NEW HORIZON
COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

DEPARTMENT OF MCA

LECTURE NOTES

(All 5 Modules Included)

DATABASE ADMINSTRATION USING DB2 LAB

23NHOPG14

Database Administration Using DB2

1. What is Database Administration (DBA)?

- A **Database Administrator (DBA)** is someone who takes care of databases in a company.
- Think of it like being the **caretaker of data** — making sure everything works smoothly.
- DBAs do things like:
    - Create and organize databases
    - Keep them secure
    - Take backups in case something goes wrong
    - Help users get the right data quickly

---

2. What is DB2?

- **DB2** is a database system made by **IBM**.
- It stores and manages **large amounts of data** for big companies (like banks, hospitals, etc.).
- It works on different systems: Windows, Linux, Unix, and even IBM mainframes.
- Used in **real-time systems**, where data must be fast and accurate.

3. Why Learn DB2?

- It's used in **real companies** – knowing it can help you get internships or jobs.
- Great for learning how **real-world data systems** work.
- IBM offers **free DB2 Community Edition** for students.

4. Basic Terms You Should Know

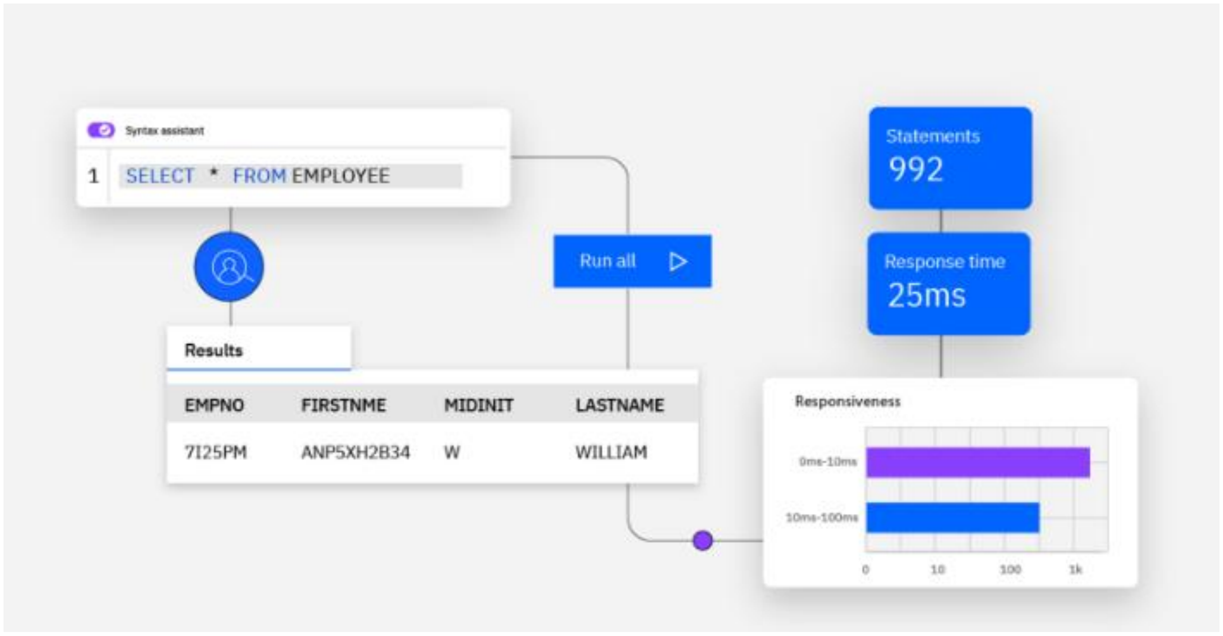
Term	Meaning
Instance	The main DB2 environment – like the manager of all databases
Database	Where data is stored – like a digital cupboard
Table	Stores data in rows and columns – like Excel
User	A person who accesses the data
Backup	A copy of the data in case of crash/loss

5. What Can a DB2 DBA Do?

- **Create and Manage Databases**
- **Give or Block Access to Users**
- **Make the Database Faster**
- **Clean and Organize Tables**
- **Take Regular Backups**
- **Check Errors and Fix Issues**

About IBM DB2

IBM DB2 is a family of data management products, including database servers, developed by IBM. It initially supported the relational model, but was extended to support object–relational features and non-relational structures like JSON and XML.



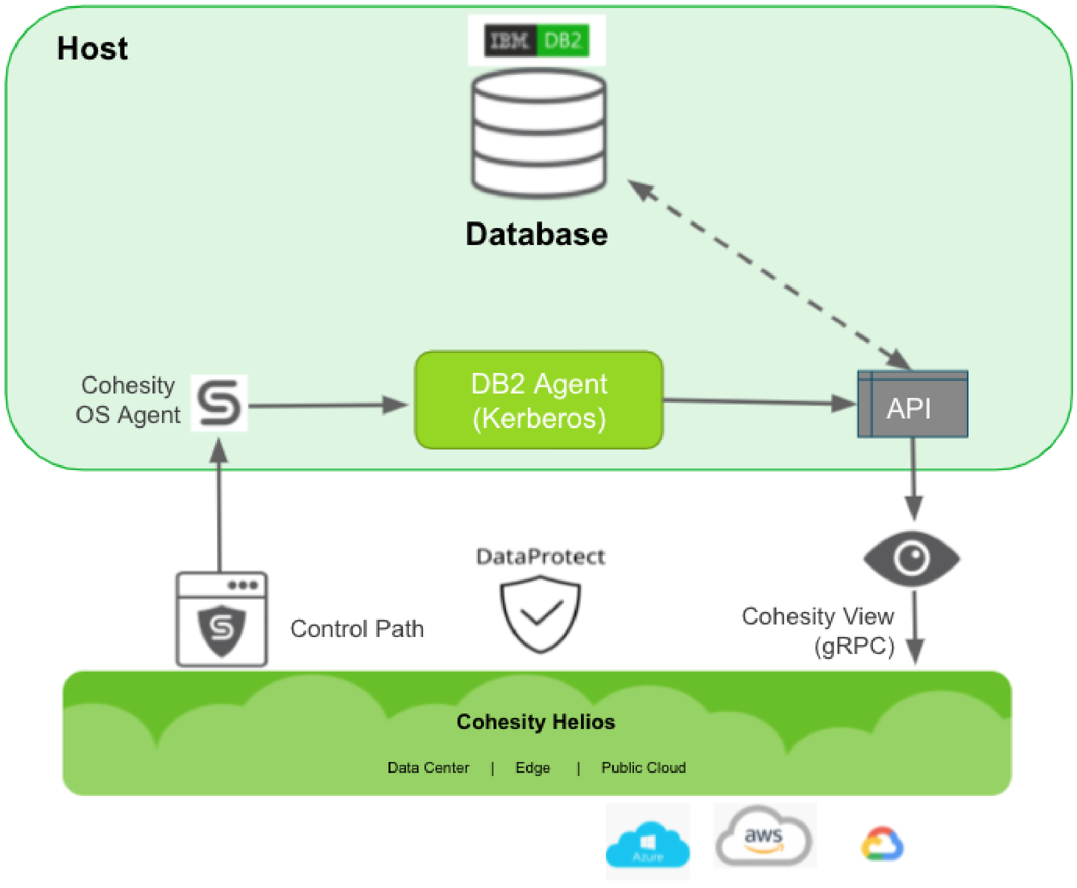
```
SELECT * FROM EMPLOYEE
```

EMPNO	FIRSTNME	MIDINIT	LASTNAME
7I25PM	ANP5XH2B34	W	WILLIAM

IBM DB2 is a cloud-native database that provides a single engine for DBAs, enterprise architects and developers to run low-latency transactions and real-time analytics at scale. It offers a single view of data across cloud data lakes and Hadoop, built-in security, scalability and availability, and tight integration with watsonx.data lakehouse and other tools.

IBM DB2 is also capable of supporting mixed transactional and analytical workloads, data governance and security, and AI-driven applications.

IBM DB2 Big SQL allows you to query data across cloud data lakes and Hadoop with a high-performance, massively parallel SQL engine built with advanced multimodal and multicloud capabilities. IBM DB2 is available as SaaS on IBM Cloud or for self-hosting.


Host
IBM DB2
Database
Cohesity
OS Agent
DB2 Agent
(Kerberos)
API
Control Path
DataProtect
Cohesity View
(gRPC)
Cohesity Helios
Data Center | Edge | Public Cloud
Azure
aws

MODULE 1 – LINUX COMMANDS

Unix commands are a set of commands that are used to interact with the Unix operating system. Unix is a powerful, multi-user, multi-tasking operating system that was developed in the 1960s by Bell Labs. Unix commands are entered at the command prompt in a terminal window, and they allow users to perform a wide variety of tasks, such as managing files and directories, running processes, managing user accounts, and configuring network settings. Unix is now one of the most commonly used Operating systems used for various purposes such as Personal use, Servers, Smartphones, and many more. It was developed in the 1970's at AT& T Labs by two famous personalities Dennis M. Ritchie and Ken Thompson.

- You'll be surprised to know that the most popular programming language C came into existence to write the Unix Operating System.
- **Linux is Unix-Like operating system.**
- The most important part of the Linux is Linux Kernel which was first released in the early 90s by Linus Torvalds.There are several Linux distros available (most are open-source and free to download and use) such as Ubuntu, Debian, Fedora, Kali, Mint, Gentoo, Arch and much more.
- Now coming to the Basic and most usable commands of Linux/Unix part. (Please note that all the linux/unix commands are run in the terminal of a linux system.Terminal is like command prompt as that of in Windows OS)
- Linux/Unix commands are **case-sensitive** i.e Hello is different from hello**.**

Basic unix commands:

1. who : The '$ who' command displays all the users who have logged into the system currently. As shown above, on my system I am the only user currently logged in.The thing tty2 is terminal line the user is using and the next line gives the current date and time

```
$ who

Output: harssh tty2 2017-07-18 09:32 (:0)
```

2. pwd : The ‘$pwd’ command stands for ‘print working directory’ and as the name says,it displays the directory in which we are currently (directory is same as folder for Windows OS users).

In the output, we are in harssh directory(folder for Windows OS that are moving to Linux),which is present inside the home directory.

```
 $ pwd

Output: /home/harssh
```

3. mkdir : The ‘$ mkdir’ stands for ‘make directory’ and it creates a new directory.We have used ‘$ cd’ (which is discussed below) to get into the newly created directory and again on giving ‘$ pwd’ command,we are displayed with the new ‘newfolder’ directory.

```
$ mkdir newfolder

$ cd newfolder

$ pwd

Output: /home/harssh/newfolder
```

4. rmdir : The ‘$ rmdir’ command deletes any directory we want to delete and you can remember it by its names ‘rmdir’ which stands for ‘remove directory’.

```
$ rmdir newfolder
```

5. cd : The ‘$ cd’ command stands for ‘change directory’ and it changes your current directory to the ‘newfolder’ directory.You can understand this a double-clicking a folder and then you do some stuff in that folder.

```
$ cd newfolder (assuming that there is a directory named 'newfolder' on
your system)
```

6. ls : The ‘ls’ command simply displays the contents of a directory.

```
$ ls

Output: Desktop Documents Downloads Music Pictures Public Scratch Templates
Videos
```

7. touch : The ‘$ touch’ command creates a file(not directory) and you can simple add an extension such as .txt after it to make it a Text File.

```
$ touch example

$ ls

Output: Desktop Documents Downloads Music Pictures Public Scratch Templates
Videos example
```

Note: *It is important to note that according to the Unix File structure, Unix treats all the stuff it has as a ‘file’, even the directories(folders) are also treated as a file.You will get to know*

more about this as you will further use Linux/Unix based OS

8. cp : This ‘$ cp ‘ command stands for ‘copy’ and it simply copy/paste the file wherever you want to. In the above example, we are copying a file ‘file.txt’ from the directory harssh to a new directory new.

```
$ cp /home/harssh/file.txt /home/harssh/new/
```

9. mv : The ‘$ mv’ command stands for ‘move’ and it simply move a file from a directory to another directory.In the above example a file named ‘file.txt’ is being moved into a new directory ‘new’

```
$ mv /home/harssh/file.txt /home/harssh/new
```

10. rm : The ‘$ rm ‘ command for remove and the ‘-r’ simply recursively deletes file. Try ‘$

rm filename.txt’ at your terminal

```
$ rm file.txt
```

11. chmod : The ‘$ chmod’ command stands for change mode command. As there are many modes in Unix that can be used to manipulate files in the Unix environment. Basically there are 3 modes that we can use with the ‘chmod’ command

1. +w (stands for write and it changes file permissions to write)
2. +r (stands for read and it changes file permissions to read)
3. +x (generally it is used to make a file executable)

```
$ chmod +w file.txt
$ chmod +r file.txt

$ chmod +x file.txt
```

12. cal : The '$ cal' means calendar and it simply display calendar on to your screen.

```
$ cal

Output : July 2017

Su Mo Tu We Th Fr Sa

1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

30 31
```

13. file : The '$ file' command displays the type of file. As I mentioned earlier Linux treats everything as a file so on executing the command file on a directory(Downloads) it displays directory as the output

```
$ ls

Output: Desktop Documents Downloads Music Pictures Public Scratch Templates
Videos

$ file Downloads

Output: Downloads: directory
```

14. sort : As the name suggests the '$ sort' sorts the contents of the file according to the ASCII rules.

```
$ sort file
```

15. grep : grep is an acronym for 'globally search a regular expression and print it'.The '$ grep' command searches the specified input fully(globally) for a match with the supplied pattern and displays it.

In the example, this would search for the word 'picture' in the file newsfile and if found,the lines containing it would be displayed on the screen.

```
$ grep picture newsfile
```

16. man : The '$ man' command stands for 'manual' and it can display the in-built manual for most of the commands that we ever need. In the above example, we can read about the '$ pwd' command.

```
$ man pwd
```

17. lpr : The '$ lpr' command send a file to the printer for printing.

```
$ lpr new.txt
```

18. passwd : The '$ passwd' command simply changes the password of the user.In above case 'harssh' is the user.

```
$ passwd

Output: Changing password for harssh.

(current) UNIX password:
```

19. clear : The '$ clear' command is used to clean up the terminal so that you can type with

more accuracy

```
$ clear
```

20. history : The '$ history' command is used to get list of previous commands may be obtained by executing the following command. you can also use parameters like **!n** to re-execute the nth command, **!!** to executes the most recent command, and **!cp** this will execute the most recent command that starts with cp.

```
$ history
```

At last, I want to say that these are the most basic and essential commands that are used in the Linux operating system. You will need them even if you get to advance the Unix. If you want to master them just keep on practicing them.

Also, it is not possible to cover all the Unix commands because they are so many in number. You can find more, just google it and you will find most of them. Also if you want to master Unix operating system, Learn Unix Shell Scripting/Bash Scripting. Trust me there are many awesome tutorials on the internet for them.

Compile your C/C++ program in the Unix terminal

First reach the directory where your .c or .cpp file is present (assume its name is new.c or new.cpp). Please note that in order to compile C you will need GCC compiler and in order to compile C++ you will need g++. I will tell you below how to install them.

For C:

```
$ gcc new.c -o new

$ ./new
```

For C++:

```
$ g++ new.cpp -o new

$ ./new
```

In this way you can compile you programs of C and C++. You can even compile many more that i will cover in my coming articles.

Getting gcc, g++

For deb based Linux such as Ubuntu,Debian,Kali etc:

```
$ sudo apt-get install gcc

$ sudo apt-get install g++
```

For rpm based Linux such as Fedora,Oracle Linux etc:

```
$ dnf install gcc

$ dnf install g++
 OR
$ yum install gcc
$ yum install g++
```

Uses of Unix Commands :

1. **File and directory management:** Unix commands like ls, cd, cp, mv, rm, mkdir, and rmdir are used for managing files and directories. These commands allow users to create, delete, copy, move, and rename files and directories.
2. **Process management:** Unix commands like ps and kill are used for managing running processes. Users can use these commands to list all running processes, view process details, and terminate a process if necessary.
3. **User management:** Unix commands like passwd, useradd, userdel, and groupadd are used for managing user accounts and groups. These commands allow system administrators to add or delete user accounts, change user passwords, and manage group memberships.
4. **Text manipulation:** Unix commands like cat, grep, sed, and awk are used for manipulating text files. These commands allow users to view, search, replace, and format text files.
5. **Networking:** Unix commands like ping, ifconfig, netstat, and traceroute are used for configuring and troubleshooting network connections. These commands allow users to test network connectivity, view network interface details, and diagnose network problems.
6. **System configuration:** Unix commands like chmod, chown, and sysctl are used for configuring system settings. These commands allow users to change file permissions, file ownership, and system parameters.
7. **Programming:** Unix commands like gcc, make, and gdb are used for compiling and debugging programs. These commands are essential for developers who write programs in C or C++.

Issues in Unix Commands :

1. **Command syntax errors:** Unix commands are sensitive to syntax errors, which can lead to unexpected behavior or errors. Users must be careful to use the correct command syntax and to include all required parameters.
2. **Permissions issues:** Unix commands are subject to file permissions, which can restrict access to certain files or directories. Users may need to use the chmod or chown commands to change file permissions or file ownership to access certain files.
3. **Security risks:** Some Unix commands can be used to compromise system security, such as allowing unauthorized access to files or executing malicious code. Users must be careful to use Unix commands only for authorized tasks and to avoid running unknown scripts or commands.
4. **Lack of user interface:** Unix commands are typically executed from the command line, which can be intimidating for users who are used to graphical user interfaces. Users may need to spend time learning the syntax and usage of Unix commands to be effective.
5. **Limited documentation:** While Unix commands are well-documented, some commands may not have complete or clear documentation. Users may need to rely on online resources or community forums to find answers to specific issues.

Linux Directory Commands

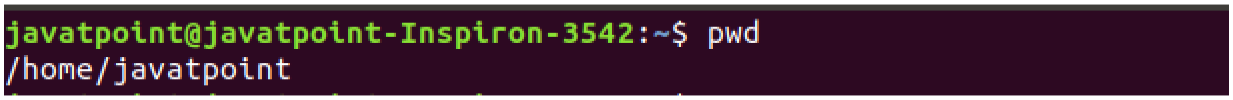
1. pwd Command

The pwd command is used to display the location of the current working directory.

Syntax:

1. pwd

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ pwd
/home/javatpoint
```

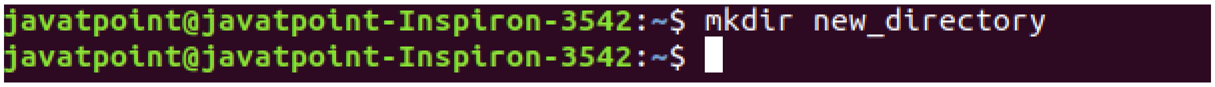
2. mkdir Command

The mkdir command is used to create a new directory under any directory.

Syntax:

1. mkdir <**directory** name>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ mkdir new_directory
javatpoint@javatpoint-Inspiron-3542:~$ 
```

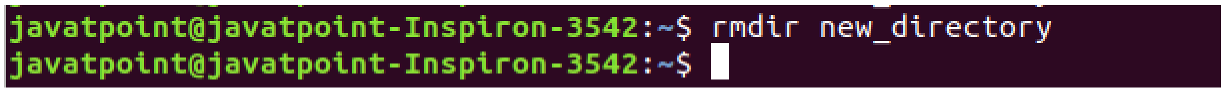
3. rmdir Command

The rmdir command is used to delete a directory.

Syntax:

1. rmdir <**directory** name>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ rmdir new_directory
javatpoint@javatpoint-Inspiron-3542:~$ 
```

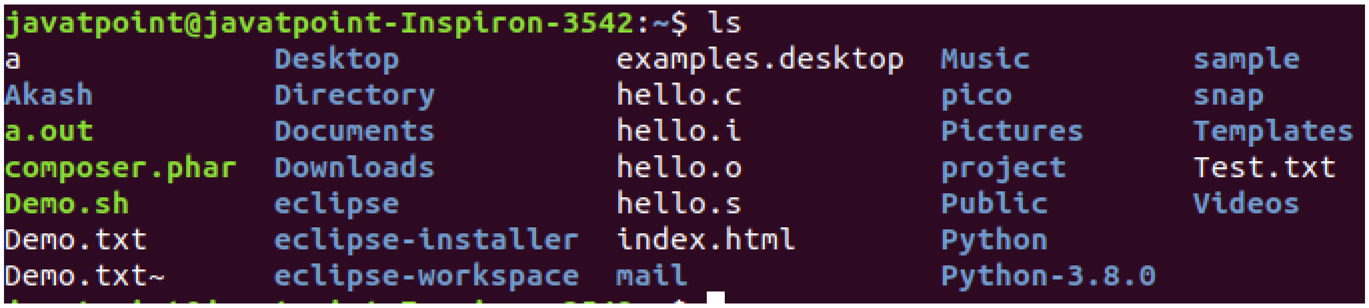
4. ls Command

The ls command is used to display a list of content of a directory.

Syntax:

1. ls

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ ls
a                Desktop            examples.desktop  Music           sample
Akash            Directory          hello.c           pico            snap
a.out            Documents          hello.i           Pictures        Templates
composer.phar    Downloads          hello.o           project         Test.txt
Demo.sh          eclipse            hello.s           Public          Videos
Demo.txt         eclipse-installer  index.html        Python
Demo.txt~        eclipse-workspace  mail              Python-3.8.0
```

5. cd Command

ADVERTISEMENT

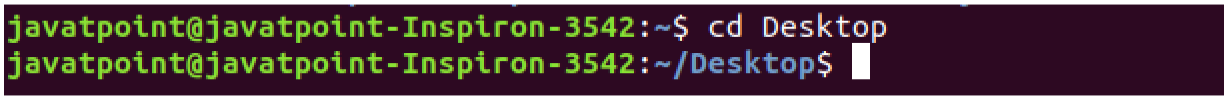
ADVERTISEMENT

The cd command is used to change the current directory.

Syntax:

1. cd <**directory** name>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ cd Desktop
javatpoint@javatpoint-Inspiron-3542:~/Desktop$
```

ADVERTISEMENT

Linux File commands

6. touch Command

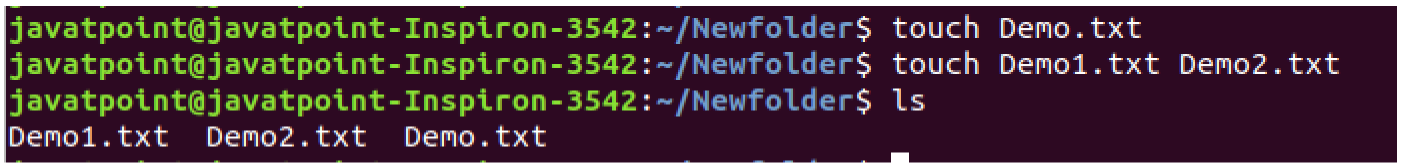
The touch command is used to create empty files. We can create multiple empty files by executing it once.

Syntax:

1. touch <file name>
2. touch <file1> <file2> ....

ADVERTISEMENT

Output:



```
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ touch Demo.txt
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ touch Demo1.txt Demo2.txt
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ ls
Demo1.txt  Demo2.txt  Demo.txt
```

7. cat Command

The cat command is a multi-purpose utility in the Linux system. It can be used to create a file, display content of the file, copy the content of one file to another file, and more.

Syntax:

1. cat [OPTION]... [FILE]..

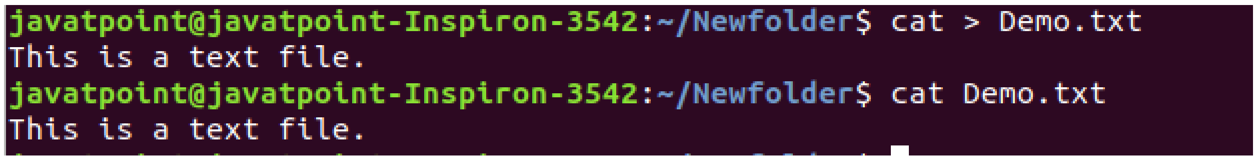
To create a file, execute it as follows:

1. cat > <file name>
2. // Enter file content

Press "**CTRL+ D**" keys to save the file. To display the content of the file, execute it as follows:

1. cat <file name>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ cat > Demo.txt
This is a text file.
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ cat Demo.txt
This is a text file.
```

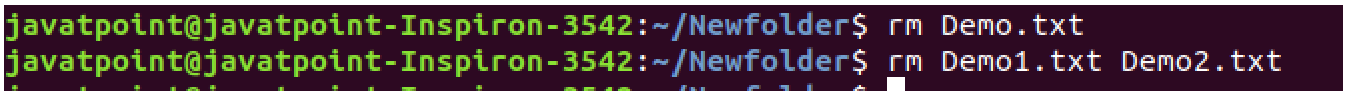
8. rm Command

The rm command is used to remove a file.

Syntax:

rm <file name>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ rm Demo.txt
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ rm Demo1.txt Demo2.txt
```

9. cp Command

The cp command is used to copy a file or directory.

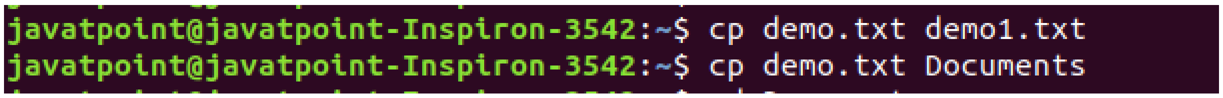
Syntax:

To copy in the same directory:

1. cp **<existing** file name**>** **<new** file name**>**

To copy in a different directory:

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ cp demo.txt demo1.txt
javatpoint@javatpoint-Inspiron-3542:~$ cp demo.txt Documents
```

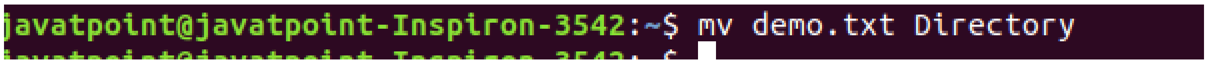
10. mv Command

The mv command is used to move a file or a directory form one location to another location.

Syntax:

1. mv <file name> <directory path>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ mv demo.txt Directory
```

11. rename Command

The rename command is used to rename files. It is useful for renaming a large group of files.

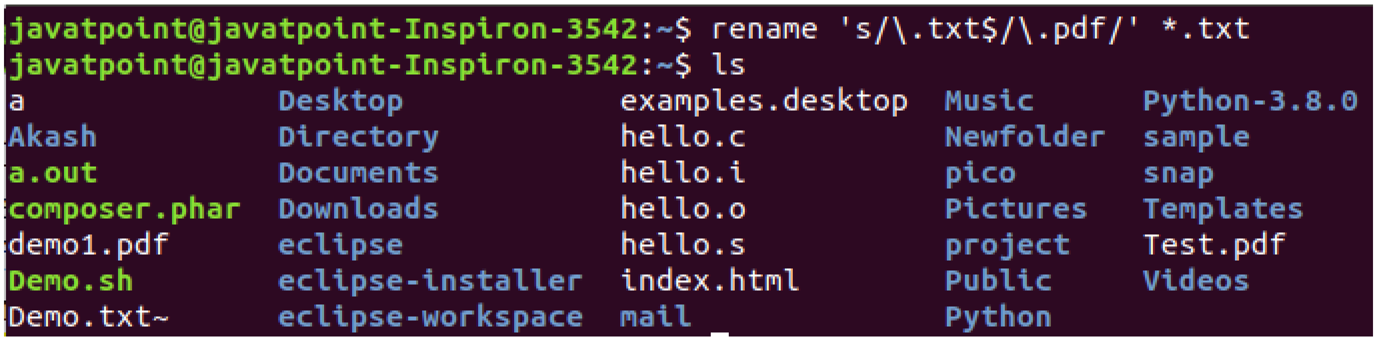
Syntax:

1. rename 's/old-name/new-name/' files

For example, to convert all the text files into pdf files, execute the below command:

1. rename 's/\.txt$/\.pdf/' *.txt

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ rename 's/\.txt$/\.pdf/' *.txt
javatpoint@javatpoint-Inspiron-3542:~$ ls
a              Desktop              examples.desktop  Music      Python-3.8.0
Akash          Directory            hello.c           Newfolder  sample
a.out          Documents            hello.i           pico       snap
composer.phar  Downloads            hello.o           Pictures   Templates
demo1.pdf      eclipse              hello.s           project    Test.pdf
Demo.sh        eclipse-installer    index.html        Public     Videos
Demo.txt~      eclipse-workspace    mail              Python
```

Linux File Content Commands

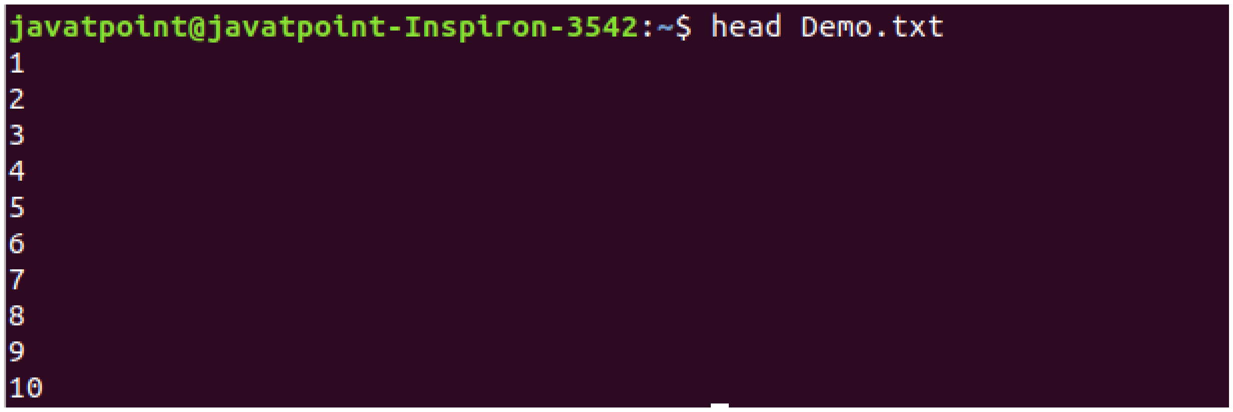
12. head Command

The head command is used to display the content of a file. It displays the first 10 lines of a file.

Syntax:

1. head <file name>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ head Demo.txt
1
2
3
4
5
6
7
8
9
10
```

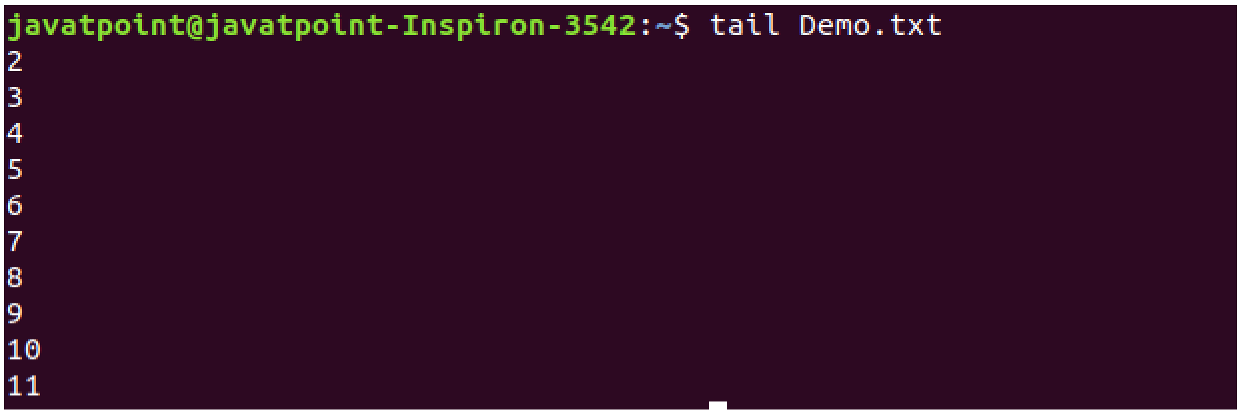
13. tail Command

The tail command is similar to the head command. The difference between both commands is that it displays the last ten lines of the file content. It is useful for reading the error message.

Syntax:

1. tail <file name>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ tail Demo.txt
2
3
4
5
6
7
8
9
10
11
```

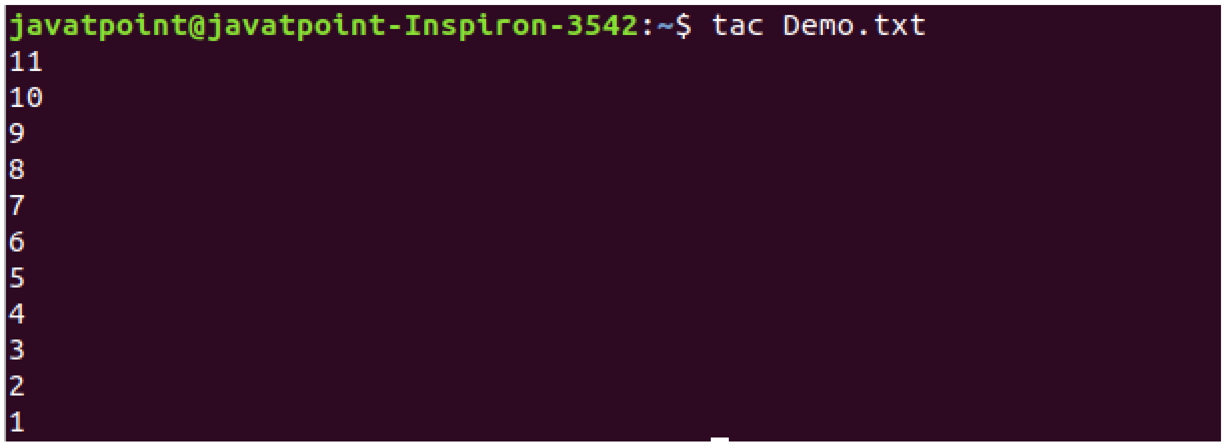
14. tac Command

The tac command is the reverse of cat command, as its name specified. It displays the file content in reverse order (from the last line).

Syntax:

1. tac <file name>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ tac Demo.txt
11
10
9
8
7
6
5
4
3
2
1
```

15. more command

The more command is quite similar to the cat command, as it is used to display the file content in the same way that the cat command does. The only difference between both commands is that, in case of larger files, the more command displays screenful output at a time.

In more command, the following keys are used to scroll the page:

ENTER key: To scroll down page by line.

Space bar: To move to the next page.

ADVERTISEMENT

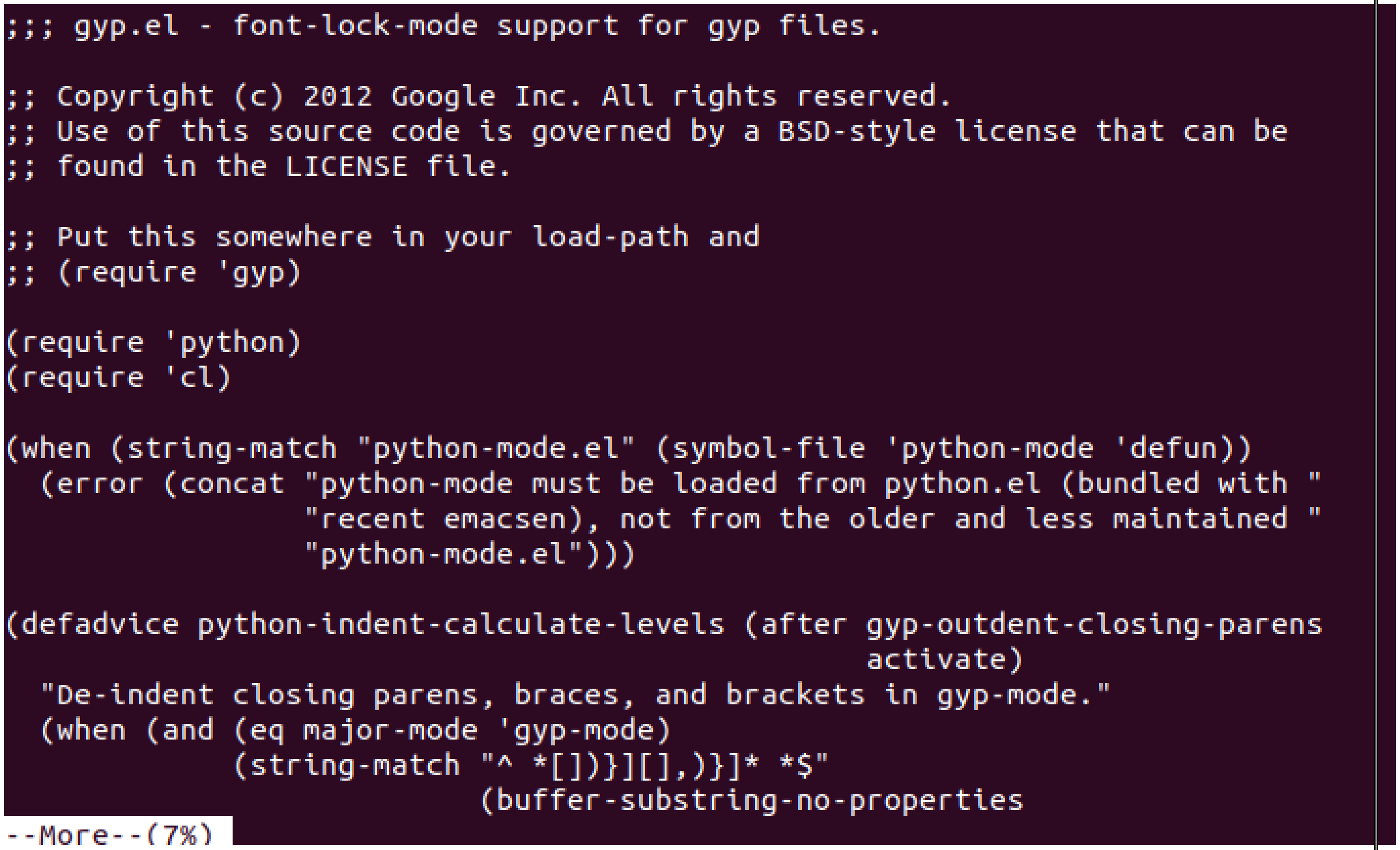
b key: To move to the previous page.

/ key: To search the string.

Syntax:

```
1. more <file name>
```

Output:



```
;;; gyp.el - font-lock-mode support for gyp files.

;; Copyright (c) 2012 Google Inc. All rights reserved.
;; Use of this source code is governed by a BSD-style license that can be
;; found in the LICENSE file.

;; Put this somewhere in your load-path and
;; (require 'gyp)

(require 'python)
(require 'cl)

(when (string-match "python-mode.el" (symbol-file 'python-mode 'defun))
  (error (concat "python-mode must be loaded from python.el (bundled with "
                 "recent emacsen), not from the older and less maintained "
                 "python-mode.el")))

(defadvice python-indent-calculate-levels (after gyp-outdent-closing-parens
                                                  activate)
  "De-indent closing parens, braces, and brackets in gyp-mode."
  (when (and (eq major-mode 'gyp-mode)
             (string-match "^ *[])}][],)}]* *$"
                           (buffer-substring-no-properties
--More--(7%)
```

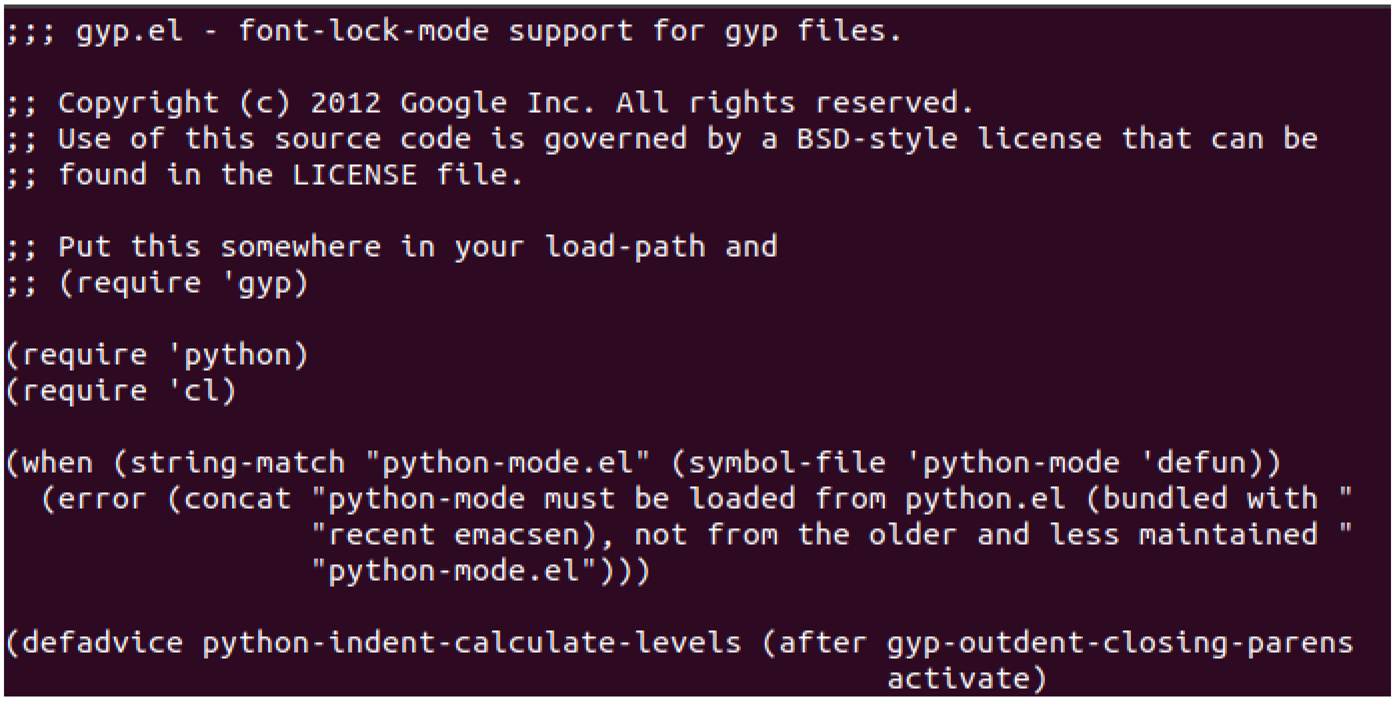
16. less Command

The less command is similar to the more command. It also includes some extra features such as 'adjustment in width and height of the terminal.' Comparatively, the more command cuts the output in the width of the terminal.

Syntax:

1. less <file name>

Output:



```
;;; gyp.el - font-lock-mode support for gyp files.

;; Copyright (c) 2012 Google Inc. All rights reserved.
;; Use of this source code is governed by a BSD-style license that can be
;; found in the LICENSE file.

;; Put this somewhere in your load-path and
;; (require 'gyp)

(require 'python)
(require 'cl)

(when (string-match "python-mode.el" (symbol-file 'python-mode 'defun))
  (error (concat "python-mode must be loaded from python.el (bundled with "
                 "recent emacsen), not from the older and less maintained "
                 "python-mode.el")))

(defadvice python-indent-calculate-levels (after gyp-outdent-closing-parens
                                                 activate)
```

Linux User Commands

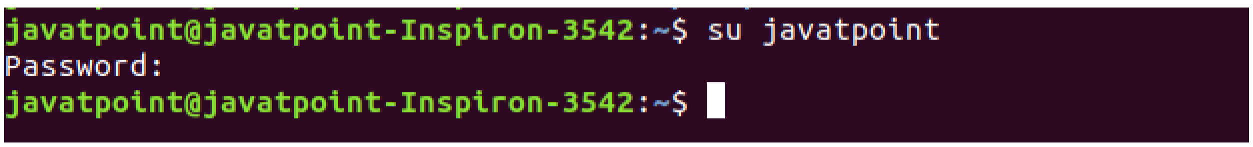
17. su Command

The su command provides administrative access to another user. In other words, it allows access of the Linux shell to another user.

Syntax:

1. su <**user** name>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ su javatpoint
Password:
javatpoint@javatpoint-Inspiron-3542:~$
```

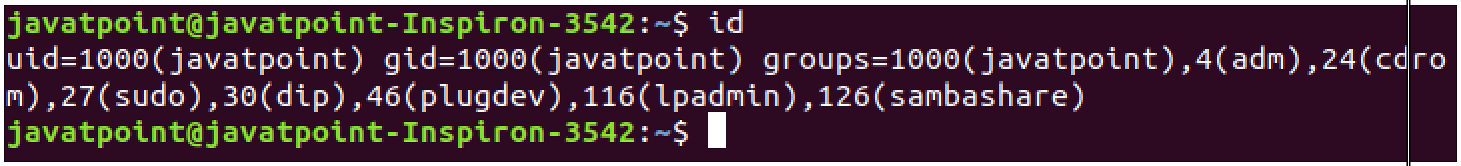
18. id Command

The id command is used to display the user ID (UID) and group ID (GID).

Syntax:

1. id

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ id
uid=1000(javatpoint) gid=1000(javatpoint) groups=1000(javatpoint),4(adm),24(cdro
m),27(sudo),30(dip),46(plugdev),116(lpadmin),126(sambashare)
javatpoint@javatpoint-Inspiron-3542:~$ 
```

19. useradd Command

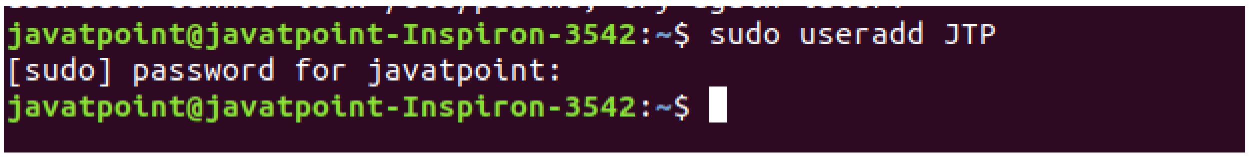
The useradd command is used to add or remove a user on a Linux server.

ADVERTISEMENT

Syntax:

1. useradd username

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ sudo useradd JTP
[sudo] password for javatpoint:
javatpoint@javatpoint-Inspiron-3542:~$ 
```

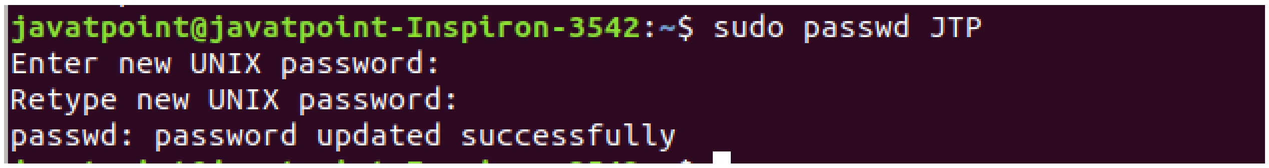
20. passwd Command

The passwd command is used to create and change the password for a user.

Syntax:

1. passwd **<username>**

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ sudo passwd JTP
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

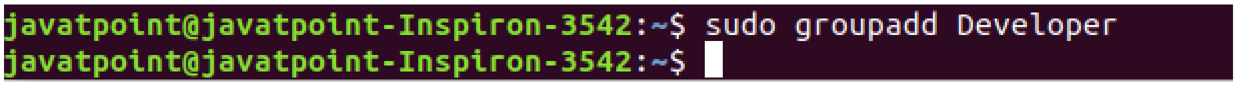
21. groupadd Command

The groupadd command is used to create a user group.

Syntax:

1. groupadd <group name>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ sudo groupadd Developer
javatpoint@javatpoint-Inspiron-3542:~$
```

Linux Filter Commands

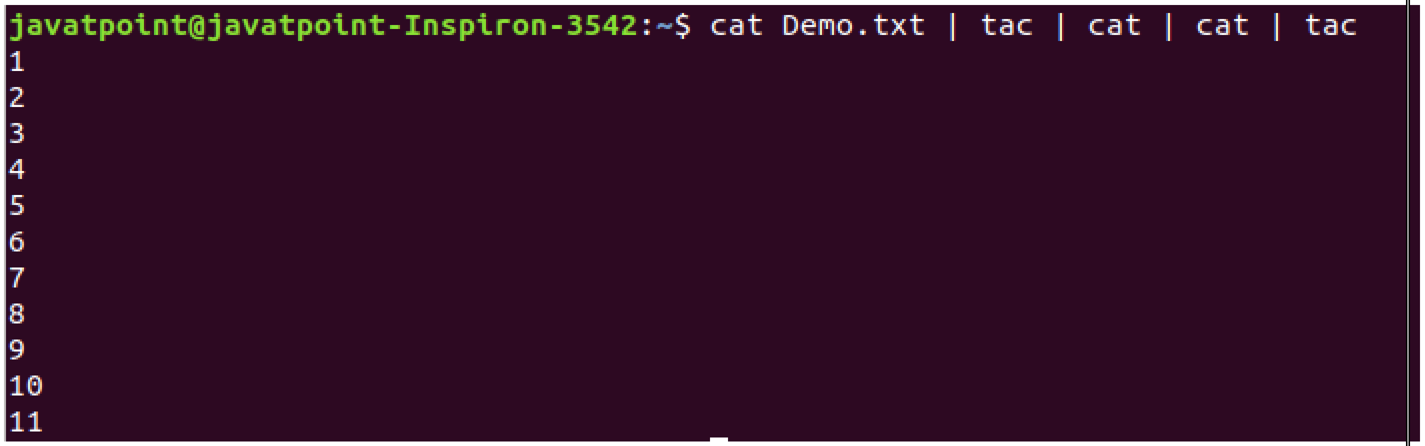
22. cat Command

The cat command is also used as a filter. To filter a file, it is used inside pipes.

Syntax:

1. cat <fileName> | cat or tac | cat or tac |. . .

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ cat Demo.txt | tac | cat | cat | tac
1
2
3
4
5
6
7
8
9
10
11
```

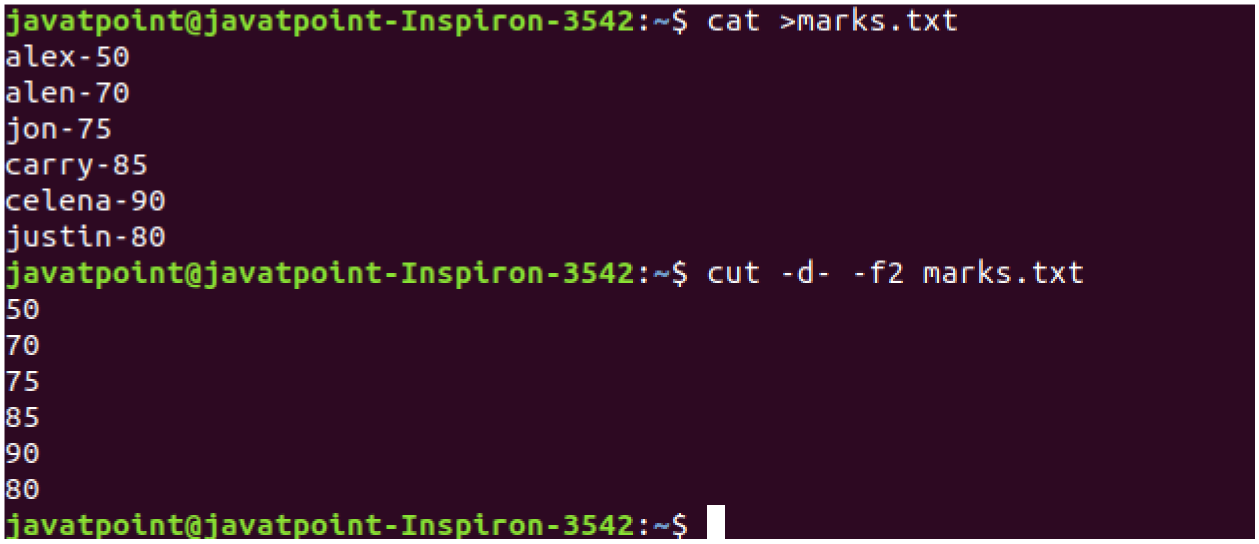
23. cut Command

The cut command is used to select a specific column of a file. The '-d' option is used as a delimiter, and it can be a space (' '), a slash (/), a hyphen (-), or anything else. And, the '-f' option is used to specify a column number.

Syntax:

1. cut -d(delimiter) -f(columnNumber) **<fileName>**

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ cat >marks.txt
alex-50
alen-70
jon-75
carry-85
celena-90
justin-80
javatpoint@javatpoint-Inspiron-3542:~$ cut -d- -f2 marks.txt
50
70
75
85
90
80
javatpoint@javatpoint-Inspiron-3542:~$
```

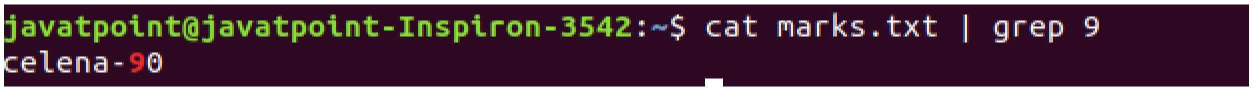
24. grep Command

The grep is the most powerful and used filter in a Linux system. The 'grep' stands for "**global regular expression print**." It is useful for searching the content from a file. Generally, it is used with the pipe.

Syntax:

1. command | grep **<searchWord>**

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ cat marks.txt | grep 9
celena-90
```

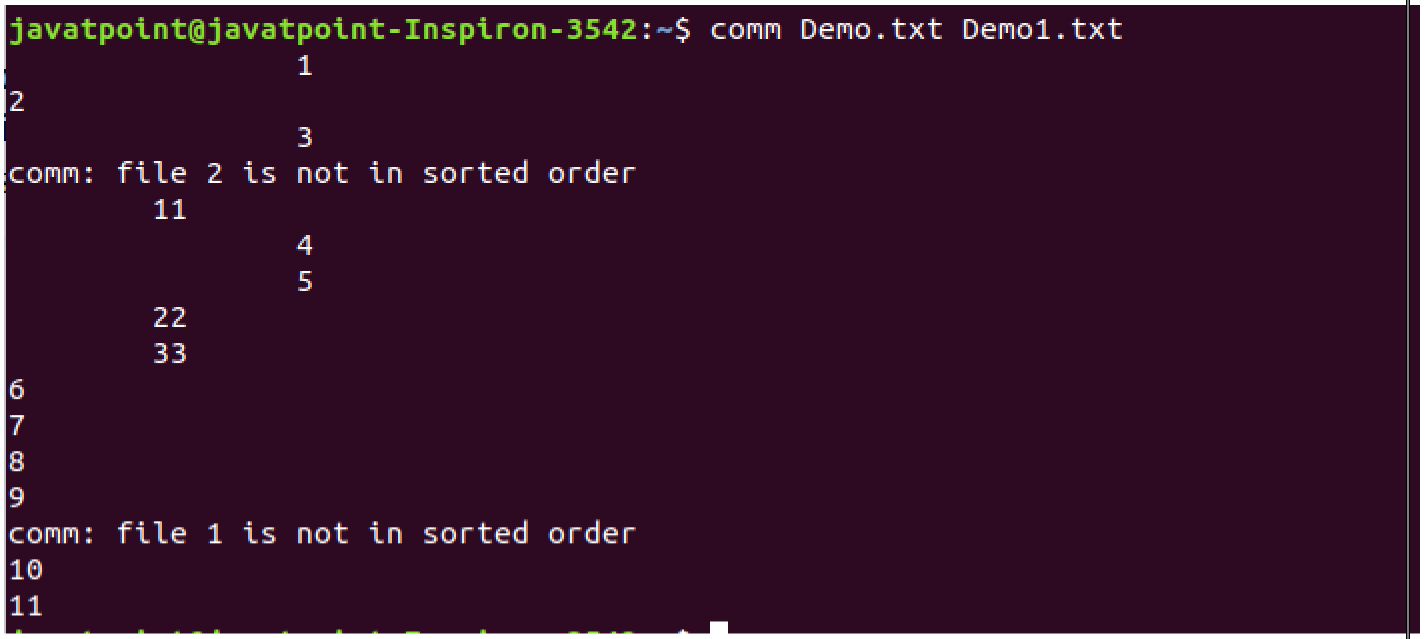
25. comm Command

The 'comm' command is used to compare two files or streams. By default, it displays three columns, first displays non-matching items of the first file, second indicates the non-matching item of the second file, and the third column displays the matching items of both files.

Syntax:

1. comm **<file1> <file2>**

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ comm Demo.txt Demo1.txt
                1
2
                3
comm: file 2 is not in sorted order
        11
                4
                5
        22
        33
6
7
8
9
comm: file 1 is not in sorted order
10
11
```

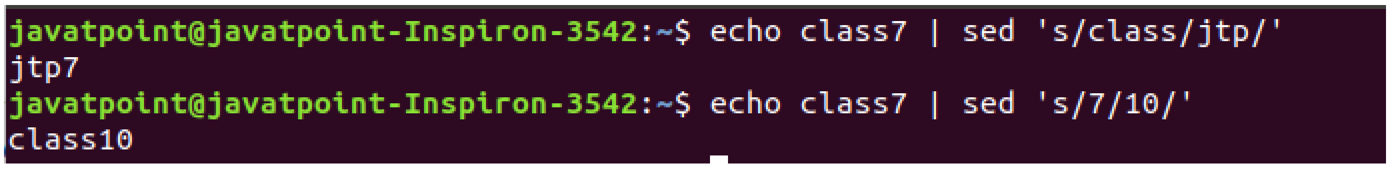
26. sed command

The sed command is also known as **stream editor**. It is used to edit files using a regular expression. It does not permanently edit files; instead, the edited content remains only on display. It does not affect the actual file.

Syntax:

1. command | sed 's/**<oldWord>**/**<newWord>**/'

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ echo class7 | sed 's/class/jtp/'
jtp7
javatpoint@javatpoint-Inspiron-3542:~$ echo class7 | sed 's/7/10/'
class10
```

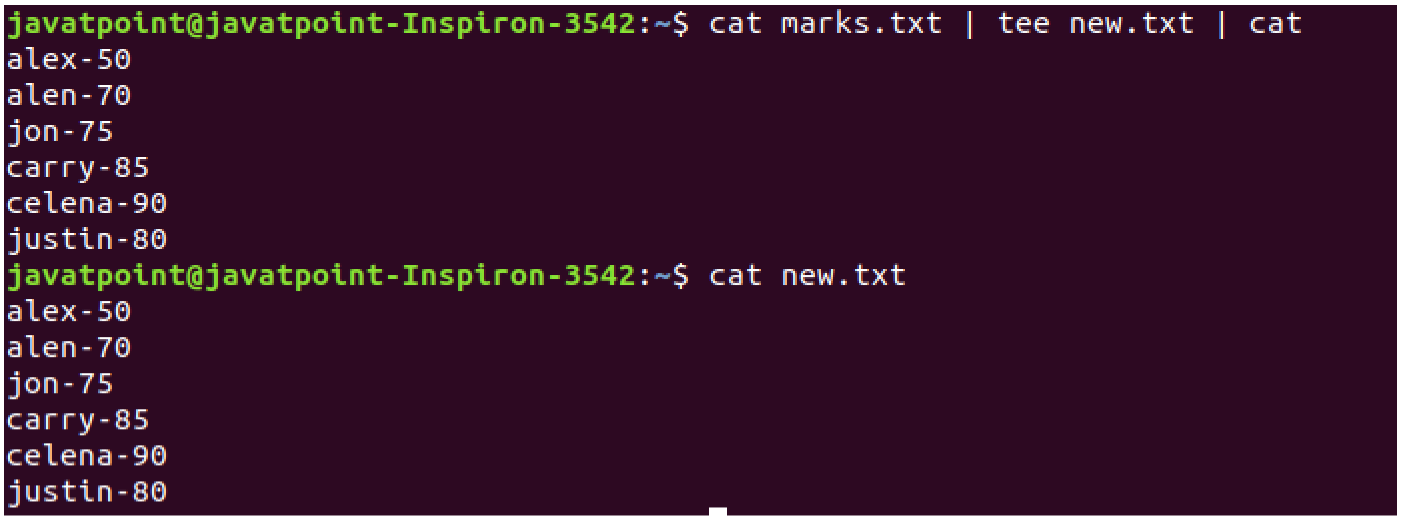
27. tee command

The tee command is quite similar to the cat command. The only difference between both filters is that it puts standard input on standard output and also write them into a file.

Syntax:

1. cat **<fileName>** | tee **<newFile>** | cat or tac |.....

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ cat marks.txt | tee new.txt | cat
alex-50
alen-70
jon-75
carry-85
celena-90
justin-80
javatpoint@javatpoint-Inspiron-3542:~$ cat new.txt
alex-50
alen-70
jon-75
carry-85
celena-90
justin-80
```

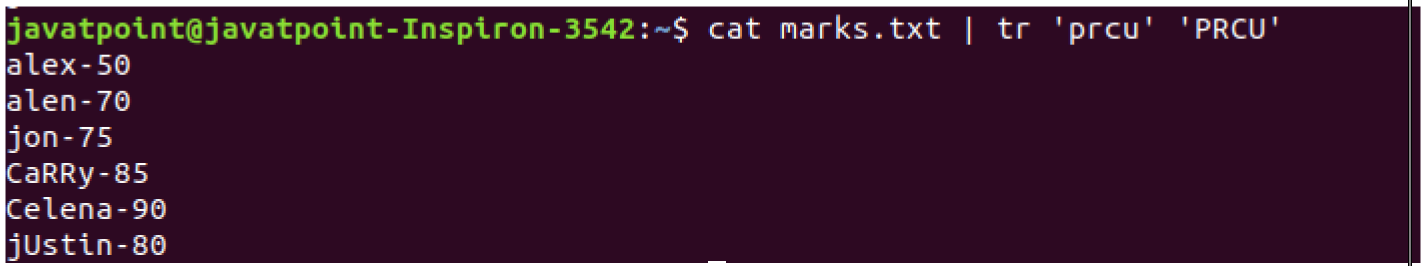
28. tr Command

The tr command is used to translate the file content like from lower case to upper case.

Syntax:

1. command | tr <'old'> <'new'>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ cat marks.txt | tr 'prcu' 'PRCU'
alex-50
alen-70
jon-75
CaRRy-85
Celena-90
jUstin-80
```

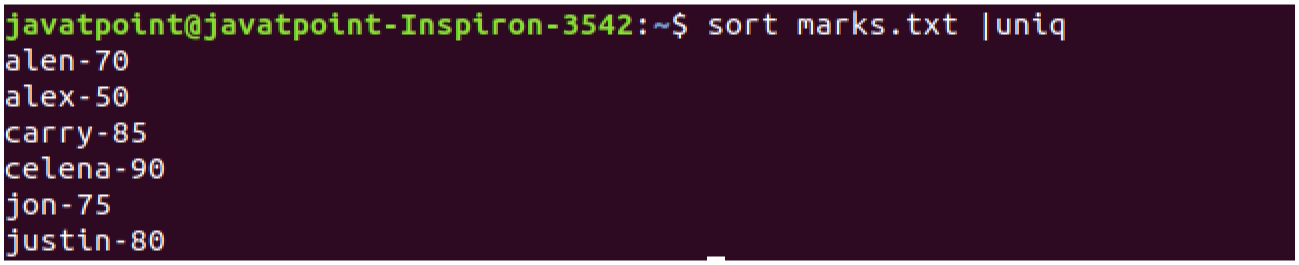
29. uniq Command

The uniq command is used to form a sorted list in which every word will occur only once.

Syntax:

1. command **<fileName>** | uniq

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ sort marks.txt |uniq
alen-70
alex-50
carry-85
celena-90
jon-75
justin-80
```

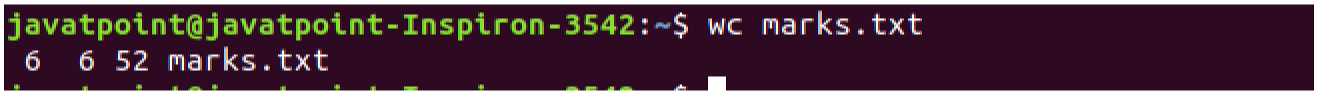
30. wc Command

The wc command is used to count the lines, words, and characters in a file.

Syntax:

1. wc **<file** name**>**

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ wc marks.txt
 6  6 52 marks.txt
```

31. od Command

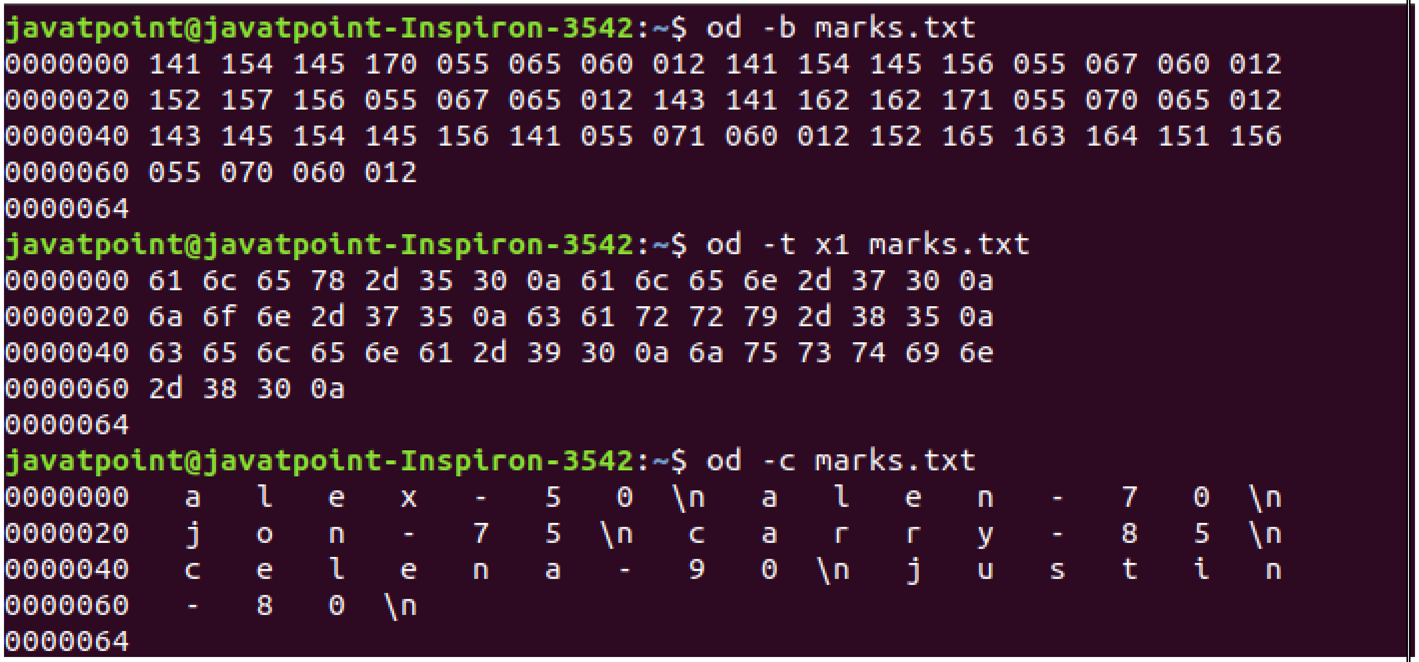
The od command is used to display the content of a file in different s, such as hexadecimal, octal, and ASCII characters.

Syntax:

1. od -b **<fileName>** // Octal format

2. od -t x1 **<fileName>** // Hexa decimal format
3. od -c **<fileName>** // ASCII character format

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ od -b marks.txt
0000000 141 154 145 170 055 065 060 012 141 154 145 156 055 067 060 012
0000020 152 157 156 055 067 065 012 143 141 162 162 171 055 070 065 012
0000040 143 145 154 145 156 141 055 071 060 012 152 165 163 164 151 156
0000060 055 070 060 012
0000064
javatpoint@javatpoint-Inspiron-3542:~$ od -t x1 marks.txt
0000000 61 6c 65 78 2d 35 30 0a 61 6c 65 6e 2d 37 30 0a
0000020 6a 6f 6e 2d 37 35 0a 63 61 72 72 79 2d 38 35 0a
0000040 63 65 6c 65 6e 61 2d 39 30 0a 6a 75 73 74 69 6e
0000060 2d 38 30 0a
0000064
javatpoint@javatpoint-Inspiron-3542:~$ od -c marks.txt
0000000   a   l   e   x   -   5   0  \n   a   l   e   n   -   7   0  \n
0000020   j   o   n   -   7   5  \n   c   a   r   r   y   -   8   5  \n
0000040   c   e   l   e   n   a   -   9   0  \n   j   u   s   t   i   n
0000060   -   8   0  \n
0000064
```

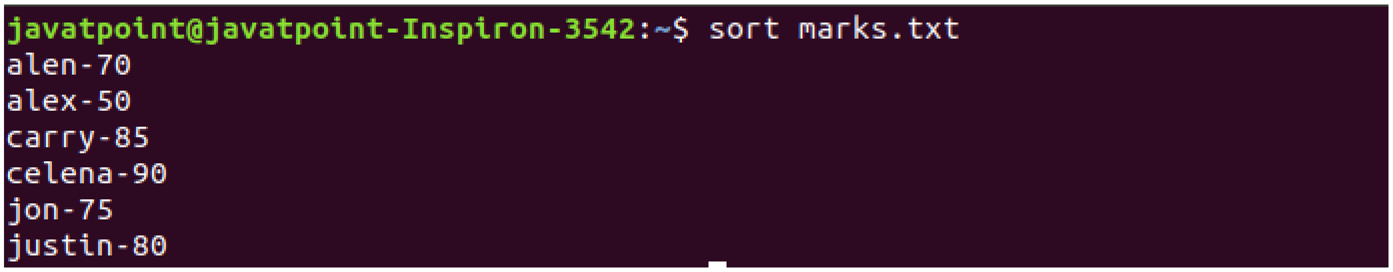
32. sort Command

The sort command is used to sort files in alphabetical order.

Syntax:

1. sort **<file** name**>**

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ sort marks.txt
alen-70
alex-50
carry-85
celena-90
jon-75
justin-80
```

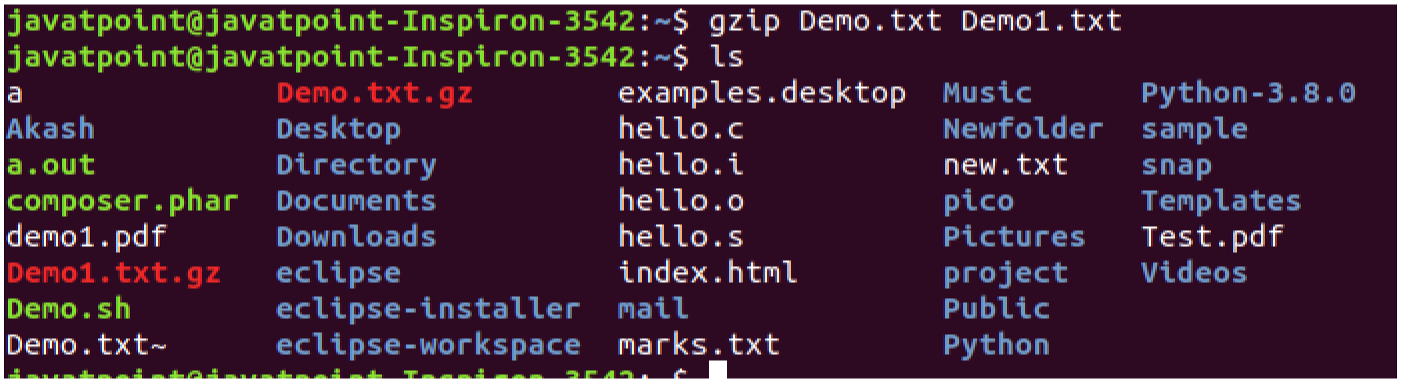
33. gzip Command

The gzip command is used to truncate the file size. It is a compressing tool. It replaces the origina file by the compressed file having '.gz' extension.

Syntax:

1. gzip **<file1> <file2> <file3>**...

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ gzip Demo.txt Demo1.txt
javatpoint@javatpoint-Inspiron-3542:~$ ls
a              Demo.txt.gz        examples.desktop  Music      Python-3.8.0
Akash          Desktop            hello.c           Newfolder  sample
a.out          Directory          hello.i           new.txt    snap
composer.phar  Documents          hello.o           pico       Templates
demo1.pdf      Downloads          hello.s           Pictures   Test.pdf
Demo1.txt.gz   eclipse            index.html        project    Videos
Demo.sh        eclipse-installer  mail              Public
Demo.txt~      eclipse-workspace  marks.txt         Python
```

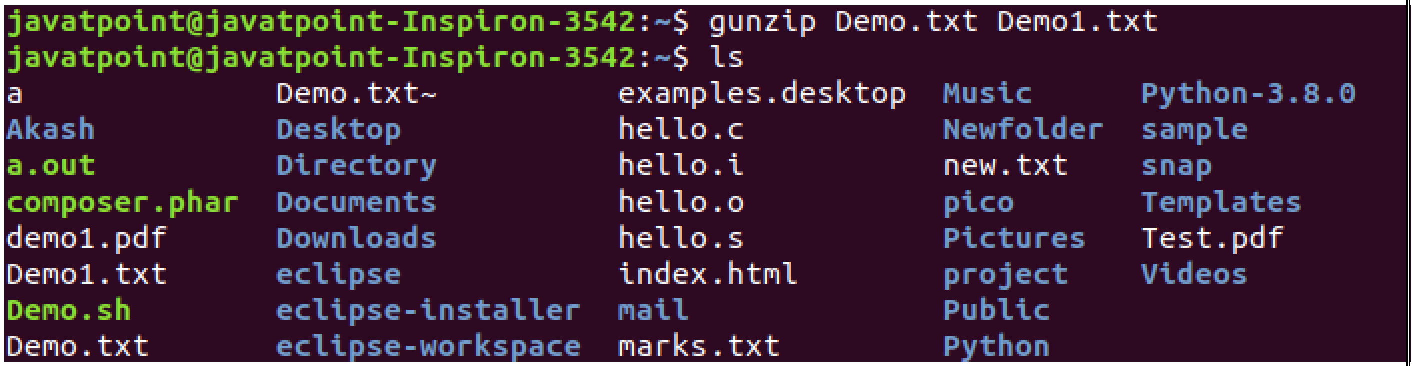
34. gunzip Command

The gunzip command is used to decompress a file. It is a reverse operation of gzip command.

Syntax:

1. gunzip **<file1> <file2> <file3>**. .

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ gunzip Demo.txt Demo1.txt
javatpoint@javatpoint-Inspiron-3542:~$ ls
a              Demo.txt~            examples.desktop  Music      Python-3.8.0
Akash          Desktop              hello.c           Newfolder  sample
a.out          Directory            hello.i           new.txt    snap
composer.phar  Documents            hello.o           pico       Templates
demo1.pdf      Downloads            hello.s           Pictures   Test.pdf
Demo1.txt      eclipse              index.html        project    Videos
Demo.sh        eclipse-installer    mail              Public
Demo.txt       eclipse-workspace    marks.txt         Python
```

Linux Utility Commands

35. find Command

The find command is used to find a particular file within a directory. It also supports various options to find a file such as byname, by type, by date, and more.

The following symbols are used after the find command:

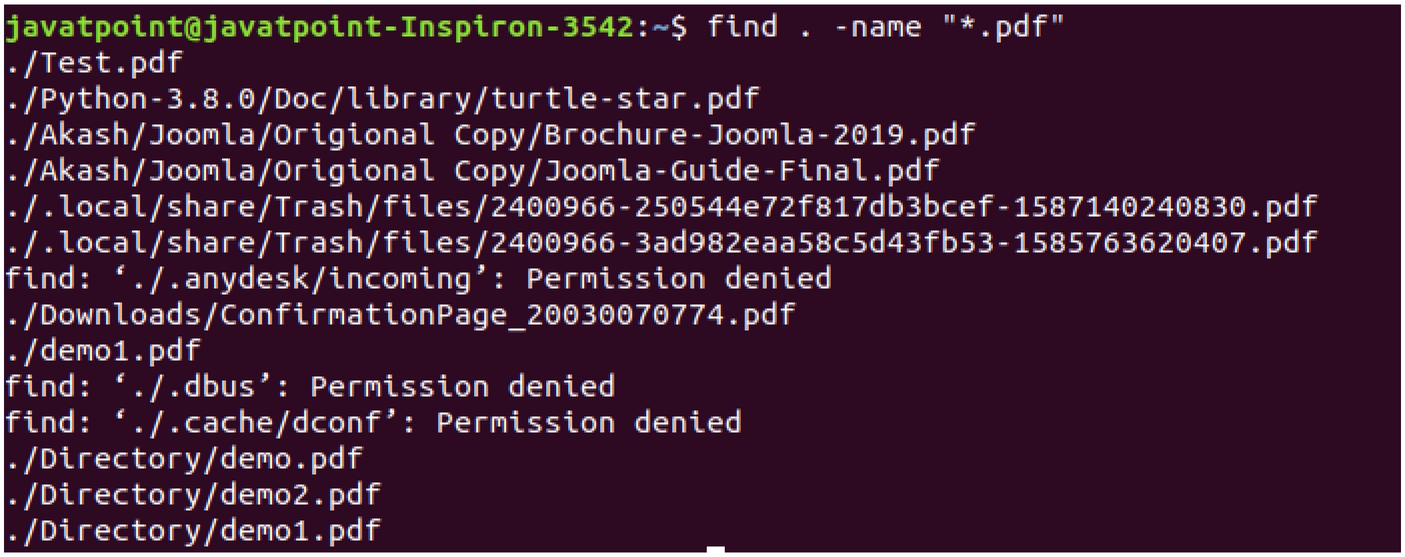
(.) : For current directory name

(/) : For root

Syntax:

1. find . -name "*.pdf"

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ find . -name "*.pdf"
./Test.pdf
./Python-3.8.0/Doc/library/turtle-star.pdf
./Akash/Joomla/Origional Copy/Brochure-Joomla-2019.pdf
./Akash/Joomla/Origional Copy/Joomla-Guide-Final.pdf
./.local/share/Trash/files/2400966-250544e72f817db3bcef-1587140240830.pdf
./.local/share/Trash/files/2400966-3ad982eaa58c5d43fb53-1585763620407.pdf
find: './.anydesk/incoming': Permission denied
./Downloads/ConfirmationPage_20030070774.pdf
./demo1.pdf
find: './.dbus': Permission denied
find: './.cache/dconf': Permission denied
./Directory/demo.pdf
./Directory/demo2.pdf
./Directory/demo1.pdf
```

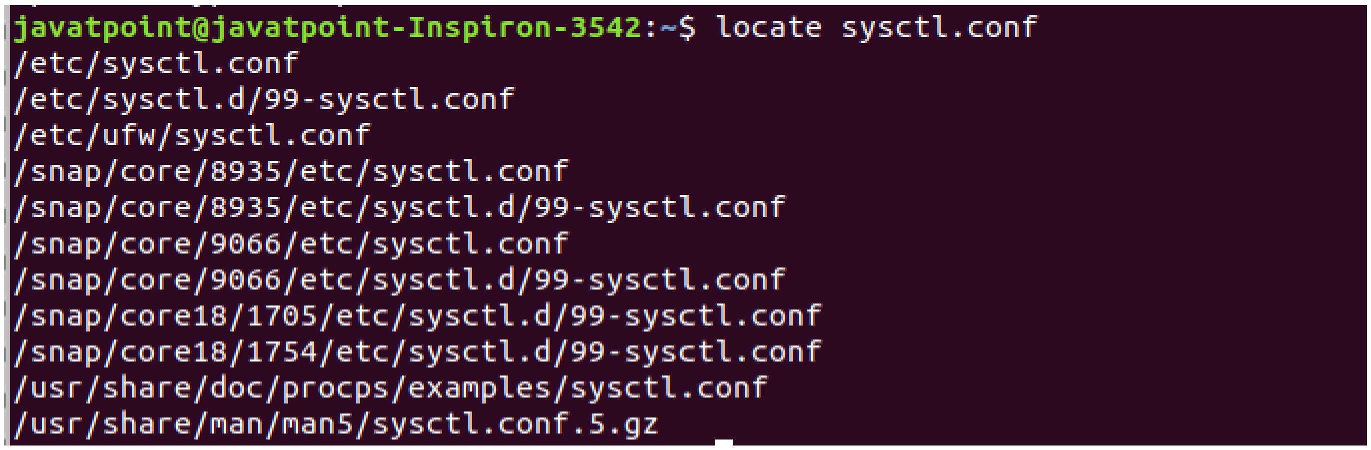
36. locate Command

The locate command is used to search a file by file name. It is quite similar to find command; the difference is that it is a background process. It searches the file in the database, whereas the find command searches in the file system. It is faster than the find command. To find the file with the locates command, keep your database updated.

Syntax:

1. locate <file name>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ locate sysctl.conf
/etc/sysctl.conf
/etc/sysctl.d/99-sysctl.conf
/etc/ufw/sysctl.conf
/snap/core/8935/etc/sysctl.conf
/snap/core/8935/etc/sysctl.d/99-sysctl.conf
/snap/core/9066/etc/sysctl.conf
/snap/core/9066/etc/sysctl.d/99-sysctl.conf
/snap/core18/1705/etc/sysctl.d/99-sysctl.conf
/snap/core18/1754/etc/sysctl.d/99-sysctl.conf
/usr/share/doc/procps/examples/sysctl.conf
/usr/share/man/man5/sysctl.conf.5.gz
```

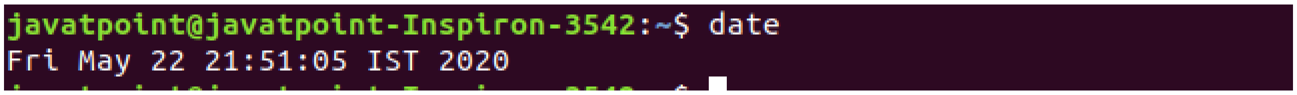
37. date Command

The date command is used to display date, time, time zone, and more.

Syntax:

1. date

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ date
Fri May 22 21:51:05 IST 2020
```

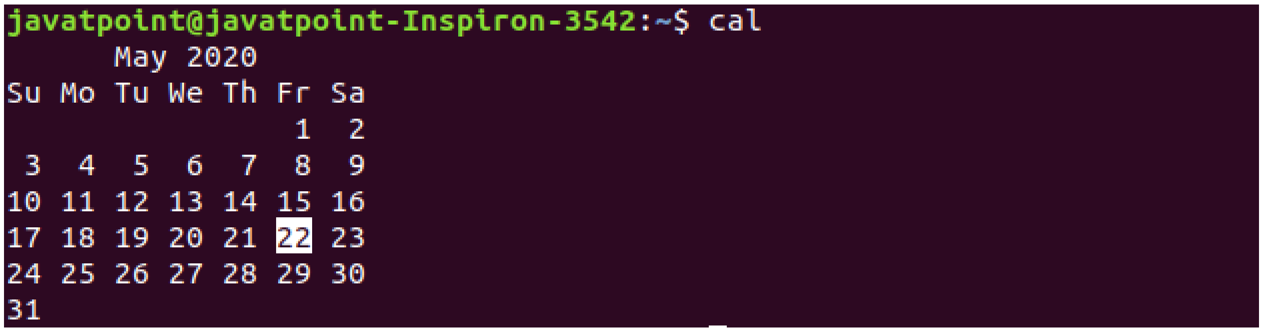
38. cal Command

The cal command is used to display the current month's calendar with the current date highlighted.

Syntax:

1. cal<

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ cal
      May 2020
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

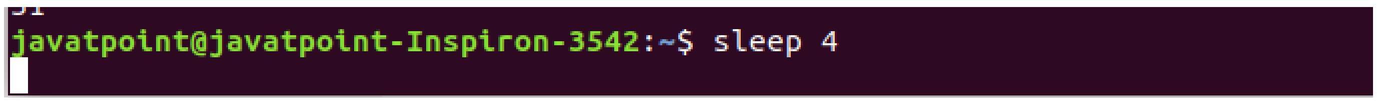
39. sleep Command

The sleep command is used to hold the terminal by the specified amount of time. By default, it takes time in seconds.

Syntax:

1. sleep **<time>**

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ sleep 4
```

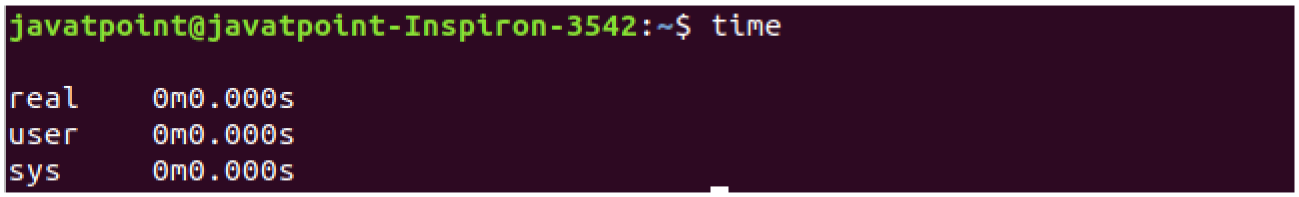
40. time Command

The time command is used to display the time to execute a command.

Syntax:

1. time

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ time

real	0m0.000s
user	0m0.000s
sys	0m0.000s
```

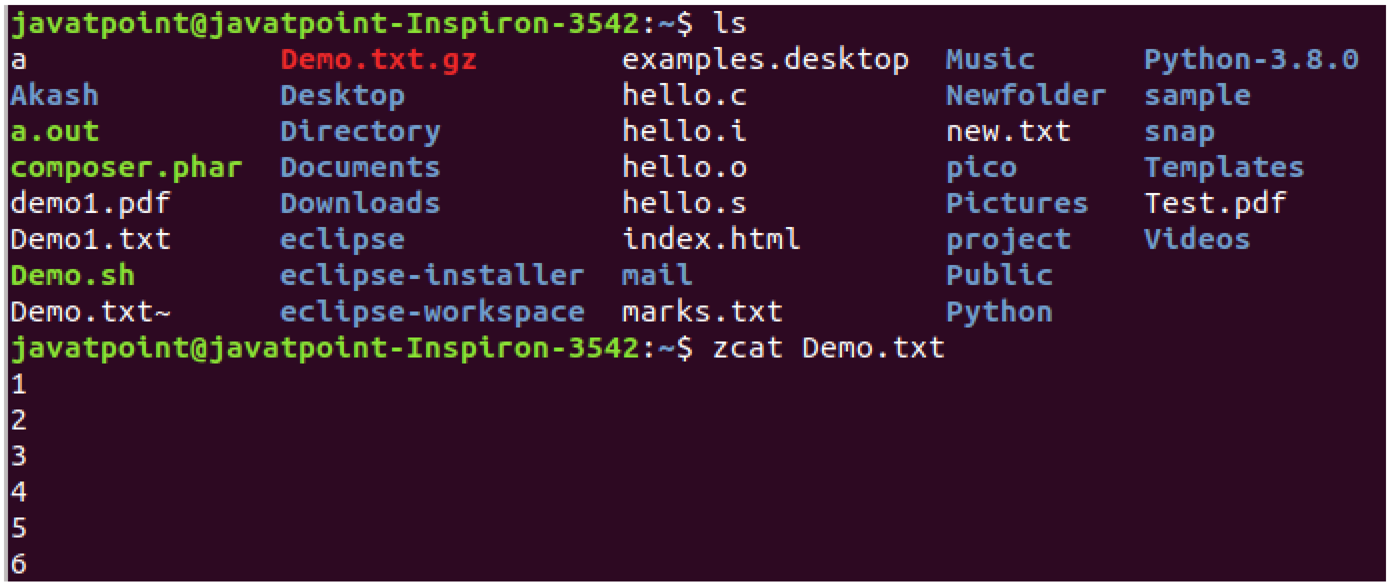
41. zcat Command

The zcat command is used to display the compressed files.

Syntax:

1. zcat **<file** name**>**

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ ls
a              Demo.txt.gz         examples.desktop  Music      Python-3.8.0
Akash          Desktop             hello.c           Newfolder  sample
a.out          Directory           hello.i           new.txt    snap
composer.phar  Documents           hello.o           pico       Templates
demo1.pdf      Downloads           hello.s           Pictures   Test.pdf
Demo1.txt      eclipse             index.html        project    Videos
Demo.sh        eclipse-installer   mail              Public
Demo.txt~      eclipse-workspace   marks.txt         Python
javatpoint@javatpoint-Inspiron-3542:~$ zcat Demo.txt
1
2
3
4
5
6
```

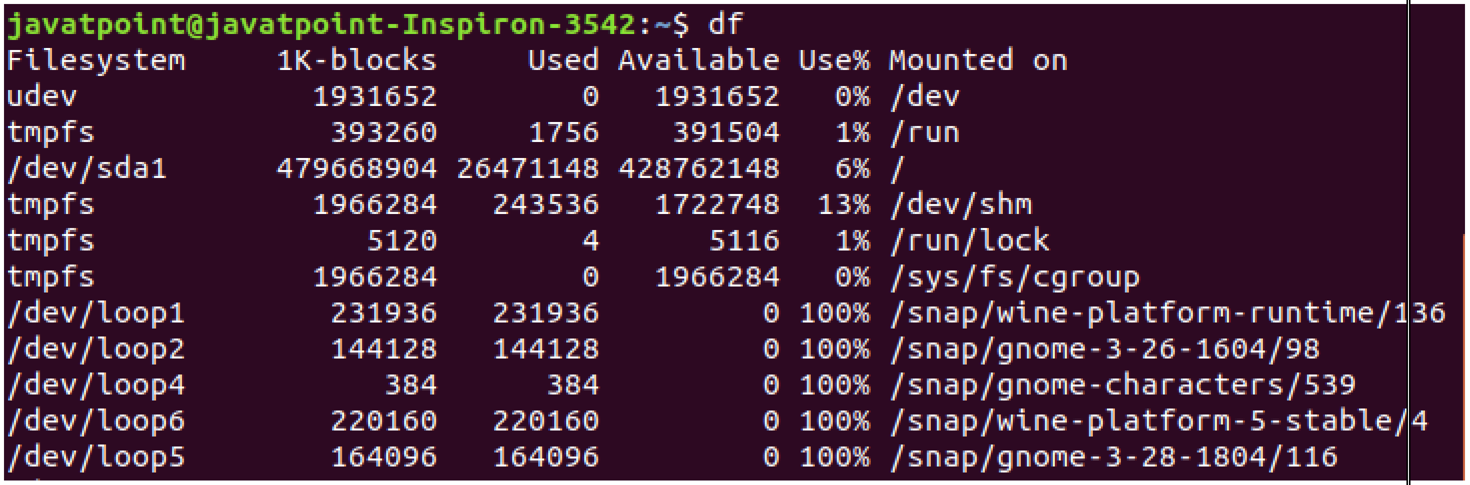
42. df Command

The df command is used to display the disk space used in the file system. It displays the output as in the number of used blocks, available blocks, and the mounted directory.

Syntax:

1. df

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev             1931652        0   1931652   0% /dev
tmpfs             393260     1756    391504   1% /run
/dev/sda1      479668904 26471148 428762148   6% /
tmpfs            1966284   243536   1722748  13% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
tmpfs            1966284        0   1966284   0% /sys/fs/cgroup
/dev/loop1        231936   231936         0 100% /snap/wine-platform-runtime/136
/dev/loop2        144128   144128         0 100% /snap/gnome-3-26-1604/98
/dev/loop4           384      384         0 100% /snap/gnome-characters/539
/dev/loop6        220160   220160         0 100% /snap/wine-platform-5-stable/4
/dev/loop5        164096   164096         0 100% /snap/gnome-3-28-1804/116
```

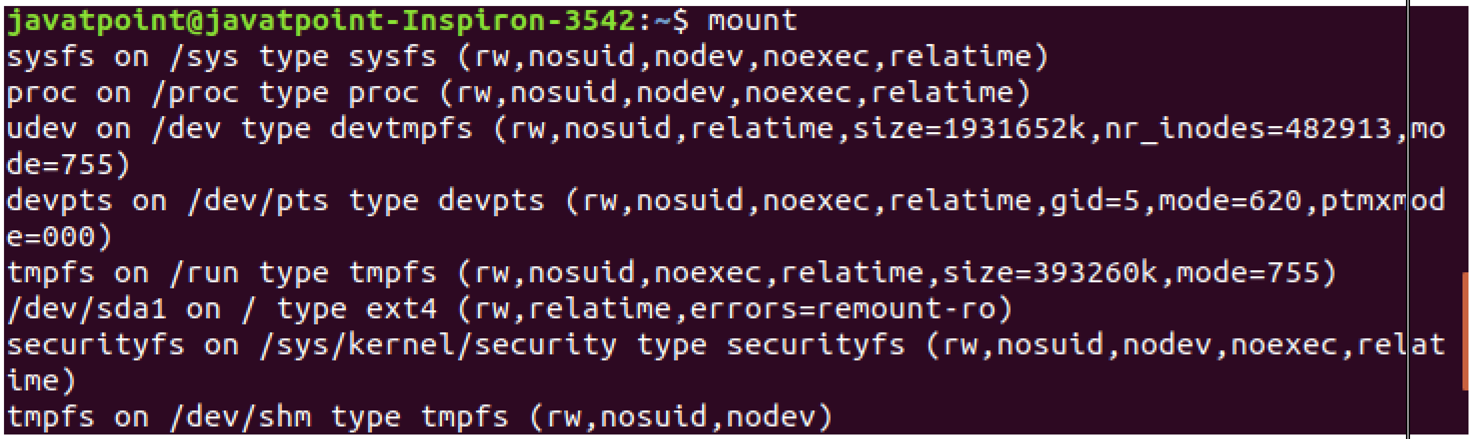
43. mount Command

The mount command is used to connect an external device file system to the system's file system.

Syntax:

1. mount -t type **<device> <directory>**

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=1931652k,nr_inodes=482913,mo
de=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmod
e=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=393260k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relat
ime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
```

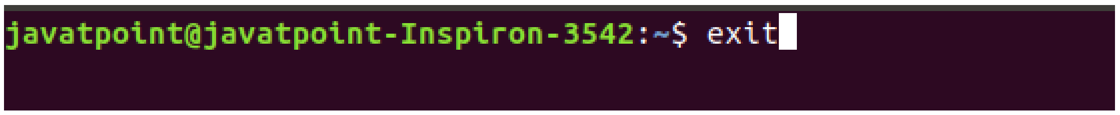
44. exit Command

Linux exit command is used to exit from the current shell. It takes a parameter as a number and exits the shell with a return of status number.

Syntax:

1. exit

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ exit
```

After pressing the ENTER key, it will exit the terminal.

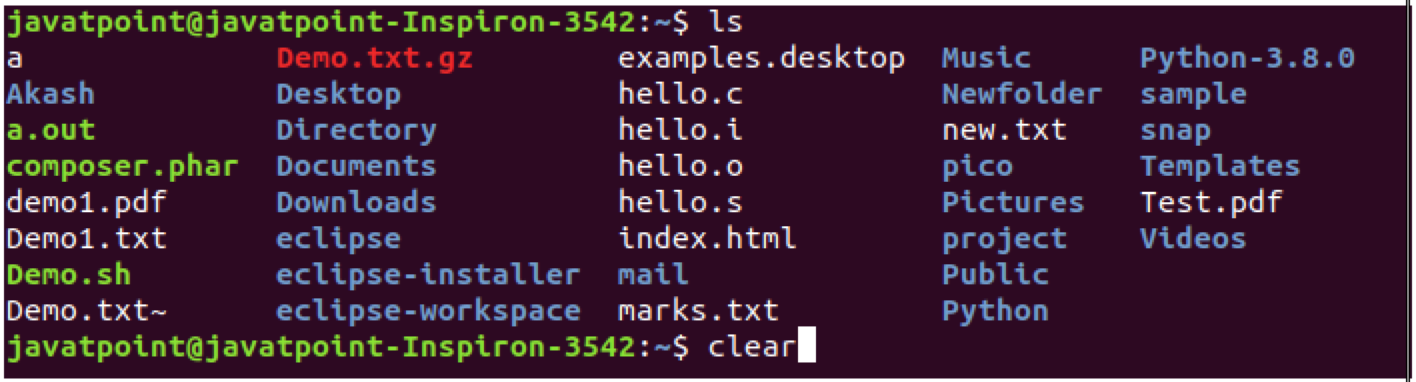
45. clear Command

Linux **clear** command is used to clear the terminal screen.

Syntax:

1. clear

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ ls
a              Demo.txt.gz         examples.desktop  Music      Python-3.8.0
Akash          Desktop             hello.c           Newfolder  sample
a.out          Directory           hello.i           new.txt    snap
composer.phar  Documents           hello.o           pico       Templates
demo1.pdf      Downloads           hello.s           Pictures   Test.pdf
Demo1.txt      eclipse             index.html        project    Videos
Demo.sh        eclipse-installer   mail              Public
Demo.txt~      eclipse-workspace   marks.txt         Python
javatpoint@javatpoint-Inspiron-3542:~$ clear
```

After pressing the ENTER key, it will clear the terminal screen.

Linux Networking Commands

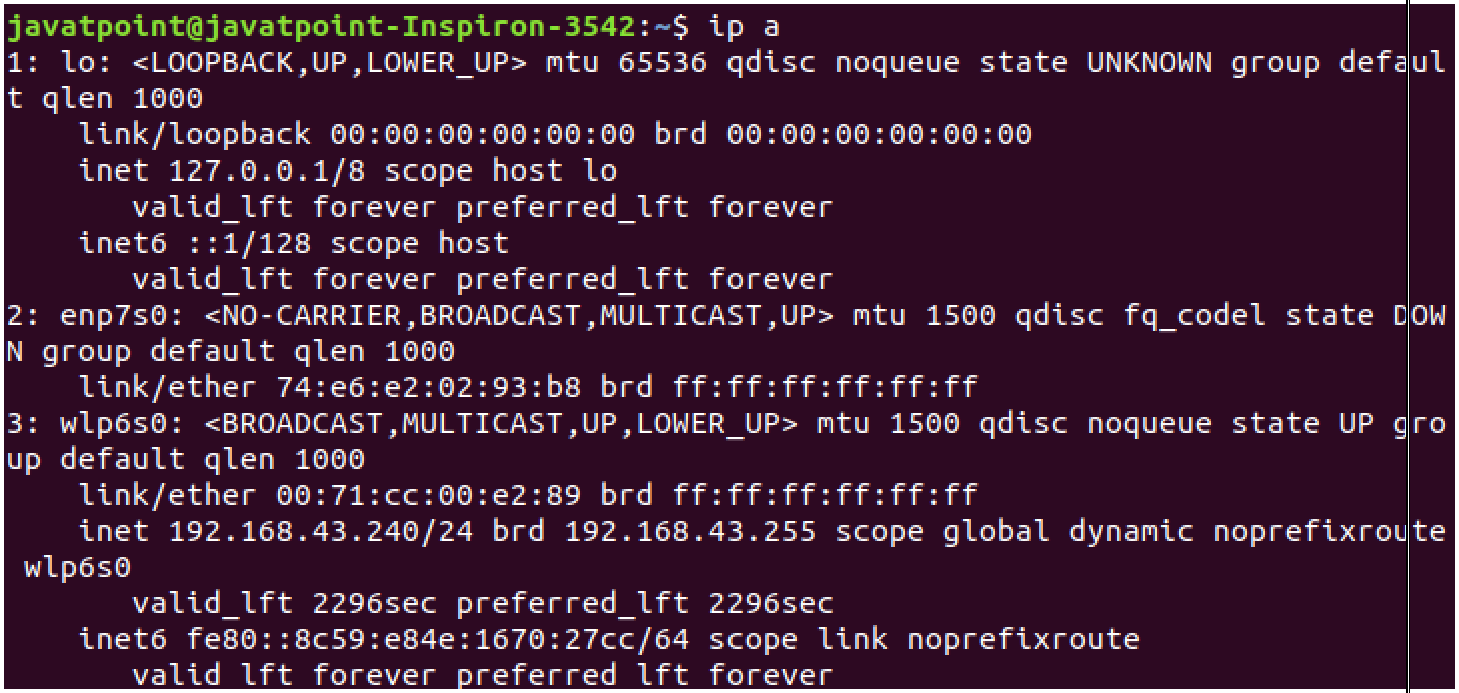
46. ip Command

Linux ip command is an updated version of the ipconfig command. It is used to assign an IP address, initialize an interface, disable an interface.

Syntax:

1. ip a or ip addr

Output:


```
javatpoint@javatpoint-Inspiron-3542:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp7s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOW
N group default qlen 1000
    link/ether 74:e6:e2:02:93:b8 brd ff:ff:ff:ff:ff:ff
3: wlp6s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP gro
up default qlen 1000
    link/ether 00:71:cc:00:e2:89 brd ff:ff:ff:ff:ff:ff
    inet 192.168.43.240/24 brd 192.168.43.255 scope global dynamic noprefixroute
 wlp6s0
       valid_lft 2296sec preferred_lft 2296sec
    inet6 fe80::8c59:e84e:1670:27cc/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

47. ssh Command

Linux ssh command is used to create a remote connection through the ssh protocol.

Syntax:

1. ssh user_name@host(IP/Domain_name)</p>

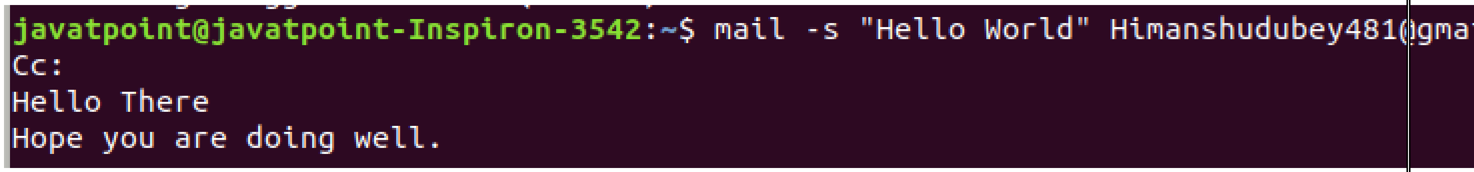
48. mail Command

The mail command is used to send emails from the command line.

Syntax:

1. mail -s "Subject" <recipient address>

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ mail -s "Hello World" Himanshudubey481@gma
Cc:
Hello There
Hope you are doing well.
```

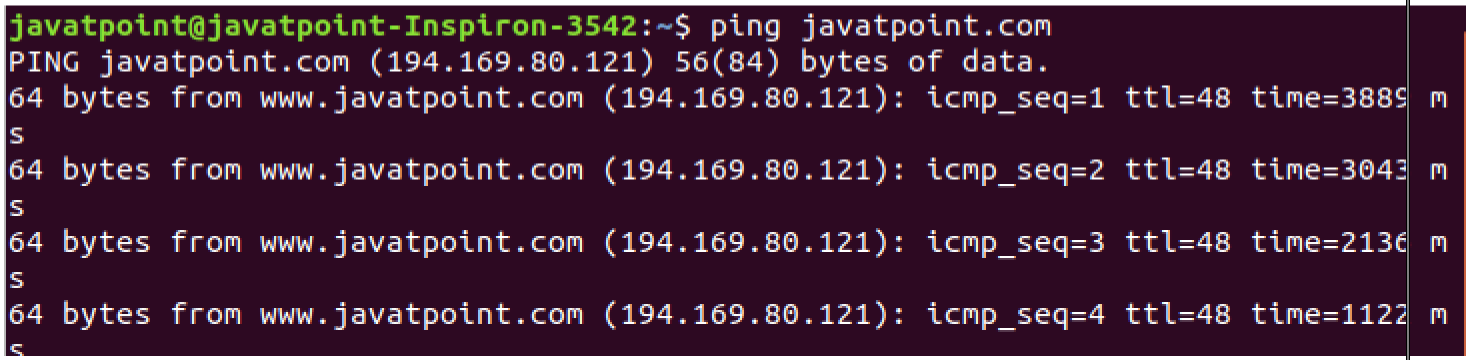
49. ping Command

The ping command is used to check the connectivity between two nodes, that is whether the server is connected. It is a short form of "Packet Internet Groper."

Syntax:

1. ping **<destination>**

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ ping javatpoint.com
PING javatpoint.com (194.169.80.121) 56(84) bytes of data.
64 bytes from www.javatpoint.com (194.169.80.121): icmp_seq=1 ttl=48 time=3889 m
s
64 bytes from www.javatpoint.com (194.169.80.121): icmp_seq=2 ttl=48 time=3043 m
s
64 bytes from www.javatpoint.com (194.169.80.121): icmp_seq=3 ttl=48 time=2136 m
s
64 bytes from www.javatpoint.com (194.169.80.121): icmp_seq=4 ttl=48 time=1122 m
s
```

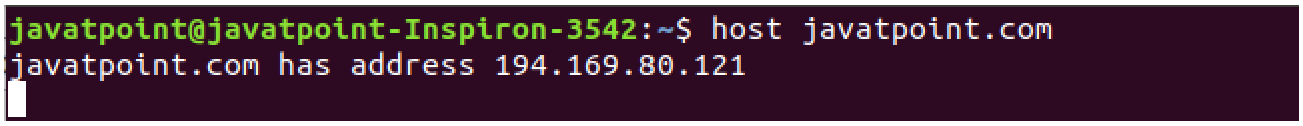
50. host Command

The host command is used to display the IP address for a given domain name and vice versa. It performs the DNS lookups for the DNS Query.

Syntax:

1. host **<domain** name**>** or **<ip** address**>**

Output:



```
javatpoint@javatpoint-Inspiron-3542:~$ host javatpoint.com
javatpoint.com has address 194.169.80.121
```

MODULE 2 – BASICS OF DB2

- **Introduction to SQL:** SQL, or Structured Query Language, is a computer language used for storing, manipulating, and retrieving data stored in a relational database. It allows you to select specific data, build complex reports, and is used in practically all technologies that process data. Here are some key operations you can perform with SQL:
    - Database Creation and Deletion: You can create and delete databases.
    - Data Manipulation: You can fetch, modify, and delete data rows.
    - Data Definition: You can define the data in a database.
    - Data Access: You can access data in the relational database management systems.
    - Permissions: You can set permissions on tables, procedures, and views.

  SQL became a standard of the American National Standards Institute (ANSI) in 1986, and the International Organization for Standardization (ISO) in 1987. Despite being a standard language, there are many different dialects of SQL, like T-SQL used by MS SQL Server and PL/SQL used by Oracle.

- **Characteristics of SQL in DB2:** SQL in DB2 is used to access data, execute queries against the database, describe and define data, manipulate data when needed, create and drop databases and tables, create views, stored procedures, functions in a database, and set permissions on tables, procedures, and views. However, the exact syntax and semantics may vary slightly due to the specific characteristics of DB2.

  Here are some characteristics of SQL in DB2:
    - **Common SQL Engine:** A query may be written once and used across products and platforms.
    - **Support for All Data Types**: DB2 can handle structured, unstructured, and relational data all on one platform.
    - **High Availability and Disaster Recovery:** DB2's replication functionality allows for safe storage and access.

- **Built-in Global Variables and Functions:** DB2 provides built-in global variables and functions.
- **Procedures:** DB2 supports system supplied procedures.

Db2 Data Types

- Integers – learn various integer types in Db2 including BIGINT, INT, and SMALLINT.
- Decimal – introduce you to decimal type and show you how to use it to store decimal numbers in tables.
- CHAR – learn how to store fixed-length character strings in the database.
- VARCHAR – store varying character strings in the database.
- DATE – discuss the DATE data type and how to store the dates in the database.
- TIME – learn about the TIME data type and how to work with time data.
- TIMESTAMP – introduce you to timestamp data type and show you how to store the timestamp in the database.

Db2 Integers: SMALLINT, INT, and BIGINT (db2tutorial.com) ---use this for further information.

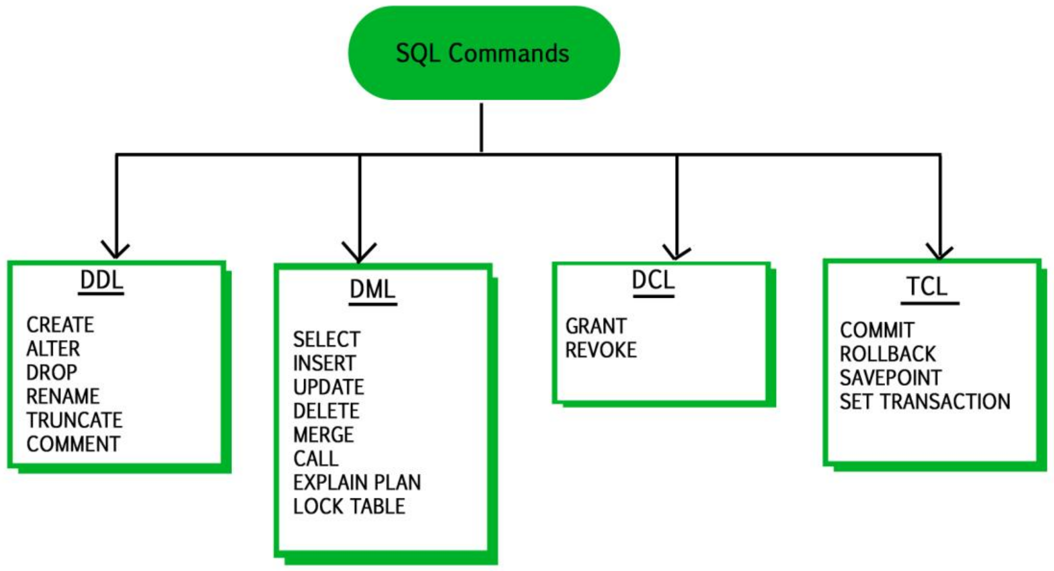
Constraints

- Primary key – learn about the primary key and how to use the primary key constraint to define the primary key for a table.
- Foreign key – enforce referential integrity between tables by using foreign key constraints.
- DEFAULT – specify a default value for a column to be inserted when the application doesn't supply the value.
- NOT NULL – prevent NULL from inserting or update to a column.
- UNIQUE – enforce the data in a column or group of columns is unique in all rows of a table.
- CHECK – place a condition on the values which are being inserted or updated into a table.

Data Definition Language

This section shows you how to manage the most important database objects including databases and tables.

- CREATE TABLE – show you how to create a new table in the database.
- Identity column – learn how to define an identity column for a table.
- ADD COLUMN – describe how to add one or more columns to a table
- ALTER COLUMN – show you how to modify definitions of existing columns in a table.
- DROP COLUMN – walk you through the process of dropping one or more columns in a table.
- DROP TABLE – show you how to delete table objects permanently from the database.
- TRUNCATE TABLE – give you a more efficient way to delete all data from a big table.
- Rename a table – learn how to change the name of an existing table to the new one.



DDL (Data Definition Language) :

Data Definition Language is used to define the database structure or schema. DDL is also used to specify additional properties of the data. The storage structure and access methods used by the database system by a set of statements in a special type of DDL called a data storage and definition language. These statements define the implementation details of the database schema, which are usually hidden from the users. The data values stored in the database must satisfy certain consistency constraints.

For example, suppose the university requires that the account balance of a department must never be negative. The DDL provides facilities to specify such constraints. The database system checks these constraints every time the database is updated. In general, a constraint can be an arbitrary predicate pertaining to the database. However, arbitrary predicates may be costly to the test. Thus, the database system implements integrity constraints that can be tested with minimal overhead.

1. **Domain Constraints :** A domain of possible values must be associated with every attribute (for example, integer types, character types, date/time types). Declaring an attribute to be of a particular domain acts as the constraints on the values that it can take.
2. **Referential Integrity :** There are cases where we wish to ensure that a value appears in one relation for a given set of attributes also appear in a certain set of attributes in another relation i.e. Referential Integrity. For example, the department listed for each course must be one that actually exists.
3. **Assertions :** An assertion is any condition that the database must always satisfy. Domain constraints and Integrity constraints are special form of assertions.
4. **Authorization :** We may want to differentiate among the users as far as the type of access they are permitted on various data values in database. These differentiation are expressed in terms of Authorization. The most common being :
   read authorization – which allows reading but not modification of data ;
   insert authorization – which allow insertion of new data but not modification of existing data
   update authorization – which allows modification, but not deletion.

DML (Data Manipulation Language) :

DML statements are used for managing data with in schema objects.

DML are of two types –

1. **Procedural DMLs** : require a user to specify what data are needed and how to get those data.
2. **Declarative DMLs** (also referred as **Non-procedural DMLs**) : require a user to specify what data are needed without specifying how to get those data.

   Declarative DMLs are usually easier to learn and use than procedural DMLs. However, since a user does not have to specify how to get the data, the database system has to figure out an efficient means of accessing data.

Some Commands :

```
SELECT: retrieve data from the database

INSERT: insert data into a table

UPDATE: update existing data within a table

DELETE: deletes all records from a table, space for the records remain
```

Example of SQL query that finds the names of all instructors in the History department :

```
select instructor.name

 from instructor

 where instructor.dept_name = 'History';
```

The query specifies that those rows from the table instructor where the dept_name is History must be retrieved and the name attributes of these rows must be displayed.

TCL (Transaction Control Language) :

Transaction Control Language commands are used to manage transactions in the database. These are used to manage the changes made by DML-statements. It also allows statements to be grouped together into logical transactions.

Examples of TCL commands –

```
COMMIT: Commit command is used to permanently save any transaction
            into the database.
ROLLBACK: This command restores the database to last committed state.
            It is also used with savepoint command to jump to a savepoint
            in a transaction.
SAVEPOINT: Savepoint command is used to temporarily save a transaction so
            that you can rollback to that point whenever necessary.
```

DCL (Data Control Language) :

A Data Control Language is a syntax similar to a computer programming language used to control access to data stored in a database (Authorization). In particular, it is a component of Structured Query Language (SQL).

Examples of DCL commands :

```
GRANT: allow specified users to perform specified tasks.
REVOKE: cancel previously granted or denied permissions.
```

The operations for which privileges may be granted to or revoked from a user or role apply to both the Data definition language (DDL) and the Data manipulation language (DML), and may include CONNECT, SELECT, INSERT, UPDATE, DELETE, EXECUTE and USAGE.

MODULE 3 – DB2 -Advanced SQL

Window Functions

SQL window functions allow you to perform operations that are often required for creating reports, e.g. ranking data, calculating running totals and moving averages, finding the difference between rows, etc. Not only that, but you can also divide data into windows, which enables you to perform operations on data subsets rather than the data as a whole

This code will show the difference in yearly numbers of cars sold, according to make (i.e. car brand):

```
SELECT  car_make,

        cars_sold,

        year,

        cars_sold - LAG(cars_sold) OVER (PARTITION BY car_make ORDER BY year) AS sales_diff

FROM cars_sale;
```

Common Table Expressions (CTEs)

CTEs will allow you to write complex queries without using subqueries keeping your code simple and straightforward. They give you the possibility to produce complex reports quickly and efficiently. They also enable you to make some calculations you wouldn't be able to do otherwise.

What is a common table expression, you might ask? It's a temporary result you can use in the SELECT statement. It works like a temporary table – you can join it with other tables, other CTEs, or with itself.

They can be helpful if you, for instance, have to report on time spent on a particular project. On one side, there's a table containing data about the date when each employee worked on this project. There's also the start time and end time. On the other side, there's a table containing employee names. You have to produce a table showing every employee's name and his or her average time spent on this project.

Here's how the CTE can help you:

```
WITH time_worked AS (

    SELECT  employee_id,

            end_time - start_time AS time

FROM project_timesheet

)

SELECT  e.first_name,
        e.last_name,

        AVG (tw.time) AS avg_time_worked

FROM employee e

LEFT JOIN time_worked tw

ON e.id = tw.employee_id

GROUP BY e.first_name, e.last_name;
```

How does this CTE work? Every CTE opens with the `WITH` clause. Then you must name your CTE; in this case, it's `time_worked`. Then you write a `SELECT` statement. Here, I'll use the CTE to calculate how much time each employee worked every time they worked on the project. I need the CTE because I don't have this information stated explicitly in the table; I only have the `start_time` and `end_time`. To calculate the average time worked, the first step is to get the time worked. That's why this CTE deducts the `start_time` from the `end_time` and shows the result in the column `time`. The data is taken from the table `project_timesheet`.

Now that I've written the CTE, I can use it in the next `SELECT` statement. First, I'll get the first name and the last name from the table `employee`. Then I'll use the `AVG()` function on the column `time` from the CTE `time_worked`. To do that, I've used the `LEFT JOIN` – and I've used it exactly like I would with any other table. Finally, the data is grouped by the employees' first and last names.

QL's `GROUP BY` extensions provide you with additional possibilities for grouping data. This, in return, can increase the complexity of your data analysis and the reports you create.

There are three `GROUP BY` extensions:

- `ROLLUP`
- `CUBE`

- `GROUPING SETS`

Unlike regular `GROUP BY`, `ROLLUP` lets you group the data into multiple data sets and aggregate results on different levels. Fancy talk, but simply put: you can use `ROLLUP` to calculate totals and subtotals, just like in Excel pivot tables.

The `CUBE` extension is similar, but there's one crucial difference. `CUBE` will generate subtotals for every combination of the columns specified.

Finally, there are `GROUPING SETs`. A grouping set is a set of columns you use in the `GROUP BY` clause. You can connect different queries containing `GROUP BY` if you use `UNION ALL`. However, the more queries you have, the messier it gets. You can achieve the same result but with much neater queries by using `GROUPING SETS`.

Let me show you how `ROLLUP` works. Suppose you're working for a guitar store that has several locations. You'll sometimes need to create a report showing the total number of guitars you have in stock. Here's a query that will do that on a manufacturer, model, and store level:

```
SELECT  manufacturer,

        model,

        store,

        SUM(quantity) AS quantity_sum

FROM guitars

GROUP BY ROLLUP (manufacturer, model, store)

ORDER BY manufacturer;
```

This doesn't look complicated. It's a simple `SELECT` statement that will give you the columns `manufacturer`, `model`, and store from the table `guitars`. I've used the aggregate function `SUM()` to get the quantity. Then I wrote `GROUP BY` followed immediately by `ROLLUP`. The data will be grouped according to the columns in the parentheses. Finally, the result is ordered by the manufacturer.

****

MODULE 4 – JOINS and LOCKS

- **Locks:** In IBM DB2, locks play a crucial role in managing concurrent access to data to maintain data integrity and consistency.
- **Lock types:**
  1. **Shared Lock (S):** Allows multiple transactions to read a resource simultaneously but prevents any of them from updating it.
  2. **Exclusive Lock (X):** Grants exclusive access to a resource, preventing other transactions from both reading and updating it.

- **Issuing locks in SQL:** To explicitly acquire a lock on a table or rows, you can use the FOR UPDATE or FOR SHARE clauses in your SQL statements.

  Code:
  ```
  SELECT * FROM your_table WHERE condition FOR UPDATE;
  ```

- **Lock Size in IBM DB2:** In IBM DB2, lock size refers to the granularity at which locks are acquired. DB2 supports different lock sizes:
  - Row-Level Locking: Locks are acquired at the row level, allowing for more concurrent access but potentially incurring higher overhead.
  - Page-Level Locking: Locks are acquired at the page level, reducing overhead but potentially limiting concurrency.
  - Table-Level Locking: Locks are acquired at the table level, providing the least

concurrency but simplifying lock management.

You can specify the lock size using the LOCKSIZE clause in the CREATE TABLE statement or the ALTER TABLE statement.

Example: creating a table with page level locking

```
CREATE TABLE your_table (column1 INT, column2 VARCHAR(255)) LOCKSIZE PAGE;
```

- **Grant and Revoke:** In IBM DB2, the GRANT and REVOKE statements are used to manage permissions and access control for database objects.
  The GRANT statement is used to give specific privileges to users or roles on database objects such as tables, views, procedures, or sequences. Privileges can include: SELECT, INSERT, UPDATE, DELETE, etc.
  The REVOKE statement is used to remove previously granted privileges from users or roles.

  Example:

  ```
  GRANT SELECT, INSERT ON your_table TO user1;
  REVOKE SELECT, INSERT ON your_table FROM user1;
  ```

- **Isolation Levels:** IBM DB2 supports different isolation levels, influencing how transactions acquire and release locks. Common isolation levels include:
  - UR (Uncommitted Read): Allows a transaction to see uncommitted changes made by other transactions.
  - CS (Cursor Stability): Ensures that the result set of a query is not changed by other transactions while it is being read.
  - RS (Read Stability): Locks are held until the end of the transaction, ensuring that the data read is stable throughout the transaction.
  - RR (Repeatable Read): Ensures that a query sees a consistent snapshot of the data throughout the transaction.

  You can set the isolation level using the ISOLATION LEVEL clause in the DECLARE CURSOR or SET CURRENT ISOLATION statement.

Example: setting isolation level to Repeatable read.

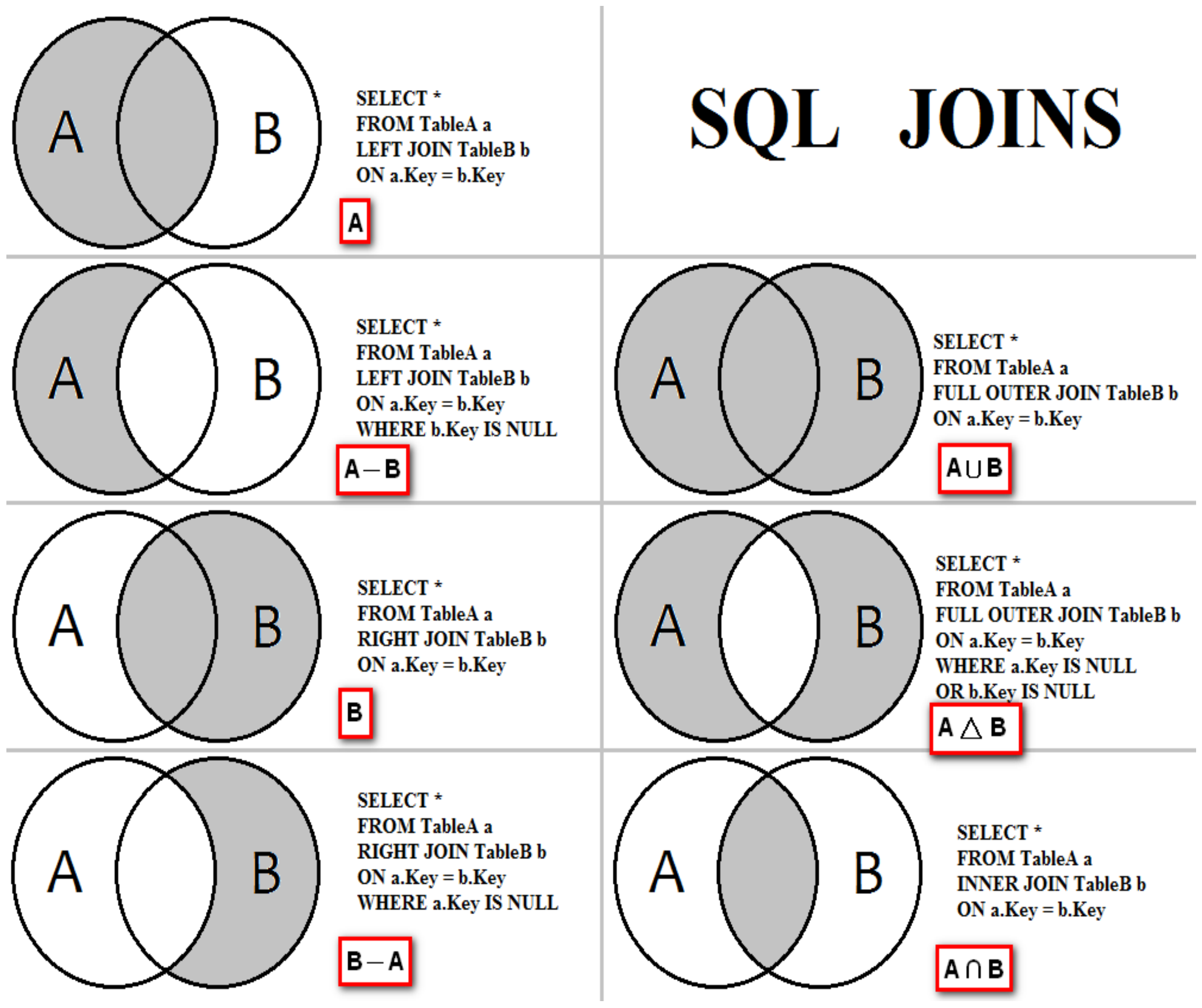
```
SET CURRENT ISOLATION RR;
```

- **Data Integrity in IBM DB2:** IBM DB2 ensures data integrity through various means, including:
  - Primary and Foreign Keys: Enforcing relationships between tables.
  - Check Constraints: Defining rules for the values allowed in a column.
  - Unique Constraints: Ensuring uniqueness of values in a column or a set of columns.

Example: creating a table with primary key.

```
CREATE TABLE your_table (id INT PRIMARY KEY, name VARCHAR(255));
```

Joins:

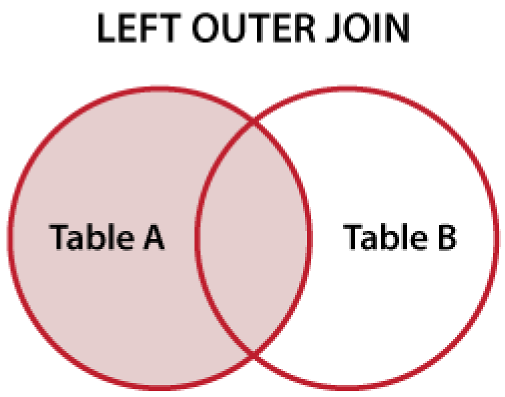

SQL JOINS
A
B
SELECT * FROM TableA a LEFT JOIN TableB b ON a.Key = b.Key
A
SELECT * FROM TableA a LEFT JOIN TableB b ON a.Key = b.Key WHERE b.Key IS NULL
A – B
SELECT * FROM TableA a FULL OUTER JOIN TableB b ON a.Key = b.Key
A ∪ B
SELECT * FROM TableA a RIGHT JOIN TableB b ON a.Key = b.Key
B
SELECT * FROM TableA a FULL OUTER JOIN TableB b ON a.Key = b.Key WHERE a.Key IS NULL OR b.Key IS NULL
A △ B
SELECT * FROM TableA a RIGHT JOIN TableB b ON a.Key = b.Key WHERE a.Key IS NULL
B – A
SELECT * FROM TableA a INNER JOIN TableB b ON a.Key = b.Key
A ∩ B

Inner Join vs Outer Join

OUTER JOINS differ from traditional INNER JOINS in their ability to return all rows from one table, regardless of whether they have matching counterparts in the other table. This feature makes OUTER JOINS invaluable for scenarios where complete information is desired, even if there are no exact matches between the tables.

- All records that match the join condition or predicate. That's the expression right after the ON keyword, much like the INNER JOIN output.
- Non-NULL values from the left table with the null counterparts from the right table. Non-NULL values from the right table with the null counterparts from the left table.

Left Outer Join



At its core, a left outer join is a method for combining rows from two or more tables based on a related column between them. Unlike inner joins, which only return rows that have matching values in both tables, a left outer join returns all rows from the left table (referred to as the "left" or "first" table) and matching rows from the right table (referred to as the “right” or “second” table). If there is no match in the right table, NULL values are returned for the columns from the right table.

Input Data

EmployeeName Table-(T1) **Position Table-(T2)**

ID	NAME
10	Sandy
20	Sam
30	Cindy

ID	TITLE	TEAMSIZE
20	Sales Mgr	5
30	Clerk	10
30	Manager	2
40	Sales Rep	7
50	Sr. Manager.	11

Query

```
SELECT T1.NAME,
       T1.ID,
       T2.ID,
       T2.TITLE,
       T2.TEAMSIZE
FROM EmployeeName T1
LEFT OUTER JOIN Position T2
ON T1.ID = T2.ID
ORDER BY T1.ID
       , T2.TITLE;
```

Result

```
ID   NAME     ID     TITLE        TEAMSIZE
10   Sandy    --     ---------    ---------
20   Sam      20     Sales Mgr     5
30   Cindy    30     Clerk        10
30   Cindy    30     Manager.      2
```

Suppose we want to return the employee with no title. To do that, add a WHERE clause to include only rows with nulls from the Position table.

Query using ON or WHERE

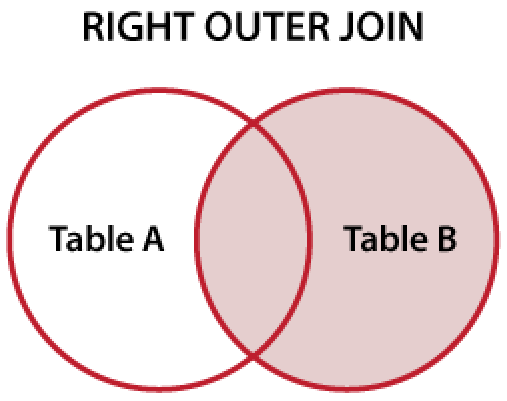
```
SELECT T1.NAME,
       T1.ID,
       T2.ID,
       T2.TITLE,
       T2.TEAMSIZE
FROM EmployeeName T1
LEFT OUTER JOIN Position T2
ON T1.ID = T2.ID
AND T2.TITLE IS NULL

SELECTT1.NAME,
       T1.ID,
       T2.ID,
       T2.TITLE,
       T2.TEAMSIZE
FROMEmployeeName T1
LEFT OUTER JOINPosition T2
ONT1.ID=T2.ID
WHERE T2.TITLE IS NULL
```

Both the above queries will give the same result.

Result

```
ID   NAME     ID     TITLE       TEAMSIZE
10   Sandy    --     ---------   ---------
```

Right Outer Join



A right outer join, also known as a right join, is a method for merging rows from two or more tables based on a related column between them. Unlike inner joins, which only return rows with matching values in both tables, a right outer join returns all rows from the right table (referred to as the "right" or "second" table) and matching rows from the left table (referred to as the "left" or "first" table). If there is no match in the left table, NULL values are returned for the columns from the left table.

Input Data

EmployeeName Table-(T1)

ID	NAME
10	Sandy
20	Sam
30	Cindy

Position Table-(T2)

ID	TITLE	TEAM
20	Sales Mgr	5
30	Clerk	10
30	Manager	2
40	Sales Rep	7
50	Sr. Manager.	11

Query

```
SELECT T1.NAME,
       T1.ID,
       T2.ID,
       T2.TITLE
FROM EmployeeName T1
RIGHT OUTER JOIN Position T2
ON T1.ID=T2.ID
```

```
ORDER BY T1.ID
       , T2.TITLE;
```

Result

```
ID   NAME     ID    TITLE
20   Sam      20    Sales Mgr
30   Cindy    30    Clerk
30   Cindy    30    Manager
--   -----    40    Sales Rep
--   -----    50    Manager
```

Suppose we want to return the employee with no title. To do that, add a WHERE clause to include only rows with nulls from the EmployeeName table.

Query using ON or WHERE

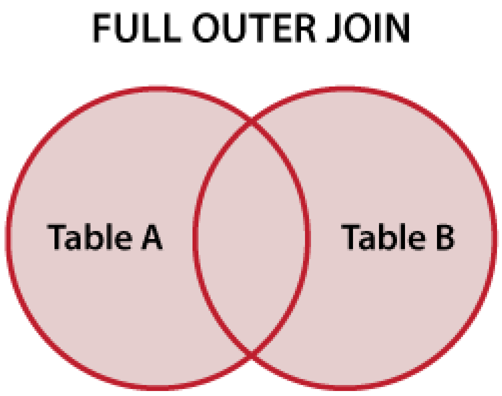
```
SELECT T1.NAME,
       T1.ID,
       T2.ID,
       T2.TITLE
FROM EmployeeName T1
RIGHT OUTER JOIN Position T2
ON T1.ID = T2.ID
AND T1.NAME IS NULL

SELECT T1.NAME,
       T1.ID,
       T2.ID,
       T2.TITLE
FROM EmployeeName T1
LEFT OUTER JOIN Position T2
ON T1.ID=T2.ID
WHERE T1.NAME IS NULL
```

Both the above queries will give the same result.

Result

```
ID    NAME     ID     TITLE
--    -----    40     Sales Rep
--    -----    50     Manager
```

Full Outer Join



A full outer join, also known simply as a full join, is a method for merging rows from two tables based on a related column between them. Unlike inner joins, which only return rows with matching values in both tables, and outer joins (left and right), which include all rows from one table and matching rows from the other, a full outer join returns all rows from both tables. If there is no match in either table, NULL values are returned for the columns from the table with no match.

Input Data

EmployeeName Table-(T1) **Position Table-(T2)**

ID	NAME	ID	TITLE	TEAM
10	Sandy	20	Sales Mgr	5
20	Sam	30	Clerk	10
30	Cindy	30	Manager	2
		40	Sales Rep	7
		50	Sr. Manager.	11

Query

```
SELECT T1.NAME,
       T1.ID,
       T2.ID,
       T2.TITLE
FROM EmployeeName T1
FULL OUTER JOIN Position T2
ON T1.ID=T2.ID
ORDER BY T1.ID,
         T2.ID
       , T2.TITLE;
```

Result

```
 ID   NAME     ID     TITLE
 10   Sandy    --     ---------
 20   Sam      20     Sales Mgr
 30   Cindy    30     Clerk
 30   Cindy    30     Manager
 --   -----    40     Sales Rep
 --   -----    50     Manager
```

Suppose we want to return the employee with ID<30. To do that, add a WHERE clause to include only rows with ID<30 from EmployeeName table.

Query using ON or WHERE

```
SELECT T1.NAME,
       T1.ID,
       T2.ID,
       T2.TITLE
FROM EmployeeName T1
FULL OUTER JOIN Position T2
ON T1.ID = T2.ID
AND T1.ID < 30
ORDER BY T1.ID,
         T2.ID
       , T2.TITLE;
```

Result

```
 ID   NAME     ID     TITLE
 10   Sandy    --     ---------
 20   Sam      20     Sales Mgr
```

In this example the “T1.ID < 30” check is done during the join where it does not any eliminate rows but rather limits those that match in the two views:

```
SELECTT1.NAME,
        T1.ID,
        T2.ID,
        T2.TITLE
 FROMEmployeeName T1
 FULL OUTER JOIN Position T2
 ONT1.ID=T2.ID
 WHERE T1.ID < 30
 ORDER BY T1.ID,
          T2.ID
        , T2.TITLE;
```

Result

```
 ID   NAME     ID     TITLE
 10   Sandy    --     ---------
 20   Sam      20     Sales Mgr
 30   Cindy    --     ---------
 --   -----    30     Clerk
 --   -----    30     Manager
 --   -----    40     Sales Rep
 --   -----    50     Manager
```

Module 5: OPERATION AND RECOVERY

DB2 basic operational concepts – starting and stopping DB2 – scheduling administrative tasks – monitoring and controlling DB2 and its connection – managing the log and the bootstrap data set – recovering from different DB2 – reading log records.

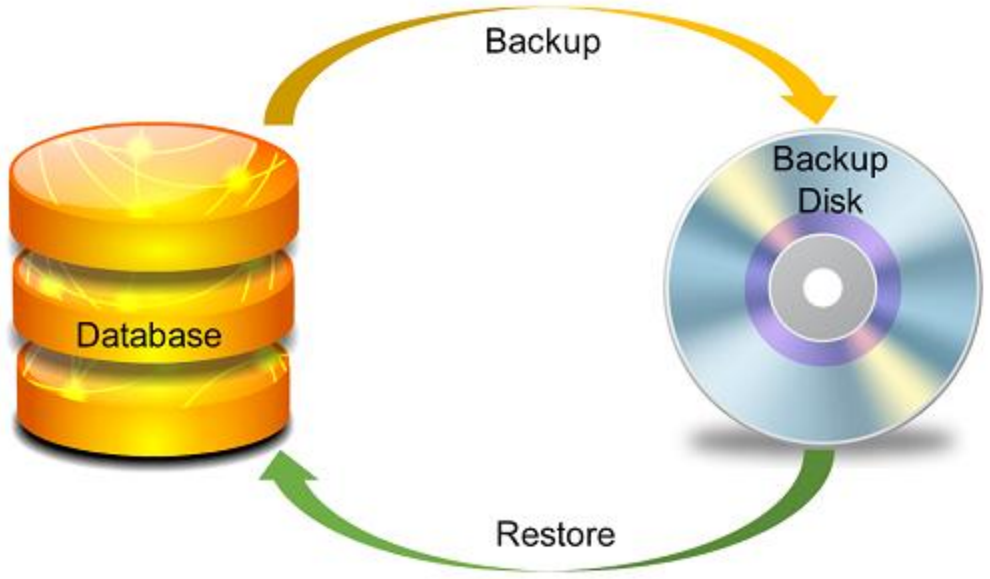
Starting and stopping DB2 You start and stop DB2 by using the START DB2 and STOP DB2 commands. Before you begin Before DB2 is stopped, the system takes a shutdown checkpoint. This checkpoint and the recovery log give DB2 the information it needs to restart.

Starting DB2 When DB2 is installed, it is defined as a formal z/OS subsystem. About this task Afterward, the following message appears during any IPL of z/OS: DSN3100I - DSN3UR00 - SUBSYSTEM ssnm READY FOR -START COMMAND where ssnm is the DB2 subsystem name. Procedure To start a DB2 subsystem: Issue the START DB2 command by using one of the following methods: v Issue the START DB2 command from a z/OS console that is authorized to issue system control commands (z/OS command group SYS). The command must be entered from the authorized console and cannot be submitted through JES or TSO. Starting DB2 by a JES batch job or a z/OS START command is impossible. The attempt is likely to start an address space for DB2 that will abend (most likely with reason code X'00E8000F'). v Start DB2 from an APF-authorized program by passing a START DB2 command to the MGCRE (SVC 34) z/OS service.

Messages at start DB2 issues a variety of messages when you start DB2. The specific messages vary based on the parameters that you specify.

Stopping DB2 Before DB2 stops, all DB2-related write to operator with reply (WTOR) messages must receive replies

DB2 - Backup and Recovery



Introduction

Backup and recovery methods are designed to keep our information safe. In Command Line Interface (CLI) or Graphical User Interface (GUI) using backup and recovery utilities you can take backup or restore the data of databases in DB2 UDB.

Logging

Log files consist of error logs, which are used to recover from application errors. The logs keep the record of changes in the database. There are two types of logging as described below:

Circular logging

It is a method where the old transaction logs are overwritten when there is a need to allocate a new transaction log file, thus erasing the sequences of log files and reusing them. You are permitted to take only full back-up in offline mode. i.e., the database must be offline to take the full backup.

Archive logging

This mode supports for Online Backup and database recovery using log files called roll forward recovery. The mode of backup can be changed from circular to archive by setting logretain or userexit to ON. For archive logging, backup setting database require a directory that is writable for DB2 process.

Backup

Using **Backup** command you can take copy of entire database. This backup copy includes database system files, data files, log files, control information and so on.

You can take backup while working offline as well as online.

Offline backup

Syntax: [To list the active applications/databases]

```
db2 list application
```

Output:

```
Auth Id  Application    Appl.      Application Id
DB       # of
         Name           Handle
Name    Agents
-------- -------------- ---------- --------------------
---------------------------------------- -------- -----
DB2INST1 db2bp          39
*LOCAL.db2inst1.140722043938
ONE      1
```

Syntax: [To force application using app. Handled id]

```
db2 "force application (39)"
```

Output:

```
DB20000I  The FORCE APPLICATION command completed
successfully.

DB21024I  This command is asynchronous and may not
be effective immediately.
```

Syntax: [To terminate Database Connection]

```
db2 terminate
```

Syntax: [To deactivate Database]

```
db2 deactivate database one
```

Syntax: [To take the backup file]

```
db2 backup database <db_name> to <location>
```

Example:

```
db2 backup database one to /home/db2inst1/
```

Output:

```
Backup successful. The timestamp for this backup image is :
20140722105345
```

Online backup

To start, you need to change the mode from **Circular logging** to **Archive Logging**.

Syntax: [To check if the database is using circular or archive logging]

```
db2 get db cfg for one | grep LOGARCH
```

Output:

```
First log archive method (LOGARCHMETH1) = OFF
 Archive compression for logarchmeth1  (LOGARCHCOMPR1) = OFF
 Options for logarchmeth1              (LOGARCHOPT1) =
 Second log archive method             (LOGARCHMETH2) = OFF
 Archive compression for logarchmeth2  (LOGARCHCOMPR2) = OFF
 Options for logarchmeth2              (LOGARCHOPT2) =
```

In the above output, the highlighted values are [logarchmeth1 and logarchmeth2] in off mode, which implies that the current database in “CIRCULLAR LOGGING” mode. If you need to work with ‘ARCHIVE LOGGING’ mode, you need to change or add path in the variables logarchmeth1 and logarchmeth2 present in the configuration file.

Updating logarchmeth1 with required archive directory

Syntax: [To make directories]

```
mkdir backup
mkdir backup/ArchiveDest
```

Syntax: [To provide user permissions for folder]

```
chown db2inst1:db2iadm1 backup/ArchiveDest
```

Syntax: [To update configuration LOGARCHMETH1]

```
db2 update database configuration for one using LOGARCHMETH1
'DISK:/home/db2inst1/backup/ArchiveDest'
```

You can take offline backup for safety, activate the database and connect to it.

Syntax: [To take online backup]

```
db2 backup database one online to
/home/db2inst1/onlinebackup/ compress include logs
```

Output:

```
db2 backup database one online to
/home/db2inst1/onlinebackup/ compress include logs
```

Verify Backup file using following command:

Syntax:

```
db2ckbkp <location/backup file>
```

Example:

```
db2ckbkp
/home/db2inst1/ONE.0.db2inst1.DBPART000.20140722112743.001
```

Listing the history of backup files

Syntax:

```
db2 list history backup all for one
```

Output:

```
                    List History File for one


Number of matching file entries = 4


Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log
Backup ID
 -- --- ------------------ ---- --- ------------ ------------
 --------------
  B  D  20140722105345001   F    D  S0000000.LOG S0000000.LOG
```

```
------------------------------------------------------------
----------------

Contains 4 tablespace(s):
00001 SYSCATSPACE

00002 USERSPACE1

00003 SYSTOOLSPACE

00004 TS1
 ------------------------------------------------------------
 ----------------
 Comment: DB2 BACKUP ONE OFFLINE

Start Time: 20140722105345

  End Time: 20140722105347

    Status: A
------------------------------------------------------------
----------------
EID: 3 Location: /home/db2inst1


Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log
Backup ID
-- --- ------------------ ---- --- ------------ ------------
```

```
--------------
 B  D  20140722112239000   N       S0000000.LOG S0000000.LOG
-------------------------------------------------------------
--------------------------------------------------------------
-------------------------------

Comment: DB2 BACKUP ONE ONLINE

Start Time: 20140722112239

  End Time: 20140722112240

    Status: A
-------------------------------------------------------------
----------------
 EID: 4 Location:
SQLCA Information

 sqlcaid : SQLCA     sqlcabc: 136   sqlcode: -2413   sqlerrml: 0

 sqlerrmc:
 sqlerrp : sqlubIni
 sqlerrd : (1) 0                (2) 0                (3) 0

           (4) 0                (5) 0                (6) 0

 sqlwarn : (1)      (2)      (3)      (4)        (5)       (6)

           (7)      (8)      (9)      (10)       (11)
```

```
sqlstate:

Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log
Backup ID
 -- --- ------------------ ---- --- ------------ ------------
 --------------
  B  D  20140722112743001   F    D  S0000000.LOG S0000000.LOG

------------------------------------------------------------
----------------
Contains 4 tablespace(s):

00001 SYSCATSPACE

00002 USERSPACE1

00003 SYSTOOLSPACE

00004 TS1
 ------------------------------------------------------------
 ----------------
 Comment: DB2 BACKUP ONE OFFLINE

Start Time: 20140722112743

  End Time: 20140722112743

    Status: A
-------------------------------------------------------------
```

```
  ----------------
 EID: 5 Location: /home/db2inst1

 Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log
 Backup ID
  -------------------------------------------------------------
  ----------------

R  D  20140722114519001   F
20140722112743

 -------------------------------------------------------------
 ----------------
 Contains 4 tablespace(s):

 00001 SYSCATSPACE

  00002 USERSPACE1

 00003 SYSTOOLSPACE

 00004 TS1
 -------------------------------------------------------------
 ----------------
Comment: RESTORE ONE WITH RF

 Start Time: 20140722114519

   End Time: 20140722115015
```

```
     Status: A


 ------------------------------------------------------------
 ----------------
  EID: 6 Location:
```

Restoring the database from backup

To restore the database from backup file, you need to follow the given syntax:

Syntax:

```
db2 restore database <db_name> from <location>
taken at <timestamp>
```

Example:

```
db2 restore database one from /home/db2inst1/ taken at
20140722112743
```

Output:

```
SQL2523W  Warning!  Restoring to an existing database that is
different from

the database on the backup image, but have matching names.
The target database

will be overwritten by the backup version.  The Roll-forward
recovery logs
```

```
associated with the target database will be deleted.

Do you want to continue ? (y/n) y

DB20000I  The RESTORE DATABASE command completed successfully.
```

Roll forward all the logs located in the log directory, including latest changes just before the disk drive failure.

Syntax:

```
db2 rollforward db <db_name> to end of logs and stop
```

Example:

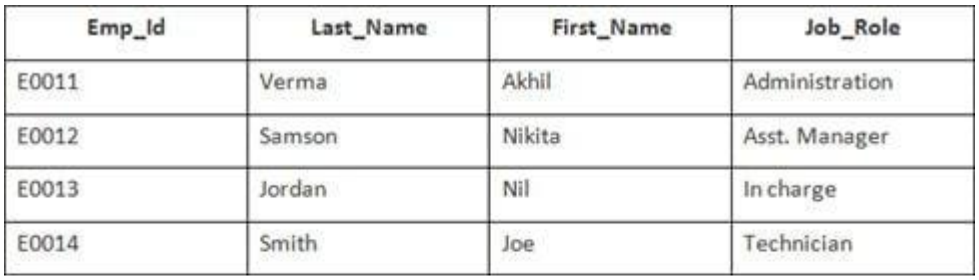
```
db2 rollforward db one to end of logs and stop
```

Output:

```
                                 Rollforward Status
 Input database alias                           = one
 Number of members have returned status = 1
 Member ID                                      = 0
 Rollforward status                             = not pending
 Next log file to be read                       =
 Log files processed                            = S0000000.LOG -
 S0000001.LOG
 Last committed transaction                    = 2014-07-22-
 06.00.33.000000 UTC
DB20000I  The ROLLFORWARD command completed successfully.
```

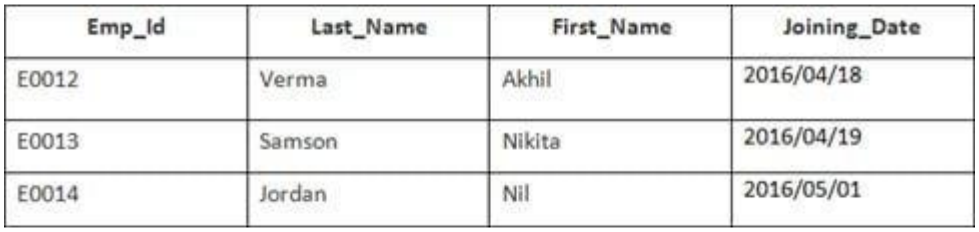
Questions for Practice:

1. In this example, we have a table **Employee** with the following data:

- 

Emp_Id	Last_Name	First_Name	Job_Role
E0011	Verma	Akhil	Administration
E0012	Samson	Nikita	Asst. Manager
E0013	Jordan	Nil	In charge
E0014	Smith	Joe	Technician

The second table's name is **Joining.**

- 

Emp_Id	Last_Name	First_Name	Joining_Date
E0012	Verma	Akhil	2016/04/18
E0013	Samson	Nikita	2016/04/19
E0014	Jordan	Nil	2016/05/01

Enter the following SQL statement:

```
SELECTEmployee.Emp_id, Joining.Joining_Date
 FROMEmployee
 INNERJOINJoining
 ONEmployee.Emp_id = Joining.Emp_id
 ORDERBYEmployee.Emp_id;
```

2. **Employee** and **Orders** tables have a matching *customer_id* value.

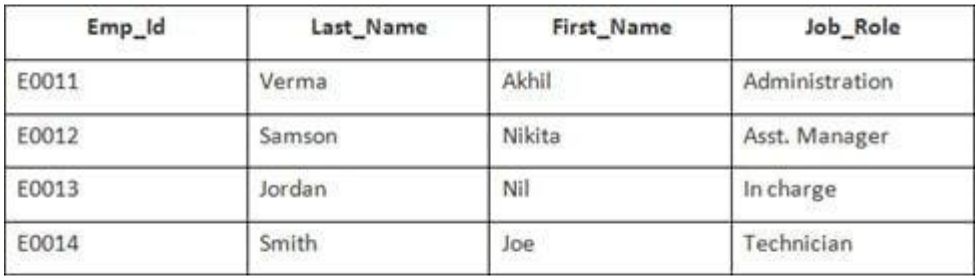
LEFT JOIN (LEFT OUTER JOIN): This join returns all rows from the LEFT table and its matched rows from a RIGHT table**.**

Syntax:

```
SELECTcolumn_name(s)
 FROMtable_name1
 LEFTJOINtable_name2
 ONcolumn_name1=column_name2;
```

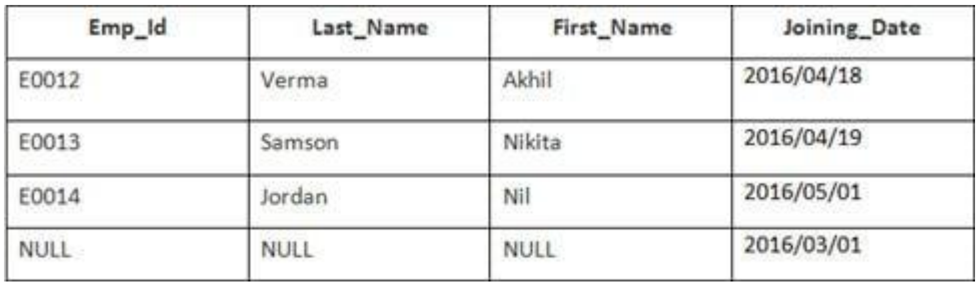
3.

In this example, we have a table **Employee** with the following data:



Emp_Id	Last_Name	First_Name	Job_Role
E0011	Verma	Akhil	Administration
E0012	Samson	Nikita	Asst. Manager
E0013	Jordan	Nil	In charge
E0014	Smith	Joe	Technician

The second table's name is **Joining.**

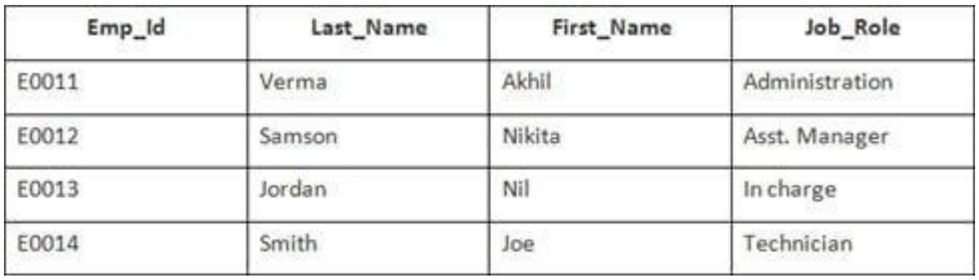


Emp_Id	Last_Name	First_Name	Joining_Date
E0012	Verma	Akhil	2016/04/18
E0013	Samson	Nikita	2016/04/19
E0014	Jordan	Nil	2016/05/01
NULL	NULL	NULL	2016/03/01

Enter the following SQL statement:

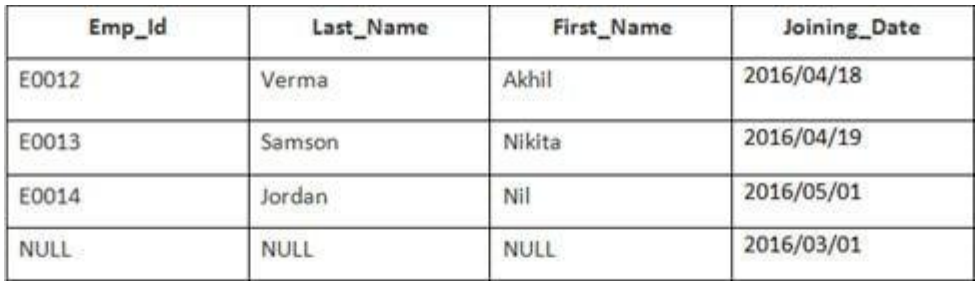
```
SELECTEmployee.Emp_id, Joining.Joining_Date
FROMEmployee
LEFTOUTERJOINJoining
ONEmployee.Emp_id = Joining.Emp_id
ORDERBYEmployee.Emp_id;
```

5. In this example, we have a table **Employee** with the following data:



Emp_Id	Last_Name	First_Name	Job_Role
E0011	Verma	Akhil	Administration
E0012	Samson	Nikita	Asst. Manager
E0013	Jordan	Nil	In charge
E0014	Smith	Joe	Technician

The second table's name is **Joining.**



Emp_Id	Last_Name	First_Name	Joining_Date
E0012	Verma	Akhil	2016/04/18
E0013	Samson	Nikita	2016/04/19
E0014	Jordan	Nil	2016/05/01
NULL	NULL	NULL	2016/03/01

Enter the following SQL statement:

```
SELECTEmployee.Emp_id, Joining.Joining_Date FROMEmployee
RIGHTJOINJoining
ONEmployee.Emp_id = Joining.Emp_id
ORDERBYEmployee.Emp_id;
```

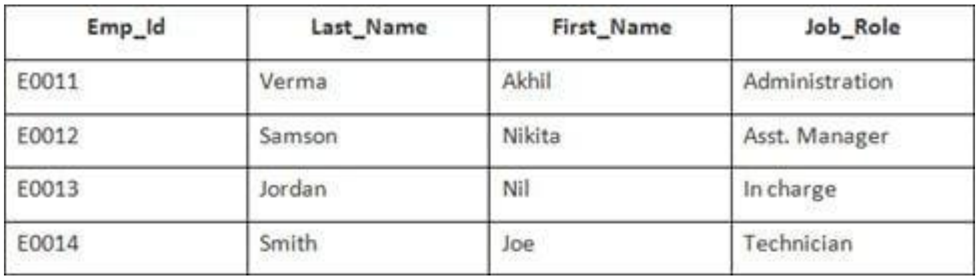
6. Returns all results when there is a match either in the RIGHT table or in the LEFT table**.**

Syntax:

```
SELECTcolumn_name(s)
 FROMtable_name1
 FULLOUTERJOINtable_name2
 ONcolumn_name1=column_name2;
```

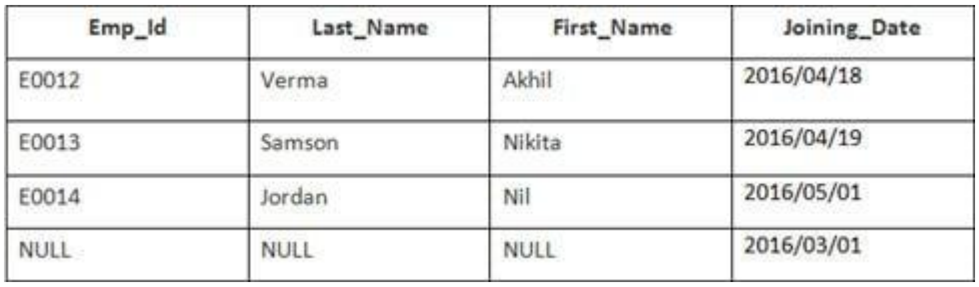
For Example,

In this example, we have a table **Employee** with the following data:



Emp_Id	Last_Name	First_Name	Job_Role
E0011	Verma	Akhil	Administration
E0012	Samson	Nikita	Asst. Manager
E0013	Jordan	Nil	In charge
E0014	Smith	Joe	Technician

The second table’s name is **Joining.**



Emp_Id	Last_Name	First_Name	Joining_Date
E0012	Verma	Akhil	2016/04/18
E0013	Samson	Nikita	2016/04/19
E0014	Jordan	Nil	2016/05/01
NULL	NULL	NULL	2016/03/01

Enter the following SQL statement:

```
SELECTEmployee.Emp_id, Joining.Joining_Date
FROMEmployee
FULLOUTERJOINJoining
ONEmployee.Emp_id = Joining.Emp_id
ORDERBYEmployee.Emp_id;
```

7.To create a new view in the database, you use the CREATE VIEW statement. The basic syntax for creating a view is the following:

CREATE VIEW view_name (view_column_list) AS select_statement;

In this syntax:

· First, specify the name of the view which you want to create after the CREATE VIEW keywords. The column names of the view will automatically derive from the select_statement. However, you can tailor the column names for the view by explicitly declare them in parentheses following the view name. · Second, specify a SELECT statement that retrieves data from columns of one or more tables.

Let’s take some examples of creating new views. We’ll use the books, book_authors, and publishers tables from the sample database for the demonstration.

8. Creating a view based on partial data of a table

See this books table:

The following statement uses the CREATE VIEW statement to create a new view based on the books table that returns all books published since January 2018.

CREATE VIEW new_books AS SELECT title, rating, isbn, published_date

FROM books WHERE published_date > '2018-01-01';

Here is the data returned via the view:

SELECT * FROM new_books

ORDER BY title;

Write the Output:

9) Creating a view based on multiple tables example

This example uses the CREATE TABLE statement to create a view based on the books and publishers tables:

CREATE VIEW book_details AS SELECT

b.title, b.rating, b.isbn, p.name publisher,b.published_date

FROM books b INNER JOIN publishers p

ON p.publisher_id = b.publisher_id;

Code language: SQL (Structured Query Language) (sql)

The following statement returns data from the view:

SELECT * FROM book_details ORDER BY title;

10) Creating a view based on summary data from tables

This statement creates a new view that returns the book title and the number of authors of each book:

```
CREATE VIEW book_author_stats (

book_title, author_count

) AS SELECT title, COUNT(A.author_id)

FROM books b INNER JOIN book_authors a

ON a.book_id = b.book_id

GROUP BY

title;
```

Code language: SQL (Structured Query Language) (sql)

The following query returns the data from the book_author_stats view:

```
SELECT book_title, author_count FROM book_author_stats ORDER BY book_title;
```