**Question 5**

Step 1: Step 1: Create a feature branch.

**Ans:** git branch “new\_branch”

Step 2: Switch to the new branch.

open the file and make some changes to it.

Add and commit the changes to the new branch.

open the same file and make some changes to it.

Add and commit the changes to the new branch.

open the same file and make some changes to it.

Add and commit the changes to the new branch.

Step 3: Use the "git log" command to view the commit history and identify the commit to which you want to reset.

A screen shot of a computer

Description automatically generated

Step 4: Use the "git reset" command followed by the desired reset type and the commit hash

A screenshot of a computer program

Description automatically generated

Step 5: Verify that the reset was successful by using the "git log" command again.

**Ans:** git log –oneline (gives the changed commits)

Step 6: Use the "git log" command to view the commit history and identify the commit that you want to reverse.

**Ans:** git log –oneline (gives the changed commits)

Step 7: Use the "git revert" command followed by the commit hash or reference to which you want to revert. (Hint: git revert <commit hash>)

A screenshot of a computer

Description automatically generated

Step 8: Verify that the revert was successful by using the "git log" command again. A screen shot of a computer

Description automatically generated

Note: Identify the difference between git log after git reset and git revert.

**Ans:** After using git reset, the commits that were removed from the branch's history will not appear in git log unless you explicitly reference them using a commit hash or a different branch. After using git revert, the reverted commits remain in the history, and you can see them in git log along with the new commit(s) created by the revert operation. These new revert commits will appear in the history alongside the original commits.