**Problem Statement**

1. **Explain the working and the differences between Maven, Gradle and SBT in detail.**

**Maven-**

Maven is a build tool which are programs that automate the creation of executable applications from source code. In small projects, developers will often manually invoke the build process. This is not practical for larger projects, where it is very hard to keep track of what needs to be built, in what sequence and what dependencies there are in the building process. Using an automation tool allows the build process to be more consistent.  
There are different types of Build tools available for different types of languages. For example  
For java – Ant,Maven,Gradle.  
For .NET framework – NAnt  
For c# – MsBuild.  
So Maven is a Build tool for java applications

Maven can do the following things-

**-Build**  
**-Document**  
**-Dependency Management**  
**-Distributions**  
**-Test**  
**-Report**  
**-SCM’s**  
**-Releases**

Maven needs an XML file, that XML file is called as POM file stands for Project Object Model.

**Building through Maven-**

Maven is based around the central concept of build life cycle. For building Maven it is only necessary to learn a small set of commands to build any Maven project, and the [POM](https://maven.apache.org/guides/introduction/introduction-to-the-pom.html) will ensure they get the results they desired.

**Life Cycle of Maven-**

Life cycle of Maven comprises of following phases-

* validate – validate the project is correct and all necessary information is available
* to compile – compile the source code of the project
* test – test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
* package – take the compiled code and package it in its distributable format, such as a JAR.
* verify – run any checks on results of integration tests to ensure quality criteria are met
* install – install the package into the local repository, for use as a dependency in other projects locally
* deploy – done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

**Gradle-**

Gradle is also an open source build automation system builds upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language (DSL) instead of the XML form used by Apache Maven of declaring the project configuration. Each gradle project consists of *tasks*. A task represents a piece of work which a build performs, e.g., compile the source code or generate the Javadoc.

A project using Gradle describes its build via a build.gradle file. This file is located in the root folder of the project. The build file for Gradle builds is based on a Domain Specific Language(DSL). Tasks can also be created and extended dynamically at runtime.

Gradle project using the command **Gradle build**. This command will compile your code and will build a jar file automatically.

After successful compilation and building of the jar, you can see a success message and in the **project\_root\_folder/build/libs**directory, you can see the jar file.

To make this project eclipse ready, you need to use the command **sbt eclipse**to build the project and classpath that are required for eclipse.

This build file defines a project and its tasks. Gradle is a general purpose build system hence this build file can perform any task

To execute any task in Gradle, type

Gradle <task name>

In the command line in the directory of the build file. If the Gradle output should be suppressed, use the -q (quiet) parameter.

**SBT**

**SBT**is an open source build tool for Scala and Java projects, similar to Java’s Maven or Ant. Its main features are native support for compiling Scala code and integrating with many Scala test frameworks.

For using SBT it should be installed on Linux plateform.

Building SBT project

First, you need to create one Project folder. Here inside the **src/main/java**directory, you need to write your code, otherwise, SBT could not find your main class file.

Now, we have created one file called **Hadoop\_app\_sbt**and inside **src/main/java**we have pasted the code of Hadoop word count program.

Inside the root directory, i.e., **Hadoop\_app\_sbt**you need to create one more file called **build.sbt**here you need to provide all the specifications and library dependencies of your project.

Now, we need to compile the code to check whether it has any errors or not. You can do that by using the command **sbt compile**. Move into the root directory where **build.sbt**is present. After compilation, all the library files that are required for your project will get downloaded. Now you can import this project into Eclipse.

For that, you need to use the command **sbt eclipse**.This will build the project files that are required for Eclipse

You can see your imported project in the list of Eclipse projects. You can do modifications when needed.

Now for building the JAR file, you need to use the command **sbt package.**After successfully packaging the class files as a JAR, you can see the jar file in the **target directory**.

Now you can run the JAR file normally as a Hadoop JAR file

you can see the output of our word count program which is packaged using SBT.

**Differences**

**SBT-** SBT is very simple and it is focused on Scala it relies on Ivy for dependency management.

**MAVEN-**Maven it's a great build tool and it enables to control the entire software lifecycle with XML files. Using the Project Object Model you can intercept all points of the software lifecycle from compile to test, packaging and deploy. Maven has it's own dependency manager.

**GRADLE-**Gradle is built on top of Maven, Ant and Ivy. It uses Maven repositories. Gradle doesn't use XML, it's a polyglot build tool. It combines the Ant API with the  Groovy language to enable developers to write a build script with an intuitive DSL. With a few lines of code you can write a Gradle build script that can do the same things that Maven can do. With Gradle you can define your own task with the Groovy language and intercept programmatically your build execution.