**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.

**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.

**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.

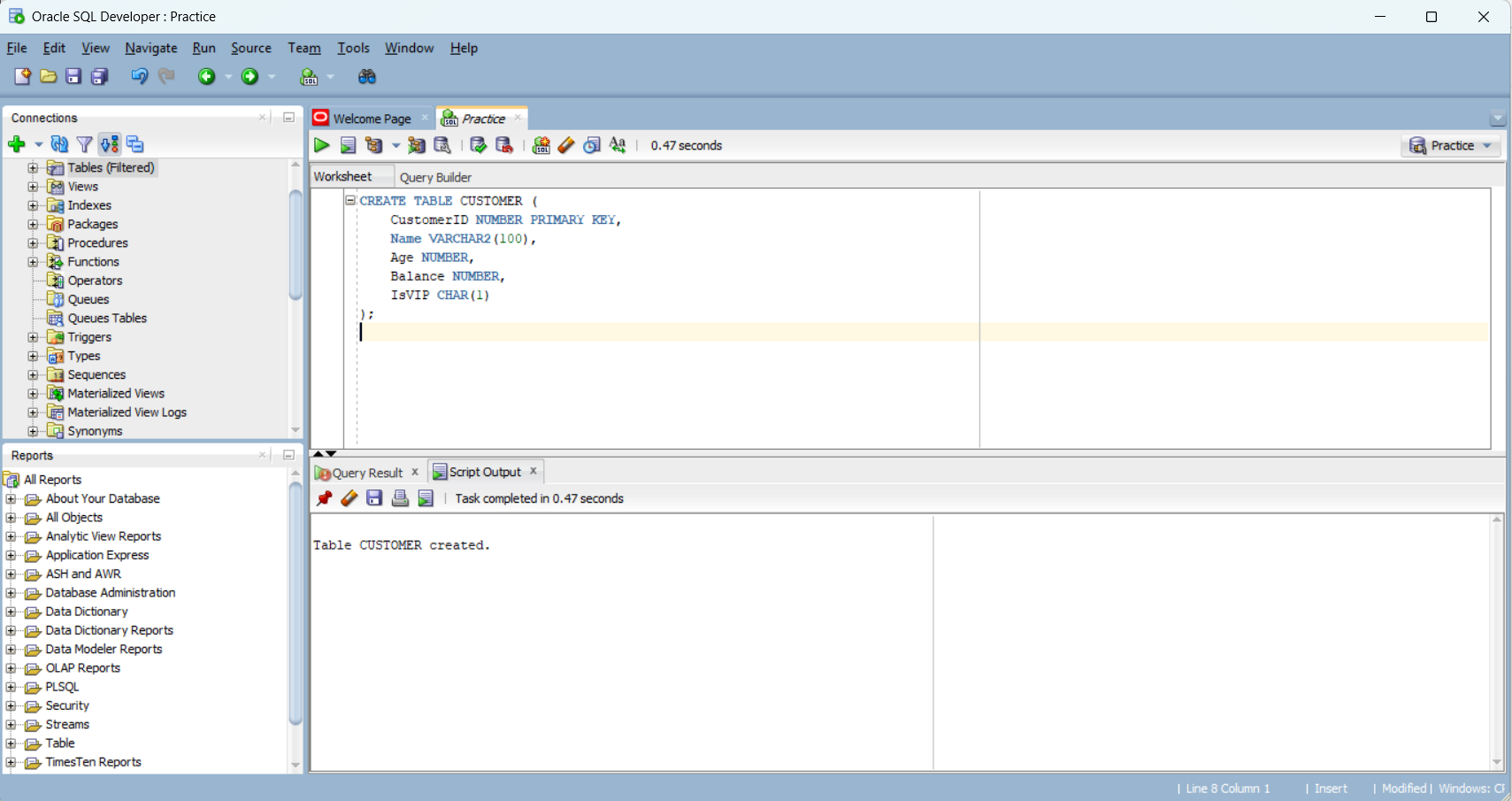
**STEP 1: CREATING TABLES AND INSERTING VALUES**

Create CUSTOMER table with CustomerID, Name, Age, Balance, IsVIP.

Query:

|  |
| --- |
| CREATE TABLE CUSTOMER (  CustomerID NUMBER PRIMARY KEY,  Name VARCHAR2(100),  Age NUMBER,  Balance NUMBER,  IsVIP CHAR(1)  ); |

Output:

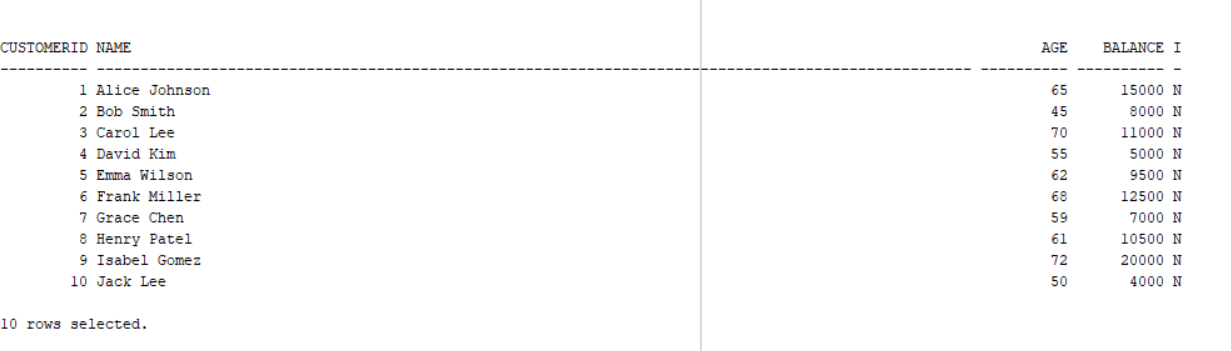


Inserting Value into CUSTOMER table.

Query:

|  |
| --- |
| INSERT INTO CUSTOMER (CustomerID, Name, Age, Balance, IsVIP) VALUES (1, 'Alice Johnson', 65, 15000, 'N');  INSERT INTO CUSTOMER (CustomerID, Name, Age, Balance, IsVIP) VALUES (2, 'Bob Smith', 45, 8000, 'N');  INSERT INTO CUSTOMER (CustomerID, Name, Age, Balance, IsVIP) VALUES (3, 'Carol Lee', 70, 11000, 'N');  INSERT INTO CUSTOMER (CustomerID, Name, Age, Balance, IsVIP) VALUES (4, 'David Kim', 55, 5000, 'N');  INSERT INTO CUSTOMER (CustomerID, Name, Age, Balance, IsVIP) VALUES (5, 'Emma Wilson', 62, 9500, 'N');  INSERT INTO CUSTOMER (CustomerID, Name, Age, Balance, IsVIP) VALUES (6, 'Frank Miller', 68, 12500, 'N');  INSERT INTO CUSTOMER (CustomerID, Name, Age, Balance, IsVIP) VALUES (7, 'Grace Chen', 59, 7000, 'N');  INSERT INTO CUSTOMER (CustomerID, Name, Age, Balance, IsVIP) VALUES (8, 'Henry Patel', 61, 10500, 'N');  INSERT INTO CUSTOMER (CustomerID, Name, Age, Balance, IsVIP) VALUES (9, 'Isabel Gomez', 72, 20000, 'N');  INSERT INTO CUSTOMER (CustomerID, Name, Age, Balance, IsVIP) VALUES (10, 'Jack Lee', 50, 4000, 'N');  COMMIT;  SELECT \* FROM CUSTOMER; |

Output:



Create LOAN table with LoanID, CustomerID, InterestRate, DueDate.

Query:

|  |
| --- |
| CREATE TABLE LOAN (  LoanID NUMBER PRIMARY KEY,  CustomerID NUMBER REFERENCES CUSTOMER(CustomerID),  InterestRate NUMBER,  DueDate DATE  ); |

Output:

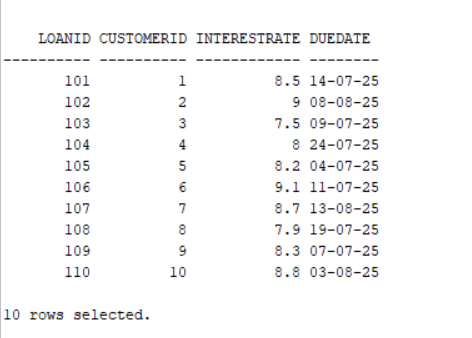


Inserting Value into CUSTOMER table.

Query:

|  |
| --- |
| INSERT INTO LOAN (LoanID, CustomerID, InterestRate, DueDate) VALUES (101, 1, 8.5, SYSDATE + 15);  INSERT INTO LOAN (LoanID, CustomerID, InterestRate, DueDate) VALUES (102, 2, 9.0, SYSDATE + 40);  INSERT INTO LOAN (LoanID, CustomerID, InterestRate, DueDate) VALUES (103, 3, 7.5, SYSDATE + 10);  INSERT INTO LOAN (LoanID, CustomerID, InterestRate, DueDate) VALUES (104, 4, 8.0, SYSDATE + 25);  INSERT INTO LOAN (LoanID, CustomerID, InterestRate, DueDate) VALUES (105, 5, 8.2, SYSDATE + 5);  INSERT INTO LOAN (LoanID, CustomerID, InterestRate, DueDate) VALUES (106, 6, 9.1, SYSDATE + 12);  INSERT INTO LOAN (LoanID, CustomerID, InterestRate, DueDate) VALUES (107, 7, 8.7, SYSDATE + 45);  INSERT INTO LOAN (LoanID, CustomerID, InterestRate, DueDate) VALUES (108, 8, 7.9, SYSDATE + 20);  INSERT INTO LOAN (LoanID, CustomerID, InterestRate, DueDate) VALUES (109, 9, 8.3, SYSDATE + 8);  INSERT INTO LOAN (LoanID, CustomerID, InterestRate, DueDate) VALUES (110, 10, 8.8, SYSDATE + 35);  COMMIT;  SELECT \* FROM LOAN; |

Output:



**STEP 2 – IMPLEMENTING SCENARIOS**

**SCENARIO 1**

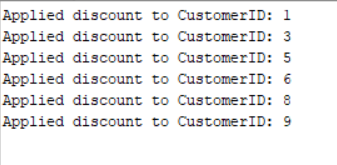
The bank wants to apply a discount to loan interest rates for customers above 60 years old.

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.

Query:

|  |
| --- |
| BEGIN  FOR cust\_rec IN (  SELECT CustomerID  FROM CUSTOMER  WHERE Age > 60  ) LOOP  UPDATE LOAN  SET InterestRate = InterestRate - 1  WHERE CustomerID = cust\_rec.CustomerID;  DBMS\_OUTPUT.PUT\_LINE('Applied discount to CustomerID: ' || cust\_rec.CustomerID);  END LOOP;  COMMIT;  END;  / |

Output:



**SCENARIO 2**

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

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.

Query:

|  |
| --- |
| BEGIN  FOR cust\_rec IN (  SELECT CustomerID, Balance  FROM CUSTOMER  ) LOOP  IF cust\_rec.Balance > 10000 THEN  UPDATE CUSTOMER  SET IsVIP = 'Y'  WHERE CustomerID = cust\_rec.CustomerID;  DBMS\_OUTPUT.PUT\_LINE('Promoted CustomerID ' || cust\_rec.CustomerID || ' to VIP.');  ELSE  UPDATE CUSTOMER  SET IsVIP = 'N'  WHERE CustomerID = cust\_rec.CustomerID;  DBMS\_OUTPUT.PUT\_LINE('Demoted CustomerID ' || cust\_rec.CustomerID || ' from VIP.');  END IF;  END LOOP;  COMMIT;  END;  / |

Output:



**SCENARIO 3**

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

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

Query:

|  |
| --- |
| BEGIN  FOR loan\_rec IN (  SELECT l.LoanID, l.CustomerID, l.DueDate, c.Name  FROM LOAN l  JOIN CUSTOMER c ON l.CustomerID = c.CustomerID  WHERE l.DueDate <= SYSDATE + 30  ) LOOP  DBMS\_OUTPUT.PUT\_LINE(  'Reminder: Customer ' || loan\_rec.Name ||  ' (ID: ' || loan\_rec.CustomerID ||  ') has Loan ID ' || loan\_rec.LoanID ||  ' due on ' || TO\_CHAR(loan\_rec.DueDate, 'DD-MON-YYYY')  );  END LOOP;  END;  / |

Output:

