Lecture 7 Notes - Monday 10/05/16

Reading Quiz

Question 1: Ans = D
Question 2: Ans = C
Question 3: Ans = E
Question 4: Ans = A

Notes

NOTE: we're going to be using Python 2.7

There are lots of MySQL connectors for Python -- we're going to be using PyMySQL
Make sure to get comfortable referencing the PyMySQL documentation, as PyMySQL isn't covered in any of our textbooks

Connection Test *connect* lets us establish the connection to the database, whereas the *cursor* is what we use to actually execute SQL commands

dual lets you run queries without actually having a table in the database

Note we are using a try/except block here, so if something goes wrong, we can see what the error was

(*finally:* means, no matter what happened (any errors, etc.) go ahead and close the connection at the end)

Concept Question 1: B.
A this doesn't really match the error -- in the last slide we saw what happens when you have a bad IP address, and this isn't it
B this is right -- you can kind of tell by how it says "Access denied for user…"
C if this were the case, it would've said something like "bad database"
D no this query is fine actually

Single Insert here we can see a basic example of using a Python function to create a connection and a cursor and use these to insert data into our database
We pass the query as a string to *execute* -- note we pretend the date is a string when we send it through the cursor, even though it really has a date type
Also, note that when we are using the connector, we have to manually commit our command using *conn.commit()*
It's best practice to close your connection when you finish, so it's not hanging around on your computer doing nothing

Note that a given cursor can only have one command active at any given time; so this is why we have cursors instead of just the connection -- if we're working on multiple commands/queries at once, we would need multiple cursors

Also, a given connection is tied to a particular database, so you'd need multiple connections to connect to multiple databases

One example of when you might need multiple cursors is if you are iterating through two parent tables to populate a junction table ; you might need a cursor for each parent table, and maybe one for the junction table too!

Concept Question 2: B.
Since we used the shortcut method for inserting values by not specifying column names, we needed to put a value for every column -- since we didn't do this, we got an error
The error message here is pretty straightforward, but you might want to keep this one in mind so if this happens to you, you'll know right away what went wrong

Multiple Inserts
If i == 0: continue ignores the headers, so we don't store them as a row in the table
Note that here we're making a new connection and cursor for each insert statement, but this is suboptimal -- in reality, we'd probably establish one connection, execute all the insert statements, and do one commit at the end

What can go wrong
We could get this PK error for a couple reasons -- maybe we ran the program twice without clearing out the table/database first; or maybe our raw data just contains duplicates, and nobody has cleaned it out. You don't necessarily need to clean up your raw data to make a "flawless insert" -> that's the whole point of having the PK constraint is MySQL doesn't let up enter duplicate rows, so it could be okay to just ignore these errors because we don't want to insert the duplicate rows anyways

Concept Question 3: B.
This method could result in "hidden" duplicates in the sense that, say we have two rows in the raw data that are exactly the same -- same eid, same name, etc. If we use a surrogate key, we're going to simply assign different PKs to each of these rows (via auto increment) and both will make it into the table. So even though these two rows will have different primary keys, they will be duplicates because they contain the exact same student info
The point is that we should try to avoid assigning surrogate keys when there are natural keys