Exercise 1: Control Structures

SCENARIO 1: 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.

```
BEGIN
  FOR cust_rec IN (
    SELECT customer_id, loan_interest_rate, age
    FROM customers
    WHERE age > 60
  ) LOOP
    UPDATE customers
    SET loan_interest_rate = loan_interest_rate - 1
    WHERE customer_id = cust_rec.customer_id;

    DBMS_OUTPUT.PUT_LINE('Discount applied for Customer ID: ' || cust_rec.customer_id);
  END LOOP;
  COMMIT;
END;
```

SCENARIO 2: 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.

```
BEGIN
  FOR cust_rec IN (
    SELECT customer_id, balance
    FROM customers
```

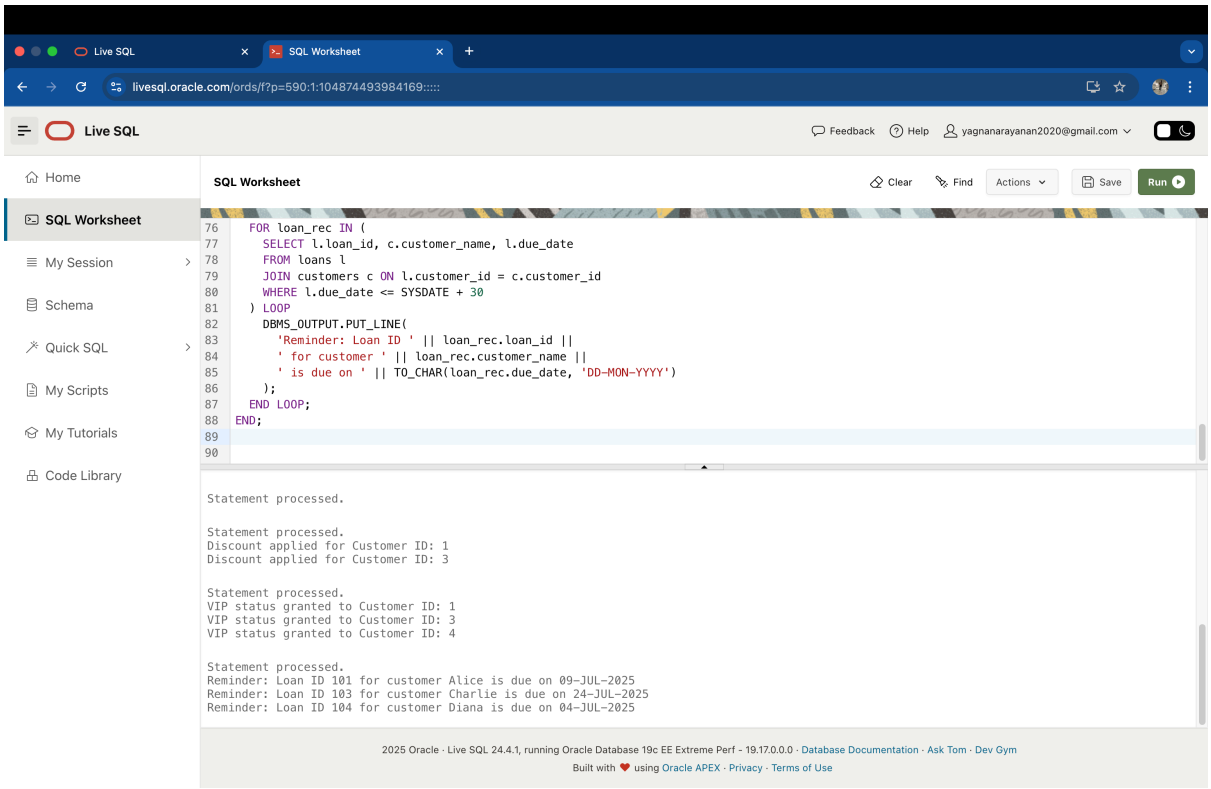
```
    WHERE balance > 10000
  ) LOOP
    UPDATE customers
    SET isvip = 'TRUE'
    WHERE customer_id = cust_rec.customer_id;

    DBMS_OUTPUT.PUT_LINE('VIP status granted to Customer ID: ' || cust_rec.customer_id);
  END LOOP;
  COMMIT;
END;
```

SCENARIO 3: Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

```
BEGIN
  FOR loan_rec IN (
    SELECT l.loan_id, c.customer_name, l.due_date
    FROM loans l
    JOIN customers c ON l.customer_id = c.customer_id
    WHERE l.due_date <= SYSDATE + 30
  ) LOOP
    DBMS_OUTPUT.PUT_LINE(
      'Reminder: Loan ID ' || loan_rec.loan_id ||
      ' for customer ' || loan_rec.customer_name ||
      ' is due on ' || TO_CHAR(loan_rec.due_date, 'DD-MON-YYYY')
    );
  END LOOP;
END;
```

OUTPUT:


Live SQL
SQL Worksheet
livesql.oracle.com/ords/f?p=590:1:104874493984169::::
Live SQL
Feedback
Help
yagnanarayanan2020@gmail.com
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
76 FOR loan_rec IN (
77 SELECT l.loan_id, c.customer_name, l.due_date
78 FROM loans l
79 JOIN customers c ON l.customer_id = c.customer_id
80 WHERE l.due_date <= SYSDATE + 30
81) LOOP
82 DBMS_OUTPUT.PUT_LINE(
83 'Reminder: Loan ID ' || loan_rec.loan_id ||
84 ' for customer ' || loan_rec.customer_name ||
85 ' is due on ' || TO_CHAR(loan_rec.due_date, 'DD-MON-YYYY')
86);
87 END LOOP;
88 END;
89
90
Statement processed.
Statement processed.
Discount applied for Customer ID: 1
Discount applied for Customer ID: 3
Statement processed.
VIP status granted to Customer ID: 1
VIP status granted to Customer ID: 3
VIP status granted to Customer ID: 4
Statement processed.
Reminder: Loan ID 101 for customer Alice is due on 09-JUL-2025
Reminder: Loan ID 103 for customer Charlie is due on 24-JUL-2025
Reminder: Loan ID 104 for customer Diana is due on 04-JUL-2025
2025 Oracle · Live SQL 24.4.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with using Oracle APEX · Privacy · Terms of Use

Exercise 3: Stored Procedures

Scenario 1: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 OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
  UPDATE accounts
  SET balance = balance + (balance * 0.01)
  WHERE UPPER(account_type) = 'SAVINGS';

  DBMS_OUTPUT.PUT_LINE('Interest applied to all savings accounts.');
  COMMIT;
END;
```

Scenario 2: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_dept       IN VARCHAR2,
  p_bonus_pct  IN NUMBER
) IS
```

```
BEGIN
  UPDATE employees
  SET salary = salary + (salary * p_bonus_pct / 100)
  WHERE LOWER(department) = LOWER(p_dept);

  DBMS_OUTPUT.PUT_LINE('Bonus of ' || p_bonus_pct || '% applied to ' || p_dept || '
department.');
  COMMIT;
END;
```

Scenario 3: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
) IS
  v_balance NUMBER;
BEGIN
  SELECT balance INTO v_balance
  FROM accounts
  WHERE account_id = p_from_account;
  IF v_balance < p_amount THEN
    RAISE_APPLICATION_ERROR(-20001, 'Not enough balance in source account.');
  END IF;
  UPDATE accounts
  SET balance = balance - p_amount
```

```
  WHERE account_id = p_from_account;
  UPDATE accounts
  SET balance = balance + p_amount
  WHERE account_id = p_to_account;
  DBMS_OUTPUT.PUT_LINE(p_amount || ' transferred from Account ' || p_from_account ||
' to Account ' || p_to_account);
  COMMIT;
END;
```

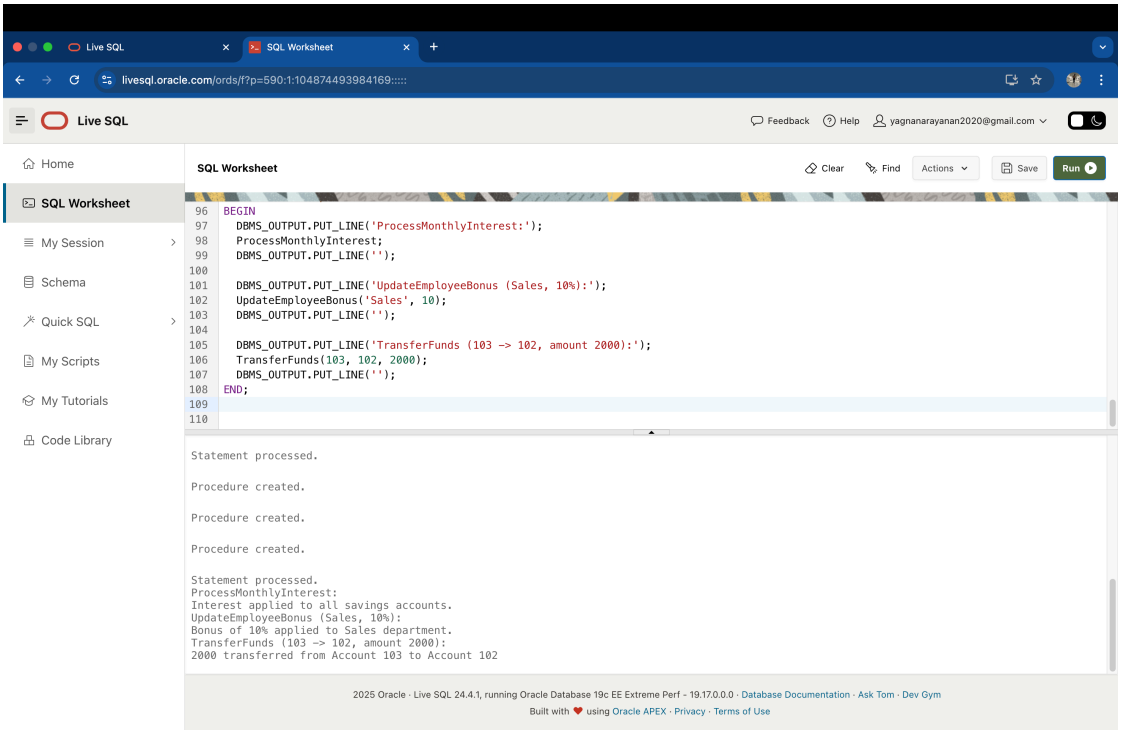
Execution of all procedures:

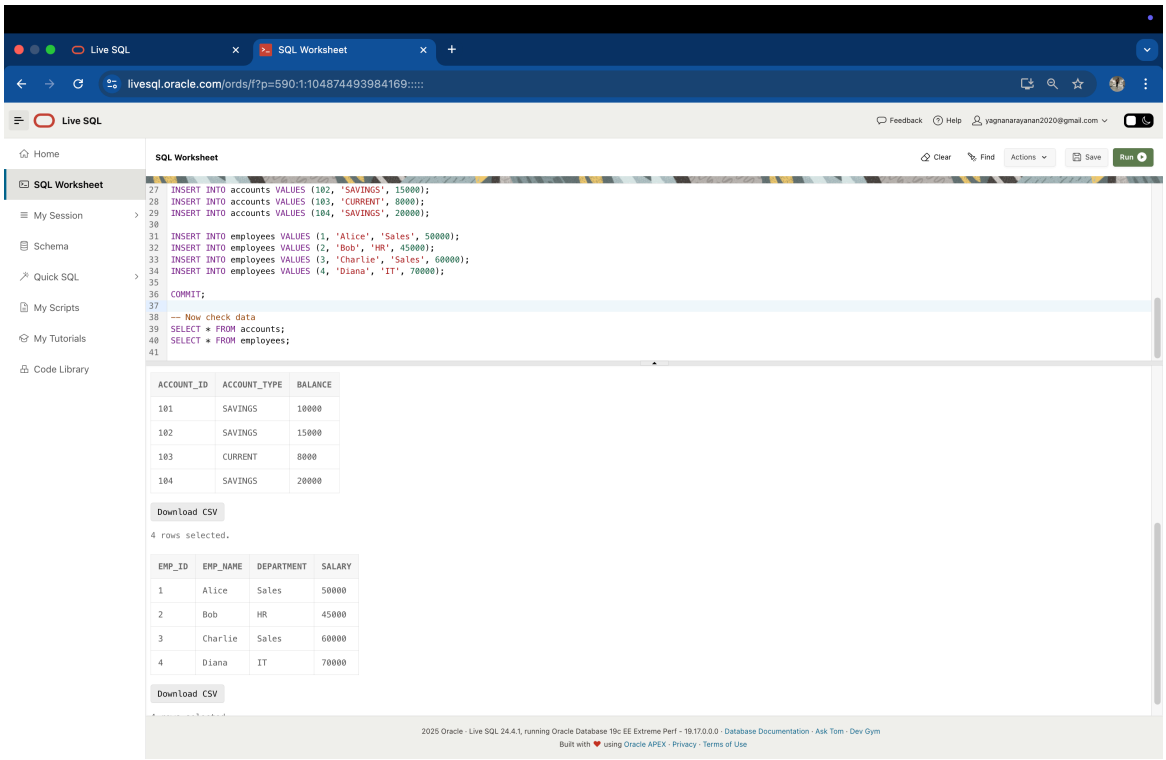
```
BEGIN
  DBMS_OUTPUT.PUT_LINE('ProcessMonthlyInterest:');
  ProcessMonthlyInterest;
  DBMS_OUTPUT.PUT_LINE('');

  DBMS_OUTPUT.PUT_LINE('UpdateEmployeeBonus (Sales, 10%):');
  UpdateEmployeeBonus('Sales', 10);
  DBMS_OUTPUT.PUT_LINE('');

  DBMS_OUTPUT.PUT_LINE('TransferFunds (103 -> 102, amount 2000):');
  TransferFunds(103, 102, 2000);
  DBMS_OUTPUT.PUT_LINE('');
END;
```

Output:


Live SQL
SQL Worksheet
livesql.oracle.com/ords/f?p=590:1:104874493984169::::
Feedback
Help
yagnanarayanan2020@gmail.com
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
Clear
Find
Actions
Save
Run
96 BEGIN
97 DBMS_OUTPUT.PUT_LINE('ProcessMonthlyInterest:');
98 ProcessMonthlyInterest;
99 DBMS_OUTPUT.PUT_LINE('');
100
101 DBMS_OUTPUT.PUT_LINE('UpdateEmployeeBonus (Sales, 10%):');
102 UpdateEmployeeBonus('Sales', 10);
103 DBMS_OUTPUT.PUT_LINE('');
104
105 DBMS_OUTPUT.PUT_LINE('TransferFunds (103 -> 102, amount 2000):');
106 TransferFunds(103, 102, 2000);
107 DBMS_OUTPUT.PUT_LINE('');
108 END;
Statement processed.
Procedure created.
Procedure created.
Procedure created.
Statement processed.
ProcessMonthlyInterest:
Interest applied to all savings accounts.
UpdateEmployeeBonus (Sales, 10%):
Bonus of 10% applied to Sales department.
TransferFunds (103 -> 102, amount 2000):
2000 transferred from Account 103 to Account 102
2025 Oracle · Live SQL 24.4.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with using Oracle APEX · Privacy · Terms of Use


Live SQL
SQL Worksheet
livesql.oracle.com/ords/f?p=590:1:104874493984169::::
Feedback
Help
yagnanarayanan2020@gmail.com
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
Clear
Find
Actions
Save
Run
27 INSERT INTO accounts VALUES (102, 'SAVINGS', 15000);
28 INSERT INTO accounts VALUES (103, 'CURRENT', 8000);
29 INSERT INTO accounts VALUES (104, 'SAVINGS', 20000);
30
31 INSERT INTO employees VALUES (1, 'Alice', 'Sales', 50000);
32 INSERT INTO employees VALUES (2, 'Bob', 'HR', 45000);
33 INSERT INTO employees VALUES (3, 'Charlie', 'Sales', 60000);
34 INSERT INTO employees VALUES (4, 'Diana', 'IT', 70000);
35
36 COMMIT;
37
38 -- Now check data
39 SELECT * FROM accounts;
40 SELECT * FROM employees;
41
ACCOUNT_ID ACCOUNT_TYPE BALANCE
101 SAVINGS 10000
102 SAVINGS 15000
103 CURRENT 8000
104 SAVINGS 20000
Download CSV
4 rows selected.
EMP_ID EMP_NAME DEPARTMENT SALARY
1 Alice Sales 50000
2 Bob HR 45000
3 Charlie Sales 60000
4 Diana IT 70000
Download CSV
2025 Oracle · Live SQL 24.4.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with using Oracle APEX · Privacy · Terms of Use