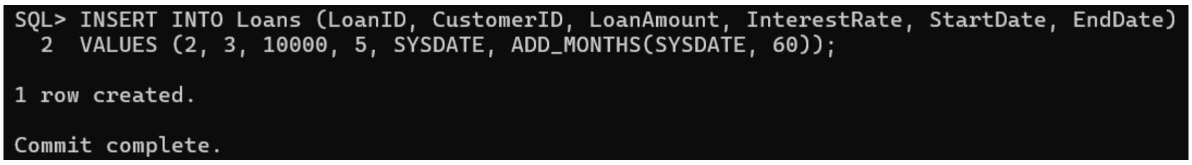
PL/SQL

Exercise 1: Control Structures

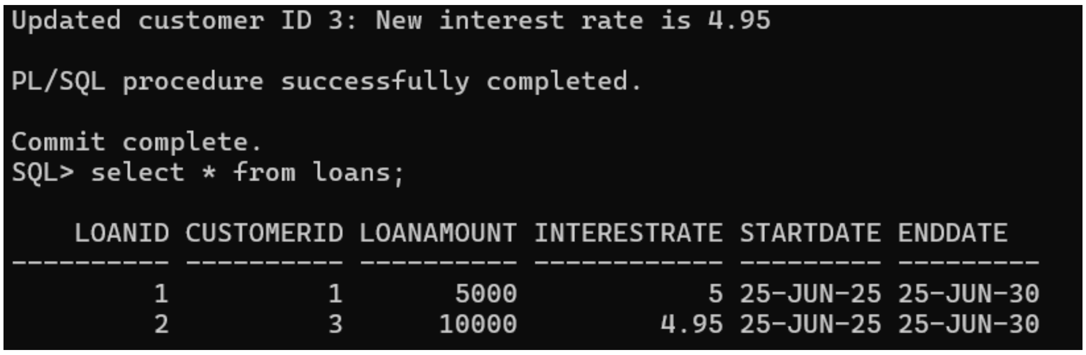
Q1> Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

➔Code

```
DECLARE
      c_id customers.customerId%type;
      c_dob customers.dob%type;
      c_age number;
      c_loanRate loans.interestRate%type;
      c_newLoanRate loans.interestRate%type;
BEGIN
      for i in (select c.customerId, c.dob from customers c) LOOP
            c_id := i.customerId;
            c_dob := i.dob;
            c_age := TRUNC(MONTHS_BETWEEN(SYSDATE, c_dob)/12);
            if(c_age > 60) then
                  select l.interestRate into c_loanRate
                  from loans l
                  where l.customerId = c_id;

                  c_newLoanRate := c_loanRate – (c_loanRate * 0.01);

                  update loans
                  set interestRate = c_newLoanRate
                  where customerId = c_id;

                  dbms_output.put_line('Updated customer ID ' || c_id || ': New interest rate is ' ||
                  c_newLoanRate);
            end if;
      end loop;
   END;
   /
```



```
SQL> INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)
  2  VALUES (2, 3, 10000, 5, SYSDATE, ADD_MONTHS(SYSDATE, 60));

1 row created.

Commit complete.
```

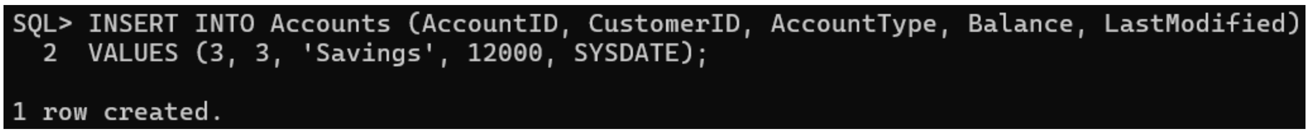


```
Updated customer ID 3: New interest rate is 4.95

PL/SQL procedure successfully completed.

Commit complete.
SQL> select * from loans;

    LOANID CUSTOMERID LOANAMOUNT INTERESTRATE STARTDATE ENDDATE
---------- ---------- ---------- ------------ --------- ---------
         1          1       5000            5 25-JUN-25 25-JUN-30
         2          3      10000         4.95 25-JUN-25 25-JUN-30
```

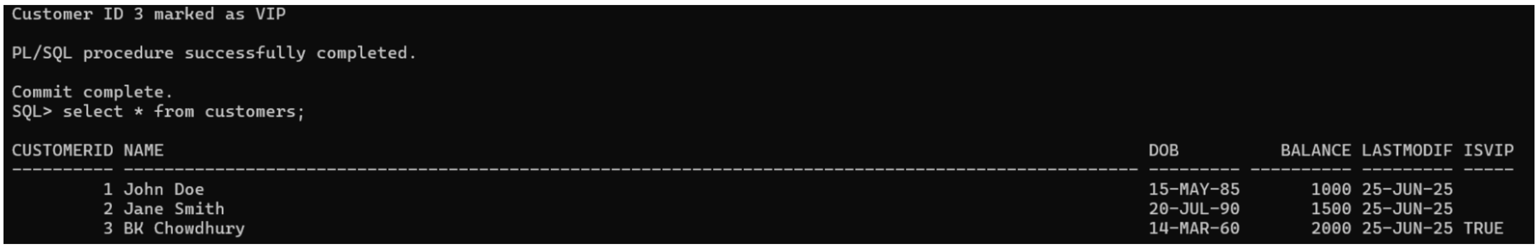
Q2> Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

➔code

```
DECLARE
   c_id customers.customerId%type;
   c_balance accounts.balance%type;
BEGIN
   for i IN (select a.customerId, a.balance from accounts a) LOOP
      c_id := i.customerId;
      c_balance := i.balance;

      if (c_balance > 10000) then
         update customers
         set IsVIP = 'TRUE'
         where customerId = c_id;

         dbms_output.put_line('Customer ID ' || c_id || ' marked as VIP');
      end if;
   end loop;
END;
/
```



```
SQL> INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)
  2  VALUES (3, 3, 'Savings', 12000, SYSDATE);

1 row created.
```

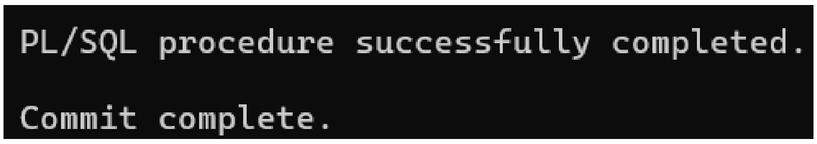


```
Customer ID 3 marked as VIP

PL/SQL procedure successfully completed.

Commit complete.
SQL> select * from customers;

CUSTOMERID NAME                                                                                                 DOB          BALANCE LASTMODIF ISVIP
---------- ---------------------------------------------------------------------------------------------------- --------- ---------- --------- -----
         1 John Doe                                                                                             15-MAY-85       1000 25-JUN-25
         2 Jane Smith                                                                                           20-JUL-90       1500 25-JUN-25
         3 BK Chowdhury                                                                                         14-MAR-60       2000 25-JUN-25 TRUE
```

Q3>Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

➔code

```
DECLARE
   c_id loans.customerId%type;
   c_loanAmt loans.loanAmount%type;
   c_dueDate loans.endDate%type;
BEGIN
     for i in (select customerId, loanAmount, endDate from loans where endDate
BETWEEN SYSDATE AND SYSDATE+30) LOOP
      c_id := i.customerId;
      c_loanAmt := i.loanAmount;
      c_dueDate := i.endDate;

      dbms_output.put_line('Reminder: Customer ID ' || c_id || ', your loan of $' || ' is due
on ' || to_Char(c_dueDate, 'DD/MM/YYYY'));
   end loop;
END;
/
```



```
PL/SQL procedure successfully completed.

Commit complete.
```

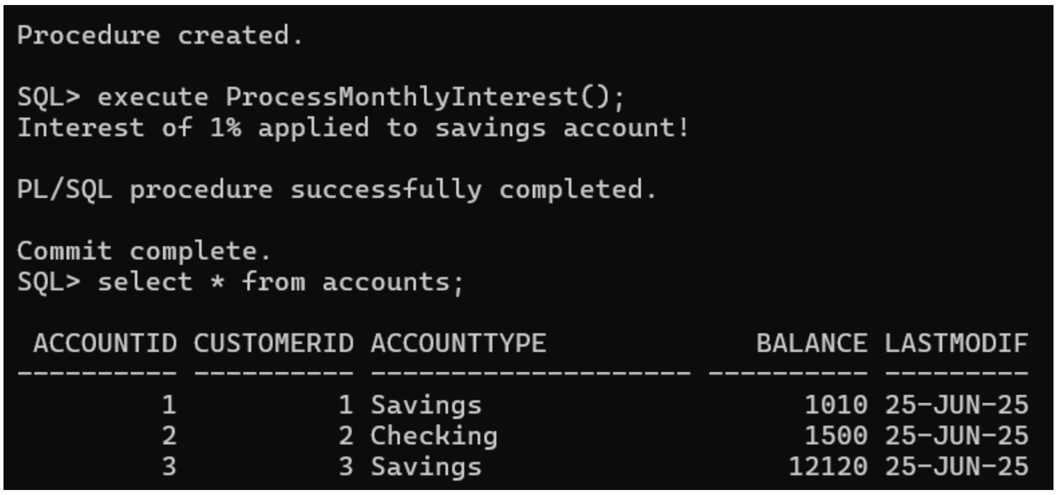
(no entry found with dues)

Exercise 3: Stored Procedures

Q1>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.

➔**code**

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
BEGIN
  UPDATE Accounts
  SET Balance = Balance + (Balance*0.01)
  WHERE AccountType = 'Savings';

  COMMIT;

  dbms_output.put_line('Interest of 1% applied to savings account!');
EXCEPTION
  when others then
    rollback;
    dbms_output.put_line('error! '|| SQLERRM);
END ProcessMonthlyInterest;
/
```

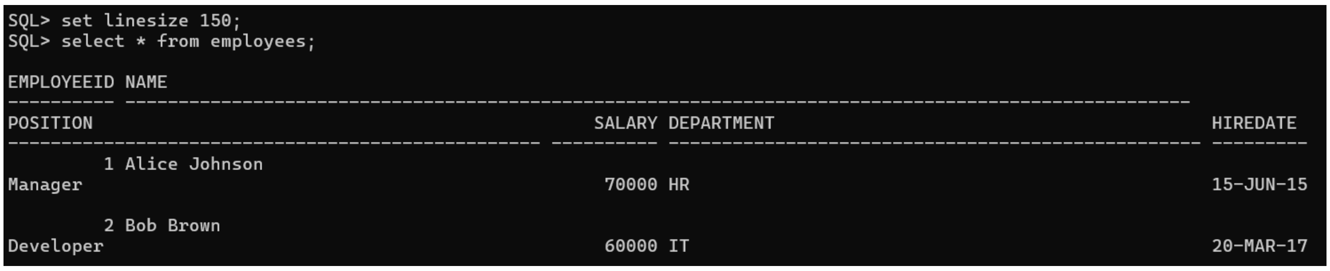


```
Procedure created.

SQL> execute ProcessMonthlyInterest();
Interest of 1% applied to savings account!

PL/SQL procedure successfully completed.

Commit complete.
SQL> select * from accounts;

 ACCOUNTID CUSTOMERID ACCOUNTTYPE             BALANCE LASTMODIF
---------- ---------- -------------------- ---------- ---------
         1          1 Savings                    1010 25-JUN-25
         2          2 Checking                   1500 25-JUN-25
         3          3 Savings                   12120 25-JUN-25
```

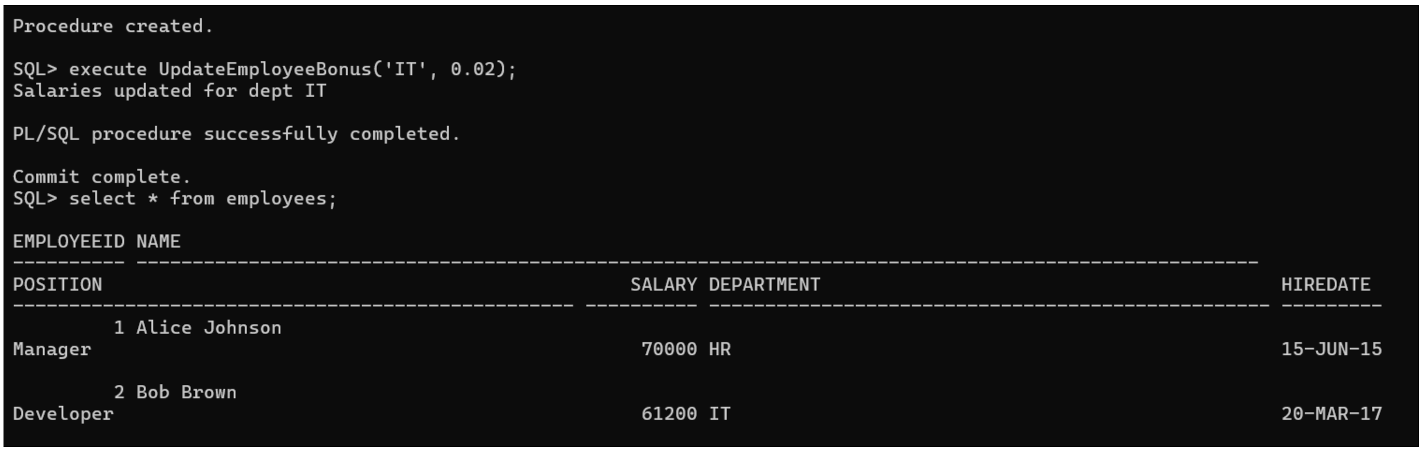
Q2>Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

➔**code**

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(
  dept IN varchar2,
  bonusPer IN number) AS
BEGIN
  UPDATE Employees
  SET salary = salary + (salary*bonusPer)
  WHERE department = dept;

  COMMIT;

  dbms_output.put_line('Salaries updated for dept ' || dept);
EXCEPTION
  WHEN others then
    rollback;
    dbms_output.put_line('error! ' || SQLERRM);
END UpdateEmployeeBonus;
/
```



```
SQL> set linesize 150;
SQL> select * from employees;

EMPLOYEEID NAME
---------- ----------------------------------------------------------------------------------------------------
POSITION                                                SALARY DEPARTMENT                                                    HIREDATE
-------------------------------------------------- ---------- -------------------------------------------------- ---------
         1 Alice Johnson
Manager                                                  70000 HR                                                        15-JUN-15

         2 Bob Brown
Developer                                                60000 IT                                                        20-MAR-17
```

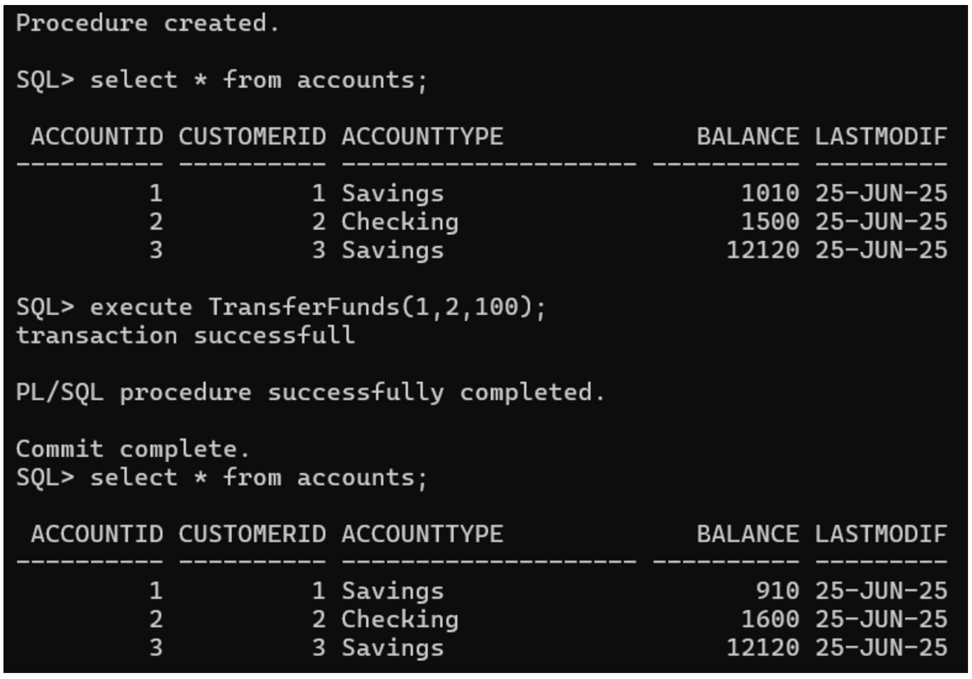


```
Procedure created.

SQL> execute UpdateEmployeeBonus('IT', 0.02);
Salaries updated for dept IT

PL/SQL procedure successfully completed.

Commit complete.
SQL> select * from employees;

EMPLOYEEID NAME
---------- ----------------------------------------------------------------------------------------------------
POSITION                                                SALARY DEPARTMENT                                                    HIREDATE
-------------------------------------------------- ---------- -------------------------------------------------- ---------
         1 Alice Johnson
Manager                                                  70000 HR                                                        15-JUN-15

         2 Bob Brown
Developer                                                61200 IT                                                        20-MAR-17
```

Q3> **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.**

➔**code**

```
CREATE OR REPLACE PROCEDURE TransferFunds(
  src_id IN number,
  target_id IN number,
  amount IN number) AS

  src_balance accounts.balance%type;
BEGIN
  select balance into src_balance from accounts
  where accountId = src_id;

  if src_balance < amount then
    dbms_output.put_line('insufficient balance!');
    return;
  end if;

  update accounts
  set balance = balance - amount
  where accountId = src_id;

  update accounts
  set balance = balance + amount
  where accountId = target_id;

  COMMIT;
  dbms_output.put_line('transaction successfull');
EXCEPTION
  when others then
    rollback;
    dbms_output.put_line('Error! ' || SQLERRM);
END TransferFunds;
/
```



```
Procedure created.

SQL> select * from accounts;

 ACCOUNTID CUSTOMERID ACCOUNTTYPE             BALANCE LASTMODIF
---------- ---------- -------------------- ---------- ---------
         1          1 Savings                    1010 25-JUN-25
         2          2 Checking                   1500 25-JUN-25
         3          3 Savings                   12120 25-JUN-25

SQL> execute TransferFunds(1,2,100);
transaction successfull

PL/SQL procedure successfully completed.

Commit complete.
SQL> select * from accounts;

 ACCOUNTID CUSTOMERID ACCOUNTTYPE             BALANCE LASTMODIF
---------- ---------- -------------------- ---------- ---------
         1          1 Savings                     910 25-JUN-25
         2          2 Checking                   1600 25-JUN-25
         3          3 Savings                   12120 25-JUN-25
```