Task 1: Initialize a Git repository and configure user details.

Step1- Create a directory practice_git (folder name).

Step2 – In that folder run command **git inti.**

Step3 – For User details configuration, Run these command

git config –global user.name "enter your name"

git config –global user. email "enter your mail_id"

Task 2: Create a GitHub repository (public), add a `README.md`, and push it to GitHub.

Step1 - Create a file README.md.

Step2 – Staged that file and commit it.

Step3 – Add the remote repository and push it to GitHub repo using these commands.

git branch -M main

git remote add origin https://github.com/username/example.git

git push -u origin main

Task 3: Clone an existing repository, make changes to a file, commit, and push it back.

Step1 – Clone the repo

git clone https://github.com/username/repository-name.git

cd repository-name

Step2 – Make a file and Write something in that file.

Step3 – Add, Commit and push it to the origin git repo

git add name.txt

git commit -m "create name.txt file"

git push origin main

Task 4: Create a. gitignore file to exclude unnecessary files from tracking.

Step1 – Create file. gitignore in the working directory.

Task 5: Create a new branch, modify files, commit, and merge it into `main` using a pull request.

Following command are used –

git checkout -b feat-b

nano example.txt

git add example.txt

git commit -m "Add new feature to example.txt"

Task 6: Clone a repository, create a new branch, make a small change, commit, and push.

Following commands are used -

git clone https://github.com/username/repo-name.git

cd repo_name

git checkout -b new-branch

nano example.txt

git add example.txt

git commit -m "Make small change to example.txt"

git push origin new_branch

Task 7: Fork a repository, make changes, and submit a pull request.

Task 8: Edit the same part of a file in different branches, merge them, and resolve conflicts.

Following commands are used –

git checkout -b b-1
nano example.txt

git add example.txt

git commit -m "Change in b-1

git checkout -b b-2

nano example.txt

git add example.txt

git commit -m "Change in branch 2"

git checkout b-1

git merge b-2

git add example.txt

git commit -m "Resolved merge conflict between branch-1 and branch-2"

Task 9: Use `git stash` to save work temporarily and reapply changes later.

Following commands are used -

Stash changes –

git stash

git stash pop

Task 10: Squash multiple commits into one using `git rebase -i`.

Step1 - git rebase -i HEAD~3

Step2 - Choose squash for commits you want to combine, then save and close the editor.

Task 11: Rebase a feature branch onto `main` and compare it with `git merge`.

Task 12: Cherry-pick a commit from one branch to another.

Steps – Get the commit hash from the other branch

Git log branch-name

Git checkout target-branch

Git cherry-pick (commit-hash)

Task 13: Revert a file to the second last commit.

Following commands are used -

git log

git checkout HEAD~2 –example.txt

git commit -m “Revert example.txt file”

Task 14: Configure Git aliases for frequently used commands.

Commands to configure Git aliases**.**

Git config –global alias. lo log –oneline

Git config --global alias.s status

Task 15: Automate a Git pre-commit hook to prevent bad commits.

Task 16: Work with multiple remotes (`origin` & `upstream`) and sync changes.

Following commands are used -

Add the upstream remote (the original repo you forked from)

git remote add upstream https://github.com/original-owner/repository-name.git

git fetch upstream

git checkout main

git merge upstream/main

git push origin main