

하둡 목차

2018년 7월 11일 수요일 오전 10:46

- [1. 하둡 살펴보기 \(하둡의 정의\)](#)
- [2. 하둡 개발준비 \(하둡설치\)](#)
- [3. Hive 설치](#)
- [4. 하둡 분산 파일 시스템 명령어](#)
- [5. Hive로 정렬 \(17장\)](#)
- [6. Hive로 조인 \(17장\)](#)
 - [하이프\(hive\)에서 지원하는 내장함수](#)
 - [가로로 출력하는 방법 \(오라클에서 sum\(decode\) \)](#)
- [7. 타조](#)
- [8. Pig](#)
- [9. 스쿱으로 오라클과 연동](#)
- [10. Mongo DB 설치](#)
 - [mogodb에 테이블 생성](#)
 - [Mogodb연산자](#)
 - [오라클의 기타 비교 연산자와 mogodb의 비교](#)
 - [Mongodb에서의 조인문장](#)
 - [Mongodb 에서의 DML문](#)
 - [MongoDB에서 DDL 문](#)
- [11. 하둡과 R 연동하는 방법](#)
- [12. 자바를 이용해서 하둡 파일 시스템의 데이터를 select 하기](#)
- [13. 두개의 파일을 하나로 합쳐서 하둡 파일 시스템에 올리는 실습](#)
- [14. WordCount 예제](#)
- [15. MySQL에서 DML문\(자동 커밋 되서 조심해야함!!\)](#)
 - [MySQL에서 DDL문](#)

- 1. 하둡 살펴보기 (하둡의 정의)
- 2. 하둡 개발준비 (하둡설치)
- 3. Hive 설치
- 4. 하둡 분산 파일 시스템 명령어
- 5. Hive로 정렬 (17장)
- 6. Hive로 조인 (17장)
- 7. 타조
- 8. Pig
- 9. 스쿱으로 오라클과 연동
- 10. Mongo DB 설치
- 11. 하둡과 R 연동하는 방법
- 12. 자바를 이용해서 하둡 파일 시스템의 데이터를 select 하기
- 13. 두개의 파일을 하나로 합쳐서 하둡 파일 시스템에 올리는 실습
- 14. WordCount 예제
- 15. MySQL에서 DML문(자동 커밋 되서 조심해야함!!)
 - MySQL에서 DDL문

- 하둡 배우는 순서

하둡 설치 --> NoSQL --> 자바 --> 하둡설치(멀티노드)

NoSQL : (Hive, Tajo, Pig, Mongo DB)

1. 하둡 살펴보기 (하둡의 정의)

- 대용량 데이터를 분산 처리할 수 있는 자바기반의 오픈소스 프레임워크

하둡이 나온 배경지식 : 구글에서 구글에 쌓여지는 수많은 빅데이터들을 구글에서도 처음에는 RDBMS(오라클)에 입력하고 데이터를 저장하고 처리하려고 시도 했으나, 너무 데이터가 많아서 실패를 하고 자체적으로 빅데이터를 저장할 기술을 개발을 했다.
그리고 대외적으로 이 기술에 대한 논문을 하나 발표했다.

그 논문을 더그커팅(하둡을 만든이)이 읽고 자바로 구현했다.

그 이름을 고민하다가 더그커팅의 얘기가 노란 코끼리 장난감을 가지고 놀면서 Hadoop! 이라고 한걸 듣고 hadoop 이라고 이름을 지었다.

그래서 그 뒤로 hadoop을 편하게 이용할 수 있도록 개발한 모든 하둡 생태계에 개발 프로그램 이름들이 다 동물 이름으로 지어지게 되었다.

Hadoop ----- Hive(벌레), Pig(돼지), 타조

↓
자바를 알아야 한다.

↓
NoSQL : SQL과 비슷한 언어로 하둡의 데이터를 분석
(Not only SQL)

- 하둡의 장점 : 무료로 사용 가능.

복제가 가능하다.

분산 처리가 가능하다.

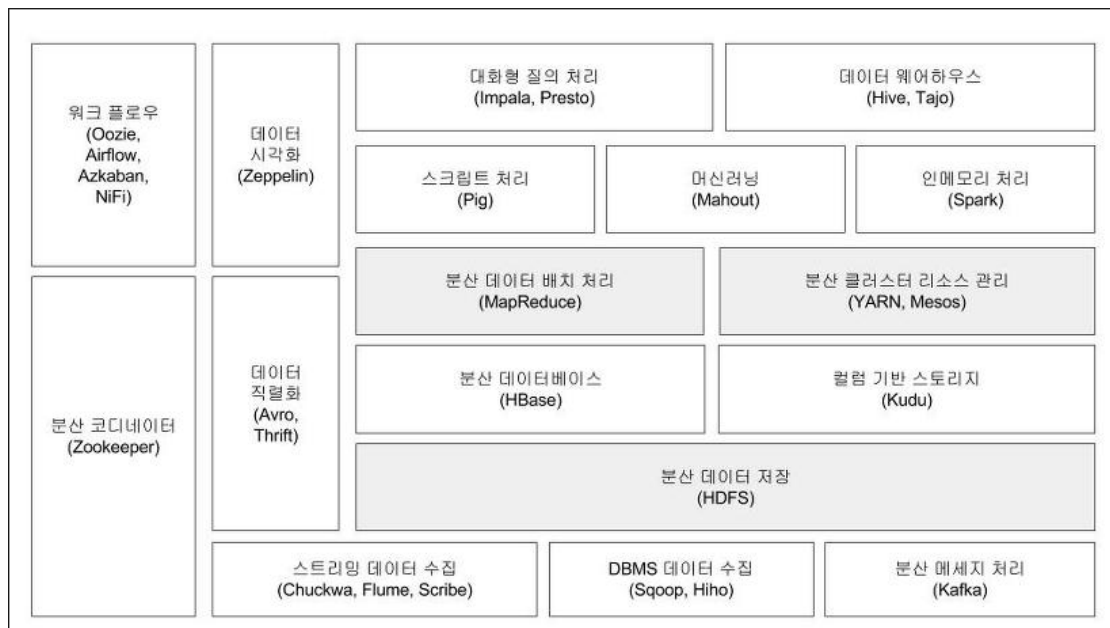
=> 여러대의 노드를 묶어서 마치 하나의 서버처럼 보이게 하고 여러 노드의 자원을 이용해서 데이터를 처리하기 때문에 처리하는 속도가 빠른 장점이 있다.

ex) 한대의 서버로 1테라 바이트의 데이터를 처리하는데 걸리는 시간이 2시간 반이 걸린다고 하면 하둡으로 여러대의 서버로 병렬로 작업한다면, 2분내로 데이터를 읽을 수 있다.

2008년 뉴욕 타임즈의 130년 분량의 신문기사 1100만 페이지를 하둡을 이용해서 하루만에 pdf로 변환을 했고, 비용이 200만원 밖에 안들었다.

만약 일반서버로 처리했다면 14년이 걸렸을 것으로 예상된다.

• 하둡 에코 시스템



빅데이터 분석



R, Python 등을 이용해서 분석



빅데이터 저장	Hbase	MongoDB	Cassandra	CouchDB
↑		↑		
분산처리 지원	Hive	Pig	Sqoop	Zookeeper
↑		↑		
분산 배치 처리	하둡(Hadoop) - MapReduce			
분산 파일 관리	하둡(Hadoop) - HDFS			

ex) O생명사의 예를 들면 보험상품에 대해서 관심이 있는 고객에 대해서 분석을 시도.

관심이 있는 고객을 선별을 하기 위해서 하둡에서 데이터를 필터링을 한다.

(40대만 추출)

40대 고객들을 K-means로 군집분석을 한다.

보험 상품에 **관심있는 고객**과 보험상품에 **관심없는 고객**을 군집화 한다.

목표 : 관심있는 고객에 대해서만 집중관리하자!!

차후과제 : 고객의 통화내용이 녹는 되는데, 음성(voice)을 텍스트로 해서 그 텍스트를 단어로 정리해서 matrix로 만들어서 군집분석을 하는 과제.
(meta-data를 관리 데이터의 위치 정보)

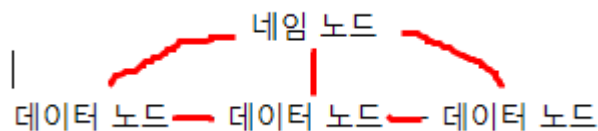
• 하이브 사용 예

```
hive> select (*) from users;
```

6040

```
hive> select * from omvies limit 5; 5개만 보여달라(head)
```

위와 같이 select를 하면 네임노드에서 movies라는 데이터가 어느 데이터 노드에 있는지 물어본다.



야후의 경우는 약 5만대를 연결해서 하둡을 사용하고, 페이스북은 약 1만대 이상의 하둡 클러스터를 이용하고 있다.

아마존 서버에 하둡 클러스터를 구성해보면 가장 하둡 설치와 관리에 대해서 아주 많이 배울수 있다.

하둡의 장점 : 저렴한 구축 비용과 비용대비 빠른 데이터 처리

하둡의 단점 : 1. 무료이다 보니 유지보수가 어렵다.

2. 네임노드가 다운되면 고가용성이 지원 안된다.

3. 한번 저장된 파일은 수정할 수 없다.

(기존 데이터에 append는 가능한데 수정은 안됨)

• 주요 하둡 배포판

리눅스도 centos, redhat, ubuntu 처럼 배포판이 있듯이 하둡도 배포판이 있다.

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

- 하둡 홈페이지

<http://hadoop.apache.org/>

2. 하둡 개발준비 (하둡설치)

1. java 설치 (하둡이 자바로 개발되었고, 데몬을 구동할때에 jar 파일을 수정하기 때문에 반드시 자바가 필요하다. jdk 1.6버전 이상 설치 권장)

```
[orcl:~]$ java -version
java version "1.6.0_18"
Java(TM) SE Runtime Environment (build 1.6.0_18-b07)
Java HotSpot(TM) Server VM (build 16.0-b13, mixed mode)
```

첨부파일 : 실습_00.Hadoop install.txt

```
#####
```

```
#      JDK 1.7 설치      #
```

```
#####
```

```
[oracle@edydr1p2 ~]$ java -version
```

```
java version "1.6.0"
```

```
OpenJDK Runtime Environment (build 1.6.0-b09)
```

```
OpenJDK Server VM (build 1.6.0-b09, mixed mode)
```

```
[oracle@edydr1p2 ~]$ su -
```

```
Password: oracle
```

```
[root@edydr1p2 ~]# mkdir -p /u01/app/java
```

```
[root@edydr1p2 ~]# mv /home/oracle/jdk-7u60-linux-i586.gz /u01/app/java/
```

```
[root@edydr1p2 ~]# cd /u01/app/java/
```

```
[root@edydr1p2 java]# tar xvfz jdk-7u60-linux-i586.gz
```

```
...
```

```
jdk1.7.0_60/bin/jrunscript
```

```
jdk1.7.0_60/release
```

```
jdk1.7.0_60/THIRDPARTYLICENSEREADME-JAVAFX.txt
```

```
jdk1.7.0_60/COPYRIGHT
```

```
[root@edydr1p2 java]# chown -R root:root jdk1.7.0_60
[root@edydr1p2 java]# exit
```

2. java 환경설정

```
[oracle@edydr1p2 ~]$ vi ~/.bash_profile
```

#export JAVA_HOME=/usr/java/jdk1.6.0_18 <--을 주석처리하고 아래 3줄을 붙여넣기.

```
=====
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
=====
```

```
[oracle@edydr1p2 ~]$ . .bash_profile
```

```
[oracle@edydr1p2 ~]$ java -version
```

```
java version "1.7.0_60"
```

```
Java(TM) SE Runtime Environment (build 1.7.0_60-b19)
```

```
Java HotSpot(TM) Server VM (build 24.60-b09, mixed mode)
```

```
#####
#           Hadoop 설치           #
#####
```

3. keygen 설정

하둡은 SSH 프로토콜을 이용해 하둡 클러스터간의 내부 통신을 수행한다.

하둡을 다 설치하고 나서 하둡을 시작하는 명령어인 start-all.sh 쉘 스크립트를 수행하면 네임노드가 설치된 서버에서 데이터 노드가 설치된 서버로 접근해 데이터 노드와 테스트 트래커를 구동하게 된다. 그런데 이때 ssh를 이용할 수 없다면 하둡을 실행할 수 없다.

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

이 키를 가지고 있으면 ssh로 다른 노드에 접속할 때 패스워드 없이 접속할 수 있다.

```
[oracle@edydr1p2 ~]$ cd
```

```
[oracle@edydr1p2 ~]$ rm -rf .ssh
```

```
[oracle@edydr1p2 ~]$ ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/home/oracle/.ssh/id_rsa):
```

```
Created directory '/home/oracle/.ssh'.
```

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/oracle/.ssh/id_rsa.

Your public key has been saved in /home/oracle/.ssh/id_rsa.pub.

The key fingerprint is:

06:52:a4:53:9e:fd:1b:b8:e4:42:2b:a5:34:cb:a4:17 oracle@edydr1p2.us.oracle.com

```
[oracle@edydr1p2 ~]$ cat /home/oracle/.ssh/id_rsa.pub >> /home/oracle/.ssh/authorized_keys
```

```
[oracle@edydr1p2 ~]$ ssh edydr1p0.us.oracle.com
```

...

Are you sure you want to continue connecting (yes/no)? yes

아래의 3개의 접속 명령어를 수행했을때 패스워드 물어보면 안되고 바로 접속되는지 확인

1. \$ ssh edydr1p0.us.oracle.com

2. \$ ssh edydr1p0

3. \$ ssh localhost

```
[orcl2:~]$ hostname
```

```
edydr1p0.us.oracle.com
```

```
[oracle@edydr1p2 ~]$ ssh edydr1p0.us.oracle.com
```

...

Are you sure you want to continue connecting (yes/no)? yes

```
[oracle@edydr1p2 ~]$ ssh localhost
```

...

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'localhost' (RSA) to the list of known hosts.

Address 127.0.0.1 maps to edydr1p2.us.oracle.com, but this does not map back to the address - POSSIBLE BREAK-IN

ATTEMPT!

Last login: Tue Aug 5 23:57:15 2014 from edydr1p2.us.oracle.com

4. 하둡 설치 파일 다운받아 압축을 해제.

```
[oracle@edydr1p2 ~]$ mkdir -p /u01/app/hadoop
```

```
[oracle@edydr1p2 ~]$ cd /u01/app/hadoop/
```

만든 디렉토리에 압출파일(hadoop-1.2.1.tar.gz) 을 올린다.

```
[orcl:hadoop]$ ls
hadoop-1.2.1.tar.gz
```

```
[orcl:hadoop]$ ls -l hadoop-1.2.1.tar.gz
-rw-r--r-- 1 oracle oinstall 63851630 Jul  3 13:56 hadoop-1.2.1.tar.gz
```

5:20

```
[oracle@edydr1p2 hadoop]$ wget http://mirror.apache-kr.org/hadoop/common/hadoop-1.2.1/hadoop-1.2.1.tar.gz
```

```
[oracle@edydr1p2 hadoop]$ tar xvfz hadoop-1.2.1.tar.gz
```

...

```
[oracle@edydr1p2 hadoop]$ rm hadoop-1.2.1.tar.gz #압축파일 날리기
```

5. 하둡 홈 디렉토리를 설정한다.

```
[oracle@edydr1p2 hadoop]$ cd
```

```
[oracle@edydr1p2 ~]$ vi .bash_profile
```

아래 2줄 내용을 export맨 아래쪽에 붙여넣는다

```
=====
export HADOOP_HOME=/u01/app/hadoop/hadoop-1.2.1
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
=====
```

```
[oracle@edydr1p2 ~]$ . .bash_profile
```

6. 하둡을 운영하기 위한 아래의 4개 파일을 수정한다.

1. hadoop-env.sh : 자바 홈디렉토리와 hadoop 홈디렉토리가 어딘지 지정.
2. core-site.xml : 하둡의 네임노드가 어느 서버인지를 지정한다.
3. mapred-site.xml : java로 만들어진 mapreduce 프레임워크와 관련된 정보를 지정.
4. hdfs-site.xml : 하둡 파일 시스템인 HDFS(Hadoop Distributed File System)와 관련된 정보를 저장하는 파일

```
[oracle@edydr1p2 ~]$ cd $HADOOP_HOME/conf
```

```
[orcl:~]$ cd $HADOOP_HOME/conf
[orcl:conf]$ pwd
/u01/app/hadoop/hadoop-1.2.1/conf
```

```
[oracle@edydr1p2 conf]$ vi hadoop-env.sh
```

아래쪽에 3줄을 붙여 넣는다.


```
# 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=/u01/app/hadoop/hadoop-1.2.1
export HADOOP_HOME_WARN_SUPPRESS=1
```

=====

```
# 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=/u01/app/hadoop/hadoop-1.2.1
#==> export HADOOP_HOME에 /u01/app/hadoop/hadoop-1.2.1(절대경로)를 넣어라.
export HADOOP_HOME_WARN_SUPPRESS=1
```

=====

[oracle@edydr1p2 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>/u01/app/hadoop/hadoop-1.2.1/hadoop-${user.name}</value>
  </property>
</configuration>
=====
```

[oracle@edydr1p2 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>
=====

```

```

[oracle@edydr1p2 conf]$ mkdir /u01/app/hadoop/hadoop-1.2.1/dfs
[oracle@edydr1p2 conf]$ mkdir /u01/app/hadoop/hadoop-1.2.1/dfs/name
[oracle@edydr1p2 conf]$ mkdir /u01/app/hadoop/hadoop-1.2.1/dfs/data

```

```

[oracle@edydr1p2 conf]$ vi hdfs-site.xml
=====
<configuration>
  <property>
    <name>dfs.name.dir</name>
    <value>/u01/app/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>/u01/app/hadoop/hadoop-1.2.1/dfs/data</value>
  </property>
</configuration>
=====

```

7. 하둡 네임노드를 포맷한다.

```

[oracle@edydr1p2 conf]$ cd
[oracle@edydr1p2 ~]$ hadoop namenode -format

```

14/08/06 00:28:19 INFO namenode.NameNode: STARTUP_MSG:

/*****

STARTUP_MSG: Starting NameNode

STARTUP_MSG: host = edydr1p2.us.oracle.com/192.168.100.102

STARTUP_MSG: args = [-format]

STARTUP_MSG: version = 1.2.1

STARTUP_MSG: build =

<https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2> -r 1503152; compiled

by 'mattf' on Mon Jul 22 15:23:09 PDT 2013

STARTUP_MSG: java = 1.7.0_60

*****/

Re-format filesystem in /u01/app/hadoop/hadoop-1.2.1/dfs/name ? (Y or N) Y (대문자)

...

14/08/06 00:28:28 INFO namenode.FSEditLog: closing edit log: position=4,

editlog=/u01/app/hadoop/hadoop-1.2.1/dfs/name/current/edits

14/08/06 00:28:28 INFO namenode.FSEditLog: close success: truncate to 4,

editlog=/u01/app/hadoop/hadoop-1.2.1/dfs/name/current/edits

14/08/06 00:28:28 INFO common.Storage: Storage directory

/u01/app/hadoop/hadoop-1.2.1/dfs/name has been successfully formatted.

14/08/06 00:28:28 INFO namenode.NameNode: SHUTDOWN_MSG:

/*****

SHUTDOWN_MSG: Shutting down NameNode at edydr1p2.us.oracle.com/192.168.100.102

*****/

8. 하둡을 시작한다.

[oracle@edydr1p2 ~]\$ start-all.sh #아침에 와서 해야한다.

...

\$ stop-all.sh #내리는 명령어

9. 하둡이 잘 시작되었는지 확인한다.

[oracle@edydr1p2 ~]\$ jps

9167 SecondaryNameNode

9030 DataNode

9402 TaskTracker

9250 JobTracker

9494 Jps

8883 NameNode

[orcl:~]\$ jps

32570 Jps

32090 DataNode

32241 SecondaryNameNode

31958 NameNode

32460 TaskTracker

32324 JobTracker

3. Hive 설치

: 자바를 몰라도 rdbms에 익숙한 데이터 분석가들을 위해서 SQL을 이용해서 하둡의 맵리듀싱 프로그램을 지원하는 프로그램 (페이스북에서 만든 오픈소스)

HiveQL

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

1) hive 설치 파일의 압축을 푼다.

```
$ tar xvzf hive-0.12.0.tar.gz
```

2) hive로 접속한다.

```
[orcl:~]$ cd /home/oracle/hive-0.12.0/bin
```

```
[orcl:bin]$ ./hive
```

```
hive> show tables;
```

```
hive> show tables;
OK
Time taken: 5.8 seconds
```

만약 이렇게 안뜨고 faied가 뜨면 exit치고 나가서 다시 start-all.sh를 한다.
이 방법도 안되면 하둡을 다시 설치해야한다.

hive 안들어가질때

```
[orcl:~]$ . .bash_profile 한번 돌리고
```

```
[orcl:~]$ start-all.sh
```

```
[orcl:~]$ cd hive-0.12.0
```

```
[orcl:hive-0.12.0]$ cd bin
```

```
[orcl:bin]$ ./hive
```

순서대로 하면됨

문제1. hive에서 emp 테이블을 생성하시오!.

답)

```
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 ;

문제2. 하둡 파일 시스템에 emp2.csv를 올리시오.

답)

emp2.csv 를 /home/oracle에 올린다.

```
$ hadoop fs -put emp2.csv emp2.csv
```

```
$ hadoop fs -ls emp2.csv
```

```
hive> exit;
[orcl:bin]$ cd
[orcl:~]$ hadoop fs -put emp2.csv emp2.csv
[orcl:~]$ hadoop fs -ls emp2.csv
Found 1 items
-rw-r--r--  3 oracle supergroup          644 2018-07-03 15:26 /user/oracle/emp2.csv
```

문제3. emp2.csv를 emp에 로드하시오.

답)

```
hive> load data inpath '/user/oracle/emp2.csv'
      overwrite into table emp;
```

```
hive> load data inpath '/user/oracle/emp2.csv'
      > overwrite into table emp;
Loading data to table default.emp
Table default.emp stats: [num_partitions: 0, num_rows: 0]
OK
Time taken: 4.506 seconds
```

```
hive> select * from emp;
hive> select * from emp;
OK
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
Time taken: 0.283 seconds, Fetched: 14 row(s)
```

• 하둡 시스템이 정상인지 확인하는 명령어

jps

```
hive> exit;
```

```
[orcl:bin]$ cd
```

```
[orcl:~]$ jps
```

DataNode : hdfs의 데이터를 입력하면 입력 데이터는 32mb 블록으로 나뉘어서 여러대의 데이터 노드에 분산되어 저장된다. 그 데이터를 저장하는 노드

SecondaryNameNode : 하둡 보조 네임노드로 네임노드의 파일 시스템 이미지 파일을 주기적으로 갱신하는 역할을 수행하는 노드 (메타 데이터를 최신으로 유지시키는 작업)

NameNode : HDFS의 모든 메타 데이터(data의 위치정보)를 관리하고 클라이언트가 hdfs에 저장된 파일에 접근할 수 있게 해준다.

TaskTracker : 사용자가 설정한 맵리듀스 프로그램을 실행하는 역할을 하며 하둡 데이터 노드에서 실행되는 데몬.

JobTracker : 하둡 클러스터에 등록된 전체 잡의 스케줄링을 관리하고 모니터링하는 데몬

Jps : 현재 jps 명령어 수행한 프로세서 (나)

\$ jps <-- 명령어를 수행했을때 6개의 데몬 말고 RunJar라는 프로세스가 뜨면 문제가 있으므로 반드시 kill 시킨다.

\$ kill -9 RunJar의 프로세스번호

문제4. 하둡 파일 시스템 관리 명령어를 자동화 시키시오.

\$ vi m2.sh

7. 하둡 파일 시스템을 중단 시키려면 5번을
8. 하둡 파일 시스템을 시작 시키려면 6번을
9. 하둡 파일 시스템의 상태를 확인하려면 7번을

4. 하둡 분산 파일 시스템 명령어

1. ls -> 지정된 디렉토리에 있는 파일의 정보를 출력
2. ls -> 현재 디렉토리 뿐만 아니라 하위 디렉토리까지 조회
3. du -> 파일의 용량 확인
4. dus -> 파일의 전체 합계 용량 확인
5. cat -> 지정된 파일의 내용을 화면에 출력
6. text -> zip 파일 형태도 text 형태로 화면에 출력
7. mkdir -> 디렉토리 생성
8. put -> 파일을 하둡 파일 시스템에 올리는 명령어
9. copyFromLocal -> 파일 복사
10. get -> 하둡 파일 시스템의 파일을 리눅스 디렉토리로 내리는 명령어

11. getmerge -> 지정된 경로에 있는 모든 파일의 내용을 합친후 하나의 파일로 복사하는 명령어
12. mv -> 파일을 이동하는 명령어
13. moveFromLocal -> 복사후 원본 파일 삭제
14. rm -> 파일 삭제
15. rmr -> 디렉토리 삭제
16. count -> 지정된 디렉토리의 파일의 개수 확인
17. tail -> 파일의 마지막 내용확인
18. chmod -> 권한 변경
19. chown -> 소유자 변경
20. touch -> 0바이트 파일 생성
21. stat -> 통계정보 조회
22. expunge -> 휴지통 비우기
23. grep -> 파일에서 특정 문자의 라인을 검색
24. awk -> 파일의 특정 컬럼을 검색

문제5. 하둡 파일 시스템 명령어의 help를 조회하시오.

```
[orcl:~]$ hadoop fs -help
[orcl:~]$ hadoop fs -help du
[orcl:~]$ hadoop fs -help du
-du <path>:      Show the amount of space, in bytes, used by the files that
                  match the specified file pattern.  Equivalent to the unix
                  command "du -sb <path>/*" in case of a directory,
                  and to "du -b <path>" in case of a file.
                  The output is in the form
                      name(full path) size (in bytes)
```

문제6. 겨울왕국 대본 winter.txt를 하둡 파일 시스템에 올리시오.

답)

```
[orcl:~]$ hadoop fs -put /home/oracle/winter.txt winter.txt
[orcl:~]$ hadoop fs -ls winter.txt
Found 1 items
-rw-r--r--  3 oracle supergroup      114548 2018-07-03 16:27 /user/oracle/winter.txt
```

문제7. 하둡 파일 시스템에 emp2.csv 파일을 올리시오.

답)

```
[orcl:~]$ hadoop fs -put emp2.csv emp2.csv
```

문제8. 하둡 파일 시스템에 emp2.csv를 cat으로 조회하시오.

```
[orcl:~]$ hadoop fs -cat emp2.csv
[orcl:~]$ hadoop fs -cat emp2.csv
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,7600,1981-02-18,1250,1400,30
```

```
[orcl:~]$ hadoop fs -cat emp2.csv
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
```

문제9. 직업이 salesman인 직원들의 모든 데이터를 출력하시오.

답)

```
[orcl:~]$ hadoop fs -cat emp2.csv | grep -i "salesman"
[orcl:~]$ hadoop fs -cat emp2.csv | grep -i "salesman"
7654,MARTIN,SALESMAN,7698,1981-09-10,1250,1400,30
7499,ALLEN,SALESMAN,7698,1981-02-11,1600,300,30
7844,TURNER,SALESMAN,7698,1981-08-21,1500,0,30
7521,WARD,SALESMAN,7698,1981-02-23,1250,500,30
```

문제10. alias를 만들어서 아래와 같이 하둡 명령어를 쓸 수 있도록 하시오.

보기)

```
$h -cat emp2.csv
```

결과)

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

답)

```
[orcl:~]$ alias h='hadoop fs'
```

문제11. hive에 접속할 때 어느 디렉토리에서든 편하게 hive라고 하면 접속되게 하시오.

```
$ cd
$ cd hive-0.12.0
$ pwd
/home/oracle/hive-0.12.0 <- 복사
```

```
$cd
```



```
$ vi .bash_profile
```

```
export HIVE_HOME=/home/oracle/hive-0.12.0
```

```
export PATH=$HIVE_HOME/bin:$PATH
```

를 붙여넣으면 어디에서든 하이브로 들어갈 수 있다.

```
[orcl:~]$ . .bash_profile
```

여기는 운영서버 입니다. 조심하 작업하세요~

저한테 왜 그러시는 건데요!!

```
[orcl:~]$ hive
```

문제12. (마지막)dept2.csv 를 하둡 파일 시스템에 로드하고, dept2 테이블을 hive에서 생성한 후 dept.csv를 dept2 테이블에 입력하시오.

일단 테이블 생성

```
create table dept
```

```
(deptno int,
```

```
  dname string,
```

```
  loc string )
```

```
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 ;
OK
Time taken: 5.964 seconds
```

```
[orcl:~]$ h -put dept2.csv dept2.csv
```

```
hive> load data inpath '/user/oracle/dept2.csv'
```

```
> overwrite into table dept;
```

```
hive> load data inpath '/user/oracle/dept2.csv'
> overwrite into table dept;
Loading data to table default.dept
Table default.dept stats: [num_partitions: 0, nu
OK
Time taken: 4.413 seconds
```

```
hive> select * from dept;
```

```
hive> select * from dept;
OK
10      ACCOUNTING      NEW YORK
```

```
hive> select * from dept;
OK
10      ACCOUNTING      NEW YORK
20      RESEARCH        DALLAS
30      SALES            CHICAGO
40      OPERATIONS      BOSTON
Time taken: 0.26 seconds, Fetched: 4 row(s)
```

문제13. hive에서 월급이 3000인 사원의 사원번호와 이름과 월급을 출력하시오.

답)

```
hive> select empno, ename, sal
> from emp
> where sal = 3000;
```

```
OK
7902    FORD      3000
7788    SCOTT     3000
Time taken: 11.907 seconds, Fetched: 2 row(s)
```

tip) hive> set hive.cli.print.header=true; 을치면 컬럼명도 같이 나온다.

```
empno  ename  job      mgr      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
Time taken: 0.066 seconds, Fetched: 14 row(s)
```

5. Hive로 정렬 (17장)

order by 정렬

문제14. 직업, 직업별 토털월급을 출력하는데 직업별 토털월급이 높은 것 부터 출력하시오.

답)

```
hive> select job, sum(sal) sumsal
> from emp
> group by job
> order by sumsal desc;
```

```
OK
job      sumsal
MANAGER  8275
ANALYST  6000
SALESMAN  5600
PRESIDENT 5000
```

- hive에서 그룹 함수를 order by 할때는 컬럼별칭을 사용해야한다.

문제15. 부서번호, 부서번호별 평균월급을 출력하는데 부서번호별 평균월급이 낮은것 부터 출력하시오.

답)

```
hive> select deptno, sum(sal) sumsal
      > from emp
      > group by deptno
      > order by sumsal asc;
```

deptno	sumsal
10	8750
30	9400
20	10875

문제16. jps 명령어를 수행했을때 RunJar 프로세스의 프로세스 번호만 출력하시오.

보기)

```
$ sh r.sh
11192
```

답)

```
$ vi r.sh
```

```
jps | grep -i 'runjar' | awk '{print $1}'
```

```
[orcl:~]$ sh r.sh
11192
```

문제17. RunJar 프로세스를 kill 시키는 쉘을 생성하시오.

보기)

```
$ sh r.sh
```

답)

```
pid=`jps | grep -i 'runjar' | awk '{print $1}'`
if [ kill -9 $pid ]; then
    echo "RunJar프로세스가 죽었습니다"
else
    echo "RunJar프로세스가 없습니다"
fi
```

문제18. RunJar 프로세스 kill 시키는 쉘을 자동화 스크립트 10번으로 추가하시오.

```
echo "
```

1. Sar 그래프
2. csv 파일 생성
3. 파일의 특정 단어 수 찾기

4. swp파일 모두 삭제
5. 두개 파일의 차이 찾기
6. SQL의 테이블 csv로 생성
7. 하둡 파일 시스템을 중단 시키려면 7번을
8. 하둡 파일 시스템을 시작 시키려면 8번을
9. 하둡 파일 시스템의 상태를 확인하려면 9번을
10. RunJar 프로세스를 죽이려면 10번을

"

echo " "

echo -n "번호를 입력하세요"

read choice

case \$choice in

1)

/home/oracle/Sar.sh ;;

2)

/home/oracle/find_file.sh ;;

3)

/home/oracle/fine_word.sh ;;

4)

/home/oracle/rmswp.sh ;;

5)

/home/oracle/diff.sh ;;

6)

/home/oracle/hr_.sh ;;

7)

stop-all.sh ;;

8)

start-all.sh ;;

9)

jps ;;

10)

sh r.sh ;;

esac

6. Hive로 조인 (17장)

full outer join

문제19. dept 테이블이 존재 하는지 확인해보고 emp와 dept를 조인해서이름과 부서위치를 출력하시오.(hive 에서는 1999 ansi 조인문법만 지원된다.)

답)

```
hive> select * from dept;
```

```
hive> select e.ename, d.loc
```

```
> from emp e full outer join dept d
```

```
> on(e.deptno = d.deptno);
```

MILLER	NEW YORK
CLARK	NEW YORK
KING	NEW YORK
JONES	DALLAS
FORD	DALLAS
SMITH	DALLAS
SCOTT	DALLAS
ADAMS	DALLAS
MARTIN	CHICAGO
ALLEN	CHICAGO
TURNER	CHICAGO
JAMES	CHICAGO
WARD	CHICAGO
BLAKE	CHICAGO
NULL	BOSTON

문제20. 부서위치, 부서위치별 토탈월급을 출력하시오.

답)

```
hive> select d.loc, sum(e.sal) sumsal
```

```
> from emp e full outer join dept d
```

```
> on(e.deptno = d.deptno)
```

```
> group by d.loc;
```

OK
BOSTON NULL
CHICAGO 9400
DALLAS 10875

문제21. full outer join이 가능한지 확인해보시오.

답)

위에서 했다 가능하다.

- hive에서 data 분석 함수가 수행되는지 확인

문제22. 이름, 월급, 순위를 출력하는데 순위가 월급이 높은 사원부터 출력되게 하시오.

답)

```
select ename, sal, rank() over (order by sal desc) rnk
```

```
from emp;
```

KING	5000	1
FORD	3000	2
SCOTT	3000	2
JONES	2975	4
BLAKE	2850	5
CLARK	2450	6
ALLEN	1600	7
TURNER	1500	8
-----	-----	-

KING	5000	1
FORD	3000	2
SCOTT	3000	2
JONES	2975	4
BLAKE	2850	5
CLARK	2450	6
ALLEN	1600	7
TURNER	1500	8
MILLER	1300	9
WARD	1250	10
MARTIN	1250	10
ADAMS	1100	12
JAMES	950	13
SMITH	800	14

문제23. 부서번호, 이름, 월급, 순위를 출력하는데 순위가 부서번호별로 각각 월급이 높은
사원순으로 순위를 부여하시오.

답)

```
select deptno, ename, sal, rank() over (partition by deptno order by sal desc) rnk
from emp;
```

10	KING	5000	1
10	CLARK	2450	2
10	MILLER	1300	3
20	FORD	3000	1
20	SCOTT	3000	1
20	JONES	2975	3
20	ADAMS	1100	4
20	SMITH	800	5
30	BLAKE	2850	1
30	ALLEN	1600	2
30	TURNER	1500	3
30	MARTIN	1250	4
30	WARD	1250	4
30	JAMES	950	6

문제24. 부서위치, 이름, 월급, 순위를 출력하는데 순위가 부서위치별로 각각 월급이 높은
사원순으로 순위를 부여하시오.

답)

```
select d.loc, e.ename, e.sal, rank() over(partition by d.loc order by e.sal desc) rnk
from emp e join dept d
on(e.deptno = d.deptno);
```

CHICAGO	BLAKE	2850	1
CHICAGO	ALLEN	1600	2
CHICAGO	TURNER	1500	3
CHICAGO	MARTIN	1250	4
CHICAGO	WARD	1250	4
CHICAGO	JAMES	950	6
DALLAS	SCOTT	3000	1
DALLAS	FORD	3000	1
DALLAS	JONES	2975	3
DALLAS	ADAMS	1100	4
DALLAS	SMITH	800	5
NEW YORK	KING	5000	1
NEW YORK	CLARK	2450	2
NEW YORK	MILLER	1300	3

문제25. 사원번호, 이름, 월급, 월급의 누적치가 출력되게 하시오.

보기)

7788 scott 3000 3000

7902 smith 1200 4200

.

.

답)

```
select empno, ename, sal, sum(sal) over (order by empno) stack
from emp;
```

7369	SMITH	800	800
7499	ALLEN	1600	2400
7521	WARD	1250	3650
7566	JONES	2975	6625
7654	MARTIN	1250	7875
7698	BLAKE	2850	10725
7782	CLARK	2450	13175
7788	SCOTT	3000	16175
7839	KING	5000	21175
7844	TURNER	1500	22675
7876	ADAMS	1100	23775
7900	JAMES	950	24725
7902	FORD	3000	27725
7934	MILLER	1300	29025

- 하이브(hive)에서 지원하는 내장함수

1. concat
2. substr
3. upper
4. lower
5. trim
6. rtrim
7. ltrim
8. regexp_replace
9. to_date
10. round
11. floor
12. ceil
13. year
14. month
15. day
16. 데이터 분석함수

문제26. 이름을 출력하는데 이름의 첫 글자만 출력하고, 그 첫 글자를 소문자로 출력하시오.

답)

```
select substr(lower(ename),1,1)
from emp;
```

OK
k
b
c
j
m

```
OK
k
b
c
j
m
a
t
j
w
f
s
s
a
m
```

문제27. 이름, 입사한 년도 4자리만 출력하시오.

답)

```
select ename, year(hiredate)
from emp;
```

```
KING      1981
BLAKE     1981
CLARK     1981
JONES     1981
MARTIN    1981
ALLEN     1981
TURNER    1981
JAMES     1981
WARD      1981
FORD      1981
SMITH     1980
SCOTT     1982
ADAMS     1983
MILLER    1982
```

문제28. 이름, 입사한 달을 출력하시오.

답)

```
select ename, month(hiredate)
from emp;
```

```
KING      11
BLAKE     5
CLARK     5
JONES     4
MARTIN    9
ALLEN     2
TURNER    8
JAMES     12
WARD      2
FORD      12
SMITH     12
SCOTT     12
ADAMS     1
MILLER    1
```

문제29. 부서번호, 부서번호별 토탈월급을 가로로 출력하시오.

답)

set hive.cli.print.header=true; <- set 을 설정 안해주면 컬럼명이 안나온다.


```
select sum(case when deptno=10 then sal end) dept10,
sum(case when deptno=20 then sal end) dept20,
sum(case when deptno=30 then sal end) dept30
from emp;
```

```
dept10 dept20 dept30
8750    10875   9400
```

가로로 출력하는 방법 (오라클에서 sum(decode))

문제30. 직업, 부서번호, 직업별 부서번호별 평균 월급을 가로로 출력하시오.

답)

```
set hive.cli.print.header=true;
```

```
select job, avg(case when deptno=10 then sal end) dept10,
avg(case when deptno=20 then sal end) dept20,
avg(case when deptno=30 then sal end) dept30
from emp
group by job;
```

```
job      dept10 dept20 dept30
ANALYST  NULL   3000.0 NULL
CLERK    1300.0 950.0  950.0
MANAGER  2450.0 2975.0 2850.0
PRESIDENT 5000.0 NULL   NULL
SALESMAN  NULL   NULL   1400.0
```

문제31. 직업, 직업별 인원수를 출력하시오.

답)

```
select job, count(*)
from emp
group by job;
```

```
job      _c1
ANALYST  2
CLERK    4
MANAGER  3
PRESIDENT 1
SALESMAN 4
```

문제32. (점심시간) 아래의 SQL을 HIVE로 구현하시오.

보기)

```
oracle> select job, count( decode(deptno,10,empno) ) "dept10",
count( decode(deptno,20,empno) ) "dept20",
count( decode(deptno,30,empno) ) "dept30"
from emp;
```

답)

```
select job, count(case when deptno=10 then empno end) dept10,
count(case when deptno=20 then empno end) dept20,
count(case when deptno=30 then empno end) dept30
from emp
group by job;
```

job	dept10	dept20	dept30
ANALYST	0	2	0
CLERK	1	2	1
MANAGER	1	1	1
PRESIDENT	1	0	0
SALESMAN	0	0	4

- hive 가 서브쿼리는 지원 안되지만 from 절의 서브쿼리는 가능하다.

문제33. 사원 테이블에서 월급의 순위가 1등인 사원의 이름과 월급과 순위를 출력하시오.
답)

```
select * from (select ename, sal, rank() over (order by sal desc) rnk from emp) aaa
where rnk = 1;
```

KING	5000	1
------	------	---

설명 : Hive에서는 from절의 서브쿼리를 alias를 사용해야 한다.

rollup

문제34. 아래의 오라클 SQL을 Hive로 구현하시오.

보기)

```
Oracle> select deptno, sum(sal)
          from emp
          group by rollup(deptno);
```

답1)

```
select *
from( select deptno, sum(sal)
      from emp
      group by deptno
      union all
      select null as deptno, sum(sal)
      from emp ) aaa;
```

답2)

```
select deptno, sum(sal)
from emp
group by deptno with rollup;
```

NULL	29025
10	8750
20	10875
30	9400

grouping

문제35. 아래의 오라클 SQL을 HIVE로 구현하시오.

보기)

```
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), () );
```

NULL	29025
10	8750
20	10875
30	9400

문제36. 아래의 오라클 SQL을 Hive로 구현하시오.

보기)

```
Oracle> select deptno, job, sum(sal)
        from emp
        group by grouping sets(deptno),(job),());
```

답)

```
select deptno, job, sum(sal)
from emp
group by deptno,job grouping sets((deptno),(job),());
```

문제37. 오라클 with절이 hive에서도 가능한지 확인하시오.

보기)

```
Oracle> with job_sumsal ( select job, sum(sal) sumsal
                        from emp
                        group by job )
        select job, sumsal
        from job_sumsal
        where sumsal > (select avg(sumsal)
                        from job_sumsal);
```

JOB	SUMSAL
MANAGER	8275
ANALYST	6000

답)

안된다!!!!!!!

문제38. 사원 테이블의 직업의 종류가 몇가지인지 출력하시오.

```
hive> select count(distinct(job))  
> from emp;  
5
```

- hive는 update, delete 는 지원 안함.
insert는 append는 가능하다.

문제39. scott의 월급을 9000으로 변경하시오.

```
hive> update emp  
> set sal=9000  
> where ename='SCOTT';  
=> 하이브에서 안된다!!!!!!
```

문제40. 사원 테이블을 전부 삭제하시오.

```
hive> delete from emp;  
=> 하이브에서 안된다!!!!!!
```

문제41. emp테이블을 emp_backup으로 생성하시오.

답)

```
hive> create table emp_backup  
> as  
> select * from emp;  
OK
```

```
hive> select * from emp_backup;  
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  
Time taken: 0.064 seconds, Fetched: 14 row(s)
```

문제42. emp 테이블의 데이터를 emp_backup에 insert 하시오.

답)

```
hive> insert into table emp_backup
> select *
> from emp;
OK
hive> select count(*) from emp_backup;
28
```

문제43. emp_backup 의 내용을 emp테이블의 내용으로 overwrite 하시오.

답)

```
hive> insert overwrite table emp_backup
> select *
> from emp;
OK

hive> select count(*) from emp_backup;

14
```

7. 타조

• 타조(Tajo) 설치

Tajo는 무엇?

:

최현식 박사는 타조를 하둡 기반의 분산데이터웨어하우스 시스템이라고 설명했다. SQL을 사용해 빅데이터를 분석할 때 맵리듀스를 사용하지 않고 HDFS 파일을 바로 읽어내는 기술이다.

그는 “아파치 타조는 하둡의 하이브를 대체하는 기술을 만들자는 목표로 시작됐고, 효율성과 로레이턴시 시스템을 향해 발전하고 있다”라고 말했다.

그는 “맵리듀스 프로그램 대신 자체 설계한 처리 엔진을 통해 하이브보다 훨씬 빠른 처리 시간을 보여준다”라며 “비공유클러스터와도 호환되고, 네트워크 보틀넥이 없는 설계구조를 갖는다”라고 설명했다.

타조 개발진은 비공유 구조 상에서 발생하는 네트워크 보틀넥을 보완하기 위해 셔플 횟수를 줄일 수 있는 실행모델을 고안했다. DB에 널리 사용되는 코스트 기반 쿼리 최적화, 컬럼 익스큐션 엔진 등의 각종 기법을 적용하면서, 레거시 시스템까지 연동할 수 있는 방안도 고민이었다.

타조는 무엇보다 온라인 상에서 SQL 입력 데이터 처리를 완료하기 전에도 결과값을 볼 수 있다는 특징을 갖고 있다.

그는 “큰 그림 위주로 결과값을 확인해야 하는 필요가 있는데, 타조는 중간 결과값을 확인하고 정확하다고 판단되면 중간에 중단하는 것도 가능하다”라고 설명했다.

특히 타조는 하나의 SQL 쿼리처리를 위해 필요한 잡의 오버헤드를 줄이는데 역점을 둔다. 그 결과 비용을 줄이면서, 아무리 큰 쿼리라도 하나의 잡으로 처리해 성능을 높인다.

그에 따르면, 작년 8월 TPC-H 테스트 당시 타조는 하이브 보다 모든 쿼리에서 앞선 성능을 보였다.

그는 타조와 임팔라를 비교하기도 했다. 드릴은 맵R을 중심으로 개발되고 있던 유사기술이다.

그는 “타조는 노드의 메모리보다 큰 데이터라고 해도 처리할 수 있는 성능을 보이지만, 임팔라는 작은 데이터만 효과적으로 처리하는 한계를 갖고 있다”라며 “또한 임팔라가 소스코드만 공개될 뿐 외부 개발자가 참여할 수 있는 부분이 적다는 것도 타조와 대비되는 점”이라고 말했다.

타조는 2010년 고려대학교 데이터베이스랩에서 시작됐다. 올해 3월 7일로 아파치재단 인큐베이터로 선정되면서 그 세를 불렀다. 그루터 인력이 적극 참여해 타조 고도화에 기여하고 있으며, 개발 속도에 힘을 붙이고 있다.

아키텍처는 마스터 워커 모델을 따른다. 마스터 한 대와 다수의 워커로 구성되고, 하둡의 안(YARN)을 리소스 플랫폼으로 사용해 맵리듀스나 하이브, 피그 등의 하둡 클러스터도 공유할 수 있다. 각 워커는 로컬 쿼리엔진이 HDFS나 로컬 파일시스템, 다른 데이터 소스 테이블을 처리할 수 있게 설계됐다.

그는 “막 인큐베이팅된 만큼 빠른 시일 내 아파치 이름으로 첫 번째 버전을 내놓을 것”이라며 “이후 API나 기능을 논의과정을 거쳐 표준화하고, SQL 풀스펙 지원, 코스트 기반 최적화 개선, 네이티브 컬럼 익스큐션 엔진 등을 개선해나갈 계획이다”라고 강조했다.

1. 타조 설치 파일을 /home/oracle 밑에 가져다 둡니다.

2. 압축 해제 합니다.

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

3. 작업을 쉽게 하기 위해서 리눅스 심볼릭 링크 이용

```
ln -s tajo-0.11.1-desktop-r1 tajo
```

- 심볼릭 링크 지우는 법

```
rm -rf [심볼릭 링크명]
```

4. tajo 디렉토리로 들어갑니다.

```
cd tajo
```

한번더 tajo를 들어가 줘야 합니다

```
cd tajo
```

압축해제 후 Tajo 셋팅 부분

자바 path는 자동으로 잡아줍니다.

```
[orcl:tajo]$ sh bin/configure.sh
```

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

추가적으로 옵션을 주기위한 부분이기 때문에 N으로 설정합니다

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

5. 타조 실행 명령어

```
[orcl:tajo]$ sh bin/startup.sh
```

```
starting master, logging to /home/oracle/tajo/tajo/bin/../logs/tajo-oracle-master-  
edydr1p0.us.oracle.com.out
```

```
Tajo master started
```

```
starting worker, logging to /home/oracle/tajo/tajo/bin/../logs/tajo-oracle-worker-  
edydr1p0.us.oracle.com.out
```

```
Tajo worker started
```

```
Tajo master web UI: http://edydr1p0.us.oracle.com:xxxxx
```

```
Tajo Client Service: edydr1p0.us.oracle.com:xxxxxx
```

#타조 실행 중단 명령어

```
#[orcl:tajo]$ sh bin/stop-tajo.sh
```

6. 타조 쉘 작동 확인

```
[orcl:tajo]$ sh bin/tsql
```

TSQL은 HDFS와 같은 실행기능을 제공하며, wdfs 옵션으로 HDFS 명령어를 실행할 수 있습니다.

Try ~~W~~? for help.

7. orcl 용 데이터베이스를 생성한 후 접속을 변경합니다.

DB 생성 명령어 이름은 orcl 로 생성

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

```
default> wc orcl;
```

8. emp table 생성 방법

```
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/oracle/emp2.csv';
```

9. emp 테이블의 메타 데이터 조회하는 방법

(테이블 및 함수의 메타 데이터는 wd 옵션으로 조회할 수 있습니다)

```
emp> Wd emp;
```

table name: emp.emp

```
table uri: file:///home/oracle/emp.csv
```

store type: TEXT

number of rows: unknown

volume: 721 B

Options:

```
'text.delimiter'=','
```

schema:

empno INT4

ename TEXT
job TEXT
mgrno INT4
hiredate TEXT
sal INT4
comm INT4
deptno INT4

emp> select * from emp;

```
7369, SMITH, CLERK, 7902, 1980-12-17, 800, , 20
7499, ALLEN, SALESMAN, 7698, 1981-02-20, 1600, 300, 30
7521, WARD, SALESMAN, 7698, 1981-02-22, 1250, 500, 30
7566, JONES, MANAGER, 7839, 1981-04-02, 2975, , 20
7654, MARTIN, SALESMAN, 7698, 1981-09-28, 1250, 1400, 30
7698, BLAKE, MANAGER, 7839, 1981-05-01, 2850, , 30
7782, CLARK, MANAGER, 7839, 1981-06-09, 2450, , 10
7788, SCOTT, ANALYST, 7566, 1987-04-19, 3000, , 20
7839, KING, PRESIDENT, , 1981-11-17, 5000, , 10
7844, TURNER, SALESMAN, 7698, 1981-09-08, 1500, 0, 30
7876, ADAMS, CLERK, 7788, 1987-05-23, 1100, , 20
7900, JAMES, CLERK, 7698, 1981-12-03, 950, , 30
7902, FORD, ANALYST, 7566, 1981-12-03, 3000, , 20
7934, MILLER, CLERK, 7782, 1982-01-23, 1300, , 10
9292, JACK, CLERK, 7782, 1982-01-23, 3200, , 70
(16 rows, 0.424 sec, 0 B selected)
```

이상 타조 비행 성공~~~~~

문제44. 월급이 3000이상인 직원들의 이름과 월급과 직업을 출력하시오.

```
orcl> select ename, sal
> from emp
> where sal >= 3000;
```

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

like

문제45. 이름의 첫글자가 S로 시작하는 직원들의 이름과 월급을 출력하시오.

답)

```
orcl> select ename, sal
```

```
> from emp
> where ename like 'S%';
SMITH, 800
SCOTT, 3000
```

문제46. 월급이 1000에서 3000사이인 직원들의 이름과 월급을 출력하시오.

답)

```
orcl> select ename, sal
> from emp
> where sal between 1000 and 3000;
```

```
BLAKE, 2850
CLARK, 2450
JONES, 2975
MARTIN, 1250
ALLEN, 1600
TURNER, 1500
WARD, 1250
FORD, 3000
SCOTT, 3000
ADAMS, 1100
MILLER, 1300
```

문제47. 커미션이 null인 직원들의 이름과 커미션을 출력하시오.

답)

```
orcl> select ename, comm
> from emp
> where comm is null;
```

null인 사원이없다!!!!

문제48. 직업, 직업별 토털월급을 출력하시오.

```
orcl> select ename, comm
> from emp
> where comm is null;
```

```
MANAGER, 8275
CLERK, 4150
PRESIDENT, 5000
SALESMAN, 5600
ANALYST, 6000
```

having 절

문제49. having절이 지원되는지 확인하시오. 직업, 직업별 토털월급을 출력하는데 직업별 토털월급이 4000이상인 것만 출력하시오.

답)

```
orcl> select job, sum(sal)
> from emp
> group by job
```

```
> having sum(sal) >= 4000;
```

```
MANAGER, 8275  
CLERK, 4150  
PRESIDENT, 5000  
SALESMAN, 5600  
ANALYST, 6000
```

case

문제50. 부서번호, 부서번호별 토탈월급을 가로로 출력하시오.

답)

```
select sum(case when deptno=10 then sal end) dept10,  
sum(case when deptno=20 then sal end) dept20,  
sum(case when deptno=30 then sal end) dept30  
from emp;  
dept10, dept20, dept30  
-----  
8750, 10875, 9400
```

문제51. dept테이블을 생성하고 데이터를 로드하시오.

답)

```
CREATE EXTERNAL TABLE dept (  
deptno INT ,  
dname text ,  
loc TEXT  
) USING CSV WITH ('text.delimiter'=',' )  
LOCATION 'file:///home/oracle/dept2.csv';
```

조인

문제52. 이름과 부서위치를 출력하시오.

답)

```
orcl> select e.ename, d.loc  
> from emp e, dept d  
> where e.deptno=d.deptno;  
=> 오라클 조인이 가능하다
```

```
KING, NEW YORK  
BLAKE, CHICAGO  
CLARK, NEW YORK  
JONES, DALLAS  
MARTIN, CHICAGO  
ALLEN, CHICAGO  
TURNER, CHICAGO  
JAMES, CHICAGO  
WARD, CHICAGO  
FORD, DALLAS  
SMITH, DALLAS  
SCOTT, DALLAS  
ADAMS, DALLAS  
MILLER, NEW YORK
```

문제53. 이름과 부서위치를 출력하는데 **right outer join**을 써서 수행하시오.

답1)

```
orcl> select e.ename, d.loc  
> from emp e right outer join dept d  
> on e.deptno=d.deptno;
```

```
KING, NEW YORK  
CLARK, NEW YORK  
MILLER, NEW YORK  
JONES, DALLAS  
FORD, DALLAS  
SMITH, DALLAS  
SCOTT, DALLAS  
ADAMS, DALLAS  
BLAKE, CHICAGO  
MARTIN, CHICAGO  
ALLEN, CHICAGO  
TURNER, CHICAGO  
JAMES, CHICAGO  
WARD, CHICAGO
```

답2)

```
orcl> select e.ename, d.loc  
> from emp e, dept d  
> where e.deptno (+) = d.deptno;
```

=> 이건 안된다!!!!

문제54. 타조에서 **full outer join**을 지원하는지 확인하시오.

답)

```
orcl> select e.ename, d.loc  
> from emp e full outer join dept d  
> on e.deptno=d.deptno;
```

```
KING, NEW YORK  
BLAKE, CHICAGO  
CLARK, NEW YORK  
JONES, DALLAS  
MARTIN, CHICAGO  
ALLEN, CHICAGO  
TURNER, CHICAGO  
JAMES, CHICAGO  
WARD, CHICAGO  
FORD, DALLAS  
SMITH, DALLAS  
SCOTT, DALLAS  
ADAMS, DALLAS  
MILLER, NEW YORK
```

문제55. 타조에서 **rollup**이 지원되는지 확인해보시오.

답)

```
orcl> select deptno, sum(sal)  
> from emp
```

```
> group by rollup(deptno);
ERROR: unsupported feature: Rollup
```

안된다!!!!!!

```
orcl> select deptno, sum(sal) sumsal
> from emp
> group by deptno
> union all
> select 0, sum(sal) <-- null as deptno하면 안된다!!!
> from emp;
0, 29025
20, 10875
30, 9400
10, 8750
```

문제56. hive에서 지원 안되는 서브쿼리가 지원되는지 확인해보시오. JONES보다 더 많은 월급을 받는 직원들의 이름과 월급을 출력하시오.

답)

```
orcl> select ename, sal
> from emp
> where sal > (select sal from emp where ename='JONES');
```

안됨!!!

문제57. 이름, 월급, 순위를 출력하는데 순위가 월급이 높은 순서에 대한 순위로 출력되도록

답)

```
orcl> select ename, sal, rank() over (order by sal desc)
> from emp;
KING, 5000, 1
FORD, 3000, 2
SCOTT, 3000, 2
JONES, 2975, 4
BLAKE, 2850, 5
CLARK, 2450, 6
ALLEN, 1600, 7
TURNER, 1500, 8
MILLER, 1300, 9
MARTIN, 1250, 10
WARD, 1250, 10
ADAMS, 1100, 12
JAMES, 950, 13
SMITH, 800, 14
```

from절 서브쿼리

문제58. 위의 결과를 다시 출력하는데 순위가 1등인 직원만 출력하시오.



답)

```
orcl> select *
> from (select ename, sal, rank() over(order by sal desc) rn from emp) aaa
```

```
> where rnk = 1;
ename, sal, rnk
-----
KING, 5000, 1
```

문제59. 타조 가장 최신버전 설치파일을 다운로드 하시오.

<https://www.apache.org/dist/tajo/>

	tajo-0.11.0/	2018-05-01 13:38	-
	tajo-0.11.1/	2018-05-01 13:39	-
	tajo-0.11.2/	2018-05-01 13:39	-
	tajo-0.11.3/	2018-05-01 13:39	-
	tajo-0.11.3-src.tar.gz.md5	2016-05-18 23:35	64
	tajo-0.11.3-src.tar.gz.sha256	2018-03-17 09:59	99
	tajo-0.11.3.tar.gz	2016-05-18 23:35	37M
	tajo-0.11.3.tar.gz.asc	2018-03-17 09:59	819
	tajo-0.11.3.tar.gz.md5	2016-05-18 23:35	60
	tajo-0.11.3.tar.gz.sha256	2018-03-17 09:59	95
	tajo-jdbc-0.11.3.jar	2016-05-18 23:35	6.3M
	tajo-jdbc-0.11.3.jar.asc	2018-03-17 09:59	819

일단 여기까지!!!

#타조 실행 중단 명령어

```
#[orcl:tajo]$ sh bin/stop-tajo.sh
[orcl:tajo]$ sh bin/stop-tajo.sh
localhost: stopping worker
stopping master
```

타조를 중단시켜주고

최신버전을 다시 압축해제 해준다.

8. Pig

: 야후에서 만든 NoSQL (야후에서는 40%이상의 JOB을 Pig로 처리)

- "돼지들은 어떠한 것도 잘 먹는다." 라는 슬로건을 갖는다.
- 어떠한 데이터든 잘 소화할 수 있다.
- 사용자 정의함수(오라클의 프로시저와 같은 언어)를 지원한다.

• Pig 설치

1. Pig 설치파일을 /home/oracle에 올린다.

2. pig 설치 파일의 압축을 푼다.

```
$ tar xvf pig-0.12.0.tar.gz
```

3. pig 환경설정을 한다.

```
[orcl:~]$ ls -ld pig*
```

```
drwxr-xr-x 15 oracle oinstall 4096 Oct 8 2013 pig-0.12.0
-rw-r--r-- 1 oracle oinstall 59941262 Jul 5 09:56 pig-0.12.0.tar.gz
```

pig파일 이름을 보기 쉽게 변경한다.

```
[orcl:~]$ mv pig-0.12.0 pig
```

```
[orcl:~]$ cd pig
```

```
[orcl:pig]$ cd conf
```

```
[orcl:conf]$ vi pig.properties
```

아래 두줄을 맨아래쪽에 붙여 넣는다

```
-----
# Set this option to true if you need to use t
# Note: Old filter optimizer PColFilterOptimiz
# pig.exec.useOldPartitionFilterOptimize=true

fs.default.name=hdfs://localhost:9000
mapred.job.tracker=localhost:9001
-----
```

4. **.bash_profile에 PIG_HOME 디렉토리를 지정한다.**

```
$ cd
```

```
[orcl:~]$ vi .bash_profile
```

아래 두줄을 붙여 넣는다.

```
-----
export PIG_HOME=/home/oracle/pig
export PATH=$PIG_HOME/bin:$PATH
-----
```

5. **.bash_profile을 실행한다.**

```
[orcl:~]$ . .bash_profile
```

6. **pig에 접속한다.**

```
[orcl:~]$ pig -x local
```

문제60. pig에서 emp 테이블을 생성하시오.

답)

```
grunt> emp = LOAD '/home/oracle/emp2.csv' USING PigStorage(',') as (empno:int,
ename:chararray, job:chararray, mgr:int, hiredate:chararray, sal:int, comm:int, deptno:int);
```

```
grunt> DUMP emp;
```

```
(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)
```

이름하고 월급 뽑기

```
grunt> data = foreach emp generate ename, sal;
```

```
grunt> dump data;
```

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

문제61. 월급이 3000이상인 직원들의 이름과 월급을 출력하시오.

답)

```
grunt> data = foreach emp generate ename, sal;
```

```
grunt> data2 = filter data by sal >= 3000;
```

```
grunt> dump data2
```

```
(KING,5000)
(FORD,3000)
(SCOTT,3000)
```

문제62. 직업이 SALESMAN인 직원들의 이름과 월급과 직업을 출력하시오.

답)

```
grunt> data3 = foreach emp generate ename, sal, job;
```

```
grunt> data4 = filter data3 by job == 'SALESMAN'; <- ""안됨!! 무조건 째따써야함
```

```
grunt> dump data4
```

```
(MARTIN,1250,SALESMAN)
(ALLEN,1600,SALESMAN)
(TURNER,1500,SALESMAN)
(WARD,1250,SALESMAN)
```

Pig 정렬

문제63. 이름과 월급을 출력하는데 월급이 높은 직원부터 출력하시오.

답)

```
grunt> data = foreach emp generate ename, sal;
grunt> data2 = order data by sal desc;
grunt> dump data2
(KING, 5000)
(FORD, 3000)
(SCOTT, 3000)
(JONES, 2975)
(BLAKE, 2850)
(CLARK, 2450)
(ALLEN, 1600)
(TURNER, 1500)
(MILLER, 1300)
(WARD, 1250)
(MARTIN, 1250)
(ADAMS, 1100)
(JAMES, 950)
(SMITH, 800)
```

문제64. 부서번호가 30번인 직원들의 이름과 월급과 부서번호를 출력하는데, 월급이 높은 직원부터 출력하시오.

답)

```
grunt> data = foreach emp generate ename, sal, deptno;
grunt> data2 = filter data by deptno == 30;
grunt> data3 = order data2 by sal desc;
grunt> dump data3
(BLAKE, 2850, 30)
(ALLEN, 1600, 30)
(TURNER, 1500, 30)
(MARTIN, 1250, 30)
(WARD, 1250, 30)
(JAMES, 950, 30)
```

Pig group 함수

문제65. 직업, 직업별 인원수를 출력하시오.

보기)

```
SQL> select job, count(*)
      from emp
      group by job;
```

답)

```
grunt> data = foreach emp generate job;
grunt> data2 = group data by job;
grunt> dump data2
(CLERK, { (CLERK), (CLERK), (CLERK), (CLERK) })
(ANALYST, { (ANALYST), (ANALYST) })
(MANAGER, { (MANAGER), (MANAGER), (MANAGER) })
(SALESMAN, { (SALESMAN), (SALESMAN), (SALESMAN), (SALESMAN) })
(PRESIDENT, { (PRESIDENT) })

grunt> data3 = foreach data2 generate group, COUNT(data); <- 대문자 써야함!!!
grunt> dump data3
```

```
(CLERK, 4)
(ANALYST, 2)
(MANAGER, 3)
(SALESMAN, 4)
(PRESIDENT, 1)
```

문제66. 직업을 출력하는데 중복제거해서 출력하시오.

답)

```
grunt> data = foreach emp generate job;
grunt> data2 = group data by job;
grunt> data3 = foreach data2 generate group;
grunt> dump data3
(CLERK)
(ANALYST)
(MANAGER)
(SALESMAN)
(PRESIDENT)
```

문제67. 직업과 직업별 토탈월급을 출력하시오.

답)

```
grunt> data = foreach emp generate job, sal;
grunt> data2 = group data by job;
grunt> data3 = foreach data2 generate group, SUM(data.sal);
(CLERK, 4150)
(ANALYST, 6000)
(MANAGER, 8275)
(SALESMAN, 5600)
(PRESIDENT, 5000)
```

설명 : SUM을 대문자로 써야하고, sal을 가져올때 data에서 가져와야한다.

문제68. 부서번호, 부서번호별 토탈월급을 출력하는데 부서번호별 토탈월급이 높은 것부터 출력되게 하시오.

답)

```
grunt> data = foreach emp generate deptno, sal;
grunt> data2 = group data by deptno;
grunt> data3 = foreach data2 generate group, SUM(data.sal) as sumsal;
grunt> data4 = order data3 by sumsal desc;
grunt> dump data4
(20, 10875)
(30, 9400)
(10, 8750)
```

문제69. 직업과 직업별 토탈월급을 출력하는데, 직업이 SALESMAN인 사원은 제외하고 출력하고, 직업별 토탈 월급이 5000이상인 것만 출력하고, 직업별 토탈월급이 높은 것부터 출력하시오.

답)

```
grunt> data = foreach emp generate job, sal;
```

```

grunt> data2 = filter data by job != 'SALESMAN';
grunt> data3 = group data2 by job;
grunt> data4 = foreach data3 generate group, SUM(data2.sal) as sumsal;
grunt> data5 = order data4 by sumsal desc;
grunt> data6 = filter data5 by sumsal >= 5000;
(MANAGER, 8275)
(ANALYST, 6000)
(PRESIDENT, 5000)

```

주의!!!) data5 만들때 filter를 먼저 하면 오류가뜬다!!! 그래서 정렬을 먼저 해주고 해줘야함
(이유는 모름)

문제70. dept2.csv를 이용해서 dept 테이블을 pig에 생성하시오.

답)

```

grunt> dept = LOAD '/home/oracle/dept2.csv' USING PigStorage(',') as (deptno:int,
dname:chararray, loc:chararray);
DUMP dept;
(10, ACCOUNTING, NEW YORK)
(20, RESEARCH, DALLAS)
(30, SALES, CHICAGO)
(40, OPERATIONS, BOSTON)

```

문제71. 이름과 부서위치를 출력하시오.

답)

```

grunt> empdept = JOIN emp BY deptno, dept BY deptno;
(7934, MILLER, CLERK, 7782, 1982-01-11, 1300, 0, 10, 10, ACCOUNTING, NEW YORK)
(7782, CLARK, MANAGER, 7839, 1981-05-09, 2450, 0, 10, 10, ACCOUNTING, NEW YORK)
(7839, KING, PRESIDENT, 0, 1981-11-17, 5000, 0, 10, 10, ACCOUNTING, NEW YORK)
(7566, JONES, MANAGER, 7839, 1981-04-01, 2975, 0, 20, 20, RESEARCH, DALLAS)
(7902, FORD, ANALYST, 7566, 1981-12-11, 3000, 0, 20, 20, RESEARCH, DALLAS)
(7369, SMITH, CLERK, 7902, 1980-12-09, 800, 0, 20, 20, RESEARCH, DALLAS)
(7788, SCOTT, ANALYST, 7566, 1982-12-22, 3000, 0, 20, 20, RESEARCH, DALLAS)
(7876, ADAMS, CLERK, 7788, 1983-01-15, 1100, 0, 20, 20, RESEARCH, DALLAS)
(7654, MARTIN, SALESMAN, 7698, 1981-09-10, 1250, 1400, 30, 30, SALES, CHICAGO)
(7499, ALLEN, SALESMAN, 7698, 1981-02-11, 1600, 300, 30, 30, SALES, CHICAGO)
(7844, TURNER, SALESMAN, 7698, 1981-08-21, 1500, 0, 30, 30, SALES, CHICAGO)
(7900, JAMES, CLERK, 7698, 1981-12-11, 950, 0, 30, 30, SALES, CHICAGO)
(7521, WARD, SALESMAN, 7698, 1981-02-23, 1250, 500, 30, 30, SALES, CHICAGO)
(7698, BLAKE, MANAGER, 7839, 1981-05-01, 2850, 0, 30, 30, SALES, CHICAGO)
grunt> data = foreach empdept generate ename, loc;
grunt> dump data

```

```
(MILLER, NEW YORK)
(CLARK, NEW YORK)
(KING, NEW YORK)
(JONES, DALLAS)
(FORD, DALLAS)
(SMITH, DALLAS)
(SCOTT, DALLAS)
(ADAMS, DALLAS)
(MARTIN, CHICAGO)
(ALLEN, CHICAGO)
(TURNER, CHICAGO)
(JAMES, CHICAGO)
(WARD, CHICAGO)
(BLAKE, CHICAGO)
```

문제72. (점심시간) 부서위치, 부서위치별 토달월급을 출력하는데, 부서위치별 토달월급이 높은 것부터 출력하시오.

답)

```
grunt> empdept = JOIN emp BY deptno, dept BY deptno;
grunt> data = foreach empdept generate loc, sal;
grunt> data2 = group data by loc;
grunt> data3 = foreach data2 generate group, SUM(data.sal) as sumsal;
grunt> data4 = order data3 by sumsal desc;
grunt> dump data4
p0
(DALLAS, 10875)
(CHICAGO, 9400)
(NEW YORK, 8750)
```

문제73. dept 테이블을 hdfs에 올려서 쿼리하시오.

답)

```
$ hadoop fs -put /home/oracle/dept2.csv dept2.csv
$ hadoop fs -ls dept2.csv
```

```
grunt> dept = LOAD 'hdfs://localhost:9000/user/oracle/dept2.csv' USING PigStorage(',') as
(deptno:int, dname:chararray, loc:chararray);
```

```
grunt> DUMP dept;
(10, ACCOUNTING, NEW YORK)
(20, RESEARCH, DALLAS)
(30, SALES, CHICAGO)
(40, OPERATIONS, BOSTON)
```

```
grunt> data = foreach dept generate deptno, loc;
grunt> dump data
(10, NEW YORK)
(20, DALLAS)
(30, CHICAGO)
(40, BOSTON)
```

문제74. emp2.csv도 하둡 파일 시스템에 올려서 pig에서 하둡파일 시스템에 있는 emp2.scv로 emp테이블을 생성하시오.

답)

```
[orcl:~]$ hadoop fs -put /home/oracle/emp2.csv emp2.csv
[orcl:~]$ hadoop fs -ls emp2.csv
Found 1 items
-rw-r--r--  3 oracle supergroup      644 2018-07-04 10:17 /user/oracle/emp2.csv
```

```
grunt> emp = LOAD 'hdfs://localhost:9000/user/oracle/emp2.csv' USING PigStorage(',') as
(deptno:int, dname:chararray, loc:chararray);
```

```
grunt> DUMP emp;
```

```
(7839,KING,PRESIDENT)
(7698,BLAKE,MANAGER)
(7782,CLARK,MANAGER)
(7566,JONES,MANAGER)
(7654,MARTIN,SALESMAN)
(7499,ALLEN,SALESMAN)
(7844,TURNER,SALESMAN)
(7900,JAMES,CLERK)
(7521,WARD,SALESMAN)
(7902,FORD,ANALYST)
(7369,SMITH,CLERK)
(7788,SCOTT,ANALYST)
(7876,ADAMS,CLERK)
(7934,MILLER,CLERK)
```

9. 스쿱으로 오라클과 연동

스쿱이란 ?

오라클과 hive와의 데이터 연동 또는 오라클과 hdfs와의 데이터 연동을 위한 툴

오라클의 emp 테이블을 hive로 바로 로드 할 수 있다.

- 스쿱 설치

1. 스쿱설치 파일을 올린다.

알집 푸는 방법(unzip)

```
$ unzip ojdbc6.zip
```

sqoop-1.4.6.bin__hadoop-1.0.0 과 unzip ojdbc6.jar를 오라클에 올린다.

2. 스쿱설치 파일의 압축을 푼다.

```
[orcl:~]$ mv sqoop-1.4.6.bin__hadoop-1.0.0 sqoop <- 이름 바꾸기
```

```
[orcl:~]$ cp /home/oracle/ojdbc6.jar /home/oracle/sqoop/lib
```

```
[orcl:~]$ tar xvfz sqoop-1.4.6.bin__hadoop-1.0.0.tar.gz
```

3. 스쿱 디렉토리의 bin 디렉토리로 가서 스쿱을 실행한다.

```
[orcl:~]$ cd sqoop
```

```
[orcl:sqoop]$ cd bin
```

4. 오라클과 jdbc와 연동

```
[orcl:bin]$ ./sqoop
```

```
Warning: /home/oracle/sqoop/bin/../../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /home/oracle/sqoop/bin/../../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/oracle/sqoop/bin/../../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/oracle/sqoop/bin/../../zookeeper does not exist! Accumulo imports will fail.
```

경고가 4개가 뜨면 정상

5. .bash_profile에 sqoop 홈디렉토리를 지정한다.

```
[orcl:~]$ vi .bash_profile
```

아래의 두줄 추가

```
-----
export SQOOP_HOME=/home/oracle/sqoop
export PATH=$SQOOP_HOME/bin:$PATH
-----
```

```
[orcl:~]$ . .bash_profile    실행
```

6. 오라클의 emp테이블을 hive로 로드한다.

```
[orcl:bin]$ vi dept_import.sh
```

- 오라클과 hive 연동 스크립 사용방법 총정리

목표 : Oracle -----> Hive 로 import

```
hive> drop table dept;
hive> drop table dept;
OK
```

```
[orcl:~]$ vi table_import.sh
```

아래 내용을 붙여 넣는다.

```
-----
#!/bin/bash
```

```
oracle_table=`echo $3 | tr '[a-z]' '[A-Z]'`
hadoop_table=`echo $3 | tr '[A-Z]' '[a-z]'`
```

```
sqoop import --username $1 ₩
--password $2 ₩
--connect jdbc:oracle:thin:@localhost:1521:orcl ₩
--table $oracle_table ₩
--hive-import ₩
--hive-table $hadoop_table ₩
```

--hive-overwrite W

-m 1

[orcl:~]\$ sh table_import.sh scott tiger DEPT

실행을 하고,

```
18/07/05 14:57:59 INFO hive.HiveImport: OK
18/07/05 14:57:59 INFO hive.HiveImport: Time taken: 0.426 seconds
18/07/05 14:57:59 INFO hive.HiveImport: Hive import complete.
```

[orcl:~]\$ hive

hive> select * from dept;

```
10.0    ACCOUNTING    NEW YORK
20.0    RESEARCH        DALLAS
30.0    SALES             CHICAGO
40.0    OPERATIONS        BOSTON
```

스콧으로 오라클의 큰 테이블을 Hive로 로드

문제75. hr 계정의 employess 테이블을 hive로 로드하시오.

답)

[orcl:~]\$ sqlplus hr/hr

만약 hr 계정이 비밀번호가 달라 안되면

SQL> alter user hr account unlock;

SQL> alter user hr identified by hr;

SQL> connect hr/hr

을 해주면 된다.

SQL> select * from employees;

```
107 rows selected.
```

[orcl:~]\$ sh table_import.sh hr hr EMPLOYEES

```
18/07/05 15:54:39 INFO hive.HiveImport: OK
18/07/05 15:54:39 INFO hive.HiveImport: Time taken: 0.421 seconds
18/07/05 15:54:39 INFO hive.HiveImport: Hive import complete.
```

hive> select * from employees;

```
Time taken: 3.882 seconds, Fetched: 107 row(s)
```

문제76. 오라클에서 hive로 데이터를 로드하는 쉘을 생성하시오.

보기)

\$ sh hadoop_manage.sh

유저명을 입력하세요 : scott

패스워드를 입력하세요 : tiger

테이블 명을 입력하세요 : emp

답)

```
[orcl:~]$ vi hadoop_manage.sh
read -p "유저명을 입력하세요~" username
read -p "패스워드를 입력하세요~" password
read -p "테이블명을 입력하세요~" table

sh table_import.sh $username $password $table
```

결과)

```
-----
18/07/05 16:33:51 INFO hive.HiveImport: OK
18/07/05 16:33:51 INFO hive.HiveImport: Time taken: 0.399 seconds
18/07/05 16:33:51 INFO hive.HiveImport: Hive import complete.
[orcl:~]$ █
```

문제77. 위에서 만든 hadoop_manage.sh를 자동화 스크립트에 추가하시오.

답)

```
$ vi m2.sh
echo "
    1. Sar 그래프
    2. csv 파일 생성
    3. 파일의 특정 단어 수 찾기
    4. swp파일 모두 삭제
    5. 두개 파일의 차이 찾기
    6. SQL의 테이블 csv로 생성
    7. 하둡 파일 시스템을 중단 시키려면 7번을
    8. 하둡 파일 시스템을 시작 시키려면 8번을
    9. 하둡 파일 시스템의 상태를 확인하려면 9번을
    10. RunJar 프로세스를 죽이려면 10번을
    11. 스쿱을 활용해서 오라클에서 Hive로 데이터를 로드하려면 11번을
"
```

```
echo " "
```

```
echo -n "번호를 입력하세요"
```

```
read choice
```

```
case $choice in
```

```
    1)
```

```
        sh /home/oracle/Sar.sh ;;
```

```
    2)
```

```
        sh /home/oracle/find_file.sh ;;
```

```
    3)
```

```
        sh /home/oracle/fine_word.sh ;;
```



```

4)
    sh /home/oracle/rmswp.sh ;;
5)
    sh /home/oracle/diff.sh ;;
6)
    sh /home/oracle/hr_sh ;;
7)
    sh stop-all.sh ;;
8)
    sh start-all.sh ;;
9)
    jps ;;
10)
    sh r.sh ;;
11)
    sh hadoop_manage.sh ;;
esac

```

문제78. hive에서 oracle로 데이터를 로드하는 스크립트를 생성하시오.

일단 d.zip파일을 압축을 풀면 d.sh 스크립트를 /home/oracle에 올려놓는다.

hive에서 dept가있는지 확인

```
hive> select * from dept;
```

```
[orcl:~]$ sqlplus scott/tiger
```

```
SQL> drop table dept cascade constraints; <- 키 제약이 걸려있는 dept를 없애주고
```

```
SQL> drop table dept cascade constraints;
Table dropped.
```

```
SQL> create table dept
( deptno number(2),
  dname varchar2(14),
  loc varchar2(13) );
```

```
Table created.
```

```
hive> show create table dept;
```

새로운 테이블을 만들어주고

```
[orcl:~]$ vi table_export.sh
```

```
#!/bin/bash
```

```
oracle_table=`echo $3 | tr 'a-z' '[A-Z]'`
```

```
sqoop export --username $1 ₩
```

```
--password $2 ₩
```

```
--connect jdbc:oracle:thin:@localhost:1521:orcl ₩
```

```
--table $oracle_table ₩
```

```
--export-dir /user/hive/warehouse/$3 ₩
```

```
--input-fields-terminated-by '₩001' ₩
```

<- ₩뒤에 공백있으면 오류남 주의!!!

```
-m 1
```

```
-----  
[orcl:~]$ sh table_export.sh scott tiger dept
```

```
[orcl:~]$ hadoop fs -lsr /user
```

```
drwxr-xr-x - oracle supergroup 0 2018-07-03 15:26 /user/hive  
drwxr-xr-x - oracle supergroup 0 2018-07-05 16:35 /user/hive/warehouse  
drwxr-xr-x - oracle supergroup 0 2018-07-05 14:57 /user/hive/warehouse/dept  
-rw-r--r-- 3 oracle supergroup 0 2018-07-05 14:57 /user/hive/warehouse/dept/_SUCCESS  
-rw-r--r-- 3 oracle supergroup 80 2018-07-05 14:57 /user/hive/warehouse/dept/part-m-0000  
drwxr-xr-x - oracle supergroup 0 2018-07-05 16:33 /user/hive/warehouse/emp
```

이 파일이 있으면 된다.

문제79. 자동화 스크립트 12번에 hive에서 oracle로 데이터 이행하는 스크립트를 추가하시오.

답)

```
echo "
```

1. Sar 그래프
2. csv 파일 생성
3. 파일의 특정 단어 수 찾기
4. swp파일 모두 삭제
5. 두개 파일의 차이 찾기
6. SQL의 테이블 csv로 생성
7. 하둡 파일 시스템을 중단 시키려면 7번을
8. 하둡 파일 시스템을 시작 시키려면 8번을
9. 하둡 파일 시스템의 상태를 확인하려면 9번을
10. RunJar 프로세스를 죽이려면 10번을
11. 스쿱을 활용해서 Oracle에서 hiv로 데이터 로드하려면 11번을
12. 스쿱을 활용해서 hive에서 Oracle로 데이터 로드하려면 12번을

```
"
```

```
echo " "
```

```
echo -n "번호를 입력하세요"
```

```
read choice
```

```

case $choice in
    1)
        sh Sar.sh ;;
    2)
        sh find_file.sh ;;
    3)
        sh fine_word.sh ;;
    4)
        sh rmswp.sh ;;
    5)
        sh diff.sh ;;
    6)
        sh hr_.sh ;;
    7)
        sh stop-all.sh ;;
    8)
        start-all.sh ;;
    9)
        jps ;;
    10)
        sh oracle/r.sh ;;
    11)
        sh hadoop_manage.sh ;;
    12)
        sh table_export.sh ;;
esac

```

10. Mongo DB 설치

Mongo DB는 우분투에서 설치하는게 좋다.

1. MongoDB Pulbic GPG Key 등록

```

oracle@oracle-virtual-machine:~$ sudo apt-key adv --keyserver
hkp://keyserver.ubuntu.com:80 --recv EA312927
password : oracle (각자의 패스워드 oracle or ubuntu)

```

2. MongoDB 를 위한 list file 생성

(자신의 Ubuntu 버전에 맞게 입력하세요. 우리는 14.04)

```

oracle@oracle-virtual-machine:~$ echo "deb http://repo.mongodb.org/apt/ubuntu
trusty/mongodb-org/3.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list

```

```
oracle@oracle-virtual-machine:~$ echo "deb http://repo.mongodb.org/apt/ubuntu
es.list.d/mongodb-org-3.2.list
deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.2 multiverse"
```

3. apt-get 을 이용하여 설치

```
oracle@oracle-virtual-machine:~$ sudo apt-get update
내려받기 7,375 k바이트, 소요시간 31초 (234 k바이트/초)
패키지 목록을 읽는 중입니다... 완료
```

- latest stable version 설치

```
oracle@oracle-virtual-machine:~$ sudo apt-get install -y mongodb-org
'mongodb' 사용자를 'mongodb' 그룹에 추가 중...
사용자 mongodb을 (들) mongodb 그룹에 등록 중
완료.
```

4. 서버 실행

```
oracle@oracle-virtual-machine:~$ sudo service mongod start
start: Job is already running: mongod
```

서버가 제대로 실행됐는지 확인

```
oracle@oracle-virtual-machine:~$ cat /var/log/mongodb/mongod.log
```

5. MongoDB 서버 접속

```
oracle@oracle-virtual-machine:~$ mongo
MongoDB shell version: 3.2.20
connecting to: test
```

[mogodb에 테이블 생성](#)

문제80. mongodb에 emp테이블을 생성하시오.

답)

```
db.emp.save({empno:7499,ename:"SMITH",job:"CLERK",mgr:7902,hiredate:"1980-12-17",sal:1800,
comm:800,deptno:20})
db.emp.save({empno:7369,ename:"ALLEN",job:"SALESMAN",mgr:7698,hiredate:"1981-02-20",sal:1
600,comm:800,deptno:20})
db.emp.save({empno:7521,ename:"WARD",job:"SALESMAN",mgr:7698,hiredate:"1981-02-22",sal:1
250,comm:500,deptno:30})
db.emp.save({empno:7566,ename:"JONES",job:"MANAGER",mgr:7839,
hiredate:"1981-04-02",sal:2975,comm:0,deptno:20})
db.emp.save({empno:7654,ename:"MARTIN",job:"SALESMAN",mgr:7698,hiredate:"1981-09-28",sal
:1250,comm:1400,deptno:30})
db.emp.save({empno:7698,ename:"BLAKE",job:"MANAGER",mgr:7839,hiredate:"1981-05-01",sal:2
850,comm:0,deptno:30})
```

```

db.emp.save({empno:7782,ename:"CLARK",job:"MANAGER",mgr:7839,hiredate:"1981-06-09",sal:2450,comm:0,deptno:10})
db.emp.save({empno:7788,ename:"SCOTT",job:"ANALYST",mgr:7566,hiredate:"1987-04-19",sal:3000,comm:0,deptno:20})
db.emp.save({empno:7839,ename:"KING",job:"PRESIDENT",mgr:0,hiredate:"1981-11-17",sal:5000,comm:0,deptno:10})
db.emp.save({empno:7844,ename:"TURNER",job:"SALESMAN",mgr:7698,hiredate:"1981-09-08",sal:1500,comm:0,deptno:30})
db.emp.save({empno:7876,ename:"ADAMS",job:"CLERK",mgr:7788,hiredate:"1987-05-23",sal:1100,comm:0,deptno:20})
db.emp.save({empno:7900,ename:"JAMES",job:"CLERK",mgr:7698,hiredate:"1981-12-03",sal:950,comm:0,deptno:30})
db.emp.save({empno:7902,ename:"FORD",job:"ANALYST",mgr:7566,hiredate:"1981-12-03",sal:3500,comm:0,deptno:20})
db.emp.save({empno:7934,ename:"MILLER",job:"CLERK",mgr:7782,hiredate:"1982-01-23",sal:1300,comm:0,deptno:20})

```

- 테이블 삭제 하는 방법 : **db.emp.drop()**

문제81. 아래의 select문을 mongodb로 구현하시오.

보기)

SQL> select * from emp;

답)

```

> db.emp.aggregate([{$group:{_id:null, count:{$sum:1}}}]])
> db.emp.aggregate([{$group:{_id:null, count:{$sum:1}}}]])
{ "_id" : null, "count" : 14 }

```

문제82. 부서번호가 10번인 사원의 사원번호와 이름과 월급을 조회하시오.

답)

```

> db.emp.find({deptno:{$all:[10]}}, {_id:0, empno:1,ename:1,sal:1})

```

```

{ "empno" : 7782, "ename" : "CLARK", "sal" : 2450 }
{ "empno" : 7839, "ename" : "KING", "sal" : 5000 }

```

설명: id:0을 지우면 이렇게나온다.

```

{ "_id" : ObjectId("5b3ec19e9dddb3fdace8f885"), "empno" : 7782, "ename" : "CLARK", "sal" : 2450 }
{ "_id" : ObjectId("5b3ec19e9dddb3fdace8f887"), "empno" : 7839, "ename" : "KING", "sal" : 5000 }

```

문제83. 월급이 3000인 사원의 이름과 월급과 직업을 출력하시오.

답)

```

> db.emp.find({sal:{$all:[3000]}}, {_id:0, empno:1,ename:1,sal:1})

```

```

{ "empno" : 7788, "ename" : "SCOTT", "sal" : 3000 }

```

문제84. 월급이 2000이상인 직원들의 이름과 월급과 직업을 출력하시오.

답)

```
> db.emp.find({sal:{$gte:2000}}, {_id:0, empno:1,ename:1,sal:1})
```

```
{ "empno" : 7566, "ename" : "JONES", "sal" : 2975 }
{ "empno" : 7698, "ename" : "BLAKE", "sal" : 2850 }
{ "empno" : 7782, "ename" : "CLARK", "sal" : 2450 }
{ "empno" : 7788, "ename" : "SCOTT", "sal" : 3000 }
{ "empno" : 7839, "ename" : "KING", "sal" : 5000 }
{ "empno" : 7902, "ename" : "FORD", "sal" : 3500 }
```

Mogodb연산자

비교연산자

operator	설명
\$eq	(equals) 주어진 값과 일치하는 값
\$gt	(greater than) 주어진 값보다 큰 값
\$gte	(greather than or equals) 주어진 값보다 크거나 같은 값
\$lt	(less than) 주어진 값보다 작은 값
\$lte	(less than or equals) 주어진 값보다 작거나 같은 값
\$ne	(not equal) 주어진 값과 일치하지 않는 값
\$in	주어진 배열 안에 속하는 값
\$nin	주어진 배열 안에 속하지 않는 값

\$cmp : 두개의 값을 비교하여 첫 번째 값이 두번째 값보다 작으면 음수, 크면 양수,같으면 0을 리턴한다.

ex)

SQL

```
select ename,sal,job
  from emp
 where sal > 2000;
```

Mongo DB

```
db.emp.find(
  {sal : {$gt:2000}}, {_id:0,
  empno:1,ename:1,sal:1})
```

논리 연산자

operator	설명
\$or	주어진 조건중 하나라도 true 일 때 true
\$and	주어진 모든 조건이 true 일 때 true
\$not	주어진 조건이 false 일 때 true
\$nor	주어진 모든 조건이 false 일때 true

산술, 문자 연산자

2) 변경 연산자

종류	유형	설명
산술 연산자	\$add	두 개의 값을 합산한 결과를 리턴 합니다.
	\$divide	두 개의 값을 나눈 결과를 리턴 합니다.
	\$mod	첫 번째 값을 두 번째 값으로 나눈 후 나머지 값을 리턴 합니다.
	\$multiply	첫 번째 값과 두 번째 값을 곱한 결과를 리턴 합니다.
	\$subtract	첫 번째 값에서 두 번째 값을 뺀 결과를 리턴 합니다.
문자 연산자	\$strcasecmp	Long 타입의 긴 문자열 2개를 비교하여 첫 번째 문자열이 두 번째 문자열보다 크면 양수 값을 리턴 하고 작으면 0 값을 리턴 합니다.
	\$substr	해당 문자열에서 첫 번째 정의된 숫자 만큼을 Skip하고 두 번째 정의된 숫자 만큼의 길이 데이터를 사용자에게 리턴 합니다.
	\$toUpper	해당 문자열의 값을 소문자로 변환합니다.
	\$toLower	해당 문자열의 값을 대문자로 변환합니다.

문제85. 직업이 SALESMAN이 아닌 직원들의 이름과 직업을 출력하시오.

답)

```
> db.emp.find({job:{$ne:'SALESMAN'}}, {_id:0,ename:1,job:1})
```

```
{ "ename" : "SMITH", "job" : "CLERK" }
{ "ename" : "JONES", "job" : "MANAGER" }
{ "ename" : "BLAKE", "job" : "MANAGER" }
{ "ename" : "CLARK", "job" : "MANAGER" }
{ "ename" : "SCOTT", "job" : "ANALYST" }
{ "ename" : "KING", "job" : "PRESIDENT" }
{ "ename" : "ADAMS", "job" : "CLERK" }
{ "ename" : "JAMES", "job" : "CLERK" }
{ "ename" : "FORD", "job" : "ANALYST" }
{ "ename" : "MILLER", "job" : "CLERK" }
```

문제86. 월급이 3000이하인 직원들의 이름과 월급을 출력하시오.

답)

```
> db.emp.find({sal:{$lte:3000}}, {_id:0,ename:1,sal:1})
```

```
{ "ename" : "SMITH", "sal" : 1800 }
{ "ename" : "ALLEN", "sal" : 1600 }
{ "ename" : "WARD", "sal" : 1250 }
{ "ename" : "JONES", "sal" : 2975 }
{ "ename" : "MARTIN", "sal" : 1250 }
{ "ename" : "BLAKE", "sal" : 2850 }
{ "ename" : "CLARK", "sal" : 2450 }
{ "ename" : "SCOTT", "sal" : 3000 }
{ "ename" : "TURNER", "sal" : 1500 }
{ "ename" : "ADAMS", "sal" : 1100 }
{ "ename" : "JAMES", "sal" : 950 }
{ "ename" : "MILLER", "sal" : 1300 }
```

문제87. 이름, 월급을 출력하는데 월급이 높은 직원부터 출력하시오.

답)

```
> db.emp.find({}, {_id:0,ename:1,sal:1}).sort({sal:-1})
```

```
{ "ename" : "KING", "sal" : 5000 }
{ "ename" : "FORD", "sal" : 3500 }
{ "ename" : "SCOTT", "sal" : 3000 }
{ "ename" : "JONES", "sal" : 2975 }
{ "ename" : "BLAKE", "sal" : 2850 }
{ "ename" : "CLARK", "sal" : 2450 }
{ "ename" : "SMITH", "sal" : 1800 }
{ "ename" : "ALLEN", "sal" : 1600 }
{ "ename" : "TURNER", "sal" : 1500 }
{ "ename" : "MILLER", "sal" : 1300 }
{ "ename" : "WARD", "sal" : 1250 }
{ "ename" : "MARTIN", "sal" : 1250 }
{ "ename" : "ADAMS", "sal" : 1100 }
{ "ename" : "JAMES", "sal" : 950 }
```

설명 : dp.emp.find({조건}, {출력할 내용}, {정렬})

문제88. 직업이 SALESMAN인 직원들의 이름과 월급과 직업을 출력하는데 월급이 낮은 직원부터 출력하시오.

답)

```
> db.emp.find({job:{ $eq:'SALESMAN'}}, {_id:0,ename:1, sal:1, job:1}).sort({sal:1})
```

```
{ "ename" : "WARD", "job" : "SALESMAN", "sal" : 1250 }
{ "ename" : "MARTIN", "job" : "SALESMAN", "sal" : 1250 }
{ "ename" : "TURNER", "job" : "SALESMAN", "sal" : 1500 }
{ "ename" : "ALLEN", "job" : "SALESMAN", "sal" : 1600 }
```

문제89. 월급이 1000 에서 3000 사이인 직원들의 이름과 월급을 출력하시오.

답)

```
> db.emp.find( { $and:[{ sal:{ $gte:1000} }, { sal:{ $lte:3000} } ] })
```

문제90. 직원번호가 7788 or 7902인 직원들의 직원번호와 이름과 월급을 출력하시오.

답)

```
> db.emp.find( { $or: [{empno:7788}, {empno:7902}] }, {_id:0,empno:1, ename:1, sal:1} )
```

문제91. 부서번호를 출력하는데 중복제거해서 출력하시오.

답)

```
db.emp.distinct("deptno")
```

```
> db.emp.distinct("deptno")
[ 20, 30, 10 ]
```

문제92. 직원테이블의 전체 건수를 출력하시오.

답)

```
> db.emp.count()
```

```
14
```

문제93. 직업이 SALESMAN인 직원들의 인원수를 출력하시오.

답)

```
> db.emp.count( {job:'SALESMAN'} )
```


문제94. 월급이 3000이상인 직원들의 인원수를 출력하시오.

답)

```
> db.emp.count( { sal:{$gte:3000} } )
```

3

문제95. 직원 테이블 전체에서 최대 월급을 출력하세요.

답)

```
> db.emp.find({}, {_id:0, sal:1}).sort({sal:-1}).limit(1)
```

```
{ "sal" : 5000 }
```

설명)

sort로 정렬한후 limit으로 1개만 뽑는다.

문제96. 직업이 SALESMAN인 직원들의 최대 월급을 출력하시오.

답)

```
> db.emp.find({job:'SALESMAN'}, {_id:0, sal:1}).sort({sal:-1}).limit(1)
```

```
{ "sal" : 1600 }
```

tip) db.emp.find 에서 . 이 의미하는게 마침표? 라고 보면된다.

예를 들어 코드가 길어지면

db.

emp.

find

이런식으로 해보면

```
> db.
... emp.
... find
```

다음줄에 이어서 쓸 수 있다.

문제97. 직원 테이블의 토털월급을 출력하시오.

답)

```
> db.emp.aggregate( [ {$group: {_id:null, total:{$sum:"$sal"}} } ] )
```

```
{ "_id" : null, "total" : 30525 }
```

문제98. 직업이 SALESMAN인 직원들의 토털월급을 출력하시오.

답)

```
> db.emp.aggregate( [ {$match: {job:'SALESMAN'}}, {$group: {_id:null, total:{$sum:"$sal"}} } ] )
```

```
{ "_id" : null, "total" : 5600 }
```

문제99. (점심시간) 부서번호가 30번인 직원들의 최대월급을 출력하시오.

답1)

```
> db.emp.find( { deptno:30 }, {_id:0, sal:1}).sort({sal:-1}).limit(1)
```

```
{ "sal" : 2850 }
```

답2)

```
> db.emp.aggregate( [ { $match: {deptno:30}}, { $group: { _id: null, maxsal: { $max: "$sal" } } } ] )
```

```
{ "_id" : null, "maxsal" : 2850 }
```

문제100. 부서번호, 부서번호별 토탈월급을 출력하시오.

답)

```
> db.emp.aggregate( [ { $group: { _id: "$deptno", total: { $sum: "$sal" } } } ] )
```

```
{ "_id" : 10, "total" : 7450 }
{ "_id" : 30, "total" : 7800 }
{ "_id" : 20, "total" : 15275 }
```

문제101. 직업, 직업별 최대 월급을 출력하시오.

답)

```
> db.emp.aggregate( [ { $group: { _id: "$job", maxsal: { $max: "$sal" } } } ] )
```

```
{ "_id" : "PRESIDENT", "maxsal" : 5000 }
{ "_id" : "ANALYST", "maxsal" : 3500 }
{ "_id" : "MANAGER", "maxsal" : 2975 }
{ "_id" : "SALESMAN", "maxsal" : 1600 }
{ "_id" : "CLERK", "maxsal" : 1800 }
```

문제102. 직업, 직업별 토탈월급을 출력하는데 직업이 SALESMAN은 제외하고 출력하시오.

답)

```
> db.emp.aggregate( [ { $match: { job: { $ne: "SALESMAN" } } }, { $group: { _id: "$job",
sumsal: { $sum: "$sal" } } } ] )
```

```
{ "_id" : "PRESIDENT", "sumsal" : 5000 }
{ "_id" : "ANALYST", "sumsal" : 6500 }
{ "_id" : "MANAGER", "sumsal" : 8275 }
{ "_id" : "CLERK", "sumsal" : 5150 }
```

문제103. 직업, 직업별 토탈월급을 출력하는데 직업이 SALESMAN은 제외하고, 직업별 토탈월급이 높은것 부터 출력하시오.

답)

```
db.emp.aggregate([ { $match : { job : { $ne: 'SALESMAN' } } }, { "$group": { "_id": '$job', "total":
{"$sum":"$sal"} } }, { $sort: { "total": -1 } } ] )
```

```
{ "_id" : "MANAGER", "total" : 8275 }
{ "_id" : "ANALYST", "total" : 6500 }
{ "_id" : "CLERK", "total" : 5150 }
{ "_id" : "PRESIDENT", "total" : 5000 }
```

- [오라클의 기타 비교 연산자와 mogodb의 비교](#)

SQL	Mongo DB
1. between .. and	문제89
2. like	문제104
3. is null	

문제104. 이름의 첫 번째 철자가 A로 시작하는 직원들의 이름, 월급을 출력하시오.

답)

```
> db.emp.find( { _id:0, name:/^A/ }, { _id:0, name:1, sal:1 } )
```

```
{ "name" : "ALLEN", "sal" : 1600 }
{ "name" : "ADAMS", "sal" : 1100 }
```

문제105. 이름의 끝 글자가 T로 끝나는 직원들의 이름, 월급을 출력하시오.

답)

```
> db.emp.find( { _id:0, name:/T$/ }, { _id:0, name:1, sal:1 } )
```

```
{ "name" : "SCOTT", "sal" : 3000 }
```

문제106. 이름에 철자 M을 포함하는 직원들의 이름과 월급을 출력하시오.

답)

```
> db.emp.find( { _id:0, name:/M/ }, { _id:0, name:1, sal:1 } )
```

```
{ "name" : "SMITH", "sal" : 1800 }
{ "name" : "MARTIN", "sal" : 1250 }
{ "name" : "ADAMS", "sal" : 1100 }
{ "name" : "JAMES", "sal" : 950 }
{ "name" : "MILLER", "sal" : 1300 }
```

문제107. 이름의 두번째 철자가 M인 직원들의 이름과 월급을 출력하시오.

답)

```
> db.emp.find( { _id:0, name:/^..M/ }, { _id:0, name:1, sal:1 } )
```

```
{ "name" : "SMITH", "sal" : 1800 }
```

Mongodb에서의 조인문장

문제108. emp테이블 생성 스크립트를 참고해서 dept 테이블을 생성하시오.

답)

```
db.dept.save({deptno:10, dname:"ACCOUNTING", loc:"NEWYORK"})
```

```
db.dept.save({deptno:20, dname:"RESEARCH", loc:"DALLAS"})
```

```
db.dept.save({deptno:30, dname:"SALES", loc:"CHICAGO"})
```

```
db.dept.save({deptno:40, dname:"OPERATIONS", loc:"BOSTON"})
```

join

문제109. 이름과 부서위치를 출력하시오.

답)

호진이형 소오스)

```
db.emp.aggregate([
```

```
{
```

```
  $lookup:
```

```

    {
      from: "dept",
      localField: "deptno",
      foreignField: "deptno",
      as: "aa"
    }
  },
  { $project : { _id: 0,
    ename:1,
    aa:{loc:1}
  }
}
])

```

```

{ "ename" : "SMITH", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "ALLEN", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "WARD", "aa" : [ { "loc" : "CHICAGO" } ] }
{ "ename" : "JONES", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "MARTIN", "aa" : [ { "loc" : "CHICAGO" } ] }
{ "ename" : "BLAKE", "aa" : [ { "loc" : "CHICAGO" } ] }
{ "ename" : "CLARK", "aa" : [ { "loc" : "NEWYORK" } ] }
{ "ename" : "SCOTT", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "KING", "aa" : [ { "loc" : "NEWYORK" } ] }
{ "ename" : "TURNER", "aa" : [ { "loc" : "CHICAGO" } ] }
{ "ename" : "ADAMS", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "JAMES", "aa" : [ { "loc" : "CHICAGO" } ] }
{ "ename" : "FORD", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "MILLER", "aa" : [ { "loc" : "DALLAS" } ] }

```

문제110. 이름과 월급과 부서명을 출력하시오.

답)

```

db.emp.aggregate([
  {
    $lookup:
    {
      from: "dept",
      localField: "deptno",
      foreignField: "deptno",
      as: "aa"
    }
  },
  { $project : { _id: 0,
    ename:1,
    sal:1,
    aa:{dname:1}
  }
}

```

```

    }
  })
}

{ "ename" : "SMITH", "sal" : 1800, "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "ALLEN", "sal" : 1600, "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "WARD", "sal" : 1250, "aa" : [ { "dname" : "SALES" } ] }
{ "ename" : "JONES", "sal" : 2975, "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "MARTIN", "sal" : 1250, "aa" : [ { "dname" : "SALES" } ] }
{ "ename" : "BLAKE", "sal" : 2850, "aa" : [ { "dname" : "SALES" } ] }
{ "ename" : "CLARK", "sal" : 2450, "aa" : [ { "dname" : "ACCOUNTING" } ] }
{ "ename" : "SCOTT", "sal" : 3000, "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "KING", "sal" : 5000, "aa" : [ { "dname" : "ACCOUNTING" } ] }
{ "ename" : "TURNER", "sal" : 1500, "aa" : [ { "dname" : "SALES" } ] }
{ "ename" : "ADAMS", "sal" : 1100, "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "JAMES", "sal" : 950, "aa" : [ { "dname" : "SALES" } ] }
{ "ename" : "FORD", "sal" : 3500, "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "MILLER", "sal" : 1300, "aa" : [ { "dname" : "RESEARCH" } ] }

```

문제111. 위의 다시 출력하는데 월급이 3000이상인 직원들만 출력하시오.

답)

```

db.emp.aggregate( [
  {$lookup:{from:"dept", localField:"deptno", foreignField:"deptno", as:"aa"}},
  {$project:{_id:0, ename:1, sal:1, aa:{dname:1}}},
  {$match:{sal:{$gte:3000}}}
] )

```

```

{ "ename" : "SCOTT", "sal" : 3000, "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "KING", "sal" : 5000, "aa" : [ { "dname" : "ACCOUNTING" } ] }
{ "ename" : "FORD", "sal" : 3500, "aa" : [ { "dname" : "RESEARCH" } ] }

```

Mongodb 에서의 DML문

1. update

문제112. SCOTT의 월급을 5600으로 변경하시오.

답)

```

> db.emp.update( { ename:"SCOTT"}, { $set : {sal:5600} }, {multi:true} )

```

```

    "sal" : 2450, "comm" : 0, "deptno" : 10 }
    id" : ObjectId("5b3ef7b3ac4035b3ffff1572"), "empno" : 7788, "ename" : "SCOTT",
    "sal" : 5600, "comm" : 0, "deptno" : 20 }

```

multi:true = 여러 개 변경하도록

문제113. 월급이 3000이상인 직원들의 comm을 7000으로 수정하시오.

답)

```

> db.emp.update( { sal:{$gte:3000} }, {$set :{comm:7000} }, {multi:true} )

```

```

WriteResult({ "nMatched" : 3, "nUpserted" : 0, "nModified" : 3 })

```

```
{ "_id" : ObjectId("5b3ef7b3ac4035b3
19", "sal" : 5600, "comm" : 7000, "d
{ "_id" : ObjectId("5b3ef7b3ac4035b3
", "sal" : 5000, "comm" : 7000, "dep
{ "_id" : ObjectId("5b3ef7b3ac4035b3
9-08", "sal" : 1500, "comm" : 0, "de
{ "_id" : ObjectId("5b3ef7b3ac4035b3
", "sal" : 1100, "comm" : 0, "deptno
{ "_id" : ObjectId("5b3ef7b3ac4035b3
", "sal" : 950, "comm" : 0, "deptno"
{ "_id" : ObjectId("5b3ef7b3ac4035b3
3", "sal" : 3500, "comm" : 7000, "de
{ "_id" : ObjectId("5b3ef7b3ac4035b3
3", "sal" : 1300, "comm" : 0, "deptn
```

2. remove

문제114. 직업이 ANALYST인 직원들을 삭제하시오.

답)

```
> db.emp.remove( { job:"ANALYST" } )
```

```
WriteResult({ "nRemoved" : 2 })
```

```
"", "job" : "CLERK", "ma
", "job" : "SALESMAN",
", "job" : "SALESMAN",
6", "job" : "MANAGER",
IN", "job" : "SALESMAN",
", "job" : "MANAGER",
", "job" : "MANAGER",
", "job" : "PRESIDENT",
ER", "job" : "SALESMAN",
5", "job" : "CLERK", "ma
5", "job" : "CLERK", "ma
ER", "job" : "CLERK", "f
```

사라져서 안보임

문제115. emp테이블을 내용 전체를 지우시오.

답)

```
db.emp.remove({})
```

```
WriteResult({ "nRemoved" : 12 })
```

3. insert

문제116. 아래의 insert를 mongodb로 구현하시오.

보기)

```
SQL> insert into emp (empno, enmae, sal)
```

```
values(7788,'SCOTT',3000);
```

답)

```
> db.emp.insert( { empno:7788, enmae:"SCOTT", sal:3000 } )
```

```
WriteResult({ "nInserted" : 1 })
```

```
> db.emp.find()  
{ "_id" : ObjectId("5b3f0f12ac4035b3ffff1581"), "empno" : 7788, "enmae" : "SCOTT", "sal" : 3000 }
```

MongoDB에서 DDL 문

1. drop

문제117. 아래의 SQL명령어를 몽고디비로 구현하시오.

보기)

```
SQL> drop table emp;
```

답)

```
> db.emp.drop()
```

```
> db.emp.drop()  
true
```

2. alter : 컬럼 추가/삭제는 안되는데 컬럼 이름변경은 가능하다.

문제118. emp테이블을 다시 생성하고, 컬럼명 sal로 salary로 변경 하시오.

답)

```
db.emp.save({empno:7499,ename:"SMITH",job:"CLERK",mgr:7902,hiredate:"1980-12-17",sal:1800,  
comm:800,deptno:20})
```

```
db.emp.save({empno:7369,ename:"ALLEN",job:"SALESMAN",mgr:7698,hiredate:"1981-02-20",sal:1  
600,comm:800,deptno:20})
```

```
db.emp.save({empno:7521,ename:"WARD",job:"SALESMAN",mgr:7698,hiredate:"1981-02-22",sal:1  
250,comm:500,deptno:30})
```

```
db.emp.save({empno:7566,ename:"JONES",job:"MANAGER",mgr:7839,  
hiredate:"1981-04-02",sal:2975,comm:0,deptno:20})
```

```
db.emp.save({empno:7654,ename:"MARTIN",job:"SALESMAN",mgr:7698,hiredate:"1981-09-28",sal:  
:1250,comm:1400,deptno:30})
```

```
db.emp.save({empno:7698,ename:"BLAKE",job:"MANAGER",mgr:7839,hiredate:"1981-05-01",sal:2  
850,comm:0,deptno:30})
```

```
db.emp.save({empno:7782,ename:"CLARK",job:"MANAGER",mgr:7839,hiredate:"1981-06-09",sal:2  
450,comm:0,deptno:10})
```

```
db.emp.save({empno:7788,ename:"SCOTT",job:"ANALYST",mgr:7566,hiredate:"1987-04-19",sal:30  
00,comm:0,deptno:20})
```

```
db.emp.save({empno:7839,ename:"KING",job:"PRESIDENT",mgr:0,hiredate:"1981-11-17",sal:5000,c
```

```

omm:0,deptno:10))
db.emp.save({empno:7844,ename:"TURNER",job:"SALESMAN",mgr:7698,hiredate:"1981-09-08",sal:1500,comm:0,deptno:30})
db.emp.save({empno:7876,ename:"ADAMS",job:"CLERK",mgr:7788,hiredate:"1987-05-23",sal:1100,comm:0,deptno:20})
db.emp.save({empno:7900,ename:"JAMES",job:"CLERK",mgr:7698,hiredate:"1981-12-03",sal:950,comm:0,deptno:30})
db.emp.save({empno:7902,ename:"FORD",job:"ANALYST",mgr:7566,hiredate:"1981-12-03",sal:3500,comm:0,deptno:20})
db.emp.save({empno:7934,ename:"MILLER",job:"CLERK",mgr:7782,hiredate:"1982-01-23",sal:1300,comm:0,deptno:20})

```

```
db.emp.update({}, { $rename: { "sal": "salary" } })
```

이렇게하면 첫 줄만 변경되는데

```
> db.emp.update({}, { $rename: { "sal": "salary" } }, {multi:true} )
```

을 써서 나머지도 변경해준다.

```

> db.emp.update({}, { $rename: { "sal": "salary" } }, {multi:true} )
WriteResult({ "nMatched" : 14, "nUpserted" : 0, "nModified" : 13 })

```

문제119. emp테이블에 email컬럼을 추가하시오.

답)

```
db.emp.update({},{$set : {"emailk":' '}}, false,true)
```

```

> db.emp.update({},{$set : {"emailk":' '}}, false,true)
WriteResult({ "nMatched" : 14, "nUpserted" : 0, "nModified" : 14 })

```

설명:

```
db.emp.update({},{$set : {"emailk":' '}}, {upsert:false,multi:true})
```

upsert:false --> insert와 update를 한번에 수행 하는데 upsert

문제120. email 컬럼을 삭제하시오.

답)

```
db.emp.update({},{$unset : {emailk:1}}, {multi:true})
```

```

> db.emp.update({},{$unset : {emailk:1}}, {multi:true})
WriteResult({ "nMatched" : 14, "nUpserted" : 0, "nModified" : 14 })

```

• MongoDB에서 index 생성하는 방법

: index란? 대용량 데이터 베이스 환경에서 데이터의 검색 속도를 향상 시켜주는 db object

문제121. emp 테이블에 월급을 검색할 때 속도를 높일수 있는 인덱스를 생성하시오.

보기)

```
SQL> create index emp_sal
      on emp(sal);
```

답)


```
> db.emp.find( {salary:3000} )
```

조금... 빨라진건가?

문제122. 아래의 index를 mongodb에서 생성하시오.

보기)

```
SQL> create index emp_deptno_sal  
      on emp( deptno, sal desc );
```

```
SQL> select deptno, sal  
      from emp  
      where deptno > 0;
```

답)

```
> db.emp.ensureIndex( { deptno:1, sal:-1} )
```

문제123. 아래의 SQL을 mogodb에서 검색하시오.

보기)

```
SQL> select ename, sal, deptno  
      from emp  
      where deptno = 30 and sal = 1600;
```

답)

```
> db.emp.find( {$and:[ { deptno:20 }, {salary:1600} ] }, {_id:0, ename:1, salary:1, deptno:1})
```

```
{ "ename" : "ALLEN", "deptno" : 20, "salary" : 1600 }
```

문제124 (마지막) 부서위치, 부서위치별 토탈월급을 출력하시오.

보기)

```
SQL> select d.loc, sum(e.sal)  
      from emp e, dept d  
      where e.deptno = d.deptno  
      group by d.loc;
```

답1)

```
db.emp.aggregate( [ { $lookup:{ from: "dept", localField: "deptno", foreignField: "deptno", as:  
"aa" } },  
  { $project : { _id: 0, ename:1, aa:{loc:1}, salary:1 } },  
  { $group: { _id:"$aa", sumsal:{ $sum:"$salary" } } } ] )
```

```
{ "_id" : [ { "loc" : "NEWYORK" } ], "sumsal" : 7450 }  
{ "_id" : [ { "loc" : "CHICAGO" } ], "sumsal" : 7800 }  
{ "_id" : [ { "loc" : "DALLAS" } ], "sumsal" : 15275 }
```

답2)

```
db.emp.aggregate( [ { $lookup:{ from: "dept", localField: "deptno", foreignField: "deptno", as:
```

```
"aa" } },
{ $group: { _id: "$aa.loc", sumsal: { $sum: "$salary" } } } ] )
```

```
{ "_id" : [ "NEWYORK" ], "sumsal" : 7450 }
{ "_id" : [ "CHICAGO" ], "sumsal" : 7800 }
{ "_id" : [ "DALLAS" ], "sumsal" : 15275 }
```

11. 하둡과 R 연동하는 방법

1. 먼저 R 을 실행한다.(root로 들어가야한다)

```
[orcl:bin]$ su -
[root@edydr1p0 ~]# [orcl:~]$ cd /home/oracle/R-3.2.3
-bash: [orcl:~]$: command not found
[root@edydr1p0 ~]# cd /home/oracle/R-3.2.3
[root@edydr1p0 R-3.2.3]# cd bin
[root@edydr1p0 bin]# ./R
```

2. 하둡을 이용할수 있도록 R 에서 환경설정

```
Sys.setenv (JAVA_HOME="/u01/app/java/jdk1.7.0_60")
Sys.setenv (HADOOP_CMD="/u01/app/hadoop/hadoop-1.2.1/bin/hadoop")
```

3. rhdfs 패키지를 다운로드 받는다.

<https://github.com/RevolutionAnalytics/RHadoop/wiki/Downloads>

rhdfs_1.0.8.tar.gz 다운로드 파일을 아래의 위치에 둔다.

```
> install.packages("/home/oracle/R-3.2.3/library/rhdfs_1.0.8.tar.gz", repos=NULL, type="source")
```

4. R 에서 아래와 같이 설치한다.

```
> install.packages("/home/oracle/rhdfs_1.0.8.tar.gz", repos=NULL, type="source")
> install.packages("/home/oracle/rhdfs_1.0.8.tar.gz", repos=NULL, type="source")
* installing *source* package 'rhdfs' ...
** R
** inst
** preparing package for lazy loading
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded
* DONE (rhdfs)
```

5. 아래와 같이 library 하고 hdf5.init() 를 불러온다.

```
> library(rhdfs)
```

```

> library(rhdfs)
Loading required package: rJava

HADOOP_CMD=/u01/app/hadoop/hadoop-1.2.1/bin/hadoop

Be sure to run hdfs.init()

> hdfs.init()
> hdfs.init()
#
# A fatal error has been detected by the Java Runtime Environment:
#
# Internal Error (threadLocalStorage.cpp:56), pid=12434, tid=308660
# guarantee(get_thread() == thread) failed: must be the same thread
#
# JRE version: (7.0_60-b19) (build )
# Java VM: Java HotSpot(TM) Server VM (24.60-b09 mixed mode linux-x86_64)
# Failed to write core dump. Core dumps have been disabled. To enable
#
# An error report file with more information is saved as:
# /home/oracle/hs_err_pid12434.log
#
# If you would like to submit a bug report, please visit:
# http://bugreport.sun.com/bugreport/crash.jsp
#
Aborted

```

6. 하둡 파일 시스템에 emp.csv 를 올리고 하둡 R 에서 읽어온다.

```
$ hadoop fs -put emp2.csv emp2.csv
```

```
> hdfs.ls("/user/oracle/emp.csv")
```

```
> tmp <- hdfs.cat("/user/oracle/emp.csv")
```

```
> emp <- read.csv(textConnection(tmp),head=T)
```

7. 이름과 월급을 조회하시오 !

```
> emp[emp$ename=='SCOTT',c("ename","sal")]
```

이렇게하면 아마도 emp테이블내용에 빈칸이 들어가있어서

```
> emp[grep('SCOTT',emp$ENAME),]
```

이렇게 하면된다.

```

> emp[grep('SCOTT',emp$ENAME),]
  EMPNO      ENAME      JOB MGR HIREDATE   SAL COMM DEPTNO
8  7788 SCOTT      ANALYST  7566 19-APR-87 3000   NA     20

```

빈칸이 없는 테이블을 가져오는 방법

```
> tmp <- hdfs.cat("/user/oracle/emp2.csv") emp2를 가져온다.
```

```
> emp <- read.csv(textConnection(tmp),head=F)
```

```
> colnames(emp) <- c("empno", "ename", "job", "mgr", "hiredate", "sal", "comm", "deptno")
```

12. 자바를 이용해서 하둡 파일 시스템의 데이터를 select 하기

- 하둡의 주용한 2가지 기능?

1. HDFS -----> java

2. MapReduce -----> java

2-1. map 기능 (함수)

2-2. Reduce 기능 (함수)

ex)

원본문서	map함수	reduce 함수
감자는 가지과의 여러해살이 풀이다.	-----> 감자 1 가지 1	-----> 감자 3 가지 1
안텍스 산맥 일대가 감자의 원산지 이다.	-----> 안텍스 산맥1 감자 1 원산지 1	-----> 안텍스 산맥 1 원산지 1 기후 1
감자는 서늘한 기후를 좋아한다.	-----> 감자 1 기후 1	

reduce : map함수에서 뽑은 걸 다 합침

```
select job, count(*)  
from emp  
group by job;
```

1. hadoop 홈디렉토리에 자바 실행 파일인 jar 파일의 위치가 어디인지 설정하는 환경설정을 하는 부분

```
[orcl:~]$ cat >> .bash_profile << EOF
```

```
> export CLASSPATH=.:$HADOOP_HOME/hadoop-ant-1.2.1.jar:$HADOOP_HOME/hadoop-  
client-1.2.1.jar:$HADOOP_HOME/hadoop-core-1.2.1.jar:$HADOOP_HOME/hadoop-  
examples-1.2.1.jar:$HADOOP_HOME/hadoop-minicluster-1.2.1.jar:$HADOOP_HOME/hadoop-  
test-1.2.1.jar:$HADOOP_HOME/hadoop-tools-1.2.1.jar:$HADOOP_HOME/lib/asm-3.2.jar:  
$HADOOP_HOME/lib/aspectjrt-1.6.11.jar:$HADOOP_HOME/lib/aspectjtools-1.6.11.jar:  
$HADOOP_HOME/lib/commons-beanutils-1.7.0.jar:$HADOOP_HOME/lib/commons-beanutils-  
core-1.8.0.jar:$HADOOP_HOME/lib/commons-cli-1.2.jar:$HADOOP_HOME/lib/commons-  
codec-1.4.jar:$HADOOP_HOME/lib/commons-collections-3.2.1.jar:  
$HADOOP_HOME/lib/commons-configuration-1.6.jar:$HADOOP_HOME/lib/commons-  
daemon-1.0.1.jar:$HADOOP_HOME/lib/commons-digester-1.8.jar:  
$HADOOP_HOME/lib/commons-el-1.0.jar:$HADOOP_HOME/lib/commons-httpclient-3.0.1.jar:
```

```

$HADOOP_HOME/lib/commons-io-2.1.jar:$HADOOP_HOME/lib/commons-lang-2.4.jar:
$HADOOP_HOME/lib/commons-logging-1.1.1.jar:$HADOOP_HOME/lib/commons-logging-
api-1.0.4.jar:$HADOOP_HOME/lib/commons-math-2.1.jar:$HADOOP_HOME/lib/commons-
net-3.1.jar:$HADOOP_HOME/lib/core-3.1.1.jar:$HADOOP_HOME/lib/hadoop-capacity-
scheduler-1.2.1.jar:$HADOOP_HOME/lib/hadoop-fairscheduler-1.2.1.jar:
$HADOOP_HOME/lib/hadoop-thriftfs-1.2.1.jar:$HADOOP_HOME/lib/hsqldb-1.8.0.10.jar:
$HADOOP_HOME/lib/jackson-core-asl-1.8.8.jar:$HADOOP_HOME/lib/jackson-mapper-
asl-1.8.8.jar:$HADOOP_HOME/lib/jasper-compiler-5.5.12.jar:$HADOOP_HOME/lib/jasper-
runtime-5.5.12.jar:$HADOOP_HOME/lib/jdeb-0.8.jar:$HADOOP_HOME/lib/jersey-core-1.8.jar:
$HADOOP_HOME/lib/jersey-json-1.8.jar:$HADOOP_HOME/lib/jersey-server-1.8.jar:
$HADOOP_HOME/lib/jets3t-0.6.1.jar:$HADOOP_HOME/lib/jetty-6.1.26.jar:
$HADOOP_HOME/lib/jetty-util-6.1.26.jar:$HADOOP_HOME/lib/jsch-0.1.42.jar:
$HADOOP_HOME/lib/junit-4.5.jar:$HADOOP_HOME/lib/kfs-0.2.2.jar:
$HADOOP_HOME/lib/log4j-1.2.15.jar:$HADOOP_HOME/lib/mockito-all-1.8.5.jar:
$HADOOP_HOME/lib/oro-2.0.8.jar:$HADOOP_HOME/lib/servlet-api-2.5-20081211.jar:
$HADOOP_HOME/lib/slf4j-api-1.4.3.jar:$HADOOP_HOME/lib/slf4j-log4j12-1.4.3.jar:
$HADOOP_HOME/lib/xmlenc-0.52.jar:$CLASSPATH
> EOF

```

```
[orcl:~]$ source .bash_profile
```

2. 하둡 홈디렉토리 밑에 labs 라는 디렉토리를 만들고 거기에 SingleFileWriteRead.java파일을 생성한다.

```

[orcl:~]$ cd $HADOOP_HOME
[orcl:hadoop-1.2.1]$ mkdir labs
[orcl:hadoop-1.2.1]$ cd labs

```

```
[orcl:labs]$ vi SingleFileWriteRead.java
```

```

=====
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

public class SingleFileWriteRead {
    public static void main(String[] args) {
        if (args.length != 2) {

```

```

        System.err.println("Usage: SingleFileWriteRead <filename> <contents>");
        System.exit(2);
    }

    try {
        Configuration conf = new Configuration();
        conf.set("fs.default.name", "hdfs://localhost:9000");
        FileSystem hdfs = FileSystem.get(conf);

        Path path = new Path(args[0]);
        if (hdfs.exists(path)) {
            hdfs.delete(path, true);
        }

        FSDataOutputStream outputStream = hdfs.create(path);
        outputStream.writeUTF(args[1]);
        outputStream.close();

        FSDataInputStream inputStream = hdfs.open(path);
        String inputString = inputStream.readUTF();
        inputStream.close();

        System.out.println("## inputString:" + inputString);

    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
=====

```

3. java 파일을 컴파일해서 class 파일을 생성 해야한다.

```
[orcl:labs]$ javac SingleFileWriteRead.java
```

4. first.txt를 하둡 파일 시스템에 만든다.

```
[orcl:labs]$ vi first.txt
```

```

-----
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
cccccccccccccccccccccccccccccccccccccccc
-----

```

5. 하둡 파일 시스템에 SingleFileWriteRead 클래스 파일을 이용해서 first.txt를 올린다.

```
[orcl:labs]$ hadoop -cp $CLASSPATH:. SingleFileWriteRead first.txt firstText
[orcl:labs]$ hadoop -cp $CLASSPATH:. SingleFileWriteRead first.txt firstText
## inputString:firstText
```

```
[orcl:labs]$ hadoop -cp $CLASSPATH:. SingleFileWriteRead second.txt "secondText thirdText"
-----
[orcl:labs]$ hadoop -cp $CLASSPATH:. :
## inputString:secondText thirdText
[orcl:labs]$
```

```
[orcl:labs]$ hadoop fs -ls
Found 6 items
-rw-r--r--  3 oracle supergroup      84 2018-07-05 14:49 /user/oracle/dept2.csv
-rw-r--r--  3 oracle supergroup    1815 2018-07-09 14:16 /user/oracle/emp.csv
-rw-r--r--  3 oracle supergroup     644 2018-07-04 10:17 /user/oracle/emp2.csv
-rw-r--r--  3 oracle supergroup      11 2018-07-09 15:46 /user/oracle/first.txt
-rw-r--r--  3 oracle supergroup      22 2018-07-09 15:46 /user/oracle/second.txt
-rw-r--r--  3 oracle supergroup   114548 2018-07-03 16:27 /user/oracle/winter.txt
[orcl:labs]$
```

```
[oracle@edydr1p2 labs]$ hadoop fs -rm first.txt second.txt
Deleted hdfs://localhost:9000/user/oracle/first.txt
Deleted hdfs://localhost:9000/user/oracle/second.txt
```

13. 두개의 파일을 하나로 합쳐서 하둡 파일 시스템에 올리는 실습

```
[orcl:labs]$ vi PutMerge.java
```

```
-----
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

public class PutMerge {

    public static void main(String[] args) throws IOException {

        Configuration conf = new Configuration();
        conf.set("fs.default.name", "hdfs://localhost:9000");

        FileSystem hdfs = FileSystem.get(conf);
        FileSystem local = FileSystem.getLocal(conf);
```

```

Path inputDir = new Path(args[0]);
Path hdfsFile = new Path(args[1]);

try {
    FileStatus[] inputFiles = local.listStatus(inputDir);
    FSDataOutputStream out = hdfs.create(hdfsFile);

    for (int i=0; i<inputFiles.length; i++) {
        System.out.println(inputFiles[i].getPath().getName());
        FSDataInputStream in = local.open(inputFiles[i].getPath());
        byte buffer[] = new byte[256];
        int bytesRead = 0;
        while( (bytesRead = in.read(buffer)) > 0) {
            out.write(buffer, 0, bytesRead);
        }
        in.close();
    }
    out.close();
} catch (IOException e) {
    e.printStackTrace();
}
}
}

```

```

[orcl:labs]$ javac PutMerge.java
[orcl:labs]$ mkdir inputtext
[orcl:labs]$ cp /etc/hosts inputtext/    #cp : classpass
[orcl:labs]$ cp /etc/group inputtext/
[orcl:labs]$ ls -l inputtext/
total 8
-rw-r--r-- 1 oracle oinstall 783 Jul  9 16:12 group
-rw-r--r-- 1 oracle oinstall 253 Jul  9 16:12 hosts

[orcl:labs]$ hadoop -cp $CLASSPATH:. PutMerge inputtext result.txt
hosts
group

[orcl:labs]$ hadoop fs -ls

```



```
Found 5 items
-rw-r--r--  3 oracle supergroup      84 2018-07-05 14:49 /user/oracle/dept2.csv
-rw-r--r--  3 oracle supergroup    1815 2018-07-09 14:16 /user/oracle/emp.csv
-rw-r--r--  3 oracle supergroup     644 2018-07-04 10:17 /user/oracle/emp2.csv
-rw-r--r--  3 oracle supergroup    1036 2018-07-09 16:16 /user/oracle/result.txt
-rw-r--r--  3 oracle supergroup   114548 2018-07-03 16:27 /user/oracle/winter.txt
```

```
[orcl:labs]$ hadoop fs -cat result.txt
```

```
[orcl:labs]$ hadoop fs -cat result.txt
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1      edydr1p0.us.oracle.com edydr1p0 localhost.localdomain localhost
10.0.2.128    edydr1p0.us.oracle.com edydr1p0 localhost.localdomain localhost
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon,lp
mem:x:8:
kmem:x:9:
wheel:x:10:root
mail:x:12:mail
news:x:13:news
uucp:x:14:uucp
man:x:15:
games:x:20:
gopher:x:30:
dip:x:40:
ftp:x:50:
```

```
[orcl:labs]$ rm -rf inputtext/
```

14. WordCount 예제

1. 하둡 파일 시스템에 input이라는 디렉토리를 만든다.

```
[orcl:labs]$ hadoop fs -mkdir input
```

2. 하둡 파일 시스템에 input 디렉토리에 로컬에 etc 밑에 hosts 파일을 올린다.

또 하둡 홈 디렉토리 밑에 README.txt를 올린다.

```
[orcl:labs]$ hadoop fs -put /etc/hosts input
```

```
[orcl:labs]$ hadoop fs -put $HADOOP_HOME/README.txt input
```

```
[orcl:labs]$ hadoop fs -lsr
```

```
[orcl:labs]$ hadoop fs -lsr
-rw-r--r--  3 oracle supergroup      84 2018-07-05 14:49 /user/oracle/dept2.csv
-rw-r--r--  3 oracle supergroup    1815 2018-07-09 14:16 /user/oracle/emp.csv
-rw-r--r--  3 oracle supergroup     644 2018-07-04 10:17 /user/oracle/emp2.csv
drwxr-xr-x  - oracle supergroup      0 2018-07-09 16:23 /user/oracle/input
-rw-r--r--  3 oracle supergroup    1366 2018-07-09 16:23 /user/oracle/input/README.txt
-rw-r--r--  3 oracle supergroup     253 2018-07-09 16:23 /user/oracle/input/hosts
-rw-r--r--  3 oracle supergroup    1036 2018-07-09 16:16 /user/oracle/result.txt
-rw-r--r--  3 oracle supergroup   114548 2018-07-03 16:27 /user/oracle/winter.txt
```

3. 하둡 홈디렉토리에 기본적으로 내장 되어있는 워드 맵리듀스 함수를 이용해서
지금 올린 2개의 파일을 워드 카운트 한다.

```
[orcl:labs]$ hadoop jar $HADOOP_HOME/hadoop-examples-*.jar wordcount input/hosts
```

output1

```
18/07/09 16:24:11 INFO input.FileInputFormat: Total input paths to process : 1
18/07/09 16:24:11 INFO util.NativeCodeLoader: Loaded the native-hadoop library
18/07/09 16:24:11 WARN snappy.LoadSnappy: Snappy native library not loaded
18/07/09 16:24:11 INFO mapred.JobClient: Running job: job_201807031417_0049
18/07/09 16:24:12 INFO mapred.JobClient:  map 0% reduce 0%
18/07/09 16:24:17 INFO mapred.JobClient:  map 100% reduce 0%
18/07/09 16:24:24 INFO mapred.JobClient:  map 100% reduce 100%

----- Map -----
18/07/09 16:24:26 INFO mapred.JobClient:  Reduce input groups=22
18/07/09 16:24:26 INFO mapred.JobClient:  Combine output records=22
18/07/09 16:24:26 INFO mapred.JobClient:  Physical memory (bytes) snapshot=172224512
18/07/09 16:24:26 INFO mapred.JobClient:  Reduce output records=22
18/07/09 16:24:26 INFO mapred.JobClient:  Virtual memory (bytes) snapshot=785133568
18/07/09 16:24:26 INFO mapred.JobClient:  Map output records=27
```

```
[orcl:labs]$ hadoop fs -lsr
```

```
-rw-r--r--  3 oracle supergroup      84 2018-07-05 14:49 /user/oracle/dept2.csv
-rw-r--r--  3 oracle supergroup    1815 2018-07-09 14:16 /user/oracle/emp.csv
-rw-r--r--  3 oracle supergroup     644 2018-07-04 10:17 /user/oracle/emp2.csv
drwxr-xr-x  - oracle supergroup      0 2018-07-09 16:23 /user/oracle/input
-rw-r--r--  3 oracle supergroup    1366 2018-07-09 16:23 /user/oracle/input/README.txt
-rw-r--r--  3 oracle supergroup     253 2018-07-09 16:23 /user/oracle/input/hosts
drwxr-xr-x  - oracle supergroup      0 2018-07-09 16:24 /user/oracle/output1
-rw-r--r--  3 oracle supergroup      0 2018-07-09 16:24 /user/oracle/output1/_SUCCESS
drwxr-xr-x  - oracle supergroup      0 2018-07-09 16:24 /user/oracle/output1/_logs
drwxr-xr-x  - oracle supergroup      0 2018-07-09 16:24 /user/oracle/output1/_logs/history
-rw-r--r--  3 oracle supergroup    13908 2018-07-09 16:24 /user/oracle/output1/_logs/history,
_word+count
-rw-r--r--  3 oracle supergroup    54759 2018-07-09 16:24 /user/oracle/output1/_logs/history,
-rw-r--r--  3 oracle supergroup      231 2018-07-09 16:24 /user/oracle/output1/part-r-00000
-rw-r--r--  3 oracle supergroup     1036 2018-07-09 16:16 /user/oracle/result.txt
-rw-r--r--  3 oracle supergroup   114548 2018-07-03 16:27 /user/oracle/winter.txt
[orcl:labs]$
```

\$확인

```
[orcl:labs]$ hadoop fs -cat output1/part-r-00000
```

```

#          2
10.0.2.128    1
127.0.0.1     1
Do           1
edydr1p0      2
edydr1p0.us.oracle.com  2
fail.         1
following      1
functionality  1
line,         1
localhost     2
localhost.localdomain  2
network       1
not           1
or            1
programs      1
remove        1
require       1
that          1
the           1
various       1
will          1

```

```

[orcl:labs]$ hadoop fs -rmr output*
Deleted hdfs://localhost:9000/user/oracle/output1

```

```

[orcl:labs]$ hadoop fs -rmr input
Deleted hdfs://localhost:9000/user/oracle/input

```

문제125. 겨울왕국 대본 winter.txt를 하둡 파일 시스템에 올리고 하둡의 맵리듀스 함수로 워드 카운트 하시오.

답)

1. 하둡 파일 시스템에 input이라는 디렉토리를 만든다.

```

[oracle@edydr1p2 labs]$ hadoop fs -mkdir input5

```

```

[orcl:labs]$ hadoop fs -put winter.txt input5

```

3. 하둡 홈디렉토리에 기본적으로 내장 되어있는 워드 맵리듀스 함수를 이용해서 지금 올린 1개의 파일을 워드 카운트 한다.

```

[orcl:labs]$ hadoop jar $HADOOP_HOME/hadoop-examples-*.jar wordcount input5/winter.txt
output4

```

확인

```

hadoop fs -cat output4/part-r-00000

```

```

hadoop fs -get output4/part-r-00000 /home/oracle/result3.txt

```

로컬로 내린 result1.txt 를 건수가 가장 높은 것부터 정렬해서 출력하시오.

```
[orcl:~]$ sort -nrk 2 result3.txt | head -50
```

MySQL in CentOS 설치

<https://shas15.github.io/install-centos-mysql/>

에서 보고 설치

oracle과 MySQL의 비교

문제126. 직업이 SALESMAN 인 직원들의 이름, 월급, 직업을 출력하는데 월급이 높은 직원부터 출력.
답)

```
mysql> select ename, sal, job  
-> from emp  
-> where job='salesman'  
-> order by sal desc;
```

```
mysql> select ename, sal, job  
+-----+-----+-----+  
| ename | sal | job |  
+-----+-----+-----+  
| ALLEN | 1600 | SALESMAN |  
| TURNER | 1500 | SALESMAN |  
| MARTIN | 1250 | SALESMAN |  
| WARD | 1250 | SALESMAN |  
+-----+-----+-----+
```

- 오라클과 다르게 salesman을 소문자로 써도 된다.

ifnull

문제127. 이름과 커미션을 출력하는데 커미션이 null인 직원들은 0으로 출력하시오.

답)

```
mysql> select ename, ifnull(comm,0) from emp;
```

```
+-----+-----+  
| ename | ifnull(comm,0) |  
+-----+-----+  
| KING | 0 |  
| BLAKE | 0 |  
| CLARK | 0 |  
| JONES | 0 |  
| MARTIN | 1400 |  
| ALLEN | 300 |  
| TURNER | 0 |  
| JAMES | 0 |  
| WARD | 500 |  
| FORD | 0 |  
| SMITH | 0 |  
| SCOTT | 0 |  
| ADAMS | 0 |  
| MILLER | 0 |  
+-----+-----+  
14 rows in set (0.01 sec)
```

sysdate()

문제128. 오늘 날짜를 출력하시오.

답)

```
mysql> select sysdate() from dual;
```

```
+-----+
| sysdate() |
+-----+
| 2018-07-10 14:05:22 |
+-----+
1 row in set (0.00 sec)
```

to_days

문제129. 이름, 입사한 날짜부터 오늘까지 총 몇일 근무했는지 출력하시오.

답)

```
mysql> select ename, to_days(sysdate())-to_days(hiredate) from emp;
```

```
+-----+-----+
| ename | to_days(sysdate())-to_days(hiredate) |
+-----+-----+
| KING | 13384 |
| BLAKE | 13584 |
| CLARK | 13576 |
| JONES | 13614 |
| MARTIN | 13452 |
| ALLEN | 13663 |
| TURNER | 13472 |
| JAMES | 13360 |
| WARD | 13651 |
| FORD | 13360 |
| SMITH | 13727 |
| SCOTT | 12984 |
| ADAMS | 12960 |
| MILLER | 13329 |
+-----+-----+
14 rows in set (0.03 sec)
```

period_diff(date_format)

문제130. 이름, 입사한 날짜부터 오늘까지 총 몇 달 근무했는지 출력하시오.

답)

```
mysql> select ename,
```

```
period_diff(date_format(now(), '%Y%m'), date_format(hiredate, '%Y%m')) as months
```

```
from emp;
```

```
+-----+-----+
| ename | months |
+-----+-----+
| KING | 440 |
| BLAKE | 446 |
| CLARK | 446 |
| JONES | 447 |
| MARTIN | 442 |
| ALLEN | 449 |
| TURNER | 443 |
| JAMES | 439 |
| WARD | 449 |
| FORD | 439 |
| SMITH | 451 |
| SCOTT | 427 |
| ADAMS | 426 |
| MILLER | 438 |
+-----+-----+
```

문제131. 오늘부터 100달 뒤에 돌아오는 날짜가 어떻게 되는가?

답)

```
mysql> select period_add(date_format(sysdate(),'%Y-%m'), 100);
```

```
+-----+
| period_add(date_format(sysdate()),'%Y-%m'), 100) |
+-----+
| 202910 |
+-----+
```

```
mysql> select sysdate() + interval 100 month;
```

```
+-----+
| sysdate() + interval 100 month |
+-----+
| 2026-11-10 14:16:09 |
+-----+
```

문제132. 오늘부터 이번달 말일까지 총 몇일 남았는지 출력하시오.

답)

```
mysql> select to_days(last_day( sysdate())) - to_days(sysdate()) from dual;
```

```
+-----+
| to_days(last_day( sysdate())) - to_days(sysdate()) |
+-----+
| 21 |
+-----+
1 row in set (0.00 sec)
```

문제133. 오늘이 무슨 요일인지 출력하시오.

답)

```
mysql> select date_format(sysdate(),'%W');
```

```
+-----+
| date_format(sysdate(),'%W') |
+-----+
| Tuesday |
+-----+
```

문제134. 이름, 입사일, 입사한 요일을 출력하시오.

답)

```
mysql> select ename, hiredate, date_format(hiredate,'%W') from emp;
```

```
+-----+-----+-----+
| ename | hiredate | date_format(hiredate,'%W') |
+-----+-----+-----+
| KING | 1981-11-17 | Tuesday |
| BLAKE | 1981-05-01 | Friday |
| CLARK | 1981-05-09 | Saturday |
| JONES | 1981-04-01 | Wednesday |
| MARTIN | 1981-09-10 | Thursday |
| ALLEN | 1981-02-11 | Wednesday |
| TURNER | 1981-08-21 | Friday |
| JAMES | 1981-12-11 | Friday |
| WARD | 1981-02-23 | Monday |
| FORD | 1981-12-11 | Friday |
| SMITH | 1980-12-09 | Tuesday |
| SCOTT | 1982-12-22 | Wednesday |
| ADAMS | 1983-01-15 | Saturday |
| MILLER | 1982-01-11 | Monday |
+-----+-----+-----+
```

tip) Y : 연도 4자리

y : 연도 2자리

D : 일

d : 일th

문제135. 이름, 월급 을 출력하는데 월급을 출력할때에 천 단위를 붙이시오.

답)

mysql> select ename, format(sal,0) from emp;

ename	format(sal,0)
KING	5,000
BLAKE	2,850
CLARK	2,450
JONES	2,975
MARTIN	1,250
ALLEN	1,600
TURNER	1,500
JAMES	950
WARD	1,250
FORD	3,000
SMITH	800
SCOTT	3,000
ADAMS	1,100
MILLER	1,300

14 rows in set (0.02 sec)

문제136. 이름과 커미션을 출력하는데 커미션을 출력할 때 커미션이 null인 직원들을 no comm이라고 출력하시오.

답)

SQL)

select ename, nvl(to_char(comm),'no comm') from emp;

mysql> select ename, ifnull(comm, 'no comm') from emp;

ename	ifnull(comm, 'no comm')
KING	no comm
BLAKE	no comm
CLARK	no comm
JONES	no comm
MARTIN	1400
ALLEN	300
TURNER	0
JAMES	no comm
WARD	500
FORD	no comm
SMITH	no comm
SCOTT	no comm
ADAMS	no comm
MILLER	no comm

문제137. 이름, 직업, 보너스를 출력하는데 직업이 SALESMAN이면 보너스를 6000을 출력하고, 아니면 0을 출력하시오.

답)

```
SQL> select ename, job, decode(job,'SALESMAN',6000,0) from emp;
```

```
mysql> select ename, job, if(job='salesman',6000,0) as bonus from emp;
```

ename	job	bonus
KING	PRESIDENT	0
BLAKE	MANAGER	0
CLARK	MANAGER	0
JONES	MANAGER	0
MARTIN	SALESMAN	6000
ALLEN	SALESMAN	6000
TURNER	SALESMAN	6000
JAMES	CLERK	0
WARD	SALESMAN	6000
FORD	ANALYST	0
SMITH	CLERK	0
SCOTT	ANALYST	0
ADAMS	CLERK	0
MILLER	CLERK	0

14 rows in set (0.00 sec)

문제138. 이름, 부서번호, 보너스를 출력하는데, 부서번호가 10번이면 보너스를 7000으로, 20번이면 9000으로, 30번이면 4000으로 출력하시오.

답1)

```
mysql> select ename, deptno, if(deptno=10,7000,if(deptno=20,9000,if(deptno=30,4000,0))) as bonus from emp;
```

ename	deptno	bonus
KING	10	7000
BLAKE	30	4000
CLARK	10	7000
JONES	20	9000
MARTIN	30	4000
ALLEN	30	4000
TURNER	30	4000
JAMES	30	4000
WARD	30	4000
FORD	20	9000
SMITH	20	9000
SCOTT	20	9000
ADAMS	20	9000
MILLER	10	7000

14 rows in set (0.00 sec)

답2)

```
mysql> select ename, deptno, case deptno when 10 then 7000 when 20 then 9000 else 4000 end as bonus from emp;
```


문제139. 아래와 같이 결과를 출력하시오.

보기)

```
10      20      30
-----
8750    10875    9400
```

답)

```
mysql> select sum(if(deptno=10,sal,null)) as "10", sum(if(deptno=20,sal,null)) as "20",
sum(if(deptno=30,sal,null)) as "30" from emp;
```

```
+-----+-----+-----+
| 10    | 20    | 30    |
+-----+-----+-----+
| 8750  | 10875 | 9400  |
+-----+-----+-----+
1 row in set (0.01 sec)
```

문제140. 아래와 같이 결과를 출력하시오.

보기)

```
SQL> select job, sum(decode( deptno,10,sal,null)) as "10",
      sum(decode( deptno,20,sal,null)) as "20",
      sum(decode( deptno,30,sal,null)) as "30"
      from emp
      group by job;
```

답)

```
mysql> select job, sum(if(deptno=10,sal,null)) as "10", sum(if(deptno=20,sal,null)) as "20",
sum(if(deptno=30,sal,null)) as "30" from emp group by job;
```

```
+-----+-----+-----+-----+
| job      | 10    | 20    | 30    |
+-----+-----+-----+-----+
| ANALYST  | NULL  | 6000  | NULL  |
| CLERK    | 1300  | 1900  | 950   |
| MANAGER  | 2450  | 2975  | 2850  |
| PRESIDENT | 5000  | NULL  | NULL  |
| SALESMAN | NULL  | NULL  | 5600  |
+-----+-----+-----+-----+
```

문제141. 아래의 SQL을 MySQL로 구현하시오.

보기)

```
select deptno, sum(sal)
      from emp
      group by rollup(deptno);
```

답)

```
mysql> select deptno, sum(sal) from emp group by deptno with rollup;
```

deptno	sum(sal)
10	8750
20	10875
30	9400
NULL	29025

문제142. 아래의 SQL을 MySQL로 구현하시오.

보기)

```
SQL> select deptno, sum(sal)
      from emp
      group by cube(deptno);
```

답)

**MySQL은 grouping sets와 cube를 지원하지 않는다.
rollup을 with rollup으로 지원한다.**

```
mysql> select null as deptno, sum(sal)
      from emp
      union all
      select deptno, sum(sal)
      from emp
      group by deptno;
```

deptno	sum(sal)
NULL	29025
10	8750
20	10875
30	9400

문제143. 이름, 입사일, 순위를 출력하는데 순위가 먼저 입사한 직원순으로 순위를 부여하시오.

보기)

```
SQL> select ename, hiredate, dense_rank() over(order by hiredate asc) 순위
      from emp;
```

답1)

```
mysql> select ename, hiredate, (select count(*) +1 from emp where hiredate < e.hiredate) rnk
      -> from emp e
      -> order by rnk asc;
```

ename	hiredate	rnk
SMITH	1980-12-09	1
ALLEN	1981-02-11	2
WARD	1981-02-23	3
JONES	1981-04-01	4
BLAKE	1981-05-01	5
CLARK	1981-05-09	6
TURNER	1981-08-21	7
MARTIN	1981-09-10	8
KING	1981-11-17	9
JAMES	1981-12-11	10
FORD	1981-12-11	10
MILLER	1982-01-11	12
SCOTT	1982-12-22	13
ADAMS	1983-01-15	14

14 rows in set (0.04 sec)

답2)

```
mysql> set @hiredate_rnk := 0;
```

```
mysql> select ename, hiredate, @hiredate_rnk := @hiredate_rnk +1 as rnk from emp,(select
@hiredate_rnk := 0) r order by hiredate;
```

ename	hiredate	rnk
SMITH	1980-12-09	1
ALLEN	1981-02-11	2
WARD	1981-02-23	3
JONES	1981-04-01	4
BLAKE	1981-05-01	5
CLARK	1981-05-09	6
TURNER	1981-08-21	7
MARTIN	1981-09-10	8
KING	1981-11-17	9
FORD	1981-12-11	10
JAMES	1981-12-11	11
MILLER	1982-01-11	12
SCOTT	1982-12-22	13
ADAMS	1983-01-15	14

문제144. 이름, 월급, 순위를 출력하시오

답)

```
mysql> select ename, sal, (select count(*) +1 from emp where sal > e.sal) rnk from emp e
order by rnk asc;
```

ename	sal	rnk
KING	5000	1
FORD	3000	2
SCOTT	3000	2
JONES	2975	4
BLAKE	2850	5
CLARK	2450	6
ALLEN	1600	7
TURNER	1500	8
MILLER	1300	9
WARD	1250	10
MARTIN	1250	10
ADAMS	1100	12
JAMES	950	13
SMITH	800	14

문제145. 부서번호, 이름, 월급, 순위를 출력하는데 순위가 부서번호별로 각각 월급이 높은 순서대로 순위를 출력하시오.

답)

```
mysql> select deptno, ename, sal, (select count(*) +1 from emp where sal > e.sal ) rnk
-> from emp e
-> order by deptno asc, rnk asc;
```

deptno	ename	sal	rnk
10	KING	5000	1
10	CLARK	2450	6
10	MILLER	1300	9
20	FORD	3000	2
20	SCOTT	3000	2
20	JONES	2975	4
20	ADAMS	1100	12
20	SMITH	800	14
30	BLAKE	2850	5
30	ALLEN	1600	7
30	TURNER	1500	8
30	WARD	1250	10
30	MARTIN	1250	10
30	JAMES	950	13

14 rows in set (0.01 sec)

문제146. 부서번호, 해당 부서번호에 속한 직원들의 이름을 가로로 출력하시오.

답)

```
mysql> select deptno, group_concat(ename order by ename asc separator ',') from emp group
by deptno;
```

deptno	group_concat(ename order by ename asc separator ',')
10	CLARK,KING,MILLER
20	ADAMS,FORD,JONES,SCOTT,SMITH
30	ALLEN,BLAKE,JAMES,MARTIN,TURNER,WARD

3 rows in set (0.01 sec)

join

문제147. 이름, 부서위치를 출력하시오.

답)

```
mysql> select e.ename, d.loc  
-> from emp e, dept d  
-> where e.deptno=d.deptno;
```

ename	loc
KING	NEW YORK
BLAKE	CHICAGO
CLARK	NEW YORK
JONES	DALLAS
MARTIN	CHICAGO
ALLEN	CHICAGO
TURNER	CHICAGO
JAMES	CHICAGO
WARD	CHICAGO
FORD	DALLAS
SMITH	DALLAS
SCOTT	DALLAS
ADAMS	DALLAS
MILLER	NEW YORK

문제148. 이름, 부서위치를 출력하는데 outer join도 되는지 확인하시오.

보기)

```
mysql> select e.ename, d.loc  
-> from emp e, dept d  
-> where e.deptno(+) = d.deptno;
```

안된다!!!!!!!

답)

```
mysql> select e.ename, d.loc from emp e right outer join dept d on(e.deptno=d.deptno);
```

ename	loc
KING	NEW YORK
BLAKE	CHICAGO
CLARK	NEW YORK
JONES	DALLAS
MARTIN	CHICAGO
ALLEN	CHICAGO
TURNER	CHICAGO
JAMES	CHICAGO
WARD	CHICAGO
FORD	DALLAS
SMITH	DALLAS
SCOTT	DALLAS
ADAMS	DALLAS
MILLER	NEW YORK
NULL	BOSTON

문제149. 아래의 full outer join을 MySQL에서 구현하시오.

보기)

```
SQL> select e.ename, d.loc
```

```
from emp e full outer join dept d
on (e.deptno=d.deptno);
```

답)

```
mysql> select e.ename, d.loc
-> from emp e left outer join dept d
-> on(e.deptno=d.deptno)
-> union
-> select e.ename, d.loc
-> from emp e right outer join dept d
-> on(e.deptno=d.deptno);
```

유니온 사용한 이유는 중복제거하려고

ename	loc
KING	NEW YORK
CLARK	NEW YORK
MILLER	NEW YORK
JONES	DALLAS
FORD	DALLAS
SMITH	DALLAS
SCOTT	DALLAS
ADAMS	DALLAS
BLAKE	CHICAGO
MARTIN	CHICAGO
ALLEN	CHICAGO
TURNER	CHICAGO
JAMES	CHICAGO
WARD	CHICAGO
NULL	BOSTON

15 rows in set (0.01 sec)

문제150. JONES의 월급보다 더 많은 월급을 받는 직원들의 이름과 월급을 출력하시오.

답)

```
mysql> select ename, sal
-> from emp
-> where sal > (select sal from emp where ename='jones');
```

ename	sal
KING	5000
FORD	3000
SCOTT	3000

15. MySQL에서 DML문(자동 커밋 되서 조심해야함!!)

문제151. SCOTT의 월급을 0 으로 변경하시오.

답)

```
mysql> update emp
set sal = 0
where ename='scott';
```

~~XXXXXX~~

```

set sal = 0
where ename='scott';

```

자동 커밋이 default가 켜있기 때문에 rollback이 안된다.

```
set autocommit=false;
```

```
select @@autocommit;
```

```

mysql> select @@autocommit;
+-----+
| @@autocommit |
+-----+
|           1 |
+-----+
1 row in set (0.03 sec)

mysql> set autocommit=false;
Query OK, 0 rows affected (0.00 sec)

mysql> select @@autocommit;
+-----+
| @@autocommit |
+-----+
|           0 |
+-----+
1 row in set (0.00 sec)

```

1 : 켜짐

0 : 꺼짐

문제152. 사원테이블을 삭제하고 rollback이 되는지 확인하시오.

답)

```

mysql> delete from emp;
Query OK, 14 rows affected (0.00 sec)

mysql> select * from emp;
Empty set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from emp;
+-----+-----+-----+-----+-----+-----+-----+-----+
| empno | ename  | job      | mgr  | hiredate | sal  | comm | deptno |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 7839  | KING   | PRESIDENT | NULL | 1981-11-17 | 5000 | NULL | 10      |
| 7698  | BLAKE  | MANAGER   | 7839 | 1981-05-01 | 2850 | NULL | 30      |
| 7782  | CLARK  | MANAGER   | 7839 | 1981-05-09 | 2450 | NULL | 10      |
| 7566  | JONES  | MANAGER   | 7839 | 1981-04-01 | 2975 | NULL | 20      |
| 7654  | MARTIN | SALESMAN  | 7698 | 1981-09-10 | 1250 | 1400 | 30      |
| 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  | NULL | 30      |
| 7521  | WARD   | SALESMAN  | 7698 | 1981-02-23 | 1250 | 500  | 30      |
| 7902  | FORD   | ANALYST   | 7566 | 1981-12-11 | 3000 | NULL | 20      |
| 7369  | SMITH  | CLERK     | 7902 | 1980-12-09 | 800  | NULL | 20      |
| 7788  | SCOTT  | ANALYST   | 7566 | 1982-12-22 | 3000 | NULL | 20      |
| 7876  | ADAMS  | CLERK     | 7788 | 1983-01-15 | 1100 | NULL | 20      |
| 7934  | MILLER | CLERK     | 7782 | 1982-01-11 | 1300 | NULL | 10      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

문제153. dept 테이블 스크립트를 이용해서 dept2테이블을 생성하고 dept2를 select 한 후에 rollback을 하면 dept2 테이블이 어떻게 되는지 확인해 보시오.

답)

```
CREATE TABLE dept2
(deptno int,
dname VARCHAR(14),
loc VARCHAR(13) );
```

```
INSERT INTO dept2 VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO dept2VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO dept2 VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO dept2 VALUES (40, 'OPERATIONS', 'BOSTON');
```

```
mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from dept2;
Empty set (0.00 sec)
```

테이블은 안사라지고 빈 테이블만 남는다.

MySQL에서 DDL문

문제154. salgrade 테이블을 생성하시오.

답)

```
create table salgrade
( grade int,
  losal int,
  hisal int );
```

```
insert into salgrade values(1,700,1200);
insert into salgrade values(2,1201,1400);
insert into salgrade values(3,1401,2000);
insert into salgrade values(4,2001,3000);
insert into salgrade values(5,3001,9999);
```

```
mysql> select * from salgrade;
```

grade	losal	hisal
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

5 rows in set (0.02 sec)

문제155. emp2.csv를 가지고 external 테이블을 생성하시오.

답)

일단 파일을 올리고

```
CREATE TABLE emp2 (  
  empno int NOT NULL,  
  ename VARCHAR(10),  
  job VARCHAR(9),  
  mgr int,  
  hiredate DATE,  
  sal int,  
  comm int,  
  deptno int );
```

load data local infile '/root/emp2.csv' into table emp2 fields terminated by ',';

select * from emp2;

empno	ename	job	mgr	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 in set (0.00 sec)

- 오라클과 다른점은 테이블이 실제로 생성된거라서 DML은 다 가능하고 인덱스도 생성된다.

문제156. price2.csv를 가지고 price 테이블을 생성하시오.

답)

price2.csv 파일은 utf8이아니라 txt로 불러와야한다.

```
mysql> alter database orcl default character set utf8 COLLATE utf8_general_ci;  
설정해주고
```

```
create table price  
(  
  P_SEQ int(10),  
  M_SEQ int(10),
```

```

M_NAME varchar(80),
A_SEQ int(10),
A_NAME varchar(60),
A_UNIT varchar(40),
A_PRICE int(10),
P_YEAR_MONTH varchar(30),
ADD_COL varchar(180),
M_TYPE_CODE varchar(20),
M_TYPE_NAME varchar(20),
M_GU_CODE varchar(10),
M_GU_NAME varchar(30) );

```

load data local infile '/root/price2.txt' into table price character set utf8 fields terminated by 'Wt';

378146	72	금남시장	315	"오징어(냉동, 국산)"	1마리
5000	Oct-12	생태			1
100000		성동구			
378147	72	금남시장	256	"고등어(30cm, 국산)"	1마리
2000	Oct-12				1
100000		성동구			
378148	72	금남시장	316	오이	1마리
3000	Oct-12				1
100000		성동구			