DML : Data Manipulation Language

Create Tables to Store Data

EXAMPLE

Table Name: S_DEPT

Column Name	deptno	dname	loc
Key Type	PK		
Nulls/Unique			
Datatype	INT	VARCHAR	VARCHAR
Maximum Length		14	13
Sample data	10	ACCOUNTING	NEW YORK
	20	RESEARCH	DALLAS
	30	SALES	CHICAGO
	40	OPERATIONS	BOSTON

```
CREATE TABLE S_DEPT
      ( deptno          INT PRIMARY KEY,
        dname           VARCHAR(14),
        loc             VARCHAR(13));
```

Create Tables to Store Data

EXAMPLE

Table Name: S_DEPT

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

Create Tables to Store Data

EXAMPLE

Table Name: S_EMP

Column Name	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
Key Type	PK			FK1				FK2
Nulls/Unique	NN, U	NN						
FK Ref Table				S_EMP				S_DEPT
FK Ref Column				EMPNO				DEPTNO
Datatype	INT	VARCHAR	VARCHAR	INT	DATE	INT	INT	INT
MaxLength		10	9					
Sample data	7839	KING	PRESIDENT	null	1981-11-17	5000	null	10
	7566	JONES	MANAGER	7839	1981-02-04	2975	null	20
	7902	FORD	ANALYST	7566	1981-03-12	3000	null	20

Create Tables to Store Data

```
SQL> create table S_EMP
( empno INT NOT NULL,
  ename VARCHAR (10) NOT NULL,
  job VARCHAR (9),
  mgr INT,
  hiredate DATE,
  sal INT,
  comm INT,
  deptno INT,
  constraint S_EMP_id_pk PRIMARY KEY (empno),
  constraint S_EMP_mgr_fk FOREIGN KEY(mgr) REFERENCES S_EMP(empno),
  constraint S_EMP_deptno_fk FOREIGN KEY(deptno) REFERENCES
S_DEPT(deptno));
```

Create Tables to Store Data

EXAMPLE

Table Name: S_EMP

```
INSERT INTO S_EMP VALUES (7839,'KING','PRESIDENT',NULL,'81-11-17',5000,NULL,10);
INSERT INTO S_EMP VALUES (7698,'BLAKE','MANAGER',7839,'81-05-01',2850,NULL,30);
INSERT INTO S_EMP VALUES (7782,'CLARK','MANAGER',7839,'81-05-09',2450,NULL,10);
INSERT INTO S_EMP VALUES (7566,'JONES','MANAGER',7839,'81-04-01',2975,NULL,20);
INSERT INTO S_EMP VALUES (7654,'MARTIN','SALESMAN',7698,'81-09-10',1250,1400,30);
INSERT INTO S_EMP VALUES (7499,'ALLEN','SALESMAN',7698,'81-02-11',1600,300,30);
INSERT INTO S_EMP VALUES (7844,'TURNER','SALESMAN',7698,'81-08-21',1500,0,30);
INSERT INTO S_EMP VALUES (7900,'JAMES','CLERK',7698,'81-12-11',950,NULL,30);
INSERT INTO S_EMP VALUES (7521,'WARD','SALESMAN',7698,'81-02-23',1250,500,30);
INSERT INTO S_EMP VALUES (7902,'FORD','ANALYST',7566,'81-12-11',3000,NULL,20);
INSERT INTO S_EMP VALUES (7369,'SMITH','CLERK',7902,'80-12-09',800,NULL,20);
INSERT INTO S_EMP VALUES (7788,'SCOTT','ANALYST',7566,'82-12-22',3000,NULL,20);
INSERT INTO S_EMP VALUES (7876,'ADAMS','CLERK',7788,'83-01-15',1100,NULL,20);
INSERT INTO S_EMP VALUES (7934,'MILLER','CLERK',7782,'82-01-11',1300,NULL,10);
```

INSERT DATA

INSERT 문을 이용해 테이블에 새로운 행을 추가할 수 있다.

Syntax

INSERT INTO *table* [*column*] VALUES (*value, value*…)

where	*table*	테이블 이름
	column	칼럼 이름
	value	열에 해당하는 값

INSERT DATA

INSERT 절에 칼럼을 나열하지 않고 테이블의 모든 칼럼에 대한 값을 삽입

Example

```
DESC S_DEPT;


+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| deptno | int         | NO   | PRI | NULL    |       |
| dname  | varchar(14) | YES  |     | NULL    |       |
| loc    | varchar(13) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

S_DEPT 테이블에 **'HR'** 부서 정보 입력

```
INSERT INTO S_DEPT
VALUES (50, 'HR', 'SEOUL');

SELECT * FROM S_DEPT;
+--------+------------+----------+
| deptno | dname      | loc      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
|     50 | HR         | SEOUL    |
+--------+------------+----------+
```

INSERT DATA

INSERT 절에 칼럼을 나열하고 테이블의 모든 칼럼에 대한 값을 삽입

Example

```
DESCRIBE S_EMP;


+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| empno    | int         | NO   | PRI | NULL    |       |
| ename    | varchar(10) | NO   |     | NULL    |       |
| job      | varchar(9)  | YES  |     | NULL    |       |
| mgr      | int         | YES  | MUL | NULL    |       |
| hiredate | date        | YES  |     | NULL    |       |
| sal      | int         | YES  |     | NULL    |       |
| comm     | int         | YES  |     | NULL    |       |
| deptno   | int         | YES  | MUL | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

INSERT DATA

S_EMP 테이블에 새로운 직원 정보 입력

Example

```
INSERT INTO S_EMP(empno, ename, job, mgr, hiredate, sal, comm, deptno)
VALUES (1234,'ALEX', 'DESIGNER', 7839, SYSDATE(), 3000, NULL, 20);

1 row inserted.

SELECT * FROM S_EMP WHERE ENAME ='ALEX';

+-------+-------+----------+------+------------+------+------+--------+
| empno | ename | job      | mgr  | hiredate   | sal  | comm | deptno |
+-------+-------+----------+------+------------+------+------+--------+
|  1234 | ALEX  | DESIGNER | 7839 | 2024-09-20 | 3000 | NULL |     20 |
+-------+-------+----------+------+------------+------+------+--------+
```

UPDATE DATA

UPDATE 문으로 기존 행 수정

Syntax

```
UPDATE table
SET VALUES (column = value)
[WHERE condition]
```

where	*table*	업데이트 할 테이블 이름
	column	업데이트 할 칼럼 이름
	value	새로운 값
	condition	조건에 맞는 행을 업데이트

UPDATE DATA

사원번호 **7634**의 부서 번호와 **ALEX**의 부서 번호와 급여 변경

Example

```
UPDATE S_EMP
       SET deptno = 20
       WHERE empno = 7634;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

UPDATE S_EMP
        SET deptno = 20, sal = 4000
        WHERE ename = 'ALEX';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
SELECT empno, ename, sal, deptno
   FROM S_EMP
   WHERE empno =7634 OR ename = 'ALEX';

+-------+-------+------+--------+
| empno | ename | sal  | deptno |
+-------+-------+------+--------+
|  1234 | ALEX  | 4000 |     20 |
|  7634 | HENRY | 1300 |     20 |
+-------+-------+------+--------+
```

UPDATE DATA

회사에서 부서 번호 **10**인 사원들에게 커미션을 **1,000**씩 지급한다.

Example

```
UPDATE S_EMP
   SET comm = 1000
   WHERE deptno = 10;


SELECT ename, comm, deptno
       FROM S_EMP
       WHERE deptno= 10;
+--------+------+--------+
| ename  | comm | deptno |
+--------+------+--------+
| CLARK  | 1000 |     10 |
| KING   | 1000 |     10 |
| MILLER | 1000 |     10 |
+--------+------+--------+
```

DELETE DATA

DELETE 문으로 기존 행 삭제

Syntax

```
DELETE FROM table
[WHERE condition]
```

where	*table*	테이블 이름
	condition	조건에 맞는 행을 삭제

DELETE DATA

S_EMP 테이블에서 ALEX 사원 삭제

Example

```
DELETE FROM S_EMP
  WHERE ename = 'ALEX';
```

```
mysql> SELECT ename
       FROM S_EMP
       WHERE ename = 'ALEX';
Empty set (0.00 sec)
```

DELETE DATA

S_EMP 테이블에서 부서 번호 **50**인 사원들 삭제

Example

```
SQL> DELETE FROM S_EMP
   WHERE deptno = 50;

2 row deleted.
```

```
SELECT ename
   FROM S_EMP
   WHERE deptno = 50;

0 row selected.
```

조건이 없이 테이블 이름만 입력한 경우, 테이블 내 모든 데이터가 삭제된다. 전체 테이블을 삭제하는 경우가 아니면 **WHERE** 절을 생략하면 안된다.

CONTROL TRANSACTIONS

- 트랜잭션(TRANSACTION) - INSERT, UPDATE, DELETE
- 데이터 조작 작업은 데이터베이스 버퍼에 영향을 준다.
- 현재 사용자는 SELECT 문으로 데이터 조작 작업의 결과를 검토할 수 있다.
- 다른 사용자는 현재 사용자에 대한 데이터 조작 작업의 결과를 볼 수 없다.
- 영향을 받은 행은 LOCK이 걸리게 되고, 다른 사용자는 행을 변경할 수 없다.
- MySQL 경우 AutoCommit이 기본 값. 이를 해제해야함.
- SELECT @@AUTOCOMMIT; #현재 AUTOCOMMIT값 확인
- SELECT AUTOCOMMIT =1; #AUTOCOMMIT 설정 /
- SELECT AUTOCOMMIT = 0; #AUTOCOMMIT 해제

Control Transaction Logic

Statement	Description
COMMIT	현재 트랜잭션을 종료하고 트랜잭션의 갱신된 내용을 데이터베이스에 반영
ROLLBACK	현재 트랜잭션을 종료하고 트랜잭션에서 갱신된 내용 모두를 취소

CONTROL TRANSACTIONS

State of the Data After COMMIT

- **COMMIT** 문을 사용하여 보류중인 모든 변경 내용**(INSERT, UPDATE, DELETE)**을 영구적으로 만든다.
- **COMMIT** 후 데이터 변경 사항이 데이터베이스 파일에 기록된다.
- 영향을 받은 행은 **LOCK**이 해제되고, 다른 사용자가 행을 변경할 수 있다.

S_DEPT 테이블에 새로운 부서 추가하고, **COMMIT**하기

Example

```
SQL> INSERT INTO S_DEPT (deptno, dname, loc)
   VALUES (60, 'LAW', 'LA');

COMMIT;

 SELECT * FROM S_DEPT;
+--------+------------+----------+
| deptno | dname      | loc      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
|     50 | HR         | SEOUL    |
|     60 | LAW        | LA       |
+--------+------------+----------+
```

CONTROL TRANSACTIONS

State of the Data After ROLLBACK

- 데이터의 변경이 취소되고, 데이터의 이전 상태가 복원
- 영향을 받은 행은 **LOCK**이 해제되고, 다른 사용자가 행을 변경할 수 있다.

CONTROL TRANSACTIONS

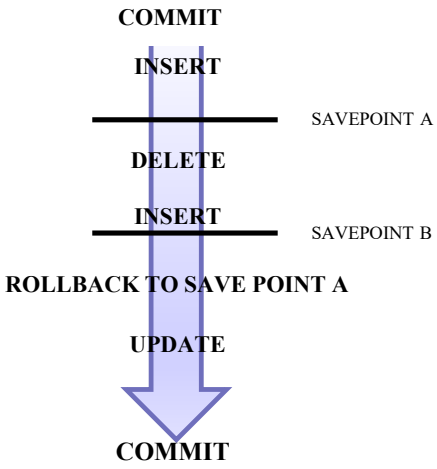
S_EMP 테이블에 이름이 'SMITH'인 사원을 실수로 삭제했다. ROLLBACK을 이용해 복원한다.

Example

```
DELETE FROM S_EMP
WHERE ENAME = 'SMITH';

5 rows deleted.

SELECT *
FROM S_EMP
WHERE ENAME = 'SMITH';
Empty set (0.00 sec)


 ROLLBACK;

SELECT *
  2 FROM S_EMP
  3 WHERE deptno = 20;

 SELECT * FROM S_EMP WHERE ENAME = 'SMITH';
+-------+-------+-------+------+------------+------+------+--------+
| empno | ename | job   | mgr  | hiredate   | sal  | comm | deptno |
+-------+-------+-------+------+------------+------+------+--------+
|  7369 | SMITH | CLERK | 7902 | 1980-12-09 |  800 | NULL |     20 |
+-------+-------+-------+------+------------+------+------+--------+
```

CONTROL TRANSACTIONS

- **SAVEPOINT** 문을 이용하여 현재 트랜잭션에 저장 지점을 만든다.
- 부분 롤백을 수행하기 위해선 저장 지점을 미리 만들어야 한다.
- 동일한 이름의 저장 지점을 설정하면 이전의 저장 지점은 삭제된다.



Alter Transaction Logic

Statement	Description
SAVEPOINT	현재 트랜잭션 내에서 저장 지점 표시
ROLLBACK TO SAVEPOINT	저장 지점이 표시된 후 보류 중인 변경 내용 삭제

CONTROL TRANSACTIONS

S_EMP 테이블에 새로운 사원의 정보를 추가하고 SAVEPOINT a, b 만들기

Example

```
INSERT INTO S_EMP(empno, ename, hiredate, sal)
   VALUES (3790, 'GOODMAN', SYSDATE(), 2000);

SAVEPOINT a;


INSERT INTO S_EMP(empno, ename, hiredate, sal)
    VALUES (3791, 'BADMAN', SYSDATE(), 1000);

SAVEPOINT b;

INSERT INTO S_EMP(empno, ename, hiredate, sal)
  VALUES (3792, 'YESMAN', SYSDATE(), 3000);
```

```
SELECT empno, ename
  FROM S_EMP;

+-------+---------+
| empno | ename   |
+-------+---------+
|  3790 | GOODMAN |
|  3791 | BADMAN  |
|  3792 | YESMAN  |
|  7369 | SMITH   |
|  7499 | ALLEN   |
|  7521 | WARD    |
|  7566 | JONES   |
|  7634 | HENRY   |
|  7654 | MARTIN  |
|  7698 | BLAKE   |
|  7782 | CLARK   |
|  7788 | SCOTT   |
|  7839 | KING    |
|  7844 | TURNER  |
|  7876 | ADAMS   |
|  7900 | JAMES   |
|  7902 | FORD    |
|  7934 | MILLER  |
+-------+---------+
```

CONTROL TRANSACTIONS

마지막 두 사원을 실행 취소하되, 첫 번째 사원은 그대로 둔다. **SAVEPOINT**를 이용해 영구적으로 변경한다.

Example

```
ROLLBACK TO SAVEPOINT a;

COMMIT;
```

```
SELECT empno, ename
   FROM S_EMP;

SELECT empno, ename
   ->    FROM S_EMP;
+-------+---------+
| empno | ename   |
+-------+---------+
|  3790 | GOODMAN |
|  7369 | SMITH   |
|  7499 | ALLEN   |
|  7521 | WARD    |
|  7566 | JONES   |
|  7634 | HENRY   |
|  7654 | MARTIN  |
|  7698 | BLAKE   |
|  7782 | CLARK   |
|  7788 | SCOTT   |
|  7839 | KING    |
|  7844 | TURNER  |
|  7876 | ADAMS   |
|  7900 | JAMES   |
|  7902 | FORD    |
|  7934 | MILLER  |
```

CONTROL TRANSACTIONS

- **COMMIT 또는 ROLLBACK을 암묵적으로 수행하는 상황에 주의해야 한다.**

Implicit Transaction Processing

Circumstance	Result
CREATE TABLE과 같은 DDL 명령어 실행	자동 COMMIT
명시적 COMMIT 또는 ROLLBACK을 하지 않고 데이터베이스 정상 종료	자동 COMMIT
비정상적인 종료 또는 시스템 오류	자동 ROLLBACK

PERFORM COMPUTATIONS WITH DATA

산술 및 SQL 함수를 사용하여 다양한 방법으로 데이터를 수정하고 조회한다.

- 숫자 및 날짜 값을 사용하여 계산 수행
- NULL 값을 포함하는 계산 처리
- 숫자, 날짜 및 문자 값을 수정
- 날짜 값을 다양한 방식으로 표시
- 행 그룹에 대한 계산 조회 및 수행

Arithmetic Operators	
+	더하기
-	빼기
*	곱하기
/	나누기

PERFORM COMPUTATIONS WITH NUMBERS

직무가 '**MANAGER**'인 사원에 대해 **$500**의 급여 인상을 계산 후 사원 이름, 급여, 인상된 급여**(NEW SALARY)**를 출력하라.

Example

```
SELECT ename, sal, sal + 500 "NEW SALARY"
   FROM S_EMP
   WHERE job = 'MANAGER'
   ORDER BY  sal + 500;
```

```
+-------+------+------------+
| ename | sal  | NEW SALARY |
+-------+------+------------+
| CLARK | 2450 |       2950 |
| BLAKE | 2850 |       3350 |
| JONES | 2975 |       3475 |
+-------+------+------------+

3 rows selected.
```

PERFORM COMPUTATIONS WITH NUMBERS

부서 번호가 **20**인 사원에 대해 급여의 **10%**를 보너스를 지급 후, 사원 이름, 급여, 보너스**(BONUS)**를 출력하라.

Example

```
SELECT ename, sal, sal * 0.1 BONUS
   FROM S_EMP
   WHERE deptno = 20
   ORDER BY sal * 0.1;
```

```
+-------+------+-------+
| ename | sal  | BONUS |
+-------+------+-------+
| SMITH |  800 |  80.0 |
| ADAMS | 1100 | 110.0 |
| HENRY | 1300 | 130.0 |
| JONES | 2975 | 297.5 |
| SCOTT | 3000 | 300.0 |
| FORD  | 3000 | 300.0 |
+-------+------+-------+
```

PERFORM COMPUTATIONS WITH NUMBERS

Rules of Precedence

산술 연산자 우선 순위는 다음과 같다.

1. 곱셈과 나누기 (*, /)
2. 덧셈과 빼기 (+, -)

Example

부서 번호 10인 사원에 대해 연간 보상을 지급한다. 이름, 급여, 연간 보상(ANNUAL COMPENSATION)을 출력하라. 연간 보상은 급여에 $100를 상여 후 12를 곱한다.

```
SELECT ename, sal, (sal + 100) * 12 "ANNUAL COMPENSATION"
   FROM S_EMP
   WHERE deptno = 10;
```

```
+--------+------+--------------------+
| ename  | sal  | ANNUAL COMPENSATION |
+--------+------+--------------------+
| CLARK  | 2450 |              30600 |
| KING   | 5000 |              61200 |
| MILLER | 1300 |              16800 |
+--------+------+--------------------+
```

PERFORM COMPUTATIONS WITH NUMBERS

ROUND

ROUND(*num1*, *num2*) – *num1*을 소수점 아래 *num2* 위치에서 반올림한 값을 반환

Example

직무가 **CLERK**인 사원에게 급여를 근무 일수로 나눈 성과급을 지급한다. 이름, 급여, 성과급(**PERFORMANCE PAY**)을 출력하라. 근무 일수는 **22**일, 성과급은 소수점 둘째 자리에서 반올림 한다.

```
SELECT ename, sal, ROUND(sal/22, 2)
   FROM S_EMP
   WHERE job = 'CLERK';
```

```
+--------+------+-----------------+
| ename  | sal  | ROUND(sal/22, 2) |
+--------+------+-----------------+
| SMITH  |  800 |           36.36 |
| ADAMS  | 1100 |           50.00 |
| JAMES  |  950 |           43.18 |
| MILLER | 1300 |           59.09 |
+--------+------+-----------------+
```

PERFORM COMPUTATIONS WITH NUMBERS

MOD

MOD(*num1, num2*) – *num1*을 *num2*로 나눈 나머지를 반환하는 함수

Example

사원 번호가 짝수인 사원의 정보를 모두 출력하라.

```
SELECT *
  FROM S_EMP
  WHERE MOD(empno, 2) = 0;
```

```
+-------+---------+----------+------+------------+------+------+--------+
| empno | ename   | job      | mgr  | hiredate   | sal  | comm | deptno |
+-------+---------+----------+------+------------+------+------+--------+
|  3790 | GOODMAN | NULL     | NULL | 2024-09-23 | 2000 | NULL |   NULL |
|  7566 | JONES   | MANAGER  | 7839 | 1981-04-01 | 2975 | NULL |     20 |
|  7634 | HENRY   | NULL     | NULL | NULL       | 1300 | NULL |     20 |
|  7654 | MARTIN  | SALESMAN | 7698 | 1981-09-10 | 1250 | 1400 |     30 |
|  7698 | BLAKE   | MANAGER  | 7839 | 1981-05-01 | 2850 | NULL |     30 |
|  7782 | CLARK   | MANAGER  | 7839 | 1981-05-09 | 2450 | 1000 |     10 |
|  7788 | SCOTT   | ANALYST  | 7566 | 1982-12-22 | 3000 | NULL |     20 |
|  7844 | TURNER  | SALESMAN | 7698 | 1981-08-21 | 1500 |    0 |     30 |
|  7876 | ADAMS   | CLERK    | 7788 | 1983-01-15 | 1100 | NULL |     20 |
|  7900 | JAMES   | CLERK    | 7698 | 1981-12-11 |  950 | NULL |     30 |
|  7902 | FORD    | ANALYST  | 7566 | 1981-12-11 | 3000 | NULL |     20 |
|  7934 | MILLER  | CLERK    | 7782 | 1982-01-11 | 1300 | 1000 |     10 |
+-------+---------+----------+------+------------+------+------+--------+
```

PERFORM COMPUTATIONS WITH DATES

산술 연산자를 이용해 날짜를 계산할 수 있다.

Date Expression
date + number
date – number
date - date

Example

급여가 **2,000** 이상인 사원의 견습 기간 종료일을 확인한다. 이름, 입사 날짜, 견습 기간 종료일을 출력하라. 견습 기간은 입사 후 **90**일까지이다.

```
SELECT ename, hiredate, hiredate + 90
   FROM S_EMP
   WHERE sal >= 2000;
```

```
+---------+------------+---------------+
| ename   | hiredate   | hiredate + 90 |
+---------+------------+---------------+
| GOODMAN | 2024-09-23 |      20241013 |
| JONES   | 1981-04-01 |       19810491 |
| BLAKE   | 1981-05-01 |       19810591 |
| CLARK   | 1981-05-09 |       19810599 |
| SCOTT   | 1982-12-22 |       19821312 |
| KING    | 1981-11-17 |       19811207 |
| FORD    | 1981-12-11 |       19811301 |
+---------+------------+---------------+
```

PERFORM COMPUTATIONS WITH DATES

LAST_DAY

LAST_DAY(*date*) - 특정 일자에 해당하는 월의 마지막 일자를 표시하는 함수

Example

현재 달의 마지막 일자를 출력하라

```
SELECT LAST_DAY(SYSDATE())
   FROM dual;
```

```
+--------------------+
| LAST_DAY(SYSDATE()) |
+--------------------+
| 2024-08-24         |
+--------------------+
```

MANIPULATE CHARACTER STRINGS

CONCAT

접합 함수로 문자열을 결합 시킨다.

Example

부서 번호 **20**인 사원의 사원 번호와 이름을 합쳐 출력한다.

```
SELECT CONCAT(empno, ' ' ,ename) "ID AND EMPLOYEE"
   FROM S_EMP
   WHERE deptno = 20;
```

```
+-----------------+
| ID AND EMPLOYEE |
+-----------------+
| 7369 SMITH      |
| 7566 JONES      |
| 7634 HENRY      |
| 7788 SCOTT      |
| 7876 ADAMS      |
| 7902 FORD       |
+-----------------+
```

MANIPULATE CHARACTER STRINGS

문자 함수

NAME : Delhi Sports

Function	EXAMPLE	RESULT
UPPER	UPPER (NAME)	DELHI SPORTS
LOWER	LOWER (NAME)	delhi sports
SUBSTR	SUBSTR (NAME, 1, 4)	Delh
LENGTH	LENGTH (NAME)	12

MANIPULATE CHARACTER STRINGS

UPPER

UPPER - 모든 문자를 대문자로 변환한다.

Example

INITCAP 함수를 이용해 출력한 직원 이름을 UPPER 함수를 이용해 대문자로 변환한다.

```
SELECT UPPER(Fname), Lname NAME
  FROM EMPLOYEE;
```

```
+--------------+---------+
| UPPER(Fname) | NAME    |
+--------------+---------+
| JOHN         | Smith   |
| FRANKLIN     | Wong    |
| JOYCE        | English |
| RAMESH       | Narayan |
| JAMES        | Borg    |
| JENNIFER     | Wallace |
| AHMAD        | Jabbar  |
| ALICIA       | Zelaya  |
+--------------+---------+
```

MANIPULATE CHARACTER STRINGS

LOWER

LOWER - 모든 문자를 소문자로 변환한다.

Example

부서 번호 30인 직원 이름, 직무를 출력한다. LOWER 함수를 이용해 모두 소문자로 변환한다.

```
SELECT LOWER(ename), LOWER(job)
   FROM S_EMP
   WHERE deptno = 30;
```

```
+--------------+------------+
| LOWER(ename) | LOWER(job) |
+--------------+------------+
| allen        | salesman   |
| ward         | salesman   |
| martin       | salesman   |
| blake        | manager    |
| turner       | salesman   |
| james        | clerk      |
+--------------+------------+
```

MANIPULATE CHARACTER STRINGS

SUBSTR

SUBSTR(*char*, *m* [, *n*] – char 내 m 번째 위치로부터 n 길이의 문자열을 추출하는 함수. n이 지정되지 않으면 마지막까지 추출

Example

부서 번호 10인 직원 이름을 뒤에서 첫 번째에서 2개까지 출력한다.

```
SELECT ename, SUBSTR(ename, 1, 2)
    FROM S_EMP
   WHERE deptno = 10;
```

```
+--------+--------------------+
| ename  | SUBSTR(ename, 1, 2) |
+--------+--------------------+
| CLARK  | CL                 |
| KING   | KI                 |
| MILLER | MI                 |
+--------+--------------------+
```

MANIPULATE CHARACTER STRINGS

LENGTH

LENGTH – 문자열의 길이를 반환하는 함수

Example

부서 번호 **30**인 직원 이름과 이름의 길이를 출력하라.

```
SELECT ename, LENGTH(ename)
   FROM S_EMP
   WHERE deptno = 30;
```

```
+--------+---------------+
| ename  | LENGTH(ename) |
+--------+---------------+
| ALLEN  |             5 |
| WARD   |             4 |
| MARTIN |             6 |
| BLAKE  |             5 |
| TURNER |             6 |
| JAMES  |             5 |
+--------+---------------+
```

이름 길이가 **6**이상인 직원의 이름을 출력하라

```
SELECT ename
   FROM S_EMP WHERE LENGTH(ename) >= 6;
```

```
ENAME
----------
MARTIN
TURNER
```

PERFORM SUMMARY COMPUTATIONS

문자 함수

NAME : Delhi Sports

Function	Description
AVG	평균값
MAX	최댓값
MIN	최솟값
SUM	합
COUNT	로우의 개수를 세는 함수

MANIPULATE CHARACTER STRINGS

AVG SUM
MIN MAX

Example

직무가 SALESMAN 인 사원들의 급여 평균(AVERAGE), 최댓값(MAXIMUM), 최솟값(MINIMUM), 합(SUM)을 출력하라

```
SELECT AVG(sal) average, MAX(sal) maximum, MIN(sal) minimum, SUM(sal) sum
   FROM S_EMP
   WHERE job = 'SALESMAN';
```

```
+-----------+---------+---------+------+
| average   | maximum | minimum | sum  |
+-----------+---------+---------+------+
| 1400.0000 |    1600 |    1250 | 5600 |
+-----------+---------+---------+------+
```

MANIPULATE CHARACTER STRINGS

COUNT

Example

S_EMP 테이블에 있는 총 사원의 수를 구하라

```
SELECT COUNT(*)
   FROM S_EMP;
```

```
+----------+
| COUNT(*) |
+----------+
|       16 |
+----------+
```

커미션을 받는 사원의 수를 구하라

```
SELECT COUNT(comm) "Employees with Comm"
  FROM S_EMP;
```

```
+--------------------+
| Employees with Comm |
+--------------------+
|                  7 |
+--------------------+
```

GROUP ROWS TOGETHER

부서 번호가 30인 사원은 6명으로 6번 표시가 된다. **GROUP BY** 절을 사용하면 각 부서에 대해 한 줄로 출력할 수 있다.

```
SELECT empno, ename, deptno
    FROM S_EMP
    WHERE deptno = 30;

+-------+--------+--------+
| empno | ename  | deptno |
+-------+--------+--------+
|  7499 | ALLEN  |     30 |
|  7521 | WARD   |     30 |
|  7654 | MARTIN |     30 |
|  7698 | BLAKE  |     30 |
|  7844 | TURNER |     30 |
|  7900 | JAMES  |     30 |
+-------+--------+--------+
```

```
SELECT deptno, COUNT(*) NUBMER
    FROM S_EMP
    WHERE deptno = 30
    GROUP BY deptno;

 +--------+--------+
| deptno | NUBMER |
+--------+--------+
|     30 |      6 |
+--------+--------+
```

GROUP ROWS TOGETHER

GROUP BY 및 HAVING 절이 있는 행 그룹에 대해 요약 결과를 출력한다.

Syntax

```
SELECT column_name
FROM table_name
WHERE condition
GROUP BY group_by_expression
```

where	*group_by_expression*	그룹화가 되는 기준의 열을 지정

Example

S_EMP 테이블에 있는 직무를 출력한다.

```
SELECT job, COUNT(*) "Number"
   FROM S_EMP
   GROUP BY job;
```

```
+-----------+--------+
| job       | Number |
+-----------+--------+
| NULL      |      2 |
| CLERK     |      4 |
| SALESMAN  |      4 |
| MANAGER   |      3 |
| ANALYST   |      2 |
| PRESIDENT |      1 |
+-----------+--------+
```

GROUP ROWS TOGETHER

Example

부서 번호에 따른 직원 수 출력

```
SELECT deptno, COUNT(*) "Head Count"
   FROM S_EMP
   GROUP BY deptno;
```

```
+--------+------------+
| deptno | Head Count |
+--------+------------+
|   NULL |          1 |
|     10 |          3 |
|     20 |          6 |
|     30 |          6 |
+--------+------------+
```

GROUP BY 절 없이 정규 열과 그룹 함수를 함께 사용하면 안된다.

```
SELECT deptno, COUNT(*) "Employees Within Titles"
   FROM S_EMP;
ERROR 1140 (42000): In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 'taba.S_EMP.deptno';
this is incompatible with sql_mode=only_full_group_by
```

GROUP ROWS TOGETHER

둘 이상의 **GROUP BY** 칼럼을 나열하여 그룹 및 하위 그룹에 대한 결과를 출력한다.

Example

```
SELECT deptno, job, COUNT(*) "Employees Within Titles"
   FROM S_EMP
   WHERE deptno = 30
   GROUP BY deptno, job;
```

```
+--------+----------+-------------------------+
| deptno | job      | Employees Within Titles |
+--------+----------+-------------------------+
|     30 | SALESMAN |                       4 |
|     30 | MANAGER  |                       1 |
|     30 | CLERK    |                     1 |
+--------+----------+-------------------------+
```

```
SELECT  job, deptno, COUNT(*) "Employees Within Titles"
    FROM S_EMP
    WHERE deptno = 30
    GROUP BY job, deptno ;
```

```
+----------+--------+-------------------------+
| job      | deptno | Employees Within Titles |
+----------+--------+-------------------------+
| SALESMAN |     30 |                       4 |
| MANAGER  |     30 |                       1 |
| CLERK    |     30 |                     1 |
+----------+--------+-------------------------+
```

DISPLAY SPECIFIC GROUPS

특정 행 또는 특정 그룹 출력

Syntax

```
SELECT column_name [,column_name]
FROM table_name
WHERE condition
GROUP BY group_by_expression
HAVING condition
```

where	*condition*	지정된 조건이 참(TRUE)인 그룹만 반환

DISPLAY SPECIFIC GROUPS

직원이 세 명 이상인 부서의 급여 평균과 직원 수를 출력하라

Example

```
SELECT deptno, AVG(sal) average, COUNT(*) "Number of Employees"
    FROM S_EMP
   GROUP BY deptno
    HAVING COUNT(*) >= 3;
```

```
+--------+-----------+--------------------+
| deptno | average   | Number of Employees |
+--------+-----------+--------------------+
|     10 | 2916.6667 |                  3 |
|     20 | 2029.1667 |                  6 |
|     30 | 1566.6667 |                  6 |
+--------+-----------+--------------------+
```

DISPLAY SPECIFIC GROUPS

각 직무 별 급여의 합이 **5,000**보다 큰 직무와 급여 합을 출력하라. 급여의 합은 내림차순으로 정렬

Example

```
SELECT job, SUM(sal) sum
   FROM S_EMP
   GROUP BY job
   HAVING SUM(sal) > 5000
   ORDER BY SUM(sal) DESC;
```

```
+----------+------+
| job      | sum  |
+----------+------+
| MANAGER  | 8275 |
| ANALYST  | 6000 |
| SALESMAN | 5600 |
+----------+------+
```