**Lecture 13 Notes - Monday 11/21/16**

**Reading Quiz**

Question 1: Ans = C
Question 2: Ans = A
Question 3: Ans = B
Question 4: Ans = D
Question 5: Ans = A

**Notes**

You can go here to experiment with your JSON paths for your queries: http://jsonpath.com.

*DDL* stands for *data definition language*; it includes create table statements and alter table statements -- the kind of stuff you're going to need to add on to your database to allow for the new JSON tables

When we include the term *stored* in these new JSON-related columns, we're doing some extra pre-processing because MySQL has to both generate and store these values in the tables, but it will be quicker once we're doing queries on the table, because MySQL will already have the values available without having to regenerate them from the JSON each time

In our Twitter API Client: Note that we are only inserting the *tweet_doc* and *major_code* values in the insert statement, but if our DDL works right, MySQL will take care of the rest of our columns for us using the json extractions

**Concept Question 1: A.** If we want to run our Twitter API Client more than once, we're going to end up getting some duplicate insert statements, because we're going to get the same tweets over again, plus some new ones. This is why we want to check whether a Tweet ID exists in our table already before trying to insert it. To see if we have any tweets with the given *$id* in our table, we want to get a count of the number of tweets in our table with that *tweet_id* value. *A* accomplishes this. (Using a *count(*)* is the best option because it always returns a number -- further, it will be either 0 or 1 because *tweet_id* is a primary key.) *B* is bad because it's inefficient -- we're getting ALL our data, and it doesn't give an immediate way to know whether to insert a record; it doesn't tell us whether the relevant *$id* is in our table already. *C* gives equivalent results as *A*, but the *distinct* is redundant because *tweet_id* is the primary key and thus guaranteed to be distinct. *D* is bad because it is inefficient, like B, though a little better because it at least shows us the record if it exists.

**Concept Question 2: B.** This problem doesn't really have to do with joins, so *C* and *D* don't really make sense here. *A* doesn't work because taking *count(t.origin_tweet_id)* is going to still give us all the retweets, because all the retweets have values in this column and will thus be counted. *B* gives us all

of the distinct values of *t.origin_tweet_id*, which is the number of original tweets, and does not include retweets.