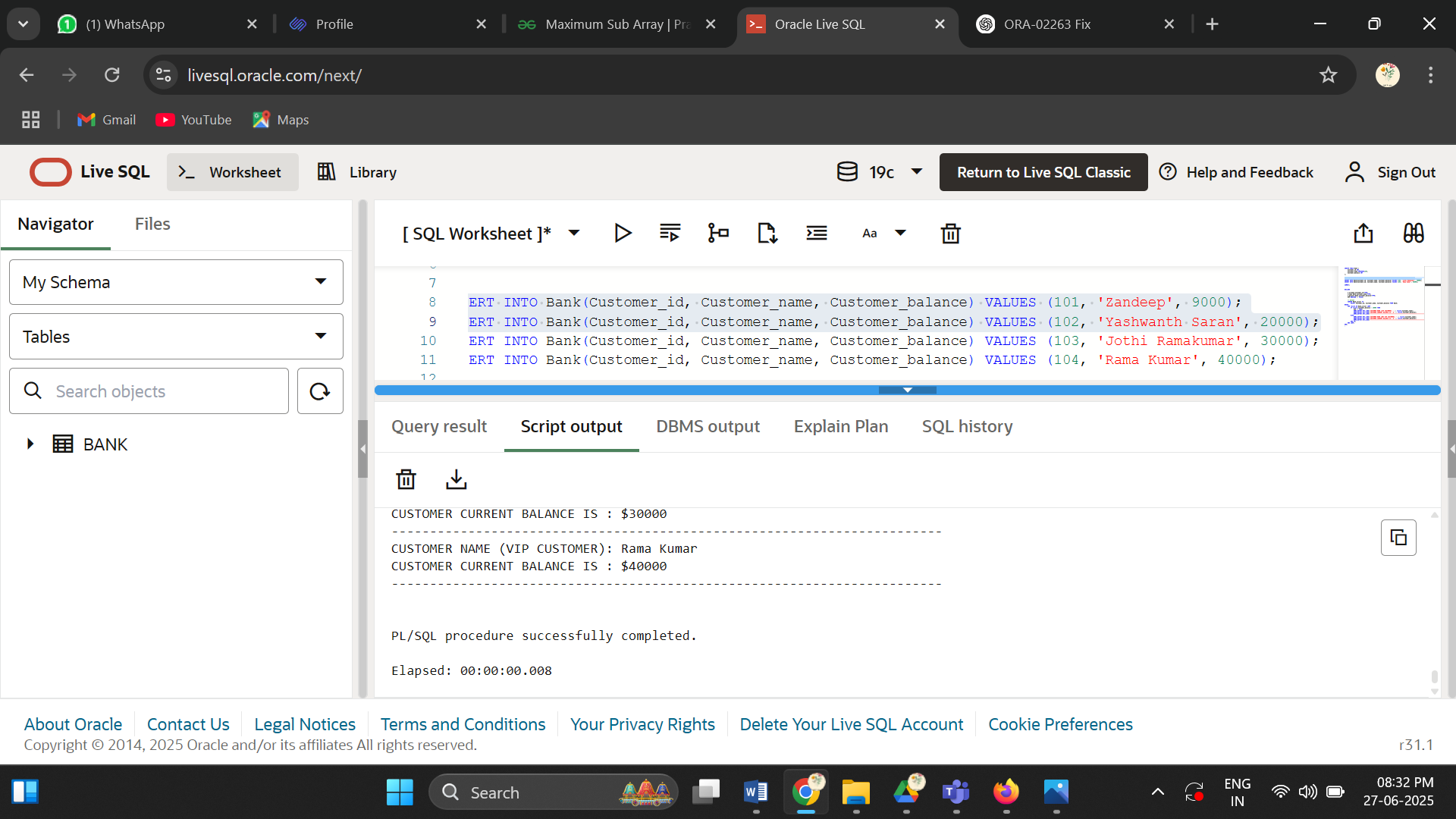
            DBMS\_OUTPUT.PUT\_LINE('------------------------------------------------------------------------');

        END IF;

    END LOOP;

END;

Output:



**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

CREATE TABLE Loan(

    Customer\_id INT,

    Customer\_name VARCHAR(50),

    Loan\_amount INT,

    Due\_date DATE

);

INSERT INTO Loan (Customer\_id, Customer\_name, Loan\_amount, Due\_date)

VALUES (1, 'DEEPTHIKHA', 30000, TO\_DATE('2025-07-30','YYYY-MM-DD'));

INSERT INTO Loan( Customer\_id, Customer\_name, Loan\_amount, Due\_date)

VALUES (2, 'ZANDEEP', 50000, TO\_DATE('2025-07-05','YYYY-MM-DD'));

INSERT INTO Loan( Customer\_id, Customer\_name, Loan\_amount, Due\_date)

VALUES (3, 'RAMAKUMAR', 80000, TO\_DATE('2025-08-04','YYYY-MM-DD'));

INSERT INTO Loan( Customer\_id, Customer\_name, Loan\_amount, Due\_date)

VALUES (4, 'JOTHI RAMALUMAR', 90000, TO\_DATE('2025-07-10','YYYY-MM-DD'));

DECLARE

    -- Cursor

    CURSOR Loan\_cursor IS

        SELECT Customer\_id, Customer\_name, Loan\_amount, Due\_date

        FROM Loan

        WHERE Due\_date BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

    FOR record IN Loan\_cursor LOOP

        DBMS\_OUTPUT.PUT\_LINE('Customer Name: ' || record.Customer\_name);

        DBMS\_OUTPUT.PUT\_LINE('Loan Amount: ' || record.Loan\_amount);

        DBMS\_OUTPUT.PUT\_LINE('Due Date: ' || TO\_CHAR(record.Due\_date, 'YYYY-MM-DD'));

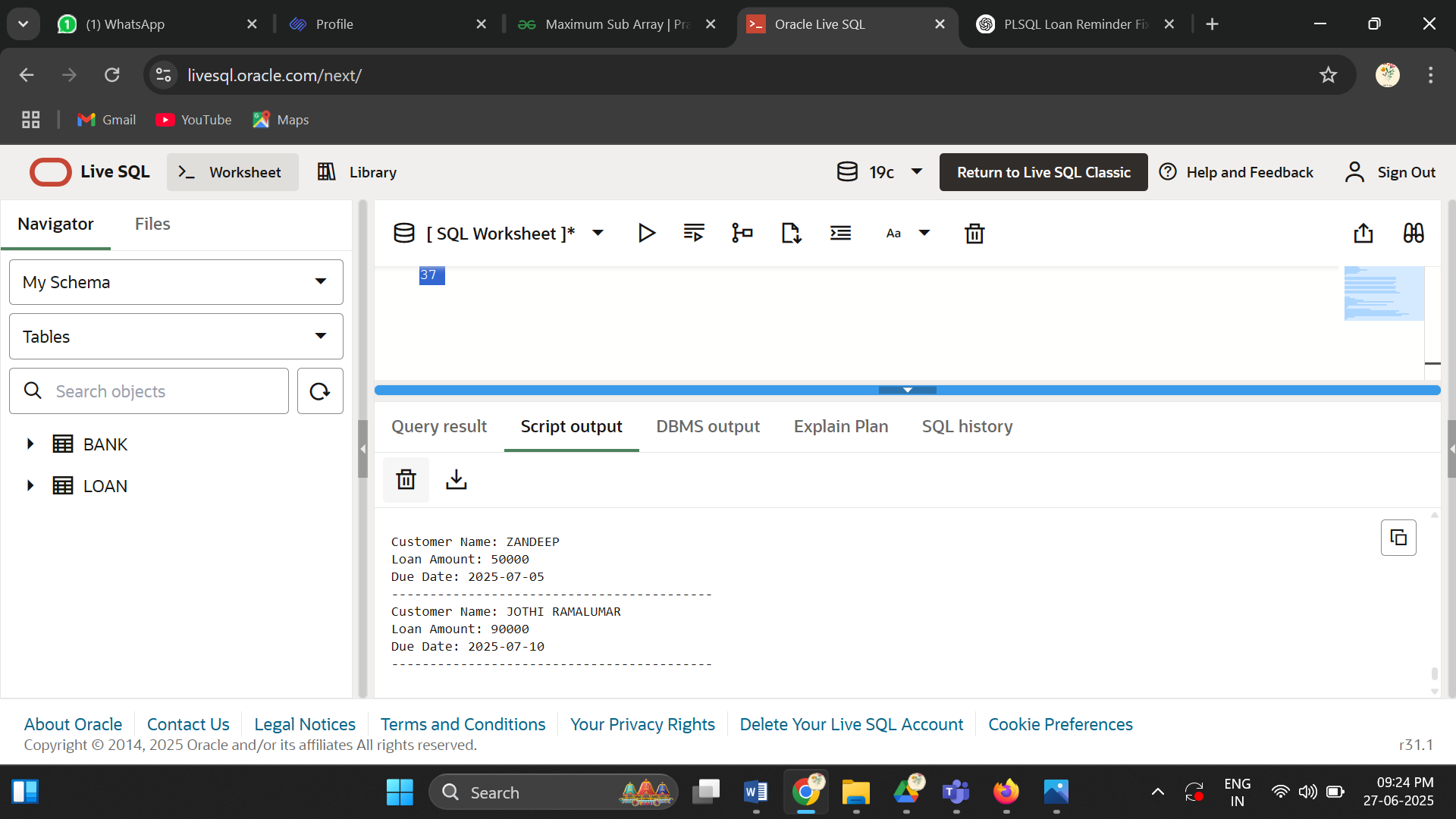
        DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

    END LOOP;

END;

/

Output:



**Exercise 2: Error Handling**

**Scenario 1:** Handle exceptions during fund transfers between accounts.

CREATE table Transaction(

    account\_owner\_id int,

    amout DECIMAL(10,2),

    account\_other\_id int

);

insert into Transaction(account\_owner\_id ,amout,account\_other\_id) values (1000125,5000,100030);

insert into Transaction(account\_owner\_id ,amout,account\_other\_id) values (1000126,20000,1000126);

insert into Transaction(account\_owner\_id ,amout,account\_other\_id) values (1000127,50000,100040);

CREATE OR REPLACE PROCEDURE SafeTransferFunds(

    account\_owner\_id IN NUMBER,

    amount IN NUMBER,

    account\_other\_id IN NUMBER

) IS

    inSufficient EXCEPTION;

BEGIN

    IF account\_owner\_id != account\_other\_id THEN

        DBMS\_OUTPUT.PUT\_LINE('Valid Transaction');

        DBMS\_OUTPUT.PUT\_LINE('Transfer the amount: ' || amount);

    ELSE

        RAISE inSufficient;

    END IF;

EXCEPTION

    WHEN inSufficient THEN

        DBMS\_OUTPUT.PUT\_LINE('ERROR!! Accounts cannot be the same.');

END SafeTransferFunds;

/

DECLARE

v\_account\_1 Transaction.account\_owner\_id %type;

v\_amount  Transaction.amout%type;

v\_account\_2 Transaction.account\_other\_id%type;

CURSOR c\_record is

select account\_owner\_id ,amout,account\_other\_id from Transaction;

BEGIN

    OPEN c\_record;

    LOOP

        FETCH c\_record INTO v\_account\_1,v\_amount,v\_account\_2;

        exit when c\_record%NOTFOUND;

        DBMS\_OUTPUT.PUT\_LINE('Processing Transaction...');

        SafeTransferFunds(v\_account\_1, v\_amount, v\_account\_2);

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

    END LOOP;

    CLOSE c\_record;

    END;

    /

Output:

Processing Transaction...  
Valid Transaction  
Transfer the amount: 5000  
------------------------------------  
Processing Transaction...  
Valid Transaction  
Transfer the amount: 20000  
------------------------------------  
Processing Transaction...  
Valid Transaction  
Transfer the amount: 50000  
------------------------------------  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.016

**Scenario 2:** Manage errors when updating employee salaries

CREATE table Employee(

    Employee\_id int,

    Employee\_name VARCHAR(50),

    Salary NUMBER

);

insert into Employee(Employee\_id,Employee\_name,Salary ) VALUES(101,'Deepthikha',50000);

insert into Employee(Employee\_id,Employee\_name,Salary ) VALUES(102,'GunaPriya',60000);

insert into Employee(Employee\_id,Employee\_name,Salary ) VALUES(101,'Jothi Ramakumar',80000);

CREATE OR REPLACE PROCEDURE UpdateSalary(

    Employee\_id IN NUMBER,

    Employee\_name IN VARCHAR2,

    Salary IN NUMBER

) IS

    current\_salary NUMBER;

    percentage CONSTANT NUMBER := 5;

    Exists\_EXCEPTION EXCEPTION;

BEGIN

    IF Employee\_id IS NOT NULL THEN

        current\_salary := Salary + (Salary \* percentage / 100);

        DBMS\_OUTPUT.PUT\_LINE('Old Salary :'||Salary);

        DBMS\_OUTPUT.PUT\_LINE('Increased the salary to: ' || current\_salary);

    ELSE

        RAISE Exists\_EXCEPTION;

    END IF;

EXCEPTION

    WHEN Exists\_EXCEPTION THEN

        DBMS\_OUTPUT.PUT\_LINE('ERROR!! Invalid Employee ID.');

END UpdateSalary;

/

DECLARE

 v\_id Employee.Employee\_id%type;

 v\_name Employee.Employee\_name%type;

 v\_salary  Employee.Salary%type;

 CURSOR c\_rec is

 select Employee\_id,Employee\_name,Salary from Employee;

BEGIN

    OPEN c\_rec;

    LOOP

        FETCH c\_rec into v\_id,v\_name,v\_salary;

        exit when c\_rec%NOTFOUND;

        UpdateSalary(v\_id,v\_name,v\_salary);

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

    END LOOP;

    close c\_rec;

    end;

    /

Output:

Old Salary :50000  
Increased the salary to: 52500  
--------------------  
Old Salary :60000  
Increased the salary to: 63000  
--------------------  
Old Salary :80000  
Increased the salary to: 84000  
--------------------  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.003

**Scenario 3:** Ensure data integrity when adding a new customer.

CREATE TABLE Customer (

    customer\_id INT,

    customer\_name VARCHAR(50)

);

INSERT INTO Customer(customer\_id, customer\_name) VALUES (1, 'Deepthikha');

INSERT INTO Customer(customer\_id, customer\_name) VALUES (2, 'Kiruthiga');

CREATE OR REPLACE PROCEDURE AddNewCustomer(

    customer\_id IN NUMBER,

    customer\_name IN VARCHAR2

) IS

    customer\_exists NUMBER;

BEGIN

    IF customer\_id IS NULL OR customer\_name IS NULL THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: Customer ID and Name must not be NULL.');

        RETURN;

    END IF;

    SELECT COUNT(\*) INTO customer\_exists FROM Customer

    WHERE customer\_id = AddNewCustomer.customer\_id;

    IF customer\_exists > 0 THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: Customer with ID ' || customer\_id || ' already exists.');

    ELSE

        INSERT INTO Customer (customer\_id, customer\_name)

        VALUES (customer\_id, customer\_name);

        DBMS\_OUTPUT.PUT\_LINE('Customer added successfully.');

    END IF;

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Unexpected ERROR: ' || SQLERRM);

END AddNewCustomer;

/

BEGIN

    AddNewCustomer(NULL,NULL);

    AddNewCustomer(102, 'Reepthikha');

END;

/

OUTPUT:

Error: Customer ID and Name must not be NULL.  
Customer added successfully.  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.006

**Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

CREATE TABLE Employee(

    Employee\_id int,

    salary int,

    balance DECIMAL(10,2),

    department varchar(50)

);

insert into Employee(Employee\_id,salary,balance,department) values(1,50000,800,'worker');

insert into Employee(Employee\_id,salary,balance,department) values(2,80000,500,'manager');

insert into Employee(Employee\_id,salary,balance,department) values(3,90000,7000,'manager');

-- 1) Change ProcessMonthlyInterest

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest(

    Employee\_id IN NUMBER,

    salary IN NUMBER,

    balance IN OUT NUMBER,

    department IN VARCHAR2

) IS

    interest\_rate CONSTANT NUMBER := 5;

BEGIN

    balance := balance + (balance \* interest\_rate / 100);

    DBMS\_OUTPUT.PUT\_LINE('Updated balance for Employee ' || Employee\_id || ': ' || balance);

END ProcessMonthlyInterest;

/

-- 2) UpdateEmployeeBonus procedure

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

    Employee\_id IN NUMBER,

    salary IN OUT NUMBER,

    balance IN NUMBER,

    department IN VARCHAR2

) IS

    m\_bonus CONSTANT NUMBER := 5;

    w\_bonus CONSTANT NUMBER := 2;

BEGIN

    IF LOWER(department) = 'manager' THEN

        salary := salary + (salary \* m\_bonus / 100);

        DBMS\_OUTPUT.PUT\_LINE('Bonus added for Manager ' || Employee\_id || ': New salary = ' || salary);

    ELSIF LOWER(department) = 'worker' THEN

        salary := salary + (salary \* w\_bonus / 100);

        DBMS\_OUTPUT.PUT\_LINE('Bonus added for Worker ' || Employee\_id || ': New salary = ' || salary);

    ELSE

        DBMS\_OUTPUT.PUT\_LINE('No bonus: Department not recognized for Employee ' || Employee\_id);

    END IF;

END UpdateEmployeeBonus;

/

DECLARE

    v\_id Employee.Employee\_id%TYPE;

    v\_salary Employee.salary%TYPE;

    v\_balance Employee.balance%TYPE;

    v\_department Employee.department%TYPE;

    CURSOR c\_rec IS

        SELECT Employee\_id, salary, balance, department FROM Employee;

BEGIN

    OPEN c\_rec;

    LOOP

        FETCH c\_rec INTO v\_id, v\_salary, v\_balance, v\_department;

        EXIT WHEN c\_rec%NOTFOUND;

        -- Update balance by passing as IN OUT

        ProcessMonthlyInterest(v\_id, v\_salary, v\_balance, v\_department);

        -- Update salary by passing as IN OUT

        UpdateEmployeeBonus(v\_id, v\_salary, v\_balance, v\_department);

    END LOOP;

    CLOSE c\_rec;

    COMMIT;

END;

/

Updated balance for Employee 1: 840  
Bonus added for Worker 1: New salary = 51000  
Updated balance for Employee 2: 525  
Bonus added for Manager 2: New salary = 84000  
Updated balance for Employee 3: 7350  
Bonus added for Manager 3: New salary = 94500  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.014

**Scenario 3:** Customers should be able to transfer funds between their accounts.

create table customer(

    account\_id varchar(50),

    amount decimal(10,2)

);

insert into customer(account\_id, amount) values('Deepthikha',9000);

insert into customer(account\_id, amount) values('Kiruthiga',5000);

insert into customer(account\_id, amount) values('Jothi Ramakumar',35000);

insert into customer(account\_id, amount) values('Rama Kumar',100000);

CREATE OR REPLACE FUNCTION HasSufficientBalance(

    amount in number

) RETURN BOOLEAN IS

   SufficientBalance BOOLEAN;

BEGIN

    IF amount >= 10000 THEN

    SufficientBalance := true;

    DBMS\_OUTPUT.PUT\_LINE('SufficientBalance!');

    else

    SufficientBalance := FALSE;

    DBMS\_OUTPUT.PUT\_LINE(' Not SufficientBalance!');

    end if;

    RETURN SufficientBalance;

END HasSufficientBalance;

/

DECLARE

    v\_id customer.account\_id%TYPE;

    v\_balance customer.amount%TYPE;

    result BOOLEAN;

    cursor cursor\_record IS

        SELECT account\_id,amount FROM customer;

BEGIN

    OPEN cursor\_record;

    LOOP

        FETCH cursor\_record INTO v\_id, v\_balance;

        EXIT WHEN cursor\_record%NOTFOUND;

        result := HasSufficientBalance(v\_balance);

        DBMS\_OUTPUT.PUT\_LINE('Customer id: ' || v\_id);

        DBMS\_OUTPUT.PUT\_LINE('Result: ' || CASE WHEN result THEN 'TRUE' ELSE 'FALSE' END);

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

    END LOOP;

    CLOSE cursor\_record;

END;

/

Output:

Not SufficientBalance!  
Customer id: Deepthikha  
Result: FALSE  
----------------------------------------  
Not SufficientBalance!  
Customer id: Kiruthiga  
Result: FALSE  
----------------------------------------  
SufficientBalance!  
Customer id: Jothi Ramakumar  
Result: TRUE  
----------------------------------------  
SufficientBalance!  
Customer id: Rama Kumar  
Result: TRUE  
----------------------------------------  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.009

**Exercise 4: Functions**

**Scenario 1:** Calculate the age of customers for eligibility checks.

create table new\_customer(

    customer\_name varchar(50),

    customer\_dob date

);

insert into new\_customer(customer\_name,customer\_dob)values('Jothi Ramakumar',TO\_DATE('1976-06-25','YYYY-MM-DD'));

insert into new\_customer(customer\_name,customer\_dob)values('Ramakumar',TO\_DATE('1970-07-02','YYYY-MM-DD'));

insert into new\_customer(customer\_name,customer\_dob)values('Deepthikha',TO\_DATE('2005-01-06','YYYY-MM-DD'));

CREATE OR REPLACE FUNCTION calculateAge(customer\_dob IN DATE)

RETURN NUMBER IS

    current\_age NUMBER;

BEGIN

    current\_age := EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM customer\_dob);

    RETURN current\_age;

END calculateAge;

/

DECLARE

    v\_name new\_customer.customer\_name%TYPE;

    v\_dob new\_customer.customer\_dob%TYPE;

    display\_age INT;

    CURSOR cursor\_customer IS

        SELECT customer\_name, customer\_dob FROM new\_customer;

BEGIN

    OPEN cursor\_customer;

    LOOP

        FETCH cursor\_customer INTO v\_name, v\_dob;

        EXIT WHEN cursor\_customer%NOTFOUND;

        display\_age := calculateAge(v\_dob);

        DBMS\_OUTPUT.PUT\_LINE('Customer name: ' || v\_name);

        DBMS\_OUTPUT.PUT\_LINE('Customer current age: ' || display\_age);

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

    END LOOP;

    CLOSE cursor\_customer;

END;

/

OUTPUT:

Customer name: Deepthikha  
Customer current age: 20  
------------------------------  
Customer name: Jothi Ramakumar  
Customer current age: 49  
------------------------------  
Customer name: Ramakumar  
Customer current age: 55  
------------------------------

PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.003  
**Scenario 2:** The bank needs to compute the monthly installment for a loan

create table customer(

    customer\_name varchar(50),

    loan\_amount decimal(10,2),

    interest\_rate decimal(10,2),

    loan\_duration int

);

insert into customer(customer\_name,loan\_amount, interest\_rate, loan\_duration) values('Deepthikha',10000,12,2);

insert into customer(customer\_name,loan\_amount, interest\_rate, loan\_duration) values('Kiruthiga',50000,35,5);

insert into customer(customer\_name,loan\_amount, interest\_rate, loan\_duration) values('Jothi Ramakumar',35000,12,4);

insert into customer(customer\_name,loan\_amount, interest\_rate, loan\_duration) values('Rama Kumar',100000,562,6);

CREATE OR REPLACE FUNCTION monthlyinstallation(

    loan\_amount IN NUMBER,

    interest\_rate IN NUMBER,

    loan\_duration IN NUMBER

) RETURN NUMBER IS

    int\_monthlyinstallment NUMBER;

BEGIN

    int\_monthlyinstallment := (loan\_amount + (loan\_amount \* interest\_rate \* loan\_duration / 100)) / (loan\_duration \* 12);

    RETURN int\_monthlyinstallment;

END monthlyinstallation;

/

DECLARE

v\_name customer.customer\_name%type;

v\_amount customer.loan\_amount%type;

v\_interest customer.interest\_rate %type;

v\_duration customer.loan\_duration%type;

result NUMBER;

cursor cursor\_record is

select customer\_name,loan\_amount, interest\_rate, loan\_duration from customer;

BEGIN

    open cursor\_record;

    LOOP

        fetch cursor\_record into v\_name,v\_amount,v\_interest,v\_duration ;

        exit when cursor\_record%NOTFOUND;

        result := monthlyinstallation(v\_amount,v\_interest,v\_duration);

        dbms\_output.put\_line('Customer name:'|| v\_name);

        dbms\_output.put\_line('Monthly installation'|| ROUND(result, 2));

        dbms\_output.put\_line('----------------------------------------');

    END LOOP;

    close cursor\_record;

END;

/

Output:

Customer name:Deepthikha  
Monthly installation516.67  
----------------------------------------  
Customer name:Kiruthiga  
Monthly installation2291.67  
----------------------------------------  
Customer name:Jothi Ramakumar  
Monthly installation1079.17  
----------------------------------------  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.004

**Scenario 3:** Check if a customer has sufficient balance before making a transaction.

create table customer(

    account\_id varchar(50),

    amount decimal(10,2)

);

insert into customer(account\_id, amount) values('Deepthikha',9000);

insert into customer(account\_id, amount) values('Kiruthiga',5000);

insert into customer(account\_id, amount) values('Jothi Ramakumar',35000);

insert into customer(account\_id, amount) values('Rama Kumar',100000);

CREATE OR REPLACE FUNCTION HasSufficientBalance(

    amount in number

) RETURN BOOLEAN IS

   SufficientBalance BOOLEAN;

BEGIN

    IF amount >= 10000 THEN

    SufficientBalance := true;

    DBMS\_OUTPUT.PUT\_LINE('SufficientBalance!');

    else

    SufficientBalance := FALSE;

    DBMS\_OUTPUT.PUT\_LINE(' Not SufficientBalance!');

    end if;

    RETURN SufficientBalance;

END HasSufficientBalance;

/

DECLARE

    v\_id customer.account\_id%TYPE;

    v\_balance customer.amount%TYPE;

    result BOOLEAN;

    cursor cursor\_record IS

        SELECT account\_id,amount FROM customer;

BEGIN

    OPEN cursor\_record;

    LOOP

        FETCH cursor\_record INTO v\_id, v\_balance;

        EXIT WHEN cursor\_record%NOTFOUND;

        result := HasSufficientBalance(v\_balance);

        DBMS\_OUTPUT.PUT\_LINE('Customer id: ' || v\_id);

        DBMS\_OUTPUT.PUT\_LINE('Result: ' || CASE WHEN result THEN 'TRUE' ELSE 'FALSE' END);

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

    END LOOP;

    CLOSE cursor\_record;

END;

/

Output:

Not SufficientBalance!  
Customer id: Deepthikha  
Result: FALSE  
----------------------------------------  
Not SufficientBalance!  
Customer id: Kiruthiga  
Result: FALSE  
----------------------------------------  
SufficientBalance!  
Customer id: Jothi Ramakumar  
Result: TRUE  
----------------------------------------  
SufficientBalance!  
Customer id: Rama Kumar  
Result: TRUE  
----------------------------------------  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.009

**Exercise 6: Cursors**

**Scenario 1:** Generate monthly statements for all customers.

CREATE TABLE Customer(

    Customer\_id INT,

    Customer\_name VARCHAR(50),

    Customer\_total\_transaction INT

);

INSERT INTO Customer( Customer\_id, Customer\_name,Customer\_total\_transaction)

VALUES(1,'Yashwanth saran',30);

INSERT INTO Customer( Customer\_id, Customer\_name,Customer\_total\_transaction)

VALUES(2,'Deepthikha ',10);

INSERT INTO Customer( Customer\_id, Customer\_name,Customer\_total\_transaction)

VALUES(3,'Jothi Ramakumar',5);

DECLARE

v\_id Customer.Customer\_id%TYPE;

v\_name Customer.Customer\_name%TYPE;

v\_transaction Customer.Customer\_total\_transaction%TYPE;

CURSOR Cursor\_customer IS

SELECT  Customer\_id, Customer\_name,Customer\_total\_transaction FROM Customer;

BEGIN

    OPEN Cursor\_customer;

    IF Cursor\_customer%ISOPEN THEN

    DBMS\_OUTPUT.PUT\_LINE('CURSOR IS OPEN.');

    END IF;

    LOOP

    FETCH Cursor\_customer INTO v\_id,v\_name,v\_transaction;

    EXIT  WHEN Cursor\_customer%NOTFOUND;

    IF Cursor\_customer%FOUND THEN

    DBMS\_OUTPUT.PUT\_LINE('Customer id : '||v\_id || ' Customer Name :' ||v\_name );

    DBMS\_OUTPUT.PUT\_LINE('TOTAL TRACTIONS FOR CURRENT MONTH:' || v\_transaction );

    END IF;

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

    END LOOP;

    CLOSE Cursor\_customer;

     IF NOT Cursor\_customer%ISOPEN THEN

        DBMS\_OUTPUT.PUT\_LINE('CURSOR IS NOW CLOSED.');

    END IF;

END;

/

OUTPUT:

Customer id : 1 Customer Name :Yashwanth saran  
TOTAL TRACTIONS FOR CURRENT MONTH:30  
------------------------------------------  
Customer id : 2 Customer Name :Deepthikha   
TOTAL TRACTIONS FOR CURRENT MONTH:10  
------------------------------------------  
Customer id : 3 Customer Name :Jothi Ramakumar  
TOTAL TRACTIONS FOR CURRENT MONTH:5  
------------------------------------------  
CURSOR IS NOW CLOSED.  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.003

**Scenario 2:** Apply annual fee to all accounts.

CREATE TABLE Customer(

    Customer\_id INT,

    Customer\_name VARCHAR(50),

    Customer\_balance INTEGER

);

INSERT INTO Customer( Customer\_id, Customer\_name, Customer\_balance)

VALUES(1,'Yashwanth saran',100);

INSERT INTO Customer( Customer\_id, Customer\_name, Customer\_balance)

VALUES(2,'Deepthikha ',5000);

INSERT INTO Customer( Customer\_id, Customer\_name, Customer\_balance)

VALUES(3,'Jothi Ramakumar',50000);

DECLARE

v\_id Customer.Customer\_id%TYPE;

v\_name Customer.Customer\_name%TYPE;

v\_BALANCE Customer.Customer\_balance%TYPE;

APPLY\_FEE\_BELOW\_5000 INTEGER := 300;  APPLY\_FEE\_ABOVE\_5000 INTEGER := 1000;

CURSOR Cursor\_customer IS

SELECT  Customer\_id, Customer\_name, Customer\_balance FROM Customer;

BEGIN

    OPEN Cursor\_customer;

    IF Cursor\_customer%ISOPEN THEN

    DBMS\_OUTPUT.PUT\_LINE('CURSOR IS OPEN.');

    END IF;

    LOOP

    FETCH Cursor\_customer INTO v\_id,v\_name,v\_balance;

    EXIT  WHEN Cursor\_customer%NOTFOUND;

    IF v\_balance <= 5000 THEN

     DBMS\_OUTPUT.PUT\_LINE('Customer Id: ' || v\_id);

     DBMS\_OUTPUT.PUT\_LINE('Customer Name:'||v\_name);

     DBMS\_OUTPUT.PUT\_LINE('Customer Balance:'||v\_balance);

    DBMS\_OUTPUT.PUT\_LINE('APPLY ANNUAL MAINTENENCE FEE :' || APPLY\_FEE\_BELOW\_5000 );

    ELSE

     DBMS\_OUTPUT.PUT\_LINE('Customer Id: ' || v\_id);

     DBMS\_OUTPUT.PUT\_LINE('Customer Name:'||v\_name);

     DBMS\_OUTPUT.PUT\_LINE('Customer Balance:'||v\_balance);

    DBMS\_OUTPUT.PUT\_LINE('APPLY ANNUAL MAINTENENCE FEE :' || APPLY\_FEE\_ABOVE\_5000);

     END IF;

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

    END LOOP;

    CLOSE Cursor\_customer;

     IF NOT Cursor\_customer%ISOPEN THEN

        DBMS\_OUTPUT.PUT\_LINE('CURSOR IS NOW CLOSED.');

    END IF;

END;

/

DROP TABLE Customer;

Output:

CURSOR IS OPEN.  
Customer Id: 1  
Customer Name:Yashwanth saran  
Customer Balance:100  
APPLY ANNUAL MAINTENENCE FEE :300  
------------------------------------------  
Customer Id: 2  
Customer Name:Deepthikha   
Customer Balance:5000  
APPLY ANNUAL MAINTENENCE FEE :300  
------------------------------------------  
Customer Id: 3  
Customer Name:Jothi Ramakumar  
Customer Balance:50000  
APPLY ANNUAL MAINTENENCE FEE :1000  
------------------------------------------  
CURSOR IS NOW CLOSED.  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.004

**Scenario 3:** Update the interest rate for all loans based on a new policy.

CREATE TABLE Customer(

    Customer\_id INT,

    Customer\_name VARCHAR(50),

    Customer\_LoanType VARCHAR(50)

);

INSERT INTO Customer( Customer\_id, Customer\_name, Customer\_LoanType)

VALUES(1,'Yashwanth saran','Agriculture policy');

INSERT INTO Customer( Customer\_id, Customer\_name,Customer\_LoanType)

VALUES(2,'Deepthikha ','Education policy');

INSERT INTO Customer( Customer\_id, Customer\_name, Customer\_LoanType)

VALUES(3,'Jothi Ramakumar','Business policy');

DECLARE

v\_id Customer.Customer\_id%TYPE;

v\_name Customer.Customer\_name%TYPE;

v\_LoanType Customer.Customer\_LoanType%TYPE;

 Agriculture\_policy INTEGER := 5;

 Education\_policy NUMBER := 2.5;

 Business\_policy INTEGER := 8;

CURSOR Cursor\_customer IS

SELECT  Customer\_id, Customer\_name,  Customer\_LoanType FROM Customer;

BEGIN

    OPEN Cursor\_customer;

    IF Cursor\_customer%ISOPEN THEN

    DBMS\_OUTPUT.PUT\_LINE('CURSOR IS OPEN');

    END IF;

    LOOP

    FETCH Cursor\_customer INTO v\_id,v\_name,v\_LoanType;

    EXIT  WHEN Cursor\_customer%NOTFOUND;

    IF v\_LoanType = 'Agriculture policy' THEN

     DBMS\_OUTPUT.PUT\_LINE('Customer Id: ' || v\_id);

     DBMS\_OUTPUT.PUT\_LINE('Customer Name:'||v\_name);

    DBMS\_OUTPUT.PUT\_LINE('Updated interest rates based on the policy.:' ||  Agriculture\_policy ||'%' );

    ELSIF v\_LoanType = 'Education policy' THEN

     DBMS\_OUTPUT.PUT\_LINE('Customer Id: ' || v\_id);

     DBMS\_OUTPUT.PUT\_LINE('Customer Name:'||v\_name);

    DBMS\_OUTPUT.PUT\_LINE('Updated interest rates based on the policy.:' ||Education\_policy ||'%');

    ELSIF v\_LoanType =  'Business policy'  THEN

    DBMS\_OUTPUT.PUT\_LINE('Customer Id: ' || v\_id);

     DBMS\_OUTPUT.PUT\_LINE('Customer Name:'||v\_name);

    DBMS\_OUTPUT.PUT\_LINE('Updated interest rates based on the policy.:' || Business\_policy ||'%');

     END IF;

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

    END LOOP;

    CLOSE Cursor\_customer;

     IF NOT Cursor\_customer%ISOPEN THEN

        DBMS\_OUTPUT.PUT\_LINE('CURSOR IS NOW CLOSED.');

    END IF;

END;

/

Output:

CURSOR IS OPEN  
Customer Id: 1  
Customer Name:Yashwanth saran  
Updated interest rates based on the policy.:5%  
------------------------------------------  
Customer Id: 2  
Customer Name:Deepthikha   
Updated interest rates based on the policy.:2.5%  
------------------------------------------  
Customer Id: 3  
Customer Name:Jothi Ramakumar  
Updated interest rates based on the policy.:8%  
------------------------------------------  
CURSOR IS NOW CLOSED.  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.005