Day 1

1. Agile
2. Git

Git setup

* You must install git in your machine
* You must have a git-hub account with your personal mail-id
* Verify that you are able to login to git-hub account from your personal mail-id

What is Git

Distributed Version Control System

Git terminologies

1. Repository - It is kind of folder that keeps track of teams work, it can be created either locally or in the remote location, it will have information about the project version and remote location

Git basic commands

init, add, commit, push

Right solution when you are working in a team

1. you must create a feature branch that is having all the versions of master/main branch
2. work in the local feature branch and push that to the remote repository
3. In remote repository review & integrate the new feature branch with the master branch, in case there’s a conflict don’t integrate inform the user that the feature branch is not up to date with the remote master

What needs to be done if there’s a merge conflict

1. Pull the changes from remote to local master
2. Create a new branch from the local master & then do the changes [or] switch to the existing branch and merge it with the master and then do the changes

Activity:

1. Update a.txt in the user02 feature branch (issue50)
2. Push the feature branch to the remote
3. In remote repository merge the branch if there’s no conflict & delete the feature branch.

Activity

1. Delete the global config that are
   1. user.name
   2. user.email
   3. credentials.helper
2. create a local repository & add that to the remote
3. Try out with 2 users with feature branch and understand what needs to be done if there’s a merge conflict.
4. Note down all the commands in your notes

Summary of GIT commands

* Git configuration - to setup user.name and user.email

git config --global user.name “your-user-name in git”

git config --global user.email “your-email-id”

* Setting up the credentials so that it doesn’t ask you to enter password each time you push the changes to the remote repository

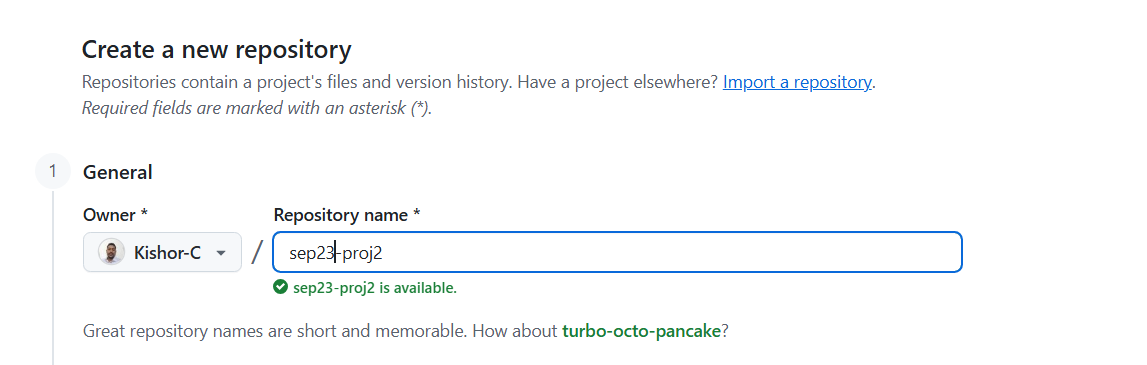
git config --global credential.helper token-id

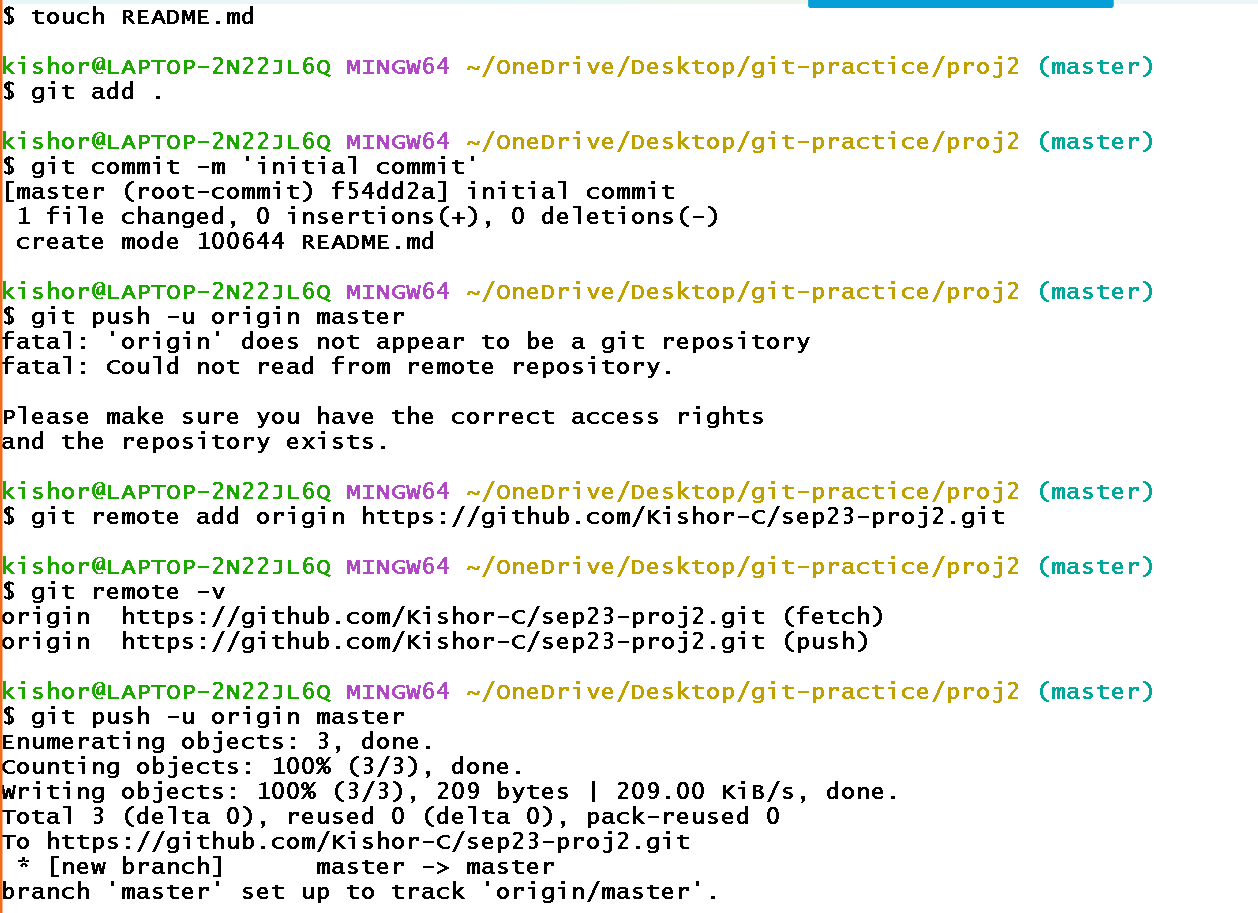
Note: You can go to developer setting to get the token-id (classic)

* init, add, commit, push, status, log, pull, clone, checkout, branch, restore commands

Day 2

Create a new repository in the git-hub with the name sep23-proj2





Commands entered

git init  
touch README.md  
git add .  
git commit -m 'initial commit'  
git remote add origin <https://github.com/Kishor-C/sep23-proj2.git>  
git push -u origin master

How to format the README.md file so that it will be easier for users to read the content

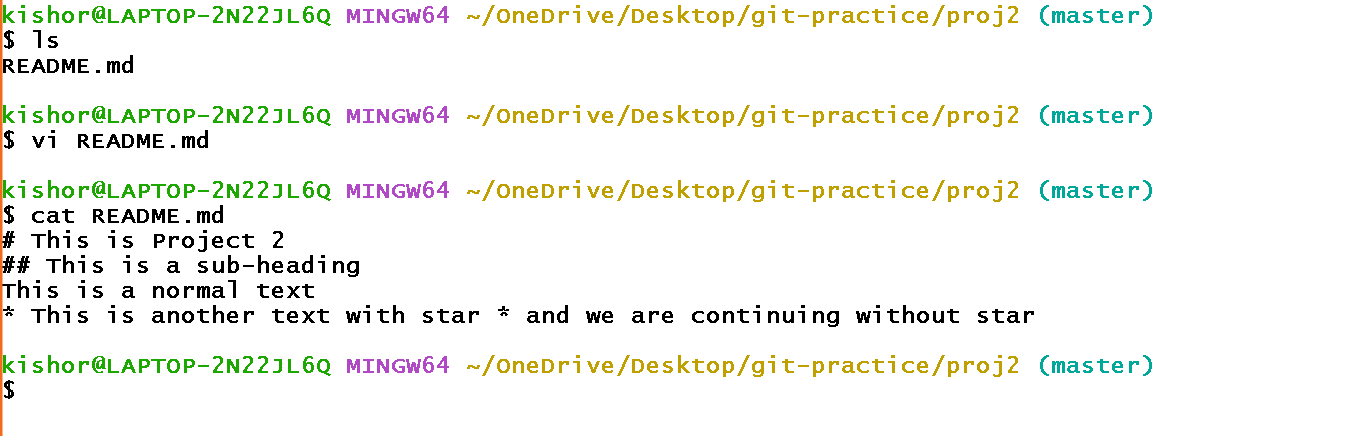
# this is for main heading like heading 1

## this is for heading 2

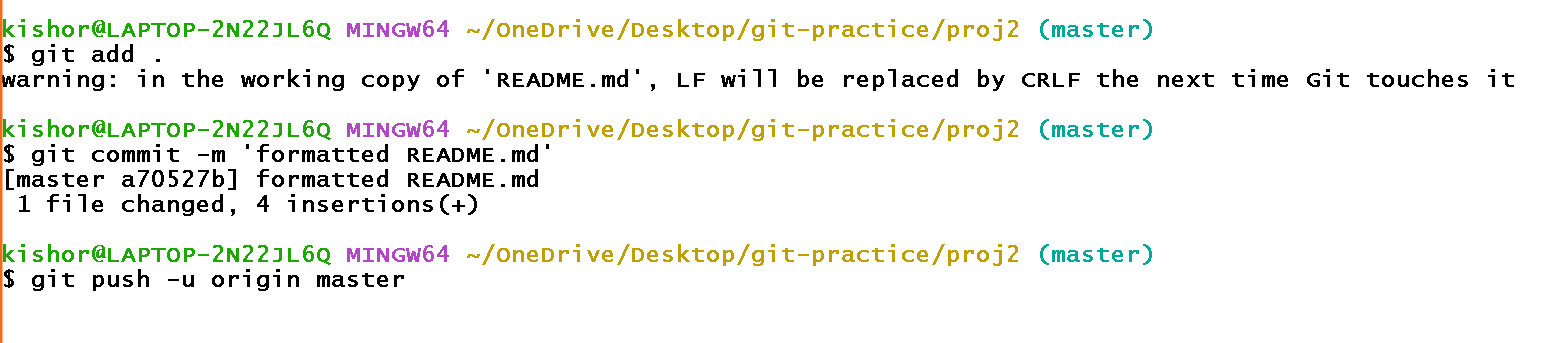
You can use maximum 6 different headings, for sixth heading you must use # 6 times.

‘\*’ is used for bullet point if you give space after the \*.

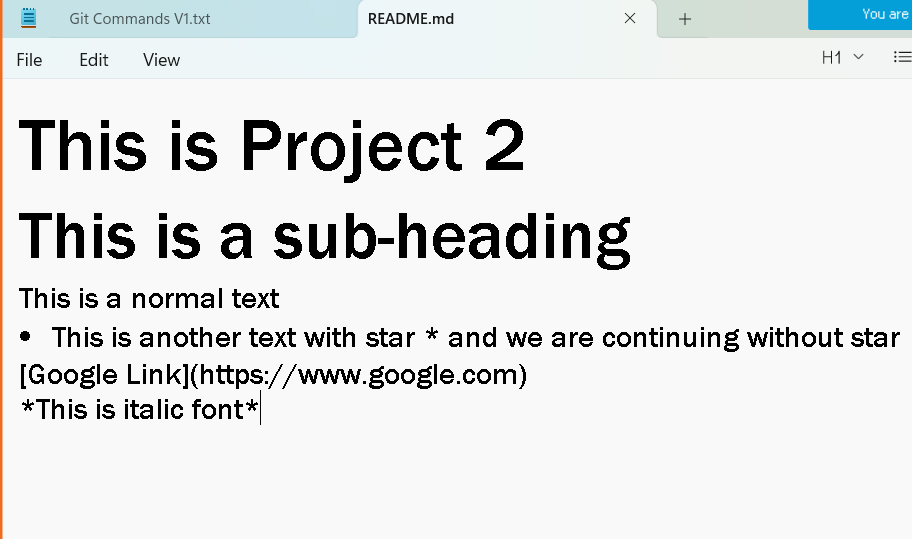
Following are the commands and the content of README.md file



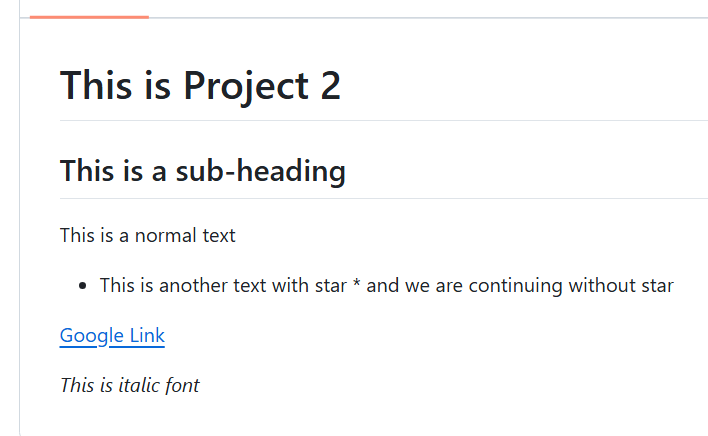
Add & Push the changes to the remote



Formatting the README.md file with some cheatsheet markup



Output:



Simple activity: With the help of cheat-sheet format the README.md file that will have bold-fonts, italic-fonts, lists, links

For new lines you can use two spaces and hit enter [or] you can also use <br> tag which is an HTML tag [or] add 2 extra new lines

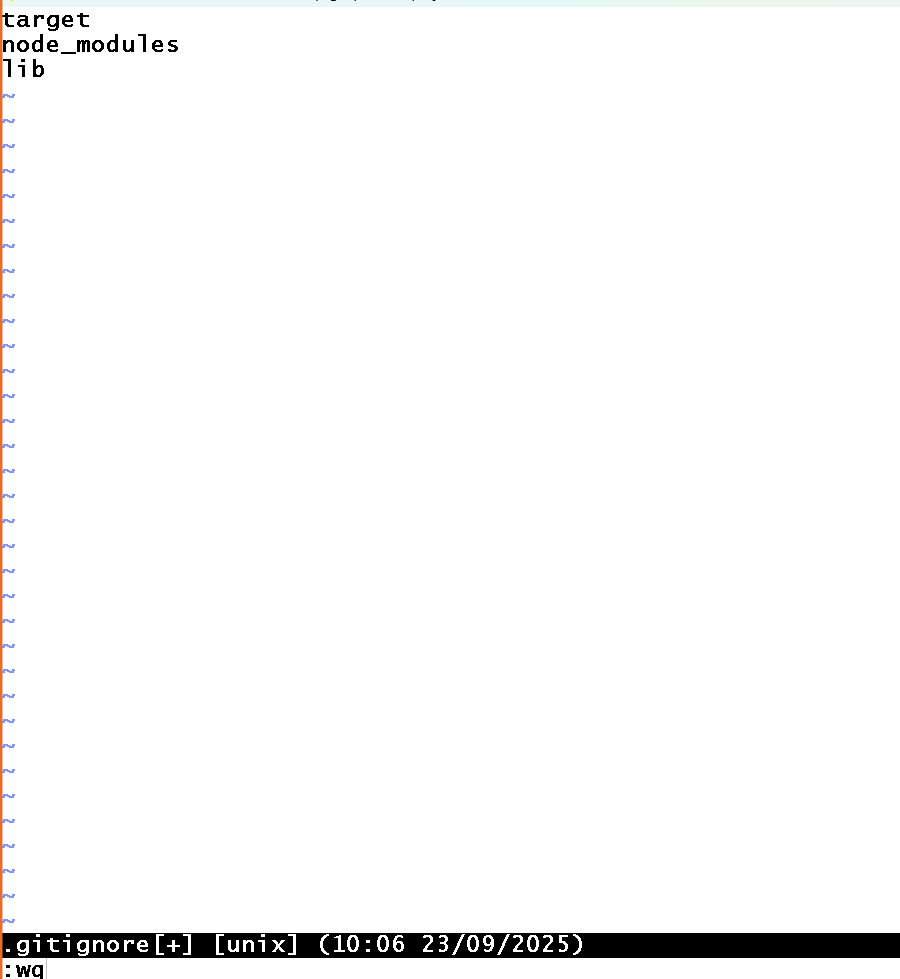
git ignore files

Sometimes you don’t have push project related libraries into git or IDE related files into git ex: node\_modules, target, .settings, .metadata

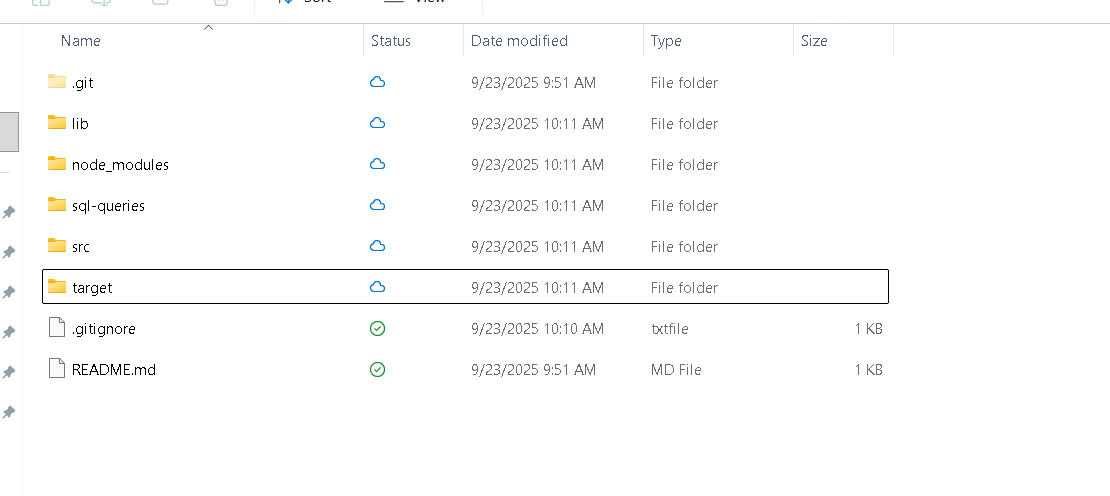
You must create a .gitignore file and mention list of files that shouldn’t be tracked.

touch .gitignore

Edit and add below list of files/folders name in the .gitignore

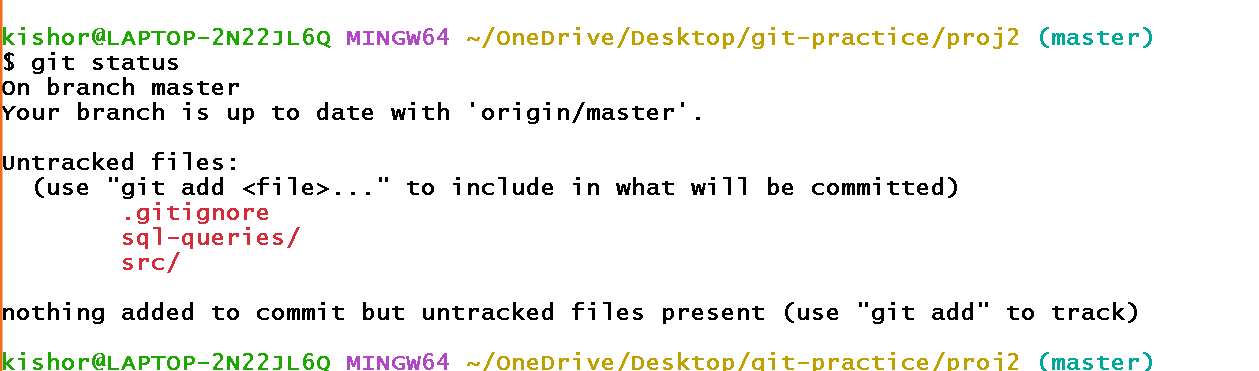


Create src, lib, target, node\_modules, sql-queries folders



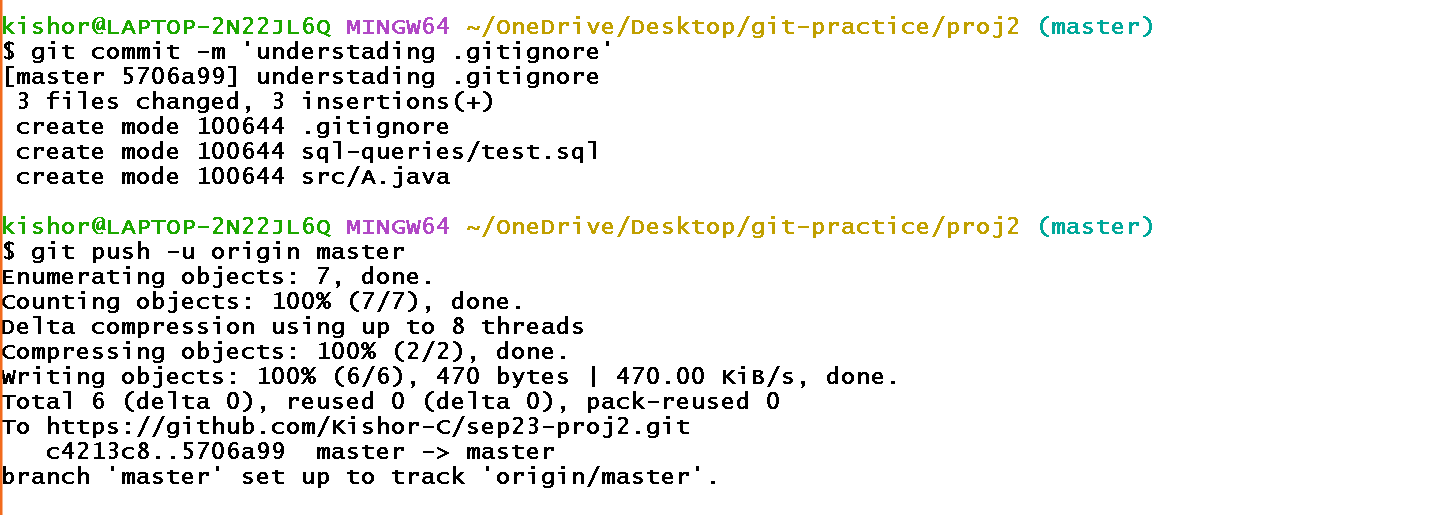


Enter git status to know all the tracked files

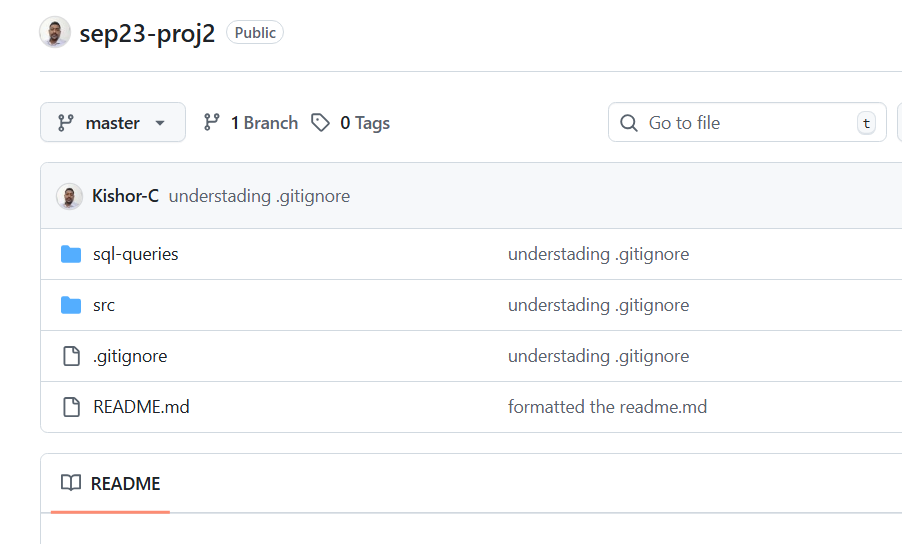


Notic only sql-queries & src are tracked and other folders like lib, target & node\_modules are not tracked, it means they are not pushed to the remote repository.

Try to push this project to the remote



Output:



Git works in many cloud platforms that don’t explicitly use GIT hub repository which are

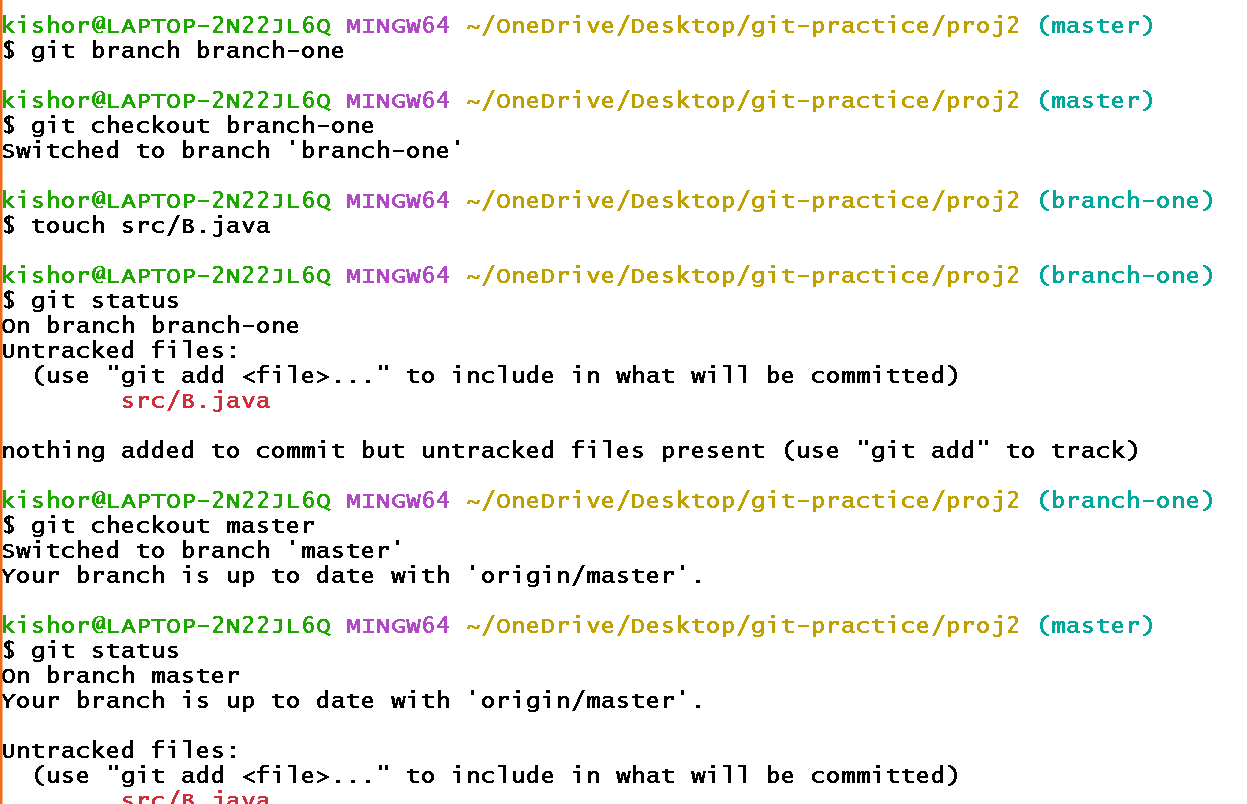
* Bit Bucket
* AWS code commit
* Azure

All these platforms have repositories which are private it means specific to the organization, they are not public for any public contribution. However Git-Hub is public others can also contribute to any project

All these platforms need a sign-in and you can create private repository and through GIT commands you can still update these private repositories

Git stash command

Sometimes you might be in the middle some work and want to switch to a different branch, then you can save using stash so that it won’t appear in other branch.



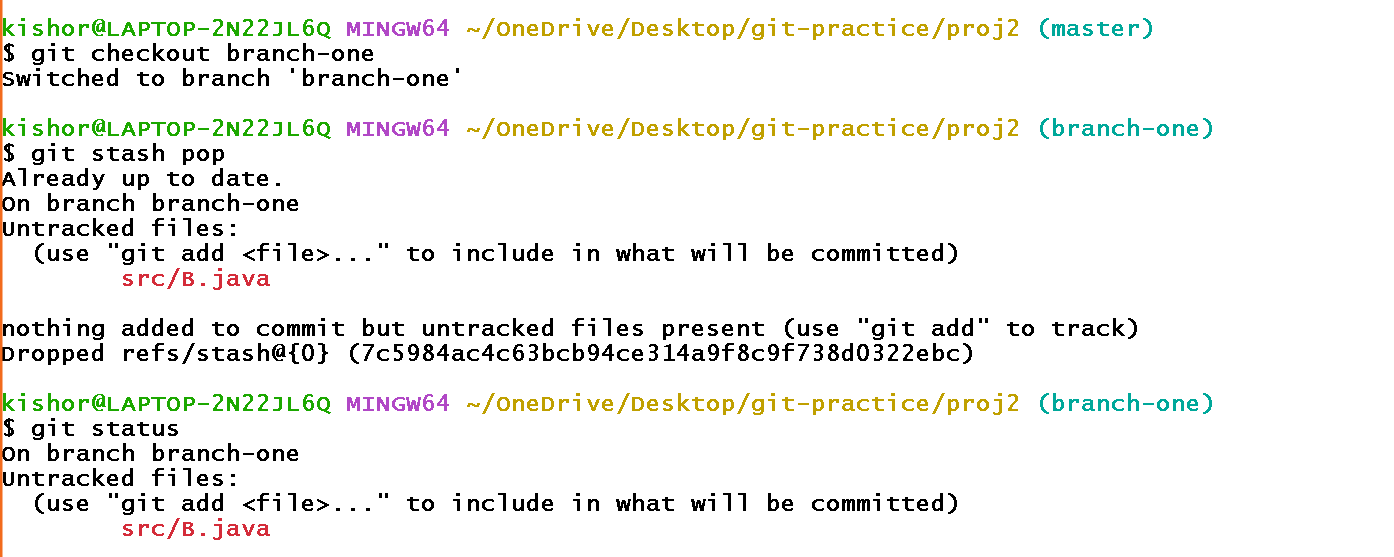
So here branch1 created some file, but when you switch to some other branch that untracked files appear, what we need to do is we must stash it so that when you switch the branch they don’t appear.



You will not see the untracked files in different branch ex: master branch



Now you can get those stashed files if you want to work on it, for that you need to again switch to the branch-one



Assuming that B.java file is completed you can push that to the remote

Listing the branch

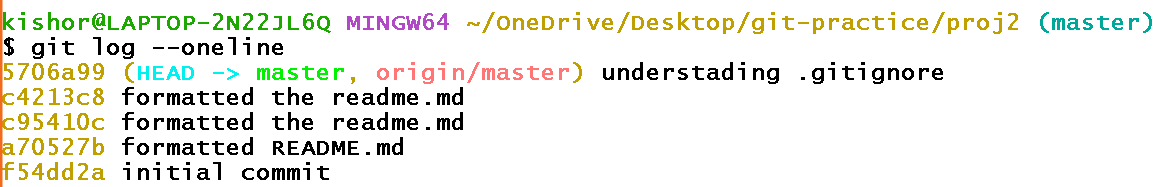
git branch



delete the branch

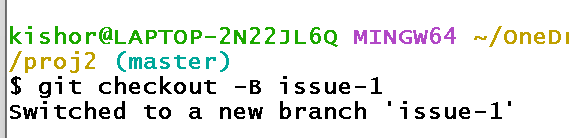


git log --oneline

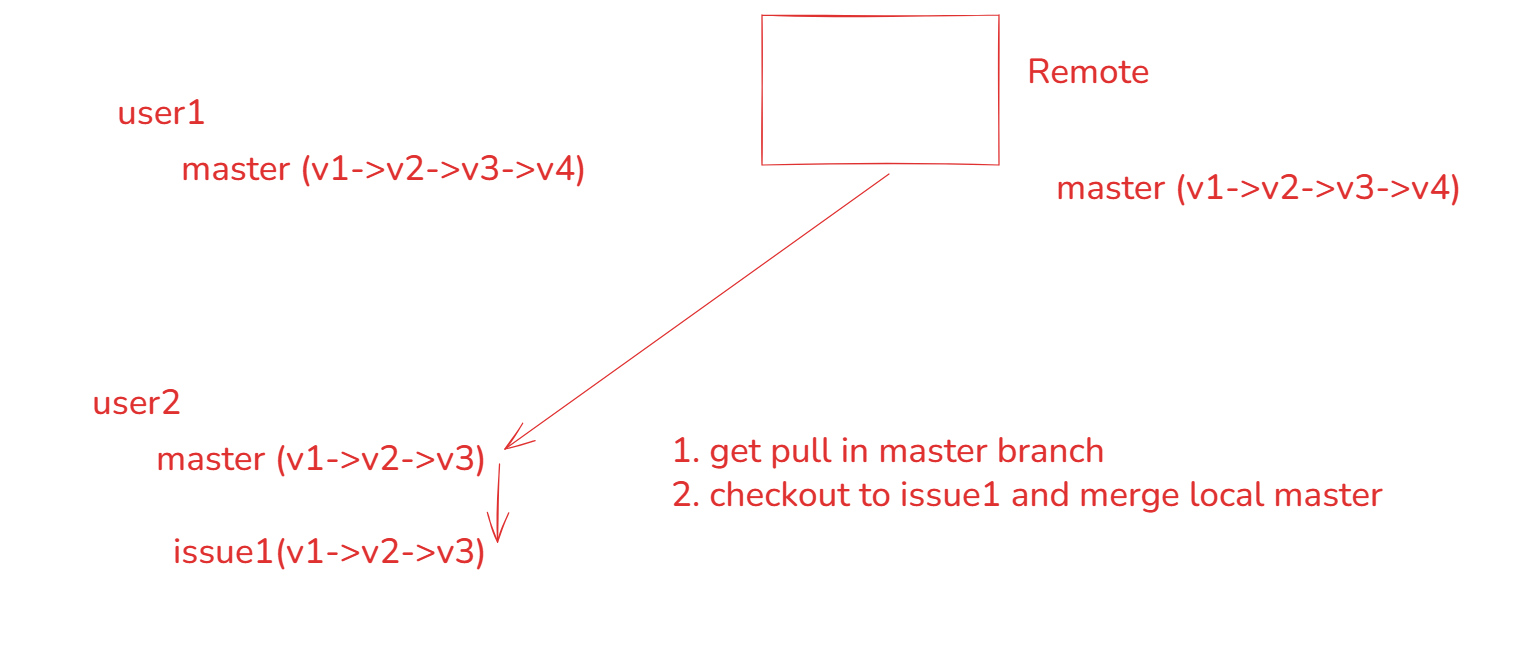


Sometimes you may have many branch in your local repository and you need to know your branch is having which version then these git log will be helpful, using which you can merge in case you want the latest version.

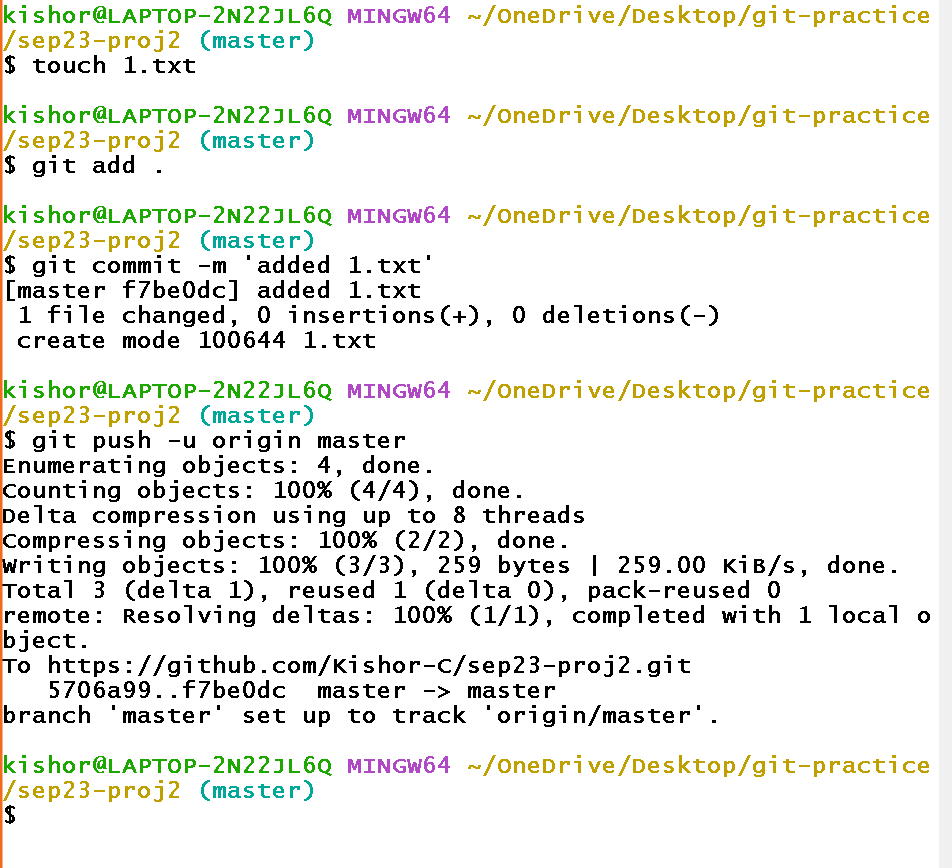
Merging a branch in another branch locally



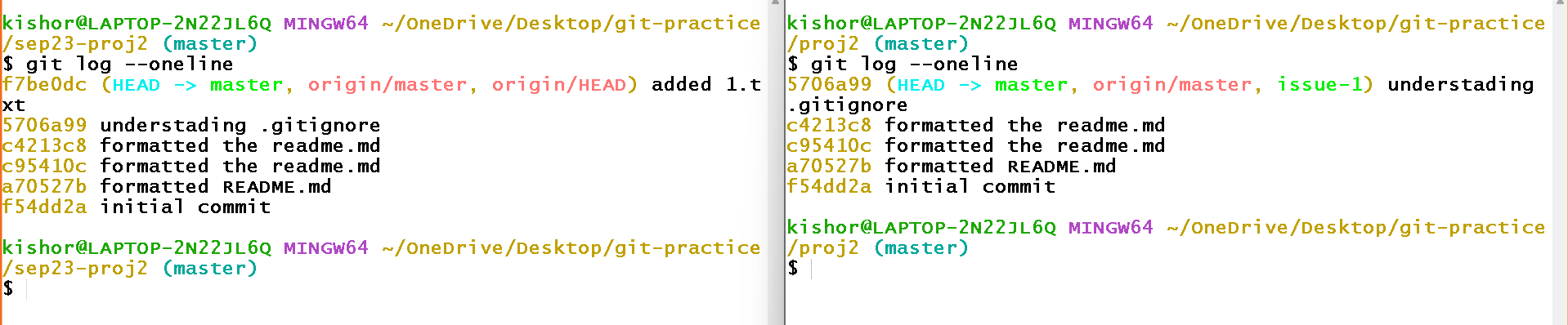
Above command creates a branch if not present



User1 does some changes to the remote master

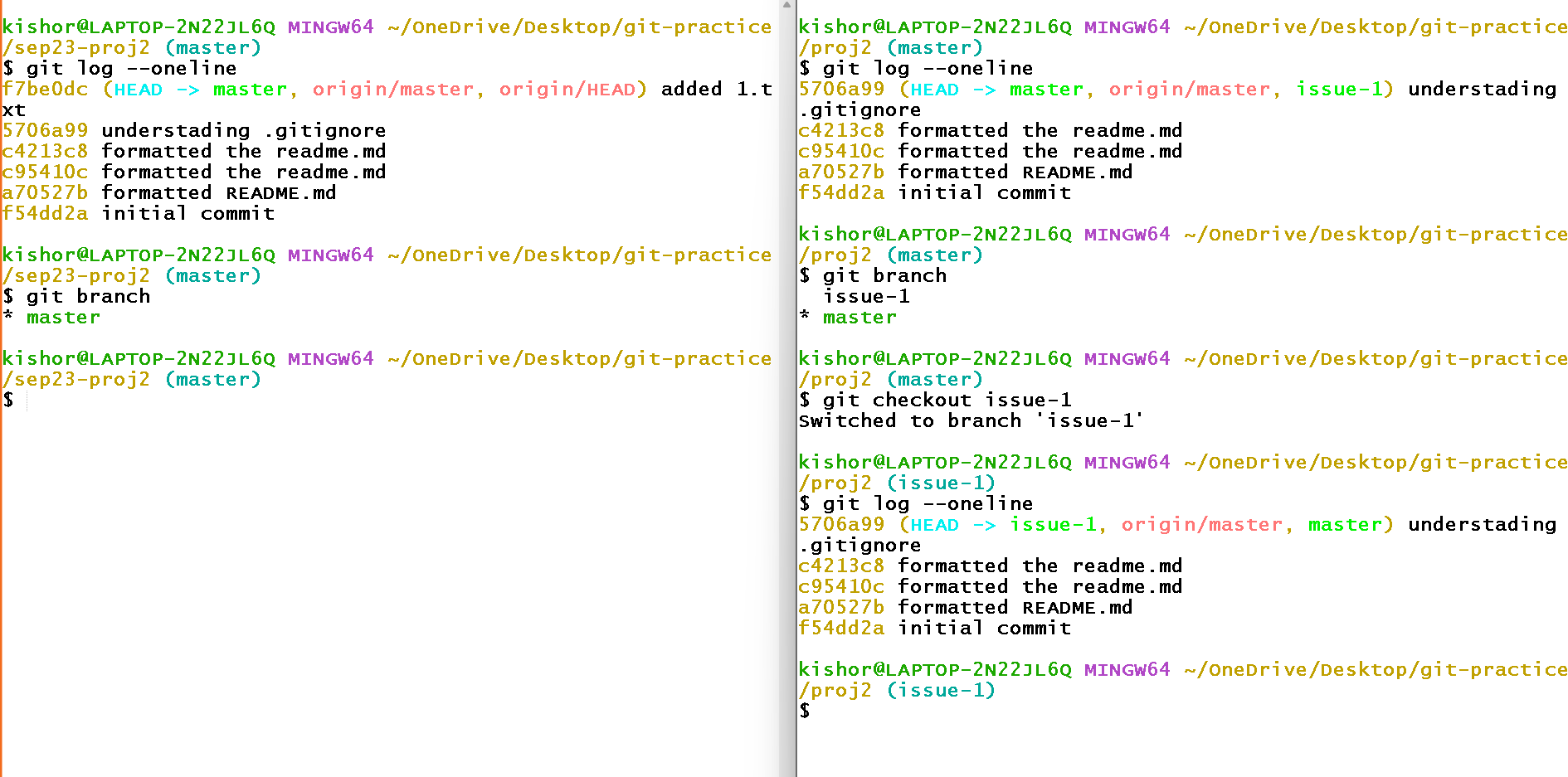


Log the history in both the terminals

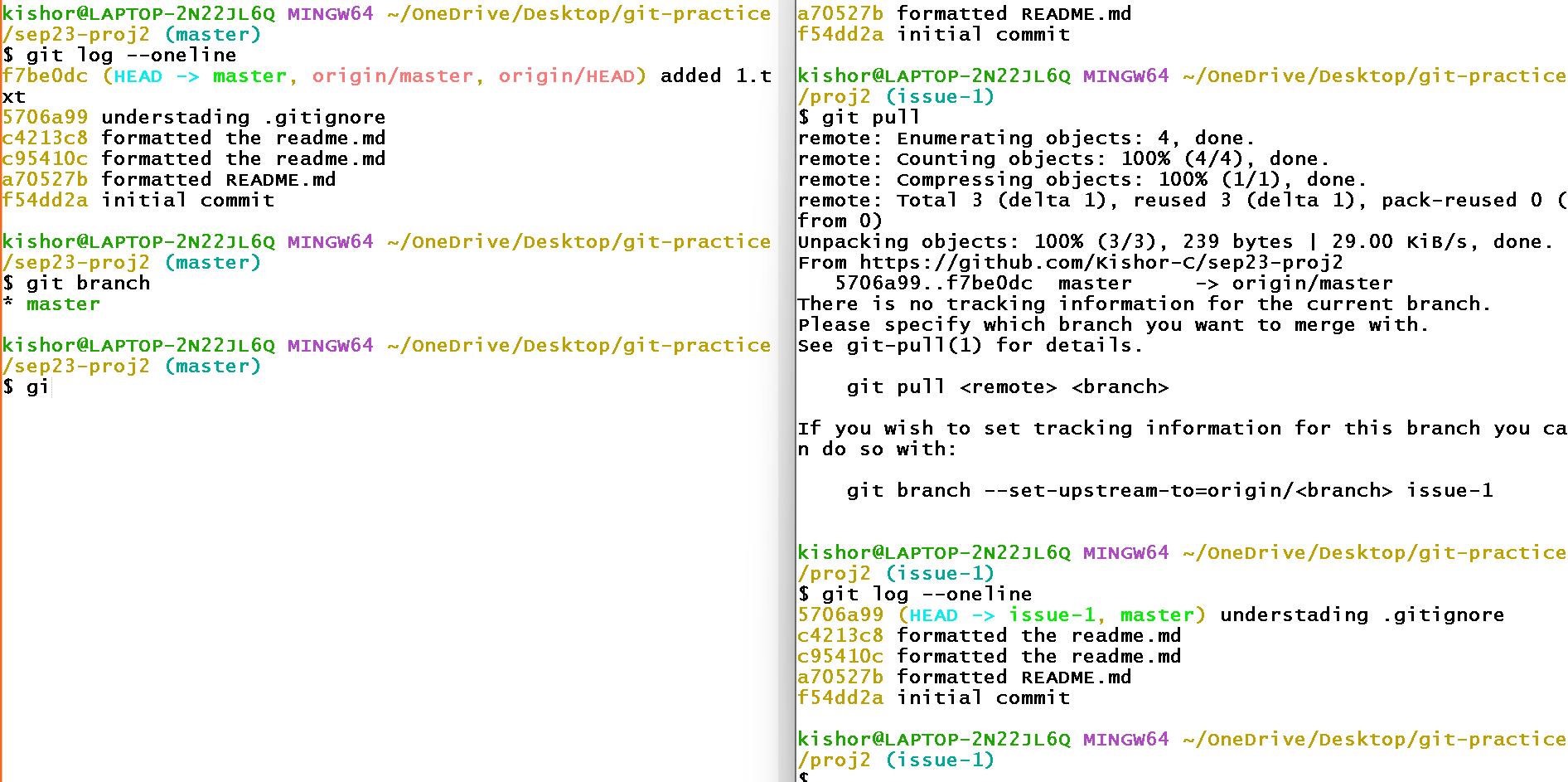


Notice the HEAD which is pointing to the recent version at both left & right terminal, you can observe left terminal is having the latest update, however the right terminal has old updates.

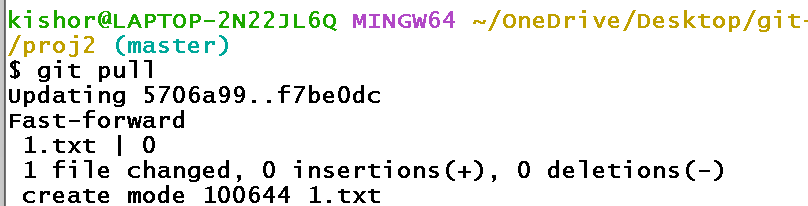
Checkout to issue-1 in the right terminal and log the updates



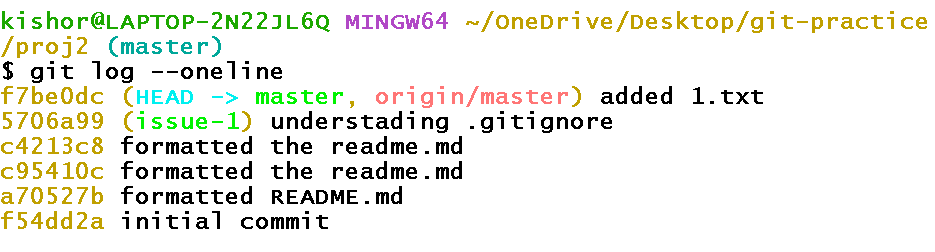
If you use git pull in the issue-1 you don’t see any changes coming from the remote, because there’s no branch with the name issue-1 in the remote



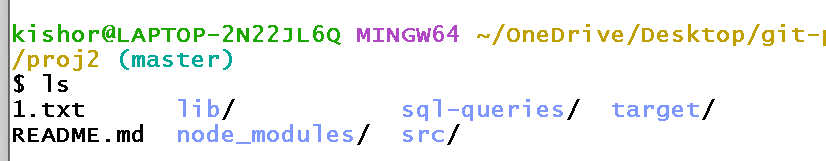
Observe in the 2nd terminal we still have old updates, what we can do is we can checkout to the master branch, pull the remote master & then checkout to the issue-1 branch and then merge so that issue-1 will have the latest update.



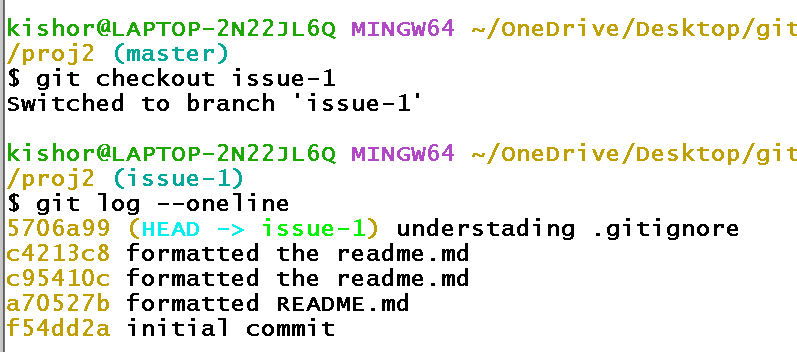
Now you can use git log and observe in the master branch the latest updates



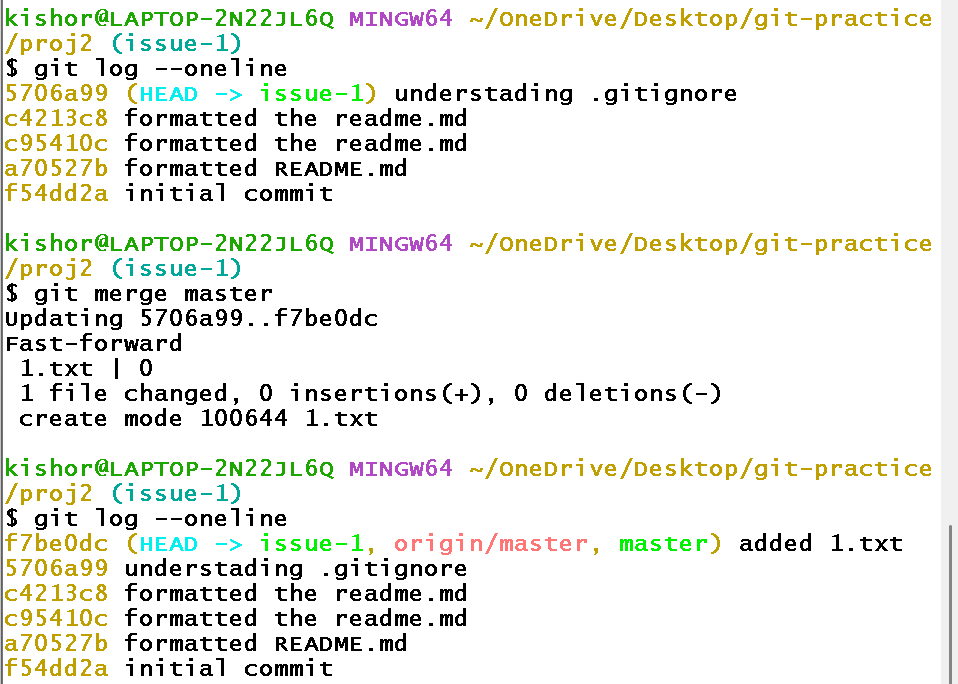
You can also use ls to see 1.txt file in the master branch



If you checkout to the issue-1 you will still have the old version, you must merge local master with issue-1 to get the new version



Notice you are still behind, you can use merge command

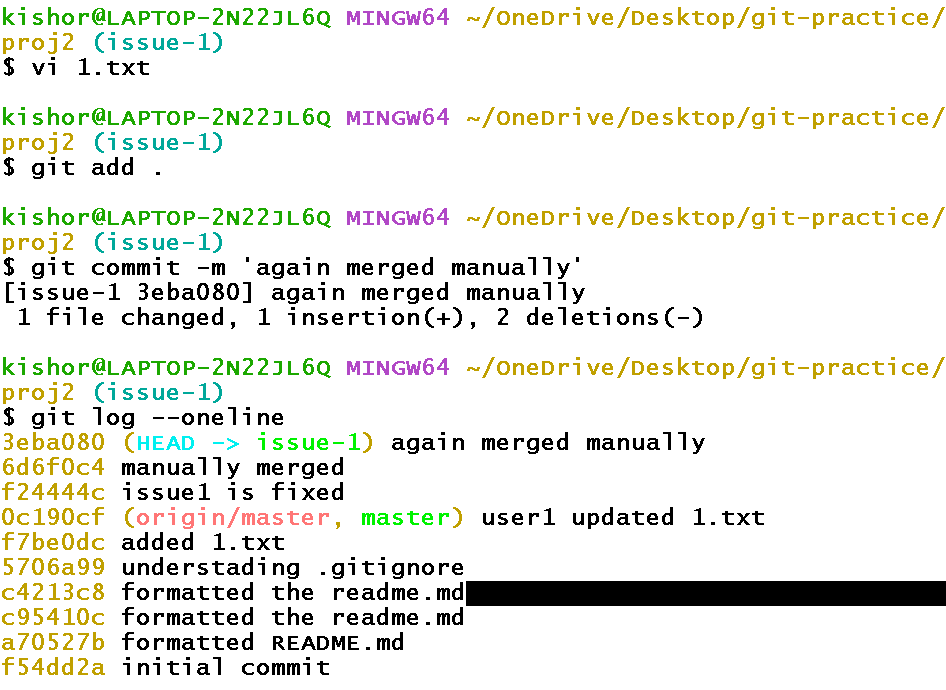


When does the conflict occur locally

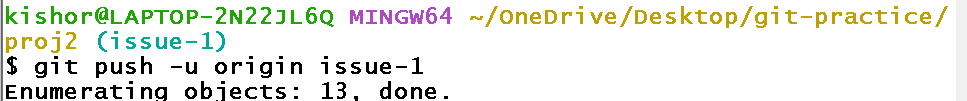
user1(master)-> update something in 1.txt -> push to the remote

user2(issue-1)-> update something in 1.txt & commit -> don’t push to the remote -> pull the remote -> try to merge the local master with issue1 -> you get conflict with a new branch (issue-1 | MERGING)

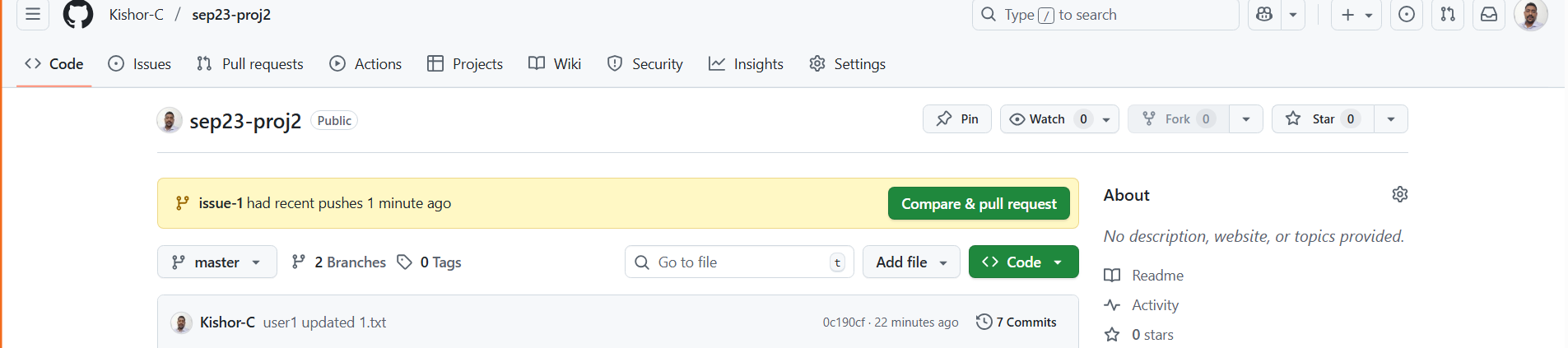
When you get the conflict fix the conflict and enter commit message



Push the issue1 to the remote so that everyone gets the latest update



Now in the remote you must merge the issue-1 to the origin/master



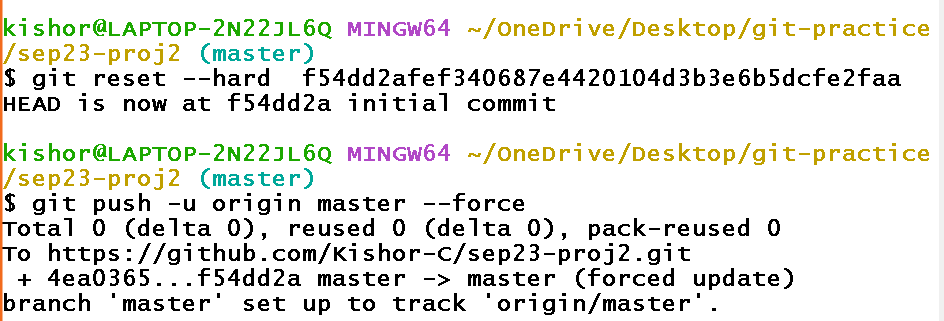
Note: In some case you may not see Compare & pull request, in that can you can manually click on the Pull requests tab and merge the master branch with the feature branch.

Activity:

1. user1(master) -> edit 1.txt -> add to staging area -> commit -> push to the remote
2. user2(issue2) -> edit 1.txt -> commit -> push to the remote
3. origin/master -> merge master with issue2 -> you will get a conflict -> close the merge request
4. user2(master) -> pull the origin/master -> switch to issue2 -> merge the local master -> resolve the conflict -> create a new commit -> push the issue2 to remote
5. remote/master -> merge the issue2 with the remote/master -> this time it succeeds
6. user1(master) & user2(master) will pull the origin/master

Hint: Merge command is “git merge master”

Git reset: It is used to go back to any of the previous commit id



fetch & pull

fetch only downloads the changes from the remote repository but doesn’t merge the changes

pull does fetch + merge

Summary of Git

Git vs Git bash vs Git hub:

GIT is a distributed version control system, which provides set of commands to manage the repository

GIT bash is a command line tool to enter git commands.

GIT hub is a cloud platform which maintains remote repositories

Git commands

git init, git add, git status, git commit, git push, git pull, git log, git branch, git checkout, git clone, git config, git fetch, git merge, git stash, git add remote, git stash, git reset

Day 3

MySQL: It is a relational database owned by oracle to maintain the data in the database in a tabular format.

What is a database

Database is a record to maintain the data in a structured way

What is Database Management System

It is a tool that provides user an interface to interact with the database to perform operations like create, update, delete, read

There are various of DBMS

1. Object -oriented
2. Hierarchical
3. Relational

RDBMS: It stands for Relational Database Management System that maintains the data in a table format which will have rows & columns

List of RDBMS software’s

1. Oracle Database
2. MySQL Database
3. PostgreSQL
4. Derby
5. MS SQL
6. DB2

The language RDBMS understand is SQL (Structured Query Language)

There are 5 types of SQL commands you can enter

1. DDL - Data Definition Language - create, alter, drop & truncate
2. DML - Data Manipulation Language - insert, update & delete
3. DQL/DRL - Data Query/Retrieval Language - select
4. DCL - Data Control Language - grant & revoke
5. TCL - Transaction Control Language - commit, rollback, savepoint

MySQL datatypes

* For Numbers: int, bigint, float, double
* For characters: char, varchar
* For Date & Time: date, timestamp
* For binary/character large object: blob, clob

Create command

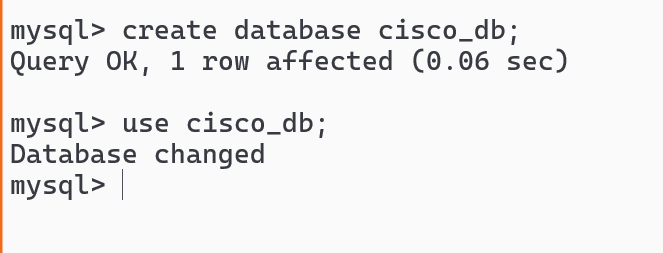
It is a command used to create a table

Syntax:

create table table\_name(column\_name type [constraint], column\_name type,…)

In MySQL you need to create a database name using the below command

create database database\_name;

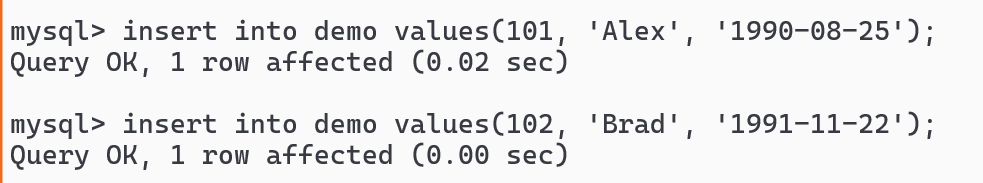


create a demo table



Insert Query

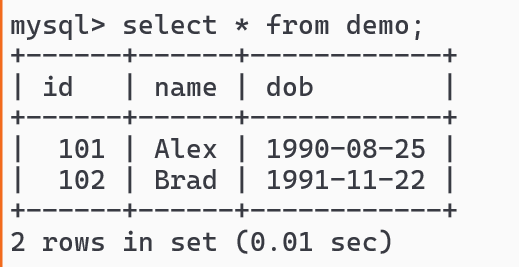
insert into table\_name(col1, col2) values(v1, v2);  
insert into table\_name values(v1, v2,…);



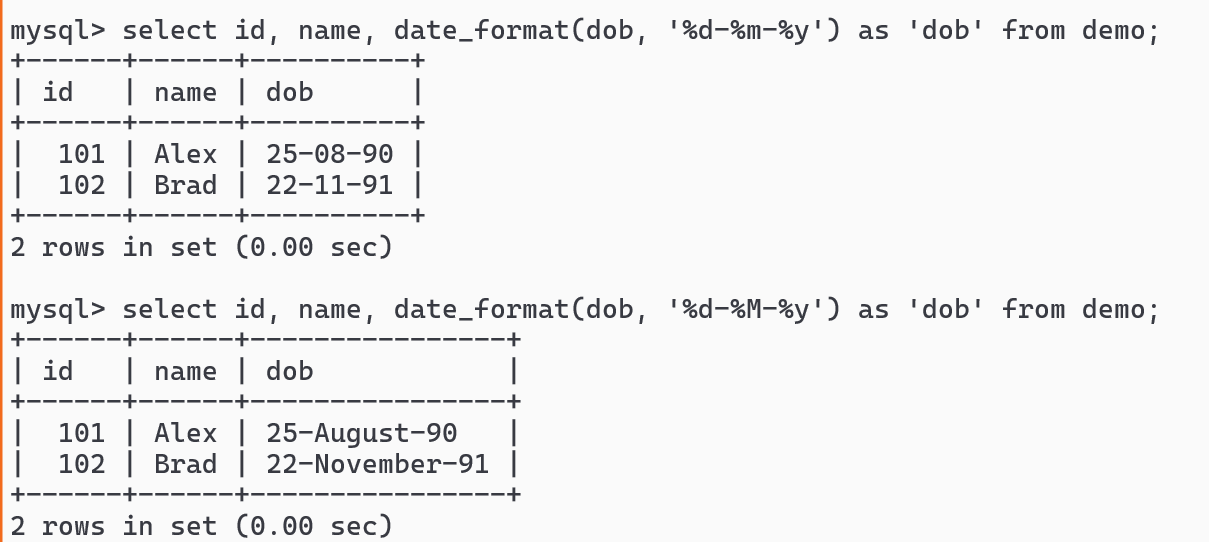
Select Query

select col1, col2 from table\_name;

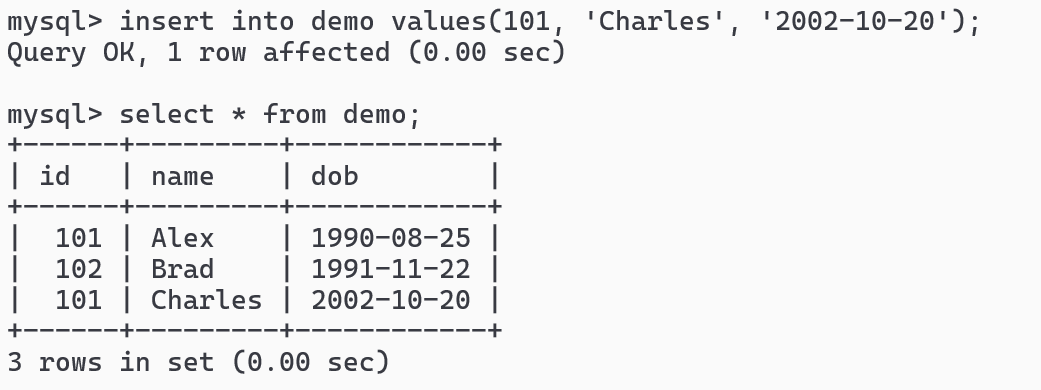
select \* from table\_name;



DATE\_FORMAT(value, ‘format’); This is a built-in function in MySQL to format the date, in the format option you must use %M for month, %d for date & %y for year



What happens if we give duplicate id in the demo table



Since there’s no constraint in demo table duplicate id’s are getting stored.

Constraints in database

Constraints are the rules you can apply on a table or a column to restrict the value, there are following constraints you can use in MySQL

1. PRIMARY KEY
2. UNIQUE
3. NOT NULL
4. CHECK
5. FOREIGN KEY

PRIMARY KEY: it is to uniquely identify the row, it doesn’t allow duplicates & null, mainly it is used for values that don’t change ex: employee\_id, customer\_id, account\_number

UNIQUE: It is to avoid duplicate values but it supports null, multiple rows can have null, each null is unique, ex: pan, aadhar, mobile\_number, email\_id

NOT NULL: It is used when a column mandatorily need value, ex: username, password, name

CHECK: It is used when a column needs to have the values that meets certain condition ex: designation, age, gender.

FOREIGN KEY: It is used when a table needs to have some dependent table i.e., customer & account table, employee & department table, customer & loan,

Adding primary key

1. You can add primary key while creating the table
2. If the table is already present you can use alter command to add the primary key, but the columns must not have any duplicates

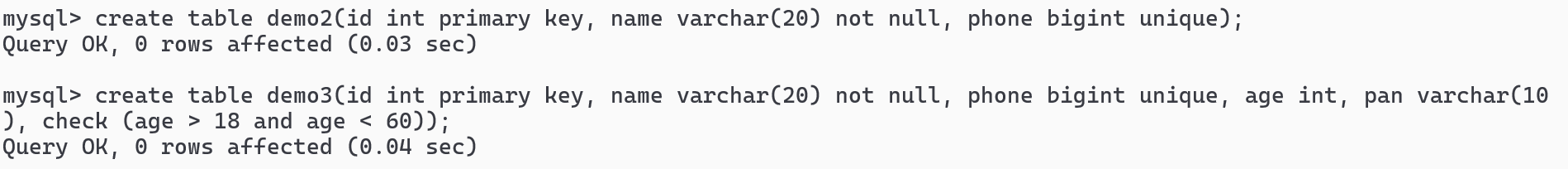
Create table with primary key

create table table\_name(col type primary key, col type, col type);

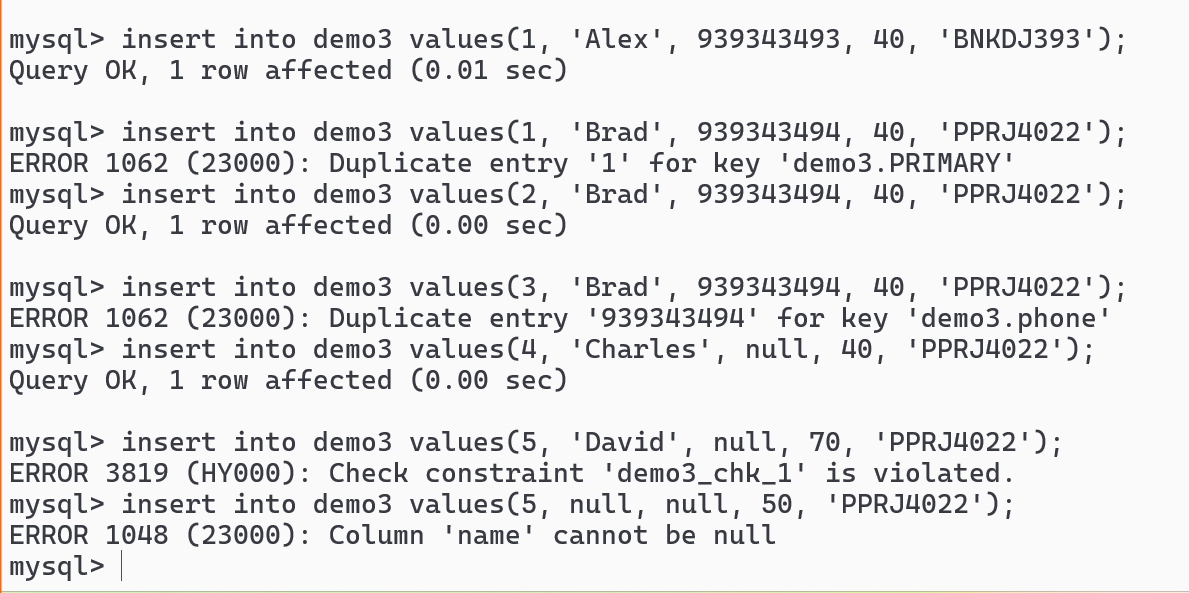
Create table with other constraints - not null, unique

create table table\_name(col type primary key, col type not null, col type unique);

Check constraint needs to be written after defining all the columns



Lets see what happens when the constraints are violated

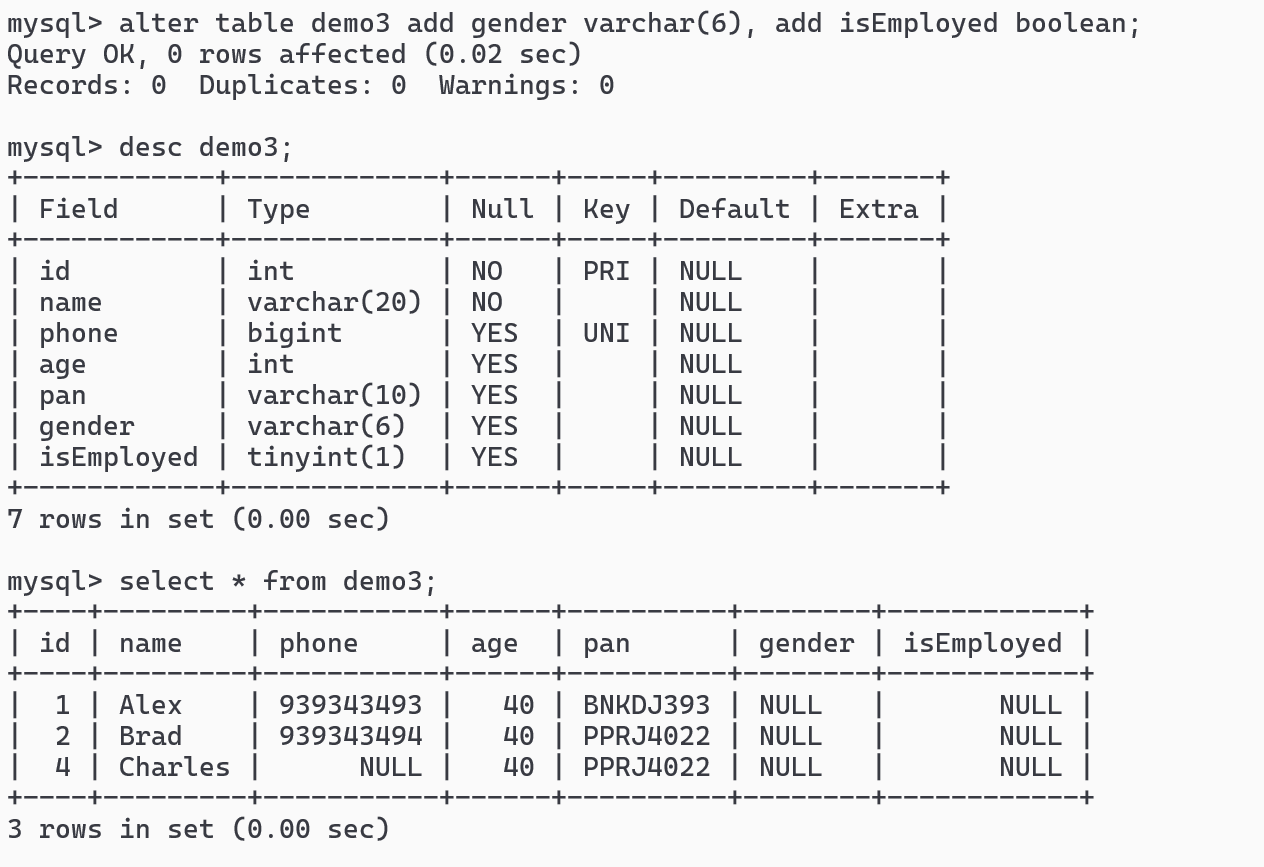


ADDING & DROPPING the columns

alter table table\_name add column\_name type, add column\_name type,…

alter table table\_name drop column column\_name, drop column column\_name

ADDING columns

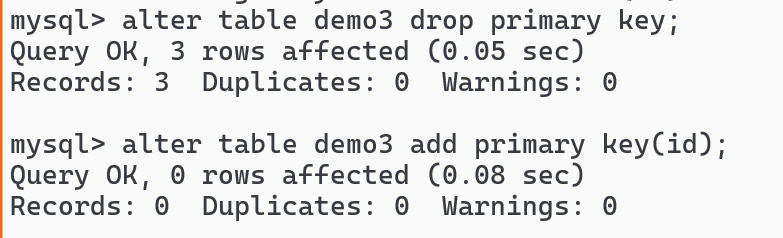


DROPPRING column



ADD & DROP primary key

alter table table\_name add primary key(column);  
alter table table\_name drop primary key;



TRUNCATE

It deletes all the records from a table and you can’t undo(rollback) the changes

truncate table table\_name

DROP

It deletes the table & you can’t undo this

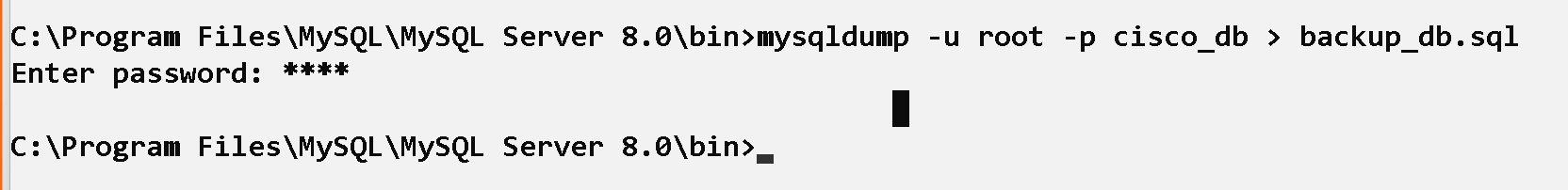
drop table table\_name;

Backup and restore

mysqldump -u username -p database\_name > backup\_db.sql

mysql -u username -p database\_name < backup\_db.sql

Backup command: Open cmd as admin in windows & for mac use sudo before the command



For mac-users: open a new terminal & enter below commands

**export PATH=”../../usr/local/mysql/bin:$PATH”**

In the same terminal: **mysqldump -u root -p cisco\_db > backup\_db.sql**

Login to the mysql

mysql -u root -p

Enter password

Switch to cisco\_db -> drop the demo table

Exit from mysql

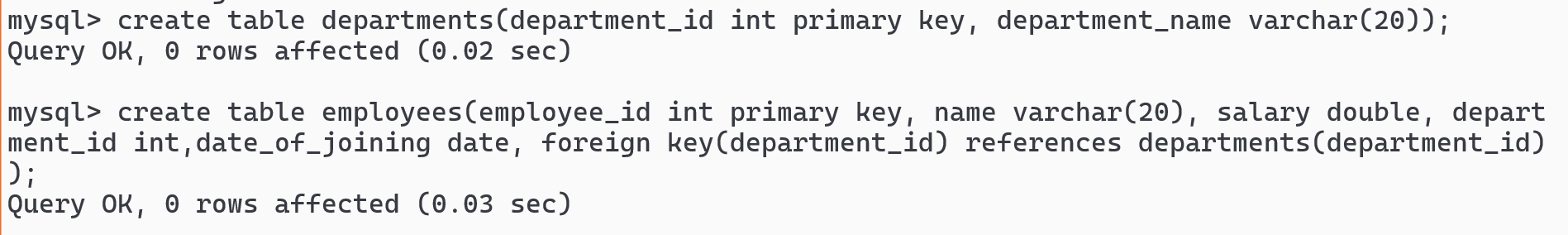
Restore the database backup file : **mysql -u root -p cisco\_db < backup\_db.sql**

Login to the mysql -> You must see the demo table in cisco\_db

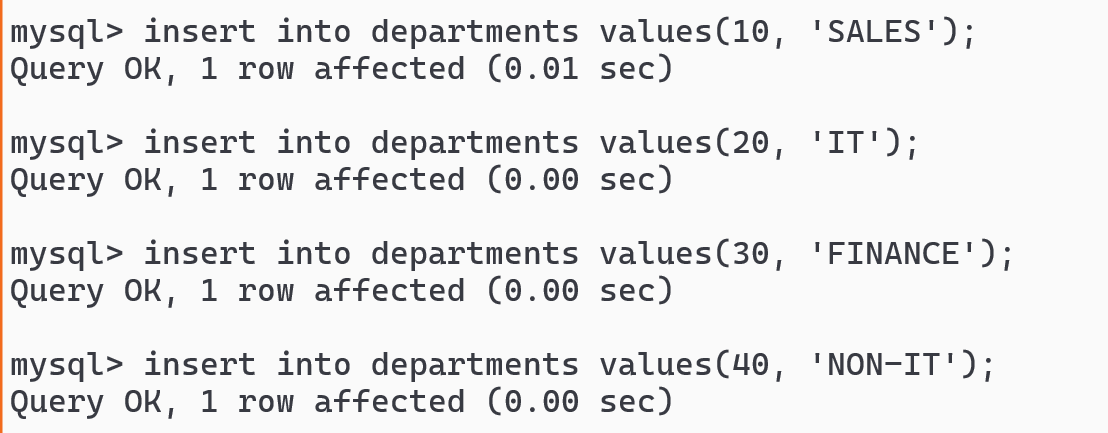
FOREIGN KEY

Whenever a table is related to another table you can use foreign key constraint, a table can have multiple foreign keys

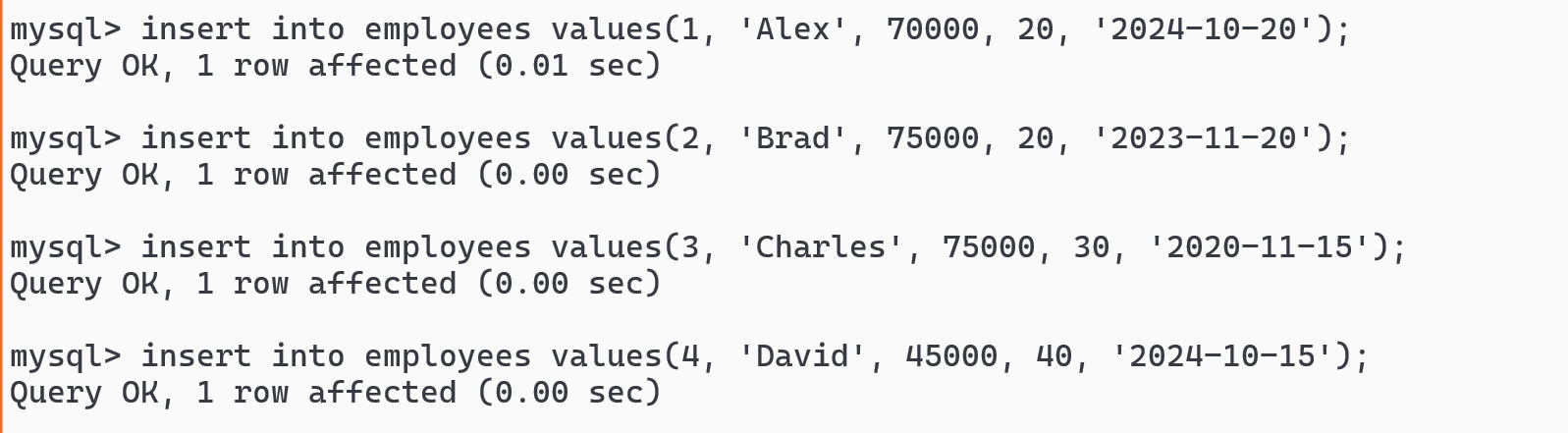
create table table\_name(col1 type, col2 type,..foreign key(column\_name) references parent\_table(primary\_key\_column)



Store some records in departments



store some records in employees

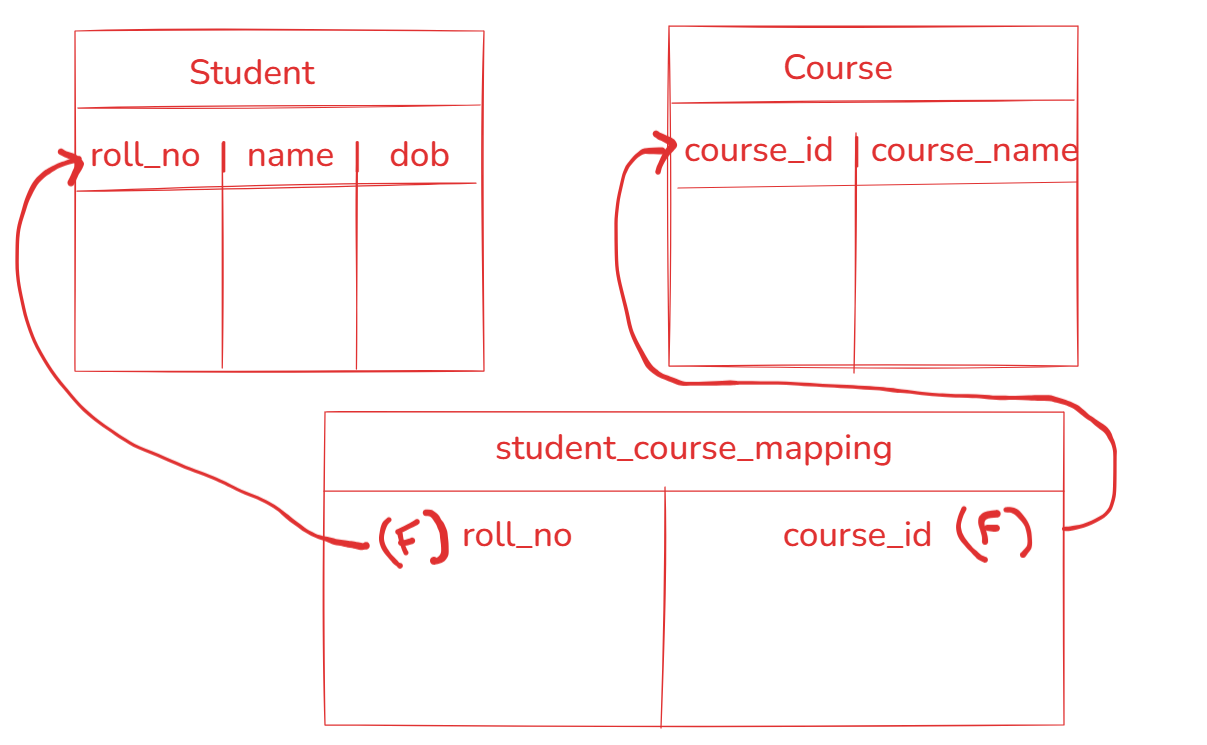


Try to store NULL in department\_id in employees table

Try to store a value that’s not part of deparments.department\_id

Activity

1. create student, course & student\_course\_mapping tables, student table will have student details, course table will have course details and the mapping table will have roll no and course id to specify which student has registered to which course, store some records in all the tables



Update & Delete commands

Update

update table\_name set col=value, col=value; # updates all the rows

update table\_name set col=value, col=value where <<condition>>; # updates only the rows matches to the condition

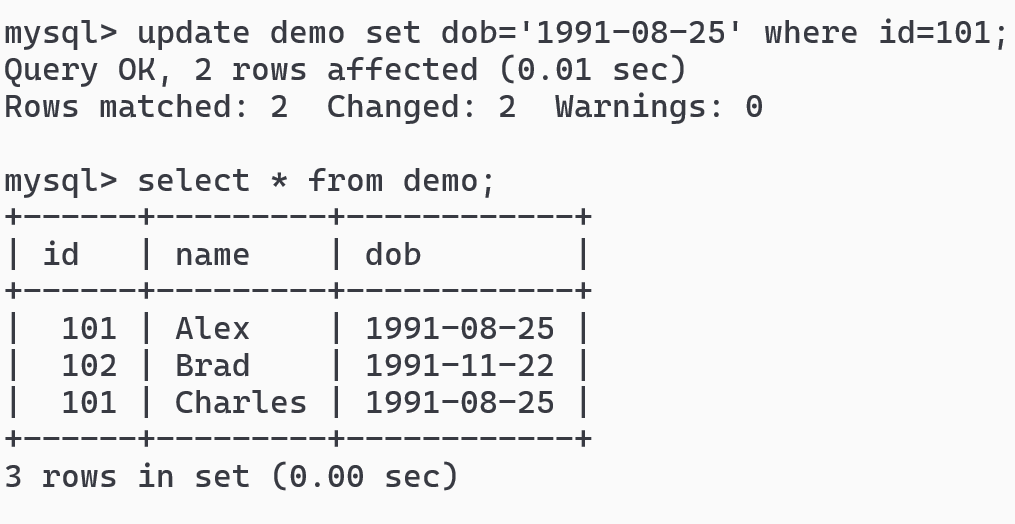
Delete

delete from table [or] delete from table where <<condition>>

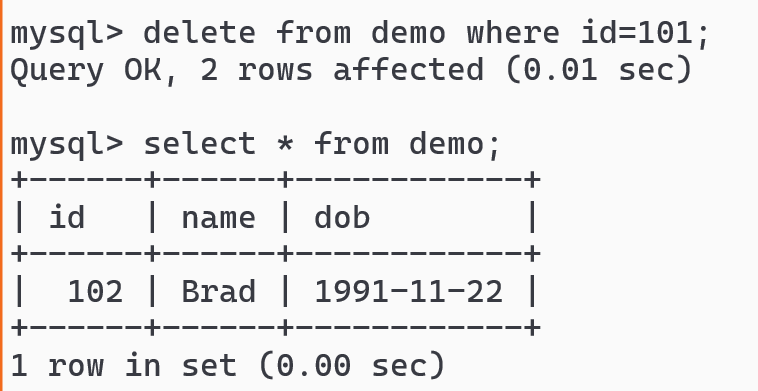
Activity on a demo table:

1. Update the dob of a row that matches to a particular id
2. Delete the row that matches to a particular id

Updating dob based on id



Deleting the row based on id

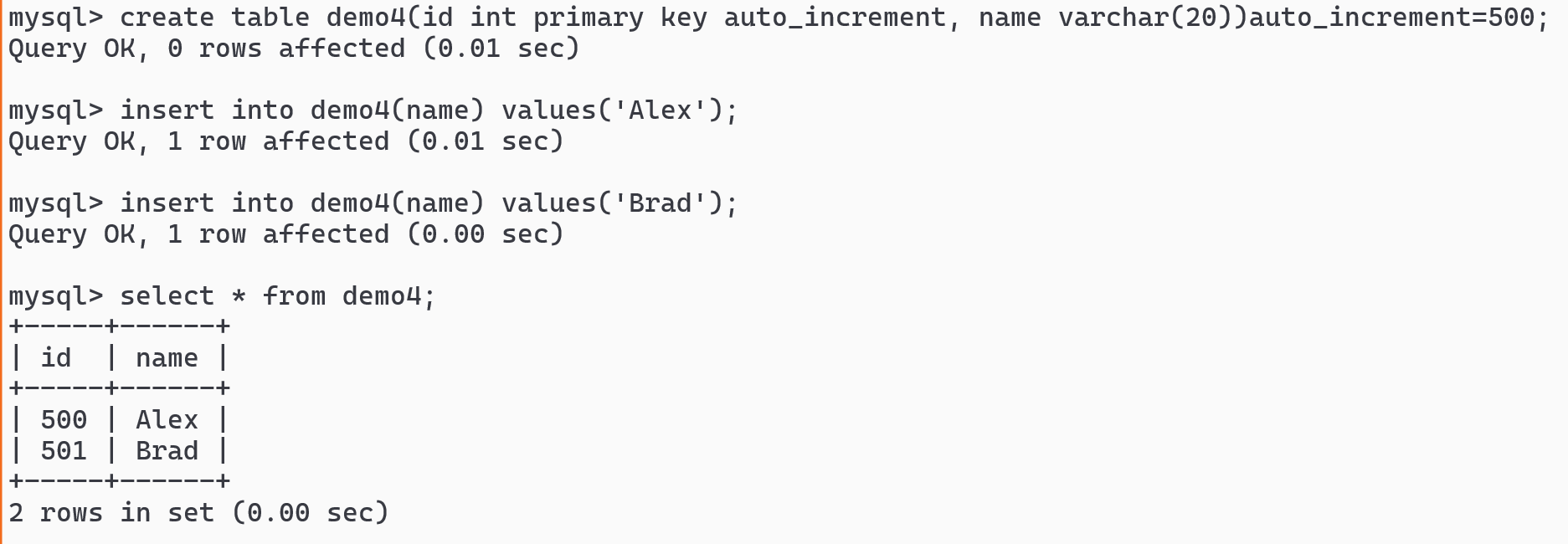


Auto\_Increment: This keyboard help primary key to automatically increment by 1

You can use this while creating a table or with alter command also.

create table table\_name(col type primary key auto\_increment, …) auto\_increment=100;

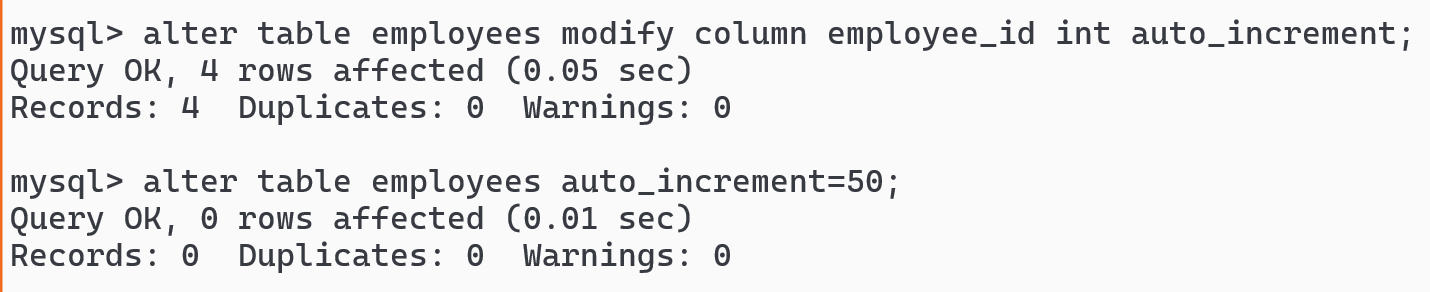
when you specify auto\_increment value then the value starts from the number you have specified else by default it starts from 1.



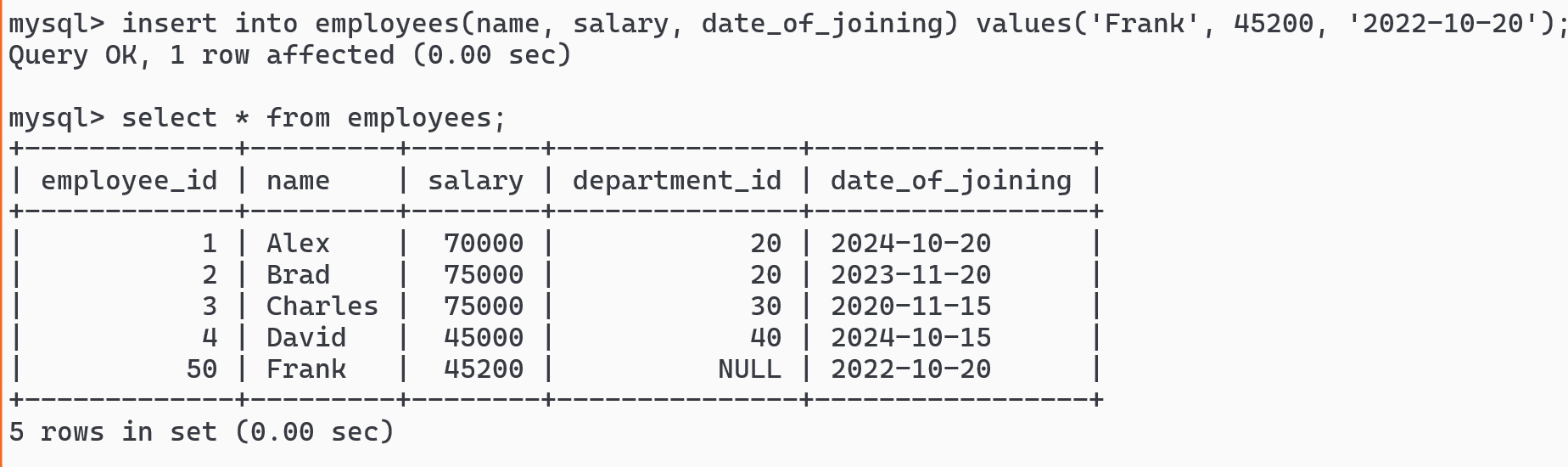
auto\_increment with alter command: when the table is already created but the column doesn’t have auto\_increment

alter table table\_name modify column column\_name type auto\_increment;

alter table table\_name auto\_increment=100;



You can now the employees record with employee\_id

Running the SQL script files

You need to have an SQL file and mention the path of that SQL file using SOURCE keyword

SOURCE path-to-SQL-file

create myscripts.sql in users folder

