Lecture 10 Notes - Wednesday 10/26/16

Reading Quiz

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

Notes

***COUNT*:** count(*) includes nulls
count(*column_name*) does not include nulls
count(*distinct column_name*) doesn't include nulls

Concept Question 1: . *A* is just going to give us the number of non-null entries in the *Department* column, 6, which isn't what we want. *B* doesn't really make sense here because you can't sum up VARCHARs / the names of departments. *C* gives you the total number of rows, regardless of *Department*, 8, which isn't what we want. *D* gives us the number of different departments, 2, which is what we want. Note that *D* doesn't count the null entries in *Department*.

***GROUP BY* Examples:**
When we do a **group-by** on a **join**, we need to be mindful of the "empty group" problem. The empty group problem happens when we lose a group from the result set due to the results of the inner join. It's important to remember that the join is performed prior to the group-by. In the example, we lose the "Dallas" group because no Dallas-based customers placed orders. In order to preserve the empty groups (e.g. Dallas), we need to use a **left outer join**. Also, note that the **having** clause is a filter on the entire group (as opposed to the **where** clause which filters individual records).

Concept Question 2: A. *A* is good because it includes the "null departments" (since it has *count(*)*). *B* is not good because it doesn't group the results -- it combines an aggregate with a non-aggregate, *Department* with *COUNT(*)* which doesn't make sense. (It will either give you a syntax error or some nasty random results.) *C* has the issue of mixing aggregates and non-aggregates, plus it doesn't catch the nulls. *D* doesn't grab the null departments.

Concept Question 3: B. We're grouping by *city* in this query but in the **select** clause we want *city*, *order_date* and *sum(total_amount)*. But what are our groups going to look like? We're aggregating across multiple *order_dates* but we're **selecting** *order_date* -- MySQL has no idea what to show you here. Exactly which *order_date* should it show? Rule: Every non-aggregated field in the **select** clause needs to show up in the **group by** clause. Since we aren't doing this here, *order_date* causes trouble.

Note we could have done ***group by*** *c.city, o.order_date*. That would work, because each unique combination of *city* and *order_date* would have its own group.

Concept Question 4: A. Since we want to check the completion of an entire test, over all its steps, we want to group by *test_name*, not *test_step*, so this takes *C* and *D* out of the running. For *B*, note that the **where** clause is applied first, to filter the data (this is just kind of a fact about the order of evaluation for a query). But this means that *B* is going to get rid of the null completion dates before we even do any grouping. So *B* will just give us a table with the names of both tests. *A* basically checks that for each *test_name* the amount of *test_steps* is the same as the number of non-null *completion_dates*. This tells us whether the test is complete, which is what we want, so *A* makes us happy.