

리눅스 강의 전체 노트 필기

[20.12.28 리눅스 강의 Day 1]

■ 리눅스 수업 목차

1. 리눅스를 왜 배워야하는지?
2. 리눅스 설치
3. 리눅스 기본 명령어
4. 리눅스에 아나콘다 설치
5. 리눅스 vi 편집기 명령어
6. 리눅스 권한 관리 명령어
7. 리눅스 디스크 관리 명령어
8. 리눅스 프로세서 관리 명령어
9. 셸스크립트 작성법

1. 리눅스를 배워야 하는 이유?

1. 데이터 분석가가 갖춰야할 기본 기술중에 하둡이 있다.
이 하둡이 리눅스에 기반을 두고 있기 때문에 리눅스를 먼저 배워야 한다.
2. 딥러닝을 구현할때 현업에서 많은 부분이 리눅스에 파이썬을 사용해서 구현을 하고 있음.
3. 빅데이터 기사 시험에 하둡이 나오는데 여러 용어들에 대한 정확한 이해가 있어야 한다.
4. 딥러닝으로 뭔가 좀 재미있는것을 만들고 싶다면 리눅스를 알아야한다.

시연 (동영상링크)

■ 리눅스의 유래

영상 링크 : [세상의 모든 법칙 덕후는 어떻게 탄생하는가?](#)

■ 리눅스란 무엇인가?

유닉스 (unix) 가 너무 고가여서 리눅스 오픈 소스를 핀란드의 리누즈 토발즈 라는 학생이 1991년 11월에 개발했다.

리누즈 토발즈가 리눅스 커널(a.k.a 자동차의 엔진) 을 개발하고 오픈소스로 공개했다.

이를 전 세계의 개발자들이 더 좋게 개선해서 다시 인터넷에 오픈소스로 올리는 작업이 반복되면서 리눅스 os 가 유닉스보다 더 가볍고 안정적이게 되었다.

GNU프로젝트

누구든지 배포된 오픈소스를 가져다 개발할 수 있고, 돈을 벌 목적으로 상용화를 하는 것도 가능한데, 한가지 지켜야할 약속은 이 소스를 가져다가 더 좋게 수정을 했으면 그 코드

레드햇(상용버전의 리눅스) -----> Cent os

↑
유료

↑
무료

■ 리눅스 설치

1. vmware player 설치 -> 가상의 컴퓨터를 내 컴퓨터 안에 만드는 것

<https://hobby.tw/442> 는 학원 컴퓨터(상업용) 이라 그냥 설치하면 불법(개인 노트북은 노상관)

1-1. 그래서 virture box 설치

<https://www.virtualbox.org/wiki/Downloads>

2. Cent Os 다운로드

[http://mirror.anigil.com/CentOS/7.9.2009/isos/x86_64/](http://mirror.anigil.com/CentOS/7.9.2009/isos/x86_64/) 에 들어가서

CentOS-7-x86_64-DVD-2009.iso 를 다운로드

[20.12.29 리눅스 강의 Day 2]

게스트 CD확장 설치

왜 이작업을 해야 하는가??

윈도우 내부에 설치된 것이기 때문에, 마우스와 키보드를 윈도우와 리눅스의 가상 서버를 왔다 갔다 하면 불편한 부분이 있어 이를 좀 더 편하게 이용하기 위해!

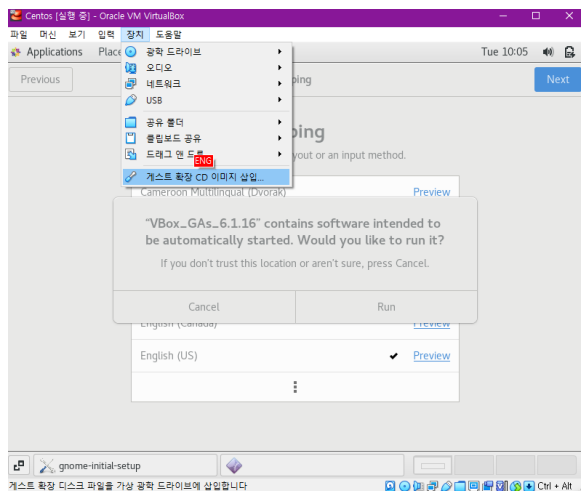
How?



1. root 로 접속한다. (어제 설정한 scott 으로 들어가면 안된다) (root 는 최상위 권한을 가진 계정)
2. 게스트 cd 를 cd롬에 입력한다.
3. 게스트 cd 를 실행한다.
4. scott 하단에 Not listed? 를 클릭한다.
5. Username 에 root 라고 입력 // p.w = 1234 (어제 설치시 설정해둠)

(계정 확인을 위해서 터미널창에서 whoami 라고 치면 나옴)

(pwd 라고 치면 어디 위치에 있는지 알려줌)





1. 버추얼 박스 → 장치 → 게스트 확장 CD이미지 삽입 클릭
2. 리눅스 시스템을 re- booting 한다. → 바탕화면 우클릭, open terminal → reboot + enter

게스트 확장이 잘 적용되어서 sf_data 에 들어가면 연동해둔 e 드라이브의 data 폴더와 연결된다.

3. 바탕화면 우클릭 → open terminal → ifconfig

■ 좌상단의 Applications 클릭 -> firefox 실행 -> 인터넷 연결이 되어 있지 않음을 확인-> 인터넷 연결을 따로 설정 해줘야 함.

■ 리눅스 시스템에서 인터넷 연결하기

Applications(리눅스의 시작버튼) -> system tools -> settings -> Network

Ethernet or Wired (랜선 연결 / 와이파이 차이인듯) 을 on으로 켜다.

랜선연결의 경우 2개가 뜨는데 그중 위에것 을 on으로 하면 됨

Firefox 접속해서 네이버 접속이 잘 되는지 확인!

접속할때마다 인터넷이 자동으로 켜져있지 않다면 우측의 톱니바퀴 -> Connect automatically 체크

이제 수업을 위한 준비 끝~!!

terminal 창에서 python 입력 + enter

파이썬이 접속됨

실행 여부 확인

파이썬이 잘 실행되는 것을 확인할 수 있다.

2. 리눅스 기본 명령어 (현업에서 리눅스 진짜 많이 쓴다.)

001 cd 명령어 (change directory)



Change Directory 의 약어로 디렉토리를 이동하는 명령어

e.g)



whoami : 접속한 주체(나) 가 누구인지 확인하는 명령어

pwd : 현재 내가 있는 디렉토리를 확인하는 명령어 (print working directory) → /root 라고 나옴.

ls : list의 약자. 현재 디렉토리에 있는 폴더와 파일을 확인하는 명령어

폴더 : 파란색, 파일 : 검정색 으로 표시됨.

cd Documents : Documents 디렉토리로 이동하라는 뜻

#pwd 를 해보면 /root/Documents 라고 나옴.

ls 를 해보면 아무것도 나오지 않음 (Documents 안에 아무것도 없어서)

cd .. : 이전 디렉토리로 이동 ★(cd 한칸 띄고 dot dot)

#pwd 해보면 /root 로 나옴

문제 1번. 현재 디렉토리에서 데스크탑 디렉토리로 이동해보시오.

```
#cd Desktop
```

```
#pwd
```

 → 이동이 잘 되었는지 확인.

```
root/Desktop
```

 라고 나옴.

문제 2번. 다시 뒤로 이동하시오.

```
# cd ..
```

```
# pwd
```

문제 3. 나의 집으로 가시오.

1. `# whoami` 로 내가 누군인지 확인 → root
2. `#cd` + enter
3. `#pwd` → 내집이 어딘지 나옴 → /root 라고 나오면 집으로 간 것.

내가 어디에 있는 상관없이 `#cd` + enter 하면 바로 집으로 돌아감

```
결과 : /root
```



* 경로에는 2가지 종류가 있다.

1. 절대경로: 내가 가고자 하는 '정확한 위치' 를 써주는 것

e.g)

```
#pwd
```

 한 상태에서

```
#cd /root/Desktop
```

 → root 밑에 Desktop 으로 가고 싶다. 는 뜻

- > 즉 세세한 경로를 모두 입력 해주는 것

2. 상대경로: 나의 현재위치를 상대로 이동하겠다.

e.g)

```
#cd ..
```

 → 나의 현재위치를 상대로 상위 디렉토리로 이동하겠다. 는 뜻

문제 4. 집으로 돌아간 후에 test01 이라는 폴더를 생성하시오.

```
#cd
```

 + enter → 집으로 돌아가는 명령어

```
#mkdir
```

 → make directory 약어. 디렉토리를 만드는 명령어

```
#mkdir test01
```

 → (현재위치에서) test01 이라는 디렉토리(폴더)를 만들어라! 는 뜻

문제 5. test01 디렉토리 폴더로 이동해 보시오

`#cd test01` 을 입력 → test01 로 change directory 하라는 뜻 → 상대경로로 입력!

or

`# cd/root/test01` → 절대 경로로 입력!

`#pwd` 로 확인 → /root/test01 이라고 나옴.

다시 집으로! `#cd` + enter (항상 문제 풀고 난뒤에는 집으로 돌아오는 습관을 들일 것!)

문제 6. 집으로 돌아가서 test02 라는 폴더(디렉토리) 를 만들고, test02로 이동하십시오.

`#cd` + enter -> 집으로 이동

`#mkdir test02` → test02 라는 디렉토리 생성

`#cd test02` → test02 로 이동!

`#pwd` 로 위치 확인 → /root/test02 라고 나옴.

`#cd` + enter → 다시 집으로!

문제 7. 집으로 돌아가서 test03 디렉토리를 만들고 test03 디렉토리로 가서 test04 디렉토리를 만드시오.

`#cd`

`# mkdir test03`

`# cd test03`

`# mkdir test04`

문제 8. 집에서 test03 밑에 test04로 이동하십시오.

`cd test04` 로는 바로 가지지 않는다.

`cd test03/test04` 로 갈 수 있다.

문제 9. 지금 이동한 디렉토리(test04)에서 방금 이동하기 전 디렉토리(/root)로 바로 이동하십시오.

`# cd -` (cd 한칸 띄고 -)

문제 9 복습

문제 10의 디렉토리 test09 까지 간다음에 한칸씩 뒤로 가보기!



디렉토리상의 순서대로 뒤로가지는 것이 아니라

tv의 이전채널! 기능 처럼 바로 직전에 있던 위치로 돌아 가는 것


그래서 root 에서 바로 test09 까지 갔었다면 중간에 test087/test08 이 있어도

`cd -` 를 입력하면 직전 위치였던 /root로 가는 것!

문제 10. 집에서 test07 directory 를 만들고, test07 디렉토리로 이동한 후에 그 안에 test08 디렉토리를 만들고, test08 디렉토리로 이동해서 test09 디렉토리를 만드시오. 그 후 test09 디렉토리로 이동하시오.

```
# cd
# mkdir test07
# cd test07
# mkdir test08
# cd test07/test08
# mkdir test09
# cd test07/test08/test09
```

002 Touch 명령어

 파일의 용량이 0인 파일을 생성하는 명령어

e.g)

```
#touch a1.txt → a1.txt 라는 파일이 생성된다.
# ls -l a1.txt
```

root 뒤에 0이라고 나오는 부분이 용량

문제 11. 집에서 아래의 파일들을 생성하시오(크기는 0으로!)
a.txt b..txt c..txt f..txt


처음에 하나씩 입력하다가, 혹시나하고 ,로 구분해서 해봤는데 작동하는듯 했으나.. b.txt, 라는 이름으로 생성됨 한번에 만들려면

```
touch a.txt b.txt c.txt ~~ 이런식으로 해야함.
```

```
ls -l *.txt → 확장자가 .txt 인파일들을 모두 표시하라는 뜻
```

크기가 모두 0인 것을 확인할 수 있다.

003 mkdir 명령어 (make directory 의 약자)

 디렉토리를 만드는 명령어

* 매뉴얼 보는 방법

```
#whatis mkdir 입력하면
```

```
#man mkdir 하면 mkdir 에 대한 매뉴얼 내용이 쭉 나옴.
```

빠져나오려면 #q 하면 됨.

문제 12. 집에서 (/root) 아래의 디렉토리들을 생성하시오.
test 30 → test31 → test32 → test 33 → test 34 → test35

각각의 디렉토리 안에 하나씩 디렉토리를 새로 만들 것

```
# cd
# mkdir test30
# cd test30
# mkdir test31
# cd test30/test31
# mkdir test32
# cd test30/test31/test32
# mkdir test33
# cd test30/test31/test32/test33
# mkdir test34
# cd test30/test31/test32/test33/test34
# mkdir test35
```

라고 했는데 한번에 하려면???

```
#mkdir -p /root/test40/test41/test42/test43/test44/test45
```

※ -p 옵션은 디렉토리를 만들면서 한번에 하위 디렉토리를 생성하는 옵션이다.

만약 디렉토리를 지우려면...

```
#rm test30 하면 된다.
```

(점심시간 문제)

문제 13. 집에서 파이썬 스크립트를 따로 관리할 디렉토리를 ptest01 로 생성하고, ptest01 디렉토리로 이동해서 파일의 크기가 0인 example01.txt 파일을 생성하시오.



```
#cd → 집으로 이동
```

```
#mkdir ptest01 → ptest01 이라는 디렉토리 생성
```

```
#ls → 생성되었는지 확인
```

```
#cd ptest01 → ptest01 디렉토리로 이동
```

```
#pwd → 이동이 잘 되었는지 확인
```

```
#touch example01.txt → example01.txt 라는 문서 생성
```

```
#ls → 생성된 목록 확인
```

```
#cd → 집으로 이동
```

004 rm 명령어



파일이나 디렉토리를 삭제하는 명령어

※ 리눅스나 유닉스는 trash라는 휴지통이 있긴 하지만 terminal 에서 삭제하면 휴지통으로 가지 않고 완전삭제가 되기 때문에 주의해야 한다.

e.g)

```
#touch bbb.txt
```

```
#ls -l bbb.txt
```

```
#rm bbb.txt → 지울거냐고 물어보면 y
```

```
#ls -l bbb.txt
```

특히 root 로 작업을 하는 경우 시스템 파일을 실수로 삭제하면 리눅스를 재설치해야 하기때문에 조심해야 한다.

문제 14. 집으로 이동해서 현재 디렉토리에 있는 a.txt, b.txt, c.txt 를 삭제 하시오.

```
rm a.txt b.txt c.txt
ls -l a.txt b.txt c.txt
```

문제 15. 집으로 이동해서 test45 라는 디렉토리를 생성하시오.

```
#cd
#mkdir test45
#ls -ld test45
```

디렉토리 정보를 확인할때 쓰는 명령어

```
ls -ld 디렉토리명
```

drwxr ~~~ 라고 나오는데 맨앞에 d가 있으면 directory 라는 뜻
-rwxr ~~~ 이렇게 나오는 경우는 파일이라는 뜻

문제 16. 집으로 이동해서 지금 방금 만든 test45 디렉토리를 삭제하시오'

```
#cd
#rm -r test45
```

-r 옵션은 디렉토리를 삭제할때 사용하는 옵션이다.

디렉토리 삭제시 삭제할 디렉토리 밑에 있는 파일과 하위 디렉토리를 모두 삭제하는 옵션이다.

문제 17. 집으로가서 test03 디렉토리 밑에 하위 디렉토리가 있는지 확인하시오.

```
#cd
#ls → test04가 있다.
```

문제 18. 집으로가서 /root 밑에 있는 test03 디렉토리와 그 하위 디렉토리를 모두 삭제하시오.

```
💡 [root@localhost test03] # cd
[root@localhost ~]# rm -r test03
[root@localhost ~]# rm: descend into directory 'test03'? y
[root@localhost ~]# rm: remove directory 'test03/test04'? y
[root@localhost ~]# rm: remove directory 'test03'? y
```



```
[root@localhost ~]# ls
a1.txt      Documents  e.txt      Music      Public     test01    test30
anaconda-ks.cfg Downloads  f.txt      Pictures   root       test02    test40
Desktop     d.txt     initial-setup-ks.cfg ptest01    Templates  test07    Videos
```

[20.12.30 리눅스 강의 Day 3]

데이터를 만들때 외부에서 조립된 소스들을 가져와서 필요에 맞게 응용/조정 하는 경우가 많은데, 대부분의 코드들이 리눅스에서 구현된 것들이 많아서 리눅스를 잘 알아두는 것이 좋다.

012. grep 명령어



파일 안의 포함된 특정 단어나 구문을 검색하는 명령어

문제 39. 이름이 allen 인 사원의 이름과 월급을 출력하시오.

```
[root@localhost ~]# grep -i 'allen' emp.txt | awk '{print $4,$2}'
7698 ALLEN
```

해석

grep : 후술하는 파일에서 ' ' 안에 포함된 특정 단어나 구문을 검색하겠다.

-i 'allen' : -i 는 대소문자를 구분하지 않겠다는 의미

emp.txt | awk '{print \$4,\$2}' : awk 는 '열'을 의미하는 명령어, \$에 숫자를 붙여 열의 순서를 지정한다.

터미널창에 복사/붙여넣기 할때

shift + insert

shift + control + c / v

문제 40. 월급이 3000인 사원의 이름과 월급을 출력하시오.

1. 월급이 3000인 (정확히는 3000이라는 수치를 갖고 있는 행을 전부) 출력

```
[root@localhost ~]# grep '3000' emp.txt
```

```
7902    FORD    ANALYST    7566    1981-12-11    3000    0    20
7788    SCOTT    ANALYST    7566    1982-12-22    3000    0    20
```

오답 정확하게 '월급'이 3000인 사람에 대한 데이터를 뽑으려면 awk 를 써야 한다.

```
[root@localhost ~]# grep '3000' emp.txt | awk '{print $2, $4}'
```

```
FORD 7566
SCOTT 7566
```

위의 emp.txt 만 이용하면 grep 만으로도 문제 40을 출력할 수 있지만, 월급이 3000인 데이터를 정확하게 검색하려면 awk 를 이용해야 한다.

만약 comm등에 3000이 있으면 grep는 어떤 열이든 3000이란 숫자를 가진 데이터가 속해 있는 열을 모두 출력하기 때문에, grep 만을 이용하지 않고, awk를 함께 사용해 정확한 데이터를 출력해야 한다.

실험을 위해 ALLEN의 comm을 3000으로 수정함

```
(base) [root@localhost ~]# grep '3000' emp2.txt
```


7499	ALLEN	SALESMAN	7698	1981-02-11	1600	3000	30
7902	FORD	ANALYST	7566	1981-12-11	3000	0	20
7788	SCOTT	ANALYST	7566	1982-12-22	3000	0	20

sal 이 3000이 아닌 ALLEN도 함께 검색됨

```
(base) [root@localhost ~]# grep '3000' emp2.txt | awk '{print $2,$6}'
```

```
ALLEN 1600
FORD 3000
SCOTT 3000
```

awk로 이름, 월급을 추려내어도 grep에서 필터링이 된 ALLEN은 여전히 출력이 된다.

답  (awk로 출력을 원하는 열을 먼저 뽑아낸 후에, grep을 적용해서 원하는 기준을 적용시킨다)

```
(base) [root@localhost ~]# awk '{print $2,$6}' emp2.txt | grep '3000'
```

```
FORD 3000
SCOTT 3000
```

grep을 사용했을때의 단점

문제 41. 부서번호가 10번인 사원들의 이름, 월급, 부서번호를 출력하시오.

```
[root@localhost ~]# grep -i '10' emp.txt | awk '{print $2, $6, $8}'
```

```
KING 5000 10
CLARK 2450 10
MARTIN 1250 30
ADAMS 1100 20 -> 은 1100 중 가운데 10이 포함되어져 있어서 함께 출력된 것
MILLER 1300 10
```

```
[root@localhost ~]# grep -iw '10' emp.txt | awk '{print $2, $6, $8}'
```

```
KING 5000 10  
CLARK 2450 10  
MARTIN 1250 30  
MILLER 1300 10
```



* iw 는 'word' 단위로 grep을 수행한다는 뜻

* 근데 MARTIN 의 입사일이 10일이어서 MARTIN 이 함께 나온다.

보완!

awk 로 원하는 컬럼을 먼저 걸러내고, 그 후에 10을 찾으면 된다!

```
[root@localhost ~]# awk '{print $2,$6,$8}' emp.txt | grep -iw '10'
```

```
KING 5000 10  
CLARK 2450 10  
MILLER 1300 10
```



* awk 로 먼저 emp.txt 에서 이름, 월급, 부서번호를 선택해서 출력하고, 그 출력된 결과에서 10을 포함하고 있는행들만 출력한 것.

grep는 emp.txt에서 특정 글자가 포함되어져 있는 행들만 검색하면 되기때문에 특정 열에 대해서 데이터를 검색

하는 것은 grep 만으로는 한계가 있다.

★ 그래서 awk 명령어를 함께 사용해주야 한다.

013. awk 명령어



특정 컬럼(열)을 출력하고자 할때 사용하는 명령어

e.g)

```
#awk '패턴 {action}' 대상 파일명
```

```
#awk '$3 == "SALESMAN" {print $2, $3}' * ' ' 과 "" 의 위치 주의!
```



* 설명 : 직업이 SALESMAN인 사원의 이름과 직업을 출력하는 명령어

* 리눅스 연산자 3가지 정리

1. 산술연산자 : +, -, *, /
2. 비교연산자 : >, <, >=, <=, ==, !=
3. 논리 연산자 : &&(=and) , || (= or), !

문제 42. 월급이 3000이상인 사원들의 이름과 월급을 출력하시오.

답

```
[root@localhost ~]# awk '$6>=3000 {print $2, $6}' emp.txt
```

해석 :



1) awk '\$6>=3000' : 월급이 3000이상인 열을 추려내고

2){print \$2, \$6}' emp.txt : 추려낸 값이 있는 열이 위치한 행에서 \$2(이름), \$6(월급) 을 출력하라

결과

```
KING 5000
FORD 3000
SCOTT 3000
```

문제 43. 직업이 salesman 이 아닌 사원들의 이름과 월급을 출력하시오.

답

```
[root@localhost ~]# awk '$3 !=toupper("salesman") {print $2,$3}' emp.txt
```



* toupper() : python 의 upper() 와 같은 기능 단 괄호 속 단어에 "" 를 반드시 돌려줘야 한다.

결과

```
KING PRESIDENT
BLAKE MANAGER
CLARK MANAGER
JONES MANAGER
JAMES CLERK
FORD ANALYST
SMITH CLERK
SCOTT ANALYST
ADAMS CLERK
MILLER CLERK
```

문제 44. 직업이 salesman 이고 월급이 1500이상인 직원들의 이름과 월급, 직업을 출력하시오.

답

```
[root@localhost ~]# awk '$3 ==toupper("salesman") && $6>=1500 {print $2,$6,$3}' emp.txt
```



* 두개의 조건을 필터링할때는 && 을 써준다.

결과

```
ALLEN 1600 SALESMAN
TURNER 1500 SALESMAN
```

문제 45. 81년도에 입사한 직원들의 이름과 입사일을 출력하시오.



substrt 같은 기능을 사용하면 간편하게 할 수 있다.

man awk 로 매뉴얼을 살펴보면..

1. man awk + enter
2. /substr <-현재 보고있는 페이지에서 substr을 검색하는 명령어
3. n 을 누르면 next 을 누르면서 원하는 명령어 찾기

substr(s, i [, n]) Return the at most n-character substring of s starting at i. If n is omitted, use the rest of s.

'i'에서 시작하는 's'의 하위 문자열의 최대 개수를 반환합니다. 'n'이 생략된 경우 나머지 's'를 사용합니다.

substr(추출할 컬럼, 출력 시작점, 출력 글자수(시작점 포함))

답

```
[root@localhost ~]# awk 'substr($5,3,2)=="81" {print$2,$5}' emp.txt
```

substr(\$5,3,2)=="81" 은 \$5 = 입사일 에서 3번째글자를 포함해 우측으로 2글자를 잘라내라!

결과

```
KING 1981-11-17
BLAKE 1981-05-01
CLARK 1981-05-09
JONES 1981-04-01
MARTIN 1981-09-10
ALLEN 1981-02-11
TURNER 1981-08-21
JAMES 1981-12-11
WARD 1981-02-23
FORD 1981-12-11
```

문제 46. 12월에 입사한 직원들의 이름과 입사일을 출력하시오.

답

```
[root@localhost ~]# awk 'substr($5,6,2)=="12" {print $2,$5}' emp.txt
```

결과 

```
JAMES 1981-12-11
FORD 1981-12-11
SMITH 1980-12-09
SCOTT 1982-12-22
```

문제 47. 81년도에 입사하지 않은 직원들의 이름, 입사일을 출력하시오.

답 

```
[root@localhost ~]# awk 'substr($5,3,2)!="81" {print$2,$5}' emp.txt
```

결과 

```
SMITH 1980-12-09
SCOTT 1982-12-22
ADAMS 1983-01-15
MILLER 1982-01-11
```

문제 48 이름의 첫글자가 'A'로 시작하는' 직원들의 이름과 월급을 출력하시오

답 

```
root@localhost ~]# awk '{print $2,$6}' emp.txt | grep -i '^a'
```



* ^는 시작을 의미!

* 정규표현식 : ^ → 시작, \$ → 끝

결과 

```
ALLEN 1600
ADAMS 1100
```

문제 49. 이름의 끝글자가 T 로 끝나는 직원들의 이름, 월급을 출력하시오.

답 

```
[root@localhost ~]# awk '{print $2,$6}' emp.txt | grep -i '$t'
```

결과 

```
MARTIN 1250
TURNER 1500
```

```
SMITH 800  
SCOTT 3000
```

로 T 만포함되면 다 나옴.

답 

```
[root@localhost ~]# awk '{print $6,$2}' emp.txt | grep -i 't$'
```

결과 

```
3000 SCOTT
```



* 설명 : 월급과 이름을 먼저 출력하고(awk~ .txt) , 그 결과에서 문자 끝에 t가 포함되는 값을 출력한 것.

* ^ 를 쓸때는 조건이 되는 문자열의 앞에, \$를 쓸때는 조건이 되는 문자열의 뒤에 붙여줘야 한다!

문제 50. 위의 문제를 awk와 substr을 이용해서 수행하세요.

답 

```
[root@localhost ~]# awk 'substr($2,5,1) == "T" {print $2, $6}' emp.txt
```

결과 

```
SCOTT 3000
```

문제 51. 이름의 2번째 철자가 m 인 직원들의 이름, 월급을 뽑으시오.

답 

```
[root@localhost ~]# awk 'substr($2,2,1) == "M" {print $2, $6}' emp.txt
```

결과 

```
SMITH 800
```

답 

```
[root@localhost ~]# awk 'substr($2,2,1) == toupper("m") {print $2, $6}' emp.txt
```

결과 

```
SMITH 800
```

014 sort 명령어



data 를 특정 컬럼을 기준으로 정렬하는 명령어 (SQL 의 order by 같은 것)
#sort 옵션 파일명

```
#sort -nk 6 emp.txt
```

```
[root@localhost ~]# sort -nk 6 emp.txt
```

결과

7369	SMITH	CLERK	7902	1980-12-09	800	0	20
7900	JAMES	CLERK	7698	1981-12-11	950	0	30
7876	ADAMS	CLERK	7788	1983-01-15	1100	0	20
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7934	MILLER	CLERK	7782	1982-01-11	1300	0	10
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7782	CLARK	MANAGER	7839	1981-05-09	2450	0	10
7698	BLAKE	MANAGER	7839	1981-05-01	2850	0	30
7566	JONES	MANAGER	7839	1981-04-01	2975	0	20
7788	SCOTT	ANALYST	7566	1982-12-22	3000	0	20
7902	FORD	ANALYST	7566	1981-12-11	3000	0	20
7839	KING	PRESIDENT	0	1981-11-17	5000	0	10



* -n 옵션을 쓰면 숫자로 변환해서 정렬을 해준다

- k를 사용하고 6을 쓰면 6번째 컬럼을 기준으로 정렬하겠다는 뜻

(6=\$6 이 되는 것)

즉, -nk 는 6번째 컬럼을 숫자로 인식해서 기준으로 정렬하겠다.

```
[root@localhost ~]# sort -nrk 6 emp.txt -> 월급이 높은 순으로!
```

7839	KING	PRESIDENT	0	1981-11-17	5000	0	10
7902	FORD	ANALYST	7566	1981-12-11	3000	0	20
7788	SCOTT	ANALYST	7566	1982-12-22	3000	0	20
7566	JONES	MANAGER	7839	1981-04-01	2975	0	20
7698	BLAKE	MANAGER	7839	1981-05-01	2850	0	30
7782	CLARK	MANAGER	7839	1981-05-09	2450	0	10
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7934	MILLER	CLERK	7782	1982-01-11	1300	0	10
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7876	ADAMS	CLERK	7788	1983-01-15	1100	0	20
7900	JAMES	CLERK	7698	1981-12-11	950	0	30
7369	SMITH	CLERK	7902	1980-12-09	800	0	20



-r 옵션은 reverse 의 약자! (desc 같은 것)

문제 52. 이름, 월급을 출력하는데 월급이 높은 사원부터 출력하시오.

답


```
[root@localhost ~]# sort -nrk 6 emp.txt | awk '{print $2,$6}'
```

결과 

```
KING 5000
FORD 3000
SCOTT 3000
JONES 2975
BLAKE 2850
CLARK 2450
ALLEN 1600
TURNER 1500
MILLER 1300
MARTIN 1250
WARD 1250
ADAMS 1100
JAMES 950
SMITH 800
```

문제 53. 직업이 SALESMAN인 직원들의 이름, 월급, 직업을 출력하는데, 월급이 높은 직원부터 출력하시오.

답 

```
[root@localhost ~]# awk '$3==toupper("salesman") {print $2,$6,$3}' emp.txt | sort -nrk 2
```

결과 

```
ALLEN 1600 SALESMAN
TURNER 1500 SALESMAN
WARD 1250 SALESMAN
MARTIN 1250 SALESMAN
```



* 먼저 이름, 월급, 직업을 출력해서 그 결과를 sort 하는 거니까, 직업은 전체 emp일때의 6번째가 아니라

조건부로 출력한 awk문(직/call업이 salesman인 데이터의 이름(순번1), 월급(순번2), 직업(순번3)) 에서의 결과인 2번째가 월급이 있는 부분이 된다.

그래서 -nrk 6이 아닌 -nrk 2 를 쓴다.

이미 한번 걸러낸 값에서 6번째 k는 없는 자료.

문제 54. 직업이 SALESMAN이 아닌 직원들의 이름, 월급, 직업을 출력하는데 월급이 낮은 직원부터 높은 직원순으로 출력하시오.

답 

```
awk '$3 !=toupper("salesman") {print $2,$6,$3}' emp.txt | sort -nk 2
```

결과 

```
SMITH 800 CLERK
JAMES 950 CLERK
ADAMS 1100 CLERK
MILLER 1300 CLERK
```

```
CLARK 2450 MANAGER
BLAKE 2850 MANAGER
JONES 2975 MANAGER
FORD 3000 ANALYST
SCOTT 3000 ANALYST
KING 5000 PRESIDENT
```

★ 문제 55. 사원 테이블의 월급의 토탈값을 출력하시오.

```
[root@localhost ~]# awk '{print$6}' emp.txt | awk '{sum += $1} END {print sum}'
```

29025



* 설명 : 월급만 추출해서 그 출력값을 파이프 (|) 다음에 오는 명령어의 입력값으로 보낸다.

sum += \$1 이 실행되면서 월급이 계속 누적된다. | 이전에 입력한 기준에 해당 하는 모든 값들이 sum에 + 되어 누적되면
END 로 종료가되고, 누적된 sum을 프린트 한다.

(sum은 함수명이 아닌 편의상 만든 이름)

문제 56. 직업이 SALESMAN인 사원들의 토탈월급을 출력하시오.

답

```
[root@localhost ~]# grep -i 'salesman' emp.txt | awk '{print$6}' | awk '{sum += $1} END {print sum}'
```

결과

5600



* 설명: grep 으로 'salesman' 이란 문자열을 가진 데이터들만 걸러낸 후에

그중에서 %6(월급) 에 해당 하는 값들만 걸러내서 맨뒤의 awk 실행문으로 보낸다!

거기서 \$1(=\$6=월급)을 sum이라는 일종의 빈 값(?)에 for 문 돌리듯이 하나씩 누적시켜라

1) 직업이 SALESMAN 인 자료들만 걸러냄

```
# grep -i 'salesman' emp.txt
```

7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30

2) `awk '{print$6}'` 1) 에서 걸러낸 자료들 중에서 6번째 컬럼 (=월급) 만 따로 걸러냄

```
[root@localhost ~]# grep -i 'salesman' emp.txt | awk '{print$6}'
```

```
1250
1600
1500
1250
```

→ 행이 6개에서 1개로 줄었음. (\$6 → \$1 이 된 것)

3) `awk '{sum += $1} END {print sum}'` 을 입력해서 위의 4개의 값을 1개씩 sum이라는 곳에 누적시킨다.

4개의 값을 다 sum에 입력하면서 + 하면 END 하고 sum을 print 한다.

```
# awk '$3==toupper("salesman"){print $6}' emp.txt | awk '{sum+= $1} END {print sum}'
```

```
5600
```



이렇게 풀어도 됨!

문제 57. 직업이 SALESMAN 이 아니고 월급이 2000 이상인 직원들의 이름과 월급과 직업을 출력하시오.

답

```
[root@localhost ~]# awk '$3!=toupper("salesman") && $6>=2000 {print $2,$6,$3}' emp.txt
```

결과

```
KING 5000 PRESIDENT
BLAKE 2850 MANAGER
CLARK 2450 MANAGER
JONES 2975 MANAGER
FORD 3000 ANALYST
SCOTT 3000 ANALYST
```

문제 58. 위의 직원들의 토탈월급을 출력하시오.

답

```
[root@localhost ~]# awk '$3!=toupper("salesman") && $6>=2000 {print $2,$6,$3}' emp.txt | awk '{sum+= $2} END {print sum}'
```

결과

```
19275
```

(점심시간 문제)

문제 59. 81년도에 입사한 직원들의 월급의 토탈값을 출력하시오.

답

```
[root@localhost ~]# awk 'substr($5,3,2)=="81" {print $6}' emp.txt | awk '{sum+=$1} END {print sum}'
```

결과

```
22825
```

풀이 과정

1) 입사일이 81년인 직원들의 월급만 출력

```
[root@localhost ~]# awk 'substr($5,3,2)=="81" {print $6}' emp.txt
```

```
5000
2850
2450
2975
1250
1600
1500
950
1250
3000
```

2) `awk '{sum+=$1} END {print sum}'` → 그중에 \$1(즉 월급) 을 sum에 누적시켜서 더한 후 출력

문제 60. 아래의 SQL 의 결과를 리눅스 명령어로 출력하시오

답

```
[root@localhost ~]# awk '{print $2 "의 월급은" $6 "입니다." }' emp.txt
```

결과

```
KING의 월급은5000입니다.
BLAKE의 월급은2850입니다.
CLARK의 월급은2450입니다.
JONES의 월급은2975입니다.
MARTIN의 월급은1250입니다.
ALLEN의 월급은1600입니다.
TURNER의 월급은1500입니다.
JAMES의 월급은950입니다.
WARD의 월급은1250입니다.
FORD의 월급은3000입니다.
SMITH의 월급은800입니다.
SCOTT의 월급은3000입니다.
ADAMS의 월급은1100입니다.
MILLER의 월급은1300입니다.
```

문제 61. 아래의 SQL 결과를 리눅스 명령어로 출력하시오.

```
SQL> select ename || '의 직업은' || job || '입니다.'
      from emp;
```

답 


```
[root@localhost ~]# awk '{print $2 "의 직업은" $3 "입니다." }' emp.txt
```

결과 

```
KING의 직업은PRESIDENT입니다.
BLAKE의 직업은MANAGER입니다.
CLARK의 직업은MANAGER입니다.
JONES의 직업은MANAGER입니다.
MARTIN의 직업은SALESMAN입니다.
ALLEN의 직업은SALESMAN입니다.
TURNER의 직업은SALESMAN입니다.
JAMES의 직업은CLERK입니다.
WARD의 직업은SALESMAN입니다.
FORD의 직업은ANALYST입니다.
SMITH의 직업은CLERK입니다.
SCOTT의 직업은ANALYST입니다.
ADAMS의 직업은CLERK입니다.
MILLER의 직업은CLERK입니다.
```

문제 62. 집에서(/root) 에서 ls -l로 수행한 결과를 출력하시오

```
[root@localhost ~]# cd
[root@localhost ~]# ls -l
```

결과  에서 빨간색 부분이 파일의 크기를 의미함

```
total 52
```

```
-rw-r--r--. 1 root root 0 Dec 29 11:47 a1.txt
-rw----- 1 root root 1604 Dec 28 16:41 anaconda-ks.cfg
.
.
.
```

문제 63. 위의 출력된 결과에서 파일의 크기에 해당하는 부분만 출력하시오.

답 

```
[root@localhost ~]# ls -l | awk '{print $5}'
```

결과 

```
0
1604
37
6
6
.
.
.
564
12176
```

문제 64. 위에서 출력된 숫자들의 총합을 출력하시오.

답

```
[root@localhost ~]# ls -l | awk '{print $5}' | awk '{sum+=$1} END {print sum}'
```

결과

```
32427 -> byte 임
```



* 참고 ! (빅분기에도 나옴) [킬>메>기>테>페>엑>제>요]

기호(이름)	값	기호	값	기호(이름) <-이진표기	
kB (킬로바이트)	$1000^1 = 10^3$	KB	$1024^1 = 2^{10}$	KiB (키비바이트)	210
MB (메가바이트)	$1000^2 = 10^6$	MB	$1024^2 = 2^{20}$	MiB (메비바이트)	220
GB (기가바이트)	$1000^3 = 10^9$	GB	$1024^3 = 2^{30}$	GiB (기비바이트)	230
TB (테라바이트)	$1000^4 = 10^{12}$	TB	$1024^4 = 2^{40}$	TiB (테비바이트)	240
PB (페타바이트)	$1000^5 = 10^{15}$	PB	$1024^5 = 2^{50}$	PiB (페비바이트)	250
EB (엑사바이트)	$1000^6 = 10^{18}$	EB	$1024^6 = 2^{60}$	EiB (엑스비바이트)	260
ZB (제타바이트)	$1000^7 = 10^{21}$	ZB	$1024^7 = 2^{70}$	ZiB (제비바이트)	270
YB (요타바이트)	$1000^8 = 10^{24}$	YB	$1024^8 = 2^{80}$	YiB (요비바이트)	280

015 unique 명령어



중복된 라인을 제거하는 명령어



문법 : #uniq 옵션 파일명

문제 65. emp.txt에서 직업만 출력하시오.

```
[root@localhost ~]# awk '{print $3}' emp.txt
```

```
PRESIDENT  
MANAGER  
MANAGER  
MANAGER  
SALESMAN
```

```
SALESMAN
SALESMAN
CLERK
SALESMAN
ANALYST
CLERK
ANALYST
CLERK
CLERK
```

문제 66. 위의 결과를 abcd 순으로 정렬하시오.

```
[root@localhost ~]# awk '{print $3}' emp.txt | sort
```

```
ANALYST
ANALYST
CLERK
CLERK
CLERK
CLERK
MANAGER
MANAGER
MANAGER
PRESIDENT
SALESMAN
SALESMAN
SALESMAN
SALESMAN
```

```
# awk '{print $3}' emp.txt | sort ("-nk 1" ◀ 생략된 부분)
```



아까는 문자열을 정렬해야 하고 몇번째 행을 정렬할지 입력하기 위해 nk 1 이라고 적었지만

지금은 딱 1개의 열만 있어서 숫자 생략

문제67. 위의 결과를 중복제거해서 출력하시오.

```
[root@localhost ~]# awk '{print $3}' emp.txt | sort | uniq
```

```
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN
```

* 설명 : uniq 사용하기 전에 앞에서 정렬을 한 이유는 uniq는 위아래가 같은 데이터만 중복을 제거하기때문
(증명)

문제 68. 사원 테이블에서 부서번호를 출력하는데 중복을 제거해서 출력하시오.

```
(base) [root@localhost ~]# awk '{print $8}' emp.txt | sort -nk 1
```

```
10
10

10
20
20
20
20
20
20
30
30
30
30
30
30
```

위와 같이 나온 값에서 위아래로 비교해서 같은 파일들은 다 지움

```
[root@localhost ~]# awk '{print $8}' emp.txt | sort -nk 1 | uniq
```

```
10
20
30
```

문제 69. 직업이 CLERK 인 직원들의 부서번호를 출력하는데 중복 제거해서 출력하시오.

```
[root@localhost ~]# awk '$3==toupper("clerk") {print $8}' emp.txt | sort -nk 1 | uniq
```

```
10
20
30
```

💡 * 문제 67에서는 sort k 1을 쓰고 여기선 sort -nk 1을 쓴이유?

n = number 을 의미,

- k 는 디폴트!

부서번호는 숫자형 데이터 이기 때문

016 echo 명령어

💡 출력하고자 하는 글자를 출력할 때 사용하는 명령어

파이썬의 print 와 같은 명령어

변수안의 글자를 출력할 때 사용하는 명령어

```
[root@localhost ~]# a=1
[root@localhost ~]# echo $a
```



```

1
[root@localhost ~]# b='scott'

[root@localhost ~]# echo $b  -> 변수명 앞에 $를 써야 b에 변수로 지정한 값이 나온다

scott

[root@localhost ~]# echo b

b

```

* 설명 : 변수안에 있는 값을 출력할때는 \$를 변수명 앞에 붙여줘야 한다.

문제 70. 직업을 물어보게 하고 직업을 입력하면 해당 직업을 갖는 직원들의 이름과 직업이 출력되게 하시오.

```

echo -n '직업명을 입력하세요-'
read job
grep -i $job emp.txt | awk '{print $2,$3}'

```



echo 뒤의 -n 은 무엇? 일종의 enter 역할을 하는 것

echo 'enter the job title~' 를 하면

```
[root@localhost ~]# echo 'Enter the job title~'
```

Enter the job title~

이라고 문자열 자체가 그냥 출력되고 끝남.

```
[root@localhost ~]# echo -n 'Enter the job title~' 라고 하면
```

Enter the job title~ salesman 이 이하에 쓴 특정 값이 밑의 명령어에 입력되는 것

read job → 위에서 입력한 salesman 을 read 하는 것.

+

\$를 job 앞에 써줘야 job을 변수로 인식한다.

\$를 쓰지 않으면 그냥 영어철자 job 일 뿐이다.

sort -n 과 echo -n 은 서로 다른 기능이다.

- n 들은 서로 다른 옵션들의 축약된 명령어일뿐

sort -n 에서 n=number

echo -n에서 n=enter 이다.

★ 위의 리눅스 명령어 여러 개를 파일로 저장해서 한번에 리눅스 터미널 창에서 수행하는 방법

1. #vi job2.sh 를 입력 + enter (vi 편집기 화면으로 들어가는 명령어)

2. #vi job2.sh 밑에 ~~라는 표시와 함께 공간이 생긴다. 여기에 'i'를 입력하면 INSERT 라는 표시가 뜬다
3. 이때부터 입력이 가능해진다. 이때 shift + ctrl + v 해서 메모장에 적은 내용 붙여넣기
4. 저장하려면 esc 키를 2번 눌러서 아래쪽에 insert 가 사라지게 한다.
5. 대문자 Z 를 누르면 저장이 된 후에 자동으로 vi 편집기 화면을 빠져나온다.
6. sh job2.sh 를 입력해 위에서 입력한 명령문을 실행한다.

```
[root@localhost ~]# vi job2.sh ->vi 편집기 열기
[root@localhost ~]# sh job2.sh -> vi 편집기로 입력된 명령문 실행하기
```

결과 

```
Enter the job title salesman
MARTIN SALESMAN
ALLEN SALESMAN
TURNER SALESMAN
WARD SALESMAN
```



sh job2.sh 에서 sh = shell script

shell script 는 리눅스 명령어를 모아 놓은 것이다.

이 shell script를 잘 작성할 줄 알면 업무의 많은 부분을 자동화 할 수 있다.

실무에서는 단순한 업무들은 자동화를 시켜두고, 좀 더 심도 깊은 데이터 분석이나, 시각화 등에 집중할 수 있음!

물론 보고서 작성 등도 딥러닝 등으로 충분히 자동화할 수 있음.

sh 명령어로 shell script 를 실행한다.

문제 76. 아래와 같이 부서번호를 물어보게 하고 부서번호를 입력하면 해당 부서번호인 직원들의 이름, 월급, 부서번호가 출력되게 하시오.

\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

```
echo -n 'Enter depart number-'
read deptno

awk -v num=$deptno '$8==num {print $2, $6, $8}' emp.txt

[root@localhost ~]# vi deptno.sh
[root@localhost ~]# sh deptno.sh
```

```
Enter depart number-10
KING 5000 10
```

```
CLARK 2450 10
MILLER 1300 10
```



awk '\$8==\$deptno '{print \$2, \$6,\$8}' emp.txt 가 될 것이라 생각했으나 안됨

왜냐하면 == 를 기준으로 비교할 2개의 대상 중 한쪽에만 \$가 있어야 함.

```
awk -v num=$deptno '$8==num {print $2, $6, $8}' emp.txt
```

그래서 일단 \$deptno 라는 함수를 num에 할당

그리고 그 num은 \$8 이라는 함수가 되도록 한 것

* 설명: awk 의 '-v 옵션'은 변수의 값을 다른 변수에 할당(assign)할 때 사용하는 옵션이다.



vi 편집기 명령어들 (<https://coding-factory.tistory.com/505>)(<https://coding-factory.tistory.com/505>))

만약 이미 입력을 하고 ZZ까지해서 다 저장하고 빠져나왔었는데 명령문을 수정하고 싶으면?

1. 다시 vi 편집모드로 해서 수정할 .sh 파일에 들어간다.
2. cc 입력 → 수정이 가능해진다.
3. 자유롭게 수정하고 저장한뒤 닫는다.

한번에 지우기는 dd

vi명령어 취소는 u

"

문제 72. 이름이 SCOTT인 사원의 이름, 월급을 출력하시오

\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

```
awk '$2 ==toupper("scott") {print $2, $6}' emp.txt
```

답

```
[root@localhost ~]# awk '$2 ==toupper("scott") {print $2, $6}' emp.txt
```

결과

```
SCOTT 3000
```

문제 73. 이름을 물어보게 하고, 이름을 입력하면 해당 사원의 이름과 월급이 출력되게 shell script를 만들고 아래와 같이 실행되게 하시오

```
sh find_sal.sh

Enter the ename ~ scott

SCOTT 3000
```

답 

```
echo -n 'Enter the ename ~~~'
read ename

awk -v str=$ename '$2==toupper(str) {print $2, $6}' emp.txt
```

결과 

```
[root@localhost ~]# vi find_sal.sh
[root@localhost ~]# sh find_sal.sh

Enter the ename ~~~scott

SCOTT 3000
```

문제 74. dept.txt 를 리눅스 서버에 올리시오~
드래그 & 드랍 해서 /root 에 두면 된다.

확인

```
[root@localhost ~]# ls -l dept.txt
-rw-rw-rw-. 1 root root 105 Dec 29 14:16 dept.txt
```

이동이 잘 된것을 확인할 수 있다.

문제 75. dept.txt 에서 부서번호 10번의 부서위치가 어딘지 출력하시오.
\$1 = deptno \$2=부서명 \$3=지역

답 

```
[root@localhost ~]# awk '$1==10 {print $3}' dept.txt
```

결과 

```
NEWYORK
```

문제 76. SCOTT이 근무하는 부서위치를 출력하시오. (emp.txt 와 dept.txt 에서 각각 자료를 출력해서 매칭 시켜야함)
\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호
\$1 = deptno \$2=부서명 \$3=지역

```
SQL> select d.loc
      from emp e, dept d
      where e.deptno = d.deptno and e.ename = 'SCOTT';
```

1) scott이 일하는 부서의 번호가 무엇인지 출력

답 

```
[root@localhost ~]# awk '$2==toupper("scott") {print $8}' emp.txt
```

결과 

```
20
```

2) scott의 부서번호를 이어지는 명령어의 입력값이 되게 한다.

```
deptno=`awk '$2==toupper("scott") {print $8}' emp.txt`
awk -v num=$deptno '$1==num {print $3}' dept.txt
```



역따옴표 (`) 는 ` ` 로 둘러진 것 안에있는 명령어로 수행된 결과를 deptno 라는 변수에 할당하겠다 라는 뜻.

그렇지 않으면 그냥 명령문이 하나의 문장으로서 통째로 들어간다.

문제 77. 위의 스크립트를 이용해서 shell script를 생성해서 아래와 같이 실행되게 하시오.

```
echo -n 'Enter the ename~~'
read ename

deptno=`awk -v num1=$ename '$2==toupper(num1) {print $8}' emp.txt`
awk -v num=$deptno '$1==num {print $3}' dept.txt
```

답 

```
[root@localhost ~]# vi find_loc.sh
[root@localhost ~]# sh find_loc.sh
```

결과 

```
Enter the ename~~scott
DALLAS
```

문제 78. 직업이 SALESMAN 인 사람들의 월급을 출력하시오.

답 

```
[root@localhost ~]# awk '$3=toupper("salesman") {print$6}' emp.txt
```

결과 

```
1250
1600
1500
1250
```

문제 79. 위의 결과를 다시 출력하는데 월급 앞에 이름도 같이 출력하시오.

답 

```
[root@localhost ~]# awk '$3=toupper("salesman") {print$2,$6}' emp.txt
```

결과 

```
MARTIN 1250
ALLEN 1600
TURNER 1500
WARD 1250
```

문제 80. 위의 스크립트를 이용해서 shell script 를 만드는데 직업을 물어보게 하고 직업을 입력하면 해당 직업인사원들의 이름과 월급이 출력되게 하시오.

\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

답 

```
echo -n 'Enter job title~'
read job

awk -v num=$job '$3==toupper(num) {print $2, $6}' emp.txt

[root@localhost ~]# vi find_job.sh

[root@localhost ~]# sh find_job.sh
```

결과 

```
Enter job title~salesman
MARTIN 1250

ALLEN 1600

TURNER 1500

WARD 1250
```



shell script 실무에서의 활용 예시

모 금융회사에서 데이터 분석을 할때 특정 보험상품이 어떤 고객에게 적합한 상품인지를 분석

이때 고객들을 연령대별로 별도의 텍스트 파일을 생성할때 유용함

e.g) 20대.txt, 30대.txt , 40대.txt

문제 81. 문제 80번 코드를 수정해서 직업을 물어보게 하고, 직업을 입력하면 해당 직업의 직원들의 모든 데이터가 다 출력되게 하시오.

```
echo -n 'Enter job title~'
read job

awk -v num=$job '$3==toupper(num) {print $0}' emp.txt
```

* \$0은 모든 컬럼(열) 을 의미한다.

```
[root@localhost ~]# rm find_job.sh -> 위에서 작업한 find_job.sh 파일을 수정할수도있지만 이처럼 삭제 해버리고 새로 만들어도 된다.
rm: remove regular file 'find_job.sh'? y
```

```
[root@localhost ~]# vi find_job.sh
```

```
[root@localhost ~]# sh find_job.sh
```

Enter job title~salesman

7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30

문제 82. 직업을 물어보게 하고 직업을 입력하면 해당 직업인 직원들의 모든 데이터로 텍스트 파일을 생성되게 하시오.

```
Enter the job ~ salesman
#ls -l salesman.txt <---salesman의 모든 데이터가 있어야 한다.

echo -n 'Enter job title~'

read job

awk -v num=$job '$3==toupper(num) {print $0}' emp.txt >> $job.txt -> 위에서 입력한 직업명.txt로 파일이 생성되도록!

[root@localhost ~]# vi find_job.sh

[root@localhost ~]# sh find_job.sh

Enter job title~salesman

[root@localhost ~]# cd

[root@localhost ~]# ls -l salesman.txt

-rw-r--r--. 1 root root 308 Dec 30 16:30 salesman.txt
```

```
[root@localhost ~]# cat salesman.txt

7654      MARTIN      SALESMAN      7698      1981-09-10      1250      1400      30

7499      ALLEN      SALESMAN      7698      1981-02-11      1600      300      30

7844      TURNER      SALESMAN      7698      1981-08-21      1500      0      30

7521      WARD      SALESMAN      7698      1981-02-23      1250      500      30

[root@localhost ~]# vi find_job.sh

[root@localhost ~]# sh find_job.sh

Enter job title-clerk

[root@localhost ~]# cd

[root@localhost ~]# ls -l clerk.txt

-rw-r--r--. 1 root root 287 Dec 30 16:31 clerk.txt

[root@localhost ~]# cat clerk.txt

7900      JAMES      CLERK      7698      1981-12-11      950      0      30

7369      SMITH      CLERK      7902      1980-12-09      800      0      20

7876      ADAMS      CLERK      7788      1983-01-15      1100      0      20

7934      MILLER      CLERK      7782      1982-01-11      1300      0      10
```

(오늘의 마지막 문제)

문제 83. 부서번호를 물어보게 하고 부서번호를 입력하면 해당 부서번호의 모든 사원의 데이터가 텍스트 파일로 생성되게 하시오.

\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

```
# vi find_dept_info.sh -> vi 편집기 열어서 명령어 입력하기
echo -n 'Enter deptno~'

read deptno

awk -v num=$deptno '$8==(num) {print $0}' emp.txt>> $deptno.txt

# sh find_dept_info.sh -> 입력한 명령문 실행하기~

Enter deptno~10

# sh find_dept_info.sh

Enter deptno~20

# sh find_dept_info.sh

Enter deptno~30
```

```
# cd
# ls -l 10.txt 20.txt 30.txt -> 부서별로 생성한 파일에 대한 정보 확인

-rw-r--r--. 1 root root 219 Dec 30 16:50 10.txt
-rw-r--r--. 1 root root 364 Dec 30 16:50 20.txt
-rw-r--r--. 1 root root 453 Dec 30 16:50 30.txt

#cat 10.txt

#cat 20.txt

#cat 30.txt
```


[20.12.31 리눅스 강의 Day 4]

문제 84. 이름이 SCOTT인 사원의 부서위치를 출력하시오.

SCOTT의 직원번호에 대한 데이터는 emp.txt에 있고 해당 부서번호에 따른 부서의 위치는 dept.txt에 있다.

scott의 직원번호를 알아내고 그 직원번호를 dept.txt에 입력해서 위치가 어디인지 출력

해결법

1. SCOTT의 부서번호를 emp.txt에서 알아낸다.
2. SCOTT의 부서번호로 부서위치(loc)를 dept.txt에서 알아낸다.

풀이 

\$1=직원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

1) scott의 부서번호 알아내기

```
awk '$2==toupper("scott") {print $8}' emp.txt
```

뜻 : \$2=이름 이 scott이면, 그 데이터의 \$8=부서번호에 해당하는 결과를 출력하라.

2) SCOTT의 부서번호로 부서위치(loc)를 dept.txt에서 알아내기

```
deptno=`awk '$2==toupper("scott") {print $8}' emp.txt`  
echo $deptno  
  
loc=`awk -v num=$deptno '$1==num {print $3}' dept.txt`  
  
echo $loc
```

답 

```
deptno=`awk '$2==toupper("scott") {print $8}' emp.txt`  
echo $deptno  
  
loc=`awk -v num=$deptno '$1==num {print $3}' dept.txt`  
  
echo $loc
```



※ 화면보호기 해제시 비번 입력하지 않고 해제하기

Applications → systemtools → settings → privacy → screen lock → off

문제 85. 위의 스크립트를 이용해서 이름을 물어보게하고 이름을 입력하면 해당 사원이 근무하는 부서위치가 출력되게 쉘 스크립트를 작성하시오

답 

```
vi find_loc.sh
```

문제 84 답 입력

실행 

```
sh find_loc.sh
```

결과 

```
Enter the ename ~~ scott
DALLAS
```

문제 86. dept.txt에서 DALLAS의 부서번호를 출력하시오.

\$1 = deptno \$2=부서명 \$3=지역

1) DALLAS 의 부서번호 출력하기

```
awk '$3==toupper("dallas") {print $1}' dept.txt
```

문제 87. 위의 부서번호를 가지고 emp.txt에서 해당 부서번호에서 근무하는 직원들의 이름과 직업을 출력하시오.

\$1=직원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

```
deptno=`awk '$3==toupper("dallas") {print $1}' dept.txt`  
awk -v num==deptno '$8==num {print $2,$3}' emp.txt
```



-v 옵션이란?

awk -v 옵션

변수에 있는 값을 다른 다른 변수에 담을때 사용하는 옵션

위의 문제에서 보면 deptno라는 값의 함수를 num에 담기위해서 사용한 것.

awk의 명령어 양끝에 ` 을 씌워줘야 한다. (~표시 키 + shift)

문제88. 위의 스크립트를 이용해서 부서위치를 물어보게하고 부서위치를 입력하면 해당 부서위치인 직원들의 이름과 직업이 출력되게 하시오

```
# sh find_loc2.sh
Enter the loc: dallas
```

```
JONES MANAGER
FORD ANALYST

SMITH CLERK

SCOTT ANALYST

ADAMS CLERK
```

```
# vi find_loc2.sh
echo -n 'Enter the loc: '

read loc

deptno=`awk -v num=$loc '$3==toupper(num) {print $1}' dept.txt`

awk -v num=$deptno '$8==num {print $2,$3}' emp.txt
```

문제 89. 위의 스크립트를 이용해서 부서위치를 물어보게하고 부서위치를 입력하면 해당 부서위치에 속하는
사원들의 모든 데이터를 아래와 같이 생성되게 하시오.

```
vi fin_loc.sh
echo -n 'Enter

read loc

deptno=`awk -v num=$loc '$3==toupper(num) {print $1}' dept.txt`

awk -v num=$deptno '$8==num {print $0}' emp.txt >> $loc.txt

sh fin_loc.sh
```

017 diff 명령어

💡 두 파일간의 차이점을 찾아서 알려주는 명령어

e.g)

```
diff emp.txt dallas.txt
```

문제 90. 어제 마지막 문제로 만들었던 find_deptno.sh 를 실행해서 20.txt를 생성하시오.

는..어제 문제 풀면서 이미 해버림..

답 

```
sh find_deptno.sh
Enter the deptno~ 20
```

하면 생성됨.

문제 91. emp.txt와 20.txt의 차이를 확인하시오

답 

```
diff emp.txt 20.txt
```

018. find 명령어



검색하고자 하는 파일을 찾을때 사용하는 명령어

e.g)

```
find 디렉토리 -name 파일명 -print
디렉토리 = 검색할 디렉토리

파일명 = 검색할 파일명

find/root -name 'emp.txt' -print
```

해석 : root 디렉토리 밑에 emp.txt란 파일이 있는지 검색하시오.

문제 92. 집에서 (/root) test100 이라는 디렉토리를 만들고 /root/test100 디렉토리 밑에 emp.txt를 복사하시오.

```
# cd
#mkdir test100

# cp /root/emp.txt /root/test100/emp.txt

# cd test101

# ls

# cd
```

설명 : /root/emp.txt 를 /root/test100/emp.txt로 복사해라~ 는 뜻

문제 93. /root밑에 emp.txt가 있는지 find 하시오.

```
find /root -name 'emp.txt' -print
```

문제 95. 위의 명령어를 이용해서 아래와 같이 검색을 편하게 할 수 있도록 shell script를 만드시오

```
sh find_file.sh
Enter the file name:

Enter the maxdepth:
```

답 

```
vi find_file.sh
echo -n 'Enter the file name: '

read file_name

echo -n 'Enter the maxdepth: '

read max_depth

find /root -maxdepth $max_depth -name $file_name -print
```

★ centos에서 아나콘다 설치하기

centos firefox에서 <https://velog.io/@shchoice/installpythonandanacondaincentos7> 로 접속한다.

'3. 아나콘다 설치 부분' 을 참고!

1. 파이썬 설치파일 다운로드



아나콘다 홈페이지 → download → 맨 하단 이동 → Linux의 64bit 부분 우클릭 (copy link location) click

→ 링크 복사됨 → terminal창을 연다. → wget명령어로 설치파일을 내려받는다.

how?

```
#wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-x86_64.sh + enter
```

→ 다운로드가 됨 // 다운로드가 100% 완료되면 다음 단계로!

→ ls 입력해 다운로드가 잘 되었는지 확인

→ #Anaconda3-2020.11-Linux- x86~ 하는 파일 있음

파일명 복사해서

→ `#sh Anaconda3-2020.11-Linux-x86_64(=위에서 복사한 파일명).sh + enter`

결과 

```
In order to continue the installation process, please review the license
agreement.
```

```
Please, press ENTER to continue
```

```
↓
```

```
Enter 입력
```

```
글이 쭉 나오는데 계속 enter 치거나
```

```
바로 빠져 나오려면 q
```

```

Do you accept the license terms? [yes|no]

라고 나오면

[no] >>> 'yes' + enter

Anaconda3 will now be installed into this location:

/root/anaconda3

위치 확인후 'yes' + enter

설치 시작

Preparing transaction: done

Executing transaction: done

installation finished.

Do you wish the installer to initialize Anaconda3
by running conda init? [yes|no]

If you'd prefer that conda's base environment not be activated on startup,
set the auto_activate_base parameter to false:

conda config --set auto_activate_base false

Thank you for installing Anaconda3!

설치 완료!

```

2. 리눅스의 환경설정 파일 bashrc 파일을 수행한다.

터미널에서 `source .bashrc` 입력

```
(base) [root@localhost ~] # 라고 나오면 성공
```

3. 아나콘다 가상환경 만들기



아나콘다 가상환경은 하나의 os에서 여러개를 설정할 수 있다.

머신러닝을 위한 파이썬 환경과 딥러닝을 위한 파이썬 환경이 서로 다르므로

각각 별도의 환경을 만들어서 파이썬을 사용하고 싶다면 가상환경을 생성하면 된다.

how?

terminal에서

```
conda create -n
```

입력

```
(base) [root@localhost ~]# conda create -n py38
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

```
## Package Plan ##
```

```
environment location: /root/anaconda3/envs/py38
```

```

Proceed ([y]/n)? y

Preparing transaction: done

Verifying transaction: done

Executing transaction: done

#

# To activate this environment, use

#

#     $ conda activate py38

#

# To deactivate an active environment, use

#

#     $ conda deactivate

```



자신이 생성한 특정 파이썬 환경에 접속하고 싶다면

```
conda activate 파이썬환경이름
```

해당 파이썬 환경에서 빠져나오고 싶다면

```
conda deactivate
```

4. 스파이더 설치하기

terminal에서

```
conda install spyder
```

```

(base) [root@localhost ~]# conda install spyder
Collecting package metadata (current_repodata.json): done

Solving environment: done

# All requested packages already installed.

```

가상환경이 잘 설정되면

[root@localhost ~] 이

(base) [root@localhost ~] 로 된다 (앞에 base가 붙음)

019. tar 명령어



파일을 압축하고 해제하는 명령어
e.g)

1. 압축할 때

`tar cvf 압축파일명 압축파일대상`

1. 압축해지 할 때

`tar xvf 압축파일명` 압축을 해제할 위치

* 옵션 :

`tar c : c=compress` , 여러 개의 파일을 하나로 만들어라

`tar v : v=view`, 압축되는 과정을 보여달라

`tar f : f=file`, 생성되는 파일명을 지정

`tar x : x=extract`, 묶어있는 파일을 풀어줘라

`tar -C` : 압축이 풀릴 위치를 지정

문제 96. 집에서(/root) emp.txt를 복사해서 emp1.txt, emp2.txt, emp3.txt를 생성하시오.

답

```
(base) [root@localhost ~]# cp emp.txt emp1.txt
(base) [root@localhost ~]# cp emp.txt emp2.txt

(base) [root@localhost ~]# cp emp.txt emp3.txt

(base) [root@localhost ~]# ls
```

문제 97. 현재 디렉토리에 emp로 시작되는 모든 text 파일들을 empall.tar 라는 이름으로 압축하시오.

```
tar cvf empall.tar ./emp*.txt
```

`./` → 현재 디렉토리 를 의미

답

```
(base) [root@localhost ~]# tar cvf empall.tar ./emp*.txt
```

결과 : 압축되는 과정이 보여지고 있는 것

```
./emp1.txt
./emp2.txt

./emp3.txt

./emp.txt
```

```
(base) [root@localhost ~]# ls -l empall.tar
-rw-r--r--. 1 root root 10240 Dec 30 23:55 empall.tar
```


로 압축이 되어 있는걸 확인할 수 있다.

문제 98. 현재 디렉토리에 test200 이라는 디렉토리를 만들고 empall.tar 파일을 /root 에서 /root/test200 으로 복사하시오.

답

```
(base) [root@localhost ~]# mkdir test200
(base) [root@localhost ~]# ls

anaconda3                Documents  empall.tar             Public
Anaconda3-2020.11-Linux-x86_64.sh  Downloads  emp.txt                 Templates
anaconda-ks.cfg           emp1.txt   initial-setup-ks.cfg   test200
dept.txt                  emp2.txt   Music                   Videos
Desktop                   emp3.txt   Pictures
(base) [root@localhost ~]# cp empall.tar ./test200/
```

./test200/ 에서 ./ 은 현재 디렉토리를 의미함.

결과

```
(base) [root@localhost ~]# cd test200
(base) [root@localhost test200]# ls

empall.tar
```

복사가 된 것을 확인할 수 있다.

문제 99. 현재 디렉토리에서 test200 디렉토리로 이동해서 empall.tar 파일의 압축을 해제하시오.

답

```
(base) [root@localhost ~]# cd test200  -> test200 디렉토리로 이동
(base) [root@localhost test200]# tar xvf empall.tar ./  압축파일을 ./ (현재위치)에 풀어라

./emp1.txt

./emp2.txt

./emp3.txt

./emp.txt
```

결과

```
(base) [root@localhost test200]# ls -l
total 28
-rw-r--r--. 1 root root 1036 Dec 30 23:51 emp1.txt
-rw-r--r--. 1 root root 1036 Dec 30 23:51 emp2.txt
-rw-r--r--. 1 root root 1036 Dec 30 23:51 emp3.txt
-rw-r--r--. 1 root root 10240 Dec 30 23:59 empall.tar
-rw-rw-rw-. 1 root root 1036 Dec 30 23:42 emp.txt
```

문제 100. /root 밑에 있는 dept.txt 를 dept1.txt dept2.txt dept3.txt 로 복사하고 /root 밑에 있는 dept로 시작하는 텍스트 파일들을 deptall.tar 로 압축한 후에 /root 밑에 test500 이라는 디렉토리를 만들고 deptall.tar 을 test500 밑에 복사한 뒤에 압축을 해제하시오.

- 1) /root 밑에 있는 dept.txt 를 dept1.txt, dept2.txt, dept3.txt 로 복사

```
(base) [root@localhost test200]# cd
(base) [root@localhost ~]# cp dept.txt dept1.txt

(base) [root@localhost ~]# cp dept.txt dept2.txt

(base) [root@localhost ~]# cp dept.txt dept3.txt
```

- 2) /root 밑에 있는 dept로 시작하는 텍스트 파일들을 deptall.tar 로 압축

```
(base) [root@localhost ~]# tar cvff deptall.tar ./dept*.txt
```

- 3) /root 밑에 test500 이라는 디렉토리를 만들고

```
(base) [root@localhost ~]# mkdir test500
```

- 4) deptall.tar 을 test500 밑에 복사한 뒤에 압축을 해제하시오.

```
(base) [root@localhost ~]# cp deptall.tar ./test500/ <-마지막 /를 꼭 입력해야 한다.!
```

그렇지 않을 경우 test500 이라는 파일을 만들어 버림.

```
(base) [root@localhost ~]# cd test500
(base) [root@localhost test500]# ls

deptall.tar

(base) [root@localhost test500]# tar xvf deptall.tar ./

./dept1.txt
./dept2.txt
./dept3.txt
./dept.txt
```

문제 101. 윈도우에서 emp.txt와 dept.txt를 압축한다. 압축파일명은 fileall.tar로 압축하시오. 반디집으로 압축시 확장자를 tar로 해서 압축하시오.

문제 102. fileall.tar 압축파일을 리눅스에 바탕화면에 가져다 놓은뒤에 HOME 폴더에 fileall.tar 을 복사해 놓으시오

문제 103. /root 밑에 있는 fileall.tar 를 압축해제 하시오.

답 

```
(base) [root@localhost ~]# tar xvf fileall.tar
```

문제 104. 윈도우에 서 jobs.txt를 jobs2.txt로 복사한 후에 두개의 파일을 jobs.zip 으로 압축하고 리눅스 서버에 바탕화면에 home폴더에 넣으시오.

```
(base) [root@localhost ~]# ls -lrt -> rt(reverse time) 생성시기 역순으로 출력
```

```
total 541624

-rw-r--r--. 1 root root 554535580 Nov 18 18:04 Anaconda3-2020.11-Linux-x86_64.sh
-rwxr-xr-x. 1 root root 1036 Dec 29 00:11 emp.txt
-rwxr-xr-x. 1 root root 105 Dec 29 00:11 dept.txt
-rw-----. 1 root root 1755 Dec 30 22:58 anaconda-ks.cfg
-rw-r--r--. 1 root root 1803 Dec 30 23:00 initial-setup-ks.cfg

drwxr-xr-x. 2 root root      6 Dec 30 23:00 Videos
drwxr-xr-x. 2 root root      6 Dec 30 23:00 Templates
drwxr-xr-x. 2 root root      6 Dec 30 23:00 Public
drwxr-xr-x. 2 root root      6 Dec 30 23:00 Music
drwxr-xr-x. 2 root root      6 Dec 30 23:00 Downloads
drwxr-xr-x. 2 root root      6 Dec 30 23:00 Documents
drwxr-xr-x. 27 root root 4096 Dec 30 23:22 anaconda3

-rw-r--r--. 1 root root 1036 Dec 30 23:51 emp1.txt
-rw-r--r--. 1 root root 1036 Dec 30 23:51 emp2.txt
-rw-r--r--. 1 root root 1036 Dec 30 23:51 emp3.txt
-rw-r--r--. 1 root root 10240 Dec 30 23:55 empall.tar

drwxr-xr-x. 2 root root      87 Dec 31 00:05 test200

-rw-r--r--. 1 root root 105 Dec 31 00:09 dept1.txt

drwxr-xr-x. 2 root root      53 Dec 31 00:09 Pictures


-rw-r--r--. 1 root root 105 Dec 31 00:09 dept2.txt
-rw-r--r--. 1 root root 105 Dec 31 00:09 dept3.txt
-rw-r--r--. 1 root root 10240 Dec 31 00:13 deptall.tar

drwxr-xr-x. 2 root root      92 Dec 31 00:14 test500

-rw-rw-rw-. 1 root root 4096 Dec 31 00:30 fileall.tar
-rw-rw-rw-. 1 root root 10530 Dec 31 00:51 jobs.zip

drwxr-xr-x. 2 root root      37 Dec 31 00:51 Desktop
```

이후 jobs.zip 을 test800 폴더를 만들어서 옮기고 압축 해제

 * zip 파일은 tar 이 아닌 unzip 을 쓴다.!

```
(base) [root@localhost ~]# mkdir test800
(base) [root@localhost ~]# cd

(base) [root@localhost ~]# cd jobs.zip ./test800/
```

```

bash: cd: jobs.zip: Not a directory

(base) [root@localhost ~]# cp jobs.zip ./test800/

(base) [root@localhost ~]# cd test800

(base) [root@localhost test800]# unzip jobs.zip

Archive:  jobs.zip
  inflating: jobs.txt
  inflating: jobs2.txt

```

020. ln 명령어

💡 link 하는 명령어

💡 os윈도우의 바로가기 같은 기능

e.g) 바탕화면의 바로가기 아이콘

자주 실행하는 문서의 경우 바탕화면에 바로가기를 가져다 두면 편리하다.

이러한 기능이 리눅스에 ln 명령어도 구현이 된다.

e.g) 집에서 (/root) 에서 find_job.sh 라는 파일이 있는지 확인하고 cat 으로 파일을 여시오.

```
(base) [root@localhost ~]# ln -s /root/find_job.sh /root/Desktop/find_job.sh
```

/root 밑의 find_job.sh 를 바탕화면 (/root/Desktop) 에 find_job.sh 로 바로가기를 생성하겠다.

가 바탕화면에 생성된 것을 확인할 수 있다.

021. sed 명령어

💡 grep 명령어는 파일의 특정 내용을 검색하는 기능을 갖는다면

sed 명령어는 검색 뿐만이 아니라 내용을 변경할 수도 있다.

e.g)

```
(base) [root@localhost ~]# cd
```

```
(base) [root@localhost ~]# sed 's/KING/yyy/' emp.txt -> emp.txt에서 KING을 yyy로 변경하겠다.
```

7839	yyy	PRESIDENT	0	1981-11-17	5000	0	10
7698	BLAKE	MANAGER	7839	1981-05-01	2850	0	30
7782	CLARK	MANAGER	7839	1981-05-09	2450	0	10

7566	JONES	MANAGER	7839	1981-04-01	2975	0	20
------	-------	---------	------	------------	------	---	----

다만 위의 명령어는 '변경해서 보여줄' 뿐이지, 실제 데이터가 변경된 것은 아니다.

```
(base) [root@localhost ~]# cat emp.txt
```

7839	KING	PRESIDENT	0	1981-11-17	5000	0	10
7698	BLAKE	MANAGER	7839	1981-05-01	2850	0	30
7782	CLARK	MANAGER	7839	1981-05-09	2450	0	10

문제 105. 위에서 KING 이름을 yyy 로 변경해서 보여주고 있는 결과화면을 저장해서 emp900.txt로 저장 되게 하시오. (redirection → '>' 사용)

답

```
(base) [root@localhost ~]# sed 's/KING/yyy/' emp.txt >> emp900.txt
(base) [root@localhost ~]# ls -l emp*.txt
```

결과

```
- rw-r--r--. 1 root root 1036 Dec 30 23:51 emp1.txt
- rw-r--r--. 1 root root 1036 Dec 30 23:51 emp2.txt
- rw-r--r--. 1 root root 1036 Dec 30 23:51 emp3.txt
- rw-r--r--. 1 root root 1035 Dec 31 01:19 emp900.txt
- rwxr-xr-x. 1 root root 1036 Dec 29 00:11 emp.txt
```

022. case 문

💡 if 문과 유사한 리눅스의 shell script 명령어

e.g) 예전에 만들었던 find_sal.sh 와 find_file.sh 가 잘 수행되는지 확인하시오.

e.g2) 아래의 스크립트를 먼저 메모장에서 작성하시오

```
echo "Press number 1 for check employees sal level
Press number 2 for confirm to search file"

echo -n "Press number"

read choice

case $choice in
```

1)

```
sh /root/find_sal.sh ;;
```

2)

```
sh /root/find_file.sh ;;
```

e.g3) root 밑에 vi a.sh를 생성하고 그 안에 위의 스크립트를 붙여넣고 수행이 되는지 확인 하시오.

```
(base) [root@localhost ~]# vi a.sh
echo "Press number 1 for check employees sal level

Press number 2 for confirm to search file"

echo -n "Press number"

read choice

case $choice in
1)
sh /root/find_sal.sh ;;
2)
sh /root/find_file.sh ;;
esac

(base) [root@localhost ~]# sh a.sh

Press number 1 for check employees sal level

Press number 2 for confirm to search file

Press number1

Enter the  ename ~~scott

SCOTT 3000
```

```
(base) [root@localhost ~]# sh a.sh

Press number 1 for check employees sal level

Press number 2 for confirm to search file

Press number2

Enter the file name: emp.txt

Enter the maxdepth: 1

/root/emp.txt
```



※ 위의 음영된 공백 이 터미널에서 깔끔하게 나오도록 하는 법

1. vi a.sh 로 수정하러 들어간다
2. 공백있는 부분에서 'x'를 누르면 파이썬에서shift tap 한것 처럼 문장이 안쪽으로 이동된다
3. 수정 완료후 빠져나오기! 끝!

문제 106. /root 밑에 있는 find_loc2.sh 를 실행해 보시오

문제 107. find_loc.sh를 위의 e.g)에서 만든 a.sh의 3번으로 추가하시오.

답 

```
echo "Press number 1 for check employees sal level
Press number 2 for confirm to search file

Press number 3 for confirm to find loc"

echo -n "Press number"

read choice

case $choice in
1)
sh /root/find_sal.sh ;;
2)
sh /root/find_file.sh ;;
3)
sh /root/find_loc.sh ;;
esac
```

: 추가된 부분

결과 

```
rm: remove regular file 'a.sh'? y
(base) [root@localhost ~]# vi a.sh

(base) [root@localhost ~]# sh a.sh

Press number 1 for check employees sal level

Press number 2 for confirm to search file

Press number 3 for confirm to find loc

Press number3

Enter the ename~~scott

DALLAS
```

023. cp 명령어 (copy)



파일을 복사하는 명령어

e.g)

cp 파일명 복사할 파일명

cp 위치/파일명 위치/복사할 파일명

cp emp.txt emp400.txt

cp /root/emp.txt /root/test01/emp400.txt

문제 108. /root/ 밑에 있는 확장자가 .txt로 파일을 전부 /root/backup/ 이라는 폴더를 만들어서 복사하시오.

답

```
(base) [root@localhost ~]# cp /root/*.txt /root/backup/
```

혹은


```
(base) [root@localhost ~]# cp /root/*.txt ./backup/
```

결과 

```
(base) [root@localhost ~]# cd backup
(base) [root@localhost backup]# ls -l

total 36
-rw-r--r--. 1 root root 105 Dec 31 02:07 dept1.txt
-rw-r--r--. 1 root root 105 Dec 31 02:07 dept2.txt
-rw-r--r--. 1 root root 105 Dec 31 02:07 dept3.txt
-rwxr-xr-x. 1 root root 105 Dec 31 02:07 dept.txt
-rw-r--r--. 1 root root 1036 Dec 31 02:07 emp1.txt
-rw-r--r--. 1 root root 1036 Dec 31 02:07 emp2.txt
-rw-r--r--. 1 root root 1036 Dec 31 02:07 emp3.txt
-rw-r--r--. 1 root root 1035 Dec 31 02:07 emp900.txt
-rwxr-xr-x. 1 root root 1036 Dec 31 02:07 emp.txt
```

024. mv 명령어

 파일의 이름을 바꾸거나 파일을 다른 디렉토리로 이동하는 명령어

e.g)

mv 기존파일명 새로운파일명

mv emp400.txt emp500.txt

```
(base) [root@localhost ~]# ls emp*.txt
```

emp1.txt emp2.txt emp3.txt emp500.txt emp900.txt emp.txt

e.g 2)

mv emp500.txt ./backup/

* 설명 : emp500.txt를 현재 디렉토리 밑에 backup 밑에 이동

문제 109. 집으로 와서 (/root) backup2 폴더를 만들고 현대 디렉토리(집) 에 있는 텍스트 파일들을 전부 backup2 폴더로 이동시키시오.


```
(base) [root@localhost ~]# mv *.txt ../backup2/
(base) [root@localhost ~]# cd backup2

(base) [root@localhost backup2]# ls *.txt

dept1.txt  dept2.txt  dept3.txt  dept.txt  emp1.txt  emp2.txt  emp3.txt  emp999.txt  emp.txt

(base) [root@localhost backup2]# cd

(base) [root@localhost ~]# ls *.txt

bash: ls*.txt: command not found...
```

★ centos에서 spyder 설치하기

1. 아래에 py38 가상 환경에 있는지 확인하시오

답 

```
(base) [root@localhost ~]# conda activate py38 -> py38이라는 가상환경으로 들어가라는 명령어
```

```
(py38) [root@localhost ~]#
```

2. spyder 설치 여부를 확인한다

```
(py38) [root@localhost ~]# conda list spyder
```

```
# packages in environment at /root/anaconda3/envs/py38:
#
# Name                               Version                               Build Channel
```

아무것도 나오지 않으면 설치되지 않은 것

2-1. spyder 를 설치한다.

`conda install spyder` 입력

만약 에러가 나면

1. 인터넷 연결상태 확인
2. 다른 에러 발생시 아래의 명령어 입력

`pip install PyQt5==5.10`

에러 발생

```
pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the current problem.
spyder 4.2.0 requires pyqtwebengine<5.13, which is not installed.
pip install pyqtwebengine
```

해결법 발견

아나콘다 설치후

선생님이 알려주신

```
pip install PyQt5==5.10
```

```
pip install pyqtwebengine==5.12
```

```
pip install pyqt5==5.12
```

위 3가지 모듈을 설치한 후 spyder 실행이 되지 않는데,

그 에러의 내용들이 대략 아래와 같다면..

spyder 4.1.5 requires pyqt5<5.13; python_version >= "3", but you'll have pyqt5 5.15.2 which is incompatible.

[21.01.04 리눅스 강의 Day 5]

리눅스는 머신 러닝, 딥러닝을 위해서는 필수적으로 알아야 하는 운영체제!

■ 지난주 배운 리눅스 명령어 복습

1. 리눅스 기본 명령어 (24가지)
2. 리눅스에서 아나콘다 설치 + 스파이더 설치

■ 이번주에 배울 것들

1. vi 편집기
2. shell script 생성
3. 기타 리눅스 관리 명령어들

■ 003. vi 편집기 명령어



vi 편집기란?

리눅스 안에서 사용할 수 있는 문서 편집기

vi는 Visual Editor 의 약자

● vi 편집기 명령모드 3가지

1. command 모드
2. edit 모드
3. last line 모드

4. command 모드

- vi 편집기의 기본모드
- vi를 실행하면 바로 보이는 화면
- 방향키로 이동할 수 있는 모드

1) edit 모드

- a, i, o, x 등을 누르면서 내용을 입력 또는 삭제하는 명령 모드
- i 누르면 하단에 INSERT 라고 바뀜 : 텍스트 입력이 가능한 모드
- x : 지우기 (백스페이스)
- esc : 커서이동 모드로 전환

● vi 편집기의 수정 명령어

a → 커서 다음에 입력하겠다.

i → 커서전에 입력하겠다.

r → 커서에 위치하는 철자를 수정하겠다.

- a 나 i 는 누르면 바로 입력모드로 바뀐다.

r 은 조합키 처럼 r 누른 상태에서 바꿀 문자/숫자를 입력해도 된다.

커서 위치를 원하는 곳에 명령어 + 입력할 것

2) last line 모드

- 입력모드에서 저장, 종료, 강제종료등의 명령어를 입력하는 모드

Aa 명령어	내용	단축키	주석
:wq	저장하고 종료	ZZ	
:q	저장 않고 종료	ZQ	

● vi 편집기 내에서 커서 이동

j → 아래로 이동

k → 위로 이동

h → 왼쪽으로 이동

l → 오른쪽으로 이동

----- 요기까진 사실 방향키가 더 편할거같은데 아래는 유용할듯!-----

w → 어절 별로 이동

1G → 맨위로 이동

G → 맨 아래로 이동

:set nu → 파일 내의 텍스트에 번호를 표시 → 스파이더에서 명령어 옆에 번호 쓰듯이 줄별로 번호가 붙는다.

추후 maria db 등에서 테이블 만들때 몇행에 오류가 나서 생성 실패. 입력 실패! 이런 것들이 발생하는데

이때 :set nu 를 통해 번호를 매기면 특정 문제가 발생한 열을 바로 찾아갈 수 있다.

:set nonu → 번호를 안보이게 하는 명령어

gg → 맨위로 이동하는 단축키

● vi 편집기에서 삭제 명령어

x → 철자 하나 삭제

dd → 한 행 삭제

dw → 커서가 있는 단어 삭제

:5,10 d → 5~10번째 '행' 삭제

D → 커서 오른쪽 행 삭제 (del)

● vi 편집기에서 취소 명령어

u → 방금 작업 했던 것 취소 (undo? 인듯) / ctrl + z 와 같은 기능

많은 연습을 통해 vi 편집기를 현란하게 다룰 줄 알아야 한다!

문제 111. 동전을 100000번 던져서 앞면이 100000만번 중에 앞면이 몇번 나오는지 출력하시오.

python coin_cnt.py

답

```
1. vi coin_cnt.py 로 편집기 들어가기
2. 코드입력

import random

coin=['Heads','Tails']

cnt=0

for i in range(1,100001):

    result=random.choice(coin)

    if result=='Heads':

        cnt+=1

    print(cnt)

실행

vi coin_cnt.py
```

결과

```
(base) [root@localhost ~]# python coin_cnt.py
49826
```

vi 편집기의 복사/붙여넣기 명령어

Aa 명령어	≡ 설명	≡ 주석
yy	하나의 행을 복사	
p	붙여넣기	
YG	현재행 부터 파일 끝까지 복사	
:1,2 co 3	1~2행을 3행 다음으로 '복사'	co = copy
:1,2 m 3	1~2행을 3행 다음으로 '이동'	m = move

문제 112. jobs.txt를 열어서 맨 위에 한줄을 복사해서 다음 라인에 붙여넣으시오.

복사하는 명령어 → yy

답

```
vi jobs.txt 입력해 vi 편집기로 jobs.txt 열기

복사할 행에가서 yy 하면 복사됨

붙여넣기할 행에가서 p 하면 붙여넣기 됨.
```

문제 113. jobs.txt 를 열어서 전체를 복사한 후에 맨 밑에 전체를 다 붙여넣으시오.

답 

```
1. vi jobs.txt 로 접속
2. yG를 눌러 전체행 복사
3. G 눌러서 맨 아래로 이동
4. p 눌러서 복사한 내용 복사

yG 로 전체행 복사 -> 맨 밑에 288 lines yanked 라고 확인표시가 뜸


G 눌러서 맨 밑에 행으로 이동 -> Thank you all very much.

여기서 p 누름

만약 복사한 내용을 다 지우고 싶다면.

dG
```

문제 114. jobs.txt 를 열어서 위아래 좌우로 이동해본다.


 j → 아래로 이동

k → 위로 이동

h → 왼쪽으로 이동

l → 오른쪽으로 이동

문제 115. jobs.txt를 열어서 라인마다 숫자 번호가 보이도록 하시오.

 :set nu → 숫자 매기는 명령어

1G == gg -> 맨윗줄로 이동

G → 맨 아래로 이동

문제 116. jobs.txt 에서 1번라인 ~ 5번라인까지 복사해서 jobs3.txt를 생성하시오.

답 

```
:1,5 w jobs3.txt
"jobs3.txt" [New][dos] 5L, 322C written 라고 결과가 나옴.

cat jobs3.txt 로 확인

(base) [scott@localhost ~]$ cat jobs3.txt

steve jobs'2005 stanford commencement address

I am honored to be with you today at your commencement from one of the finest universities in the world.

I never graduated from college.

Truth be told, this is the closest I've ever gotten to a college graduation.
```

Today I want to tell you three stories from my life.

문제 117. jobs.txt 를 열어서 1번라인부터 20번 라인까지의 내용을 복사해 jobs7.txt로 생성하시오.,

```
1. vi jobs.txt 로 연다
2. 넘버링을 한다. -> :set nu
3. 1~20을 복사해서 jobs7.txt 로 만든다.
  -> :1,20 w jobs7.txt
1. esc, ZZ 눌러서 저장하고 나옴
2. cat jobs7.txt 로 확인
3. wc -l jobs7.txt 로 파일 있는지도 확인
```

문제 118. 구구단 2단을 출력하는 파이썬 코드를 리눅스 에서 작성하고, 아래와 같이 실행되게 하시오.

python p_2_dan.py

답

```
1. vi 편집기로 스트림입력 파일 생성

$ vi p_2_dan.py

1. 구구단 출력할 스크립트 입력

for i in range(1,10):

print(f'2x{i}=2*{i}')

1. 입력한 스크립트 파이썬으로 출력

(base) [scott@localhost ~]$ python p_2_dan.py
```

결과

```
2x1=2*1
2x2=2*2

2x3=2*3

2x4=2*4

2x5=2*5

2x6=2*6

2x7=2*7

2x8=2*8

2x9=2*9
```

문제 119. 위와 같은 과정을 거쳐 구구단 전체를 출력하시오.

python gugu_dan.py

2단 부터 9단까지 나오도록

답

```
1. vi 편집기로 입력할 파일 생성

vi gugu_dan.py

2. 출력할 스크립트 작성 ( i ) 누르고
```

```
for j in range(2,10):
    for i in range (1,10) :
        print(j , 'x' , i , '=', j*i)

3. 저장하고 나와서 python gugu_dan.py로 실행
```

결과 

```
(base) [scott@localhost ~]$ python gugu_dan.py
2 x 1 = 2

2 x 2 = 4

2 x 3 = 6

2 x 4 = 8

.


.

.

9 x 8 = 72

9 x 9 = 81
```

vi 편집기 내에서 특정 문자를 검색하는 방법

 ★문법: `:/문자 + enter`

검색이 된 후에

vi 편집기 내에서 특정 문자를 검색하는 방법의 사본

Aa 명령어	≡ 설명
<code>n</code>	다음 검색된 검색어
<code>shift + n</code>	이전 검색된 검색어

| 문제 120. emp.txt를 열어서 SCOTT 이라는 문자가 있는지 검색하시오.


답 

```
vi emp.txt

:/SCOTT
```

* toupper 같은 기능은 없으니 정확하게 입력해줘야 한다!

vi 편집기 명령어로 문자를 변경하는 방법

 ★문법: `:%s/기존문자/변경할문자/g`

e.g) emp.txt 에서 KING을 aaa로 변경하시오.

답

```
:%s/KING/aaa/g + enter
```

💡 g 는 앞에서 지정한 변경할 문자열에 해당하면 모두 바꾸라는 뜻.

g=global

문제 121. 다시 aaa를 KING으로 변경 하시오.

답

```
vi emp.txt
로 편집기 진입후
아래의 명령어 입력

:%s/aaa/KING/g
```

● 모든 데이터를 변경하는 것이 아니라 특정 일치하는 값 하나만 바꾸고 싶다면?

💡 ★문법: :s/기존문자/변경할문자

지금 커서가 있는 현재행의 기존 문자를 변경할 문자로 변경하겠다는 뜻.

e.g) emp.txt를 열어서 SALESMAN에 커서를 대고 aaa로 바꿔 보면

7839	KING	PRESIDENT	0	1981-11-17	5000	0	10
7698	BLAKE	MANAGER	7839	1981-05-01	2850	0	30
7782	CLARK	MANAGER	7839	1981-05-09	2450	0	10
7566	JONES	MANAGER	7839	1981-04-01	2975	0	20
7654	MARTIN	aaa	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950	0	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30

.
. .
.

커서가 있던 부분의 문자만 바뀐다.

커서가 있는 위치에 지정한 문자가 없으면 실행되지 않는다.

문제 122. emp.txt 를 열고 이름이 ALLEN인 사원의 직업을 bbbb로 변경하시오.

답

```
vi emp.txt

ALLEN의 직업 란(SALESMAN) 에 가서

:s/SALESMAN/bbbb
```

결과

7839	KING	PRESIDENT	0	1981-11-17	5000	0	10
7698	BLAKE	MANAGER	7839	1981-05-01	2850	0	30
7782	CLARK	MANAGER	7839	1981-05-09	2450	0	10
7566	JONES	MANAGER	7839	1981-04-01	2975	0	20
7654	MARTIN	aaa	7698	1981-09-10	1250	1400	30
7499	ALLEN	bbbb	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950	0	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
.							
.							
.							

문제 123. emp.txt를 열고 직업이 MANAGER인 사람들의 직업을 모두 SALEMAN 으로 변경하시오.

답

```
vi emp.txt

:%s/MANAGER/SALEMAN/g
```

결과

7839	KING	PRESIDENT	0	1981-11-17	5000	0	10
7698	BLAKE	SALESMAN	7839	1981-05-01	2850	0	30
7782	CLARK	SALESMAN	7839	1981-05-09	2450	0	10
7566	JONES	SALESMAN	7839	1981-04-01	2975	0	20
7654	MARTIN	aaa	7698	1981-09-10	1250	1400	30
7499	ALLEN	bbbb	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950	0	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000	0	20
7369	SMITH	CLERK	7902	1980-12-09	800	0	20
7788	SCOTT	ANALYST	7566	1982-12-22	3000	0	20

7876	ADAMS	CLERK	7788	1983-01-15	1100	0	20
7934	MILLER	CLERK	7782	1982-01-11	1300	0	10

문제 124. emp.txt 를 emp1.txt ~ emp20.txt로 복사하시오.

답 

```
(base) [scott@localhost ~]$ cp emp.txt emp1.txt
(base) [scott@localhost ~]$ cp emp.txt emp2.txt
(base) [scott@localhost ~]$ cp emp.txt emp3.txt
(base) [scott@localhost ~]$ cp emp.txt emp4.txt
(base) [scott@localhost ~]$ cp emp.txt emp5.txt
(base) [scott@localhost ~]$ cp emp.txt emp6.txt
(base) [scott@localhost ~]$ cp emp.txt emp7.txt
(base) [scott@localhost ~]$ cp emp.txt emp8.txt
(base) [scott@localhost ~]$ cp emp.txt emp9.txt
(base) [scott@localhost ~]$ cp emp.txt emp10.txt
(base) [scott@localhost ~]$ cp emp.txt emp11.txt
(base) [scott@localhost ~]$ cp emp.txt emp12.txt
(base) [scott@localhost ~]$ cp emp.txt emp13.txt
(base) [scott@localhost ~]$ cp emp.txt emp14.txt
(base) [scott@localhost ~]$ cp emp.txt emp15.txt
(base) [scott@localhost ~]$ cp emp.txt emp16.txt
(base) [scott@localhost ~]$ cp emp.txt emp17.txt
(base) [scott@localhost ~]$ cp emp.txt emp18.txt
(base) [scott@localhost ~]$ cp emp.txt emp19.txt
(base) [scott@localhost ~]$ cp emp.txt emp20.txt
```

결과 

```
(base) [scott@localhost ~]$ ls -l emp*.txt

-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp10.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp11.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp12.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp13.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp14.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp15.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp16.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp17.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp18.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp19.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 00:59 emp1.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp20.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 00:59 emp2.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 00:59 emp3.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 00:59 emp4.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp5.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp6.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp7.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp8.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 01:00 emp9.txt
-rwxrwx---. 1 scott scott 1030 Jan 4 00:56 emp.txt
```



* 사실 shell script를 쓸줄 알면 한번에 끝낼 수도있는데, 그걸 모르면 이렇게 노가다를 해야 한다는 것을 실습한것.....ㅎ

문제 125. emp1.txt~emp20.txt 의 내용중에 SALESMAN을 jjjj로 변경하시오.

답 

```
vi emp*.txt

:argdo %s/SALESMAN/jjjj/g | update
```

결과 

```
"emp10.txt" [dos] 14L, 1010C written
"emp11.txt" [dos format] 14 lines, 1030 characters
"emp11.txt" [dos] 14L, 1010C written
"emp12.txt" [dos format] 14 lines, 1030 characters
"emp12.txt" [dos] 14L, 1010C written
"emp13.txt" [dos format] 14 lines, 1030 characters
"emp13.txt" [dos] 14L, 1010C written
"emp14.txt" [dos format] 14 lines, 1030 characters
"emp14.txt" [dos] 14L, 1010C written
"emp15.txt" [dos format] 14 lines, 1030 characters
"emp15.txt" [dos] 14L, 1010C written
"emp16.txt" [dos format] 14 lines, 1030 characters
"emp16.txt" [dos] 14L, 1010C written
"emp17.txt" [dos format] 14 lines, 1030 characters
"emp17.txt" [dos] 14L, 1010C written
"emp18.txt" [dos format] 14 lines, 1030 characters
"emp18.txt" [dos] 14L, 1010C written
"emp19.txt" [dos format] 14 lines, 1030 characters
"emp19.txt" [dos] 14L, 1010C written
"emp1.txt" [dos format] 14 lines, 1030 characters
"emp1.txt" [dos] 14L, 1010C written
"emp20.txt" [dos format] 14 lines, 1030 characters
"emp20.txt" [dos] 14L, 1010C written
"emp2.txt" [dos format] 14 lines, 1030 characters
"emp2.txt" [dos] 14L, 1010C written
"emp3.txt" [dos format] 14 lines, 1030 characters
"emp3.txt" [dos] 14L, 1010C written
"emp4.txt" [dos format] 14 lines, 1030 characters
"emp4.txt" [dos] 14L, 1010C written
"emp5.txt" [dos format] 14 lines, 1030 characters
"emp5.txt" [dos] 14L, 1010C written
"emp6.txt" [dos format] 14 lines, 1030 characters
```

```
"emp6.txt" [dos] 14L, 1010C written
"emp7.txt" [dos format] 14 lines, 1030 characters
"emp7.txt" [dos] 14L, 1010C written
"emp8.txt" [dos format] 14 lines, 1030 characters
"emp8.txt" [dos] 14L, 1010C written
"emp9.txt" [dos format] 14 lines, 1030 characters
"emp9.txt" [dos] 14L, 1010C written
"emp.txt" [dos format] 14 lines, 1030 characters
"emp.txt" [dos] 14L, 1010C written

7698    BLAKE    jjjj    7839    1981-05-01    2850    0    30
7782    CLARK    jjjj    7839    1981-05-09    2450    0    10
7566    JONES    jjjj    7839    1981-04-01    2975    0    20
7654    MARTIN   aaa     7698    1981-09-10    1250    1400    30
7499    ALLEN    bbbb    7698    1981-02-11    1600    300    30
7844    TURNER   jjjj    7698    1981-08-21    1500    0    30
7900    JAMES    CLERK    7698    1981-12-11    950     0    30
7521    WARD     jjjj    7698    1981-02-23    1250    500    30
7902    FORD     ANALYST   7566    1981-12-11    3000    0    20
7369    SMITH    CLERK    7902    1980-12-09    800     0    20
7788    SCOTT    ANALYST   7566    1982-12-22    3000    0    20
7876    ADAMS    CLERK    7788    1983-01-15    1100    0    20
7934    MILLER   CLERK    7782    1982-01-11    1300    0    10
```

라고 나옴.

저장후에 빠져 나오면

```
(base) [scott@localhost ~]$ vi emp*.txt

21 files to edit
```

라고 메시지가 뜸

문제 126. 다시 emp1.txt ~ emp20.txt를 모두 열어서 jjjj를 SALESMAN으로 변경하시오.

답

```
vi emp*.txt

:argdo %s/jjjj/SALESMAN/g | update
```

결과

```
7839    KING     PRESIDENT  0    1981-11-17    5000    0    10
7698    BLAKE    SALESMAN   7839    1981-05-01    2850    0    30
7782    CLARK    SALESMAN   7839    1981-05-09    2450    0    10
7566    JONES    SALESMAN   7839    1981-04-01    2975    0    20
7654    MARTIN   aaa     7698    1981-09-10    1250    1400    30
```

```

7499    ALLEN    bbbb    7698    1981-02-11    1600    300    30

7844    TURNER    SALESMAN    7698    1981-08-21    1500    0    30

7900    JAMES    CLERK    7698    1981-12-11    950    0    30

7521    WARD    SALESMAN    7698    1981-02-23    1250    500    30

7902    FORD    ANALYST    7566    1981-12-11    3000    0    20

7369    SMITH    CLERK    7902    1980-12-09    800    0    20

7788    SCOTT    ANALYST    7566    1982-12-22    3000    0    20

7876    ADAMS    CLERK    7788    1983-01-15    1100    0    20

7934    MILLER    CLERK    7782    1982-01-11    1300    0    10

(base) [scott@localhost ~]$ vi emp*.txt

21 files to edit

```

문제 127. jobs.txt를 jobs1.txt ~ jobs10.txt로 10개를 복사하시오.

답 

```

7839    KING    PRESIDENT    0    1981-11-17    5000    0    10

7698    BLAKE    SALESMAN    7839    1981-05-01    2850    0    30

7782    CLARK    SALESMAN    7839    1981-05-09    2450    0    10

7566    JONES    SALESMAN    7839    1981-04-01    2975    0    20

7654    MARTIN    aaa    7698    1981-09-10    1250    1400    30

7499    ALLEN    bbbb    7698    1981-02-11    1600    300    30

7844    TURNER    SALESMAN    7698    1981-08-21    1500    0    30

7900    JAMES    CLERK    7698    1981-12-11    950    0    30

7521    WARD    SALESMAN    7698    1981-02-23    1250    500    30

7902    FORD    ANALYST    7566    1981-12-11    3000    0    20

7369    SMITH    CLERK    7902    1980-12-09    800    0    20

7788    SCOTT    ANALYST    7566    1982-12-22    3000    0    20

7876    ADAMS    CLERK    7788    1983-01-15    1100    0    20

7934    MILLER    CLERK    7782    1982-01-11    1300    0    10

(base) [scott@localhost ~]$ vi emp*.txt

21 files to edit

```

결과 

```

(base) [scott@localhost ~]$ ls -l jobs*.txt

-rwxrwx---. 1 scott scott 12178 Jan 4 01:14 jobs10.txt
-rwxrwx---. 1 scott scott 12178 Jan 4 01:14 jobs1.txt
-rw-rw-r--. 1 scott scott 12178 Jan 4 01:14 jobs2.txt
-rw-rw-r--. 1 scott scott 12178 Jan 4 01:14 jobs3.txt
-rwxrwx---. 1 scott scott 12178 Jan 4 01:14 jobs4.txt
-rwxrwx---. 1 scott scott 12178 Jan 4 01:14 jobs5.txt
-rwxrwx---. 1 scott scott 12178 Jan 4 01:14 jobs6.txt
-rw-rw-r--. 1 scott scott 12178 Jan 4 01:14 jobs7.txt
-rwxrwx---. 1 scott scott 12178 Jan 4 01:14 jobs8.txt
-rwxrwx---. 1 scott scott 12178 Jan 4 01:14 jobs9.txt
-rwxrwx---. 1 scott scott 12178 Jan 3 23:52 jobs.txt

```

문제 128. jobs1.txt ~ jobs10.txt를 열어서 about를 aaa 로 변경하시오.

답 

```
vi jobs*.txt

:argdo %s/about/aaa/g | update
```

결과 

```
"jobs10.txt" [dos] 144L, 12160C written
"jobs1.txt" [dos format] 144 lines, 12178 characters
"jobs1.txt" [dos] 144L, 12160C written
"jobs2.txt" [dos format] 144 lines, 12178 characters
"jobs2.txt" [dos] 144L, 12160C written
"jobs3.txt" [dos format] 144 lines, 12178 characters
"jobs3.txt" [dos] 144L, 12160C written
"jobs4.txt" [dos format] 144 lines, 12178 characters
"jobs4.txt" [dos] 144L, 12160C written
"jobs5.txt" [dos format] 144 lines, 12178 characters
"jobs5.txt" [dos] 144L, 12160C written
"jobs6.txt" [dos format] 144 lines, 12178 characters
"jobs6.txt" [dos] 144L, 12160C written
"jobs7.txt" [dos format] 144 lines, 12178 characters
"jobs7.txt" [dos] 144L, 12160C written
"jobs8.txt" [dos format] 144 lines, 12178 characters
"jobs8.txt" [dos] 144L, 12160C written
"jobs9.txt" [dos format] 144 lines, 12178 characters
"jobs9.txt" [dos] 144L, 12160C written
"jobs.txt" [dos format] 144 lines, 12178 characters
"jobs.txt" [dos] 144L, 12160C written
(base) [scott@localhost ~]$ vi jobs*.txt

11 files to edit
```

■ 리눅스 수업 목차

1. 리눅스란 무엇인가
2. 리눅스 설치
3. 리눅스 기본명령어
4. 아나콘다 설치
5. vi 편집기 명령어
6. 권한관리 명령어

■ 006 권한관리 명령어



리눅스에 하둡을 설치하고 운영을 할때, 여러가지 문제들이 발생하는데 그중 많은 문제들이 권한에 관련한 오류들이 대부분이다.

그래서 하둡운영을 원활하게 하기위해서는 '권한관리명령어'를 잘 숙지하고 있어야 한다.

1. chmod → change mode
2. chown → change ownership of a file
3. chattr → change file attributes

권한 관리표의 사본

# 번호	Aa 권한	≡ 대표문자	≡ about 파일	≡ about 디렉토리
4	<u>읽기권한</u>	r	read, copy	디렉토리에서 ls 가능
2	<u>쓰기권한</u>	w	수정	디렉토리에서 파일생성 가능
1	<u>실행권한</u>	x	실행	디렉토리에서 cd로 접근가능

e.g) ls -l 로 특정 파일을 조회했을때 나오는 권한 부분을 해석

```
$ ls -l emp.txt
-rwxrwxrwx. 1 root root 1030 Jan 4 01:08 emp.txt
```

↑ ↑
1 2
권한에 대한 정보 그룹이름

해석

```
- r w x r w x r w x
↑ ↑ ↑ ↑
1 2 3 4
```

1: 디렉토리인지 아닌지. 디렉토리면 d로 나오고, - 면 파일을 의미

2: rwx : 파일 소유자의 권한 (읽기, 쓰기, 실행 모두가 가능하다는 뜻)

3: rwx: 파일의 그룹에 속한 유저들의 권한

4: rwx : 기타 유저들의 권한

emp.txt의 소유자는 root => 2

즉 root는 emp.txt 파일을 -r w x할 수 있다.

scott은 기타유저 => 4 에 그 권한이 표시되어 있다.

e.g) 숫자 1부터 10까지 출력하는 a20.py라는 파이썬 파일을 만드시오.

답

```
vi a20.py

for i in range(1,11):
    print(i)

python a20.py #py형식의 파일은 python을 앞에 입력해줘야 실행된다.
```

결과

```
# python a20.py
```

```
1
2
3
4
5
6
7
8
9
10
```

소유자 확인

```
$ ls -l a20.py
```

```
- rw-rw-r--. 1 scott scott 32 Jan 4 01:55 a20.py
- rw-rw-r--. 해석
- rw- rw- r--.
```

```
↑      ↑      ↑
```

```
1      2      3
```

1: - 는 파일을 의미

2: 소유자의 권한은 읽고(r) 쓰기(w) 만 가능 (읽고 쓰기만) // 실행권(x)는 없음.

3: 그룹유저(scott)들도 읽고(r) 쓰는(w) 권한만 있고 실행(x) 권한은 없다.

문제 130. a20.py 파일의 소유자가 실행할 수 있는 권한도 가질 수 있도록 하시오./

답 

```
chmod u+x a20.py
```

해석 : u(user) 에게 x(실행) 권한을 주어라. a20.py 파일에 대해서

결과 

```
(base) [scott@localhost ~]$ ls -l a20.py
```

```
- rwx rw- r--. 1 scott scott 32 Jan 4 01:55 a20.py
```

유저의 권한에 x 가 늘어난 것을 볼 수 있다.



※설명 :

$u+x$ = 소유자에게 실행권한을 주겠다.

$u-x$ = 소유자에게 실행권한을 뺏겠다.

$u+r$ = 소유자에게 읽기 권한을 주겠다.

$u-r$ = 소유자에게 읽기 권한을 뺏겠다.

$u+w$ = 소유자에게 쓰기 권한을 주겠다.

$u-w$ = 소유자에게 쓰기 권한을 뺏겠다.



한꺼번에 쓰기도 가능

$u+rw$ = 소유자에게 읽기, 쓰기, 실행 권한을 한번에 주겠다.

혼합해서 쓰기도 가능

$u+r-x$ = 소유자에게 읽기 권한은 주고, 실행권한은 뺏겠다.

문제 131. a20.py의 소유자가 a20.py를 읽고, 쓰고, 실행할 수 있는 권한을 모두 빼시오.

답

```
$ chmod u-rwx a20.py
```

결과

```
$ ls -l a20.py
- ---rw-r--. 1 scott scott 32 Jan 4 01:55 a20.py
-          ---          rw-          r--
```

↑ ↑ ↑ ↑

파일유형 유저권한 그룹 권한 기타 유저 권한

vi a20.py해서 들어가면

"a20.py" [Permission Denied]

라고 뜬.

권한이 없기 때문

문제 132. a20.py의 소유자가 a20.py를 읽고, 쓸 수 있도록 권한을 넣고 실행권한은 넣지 마시오.

답

```
$ chmod u+rw a20.py
```

결과 

```
$ ls -l a20.py  
- rw-rw-r--. 1 scott scott 32 Jan 4 01:55 a20.py
```

문제 133. a20.py의 소유자는 a20.py를 읽고, 쓰고, 실행할 수 있게 하고, 그룹유저들과 기타유저들은 아무 것도 못하게 하시오.

그룹유저 = g

기타유저 = 0

답 

```
$ chmod u+rx,g-rwx,o-rwx a20.py
```

결과 

```
$ ls -l a20.py  
- rwx-----. 1 scott scott 32 Jan 4 01:55 a20.py
```

*설명 : root 유저는 scott과는 다르게 모든 파일에 대해서 읽고, 쓰고, 실행할 수 있는 권한을 가질 수 있도록 chmod 명령어를 수행할 수 있다.

*설명 : 각 유저들별로 권한을 부여할 때는 , 로 명령어를 구분해준다. (띄어쓰기 금지)

!!u / g / o 는 순서가 정해져 있다. 바꿔 쓰면 실행되지 않는다. 순서를 바꿔쓰면 아래와 같이 에러가 뜬다.

```
$ chmod o+rx, u+rx, g+rx a20.py  
chmod: invalid mode: 'o+rx,'  
Try 'chmod --help' for more information.
```

문제 134. root 유저로 접속하시오.

답 

```
base) [scott@localhost ~]$ su -  
Password:  
안보이지만,  
비밀번호 : 1234
```

결과 

```
# whoami  
root  
# pwd
```

```
/root
```

*설명 : socat일때는 \$로 시작했었는데, root로 바꾸면 #로 시작한다.

문제 135. 다시 scott으로 접속하시오.

답

```
★문법: su - 유저명  
# su - scott ( - 를 기준으로 양옆 한칸씩 띄어야 한다)
```

결과

```
Last login: Mon Jan 4 01:54:49 EST 2021 on pts/0  
(base) [scott@localhost ~]$ whoami  
scott  
(base) [scott@localhost ~]$ pwd  
/home/scott
```

문제 136. 터미널 창에서 다시 root로 접속하고, scott의 집인 scott으로 이동하시오.

답

```
$ su -  
Password:  
Last login: Mon Jan 4 02:21:18 EST 2021 on pts/0  
(base) [root@localhost ~]# cd /home/scott
```

결과

```
# pwd  
/home/scott  
(base) [root@localhost scott]# ls -l a20.py  
- rwX----- 1 scott scott 32 Jan 4 01:55 a20.py
```

*설명 : root는 최고권한자이기 때문에 얼마든지 a20.py를 볼 수 있고, 권한도 변경할 수 있다.

문제 137. root가 a20.py의 소유자가 갖고 있는 읽고, 쓰고, 실행하는 권한을 모두 빼시오.

답

```
# pwd  
/home/scott ->위치확인  
# chmod u-rwx a20.py ->권한 빼기
```

결과

```
(base) [root@localhost scott]# ls -l a20.py -> 확인
- - - - - 1 scott scott 32 Jan 4 01:55 a20.py
```

문제 138. 다시 scott으로 접속해서 a20.py를 vi로 열어보시오

답

```
# su - scott

Last login: Mon Jan  4 02:21:25 EST 2021 on pts/0
(base) [scott@localhost ~]$ pwd

/home/scott

(base) [scott@localhost ~]$ whoami

scott

답
vi a20.py
```

결과

```
"a20.py" [Permission Denied]
```

라고 뚝!

* 하지만 소유자는 scott이기때문에 스스로 권한 부여를 다시할 수는 있다. // 소유자가 root 혹은 제3의 유저라면 소유권을 가진 유저로 접속해 변경해줘야 한다.

문제 139. scott 유저에서 a20.py에서 읽고, 쓰고, 실행할 수 있는 권한을 넣으시오.

답

```
$ chmod u+rwX a20.py
```

결과

```
(base) [scott@localhost ~]$ ls -l a20.py
-rwx----- 1 scott scott 32 Jan 4 01:55 a20.py
```

(오늘의 마지막 문제)문제 140 . 리눅스에서 spyder를 실행해서 아래의 통계문제를 풀고, 화면 캡처를 해서 카페에 답글로 올리시오

도박사의 동전을 10번 던져서 뒷면이 나오는 결과를 아래와 같이 출력되게 하시오.

동전을 10번 던졌을 때 뒷면이 나오는 횟수가 1번이 나올 확률은 신뢰구간 95%안에 없습니다.

동전을 10번 던졌을 때 뒷면이 나오는 횟수가 2번이 나올 확률은 신뢰구간 95%안에 있습니다.

동전을 10번 던졌을 때 뒷면이 나오는 횟수가 3번이 나올 확률은 신뢰구간 95%안에 있습니다.

동전을 10번 던졌을 때 뒷면이 나오는 횟수가 4번이 나올 확률은 신뢰구간 95%안에 있습니다.

동전을 10번 던졌을 때 뒷면이 나오는 횟수가 5번이 나올 확률은 신뢰구간 95%안에 있습니다.

동전을 10번 던졌을 때 뒷면이 나오는 횟수가 6번이 나올 확률은 신뢰구간 95%안에 있습니다.
 동전을 10번 던졌을 때 뒷면이 나오는 횟수가 7번이 나올 확률은 신뢰구간 95%안에 없습니다.
 동전을 10번 던졌을 때 뒷면이 나오는 횟수가 8번이 나올 확률은 신뢰구간 95%안에 없습니다.
 동전을 10번 던졌을 때 뒷면이 나오는 횟수가 9번이 나올 확률은 신뢰구간 95%안에 없습니다.
 동전을 10번 던졌을 때 뒷면이 나오는 횟수가 10번이 나올 확률은 신뢰구간 95%안에 없습니다.

코드 

```
import random

import numpy as np

def coin_avg_std(num):
    #동전 던지기의 평균과 표준편차 를 생성

    result=[]

    for i in range(num):

        cnt = 0

        for t in range(8):

            cnt += (random.randint(0,1))

        result.append(cnt)

    c_m = np.mean(result)

    c_s = np.std(result)

    return c_m, c_s

def coin_hypo(num):

    c_m,c_s = coin_avg_std(10000)      # 동전을 던지는 횟수 선택

    if c_m-c_s*1.96<=num<=c_m+c_s*1.96:

        return f'동전을 10번 던졌을 때 뒷면이 나오는 횟수가 {num}번이 나올 확률은 신뢰구간 95%안에 있습니다.'

    else:

        return f'동전을 10번 던졌을 때 뒷면이 나오는 횟수가 {num}번이 나올 확률은 신뢰구간 95%안에 없습니다.'

    for i in range(1,11):

        print(coin_hypo(i))
```

[21.01.05 리눅스 강의 Day 6]

■ 리눅스 수업 목차

1. 리눅스란 무엇인가? 리눅스를 왜 배워야 하는가?

답: 하둡을 이용하기 위해서 // 딥러닝 환경을 구축하기 위해서

(업무의 범위)데이터 분석가 < 데이터 사이언티스트

데이터 사이언티스트

1. 데이터 분석(기본)
2. 환경구축(리눅스설치,하둡설치,아나콘다설치)

3. 파이썬으로 데이터 분석 코딩

■ 리눅스 수업 목차

1. 리눅스 기본 명령어
2. 리눅스에 아나콘다 설치, 가상환경 만들기, spyder설치
3. 리눅스 vi 편집기
4. 리눅스 vi 편집기 명령어
5. 리눅스 권한관리 명령어(★중요!)
 - 1) chmod : 특정파일의 권한을 조정하는 명령어
 - 2) chown : 특정파일/디렉토리의 소유자를 조정하는 명령어
 - 3) chattr : root 유저만 권한을 조정할 수 있도록 설정하는 명령어
6. 리눅스 디스크 관리 명령어
7. 리눅스 프로세서 관리 명령어
8. 셸스크립트 작성법

문제 141. dept.txt에 권한이 어떻게 되어 있는지 확인하시오.

답

```
(base) [scott@localhost ~]$ ls -l dept.txt
```

결과

```
- rw-rw-rw-. 1 scott scott 104 Jan 2 20:20 dept.txt
- rwx(소유자권한) rwx(그룹유저권한) rwx.(기타사용자권한)
```

소유자(=scott) 과 그룹유저는 r(읽기),w(쓰기),x(실행) 권한이 있고, 기타사용자는 아무권한도 없다.

문제 142. dept.txt에서 권한이 아래와 같이 나타나게 하시오.

rw-r-xr-x. = 그룹유저의 w 권한을 빼고, 기타유저의 권한에서도 w를 빼는 것

답

```
$ chmod g-w,o-w dept.txt
```

결과

```
(base) [scott@localhost ~]$ ls -l dept.txt
-rw-r-xr-x. 1 scott scott 104 Jan 2 20:20 dept.txt
```

만약 권한 설정에 문제가 있다면 소유자가 root인데 scott으로 변경하려고 해서 그런것.

이때는 chown 명령어로 소유자 변경을 해줘야 한다.

chown 명령어



change owner 의 약어, 파일이나 디렉토리의 소유자를 변경하는 명령어



문법: chown 유저명:그룹명 파일명.확장자

e.g) emp.txt의 소유자를 확인하는 방법

```
ls -l emp.txt  
- rw-rw----. 1 root root 1030 Jan 4 01:08 emp.txt
```



소유자 정보

하나씩 뜯어보기!

- rw-rw----. : 권한

1 : 바로가기 개수

root : 소유자

root : 그룹명

1030 : 파일크기

Jan 4 01:08 : 생성 or 수정한날짜

emp.txt : 파일명

e.g2) 소유자 권한을 바꿀 수 있는 것은 root 이므로 root로 접속해서 소유자를 변경한다.

1. scott의 터미널에서 root 로 접속하기

답

```
$ su -
```

Password: 1234 (보이지 않지만 입력은 된다)

Last login: Mon Jan 4 02:22:13 EST 2021 on pts/0

2. pwd 하여 위치확인후 scott의 home로 이동해 소유자 변경

답

```
# cd /home/scott  
(base) [root@localhost scott]# chown scott:scott emp.txt  
  
(base) [root@localhost scott]# ls -l emp.txt  
- rw-rw----. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

*설명 :

```
chown          scott:scott  emp.txt  
               ↓           ↓  
            유저명   그룹명
```

그룹명?



그룹명이란, 권한관리를 쉽게하기 위해서 그룹을 생성해서 그룹으로 권한관리를 하는데 이 그룹의 이름.

예를 들면 카페등을 운영할때 특정 회원들을 우수회원으로 지정했을때, 우수회원이라는 그룹에 속해있으면 카페의 모든 게시판을 열람할 수 있는 권한을 주는 것.

문제 143. dept.txt의 소유자를 scott으로 변경하고 그룹도 scott으로 변경하시오.

답

```
# pwd
/home/scott
(base) [root@localhost scott]# chown scott:scott dept.txt
```

결과

```
# ls -l dept.txt
-rwxr-xr-x. 1 scott scott 104 Jan 2 20:20 dept.txt
```

문제 144. 다시 scott유저로 swith user(su) 하시오.

답

```
# su - scott
Last login: Mon Jan  4 02:26:11 EST 2021 on pts/0
```

결과

```
(base) [scott@localhost ~]$ whoami
scott
```

상위계정에서 하위계정으로 갈때는 p.w를 물어보지 않고 접속한다. (반대의 경우는 물어봄)

문제 145. emp.txt의 권한을 아래와 같이 나타나게 하시오.

```
ls -l emp.txt
-rwxr-xr-x 1 scott scott emp.txt
```

답

```
$ chmod g-w,o-w emp.txt
```

결과


```
$ ls -l emp.txt
-rwxr-xr-x. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

문제 146. dept.txt도 emp.txt와 똑같이 권한이 관리되게 하시오.

답

```
$ chmod g-w,o-w dept.txt
```

결과

```
$ ls -l dept.txt
-rwxr-xr-x. 1 scott scott 104 Jan 2 20:20 dept.txt
```

문제 147. dept.txt의 권한이 아래와 같이 나타나게 하시오.

```
-rwx-----. 1 scott scott 104 Jan 2 20:20 dept.txt
```

답

```
$ chmod g-rx,o-rx dept.txt
$ ls -l dept.txt
-rwx-----. 1 scott scott 104 Jan 2 20:20 dept.txt
```

★ u: user // g: group // o:others

■ 숫자로 권한을 변경하는 방법

권한 관리표의 사본

# 번호	Aa 권한	≡ 대표문자	≡ about 파일	≡ about 디렉토리
4	<u>읽기권한</u>	r	read, copy	디렉토리에서 ls 가능
2	<u>쓰기권한</u>	w	수정	디렉토리에서 파일생성 가능
1	<u>실행권한</u>	x	실행	디렉토리에서 cd로 접근가능

e.g) emp.txt의 권한을 유저,그룹, 기타유저 모두 읽을 수만 있도록 변경하시오

답 (어떤걸 빼고넣을지 헛갈려 하느니 차라리 다 지워버리고 넣고 싶은 것만 넣는것이 정확하다)

```
$ chmod u-rwx,g-rwx,o-rwx emp.txt
$ chmod u+r,g+r,o+r
```


결과

```
$ ls -l emp.txt
```

```
- r--r--r--. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

e.g2) 위의 작업을 숫자로 하는 방법

```
$ chmod u-rwx,g-rwx,o-rwx emp.txt  
$ ls -l emp.txt  
- - - - - . 1 scott scott 1030 Jan 4 01:08 emp.txt
```

답  (읽기 권한을 3종류의 유저에게 부여)

```
$ chmod 444 emp.txt
```

결과 

```
$ ls -l emp.txt  
- r--r--r--. 1 scott scott 1030 Jan 4 01:08 emp.txt  
*설명 : 읽기는 숫자4, 쓰기는 숫자2, 실행은 숫자 1이다.
```

문제 148. emp.txt에 대해서 아래와 같이 권한이 들어가게 하시오.

```
-rw-rw-rw-. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

답 

```
$ chmod 666 emp.txt
```

결과 

```
(base) [scott@localhost ~]$ ls -l emp.txt  
- rw-rw-rw-. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

*설명 : 읽기는 4, 쓰기는 2이므로 $4+2=6$ 이어서 666을 입력해준것.

문제 149. emp.txt 에 대해서 아래와 같이 권한이 들어가게 하시오.

```
-rwxrwxrwx. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

답 

```
$ chmod 777 emp.txt -> 모든 권한 부여
```

결과 

```
$ ls -l emp.txt  
- rwxrwxrwx. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

*설명 : $4+2+1 = 7$ 이니까 777을 입력해준다.

문제 150. emp.txt 에 대해서 아래와 같이 권한이 들어가게 하시오.

```
- --x--x--x. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

답 

```
$ chmod -666 emp.txt  
or  
$ chmod -777 emp.txt  
$ chmod 111 emp.txt
```

결과 

```
$ ls -l emp.txt  
- --x--x--x. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

문제 151. emp.txt에 대해서 아래와 같이 권한이 들어가게 하시오.

```
- rwx r-x --x. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

답 

```
$ chmod -777 emp.txt # 일단 모든 권한을 지운뒤  
$ chmod 751 emp.txt # 넣으려는 권한에 맞게 번호 입력
```

결과 

```
$ ls -l emp.txt  
- rwxr-x--x. 1 scott scott 1030 Jan 4 01:08 emp.txt  
  
rwx : 7  
r-x : 7-2 = 5  
- x : 7-4-2=1
```

문제 152. emp.txt에 대해서 아래와 같이 권한이 들어가게 하시오.

```
- ----- . 1 scott scott 1030 Jan 4 01:08 emp.txt
```

답 

```
$ chmod -777 emp.txt
```

결과 

```
$ ls -l emp.txt  
- - - - - . 1 scott scott 1030 Jan 4 01:08 emp.txt
```

문제 153. emp.txt에 아래와 같이 권한이 들어가게 하시오.

```
- rwxrw-rw-. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

답 

```
$ chmod 766 emp.txt
```

결과 

```
$ ls -l emp.txt  
- rwxrw-rw-. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

■ 디렉토리의 소유자 권한을 변경하는 방법



디렉토리의 소유자 권한을 변경하게 되면 그 디렉토리에 있는 모든 파일들의 소유자를 한번에 바꿀 수 있다.

e.g) scott유저에서 /home/scott 밑에 test501이라는 디렉토리를 생성하시오


답 

```
$ cd  
$ mkdir test501
```

e.g 2) test501 디렉토리 밑에 emp.txt와 dept.txt를 복사한다.

답 

```
$ cp emp.txt ../test501  
$ cp dept.txt ../test501
```

*설명 :  는 현재디렉토리 를 의미

결과 

```
$ cd  
$ cd test501  
$ ls -l *.txt  
- rwx----- 1 scott scott 104 Jan 4 21:11 dept.txt  
- rwxrw-r-- 1 scott scott 1030 Jan 4 21:11 emp.txt
```

"e.g 3) test501의 소유자가 누구인지 확인하시오.

답 

```
$ ls -ld test501
```

결과 

```
drwxrwxr-x. 2 scott scott 37 Jan  4 21:11 test501
```

*설명 : ls 명령어의 -l 옵션 중에 d 옵션(d=directory) 을 사용하면 디렉토리의 상세정보를 볼 수 있다.

```
drwxrwxr-x. 2 scott scott 37 Jan  4 21:11 test501
```



맨 앞에 d가 있는건 directory 라는 뜻.

소유자와 그룹유저 모두 scott 이다.

| e.g 4) test501 디렉토리 의 소유자를 root로 변경한다.

* root로 들어가서 해야 한다.

답 

1. root로 접속

```
$ su -  
Password:  
  
Last login: Mon Jan  4 20:09:02 EST 2021 on pts/0
```

2. test501이 있는 scott의 home로 이동해서 test501 의 소유자 확인

```
# cd /home/scott  
  
# ls -ld test501  
  
drwxrwxr-x. 2 scott scott 37 Jan  4 21:11 test501
```

3. 소유자 변경

```
# chown root:root test501
```

결과 

```
# ls -ld test501  
  
drwxrwxr-x. 2 root root 37 Jan  4 21:11 test501
```

그러나!

```
# cd test501  
  
# ls -l  
  
total 8  
  
- rwX----- 1 scott scott 104 Jan 4 21:11 dept.txt  
- rwxrw-r-- 1 scott scott 1030 Jan 4 21:11 emp.txt  
  
emp.txt // dept.txt의 소유자는 여전히 scott인 것을 확인할 수 있다.
```

*설명 : test501 디렉토리의 소유자는 root로 변경했지만 test501 디렉토리 안에 있는 emp.txt와 dept.txt의 소유자는 scott으로 그대로 유지되고 있다.

★e.g 5) 만약 test501 디렉토리의 소유자를 root로 변경하면서 test501 디렉토리에 있는 모든 텍스트 파일의 소유자도 같이 root로 변경하고 싶다면? -R 옵션을 사용해준다!

답

```
# cd /home/scott
# chown -R root:root test501
```

결과

```
# ls -ld test501
drwxrwxr-x. 2 root root 37 Jan  4 21:11 test501

# cd test501
# ls -l
total 8
-rwx-----. 1 root root 104 Jan 4 21:11 dept.txt
-rwxrw-r--. 1 root root 1030 Jan 4 21:11 emp.txt

emp.txt와 dept.txt 의 소유자도 root 으로 변경된 것을 확인할 수 있다.
```

※설명 : -R 옵션을 붙이면 디렉토리의 소유자 뿐만 아니라 디렉토리 아래에 있는 모든 파일과 하위 디렉토리의 소유자도 한번에 변경할 수 있다.

문제 154. /home/scott 밑에 있는 test501 디렉토리와 그 밑에 있는 파일들의 소유자를 모두 scott으로 변경하시오.

답

```
# cd /home/scott
# chown -R scott:scott test501
```

결과

```
# ls -ld test501
drwxrwxr-x. 2 scott scott 37 Jan  4 21:11 test501

# cd test501
# ls -l
total 8
-rwx-----. 1 scott scott 104 Jan 4 21:11 dept.txt
-rwxrw-r--. 1 scott scott 1030 Jan 4 21:11 emp.txt
```

chattr 명령어



★ 문법: `chattr +i` 파일명.확장자 (`chmod`를 못하게 막을때)

`chattr -i` 파일명.확장자(막아둔 `chmod`금지 권한을 풀때)

잘 적용되었는지 결과 확인은

`lsattr` 파일명.확장자

`chattr` 명령어를 이용하면 `chmod` 명령어 수행을 막을 수 있다.

`chattr` 명령어는 최상위 계정인 `root`에서만 수행할 수 있습니다.

※ 보통 `root`의 패스워드는 회사에서는 극소수의 인원만 알고 있다.

일반 사원들은 `scott` 같은 일반 유저의 패스워드만 알고 사용할 수 있다.

(현재 `scott`유저만 있기 때문에)

`scott` 유저로 특정파일에 대해서 `chmod` 명령어를 수행하지 못하도록 막는 명령어

e.g)

```
#su - 로 root로 이동
#cd /home/scott 으로 scott의 home로 이동

# chattr +i emp.txt

# lsattr emp.txt

- ---i----- emp.txt

lsattr emp.txt 해서 위와 같이 결과가 나오면 emp.txt에 대해서 root유저외에 다른 유저는 삭제, 변경( 권한변경포함), 내용추가 등을 할 수 없다.

확인

# su - scott

Last login: Mon Jan  4 20:18:30 EST 2021 on pts/0

$ ls -l emp.txt

- rwxrw-rw-. 1 scott scott 1030 Jan 4 01:08 emp.txt

$ chmod 777 emp.txt

chmod: changing permissions of 'emp.txt': Operation not permitted
```

※설명 : 소유자가 `scott`(자기자신)인데도, `chmod` 를 사용할 수가 없다.

```
#cat emp.txt
```

```
#vi emp.txt
```

로 읽거나 접속은 가능하지만

`vi emp.txt`에서 문서를 편집하고 저장하려고 하면

E45: 'readonly' option is set (add ! to override) 라는 에러가 뜨면서 수정이 되지 않는다.

e.g 2) 그러면 다시 `scott`이 `emp.txt` 를 `chmod`할 수 있도록 설정 하시오.

```
1. root 유저로 접속

$ su -

Password:

Last login: Mon Jan  4 21:15:59 EST 2021 on pts/0
```

```

2. 해제할 파일 위치로 이동하여 해제 (-i) 옵션 사용

# cd /home/scott

# chattr -i emp.txt

#lsattr emp.txt

- ----- emp.txt

t

chattr +i emp.txt 했을때 ----i----- emp.txt 의 i가 사라졌다.

3. chmod 사용해보기

# su - scott

Last login: Mon Jan  4 21:50:01 EST 2021 on pts/0

$ chmod 777 emp.txt

$ ls -l emp.txt

$ ls -l emp.txt

- rwxrwxrwx. 1 scott scott 1030 Jan 4 01:08 emp.txt

```

※설명 : chattr 명령어는 언제 유용한가?

이 파일은 우리회사에서 너무나도 중요한 파일이어서 root 만 변경할 수 있고 나머지 유저는 읽기만 할 수 있도록 하고 싶을때 유용하다.

문제 155. emp.txt에서 아래와 같이 권한이 유지되게 하고 root 유저외에는 chmod를 할 수 없도록 막으시오.

`$ls -l emp.txt` 를 입력했을때 아래와 같이 결과가 나오도록!
`-r--r--r--. 1 scott scott 1030 Jan 4 01:08 emp.txt`

답 

```

1. scott에서 chmod로 권한변경

$ chmod -777 emp.txt

$ ls -l emp.txt

- -----. 1 scott scott 1030 Jan 4 01:08 emp.txt

$ chmod 444 emp.txt

$ ls -l emp.txt

- r--r--r--. 1 scott scott 1030 Jan 4 01:08 emp.txt

2. chattr 사용하기 위해 root로 이동 해 chattr 적용

$ su -

Password:

Last login: Mon Jan  4 22:00:37 EST 2021 on pts/0

# cd /home/scott

# chattr +i emp.txt

# lsattr emp.txt

- ---i----- emp.txt

```

3. 결과 확인


```
# su - scott

Last login: Mon Jan  4 22:01:03 EST 2021 on pts/0

$ cd

$ chmod 777 emp.txt

chmod: changing permissions of 'emp.txt': Operation not permitted
```

문제 156. 다시 emp.txt를 scott유저가 chmod할 수 있도록 변경하시오.

답 

```
1. root 로 접속

$ su -

Password:

Last login: Mon Jan  4 22:01:48 EST 2021 on pts/0

2. scott에 접속해서 금지한 권한 해제

# cd /home/scott

# chmod -i emp.txt

# lsattr emp.txt

- ----- emp.txt

3. 다시 scott으로 이동하여 확인

# su - scott

Last login: Mon Jan  4 22:02:15 EST 2021 on pts/0

$ cd

$ chmod 777 emp.txt
```

결과 

```
$ ls -l emp.txt

-rwxrwxrwx. 1 scott scott 1030 Jan 4 01:08 emp.txt
```

문제 157. dept.txt에 대해서 chmod를 할 수 없도록 막으시오. + 권한이 아래와 같도록 하시오.

```
- rw-r--r--. 1 scott scott 104 Jan 2 20:20 dept.txt
```

```
1. chmod로 권한 변경

$ chmod -777 dept.txt

$ chmod 644 dept.txt

$ ls -l dept.txt

-rw-r--r--. 1 scott scott 104 Jan 2 20:20 dept.txt

1. chmod 적용위해 root로 이동

$ su -
```

```
Password:

Last login: Mon Jan  4 22:07:48 EST 2021 on pts/0

# cd /home/scott

# chattr +i dept.txt

# lsattr dept.txt

- ---i----- dept.txt
```

```
1. 결과확인 위해 scott으로 이동

# su - scott

Last login: Mon Jan  4 22:08:05 EST 2021 on pts/0

$ cd

$ chmod 777 dept.txt

chmod: changing permissions of 'dept.txt': Operation not permitted
```

(점심시간문제)

문제 158. 숫자 1부터 20까지 출력하는 파이썬 코드를 작성하고 그 파이썬 코드의 권한이 아래와 같이 들어가게 하시오.

```
python>
for i in range(1,21):

print(i)
```

답 

```
1. vi 편집기로 파이썬 파일 생성하고, 1~20까지 출력하는 파이썬 코드 생성

$ vi a158.py

결과확인

$ python a158.py

1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```

15
16
17
18
19
20
2) 만들어진 파일에 대한 권한 리셋후 문제에 제시된대로 설정하기

$ chmod -777 a158.py

$ ls -l a158.py
- - - - - . 1 scott scott 32 Jan 4 22:14 a158.py

$ chmod 766 a158.py

$ ls -l a158.py
- rwxrw-rw-. 1 scott scott 32 Jan 4 22:14 a158.py
3) root 계정으로 이동해 chattr 적용

$ su -

Password:

Last login: Mon Jan  4 22:09:46 EST 2021 on pts/0

# cd /home/scott

# chattr +i a158.py

# lsattr a158.py
- ---i----- a158.py
4) 결과 확인위해 scott 계정으로 이동해 chmod실행

# su - scott

Last login: Mon Jan  4 22:10:07 EST 2021 on pts/0

$ cd

$ chmod 777 a158.py

chmod: changing permissions of 'a158.py': Operation not permitted

```

7. 리눅스 디스크(disk) 관리 명령어



disk : 데이터를 저장하는 저장 영역 (컴퓨터의 하드디스크 등)

크게 3가지만 잘 알면 된다.

1. df 명령어
2. du 명령어
3. sar 명령어

1) df 명령어



"현재 파일 시스템의 총 사용율을 사용하는 명령어"

e.g) `$ df -h // -h : human` → human이 보기 편하도록 size를 M 단위로 표기하라는 뜻

결과 

```
$ df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
devtmpfs	903M	0	903M	0%	/dev
tmpfs	919M	0	919M	0%	/dev/shm
tmpfs	919M	9.5M	910M	2%	/run
tmpfs	919M	0	919M	0%	/sys/fs/cgroup
/dev/mapper/centos-root	50G	9.5G	41G	19%	/
/dev/mapper/centos-home	27G	5.1G	22G	19%	/home
/dev/sda1	1014M	239M	776M	24%	/boot
data	477G	101G	377G	22%	/media/sf_data
/dev/sr0	59M	59M	0	100%	/run/media/scott/VBox_GAs_6.1.16
tmpfs	184M	36K	184M	1%	/run/user/1000

Size : 전체 용량
Used : 사용된 용량
Avali : 사용가능 용량


유의해서 봐야할 곳은 -> /dev/mapper/centos-root 50G 9.5G 41G 19% / 이 부분이 리눅스가 설치되어진 곳

2) du 명령어



" 현재 파일/디렉토리의 디스크 상요량을 표시하는 명령어"

e.g) 현재 디렉토리의 텍스트 파일들의 총 크기를 구하려면?

 숫자 부분이 크기

```
$ du *.txt
```

4	10.txt
4	20.txt
4	30.txt
0	a1.txt
4	DALLAS.txt
4	dept.txt
4	detpno.txt
12	jobs.txt
4	list.txt

```

.
.
.
4      salesman.txt

```

총 합계는 du -c 를 입력

```

base) [scott@localhost ~]$ du -c *.txt

4      10.txt
4      20.txt
4      30.txt
0      a1.txt
4      DALLAS.txt
4      dept.txt
.
.
.
4      list.txt
4      salesman.txt
260    total  ◀ 모든 txt 파일의 크기의 합이 260 byte 임을 알 수 있다.

```

문제 159. /home/scott 밑에 있는 확장자가 .sh 로 끝나는 파일들의 총 크기를 구하시오.

답 

```
$ du -c *.sh
```

```

541540      Anaconda3-2020.11-Linux-x86_64.sh
4           a.sh
4           find_dept_info.sh
4           find_file.sh
4           find_job.sh
4           find_loc.sh
4           find_sal.sh
4           fin_loc.sh
결과↓
541568      total  byte

```

만약 Mega 로 보고 싶다면? h를 붙여준다.

```
$ du -ch *.sh
```

```

529M      Anaconda3-2020.11-Linux-x86_64.sh
4.0K      a.sh
4.0K      find_dept_info.sh

```

```

4.0K      find_file.sh
4.0K      find_job.sh
4.0K      find_loc.sh
4.0K      find_sal.sh
4.0K      fin_loc.sh
529M      total

```

3) sar 명령어



"디스크의 사용량을 확인하는 명령어"

작업을 하다가 리눅스 서버가 너무 느려서 작업이 진행이 잘 안될때 자주 사용한다.

e.g) `$ sar 1 100`

*설명 : 1번에서 100번까지 출력하겠다는 것

시간대 별로 작업에 사용된 디스크 사용량이 순차적으로 출력되는데, 이를 통해 언제 과부하가 되었는지 확인할 수 있다.

`$ sar 1 100`

```

Linux 3.10.0-1160.11.1.el7.x86_64 (localhost.localdomain)      01/05/2021      _x86_64_      (1 CPU)
12:00:48 AM  CPU      %user      %nice      %system      %iowait      %steal      %idle
12:00:49 AM  all        0.99        0.00        0.99        0.00        0.00      98.02
12:00:50 AM  all        0.00        0.00        0.00        0.00        0.00     100.00
12:00:51 AM  all        2.02        0.00        0.00        0.00        0.00      97.98
12:00:52 AM  all        3.12        0.00        1.04        0.00        0.00      95.83
12:00:53 AM  all        3.09        0.00        0.00        0.00        0.00      96.91
12:00:54 AM  all        1.01        0.00        0.00        0.00        0.00      98.99
12:00:55 AM  all        2.06        0.00        1.03        0.00        0.00      96.91
12:00:56 AM  all        3.03        0.00        1.01        0.00        0.00      95.96
12:00:57 AM  all        3.19        0.00        1.06        0.00        0.00      95.74
.
.
.

```

*설명 :

i/o = input / output , 컴퓨터 및 주변장치에 대하여 데이터를 전송하는 프로그램, 운영 혹은 장치를 일컫는 말

%User : scott 유저와 같이 일반유저가 사용하는 disk i/o 의 양

악성 SQL 이나 무한루프를 수행하면 %User의 사용율이 올라간다.

%nice : cpu를 양보하는 친절도 // 간혹 너무 친절하면 작업이 잘 안돌아 간다. 이럴때는 친절도를 낮춰줘야 한다.

%system : 시스템이 사용하는 disk i/o

%iowait : i/o 를 일으키면서 얼마나 대기하는지

%steal : 다른 프로세서의 자원을 얼마나 뺏고 있는지

%idle : 작업을 안하고 있는 idle한 상태

%User 부분을 집중적으로 모니터링 하면 된다.

만약 리눅스 서버에 부하가 심한 작업을 수행하면 %User 부분이 사용율이 100에 가깝게 올라간다.

※ 어떤 작업들이 부하를 주는가?(예시)

1. 악성 SQL 을 수행

2. 무한 루프(python while 문)

지금은 사용량이 많지 않아 사용량이 크지 않다.

무한 루프를 돌려서 어떻게 변하는지 확인해보기

문제 160. 아래와 같이 1부터 100000000000 숫자를 출력하는 파이썬 프로그램을 수행하고 , 다른 터미널 창을 하나 더 열어

sar로 사용량을 모니터링 하시오.

답

```
vi a.py

python>

for i in range(1,100000000001):

print(i)
```

결과

```
Linux 3.10.0-1160.11.1.el7.x86_64 (localhost.localdomain)      01/05/2021      _x86_64_      (1 CPU)

12:11:07 AM      CPU      %user      %nice      %system      %iowait      %steal      %idle
12:11:08 AM      all      64.58      0.00      35.42      0.00      0.00      0.00
12:11:09 AM      all      63.64      0.00      36.36      0.00      0.00      0.00
12:11:10 AM      all      65.31      0.00      34.69      0.00      0.00      0.00
12:11:11 AM      all      65.66      0.00      34.34      0.00      0.00      0.00
12:11:12 AM      all      63.92      0.00      36.08      0.00      0.00      0.00
12:11:13 AM      all      64.00      0.00      36.00      0.00      0.00      0.00
12:11:14 AM      all      61.22      0.00      38.78      0.00      0.00      0.00
12:11:15 AM      all      64.29      0.00      35.71      0.00      0.00      0.00
12:11:16 AM      all      64.29      0.00      35.71      0.00      0.00      0.00
12:11:17 AM      all      63.27      0.00      36.73      0.00      0.00      0.00
.
.
.

사용량이 급격히 증가한 것을 볼 수 있다.

ctrl + c를 누르면 중단된다.

^C

Traceback (most recent call last):

File "a.py", line 2, in <module>

print(i)
```

KeyboardInterrupt

중단 된 후의 사용량 체크

]\$ sar 1 100

```
Linux 3.10.0-1160.11.1.el7.x86_64 (localhost.localdomain)      01/05/2021      _x86_64_      (1 CPU)

12:12:15 AM      CPU      %user      %nice      %system      %iowait      %steal      %idle
12:12:16 AM    all         1.02         0.00         0.00         0.00         0.00      98.98
12:12:17 AM    all         2.08         0.00         1.04         0.00         0.00      96.88
12:12:18 AM    all         4.26         0.00         1.06         0.00         0.00      94.68
12:12:19 AM    all         3.09         0.00         0.00         0.00         0.00      96.91
12:12:20 AM    all         2.06         0.00         1.03         0.00         0.00      96.91
12:12:21 AM    all         1.01         0.00         1.01         0.00         0.00      97.98
12:12:22 AM    all         6.67         0.00         1.11         0.00         0.00      92.22
.
.
.
```

사용량이 현격히 줄어든 것을 확인할 수 있다.

문제 161. 내가 리눅스 shell 프로그램을 만들고 돌린후에 회의를 가거나 퇴근을 해서 shell 프로그램이 시스템에 부하를 일으키고 있는지 확인할 수 없다면 어떻게 해야 하는가? 퇴근도 해야하고! 회의도 가야하니까! 하지만 일도 해야 하고!

답

sar 명령어로 수행한 결과를 text 파일로 생성되게 한다. (redirection 사용)

```
$ sar 1 20 >> sal_20210105.txt
```

딜레이가 조금 있다가

완료가 되면 터미널 입력커서가 다시 생신다.

자료 확인

```
vi sal_20210105.txt
```

이때 vi sal 까지만 쓰고 tap 키를 누르면 자동으로 파일명이 입력된다.

결과

```
Linux 3.10.0-1160.11.1.el7.x86_64 (localhost.localdomain)      01/05/2021      _x86_64_      (1 CPU)

12:19:21 AM      CPU      %user      %nice      %system      %iowait      %steal      %idle
12:19:22 AM    all         4.04         0.00         1.01         0.00         0.00      94.95
12:19:23 AM    all         0.00         0.00         0.00         0.00         0.00     100.00
12:19:24 AM    all         0.00         0.00         0.00         0.00         0.00     100.00
12:19:25 AM    all         6.25         5.21         1.04         0.00         0.00      87.50
12:19:26 AM    all         6.19         0.00         1.03         0.00         0.00      92.78
12:19:27 AM    all         2.13         0.00         1.06         0.00         0.00      96.81
12:19:28 AM    all         4.12         0.00         2.06         0.00         0.00      93.81
12:19:29 AM    all         1.01         0.00         0.00         0.00         0.00      98.99
12:19:30 AM    all         0.00         0.00         0.00         0.00         0.00     100.00
12:19:31 AM    all         0.00         0.00         0.00         0.00         0.00     100.00
```



```


12:19:32 AM    all     0.00     0.00     0.00     0.00     0.00    100.00
12:19:33 AM    all     1.00     0.00     0.00     0.00     0.00     99.00
12:19:34 AM    all     3.09     0.00     1.03     0.00     0.00     95.88
12:19:35 AM    all     5.32     0.00     2.13     0.00     0.00     92.55
12:19:36 AM    all     5.21     0.00     1.04     0.00     0.00     93.75
12:19:37 AM    all     4.21     0.00     1.05     0.00     0.00     94.74
12:19:38 AM    all     4.12     0.00     1.03     0.00     0.00     94.85
12:19:39 AM    all     3.06     0.00     0.00     0.00     0.00     96.94
12:19:40 AM    all     0.00     0.00     0.00     0.00     0.00    100.00
12:19:41 AM    all     0.00     0.00     2.00     0.00     0.00     98.00
Average:      all     2.46     0.26     0.72     0.00     0.00     96.57

$ ls -l sal_20210105.txt
-rw-rw-r--. 1 scott scott 1849 Jan 5 00:19 sal_20210105.txt

```

실행이 완료가 된 후에 새로운 입력창이 열리는데 만약 sar 1 20이 아니라 100 , 1000으로 해두면 오랜 시간이 걸린다.
그래서 이를 백그라운드에서 작업하게 하는 방법!

문제 162. 만약에 위의 명령어를 백그라운드로 돌아가게 하고 싶다면?

답  같은 명령어의 맨 끝에 앤퍼센트('&') 를 붙여 준다.

```
$ sar 1 20 >> sal_20210105.txt &
```

결과 

[1] 19924

결과가 출력되면 아래와 같이 결과가 나온다.

```

$
[1]+  Done                  sar 1 20 >> sal_20210105.txt

```

8. 리눅스 프로세서 관리 명령어



ps 명령어 : " 현재 시스템에서 수행되고 있는 프로세서의 정보를 표시하는 명령어"

top 명령어 :

kill 명령어 :

jobs 명령어 : 동작중인 작업의 상태를 확인하는 명령어

● jobs 명령어

" 동작중인 작업의 상태를 확인하는 명령어"

* 상태정보 4가지

running : 실행중

stopped : 일시중단 중

Done : 종료

terminated : 강제종료됨

| e.g)

\$vi hhh.txt 를 만들고 들어가서

select ename, sal, job, deptno

from emp

where

까지만 입력하고 esc 누른뒤 ctrl+z 눌러서 빠져나온다.

결과

```
[1]+  Stopped                  vim hhh.txt 라고 뜬다.
[1]          +                  Stopped                  vim hhh.txt 라고 뜬다.
↑          ↑          ↑          ↑
job 번호    현재 진행중 이었던 job을 의미    일시중단중    파일명
vi hhh.txt로 열었지만 vim hhh.txt로 출력된다.
```

| 여기서 이 작업을 이어서 다시 하고 싶다면?

← 현재 진행중이었던 job으로 진입하는 명령어를 입력하면 된다.

원래 작업하던 vi hhh.txt 의 편집기창으로 진입된다.

저장을 정상적으로 하고 나와서 \$jobs 를 입력하면 이전 처럼 vims~ 가 나오지 않는다.

[21.01.06 리눅스 강의 Day 7]

9. Shell script 작성법

if 조건문에 들어가는 비교연산자

- 문자열 비교

1) "문자열1"=="문자열2" ⇒ 두 문자열이 같으면 True → [[]] 로 2개를 둘러줘야 함. <- if 문 사용시!

"문자열1" = "문자열2" ⇒ 두 문자열이 같으면 True → [] 로 1개를 둘러줘야 함. <- if 문 사용시!

2) "문자열1"!="문자열2" ⇒ 두 문자열이 같지 않으면 False

숫자열 비교의 사본

#	순번	Aa ★문법:	≡ 의미	≡ 수학기호	≡ note
1		숫자1 -eq 숫자2	두 숫자가 같으면 True	=	띄어쓰기 유의
2		숫자1 -ne 숫자2	두 숫자가 같지 않으면 False	!=	띄어쓰기 유의
3		숫자1 -gt 숫자2	숫자 1이 숫자2보다 크다면 True	>	띄어쓰기 유의
4		숫자1 -ge 숫자2	숫자 1이 숫자2보다 크거나 같다면 True	>=	띄어쓰기 유의
5		숫자1 -lt 숫자2	숫자 1이 숫자2보다 작다면 True	<	띄어쓰기 유의
6		숫자1 -le 숫자2	숫자1이 숫자2보다 작거나 같으면 True	<=	띄어쓰기 유의
7		!숫자1	숫자1이 거짓이 아니라면 True		띄어쓰기 유의

e.g) `$ vi if1.sh`

- 띄어쓰기 주의

```
if [ 100 -eq 200 ]; then
echo "100 and 200 are equal."

else

echo "100 and 200 are not equal."

fi
```

!?: if문을 작성할때는 중간에는 if가 아닐 경우의 실행값인 `else` 를 그리고 맨 마지막에는 `fi` 를 꼭 붙여줘야 한다.

결과 ↓

`$ sh if1.sh`

```
100 and 200 are not equal.
```

문제 147. 위의 스크립트에 파라미터 변수를 이용해서 아래와 같이 실행될 수 있도록 하시오.
파라미터 변수 = \$0,\$1,\$2.....\$n 을 의미

```
    $1  $2
    ↓   ↓
$ sh if1.sh 100 200
```

```
100 and 200 are not equal.
```

답 ↓

```
$ vi if1.sh 로 편집기에 진입 후

if [ $1 -eq $2 ]; then

echo "$1 and $2 are equal."

else

echo "$1 and $2 are not equal."

fi
```

결과 

```
$ sh if1.sh 100 200
```

```
100 and 200 are not equal.
```

● 리눅스의 논리 연산자의 사본

# 번호	Aa 의미	≡ 기호
1	<u>and</u>	&& / -a
2	<u>or</u>	/ -o
3	<u>not</u>	!

e.g)

```
and
```



```
if [ $sal -lt 2000 ] && [ $job=="SALESMAN" ]; then ~~
```

또는

```
if [ $sal -lt 2000 -a $job=="SALESMAN" ]; then ~~
```

*설명 :

&&를 써서 and를 표현할때는 조건 별로 [] 를 두르지만

- a 를 써서 and를 표현할때는 를 양 끝에 1개만 두르면 된다.

★ Tip.s

As @Renich suggests (but with an important typo that has not been fixed unfortunately), you can also use extended globbing for pattern matching. So you can use the same patterns you use to match files in command arguments (e.g. ls *.pdf) inside of bash comparisons.

For your particular case you can do the following.

```
if [[ "${cms}" != @(wordpress|magento|typo3) ]]
```

The @ means "Matches one of the given patterns". So this is basically saying cms is not equal to 'wordpress' OR 'magento' OR 'typo3'. In normal regular expression syntax @ is similar to just ^(wordpress|magento|typo3)\$.

Mitch Frazier has two good articles in the Linux Journal on this Pattern Matching In Bash and Bash Extended Globbing.

For more background on extended globbing see Pattern Matching (Bash Reference Manual).

<https://www.linuxjournal.com/content/pattern-matching-bash>

<https://www.linuxjournal.com/content/bash-extended-globbing>

문제 175. emp.txt에서 SCOTT의 월급을 출력하시오.

awk 이용!

\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

답

```
$ awk '$2==toupper("scott") {print $6}' emp.txt
```

결과

```
3000
```

문제 176. 위의 스크립트와 파라미터 변수를 이용해서 아래와 같이 실행되는 shell script를 작성하시오.

```
$ sh find_sal.sh scott
```

The salary scott is 3000

답

```
$ vi find_sal.sh 로 편집기에 진입후

sal=`awk -v name=$1 '$2==toupper(name) {print $6}' emp.txt`

echo "The salary $1 is $sal"
```

결과

```
$ sh find_sal.sh scott
```

```
The salary scott is 3000
```

문제 177. 위의 스크립트를 이용해서 아래와 같이 실행되게 shell script를 생성하시오.

\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

```
$sh find_job.sh allen
```

Allen's job is SALESMAN

답

이름을 입력했을때 직업을 출력하는 스크립트

```
job=`awk -v name=$1 '$2==toupper(name) {print $3}' emp.txt`
```

```
$vi find_job.sh 로 편집기에 진입 후

job=`awk -v name=$1 '$2==toupper(name) {print $3}' emp.txt`

echo "$1's job is $job"
```

결과

```
$ sh find_job.sh allen
```

```
allen's job is SALESMAN
```

문제 178. 위의 스크립트와 if문을 사용하여 아래의 shell script를 생성하시오.
월급을 2000을 기준으로 월급의 인상여부를 출력하게 하시오.

```
sh fin_sal.sh scott
```

월급 인상 대상자가 아닙니다.

답

```
이름 넣으면 sal이 올 검색하여 sal에 넣는 스크립트

sal=`awk -v name=$1 '$2==toupper(name) {print $6}' emp.txt`

+

if문 작성

답↓

sal=`awk -v name=$1 '$2==toupper(name) {print $6}' emp.txt`

if [ $sal -lt 2000 ]; then

echo "$1 is eligible for a raise"

else

echo "$1 is not eligible for a raise"

fi
```

결과

```
$ sh find_sal.sh martin

martin is eligible for a raise
```

```
$ sh find_sal.sh scott

scott is not eligible for a raise
```

문제 179. 위의 스크립트를 수정해서 부서번호가 30번이고 월급이 2000보다 작다면
"~is eligible for a raise" 라고 출력하고 그렇지 않다면 "~is not eligible for a raise" 라고 출력하시오.
\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

답

```
$vi find_sal.sh

sal=`awk -v name=$1 '$2==toupper(name) {print $6}' emp.txt`

deptno=`awk -v name=$1 '$2==toupper(name) {print $8}' emp.txt`

if [ ${sal:0:4} -lt 2000 -a ${deptno:0:2} -eq 30 ]; then

echo "월급 인상 대상자 입니다."

else

echo "월급 인상 대상자가 아닙니다."

fi
```

```
$sh find_sal.sh scott
```

```
$sh find_sal.sh martin
```

리눅스 shell에서 loop 문 사용법



문법

```
for qstn in value1,value2,value3
```

```
do
```

```
반복할문장
```

```
done
```

문제 180. 구구단 2단을 출력하시오.

답

`vi gop.sh`

```
for i in {1..19}
do
let gop=2*$i
echo "2 X $i = $gop"
done
```

결과

`$ sh gop.sh 2`

```
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
```

이중 루프문

```
for i in {2..9}
do # for 문이 하나 끝날때마다 do 를 붙여 준다.
for k in {1..9}
do
echo "$i $k" # 각 for문의 i와 k를 둘다 적어줘야한다.
done
```

```
done # done는 맨 밑에 두개 연달아
```

문제 181. 구구단 전체 출력하기

```
vi gugudan.sh
```

```
for i in {2..9}
do
  for k in {1..9}
  do
    let gugudan=$i*$k
    echo "$i X $k = $gugudan"
  done
done
```

결과

```
$ sh gugudan.sh
```

```
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
.
.
.
9 X 7 = 63
9 X 8 = 72
9 X 9 = 81
```

문제 182. 아래의 별표를 출력하는 shell script를 작성하시오.

```
★
★★
★★★
★★★★
★★★★★
```

답

```
$ sh star.sh
```

```
star=""
for i in {1..5}
do
```



```
star="$star★"

echo $star


done
```

결과 

```
$ sh star.sh
```

```
★
★★
★★★
★★★★
★★★★★
```

*설명 : 더블 쿼테이션 두개("") 로 null값을 나타냄. // 파이썬에서 cnt=0 으로 하는 것과 같은 원리

 do ⇒ loop 문 시작 // done ⇒ loop문 종료

 파이썬 문법은 # star=star *'★' 이지만 리눅스에서는 star="\$star★"

문제 183. 위의 코드를 수정해서 숫자를 물어보게 하고 숫자를 입력하면 해당 숫자만큼 ★이 출력되게 하시오.

답 

```
echo -n "숫자를 입력하세요~" # 여기서 숫자를 입력하면

read num # 그 숫자를 num으로 받아라.

star="" # star는 null값으로 비워둔 상태에서 시작

for i in `eval echo {1..$num}` #for 문의 범위는 1부터 num(입력한 숫자) 까지 이다.

do

star="$star★" # star 는 위에서 지정한 star 값(=null) 에 ★을 더해라

echo $star # for문으로 반복된 ★ 을 출력하라. (★ 이 1개일때 출력...★이 2개일때 출력... ★이 n개일때 출력

done
```

*설명 : for 반복문 안에서 `echo {1..$num}` 이 실행되게 하려면 `eval` 을 써줘야 한다.

문제 184. emp.txt를 복사해서 emp1.txt ~ emp100.txt 로 복사하는 shell script를 작성하시오.

답 

```
vi cp_emp.sh
```

```
for i in {1..100}

do

cp emp.txt emp$i.txt

done
```

```
$ssh cp_emp.sh
```

(점심시간 문제)

문제 185. 지금 복사한 emp1.txt ~ emp100.txt의 이름을 employee1.txt ~ employee100.txt로 변경하시오.

(이름 변경하는 명령어는 mv이다)



mv 변경하려는파일.확장자(공백)변경될파일.확장자

답

```
for i in {1..100}
do
mv emp$i.txt employee$i.txt
done
```

결과

```
$ vi mv_emp.sh
```

.

문제 186. 옆에 짝궁이 없으니까 스스로 employee1.txt ~ employee100.txt 중 하나를 랜덤으로 골라서 파일을 연 후에
숫자 3000을 3900으로 변경해보시오.

답

```
vi employee80.txt
```

```
:%s/3000/3900/g 만 입력
```

문제 187. emp.txt와 employee80.txt 의 파일의 차이가 있는지 확인하시오
diff 명령어를 활용하면 된다!

답

```
$ diff emp.txt employee80.txt
```

결과

```
10c10
< 7902      FORD      ANALYST      7566      1981-12-11      3000      0      20
-  --
> 7902      FORD      ANALYST      7566      1981-12-11      3900      0      20

12c12
< 7788      SCOTT      ANALYST      7566      1982-12-22      3000      0      20
-  --
> 7788      SCOTT      ANALYST      7566      1982-12-22      3900      0      20
```

★ diff --brief 로 두 파일간의 차이가 있는지 유무만 확인하고 싶을때?

`$ diff --brief emp.txt employee80.txt` ← `--brief` 를 붙여주면 된다.

```
Files emp.txt and employee80.txt differ
```

 <- 두 파일간 차이가 있으면 이런 결과↓가 나온다.

```
$ diff --brief emp.txt employee81.txt
```

 <- 두 파일간에 차이가 없으면 아무런 결과도 나오지 않는다.

문제 188. for loop 문을 이용해서 employee1.txt ~ employee100.txt 중에 emp.txt와 차이가 있는 파일이 무엇인지 한번에 알아내시오

답

```
for i in {1..100}
do
diff --brief emp.txt employee$i.txt
done
```

결과

```
$ sh check_diff.sh
```

```
Files emp.txt and employee80.txt differ
```

문제 189. employee 로 시작하는 text파일을 모두 지우시오.

답

```
$ rm employee*.txt
```

결과

```
$ ls -l employee*.txt

ls: cannot access employee*.txt: No such file or directory
```

문제 190. `ls -l find_sal.sh` 를 했을때 출력되는 결과에서 크기에 해당하는 부분만 출력하시오.

답

```
$ ls -l find_sal.sh
```


 해서 크기의 위치가 몇번째 \$인지 확인

```
$ ls -l find_sal.sh | awk '{print $5}'
```

결과

```
271
```

문제 191. size100 이라는 폴더를 만드시오.

답  (항상 자신의 위치가 어디인지 확인하는 버릇을 들이는 것이 좋다.)

```
$ cd

$ pwd

/home/scott

$ mkdir size100
```

결과 

```
$ ls -ld size100
```

```
drwxrwxr-x. 2 scott scott 6 Jan  6 00:11 size100
```

문제 192. 확장자가 .sh 인 shell script중에서 사이즈가 100 byte 이상인 사이즈만 출력하시오.

```
vi mv_size_100.sh
```

```
list=`ls -l *.sh` | awk '{print $9}' # sh로 된 파일명을 가져오는 코드(9번째가 파일명) // list에 담은 값을 아래에 사용하기 위해 명령문을 ``로 감싸주

for i in $list
do
echo $i # 전체 파일명 리스트에서 각각의 파일명을 출력
done

list=`ls -l *.sh` | awk '{print $5}' # 파일 사이즈를 가져오는 코드 // 출력값중 사이즈는 5번째

for i in $list
do
echo $i
done
```

답 

```
size2=`ls -l *.sh | awk '{print $5}'` # .sh 파일중에서 파일의 사이즈 부분만 뽑아낸다

for i in $size2 # 사이즈들을 하나씩 루프문으로 가져온다.
do
if [ ${i:0:3} -gt 100 ]; then # i의 사이즈부분은 앞의 3글자이기때문에 substr하듯이 잘라준 것. + ``-gt``로 숫자열 비교를 해준 것
echo $i # ${i:0:3} 문자를 0번째 ~ 3번째 까지 잘라서 뽑아라
fi
done
```



!? 기본적으로 비교명령문은 문자열과 숫자열의 기호가 각각 다르지만, 명령문에 값이 입력될때 리눅스가 자동으로 문맥에 맞게 숫자형 or 문자형으로 변경이 된다고 한다.
따라서 오류가 날때는 비교명령문 유형을 (숫자to 문자 vice versa) 로 해보면 된다. 그도 아니면 위의 것 처럼 원하는 부분만을 잘라서 입력시켜도 된다.

문제 193. 확장자가 .sh 인 shell script 중에서 size가 100byte 이상인 shell script는 size100 폴더에 이동시키시오.

답

```
list=`ls -l *.sh | awk '{print$9}'` # 파일 리스트 가져오는 코드
```

```
for i in $list      # for 문으로 파일을 하나씩 불러온다.
do
size=`ls -l $i | awk '{print$5}'` # 해당 파일의 사이즈를 뽑아서
if [ $size -ge 100 ]; then      # 사이즈가 100 이상이면
mv $i size100                  # 해당 파일을 size100 폴더로 이동한다.
fi
done                            # 루프문을 종료한다
```

리눅스 쉘을 현업에서 어떻게 활용하는가?



라이나 생명에서 데이터 분석을 했던 사례

1. 고객데이터를 리눅스 쉘을 이용해서 데이터를 필터링 한다. by Shell (python도 가능하지만 shell이 훨씬 빠름)

e.g) 라이나 생명 고객중에 40대만 따로 분리해서 text 파일을 생성한다.

2. 40대의 데이터를 가지고 군집분석(k-mean) 을 한다. by R / Python

3. 40대의 데이터를 가지고 연관분석(아프리오 알고리즘) by R / Python

새로운 보험 상품이 출시가 되었을때 보험 가입을 유도해야 하는데 모든 보험 가입자들에게 연락해서

가입을 유도하는 것보다 보험 가입이 가능할 것 같은 고객들만 추려서 연락을 하는게 훨씬 효과적이기 때문.

문제 194. emp.txt에서 직업이 SALESMAN 인 직원들의 데이터만 가지고 salesman.txt 파일을 생성하시오.

\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

답

```
$ awk '$3==toupper("salesman") {print $0}' emp.txt >> salesman.txt
```

```
$ cat salesman.txt
```

결과

7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30

7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30

*설명 : 라이나 생명에서는 10대.txt , 20대.txt, 30대.txt 이런식으로 만들었듯이 우리도 emp.txt를 가지고 salesman.txt , analyst.txt ,clerk.txt, manager.txt를 자동으로 다 만들게 하고 싶은것

물론 194번 처럼 하나씩 넣어도 되지만.. 번거로우니까

문제 195. emp.txt에서 직업을 출력하는데 중복을 제거해서 출력하시오.

\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

" uniq 함수를 써서 중복되는 값을 제거해준다! "

답

```
$ awk '{print $3}' emp.txt | sort | uniq
```

결과

```
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN
```

문제 196. 위의 직업을 하나씩 불러와서 for loop 문에서 출력하는 코드를 작성하시오

\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

답

```
vi for_job.sh
```

```
job=`awk '{print $3}' emp.txt | sort | uniq`
for i in $job
do
echo $i
done
```

결과

```
$ sh for_job.sh
```

```
ANALYST
```

CLERK
MANAGER
PRESIDENT
SALESMAN

문제 197. 위의 스크립트를 이용해서 해당 직업의 직원들의 데이터만 직업명을 이름으로 해서 아래와 같이 생성되게 하시오.

\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

답

```
$vi generate_job.sh
```

```
job=`awk '{print $3}' emp.txt | sort | uniq` # $3=직업부분의 데이터만 emp에서 뽑아와서 정렬을하고, 중복되는 값들을 제거해서 job에 담는다
for i in $job
do
grep -i $i emp.txt >> $i.txt # $i = 직업명 이다. emp.txt에서 직업명에 해당하는 부분만 뽑아내서, $i=직업명.txt로 파일 이름을 바꿔서 저장하라
cat $i.txt # 그 결과를 출력하라
done
```

결과

```
$ssh generate_job.sh
```

문제 198. 위의 스크립트를 이용해서 부서번호별로 emp.txt 가 쪼개지게끔 스크립트를 생성하시오.

\$1=사원번호 \$2=이름 \$3=직업 \$4=mgr \$5=입사일 \$6=월급 \$7=comm \$8=부서번호

답

```
$vi make_deptno.sh
```

```
deptno=`awk '{print $8}' emp.txt | sort | uniq`
for i in $deptno
do
awk -v num=$i '$8==num {print $0}' emp.txt >> deptno${i:0:2}.txt
cat deptno$i.txt
done
```

결과

```
$ls -lrt
```

```
.
.
.
.
-rw-rw-r--. 1 scott scott 219 Jan 6 02:06 deptno10.txt
-rw-rw-r--. 1 scott scott 364 Jan 6 02:06 deptno20.txt
-rw-rw-r--. 1 scott scott 453 Jan 6 02:06 deptno30.txt
```

[21.01.07 리눅스 강의 Day 8 // 하둡]

- 하둡(hadoop) ?

대용량 데이터를 분산 처리할 수 있는 자바 기반의 오픈소스 프레임 네트워크 (빅분기도 나눔)

- 빅데이터 분석기사에 나온 하둡의 정의

대용량 데이터를 분산 처리할 수 있는 '자바 기반'의 오픈소스 프레임 네트워크로서, 하둡은 '분산 파일 시스템인 HDFS(Hadoop Distrubuted File System)' 에 데이터를

저장하고 분산처리시스템인 맵리듀스를 이용해 데이터를 처리한다.

- 현업에서 데이터 분석가들은 어떻게 하둡을 활용하는가?

데이터의 종류!

1. 정형화된 데이터 : emp의 컬럼과 갯로 이루어진 rdmb에 저장되는 테이블 형태의 데이터
2. 반 정형화된 데이터 : 웹로그와 sns 데이터 , html, json
3. 비정형화된 데이터 : 동영상이나 이미지 , 텍스트 데이터

▼ RDMBS vs 비 RDMBS

RDBMS vs

Oracle

mssql

mysql

maria db(무료)

PostgreSQL

비 RDMBS(대표적으로 Hadoop)

하둡 (무료)

스칼라

오라클은 프로그램 사용에 대한 관리비(a/s 포함) 를 매달 결제해야함 // 가장 높은 골드 등급의 관리비는 매달 1억정도 때문에 많은 기업들이 maria db 나 하둡 등(무료프로그램) 을 사용하는 경우가 많음.

- 실무에서의 데이터 분석 예시

라이나 생명 데이터 분석가

1. 하둡에 고객 데이터 (정형화된 데이터) 를 저장하고 나이대별로 분리해서 csv파일 생성
2. 나이대별로 분리한 csv파일 데이터를 가지고 군집분석
3. 나이대별로 분리한 csv파일 데이터를 가지고 연관분석
4. 보험 상품을 잘 구매할(혹은 더 적합한) 고객에게 우선적으로 제품 안내 홍보에 활용

- 아프리카 tv 데이터 분석가

1. 채팅데이터를 전부 하둡에 저장 (오라클에 저장할 경우 방법도 어렵고, 돈도 엄청 많이 든다. 하둡은 공짜!)
2. 특정 단어를 자주 사용하는 채팅방 형태를 분석한다.
3. 머신러닝을 사용해서 가공한 채팅 데이터를 학습시키고 특정 단어를 사용하는 사람은 불건전 대화를 이끌 가능성이 높다는 것을 미리 예측

- 하둡이 나온 배경

구글에서 구글에 쌓여지는 수많은 빅데이터들을 처음에는 RDBMS(오라클) 에 입력하고 데이터를 저장하려고 시작했으나 데이터의 양이 너무 많다보니 실패를 하고 자체적으로 빅데이터를 저장할 기술을 개발했다. 그리고 대외적으로 이 기술에 대한 논문을 하나 발표했다. (오픈소스는 열심히 개발한 코드자체를 모두 공개하는 것이니까 하지 않음)

웹페이지의 글들을 오라클에 입력하려면 테이블에 입력할 수 있는 insert 문장으로 만들어 준 뒤에 입력을 해야 한다. 하지만 그렇게 만들수가 없기 때문에 다른 기술을 개발한 것.

구글에서 만든 그 논문을 Doug Cutting 이라는 사람이 읽고 Java로 구현을 했다.(당시에 파이썬은 유명하지 않았음. 아마 파이썬이 유명했다면 파이썬으로 했을가능성이 클듯)

하둡은 자바로 만들어 졌다. 자바를 몰라도 하둡의 데이터를 쉽게 다룰 수 있도록 NoSQL을 제공해서 SQL과 같은 언어로 쉽게 빅데이터를 다룰 수 있게 되었다.

이것을 Hive라고 칭한다. (벌떼 라는 뜻)

- 빅데이터 분석기사 내용

NoSQL (Not Only SQL)

추가메모 내용 있음 : 노트 필르기 보고 추가하기

NoSQL은 전통적인 RDBMS와 다른 DBMS를 지칭하기 위한 용어.

e.g)

1. Hive → SQL과 똑같다.
2. Pig → SQL과 다름
3. Mongo db → SQL과 다름

현업에 가면 2가지 종류의 서버를 다루는 회사가 있음

OTTP서버

DW(Data Warehouse)서버 (Oracle, 하둡, mySQL,Maria db, 스칼라 등 사용)

부서 :

DW부서, DATA MART



DBA,

DW 개발자,

데이터 분석가



database 관리자

SQL과 PLSQL을 사용해서 데이터에서 정보를 얻어 내는 역할

더그 커팅이라는 사람이 하둡을 구현한뒤에 이름을 무엇으로 지을까 고민을 하던 와중, 그의 아기가 노란 코끼리 장난감을 가지고 놀면서 hadoop? 이라고 말했다고 한다.

그래서 이 장면을 보고 하둡이라 이름지었다고 한다..ㅎ

그래서 그 뒤로 Hadoop 을 편하게 이용할 수 있도록 개발한 모든 하둡 생태계의 개발 프로그램 이름들이 전부 동물이름으로 지어지게 되었다.

Hadoop(일종의 OS) —————→ Hive(벌떼)

—————→ Pig(돼지)

—————→ Mongo db

- 하둡의 장점?

★ "공짜이다"

"분산 처리가 가능하다"

분산처리?

여러대의 노드(컴퓨터)를 묶어서 마치 하나의 서버처럼 보이게하고 여러 노드의 자원을 이용해서 데이터를 처리하기 때문에 처리하는 속도가 아주 빠르다는 장점이 있다.

e.g)

우리반 컴퓨터 한대의 메모리가 8기가인데, 총 30대가 있으므로 이들을 모두 묶어서 분산처리 파일 시스템을 구성하게 되면 30대의 컴퓨터를 하나의 컴퓨터처럼 보이게 할 수 있다.

즉, 총 240 Giga 의 메모리를 사용할 수 있고, 30개의 cpu를 다 사용할 수 있으며, 하드디스크의 용량도 30대 각 용량의 총합과 같다)

이러한 것이 가능하도록 지원하는 것이 하둡이다.

e.g)

한대의 서버로 1TB의 데이터를 처리하는데 걸리는 시간이 2시간 반이라면, 하둡으로 여러대의 서버를 병렬로 묶어서 작업하면 2분내에 끝낼 수 있다.

실제로 2008년 뉴욕타임즈는 130년 분량의 신문기사를 1100만 페이지를 하둡을 이용해서 하루만에 pdf로 변환을 했다.

이 때 든 비용은 고작 200만원이었다.

전문가들의 말에 의하면 만약 이 작업을 하둡이 아닌 일반 서버로 처리했다면 4년이 걸렸을것으로 예상했다. 하둡의 가성비와 대단함을 잘 알 수 있는 예시.



● 하둡 에코 시스템 (하둡의 생태계)

빅데이터 분석	R, Python 등을 이용해서 분석
▲	
빅데이터 저장(DB)	Hbase, MongoDB, Cassandra, CouchDb
▲	
분산 처리 지원	Hive, Pig, Sqoop, Zookeeper
▲	
분산 배치 처리	하둡(hadoop) - MapReduce (made by Java code)
▲	
분산 파일 관리	하둡 (hadoop) - HDFS(Hadoop Distrubuted File System)



Hive 사용 예

hive> select count(*) from emp; —————→ Java 로 자동 변환해서 결과를 출력해준다.

14

야후의 경우 약5만대의 서버(컴퓨터)를 연결해서 하둡을 운영하고 있고 페이스북은 약 1만대 이상의 하둡 클러스터를 사용중이다.



하둡의 장점

무료이다.

분산 처리 시스템을 구현할 수 있다.



-하둡의 단점 :

1. 무료이다 보니 유지보수가 어렵다.

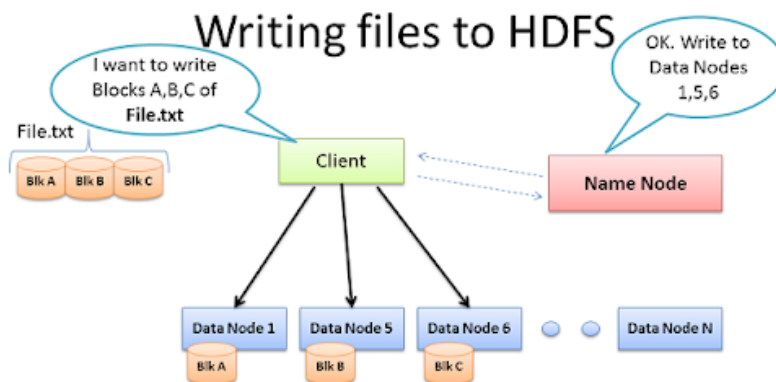
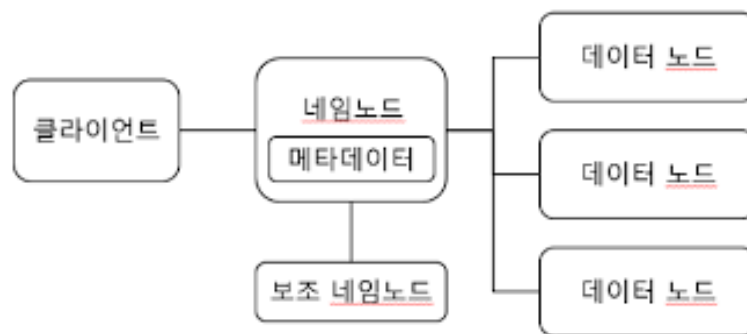
2. 네임노드가 다운되면고가용성이 지원이 안된다.

3. 한번 저장된 파일을 수정할 수가 없다.

기존 데이터의 내용은 append되는데 update가 안된다.

e.g) emp.csv 내용을 변경하려면 emp.csv를 직접 열어서 수정해야 한다.

이러한 이유로 주로 변동가능성이 거의 없는 과거의 데이터, 고객의 리뷰, 등을 저장하고 다룰때 사용한다.



- Client consults Name Node
- Client writes block directly to one Data Node
- Data Nodes replicates block
- Cycle repeats for next block

BRAD HEDLUND .com

네임노드 : 메타 데이터를 저장하고 있다. (메타데이터, 다른 데이터를 설명하는 데이터) - 일종의 index 같은 개념. 파일의 위치를 설명한다.

만약 네임노드가 아누되면 메타데이터를 불러올 수 없기 때문에 원하는 데이터를 불러올 수가 없다.



* 하둡의 주요 배포판

리눅스도 centos, redhat, ubuntu 등이 있는 것처럼 하둡도 배포판이 있다.

1. Cloudera 업체에서 나온 CDH
2. Hortonworks 에서 나온 HDP
3. 아마존에서 나온 EMR(Elastic Mapreduce)
4. Hstreaming 에서 나온 Hstreaming

하둡 홈페이지

<http://hadoop.apache.org>

하둡설치

하둡 설치를 위해서는 먼저 Virtual Box에 하둡을 설치할 리눅스 운영체제를 만들어줘야 한다.

이전 실습(리눅스 + 파이썬실습) 때 만들어둔 가상 리눅스가 이미 있으나 프로그램의 오류등으로 작업에 문제가 생길 여지가 있으므로 가급적이면 하둡 전용 운영체제를 새로 만든다.

1. 순정 리눅스 운영체제를 하나 만든다. (+ 게스트 확장자 설치까지 완료)
2. 순정 리눅스는 복제용으로 두고, 3번째 리눅스(for Hadoop)을 만든다.
centos 파일 우클릭 → 복제 선택 → 다음 → 완전한 복제 클릭 → 대략 2~3분 걸림



하둡 설치의 큰 그림 (환경 구성이 어렵지, 설치 자체는 쉽다!)

1. java 설치 → 하둡이 자바로 만들어져 있어서 자바를 설치해야 한다.
2. keygen 생성 → 여러개의 노드들을 묶어서 하나의 컴퓨터처럼 보이게 하는게 하둡의 목표이므로 다른 컴퓨터로 접속할때 패스워드를 물어보지 않아도 접속될 수 있게 해야 해서
keygen을 생성하는 것
3. 하둡 설치 → 하둡 설치 자체는 아주 간단! 4개의 파일의 내용만 수정하면 끝!



하둡 설치에 앞서 해야할 작업들

1. 현업에서 처럼 리눅스 서버에 putty 와 같은 접속 프로그램을 이용해서 접속하도록 설정
아래의 사이트 참고함

리눅스 가상머신에 PuTTY로 SSH 원격 접속하는 방법

기본적으로 SSH 원격 접속을 하려면 대상 리눅스 머신에서 sshd 데몬이 실행중이고 방화벽이 허용 상태여야 한다. 윈도우에서는 PuTTY를 이용해 접속할 수 있다. - 보호된 원격 로그인, 원격 데이터 통신에 사용되는 프로토콜- 패킷을 암호화하여 전송하기 때문에 도청, 위변조 차단이 가능- 서버/클라이언트 구조로 동작하며 보통 TCP 포트 22 사용

☞ <https://atoz-develop.tistory.com/entry/%EB%A6%AC%EB%88%85%EC%8A%A4-%EA%B0%80%EC%83%B1%EB%A8%B8%EC%8B%A0%EC%97%90-PuTTY%EB%A1%9C-SSH-%EC%9B%90%EA%B2%A9-%EC%A0%91%EC%86%8D%ED%95%98%EB%8A%94-%EB%B0%A9%EB%B2%95>

💡 Putty 설치법



Putty 설치를 위한 준비 사항

1. 리눅스 머신에서 **sshd** 데몬 실행 상태 확인
2. 리눅스 머신에서 ssh 포트 **방화벽** 허용 확인
3. 리눅스 머신에서 **호스트 allow** 설정
4. 가상머신 **포트 포워딩** 설정
5. 윈도우 PC에서 **PuTTY** 프로그램 준비

기본적으로 SSH 원격 접속을 하려면 대상 리눅스 머신에서 sshd 데몬이 실행중이고 방화벽이 허용 상태여야 한다. 윈도우에서는 PuTTY를 이용해 접속할 수 있다.

SSH(Secure Shell)



- 보호된 원격 로그인, 원격 데이터 통신에 사용되는 프로토콜
- 패킷을 암호화하여 전송하기 때문에 도청, 위변조 차단이 가능
- 서버/클라이언트 구조로 동작하며 보통 TCP 포트 22 사용

SSH를 사용하기 위해서는 SSH 지원 도구가 서버, 클라이언트에 모두 필요하다.

- **SSH 서버 프로그램 : sshd**
- **SSH 클라이언트 프로그램 : ssh, sftp, scp 등 사용 목적에 따라 다양**

여기서는 원격 접속을 할 것이므로 ssh 클라이언트 프로그램이 해당된다.

1. OpenSSH 패키지 설치



- SSH 프로토콜을 사용하는 공개 소스 프로그램 패키지
- 서버 프로그램인 sshd와 클라이언트 프로그램들을 모두 포함한다.

현재 대부분의 리눅스 배포판에는 OpenSSH가 포함되어 있어 보통 별도 설치가 필요하지 않다.

그래도 혹시 설치가 필요한지 확인하려면

OpenSSH 설치 확인 방법



```
[user1@localhost ~]$ yum list installed | grep openssh
openssh.i686      5.3p1-123.el6_9 @anaconda-CentOS-201806291517.i386/6.10
openssh-askpass.i686 5.3p1-123.el6_9 @anaconda-CentOS-201806291517.i386/6.10
openssh-clients.i686 5.3p1-123.el6_9 @anaconda-CentOS-201806291517.i386/6.10
openssh-server.i686 5.3p1-123.el6_9 @anaconda-CentOS-201806291517.i386/6.10
```

`yum list installed | grep openssh` 를 입력해 Open 설치 패키지가 설치되어있는지 확인. 위와 같은 결과가 출력된다면 설치가 되어 있는 것. 아니면 설치 해야 함.

아래의 명령어를 통해 패키지를 설치 or 업데이트 해줘야 한다.

```
yum -y install openssh      # OpenSSH 패키지 설치
yum -y update openssh      # OpenSSH 패키지 업데이트
```

2. SSH 서버 실행



가상 머신의 리눅스로 원격 접속하기 위해서는 SSH 서버 프로그램인 sshd 데몬이 실행중인 상태여야 한다. sshd 실행 상태는 `service sshd status` 명령으로 확인할 수 있다. OpenSSH 패키지가 이미 설치되어 있다면 기본적으로 부팅 시 자동으로 실행되도록 설정되어 있어 sshd 데몬도 실행 중일 가능성이 높다.

```
[user1@localhost ~]$ service sshd status
openssh-daemon (pid 2324) is running...
```

```
service sshd start      # sshd 실행
chkconfig sshd on       # 부팅 시 sshd 자동 실행
```

실행중이지 않으면 `service sshd start` 명령으로 실행한다. 부팅 시 sshd가 자동으로 실행되게 하려면 `chkconfig sshd on` 명령을 입력한다.

3. 방화벽 설정

1) tcp 목적지 포트 허용설정 확인

SSH가 사용하는 22번 포트를 허용하도록 방화벽 설정이 되어있어야 한다. `iptables -nL` 명령을 입력하여 다음의 ACCEPT...tcp dpt:22처럼 INPUT 체인에 tcp 목적지 포트 22가 허용 설정이 되어있는지 확인한다.

```
[user1@localhost ~]$ iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source                destination           state RELATED,ESTABLISHED
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0
ACCEPT     icmp --  0.0.0.0/0             0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0             0.0.0.0/0             state NEW tcp dpt:22 <-이부분!
REJECT     all  --  0.0.0.0/0             0.0.0.0/0             reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination           reject-with icmp-host-prohibited
REJECT     all  --  0.0.0.0/0             0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

만약 이렇게 설정이 되어 있지 않다면 `iptables -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT` 를 입력해 SSH 접속 허용 설정을 해주고, `service iptables save` 를 입력하여 저장한다.

```
[user1@localhost ~]$ iptables -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
[user1@localhost ~]$ service iptables save
```

!? 이때 아래와 같은 에러가 발생한다면?



The service command supports only basic LSB actions (start, stop, restart, try-restart, reload, force-reload, status). For other actions, please try to use systemctl.

다음의 사이트 참고함 (<https://www.programmersought.com/article/12562173941/>)

1. It is possible to go back to a more classic iptables setup. First, stop and mask the firewalld service:

```
systemctl stop firewalld
systemctl mask firewalld
```

2. Then, install the iptables-services package:

```
yum install iptables-services
```

3. Enable the service at boot-time:

```
systemctl enable iptables
```

4. Managing the service (choose one that you need)

```
systemctl stop iptables
systemctl start iptables
systemctl restart iptables
```

5. Saving your firewall rules can be done as follows:

```
service iptables save
or
/usr/libexec/iptables/iptables.init save
```

4. 호스트 allow 설정

- 1) 리눅스 머신의 TCP/IP 서비스를 이용할 수 있는 호스트를 생성한다.

▼ VirtualBox의 [파일] - [호스트 네트워크 관리자] 메뉴를 클릭한다.

5. Putty로 접속

PuTTY는 윈도우에서 사용할 수 있는 SSH 클라이언트 프로그램이다. 프로그램이 없을 경우 아래 URL에서 다운로드 받을 수 있다.

Download PuTTY: latest release (0.74)

This page contains download links for the latest released version of PuTTY. Currently this is 0.74, released on 2020-06-27. When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternatively, here is a permanent link to the 0.74 release.

 <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

1) 자신의 운영체제에 맞는 실행파일 다운로드

위의 사이트에 접속해 자신의 운영체제 및 버전에 맞는 실행파일을 다운 받는다.

2) 접속할 리눅스 머신의 IP주소를 확인

ifconfig 입력하여 IP 주소 확인

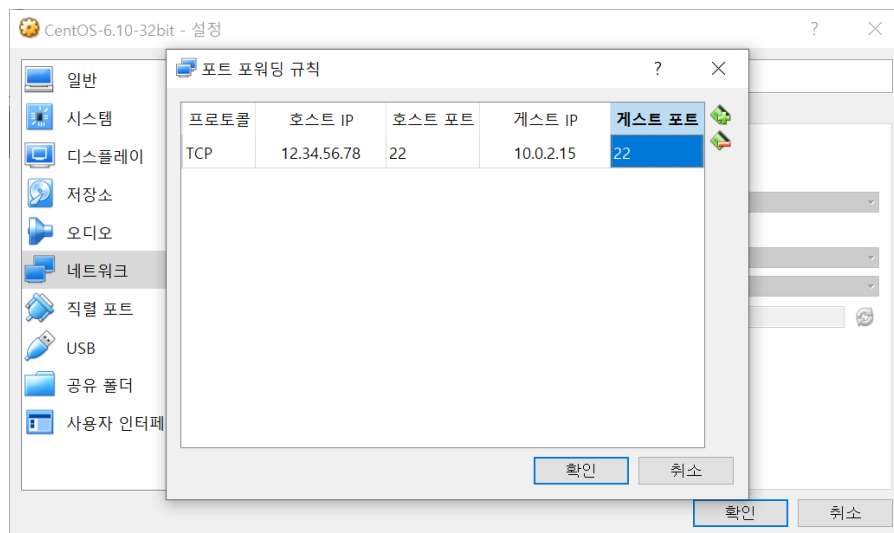
```
[user1@localhost ~]$ ifconfig
eth3      Link encap:Ethernet  HWaddr 08:00:27:F1:7E:0C
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
```

위와 같은 출력 결과에서 IP주소는 10.0.2.15이다.

3) Virtual Box의 가상 머신 인터넷 설정

Virtual Box를 실행 시킨 후

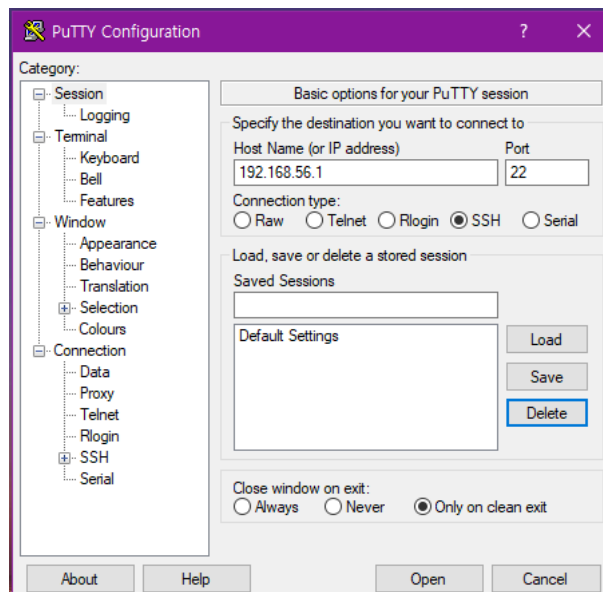
설정하려는 머신 선택후 [설정] → [네트워크] → [어댑터1] → [고급] → [포트포워딩] → [포트 포워딩 규칙] → 우측의 + 버튼 클릭 아래의 사진과 같이 입력



★프로토콜은 TCP, 호스트 IP는 VirtualBox Host-Only Ethernet Adapter IP(4-1) 에서 확인한 ip) , 게스트 IP는 ifconfig 명령으로 확인한 IP(5-2) 에서 확인) , 포트는 22를 입력하고 확인을 클릭한다.

4) Putty 로 접속

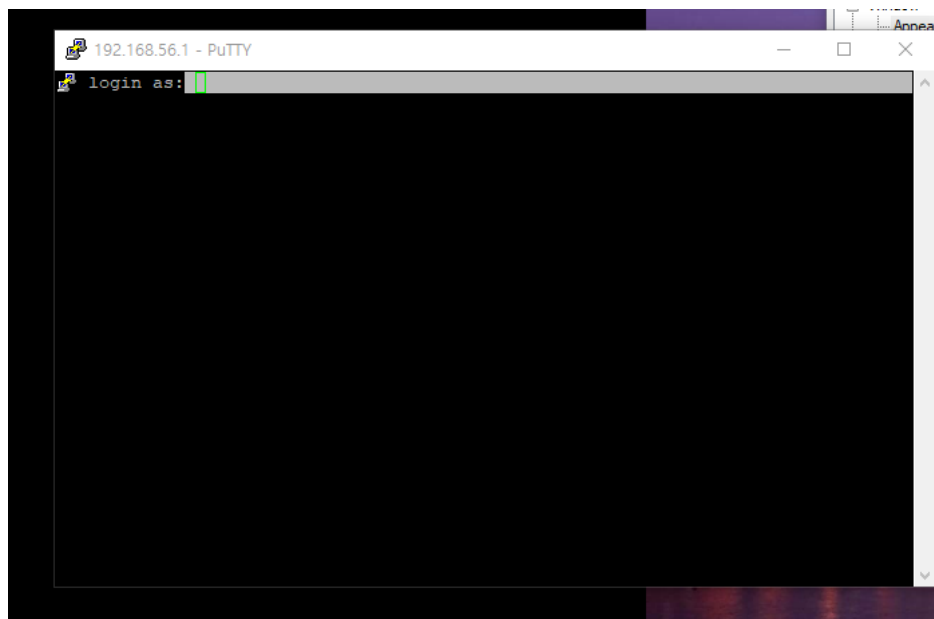
아래와 같이 호스트 IP 를 입력하고 Port 는 22로 설정한 후 open을 클릭하면 창이 열린다.



5) 리눅스 접속

푸티를 모두 설치한후 아래와 같은 창이 뜨면 계정명, 비밀번호를 순차적으로 입력하고 접속하면 된다.

! Virtual box의 centos가 켜진상태여야지 푸티로도 접속이 가능하다.



이런 화면이 뜨면 성공!

여기서 화면이나 크기가 불편하다면 Putty 창의 작업창 에서 우클릭 → change settings → appearance → Font settings → Changes.. 클릭 → 원하는 글자 서식 설정

현재 상태까지 진행이 되었다면, 이제 따로 센토스에 접속할 필요없이 Putty 창을 통해 윈도우에서 실행을 하면 된다.

복사 & 붙여넣기도 윈도우의 메모장 등 문서 프로그램에서 작성을 하고 바로 Putty 창에 붙여넣기를 해도 잘 수행이 된다.
현업에서는 리눅스 창을 보고 작업할때는 서버실에 들어가서 서버에 연결되어있는 모니터를 켜고 하는 경우 뿐! (서버실 컴퓨터는 24/7 on 상태여서 따로 켤필요가없음)
실제로는 개인 컴퓨터에서 원격으로 접속해서 Putty 창을 통해 작업을 하게 된다.

문제200. (점심시간 문제) putty 로 centos 리눅스 서버에 접속한 화면을 캡쳐해서 올립니다. root 에서 작업하세요 ~~

▼ 실제 회사환경

내자리 컴퓨터 (노트북) —————> 서버실(리눅스 컴퓨터)
Putty ←- 서버 아이피 주소 입력 하여 open
회사는 아이피주소, 유저이름, 패스워드만 사원에게 알려줄 것.

그외에 필요한 모든 것은 서버에 설치되어있음 .
일을 하기 위해 필요한 것은 putty 하나뿐!

▼ 학습환경

서버실이 따로 없기에 우리 컴퓨터 안에 가상의 서버를 만들고 그 서버에 Putty로 접속한 것

Putty로 문제 풀기!

문제201. /home/scott 디렉토리로 이동해서 test100 이라는 디렉토리를 생성하시오 !

```
$ cd /home/scott
$ pwd
$ mkdir test100
```

문제202. 리눅스 명령어로 1 부터 5까지의 숫자를 출력하는 쉘스크립트를 작성하고 아래와 같이 실행하시오 !

```
$ sh a.sh
1
2
3
4
5

for i in {1..5}
do
    echo $i
done
```

문제203. 공유폴더 접근하시오 ~

```
$ cd /media/sf_data
$ ls -l emp.txt
```

문제204. 공유폴더에 있는 emp.txt 를 /home/scott 밑으로 복사하시오 !

```
[scott@localhost sf_data]$ cp emp.txt /home/scott/
```

문제205. 다시 집(/home/scott)으로 와서 emp.txt 를 cat 으로 열어보시오!

```
$ cd
$ pwd
$ cat emp.txt
```

문제206. 공유폴더에 있는 dept.txt 도 /home/scott 밑으로 복사하고 cat 으로 열어보시오 !

```
$ cp /media/sf_data/dept.txt .
↑
설명: 점(.) 이 의미하는것은 현재 디렉토리 이다.
```

하둡 설치 과정

★ 설치시 주의사항

항상 접속할 때는 scott 으로 접속하고 root 에서 작업할 때는 scott 으로 접속한 상태에서 터미널 창에서 su - 로 스위치 유져하여 작업해야 한다.

1. 하둡을 설치하기 위해서 필요한 파일

1) 아래의 설치 파일을 공유폴더에 둔다.

1. jdk-7u60-linux-i586.gz
2. hadoop-1.2.1.tar.gz
3. protobuf-2.5.0.tar.gz

2) 위의 3개의 파일을 바탕화면에 home 폴더로

#cd /media/sf_data 로 일단 공유폴더로 이동. (리눅스의 일반 계정 :scott)

```
#cp jdk-7u60-linux-i586.gz /home/scott
#cp hadoop-1.2.1.tar.gz /home/scott
#cp protobuf-2.5.0.tar.gz /home/scott
```

2. 네트워크 설정을 변경

먼저 인터넷이 가능한 상태로 설정해야 한다.

리눅스의 시작 버튼을 누르고 system tools 에 settings 에 network 에서 첫번째 인터넷을 켜준다.

그리고 firefox 를 열어서 네이버에 접속해 인터넷이 잘 접속 되는지 확인.

1) network 설정

가상 리눅스의 기본 인터넷(enp0s3)과 호스트용 인터넷(enp0s8)이 있는지 확인한다.

```
[scott@localhost ~]$ su - # 로 root 계정으로 접속
```

```
root@localhost ~]# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::2be9:9537:c024:5ffc prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:3f:4b:cf txqueuelen 1000 (Ethernet)
    RX packets 123 bytes 14220 (13.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 126 bytes 15409 (15.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.1 netmask 255.255.255.0 broadcast 192.168.56.255
    inet6 fe80::8786:e1a5:a931:729a prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:cd:8a:06 txqueuelen 1000 (Ethernet)
    RX packets 11 bytes 660 (660.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 28 bytes 4070 (3.9 KiB)

이하 생략 ....
```

▼ enp0s8이 없을때?

1. virtual box에서 해당 os의 [설정] → [네트워크 설정] → [어댑터 2] 활성화 → [호스트전용 어댑터] 로 설정 이 되어 있는지 확인.
2. 그래도 만약 enp0s8이 없다면 생성하고 스크립트 입력을 해줘야 한다.

```
[root@localhost ~]# cd/etc/sysconfig/network-scripts
[root@localhost ~]# vi ifcfg-enp0s8
```

로 enp0s8을 새로 생성하여 만들고 아래의 스크립트를 입력

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp0s8
UUID=d0d40ac6-330a-4d08-8309-ad74f3ac80e6
DEVICE=enp0s8
#ONBOOT=no
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=none
NETMASK=255.255.255.0
IPADDR=192.168.56.101 <- 가상 머신 설정에 설정되어 있는 ip
```

2) 리눅스의 네트워크 설정 부분의 스크립트들이 모여져 있는 곳으로 이동

```
[root@localhost ~]# cd /etc/sysconfig/network-scripts
```

```
[root@localhost ~]# ls
결과
ifcfg-enp0s3  ifdown-ppp      ifup-ib         ifup-Team
ifcfg-lo      ifdown-routes   ifup-ippv       ifup-TeamPort
ifdown        ifdown-sit      ifup-ipv6       ifup-tunnel
ifdown-bnep   ifdown-Team     ifup-isdn       ifup-wireless
ifdown-eth    ifdown-TeamPort ifup-plip       init.ipv6-global
ifdown-ib     ifdown-tunnel   ifup-plusb      network-functions
ifdown-ippv   ifup            ifup-post       network-functions-ipv6
ifdown-ipv6   ifup-aliases    ifup-ppp
ifdown-isdn   ifup-bnep       ifup-routes
ifdown-post   ifup-eth        ifup-sit

enp0s3 의 ip 가 10.0.2.15로 되어 있어야 정상!
```

ifcfg-enp0s3 수정

```
[root@localhost network-scripts]# vi ifcfg-enp0s3
ONBOOT=yes
```

ifcfg-enp0s8 수정

```
[root@localhost network-scripts]# vi ifcfg-enp0s8
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=none
NETMASK=255.255.255.0
IPADDR=192.168.56.101
```

설명 : 192.168.56.101 는 다른 컴퓨터에서 내 리눅스 컴퓨터로 접속할때 사용하는 ip주소이다.

3. hostname(=컴퓨터이름) 을 변경합니다.

위에서 설정한 아이피 주소와 함께 hostname를 centos로 하겠다고 지정해줘야 한다.

1) hostname 설정

vi 편집기로 hosts 파일에 들어가 ip주소와 바꿀 이름을 입력

```
[root@localhost network-scripts]# vi /etc/hosts
```

192.168.56.101 centos (ip주소는 virtual box에서 확인 되는 고유의 ip)

2) hostname 변경(세션을 다시 시작하면 변경 정보 인식)

현재 상태의 호스트 이름이 무엇인지 확인

```
[root@localhost network-scripts]# hostname
localhost.localdomain
```

호스트 이름을 변경하도록 입력

```
[root@localhost network-scripts]# hostnamectl set-hostname centos
```

제대로 바뀌었는지 확인

```
[root@localhost network-scripts]# hostname
centos
라고 나오면 성공적으로 바뀐 것.
```

네트워크 서비스를 재시작

설명 : 이전단계에서 네트워크 구성을 변경했으므로 변경된 내용을 반영하려면 다시 네트워크를 재시작 해야 한다.

```
[root@localhost network-scripts]# service network restart
```

Restarting network (via systemctl): [OK] 라고 뜨면 잘 된 것

만약 안되면 재부팅을 하면 만사 OK!

```
[root@localhost network-scripts]# ifconfig
```

4. 방화벽 해지

```
[root@centos Desktop]# iptables -F
[root@centos Desktop]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
Chain FORWARD (policy ACCEPT)
target     prot opt source                destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

5. 시스템을 rebooting 합니다.

```
[root@localhost ~]# reboot
```

★항상 접속할 때는 scott 으로 접속하고 root 에서 작업할 때는
scott 으로 접속한 상태에서 터미널 창에서 su - 로 스위치 유지하여 작업해야 함을 잊지 말것!

6. ld-linux.so.2 를 설치

설명 : ld-linux.so.2 를 설치해야 하는 이유는 java를 설치할때 필요한 파일이기 때문이다.
ld-linux.so.2 설치파일은 따로 필요없고 이미 리눅스 서버에 내장되어 있기 때문에 불러와서 설치만 하면 된다.

```
[scott@centos ~]$ su -
[root@centos ~]# yum install ld-linux.so.2
```

```
[root@centos ~]# reboot
```

7. 자바(Java) 설치

1) 설치전 현재 자바의 버전 확인하기

```
[scott@centos ~]$ java -version
openjdk version "1.8.0_262"
OpenJDK Runtime Environment (build 1.8.0_262-b10)
OpenJDK 64-Bit Server VM (build 25.262-b10, mixed mode)
[scott@centos ~]$ su -
```

version "1.8.0_262" 라고 나오면 자바 설치가 필요한 것.

2) 자바를 설치할 디렉토리 만들기

```
[scott@centos ~]$ su -
[root@centos ~]# mkdir -p /u01/app/java (-p 옵션 : 경로에 적힌 모든 파일을 한번에 만드는 것)
```

3) 위에서 만든 디렉토리에 jdk 파일을 mv로 이동 후 압축 풀기 (설치 끝!)

```
[root@centos ~]# mv /home/scott/jdk-7u60-linux-i586.gz /u01/app/java/
[root@centos ~]# cd /u01/app/java/
[root@centos java]# tar xvfz jdk-7u60-linux-i586.gz
...
jdk1.7.0_60/bin/jrunscript
jdk1.7.0_60/release
jdk1.7.0_60/THIRDPARTYLICENSEREADME-JAVAFX.txt
jdk1.7.0_60/COPYRIGHT
```

설치 끝!

4) 추가 작업

설명 : 압축을 풀면 jdk1.7.0_60 이라는 디렉토리가 생기는데 이 디렉토리의 소유자를 root유저로 변경해 줘야 한다.

```
[root@centos java]# chown -R root:root jdk1.7.0_60
```

잘 되었다면

ls -l 했을때

```
drwxr-xr-x. 8 root root      233 May  7 2014 jdk1.7.0_60
-rwxr-x---. 1 root root 143607445 Jan 7 23:13 jdk-7u60-linux-i586.gz
```

결과과 위와 같다면 권한 변경이 잘 이뤄진 것. 다시 scott(일반유저) 로 돌아가기
[root@centos java]# exit

5) 추가작업 2 (.bashrc 파일 수정하기)

설명 : .bashrc 파일은 리눅스 환경설정 파일이다. 이 파일에 적어주 내용대로 환경 설정이 이뤄지는데 scott으로 로그인할 때마다 .bashrc 파일이 자동으로 수행된다.

아래의 작업은 .bashrc 파일에 자바가 이 시스템에서 어디에 설치되어있다 라는 위치를 알려주는 작업이다.

그래서 vi 편집기로 .bashrc 파일을 열고 아래의 3줄을 맨 아래쪽에 입력해 줘야 한다.

```
[scott@centos ~]$ vi .bashrc
```

로 .bashrc 파일에 들어간 후 맨 밑에 아래의 코드 입력

```
export JAVA_HOME=/u01/app/java/jdk1.7.0_60
export PATH=/u01/app/java/jdk1.7.0_60/bin:$PATH
export CLASSPATH=.:usr/java/jdk1.7.0_60/lib:$CLASSPATH
```



Source global definitions

```
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
```

Uncomment the following line if you don't like systemctl's auto-paging feature:
export SYSTEMD_PAGER=

User specific aliases and functions

```
export JAVA_HOME=/u01/app/java/jdk1.7.0_60
export PATH=/u01/app/java/jdk1.7.0_60/bin:$PATH
export CLASSPATH=.:usr/java/jdk1.7.0_60/lib:$CLASSPATH
```

위와 같이 입력이 되면 잘 된 것!

수정한 .bashrc 파일 내용 리눅스 서버에 반영시키기

설명 : 위에서 복사해서 붙여넣은 뒤 설정한 내용을 리눅스 서버에 반영을 해야 한다. 카카오톡 환경설정에서도 설정을 변경했으면 반영을 할때 apply 같은 버튼을 눌러주듯이 리눅스에도 반영을 하려면 .bashrc 파일을 실행하면 되는데, 실행하는 명령어가 아래와 같이 콤마(.) 를 쓰고 한칸 띄고 .bashrc 를 하면 된다.

```
[scott@centos ~]$ . .bashrc
```

6) 자바 가 잘 설치되었는지 확인

```
[scott@centos ~]$ java -version
java version "1.7.0_60" 라고 나와야 잘 설치가 된 것.
Java(TM) SE Runtime Environment (build 1.7.0_60-b19)
Java HotSpot(TM) Server VM (build 24.60-b09, mixed mode)
```

8.protobuf 를 설치

설명 : protobuf를 설치해야 하는 이유는 하둡을 사용하려면 여러대의 컴퓨터가 서로 접속을 자유롭게 하면서 패스워드를 물어보지 않고 그냥 자유롭게 서로 접속을 할수있어야 하기 때문이다.

1) 다시 root 로 접속해 protobuf 설치파일을 복사하고 파일의 상태 확인

```
[scott@centos ~]$ su -
Password:
[root@centos ~]# cp -v /home/scott/protobuf-2.5.0.tar.gz /usr/local
[root@centos ~]# cd /usr/local
[root@centos local]# ll
```

```
drwxr-xr-x. 2 root root    6 Apr 11 2018 bin
drwxr-xr-x. 2 root root    6 Apr 11 2018 etc
drwxr-xr-x. 2 root root    6 Apr 11 2018 games
drwxr-xr-x. 2 root root    6 Apr 11 2018 include
drwxr-xr-x. 2 root root    6 Apr 11 2018 lib
drwxr-xr-x. 2 root root    6 Apr 11 2018 lib64
drwxr-xr-x. 2 root root    6 Apr 11 2018 libexec
-rwxr-x---. 1 root root 2401901 Jan  7 02:40 protobuf-2.5.0.tar.gz ← 복사해온 파일이 잘 복사되었는지 확인
drwxr-xr-x. 2 root root    6 Apr 11 2018 sbin
drwxr-xr-x. 5 root root   49 Jan  6 02:32 share
drwxr-xr-x. 2 root root    6 Apr 11 2018 src
```

2) 복사한 파일 설치하기

```
[root@centos local]# tar xvfz protobuf-2.5.0.tar.gz
[root@centos local]# cd protobuf-2.5.0
[root@centos protobuf-2.5.0]# ./configure
```

까지 했는데

```
configure: error: in `/usr/local/protobuf-2.5.0':
configure: error: C++ preprocessor "/lib/cpp" fails sanity check
```

라는 에러가 뜬다면?

./configure가 알려진 많은 컴파일러 이름을 하나씩 확인하여 C ++ 컴파일러를 찾으려고 시도했음을 보여준것인데 c++ 를 찾지 못한 것

해결법

```
yum install gcc-c++
```

로 C++를 설치해줘야 함.

설치 가 끝나면

다시

```
[root@centos protobuf-2.5.0]# ./configure
```

해주고

```
[root@centos protobuf-2.5.0]# make
[root@centos protobuf-2.5.0]# make install

[root@centos protobuf-2.5.0]# protoc --version
libprotoc 2.5.0
```

라고 나오면 정상적으로 잘 된 것.

3) 다시 reboot !

```
[root@centos protobuf-2.5.0]# reboot
```

9. keygen 생성

하둡은 SSH 프로토콜을 이용해 하둡 클러스터간의 내부 통신을 수행한다.



컴퓨터가 네트워크로 대화를 나눌때 쓰는 언어

하둡을 다 설치하고 나서 하둡을 시작하는 명령어인 start-all.sh 쉘 스크립트를 수행하면 네임노드가 설치된 서버에서 데이터 노드가 설치된 서버로 접근해 데이터 노드와 테스크 트래커를 구동하게 된다. 그런데 이때 ssh 를 이용할 수 없다면 하둡을 실행할 수 없다.

네임노드에서 공개키를 설정하고 이 공개키(authorized_keys) 를 다른 데이터 노드에 다 복사해줘야한다.

e.g) 우리반 컴퓨터 30대를 모두 하둡으로 묶고 싶다면 30명 자리의 패스워드를 모두 공개키(authorized_keys)에 저장하고, 모든 자리에

공개키(authorized_keys)를 배포해 줘야 한다.

이 키를 가지고 있으면 ssh 로 다른 노드에 접속할때 패스워드 없이 접속할 수 있다.

1) 공개키 생성

설명 : 숨김 디렉토리인 .ssh 디렉토리를 생성할 것인데, 혹시 있을지도 모르니 삭제를 먼저 해준다.

```
[scott@centos ~]$ rm -rf .ssh
```

설명 : ssh-keygen 명령어를 이용해서 .ssh 디렉토리 생성을 한다.

```
[scott@centos ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/scott/.ssh/id_rsa): <- enter key
Created directory '/home/scott/.ssh'.
Enter passphrase (empty for no passphrase): <- enter key (그래야 비번 없어서 자동 접속)
Enter same passphrase again: <- enter key (그래야 비번 없어서 자동 접속)
Your identification has been saved in /home/scott/.ssh/id_rsa.
Your public key has been saved in /home/scott/.ssh/id_rsa.pub.
The key fingerprint is:
b8:e5:55:fd:f9:72:4d:48:e6:47:22:bd:44:cb:41:7d scott@centos
The key's randomart image is: < 일종의 지문 같은 것
```

그러므로 모양은 모두가 다른 것이 정상!

```
+--[ RSA 2048]-----+
|           .+. |
|           = oE|
|           o X o|
|           . * *.|
|           . S . +.+|
|           + . oo|
|           . . +|
|           o |
|           |
+-----+

```

설명 : 아래의 명령어로 scott으로 192.168.56.101(=내 서버) 서버에 접속을 시도한다.
즉 내 리눅스 서버로 접속하면서 패스워드를 물어볼텐데 그때 패스워드를 입력을 하면 패스워드가 authorized key에 담겨서 만들어 지게 된다.

```
[scott@centos ~]$ ssh-copy-id -i /home/scott/.ssh/id_rsa.pub scott@192.168.56.101

The authenticity of host '192.168.56.101 (192.168.56.101)' can't be established.
ECDSA key fingerprint is 6b:de:64:c5:27:c4:c6:d7:62:2a:9c:fa:83:d5:65:13.
Are you sure you want to continue connecting (yes/no)? yes <- 입력

/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
scott@192.168.56.101's password: 1234 <- 기본유저의 패스워드입력

```

아래와 같이 나온다면 잘 성공 된 것

```
Number of key(s) added: 1
Now try logging into the machine, with: "ssh 'scott@192.168.56.101'"
and check to make sure that only the key(s) you wanted were added.

```

2) 공개키 잘 생성되었는지 확인

```
[scott@centos ~]$ ssh scott@192.168.56.101
Last login: Thu Aug 6 22:17:11 2020

```

※ 중요: 여기서 접속시 패스워드를 물어보면 안된다.

[21.01.08 리눅스 강의 Day 9 //하둡]

■ 어제 배웠던 내용 복습

1. 하둡이란 무엇인가?

JAVA 로 만든 분산파일 시스템

하둡이 왜 필요한가? 빅데이터 저장을 위해 필요함.

▼ 빅데이터의 종류 3가지

정형화 데이터 : 컬럼과 로우로 이뤄진 테이블

반정형화 데이터 : 웹로그, html, json

비정형화 데이터 : 동영상, 이미지, 데이터, 텍스트, CSV

e.g 1) 데이터 분석가가 하둡을 사용해 하는 일의 과정

하둡, maria db → 무료 프로그램

네이버 영화 평점 댓글 수집 → 텍스트 → 하둡 → hive → 데이터 검색과 전처리 → 파이썬, R을 이용해서 분석, 시각화

e.g 2) 실제 학원 수료생의 실무 사례

지방흡입 기계(센서) 에서 발생하는 센서 데이터를 수집을 한 뒤 그 데이터를 분석해, 어떻게 지방흡입을 해야 명이 적게 드는지 그 방법을 연구하는 작업.

작업 방식의 위의 e.g 1) 의 과정과 비슷

2. 하둡 설치

- 리눅스 설치후 putty 로 접속할 수 있게 환경 구성
- 하둡 설치를 위해 필요한 3가지 파일 준비
 - 1) jdk 파일
 - 2) hadoop 설치파일
 - 3) Prototype 파일
- 자바 설치
- SSH로 리눅스 서버에 접속할 때 패스워드 물어보지 않게 설정

설치순서는 어제 필기에 이어서 번호 매김

10. hadoop 디렉토리를 만들고 거기에 하둡설치 파일 압축을 풀어준다.

```
[scott@centos ~]$ cd
[scott@centos ~]$ mkdir hadoop
[scott@centos ~]$ cd hadoop
```

hadoop 설치파일을 hadoop폴더로 복사한다.

아래의 명령어는 현재 디렉토리(/home/scott/hadoop) 바로전 상위 디렉토리(/home/scott/) 에 있는 hadoop~~ 파일을 현재 디렉토리에 복사하라는 뜻

만약 권한이 없어서 안된다거나 permission denied 라고 뜬다면 root로 가서 파일에 대한 권한 변경

▼ 권한변경방법

```
chown scott:scott hadoop-1.2.1.tar.gz
```

```
[scott@centos hadoop]$ cp ../hadoop-1.2.1.tar.gz .
[scott@centos hadoop]$ tar xvf hadoop-1.2.1.tar.gz
...
[scott@centos hadoop]$ rm hadoop-1.2.1.tar.gz
```

11. 하둡 홈 디렉토리를 설정한다.

설명 : 리눅스의 환경설정 파일인 .bashrc 파일을 vi편집기로 열어서 맨 아래에 HADOOP의 HOME(집) 디렉토리가 어디라고 지정해줘야 한다.

```
[scott@centos hadoop]$ cd
[scott@centos ~]$ vi .bashrc
```

```
export HADOOP_HOME=/home/scott/hadoop/hadoop-1.2.1
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

설명 : PATH는 리눅스에 설치되어져 있는 소프트웨어가 어디에 설치되어 있다고 등록하는 환경 변수

▼ 실행결과

```
scott@centos:~
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
export JAVA_HOME=/u01/app/java/jdk1.7.0_60
export PATH=/u01/app/java/jdk1.7.0_60/bin:$PATH
export CLASSPATH=.:usr/java/jdk1.7.0_60/lib:$CLASSPATH
export HADOOP_HOME=/home/scott/hadoop/hadoop-1.2.1
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
~
~
~
```

```
[scott@centos ~]$ . .bashrc
```

설명 : `. .bashrc` 명령어로 `. .bashrc` 스크립트를 실행한다.

12. 하둡 환경설정을 하기 위해 아래의 4개의 파일을 셋팅해야한다.



1. `[hadoop-env.sh]` (<http://hadoop-env.sh/>) : 자바 홈디렉토리와 hadoop 홈디렉토리가 어딘지 지정한다. 하둡이 자바로 만들어져 있기 때문에 자바가 어디에 설치되어져 있는지 하둡이 알고 있어야 하므로 자바 홈디렉토리를 `hadoop-env.sh` 파일에 지정한다.

2. `core-site.xml` : 하둡의 네임노드가 어느 서버인지를 지정한다.

하둡은 네임노드와 데이터 노드들로 구성되어져 있는데 네임노드에는 우리가 검색하고자 하는 데이터의 위치정보인 '메타 정보'가 들어있다

3. `mapred-site.xml` : java 로 만들어진 mapreduce 프레임워크와 관련된 정보를 지정하는 파일

하둡은 자바로 만들어진 큰 2개의 함수로 되어있다. mapping 함수와 reducing 함수 2개이다. 우리가 하둡에 있는 데이터를 검색하면 이 2개의 함수가 작동을 하면서 데이터를 검색하고 결과를 보여준다.

4. `hdfs-site.xml` : 하둡 파일 시스템인 HDFS(Hadoop DistributedFile System) 와 관련된 정보를 저장하는 파일. 여러대의 컴퓨터를 묶어서 하나의 강력한 슈퍼 컴퓨터로 사용하기 위해 반드시 필요한 하둡의 '필수 셋팅 부분'

1) hadoop-env.sh 수정

```
[scott@centos ~]$ cd $HADOOP_HOME/conf  
[scott@centos conf]$ vi hadoop-env.sh
```

설명 : 4개의 파일을 수정만 하면 되는데 첫번째로 자바의 홈디렉토리 와 하둡의 홈 디렉토리가 어디다 라고 지정하기 위해 hadoop-env.sh를 수정한다.

The java implementation to use. Required.

```
export JAVA_HOME=/usr/lib/j2sdk1.5-sun  
export JAVA_HOME=/u01/app/java/jdk1.7.0_60  
export HADOOP_HOME=/home/scott/hadoop/hadoop-1.2.1  
export HADOOP_HOME_WARN_SUPPRESS=1
```

▼ 결과

```

scott@centos:~/hadoop/hadoop-1.2.1/conf
# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME. All others are
# optional. When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use. Required.
# export JAVA_HOME=/usr/lib/j2sdk1.5-sun

# Extra Java CLASSPATH elements. Optional.
# export HADOOP_CLASSPATH=

# The maximum amount of heap to use, in MB. Default is 1000.
# export HADOOP_HEAPSIZE=2000

# Extra Java runtime options. Empty by default.
# export HADOOP_OPTS=-server

# Command specific options appended to HADOOP_OPTS when specified
export HADOOP_NAMENODE_OPTS="-Dcom.sun.management.jmxremote $HADOOP_NAMENODE_OPTS"
export HADOOP_SECONDARYNAMENODE_OPTS="-Dcom.sun.management.jmxremote $HADOOP_SECONDARYNAMENODE_OPTS"
export HADOOP_DATANODE_OPTS="-Dcom.sun.management.jmxremote $HADOOP_DATANODE_OPTS"
export HADOOP_BALANCER_OPTS="-Dcom.sun.management.jmxremote $HADOOP_BALANCER_OPTS"
export HADOOP_JOBTRACKER_OPTS="-Dcom.sun.management.jmxremote $HADOOP_JOBTRACKER_OPTS"
# export HADOOP_TASKTRACKER_OPTS=
# The following applies to multiple commands (fs, dfs, fsck, distcp etc)
# export HADOOP_CLIENT_OPTS

# Extra ssh options. Empty by default.
# export HADOOP_SSH_OPTS="-o ConnectTimeout=1 -o SendEnv=HADOOP_CONF_DIR"

# Where log files are stored. $HADOOP_HOME/logs by default.
# export HADOOP_LOG_DIR=${HADOOP_HOME}/logs

# File naming remote slave hosts. $HADOOP_HOME/conf/slaves by default.
# export HADOOP_SLAVES=${HADOOP_HOME}/conf/slaves

# host:path where hadoop code should be rsync'd from. Unset by default.
# export HADOOP_MASTER=master:/home/$USER/src/hadoop

# Seconds to sleep between slave commands. Unset by default. This
# can be useful in large clusters, where, e.g., slave rsyncs can
# otherwise arrive faster than the master can service them.
# export HADOOP_SLAVE_SLEEP=0.1

# The directory where pid files are stored. /tmp by default.
# NOTE: this should be set to a directory that can only be written to by
# the users that are going to run the hadoop daemons. Otherwise there is
# the potential for a symlink attack.
# export HADOOP_PID_DIR=/var/hadoop/pids

# A string representing this instance of hadoop. $USER by default.
# export HADOOP_IDENT_STRING=$USER

# The scheduling priority for daemon processes. See 'man nice'.
# export HADOOP_NICENESS=10
export JAVA_HOME=/usr/lib/j2sdk1.5-sun
export JAVA_HOME=/u01/app/java/jdk1.7.0_60
export HADOOP_HOME=/home/scott/hadoop/hadoop-1.2.1
export HADOOP_HOME_WARN_SUPPRESS=1
~
~
~
-- INSERT --

```

2) core-site.xml 파일 수정

설명 : 하둡을 구성하기 위한 두번째 파일은 core-site.xml 파일을 수정하는데 여기에는 네임노드와 데이터 노드에 대한 정보를 저장한다.

```
[scott@centos conf]$ vi core-site.xml
```

접속하면 아래와 같이 되어있을건데
<configuration>

// 스크립트 넣을 공간 //

</configuration>

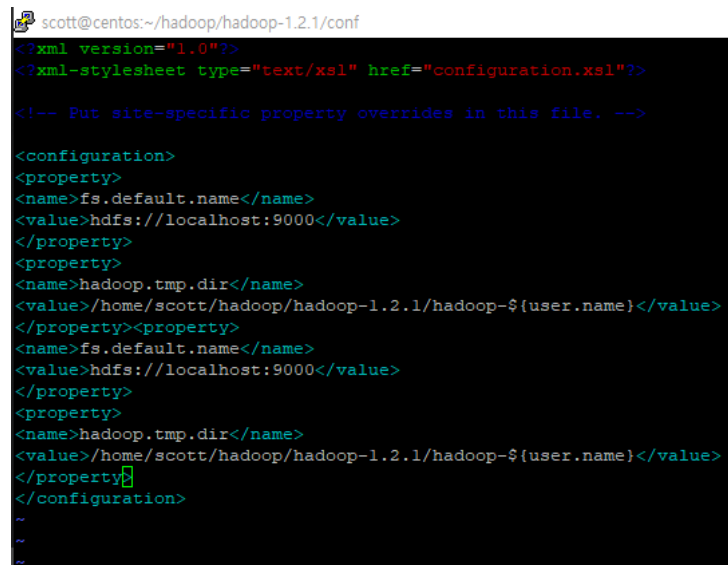
아래의 스크립트들을 복사해서 이 사이에 넣으면 된다.


```

<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/scott/hadoop/hadoop-1.2.1/hadoop-${user.name}</value>
</property>
</configuration>

```

▼ 결과



```

scott@centos:~/hadoop/hadoop-1.2.1/conf
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl">

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/scott/hadoop/hadoop-1.2.1/hadoop-${user.name}</value>
</property><property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/scott/hadoop/hadoop-1.2.1/hadoop-${user.name}</value>
</property>
</configuration>
~
~
~

```

3) mapred-site.xml 파일 수정

설명 :

위와 마찬가지로 <configuration> 사이에 스크립트들을 붙여넣고 나옴

```
[scott@centos conf]$ vi mapred-site.xml
```

```

<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:9001</value>
</property>
<property>
<name>mapred.local.dir</name>
<value>${hadoop.tmp.dir}/mapred/local</value>
</property>
<property>
<name>mapred.system.dir</name>
<value>${hadoop.tmp.dir}/mapred/system</value>

```

```
</property>
</configuration>
```

▼ 결과

```
scott@centos:~/hadoop/hadoop-1.2.1/conf
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:9001</value>
</property>
<property>
<name>mapred.local.dir</name>
<value>${hadoop.tmp.dir}/mapred/local</value>
</property>
<property>
<name>mapred.system.dir</name>
<value>${hadoop.tmp.dir}/mapred/system</value>
</property><property>
<name>mapred.job.tracker</name>
<value>localhost:9001</value>
</property>
<property>
<name>mapred.local.dir</name>
<value>${hadoop.tmp.dir}/mapred/local</value>
</property>
<property>
<name>mapred.system.dir</name>
<value>${hadoop.tmp.dir}/mapred/system</value>
</property>
</configuration>
~
~
~
~
```

4) 네임노드와 데이터 노드의 위치 로컬에 생성

설명 : 하둡은 크게 네임노드와 데이터 노드들로 구성되어 있는데 원래는 네임노드와 데이터 노드가 서로 다른 컴퓨터에 분리되어 있어야 하는데 우리는 컴퓨터가 하나이므로 네임노드와 데이터 노드를 같은 컴퓨터로 구성할 것이다.
그래서 아래와 같이 네임노드의 데이터가 축적될 위치와 데이터 노드의 데이터가 축적될 위치를 로컬에 생성한다.

```
[scott@centos conf]$ mkdir /home/scott/hadoop/hadoop-1.2.1/dfs
[scott@centos conf]$ mkdir /home/scott/hadoop/hadoop-1.2.1/dfs/name
[scott@centos conf]$ mkdir /home/scott/hadoop/hadoop-1.2.1/dfs/data
[scott@centos conf]$ vi hdfs-site.xml
```

```
<configuration>
<property>
<name>dfs.name.dir</name>
<value>/home/scott/hadoop/hadoop-1.2.1/dfs/name</value>
</property>
<property>
<name>dfs.name.edits.dir</name>
<value>${dfs.name.dir}</value>
</property>
<property>
<name>dfs.data.dir</name>
```

```
<value>/home/scott/hadoop/hadoop-1.2.1/dfs/data</value>
</property>
</configuration>
```

▼ 결과

```
scott@centos:~/hadoop/hadoop-1.2.1/conf
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.name.dir</name>
    <value>/home/scott/hadoop/hadoop-1.2.1/dfs/name</value>
  </property>
  <property>
    <name>dfs.name.edits.dir</name>
    <value>${dfs.name.dir}</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>/home/scott/hadoop/hadoop-1.2.1/dfs/data</value>
  </property>
</configuration>
~
~
~
```

12. data 와 name 디렉토리의 권한을 조정한다. chmod 755 로 바꿔주면 된다.

```
[scott@centos dfs]$ cd /home/scott/hadoop/hadoop-1.2.1/dfs
[scott@centos dfs]$ chmod 755 data
[scott@centos dfs]$ chmod 755 name
[scott@centos dfs]$ ls -l
total 0
drwxr-xr-x. 2 scott scott 6 Jan  6 09:01 data # 처음에 확인해보면 이부분의 권한이 생략과 다를 수 있는데 그걸 같게 해주기 위해 위에 처럼 조정하는 것
drwxr-xr-x. 2 scott scott 6 Jan  6 09:07 name
```

13. 하둡 네임노드를 포맷한다.

```
[scott@centos conf]$ cd
[scott@centos ~]$ hadoop namenode -format
```

```
14/08/06 00:28:19 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = edydr1p2.us.oracle.com/192.168.100.102
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 1.2.1
STARTUP_MSG: build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2 -r 1503152; compiled
by 'mattf' on Mon Jul 22 15:23:09 PDT 2013
STARTUP_MSG: java = 1.7.0_60
/
Re-format filesystem in /u01/app/hadoop/hadoop-1.2.1/dfs/name ? (Y or N) Y (대문자) 대문자 아니면 에러남
...
14/08/06 00:28:28 INFO namenode.FSEditLog: closing edit log: position=4, editlog=/u01/app/hadoop/hadoop-
1.2.1/dfs/name/current/edits
```

```
14/08/06 00:28:28 INFO namenode.FSEditLog: close success: truncate to 4, editlog=/u01/app/hadoop/hadoop-1.2.1/dfs/name/current/edits
14/08/06 00:28:28 INFO common.Storage: Storage directory /u01/app/hadoop/hadoop-1.2.1/dfs/name has been successfully formatted.
14/08/06 00:28:28 INFO namenode.NameNode: SHUTDOWN_MSG:
/
SHUTDOWN_MSG: Shutting down NameNode at edydr1p2.us.oracle.com/192.168.100.102
*****/
```

14. 하둡파일 시스템을 올린다.

```
[scott@centos ~]$ cd
[scott@centos ~]$ start-all.sh
...
Are you sure you want to continue connecting (yes/no)? yes
```

하둡파일 시스템의 데몬이 잘 올라왔는지 확인한다. 아래와 같이 6개가 떠야 한다

```
[scott@centos ~]$ jps
9167 SecondaryNameNode
9030 DataNode
9402 TaskTracker
9250 JobTracker
9494 Jps
8883 NameNode
```

★잘 안되었을 때의 조치 방법

1. 하둡을 모두 내립니다.

```
[scott@centos conf]$ stop-all.sh
```

2. 디렉토리 권한을 조정합니다.

```
[scott@centos dfs]$ cd /home/scott/hadoop/hadoop-1.2.1/dfs 로 이동한 후에
권한 조정
(base) [scott@centos dfs]$ ls -l
total 0
drwxr-xr-x. 6 scott scott 106 Jan 5 12:39 data
drwxr-xr-x. 5 scott scott 80 Jan 5 12:39 name
(base) [scott@centos dfs]$
(base) [scott@centos dfs]$
```

3. 하둡 네임노드를 포맷한다.

```
[scott@centos conf]$ cd
[scott@centos ~]$ hadoop namenode -format
```

4. 하둡 데이터 노드와 네임노드의 데이터를 다 지웁니다.

```
[scott@centos dfs]$ pwd
/home/scott/hadoop/hadoop-1.2.1/dfs
[scott@centos dfs]$ ls
data  name
[scott@centos dfs]$ cd data
[scott@centos data]$ rm -rf *
[scott@centos data]$ cd ../name
[scott@centos name]$ rm -rf *
```

■ Hive 설치

"자바를 몰라도 rdbms 에 익숙한 데이터 분석가들을 위해서 SQL 을 이용해서 하둡의 맵리듀싱 프로그램을 지원하는 프로그램"

페이스북에서 만든 오픈소스

HiveQL

1. SELECT 는 되는데 update 와 delete 명령어는 지원 안함
2. from 절의 서브쿼리는 사용가능
3. select 문 사용시 having 절은 사용 불가능
4. PL/SQL 의 프로시저는 hive2.0 부터 가능 (보통은 SQL로 다 조회함)



SQL + C 언어를 이용해서 프로그래밍

▼ OLTP 파트 vs DW 파트

DW 파트 : DW 개발자, 데이터 분석가, DBA

1. hive 설치파일을 공유폴더(e://data) 에 둔다.

2. Hive 설치 파일(hadoop-1.2.1.tar.gz)을 scott으로 옮기고 압축을 푼다

```
$cd
$cp /media/sf_data/hive-0.12.0.tar.gz /home/scott/
$tar xvfz hive-0.12.0.tar.gz
```

3. hive 로 접속한다. (끝)

```
$ cd /home/scott/hive-0.12.0/bin
$ ./hive
```

```
hive> show tables;
```

문제209. hive 에서 emp 테이블을 생성하시오 !

```
drop table emp;
create table emp
(empno int,
```

```

ename string,
job string,
mgr int,
hiredate string,
sal int,
comm int,
deptno int)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE ;

```

문제210. 하둡 파일 시스템에 emp2.csv 를 올리시오 !

```

hive>exit;

$cd

$cp /media/sf_data/emp2.csv /home/scott

$hadoop fs -put emp2.csv emp2.csv

$hadoop fi -ls emp2.csv

여기서
Error: Could not find or load main class fi 라는 에러가 뜰시 아래와 같이 실행

$ . .bash_profile

$ hadoop fs -put emp2.csv emp2.csv

$ hadoop fs -ls emp2.csv

```

(점심시간 문제) 문제 211. 하둡 파일 시스템의 emp2.csv 를 emp 에 로드 하시오 !

```

$ cd hive-0.12.0/bin/
$ ./hive

drop table emp;

create table emp
(empno int,
ename string,
job string,
mgr int,
hiredate string,
sal int,
comm int,
deptno int)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE ;

hive> load data inpath '/user/scott/emp2.csv'
      overwrite into table emp;

hive> select * from emp;

```

■ 하둡 시스템이 정상인지 확인하는 명령어

```
$ jps
```

노드에 대한 설명



8619 NameNode : HDFS 의 모든 메타 데이터(data 의 위치정보)를 관리하고 클라이언트가 hdfs 에 저장된 파일에 접근할 수 있게 해준다.

10750 Jps : 현재 jps 명령어 수행한 프로세서 (나)

8982 JobTracker : 하둡 클러스터에 등록된 전체 잡의 스케줄링을 관리하고 모니터링하는 데몬

8895 SecondaryNameNode : 하둡 보조 네임노드로 네임노드의 파일 시스템 이미지 파일을 주기적으로 갱신하는 역할을 수행하는 노드 (메타 데이터를 최신것으로 유지시키는 작업)

8743 DataNode : hdfs 의 데이터를 입력하면 입력 데이터는 32mb 의 블록으로 나뉘어져서 여러대의 데이터 노드에 분산되어 저장된다. 그 데이터를 저장하는 노드

9111 TaskTracker : 사용자가 설정한 맵리듀스 프로그램을 실행하는 역할을 하며 하둡 데이터 노드에서 실행되는 데몬

* 하둡 운영에 문제가 생겼을때 조치방법

\$ jps ← 명령어를 수행했을때 위의 6개의 데몬 말고 RunJar 라는 프로세서가 뜨면 문제가 있으므로 반드시 kill 시킨다.

\$ kill -9 19093

■ 하둡 시스템이 정상인지 확인하는 명령어

\$ jps

노드에 대한 설명



8619 NameNode : HDFS 의 모든 메타 데이터(data 의 위치정보)를 관리하고 클라이언트가 hdfs 에 저장된 파일에 접근할 수 있게 해준다.

10750 Jps : 현재 jps 명령어 수행한 프로세서 (나)

8982 JobTracker : 하둡 클러스터에 등록된 전체 잡의 스케줄링을 관리하고 모니터링하는 데몬

8895 SecondaryNameNode : 하둡 보조 네임노드로 네임노드의 파일 시스템 이미지 파일을 주기적으로 갱신하는 역할을 수행하는 노드
(메타 데이터를 최신것으로 유지시키는 작업)

8743 DataNode : hdfs 의 데이터를 입력하면 입력 데이터는 32mb 의 블록으로 나뉘어져서 여러대의 데이터 노드에 분산되어 저장된다. 그 데이터를 저장하는 노드

9111 TaskTracker : 사용자가 설정한 맵리듀스 프로그램을 실행하는 역할을 하며 하둡 데이터 노드에서 실행되는 데몬

* 하둡 운영에 문제가 생겼을때 조치방법

\$ jps ← 명령어를 수행했을때 위의 6개의 데몬 말고 RunJar 라는 프로세서가 뜨면 문제가 있으므로 반드시 kill 시킨다.

\$ kill -9 19093(작업중인 프로세서의 번호)

```
$ jps
6471 DataNode
9056 Jps
6722 JobTracker
6629 SecondaryNameNode
6851 TaskTracker
6352 NameNode
8781 RunJar

$ kill -9 8781

$ start-all.sh

$ jps
```

문제212. 월급이 3000 인 사원의 이름과 월급을 출력하시오 !

```
hive> select ename, sal
> from emp
> where sal = 3000;
```

문제213. 직업이 SALESMAN 인 사원들의 이름과 직업을 출력하시오 !

```
hive> select ename, job
from emp
```



```
where job='SALESMAN' ;
```

```
MARTIN SALESMAN
ALLEN SALESMAN
TURNER SALESMAN
WARD SALESMAN
```

문제214. 월급이 1200 이상이고 직업이 SALESMAN 인 사원들의 이름과 월급과 직업을 출력하시오 !

```
hive> SELECT ename, job, sal
> from emp
> where sal >= 1200 and job='SALESMAN';
```

설명: emp 와 같은 작은 데이터를 검색하기 위해서 하둡을 사용하는것이 아니라 대용량 데이터를 빠르게 검색하기 위해서 하둡을 사용하는 것이다.
emp.csv 는 hive 보다는 maria db 를 이용하는게 좋다.
몇천만건, 몇억건씩 되는 큰 대용량 csv 파일이나 텍스트 파일은 hive 를 이용하면 빠르게 검색할 수 있다.

문제215. dept 테이블을 생성하시오 !

순서:

1. dept2.csv 파일을 하둡 파일 시스템에 올린다.
2. dept 테이블 생성 스크립트를 가지고 생성한다.
3. 하둡 파일 시스템에 올라온 dept2.csv 파일을 dept 테이블에입력 한다.

```
$ cd
$ cp /media/sf_data/dept2.csv /home/scott/
$ hadoop fs -put dept2.csv dept2.csv
$ hadoop fs -ls dept2.csv
```

```
hive> create table dept
( deptno int,          # 컬럼은 문자면 string, 숫자면 int 를 씁니다.
  dname string,
  loc string )
ROW FORMAT DELIMITED   # 반드시 써줘야하는 문법
FIELDS TERMINATED BY ',' # 컬럼과 컬럼은 콤마로 구분하겠다.
LINES TERMINATED BY '\n' # 행과 행은 엔터 구분한다.
STORED AS TEXTFILE ;    # 반드시 써줘야하는 문법

hive> load data inpath '/user/scott/dept2.csv'
overwrite into table dept;

hive> select * from dept;
```

문제216. 이름과 부서위치를 출력하시오

```
hive> select e.ename, d.loc
> from emp e join dept d
> on (e.deptno=d.deptno);
```

설명: hive 에서 조인하려면 오라클 조인문법 말고 1999 ansi 문법으로 조인해야한다.

문제217. DALLAS 에서 근무하는 직원들의 이름과 월급과 부서위치를 출력하시오 !

```
hive> select  e.ename, e.sal, d.loc
from emp e join dept d
on ( e.deptno=d.deptno)
where d.loc='DALLAS';
```

문제218. 직업, 직업별 토털월급을 출력하시오 !

```
hive> select job, sum(sal)
> from emp
> group by job;
```

문제219. 부서번호, 부서번호별 평균월급을 출력하는데 부서번호별 평균월급이 높은것 부터 출력하시오 !

```
hive> select deptno, avg(sal) as avgsal
from emp
group by deptno
order by avgsal desc;
```

설명: hive 는 오라클과는 다르게 그룹함수를 써서 정렬을 할때
order by 절에 컬럼별칭을 사용해야 한다.

문제220. 위의 결과를 다시 출력하는데 소수점 이하는 안나오게 반올림하시오

```
hive> select deptno, round( avg(sal) ) as avgsal
from emp
group by deptno
order by avgsal desc;
```

오라클과 hive 의 함수의 다른점

오라클	vs	하이버
to_char(hiredate, 'RRRR')		year(to_date(hiredate))
to_char(hiredate, 'MM')		month(to_date(hiredate))
to_char(hiredate, 'DD')		day(to_date(hiredate))

문제221. 이름과 입사한 년도(4자리) 를 출력하시오 !

```
KING 1981
BLAKE 1981
:
```

답:

```
hive> select  ename, year( to_date(hiredate) )
from emp;
```

문제222. 1981 년도에 입사한 직원들의 이름과 입사일을 출력하시오 !

```
hive> select  ename, hiredate
from emp
where year( to_date(hiredate) ) = '1981';
```

설명: 오라클을 이용하지 않고 하둡을 이용하는 이유 ?

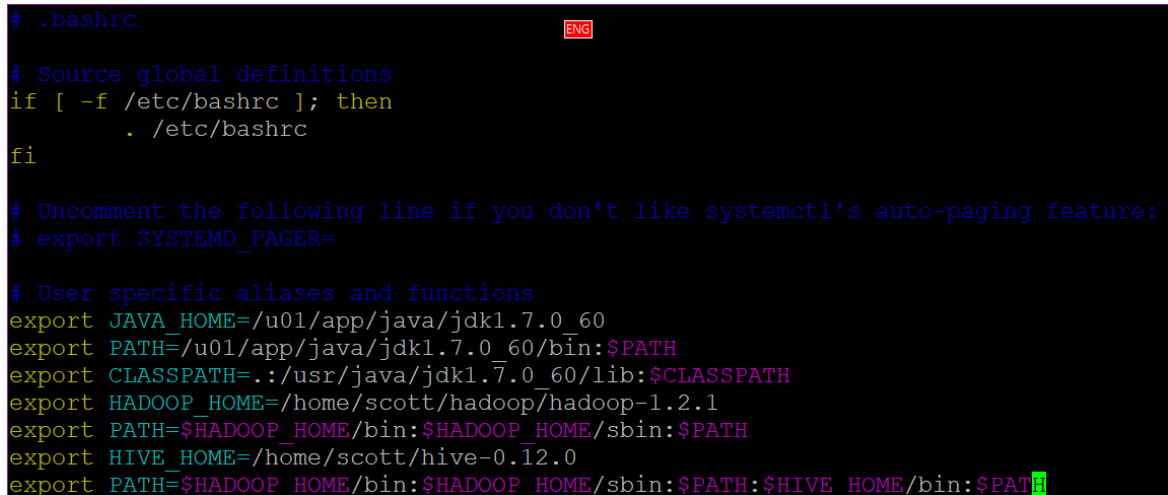
1. 하둡 is free (of charge)
2. 분산파일시스템의 장점을 이용할 수 있어서이다.
(여러개의 컴퓨터를 하나로 묶어서 슈퍼컴퓨터를 만들 수 있다)
3. 큰 텍스트 파일을 테이블에 insert 하지 않고 그냥 os 에
그냥 두고 링크만 걸어서 select 할 수 있다.
(오라클의 external table 과 유사하다)

💡 리눅스 os 에서 어디에서든지 hive 라고 치면 hive 에 접속하게 하는 방법

1. .bashrc 열어서 아래의 두줄을 맨 아래에 넣는다.

```
$ cd
$ vi .bashrc
export HIVE_HOME=/home/scott/hive-0.12.0
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH:$HIVE_HOME/bin:$PATH
```

▼ 결과



```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
export JAVA_HOME=/u01/app/java/jdk1.7.0_60
export PATH=/u01/app/java/jdk1.7.0_60/bin:$PATH
export CLASSPATH=.:usr/java/jdk1.7.0_60/lib:$CLASSPATH
export HADOOP_HOME=/home/scott/hadoop/hadoop-1.2.1
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
export HIVE_HOME=/home/scott/hive-0.12.0
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH:$HIVE_HOME/bin:$PATH
```

2. 그리고 저장하고 나와서 .bashrc 스크립트를 수행한다.

```
$ . .bashrc
```

3. hive 로 접속한다.

☀영화 평점에 대한 큰 데이터를 내려받아 hive 에서 분석을 해보기

1. 먼저 대용량 텍스트 파일의 압축파일을 다운로드 받는다.

```
$ cd
$ wget http://www.grouplens.org/system/files/ml-1m.zip
```

2. 위의 압축파일의 압축을 해제합니다.

```
$ unzip ml-1m.zip
$ cd ml-1m/
```

문제223. movies.dat 파일의 위의 10줄만 열어서 보시오 (head 명령어)

```
$ head 10 movies.dat
```

문제224. movies.dat 파일의 총 라인수가 어떻게 되는지 확인하시오 !
(wc 명령어사용)

```
$ wc -l movies.dat
```

문제225. ratings.dat 파일의 총 라인수가 어떻게 되는지 확인하시오 !

```
$ wc -l ratings.dat
1000209 ratings.dat
```

문제226. ratings.dat 파일의 위에 10줄만 출력하시오 !

```
$ head 10 ratings.dat
```

설명: ratings.dat 의 컬럼 UserID::MovieID::Rating(평점)::Timestamp
movies.dat 의 컬럼 MovieID::Title::Genres

위의 컬럼에 대한 자세한 소개는 README.txt 를 읽어보면 된다.

- UserIDs range between 1 and 6040
- MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratingonly)

- Timestamp is represented in seconds since the epoch are turned by time(2)
- Each user has at least 20 ratings

설명: movies.dat, ratings.dat, users.dat 이 빅데이터를 하둡 저장하고
hive 를 이용해서 SQL 로 분석을 하고자 한다면 ?

3. movies.dat, ratings.dat, users.dat 의 컬럼과 컬럼 구분을 콤마(,) 가 되도록 데이터 전처리를 해야한다. (리눅스 명령어)

e.g)

```
$ head 10 movies.dat
```

설명: 확인해보니 컬럼과 컬럼이 :: 로 구분되어져 있다.

::를 없애주는 전처리 과정 (notion에서 복붙해서 안될경우 메모장 등을 경유해서 해볼 것!)

```
$ sed s/::/,/g movies.dat >> movies_coma.dat
$ sed s/::/,/g movies.dat >> movies_coma.dat
$ head -5 movies_coma.dat

$ sed s/::/,/g ratings.dat >> ratings_coma.dat
$ head -5 ratings_coma.dat

$ sed s/::/,/g users.dat >> users_coma.dat
$ head -5 users_coma.dat
```

4. hive 에서 테이블 생성 스크립트를 만든다. (테이블명: movies, ratings, users)

```
$ hive
hive> create table users
( userid int,
  gender string,
  age int,
  occupation int,
  zipcode string )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE ;

hive> create table movies
( movieid int,
  title string,
  genres string )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE ;

hive> create table ratings
( userid int,
  movieid int,
  rating int,
  tstamp string )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE ;
```

1) movies_coma.dat, ratings_coma.dat, users_coma.dat 이 3개의 파일을하둡 파일 시스템에 올린다.

윈도우 ----> 리눅스 ----> 하둡 파일 시스템
실제세상 꿈 꿈

```
$ cd /home/scott/ml-1m
$ ls
$ hadoop fs -put movies_coma.dat movies_coma.dat
$ hadoop fs -put ratings_coma.dat ratings_coma.dat
$ hadoop fs -put users_coma.dat users_coma.dat
$ hadoop fs -ls
```

2) 하둡 파일 시스템에 올린 3개의 파일을 각각 movies 와 users , ratings 테이블에 로드 한다.

```
hive> load data local inpath '/home/scott/ml-1m/movies_coma.dat'
overwrite into table movies;
hive> load data local inpath '/home/scott/ml-1m/users_coma.dat'
overwrite into table users;

hive> load data local inpath '/home/scott/ml-1m/ratings_coma.dat'
overwrite into table ratings;
```

3) load한 데이터가 잘 load되었는지 확인

```
hive> select count(*) from movies;
hive> select count(*) from users;
hive> select count(*) from ratings;
hive> select * from movies limit 5;
hive> select * from ratings limit 5;
```

!ratings 는 영화를 본 사람들이 영화번호, 영화평점을 올려놓은 데이터

문제227. (오늘의 마지막 문제) ratings 테이블에서 영화번호, 영화번호별 영화평점의 평균값을 출력하는데 영화번호별 영화평점의 평균값이 높은것부터 출력하시오 !

```
select movieid, avg(rating) as avgr
from ratings
group by movieid
order by avgr desc;
```

[21.01.11 리눅스 강의 Day 10 //하둡]

지난주에 배웠던 리눅스와 하둡 수업 복습

1. 왜 리눅스를 배워야 하는가??
2. 리눅스 설치
3. 리눅스 기본명령어
4. vi 편집기 명령어
5. 리눅스에 아나콘다와 spyder설치
6. 리눅스 권한관리 명령어
7. 디스크 관리 명령어
8. 프로세서 관리 명령어
9. shellscript 작성법

10. Hadoop 설치

11. Hive설치

분석을 위한 도구

파이썬, R → for 시각화

현업에서 어떻게 하둡과 하이브, 스칼라를 이용하고 있는지 현직자 인터뷰를 해보니..

1. 하둡 & Hive를 약 50%, 그외의 50%는 스칼라
 2. 하둡, 하이브, 스칼라에서는 데이터(csv, text) 를 로드하는 테이블 생성을 위주로함
 3. 하둡과 하이브와 스칼라에서 데이터 검색을 해서 분석
(GUI 툴인 squirrel(다람쥐)를 이용 → SQL developer 와 같은 툴)
 4. 하이브와 파이썬 연동, 스칼라와 파이썬 연동을해서 데이터 시각화와 분석을 하는 것이 데이터 분석가의 목표
- e.g) '큰 질문' (코로나는 어느 장소에서 많이 감염이 되는가?)
를 해결 하기 위해 다음과 같은 활동을 수행해야 함.

1. 코로나 환자들의 동선 정보가 있는 데이터를 구한다.(text , csv)
2. 리눅스 서버에 데이터를 넣고, 하둡 하이브에서 테이블을 생성한다.
3. 하이브(다람쥐 GUI) 를 이용해서 쿼리 수행해서 장소별 순위를 출력한다.
4. 하이브와 파이썬을 연동해서 시각화(막대그래프, 파이그래프 등)
5. 분석을 의뢰한 고객에게 시각화된 자료와 함께 메일을 보낸다.

리눅스와 하둡시험 (NCS 시험) : 위의 질문을 스스로 정해서 데이터를 구해서 하둡에 넣고 분석을 해서 시각화한 결과물(워드,한글, PPT등) 로 제출

로 진행할 예정! 어떤 질문을 할지 생각을 해둘 것!

1. 하둡 시스템이 정상인지 확인

`$jps` 입력 // 아래와 같이 6개가 나와야 정상

```
2795 DataNode
2930 SecondaryNameNode
4879 TaskTracker
4730 JobTracker
4955 Jps
4463 NameNode
```

6개인 경우

```
$start-all.sh
$jps
```

2. 하이브로 접속

```
$hive
```

만약 jps가 6개가 아니라면

- 1) 하둡 data노드와 name 노드의 디렉토리 데이터를 다 지운다.

```
cd $HADOOP_HOME/dfs
cd name
rm -rf *

cd ..

cd data
rm -rf *
```

2) 네임노드 포맷한다.

```
hadoop namenode -format
>> Y 입력
```

3) start-all.sh

4) jps

로 6개 뜨는지 확인 후

5) hive접속

문제 228.

문제 232. 사원번호, 이름, 월급, 월급의 누적치를 출력하시오.

```
SQL>
select empno, ename, sal, sum(sal) over (order by empno asc) sumsal
from emp;
```

```

      OLTP      VS      DW ( DataWarehouse)
      ↓          ↓
      오라클      vs      하둡 , 스칼라
      ↓          ↓
      빨리빨리 결과를      데이터 분석함수
      봐야하는 쿼리들

```

문제233. 오라클의 데이터 분석함수의 lag 와 lead 함수가 hive 에서도 되는지 확인하시오 !

```
SQL> select deptno, ename, sal,
lag(sal,1) over ( order by empno asc) 그전행,
lead(sal,1) over ( order by empno asc) 다음행
from emp;
```

```
hive> select deptno, ename, sal,
lag(sal,1,0) over ( order by empno asc) lag,
lead(sal,1,0) over ( order by empno asc) lead
from emp;
```

문제234. 오라클의 데이터 분석함수의 lag 와 lead 함수가 hive 에서도 되는지 확인하시오 !

```
SQL> select deptno, ename, sal,
lag(sal,1) over ( partition by deptno
order by empno asc) 그전행,
lead(sal,1) over ( partition by deptno
```



```
order by empno asc) 다음행
from emp;
```

```
hive> select deptno, ename, sal,
lag(sal,1,0) over ( partition by deptno
order by empno asc) lag,
lead(sal,1,0) over ( partition by deptno
order by empno asc) lead
from emp;
```

문제235. 부서번호, 이름, 월급, 자기가 속한 부서번호의 평균월급을 출력하시오 !

```
hive> select deptno, ename, sal,
avg(sal) over ( partition by deptno) avgsal
from emp;
```

설명: hive 에서 결과 출력에서 컬럼명 나오게 하는 방법

```
hive> set hive.cli.print.header=true;
```

문제236. 위의 결과를 다시 출력하는데 자기의 월급이 자기가 속한 부서번호의
평균월급보다 더 큰 직원들만 출력하시오 !
(힌트: from 절의 서브쿼리를 사용해야 한다.)

```
select *
from ( select deptno, ename, sal,
avg(sal) over ( partition by deptno) avgsal
from emp ) aaa
where sal > avgsal ;
```

설명: hive 에서 from 절의 서브쿼리를 사용하려면 반드시 alias 를 사용해야 합니다.

문제237. DW 에서 많이 사용하는 SQL 중 하나가 WITH 절입니다. 오라클의 WITH 절이
Hive 에서도 되는지 확인하세요 ~

설명: with 절은 hive 1.2.1 버전은 지원 안되고 그 이후 버전부터 가능하다.

```
with emp77 ( select job, sum(sal) sumsal
from emp
group by job )
select job, sumsal
from emp77;
```

문제238. DW 에서 많이 사용하는 집계값 구하는 쿼리를 수행하시오 !
부서번호, 부서번호별 토달월급을 출력하는데 맨 아래쪽에 전체 토달월급이
출력되게하시오 !

```
Oracle> select deptno, sum(sal)
from emp
group by rollup(deptno);
```

```
Hive> select deptno, sum(sal)
from emp
group by deptno with rollup;
```

문제239. 위의 결과에서 전체 토탈월급이 아래에 나오도록 order by 절을 사용하시오 !

```
Hive> select deptno, sum(sal) sumsal
from emp
group by deptno with rollup
order by sumsal asc;
```

큰 질문 ? 코로나는 어느 장소에서 많이 감염이 되는가 ?



1. 코로나 환자들의 동선 정보가 있는 데이터를 구한다. (text, csv)
2. 리눅스 서버에 데이터를 넣고 하둡 하이브에서 테이블 생성한다.
3. 하이브(다람쥐GUI) 를 이용해서 쿼리를 수행해서 장소별 순위를 출력한다.
4. 하이브와 파이썬을 연동해서 시각화(막대 그래프, 파이 그래프)
5. 분석을 의뢰한 고객(직원)에게 시각화된 자료와 함께 메일을 보낸다.

문제240. 케글에서 코로나 데이터 분석에 필요한 코로나 데이터를 내려받아리눅스 시스템에 넣으시오 !
/home/scott 밑에 가져다 두세요 !

```
cp /media/sf_data/PatientInfo.csv /home/scott/
cp /media/sf_data/Case.csv /home/scott/
```

문제241. PatientInfo.csv 를 hive 의 테이블로 구성하시오 !

```
$ head PatientInfo.csv
$ vi PatientInfo.csv
로 들어가 맨위의 컬럼행을 dd 로 지우고 저장하고
```

```
$ hadoop fs -put PatientInfo.csv PatientInfo.csv
$ hive
hive>
drop table patient;

create table patient
(patient_id string,
sex string,
age string,
country string,
province string,
city string,
infection_case string,
infected_by string,
contact_number int,
symptom_onset_date string,
```

```
confirmed_date string,
released_date string,
deceased_date string,
state string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE ;
```

💡 만약 하둡에 올라간 데이터를 지우고 싶다면?

```
$ hadoop fs -put /user/scott/PatientInfo.csv
```

문제242. (점심시간 문제) 코로나 환자 데이터를 하둡에 올리고 hive 로 아래와 같이위에 5건만 조회하고 결과를 캡처해서 올리시오 ~

```
hive> load data inpath '/user/scott/PatientInfo.csv'
overwrite into table patient;
```

```
hive>select * from patient limit 5;
```

문제243. 나이대(age) , 나이대별 인원수를 출력하시오 !

```
hive> select age, count(*)
from patient
group by age;
```

- patient 테이블 다시 생성 스크립트

```
hive> drop table patient;
hive> create table patient
(patient_id string,
sex string,
age string,
country string,
province string,
city string,
infection_case string,
infected_by string,
contact_number int,
symptom_onset_date string,
confirmed_date string,
released_date string,
deceased_date string,
state string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE ;

hive> exit;
```

```
$ hadoop fs -put PatientInfo.csv PatientInfo.csv
$ hive
hive> load data inpath '/user/scott/PatientInfo.csv'
overwrite into table patient;
```

문제244. 위의 결과를 다시 출력하는데 인원수가 높은것부터 출력하시오 !

```
hive> select age, count(*) cnt
from patient
group by age
order by cnt desc;
```

문제245. 오라클의 grouping sets 가 hive 에서도 지원되는지 확인하시오 !

```
Oracle> select deptno, sum(sal)
from emp
group by grouping sets( deptno, ( ) );
```

```
hive> select deptno, sum(sal)
from emp
group by deptno grouping sets( deptno, ( ) );
```

설명: () 는 전체 토탈이다.

문제246. 코로나 환자 정보에서 나이대, 나이대별 인원수를 출력하는데 맨 아래에 전체 인원수도 출력하시오 !

```
hive> select age, count(*) cnt
from patient
group by age grouping sets( age, ( ) )
order by cnt asc;
```

하둡 + HIVE 50% , 스칼라 50%

문제247. DW 쪽에서 가장 빈번히 수행하는 쿼리문이 아래의 SQL 인데 이 아래의 SQL 을 HIVE 로 구현하시오 !

```
SQL> select sum( decode( deptno, 10, sal) ) as "10",
sum( decode( deptno, 20, sal) ) as "20",
sum( decode( deptno, 30, sal) ) as "30"
from emp;
```

```
hive> set hive.cli.print.header=true;
hive> select sum( case when deptno=10 then sal end) dept10,
sum( case when deptno=20 then sal end) dept20,
sum( case when deptno=30 then sal end) dept30
from emp;
```

```
dept10 dept20 dept30
8750 10875 9400
```

문제248. DW 쪽에서 많이 사용하는 아래의 SQL 에 GROUP by 까지 사용한 결과를 hive 로 구현하시오 !

```
SQL> select job, sum( decode( deptno, 10, sal) ) as "10",
sum( decode( deptno, 20, sal) ) as "20",
sum( decode( deptno, 30, sal) ) as "30"
from emp
group by job;
```

```
hive> select job, sum( case when deptno=10 then sal end) dept10,
sum( case when deptno=20 then sal end) dept20,
sum( case when deptno=30 then sal end) dept30
from emp
group by job;
```

!?현업에서 hive 사용할 때 중요한게 csv 파일을 받았으면 csv 파일을 가지고 table 생성을 잘 할 수 있어야 합니다.

문제249. 코로나 데이터중에 감염경로를 알려주는 Case.csv 를 가지고 hive 에 테이블을 생성하시오 !

case_id,province,city,group,infection_case,confirmed,latitude,longitude

```
create table doong
(caseid int,
province string,
city string,
group string,
infection string,
confirmed int,
latitude float,
longitude float)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE ;
```

```
load data inpath '/user/scott/Case.csv'
overwrite into table doong;
```

문제250. 감염경로(infection_case)를 중복제거해서 출력하시오 !

```
SQL> select distinct infection
from doong ;
```

☀ 스파크 설치

1. scott 의 홈디렉토리로 이동합니다.

```
$ cd
```

2. 설치 파일을 다운로드 받는다.

```
$ wget https://archive.apache.org/dist/spark/spark-2.0.2/spark-2.0.2-bin-hadoop2.7.tgz
```

3. 압축을 풉니다.

```
$ tar xvfz spark-2.0.2-bin-hadoop2.7.tgz
```

4. 압축을 풀고 생긴 디렉토리의 이름을 spark 로 변경합니다.

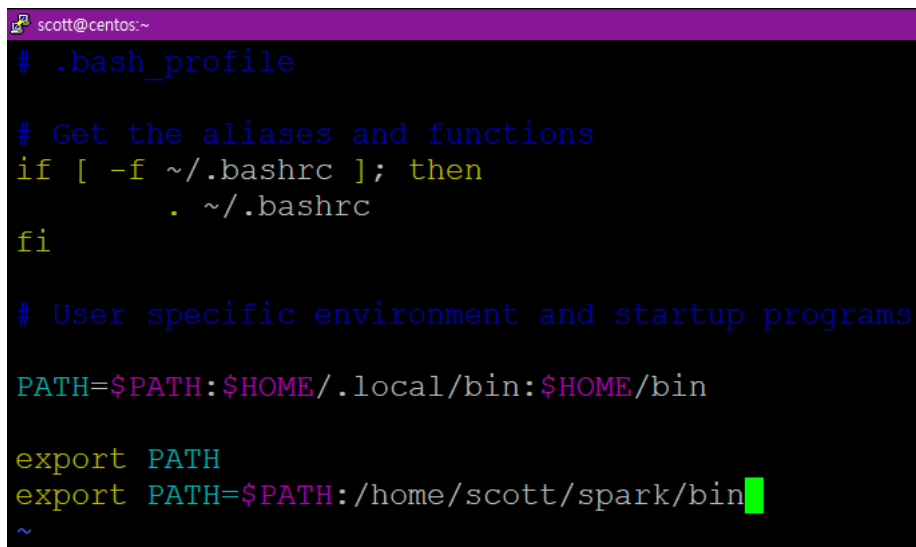
```
$ mv spark-2.0.2-bin-hadoop2.7 spark
```

5. .bash_profile 를 열어서 맨 아래에 아래의 export 문을 입력합니다.

```
$ vi .bash_profile
```

```
export PATH=$PATH:/home/scott/spark/bin
```

▼ 결과



```
scott@centos:~  
# .bash_profile  
  
# Get the aliases and functions  
if [ -f ~/.bashrc ]; then  
    . ~/.bashrc  
fi  
  
# User specific environment and startup programs  
  
PATH=$PATH:$HOME/.local/bin:$HOME/bin  
  
export PATH  
export PATH=$PATH:/home/scott/spark/bin  
~
```

6. .bash_profile 을 수행합니다.

```
$ source .bash_profile
```

7. spark-shell 로 접속하여 테이블 생성을 하고 select 합니다.

아래의 사이트를 참고 합니다.

https://www.tutorialspoint.com/spark_sql/spark_sql_hive_tables.htm

```
[scott@localhost ~]$ vi employee.txt  
[scott@localhost ~]$  
[scott@localhost ~]$  
[scott@localhost ~]$ pwd  
/home/scott  
[scott@localhost ~]$ cat employee.txt  
1201,satish,25  
1202,krishna,28  
1203,amith,39  
1204,javed,23  
1205,prudvi,23  
  
[scott@localhost ~]$ spark-shell  
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel).  
21/01/09 08:54:43 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes w  
21/01/09 08:54:43 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 10.0.2.15 i
```

```

      _ _ _ _ _
     / / _ _ _ / /
    _ \ \ _ \ \ _ \ \
   / _ \ _ \ / _ \ / _ \
  / _ \ _ \ / _ \ / _ \
                                     version 2.0.2
   / /

```

```
scala> val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
```

```
scala> sqlContext.sql("CREATE TABLE IF NOT EXISTS employee(id INT,name STRING,age INT) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'")
```

```
scala> sqlContext.sql("LOAD DATA LOCAL INPATH 'employee.txt' INTO TABLE employee")
res1: org.apache.spark.sql.DataFrame = []
```

```
scala> val result = sqlContext.sql("FROM employee SELECT id, name, age")
result: org.apache.spark.sql.DataFrame = [id: int, name: string ... 1 more field]
```

```
scala> result.show()
+----+-----+-----+
| id | name | age |
+----+-----+-----+
|1201| satish | 25 |
|1202| krishna | 28 |
|1203| amith | 39 |
|1204| javed | 23 |
|1205| prudvi | 23 |
+----+-----+-----+
```

```
scala> sql("select * from employee").show()
+-----+-----+
| id | name | age |
+-----+-----+
|1201| satish | 25 |
|1202| krishna | 28 |
|1203| amith | 39 |
|1204| javed | 23 |
```


```
|1205| prudvi| 23|  
+-----+-----+-----+
```

문제251. id 가 1202 인 사원의 이름과 나이를 출력하시오 !

```
scala> sql("select * from employee where id=1202").show()
```

문제252. 이름이 satish 인 사원의 이름과 나이를 출력하시오 !

```
scala> sql("select * from employee where name='satish']").show()
```

 면접질문 : 데이터 분석시에 하둡 하이브 또는 스칼라에서 구현해야하는 것 ?

" 빅데이터를 저장할 테이블 생성 입니다. "

그 이후에 파이썬에서 시각화 또는 머신러닝 데이터 분석을 하는 것!

문제253. emp 테이블을 스칼라에서 생성하시오

```
scala> val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)  
scala> sqlContext.sql("create table IF NOT EXISTS emp (empno int, ename string, job string, mgr int, hiredate string, sal int
```

문제254. 스칼라에서 생성한 emp 테이블에 emp.csv 를 로드하시오 !

```
sqlContext.sql("LOAD DATA LOCAL INPATH '/home/scott/emp2.txt' INTO TABLE emp")  
sql("select * from emp").show()
```

문제255. (오늘의 마지막 문제) 스칼라에서 dept 테이블을 생성하고 dept 테이블에 데이터를 입력하시오
(캡처해서 카페에 업로드!)

tip. NEWYORK 을 붙여서 넣으세요 // dept2.csv 파일을 이용해서 하세요 ~~~

[21.01.12 리눅스 강의 Day 11 //하둡]

■ 리눅스 수업후 하둡 수업 내용

데이터 엔지니어, 데이터 분석가, 데이터 사이언티스, 딥러닝 개발자

1. 리눅스 수업(리눅스 설치, 리눅스 기본명령어, vi 편집기, 기타)
2. 하둡 설치
3. hive 설치
4. hive 에서 SQL 로 빅데이터를 검색하는 연습 (오라클과 Hive 비교)
5. 스칼라 설치
6. 스칼라에서 SQL 로 빅데이터를 검색하는 연습
7. (최종목표) 하이브와 스칼라에서 데이터를 검색하고 전처리한후
이 데이터를 파이썬에서 시각화

■ 스칼라에서 emp 테이블이 어느자리에서는 만들어지고 어느 자리에서는 안만들어지는 이유 ?

일단 스칼라에서 SQL 로 작업을 하다가 예러가 났는데 그 예러가 좀 난이도가 높은 예러였으면 다음에 정상적인 스칼라 SQL 문법을 수행해도 수행이 안되고 계속 예러가 난다.

이럴때는 스칼라를 내렸다 올리고 다시 스칼라에 접속하면 해결됩니다.

■ 스칼라를 내렸다 올리고 다시 접속하는 명령어 정리

1. 스파크의 sbin 디렉토리로 이동합니다.

```
$ cd /home/scott/spark/sbin
$ ls
```

start-all.sh 와 stop-all.sh 가 있는지 확인해야 합니다.

설명: 위의 스파크 디렉토리의 start-all.sh 와 stop-all.sh 는 하둡의 start-all.sh 와 stop-all.sh 와는 별개의 파일입니다.

```
$ ./stop-all.sh
$ ./start-all.sh
$ spark-shell
```

■ 스칼라에서 emp 테이블 생성하는 방법

1. hive SQL 을 스칼라에서 쓰겠다는 명령어

```
scalar> val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
```

2. emp 테이블을 생성하는 명령어

```
scalar>sqlContext.sql("create table IF NOT EXISTS emp (empno int, ename string, job string, mgr int, hiredate string, sal int,
```

3. emp 테이블에 emp2.txt 를 로드하는 명령어

```
scalar>sqlContext.sql("LOAD DATA LOCAL INPATH '/home/scott/emp2.txt' INTO TABLE emp")

결과
res1: org.apache.spark.sql.DataFrame = []
```

4. emp 테이블의 내용을 확인하는 명령어

```
scala> sql("select * from emp").show()
```

만약에 emp 테이블의 로우의 수가 중복되어서 28개 보인다면 ? → 테이블을 drop 하고 다시 생성하면 된다.

■ emp 테이블을 drop 하고 다시 생성하는 방법

```
scalar> sql("drop table emp")
```

1. hive SQL 을 스칼라에서 쓰겠다는 명령어

```
scalar> val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
```

2. emp 테이블을 생성하는 명령어

```
scalar>sqlContext.sql("create table IF NOT EXISTS emp (empno int, ename string, job string, mgr int, hiredate string, sal int,
```

3. emp 테이블에 emp2.txt 를 로드하는 명령어

```
scalar>sqlContext.sql("LOAD DATA LOCAL INPATH '/home/scott/emp2.txt' INTO TABLE emp")
```

4. emp 테이블의 내용을 확인하는 명령어

```
scalar> sql("select * from emp").show()
```

5. emp 테이블이 전체 몇건인 확인하는 명령어

```
scalar> sql("select * from emp").count()  
res7: Long = 14
```

문제256. dept2.csv 를 복사해서 dept2.txt 로 생성하고 스칼라에서 dept 테이블을 생성하시오 !

```
(또다른 터미널창) $ cp dept2.csv dept2.txt
```

```
(또다른 터미널창) $ cat dept2.txt
```

```
(스칼라 접속창) scalar> val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
```

```
(스칼라 접속창) scalar> sql("drop table dept")
```

```
(스칼라 접속창) scalar> sqlContext.sql("CREATE TABLE IF NOT EXISTS dept(deptno int, dname string, loc string) ROW FORMAT DELIMITED
```

```
(스칼라 접속창) scalar> sqlContext.sql("LOAD DATA LOCAL INPATH '/home/scott/dept2.txt' INTO TABLE dept")
```

```
(스칼라 접속창) scalar> sql("select * from dept").show()
```

■ 아래와 같이 나오면 준비 작업 끝

```
(스칼라 접속창) scalar> sql("select * from emp").count()  
14건인지 확인
```

```
(스칼라 접속창) scala> sql("select * from dept").count()  
4건인지 확인
```

문제257. 스칼라에서 부서번호가 30번인 직원들의 모든 데이터를 출력하시오 !

```
scala> sql("select * from emp where deptno=30").show()
```

문제258. 위에서 출력된 결과의 건수가 어떻게 되는지 확인하시오 !

```
scala> sql("select * from emp where deptno=30").count()
```

문제259. 직업이 SALESMAN 인 직원들의 이름과 직업과 월급을 출력하시오 !

```
scala> sql("select ename, job, sal from emp where job='SALESMAN'").show()
```

설명: 스칼라는 in memory db 여서 hive 보다 속도가 빠릅니다.

문제260. 부서 테이블 전체를 출력하시오 !

```
scala> sql("select * from dept").show()
```

문제261. (조인이 가능한지 확인) 이름과 부서위치를 출력하시오 !

힌트 : 1999 ANSI 조인 문법으로 수행해야함

```
scala> sql("select e.ename, d.loc from emp e join dept d on (e.deptno=d.deptno)").show()
```

설명: 한줄로 작성해야 한다!

문제262. DALLAS 에서 근무하는 직원들의 이름과 부서위치를 출력하시오 !

```
scala> sql("""select e.ename, d.loc from emp e join dept d  
on (e.deptno=d.deptno)  
where d.loc='DALLAS'""").show()
```

설명 : 스칼라에서 여러줄로 작성을 하기 위해서는 "를 앞뒤로 3개씩 둘러줘야 한다.

문제263. 아우터 조인도 가능한지 확인하세요. 아래의 SQL을 scala 에서 돌려보세요.

```
SQL> select e.ename, d.loc  
from emp e right outer join dept d  
on ( e.deptno = d.deptno)
```

```
scala> sql("""select e.ename, d.loc
from emp e right outer join dept d
on ( e.deptno = d.deptno)""").show()
```

문제264. (서브쿼리가 가능한지 확인) JONES 보다 더 많은 월급을 받는 직원들의 이름과 월급을 출력하십시오 !

```
sql(""" select ename, sal
from emp
where sal > (select sal from emp where ename='JONES') """).show()
```

문제265. (서브쿼리가 가능한지 확인) ALLEN 보다 늦게 입사한 직원들의 이름과 입사일을 출력하십시오 !

```
scala> sql("""select ename, hiredate
from emp
where hiredate > (select hiredate
from emp
where ename = 'ALLEN')""").show()
```

문제266. (분석함수가 지원되는지 확인) 이름, 월급, 월급에 대한 순위를 출력하십시오 !

```
sql("""select ename,sal, rank()over (order by sal desc)
from emp""").show()
```

문제267. 직업, 직업별 토탈월급을 출력하십시오 !

```
sql("""select job, sum(sal)
from emp
group by job""").show()
```

문제268. (스칼라의 장점중에서 하나가 결과값중에 숫자값에 대한 통계정보를 확인할 수 있습니다.) 위의 결과중에 숫자로 나오는 값에 대한 통계값을 출력하십시오!

```
scala> sql("""select job, sum(sal)
from emp
group by job""").describe().show()
```

설명: 카페에 가면 스칼라 메뉴얼 사이트가 있는데 거기서 확인합니다.
게시글 385 스파크 명령어 메뉴얼



스파크 SQL은 RDD에서 표현하지 못하는 스키마 정보를 표현가능하도록 보완된 모델입니다.

대표적으로 **데이터 프레임**과 **데이터셋**이 있습니다.

스파크 SQL은 언어별 사용유무는 아래와 같습니다.

Aa 스파크 SQL	📄 스칼라	📄 자바	📄 파이썬
데이터 프레임	@2021년 10월 1일		@2021년 10월 1일
데이터 셋	@2021년 10월 1일	@2021년 10월 1일	

1) 연산 종류

- 트랜스포메이션 연산
- 액션연산

2) 프로그래밍 구성요소

스파크 SQL을 사용하여 프로그래밍 시 필요한 구성요소는 아래와 같습니다.

Aa 구성요소	📖 의미
스파크 세션	1. 데이터 프레임을 생성하기 위해 이용. 2. 인스턴스 생성을 위한 build() 메서드 제공. 하이브 지원 default
데이터 셋	RDD와 같은 타입 기반 연산 + SQL과 같은 비타입 연산 제공
데이터 프레임	ROW 타입으로 구성된 데이터 셋
DataFrameReader	1. SparkSession의 read() 메서드를 통해 접근 가능. ["jdbc", "json", "parquet"] 등 다양한 input 제공.
DataFrameWriter	1. SparkSession의 write() 메서드를 통해 접근 가능
로우, 칼럼	데이터 프레임을 구성하는 요소 (한 로우에 1개 이상의 칼럼 존재)
functions	데이터를 처리할 때 사용할 수 있는 각종 함수를 제공하는 오브젝트
StructType, StructField	데이터에 대한 스키마 정보를 나타내는 API
GroupedData, GroupedDataSet	1. groupBy() 메서드로 인해 사용. 집계와 관련된 연산 제공

3) 코드 작성 절차

1. 스파크 세션 생성
2. 스파크 세션으로부터 데이터셋 또는 데이터 프레임 생성
3. 생성된 데이터셋 또는 데이터프레임을 이용해 데이터 처리
4. 처리된 결과 데이터를 외부 저장소에 저장
5. 스파크 세션 종료

4) 스파크 세션

스파크 세션의 사용목적은 아래와 같습니다.

1. 데이터 프레임, 데이터 셋 생성.
2. 사용자 정의 함수(UDF)를 등록.

다음은 스파크 세션의 예제입니다.

```
from pyspark.sql import SparkSession
spark = SparkSession \
    .builder \
    .appName("sample") \
    .master("local[*]") \
    .getOrCreate()
```

3. 데이터프레임

데이터 프레임은 Row라는 스파크 데이터 모델을 사용하는 wrapper 모델입니다.

RDD와의 차이점으로는 데이터 값뿐만이 아닌 스키마 정보까지 함께 다룬다는 차이가 있습니다.

1) 데이터 프레임 생성 방법

1. 외부의 데이터 소스

- DataFrameReader의 read() 메서드 사용
- DataFrameReader가 제공하는 주요 메서드는 아래와 같습니다.

Aa 메소드	≡ 의미
<u>format()</u>	1. input 데이터의 유형을 지정2. orc, libsvm, kafka, csv, jdbc, json, parquet, text, console, socket 등 사용 가능3. 써드 파티 라이브러리 존재
<u>option/options()</u>	키와 값형태로 설정 정보 지정.
<u>load()</u>	input으로 부터 실제 데이터 프레임을 생성.
<u>jdbc()</u>	데이터베이스를 input으로 사용하기위한 간편 메서드
<u>json()</u>	json을 input으로 사용하기 위한 간편 메서드(파일의 경우 각 라인 단위로 json이 구성되어야 함)
<u>orc()</u>	orc 형식 데이터 다룰수 있음
<u>parquet()</u>	parquet형식 데이터를 가져와 데이터 프레임 생성
<u>schema()</u>	사용자 정의 스키마 지정가능
<u>table()</u>	테이블 데이터를 데이터프레임으로 생성
<u>text()</u>	텍스트 파일을 읽어 데이터 프레임 생성
<u>csv()</u>	1. csv 파일을 읽어 데이터 프레임 생성2. 2.0버전부터 제공

2. 기존 RDD 및 로컬 컬렉션으로부터

- RDD와 달리 스키마정보를 함께 지정하여 생성해야 합니다.
- 스키마 지정 방법
 1. 리플렉션 = 스키마 정의를 추가하지 않고 데이터 값으로부터 알아서 스키마 정보를 추출하여 사용하는 방법입니다.
 2. 명시적 타입 지정 = 말 그대로 명시적으로 스키마를 정의하는것입니다. → (StructField, StructType 사용)
 3. 이미지 파일을 이용한 데이터 생성
 - 2.3부터 지원
 - ImageSchema.readImages를 사용하여 생성 가능합니다.
 - 이미지 파일 경로, 가로, 세로, 채널정보 등 추출 가능합니다.

2) 주요 연산 및 사용법

1. 액션 연산

Aa 연산	≡ 설명
<u>show()</u>	1. 데이터를 화면에 출력하는 연산2. 3개의 인자 존재(레코드 수, 표시할 값의 길이, 세로 출력 여부)3. 기본값은 20개 레코드수, 20바이트 길이 제한
<u>head()</u> , <u>first()</u>	데이터셋의 첫번째 ROW를 반환.
<u>take()</u>	첫 n개의 ROW 반환
<u>count()</u>	데이터셋의 로우 갯수 반환
<u>collect()</u> , <u>collectAsList()</u>	데이터셋의 모든 데이터를 컬렉션 형태로 반환(Array, List)
<u>describe()</u>	숫자형 칼럼에 대해 기초 통계값 제공(건수, 평균값, 표준편차, 최솟값, 최댓값)

2. 기본 연산

Aa 연산	≡ 설명
<u>cache()</u> , <u>persist()</u>	1. 데이터를 메모리에 저장2. [NONE, DISK_ONLY, DISK_ONLY_2, MEMORY_ONLY, MEMORY_ONLY_2, MEMORY_ONLY_SER, MEMORY_ONLY_SER_2, MEMORY_AND_DISK, MEMORY_AND_DISK_2, MEMORY_AND_DISK_SER, MEMORY_AND_DISK_SER_2, OFF_HEAP]옵션 가능3. cache는 MEMORY_AND_DISK 옵션의 persist
<u>createOrReplaceTempView()</u>	1. 데이터프레임을 테이블로 변환2. 스파크 세션인 종료되면 사라짐.
<u>explain()</u>	데이터 프레임 처리와 관련된 실행 계획 정보를 출력.

3. 비트입 트랜스포메이션 연산

Aa 연산	≡ 설명
<u>alias()</u> , <u>as()</u> .	칼럼이름에 별칭 부여
<u>isin()</u> .	칼럼의 값이 인자로 지정된 값에 포함되어 있는지 여부 확인
<u>when()</u> .	1. if~else와 같은 분기 처리 연산 수행 2. functions, Column 모두 제공(최초 사용시에는 functions의 메소드를 사용해야함)
<u>max()</u> , <u>mean()</u> .	칼럼의 최대값, 평균값 계산.
<u>collect_list()</u> , <u>collect_set()</u> .	특정 칼럼을 모아서 list 혹은 set을 반환
<u>count()</u> , <u>countDistinct()</u> .	특정 칼럼에 속한 데이터의 갯수, 중복을 제거한 데이터의 갯수
<u>sum()</u> .	특정 칼럼 값의 합계
<u>grouping()</u> , <u>grouping_id()</u> .	소계 결과에 그룹화 수준을 파악하기 위해 사용.
<u>array_contains()</u> , <u>size()</u> , <u>sort_array()</u> .	특정값이 배열에 존재하는지, 배열의 사이즈, 배열 정렬 기능
<u>explode()</u> , <u>posexplode()</u> .	1. 배열, map 데이터를 여러개의 row로 변환 2. posexplode의 경우 순서정보도 부여
<u>current_data()</u> , <u>unix_timestamp()</u> , <u>to_date()</u> .	1. current_data = 현재 시간값 2. unix_timestamp = string을 Date로 변환(yyyy-MM-dd HH:mm:ss) 3. to_date = string을 Date로 변환(yyyy-MM-dd)
<u>add_months()</u> , <u>date_add()</u> , <u>last_day()</u> .	1. 달 더하는 연산 2. 일 더하는 연산 3. 해당 달의 마지막 날짜.
<u>window()</u> .	1. DataType 칼럼을 대상으로 적용 가능 2. 일정크기의 시간을 기준으로 윈도우를 생성하여 각종 집계 연산 수행.
<u>round()</u> , <u>sqrt()</u> .	반올림, 제곱근 값 계산
<u>array()</u> .	여러 칼럼값을 배열로 만들.
<u>desc()</u> , <u>asc()</u> .	sort() 메소드와 함께 사용하는 정렬 방법 메소드
<u>desc_nulls_first</u> , <u>desc_nulls_last</u> , <u>asc_nulls_first</u> , <u>asc_nulls_last</u>	정렬시 null값 기준 정의 메소드.
<u>split()</u> , <u>length()</u> .	문자열 분리, 길이 반환 메소드
<u>rownum()</u> , <u>rank()</u> .	오라클의 partitionBy개념의 Window 범위 안에서 사용하는 메소드
<u>udf()</u> .	1. 사용자 정의 함수 2. 파이썬에서 자바 UDF사용하고 싶을 시 spark.udf.registerJavaFunction 메소드 사용
<u>select()</u> , <u>drop()</u> .	특정 칼럼을 포함/제외한 데이터 프레임 생성
<u>filter()</u> , <u>where()</u> .	특정 조건을 만족하는 데이터프레임만을 반환
<u>agg()</u> .	특정칼럼에 대해 sum, max와 같은 집계연산을 수행하기 위해 사용.
<u>apply()</u> , <u>col()</u> .	데이터프레임의 칼럼 객체 생성
<u>groupBy()</u> .	SQL문의 groupBy 연산 수행, 집계연산 수행 가능
<u>cube()</u> .	데이터의 소계를 반환(= 소계대상인 칼럼들의 모든 가능한 조합의 부분합)
<u>distinct()</u> , <u>dropDuplicates()</u> .	1. distinct의 경우 모든 컬럼값이 같을때 중복 판단 2. dropDuplicates는 중복을 제거하고자 하는 컬럼 지정 가능
<u>intersect()</u> .	두개의 데이터프레임의 교집합
<u>except()</u> .	두개의 데이터프레임의 차집합
<u>join()</u> .	1. 두 개의 데이터프레임의 join 2. inner, outer, left_outer, right_outer, leftsemi 등 join 종류 모두 지원 3. 조인 칼럼에 대해서는 중복으로 포함되지 않는 특징 존재
<u>crossjoin()</u> .	두 데이터프레임의 카테시안곱 지원
<u>na()</u> .	1. null, NaN값이 포함되는 경우 사용 2. DataFrameNaFunctions 인스턴스 반환 3. DataFrameNaFunctions가 가지고 있는 메서드 사용(drop, fill, replace 등등)
<u>orderBy()</u> .	SQL의 orderBy와 같은 동작 수행
<u>stat()</u> .	1. 특정 칼럼값의 통계수치를 제공하는 DataFrameStatFunctions 인스턴스 제공 2. corr, cov, crosstab, freqItems, sampleBy 등의 메소드 사용 가능
<u>withColumn()</u> , <u>withColumnRenamed()</u> .	새로운 칼럼 추가 혹은 이름을 변경하는 메서드
<u>repartitionByRange()</u> .	repartition시 데이터 프레임의 갯수를 기준으로 균등하게 분포(2.3부터 가능)
<u>colRegax()</u> .	정규식을 이용하여 칼럼을 선택할 수 있는 기능(2.3부터 가능)
<u>unionByName()</u> .	1. union할 시 데이터프레임의 칼럼 이름을 기준으로 수행 2. 그냥 union은 데이터프레임의 데이터 순서로 수행되기 때문에 데이터가 엉킬수 있음.
<u>to_json</u> , <u>from_json</u>	칼럼 수준에서 데이터를 json으로 저장 및 가져오는 기능 제공.

3) 데이터 저장

DataFrameWriter.write 사용하여 데이터프레임 데이터를 저장합니다.

DataFrameWriter의 주요 메소드는 아래와 같습니다.

Aa 메소드	≡ 의미
<u>save</u>	데이터를 저장하는 메소드
<u>format</u>	저장하는 데이터의 format을 지정.
<u>partitionBy</u>	특정 칼럼값을 기준으로 파티션 설정
<u>options</u>	데이터 저장시 추가 설정 시 사용
<u>mode</u>	저장 모드 설정(Append, Overwrite, ErrorExists, Ignore)
<u>saveAsTable</u>	테이블 형태로 저장 및 영구 저장 가능

4) Pandas 연동

PyArrow 라이브러리를 사용하여 스파크의 데이터프레임과 파이썬의 Pandas의 상호 변환이 가능합니다.

스파크 2.3버전부터 가능합니다. 추가로, spark.sql.execution.arrow.enabled 옵션을 true로 하여야 합니다.

4. 데이터 셋

데이터 셋은 Row가 아닌 개발자 지정 클래스를 Type으로 쓰는 wrapper 모델입니다.

데이터셋의 장단점은 아래와 같습니다.

- 장점 = 지정 클래스를 사용함으로 인해 **컴파일 시 타입오류 검증이 가능합니다**.
- 단점 = 사용자 정의 타입 클래스를 사용하기 때문에 Spark SQL 최적화가 안될 수 있습니다.

1) 데이터 셋 생성

- 생성 방법 = 자바 객체, 기존 RDD, 데이터 프레임, 외부 source 가 있습니다.
- 생성 시 인코더 지정(필수) = 성능 최적화를 위해 기존 오브젝트를 스파크 내부 최적화 포맷으로 변환해야 하는데 그때 사용합니다. (데이터셋은 내부적으로 InternalRow클래스를 사용합니다)
- 스칼라의 경우, 기본타입의 경우에는 import를 통하여 암묵적 인코더 사용 가능합니다.
- 자바의 경우, 데이터 프레임을 만든 후 이것을 데이터 셋으로 변환하여 사용해야 합니다.
- 데이터 프레임으로부터 생성시에는 type을 지정해야하기 때문에 as() 메서드를 통하여 생성할 수 있습니다.
- range() 메서드를 통하여 간단한 샘플데이터 또한 생성 가능합니다.

2) 타입 트랜스포메이션 연산

Aa 연산	≡ 설명
<u>select()</u>	1. 데이터 프레임의 select 메서드와 동일2. 단, as() 메서드를 통해 type을 지정해야함.
<u>as()</u>	1. 데이터 셋에 별칭 부여2. column에 부여하는 것이 아닌 데이터 셋에 타입을 부여.
<u>distinct()</u>	중복을 제외한 요소만으로 데이터셋 반환
<u>dropDuplicates()</u>	distinct() 메서드에서 칼럼을 지정할 수 있는 기능 추가.
<u>filter()</u>	사용자 정의 조건을 만족하는 데이터만으로 구성된 데이터 셋 반환.
<u>map()</u> , <u>flatMap()</u>	1. RDD의 map(), flatMap() 메서드와 동일2. 단 데이터 셋의 타입을 인자로 사용 가능3. 자바의 경우 인코더 지정 필수
<u>groupByKey()</u>	1. RDD의 groupByKey와 기능 동일2. KeyValueGroupedDataset 클래스 반환
<u>agg()</u>	1. 데이터 프레임에서 본 agg() 메서드와 동일2. 단, 사용하는 칼럼은 반드시 TypedColumn 클래스여야 한다3. 사용할 수 있는 집계 연산 = avg(), count(), sum(), sumLong()
<u>mapValues()</u> , <u>reduceGroups()</u>	1. KeyValueGroupedDataset클래스의 메서드2. 맵연산, 리듀스 연산을 수행.

5. QueryExecution

QueryExecution은 Spark SQL 사용시 최적화 과정에서 일어나는 일들을 확인할 수 있는 API입니다.

Spark Sql의 Query 실행은 아래와 같은 순으로 진행되고 최적화됩니다.

# 순서	Plan 종류	확인 메소드 종류
1	<u>LogicalPlan</u>	QueryExecution.logical
2	<u>SessionState의 Analyzer가 적용된 LogicalPlan</u>	QueryExecution.analyzed
3	<u>SessionState의 Optimizer가 적용된 LogicalPlan</u>	QueryExecution.optimizedPlan
4	<u>SessionState의 SparkPlanner가 적용된 SparkPlan</u>	QueryExecution.sparkPlanner
5	<u>SparkPlan에 추가적인 최적화 과정 적용하여 SparkPlan</u>	QueryExecution.executedPlan

describe() 숫자형 칼럼에 대해 기초 통계값 제공(건수, 평균값, 표준편차, 최솟값, 최댓값)

문제269. 사원 테이블을 출력하는데 맨 위에 한 행만 출력하시오 !
(힌트: head(), first() : 데이터셋의 첫번째 ROW를 반환)

```
scala> sql("""select *
from emp""").head()
```

데이터 분석을 위한 큰그림에서의 하둡과 스칼라의 역할은 ?

1. 빅데이터를 저장하기 위한 테이블 생성
2. 테이블에서 데이터를 전처리하고 필요한 데이터 검색
3. 검색한 데이터를 csv 파일로 저장해서 파이썬, R 로 보낸다.
4. 파이썬이나 R 에서 가져온 csv 로 데이터 시각화

문제270. 위에서 출력한 직업과 직업별 토달월급의 결과를 csv 파일로 저장하시오 !

```
scala> sql("""select job, sum(sal)
from emp
group by job""").coalesce(1).write.option("header","true").option("sep",",").mode("overwrite").csv("/home/scott/dd")
```

설명: .write.option("header","true") : 컬럼명 나오게 해라 ~
.option("sep",",") : csv 파일 형태로 만들어라 ~~
.mode("overwrite").csv("/home/scott/dd") : /home/scott/dd 라는 폴더안에 생성해라~

아래의 결과를 복사해서 윈도우에서 notepad 를 열고 붙여넣은 다음 job.csv 로 저장합니다. 저장할 때 파일형식은 모든파일로 선택해야 합니다.
저장위치는 c 드라이브 밑에 data 밑에 저장합니다.

```
job,sum(sal)
ANALYST,6000
SALESMAN,5600
CLERK,4150
MANAGER,8275
PRESIDENT,500
```

윈도우에서 spyder 를 켜세요 ~~ 아래의 내용을 코딩하세요 !

```
import pandas as pd
emp = pd.read_csv("c:\\data\\job.csv") # csv 파일 불러와서 emp 데이터프레임 생성
print(emp.columns) # 컬럼명 확인
```

문제271. job.csv 의 내용을 막대그래프로 시각화 하시오 ! (spyder)

```
import pandas as pd
emp = pd.read_csv("c:\\data\\job.csv")
result= emp['sum(sal)']
result.index=emp['job']
result.plot(kind='bar', color='red')
```

문제272.(점심시간 문제) 스칼라를 이용해서 데이터 전처리 및 검색을 하고 파이썬에서 데이터 시각화하는 큰그림을 구현하시오 !

부서번호, 부서번호별 토달월급을 막대 그래프로 그리시오 (색깔은 다른색깔로 정하고 캡처해서 올리세요 ~)

1. 스칼라에서는 부서번호,부서번호별 토달월급을 csv 파일로 뺀고
2. 그 csv 파일을 윈도우의 파이썬에서 막대 그래프로 시각화 하세요 ~

하둡 설치, 하이브설치, 스칼라 설치, 오라클 설치 → 데이터 엔지니어(회사에 해당 직무가 따로 있음. 설치법을 알면 좋지만, 이것 때문에 프로그램을 다루는 법을 잘 못배우는 일은 없어야 함)

하둡과 하이브와 스칼라는 빅데이터를 저장하는 테이블 생성 위주로 현업에 활용을 하니까 테이블 생성 해서 데이터를 저장하고 파이썬이나 R 로 시각화 또는 분석을 한다. → 데이터 분석가

■ 지금 처럼 emp 테이블은 데이터가 작으니까 그냥 결과를 복사해서 윈도우의 메모장에다가 붙여넣었지만 현업에서는 대용량 데이터이므로 이렇게 할 수 가 없습니다. csv 파일을 따로 다운로드 받게 해야 합니다.

이렇게 하는 방법은 다음과 같습니다.

- 1. mobaxterm 프로그램을 다운로드 받아서 설치합니다.
- 2. mobaxterm 으로 리눅스 centos 에 접속을 합니다.
- 3. mobaxterm 에서 파일을 다운로드 받습니다.

문제273. mobaxterm 을 이용해서 sample.csv 를 /home/scott 밑에 올리시오

설명: 현업에서는 오라클 vm 으로 가상환경을 만드는게 아니라 실제 리눅스 서버를 사용하므로 내자리의 노트북에 있는 파일을 별도의 컴퓨터로 구성되어 있는 리눅스 서버에 올리려면 mobaxterm 과 같은 툴을 이용해서 올립니다. 리눅스 서버의 대용량 데이터를 내 노트북으로 내릴때도 mobaxterm 을 이용해서 편하게 합니다.

■ 리눅스의 설치되어있는 아나콘다의 spyder 를 내 자리의 컴퓨터(내 노트북) 의 화면에 띄우게 하는 방법 (리눅스에 아나콘다 & spyder 설치 되어 있어야 함)

<https://noooop.tistory.com/entry/ssh-환경에서-GUI-사용하기-X11-forwardingX11-포워딩>

루트로 접속해서

```
cd /etc/ssh/
```

```
vi sshd_config
X11Forwarding yes    <-- 맨 아래에 추가
```

```
yum install xclock
```

```
xclock
```

■ putty 에서 xclock 여는 방법 (카페 게시글 412번)

1. 도스창을 열고 ipconfig 해서 자기 자리의 ip 주소를 확인합니다.

```
ipconfig
이더넷 어댑터 이더넷:

연결별 DNS 접미사. . . . . :
링크-로컬 IPv6 주소 . . . . . : fe80::2dcb:2eaa:e5c0:16c0%4
IPv4 주소 . . . . . : 192.168.19.31
서브넷 마스크 . . . . . : 255.255.0.0
기본 게이트웨이 . . . . . : 192.168.0.1
```

2. 확인한 아이피 주소를 가지고 리눅스에서 아래와 같이 export 합니다.
(putty 에서 scott 에서 수행합니다.)

```
$ export DISPLAY=192.168.19.31:0.0
$ xhost
```

3. xclock 을 실행합니다. (mobaxterm 이 실행되어져 있어야합니다.)

```
$ xclock
```

설명: 위에서 수행할 mobaxterm 경고창이 안나오게 하려면 ?

mobaxterm → settings → configuration → x11 탭 → x11 remote access 을 full 로 변경

설명: mobaxterm 은 파일 전송하고 내려받는거 위주로 사용하고 대부분 작업을 putty 에서 수행하면 됩니다.

■ 지금까지 설정한 리눅스 환경에 아나콘다 설치하기!

복제한 환경에는 아직 아나콘다 설치 안했고 복제 환경에는 하둡, 하이브, 스칼라, putty 설정이 다 되었으므로 아나콘다만 설치하면 됩니다.

■ 실수로 다 꺼버리거나 putty 를 다시 접속해야하는 상황이면

1. putty 로 다시 scott 으로 접속하고
2. py389 를 activate 시킵니다.

```
$ conda activate py389
```

3. DISPLAY 를 export 시키고 시계가 잘 열리는지 확인합니다.

```
$ export DISPLAY=192.168.19.31:0.0
$ xclock
$ spyder
```

문제274. emp2.csv 를 리눅스의 spyder 에서 판다스로 불러오고 프린트하시오 !

```
import pandas as pd
emp=pd.read_csv("/home/scott/emp2.csv", header=None)
print(emp)
```

문제275. emp2.csv 에서 emp 로 로드하고 사원 테이블의 월급을 바로 막대그래프로 시각화 하시오 !

```
import pandas as pd
emp=pd.read_csv("/home/scott/emp2.csv", header=None)
result = emp[5]
result.index=emp[1]
result.plot(kind='bar', color='red')
```

문제276.(오늘의 마지막 문제) 리눅스 하둡 시험문제 제출 포맷

현업에서 데이터 분석가들이 분석 요청이 들어오면 하는 일

Case.csv 를 스칼라로 로드해서 테이블 생성한후에 지역, 지역별 코로나 감염자 수를 count 하고 결과를 가지고 파이썬에서 막대그래프로 시각화 하시오 ! (막대 그래프 결과를 색깔을 달리해서 올리세요 ~)

```
select province , sum(confirmed)
from case2
group by province;
```

[21.01.13 리눅스 강의 Day 12 //하둡]

■ 리눅스와 하둡 수업 복습

뭘 하기 위해서 리눅스와 하둡 수업을 배우는가?

빅데이터(G,T 급) 를 분석 및 시각화 하기 위해!

e.x) 코로나 테라급 데이터 (csv, text)를 가지고 코로나의 확산을 막기 위해 데이터 분석 질문을 던진다고 한다면?

기존의 오라클에 테라급 데이터를 넣어서 쿼리를 수행한다면 비용(돈) 과 시간이 너무 많이 필요함.

때문에 무료인 하둡, 스칼라, maria db 등을 이용해서 빅데이터를 저장하고 빅데이터를 빠르게 검색하여 원하는 정보를 얻어내려면 리눅스와 하둡을 사용해야 한다.

small data라면 오라클이나 mySQL을 사용해서 분석을 해도 된다.

하지만 데이터가 많을 수록 더 설득력있고 의미있는 정보를 얻어낼 수가 있어서 빅데이터를 저렴한 비용으로 저장하고 관리할 수 있는 리눅스 서버와 하둡 기술이 나오면서 빅데이터에 대한 관심이 높아졌고 많은 기업들이 빅데이터를 이용해서 사업을 많이 하게 되었다.

■ 하둡을 사용하면 빅데이터 처리가 빨라지는 이유?

```
$ hadoop fs -put emp2.csv emp2.csv
```



리눅스에 있는 emp.csv

하둡 파일 시스템에 올릴 emp2.csv



/home/scott/emp2.csv



/user/scott/emp2.csv

인셉션으로 비교해보는 하둡

윈도우 —————>> 리눅스 가상 머신 —————> 하둡 파일 시스템

현실세계

꿈속

꿈속의 꿈

1. 아래의 명령어는 local을 사용했기 때문에 리눅스 서버의 /home/scott 밑에 있는 emp2.csv의 데이터를 hive의 emp에 로드해라~는 뜻

```
hive> LOAD DATA LOCAL INPATH '/home/scott/emp2.csv'
      INTO TABLE emp;

hive> select * from emp1; # 현대의 컴퓨터만 작동해서 데이터를 검색한다.
```

불러오는 위치를 보면
home/scott/emp2.csv --> 리눅스에 있는 emp2.csv

2. 아래의 명령어는 local을 사용하지 않았기 때문에 하둡 파일 시스템에 올린 emp2.csv를 emp테이블에 입력해라~는 뜻.

```
hive> LOAD DATA INPATH '/user/scott/emp2.csv'
      overwrite INTO TABLE emp;

hive> select * from emp; # 30대의 컴퓨터가 동시에 다 움직이면서 데이터를 검색한다.
```

불러오는 위치를 보면
user/scott/emp2.csv --> 하둡 시스템 파일에 있는 emp2.csv

하둡을 이용하면 컴퓨터 한대의 자원만을 활용하는 것이 아니라 여러대의 컴퓨터를 묶어서 마치 하나의 서버처럼 보이게 해서 여러대의 컴퓨터 자원을 이용할 수 있다라는 장점이 있다.

일반 컴퓨터를 예로 들어볼때

하둡 이용하지 않을 경우 → 서버 1대 (64기가 메모리=최신, 최고사양 컴퓨터 1대)

하둡을 이용할 경우 → 30대의 컴퓨터 (8기가 x 30대 = 250 기가)

■ 하둡 분산파일 시스템 명령어

■ 1. ls 명령어



"지정된 디렉토리에 있는 파일의 정보를 출력 "

예:

```
$ cd
$ ls -l emp2.csv
$ hadoop fs -put /home/scott/emp2.csv /user/scott/emp3.csv
$ hadoop fs -ls /user/scott/emp3.csv
```

- 하둡 파일 시스템에 올린 파일들을 웹페이지로 확인하는 방법

```
$ export DISPLAY=192.168.19.31:0.0
$ xclock
$ firefox centos:50070
```

■ 2. lsr 명령어



현재 디렉토리 뿐만 아니라 하위 디렉토리까지 조회하는 명령어

```
$ hadoop fs -lsr /user/
```

문제277. 하둡 파일 시스템의 / (루트) 밑에 있는 모든 하위 디렉토리를 조회하시오 !

답: \$ hadoop fs -lsr /

3. du 명령어



파일의 용량을 확인하는 명령어

```
$hadoop fs -du
```

4. dus명령어



파일의 전체 합계 용량을 확인하는 명령어

```
$ hadoop fs -dus
```

5. cat 명령어

6. text



지정된 파일의 내용을 화면에 출력하는 명령어

```
$ hadoop fs -text /user/scott/emp3.csv
```

문제 278. 하둡 파일 시스템의 리눅스 /home/scott 밑에 있는 Case.csv를 /user/scott 밑에 case3.csv로 올리시오.

```
$ hadoop fs -put /home/scott/Case.csv /user/scott/case.csv
```

문제 279. 하둡 파일 시스템에 올린 case3.csv의 내용을 조회하시오.

```
$ hadoop fs -cat /user/scott/case3.csv
```

7. mkdir 명령어



디렉토리를 생성하는 명령어

```
$ hadoop fs -mkdir /user/scott/test
```

문제 280. 위에서 만든 test 디렉토리가 잘 생성되었는지 확인하시오!

```
$ hadoop fs -ls
```

문제 281. 앞으로는 test 디렉토리에 데이터를 올리고 관리를 하기 위해 리눅스의 /home/scott 밑에 있는 emp2.csv를 /user/scott/test/ 에 emp2.csv 로 올리시오.

```
$ hadoop fs -put /home/scott/emp2.csv /user/scott/test/emp2.csv
```

8. put 명령어



하둡 파일 시스템에 파일을 올리는 명령어

```
$ hadoop fs -put /옮길파일위치/옮길파일명. 확장자 /옮겨질파일위치/옮길파일명. 확장자
```

```
$ hadoop fs -put /home/scott/emp2.csv /user/scott/test/emp2.csv
```

9. get 명령어



하둡 파일 시스템에 올린 파일을 리눅스 디렉토리로 내리는 명령어

```
$ cd
$ rm emp2.csv
$ ls -l emp2.csv
---로 일단 리눅스에서 있는 파일 한번 지움
$ hadoop fs -get /user/scott/test/emp3.csv /home/scott/emp2.csv
$ ls -l emp2.csv
```

10. rm명령어



하둡 파일 시스템에 있는 파일을 지우는 명령어

```
$ hadoop fs -lsr /user/  
$ hadoop fs -rm /user/scott/emp3.csv
```

문제 282. 리눅스에 /home/scott 밑에 있는 emp2.csv를 하둡에 /user/scott 밑에 emp3.csv로 올리시오.

```
$ hadoop fs -put /home/scott/emp2.csv /user/scott/emp3.csv
```

11. rmr 명령어



하둡 파일 시스템의 디렉토리를 삭제하는 명령어

```
$ hadoop fs -rmr /user/scott/test
```

문제 283. 코로나 데이터중 감염자 정보중에 성별 데이터가 있는 csv 파일을 하둡 파일 시스템에 /user/scott 밑에 올리시오

```
$ hadoop fs -put /home/scott/TimeGende.csv /user/scott/timegender.csv
```

12. grep 명령어



파일에서 특정 문자의 행의 데이터를 검색하는 명령어

```
$ hadoop fs -put /home/scott/emp2.csv /user/scott/emp7.csv  
$ hadoop fs -cat /user/scott/emp7.csv | grep -i 'salesman'
```

문제 284. 하둡 파일 시스템에 올린 코로나 데이터인 timegender.csv 파일을 cat으로 여시오.

```
$ hadoop fs -cat /user/scott/timegender.csv
```

문제 285. 하둡 파일시스템에 올린 코로나 데이터인 timegender.csv 파일에서 남자가 모두 몇명인지 확인하시오.

```
$ hadoop fs -cat /user/scott/timegender.csv | grep -i 'male' | wc -l
```


문제 286. 판다스를 이용해서 원형 그래프를 그리시오.

1) 리눅스의 spyder를 켜다

리눅스 /home/scott/ 밑에 있는 emp2.csv를 판다스로 로드해서 emp2.csv를 emp데이터 프레임으로 만들어서 프린트 하시오.

```
import pandas as pd

emp=pd.read_csv("/home/scott/emp2.csv", header=None)
print(emp)
```

2) 직업과 직업별 인원수를 출력한다.

```
import pandas as pd

emp=pd.read_csv("/home/scott/emp2.csv", header=None)
result=emp[2].value_counts()
print(result[0])
```

3) 직업과 직업별 인원수로 막대 그래프를 그리시오

```
import pandas as pd

emp=pd.read_csv("/home/scott/emp2.csv", header=None)
result=emp[2].value_counts()
result.plot(kind='bar')
```

설명: kind 옵션: bar : 막대 그래프 , pie : 원형 그래프 , line : 선 그래프
'pie' 원형 그래프 'line' 선그래프
'bar' 수직 막대그래프 'kde' 커널 밀도 그래프
'barh' 수평 막대그래프 'area' 면적 그래프
'his' 히스토그램 그래프 'scatter' 산점도 그래프
'box' 박스(사분위수) 그래프 'hexbin' 고밀도 산점도 그래프

문제 286. 판다스로 직업, 직업별 토탈월급을 출력하시오. emp2.csv에 컬럼명을 맨 위에 vi 편집기로 넣고 하시오~!

```
import pandas as pd
emp=pd.read_csv("/home/scott/emp2.csv")
result=emp.groupby('job')['sal'].sum()
print(result)
```

13. awk 명령어

특정 컬럼을 검색하는 명령어

하둡 파일 시스템에 올린 emp3.csv 파일에서 SCOTT 이 사원의 이름과 월급을 조회하시오,

```
$ hadoop fs -cat /user/scott/emp2.csv | grep -i 'scott' | awk -F "," '{print $2,$6}'
```

-F "," 넣어준 이유?

데이터가 콤마(,) 로 구분되어 있으면 awk -F 옵션을 써서 "," 로 구분되어 있다는 것을 알려줘야 한다.

scott/2000/salesman → awk -F ","

scott,2000,salesman → awk -F "/"

14. count 명령어

지정된 디렉토리에 파일의 갯수를 확인하는 명령어

```
$ hadoop fs lsr user/scott
로 실제로 몇개 있는지 확인

$ hadoop fs -count /user/scott/
를 입력해 위에서 확인한 결과와 같은지 확인
```

15. 하둡 파일 시스템 명령어 메뉴얼 보는 방법

```
$ hadoop fs -help
```

타조 설치

🌟 tajo 를 이용해서 데이터 검색하기 (OLTP 용) - 오라클 대응

hive 보다 훨씬 빠른 noSQL이다. 자체 설계한 처리 엔진을 통해 HIVE 보다 훨씬 빠른 처리시간을 보여주는 것이 특징.

- tajo 설치

1. 타조 설치 파일을 /home/scott 밑에 가져다 둔다.

```
cp /media/sf_data/tajo-0.11.1.tar.gz /home/scott/
```

2. 압축을 해제한다.

```
$ tar -xvzf tajo-0.11.1.tar.gz
```

3. 압축 풀린 디렉토리 이름을 간단하게 tajo 로 변경한다.

```
$ mv tajo-0.11.1-desktop-r1 tajo
```

4. 타조 홈디렉토리의 bin 디렉토리로 가서 [configure.sh](#) 파일을 수정합니다. [configure.sh](#) 파일 수행시 JAVA_HOME 을 입력하라고 나오면 그냥 엔터!advanced 옵션을 지정하겠냐고 물어보면 N 입력!

```
$ cd tajo
$ cd tajo (똑같은 폴더가 하나 더있어서 또 들어가줘야 함)
/home/scott/tajo/tajo/bin
(base) [scott@centos bin]$ ./configure.sh
```

5. 타조를 시작 시킨다.

```
$ sh bin/startup.sh
```

```
(base) [scott@centos bin]$ sh startup.sh
starting master, logging to /home/scott/tajo/tajo/bin/../logs/tajo-scott-master-centos.out
Java HotSpot(TM) Client VM warning: You have loaded library /home/scott/tajo/tajo/hadoop/lib/native/libhadoop.so which might ha
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
Tajo master started
```

```
starting worker, logging to /home/scott/tajo/tajo/bin/../logs/tajo-scott-worker-centos.out
Tajo worker started
Tajo master web UI: http://centos:26080
Tajo Client Service: centos:26002
(base) [scott@centos bin]$
```

6. 타조에 접속 한다.

```
/home/scott/tajo/tajo/bin 에서
$ ./tsql
or
$ sh tsql
```

```
(base) [scott@centos bin]$ ./tsql

welcome to

  _ _ _ _ _
 / _ _ _ _ \ | _ _ _ _ \
 / _ _ _ _ \ | _ _ _ _ \
 / _ _ _ _ \ | _ _ _ _ \
 / _ _ _ _ \ | _ _ _ _ \ 0.11.1-p1

Try \? for help.
default>
```

10. 타조에서 데이터 베이스를 생성하시오.

```
default> create database orcl;
OK 라고 나옴
```

11. orcl database에 접속

```
default> \c orcl;
아래와 같이 나옴
You are now connected to database "orcl" as user "scott".
orcl> 로 입력창이 바뀜
```

12. 타조에서 emp 테이블 생성하기

```
orcl> CREATE EXTERNAL TABLE emp (
> empno INT ,
> ename text ,
> job TEXT ,
> mgrno INT,
> hiredate text,
> sal INT,
> comm INT,
> deptno int
> ) USING CSV WITH ('text.delimiter','=',') LOCATION 'file:///home/scott/emp2.csv';
OK
```

13. emp의 데이터 잘 조회 되는지 확인

```
orcl> select * from emp;
empno, ename, job, mgrno, hiredate, sal, comm, deptno
-----
7839, KING, PRESIDENT, 0, 1981-11-17, 5000, 0, 10
7698, BLAKE, MANAGER, 7839, 1981-05-01, 2850, 0, 30
7782, CLARK, MANAGER, 7839, 1981-05-09, 2450, 0, 10
7566, JONES, MANAGER, 7839, 1981-04-01, 2975, 0, 20
7654, MARTIN, SALESMAN, 7698, 1981-09-10, 1250, 1400, 30
7499, ALLEN, SALESMAN, 7698, 1981-02-11, 1600, 300, 30
7844, TURNER, SALESMAN, 7698, 1981-08-21, 1500, 0, 30
```

```
7900, JAMES, CLERK, 7698, 1981-12-11, 950, 0, 30
7521, WARD, SALESMAN, 7698, 1981-02-23, 1250, 500, 30
7902, FORD, ANALYST, 7566, 1981-12-11, 3000, 0, 20
7369, SMITH, CLERK, 7902, 1980-12-09, 800, 0, 20
7788, SCOTT, ANALYST, 7566, 1982-12-22, 3000, 0, 20
7876, ADAMS, CLERK, 7788, 1983-01-15, 1100, 0, 20
7934, MILLER, CLERK, 7782, 1982-01-11, 1300, 0, 10
(14 rows, 0.189 sec, 0 B selected)
```

문제 288. 월급이 1000에서 3000 사이인 직원들의 이름과 월급을 출력하시오.

```
oracle>select ename, sal
>from emp
>where sal between 1000 and 3000;
```

마리아 db 설치

하둡과 상관없이 설치 및 실행 가능

OLTP환경 vs DW 환경

Aa OLTP	DW
Oracle(유료)	하둡과 하이브(무료)
mysql(유료)	스칼라(무료)
maria db(무료)	

maria db는 MySQL을 만든 개발자가 회사를 나와서 만든 무료 Database

다음의 링크를 참조함(<https://suwoni-codelab.com/linux/2017/05/24/Linux-CentOS-MariaDB/>)

실행 이전에 scott에서 sudo 명령어 실행을 위해 파일을 만들어줘야함(<https://myjamong.tistory.com/3>)
를 해주었으나 2번째 단계부터 root에서 해야 한다고 해서 결국엔 root에서 실행

1. MariaDB 설치 (root 계정에서 실행!)

- MariaDB를 설치 하기 위해서 아래의 명령어를 실행합니다.

```
$ curl -ss https://downloads.mariadb.com/MariaDB/mariadb_repo_setup | sudo bash
```

```

1. root@localhost:/etc/yum.repos.d (ssh)

[root@localhost ~]# cd /etc/
Display all 151 possibilities? (y or n)
[root@localhost ~]# cd /etc/yum
yum/          yum.repos.d/
[root@localhost ~]# cd /etc/yum
yum/          yum.repos.d/
[root@localhost ~]# cd /etc/yum.repos.d/
[root@localhost yum.repos.d]# ls
CentOS-Base.repo      CentOS-Media.repo    CentOS-fasttrack.repo
CentOS-CR.repo        CentOS-Sources.repo
CentOS-Debuginfo.repo CentOS-Vault.repo
[root@localhost yum.repos.d]# curl -sS https://downloads.mariadb.com/MariaDB/mariadb_repo_setup | sudo bash
[INFO] Repository file successfully written to /etc/yum.repos.d/mariadb.repo.
[INFO] Adding trusted package signing keys...
[INFO] Successfully added trusted package signing keys.
[root@localhost yum.repos.d]# ls
CentOS-Base.repo      CentOS-Media.repo    CentOS-fasttrack.repo
CentOS-CR.repo        CentOS-Sources.repo  mariadb.repo
CentOS-Debuginfo.repo CentOS-Vault.repo
[root@localhost yum.repos.d]#

```

- 명령을 실행한 후 콘솔 화면을 보시게되면 /etc/yum/yum.repos.d/ 디렉토리를 보시게되면 mariadb.repo라는 파일이 생성이 됩니다.
- mariadb.repo 파일을 vi에디터로 열어보시면 MariaDB설치를 위한 repository정보가 포함되어 있습니다.
- mariadb.repo 파일이 생성 됨으로써 yum 명령어를 통해 최신의 MariaDB를 설치하실 수 있습니다.
- 이제 yum 명령어를 통해 MariaDB를 Install 합니다.

```
$ yum install MariaDB-server
```

```

1. root@localhost:/etc/yum.repos.d (ssh)

Verifying : MariaDB-client-10.2.6-1.el7.centos.x86_64 2/9
Verifying : boost-program-options-1.53.0-26.el7.x86_64 3/9
Verifying : galera-25.3.20-1.rhel7.el7.centos.x86_64 4/9
Verifying : MariaDB-common-10.2.6-1.el7.centos.x86_64 5/9
Verifying : MariaDB-server-10.2.6-1.el7.centos.x86_64 6/9
Verifying : 1:mariadb-5.5.52-1.el7.x86_64 7/9
Verifying : 1:mariadb-server-5.5.52-1.el7.x86_64 8/9
Verifying : 1:mariadb-libs-5.5.52-1.el7.x86_64 9/9

Installed:
MariaDB-client.x86_64 0:10.2.6-1.el7.centos
MariaDB-compat.x86_64 0:10.2.6-1.el7.centos
MariaDB-server.x86_64 0:10.2.6-1.el7.centos

Dependency Installed:
MariaDB-common.x86_64 0:10.2.6-1.el7.centos
boost-program-options.x86_64 0:1.53.0-26.el7
galera.x86_64 0:25.3.20-1.rhel7.el7.centos

Replaced:
mariadb.x86_64 1:5.5.52-1.el7 mariadb-libs.x86_64 1:5.5.52-1.el7
mariadb-server.x86_64 1:5.5.52-1.el7

Complete!
[root@localhost yum.repos.d]#

```

2. MariaDB 설정

- MariaDB 서비스를 부팅시 자동으로 실행되게 설정을 변경합니다.
- ```
$ systemctl enable mariadb
```
- MariaDB를 시작합니다.
- ```
$ systemctl start mariadb
```
- MariaDB의 root암호 및 기본 보안 설정을 하기위해 아래의 명령어를 실행합니다.
- ```
$ mysql_secure_installation
```

```
1. root@localhost:/etc/yum.repos.d (ssh)
[root@localhost yum.repos.d]# systemctl enable mariadb
[root@localhost yum.repos.d]# systemctl start mariadb
[root@localhost yum.repos.d]# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
... Success!
```

- 여러 질문이 나오는데요.
- 처음은 root 패스워드 설정하겠느냐 Y를 누르면 설정하실 수 있습니다.
- 그 다음 질문은 anonymous users 를 삭제하겠느냐는 질문입니다.
- 그 다음은 원격지에서 root로그인을 허용하겠느냐는 질문입니다.
- 기본으로 누구든지 access할수 있는 Test db 를 삭제하겠느냐는 질문입니다.
- 마지막 질문은 설정한 권한 모두 리로드해서 적용하겠느냐는 질문입니다.

```
1. root@localhost:/etc/yum.repos.d (ssh)

By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] Y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] Y
... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] Y
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

- 보안상 기본포트를 변경을 하려고 합니다.
- Selinux의 보안 정책상 특정 port 이외에는 변경을 막고 있는데요, 확인작업은 아래 명령어를 실행합니다.

```
$ semanage port -l | grep mysqld_port_t
```

```
1. root@localhost:~ (ssh)
[root@localhost ~]# semanage port -l | grep mysqld_port_t
mysqld_port_t tcp 1186, 3306, 63132-63164
[root@localhost ~]#
```

- 위 항목의 port이외에는 Selinux 가 포트변경을 막게 됩니다.

▼ If necessary ( 추가 포트를 설정하고 싶다면)

- (If necessary) 위 포트 이외에 다른 포트를 지정하기를 원하면 아래의 명령어를 실행합니다.
- 저는 3456 포트를 지정하였습니다. 포트를 확인하는 명령어를 다시 실행하여 정상적으로 지정되었음을 확인할 수 있습니다.

```
$ semanage port -a -t mysqld_port_t -p tcp 3456
```

```
1. root@localhost:~ (ssh)
[root@localhost ~]# semanage port -l | grep mysqld_port_t
mysqld_port_t tcp 1186, 3306, 63132-63164
[root@localhost ~]# semanage port -a -t mysqld_port_t -p tcp 3456
[root@localhost ~]# semanage port -l | grep mysqld_port_t
mysqld_port_t tcp 3456, 1186, 3306, 63132-63164
[root@localhost ~]#
```

- 이제 MariaDB 에서 port를 변경하기 위하여 /etc/my.cnf.d/server.cnf파일을 에디터로 오픈하고,
- [mysqld] 항목 아래에 변경할 port를 입력하고 저장합니다.

```
$ vi /etc/my.cnf.d/server.cnf
```

```
1. root@localhost:/etc/yum.repos.d (ssh)
#
These groups are read by MariaDB server.
Use it for options that only the server (but not clients) should see
#
See the examples of server my.cnf files in /usr/share/mysql/
#
this is read by the standalone daemon and embedded servers
[server]
#
this is only for the mysqld standalone daemon
[mysqld]
port = 3456
#
* Galera-related settings
#
[galera]
Mandatory settings
#wsrep_on=ON
#wsrep_provider=
#wsrep_cluster_address=
#binlog_format=row
#default_storage_engine=InnoDB
#innodb_autoinc_lock_mode=2
```

- MariaDB를 restart 합니다.

```
$ systemctl restart mariadb
```

- 설정한 포트로 방화벽 포트를 오픈 합니다.

```
$ firewall-cmd --permanent --add-port=3456/tcp
```

- firewall데몬을 리로드 해주어야 적용이 됩니다.

```
$ firewall-cmd --reload
```

### 3. MariaDB계정 생성

- MariaDB의 DB를 생성하기 위하여 콘솔창에서 아래의 명령어를 입력하여 MariaDB로 접속합니다.

```
$ mysql -u root -p
```

```
1. root@localhost:~ (ssh)
[root@localhost ~]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.2.6-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

- 계정을 생성합니다

```
create user '아이디'@'%' identified by '패스워드';
```

- 권한을 부여합니다

```
grant all privileges on *.* to '생성한아이디'@'%' identified by '패스워드';
flush privileges;
```

- 아이디 생성과 권한 부여가 완료되었습니다.

- MySql 접속 프로그램으로 접속한 모습입니다.



- MariaDB랑 Mysql은 거의(?) 똑같이 때문에 Mysql사이트에서 Workbench프로그램만 다운로드후 사용하실 수 있습니다.  
<https://dev.mysql.com/downloads/workbench/>

## 4. 데이터 베이스 생성

```
MariaDB [(none)]> create database orcl;

결과
Query OK, 1 row affected (0.001 sec)
```

## 5. 생성된 데이터 베이스로 접속

```
MariaDB [(none)]> use orcl;

결과
Database changed
```

## 6. maria db에서 emp 테이블 생성하기

아래의 스크립트를 maria db에 붙여넣는다.

### ▼ script

```
Create table emp
(empno int(4) not null,
Ename varchar(10),
Job varchar(9),
Mgr int(4),
Hiredate date,
Sal int(7),
Comm int(7),
Deptno int(4));

create table dept
(deptno int(2),
dname varchar(20),
loc varchar(20));

INSERT INTO dept VALUES (10,'ACCOUNTING','NEW YORK');
INSERT INTO dept VALUES (20,'RESEARCH','DALLAS');
INSERT INTO dept VALUES (30,'SALES','CHICAGO');
INSERT INTO dept VALUES (40,'OPERATIONS','BOSTON');

INSERT INTO emp VALUES (7839,'KING','PRESIDENT',NULL,'81-11-17',5000,NULL,10);
INSERT INTO emp VALUES (7698,'BLAKE','MANAGER',7839,'81-05-01',2850,NULL,30);
INSERT INTO emp VALUES (7782,'CLARK','MANAGER',7839,'81-05-09',2450,NULL,10);
INSERT INTO emp VALUES (7566,'JONES','MANAGER',7839,'81-04-01',2975,NULL,20);
INSERT INTO emp VALUES (7654,'MARTIN','SALESMAN',7698,'81-09-10',1250,1400,30);
INSERT INTO emp VALUES (7844,'TURNER','SALESMAN',7698,'81-08-21',1500,0,30);
INSERT INTO emp VALUES (7900,'JAMES','CLERK',7698,'81-12-11',950,NULL,30);
INSERT INTO emp VALUES (7521,'WARD','SALESMAN',7698,'81-02-23',1250,500,30);
INSERT INTO emp VALUES (7902,'FORD','ANALYST',7566,'81-12-11',3000,NULL,20);
INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,'80-12-09',800,NULL,20);
INSERT INTO emp VALUES (7788,'SCOTT','ANALYST',7566,'82-12-22',3000,NULL,20);
INSERT INTO emp VALUES (7876,'ADAMS','CLERK',7788,'83-01-15',1100,NULL,20);
```

```
INSERT INTO emp VALUES (7934,'MILLER','CLERK',7782,'82-01-11',1300,NULL,10);
commit;
```

## 7. 테이블 생성 잘 되었는지 확인

```
MariaDB [orcl]> select * from emp;
```

| empno | Ename  | Job       | Mgr  | Hiredate   | Sal  | Comm | Deptno |
|-------|--------|-----------|------|------------|------|------|--------|
| 7839  | KING   | PRESIDENT | NULL | 1981-11-17 | 5000 | NULL | 10     |
| 7698  | BLAKE  | MANAGER   | 7839 | 1981-05-01 | 2850 | NULL | 30     |
| 7782  | CLARK  | MANAGER   | 7839 | 1981-05-09 | 2450 | NULL | 10     |
| 7566  | JONES  | MANAGER   | 7839 | 1981-04-01 | 2975 | NULL | 20     |
| 7654  | MARTIN | SALESMAN  | 7698 | 1981-09-10 | 1250 | 1400 | 30     |
| 7844  | TURNER | SALESMAN  | 7698 | 1981-08-21 | 1500 | 0    | 30     |
| 7900  | JAMES  | CLERK     | 7698 | 1981-12-11 | 950  | NULL | 30     |
| 7521  | WARD   | SALESMAN  | 7698 | 1981-02-23 | 1250 | 500  | 30     |
| 7902  | FORD   | ANALYST   | 7566 | 1981-12-11 | 3000 | NULL | 20     |
| 7369  | SMITH  | CLERK     | 7902 | 1980-12-09 | 800  | NULL | 20     |
| 7788  | SCOTT  | ANALYST   | 7566 | 1982-12-22 | 3000 | NULL | 20     |
| 7876  | ADAMS  | CLERK     | 7788 | 1983-01-15 | 1100 | NULL | 20     |
| 7934  | MILLER | CLERK     | 7782 | 1982-01-11 | 1300 | NULL | 10     |

```
13 rows in set (0.000 sec)
```

문제 289. (maria db에서 JOIN이 가능한지 확인하기) DALLAS에서 근무하는 직원들의 이름과 부서 위치를 출력하시오.

```
MariaDB [orcl]> select e.ename, d.loc
-> from emp e, dept d
-> where e.deptno=d.deptno and d.loc='DALLAS';
```

| ename | loc    |
|-------|--------|
| JONES | DALLAS |
| FORD  | DALLAS |
| SMITH | DALLAS |
| SCOTT | DALLAS |
| ADAMS | DALLAS |

```
5 rows in set (0.000 sec)
```

```
MariaDB [orcl]> select e.ename, d.loc
-> from emp e join dept d
-> on (e.deptno=d.deptno)
-> where d.loc='DALLAS';
```

| ename | loc    |
|-------|--------|
| JONES | DALLAS |
| FORD  | DALLAS |
| SMITH | DALLAS |
| SCOTT | DALLAS |
| ADAMS | DALLAS |

```
5 rows in set (0.000 sec)
```

문제 290. (maria db에서 서브쿼리 가능한지 확인) JONES보다 더 많은 월급을 받는 직원들의 이름과 월급을 출력하시오.

```
MariaDB [orcl]> select ename, sal
-> from emp
-> where sal > (select sal
-> from emp
```

```

-> where ename = 'JONES');
+-----+-----+
| ename | sal |
+-----+-----+
KING	5000
FORD	3000
SCOTT	3000
+-----+-----+
3 rows in set (0.000 sec)

```

문제 291. (maria db에서 데이터 update=수정 이 되는지 확인 // hive는 update가 불가능)  
이름이 SCOTT 인 사원의 월급을 9000으로 변경하시오.

```

MariaDB [orcl]> update emp
-> set sal = 9000
-> where ename ='SCOTT';
Query OK, 1 row affected (0.009 sec)
Rows matched: 1 Changed: 1 Warnings: 0

```

결과 확인

```

MariaDB [orcl]> select ename, sal
-> from emp
-> where ename='SCOTT';
+-----+-----+
| ename | sal |
+-----+-----+
| SCOTT | 9000 |
+-----+-----+
1 row in set (0.000 sec)

```

그런데 rollback을 해보면....?

```

MariaDB [orcl]> rollback;
Query OK, 0 rows affected (0.000 sec)

MariaDB [orcl]> select ename, sal from emp where ename='SCOTT';
+-----+-----+
| ename | sal |
+-----+-----+
| SCOTT | 9000 |
+-----+-----+
1 row in set (0.000 sec)

```

mysql 처럼 기본적으로 자동 commit이 활성화 되어있어서 rollback이 되지 않는다.

★ maria db에서 autocommit 설정을 확인하는법

```
SELECT @@AUTOCOMMIT;
```

```

MariaDB [orcl]> SELECT @@AUTOCOMMIT;
+-----+
| @@AUTOCOMMIT |
+-----+
| 0 |
+-----+

```



**AUTOCOMMIT ON**

```
SELECT AUTOCOMMIT = 1;
```

**AUTOCOMMIT OFF**

```
SELECT AUTOCOMMIT = 0;
```

오토 커밋은 해제해준다.

```
MariaDB [orcl]> SET AUTOCOMMIT = 0;
Query OK, 0 rows affected (0.000 sec)
```

## ■ maria db에서 파티션 테이블 생성하는 방법

### 1. 파티션 테이블이 왜 필요한가?

데이터가 대용량 데이터이면 테이블에서 데이터를 검색할 때 시간이 많이 걸린다. 인덱스를 생성한다해도 인덱스(목차) 도 사이즈가 크기 때문에 인덱스를 통한 데이터 검색효과를 볼 수가 없다.  
그럴때 파티션 테이블을 생성하면 검색속도를 향상 시킬 수 있다.

e.g) 옷장에 셔랍이 4개가 있을때 셔랍마다 봄,여름,가을,겨울 옷을 구분해서 넣어두면 필요한 계절마다 옷을 찾기가 더 편해진다.

즉 물리적으로 데이터를 분리해서 저장하는 것을 '파티션 테이블' 이라 한다.

### 2. 파티션 테이블 만들기

e.g) emp 테이블을 부서번호마다 별도의 파티션에 데이터가 저장되도록 파티션 테이블을 구성한다.

부서번호 10번 → 1번 파티션  
부서번호 20번 → 2번 파티션  
부서번호 30번 → 3번 파티션

#### ▼ script

```
Create table emp_partition
(empno int(4) not null,
 ename varchar(10),
 job varchar(9),
 mgr int(4),
 hiredate date,
 sal int(7),
 comm int(7),
 deptno int(4)) partition by list(deptno)
 (partition p1 values in (10),
 partition p2 values in (20),
 partition p3 values in (30));
```

```
MariaDB [orcl]> Create table emp_partition
-> (empno int(4) not null,
-> ename varchar(10),
-> job varchar(9),
-> mgr int(4),
-> hiredate date,
-> sal int(7),
-> comm int(7),
-> deptno int(4)) partition by list(deptno)
-> (partition p1 values in (10),
-> partition p2 values in (20),
-> partition p3 values in (30));
결과
Query OK, 0 rows affected (0.023 sec)
```

## 데이터 입력

```
INSERT INTO emp_partition VALUES (7839,'KING','PRESIDENT',NULL,'81-11-17',5000,NULL,10);
INSERT INTO emp_partition VALUES (7698,'BLAKE','MANAGER',7839,'81-05-01',2850,NULL,30);
INSERT INTO emp_partition VALUES (7782,'CLARK','MANAGER',7839,'81-05-09',2450,NULL,10);
INSERT INTO emp_partition VALUES (7566,'JONES','MANAGER',7839,'81-04-01',2975,NULL,20);
INSERT INTO emp_partition VALUES (7654,'MARTIN','SALESMAN',7698,'81-09-10',1250,1400,30);
INSERT INTO emp_partition VALUES (7844,'TURNER','SALESMAN',7698,'81-08-21',1500,0,30);
```

```

INSERT INTO emp_partition VALUES (7900,'JAMES','CLERK',7698,'81-12-11',950,NULL,30);
INSERT INTO emp_partition VALUES (7521,'WARD','SALESMAN',7698,'81-02-23',1250,500,30);
INSERT INTO emp_partition VALUES (7902,'FORD','ANALYST',7566,'81-12-11',3000,NULL,20);
INSERT INTO emp_partition VALUES (7369,'SMITH','CLERK',7902,'80-12-09',800,NULL,20);
INSERT INTO emp_partition VALUES (7788,'SCOTT','ANALYST',7566,'82-12-22',3000,NULL,20);
INSERT INTO emp_partition VALUES (7876,'ADAMS','CLERK',7788,'83-01-15',1100,NULL,20);
INSERT INTO emp_partition VALUES (7934,'MILLER','CLERK',7782,'82-01-11',1300,NULL,10);
commit;

```

## 1. 파티션 되어있지 않은 emp 테이블에서 20번 부서번호를 조회

```

MariaDB [orcl]> select ename, sal, deptno
-> from emp
-> where deptno = 20 and ename = 'SCOTT';
+-----+-----+
| ename | sal | deptno |
+-----+-----+
JONES	2975	20
FORD	3000	20
SMITH	800	20
SCOTT	9000	20
ADAMS	1100	20
+-----+-----+
5 rows in set (0.000 sec)

```

## 2. 파티션 테이블에서 부서번호 20번 조회

```

MariaDB [orcl]> select ename, sal, deptno from emp_partition where deptno = 20 and ename = 'SCOTT';
+-----+-----+
| ename | sal | deptno |
+-----+-----+
JONES	2975	20
FORD	3000	20
SMITH	800	20
SCOTT	3000	20
ADAMS	1100	20
+-----+-----+
5 rows in set (0.000 sec)

```

지금은 데이터 수가 워낙 적어서 둘 사이에 차이가 없지만, 대용량 데이터라면 검색속도에 큰 차이를 보이게 된다.

파티션 테이블로 분류가 가능한 경우는 위의 경우 처럼 부서번호별로 구분이 가능하듯 분류 기준이 명확한 것들!

```

MariaDB [orcl]> select * from emp;
+-----+-----+-----+-----+-----+-----+-----+
| empno | Ename | Job | Mgr | Hiredate | Sal | Comm | Deptno |
+-----+-----+-----+-----+-----+-----+-----+
7839	KING	PRESIDENT	NULL	1981-11-17	5000	NULL	10
7698	BLAKE	MANAGER	7839	1981-05-01	2850	NULL	30
7782	CLARK	MANAGER	7839	1981-05-09	2450	NULL	10
7566	JONES	MANAGER	7839	1981-04-01	2975	NULL	20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950	NULL	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000	NULL	20
7369	SMITH	CLERK	7902	1980-12-09	800	NULL	20
7788	SCOTT	ANALYST	7566	1982-12-22	9000	NULL	20
7876	ADAMS	CLERK	7788	1983-01-15	1100	NULL	20
7934	MILLER	CLERK	7782	1982-01-11	1300	NULL	10
+-----+-----+-----+-----+-----+-----+-----+
13 rows in set (0.000 sec)

```

```

MariaDB [orcl]> select * from emp_partition;
+-----+-----+-----+-----+-----+-----+-----+
| empno | ename | job | mgr | hiredate | sal | comm | deptno |
+-----+-----+-----+-----+-----+-----+-----+
7839	KING	PRESIDENT	NULL	1981-11-17	5000	NULL	10
7782	CLARK	MANAGER	7839	1981-05-09	2450	NULL	10
7934	MILLER	CLERK	7782	1982-01-11	1300	NULL	10

```

```

7566	JONES	MANAGER	7839	1981-04-01	2975	NULL	20
7902	FORD	ANALYST	7566	1981-12-11	3000	NULL	20
7369	SMITH	CLERK	7902	1980-12-09	800	NULL	20
7788	SCOTT	ANALYST	7566	1982-12-22	3000	NULL	20
7876	ADAMS	CLERK	7788	1983-01-15	1100	NULL	20
7698	BLAKE	MANAGER	7839	1981-05-01	2850	NULL	30
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950	NULL	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
+-----+-----+-----+-----+-----+-----+-----+
13 rows in set (0.000 sec)
파티션 테이블은 부서번호를 기준으로 정렬이 되어 있다.

```

실제로 빅데이터 환경에서는 큰테이블 몇개는 파티션 테이블로 구성되어 있다.

## 오늘의 마지막문제

문제 292. maria db에서 emp\_partition3라는 이름으로 파티션 테이블을 생성하는데 파티션 키 컬럼을 job으로해서 생성하시오.

기본적으로 mySQL/maria db는 int, range, date, hash 종류의 데이터만 파티션이 가능함.  
컬럼명을 기준으로 할때는 'partition by list columns(컬럼명)' 라고 적어 줘야 함.

```

Create table emp_partition3
(empno int(4) not null,
ename varchar(10),
job varchar(9),
mgr int(4),
hiredate date,
sal int(7),
comm int(7),
deptno int(4)) partition by list columns(job)
(partition p1 values in ('SALESMAN'),
partition p2 values in ('PRESIDENT'),
partition p3 values in ('ANALYST'),
partition p4 values in ('MANAGER'),
partition p5 values in ('CLERK'));

INSERT INTO emp_partition3 VALUES (7839, 'KING', 'PRESIDENT', NULL, '81-11-17', 5000, NULL, 10);
INSERT INTO emp_partition3 VALUES (7698, 'BLAKE', 'MANAGER', 7839, '81-05-01', 2850, NULL, 30);
INSERT INTO emp_partition3 VALUES (7782, 'CLARK', 'MANAGER', 7839, '81-05-09', 2450, NULL, 10);
INSERT INTO emp_partition3 VALUES (7566, 'JONES', 'MANAGER', 7839, '81-04-01', 2975, NULL, 20);
INSERT INTO emp_partition3 VALUES (7654, 'MARTIN', 'SALESMAN', 7698, '81-09-10', 1250, 1400, 30);
INSERT INTO emp_partition3 VALUES (7844, 'TURNER', 'SALESMAN', 7698, '81-08-21', 1500, 0, 30);
INSERT INTO emp_partition3 VALUES (7900, 'JAMES', 'CLERK', 7698, '81-12-11', 950, NULL, 30);
INSERT INTO emp_partition3 VALUES (7521, 'WARD', 'SALESMAN', 7698, '81-02-23', 1250, 500, 30);
INSERT INTO emp_partition3 VALUES (7902, 'FORD', 'ANALYST', 7566, '81-12-11', 3000, NULL, 20);
INSERT INTO emp_partition3 VALUES (7369, 'SMITH', 'CLERK', 7902, '80-12-09', 800, NULL, 20);
INSERT INTO emp_partition3 VALUES (7788, 'SCOTT', 'ANALYST', 7566, '82-12-22', 3000, NULL, 20);
INSERT INTO emp_partition3 VALUES (7876, 'ADAMS', 'CLERK', 7788, '83-01-15', 1100, NULL, 20);
INSERT INTO emp_partition3 VALUES (7934, 'MILLER', 'CLERK', 7782, '82-01-11', 1300, NULL, 10);
commit;

```

```

MariaDB [orcl]> select * from emp_partition3;
+-----+-----+-----+-----+-----+-----+-----+
| empno | ename | job | mgr | hiredate | sal | comm | deptno |
+-----+-----+-----+-----+-----+-----+-----+
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7839	KING	PRESIDENT	NULL	1981-11-17	5000	NULL	10
7902	FORD	ANALYST	7566	1981-12-11	3000	NULL	20
7788	SCOTT	ANALYST	7566	1982-12-22	3000	NULL	20
7698	BLAKE	MANAGER	7839	1981-05-01	2850	NULL	30
7782	CLARK	MANAGER	7839	1981-05-09	2450	NULL	10
7566	JONES	MANAGER	7839	1981-04-01	2975	NULL	20
7900	JAMES	CLERK	7698	1981-12-11	950	NULL	30
7369	SMITH	CLERK	7902	1980-12-09	800	NULL	20
7876	ADAMS	CLERK	7788	1983-01-15	1100	NULL	20
7934	MILLER	CLERK	7782	1982-01-11	1300	NULL	10
+-----+-----+-----+-----+-----+-----+-----+
13 rows in set (0.000 sec)

```

## [21.01.14 리눅스 강의 Day 13 //하둡]

■ 리눅스와 하둡으로 할 수 있는게 무엇인가?

시스템 구축

1. 리눅스 (centos) 설치
2. 하둡 설치
3. 하이버 설치
4. 스칼라 설치
5. 파이썬 아나콘다 설치
6. maria db 설치
7. 부가적 작업 : 원격에서 putty로 접속하게 설정  
mobaxterm을 이용해서 파일 전송과 윈도우에서  
리눅스 스파이더 실행할 수 있도록 함.

이 모든 것은 빅데이터 분석 및 시각화를 구현하기 위한 밑작업!

현업에서 이러한 시스템을 구축해놓고 활용하는 사례?  
미래에셋의 인공지능 프로그램인 로보 어드바이저(주식예측인공지능)



파이썬으로 주식데이터를 웹 스크롤링 → 리눅스서버 → 마리아 db → 주식데이터 저장 → 파이썬과 마리아db연동  
→ 파이썬에서 머신러닝 & 딥러닝 알고리즘으로 주가를 예측 → 파이썬 장고로 화면 구현

하나의 포트폴리오도 이러한 과정을 똑같이 반복한다.

오늘은 구동한 구현한 리눅스 서버를 활용하여 데이터를 시각화하는 것을 실습할 예정!

1. 리눅스 & 하둡 수업의 시험 평가를 제출 포맷
2. 주식 데이터를 웹스크롤링해서 마리아 db에 저장
3. 파이썬과 연동해서 데이터 시각화



### maria db와 파이썬을 연동하는 방법

#### 1. putty로 터미널 창을 열고 py38 를 activate 시키고 mysql 모듈을 설치한다.

```
$ conda activate py389
```

```
(py389) [scott@centos ~]$ pip install mysql-connector-python-rf
```

▼ 결과 : Successfully installed mysql-connector-python-rf-2.2.2 라고 나오면 성공

```
(base) [scott@centos ~]$ conda activate py38
(py38) [scott@centos ~]$ spyder
qt.qpa.screen: QXcbConnection: Could not connect to display
Could not connect to any X display.
(py38) [scott@centos ~]$ pip install mysql-connector-python-rf
Collecting mysql-connector-python-rf
 Downloading mysql-connector-python-rf-2.2.2.tar.gz (11.9 MB)
 |██| 11.9 MB 682 kB/s
Building wheels for collected packages: mysql-connector-python-rf
 Building wheel for mysql-connector-python-rf (setup.py) ... done
 Created wheel for mysql-connector-python-rf: filename=mysql_connector_python_rf-2.2.2-cp38-cp38-linux_x86_64.whl size=249457 sha256=38084658843e956a783619389a9b1ad82eeb33d48a7c4649551d4cdfe825b1a0
 Stored in directory: /home/scott/.cache/pip/wheels/f5/66/87/6d9cef740fd440ef390930fdb6c761dc1efef78ec94a288fd
Successfully built mysql-connector-python-rf
Installing collected packages: mysql-connector-python-rf
Successfully installed mysql-connector-python-rf-2.2.2
(py38) [scott@centos ~]$
```

설치결과를 확인하려면 `$ conda list mysql-connector-python-rf` 를 입력해본다.

## 2. 다른 터미널 창을 하나 더 열고 mysql 로 접속해서 모든 ip 의 접속 권한을 부여합니다.

```

maria db를 root에 설치했으므로
(base) [scott@centos ~]$ su -
Password:
Last login: Wed Jan 13 01:11:52 EST 2021 on pts/1

[root@centos scott]# mysql -u root -p # maria db 로 접속하는 명령어
MariaDB [(none)]> use orcl; # 미리 만들어준 데이터베이스로 입장

#maria db의 db유저에 대한 권한정보를 확인하는 명령어
MariaDB [orcl]> select Host,User,plugin,authentication_string FROM mysql.user; #root 유저에게 모든 권한을 주는 것(권한이 많아야 예러가 안남

MariaDB [orcl]> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '1234';
Query OK, 0 rows affected (0.021 sec)

#다음 단계를 위해 root로 돌아간다.
MariaDB [orcl]> exit;
```

## 3. 마리아 db에 접속하도록 방화벽 설정을 합니다.

💡 #포트3306은 마리아 db를 위한 기본 port 이다. 이 포트를 열어서 파이썬과의 연동이 원활하게 되도록 해줘야 한다.  
 # 만약 아래의 명령어 입력하는데 FirewallD is not running 라는 경고가 뜨면  
 # yum update 를 해준다.  
 이거도 안되면 아래의 두개 사이트 링크 참조  
 link 1: <https://yjshin.tistory.com/entry/CentOS-firewalld-%EB%B0%A9%ED%99%94%EB%B2%BD-%EC%98%A4%EB%A5%98>  
 link 2: <https://tacker.tistory.com/12>

```
[root@centos ~]# firewall-cmd --list-all-zones
[root@centos ~]# firewall-cmd --permanent --zone=public --add-port=3306/tcp
success
[root@centos ~]# exit
logout
```



4. putty에서 스파이더를 실행해서 아래의 코드를 실행합니다.(mobaxterm이 먼저 실행되어 있어야 하고 scott으로 접속한 상태여야 한다.)

```
#이거는 putty에서 실행하는것! // DISPLAY는 자기 자신의 ip주소 (윈도우-cmd-ipconfig입력해서 확인가능, 와이파이인 경우 이 링크 참조)
(py389) [scott@centos ~]$ export DISPLAY=192.168.19.31:0.0
(py389) [scott@centos ~]$ spyder

#아래의 코드가 파이썬에서 해야할 것
```

코드:

```
import mysql.connector
config = {
 "user": "root",
 "password": "1234",
 "host": "192.168.56.1", #local
 "database": "orcl", #Database name
 "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

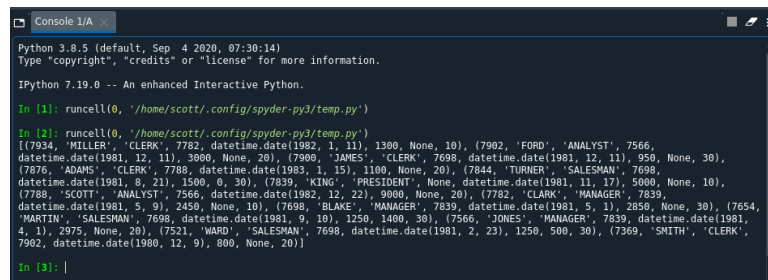
conn = mysql.connector.connect(**config)
db select, insert, update, delete 작업 객체
cursor = conn.cursor()

실행할 select 문 구성
sql = "SELECT * FROM emp ORDER BY 1 DESC"

cursor 객체를 이용해서 수행한다.
cursor.execute(sql)

select 된 결과 셋 얻어오기
resultList = cursor.fetchall() # tuple 이 들어있는 list
print(resultList)
```

결과



```
Console 1/A
Python 3.8.5 (default, Sep 4 2020, 07:30:14)
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: runcell(0, '/home/scott/.config/spyder-py3/temp.py')

In [2]: runcell(0, '/home/scott/.config/spyder-py3/temp.py')
[(7934, 'MILLER', 'CLERK', 7782, datetime.date(1982, 1, 11), 1300, None, 10), (7902, 'FORD', 'ANALYST', 7566,
datetime.date(1981, 12, 11), 3000, None, 20), (7200, 'JAMES', 'CLERK', 7698, datetime.date(1981, 12, 11), 950, None, 30),
(7876, 'ADAMS', 'CLERK', 7788, datetime.date(1983, 1, 15), 1100, None, 20), (7844, 'TURNER', 'SALESMAN', 7698,
datetime.date(1981, 8, 21), 1500, 0, 30), (7839, 'KING', 'PRESIDENT', None, datetime.date(1981, 11, 17), 5000, None, 10),
(7788, 'SCOTT', 'ANALYST', 7566, datetime.date(1982, 12, 22), 9000, None, 20), (7782, 'CLARK', 'MANAGER', 7839,
datetime.date(1981, 5, 9), 2450, None, 10), (7698, 'BLAKE', 'MANAGER', 7839, datetime.date(1981, 5, 1), 2850, None, 30), (7654,
'MARTIN', 'SALESMAN', 7698, datetime.date(1981, 9, 10), 1250, 1400, 30), (7566, 'JONES', 'MANAGER', 7839, datetime.date(1981,
4, 1), 2975, None, 20), (7521, 'WARD', 'SALESMAN', 7698, datetime.date(1981, 2, 23), 1250, 500, 30), (7369, 'SMITH', 'CLERK',
7902, datetime.date(1980, 12, 9), 800, None, 20)]

In [3]: |
```

왼쪽과 같이 파이썬 콘솔창에 결과가 뜨면 잘 설치가 되고 실행된 것!

문제 293. 마리아 디비와 파이썬이 연동된 상태에서 spyder에서 이름과 월급을 출력하시오.

```
import mysql.connector
config = {
 "user": "root",
 "password": "1234",
 "host": "192.168.56.1", #local
 "database": "orcl", #Database name
 "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)
db select, insert, update, delete 작업 객체
cursor = conn.cursor()
```

```
실행할 select 문 구성
sql = "SELECT * FROM emp ORDER BY 1 DESC"

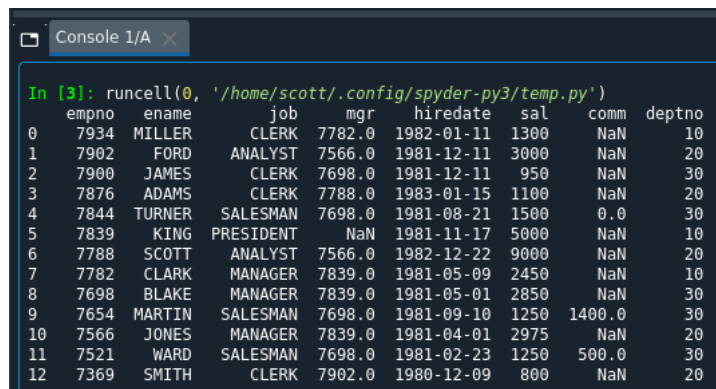
cursor 객체를 이용해서 수행한다.
cursor.execute(sql)

select 된 결과 셋 얻어오기
resultList = cursor.fetchall() # tuple 이 들어있는 list
#print(resultList)

전체 결과의 가독성을 높이기 위해 컬럼을 붙여주는 작업
import pandas as pd
emp=pd.DataFrame(resultList)
emp.columns=['empno','ename','job','mgr','hiredate','sal','comm','deptno'] # 컬럼을 명시해준다.
#print(emp)

#위와 같이 판다스 모듈로 데이터 셋을 바꿔주면 스파이더에서 데이터 처리 작업이 수월해진다.
```

```
#문제에 대한 답
print(emp[['ename','sal']])
```



|    | empno | ename  | job       | mgr    | hiredate   | sal  | comm   | deptno |
|----|-------|--------|-----------|--------|------------|------|--------|--------|
| 0  | 7934  | MILLER | CLERK     | 7782.0 | 1982-01-11 | 1300 | NaN    | 10     |
| 1  | 7902  | FORD   | ANALYST   | 7566.0 | 1981-12-11 | 3000 | NaN    | 20     |
| 2  | 7900  | JAMES  | CLERK     | 7698.0 | 1981-12-11 | 950  | NaN    | 30     |
| 3  | 7876  | ADAMS  | CLERK     | 7788.0 | 1983-01-15 | 1100 | NaN    | 20     |
| 4  | 7844  | TURNER | SALESMAN  | 7698.0 | 1981-08-21 | 1500 | 0.0    | 30     |
| 5  | 7839  | KING   | PRESIDENT | NaN    | 1981-11-17 | 5000 | NaN    | 10     |
| 6  | 7788  | SCOTT  | ANALYST   | 7566.0 | 1982-12-22 | 9000 | NaN    | 20     |
| 7  | 7782  | CLARK  | MANAGER   | 7839.0 | 1981-05-09 | 2450 | NaN    | 10     |
| 8  | 7698  | BLAKE  | MANAGER   | 7839.0 | 1981-05-01 | 2850 | NaN    | 30     |
| 9  | 7654  | MARTIN | SALESMAN  | 7698.0 | 1981-09-10 | 1250 | 1400.0 | 30     |
| 10 | 7566  | JONES  | MANAGER   | 7839.0 | 1981-04-01 | 2975 | NaN    | 20     |
| 11 | 7521  | WARD   | SALESMAN  | 7698.0 | 1981-02-23 | 1250 | 500.0  | 30     |
| 12 | 7369  | SMITH  | CLERK     | 7902.0 | 1980-12-09 | 800  | NaN    | 20     |

이전보다 깔끔하게 나온다.

문제 294. 월급이 3000이상인 직원들의 이름과 월급을 출력하시오. (판다스 문법으로)

```
#앞의 코드들-

print(emp[['ename','sal']] [emp['sal']>=3000])
```

### 문제 295. 직업, 직업별 토탈월급을 출력하시오(sql로)

```
import mysql.connector
config = {
 "user": "root",
 "password": "1234",
 "host": "192.168.56.1", #local
 "database": "orcl", #Database name
 "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)
db select, insert, update, delete 작업 객체
cursor = conn.cursor()

실행할 select 문 구성
sql = """SELECT job, sum(sal)
 FROM emp
 group by job"""

cursor 객체를 이용해서 수행한다.
cursor.execute(sql)

select 된 결과 셋 얻어오기
resultList = cursor.fetchall() # tuple 이 들어있는 list

import pandas as pd
emp=pd.DataFrame(resultList)
emp.columns=['job', 'sumsal']
print(emp)
```

```
job sumsal
0 ANALYST 12000
1 CLERK 4150
2 MANAGER 8275
3 PRESIDENT 5000
4 SALESMAN 4000
```

### 문제 296. 직업, 직업별 토탈월급을 출력하는데 직업이 SALESMAN 은 제외하고, 직업별 토탈월급이 4000이상인것만 출력하고, 직업별 토탈월급이 높은 것부터 출력되게 하시오.

```
import mysql.connector
config = {
 "user": "root",
 "password": "1234",
 "host": "192.168.56.1", #local
 "database": "orcl", #Database name
 "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)
db select, insert, update, delete 작업 객체
cursor = conn.cursor()

실행할 select 문 구성
sql = """SELECT job, sum(sal)
 from emp
 where job!='SALESMAN'
 group by job
 having sum(sal) >=4000
 order by 2 desc""" #전체 컬럼이 2개뿐이고 sumsal은 2번째 컬럼이어서 2입력!

cursor 객체를 이용해서 수행한다.
cursor.execute(sql)

select 된 결과 셋 얻어오기
resultList = cursor.fetchall() # tuple 이 들어있는 list
```

```
import pandas as pd
emp=pd.DataFrame(resultList)
emp.columns=['job','sumsal']
print(emp)
```

문제 297. 부서위치, 부서위치별 토탈월급을 출력하시오.

```
import mysql.connector
config = {
 "user": "root",
 "password": "1234",
 "host": "192.168.56.1", #local
 "database": "orcl", #Database name
 "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)
db select, insert, update, delete 작업 객체
cursor = conn.cursor()

실행할 select 문 구성 (join문주의!)
sql = """SELECT d.loc, sum(e.sal)
 from emp e join dept d
 on (e.deptno = d.deptno)
 group by d.loc"""

cursor 객체를 이용해서 수행한다.
cursor.execute(sql)

select 된 결과 셋 얻어오기
resultList = cursor.fetchall() # tuple 이 들어있는 list

import pandas as pd
emp=pd.DataFrame(resultList)
emp.columns=['loc','sumsal']
print(emp)
```

문제 298. 위의 결과를 막대그래프로 시각화 하시오.

```
import mysql.connector
config = {
 "user": "root",
 "password": "1234",
 "host": "192.168.56.1", #local
 "database": "orcl", #Database name
 "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

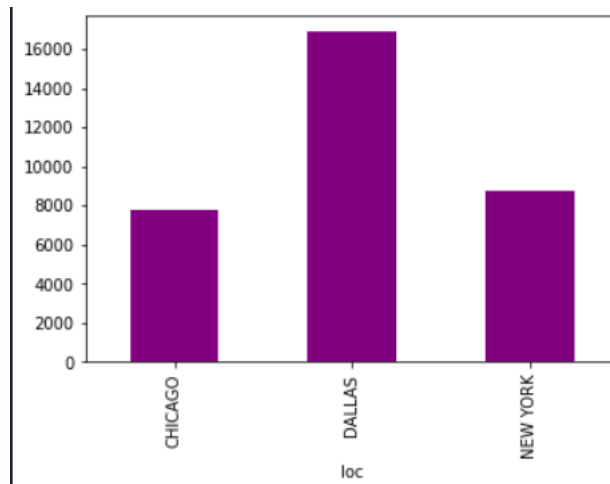
conn = mysql.connector.connect(**config)
db select, insert, update, delete 작업 객체
cursor = conn.cursor()

실행할 select 문 구성
sql = """SELECT d.loc, sum(e.sal)
 from emp e join dept d
 on (e.deptno = d.deptno)
 group by d.loc"""

cursor 객체를 이용해서 수행한다.
cursor.execute(sql)

select 된 결과 셋 얻어오기
resultList = cursor.fetchall() # tuple 이 들어있는 list

import pandas as pd
emp=pd.DataFrame(resultList)
emp.columns=['loc','sumsal']
result=emp['sumsal'].astype(int) # 현재의 데이터는 object타입이라 int 타입으로 바꿔줘야 한다.
result.index=emp['loc']
result.plot(kind='bar',color='purple')
```



문제 299. 현업에서 많이 사용하는 maria db와 mySQL을 편하게 사용할 수 있도록 하는 오라클의 SQL developer와 같은 툴을 설치하고 리눅스 서버의 maria db의 emp 테이블을 조회하시오.

#### ■ 윈도우에서 mySQLworkbench로 리눅스 서버의 마리아 디비 사용하기

"오라클의 sqldeveloper 같은 툴로 마리아 디비에 접속하기"

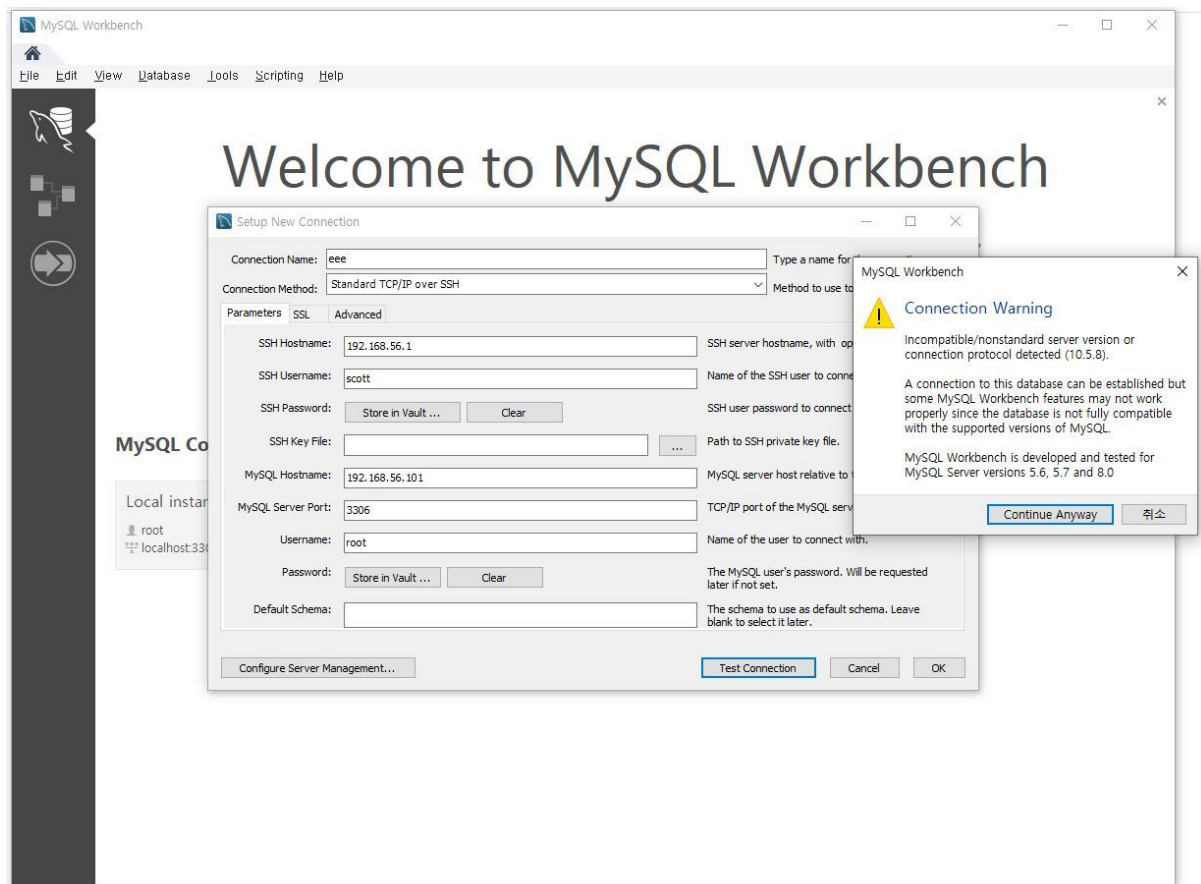
1. 윈도우에서 아래의 프로그램을 설치합니다.

<https://dev.mysql.com/downloads/workbench/>

2. 프로그램을 실행 후 실행하여 접속 정보를 아래와 같이 기술합니다.

SSH Hostname과 MySQL Hostname 는 자신의 ip주소 및 설치한 mySQL(혹은 maria db)의 ip주소가 어떻게 되는지 잘 확인하고!

나의 경우 위아래 둘다 192.168.56.1 이었음.



(점심시간 문제)

문제 300. workbench에서 아래의 문제의결과를 출력하시오.

문제 : 직업, 직업별 평균월급을 출력하는데, 직업별 평균 월급이 2000이상인 경우만 출력하고 직업별 평균 월급이 높은 순으로 출력하시오.

(workbench 결과를 캡처해서 카페에 답글로 달기!)

```
select job, avg(sal)
from emp
group by job
having avg(sal) >=2000
order by 2 desc;
```

★ 마리아 db와 연동하여 파이썬에서 시각

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/cbd6030d-61e8-4f5f-a0b2-e391a5c5da44/Cas e.csv>

1. 마리아 db에 코로나 데이터 입력

1) 데이터 넣을 테이블 생성

```

use orcl;
drop table cov_case;

create table cov_case
(
case_id int(8),
province varchar(30),
city varchar(20),
group2 varchar(20),
infection_case varchar(50),
confirmed float,
latitude varchar(20),
longitude varchar(20));

```

2) home/scott(리눅스서버) 에 저장된 Case.csv를 불러온다. (마리아 db가 있는 터미널 창에서!)

```

LOAD DATA LOCAL INFILE '/home/scott/Case.csv'
REPLACE

INTO TABLE orcl.cov_case

fields TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'

IGNORE 1 LINES

(case_id, province, city, group2, infection_case, confirmed, latitude, longitude);

```

3) 불러온 데이터가 잘 되었는지 확인.

```

select * from cov_case;

```

## 2. 스파이더에서 마리아 디비의 데이터 불러오기

```

import mysql.connector
import pandas as pd

config = {
"user": "root",
"password": "1234",
"host": "192.168.56.101", #local
"database": "orcl", #Database name
"port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)

db select, insert, update, delete 작업 객체
cursor = conn.cursor()

실행할 select 문 구성
sql = """SELECT city, sum(confirmed) sum2

```

```

FROM cov_case

where city is not null

group by city

order by sum2 desc

"""

cursor 객체를 이용해서 수행한다.

cursor.execute(sql)

select 된 결과 셋 얻어오기

resultList = cursor.fetchall() # tuple 이 들어있는 list

df = pd.DataFrame(resultList)

print(df)

```

### 3. 출력하고자 하는 데이터의 SQL을 작성

```

SELECT ifnull(city,'esc'), sum(confirmed) sum2
FROM cov_case
where trim(city) is not null and city !=''
group by city
order by sum2 desc;

```

### 4. 시각화 하기

```

result = df['cnt']
result.index = df['city']
result.plot(kind='bar', color='red')

```

문제 301. 도시명, 도시명별 토탈 감염자수를 출력하시오.

```

select city, sum(confirmed)
from cov_case
group by city;

```

는 null값은 아니지만 city가 분류되지 않은 확진자의 데이터가 있다.

문제 302. 위의 결과를 다시 출력하는데 도시명이 null 이 아닌 것만 출력하시오.

```

select city, sum(confirmed)
from cov_case
where city is not null
group by city;

```

를 했으나 null값이 아니라 단순 공백이어서 인지 제외 되지 않음

정답

```

select city, sum(confirmed)
from cov_case

```



```
where trim(city) is not null and city !='' #싱글 쿼테이션 2개
group by city;
```



기존 Case.csv는 도시나 구 마다 youngsan-gu 이런식으로 -가 붙어있음.  
이걸 지워주려면 리눅스에서 `vi case.csv` 한 뒤에 `:%s/-//g` 를 순서대로 입력해준다.  
여기서 /(사이) // 사이에 있는것이 삭제할 문자열.

문제 303. 위의 결과를 다시 출력하는데 토탈 확진자수가 높은 것부터 출력하시오.

```
select city, sum(confirmed)
from cov_case
where trim(city) is not null and city !='' #싱글 쿼테이션 2개
group by city
order by 2 desc;
```

문제 304. 위의 결과를 다시 출력하는데 토탈 확진자수가 100명 이상인 것만 출력하시오.  
(그룹된 컬럼의 조건절은 having절에 입력해야 한다!)

```
select city, sum(confirmed)
from cov_case
where trim(city) is not null and city !=''
group by city
having sum(confirmed) >= 100
order by 2 desc;
```

문제 305. 위의 결과를 spyder에서 막대그래프로 시각화 하시오.

```
import mysql.connector
import pandas as pd

config = {
 "user": "root",
 "password": "1234",
 "host": "192.168.56.101", #local
 "database": "orcl", #Database name
 "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)

db select, insert, update, delete 작업 객체
cursor = conn.cursor()

실행할 select 문 구성
sql = """ SELECT ifnull(city,'esc'), round(sum(confirmed)) sum2
FROM cov_case
where trim(city) is not null and city !=''
group by city
having sum(confirmed) > 50
order by sum2 desc
"""

cursor 객체를 이용해서 수행한다.
cursor.execute(sql)

select 된 결과 셋 얻어오기
resultList = cursor.fetchall() # tuple 이 들어있는 list
```

```
df = pd.DataFrame(resultList)
df.columns = ['city', 'cnt']
print(df[['city', 'cnt']])

시각화
result = df['cnt']
result.index = df['city']
result.plot(kind='bar', color='red')
```

설명 : os의 csv파일을 maria db에 넣고 SQL로 데이터를 추출해서 파이썬으로 시각화 한 것.

문제 306. province를 중복제거해서 출력하시오.

```
select distinct province
from cov_case;
```

문제 307. 지역(province), 지역별 토탈 확진자수를 막대그래프로 시각화 하시오.

```
import mysql.connector
import pandas as pd

config = {
 "user": "root",
 "password": "1234",
 "host": "192.168.56.1", #local
 "database": "orcl", #Database name
 "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)

db select, insert, update, delete 작업 객체
cursor = conn.cursor()

실행할 select 문 구성
sql = """ SELECT ifnull(province,'esc'), round(sum(confirmed)) sum2
FROM cov_case
where trim(city) is not null and city !=''
group by province
having sum(confirmed) > 50
order by sum2 desc
"""

cursor 객체를 이용해서 수행한다.
cursor.execute(sql)

select 된 결과 셋 얻어오기
resultList = cursor.fetchall() # tuple 이 들어있는 list
df = pd.DataFrame(resultList)
df.columns = ['city', 'cnt']
print(df[['city', 'cnt']])

시각화
result = df['cnt']
result.index = df['city']
result.plot(kind='bar', color='red')
```



하둡과 리눅스 NCS 시험 → 제출물로 변경 예정!

제출물 만드는 순서

1. 큰 질문 (예시) 코로나는 남자와 여자중에 누가 더 많이 감염되었는가?
2. csv/txt 데이터 구한다. (수집)
3. maria db에 테이블로 생성한다.
4. 파이썬(spyder)에서 시각화를 한다.
5. 시각화 그래프와 함께 짧게 결론(시각화한 결과 설명)을 내린다.

제출 형식 : 카페 게시판에 제출

제출 기한 : 다음주 금요일(1/22)까지

## 제출전 연습

1. 큰 질문 (예시) 코로나는 남자와 여자중에 누가 더 많이 감염되었는가?

문제 308. 코로나는 남자와 여자중에 누가 더 많이 감염 되었는가?

2. csv/txt 데이터 구한다. (수집) - 케글에서 수집한 자료로 만들어준 데이터가 있음.

```
$ head Patientinfo.csv
```

3. maria db에 테이블로 생성한다.

```
drop table pati;

create table pati
(patient_id float,
 sex varchar(30),
 age varchar(30),
 country varchar(30),
 province varchar(30),
 city varchar(30),
 infection_case varchar(60),
 infected_by varchar(60),
 contact_number varchar(30),
 symptom_onset_date varchar(30),
 confirmed_date varchar(30),
 released_date varchar(30),
 deceased_date varchar(30),
 state varchar(30));

LOAD DATA LOCAL INFILE '/home/scott/PatientInfo.csv'
REPLACE
INTO TABLE orcl.pati
fields TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(patient_id,sex,age,country,province,city,infection_case,infected_by,contact_number,symptom_onset_date,confirmed_date,released_
```

성별, 성별별 인원수를 출력하시오.

```
select ifnull(sex,'no response'), count(*) #ifnull -> null값일 경우 어떻게 처리할지
from pati
group by sex;
```

```
select case when sex='' then 'no response'
 else sex end as gender, count(*)
from pati
group by sex;
```

#### 4. 파이썬(spyder)에서 시각화를 한다.

```
import mysql.connector
import pandas as pd

config = {
 "user": "root",
 "password": "1234",
 "host": "192.168.56.1", #local
 "database": "orcl", #Database name
 "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)

db select, insert, update, delete 작업 객체
cursor = conn.cursor()

실행할 select 문 구성(위에서 작성한 sql 붙여넣기)
sql = """ select case when sex='' then 'no response'
 else sex end as gender, count(*)
from pati
group by sex;
"""

cursor 객체를 이용해서 수행한다.
cursor.execute(sql)

select 된 결과 셋 얻어오기
resultList = cursor.fetchall() # tuple 이 들어있는 list
df = pd.DataFrame(resultList)
df.columns = ['city', 'cnt']
print(df[['city', 'cnt']])

시각화
result = df['cnt']
result.index = df['city']
result.plot(kind='bar', color='red')
```

#### 5. 시각화 그래프와 함께 짧게 결론(시각화한 결과 설명)을 내린다.

