**Week 2**

**Mandatory Hands-on**

**PL/SQL PROGRAMMING**

Exercise 1

**TABLES**

**CUSTOMERS**

**A screenshot of a computer

AI-generated content may be incorrect.**

**LOANS**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Code**

**Scenario 1**

DECLARE

CURSOR cust\_cursor IS

SELECT customer\_id, loan\_id, interest\_rate

FROM customers c

JOIN loans l ON c.customer\_id = l.customer\_id

WHERE TRUNC(MONTHS\_BETWEEN(SYSDATE, c.dob)/12) > 60;

BEGIN

FOR rec IN cust\_cursor LOOP

UPDATE loans

SET interest\_rate = interest\_rate \* 0.99

WHERE loan\_id = rec.loan\_id;

DBMS\_OUTPUT.PUT\_LINE('1% Discount applied for Customer ID: ' || rec.customer\_id);

END LOOP;

COMMIT;

END;

**OUTPUT**

**A black text on a white background

AI-generated content may be incorrect.**

**Scenario 2**

Code

BEGIN

    FOR rec IN(SELECT customer\_id,balance FROM customers WHERE balance>10000) LOOP

        UPDATE CUSTOMERS

        SET ISVIP="TRUE"

        WHERE CUSTOMER\_ID=rec.CUSTOMER\_ID;

        DBMS\_OUTPUT.PUT\_LINE('THIS CUSTOMER IS VIP NOW: customer\_id '||rec.CUSTOMER\_ID);

    END LOOP;

    COMMIT;

END;

A group of black text

AI-generated content may be incorrect.**Output**

**Scenario 3**

**Code**

DECLARE

    CURSOR loan\_cursor IS

        SELECT c.customer\_id, c.name, l.due\_date

        FROM loans l

        JOIN customers c ON l.customer\_id = c.customer\_id

        WHERE l.due\_date BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

    FOR rec IN loan\_cursor LOOP

        DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan for Customer ' || rec.name ||

                             ' (ID: ' || rec.customer\_id ||

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

    END LOOP;

END;

**OUTPUT**

****

**EXERCISE 3**

**TABLES**

ACCOUNTS

A screenshot of a computer

AI-generated content may be incorrect.

EMPLOYEES

A screenshot of a computer

AI-generated content may be incorrect.

**CODE**

**SCENARIO 1**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

    UPDATE Accounts

    SET Balance = Balance \* 1.01

    WHERE LOWER(Account\_Type) = 'savings';

    DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

END;

EXEC ProcessMonthlyInterest;

**OUTPUT**

**A screenshot of a computer

AI-generated content may be incorrect.**

**SCENARIO 2**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

    dept\_name IN VARCHAR2,

    bonus\_percent IN NUMBER

) AS

BEGIN

    UPDATE Employees

    SET Salary = Salary + (Salary \* bonus\_percent / 100)

    WHERE LOWER(Department) = LOWER(dept\_name);

    DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department: ' || dept\_name);

END;

/

EXEC UpdateEmployeeBonus('HR', 10);

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.

**SCENARIO 3**

CREATE OR REPLACE PROCEDURE TransferFunds (

    from\_acc IN NUMBER,

    to\_acc IN NUMBER,

    amount IN NUMBER

) AS

    insufficient\_balance EXCEPTION;

    from\_balance NUMBER;

BEGIN

    -- Get balance of source account

    SELECT Balance INTO from\_balance FROM Accounts WHERE Account\_ID = from\_acc FOR UPDATE;

    IF from\_balance < amount THEN

        RAISE insufficient\_balance;

    END IF;

    -- Deduct from source

    UPDATE Accounts SET Balance = Balance - amount WHERE Account\_ID = from\_acc;

    -- Add to destination

    UPDATE Accounts SET Balance = Balance + amount WHERE Account\_ID = to\_acc;

    DBMS\_OUTPUT.PUT\_LINE('Transferred ' || amount || ' from Account ' || from\_acc || ' to ' || to\_acc);

EXCEPTION

    WHEN insufficient\_balance THEN

        DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient funds in account ' || from\_acc);

    WHEN NO\_DATA\_FOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('Transfer failed: One or both account IDs not found.');

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM);

END;

EXEC TransferFunds(102, 105, 2000);

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.

**TDD using JUnit5 and Mockito**

**JUnit\_Basic Testing Exercises**

**Exercise 1**

**Code**

main

package main.java;  
  
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}

test

package test.java;  
  
import main.java.Calculator;  
import org.junit.Test;  
import static org.junit.Assert.*assertEquals*;  
  
public class CalculatorTest {  
 @Test  
 public void testAdd() {  
 Calculator calc = new Calculator();  
 int result = calc.add(5, 3);  
 System.*out*.println("Sum is: " + result);  
 *assertEquals*(8, result);  
 }  
}

**Output**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Exercise 3**

**Code**

AssertionsTest

import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class AssertionsTest {  
 @Test  
 public void testAssertions() {  
 // Assert equals  
 *assertEquals*(5, 2 + 3);  
  
 // Assert true  
 *assertTrue*(5 > 3);  
  
 // Assert false  
 *assertFalse*(5 < 3);  
  
 // Assert null  
 *assertNull*(null);  
  
 // Assert not null  
 *assertNotNull*(new Object());  
 }  
}

**Output**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 4**

**Code**

CalculatorTest

import main.java.Calculator;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorWithAAATest {

private Calculator calc;

// Arrange: setup method runs before each test

@Before

public void setUp() {

calc = new Calculator();

System.out.println("Setup: Calculator initialized");

}

// Act + Assert

@Test

public void testAddition() {

// Act

int result = calc.add(10, 5);

// Assert

assertEquals(15, result);

}

// Teardown: runs after each test

@After

public void tearDown() {

calc = null;

System.out.println("Teardown: Calculator cleared");

}

}

**Output**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Mockito exercises**

**Exercise 1**

**MyService**

package main.java;  
  
public class MyService {  
 private ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData();  
 }  
}

**MyServiceTest**

package test.java;  
  
import main.java.ExternalApi;  
import main.java.MyService;  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
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);  
 }  
}

**Output**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 2**

**MyServiceTest**

package test.java;  
  
import main.java.ExternalApi;  
import main.java.MyService;  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
  
public class MyServiceTest {  
  
 @Test  
 public void testVerifyInteraction() {  
 ExternalApi mockApi = *mock*(ExternalApi.class);  
 MyService service = new MyService(mockApi);  
 service.fetchData();  
 *verify*(mockApi).getData();  
 }  
}

**Output**

**A screenshot of a computer

AI-generated content may be incorrect.**

**SL4J Logging exercises**

**Exercise 1**

**LoggingExample**

package main.java;  
  
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");  
 }  
}

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>LoggingExample</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- SLF4J API -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<!-- Logback Classic: SLF4J Binding Implementation -->

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

</project>

**Output**

**A screen shot of a computer

AI-generated content may be incorrect.**