**Exercise 1: Control Structures**

Scenario 1: Apply 1% Discount to Customers Above 60

BEGIN

FOR rec IN(SELECT id,age,interest\_rate FROM customers)LOOP

IF rec.age>60 THEN

UPDATE CUSTOMERS

SET interest\_rate=interest\_rate-1

WHERE id=rec.id;

DBMS\_OUTPUT.PUT\_LINE('DISCOUNT APPLIED TO ID'|| rec.id);

END IF;

END LOOP;

END;

Output:

DISCOUNT APPLIED TO ID3  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.080

**Scenario 2: Promote to VIP If Balance > 10,000**

BEGIN

FOR rec IN(SELECT name,id,isvip,balance FROM customers) LOOP

IF rec.balance>10000 THEN

UPDATE customers

SET isvip =TRUE

WHERE id=rec.id;

DBMS\_OUTPUT.PUT\_LINE('Customer '|| rec.name|| ' is promoted to vip');

END IF;

END LOOP;

END;

Output:

Customer Dinesh is promoted to vip  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.008

**Scenario 3: Reminders for Loans Due in Next 30 Days**

BEGIN

FOR rec IN(SELECT l.loan\_id,c.name,l.due\_date

FROM loans l

JOIN customerS c ON l.customer\_id=c.id

WHERE l.due\_date<= SYSDATE+30) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: '|| rec.name||' your loan ID'|| rec.loan\_id ||' is due on ' || TO\_CHAR(rec.due\_date, 'DD-Mon-YYYY'));

END LOOP;

END;

Output:

Reminder: Dinesh your loan ID103 is due on 28-Jun-2025

**Exercise 2: Error Handling**

**Scenario 1:** Handle exceptions during fund transfers between accounts.

CREATE TABLE accounts(

accid NUMBER PRIMARY KEY,

accname VARCHAR(50),

balance NUMBER

);

INSERT INTO accounts VALUES (1, 'Yash', 10000);

INSERT INTO accounts VALUES (2, 'Saran', 8000);

CREATE OR REPLACE PROCEDURE safetransfer(

from\_acc IN NUMBER,

to\_acc IN NUMBER,

amount IN NUMBER

)

IS

from\_balance NUMBER;

BEGIN

SELECT balance INTO from\_balance FROM accounts WHERE accid=from\_acc;

IF from\_balance < amount THEN

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

END IF;

UPDATE accounts SET balance = balance-amount WHERE accid=from\_acc;

UPDATE accounts SET balance= balance+amount WHERE accid=to\_acc;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('TRANSFER SUCCESSFUL');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('ERROR OCCURED' ||SQLERRM);

END;

BEGIN

safetransfer(1,2,2000);

END;

Output:

TRANSFER SUCCESSFUL

PL/SQL procedure successfully completed.

**Scenario 2:** Manage errors when updating employee salaries.

CREATE TABLE employees (

empid NUMBER,

empname VARCHAR(50),

salary NUMBER

);

INSERT INTO employees VALUES (101, 'Arun', 30000);

INSERT INTO employees VALUES (102, 'Divya', 45000);

CREATE OR REPLACE PROCEDURE Incrementsalary(

pemp\_id IN NUMBER,

percent IN NUMBER

)

IS

BEGIN

UPDATE employees

SET salary= salary+(salary\*percent/100)

WHERE empid=pemp\_id;

IF SQL%ROWCOUNT=0 THEN

RAISE\_APPLICATION\_ERROR(-20002,'ID NOT FOUND');

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('SALARY UPDATED SUCCESSFULLY');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('ERROR'|| SQLERRM);

END;

Output:

SQL> BEGIN  
Incrementsalary(101,10);  
Incrementsalary(999,10);  
END;

SALARY UPDATED SUCCESSFULLY  
ERRORORA-20002: ID NOT FOUND  
  
  
PL/SQL procedure successfully completed.

**Scenario 3:** Ensure data integrity when adding a new customer.

CREATE TABLE customers2 (

id NUMBER PRIMARY KEY,

name VARCHAR2(50),

age NUMBER,

balance NUMBER,

isvip BOOLEAN,

interest\_rate NUMBER

);

CREATE OR REPLACE PROCEDURE addnewcustomer(

p\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_age IN NUMBER,

p\_balance IN NUMBER

)

IS

BEGIN

INSERT INTO customers2(id,name,age,balance,isvip,interest\_rate)

VALUES(p\_id,p\_name,p\_age,p\_balance,FALSE,9.0);

COMMIT;

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

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Customer with ID ' || p\_id || ' already exists.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Some other error occurred: ' || SQLERRM);

END;

BEGIN

ADDNEWCUSTOMER(1,'Yash',25,12000);

END;

BEGIN

AddNewCustomer(1, 'SARAN', 30, 10000);

END;

Output:

SQL> BEGIN   
ADDNEWCUSTOMER(1,'Yash',25,12000);  
END;

Customer added successfully.  
  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.021

SQL> BEGIN  
AddNewCustomer(1, 'SARAN', 30, 10000);  
END;

Error: Customer with ID 1 already exists.  
  
  
PL/SQL procedure successfully completed.

**EXERCISE 3:** **Stored Procedures**

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

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

IS

BEGIN

UPDATE ACCOUNTS

SET BALANCE= BALANCE + (BALANCE\*0.01)

WHERE ACCOUNT\_TYPE ='Savings';

DBMS\_OUTPUT.PUT\_LINE('MONTHLY INTEREST ADDED TO SAVINGS ACCOUNT');

END;

BEGIN

ProcessMonthlyInterest;

END;

Output:

SQL> BEGIN   
ProcessMonthlyInterest;  
END;

MONTHLY INTEREST ADDED TO SAVINGS ACCOUNT  
  
  
PL/SQL procedure successfully completed.

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

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

DEPT\_ID IN NUMBER,

BONUS\_PERCENT IN NUMBER

)

IS

BEGIN

UPDATE EMPLOYEES

SET SALARY=SALARY+(SALARY\* BONUS\_PERCENT/100)

WHERE DEPARTMENT\_ID=DEPT\_ID;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department'|| dept\_id);

END;

BEGIN

UpdateEmployeeBonus(10,10);

END;

Output:

SQL> BEGIN   
UpdateEmployeeBonus(10,10);  
END;

Bonus applied to department10  
  
  
PL/SQL procedure successfully completed.

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

CREATE OR REPLACE PROCEDURE TransferFunds(

FROM\_ACCOUNT IN NUMBER,

TO\_ACCOUNT IN NUMBER,

AMOUNT IN NUMBER

)

IS

FROM\_BALANCE NUMBER;

BEGIN

SELECT BALANCE INTO FROM\_BALANCE FROM ACCOUNTS

WHERE ACCOUNT\_ID=FROM\_ACCOUNT FOR UPDATE;

IF FROM\_BALANCE>=AMOUNT THEN

UPDATE ACCOUNTS

SET BALANCE = BALANCE-AMOUNT

WHERE ACCOUNT\_ID=FROM\_ACCOUNT;

UPDATE ACCOUNTS

SET BALANCE = BALANCE+AMOUNT

WHERE ACCOUNT\_ID=TO\_ACCOUNT;

DBMS\_OUTPUT.PUT\_LINE('TRANSFERRED '||AMOUNT|| ' FROM ' || FROM\_ACCOUNT || ' TO ACCOUNT ' || TO\_ACCOUNT);

ELSE

DBMS\_OUTPUT.PUT\_LINE('ERROR, INSUFFICIENT FUNDS'|| FROM\_ACCOUNT);

END IF;

END;

BEGIN

TransferFunds(2, 1, 500);

END;

OUTPUT:

SQL> BEGIN  
TransferFunds(2, 1, 500);  
END;

TRANSFERRED 500 FROM 2 TO ACCOUNT 1  
  
  
PL/SQL procedure successfully completed.