Q1 create a folder /dummy/newfolder using the single command inside the Desktop

- assume both the folders are not present

**Answer** ----->> *mkdir -p /dummy/newfolder*

Q2 You need to create the folder in the particular pattern as emp, emp2, emp3 .... till. emp100 [only with single line of code]

- Take its backup to the another folder inside the desktop folder

**Answer** ----->> *mkdir emp{1..100}*

Q3 Search for any file having:

{ am } any file with am in the middle of the filename

**Answer** ---->> *ls ?am\**

- search for any file & folder with second last word as {s} & starting words as {t}

**Answer** ----->> *ls -d t\*s?*

- Search for any file that can start from character such as {A-Z or a..z, 0-9 }

**Answer** ----->> *ls A[A-z]\* a[a-z]\* 0[0-9]\**

- Find out any files inside the Desktop location created by the cloudera user.

[Hint: You need to use the find commands for the same]

**Answer** ----->> *sudo find / -user cloudera*

----------------------

Q4 How to create user with any name

**Answer** ----->> *sudo useradd deep*

- Create user with userid 11100

**Answer** ------>> *sudo useradd -u 11100 temp*

- Create a user with home-directory inside /home

**Answer** ---->> *sudo useradd -m -d /var/www/ home\_new*

- Creating a user and giving comment -> -c option

**Answer** ----->> *sudo useradd -c "Deepika Goel" deeps*

-------------

Q5: What are group, primary & secondary group?

**Groups:** In Linux, a group is a collection of users. The main purpose of the groups is to define a set of privileges like read, write, or execute permission for a given resource that can be shared among the users within the group. Users can be added to an existing group to utilize the privileges it grants.

**Secondary or supplementary group** - used to grant certain privileges to a set of users. A user can be a member of zero or more secondary groups. Secondary groups are those that users might be added to once they already have accounts. Secondary group memberships show up in the /etc/group file. The /etc/group file assigns names to user groups (e.g., 500 = devops) and records secondary group members.

**Primary or login group** is the group that is assigned to the files that are created by the user. Usually, the name of the primary group is the same as the name of the user. Each user must belong to exactly one primary group.The primary group is the one that is recorded in the /etc/passwd file, configured when an account is set up. When a user creates a file, it's their primary group that is associated with it.

Q6 Permission in Linux, values for each permission

**Owner permissions:** The owner's permissions determine what actions the owner of the file can perform on the file.

**Group permissions:** The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.

**Other (world) permissions** The permissions for others indicate what action all other users can perform on the file.

While using ls -l command, it displays various information related to file permission as follows:

*ls -l /home/amrood*

*-rwxr-xr-- 1 amrood users 1024 Nov 2 00:10 myfile*

*drwxr-xr--- 1 amrood users 1024 Nov 2 00:10 mydir*

the first column represents different access modes, i.e., the permission associated with a file or a directory.

The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x)

The first three characters (2-4) represent the permissions for the file's owner. For example, -rwxr-xr-- represents that the owner has read (r), write (w) and execute (x) permission.

The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example, -rwxr-xr-- represents that the group has read (r) and execute (x) permission, but no write permission.

The last group of three characters (8-10) represents the permissions for everyone else. For example, -rwxr-xr-- represents that there is read (r) only permission.

**File Access Modes**

The permissions of a file are the first line of defense in the security of a Unix system. The basic building blocks of Unix permissions are the read, write, and execute permissions, which have been described below

**Read**

Grants the capability to read, i.e., view the contents of the file.

**Write**

Grants the capability to modify, or remove the content of the file.

**Execute**

User with execute permissions can run a file as a program.

**Directory Access Modes**

Directory access modes are listed and organized in the same manner as any other file. There are a few differences that need to be mentioned âˆ’

**Read**

Access to a directory means that the user can read the contents. The user can look at the filenames inside the directory.

**Write**

Access means that the user can add or delete files from the directory.

**Execute**

Executing a directory doesn't really make sense, so think of this as a traverse permission.A user must have execute access to the bin directory in order to execute the ls or the cd command.