

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.

Output:

The screenshot displays the Oracle Live SQL web interface. On the left, the 'Navigator' pane shows a schema named 'My Schema' with a table 'CUSTOMERS' containing columns 'CUSTOMER_ID', 'NAME', 'DOB', and 'LOAN_INTEREST'. The main editor area, titled '[SQL Worksheet]', contains a PL/SQL block with the following code:

```
26
27
28 LOOP
29   DECLARE
30     v_age NUMBER;
31   BEGIN
32     v_age := TRUNC(MONTHS_BETWEEN(SYSDATE, cust_rec.DOB) / 12);
33     IF v_age > 60 THEN
34       UPDATE CUSTOMERS
35       SET LOAN_INTEREST = cust_rec.LOAN_INTEREST * .99
36       WHERE CUSTOMER_ID = cust_rec.CUSTOMER_ID;
37     END IF;
38     DBMS_OUTPUT.PUT_LINE('Discount applied to Customer ID: ' || cust_rec.CUSTOMER_ID);
39   END;
40 END LOOP;
41
42 COMMIT;
```

Below the code editor, the 'Script output' tab is active, showing the execution results:

```
SQL> BEGIN
FOR cust_rec IN (
  SELECT CUSTOMER_ID, DOB, LOAN_INTEREST
  FROM CUSTOMERS...
Show more...

Discount applied to Customer ID: 1
Discount applied to Customer ID: 3

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.003
```

The footer of the interface includes links for 'About Oracle', 'Contact Us', 'Legal Notices', 'Terms and Conditions', 'Your Privacy Rights', 'Delete Your Live SQL Account', and 'Cookie Preferences', along with the copyright notice 'Copyright © 2014, 2025 Oracle and/or its affiliates. All rights reserved.' and the version number 'v31.1'.

Live SQL

Worksheet

Library

23at

Return to Live SQL Classic

Help and Feedback

Sign Out

Navigator

Files

My Schema

Tables

Search objects

CUSTOMERS

CUSTOMER_ID

NAME

DOB

LOAN_INTEREST

[SQL Worksheet]*

26

}

27

LOOP

28

DECLARE

29

v_age NUMBER;

30

BEGIN

31

v_age := TRUNC(MONTHS_BETWEEN(SYSDATE, cust_rec.DOB) / 12);

32

IF v_age > 60 THEN

33

UPDATE CUSTOMERS

34

SET LOAN_INTEREST = cust_rec.LOAN_INTEREST - 1

35

WHERE CUSTOMER_ID = cust_rec.CUSTOMER_ID;

36

DBMS_OUTPUT.PUT_LINE('Discount applied to Customer ID: ' || cust_rec.CUSTOMER_ID);

37

END IF;

38

END;

39

END LOOP;

40

41

42

COMMIT;

43

Query result

Script output

DBMS output

Explain Plan

SQL history

Discount applied to Customer ID: 1

Discount applied to Customer ID: 3

Discount applied to Customer ID: 1

Discount applied to Customer ID: 3

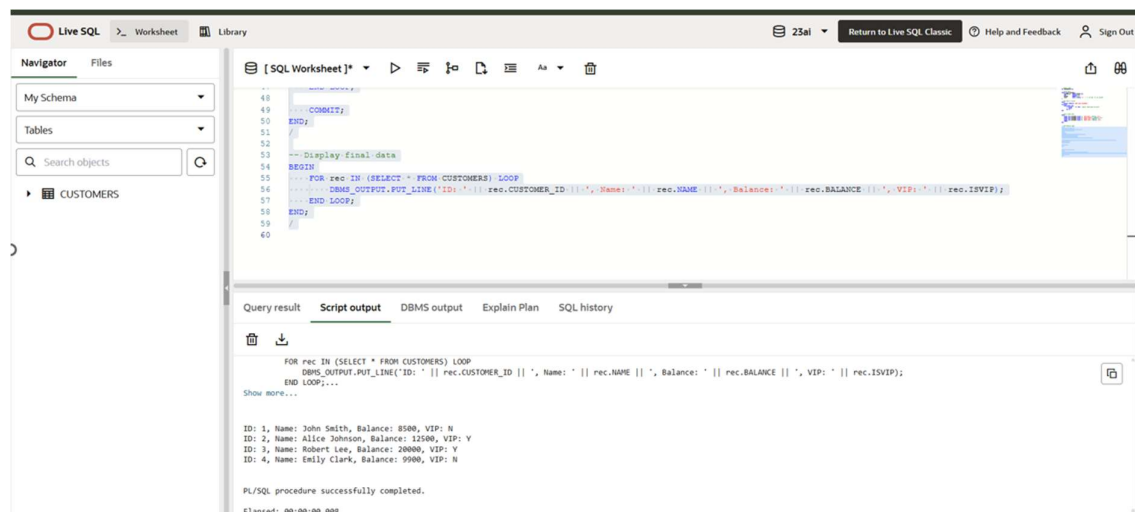
Scenario 2:

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.

Output:



```
48  
49  
50 COMMIT;  
51  
52  
53 -- Display final data  
54 BEGIN  
55   FOR rec IN (SELECT * FROM CUSTOMERS) LOOP  
56     DBMS_OUTPUT.PUT_LINE('ID: ' || rec.CUSTOMER_ID || ', Name: ' || rec.NAME || ', Balance: ' || rec.BALANCE || ', VIP: ' || rec.ISVIP);  
57   END LOOP;  
58 END;  
59  
60
```

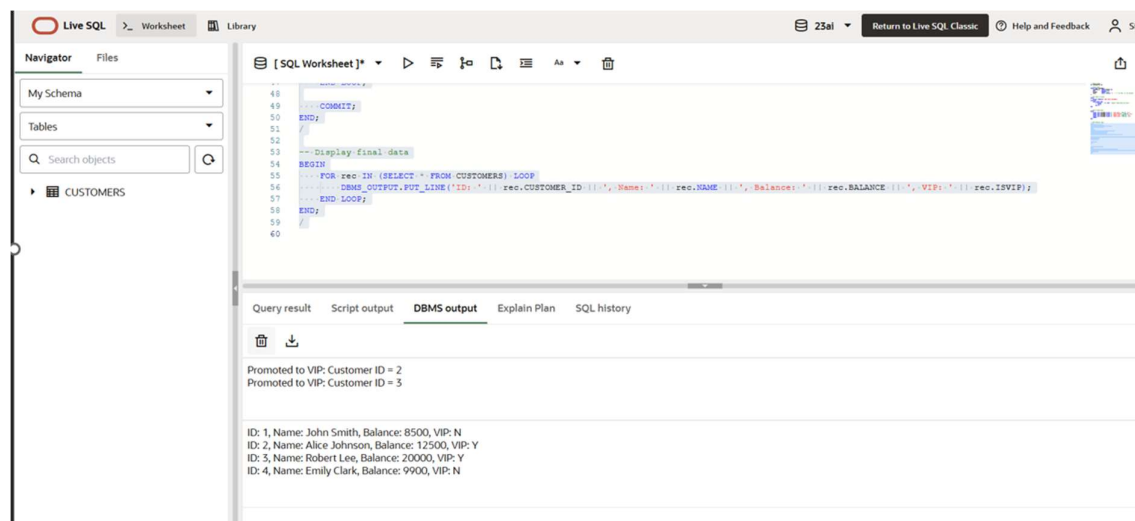
Query result Script output DBMS output Explain Plan SQL history

FOR rec IN (SELECT * FROM CUSTOMERS) LOOP
 DBMS_OUTPUT.PUT_LINE('ID: ' || rec.CUSTOMER_ID || ', Name: ' || rec.NAME || ', Balance: ' || rec.BALANCE || ', VIP: ' || rec.ISVIP);
END LOOP;...

Show more...

ID: 1, Name: John Smith, Balance: 8500, VIP: N
ID: 2, Name: Alice Johnson, Balance: 12500, VIP: Y
ID: 3, Name: Robert Lee, Balance: 20000, VIP: Y
ID: 4, Name: Emily Clark, Balance: 9900, VIP: N

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.008



```
48  
49  
50 COMMIT;  
51  
52  
53 -- Display final data  
54 BEGIN  
55   FOR rec IN (SELECT * FROM CUSTOMERS) LOOP  
56     DBMS_OUTPUT.PUT_LINE('ID: ' || rec.CUSTOMER_ID || ', Name: ' || rec.NAME || ', Balance: ' || rec.BALANCE || ', VIP: ' || rec.ISVIP);  
57   END LOOP;  
58 END;  
59  
60
```

Query result Script output DBMS output Explain Plan SQL history

Promoted to VIP: Customer ID = 2
Promoted to VIP: Customer ID = 3

ID: 1, Name: John Smith, Balance: 8500, VIP: N
ID: 2, Name: Alice Johnson, Balance: 12500, VIP: Y
ID: 3, Name: Robert Lee, Balance: 20000, VIP: Y
ID: 4, Name: Emily Clark, Balance: 9900, VIP: N

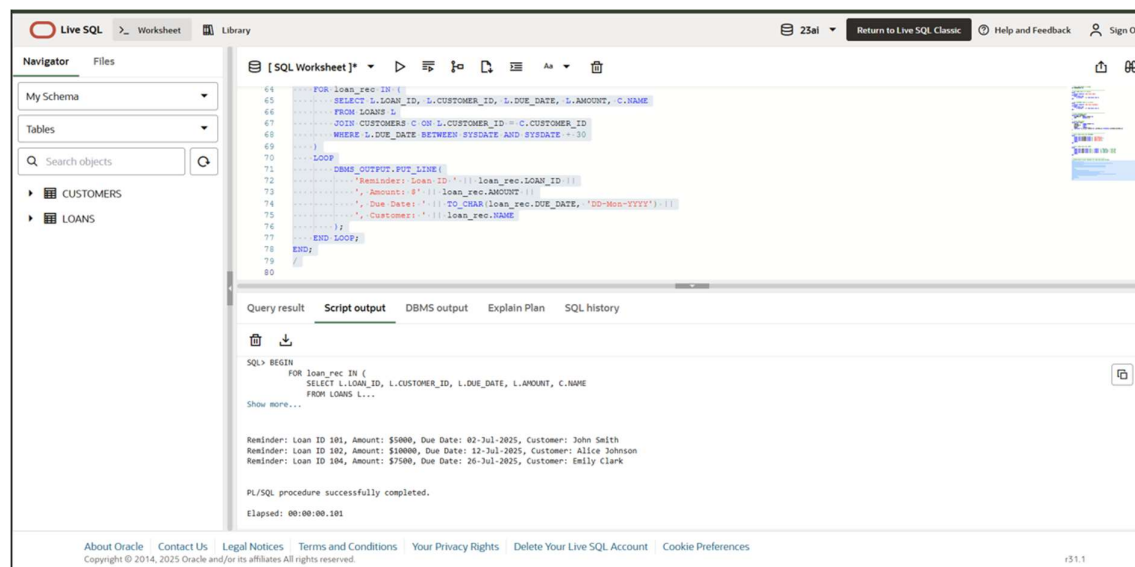
Scenario 3:

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.

Output:



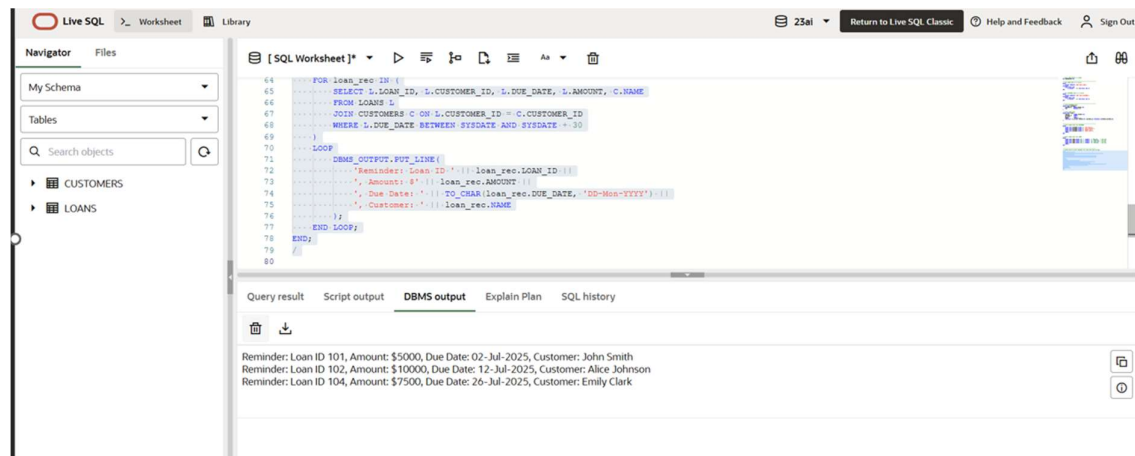
```
64 FOR loan_rec IN (
65     SELECT L.LOAN_ID, L.CUSTOMER_ID, L.DUE_DATE, L.AMOUNT, C.NAME
66     FROM LOANS L
67     JOIN CUSTOMERS C ON L.CUSTOMER_ID = C.CUSTOMER_ID
68     WHERE L.DUE_DATE BETWEEN SYSDATE AND SYSDATE + 30
69 )
70 LOOP
71     DBMS_OUTPUT.PUT_LINE(
72         'Reminder: Loan ID: ' || loan_rec.LOAN_ID ||
73         ', Amount: $' || loan_rec.AMOUNT ||
74         ', Due Date: ' || TO_CHAR(loan_rec.DUE_DATE, 'DD-Mon-YYYY') ||
75         ', Customer: ' || loan_rec.NAME
76     );
77 END LOOP;
78 END;
```

SQL> BEGIN
FOR loan_rec IN (
SELECT L.LOAN_ID, L.CUSTOMER_ID, L.DUE_DATE, L.AMOUNT, C.NAME
FROM LOANS L...

Show more...

Reminder: Loan ID 101, Amount: \$5000, Due Date: 02-Jul-2025, Customer: John Smith
Reminder: Loan ID 102, Amount: \$10000, Due Date: 12-Jul-2025, Customer: Alice Johnson
Reminder: Loan ID 104, Amount: \$7500, Due Date: 26-Jul-2025, Customer: Emily Clark

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.101



```
64 FOR loan_rec IN (
65     SELECT L.LOAN_ID, L.CUSTOMER_ID, L.DUE_DATE, L.AMOUNT, C.NAME
66     FROM LOANS L
67     JOIN CUSTOMERS C ON L.CUSTOMER_ID = C.CUSTOMER_ID
68     WHERE L.DUE_DATE BETWEEN SYSDATE AND SYSDATE + 30
69 )
70 LOOP
71     DBMS_OUTPUT.PUT_LINE(
72         'Reminder: Loan ID: ' || loan_rec.LOAN_ID ||
73         ', Amount: $' || loan_rec.AMOUNT ||
74         ', Due Date: ' || TO_CHAR(loan_rec.DUE_DATE, 'DD-Mon-YYYY') ||
75         ', Customer: ' || loan_rec.NAME
76     );
77 END LOOP;
78 END;
```

Reminder: Loan ID 101, Amount: \$5000, Due Date: 02-Jul-2025, Customer: John Smith
Reminder: Loan ID 102, Amount: \$10000, Due Date: 12-Jul-2025, Customer: Alice Johnson
Reminder: Loan ID 104, Amount: \$7500, Due Date: 26-Jul-2025, Customer: Emily Clark

Exercise 2: Error Handling

Scenario 1:

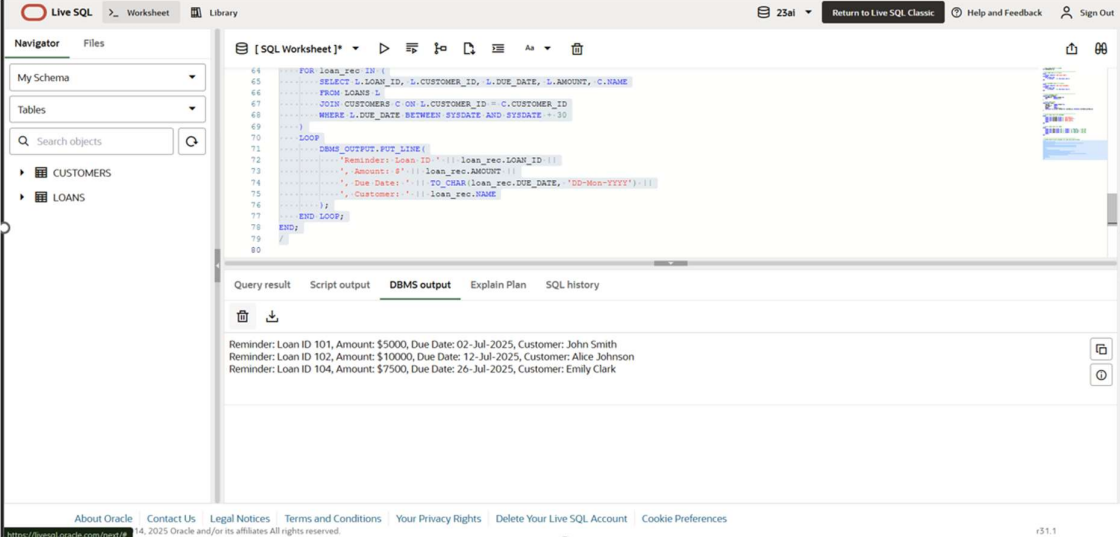
Handle exceptions during fund transfers between accounts.

Question:

Write a stored procedure SafeTransferFunds that transfers funds between two accounts.

Ensure that if any error occurs (e.g., insufficient funds), an appropriate error message is logged and the transaction is rolled back.

Output:



The screenshot displays the Oracle Live SQL interface. On the left, the 'Navigator' pane shows a schema named 'My Schema' with tables 'CUSTOMERS' and 'LOANS'. The main editor shows a PL/SQL procedure named 'SafeTransferFunds' with the following code:

```
64 FOR loan_rec IN (
65     SELECT l.LOAN_ID, l.CUSTOMER_ID, l.DUE_DATE, l.AMOUNT, c.NAME
66     FROM LOANS l
67     JOIN CUSTOMERS c ON l.CUSTOMER_ID = c.CUSTOMER_ID
68     WHERE l.DUE_DATE BETWEEN SYSDATE AND SYSDATE + 30
69 )
70 LOOP
71     DBMS_OUTPUT.PUT_LINE(
72         'Reminder: Loan ID ' || loan_rec.LOAN_ID ||
73         ', Amount: $' || loan_rec.AMOUNT ||
74         ', Due Date: ' || TO_CHAR(loan_rec.DUE_DATE, 'DD-Mon-YYYY') ||
75         ', Customer: ' || loan_rec.NAME
76     );
77 END LOOP;
78
79
80
```

Below the code editor, the 'DBMS output' tab is selected, showing the following output:

```
Reminder: Loan ID 101, Amount: $5000, Due Date: 02-Jul-2025, Customer: John Smith
Reminder: Loan ID 102, Amount: $10000, Due Date: 12-Jul-2025, Customer: Alice Johnson
Reminder: Loan ID 104, Amount: $7500, Due Date: 26-Jul-2025, Customer: Emily Clark
```

The footer of the interface includes links for 'About Oracle', 'Contact Us', 'Legal Notices', 'Terms and Conditions', 'Your Privacy Rights', 'Delete Your Live SQL Account', and 'Cookie Preferences'. The URL 'https://livesql.oracle.com/html/' is visible on the left, and 'r31.1' is on the right.

Live SQL

Worksheet

Library

25s

Return to Live SQL Classic

Help and Feedback

Sign Out

NavigatorFiles

My Schema

Tables

Search objects

ACCOUNTS

CUSTOMERS

LOANS

[SQL Worksheet]

```
46 BEGIN
47   UPDATE ACCOUNTS
48   SET BALANCE = BALANCE + (BALANCE * 0.01)
49   WHERE ACCOUNT_TYPE = 'SAVINGS';
50
51   DBMS_OUTPUT.PUT_LINE('Monthly interest applied to all savings accounts.');
```

Query result

Script output

DBMS output

Explain Plan

SQL history

Monthly interest applied to all savings accounts.

ID: 1, Name: John Smith, Type: SAVINGS, Balance: 10100
ID: 2, Name: Alice Johnson, Type: CURRENT, Balance: 20000
ID: 3, Name: Robert Lee, Type: SAVINGS, Balance: 5050
ID: 4, Name: Emily Clark, Type: SAVINGS, Balance: 15150

About Oracle

Contact Us

Legal Notices

Terms and Conditions

Your Privacy Rights

Delete Your Live SQL Account

Cookie Preferences

r31.1

Scenario 2:

Manage errors when updating employee salaries.

Question:

Write a stored procedure UpdateSalary that increases the salary of an employee by a given percentage. If the employee ID does not exist, handle the exception and log an error message.

Output:

The screenshot shows the Live SQL interface with a PL/SQL script being executed. The script defines a procedure `UpdateEmployeeBonus` that takes a department name and a bonus percentage as input. It uses a `FOR` loop to iterate over all employees in the specified department and updates their salary by the given percentage. The output shows the results of the update for the SALES department.

```
42 SET SALARY = SALARY + (SALARY * bonus_pct / 100)
43 WHERE UPPER(DEPARTMENT) = UPPER(dept_name);
44 DBMS_OUTPUT.PUT_LINE('Bonus of ' || bonus_pct || '% applied to department: ' || dept_name);
45 END;
46 /
47
48
49 BEGIN
50   UpdateEmployeeBonus('SALES', 10);
51 END;
52 /
53
54 BEGIN
55   FOR emp IN (SELECT * FROM EMPLOYEES) LOOP
56     DBMS_OUTPUT.PUT_LINE('ID: ' || emp.EMP_ID ||
57                           ' Name: ' || emp.NAME ||
58                           ' Dept: ' || emp.DEPARTMENT ||
59                           ' Salary: ' || emp.SALARY);
60   END LOOP;
61 END;
62 /
63
```

Query result Script output DBMS output Explain Plan SQL history

ID: 1, Name: Alice, Dept: SALES, Salary: 55000
ID: 2, Name: Bob, Dept: HR, Salary: 45000
ID: 3, Name: Charlie, Dept: SALES, Salary: 60500
ID: 4, Name: Diana, Dept: IT, Salary: 60000
ID: 5, Name: Ethan, Dept: SALES, Salary: 57200

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.011

The screenshot shows the Live SQL interface with the same PL/SQL script being executed. The output shows the results of the update for the SALES department, including the bonus percentage applied and the updated salaries for all employees in the department.

```
42 SET SALARY = SALARY + (SALARY * bonus_pct / 100)
43 WHERE UPPER(DEPARTMENT) = UPPER(dept_name);
44 DBMS_OUTPUT.PUT_LINE('Bonus of ' || bonus_pct || '% applied to department: ' || dept_name);
45 END;
46 /
47
48
49 BEGIN
50   UpdateEmployeeBonus('SALES', 10);
51 END;
52 /
53
54 BEGIN
55   FOR emp IN (SELECT * FROM EMPLOYEES) LOOP
56     DBMS_OUTPUT.PUT_LINE('ID: ' || emp.EMP_ID ||
57                           ' Name: ' || emp.NAME ||
58                           ' Dept: ' || emp.DEPARTMENT ||
59                           ' Salary: ' || emp.SALARY);
60   END LOOP;
61 END;
62 /
63
```

Query result Script output DBMS output Explain Plan SQL history

Bonus of 10% applied to department: SALES

ID: 1, Name: Alice, Dept: SALES, Salary: 55000
ID: 2, Name: Bob, Dept: HR, Salary: 45000
ID: 3, Name: Charlie, Dept: SALES, Salary: 60500
ID: 4, Name: Diana, Dept: IT, Salary: 60000
ID: 5, Name: Ethan, Dept: SALES, Salary: 57200

Scenario 3:

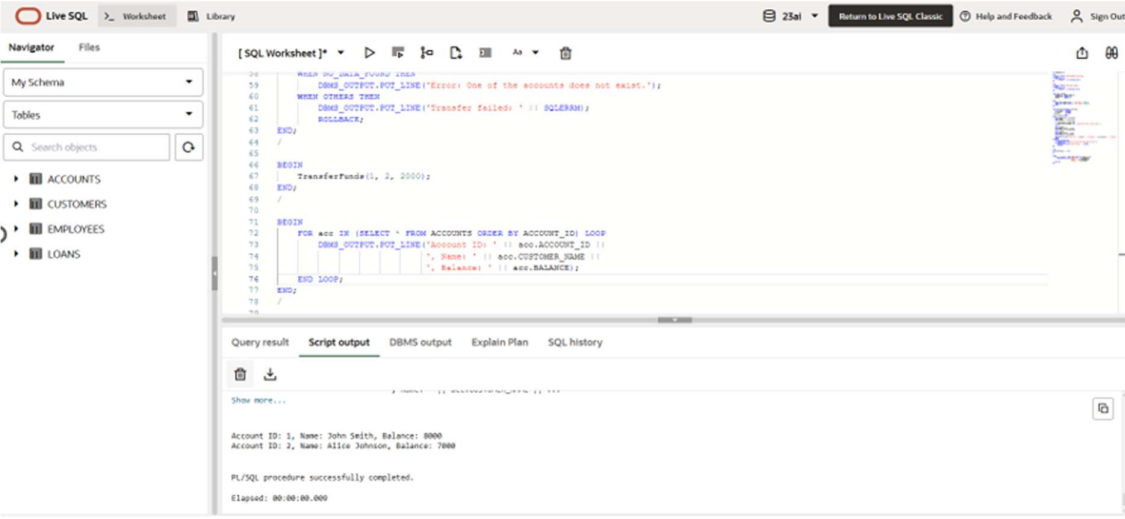
Ensure data integrity when adding a new customer.

Question:

Write a stored procedure AddNewCustomer that inserts a new customer into the Customers table.

If a customer with the same ID already exists, handle the exception by logging an error and preventing the insertion.

Output:



The screenshot displays the Oracle Live SQL web interface. On the left, a 'Navigator' pane shows a schema named 'My Schema' with tables 'ACCOUNTS', 'CUSTOMERS', 'EMPLOYEES', and 'LOANS'. The main editor area, titled '[SQL Worksheet]', contains a PL/SQL procedure named 'AddNewCustomer'. The procedure includes a loop to insert new customers, with an exception handler that logs an error and rolls back the transaction if a duplicate ID is encountered. Below the editor, the 'Script output' tab is active, showing the results of the procedure execution. The output lists two accounts: 'Account ID: 1, Name: John Smith, Balance: 8000' and 'Account ID: 2, Name: Alice Johnson, Balance: 7000'. A message at the bottom states 'PL/SQL procedure successfully completed.' and 'Elapsed: 00:00:00.000'.

```
1000 CREATE OR REPLACE PROCEDURE AddNewCustomer
1001   (p_customer_id IN NUMBER,
1002    p_customer_name IN VARCHAR2,
1003    p_customer_balance IN NUMBER)
1004 AS
1005   BEGIN
1006     IF (SELECT COUNT(*) FROM CUSTOMERS WHERE customer_id = p_customer_id) > 0
1007     THEN
1008       DBMS_OUTPUT.PUT_LINE('Error: One of the accounts does not exist.');
```

Account ID: 1, Name: John Smith, Balance: 8000
Account ID: 2, Name: Alice Johnson, Balance: 7000

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.000

Live SQL

Worksheet

Library

25ai

Return to Live SQL Classic

Help and Feedback

Sign Out

Navigator

Files

My Schema

Tables

Search objects

ACCOUNTS

CUSTOMERS

EMPLOYEES

LOANS

[SQL Worksheet]*

```
58 BEGIN
59   DBMS_OUTPUT.PUT_LINE('Error: One of the accounts does not exist.');
```

Query result

Script output

DBMS output

Explain Plan

SQL history

Transfer of \$2000 from Account 1 to Account 2 successful.

Account ID: 1, Name: John Smith, Balance: 8000
Account ID: 2, Name: Alice Johnson, Balance: 7000

About Oracle

Contact Us

Legal Notices

Terms and Conditions

Your Privacy Rights

Delete Your Live SQL Account

Cookie Preferences

14. 2025 Oracle and/or its affiliates. All rights reserved.

v11.1