**Cognizant Digital Nurture 4.0**

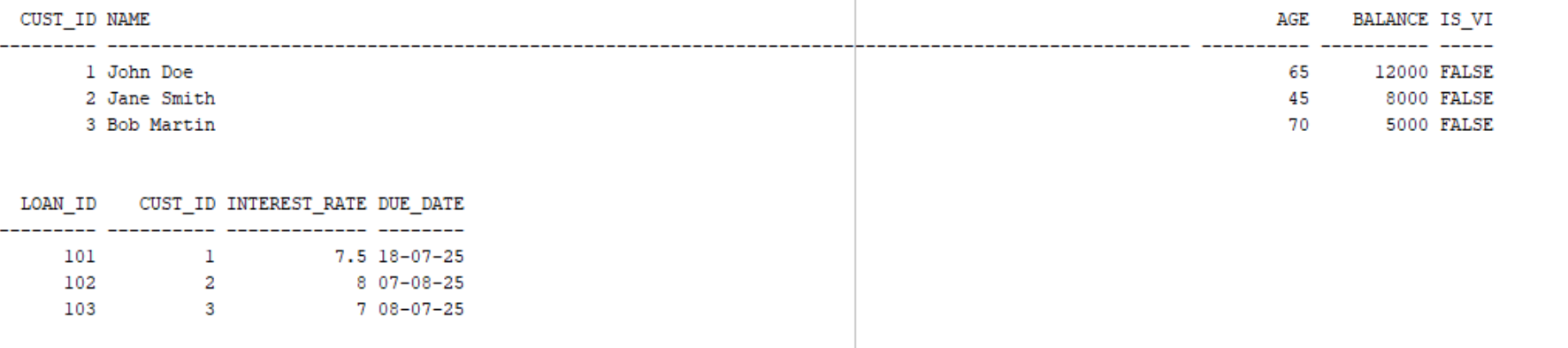
***WEEK-2 Module 3 - PL/SQL Programming***

**Note: all the exercises are done in Oracle SQL developer.**  
**Exercise 1:**

**Control Structures**

Let’s create a structure which can support all three scenarios.

Two tables: customers and loans are created and sample data is inserted.

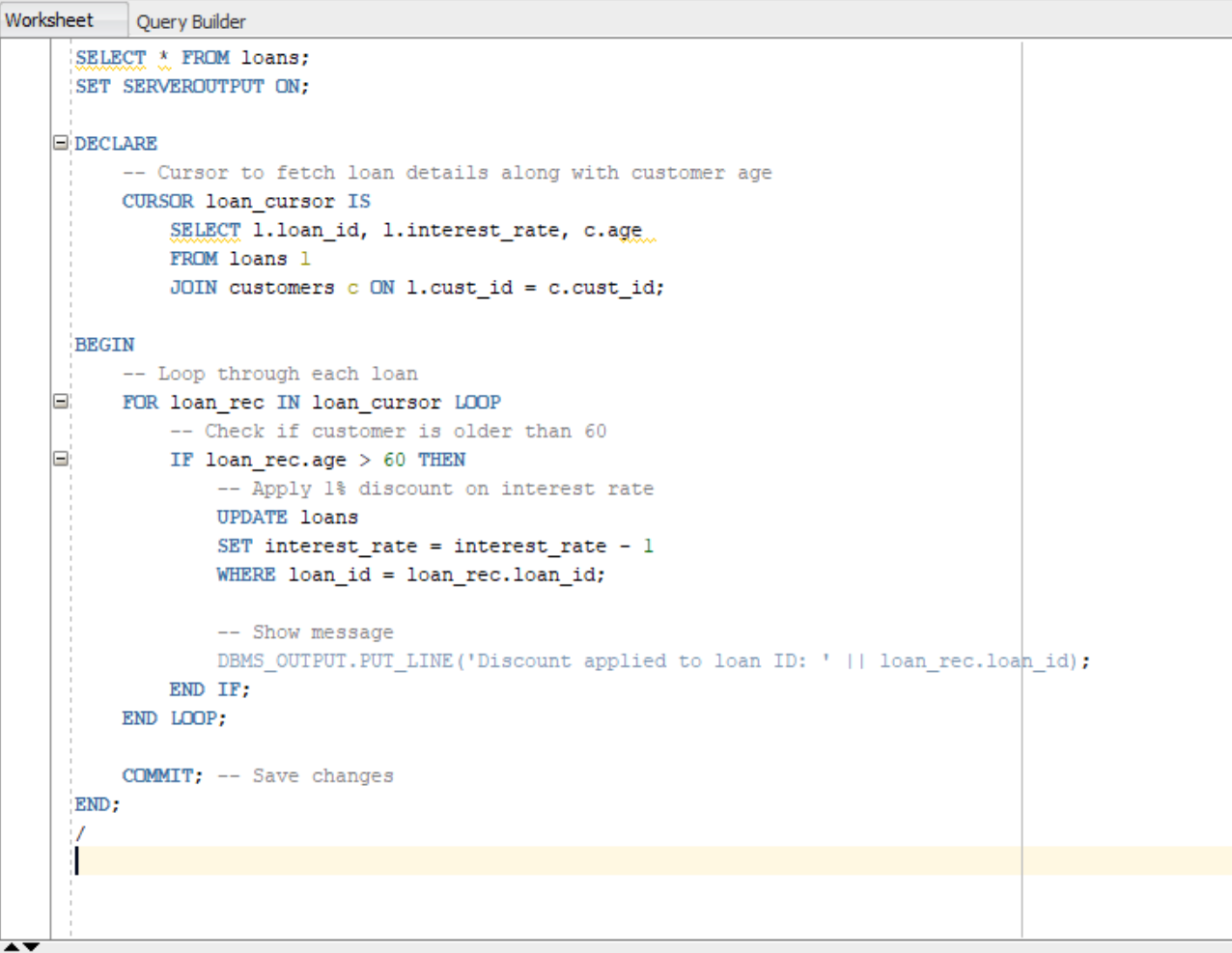


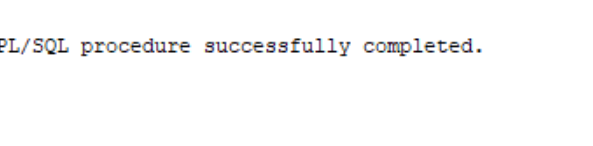
**Scenario 1:**

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

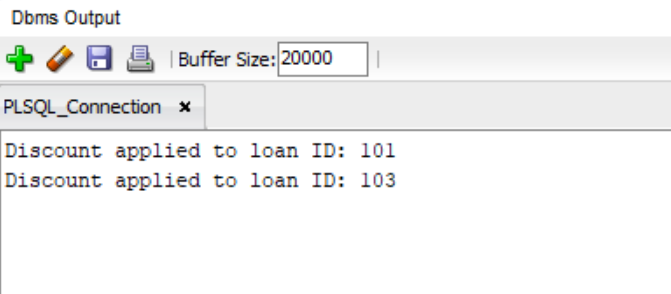
Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

**Code:**





**Output:**



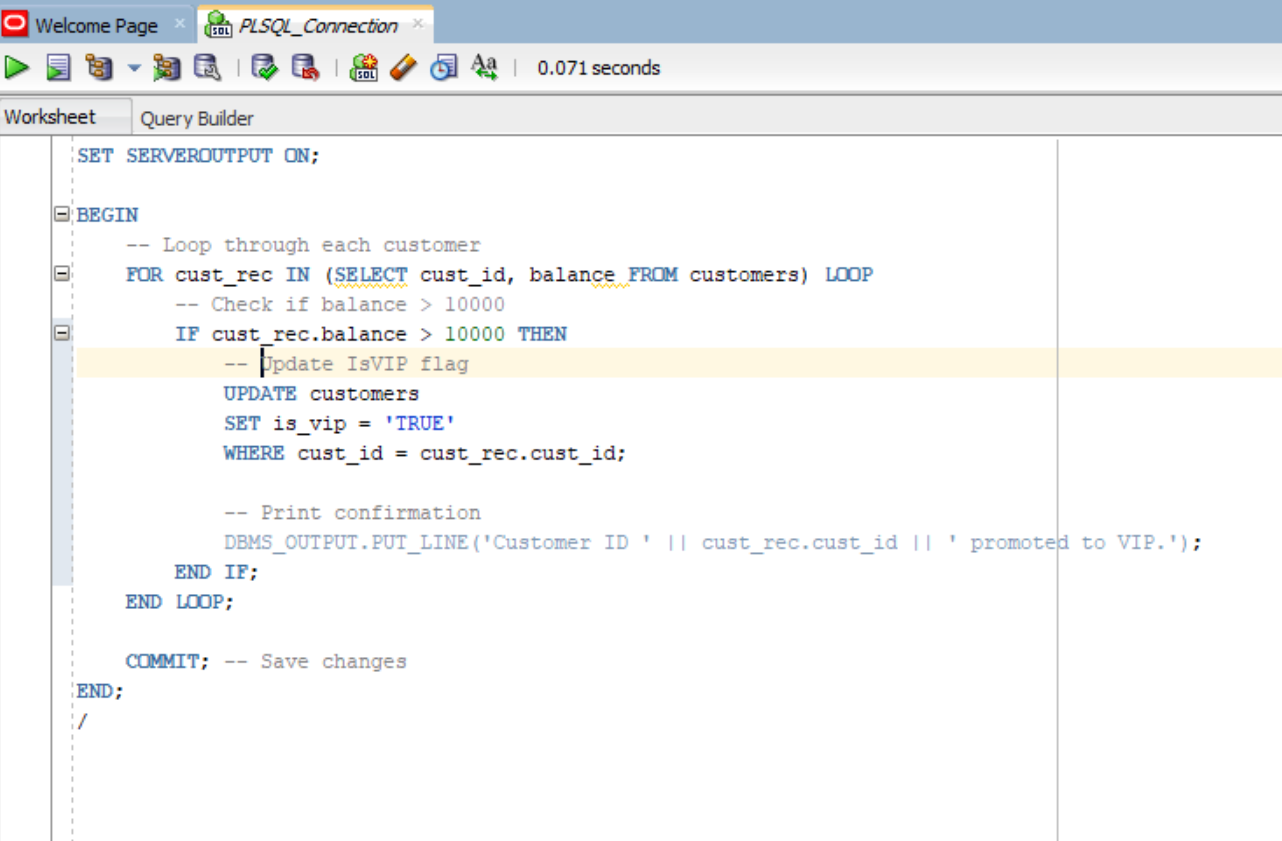
Customers with loans ID 101 and 103 are aged 65 and 70, respectively, so we are offering a discount of 1% to them.

**Scenario 2:**

A customer can be promoted to VIP status based on their balance.

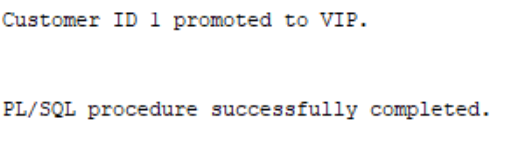
**Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

**Code:**



We can see in the sample database, User with ID 101 have about 12000 in his account. He is a user have balance over $10,000, thus, the script written will make him a VIP.

**Output:**

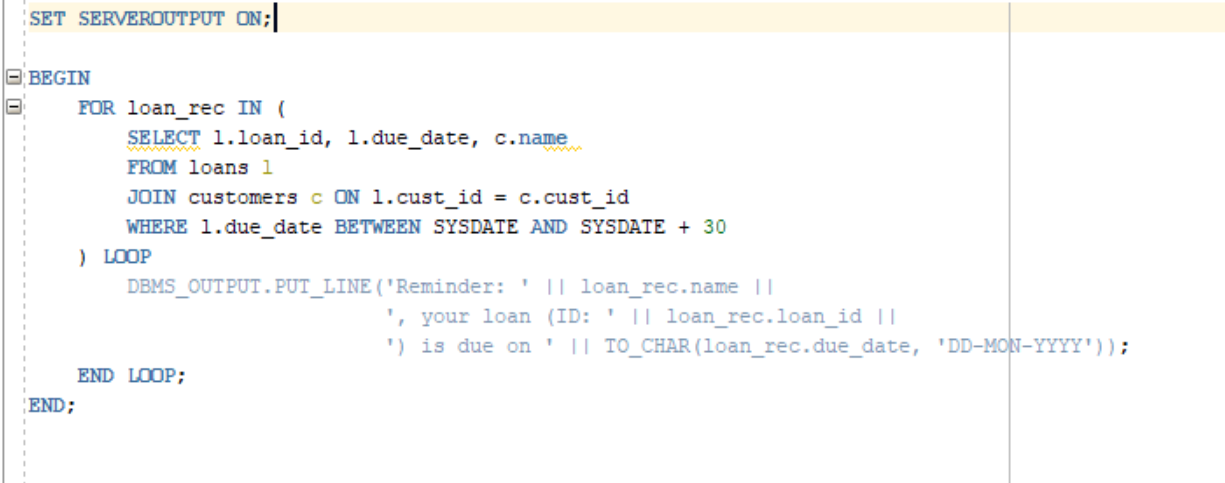


**Scenario 3:**

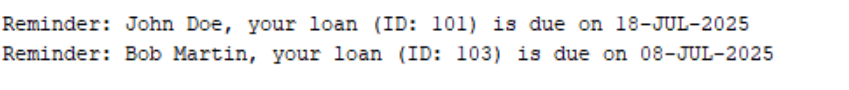
The bank wants to send reminders to customers whose loans are due within the next 30 days.

**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:**



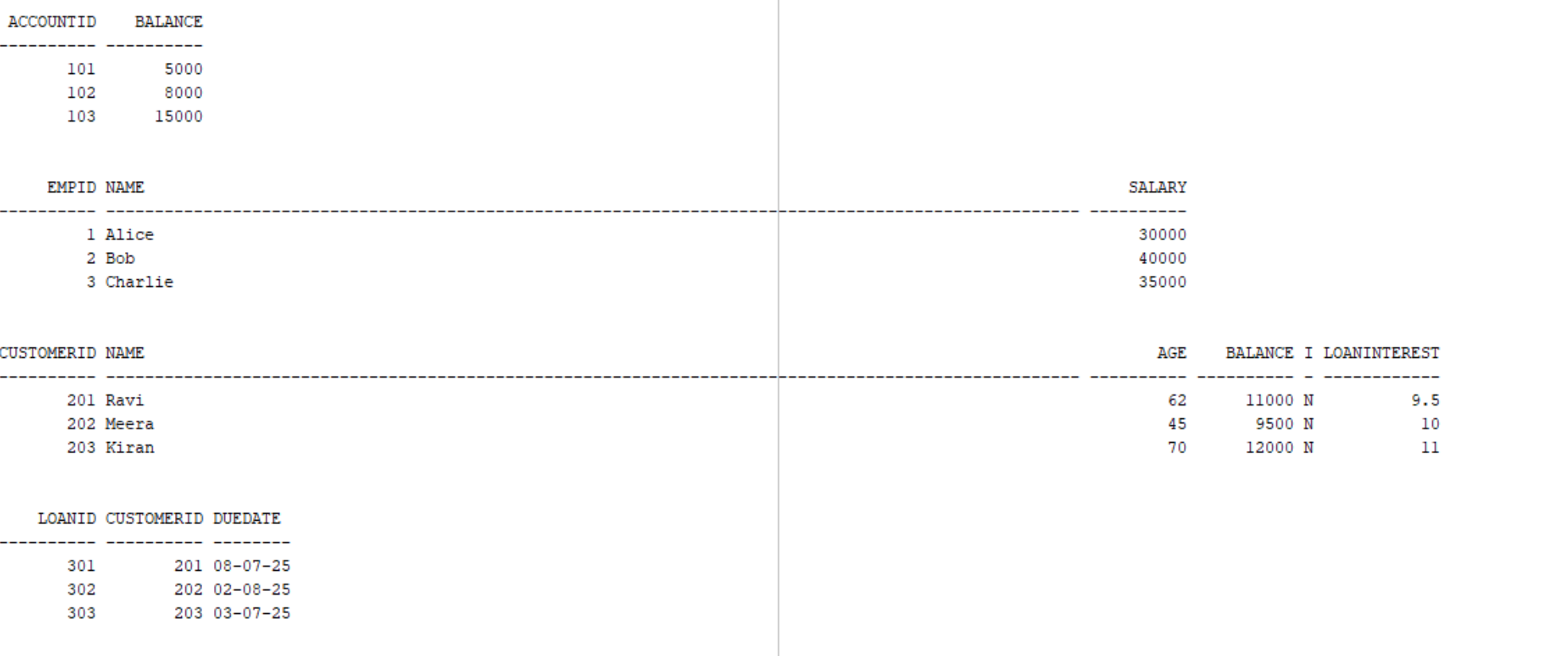
**Output:**



The motive is to print a remainder to people whose loans are due next month.

**Exercise 2: Error Handling**

Firstly, let’s create the required table and insert the sample data.



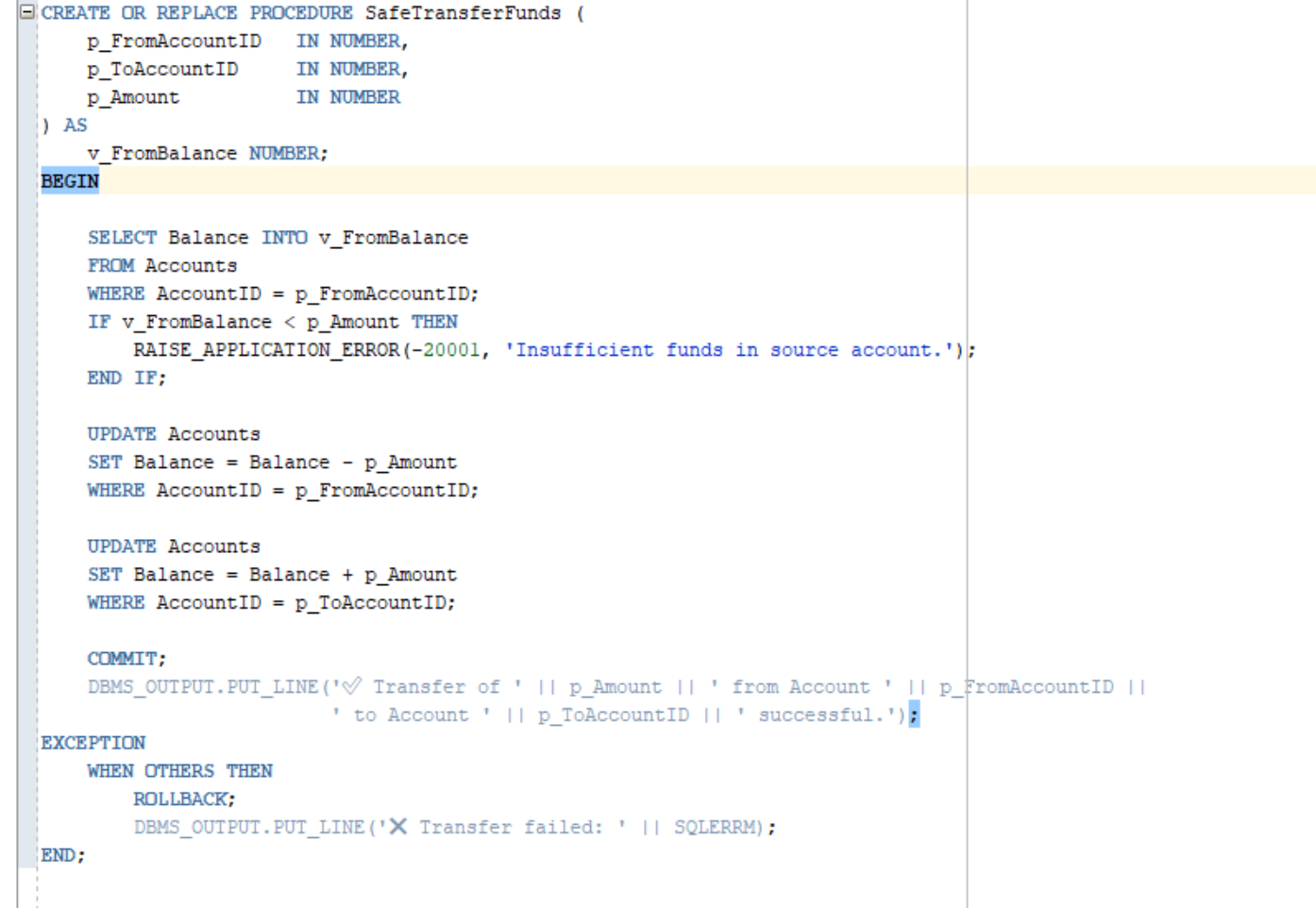
**Scenario 1:**

Handle exceptions during fund transfers between accounts.

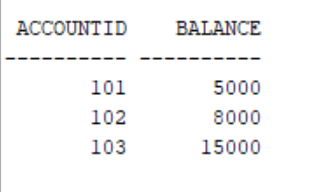
**Question:**

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.

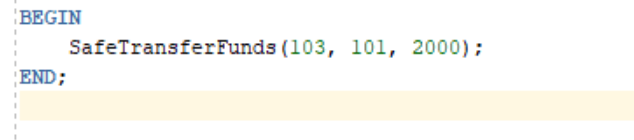
**Code:**

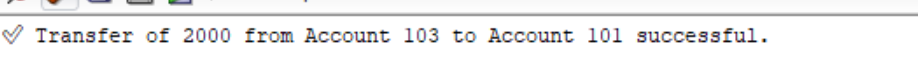


Original Accounts table:

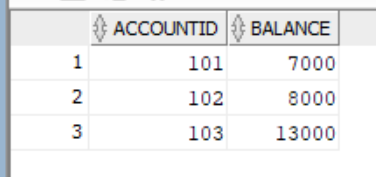


Now let’s see if this stored procedure works right:

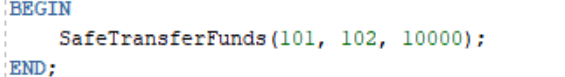




Let’s see the accounts table again:



Let’s see a case where the transaction fails:



Here 101 has only 7000, it can’t transfer 10000.

**Output:**



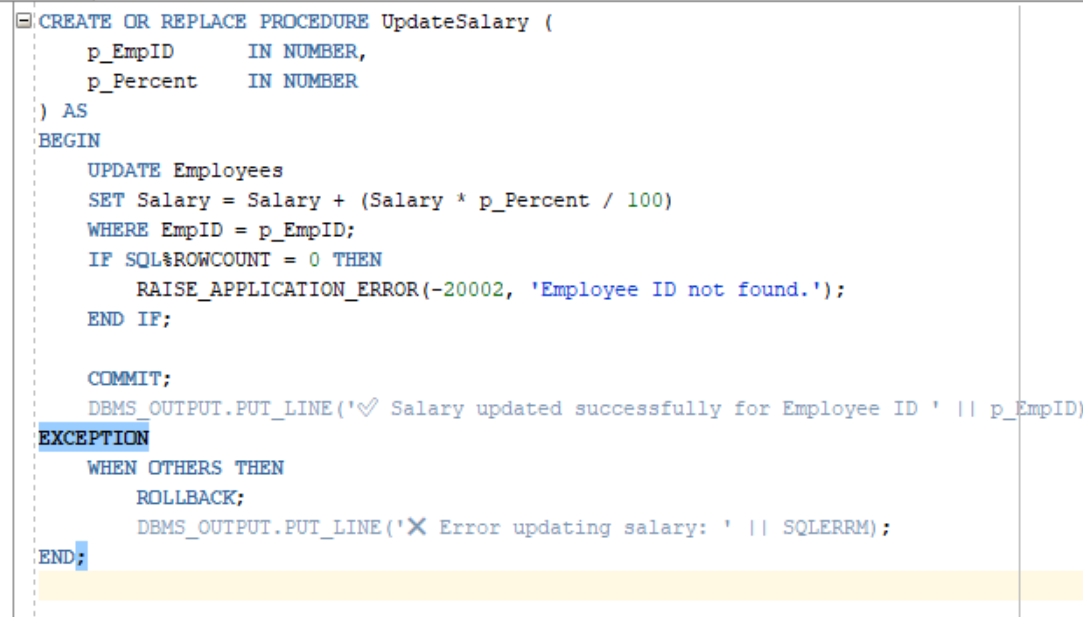
**Scenario 2:**

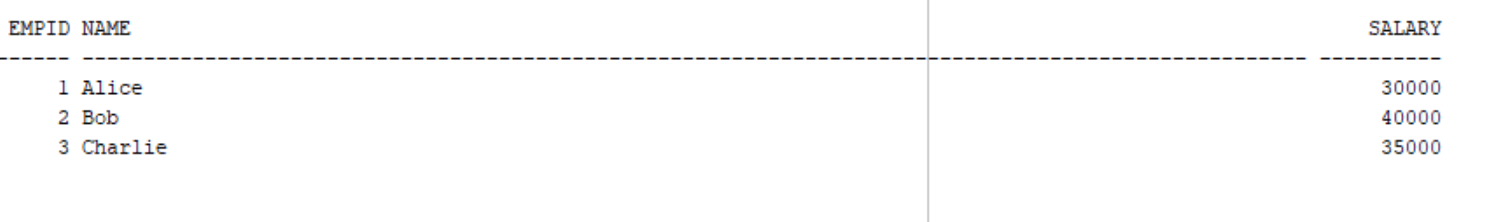
Manage errors when updating employee salaries.

**Question:**

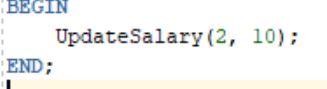
Write a stored procedure **UpdateSalary** that increases the salary of an employee by a given percentage. If the employee ID does not exist, handle the exception and log an error message.

**Code:**



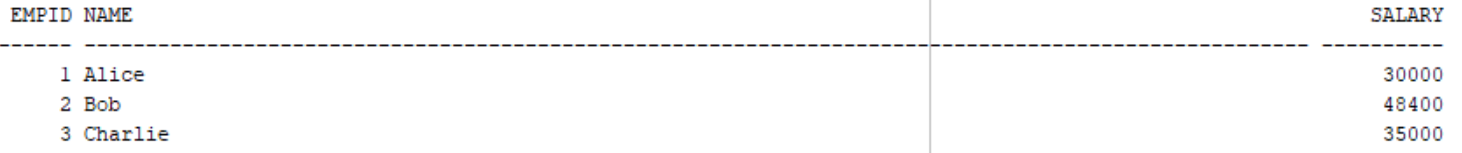
**Actual employee table:**

Let’s check its working:



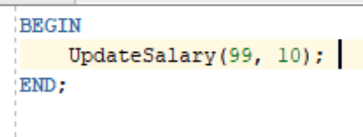
**Output:**

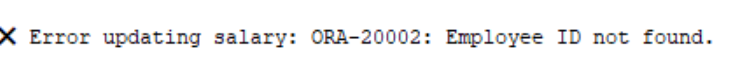




You can se that employee Bob with empId “2” has his salary increased by 10%.

Let’s see a case where the empId is not found:

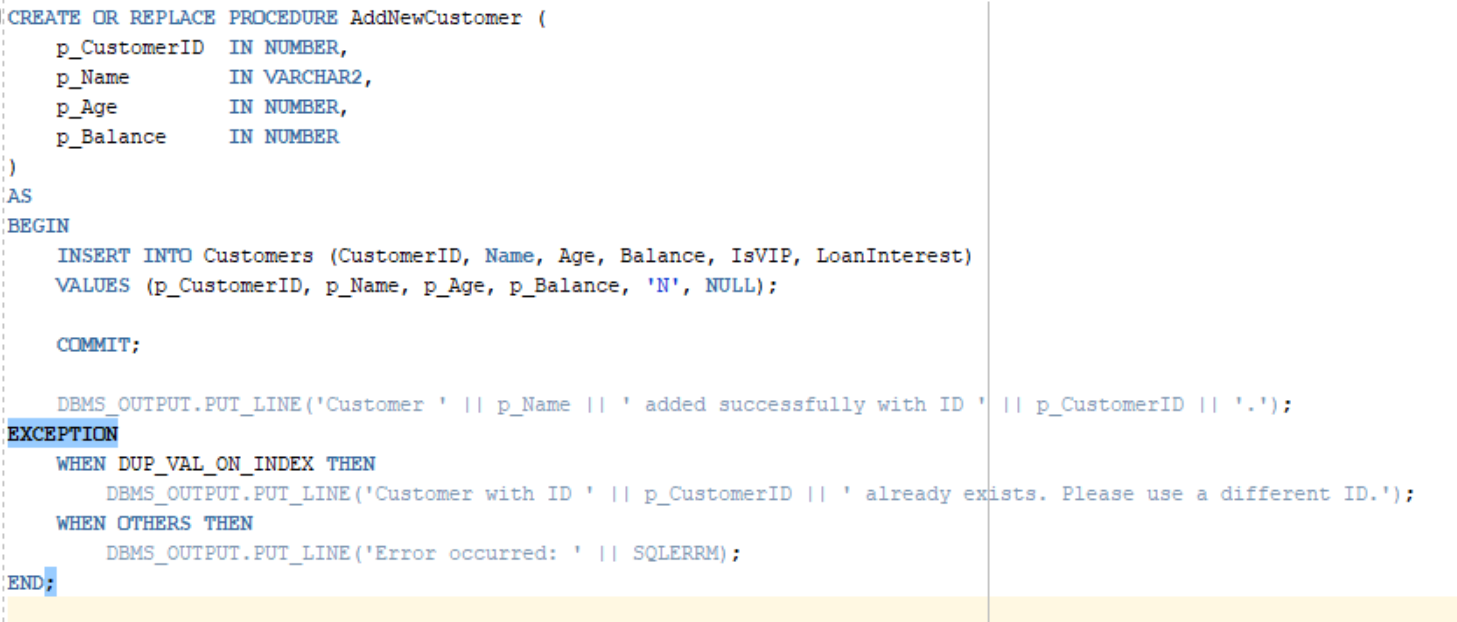




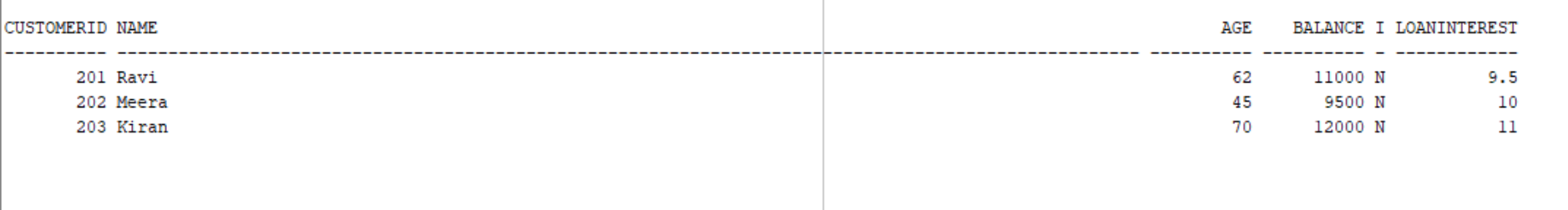
**Scenario 3:**

Ensure data integrity when adding a new customer.

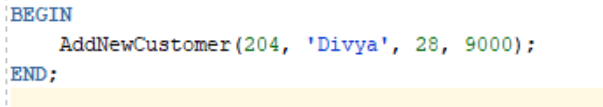
**Question:**

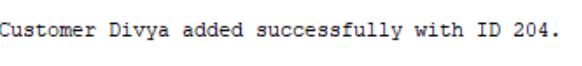
Write a stored procedure **AddNewCustomer** that inserts a new customer into the Customers table. If a customer with the same ID already exists, handle the exception by logging an error and preventing the insertion.  
**Code:**

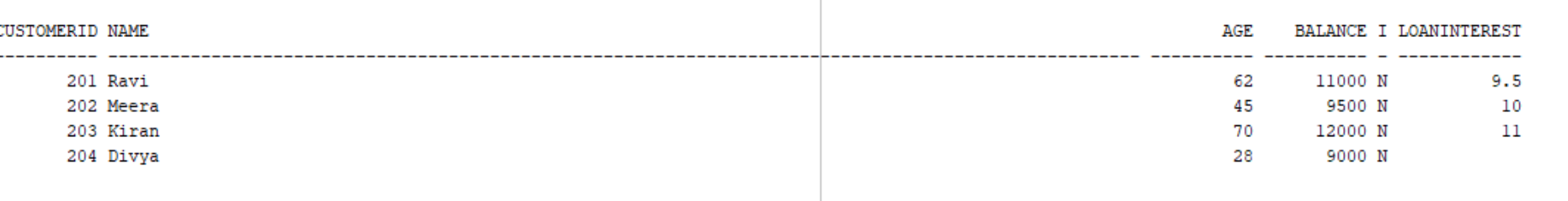
**Actual customer table:**



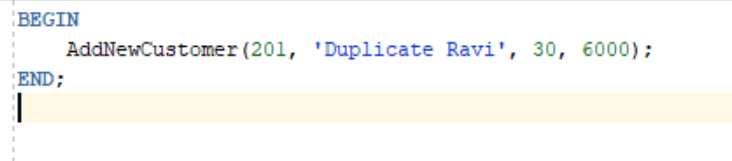
**We added a new customer:**







Let’s see how the code work in case of duplicate customer Id:





**Exercise 3: Stored Procedures**

Let’s create the required tables first:



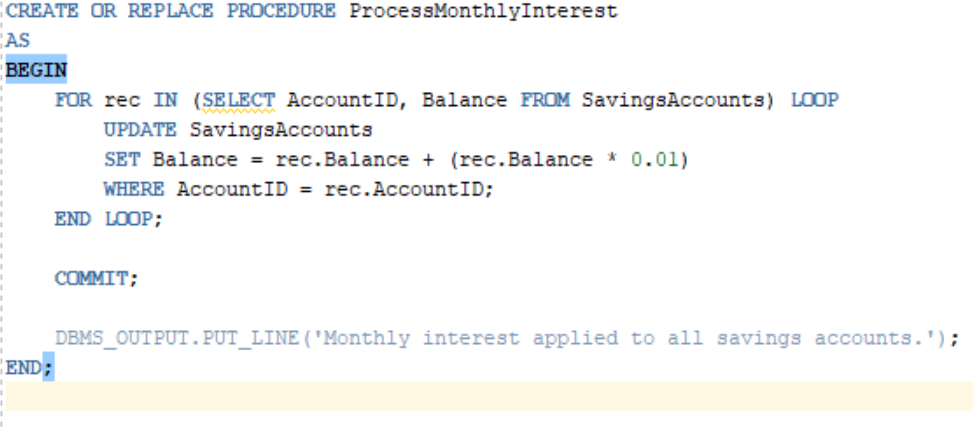
**Scenario 1:**

The bank needs to process monthly interest for all savings accounts.

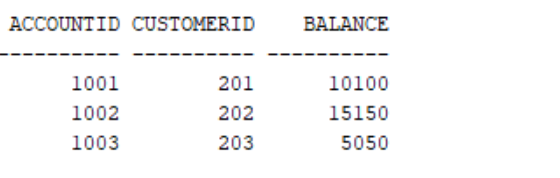
**Question:**

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:**



we run the code. Let’s see the changes in the table:



We used this stored procedure to update the balance in-place with 1% interest applied to all records in the table.

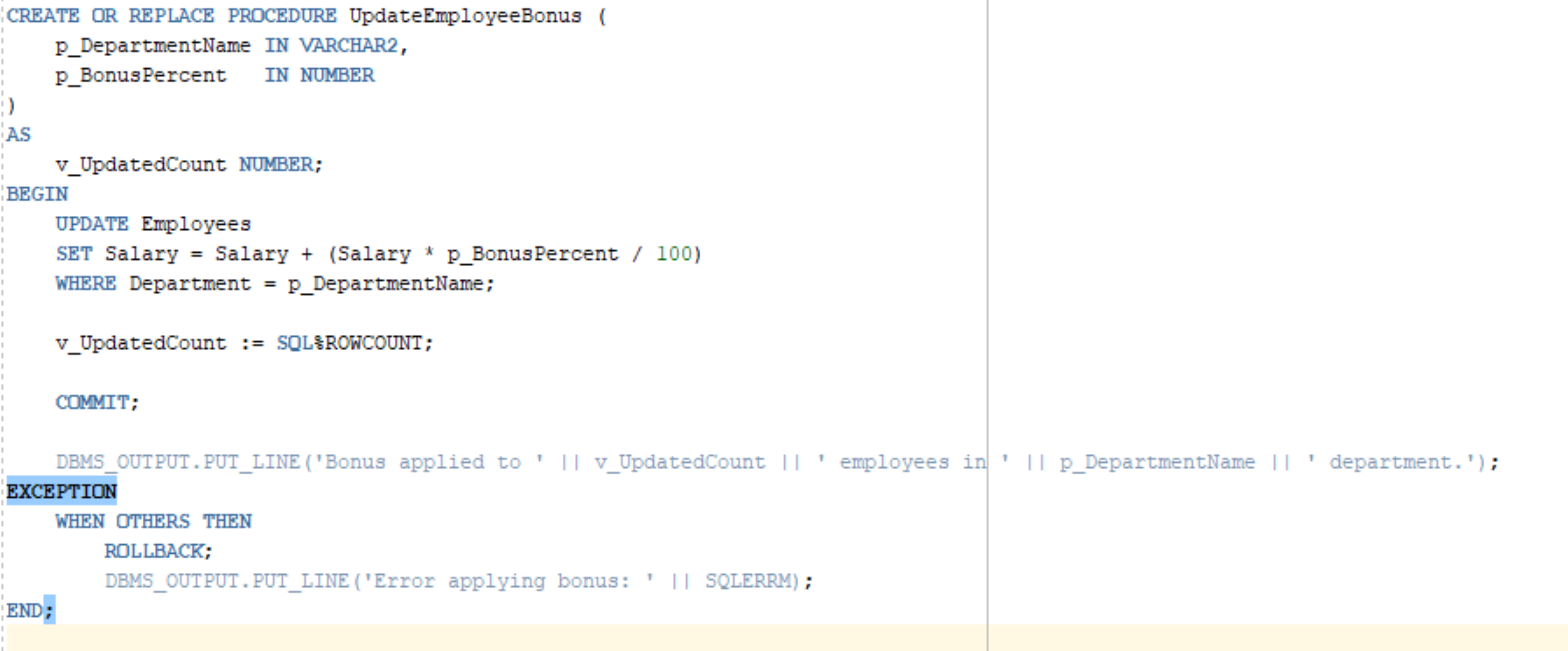
**Scenario 2:**

The bank wants to implement a bonus scheme for employees based on their performance.

**Question:**

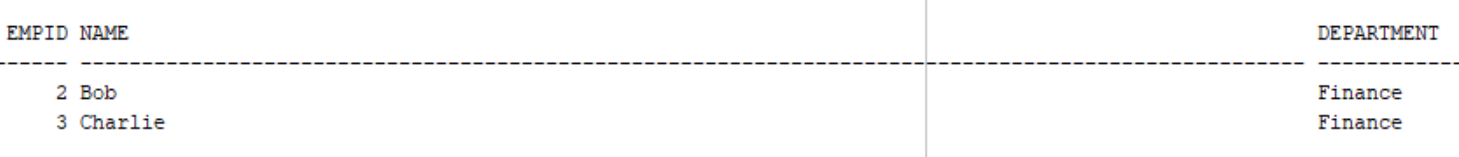
Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**Code:**



We run the script to make changes in the table. Here is the changes made. The original salaries are increased by 10%

**Output:**



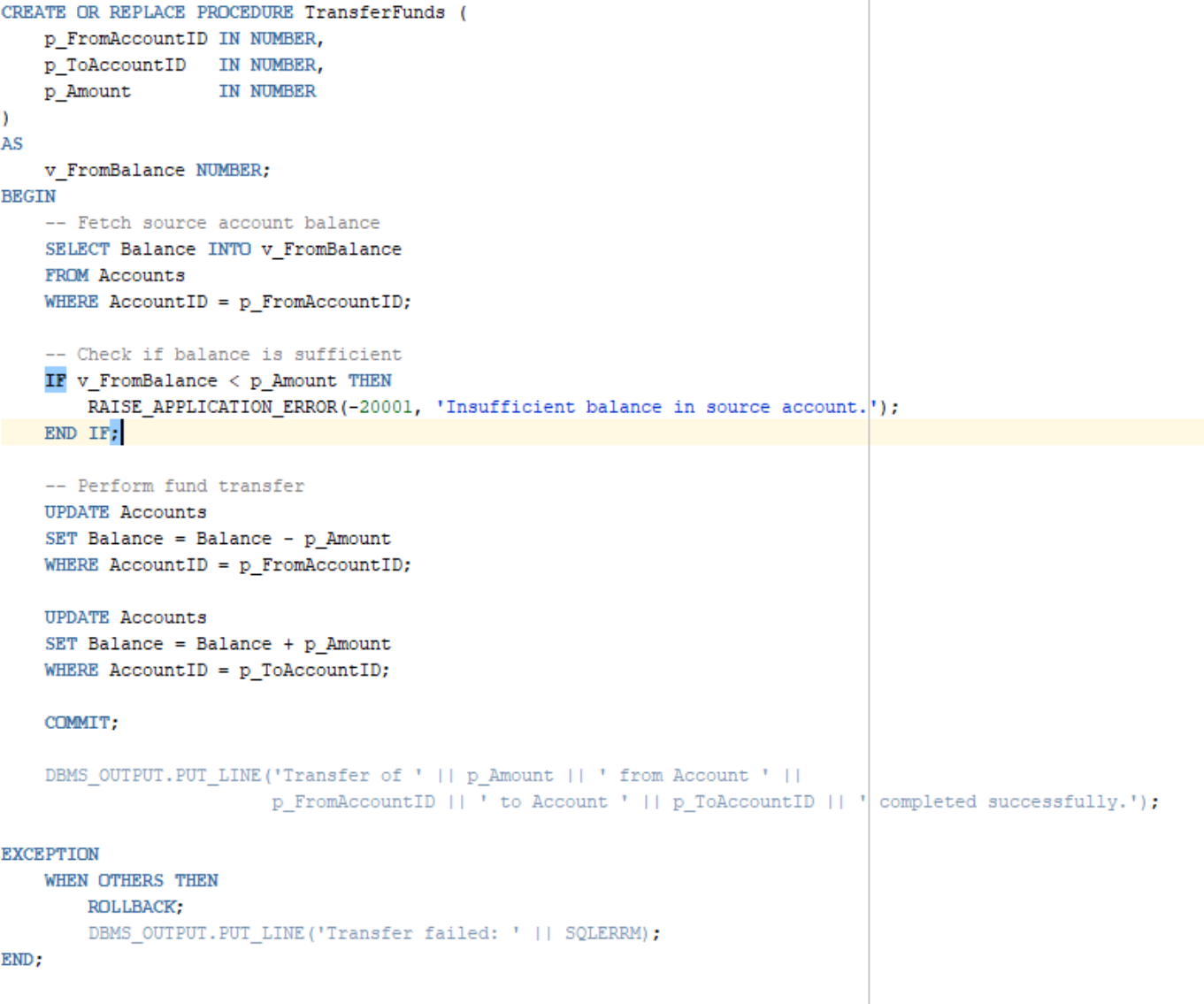
**Scenario 3:**

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:**



**Output:**

