Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor) Write a PL/SQL block of code using parameterized Cursor that will merge the data availablein the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped
ANS=
o create a PL/SQL block using a parameterized cursor that merges data from the newly created table N_RollCall into the existing table O_RollCall, we can follow these steps:

1. **Define the tables:** For the purpose of this example, let's assume that both tables have the same structure and contain the following columns: Roll_No, Student_Name, and Attendance_Date.
2. **Create the PL/SQL block:** We'll define a parameterized cursor to fetch records from N_RollCall, check if they exist in O_RollCall, and if not, insert them into O_RollCall.

Step 1: Create the Sample Tables

Assuming the tables look something like this:

sql

Copy code

```
CREATE TABLE O_RollCall (
  Roll_No NUMBER PRIMARY KEY,
  Student_Name VARCHAR2(100),
  Attendance_Date DATE
);


CREATE TABLE N_RollCall (
  Roll_No NUMBER,
  Student_Name VARCHAR2(100),
  Attendance_Date DATE
);
```

Step 2: PL/SQL Block with Parameterized Cursor

Here's the complete PL/SQL block that uses a parameterized cursor to merge data:

sql

Copy code

```
DECLARE
  -- Parameterized cursor to select records from N_RollCall
  CURSOR N_RollCall_Cursor (p_roll_no NUMBER) IS
```

```
    SELECT Roll_No, Student_Name, Attendance_Date
    FROM N_RollCall
    WHERE Roll_No = p_roll_no;  -- Parameter for filtering records

  v_roll_no O_RollCall.Roll_No%TYPE;
  v_student_name O_RollCall.Student_Name%TYPE;
  v_attendance_date O_RollCall.Attendance_Date%TYPE;
BEGIN
  -- Loop through each record in N_RollCall
  FOR rec IN (SELECT Roll_No, Student_Name, Attendance_Date FROM N_RollCall) LOOP
    -- Fetch data using the parameterized cursor
    OPEN N_RollCall_Cursor(rec.Roll_No);
    FETCH N_RollCall_Cursor INTO v_roll_no, v_student_name, v_attendance_date;

    -- Check if the record already exists in O_RollCall
    IF N_RollCall_Cursor%FOUND THEN
      DECLARE
        v_exists NUMBER;
      BEGIN
        -- Check if the record exists in O_RollCall
        SELECT COUNT(*)
        INTO v_exists
        FROM O_RollCall
        WHERE Roll_No = v_roll_no
          AND Attendance_Date = v_attendance_date;

        -- If the record does not exist, insert it
        IF v_exists = 0 THEN
          INSERT INTO O_RollCall (Roll_No, Student_Name, Attendance_Date)
          VALUES (v_roll_no, v_student_name, v_attendance_date);
        END IF;
      END;
    END IF;

    CLOSE N_RollCall_Cursor;  -- Close the cursor after processing
  END LOOP;

  COMMIT;  -- Commit the transaction after all inserts
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;  -- Rollback in case of any error
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

Explanation of the PL/SQL Block

1. **Parameterized Cursor Declaration:**
   - A cursor N_RollCall_Cursor is defined with a parameter p_roll_no. It fetches records from N_RollCall based on the provided roll number.
2. **Loop through Records:**
   - A FOR loop iterates through all records in N_RollCall.
3. **Open and Fetch Cursor:**
   - The cursor is opened with the current roll number and data is fetched into local variables (v_roll_no, v_student_name, v_attendance_date).
4. **Check for Existence:**
   - A nested block checks if the record already exists in O_RollCall using a SELECT COUNT(*). If the count is zero, it means the record does not exist.
5. **Insert Records:**
   - If the record doesn't exist, it is inserted into O_RollCall.
6. **Exception Handling:**
   - An exception block is included to handle any errors, performing a rollback if necessary and printing an error message.

Note

- Make sure you have sample data in the N_RollCall and O_RollCall tables before running this PL/SQL block.

- Adjust the table structure and column types as necessary based on your actual requirements.
- The cursor's parameter is useful when you want to fetch specific records based on input criteria. In this example, we used it for demonstration, but it's fetched from the outer loop. You could also refactor the code to use the cursor parameter more explicitly depending on the requirements.