Mysql 2

Index

11. GRUOP BY HAVING

GROUP BY는 여러개의 동일한 데이터를 가지는 특정 컬럼을 합쳐주는 역할을 하는 명령입니다.

SQL에는 아래와 같은 그룹함수가 있습니다.

COUNT, MAX, MIN, AVG, VAR_SAMP, STDDEV

world 데이터 베이스로 이동

world 데이터 베이스는 city, country, countrylanguage 테이블이 있는 데이터 베이스 입니다.

sql> use world

COUNT

city 테이블의 CountryCode를 묶고 각 코드마다 몇개의 데이터가 있는지 확인

```
sql> SELECT CountryCode, COUNT(CountryCode)
        FROM city
        GROUP BY CountryCode
```

countrylanguage 테이블에서 전체 언어가 몇개 있는지 구하시오.

DISTINCT 중복을 제거해주는 문법

```
sql > SELECT COUNT(DISTINCT(Language)) as language_count
         FROM countrylanguage
```

MAX

대륙별 인구수와 GNP 최대 값을 조회

```
sql> SELECT continent, MAX(Population) as Population, MAX(GNP) as GNP
        FROM country
        GROUP BY continent
```

MIN

대륙별 인구수와 GNP 최소 값을 조회 (GNP와 인구수가 0이 아닌 데이터 중에서)

```
sql> SELECT continent, MIN(Population) as Population, MIN(GNP) as GNP
     FROM country
     WHERE GNP != 0 AND Population != 0
     GROUP BY continent
```

SUM

대륙별 총 인구수와 총 GNP

```
sql> SELECT continent, SUM(Population) as Population, SUM(GNP) as GNP
     FROM country
     WHERE GNP != 0 AND Population != 0
     GROUP BY continent
```

AVG

대륙별 평균 인구수와 평균 GNP 결과를 인구수로 내림차순 정렬

```
sql> SELECT continent, AVG(Population) as Population, AVG(GNP) as GNP
     FROM country
     WHERE GNP != 0 AND Population != 0
     GROUP BY continent
     ORDER BY Population DESC
```

HAVING

GROUP BY에서 반환되는 결과에 조건을 줄수 있습니다.

대륙별 전체인구를 구하고 5억이상인 대륙만 조회

```
sql> SELECT continent, SUM(Population) as Population
        FROM country
        GROUP BY continent
        HAVING Population > 500000000
```

대륙별 평균 인구수, 평균 GNP, 1인당 GNP한 결과를 1인당 GNP가 0.01 이상인 데이터를 조회하고 1인당 GNP를 내림차순으로 정렬

```
sql> SELECT continent, AVG(Population) as Population, AVG(GNP) as GNP,
                AVG(GNP) / AVG(Population) * 1000 as AVG
        FROM country
        WHERE GNP != 0 AND Population != 0
        GROUP BY continent
        HAVING AVG > 0.01
        ORDER BY AVG DESC
```

WITH ROLLUP

sakila

고객과 스탭별 매출과 고객별 매출의 총합을 출력

```
sql> SELECT customer_id, staff_id, SUM(amount) as amount
        FROM payment
        GROUP BY customer_id, staff_id
        WITH ROLLUP
```

12. CREATE USE ALTER DROP

CREATE

```
# 데이터 베이스 생성
CREATE DATABASE <database_name>;

# test 데이터 베이스 생성
CREATE DATABASE test;
```

USE

```
# test 데이터 베이스 선택
USE test;

# 현재 데이터 베이스 확인
SELECT DATABASE()

# 테이블 생성
CREATE TABLE <table_name> (
        column_name_1 column_data_type_1 column_constraint_1,
        column_name_2 column_data_type_2 column_constraint_2,
        ...
)

# 제약조건이 없는 user1 테이블 생성
CREATE TABLE user1(
```

```
    user_id INT,
    name Varchar(20),
    email Varchar(30),
    age INT(3),
    rdate DATE
)

# 제약조건이 있는 user2 테이블 생성
CREATE TABLE user2(
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    name Varchar(20) NOT NULL,
    email Varchar(30) UNIQUE NOT NULL,
    age INT(3) DEFAULT '30',
    rdate TIMESTAMP
)
```

ALTER

Database

```
# 사용중인 데이터베이스의 인코딩 방식 확인
SHOW VARIABLES LIKE "character_set_database"

# test 데이터 베이스의 문자열 인코딩을 utf8으로 변경
ALTER DATABASE world CHARACTER SET = ascii
ALTER DATABASE world CHARACTER SET = utf8

# 사용중인 데이터베이스의 인코딩 방식 확인
SHOW VARIABLES LIKE "character_set_database"
```

Table

ALTER를 이용하여 Table의 컬럼을 추가하거나 삭제하거나 수정할수 있습니다.

ADD

user2 테이블에 TEXT 데이터 타입을 갖는 tmp 컬럼을 추가

ALTER TABLE user2 ADD tmp TEXT

MODIFY

user2 테이블에 INT 데이터 타입을 갖는 tmp 컬럼으로 수정

ALTER TABLE user2 MODIFY COLUMN tmp INT

DROP
user2 테이블의 tmp 컬럼을 삭제

ALTER TABLE user2 DROP tmp

DROP

DATABASE

tmp 데이터 베이스 생성

CREATE DATABASE tmp;

SHOW DATABASES;

tmp 데이터 베이스 삭제

DROP DATABASE tmp;

SHOW DATABASES;

TABLE

tmp 데이터 베이스 생성

CREATE DATABASE tmp;

tmp 데이터 베이스 선택

USE tmp;

tmp 테이블 생성

CREATE TABLE tmp(id INT);

tmp 테이블 삭제

DROP TABLE tmp;

13. DATA TYPE

데이터 베이스의 테이블을 생성할때 각 컬럼은 데이터 타입을 가집니다.

reference : https://dev.mysql.com/doc/refman/5.7/en/data-types.html

Numberic

reference : https://dev.mysql.com/doc/refman/5.7/en/numeric-types.html

정수 타입 (integer types)

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^63	0	2^63-1	2^64-1

실수 (floating-point types)

소수점을 나타내기 위한 데이터 타입으로 아래의 두가지 데이터 타입이 있습니다. 두가지의 데이터 타입은 데이터 저장공간의 차이가 있습니다.

FLOAT (4byte), DOUBLE (8byte)

또한 아래와 같이 고정 소수점 타입으로도 사용이 가능합니다.

FLOAT(M,D), DOUBLE(M,D)

Date & Time

reference : https://dev.mysql.com/doc/refman/5.7/en/date-and-time-types.html

DATE

DATE는 날짜를 저장하는 데이터 타입이며, 기본 포멧은 "년-월-일" 입니다.

DATETIME

DATETIME은 날짜와 시간을 저장하는 데이터 타입이며, 기본 포멧은 "년-월-일 시:분:초" 입니다.

TIMESTAME

TIMESTAME는 날짜와 시간을 저장하는 데이터 타입이며, DATETIME과 다른점은 날짜를 입력하지 않으면 현재 날짜와 시간을 자동으로 저장할수 있는 특징이 있습니다.

TIME

TIME은 시간을 저장하는 데이터 타입이며, 기본 포멧은 "시:분:초" 입니다.

YEAR

YEAR는 연도를 저장할수 있는 데이터 타입입니다.

YEAR(2)는 2자리의 연도를 저장할수 있으며 YEAR(4)는 4자리의 연도를 저장할수 있습니다.

String

reference

https://dev.mysql.com/doc/refman/5.7/en/string-types.html

CHAR & VARCHAR

Value	CHAR(4)	Storage Required	VARCHAR(4)	Storage Required
''	' '	4 bytes	''	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

TEXT

CHAR와 VARCHAR는 대체로 크기가 작은 문자열을 저장할때 사용되며 크기가 큰 문자열을 저장할 때는 TEXT를 사용합니다. TEXT의 타입에 따라서 아래와 같이 크기를 가집니다.

```
      Type | Maximum length
-----------+---------------------------------------
  TINYTEXT |           255 (2^8-1) bytes
      TEXT |        65,535 (2^16-1) bytes = 64 KiB
MEDIUMTEXT |    16,777,215 (2^24-1) bytes = 16 MiB
  LONGTEXT | 4,294,967,295 (2^32-1) bytes =  4 GiB
```

14. Constraint : 제약조건

데이터 베이스의 테이블을 생성할때 각 컬럼은 각각의 제약조건을 갖습니다.

NOT NULL

NULL 값 (비어있는 값)을 저장할수 없습니다.

UNIQUE

같은 값을 저장할수 없습니다.

PRIMARY KEY

NOT NULL과 UNIQUE 의 제약조건을 동시에 만족해야 합니다. 그러므로 컬럼에 비어 있는 값과 동일한 값을 저장할수 없습니다. 하나의 테이블에 하나의 컬럼만 조건을 설정할수 있습니다.

FOREIGN KEY

다른 테이블과 연결되는 값이 저장됩니다.

DEFAULT

데이터를 저장할때 해당 컬럼에 별도의 저장값이 없으면 DEFAULT로 설정된 값이 저장됩니다.

AUTO_INCREMENT

주로 테이블의 PRIMARY KEY 데이터를 저장할때 자동으로 숫자를 1씩 증가시켜 주는 기능으로 사용합니다.

15. INSERT

테이블 이름 뒤에 오는 컬럼이름은 생략이 가능하며 대신에 VALUES 뒤에 value 값이 순서대로 와야 합니다.

INSERT INTO <table_name>(<column_name_1>, <column_name_2>, ...)

VALUES(<value_1>, <value_2>, …)

test 데이터 베이스 선택

sql> USE test;

user1 테이블에 user_id, name, email, age, rdate를 입력

```
INSERT INTO user1(user_id, name, email, age, rdate)
VALUES (1, "jin", "pdj@gmail.com", 30, now()),
(2, "peter", "peter@daum.net", 33, '2017-02-20'),
(3, "alice", "alice@naver.com", 23, '2018-01-05'),
(4, "po", "po@gmail.com", 43, '2002-09-16'),
(5, "andy", "andy@gmail.com", 17, '2016-04-28'),
(6, "jin", "jin1224@gmail.com", 33, '2013-09-02');

# city_2 테이블 생성
CREATE TABLE city_2 (
        Name VARCHAR(50),
        CountryCode CHAR(3),
        District VARCHAR(50),
        Population INT
)

# select 절에서 나온 결과데이터를 Insert
INSERT INTO city_2
SELECT Name, CountryCode, District, Population
FROM city
WHERE Population > 8000000;
```

16. UPDATE SET

업데이트시에는 항상 select-where로 변경할 데이터를 확인하고 update 해줘야 실수를 줄일수 있습니다. 또한 limit도 함께 사용해주면 좋습니다.

```
UPDATE <table_name>
SET <column_name_1> = <value_1>, <column_name_2> = <value_2>,
```

WHERE <condition>

jin 이름을 가지고 있는 사람의 나이를 20, 이메일을 pdj@daum.net으로 변경

sql> UPDATE user1

SET age=20, email="pdj@daum.net"

WHERE name="jin"

17. DELETE, DROP, TRUNCATE

조건을 설정하여 데이터를 삭제할수 있습니다.

DELETE FROM <table_name>

WHERE <condition>

2016-01-01 이전 데이터 삭제 (DML)

sql> DELETE FROM user1

WHERE rdate < "2016-01-01"

테이블 구조를 남기고 모든 데이터를 삭제 (DLL)

sql> TRUNCATE FROM user1

테이블 전체를 모두 삭제 (DLL)

sql> DROP FROM user1