**Exercise 1: PL/SQL Control Structures**

**Scenario 1: Apply 1% Interest Discount for Customers Above 60**

**Objective:**

To apply a **1% discount** to the loan interest rate for customers who are **older than 60 years**, using a loop and conditional logic based on the Customers and Loans tables.

**PL/SQL Code:**

CREATE OR REPLACE PROCEDURE ApplySeniorDiscount AS

BEGIN

FOR rec IN (

SELECT l.LoanID, c.DOB

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

) LOOP

IF MONTHS\_BETWEEN(SYSDATE, rec.DOB) / 12 > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = rec.LoanID;

END IF;

END LOOP;

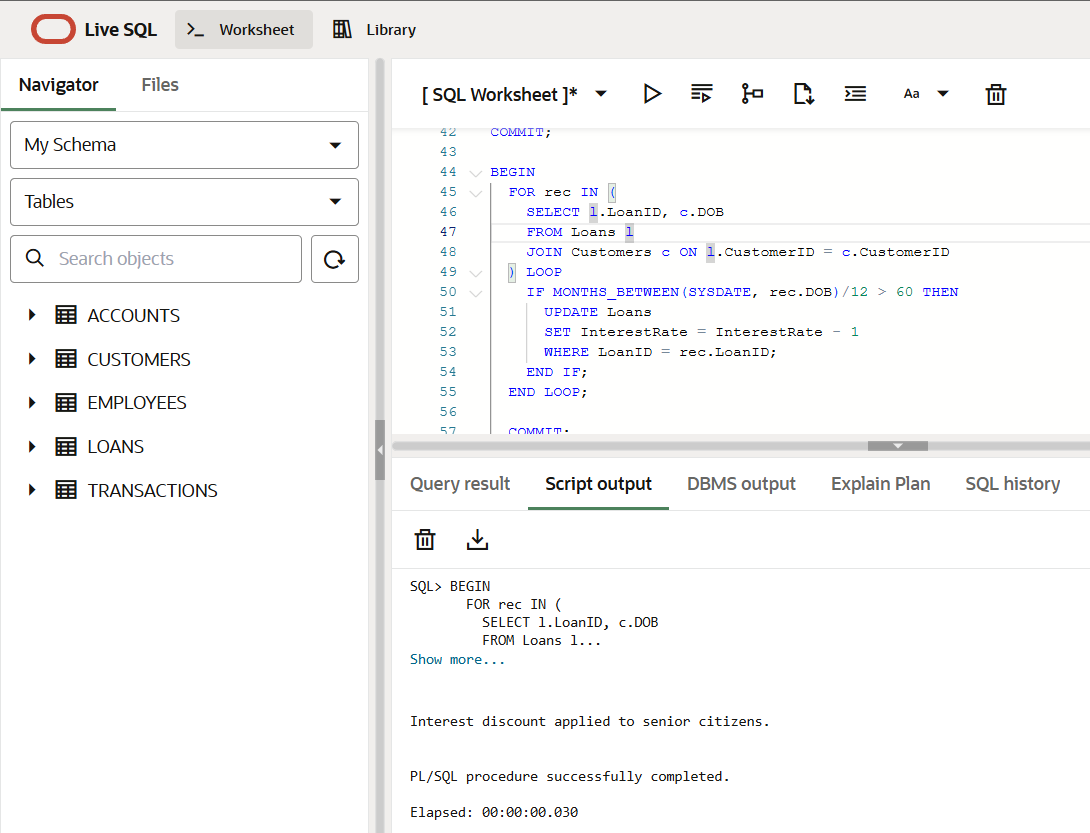
COMMIT;

DBMS\_OUTPUT.PUT\_LINE('1% discount applied for customers over 60.');

END;

EXEC ApplySeniorDiscount;

**Output:**



**Scenario 2: Promote Customers to VIP Based on Balance**

**Objective:**

To promote customers to **VIP status** by setting IsVIP = 'TRUE' for those having a **balance greater than $10,000** using a loop and condition.

**PL/SQL Code:**

BEGIN

FOR rec IN (SELECT CustomerID, Balance FROM Customers) LOOP

IF rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

END IF;

END LOOP;

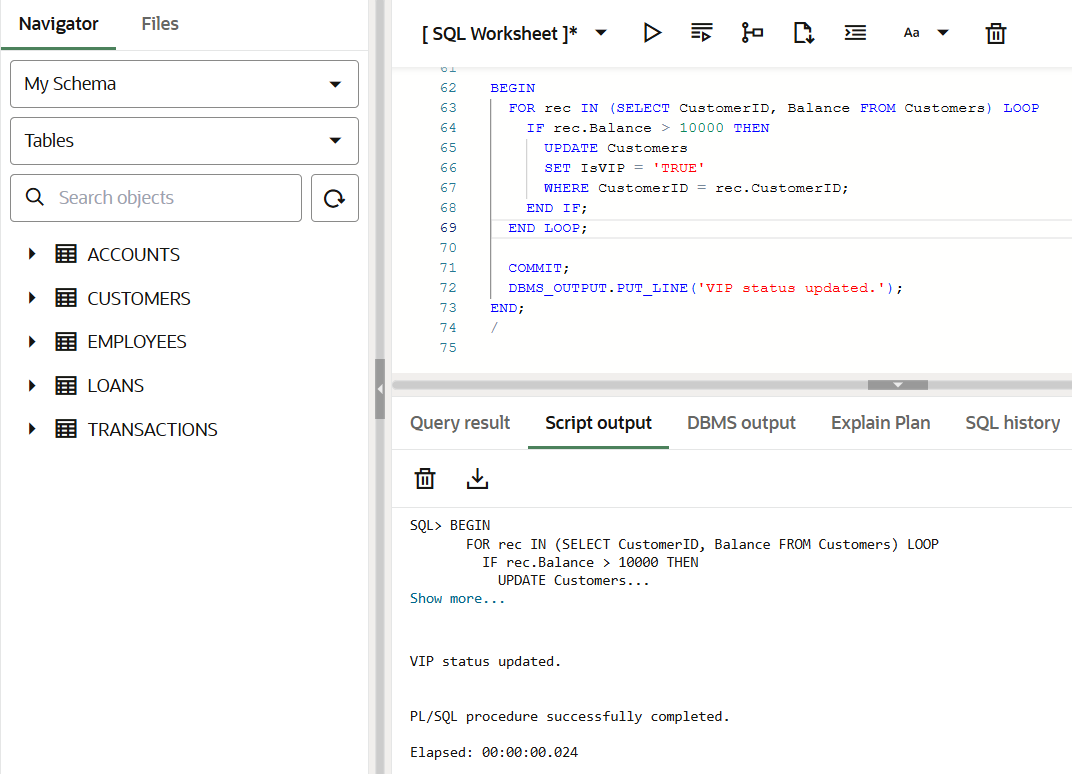
COMMIT;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated.');

END;

/

**Output:**



**Scenario 3: Send Loan Due Reminders Within 30 Days**

**Objective:**

To **print reminder messages** for all customers whose loans are **due within the next 30 days**, using a loop and SYSDATE check.

**PL/SQL Code:**

BEGIN

FOR rec IN (

SELECT c.Name, l.EndDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate <= SYSDATE + 30

) LOOP

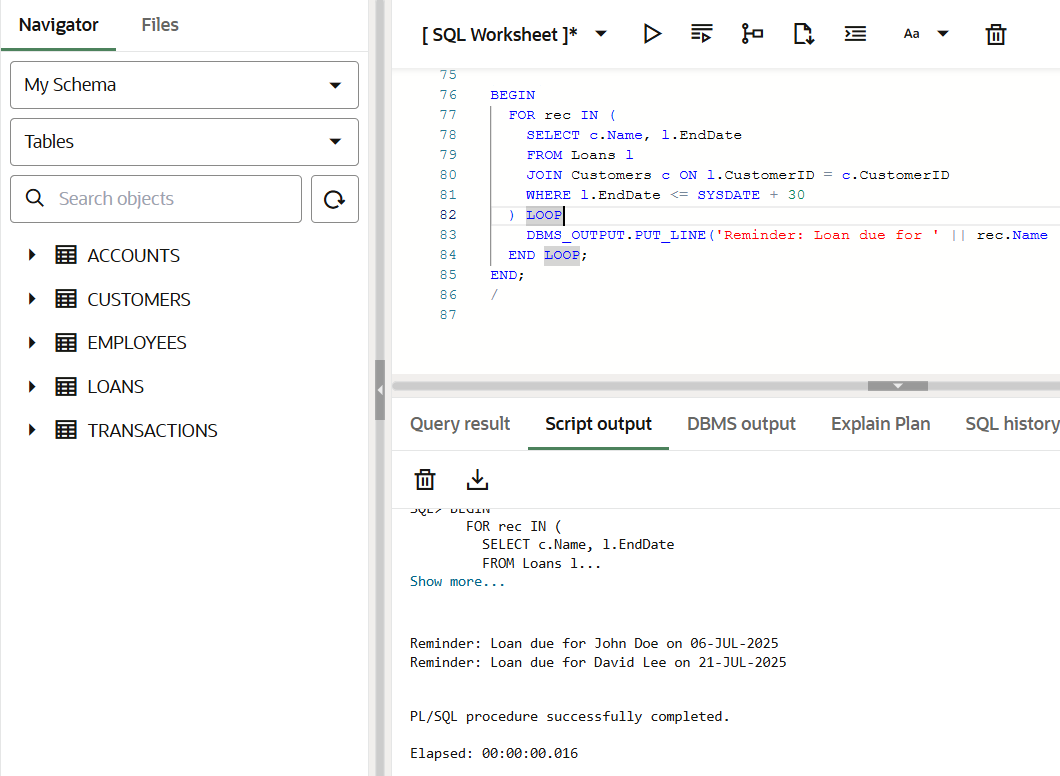
DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan due for ' || rec.Name || ' on ' || TO\_CHAR(rec.EndDate, 'DD-MON-YYYY'));

END LOOP;

END;

/

**Output:**



**Exercise 3: Stored Procedures (Using Given Tables)**

**Scenario 1: Process Monthly Interest for Savings Accounts**

**Objective:**

Create a stored procedure ProcessMonthlyInterest that updates the **balance of all 'Savings' accounts** by applying **1% monthly interest** using your Accounts table.

**PL/SQL Code:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR rec IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

SET Balance = Balance + (rec.Balance \* 0.01),

LastModified = SYSDATE

WHERE AccountID = rec.AccountID;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('1% monthly interest applied to savings accounts.');

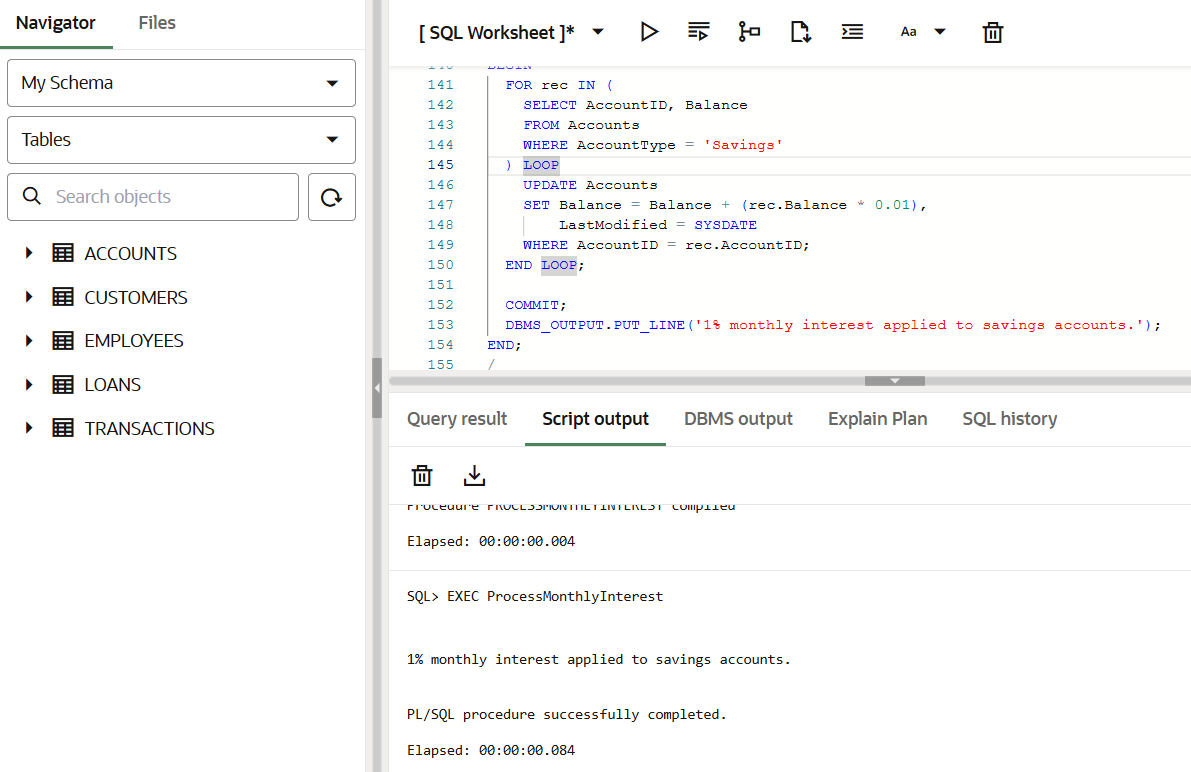
END;

/

**Execution:**

EXEC ProcessMonthlyInterest;

**Output:**



**Scenario 2: Update Employee Bonus by Department**

**Objective:**

Create a stored procedure UpdateEmployeeBonus that **increases the salary** of all employees in a **given department** by a **bonus percentage** passed as a parameter.

**PL/SQL Code:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE Department = dept\_name;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || bonus\_percent || '% applied to department: ' || dept\_name);

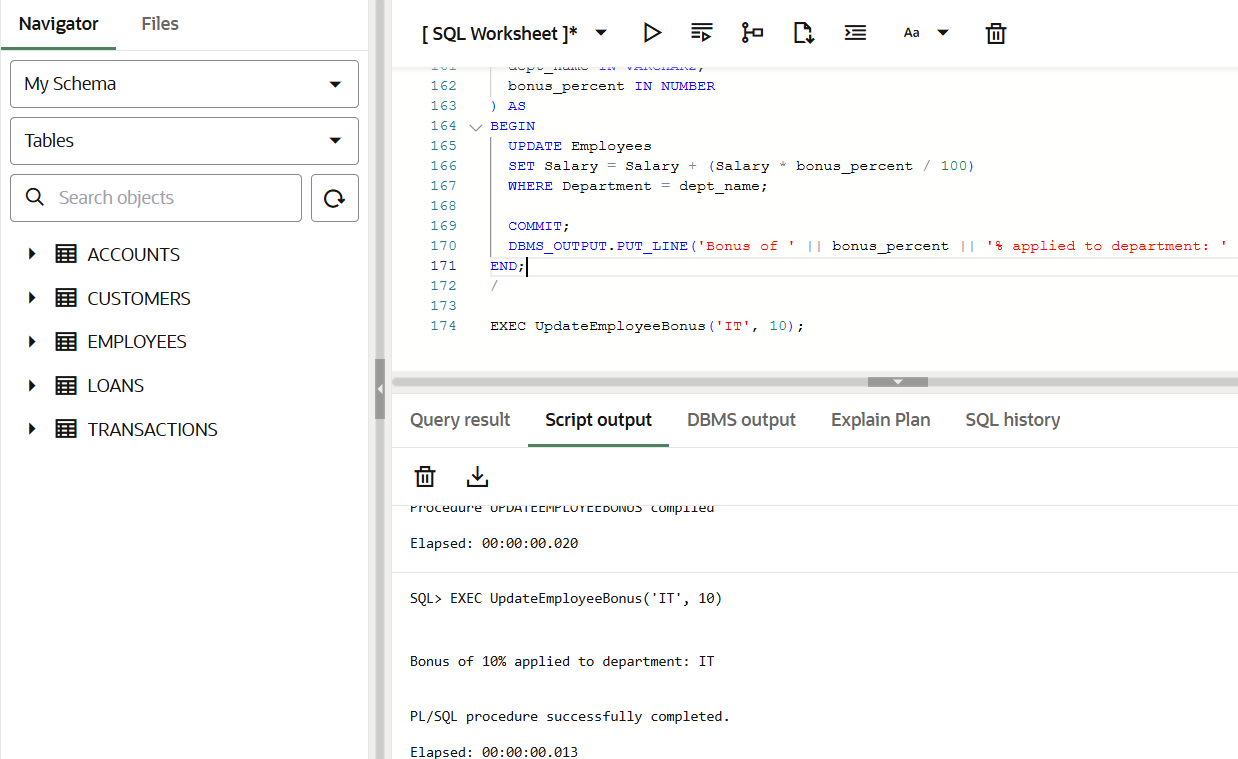
END;

/

**Execution:**

EXEC UpdateEmployeeBonus('IT', 10);

**Output:**



**Scenario 3: Transfer Funds Between Accounts**

**Objective:**

Create a stored procedure TransferFunds that transfers funds between two accounts, ensuring **sufficient balance** in the source account.

**PL/SQL Code:**

CREATE OR REPLACE PROCEDURE TransferFunds(

from\_acc IN NUMBER,

to\_acc IN NUMBER,

amount IN NUMBER

) AS

from\_balance NUMBER;

BEGIN

-- Get balance from source account

SELECT Balance INTO from\_balance

FROM Accounts

WHERE AccountID = from\_acc;

IF from\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient balance in Account ID ' || from\_acc);

RETURN;

END IF;

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - amount,

LastModified = SYSDATE

WHERE AccountID = from\_acc;

-- Add to target

UPDATE Accounts

SET Balance = Balance + amount,

LastModified = SYSDATE

WHERE AccountID = to\_acc;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || amount || ' from Account ' || from\_acc || ' to Account ' || to\_acc);

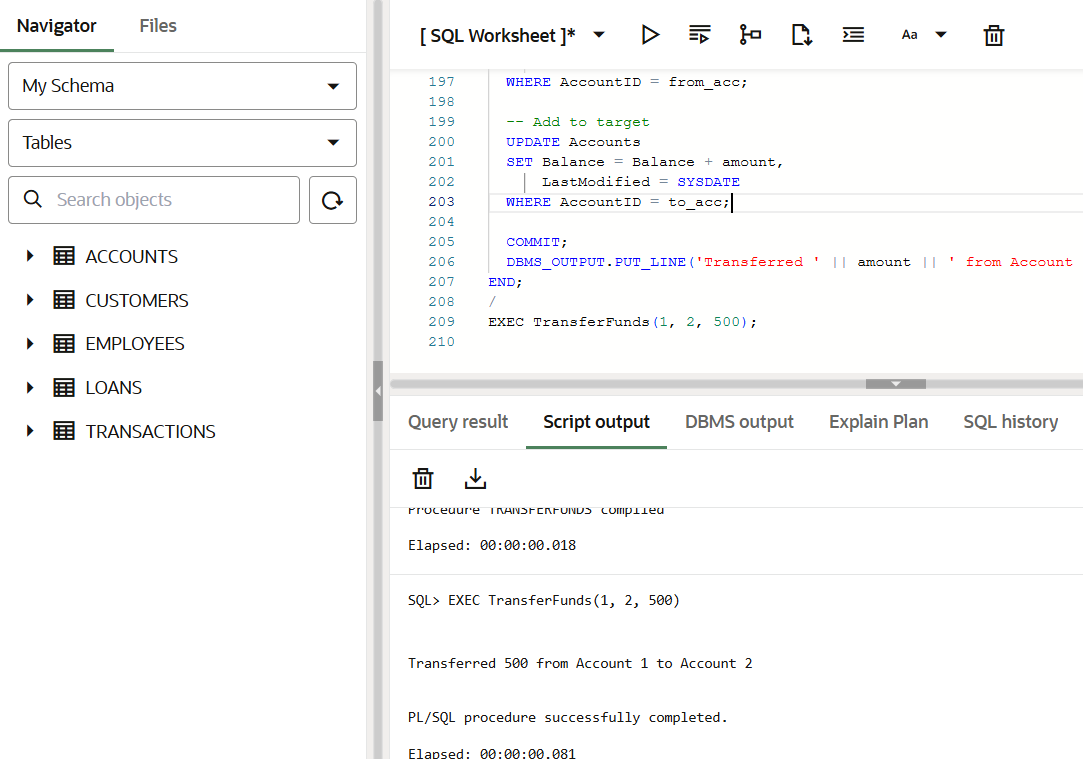
END;

/

**Execution:**

EXEC TransferFunds(1, 2, 500);

**Output:**



**Additional Setup Used:**

**Tables:**

*CREATE TABLE Customers (*

*CustomerID NUMBER PRIMARY KEY,*

*Name VARCHAR2(100),*

*DOB DATE,*

*Balance NUMBER,*

*LastModified DATE*

*);*

*CREATE TABLE Accounts (*

*AccountID NUMBER PRIMARY KEY,*

*CustomerID NUMBER,*

*AccountType VARCHAR2(20),*

*Balance NUMBER,*

*LastModified DATE,*

*FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)*

*);*

*CREATE TABLE Transactions (*

*TransactionID NUMBER PRIMARY KEY,*

*AccountID NUMBER,*

*TransactionDate DATE,*

*Amount NUMBER,*

*TransactionType VARCHAR2(10),*

*FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)*

*);*

*CREATE TABLE Loans (*

*LoanID NUMBER PRIMARY KEY,*

*CustomerID NUMBER,*

*LoanAmount NUMBER,*

*InterestRate NUMBER,*

*StartDate DATE,*

*EndDate DATE,*

*FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)*

*);*

*CREATE TABLE Employees (*

*EmployeeID NUMBER PRIMARY KEY,*

*Name VARCHAR2(100),*

*Position VARCHAR2(50),*

*Salary NUMBER,*

*Department VARCHAR2(50),*

*HireDate DATE*

*);*

**Sample Data Inserted:**

*INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)*

*VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);*

*INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)*

*VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);*

*INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)*

*VALUES (1, 1, 'Savings', 1000, SYSDATE);*

*INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)*

*VALUES (2, 2, 'Checking', 1500, SYSDATE);*

*INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)*

*VALUES (1, 1, SYSDATE, 200, 'Deposit');*

*INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)*

*VALUES (2, 2, SYSDATE, 300, 'Withdrawal');*

*INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)*

*VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));*

*INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)*

*VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));*

*INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)*

*VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));*