

CIND-110
DATA ORGANIZATION FOR DATA ANALYSTS

MODULE 3
BASIC SQL OPERATIONS

LEAD INSTRUCTOR: TAMER ABDOU, PhD

Objective

To use basic operations to retrieve and modify data. To implement constraints, schema, queries and operations in SQL.

TABLE OF CONTENTS

1. OPENING MYSQL.....	3
2. CREATING A DATABASE.....	3
3. CREATING TABLES.....	5
4. INSERTING DATA	7
5. ALTERING TABLES	9
6. ADDING MULTIPLE COLUMNS IN SINGLE COMMAND	10
7. UPDATE COMMAND.....	10
8. MODIFY COMMAND.....	11
9. CREATING IDENTICAL TABLES	12
10. ANOTHER FORM OF CREATE.....	12
11. DELETE COMMAND.....	13
12. MORE COMMANDS	13
13. REVISITING CONSTRAINTS.....	14

1. OPENING MYSQL

Open Terminal in your Ubuntu VM terminal and type out the following command to start mysql

```
$ mysql -u root -p
```

In order to be able to add and manipulate data, you first have to create a database. When creating tables, you must also decide on the structure of each table: the number of columns, the type of data each column may hold, how the tables will be indexed, and several other factors. There are a few basic things to decide when creating a structure for your data:

- The number of tables to include in your database, as well as the table names
- For each table, the number of columns it should contain, as well as the column names
- For each column, what kind of data is to be stored

2. CREATING A DATABASE

Creating a database is simple, mostly because there's nothing much to it. Use the SQL statement CREATE DATABASE. You will have to provide a name for the database with this SQL statement. To do this, enter the following from within the *mysql* client:

```
CREATE DATABASE rookery;
```

- This SQL statement will create a subdirectory called 'rookery' on the file system in the data directory for MySQL.
- It will not create any data. It will just set up a place to add tables, which will in turn hold data.
- Let's drop the 'rookery' database and create it again like so:

```
DROP DATABASE rookery;
```

- To get a list of databases, enter the following SQL statement:

```
SHOW DATABASES;
```

- Incidentally, if you don't like the keyword DATABASE, you can use SCHEMA instead. The results are the same.

```
CREATE SCHEMA database_name
```

The results below show the rookery database, and three other databases that were created when MySQL was installed on the server.

```
mysql
performance_schema
sys
-----+-----
5 rows in set (0.00 sec)

mysql> create schema rookery;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
-----+-----
Database
-----+-----
information_schema
MyFirstSchema
UNIVERSITY
mysql
performance_schema
rookery
sys
-----+-----
7 rows in set (0.00 sec)

mysql>
```

- Before beginning to add tables to the rookery database, enter the following command into the *mysql* client:

USE rookery;

This command will set the new database that was just created as the default database for the *mysql* client. It will remain the default database until you change it to a different one or until you exit the client. This makes it easier when entering SQL statements to create tables or other SQL statements related to tables. Otherwise, when you enter each table related SQL statement, you would have to specify each time the database where the table is located.

3. CREATING TABLES

The next step for structuring a database is to create tables. Enter the following SQL statement into *mysql*:

```
CREATE TABLE birds (bird_id INT AUTO_INCREMENT  
PRIMARY KEY, scientific_name VARCHAR(255)  
UNIQUE, common_name VARCHAR(50), family_id INT,  
description TEXT);
```

The above SQL statement creates the table *birds* with five fields, or columns, with commas separating the information about each column. The *AUTO_INCREMENT* option tells MySQL to automatically increment the value of this field. It will start with the number 1, unless we specify a different number.

```
UNIVERSITY |  
mysql      |  
performance_schema |  
rookery    |  
sys        |  
+-----+  
7 rows in set (0.00 sec)  
  
mysql> drop database rookery;  
Query OK, 0 rows affected (0.02 sec)  
  
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| MyFirstSchema      |  
| UNIVERSITY         |  
| mysql              |  
| performance_schema |  
| sys                |  
+-----+  
6 rows in set (0.00 sec)  
  
mysql> create schema rookery;  
Query OK, 1 row affected (0.00 sec)  
  
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| MyFirstSchema      |  
| UNIVERSITY         |  
| mysql              |  
| performance_schema |  
| rookery            |  
| sys                |  
+-----+  
7 rows in set (0.00 sec)  
  
mysql> use rookery;  
Database changed  
mysql> CREATE TABLE BIRDS(  
-> bird_id INT AUTO_INCREMENT PRIMARY KEY,  
-> scientific_name VARCHAR(255) UNIQUE,  
-> common_name VARCHAR(50),  
-> family_id INT,  
-> description TEXT);  
Query OK, 0 rows affected (0.07 sec)  
  
mysql>
```

To see how the table looks, use the *DESCRIBE* statement. It displays information about the columns of a table, or the table schema—not the data itself. To use this SQL statement to get information on the table we just created, you should enter the following SQL statement:

```
DESCRIBE birds;
```

```
mysql> DESCRIBE BIRDS;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| bird_id        | int(11)        | NO   | PRI | NULL    | auto_increment |
| scientific_name | varchar(255)   | YES  | UNI | NULL    |                |
| common_name    | varchar(50)    | YES  |     | NULL    |                |
| family_id      | int(11)        | YES  |     | NULL    |                |
| decsription    | text           | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql>
```

Notice that these results are displayed in a table format made with ASCII characters.

Description of the rows:

- ◇ The first row of this results set contains column headings describing the rows of information that follow it.
- ◇ In the first column of this results set, *Field* contains the fields or columns of the table created.
- ◇ The second column, *Type*, lists the data type for each field. Notice that for the table’s columns in which we specified the data type VARCHAR with the specific widths within parentheses, those settings are shown here (e.g., VARCHAR(255)). Where we didn’t specify the size for the INT columns, the defaults were assumed and are shown here.
- ◇ The third column in the preceding results, *Null*, indicates whether each field may contain NULL values. NULL is nothing; it’s nonexistent data or missing data (in some cases). This is different from blank or empty content in a field.
- ◇ The fourth column, *Key*, indicates whether a field is a key field—an indexed column.
- ◇ The bird_id column is a primary key, shortened to PRI in this display.
- ◇ We set scientific_name to another type of key or index, one called UNIQUE, which is abbreviated UNI here.
- ◇ The next-to-last column in the display, *Default*, would contain any default value set for each field.
- ◇ The last column, *Extra*, provides any extra information the table maintains on each column. In the example shown, we can see that the values for bird_id will be incremented automatically.

Besides the DESCRIBE statement, there's another way to look at how a table is structured. You can use the SHOW CREATE TABLE statement. It shows the default settings assumed by the server, ones that you might not have specified when you ran the CREATE TABLE statement.

- Let's run the command:

```
SHOW CREATE TABLE birds \G
```

```
Database changed
mysql> SHOW CREATE TABLE BIRDS \G
***** 1. row *****
      Table: BIRDS
Create Table: CREATE TABLE `BIRDS` (
  `bird_id` int(11) NOT NULL AUTO_INCREMENT,
  `scientific_name` varchar(255) DEFAULT NULL,
  `common_name` varchar(50) DEFAULT NULL,
  `family_id` int(11) DEFAULT NULL,
  `decsription` text,
  PRIMARY KEY (`bird_id`),
  UNIQUE KEY `scientific_name` (`scientific_name`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

mysql>
```

4. INSERTING DATA

- Enter the following on your server using the *mysql* client:

```
INSERT INTO birds (scientific_name, common_name) VALUES ('Charadrius', 'Killdeer'),
('Gavia', 'Great Northern Loon'), ('Aix', 'Wood Duck'), ('Chordeiles', 'Common Nighthawk'),
('Sitta', 'White-breasted Nuthatch'), ('Apteryx', 'North Island Brown Kiwi');
```

This will create six rows of data for six birds.

- Now enter the following from the *mysql* client to see the contents of the table:

```
SELECT * FROM birds;
```

```
mysql> SELECT * FROM BIRDS;
+-----+-----+-----+-----+-----+
| bird_id | scientific_name | common_name | family_id | decsription |
+-----+-----+-----+-----+-----+
| 1 | Charardrius | Killdeer | NULL | NULL |
| 2 | Gavia | Great Northern loon | NULL | NULL |
| 3 | Aix | Wood Duck | NULL | NULL |
| 4 | Chordeiles | Common Nithhawk | NULL | NULL |
| 5 | Sitta | White-breasted Nuthatch | NULL | NULL |
| 6 | Apteryx | North Island Brown Kiwi | NULL | NULL |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

As you can see from the results, MySQL put values in the two columns we gave it, and set the other columns to their default values (i.e., NULL).

- Let's create another database that contains information about people who are interested in bird-watching and insert values in that table:



```
CREATE DATABASE birdwatchers;
```

```
CREATE TABLE birdwatchers.humans (human_id INT AUTO_INCREMENT PRIMARY KEY,  
formal_title VARCHAR(25), name_first VARCHAR(25), name_last VARCHAR(25),  
email_address VARCHAR(255));
```

```
INSERT INTO birdwatchers.humans (formal_title,name_first, name_last, email_address) VALUES  
( 'Mr.', 'Russell', 'Dyer', 'russell@mysqlresources.com'), ('Mr.', 'Richard', 'Stringer',  
'richard@mysqlresources.com'), ('Ms.', 'Rusty', 'Osborne', 'rusty@mysqlresources.com'), ('Ms.',  
'Lexi', 'Hollar', 'alexandra@mysqlresources.com');
```

```
mysql> insert into HUMANS(formal_title,name_first,name_last,email_address) values ('Mr','Russel','Dyer','russel@mysqlresources.com'),('Mr','Richard','S  
tringer','richard@mysqlresources.com'),('Ms','Rusty','Osborne','rusty@mysqlresources.com'),('Ms','Lexi','Hollar','alexander@mysqlresources.com');  
ERROR 1046 (3D000): No database selected  
mysql> use birdwatchers;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
  
Database changed  
mysql> insert into HUMANS(formal_title,name_first,name_last,email_address) values ('Mr','Russel','Dyer','russel@mysqlresources.com'),('Mr','Richard','S  
tringer','richard@mysqlresources.com'),('Ms','Rusty','Osborne','rusty@mysqlresources.com'),('Ms','Lexi','Hollar','alexander@mysqlresources.com');  
Query OK, 4 rows affected (0.00 sec)  
Records: 4 Duplicates: 0 Warnings: 0  
  
mysql>
```

- The next table we'll create is bird_families. This will hold information about bird families, which are groupings of birds. This will tie into the family_id column in the birds table.

```
CREATE TABLE bird_families (family_id INT AUTO_INCREMENT PRIMARY KEY,  
scientific_name VARCHAR(255) UNIQUE, brief_description VARCHAR(255) );
```

- Let us next create a table for information about the orders of the birds. This is a grouping of families of birds. We will name it bird_orders. Enter the following SQL statement:

```
CREATE TABLE bird_orders (order_id INT AUTO_INCREMENT PRIMARY KEY,  
scientific_name VARCHAR(255) UNIQUE, brief_description VARCHAR(255),  
order_image BLOB) DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci;
```


Notice that for this image file, the data type we are using is a BLOB. It stands for *binary large object*. We can store an image file, such as a JPEG file, in the column, but it can make the table large, which can be a problem when backing up the database.

It might be better to store the image files on the server and then store a file path or URL address in the database, pointing to where the image file is located.

After the list of columns, we have included the default character set and collation to be used when creating the columns. We are using UTF-8 (i.e., UCS Transformation Format, 8-bit), because some of the names may include characters that are not part of the default latin1 character set.

5. ALTERING TABLES

- The command below adds another column named 'order_id' of type integer to the bird_families table.

```
ALTER TABLE bird_families ADD COLUMN order_id INT;
```

- The above command adds another column named wing_id of character type to the 'birds' table.

```
ALTER TABLE birds ADD COLUMN wing_id CHAR(2)
```

- Check the result by giving the DESCRIBE command

```
mysql> ALTER TABLE BIRD_FAMILIES ADD COLUMN order_id INT;
Query OK, 0 rows affected (0.13 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE BIRDS ADD COLUMN wing_id CHAR(2);
Query OK, 0 rows affected (0.12 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> DESCRIBE BIRD_FAMILIES;
+-----+-----+-----+-----+-----+-----+
| Field                | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| family_id            | int(11)       | NO   | PRI | NULL    | auto_increment |
| scientific_name       | varchar(255)  | YES  | UNI | NULL    |                |
| brief_description    | varchar(255)  | YES  |     | NULL    |                |
| order_id             | int(11)       | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> DESCRIBE BIRDS;
+-----+-----+-----+-----+-----+-----+
| Field                | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| bird_id              | int(11)       | NO   | PRI | NULL    | auto_increment |
| scientific_name       | varchar(255)  | YES  | UNI | NULL    |                |
| common_name          | varchar(50)   | YES  |     | NULL    |                |
| family_id            | int(11)       | YES  |     | NULL    |                |
| decscription         | text          | YES  |     | NULL    |                |
| wing_id              | char(2)       | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

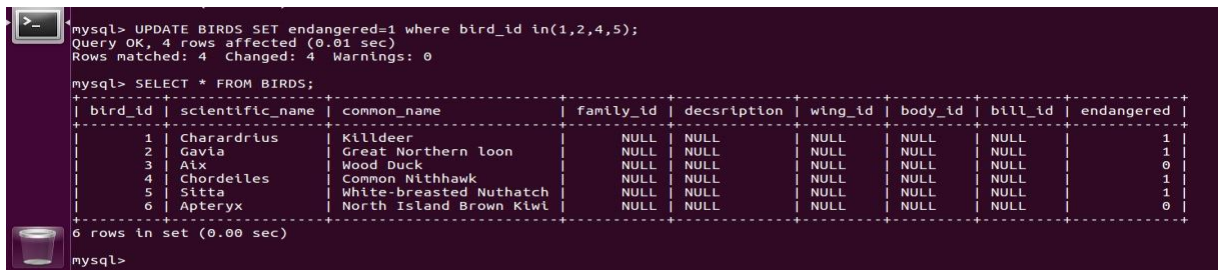
6. ADDING MULTIPLE COLUMNS IN SINGLE COMMAND

```
ALTER TABLE birds ADD COLUMN body_id CHAR(2) AFTER wing_id, ADD COLUMN bill_id CHAR(2) AFTER body_id, ADD COLUMN endangered BOOLEAN DEFAULT 0 AFTER bill_id, CHANGE COLUMN common_name common_name VARCHAR(255);
```

In the CHANGE COLUMN clause, notice that we listed the name of the common_name column twice. The first time is to name the column that is to be changed. The second time is to provide the new name, if we wanted to change it.

7. UPDATE COMMAND

```
UPDATE birds SET endangered = 1 WHERE bird_id IN(1,2,4,5);
```



```
mysql> UPDATE BIRDS SET endangered=1 where bird_id in(1,2,4,5);
Query OK, 4 rows affected (0.01 sec)
Rows matched: 4  Changed: 4  Warnings: 0

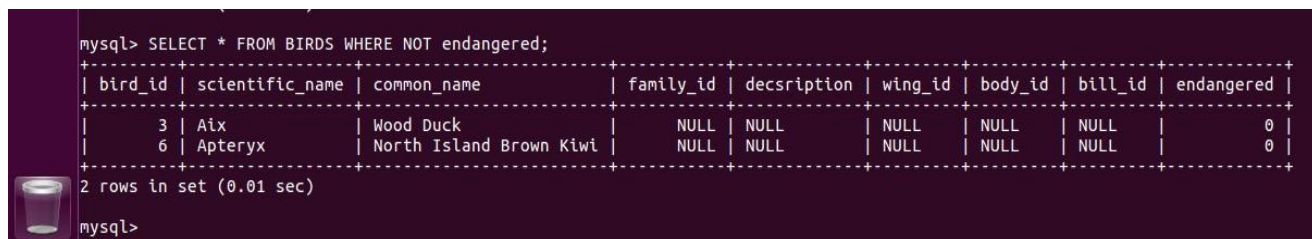
mysql> SELECT * FROM BIRDS;
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| bird_id | scientific_name | common_name | family_id | description | wing_id | body_id | bill_id | endangered |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Charadrius | Killdeer | NULL | NULL | NULL | NULL | NULL | 1 |
| 2 | Gavia | Great Northern loon | NULL | NULL | NULL | NULL | NULL | 1 |
| 3 | Aix | Wood Duck | NULL | NULL | NULL | NULL | NULL | 0 |
| 4 | Chordeiles | Common Nighthawk | NULL | NULL | NULL | NULL | NULL | 1 |
| 5 | Sitta | White-breasted Nuthatch | NULL | NULL | NULL | NULL | NULL | 1 |
| 6 | Apteryx | North Island Brown Kiwi | NULL | NULL | NULL | NULL | NULL | 0 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Observe the output by giving the SELECT * FROM birds command. The endangered column values for the records with bird_id as 1,2,4,5 got updated to 1.

- Now let's query all records which have birds which are NOT in endangered state.

```
SELECT * FROM birds WHERE NOT endangered;
```



```
mysql> SELECT * FROM BIRDS WHERE NOT endangered;
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| bird_id | scientific_name | common_name | family_id | description | wing_id | body_id | bill_id | endangered |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 3 | Aix | Wood Duck | NULL | NULL | NULL | NULL | NULL | 0 |
| 6 | Apteryx | North Island Brown Kiwi | NULL | NULL | NULL | NULL | NULL | 0 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql>
```

8. MODIFY COMMAND

```
ALTER TABLE birds MODIFY COLUMN endangered ENUM('Extinct', 'Extinct in Wild', 'Threatened - Critically Endangered', 'Threatened - Endangered', 'Threatened - Vulnerable', 'Lower Risk - Conservation Dependent', 'Lower Risk - Near Threatened', 'Lower Risk - Least Concern') AFTER family_id;
```

- Try the following command to observe the change:

```
SHOW COLUMNS FROM birds LIKE 'endangered'
```

```
mysql> SHOW COLUMNS FROM BIRDS LIKE 'endangered';
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| endangered | enum('Extinct','Extinct In Wild','Threatened-Critically Endangered','Threatened-Endangered','Threatened-Vulnerable','Lower Risk-Conservation Dependent','Lower Risk-Near Threatened','Lower Risk-Least Concerned') | YES | | NULL | |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

- Run the following command and observe that all the birds in the table are set to 'Lower Risk-Near Threatened'.

```
UPDATE birds SET endangered = 7;
```

```
mysql> UPDATE BIRDS SET endangered=7;
Query OK, 6 rows affected (0.01 sec)
Rows matched: 6 Changed: 6 Warnings: 0

mysql> SELECT * FROM BIRDS;
+-----+-----+-----+-----+-----+-----+-----+-----+
| bird_id | scientific_name | common_name | family_id | description | wing_id | body_id | bill_id | endangered |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Charadrius | Killdeer | NULL | NULL | NULL | NULL | NULL | Lower Risk-Near Threatened |
| 2 | Gavia | Great Northern loon | NULL | NULL | NULL | NULL | NULL | Lower Risk-Near Threatened |
| 3 | Aix | Wood Duck | NULL | NULL | NULL | NULL | NULL | Lower Risk-Near Threatened |
| 4 | Chordeiles | Common Nighthawk | NULL | NULL | NULL | NULL | NULL | Lower Risk-Near Threatened |
| 5 | Sitta | White-breasted Nuthatch | NULL | NULL | NULL | NULL | NULL | Lower Risk-Near Threatened |
| 6 | Apteryx | North Island Brown Kiwi | NULL | NULL | NULL | NULL | NULL | Lower Risk-Near Threatened |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

```
ALTER TABLE birds CHANGE COLUMN endangered conservation_status_id INT DEFAULT 8;
```

```
mysql> ALTER TABLE BIRDS CHANGE COLUMN endangered conservations_status_id INT DEFAULT 8;
Query OK, 6 rows affected (0.10 sec)
Records: 6 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM BIRDS;
```

bird_id	scientific_name	common_name	family_id	description	wing_id	body_id	bill_id	conservations_status_id
1	Charadrius	Killdeer	NULL	NULL	NULL	NULL	NULL	7
2	Gavia	Great Northern loon	NULL	NULL	NULL	NULL	NULL	7
3	Aix	Wood Duck	NULL	NULL	NULL	NULL	NULL	7
4	Chordeiles	Common Nighthawk	NULL	NULL	NULL	NULL	NULL	7
5	Sitta	White-breasted Nuthatch	NULL	NULL	NULL	NULL	NULL	7
6	Apteryx	North Island Brown Kiwi	NULL	NULL	NULL	NULL	NULL	7

```
6 rows in set (0.00 sec)

mysql>
```

There is a particular usage of the DROP keyword doesn't delete data in the columns. It just alters the column settings so there is no default value

```
ALTER TABLE birds
ALTER conservation_status_id DROP DEFAULT;
```

- Now let's run the following command:

```
SELECT auto_increment FROM information_schema.tables WHERE table_name = 'birds'
```

- The output:

```
+-----+
| auto_increment |
+-----+
|          7     |
+-----+
```

Because we entered data for only six birds in the birds table, and the value of AUTO_INCREMENT was not set when the table was created, it started at 1 and now has a value of 7. That means the *next* row we add to the table will have 7 in the column.

```
ALTER TABLE birds AUTO_INCREMENT = 10;
```

The above command will cause the bird_id to be set to 10 for the next row of data on a bird that we enter into the birds table

9. CREATING IDENTICAL TABLES

```
CREATE TABLE birds_new LIKE birds;
```

10. ANOTHER FORM OF CREATE

```
CREATE TABLE birds_details SELECT bird_id, description FROM birds;
```

11. DELETE COMMAND

- First let's insert values:

```
INSERT INTO bird_families VALUES(100, 'Gaviidae', "Loons or divers are aquatic birds found mainly in the Northern Hemisphere.", 103);
```

- Now let's delete this value:

```
DELETE FROM bird_families WHERE family_id = 100;
```

This will delete only one row: the one where the family_id equals 100. Be careful with the DELETE statement. There's no UNDO statement, per se, when working with the data like this. If you don't include the WHERE clause, you will delete all of the data in the table.

12. MORE COMMANDS

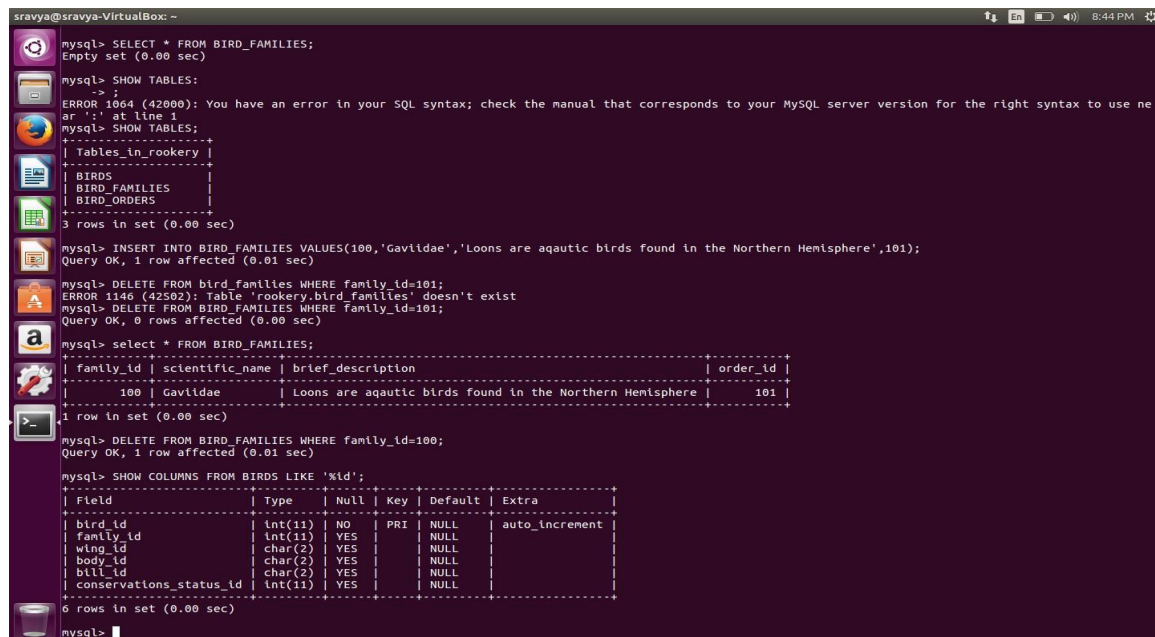
ORDER BY-

```
SELECT family_id, scientific_name FROM bird_families ORDER BY scientific_name;
```

LIKE-

```
SHOW COLUMNS FROM birds LIKE '%id';
```

You will get a list of reference columns—columns that we labeled with the ending, _id.



```
mysql> SELECT * FROM BIRD_FAMILIES;
Empty set (0.00 sec)

mysql> SHOW TABLES;
+-----+
| Tables_in_rookery |
+-----+
| BIRDS              |
| BIRD_FAMILIES      |
| BIRD_ORDERS        |
+-----+
3 rows in set (0.00 sec)

mysql> INSERT INTO BIRD_FAMILIES VALUES(100, 'Gaviidae', 'Loons are aquatic birds found in the Northern Hemisphere', 101);
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM bird_families WHERE family_id=101;
ERROR 1146 (42502): Table 'rookery.bird_families' doesn't exist
mysql> DELETE FROM BIRD_FAMILIES WHERE family_id=101;
Query OK, 0 rows affected (0.00 sec)

mysql> select * FROM BIRD_FAMILIES;
+-----+-----+-----+-----+
| family_id | scientific_name | brief_description | order_id |
+-----+-----+-----+-----+
| 100      | Gaviidae       | Loons are aquatic birds found in the Northern Hemisphere | 101      |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> DELETE FROM BIRD_FAMILIES WHERE family_id=100;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW COLUMNS FROM BIRDS LIKE '%id';
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| bird_id | int(11) | NO | PRI | NULL | auto_increment |
| family_id | int(11) | YES | | NULL | |
| wing_id | char(2) | YES | | NULL | |
| body_id | char(2) | YES | | NULL | |
| bill_id | char(2) | YES | | NULL | |
| conservations_status_id | int(11) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

13. REVISITING CONSTRAINTS

CONSTRAINT	DESCRIPTION
NOT NULL	In MySQL NOT NULL constraint allows to specify that a column can not contain any NULL value. MySQL NOT NULL can be used to CREATE and ALTER a table.
UNIQUE	The UNIQUE constraint in MySQL does not allow to insert a duplicate value in a column. The UNIQUE constraint maintains the uniqueness of a column in a table. More than one UNIQUE column can be used in a table.
PRIMARY KEY	A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column and this constraint create a unique index for accessing the table faster.
FOREIGN KEY	A FOREIGN KEY in MySQL creates a link between two tables by one specific column of both table. The specified column in one table must be a PRIMARY KEY and referred by the column of another table known as FOREIGN KEY.
DEFAULT	In a MySQL table, each column must contain a value (including a NULL). While inserting data into a table, if no value is supplied to a column, then the column gets the value set as DEFAULT.

Null Constraint:

```
CREATE DATABASE bookshop;
```

```
USE bookshop;
```

```
CREATE TABLE newauthor (aut_id varchar(8) NOT NULL, aut_name varchar(50) NOT NULL,  
country varchar(25) NOT NULL, home_city varchar(25) NOT NULL );
```



```

Copyright (c) 2000, 2016, 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> create table NewAuthor
  -> (auth_id varchar(8) NOT NULL,
  -> auth_name varchar(8) NOT NULL,
  -> country varchar(25) NOT NULL,
  -> home_city varchar(25) NOT NULL);
ERROR 1046 (3D000): No database selected
mysql> create database bookshop;
Query OK, 1 row affected (0.02 sec)

mysql> use bookshop;
Database changed
mysql> create table NewAuthor (auth_id varchar(8) NOT NULL, auth_name varchar(8)
  NOT NULL, country varchar(25) NOT NULL, home_city varchar(25) NOT NULL);
Query OK, 0 rows affected (0.12 sec)

mysql>

```

- Execute the following Insert Command:

```
INSERT INTO newauthor VALUES(null,'Sravya','Canada','Toronto');
```

Observe the error message.

```

owners.

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

mysql> create table NewAuthor
  -> (auth_id varchar(8) NOT NULL,
  -> auth_name varchar(8) NOT NULL,
  -> country varchar(25) NOT NULL,
  -> home_city varchar(25) NOT NULL);
ERROR 1046 (3D000): No database selected
mysql> create database bookshop;
Query OK, 1 row affected (0.02 sec)

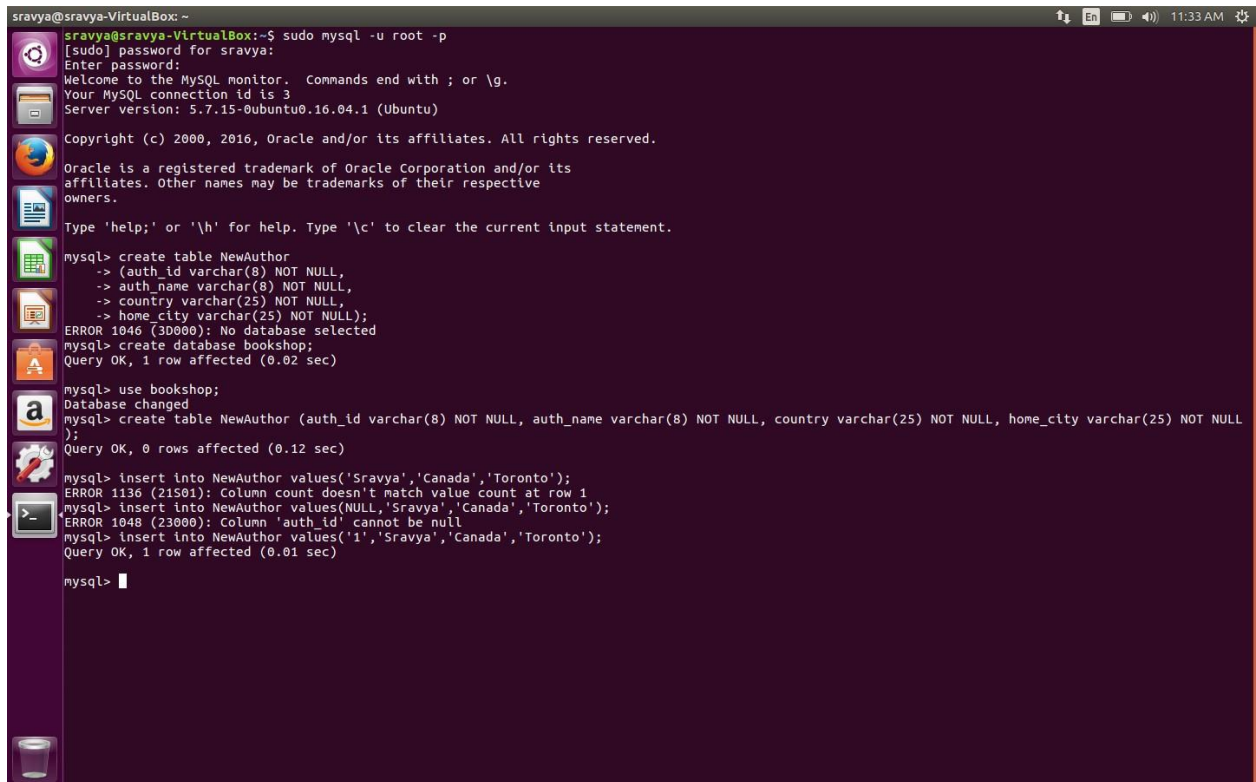
mysql> use bookshop;
Database changed
mysql> create table NewAuthor (auth_id varchar(8) NOT NULL, auth_name varchar(8)
  NOT NULL, country varchar(25) NOT NULL, home_city varchar(25) NOT NULL);
Query OK, 0 rows affected (0.12 sec)

mysql> insert into NewAuthor values('Sravya','Canada','Toronto');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> insert into NewAuthor values(NULL,'Sravya','Canada','Toronto');
ERROR 1048 (23000): Column 'auth_id' cannot be null
mysql>

```

- Now type the following command:

```
INSERT INTO newauthor VALUES(1,'Sravya','Canada','Toronto');
```



```
sravya@sravya-VirtualBox: ~  
sravya@sravya-VirtualBox:~$ sudo mysql -u root -p  
[sudo] password for sravya:  
Enter password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 3  
Server version: 5.7.15-0ubuntu0.16.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2016, 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> create table NewAuthor  
-> (auth_id varchar(8) NOT NULL,  
-> auth_name varchar(8) NOT NULL,  
-> country varchar(25) NOT NULL,  
-> home_city varchar(25) NOT NULL);  
ERROR 1046 (3D000): No database selected  
mysql> create database bookshop;  
Query OK, 1 row affected (0.02 sec)  
  
mysql> use bookshop;  
Database changed  
mysql> create table NewAuthor (auth_id varchar(8) NOT NULL, auth_name varchar(8) NOT NULL, country varchar(25) NOT NULL, home_city varchar(25) NOT NULL  
);  
Query OK, 0 rows affected (0.12 sec)  
  
mysql> insert into NewAuthor values('Sravya','Canada','Toronto');  
ERROR 1136 (21S01): Column count doesn't match value count at row 1  
mysql> insert into NewAuthor values(NULL,'Sravya','Canada','Toronto');  
ERROR 1048 (23000): Column 'auth_id' cannot be null  
mysql> insert into NewAuthor values('1','Sravya','Canada','Toronto');  
Query OK, 1 row affected (0.01 sec)  
  
mysql>
```

Observe that the data got inserted. Check the data by giving the ‘SELECT’ command.

Unique Constraint:

The UNIQUE constraint creates an index such that, all values in the index column must be unique. An error occurs when anybody tries to add a new row with a key value that already exists in that row.

EXAMPLE:

The MySQL statement stated below will create a table 'newauthor' with a column 'aut_id' which will store unique values only since UNIQUE (aut_id) is used.

```
CREATE TABLE IF NOT EXISTS  
newauthor(aut_id varchar(8) NOT NULL ,  
aut_name varchar(50) NOT NULL, country  
varchar(25) NOT NULL, home_city  
varchar(25) NOT NULL , UNIQUE (aut_id));
```

Creating a table with the same name as an existing table (i.e. newauthor) will typically generate an error. In this case, including the phrase IF NOT EXISTS in the CREATE TABLE statement will prevent the error message from occurring, but will not generate the table unless the original table with that name has first been dropped

The above command can also be given like this:

```
CREATE TABLE newauthor (aut_id varchar(8) NOT NULL UNIQUE , aut_name varchar(50)  
NOT NULL, country varchar(25) NOT NULL, home_city varchar(25) NOT NULL);
```

Note: Try to insert duplicate values into the auth_id column and observe the result.

Default Constraint:

The MySQL statement also sets the default value white space for pub_id, pub_name, pub_city columns and 'India' as default value for country column.

```
CREATE TABLE newpublisher (pub_id varchar(8) NOT NULL UNIQUE DEFAULT  
",pub_name varchar(50) NOT NULL DEFAULT " , pub_city varchar(25) NOT NULL  
DEFAULT " , country varchar(25) NOT NULL DEFAULT 'India',country_office varchar(25),  
no_of_branch int(3), estd date, PRIMARY KEY (pub_id));
```

Notice that each of the circled defaults consists of two single-quote characters (i.e. ' '), not of one double-quote character (i.e. ").

Insert data into the table with no value for the 'country' column. Observe that the default value inserted would be 'India'.

Auto-Increment:

MySQL allows you to set AUTO_INCREMENT to a column. Doing so will increase the value of that column by 1 automatically, each time a new record is added.

EXAMPLE

The MySQL statement stated below will create a table 'newauthor' with a PRIMARY KEY on 'id' column and the 'id' column is an auto incremented field.

```
CREATE TABLE newauthor (id int NOT NULL AUTO_INCREMENT, aut_id varchar(8),  
aut_name varchar(50), country varchar(25), home_city varchar(25) NOT NULL, PRIMARY  
KEY (id));
```

Primary Key Constraint:

EXAMPLE

The MySQL statement stated below will create a table 'newauthor' in which PRIMARY KEY set to the column aut_id.

```
CREATE TABLE newauthor(aut_id varchar(8) NOT NULL , aut_name varchar(50) NOT NULL,  
country varchar(25) NOT NULL, home_city varchar(25) NOT NULL, PRIMARY KEY (aut_id));
```

Primary Key on Multiple Columns

EXAMPLE

The MySQL statement stated below will create a table 'newauthor' in which PRIMARY KEY is set with the combination of aut_id and home_city columns.

```
CREATE TABLE newauthor(aut_id varchar(8) NOT NULL , aut_name varchar(50) NOT NULL,
country varchar(25) NOT NULL, home_city varchar(25) NOT NULL, PRIMARY KEY
(aut_id, home_city));
```

FOREIGN KEY CONSTRAINT:

Syntax:

```
FOREIGN KEY ([column list]) REFERENCES [primary key table] ([column list]); Arguments
```

EXAMPLE:

```
CREATE TABLE newbook_master (book_id varchar(15) NOT NULL PRIMARY KEY,
book_name varchar(50) , isbn_no varchar(15) NOT NULL, cate_id varchar(8), aut_id
varchar(8), pub_id varchar(8), dt_of_pub date , pub_lang varchar(15) , no_page decimal(5,0) ,
book_prices decimal(8,2), FOREIGN KEY (aut_id) REFERENCES newauthor(aut_id));
```

The MySQL statement does the below tasks

- ◇ A new table 'newbook_master' is created.
- ◇ The PRIMARY KEY for that table 'newbook_master' is 'book_id'.
- ◇ The FOREIGN KEY for the table 'newbook_master' is 'aut_id'.
- ◇ The 'aut_id' is the PRIMARY KEY for the table 'newauthor'.
- ◇ The FOREIGN KEY 'aut_id' for the table 'newbook_master' points to the PRIMARY KEY 'aut_id' of the table 'newauthor'.

Execute the above statement.

- Insert a value into the 'auth_id' column of the above table which is not present in the 'auth_id' column of the 'newauthor' table.
- Observe what happens and correct the error by inserting the appropriate value into the above table.