1. What is Degenerate, speciality and normal merge?

**Degenerate Merge:**

There are 2 degenerate scenarios which are called already up-to-date and fast forward. some might consider them not to be true merge strategies because they don’t introduces a new merge commit after performing the git merge. Already up-to-date is when all the commits from the other branch are already present in your target branch, even if it has advanced on its own, the target branch is said to be already up to date. As a result, no new commits are added to your branch. Fast Forward is a fast forward merge happens when your branch Head is already fully present and represented in the other branch. This is the inverse of the already up-to-date case.

**Normal Merge:**

There are three merge strategies all produce a final commit; Resolve, Recursive, and Octopus. The resolve strategy operates on only two branches, locating the common ancestor as the merge basis and performing a direct three-way merge by applying the changes from the merge base to the tip of the other branch HEAD on to the current branch. The recursive strategy is similar to the resolve strategy In that it can only join two branches at once. However, it is designed to handle the scenario where there is more than one merge base between the two branches. In these cases, git forms a temporary merge of all common merge bases and then uses that as the base from which to deserve resulting merge of the two given branches via a normal three way merge algorithm. The octopus strategy is specifically designed to merge together more than two branches simultaneously.

**Specialty Merges:**

There are two merge strategies that you should be aware of because they can sometimes help you to solve strange problems. The two special strategies are Ours and subtree. The ours strategy merges in any number of other branches, but it actually discards changes from those branches and uses only the files from the current branch. The result of an ours merge is identical to the current HEAD, but any other named branches are also recorded as commit parents. The subtree strategy merges in another branch, but everything in that branch is merged into a particular subtree of the current tree. You don’t specify which subtree; git determines that automatically.