PL/SQL\_Excercise

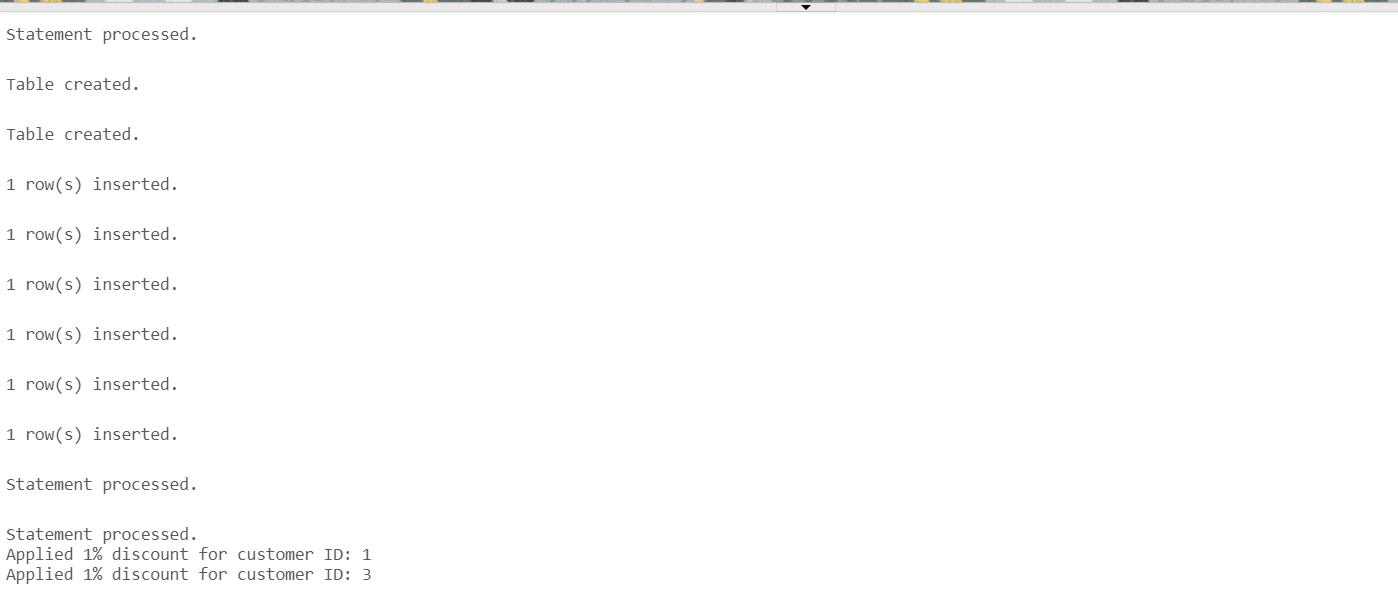
EXERCISE -1

# Scenario 1: Apply Interest Rate Discount for Customers Over 60

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.

Solution:

DECLARE  
 CURSOR cur\_customers IS  
 SELECT customer\_id  
 FROM customers  
 WHERE age > 60;  
  
BEGIN  
 FOR cust IN cur\_customers LOOP  
 UPDATE loans  
 SET interest\_rate = interest\_rate - 1  
 WHERE customer\_id = cust.customer\_id;  
  
 DBMS\_OUTPUT.PUT\_LINE('Applied 1% discount for customer ID: ' || cust.customer\_id);  
 END LOOP;  
  
 COMMIT;  
END;

Output:

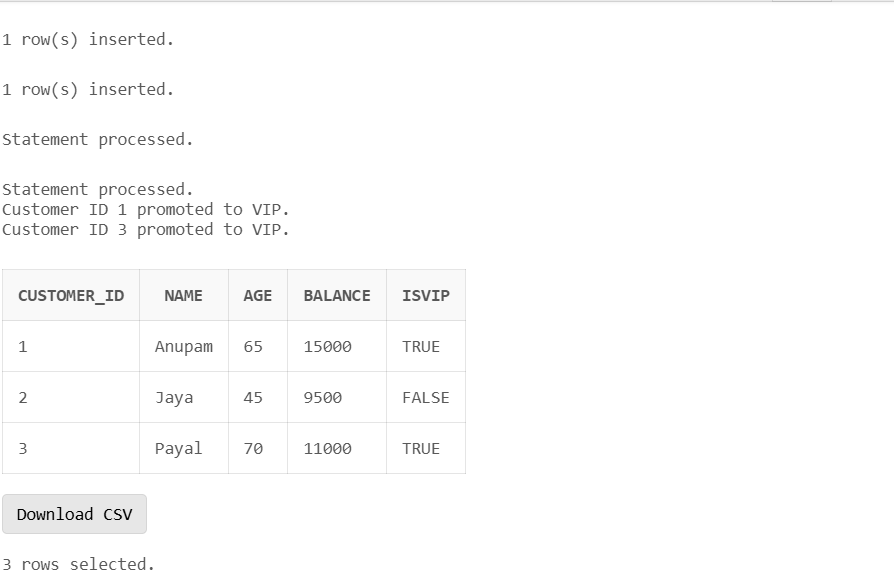
# Scenario 2: Promote Customers to VIP if Balance > $10,000

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.

Solution:

DECLARE  
 CURSOR cur\_customers IS  
 SELECT customer\_id, balance  
 FROM customers  
 WHERE balance > 10000;  
  
BEGIN  
 FOR cust IN cur\_customers LOOP  
 UPDATE customers  
 SET isvip = 'TRUE'  
 WHERE customer\_id = cust.customer\_id;  
  
 DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || cust.customer\_id || ' promoted to VIP.');  
 END LOOP;  
  
 COMMIT;  
END;

Output:



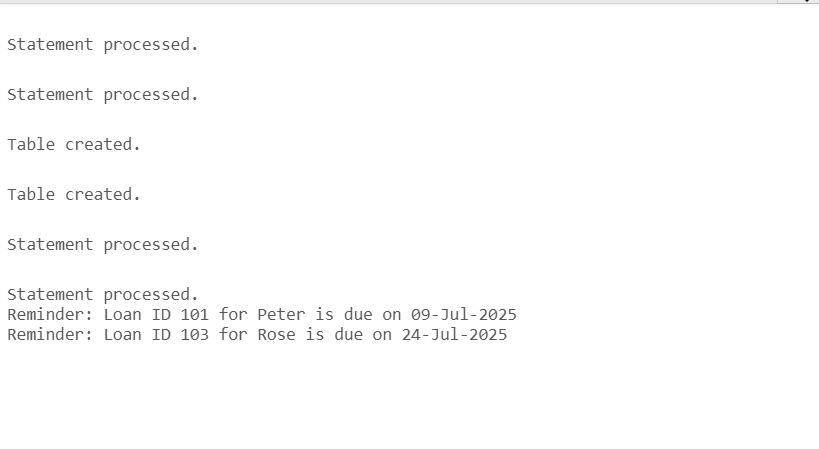
# Scenario 3: Loan Reminders for Due Dates Within 30 Days

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

Solution:

DECLARE  
 CURSOR cur\_loans IS  
 SELECT loan\_id, customer\_id, due\_date  
 FROM loans  
 WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30;  
  
 v\_customer\_name VARCHAR2(100);  
BEGIN  
 FOR loan\_rec IN cur\_loans LOOP  
 SELECT name INTO v\_customer\_name  
 FROM customers  
 WHERE customer\_id = loan\_rec.customer\_id;  
  
 DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.loan\_id ||  
 ' for customer ' || v\_customer\_name ||  
 ' is due on ' || TO\_CHAR(loan\_rec.due\_date, 'DD-MON-YYYY'));  
 END LOOP;  
END;

OUTPUT:



**EXCERCISE– 3**

**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.

CREATE TABLE savings\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER

);

INSERT INTO savings\_accounts VALUES (1, 101, 1000);

INSERT INTO savings\_accounts VALUES (2, 102, 2000);

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE savings\_accounts

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

DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

END;

/

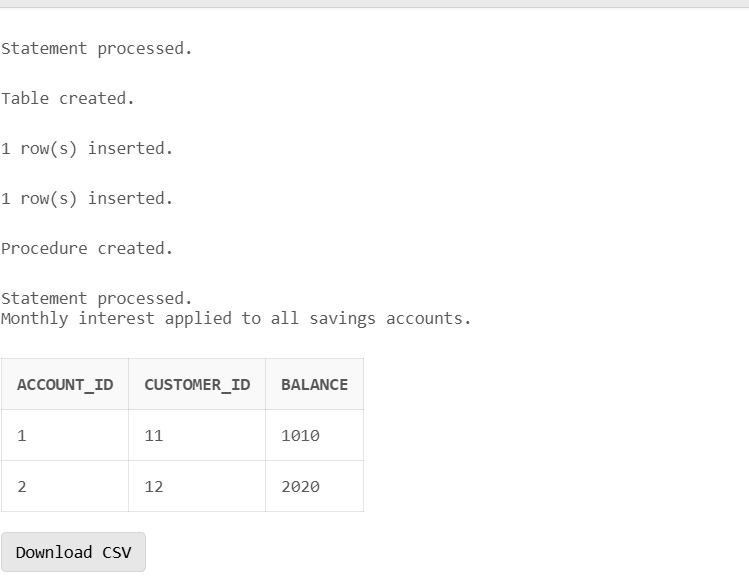
BEGIN

ProcessMonthlyInterest;

END;

/

OUTPUT:

****

**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.

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER

) AS

BEGIN

UPDATE employees

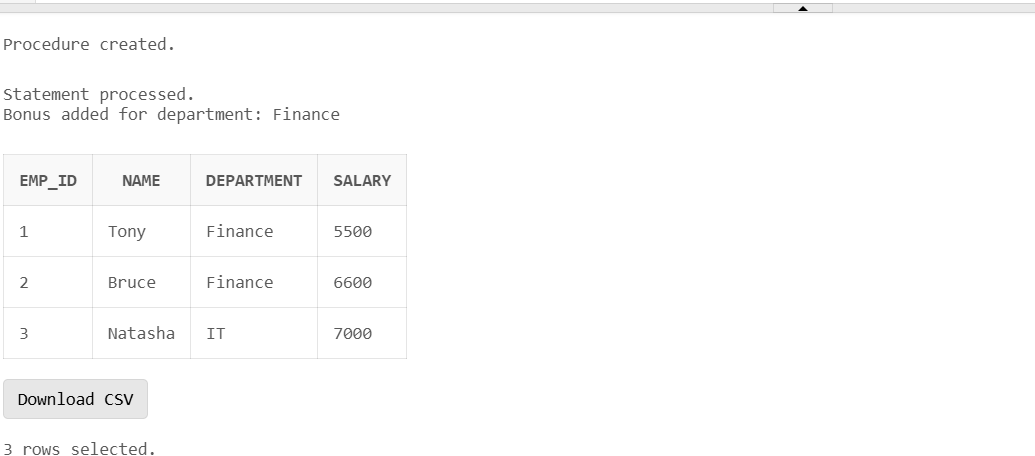
SET salary = salary + (salary \* p\_bonus\_pct / 100)

WHERE department = p\_department;

DBMS\_OUTPUT.PUT\_LINE('Bonuses updated for department: ' || p\_department);

END;

OUTPUT:

****

**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.

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

-- Check balance of source account

SELECT balance INTO v\_balance

FROM bank\_accounts

WHERE account\_id = p\_from\_account;

IF v\_balance < p\_amount THEN

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

END IF;

-- Deduct from source

UPDATE bank\_accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account;

-- Add to destination

UPDATE bank\_accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: ' || p\_amount || ' transferred from ' || p\_from\_account || ' to ' || p\_to\_account);

END;

OUTPUT: