Appendix: Python Code in Jupyter Notebook

(Available at github.com/pi-prescott/spark-standalone)

In [1]:
```
# [Q1] first we initiate a SparkSession with the spark-master

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("pyspark-notebook").\
        master("spark://spark-master:7077").\
        config("spark.executor.memory", "1024m").\
        getOrCreate()

spark
```

Out[1]: **SparkSession - in-memory**

SparkContext

Spark UI

Version v3.0.0
Master spark://spark-master:7077
AppName pyspark-notebook

In [2]:
```
# [Q2a] load CSV into spark dataframe
# and [Q2b] check schema
wrong_schema = spark.read.csv(path='../data/covid19.csv',header=True)
# by default the type of each column is apparently assumed to be String.
print(wrong_schema.printSchema())
```

```
root
 |-- continent: string (nullable = true)
 |-- location: string (nullable = true)
 |-- date: string (nullable = true)
 |-- total_cases: string (nullable = true)
 |-- new_cases: string (nullable = true)
 |-- total_deaths: string (nullable = true)
 |-- new_deaths: string (nullable = true)

None
```

In [3]:
```
# if you ask explicitly, spark will try to infer the schema automatically
infer_schema = spark.read.csv(
    path='../data/covid19.csv',header=True,inferSchema=True)
infer_schema.printSchema()
# in this case it gets the integers right, but just treats the date as a string
```

```
root
 |-- continent: string (nullable = true)
 |-- location: string (nullable = true)
 |-- date: string (nullable = true)
 |-- total_cases: integer (nullable = true)
 |-- new_cases: integer (nullable = true)
 |-- total_deaths: integer (nullable = true)
 |-- new_deaths: integer (nullable = true)
```

In [4]:
```
# or you can specify the schema explicitly

from pyspark.sql.types import (StructField,
                               StringType,
                               IntegerType,
                               DateType,
                               StructType)

data_schema = [StructField('continent',StringType(),True),
              StructField('location',StringType(),True),
              StructField('date',DateType(),True),
              StructField('total_cases',IntegerType(),True),
              StructField('new_cases',IntegerType(),True),
              StructField('total_deaths',IntegerType(),True),
              StructField('new_deaths',IntegerType(),True)]

correct_struc = StructType(fields=data_schema)

dataframe = spark.read.csv(
    path='../data/covid19.csv', header=True, schema=correct_struc)

# and we can confirm that this time the types are correct
print(dataframe.printSchema())
```

```
root
 |-- continent: string (nullable = true)
 |-- location: string (nullable = true)
 |-- date: date (nullable = true)
 |-- total_cases: integer (nullable = true)
 |-- new_cases: integer (nullable = true)
 |-- total_deaths: integer (nullable = true)
 |-- new_deaths: integer (nullable = true)

None
```

In [5]:
```
# if we wanted to convert to the older-style RDD we easily could
rdd = dataframe.rdd
print(f'Created `rdd` {type(rdd)} from `dataframe` {type(dataframe)}.')
# ... and vice versa
new_dataframe = rdd.toDF()
print(f'Created `new_dataframe` {type(new_dataframe)}'
      + f' from `rdd` {type(rdd)}.')
```

```
Created `rdd` <class 'pyspark.rdd.RDD'> from `dataframe` <class 'pyspark.sql.data
frame.DataFrame'>.
Created `new_dataframe` <class 'pyspark.sql.dataframe.DataFrame'> from `rdd` <cla
ss 'pyspark.rdd.RDD'>.
```

In [6]:
```
# the simplest way to drop null values from a spark 2.0 dataframe
# ...is like this
drop_na = dataframe.dropna()
```

In [7]:
```
# [Q2c] but we can use the `.filter()` method if we like
filtered_df = dataframe.filter(
        ' and '.join([f'{x} is not null' for x in dataframe.columns])
        )
```

In [8]:
```
print(f'Before filtering we had {dataframe.count()} rows...')
print(f'Using `.dropna()` leaves us {drop_na.count()} rows.')
print(f'Using `.filter()` leaves us {filtered_df.count()} rows.')
if drop_na.count() == filtered_df.count():
    print('Good, those numbers are the same!')
else:
    print('Not good -- those numbers should be the same...')
```

```
Before filtering we had 53087 rows...
Using `.dropna()` leaves us 39974 rows.
Using `.filter()` leaves us 39974 rows.
Good, those numbers are the same!
```

In [9]:
```
# [Q3] use aggregate and groupBy functions to see highest `total_deaths` in each
hi_total_deaths = filtered_df.groupBy('location')\
                        .agg({'total_deaths':'max'})
```

In [10]:
```python
hi_total_deaths.show()
```

```
+--------------------+-----------------+
|            location|max(total_deaths)|
+--------------------+-----------------+
|                Chad|               96|
|            Paraguay|             1327|
|              Russia|            26589|
|               Yemen|              600|
|             Senegal|              322|
|              Sweden|             5918|
|              Guyana|              119|
|              Jersey|               32|
|         Philippines|             7053|
|            Djibouti|               61|
|            Malaysia|              238|
|           Singapore|               28|
|                Fiji|                2|
|              Turkey|             9950|
|United States Vir...|               21|
|      Western Sahara|                1|
|              Malawi|              183|
|                Iraq|            10724|
|Sint Maarten (Dut...|               22|
|             Germany|            10183|
+--------------------+-----------------+
only showing top 20 rows
```

In [11]:
```python
# the assignment suggests that the number of total_deaths for Sweden should be 98
# however it is actually 5918
hi_total_deaths.filter(hi_total_deaths.location=='Sweden').show()
```

```
+--------+-----------------+
|location|max(total_deaths)|
+--------+-----------------+
|  Sweden|             5918|
+--------+-----------------+
```

In [12]:
```python
# however, we would get the result 986 if we hadn't
# explicitly made sure to load the CSV with the correct schema
wrong_schema.groupBy('location').agg({'total_deaths':'max'})\
            .filter(wrong_schema.location=='Sweden').show()
```

```
+--------+-----------------+
|location|max(total_deaths)|
+--------+-----------------+
|  Sweden|              986|
+--------+-----------------+
```

In [13]:
```python
# [Q4] use max and min functions to see which country
# has highest and lowest `total_cases`
# NB: 'total_cases' are given for every date,
# so for country with lowest can't simply find min(total_cases)
# as we'll get an earlier date with a lower figure
# rather than the country with the lowest final total_cases
# -- however, this is obviously not an issue for the maximum figure
import pyspark.sql.functions as F

filtered_df.select(F.max('total_cases')).show()
filtered_df.groupBy('location').max('total_cases')\
            .select(F.min('max(total_cases)')).show()
```

```
+----------------+
|max(total_cases)|
+----------------+
|         8779653|
+----------------+

+---------------------+
|min(max(total_cases))|
+---------------------+
|                   13|
+---------------------+
```

In [14]:
```python
# to see a list of the countries with the highest and lowest total_cases count...
total_cases = filtered_df.groupBy('location').max('total_cases')
print('Countries with Highest Total Number of Cases')
total_cases.orderBy('max(total_cases)',ascending=False).show()
print('Countries with Lowest Total Number of Cases')
total_cases.orderBy('max(total_cases)',ascending=True).show()
```

```
Countries with Highest Total Number of Cases
+--------------+----------------+
|      location|max(total_cases)|
+--------------+----------------+
| United States|         8779653|
|         India|         7990322|
|        Brazil|         5439641|
|        Russia|         1547774|
|        France|         1198695|
|         Spain|         1116738|
|     Argentina|         1116596|
|      Colombia|         1033218|
|United Kingdom|          917575|
|        Mexico|          901268|
|          Peru|          892497|
|  South Africa|          717851|
|          Iran|          581824|
|         Italy|          564778|
|         Chile|          504525|
|       Germany|          464239|
|          Iraq|          459908|
|    Bangladesh|          401586|
|     Indonesia|          396454|
|   Philippines|          373144|
+--------------+----------------+
only showing top 20 rows

Countries with Lowest Total Number of Cases
+--------------------+----------------+
|            location|max(total_cases)|
+--------------------+----------------+
|          Montserrat|              13|
|                Fiji|              33|
|British Virgin Is...|              72|
|Northern Mariana ...|              92|
| Antigua and Barbuda|             124|
|              Brunei|             148|
|Bonaire Sint Eust...|             150|
|             Bermuda|             194|
|            Barbados|             233|
|      Cayman Islands|             239|
|            Guernsey|             266|
|              Monaco|             320|
|         Isle of Man|             352|
|           Mauritius|             439|
|       Liechtenstein|             483|
|            Tanzania|             509|
|             Comoros|             517|
|              Taiwan|             550|
|             Burundi|             558|
|              Jersey|             560|
+--------------------+----------------+
only showing top 20 rows
```