RDBMS

Agenda

1. INSERT query
2. SELECT query
    - Projection
    - Computed Columns
    - Order By
    - Limit
    - Where
3. UPDATE query

Questions

- Command; vs Command (Semi-colon)
    - MySQL client commands
        - SOURCE /path/of/sql/file
        - QUIT
    - SQL commands --> sent to server
        - SELECT * FROM students;
- MySQL Storage Engine
    - MYISAM --> .myd, .myi, ...
    - Innodb --> .ibd
- Import .sql file
    - .sql file --> SQL commands
    - Login with "root".
        - CREATE DATABASE dbname1;
        - USE dbname1;
        - SOURCE /path/of/sql1
        - CREATE DATABASE dbname2;
        - USE dbname2;
        - SOURCE /path/of/sql2

SQL

INSERT query (DML)

- To insert row(s) into database.
- Syntax: INSERT INTO table_name VALUES (col1_value1, col1_value2, col1_value3, ...);
    - Strings and Date/Time --> 'enclose in single quotes'.

    - Numeric --> not in single quotes.

- Syntax: INSERT INTO table_name VALUES (col1_value1, col1_value2, col1_value3, ...), (col1_value1, col1_value2, col1_value3, ...), (col1_value1, col1_value2, col1_value3, ...), (col1_value1, col1_value2, col1_value3, ...), ...;
- Syntax: INSERT INTO table_name SELECT ... FROM another_table_name;
- cmd> mysql -u sunbeam -psunbeam classwork

```
SELECT USER(), DATABASE();

CREATE TABLE people(name VARCHAR(30), gender CHAR, birth DATE);

DESCRIBE people;

INSERT INTO people VALUES ('James Bond', 'M', '1960-02-23');

INSERT INTO people VALUES ('Superman', 'M', '1980-04-21'), ('Spiderman', 'M',
'1984-12-01'), ('Batman', 'M', '1979-05-20');

INSERT INTO people VALUES ('John Galt', 'M'); -- error

INSERT INTO people(name, gender) VALUES ('John Galt', 'M');

SELECT * FROM people;

INSERT INTO people(gender, name) VALUES ('F', 'Dagny Tagort'), ('M', 'Fransisco
Dankonia');

SELECT * FROM people;
```

```
CREATE TABLE newpeople(name VARCHAR(30), gender CHAR, birth DATE);

SELECT * FROM newpeople;

INSERT INTO newpeople SELECT * FROM people;

SELECT * FROM newpeople;

DELETE FROM newpeople;
-- delete all rows from table

SELECT * FROM newpeople;

INSERT INTO newpeople (name, gender) SELECT name, gender FROM people;

SELECT * FROM newpeople;
```

SELECT query (DQL)

- To retrieve records from database.
- Do not change anything in table/database.
- Syntax: SELECT columns FROM table_name;
- SELECT * FROM table_name;
  - '*' means all columns in order of CREATE TABLE.

```
DESCRIBE emp;

SELECT * FROM emp;

SELECT ename, job, deptno FROM emp;

SELECT job, ename, sal FROM emp;

SELECT ename, sal AS salary, comm AS commision FROM emp;
-- salary is alias for sal column, commision is alias for comm column -- only for
display in current query.
-- no changes are done in database.

SELECT ename, sal salary, comm commision FROM emp;
-- AS keyword is optional.

DESC emp;

SELECT ename `emp name`, sal `salary in $`, comm `commision in $` FROM emp;
-- using space, # or $ in alias name use `back quotes`
```

Computed Columns

- Columns calculated at Runtime (when SELECT query is executed on server).
- Also called as Derived columns, Pseudo columns or Virtual columns.

```
-- assuming PF is 5% of sal, print name, sal & PF.
SELECT ename, sal, sal * 0.05 FROM emp;

SELECT ename, sal, sal * 0.05 AS pf FROM emp;

SELECT ename, sal, sal * 0.05 AS `provident fund` FROM emp;

-- Poor: sal <= 1500.0, Middle: 2500.0 >= sal > 1500.0, Rich: sal > 2500.0
SELECT ename, sal,
CASE
WHEN sal <= 1500 THEN 'Poor'
WHEN sal > 1500 AND sal <= 2500 THEN 'Middle'
ELSE 'Rich'
END AS category
FROM emp;
```

```
-- ename, deptno + deptno 10: ACCOUNTS, 20: RESEARCH, 30: SALES, ELSE: OPERATIONS
SELECT ename, deptno,
CASE
WHEN deptno = 10 THEN 'ACCOUNTS'
WHEN deptno = 20 THEN 'RESEARCH'
WHEN deptno = 30 THEN 'SALES'
ELSE 'OPERATIONS'
END AS dname
FROM emp;
```

DISTINCT

- Get unique values/combinations.

```
SELECT job FROM emp;

SELECT DISTINCT job FROM emp;
-- unique job are printed.

SELECT deptno FROM emp;

SELECT DISTINCT deptno FROM emp;
-- unique deptno are printed.

SELECT deptno, job FROM emp;

SELECT DISTINCT deptno, job FROM emp;
-- unique combinations of deptno & job are printed.
```

LIMIT clause

- Get limited rows.
- Used to implement pagination feature in web applications.

```
SELECT * FROM emp;
-- get all records from server to client
-- heavy network traffic for huge data

SELECT * FROM emp LIMIT 3;
-- get first 3 records.

SELECT * FROM emp LIMIT 3, 3;
-- get 3 records after skipping first 3 records.

SELECT * FROM emp LIMIT 6, 3;
-- get 3 records after skipping first 6 records.
```

```
SELECT * FROM emp LIMIT 9, 3;
-- get 3 records after skipping first 9 records.

SELECT * FROM emp LIMIT -3;
-- error: number of records must be +ve value.
```

ORDER BY clause

```
SELECT * FROM emp;

SELECT * FROM emp ORDER BY ename;
-- get emp records sorted by name.

SELECT * FROM emp ORDER BY deptno;
-- get emp records sorted by deptno.

SELECT * FROM emp ORDER BY hire ASC;
-- get emp records sorted by hire date.

SELECT * FROM emp ORDER BY sal DESC;
-- get emp records sorted by sal in desc order.

SELECT ename, job, sal FROM emp ORDER BY sal;
-- get ename, job & sal sorted by sal (column name).

SELECT ename, job, sal FROM emp ORDER BY 2;
-- get ename, job & sal sorted by second column in SELECT.

SELECT ename, job, sal AS salary FROM emp ORDER BY salary;
-- get ename, job & sal sorted by sal (alias).

SELECT empno, ename, deptno, job FROM emp ORDER BY deptno, job;
-- sort by deptno first, if deptno is same then sort by job.

SELECT empno, ename, deptno, job FROM emp ORDER BY deptno, job, ename;
-- sort by deptno first, if deptno is same then sort by job & if job is also same
then sort by ename.

SELECT empno, ename, deptno, job FROM emp ORDER BY deptno DESC, job ASC, ename
DESC;
-- sort by deptno first desc, if deptno is same then sort by job asc & if job is
also same then sort by ename desc.
```

```
-- get emp with highest sal.
SELECT * FROM emp
ORDER BY sal DESC;

SELECT * FROM emp
ORDER BY sal DESC
```

```
LIMIT 1;

-- get emp hired first.
SELECT * FROM emp;

SELECT * FROM emp
ORDER BY hire;

SELECT * FROM emp
ORDER BY hire
LIMIT 1;

-- get third emp recruited in company
SELECT * FROM emp
ORDER BY hire;

SELECT * FROM emp
ORDER BY hire
LIMIT 2, 1;

-- find three emps with min sal.
SELECT * FROM emp
ORDER BY sal
LIMIT 3;
```

```
-- find three emps with min sal.
SELECT * FROM emp
LIMIT 3
ORDER BY sal;
-- error: wrong syntax, LIMIT must be after ORDER BY.

SELECT * FROM emp
ORDER BY sal
LIMIT 3;
-- correct syntax.
```

WHERE clause

- Fetch/Filter records (rows) based on some condition.
- Syntax: SELECT columns FROM table_name WHERE condition;
- Condition is written using operators
  - Relational opeators: <, >, <=, >=, =, != or <>
  - NULL related operators: <=> or IS NULL, IS NOT NULL
  - Logical operators: to combine two conditions
    - AND ~~&&~~, OR ~~||~~, NOT ~~!~~

```sql
-- show all emps
SELECT * FROM emp;

-- get all emps having sal more than 2500.
SELECT * FROM emp
WHERE sal > 2500;

-- get all emps of deptno 20
SELECT * FROM emp
WHERE deptno = 20;

-- get all emps who are not SALESMAN
SELECT * FROM emp
WHERE job <> 'SALESMAN';

SELECT * FROM emp
WHERE job != 'SALESMAN';

-- get all emps whose comm is null.
SELECT * FROM emp
WHERE comm = NULL;
-- relational operator doesn't work with NULL.

SELECT * FROM emp
WHERE comm IS NULL;

-- get all emps whose comm is not null.
SELECT * FROM emp
WHERE comm IS NOT NULL;

SELECT * FROM emp
WHERE comm <=> NULL;

-- all emps recruited in year 1982 i.e. (1982-1-1 to 1982-12-31).
SELECT * FROM emp
WHERE hire >= '1982-01-01' AND hire <= '1982-12-31';

-- all managers and analyst
SELECT * FROM emp
WHERE job = 'MANAGER' OR job = 'ANALYST';

-- all emps who are not managers and analyst
SELECT * FROM emp
WHERE NOT(job = 'MANAGER' OR job = 'ANALYST');

SELECT * FROM emp
WHERE job <> 'MANAGER' AND job <> 'ANALYST';
```

HELP

- To see help of command or functions.

```
HELP;
-- show list of client commands.

HELP SELECT;
```