**PL/SQL PROGRAMMING**

**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:**

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),

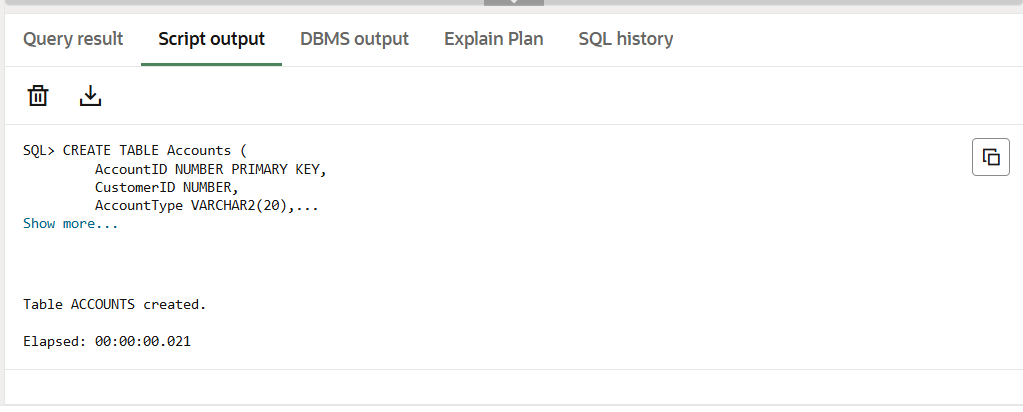
Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

**OUTPUT:**



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

VALUES (1, 'John Doe', TO\_DATE('1950-05-15', 'YYYY-MM-DD'), 5000, SYSDATE);

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

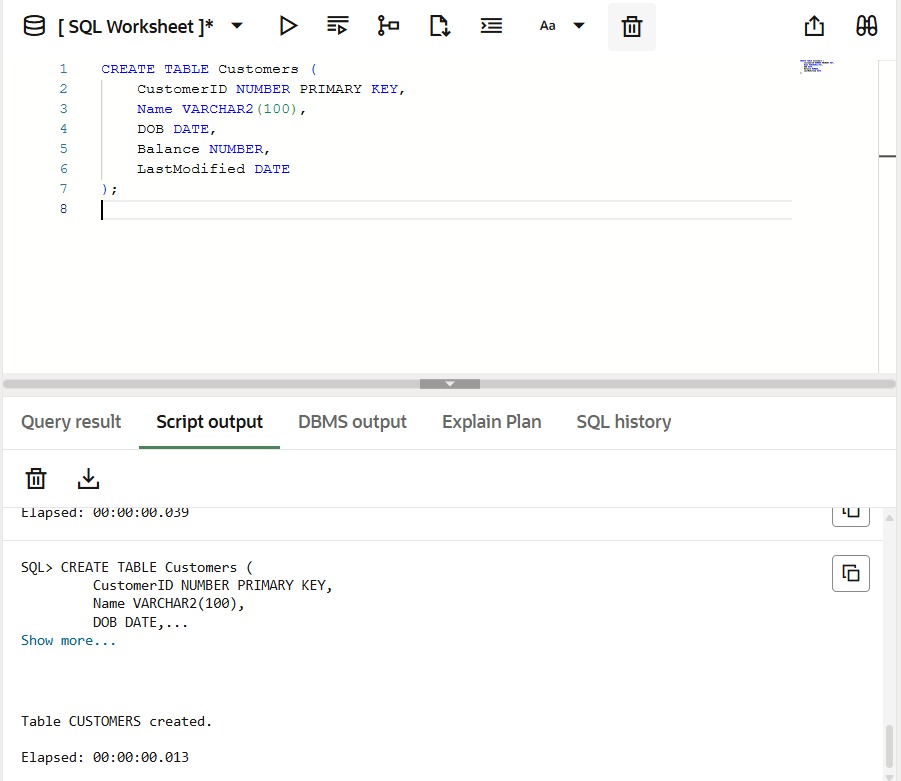
VALUES (2, 'Jane Smith', TO\_DATE('1985-07-20', 'YYYY-MM-DD'), 7000, SYSDATE);

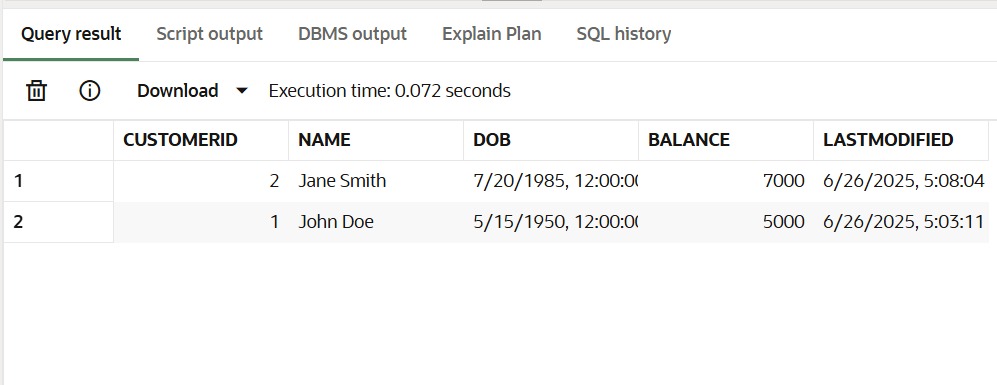
INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

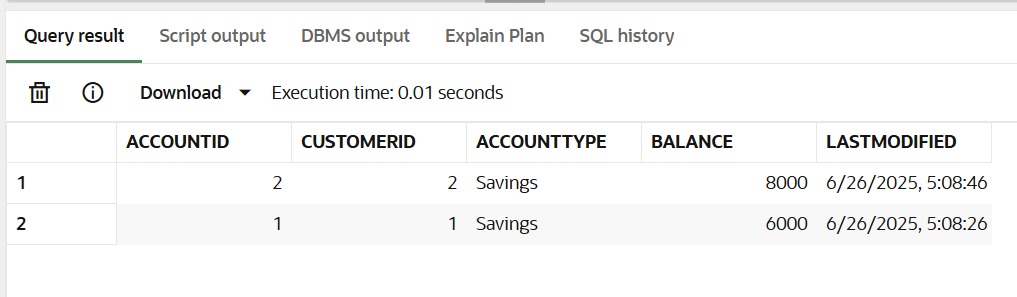
VALUES (1, 1, 'Savings', 6000, SYSDATE);

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

VALUES (2, 2, 'Savings', 8000, SYSDATE);

**OUTPUT:**





CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE Accounts

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

LastModified = SYSDATE

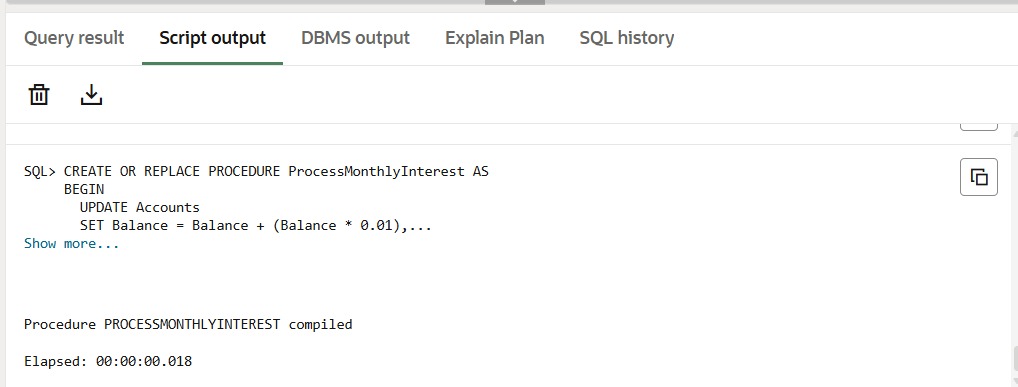
WHERE AccountType = 'Savings';

COMMIT;

END;

/

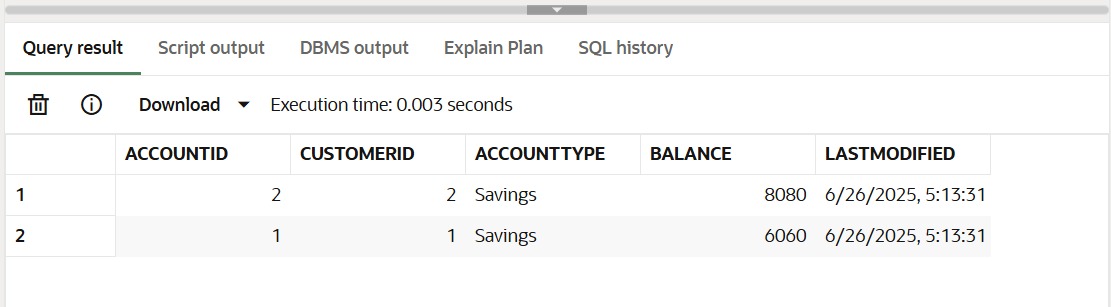
**OUTPUT:**



EXEC ProcessMonthlyInterest;

SELECT \* FROM Accounts;

**OUTPUT:**



**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.**

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

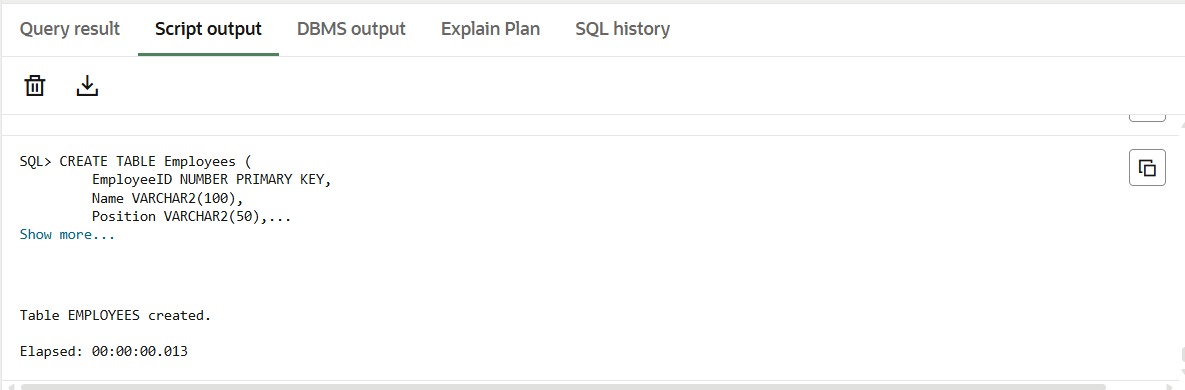
Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

**OUTPUT:**



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

VALUES (1, 'Alice', 'Manager', 60000, 'HR', TO\_DATE('2018-06-01', 'YYYY-MM-DD'));

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

VALUES (2, 'Bob', 'Developer', 50000, 'IT', TO\_DATE('2019-09-01', 'YYYY-MM-DD'));

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

VALUES (3, 'john', 'Manager', 60000, 'IT', TO\_DATE('2018-06-01', 'YYYY-MM-DD'));

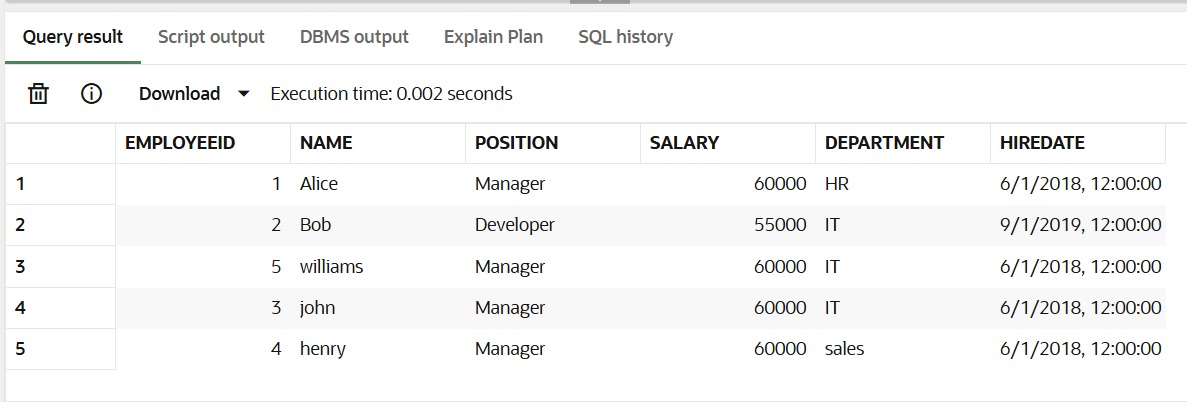
INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (4, 'henry', 'Developer', 60000, 'sales', TO\_DATE('2019-09-01', 'YYYY-MM-DD'));

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

VALUES (5, 'williams', 'Developer', 60000, 'IT', TO\_DATE('2019-09-01', 'YYYY-MM-DD'));

**OUTPUT:**



CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE Employees

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

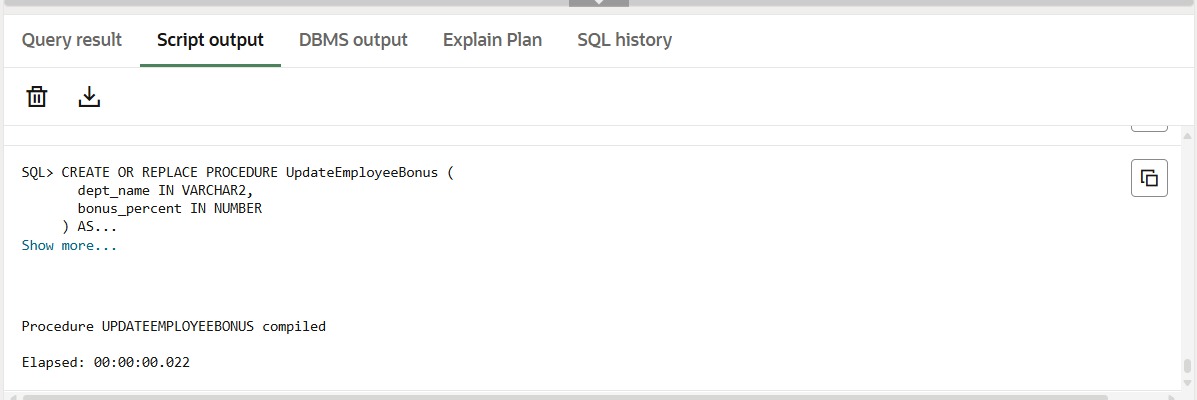
WHERE Department = dept\_name;

COMMIT;

END;

/

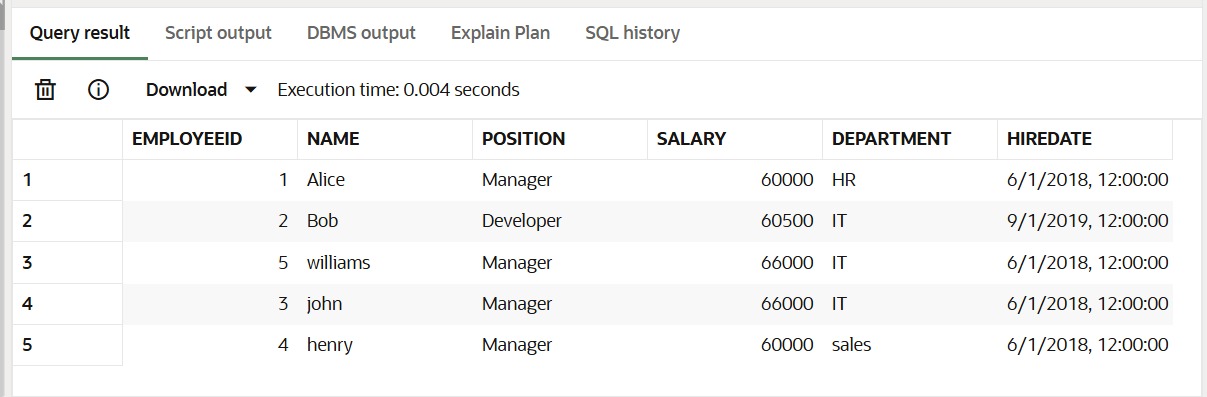
**OUTPUT:**



EXEC UpdateEmployeeBonus('IT', 10);

SELECT \* FROM Employees;

**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:**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

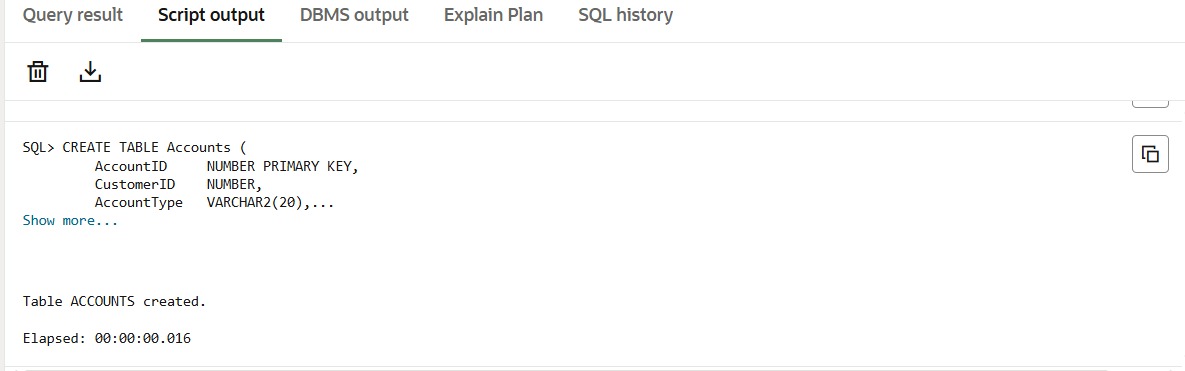
Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

**OUTPUT:**



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

VALUES (1, 1, 'Savings', 5000, SYSDATE);

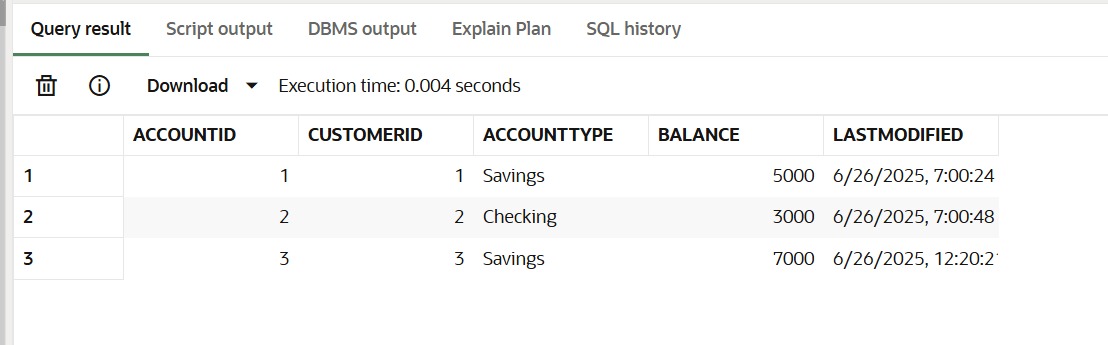
INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checkings', 3000, SYSDATE);

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

VALUES (3, 3, 'Savings', 7000, SYSDATE);

**OUTPUT:**



CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

transfer\_amount IN NUMBER

) AS

from\_balance NUMBER;

BEGIN

SELECT Balance INTO from\_balance

FROM Accounts

WHERE AccountID = from\_account\_id;

IF from\_balance >= transfer\_amount THEN

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - transfer\_amount,

LastModified = SYSDATE

WHERE AccountID = from\_account\_id;

UPDATE Accounts

SET Balance = Balance + transfer\_amount,

LastModified = SYSDATE

WHERE AccountID = to\_account\_id;

COMMIT;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient funds in Account ID ' || from\_account\_id);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Invalid account ID');

WHEN OTHERS THEN

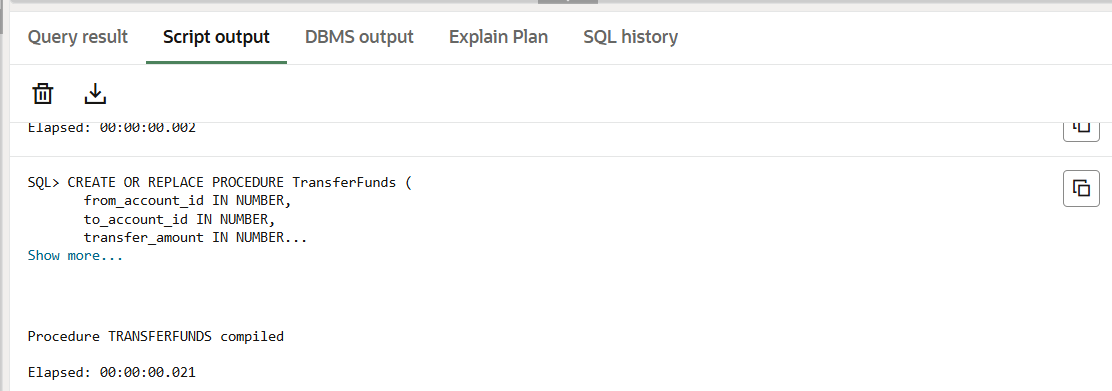
DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

ROLLBACK;

END;

/

**OUTPUT:**



EXEC TransferFunds(1, 2, 1000);

SELECT \* FROM Accounts;

**OUTPUT:**

