I decided to choose Git Flow merging strategy because of the next factors:

* Number of developers: since there are 5 developers on the project, they can work in parallel on at least 5 features simultaneously. Creating an additional branch ‘develop’ reduce the chances of merge conflicts in the ‘release’ and ‘master’ branches.
* Release frequency: because of the different release frequencies on PROD and non-PROD environments it is safer to create additional branch ‘release’. This branch can be used for non-PROD environments, while ‘master’ branch will be used for PROD. This will reduce the risk of a code error because the code will be monitored in the ‘release’ branch at least twice before merging into the ‘master’ branch (Note: such approach works if development is performed in one Git repository. If development is performed in multiple repositories, then ‘release’ branch can be removed).

According to this, next branches were created for the repository Data\_Quality\_Engineering\_Intermediate\_9:

* ‘main’ – master branch. Nothing is pushed to this branch from next branches
* ‘5\_CI\_CD\_release’ – branch, for which Jenkins CD pipeline is created
* ‘5\_CI\_CD\_developer’ – branch, from which Jenkins CD pipeline takes changes and deploys it to the ‘release’ branch
* ‘5\_CI\_CD\_feature\_<number\_of\_feature>’ – multiple branches with changes for each feature

‘5\_CI\_CD\_release’, ‘5\_CI\_CD\_developer’, and ‘5\_CI\_CD\_feature\_#’ branches contain one file with changes in each branch. Jenkins CD pipeline takes changes from the ‘developer’ pipeline and deploys in the ‘release’. After deployment, activates Jenkins CI pipeline that performs PyTest tests on SQL Server.