Homework Assignment #6

(Spark)

You have been working all quarter long on different database systems and experiencing their varying environments and approaches to handling and querying data to find useful insights. Your curiosity is driving you to explore another interesting framework, one that enables users to execute powerful parallel query tasks simultaneously. You have been hearing about Spark for a while now and feel that this is the perfect time to get some hands-on experience with it. You decided to talk to a colleague who has some experience using Spark to get help in loading the cleaned version of the ShopALot data, and your colleague delivered - so your data now awaits in the cloud! They have advised you to use the Databricks Community Edition of Spark that is provided by Databricks for free exploratory use in the cloud.

Get Ready… (and set)

To start, follow the (separately provided) setup instructions **carefully** in order to create a Community Edition account with Databricks to use for your adventures here.

Go...!

It's time to explore the new system! Begin your adventure by opening the provided Spark notebook template file and then **describe**-ing the schema of the preloaded data frames along with their associated created views.

1. Start by running some schema description statements in order to better understand the preloaded relations.
    A. First explore the schema for the *orders* relation. Write a **python** fragment using Spark **dataframes** to show the schema of *orders*.
    B. What is the data type of the *time_fulfilled* field?
    C. Now let's see how to view a schema using SQL in Spark. Write a **SQL statement** to view the schema of *users*.
    D. What is the data type of the *phones* field?

2. After scratching the surface on how to use Spark, you are eager to start writing queries and analyzing the data! Answer questions [2.A - 2.F] in **TWO** ways - first by providing dataframe code fragments and then by providing SQL statements - to answer each of the desired queries.
    A. To start your Spark journey, you would like to view the orders(s) for the customer with customer_id 'JVN1X'.
    B. Running aggregation queries on top of data is a powerful way to analyze data. Print the number of products in each **multi-kind product category** (one with more than one kind of thing, detectable by the presence of an '&' in its name, such as 'Fruits & Vegetables' or 'Paper, Cleaning, & Home'). Include both the name of the multi-kind category and its total number of products.

C. You want to view some information about the most popular stores. Find the top 5 stores in terms of the number of orders that they have fulfilled. Return only their store_ids. **Note:** "top 5" implies they should be listed in descending order.
D. You have heard that Spark can deal with array data fairly well and you want to explore this feature. Write a query that prints the category carried by the largest number of stores. Return only the category's name.
E. Join is one of the most interesting operations for data analysis since it helps in connecting the dots when it comes to data, so you want to understand how joins work in Spark. Find the names of shoppers that fulfilled an order between 2020-05-01 inclusive and 2020-06-01 exclusive (*time_fulfilled* field). Include the shopper's full name and order your output by the time_fulfilled, in ascending order, and limit your results to 5 names.
F. Using joins and aggregation together can be a bit tricky. You want to view the **big customers**, namely the ones who have purchased a total of more than $650 worth of orders and who also own more than one vehicle. Return the names and emails of these big customers and order your report by their email addresses in ascending order.
G. [For this query you need to provide only a **SQL statement.**] You now have some mastery of the Spark engine and you're feeling like you're ready to tackle a more complex problem. For each store, we would like to view the category that has been ordered from the most and the average selling price of products in that category for that specific store. Display the store name, the category, and the average price. Order your results by the average price in descending order and limit your results to the top 10 stores.
H. **[Extra Credit]** Having heard about Map/Reduce jobs and their important role in Big Data history, your curiosity is getting the better of you. Spark still supports the Map/Reduce model (and more) via its low-level RDDs. Using only Spark's map and reduce RDD transformation functions, rewrite query 2B in the Map/Reduce paradigm. (Now you can claim to be a **real** data engineer!) Note that you will first have to convert any/all relevant dataframes to RDDs in order to use the (Python) Map/Reduce operations on them. A link that might help: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.rdd.html#pyspark.sql.DataFrame.rdd
Note: Please refrain from (publicly) discussing the extra credit on Piazza! This is a self-service, all-on-your own (with or without your brainstorming buddy) problem for those feeling adventurous!

What To Turn In

When you have finished your assignment, you should use Gradescope to turn in the pdf of your notebook file that lists all queries you ran -from start to finish-. Include your answers to questions that aren't queries as comments. Please follow the steps below in order to generate the file for submission:

1. Download the sample .ipynb file from the CS122D website. You should have used this in the setup document.
2. After answering the queries both ways (i.e., Dataframe and SQL), make sure you can run the entire file from start-to-finish and get no execution errors when you run it.
3. Click File -> export -> HTML. Then, you can right click -> Print Preview and then PDF-print the result. Once done, double check that all the code and **the results** are visible in your PDF file.
4. Submit your PDF file to Gradescope!