Git Assessment-2

1. How do you rename the current branch you are on to a new-branch-name?  
   git branch -m <new branch name>
2. What is the command to stash your changes and include untracked files?  
   git stash -u
3. How would you merge changes from the branch feature-new into your current branch?  
   git merge feature-new
4. What command do you use to view the differences between your working directory and the last commit?  
   git diff HEAD
5. Explain how to resolve a merge conflict that occurs during a git merge. What steps would you take?  
   Identify- Use git status to find conflicted files.

Edit- Open the file and fix the conflict manually.

Stage- Use git add to mark the file as resolved.

Commit- Create a new commit with git commit -m "Merge conflict resolved".

1. Explain the purpose and use case of git rebase with an example. How does it differ from git merge?
2. What does the command git cherry-pick <commit hash> do? Provide a scenario where you might use it.  
   git cherry-pick is useful for selectively applying changes from one branch to another without merging the full branch
3. Describe the differences between git reset --soft, git reset --mixed, and git reset --hard. When would you use each?  
   git reset –soft- Unstages changes, keeps them in the working directory.

git reset –mixed- Unstages changes, discards them from the working directory.

git reset –hard- Unstages changes, discards them, and resets the HEAD.

1. Describe how you would revert a commit that has already been pushed to a shared repository. What command would you use, and what should you consider before doing this?

To revert a pushed commit, use git revert <commit hash>. This creates a new commit that undoes the changes. Resolve any conflicts and push the revert commit.

1. What is the difference between git pull --rebase and git pull? When would you prefer one over the other?  
   "git pull" allows for detailed merge point tracking, while "git pull --rebase" offers a linear commit history without merge commits, ideal for feature branches requiring frequent sync with the main branch.
2. How can you view the changes made by a specific commit? What command do you use?  
   git show <commit hash>  
   The git show command in Git allows you to view changes made by a specific commit, providing detailed information like author, date, message, and changes introduced.
3. What is branch protection in Git, and why is it important for a collaborative development environment?
4. Explain how to set up branch protection rules in GitHub.
5. What happens if a user tries to push changes directly to a protected branch?
6. What is the purpose of a .gitignore file, and how do you create one? Provide an example of what you might include in it.