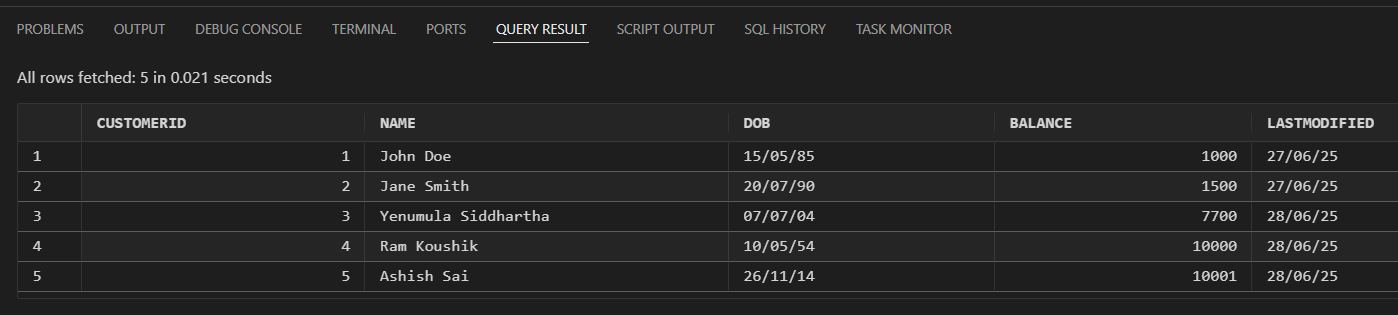
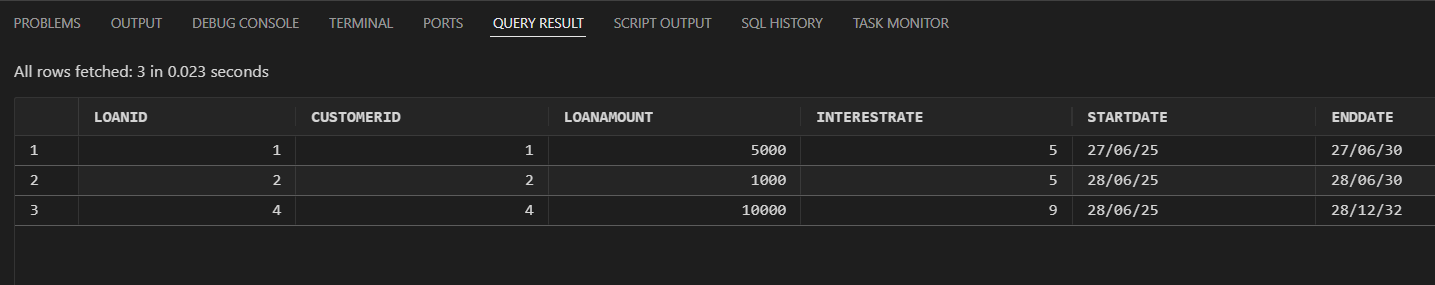
**WEEK 2 PL/SQL EXERCISES**

**TABLES CREATED :**

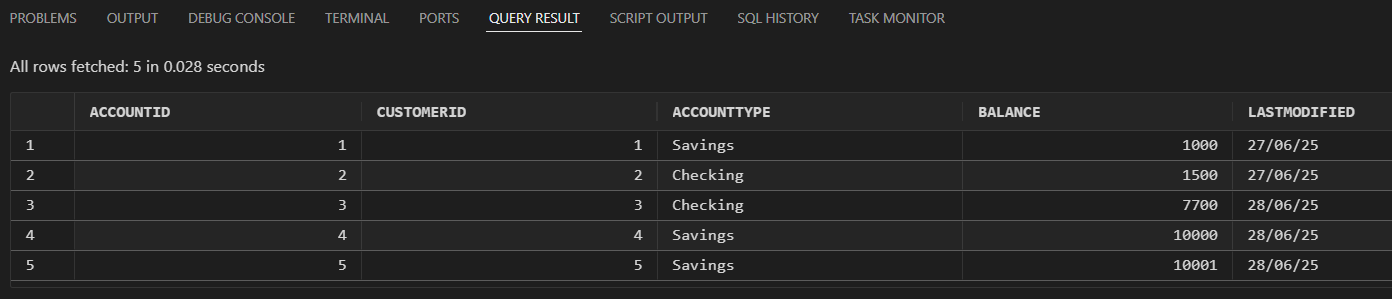
**Customers**

****

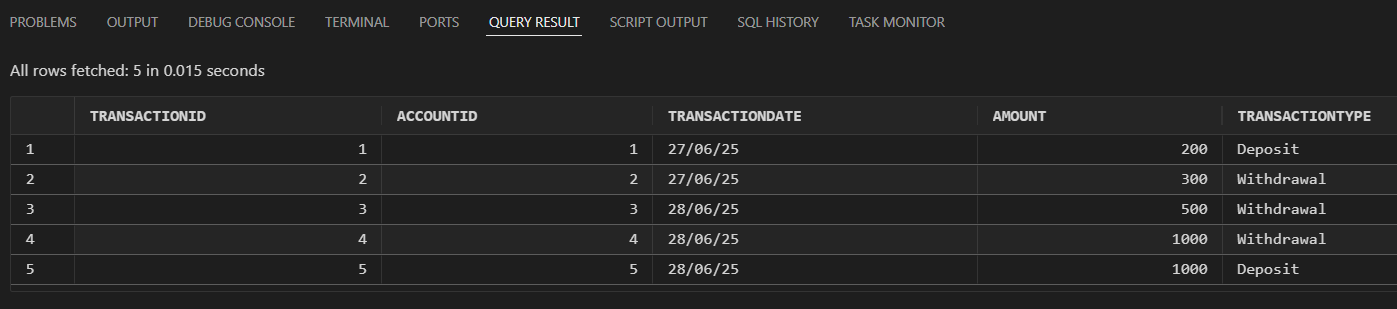
**Loans**



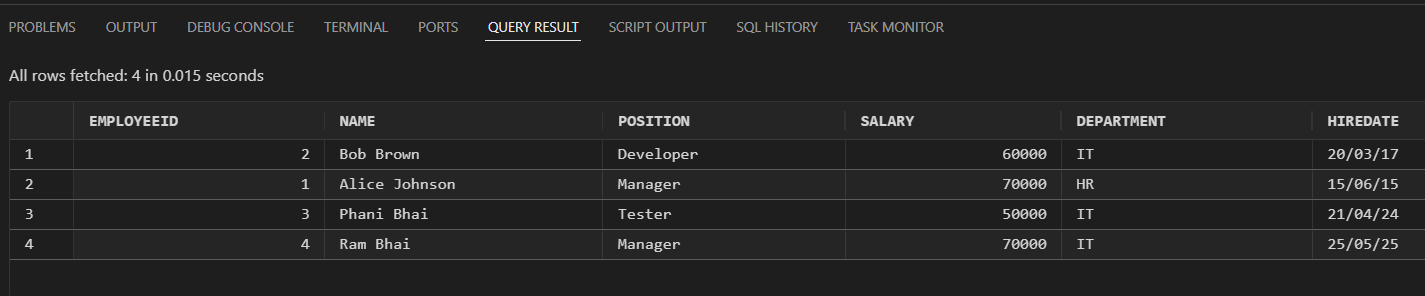
**Accounts**

****

**Transactions**

****

**Employees**

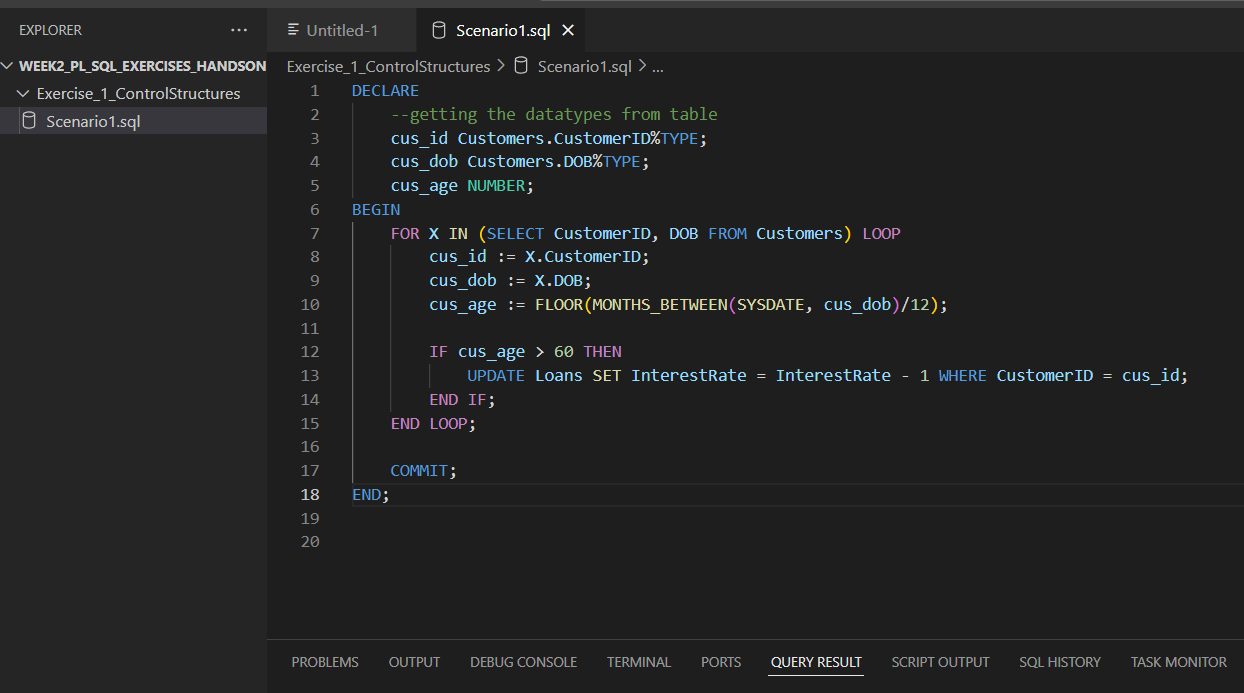
****

**Exercise 1: Control Structures**

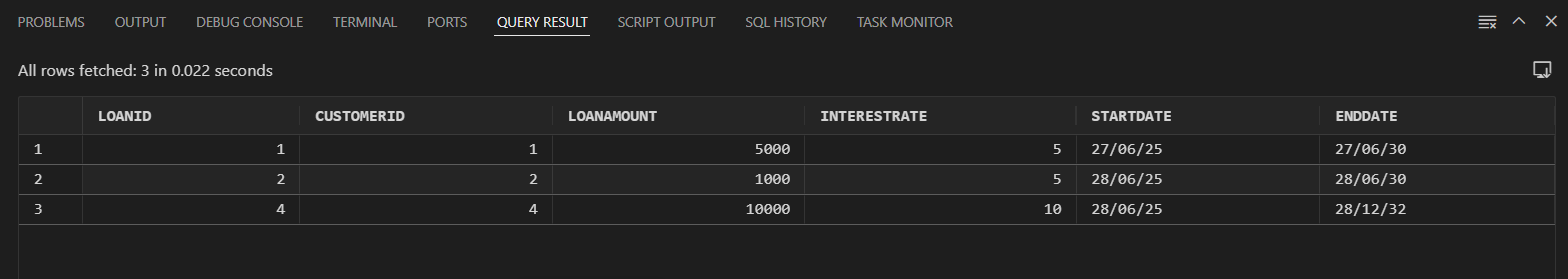
**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

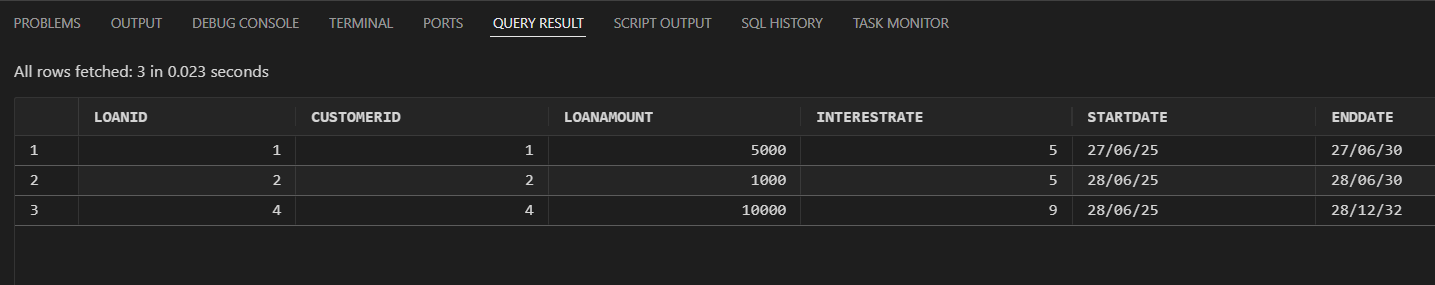
**Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

CODE :



RESULT:

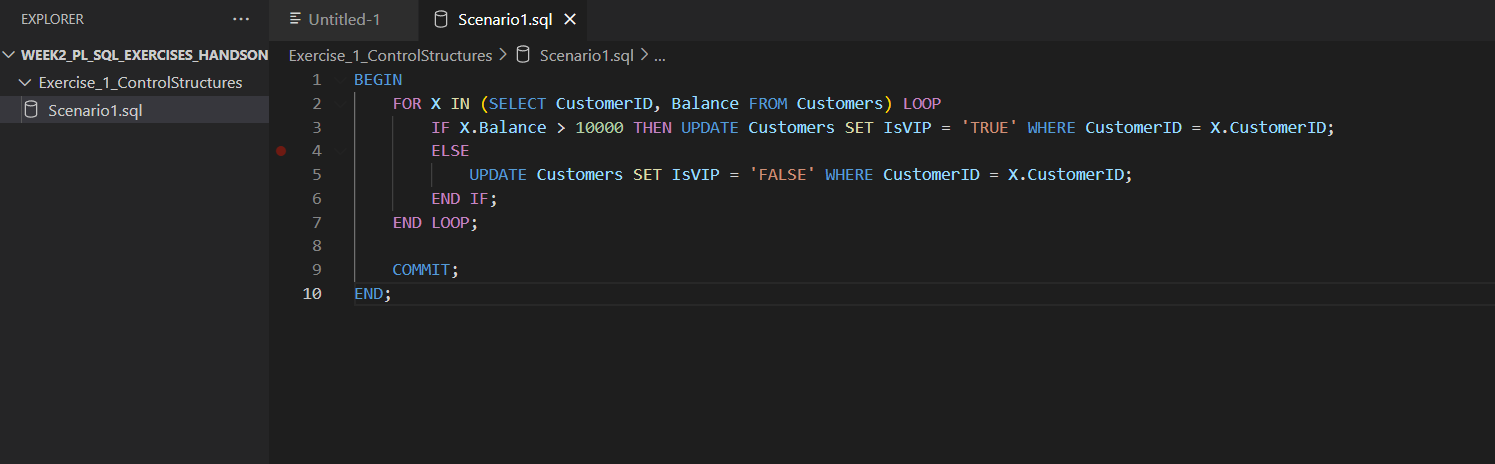




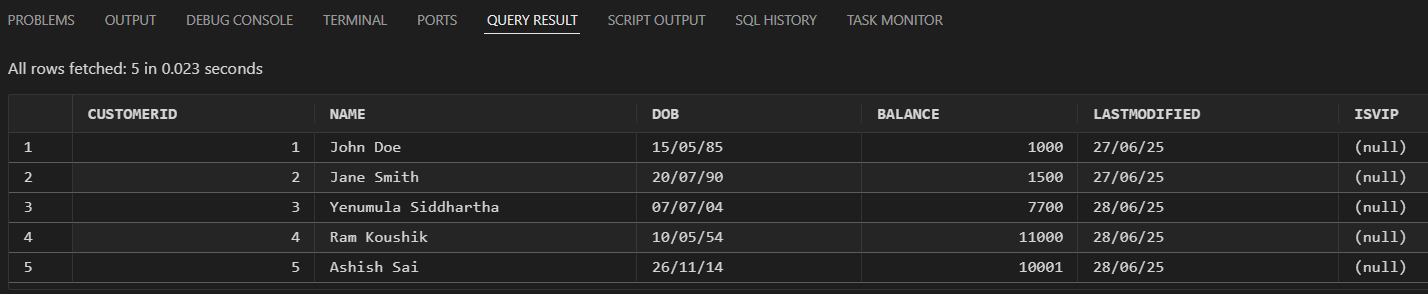
**Scenario 2:** A customer can be promoted to VIP status based on their balance.

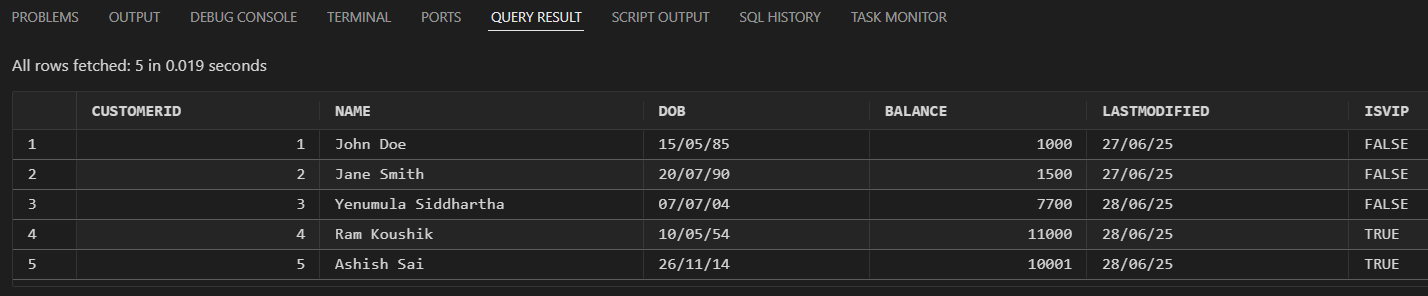
**Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

CODE :



RESULT :

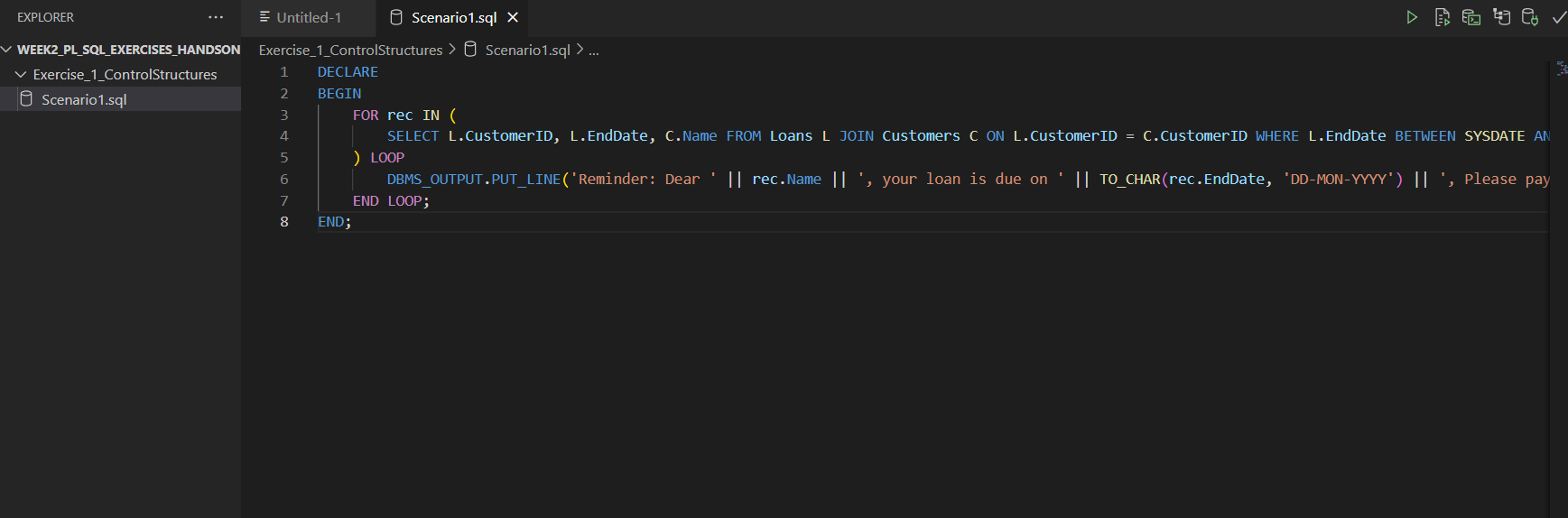




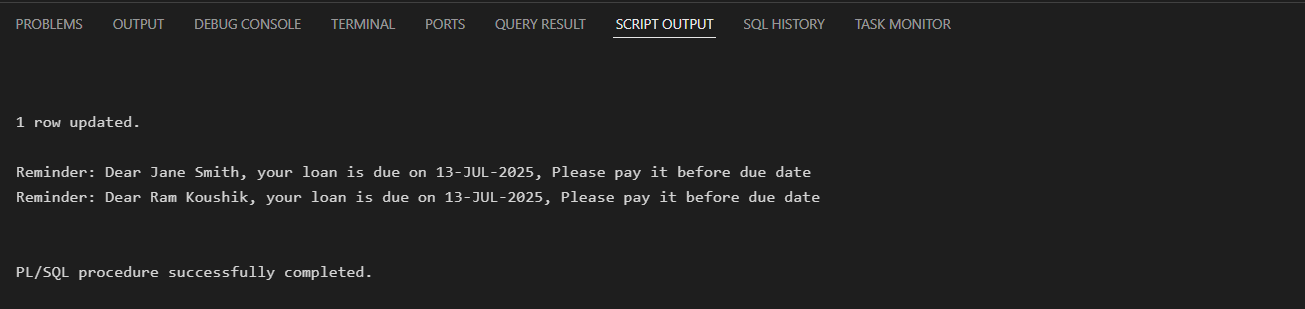
**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

**Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

CODE :



RESULT :

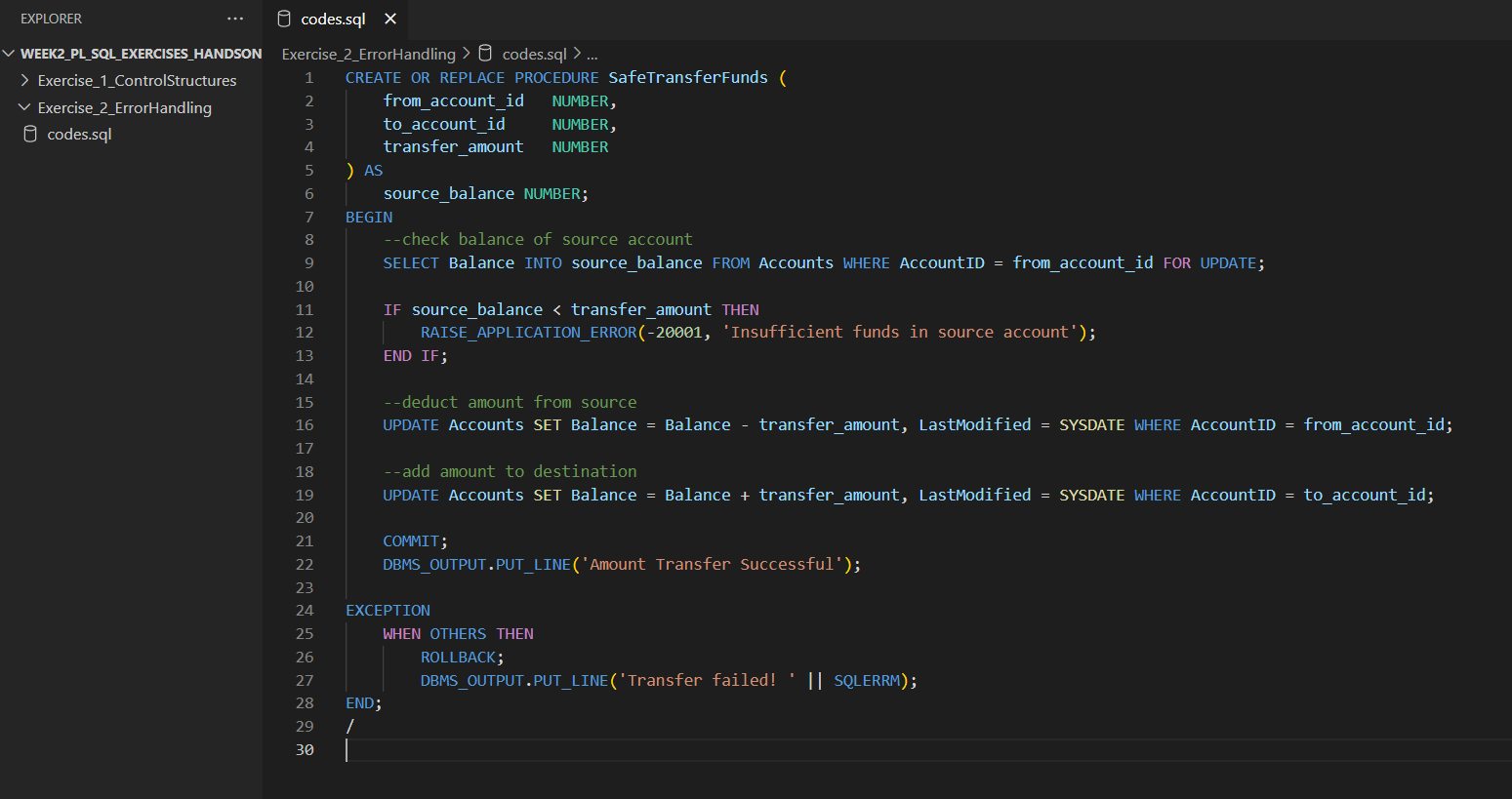


**Exercise 2: Error Handling**

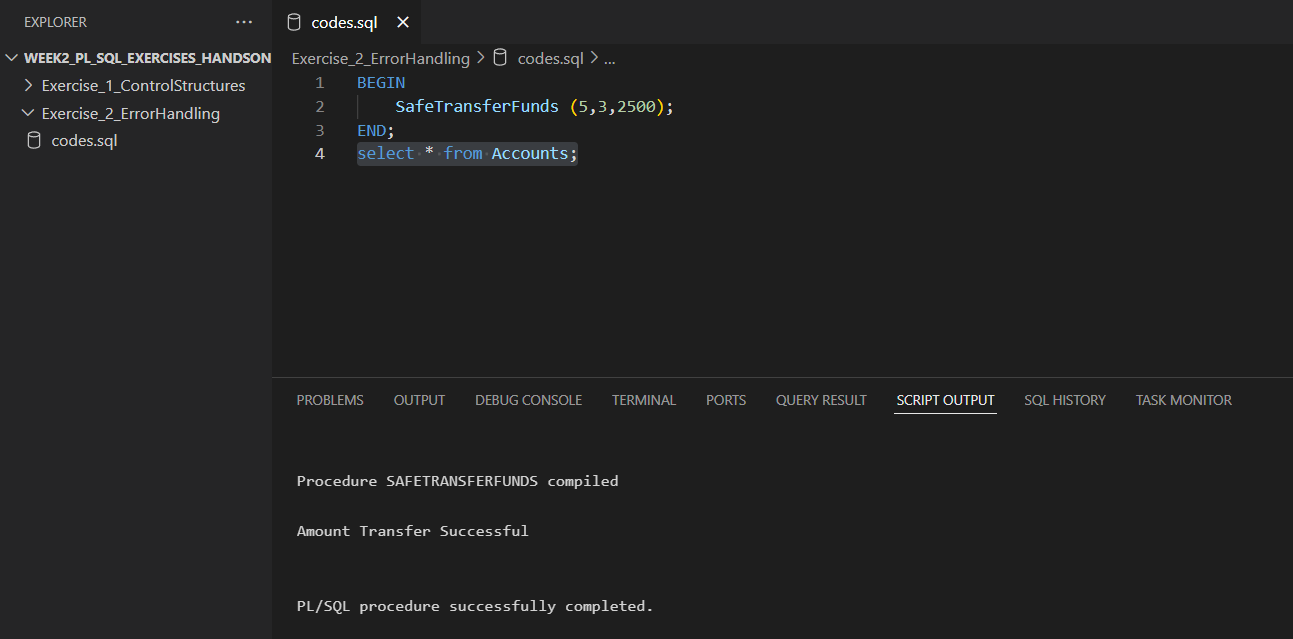
**Scenario 1:** Handle exceptions during fund transfers between accounts.

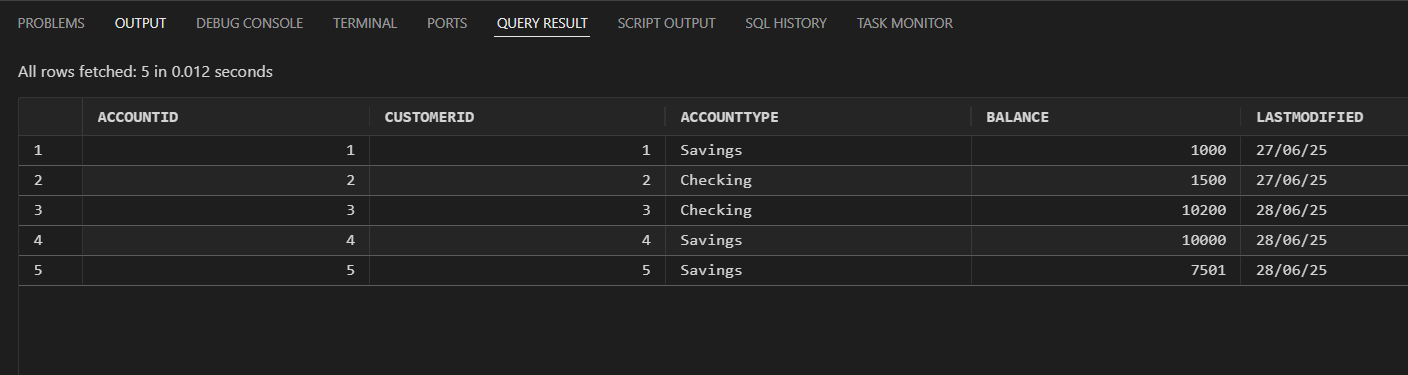
**Question:** Write a stored procedure **SafeTransferFunds** that transfers funds between two accounts. Ensure that if any error occurs (e.g., insufficient funds), an appropriate error message is logged and the transaction is rolled back.

CODE :



RESULT :

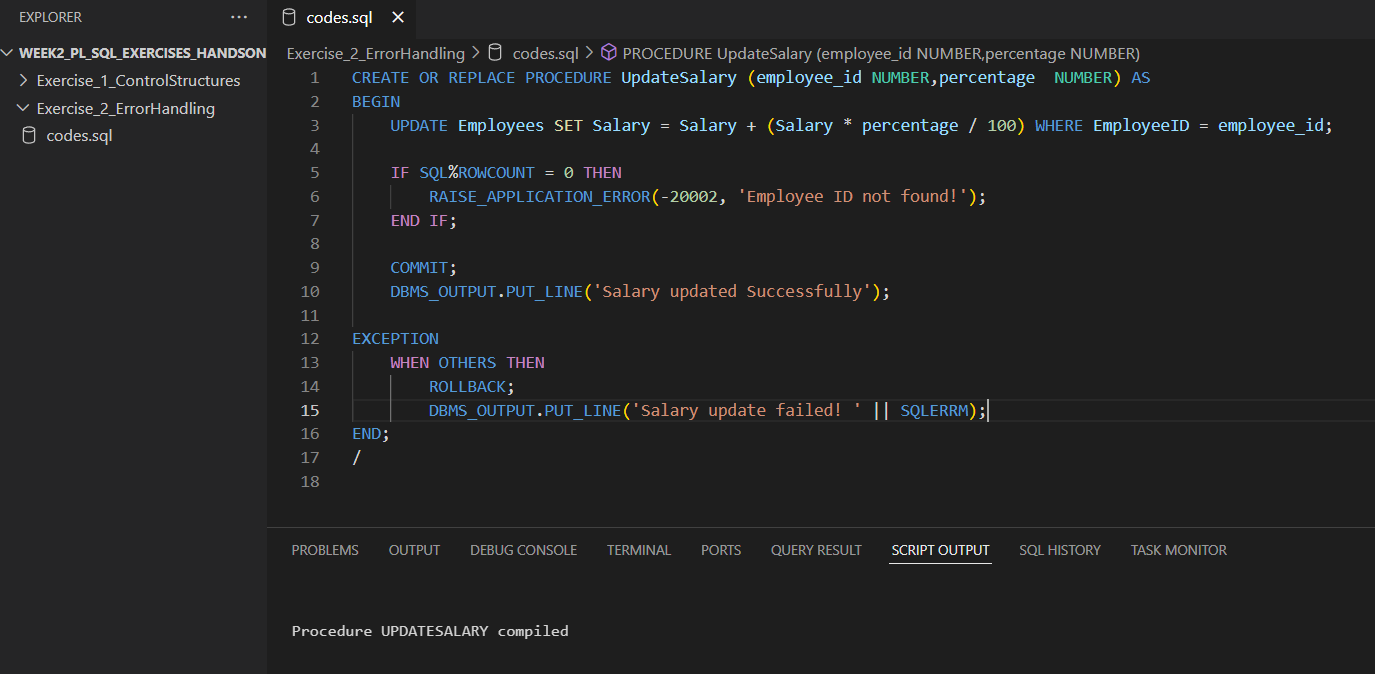




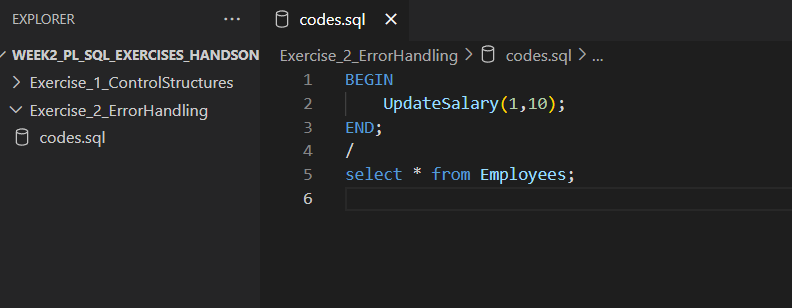
**Scenario 2:** Manage errors when updating employee salaries.

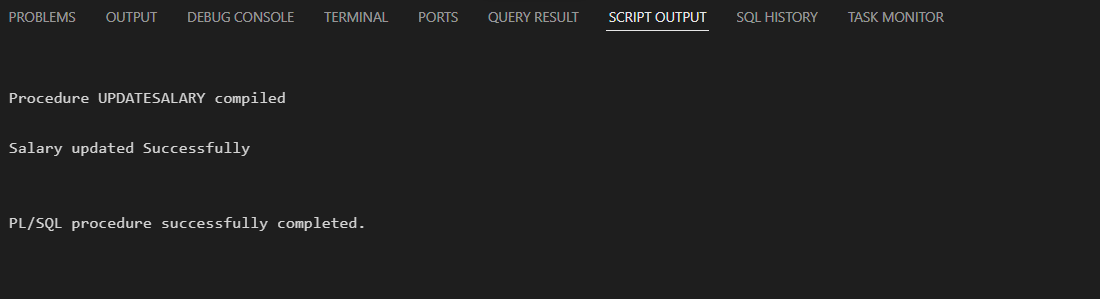
**Question:** Write a stored procedure **UpdateSalary** that increases the salary of an employee by a given percentage. If the employee ID does not exist, handle the exception and log an error message.

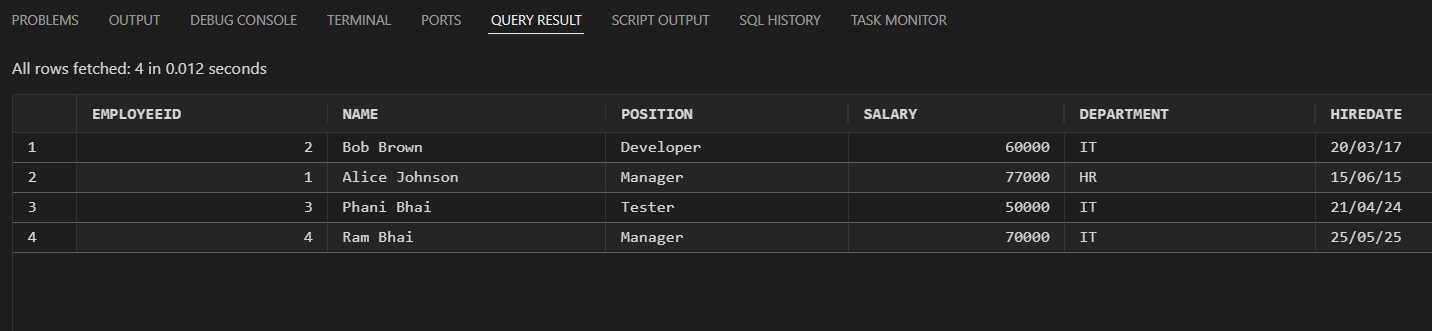
CODE :



RESULT :



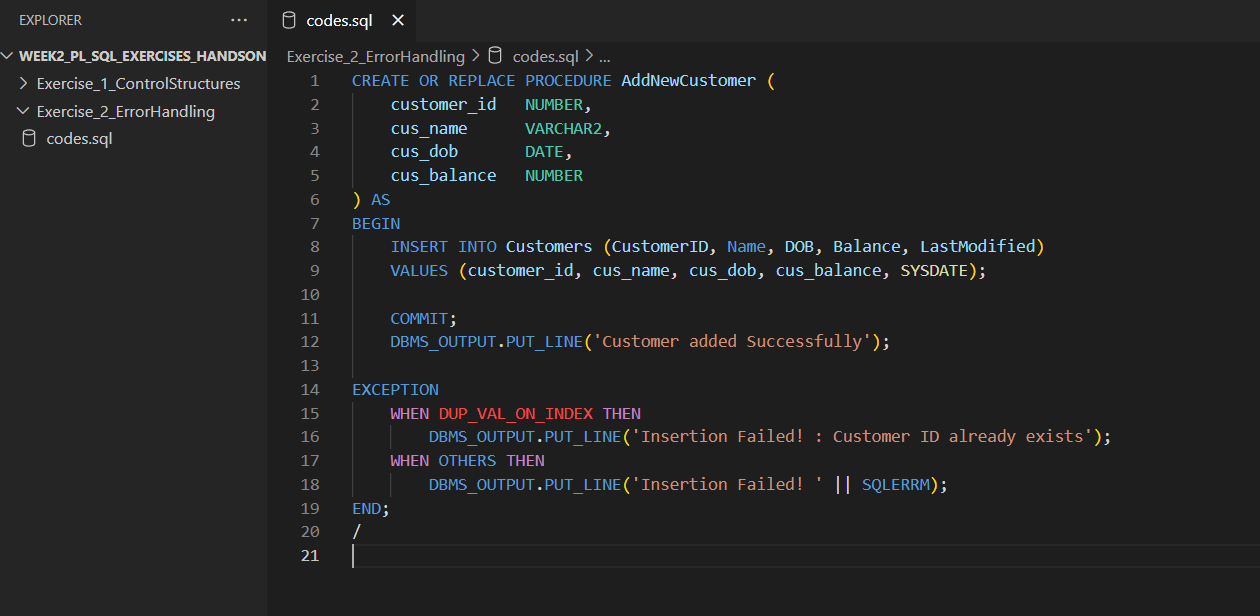




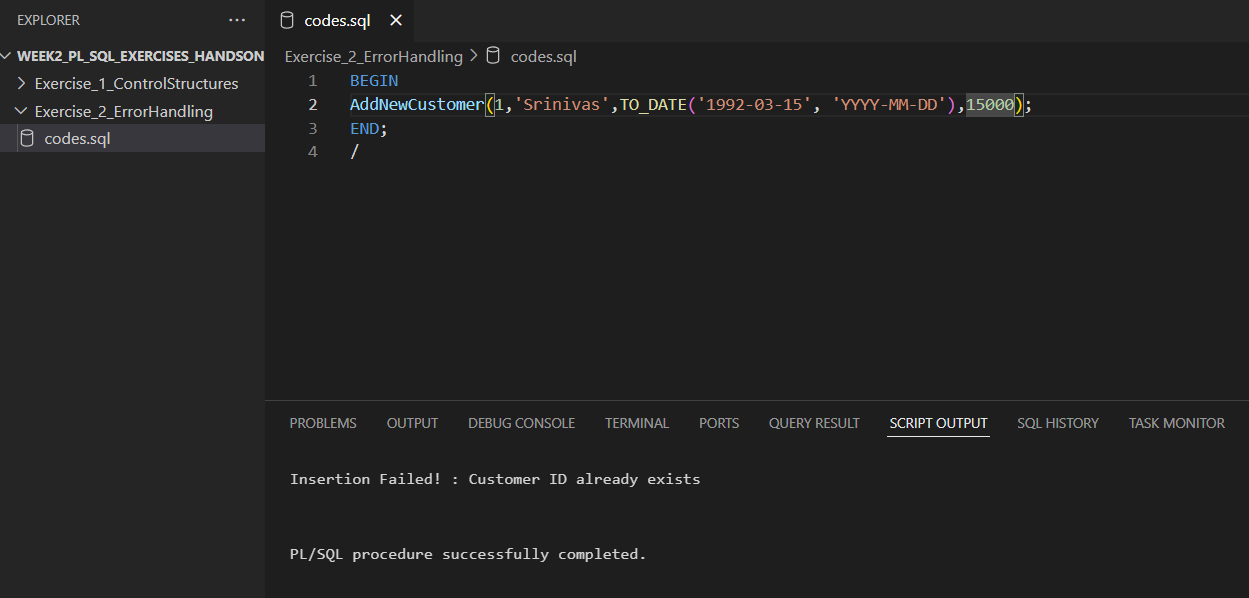
**Scenario 3:** Ensure data integrity when adding a new customer.

**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.

CODE :



RESULT :

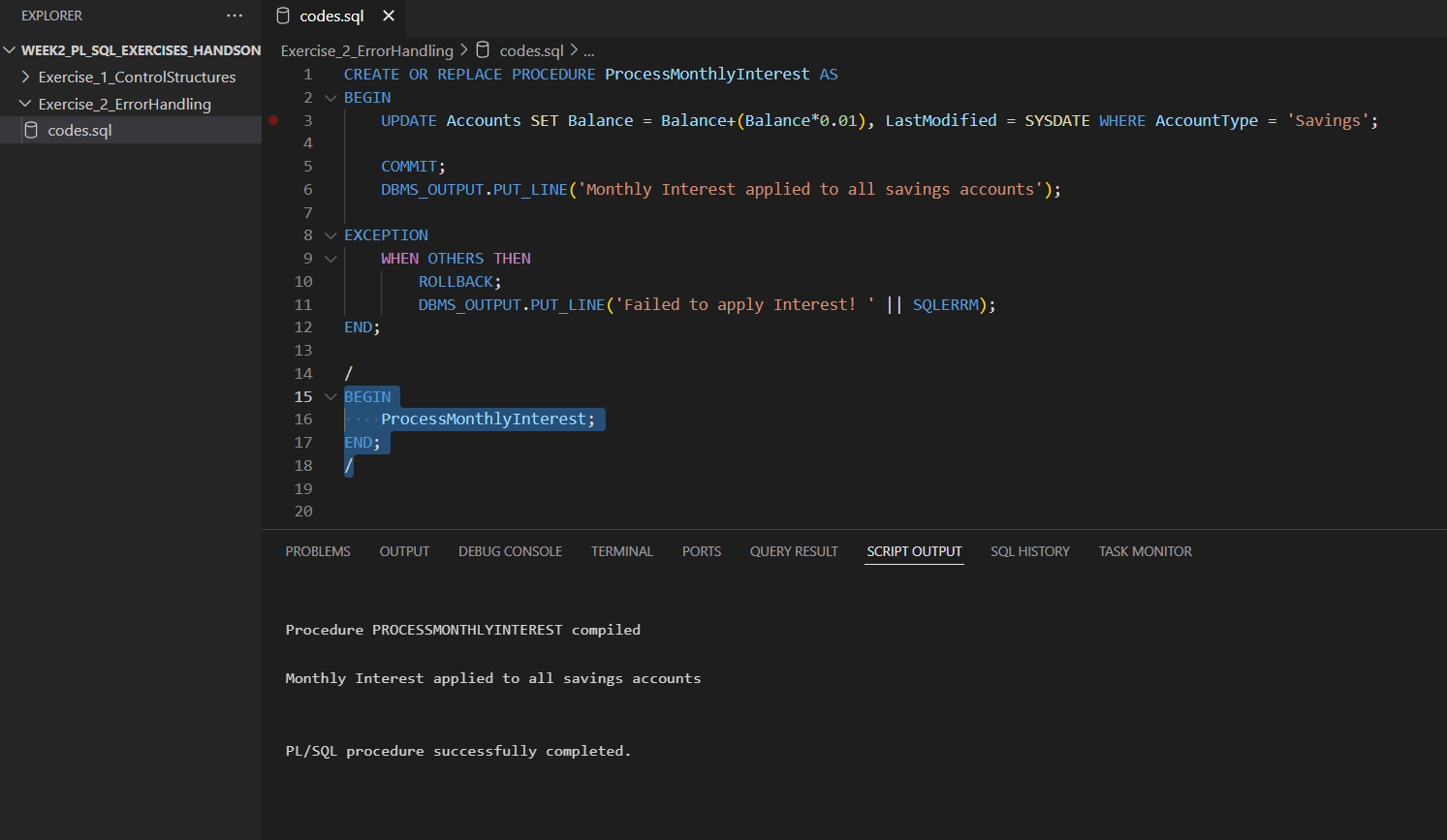


**Exercise 3: Stored Procedures**

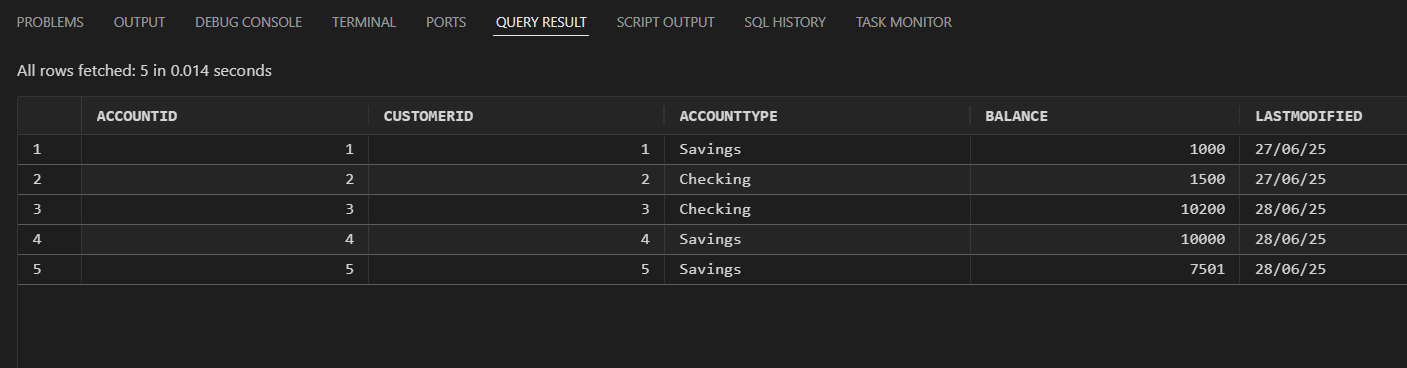
**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

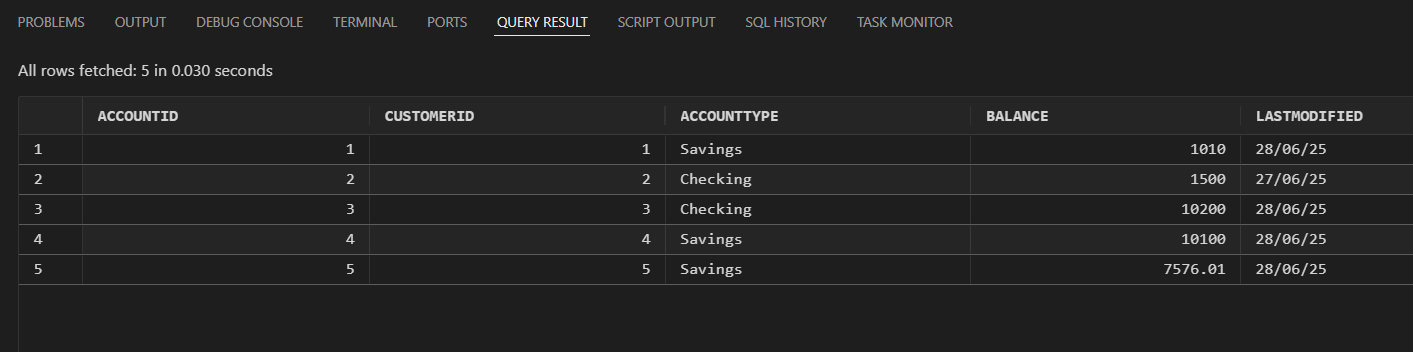
**Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

CODE :



RESULT :

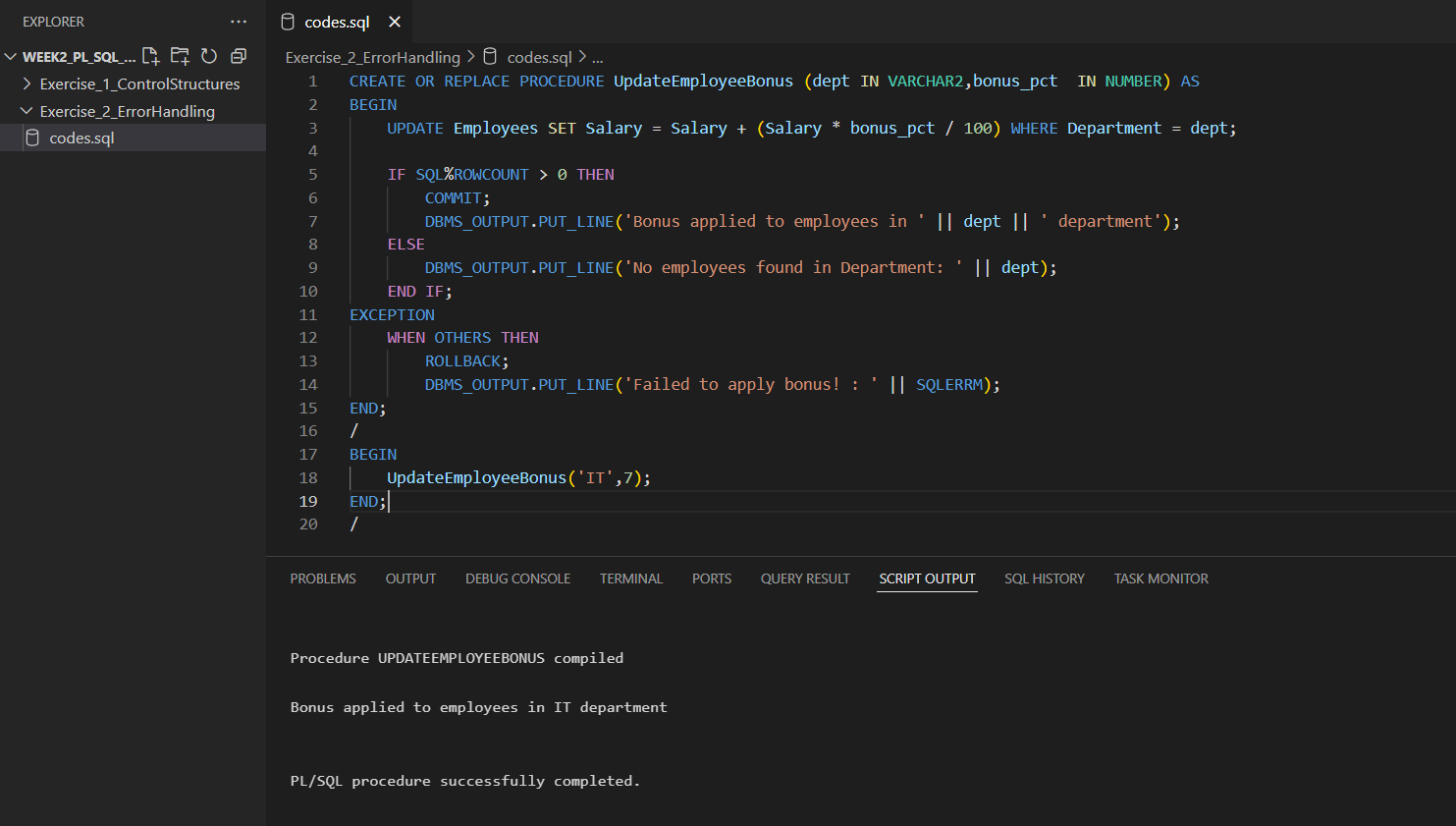




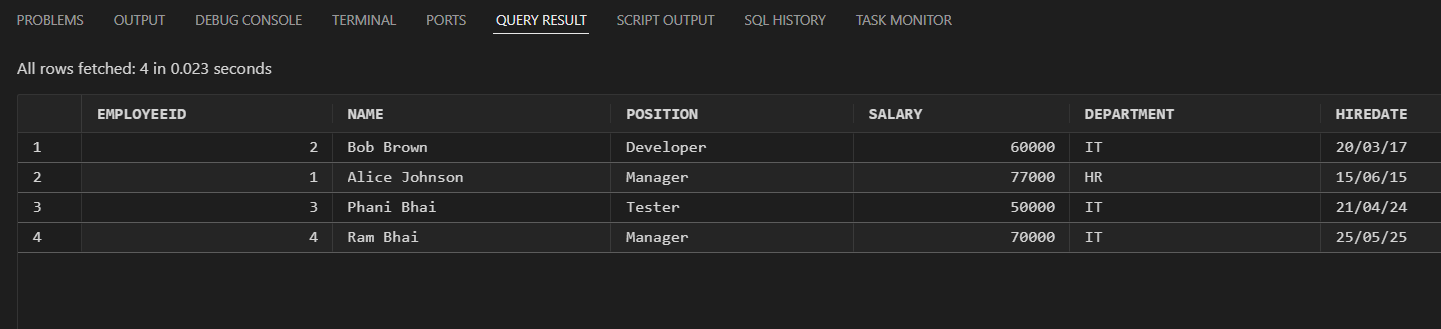
**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

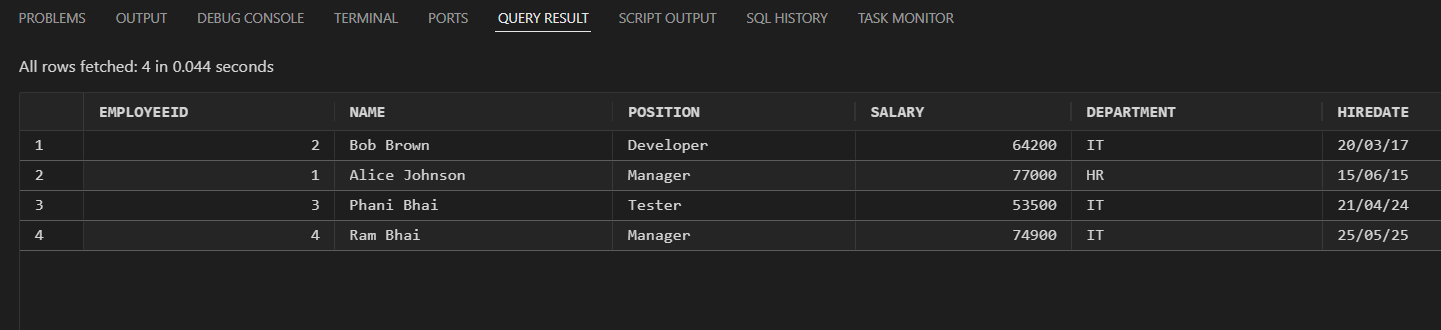
**Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

CODE :



RESULT :

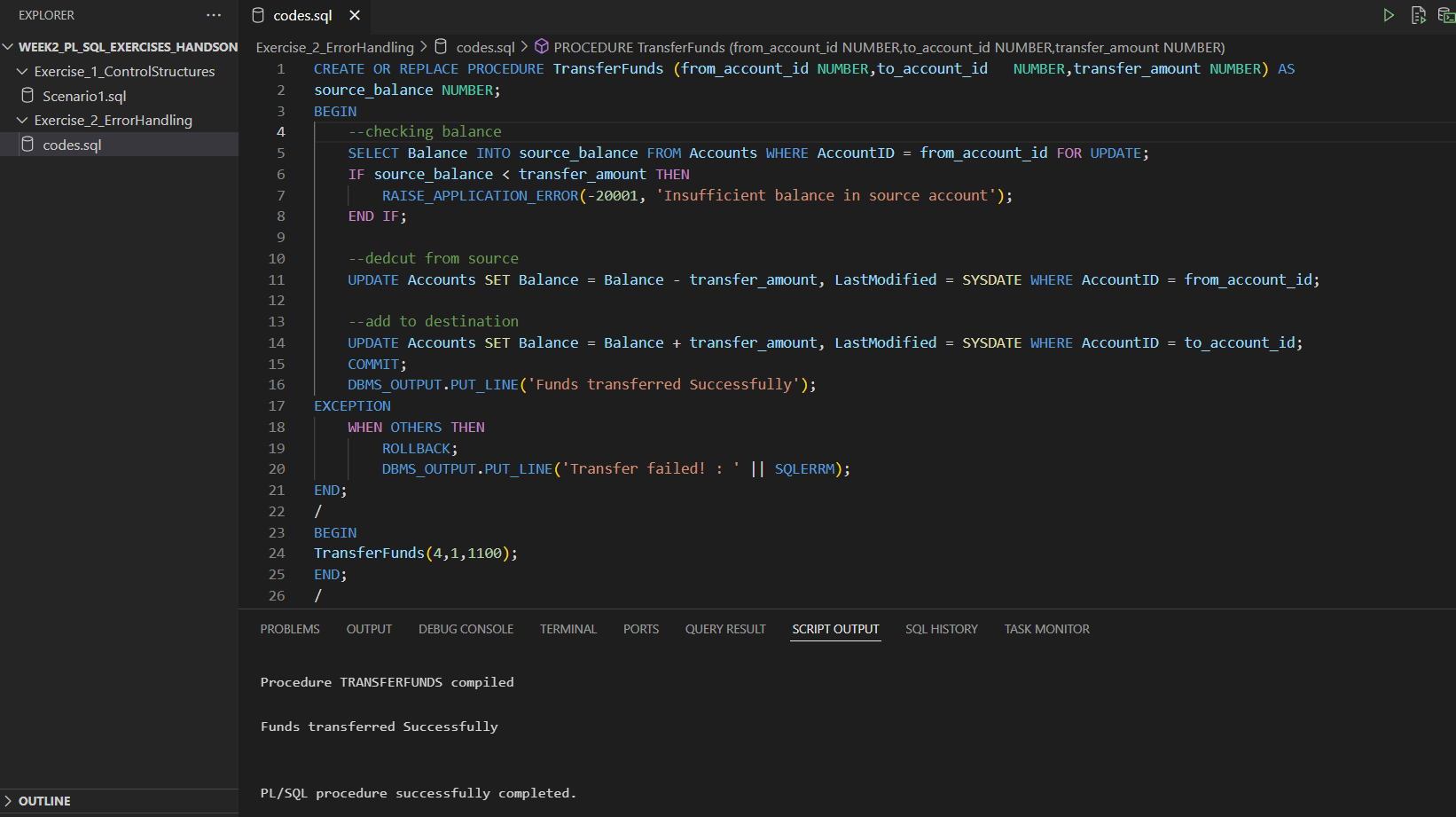




**Scenario 3:** Customers should be able to transfer funds between their accounts.

**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.

CODE :



RESULT :

