WEEK 2
PL/SQL

TABLE CREATION AND DATA INSERTION:

```
-- Create the Customers table
CREATE TABLE Customers (
    CustomerID NUMBER PRIMARY KEY,
    Name VARCHAR2(100),
    DOB DATE,
    Balance NUMBER,
    LastModified DATE
);

-- Create the Accounts table
CREATE TABLE Accounts (
    AccountID NUMBER PRIMARY KEY,
    CustomerID NUMBER,
    AccountType VARCHAR2(20),
    Balance NUMBER,
    LastModified DATE,
    FOREIGN KEY (CustomerID) REFERENCES
Customers(CustomerID)
```

```
);

-- Create the Transactions table
CREATE TABLE Transactions (
    TransactionID NUMBER PRIMARY KEY,
    AccountID NUMBER,
    TransactionDate DATE,
    Amount NUMBER,
    TransactionType VARCHAR2(10),
    FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)
);

-- Create the Loans table
CREATE TABLE Loans (
    LoanID NUMBER PRIMARY KEY,
    CustomerID NUMBER,
    LoanAmount NUMBER,
    InterestRate NUMBER,
    StartDate DATE,
    EndDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

-- Create the Employees table

```
CREATE TABLE Employees (
    EmployeeID NUMBER PRIMARY KEY,
    Name VARCHAR2(100),
    Position VARCHAR2(50),
    Salary NUMBER,
    Department VARCHAR2(50),
    HireDate DATE
);
```

-- Create the AuditLog table

```
CREATE TABLE AuditLog (
    LogID NUMBER PRIMARY KEY,
    TransactionID NUMBER,
    LogDate DATE,
    Message VARCHAR2(255),
    FOREIGN KEY (TransactionID) REFERENCES Transactions(TransactionID)
);
```

-- Insert sample data into the Customers table

```
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 sample data into the Accounts table

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 sample data into the Transactions table

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 sample data into the Loans table

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

VALUES (1, 1, 5000, 5, SYSDATE, ADD_MONTHS(SYSDATE, 60));

-- Insert sample data into the Employees table

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'));

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.

```
CREATE OR REPLACE PROCEDURE SafeTransferFunds (
    p_from_account_id IN Accounts.AccountID%TYPE,
    p_to_account_id IN Accounts.AccountID%TYPE,
    p_amount IN NUMBER
) AS
    v_from_balance Accounts.Balance%TYPE;
    v_to_balance Accounts.Balance%TYPE;
BEGIN
    -- Check balances
    SELECT Balance INTO v_from_balance FROM Accounts WHERE AccountID = p_from_account_id FOR UPDATE;
    SELECT Balance INTO v_to_balance FROM Accounts WHERE AccountID = p_to_account_id FOR UPDATE;

    IF v_from_balance < p_amount THEN
        RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds in the source account.');
    END IF;

    -- Perform fund transfer
    UPDATE Accounts SET Balance = v_from_balance - p_amount WHERE AccountID = p_from_account_id;
    UPDATE Accounts SET Balance = v_to_balance + p_amount WHERE AccountID = p_to_account_id;

    COMMIT;
```

```
    DBMS_OUTPUT.PUT_LINE('Funds transferred successfully.');

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error during fund transfer: ' ||
SQLERRM);
END;
/
```

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.

```
CREATE OR REPLACE PROCEDURE UpdateSalary (
    p_employee_id IN Employees.EmployeeID%TYPE,
    p_percentage IN NUMBER
) AS
    v_salary Employees.Salary%TYPE;
BEGIN
    -- Attempt to update salary
    SELECT Salary INTO v_salary FROM Employees WHERE
EmployeeID = p_employee_id FOR UPDATE;

    UPDATE Employees
    SET Salary = Salary + (Salary * p_percentage / 100)
```

```
    WHERE EmployeeID = p_employee_id;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Salary updated successfully.');

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: Employee ID not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error updating salary: ' ||
SQLERRM);
END;
/
```

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.

```
CREATE OR REPLACE PROCEDURE AddNewCustomer (
    p_customer_id IN Customers.CustomerID%TYPE,
    p_name IN Customers.Name%TYPE,
    p_dob IN Customers.DOB%TYPE,
    p_balance IN Customers.Balance%TYPE
) AS
BEGIN
```

```
    -- Attempt to insert a new customer
    INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)
    VALUES (p_customer_id, p_name, p_dob, p_balance, SYSDATE);

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('New customer added successfully.');

EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Error: Customer with the same ID already exists.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error adding new customer: ' || SQLERRM);
END;
/
```