WEEK 2 TERADATA EXERCISE GUIDE

Specific Goals for Week 2 Teradata Exercises

Use these exercises to:

- Become comfortable with the SQL Scratchpad
- Become familiar with the Dillard's database
- Recognize syntax differences between Teradata and MySQL
- Ensure you understand how to implement all the SQL syntax we learned this week

Getting started with Teradata

Please refer to the "How to Login to and Use Teradata Viewpoint (written instructions)" handout (https://www.coursera.org/learn/analytics-mysql/supplement/nyfp1/how-to-login-to-and-use-teradata-viewpoint-written-instructions) and the "How to Use Teradata Viewpoint and SQL Scratchpad" video to learn how to gain access to the Dillard's dataset, and how to navigate the SQL Scratchpad interface. This guide assumes you have signed into your Teradata account successfully and know how to execute queries in the query window.

SQL Scratchpad is exclusively configured for SQL, so unlike Jupyter, you do not need to write a line of code to tell it to load an SQL library or include "%sql" or "%%sql" before your queries (in fact, the query will crash if you do). However, it is a good idea to make the Dillard's database your default database. To do that, execute the following command:

```
DATABASE ua_dillards;
```

I suggest that you execute this command at the beginning of every Teradata session.

Get to know your data in Teradata

As I told you in the Jupyter exercises, one of the first things you should do when you start working with a database is confirm how many tables each database has, and identify the fields contained in each table of the database. You use different commands to do this in Teradata than you use in MySQL. Instead of using SHOW or DESCRIBE to get a list of columns in a table, use:

```
HELP TABLE [name of table goes here; don't include the brackets when executing the query]
```

To get information about a single column in a table, you could write:

```
HELP COLUMN [name of column goes here; don't include the brackets when executing the query]
```

The output of these commands will be a table with many columns. The important columns for you to pay attention to are "Column Name" which tells you the name of the column, and "Nullable" which will have a "Y" if null values are permitted in that column and an "N" if null values are not permitted.

One thing that is missing from the information outputted by HELP is whether or not a column is a primary or foreign key. In order to get that information, use a SHOW command:

```
SHOW table [insert name of table here; don't include the brackets when executing the query];
```

However, SHOW does something different in Teradata than it does in MySQL. Teradata uses SHOW to give you the actual code that was written to create the table. You can ignore much of the SHOW output for our purposes, but the end of the create table statement tells you what defaults were set for each column in the table. For example, in the following output:

```
STORE INTEGER NOT NULL,
CITY CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,
STATE CHAR(2) CHARACTER SET LATIN NOT CASESPECIFIC,
ZIP CHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
PRIMARY KEY ( STORE ))
```

The STORE column was configured so that it would not accept null values, the CITY, STATE, and ZIP columns were configured so their values would not be case-specific, and the column STORE was defined as the primary key of the table.

Exercise 1. Use HELP and SHOW to confirm the relational schema provided to us for the Dillard's dataset shows the correct column names and primary keys for each table.

The relational schema can be found in the "Dillard's Database Information" reading in the "Meet Your Business Data sets" lesson.

Look at your raw data

Most of the syntax you use to look at your data in Teradata is the same you use in MySQL. One of the main differences is Terdata uses a TOP operator instead of a LIMIT operator to restrict the length of a query output. Whereas LIMIT comes at the end of a MySQL query, TOP comes immediately after SELECT in a Teradata query. The following statement would select the first 10 rows of the strinfo table as they are stored in the native database:

```
SELECT TOP 10 *
FROM strinfo
```

The following statement would select the first 10 rows of the strinfo table, ordered in ascending alphabetical order by the city name (you would retrieve names that start with "a"):

```
SELECT TOP 10 *
FROM strinfo
ORDER BY city ASC
```

The following statement would select the first 10 rows of the strinfo table, ordered in descending alphabetical order by the city name (you would retrieve names that start with "w" and "y"):

```
SELECT TOP 10 *
FROM strinfo
ORDER BY city DESC
```

The documentation for the TOP function can be found here:

http://www.info.teradata.com/htmlpubs/db_ttu_13_10/index.html#page/SQL_Reference/B035_1146_109A/ch01.3.095.html

The Teradata TOP function does not allow you to start the number of rows you select at a certain row number. To select specific rows you need to use functions such as RANK or ROW_NUMBER within subqueries, but we will not learn about subqueries until the last week of the course. The SQL Scratchpad does allow you to page through your output, though. In addition, there is a function available in Teradata (but not MySQL) called SAMPLE that allows you to select a random sampling of the data in a table:

http://www.teradatawiki.net/2013/10/Teradata-SAMPLE-Function.html

The following query would retrieve 10 random rows from the strinfo table:

```
SELECT *
FROM strinfo
SAMPLE 10
```

The following query would retrieve a random 10% of the rows from the strinfo table:

```
SELECT *
FROM strinfo
SAMPLE .10
```

Every time you run the queries, they will return a different selection of rows.

The other important differences between Teradata and MySQL you need to know about are:

- DISTINCT and LIMIT can be used in the same query statement in MySQL, but DISTINCT and TOP cannot be used together in Teradata
- MySQL accepts either double or single quotation marks around strings of text in queries, but Teradata will only accept single quotation marks
- MySQL will accept the symbols "!=" and "<>" to indicate "does not equal" but Teradata will only accept "<>" (other operators, like "IN", "BETWEEN", and "LIKE" are the same: http://www.teradatawiki.net/2013/09/Teradata-Operators.html)

Keep these differences in mind, and remember that the Dillard's database was configured so that the names of the tables and columns are case insensitive.

Exercise 2. Look at examples of data from each of the tables. Pay particular attention to the skuinfo table.

Some things to note:

- There are two types of transactions: purchases and returns. We will need to make sure we specify which type we are interested in when running queries using the transaction table.
- There are a lot of strange values in the "color", "style", and "size" fields of the skuinfo table. The information recorded in these columns is not always related to the column title (for example there are entries like "BMK/TOUR K" and "ALOE COMBO" in the color field, even though those entries do not represent colors).
- The department descriptions seem to represent brand names. However, if you look at entries in the skuinfo table from only one department, you will see that many brands are in the same department.

Exercise 3. Examine lists of distinct values in each of the tables.

Note which tables have fewer distinct rows that they have total rows.

Exercise 4. Examine instances of transaction table where "amt" is different than "sprice". What did you learn about how the values in "amt", "quantity", and "sprice" relate to one another?

Exercise 5. Even though the Dillard's dataset had primary keys declared and there were not many NULL values, there are still many bizarre entries that likely reflect entry errors. To see some examples of these likely errors, examine:
(a) rows in the trsnact table that have "0" in their orgprice column (how could the original price be 0?),
(b) rows in the skstinfo table where both the cost and retail price are listed as 0.00, and

(c) rows in the skstinfo table where the cost is greater than the retail price (although occasionally retailers will sell an item at a loss for strategic reasons, it is very unlikely that a manufacturer would provide a suggested retail price that is lower than the cost of the item).

Exercise 6. Write your own queries that retrieve multiple columns in a precise order from a table, and that restrict the rows retrieved from those columns using "BETWEEN", "IN", and references to text strings. Try at least one query that uses dates to restrict the rows you retrieve.

Dates in the Dillard's database

Write a query to look at the sale price and date of transactions in the trnsact table. You will see that the dates of the transactions are outputted in the format YY/MM/DD. Seeing this result, you might make the reasonable assumption that you should use the YY/MM/DD format when referring to dates in your queries. If you follow that assumption, though, you will get an error that looks something like this:

```
Error Message - [Teradata Database] [TeraJDBC 15.10.00.05] [Error 3535] [SQLState
22003] A character string failed conversion to a numeric value.
```

The reason you get that answer can be learned through entering the query:

```
HELP TABLE trnsact
```

The result of this query shows you that saledate was configured as a date in the format of "YYYY-MM-DD", so although date data were permitted to be entered in a different format, the "YYYY-MM-DD" format is what you must use in your queries. If you use the "YYYY-MM-DD" format in your queries, you will not get the character string error discussed above.

We will not be exporting data from the Dillard's database

We made an agreement with University of Arkansas and Teradata that we will not export or share the data in the Dillard's data set. Please honor this agreement, so that we can make sure this terrific resource remains available for future students!

What to do when you don't know how to answer an exercise

If you are having trouble writing some of your queries, don't worry! Here are some things you can try:

1. Make sure your query is consistent with the requirements listed in the Syntax Error Checklist: https://www.coursera.org/learn/analytics-mysql/resources/AaIGu

2. Break down your query into the smallest pieces, or clauses, possible. Once you make sure each small piece works on its own, add in another piece or clause, one by one, always making sure the modified query works before you add in another clause.

3. Search the previous Discussion forum posts to see if other students had similar questions or challenges.

4. If you've exhausted all your other options, ask for help in the Discussion forums. Remember to (a) list the exercise name and question number in the title of the post or at the very beginning of the post, and (b) copy and paste the text of the question you are trying to answer at the beginning of the post. If you would like help troubleshooting a query, include what query (or queries) you tried, the error(s) you received, and the logic behind why you wrote the query (or queries) the way you did.

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:

- retrieve multiple columns in a precise order from a table
- select distinct rows from a table
- rename columns in a query output
- restrict the data you retrieve to meet certain criteria
- sort your output
- reference parts of text "strings"
- use "BETWEEN" and "IN" in your query statements