

■ 리눅스 수업

리눅스를 배워야 하는 이유?

1. 데이터 분석가가 갖추어야 할 기본 기술 중에 하둡 사용이 있습니다.
(하둡이 리눅스에 기반을 두고 있기 때문에 리눅스를 먼저 배워야 합니다.)
2. 딥러닝을 구현할 때 현업에서 많은 부분이 리눅스에 파이썬을 사용해서 구현을 하고 있습니다.
3. 빅데이터 기사 시험에 하둡이 나오는데 여러 용어들에 대한 정확한 이해가 있어야 시험에 합격할 수 있습니다.
4. 딥러닝으로 뭔가 좀 재미있는 것을 만들고 싶다면 리눅스를 알아야합니다.

■ 리눅스를 배워야 하는 이유

■ 리눅스란 무엇인가?

유닉스(unix)가 너무 고가여서 리눅스 오픈 소스를 핀란드의 리누즈 토발즈 학생이 1991년 11월에 개발을 했다. 리누즈 토발즈가 리눅스 커널(자동차의 엔진과 같음)을 개발하고 소스를 무료로 공개했다. 그리고 전세계의 개발자들의 이 오픈 소스를 가져다가 더 좋게 개선해서 다시 인터넷에 올리고 또 올리고 하는 작업을 반복하다 보니 리눅스 os가 유닉스보다 더 가볍고 안정적이게 되었다.

GNU 프로젝트 --->

누구든지 배포된 오픈소스를 가져다가 개발 할 수 있고 돈을 벌 목적으로 상용화를 할 수도 있는데 한가지 지켜야 할 약속은 이 소스를 가져다가 더 좋게 수정했으면 그 코드를 인터넷에 올려줘야 하는 약속입니다.

레드햇(상용버전) -----> Cent os

유료

무료

• 리눅스의 종류

1. Oracle linus
2. Cent os
3. Ubunt

■ 리눅스 설치

1. Oracle virtual box를 설치합니다.
(가상의 컴퓨터를 내 컴퓨터 안에 만든다)

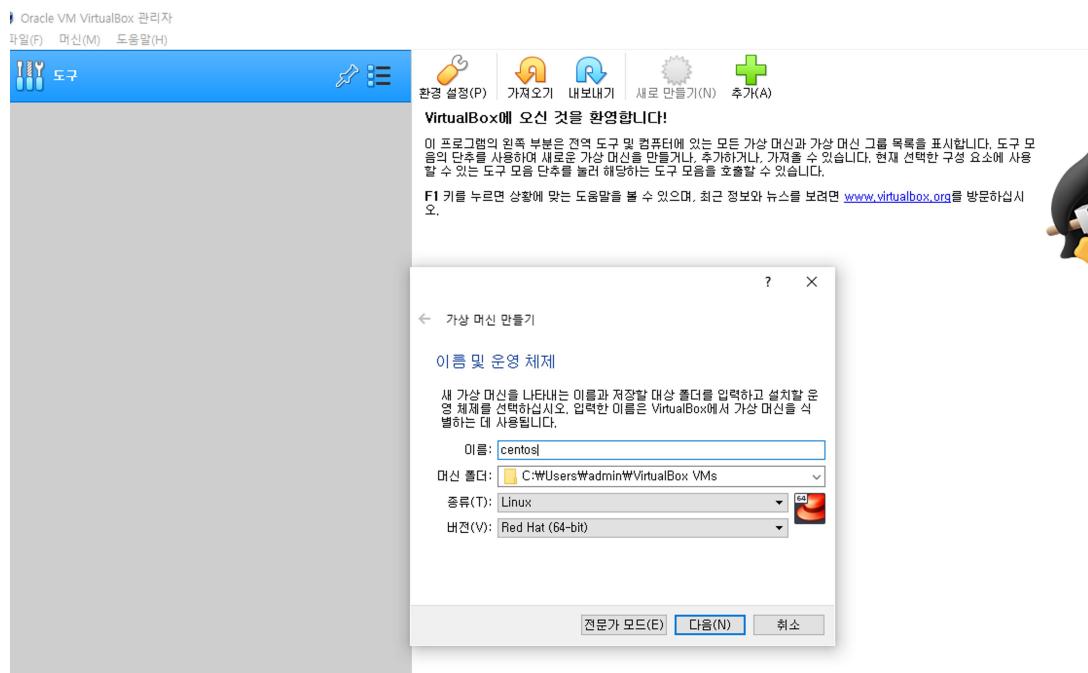
<https://hobby.tw/442>

2. centos 7버전 다운로드

http://mirror.anigel.com/CentOS/7.9.2009/isos/x86_64/

Index of /CentOS/7.9.2009/isos/x86_64

mirror.anigel.com



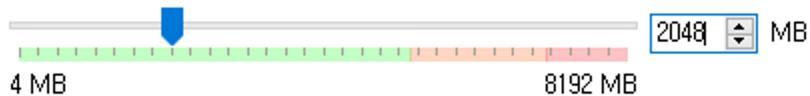
? X

← 가상 머신 만들기

메모리 크기

가상 머신에 할당할 메모리(RAM) 크기를 메가바이트 단위로 입력하십시오.

추천 메모리 크기는 **1024 MB**입니다.



[다음(N)] [취소]

? X

← 가상 머신 만들기

하드 디스크

필요하다면 새 가상 머신에 가상 하드 디스크를 추가할 수 있습니다. 새 하드 디스크 파일을 만들거나, 목록에서 선택하거나, 풀더 아이콘을 통하여 다른 위치에 있는 가상 하드 디스크 파일을 선택할 수 있습니다.

더 자세한 구성이 필요하다면 이 단계를 건너뛰고 가상 머신을 만든 다음 설정을 진행하십시오.

추천하는 하드 디스크 크기는 **8.00 GB**입니다.

- 가상 하드 디스크를 추가하지 않음(D)
- 지금 새 가상 하드 디스크 만들기(C)
- 기존 가상 하드 디스크 파일 사용(U)

비어 있음

[만들기] [취소]

? ×

← 가상 하드 디스크 만들기

하드 디스크 파일 종류

새 가상 하드 디스크 파일 형식을 선택하십시오. 다른 가상화 소프트웨어에서 디스크를 사용하지 않으려면 선택을 변경하지 않아도 됩니다.

- VDI(VirtualBox 디스크 이미지)
- VHD(가상 하드 디스크)
- VMDK(가상 머신 디스크)

전문가 모드(E) 다음(N) 취소

? ×

← 가상 하드 디스크 만들기

물리적 하드 드라이브에 저장

새 가상 하드 디스크 파일을 사용하는 대로 커지게 할 것인지(동적 할당) 최대 크기로 만들 것인지(정적 할당) 선택하십시오.

동적 할당 하드 디스크 파일은 가상 디스크를 사용할 때 고정된 **최대 크기까지** 파일 크기가 커지지만, 사용량이 줄어들어도 자동적으로 작아지지는 않습니다.

고정 크기 하드 디스크 파일은 만드는 데 더 오래 걸리지만 사용할 때 더 빠릅니다.

- 동적 할당(D)
- 고정 크기(F)

다음(N) 취소

? X

← 가상 하드 디스크 만들기

파일 위치 및 크기

새 가상 하드 디스크 파일의 이름을 아래 상자에 입력하거나 풀더 아이콘을 클릭해서 파일을 생성할 풀더를 지정할 수 있습니다.

C:\Users\admin\VirtualBox VMs\centos\centos.vdi 

새 가상 하드 디스크 크기를 메가바이트 단위로 입력하십시오. 가상 머신에서 가상 하드 드라이브에 저장할 수 있는 데이터의 최대 크기입니다.

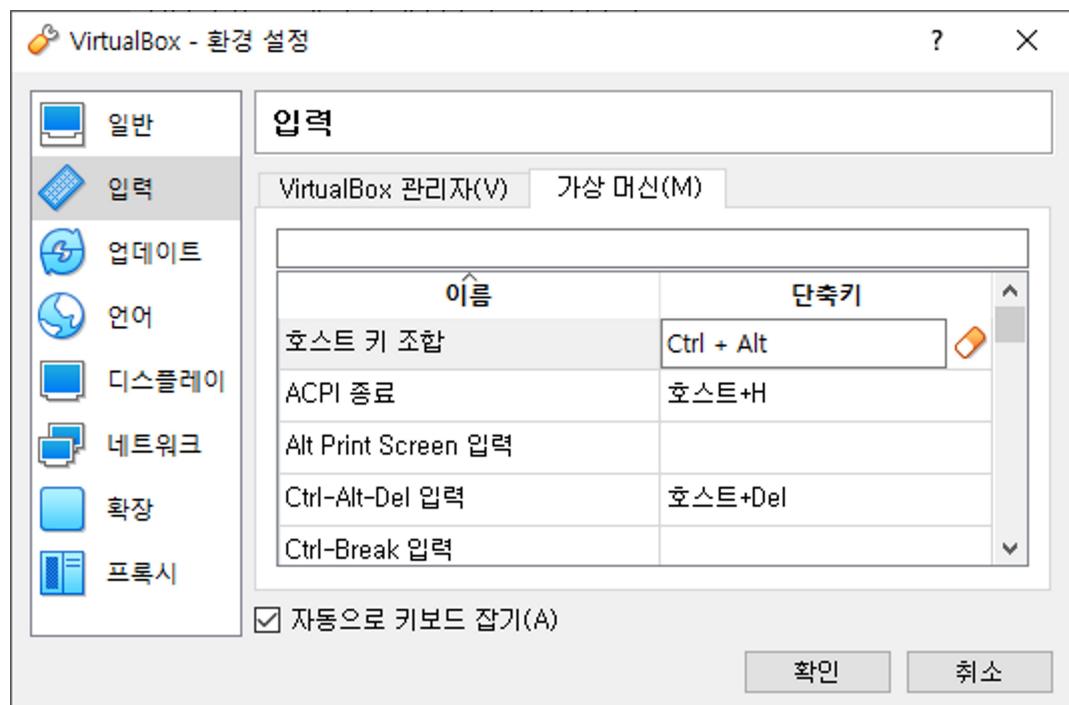


4.00 MB 2.00 TB

30.83 GB

만들기

취소



■ 게스트 cd 확장 설치

- 왜 이작업을 해야 하는가?
 - 마우스와 키보드로 윈도우와 리눅스 가상 서버를 왔다갔다 하게 되면 불편하게 있어서 좀 편하게 가상 서버의 리눅스를 이용하기 위해 작업을 해야 합니다.
- 1. root로 접속한다.
2. 게스트 cd를 cd롬에 입력한다.
3. 게스트 cd를 실행한다.
4. 리눅스 시스템을 rebooting 합니다.

오른쪽 마우스 -> open terminal 들어가서 ifconfig 치기

■ 리눅스 시스템에서 인터넷이 되게 설정하는 방법

1. 리눅스의 시작 버튼을 누른다
2. system tools의 settings를 누른다.
3. network 텁을 클릭합니다.
4. Ethernet 위의 것을 on으로 켠다.
5. firefox를 켜고 naver로 접속합니다.

■ 리눅스 기본 명령어

田 1. cd 명령어

Change Directory 명령어로 디렉토리를 이동하는 명령어입니다.

예제 : # whoami <-- 접속한 내가 누구인지 확인하는 명령어

```
# pwd      <-- 현재 내가 있는 디렉토리를 확인하는 명령어  
(print working directory)
```

```
# ls       <-- list 명령어로 현재 디렉토리에 있는 폴더와 파일을 확인  
# cd Documents <-- Documents 디렉토리로 이동  
# pwd      <-- 현재 내가 있는 디렉토리를 확인하는 명령어  
# cd ..    <-- 뒤로 이동 ( 그 전 디렉토리로 이동 )  
# pwd      <-- 현재 내가 있는 디렉토리를 확인하는 명령어
```

- 경로(path)에는 크게 2가지 경로가 있습니다.

- a. 절대경로 : cd 내가 가고자 하는 정확한 위치
예 : # pwd
cd / root/ Desktop
- b. 상대경로 : 나의 현재위치를 상대로 이동하겠다.
예 : # cd .. <-- 나의 현재 위치를 상대로 상위 디렉토리로 이동하겠다.

② 2. touch 명령어

파일의 용량이 0인 파일을 생성하는 명령어 입니다.

예 : # touch a1.txt
ls -l a1.txt <-- a1.txt의 용량을 확인합니다

③ 3. mkdir 명령어

디렉토리를 만드는 명령어
make directory의 약자입니다.

- ❖ 명령어에 대한 매뉴얼 보는 방법
man mkdir

빠져 나오려면 q를 누르면 됩니다.

④ 4. rm 명령어

파일이나 디렉토리를 삭제하는 명령어

- ❖ 주의사항!!!
리눅스랑 유닉스는 삭제할 때 특히 주의해야 합니다!!!!

예제: # touch bbb.txt
ls -l bbb.txt
rm bbb.txt
ls -l bbb.txt

⑤ 5. rmdir 명령어

rmdir 명령어는 디렉토리를 삭제하는 명령어

예제 : # mkdir test50
ls -ld test50
rmdir test50

⑥ 6. alias 명령어

자주 수행하는 명령어들을 쉽게 사용할 수 있도록 설정하는 명령어

```
예제 : # pwd  
      # alias p = pwd
```

설명 : pwd라고 다 쓰지 않고 p만 써서 수행하겠다.

⑦ 7. cat 명령어

파일의 내용을 화면에 출력하는 명령어

```
예제 : # cat emp.txt
```

⑧ 8. redirection 명령어

화면에 출력되는 결과를 파일로 저장하는 명령어

>> : 없으면 파일을 생성하고 있으면 기존 파일 뒤에 덧붙이겠다.
> : 파일을 생성하겠다. 기존에 파일이 있으면 그냥 덮어쓰겠다.

```
예: # cat emp.txt >> emp2.txt  
    # ls -l emp*.txt  
    # cat emp2.txt
```

설명 : cat emp.txt로 본 화면의 결과를 emp2.txt로 저장하겠다.

⑨ 9. more 명령어

1페이지가 넘는 문서의 내용을 화면에 출력할 때 페이지 단위로 볼 수 있는 명령어

예제: #more jobs.txt
전진키 : 스페이스
후진키 : b
페이지 단위로 내리기 : f

⑩ 10. head 명령어

문서의 처음 몇줄을 화면에 출력하는 명령어

```
예제 : # head 출력줄수 파일명  
      # head -5 jobs.txt
```

② 11. wc 명령어

파일 안에 단어의 개수 또는 라인수를 출력하는 명령어

예제 : # wc -l 파일명
wc -l jobs.txt

옵션 : -l : 라인수

-w : 단어의 개수

-c : 철자의 개수

② 12. grep 명령어

파일 안에 포함된 특정 단어나 구문을 검색하는 명령어

예제: # grep '찾고싶은 단어' 파일명
grep 'SCOTT' emp.txt

❖ 위의 결과를 보면 emp.txt에서 SCOTT이 포함된 행만 출력하고 있습니다.

예제 : # grep -i 'scott' emp.txt

❖ 설명 : -i 옵션은 대소문자를 구분하지 않겠다는 뜻입니다.

12/30

2020년 12월 30일 수요일 오전 10:14

⑩ 13. awk 명령어

특정 컬럼을 출력하고자 할 때 사용하는 명령어

예제 : # awk '패턴 {action}' 대상 파일명
awk '\$3=="SALESMAN" {print \$2, \$3}' emp.txt

❖ 설명 : 업이 SALESMAN인 사원의 이름과 직업을 출력하는 명령어

❖ 리눅스 연산자 3가지 정리

1. 산술 연산자 : +, -, *, /
2. 비교 연산자 : >, <, >=, <=, ==, !=
3. 논리 연산자 : &&, ||, !

- 만약에 오라클의 substr과 같은 기능이 있는지 확인을 하고 싶다면?
man awk

/substr <--- 현재 보고 있는 페이지에서 substr을 검색하는 명령어
n을 누르면 다음으로 넘어갑니다.

⑪ 14. sort 명령어

data를 특정 컬럼을 기준으로 정렬하는 명령어

예제 : #sort 옵션 파일명
sort -nk 6 emp.txt

❖ 설명 : -n옵션을 사용하면 숫자로 변환해서 정렬을 해줍니다.

-k를 사용하고 6을 쓰면 6번째 컬럼을 기준으로 정렬하겠다는 뜻입니다.

아래와 같이 -r옵션을 사용하면 월급이 높은 사원부터 출력됩니다.

(reverse의 약자)

sort -nkr 6 emp.txt

⑫ 15. uniq 명령어

중복된 라인을 제거하는 명령어

예: # uniq 옵션 파일명

② 16. echo 명령어

출력하고자 하는 글자를 출력할 때 사용하는 명령어

파이썬의 print와 같은 명령어입니다.

변수 안의 글자를 출력할 때 사용하는 명령어

예제 : # a = 1

```
# echo $a  
# b='scott'  
# echo $b
```

설명 : 변수 안에 있는 값을 출력할 때는 \$를 앞에 붙여줘야합니다.

• 위의 리눅스 명령어 여러개를 파일로 저장해서 한번에 수행하는 방법:

vi 편집기 화면으로 들어가서 위의 3줄을 복사해서 붙여넣으려면 알파벳 i를 누르고 아래쪽에 insert가 나오는지 확인.

shift +ctrl+c 혹은 shift + insert 눌러서 붙여넣고 저장을 해야하는데 저장을 하려면 esc 키를 눌러서 아래쪽에 insert를 사라지게 하고 대문자 Z를 두번 연속으로 눌러서 저장하고 빠져나오면 된다.

```
# vi job2.sh  
# i 누르고 복사해서 shift + ctrl+ c  
#esc 누르기  
대문자 Z 두번 연속 누르기
```

그리고 sh job2.sh 입력

Enter the job name~~~ 나오면 원하는 job의 이름을 입력해준다.

sh job2.sh <---- 확장자가 .sh는 쉘스크립트 입니다.

쉘스크립트는 리눅스 명령어를 모아놓은 것입니다.

쉘스크립트를 잘 작성할 줄 알면 업무의 많은 부분을 자동화 할 수 있습니다.

sh 명령어로 쉘스크립트를 실행합니다.

vi job2.sh에 저장한거 지우기 :

insert 모드 들어가서 DD 입력

➤ 모 금융회사에서 데이터 분석을 할 때 고객들을 나이대별로(20대, 30대, 40대) 별도의

텍스트 파일을 생성할 때 유용함

예제 : 20대.txt, 30대.txt, 40대.txt

12/31

2020년 12월 31일 목요일 오전 9:48

- ❖ 스크린의 화면 잠금이 되면서 다시 패스워드를 입력해야 되는 것을 해지하는 방법
리눅스 시작버튼 --> system tools ---> settings ---> privacy ---> screenlock ---> off

② 17. diff 명령어

두 파일간의 차이점을 찾아서 알려주는 명령어

예: # diff emp.txt dallas.txt

③ 18. find 명령어

검색 하고자하는 파일을 찾을 때 사용하는 명령어

예: # find 디렉토리 -name 파일명 -print

 ↑ ↑

검색할 디렉토리 검색할 파일명

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

설명 : /root 디렉토리 밑에 emp.txt라는 파일이 있는지 검색하시오!

결과:

/root/Desktop/emp.txt

/root/emp.txt

emp.txt파일이 어디어디 있는지 다 찾아주고 있다.

■ 리눅스 centos7에서 아나콘다 설치하기

1. 리눅스에서 firefox를 실행하고 아래의 url로 접속한다. 아나콘다 링크로 들어가 리눅스 용 다운 쪽으로 가서

오른쪽 버튼 - copy link location을 눌러서 링크를 복사한다.

2. 터미널창을 열고 wget명령어로 링크를 실행한다.

wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-x86_64.sh

3. 설치 파일을 실행합니다.

sh Anaconda3-2020.11-Linux-x86_64.sh

엔터치기, yes

4. 리눅스의 환경설정 파일 bashrc 파일을 수행합니다.

```
# source .bashrc
```

5. 아나콘다 가상환경 만들기

아나콘다 가상환경이란? 머신러닝을 위한 파이썬 환경과 딥러닝을 위한 파이썬 환경이 서로 다르므로 각각 별도의 환경을 만들어서 파이썬을 사용하고 싶다면 가상환경을 생성하면 됩니다.

```
# conda create -n py38
# conda activate py38
```

6. spyder를 설치합니다.

```
#conda install spyder
```

여기서 에러가 나면

```
# pip install spyder
```

② 19. tar 명령어

파일을 압축하고 압축을 해제하는 명령어

예제: 1. 압축할 때

```
# tar cvf 압축파일명 압축파일대상
```

2. 압축을 해제할 때

```
#tar xvf 압축파일명 압축을 해제할 위치
```

❖ 옵션: c : compress, 여러개의 파일을 하나로 만들어라~

v : view, 압축되는 과정을 보여달라

f : file, 생성되는 파일명을 지정

x : extract, 묶어있는 파일을 풀어줘라

-C : 압축이 풀릴 위치를 지정

② 20. ln 명령어

os 윈도우의 바로가기 기능과 유사함

os 윈도우 이용할 때 바탕화면에서 바로가기 버튼을 가져다 두면 바로가기만 누르면 빨리 실행할 수 있게 되므로 자주 실행하는 문서의 경우 바탕화면에 바로가기를 가져다 두면 편

리합니다. 이 기능이 리눅스 ln 명령어로 지원이 됩니다.

예제. # cd

```
# ls -l find_job.sh  
# cat find_job.sh  
# ln -s /root/find_job.sh /root/Desktop?find_job.sh
```

설명: 현재 디렉토리의 find_job.sh를 바탕화면(/root/Desktop)에 find_job.sh로 바로가기로 생성하겠다.

② 21. sed 명령어

grep 명령어는 파일의 특정 내용을 검색하는 기능을 갖는다면 sed 명령어는 검색뿐만 아니라 내용을 변경할 수도 있습니다.

예제: # cd

```
# sed 's/KING/yyy/' emp.txt
```

설명: emp.txt에서 KING을 yyy로 변경하겠다. 그런데 위의 명령어는 변경해서 보여주는 것 이지 실제로 데이터가 변경된 것은 아닙니다.

```
# cat emp.txt
```

③ 22. case 명령어

if문과 유사한 리눅스의 쉘 스크립트 명령어

예제1. 예전에 만들었던 find_job.sh와 find_deptno.sh가 잘 수행되는지 확인하시오.

```
# sh find_sal.sh  
# sh find_file.sh
```

예제2. 아래의 스크립트를 먼저 메모장에서 작성하세요~

```
echo " Press number 1 for confirm employees sal  
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
```

예제3. /root 밑에서 vi a.sh를 열어서 위의 스크립트를 복사해서 붙여넣고 아래와 같이 실행 하시오!

```
# cd  
# vi a.sh  
  
# sh a.sh
```

㉓ cp 명령어

파일을 복사하는 명령어

예제: # cp 파일명 복사할 파일명

```
# cp 위치/파일명 위치/복사할 파일명  
# cp emp.txt emp400.txt  
# cp /root/emp.txt /root/test01/emp400.txt
```

㉔ mv 명령어

파일의 이름을 바꾸거나 파일을 다른 디렉토리로 이동하는 명령어

예제: # mv 기존파일명 새로운 파일명

```
# mv emp.txt emp500.txt  
# ls emp500.txt  
# mv emp500.txt ./backup/  
설명 : emp500.txt를 현재 디렉토리 밑에 backup 밑에 이동
```

■ centos에서 spyder 설치하기

1. 아래의 py38 가상환경에 spyder가 있는지 확인하시오!
(py38) [root@localhost ~]

만약에 basse에 있다면 conda activate py38이라고 하세요~

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

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

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

■ 리눅스에서 아나콘다 설치하고 spyder 설치

※ 중요: 아나콘다 설치하고 spyder 설치하는 작업을 root에서 하지말고 scott에서 수행해야 합니다.

1. root에서 로그아웃하고 scott으로 접속한다.
2. 아나콘다 실행파일을 다운로드 받습니다.
3. \$ sh A

■ 지난주에 배웠던 리눅스 수업 내용

1. 리눅스 기본 명령어 24가지 명령어
2. 리눅스에서 아나콘다 설치
3. vi 편집기
4. 쉘 스크립트 생성

■ 리눅스 수업 목차

1. 리눅스란 무엇인가?
2. 리눅스 설치
3. 리눅스 기본 명령어
4. 아나콘다 설치
5. vi 편집기 명령어
6. 권한 관리 명령어

■ vi 편집기 명령어

- vi 편집기란? 리눅스 안에서 사용할 수 있는 문서 편집기
- vi는 Visual Editor의 약자
- vi 편집기 명령모드 3가지
 - a. command 모드 : vi 편집기의 기본 모드이며 vi를 실행하면 바로 보이는 화면 방향키로 왔다갔다 할 수 있는 모드
 - b. edit 모드 :
a, i, o, x 등을 누르면서 내용을 입력 또는 삭제하는 명령모드
 - c. lastline 모드 :
입력 모드에서 저장, 종료, 강제종료등의 명령어를 입력하는 모드

:wq 저장하고 종료 (단축키 ZZ)
:q 저장안하고 종료(단축키:ZQ)

■ vi 편집기 내에서 커서 이동

1. j : 아래로 이동
2. k : 위로 이동

3. h : 왼쪽으로 이동
4. l : 오른쪽으로 이동
5. 1G : 맨 위로 이동
6. G : 맨 아래로 이동
7. :set nu : 파일 내의 텍스트에 번호 표시
8. :set nonu: 번호를 안보이게 하는 명령어
9. gg : 맨 위로 이동하는 단축키

■ vi 편집기의 삭제 명령어

1. x : 철자 하나 삭제
2. dd : 한 행 삭제
3. dw : 커서가 있는 단어 삭제
4. :5, 10 d : 5~10번째 행 삭제
5. D : 커서 오른쪽 행 삭제

■ vi 편집기의 취소 명령어

u : 방금 작업했던것을 취소하겠다.

■ vi 편집기의 수정 명령어

a : 커서 다음에 입력하겠다.

i : 커서 전에 입력하겠다.

r : 커서에 위치하는 철자를 수정하겠다.

■ 리눅스에서 아나콘다 설치하고 spyder 설치

※ 중요: 아나콘다 설치하고 spyder 설치하는 작업을 root 에서 하지 말고 scott 에서 수행해야합니다.

1. root 에서 로그아웃하고 scott 으로 접속한다.
2. 아나콘다 실행파일을 다운로드 받습니다.
3. \$ sh Anaconda3-2020.11-Linux-x86_64.sh
4. \$ source .bashrc
5. \$ conda create -n py389
6. \$ conda activate py389
7. \$ conda install spyder
8. \$ pip install PyQt5==5.10
9. \$ pip install pyqtwebengine==5.12
10. \$ pip install pyqt5==5.12

■ vi 편집기의 복사/붙여넣기 명령어

1. yy : 하나의 행을 복사
2. p : 붙여넣기
3. yG : 현재행부터 파일 끝까지 복사
4. :1,2 co 3 : 1~2행을 3행 다음으로 복사
5. :1,2 m 3: 1~2행을 3행 다음으로 이동

- 옮기기

```
$ su -  
Password : 1234 치고 (안보임)  
# cd /root/Desktop  
# ls  
#cp *.txt /home/scott/
```

하기

```
#exit 해서  
# scott으로 들어감
```

권한 올리기 chmod 777 *.txt

-rwxrwxrwx 나오면 성공한것!!

■ vi 편집기내에서 특정 문자를 검색하는 방법

문법 : :/검색할문자

예 : \$ vi jobs.txt

: /about 엔터치기

n을 누르면 전진하면서 다음 about을 검색

shift + n을 누르면 후진하면서 이전 about을 검색

n을 누르면 그 다음것이 검색이 된다. shift+n누르면 뒤로 간다.

■ vi 편집기 명령어로 문자를 변경하는 방법

문법 : s/기존문자/ 변경할 문자/g

예제 : KING을 aaa로 변경하시오!

```
$ vi emp.txt
```

: s/KING/aaa/g 엔터 KING이란 글씨는 다 바꿔줌

■ 모든 데이터만 다 변경하는게 아니라 하나만 변경하고자 할 때

문법: \$ vi 파일명.txt

: s/기존문자/변경할 문자

설명 : 지금 커서가 있는 현재행의 기존문자를 변경할 문자로 변경하겠다.

예제 : \$ vi emp.txt

: s/SALESMAN/aaaaaa

커서를 댄 그 하나의 문자만 변경이 된다.

- swap file 떠서 vi로 연 파일이 제대로 실행이 안 될때:

\$ rm .emp10.txt.swp (해당파일)

그리고 다시

vi emp*.txt

■ 권한 관리 명령어

리눅스에 하둡을 설치하고 운영을 할 때 여러가지 문제들이 발생하는데 그 중 권한에 관련한 오류들이 가장 많다. 그래서 하둡운영을 원활하게 하기 위해서는 권한관리 명령어를 잘 숙지하고 있어야 합니다.

* 권한관리 명령어 3가지

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

* 권한 관리표

번호	권한	대표문자	파일	디렉토리
1	읽기권한	r	읽고, copy	디렉토리에서 ls 가능
2	쓰기권한	w	수정	디렉토리에서 파일생성 가능
3	실행권한	x	실행	디렉토리에서 cd로 접근 가능

* ls -l로 어떤 특정파일을 조회했을 때 나오는 권한 부분 해석

\$ ls -l emp.txt

-rwxrwxrwx. 1 scott scott 1032 Jan 4 15:08 emp.txt

- rwx	rwx	rwx
↑	↑	↑
파일의 소유자의 권한	파일의 그룹에 속한 유저들의 권한	기타 유저들의 권한
↑		
디렉토리인지 아닌지 디렉토리면 d라고 나옵니다.		
- 면 파일입니다.		

g : 그룹유저

o : 기타유저

■ 리눅스 수업 목차

1. 리눅스란 무엇인가? 리눅스를 왜 배워야 하는가?
답 : 하둡을 이용하기 위해서, 딥러닝 환경을 구축하기 위해서

데이터 분석가 < 데이터 사이언티스트

1. 데이터 분석
2. 환경구축 (리눅스 설치, 하둡 설치, 아나콘다 설치)
3. 파이썬으로 데이터 분석 코딩
2. 리눅스 기본 명령어
3. 리눅스에 아나콘다 설치, 가상환경 만들기, spyder설치
4. 리눅스 vi 편집기
5. 리눅스 권한관리 명령어
 - a. chmod : 특정파일의 권한을 조정하는 명령어
 - b. chown : 특정파일이나 디렉토리의 소유자를 변경하는 명령어
 - c. chattr : 루트 유저만 권한을 조정할 수 있도록 설정하는 명령어

■ chown 명령어

"파일이나 디렉토리의 소유자를 변경하는 명령어"

예 : emp.txt의 소유자를 확인하는 방법

\$ ls -l emp.txt

```
-rwxrwxrwx. 1          root      root 1032 Jan 4 15:08 emp.txt
      ↑      ↑      ↑      ↑      ↑      ↑      ↑
    권한  바로가기개수 소유자 그룹명 파일크기 수정날짜 파일명
```

예: root로 접속해서 emp.txt의 소유자를 scott으로 변경한다.

```
$ su -
# pwd
# cd/home/scott
# ls-l emp.txt (소유자가 root)
# chown scott:scott emp.txt
      ↑      ↑
    유저명    그룹명
```

```
# ls -l emp.txt
```

설명 : 그룹명이란 권한관리를 쉽게 하기 위해서 그룹을 생성해서 그룹으로 권한관리를 한다. 예를 들면 다음 카페에서 우리반 학생들은 우수회원으로 다 올렸는데 우수회원이라는 그룹에 포함되어있으면 다음 카페 게시판에 우리반 게시판은 뭐든지 다 볼 수 있는 권한이 생긴다.

■ 숫자로 권한을 변경하는 방법

* 권한 관리표

번호	권한	대표문자	파일	디렉토리
4	읽기권한	r	읽고, copy	디렉토리에서 ls 가능
2	쓰기권한	w	수정	디렉토리에서 파일생성 가능
1	실행권한	x	실행	디렉토리에서 cd로 접근 가능

예 : emp.txt의 권한을 유저, 그룹, 기타로 모두 읽을 수만 있도록 변경하시오!

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

```
-r--r--r--. 1 scott scott 1032 Jan 4 15:08 emp.txt
```

예 : 위의 작업을 숫자로 하는 방법

```
$ chmod u-rwx,g-rwx,o-rwx emp.txt  
$ ls -l emp.txt  
$ chmod 444 emp.txt
```

설명 : 읽기는 숫자 4, 쓰기는 숫자2, 실행은 숫자 1입니다.

■ 디렉토리의 소유자의 권한을 변경하는 방법

디렉토리의 소유자의 권한을 변경하게 되면 그 디렉토리에 있는 모든 파일들의 소유자를 한 번에 변경할 수 있습니다.

예제1. scott 유저에서 /home/scott 밑에 test 500 디렉토리를 생성합니다.

```
$ cd  
$ pwd  
$ mkdir test500
```

예제2. test500 디렉토리 밑에 emp.txt와 dept.txt를 복사합니다.

```
$ cd  
$ cp emp.txt ./test500/  
$ cp dept.txt ./test500/
```

예제3. test500 디렉토리의 소유자가 누구인지 확인합니다.

```
$ ls -ld test500  
drwxrwxr-x/ 2 scott scott 37 Ja n5 11:12 test500
```

예제4. test500 디렉토리의 소유자를 root로 변경합니다.

```
$ su -
```

```
# whoami  
# pwd  
# cd /home/scott  
# ls -ld test500  
# chown root:root test500  
# cd test500  
# ls -l
```

설명 : test500 디렉토리의 소유자는 root로 변경했지만 test500 디렉토리 안에 emp.txt와 dept.txt의 소유자는 scott으로 그대로 유지됨

예제5. 만약 test500 디렉토리의 소유자를 root로 변경하면서 test500 디렉토리에 있는 모든 텍스트 파일의 소유자도 같이 root로 변경하고 싶다면?

```
# cd /home/scott  
# chown -R root:root test500
```

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

■ chattr 명령어

chattr 명령어를 이용하면 chmod 명령어 수행을 막을 수 있습니다.

chattr 명령어는 최상위 계정인 root에서만 수행할 수 있습니다.

scott 유저로 특정 파일에 대해서 chmod 명령어를 수행하지 못하도록 막는 명령어

예제1: \$ su -

```
# cdd /home/scott  
# chattr +i emp.txt  
# lsattr emp.txt  
-----emp.txt
```

설명 : emp.txt에 대해서 root 유저 외에 다른 유저는 삭제, 변경, 내용추가가 불가능하게 된다.

```
# su - scott  
$ ls -l emp.txt  
$ chmod 777 emp.txt  
chmod : changing permissions of 'emp.txt':
```

설명 : 소유자가 자기 자신인 scott임에도 불구하고 chmod 명령어가 수행되지 않습니다.

예제2. 그러면 다시 scott이 emp.txt를 chmod 할 수 있도록 설정하시오.

```
$ su -  
# cd /home/scott  
# chattr -i emp.txt  
# lsattr emp.txt  
# su - scott  
$ chmod 777 emp.txt
```

설명 : chattr 명령어는 언제 유용한가? 이 파일은 우리 회사에 너무나도 중요한 파일이어서 root만 변경할 수 있고 나머지 유저는 읽기만 할 수 있도록 싶을때 유용합니다.

■ 리눅스 수업 목차

- 1 . 리눅스를 왜 배워야 하는지 ?
- 2 . 리눅스 설치
- 3 . 리눅스 기본 명령어
- 4 . 리눅스에 아나콘다 설치
- 5 . 리눅스 v i 편집기 명령어
- 6 . 리눅스 권한 관리 명령어
- 7 . 리눅스 디스크 관리 명령어
8. 프로세스 관리 명령어
 1. ps 명령어
 2. top 명령어
 3. kill 명령어
 4. jobs 명령어
9. 쉘스크립트 작성법

■ 리눅스 디스크 (d i s k) 관리 명령어

- * 디스크 관리 명령어 3 가지
- 1 . d f 명령어
 - 2 . d u 명령어
 - 3 . s a r 명령어

■ 1. df명령어

" 현재 파일 시스템의 총 사용율을 확인하는 명령어 "

예제 : \$ df -h

■ 2. du명령어

" 현재 파일 / 디렉토리의 디스크 사용량을 표시하는 명령어 "

예제 : 현재 디렉토리의 텍스트 파일들의 총 크기를 구하려면 ?

```
$ du * .txt  
$ du -c * .txt
```

설명 : 합계를 확인하려면 -c 옵션을 붙인다 .

■ 3. sar명령어

만약 작업하다가 리눅스 서버가 너무 느려서 작업 진행이 잘 안될때 디스크의 사용률을 확인하는 명령어

예제: \$ sar 1 100

설명 : %user 부분을 집중적으로 모니터링 하면 됩니다.

리눅스 서버에 부하가 심한 작업을 수행하면 %user 부분에 사용율이 100에 가깝게 올라갑니다.

어떤 작업들이 부하를 주는 작업인가?

1. 악성 SQL을 수행
2. 무한루프 수행

* sar 명령어 결과의 컬럼 소개

%user : scott유저와 같이 일반 유저가 사용하는 disk i/o입니다.

악성 SQL을 수행거나 무한루프를 돌리면 %user 사용률이 올라갑니다.

%nice: cpu를 양보하는 친절도

%system : system이 사용하는 disk i/o

%iowait: i/o를 일으키면서 얼마나 대기하는지

%idle : 작업을 안하고 있는 idle한 상태

%steal : 다른 프로세서의 자원을 얼마나 뺏고 있는지

ls -l : 그냥 리스트를 보여달라~

ls -rlt : 리스트를 보여주는 생성한 시간 순서대로 보여달라

■ jobs 명령어

동작중인 작업의 상태를 확인하는 명령어

- 상태정보 4가지
1. running : 실행중
 2. stopped : 일시중단중
 3. Done : 종료
 4. terminated : 강제 종료됨

예제: \$ vi hhh.txt

```
select ename, sal, job  
  from emp  
 where
```

esc 키를 누르고 ctrl+z를 눌러서 하던 작업을 취소합니다.

[1]	+	stopped	vim hhh.txt
↑	↑	↑	↑
job 번호	현재 진행중이었던 job	일시 중단중	

\$ jobs : 동작중인 작업의 상태를 확인하는 명령어

\$ fg : 현재 작업중이었던 작업들을

■ ps 명령어

" 현재 시스템에서 수행되고 있는 프로세서의 정보를 표시하는 명령어"

문법 : \$ ps 옵션 프로세서번호

예제 : \$ ps -p 12006

결과:

PID	TTY	TIME	CMD
[4]	Done	sar 1 100 >> sar_2021.txt	

옵션:

- p : 프로세서 아이디
- e : 현재 실행중인 모든 프로세서
- f : 실제 유저명, 개시시간 등을 표시
- l : 프로세서의 상태, 우선도 등과 같은 상세한 정보를 표시

■ kill 명령어

" 프로세서를 죽이는 명령어"

```
$ ps -ef | grep vim
```

```
scott 11701 9287 0 14:48 pts/0 00:00:00 vim hhh.txt  
scott 11866 9287 0 14:57 pts/0 00:00:00 vim hhh4.txt
```

```
$ ps -ef | grep vim 해서 위에껄 알아냄
```

```
$ kill -9 11701  
$ kill -9 11866
```

옵션: -9를 사용하면 강제로 죽이겠다는 뜻

■ top 명령어

"지금 현재 작동중인 프로세서들의 cpu 사용율과 메모리 사용율을 확인하는 명령어"

내가 돌린 파이썬 프로그램이 cpu를 얼마나 많이 사용하고 있는지 확인하고 싶을때나 내가 잘못 프로그래밍해서 무한루프를 돌렸거나 리눅스의 자원을 많이 사용하고 있다면 문제가 발생하는 것이므로 top 명령어로 확인해 볼 필요가 있다.

```
$ top
```

그리고 q눌러 취소하기

설명 : top에서 집중적으로 봐야할 부분은 cpu 사용율이 높은 프로세서가 무엇이고 어떤 작업을 하고 있는지 알아내야 합니다.

■ 9. 쉘스크립트 작성법

➤ 쉘 (shell) 이란 무엇인가?

shell이란 운영체제에서 제공하는 명령을 실행하는 프로그램입니다.

➤ 쉘 (shell) 스크립트란?

인터프리터(통역사) 역할을 하는 것으로 시스템에서 지원하는 명령어들의 집합을 묶어서 프로그램화 한것을 말한다.

➤ 쉘 (shell)의 종류

1. Bourne shell
2. C shell
3. korn shell
4. bash shell : 가장 많이 사용하는 쉘

쉘스크립트 작성시에 맨 위에다가 지금부터 작성하는 shell 스크립트 문법은 bash shell이다라고 명시하고 프로그래밍을 합니다.

!/bin/bash ----> 쉘 중에 bash 쉘을 쓰겠다.

쉘 프로그래밍 작성

➤ 쉘 스크립트 프로그래밍이란?

1. c언어와 유사한 프로그래밍
2. 변수, 반복문(loop문), 제어문(if문)이 사용가능
3. 별도로 컴파일을 하지 않고 텍스트 파일 형태로 바로 실행이 가능합니다.
4. vi로 작성이 가능합니다.
5. 리눅스의 많은 부분이 쉘 스크립트로 작성이 되어있습니다.

➤ 쉘 스크립트를 작성하고 실행하는 방법

\$ vi a.sh

echo "yahoo~~~~~"

\$ sh a.sh

쉘 스크립트를 실행하는 또 다른 방법

\$./a.sh

bash: ./a.sh: Permission denied

위와 같이 에러가 납니다. 그래서 아래와 같이 권한을 확인해야 합니다.

\$ ls -l a.sh

-rw-rw-r--. 1 scott scott 22 Jan 5 16:09 a.sh

소유자에게 실행권한을 넣어 줍니다.

\$ chmod 764 a.sh

\$ ls -l a.sh

\$./a.sh

설명 : ./파일명.sh로 실행하려면 실행권한이 있어야 합니다.

sh 파일명.sh는 파일의 실행권한이 없어도 실행됩니다.

➤ 변수 사용법

1. 모든 변수는 문자열(string)로 취급됩니다.

2. 변수 이름은 대소문자를 구분합니다.
3. 변수에 값을 대입할 때는 '=' 좌우에 공백이 없어야 합니다.
4. 변수에 들어간 문자를 출력하려면 변수 앞에 \$를 붙이고 echo 명령어로 출력하면 됩니다.

예 : \$ myvar="Hi~~~~~"

```
$ echo $myvar
```

➤ 변수의 숫자 계산하는 방법

1. 변수에 대입한 값은 모두 문자열로 취급이 됩니다.
2. 변수에 들어있는 값을 숫자로 해서 사칙연산(+-*/)을 하려면 expr을 사용해야 합니다.
3. 수식에 괄호 또는 곱하기(*)를 사용하려면 그 앞에 반드시 역래쉬(\$)를 붙여야 합니다.

예제 : \$ num1=100

```
$ num2=200
```

```
$ echo $num1
```

```
$ echo $num2
```

```
$ echo $num1 + $num2 >>> 100+200 이렇게 나옴
```

그래서

```
$ expr $num1 + $num2 이렇게 해줘야 함.
```

■ 파라미터 변수

1. 파라미터 변수는 \$0, \$1, \$2,...의 형태를 가집니다.
2. 전체 파라미터는 \$*로 표현합니다.

예 : \$ vi add.sh

```
num1=$1  
num2=$2  
num3=`expr $num1 + $num2`  
echo "$num1 와 $num2 를 더하면 $num3 입니다"
```

```
$ sh add.sh 24 18
```

결과:

24 와 18 를 더하면 42 입니다

설명 : 역따옴표를 써야 역따옴표 안이 실행문이 실행이 되어서 num3 변수에 할당이 되어집니다.

변수='리눅스 명령어'

리눅스 명령어에 의해서 수행된 결과가 변수에 입력되어야 한다면 역따옴표를 사용해야 합니다.

- 학원 컴퓨터를 원격으로 접속해서 사용하려면?
1. 학원 나오는 날 학원에 나와서 크롬 원격 지원 프로그램을 설치한다.
 2. 원격으로 학원 컴퓨터에 접속이 되는지 확인한다.

■ 리눅스 수업 목차

1. 리눅스를 왜 배워야 하는지 ?
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. 프로세스 관리 명령어
 - a. ps 명령어
 - b. top 명령어
 - c. kill 명령어
 - d. jobs 명령어
9. 스크립트 작성법

■ if 조건문에 들어가는 비교 연산자

- 문자열 비교

1. "문자열1"=="문자열2" : 두 문자열이 같으면 True ('=' 양옆으로 딱 붙이기)
2. "문자열1"!="문자열2" : 두 문자열이 같지 않으면 True

- 숫자열 비교

1. 숫자1 -eq 숫자2 : 두 숫자가 같으면 True (=)
2. 숫자1 -ne 숫자2 : 두 숫자가 같지 않으면 True (!=)
3. 숫자1 -gt 숫자2 : 숫자1이 숫자2보다 크다면 True (>)
4. 숫자1 -ge 숫자2: 숫자1이 숫자2보다 크거나 같다면 True (>=)
5. 숫자1 -lt 숫자2: 숫자1이 숫자2보다 적으면 True (<)
6. 숫자1 -le 숫자2: 숫자1이 숫자2보다 작거나 같으면 True (<=)
7. !숫자1 : 숫자1이 거짓이라면 True (not)

예제: (띄어쓰기 주의!!)

\$ vi if1.sh

```
if [ 100 -eq 200 ]; then
    echo "100과 200은 같습니다."
else
    echo "100과 200은 같지 않습니다."
fi
```

저장하고 나와서

```
$ sh if1.sh
```

결과:

100과 200은 같지 않습니다.

if로 시작했으면 fi로 반드시 끝나야 한다.

■ 리눅스의 논리 연산자

1. and ---> && 또는 -a
2. or ---> || 또는 -o
3. not ---> !

예제: if [\$sal -lt 2000] && [\$job=="SALESMAN"]; then
또는
if [\$sal -lt 2000 -a \$job=="SALESMAN"]; then

■ 리눅스 쉘에서 loop문 사용법

1. for loop 문 사용법

```
for 변수 in 값1, 값2, 값3
do
    반복할 문장
done
```

예제: \$ vi for1.sh

```
for i in {1..10}
do
    echo $i
done
```

```
$ sh for1.sh
```

결과:

1
2
3
4
5
6
7
8
9
10

■ 이중 루프문

```
for i in {2..9}
do
    for k in {1..9}
    do
        echo "$i $k"
    done
done
```

- 아프리카tv에서 딥러닝 연구원(데이터 분석가)이 하는일?
1. 아프리카 tv 채팅창의 대화들을 하둡에 저장다.
 2. 대화들을 hive를 이용해서 검색을 한다.
 3. 혹시 대화에서 "약"또는 불건전한 대화를 하면 바로 방폐쇄를 시키게끔 하는 것을 자동화한다.

■ 리눅스 쉘을 현업에서 어떻게 활용하는가?

라이나 생명에서 데이터 분석을 했던 사례

1. 고객 데이터를 리눅스 쉘을 이용해서 데이터를 필터링 한다. (리눅스 쉘)
예 : 라이나 생명 고객중에 40대만 따로 분리해서 text파일을 생성한다.
2. 40대의 데이터를 가지고 군집분석(k-mass)을 한다. :R과 파이썬
3. 40대의 데이터를 가지고 연관분석(아프리오 알고리즘) R과 파이썬

새로운 보험 상품이 출시 되었는데 보험 가입 유도 시 모든 보험 가입자들에게 모두 연락해서 가입을 유도하는 것보다 보험 가입이 가능할 것 같은 고객들만 필터링해서 연락하는게 더 효율적이다.

■ 하둡 수업

1. centos 리눅스를 새로 생성합니다.
가상 환경 이미지 이름: centos_2

1/7 (하둡시작)

2021년 1월 7일 목요일 오전 9:50

■ 하둡목차

1. 하둡이란 무엇인가?
2. 하둡설치

■ 1. 하둡이란 무엇인가?

- **하둡(hadoop) ?**

대용량 데이터를 분산 처리할 수 있는 자바 기반의 오픈 소스 프레임워크로서, 하둡은 분산 파일 시스템인 HDFS(Hadoop Distributed File System)에 데이터를 저장하고, 분산처리시스템인 멘리듀스를 이용해 데이터를 처리한다.

- 현업에서 데이터 분석가들이 하둡을 어떻게 활용하는가?

데이터의 종류?
1. 정형화 된 데이터 : emp와 같이 컬럼과 row로 이루어진 rdbms에 저장되는 테이블 형태의 데이터

2. 반정형화된 데이터 : 웹로그와 sns 데이터, html, json
3. 비정형화된 데이터 : 동영상이나 이미지 데이터, 텍스트 데이터

RDBMS	vs	비 rdbms 제품
Oracle		하둡 (무료)
mssql		스칼라
maria db (무료)		
postgreSQL		

라이나 생명 데이터 분석가

1. 히든에 고객 데이터(정형화 된 데이터)를 저장하고 나이대별로 분리해서 csv파일을 생성합니다.
2. 나이대별로 분리한 csv 파일 데이터를 가지고 군집분석
3. 나이대별로 분리한 csv 파일 데이터를 가지고 연관분석

아프리카 tv 데이터 분석가

1. 채팅 데이터를 다 하둡에 저장한다.
2. 특정 단어를 자주 사용하는 채팅방의 형태를 분석한다.
3. 머신러닝을 사용해서 가공한 채팅 데이터를 학습시키고 특정 단어를 사용하는 사람은 불건전한 대화를 이끌 가능성이 높다는 것을 예측하고 채팅방을 폐쇄시킨다. (자동)

* 하둡 유래 배경지식?

구글에서 구글에 쌓여지는 수많은 빅데이터들을 처음에는 RDBMS(오라클)에 입력하고 데이터를 저장하려고 시도를 했으나 너무 데이터가 많아서 실패를 하고 자체적으로 빅데이터를 저장할 기술을 개발을 했다. 그리고 대

외적으로 이 기술에 대한 논문을 하나 발표했다.

웹페이지의 글들을 오라클에 입력하려면 테이블에 입력할 수 있는 insert 문장으로 만들어서 입력을 해야하는데 그렇게 만들 수가 없어서 다른 기술을 개발했다.

그 논문을 더그커텁(하둡을 만든이)이라는 분이 읽고 자바로 구현했다. 하둡은 자바로 만들어졌다. 자바를 몰라도 하둡의 데이터를 쉽게 다룰 수 있도록 NoSQL을 제공해서 SQL과 같은 언어로 쉽게 빅데이터를 다룰 수 있게 되었다. ----> Hive (별떼)

♣ 빅데이터 시험

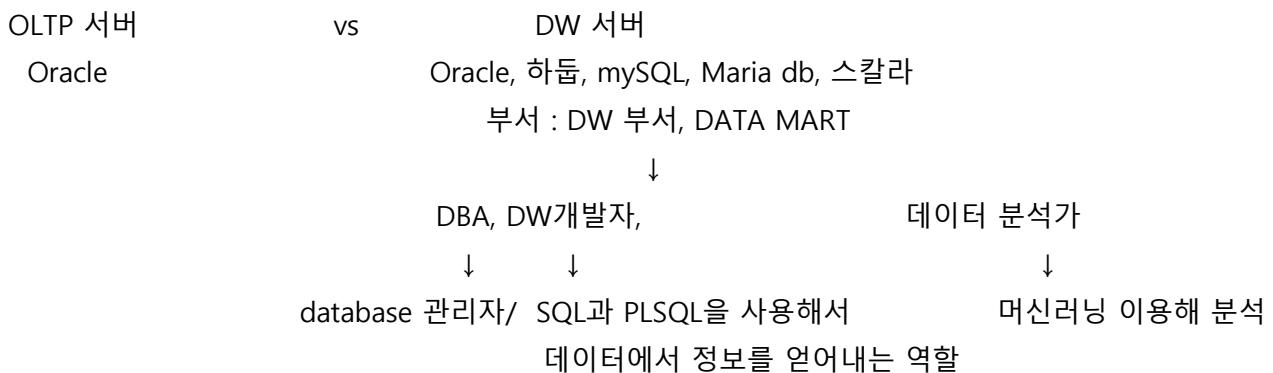
NoSQL ? (Not Only SQL)

NoSQL은 전통적인 RDBMS와 다른 DBMS를 지칭하기 위한 용어입니다. 자바를 몰라도 활용할 수 있는 프로그램 입니다.

예: 1. Hive -----> SQL과 똑같다.

2. Pig -----> SQL과 다름

3. Mongo db -----> SQL과 다름



더그커텁이 새로 만든 프로그램 이름을 뭐로 할까하다가 자신의 아이가 코끼리를 보고 하둡? 이라고 하는 것을 들었다. 그래서 이름이 하둡이 되었다.

그래서 그 뒤로 hadoop을 편하게 이용할 수 있도록 개발한 모든 하둡 생태계의 개발 프로그램 이름들이 다 동물이름으로 지어지게 되었다.

hadoop -----> Hive(별떼)

Pig (돼지)

Mongo db

Tajo (타조)

* 하둡의 장점?

"공짜이다"

"분산처리가 가능하다"

여러대의 노드(컴퓨터)를 묶어서 마치 하나의 서버처럼 보이게하고 여러노드의 자원을 이용해서 데이터를 처리

하기 때문에 처리하는 속도가 아주 빠른 장점이 있다.

예 : 우리반 컴퓨터 한대의 메모리 - 8기가

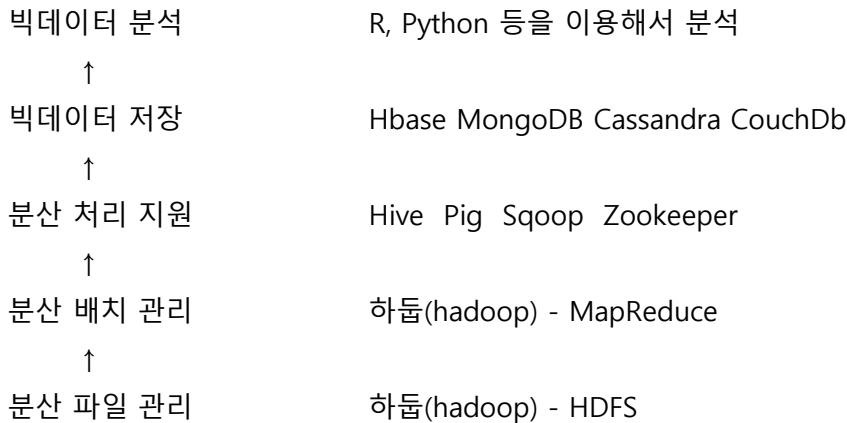
컴퓨터 30대를 다 묶어서 분산 처리 파일 시스템을 구성했으면 마치 30대의 컴퓨터를 하나의 컴퓨터처럼 보이게 할 수 있다. (240기가의 메모리를 사용할 수 있게 되고 30개의 cpu를 다 사용할 수 있게 된다.)

이걸 지원하는 것이 하둡이다.

예 : 한대의 서버로 1테라 바이트의 데이터를 처리하는데 걸리는 시간이 2시간 반이 걸린다고 하면 여러대의 서버를 병렬로 묶어서 작업하면 2분 내에 데이터를 읽을 수 있다.

예 : 2008년 뉴욕타임즈는 130년 분량의 신문기사 1100만 페이지를 하둡을 이용해서 하루만에 pdf로 변환했다. 이 때 든 비용이 200만원 뿐이다. 만약 하둡이 아닌 일반 서버로 처리했다면 14년이 걸렸을 것으로 예상된다.

■ 하둡 생태계 시스템



* 하이브(Hive) 사용 예

Hive> select count(*) from emp; -----> java로 자동변환해서 결과를 출력해준다.
14

야후의 경우 약 5만대의 서버(컴퓨터)를 연결해서 하둡을 운영하고 있고 페이스북은 약 1만대 이상의 하둡 클러스터를 이용하고 있다.

- **하둡의 장점** : 저렴한 구축비용과 비용대비 빠른 데이터 처리

- **하둡의 단점** : 무료다 보니 유집보수가 어렵다.

네임노드가 다운되면 고 가용성이 지원이 안된다.

한번 저장된 파일은 수정할 수가 없다.

기존 데이터에 append는 되는데 update가 안된다.

>>> emp.csv의 내용을 update문으로 update 할 수 없다.

emp.csv의 내용을 변경하려면 emp.csv를 직접 열어서 수정해야 합니다.

여러대의 컴퓨터 中 한 대가 네임노드, 나머지는 데이터 노드들

네임노드 : 메타 데이터를 저장하고 있다.

메타 데이터 : emp.csv가 00이 자리 컴퓨터에 있다라는 걸 알려주는 주소 정보 (인덱스 같은거)

* 하둡의 주요 배포판

리눅스도 centos, redhat, ubuntu가 있는 것처럼 하둡도 배포판이 있다.

1. Cloudera 업체에서 나온 CDH <---- 가장 신뢰 높음
2. Hortonworks 에서 나온 HDP
3. 아마존에서 나온 EMR(Elastic Mapreduce)
4. Hstreaming에서 나온 Hstreaming

• 하둡 홈페이지

<https://hadoop.apache.org>

* 하둡 설치의 큰 그림

1. java 설치 ----> 하둡이 자바로 만들어져 있어서 자바를 설치해야 함.
2. keygen 생성 ----> 여러 노드들을 묶어서 마치 하나의 컴퓨터처럼 보이게 하는게 하둡의 목표이므로 다른 컴퓨터로 접속할 때 패스워드를 치지 않아도 접속될 수 있게 해야해서 설치
3. 하둡 설치 ----> 4개의 파일의 내용만 수정

* 하둡 설치에 앞서서 먼저 해야할 일

1. 현업에서처럼 리눅스 서버에 putty와 같은 접속 프로그램을 이용해서 접속하도록 설정
카페 리눅스 게시판에 putty로 리눅스 서버에 접속하는 방법에서 나온 블로그대로 구성을 하고 구성을 끝냈으면 putty프로그램 다운로드를 받고 접속을 합니다.

<https://atoz-develop.tistory.com/entry/%EB%A6%AC%EB%88%85%EC%8A%A4-%EA%B0%80%EC%83%81%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>

내자리 컴퓨터(노트북) -----> 서버실(리눅스 컴퓨터)
putty <-- 서버

항상 접속할 때는 scott 으로 접속하고 root 에서 작업할 때는
scott 으로 접속한 상태에서 터미널 창에서 su - 로 스위치 유저하여 작업합니다

1. 하둡을 설치하기 위해서 필요한 파일

아래의 설치 파일을 공유폴더인 c 드라이브 밑에 data 밑에 둡니다.

1. jdk-7u60-linux-i586.gz
2. hadoop-1.2.1.tar.gz

3. protobuf-2.5.0.tar.gz

2. 네트워크 설정을 변경합니다.

리눅스 서버의 아이피주소를 확인하는 명령어인 ifconfig를 수행합니다. 10.0.2.15 가 나오면 인터넷 연결이 된 것

* 리눅스의 네트워크 설정 부분의 스크립트들이 모아져 있는 곳으로 이동합니다.

```
[root@localhost ~]# cd /etc/sysconfig/network-scripts  
[root@localhost ~]# ls
```

설명 : ifcfg-enp0s3은 인터넷 연결을 위한 네트워크

ifcfg-enp0s8은 다른 서버가 내 서버에 접속하기 위해 ip 주소를 셋팅하는 네트워크
이 두가지가 중요!!

```
# ifcfg-enp0s8 설정  
[root@localhost network-scripts]# vi ifcfg-enp0s8
```

```
ONBOOT=yes  
NM_CONTROLLED=yes  
BOOTPROTO=none  
NETMASK=255.255.255.0  
IPADDR=192.168.56.101
```

>>> 자동으로 인터넷을 켜지게 하기

설명: 192.168.56.101은 다른 컴퓨터에서 내 리눅스 컴퓨터로 접속할 때 사용하는 ip 주소입니다.

3. hostname 을 변경합니다

설명 : hostname은 컴퓨터 이름입니다.

```
# hostname 설정
```

설명 : 위에서 설정한 아이피 주소와 함께 hostname을 centos로 하겠다고 지정합니다.

```
[root@localhost network-scripts]# vi /etc/hosts
```

```
192.168.56.101 centos
```

```
# hostname 변경(세션을 다시 시작하면 변경 정보 인식)
```

```
[root@localhost network-scripts]# hostname  
localhost.localdomain
```

```
[root@localhost network-scripts]# hostnamectl set-hostname centos
```

* 네트워크 서비스를 재시작 합니다.

설명 : 네트워크 구성을 변경했으므로 변경된 내용을 반영하려면 다시 네트워크를 재시작 해야합니다.

```
[root@localhost network-scripts]# service network restart  
[root@localhost network-scripts]# ifconfig
```

4. 방화벽을 해지 합니다.

```
## 방화벽 해지
```

```
[root@centos Desktop]# iptables -F  
[root@centos Desktop]# iptables -L
```

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 유져하여 작업합니다.

6. ld-linux.so.2 를 설치 합니다.

설명 : ld-linux.so.2를 설치해야 하는 이유는 java를 설치할 때 필요한 파일이기 때문입니다.

ld-linux.so.2 설치 파일은 따로 필요없고 이미 리눅스 서버에 내장되어 있어서 불러와서 설치하면 됩니다.

```
[scott@centos ~]$ su -
```

```
[root@centos ~]# yum install ld-linux.so.2  
[root@centos ~]# reboot
```

7. 자바를 설치 합니다.

설명 : 자바를 설치해야하는 이유는 하둡을 자바로 만들었기 때문입니다.

```
[scott@centos ~]$ java -version  
openjdk version "1.8.0_262" (옛버전)
```

반드시 root로 접속!!

```
[scott@centos ~]$ su -
```

설명 : 자바를 설치할 디렉토리를 만들고 그 곳에 jdk 파일의 압축을 풀면 자바설치 끝 입니다.

```
[root@centos ~]# mkdir -p /u01/app/java
```

설명 : 위의 디렉토리에 jdk 파일을 mv로 이동해 놓습니다.

```
[root@centos ~]# mv /home/scott/jdk-7u60-linux-i586.gz /u01/app/java/  
[root@centos ~]# cd /u01/app/java/  
[root@centos java]# tar xvzf jdk-7u60-linux-i586.gz
```

설명 : 압축을 풀면 jdk1.7.0_60 이라는 디렉토리가 생기는데 이 디렉토리의 소유자를 root 유져로 변경합니다.

```
[root@centos java]# chown -R root:root jdk1.7.0_60  
[root@centos java]# exit
```

설명 : .bashrc 파일은 리눅스 환경설정 파일입니다. 이 파일에 적어준 내용대로 환경설정이 이뤄지는데 scott으로 로그인을 할 때마다 .bashrc 파일이 자동으로 수행이 되어집니다. 아래의 작업은 .bashrc 파일에 자바가 이 시스템에서 어디에 설치되어있다라는 위치를 알려주는 작업입니다.

vi 편집기 명령어로 .bashrc 파일을 열고 아래의 3줄을 맨 아래쪽에 붙여넣습니다. export 명령어가 수행되면서 JAVA_HOME은 /u01/app/java/jdk1.7.0_60 위치라는 것을 명시하게 되는 것입니다.

```
[scott@centos ~]$ vi .bashrc
=====
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
=====
```

설명 : 위에서 복사해서 붙여넣었으면 설정한 내용을 리눅스 서버에 반영을 해야합니다.

카카오톡 환경설정에서도 설정을 변경했으면 반영을 할 때 apply 같은 버튼을 누르듯이 리눅스에서 반영을 하려면 .bashrc 파일을 실행을 하면 되는데 실행하는 명령어가 아래와 같이 점(.)을 쓰고 한칸 띄우고 .bashrc 하면 됩니다.

8. protobuf 를 설치 합니다.

설명 : protobuf를 설치를 해야하는 이유는 하둡을 사용하려면 여러대의 컴퓨터가 서로 접속을 자유롭게 하면서 패스워드를 물어보지 않고 그냥 자유롭게 서로 접속을 할 수 있어야 합니다. 이 프로그램은 구글에서 만들었고 이 프로그램 설치를 해야 서버끼리 자유롭게 접속할 수 있는 상태가 됩니다.

다시 root 로 접속합니다.

```
[scott@centos ~]$ su -
```

```
[root@centos ~]# cp -v /home/scott/protobuf-2.5.0.tar.gz /usr/local
[root@centos ~]# cd /usr/local
[root@centos local]# ll
```

```
[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
```

```
[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
```

다시 reboot 합니다.

9. keygen 생성

하둡은 SSH 프로토콜을 이용해 하둡 클러스트간의 내부 통신을 수행한다.



컴퓨터가 네트워크로 대화를 나눌 때 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.
이때 ssh 를 이용할 수 없다면 하둡을 실행할 수 없다.

네임노드에서 공개키를 설정하고 이 공개키(authorized_keys) 를 다른 데이터 노드에 다 복사해줘야한다.

예 : 우리반 30명 자리의 컴퓨터를 하둡으로 묶어버리고 싶다면 30명 자리의 패스워드를 모두 공개키(authorized_keys) 에 저장하고 모든 자리에 authorized_keys를 배포 해 줘야 합니다.

이 키를 가지고 있으면 ssh 로 다른 노드에 접속할때 패스워드 없이 접속할 수 있다.

공개키 생성

설명 : 숨김 디렉토리인 .ssh 디렉토리를 우리가 생성할텐데 혹시 있을지 몰라서 삭제를 먼저 합니다.
[scott@centos ~]\$ rm -rf .ssh

설명 : ssh-keygen 명령어를 이용해서 .ssh 디렉토리를 생성을 합니다.

```
[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
Enter same passphrase again: <- enter key
```

설명 : 아래의 명령어로 scott으로 192.168.56.101 서버에 접속을 시도합니다. 즉 내 리눅스 서버로 접속하면서 패스워드를 물어볼텐데 그 때 패스워드를 입력하면 그 패스워드가 authorized key에 담겨서 만들어지게 됩니다.

```
[scott@centos ~]$ ssh-copy-id -i /home/scott/.ssh/id_rsa.pub scott@192.168.56.101
```

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 안됩니다.

■ 어제 배웠던 내용 복습

1. **하둡이란 무엇인가?** 자바로 만든 분산 파일 시스템입니다.

하둡이 왜 필요한가? 빅데이터를 저장하기 위해 필요합니다.



정형화 데이터 : 컬럼과 로우로 이루어진 테이블

반정형화 데이터 : 웹로그, html, json

비정형화 데이터 : 동영상, 이미지, 텍스트

하둡, maria db ---> 무료

데이터 분석가의 일:

Ex)

네이버 영화 평점 댓글 수집 ---> 텍스트 ---> 하둡 ---> hive

데이터 검색과 데이터 전처리 ----> 파이썬, R을 이용하여 분석, 시각화

지방흡입 기계(센서)에서 발생하는 센서 데이터(웹로그 데이터)를 수집, 분석 해서 어떻게 지방 흡입을 해야 멍이 적게 드는지 그 방법을 연구하는 작업

2. 하둡 설치

- 리눅스 설치 후 putty로 접속할 수 있게 환경 구성
- 하둡 설치를 위해 필요한 3가지 파일 준비
 1. jdk 파일
 2. hadoop 설치파일
 3. prototype 파일

어제 하둡 설치 이어서!!

10. hadoop 디렉토리를 만들고 거기에 하둡설치 파일 압축을 풁니다.

```
[scott@centos ~]$ cd  
[scott@centos ~]$ mkdir hadoop  
[scott@centos ~]$ cd hadoop
```

설명 : 아래의 cp/hadoop-1.2.1.tar.gz . 명령어는 현재 디렉토리의 바로 전 상위 디렉토리에 있는 hadoop-1.2.1.tar.gz 파일을 현재 디렉토리에 복사하라는 뜻

```
[scott@centos hadoop]$ cp ..../hadoop-1.2.1.tar.gz .  
[scott@centos hadoop]$ tar xvzf hadoop-1.2.1.tar.gz  
...  
[scott@centos hadoop]$ rm hadoop-1.2.1.tar.gz
```

11. 하둡 홈 디렉토리를 설정한다.

설명 : 리눅스의 환경설정파일인 .bashrc 파일의 vi 편집기로 열어서 맨 아래에 HADOOP의 홈(집) 디렉토리가 어디다라고 지정을 해줘야 합니다.

```
[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 : PATH에는 리눅스에 설치되어져있는 소프트웨어가 어디에 설치되었다라고 등록하는 환경변수

설명 : .bashrc 명령어로 .bashrc 스크립트를 수행합니다. (적용버튼 누르는 것과 같음)

예: 카카오톡의 환경설정에 가서 환경설정 변경하고 적용을 누르듯이 아래의 명령어는 적용버튼을 누르는 명령어입니다.

```
[scott@centos ~]$ . .bashrc
```

12. 하둡 환경설정을 하기 위해 아래의 4개의 파일을 셋팅해야한다.

1. hadoop-env.sh : 자바 홈디렉토리와 hadoop 홈디렉토리가 어딘지 지정한다.

하둡이 자바로 만들어 졌기 때문에 자바가 어디 설치 되어져 있는지 하둡이 알고 있어야 하므로 홈 디렉토리를 hadoop-env.sh 파일에 지정한다.

2. core-site.xml : 하둡의 네임노드가 어느 서버인지를 지정한다.

하둡은 네임노드와 데이터 노드들로 구성이 되어져 있는데 네임노드에는 우리가 검색하고자 하는 데이터의 위치정보인 메타정보가 들어 있다.

3. mapred-site.xml : java 로 만들어진 mapreduce 프레임워크와 관련된 정보를 지정하는 파일

하둡은 자바로 만들어진 큰 2개의 함수로 되어져 있습니다. 그 2개는 mapping 함수와 reducing 함수 이다.

우리가 하둡에 있는 데이터를 검색하면 이 2개의 함수가 작동하면서 데이터를 검색하고 결과를 보여준다.

4. hdfs-site.xml : 하둡 파일 시스템인 HDFS(Hadoop Distributed File System) 와 관련된 정보를 저장하는 파일

여러대의 컴퓨터를 둑어서 하나의 강력한 슈퍼 컴퓨터로 사용하기 위해 반드시 필요한 하둡의 필수 세팅부분

```
[scott@centos ~]$ cd $HADOOP_HOME/conf
```

설명 : 4개의 파일을 수정만 하면 되는데 첫번째로 자바의 홈디렉토리와 하둡의 홈 디렉토리가 어디다라고 지정하기 위해 hadoop-env.sh를 수정한다.

```
[scott@centos conf]$ vi hadoop-env.sh  
=====  
# 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
```

=====

설명 : 하둡을 구성하기 위한 두 번째 파일인 core-site.xml 파일을 수정하는데 여기에는 네임노드와 데이터 노드에 대한 정보를 저장합니다. port - 건물의 복도 같은 것

```
[scott@centos conf]$ vi core-site.xml
=====
<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>
=====
```

설명 : 하둡의 핵심 함수 2개인 mapping 함수와 reducing 함수에 대한 코드가 있는 맵리듀싱 프레임워크에 대한 정보를 기재하는 mapred-site.xml을 수정합니다.

```
[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>
=====
```

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 것입니다. 그래서 아래와 같이 네임노드의 데이터가 축적될 위치와 데이터 노드의 데이터가 축적될 위치를 로컬(우리컴퓨터)에 생성합니다.

```
[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>
=====
data 와 name 디렉토리의 권한을 조정합니다.

```

```

[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
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. 하둡파일 시스템을 올린다.

```
[scott@centos ~]$ start-all.sh
```

```
...
```

```
stop-all.sh
```

15. 하둡파일 시스템의 데몬이 잘 올라왔는지 확인한다.

```

[scott@centos ~]$ jps
9167 SecondaryNameNode : 하둡 보조 네임노드로 네임노드의 파일 시스템 이미지 파일을 주기적으로
                           갱신하는 역할을 수행하는 노드
                           (메타 데이터를 최신것으로 유지시키는 작업)
9030 DataNode : HDFS의 데이터를 입력하면 입력 데이터는 32mb의 블럭으로 나눠져서 여러대의 데이터 노드에
                  분산되어 저장된다. 그 데이터를 저장하는 노드
9402 TaskTracker : 사용자가 설정한 맵리듀스 프로그램을 실행하는 역할을 하며 하둡 데이터 노드에서 실행되는 데몬
9250 JobTracker : 하둡 클러스터에 등록된 전체 잡(job)의 스케줄링(계획)을 관리하고 모니터링하는 데몬
9494 Jps : 현재 jps 명령어 수행한 프로세서(나)
8883 NameNode : HDFS의 모든 메타 데이터(data의 위치정보)를 관리하고 클라이언트가 hdfs에 저장된 파일에
                  접근할 수 있게 해준다

```

이렇게 6개가 나오면 성공~

16. 잘 안되었을 때의 조치 방법

1. 하둡을 모두 내립니다.

```
[scott@centos conf]$ stop-all.sh
```

2. 디렉토리 권한을 조정합니다.

```
(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]$
```

2. 하둡 네임노드를 포맷한다.

```
[scott@centos conf]$ cd  
[scott@centos ~]$ hadoop namenode -format
```

3. 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

"사바를 몰라도 rdbms에 익숙한 데이터 분석가들을 위해서 SQL을 이용해서 하둡의 멤피디싱 프로그램을 지원하는 프로그램"

페이스북에서 만든 오픈소스

* HiveQL

1. SELECT는 되는데 update와 delete 명령어는 지원 안함
2. from 절의 서브쿼리는 사용 가능
3. select 문 사용시 having 절은 사용 불가능
4. PL/SQL의 프로시저는 hive2.0부터 가능

↓

SQL + C 언어를 이용해서 프로그래밍

c 드라이브 밑에 data 폴더 밑에 둡니다.

1. hive 설치 파일의 압축을 푸다

\$ cd

```
$ cp /media/sf_data/hive-0.12.0.tar.gz /home/scott/  
$ tar xvzf hive-0.12.0.tar.gz
```

2. hive로 접속한다.

```
$ cd /home/scott/hive-0.12.0/bin  
$ ./hive
```

```
hive> show tables;
```

■ 하둡 시스템이 정상인지 확인하는 명령어

```
$ jps
```

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 잡(job)의 스케줄링(계획_)을 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 블럭으로 나눠져서 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
```

■ 오라클과 hive의 함수들의 다른점

오라클	vs	하이브
to_char(hiredate, 'RRRR')		year(to_date(hiredate))
to_char(hiredate, 'MM')		month(to_date(hiredate))
to_char(hiredate, 'DD')		day(to_date(hiredate))

영화 평점 관련 댓글 다운

1. 먼저 대용량 텍스트 파일의 압축파일을 다운로드 받는다.
\$ cd

```
$ wget http://files.grouplens.org/papers/ml-1m.zip
```

2. 위의 압축 파일의 압축을 해제합니다.
\$ unzip ml-1m
\$ cd ml-1m
\$ ls -lh : 파일 크기 볼 수 있음

<movies.dat, ratings.dat 이 빅데이터를 하둡에 저장하고 hive를 이용해서 SQL로 분석하고자 한다면?>

1. movies.dat, ratings.dat의 컬럼과 컬럼 구분을 콤마(,)가 되게끔 데이터 전처리를 해야됩니다. (리눅스 명령어)

예제 : \$ head 10 movies.dat

설명: 확인해보니 컬럼과 컬럼이 :: 로 구분되어져 있다.

\$ sed s:::/g movies.dat >> movies_coma.dat

\$ head 5 movies_coma.dat

설명 : sed 명령어를 이용해서 ::를 콤마() 변경을 함

보여지는 결과를 movies_coma.dat로 저장함

\$ 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

2. hive에서 테이블 생성 스크립트를 만든다.

(테이블명 : movies, users)

* os에서 어디든지 hive라고 치면 hive에 접속하게 되는 방법

\$ 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

그리고 .bashrc 스크립트를 수행한다.

\$. .bashrc

hive로 접속한다.

\$ hive

create table movies

(movieid int,

title string,

genres string)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ''

LINES TERMINATED BY '\n'

STORED AS TEXTFILE ;

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 ;

create table ratings

(userid int,

movieid int,

rating int,

tstamp string)

```
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ''
LINES TERMINATED BY '\n'
STORED AS TEXTFILE ;
```

3. **movies_coma.dat, ratings_coma.dat, users_coma.dat** 이 세개의 파일을 하둡 파일 시스템에 올린다.

윈도우 -----> 리눅스 -----> 하둡 파일 시스템

실제 세상 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
```

4. 하둡 파일 시스템에 올린 두개의 파일을 각각 **movies**와 **users** 테이블에 로드합니다.

```
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;
```

```
hive> select count(*)from movies;
hive> select count(*)from users;
hive> select count(*)from ratings;
hive> select * from movies limit 5; (5개행만 제한해서 보여줌)
```

■ 지난주에 배웠던 리눅스와 하둡 수업 복습

1. 리눅스를 왜 배워야 하는지?
2. 리눅스 설치
3. 리눅스 기본 명령어
4. vi편집기 명령어
5. 리눅스에 아나콘다와 spyder 설치
6. 리눅스 권한관리 명령어
7. 디스크 관리 명령어
8. 프로세서 관리 명령어
9. 쉘스크립트 작성법
10. 하둡설치
11. 하이브 설치

현업에서 어떻게 하둡과 하이브, 스칼라를 이용하고 있을까?

1. 하둡과 하이브와 스칼라에서는 데이터(csv, text)를 로드하는 테이브 생성을 위주로 함
2. 하둡과 하이브와 스칼라에서 데이터 검색을 해서 분석
(GUI 툴 다람쥐 squirrel를 이용 --> SQL developer와 같은 툴)
3. 하이브와 파이썬 연동, 스칼라와 파이썬 연동을 해서 데이터 시각화와 분석

큰 질문? Ex) 코로나는 어느 장소에서 많이 감염이 되는가?

1. 코로나 환자들의 동선 정보가 있는 데이터를 구한다. (text, csv)
2. 리눅스 서버에 데이터를 넣고 하둡 하이브에서 테이블을 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.

리눅스와 하둡 시험(NCS 시험) :

위의 질문을 스스로 정해서 데이터를 구해서 하둡에 넣고 분석을 해서 시각화한 결과물 (워드, 한글, PPT)로 제출

1. 하둡 시스템이 정상인지 확인

\$ jps <--- 6개의 프로세서가 올라와야 합니다.

\$ start-all.sh <--- 하둡을 저장하지 않고 껐으면 프로세서가 1개만 나오므로 하둡을 올려준다.
\$ jps

그래도 안뜨면 해결방법

```
$ stop-all.sh  
**하둡 data 노드와 name 노드의 디렉토리의 데이터를 다 지운다.  
$ cd $HADOOP_HOME/dfs  
$ cd name  
$ rm -rf *  
$ cd ..  
$ cd data  
$ rm -rf *  
3. 네임노드 포맷한다.  
$ hadoop namenode -format  
4. start-all.sh  
5. jps
```

2. hive 로 접속합니다.

```
$ hive
```

```
hive>  
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>  
create table dept  
( deptno int,  
  dname string,  
  loc   string )  
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> load data inpath '/user/scott/dept2.csv'  
      overwrite into table dept;
```

테이블 생성 코드

```
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,
decreased_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;
```

* tajo를 이용해서 데이터 검색하기

hive보다 훨씬 빠른 NoSQL입니다. 자체 설계한 처리 엔진을 통해 Hive보다 훨씬 빠른 처리시간을 보여줍니다.

■ tajo 설치

1. 타조 설치 파일을 /home/scott 밑에 가져다 둡니다.
2. 압축을 해제합니다.

```
$ tar -xvzf tajo-0.11.1-src.tar.gz
```

3. 압축 풀린 디렉토리 이름을 간단하게 tajo로 변경합니다.

```
$ mv tajo-0.11.1-desktop-r1 tajo
```

3. 압축을 풀면 tajo 로 시작하는 디렉토리가 생성되는데 아래의 디렉토리가 잘 생성되었는지 확인하시오 !

```
$ ls -ld tajo-0.11.1-desktop-r1
```

4. tajo-0.11.1-desktop-r1 의 이름을 tajo 로 변경합니다.

```
$ mv tajo-0.11.1-desktop-r1 tajo
```

5. tajo 디렉토리로 cd 명령어로 이동해서 ls 해봅니다.

```
$ cd tajo
```

6. cd 명령어로 /home/scott/tajo/tajo/bin 디렉토리로 이동합니다.
(bin 디렉토리는 실행파일 모아놓은곳)

```
$ cd /home/scott/tajo/tajo/bin  
$ pwd
```

7. 타조 환경구성 파일인 configure.sh 를 실행합니다.

```
$ ./configure.sh  
Enter JAVA_HOME [default: /u01/app/java/jdk1.7.0_60]
```

Would you like advanced configure? [y/N]
N

8. 타조를 시작 시킵니다.

```
$ cd /home/scott/tajo/tajo/bin  
$ sh startup.sh
```

9. 타조에 접속합니다.

```
$ cd /home/scott/tajo/tajo/bin  
$ sh tsq1
```

■ 스파크 설치

1. scott 의 흡디렉토리로 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 xvzf 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 에 아래의 명령어를 맨 아래에 입력합니다.

```
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 where applicable
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 instead (on interface enp0s3)
21/01/09 08:54:43 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
21/01/09 08:54:45 WARN SparkContext: Use an existing SparkContext, some configuration may not take effect.
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id = local-1610200485296).
Spark session available as 'spark'.
Welcome to
```

/ \ _ _ _ _ / _
_ # # / _ # / _ ` / _ ' / _
/ _ / . _ # _ / / _ # _ version 2.0.2
/ _

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_262)
Type in expressions to have them evaluated.
Type :help for more information.

설명 : 테이블 생성전에 아래의 명령어를 실행해야 한다. '지금부터 SQL을 사용하겠다' 라고 저장한다.

```
scala> val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
warning: there was one deprecation warning; re-run with -deprecation for details
sqlContext: org.apache.spark.sql.hive.HiveContext = org.apache.spark.sql.hive.HiveContext@66b0e207
```

설명: OS에 저장한 employee.txt 데이터를 저장할 테이블을 설정

```
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'")  
res0: org.apache.spark.sql.DataFrame = []
```

설명 : os의 employee.txt 파일을 employee에 로드 한다.

```
scala> sqlContext.sql("LOAD DATA LOCAL INPATH 'employee.txt' INTO TABLE employee")  
res1: org.apache.spark.sql.DataFrame = []
```

설명 : employee 테이블에서 id와 name과 age를 조회한다.

```
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|  
+---+---+
```

설명 : 위에처럼 result라는 변수에 담지 않고 바로 sql을 실행하고 싶다면 아래와 같이 수행하면 된다.

```
scala> sql("select * from employee").show()  
+---+---+  
| id| name|age|  
+---+---+  
|1201| satish| 25|  
|1202|krishna| 28|  
|1203| amith| 39|  
|1204| javed| 23|  
|1205| prudvi| 23|  
+---+---+
```

spark-shell 해서 들어가고 :quit 해서 나옴

데이터 분석시에 하둡 하이브 또는 스칼라에서 구현해야 하는것?

---> 빅데이터를 저장할 테이블 생성.

파이썬에서 시각화 또는 머신러닝 데이터 분석을 한다

■ 리눅스 수업 후 하둡 수업 내용

0. 리눅스 수업 (리눅스 설치, 리눅스 기본 명령어, vi 편집기, 기타)
1. 하둡 설치
2. hive 설치
3. hive에서 SQL로 빅데이터를 검색하는 연습(오라클과 Hive 비교)
4. 스칼라 설치
5. 스칼라에서 SQL로 빅데이터를 검색하는 연습
6. (최종목표) 하이브와 스칼라에서 데이터를 검색하고 전처리한 후 이 데이터를 파이썬에서 시각화

■ 스칼라에서 emp 테이블이 어느자리에서는 만들어지고 어느 자리에서는 안 만들어지는 이유 ?

일단 스칼라에서 SQL로 작업을 하다가 에러가 났는데 그 에러가 좀 난이도가 높은 에러였으면 다음에 정상적인 스칼라 SQL문법을 수행해도 수행이 안되고 계속 에러가 납니다. 이럴때는 스칼라를 내렸다가 올리고 다시 스칼라에 접속하면 해결됩니다.

■ 스칼라를 내렸다가 다시 접속하는 명령어

1. 스파크의 sbin 디렉토리로 이동합니다.

```
$ cd /home/scott/spark/sbin  
$ ls  
start-all.sh 와 stop-all.sh가 있는지 확인해야 합니다.
```

설명 : 위의 스파크 디렉토리의 start-all.sh와 stop-all.sh는 하둡의 start-all.sh와 stop-all.sh와는 별개의 파일입니다.

```
[scott@centos sbin]$ ./stop-all.sh  
[scott@centos sbin]$ ./start-all.sh
```

```
$ spark-shell
```

■ 스칼라에서 emp 테이블을 생성하는 방법

1. hive SQL을 스칼라에서 쓰겠다는 명령어

```
scala> val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)  
warning: there was one deprecation warning; re-run with -deprecation for details
```

```
sqlContext: org.apache.spark.sql.hive.HiveContext =  
org.apache.spark.sql.hive.HiveContext@e43a34
```

2. emp 테이블을 생성하는 명령어

```
scala> sqlContext.sql("create table IF NOT EXISTS 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'")  
res7: org.apache.spark.sql.DataFrame = []
```

3. emp테이블에 emp2.txt를 로드하는 명령어

```
scala> sqlContext.sql("LOAD DATA LOCAL INPATH '/home/scott/emp2.txt' INTO TABLE  
emp")  
res8: org.apache.spark.sql.DataFrame = []
```

4. emp 테이블의 내용을 확인하는 명령어

```
scala> sql("select * from emp").show()  
+----+-----+-----+-----+-----+-----+  
|empno|ename|job|mgr|hiredate|sal|comm|deptno|  
+----+-----+-----+-----+-----+-----+  
| 7839| KING|PRESIDENT| 0|19811117|5000| 0| 10|  
| 7698| BLAKE| MANAGER|7839|19810501|2850| 0| 30|  
| 7782| CLARK| MANAGER|7839|19810509|2450| 0| 10|  
| 7566| JONES| MANAGER|7839|19810401|2975| 0| 20|  
| 7654|MARTIN| SALESMAN|7698|19810910|1250|1400| 30|  
| 7499| ALLEN| SALESMAN|7698|19810211|1600| 300| 30|  
| 7844|TURNER| SALESMAN|7698|19810821|1500| 0| 30|  
| 7900| JAMES| CLERK|7698|19811211| 950| 0| 30|  
| 7521| WARD| SALESMAN|7698|19810223|1250| 500| 30|  
| 7902| FORD| ANALYST|7566|19811211|3000| 0| 20|  
| 7369| SMITH| CLERK|7902|19801209| 800| 0| 20|  
| 7788| SCOTT| ANALYST|7566|19821222|3000| 0| 20|  
| 7876| ADAMS| CLERK|7788|19830115|1100| 0| 20|  
| 7934|MILLER| CLERK|7782|19820111|1300| 0| 10|  
+----+-----+-----+-----+-----+-----+
```

5. emp 테이블이 전체 몇건인지 확인하는 명령어

```
scalar > sql("select * from emp").count()
```

결과 : res14: Long = 14

■ emp 테이블을 drop하고 다시 생성하는 방법

```
scalar> sql("drop table emp")
```

데이터 분석을 위한 쿤그림에서의 하둡과 스칼라의 역할은?

1. 빅데이터를 저장하기 위한 테이블 생성
2. 테이블에서 데이터를 전처리하고 필요한 데이터 검색
3. 검색한 데이터를 csv 파일로 저장해서 파이썬이나 R로 보낸다.

4. 파이썬이나 R에서 가져온 csv파일로 데이터 시각화

아래의 결과를 복사해서 윈도우에서 notepad를 열고 붙여넣은 다음 job.csv로 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,5000
```

윈도우에서 spyder를 켜고 아래의 내용을 코딩한다.

■ 하둡과 하이브와 스칼라는 빅데이터를 저장하는 테이블 생성 위주로 현업에 활용을 하니까 테이블을 생성해서 데이터를 저장하고 파이썬이나 R로 시각화 또는 분석을 한다.

■ 지금처럼 emp 테이블은 데이터가 작으니까 그냥 결과를 복사해서 윈도우의 메모장에다가 붙여넣었지만 현업에서는 대용량 데이터이므로 이렇게 할 수 없다. csv 파일을 따로 다운로드 받아야 한다.

이렇게 하는 방법은 다음과 같다.

1. mobaxterm 프로그램을 다운로드 받아서 설치한다.
2. mobaxterm으로 리눅스 centos에 접속을 한다.
3. mobaxterm에서 파일을 다운로드 받는다.

■ 리눅스에 설치되어 있는 아나콘다의 spyder를 내 자리의 컴퓨터(내 노트북)의 화면에 띄우게 하는 방법

```
[root@centos ~]# cd /etc/ssh/
[root@centos ssh]# ls -l sshd_config
[root@centos ssh]# yum install xclock
```

■ putty에서 xclock 여는 방법

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.

```
$ export DISPLAY=192.168.19.31:0.0
```

```
$ xhost
```

3. xclock 을 실행합니다. (mobaxterm 이 실행되어져 있어야합니다.)

```
$ xclock
```

설명 : 위에서 수행할 때 mobaxterm 경고창이 안나오게 하려면?

mobaxterm ----> settings ----> configuration ----> X11 탭 ---> x11 remote access을 full로 변경

설명 : mobaxterm은 파일 전송하고 내려받은거 위주로 사용하고 대부분 작업을 putty에서 수행하면 됩니다.

■ 지금까지 설정한 리눅스 환경에 아나콘다 설치하기!

복제한 환경:

하둡, 하이브, 스칼라, putty 설정이 다 되었으므로 아나콘다만 설치하면 된다.

※ 중요: 아나콘다 설치하고 spyder 설치하는 작업을 root 에서 하지 말고 scott 에서 수행해야합니다.

1. root 에서 로그아웃하고 scott 으로 접속한다.

2. \$ wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-x86_64.sh

3. \$ sha256sum Anaconda3-2020.11-Linux-x86_64.sh

4. \$ bash Anaconda3-2020.11-Linux-x86_64.sh

5. \$ source .bashrc

6. \$ conda create -n py389

7. \$ conda activate py389

8. \$ conda install spyder

9. \$ pip install PyQt5==5.10

10. \$ pip install pyqtwebengine==5.12

11. \$ pip install pyqt5==5.12

12. \$ conda install pandas <<< 판다스 모듈 설치 (py389에서 설치)

13. \$ conda install matplotlib <<< matplotlib 설치 (py389에서 설치)

14. \$ spyder

※ 혹시 설치과정에서 0% 에서 멈춰있으면 ctl +c 를 누르고 취소하고 /home/scott/ 밑에 anaconda 폴더를 지웁니다.

그리고 설치파일의 권한을 777 로 넣어줍니다. 그리고 rebooting 을 하고 다시 시작합니다.

그래도 안되면 설치 파일인 Anaconda3-2019.10-Linux-x86_64.sh 도 지우고 다시 받아 설치합니다.

위의 방법으로 잘 안되면 아래에 방법으로 다시 설치

<https://nowonbun.tistory.com/691>

centos spyder에서

```
import pandas as pd  
emp= pd.read_csv("/home/scott/dd/deptno.csv")  
print(emp)
```

이렇게 하면 잘 나온다.

■ 실수로 다 꺼버리거나 putty를 다시 접속해야 하는 상황이면

1. putty로 다시 scott 으로 접속하고

■ 리눅스와 하둡 수업 복습

리눅스와 하둡을 배우는 이유?

>>> 빅데이터 데이터 분석 및 시각화

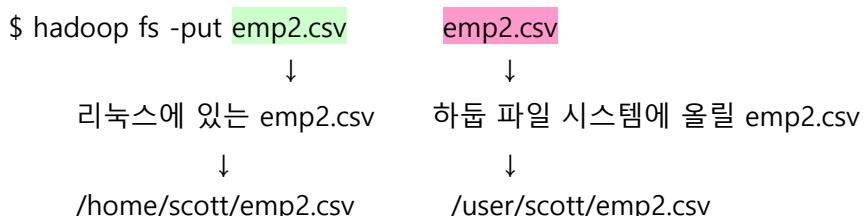
코로나 테라급 데이터(csv, text)를 가지고 코로나의 확산을 막기 위해서 데이터 분석 질문을 던진다고 할 때 기존의 오라클에 그 테라급 데이터를 넣어서 쿼리를 수행한다면 비용과 시간이 너무 많이 듭니다.

무료인 하둡과 스칼라 그리고 maria db 등을 이용해서 빅데이터를 저장하고 빅데이터를 빠르게 검색하여 원하는 정보를 얻어내려면 리눅스와 하둡을 사용해야 합니다.

small data이면 오라클이나 MySQL을 사용해서 분석을 해도 됩니다.

데이터가 많을수록 더 설득력 있고 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 되었습니다.

하둡을 사용하면 왜 빅데이터 처리를 빨리 할 수 있는가?



윈도우 -----> 리눅스 가상 머신 -----> 하둡 파일 시스템
현실세계 꿈속 꿈속의 꿈속

hive> create table emp

1. 아래의 명령어는 local을 사용했기 때문에 리눅스 서버의 /home/scott 밑에 있는 emp2.csv의 데이터를 hive의 emp에 로드해라 라는 뜻

hive> LOAD DATA LOCAL INPATH '/home/scott/emp2.csv' INTO TABLE emp;

로컬에만 올리고

hive > select * from emp; # 컴퓨터를 한대만 작동해서 데이터를 검색해줌

2. 하둡 파일 시스템에 올린 emp2.csv를 emp 테이블에 입력해라~ 라는 뜻
아래의 명령어에는 local이 없습니다.

```
hive> LOAD DATA INPATH '/home/scott/emp2.csv' INTO TABLE emp;
```

하둡 시스템에 올리고

```
hive > select * from emp; # 컴퓨터 30대가 다 움직이면서 데이터를 검색해줌
```

뉴욕타임즈 기사 수십년 신문기사를 하둡을 이용해서 pdf로 변환하는데 10년 이상이 걸려야하는 작업을 하루에 다 처리함.

하둡을 이용하면 컴퓨터 한대의 자원만 사용하는게 아니라 여러대의 컴퓨터를 묶어서 마치 하나의 서버처럼 보이게 해서 여러대의 컴퓨터의 자원을 이용할 수 있다라는 장점이 있다.

하둡을 이용하지 않았을 때

vs

하둡을 이용했을 때

서버 1대 (64기가 메모리)

컴퓨터 30대(8기가x 30 = 240기가)

■ 하둡 분산 파일 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=192.168.19.3:0.0
```

```
$ xclock
```

```
$ firefox centos:50070
```

■ 2. lsр 명령어

"현재 디렉토리 뿐만 아니라 하위 디렉토리까지 조회하는 명령어"

예제: \$ hadoop fs -lsr /user/

■ 3. du 명령어

"파일의 용량을 확인하는 명령어"

예제 : \$ hadoop fs -du

결과:

```
84    hdfs://localhost:9000/user/scott/dept2.csv
644   hdfs://localhost:9000/user/scott/emp2.csv
644   hdfs://localhost:9000/user/scott/emp3.csv
163542  hdfs://localhost:9000/user/scott/movies_coma.dat
21593504 hdfs://localhost:9000/user/scott/ratings_coma.dat
110208  hdfs://localhost:9000/user/scott/users_coma.dat
```

■ 4. dus 명령어

"파일의 전체 합계 용량을 확인하는 명령어"

예제 : \$ hadoop fs -dus

■ 5. cat 명령어

"지정된 파일의 내용을 확인하는 명령어"

예제 : \$ hadoop fs -cat /user/scott/emp3.csv

■ 6. text 명령어

"지정된 파일의 내용을 화면에 출력하는 명령어" (리눅스에는 없는, 하둡에만 있는 명령어/ cat 명령어와 비슷)

예제 : \$ hadoop fs -text /user/scott/emp3.csv

■ 7. mkdir 명령어

"디렉토리를 생성하는 명령어"

예제 : \$ hadoop fs -mkdir /user/scott/test

■ 8. put 명령어

"하둡 파일 시스템에 파일을 올리는 명령어"

예제 : \$ hadoop fs -put /home/scott/emp2.csv /user/scott/test/emp5.csv

■ 9. get 명령어

"하둡 파일 시스템에 올린 파일을 리눅스 디렉토리로 내리는 명령어"

예제 : \$ cd

```
$ rm emp2.csv
$ ls -l emp2.csv #하면 안나옴/ 지웠으니
$ hadoop fs -get /user/scott/test/emp5.csv /home/scott/emp2.csv
$ ls -l emp2.csv # 나옴
```

■ 10. rm 명령어

"하둡 파일 시스템에 있는 파일을 지우는 명령어"

```
예제: $ hadoop fs -lsr /user/  
$ hadoop fs -rm /user/scott/emp3.csv
```

결과:

```
Deleted hdfs://localhost:9000/user/scott/emp3.csv
```

■ 11. rmr 명령어

"하둡 파일 시스템의 디렉토리를 삭제하는 명령어"

```
예제 : $ hadoop fs -rmr /user/scott/test
```

결과: Deleted hdfs://localhost:9000/user/scott/test

■ 12. grep 명령어

"파일에서 특정 문자의 행에 데이터를 검색하는 명령어"

```
예제 : $ hadoop fs -put /home/scott/emp2.csv /user/scott/emp7.csv  
$ hadoop fs -cat /user/scott/emp7.csv | grep -i 'salesman'
```

1. 리눅스 /home/scott/ 밑에 있는 emp2.csv를 판다스로 로드해서 emp2.csv를 emp 데이터 프레임으로 만들 어서 프린트 하시오

답 :

```
import pandas as pd
```

```
emp= pd.read_csv("/home/scott/emp2.csv", header=None)  
print(emp)
```

2. 직업과 직업별 인원수를 출력하시오!

```
import pandas as pd
```

```
emp= pd.read_csv("/home/scott/emp2.csv", header=None)  
  
result=emp[2].value_counts()  
print(result)
```

결과:

```
CLERK      4  
SALESMAN   4  
MANAGER    3  
ANALYST    2  
PRESIDENT  1
```

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()
```

```
print(result)
```

```
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

■ 13. awk 명령어

"특정 컬럼을 검색하는 명령어"

예: 하둡 파일 시스템에 올린 emp3.csv 파일에서 SCOTT이 사원의 이름과 월급을 조회하시오!

```
$ hadoop fs -cat /user/scott/emp2.csv | grep -i 'scott' | awk -F "," '{print $2, $6}'
```

설명 : 데이터가 콤마()로 구분되어 있으면 awk에 -F 옵션을 써서 ","로 구분되어 있다는 것을 알려주야 한다.

scott,2000,salesman ---> awk -F ","

scott/2000/salesman ---> awk -F "/"

■ 14. count 명령어

"지정된 디렉토리의 파일의 개수를 확인하는 명령어"

예제: \$ hadoop fs -lsr /user/scott

```
$ hadoop fs -count /user/scott
```

■ 15. 하둡 파일 시스템 명령어 매뉴얼 보는 방법

```
$ hadoop fs -help
```

■ 타조 설치

1. 타조 설치파일이 /home/scott 밑에 있어야 합니다.

```
$ ls -l tajo-0.11.1.tar.gz
```

2. 설치파일의 압축을 해제합니다.

```
$ tar -xvzf tajo-0.11.1.tar.gz
```

3. 압축을 풀면 tajo로 시작하는 디렉토리가 생성되는데 아래의 디렉토리가 잘 생성되었는지 확인하시오!

```
$ ls -ld tajo-0.11.1-desktop-r1
```

4. tajo-0.11.1-desktop-r1의 이름을 tajo로 변경합니다.

```
$ mv tajo-0.11.1-desktop-r1 tajo
```

5. tajo 디렉토리로 cd 명령어로 이동해서 ls 해봅니다.

```
$ cd tajo
```

6. cd 명령어로 /home/scott/tajo/tajo/bin 디렉토리로 이동합니다.

```
$ cd/home/scott/tajo/tajo/bin
```

```
$ pwd
```

7. 타조 환경구성 파일인 configure.sh 를 실행합니다.

```
$ ./configure.sh
```

```
Enter 치고 N 쓰기
```

8. 타조를 시작시킵니다.

```
$ cd /home/scott/tajo/tajo/bin
```

```
$ sh startup.sh
```

9. 타조에 접속합니다.

```
$ cd /home/scott/tajo/tajo/bin
```

```
$ sh tsqI
```

10. 타조에서 database를 생성한다.

```
default > create database orcl;
```

11. orcl 데이터베이스에 접속합니다.

```
default> Wc orcl;
```

결과:

```
You are now connected to database "orcl" as user "scott".
```

12. 타조에서 emp 테이블 생성하기

```
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';
```

```
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, 0.044 sec, 0 B selected)
```

■ 마리아 db 설치

하둡과 상관 없이 설치할 수 있습니다.

oltp 환경	vs	dw 환경
Oracle (유료)		하둡과 하이브(무료)
mysql (유료)		스칼라 (무료)
maria db (무료)		

maria db는 mySQL을 만든 개발자가 회사를 나와서 만든 무료 Database

1. scott에서 sudo 명령어를 수행할 수 있도록 설정

<https://myjamong.tistory.com/3>

2. scott에서 maria 디비를 설치를 합니다.

<https://suwoni-codelab.com/linux/2017/05/24/Linux-CentOS-MariaDB/>

```
[root@centos scott]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or #.
Your MariaDB connection id is 13
Server version: 10.5.8-MariaDB MariaDB Server
```

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or 'Wh' for help. Type 'Wc' to clear the current input statement.

```
MariaDB [(none)]>
MariaDB [(none)]> create database orc;
Query OK, 1 row affected (0.001 sec)
```

```
MariaDB [(none)]> create database orcl;
Query OK, 1 row affected (0.000 sec)
```

```
MariaDB [(none)]> use orcl;
Database changed
```

```
# maria db 시작
$ systemctl start mariadb

# maria db 접속(sudo 상태에서 실행)
[root@centos scott]# mysql -u root -p
Enter password: (1234 입력)
```

```
# database 생성하기  
MariaDB [(none)]> create database orcl;  
Query OK, 1 row affected (0.000 sec)
```

```
MariaDB [(none)]> use orcl;  
Database changed
```

```
MariaDB [orcl]>
```

● AUTOCOMMIT check in MariaDB
<1>default value
MariaDB [orcl]> select @@autocommit;
+-----+
| @@autocommit |
+-----+
| 1 |
+-----+
1 row in set (0.000 sec)

<2>change commit option
set autocommit=1; # autocommit 설정 on
set autocommit=0; # autocommit 설정 off

<3>check
MariaDB [orcl]> set autocommit=0;
Query OK, 0 rows affected (0.000 sec)

```
MariaDB [orcl]> select @@autocommit;  
+-----+  
| @@autocommit |  
+-----+  
| 0 |
```

■ MariaDb 파티션 테이블 생성하기

1. 파티션 테이블이 왜 필요한가?

데이터가 대용량 데이터이면 테이블에서 데이터를 검색할 때 시간이 아주 많이 걸리게 된다. 인덱스 생성해도 인덱스(목차)도 사이즈가 크기 때문에 인덱스를 통한 데이터 검색 효과를 볼 수가 없다. 그럴 때 파티션 테이블을 생성하면 된다.

예 : 옷장에 서랍이 4개가 있으면 서랍마다 봄옷, 여름옷, 가을옷, 겨울옷 분리해서 각각 저장해두면 옷을 찾기가 편해집니다.

물리적으로 데이터를 분리해서 저장하는 것을 말합니다.

예제 : emp 테이블을 부서번호마다 별도의 파티션에 데이터가 저장되게끔 파티션 테이블을 구성합니다.

부서번호 10번 -----> 1번 파티션

부서번호 20번 -----> 2번 파티션

부서번호 30번 -----> 3번 파티션

1. 파티션 되어있지 않은 emp 테이블에서 20번 부서번호를 조회
select ename, sal, deptno

```
from emp  
where deptno = 20;
```

1. 파티션이 되어있는 emp_partition 테이블에서 20번 부서번호를 조회
select ename, sal, deptno

```
from emp_partition  
where deptno=20  
and ename='SCOTT';
```

빅데이터 환경에서는 큰 테이블 몇개는 파티션 테이블로 구성되어 있습니다.

```
=====  
==  
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) );
```

```
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>emp table  
MariaDB [orcl]>  
select ename, sal, deptno  
from emp  
where deptno = 20;  
# deptno = 20 조건을 찾기 위해 전 구간 scan
```

```
<2>emp_partition table  
MariaDB [orcl]>  
select ename, sal, deptno  
from emp_partition
```

```
where deptno = 20;
# deptno = 20 이 물리적으로 분리되어 있기 때문에 해당 부분만 scan
# emp 테이블은 정렬되어있지 않지만 emp_partition 은 deptno asc 정렬이 되어있다.
(MariaDB [orcl]> select * from emp_partition 으로 확인)
# bigdata 환경에서 <1>과 <2>는 속도 차이가 많이 난다
# 대부분의 빅데이터 환경에서 큰 테이블 몇 개는 파티션 테이블로 구성이 되어있다.
```

■ 리눅스와 하둡으로 할 수 있는게 무엇인가?

시스템 구축

1. 리눅스 centos 설치
2. 하둡 설치
3. hive 설치
4. 스칼라 설치
5. 파이썬 아나콘다 설치
6. maria db 설치
7. 부가적인 작업 : 원격에서 putty로 접속하게 설정
mobaxterm을 이용해서 파일 전송과 윈도우에서 리눅스의 spyder를 실행

현업에서 위와 같은 시스템을 구축해 놓고 활용하는 사례?

미래에셋의 주식 인공지능 프로그램인 로보 어드바이저

파이썬을 주식 데이터를 웹스크롤링 ---> 리눅스 서버 ---> 마리아 db ---> 주식 데이터 저장 ---> 파이썬과 마리아 db 연동 ---> 파이썬에서 머신러닝과 딥러닝 알고리즘으로 주가를 예측 --> 파이썬 장고로 화면 구현

구현한 리눅스 서버를 활용하여 데이터 시각화 :

1. 리눅스와 하둡 수업의 시험 평가물 제출 포멧
2. 주식 데이터를 웹스크롤링 해서 주식 데이터를 마리아 디비에 저장, 파이썬과 연동해서 데이터 시각화

■ 마리아 디비와 파이썬 연동

1. py389 를 activate 시키고 mysql 모듈을 설치한다.

```
$ conda activate py389
```

```
(py389) [scott@centos ~]$ pip install mysql-connector-python-rf
```

설명 : 마리아 디비 또는 MySQL과 파이썬과 연동을 하기 위한 모듈 설치

설치가 잘 되었는지 다음과 같이 확인한다.

```
(py389) [scott@centos ~]$ conda list mysql-connector-python-rf
```

2. mysql로 접속해서 모든 ip의 접속 권한을 부여합니다.

```
(base) [scott@centos ~]$ su -
```

Password:

```
[root@centos scott]# mysql -u root -p
```

설치를 했으면 또 안해도 됨. 바로 use orcl로 들어가기

```
MariaDB [(none)]> create database orcl;
Query OK, 1 row affected (0.000 sec)
```

```
MariaDB [(none)]> use orcl;
```

아래의 쿼리문은 Maria 디비의 디비 유저에 대한 권한 정보를 확인하는 명령어

```
MariaDB [orcl]> select Host,User,plugin,authentication_string FROM mysql.user;
```

```
MariaDB [orcl]> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '1234';
Query OK, 0 rows affected (0.021 sec)
```

3. 마리아 디비에 접속하게끔 방화벽 설정을 합니다.

```
MariaDB [orcl]> exit;
```

```
[root@centos ~]# firewall-cmd --list-all-zones
```

설명 : 포트 3306은 마리아 디비를 위한 기본 포트(port)

이 포트를 열어서 파이썬과 연동이 원활하게 되게 해줘야 합니다.

```
[root@centos ~]# firewall-cmd --permanent --zone=public --add-port=3306/tcp
success
```

```
[root@centos ~]# exit
```

```
logout
```

4. 스파이더를 실행해서 아래의 코드를 실행합니다.

(mobaxterm을 먼저 실행하고 scott으로 접속한 상태에서 다음 작업을 실행합니다)

```
(py389) [scott@centos ~]$ export DISPLAY=192.168.19.3:0.0
```

```
(py389) [scott@centos ~]$ spyder
```

코드:

```
import mysql.connector
```

```

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 (마리아 디비에 있는 emp 테이블을 select 한다)
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)

```

■ 마리아 db의 장점

1. 무료이다
2. MySQL과 기능이 똑같다.
3. 리눅스의 CSV, text 파일도 로드할 수 있다.

■ 리눅스 csv파일을 마리아 db로 로드하는 방법

마리아 디비 연동하여 파이썬에서 시각화 하기 총정리

Case.csv
11.00KB

1. 마리아 디비에 코로나 데이터 입력

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) );
```

아래는 터미널 창에서 할 것

```
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);

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. SQL을 다시 작성합니다.

```

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')

```

전체 코드:

```

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
"""

```

```
"""\n\n# cursor 객체를 이용해서 수행한다.\ncursor.execute(sql)\n\n# select 된 결과 셋 얻어오기\nresultList = cursor.fetchall() # tuple 이 들어있는 list\ndf = pd.DataFrame(resultList)\ndf.columns = ['city', 'cnt']\nprint(df[['city','cnt']])
```

```
# 시각화\nresult = df['cnt']\nresult.index = df['city']\nresult.plot(kind='bar', color='red')
```

리눅스와 하둡 ncs 시험 - 제출물

- 제출물 만드는 순서
 1. 큰 질문: 코로나는 남자와 여자 중에 누가 더 많이 감염되었는가?
 2. csv 데이터를 구한다.
 3. 마리아 db에 테이블로 생성한다.
 4. 파이썬에서 시각화를 한다.
 5. 시각화 그래프와 함께 짧게 시각화한 결과를 설명

오늘의 문제(001-308)

2020년 12월 29일 화요일 오전 11:03

1. 현재 디렉토리에서 Desktop 디렉토리로 이동하시오!

답 : # cd Desktop

2. <--

3. 다시 뒤로 이동하시오.

답: # cd ..

❖ 설명 : cd하고 한칸 띄고 ..(점점) 을 쓰세요~

3. 나의 집으로 가시오

답 : # whoami <-- 내가 누구인지 확인한다.

cd <-- 내가 어디에 있는지 cd하고 엔터치면 집에간다.

pwd

결과: /root <-- 여기가 root의 집입니다.

4. 집으로 가서 집에서 test01이라는 폴더를 생성하시오!

답: # cd

mkdir test01

ls

5. test01 폴더(디렉토리)로 이동하시오!

답: # cd /root/test01

또는

cd test01

6. 집에 가서 test02라는 폴더(디렉토리)를 생성하고 test02로 이동하시오.

답: # cd

mkdir test02

cd test02

7. 집에가서 test03 디렉토리를 만들고 test03 디렉토리로 가서 test04 디렉토리를 만들고 test04 디렉토리로 이동하시오.

답 : # cd

mkdir test03

cd test03

pwd

mkdir test04

cd test04

pwd

cd

8. 집에서 test03 밑에 test04로 이동하시오!

답: # cd

cd test03/test04

pwd

9. 지금 이동한 디렉토리에서 방금 이동하기 전 디렉토리로 바로 이동하시오!

답 : # pwd
cd -
pwd

10. 집에서 test07 디렉토리를 만들고 test07 디렉토리로 이동하고 거기서 test08 디렉토리를 만들고 test08 디렉토리로 이동한 후에 test09 디렉토리를 만들고 test09 디렉토리로 이동하시오!

답 : # cd
mkdir test07
cd test07
mkdir test08
cd test08
pwd
mkdir test09
cd test09
pwd

11. 집에서 아래의 파일들을 크기를 0으로 해서 생성하시오!

a.txt b.txt c.txt d.txt e.txt f.txt

답: # touch a.txt b.txt c.txt d.txt e.txt f.txt
ls -l *.txt <--- 확장자가 .txt인 파일들만 보여달라

12. 집에서 아래의 디렉토리들을 생성하시오!

/root/test30/test31/test32/test33/test34/test35

답: # cd
mkdir -p test40/test41/test42/test43/test44/test45

❖ 설명 : -p 옵션은 디렉토리를 만들면서 한번에 하위 디렉토리를 생성하는 옵션입니다.

13. (점심시간문제) 집에서 파이썬 스크립트를 따로 관리할 디렉토리를 ptest01로 생성한 ptest01 디렉토리로 이동해서 파일의 크기가 0인 example01.txt 파일을 생성하시오.

답: # cd
mkdir ptest01
cd ptest01
touch example01.txt
cd

14. 집으로 이동해서 현재 디렉토리에 있는 a.txt와 b.txt와 c.txt를 삭제하시오!

답 : # cd
ls -l a.txt b.txt c.txt

15. 집으로 이동해서 test45라는 디렉토리를 생성하시오!

답 : # cd
mkdir test45
ls -ld test45

설명 : -ld 옵션을 사용하면 test45 디렉토리의 정보만 따로 확인할 수 있습니다.

16. 집으로 이동해서 방금 만든 test45 디렉토리를 삭제하시오!

답 : # cd

```
# rm test45
```

----> 안지워짐

따라서

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

이렇게 해주어야 함.

❖ -r옵션은 디렉토리를 삭제할 때 사용하는 옵션입니다.

디렉토리 삭제 시 삭제할 디렉토리 밑에 있는 파일과 하위 디렉토리를 모두 다 삭제하는 옵션입니다.

진짜 조심할 것!!!

17. 집으로 가서 test03 디렉토리 밑에 하위 디렉토리가 있는지 확인하시오!

```
# cd  
# cd test03  
# ls  
# cd test04  
# ls  
# cd
```

18. 집으로 가서 /root 밑에 있는 test03 디렉토리와 그 하위 디렉토리를 모두 삭제하시오!

답 : # cd
rm -r test03

19. 집으로 가서 /root 밑에 test51 디렉토리를 생성하고 test51 디렉토리에 emp.txt를 올리시오!

(카페에서 내려받으세요~)

장치 --> 클립보드 공유 , 드래그엔 드롭--> 양방향 공유를 하세요!!

답: # cd
mkdir test51

20. ls -l 명령어를 그냥 l로 쉽게 수행할 수 있도록 alias를 만드시오!

답: # alias l='ls -l'
l

21. 다시 alias를 지우고 싶다면?

답 : unalias p

22. alias로 등록 된 목록을 확인하시오!

답 : # alias

23. l로 만들었던 alias도 지우시오!

unalias l

24. 스티브 잡스 연설문(jobs.txt)를 /root 밑에 올리시오.

home에 갖다 넣기

25. jobs.txt를 cat으로 여시오!

답 : # cd
cat jobs.txt

26. **jobs.txt**를 천천히 읽을 수 있도록 **more** 명령어로 여시오!

답 : # more jobs.txt

중간에 그만 빠져나오고 싶으면 q를 누르면 된다.

27. **jobs.txt**를 **cat**으로 열어서 본 결과를 **jobs2.txt**로 저장하시오!

답: # cat jobs.txt >> jobs2.txt

28. 집에서 **ls -l**로 본 결과를 **list.txt**로 저장하시오

답 : # cd
ls -l >> list.txt
ls -l list.txt
cat list.txt

29. **jobs.txt**의 위의 10줄을 **jobs_10.txt**로 저장하시오!

답 : # head -10 jobs.txt >> jobs_10.txt

30. 집으로 가서(**/root**) 확장자가 .txt인 파일을 조회하시오!

답 : # cd
ls -l *.txt

31. 문제 30번에 출력되는 결과의 라인 수를 출력하시오!

답 : # ls -l *.txt >> list2.txt
wc -l list2.txt

32. 위의 작업을 파일을 생성하지 않고 더 간단하게 하는 방법은?

답 : # ls -l *.txt | wc -l
↑
pipe 명령어

⇒ pipe 명령어? 앞의 명령어 | 뒤의 명령어

앞의 명령어의 출력을 뒤의 명령어의 입력으로 보냄으로써 명령어의 실행결과를 다음 명령어로 전달하는 기능

33. **/root** 밑에 디렉토리(폴더)가 몇개가 있는지 조회하시오!

답 : # cd
pwd
ls -ld */ | wc -l

❖ 설명 : **ls -ld */** 라고 하면 현재 디렉토리에 있는 디렉토리 리스트들을 보여달라는 것입니다. 이 출력결과를 파이프 다음 명령어의 입력으로 전달합니다. 파이프 다음 명령어가 **wc -l**의 입력으로 전달하면 그 결과의 라인 수를 출력합니다.

34. 직업이 **SALESMAN**인 사원들의 모든 행을 출력하시오!

답: # grep -i 'salesman' emp.txt

35. 직업이 **SALESMAN**인 사원들은 전부 몇명인지 출력하시오!

답 : # grep -i 'salesman' emp.txt | wc -l

36. 직업이 **SALESMAN**인 사원들의 이름과 월급을 출력하시오!

답 : # grep -i 'salesman' emp.txt | awk '{ print \$2, \$6 }'

- ❖ 설명 : awk 명령어는 열을 선택하는 명령어입니다. \$2는 emp.txt에서 두번째 열입니다. 열과 열은 공백으로 구분되어져 있습니다.

37. 직업이 ANALYST인 사원들의 이름과 월급과 부서번호를 출력하시오!

답 : # grep -i 'analyst' emp.txt | awk '{print \$2, \$6, \$8}'

38. (오늘의 마지막 문제12/29) 권세원씨 코드를 참고해서 신뢰구간 문제를 푸는데 동전을 10번 던져서 뒷면이 0번~10번이 나오는 횟수로 신뢰구간 95%에 속하는지 여부를 출력되게 하시오!

```
import random
import numpy as np

def coin_avg_std(num):          #동전을 몇번 던지냐에 따른 평균과 표준편차를 생성
    result=[]
    for i in range(num):
        cnt = 0
        for t in range(10):
            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)      # 동전 몇번 던진지를 선택
    if c_m-2*c_s<=num<=c_m+2*c_s:
        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.'

print(coin_hypo(4))
print(coin_hypo(10))
```

39. 이름이 ALLEN인 사원의 이름과 월급을 출력하시오!

답: # grep -i 'allen' emp.txt | awk '{print \$2, \$6}'

설명 : 메모장에서 복사한 스크립트를 Oracle VirtualBox 터미널 창에 붙여넣으려면 shift + ctrl + c키를 누르면 됩니다.

40. 월급이 3000인 사원의 이름과 월급을 출력하시오!

답 : # grep '3000' emp.txt

설명 : emp.txt만 이용하면 grep만으로도 위의 문제를 해결할 수 있지만 월급이 3000인 데이터를 정확하게 검색하려면 awk를 이용해야 합니다. grep은 emp.txt에서 3000이라는 글씨가 포함되어져 있는 행을 다 출력하는 것입니다. 만약에 누군가가 커미션을 3000을 받는다면 그 사원도 조회가 되므로 그럴때는 grep을

이용하지 말고 awk로만 조회해야 합니다.

41. 부서번호가 10번이 사원들의 이름과 월급과 부서번호를 출력하시오.

답 : # grep -i '10' emp.txt | awk '{print \$2, \$6, \$8}'

- ❖ 설명 : 위와 같이 검색을 하면 ADAMS 1100 20 데이터도 같이 나옵니다. 왜 같이 나오냐면 월급 1100에 10이 포함되어져 있기 때문입니다.

따라서

정답 : awk '{print \$2, \$6, \$8}' emp.txt | grep -iw '10'

- ❖ 설명 : -w 옵션은 단어별 검색입니다.

awk로 먼저 emp.txt에서 이름과 월급과 부서번호를 선택해서 출력하고 그 출력된 결과에서 10을 포함하고 있는 행들만 출력한다.

- ❖ grep은 emp.txt에서 특정 글자가 포함되어져 있는 행들만 검색하기 때문에 특정 열에 대해서 데이터를 검색하는 것은 grep 만으로는 어렵습니다. 그래서 awk명령어를 사용해야 합니다.

42. 월급이 3000이상인 사원들의 이름과 월급을 출력하시오!

답: # awk '\$6 >= 3000 {print \$2, \$6}' emp.txt

43. 직업이 salesman이 아닌 사원들의 이름과 직업을 출력하시오!

답 : # awk '\$3 != toupper("salesman") {print \$2, \$3}' emp.txt

44. 직업이 salesman 이고 월급이 1500 이상인 사원들의 이름과 월급과 직업을 출력하시오!

답 : awk '\$3 = toupper("salesman") && \$6 >= 1500 {print \$2, \$3, \$6}' emp.txt

45. 81년도에 입사한 사원들의 이름과 입사일을 출력하시오!

답 : # awk 'substr(\$5, 3, 2) == "81" {print \$2, \$5}' emp.txt

46. 12월에 입사한 사원들의 이름과 입사일을 출력하시오!

답 : # awk 'substr(\$5, 6, 2) == "12" {print \$2, \$5}' emp.txt

외부에서 test, excel, csv를 받으면 ---> Oracle이나 하둡에 데이터를 넣고 SQL이나 Hive를 써서 데이터 검색을 하는데 그냥 바로 리눅스 명령어로 위와 같이 조회해서 데이터를 봐야할 때도 많습니다.

47. 81년도에 입사하지 않은 사원들의 이름과 입사일을 출력하시오!

답: # awk 'substr(\$5, 3, 2) != "81" {print \$2, \$5}' emp.txt

48. 이름의 첫글자가 A로 시작하는 사원들의 이름과 월급을 출력하시오.

- ❖ 정규표현식 : ^ --> 시작 , \$ --> 끝

답: # awk '{print \$2, \$6}' emp.txt | grep -i '^a'

49. 이름의 끝글자가 T로 끝나는 사원들의 이름과 월급을 출력하시오!

답 : awk '{print \$6, \$2}' emp.txt | grep -i 't\$'

설명: 월급과 이름을 뽑고 출력하고 그 문자 끝이 t인 행을 출력한다.

50. 문제 49번을 awk와 substr를 이용해서 수행하세요!

답 : awk 'substr(\$2, 5, 1)=="T" {print \$6,\$2}' emp.txt

51. 이름의 두번째 철자가 M인 사원들의 이름과 월급을 출력하시오!

답: # awk 'substr(\$2, 2, 1) == "M" {print \$2, \$6}' emp.txt

52. 이름과 월급을 출력하는데 월급이 높은 사원부터 출력하시오!

sort -nrk 6 emp.txt | awk '{print \$2, \$6}'

53. 직업이 SALESMAN인 사원들의 이름과 월급과 직업을 출력하는데 월급이 높은 사원부터 출력하시오!

답 : # awk '\$3==toupper("salesman") {print \$2, \$6, \$3}' emp.txt | sort -nrk 2

설명 : 이번에는 awk를 먼저해서 emp.txt에서 직업이 SALESMAN인 사원들의 이름과 월급과 직업을 선택하고 출력하고 이 출력을 파이프 다음에 입력으로 보내서 정렬을 하는데 입력으로 보낸 3개의 컬럼 중에 두번째 컬럼을 기준으로 정렬을 했다.

awk '\$3==toupper("salesman") {print \$2, \$6, \$3}' emp.txt 의 결과:

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

여기서 sal 컬럼은 두번째!!

54. 직업이 SALESMAN이 아닌 사원들의 이름과 월급과 직업을 출력하는데 월급이 낮은 사원 순으로 출력하시오!

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

55. 사원 테이블의 월급의 총합값을 출력하시오!

답 : awk '{print \$6}' emp.txt | awk '{sum +=\$1}END {print sum}'

설명 : 월급만 추출해서 그 출력값을 파이프 다음에 입력값으로 보냅니다. sum += \$1이 실행되면서 월급이 계속 누적이 됩니다. END로 종료되면 sum을 프린트 합니다.

56. 직업이 salesman인 사원들의 총월급을 출력하시오!

답 : grep -i 'salesman' emp.txt | awk '{print \$6}' | awk '{sum +=\$1}END {print sum}'

설명: emp.txt에서 salesman이 포함된 행만 추출해서 파이프 다음에 입력으로 보냅니다. 그리고 그 중 6번째 열만 추출해서 파이프 다음에 입력으로 보냅니다. 그리고 그 컬럼 하나의 값을 누적해서 출력한다.

57. 직업이 SALESMAN이 아니고 월급이 2000 이상인 사원들의 이름과 월급과 직업을 출력하시오!

답 : # awk '\$3 != toupper("salesman") && \$6 >= 2000 {print \$2, \$6, \$3}' emp.txt

58. 위의 사원들의 월급의 총합값을 출력하시오!

답: awk '\$3 != toupper("salesman") && \$6 >= 2000 {print \$2, \$6, \$3}' emp.txt | awk '{sum += \$2} END {print sum}'

59. (점심시간 문제) 81년도에 입사한 사원들의 월급과 토탈값을 출력하시오!

답 : # awk 'substr(\$5, 3, 2) == "81" {print \$6}' emp.txt | awk '{sum += \$1} END {print sum}'

60. 아래의 SQL의 결과를 리눅스 명령어로 출력하시오!

SQL > select ename || '의 월급은' || sal || '입니다.'
from emp;

답 : # awk '{print \$2 "의 월급은 " \$6 "입니다")' emp.txt

61. 아래의 SQL의 결과를 리눅스 명령어로 출력하시오!

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

답: awk '{print \$2 "의 직업은 " \$3 "입니다")' emp.txt

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

답 : # cd
ls -l

63. 위에서 출력된 결과에서 파일의 크기에 해당하는 부분만 출력하시오.

답 : # ls -l | awk '{print \$5}'

64. 문제 63번에서 출력된 숫자들의 총 합을 출력하시오!

답 : # ls -l | awk '{print \$5}' | awk '{sum += \$1} END {print sum}'
32438 바이트

설명 :

바이트

킬로바이트 1000^1

메가바이트 1000^2

기ガ바이트 1000^3

테라바이트 1000^4

페타바이트 1000^5

엑사바이트 1000^6

제타바이트 1000^7

요타바이트 1000^8

65. emp.txt에서 직업만 출력하시오!

답 : # awk '{print \$3 }' emp.txt

66. 문제 65번의 결과를 abcd순으로 정렬해서 출력하시오!

답 : # awk '{print \$3 }' emp.txt | sort -k 1

문자만 정렬하는 거라서 n을 적지 않고 k만 적어준다.

67. 문제 66번의 결과를 중복제거해서 출력하시오!

답 : # awk '{print \$3 }' emp.txt | sort -k 1 | uniq

정렬을 다 해놓고 uniq 써서 중복제거 해준다.

설명 : uniq 사용하기 전에 앞에서 정렬을 한 이유는 위 아래가 같은 데이터만 중복을 제거하기 때문입니다.

68. 사원 테이블에서 부서번호를 출력하는데 중복을 제거해서 출력하시오!

답 : # awk '{print \$8}' emp.txt| sort -nk 1 | uniq

69. 직업이 CLERK인 사원들의 부서번호를 출력하는데 중복제거해서 출력하시오!

답 : # awk '\$3 == toupper("clerk") {print \$8}' emp.txt | sort -nk 1 | uniq

70. 직업을 물어보게 하고 직업을 입력하면 해당 직업을 갖는 사원들의 이름과 직업이 출력되게 하시오~

```
echo -n 'Enter the job name~~~'  
read job  
grep -i $job emp.txt | awk '{print $2, $3}'
```

설명 : -n 은 엔터같은 역할입니다.

```
# sh job2.sh  
Enter the job name~~~ salesman  
salesman이라는 글씨가 job변수에 입력되었다.  
$를 job 앞에 써줘야 job을 변수로 인정하는 것입니다.  
$를 안써주면 그냥 영어 철자 job일 뿐입니다.
```

위의 리눅스 명령어 여러개를 파일로 저장해서 한번에 수행하는 방법:

vi 편집기 화면으로 들어가서 위의 3줄을 복사해서 붙여넣으려면 알파벳 i를 누르고 아래쪽에 insert가 나오는지 확인한다.

그리고 shift +ctrl+c 혹은 shift + insert 눌러서 붙여넣고 저장을 해야하는데 저장을 하려면 esc 키를 눌러서 아래쪽에 insert를 사라지게 하고 대문자 Z를 두번 연속으로 눌러서 저장하고 빠져나오면 된다.

```
#vi job2.sh  
#i 누르고 복사해서 shift + ctrl+ c  
#esc 누르기  
대문자 Z 두번 연속 누르기
```

그리고 sh job2.sh 입력

Enter the job name~~~ 나오면 원하는 job의 이름을 입력해준다.

sh job2.sh <---- 확장자가 .sh는 쉘스크립트입니다.

쉘스크립트는 리눅스 명령어를 모아놓은것입니다.

쉘스크립트를 잘 작성할 줄 알면 업무의 많은 부분을 자동화 할 수 있습니다.

sh 명령어로 쉘스크립트를 실행합니다.

71. 아래와 같이 부서번호를 물어보게 하고 부서번호를 입력하면 해당 부서번호인 사원들의 이름과 월급과 부서번호가 출력되게 하시오.

```
답 : # echo -n 'Enter the deptno ~~'  
      read deptno  
      awk -v num=$deptno '$8==num{print $2, $6, $8}' emp.txt
```

설명 : awk의 -v 옵션은 변수의 값을 다른 변수에 할당할 때 사용하는 옵션입니다.

```
# vi deptno.sh  
위의 3줄을 복사해서 저장합니다.  
# sh deptno.sh
```

72. 이름이 SCOTT인 사원의 이름과 월급을 출력하시오!

답 : # awk '\$2 == toupper("scott") {print \$2, \$6}' emp.txt

73. 이름을 물어보게 하고 이름을 입력하면 해당 사원의 이름과 월급이 출력되게 쉘스크립트를 만들고 아래와 같이 실행되게 하시오.

결과:

```
# sh find_sal.sh  
Enter the ename ~~ scott  
SCOTT 3000
```

답:

```
# echo -n 'Enter the ename~~'  
read ename  
awk -v ename=$ename '$2==toupper(ename){print $2, $6}' emp.txt
```

grep보다 awk가 정확하다.

74. dept.txt를 리눅스 서버에 올리시오~

(/root/밑에 올리세요~)

```
# ls -l dept.txt
```

75. dept.txt에서 부서번호가 10번의 부서위치가 어딘지 출력하시오!

답 : # awk '\$1==10{print \$3}' dept.txt

결과: NEWYORK

76. SCOTT이 근무하는 부서위치를 출력하시오!

SQL > select d.loc

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

답: # deptno=`awk '\$2==toupper("scott"){print \$8}' emp.txt`

설명 : 양쪽 역따옴표 안쪽에 있는 명령어로 수행한 결과를 deptno변수에 할당하겠다.

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

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

```
# sh find_loc.sh
```

```
Enter the name ~~ scott  
DALLAS
```

답: #vi sh find_loc.sh

```
echo -n "Enter the name~~"  
read ename  
deptno=`awk -v num1=$ename '$2==toupper(num1){print $8}' emp.txt`
```

```
awk -v num=$deptno '$1==num{print $3}' dept.txt  
# sh find_loc.sh
```

78. 직업이 SALESMAN인 사원들의 월급을 출력하시오!

답 : awk '\$3=="SALESMAN"{print \$6}' emp.txt

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

답 : awk '\$3=="SALESMAN'{print \$2, \$6}' emp.txt

80. 문제 79번의 스크립트를 이용해서 shell script를 만드는데 직업을 물어보게하고 직업을 입력하면 해당 직업 인 사원들의 이름과 월급이 출력되게하시오!

```
# sh find_job.sh
```

Enter the job ~~ salesman

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

답:

```
#vi find_job.sh
```

```
echo -n "Enter the job~~"  
read job  
awk -v num=$job '$3==toupper(num) {print $2, $6}' emp.txt
```

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

답:

```
#vi find_job.sh
```

```
echo -n "Enter the job~~"  
read job  
awk -v num=$job '$3==toupper(num) {print $0}' emp.txt
```

82. 직업을 물어보게하고 직업을 입력하면 해당 직업을 가진 사원들의 모든 데이터로 텍스트 파일을 생성되게 하시오!

Enter the job ~~ salesman

```
# ls -l salesman.txt
```

```
echo -n "Enter the job~~"  
read job  
awk -v num=$job '$3==toupper(num) {print $0}' emp.txt >> $job.txt
```

83. (오늘의 마지막 문제 12/30) 부서번호를 물어보게하고 부서번호를 입력하면 해당 부서번호의 모든 사원의 데이터가 텍스트 파일로 생성되게 하시오!

Enter the deptno ~ 10

```
# ls -l 10.txt <--- 10번 부서번호인 사원들이 모든 데이터가 있어야 합니다.
```

```
답: # vi find_deptno.sh  
echo -n "Enter the deptno~"  
read deptno
```

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

84. 이름이 SCOTT인 사원의 부서위치를 출력하시오!

- SCOTT의 부서번호를 emp.txt에서 알아낸다.
awk '\$2==toupper("scott") {print \$8}' emp.txt
- SCOTT의 부서번호로 부서위치(loc)를 dept.txt에서 알아낸다.
deptno=`awk '\$2==toupper("scott") {print\$8}' emp.txt`
echo \$deptno
loc = `awk -v num=\$deptno '\$1==num{print\$3}' dept.txt`
echo \$loc

85. 문제 84번의 스크립트를 이용해서 이름을 물어보게 하고 이름을 입력하고 해당 사원이 근무하는 부서위치가 출력되게 쉘 스크립트를 작성하시오!

```
# sh find_loc.sh  
Enter the ename ~~ scott  
DALLAS
```

#어제 푼 문제 참조하기

86. dept.txt에서 DALLAS의 부서번호를 출력하시오!

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

87. 위의 부서번호를 가지고 emp.txt에서 해당 부서번호에서 근무하는 사원들 이름과 직업을 출력하시오!

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

설명 : awk의 -v 옵션은 변수에 있는 값을 다른 변수에 담을 때 사용하는 옵션

88. 위의 스크립트를 이용해서 부서위치를 물어보게 하고 부서위치를 입력하면 해당 부서위치인 사원들의 이름과 월급이 출력되게 하시오

```
# sh find_loc2.sh
```

Enter the loc: dallas

JONES MANAGER
FORD ANALYST
SMITH CLERK
SCOTT ANALYST
ADAMS CLERK

답:

```
# 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
```

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

Enter the loc: dallas

```
# ls -l dallas.txt <-- dallas 에서 근무하는 사원들의 모든 행이 emp.txt에서 불러와져서 있어야합니다.
```

답:

```
# vi find_loc3.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 >> $loc.txt
```

90. 어제 마지막 문제로 만들었던 `find_deptno.sh`를 실행해서 `20.txt`를 생성하시오!

답 : # sh find_deptno.sh
Enter the deptno: 20

91. `emp.txt`와 `20.txt`의 차이를 확인하시오!

답: diff 20.txt emp.txt

>> 공통된 부분 빼고 차이있는 부분(deptno가 10이거나 30인 경우에 대해서 나옴)

92. 집에서(`/root`) `test100`이라는 디렉토리를 만들고 `/root/test100` 디렉토리 밑에 `emp.txt`를 복사하시오~

답 : # cd
pwd
mkdir test100
cp /root/emp.txt /root/test100/emp.txt

설명 : `/root/emp.txt`를 `/root/test100/emp.txt`로 복사해라~

93. `/root` 밑에 `emp.txt`가 있는지 `find` 하시오!

답: # find /root -name 'emp.txt' -print

결과:

```
/root/Desktop/emp.txt
/root/emp.txt
/root/test100/emp.txt
```

94. 문제 93번을 다시 수행하는데 `/root` 밑에 하위디렉토리까지 검색하지 말고 그냥 바로 `/root` 밑에 `emp.txt`가 있는지 검색하시오.

답: #find /root -maxdepth 1 -name 'emp.txt' -print

-maxdepth 1 : root 바로 밑에 있는 파일만 보여줌

95. 위의 명령어를 이용해서 아래와 같이 검색을 편하게 할 수 있도록 shell 스크립트를 만드시오!

답:

```
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
```

96. 집에서(`/root`) `emp.txt`를 복사해서 `emp1.txt`, `emp2.txt`, `emp3.txt`를 생성하시오.

답:
cp emp.txt emp1.txt
cp emp.txt emp2.txt

```
# cp emp.txt emp3.txt  
# ls -l emp*.txt (만들어졌는지 확인)
```

97. 현재 디렉토리에 emp로 시작하는 모든 text파일을 empall.tar라는 이름으로 압축하시오!

답: # tar cvf empall.tar ./emp*.txt
 ↑
 현재 디렉토리

```
#ls -l empall.tar
```

98. 현재 디렉토리에 test200이라는 디렉토리를 만들고 empall.tar 파일을 test200 디렉토리에 카피하시오!

답: # cd
 # mkdir test200
 # cp empall.tar ./test200

99. 현재 디렉토리에서 test200 디렉토리로 이동해서 empall.tar파일의 압축을 해제하시오.

답 : # cd
 # cd ./test200
 # pwd
 # tar xvf empall.tar
 # ls

100. ./root 밑에 있는 dept.txt를 dept1.txt, dept2.txt, dept3.txt로 복사하고 /root 밑에 있는 dept로 시작하는 텍스트 파일들을 deptall.tar로 압축한 후에 /root밑에 있는 test500이라는 폴더를 만들고 deptall.tar를 test500밑에 복사하고 압축을 해제하시오.

답 :
cd
cp dept.txt dept1.txt
cp dept.txt dept2.txt
cp dept.txt dept2.txt
tar cvf deptall.tar ./dept*.txt
mkdir test500
cp deptall.tar ./test500/ (마지막에 / 꼭 써주기! 안써주면 파일 자체가 복사됨)
cp ./test500
tar xvf deptall.tar

101. 윈도우에서 emp.txt와 dept.txt를 압축을 합니다.

압축 파일명은 fileall.tar로 압축하시오!

102. fileall.tar 압축하일을 리눅스 바탕화면에 가져다 놓으세요. 바탕화면 HOME폴더 안에 fileall.tar를 복사해 놓으세요.

103. /root 밑에 있는 fileall.tar를 압축해제하세요.

답: # tar xvf fileall.tar

104. 윈도우에서 jobs.txt를 jobs2.txt로 복사한 후에 두개의 파일을 jobs.zip으로 압축하고 리눅스 서버에 바탕 화면에 home 폴더에 넣으시오!

답: # cd
 # mkdir test800
 # cp jobs.zip ./test800/
 # cd test800
 # ls

```
# unzip jobs.zip
```

만약 리눅스 바탕화면 디렉토리를 터미널창에서 이동하고 싶다면?

```
# cd /root/Desktop  
# pwd  
# ls
```

105. 위에서 KING의 이름을 yyy로 변경해서 보여주고 있는 결과화면을 저장해서 emp900.txt로 저장되게 하시오!

답 : # sed 's/KING/yyy/' emp.txt >> emp900.txt

106. /root 밑에 있는 find_loc2.sh를 실행해보시오!

```
# sh find_loc2.sh
```

107. find_loc2.sh를 위의 예제에서 만든 a.sh 의 3번으로 추가하시오~ (#22 case명령어 예제)

답:

```
echo " Press number 1 for confirm employees sal  
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_loc2.sh ;;  
esac
```

108. /root/ 밑에 있는 확장자가 .txt로 파일을 전부 /root/ 밑에 backup이라는 폴더를 만들고 거기로 다 복사 하시오.

답:

```
#cd  
#pwd  
#mkdir backup  
#cp *.txt ./backup/      '/'를 넣어주는 것이 중요!!
```

109. 집으로 와서 backup2 폴더를 만들고 현재 디렉토리(집)에 있는 텍스트 파일들을 전부 backup2 폴더로 이동시키시오!

```
# cd  
# mkdir backup2  
# mv *.txt ./backup2/  
# ls *.txt >>> 이렇게 하면 No such file or directory 문구가 떠야 정상
```

110. (오늘의 마지막 문제 12/31) 리눅스 환경에서 python 스크립트를 수행하시오~

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

```
# vi a.py 해서 저장  
# python a.py
```

리눅스 환경에서 아래의 문제를 풀고 결과 화면을 캡쳐해서 올리세요!

주사위를 던져서 주사위의 눈이 3이 나올 확률을 출력하시오~ (주사위를 10000번 던지세요)



111. 동전을 10만번 던져서 앞면이 10만번중에 앞면이 몇번 나오는지 파이썬으로 실험하시오!

```
$ vi coin_cnt.py
```

```
-----  
import random
```

```
coin = ['앞면', '뒷면']  
cnt = 0  
  
for i in range(1, 10001):  
    result = random.choice(coin)  
    if result == '앞면':  
        cnt = cnt+1
```

```
print(cnt)
```

```
-----  
$ python coin_cnt.py
```

112. jobs.txt를 열어서 맨 위에 한 줄을 복사해서 다음라인에 붙여넣으시오!

```
$ vi jobs.txt
```

```
yy 누르고 p 누르기
```

113. jobs.txt를 열어서 전체를 복사한 후에 맨 밑에 전체를 다 붙여넣으시오!

```
답 : $ vi jobs.txt 를 연다.
```

```
yG를 눌러서 전체복사를 한다.
```

G를 누르고 맨 아래로 내려온다.

p를 누르고 불여넣는다.

dG를 누르면 커서위치부터 맨 아래까지 다 지운다.

114. jobs.txt를 열어서 위아래 좌우로 이동해 봅니다.

j : 아래

k : 위

h : 왼쪽

l : 오른쪽

115. jobs.txt를 열어서 라인마다 숫자번호가 붙게 하시오!

답 : :set nu

116. jobs.txt에서 1번째 라인~ 5번 라인까지 복사해서 jobs2.txt로 생성하시오!

답 : :1,5 w jobs3.txt

117. jobs.txt를 열어서 1번 라인~20번라인까지의 내용을 jobs7.txt로 생성하시오!

답 :

\$ vi jobs.txt

: set nu

: 1,20 w jobs7.txt

\$ wc -l jobs7.txt <<< 20라인이 맞는지 확인

118. (vi 편집기 명령어 연습) 구구단 2단을 출력하는 파이썬 코드를 리눅스에서 직접 작성하고 아래와 같이 실행되게 하시오!

\$ python p_2_dan.py

2 x 1 = 2

2 x 2 = 4

: :

2 x 9 = 18

답:

\$ vi p_2_dan.py

for i in range(1,10):

 print('2 x ', i, ' = ', 2*i)

\$ python p_2_dan.py

119. 구구단 전체를 출력하시오!

\$ python gugu_dan.py

2 x 1 = 2

2 x 2 = 4

: :

9 x 9 = 81

답:

```
$ vi gugu_dan.py
```

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

```
$ python gugu_dan.py
```

120. **emp.txt**를 열어서 SCOTT이라는 문자가 있는지 검색하시오!

답:

```
$ vi emp.txt
: /SCOTT 엔터
```

121. **\$ vi emp.txt**

: s/KING/aaa/g 된 상태에서 다시 aaa를 KING으로 변경하시오!

답:

```
$ vi emp.txt
: s/aaa/KING/g
```

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

답:

```
$ vi emp.txt 로 emp.txt를 ↵ 르어서
```

ALLEN의 행의 직업쪽에 커서를 대고

:s/SALESMAN/bbbb 라고 하고 엔터

123. **emp.txt**를 열고 직업이 MANAGER인 사원들의 직업을 모두 SALESMAN으로 변경하시오!

답:

```
$ vi emp.txt
```

: %s/MANAGER/SALESMAN/g 엔터

124. **emp.txt**를 **emp1.txt ~ emp20.txt**로 복사하시오!

답:

```
$ cp emp.txt emp1.txt
$ cp emp.txt emp2.txt
$ cp emp.txt emp1.txt
:
$ cp emp.txt emp20.txt
```

125. **emp1.txt ~ emp20.txt**의 내용중에서 SALESMAN을 jjj로 변경하시오!

답 : \$ vi emp*.txt

:argdo %s/SALESMAN/jjj/g | update

설명 : argdo를 이용하면 여러개의 파일을 한번에 변경할 수 있습니다.

126. 다시 **emp1.txt ~ emp20.txt**를 열어서 jjj를 SALESMAN으로 변경하시오!

답:

```
$ vi emp*.txt  
: argdo %s/SALESMAN/jjj/g | update 엔터
```

127. jobs.txt를 jobs1.txt ~ jobs10.txt로 복사하시오!

```
$ cp jobs.txt jobs1.txt  
$ cp jobs.txt jobs2.txt  
:  
$ cp jobs.txt jobs10.txt
```

128. jobs1.txt ~ jobs10.txt 를 열어서 about를 aaa로 변경하시오!

```
$ vi jobs*.txt  
: argdo %s/about/aaa/g | update
```

129. 숫자 1부터 10까지 출력하는 a20.py라는 파이썬 파일을 생성하시오!

```
$ python a20.py
```

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

```
$ ls -l a20.py 해보면
```

```
-rw-rw-r--. 1 scott scott 32 Jan 4 15:54 a20.py 이렇게 나옴
```

설명: 소유자는 읽고(r) 쓰는(w) 권한을 가지고 있고 실행권한은 없습니다.

그룹 유저들도 읽고(r) 쓰는(w) 권한을 가지고 있고 실행권한은 없습니다.

기타 유저들은 읽을 수 있는(r) 권한만 있습니다.

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

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

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

결과 :

```
-rwxrwxr--. 1 scott scott 32 Jan 4 15:54 a20.py
```

설명 : u+x --> 소유자에게 실행권한을 주겠다.

u-x --> 소유자가 가지고 있는 실행권한을 빼겠다.

u+r --> 소유자에게 읽기 권한을 주겠다.

u-r --> 소유자에게 읽기 권한을 빼겠다.

u+w --> 소유자에게 쓰기 권한을 주겠다.

u-w --> 소유자에게 쓰기 권한을 빼겠다.

u+wx --> 소유자에게 읽기, 쓰기, 실행권한을 주겠다.

실행권한 : sh로도 실행될 수 있는 권한

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

답 : \$ chmod u-rwx a20.py
\$ ls -l a20.py

결과:

```
----rw-r--. 1 scott scott 32 Jan 4 15:54 a20.py
```

\$ vi a20.py 하면 permission denied 가 뜨면서 파일이 안 열림.

132. a20.py의 소유가 a20.py를 읽고 쓸 수 있도록 권한을 넣고 실행 권한은 넣지 마시오!

답 : \$ chmod u+rwx a20.py

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

133. a20.py의 소유자는 a20.py를 읽고, 쓰고, 실행할 수 있게 하고 그룹유저들과 기타 유저들은 아무것도 못하게 하시오!

답: \$ chmod u+rwx,g-rwx,o-rwx a20.py

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

결과:

```
-rwx-----. 1 scott scott 32 Jan 4 15:54 a20.py
```

설명 : u+rwx,g-rwx,o-rwx 는 꼭 붙여서 써주기!!!!

root 유저는 scott과는 다르게 모든 파일에 대해 읽고, 쓰고, 실행할수 있는 권한을 가질 수 있도록 chmod 명령어를 수행할 수 있습니다.

134. 터미널 창에서 root 유저로 접속하시오!

```
$ whoami >>> scott
```

```
$ su -
```

패스워드 : 1234 <--- 패스워드 쳐도 안보임

```
#whoami
```

```
#pwd
```

/root <--- root의 집으로 나옵니다.

135. 다시 터미널 창에서 scott으로 접속하시오!

답 : # su - scott ('-' 양 옆으로 한칸씩 띄우기)

```
$ whoami
```

```
$ pwd
```

136. 다시 터미널 창에서 root로 접속하고 scott의 집인 /home/scott으로 이동하시오!

답 : \$ su -

```
# cd /home/scott
```

```
# pwd
```

설명 : root는 최고권한자이기 때문에 얼마든지 a20.py를 볼 수도 있고 권한도 변경할 수 있습니다.

137. root가 a20.py의 소유자가 가지고 있는 읽고, 쓰고, 실행할 수 있는 권한을 모두 빼시오~

```
# cd /home/scott
```

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

```
# ls -l a20.py
```

결과:

```
-----. 1 scott scott 32 Jan 4 15:54 a20.py
```

138. 다시 scott으로 터미널창에서 접속해서 scott에서 a20.py를 열어보시오~

```
# su - scott
```

설명 : permission denied 라고 나오면서 열리지 않습니다. 그래도 소유자가 scott이므로 권한을 다시 자신이 직접 넣을 수 있습니다.

139. scott 유저에 다시 a20.py에 대해 읽고, 쓰고, 실행할 수 있는 권한을 넣으시오!

답 : \$ chmod u+rwx a20.py
\$ ls -l a20.py

140. (오늘의 마지막 문제 1/4) 리눅스에서 spyder를 실행해서 아래의 통계문제를 풀고 화면 캡쳐를 해서 올리세요~

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

동전을 10번 던졌을 때 뒷면이 나오는 횟수가 0번이 나올 확률은 신뢰구간 95%안에 없습니다.
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 있습니다.
동전을 10번 던졌을 때 뒷면이 나오는 횟수가 8번이 나올 확률은 신뢰구간 95%안에 있습니다.
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.



\$ activate py389
\$ spyder

해주면 spyder가 열림

141. dept.txt의 권한이 어떻게 되어 있는지 확인하시오.

```
ls -l dept.txt  
-rw-r--r--. 1 root root 82 Jan 5 09:47 dept.txt
```

142. dept.txt에서 권한이 아래와 같이 나타나게 하시오!

```
$ ls -l dept.txt
```

답: \$ chmod g-w,o-w dept.txt --- 실행안됨

➤ chown 명령어 먼저 실행 후 해야함

143. dept.txt의 소유자 scott으로 변경하고 그룹도 scott으로 변경하시오!

```
답 : # chown scott:scott dept.txt  
# ls -l dept.txt  
-rwxrwxrwx. 1 scott scott 104 Jan 4 12:09 dept.txt
```

144. 다시 scott으로 switch user하시오!

```
답 : # su - scott  
$ whoami
```

145. emp.txt의 권한을 아래와 같이 나타나게 하시오!

```
$ ls -l emp.txt  
_rwx r-x r-x 1 scott scott emp.txt
```

답 : \$ chmod g-w,o-w emp.txt
\$ ls -l emp.txt

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

```
$ ls -l dept.txt  
_rwx r-x r-x 1 scott scott dept.txt
```

답: \$ chmod g-w,o-w dept.txt
\$ ls -l dept.txt

147. dept.txt의 권한이 아래와 같이 나타나게 하시오!

```
_rwx --- --- 1 scott scott dept.txt
```

답 : \$ chmod g-rx,o-rx dept.txt

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

```
$ ls -l emp.txt  
-rw- rw- rw-. 1 scott scott 1032 Jan 4 15:08 emp.txt
```

답:

```
$ chmod 666 emp.txt  
4+2 라서 6이다.
```

149. emp.txt에 대해서 아래와 같이 권한 들어가게 하시오!

```
$ ls -l emp.txt  
-rwx rwx rwx. 1 scott scott 1032 Jan 4 15:08 emp.txt
```

답 : \$ chmod 777 emp.txt

설명 : $4+2+1 = 7$ 이라서!!

150. **emp.txt**에 대해서 아래와 같이 권한이 들어가게 하시오!

```
$ ls -l emp.txt  
---x --x --x. 1 scott scott 1032 Jan 4 15:08 emp.txt
```

답 : chmod 111 emp.txt

151. **emp.txt**에 대해서 아래와같이 권한이 들어가게 하시오!

```
$ ls -l emp.txt  
-rwx r-x --x. 1 scott scott 1032 Jan 4 15:08 emp.txt
```

답: chmod 751 emp.txt

152. **emp.txt**에 대해서 아래와같이 권한이 들어가게 하시오!

```
$ ls -l emp.txt  
---- --- ---. 1 scott scott 1032 Jan 4 15:08 emp.txt  
답: chmod 000 emp.txt
```

153. **emp.txt**에 대해서 아래와 같이 권한이 들어가게 하시오!

```
$ ls -l emp.txt  
-rwx rw- rw-. 1 scott scott 1032 Jan 4 15:08 emp.txt
```

답 : \$ chmod 766 emp.txt

154. **/home/scott** 밑에 있는 **test500** 디렉토리와 그 밑에 있는 파일들의 소유자를 모두 **scott**으로 변경하시오!

```
# cd /home/scott  
# chown -R scott:scot test500  
# ls -ld test500  
# ls -l test500
```

155. **emp.txt**에 대해서 아래와 같이 권한이 유지되게하고 root 유저 외에는 **chmod**을 할 수 없도록 막으시오!

```
$ ls -l emp.txt  
-r--r--r--. 1 scott scott 1032 Jan 5 11:12 emp.txt
```

답: \$ chmod 444 emp.txt
\$ su -
cd /home/scott
chattr +i emp.txt

156. 다시 **emp.txt**를 **scott** 유저가 **chmod**할 수 있도록 변경하시오!

```
# chattr -i emp.txt  
# lsattr emp.txt
```

157. **dept.txt**에 대해서 **chmod**을 할 수 없도록 막으시오! 그리고 **dept.txt**의 권한을 아래와 같이 유지되게 하시오!

```
$ ls -l dept.txt
```

```
-rw-r--r--. 1 scott scott 1032 Jan 5 11:12 dept.txt
```

답: \$ chmod 644 dept.txt
\$ su -

```
# cd /home/scott  
# chattr +i dept.txt
```

158. (점심시간 문제) 숫자 1부터 20까지 출력하는 파이썬 코드를 작성하고 그 파이썬 코드의 권한이 아래와 같이 들어가게하시오

```
$ ls -l a158.py  
-rwxrw-rw-. 1 scott scott 1032 Jan 5 11:12 a158.py
```



159. /home/scott 밑에 있는 확장자 .sh 로 끝나는 파일들의 총 크기를 확인하시오 !

```
$ du -c *.sh  
541540      Anaconda3-2020.11-Linux-x86_64.sh  
541540      total
```

160. 아래와 같이 1~10000000000 숫자를 출력하는 파이썬 프로그램을 돌리고 다른 터미널 창에서 sar로 모니터링하시오!

설명: sar 명령어 결과의 컬럼 소개

%user : scott 유저와 같이 일반 유저가 사용하는 disk i/o 입니다.

악성 SQL을 수행하거나 무한루프를 돌리면 %user 사용율이 올라갑니다.

%nice : cpu 를 양보하는 친절도

%system : system 이 사용하는 disk i/o

%iowait : i/o 를 일으키면서 얼마나 대기하는지

%idle : 작업을 안하고 있는 idle 한 상태

%steal : 다른 프로세서의 자원을 얼마나 뺏고 있는지

161. 내가 리눅스 쉘 프로그램을 만들고 돌린 후에 회의를 가거나 퇴근을 해서 내가 돌린 쉘 프로그램이 시스템

에 부하를 일으키고 있는지 확인할 수 없다면 어떻게 해야하는가?

답: sar 명령어로 수행한 결과를 text파일로 생성되게 한다.

```
$ sar 1 20 >> sar_20210105.txt
```

vi sar 누르고 텁 누르면 자동으로 써진다.

162. 만약에 위의 명령어를 백그라운드로 돌아가게 하고 싶다면?

```
$ sar 1 20 >> sar_20210106.txt &
```

163. 아래와 같이 여러개의 jobs를 만드시오.

```
$ vi hhh2.txt
```

select

esc 키 누르고 ctl + z 를 누른다.

```
$ vi hhh3.txt
```

select ename

esc 키 누르고 ctl + z 를 누른다.

```
$ vi hhh4.txt
```

select ename, sal

```
$ jobs
[1] Stopped vim hhh.txt
[2]- Stopped vim hhh3.txt
[3]+ Stopped vim hhh4.txt
```

❖ 설명 : + : 현재 job

- : 현재 이전에 현재 job이었던 job

164. 위의 상태에서 2번 job에 들어가고 싶다면?

```
$ fg 2
```

165. 위의 상태에서 아래의 백그라운드로 돌리는 sar 명령어를 수행하고 출력되는 결과를 해석하시오!

```
$ sar 1 100 >> sar_2021.txt
```

[4] 12006

↑ ↑

job번호 프로세서 번호

166. 현재 리눅스 시스템에 떠있는 모든 프로세서를 다 출력하시오!

```
$ ps -ef | grep scott (scott이 포함되어져 있는 걸 다 보여달라)
```

167. 현재 리눅스 시스템에 떠있는 프로세서들 중에서 vim으로 vi편집기 작업중이었던 프로세서들만 조회하시오!

```
$ ps -ef | grep vim
```

168. 위의 vim으로 작업중이었던 세션을 모두 다 죽이시오!

```
$ kill -9 11701
```

```
$ kill -9 11866
```

169. 숫자 1~100000000000 를 출력하는 파이썬 프로그램을 실행하고 다른 터미널 창을 열어서 top을 수행합니다.

다른 터미널 창을 열어서 top 수행합니다.

170. 지금 cpu를 많이 사용하고 있는 프로세서 25283 프로세서를 강제 종료하시오!

```
$ kill -9 25283
```

171. num1과 num2의 곱을 구하세요~

답 : \$ expr \$num1 * \$num2

설명 : 곱하기(*)와 괄호 앞에는 원화(₩)를 붙여줘야 합니다.

172. 아래의 계산식을 구현하시오!

```
($num2 + 200) *$num1
```

답 :

```
expr $( $num2 + 200 ) *$num1
```

설명: 웬만하면 한칸씩 빼고 대입연산자만 양쪽에 딱 붙입니다.

173. (오늘의 마지막 문제 1/5) 아래의 두 수를 곱한 결과가 출력되는 쉘 스크립트를 작성하시오!



174. 위의 스크립트에 파라미터 변수를 이용해서 아래와 같이 실행될 수 있게 하시오.

```
$ sh if1.sh 100 200
```

100과 200은 같지 않습니다.

\$ sh if1.sh 100 100

100과 100은 같습니다.

답:

\$ vi if1.sh

```
if [ $1 -eq $2 ]; then
    echo "$1과 $2은 같습니다."
else
    echo "$1과 $2은 같지 않습니다."
fi
```

\$ sh if1.sh 100 200

\$ sh if1.sh 100 100

175. emp.txt에서 scott의 월급을 출력하시오!(awk를 이용하시오!)

답:

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

176. 위의 스크립트와 파라미터 변수를 이용해서 아래와 같이 실행되는 쉘 스크립트를 작성하시오!

\$ sh find_sal.sh scott

scott의 월급은 3000입니다.

답:

```
sal=`awk -v name=$1 '$2==toupper(name) {print $6}' emp.txt`
echo "$1의 월급은 $sal입니다"
```

>>> ``써서 sal 안에 ``에 있는 문장이 들어가게 한다.

177. 위의 스크립트를 이용해서 아래와 같이 실행되게 쉘 스크립트를 생성하시오!

\$ sh find_job.sh allen

allen의 직업은 SALESMAN입니다.

답:

```
job=`awk -v name=$1 '$2==toupper(name) {print $3}' emp.txt`
echo "$1의 직업은 $job입니다."
```

178. 위의 스크립트와 if문을 이용해서 아래의 쉘 스크립트를 생성하시오. 월급 2000을 기준으로 월급의 인상여부를 출력하게 하시오! (2000이상이 월급 인상대상자가 아닙니다.)

\$ sh find_sal.sh scott

월급 인상 대상자가 아닙니다.

\$ sh find_sal.sh martin

월급 인상 대상자입니다.

답:

\$ vi find_sal.sh

```
sal=`awk -v name=$1 '$2==toupper(name) {print $6}' emp.txt`
```

```
if [ $sal -gt 2000 ]; then
```

```
    echo "월급 인상 대상자가 아닙니다."
```

```
else
```

```
    echo "월급 인상 대상자 입니다."
fi
```

```
$ sh find_sal.sh scott
$ sh find_sal.sh martin
```

179. 위의 스크립트를 수정해서 부서번호가 30번이고 월급이 2000보다 작다면 "월급 인상 대상자 입니다"를 출력하고 그렇지 않다면 "월급 인상 대상자가 아닙니다"를 출력하시오!

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:0:5} -lt 2000 ] && [ ${deptno:0:9}=='30' ]; then
    echo "월급 인상 대상자 입니다."
else
    echo "월급 인상 대상자가 아닙니다."
fi
```

180. 구구단 2단을 출력하세요!

```
2 x 1 = 2
2 x 2 = 4
:
2 x 9 = 18
```

답:

```
for i in {1..9}
do
    let gop=2*$i
    echo "2 x $i = $gop"
done
```

#리눅스의 for loop문
loop문의 시작
let 2*\$i 의 결과를 gop 변수에 할당한다
#loop 문의 끝

181. 구구단 전체를 출력하시오!

답:

```
$ vi gugudan.sh
```

```
for i in {2..9}
do
    for k in {1..9}
    do
        let gop=$i*$k
        echo "$i x $k = $gop"
    done
done
```

```
$ sh gugudan.sh
```

182. 아래의 별표를 출력하는 쉘스크립트를 작성하시오!

```
$ sh star.sh
```

```
★
★★
★★★
```

★★★★★
★★★★★

답:

```
star="" # 더블 쿼테이션 마크 2개를 붙여서 null값을 나타냄
for i in {1..5} # 1~5까지 loop를 돌리겠다.
do      # 루프문 시작 부분
    star="$star★" # star = ""★ ---> star=★
    echo $star     # star = 변수에 있는 값을 출력해라~
done      # 루프문 종료 부분
```

설명: i = 1 일때 star=★

i = 2 일때 star=★★

i = 3 일때 star=★★★

i = 4 일때 star=★★★★

i = 5 일때 star=★★★★★

183. 위의 코드를 수정해서 숫자를 물어보게 하고 숫자를 입력하면 해당 숫자만큼 ★이 출력되게 하시오!

```
$ sh star.sh
숫자를 입력하세요~ 5
★
★★
★★★
★★★★
★★★★★
```

답:

```
$ vi star.sh
```

```
echo -n "숫자를 입력하세요~"
read num
star=""
for i in `eval echo {1..$num}`
do
    star="$star★"
    echo $star
done
```

설명: for 반복문안에서 echo {1..\$num}이 실행되게하려면 eval을 써줘야 합니다.

184. emp.txt를 복사해서 emp1.txt~emp100.txt로 복사하는 쉘 스크립트를 작성하시오!

```
$ vi cp_emp.sh
for i in {1..100}
do
    cp emp.txt emp$i.txt
done
$ sh cp_emp.sh
```

185. (점심시간 문제) 지금 복사한 emp1.txt ~ emp100.txt 의 이름을 employee1.txt~ employee100.txt로 변

경하시오~

(이름 변경하는 명령어는 mv입니다)

```
$ sh mv_emp.sh
```

답:

```
$ vi mv_emp.sh
```

```
for i in {1..100}
do
    mv emp$i.txt employee$i.txt
done
```

```
$sh mv_emp.sh
```

186. **employee1.txt ~ employee100.txt** 중 하나를 랜덤으로 골라서 파일을 연 후에 숫자 3000을 3900으로 변

경하시오~

답:

```
$ vi employee77.txt
```

```
:%s/3000/3900/g
```

187. **emp.txt와 employee80.txt의 파일의 차이가 있는지 확인하시오!**

```
$ diff --brief emp.txt employee80.txt
```

Files emp.txt and employee80.txt differ <-- 차이가 있으면 출력이 됩니다.

```
$ diff --brief emp.txt employee81.txt <-- 차이가 없으면 출력이 안됩니다.
```

188. **for loop문을 이용해서 employee1.txt ~ employee100.txt** 중에 emp.txt와 차이가 있는 파일이 무엇인지 한번에 알아내시오!

```
$ sh diff.sh
```

답 : \$ vi diff.sh

```
for i in {1..100}
do
    diff --brief emp.txt employee$i.txt
done
```

결과:

Files emp.txt and employee77.txt differ
Files emp.txt and employee100.txt differ

189. **employees로 시작하는 txt파일을 모두 지우시오!**

답 : \$ rm employee*.txt

```
$ ls
```

190. **ls -l find_sal.sh를 했을 때 출력되는 결과에서 크기에 해당하는 부분만 출력하시오!**

답 : \$ ls -l find_sal.sh |awk '{print \$5}'

191. size100이라는 디렉토리를 /home/scott 밑에 만드시오.

```
$ mkdir size100
```

192. 확장자가 .sh인 쉘 스크립트 중에서 사이즈가 100바이트 이상인 사이즈만 출력하시오!

답 : \$vi mv_size_100.sh

```
size2='ls -l *.sh | awk '{print $5}'` # .sh가 확장자인 파일의 사이즈 부분만 뽑아낸다  
for i in $size2 #사이즈들을 하나씩 루프문으로 가져온다.  
do # 루프문 시작  
    if [ ${i:0:3} -gt 100 ]; then # i가 문자형인데 숫자형으로 변경하려면  
        echo $i # ${i:0:3}로 하면 i가 숫자로 변경이 된다.  
        # 사이즈가 3자리가 안넘어가므로 3을 적어준다  
    fi  
done # 루프문 종료
```

193. 확장자가 .sh인 쉘 스크립트 중에 사이즈 100 바이트 이상인 쉘 스크립트는 size100 폴더에 이동시키시오!

답:

```
$vi mv_size_100.sh
```

```
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 # if문을 종료한다.  
done # 루프문을 종료한다.
```

설명 : 위에 \$size를 \${size:0:3}이라고 안해도 잘 되는 이유는 공백이 없기 때문입니다.

194. emp.txt에서 직업이 SALESMAN인 사원들의 data만 가지고 salesman.txt라는 파일을 생성하시오!

```
$ awk '$3==toupper("salesman") {print $0}' emp.txt >> salesman.txt
```

설명: 라이나 생명에서는 10대.txt, 20대.txt, 30대.txt, 40대.txt 이렇게 만들었듯이 우리도 emp.txt를 가지고 salesman.txt, analyst.txt, clerk.txt, manager.txt를 자동으로 다 만들어지게 하고 싶은 것입니다.

195. emp.txt에서 직업을 출력하는데 중복을 제거해서 출력하시오!

답: awk '{print\$3}' emp.txt|sort|uniq

196. 위의 직업을 하나씩 불러와서 for loop문에서 출력하는 코드를 작성하시오!

답: \$ vi for_job.sh

```
job=`awk '{print$3}' emp.txt|sort|uniq`  
  
for i in $job  
do  
    echo $i  
done
```

```
$ sh for_job.sh
```

197. 위의 스크립트를 이용해서 해당 직업의 사원들의 데이터만 직업을 이름으로 해서 아래와 같이 생성되게 하시오!

```
$ sh generate_job.sh  
$ ls  
salesman.txt presiden.txt manager.txt
```

답:

```
$ vi generate_job.sh  
  
job=`awk '{print$3}' emp.txt|sort|uniq`  
  
for i in $job  
do  
    grep -i $i emp.txt >> $i.txt  
  
done
```

198. 위의 스크립트를 이용해서 부서번호별 emp.txt가 쪼개지게끔 스크립트를 생성하시오!

```
$ sh make_deptno.sh  
  
$ ls -lrt  
  
deptno10.txt      deptno20.txt      deptno30.txt  
  
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  
done
```

199. (오늘의 마지막 문제 1/6) 리눅스 설치를 완료하고 아래 2가지를 설정해 놓으세요.

- 게스트 확장 설치
- 공유폴더에 접근할 수 있게 하는 방법
 - sudo usermod -a -G vboxsf scott 입력
 - sudo chown -R scott:users /media/sf_data/ 입력

200. (점심시간 문제) putty로 centos 리눅스 서버에 접속한 화면을 캡쳐해서 올립니다.



201. /home/scott 디렉토리로 이동해서 test100이라는 디렉토리를 생성하시오!

답 : \$ cd
\$ pwd
\$ mkdir test100

202. 리눅스명령어로 1부터 5까지숫자를 출력하는 쉘스크립트를 작성하고 아래와 같이 실행하시오!

```
$ vi a.sh
for i in {1..5}
do
    echo $i
done

$ sh a.sh
```

203. 공유폴더로 접근하시오!

```
$ cd /media/sf_data
```

204. 공유폴더에 있는 emp.txt를 /home/scott 밑으로 복사하시오!

답 : \$ cp emp.txt /home/scott/

205. 다시 집(/home/scott) 으로 와서 emp.txt를 cat으로 열어보시오!

답 : \$ cd
\$ pwd
\$ cat emp.txt

206. 공유폴더에 있는 dept.txt도 /home/scott 밑으로 복사하고 cat으로 열어보시오!

답 : cp /media/sf_data/emp.txt .

설명 : 끝에 한칸 띄고 . 넣기. 점(.) 이 의미하는 것은 현재 디렉토리 입니다.

207. 다음 3개의 파일을 공유폴더에서 복사하여 /home/scott 밑에 두시오.

1. jdk-7u60-linux-i586.gz
2. hadoop-1.2.1.tar.gz
3. protobuf-2.5.0.tar.gz

```
$ cd /media/sf_data
$ 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/
```

```
$ ls -rlt
```

리눅스 서버의 아이피주소를 확인하는 명령어인 ifconfig를 수행합니다.

208. (오늘의 마지막 문제 1/7) centos2를 다시 복제해서 centos2_3으로 만들고 처음부터 지금까지의 내용을 다시 수행하며 복습합니다.

마지막 로그인 부분을 캡쳐해서 올리세요~



209. hive에서 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 ;
```

210. 하둡 파일 시스템에 emp2.csv 를 올리시오 !

```
hive > exit;  
$ cd  
  
$ cp /media/sf_data/emp2.csv emp2.csv  
  
$ . .bash_profile  
  
$ hadoop fs -put emp2.csv emp2.csv  
  
$ hadoop fs -ls emp2.csv
```

(헷갈리는 용어 정리)

put

로컬의 파일을 HDFS에 복사할 때 사용합니다.

```
주요옵션: -f  
$ hadoop fs -put ./data1.txt /user/  
# 동일한 이름의 파일이 존재하면 덮어씀  
$ hadoop fs -put -f ./data1.txt /user/
```

211. (점심시간 문제) 하둡 파일 시스템의 emp2.csv를 emp에 로드하시오!

```
$ cd hive-0.12.0/bin  
$ ./hive
```

212. 월급이 3000인 사원의 이름과 월급을 출력하시오.

```
hive > select ename, sal  
> from emp  
> where sal =3000;
```

213. 직업이 SALESMAN인 사원들의 이름과 직업을 출력하시오!

```
hive> select ename, job  
> from emp  
> where job = 'SALESMAN';
```

214. 월급이 1200 이상이고 직업이 SALESMAN인 사원들의 이름과 월급과 직업을 출력하시오!

```
hive> select ename, sal, job  
> from emp  
> where sal >= 1200 and job = 'SALESMAN';
```

설명 : emp와 같은 작은 데이터를 검색하기 위해서 하둡을 사용하는 것이 아니라 대용량 데이터를 빠르게 검색하기 위해서 하둡을 사용하는 것 입니다.

emp.csv는 hive 보다는 maria db를 이용하는게 좋습니다.

몇억건씩 되는 큰 대용량 csv파일이나 텍스트 파일은 hive를 이용하면 빠르게 검색할 수 있습니다.

215. dept 테이블을 생성하시오!

- 순서 : 1. dept.csv 파일을 하둡 파일 시스템에 올립니다.
2. dept 테이블 생성 스크립트를 가지고 생성합니다.
3. 하둡 파일 시스템에 올라온 dept.csv 파일을 dept 테이블에 입력합니다.

```
[scott@centos bin]$ cd  
[scott@centos ~]$ cp /media/sf_data/dept2.csv /home/scott/  
[scott@centos ~]$ hadoop fs -put dept2.csv dept2.csv  
[scott@centos ~]$ hadoop fs -ls dept2.csv
```

```
hive >create table dept  
(deptno int, #컬럼은 문자면 string, 숫자면 int를 습니다.  
dname string,  
loc string)  
ROW FORMAT DELIMITED #반드시 써줘야하는 문법  
FIELDS TERMINATED BY ',' # 컬럼과 컬럼은 콤마로 구분하겠다.  
LINES TERMINATED BY '\n' # 행과 행은 엔터로 구분한다  
STORED AS TEXTFILE ; #반드시 써줘야하는 문법
```

```
hive> load data inpath '/user/scott/dept2.csv'  
      overwrite into table dept;
```

216. 이름과 부서위치를 출력하시오!

```
hive > select e.ename, d.loc  
> from emp e join dept d  
> on (e.deptno=d.deptno)
```

설명 : hive에서 조인하려면 오라클 조인문법 말고 1999 ansi 문법으로 조인해야합니다.

217. DALLAS에서 근무하는 사원들의 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';
```

218. 직업, 직업별 토탈월급을 출력하시오!

```
hive> select job, sum(sal)  
> from emp  
> group by job;
```

219. 부서번호, 부서번호별 평균월급을 출력하는데 부서번호별 평균월급이 높은 것 부터 출력하시오!

```
hive> select deptno, avg(sal)  
> from emp  
> group by deptno  
> order by avg(sal) desc;
```

이렇게 하면 에러가 남

```
hive> select deptno, avg(sal) as avgsal  
      from emp  
      group by deptno  
      order by avgsal desc;
```

또는

```
hive> select deptno, avg(sal)  
> from emp  
> group by deptno  
> order by 2 desc;
```

설명 : hive는 오라클과는 다르게 그룹함수를 써서 정렬을 할 때 order by 절에 컬럼별칭을 사용해야 합니다.

220. 위의 결과를 다시 출력하는데 소수점 이하는 안나오게 반올림하시오.

```
hive> select deptno, round(avg(sal))  
      from emp  
      group by deptno  
      order by 2 desc;
```

221. 이름과 입사한 년도(4자리)를 출력하시오!

KING 1981

BLAKE 1981

: :

답: hive > select ename, year(to_date(hiredate))
from emp;

222. 1981년도에 입사한 사원들의 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
2. 분산파일 시스템의 장점을 이용할 있어서이다.
(여러개의 컴퓨터를 하나로 묶어서 슈퍼컴퓨터를 만들 수 있다)
3. 큰 텍스트 파일을 테이블에 insert하지 않고 그냥 os 두고 링크만 걸어서 select 할 수 있다.
(오라클의 external table과 유사하다.)

223. movies.dat 파일의 위의 10줄만 열어서 보시오 (head 명령어)

\$ head 10 movies.dat

224. movies.dat 파일의 총 라인수가 어떻게 되는지 확인하시오.

답: \$ wc -l movies.dat

결과:

3883 movies.dat

225. ratings.dat 파일의 총 라인수가 어떻게 되는지 확인하시오.

답: \$ wc -l ratings.dat

결과:

1000209 ratings.dat

226. ratings.dat 파일의 위의 10줄만 출력하시오!

답 : \$ head 10 ratings.dat

설명 : ratings.dat의 컬럼 userId,movieId,rating,timestamp

movies.dat의 컬럼 movieId,title,genres

위의 컬럼에 대한 자세한소개는 README.txt를 읽어봅니다.

ratings는 영화를 본 사람들이 영화번호, 영화평점을 올려놓은 데이터

227. (오늘의 마지막 문제 1/8) ratings 테이블에서 영화번호, 영화번호별 영화평점의 평균값을 출력하는데 영화번호별 영화평점의 평균값이 높은것부터 출력하시오!

답 :

hive> select movieid, avg(rating) as r
from ratings
group by movieid
order by r desc;

228. 다시 emp와 dept 테이블을 생성하고 데이터를 로드하시오.

```
$ hadoop fs -put emp2.csv emp2.csv  
$ hadoop fs -put dept2.csv dept2.csv  
$ hive  
  
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 ;  
  
create table dept  
( deptno int,  
  dname string,  
  loc   string )  
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> load data inpath '/user/scott/dept2.csv'  
      > overwrite into table dept;
```

229. 이름, 월급, 월급에 대한 순위를 출력하시오!

```
hive> select ename, sal,  
       dense_rank() over(order by sal desc) rnk  
     from emp;
```

230. 이름, 월급, ntile 함수를 이용해서 월급의 등급을 출력하시오!

(월급의 등급을 4등급으로 분류하시오~)

0 ~ 25% : 1
26 ~ 50% : 2
51 ~ 75% : 3
76 ~ 100% : 4

답:

```
hive> select ename, ntile(4) over (order by sal desc) grade  
      from emp;
```

231. 아래의 SQL을 Hive로 구현하시오!

```
SQL > select deptno, listagg(ename, ',') within group (order by ename)  
      from emp
```

```
group by deptno;
```

답:

```
Hive> select deptno, CONCAT_WS(',', COLLECT_SET(ename))
      from emp
      group by deptno;
```

232. 사원번호, 이름, 월급, 월급의 누적치를 출력하시오!

```
Hive > select empno, ename, sal, sum(sal) over (order by empno asc) sumsal
      from emp;
```

233. 오라클의 데이터 분석함수의 lag와 lead 함수가 hive에서도 되는지 확인하시오!

```
SQL> select deptno, ename, sal,
      lag(sal, 1) over(order by empno asc) 그 전행,
      lead(sal,1) over (order by empno asc) 다음행
      from emp;
```

```
hive> select deptno, ename, sal,
      lag(sal,1,0) over (order by empno) lag,
      lead(sal,1,0) over (order by empno) lead
      from emp;
```

234. 오라클 데이터 분석함수의 lag와 lead 함수가 hive에서도 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;
```

235. 부서번호, 이름, 월급, 자기가 속한 부서번호의 평균월급을 출력하시오!

```
Hive> select deptno, ename, sal,
      avg(sal) over (partition by deptno) avgsal
      from emp;
```

설명 : hive에서 결과 출력 컬럼명 나오게 하는 방법

```
hive> set hive.cli.print.header=true;
```

236. 위의 결과를 다시 출력하는데 자기의 월급이 자기가 속한 부서번호의 평균월급보다 더 큰 사원들만 출력 하시오!

(힌트 : from 절의 서브쿼리를 사용해야 합니다)

```
hive>select *
      from ( select deptno, ename, sal,
                  avg(sal) over( partition by deptno) avgsal
              from emp ) aaa
      where sal > avgsal;
```

설명 : hive에서 from 절의 서브쿼리를 사용하려면 반드시 alias를 사용해야 합니다.

237. DW에서 많이 사용하는 SQL중 하나가 WITH 절입니다. 오라클의 WITH절이 Hive에서도 되는지 확인하세요~

설명 : with 절은 hive 1.2.1 버전은 지원 안되고 그 이후 버전부터 가능합니다.

답 : with emp77(select job, sum(sal) sumsal
 from emp
 group by job_
 select job, sumsal
 from emp77;

238. DW에서 많이 사용하는 집계값을 구하는 쿼리를 수행하시오!

부서번호, 부서번호별 토탈월급을 출력하는 맨 아래쪽에 전체 토탈월급이 출력되게 하시오!

SQL > select deptno, sum(sal)
 from emp
 group by rollup(deptno);

Hive> select deptno, sum(sal)
 from emp
 group by deptno with rollup;

결과 :

NULL 29025
10 8750
20 10875
30 9400

239. 문제 238번의 결과에서 전체 토탈월급이 아래쪽에 나오도록 order by절을 사용하세요!

select deptno, sum(sal) sumsal
 from emp
 group by deptno with rollup
 order by sumsal asc;

240. 캐글에서 코로나 데이터 분석에 필요한 코로나 데이터를 내려받아 리눅스 시스템에 넣으시오!

/home/scott 밑에 가져다 두세요!

241. PatientInfo.csv를 hive의 테이블로 구성하시오!

\$ head PatientInfo.csv
\$ vi PatientInfo.csv

맨위의 컬럼행을 dd로 지우고저장하고 나오세요!

\$ hadoop fs -put PatientInfo.csv PatientInfo.csv
\$ hive

하이브 들어가서
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,  
decreased_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 -rm /user/scott/PatientInfo.csv
```

**242. (점심시간 문제) 코로나 환 데이터를 하둡에 올리고 hive로 아래와같이 위에 5건만 조회하고 결과를 캡쳐
를 올리시오!**

```
hive >select * from patient limit 5;
```

답:

```
hive> load data inpath '/user/scott/PatientInfo.csv'  
      overwrite into table patient;  
hive >select * from patient limit 5;
```



243. 나이대(age), 나이대별 인원수를 출력하시오!

```
hive > select age, count(*)  
      from patient  
      group by age;
```

244. 위의 결과를 다시 출력하는데 인원수가 높은 것부터 출력하시오!

```
hive >select age, count(*) c  
      from patient  
      group by age  
      order by c desc;
```

245. 오라클의 grouping sets가 hive에서도 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 토탈입니다.

246. 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%

247. DW 쪽에서 가장 빈번히 수행하는 쿼리문이 아래의 SQL인데 이 아래의 SQL을 HIVE로 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;
```

답 :

```
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;
```

결과:

```
dept10  dept20  dept30
8750    10875   9400
```

248. DW 쪽에서 많이 사용하는 아래의 SQL에 Group by 까지 사용한 결과를 hive로 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;
```

답:

```
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 csv파일을 받았으면 csv 파일을 가지고 table 생성을 잘 할 수 있어야 합니다.

249. 코로나 데이터 중에 감염경로를 알려주는 Case.csv를 가지고 hive에 테이블을 생성하시오!

```
create table case1
(
case_id int,
province string,
city string,
group string,
```

```
infection_case string,  
confirmed string,  
latitude string,  
longitude string  
)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ''  
LINES TERMINATED BY '\n'  
STORED AS TEXTFILE ;
```

```
$ hadoop fs -put Case.csv Case.csv
```

```
$ hive
```

```
hive> load data inpath '/user/scott/Case.csv'  
      overwrite into table case1;
```

250. 감염경로(infection_case)를 중복 제거해서 출력하시오!

```
SQL> select distinct infection_case  
      from case1;
```

(hive에서 똑같이 하면 됨)

251. id가 1202인 사원의 이름과 나이를 출력하시오!

```
scala> sql("select * from employee where id=1202").show()
```

252. 이름이 satish인 사원의 이름과 나이를 출력하시오!

```
scala> sql("select * from employee where name='satish'").show()
```

253. emp 테이블을 스칼라에서 생성하시오!

```
scala> val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
```

```
scala> sqlContext.sql("DROP TABLE emp")
```

```
scala> sqlContext.sql("CREATE TABLE IF NOT EXISTS 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'")
```

254. 스칼라에서 생성한 emp테이블에 emp.csv를 로드하시오!

```
scala> sqlContext.sql("LOAD DATA LOCAL INPATH 'emp2.txt' INTO TABLE emp")  
scala> sql("select * from emp").show()
```

255. (오늘의 마지막 문제 1/11) 스칼라에서 dept 테이블을 생성하고 dept 테이블에 데이터를 입력하시오~ (NEWYORK을 붙여서 넣으세요~)

```
scala> :quit  
[scott@centos ~]$ vi dept2.csv  
[scott@centos ~]$ cp dept2.csv dept2.txt  
[scott@centos ~]$ spark-shell
```

```
scala> val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
```

```
scala> sql("DROP TABLE dept")
```

```
scala> sqlContext.sql( "CREATE TABLE IF NOT EXISTS dept(deptno int, dname string, loc string) ROW  
FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' ")  
  
scala> sqlContext.sql("LOAD DATA LOCAL INPATH 'dept2.txt' INTO TABLE dept")  
  
scala> sql("select * from dept").show()
```



256. dept2.csv를 복사해서 dept2.txt로 생성하고 스칼라에서 dept 테이블을 생성하시오!

```
[scott@centos ~]$ vi dept2.csv  
[scott@centos ~]$ cp dept2.csv dept2.txt  
[scott@centos ~]$ spark-shell  
  
scala> val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)  
  
scala> sql("DROP TABLE dept")  
  
scala> sqlContext.sql( "CREATE TABLE IF NOT EXISTS dept(deptno int, dname string, loc string) ROW  
FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' ")  
  
scala> sqlContext.sql("LOAD DATA LOCAL INPATH 'dept2.txt' INTO TABLE dept")  
  
scala> sql("select * from dept").show()
```

아래와 같이 나오면 준비 작업 끝

```
scala> sql("select * from emp").count()  
>>> 14건  
scala> sql("select * from dept").count()  
>>> 4건
```

257. 스파크에서 부서번호가 30번인 사원들의 모든 데이터를 출력하시오!

```
scala> sql("select * from emp where deptno=30").show()
```

258. 위에서 출력된 결과의 건수가 어떻게 되는지 확인하시오!

```
scala> sql("select * from emp where deptno=30").count()
```

259. 직업이 SALESMAN인 사원들의 이름과 직업과 월급을 출력하시오!

```
sql("select ename, job, sal from emp where job='SALESMAN'").show()
```

260. 부서 테이블 전체를 출력하시오!

```
scala> sql("select * from dept").show()
```

261. (조인이 가능한지 확인) 이름과 부서위치를 확인하시오! 힌트 : 1999 ANSI 조인 문법으로 수행해야함

```
scala> sql("select e.ename, d.loc from emp e join dept d where e.deptno=d.deptno").show()
```

262. DALLAS에서 근무하는 사원들의 이름과 부서위치를 출력하시오!

```
scala> sql("select e.ename, d.loc from emp e join dept d where e.deptno=d.deptno and d.loc='DALLAS'").show()
```

설명 : 준혁씨가 알려준 여러줄 라인 작성 방법은 더블 쿼테이션 마크를 3개를 둘러주면 됩니다.

263. 애우터 조인도 가능한지 확인하세요. 아래의 SQL을 scala에서 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()
```

264. (서브쿼리가 가능한지 확인) JONES보다 더 많은 월급을 받는 사원들의 이름과 월급을 출력하시오!

```
scala>sql ("""  
select ename, sal  
      from emp  
    where sal > (select sal  
                  from emp  
                where ename='JONES')""").show()
```

265. (서브쿼리가 가능한지 확인) ALLEN보다 늦게 입사한 사원들의 이름과 입사일을 출력하시오!

```
scala>sql ("""  
select ename, hiredate  
      from emp  
    where hiredate > ( select hiredate  
                      from emp  
                    where ename='ALLEN')""").show()
```

266. (분석함수가 지원되는지 확인) 이름, 월급, 월급에 대한 순위를 출력하시오.

```
scala>sql ("""  
select ename, sal, dense_rank() over (order by sal desc)  
      from emp""").show()
```

267. 직업, 직업별 토탈월급을 출력하시오!

```
scala>sql("""select job, sum(sal)  
          from emp  
        group by job""").show()
```

268. (스칼라의 장점중 하나가 결과 값 중에 숫자값에 대한 통계정보를 확인할 수 있습니다.) 위의 결과중에 숫자로 나오는 값에 대한 통곗값을 출력하시오.

```
scala>sql("""select job, sum(sal)  
          from emp  
        group by job""").describe().show()
```

설명 : 카페에 가면 스칼라 메뉴얼 사이트가 있는데 거기서 확인합니다. 게시글385 스파크 명령어 매뉴얼

결과:

```
+-----+-----+
|summary|      sum(sal)|
+-----+-----+
| count|      5|
| mean|    5805.0|
| stddev|1546.6091943344964|
| min|    4150|
| max|    8275|
+-----+-----+
```

269. 사원 테이블을 출력하는데 맨 위에 한 행만 출력하시오!

(힌트: head(), first() : 데이터셋의 첫번째 ROW만 반환)

```
scala>sql("""select *
           from emp""").head()
```

결과: Row = [7839,KING,PRESIDENT,0,1981-11-17,5000,0,10]

270. 위에서 출력한 직업과 직업별 토탈월급의 결과를 csv파일로 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") : 컬럼명이 나오게 해라~

option("sep",",") : csv 파일 형태로 만들어라~

mode("overwrite").csv("/home/scott/dd") : /home/scott/dd라는 폴더안에 생성해라~

```
[scott@centos ~]$ cd dd
[scott@centos dd]$ cat part-r-00000-1689116d-3942-47a6-8958-5fd1280934eb.csv ### p 누르고 탭하기
```

```
job,sum(sal)
ANALYST,6000
SALESMAN,5600
CLERK,4150
MANAGER,8275
PRESIDENT,5000
```

```
[scott@centos dd]$ mv part-r-00000-1689116d-3942-47a6-8958-5fd1280934eb.csv job.csv
[scott@centos dd]$ ls
job.csv _SUCCESS
```

271. jobs.csv의 내용을 막대그래프로 시각화 하시오!

```
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')
```

272. (점심시간 문제) 스칼라를 이용해서 데이터 전처리 및 검색을 하고 파이썬에서 데이터 시각화하는 큰 그림을 구현하시오!
- 부서번호, 부서번호별 토탈월급을 막대 그래프로 그리시오.
(색깔은 다른색깔로 정하고 캡쳐해서 올리세요~)

1. 스칼라에서는 부서번호, 부서번호별 토탈월급을 csv 파일로 뽑는다.
2. 그 csv 파일을 윈도우의 파이썬에서 막대 그래프로 시각화하세요~

```
scala> sql("""select deptno, sum(sal)
   from emp
  group by deptno
 order by deptno
asc""").coalesce(1).write.option("header","true").option("sep",",").mode("overwrite").csv("/home/scott/dd")
```

scott 창에서

```
[scott@centos ~]$ cd dd
[scott@centos dd]$ mv part-r-00000-2798198e-0e64-400c-b5d7-974bdbac4f47.csv deptno.csv
[scott@centos dd]$ cat deptno.csv
deptno,sum(sal)
20,10875
10,8750
30,9400
```



메모장에 복사 붙여넣기, deptno.csv로 저장

```
<Spyder>
import pandas as pd

emp= pd.read_csv("C:\data\deptno.csv")
result= emp['sum(sal)']
result.index=emp['deptno']
result.plot(kind='bar', color='orange')
```

273. mobaxterm을 이용해서 sample.csv를 /home/scott 밑에 올리시오

↑ 표시 있는것을 클릭하면 파일이 올라감.

설명 : 현업에서는 오라클 vm으로 가상환경을 만드는게 아니라 실제 리눅스 서버를 사용하므로 내자리의 노트북에 있는 파일을 별도의 컴퓨터로 구성되어 있는 리눅스 서버에 올릴려면 mobaxterm과 같은 툴을 이용해서 올립니다.

리눅스 서버의 대용량 데이터를 내 노트북으로 내릴때도 mobaxterm을 이용해서 편하게 합니다.

274. emp.2.csv를 리눅스의 spyder에서 판다스로 불러오고 프린트하시오!

```
import pandas as pd  
emp=pd.read_csv("/home/scott/emp2.csv")  
print(emp)
```

275. emp2.csv를 emp로 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='blue')
```

276. (오늘의 마지막 문제 1/12) 리눅스 하둡 시험문제 제출 포멧

현업에서 데이터 분석가들이 분석 요청이 들어오면 하는 일

Case.csv를 스칼라로 로드해서 테이블을 생성한 후에 지역, 지역별 코로나 감염자 수를 카운트 하고 결과를 가지고 파이썬에서 막대그래프로 시각화 하시오!

scott에서 시작

```
(py389) [scott@centos ~]$ cp Case.csv Case.txt  
(py389) [scott@centos ~]$ spark-shell # 스칼라로 들어옴
```

```
scala> val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
```

```
scala> sql("drop table case1") # 원활한 진행을 위해 테이블 드랍 먼저 시행
```

```
scala> sqlContext.sql( "CREATE TABLE IF NOT EXISTS case1(case_id int, province string, city string, group string, infection_case string, confirmed string, latitude string, longitude string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' ")
```

```
scala> sqlContext.sql("LOAD DATA LOCAL INPATH 'Case.txt' INTO TABLE case1")
```

```
scala> sql("""select province, sum(confirmed)  
      from case1  
      group by  
      province""").coalesce(1).write.option("header", "true").option("sep", ",").mode("overwrite").csv("/home/scott/dd")
```

scala > :q #스칼라 빠져나옴

```
(py389) [scott@centos ~]$ cd dd #dd 디렉토리로 들어감
```

```
(py389) [scott@centos dd]$ ls  
part-r-00000-4bd2fd7b-f1f8-443a-b38f-bc1bf914ffe5.csv _SUCCESS
```

```
(py389) [scott@centos dd]$ mv part-r-00000-4bd2fd7b-f1f8-443a-b38f-bc1bf914ffe5.csv case1.csv #파일 이름을 case1.csv로 변경
```

<spyder 창에서>

```
import pandas as pd
```

```
case = pd.read_csv("/home/scott/dd/case1.csv")

result=case['sum(CAST(confirmed AS DOUBLE))']
result.index=case['province']
result.plot(kind='bar', color='green')
```



×

277. 하둡 파일 시스템의 /(루트) 밑에 있는 모든 하위 디렉토리를 조회하시오!

답 : \$ hadoop fs -lsr /

278. 하둡 파일 시스템의 리눅스 /home/scott 밑에 있는 Case.csv를 /user/scott 밑에 case3.csv로 올리시오!

답 : \$ hadoop fs -put /home/scott/Case.csv /user/scott/case3.csv

279. 하둡 파일 시스템에 올린 case3.csv의 내용을 조회하시오!

답 : \$ hadoop fs -cat /user/scott/case3.csv
\$ hadoop fs -text /user/scott/case3.csv

280. 다음에서 만든 test 디렉토리가 잘 생성되었는지 확인하시오!

\$ hadoop fs -mkdir /user/scott/test

답 : \$ hadoop fs -ls

281. 앞으로는 test 디렉토리에 데이터를 올리고 관리를 하기 위해서 리눅스에 /home/scott 밑에 있는 emp2.csv를 /user/scott/test에 emp2.csv로 올리시오~

답 : \$ hadoop fs -put /home/scott/emp2.csv /user/scott/test/emp2.csv

281. 리눅스에 /home/scott 밑에 있는 emp2.csv를 하둡에 /user/scott 밑에 emp3.csv로 올리시오~

답 : \$ hadoop fs -put /home/scott/emp2.csv /user/scott/emp3.csv

282. 리눅스에 /home/scott 밑에 있는 emp2.csv 를 하둡에 /user/scott 밑에 emp3.csv 로 올리시오 ~

답: \$ hadoop fs -put /home/scott/emp2.csv /user/scott/emp3.csv

283. 코로나 데이터 중 감염자 정보중에 성별 데이터가 있는 csv파일을 하둡 파일 시스템에 /user/scott 밑에

올리시오~

답 : \$ hadoop fs -put /home/scott/TimeGender.csv /user/scott/TimeGender.csv

284. 하둡 파일 시스템에 올린 코로나 데이터인 timegender.csv 파일을 cat으로 여시오.

답 : \$ hadoop fs -cat /user/scott/TimeGender.csv

285. 하둡 파일 시스템에 올린 코로나 데이터인 TimeGender.csv 파일에서 남자가 모두 몇명인지 확인하시오!

답 : \$ hadoop fs -cat /user/scott/TimeGender.csv | grep -i 'male' | wc -l

결과: 242

286. 판다스를 이용해서 원형 그래프를 그리시오!

```
import pandas as pd

emp= pd.read_csv("/home/scott/emp2.csv", header=None)

result=emp[2].value_counts()
print(result)

result.plot(kind='pie')
```

287. (점심시간문제) 판다스로 직업, 직업별 토탈월급을 출력하시오! emp2.csv에 컬럼명을 맨 위에 vi 편집기로 넣고 하세요~

```
import pandas as pd
emp = pd.read_csv("/home/scott/emp2.csv")
result = emp.groupby('job')['sal'].sum()
print( result )
```

288. 월급이 1000에서 3000사이인 사원들의 이름과 월급을 출력하시오!

```
orcl > select ename, sal
  >   from emp
  > where sal between 1000 and 3000;
```

289. (조인 가능한지 확인) DALLAS에서 근무하는사원들의 이름과 부서위치를 출력하시오!

<1> 1999 ANSI join
MariaDB [orcl]>
select e.ename, d.loc
 from emp e join dept d
 on (e.deptno=d.deptno)
 where d.loc='DALLAS';
... [결과값 출력 잘 된다]

<2> oracle join
select e.ename, d.loc
 from emp e, dept d
 where e.deptno=d.deptno;
err [출력x]

290. (서브쿼리 가능한지 확인) JONES보다 더 많은 월급을 받는 사원들의 이름과 월급을 출력하시오!

```
select ename, sal
from emp
where sal > (select sal
               from emp
```

```
where ename='JONES')
```

291. hive는 데이터 수정이 안되지만 maria db는 가능합니다. 이름이 SCOTT인 사원의 월급을 9000으로 변경 하시오!

```
<1>update  
update emp  
set sal = 9000  
where ename='SCOTT';
```

```
<2>check res  
select ename, sal from emp where ename='SCOTT';
```

```
<3>auto commit  
rollback;  
select ename, sal from emp where ename='SCOTT'; # 데이터가 그대로임
```

설명 : 마리아 db에서 autocommit 설정확인하는 방법

```
SELECT @@AUTOCOMMIT;  
SET AUTOCOMMIT = 0; 해줌!!
```

292. (오늘의 마지막 문제 1/13) mariadb에서 emp_partition3 라는 이름으로 파티션 테이블을 생성하는데 파티션 키 컬럼을 job으로 해서 리스트 파티션을 생성하시오
생성하고 데이터도 입력해 놓으세요~

```
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 ('PRESIDENT'),  
      partition p2 values in ('MANAGER'),  
      partition p3 values in ('SALESMAN'),  
      partition p4 values in ('CLERK'),  
      partition p5 values in ('ANALYST'));
```

```
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;
```



293. 마아 디비와 파이썬과 연동된 상태에서 spyder에서 이름과 월급을 출력하시오!

앞에 코드들

```
import pandas as pd
emp=pd.DataFrame(resultList)
emp.columns=['empno', 'ename', 'job', 'mgr', 'hiredate', 'sal', 'comm', 'deptno']
print(emp)
```

만약 pandas 모듈이 없다고 나오면 아래와 같이 py389 가상환경을 activate 한 putty 터미널 창에서 판다스 모듈을 install 합니다.

```
$ conda install pandas
```

294. 월급이 3000이상인 사원들의 이름과 월급을 출력하시오!

판다스 문법으로 구현

앞의 코드들 ...

```
print( emp[ [ 'ename', 'sal']] [emp['sal'] >= 3000 ])
```

295. 직업과 직업별 토탈 월급을 출력하시오! (SQL로 하세요~)

sql 부분만 고치기

```
sql = """SELECT job, sum(sal)
      from emp
      group by job"""


```

```
import pandas as pd
emp=pd.DataFrame(resultList)
emp.columns=['job', 'sumsal']
print(emp)
```

296. 직업, 직업별 토탈 월급을 출력하는데 직업이 SALESMAN인 사원을 제외하고 출력하고 직업별 토탈 월급이 4000이상인 것만 출력하고 직업별 토탈월급이 높은것부터 출력되게 하시오!

앞의 코드들...

```
sql= """SELECT job, sum(sal)
      from emp
      where job !='SALESMAN'"""


```

```
group by job  
having sum(sal) >= 4000  
order by 2 desc"""
```

297. 부서위치, 부서위치별 토탈월급을 출력하시오!

```
sql = """Select d.loc, sum(e.sal)  
      from emp e join dept d  
      on (e.deptno=d.deptno)  
      group by d.loc """
```

298. 위의 결과를 막대 그래프로 시각화 하시오!

```
import mysql.connector
```

```
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 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['sumsal'])
```

```
result = emp['sumsal'].astype(int)  
result.index= emp['loc']
```

299. 현업에서 많이 사용하는 마리아 db와 mySQL을 편하게 사용할 수 있도록 하는 오라클의 sql developer와 같은 툴을 설치하고 리눅스 서버의 바리아 디비의 emp 테이블을 조회 하시오!

```
use orcl;  
select * from emp;
```

300. (점심시간 문제) workbench에서 아래의 문제의 결과를 출력하시오!

직업, 직업별 평균 월급을 출력하는데 직업별 평균월급이 2000 이상인 것만 출력하고 직업별 평균 월급이 높은 것부터 출력하시오!



301. 도시명, 도시명별 토탈 감염자수를 출력하시오!

```
select city, sum(confirmed)  
from cov_case  
group by city;
```

결과:

3625	
Anyangsi	39
Bonghwagun	68
Bucheonsi	67
Changnyeonggun	7
Changwonsi	7
Cheonansi	103
Cheongdogun	119
Chilgokgun	36
Dalseonggun	297
Dobonggu	43
Dongdaemungu	17
Donggu	44
Dongnaegu	39
Eunpyeonggu	14
from other city	1217
Gangnamgu	18
Gangseogu	0
Geochanggun	18
Geumcheongu	6
Goesangun	11
Gumisi10	
Gurogu	139
Gwanakgu	149

Gyeongsansi 99
Haeundaegu 6
Jingu 4
Jinjusi 9
Jongnogu 17
Junggu 23
Muangun 2
Nangu 4511
Sejong39
Seochogu 5
Seodaemungu 5
Seogu 151
Seongdonggu 13
Seongnamsi 94
Seosansi 9
Suwonsi 25
Suyeonggu 5
Uijeongbusi 50
Wonjusi 4
Yangcheongu 46
Yangsansi 3
Yechungun 40
Yeongdeungpogu 3
Yongsangu 139

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;
```

이렇게 해도 맨 위의 결과가 나오므로

```
select city, sum(confirmed)
  from cov_case
 where trim(city) is not null and city != ''
 group by city;
```

설명 : trim을 사용하면 양쪽 공백을 잘라 버리겠다. 라는 뜻

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 != ''
 group by city
 order by 2 desc;
```

304. 위의 결과를 다시 출력하는데 토탈 확진자수가 100명 이상인것만 출력하시오!

```
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;
```

305. 위의 결과를 막대 그래프로 시각화하시오!

전체 코드:

```
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
"""

# sql을 실행해서 cursor(메모리)에 담는다
cursor.execute(sql)

# cursor(메모리)에 담긴 결과 셋 얻어오기
resultList = cursor.fetchall() # tuple 이 들어있는 list

df = pd.DataFrame(resultList) #판다스 데이터 프레임으로 변환
df.columns = ['city', 'cnt'] #컬럼명을 만들어 줍니다.
print(df[['city','cnt']])

# 시각화
result = df['cnt']
result.index = df['city']
result.plot(kind='bar', color='red')

설명 : os의 csv 파일을 maria db에 넣고 SQL로 데이터 전처리를 한 후 파이썬으로 시각화 함
```

306. province를 중복 제거해서 출력하시오!

```
select distinct province
from cov_case;
```

307. 지역(province), 지역별 토탈 확진자수를 막대 그래프로 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 distinct province, sum(confirmed)
            from cov_case
            group by province
"""

# sql
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')

```

308. 코로나는 남자와 여자중에 누가더 많이 감염 되었는가?

큰 질문: 코로나는 남자와 여자 중에 누가 더 많이 감염되었는가?

1. 큰 질문 : 코로나는 남자와 여자중에 누가 더 많이 감염되었는가?

2. csv 데이터를 구한다.

\$ head PatientInfo.csv

3. 마리아 디비에 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,confir
med_date,released_date,deceased_date,state );

```

성별, 성별별 인원수를 출력하시오 !

```

MariaDB [orcl]> select case when sex = ' ' then 'no reponse'
      else sex end gender, count(*)
      from pati
      group by case when sex = ' ' then 'no reponse' else sex end;
+-----+-----+
| gender | count(*) |
+-----+-----+
| female |    2218 |
| male   |    1824 |
| no reponse |    1122 |
+-----+-----+

```

4. 파이썬에서 시각화를 한다.

```

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 case when sex = ' ' then 'no reponse'
           else sex end gender, count(*)
           from pati
           group by case when sex = ' ' then 'no reponse' else sex end
"""

# 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 = ['gender', 'cnt']

```

```
print(df[['gender','cnt']])  
  
# 시각화  
result = df['cnt']  
result.index = df['gender']  
result.plot(kind='bar', color='green')
```

5. 시각화 그래프와 함께 짧게 시각화한 결과를 설명

with graph

설명: 확진자수는 남자는 몇명이고 여자이는 몇명이고 무응답은 몇명이었습니다 그램프를 보면 남자보다는 여자가 더 많이 확진되었음을 확인할 수 있었습니다.

6. 아래의 데이터 프레임의 데이터를 편하게 볼 수 있는 View함수를 가지고 v라는 함수를 만들고 실행하시오!

```
v <- function() {View(emp)}
```

```
v( )
```

7. v 함수를 my_fun2.R에 6번째로 추가 시키시오.