WEEK -2 HANDS ON EXERCISES

PL/SQL PROGRAMMING

1. Control Structures

```
SET SERVEROUTPUT ON;

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

CREATE TABLE CUSTOMERS (
   CUST_ID     NUMBER PRIMARY KEY,
   NAME        VARCHAR2(100),
   DOB         DATE,
   BALANCE     NUMBER(10,2),
   ISVIP       VARCHAR2(5) DEFAULT 'FALSE'
);

CREATE TABLE LOANS (
   LOAN_ID         NUMBER PRIMARY KEY,
   CUST_ID         NUMBER REFERENCES CUSTOMERS(CUST_ID),
   INTEREST_RATE   NUMBER(5,2),
   DUE_DATE        DATE
);
```

```
INSERT INTO CUSTOMERS VALUES (1, 'John Doe',
TO_DATE('1960-06-01', 'YYYY-MM-DD'), 15000, 'FALSE');
INSERT INTO CUSTOMERS VALUES (2, 'Jane Smith',
TO_DATE('1985-03-15', 'YYYY-MM-DD'), 8000, 'FALSE');
INSERT INTO CUSTOMERS VALUES (3, 'Robert Brown',
TO_DATE('1955-12-20', 'YYYY-MM-DD'), 12000, 'FALSE');
INSERT INTO CUSTOMERS VALUES (4, 'Alice Green',
TO_DATE('2000-01-01', 'YYYY-MM-DD'), 3000, 'FALSE');

INSERT INTO LOANS VALUES (101, 1, 8.5, SYSDATE + 10);
INSERT INTO LOANS VALUES (102, 2, 7.0, SYSDATE + 40);
INSERT INTO LOANS VALUES (103, 3, 9.0, SYSDATE + 5);
INSERT INTO LOANS VALUES (104, 4, 6.5, SYSDATE + 25);

COMMIT;

-- Scenario 1: Apply 1% discount to interest rates for customers above 60
BEGIN
   FOR rec IN (
      SELECT l.LOAN_ID
      FROM CUSTOMERS c
      JOIN LOANS l ON c.CUST_ID = l.CUST_ID
      WHERE  TRUNC(MONTHS_BETWEEN(SYSDATE,  c.DOB)/12)
> 60
   ) LOOP
      UPDATE LOANS
      SET INTEREST_RATE = INTEREST_RATE - 1
      WHERE LOAN_ID = rec.LOAN_ID;
   END LOOP;
```

```
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(' Scenario 1: 1% interest discount applied for customers above 60.');
END;
/

-- Scenario 2: Set IsVIP = TRUE for customers with balance > $10,000
BEGIN
    FOR rec IN (
        SELECT CUST_ID
        FROM CUSTOMERS
        WHERE BALANCE > 10000
    ) LOOP
        UPDATE CUSTOMERS
        SET ISVIP = 'TRUE'
        WHERE CUST_ID = rec.CUST_ID;
    END LOOP;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE(' Scenario 2: VIP status set for customers with balance > $10,000.');
END;
/

-- Scenario 3: Print reminders for loans due within the next 30 days
BEGIN
    FOR rec IN (
        SELECT l.LOAN_ID, l.DUE_DATE, c.NAME
        FROM LOANS l
```

```
      JOIN CUSTOMERS c ON l.CUST_ID = c.CUST_ID
      WHERE l.DUE_DATE BETWEEN SYSDATE AND SYSDATE + 30
   ) LOOP
      DBMS_OUTPUT.PUT_LINE(
         'Reminder: Dear ' || rec.NAME || ', your loan ID ' || rec.LOAN_ID
||
         ' is due on ' || TO_CHAR(rec.DUE_DATE, 'DD-MON-YYYY')
      );
   END LOOP;
END;
/

SELECT * FROM CUSTOMERS;
SELECT * FROM LOANS;
```

OUTPUT:

```
Scenario 1: 1% interest discount applied for customers above 60.
Scenario 2: VIP status set for customers with balance > $10,000.
 Reminder: Dear John Doe, your loan ID 101 is due on 07-JUL-2025
 Reminder: Dear Robert Brown, your loan ID 103 is due on 02-JUL-2025
 Reminder: Dear Alice Green, your loan ID 104 is due on 22-JUL-2025
```

2. Stored Procedures

```
CREATE TABLE SavingsAccounts (
   AccountID INT PRIMARY KEY,
   AccountHolderName VARCHAR(100),
   Balance DECIMAL(10, 2)
);

INSERT INTO SavingsAccounts VALUES
```

```
(101, 'Alice', 1000.00),
(102, 'Bob', 2000.00),
(103, 'Charlie', 1500.00);

CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100),
    Department VARCHAR(50),
    Salary DECIMAL(10, 2)
);

INSERT INTO Employees VALUES
(1, 'David', 'Sales', 30000.00),
(2, 'Eve', 'HR', 25000.00),
(3, 'Frank', 'Sales', 28000.00);

CREATE TABLE BankAccounts (
    AccountID INT PRIMARY KEY,
    CustomerName VARCHAR(100),
    Balance DECIMAL(10,2)
);

INSERT INTO BankAccounts VALUES
(201, 'George', 5000.00),
(202, 'Helen', 3000.00);

DELIMITER //

CREATE PROCEDURE ProcessMonthlyInterest()
BEGIN
```

```
    UPDATE SavingsAccounts
    SET Balance = Balance + (Balance * 0.01);
END //

CREATE PROCEDURE UpdateEmployeeBonus(IN dept VARCHAR(50),
IN bonus_percent DECIMAL(5, 2))
BEGIN
    UPDATE Employees
    SET Salary = Salary + (Salary * bonus_percent / 100)
    WHERE Department = dept;
END //

CREATE PROCEDURE TransferFunds(
    IN from_account INT,
    IN to_account INT,
    IN amount DECIMAL(10,2)
)
BEGIN
    DECLARE from_balance DECIMAL(10,2);
    SELECT Balance INTO from_balance
    FROM BankAccounts
    WHERE AccountID = from_account;
    IF from_balance >= amount THEN
        UPDATE BankAccounts
        SET Balance = Balance - amount
        WHERE AccountID = from_account;
        UPDATE BankAccounts
        SET Balance = Balance + amount
        WHERE AccountID = to_account;
    ELSE
```

```
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Insufficient funds';
    END IF;
END //

DELIMITER ;

CALL ProcessMonthlyInterest();
CALL UpdateEmployeeBonus('Sales', 10);
CALL TransferFunds(201, 202, 1000.00);

SELECT * FROM SavingsAccounts;
SELECT * FROM Employees;
SELECT * FROM BankAccounts;
```

OUTPUT:

ACCOUNTID	ACCOUNTHOLDERNAME	BALANCE
101	Alice	1010
102	Bob	2020
103	Charlie	1515

EMPLOYEEID	NAME	DEPARTMENT	SALARY
1	David	Sales	33000
2	Eve	HR	25000
3	Frank	Sales	30800

ACCOUNTID	CUSTOMERNAME	BALANCE
201	George	4000
202	Helen	4000
ACCOUNTID	CUSTOMERNAME	BALANCE
201	George	4000
202	Helen	4000

UNIT TESTING

1. Setting up unit

```
//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);
    }
}
```

//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>JUnitSetup</artifactId>
  <version>1.0-SNAPSHOT</version>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.13.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

OUTPUT:

```
INFO]
INFO] -------------------------------------------------------
INFO]  T E S T S
INFO] -------------------------------------------------------
INFO] Running CalculatorTest
INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.092 s -- in Calcul
torTest
INFO]
INFO] Results:
INFO]
INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
INFO]
INFO] ------------------------------------------------------------------------
INFO] BUILD SUCCESS
INFO] ------------------------------------------------------------------------
INFO] Total time:  2.188 s
INFO] Finished at: 2025-06-26T21:45:23+05:30
INFO] ------------------------------------------------------------------------
```

2. Assertions in Junit

```
//Assertion Test.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());
    }
}
```

OUTPUT:

```
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running AssertionsTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.187 s -- in Assert
ionsTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  8.579 s
[INFO] Finished at: 2025-06-26T21:51:38+05:30
```

3. 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;
    }
}
```

//CalulatorTest.java

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

public class CalculatorTest {

    private Calculator calculator;

    @Before
    public void setUp() {
        // Arrange - Setup before each test
        calculator = new Calculator();
        System.out.println("Setup completed.");
    }

    @After
    public void tearDown() {
        // Cleanup after each test
        calculator = null;
        System.out.println("Teardown completed.");
    }
```

```
    @Test
    public void testAddition() {
        // Act
        int result = calculator.add(2, 3);

        // Assert
        assertEquals(5, result);
    }
}
```

OUTPUT:

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running CalculatorTest
Setup completed.
Teardown completed.
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.228 s -- in Calcul
atorTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.718 s
[INFO] Finished at: 2025-06-26T21:55:21+05:30
```

MOKITO

1. Mocking and Stubbing

```
//ExternalApi.java
public interface ExternalApi {
String getData();}
```

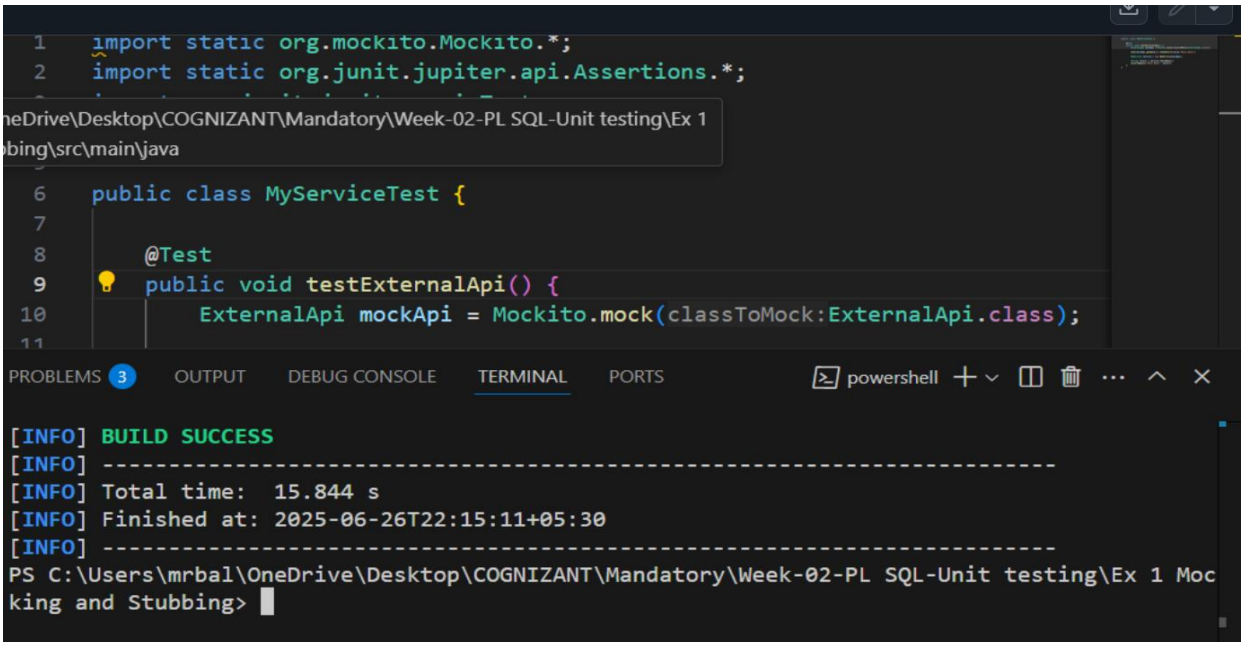
```
//MyServicrTest.java
import static org.mockito.Mockito.*;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;

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);
    }
}
```

```
//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>MockitoMocking</artifactId>
```

```
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>4.11.0</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.8.2</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>5.8.2</version>
            <scope>test</scope>
        </dependency
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
```

```
            <version>2.22.2</version>
        </plugin>
    </plugins>
  </build>
</project>
```

OUTPUT:



2. Verifying Instructions

```
//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.*;
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();
    }
}
```

OUTPUT:

```
[INFO] -------------------------------------------------------
[INFO] Running MyServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.521 s - in MyServi
ceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.866 s
[INFO] Finished at: 2025-06-26T22:22:07+05:30
```