1. **Pull Request and Merge event:**
   - ## PULL Request
     - a pull request is assentialy a devloper "Leomeet" working on a **branch "B"** of a Repo **Solid** with a **"master"** branch
     - and wants to merge the changes of **branch "B"** to branch **master** thus raising a pull request stating to add the changes of **branch B** to branch **master**
     - this pull request will then be reviewd & the code in both **branches** latest commits will be compared if there are any changes more commits will be added and then
     - a ***Merge*** event will trigger adding the content of **branch B** to the **master** branch and deleting the **branch B** afterwords
     - *you can raise a pull request from **github** → **pull reqests** → **new pull reqest**.*
   - ## Merge Event
     - a merge event is 2 conditions with remote with local
     - with remote
       - a remote merge event will be triggerd when the changes code have been reviewed and none conflict are present
         we will use github gui to accept and merge a **branch b** to **master branch**
     - with local
       - when you are working with local you need to have 2 different branches
       - lets say a **feature** and a **master** branch
       - got to master branch
         `git checkout master`
       - aftr that `git merge feature`
       - this will add the changes in feature branch to master branch
       - after that delete the remaining branch that is **feature** with
         `git branch -d feature`

---

2. **Rebase**
   - ## Git Rebase
     - git rebase is a way to put the commits from a *feature branch* *on top of the master branch
     - command
       `git checkout feature` now i want to put changes of feature branch on top of master branch

> `git rebase master` will work as command to rebase the changes on *top of master*

- after rebasing feature branch to master branch the **master head will point at the latest commit of master branch
  - *to add it to feature branch you need to* `checkout master` and then `git rebase feature` *after this you can delete the feature branch by* `git branch -d feature`

---

3. **Change commit messages**
   - ## Git commit message
     - to change the latest commit's massege we can use `git commit --amend` command
     - we can also use `git rebase -i [hash]` to open an interactive rebase enviroment and then use `e` to edit the commits or maybe squash and pick them in one

---

4. **Cherry Pick**
   - ## Git cherry-pick
     - cherry-pick allowes you to add a perticuler commit from other at the top of a branch
     - > code `git checkout main` → `git cherry-pick [hash_of_commit]` → this will add commit to top of the main branch

---

5. **Drop commit**
   - ## Drop commit
     - > `git reset --soft [hash_of_commit_you_want_to_drop]` will delete the commit from normal log but you can still access the hash from **reflog**
     - > `git reset --hard [hash_of_commit_need_to_be_droped]` will delete the commit and its hash from inside out and point to previous commit