**Exercise 1: Control Structures**

**Scenario 1:**

DECLARE

c\_dob number;

currentDate number;

c\_id customers.customerid%type;

total\_rows number := 0;

CURSOR ageCalc IS

SELECT CustomerID,EXTRACT(YEAR from DOB) from customers;

BEGIN

OPEN ageCalc;

currentDate := EXTRACT(YEAR from SYSDATE);

LOOP

FETCH ageCalc into c\_id, c\_dob;

EXIT WHEN ageCalc%notfound;

IF(currentDate - c\_dob >= 60)THEN

--provide discount of 1% to loans

UPDATE loans

SET interestrate = interestrate-1

WHERE customerid = c\_id;

--check number of rows affected

total\_rows := total\_rows + sql%rowcount;

END IF;

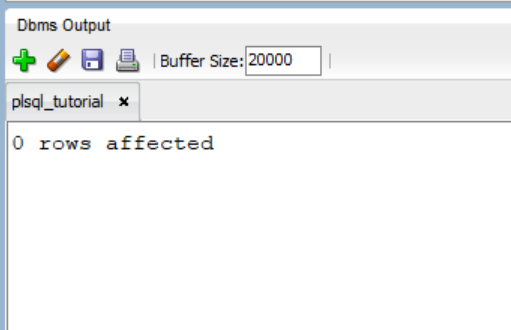
END LOOP;

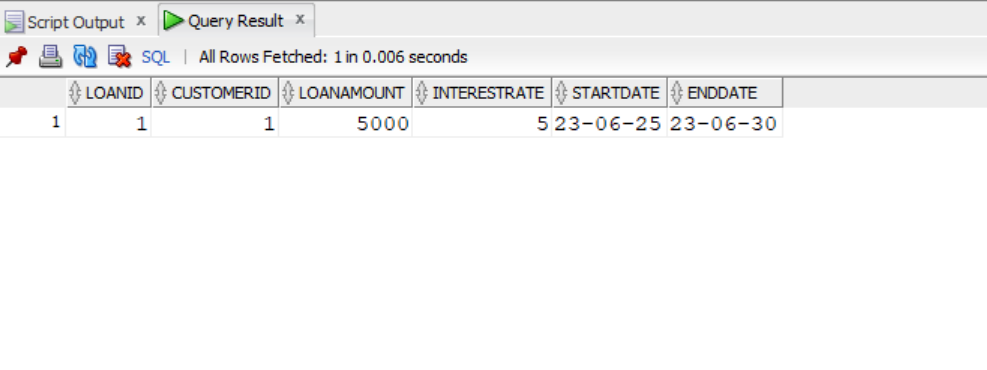
dbms\_output.put\_line(total\_rows || ' rows affected');

CLOSE ageCalc;

END;

**OUTPUT:**

****

* ***Loans* Table before and after the query remains the same:**

**Scenario 2:**

--add isVip column to customers table

ALTER TABLE customers ADD isVip VARCHAR2(5);

DECLARE

CURSOR vipCheck IS

SELECT customerid,balance FROM customers;

BEGIN

FOR c IN vipCheck

LOOP

IF c.balance > 10000 THEN

UPDATE customers

SET isVip = 'TRUE'

WHERE customerid = c.customerid;

ELSE

UPDATE customers

SET isVip = 'FALSE'

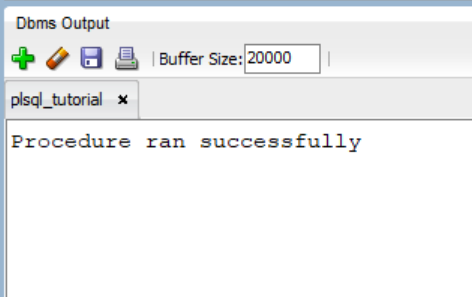
WHERE customerid = c.customerid;

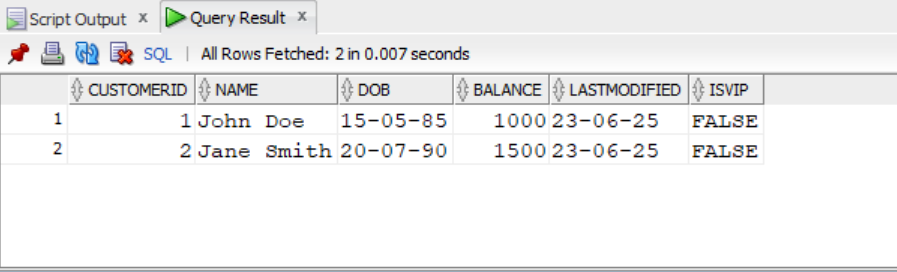
END IF;

END LOOP;

dbms\_output.put\_line('Procedure ran successfully');

END;

**OUTPUT:**

***Customers table after running above procedure:***

**Scenario 3:**

DECLARE

daysLeft number;

monthsLeft number;

CURSOR loanData IS

SELECT customerid,loanamount,EXTRACT(DAY FROM startdate) AS startdate FROM loans;

BEGIN

FOR c in loanData

LOOP

daysLeft := c.startdate - EXTRACT(DAY FROM SYSDATE);

--if number of days left is negative, we add 30 to it

IF(daysLeft < 0) THEN

daysLeft := daysLeft + 30;

END IF;

--check if days left is less than 30

IF (daysleft<= 30) THEN

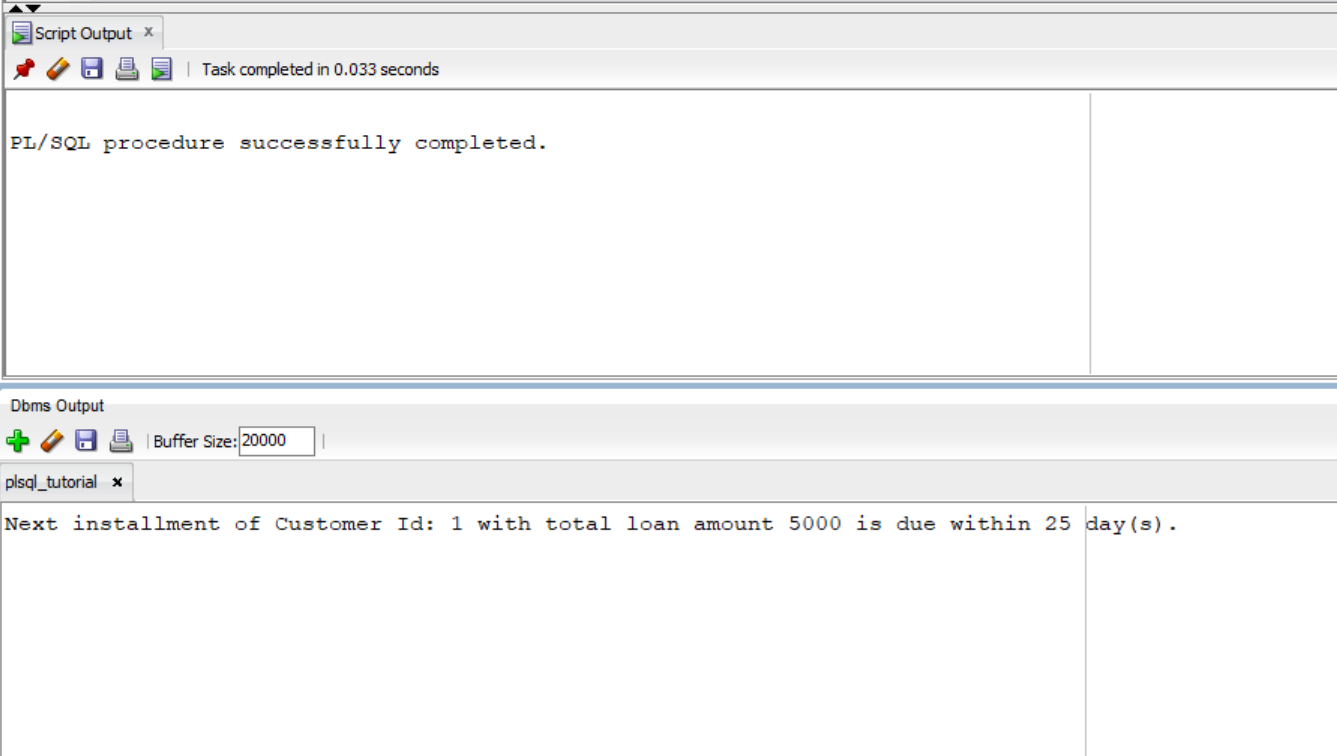
dbms\_output.put\_line('Next installment of Customer Id: '|| c.customerid || ' with total loan amount ' || c.loanamount || ' is due within ' ||daysLeft || ' day(s).');

END IF;

END LOOP;

END;

**OUTPUT:**



**Exercise 3: Stored Procedures**

**Scenario 1:**

CODE:

--procedure to update balance

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

IS

cid NUMBER;

newAmount NUMBER;

cust\_balance NUMBER;

accountType accounts.accounttype%type;

acc\_balance NUMBER;

CURSOR fetchDetails IS

SELECT customers.customerid,customers.balance, accounts.balance,accounts.accounttype FROM customers

INNER JOIN accounts

ON customers.customerid = accounts.customerid;

BEGIN

OPEN fetchDetails;

LOOP

FETCH fetchDetails into cid,cust\_balance, acc\_balance,accountType;

EXIT WHEN fetchDetails%notfound;

--check for savings account

IF (upper(accountType) = 'SAVINGS') THEN

newamount := (cust\_balance \* 0.01)/12;

--update customers table

UPDATE customers

SET balance = balance + newamount

WHERE customerid = cid;

--update accounts table

UPDATE accounts

SET balance = balance + newamount

WHERE customerid = cid;

END IF;

END LOOP;

CLOSE fetchDetails;

END;

/

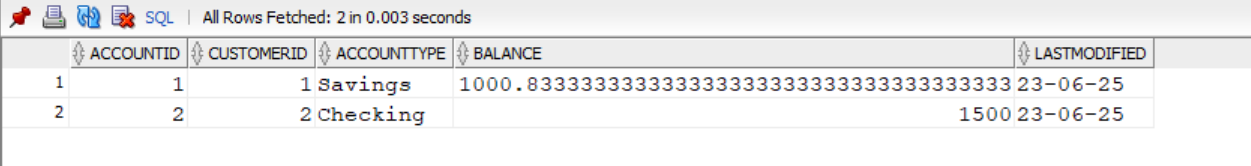
BEGIN

processmonthlyinterest;

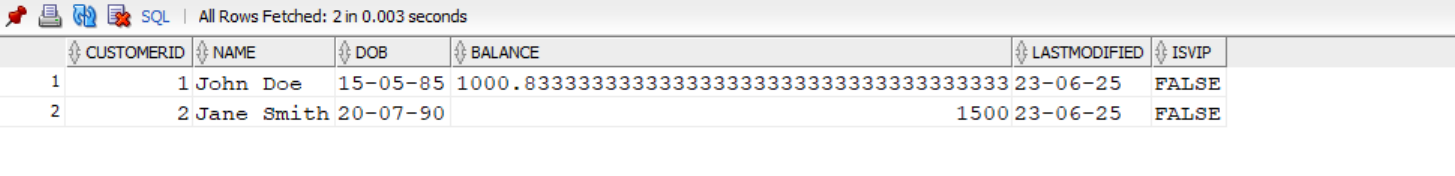
END;

**OUTPUT:**

Accounts table:

****

Customers table:



**Scenario 2:**

CODE:

--procedure to add bonus

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (bonusPercent IN NUMBER, dept IN VARCHAR2)

IS

newSalary employees.salary%type;

BEGIN

FOR c in (SELECT employeeid,salary FROM employees WHERE department = dept) LOOP

newSalary := c.salary \* (1+ (bonusPercent/100));

--update table

UPDATE employees

SET salary = newSalary

WHERE employeeid = c.employeeid;

END LOOP;

END;

/

BEGIN

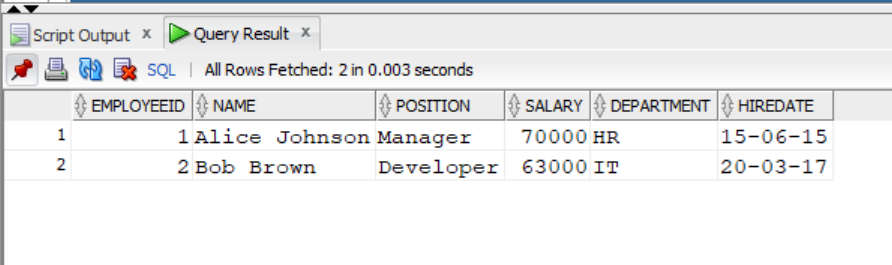
--bonus is 5% and department is IT

updateemployeebonus(5,'IT');

END;

OUTPUT:

*Employee table:*

**

**Scenario 3:**

CODE:

--procedure to add bonus

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (bonusPercent IN NUMBER, dept IN VARCHAR2)

IS

newSalary employees.salary%type;

BEGIN

FOR c in (SELECT employeeid,salary FROM employees WHERE department = dept) LOOP

newSalary := c.salary \* (1+ (bonusPercent/100));

--update table

UPDATE employees

SET salary = newSalary

WHERE employeeid = c.employeeid;

END LOOP;

END;

/

BEGIN

--bonus is 5% and department is IT

updateemployeebonus(5,'IT');

END;

--Scenario 3

--procedure to add bonus

CREATE OR REPLACE PROCEDURE TransferFunds (sender\_id IN NUMBER, receiver\_id IN NUMBER, transfer\_amount IN NUMBER)

IS

sender\_balance accounts.balance%type;

sender\_cust\_id accounts.customerid%type;

receiver\_cust\_id accounts.customerid%type;

CURSOR sender IS

SELECT balance, customerid FROM accounts WHERE accountid = sender\_id;

CURSOR receiver IS

SELECT customerid FROM accounts WHERE accountid = receiver\_id;

BEGIN

--fetch sender details

OPEN sender;

FETCH sender INTO sender\_balance, sender\_cust\_id;

CLOSE sender;

--fetch receiver details

OPEN receiver;

FETCH receiver INTO receiver\_cust\_id;

CLOSE receiver;

IF sender\_balance < transfer\_amount THEN

dbms\_output.put\_line('Balance is insufficient');

ELSE

--update sender

UPDATE accounts

SET balance = balance - transfer\_amount

WHERE accountid = sender\_id;

UPDATE customers

SET balance = balance - transfer\_amount

WHERE customerid = sender\_cust\_id;

--update receiver

UPDATE accounts

SET balance = balance + transfer\_amount

WHERE accountid = receiver\_id;

UPDATE customers

SET balance = balance + transfer\_amount

WHERE customerid = receiver\_cust\_id;

END IF;

END;

/

BEGIN

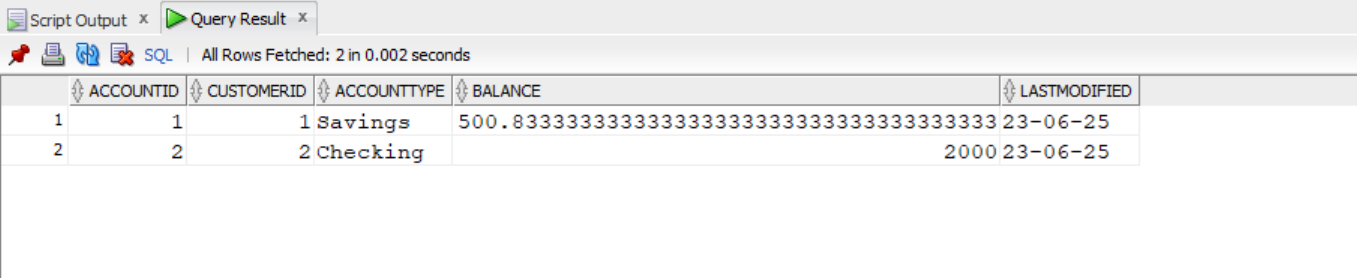
--transfer amount of 500 from account id 1 to account id 2

transferfunds(1,2,500);

END;

OUTPUT:

Accounts table:



Employees table:

