DATA TYPES

I. Numeric Data Types

- **INT** – A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from 2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
- **TINYINT** – A small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
- **SMALLINT --** A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
- **MEDIUMINT --** A small-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
- **BIGINT** -- A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 11 digits.
- **FLOAT (M,D)** – A floating-point number that cannot be unsigned. You can define the display length (M) and the no. of decimals (D). This is not required and will default to 10,2, where 2 is the no. decimals and 10 is the total number digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
- **DOUBLE (M,D)** -- A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the no. of decimals (D). This is not required and will default to 16,4 , where 4 is the no. decimals. Decimal precision can go to 53 places for a DOUBLE.
- **DECIMAL** – An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the no. of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

II. Data and Time Types

- **DATE –** A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31.
- **DATETIME –** A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 to 9999-12-31 23:59:59.
- **TIMESTAMP** -- A time stamp between midnight, January 1, 1970, and sometime in 2037. You can define multiple lengths to the TIMESTAMP field, which directly correlates to what is stored in it. The default length for TIMESTAMP is 14, which stores YYYYMMDDHHMMSS. Other definitions are 12(YYMMDDHHMMSS), 8(YYYYMMDD), and 6 (YYMMDD).
- **TIME** – Stores the time in HH:MM:SS format
- **YEAR(M)** – Stores a year in 2-digit or 4-digit format. If the length is specifies as 2 (for example,YEAR(2)), YEAR can be 1970to 2069 (70 to 69). The default length is 4.

III. String Types

- **CHAR(M) –** A fixed-length string between 1 and 255 characters in length. Defining a length is not required but the default is 1.
- **VARCHAR(M) –** A variable-length string between 1 and 255 characters in length; you must define a length when creating a VARCHAR field.
- **BLOB or TEXT –** A field with a max. length of 65535 characters. BLOB are “Binary Large Objects” and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also holds large amounts of data; the difference between the two is that sorts and comparisons on stored data are case-sensitive on BLOBS and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.
- **TINYBLOB or TINYTEXT –** A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
- **MEDIUMBLOB or MEDIUMTEXT --** A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
- **LONGBLOB or LONGTEXT** -- A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
- **ENUM** – An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be null). For example ENUM(‘A’, ‘B’,’C’). ENUMs can have 65535 different values.

BASIC SQL COMMANDS

SQL Data Manipulation Language (DML)

SQL (Structured Query Language) is a syntax for executing queries. But the SQL language also includes a syntax to update, insert, and delete records.

These query and update commands together form the Data Manipulation Language (DML) part of SQL:

- **SELECT** - extracts data from a database table
- **UPDATE** - updates data in a database table
- **DELETE** - deletes data from a database table
- **INSERT INTO** - inserts new data into a database table

SQL Data Definition Language (DDL)

The Data Definition Language (DDL) part of SQL permits database tables to be created or deleted. We can also define indexes (keys), specify links between tables, and impose constraints between database tables.

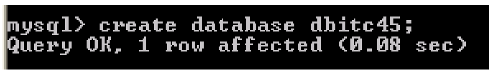
The most important DDL statements in SQL are:

- **CREATE TABLE** - creates a new database table
- **ALTER TABLE** - alters (changes) a database table
- **DROP TABLE** - deletes a database table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

Create a Database

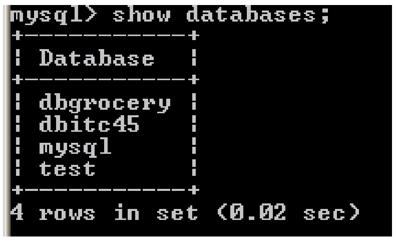
To create a database:

CREATE DATABASE database_name



```
mysql> create database dbitc45;
Query OK, 1 row affected (0.08 sec)
```

Showing/Displaying the databases



```
mysql> show databases;
+-----------+
| Database  |
+-----------+
| dbgrocery |
| dbitc45   |
| mysql     |
| test      |
+-----------+
4 rows in set (0.02 sec)
```

Deleting the Database

DROP DATABASE database_name

Using Database

It is necessary to activate or choose what database you are going to use in creating your table.

Use databasename ---→ ex. Use itc45;

Create a Table

To create a table in a database:

CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
....
)

Example

This example demonstrates how you can create a table named "Person", with four columns. The column names will be "LastName", "FirstName", "Address", and "Age":

```
CREATE TABLE Person
(
LastName varchar,
FirstName varchar,
Address varchar,
Age int
)
```

Create table named PERSONNEL with the following fields.

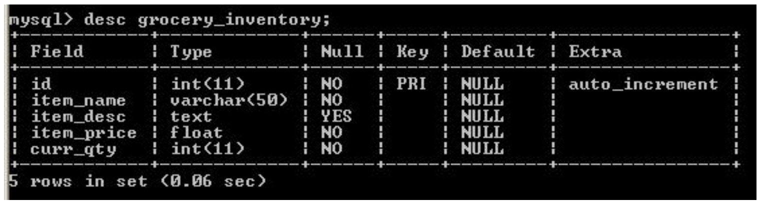
```
CREATE TABLE personnel
(
id int NOT NULL AUTO_INCREMENT,
firstname varchar(25),
lastname varchar(20),
nick varchar(12),
email varchar(35),
salary int,
PRIMARY KEY (id),
UNIQUE id (id)
);
```

This example demonstrates how you can specify a maximum length for some columns:

```
CREATE TABLE Person
(
LastName varchar(30),
FirstName varchar,
Address varchar,
Age int(3)
)
```

Displaying/Showing the structure of the table

DESC table_name or DESCRIBE table_name



```
mysql> desc grocery_inventory;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | int(11)     | NO   | PRI | NULL    | auto_increment |
| item_name  | varchar(50) | NO   |     | NULL    |                |
| item_desc  | text        | YES  |     | NULL    |                |
| item_price | float       | NO   |     | NULL    |                |
| curr_qty   | int(11)     | NO   |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
5 rows in set (0.06 sec)
```

Delete a Table

To delete a table (the table structure, attributes, and indexes will also be deleted):

DROP TABLE table_name

Delete/Truncate a Table

What if we only want to get rid of the data inside a table, and not the table itself? Use the TRUNCATE TABLE command (deletes only the data inside the table):

TRUNCATE TABLE table_name or
Delete from table_name

Using the INSERT Command

The INSERT INTO statement is used to insert new rows into a table.

Syntax

INSERT INTO table_name VALUES (value1, value2,....)

You can also specify the columns for which you want to insert data:

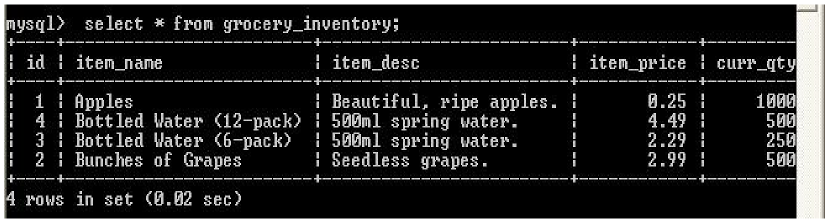
INSERT INTO table_name (column1, column2,...) VALUES (value1, value2,....)

INSERT INTO grocery_inventory (id,item_name,item_desc,item_price,curr_qty) VALUES
(4,'Bottled Water (12-pack)','500ml spring water.',4.49,500),
(3,'Bottled Water (6-pack)','500ml spring water.',2.29,250),
(2,'Bunches of Grapes','Seedless grapes.',2.99,500);
('1','Apples','Beautiful, ripe apples.','0.25',1000);

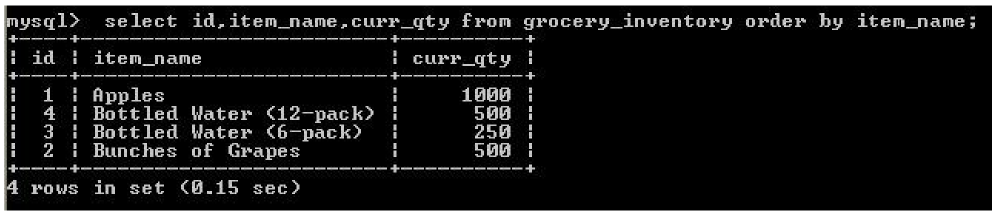
INSERT INTO table_name (“O’Connor said \”Boo\” “);

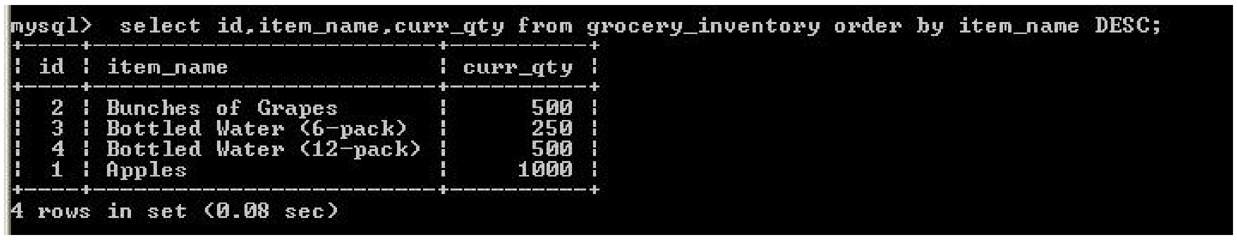
INSERT INTO table_name (‘O\’Connor said ”Boo”’);

Using the SELECT Command

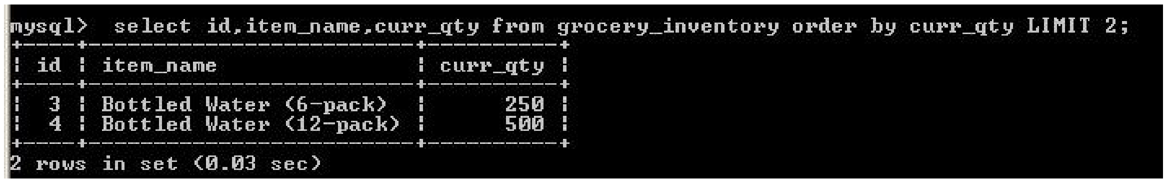

```
mysql>  select * from grocery_inventory;
+----+-------------------------+-------------------------+------------+---------
| id | item_name               | item_desc               | item_price | curr_qty
+----+-------------------------+-------------------------+------------+---------
|  1 | Apples                  | Beautiful, ripe apples. |       0.25 |     1000
|  4 | Bottled Water (12-pack) | 500ml spring water.     |       4.49 |      500
|  3 | Bottled Water (6-pack)  | 500ml spring water.     |       2.29 |      250
|  2 | Bunches of Grapes       | Seedless grapes.        |       2.99 |      500
+----+-------------------------+-------------------------+------------+---------
4 rows in set (0.02 sec)
```

Ordering SELECT Results


```
mysql>  select id,item_name,curr_qty from grocery_inventory order by item_name;
+----+-------------------------+----------+
| id | item_name               | curr_qty |
+----+-------------------------+----------+
|  1 | Apples                  |     1000 |
|  4 | Bottled Water (12-pack) |      500 |
|  3 | Bottled Water (6-pack)  |      250 |
|  2 | Bunches of Grapes       |      500 |
+----+-------------------------+----------+
4 rows in set (0.15 sec)
```

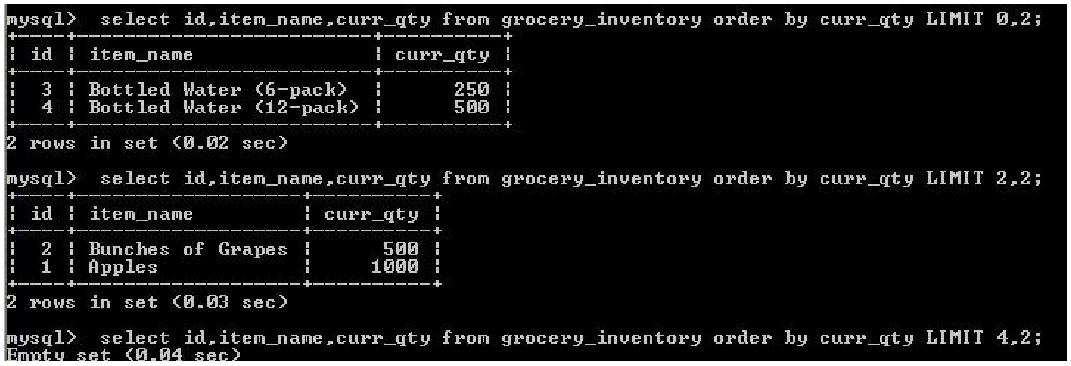

```
mysql>  select id,item_name,curr_qty from grocery_inventory order by item_name DESC;
+----+-------------------------+----------+
| id | item_name               | curr_qty |
+----+-------------------------+----------+
|  2 | Bunches of Grapes       |      500 |
|  3 | Bottled Water (6-pack)  |      250 |
|  4 | Bottled Water (12-pack) |      500 |
|  1 | Apples                  |     1000 |
+----+-------------------------+----------+
4 rows in set (0.08 sec)
```

Limiting your Results



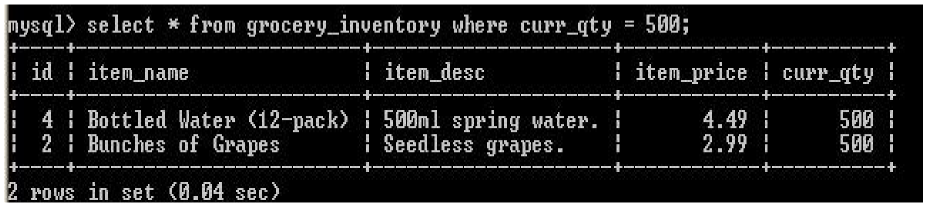
```
mysql>  select id,item_name,curr_qty from grocery_inventory order by curr_qty LIMIT 2;
+----+-------------------------+----------+
| id | item_name               | curr_qty |
+----+-------------------------+----------+
|  3 | Bottled Water (6-pack)  |      250 |
|  4 | Bottled Water (12-pack) |      500 |
+----+-------------------------+----------+
2 rows in set (0.03 sec)
```

```
SELECT * from grocery_inventory ORDER BY curr_qty LIMIT 0,2;

SELECT * from grocery_inventory ORDER BY curr_qty LIMIT 2,2;

SELECT * from grocery_inventory ORDER BY curr_qty LIMIT 4,2;
```



```
mysql>  select id,item_name,curr_qty from grocery_inventory order by curr_qty LIMIT 0,2;
+----+-------------------------+----------+
| id | item_name               | curr_qty |
+----+-------------------------+----------+
|  3 | Bottled Water (6-pack)  |      250 |
|  4 | Bottled Water (12-pack) |      500 |
+----+-------------------------+----------+
2 rows in set (0.02 sec)

mysql>  select id,item_name,curr_qty from grocery_inventory order by curr_qty LIMIT 2,2;
+----+-------------------+----------+
| id | item_name         | curr_qty |
+----+-------------------+----------+
|  2 | Bunches of Grapes |      500 |
|  1 | Apples            |     1000 |
+----+-------------------+----------+
2 rows in set (0.03 sec)

mysql>  select id,item_name,curr_qty from grocery_inventory order by curr_qty LIMIT 4,2;
Empty set (0.04 sec)
```

Using WHERE in your Queries

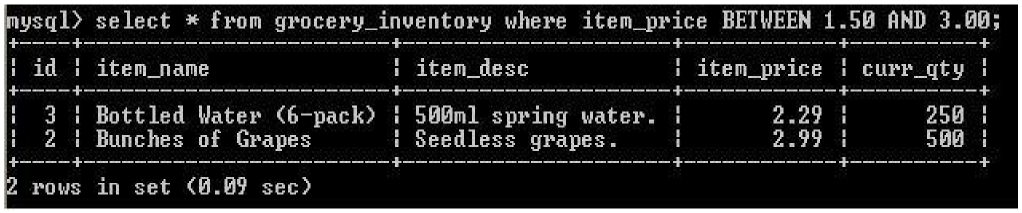


```
mysql> select * from grocery_inventory where curr_qty = 500;
+----+-------------------------+---------------------+------------+----------+
| id | item_name               | item_desc           | item_price | curr_qty |
+----+-------------------------+---------------------+------------+----------+
|  4 | Bottled Water (12-pack) | 500ml spring water. |       4.49 |      500 |
|  2 | Bunches of Grapes       | Seedless grapes.    |       2.99 |      500 |
+----+-------------------------+---------------------+------------+----------+
2 rows in set (0.04 sec)
```

Using Operators in WHERE Clauses

Table 1.0 Basic Comparison Operators and Their Meanings

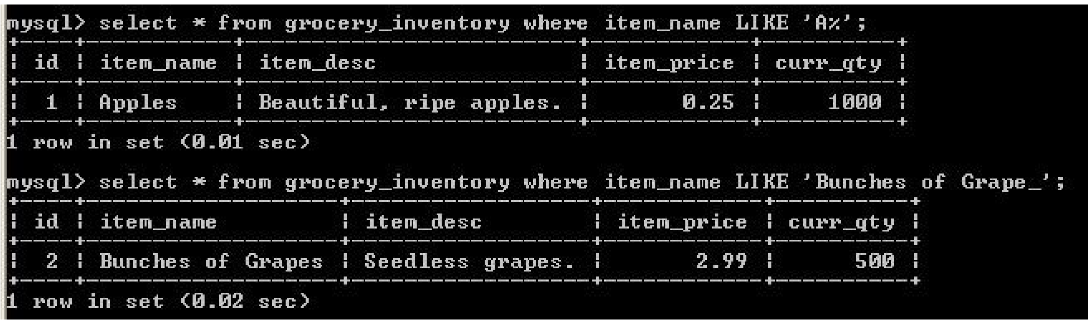
Operator	**Meaning**
=	Equal to
!=	Not equal to
<=	Less than or equal to
<	Less than
>=	Greater than or equal to
>	Greater than



```
mysql> select * from grocery_inventory where item_price BETWEEN 1.50 AND 3.00;
+----+-------------------------+---------------------+------------+----------+
| id | item_name               | item_desc           | item_price | curr_qty |
+----+-------------------------+---------------------+------------+----------+
|  3 | Bottled Water (6-pack)  | 500ml spring water. |       2.29 |      250 |
|  2 | Bunches of Grapes       | Seedless grapes.    |       2.99 |      500 |
+----+-------------------------+---------------------+------------+----------+
2 rows in set (0.09 sec)
```

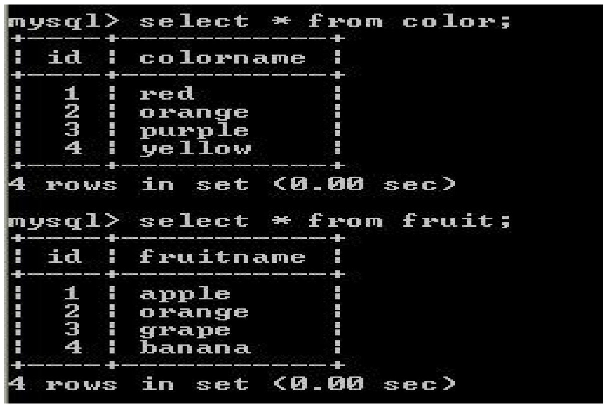
String Comparison Using LIKE

- % -- Matches multiple characters
- _ -- Matches exactly one character

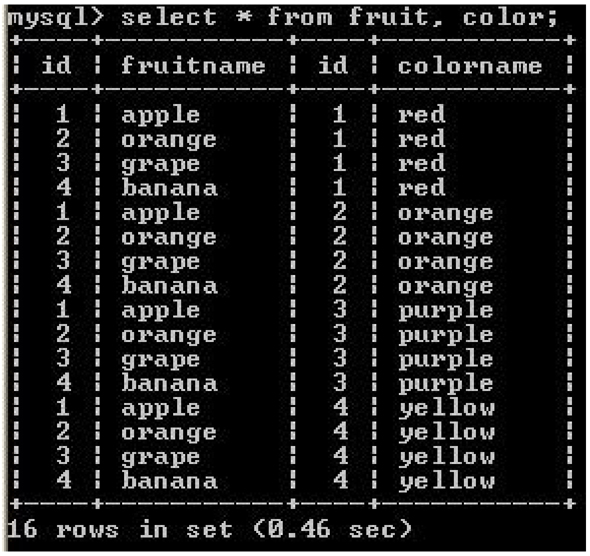


```
mysql> select * from grocery_inventory where item_name LIKE 'A%';
+----+-----------+-------------------------+------------+----------+
| id | item_name | item_desc               | item_price | curr_qty |
+----+-----------+-------------------------+------------+----------+
|  1 | Apples    | Beautiful, ripe apples. |       0.25 |     1000 |
+----+-----------+-------------------------+------------+----------+
1 row in set (0.01 sec)

mysql> select * from grocery_inventory where item_name LIKE 'Bunches of Grape_';
+----+-------------------+------------------+------------+----------+
| id | item_name         | item_desc        | item_price | curr_qty |
+----+-------------------+------------------+------------+----------+
|  2 | Bunches of Grapes | Seedless grapes. |       2.99 |      500 |
+----+-------------------+------------------+------------+----------+
1 row in set (0.02 sec)
```

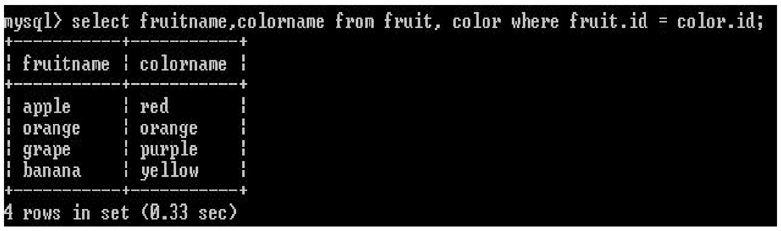
Selecting from Multiple Tables



```
mysql> select * from color;
+----+-----------+
| id | colorname |
+----+-----------+
|  1 | red       |
|  2 | orange    |
|  3 | purple    |
|  4 | yellow    |
+----+-----------+
4 rows in set (0.00 sec)

mysql> select * from fruit;
+----+-----------+
| id | fruitname |
+----+-----------+
|  1 | apple     |
|  2 | orange    |
|  3 | grape     |
|  4 | banana    |
+----+-----------+
4 rows in set (0.00 sec)
```

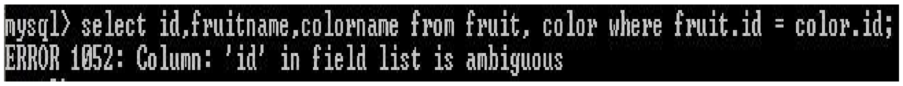


```
mysql> select * from fruit, color;
+----+-----------+----+-----------+
| id | fruitname | id | colorname |
+----+-----------+----+-----------+
|  1 | apple     |  1 | red       |
|  2 | orange    |  1 | red       |
|  3 | grape     |  1 | red       |
|  4 | banana    |  1 | red       |
|  1 | apple     |  2 | orange    |
|  2 | orange    |  2 | orange    |
|  3 | grape     |  2 | orange    |
|  4 | banana    |  2 | orange    |
|  1 | apple     |  3 | purple    |
|  2 | orange    |  3 | purple    |
|  3 | grape     |  3 | purple    |
|  4 | banana    |  3 | purple    |
|  1 | apple     |  4 | yellow    |
|  2 | orange    |  4 | yellow    |
|  3 | grape     |  4 | yellow    |
|  4 | banana    |  4 | yellow    |
+----+-----------+----+-----------+
16 rows in set (0.46 sec)
```

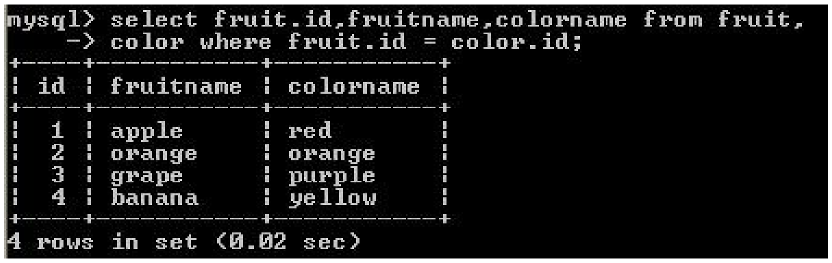


```
mysql> select fruitname,colorname from fruit, color where fruit.id = color.id;
+-----------+-----------+
| fruitname | colorname |
+-----------+-----------+
| apple     | red       |
| orange    | orange    |
| grape     | purple    |
| banana    | yellow    |
+-----------+-----------+
4 rows in set (0.33 sec)
```

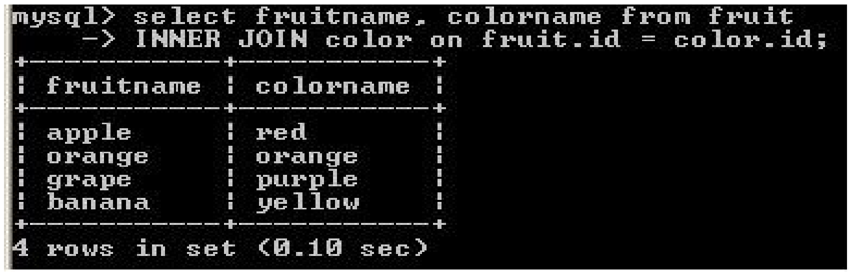
If you attempt to select a column that appears in both tables with the same name, you will get an ambiguity error:


```
mysql> select id,fruitname,colorname from fruit, color where fruit.id = color.id;
ERROR 1052: Column: 'id' in field list is ambiguous
```

If you mean to select the ID from the fruit table, you would use:

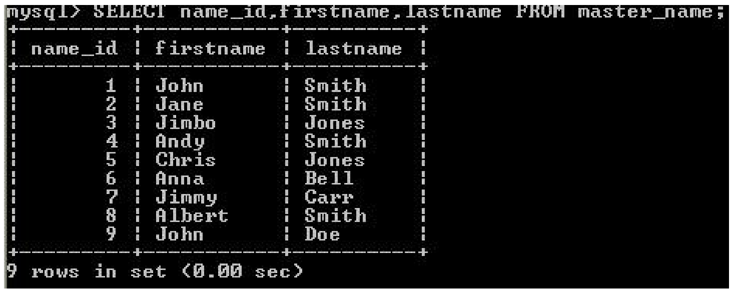

```
mysql> select fruit.id,fruitname,colorname from fruit,
    -> color where fruit.id = color.id;
+----+-----------+-----------+
| id | fruitname | colorname |
+----+-----------+-----------+
|  1 | apple     | red       |
|  2 | orange    | orange    |
|  3 | grape     | purple    |
|  4 | banana    | yellow    |
+----+-----------+-----------+
4 rows in set (0.02 sec)
```

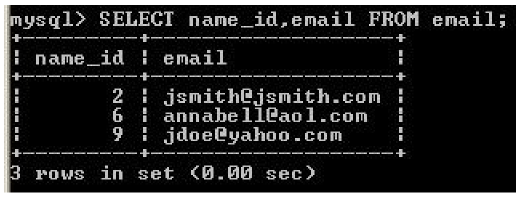
The INNER JOIN returns all rows from both tables where there is a match

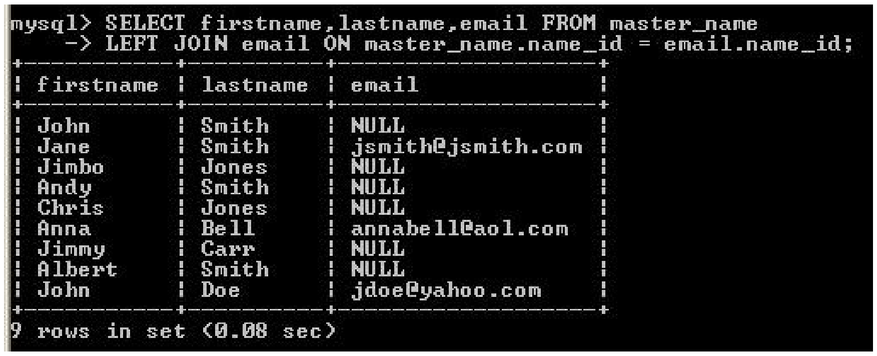

```
mysql> select fruitname, colorname from fruit
    -> INNER JOIN color on fruit.id = color.id;
+-----------+-----------+
| fruitname | colorname |
+-----------+-----------+
| apple     | red       |
| orange    | orange    |
| grape     | purple    |
| banana    | yellow    |
+-----------+-----------+
4 rows in set (0.10 sec)
```

LEFT JOIN

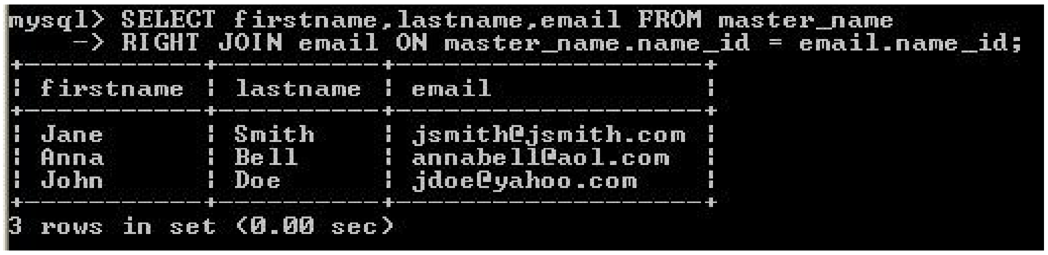
The LEFT JOIN returns all the rows from the first table, even if there are no matches in the second table


```
mysql> SELECT name_id,firstname,lastname FROM master_name;
+---------+-----------+----------+
| name_id | firstname | lastname |
+---------+-----------+----------+
|       1 | John      | Smith    |
|       2 | Jane      | Smith    |
|       3 | Jimbo     | Jones    |
|       4 | Andy      | Smith    |
|       5 | Chris     | Jones    |
|       6 | Anna      | Bell     |
|       7 | Jimmy     | Carr     |
|       8 | Albert    | Smith    |
|       9 | John      | Doe      |
+---------+-----------+----------+
9 rows in set (0.00 sec)
```


```
mysql> SELECT name_id,email FROM email;
+---------+-------------------+
| name_id | email             |
+---------+-------------------+
|       2 | jsmith@jsmith.com |
|       6 | annabell@aol.com  |
|       9 | jdoe@yahoo.com    |
+---------+-------------------+
3 rows in set (0.00 sec)
```


```
mysql> SELECT firstname,lastname,email FROM master_name
    -> LEFT JOIN email ON master_name.name_id = email.name_id;
+-----------+----------+-------------------+
| firstname | lastname | email             |
+-----------+----------+-------------------+
| John      | Smith    | NULL              |
| Jane      | Smith    | jsmith@jsmith.com |
| Jimbo     | Jones    | NULL              |
| Andy      | Smith    | NULL              |
| Chris     | Jones    | NULL              |
| Anna      | Bell     | annabell@aol.com  |
| Jimmy     | Carr     | NULL              |
| Albert    | Smith    | NULL              |
| John      | Doe      | jdoe@yahoo.com    |
+-----------+----------+-------------------+
9 rows in set (0.08 sec)
```

A RIGHT JOIN works like LEFT JOIN but with the table order reversed. In other words, all rows from the second table will be returned, no matter whether there are matches in the first table or not.



```
mysql> SELECT firstname,lastname,email FROM master_name
    -> RIGHT JOIN email ON master_name.name_id = email.name_id;
+-----------+----------+--------------------+
| firstname | lastname | email              |
+-----------+----------+--------------------+
| Jane      | Smith    | jsmith@jsmith.com  |
| Anna      | Bell     | annabell@aol.com   |
| John      | Doe      | jdoe@yahoo.com     |
+-----------+----------+--------------------+
3 rows in set (0.00 sec)
```

ALTER TABLE
The ALTER TABLE statement is used to add or drop columns in an existing table.

ALTER TABLE table_name
ADD column_name datatype

Example
To add a column named "City" in the "Person" table:
ALTER TABLE Person ADD City varchar(30)

ALTER TABLE table_name
DROP COLUMN column_name

Example
To drop the "Address" column in the "Person" table:
ALTER TABLE Person DROP COLUMN Address

ALTER TABLE table_name
MODIFY COLUMN column_name

ALTER TABLE table_name
RENAME newtablename

The Update Statement
The UPDATE statement is used to modify the data in a table.

Syntax
UPDATE table_name SET column_name = new_value

Update one Column in a Row
We want to add a first name to the person with a last name of "Rasmussen":
UPDATE Person SET FirstName = 'Nina' WHERE LastName = 'Rasmussen'

Update several Columns in a Row
We want to change the address and add the name of the city:
UPDATE Person SET Address = 'Stien 12', City = 'Stavanger' WHERE LastName = 'Rasmussen'

FREQUENTLY USED STRING FUNCTIONS IN MYSQL

Length and Concatenation Functions

```
Select LENGTH('Ako ay Pilipino');   ---------------->  15

SELECT CONCAT('My','S','QL'); ------------→ MySQL

 SELECT CONCAT(firstname,lastname)FROM master_name;
+----------------------------+
| CONCAT(firstname,lastname) |
+----------------------------+
| JohnSmith                  |
| JaneSmith                  |
| JimboJones                 |
| AndySmith                  |
| ChrisJones                 |
| AnnaBell                   |
| JimmyCarr                  |
| AlbertSmith                |
| JohnDoe                    |
+----------------------------+
9 rows in set (0.02 sec)
=======================================================
 SELECT CONCAT('firstname','lastname')FROM master_name;
+--------------------------------+
| CONCAT('firstname','lastname') |
+--------------------------------+
| firstnamelastname              |
| firstnamelastname              |
| firstnamelastname              |
| firstnamelastname              |
| firstnamelastname              |
| firstnamelastname              |
| firstnamelastname              |
| firstnamelastname              |
| firstnamelastname              |
+--------------------------------+
9 rows in set (0.00 sec)
========================================================
SELECT CONCAT_WS(' ',firstname,lastname)AS Fullname FROM master_name;
+--------------+
| Fullname     |
+--------------+
| John Smith   |
| Jane Smith   |
| Jimbo Jones  |
| Andy Smith   |
| Chris Jones  |
| Anna Bell    |
| Jimmy Carr   |
| Albert Smith |
| John Doe     |
+--------------+
9 rows in set (0.00 sec)
```

Trimming and Padding Functions

```
Select RTRIM('itc45itc45   ');
+------------------------+
| RTRIM('itc45itc45   ') |
+------------------------+
| itc45itc45             |
+------------------------+
1 row in set (0.03 sec)
==========================================
Select LTRIM('  itc45itc45');
+-----------------------+
| LTRIM('  itc45itc45') |
+-----------------------+
| itc45itc45            |
+-----------------------+
1 row in set (0.00 sec)
==========================================
Select TRIM(LEADING 'X' from 'XXXneedeleXXX');
+----------------------------------------+
| TRIM(LEADING 'X' from 'XXXneedeleXXX') |
+----------------------------------------+
| needeleXXX                             |
+----------------------------------------+
1 row in set (0.00 sec)
=============================================
Select TRIM(TRAILING 'X' from 'XXXneedleXXX');
+----------------------------------------+
| TRIM(TRAILING 'X' from 'XXXneedleXXX') |
+----------------------------------------+
| XXXneedle                              |
+----------------------------------------+
1 row in set (0.00 sec)
==============================================
Select TRIM('X' from 'XXXneedleXXX');
+-------------------------------+
| TRIM('X' from 'XXXneedleXXX') |
+-------------------------------+
| needle                        |
+-------------------------------+
1 row in set (0.00 sec)
===============================================
select LCASE('MYSQL');
+----------------+
| LCASE('MYSQL') |
+----------------+
| mysql          |
+----------------+
1 row in set (0.00 sec)
==============================================
select UCASE('mysql');
+----------------+
| UCASE('mysql') |
+----------------+
| MYSQL          |
```