**Module 3**

**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.

**Answer**

Declare

cursor cust\_cursor is select CustomerID,DOB from Customers;

v\_cusId Customers.CustomerID%type;

v\_dob Customers.dob%type;

v\_age NUMBER;

begin

for cust\_rec in cust\_cursor loop

v\_cusId := cust\_rec.CustomerID;

v\_dob := cust\_rec.dob;

v\_age := floor(Months\_between(SYSDATE,v\_dob)/12);

if v\_age >60 then

update Loans

set InterestRate=InterestRate-1

where CustomerID=v\_cusId;

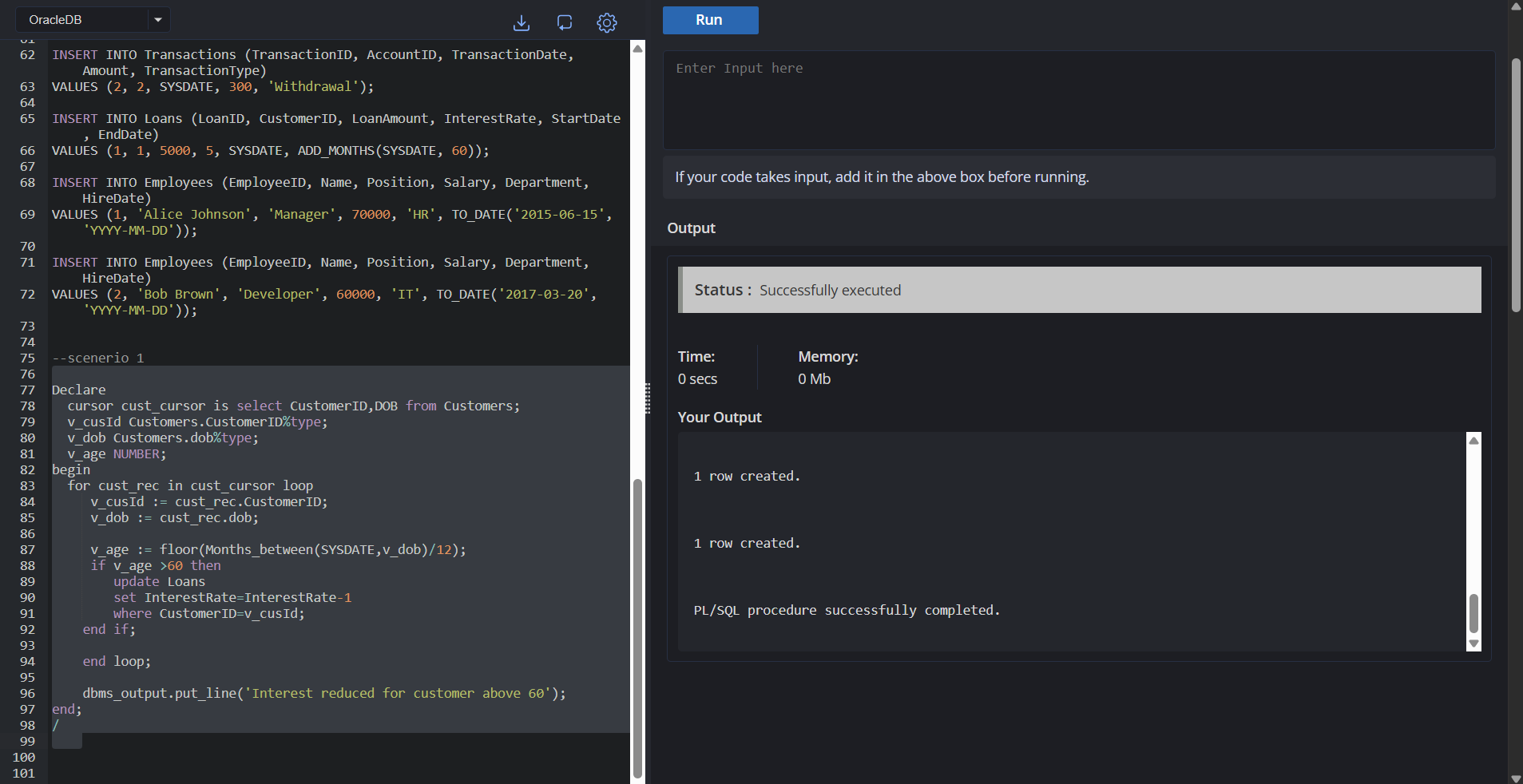
end if;

end loop;

dbms\_output.put\_line('Interest reduced for customer above 60');

end;

/



**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.

**Answer**

alter table Customers add IsVip char(1);

Declare

cursor cust\_cursor is select CustomerID ,Balance from Customers;

v\_cusId Customers.CustomerID%type;

v\_balance Customers.Balance%type;

begin

for cust\_rec in cust\_cursor loop

v\_cusId := cust\_rec.CustomerID;

v\_balance := cust\_rec.Balance;

if v\_balance >10000 then

update customers set IsVip='Y' where CustomerID=v\_cusId;

else

update customers set IsVip='N' where CustomerID=v\_cusId;

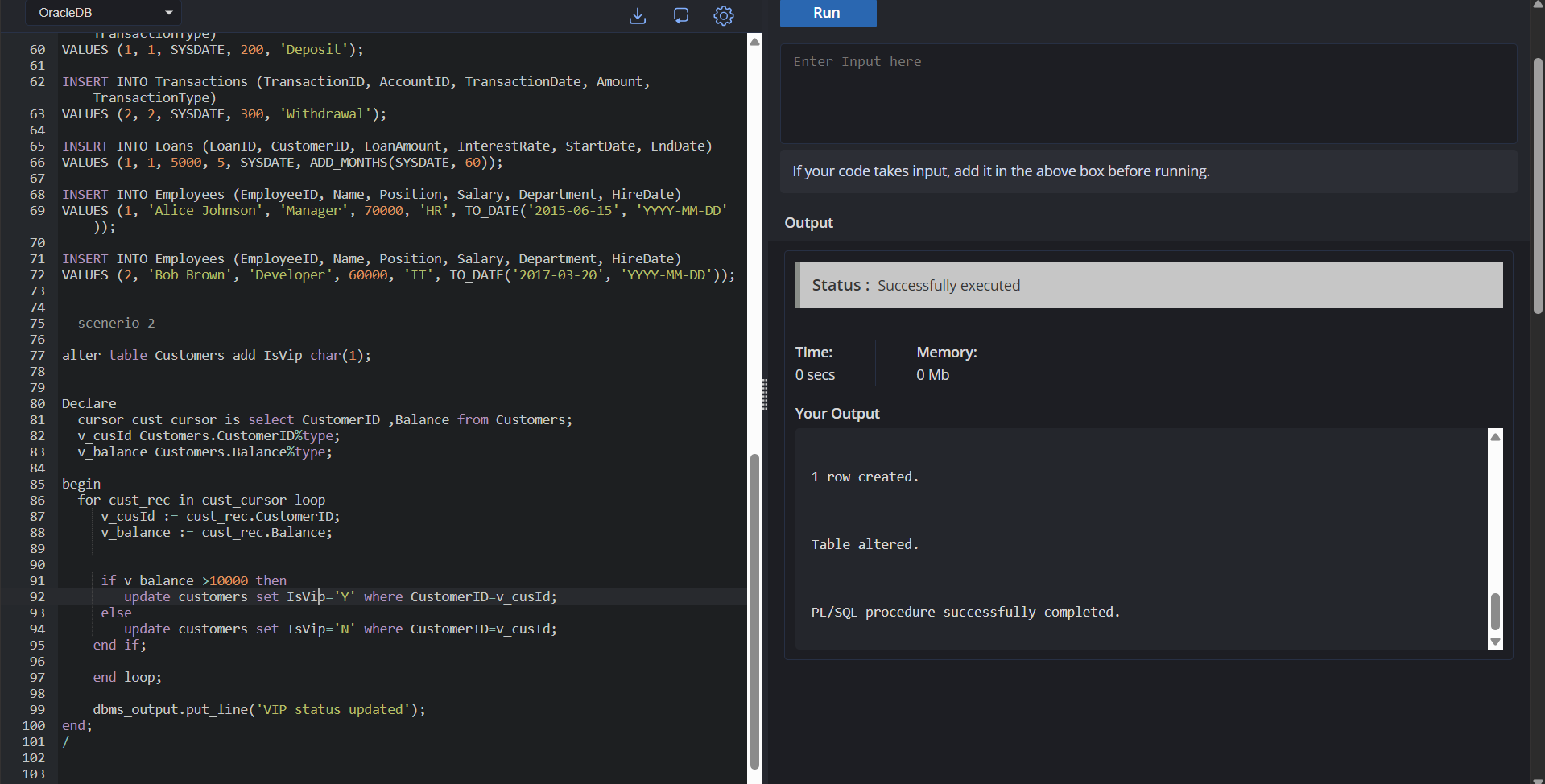
end if;

end loop;

dbms\_output.put\_line('VIP status updated');

end;

/



**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.

**ANSWER**

declare

cursor due\_loans\_cursor is

select l.LoanID,l.CustomerID,l.EndDate,c.Name from loans l join

Customers c on l.CustomerID=c.CustomerID

where l.EndDate between SYSDATE and SYSDATE+30;

v\_loadId loans.LoanID%type;

v\_cusId loans.CustomerID%type;

v\_enddate loans.EndDate%type;

v\_name Customers.Name%type;

begin

for loan\_rec in due\_loans\_cursor loop

v\_loadId := loan\_rec.LoanID;

v\_cusId := loan\_rec.CustomerID;

v\_enddate := loan\_rec.EndDate;

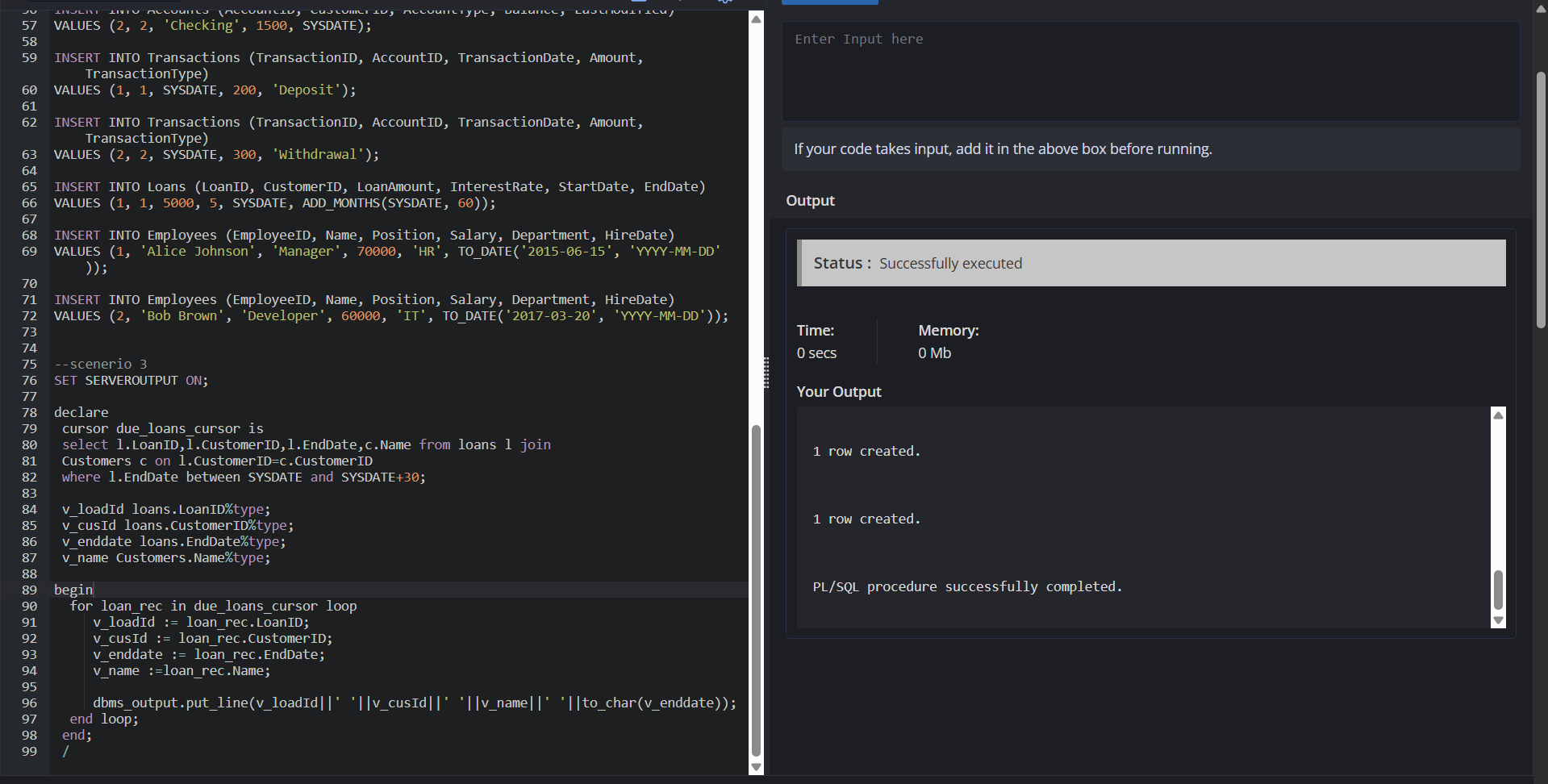
v\_name :=loan\_rec.Name;

dbms\_output.put\_line(v\_loadId||' '||v\_cusId||' '||v\_name||' '||to\_char(v\_enddate));

end loop;

end;

/



**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.

**ANSWER**

create or replace procedure ProcessMonthlyInterest is

Begin

update Accounts

set Balance = Balance +(Balance\*0.01)

where AccountType='Savings';

End;

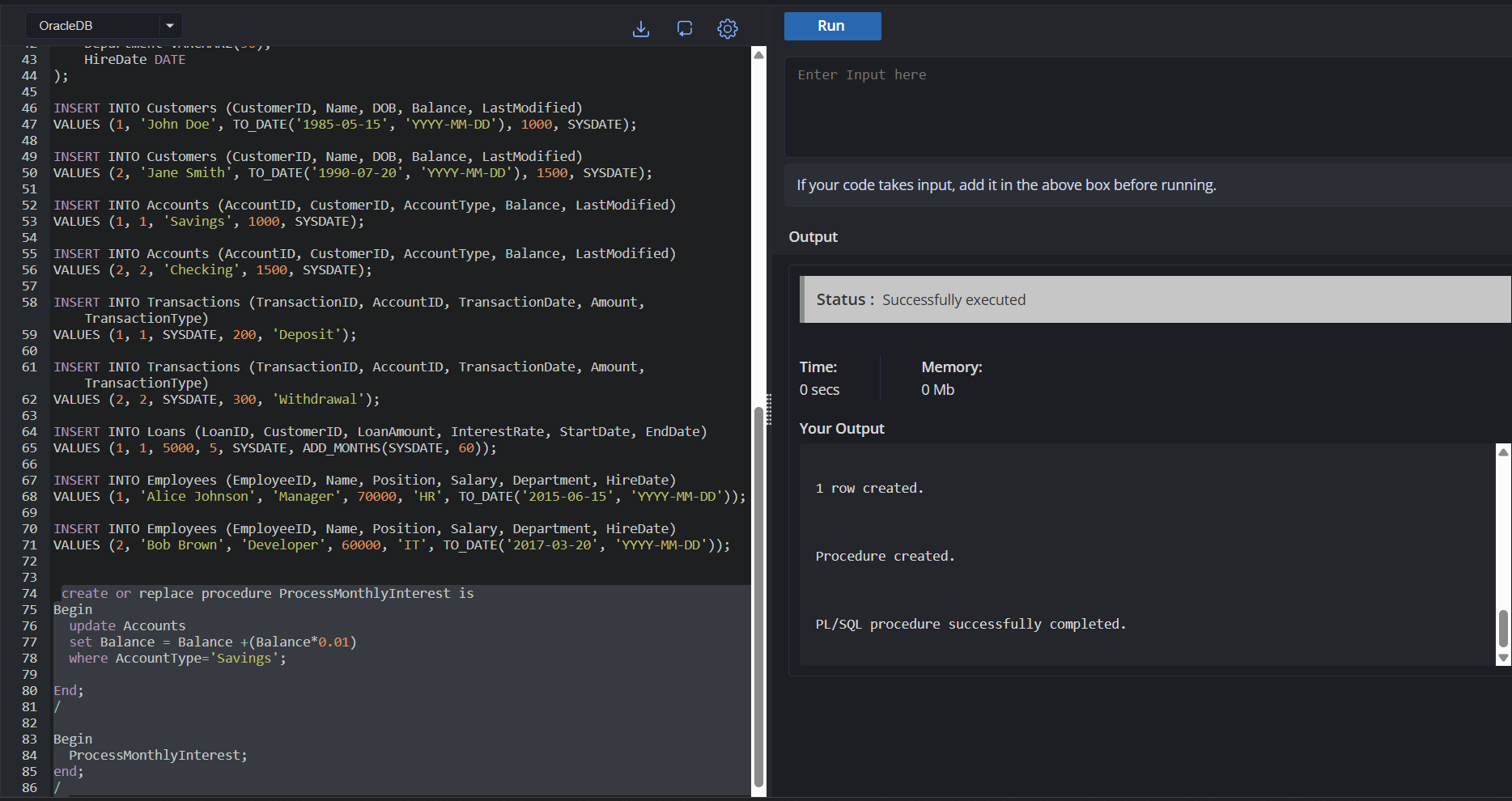
/

Begin

ProcessMonthlyInterest;

end;

/



**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.

**ANSWER**

create or replace procedure UpdateEmployeeBonus(

p\_dept in Employees.Department%type,

bonus in NUMBER

) is

Begin

update Employees

set salary = salary +(salary\*(bonus/100))

where Department=p\_dept;

End;

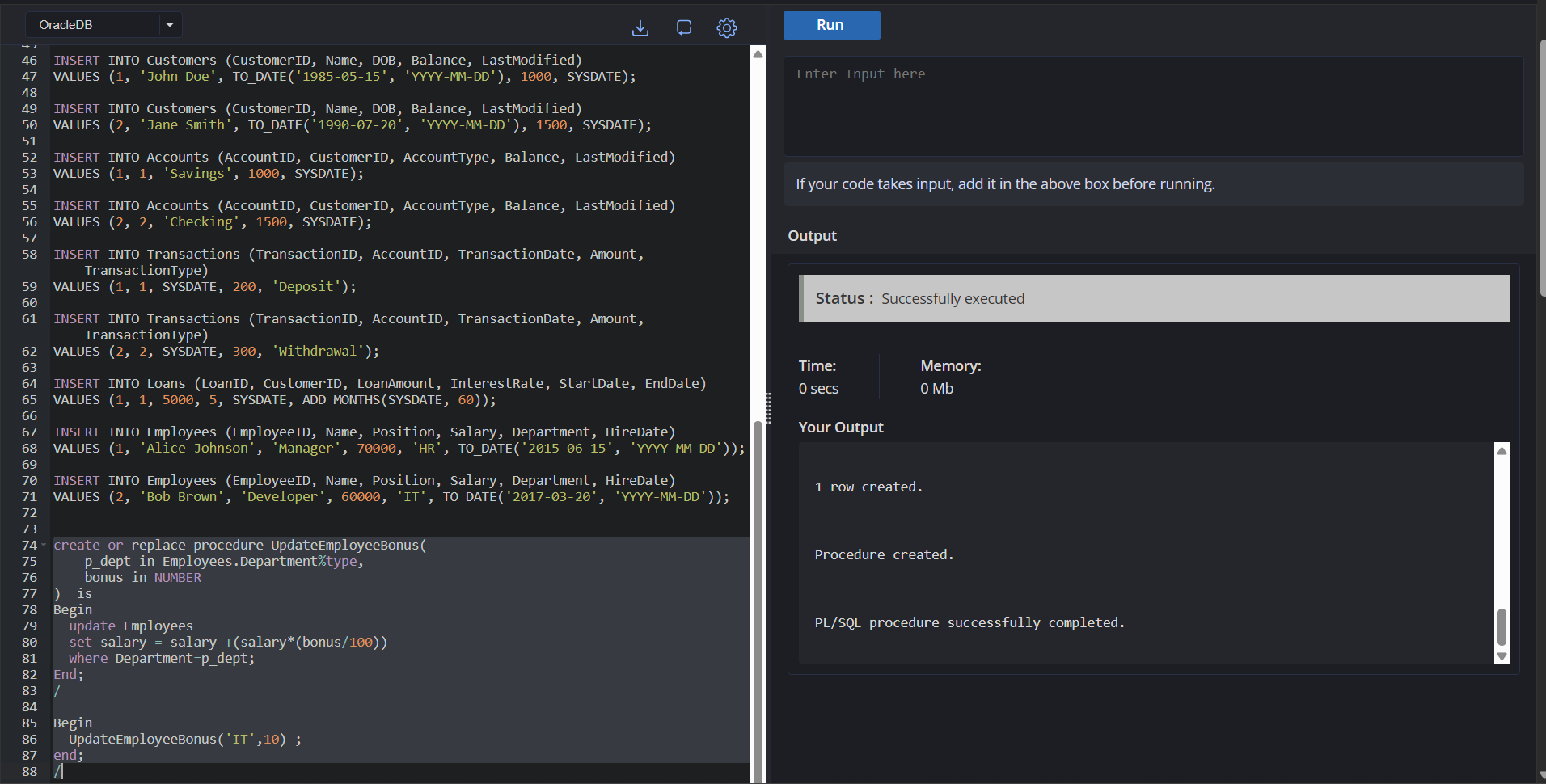
/

Begin

UpdateEmployeeBonus('IT',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.

**ANSWERS**

create or replace procedure TransferFunds(

sourceId in Customers.CustomerID%type,

destiId in Customers.CustomerID%type,

fundAmount NUMBER;

) is

sourceBalance Accounts%Balance.type;

Begin

select Balance into sourceBalance from Accounts where AccountID=sourceId for update;

if(sourceBalance>fundAmount) then

update Accounts

set Balance=Balance+fundAmount where AccountID=destiId;

update Accounts

set Balance=Balance-fundAmount where AccountID=sourceId;

else

dbms\_output.put\_line('Insufficient money in source account');

End if;

End;

/

Begin

TransferFunds(1,2,100) ;

end;

/

