WEEK 2: PL/SQL

PL/SQL:

Tables used: Table Name: ACCOUNTS

"ACCOUNTID" "CUSTOMERID" "ACCOUNTTYPE" "BALANCE" "LASTMODIFIED"

1 1 "Savings" 1000 "2025-06-28T03:30:24Z"

2 2 "Checking" 1500 "2025-06-28T03:40:25Z"

3 3 "Checking" 10000 "2025-06-28T03:51:54Z"

4 4 "Saving" 11000 "2025-06-28T03:53:07Z"

5 5 "Saving" 1100 "2025-06-28T03:53:23Z"

6 6 "checking" 500 "2025-06-28T03:53:44Z"

7 7 "Checking" 3200 "2025-06-28T03:55:19Z"

Table Name: CUSTOMERS

"CUSTOMERID" "NAME" "DOB" "BALANCE" "LASTMODIFIED" "ISVIP"

7 "Jane" "2000-07-20T00:00:00Z" 15000 "2025-06-28T03:20:41Z" "TRUE"

1 "John Doe" "1985-05-15T00:00:00Z" 1000 "2025-06-28T03:11:12Z"

3 "John " "1967-05-10T00:00:00Z" 10000 "2025-06-28T03:15:42Z"

4 "sony" "1987-05-15T00:00:00Z" 11000 "2025-06-28T03:17:10Z" "TRUE"

5 "samuel" "1990-05-30T00:00:00Z" 1100 "2025-06-28T03:17:54Z"

2 "Mahesh" "1990-06-20T00:00:00Z" 500 "2025-06-28T03:12:45Z"

6 "Jane Smith" "1990-07-20T00:00:00Z" 1500 "2025-06-28T03:20:16Z"

Table Name:EMPLOYEES

"EMPLOYEEID" "NAME" "POSITION" "SALARY" "DEPARTMENT" "HIREDATE"

5 "david" "tester" 60000 "it" "2020-03-20T00:00:00Z"

6 "khan" "developer" 55000 "it" "2019-06-10T00:00:00Z"

7 "sony" "sales executive" 65000 "sales" "2022-07-01T00:00:00Z"

3 "Brown" "Analyst" 50000 "Finance" "2017-01-20T00:00:00Z"

4 "Frank" "Developer" 62000 "it" "2018-03-20T00:00:00Z"

1 "Alice Johnson" "Manager" 70000 "HR" "2015-06-15T00:00:00Z"

2 "Bob Brown" "Developer" 60000 "IT" "2017-03-20T00:00:00Z"

Table Name:Loans

"LOANID" "CUSTOMERID" "LOANAMOUNT" "INTERESTRATE" "STARTDATE" "ENDDATE"

4 4 5000 2 "2025-06-28T03:58:30Z" "2030-06-28T03:58:30Z"

5 5 50000 6 "2025-06-28T03:58:52Z" "2027-02-28T03:58:52Z"

7 7 20000 5 "2025-06-28T03:59:54Z" "2027-02-28T03:59:54Z"

1 1 5000 5 "2025-06-28T03:56:06Z" "2030-06-28T03:56:06Z"

2 2 10000 3 "2025-06-28T03:57:40Z" "2027-12-28T03:57:40Z"

3 3 1500 7 "2025-06-28T03:58:09Z" "2030-06-28T03:58:09Z"

6 6 5200 6 "2025-06-28T03:59:19Z" "2026-04-28T03:59:19Z"

Exercise 1: Control Structures:

Code:

--WEEK 2: PL/SQL, JUnit\_Basic Testing, Mockito, SL4J Logging.

--PL/SQL:

--Exercise 1: Control Structures:

--Code:

--Tables used: Customers, loans

ALTER TABLE CUSTOMERS ADD ISVIP VARCHAR2(5);

SET SERVEROUTPUT ON;

BEGIN

    DBMS\_OUTPUT.PUT\_LINE('Case1:');

    FOR c IN (

        SELECT c.CustomerID, c.DOB, l.LoanID, l.InterestRate

        FROM Customers c

        JOIN Loans l ON c.CustomerID = l.CustomerID

    )

    LOOP

        IF MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12 > 60 THEN

            UPDATE Loans

            SET InterestRate = InterestRate - 1

            WHERE LoanID = c.LoanID;

            DBMS\_OUTPUT.PUT\_LINE('DISCOUNT:' || c.LoanID ||

                                 ' (CUSTOMERID: ' || c.CustomerID );

        END IF;

    END LOOP;

  DBMS\_OUTPUT.PUT\_LINE('CASE2:');

FOR c IN (

        SELECT CustomerID, Name, Balance

        FROM Customers

        WHERE Balance > 10000

    )

    LOOP

        DBMS\_OUTPUT.PUT\_LINE('VIP:' || c.Name || ' | Balance:' || c.Balance);

    END LOOP;

    DBMS\_OUTPUT.PUT\_LINE('CASE3:');

    FOR l IN (

        SELECT c.Name, l.LoanID, l.EndDate

        FROM Loans l

        JOIN Customers c ON c.CustomerID = l.CustomerID

        WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

    )

    LOOP

        DBMS\_OUTPUT.PUT\_LINE('Reminder:' || l.LoanID || ' for ' || l.Name ||

                             TO\_CHAR(l.EndDate, 'YYYY-MM-DD'));

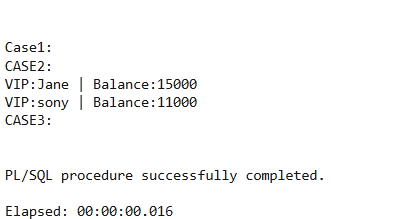
    END LOOP;

    COMMIT;

END;

/

Output:



Case1: is empty because there are no people whose age is below 60.

Exercise 3: Stored Procedures:

Code:

SET SERVEROUTPUT ON;

BEGIN

    DBMS\_OUTPUT.PUT\_LINE('Case1:');

    FOR ac IN (

        SELECT AccountID, Balance

        FROM Accounts

        WHERE UPPER(AccountType) IN 'SAVINGS'

    )

    LOOP

        DBMS\_OUTPUT.PUT\_LINE('Account: ' || ac.AccountID ||

                             ' With Balance = ' || TO\_CHAR(ac.Balance \* 0.01));

    END LOOP;

    DBMS\_OUTPUT.PUT\_LINE('case2:');

    FOR emp IN (

        SELECT Name, Salary

        FROM Employees

        WHERE UPPER(Department) = 'IT'

    )

    LOOP

        DBMS\_OUTPUT.PUT\_LINE(emp.Name ||

                             ': Bonus for the employee will be:  ' || TO\_CHAR(emp.Salary \* 0.10));

    END LOOP;

    -- Scenario 3: Simulated Fund Transfer

    DBMS\_OUTPUT.PUT\_LINE('case3:');

    DECLARE

        us1 NUMBER;

        us2   NUMBER;

    BEGIN

        SELECT Balance INTO us1 FROM Accounts WHERE AccountID = 4;

        SELECT Balance INTO us2   FROM Accounts WHERE AccountID = 5;

        IF us1 >= 500 THEN

            DBMS\_OUTPUT.PUT\_LINE('success');

            DBMS\_OUTPUT.PUT\_LINE('New (after transaction) ' || (us1 - 500));

            DBMS\_OUTPUT.PUT\_LINE('New (after transcation): ' || (us2 + 500));

        ELSE

            DBMS\_OUTPUT.PUT\_LINE('Transfer failed');

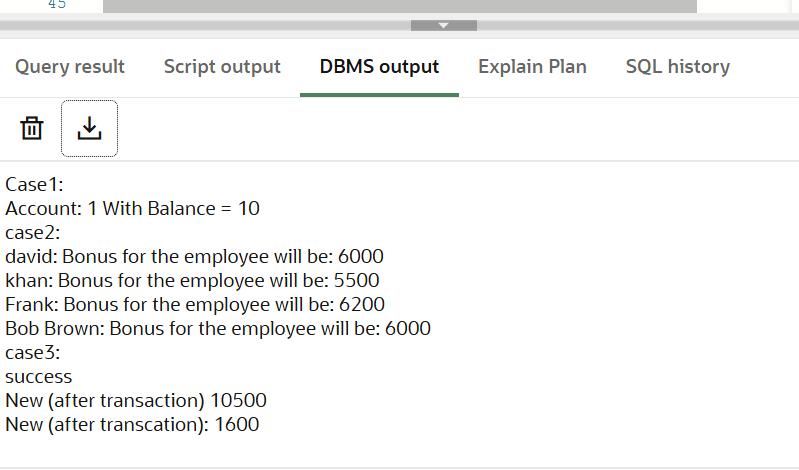
        END IF;

    END;

END;

/

Output:



JUNIT TESTING , Mockito and SL4J Logging

Exercise 1: Setting Up Junit:

Code:

Square.java:

package com.example;

public class Square {

public double square(double a) {

return a \* a;

}

} SquareTest.java:

package com.example;

import static org.junit.Assert.\*;

import org.junit.Test;

public class SquareTest {

Square s = new Square();

*@Test*

public void test1() {

double res = s.square(3);

*assertEquals*(9.0, res, 0.01);

}

*@Test*

public void test2() {

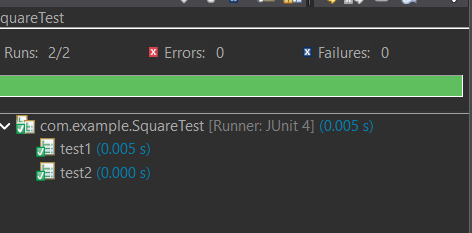
double res = s.square(8);

*assertEquals*(64.0, res, 0.001);

}

}

Output:



Exercise 3: Assertions in Junit:

Code:

package com.example;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AssertionTest {

*@Test*

public void testAssertions() {

// Assert equals

*assertEquals*(5, 2 + 3);

// Assert true

*assertTrue*(5 > 3);

// Assert false

*assertFalse*(5 < 3);

// Assert null

*assertNull*(null);

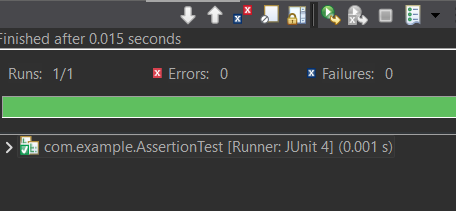
// Assert not null

*assertNotNull*(new Object());

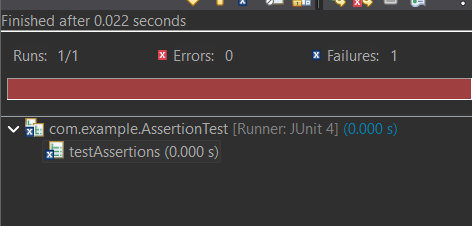
}

}

Output:



*When assertTrue*(5 < 3);



Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in Junit:

Code:

package com.example;

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

public class SquareTest {

private Square s;

*@Before*

public void setUp() {

System.***out***.println("Before");

s = new Square();

}

*@After //we can also use @afterall, @aftereeach,@afterclass*

public void tearDown() {

System.***out***.println("Teardown: Nullifying Calculator object");

s = null; // db.close();

}

*@Test*

public void square(){

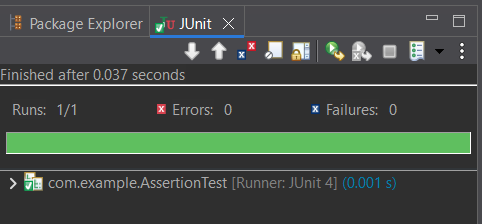
double res = s.square(5.0);

*assertEquals*(25.0, res,0.01);

}

}

Output:



Exercise 1: Mocking and Stubbing:

Code: External.java

package com.example;

public interface External {

String getValue();

}

Service.java:

package com.example;

public class Service {

private External ext;

public Service(External ext) {

this.ext =ext;

}

public String fetchData() {

return ext.getValue();

}

}

ServiceTest.java:

package com.example;

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

import static org.mockito.Mockito;

import org.junit.jupiter.api.Test;

public class ServiceTest {

*@Test*

public void testFetchData\_WithMockApi() {

External mockApi = *mock*(External.class);

*when*(mockApi.getValue()).thenReturn("Mock Data");

Service serv = new Service(mockApi);

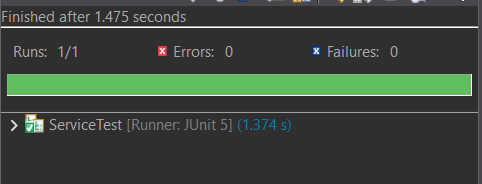
String res = serv.fetchData();

*assertEquals*("Mock Data", res);

}

}

Output:



Exercise 2: Verifying Interactions:

Code:

Service.java, External.java were used from above code formatons and the code that is given as solution is executed. The only need is to change the dependencies in maven.

package com.example;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class ServiceTest {

*@Test*

public void testVerifyInteraction() {

External mockApi = Mockito.*mock*(External.class);

Service service = new Service(mockApi);

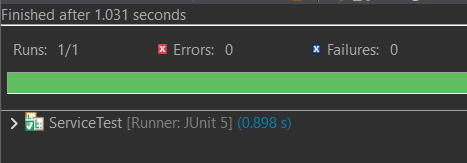
service.fetchData();

*verify*(mockApi).getValue();

}

}

Output:



Exercise 1: Logging Error Messages and Warning Levels:

Code:

// as maven code and Logger example code was given already to run this application we further needed is an xml document

Log.xml:

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{yyyy-dd-mm} -%msg%n</pattern>

</encoder>

</appender>

<root level="debug">

<appender-ref ref="STDOUT" />

</root>

</configuration>

Output:

