superset id: 6405870

Stored Procedure

What is a Stored Procedure?
A **Stored Procedure** is a **precompiled block of code** stored in the database that performs a specific task.

Key Points:

- Written using **PL/SQL (Oracle)** or **T-SQL (SQL Server)**.
- Can accept **parameters (IN, OUT, IN OUT)**.
- Improves performance (compiled once, reused).
- Helps in **code reuse**, **modularization**, and **security**.
- Stored and executed **on the database server**.

---

2. Table Structures

- **Table: Accounts**
  - AccountID: NUMBER (Primary Key)
  - AccountHolderName: VARCHAR2(100)
  - AccountType: VARCHAR2(20)
  - Balance: NUMBER(15,2)
- **Table: Employees**
  - EmployeeID: NUMBER (Primary Key)
  - EmployeeName: VARCHAR2(100)
  - DepartmentID: NUMBER
  - Salary: NUMBER(10,2)

---

3. Table Creation Script

```
CREATE TABLE Accounts (
   AccountID NUMBER PRIMARY KEY,
   AccountHolderName VARCHAR2(100),
   AccountType VARCHAR2(20),
   Balance NUMBER(15, 2)
);

CREATE TABLE Employees (
   EmployeeID NUMBER PRIMARY KEY,
   EmployeeName VARCHAR2(100),
   DepartmentID NUMBER,
   Salary NUMBER(10, 2)
);
```

---

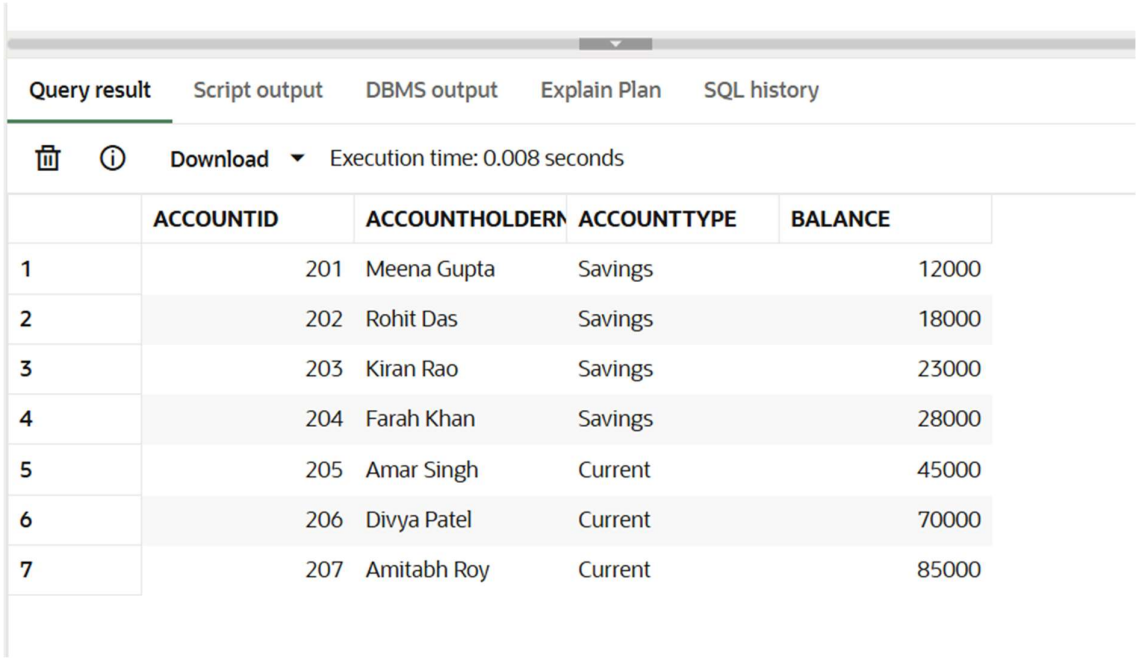
4. Insert Sample Data

```
-- Accounts
INSERT INTO Accounts VALUES (201, 'Meena Gupta', 'Savings', 12000);
INSERT INTO Accounts VALUES (202, 'Rohit Das', 'Savings', 18000);
INSERT INTO Accounts VALUES (203, 'Kiran Rao', 'Savings', 23000);
INSERT INTO Accounts VALUES (204, 'Farah Khan', 'Savings', 28000);
INSERT INTO Accounts VALUES (205, 'Amar Singh', 'Current', 45000);
INSERT INTO Accounts VALUES (206, 'Divya Patel', 'Current', 70000);
INSERT INTO Accounts VALUES (207, 'Amitabh Roy', 'Current', 85000);

-- Employees
INSERT INTO Employees VALUES (301, 'Vinay Kumar', 101, 48000);
```

superset id: 6405870

```
INSERT INTO Employees VALUES (302, 'Rani Paul', 101, 58000);
INSERT INTO Employees VALUES (303, 'Sanjay Naik', 101, 62000);
INSERT INTO Employees VALUES (304, 'Anjali Jain', 202, 53000);
INSERT INTO Employees VALUES (305, 'Ravi Reddy', 202, 59000);

COMMIT;
```

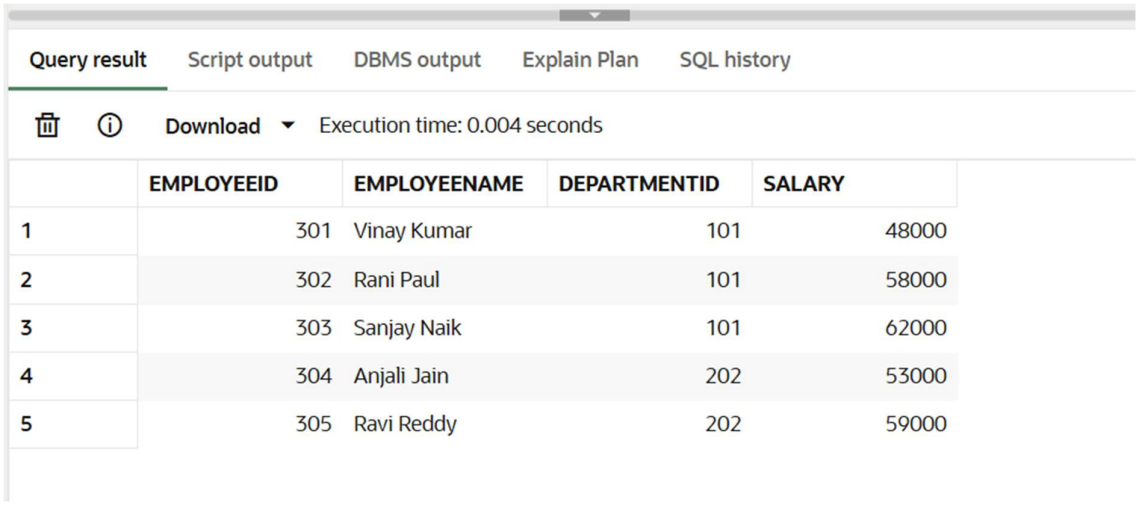
Acounts table:



Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.008 seconds

	ACCOUNTID	ACCOUNTHOLDERN	ACCOUNTTYPE	BALANCE
1	201	Meena Gupta	Savings	12000
2	202	Rohit Das	Savings	18000
3	203	Kiran Rao	Savings	23000
4	204	Farah Khan	Savings	28000
5	205	Amar Singh	Current	45000
6	206	Divya Patel	Current	70000
7	207	Amitabh Roy	Current	85000



Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.004 seconds

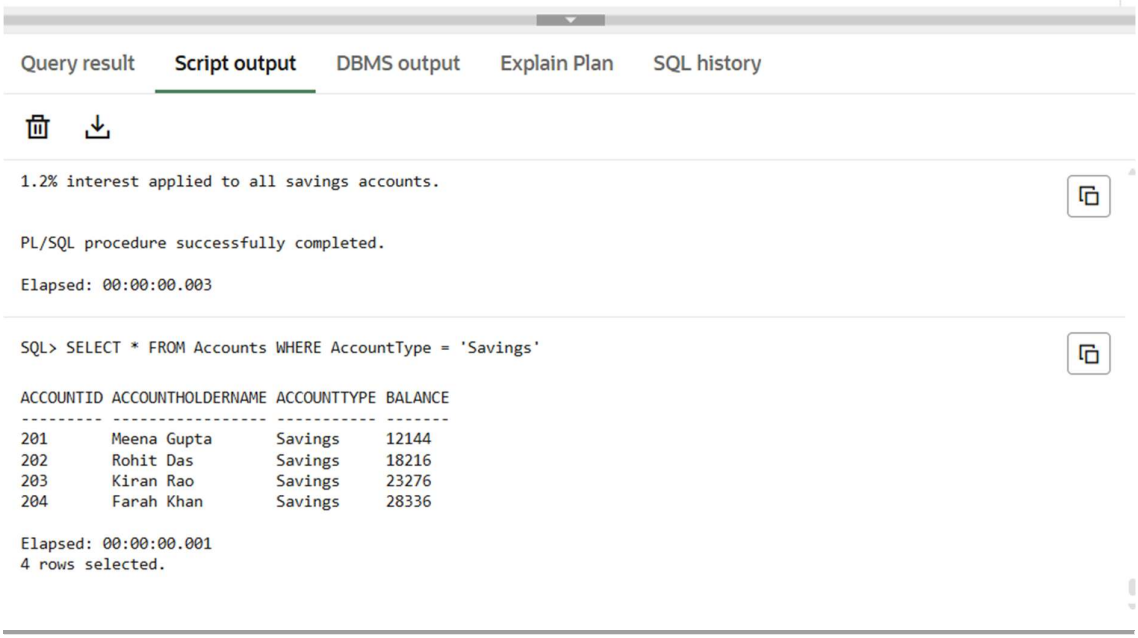
	EMPLOYEEID	EMPLOYEENAME	DEPARTMENTID	SALARY
1	301	Vinay Kumar	101	48000
2	302	Rani Paul	101	58000
3	303	Sanjay Naik	101	62000
4	304	Anjali Jain	202	53000
5	305	Ravi Reddy	202	59000

Scenario 1: Process Monthly Interest (for Savings Accounts)
Description:
Apply 1.2% interest to all savings accounts.
Procedure Code

superset id: 6405870

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
   UPDATE Accounts
   SET Balance = Balance + (Balance * 0.012)
   WHERE AccountType = 'Savings';

   DBMS_OUTPUT.PUT_LINE('1.2% interest applied to all savings accounts.');
END;
/
```

Explanation:
This procedure identifies all rows where AccountType = 'Savings' and updates their balances with 1.2% interest.

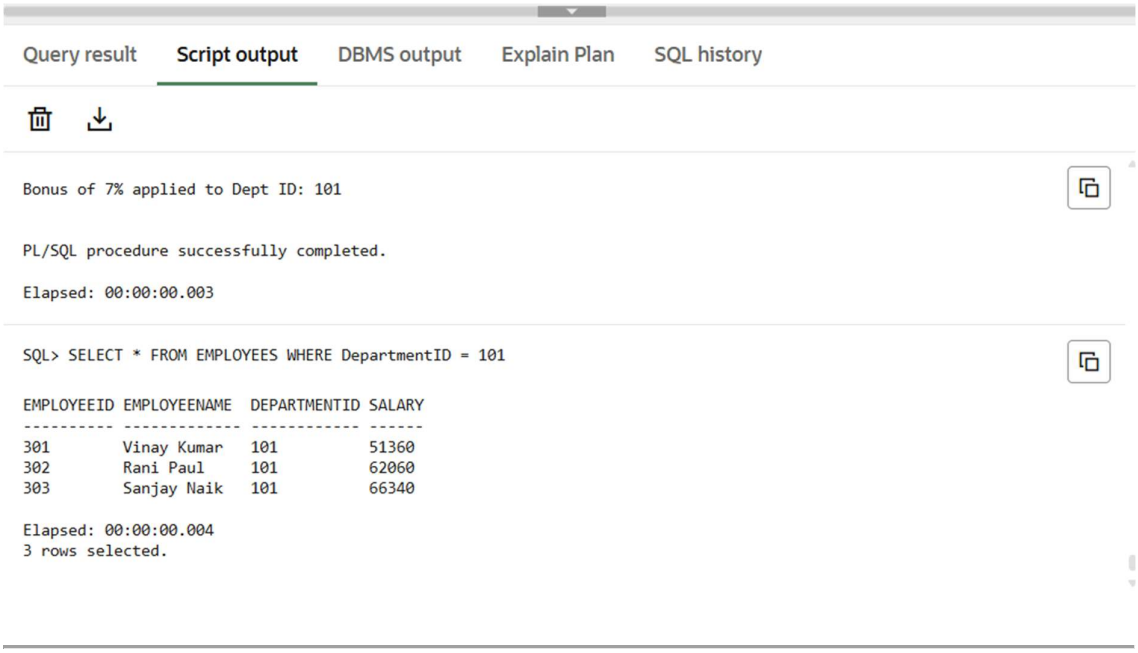


Scenario 2: Update Employee Bonus (Department-wise)
Description:
Increase salary for all employees in a specific department by a given bonus percentage.
Procedure Code:

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
   dept_id IN NUMBER,
   bonus_percent IN NUMBER
) IS
BEGIN
   UPDATE Employees
   SET Salary = Salary + (Salary * bonus_percent / 100)
   WHERE DepartmentID = dept_id;

   DBMS_OUTPUT.PUT_LINE('Bonus of ' || bonus_percent || '% applied to Dept ID: ' || dept_id);
END;
/
```

Explanation:
This stored procedure increases salaries of employees in a selected department by the provided percentage. It uses parameters and basic arithmetic logic.

superset id: 6405870



Scenario 3: Transfer Funds Between Accounts
Description:
Transfer funds from one account to another with validation and error handling.
Procedure Code:

```
CREATE OR REPLACE PROCEDURE TransferFunds (
    from_account IN NUMBER,
    to_account IN NUMBER,
    amount IN NUMBER
) IS
    source_balance NUMBER;
BEGIN
    SELECT Balance INTO source_balance
    FROM Accounts
    WHERE AccountID = from_account
    FOR UPDATE;

    IF source_balance < amount THEN
        RAISE_APPLICATION_ERROR(-20001, 'Transfer failed: insufficient funds.');
    END IF;

    UPDATE Accounts
    SET Balance = Balance - amount
    WHERE AccountID = from_account;

    UPDATE Accounts
    SET Balance = Balance + amount
    WHERE AccountID = to_account;

    DBMS_OUTPUT.PUT_LINE('₹' || amount || ' transferred from Account ' || from_account || ' to
Account ' || to_account);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
```

```
      DBMS_OUTPUT.PUT_LINE('Transfer failed: Invalid account ID(s).');
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Unexpected Error: ' || SQLERRM);
END;
/
```

Explanation:

- Retrieves balance of sender account using SELECT ... FOR UPDATE
- Validates if sender has enough balance
- Deducts from sender and adds to receiver
- Handles exceptions like NO_DATA_FOUND and generic errors using WHEN OTHERS