WEEK 3 TERADATA EXERCISES GUIDE

The Teradata exercises for this week assume that you have completed all of the MySQL exercises, and know how to use GROUP BY, HAVING, and JOIN clauses. The quiz for this week will build off of the exercises below, so although I will give you hints to the answers, most of the time I will not give you the answers explicitly. If you can complete all the exercises listed below, you will do very well on the quiz. You might find it useful to save the queries you write during the exercises either in your Teradata account or in a text file on your computer so that you can use them for reference during the quiz.

Please note that queries that join tables take much longer than those that don't, so many of the exercises you complete this week (including those required to answer quiz questions) may take much longer to complete than those you completed last week. You will want to plan accordingly when you are ready to take the quiz.

Specific Goals for Week 3's Teradata exercises

Use these exercises to:

- Understand the different subsets of sku numbers and departments in each of the tables in the Dillard's database (it is important to understand this in order to draw conclusions about Dillard's sales patterns)
- Learn the differences between Teradata's and MySQL's GROUP BY syntax
- Practice joining tables using a relational schema as a guide

GROUP BY clauses in Teradata vs. MySQL

The syntax for standard aggregate functions, HAVING clauses, and joins is the same in Teradata as it is in MySQL. However, GROUP BY clauses in Teradata differ from GROUP BY clauses in MySQL. In MySQL Exercises 5 and 6, we discussed how the outputs of a query must all be aggregated at the same level. If you include a GROUP BY clause in your query, but forget to aggregate a column included in your SELECT list, MySQL will output a random row from the un-aggregated column when the query is executed. This functionality is convenient when you know that all the values within a group of an un-aggregated column are the same, but also makes it easy to misinterpret the results of poorly written queries. To prevent users from having the opportunity to write poor queries in the first place, Teradata has a different rule for GROUP BY statements than does MySQL. In Teradata (and many other database systems):

Any non-aggregate column in the SELECT list or HAVING list of a query with a GROUP BY clause must also listed in the GROUP BY clause.

This rule makes it less convenient to write some queries, but ensures that outputs that aggregate across groups of records aren't mixed with outputs that only represent single records.

To understand the aggregation rule better, consider this query:

```
SELECT sku, COUNT(sku), retail, cost
FROM skstinfo
GROUP BY sku
```

This query would be executed in MySQL (if the Dillard's dataset was a MySQL database), but a random value in the retail column and the cost column would be outputted for each output row that reports a count of a given sku number. Teradata, on the other hand, will return this error if you tried to execute the query above:

> Error Code - 3504
> Error Message - [Teradata Database] [TeraJDBC 15.10.00.05] [Error 3504] [SQLState HY000] Selected non-aggregate values must be part of the associated group.

To resolve the error, either retail and cost both need to be included in the GROUP BY clause (but note that this query would output a separate row for each distinct combination of sku, retail, and cost rather than a row for each sku, on its own):

```
SELECT sku, retail, cost, COUNT(sku)
FROM skstinfo
GROUP BY sku, retail, cost
```

…or retail and cost need to be aggregated to match the sku aggegration:

```
SELECT sku, COUNT(sku), AVG(retail), AVG(cost)
FROM skstinfo
GROUP BY sku
```

The "Selected non-aggregate values must be part of the associated group" error is one of the most common errors new SQL learners make, so keep an eye out for it. If you see it, immediately check the columns you have included in your SELECT and GROUP BY clauses.

Integrate everything you learned this week

Keep your Dillard's relational schema handy while you work through these exercises. You will likely need to look up column names, and will want to refer to the diagram to determine which columns to use to join your tables.

Exercise 1: (a) Use COUNT and DISTINCT to determine how many distinct skus there are in the skuinfo, skstinfo, and trnsact tables. Which skus are common to all tables, or unique to specific tables?

You should see that:
distinct skus in skuinfo > # distinct skus in skstinfo > # distinct skus in trnsact

(b) Use COUNT to determine how many instances there are of each sku associated with each store in the skstinfo table and the trnsact table?

You should see that there are multiple instances of every sku/store combination in the trnsact table, but only one instance of every sku/store combination in the skstinfo table. Therefore you could join the trnsact and skstinfo tables, but you would need to join them on both of the following conditions: trnsact.sku= kstinfo.sku AND trnsact.store= skstinfo.store.

Exercise 2: Use COUNT and DISTINCT to determine how many distinct stores there are in the strinfo, store_msa, skstinfo, and trnsact tables. Which stores are common to all tables, or unique to specific tables?

You should see that:
distinct stores in strinfo > # distinct stores in skstinfo > # distinct stores in store_msa
> # distinct stores in trnsact

Exercise 3: It turns out that there are many skus in the trnsact table that are not skstinfo table. As a consequence, we will not be able to complete many desirable analyses of Dillard's profit, as opposed to revenue, because we do not have the cost information for all the skus in the transact table. Examine some of the rows in the trnsact table that are not in the skstinfo table…can you find any common features that could explain why the cost information is missing?

Please note, the join you will need to complete this analysis will take a long time to run. This query will give you a good feeling for what working with enterprise-sized data feels like. Also, you might want to pin the results once you retrieve them so that you can examine the results later in the session.

Exercise 4: Although we can't complete all the analyses we'd like to on Dillard's profit, we can look at general trends. What is Dillard's average profit per day?

Make sure to specify the correct stype in your query. If you are interested in looking at the total value of goods purchased or returned, use the "amt" field. If you are interested in looking at the total number of goods purchased or returned, use the "quantity" field. Pay close attention to what we learned in Exercise 1b.

Exercise 5: On what day was the total value (in $) of returned goods the greatest? On what day was the total number of individual returned items the greatest?

Make sure to specify the correct stype in your query. If you are interested in looking at the total value of goods purchased or returned, use the "amt" field. If you are interested in looking at the total number of goods purchased or returned, use the "quantity" field.

Exercise 6: What is the maximum price paid for an item in our database? What is the minimum price paid for an item in our database?

The answer to the minimum price question is likely evidence of more incorrect entries.

Exercise 7: How many departments have more than 100 brands associated with them, and what are their descriptions?

A HAVING clause will be helpful for addressing this question. You will also need a join to combine the skuinfo and deptinfo tables in order to retrieve the descriptions of the departments.

Exercise 8: Write a query that retrieves the department descriptions of each of the skus in the skstinfo table.

You will need to join 3 tables in this query. Start by writing a query that connects the skstinfo and skuinfo tables. Once you are sure that query works, connect the deptinfo table. You may want to explore what happens when you incorporate aggregate functions into your query. When you do so, make sure all of your column names are referenced by the correct table aliases.

Exercise 9: What department (with department description), brand, style, and color had the greatest total value of returned items?

You will need to join 3 tables in this query. Start by writing a query that connects 2 tables. Once you are sure that query works, connect the 3rd table. Make sure you include both of these fields in the SELECT and GROUP BY clauses. Make sure any non-aggregate column in your SELECT list is also listed in your GROUP BY clause.

Exercise 10: In what state and zip code is the store that had the greatest total revenue during the time period monitored in our dataset?

You will need to join two tables to answer this question, and will need to divide your answers up according to the "state" and "zip" fields. Make sure you include both of these fields in the SELECT and GROUP BY clauses. Don't forget to specify that you only want to examine purchase transactions (not returns).

Syntax error checklist

If your query is crashing and you can't figure out why, make sure:

- all keywords are spelled correctly
- all keywords are in the correct order
- aliases do not have keywords in them
- quotation marks are of the correct type (note that when you paste code you worked on in a text document outside of the Teradata interface into SQL Scratchpad, sometimes the quotation marks do not translate appropriately and need to be corrected)
- all text strings are enclosed with the appropriate types of quotation marks
- semi-colons are at the end of your query, not in the middle
- all opening parentheses are matched with a closing parentheses
- there are commas between all the items in a list
- there is NOT a comma after the last item in a list
- each column name is linked with the correct table name
- all the necessary join conditions are included for each join
- aliases and title names do not contain white spaces (unless the full title is encased in quotation marks)

Test your understanding using this week's graded quiz once you feel you fully understand how to do the following in both MySQL and Teradata:

- Summarize your data using the aggregate functions MIN, MAX, SUM, and AVG
- Count the number of total rows, and the number of distinct rows, in your data
- Group your data and/or summaries according to values in a column
- Restrict the groups you retrieve according to specific criteria
- Combine information from two tables using inner, left, and right joins
- Combine information from more than two tables using inner, left, and right joins