

WEEK 2

PL_SQL Exercise

Output snapshot:

Exercise 1: Control Structures

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.

Loan table after condition:

The screenshot shows the Live SQL interface. On the left, the Navigator pane shows 'My Schema' with tables 'CUSTOMERS' and 'LOANS'. The main editor displays a PL/SQL script that loops through all customers, checks if their age is greater than 60, and applies a 1% discount to their loan interest rates. The script is as follows:

```
32 BEGIN
33   FOR rec IN (
34     SELECT c.customer_id, l.loan_id, l.interest_rate
35     FROM customers c
36     JOIN loans l ON c.customer_id = l.customer_id
37     WHERE c.age > 60
38   ) LOOP
39     UPDATE loans
40     SET interest_rate = interest_rate - 1
41     WHERE loan_id = rec.loan_id;
42   END LOOP;
43   DBMS_OUTPUT.PUT_LINE('1% discount applied to loan ID ' || rec.loan_id || ' for customer ID ' || rec.customer_id);
44 END LOOP;
45 END;
```

Below the script, the 'Query result' tab shows the execution time as 0.072 seconds. The results are displayed in a table with columns LOAN_ID, CUSTOMER_ID, INTEREST_RATE, and DUE_DATE.

	LOAN_ID	CUSTOMER_ID	INTEREST_RATE	DUE_DATE
1	101	1	7.5	7/10/2025, 11:19:01
2	102	2	8	8/4/2025, 11:19:08
3	103	3	6.5	7/5/2025, 11:19:08
4	104	4	7	6/30/2025, 11:19:01

At the bottom of the interface, there are links for 'About Oracle', 'Contact Us', 'Legal Notices', 'Terms and Conditions', 'Your Privacy Rights', 'Delete Your Live SQL Account', and 'Cookie Preferences'. A copyright notice for 2014-2025 Oracle is also present.

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

Live SQL Worksheet

Navigator: My Schema, Tables, Search objects, CUSTOMERS, LOANS

```

47 BEGIN
48   FOR rec IN (
49     SELECT customer_id FROM customers WHERE balance > 10000
50   ) LOOP
51     UPDATE customers
52     SET is_vip = 'Y'
53     WHERE customer_id = rec.customer_id;
54   END LOOP;
55   DBMS_OUTPUT.PUT_LINE('Customer ID: ' || rec.customer_id || ' promoted to VIP. ');
56 END;
57
58 BEGIN
59   FOR rec IN (
60     SELECT c.name, l.loan_id, l.due_date
61     FROM customers c
62     JOIN loans l ON c.customer_id = l.customer_id
63   ) LOOP

```

Query result: Download, Execution time: 0.001 seconds

	CUSTOMER_ID	NAME	AGE	BALANCE	IS_VIP
1	1	Alice	65	12000	Y
2	2	Bob	45	8000	N
3	3	Charlie	70	20000	Y
4	4	David	30	9500	N

About Oracle | Contact Us | Legal Notices | Terms and Conditions | Your Privacy Rights | Delete Your Live SQL Account | Cookie Preferences

Copyright © 2014, 2025 Oracle and/or its affiliates All rights reserved.

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.

Live SQL Worksheet

Navigator: My Schema, Tables, Search objects, CUSTOMERS, LOANS

```

57 END;
58
59 BEGIN
60   FOR rec IN (
61     SELECT c.name, l.loan_id, l.due_date
62     FROM customers c
63     JOIN loans l ON c.customer_id = l.customer_id
64     WHERE l.due_date <= SYSDATE + 30
65   ) LOOP
66     DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || rec.loan_id || ' for ' || rec.name || ' is due on ' || TO_CHAR(rec.due_date, 'YYYY-MM-DD HH:MM:SS'));
67   END LOOP;
68 END;
69 select * from loans

```

Query result: Download, Execution time: 0.005 seconds

	LOAN_ID	CUSTOMER_ID	INTEREST_RATE	DUE_DATE
1	101	1	6.5	7/10/2025, 11:19:08 AM
2	102	2	8	8/4/2025, 11:19:08 AM
3	103	3	5.5	7/5/2025, 11:19:08 AM
4	104	4	7	6/30/2025, 11:19:08 AM

DBMS output:

Query result	Script output	DBMS output	Explain Plan	SQL history
<div><div><div></div><div></div></div></div>				
1% discount applied to loan ID 101 for customer ID 1 1% discount applied to loan ID 103 for customer ID 3				
Customer ID 1 promoted to VIP. Customer ID 3 promoted to VIP.				
Customer ID 1 promoted to VIP. Customer ID 3 promoted to VIP.				
Customer ID 1 promoted to VIP. Customer ID 3 promoted to VIP.				

Query result	Script output	DBMS output	Explain Plan	SQL history
<div><div><div></div><div></div></div></div>				
1% discount applied to loan ID 101 for customer ID 1 1% discount applied to loan ID 103 for customer ID 3				<div><div></div><div></div></div>
Customer ID 1 promoted to VIP. Customer ID 3 promoted to VIP.				<div><div></div><div></div></div>
Reminder: Loan ID 101 for Alice is due on 10-Jul-2025 Reminder: Loan ID 103 for Charlie is due on 05-Jul-2025 Reminder: Loan ID 104 for David is due on 30-Jun-2025				<div><div></div><div></div></div>

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.

The screenshot shows the Live SQL interface with a PL/SQL script in the editor. The script iterates through all savings accounts and updates their balances by 1%. The output shows the results of the script execution.

```
82 BEGIN
83   FOR acc IN (SELECT account_id, balance FROM savings_accounts) LOOP
84     UPDATE savings_accounts
85       SET balance = balance + (balance * 0.01)
86       WHERE account_id = acc.account_id;
87   END LOOP;
88   DBMS_OUTPUT.PUT_LINE('Monthly interest applied to all savings accounts.');
```

Query result

	ACCOUNT_ID	CUSTOMER_ID	BALANCE
1	201	1	10100
2	202	2	8585
3	203	3	12120

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.

The screenshot shows the Live SQL interface with a PL/SQL script in the editor. The script updates the salary of employees in a given department by adding a bonus percentage. The output shows the results of the script execution.

```
109 UPDATE employees
110 SET salary = salary + (salary * p_bonus_pct / 100)
111 WHERE dept_id = p_dept_id;
112 DBMS_OUTPUT.PUT_LINE('Bonus applied to department ' || p_dept_id);
113 END;
114 call UpdateEmployeeBonus(10,15)
115 select * from employees
116 CREATE OR REPLACE PROCEDURE TransferFunds (
117   p_from_account IN NUMBER,
118   p_to_account IN NUMBER,
119   p_amount IN NUMBER
120 ) IS
121   v_balance NUMBER;
122 BEGIN
123   -- Check balance of source account
124
```

Query result

	EMP_ID	ENAME	DEPT_ID	SALARY
1	301	Alice	10	57500
2	302	Bob	20	45000
3	303	Charlie	10	69000

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.

The screenshot shows the Live SQL interface with a SQL worksheet containing the following code:

```
147 -- Add to destination
148 UPDATE bank_accounts
149 SET balance = balance + p_amount
150 WHERE account_id = p_to_account;
151
152 DBMS_OUTPUT.PUT_LINE('Transferred ' || p_amount || ' from ' || p_from_account || ' to ' || p_to_account);
153 END IF;
154 END;
155
156 call TransferFunds(401, 402, 2000);
157 select * from savings_accounts
```

The **Query result** tab is active, displaying a table with the following data:

	ACCOUNT_ID	CUSTOMER_ID	BALANCE
1	201	1	10100
2	202	2	8585
3	203	3	12120

DBMS output:

The screenshot shows the Live SQL interface with the same SQL worksheet as above. The **DBMS output** tab is active, displaying the following output:

```
Monthly interest applied to all savings accounts.
Bonus applied to department 10
Transferred 2000 from 401 to 402
```