**Ques 1 : Control Structures**

Scenario 1: Interest Rate Discount for Seniors

Objective:

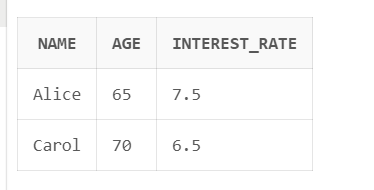
Apply a 1% interest rate discount for customers aged over 60.

PL/SQL Code:

BEGIN  
 FOR cust IN (  
 SELECT customer\_id, interest\_rate  
 FROM my\_customers  
 WHERE age > 60  
 ) LOOP  
 UPDATE my\_customers  
 SET interest\_rate = interest\_rate - 1  
 WHERE customer\_id = cust.customer\_id;  
 END LOOP;  
END;

Sample Output:

SELECT name, age, interest\_rate  
FROM my\_customers  
WHERE age > 60;  
  
Output :



Scenario 2: Promote to VIP Based on Balance

Objective:

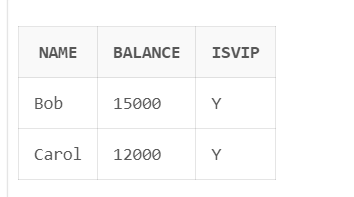
Set ISVIP = 'Y' for customers with balance over $10,000.

PL/SQL Code:

BEGIN  
 FOR cust IN (  
 SELECT customer\_id  
 FROM my\_customers  
 WHERE balance > 10000  
 ) LOOP  
 UPDATE my\_customers  
 SET isvip = 'Y'  
 WHERE customer\_id = cust.customer\_id;  
 END LOOP;  
END;

Sample Output:

SELECT name, balance, isvip  
FROM my\_customers  
WHERE isvip = 'Y';  
  
 Output:



Scenario 3: Loan Reminder Messages

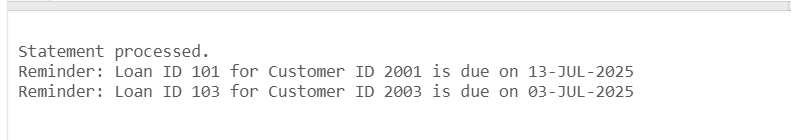
Objective:

Send reminder messages for loans due within the next 30 days.

PL/SQL Code:

BEGIN  
 FOR loan\_rec IN (  
 SELECT loan\_id, customer\_id, due\_date  
 FROM loans  
 WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30  
 ) LOOP  
 DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.loan\_id ||  
 ' for Customer ID ' || loan\_rec.customer\_id ||  
 ' is due on ' || TO\_CHAR(loan\_rec.due\_date, 'DD-MON-YYYY'));  
 END LOOP;  
END;

Output:



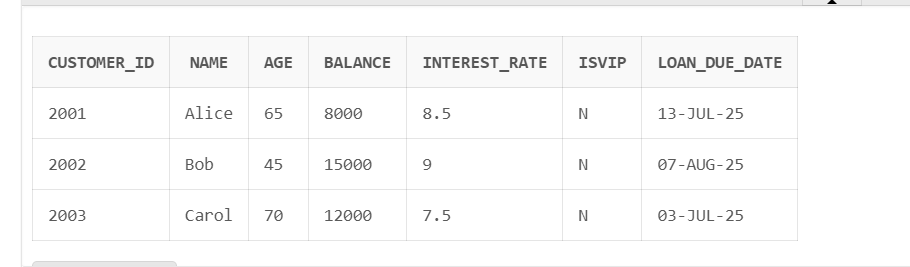
Appendix: Table Creation and Sample Data

-- Table: MY\_CUSTOMERS

CREATE TABLE MY\_CUSTOMERS (  
 CUSTOMER\_ID NUMBER,  
 NAME VARCHAR2(50),  
 AGE NUMBER,  
 BALANCE NUMBER,  
 INTEREST\_RATE NUMBER,  
 ISVIP CHAR(1),  
 LOAN\_DUE\_DATE DATE  
);  
  
-- Sample Data

INSERT INTO MY\_CUSTOMERS VALUES (2001, 'Alice', 65, 8000, 8.5, 'N', SYSDATE + 15);  
INSERT INTO MY\_CUSTOMERS VALUES (2002, 'Bob', 45, 15000, 9.0, 'N', SYSDATE + 40);  
INSERT INTO MY\_CUSTOMERS VALUES (2003, 'Carol', 70, 12000, 7.5, 'N', SYSDATE + 5);  
COMMIT;

--Table

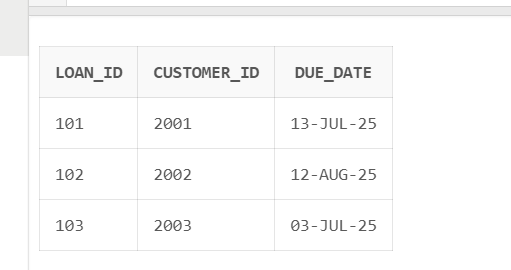


Alternative Table: LOANS (if separate)

-- Table: LOANS  
CREATE TABLE LOANS (  
 LOAN\_ID NUMBER,  
 CUSTOMER\_ID NUMBER,  
 DUE\_DATE DATE  
);  
  
-- Sample Data

INSERT INTO LOANS VALUES (101, 2001, SYSDATE + 15);  
INSERT INTO LOANS VALUES (102, 2002, SYSDATE + 45);  
INSERT INTO LOANS VALUES (103, 2003, SYSDATE + 5);  
COMMIT;

--Table



**Ques 2 : Stored Procedures**

Scenario 1: Monthly Interest Processing

Objective:

Apply a 1% interest rate to all savings accounts.

Step 1: Table Creation and Sample Data

CREATE TABLE savings\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

balance NUMBER

);

-- Sample Data

INSERT INTO savings\_accounts VALUES (101, 'Alice', 10000);

INSERT INTO savings\_accounts VALUES (102, 'Bob', 15000);

INSERT INTO savings\_accounts VALUES (103, 'Charlie', 8000);

COMMIT;

Step 2: Stored Procedure

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01);

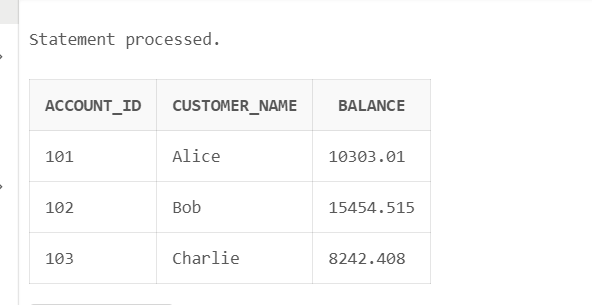
END;

Step 3: Verification

EXEC ProcessMonthlyInterest;

SELECT \* FROM savings\_accounts;

Output :



Scenario 2: Employee Bonus Update

Objective**:**

Add a bonus percentage to employees in a specific department.

Step 1:Table Creation and Sample Data

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department\_id NUMBER,

salary NUMBER

);

-- Sample Data

INSERT INTO employees VALUES (1, 'John', 101, 50000);

INSERT INTO employees VALUES (2, 'Jane', 101, 55000);

INSERT INTO employees VALUES (3, 'Tom', 102, 60000);

COMMIT;

Step 2: Stored Procedure

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;

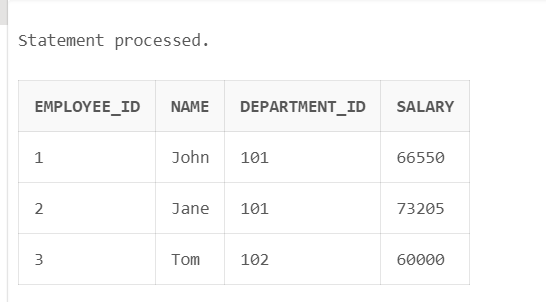
END;

Step 3: Verification

EXEC UpdateEmployeeBonus(101, 10);

SELECT \* FROM employees;

Output :



Scenario 3: Fund Transfer Between Accounts

Objective:

Transfer funds between two accounts if source has enough balance.

Step 1: Table Creation and Sample Data

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

balance NUMBER

);

-- Sample Data

INSERT INTO accounts VALUES (1001, 'David', 10000);

INSERT INTO accounts VALUES (1002, 'Eva', 7000);

INSERT INTO accounts VALUES (1003, 'Frank', 12000);

COMMIT;

Step 2: Stored Procedure

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;

IF from\_balance < amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance in source account.');

END IF;

UPDATE accounts SET balance = balance - amount WHERE account\_id = from\_account;

UPDATE accounts SET balance = balance + amount WHERE account\_id = to\_account;

END;

Step 3: Verification

EXEC TransferFunds(1001, 1002, 500);

SELECT \* FROM accounts;

Output :

