Schema:

```
CREATE TABLE Customers (
        CustomerID NUMBER PRIMARY KEY,
        Name VARCHAR2(100),
        DOB DATE,
        Balance NUMBER,
        LastModified DATE
);


CREATE TABLE Accounts (
        AccountID NUMBER PRIMARY KEY,
        CustomerID NUMBER,
        AccountType VARCHAR2(20),
        Balance NUMBER,
        LastModified DATE,
        FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);


CREATE TABLE Transactions (
        TransactionID NUMBER PRIMARY KEY,
        AccountID NUMBER,
        TransactionDate DATE,
        Amount NUMBER,
        TransactionType VARCHAR2(10),
        FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)
);

CREATE TABLE Loans (
        LoanID NUMBER PRIMARY KEY,
        CustomerID NUMBER,
        LoanAmount NUMBER,
        InterestRate NUMBER,
        StartDate DATE,
        EndDate DATE,
        FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE Employees (
        EmployeeID NUMBER PRIMARY KEY,
        Name VARCHAR2(100),
        Position VARCHAR2(50),
        Salary NUMBER,
        Department VARCHAR2(50),
        HireDate DATE
);
```

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.

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.

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.

Scenario 1

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
      cursor c_senior_customers is select c.CustomerID from Customers c
where MONTHS_BETWEEN(SYSDATE, c.DOB)/ 12 > 60;

begin
      for customer_record in c_senior_customers loop
            update Loans
            set InterestRate = InterestRate - 1
            where customer_record.CustomerID = Loans.CustomerID;
            dbms_output.put_line('edited for ' ||
customer_record.CustomerID);
      end loop;
end;
/
```

Output Screenshots

Customers Table

```
SQL> select * from Customers;

CUSTOMERID NAME
---------- --------------------------------------------------------------------------------------------------------
DOB            BALANCE LASTMODIFIE
---------- ---------- ----------
         1 John Doe
15-MAY-1985       1000 24-JUN-2025

         2 Jane Smith
20-JUL-1990       1500 24-JUN-2025

         3 Ravi Shankar
10-FEB-1959       2000 24-JUN-2025

         4 Meena Das
05-SEP-1952       1800 24-JUN-2025

         5 Karan Malhotra
22-APR-2000       1200 24-JUN-2025

         6 Lalitha Nair
01-JUN-1963       3000 24-JUN-2025

         7 Anil Kapoor
17-NOV-1980       1700 24-JUN-2025

         8 Shobha Iyer
30-JAN-1956       2400 24-JUN-2025


8 rows selected.
```

Customers Older than 60 years

```
SQL> select * from Customers where MONTHS_BETWEEN(SYSDATE, DOB)/12 > 60;

CUSTOMERID NAME
---------- --------------------------------------------------------------------------------------------------------
DOB            BALANCE LASTMODIFIE
---------- ---------- ----------
         3 Ravi Shankar
10-FEB-1959       2000 24-JUN-2025

         4 Meena Das
05-SEP-1952       1800 24-JUN-2025

         6 Lalitha Nair
01-JUN-1963       3000 24-JUN-2025

         8 Shobha Iyer
30-JAN-1956       2400 24-JUN-2025


4 rows selected.
```

Loans Table

```
SQL> select * from Loans;

    LOANID CUSTOMERID LOANAMOUNT INTERESTRATE STARTDATE   ENDDATE
---------- ---------- ---------- ------------ ----------- -----------
         1          1       5000            5 24-JUN-2025 24-JUN-2030
         2          3      10000          6.5 24-JUN-2025 24-JUN-2030
         3          4       8000          7.2 24-JUN-2025 24-JUN-2029
         4          6       6000          5.8 24-JUN-2025 24-JUN-2028
         5          8       9000          6.9 24-JUN-2025 24-JUN-2030

5 rows selected.
```

Running the Program

```
SQL> set serveroutput on;
SQL> declare
  2     cursor c_senior_customers is select c.CustomerID from Customers c where MONTHS_BETWEEN(SYSDATE, c.DOB)/ 12 > 60;
  3
  4  begin
  5     for customer_record in c_senior_customers loop
  6             update Loans
  7             set InterestRate = InterestRate - 1
  8             where customer_record.CustomerID = Loans.CustomerID;
  9             dbms_output.put_line('edited for ' || customer_record.CustomerID);
 10     end loop;
 11  end;
 12  /
edited for 3
edited for 4
edited for 6
edited for 8

PL/SQL procedure successfully completed.
```

Updated Interest Rates for Customers with ID 3,4,6,8

```
SQL> select * from Loans;

    LOANID CUSTOMERID LOANAMOUNT INTERESTRATE STARTDATE   ENDDATE
---------- ---------- ---------- ------------ ----------- -----------
         1          1       5000            5 24-JUN-2025 24-JUN-2030
         2          3      10000          5.5 24-JUN-2025 24-JUN-2030
         3          4       8000          6.2 24-JUN-2025 24-JUN-2029
         4          6       6000          4.8 24-JUN-2025 24-JUN-2028
         5          8       9000          5.9 24-JUN-2025 24-JUN-2030

5 rows selected.
```

Scenario 2

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
	cursor c_customersWithHighBalance is select * from Customers where
Balance > 10000;


begin
	for currCustomer in c_customersWithHighBalance loop
		update Customers set isVIP = 'Y';
		dbms_output.put_line(currCustomer.Name || 'has been promoted
to VIP');
	end loop;
end;
/
```

Output Screenshots

Customers Table

```
SQL> update Customers set Balance = Balance + 10000;

8 rows updated.

SQL> select * from Customers;

CUSTOMERID NAME
---------- ----------------------------------------------------------------------------------------------------
DOB            BALANCE LASTMODIFIE
---------- ---------- ----------
         1 John Doe
15-MAY-1985      11000 24-JUN-2025

         2 Jane Smith
20-JUL-1990      11500 24-JUN-2025

         3 Ravi Shankar
10-FEB-1959      12000 24-JUN-2025

         4 Meena Das
05-SEP-1952      11800 24-JUN-2025

         5 Karan Malhotra
22-APR-2000      11200 24-JUN-2025

         6 Lalitha Nair
01-JUN-1963      13000 24-JUN-2025

         7 Anil Kapoor
17-NOV-1980      11700 24-JUN-2025

         8 Shobha Iyer
30-JAN-1956      12400 24-JUN-2025


8 rows selected.
```

Add isVIP column to Customers

```
SQL> alter table Customers add isVIP Char default 'N' Check (isVIP in ('Y', 'N'));

Table altered.

SQL> select * from Customers;

CUSTOMERID NAME
---------- ----------------------------------------------------------------------------------------------------
DOB           BALANCE LASTMODIFIE I
----------- ---------- ----------- -
         1 John Doe
15-MAY-1985      11000 24-JUN-2025 N

         2 Jane Smith
20-JUL-1990      11500 24-JUN-2025 N

         3 Ravi Shankar
10-FEB-1959      12000 24-JUN-2025 N

         4 Meena Das
05-SEP-1952      11800 24-JUN-2025 N

         5 Karan Malhotra
22-APR-2000      11200 24-JUN-2025 N

         6 Lalitha Nair
01-JUN-1963      13000 24-JUN-2025 N

         7 Anil Kapoor
17-NOV-1980      11700 24-JUN-2025 N

         8 Shobha Iyer
30-JAN-1956      12400 24-JUN-2025 N


8 rows selected.
```

Running the program

```
SQL> @D:\oracle\installclient\porgrams\sc2.sql
John Doehas been promoted to VIP
Jane Smithhas been promoted to VIP
Ravi Shankarhas been promoted to VIP
Meena Dashas been promoted to VIP
Karan Malhotrahas been promoted to VIP
Lalitha Nairhas been promoted to VIP
Anil Kapoorhas been promoted to VIP
Shobha Iyerhas been promoted to VIP

PL/SQL procedure successfully completed.
```

Updated Customers Table

```
SQL> select * from Customers;

CUSTOMERID NAME
---------- ----------------------------------------------------------------------------------------------------
DOB            BALANCE LASTMODIFIE I
---------- ---------- ---------- -
         1 John Doe
15-MAY-1985      11000 24-JUN-2025 Y

         2 Jane Smith
20-JUL-1990      11500 24-JUN-2025 Y

         3 Ravi Shankar
10-FEB-1959      12000 24-JUN-2025 Y

         4 Meena Das
05-SEP-1952      11800 24-JUN-2025 Y

         5 Karan Malhotra
22-APR-2000      11200 24-JUN-2025 Y

         6 Lalitha Nair
01-JUN-1963      13000 24-JUN-2025 Y

         7 Anil Kapoor
17-NOV-1980      11700 24-JUN-2025 Y

         8 Shobha Iyer
30-JAN-1956      12400 24-JUN-2025 Y


8 rows selected.
```

Scenario 3

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
      cursor c_custLoansDuein30Days is select * from Customers c left
join Loans l on c.CustomerID = l.CustomerID where l.enddate between
sysdate and sysdate + 30;

begin
      for cust_rec in c_custLoansDuein30Days loop
            dbms_output.put_line('Hey ' || cust_rec.Name ||'! Your loan
is due in ' || round(cust_rec.enddate - sysdate) || ' days');
            dbms_output.put_line('Your last date is ' ||
cust_rec.enddate);
            dbms_output.put_line('');
      end loop;
end;
/
```

Output Screenshots

A view of the Loans Table

```
SQL> select * from Loans;

    LOANID CUSTOMERID LOANAMOUNT INTERESTRATE STARTDATE   ENDDATE
---------- ---------- ---------- ------------ ----------- -----------
         1          1       5000            5 24-JUN-2025 24-JUN-2030
         2          3      10000          5.5 24-JUN-2025 24-JUN-2030
         3          4       8000          6.2 24-JUN-2025 24-JUN-2029
         4          6       6000          4.8 24-JUN-2025 24-JUN-2028
         5          8       9000          5.9 24-JUN-2025 24-JUN-2030
         6          2       4000          6.5 24-JUN-2025 01-JUL-2025
         7          4       3500            6 24-JUN-2025 09-JUL-2025
         8          6       5000          5.2 24-JUN-2025 23-JUL-2025

8 rows selected.

SQL> select * from Loans order by Enddate;

    LOANID CUSTOMERID LOANAMOUNT INTERESTRATE STARTDATE   ENDDATE
---------- ---------- ---------- ------------ ----------- -----------
         6          2       4000          6.5 24-JUN-2025 01-JUL-2025
         7          4       3500            6 24-JUN-2025 09-JUL-2025
         8          6       5000          5.2 24-JUN-2025 23-JUL-2025
         4          6       6000          4.8 24-JUN-2025 24-JUN-2028
         3          4       8000          6.2 24-JUN-2025 24-JUN-2029
         1          1       5000            5 24-JUN-2025 24-JUN-2030
         5          8       9000          5.9 24-JUN-2025 24-JUN-2030
         2          3      10000          5.5 24-JUN-2025 24-JUN-2030

8 rows selected.
```

Checking who has loans due within next 30 days

```
SQL> select l.loanid, l.enddate, c.Name from Loans l, Customers c where c.customerid = l.customerid and l.Enddate between sysdate and SYSDATE + 30;

    LOANID ENDDATE
---------- -----------
NAME
--------------------------------------------------------------------------------------------------
         6 01-JUL-2025
Jane Smith

         7 09-JUL-2025
Meena Das

         8 23-JUL-2025
Lalitha Nair


3 rows selected.
```

Running the Program

```
SQL> @D:\oracle\installclient\porgrams\sc3.sql
Hey Jane Smith! Your loan is due in 7 days
Your last date is 01-JUL-2025
Hey Meena Das! Your loan is due in 15 days
Your last date is 09-JUL-2025
Hey Lalitha Nair! Your loan is due in 29 days
Your last date is 23-JUL-2025

PL/SQL procedure successfully completed.
```

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.

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.

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.

Scenario 1

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 is
begin
      for curr in (select * from accounts where lower(AccountType) =
'savings') loop
      update Accounts set balance = curr.balance + (0.1 * curr.balance),
LastModified = sysdate where accountid = curr.accountid;

       dbms_output.put_line('updated for ' || curr.customerID);

      end loop;
end;
/
```

Output Screenshots

Accounts Table

```
SQL> SELECT * FROM Accounts;

 ACCOUNTID CUSTOMERID ACCOUNTTYPE            BALANCE LASTMODIFIE
---------- ---------- -------------------- ---------- -----------
         1          1 Savings                    1000 24-JUN-2025
         2          2 Checking                   1500 24-JUN-2025
         3          3 Savings                    2000 24-JUN-2025
         4          4 Savings                    3500 24-JUN-2025
         5          6 Savings                    5000 24-JUN-2025

5 rows selected.
```

Running the program

```
SQL> @D:\oracle\installclient\porgrams\sc4.sql

Procedure created.

SQL> exec ProcessMonthlyInterest
updated for 1
updated for 3
updated for 4
updated for 6

PL/SQL procedure successfully completed.
```

Updated Accounts Table

```
SQL>  SELECT * FROM Accounts;

 ACCOUNTID CUSTOMERID ACCOUNTTYPE            BALANCE LASTMODIFIE
---------- ---------- -------------------- ---------- -----------
         1          1 Savings                    1100 24-JUN-2025
         2          2 Checking                   1500 24-JUN-2025
         3          3 Savings                    2200 24-JUN-2025
         4          4 Savings                    3850 24-JUN-2025
         5          6 Savings                    5500 24-JUN-2025

5 rows selected.
```

Scenario 2

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(bonusPercentage in
number, dept in varchar2) is
begin
      for currEmp in (select * from Employees where Department = dept)
loop
            update Employees
            set Salary = currEmp.Salary +
                        (bonusPercentage * currEmp.Salary) / 100
            where employeeid = currEmp.EmployeeID;

            dbms_output.put_line('bonus updated for employee with ID: '
|| currEmp.employeeid);

      end loop;
end;
/
```

Output Screenshots

Employees Table

```
SQL> select * from Employees;

EMPLOYEEID NAME
---------- ----------------------------------------------------------------------------------------------------
POSITION                                           SALARY DEPARTMENT
-------------------------------------------------- ---------- --------------------------------------------------
HIREDATE
-----------
         1 Alice Johnson
Manager                                             70000 HR
15-JUN-2015

         2 Bob Brown
Developer                                           60000 IT
20-MAR-2017

         3 Catherine Lee
Analyst                                             55000 Finance
05-NOV-2018

         4 David Kumar
Support Engineer                                    48000 IT
12-FEB-2020

         5 Elena Fernandez
Recruiter                                           50000 HR
25-AUG-2019

         6 Faisal Khan
Team Lead                                           75000 Operations
17-APR-2016

         7 Grace Thomas
Data Scientist                                      82000 Analytics
01-OCT-2021

         8 Harsh Mehta
Intern                                              25000 Finance
10-JUN-2024

8 rows selected.
```

Running the program

```
SQL> @D:\oracle\installclient\porgrams\sc5.sql

Procedure created.

SQL> exec UpdateEmployeeBonus(10, 'Finance');
bonus updated for employee with ID: 3
bonus updated for employee with ID: 8

PL/SQL procedure successfully completed.
```

Updated Employees table for ID 3,8

```
SQL> select * from Employees;

EMPLOYEEID NAME
---------- --------------------------------------------------------------------------------------------------
POSITION                                           SALARY DEPARTMENT
-------------------------------------------------- ---------- --------------------------------------------------
HIREDATE
-----------
         1 Alice Johnson
Manager                                             70000 HR
15-JUN-2015

         2 Bob Brown
Developer                                           60000 IT
20-MAR-2017

         3 Catherine Lee
Analyst                                             60500 Finance
05-NOV-2018

         4 David Kumar
Support Engineer                                    48000 IT
12-FEB-2020

         5 Elena Fernandez
Recruiter                                           50000 HR
25-AUG-2019

         6 Faisal Khan
Team Lead                                           75000 Operations
17-APR-2016

         7 Grace Thomas
Data Scientist                                      82000 Analytics
01-OCT-2021

         8 Harsh Mehta
Intern                                              27500 Finance
10-JUN-2024


8 rows selected.
```

Scenario 3

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
(fromAcc in number, toAcc in number, amountToSend in number)
is
      curr_record Accounts%ROWTYPE;
begin
      select * into curr_record from Accounts where AccountID = fromAcc;

      if curr_record.Balance >= amountToSend then

            update Accounts set Balance = Balance + amountToSend,
LastModified = SYSDATE where AccountID = toAcc;

            update Accounts set Balance = Balance - amountToSend,
LastModified = SYSDATE where AccountID = fromAcc;

            insert into Transactions (TransactionID, AccountID,
TransactionDate, Amount, TransactionType) values
(Transactions_SEQ.NEXTVAL, fromAcc, SYSDATE, amountToSend, 'Transfer');

            dbms_output.put_line('successfully transferred! have fun
:)');

      else
            dbms_output.put_line('oops, insufficient funds :''( ');
      end if;
      commit;

exception
when no_data_found then
dbms_output.put_line('oops, account not found :( ');

when others then
rollback;
dbms_output.put_line('oops, error :( ');

end;
/
```

Output Screenshots

Created a sequence on Transaction ID

```
SQL> @D:\oracle\installclient\porgrams\sc6.sql

Warning: Procedure created with compilation errors.

SQL> SHOW ERRORS PROCEDURE TransferFunds;
Errors for PROCEDURE TRANSFERFUNDS:

LINE/COL ERROR
-------- -----------------------------------------------------------------
10/3     PL/SQL: SQL Statement ignored
10/35    PL/SQL: ORA-02289: sequence does not exist
```

```
SQL> CREATE SEQUENCE Transactions_SEQ
  2    START WITH 1
  3    INCREMENT BY 1
  4    NOCACHE
  5    NOCYCLE;

Sequence created.
```

Accounts Table

```
SQL> select * from Accounts;

 ACCOUNTID CUSTOMERID ACCOUNTTYPE            BALANCE LASTMODIFIE
---------- ---------- -------------------- ---------- -----------
         1          1 Savings                    1100 24-JUN-2025
         2          2 Checking                   1500 24-JUN-2025
         3          3 Savings                    2200 24-JUN-2025
         4          4 Savings                    3850 24-JUN-2025
         5          6 Savings                    5500 24-JUN-2025

5 rows selected.
```

Running the program

```
SQL> @D:\oracle\installclient\porgrams\sc6.sql

Procedure created.
```

Case when transaction was successful:

```
SQL> exec TransferFunds(5, 1, 500);
successfully transferred! have fun :)

PL/SQL procedure successfully completed.
```

Accounts Table with updated value for Accounts 1 and 5

```
SQL> select * from Accounts;

 ACCOUNTID CUSTOMERID ACCOUNTTYPE            BALANCE LASTMODIFIE
---------- ---------- -------------------- ---------- -----------
         1          1 Savings                    1600 25-JUN-2025
         2          2 Checking                   1500 24-JUN-2025
         3          3 Savings                    2200 24-JUN-2025
         4          4 Savings                    3850 24-JUN-2025
         5          6 Savings                    5000 25-JUN-2025

5 rows selected.
```

Updated Transactions Table

```
SQL> select * from Transactions;

TRANSACTIONID  ACCOUNTID TRANSACTION     AMOUNT TRANSACTIO
------------- ---------- ----------- ---------- ----------
            1          1 24-JUN-2025        200 Deposit
            2          2 24-JUN-2025        300 Withdrawal
            3          5 25-JUN-2025        500 Transfer

3 rows selected.
```

Case when transaction was unsuccessful:

```
SQL> exec TransferFunds(1, 5, 5000);
oops, insufficient funds :'(

PL/SQL procedure successfully completed.
```

Unchanged Transactions Table

```
SQL> select * from Transactions;

TRANSACTIONID  ACCOUNTID TRANSACTION     AMOUNT TRANSACTIO
------------- ---------- ----------- ---------- ----------
            1          1 24-JUN-2025        200 Deposit
            2          2 24-JUN-2025        300 Withdrawal
            3          5 25-JUN-2025        500 Transfer

3 rows selected.
```