1. **Version control adds additional steps like creating a repo, commit, etc., and it slows me down from completing my work. Is that overhead worthwhile?**
2. **Version control** is nothing but having control on versions. Like example we can take Google doc not only in **Google doc** everything has an **undo option** like to **restore the original content**. I Google doc we can look at all our changes and if we wish to replace the original content with old one we can do it as well. When we commit we will provide a **message** to an added file so by using **gitlog** command we can get to know what **changes** have been made. Version control is very much helpful in speeding up our work when we struck at some point and it brings some clarity on what work we are doing.
3. **What is a good logical way to structure the commits when creating web pages?**
4. After adding the file we have to commit by using **git commit –m “providing with a message”.** Using this **git commit** message we will have a track of changes that we have done. GIT commit will comes under the logical procedure of pushing the files into our GIT repo.
5. **Is it good to include assets like images and videos in the git repo or is it better to keep them outside the repo? What if there are videos on the web page? How do we ignore them from staging and committing to the repo?**
6. **GIT repo** is basically a cloned part of our local repo. There should be same files in **local and cloud storage** GIT repo. It doesn’t throws error when we store files outside but only if we store the files in local repo and not in GIT repo then only it will throw error. By using **.gitignore** we can ignore all the file patterns.
7. **It looks like staging is not required before making a commits to the repository. Do you agree?**
8. As **staging** allows you to continue making edits to the files/working directory, and make commits in parts when you think things are ready, you can use separate stages for logically unrelated edits. Staging is the intermediate process before pushing it to the repo. When we make some changes it will goes to staging part it will remains at the staging part until we push those files. So it is very important which will keep any eye changes that we make.
9. **Imagine a scenario where the old project files, that are previously in the version history, are deleted from the project folder and commits are done to the same repo. Will we lose the old files forever? How does this work?**
10. GIT repo is basically a clone part of the local repo so if files got deleted from repo we can restore those files using local repo. Here in local repo no files get affected If we deleted on other side.
11. **I’m working on a data science project with the Google collab. Collab already maintains version control. How is this different from using git? Which one is better? Is it possible to maintain collab files on git?**
12. GIT is just cloud storage where we store any type of files and also we can store **COLLAB** files too.

**COLLAB** is not much complex as GIT anyone can understand easily. It is also very flexible to use with better options.