STRUCTURED QUERY LANGUAGE I

Lecturer: Jailani Abdul Rahman

Overview

- Structured Query Language (SQL) is the most popular language used for relational database management system.
- For this module, we will be using SQL version called MySQL.
- During practical session, you will be asked to install MySQL server and MySQL Workbench.
- But we will be concentrating on using MySQL Command Line.
- The screenshots are generated from MySQL Command Line.

MySQL Command Line

MySQL 5.6 Command Line Client

```
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.26-log MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

Creating Database

- Before implementing a database design, it is important to create database first.

CREATE DATABASE database_name;

Give appropriate name for the database.

- Once you run the above statement, it will create a database where you can interact with it.

```
mysql> CREATE DATABASE movierating;
Query OK, 1 row affected (0.00 sec)
```

- But for the moment it is empty, imagine it as an empty container.

movierating

Currently an empty database.

Choosing Database

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| classicmodels      |
| movierating        |
| mysql              |
| performance_schema |
| sakila             |
| studentmarks       |
+--------------------+
7 rows in set (0.01 sec)
```

- It is possible for you to check for existing database.

SHOW DATABASES;

- To interact with the database, it is important to state which database we are going to interact with.

USE database_name;

Choose existing database

- Imagine you have two databases.

studentmarks movierating

- If you want to interact with movierating database.

```
mysql> USE movierating;
Database changed
```

USE movierating;

studentmarks movierating

(Now you can interact with movierating database)

Deleting Database

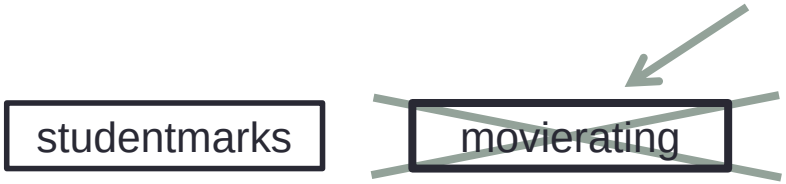
- If you no longer need the database anymore, it is possible to delete the database.

DROP DATABASE database_name;

Choose existing database

- **WARNING:** You have to be careful when using this MySQL command. It will delete the database, the tables inside the database and the records inside the database.

- Example: **DROP DATABASE** movierating;



Creating Table

- After creating the database, it is still not possible to insert data into the database. You need to create tables.

Give appropriate name for the table.

```
CREATE TABLE table_name (
    column_name1 data_type,
    column_name2 data_type,
    column_name3 data_type,
    ……..
);
```

Give appropriate name for each column

Give appropriate data type for each column

Deleting Tables

- It is also possible for you to delete the tables:

```
DROP TABLE table_name;
```

- **WARNING:** You have to be careful when using this MySQL command. It will delete the table and the records inside the database.

Show Tables

```
mysql> SHOW TABLES;
+-----------------------+
| Tables_in_movierating |
+-----------------------+
| movie                 |
| rating                |
| reviewer              |
+-----------------------+
3 rows in set (0.00 sec)
```

- To check the list of tables exists in a database:

SHOW TABLES;

- To check the list of columns, data types, etc:

DESCRIBE table_name;

```
mysql> DESCRIBE Movie;
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| mID      | int(11) | YES  |     | NULL    |       |
| title    | text    | YES  |     | NULL    |       |
| year     | int(11) | YES  |     | NULL    |       |
| director | text    | YES  |     | NULL    |       |
+----------+---------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Altering Table

- It is possible for you to Edit Table, there are three ways:

1. Adding Columns

```
ALTER TABLE table_name
ADD column_name datatype;
```

1. Deleting Columns

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

1. Editing Columns

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

Data Type

- There are many Data Types in MySQL.
- They are divided into three main types:
  1. Text
  2. Number
  3. Date/Time

Data Type - Text

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. **Note:** The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

Data Type - Number

Data type	Description
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

*The integer types have an extra option called UNSIGNED. Normally, the integer goes from an negative to positive value. Adding the UNSIGNED attribute will move that range up so it starts at zero instead of a negative number.

Data Type – Date/Time

Data type	Description
DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS **Note:** The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. **Note:** Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

*Even if DATETIME and TIMESTAMP return the same format, they work very differently. In an INSERT or UPDATE query, the TIMESTAMP automatically set itself to the current date and time. TIMESTAMP also accepts various formats, like YYYYMMDDHHMISS, YYMMDDHHMISS, YYYYMMDD, or YYMMDD.

Example Creating Table

- Create three tables Movie, Reviewer and Rating and give attributes with proper data type.

```
mysql> create table Movie(mID int, title text, year int, director text);
Query OK, 0 rows affected (0.23 sec)
```

```
mysql> create table Rating(rID int, mID int, stars int, ratingDate date);
Query OK, 0 rows affected (0.34 sec)
```

```
mysql> create table Reviewer(rID int, name text);
Query OK, 0 rows affected (0.25 sec)
```

Visualisation of **movierating** table

Movie	mID	title	year	director
Rating	rID	mID	stars	ratingDate
Reviewer	rID	name		

Inserting Data

- Once you have created all the required tables, it is now possible to insert data into the tables.
- There are four ways to insert data.
  1. Inserting single data without specifying the columns.
  2. Inserting single data with specific columns.
  3. Inserting multiple data without specifying the columns.
  4. Inserting multiple data with specific columns.

Inserting Single Data without Specifying the Columns

- The first one is inserting a single data without specifying the columns.
- You can use this if you know the order of the columns in a table.

```
INSERT INTO table_name VALUES
(value1, value2, value3, …..);
```

Example Inserting Single Data without Specifying the Columns

- From the previous tables we insert data into it.

```
mysql> INSERT INTO Movie VALUES
    -> (101, 'Gone with the Wind', 1939, 'Victor Fleming');
Query OK, 1 row affected (0.06 sec)
```

- To insert another data into the table, do another insert statement.

```
mysql> INSERT INTO Movie VALUES
    -> (102, 'Star Wars', 1977, 'George Lucas');
Query OK, 1 row affected (0.10 sec)
```

- You can do this multiple times to insert more data into the table.

Inserting Single Data with Specific Columns

- The first one is inserting a single data with specific columns.
- You can use this if you don’t know the order of the columns in a table.
- Or you can use this if you want to insert into specific columns in a table.

```
INSERT INTO table_name
(column1, column2, column3, …..)
VALUES
(value1, value2, value3, …..);
```

Example Inserting Single Data with Specific Columns

- This is the example when inserting single data with specific columns.
- In Movie table, the columns are mID, title, year and director.

```
mysql> INSERT INTO Movie (mID, title, year, director)
    -> VALUES (101, 'Gone with the Wind', 1939, 'Victor Fleming');
Query OK, 1 row affected (0.08 sec)
```

```
mysql> INSERT INTO Movie (mID, title, year, director)
    -> VALUES (102, 'Star Wars', 1977, 'George Lucas');
Query OK, 1 row affected (0.07 sec)
```

Inserting Multiple Data without Specifying the Columns.

- It is also possible for you to insert multiple data using a single INSERT statement.

```
INSERT INTO table_name VALUES
(A1, A2, A3, …..) ,
(B1, B2, B3, …..) ,
(C1, C2, C3, …..) ;
```

Example Inserting Multiple Data without Specifying the Columns.

```
mysql> INSERT INTO Movie VALUES
    -> (103, 'The Sound of Music', 1965, 'Robert Wise'),
    -> (104, 'E.T.', 1982, 'Steven Spielberg'),
    -> (105, 'Titanic', 1997, 'James Cameron');
Query OK, 3 rows affected (0.08 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

- You can do this multiple times to insert more data into the table.

Inserting Multiple Data with Specific Columns.

- It is also possible for you to insert multiple data using a single INSERT statement.

```
INSERT INTO table_name
(column1, column2, column3, …..)
VALUES
(A1, A2, A3, …..) ,
(B1, B2, B3, …..) ,
(C1, C2, C3, …..) ;
```

Example Inserting Multiple Data with Specific Columns

```
mysql> INSERT INTO Movie (mID, title, year, director)
    -> VALUES (101, 'Gone with the Wind', 1939, 'Victor Fleming'),
    -> (102, 'Star Wars', 1977, 'George Lucas');
Query OK, 2 rows affected (0.07 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

- You can do this multiple times to insert more data into the table.

Inserting Data Depending on Data Types

- Inserting Text Data Type:
  - It is required to include apostrophe to surround the text. E.g. ‘Text1’
- Inserting Number Data Type:
  - This does not require special symbol to insert number to mysql. E.g. 199
- Inserting Date/Time Type:
  - It is required to include apostrophe to surround the date/time and it should follow the format stated in Date/Time Data Type (refer to slide 12).
  - **E.g.**
  - DATE() – Format ‘YYYY-MM-DD’ ► ‘1999-01-31’
  - DATETIME() – Format ‘YYYY-MM-DD HH:MI:SS’ ► ‘1999-01-31 15:45:23’

Retrieving Data

- With the tables populated with data. Now it is possible to retrieve them.
- NOTE: Retrieving data will not change the data inside the tables. It is just simply get the data.
- To retrieve data in MySQL database, we use the SELECT statement.
- There are four basic way to retrieve data:

1. Retrieve data from a single table with all columns.
2. Retrieve data from a single table with specific columns.
3. Retrieve data with conditions.
4. Retrieve data from multiple tables.

Retrieving Data from a single table with all columns

- In MySQL, there is a symbol that can be use to select all columns in the table.
- The symbol is asterisk (*).
- The following is how you use the symbol.

SELECT * **FROM** table_name;

Table Movie columns mID, title, year and director.

```
mysql> SELECT * FROM Movie;
+------+-------------------------+------+------------------+
| mID  | title                   | year | director         |
+------+-------------------------+------+------------------+
|  101 | Gone with the Wind      | 1939 | Victor Fleming   |
|  102 | Star Wars               | 1977 | George Lucas     |
|  103 | The Sound of Music      | 1965 | Robert Wise      |
|  104 | E.T.                    | 1982 | Steven Spielberg |
|  105 | Titanic                 | 1997 | James Cameron    |
|  106 | Snow White              | 1937 | NULL             |
|  107 | Avatar                  | 2009 | James Cameron    |
|  108 | Raiders of the Lost Ark | 1981 | Steven Spielberg |
+------+-------------------------+------+------------------+
8 rows in set (0.00 sec)
```

Retrieve Data from a single table with specific columns

- Another way to retrieve data from a single table is to get specific columns.

SELECT column_name1, column_name2, …..
FROM table_name;

Table Movie columns mID, title, year and director.

But we're only retrieving mID and title columns.

```
mysql> SELECT mID, title FROM Movie;
+------+-------------------------+
| mID  | title                   |
+------+-------------------------+
|  101 | Gone with the Wind      |
|  102 | Star Wars               |
|  103 | The Sound of Music      |
|  104 | E.T.                    |
|  105 | Titanic                 |
|  106 | Snow White              |
|  107 | Avatar                  |
|  108 | Raiders of the Lost Ark |
+------+-------------------------+
8 rows in set (0.00 sec)
```

Retrieving Data with conditions

- In previous slides, you retrieve all the data in the table.
- But what if you need to retrieve a specific data in the table.
- You need to add conditions into the SELECT statement.
- It will retrieve data that satisfies the conditions.
- To do that add WHERE after the table name followed by the conditions.

```
SELECT * FROM table_name
WHERE conditions;
```

Conditions

- The conditions in the select statement will filter the data.
- The way to write the condition depends on the Data Type:
  1. Conditions for Data Type Text
  2. Conditions for Data Type Number
  3. Conditions for Data Type Date/Time
- Another way is to add Logical Operators to the condition

Conditions for Data Type - Text

- For text we usually use the statement LIKE. In the condition,

SELECT * **FROM** table_name
WHERE column_name **LIKE** ‘Text to be searched’;

- If you’re not sure the actual text, it is possible to replace the text into Wildcard.

Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for a single character

Example: Using Wildcard %

- Wildcard % can be used to substitute for zero or more characters.

```
mysql> SELECT * FROM Movie WHERE title LIKE 's%';
+------+------------+------+--------------+
| mID  | title      | year | director     |
+------+------------+------+--------------+
|  102 | Star Wars  | 1977 | George Lucas |
|  106 | Snow White | 1937 | NULL         |
+------+------------+------+--------------+
2 rows in set (0.00 sec)
```

Any title that starts with letter ‘s’

Any title that ends with letter ‘c’

```
mysql> SELECT * FROM Movie WHERE title LIKE '%c';
+------+--------------------+------+---------------+
| mID  | title              | year | director      |
+------+--------------------+------+---------------+
|  103 | The Sound of Music | 1965 | Robert Wise   |
|  105 | Titanic            | 1997 | James Cameron |
+------+--------------------+------+---------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM Movie WHERE title LIKE 'g%d';
+------+--------------------+------+----------------+
| mID  | title              | year | director       |
+------+--------------------+------+----------------+
|  101 | Gone with the Wind | 1939 | Victor Fleming |
+------+--------------------+------+----------------+
1 row in set (0.00 sec)
```

Any title that starts with letter ‘g’ and ends with letter ‘d’

Example: Using Wildcard %

```
mysql> SELECT * FROM Movie WHERE title LIKE '%r%';
+------+-------------------------+------+------------------+
| mID  | title                   | year | director         |
+------+-------------------------+------+------------------+
|  102 | Star Wars               | 1977 | George Lucas     |
|  107 | Avatar                  | 2009 | James Cameron    |
|  108 | Raiders of the Lost Ark | 1981 | Steven Spielberg |
+------+-------------------------+------+------------------+
3 rows in set (0.00 sec)
```

Any title that has a letter ‘r’ in the middle.

You’re not sure if title star and wars has space in between.

```
mysql> SELECT * FROM Movie WHERE title LIKE 'star%wars';
+------+-----------+------+--------------+
| mID  | title     | year | director     |
+------+-----------+------+--------------+
|  102 | Star Wars | 1977 | George Lucas |
+------+-----------+------+--------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM Movie WHERE title LIKE 's%r%s';
+------+-----------+------+--------------+
| mID  | title     | year | director     |
+------+-----------+------+--------------+
|  102 | Star Wars | 1977 | George Lucas |
+------+-----------+------+--------------+
1 row in set (0.00 sec)
```

Any title that starts with letter ‘s’, contain letter ‘r’ in the middle and ends with letter ‘s’.

Example: Using Wildcard _

- Wildcard _ can be used to substitute a single character.

```
mysql> SELECT * FROM Movie WHERE title LIKE 'st_r wars';
+------+-----------+------+--------------+
| mID  | title     | year | director     |
+------+-----------+------+--------------+
|  102 | Star Wars | 1977 | George Lucas |
+------+-----------+------+--------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM Movie WHERE title LIKE 's_a_ wars';
+------+-----------+------+--------------+
| mID  | title     | year | director     |
+------+-----------+------+--------------+
|  102 | Star Wars | 1977 | George Lucas |
+------+-----------+------+--------------+
1 row in set (0.04 sec)
```

```
mysql> SELECT * FROM Movie WHERE title LIKE 's__r wars';
+------+-----------+------+--------------+
| mID  | title     | year | director     |
+------+-----------+------+--------------+
|  102 | Star Wars | 1977 | George Lucas |
+------+-----------+------+--------------+
1 row in set (0.00 sec)
```

Conditions for Data Type - Number

- For number we usually use comparator. In the condition,

```
SELECT * FROM table_name
WHERE column_name[comparator]value;
```

- [comparator] should be replaced with below:

Comparator	Description
=	Equal to
>	More than
<	Less than
>=	More than or equal to
<=	Less than or equal to
<>	Not equal to

https://dev.mysql.com/doc/refman/5.0/en/comparison-operators.html

Example: Conditions for Data Type - Number

```
mysql> SELECT * FROM Movie WHERE year=1997;
+------+---------+------+---------------+
| mID  | title   | year | director      |
+------+---------+------+---------------+
|  105 | Titanic | 1997 | James Cameron |
+------+---------+------+---------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM Movie WHERE year>1982;
+------+---------+------+---------------+
| mID  | title   | year | director      |
+------+---------+------+---------------+
|  105 | Titanic | 1997 | James Cameron |
|  107 | Avatar  | 2009 | James Cameron |
+------+---------+------+---------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM Movie WHERE year>=1982;
+------+---------+------+-----------------+
| mID  | title   | year | director        |
+------+---------+------+-----------------+
|  104 | E.T.    | 1982 | Steven Spielberg |
|  105 | Titanic | 1997 | James Cameron   |
|  107 | Avatar  | 2009 | James Cameron   |
+------+---------+------+-----------------+
3 rows in set (0.00 sec)
```

Conditions for Data Type – Date/Time

- For date/time we usually use comparator. In the condition,

SELECT * **FROM** table_name
WHERE column_name**[comparator]**'date/time according to format';

- [comparator] should be replaced with below:

Comparator	Description
=	Equal to
>	More than
<	Less than
>=	More than or equal to
<=	Less than or equal to
<>	Not equal to

https://dev.mysql.com/doc/refman/5.0/en/comparison-operators.html

Example Conditions for Data Type – Date/Time

```
mysql> SELECT * FROM Rating WHERE ratingDate='2011-01-22';
+------+------+-------+------------+
| rID  | mID  | stars | ratingDate |
+------+------+-------+------------+
|  201 |  101 |     2 | 2011-01-22 |
|  205 |  104 |     2 | 2011-01-22 |
+------+------+-------+------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM Rating WHERE ratingDate>'2011-01-22';
+------+------+-------+------------+
| rID  | mID  | stars | ratingDate |
+------+------+-------+------------+
|  201 |  101 |     4 | 2011-01-27 |
|  203 |  108 |     2 | 2011-01-30 |
|  205 |  103 |     3 | 2011-01-27 |
+------+------+-------+------------+
3 rows in set (0.00 sec)
```

Adding Logical Operator

- In MySQL, logical operators are:

Logical Operator	Description
AND	Condition1 AND Condition2
OR	Condition1 OR Condition2
NOT	NOT Condition

```
SELECT * FROM table_name
WHERE condition1 AND condition2;

SELECT * FROM table_name
WHERE condition1 OR condition2;

SELECT * FROM table_name
WHERE NOT condition;
```

Example: Logical Operator

Movie title that starts with letter ‘s’ or ‘t’

```
mysql> SELECT * FROM Movie WHERE title LIKE 's%' OR title LIKE 't%';
+------+--------------------+------+---------------+
| mID  | title              | year | director      |
+------+--------------------+------+---------------+
|  102 | Star Wars          | 1977 | George Lucas  |
|  103 | The Sound of Music | 1965 | Robert Wise   |
|  105 | Titanic            | 1997 | James Cameron |
|  106 | Snow White         | 1937 | NULL          |
+------+--------------------+------+---------------+
4 rows in set (0.00 sec)
```

Movie year between 1960 and 1980

```
mysql> SELECT * FROM Movie WHERE year>=1960 AND year<=1980;
+------+--------------------+------+--------------+
| mID  | title              | year | director     |
+------+--------------------+------+--------------+
|  102 | Star Wars          | 1977 | George Lucas |
|  103 | The Sound of Music | 1965 | Robert Wise  |
+------+--------------------+------+--------------+
2 rows in set (0.00 sec)
```

Example: Logical Operator

- Movie title that does not start with letter ‘s’

```
mysql> SELECT * FROM Movie WHERE NOT title LIKE 't%';
+------+-------------------------+------+------------------+
| mID  | title                   | year | director         |
+------+-------------------------+------+------------------+
|  101 | Gone with the Wind      | 1939 | Victor Fleming   |
|  102 | Star Wars               | 1977 | George Lucas     |
|  104 | E.T.                    | 1982 | Steven Spielberg |
|  106 | Snow White              | 1937 | NULL             |
|  107 | Avatar                  | 2009 | James Cameron    |
|  108 | Raiders of the Lost Ark | 1981 | Steven Spielberg |
+------+-------------------------+------+------------------+
6 rows in set (0.00 sec)
```

Retrieve Data from multiple tables

- It is possible to retrieve data from multiple tables.

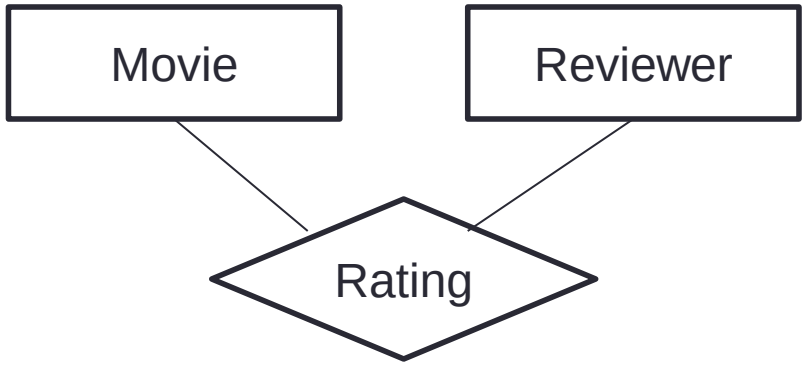
- NOTE: Make sure when selecting multiple tables, all tables are related.

```
SELECT * FROM table1, table2
WHERE table1.column=table2.column;
```

Column of table1
and table2 that are
related to each other.

Example Retrieve Data from multiple tables

```
mysql> SELECT * FROM Rating;
+------+------+-------+------------+
| rID  | mID  | stars | ratingDate |
+------+------+-------+------------+
|  201 |  101 |     2 | 2011-01-22 |
|  201 |  101 |     4 | 2011-01-27 |
|  202 |  106 |     4 | NULL       |
|  203 |  103 |     2 | 2011-01-20 |
|  203 |  108 |     4 | 2011-01-12 |
|  203 |  108 |     2 | 2011-01-30 |
|  204 |  101 |     3 | 2011-01-09 |
|  205 |  103 |     3 | 2011-01-27 |
|  205 |  104 |     2 | 2011-01-22 |
|  205 |  108 |     4 | NULL       |
|  206 |  107 |     3 | 2011-01-15 |
|  206 |  106 |     5 | 2011-01-19 |
|  207 |  107 |     5 | 2011-01-20 |
|  208 |  104 |     3 | 2011-01-02 |
+------+------+-------+------------+
14 rows in set (0.00 sec)
```



Rating table has rID for reviewer ID and mID for movie ID. But if we just retrieve table from rating table, we don't know what does rID and mID represents.

That is why we need to retrieve data from multiple tables.

Example Retrieve Data from multiple tables

In this example, **Movie** table we rename it as **mov**, **Reviewer** as **rev** and **Rating** as **rat**. If you do **mov.title**, it refer to **column title in Movie table**. To relate the tables, we compare **mov.mID=rat.mID AND rev.rID=rat.rID**

```
mysql> SELECT mov.title, rev.name, rat.stars, rat.ratingDate
    -> FROM Movie mov, Reviewer rev, Rating rat
    -> WHERE mov.mID=rat.mID AND rev.rID=rat.rID;
+-------------------------+------------------+-------+------------+
| title                   | name             | stars | ratingDate |
+-------------------------+------------------+-------+------------+
| Gone with the Wind      | Sarah Martinez   |     2 | 2011-01-22 |
| Gone with the Wind      | Sarah Martinez   |     4 | 2011-01-27 |
| Snow White              | Daniel Lewis     |     4 | NULL       |
| The Sound of Music      | Brittany Harris  |     2 | 2011-01-20 |
| Raiders of the Lost Ark | Brittany Harris  |     4 | 2011-01-12 |
| Raiders of the Lost Ark | Brittany Harris  |     2 | 2011-01-30 |
| Gone with the Wind      | Mike Anderson    |     3 | 2011-01-09 |
| The Sound of Music      | Chris Jackson    |     3 | 2011-01-27 |
| E.T.                    | Chris Jackson    |     2 | 2011-01-22 |
| Raiders of the Lost Ark | Chris Jackson    |     4 | NULL       |
| Avatar                  | Elizabeth Thomas |     3 | 2011-01-15 |
| Snow White              | Elizabeth Thomas |     5 | 2011-01-19 |
| Avatar                  | James Cameron    |     5 | 2011-01-20 |
| E.T.                    | Ashley White     |     3 | 2011-01-02 |
+-------------------------+------------------+-------+------------+
14 rows in set (0.00 sec)
```

Since we relate mID and rID, we can get the title of the movie and name of the reviewer.

Edit Data

- In MySQL, it is possible to edit the data:

```
UPDATE table_name
SET column1=value1, column2=value2, ….
WHERE conditions;
```

- WARNING: Make sure the conditions stated will retrieve the actual data you want to edit since it will change all values of the columns stated for the data (records) retrieved.

Deleting Data

- In MySQL, it is possible to delete data:

```
DELETE FROM table_name
WHERE conditions;
```

- WARNING: Make sure the conditions stated will retrieve the actual data you want to delete since it will the data (records) retrieved.