

Restaurant Inspection Data for Ottawa

MYSQL Tutorial #3

This tutorial will deal with creating, loading, and querying a Relational Database.

A Relational database is a database that has relationships between the different tables in it. This allows creation of webs of tables, for displaying complicated data in an easier-to-understand way.

There are two concepts that are very important for understanding relational Databases. These connects are Primary Keys and Foreign Keys.

When creating a table in SQL, you can designate a column as a Primary Key. A Primary Key is a unique column (no value can be repeated in the column) that is used as an identifier for the row. In a single table, you should be able to find a single row by looking for a single Primary Key value.

A Foreign Key is a column definition that means that the column must reference a Primary Key in a different table. Each value in the Foreign Key must also be present as a Primary Key value in a different table.

If you don't understand this yet, the tutorial should help you figure it out.

1. Firstly, on the open data site for Ottawa, find the [Public Health Inspections Dataset](#). From there, download the "restaurant" CSV. This is in a zipped format, so you'll have to unzip it to access the CSV files.
2. After unzipping, place the 5 files into a folder you can access, as you'll need the file-path later.
 - a. Delete the feed_info file. It is unnecessary for this tutorial, and it provides no useful data.
3. Create a new Schema for the Tutorial (call it tutorial 3, or Ottawa_Restaurants, etc.)
 - a. This can be done either through the wizard (right clicking on the schema section of workbench
 - b. It can also be done as a script, by simply typing "CREATE SCHEMA *schema_name*;" into the query tab.
4. Set your newly created Schema as the default schema
 - a. This can be done either through the workbench tools (right clicking on the schema you created, and choosing set as default)
 - b. It can also be done as a script by simple typing "USE *Schema_name*;" into the query tab.
5. Now you'll have to create the tables. You will need 1 table for each file, however there are far less columns in these tables than what we've worked with before with the parking-ticket data. For initial creation, do not concern yourself with Foreign Keys, but be sure to include Primary Keys
 - a. To ease creation, the datatypes for each table are as follows



Table Name:

Collation:

Comments:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
business_id	VARCHAR(40)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
name	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
address	VARCHAR(40)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
city	VARCHAR(15)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
state	VARCHAR(4)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
postal_code	VARCHAR(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
latitude	VARCHAR(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
longitude	VARCHAR(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
phone_number	VARCHAR(15)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Note the PK box is checked next to business_ID. This is crucial, otherwise the foreign key references that we will set up in the next step will break.



Table Name:

Collation:

Comments:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
business_id	VARCHAR(36)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
date	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
code	VARCHAR(5)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
description	VARCHAR(200)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL



Table Name:

Collation:

Comments:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
minimum	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
maximum	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
description	VARCHAR(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL



Table Name:

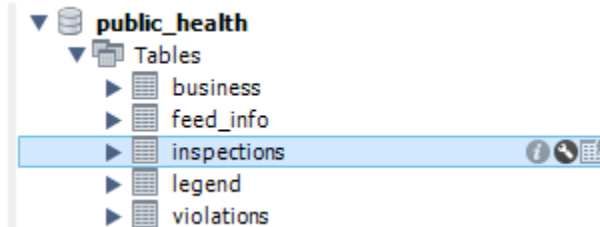
Collation:

Comments:

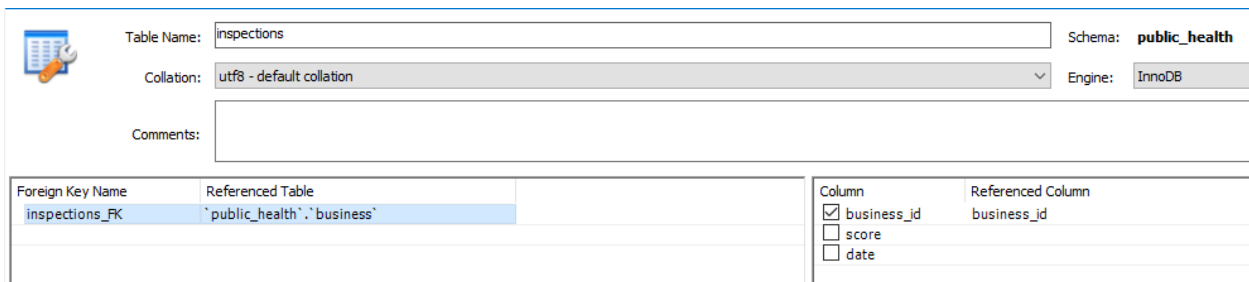
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
business_id	VARCHAR(40)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
score	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
date	VARCHAR(8)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

- We are now going to create a foreign key for violations and inspections. These tables will both reference the businesses table. This will allow us, once completed, to search all three tables by business_id, as it is a common column between each one.

- a. Refresh your navigator along the left hand side of the screen (right click, and go to refresh all)
- b. All of your tables should show up (if they haven't already)
- c. Mouse over inspections, and choose the small wrench icon



- d. This will bring up the table in a view where you can make changes. Along the bottom of this page, there will be a series of tabs. Choose the Foreign Keys tab
- e. Click in foreign key name, and write "inspections_FK"
- f. Click in referenced table – you will get a dropdown menu, choose the option that contains business.
- g. If business is selected, the business table should appear on the right hand side of the screen – check the box next to business_id to finalize the reference. It should be very similar to the picture below when completed.



- h. You should be able to complete the violations reference following the exact same steps.
7. Now that the tables have been created, we will make copies of them, for easier creation in the future.
- a. Open a new query tab to save all your creation scripts
 - b. Right click on each of the tables in your schema, highlight "send to editor", then click create statement. Do this for each of your tables. If you look at the scripts it produces, you will see that primary key and foreign keys are both explicitly stated in the table

creation. Your query tab should look like this.

```
1 CREATE TABLE `business_1` (  
2   `Business_id` varchar(40) NOT NULL,  
3   `Name` varchar(100) DEFAULT NULL,  
4   `Address` varchar(40) DEFAULT NULL,  
5   `City` varchar(15) DEFAULT NULL,  
6   `State` varchar(4) DEFAULT NULL,  
7   `Postal_Code` varchar(10) DEFAULT NULL,  
8   `Latitude` varchar(10) DEFAULT NULL,  
9   `Longitude` varchar(10) DEFAULT NULL,  
10  `Phone_Number` varchar(15) DEFAULT NULL,  
11  PRIMARY KEY (`Business_id`),  
12  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
13  
14 CREATE TABLE `inspections_1` (  
15   `business_id` varchar(36) DEFAULT NULL,  
16   `score` int(11) DEFAULT NULL,  
17   `date` varchar(11) DEFAULT NULL,  
18   KEY `inspections_fk_idx` (`business_id`),  
19   CONSTRAINT `inspections_fk` FOREIGN KEY (`business_id`) REFERENCES `business_1` (`Business_id`) ON DELETE NO ACTION ON UPDATE NO ACTION  
20  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
21  
22 CREATE TABLE `violations_1` (  
23   `Business_id` varchar(36) NOT NULL,  
24   `Date` datetime DEFAULT NULL,  
25   `Code` varchar(5) DEFAULT NULL,  
26   `Description` varchar(200) DEFAULT NULL,  
27   KEY `violations_fk_idx` (`Business_id`),  
28   CONSTRAINT `violations_fk` FOREIGN KEY (`Business_id`) REFERENCES `business_1` (`Business_id`) ON DELETE NO ACTION ON UPDATE NO ACTION  
29  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
30  
31 CREATE TABLE `legend` (  
32   `Minimum` int(11) DEFAULT NULL,  
33   `Maximum` int(11) DEFAULT NULL,  
34   `Legend` varchar(20) DEFAULT NULL,  
35  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
36  
37
```

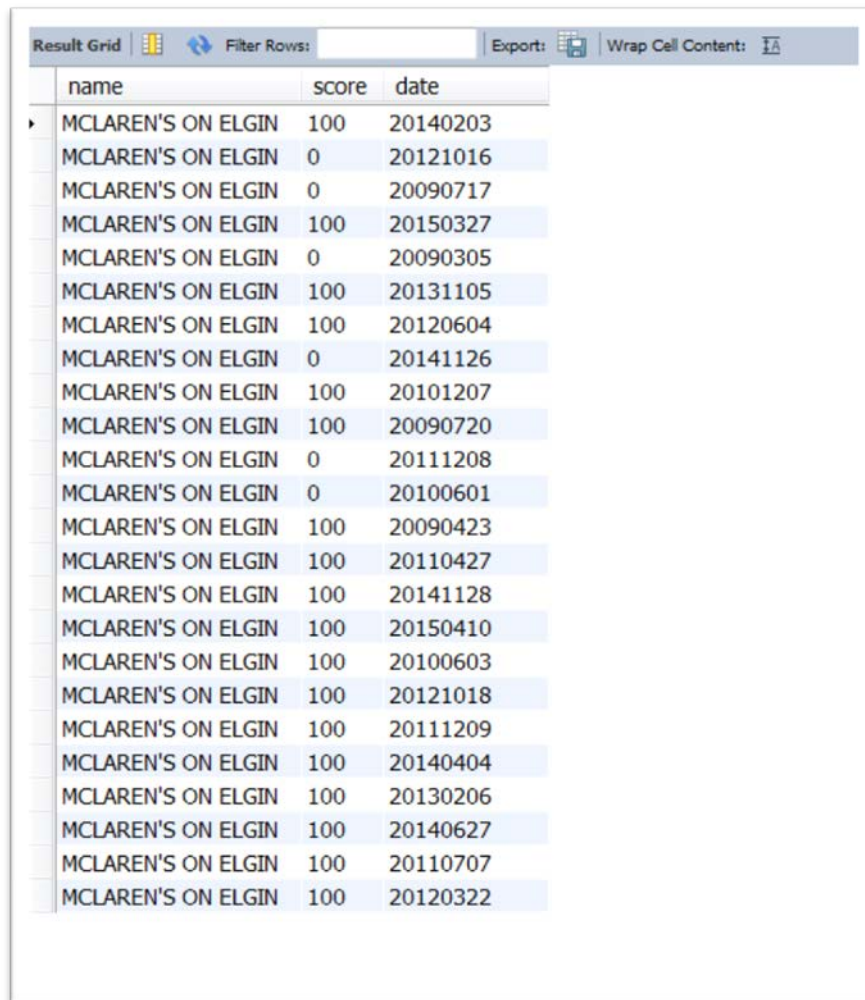
- c. Save this file
 - d. There are more detailed instructions for this step in the [previous tutorial](#).
8. Ok, we've created our tables, we've saved a copy of the scripts to create them, now let's load them with data.
 - a. The first step is creating the initial load statement. This is exactly the same as used last time with the first tutorial.
LOAD DATA LOCAL INFILE "*file-path*" INTO TABLE "*table name*"
FIELDS TERMINATED BY "," ENCLOSED BY """" LINES TERMINATED BY "\n"
IGNORE 1 LINES;
 - b. If you're having problems, ensure that the file path is correct, and that you are using forward slashes (/ - yes) instead of back slashes (\ - no).
 - c. You will get truncation warnings for the inspections table – however it is a problem with the initial dataset, and you can ignore them, as it is garbage data that is being deleted.
9. Ok, now that we have a working dataset, we can manipulate and play with it. Combining two tables such that you can see both of their data is known as Joining. There are a variety of different join types, however we will deal with the most common -- the standard or left join.
 - a. A standard join is done through several methods, however by far the easiest way is the "WHERE" command.
 - b. To begin, we'll have to pick a business to focus on. To do this, query the business table. ("SELECT * FROM *business* ;)
 - c. For example, let's use "MClaren's on Elgin".
 - d. Now let's see if we can find out when they were inspected
SELECT * FROM *inspections*;
When you perform this query, you'll notice that you are only given a *business_id*, which doesn't help us differentiate... We'll have to join the tables

- e. So, we'll have to perform a basic join to combine the two tables, and specify that we only want to see MClaren's Pub.

```
SELECT business.name, inspections.score, inspections.date FROM inspections NATURAL JOIN business WHERE business.name like "MCLAREN%";
```

So, now we can see that they were inspected 24 times. Let's break down the above statement, to make it easier to understand.

- i. As there are two tables referenced in the FROM section, we need to identify each column we need to put into the table. So, business.name specifies that we want the "name" column from the "business" table, and inspections.score means that we want the "score" column from the "inspections" table.
 - ii. NATURAL JOIN means that we want the tables to join together naturally. Which in this case, means that the two key values (business_id, which is a Primary key in Business and a foreign key in inspections) become a single column, where the two tables combine.
 - iii. The % sign at the end of MCLAREN is a wildcard character used in MYSQL. This means it can match to any characters, or none.
- f. So, using this SELECT statement we can see that MClaren's has been inspected 24 times.



The screenshot shows a database query result grid with the following columns: name, score, and date. The data is as follows:

name	score	date
MCLAREN'S ON ELGIN	100	20140203
MCLAREN'S ON ELGIN	0	20121016
MCLAREN'S ON ELGIN	0	20090717
MCLAREN'S ON ELGIN	100	20150327
MCLAREN'S ON ELGIN	0	20090305
MCLAREN'S ON ELGIN	100	20131105
MCLAREN'S ON ELGIN	100	20120604
MCLAREN'S ON ELGIN	0	20141126
MCLAREN'S ON ELGIN	100	20101207
MCLAREN'S ON ELGIN	100	20090720
MCLAREN'S ON ELGIN	0	20111208
MCLAREN'S ON ELGIN	0	20100601
MCLAREN'S ON ELGIN	100	20090423
MCLAREN'S ON ELGIN	100	20110427
MCLAREN'S ON ELGIN	100	20141128
MCLAREN'S ON ELGIN	100	20150410
MCLAREN'S ON ELGIN	100	20100603
MCLAREN'S ON ELGIN	100	20121018
MCLAREN'S ON ELGIN	100	20111209
MCLAREN'S ON ELGIN	100	20140404
MCLAREN'S ON ELGIN	100	20130206
MCLAREN'S ON ELGIN	100	20140627
MCLAREN'S ON ELGIN	100	20110707
MCLAREN'S ON ELGIN	100	20120322

In those 24, we see it has failed inspections 6 times. From the legend, you determine

that restaurants with a score of less than 100 fail the inspection. If you wanted to identify only those restaurants, you can add a second criterion to your WHERE clause to specify that you only want restaurants with failing grades, in this case less than 100 (use the legend as a guide). So your new WHERE clause would look like this: “WHERE business.name like 'MCLAREN%' AND inspections.score < 100”. The result looks like this:

name	score	date
MCLAREN'S ON ELGIN	0	20121016
MCLAREN'S ON ELGIN	0	20090717
MCLAREN'S ON ELGIN	0	20090305
MCLAREN'S ON ELGIN	0	20141126
MCLAREN'S ON ELGIN	0	20111208
MCLAREN'S ON ELGIN	0	20100601

10. Let's see why they failed their inspections

a. We'll now start a query using business and violations.

```
SELECT business.name, violations.date, violations.code, violations.description FROM
business NATURAL JOIN violations where business.name like "MCLAREN%";
```

business	date	code	description
MCLAREN'S ON ELGIN	2010-06-01 00:00:00	FWA	Floors, walls, and ceilings clean and in good repair
MCLAREN'S ON ELGIN	2010-06-01 00:00:00	FFA	Food is frozen at -18
MCLAREN'S ON ELGIN	2009-03-05 00:00:00	EQA	Equipment, non-food contact surfaces and linen are maintained, designed, constructed, installed and accessible for cleaning
MCLAREN'S ON ELGIN	2009-07-17 00:00:00	FCD	Food is held at 4
MCLAREN'S ON ELGIN	2014-11-26 00:00:00	STA	Sanitize test kit / thermometer readily available for verifying dishwashing and sanitizing temperatures
MCLAREN'S ON ELGIN	2012-10-16 00:00:00	MDC	Mechanical dishwashing: Wash / rinse water clean, water temperature, timing cycles, sanitizer
MCLAREN'S ON ELGIN	2011-12-08 00:00:00	MDC	Mechanical dishwashing: Wash / rinse water clean, water temperature, timing cycles, sanitizer

b. You can see the violations that occurred for each inspection, as well as the code that was violated.

c. To get a bit more practice, use the WHERE clause to create other tables, which you can also save as a csv file (we learned this in step [43 of the previous tutorial](#)), and then export to the folder you're using for this tutorial.