```
-- Goal: Create tables, INSERT Data, talk about NULL and constraints.

-- INSERT is often referred to as the "Create" portion of
--  ("CREATE","READ", "UPDATE", "DELETE") = CRUD processes
-- NOTE: semi-colons are a delimiter that separates statements
-- when multiple statements are wanted.
-- I recommend that you always use them when writing MySQL code.

use <dbnamehere>;


-- let's create a table...
Create Table Student
      (ID int auto_increment,
      FirstName varchar(30) NOT NULL,
      LastName varchar(30) NOT NULL,
      BirthDate date,
      Primary key (ID));

-- Discuss: autoincrement



-- There are different ways to get data into a table
-- (Do you see that this instruction has no data for the BirthDate field?
--      Let's run it anyway…)
Insert Into Student (FirstName, LastName)
      Values('Ross','Geller');

-- What happened to the BirthDate data??
-- What do you think is in that record and column?




-- Let's see what the data looks like:
SELECT * FROM Student;

-- DISCUSS:  NULL vs  NOT NULL
```

```
-- So… some columns can be skipped.
-- However, NOT NULL columns MUST be given data, NOT NULL is a constraint
-- that requires data in the records for those columns that have the constraint
-- check out the error here when we ignore the NOT NULL column (FirstName)
Insert Into Student (LastName)
      Values('Geller');




-- Other constraints are also acting to protect
--  the integrity of the data in the table...
-- What other constraints are defined on the table above?




-- the Primary Key is a constraint that says the column can only contain unique
content..
-- What column from the table above is our Primary key column?




-- so what might happen if I try this?
```

```
Insert Into Student (ID,FirstName,LastName)
      Values(1, 'Ross','Geller');









-- Back to creating table data..
-- let's see some other types of INSERT instructions



-- We've added one using VALUE, but we can also use many VALUES
-- many values looks like this:
Insert Into Student (FirstName, LastName)
      Values('Hope','Floats')
      ,('Perry','Mason')
      ,('Hermann','Munster')
      ,('Lily','Tomlin');

-- VALUES are useful when you just want to type or use copy/paste
--    to put text directly into the records of data

-- This is the main purpose of the VALUES form of inserts.

-- You won't see VALUES as insert instructions when
--     you're writing applications and moving data between
--     a user of the application and the database.









-- We can use a "result set" to put data into a table...
```

```
-- So... now you want to know, “what is a result set”?




-- a result set is the results of any SELECT statement
-- SELECTed values can come from a table,
--  or from variables or parameters,
--  or just ad-hoc data
--  or any combination of these.


-- (Technically the VALUES example above don’t represent a “result set”
-- VALUES doesn’t generate a result set the way a select statement does.)




-- First, I’m going create an ad hoc set of data that I want for the table
Select 'Phoebe','','1975-08-01';

-- (note: this also works when you have variables or parameters with data..
-- We’ll see some examples of variables and parameters in later classes)


-- Here’s how it applies to an INSERT
Insert Into Student (FirstName, LastName, BirthDate)
Select 'Phoebe','','1975-08-01';




-- Now Let’s SELECT the data and look at it:
SELECT * FROM Student;

-- What’s the difference between “NULL” and empty string
-- (see the Phoebe example in the result set… LastName is an empty
string..)
```

```
-- empty strings are “something” but NULL is the absence of anything



-- Discussion if time allows:
-- We don’t have a DEFAULT or UNIQUE constraint on this table
-- How might these other constraints apply to this table?
--         What could we do with them,
--                   and would it makes sense to do that?
--         Can either of those constraints help the data be better
--                   (i.e. have greater “Data Integrity”)?
```