1. **What is Maven and why is it used?**
   Maven is a powerful build automation tool mainly used for Java-based projects. It simplifies the build process and project management by providing a standard approach. Maven uses a pom.xml to configure dependencies, plugins, and various project settings. It reduces the need for manual scripting.
   Use of Maven :
   • Managing large-scale Java projects with multiple dependencies.
   • Making a standardized build process across development teams.
   • Automating repetitive tasks such as testing and packaging.
   • Integrating with CI/CD pipelines to streamline deployments.

2. **Explain the POM file in Maven.**
   The pom.xml (Project Object Model) file is the core configuration file in a Maven project. It contains essential information such as project dependencies, build configurations, plugin and various project settings in it. The pom.xml file allows developers and operators to manage dependencies and automate build processes efficiently.

3. **What are Maven coordinates and what do they represent?**

4. **How do you manage dependencies in Maven?**
   Dependencies in Maven means external libraries or modules required by a project. Maven automatically downloads these dependencies from repositories and manages their versions. The dependency management system prevents conflicts between different libraries.

5. **What is a Maven repository and what are its types?**
   Three Types of Maven Repositories, Maven uses a repository system to store and retrieve dependencies.
   The three main types are:
   1. Local Repository - A directory on the developer's machine where Maven caches downloaded dependencies.
   2. Central Repository - A globally maintained repository that provides publicly available libraries.
   3. Remote Repository - A private or organization-specific repository used to store custom dependencies.
6. **Explain the concept of Maven lifecycle phases.**
   Maven Lifecycle Maven follows a structured lifecycle consisting of three primary phases:
   1. Clean Lifecycle - Cleans up previous build artifacts.
   2. Default Lifecycle - Handles project compilation, testing, packaging, and deployment.
   3. Site Lifecycle - Generates project documentation.

7. **What are Maven goals and how do they differ from phases?**

8. **How do you create a Maven project?**

Step 1: Validate the Maven Project mvn validate This checks whether the pom.xml file is correct.
Step 2: Compile the Java Code mvn compile This converts Java source code into bytecode (.class files)
Step 3: Run Tests mvn test This executes all unit tests in the project.
Step 4: Package the Project mvn package This generates a .war file.
Step 5: Verify the Build mvn verify This checks if the generated package meets project requirements.
Step 6: Install the Package Locally mvn install This installs the package in Maven's local repository (~/.m2/repository/)

9. **What is a Maven plugin and how is it used?**
   Plugins in Maven extend its functionality and enable the execution of predefined tasks such as compiling code, running tests, packaging applications, and deploying artifacts.
   Some commonly used plugins include:
   • Compiler Plugin - Compiles Java source code.
   • Surefire Plugin - Runs unit tests.

10. **How do you handle versioning in Maven projects?**

11. **Explain the PEM file in maven?**
    The pom.xml (Project Object Model) file is the core configuration file in a Maven project. It contains essential information such as project dependencies, build configurations, plugin and various project settings in it. The pom.xml file allows developers and operators to manage dependencies and automate build processes efficiently.

12. **when we generate jar/war/ear file in target?**

    • JAR (Java Archive) – A JAR file is generated when we are using Java libraries/applications.
    • WAR (Web Application Archive) - A WAR file is used to package web applications.
    • EAR (Enterprise Archive) - A EAR file is used in enterprise applications that combine both JAR and WAR files.

13. **what is home directory in maven?**
    The Maven home directory is the location where Maven is installed..
    By default, it is set in the M2_HOME environment variable.

14. **where will be build files stored?**
    In Maven, build files are stored in the target/ directory of the project by default.
    Dependencies downloaded by Maven are stored in the local repository at ~/.m2/repository/.

15. **what is meant by build tool?**
A build tool in DevOps is a software tool that automates the process of transforming source code into an executable application form like compiling the source code, linking dependencies, packaging binaries, and preparing the application for deployment. Build tools are important in continuous integration and deployment (CI/CD) pipelines, and automation in software development.

16. **Explain the process of building in maven?**
<u>Maven Build Lifecycle/Workflow</u>
Building process in maven generally include 6 stages like

- clean - Removes compiled files
- Validate – ensures the pom.xml is correct
- compile - Compiles source code
- test - Runs unit tests
- package - Packages the project
- verify – checks if package meets the requirements
- install - Installs the package to local repo
- deploy - Deploys to a remote repo

17. **Does maven support all types of projects to build?**
Maven supports Java based projects mainly. This doesn't support all types of projects to build.

18. **what is the difference between compile & validate?**

Compile and validate are intial steps in Building the Maven Project, has different puroposes like

- **Validate** the Maven Project mvn validate This ensures the pom.xml file is correct.
- **Compile** the Java Code mvn compile This converts Java source code into bytecode (.class files)

19. **can you create only one jar file or can we create multiple,explain?**
Depending on the application, multiple jar/war files are created. For example in EAR file , which is used in enterprise archive applications, which consists of both jar and war files.

20. **What is Git and why is it used?**

GIT (global Information Tracker) is a open source version control system(VCS) developed by linus torvalds, This is used to track the changes in the code, manage the changes, manage the different versions of a system.

21. **Explain the difference between Git and other version control systems.**

a. Git is a distributed version control system, meaning every developer has a full copy of the repository.
b. Traditional VCS like SVN or CVS are centralized, requiring a connection to a central server.
c. Git offers branching and merging capabilities that are faster and more flexible compared to older VCS tools.
d. It supports offline work, making it more efficient for modern development workflows

22. **How do you initialize a Git repository?**

    To initialize a Git repository we use
    Git init
    This command will inialize a git repo, you can use this multiple times in git, to re-initalize a git repo.

23. **What is the purpose of the .gitignore file?**
    The .gitignore file is used to specify files and directories that should be ignored by Git and not tracked in a repository. This helps prevent unnecessary or sensitive files from being committed.

24. **How do you stage changes in Git?**

    To stage changes in GIT , we use
    Git add .
    This commad will stage all changes into git.
    Git add file_name
    This command will stage all changes in that particular file.

25. **What is the difference between git commit and git commit -m?**
    Git commit -m
    is a command used in git, to specify a message to clearly understand the changes made .
    we need to specify a message after -m in quotes, this message will be saved in commit history making it easier to go back to the changes and track the changes.
    Git commit -m “your message”
    Git commit , This command will commit the changes of stages files without a message. Making it difficult to track the necessary changes .

26. **How do you create a new branch in Git?**

    We can create a new branch in Git using

Git branch branch_name
This command will create a new branch with your specified name.
Git branch checkout branch_name
This command will create a new branch with your specified name and also switches to the new branch.

27. **What is the difference between git merge and git rebase?**
    git merge
    Combines two branches and keeps both histories
    Creates a merge commit
    When you want to keep track of branch history

    git rebase
    Moves your branch on top of another, making history linear
    Rewrites commit history
    When you want a clean, linear history
28. **How do you resolve merge conflicts in Git?**
    Whenever we come across the merge conflicts, we can resolve them manually with merge markers or you can cancel the merge request and make the changes and merge again.

29. **What is the purpose of git stash?**
    git stash temporarily saves your changes so you can switch branches without committing them.

30. **Explain the use of git pull and git fetch.**
    - *Git fetch*

      Downloads new changes from remote but does not merge them.
    - *Git pull*

      Downloads and merges changes into your branch