**PL/SQL Solutions**

**Candidate Name: Samarth Chourey  
Superset ID: 6394625**

**Exercise 1: Control Structures**

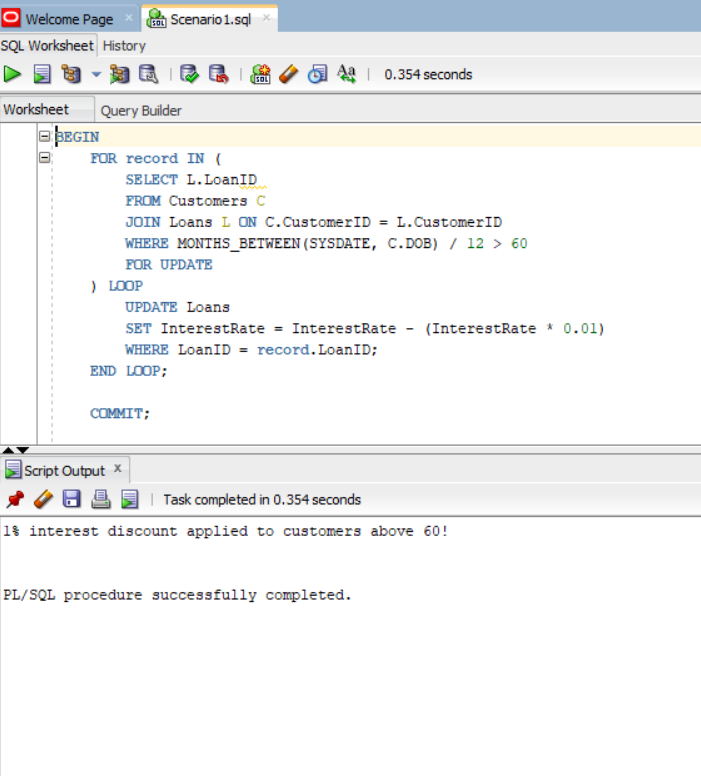
**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* + **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.

**SOLUTION SCRIPT:**

BEGIN FOR record IN (  
 SELECT L.LoanID  
 FROM Customers C  
 JOIN Loans L ON C.CustomerID = L.CustomerID  
 WHERE MONTHS\_BETWEEN(SYSDATE, C.DOB) / 12 > 60  
 FOR UPDATE  
 ) LOOP  
 UPDATE Loans   
 SET InterestRate = InterestRate - (InterestRate \* 0.01)  
 WHERE LoanID = record.LoanID;  
 END LOOP;  
  
 COMMIT;  
  
 DBMS\_OUTPUT.PUT\_LINE('1% interest discount applied to customers above 60!');  
END;

**OUTPUT:**

****

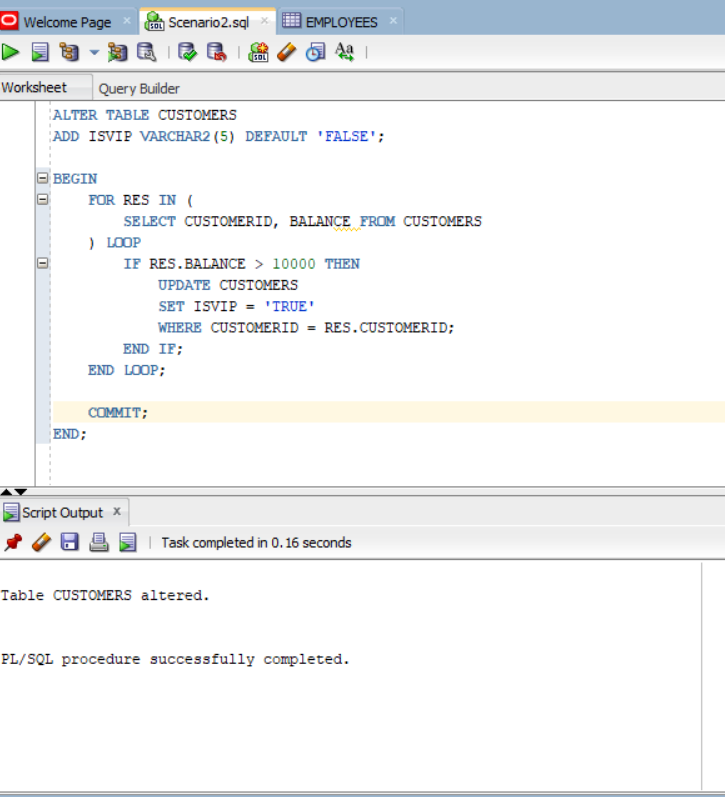
**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* + **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.

**SOLUTION SCRIPT:**

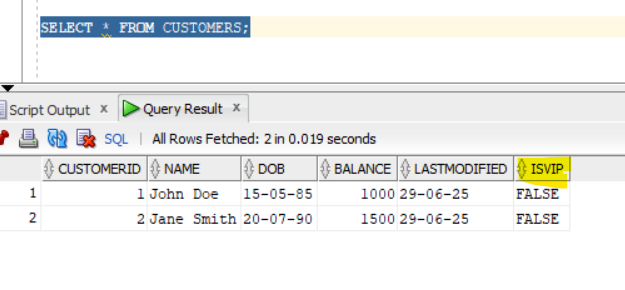
ALTER TABLE CUSTOMERS  
ADD ISVIP VARCHAR2(5) DEFAULT 'FALSE';  
  
BEGIN  
 FOR RES IN (  
 SELECT CUSTOMERID, BALANCE FROM CUSTOMERS  
 ) LOOP  
 IF RES.BALANCE > 10000 THEN  
 UPDATE CUSTOMERS  
 SET ISVIP = 'TRUE'  
 WHERE CUSTOMERID = RES.CUSTOMERID;  
 END IF;  
 END LOOP;  
  
 COMMIT;  
END;

**OUTPUT:**



**FOR EXECUTION:**

**SELECT \* FROM CUSTOMERS;**

****

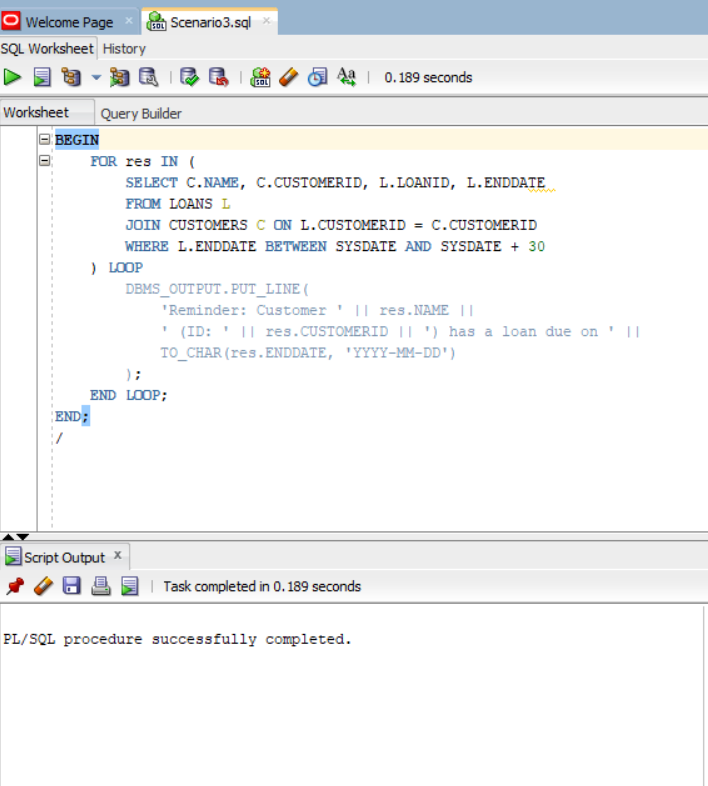
**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* + **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**SOLUTION SCRIPT:**

BEGIN  
 FOR res IN (  
 SELECT C.NAME, C.CUSTOMERID, 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: Customer ' || res.NAME ||  
 ' (ID: ' || res.CUSTOMERID || ') has a loan due on ' ||  
 TO\_CHAR(res.ENDDATE, 'YYYY-MM-DD')  
 );  
 END LOOP;  
END;  
/

**OUTPUT:**

****

**Exercise 3: Stored Procedures**

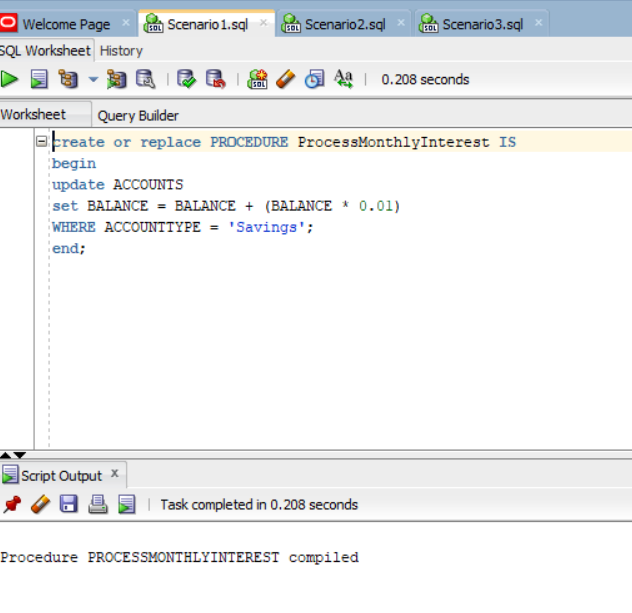
**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

* + **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.

**SOLUTION SCRIPT:**

create or replace PROCEDURE ProcessMonthlyInterest IS  
begin  
update ACCOUNTS  
set BALANCE = BALANCE + (BALANCE \* 0.01)  
WHERE ACCOUNTTYPE = 'Savings';  
end;

**OUTPUT:**

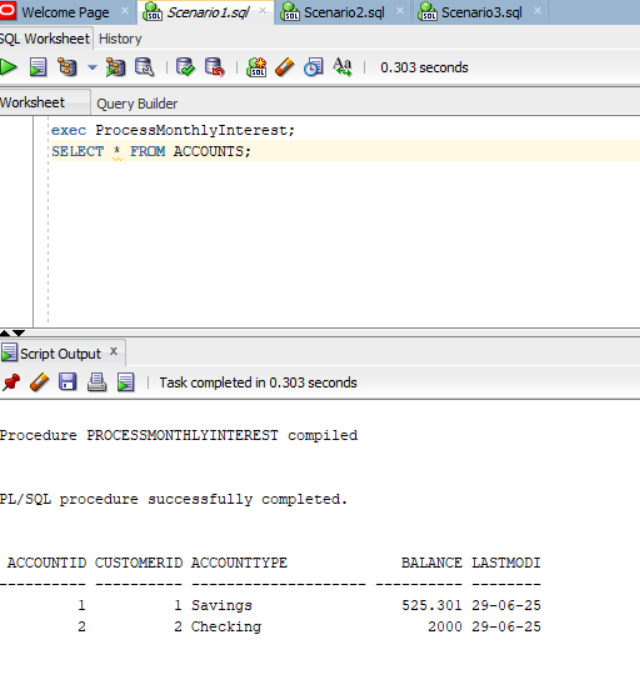


**FOR EXECUTION:**

**SET SERVEROUTPUT ON;**

**exec ProcessMonthlyInterest;**

**Select \* from ACCOUNTS;**

****

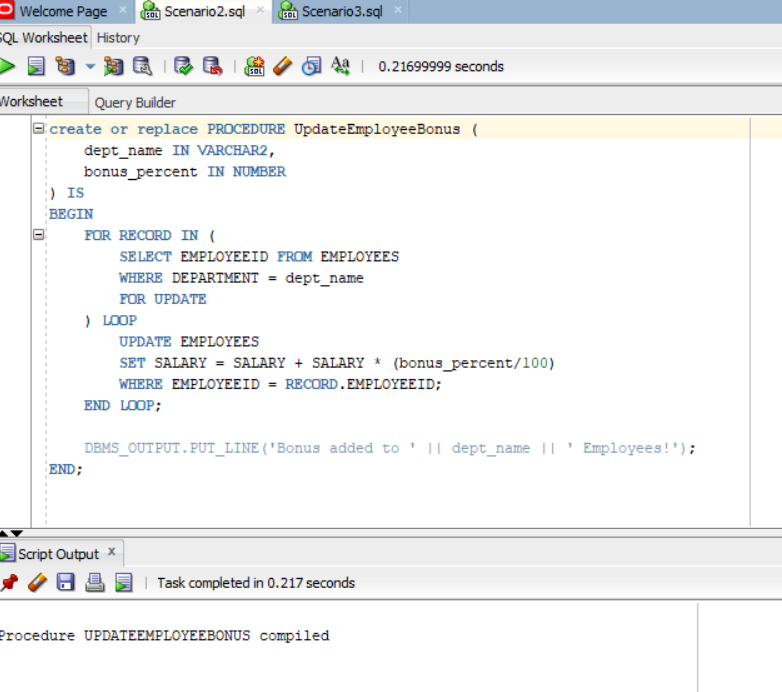
**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

* + **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.

**SOLUTION SCRIPT:**

create or replace PROCEDURE UpdateEmployeeBonus (  
 dept\_name IN VARCHAR2,  
 bonus\_percent IN NUMBER  
) IS  
BEGIN  
 FOR RECORD IN (  
 SELECT EMPLOYEEID FROM EMPLOYEES  
 WHERE DEPARTMENT = dept\_name  
 FOR UPDATE  
 ) LOOP  
 UPDATE EMPLOYEES  
 SET SALARY = SALARY + SALARY \* (bonus\_percent/100)  
 WHERE EMPLOYEEID = RECORD.EMPLOYEEID;  
 END LOOP;  
  
 DBMS\_OUTPUT.PUT\_LINE('Bonus added to ' || dept\_name || ' Employees!');  
END;

**OUTPUT:**



**FOR EXECUTION:**

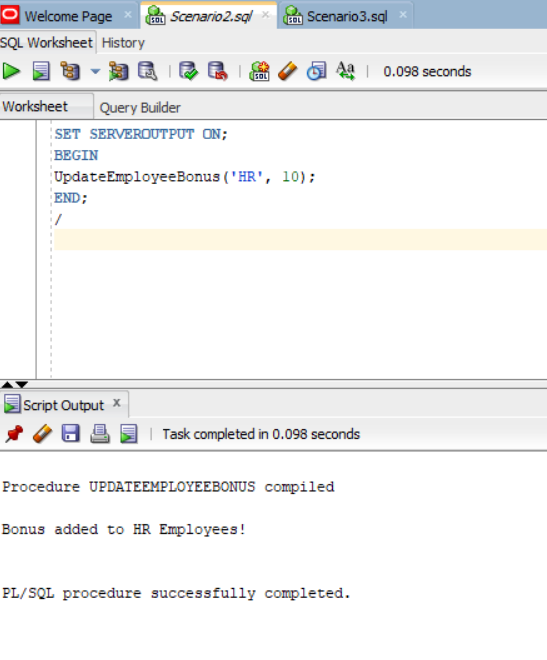
**SET SERVEROUTPUT ON;**

**BEGIN**

**UpdateEmployeeBonus('HR', 10);**

**END;**

**/**

****

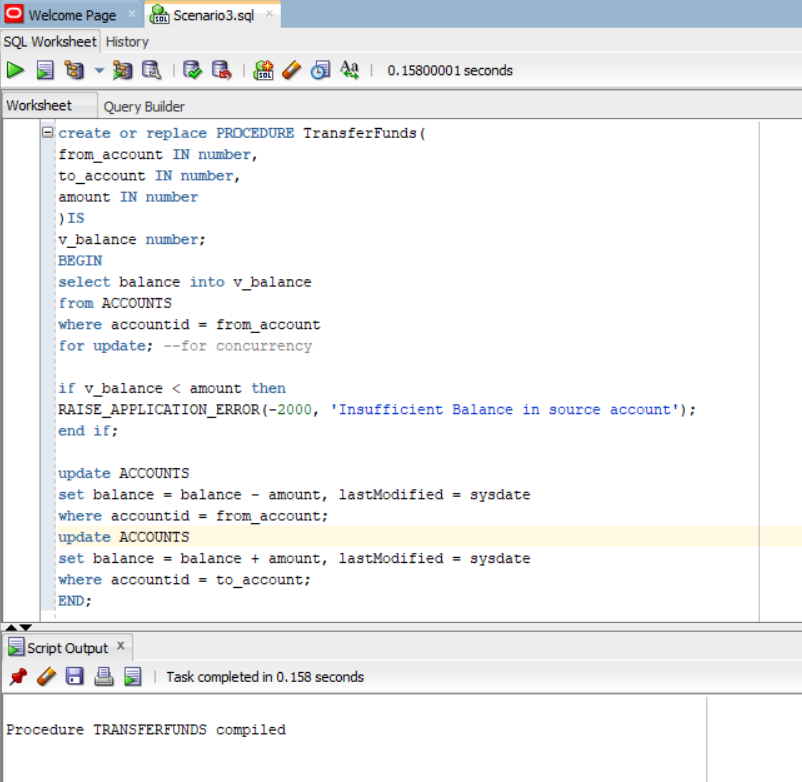
**Scenario 3:** Customers should be able to transfer funds between their accounts.

* + **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.

**SOLUTION SCRIPT:**

create or replace PROCEDURE TransferFunds(  
from\_account IN number,  
to\_account IN number,  
amount IN number  
)IS  
v\_balance number;  
BEGIN  
select balance into v\_balance  
from ACCOUNTS  
where accountid = from\_account  
for update; --for concurrency  
  
if v\_balance < amount then  
RAISE\_APPLICATION\_ERROR(-2000, 'Insufficient Balance in source account');  
end if;  
  
update ACCOUNTS  
set balance = balance - amount, lastModified = sysdate  
where accountid = from\_account;  
update ACCOUNTS  
set balance = balance + amount, lastModified = sysdate  
where accountid = to\_account;  
END;

**OUTPUT:**

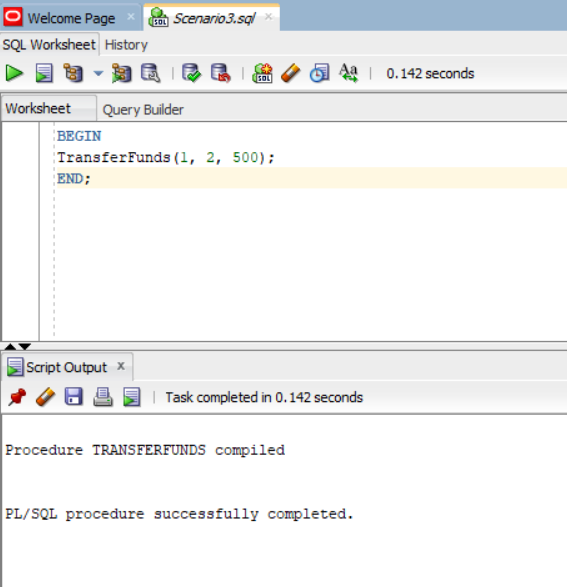
****

**FOR EXECUTION:**

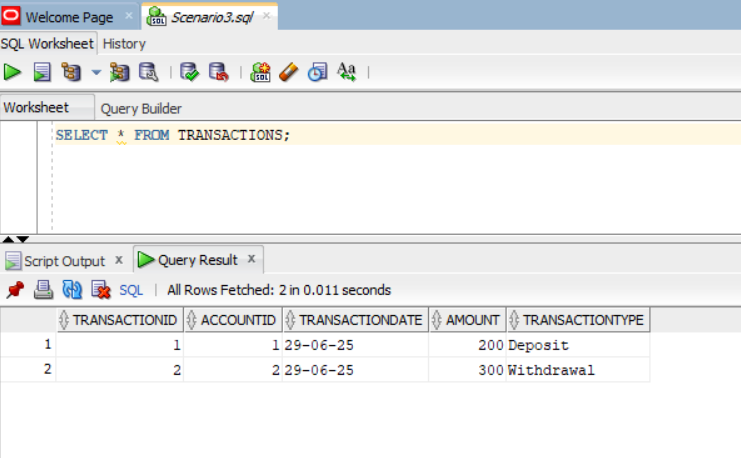
**begin**

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

**end;**



**SELECT \* FROM TRANSACTIONS;**

****