Q1. Give 3 reasons this team will benefit from using Git instead of google drive for managing the codebase.

Github is an open source platform which is helpful for both individual and team. This concept helped developers a lot in terms of saving their code and some tools make it easier for developers to work with complex projects which includes a team of developers.

Ease of use is the biggest strength of Github. This site is structured about git, which is popular and used by developers around the world. The UI of git is minimal hence allows developers to focus on what is really important.

Here are some points follows which tell us about benefits of using git instead of google drive:

1. First of all, the UI of git is very simple and doesn't contain any useless information.
2. For every repository we have an issue page where bug or errors can be reported. Additionally, we can also add filters like closed issues or open issues etc.
3. It has a very huge community. Git has the largest code host on the planet with aroud 22 million repositories.
4. Github has very useful tools which makes it easy for a developer to execute some complex tasks very easily. Team members can be given different powers on different projects.
5. Tools of git also allow you to branch in code so that no conflict arises when working in a team. These tools give privileges to the team that they can work on different modules at the same time.
6. Many popular used cloud hosting services are easily integrated with Github. Any project hosted on github can be easily migrated these platforms for e.g, AWS, Heroku, Google cloud etc
7. Github has a tool called “Projects”. It helps teams to organize the work they are doing by creating roadmap

Q2. Which 3 features of Git should this team learn to be successful?

When we talk about a team, the most important feature to remember is branching which allows parallelism on different modules of the project.
Basically there are three kinds of branches when working as a team in git:

1. master branch
2. staging branch
3. feature branch

STATUS OF CODE

```
git status
git log
```

Whereas the former command shows your changed files in staged and unstaged mode – important

BRANCH COMMAND
There are two essential commands to 1) create a new branch or to 2) check out an available branch:

```
git checkout -b <branch_name>
git checkout <branch_name>
```

Subsequently, if we want to delete some branch we can do using this command:

```
git branch -d <branch_name>
```

Now, if we have many branches ready, and we want all of them at one place, then this command is very useful to bring changes in all branches to one place.

```
git merge
```

KEEP BRANCH UPDATED
When you are working in a team, sometimes you have to work on a module which was previously modified by another programmer. Now, if you have a previous copy with you and don't have a new updated copy then we can use this command to update code before doing some work.

```
git pull
```

ADDING FILES TO COMMIT

Git has a very useful feature called “add”, which allows the user to add some selective or all files to be committed. These are the following commands.

When you want to include all changes/files to be added from the repo directory use this command.

```
git add --all
```

But if you want to include only one file and its changes then this command will be useful for you

```
git add <filename>
```

COMMITING YOUR LATEST CODE

After making changes to your code, we can use

```
git commit
```

This will open a nano editor to write a commit message, but you can also write a commit message in the command line itself using this command.

```
git commit -m "commit message"
```

Above in this document many features and commands are discussed which are at least known by professionals when working in a team.

Q3. What are 3 common mistakes people make when using Git in a team?

Here no one is perfect, there is always a 1% error chance. So now we will discuss what are the mistakes a developer can do while using git. Here are some common mistakes followed:

1. Spelling Mistake: very basic and common mistake anyone can make while committing the changes. And someone can write wrong in commit message, then he/she should now this command to change commit message.

```
git commit --amend
```

2. Forgot to add some files to last commit. It is so common when you don't include some important files to last commit. Then this command is useful.

```
git add missed-file.txt
git commit --amend
```

3. Spelling mistake in branch name. We rename this branch in a similar way to how we rename a file with the “mv” command: by moving it to a new location with the correct name.

```
git branch -m future-brunch feature-branch
```

4. Wrong data committed: Try to check the repository thoroughly that you don't include any personal or irrelevant data, if you commit it by mistake then this command is useful to reset the commit.

```
git reset --mixed <commit-id>
```

5. Unnecessary commits and push. This is the most common mistake which developers make while working on code. They commit very small changes and sometimes go to the previous stage by committing which should not be there because it creates a lot of confusion.