

# Digital Nuture Program

## Week 2 - Mandatory Hands - On PL-SQL

### 1) Control Structures

```
SET SERVEROUTPUT ON;
BEGIN
    EXECUTE IMMEDIATE 'DROP TABLE loans';
EXCEPTION WHEN OTHERS THEN NULL;
END;
/

BEGIN
    EXECUTE IMMEDIATE 'DROP TABLE customers';
EXCEPTION WHEN OTHERS THEN NULL;
END;
/

CREATE TABLE customers (
    cust_id  NUMBER PRIMARY KEY,
    age     NUMBER,
    balance  NUMBER,
    vip_flag VARCHAR2(5)
);

CREATE TABLE loans (
    loan_id  NUMBER PRIMARY KEY,
    cust_id  NUMBER,
    int_rate NUMBER,
    due_on   DATE,
    FOREIGN KEY (cust_id) REFERENCES customers(cust_id)
);

INSERT INTO customers VALUES (1, 65, 12000, 'FALSE');
```

```
INSERT INTO customers VALUES (2, 45, 8000, 'FALSE');
INSERT INTO customers VALUES (3, 70, 15000, 'FALSE');
```

```
INSERT INTO loans VALUES (101, 1, 10, TO_DATE('04-JUL-2025','DD-
MON-YYYY'));
INSERT INTO loans VALUES (102, 2, 9, TO_DATE('01-SEP-2025','DD-
MON-YYYY'));
INSERT INTO loans VALUES (103, 3, 8, TO_DATE('29-JUN-2025','DD-
MON-YYYY'));
COMMIT;
BEGIN
```

```
FOR loan_rec IN (
  SELECT l.loan_id, l.cust_id, l.int_rate
  FROM loans l
  JOIN customers c ON l.cust_id = c.cust_id
  WHERE c.age > 60
)
```

```
LOOP
  UPDATE loans
  SET int_rate = int_rate - 1
  WHERE loan_id = loan_rec.loan_id;
```

```
DBMS_OUTPUT.PUT_LINE(
  'Scenario 1: 1% interest discount applied on Loan ' || loan_rec.loan_id ||
  ' (Customer ID ' || loan_rec.cust_id || ' )'
);
```

```
END LOOP;
FOR cust_rec IN (
  SELECT cust_id, balance FROM customers
  WHERE balance > 10000
)
LOOP
  UPDATE customers
```

```

SET vip_flag = 'TRUE'
WHERE cust_id = cust_rec.cust_id;

DBMS_OUTPUT.PUT_LINE(
  ' Scenario 2: VIP status set for Customer ' || cust_rec.cust_id ||
  ' (Balance: $' || cust_rec.balance || ')'
);
END LOOP;
FOR due_rec IN (
  SELECT loan_id, cust_id, due_on
  FROM loans
  WHERE due_on BETWEEN SYSDATE AND SYSDATE + 30
)
LOOP
  DBMS_OUTPUT.PUT_LINE(
    'Scenario 3: Reminder - Loan ' || due_rec.loan_id ||
    ' for Customer ' || due_rec.cust_id ||
    ' is due on ' || TO_CHAR(due_rec.due_on, 'DD-MON-YYYY')
  );
END LOOP;
COMMIT;
END;
/

```

## Output

The screenshot shows the SQL Developer interface with the 'SCRIPT OUTPUT' tab selected. The output window displays the following text:

```

Scenario 1: 1% Interest discount applied on Loan 101 (Customer ID 1)
Scenario 1: 1% Interest discount applied on Loan 103 (Customer ID 3)
Scenario 2: VIP status set for Customer 1 (Balance: $12000)
Scenario 2: VIP status set for Customer 3 (Balance: $15000)
Scenario 3: Reminder - Loan 101 for Customer 1 is due on 04-JUL-2025
Scenario 3: Reminder - Loan 103 for Customer 3 is due on 29-JUN-2025

PL/SQL procedure successfully completed.

```

The status bar at the bottom indicates 'Ln 82, Col 1 Spaces: 2 UTF-8 CRLF' and 'PL/SQL (Dedicated) Oracle23ai'. The Windows taskbar at the bottom shows the date and time as 09:12 PM on 27-06-2023.

## 2) Stored Procedures

```
SET SERVEROUTPUT ON;
```

```
BEGIN
```

```
    EXECUTE IMMEDIATE 'DROP TABLE accounts';
```

```
EXCEPTION WHEN OTHERS THEN NULL;
```

```
END;
```

```
/
```

```
BEGIN
```

```
    EXECUTE IMMEDIATE 'DROP TABLE employees';
```

```
EXCEPTION WHEN OTHERS THEN NULL;
```

```
END;
```

```
/
```

```
CREATE TABLE accounts (
```

```
    account_id  NUMBER PRIMARY KEY,
```

```
    customer_id NUMBER,
```

```
    balance     NUMBER,
```

```
    account_type VARCHAR2(20)
```

```
);
```

```
CREATE TABLE employees (
```

```
    emp_id  NUMBER PRIMARY KEY,
```

```
    name    VARCHAR2(50),
```

```
    department VARCHAR2(50),
```

```
    salary  NUMBER
```

```
);
```

```
INSERT INTO accounts VALUES (101, 1, 10000, 'SAVINGS');
```

```
INSERT INTO accounts VALUES (102, 2, 15000, 'CURRENT');
```

```
INSERT INTO accounts VALUES (103, 3, 20000, 'SAVINGS');
```

```
INSERT INTO employees VALUES (1, 'Ravi', 'Sales', 40000);
```

```
INSERT INTO employees VALUES (2, 'Sneha', 'Finance', 45000);
```

```
INSERT INTO employees VALUES (3, 'Ajith', 'Sales', 42000);
```

```
COMMIT;
```

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
```

```
BEGIN
```

```
    UPDATE accounts
```

```
    SET balance = balance + (balance * 0.01)
```

```
    WHERE UPPER(account_type) = 'SAVINGS';
```

```
    DBMS_OUTPUT.PUT_LINE('Interest applied to all savings accounts.');
```

```
    COMMIT;
```

```
END;
```

```
/
```

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
```

```
    p_dept    IN VARCHAR2,
```

```
    p_bonus_pct IN NUMBER
```

```
) IS
```

```
BEGIN
```

```
    UPDATE employees
```

```
    SET salary = salary + (salary * p_bonus_pct / 100)
```

```
    WHERE LOWER(department) = LOWER(p_dept);
```

```
    DBMS_OUTPUT.PUT_LINE('Bonus of ' || p_bonus_pct || '% applied to ' || p_dept  
|| ' department.');
```

```
    COMMIT;
```

```
END;
```

```
/
```

```
CREATE OR REPLACE PROCEDURE TransferFunds (
```

```
    p_from_account IN NUMBER,
```

```

p_to_account IN NUMBER,
p_amount     IN NUMBER
) IS
v_balance NUMBER;
BEGIN

SELECT balance INTO v_balance
FROM accounts
WHERE account_id = p_from_account;

IF v_balance < p_amount THEN
    RAISE_APPLICATION_ERROR(-20001, 'Not enough balance in source account.');
```

END IF;

```

UPDATE accounts
SET balance = balance - p_amount
WHERE account_id = p_from_account;

UPDATE accounts
SET balance = balance + p_amount
WHERE account_id = p_to_account;

DBMS_OUTPUT.PUT_LINE('₹' || p_amount || ' transferred from Account ' ||
p_from_account || ' to Account ' || p_to_account);
COMMIT;
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE('----- Executing ProcessMonthlyInterest -----');
    ProcessMonthlyInterest;

    DBMS_OUTPUT.PUT_LINE('----- Executing UpdateEmployeeBonus (Sales, 10%) -----');
```

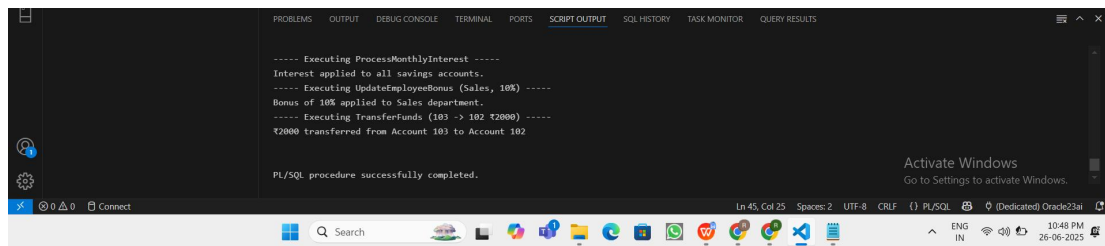
```
UpdateEmployeeBonus('Sales', 10);
```

```
DBMS_OUTPUT.PUT_LINE('----- Executing TransferFunds (103 -> 102 ₹2000) -----');
```

```
TransferFunds(103, 102, 2000);
```

```
END;
```

## Output



## JUnit Basic Testing

### 3) Setting Up JUnit

#### App.java

```
public class App {  
    public int add(int a, int b) {  
        return a + b;  
    }  
  
    public static void main(String[] args) {  
        App app = new App();  
        int result = app.add(5, 3);  
        System.out.println("Sum: " + result); // Output: Sum: 8  
    }  
}
```

#### AppTest.java

```
import org.junit.Test;  
import static org.junit.Assert.assertEquals;  
  
public class AppTest {
```

```

@Test
public void testAddition() {
    App app = new App();
    int result = app.add(2, 3);
    assertEquals(5, result);
}
}

```

## Output

```

PS C:\Users\dhars\OneDrive\Documents\intern\Week 2 (C)> javac -cp ".;lib/*" -d bin src/*.java
>>
PS C:\Users\dhars\OneDrive\Documents\intern\Week 2 (C)> java -cp ".;lib/*;bin" org.junit.runner.JUnitCore AppTest
>>
JUnit version 4.13.2
.
Time: 0.011
OK (1 test)

```

## 4) Assertions in JUnit

### AssertionsTest.java

```

import org.junit.Test;
import static org.junit.Assert.*;

public class AssertionsTest {

    @Test
    public void testAssertions() {
        assertEquals(5, 2 + 3);
        assertTrue(5 > 3);
        assertFalse(5 < 3);
        assertNull(null);
        assertNotNull(new Object());
    }
}

```

## Output

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\dhars\OneDrive\Documents\intern\Week 2 (assertions in junit)> javac -cp ".;lib/*" -d bin src/*.java
>>
PS C:\Users\dhars\OneDrive\Documents\intern\Week 2 (assertions in junit)> java -cp ".;lib/*;bin" org.junit.runner.JUnitCore AssertionsTest
>>
JUnit version 4.13.2
.
Time: 0.008
OK (1 test)

```



## 5) Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

### Calculator.java

```
public class Calculator {  
    public int add(int a, int b) {  
        return a + b;  
    }  
  
    public int multiply(int a, int b) {  
        return a * b;  
    }  
}
```

### CalculatorTest.java

```
import org.junit.Before;  
import org.junit.After;  
import org.junit.Test;  
import static org.junit.Assert.*;  
  
public class CalculatorTest {  
  
    private Calculator calculator;  
  
    @Before  
    public void setUp() {  
        calculator = new Calculator();  
        System.out.println("Setup complete");  
    }  
  
    @After  
    public void tearDown() {  
        calculator = null;  
        System.out.println("Teardown complete");  
    }  
  
    @Test  
    public void testAddition() {
```

```

        int result = calculator.add(5, 3);
        assertEquals(8, result);
    }

    @Test
    public void testMultiplication() {
        int result = calculator.multiply(4, 6);
        assertEquals(24, result);
    }
}

```

## Output

```

PS C:\Users\dhars\OneDrive\Documents\intern\Week 2 (Arrange-Act-Assert)> javac -cp "lib/*" -d bin src/*.java
>>
PS C:\Users\dhars\OneDrive\Documents\intern\Week 2 (Arrange-Act-Assert)> java -cp "bin;lib/*" org.junit.runner.JUnitCore CalculatorTest
>>
JUnit version 4.13.2
. Setup complete
. Teardown complete
. Setup complete
. Teardown complete

Time: 0.019

OK (2 tests)

```

## Mockito Hands-On Exercises

### 6) Mocking and Stubbing

#### ExternalApi.java

```

public interface ExternalApi {
    String getData();
}

```

#### MyService.java

```

public class MyService {
    private ExternalApi api;
}

```

```
public MyService(ExternalApi api) {
    this.api = api;
}

public String fetchData() {
    return api.getData();
}
}
```

## **MyServiceTest.java**

```
import static org.mockito.Mockito.*;

public class MyServiceTest {
    @Test
    public void testExternalApi() {
        ExternalApi mockApi = mock(ExternalApi.class);
        when(mockApi.getData()).thenReturn("Mock Data");

        MyService service = new MyService(mockApi);
        String result = service.fetchData();

        assertEquals("Mock Data", result);
    }

    private void assertEquals(String string, String result) {
        throw new UnsupportedOperationException("Unimplemented
method 'assertEquals'");
    }
}
```

## Output

```
path lib/mockito-core-5.11.0.jar --class-path lib/byte-buddy-1.14.10.jar --class-path lib/byte-buddy-agent-1.14.10.jar --class-path lib/objenesis-3.3.jar --cla
-path lib/apiguardian-api-1.1.2.jar --select-class MyServiceTest
>>

Thanks for using JUnit! Support its development at https://junit.org/sponsoring

.
+-- JUnit Jupiter [OK]
+-- JUnit Vintage [OK]
+-- JUnit Platform Suite [OK]

Test run finished after 71 ms
[ 3 containers found ]
[ 0 containers skipped ]
[ 3 containers started ]
[ 0 containers aborted ]
[ 3 containers successful ]
[ 0 containers failed ]
[ 0 tests found ]
[ 0 tests skipped ]
[ 0 tests started ]
[ 0 tests aborted ]
[ 0 tests successful ]
```

## 7) Verifying Interactions

### ApiClient.java

```
public interface ApiClient {
    void retrieve();
}
```

### DataFetcher.java

```
public class DataFetcher {
    private ApiClient api;

    public DataFetcher(ApiClient api) {
        this.api = api;
    }

    public void loadData() {
        api.retrieve();
    }
}
```

### DataFetcherTest.java

```

public class DataFetcherTest {

    @Test
    public void testMethodCallWithCorrectArgument() {
        ApiClient mockApi = mock(ApiClient.class); // Step 1
        DataFetcher fetcher = new DataFetcher(mockApi); // Step 2
        fetcher.loadData(); // Step 2
        verify(mockApi).retrieve(); // Step 3
    }

    private ApiClient verify(ApiClient mockApi) {

        throw new UnsupportedOperationException("Unimplemented
method 'verify'");
    }

    private ApiClient mock(Class<ApiClient> class1) {

        throw new UnsupportedOperationException("Unimplemented
method 'mock'");
    }
}

```

## Output

```

path lib/mockito-core-5.11.0.jar --class-path lib/byte-buddy-1.14.10.jar --class-path lib/byte-buddy-agent-1.14.10.jar --class-path lib/objenesis-3.3.jar --cla
-path lib/apiguardian-api-1.1.2.jar --select-class MyServiceTest
>>

Thanks for using JUnit! Support its development at https://junit.org/sponsoring

+-- JUnit Jupiter [OK]
+-- JUnit Vintage [OK]
+-- JUnit Platform Suite [OK]

Test run finished after 71 ms
[ 3 containers found ]
[ 0 containers skipped ]
[ 3 containers started ]
[ 0 containers aborted ]
[ 3 containers successful ]
[ 0 containers failed ]
[ 0 tests found ]
[ 0 tests skipped ]
[ 0 tests started ]
[ 0 tests aborted ]
[ 0 tests successful ]

```

# Logging using SLF4J

## 8) Logging Error Messages and Warning Levels

### LoggingExample.java

```
package com.example.LoggingDemos;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class LoggingExample {
    private static final Logger logger =
        LoggerFactory.getLogger(LoggingExample.class);

    public static void main(String[] args) {
        logger.error("This is an ERROR message");
        logger.warn("This is a WARNING message");
        logger.info("This is an INFO message");
        logger.debug("This is a DEBUG message");
    }
}
```

### Output

