WEEK-2 HANDS ON

PL/SQL

Exercise 1: Control Structures

- Create required tables “customers” and “loans” with required fields. And insert data into the tables. Commit the tables.
- Initially set the isvip columns to False.

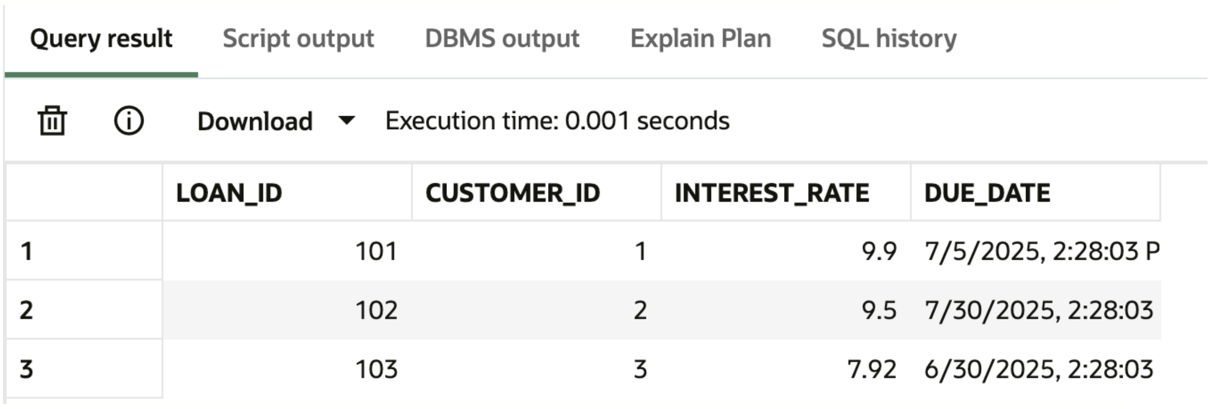
```
Create table customers (
   customer_id number primary key,
   customer_name varchar(100),
   age number,
   balance number,
   isvip varchar(5)
);

insert into customers values (1, 'ABC', 67, 15000, 'FALSE');
insert into  customers values (2, 'DEF', 45, 9000, 'FALSE');
insert into  customers values (3, 'ghi', 71, 12000, 'FALSE');


Create table loans (
   loan_id number primary key,
   customer_id number,
   interest_rate number,
   due_date date,
   foreign key (customer_id) references customers(customer_id)
);

insert into  loans values (101, 1, 10.0, SYSDATE+10 );
insert into  loans values (102, 2, 9.5, SYSDATE+35 );
insert into  loans values (103, 3, 8.0, SYSDATE+5 );


commit;
```

Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.

- **Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

```
begin

 for n in (select customer_id from customers where age > 60)

loop

update loans set interest_rate = interest_rate - (interest_rate*0.01)
```

```
where customer_id = n.customer_id;

end loop;

  commit;

end;

/

Select * from loans;
```



Query result | Script output | DBMS output | Explain Plan | SQL history

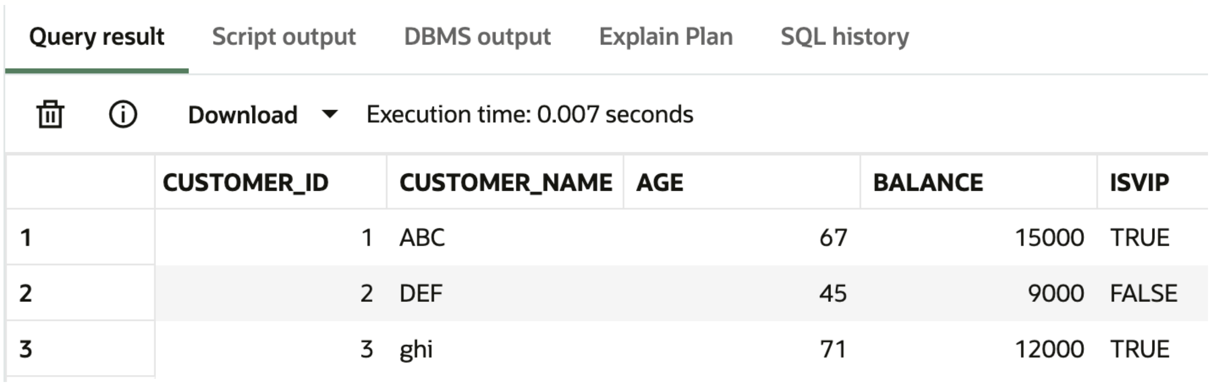
Download ▾ Execution time: 0.001 seconds

	LOAN_ID	CUSTOMER_ID	INTEREST_RATE	DUE_DATE
1	101	1	9.9	7/5/2025, 2:28:03 P
2	102	2	9.5	7/30/2025, 2:28:03
3	103	3	7.92	6/30/2025, 2:28:03

Scenario 2: A customer can be promoted to VIP status based on their balance.

- **Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

```
begin

 for r in ( select customer_id  from customers where balance > 10000)

loop

update customers set isvip = 'TRUE where customer_id = r.customer_id;

end loop;

commit;

end;

/
```


Query result | Script output | DBMS output | Explain Plan | SQL history

Download ▼ Execution time: 0.007 seconds

	CUSTOMER_ID	CUSTOMER_NAME	AGE	BALANCE	ISVIP
1	1	ABC	67	15000	TRUE
2	2	DEF	45	9000	FALSE
3	3	ghi	71	12000	TRUE

Scenario 3: The bank wants to send reminders to customers whose loans are due within the next 30 days.

- **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

```
begin

  for r in ( select loan_id, due_date, customer_id from loans where due_date between sysdate and
sysdate + 30)

loop

declare

customer_name customers.customer_name%type;

 begin

 select customer_name into customer_name from customers where customer_id = r.customer_id;

 dbms_outline.put_line(  'Reminder: Loan ID ' || r.loan_id ||  ' is due on ' || to_char(r.due_date,
'DD-MON-YYYY') ||  ' for customer ' || customer_name);

   end;

  end loop;

end;

/
```

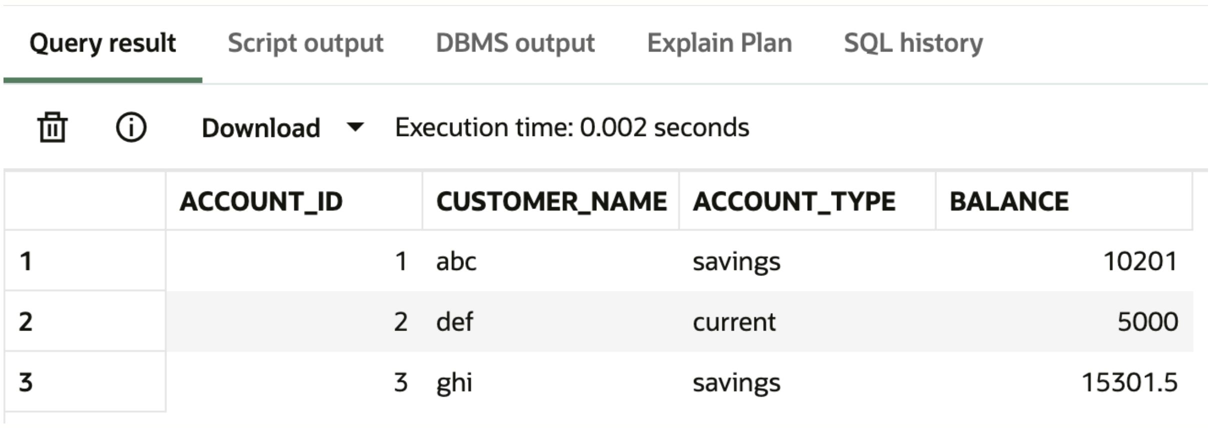
```
Reminder: Loan ID 101 is due on 05-JUL-2025 for customer ABC
Reminder: Loan ID 103 is due on 30-JUN-2025 for customer ghi


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.014
```

Exercise 2: Stored Procedures

Scenario 1: The bank needs to process monthly interest for all savings accounts.

- **Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

```
create table accounts ( account_id number primary key,  customer_name varchar2(100),
account_type varchar2(20),  balance number);

insert into accounts values (1, 'abc', 'savings', 10000);

insert into accounts values (2, 'def', 'current', 5000);

insert into accounts values (3, 'ghi', 'savings', 15000);

create or replace procedure ProcessMonthlyInterest is

begin

 update accounts set balance = balance + (balance * 0.01) where account_type = 'savings';

 commit;

end;

/

Exec ProcessMonthlyInterest;

Select * from accounts;
```

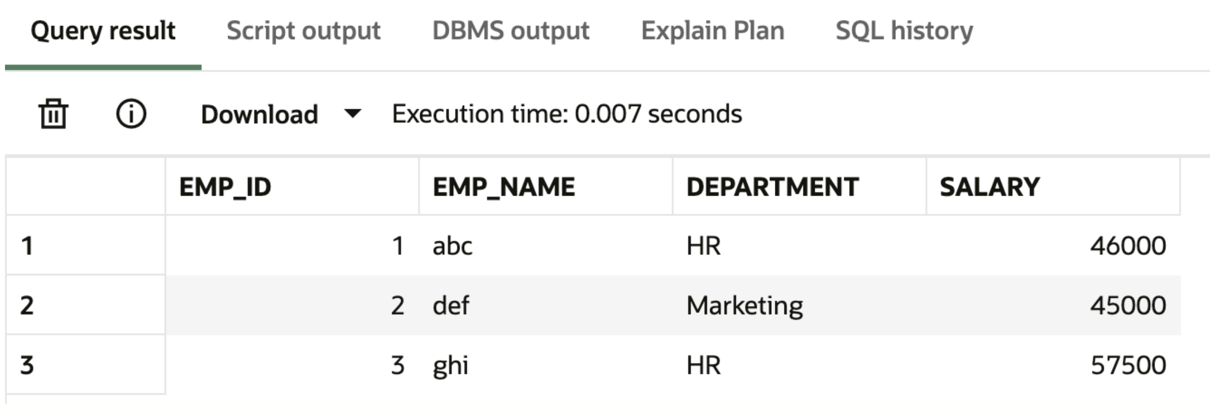


	ACCOUNT_ID	CUSTOMER_NAME	ACCOUNT_TYPE	BALANCE
1	1	abc	savings	10201
2	2	def	current	5000
3	3	ghi	savings	15301.5

Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.

- **Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

```
create table employees ( emp_id number primary key, emp_name varchar2(100)  department varchar2(50), salary number);

insert into employees values (1, 'abc', 'HR', 40000);

insert into employees values (2, 'def', 'Marketing', 45000);

insert into employees values (3, 'ghi', 'HR', 50000);

create or replace procedure UpdateEmployeeBonus is

begin

 update employees set salary = salary + (salary * 15 / 100)where department = 'HR';

commit;

end;

/

exec UpdateEmployeeBonus;

select * from employees;
```



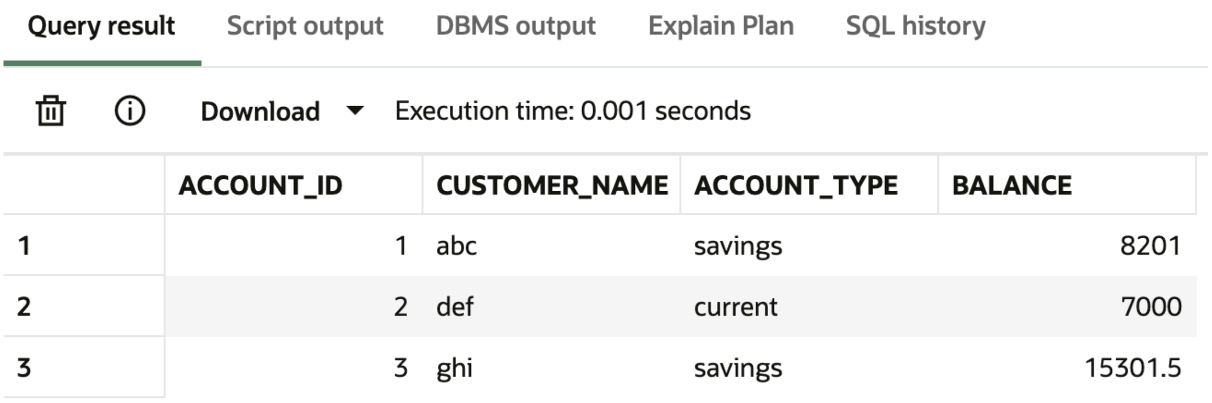
	EMP_ID	EMP_NAME	DEPARTMENT	SALARY
1	1	abc	HR	46000
2	2	def	Marketing	45000
3	3	ghi	HR	57500

Scenario 3: Customers should be able to transfer funds between their accounts.

- **Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

We make use of the accounts table for this procedure.

```
create or replace procedure TransferFunds(from_acc in number, to_acc in number, amount in number
) is
from_balance number;
insufficient_balance exception;
begin
  select balance into from_balance from accounts where account_id=from_acc;
  if from_balance < amount then
  raise insufficient_balance;
  end if;
  update accounts set balance= balance - amount where account_id=from_acc;
  update accounts set balance= balance + amount where account_id=to_acc;
  commit;
exception
  when insufficient_balance then
  dbms_output.put_line('Balance is not sufficient.');
end;
/

exec TransferFunds(1,2,2000);

select * from accounts;
```



Execution time: 0.001 seconds

	ACCOUNT_ID	CUSTOMER_NAME	ACCOUNT_TYPE	BALANCE
1	1	abc	savings	8201
2	2	def	current	7000
3	3	ghi	savings	15301.5

TDD using JUnit5 and Mockito

Exercise 1: Setting Up JUnit

- First we have to create a project named “JUnitDemo”.
- Add the package com.example and add the classes “Calculator” and “CalculatorTest”.
- Then, we go to properties of the project > Java Build Path > Libraries.
- Under ClassPath, we need to add a new library “JUnit”.

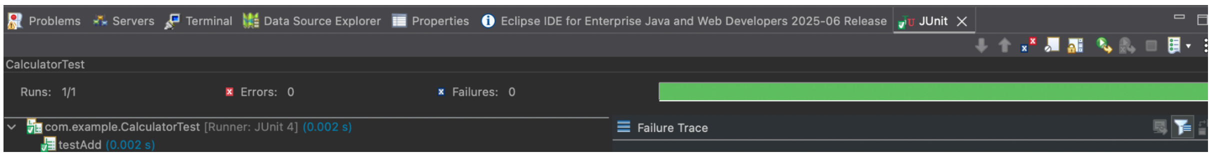
Calculator.java

```
package com.example;
public class Calculator {
   public int add(int a, int b) {
       return a + b;
   }
}
```

CalculatorTest.java

```
package com.example;
import static org.junit.Assert.*;
import org.junit.Test;
public class CalculatorTest {
   Calculator calculator = new Calculator();
   @Test
   public void testAdd() {
       assertEquals(5, calculator.add(2, 3));
   }
}
```

Output:



```
CalculatorTest
Runs:  1/1          Errors:  0          Failures:  0

com.example.CalculatorTest [Runner: JUnit 4] (0.002 s)
    testAdd (0.002 s)
```

Exercise 3: Assertions in JUnit

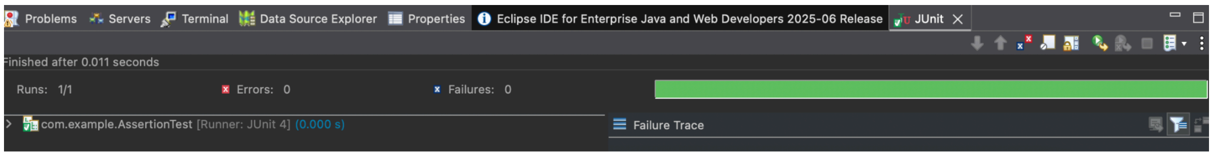
- In the same project, add another class to test the assertions.

AssertionTest.java

```
package com.example;
import org.junit.Test;
import static org.junit.Assert.*;
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:



```
Problems  Servers  Terminal  Data Source Explorer  Properties  Eclipse IDE for Enterprise Java and Web Developers 2025-06 Release  JUnit
inished after 0.011 seconds
Runs: 1/1          Errors: 0          Failures: 0
> com.example.AssertionTest [Runner: JUnit 4] (0.000 s)          Failure Trace
```

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

- Create a new class “CalculatorTestWithSetup” and “Calculator” (a previous class).

CalculatorTestWithSetup.java

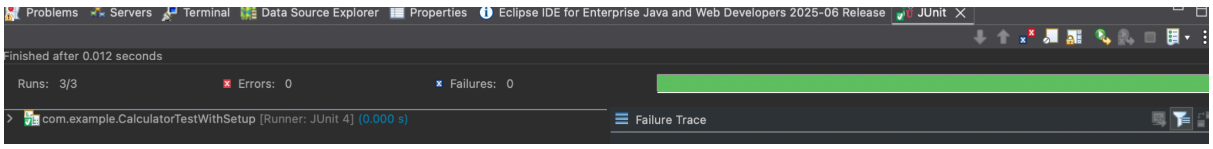
```
package com.example;
import org.junit.Before;
import org.junit.After;
import org.junit.Test;
import static org.junit.Assert.*;
public class CalculatorTestWithSetup {
    private Calculator calculator;
    @Before
    public void setUp() {
        calculator = new Calculator();
        System.out.println("Setup done");
    }
    @After
    public void tearDown() {
        calculator = null;
        System.out.println("Teardown done");
    }
    @Test
    public void testAddition() {
        int result = calculator.add(2, 3);
        assertEquals(5, result);
    }
    @Test
    public void testSubtraction() {
        int result = calculator.subtract(10, 3);
        assertEquals(7, result);
    }
```

```
  @Test
  public void testMultiplication() {
    int result = calculator.multiply(4, 5);
    assertEquals(20, result);
  }
}
```

Calculator.java

```
package com.example;
public class Calculator {
  public int add(int a, int b) { return a + b; }
  public int subtract(int a, int b) { return a - b; }
  public int multiply(int a, int b) { return a * b; }
}
```

Output:



```
Finished after 0.012 seconds
Runs: 3/3          Errors: 0          Failures: 0
> com.example.CalculatorTestWithSetup [Runner: JUnit 4] (0.000 s)
```

Mockito

Exercise 1: Mocking and Stubbing

- Create a Maven project structure in eclipse IDE.
- Modify the pom.xml file by adding the required dependencies.

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>MockitoDemo</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 <packaging>jar</packaging>
 <name>MockitoDemo</name>
 <url>http://maven.apache.org</url>
 <properties>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 </properties>
 <dependencies>
   <dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>3.8.1</version>
     <scope>test</scope>
   </dependency>
```

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

  <dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>4.8.1</version>
    <scope>test</scope>
  </dependency>
 </dependencies>
</project>
```

- Create files “ExternalApi.java” and “MyService.java” in the src/main folder.
- Create file “MyServiceTest.java” in src/test folder.

ExternalApi.java

```
package com.example.MockitoDemo;
public interface ExternalApi {
  String getData();
  int getStatusCode();
}
```

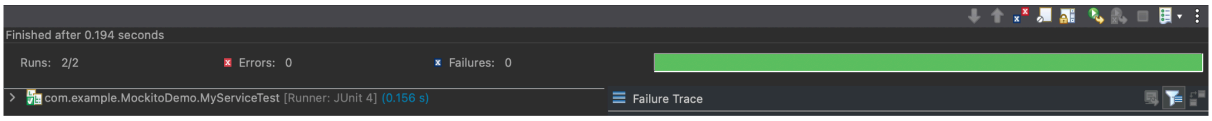
MyService.java

```
package com.example.MockitoDemo;
public class MyService {
  private ExternalApi externalApi;
  public MyService(ExternalApi externalApi) {
    this.externalApi = externalApi;
  }
  public String fetchData() {
    return externalApi.getData();
  }
  public int checkStatus() {
    return externalApi.getStatusCode();
  }
}
```

MyServiceTest.java

```
package com.example.MockitoDemo;
import org.junit.Test;
import static org.junit.Assert.*;
import static org.mockito.Mockito.*;
public class MyServiceTest {
  @Test
  public void testFetchData() {
        ExternalApi mockApi = mock(ExternalApi.class);
        when(mockApi.getData()).thenReturn("Mocked Data");
        MyService service = new MyService(mockApi);
        String result = service.fetchData();
        assertEquals("Mocked Data", result);
        verify(mockApi).getData();
    }
    @Test
    public void testCheckStatus() {
        ExternalApi mockApi = mock(ExternalApi.class);
        when(mockApi.getStatusCode()).thenReturn(200);
        MyService service = new MyService(mockApi);
        int status = service.checkStatus();
        assertEquals(200, status);
        verify(mockApi).getStatusCode();
    }
}
```

Output:



```
Finished after 0.194 seconds
Runs: 2/2        Errors: 0        Failures: 0
> com.example.MockitoDemo.MyServiceTest [Runner: JUnit 4] (0.156 s)
```

Exercise 2: Verifying Interactions

It has the same directory structure as the previous one.

ExternalApi.java

```
package com.example.MockitoDemo;
public interface ExternalApi {
    String getData();
    int getStatusCode();
}
```

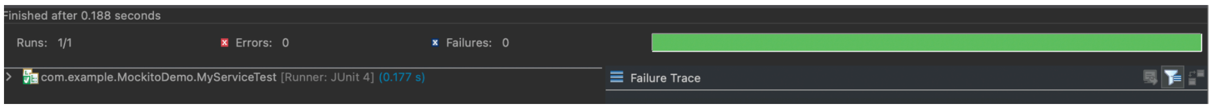
MyService.java

```
package com.example.MockitoDemo;   //this code for verifying interactions
public class MyService {
    private ExternalApi externalApi;
    public MyService(ExternalApi externalApi) {
        this.externalApi = externalApi;
    }
    public String fetchData() {
        return externalApi.getData();
    }
}
```

MyServiceTest.java

```
package com.example.MockitoDemo;
import org.junit.Test;
import static org.junit.Assert.*;
import static org.mockito.Mockito.*;
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:



```
inished after 0.188 seconds
Runs: 1/1        Errors: 0        Failures: 0
com.example.MockitoDemo.MyServiceTest [Runner: JUnit 4] (0.177 s)
Failure Trace
```

SL4J Logging Exercise

Exercise 1: Logging Error Messages and Warning Levels

- Create a maven project.
- Modify the pom.xml file by adding the dependencies.

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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.yourcompany</groupId> <artifactId>JUnitExampleProject</artifactId>
<version>1.0.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <properties>
      <maven.compiler.source>11</maven.compiler.source>
      <maven.compiler.target>11</maven.compiler.target>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
      <dependency>
          <groupId>junit</groupId>
          <artifactId>junit</artifactId>
          <version>4.13.2</version>
          <scope>test</scope> </dependency>
          <dependency>
```

```
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>4.11.0</version> <scope>test</scope>
        </dependency>
         <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>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version> <configuration>
                    <source>${maven.compiler.source}</source>
                    <target>${maven.compiler.target}</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

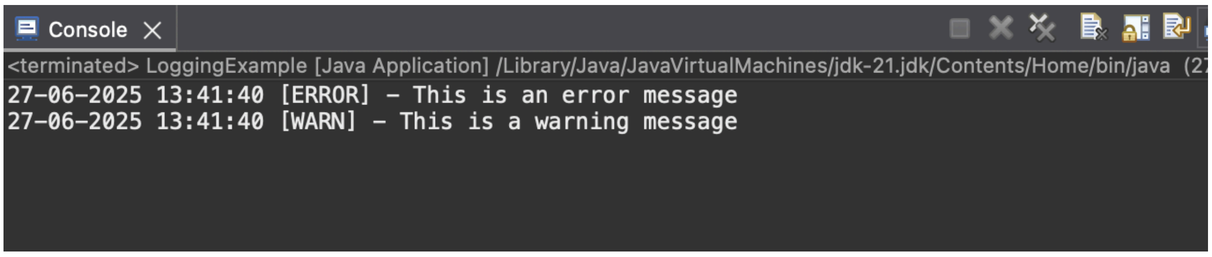
LoggingExample.java

```
package com.example.LoggingDemo;
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");
    }
}
```

Logging.xml

```
<configuration>
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{dd-MM-YYYY HH:mm:ss} [%level] - %msg%n</pattern>
        </encoder>
    </appender>
    <root level="debug">
```

```
        <appender-ref ref="STDOUT" />
    </root>
</configuration>
```



```
<terminated> LoggingExample [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (27
27-06-2025 13:41:40 [ERROR] - This is an error message
27-06-2025 13:41:40 [WARN] - This is a warning message
```