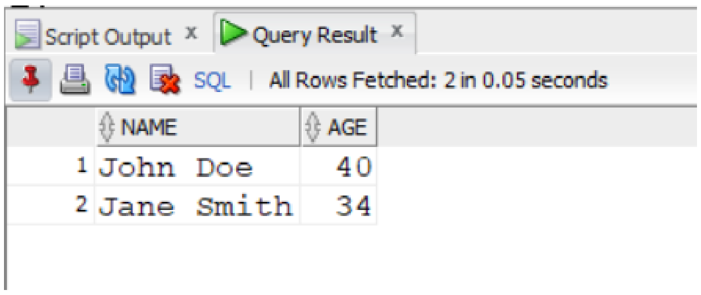
Exercise 4: Functions

Scenario 1: Calculate the age of customers for eligibility checks.
Question: Write a function CalculateAge that takes a customer's date of birth as input and returns their age in years.

Solution:

```
CREATE OR REPLACE FUNCTION CalculateAge (
    dob IN DATE
) RETURN NUMBER IS
    age NUMBER;
BEGIN
    age := FLOOR(MONTHS_BETWEEN(SYSDATE, dob) / 12);
    RETURN age;
END;
/

SELECT Name, CalculateAge(DOB) AS Age
FROM Customers;
```



	NAME	AGE
1	John Doe	40
2	Jane Smith	34

Scenario 2: The bank needs to compute the monthly installment for a loan.
Question: Write a function **CalculateMonthlyInstallment** that takes the loan amount, interest rate, and loan duration in years as input and returns the monthly installment amount.

Solution:

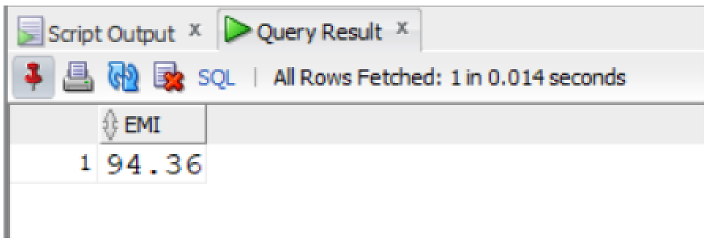
```
CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment (
    loan_amount IN NUMBER,
    annual_interest_rate IN NUMBER,
    loan_duration_years IN NUMBER
) RETURN NUMBER IS
```

```
    monthly_rate NUMBER;
    total_months NUMBER;
    emi NUMBER;
BEGIN
    monthly_rate := annual_interest_rate / 12 / 100;
    total_months := loan_duration_years * 12;

    IF monthly_rate = 0 THEN
        emi := loan_amount / total_months;
    ELSE
        emi := loan_amount * monthly_rate * POWER(1 + monthly_rate, total_months) /
               (POWER(1 + monthly_rate, total_months) - 1);
    END IF;

    RETURN ROUND(emi, 2);
END;
/

SELECT CalculateMonthlyInstallment(5000, 5, 5) AS EMI FROM dual;
```



```
All Rows Fetched: 1 in 0.014 seconds

   EMI
1  94.36
```

Scenario 3: Check if a customer has sufficient balance before making a transaction.
Question: Write a function **HasSufficientBalance** that takes an account ID and an amount as input and returns a boolean indicating whether the account has at least the specified amount.

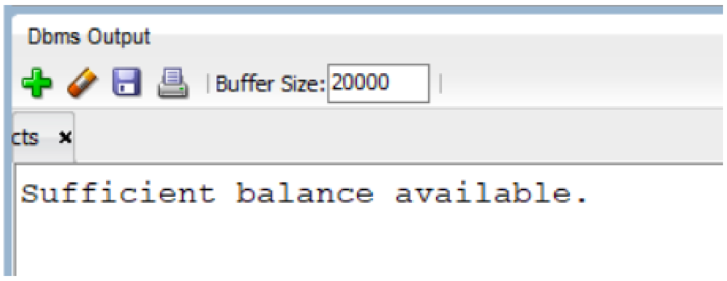
Solution:

```
CREATE OR REPLACE FUNCTION HasSufficientBalance (
    acc_id IN NUMBER,
    amt IN NUMBER
) RETURN BOOLEAN IS
    current_balance NUMBER;
BEGIN
    SELECT Balance INTO current_balance
    FROM Accounts
    WHERE AccountID = acc_id;

    RETURN current_balance >= amt;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN FALSE;
END;
/

DECLARE
    result BOOLEAN;
BEGIN
    result := HasSufficientBalance(1, 500);

    IF result THEN
        DBMS_OUTPUT.PUT_LINE('Sufficient balance available.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Insufficient balance.');
    END IF;
END;
/
```



```
Sufficient balance available.
```