

MySQL 교안

강사 : 강병준

MySQL의 특징

1. SQL이란 무엇인가?

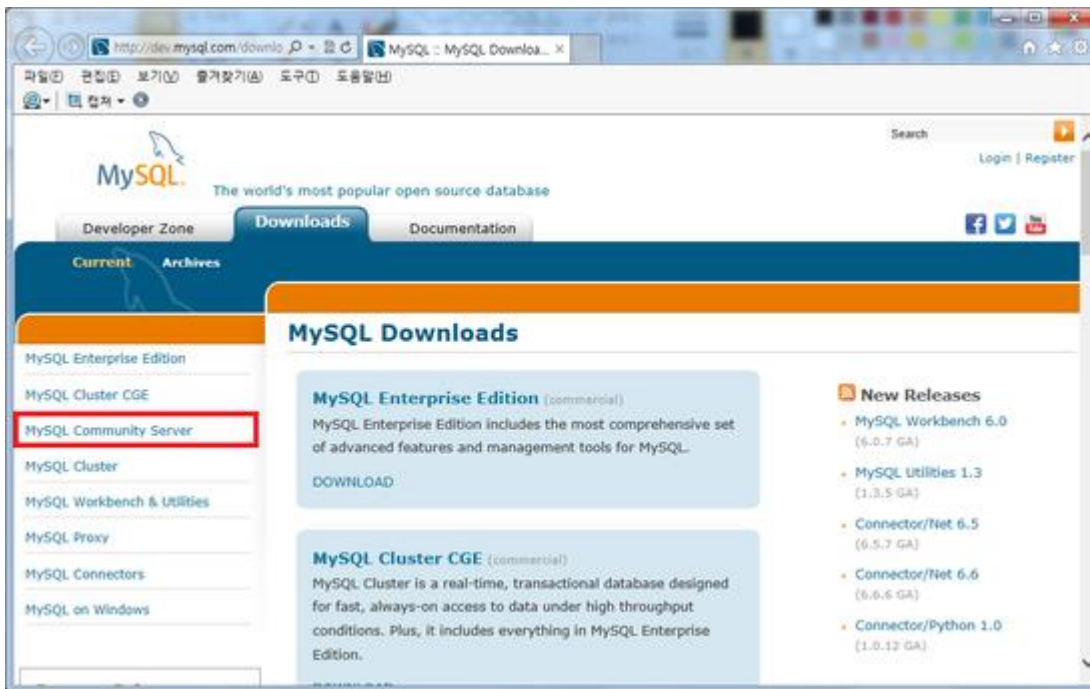
- 1) 비탐색적(NON NAVIGATIONAL)입니다. 이것은 SQL을 통하여 원하는 데이터를 DB에 알려주기만 하면 되므로, DB가 해당 데이터를 얻는 방법을 지정할 필요가 없습니다. 예를 들어, DB는 데이터를 검색하기 위하여 인덱스(INDEX)를 사용할지 여부를 결정합니다.
- 2) SQL은 **비절차적(NON-PROCEDURAL)** 언어입니다. 절차적 언어는 프로그래밍 언어입니다. SQL은 프로그래밍 언어가 아니므로 반복 수행하거나 확장 IF-THEN-ELSE 단계를 통하여 한 번에 하나의 레코드를 검색하지 않습니다. SQL은 테이블의 일련의 레코드(RECORD)들을 처리하며 해당 데이터에 대한 절차적 논리를 수행하는 프로그래밍 언어 안에 포함되어 해당 언어의 기능을 활용할 수 있습니다

2. MySQL의 특징

- 1) SQL에 기반을 둔 관계형 DBMS 중 하나
- 2) Oracle, IBM, Infomix 등의 데이터베이스는 고가이지만 MySQL 데이터베이스는 무료(배포판)
- 3) 리눅스, 유닉스, 윈도우 등 거의 모든 운영체제에서 사용가능
- 4) 처리 속도가 상당히 빠르고 대용량에 데이터 처리 용이
- 5) 설치 방법이 쉽고 초보자도 익히기 쉬움
- 6) 보안성이 우수

MySQL 설치

- ❖ <http://dev.mysql.com/downloads/> 사이트에 가서 왼쪽의 MySQL Community Server 을 선택합니다.



MySQL 설치

❖ MSI 버전은 닷넷 프레임워크 4.0이 설치되어 있어야 합니다.



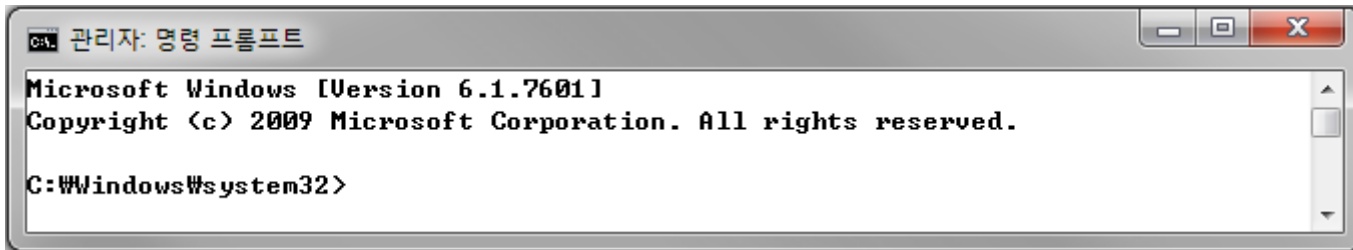
MySQL 설치

- ❖ 다운받은 설치파일을 실행해서 Install MySQL Products 를 선택합니다.



MySQL의 시작

명령 프롬프트의 실행



```
관리자: 명령 프롬프트
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

MySQL 접속 명령 1

```
C:\W> mysql -u 계정 -p비밀번호
mysql> use 데이터베이스명
```

MySQL 접속 명령 2

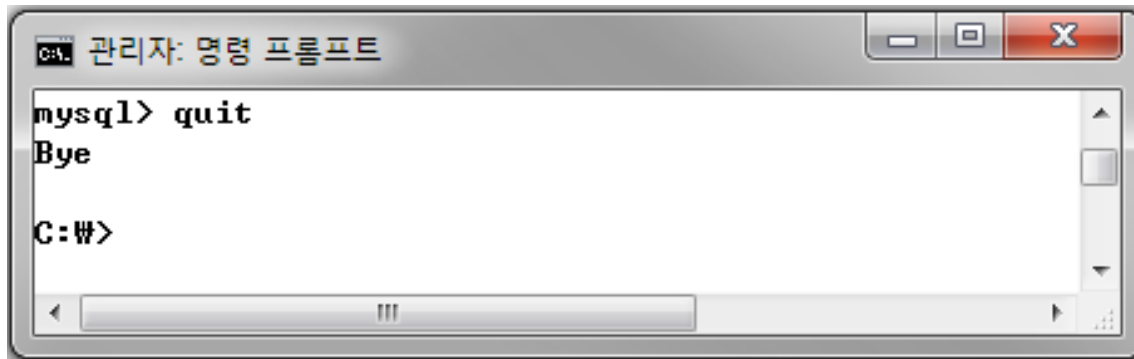
```
C:\W> mysql -u 계정 -h hostname -p비밀번호 데이터베이스명
```

ex)

```
계정 : kdhong, 비밀번호:1234, DB명:kdhong_db
C:\W> mysql -ukdhong -p1234 kdhong_db
```

MySQL의 시작

MySQL 접속 종료



A screenshot of a Windows command prompt window. The title bar reads "관리자: 명령 프롬프트" (Administrator: Command Prompt). The window content shows the following text: `mysql> quit` followed by `Bye` on the next line. Below that, the prompt `C:₩>` is visible. The window has standard Windows XP-style controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

```
mysql> quit
Bye

C:₩>
```


MySQL 데이터 타입

⊙ CHAR (M)

CHAR 데이터 타입은 고정된 길이의 문자열을 나타내는데 사용된다. 하나의 CHAR 문자열은 1-255 자 범위의 문자를 저장할 수 있다.

ex. car_model CHAR(10);

⊙ VARCHAR (M)

VARCHAR 데이터 타입은 가변적인 길이의 문자열을 저장하므로 CHAR 보다 는 좀더 융통성 있는 데이터 타입이다. VARCHAR 문자열은 1-255 자 범위의 문자를 저장할 수 있다.

(CHAR 는 포함된 데이터의 크기에는 관계없이 이미 지정된 가변적인 전체의 길이를 저장하는 반면에 VARCHAR 는 오직 들어가는 데이터의 양만을 저장하므로 데이터베이스 파일의 크기를 줄 일수 있다.)

ex. car_model VARCHAR(10);

⊙ INT (M) [Unsigned]

INT 데이터타입은 -2147483648 에서 2147483647 사이의 정수를 저장한다.

"unsigned" 옵션과 함께 0부터 4294967295 범위의 정수를 나타낼 수도 있다.

ex. light_years INT;

ex. light_years INT unsigned;

⊙ FLOAT [(M,D)]

FLOAT는 다소 정확한 숫자의 표시가 필요할 때 사용되어지며, 작은 양의 소수점 숫자를 나타낸다.

ex. rainfall FLOAT (4,2);

이것은 소수점 값이 될 수 있는 일년간 평균강수량을 나타낼 수 있다. 좀 더 명확하게 말하면 FLOAT (4,2) 는 4개의 저장할 수 있는 최대 자리 수와 2개의 소수점 이하의 자리 수를 가리킨다.

주의) FLOAT 는 어림잡은 수이기 때문에 MySQL 내에 포함 된 정수의 값이 아닌 데이터 타입으로 돈의 값을 나타낼 때에는 DECIMAL을 사용하는 것이 더 현명한 방법이다.

⊙ DATE

날짜와 관련된 정보를 저장한다. 디폴트 형식은 'YYYY-MM-DD' 이며, '0000-00-00' 에서 '9999-12-31'까지의 범위를 갖는다.

ex. the_date DATE;

⊙ TEXT / BLOB

text 와 blob 데이터 타입은 255 - 65535 자의 문자열을 저장할 때 사용된다. 기사와 같은 것을 저장하기에 유용하다. 그러나 VARCHAR 와 CHAR처럼 딱 잘라 비교할 수는 없다. 단지 BLOB 와 TEXT 사이에 차이점이 있다면 BLOB은 변하기 쉬운 경우에 비유할 수 있고, TEXT는 영향을 받지 않는 무감각한 경우에 비유할 수 있다.

⊙ SET

지정된 값으로부터 어떤 주어진 값을 선택하는, 정해진 문자열의 데이터 타입으로 그것은 하나의 값이 될 수도 있고 여러개의 값을 가질 수도 있다. 64개의 값까지 지정할 수 있다.

ex. transport SET ("truck", "wagon") NOT NULL;

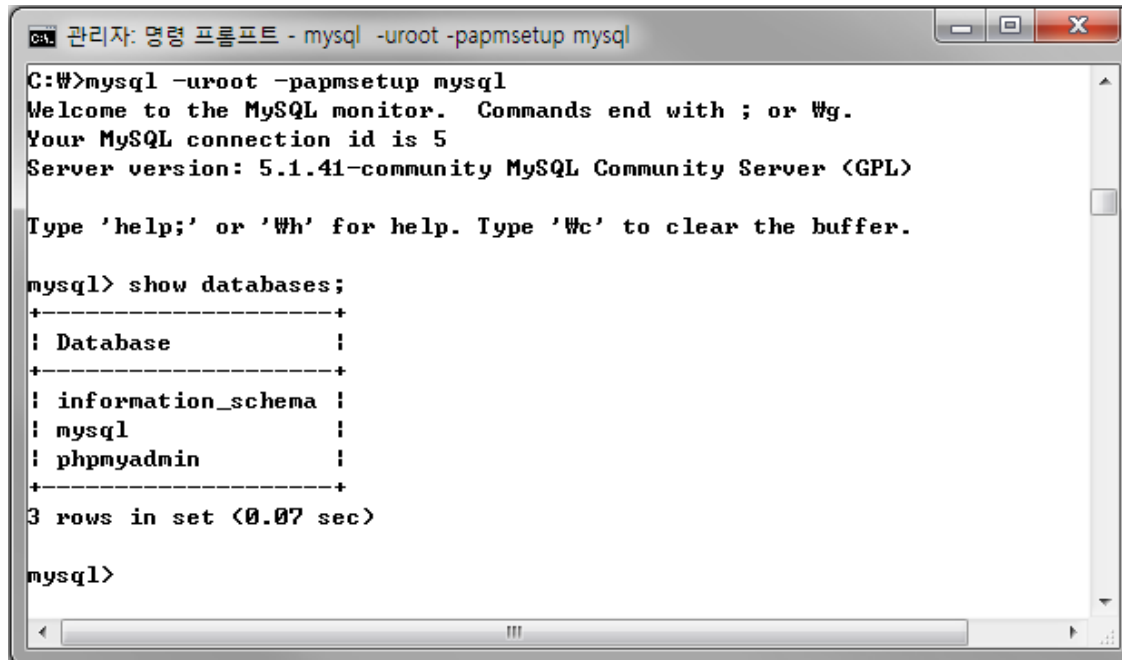
⊙ ENUM

SET 데이터 타입과 비슷한 특징을 갖는 정해진 문자열의 데이터 타입이지만 선택할 수 있는 값이 하나로만 정해져 있다는 점이 다르다. 한 공간의 바이트만을 가지므로 테이블내의 시간과 공간을 절약할 수 있다.

ex. transport ENUM ("truck", "wagon") NOT NULL;

MySQL의 시작

- ❖ 데이터베이스에 관리자 계정으로 접속
- ❖ 존재하는 데이터베이스 목록보기
mysql> show databases;



```
관리자: 명령 프롬프트 - mysql -uroot -papmsetup mysql
C:\#>mysql -uroot -papmsetup mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.1.41-community MySQL Community Server (GPL)

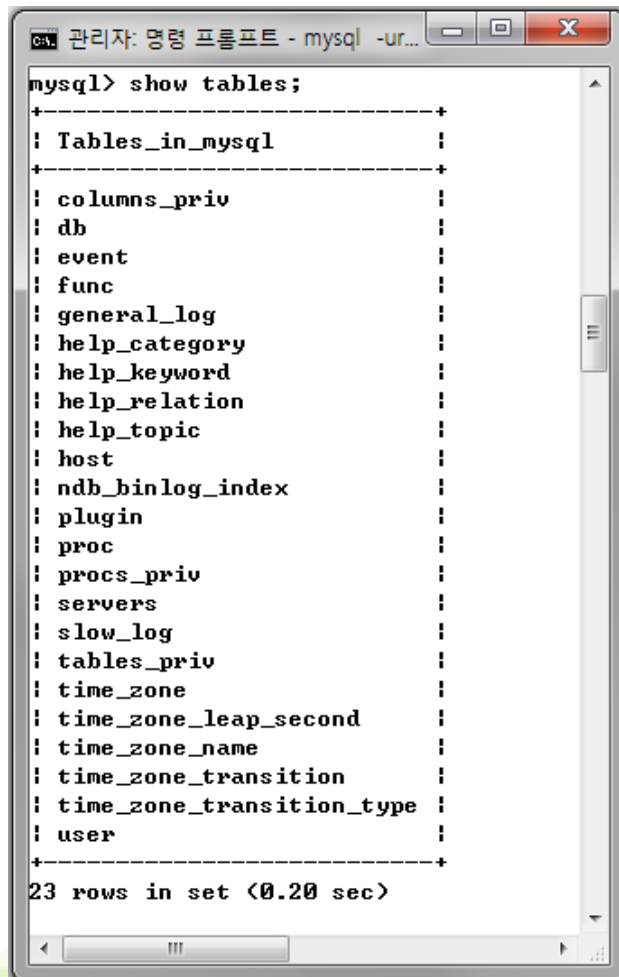
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+-----+
| Database                |
+-----+
| information_schema      |
| mysql                   |
| phpmyadmin              |
+-----+
3 rows in set (0.07 sec)

mysql>
```

관리자 계정으로 접속

MySQL의 시작



```
관리자: 명령 프롬프트 - mysql -ur...
mysql> show tables;
+-----+
| Tables_in_mysql |
+-----+
| columns_priv    |
| db              |
| event          |
| func            |
| general_log     |
| help_category   |
| help_keyword    |
| help_relation   |
| help_topic      |
| host            |
| ndb_binlog_index|
| plugin          |
| proc            |
| procs_priv      |
| servers         |
| slow_log        |
| tables_priv     |
| time_zone       |
| time_zone_leap_second|
| time_zone_name  |
| time_zone_transition|
| time_zone_transition_type|
| user            |
+-----+
23 rows in set (0.20 sec)
```

mysql 데이터베이스의 테이블

테이블 목록보기

mysql> show tables;

MySQL의 시작

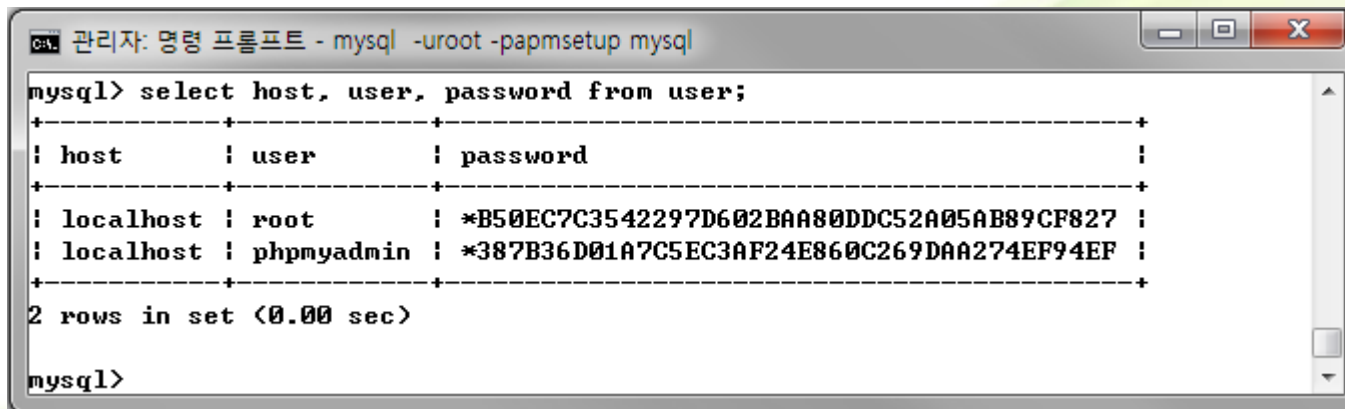
데이터베이스 생성 명령

```
mysql> create database 데이터베이스명;
```

테이블 구조 출력 명령

```
mysql> desc 테이블명;
```

user 테이블에 등록된 계정 목록 확인



```
C:\> 관리자: 명령 프롬프트 - mysql -uroot -pappmsetup mysql

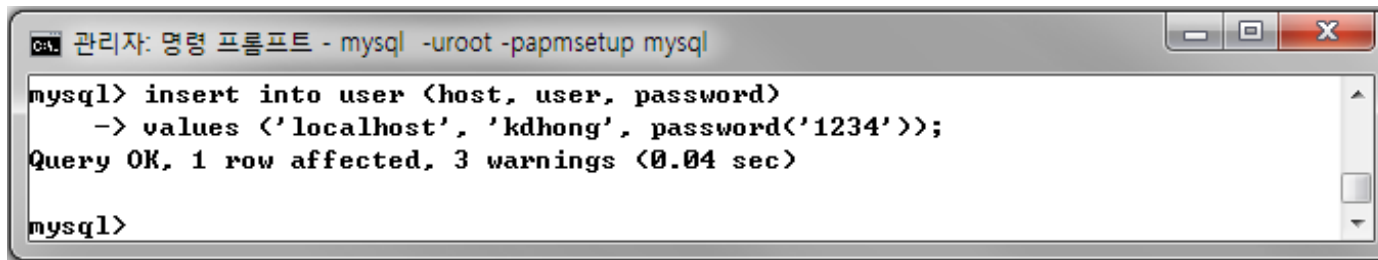
mysql> select host, user, password from user;
+-----+-----+-----+
| host      | user      | password                                     |
+-----+-----+-----+
| localhost | root      | *B50EC7C3542297D602BAA80DDC52A05AB89CF827 |
| localhost | phpmyadmin | *387B36D01A7C5EC3AF24E860C269DAA274EF94EF |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

MySQL의 시작

필드에 새로운 데이터 입력

```
mysql> insert into 테이블명 (필드1, 필드2, 필드3) values  
      (필드1_값, 필드2_값, 필드3_값);
```

A screenshot of a Windows command prompt window titled "관리자: 명령 프롬프트 - mysql -uroot -p" with the command "paprmssetup mysql" in the title bar. The prompt shows the execution of an SQL insert statement into a table named 'user'. The statement is: "mysql> insert into user (host, user, password) -> values ('localhost', 'kdhong', password('1234'));" The output of the query is: "Query OK, 1 row affected, 3 warnings (0.04 sec)" followed by a new prompt "mysql>".

```
C:\> 관리자: 명령 프롬프트 - mysql -uroot -paprmssetup mysql  
mysql> insert into user (host, user, password)  
      -> values ('localhost', 'kdhong', password('1234'));  
Query OK, 1 row affected, 3 warnings (0.04 sec)  
mysql>
```

user 테이블에 계정(kdhong)과 비밀번호(1234) 등록

user 테이블에 계정 등록

```
mysql> insert into user (host, user, password)  
      -> values ( 'localhost' , 'kdhong' , password( '1234' ));
```

MySQL의 시작

```
C:\> 관리자: 명령 프롬프트 - mysql -uroot -papmsetup mysql
mysql> desc db;
```

Field	Type	Null	Key	Default	Extra
Host	char(60)	NO	PRI		
Db	char(64)	NO	PRI		
User	char(16)	NO	PRI		
Select_priv	enum('N','Y')	NO		N	
Insert_priv	enum('N','Y')	NO		N	
Update_priv	enum('N','Y')	NO		N	
Delete_priv	enum('N','Y')	NO		N	

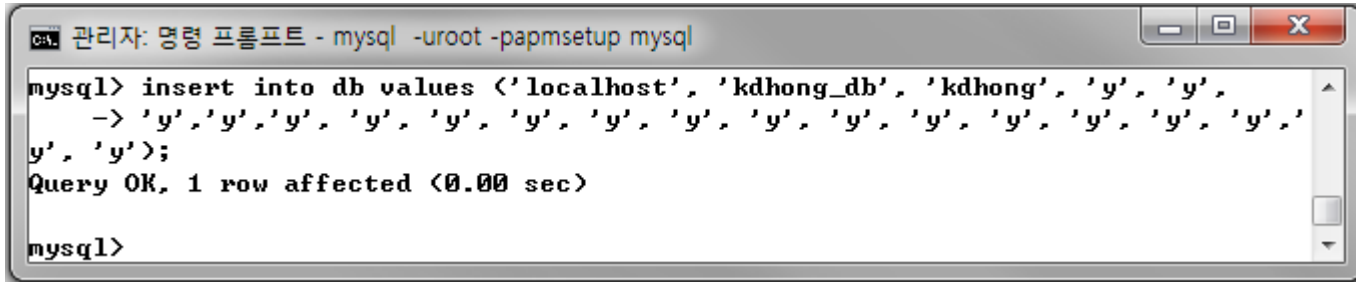
db 테이블의 구조

db 테이블에 데이터베이스 사용 권한 등록

데이터베이스 테이블 구조 파악

```
mysql> desc db;
```


MySQL의 시작



```
관리자: 명령 프롬프트 - mysql -uroot -padminsetup mysql
mysql> insert into db values ('localhost', 'kdhong_db', 'kdhong', 'y', 'y',
-> 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y',
y', 'y');
Query OK, 1 row affected (0.00 sec)

mysql>
```

데이터베이스 사용 권한 설정

db 테이블에 사용 권한 설정

```
mysql> insert into db values ('localhost', 'kdhong_db', 'kdhong', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y');
```

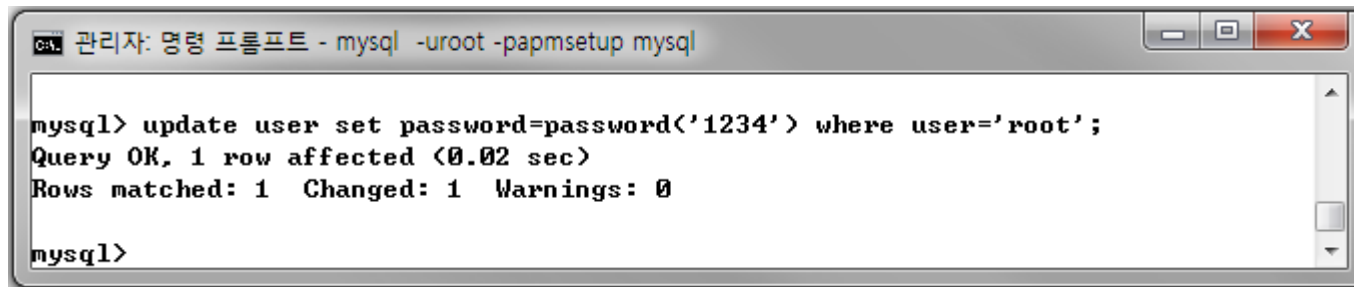
user, db 테이블의 변경된 내용 적용

```
mysql> flush privileges;
```

MySQL의 시작

update 명령

```
mysql> update 테이블명 set 필드명 = password('새로운_비밀번호')  
      where 필드 = '필드값';
```



The screenshot shows a Windows command prompt window titled "관리자: 명령 프롬프트 - mysql -uroot -padminsetup mysql". The command entered is `mysql> update user set password=password('1234') where user='root';`. The output shows the query was successful, affecting 1 row in 0.02 seconds. The status is "Query OK, 1 row affected (0.02 sec)" and "Rows matched: 1 Changed: 1 Warnings: 0". The prompt returns to `mysql>`.

```
관리자: 명령 프롬프트 - mysql -uroot -padminsetup mysql  
  
mysql> update user set password=password('1234') where user='root';  
Query OK, 1 row affected (0.02 sec)  
Rows matched: 1  Changed: 1  Warnings: 0  
  
mysql>
```

root의 비밀번호 변경

데이터베이스 접속 명령

```
mysql -u계정 -p비밀번호 데이터베이스명
```

데이터베이스 생성 명령

```
create database 데이터베이스명;
```

데이터베이스 목록 출력 명령

```
show databases;
```

데이터베이스 삭제 명령

```
drop database 데이터베이스명;
```

MySQL을 설치하고 root 계정으로 접속 한 후 데이터베이스를 생성

grant all privileges on 데이터베이스이름.* to 계정@'%' identified by '비밀번호'

❖ %대신에 ip를 기재하면 특정 ip에서만 접속이 허용됩니다.

데이터베이스 관련 명령

DCL(Data Control Language)

① 권한 부여

grant 권한 on DB명.table명 to 유저명 [identified by '암호'] [with grant option]
ex) grant all privileges on *.* to kim identified by '1234' with grant option;
grant all on test.* to park identified by '1234';
grant select,insert on test.* to lee identified by '7890';

② 권한박탈

revoke 권한 on DB명 from 유저명; ex) revoke all on test.* from park;

③ mysql DB

mysql데이터베이스 권한에 관한 규정

- . user table : mysql권한에 대한 정보규정
- . db : db권한에 대한 정보 저장(root제외), host이름 %는 어디서나 접속 가능
- . tables-priv : 특정 table접속 가능권한
- . columns-priv : 특정 column 접속 가능 권한
- . host : 특정 computer 접속 가능 권한

. root password 변경

update user set password=password('1234') where user='root';

. User삭제 delete from user where user='lee';

④ 권한 적용

flush privileges;

테이블(The Bigger Picture: Tables)

```
CREATE TABLE sawon (num int not null primary key, name VARCHAR (15),  
    phone INT, age int, sex enum('man','woman') not null default 'woman');
```

테이블이 성공적으로 만들어지면 다음과 같은 화면이 출력될 것이다.

Query OK, 0 rows affected (0.10 sec)

주의(1) : 서로 다른 두 개의 테이블은 같은 이름을 가질 수 없다.

주의(2) : 각각의 데이터 공간은 컬럼(column)이라는 개체의 속성으로 나타낸다.

1) 컬럼(Column)의 특징 :

하나의 이름이 숫자로만 구성될 수 없다. 하나의 이름은 숫자로 시작할 수 있다.
하나의 이름은 64자까지 가능하다.

2) 테이블의 다른 옵션들 :

. 기본 키(Primary Key) :

다른 속성들로부터 하나의 레코드를 유일하게 식별하는데 사용되어지며, 두 개의 레코드는 똑같은 기본 키를 가질 수 없다. 이것은 두 개의 레코드가 중복되어지는 실수를 없애려고 할 때 아주 적합하게 사용된다.

. auto_Increment :

이 함수(function)를 가진 컬럼(column)은 레코드에 삽입될 때 자동적으로 한 개씩 값이 (previous + 1) 증가한다. data가 'NULL'값 일 때도 컬럼(column)에 자동적으로 한 개씩 값이 증가하여 삽입된다.

. NOT NULL :

NULL값(값이 없는)을 절대로 허용하지 않는 컬럼(column)을 의미한다. 숫자는 0 문자는 null 이 들어감, 날짜 0000-00-00

ex) soc_sec_number INT PRIMARY KEY;

Ex) ID_NUMBER INT AUTO_INCREMENT;

. Auto_Increment :

이 함수(function)를 가진 컬럼(column)은 레코드에 삽입될 때 자동적으로 한 개씩 값이 (previous + 1) 증가한다. data가 'NULL'값 일 때도 컬럼(column)에 자동적으로 한 개씩 값 이 증가하여 삽입된다.

. NOT NULL :

NULL값(값이 없는)을 절대로 허용하지 않는 컬럼(column)을 의미한다.
숫자는 0 문자는 null이 들어감, 날짜 0000-00-00

데이터베이스 테이블 생성 명령

```
mysql> create table 테이블명(  
    필드명1 타입 컬럼제약조건,  
    필드명2 타입,  
    필드명3 타입,  
    .....  
    제약 조건);
```

❖자료형: tinyint, char, varchar, smallint, int, integer, bigint, float, double, char(n), varchar(n), BLOB, text, date, datetime, timestamp, time

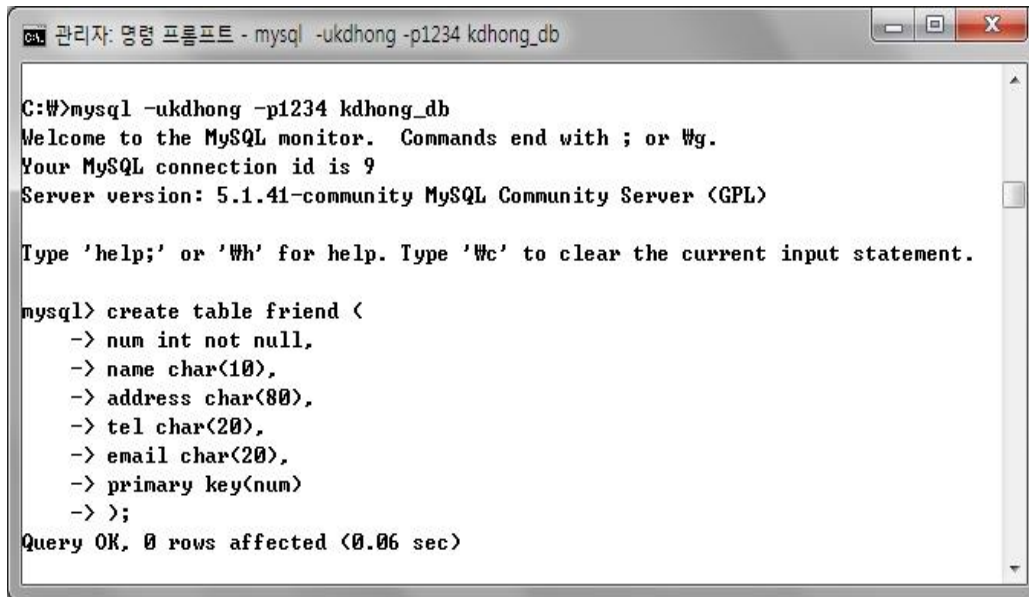
❖제약조건: unique, primary key, not null, default 값, check 조건, foreign key

❖create table School(code int not null primary key,
name varchar(15));

❖create table Junior(stu_id int not null auto_increment primary key,
name varchar(15),school_code int,
foreign key(school_code) references School(code)
on delete cascade
on update cascade);

❖create table Senior(stu_id int not null auto_increment primary key, name
varchar(15), school_code int, foreign key(school_code) references School(code)
on delete set null on update cascade);

데이터베이스 테이블 관련 명령



```
C:\>관리자: 명령 프롬프트 - mysql -ukdhong -p1234 kdhong_db

C:\>mysql -ukdhong -p1234 kdhong_db
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.1.41-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create table friend (
-> num int not null,
-> name char(10),
-> address char(80),
-> tel char(20),
-> email char(20),
-> primary key(num)
-> );
Query OK, 0 rows affected (0.06 sec)
```

주소록 테이블(friend) 만들기

```
create table friend (
-> num int not null,
-> name char(10),
-> address char(80),
-> tel char(20),
-> email char(20),
-> primary key(num)
-> );
```

데이터베이스 테이블 관련 명령

테이블 변경하기(Alter table)

① 용도

- 새로운 열 추가하기, 열의 속성 바꾸기, 열 삭제하기, 테이블 이름 바꾸기

② alter table 테이블 변경명령;

Example: 테이블 이름 변경하기(Rename the table)

변경명령 : rename 새로운 테이블명; add 필드형 자료형 옵션
 drop 필드명; change 필드명 새로운 필드명 옵션;
 modify 필드명 자료형 옵션; add primary key (필드명);
 drop primary key; add index 인덱스명 (필드명);
 drop index 인덱스명;

확인

desc table명;
show index from table명;

데이터베이스 테이블의 필드 추가 명령

alter table 테이블명 add 새로운 필드명 필드타입 [first 또는 after 필드명];

```
관리자: 명령 프롬프트 - mysql -ukdhong -p1234 kdhong_db

mysql> alter table friend add age int;
Query OK, 0 rows affected (0.06 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc friend;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| num   | int(11) | NO   | PRI | NULL    |       |
| name  | char(10) | YES  |     | NULL    |       |
| address | char(80) | YES  |     | NULL    |       |
| tel   | char(20) | YES  |     | NULL    |       |
| email | char(20) | YES  |     | NULL    |       |
| age   | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>
```

새로운 필드 추가 명령

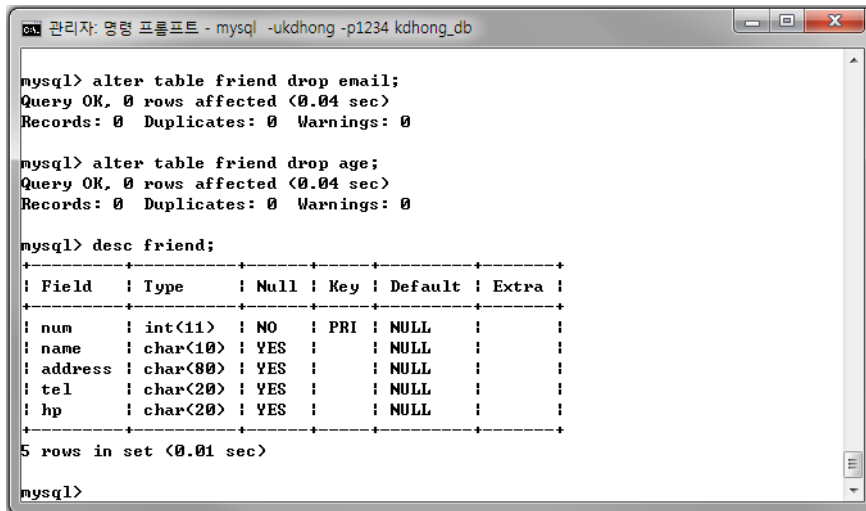
ex) 앞서 만든 friend 테이블 나이 필드를 정수형으로 추가

1. mysql> alter table friend add age int;
2. mysql> desc friend;

데이터베이스 테이블 관련 명령

데이터베이스 테이블의 특정 필드 삭제 명령

alter table 테이블명 drop 삭제할 필드명1, 삭제할 필드명2;



```
관리자: 명령 프롬프트 - mysql -ukdhong -p1234 kdhong_db

mysql> alter table friend drop email;
Query OK, 0 rows affected (0.04 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table friend drop age;
Query OK, 0 rows affected (0.04 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc friend;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| num   | int(11) | NO   | PRI | NULL    |       |
| name  | char(10) | YES  |     | NULL    |       |
| address | char(80) | YES  |     | NULL    |       |
| tel   | char(20) | YES  |     | NULL    |       |
| hp    | char(20) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql>
```

필드 삭제

ex) friend 테이블에서 email과 age 필드 삭제

1. mysql> alter table friend drop email;
2. mysql> alter table friend drop age;
3. mysql> desc friend;

데이터베이스 테이블 관련 명령

데이터베이스 테이블의 필드 수정 명령

alter table 테이블명 change 이전 필드명 새로운 필드명 필드 타입;

```
C:\ 관리자: 명령 프롬프트 - mysql -ukdhong -p1234 kdhong_db

mysql> alter table friend change tel phone int;
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc friend;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| num   | int(11) | NO   | PRI | NULL    |       |
| name  | char(10) | YES  |     | NULL    |       |
| address | char(80) | YES  |     | NULL    |       |
| phone | int(11) | YES  |     | NULL    |       |
| hp    | char(20) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql>
```

필드 수정 명령

ex) friend 테이블 필드 중 tel char(20)을 phone int로 변경

mysql> alter table friend change tel phone int;

데이터베이스 테이블 필드 타입 수정 명령

```
alter table 테이블명 modify 필드명 새로운 타입;
```

데이터베이스 테이블명 수정 명령

```
alter table 이전 테이블명 rename 새로운 테이블명;
```

필드 타입 수정 명령

ex) name 필드의 타입을 int로 변경

```
mysql> alter table friend modify name int;
```

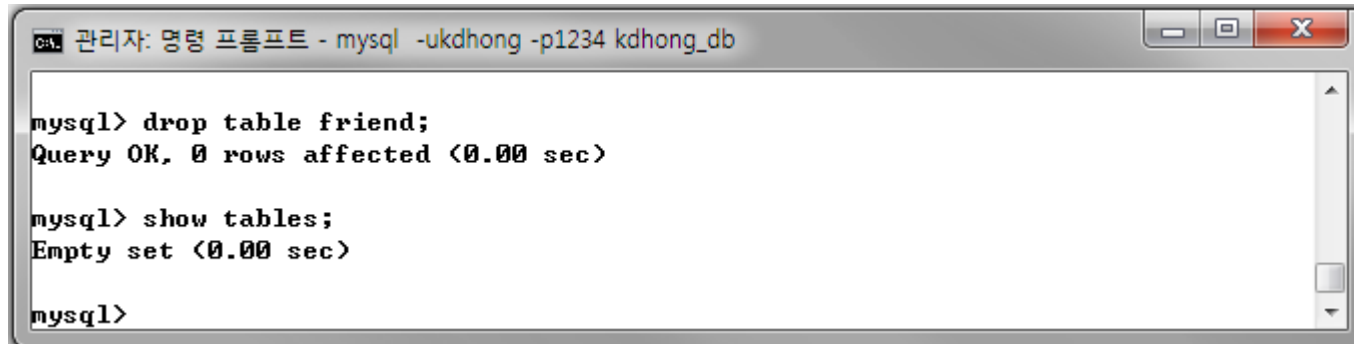
데이터베이스 테이블명 수정 명령

ex) 테이블명 friend에서 student로 변경

```
mysql> alter table friend rename student;
```

데이터베이스 테이블 삭제 명령

drop table 테이블명;



```
관리자: 명령 프롬프트 - mysql -ukdhong -p1234 kdhong_db

mysql> drop table friend;
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
Empty set (0.00 sec)

mysql>
```

테이블 삭제

ex) friend 테이블 삭제

1. mysql> drop table friend;
2. mysql> show tables;

DML(Data Manipulation Language)

① 데이터 삽입

```
INSERT INTO PROJ VALUES ('MA2114;',',','B01',',',NULL,CURRENT DATE, NULL);
```

INSERT 문을 사용하면 테이블에 데이터 행을 추가할 수 있습니다.

이런 작업은 여러 가지 방법으로 수행할 수 있습니다. PROJECT 테이블에서 컬럼이 만들어진 것과 같은 순서로 각 컬럼에 대한 값과 컬럼 값을 지정해야 합니다.

PROJNO 컬럼에는 MA2114, PROJNAME 컬럼에는 공백, DEPTNO 컬럼에는 B01, RESPEMP 컬럼에는 공백, PRSTAFF 컬럼에는 NULL 값, PRSTDAT 컬럼에는 현재 날짜 값, PRENDATE 컬럼에는 NULL 값이 들어갑니다. 화면의 아래쪽에는 PROJ 테이블에 이 값들을 삽입한 결과가 보입니다. PRSTAFF와 PRENDATE 컬럼의 대신에 주목하십시오. 이 대신은 INSERT 문에 지정한 NULL 값을 나타냅니다.

```
INSERT INTO PROJ (DEPTNO, PROJNO, PROJNAME, RESPEMP PRSTDAT) VALUES ('B01','MA2114;',',',',',CURRENT DATE)
```

또는 컬럼 이름을 괄호로 묶어서 지정하는 INSERT 문을 사용할 수도 있습니다. 화면에 이 방법이 나타나 있습니다. 이 구문 형식을 사용하면 모든 컬럼을 지정할 필요가 없습니다. 값이 주어지지 않은 NULL 지정이 가능한 컬럼은 자동으로 NULL로 지정됩니다. 값이 해당 컬럼에 대응하면 컬럼을 특정 순서로 지정하지 않아도 됩니다.

SQL 명령의 일괄 실행

- 명령 일괄 실행 과정

- 텍스트 파일에 데이터베이스 테이블 생성 명령 저장
- 파일이 저장된 폴더로 이동
- SQL 명령 일괄 실행 및 실행 확인

- 파일을 이용한 데이터 저장

grant file on *.* to 계정;

. 데이터 생성(mysql의 해당 directoy에 text file만들어 저장)

. load data infile "파일명" replace into table 테이블 명;

- 데이터 삭제

delete from table명 where 조건;

delete from personal;

- 데이터 수정

update 테이블 명 set 필드명=값 where 조건;

update personal set bonus=null where bonus=0;

update division set position='daegu' phoneno='051-122-4567' where dno=40;

샘플 데이터 작성

pno	pname	job	manager	startdate	pay	bonus	dno
1111	smith	manager	1001	1990-12-17	1000	null	10
1112	ally	salesman	1116	1991-02-20	1600	500	30
1113	word	salesman	1116	1992-02-24	1450	300	30
1114	james	manager	1001	1990-04-12	3975	null	20
1001	bill	president	null	1989-01-10	7000	null	10
1116	johnson	manager	1001	1991-05-01	3550	null	30
1118	martin	analyst	1111	1991-09-09	3450	null	10
1121	kim	clerk	1114	1990-12-08	4000	null	20
1123	lee	salesman	1116	1991-09-23	1200	0	30
1226	park	analyst	1111	1990-01-03	2500	null	10

personal table

pno : int not null primary key

pname : varchar(10) not null

job : varchar(15) not null


manager : int

startdate : date

pay : int

bonus : int

dno : int not null



dno	dname	phone	position
10	finance	032-277-0411	busan
20	research	061-535-1212	
30	sales	02-555-4985	
40	marketing	031-284-3800	

division table

dno : int not null primary key

dname : varchar(15) not null

phone : varchar(20) not null

position : varchar(10) not null



Query

1. 기본 SELECT

. SELECT 문에서 SELECT, FROM, WHERE, 그 다음 ORDER BY와 같은 정해진

1) SELECT 문에서 모든 컬럼 가져오기

```
select * from table명;
```

```
SELECT * FROM personal;
```

```
Select pno,pname,pay from personal;
```

테이블의 모든 컬럼을 선택하는 간단한 방법은 SELECT *를 지정하는 것입니다.

2) 행 제어하기

```
SELECT * from personal where dno=10;
```

```
Select pno,pname,pay from personal where dno=20;
```

```
Select * from personal where job='salesman';
```

3) 비교 연산자

WHERE 절에서는 등호 연산자 외에도 기타 비교 연산자를 사용할 수 있습니다.

>, <, >=, <=, =, != >= 100과 JOB <>'manager'식으로 사용합니다

4) NULL특성 과 NOT NULL WITH DEFAULT

NULL 특성은 관계형 이론의 일부입니다.

컬럼이 NOT NULL인 경우에는 값을 가지고 있어야 하며, 숫자 컬럼의 경우는 0, 문자 컬럼의 경우는 공백이 올 수 있습니다. NULL 지정이 가능한 컬럼에 값을 제공할 필요는 없습니다. NULL 지정이 가능한 것으로 지정된 컬럼에는 해당 필드가 값을 가지고 있는지 여부를 나타내는 1바이트 FLAG가 추가됩니다.

NULL로 플래그가 지정된 필드는 알 수 없는 값을 포함하고 있습니다. 0은 숫자 필드에서, 공백은 문자 데이터에서 의미 있는 값으로 사용되기 때문에 이 알 수 없는 값은 0이나 공백은 아닙니다. NULL 값은 문자 그대로 "알 수 없는" 값입니다.

컬럼을 NOT NULL WITH DEFAULT로 지정할 수도 있습니다. 이 컬럼은 값이 필요하며, 값을 지정하지 않으면 시스템이 기본값을 제공합니다.

- 숫자 데이터의 경우 기본값은 0입니다.
- 문자 데이터의 기본값은 고정 길이의 경우 공백(BLANK)이며 가변 길이의 경우 제로 길이(ZERO LENGTH)입니다.

NULL 값을 알 수 없기 때문에 이 값을 검색하려면 WHERE 절에서 특수 구문을 사용해야 합니다.

```
Select * from personal where bonus is null;
```

5) 다중 조건식

select * from personal where dno > 20 and pno > 1112;

select * from personal where dno > 20 or pno > 1112;

select * from personal where dno > 20 && pno > 1112;

select * from personal where dno > 20 || pno > 1112;

6) IN

WHERE job IN('sales', 'manager', 'clerk')

WHERE JOB = 'manager' OR JOB = 'sales' OR JOB = 'clerk'

7) BETWEEN

where dno between 20 and 30; >= 20 그리고 <= 30와 같습니다.

문자 값의 범위를 선택할 수도 있습니다.

8) 부분 검색

SQL은 **LIKE** 키워드와 함께 사용되는 % 및 _와 같은 특수 기호를 사용

select * from personal where job like 's%';

Select * from personal where job like '%r';

Select * from dno like '3_';

9) 부정

WHERE NOT LIKE 'G%'; Where not between

2.테이블 행의 결과

1) OEDER BY

ORDER BY 절은 컬럼 값을 기준으로 결과를 정렬합니다.

```
select * from personal order by bonus;
```

```
select * from personal order by bonus, dno;
```

하나 이상의 컬럼에 ORDER BY를 수행할 수 있습니다. 지정된 첫 번째 컬럼에 따라 결과가 정렬되며, 이 결과가 동일하면 ORDER BY의 두 번째 컬럼에 따라 정렬
이 예제에서 결과는 JOB을 기준으로 오름차순으로 정렬됩니다. 오름차순은 기본값입니다. 다른 방법으로 ASC를 지정할 수 있습니다.

```
Select * from personal order by dno desc;
```

이 예제에서 볼 수 있는 것처럼 먼저 YEARS DESC를 지정하여 결과의 순서를 변경할 수 있습니다.

2) DISTINCT

```
select dno from personal;
```

중복 값을 제거하고 회사 내의 다른 부서만 보려면 SELECT 바로 뒤에 키워드 DISTINCT를 지정합니다.

```
SELECT DISTINCT Dno FROM personql;
```

3) 표현식을 이용한 값

SELECT 문에서 수학 계산을 사용할 수 있습니다. +(더하기), -(빼기), *(곱하기), /(나누기)를 사용합니다.

2개의 컬럼을 더할 수도 있습니다. 이 예제에서는 부서의 각 직원별 급여와 커미션을 알아보려고 합니다. 또한 총 급여를 알기 위해 커미션이 추가된 급여를 보고 싶을 수도 있습니다.

```
select pname,pay,pay + bonus from personal;
```

※. Null + 값 = null

4) AS

AS 절을 사용하여 계산된 컬럼에 이름을 지정하고 다른 컬럼에 더 읽기 쉬운 이름을 지정할 수 있습니다.

```
SELECT pname,pay,pay+bonus as income from personal;
```

```
SELECT pname,pay,pay+bonus as 'month income' from personal;
```

```
SELECT pname,pay,pay+bonus as income from personal order by 1;
```

```
SELECT pname,pay,pay+bonus as 'month income' from personal order by  
'month income';
```

컬럼함수 / GROUP BY

1) 컬럼 함수란 무엇인가?

이 함수들은 SUM, AVG, MIN, MAX, COUNT입니다. 컬럼 함수는 컬럼 값들을 단일 값으로 요약하거나 집계.

2) 컬럼 함수 사용

```
SELECT SUM(pay), AVG(pay), MIN(pay), MAX(pay) FROM personal;  
AVG(pay + bonus)을 사용 가능
```

3) COUNT(*) 컬럼 함수

```
select count(*), count(distinct dno) from personal;  
SELECT COUNT(DISTINCT dno), COUNT(*), AVG(pay) FROM personal  
WHERE pay > 18000;
```

4) GROUP BY

```
Select sum(pay) from personal;  
Select dno,sum(pay) from personal group by dno;  
Select dno,sum(pay) from personal where pay > 1500 group by dno;  
SELECT DEPT, SUM(pay) AS TOTAL_SALARY, SUM(bonus) AS  
TOTAL_COMM FROM STAFF WHERE JOB <> 'manager'GROUP BY dno;
```

SELECT에 지정된 모든 항목은 일관성을 지녀야 합니다. 즉, 모든 세부 열만을 선택하거나 집계값 만을 선택해야 합니다.

세부 열과 집계값을 혼용하는 경우 모든 세부 열은 GROUP BY 문에 의해 포함되어야 합니다. 이 SELECT 문에서 GROUP BY를 실행할 때 그 결과에 유의하십시오.

```
SELECT DEPT, SUM(SALARY), SUM(COMM.) FROM STAFF WHERE JOB <>
'MGR' GROUP BY DEPT
```

```
SELECT dno, JOB FROM STAFF WHERE JOB <> 'MGR' GROUP BY dno;
```

이 SELECT 문은 실행되지 않고 오류가 발생합니다.

모든 데이터를 집계하려면 GROUP BY 절에 모든 세부 사항을 지정
이 SELECT 문은 GROUP BY DEPT, JOB S를 사용하는 경우에만 실행됩니다.

GROUP BY 와 HAVING

조건을 지정하려면 WHERE 또는 HAVING 절을 사용합니다. WHERE 절은 행 수준에서 조건을 지정하며 HAVING 절은 그룹 수준에서 조건을 지정
HAVING 절은 GROUP BY 절 바로 다음에 나옵니다.

```
SELECT DEPT, SUM(pay) FROM personal GROUP BY dno;
```

```
SELECT DEPT, SUM(pay) FROM personal GROUP BY dno HAVING SUM(pay) >
65000
```

처리 순서

- ① SELECT
- ② FROM
- ③ WHERE
- ④ GROUP BY
- ⑤ HAVING
- ⑥ ORDER BY

4. SACLAR함수

1) 스칼라 함수의 소개

집계 값이 아닌 단일 값을 산출합니다. 스칼라 함수는 중첩될 수 있습니다.

즉 내부에 다른 스칼라 함수를 포함할 수 있습니다.

또한 스칼라 함수에 컬럼 함수를 포함할 수 있으며 그 반대의 경우도 가능합니다.

. 문자열 추출하기 - SUBSTR

```
SELECT SUBSTR(job, 1, 4) FROM personal;
```

```
SELECT * FROM personal WHERE SUBSTR(job,1,1) = 'm';
```

. 문자열 결합하기 - CONCAT

```
select concat(pname,' job is ',job) from personal;
```

2) 숫자 관련 함수

- ▶ ABS(숫자) : 절대값 출력.

```
select abs(123);
```

- ▶ CEILING(숫자) : 값보다 큰 정수 중 가장 작은 수.

--양수일 경우는 소숫점 자리에서 무조건 반올림(4.0과 같은 0 값은 제외)

--음수일 경우는 소숫점 자리를 무조건 버림

```
select ceiling(4.0);      select ceiling(4.1);      select ceiling(4.9);
```

- ▶ FLOOR(숫자) : 값보다 작은 정수 중 가장 큰 수[실수를 무조건 버림(음수일 경우는 제외)].

--음수일 경우는 [.0/.00/.000/...] 을 제외하고 무조건 소숫점을 버리고 반내림(?)

```
select floor(4.0);  select floor(4.1);  select floor(4.9);  select floor(-4.6789);
```

- ▶ ROUND(숫자, 자릿수) : 숫자를 소수점 이하 자릿수에서 반올림

--자릿수를 생략하면 소숫점이 5 이상일 때 반올림/자릿수를 지정하면 지정한 자리수에서 반올림

```
select round(4.5);      select round(4.55);
```

```
select round(-4.5);      select round(4.556);
```

```
select round(4.556,0);  select round(4.556,1);
```

```
select round(4.556,2);  select round(45.556,-1);  select round(455.556,-2);
```

▶ TRUNCATE(숫자, 자릿수) : 숫자를 소수점 이하 자릿수에서 버림.

➔ 소숫점 이전으로 정하면 소숫점이하는 버리고 나머지 값은 0 값으로 처리

/ 예) truncate(9999, -3) --> 9000

➔ 소숫점이하로 정하며, 해당숫자가 자릿수보다 소숫점이 모자랄경우 0 값 대치

/ 예) truncate(999, 3) --> 999.000

--음수일 경우는 해당자릿수에서 소숫점을 버리면서 무조건 반올림

==> (자릿수 숫자에서 이후 숫자가 0 일 경우는 제외 / 예) -4.0, 0/-400, -2/-4.1230, 4)

==> 음수도 소숫점이하로 정하며, 자릿수보다 소숫점이 모자랄경우 0 값 대치

➔ 소숫점 이전으로 정하면 소숫점이하는 버리고 나머지 값은 역시 0 값으로 처리

▶ POW(X, Y) 또는 POWER(X, Y) : X의 Y승

--소숫점이 있는 경우도 실행, 단 음수는 양수로 승처리

select pow(-2.5, 2); select pow(1.5, 2);

▶ MOD (분자, 분모) : 분자를 분모로 나눈 나머지를 구한다. (연산자 %와 같음)

select mod(12, 5); ==> 2 select 12%5; ==> 2

▶ GREATEST(숫자1, 숫자2, 숫자3...) : 주어진 수 중 제일 큰 수 리턴.

select greatest(100, 101, 90);

▶ LEAST(숫자1, 숫자2, 숫자3...) : 주어진 수 중 제일 작은 수 리턴.

select least(100, 101, 90);

▶ INTERVAL(a, b, c, d,) : a(숫자)의 위치 반환

--두 번째 이후는 오름차순 정렬이 되어야 함

INTERVAL(5, 2, 4, 6, 8) ==> 2(5는 4와 6사이에 존재, 4~6사이의 위치가 앞에서 2번째)

select interval(4, 1, 2, 3, 5, 6);

3) 문자 관련 함수

- ▶ ASCII(문자) : 문자의 아스키 코드값 리턴.

```
SELECT ASCII('문자');      select ascii('A');
```

- ▶ CONCAT('문자열1','문자열2','문자열3'...) : 문자열들을 이어준다.

```
select concat('ASP','PHP','SQL',' WEB STUDY');
```

- ▶ INSERT('문자열','시작위치','길이','새로운문자열') : 문자열의 시작위치부터 길이만큼 새로운 문자열로 대치

'시작위치' 와 '길이'는 문자열이 아니므로 작은따옴표로 굳이 묶어주지 않아도 된다.

```
select insert('MySql web study','7','3','offline');
```

```
select insert('MySql web study',7,3,'offline');
```

- ▶ REPLACE('문자열','기존문자열','바뀔문자열') : 문자열 중 기존문자열을 바뀔 문자열로 바꾼다.

```
select replace('MySql web study','web','offline');
```

- ▶ INSTR('문자열','찾는문자열') : 문자열 중 찾는 문자열의 위치값을 출력 -> 값이 존재하지 않으면 0값 리턴

```
select instr('MySql web study','s');
```

```
select instr('MySql web study','S');
```


▶ LEFT('문자열',개수) : 문자열 중 왼쪽에서 개수만큼을 추출.

```
select left('MySQL web study',5);      select left('MySQL web study','5');
```

▶ RIGHT('문자열',개수) : 문자열 중 오른쪽에서 개수만큼을 추출.

```
select right('MySQL web study',5);      select right('MySQL web study','5');
```

▶ MID('문자열',시작위치,개수) : 문자열 중 시작위치부터 개수만큼 출력

```
select mid('MySQL web study',7,3);      select mid('MySQL web study','7','3');
```

▶ SUBSTRING('문자열',시작위치,개수) : 문자열 중 시작위치부터 개수만큼 출력

```
select substring('Mysql web study',11,5);  
select substring('Mysql web study','11','5');
```

▶ LTRIM('문자열') : 문자열 중 왼쪽의 공백을 없앤다.

```
select ltrim('      web study');
```

▶ RTRIM('문자열') : 문자열 중 오른쪽의 공백을 없앤다.

```
select rtrim('web study      ');
```

▶ TRIM('문자열') : 양쪽 모두의 공백을 없앤다.

```
select trim('    web study    ');
```

▶ LCASE('문자열') 또는 LOWER('문자열') : 소문자로 바꾼다.

```
select lcase('MYSQL');      select lower('MySQL');
```

▶ UCASE('문자열') 또는 UPPER('문자열') : 대문자로 바꾼다.

```
select ucase('mysql');      select upper('mysql');
```

▶ REVERSE('문자열') : 문자열을 반대로 나열한다.

예) REVERSE('abcde') ==> edcba

```
select reverse('lqSyM');
```

4) 날짜 관련 함수

▶ NOW() 또는 SYSDATE() 또는 CURRENT_TIMESTAMP() 현재 날짜와 시간 출력

※ 함수의 상황이 숫자인지 문자열인지에 따라

YYYYMMDDHHMMSS 또는 'YYYY-MM-DD HH:MM:SS' 형식으로 반환한다.

예) select now(); ==> '2001-05-07 09:10:10'

select now() + 0; ==> 20010507091010

▶ CURDATE() 또는 CURRENT_DATE() 현재 날짜 출력

※ 함수의 상황이 숫자인지 문자열인지에 따라

YYYYMMDD 또는 'YYYY-MM-DD' 형식으로 반환한다.

예) select curdate(); ==> '2001-05-07'

select curdate() + 0; ==> 20010507

▶ CURTIME() 또는 CURRENT_TIME() 현재 시간 출력

※ 함수의 상황이 숫자인지 문자열인지에 따라 HHMMSS 또는 'HH:MM:SS' 형식

예) select curtime(); ==> '09:10:10'

select curtime() + 0; ==> 091010

▶ DATE_ADD(날짜, INTERVAL 기준값) 날짜에서 기준값 만큼 더한다.

※ 기준값 : YEAR, MONTH, DAY, HOUR, MINUTE, SECOND

예) select date_add(now(), interval 2 day); ==> 오늘보다 2일 후의 날짜와 시간 출력.

select date_add(curdate(), interval 2 day); ==> 오늘보다 2일 후의 날짜 출력.

▶ DATE_SUB(날짜,INTERVAL 기준값) 날짜에서 기준값 만큼 뺀다.

※ 기준값 : YEAR, MONTH, DAY, HOUR, MINUTE, SECOND

select date_sub(now(),interval 2 day); ==> 오늘보다 2일 전의 날짜와 시간 출력.

select date_sub(curdate(), interval 2 day); ==> 오늘보다 2일 전의 날짜 출력.

▶ YEAR(날짜) : 날짜의 연도 출력.

select year('20000101'); select year(20000101);

select year('2000-01-01'); select year(now());

select year(curdate()); select year(date_add(now(),interval 2 year));

select year(date_sub(curdate(),interval 2 year));

▶ MONTH(날짜) : 날짜의 월 출력.

select month('20001231'); select month(20001231);

select month('2000-12-31'); select month(now());

select month(curdate()); select month(date_add(now(),interval 2 month));

select month(date_sub(curdate(),interval 2 month));

▶ MONTHNAME(날짜) : 날짜의 월을 영어로 출력.

select monthname(20021221); select monthname('20000721');

select monthname('2000-08-10'); select monthname(now());

select monthname(curdate());

select monthname(date_add(now(),interval 17 month));

select monthname(date_sub(curdate(),interval 11 month));

- ▶ DAYNAME(날짜) : 날짜의 요일일 영어로 출력.

```
select dayname(20000121);      select dayname('20010123');  
select dayname('2001-06-22');  select dayname(now());  
select dayname(curdate());  
select dayname(date_add(now(),interval 21 day));  
select dayname(date_sub(curdate(),interval 333 day));
```

- ▶ DAYOFMONTH(날짜) : 날짜의 월별 일자 출력.

```
select dayofmonth(20030112);    select dayofmonth('20011231');  
select dayofmonth('2001-12-23'); select dayofmonth(now());  
select dayofmonth(curdate());  
select dayofmonth(date_add(now(),interval 56 day));  
select dayofmonth(date_sub(curdate(),interval 33 day));
```

- ▶ DAYOFWEEK(날짜) : 날짜의 주별 일자 출력(월요일(2),화요일(3)...일요일(1))

```
select dayofweek(20011209);     select dayofweek('20001212');  
select dayofweek('2003-03-21'); select dayofweek(now());  
select dayofweek(curdate());  
select dayofweek(date_add(now(),interval 23 day));  
select dayofweek(date_sub(curdate(),interval 31 day));
```

▶ WEEKDAY(날짜) : 날짜의 주별 일자 출력(월요일(0),화요일(1)...일요일(6))

```
select weekday(20000101);      select weekday('20030223');  
select weekday('2002-10-26');  select weekday(now());  
select weekday(curdate());     select weekday(date_add(now(),interval 23 day));  
select weekday(date_sub(curdate(),interval 33 day));
```

▶ DAYOFYEAR(날짜) : 일년을 기준으로 한 날짜까지의 날 수.

```
select dayofyear(20020724);    select dayofyear('20001231');  
select dayofyear('2002-01-01'); select dayofyear(now());  
select dayofyear(curdate());   select dayofyear(date_add(curdate(),interval 44 year));  
select dayofyear(date_sub(now(),interval 25 month));  
select dayofyear(date_add(now(),interval 55 day));  
select dayofyear(date_sub(curdate(),interval 777 hour));  
select dayofyear(date_add(now(),interval 999999 minute));
```

▶ WEEK(날짜) : 일년 중 몇 번째 주.

```
select week(now());    select week(date_sub(curdate(),interval 12 month));
```

▶ FROM_DAYS(날 수)

--00년 00월 00일부터 날 수 만큼 경과한 날의 날짜 출력.

※ 날 수는 366 이상을 입력 그 이하는 무조건 '0000-00-00' 으로 출력.

--또한 9999-12-31 [from_days(3652424)] 까지의 날짜가 출력가능

--따라서 날 수는 366 이상 3652424[3652499] 이하가 되어야 한다.

```
select from_days(3652424);    select from_days('3652499');
```

▶ TO_DAYS(날짜)

--00 년 00 월 00일 부터 날짜까지의 일자 수 출력. 정확한 날짜범위는 3652424 일 수 까지
 select to_days(now()) - to_days('1970-10-10');

응용 예제2) 살아 온 날수를 이용하여 자신의 나이를 만으로 구하기

select (to_days(now())-to_days('1970-10-10'))/365;

- ▶ DATE_FORMAT(날짜, '형식') : select date_format(now(), '%Y:%M:%p'); => 2001:May:PM

타입	기호	설명	기호	설명
년도	%Y	4자리 연도	%y	2자리 연도
월	%M	긴 월 이름 (January, ...)	%m	숫자의 월 (01...12)
	%b	짧은 월 이름 (Jan, ...)	%c	숫자의 월 (1...12)
요일	%W	긴 요일 이름 (Sunday, ...)	%a	짧은 요일 이름 (Sun, ...)
일	%D	월 내에서 서수 형식의 일(1th, ...)	%d	월 내의 일자 (01...31)
	%w	숫자의 요일 (0=Sunday, ...)	%e	월 내의 일자 (1...31)
			%j	일년 중의 날수 (001...366)
시	%I	12시간제의 시 (1...12)	%k	12시간제의 시 (0...23)
	%h	12시간제의 시 (01...12)		12시간제의 시 (00...23)
	%l	12시간제의 시 (01...12)		
분	%i	숫자의 분 (00...59)		
초	%S	숫자의 초 (00...59)	%s	숫자의 초 (00...59)
시간	%r	12시간제의 시간 (hh:mm:ss AM 또는 PM)	%T	24시간제의 시간 (hh:mm:ss)
주	%U	일요일을 기준으로 한 주 (0...52)	%u	월요일을 기준으로 한 주 (0...52)
기타	%%	문자 '%'	%p	AM 또는 PM

5) 논리관련함수

- ▶ IF(논리식, 참일 때 값, 거짓일 때 값)

논리식이 참이면 참일 때 값을 출력하고 논리식이 거짓이면 거짓일 때 출력한다.

```
Select pno,pname,pay,if(pay>=1500,'good','poor' as result from personal;
```

- ▶ IFNULL(값1,값2)

값1이 NULL 이면 값2로 대체하고 그렇지 않으면 값1을 출력

```
select pno, pname, pay, pay + ifnull(bonus,0) from personal;
```

6) 그밖의 함수

- ▶ limit select pname from personal limit 5;

- ▶ select DATABASE() : 현재의 데이터베이스 이름을 출력한다.

- ▶ PASSWORD('문자열') : 문자열을 암호화한다.

- ▶ FORMAT(숫자,소수이하자리수) : 숫자를 #,###,###.## 형식으로 출력

--임의의 소수점자리수를 생성한다./소숫점을 필요한 만큼 취한다.

--소숫점을 만들어 같은 길이로 불러와서 소숫점을 버리고 출력하는 등에 응용

```
select format(123,5);          select format(123.12345600123,9);  select format(123.123,-3);
```

※ 소숫점이하자리수가 0 이나 음수값은 해당이 안됨

연습

업무가 clerk이나 manager가 아닌 사원, 입사일이 1991년 이후이며 급여가 2000이사인 사원

업무가 manager이거나 sales, 사원의 이름 첫 글자가 A~S인 사원중에서 이른 순으로 정렬

5. 테이블 조인

1) 개념

두 개 이상의 테이블에서 공동 컬럼을 이용하여 조회, parent child관계에서 사용

```
Select pname, personal.dno, pay, dname from personal, division where  
personal.dno = division.dno;
```

2) 카티션 곱

```
select pname, dno, dname from personal, division;
```

3) 내부조인

select 필드명,... from table1, table2 where 연결조건;

```
select pname, p.dno, pay, dname from personal p, division d where p.dno=d.dno;
```

4) 외부조인

```
select pname, p.dno, pay, dname from personal p left outer join division d on  
p.dno = d.dno;
```

```
select pname, p.dno, pay, dname from personal p right outer join division d on  
p.dno = d.dno;
```

5) 셀프조인

테이블 한 개를 다른 이름으로 join시켜 사용

```
select w.pno, w.pname, w.pay, m.pname from personal w, personal m where  
w.manager = m.pno;
```


6. SUB QUERY

1) 서브쿼리

서브쿼리는 다른 SELECT 문의 검색 조건에 포함된 하나의 SELECT 문입니다. 내부 서브쿼리가 먼저 해결되고 쿼리에 대한 응답은 외부 SELECT 문으로 전달되는데, 이 외부 SELECT 문을 사용하여 마지막으로 최종 결과를 표시합니다

```
SELECT pname, MAX(pay) from personal;
```

원하는 결과를 이끌어낼 방법이 없는 특정한 경우에 서브쿼리를 이용할 수 있습니다. 회사에서 가장 높은 급여를 받는 사람을 알고 싶다면, 예제와 같은 SELECT 문을 쓸 수 있을 것입니다. personal에서 가장 높은 급여를 받는 사람의 이름을 알아보려고 합니다. 그런데 이 SELECT 문에서 무언가 잘못되었음을 알게 됩니다. 세부 컬럼과 집계 함수를 혼합한 것입니다. 이 SELECT 문이 작동하도록 하려면 GROUP BY 절이 필요합니다.

```
SELECT pname MAX(pay) FROM personal GROUP BY pname;
```

GROUP BY 절을 추가하자 SELECT 문이 작동합니다. 그러나 여전히 찾으려는 답을 얻지는 못한 상태입니다. 간단한 기본 쿼리 구조를 사용하여 올바른 답을 찾는 유일한 방법은 두 개의 쿼리를 코딩하는 것입니다. 첫 번째 쿼리에서는 가장 높은 급여 값을 얻습니다. 다음 두 번째 쿼리에 이 값을 사용하여 이 급여를 받는 사람의 이름을 찾을 수 있습니다.

```
SELECT pname, pay from personal WHERE SALARY = (SELECT MAX(pay) FROM personal);
```

원하는 답을 얻는 좀더 간단한 방법은 서브쿼리를 사용하여 가장 높은 급여액을 찾고, 그 금액을 넘겨주어 외부 쿼리를 해결하는 데에 사용하는 것입니다.

여기서 몇 개의 구문 규칙에 유의하십시오. 서브쿼리는 WHERE 절 안의 조건 오른쪽에 있으며 괄호로 묶여 있습니다. 서브쿼리의 SELECT 절에서는 단 하나의 컬럼만 지정할 수 있습니다. 또한 서브쿼리의 결과는 표시되지 않고 외부 쿼리로 전달된다는 점에 유의하십시오.

SELECT NAME, SALARY FROM STAFF WHERE SALARY >= (SELECT AVG(SALARY)
FROM STAFF)

외부 쿼리로 전달되는 값은 단 하나뿐 : =, <, <=, > 또는 >= 연산자 사용

사원번호가 1121인 사원보다 많은 급여를 받는 사람

select * from personal where pay > (select pay from personal where pno=1121);

martine과 같은 업무에 종사하는 사람, 단 마틴은 제외

select * from personal where pname != 'martine' and job = (select job from
personal where pame='martine');

sales부서에 근무하는 사람

select * from personal where dno=(select dno from division where dname='sales');

평균급여보다 많이 받는 사람

select * from personal where pay > (select avg(pay) from personal);

2) IN 을 이용한 서브 쿼리

select * from personal where dno in (select dno from division where
position='seoul');

서브쿼리는 여러 부서 번호를 반환합니다. 연산자는 IN으로 바뀌어 다중 값을
수용해야 합니다. 그렇지 않으면 오류를 발생하게 됩니다. (IN을 대체할 수 있는
것 = ANY, some)

3) 여러 개의 값을 이용하는 서브 쿼리 : ALL

SELECT AVG(pay) FROM personal GROUP BY dno

모든 부서의 부서별 평균 급여보다 높은 급여를 받는 모든 직원의 이름을 목록으로 만들려고 합니다. 우선, 각 부서 별로 직원의 평균 급여를 찾아야 합니다. 그렇게 하려면 GROUP BY 절을 사용해야 합니다. 외부 쿼리에 사용할 다수의 결과를 얻습니다.

서브쿼리의 결과와 각 직원의 급여를 비교하십시오. 구하는 결과는 부서의 평균 급여보다 높은 급여입니다. 이런 결과를 얻으려면 서브쿼리 전에 '> ALL'을 사용합니다.

Select * from personal where pay > all (select avg(pay) from personal group by dno);

4) 서브쿼리 예

어떤 부서의 평균급여보다 높은 급여를 받는 직원

Select * from personal where pay > any (select avg(pay) from personal group by dno);

같은 부서의 평균급여보다 많이 받는 사람

select * from personal p1 where pay > (select avg(pay) from personal p2 where p1.dno = p2.dno);

급여가 가장 적은 사람

select * from personal where pay= (select min(pay) from personal);

입사일이 가장 빠른 사람

select * from personal where startdate = (select min(startdate) from personal);

각 부서의 평균 급여중에서 가장 많은 급여를 받는 부서의 번호와 급여 출력

select dno, avg(pay) from personal group by dno having avg(pay) >= all (select avg(pay) from personal group by dno);

5) KEY POINTS

- . 서브쿼리는 WHERE 또는 HAVING 절에 포함됩니다.
- . 서브쿼리는 다른 서브쿼리 내에 중첩될 수 있습니다.
- . 서브쿼리는 검색 조건의 오른쪽에 있으며 괄호로 묶여 있습니다.
- . 서브쿼리는 단일 값 또는 다중 값을 반환합니다. 이 서브쿼리는 SELECT 절에 지정된 단 하나의 컬럼만 가질 수 있습니다.
- . 서브쿼리가 반환한 값의 수는 외부 SELECT 절에서 사용되는 연산자를 결정합니다.

6) 중첩 서브쿼리

```
SELECT NAME FROM STAFF WHERE DEPT IN (SELECT DEPT FROM STAFF  
WHERE NAME LIKE '%S' AND ID IN (SELECT SALESREPNO FROM SALES  
WHERE PRODNO > 100))
```

중첩 서브쿼리는 다른 서브쿼리 내에 포함된 것입니다.

7. 기타 SQL

1) 서브쿼리 활용

- . create table salesman as select * from personal where job='salesman';
- . create table sawon as select pno, pname, pay, dno from personal;
- . insert into sawon2 as select pno, pname, pay from personal where dno=20;
- . update personal set job='salesman' where dno = (select dno from division where dname='sales');
- . delete from sawon where dno – (select dno from division where position = 'seoul');

2)TCL(transaction Control Language)

- . Transaction : DML명령을 하나의 처리단위로 만들어 놓은 것
- . Transaction의 시작 : DML명령사용
- . Transaction 적용 : Commit
- . Transaction 취소 : rollback

자동 commit

- . DML, DCL 명령사용, . SQL 정상종료

자동 Rollback

- . SQL 비정상 종료

8. Data BACKUP 및 복구

1) DataBase

```
]# mysqldump -u 계정 -h 서버 -p DB명 > 파일명
```

```
]# mysql -u 계정 -h 서버 -p DB명 < 파일명
```

2) Table

```
mysqldump -u root -pmysql test personal < a.txt
```

```
mysql -u root -pmysql test < a.txt
```

```
mysql> lock tables personal read;
```

```
set sql_safe_updates=0;
```

```
mysql> select * into outfile '파일네임' from personal;
```

```
mysql> load data local infile '파일네임' replace into table personal;
```

```
load data local infile '파일네임' into table personal;
```

```
LOAD DATA LOCAL INFILE 'item.csv' into TABLE ITEM FIELDS TERMINATED BY ','
```

```
LINES TERMINATED BY '\r\n';
```

```
mysql> unlock tables;
```

3) DB내용

/usr/local/mysql/var : db가 디렉토리별로 되어 있음

4) SQL계정삭제

```
mysql> delete from user where user = '계정';
```

```
mysql> flush privileges;
```

권한 재적용 mysql db에서 권한 재적용할 때에 사용하는 명령어

Mysql procedure

```
DROP PROCEDURE IF EXISTS division_insert;
delimiter //
create procedure division_insert (
    vjno int, vname varchar(20),
    vphone varchar(20), vposition varchar(20))
begin
    insert into division
        values(vjno,vname,vphone,vposition);
end;
//
delimiter ;
실행
sql> call division_insert(50,'대박','010-1111-1111','서울');
```