Q1 - What is a version control system?

A - Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.

Q2 - Why did a version control system develop? What were the necessities?

A - Version control systems allow multiple developers, designers, and team members to work together on the same project.It helps them work smarter and faster! A version control system is critical to ensure everyone has access to the latest code and modifications are tracked.

Three reasons for version control

- Collaboration. Without a VCS in place, you're probably working together in a shared folder on the same set of files.
- Storing Versions (Properly) Saving a version of your project after making changes is an essential habit.
- Restoring Previous Versions.
- Understanding What Happened.
- Backup.

Q3 - Define the different types of version control systems.

A - The two most popular types are centralized and distributed. Centralized version control systems store all the files in a central repository, while distributed version control systems store files across multiple repositories. Other less common types include lock-based and optimistic.

Q4 - List a few differences between the two version control system types.

A -

Centralized Version Control	Distributed Version Control

In CVS, a client need to get local copy of source from server, do the changes and commit those changes to central source on server.	In DVS, each client can have a local branch as well and have a complete history on it. Client need to push the changes to branch which will then be pushed to server repository.
CVS systems are easy to learn and set up.	DVS systems are difficult for beginners. Multiple commands needs to be remembered.
Working on branches in difficult in CVS. Developer often faces merge conflicts.	Working on branches in easier in DVS. Developer faces lesser conflicts.
CVS system do not provide offline access.	DVD systems are workable offline as a client copies the entire repository on their local machine.
CVS is slower as every command need to communicate with server.	DVS is faster as mostly user deals with local copy without hitting server everytime.
If CVS Server is down, developers cannot work.	If DVS server is down, developer can work using their local copies.

Q5 - What is Git?

A - Git is an open-source project that tracks changes in source code during software development. It can handle both small and large projects with speed and efficiency. It is one of the best source code management with features like task management and bug tracking and is widely used by most industry experts nowadays.

Q6 - List a few features of Git.

A - **Features of Git**

- Tracks history.
- Free and open source.

- Supports non-linear development.
- Creates backups.
- Scalable.
- Supports collaboration.
- Branching is easier.
- Distributed development.

Q7 - State any three commands of Git and why we use them.

A - **1. git add**

Usage (i): git add [file(s) name]

This will add the specified file(s) into the Git repository, the staging area, where they are already being tracked by Git and now ready to be committed.

2. git clone

Usage: git clone [URL]

Suppose, we want to work on a file that is on a remote Github repository as another developer. How can we do that? We can work on this file by clicking on Clone or Download and copying the link and pasting it on the terminal with the git clone command. This will import the files of a project from the remote repository to our local system.

3. git checkout

Usage (i): git checkout [name-of-the-new-branch]

We use this command to navigate to an existing branch, add new files, and commit the files.

Q8 - Is Git the same as Github? Why or Why not?

A - While Git is a tool that's used to manage multiple versions of source code edits that are then transferred to files in a Git repository, GitHub serves as a location for uploading copies of a Git repository. In a sense, then, there's no comparison when it comes to Git vs. GitHub as far as their function.

Q9 - What is the command to get the installed version of Git?

A - You can check your current version of Git by running the **git --version** command in a terminal (Linux, macOS) or command prompt (Windows).

Q10 - What is the command to add all files and changes of the current folder to the staging environment of the Git repository?

A - To add and commit files to a Git repository
Enter **git add --all** at the command line prompt in your local project directory to add the files or changes to the repository.

Q11 - What is the difference between git status and git log commands?

A - The git log command displays committed snapshots. It lets you list the project history, filter it, and search for specific changes. While git status lets you inspect the working directory and the staging area, git log only operates on the committed history.

Q12 - What is the command to initialize Git on the current repository?

A - Initializing a new repository: **git init**

Q13 - What are the different states of a file in Git? Explain them along with the associated commands.

A - The Three States

- Modified means that you have changed the file but have not committed it to your database yet.
- Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
- Committed means that the data is safely stored in your local database.

This leads us to the three main sections of a Git project: the working tree, the staging area, and the Git directory.

Q14 - Git automatically adds new files to the repository and starts tracking them. True or False? Give reasons

A - False

The **git add** command takes a path name for either a file or a directory; if it's a directory, the command adds all the files in that directory recursively.

Q15 - What is the command to commit the staged changes for the Git repository?

A - The **git commit** command

Q16 - What is the command to commit with the message "New email"?

A - You can use the **git config** command to change the email address you associate with your Git commits. The new email address you set will be visible in any future commits you push to GitHub.com from the command line.

Q18 - What is a branch in Git?

A - A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process. You can think of them as a way to request a brand new working directory, staging area, and project history.

Q19 - What is the command to create a new branch named "new-email"?

A - git branch new-email

Q20 - What is the command to move to the branch named "new-email"?

A - git checkout new-email

Q21 - What is the option, when moving to a branch, to create the branch it if it does not exist?

A - Checkout a New Branch or Reset a Branch to a Start Point
If the BRANCH-NAME branch doesn't exist, Git will create it and start it at START-POINT . If the BRANCH-NAME branch already exists, then Git resets the branch to START-POINT . This is equivalent to running git branch with -f .

Q22 - What does the git init command does?

A - The git init command creates a new Git repository. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository.

Q23 - What is a fork? How is it different from clone in Git? How do you fork and clone a repository?

A - A fork is a copy of a repository that you manage. Forks let you make changes to a project without affecting the original repository. You can fetch updates from or submit changes to the original repository with pull requests.
Any public Git repository can be forked or cloned. A fork creates a completely independent copy of Git repository. In contrast to a fork, a Git clone creates a linked copy that will continue to synchronize with the target repository.

Q24 - What does ‘push’ mean in Git? Give the command.

A - The git push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo.