Criminal Case Reversals in the State of Massachusetts

Database Design: Techniques and Summary
Navraj Narula
May 1, 2016

My final project for my directed study alongside Maggie Mulvihill involved building an entire model for a database from start to finish. The process was completed in MySQL Workbench, a free tool made available by Oracle that allows database architects and developers to create the structure of a DBMS (database management system).

The one that I am choosing to design for our project is relational, meaning that it will contain a set of tables holding data fitted into predefined categories. A table itself may pertain either only to one entity or serve as a link between two or more.

For instance, I may have one table dedicated to a person. This will hold all attributes pertaining to the person such as their name, age, gender, and date of birth. I may then have another table pertaining to a case. This will hold information such as the case number, case title, or date it was filed. In order to link a person to a case, I will create a separate table. The values in this table will be the primary key attributes from the person and case table. In this new separate table, they are referred to as foreign keys. I will follow this pattern of design for all tables in our database to maintain consistency, building in primary keys and foreign keys wherever they may be necessary.

A script will be generated after building the database. This will allow any person who has the script to simply run it in the Workbench tool and recreate the full database. All tables and constraints will be in place, including row entries if any values have been inputted. I will go through the process on how you can set this up on your own computer towards the end of the report.

Discussion of Tables

Person table
PersonID — serves as a unique identifier for the person
LastName — a person's last name
FirstName — a person's first name
Position — role the person plays in case proceedings

Person			
PersonID	*LastName*	*FirstName*	*Position*
1023	Jones	Martin	Defendant
1024	Walker	Lauren	Defense Attorney

Case table
CaseID — serves as the unique identifier of the case (i.e. docket number)
CaseName — the name of the case
DateDecided — the date that the case was decided on
DateArgued — the date that the case was argued on
DateFiled — the date that the case was filed on

Case				
CaseID	*CaseName*	*DateDecided*	*DateArgued*	*DateFiled*
SJC-11824	Commonwealth v. Herbert Dorazio	03/14/15	02/13/14	02/11/14
SJC-11672	Commonwealth v. Alex Scesny	11/16/15	04/17/15	01/01/15

Status table
CaseID — serves as the unique identifier of the case (i.e. docket number)
Cause — cause of conviction
RevReason — reason why the case was reversed
Disposition — whether or not the convictions have been reversed completely, in part, or not at all
Status — whether or not the case is closed

Status				
CaseID	*Cause*	*RevReason*	*Disposition*	*Status*
SJC-11824	Assault and Battery	Judicial Error	Convictions Reversed	Closed
SJC-11672	Child Abuse	Insufficient Evidence	Convictions Reversed, in part	Closed

Proceedings table
CaseID — serves as the unique identifier of the case (i.e. docket number)
TrialCourt — location of where the trial was held
AppCourt — appellate court, where an appeal of a trial court is heard
County — county where the case took place

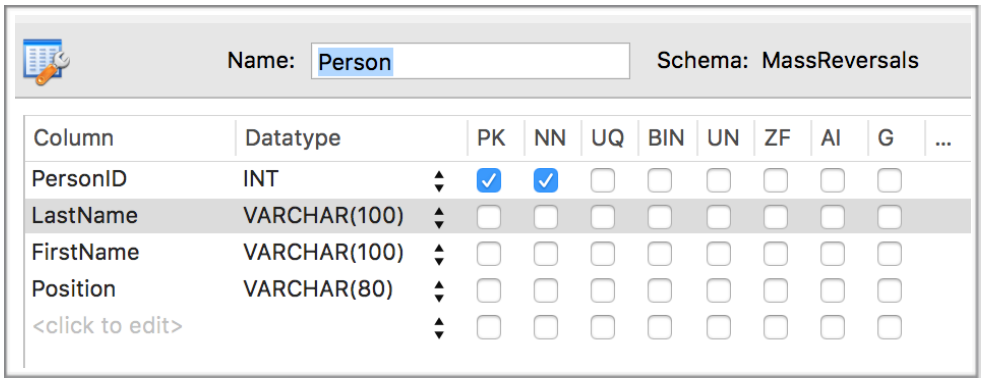
Proceedings			
CaseID	*TrialCourt*	*AppCourt*	*County*
SJC-11824	Springfield District Court	MA Appeals Court	Hampden
SJC-11672	Westfield District Court	MA Appeals Court	Hampden

CasePerson table
PersonID — serves as a unique identifier for the person
CaseID — serves as the unique identifier of the case (i.e. docket number)

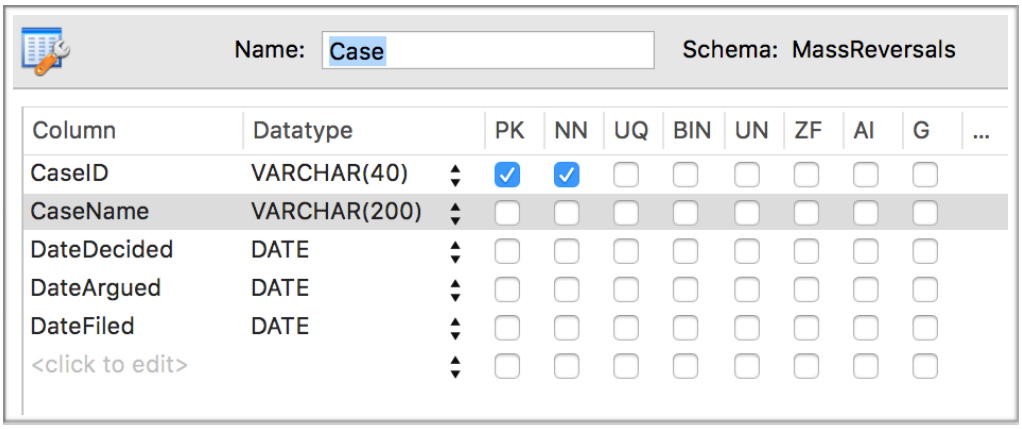
CasePerson	
PersonID	*CaseID*
1023	SJC-11824
1024	SJC-11672

Here are what the table definitions look like in MySQL Workbench itself:

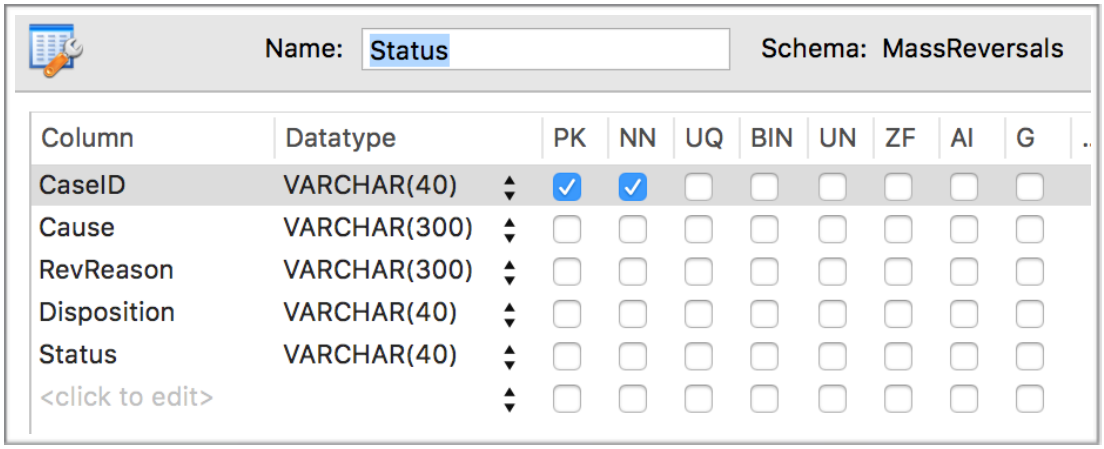
Person table



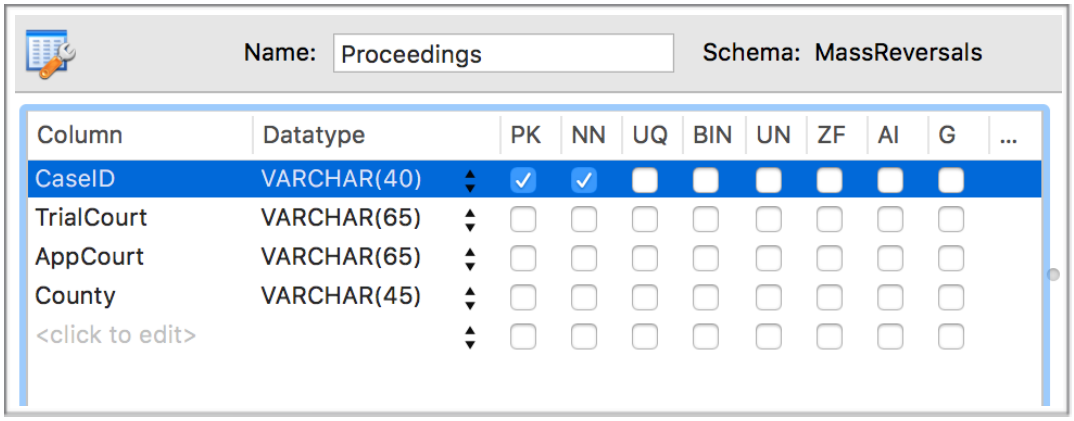
Case table



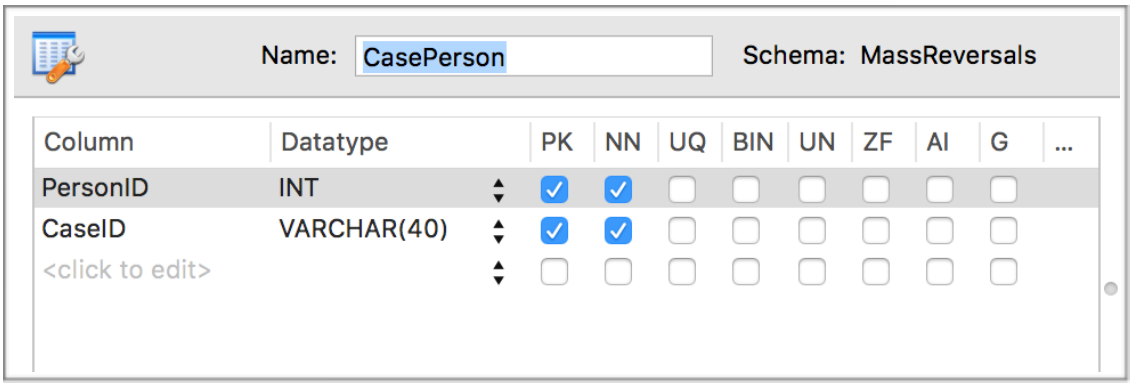
Status table

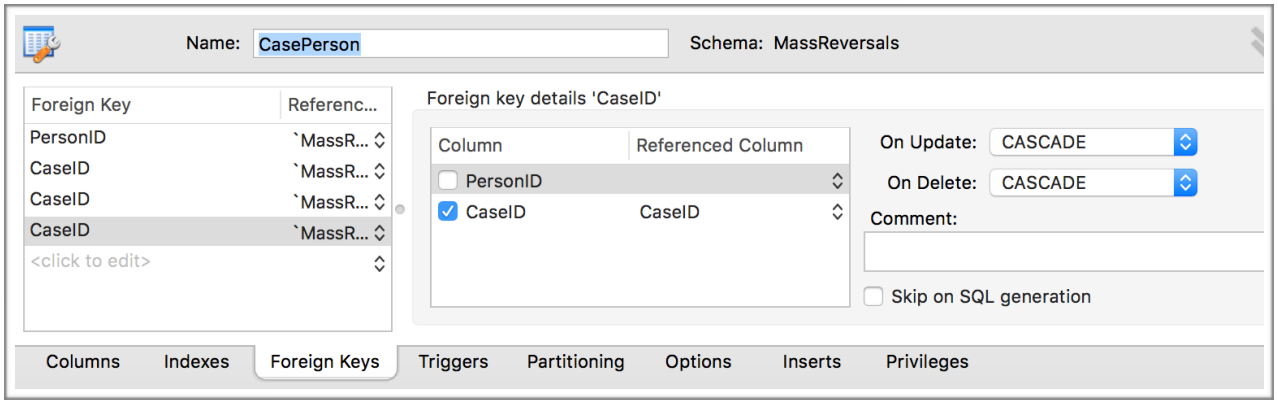


Proceedings table

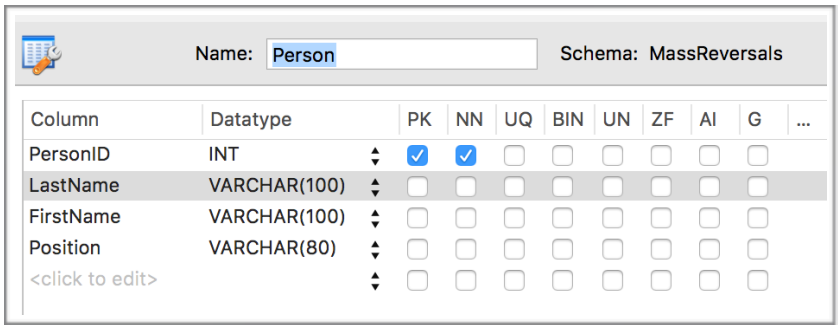


CasePerson table



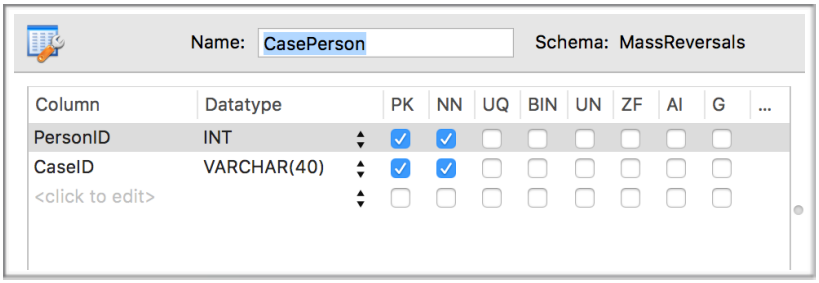


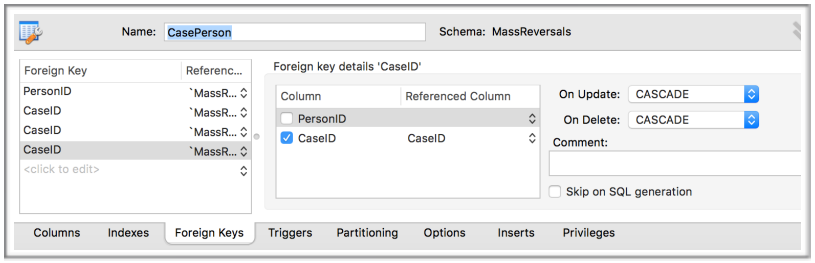
Let's zoom in on a particular table in order to understand how attributes need to be defined within MySQL Workbench. I am going to focus on the Person table.



The PersonID attribute is specified as an integer. This means when a user inserts a new value into the table, they must specify the person's ID as an integer. Otherwise, it will be rejected by the DBMS. Notice that the PersonID is also the primary key, as indicated by PK. This means that the DBMS will reject a duplicate value if the user chooses to enter it into the table. This value may also never be null, as specified by NN. The rest of the attributes in this table can be inputted as strings. VARCHAR(n) allows the user to input a string that may up to as long as n. For instance, a person's last name—as it is specified above—may not be longer than 100 characters. You may not specify a limitless string in MySQL Workbench so having an idea of what your data looks like helps in specifying the number. I have definitely seen some longer names, so I wanted to make sure to accommodate those here.

All my other tables follow the same general rule as the Person table. I would, however, like to highlight the CasePerson table since it serves as a linking table within our schema.





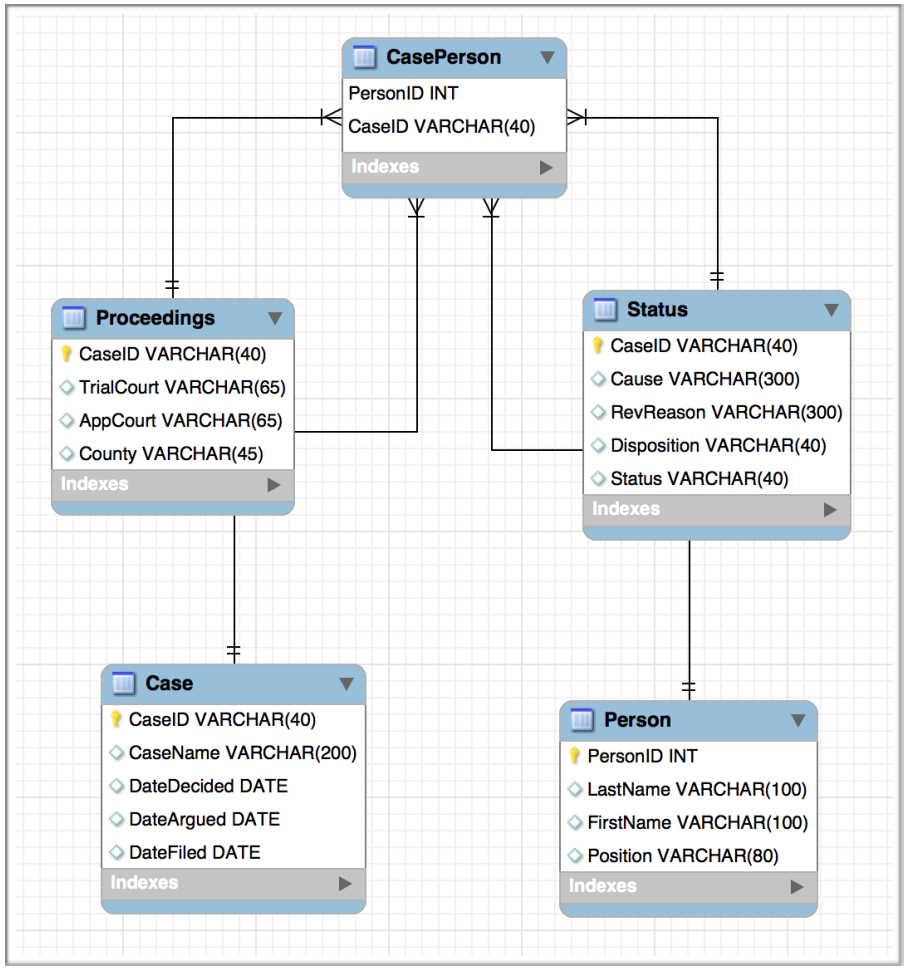
The CasePerson table contains two primary keys: the PersonID and the CaseID. Below the table definition, you will notice that I have referenced these keys to their parent table. Though they are primary keys for this particular relation, they act as foreign keys since the values specified under these attributes "point to" their parent. To summarize:

- PersonID in the CasePerson table refers to PersonID in the Person table
- CaseID in the CasePerson table refers to CaseID in the Case table
- CaseID in the CasePerson table refers to CaseID in the Proceedings table
- CaseID in the CasePerson table refers to CaseID in the Status table

Notice how the CaseID in the CasePerson table actually holds a reference to three separate tables. While this is the way it is defined in the schema, the script that I generated will not support this feature. Therefore, I have chosen to minimize the pointers. Now, the CaseID in the CasePerson table will only point to the CaseID in the Case table. This is completely fine and even makes for a more efficient database since references are minimized. Via SQL commands, all tables are considered connected nonetheless.

Entity-Relational Diagram

Below is a diagram that I created that shows how all of the tables are connected:



SQL Script

Unfortunately, I am unable to convert the .sql script into a .pdf document. I have instead included screenshots of the entire script here:

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema MassReversals
-- -----------------------------------------------------


-- -----------------------------------------------------
-- Schema MassReversals
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `MassReversals` DEFAULT CHARACTER SET utf8 ;
USE `MassReversals` ;

-- -----------------------------------------------------
-- Table `MassReversals`.`Person`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MassReversals`.`Person` (
  `PersonID` INT NOT NULL,
  `LastName` VARCHAR(100) NULL,
  `FirstName` VARCHAR(100) NULL,
  `Position` VARCHAR(80) NULL,
  PRIMARY KEY (`PersonID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `MassReversals`.`Case`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MassReversals`.`Case` (
  `CaseID` VARCHAR(40) NOT NULL,
  `CaseName` VARCHAR(200) NULL,
  `DateDecided` DATE NULL,
  `DateArgued` DATE NULL,
  `DateFiled` DATE NULL,
  PRIMARY KEY (`CaseID`))
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `MassReversals`.`Status`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MassReversals`.`Status` (
  `CaseID` VARCHAR(40) NOT NULL,
  `Cause` VARCHAR(300) NULL,
  `RevReason` VARCHAR(300) NULL,
  `Disposition` VARCHAR(40) NULL,
  `Status` VARCHAR(40) NULL,
  PRIMARY KEY (`CaseID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `MassReversals`.`Proceedings`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MassReversals`.`Proceedings` (
  `CaseID` VARCHAR(40) NOT NULL,
  `TrialCourt` VARCHAR(65) NULL,
  `AppCourt` VARCHAR(65) NULL,
  `County` VARCHAR(45) NULL,
  PRIMARY KEY (`CaseID`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `MassReversals`.`CasePerson`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MassReversals`.`CasePerson` (
  `PersonID` INT NOT NULL,
  `CaseID` VARCHAR(40) NOT NULL,
  PRIMARY KEY (`PersonID`, `CaseID`),
  INDEX `CaseID_idx` (`CaseID` ASC),
  CONSTRAINT `PersonID`
    FOREIGN KEY (`PersonID`)
    REFERENCES `MassReversals`.`Person` (`PersonID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `CaseID`
    FOREIGN KEY (`CaseID`)
    REFERENCES `MassReversals`.`Case` (`CaseID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

I would like to draw attention to this particular part of the script. You'll notice hear that I am defining the CasePerson table (i.e. the linking table). If an update happens (i.e. a person's last name is changed in the parent table) or a delete happens (an entire row is deleted from the parent table), the result will reflect itself in the linking table as well. This is made possible by specifying the "Cascade" keyword.

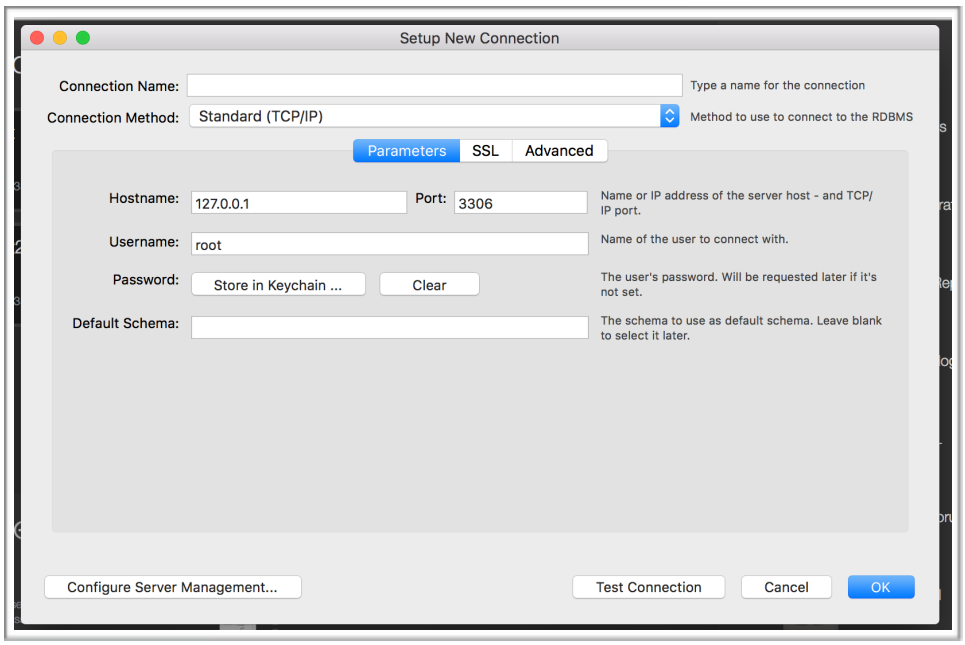
You may need to download Sublime to view the script. You may do so here: https://www.sublimetext.com/download

Any text editor should be fine, though!

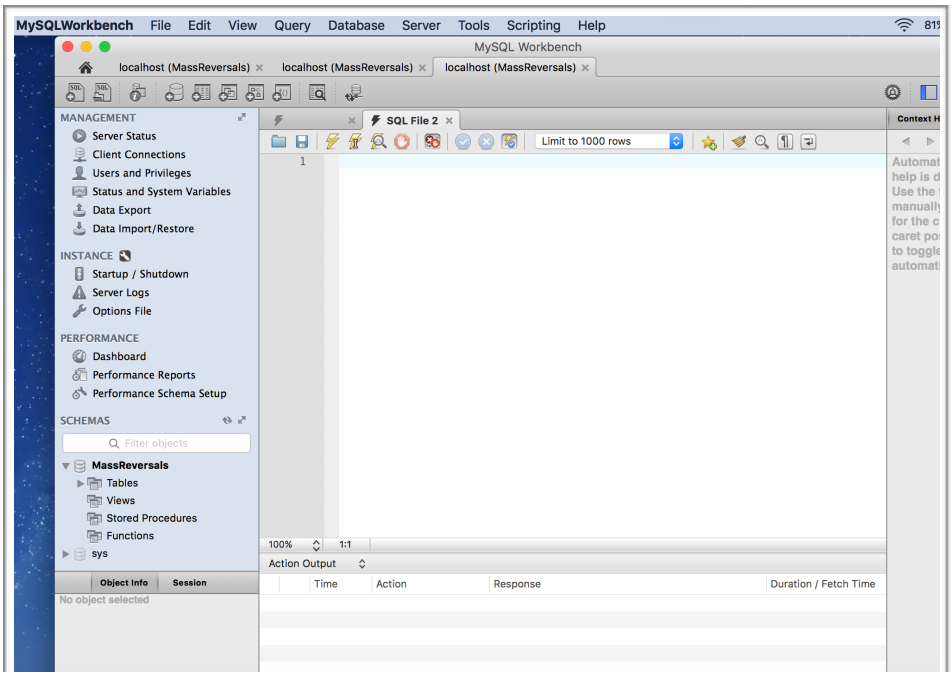
Loading the Script and Testing the Database

Before loading the script, you must make sure to have the latest version of MySQL Workbench installed. You may find the appropriate version for your operating system by following this link: https://dev.mysql.com/downloads/workbench/

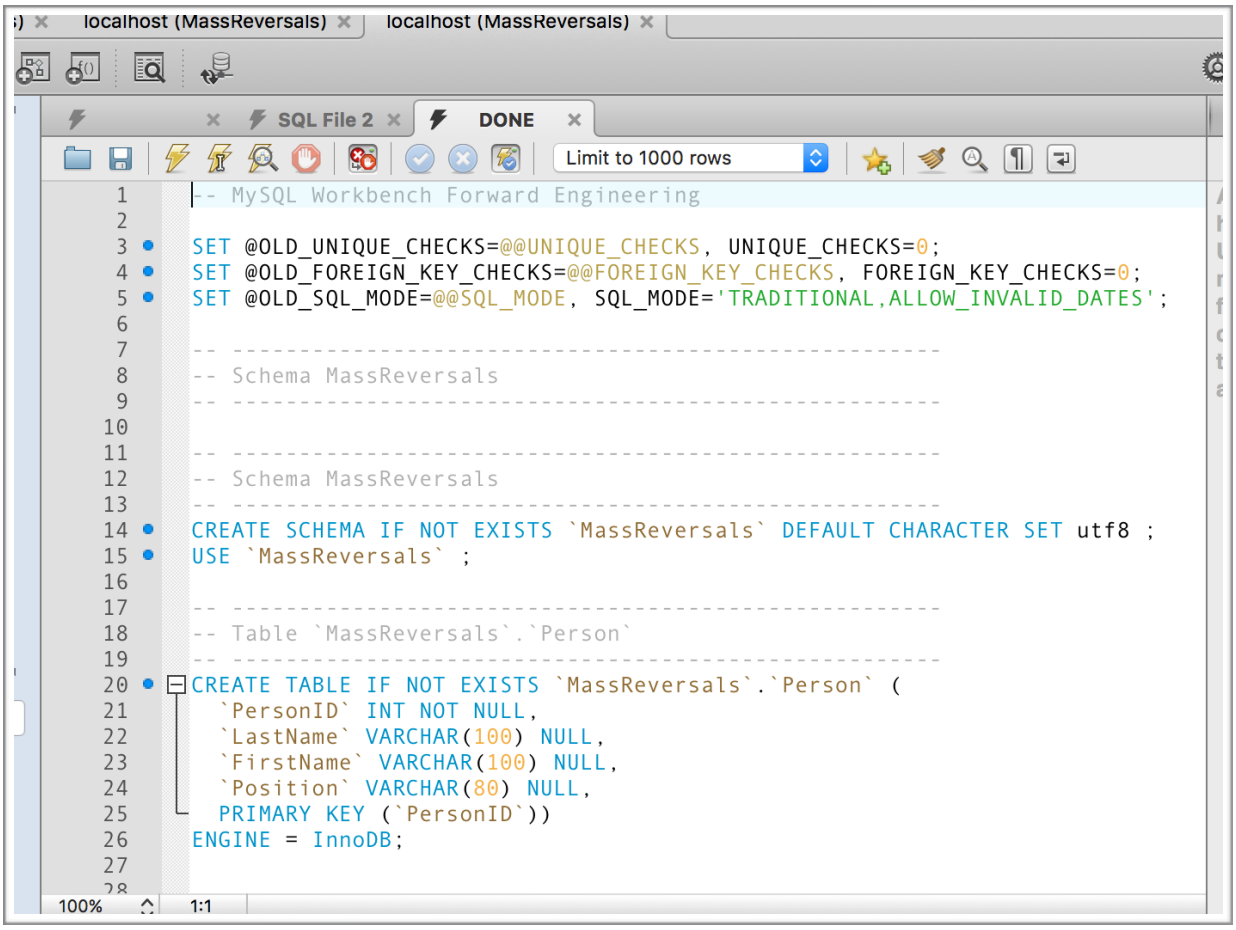
Once you have installed the software tool, you'll need to open up a connection. You'll see this dialog box with the default name set to root. I would suggest naming the connection "localhost," simply because it's standard practice.



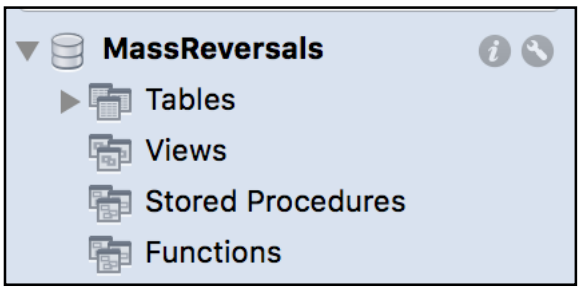
After that, you will be in the Workbench tool itself:



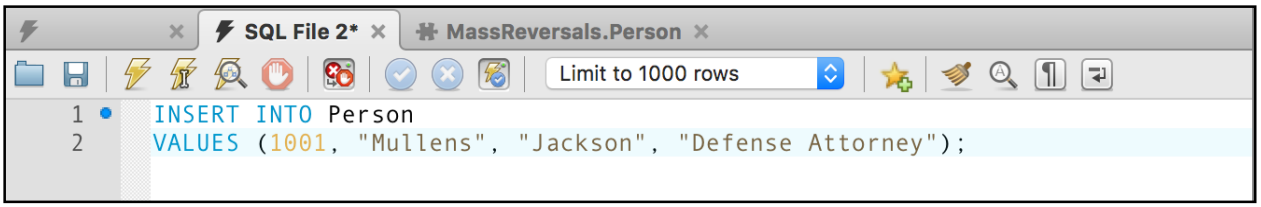
To load the script I have passed along, click File —> Open SQL Script… and choose the file. Your screen should now look like this. Click the lightning bolt to run the script.



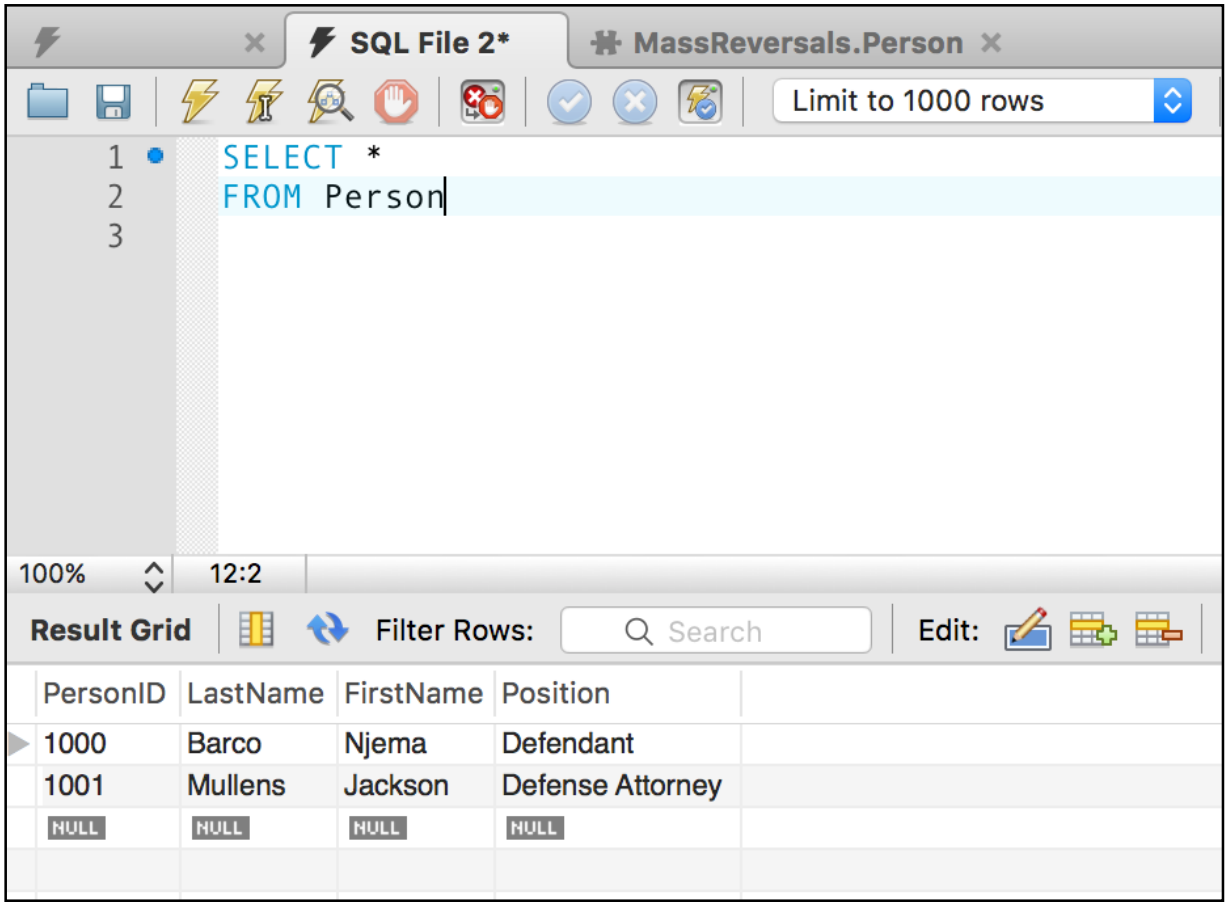
After you run the script, you'll notice that the entire database has been loaded in. I called our database **MassReversals**. If you do not see this box on the left hand side, try restarting the Workbench tool again.



I will now test to see if the database truly works by entering rows in the Person table.



To confirm whether or not my SQL command went through, I chose to execute another to view the updated table:



I notice that my table has been successfully updated! This confirms that the database works and is responsive to SQL commands. Since this is the case, opening up a further connection to this database within any programming language (namely Java or Python) would be extremely easy. Having this access also paves the way for a front-end tool that could potentially go live once we collect enough data to make this database available to anyone that we may so choose to.