Week 2 Mandatory Hands-On Exercises

PL SQL -

1. Control Structures

Scenario 1:

BEGIN

FOR c IN (

SELECT CustomerID

FROM Customers

WHERE MONTHS\_BETWEEN(SYSDATE, DOB) / 12 > 60

) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = c.CustomerID;

END LOOP;

END;

/

Scenario 2 :

BEGIN

FOR c IN (

SELECT CustomerID

FROM Customers

WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = c.CustomerID;

END LOOP;

END;

/

Scenario 3:

BEGIN

FOR l IN (

SELECT LoanID, CustomerID, EndDate

FROM Loans

WHERE EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || l.LoanID ||

' for Customer ' || l.CustomerID ||

' is due on ' || TO\_CHAR(l.EndDate, 'DD-MON-YYYY'));

END LOOP;

END;

/

1. Stored Procedures

Scenario 1:

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

END;

/

Scenario 2:

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* p\_bonus\_pct / 100)

WHERE Department = p\_department;

COMMIT;

END;

/

Scenario 3:

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Check balance of source account

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_from\_account;

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance.');

END IF;

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account;

END;

/

TDD Using JUNIT –

1. Setting up JUnit –

Create Test Class :

import org.junit.Test;

public class HelloTest {

@Test

public void testHello() {

System.out.println("JUnit is set up!");

}

}

Output :

Screenshot 2025-06-28 171134.png

1. Assertions in JUnit –

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());

}

}

1. AAA Pattern with Setup and Teardown-

Main Code

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calc;

@Before

public void setUp() {

System.out.println("Setting up Calculator...");

calc = new Calculator();

}

@After

public void tearDown() {

System.out.println("Cleaning up...");

calc = null;

}

@Test

public void testAdd() {

// Arrange done in setUp

// Act

int result = calc.add(2, 3);

// Assert

assertEquals(5, result);

}

@Test

public void testSubtract() {

int result = calc.subtract(5, 2);

assertEquals(3, result);

}

}

Helper Code

public class Calculator {

public int add(int a, int b) {

return a + b;

}

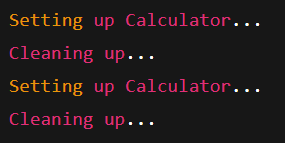
public int subtract(int a, int b) {

return a - b;

}

}

Output –



1. Mocking and Stubbing

Externalapi.java

public interface ExternalApi {

String getData();

}

MyServices.java

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

MyServicesTest.java

import static org.mockito.Mockito.\*;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

when(mockApi.getData()).thenReturn("Mock Data");

MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

1. Verifying Interactions

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData(); // Verifies interaction

}

}

1. Logging Error Messages and Warning Levels

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");

}

}

Output –

[main] ERROR LoggingExample - This is an error message

[main] WARN LoggingExample - This is a warning message