**Exercise 1: Control Structures**

**Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Loans CASCADE CONSTRAINTS';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -942 THEN

RAISE;

END IF;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Customers CASCADE CONSTRAINTS';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -942 THEN

RAISE;

END IF;

END;

/

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE,

IsVIP CHAR(1) DEFAULT 'N'

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (1, 'John Doe', TO\_DATE('1950-05-15', 'YYYY-MM-DD'), 12000, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (2, 2, 7000, 6, SYSDATE, ADD\_MONTHS(SYSDATE, 48));

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (3, 1, 8000, 4.5, SYSDATE, SYSDATE + 10);

COMMIT;

BEGIN

FOR c IN (SELECT CustomerID, DOB FROM Customers) LOOP

IF MONTHS\_BETWEEN(SYSDATE, c.DOB)/12 > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = c.CustomerID;

END IF;

END LOOP;

END;

/

SET SERVEROUTPUT ON;

BEGIN

FOR c IN (SELECT CustomerID, Name, Balance FROM Customers) LOOP

IF c.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'Y',

LastModified = SYSDATE

WHERE CustomerID = c.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer ' || c.Name || ' marked as VIP.');

END IF;

END LOOP;

END;

/

BEGIN

FOR l IN (

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

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || l.LoanID ||

' for customer ' || l.Name ||

' is due on ' || TO\_CHAR(l.EndDate, 'YYYY-MM-DD'));

END LOOP;

END;

/

SELECT \* FROM Customers;

SELECT \* FROM Loans;

**Output:**

Customer John Doe marked as VIP.

Reminder: Loan ID 3 for customer John Doe is due on 2025-07-06

CUSTOMERID

----------

NAME

--------------------------------------------------------------------------------

DOB BALANCE LASTMODIF I

--------- ---------- --------- -

1

John Doe

15-MAY-50 12000 26-JUN-25 Y

2

Jane Smith

20-JUL-90 1500 26-JUN-25 N

CUSTOMERID

----------

NAME

--------------------------------------------------------------------------------

DOB BALANCE LASTMODIF I

--------- ---------- --------- -

LOANID CUSTOMERID LOANAMOUNT INTERESTRATE STARTDATE ENDDATE

---------- ---------- ---------- ------------ --------- ---------

1 1 5000 4 26-JUN-25 26-JUN-30

2 2 7000 6 26-JUN-25 26-JUN-29

3 1 8000 3.5 26-JUN-25 06-JUL-25

LOANID CUSTOMERID LOANAMOUNT INTERESTRATE STARTDATE ENDDATE

---------- ---------- ---------- ------------ --------- ---------

1 1 5000 4 26-JUN-25 26-JUN-30

2 2 7000 6 26-JUN-25 26-JUN-29

3 1 8000 3.5 26-JUN-25 06-JUL-25

**Exercise 2: Error Handling**

**Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Accounts CASCADE CONSTRAINTS';

EXCEPTION WHEN OTHERS THEN IF SQLCODE != -942 THEN RAISE; END IF;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Employees CASCADE CONSTRAINTS';

EXCEPTION WHEN OTHERS THEN IF SQLCODE != -942 THEN RAISE; END IF;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Customers CASCADE CONSTRAINTS';

EXCEPTION WHEN OTHERS THEN IF SQLCODE != -942 THEN RAISE; END IF;

END;

/

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE,

IsVIP CHAR(1) DEFAULT 'N'

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified 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

);

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 10000, SYSDATE, 'N');

INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 5000, SYSDATE, 'N');

INSERT INTO Accounts VALUES (1, 1, 'Savings', 5000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Savings', 2000, SYSDATE);

INSERT INTO Employees VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

COMMIT;

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

p\_fromAccountID IN NUMBER,

p\_toAccountID IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_fromAccountID FOR UPDATE;

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds.');

END IF;

UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_fromAccountID;

UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_toAccountID;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error during transfer: ' || SQLERRM);

END;

/

CREATE OR REPLACE PROCEDURE UpdateSalary (

p\_empID IN NUMBER,

p\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* p\_percent / 100)

WHERE EmployeeID = p\_empID;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Employee not found.');

END IF;

DBMS\_OUTPUT.PUT\_LINE('Salary updated.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error updating salary: ' || SQLERRM);

END;

/

CREATE OR REPLACE PROCEDURE AddNewCustomer (

p\_customerID IN NUMBER,

p\_name IN VARCHAR2,

p\_dob IN DATE,

p\_balance IN NUMBER

) IS

BEGIN

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (p\_customerID, p\_name, p\_dob, p\_balance, SYSDATE);

DBMS\_OUTPUT.PUT\_LINE('Customer added.');

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Customer ID already exists.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Unexpected error: ' || SQLERRM);

END;

/

SET SERVEROUTPUT ON;

BEGIN

SafeTransferFunds(1, 2, 1000);

END;

/

BEGIN

SafeTransferFunds(2, 1, 10000);

END;

/

BEGIN

UpdateSalary(1, 10);

END;

/

BEGIN

UpdateSalary(999, 10);

END;

/

BEGIN

AddNewCustomer(3, 'Robert Taylor', TO\_DATE('1988-08-10', 'YYYY-MM-DD'), 8000);

END;

/

BEGIN

AddNewCustomer(3, 'Duplicate', TO\_DATE('1980-01-01', 'YYYY-MM-DD'), 3000);

END;

/

SELECT \* FROM Customers;

SELECT \* FROM Accounts;

SELECT \* FROM Employees;

**Output:**

Transfer successful.

Error during transfer: ORA-20001: Insufficient funds.

Salary updated.

Error updating salary: ORA-20002: Employee not found.

Customer added.

Customer ID already exists.

CUSTOMERID

----------

NAME

--------------------------------------------------------------------------------

DOB BALANCE LASTMODIF I

--------- ---------- --------- -

1

John Doe

15-MAY-85 10000 26-JUN-25 N

2

Jane Smith

20-JUL-90 5000 26-JUN-25 N

CUSTOMERID

----------

NAME

--------------------------------------------------------------------------------

DOB BALANCE LASTMODIF I

--------- ---------- --------- -

3

Robert Taylor

10-AUG-88 8000 26-JUN-25 N

ACCOUNTID CUSTOMERID ACCOUNTTYPE BALANCE LASTMODIF

---------- ---------- -------------------- ---------- ---------

1 1 Savings 4000 26-JUN-25

2 2 Savings 3000 26-JUN-25

EMPLOYEEID

----------

NAME

--------------------------------------------------------------------------------

POSITION SALARY

-------------------------------------------------- ----------

DEPARTMENT HIREDATE

-------------------------------------------------- ---------

1

Alice Johnson

Manager 77000

HR 15-JUN-15

EMPLOYEEID

----------

NAME

--------------------------------------------------------------------------------

POSITION SALARY

-------------------------------------------------- ----------

DEPARTMENT HIREDATE

-------------------------------------------------- ---------

2

Bob Brown

Developer 60000

IT 20-MAR-17

EMPLOYEEID

----------

NAME

--------------------------------------------------------------------------------

POSITION SALARY

-------------------------------------------------- ----------

DEPARTMENT HIREDATE

-------------------------------------------------- ---------

1

Alice Johnson

Manager 77000

HR 15-JUN-15

EMPLOYEEID

----------

NAME

--------------------------------------------------------------------------------

POSITION SALARY

-------------------------------------------------- ----------

DEPARTMENT HIREDATE

-------------------------------------------------- ---------

2

Bob Brown

Developer 60000

IT 20-MAR-17

**Exercise 3: Stored Procedures**

**Code:**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

INSERT INTO Accounts VALUES (1, 1, 'Savings', 5000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Savings', 2000, SYSDATE);

INSERT INTO Employees VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP

UPDATE Accounts

SET Balance = Balance + (acc.Balance \* 0.01),

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied to Account ID: ' || acc.AccountID);

END LOOP;

COMMIT;

END;

/

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

FOR emp IN (

SELECT EmployeeID, Salary FROM Employees WHERE Department = p\_department

) LOOP

UPDATE Employees

SET Salary = Salary + (emp.Salary \* p\_bonus\_percent / 100)

WHERE EmployeeID = emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Bonus added to Employee ID: ' || emp.EmployeeID);

END LOOP;

COMMIT;

END;

/

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_fromAccountID IN NUMBER,

p\_toAccountID IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_fromAccountID FOR UPDATE;

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20010, 'Insufficient funds for transfer.');

END IF;

UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_fromAccountID;

UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_toAccountID;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer completed.');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error during transfer: ' || SQLERRM);

END;

/

SET SERVEROUTPUT ON;

BEGIN

ProcessMonthlyInterest;

END;

/

BEGIN

UpdateEmployeeBonus('HR', 10);

END;

/

BEGIN

TransferFunds(1, 2, 500);

END;

/

BEGIN

TransferFunds(2, 1, 999999);

END;

/

SELECT \* FROM Accounts;

SELECT \* FROM Employees;

**Output:**

Interest applied to Account ID: 1

Interest applied to Account ID: 2

Bonus added to Employee ID: 1

Transfer completed.

Error during transfer: ORA-20010: Insufficient funds for transfer.

ACCOUNTID CUSTOMERID ACCOUNTTYPE BALANCE LASTMODIF

---------- ---------- -------------------- ---------- ---------

1 1 Savings 4550 26-JUN-25

2 2 Savings 2520 26-JUN-25

EMPLOYEEID

----------

NAME

--------------------------------------------------------------------------------

POSITION SALARY

-------------------------------------------------- ----------

DEPARTMENT HIREDATE

-------------------------------------------------- ---------

1

Alice Johnson

Manager 77000

HR 15-JUN-15

EMPLOYEEID

----------

NAME

--------------------------------------------------------------------------------

POSITION SALARY

-------------------------------------------------- ----------

DEPARTMENT HIREDATE

-------------------------------------------------- ---------

2

Bob Brown

Developer 60000

IT 20-MAR-17

EMPLOYEEID

----------

NAME

--------------------------------------------------------------------------------

POSITION SALARY

-------------------------------------------------- ----------

DEPARTMENT HIREDATE

-------------------------------------------------- ---------

1

Alice Johnson

Manager 77000

HR 15-JUN-15

EMPLOYEEID

----------

NAME

--------------------------------------------------------------------------------

POSITION SALARY

-------------------------------------------------- ----------

DEPARTMENT HIREDATE

-------------------------------------------------- ---------

2

Bob Brown

Developer 60000

IT 20-MAR-17

**Exercise 4: Functions**

**Code:**

SET SERVEROUTPUT ON;

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Accounts CASCADE CONSTRAINTS';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Customers CASCADE CONSTRAINTS';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

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)

);

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1980-01-01', 'YYYY-MM-DD'), 5000, SYSDATE);

INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1995-06-15', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Accounts VALUES (1, 1, 'Savings', 6000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Savings', 1000, SYSDATE);

COMMIT;

CREATE OR REPLACE FUNCTION CalculateAge(p\_dob DATE)

RETURN NUMBER IS

v\_age NUMBER;

BEGIN

v\_age := TRUNC(MONTHS\_BETWEEN(SYSDATE, p\_dob) / 12);

RETURN v\_age;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Age of Jane Smith: ' || CalculateAge(TO\_DATE('1995-06-15', 'YYYY-MM-DD')));

END;

/

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment (

p\_loanAmount NUMBER,

p\_annualRate NUMBER,

p\_durationYears NUMBER

)

RETURN NUMBER IS

v\_monthlyRate NUMBER;

v\_months NUMBER;

v\_emi NUMBER;

BEGIN

v\_monthlyRate := p\_annualRate / (12 \* 100);

v\_months := p\_durationYears \* 12;

IF v\_monthlyRate = 0 THEN

v\_emi := p\_loanAmount / v\_months;

ELSE

v\_emi := p\_loanAmount \* v\_monthlyRate \* POWER(1 + v\_monthlyRate, v\_months) /

(POWER(1 + v\_monthlyRate, v\_months) - 1);

END IF;

RETURN ROUND(v\_emi, 2);

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Monthly EMI for 100000 loan: ' || CalculateMonthlyInstallment(100000, 7.5, 5));

END;

/

CREATE OR REPLACE FUNCTION HasSufficientBalance (

p\_accountID IN NUMBER,

p\_amount IN NUMBER

)

RETURN BOOLEAN IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_accountID;

IF v\_balance >= p\_amount THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

END;

/

DECLARE

result BOOLEAN;

BEGIN

result := HasSufficientBalance(1, 2000);

IF result THEN

DBMS\_OUTPUT.PUT\_LINE('Account 1 has sufficient balance.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Account 1 does not have sufficient balance.');

END IF;

END;

/

**Output:**

Age of Jane Smith: 30

Monthly EMI for 100000 loan: 2003.79

Account 1 has sufficient balance.

**Exercise 5: Triggers**

**Code:**

SET SERVEROUTPUT ON;

BEGIN EXECUTE IMMEDIATE 'DROP TABLE AuditLog CASCADE CONSTRAINTS'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

BEGIN EXECUTE IMMEDIATE 'DROP TABLE Transactions CASCADE CONSTRAINTS'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

BEGIN EXECUTE IMMEDIATE 'DROP TABLE Accounts CASCADE CONSTRAINTS'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

BEGIN EXECUTE IMMEDIATE 'DROP TABLE Customers CASCADE CONSTRAINTS'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

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 AuditLog (

LogID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

TransactionID NUMBER,

AccountID NUMBER,

ActionDate DATE,

ActionType VARCHAR2(20)

);

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1985-01-01', 'YYYY-MM-DD'), 10000, SYSDATE);

INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1990-06-15', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Accounts VALUES (1, 1, 'Savings', 8000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Checking', 2000, SYSDATE);

COMMIT;

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

BEFORE UPDATE ON Customers

FOR EACH ROW

BEGIN

:NEW.LastModified := SYSDATE;

END;

/

CREATE OR REPLACE TRIGGER LogTransaction

AFTER INSERT ON Transactions

FOR EACH ROW

BEGIN

INSERT INTO AuditLog (TransactionID, AccountID, ActionDate, ActionType)

VALUES (:NEW.TransactionID, :NEW.AccountID, SYSDATE, 'INSERT');

END;

/

CREATE OR REPLACE TRIGGER CheckTransactionRules

BEFORE INSERT ON Transactions

FOR EACH ROW

DECLARE

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = :NEW.AccountID;

IF :NEW.TransactionType = 'Withdrawal' AND :NEW.Amount > v\_balance THEN

DBMS\_OUTPUT.PUT\_LINE('Warning: Withdrawal exceeds balance. Transaction skipped.');

:NEW.Amount := 0;

:NEW.TransactionType := 'Rejected';

ELSIF :NEW.TransactionType = 'Deposit' AND :NEW.Amount <= 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Warning: Invalid deposit amount. Transaction skipped.');

:NEW.Amount := 0;

:NEW.TransactionType := 'Rejected';

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Warning: Account not found.');

END;

/

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (1, 1, SYSDATE, 500, 'Deposit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (3, 2, SYSDATE, 999999, 'Withdrawal');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (4, 2, SYSDATE, -500, 'Deposit');

SELECT \* FROM Customers;

SELECT \* FROM Transactions;

SELECT \* FROM AuditLog;

**Output:**

Warning: Withdrawal exceeds balance. Transaction skipped.

Warning: Invalid deposit amount. Transaction skipped.

CUSTOMERID

----------

NAME

--------------------------------------------------------------------------------

DOB BALANCE LASTMODIF

--------- ---------- ---------

1

John Doe

01-JAN-85 10000 26-JUN-25

2

Jane Smith

15-JUN-90 1500 26-JUN-25

CUSTOMERID

----------

NAME

--------------------------------------------------------------------------------

DOB BALANCE LASTMODIF

--------- ---------- ---------

TRANSACTIONID ACCOUNTID TRANSACTI AMOUNT TRANSACTIO

------------- ---------- --------- ---------- ----------

1 1 26-JUN-25 500 Deposit

2 2 26-JUN-25 300 Withdrawal

3 2 26-JUN-25 0 Rejected

4 2 26-JUN-25 0 Rejected

LOGID TRANSACTIONID ACCOUNTID ACTIONDAT ACTIONTYPE

---------- ------------- ---------- --------- --------------------

1 1 1 26-JUN-25 INSERT

2 2 2 26-JUN-25 INSERT

3 3 2 26-JUN-25 INSERT

4 4 2 26-JUN-25 INSERT

LOGID TRANSACTIONID ACCOUNTID ACTIONDAT ACTIONTYPE

---------- ------------- ---------- --------- --------------------

1 1 1 26-JUN-25 INSERT

2 2 2 26-JUN-25 INSERT

3 3 2 26-JUN-25 INSERT

4 4 2 26-JUN-25 INSERT

**Exercise 6: Cursors**

**Code:**

BEGIN EXECUTE IMMEDIATE 'DROP TABLE Transactions CASCADE CONSTRAINTS'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

BEGIN EXECUTE IMMEDIATE 'DROP TABLE Accounts CASCADE CONSTRAINTS'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

BEGIN EXECUTE IMMEDIATE 'DROP TABLE Loans CASCADE CONSTRAINTS'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

BEGIN EXECUTE IMMEDIATE 'DROP TABLE Customers CASCADE CONSTRAINTS'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

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 Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate 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)

);

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1985-01-01', 'YYYY-MM-DD'), 10000, SYSDATE);

INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1990-06-15', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Accounts VALUES (1, 1, 'Savings', 8000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Checking', 2000, SYSDATE);

INSERT INTO Loans VALUES (1, 1, 5000, 5.0, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

INSERT INTO Loans VALUES (2, 2, 9000, 6.0, SYSDATE, ADD\_MONTHS(SYSDATE, 48));

INSERT INTO Transactions VALUES (1, 1, SYSDATE, 1000, 'Deposit');

INSERT INTO Transactions VALUES (2, 2, SYSDATE, 500, 'Withdrawal');

COMMIT;

SET SERVEROUTPUT ON;

BEGIN

FOR txn IN (

SELECT t.TransactionID, t.AccountID, t.Amount, t.TransactionType, t.TransactionDate, c.Name

FROM Transactions t

JOIN Accounts a ON t.AccountID = a.AccountID

JOIN Customers c ON a.CustomerID = c.CustomerID

WHERE TRUNC(t.TransactionDate, 'MM') = TRUNC(SYSDATE, 'MM')

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Statement: Customer ' || txn.Name ||

', AccountID ' || txn.AccountID ||

', Transaction: ' || txn.TransactionType ||

', Amount: ' || txn.Amount ||

', Date: ' || TO\_CHAR(txn.TransactionDate, 'DD-MON-YYYY'));

END LOOP;

END;

/

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM Accounts) LOOP

UPDATE Accounts

SET Balance = Balance - 200,

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Annual fee of ₹200 deducted from Account ID ' || acc.AccountID);

END LOOP;

END;

/

BEGIN

FOR l IN (SELECT LoanID, LoanAmount, InterestRate FROM Loans) LOOP

IF l.LoanAmount > 7000 THEN

UPDATE Loans

SET InterestRate = InterestRate - 0.5

WHERE LoanID = l.LoanID;

DBMS\_OUTPUT.PUT\_LINE('Interest rate reduced for Loan ID ' || l.LoanID);

END IF;

END LOOP;

END;

/

SELECT \* FROM Accounts;

SELECT \* FROM Loans;

**Output:**

Statement: Customer John Doe, AccountID 1, Transaction: Deposit, Amount: 1000,

Date: 26-JUN-2025

Statement: Customer Jane Smith, AccountID 2, Transaction: Withdrawal, Amount:

500, Date: 26-JUN-2025

Annual fee of ???200 deducted from Account ID 1

Annual fee of ???200 deducted from Account ID 2

Interest rate reduced for Loan ID 2

ACCOUNTID CUSTOMERID ACCOUNTTYPE BALANCE LASTMODIF

---------- ---------- -------------------- ---------- ---------

1 1 Savings 7800 26-JUN-25

2 2 Checking 1800 26-JUN-25

LOANID CUSTOMERID LOANAMOUNT INTERESTRATE STARTDATE ENDDATE

---------- ---------- ---------- ------------ --------- ---------

1 1 5000 5 26-JUN-25 26-JUN-30

2 2 9000 5.5 26-JUN-25 26-JUN-29

LOANID CUSTOMERID LOANAMOUNT INTERESTRATE STARTDATE ENDDATE

---------- ---------- ---------- ------------ --------- ---------

1 1 5000 5 26-JUN-25 26-JUN-30

2 2 9000 5.5 26-JUN-25 26-JUN-29

**Exercise 7: Packages**

**Code:**

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 Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

CREATE OR REPLACE PACKAGE CustomerManagement AS

PROCEDURE AddCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_dob DATE, p\_balance NUMBER);

PROCEDURE UpdateCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_balance NUMBER);

FUNCTION GetCustomerBalance(p\_id NUMBER) RETURN NUMBER;

END CustomerManagement;

/

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_dob DATE, p\_balance NUMBER) IS

BEGIN

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (p\_id, p\_name, p\_dob, p\_balance, SYSDATE);

END;

PROCEDURE UpdateCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_balance NUMBER) IS

BEGIN

UPDATE Customers

SET Name = p\_name,

Balance = p\_balance,

LastModified = SYSDATE

WHERE CustomerID = p\_id;

END;

FUNCTION GetCustomerBalance(p\_id NUMBER) RETURN NUMBER IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Customers WHERE CustomerID = p\_id;

RETURN v\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN NULL;

END;

END CustomerManagement;

/

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireEmployee(p\_id NUMBER, p\_name VARCHAR2, p\_position VARCHAR2,

p\_salary NUMBER, p\_department VARCHAR2, p\_hiredate DATE);

PROCEDURE UpdateEmployee(p\_id NUMBER, p\_salary NUMBER);

FUNCTION CalculateAnnualSalary(p\_id NUMBER) RETURN NUMBER;

END EmployeeManagement;

/

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireEmployee(p\_id NUMBER, p\_name VARCHAR2, p\_position VARCHAR2,

p\_salary NUMBER, p\_department VARCHAR2, p\_hiredate DATE) IS

BEGIN

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (p\_id, p\_name, p\_position, p\_salary, p\_department, p\_hiredate);

END;

PROCEDURE UpdateEmployee(p\_id NUMBER, p\_salary NUMBER) IS

BEGIN

UPDATE Employees

SET Salary = p\_salary

WHERE EmployeeID = p\_id;

END;

FUNCTION CalculateAnnualSalary(p\_id NUMBER) RETURN NUMBER IS

v\_salary NUMBER;

BEGIN

SELECT Salary INTO v\_salary FROM Employees WHERE EmployeeID = p\_id;

RETURN v\_salary \* 12;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN NULL;

END;

END EmployeeManagement;

/

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenAccount(p\_id NUMBER, p\_custid NUMBER, p\_type VARCHAR2, p\_balance NUMBER);

PROCEDURE CloseAccount(p\_id NUMBER);

FUNCTION GetTotalBalance(p\_custid NUMBER) RETURN NUMBER;

END AccountOperations;

/

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenAccount(p\_id NUMBER, p\_custid NUMBER, p\_type VARCHAR2, p\_balance NUMBER) IS

BEGIN

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (p\_id, p\_custid, p\_type, p\_balance, SYSDATE);

END;

PROCEDURE CloseAccount(p\_id NUMBER) IS

BEGIN

DELETE FROM Accounts WHERE AccountID = p\_id;

END;

FUNCTION GetTotalBalance(p\_custid NUMBER) RETURN NUMBER IS

v\_total NUMBER := 0;

BEGIN

SELECT SUM(Balance) INTO v\_total FROM Accounts WHERE CustomerID = p\_custid;

RETURN NVL(v\_total, 0);

END;

END AccountOperations;

/

SET SERVEROUTPUT ON;

BEGIN

CustomerManagement.AddCustomer(101, 'Emily Clark', TO\_DATE('1988-12-12', 'YYYY-MM-DD'), 5000);

CustomerManagement.UpdateCustomer(101, 'Emily Clark Updated', 6000);

DBMS\_OUTPUT.PUT\_LINE('Customer 101 Balance: ' || CustomerManagement.GetCustomerBalance(101));

EmployeeManagement.HireEmployee(201, 'David Grey', 'Analyst', 50000, 'Finance', SYSDATE);

DBMS\_OUTPUT.PUT\_LINE('Annual Salary for Employee 201: ' || EmployeeManagement.CalculateAnnualSalary(201));

AccountOperations.OpenAccount(301, 101, 'Savings', 6000);

DBMS\_OUTPUT.PUT\_LINE('Total Balance for Customer 101: ' || AccountOperations.GetTotalBalance(101));

AccountOperations.CloseAccount(301);

END;

/

**Output:**

Customer 101 Balance: 6000

Annual Salary for Employee 201: 600000

Total Balance for Customer 101: 6000