EECS 4415 - Task 4

Analyzing COVID-19 with Spark's SQL API

Setup

Let's set up Spark on your Colab environment. Run the cell below!

```
!pip install pyspark
!pip install -U -q PyDrive
!apt install openjdk-8-jdk-headless -qq
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
```

Now we authenticate a Google Drive client to download the files we will be processing in our Spark job.

Make sure to follow the interactive instructions.

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# Authenticate and create the PyDrive client
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

```
WARNING:root:pydrive is deprecated and no longer maintained. We recommend that you migrate your projects to pydrive2, the maintained fork of pydrive
```

```
id='1YT7ttUAafCjbVdm6obeHp1TWAKOrEtoR'
downloaded = drive.CreateFile({'id': id})
downloaded.GetContentFile('time_series_covid19_confirmed_global.csv')

id='1YxEA5UQ2EFJ_9oLssM__Gs1ncVNufGNA'
downloaded = drive.CreateFile({'id': id})
downloaded.GetContentFile('time_series_covid19_deaths_global.csv')

id='1CNxszuZTelw-5cF5yrzKMZdb1qVOhSoy'
downloaded = drive.CreateFile({'id': id})
downloaded.GetContentFile('time_series_covid19_recovered_global.csv')
```

If you executed the cells above, you should be able to see the dataset we will use for this Colab under the "Files" tab on the left panel.

Next, we import some of the common libraries needed for our task.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import pyspark
from pyspark.sql import *
from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark import SparkContext, SparkConf
```

Let's initialize the Spark context. (If there is an errror based on the previous sessions, go to Runtime-> Restart session.)

```
# create the session

conf = SparkConf().set("spark.ui.port", "4050")

# create the context
```

```
sc = pyspark.SparkContext(conf=conf)
spark = SparkSession.builder.getOrCreate()
```

Data Loading

In this Colab, we will be analyzing the time series data of the Coronavirus COVID-19 Global Cases, collected by Johns Hopkins CSSE.

Here you can check a dashboard based on this dataset: https://www.arcgis.com/apps/opsdashboard/index.html?fbclid=IwAR2hQKsEZ3D38wVtXGryUhP9CG0Z6MYbUM_boPEaV8FBe71wUvDPc65ZG78#/bda7594740fd40299423467b48e9ecf6

- `confirmed`: dataframe containing the cumulative number of confirmed COVID-19 cases, divided by geographical area
- `deaths`: dataframe containing the cumulative number of deaths due to COVID-19, divided by geographical area
- `recovered`: dataframe containing the cumulative number of recoevered patients, divided by geographical area

The data sets contain data entries for each day, representing the cumulative totals as of that day.

```
confirmed = spark.read.csv('time_series_covid19_confirmed_global.csv', header=True)
deaths = spark.read.csv('time_series_covid19_deaths_global.csv', header=True)
recovered = spark.read.csv('time_series_covid19_recovered_global.csv', header=True)
```

```
confirmed.printSchema()
```

```
confirmed.show()
```

Your Task

Consider the entries for May 1, 2021, in the timeseries, and compute:

- number of confirmed COVID-19 cases across the globe
- number of deaths due to COVID-19 (across the globe)
- number of recovered patients across the globe

```
''' 6 lines of code in total expected. '''

# YOUR CODE HERE
confirmed_5_1_21 = confirmed.select(sum("5/1/21").alias("# of confirmed"))
deaths_5_1_21 = deaths.select(sum("5/1/21").alias("# of deaths"))
recovered_5_1_21 = recovered.select(sum("5/1/21").alias("# of recovered"))
confirmed_5_1_21.show()
deaths_5_1_21.show()
recovered_5_1_21.show()
```

```
+--------------+
|# of confirmed|
+--------------+
|  1.52196159E8|
+--------------+

+-----------+
|# of deaths|
+-----------+
|  3192930.0|
+-----------+

+--------------+
|# of recovered|
+--------------+
|   8.8919401E7|
+--------------+
```

Consider the data points for March 1, 2020, and March 1, 2021, and filter out the geographical locations where less than 50 cases have been confirmed. For the areas still taken into consideration after the filtering step, compute the ratio between number of deaths and number of confirmed cases. Show top 20 rows.

Hint: You do not need to sum over the country(combine Province/State having the same Country/Region). In other words, the column Province/State should exist in the final dataframe for this question.

```
''' 16-20 lines of code in total expected but can differ based on your style. '''

# YOUR CODE HERE
confirmed_filtered = confirmed.select("Province/State", "Country/Region",col("3/1/20").alias("confirmed_3/1/20"),W
                                      col("3/1/21").alias("confirmed_3/1/21"))W
                                      .where((confirmed["3/1/20"] >= 50) & (confirmed["3/1/21"] >= 50))

deaths_filtered = deaths.select("Province/State", "Country/Region",col("3/1/20").alias("deaths_3/1/20"), col("3/1/21").alias("deaths_3/1/21"))

combined = confirmed_filtered.join(deaths_filtered,W
                                   (confirmed_filtered["Province/State"] == deaths_filtered["Province/State"])W
                                   & (confirmed_filtered["Country/Region"] == deaths_filtered["Country/Region"]), "inner")

ratio = combined.select((col("deaths_3/1/20")/col("confirmed_3/1/20")).alias("ratio_3/1/20"),W
                        (col("deaths_3/1/21")/col("confirmed_3/1/21")).alias("ratio_3/1/21"))

ratio.show(20)
```

```
+-------------------+-------------------+
|        ratio_3/1/20|        ratio_3/1/21|
+-------------------+-------------------+
|0.006060606060606061|0.006036217303822937|
| 0.01937046004842615|0.008579599618684462|
|0.010416666666666666| 0.01015228426395939|
|0.003378378378378...|0.001814882032667...|
| 0.02197802197802198|  0.0106951871657754|
|0.005189028910303929|0.003611738148984...|
|0.007936507936507936| 0.00749063670411985|
|  0.0136986301369863|0.013605442176870748|
| 0.02976190476190476| 0.03508771929824561|
|0.018867924528301886|0.005315110098709187|
|0.027083333333333334|0.008074534161490683|
| 0.01729559748427673| 0.01685823754789272|
|0.020833333333333332|0.018150467374534893|
|  0.0412662352220246| 0.06620592507813532|
|0.0039292730844793 71|0.003861003861003861|
|                0.0|0.002724795640326...|
|                0.0|                0.0|
|0.001069518716577...|0.001069518716577...|
|0.010752688172043012|0.005235602094240838|
| 0.00819672131147541|0.004926108374384...|
+-------------------+-------------------+
only showing top 20 rows
```

Consider the data points for March 1, 2021, and May 1, 2021, in the timeseries, and filter out the geographical locations where less than 50 deaths have been confirmed. Show the top 20 rows for May 1, 2021 (as March 1, 2021 has already been shown before).

For the areas still taken into consideration after the filtering step, compute **the percent increase in cumulative deaths** between the two dates. Show top 20 rows.

```
''' 5-12 lines of code in total expected but can differ based on your style.'''

# YOUR CODE HERE
deaths_increment = deaths.select("Province/State","Country/Region","3/1/21", "5/1/21")W
.where((deaths["3/1/21"] >= 50) & (deaths["5/1/21"] >= 50))

deaths_increment.select("Province/State","Country/Region", "5/1/21").show(20)
```

```
+--------------+-------------------+------+
| Province/State|      Country/Region|5/1/21|
+--------------+-------------------+------+
|          NULL|        Afghanistan|  2631|
|          NULL|            Albania|  2396|
|          NULL|            Algeria|  3261|
|          NULL|            Andorra|   125|
|          NULL|             Angola|   600|
|          NULL|          Argentina| 64096|
|          NULL|            Armenia|  4128|
|New South Wales|          Australia|    54|
|      Victoria|          Australia|   820|
|          NULL|            Austria| 10233|
|          NULL|         Azerbaijan|  4538|
|          NULL|            Bahamas|   199|
|          NULL|            Bahrain|   648|
|          NULL|         Bangladesh| 11510|
```

```
|          NULL|             Belarus|  2552|
|          NULL|             Belgium| 24258|
|          NULL|              Belize|   323|
|          NULL|               Benin|    99|
|          NULL|             Bolivia| 13009|
|          NULL|Bosnia and Herzeg...|  8551|
+--------------+--------------------+------+
only showing top 20 rows
```

```
deaths_increment.select(((col("5/1/21")-col("3/1/21"))/col("3/1/21") * 100).alias("percentage of increase")).show(20)
```

```
+--------------------+
|percentage of increase|
+--------------------+
|    7.6513911620294595|
|    31.938325991189426|
|     9.173083361232006|
|    13.636363636363635|
|     18.11023622047244|
|    23.079286441231254|
|    29.201877934272304|
|                   0.0|
|                   0.0|
|    19.349195241427573|
|     40.80049643189575|
|      11.1731843575419|
|     43.36283185840708|
|     36.76330798479088|
|     28.56423173803526|
|     9.734913598118158|
|    2.5396825396825395|
|     41.42857142857143|
|    11.512086404937424|
|     68.62551764937882|
+--------------------+
only showing top 20 rows
```

Write a paragraph of conclusions below summarizing your insights.

The query object alone does not hold any subset of dataframe resides in the disk. Only when data retrieval commnad is issued, such as collect() or show(), the dataset is filtered by the query and brought in to the memory using SQL optimazation schme and more. So, there is little or no overhead of bring the whole dataset first into the memroy and filter it as long as our query filters fair enough amount of rows in the dataset. Although, if our query filters out little or no rows/columns, it would be the same as brining the whole dataset into the memory anyway.

Once you have working code for each cell above, **head over to eClass, and submit your solution for this Colab**!