STRUCTURED QUERY LANGUAGE II

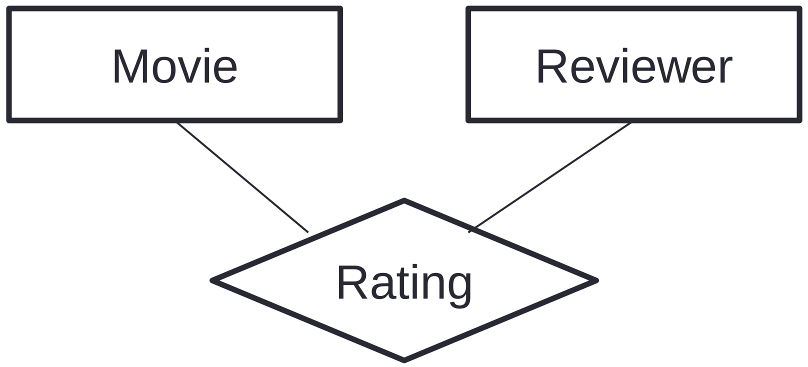
Lecturer: Jailani Abdul Rahman

Overview

- In previous lecture, you had learned the basic in MySQL.
- Now, we are going to cover how to enhance your MySQL statements.

Protecting your data

- Currently it is not possible to protect your data.
- It is still possible to have duplicate data and it is still possible to manipulate the column that has reference to another table.



- From previous lecture example: Rating table a relationship of Movie and Reviewer. In theory, inserting data into Rating with mID that does not exist in Movie table should not be valid. But since we haven’t set the rules in the table. It is still a valid insert.

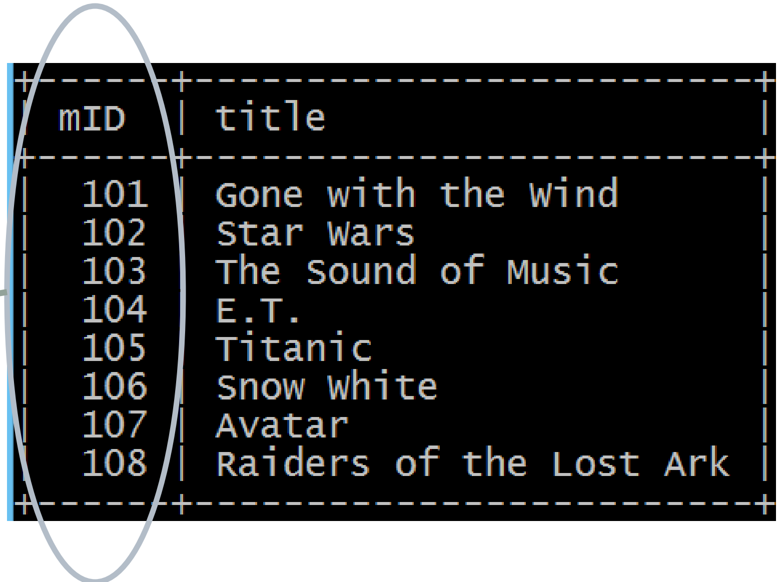
Primary Key

- A primary key is column(s) in a table which uniquely identifies each record.
- So if you set a column as Primary Key, there will be no duplicate record allowed for the column in the table.
- Example: if we set mID as Primary Key, mID will have unique record.
- When we set the column as

Primary Key, MySQL will block

the insert data if it detect duplicate.

NO DUPLICATE



```
+------+------------------------+
| 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 |
+------+------------------------+
```

Primary Key (cont.)

- To implement Primary Key into MySQL you either state it during the creation of the table or edit the table by adding primary key.

Setting PRIMARY KEY when creating new table.

```
CREATE TABLE table_name (
    column_name1 data_type,
    column_name2 data_type,
    column_name3 data_type,
PRIMARY KEY(some_column)
);
```

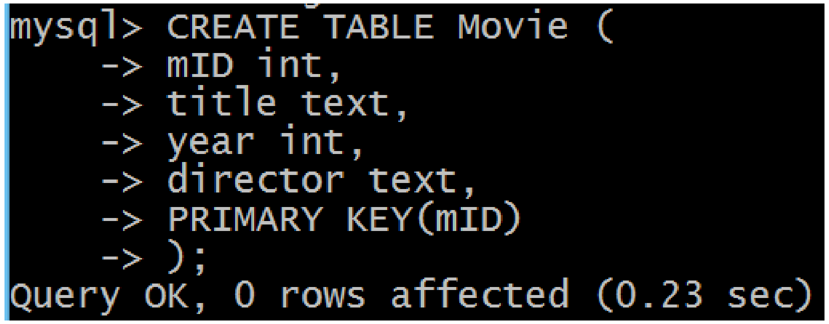
Only columns that are in the same table can be set as primary key

Setting PRIMARY KEY for an existing table.

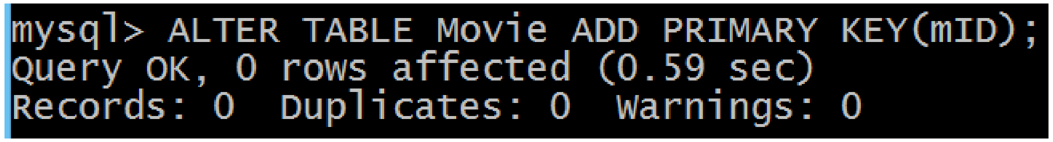
```
ALTER TABLE table_name
ADD PRIMARY KEY(some_column);
```

Example: Primary Key

- Setting mID as Primary Key

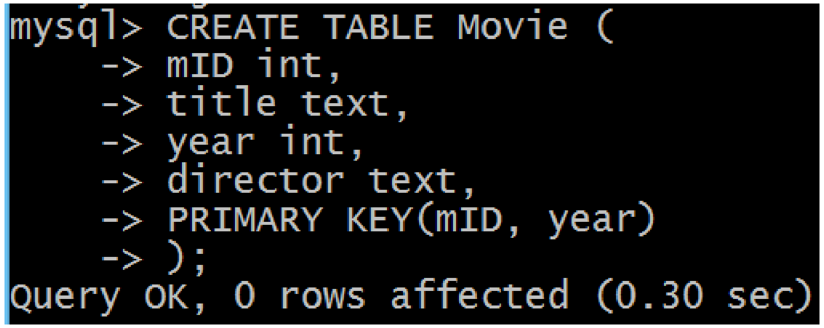


```
mysql> CREATE TABLE Movie (
    -> mID int,
    -> title text,
    -> year int,
    -> director text,
    -> PRIMARY KEY(mID)
    -> );
Query OK, 0 rows affected (0.23 sec)
```

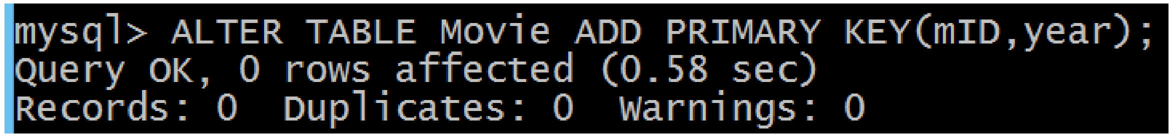


```
mysql> ALTER TABLE Movie ADD PRIMARY KEY(mID);
Query OK, 0 rows affected (0.59 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- Setting mID and year as Primary Key



```
mysql> CREATE TABLE Movie (
    -> mID int,
    -> title text,
    -> year int,
    -> director text,
    -> PRIMARY KEY(mID, year)
    -> );
Query OK, 0 rows affected (0.30 sec)
```



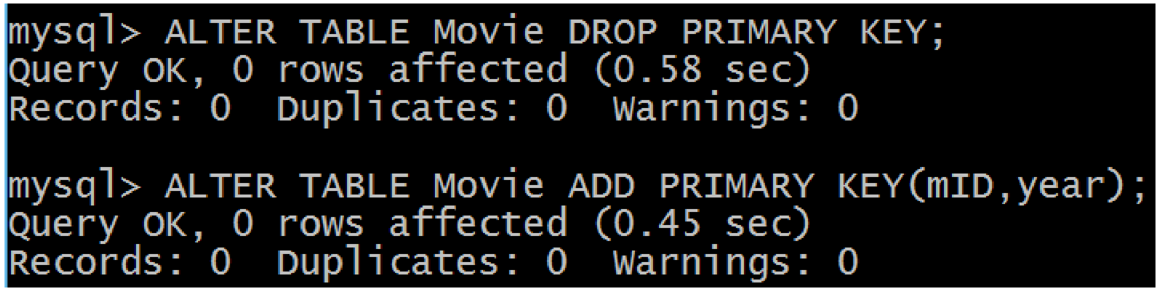
```
mysql> ALTER TABLE Movie ADD PRIMARY KEY(mID,year);
Query OK, 0 rows affected (0.58 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Delete and Changing Primary Key from Table

- To delete the Primary Key, you use the Alter Table statement.

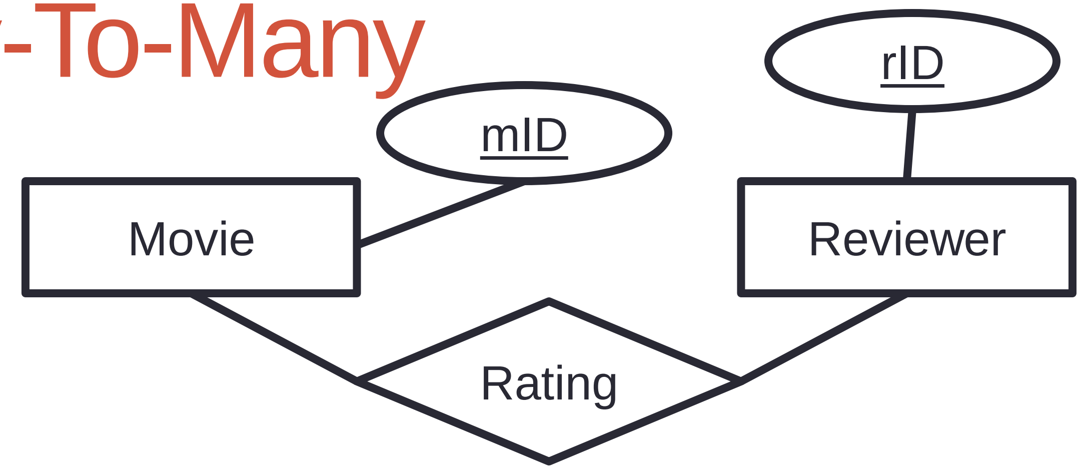
```
ALTER TABLE table_name
DROP PRIMARY KEY;
```

- To change the Primary Key:
  - **YOU NEED TO DELETE THE PRIMARY KEY FIRST**
  - Then add Primary Key.



```
mysql> ALTER TABLE Movie DROP PRIMARY KEY;
Query OK, 0 rows affected (0.58 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Movie ADD PRIMARY KEY(mID,year);
Query OK, 0 rows affected (0.45 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Example: Many-To-Many



- The Primary for:
  - Movie ► mID
  - Reviewer ► rID
- Therefore, the relationship table, The Primary Key for:
  - Rating ► mID, rID
- Definition from E-R Diagram:
  - **ONE Movie** can be rated by **MANY Reviewer**.
  - **ONE Reviewer** can rate **MANY Movie**
- The combination of mID and rID is Unique..

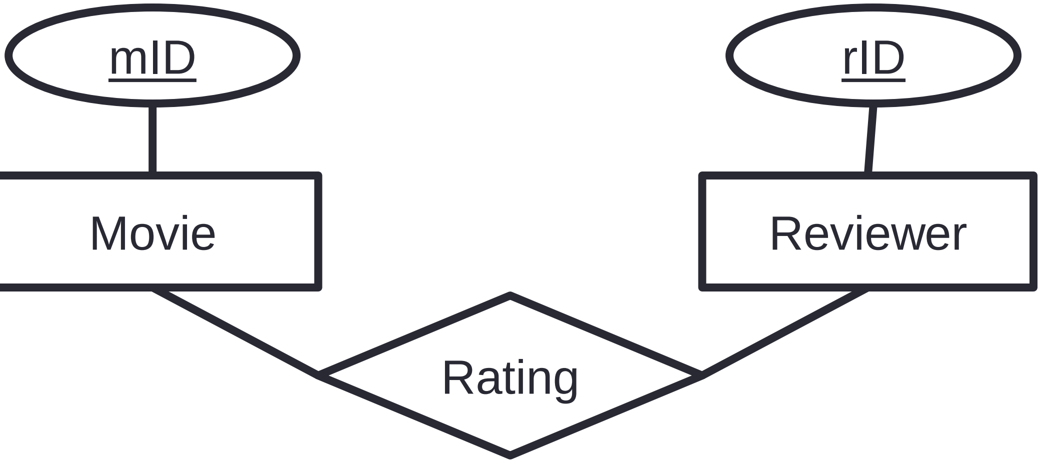
This is a valid Many-To-Many Data

Rating

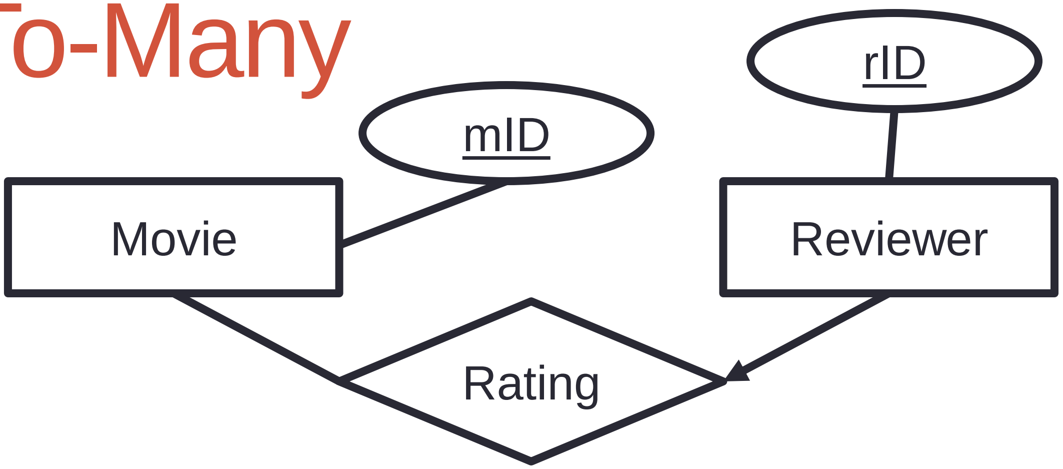
mID	**rID**
101	201
102	201
101	202
103	203

MySQL: Many-To-Many

```
CREATE TABLE Rating(
rID int,
mID int,
PRIMARY KEY(mID, rID));
```



Example: One-To-Many



- The Primary for:
  - Movie ▶ mID
  - Reviewer ▶ rID
- Therefore, the relationship table, The Primary Key for:
  - Rating ▶ rID
- Definition from E-R Diagram:
  - **ONE Movie** can be rated by **MANY Reviewer**.
  - **ONE Reviewer** can rate **ONE Movie**
- rID is Unique..

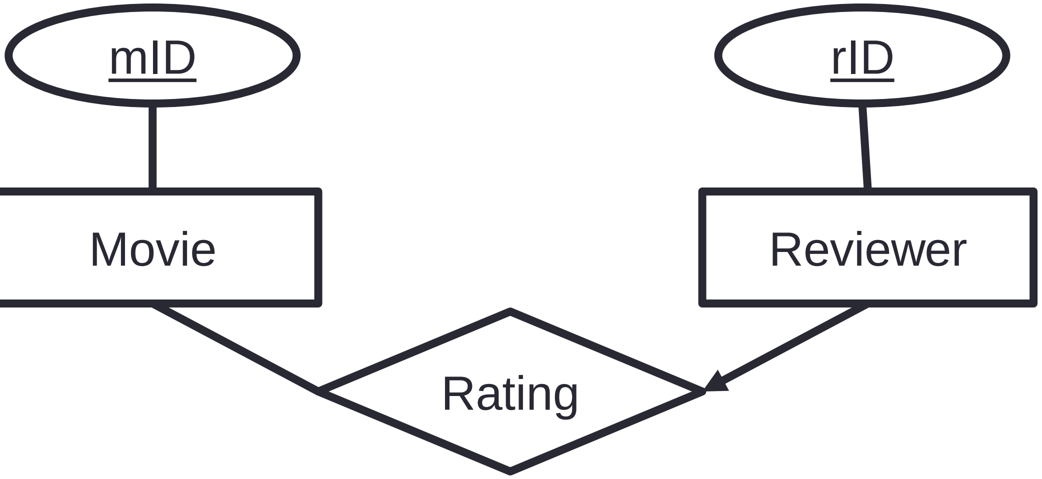
This is a valid One-To-Many Data

Rating

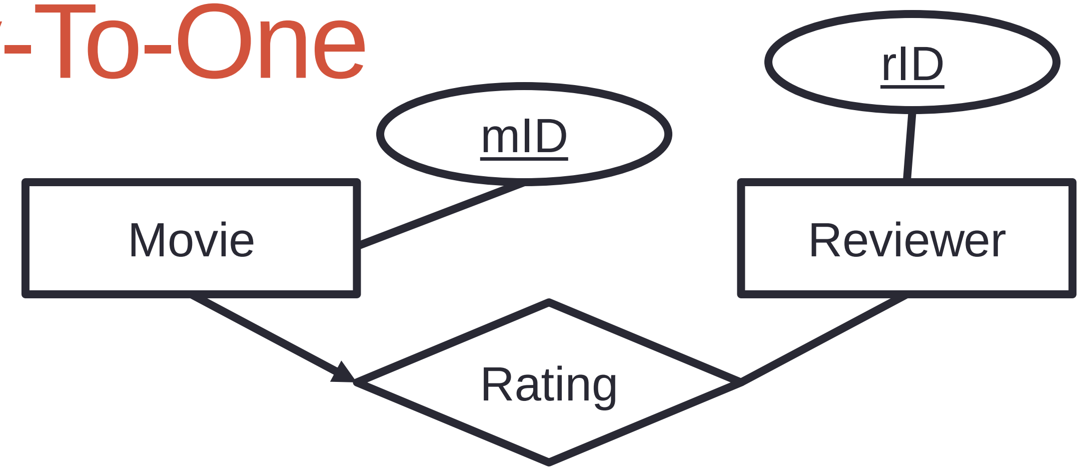
mID	**rID**
101	201
101	202
102	203
103	204

MySQL: One-To-Many

```
CREATE TABLE Rating(
rID int,
mID int,
PRIMARY KEY(rID));
```



Example: Many-To-One



- The Primary for:
  - Movie ▶ mID
  - Reviewer ▶ rID
- Therefore, the relationship table, The Primary Key for:
  - Rating ▶ mID
- Definition from E-R Diagram:
  - **ONE Movie** can be rated by **ONE Reviewer**.
  - **ONE Reviewer** can rate **MANY Movie**
- mID is Unique..

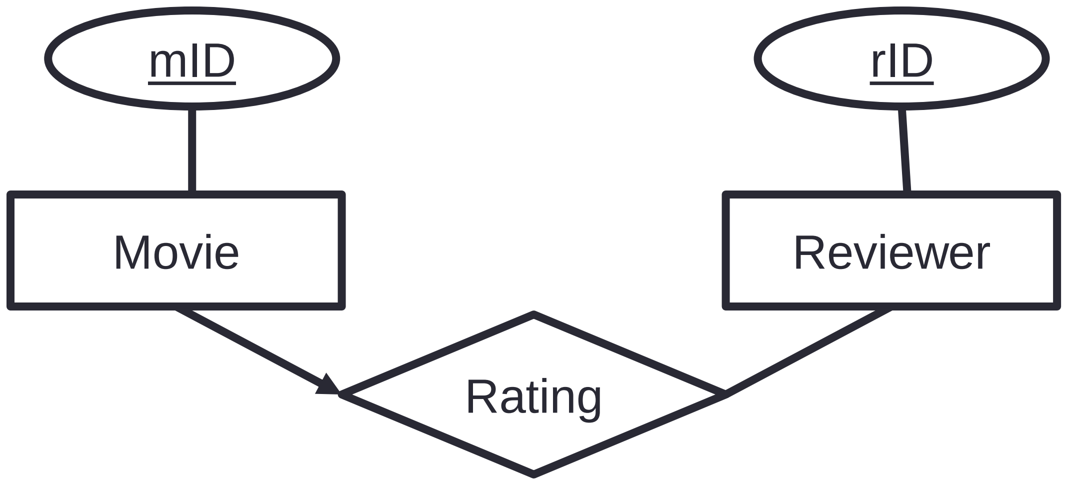
This is a valid Many-To-One Data

Rating

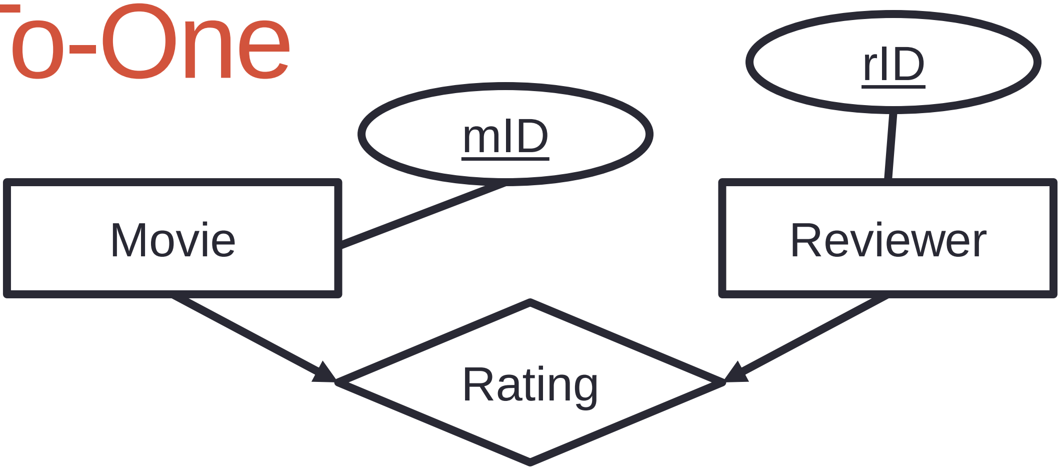
mID	**rID**
101	201
102	203
103	201
104	202

MySQL: One-To-Many

```
CREATE TABLE Rating(
rID int,
mID int,
PRIMARY KEY(mID));
```



Example: One-To-One



- The Primary for:
  - Movie ▶ mID
  - Reviewer ▶ rID
- Therefore, the relationship table: Rating
  - Rating ▶ Set mID as Primary Key and Set rID as Unique
- Definition from E-R Diagram:
  - **ONE Movie** can be rated by **ONE Reviewer**.
  - **ONE Reviewer** can rate **ONE Movie**
- mID is Unique and rID is Unique.

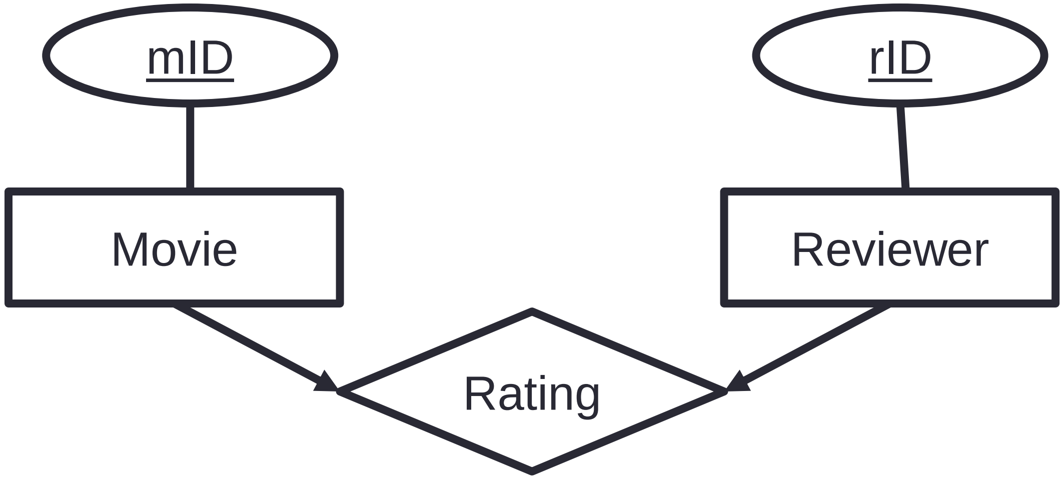
This is a valid Many-To-One Data

Rating

mID	rID
101	201
102	203
103	204
104	202

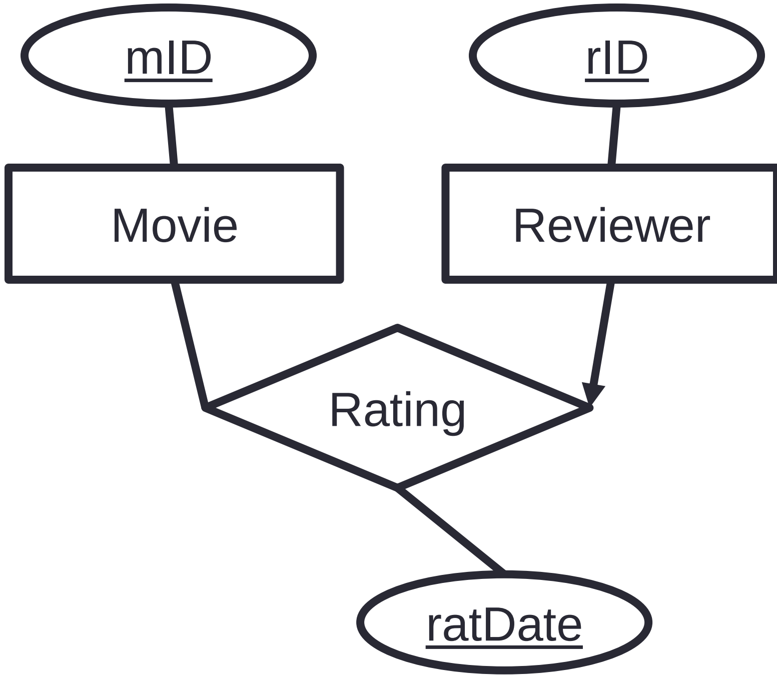
MySQL: One-To-One

```
CREATE TABLE Rating(
rID int,
mID int,
PRIMARY KEY(mID),
UNIQUE(rID));
```



Extra Primary Key in Relationship

- The examples only shows that we set the Primary Key from other entity.
- What if the Relationship itself has attribute that designed with Primary Key.
- Using the One-To-Many Example
- Rating Table Primary Key:
  - ► rID and ratDate.
- Definition:
  - **ONE Movie** can be rated by **MANY Reviewer** in a given date.
  - **ONE Reviewer** can rate **ONE Movie** in a given date.
- Combination of rID and ratDate is Unique

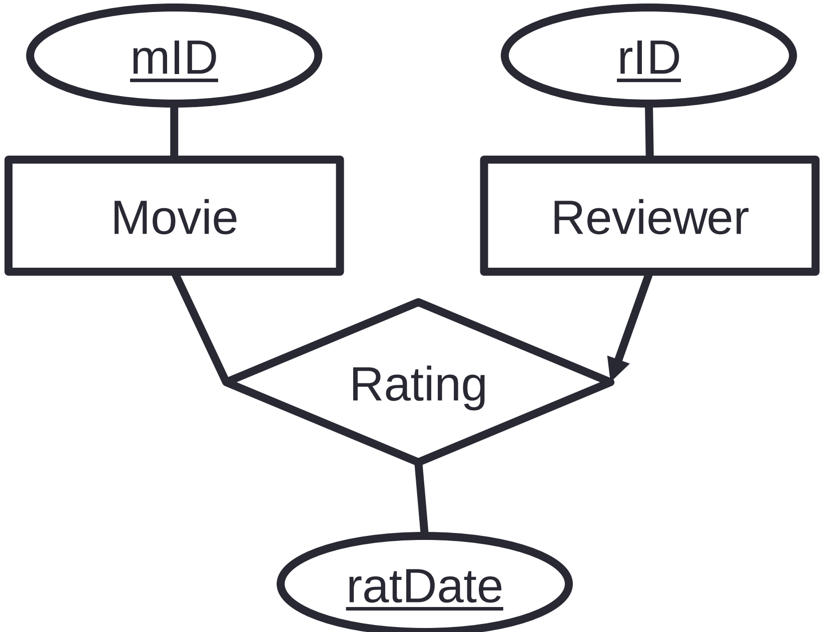


MySQL: Extra Primary Key in Relationship

```
CREATE TABLE Rating(
rID int,
mID int,
ratDate date
PRIMARY KEY(rID, ratDate));
```

Valid Data for Rating Table:

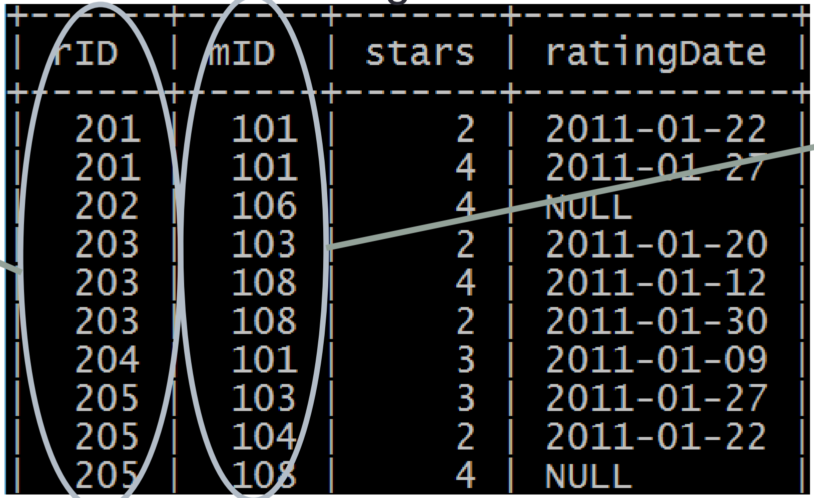
mID	rID	date
101	201	2015-12-02
101	203	2015-12-02
101	201	2015-12-13
104	202	2015-12-13
103	202	2015-12-20



Foreign Key

- Foreign key is the reference of the Primary Key from OTHER table.

Rating Table

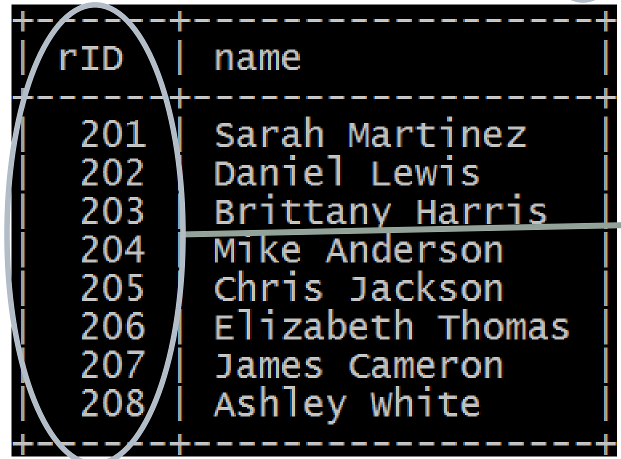


```
+-----+-----+-------+------------+
| rID | mID | stars | ratingDate |
+-----+-----+-------+------------+
| 201 | 101 |     2 | 2011-01-22 |
| 201 | 101 |     4 | 2011-01-27 |
| 202 | 106 |     4 | NULL       |
| 203 | 103 |     2 | 2011-01-20 |
| 203 | 108 |     4 | 2011-01-12 |
| 203 | 108 |     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       |
```

Attribute that reference to Reviewer Table Primary Key

Attribute that reference to Movie Table Primary Key

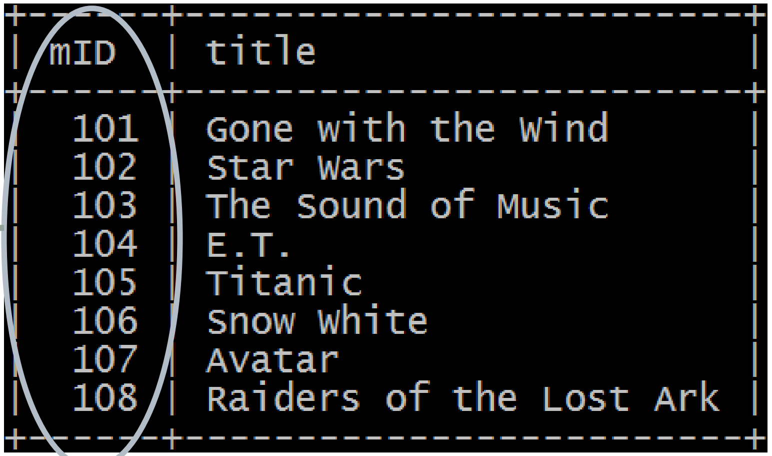
ReviewerTable



```
+-----+------------------+
| rID | name             |
+-----+------------------+
| 201 | Sarah Martinez   |
| 202 | Daniel Lewis     |
| 203 | Brittany Harris  |
| 204 | Mike Anderson    |
| 205 | Chris Jackson    |
| 206 | Elizabeth Thomas |
| 207 | James Cameron    |
| 208 | Ashley White     |
+-----+------------------+
```

Primary Key for Movie and Reviewer Table

Movie Table



```
+-----+-------------------------+
| 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 |
+-----+-------------------------+
```

Foreign Key (cont.)

- To set Foreign Key in a table:

Setting FOREIGN KEY when creating new table.

```
CREATE TABLE table_name (
    column_name1 data_type,
    column_name2 data_type,
    column_name3 data_type,
FOREIGN KEY (some_column) REFERENCES
    other_table(other_column)
);
```

The column in the same table

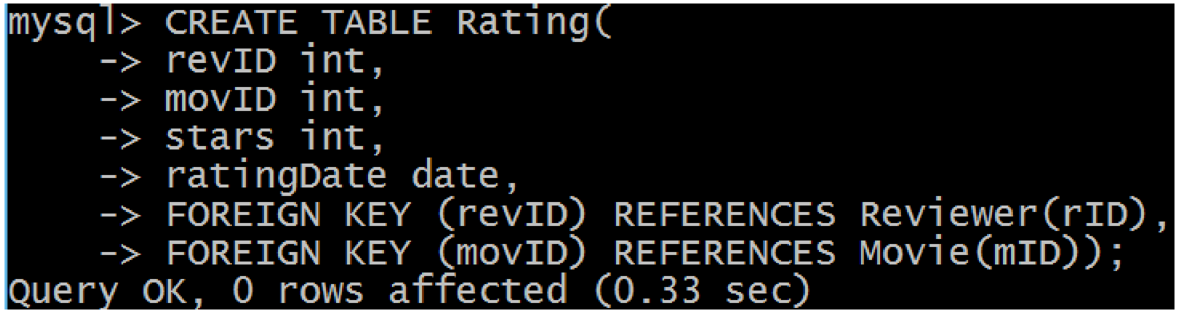
The column from the other table that it referenced to.

Setting FOREIGN KEY for an existing table.

```
ALTER TABLE table_name
ADD FOREIGN KEY (some_column) REFERENCES
    other_table(other_column);
```

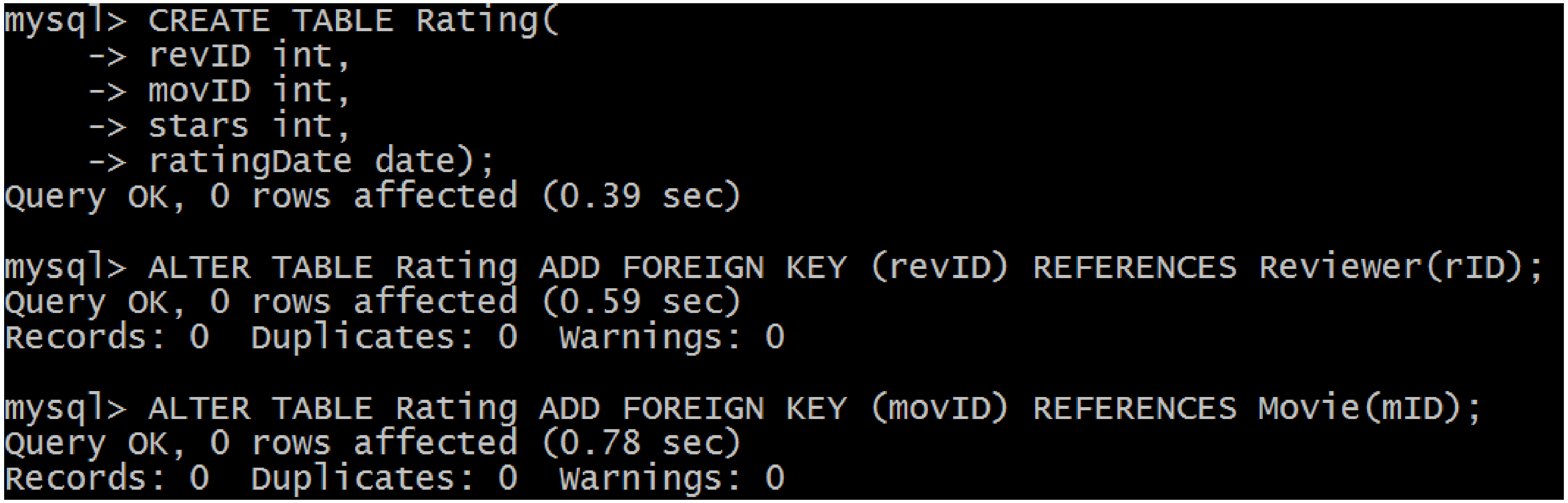
Example Foreign Key

- revID and movID in Rating table REFERENCES rID and mID in Reviewer and Movie table respectively.



```
mysql> CREATE TABLE Rating(
    -> revID int,
    -> movID int,
    -> stars int,
    -> ratingDate date,
    -> FOREIGN KEY (revID) REFERENCES Reviewer(rID),
    -> FOREIGN KEY (movID) REFERENCES Movie(mID));
Query OK, 0 rows affected (0.33 sec)
```

- Create table without Foreign Key. But Alter the table by adding foreign key.



```
mysql> CREATE TABLE Rating(
    -> revID int,
    -> movID int,
    -> stars int,
    -> ratingDate date);
Query OK, 0 rows affected (0.39 sec)

mysql> ALTER TABLE Rating ADD FOREIGN KEY (revID) REFERENCES Reviewer(rID);
Query OK, 0 rows affected (0.59 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Rating ADD FOREIGN KEY (movID) REFERENCES Movie(mID);
Query OK, 0 rows affected (0.78 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Delete and Changing Foreign Key from Table

- Deleting Foreign Key is **NOT THE SAME** as deleting Primary Key.

Not valid statement

~~**ALTER TABLE** table_name~~
~~**DROP FOREIGN KEY;**~~

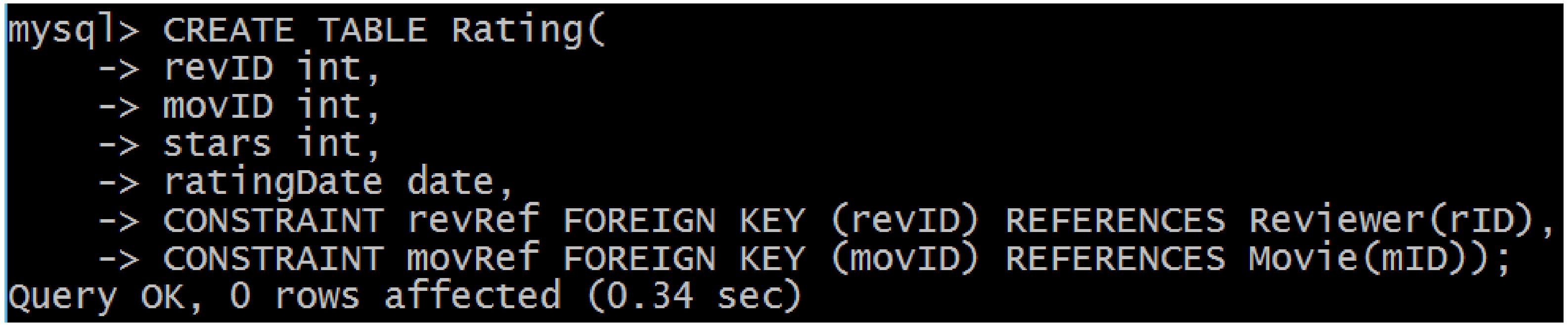
- To delete the foreign key, the foreign key constraint has to be named first. To do that, make sure you give it a name during table creation or altering.

CREATE TABLE table_name (column_name1 data_type, column_name2 data_type, column_name3 data_type, **CONSTRAINT** constraint_name **FOREIGN KEY** (some_column) **REFERENCES** other_table(other_column));	**ALTER TABLE** table_name **ADD** **CONSTRAINT** constraint_name **FOREIGN KEY** (some_column) **REFERENCES** other_table(other_column);

Add this statement

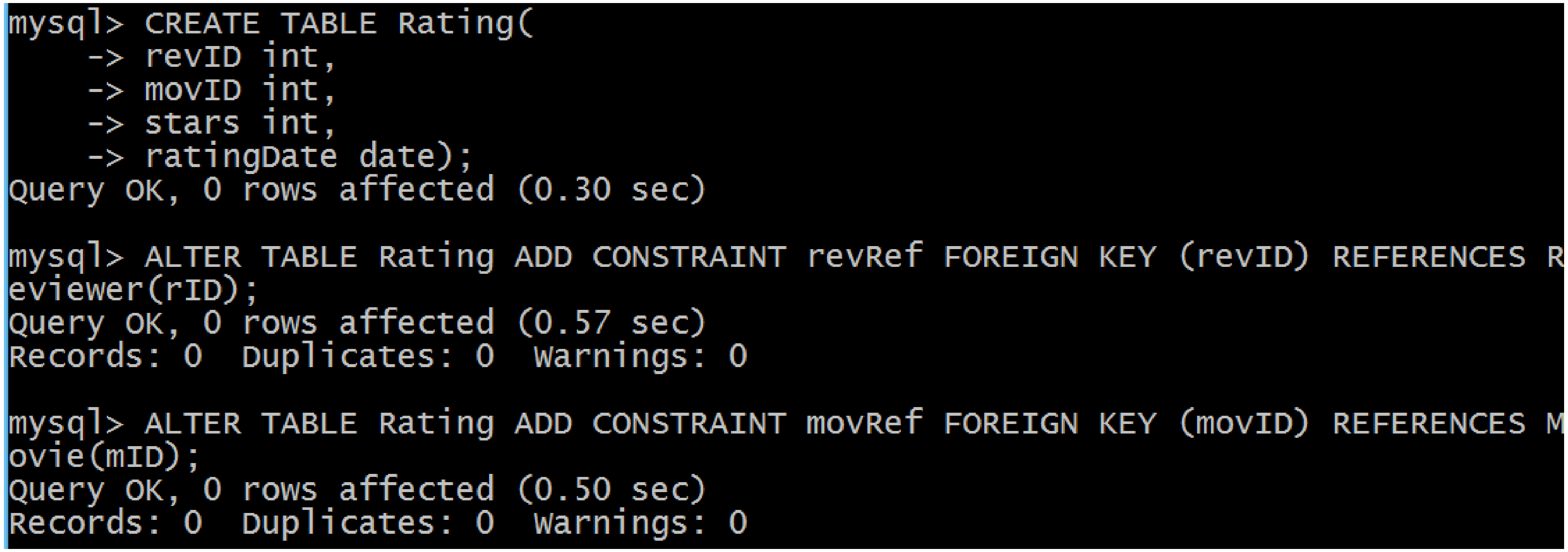
Example Adding Constraint Name

- Include CONSTRAINT statement for foreign key when creating table.



```
mysql> CREATE TABLE Rating(
    -> revID int,
    -> movID int,
    -> stars int,
    -> ratingDate date,
    -> CONSTRAINT revRef FOREIGN KEY (revID) REFERENCES Reviewer(rID),
    -> CONSTRAINT movRef FOREIGN KEY (movID) REFERENCES Movie(mID));
Query OK, 0 rows affected (0.34 sec)
```

- Include CONSTRAINT statement for foreign key when altering table.



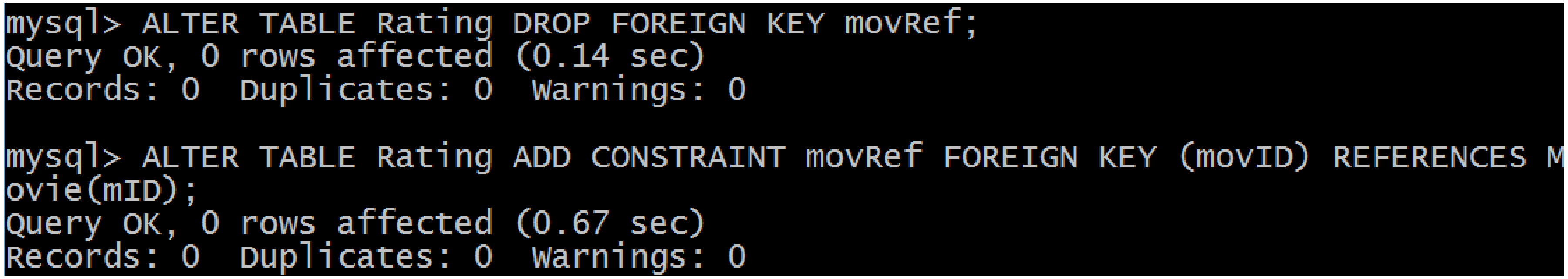
```
mysql> CREATE TABLE Rating(
    -> revID int,
    -> movID int,
    -> stars int,
    -> ratingDate date);
Query OK, 0 rows affected (0.30 sec)

mysql> ALTER TABLE Rating ADD CONSTRAINT revRef FOREIGN KEY (revID) REFERENCES R
eviewer(rID);
Query OK, 0 rows affected (0.57 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Rating ADD CONSTRAINT movRef FOREIGN KEY (movID) REFERENCES M
ovie(mID);
Query OK, 0 rows affected (0.50 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Delete and Changing Foreign Key from Table

- After naming the Foreign Key constraint, now it is possible to delete the Foreign Key

ALTER TABLE table_name
DROP FOREIGN KEY constraint_name**;**

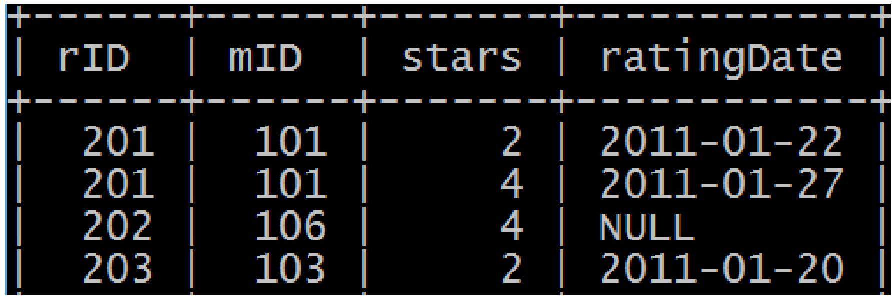
- Changing Foreign Key:
  - YOU NEED TO DELETE THE FOREIGN KEY FIRST
  - Then adding foreign key



```
mysql> ALTER TABLE Rating DROP FOREIGN KEY movRef;
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Rating ADD CONSTRAINT movRef FOREIGN KEY (movID) REFERENCES M
ovie(mID);
Query OK, 0 rows affected (0.67 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

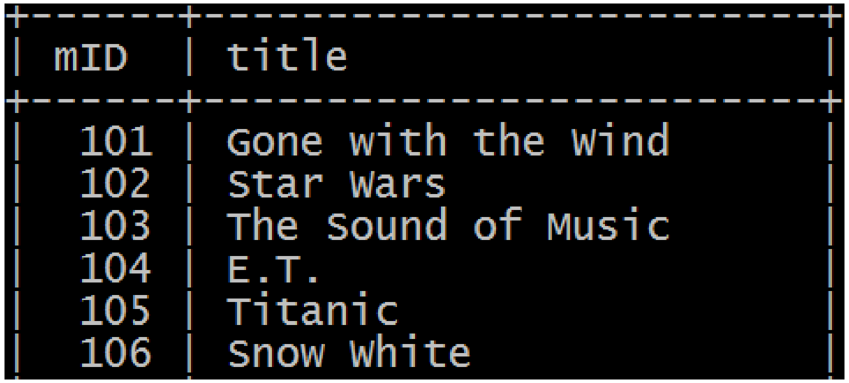
Why Use Foreign Key?

Rating Table



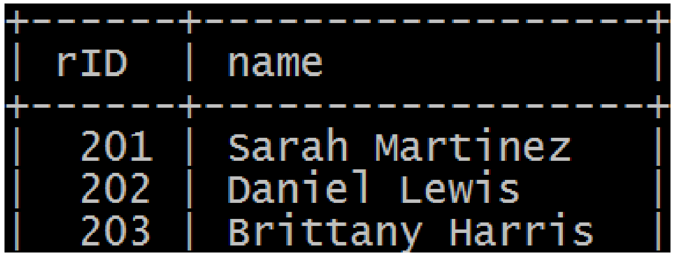
```
+-------+-------+--------+------------+
| rID   | mID   | stars  | ratingDate |
+-------+-------+--------+------------+
|   201 |   101 |      2 | 2011-01-22 |
|   201 |   101 |      4 | 2011-01-27 |
|   202 |   106 |      4 | NULL       |
|   203 |   103 |      2 | 2011-01-20 |
```

Movie Table



```
+------+-------------------------+
| mID  | title                   |
+------+-------------------------+
|  101 | Gone with the Wind      |
|  102 | Star Wars               |
|  103 | The Sound of Music      |
|  104 | E.T.                    |
|  105 | Titanic                 |
|  106 | Snow White              |
```

Reviewer Table



```
+------+------------------+
| rID  | name             |
+------+------------------+
|  201 | Sarah Martinez   |
|  202 | Daniel Lewis     |
|  203 | Brittany Harris  |
```

- Using the above example table where Rating table rID and mID referenced to rID and mID from Reviewer and Movie table respectively.
  - Rating table has rID=201, it is not possible to delete rID=201 in Reviewer table.
  - Rating table has mID=106, it is not possible to delete mID=106 in Movie table.
  - Deleting table Movie and Reviewer will not be possible.
  - Inserting into Rating table where mID=108 will not be possible, because Movie table does not have mID=108.

Auto Increment ID

- Sometimes, it is better if you don’t allow the user to enter the ID for a table.
- You will set the ID to be auto incremented by MySQL.
- In MySQL, it is called AUTO_INCREMENT
- To use this statement:

```
CREATE TABLE table_name (
id INT AUTO_INCREMENT,
PRIMARY KEY(id));
```

- AUTO_INCREMENT requires a Number Type column (Typically INT) and the column is set as PRIMARY KEY.
- Once created, the id will starts from 1.

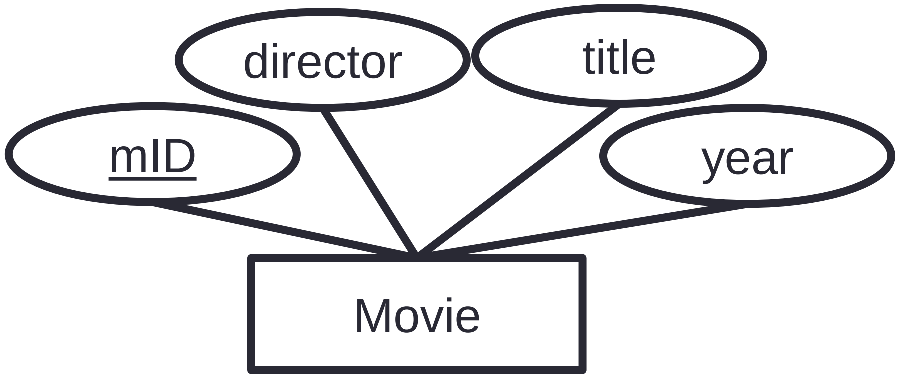
Auto Increment ID (cont.)

- Once the auto increment is set into the table, usually it starts from number 1.
- But it is still possible to set the auto increment starting number to different number.
- For example: A movie ID (mID) should start from 101.
- To implement this. You create a movie table and set mID auto increment. Then you need to alter the table.

```
ALTER TABLE table_name AUTO_INCREMENT=101;
```

Example Auto Increment ID

```
CREATE TABLE Movie(
mID INT AUTO_INCREMENT,
director TEXT,
title TEXT,
year INT,
PRIMARY KEY (mID));
```



- To insert data into the table:

```
INSERT INTO Movie (director, title, year)
VALUES (‘George Lucas’, ‘Star Wars’, 1977);
```

- You don’t need to insert anything into mID column.

Auto Date Input

- If your table require a column that requires the current date and time. It is possible for you to set the data type of the column that automatically input the date when new data is inserted.
- The data type is called **TIMESTAMP**. Basically, this data type will get the current date and time of the MySQL Server and insert that into the table column.
- To implement TIMESTAMP:

```
CREATE TABLE table_name (
column_name1 TIMESTAMP,
………);
```

Example Auto Date Input

- Example: Comment table

```
CREATE TABLE Comment(
cID INT AUTO_INCREMENT,
comment TEXT,
dT TIMESTAMP);
```

- Inserting into Comment table
  - Current date: 11th September 2015
  - Current time: 9:42pm

```
INSERT INTO Comment (comment)
VALUES (‘Hello, I am commenting);
```

cID	comment	dT
1	Hello, I am commenting	2015-09-11 21:42:32

Organising Retrieved Data

- Usually the data retrieved using the SELECT statement is not organised. Sometimes, you want your data to be organised in ascending or descending order according to which column(s).
- To do this, the MySQL statement is **ORDER BY**.

ASCENDING	DESCENDING
SELECT column1, column2, …. **FROM** table_name, … **ORDER BY** some_column **ASC**;	**SELECT** column1, column2, …. **FROM** table_name, … **ORDER BY** some_column **DESC**;

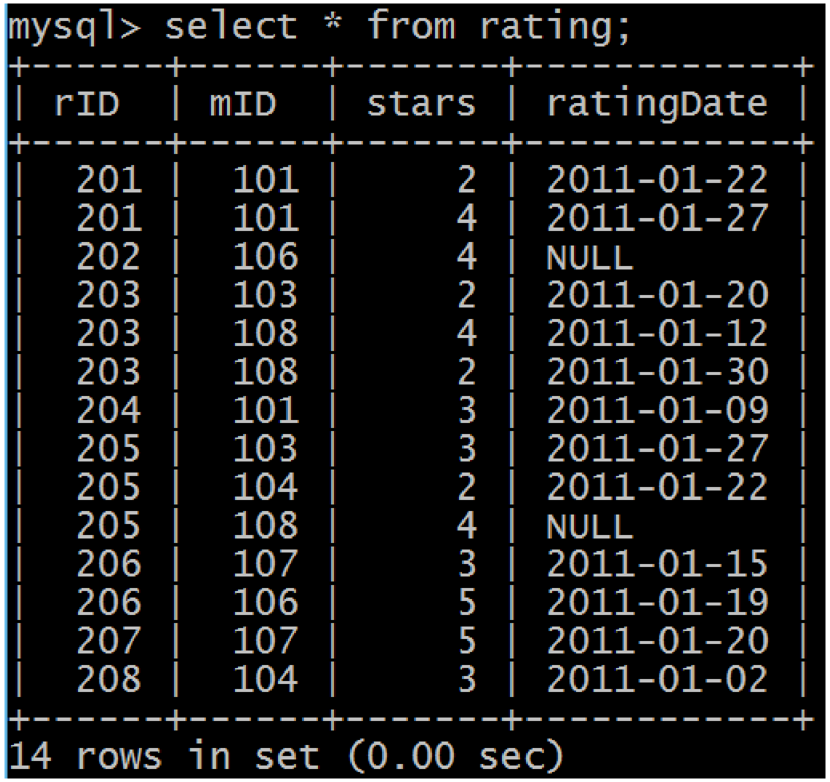
- ORDER BY multiple columns.

```
SELECT column1, column2, ….
FROM table_name, …
ORDER BY some_column ASC|DESC, other_column ASC|DESC, ……;
```

- If **WHERE** statement included, ORDER BY will be stated after that.

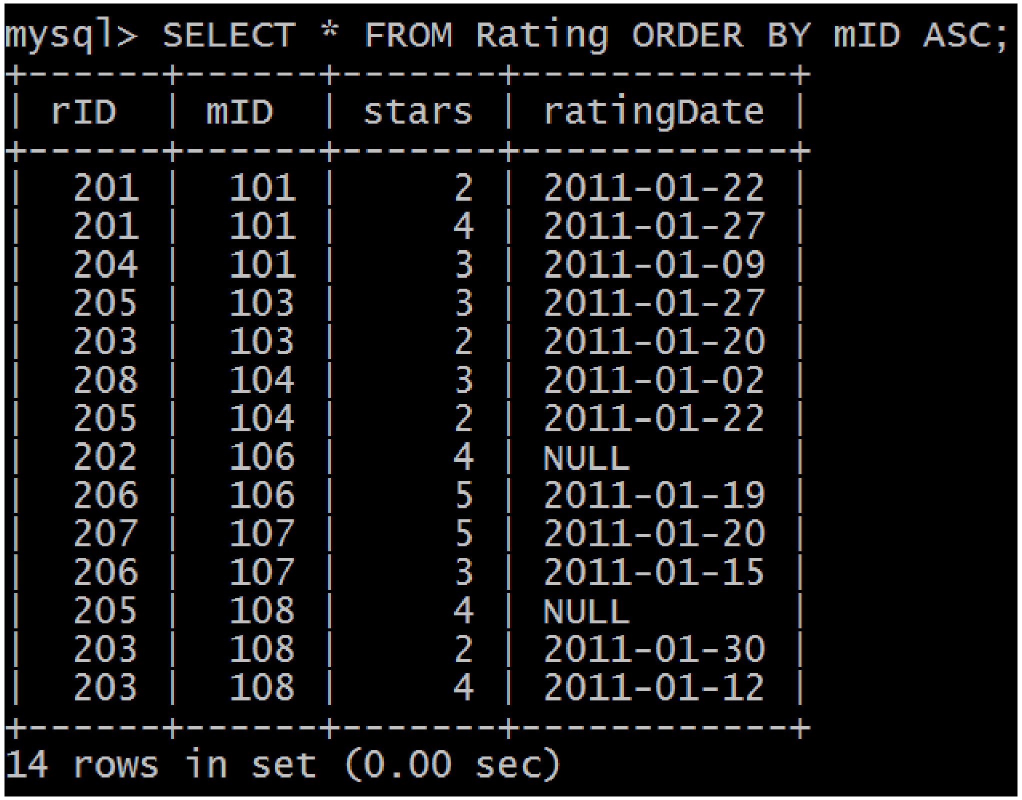
Example: Organising Retrieved Data

Raw Data Rating Table



```
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)
```

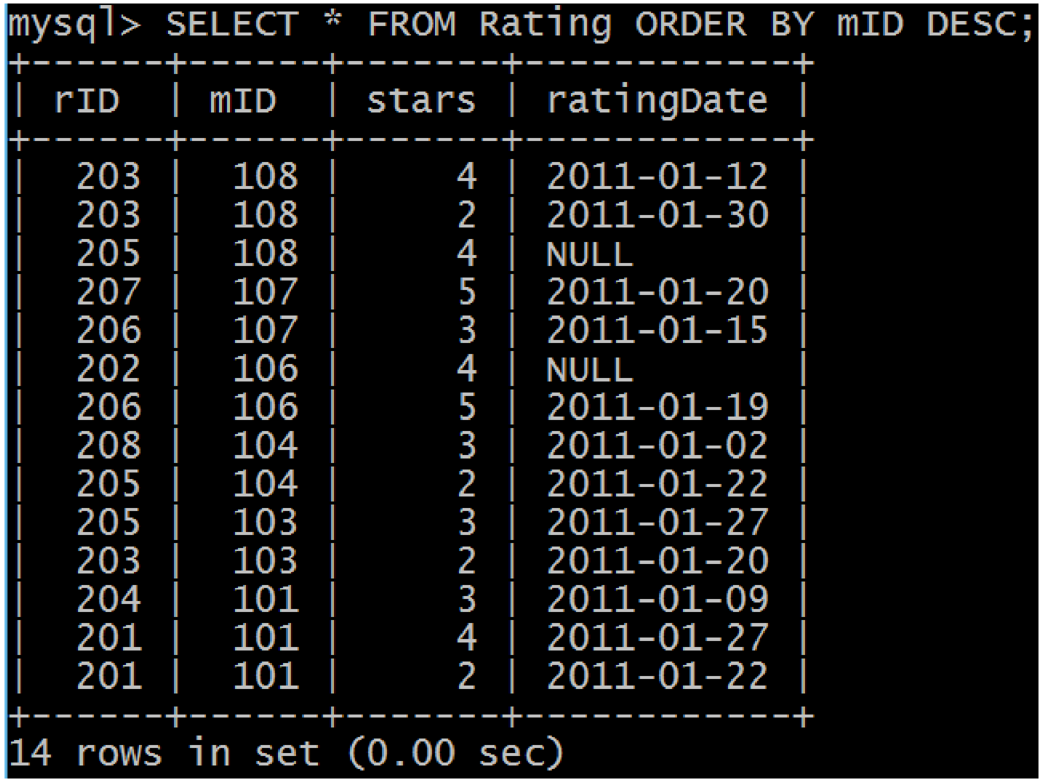
mID in Ascending Order



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

Example: Organising Retrieved Data

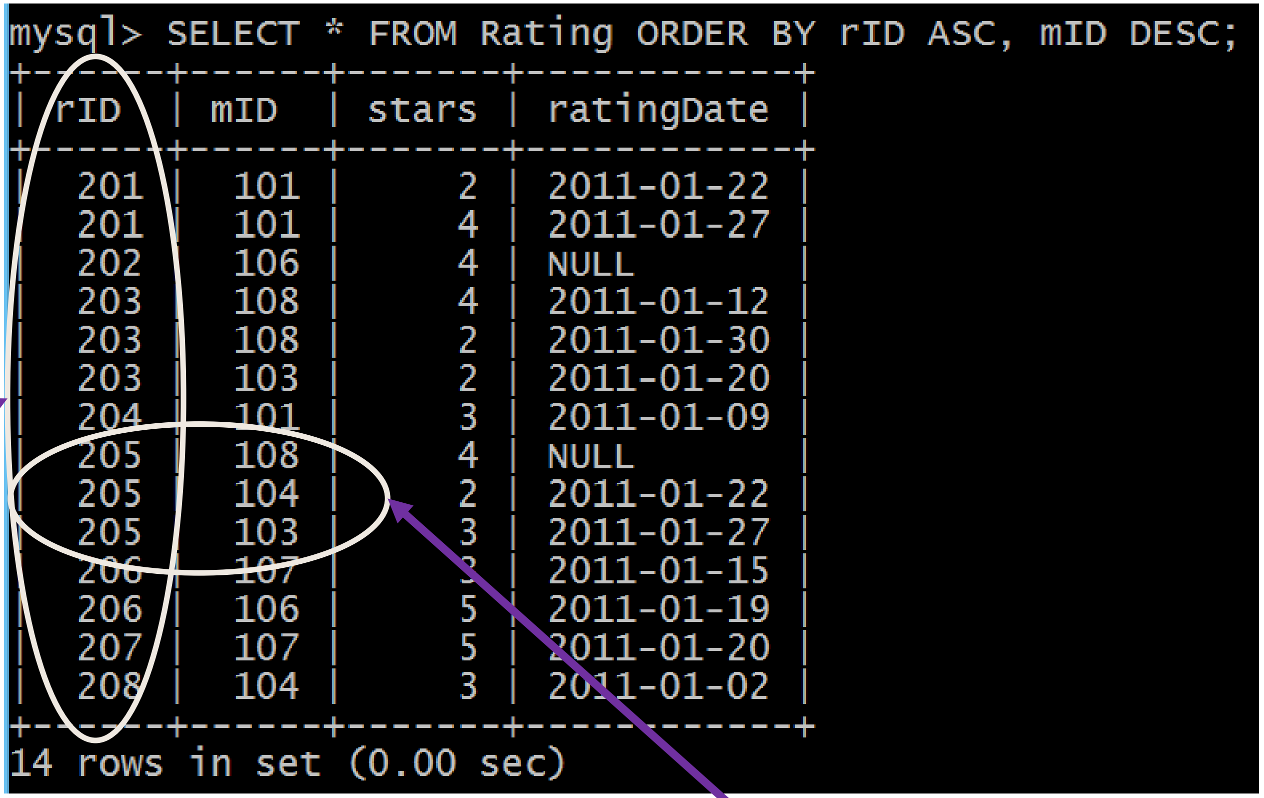
<u>mID in Ascending Order</u>



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

Example: Organising Retrieved Data

rID in Ascending Order and mID in Descending Order



```
mysql> SELECT * FROM Rating ORDER BY rID ASC, mID DESC;
+------+------+-------+------------+
| rID  | mID  | stars | ratingDate |
+------+------+-------+------------+
|  201 |  101 |     2 | 2011-01-22 |
|  201 |  101 |     4 | 2011-01-27 |
|  202 |  106 |     4 | NULL       |
|  203 |  108 |     4 | 2011-01-12 |
|  203 |  108 |     2 | 2011-01-30 |
|  203 |  103 |     2 | 2011-01-20 |
|  204 |  101 |     3 | 2011-01-09 |
|  205 |  108 |     4 | NULL       |
|  205 |  104 |     2 | 2011-01-22 |
|  205 |  103 |     3 | 2011-01-27 |
|  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)
```

rID ASC

mID ASC

Function

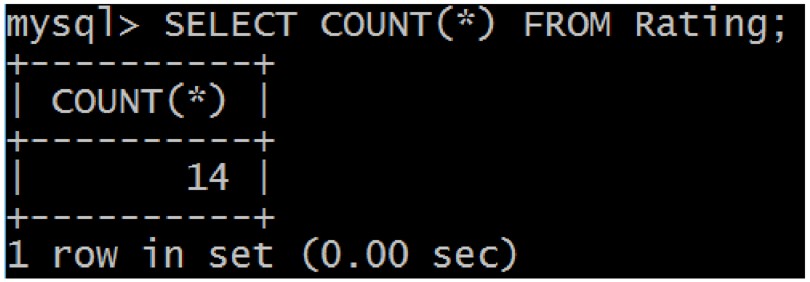
- MySQL supports the use of function. These functions can be used in INSERT, DELETE, UPDATE and SELECT statement.

  https://dev.mysql.com/doc/refman/5.5/en/functions.html

- Examples of MySQL Function:

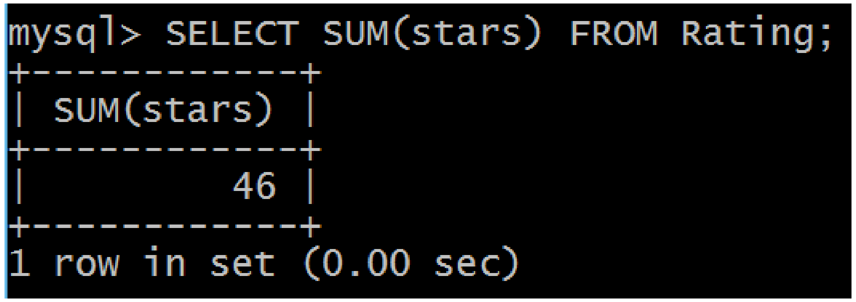
Function	Description
NOW()	Get the current time and date. 'YYYY-MM-DD HH:MM;SS'
CURDATE()	Get the current date. 'YYYY-MM-DD'
CURTIME()	Get the current time. 'HH:MM:SS'
COUNT(column)	Get the number of rows.
SUM(column)	Get the sum of values in a column.
AVG(column)	Get the average of values in a column.
CONCAT(column1, column2, …)	Get the text of columns combination.

Example: Function in SELECT



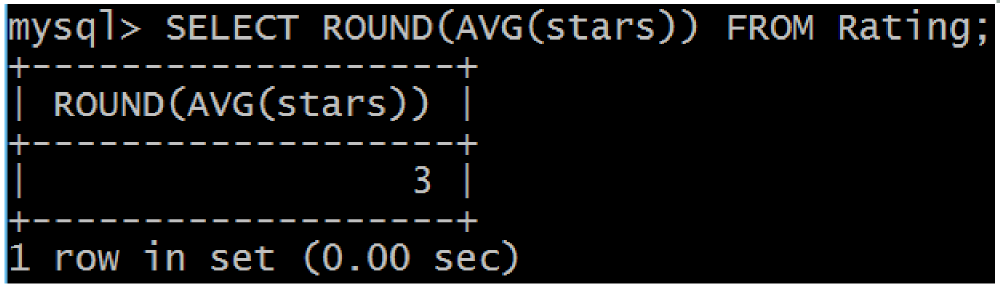
```
mysql> SELECT COUNT(*) FROM Rating;
+----------+
| COUNT(*) |
+----------+
|       14 |
+----------+
1 row in set (0.00 sec)
```

Get the total number of Rows in Rating table



```
mysql> SELECT SUM(stars) FROM Rating;
+------------+
| SUM(stars) |
+------------+
|         46 |
+------------+
1 row in set (0.00 sec)
```

Get the sum of stars in Rating table

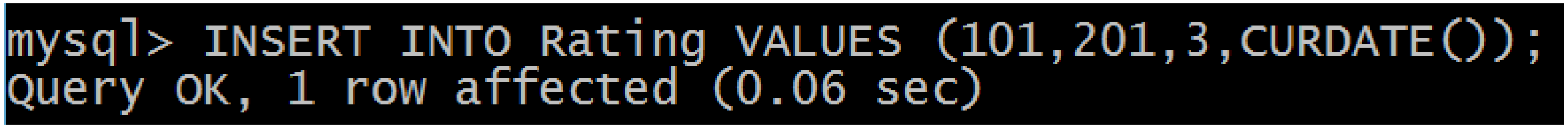


```
mysql> SELECT ROUND(AVG(stars)) FROM Rating;
+-------------------+
| ROUND(AVG(stars)) |
+-------------------+
|                 3 |
+-------------------+
1 row in set (0.00 sec)
```

It is possible to have a function surrounded by a function.
Get the Rounded Off value of the average of stars.

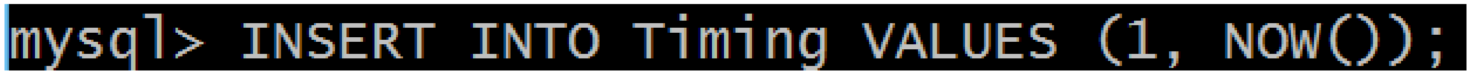
Example: Function in INSERT and DELETE

Get the current date and insert them into table



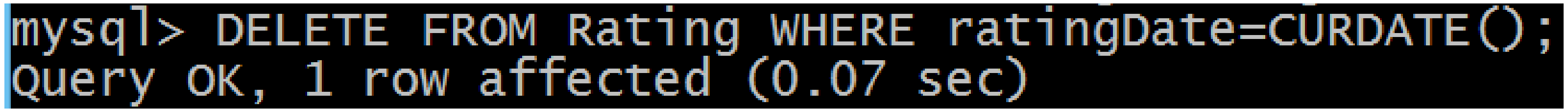
```
mysql> INSERT INTO Rating VALUES (101,201,3,CURDATE());
Query OK, 1 row affected (0.06 sec)
```

Get the current date and time then insert them into table



```
mysql> INSERT INTO Timing VALUES (1, NOW());
```

Delete data in Rating table where the date is Today.



```
mysql> DELETE FROM Rating WHERE ratingDate=CURDATE();
Query OK, 1 row affected (0.07 sec)
```

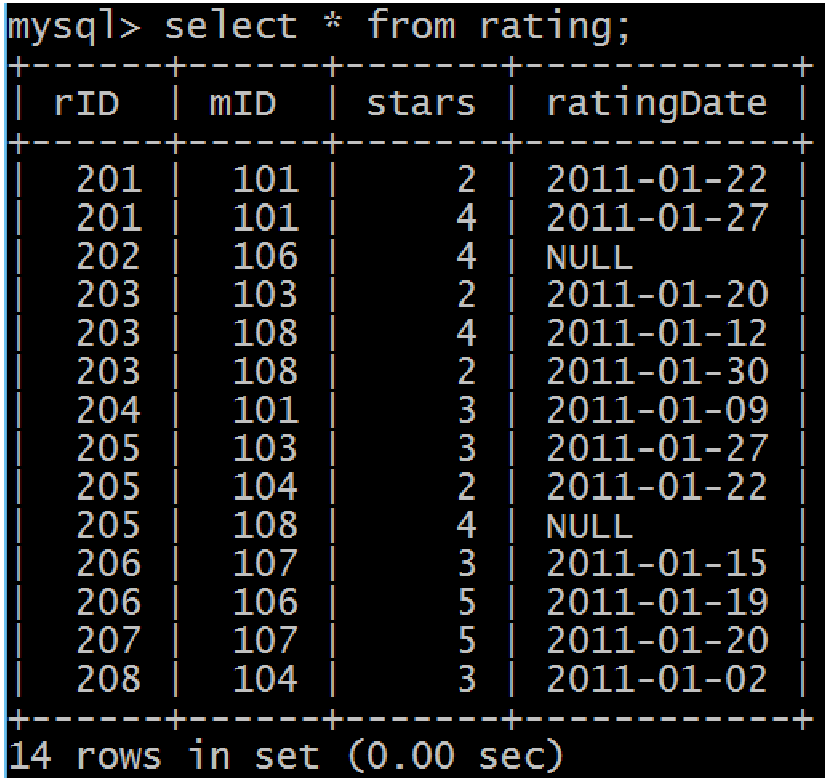
Distinct

- When you retrieved data, sometimes you want to get the distinct data only. That means only show duplicate value once.
- In MySQL, the statement is **DISTINCT**.

```
SELECT DISTINCT column_name
FROM table_name;
```

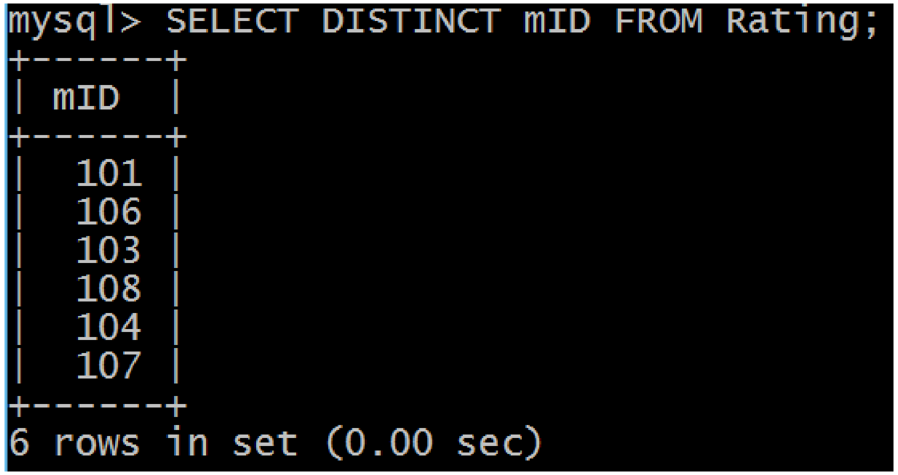
Example: Distinct

Raw data in Rating table



```
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)
```

We want to know which Movie already reviewed.



```
mysql> SELECT DISTINCT mID FROM Rating;
+------+
| mID  |
+------+
|  101 |
|  106 |
|  103 |
|  108 |
|  104 |
|  107 |
+------+
6 rows in set (0.00 sec)
```

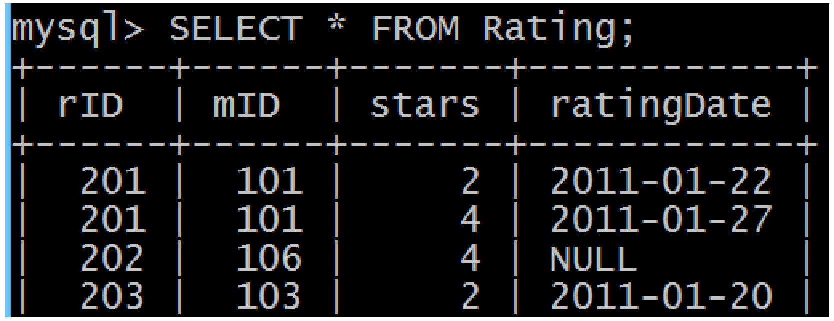
Alias

- Sometimes the names of the columns are so technical that makes the result of SELECT statement hard to understand.
- MySQL provide Alias feature for this. The statement is **AS**.

```
SELECT column_name AS ‘alias_name’
FROM table_name;
```

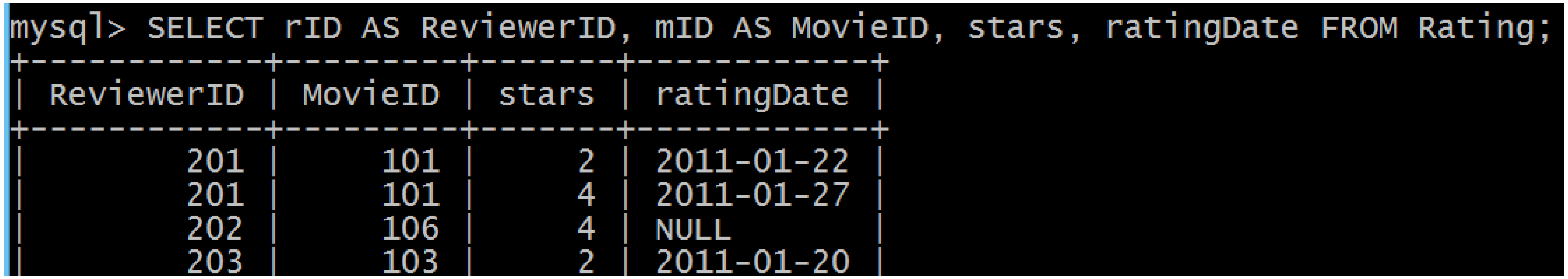
Example: Alias

If other people read this, they don't know rID and mID means.



```
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 |
```

Now rID is named as ReviewerID and mID is named as MovieID



```
mysql> SELECT rID AS ReviewerID, mID AS MovieID, stars, ratingDate FROM Rating;
+------------+---------+-------+------------+
| ReviewerID | MovieID | stars | ratingDate |
+------------+---------+-------+------------+
|        201 |     101 |     2 | 2011-01-22 |
|        201 |     101 |     4 | 2011-01-27 |
|        202 |     106 |     4 | NULL       |
|        203 |     103 |     2 | 2011-01-20 |
```

Views

- A view is basically a virtual table. You will not be able to manipulate the table. It just store the SELECT statement.
- MySQL Statement for view:

  CREATE VIEW view_name **AS** select_statement;

- To show the view:

  SELECT columns **FROM** view_name;

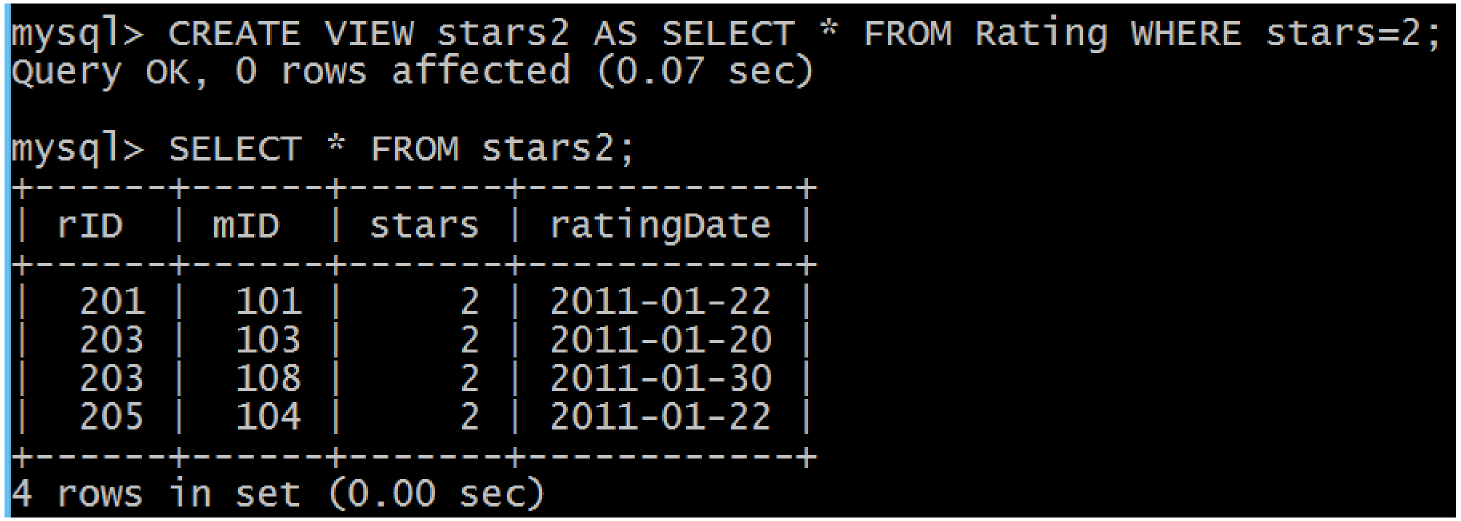
- If the view no longer needed, it can be deleted.

  DROP VIEW view_name;

- **SHOW FULL TABLES** – will show the list of tables and views.

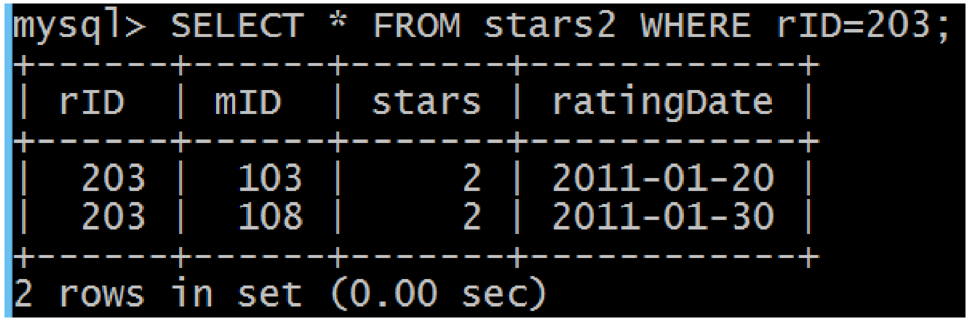
Example: Views

Creating a view and retrieve data from the view like retrieving data from a normal table.



```
mysql> CREATE VIEW stars2 AS SELECT * FROM Rating WHERE stars=2;
Query OK, 0 rows affected (0.07 sec)

mysql> SELECT * FROM stars2;
+------+------+-------+------------+
| rID  | mID  | stars | ratingDate |
+------+------+-------+------------+
|  201 |  101 |     2 | 2011-01-22 |
|  203 |  103 |     2 | 2011-01-20 |
|  203 |  108 |     2 | 2011-01-30 |
|  205 |  104 |     2 | 2011-01-22 |
+------+------+-------+------------+
4 rows in set (0.00 sec)
```

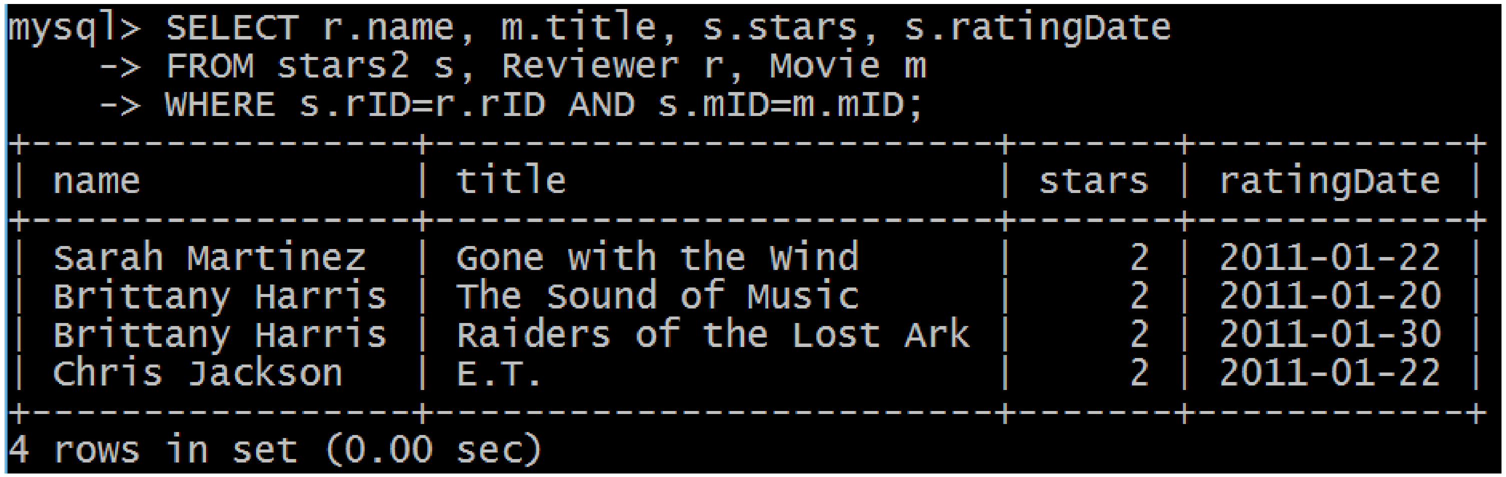
You can add WHERE statement when using view.



```
mysql> SELECT * FROM stars2 WHERE rID=203;
+------+------+-------+------------+
| rID  | mID  | stars | ratingDate |
+------+------+-------+------------+
|  203 |  103 |     2 | 2011-01-20 |
|  203 |  108 |     2 | 2011-01-30 |
+------+------+-------+------------+
2 rows in set (0.00 sec)
```

Example: Views

Retrieving data from view and other tables and get the reviewer name in Reviewer table and movie title in Movie table.



```
mysql> SELECT r.name, m.title, s.stars, s.ratingDate
    -> FROM stars2 s, Reviewer r, Movie m
    -> WHERE s.rID=r.rID AND s.mID=m.mID;
+------------------+--------------------------+-------+------------+
| name             | title                    | stars | ratingDate |
+------------------+--------------------------+-------+------------+
| Sarah Martinez   | Gone with the Wind       |     2 | 2011-01-22 |
| Brittany Harris  | The Sound of Music       |     2 | 2011-01-20 |
| Brittany Harris  | Raiders of the Lost Ark  |     2 | 2011-01-30 |
| Chris Jackson    | E.T.                     |     2 | 2011-01-22 |
+------------------+--------------------------+-------+------------+
4 rows in set (0.00 sec)
```