1. what is the difference between maven and gradle?

* **Gradle:-**
* Gradle id a built automation tool is an open source and builds based on the concepts of Apache Maven and Apache Ant
* The main goal is to add functionality to the project
* Gradle suppoerts for the building testing and deploying software on different platforms
* It supports dependency management
* It supports a multi-project structure
* It provides high performance and scalable builds
* Gradle integration process is quite easier
* It avoids compilation
* It has Complexities like Steep learning curve
* It is easy to use and maintain
* It does not have XML file
* Gradle is faster than the maven
* **Maven:-**
* Maven is open-sourcetool that offers a convention over-configuration approach to project build management
* Its main goal is related to the projectphase
* XML file describes the building process of a project
* It has an dependency management
* No need to store the binary libraries with in the source control
* It efficiently manages the heirarichial dependency
* It reduces the duplication within the project
* Provides better co-operation among the source code, plugin,libraries and IDE
* Maven is slower than the gradle
* There is compilation in maven
* If the code for an existing dependency is unavailable therefore you can not add that dependency using maven itself

1. what is difference between yaml and properties file

* **YAML:-**
* YAML configuration file in spring boot provides a very convenient syntax for storing logging configuration in a heirarichail structure
* The purpose of YAML is data serilization that often used for configuration files
* It has more clarity and human-friendliness in comparision to propties file
* It provides several unique and useful features for us
* It Supports key/val, basically map, List and scalar types (int, string etc.)
* By using spring profiles, we can have multiple profile in one single .YML file
* YAML is more readable tha the properties files
* Spring frameworks doesn’t support @propertySources with .yml files
* To access the YAML propertiesb, we will create an object of the config class and can access the properties using that object
* **properties file:-**
* Properties files are used to keep ‘n ’ no.of properties in a single file to run the application ina different environment
* The application.properties files are loated in the src/main/resources directory
* Properties is a files extension for the files mainly used in java-related technologies to store the parameters of an application
* It can also be used for storing strimgs for internationalization and localization
* Each profile needs to crete one separate .propoerties files
* It has non Hierarchial structure
* While in case of .properties files we get stringd regardless of what the actual value type is in the configuration
* It supports @propertySources with .Properties files
* It Supports key/val, but doesn't support values beyond the string
* To read properties files in spring we can use @popertySource Annotation

3.what is profiles in spring boot?

* **spring profile:-**
* A profile is a set of configuration settings. Profiles in simple term can be termed as the different environments which every application comprises
* Spring Boot allows to define profile specific property files in the form of application in the form of .properties
* It automatically loads the properties in an application. properties file for all profiles, and the ones in profile-specific property files only for the specified profile
* Spring Profiles provides a way to segregate parts of your application configuration and make it only available in certain environments.
* Those @Component or @Configuration can be marked with @Profile to limit when it is loaded
* we can maintain profiles with the help of properties and file files in springs
* Spring profiles can also be activated via Maven profiles, by specifying the spring. profiles. active configuration property
* We have @profile annotation in profiles,The @Profile annotation is one of the ways to react to an activated profile in a Spring Boot application
* Profile annotation is a logical grouping that can be activated programmatically. It can be used on type-level annotation on any class or as a method-level annotation on any @Bean method
* **Spring profile used for:-**
* When we have have a large set of spring application configurations that vary across environments then we can use Spring profiles
* Spring Profiles helps to easily set right configurations on right environments. Otherwise, without having Spring Profiles it is a big pain to manage environment specific configurations. For example, your application may have to rely on the configurations externalised on the environments.
* We can use spring.profile.active environment property to active profiles. We can replace this by using command line
* To set active profiles by calling springApplication.setAdditionalProfiles before your application runs.
* It is also possible to activate profiles using Spring’s configurableEnvironment  interface.
* The default profile is always active. Spring Boot loads all properties in application. yml into the default profile

4.what is the difference between internal embedded and external db config

* The in-memory database is an embedded database and these are volatile by default and also all stored data loss when we restart the application

5.what is entity and different types of mappings?

* **Entity:**-
* An entity is a persistence domain object. Typically, an n entity represents a table in a relational database, and each entity instance corresponds to a row in that table.
* Entity classes have a stereotype of entity. An entity class is essentially an object wrapper for a database table. The attributes of an entity are transformed to columns on the database table.
* The @Entity annotation specifies that the class is an entity and is mapped to a database table
* **An entity class must follow these requirements**.
* The class must be annotated with the javax.persistence.Entity annotation.
* The class must have a public or protected, no-argument constructor. The class may have other constructors.
* The class must not be declared final. No methods or persistent instance variables must be declared final.
* If an entity instance is passed by value as a detached object, such as through a session bean’s remote business interface, the class must implement the Serializable interface.
* Entities may extend both entity and non-entity classes, and non-entity classes may extend entity classes.
* Persistent instance variables must be declared private, protected, or package-private and can be accessed directly only by the entity class’s methods. Clients must access the entity’s state through accessor or business methods.
* **Different types of mappings:-**
* **one-to-one mapping:-**
* The One-To-One mapping represents a single-valued association where an instance of one entity is associated with an instance of another entity
* **one-to-many mapping:-**
* A one-to-many relationship between two entities is defined by using the @OneToMany annotation
* **many-to-one mapping:-**
* A many-to-one mapping means that many instances of this entity are mapped to one instance of another entity – many items in one cart.
* many-to-many mapping:-
* We can use @ManyToMany annotation to create a many-to-many relationship between two entities

6.logging in spring boot application?

A.

* Spring Boot uses Apache Commons logging for all internal logging.
* Spring Boot's default configurations provides a support for the use of Java Util Logging, Log4j2, and Logback.
* Using these, we can configure the console logging as well as file logging.
* Spring Boot uses Commons Logging for all internal logging but leaves the underlying log implementation open.
* By using Default configurations in each case, loggers are pre-configured to use console output with optional file output also available.