Assignment 2

**What is prune in fetch ?**

In the context of fetching data from a database or a data source, "prune" typically refers to the process of removing or filtering out certain elements or data points from the fetched data based on specific criteria. Pruning is often done to reduce the size of the data or to remove irrelevant or sensitive information

Here's how pruning may be used in the context of data fetching:Removing Unnecessary Fields: You might prune (remove) certain fields or columns from the fetched data if they are not needed for the current task or if they contain sensitive information that should not be exposed.Filtering Rows: Pruning can involve filtering rows or records based on certain conditions. For example, you might fetch a large dataset from a database and then prune it to only include records that meet specific criteria, such as a certain date range or a particular category.Optimizing Data: In some cases, you might prune data to optimize it for a

**2 Learn what is squash and interactive interface of squash**

In the context of version control systems like Git, "squash" refers to the process of combining multiple commits into a single, more concise commit. This is typically done to streamline the commit history, making it easier to read and understand. Squashing commits is often used before merging a feature branch into the main branch to present a cleaner and more logical commit history.

The primary benefit of squashing commits is that it simplifies the commit history, making it easier for reviewers and future maintainers to understand the changes introduced by the feature branch. It helps avoid cluttering the history with numerous small, incremental commits that may not provide meaningful context on their own.However, be cautious when squashing commits, especially if you're working on a collaborative project with other developers, as it permanently combines commits and may make it harder to track the individual changes made during development. Always follow your team's guidelines and best practices for commit

**Interactive Interface of Squash**

The interactive interface for squashing commits in Git is typically accessed through the git rebase command with the -i or --interactive flag. This interface allows you to interactively choose which commits to squash, edit commit messages, reorder commits, and perform other operations on your commit history. Here's how to use the interactive interface to squash commits.

This interactive interface provides you with control over how you want to combine and organize your commit history. It's a powerful tool for maintaining a clean and meaningful Git history while working on your codebase

**What is merge conflicts and how to resolve it ?**

In the context of version control systems like Git, a "merge conflict" occurs when two or more branches or individuals make changes to the same part of a file or codebase, and the system cannot automatically determine which changes should take precedence. When a merge conflict arises, Git needs human intervention to resolve the conflict and determine how to combine the conflicting changes. Here's how to resolve a merge conflict.

Keep in mind that merge conflicts are a normal part of collaborative development when multiple people are working on the same codebase. Good communication and coordination with your team can help reduce the frequency of conflicts, but knowing how to resolve them is an essential skill for any developer using version control systems like Git.



