Homework 6: Reflection

Generally, my only true design approaches were 1) use natural joins whenever possible and necessary, and 2) avoid subquerying whenever possible. Natural joins, in my mind, are quite elegant and ensure that things line up where you think they are. I've avoided subqueries whenever I can because they are both slow and confusing; I frankly don't trust myself to get them right every time. Other than that, things were generally normal SQL. I've tried to make my code elegant whenever possible; however, the code is inelegant when it comes to subquerying, as was necessary in some cases.

In the queries involving subqueries, I initially assumed the most efficient way would be a single subquery. This being said, I actually found a twofold subquery to be more elegant when using the aggregate MAX function. Additionally, for #9, I assumed that I could group the table by either employee name, or by film title. I quickly realized that this was impossible - grouping by one dropped necessary information/rows from the other. I decided, then, to group by both, so as to preserve information more thoroughly, and make writing my query more elegant. For #10, the last query, I initially tried to make a query that would return movie titles that were rented outside of the standard deviation of all titles rented (films that did exceptionally well/poorly). I found this to get convoluted very quickly, and instead, I opted for a different query. Overall, I found out how convoluted subqueries, in general, can get.

My testing strategy was generally this:

1. Create an initial query (subquery if appropriate, really whatever the innermost query is)
2. Run the query, and analyze results.
3. Pick a few cases - eg COUNT/MAX, which I'd like reserved
4. Expand the query to select the max values, or what I actually want
5. Compare the max values to those preserved from previous steps
6. Expand query further (go to 3), or stop if I find that my results are accurate.

My query generation strategy, generally, is:

1. Run a few small queries (select * from …), to get a feel of the table
2. Run a few aggregate queries (Select max, select count, etc), to get a sense for the value range I'll be working with in the table (develop a sense for what is reasonable)
3. Draw the tables out on a whiteboard with pointers between, wherever I'd be performing joins and the like
4. Write the general select from where query
5. Begin testing (above)