Problem Set E Submission Form

# Overview

|  |  |
| --- | --- |
| Your Name | Jaya Varshini Prabakar |
| Your SU Email | jprabaka@syr.edu |

# Instructions

Put your name and SU email at the top. Answer these questions all from the lab. When asked to include screenshots, please follow the screen shot guidelines from the first homework.

Remember as you complete the homework it is not only about getting it right / correct. We will discuss the answers in class so it’s important to articulate anything you would like to contribute to the discussion in your answer:

* If you feel the question is vague, include any assumptions you've made.
* If you feel the answer requires interpretation or justification provide it.
* If you do not know the answer to the question, articulate what you tried and how you are stuck.
* Highlight any doubts or questions you would like me to review.

This how you receive credit for answering questions which might not be correct. In addition, you must complete the reflection portion of the homework assignment for full credit. Since most answers will be similar this is an important part of your individual submission.

Complete Part II of this document first, then go back and complete the Reflection in Part I.

# Part I - Reflection

Use this section to reflect on your learning. To achieve the highest grade on the assignment you must be as descriptive and personal as possible with your reflection.

1. As you completed this assignment, identify what you learned.  
   Apache Drill, Spark SQL and interactions.
2. What barriers or challenges did you encounter while completing this assignment?  
   I did not face any barriers or challenges
3. How prepared were you to complete this assignment? What can you do to be better prepared?  
   Yes, I was prepared to complete the assignment
4. Rate your comfort level with this week’s material. Use the rubric provided.

4  
4 ==> I understand this material and can explain it to others.  
3 ==> I understand this material.  
2 ==> I somewhat understand the material but sometimes need guidance from others.  
1 ==> I understand very little of this material and need extra help.

# Part II – Questions

**For each question, include a copy of the code required to complete the question along with a screenshot of the code and a screenshot of the output.**

1. Configure a Drill storage plugin for the Minio **labe** bucket. Then write a drill query for **syracuse-ny.csv** to demonstrate you can read the file with headers.   
   **Drill Storage Plugin for CSV:**

**Code:**

"csv": {

"type": "text",

"extensions": [

"csv"

],

"lineDelimiter": "\n",

"fieldDelimiter": ",",

"quote": "\"",

"escape": "\"",

"comment": "#",

"extractHeader": true

}

A screenshot of a computer

Description automatically generated

**Drill Query:**

**A screenshot of a computer

Description automatically generated**

**Output:**

**A screenshot of a computer

Description automatically generated**

1. Write a Drill SQL Query to get the overall average min and max temperatures by year and month. Use drill’s SPLIT() function to separate Year, Month. You might need to use cast() to ensure the min and max temperatures are numeric types. You output should include 4 columns: Year, Month, the average minimum temperature for that month, and the average maximum temperature for that month.   
   **Query:**

with temp as (

select

cast(split(`EST`,'-')[0] as int) as year,

cast(split(`EST`,'-')[1] as int) as month,

cast(Min\_TemperatureF as int) as average\_min\_temp,

cast(Max\_TemperatureF as int) as average\_max\_temp

from labe.`syracuse-ny.csv`

)

select year, month, avg(average\_min\_temp) as avgmin, avg(average\_max\_temp) as avgmax

from temp

group by year,month

order by year,month

**Query Screenshot:**

**A screenshot of a computer

Description automatically generated**

1. Create a view called **monthly\_syracuse\_weather\_averages** from the query you wrote in question 2 and store it back on the **labe** bucket. (If you cannot get question 2 working, use a similar query). Provide your drill SQL code and a screenshot showing the view file is on the Minio bucket.  
   NOTE: If you get an error about an immutable object, you need to change your storage config so you can write to the storage location.  
   **Drill SQL code:**

create view labe.monthly\_syracuse\_weather\_averages

as

with temp as (

select

cast(split(`EST`,'-')[0] as int) as year,

cast(split(`EST`,'-')[1] as int) as month,

cast(Min\_TemperatureF as int) as average\_min\_temp,

cast(Max\_TemperatureF as int) as average\_max\_temp

from labe.`syracuse-ny.csv`

)

select year, month, avg(average\_min\_temp) as avgmin, avg(average\_max\_temp) as avgmax

from temp

group by year,month

order by year,month

**Minio Screenshot:**

**A screenshot of a computer

Description automatically generated**

1. Use the view you created in question 3 to show the weather data only the month of July.   
   **Drill SQL Query:**

select \* from labe.monthly\_syracuse\_weather\_averages

where month = 7

**Output:**

**A screenshot of a computer

Description automatically generated**

1. Configure spark to read from Minio **labe** bucket, then load **syracuse-ny.csv** into a DataFrame as register it as the table **weather   
   spark code:**

spark.read.option("header",True)\

.option("inferSchema",True)\

.csv("s3a://labe/syracuse-ny.csv")\

.createOrReplaceTempView("weather")

spark.sql("select \* from weather").show()

**Output:**

**A screenshot of a computer

Description automatically generated**

1. Rewrite question 2 using pure Spark SQL and the **weather** temp view. NOTE: There will be some subtle differences with how you must write the code, so be sure to **printSchema()** so you can see what the columns are.

**Spark Query:**

query = '''

with source as(

select

cast(split(EST,'-')[0] as int) as year,

cast(split(EST,'-')[1] as int)as month,

`Min TemperatureF` as mintemp,

`Max TemperatureF` as maxtemp

from weather

)

select

year, month, avg(mintemp) as avgmin,

avg(maxtemp) as avgmax

from source

group by year, month

order by year, month

'''

spark.sql(query).show()

**Output:**

A screenshot of a computer

Description automatically generated

1. Save the output from the DataFrame in question 6 to the temp view **monthly\_syracuse\_weather\_averages**. Prove the view is there by querying it.

**Code:**

query = '''

with source as(

select

cast(split(EST,'-')[0] as int) as year,

cast(split(EST,'-')[1] as int)as month,

`Min TemperatureF` as mintemp,

`Max TemperatureF` as maxtemp

from weather

)

select

year, month, avg(mintemp) as avgmin,

avg(maxtemp) as avgmax

from source

group by year, month

order by year, month

'''

spark.sql(query).createOrReplaceTempView("monthly\_syracuse\_weather\_averages")

spark.sql("show tables").show()

spark.sql(f"select \* from monthly\_syracuse\_weather\_averages where month =7").show()

**Output:**

A screenshot of a computer

Description automatically generated

1. CHALLENGE YOURSELF! At the bottom of the **work/content/E-Drill-Spark.ipynb**  file there is a section Called “Big Data to Small Data”. Try to write a complete program that:
   1. Inputs a month 1 – 12 at run-time.

**Code:**

month = input("Enter Month:")

spark.sql(f"select \* from monthly\_syracuse\_weather\_averages where month = {month}").show()

**Output:**

**A screenshot of a computer

Description automatically generated**

* 1. Displays a scatter plot of min/max average monthly temperatures, where year is on the X-Axis.

**Code:**

from IPython.display import display,HTML

from ipywidgets import interact\_manual

import matplotlib.pyplot as plt

display(HTML("<h1>Syracuse Weather<h1>"))

@interact\_manual(Month=(1,12))

def doit(Month):

df = spark.sql(f"select \* from monthly\_syracuse\_weather\_averages where month = {Month}").toPandas()

display(df)

df.set\_index("year")

plt.scatter(df["year"], y = df["avgmin"], label = 'monthly avg min')

plt.scatter(df["year"], y = df["avgmax"],label = 'monthly avg max')

plt.legend(title =f"Temps for Month {Month}")

plt.show()

**Output:**

**A screenshot of a computer

Description automatically generated**