→ cheaking difference b/w stagged file & PWd and last commit & stagged file :-

Let's you are working on a project and you have modified some files and you check status then it shows you have

modified those files.

Now if you write cmd 'git diff'

It will show difference b/w stagged file & persent working file.

If you have added file to stagged area & then modified it and then check difference you will see difference b/w them as that modified file & lines you have modified.

If you want to check difference b/w last commit & stagged file, you have to write

git diff --staged

it will show edited files with edited lines

→ Skipping staging area :-

In a workflow to commit a file we have modified we have to first add them to staging area & then make have to make commit on it.

But if you want to commit directly by skipping staging area for a modified file

you have to write command

git commit -a -m "massage"

but also note that you can skip staging area for modified files / folder only if you

have a new new untracked file then to commit it you have to add it to staging area then commit it, you can't directly commit an untracked file.

→ Removing any file/folder from git directory:-

If you want to remove file/folder from git directory you can delete it manually from git directory and you are good to go, but there is special command for it

git rm file name

We generally use this command as if you manually delete a file then you have to add it & then commit it

but if you do it by command it bydefault add it to stagging area, so you don't need to bother it.

→ Renaming any file in git directory :-

If you want to rename file in git directory you can manually rename it then add it to stagging area then make commit.

but you can do it by command too.

git mv filename new file name

Using command make it easier as it by default add new file in stagging area.

Sometimes if you want to ignore a file and add that file to .gitignore, git then also track that file if you want to untrack that file just remove that file using rm command.

git rm --cached file name

This command also used to unstage a file.

→ Viewing & changing commit by log command :-

If you are working on a project with lots of contributes you needed log command/ log related commands to view commits and their details.

- git log

This command tells up commits made on this project with ~~name~~ identity of editor & time. Also message with commit.

- git log -p

This command shows what coad has been removed & added in ~~this~~ that particular commit.

So, basically it used to see differences.

- git log -p -n

In this coad n is an integers like 2, 3, 10 etc.

if you write git log -p -3 it only shows 3 commits with differences.

- git log --stat

This commands give short view on what happened in this commit.

i.e.

which file has been edited with how many lines has been edited.

- git log --pretty=oneline

This command shows each commits in one line. so that it is easier to move through each commit search for any specific commit.

- git log --pretty=short

This command shows each commits in short form.

Containing author name & message.

- git log --pretty=full

This show each commit in full form.

Containg author & commitor name with massage.

- git log --since=2.days

This command filter commits in blw two days.

- git log --since=2.months

This command filter commits in 2 months.

If you want to view log details in coustum format you have to use

```
git log --pretty=format:" format codes "
```

To know which format codes you want visit

Grit SCM websit and you get different types of format codes like

```
git log --pretty=format:"%h -- %an"
```

this will show commits in form such that first it show hash value then -- then authour name for all commits.

you can use different format codes which you want

%h → abbrivated hash of commit

%an → authour name
%ae → authour email

etc.

There is a another command

git commit --amend

which used to amend/append in older commit.

lets you have edited some files in your code and you don't want to commit it again but to merge it in last commit with new

It can also be used to change commit massages of previous commit. (amend)

massage you first add that new edit in staging area then write

```
git commit --amend
```

this will open vim editor, to start writting in vim editor

type 'i' you can insert/write in vim editor

to exit from vim editor you have to type

:wq in terminal of vim.

To go back insert mode to terminal from insert press escape button.