**1.Basic Git Commands:**

* git init : Initializes a new Git repository.
* git clone <repository URL>: Clones a repository into a new directory.
* git status: Shows the status of changes (staged, unstaged, untracked).
* git add <file>: Stages a file for commit.
* git add .: Stages all changed files.
* git commit -m "message”: Commits the staged files with a message.
* git push: Pushes committed changes to a remote repository.
* git pull: Fetches and merges changes from the remote repository.
* git fetch: Fetches changes from a remote repository but does not merge them.
* git merge <branch>: Merges a branch into the current branch.
* git rm --cached filename: staging area to working directory
* git restore filename: to restore deleted file
* git restore --staged filename: staging area to working directory

**2. Branching and Merging:**

* git branch: Lists all branches in the repository.
* git branch <branch\_name>: Creates a new branch.
* git checkout <branch\_name>: Switches to the specified branch.
* git checkout -b <branch\_name>: Creates and switches to a new branch.
* git branch -d <branch\_name>: Deletes a branch.
* git merge <branch\_name>: Merges the specified branch into the current branch.

**3. Remote Repositories:**

* git remote -v: Displays the remote repository URL.
* git remote add <name> <url>: Adds a remote repository.

**4. Undoing Changes:**

* git reset <file>: Unstages a file without removing its changes.
* git reset --soft <commit> leaves the working directory and the staging area unchanged. The changes from the commits that were "undone" are still staged
* git reset --mixed <commit>It moves the HEAD to the specified commit and unstages all changes from the commits that were undone
* git reset --hard: Resets the current branch to a previous commit, discarding all changes.
* git revert <commit: Creates a new commit that undoes changes made in a previous commit.
* git checkout -- <file>: Discards changes in a working directory for a specific file.
* git stash: Temporarily saves changes for later.
* git stash apply [stash@{index}] Applies stashed changes back into the staging directory.
* git stash drop [stash@{index}] to remove the stash history of particular commit from stash list.
* git stash pop: Applies stashed changes back into the working directory.
* git stash list: it will show the history of stash ids

**5. Viewing History:**

* + git log: Shows commit history.
  + git log --online: Shows a simplified view of the commit history.
  + git diff --staged: the changes that are in the staging area but have not yet been committed.
  + git diff commit1 commit2: differences between the two commits commit-1 and commit-2
  + git diff main feature-branch: This displays the differences between the main branch and the feature-branch.
  + git diff <branch\_name>: Shows differences between the current branch and another branch.
  + `git show <commit>: Shows the changes made in a specific commit.

**6. Collaborating:**

* + git cherry-pick <commit>: Applies changes from a specific commit onto the current branch.

**7. Advanced Commands:**

* + `git rebase <branch name>: Re-applies commits on top of another branch.
  + Git ignore:
  1. touch .gitignore
  2. touch file1
  3. cat .gitignore
  4. mkdir folder
  5. vim .gitignore
  6. git status
  7. git add .