DSCI 617 – HW 03 Instructions

General Instructions

Navigate to the **Homework** folder inside of your user directory in the Databricks workspace. Create a notebook named **HW_03** inside the **Homework** folder.

Any set of instructions you see in this document with an orange bar to the left will indicate a place where you should create a markdown cell. For each new problem, create a markdown cell that indicates the title of that problem as a level 2 header. Any set of instructions you see with a blue bar to the left will provide instructions for creating a single code cell.

Add a markdown cell that displays the following text as a level 1 header: **DSCI 617 – Homework 03**. Within the same cell, on the line below the header, add your name in bold.

Add a code cell to set up the environment for this assignment. Import **`pandas`** and **`matplotlib.pyplot`** using the standard aliases. Import the **`SparkSession`** class as well as the **`col`** and **`expr`** functions.

Then create a **`SparkSession`** object named **`spark`**.

Problem 1: Terminology

Create a markdown containing a numbered list with answers to the following questions. You only need to provide the answers and are not required to retype the questions.

1. What is the name of the class that is used to define a schema?
2. What is the name of the class that is used to represent a column within a schema?
3. What is the name of a class used to indicate that a column is intended to store floating point values?
4. What DataFrame method is used to display the contents of the DataFrame?
5. What two DataFrame methods are used to display summary statistics (such as count, mean, standard deviation, min, and max) for each column in a DataFrame?
6. What DataFrame method is used to remove rows with missing entries from the DataFrame?
7. Name two methods that can be used to add columns to a DataFrame.
8. What method is used to perform one or more custom aggregations after grouping?
9. Name two Spark functions that accept as arguments strings representing names of DataFrame columns and return column objects.
10. What DataFrame transformation returns a DataFrame containing all of the columns from the original DataFrame, but only the rows that satisfy a stated condition?

Problem 2: Columns and Expressions

Assume that you have access to a DataFrame named **`sales_df`** with the following schema:

```
root
|-- customerID: string (nullable = true)
|-- productID: string (nullable = true)
|-- units: long (nullable = true)
|-- unit_price: double (nullable = true)
```

Determine which of the lines of code shown below would run without error. Create a markdown cell to provide your answer. The answer should be given in the form of a list of integers corresponding to the numbers of the lines that would run **without error**.

1. `sales_df.select('unit_price', 'units').show()`
2. `sales_df.select(col('unit_price'), expr('units')).show()`
3. `sales_df.select('unit_price * units').show()`
4. `sales_df.select(col('unit_price * units')).show()`
5. `sales_df.select(expr('unit_price * units')).show()`
6. `sales_df.select(col('unit_price') * col('units')).show()`
7. `sales_df.select(expr('unit_price') * expr('units')).show()`
8. `sales_df.select('sum(units)').show()`
9. `sales_df.select(col('sum(units)')).show()`
10. `sales_df.select(expr('sum(units)')).show()`

Problem 3: Diamonds Data (Part 1)

For the remainder of this assignment, we will again be working with the Diamonds dataset. Recall that the data file for this dataset is located at the path: **`/FileStore/tables/diamonds.txt`**

Recall also that each row of this data file contains the following information for a single diamond:

- **`carat`** Weight of the diamond.
- **`cut`** Quality of the cut. **Levels: `Fair`, `Good`, `Very Good`, `Premium`, `Ideal`**
- **`color`** Diamond color. **Levels: `J`, `I`, `H`, `G`, `F`, `E`, `D`**
- **`clarity`** A measure of diamond clarity. **Levels: `I1`, `SI2`, `SI1`, `VS2`, `VS1`, `VVS2`, `VVS1`, `IF`**
- **`depth`** Total depth percentage
- **`table`** Width of top of diamond relative to widest point
- **`price`** Price in US dollars
- **`x`** Length in mm
- **`y`** Width in mm
- **`z`** Depth in mm

We will begin by loading the dataset into a DataFrame. For the purposes of creating a schema, note that the **`price`** column contains integer values. The **`cut`**, **`color`**, and **`clarity`** columns each contain string values. All other columns contain floating point (or double) values.

Create a code cell to complete the following tasks:

1. Create a custom schema for this dataset. I recommend using a DDL string.
2. Read the file into a DataFrame named **`diamonds`** using **`spark.read()`**. Note that the file is tab-delimited and has a header.
3. Use **`printSchema()`** to display the DataFrame's schema.

Next, we will calculate the number of records in in the dataset.

Create a new code cell and use it to display the number of rows in the **`diamonds`** DataFrame.

To get a sense as to what the data looks like, we will display the first 10 rows of the DataFrame.

Use the **`show()`** method to display the first 10 rows of the **`diamonds`** DataFrame.

Lastly, we will explore the relationship between carat size and diamond price by drawing a sample from the DataFrame and then generating a plot of price versus carat size using the observations in the sample.

Create a code cell to complete the following tasks:

1. Use the **`sample()`** method to draw a sample from **`diamonds`**. Use **`fraction=0.25`** and **`seed=1`**. Convert the sample to a Pandas DataFrame and store the result in **`sample_pdf`**.
2. Use the data in the sample to create a scatter plot of **`price`** versus **`carat`**. When creating the scatter plot, set **`alpha=0.5`** and select a named color for the points. Label the x-axis **`"Carat"`** and label the y-axis **`"Price"`**. Use **`plt.show()`** to display the plot.

Note that we could have easily plotted the entire dataset in the cell above, rather than limiting ourselves to a sample. However, when working with very large dataset in a distributed setting, this is not always feasible. The process above can be used in these situations.

Problem 4: Diamonds Data (Part 2)

We will continue working with the Diamonds dataset in this problem. We will start by using sorting to identify the most expensive diamonds in the dataset.

Sort the contents of the **`diamonds`** DataFrame in descending order by **`price`**. Use **`show()`** to display the first 5 rows of the sorted DataFrame.

New, we will identify the largest five diamonds in the dataset.

Sort the contents of the **`diamonds`** DataFrame in descending order by **`carat`**. Use **`show()`** to display the first 5 rows of the sorted DataFrame.

In the next two cells, we will explore the price per carat for diamonds in the dataset.

Create a code cell to complete the following tasks:

1. Create a new DataFrame named **`diamonds_ppc`**. This DataFrame should contain all columns from **`diamonds`** but should also contain a column named **`price_per_carat`**. Values in this new column should be equal to the price of the diamond divided by the carat size, rounded to 2 decimal places.
2. Sort the contents of the **`diamonds_ppc`** DataFrame in descending order by **`price_per_carat`**. Use **`show()`** to display the first 5 rows of the sorted DataFrame.

In the previous cell, we identified the diamonds with the highest price per carat. We will now identify the diamonds with the lowest.

Sort the contents of the **`diamonds_ppc`** DataFrame in ascending order by **`price_per_carat`**. Use **`show()`** to display the first 5 rows of the sorted DataFrame.

In the last part of this problem, we will graphically explore the relationship between **`price_per_carat`** and **`carat`**.

Create a code cell to complete the following tasks:

1. Use the **`sample()`** method to draw a sample from **`diamonds_ppc`**. Use **`fraction=0.25`** and **`seed=1`**. Convert the sample to a Pandas DataFrame and store the result in **`ppc_ample_pdf.`**
2. Use the data in the sample to create a scatter plot of **`price_per_carat`** versus **`carat`**. When creating the scatter plot, set **`alpha=0.5`** and select a named color for the points. Label the x-axis **`"Carat"`** and label the y-axis **`"Price per Carat"`**. Use **`plt.show()`** to display the plot.

Submission Instructions

When you are done, click **Clear State and Run All**. If any cell produces an error, then manually run every cell after that one, in order. Export the notebook as an HTML file and then upload this file to Canvas. Do not alter your notebook on DataBricks after submitting unless you intend to resubmit a new HTML file. The notebook on Databricks should match the HTML file on Canvas.