PL/SQL Exercise 1: Control Structures

CREATE TABLES AND INSERT SAMPLE DATA

```
 // Create CUSTOMERS table  CREATE TABLE
customers ( customer_id NUMBER PRIMARY KEY,
name VARCHAR2(100),
age NUMBER,
balance NUMBER(10, 2), isvip
VARCHAR2(5)
);

// Create LOANS table CREATE TABLE
loans (
loan_id NUMBER PRIMARY KEY,
customer_id NUMBER, interest_rate
NUMBER(5, 2), due_date DATE,
FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

// Create SAVINGS_ACCOUNTS table CREATE
TABLE savings_accounts ( account_id NUMBER
PRIMARY KEY, customer_id NUMBER,
balance NUMBER(10, 2),
FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

// Create ACCOUNTS table (for transfers) CREATE TABLE
accounts (
account_id NUMBER PRIMARY KEY, customer_id
NUMBER,
balance NUMBER(10, 2),
FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);


// Create EMPLOYEES table CREATE TABLE
employees (
emp_id NUMBER PRIMARY KEY, name
VARCHAR2(100),
department_id NUMBER, salary
NUMBER(10, 2)
);
-- Insert Sample Data into CUSTOMERS
INSERT INTO customers VALUES (1, 'Esha', 34, 5600.00, 'FALSE');
INSERT INTO customers VALUES (2, 'Ravi', 61, 11200.00, 'FALSE');
INSERT INTO customers VALUES (3, 'Mira', 47, 10200.00, 'FALSE');
INSERT INTO customers VALUES (4, 'Kabir', 70, 9000.00, 'FALSE');

-- Insert Sample Data into LOANS
INSERT INTO loans VALUES (201, 1, 6.0, SYSDATE + 20);
INSERT INTO loans VALUES (202, 2, 7.5, SYSDATE + 25);
INSERT INTO loans VALUES (203, 3, 7.0, SYSDATE + 5);

-- Insert Sample Data into SAVINGS_ACCOUNTS
INSERT INTO savings_accounts VALUES (301, 1, 1200.00);
INSERT INTO savings_accounts VALUES (302, 2, 2800.00);

-- Insert Sample Data into ACCOUNTS
INSERT INTO accounts VALUES (4001, 1, 3000.00);
INSERT INTO accounts VALUES (4002, 2, 1800.00);

-- Insert Sample Data into EMPLOYEES
INSERT INTO employees VALUES (501, 'Anita', 20, 3200.00);
INSERT INTO employees VALUES (502, 'Vikram', 20, 3700.00);
```

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.

<u>QUERY:</u>

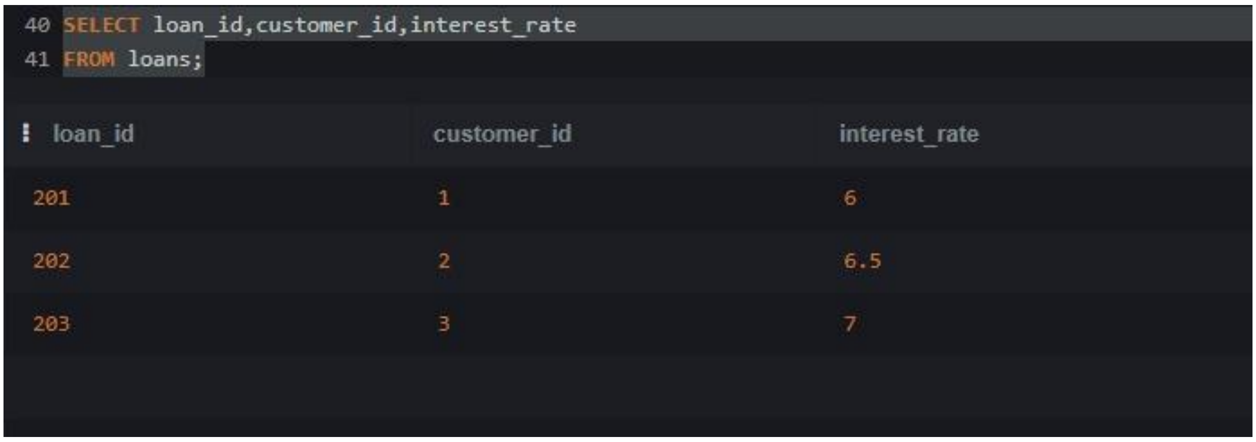
```
UPDATE loans

SET interest_rate = interest_rate - 1 WHERE

customer_id IN (

        SELECT customer_id FROM customers WHERE age > 60

);
```



```
SELECT loan_id,customer_id,interest_rate
FROM loans;
```

loan_id	customer_id	interest_rate
201	1	6
202	2	6.5
203	3	7

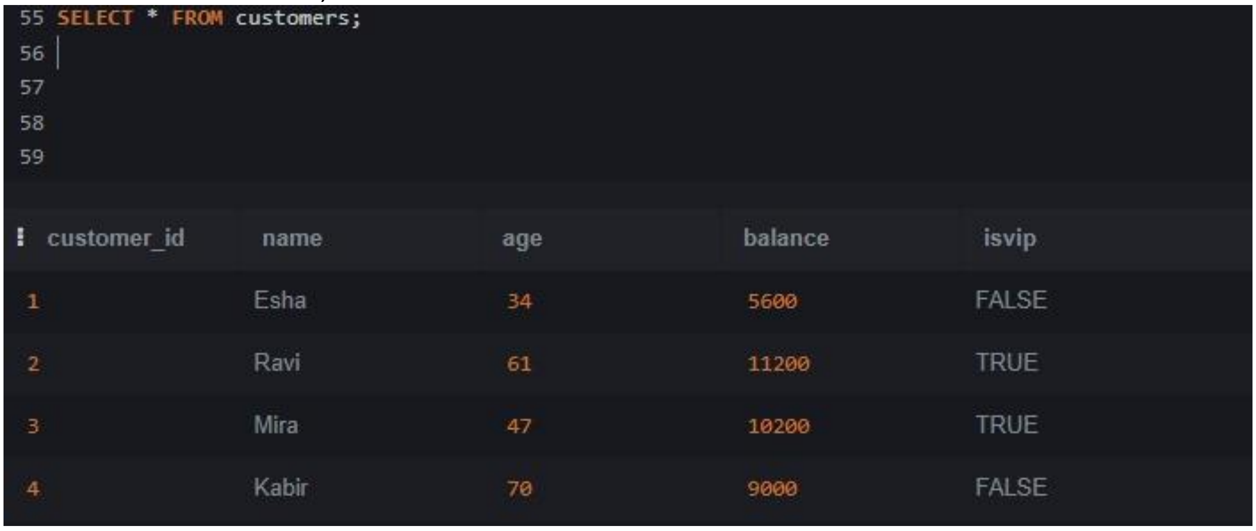
Scenario 2: A customer can be promoted to VIP status based on their balance.

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.

QUERY:

```
UPDATE customers
SET isvip = 'TRUE'
WHERE balance > 10000;
```



```
SELECT * FROM customers;
```

customer_id	name	age	balance	isvip
1	Esha	34	5600	FALSE
2	Ravi	61	11200	TRUE
3	Mira	47	10200	TRUE
4	Kabir	70	9000	FALSE

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.

QUERY:

SELECT

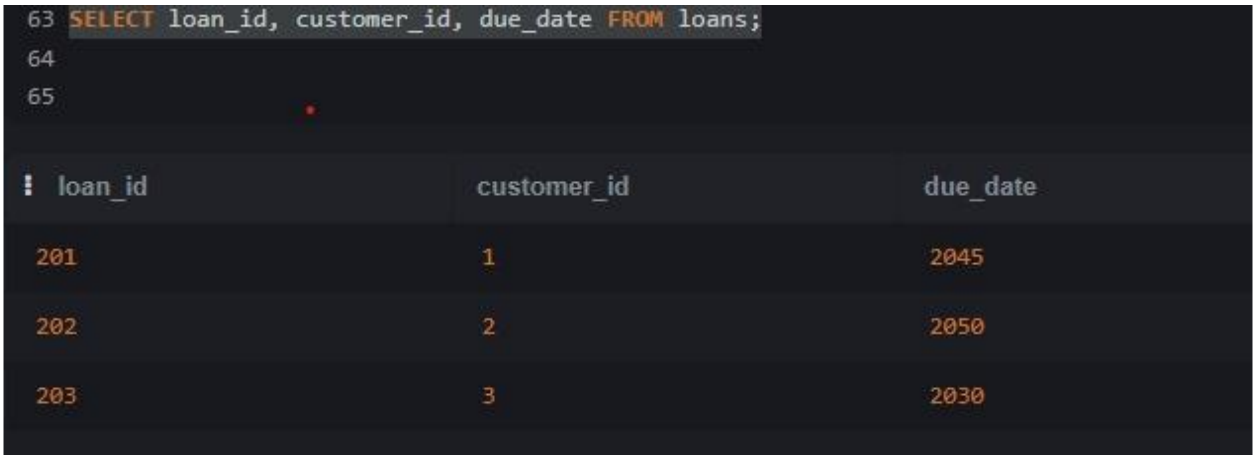
```
 'Reminder: Loan ID ' || loan_id ||

 ' for Customer ID ' || customer_id ||

 ' is due on ' || strftime('%d-%b-%Y', due_date) AS reminder_message

FROM loans

WHERE due_date BETWEEN DATE('now') AND DATE('now', '+30 day');
```



```
SELECT loan_id, customer_id, due_date FROM loans;
```

loan_id	customer_id	due_date
201	1	2045
202	2	2050
203	3	2030

Exercise 3: Stored Procedures

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

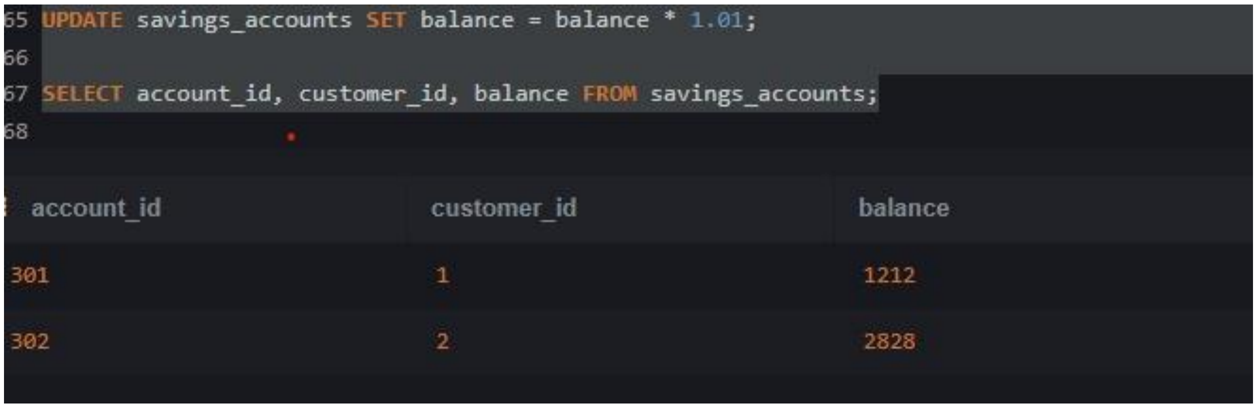
Question:

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.

QUERY:

```
UPDATE savings_accounts SET
balance = balance * 1.01;
```

SELECT account_id, customer_id, balance
FROM savings_accounts;



```
UPDATE savings_accounts SET balance = balance * 1.01;

SELECT account_id, customer_id, balance FROM savings_accounts;
```

account_id	customer_id	balance
301	1	1212
302	2	2828

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

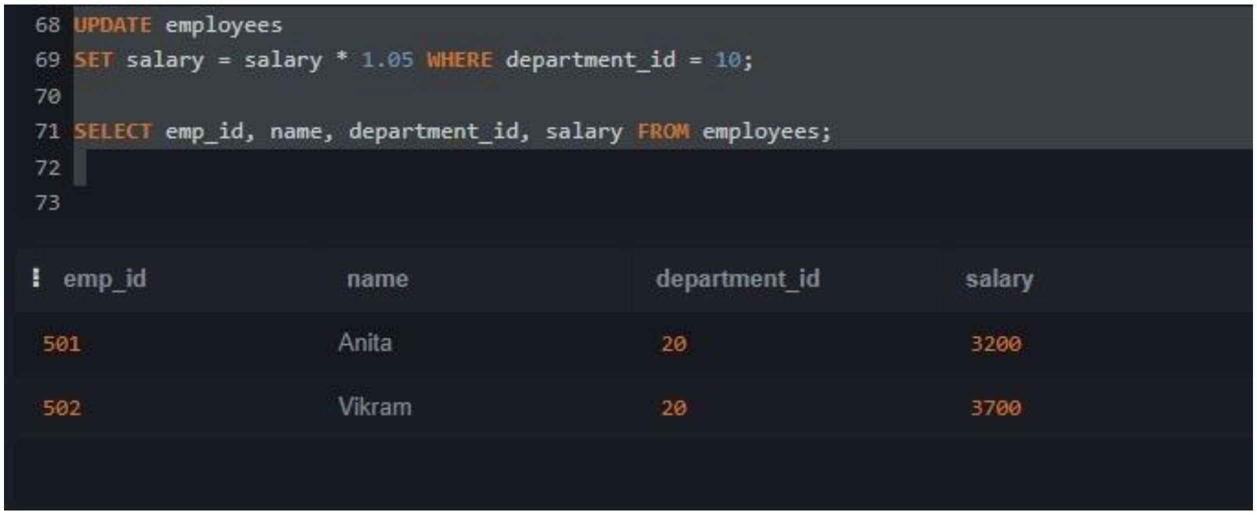
<u>Question:</u>

Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

<u>QUERY:</u>

UPDATE employees
SET salary = salary *
1.05 WHERE
department_id=10;

SELECT emp_id, name, department_id,
salary FROM employees;



```
UPDATE employees
SET salary = salary * 1.05 WHERE department_id = 10;

SELECT emp_id, name, department_id, salary FROM employees;
```

emp_id	name	department_id	salary
501	Anita	20	3200
502	Vikram	20	3700

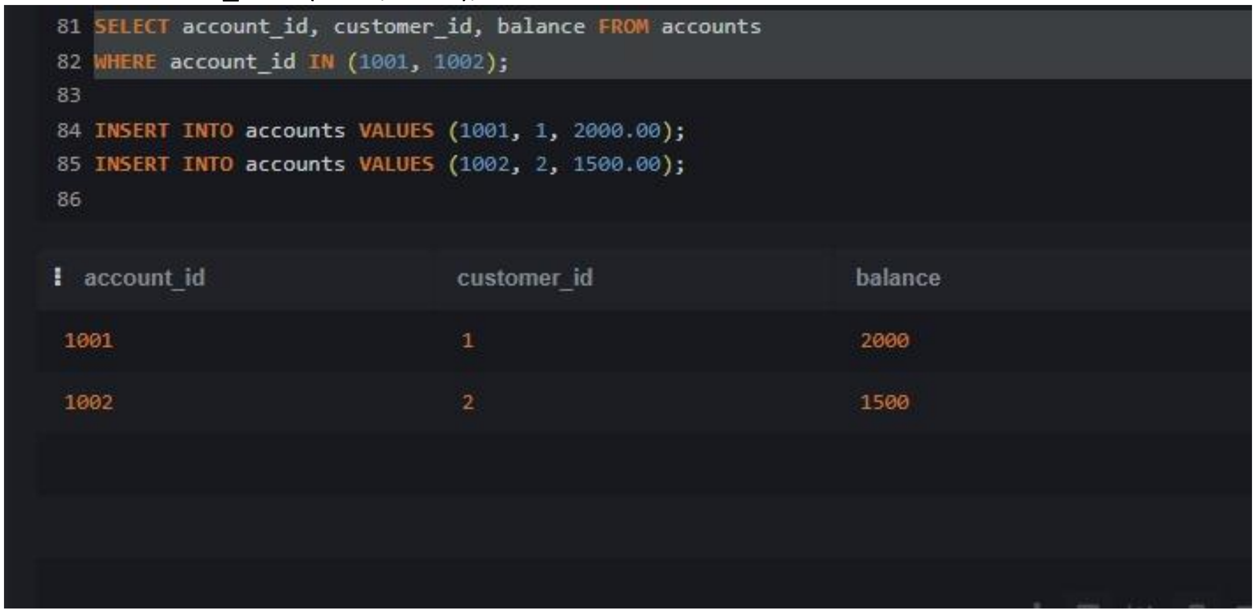
Scenario 3: Customers should be able to transfer funds between their accounts.

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.

QUERY:

SELECT account_id, customer_id, balance FROM accounts
WHERE account_id IN (1001, 1002);



```
SELECT account_id, customer_id, balance FROM accounts
WHERE account_id IN (1001, 1002);

INSERT INTO accounts VALUES (1001, 1, 2000.00);
INSERT INTO accounts VALUES (1002, 2, 1500.00);
```

account_id	customer_id	balance
1001	1	2000
1002	2	1500

KANMANI MURUGHAIYAN
Superset id: 6407636