WEEK 2 TERADATA PRACTICE EXERCISES GUIDE

I hope you enjoyed learning to write your own queries through the Jupyter interface that allowed you to analyze Dognition's data! With that practice under your belt, you are ready to try writing queries more independently. The weekly "Teradata Practice Exercises," in combination with the weekly graded quizzes, will guide you through how to do that using the SQL Scratchpad in Teradata's Viewpoint online user interface.

The reason you are learning SQL is so that you can retrieve whatever data you need to analyze from a database, without having to ask somebody else in your company to do it for you. Sometimes you will be making the judgment about what data are important to retrieve, while other times your team members or bosses will be telling you what data they want you to retrieve. In either case, you will need to be able to sit in front of a blank query screen and somehow get the data you intend. That experience, emotionally, feels very different than writing queries through the Jupyter interface, because you will not have any text or guidance to draw upon to help you figure out how to interpret the database and generate a query.

The Teradata exercises you will complete in this course are designed to serve as a stepping stone between these real-life situations and the Jupyter exercises. I want you to be able to try out writing your own queries to answer real business questions with real business data just as you would in a company, but be able to take as much time as you need to figure out how to arrive at the correct answer, and be able to check your answers after you have finalized your query. I am going to give you much less guidance in the Teradata exercises than I did in the Jupyter exercises so that you can practice generating queries on your own. If you find it challenging to do this at first, don't worry, many people do! That's exactly why I wanted you to be able to overcome those challenges now, instead of when you are in your first analyst job.

Another important opportunity these Teradata exercises provide is the opportunity to see for yourself what working with "Big Data" feels like. I have tried to design some questions to start with that won't take too long to run, but as the course goes on and queries become more complicated, you will find that it may take minutes for some queries to output a result. Teradata is a very fast platform, so these query times represent some of the fastest query times you are likely to see in a business setting. I encourage you to really take advantage of this opportunity to get a feeling for how long different types of queries take to implement.

The best way to become truly proficient with SQL is to practice. To motivate you to practice, the quizzes in this course will be mostly based on the answers you retrieve from queries you write to analyze the Dillard's dataset. You may want to save some of the queries you write going through these ungraded exercises so that you can use them to answer graded quiz questions (you can save the queries in your Teradata account, or copy them to a document you save on your computer). Saving the queries will also give you a portfolio you can show to employers, or refer to after you have completed the course.

You should plan on allotting a reasonable amount of time to finishing the quizzes each week. Since the quizzes are based on writing actual queries, they may take a while to complete. More complex queries also take longer to run, so the exercises and quizzes in the next two weeks of the course will take longer to complete than this week's exercises and quiz. Take as much time as you need, and help each other out. I strongly encourage you to use the course discussion forums and Stack Overflow (www.stackoverflow.com) to ask questions and help other people when they have questions as well.

The one last thing I want to say before starting the Teradata exercises is that this course is primarily designed to help you gain the technical skillset that will allow you to write SQL queries. However, that will be only part of your job as a data analyst. The other part of your job will be to know your dataset deeply enough to write queries that give **correct, meaningful answers that help you and your company make good decisions about what actions to take.** Real data are messy, and if you don't pay attention to the details of the mess, you might execute SQL queries that lead you and your company to incorrect conclusions. We won't go into depth about how to analytically or physically clean data in this course, but I will ask you to write queries that take some of the abnormalities of the data into account. Doing so will make your queries more complicated, but it is important that you learn how to write queries that address imperfections in the data. I encourage you to think about these exercises as an opportunity to learn how to arrive at correct answers, more so than an opportunity to run queries that don't crash.

Overall Objectives for the Teradata Practice Exercises

Week 2: This week your aims should be to become comfortable with the Teradata front end interface, and the small differences in MySQL and Teradata SQL syntax. You should also aim to get a good sense of what the Dillard's data set contains, and what kind of analysis questions the data set could help you answer. Finish the week knowing how to retrieve, and change the format of, data from single tables that meet specified criteria.

Week 3: In Week 3 your aims should be to become very confident in your ability to use the main keywords in a SQL query, and to start to appreciate how long queries take when you are combining data from multiple tables with millions of rows. Finish the week knowing how to summarize data, how to segment summaries of your data, and how to combine data across multiple tables.

Week 4: In Week 4 your aims should be to become comfortable translating analysis questions into SQL queries, and confident that you can generate creative approaches to retrieving data in formats that are different from the way they are stored in a database. You should also aim to gain a deeper appreciation of how long real queries take to write and to execute. Overall, you want to finish this week feeling like a completely independent SQL analyst. You may still have questions, but you will know what questions to ask to succeed, and will know how to implement any answers or help you receive.

Specific Goals for Week 2's 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 Use Teradata Viewpoint and SQL Scratchpad Written Instructions" and "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]
```

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

```
HELP COLUMN [name of column goes here]
```

The output of these commands will be a table with several columns. The important columns for you to notice 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];
```

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 that 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 that the relational schema provided to us for the Dillard's dataset shows the correct column names and primary keys for each table.

Look at your raw data

Most of the syntax you use to look at your data in Teradata is the same as you use in MySQL. One of the main differences is that 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 would 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)

Keeping these differences in mind, and knowing 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 in which type we are interested, 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 strange 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.

We will not be exporting data from the Dillard's dataset, so 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