1 What is prune in fetch ?

Ans :

In the context of databases and data manipulation, "pruning" typically refers to the process of removing unnecessary or redundant data in order to optimize performance or reduce storage space.

However, it seems like you're asking about "prune" in the context of "fetch". In this case, it might be referring to a specific operation or command in a particular programming language or framework that involves fetching data and then pruning it.

Without more context, it's difficult to provide a precise answer. If you can provide more information about the specific programming language or framework you're working with, or any additional context, I'd be happy to try to give a more detailed explanation.

2 Learn what is squash and interactive interface of squash

Ans :

Normal Workflow:

* You make a series of small, incremental commits as you work on a feature or fix a bug.
* Each commit represents a logical step in the development process.

Squashing:

* When you're ready to merge your changes into the main branch, you can use a command like git rebase with the interactive mode to combine multiple commits into one.
* During the interactive rebase, you can choose which commits to squash together.

The interactive interface of squash allows you to:

Choose Commits: You can select which commits you want to squash. This is done by specifying which commits to include in the squashing process.

Edit Commit Messages: You can also edit the commit messages during this process. This can be useful if you want to provide a more descriptive or concise message for the squashed commit.

Reorder Commits: While in the interactive rebase interface, you can reorder commits, which means you can change the order in which the commits are applied.

Delete Commits: If you decide that a commit is not necessary, you can choose to omit it from the final squashed commit.

Combine Commits: The main goal is to combine multiple commits into a single commit that encapsulates all the changes made in those individual commits.

3 What is merge conflicts and how to resolve it ?

Ans :

Merge conflicts occur in version control systems like Git when there are conflicting changes in the same part of a file or files between different branches. This happens when Git is unable to automatically merge the changes because it's unclear which version should be used.

Here are the steps to resolve a merge conflict:

Identify the Conflict: When you attempt to merge branches and Git encounters a conflict, it will notify you. You'll see a message indicating that there's a conflict and which files are affected.

Open the Affected Files: Open the files with conflicts in a text editor. Git will mark the conflicting sections with special markers, usually <<<<<<<, =======, and >>>>>>>.

Review and Edit: Examine the conflicting sections and decide which changes should be kept. You may want to consult with your team members if you're unsure about which version to choose.

Remove Conflict Markers: Delete the conflict markers (<<<<<<<, =======, >>>>>>>) once you've resolved the conflict.

Save the Changes: After you've edited the file to your satisfaction, save it.

Add the Files: Use git add <file> to stage the resolved files. If there are multiple conflicted files, add each one individually.

Commit the Changes: Use git commit to create a new commit that finalizes the merge. Git will automatically recognize that you're resolving a merge conflict.

Complete the Merge: If there are no other conflicts, you can finish the merge by using git merge --continue.

Push the Changes: If you're merging branches on a remote repository, you'll need to push the changes to the remote using git push.