**Scenario 1: Apply 1% Discount for Customers Above 60**BEGIN

FOR cust\_rec IN (

SELECT customer\_id, age

FROM customers

WHERE age > 60

) LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = cust\_rec.customer\_id;

END LOOP;

COMMIT;

END;  
  
  
**Scenario 2: Set IsVIP Flag for High Balance Customers**BEGIN

FOR cust\_rec IN (

SELECT customer\_id

FROM customers

WHERE balance > 10000

) LOOP

UPDATE customers

SET IsVIP = 'TRUE'

WHERE customer\_id = cust\_rec.customer\_id;

END LOOP;

COMMIT;

END;  
  
  
**Scenario 3: Print Loan Reminders for Upcoming Dues**DECLARE

v\_due\_date loans.due\_date%TYPE;

v\_customer\_id loans.customer\_id%TYPE;

BEGIN

FOR loan\_rec IN (

SELECT customer\_id, due\_date

FROM loans

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Customer ID ' || loan\_rec.customer\_id ||

' has a loan due on ' || TO\_CHAR(loan\_rec.due\_date, 'DD-MON-YYYY')

);

END LOOP;

END;

**3 Question  
  
Scenario 1: ProcessMonthlyInterest for Savings Accounts**CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc\_rec IN (

SELECT account\_id, balance

FROM accounts

WHERE account\_type = 'SAVINGS'

) LOOP

UPDATE accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_id = acc\_rec.account\_id;

END LOOP;

COMMIT;

END;  
  
**Scenario 2: UpdateEmployeeBonus for Department-Based Bonus**

sql

CopyEdit

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_id IN NUMBER,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE employees

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

WHERE department\_id = dept\_id;

COMMIT;

END;

**Scenario 3: TransferFunds Between Two Accounts**

sql

CopyEdit

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

-- Get the balance of the source account

SELECT balance INTO v\_balance

FROM accounts

WHERE account\_id = from\_account\_id

FOR UPDATE;

-- Check if balance is sufficient

IF v\_balance < amount THEN

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

ELSE

-- Deduct from source account

UPDATE accounts

SET balance = balance - amount

WHERE account\_id = from\_account\_id;

-- Add to destination account

UPDATE accounts

SET balance = balance + amount

WHERE account\_id = to\_account\_id;

COMMIT;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20002, 'One of the account IDs does not exist.');

WHEN OTHERS THEN

ROLLBACK;

RAISE;

END;

**JUnit Setup and Example Code**

**JUnit Maven Dependency (pom.xml)**

<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
</dependency>  
  
<dependency>  
 <groupId>org.hamcrest</groupId>  
 <artifactId>hamcrest-core</artifactId>  
 <version>1.3</version>  
 <scope>test</scope>  
</dependency>

**Calculator.java**

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

**CalculatorTest.java**

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(2, 3);  
 assertEquals(5, result);  
 }  
}

**JUnit Assertions and AAA Pattern Examples**

**Exercise 3: AssertionsTest.java**

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());  
 }  
}

**Exercise 4: CalculatorTestAAA.java**

import org.junit.Before;  
import org.junit.After;  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class CalculatorTestAAA {  
  
 private Calculator calculator;  
  
 @Before  
 public void setUp() {  
 // Arrange: Initialize resources  
 calculator = new Calculator();  
 System.out.println("Setup complete.");  
 }  
  
 @After  
 public void tearDown() {  
 // Cleanup after test  
 calculator = null;  
 System.out.println("Teardown complete.");  
 }  
  
 @Test  
 public void testAddition() {  
 // Act: Call the method  
 int result = calculator.add(10, 5);  
  
 // Assert: Verify the result  
 assertEquals(15, result);  
 }  
}

**Mockito: Mocking, Stubbing, and Interaction Verification**

**Exercise 1: Mocking and Stubbing**

import static org.mockito.Mockito.\*;  
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);  
 }  
}

**Exercise 2: 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();  
 }  
}

from docx import Document

from docx.shared import Pt

from docx.oxml.ns import qn

from docx.oxml import OxmlElement

# Create a new Word document

doc = Document()

doc.add\_heading('SLF4J Logging Example with Error and Warning Levels', level=1)

# Helper function to add code blocks

def add\_code\_block(doc, title, code):

doc.add\_heading(title, level=2)

paragraph = doc.add\_paragraph()

run = paragraph.add\_run(code)

font = run.font

font.name = 'Courier New'

font.size = Pt(10)

rPr = run.\_element.rPr

rFonts = OxmlElement('w:rFonts')

rFonts.set(qn('w:ascii'), 'Courier New')

rFonts.set(qn('w:hAnsi'), 'Courier New')

rPr.append(rFonts)

# SLF4J and Logback dependencies

dependencies\_code = """

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

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

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

"""

add\_code\_block(doc, "Step 1: Add to pom.xml", dependencies\_code)

# Java logging class example

logging\_class\_code = """

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");

}

}

"""

add\_code\_block(doc, "Step 2: LoggingExample.java", logging\_class\_code)

# Save the document

doc\_path = "/mnt/data/SLF4J\_Logging\_Example.docx"

doc.save(doc\_path)

doc\_path