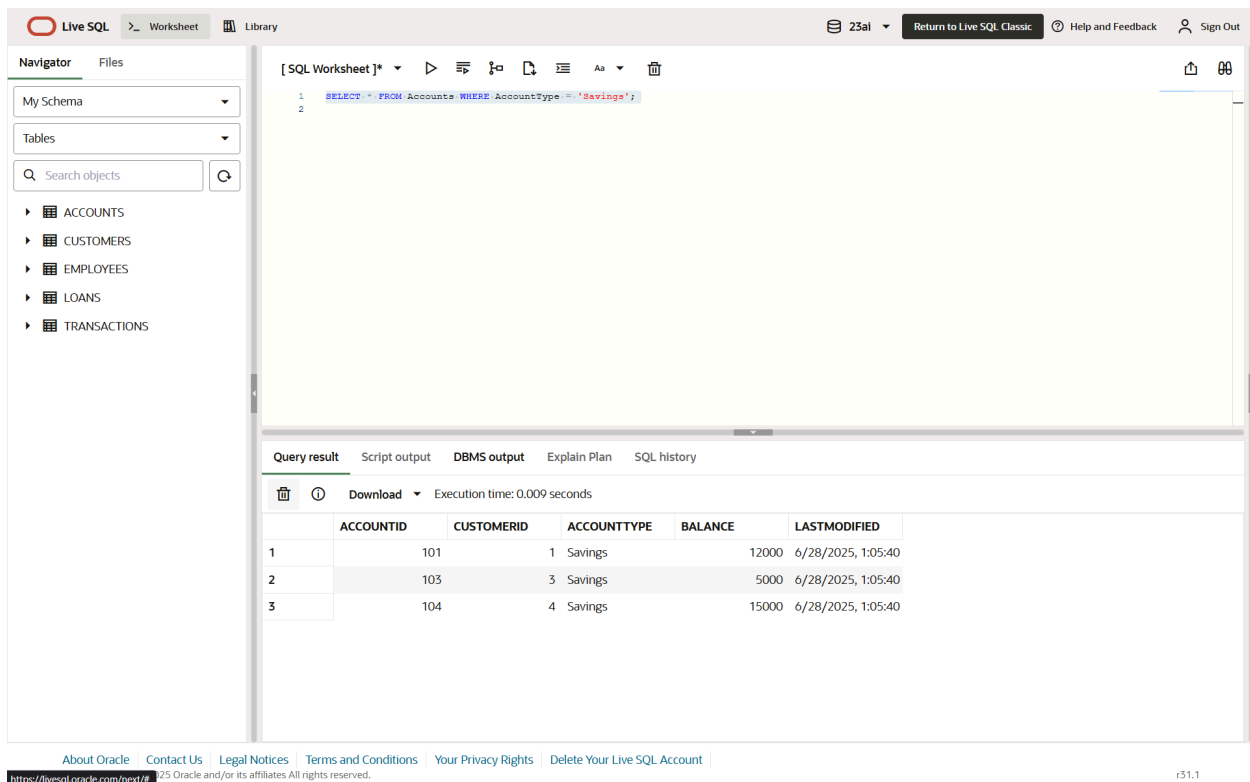


EXERCISE 3 Stored Procedures

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

- before



The screenshot shows the Live SQL interface. On the left, the Navigator pane shows a schema named 'My Schema' with tables: ACCOUNTS, CUSTOMERS, EMPLOYEES, LOANS, and TRANSACTIONS. The main editor shows a SQL query: `SELECT * FROM Accounts WHERE AccountType = 'Savings';`. Below the editor, the 'Query result' tab is active, displaying a table with 5 columns: ACCOUNTID, CUSTOMERID, ACCOUNTTYPE, BALANCE, and LASTMODIFIED. The table contains 3 rows of data. At the bottom, there are links for 'About Oracle', 'Contact Us', 'Legal Notices', 'Terms and Conditions', 'Your Privacy Rights', and 'Delete Your Live SQL Account'. The footer also includes a URL and a version number 'r31.1'.

	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	LASTMODIFIED
1	101	1	Savings	12000	6/28/2025, 1:05:40
2	103	3	Savings	5000	6/28/2025, 1:05:40
3	104	4	Savings	15000	6/28/2025, 1:05:40

- Stored Procedure: `ProcessMonthlyInterest`

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
  FOR acc IN (
    SELECT AccountID, Balance
    FROM Accounts
    WHERE AccountType = 'Savings'
  ) LOOP
    UPDATE Accounts
```

```

SET Balance = Balance + (Balance * 0.01),
LastModified = SYSDATE
WHERE AccountID = acc.AccountID;
END LOOP;

```

```

COMMIT;
END;

```

The screenshot displays the Oracle Live SQL web interface. On the left, a 'Navigator' pane shows a tree view of database objects under 'My Schema', including ACCOUNTS, CUSTOMERS, EMPLOYEES, LOANS, and TRANSACTIONS. The main area is a 'Worksheet' titled '[SQL Worksheet]*' containing a PL/SQL procedure named 'ProcessMonthlyInterest'. The procedure iterates over accounts, updating their balance by 0.01% and setting the last modified date to SYSDATE. Below the worksheet, the 'Script output' tab is active, showing the execution of the procedure, which compiled successfully. The output also indicates the elapsed time as 00:00:00.003. At the bottom of the page, there are links for 'About Oracle', 'Contact Us', 'Legal Notices', 'Terms and Conditions', 'Your Privacy Rights', and 'Delete Your Live SQL Account', along with a copyright notice for 2014-2025 Oracle and the version number 'r31.1'.

```

1 CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
2 BEGIN
3   FOR acc IN (
4     SELECT AccountID, Balance
5     FROM Accounts
6     WHERE AccountType = 'Savings'
7   ) LOOP
8     UPDATE Accounts
9     SET Balance = Balance + (Balance * 0.01),
10        LastModified = SYSDATE
11     WHERE AccountID = acc.AccountID;
12   END LOOP;
13   COMMIT;
14 END;
15 /
16
17

```

Query result Script output DBMS output Explain Plan SQL history

SQL> CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
 BEGIN
 FOR acc IN (
 SELECT AccountID, Balance...
 Show more...

Procedure PROCESSMONTHLYINTEREST compiled
 Elapsed: 00:00:00.003

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your Live SQL Account](#)
 Copyright © 2014, 2025 Oracle and/or its affiliates All rights reserved. r31.1

- Output

The screenshot displays the Oracle Live SQL web interface. On the left, the 'Navigator' pane shows a schema named 'My Schema' with a list of tables: ACCOUNTS, CUSTOMERS, EMPLOYEES, LOANS, and TRANSACTIONS. The main editor area, titled '[SQL Worksheet]', contains a PL/SQL procedure named 'ProcessMonthlyInterest'. The procedure is defined as follows:

```
1 BEGIN
2   ProcessMonthlyInterest;
3 END;
4
5
```

Below the editor, the 'Script output' tab is active, showing the execution results. The output includes the SQL command to create or replace the procedure, the compilation status, and the execution time.

```
SQL> CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
  FOR acc IN (
    SELECT AccountID, Balance...
  Show more...

Procedure PROCESSMONTHLYINTEREST compiled
Elapsed: 00:00:00.003

SQL> BEGIN
  ProcessMonthlyInterest;
END;

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.004
```

The footer of the interface contains links for 'About Oracle', 'Contact Us', 'Legal Notices', 'Terms and Conditions', 'Your Privacy Rights', and 'Delete Your Live SQL Account'. The copyright notice states 'Copyright © 2014, 2025 Oracle and/or its affiliates All rights reserved.' and the version is 'r31.1'.

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

- Before

Live SQL Worksheet Library 23ai Return to Live SQL Classic Help and Feedback Sign Out

Navigator Files

My Schema

Tables

Search objects

ACCOUNTS
CUSTOMERS
EMPLOYEES
LOANS
TRANSACTIONS

```

1 SELECT * FROM Employees WHERE Department = 'IT';
2

```

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.01 seconds

	EMPLOYEEID	NAME	POSITION	SALARY	DEPARTMENT	HIREDATE
1	402	Bob Brown	Developer	60000	IT	3/20/2017, 12:00:00
2	403	Neha Sharma	Developer	50000	IT	1/10/2020, 12:00:00

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your Live SQL Account](#)
 Copyright © 2014, 2025 Oracle and/or its affiliates All rights reserved.

r31.1

- **Stored Procedure:** **UpdateEmployeeBonus**

```

sql
CopyEdit
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
  dept_name IN VARCHAR2,
  bonus_percent IN NUMBER
) IS
BEGIN
  UPDATE Employees
  SET Salary = Salary + (Salary * bonus_percent / 100)
  WHERE Department = dept_name;

  COMMIT;
END;

```

Live SQL Worksheet

My Schema

Tables

Search objects

ACCOUNTS

CUSTOMERS

EMPLOYEES

LOANS

TRANSACTIONS

```

1
2 CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
3   dept_name IN VARCHAR2,
4   bonus_percent IN NUMBER
5 ) IS
6 BEGIN
7   UPDATE Employees
8   SET Salary = Salary * (Salary * bonus_percent / 100)
9   WHERE Department = dept_name;
10
11 COMMIT;
12
13 END;

```

Query result Script output DBMS output Explain Plan SQL history

SQL> CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
 dept_name IN VARCHAR2,
 bonus_percent IN NUMBER
) IS...
Show more...

Procedure UPDATEEMPLOYEEBONUS compiled

Elapsed: 00:00:00.003

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

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

r31.1

• output

Live SQL Worksheet

My Schema

Tables

Search objects

ACCOUNTS

CUSTOMERS

EMPLOYEES

LOANS

TRANSACTIONS

```

1 SELECT * FROM Employees WHERE Department = 'IT';
2

```

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.007 seconds

	EMPLOYEEID	NAME	POSITION	SALARY	DEPARTMENT	HIREDATE
1	402	Bob Brown	Developer	66000	IT	3/20/2017, 12:00:00
2	403	Neha Sharma	Developer	55000	IT	1/10/2020, 12:00:00

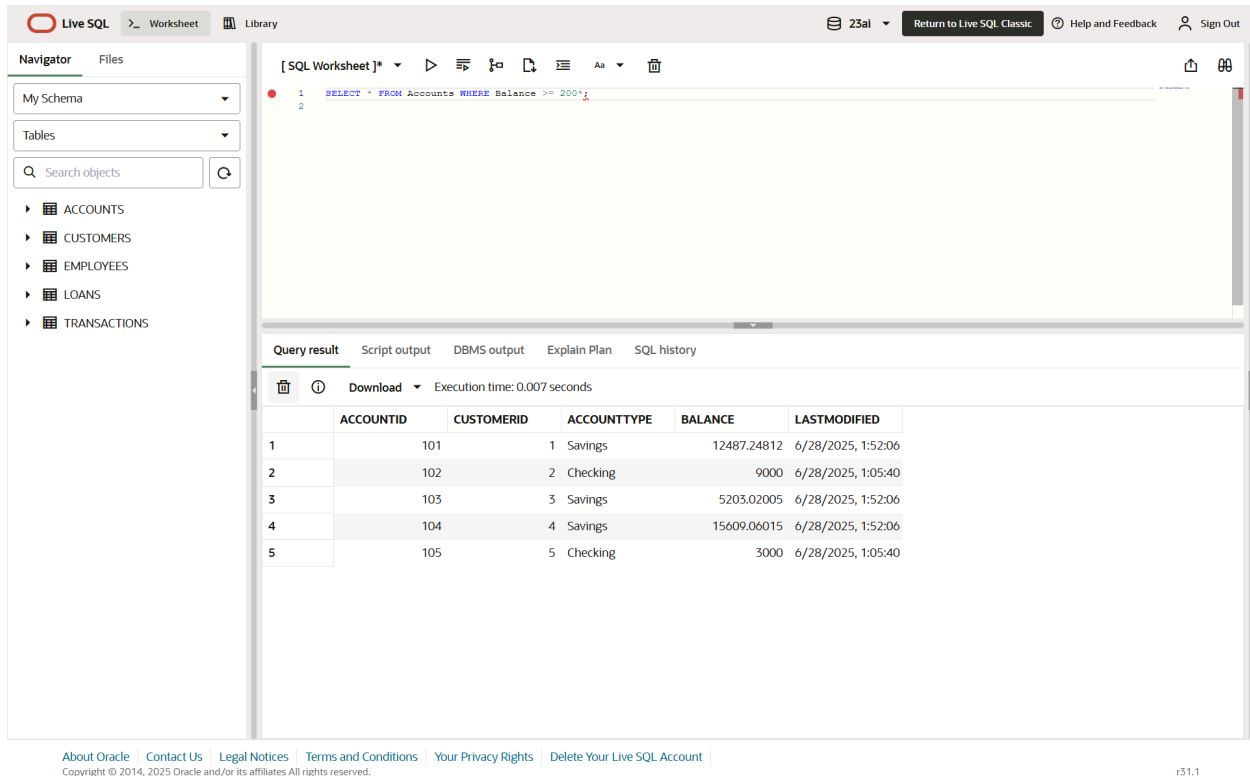
About Oracle Contact Us Legal Notices Terms and Conditions Your Privacy Rights Delete Your Live SQL Account

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

r31.1

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

- Before



The screenshot shows the Live SQL interface. On the left is a Navigator pane with a tree view containing 'My Schema', 'Tables', and a search bar. The main area displays a SQL query: `SELECT * FROM Accounts WHERE Balance >= 200;`. Below the query editor, the 'Query result' tab is active, showing a table with 5 rows. The table has columns: ACCOUNTID, CUSTOMERID, ACCOUNTTYPE, BALANCE, and LASTMODIFIED. The execution time is 0.007 seconds.

	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	LASTMODIFIED
1	101	1	Savings	12487.24812	6/28/2025, 1:52:06
2	102	2	Checking	9000	6/28/2025, 1:05:40
3	103	3	Savings	5203.02005	6/28/2025, 1:52:06
4	104	4	Savings	15609.06015	6/28/2025, 1:52:06
5	105	5	Checking	3000	6/28/2025, 1:05:40

- **Stored Procedure** **TransferFunds**

```
CREATE OR REPLACE PROCEDURE TransferFunds (  
  p_FromAccountID IN NUMBER,  
  p_ToAccountID   IN NUMBER,  
  p_Amount        IN NUMBER  
) IS  
  v_FromBalance NUMBER;  
BEGIN  
  -- Get source account balance  
  SELECT Balance INTO v_FromBalance  
  FROM Accounts  
  WHERE AccountID = p_FromAccountID
```

```
FOR UPDATE;

-- Check if sufficient funds
IF v_FromBalance < p_Amount THEN
    RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance in source account.')
END IF;

-- Deduct from source
UPDATE Accounts
SET Balance = Balance - p_Amount,
    LastModified = SYSDATE
WHERE AccountID = p_FromAccountID;

-- Add to destination
UPDATE Accounts
SET Balance = Balance + p_Amount,
    LastModified = SYSDATE
WHERE AccountID = p_ToAccountID;

COMMIT;
END;
/
```

