Data Analytics using Spark SQL

Coronavirus file exist in hdfs as seen below:

```
tdend2@node00:~
[tdend2@node00 ~]$ hdfs dfs -ls /user/tdend2/COVID19/
Found 2 items
-rw-r--r--   3 tdend2 hadoop     149569 2024-09-08 12:00 /user/tdend2/COVID19/coronavirus-text-only-1000.txt
drwxr-xr-x   - tdend2 hadoop          0 2024-09-08 12:17 /user/tdend2/COVID19/output-text
[tdend2@node00 ~]$
```

textFile = spark.read.text("/user/tdend2/COVID19/coronavirus-text-only-1000.txt")

```
tdend2@node00:~
>>> textFile = spark.read.text("/user/tdend2/COVID19/coronavirus-text-only-1000.txt")
>>> textFile.count()
1000
>>>
```

Number of rows in the DataFrame are 1000

The first row:

```
tdend2@node00:~
>>> textFile.first()
Row(value=u'text')
>>>
```

All rows are shown below. It shows the first part of each row.

```
tdend2@node00:~
>>> textFile.show()
+--------------------+
|               value|
+--------------------+
|                text|
|Studies look at t...|
|"RT @EricTopol: T...|
|"RT @NPR: Working...|
|"RT @Harvey_Walke...|
|RT @CNNEE: La far...|
|"RT @ReutersWorld...|
|"RT @CNN: This Il...|
|"RT @Censelio: Ar...|
|RT @jilevin: Trum...|
|RT @propublica: P...|
|NSW to close Vict...|
|RT @ASlavitt: Tru...|
|"RT @ClayTravis: ...|
|RT @JamesGunn: I'...|
|RT @NatashaFatah:...|
|"RT @crissles: Y,...|
|"RT @Censelio: Ar...|
|RT @Villarruel_cl...|
|"RT @JaxAlemany: ...|
+--------------------+
only showing top 20 rows

>>>
```

To see the top 2 rows:

```
tdend2@node00:~

>>> textFile.show(2)
+--------------------+
|               value|
+--------------------+
|                text|
|Studies look at t...|
+--------------------+
only showing top 2 rows

>>>
```

To show all the texts without truncating contents:
textFile.show(10, False)

```
tdend2@node00:~
>>> textFile.show(10, False)
+-----------------------------------------------------------------------------------------------------------------+
|value
+-----------------------------------------------------------------------------------------------------------------+
|text
|Studies look at the potential of natural remedies for treating coronavirus https://t.co/UQMDXZCD
|"RT @EricTopol: These rapid home tests, especially if accurate for transmissibility and cheap, c
|"RT @NPR: Working moms now spend 15 more hours than working dads on childcare and housework, a r
|"RT @Harvey_Walker96: To Al Jazeera,   Malaysia didnt lock these Illegal Immigrants because of "
|RT @CNNEE: La farmacv©utica estadounidense Pfizer y la compav‡v≠a alemana de biotecnologv≠a BioN
|"RT @ReutersWorld: Hundreds of scientists say coronavirus is airborne, ask WHO to revise recomme
|"RT @CNN: This Illinois teen's coronavirus-themed dress, made entirely out of duct tape, feature
|"RT @Censelio: Argentina: Organizaba marchas anticuarentena y murivâ‰¥ por coronavirus | ""Decv≠a
|RT @jilevin: Trump Falsely Claims '99 Percent' of Virus Cases Are 'Totally Harmless' https://t.c
+-----------------------------------------------------------------------------------------------------------------+
only showing top 10 rows
```

To show rows that have a specific keyword "wear masks":
textFile.filter(textFile.value.contains("wear masks")).show(10,False)

```
tdend2@node00:~
>>> textFile.filter(textFile.value.contains("wear masks")).show(10,False)
+-----------------------------------------------------------------------------------------------------------------+
|value
+-----------------------------------------------------------------------------------------------------------------+
|"RT @SkyNews: #Coronavirus: Sky's @AlexCrawfordSky visits Texas, where some people refuse to wea
+-----------------------------------------------------------------------------------------------------------------+

>>>
```

close the Spark shell: exit

df = spark.read.option("header","true").csv("/user/data/CSC534BDA/COVID19/COVID19-worldwide.csv")

Spark imports all data as a string type, To see the schema in a tree format, type printSchema()

```
tdend2@node00:~
>>> df = spark.read.option("header","true").csv("/user/data/CSC534BDA/COVID19/COVID19-worldwide.c
>>> df.printSchema()
root
 |-- dateRep: string (nullable = true)
 |-- day: string (nullable = true)
 |-- month: string (nullable = true)
 |-- year: string (nullable = true)
 |-- cases: string (nullable = true)
 |-- deaths: string (nullable = true)
 |-- countriesAndTerritories: string (nullable = true)
 |-- geoId: string (nullable = true)
 |-- countryterritoryCode: string (nullable = true)
 |-- popData2019: string (nullable = true)
 |-- continentExp: string (nullable = true)
 |-- Cumulative_number_for_14_days_of_COVID-19_cases_per_100000: string (nullable = true)

>>>
```

We see the rows stored in the DataFrame by typing a show() action:

```
tdend2@node00:~
>>> df.show()
+--------+---+-----+----+-----+------+----------------------+-----+--------------------+--------
| dateRep|day|month|year|cases|deaths|countriesAndTerritories|geoId|countryterritoryCode|popData2
+--------+---+-----+----+-----+------+----------------------+-----+--------------------+--------
|10/22/20| 22|   10|2020|  135|     2|           Afghanistan|   AF|                 AFG|   38041
|10/21/20| 21|   10|2020|   88|     2|           Afghanistan|   AF|                 AFG|   38041
|10/20/20| 20|   10|2020|   87|     5|           Afghanistan|   AF|                 AFG|   38041
|10/19/20| 19|   10|2020|   59|     4|           Afghanistan|   AF|                 AFG|   38041
|10/18/20| 18|   10|2020|   68|     3|           Afghanistan|   AF|                 AFG|   38041
|10/17/20| 17|   10|2020|   47|     4|           Afghanistan|   AF|                 AFG|   38041
|10/16/20| 16|   10|2020|    0|     0|           Afghanistan|   AF|                 AFG|   38041
|10/15/20| 15|   10|2020|   32|     1|           Afghanistan|   AF|                 AFG|   38041
|10/14/20| 14|   10|2020|   66|     0|           Afghanistan|   AF|                 AFG|   38041
|10/13/20| 13|   10|2020|  129|     3|           Afghanistan|   AF|                 AFG|   38041
|10/12/20| 12|   10|2020|   96|     4|           Afghanistan|   AF|                 AFG|   38041
|10/11/20| 11|   10|2020|    0|     0|           Afghanistan|   AF|                 AFG|   38041
|10/10/20| 10|   10|2020|   10|     1|           Afghanistan|   AF|                 AFG|   38041
| 10/9/20|  9|   10|2020|   77|     2|           Afghanistan|   AF|                 AFG|   38041
| 10/8/20|  8|   10|2020|   68|     1|           Afghanistan|   AF|                 AFG|   38041
| 10/7/20|  7|   10|2020|   62|     2|           Afghanistan|   AF|                 AFG|   38041
| 10/6/20|  6|   10|2020|  145|     5|           Afghanistan|   AF|                 AFG|   38041
| 10/5/20|  5|   10|2020|   44|     0|           Afghanistan|   AF|                 AFG|   38041
| 10/4/20|  4|   10|2020|    7|     4|           Afghanistan|   AF|                 AFG|   38041
| 10/3/20|  3|   10|2020|    5|     0|           Afghanistan|   AF|                 AFG|   38041
+--------+---+-----+----+-----+------+----------------------+-----+--------------------+--------
only showing top 20 rows

>>>
```

To see the daily cases/deaths number of countries in the world,
let's select the "dateRep", "cases", "deaths", and "countriesAndTerritories" columns.

df.select("dateRep", "cases", "deaths", "countriesAndTerritories").show()

```
tdend2@node00:~
>>> df.select("dateRep", "cases", "deaths", "countriesAndTerritories").show()
+--------+-----+------+----------------------+
| dateRep|cases|deaths|countriesAndTerritories|
+--------+-----+------+----------------------+
|10/22/20|  135|     2|           Afghanistan|
|10/21/20|   88|     2|           Afghanistan|
|10/20/20|   87|     5|           Afghanistan|
|10/19/20|   59|     4|           Afghanistan|
|10/18/20|   68|     3|           Afghanistan|
|10/17/20|   47|     4|           Afghanistan|
|10/16/20|    0|     0|           Afghanistan|
|10/15/20|   32|     1|           Afghanistan|
|10/14/20|   66|     0|           Afghanistan|
|10/13/20|  129|     3|           Afghanistan|
|10/12/20|   96|     4|           Afghanistan|
|10/11/20|    0|     0|           Afghanistan|
|10/10/20|   10|     1|           Afghanistan|
| 10/9/20|   77|     2|           Afghanistan|
| 10/8/20|   68|     1|           Afghanistan|
| 10/7/20|   62|     2|           Afghanistan|
| 10/6/20|  145|     5|           Afghanistan|
| 10/5/20|   44|     0|           Afghanistan|
| 10/4/20|    7|     4|           Afghanistan|
| 10/3/20|    5|     0|           Afghanistan|
+--------+-----+------+----------------------+
only showing top 20 rows

>>>
```

Used ‘filter’ to choose a specific country like the one below:
df.select("dateRep","cases","deaths","countriesAndTerritories").filter("countryterritoryCode == 'USA'").show()

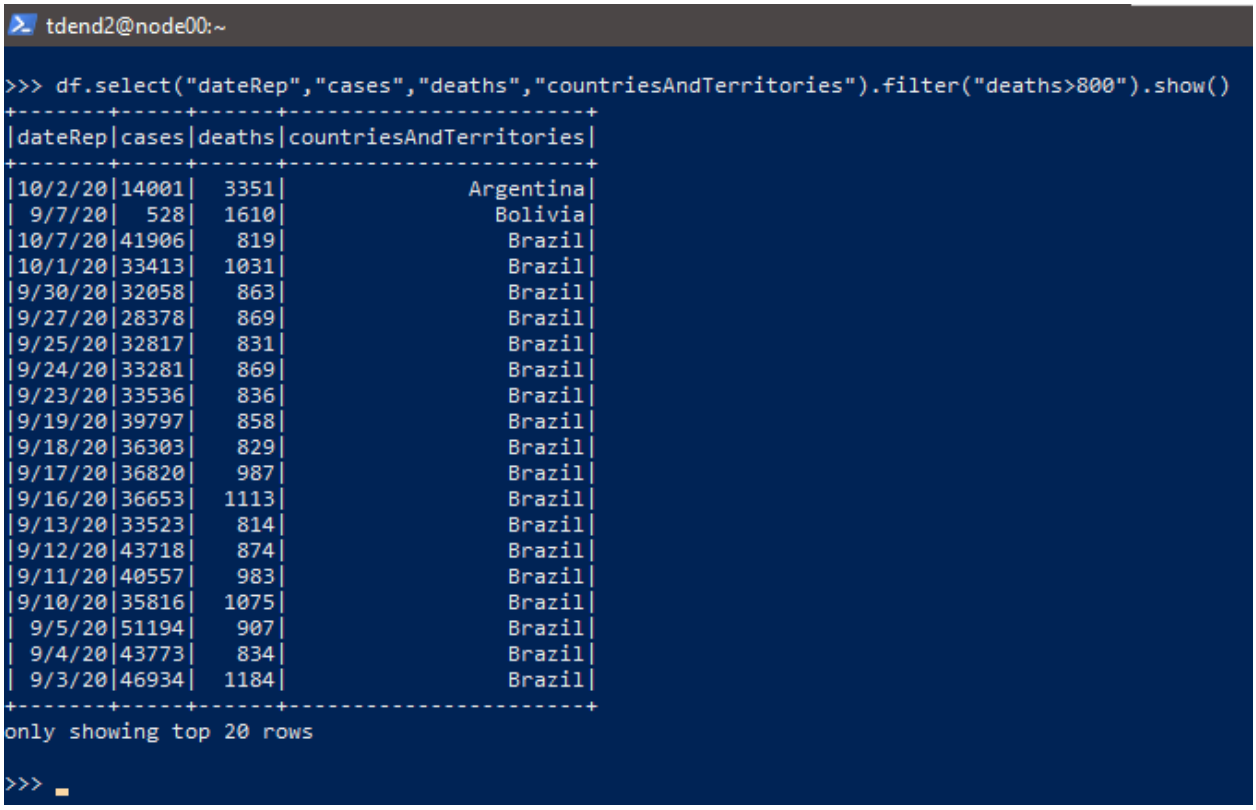
```
tdend2@node00:~
>>> df.select("dateRep","cases","deaths","countriesAndTerritories").filter("countryterritoryCode == 'USA'").show()
+--------+-----+------+----------------------+
| dateRep|cases|deaths|countriesAndTerritories|
+--------+-----+------+----------------------+
|10/22/20|62978|  1135|   United_States_of_...|
|10/21/20|58549|   933|   United_States_of_...|
|10/20/20|60160|   459|   United_States_of_...|
|10/19/20|47843|   385|   United_States_of_...|
|10/18/20|56611|   690|   United_States_of_...|
|10/17/20|70256|   899|   United_States_of_...|
|10/16/20|63785|   828|   United_States_of_...|
|10/15/20|59386|   970|   United_States_of_...|
|10/14/20|52517|   817|   United_States_of_...|
|10/13/20|41653|   314|   United_States_of_...|
|10/12/20|43597|   394|   United_States_of_...|
|10/11/20|54271|   590|   United_States_of_...|
|10/10/20|58082|  1014|   United_States_of_...|
| 10/9/20|56800|   972|   United_States_of_...|
| 10/8/20|48182|   892|   United_States_of_...|
| 10/7/20|43062|   717|   United_States_of_...|
| 10/6/20|40705|   398|   United_States_of_...|
| 10/5/20|34901|   400|   United_States_of_...|
| 10/4/20|50659|   678|   United_States_of_...|
| 10/3/20|54471|   908|   United_States_of_...|
+--------+-----+------+----------------------+
only showing top 20 rows

>>>
```

Write and run a Spark command (not SQL query) to show the date when # of deaths was severe
(more than 800 deaths), as well as # of confirmed cases, # of deaths, and country using the filter
function. The output should be like the one below.

```
+--------+-----+------+----------------------+
| dateRep|cases|deaths|countriesAndTerritories|
+--------+-----+------+----------------------+
```

df.select("dateRep","cases","deaths","countriesAndTerritories").filter("deaths>800").show()



Query for all countries but show U.S.A. data only for display purposes:
df.select("dateRep","cases","deaths","countriesAndTerritories").filter("deaths>800").filter("countriesAndTerritories == 'United_States_of_America'").show()

```
tdend2@node00:~
>>> df.select("dateRep","cases","deaths","countriesAndTerritories").filter("deaths>800").filter("countriesAndTerritories == 'United_States_of_America'").show()
+--------+-----+------+----------------------+
| dateRep|cases|deaths|countriesAndTerritories|
+--------+-----+------+----------------------+
|10/22/20|62978|  1135|   United_States_of_...|
|10/21/20|58549|   933|   United_States_of_...|
|10/17/20|70256|   899|   United_States_of_...|
|10/16/20|63785|   828|   United_States_of_...|
|10/15/20|59386|   970|   United_States_of_...|
|10/14/20|52517|   817|   United_States_of_...|
|10/10/20|58082|  1014|   United_States_of_...|
| 10/9/20|56800|   972|   United_States_of_...|
| 10/8/20|48182|   892|   United_States_of_...|
| 10/3/20|54471|   908|   United_States_of_...|
| 10/2/20|44771|   880|   United_States_of_...|
| 10/1/20|41982|   930|   United_States_of_...|
| 9/30/20|43017|   928|   United_States_of_...|
| 9/26/20|55013|   964|   United_States_of_...|
| 9/25/20|44213|   901|   United_States_of_...|
| 9/24/20|37930|  1102|   United_States_of_...|
| 9/23/20|38307|   926|   United_States_of_...|
| 9/19/20|50209|   956|   United_States_of_...|
| 9/18/20|43567|   831|   United_States_of_...|
| 9/17/20|24598|   865|   United_States_of_...|
+--------+-----+------+----------------------+
only showing top 20 rows

>>>
```

By default, every column in a CSV file is treated as a string type. However, we have some numerical columns, e.g. cases, deaths, and Cumulative_number_for_14_days_of_COVID19_cases_per_100000.

Spark's CSV reader provides the functionality for us via options that we can set on the reader API.

```
tdend2@node00:~
>>> df2 = spark.read.option("header","true").option("inferSchema","true").csv("/user/data/CSC534BDA/COVID19/COVID19-worldwide.csv")
```

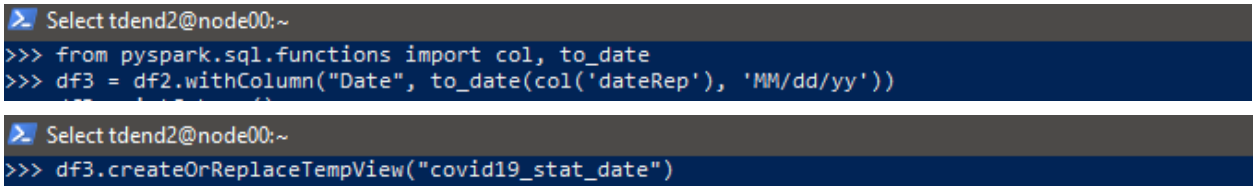
To see the inferred type for each column, we can print the schema of the parsed DataFrame below.

```
Select tdend2@node00:~
>>> df2.printSchema()
root
 |-- dateRep: string (nullable = true)
 |-- day: integer (nullable = true)
 |-- month: integer (nullable = true)
 |-- year: integer (nullable = true)
 |-- cases: integer (nullable = true)
 |-- deaths: integer (nullable = true)
 |-- countriesAndTerritories: string (nullable = true)
 |-- geoId: string (nullable = true)
 |-- countryterritoryCode: string (nullable = true)
 |-- popData2019: integer (nullable = true)
 |-- continentExp: string (nullable = true)
 |-- Cumulative_number_for_14_days_of_COVID-19_cases_per_100000: double (nullable = true)
```

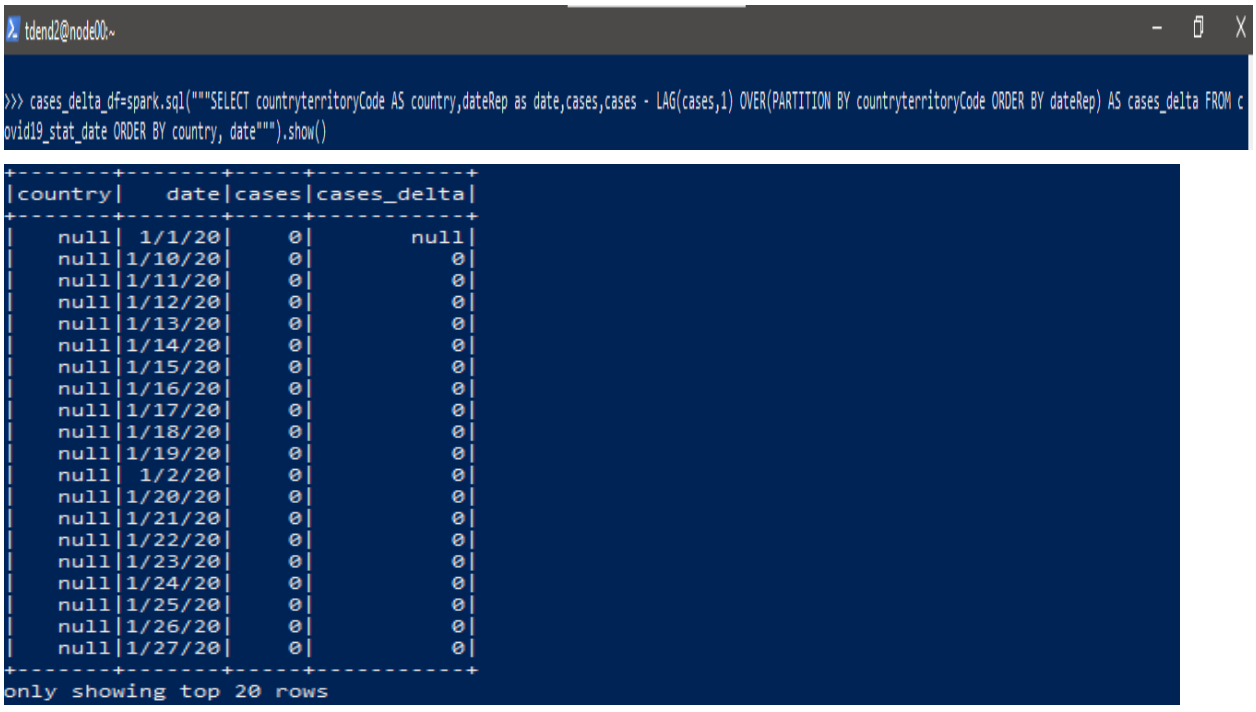
Write and run a Spark SQL query, e.g., spark.sql("""..."""), or Spark command to calculate the
delta (the changes) of cases from the previous day. The output should be like the one below.

+-------+----------+-----+-----------+
|country| date|cases|cases_delta|
+-------+----------+-----+-----------+

Note: Country is an alias of the countryterritoryCode column
Note2: Write commands/queries for all countries but show U.S.A. data only for display purposes
using filter.
Note3: No need to include all output data in the screenshots, just the first five lines (from 12/31/19) and the last five lines (until 10/22/20) of the output data.
Note4: Order by date (ascending) and double-check if your delta is calculated correctly.



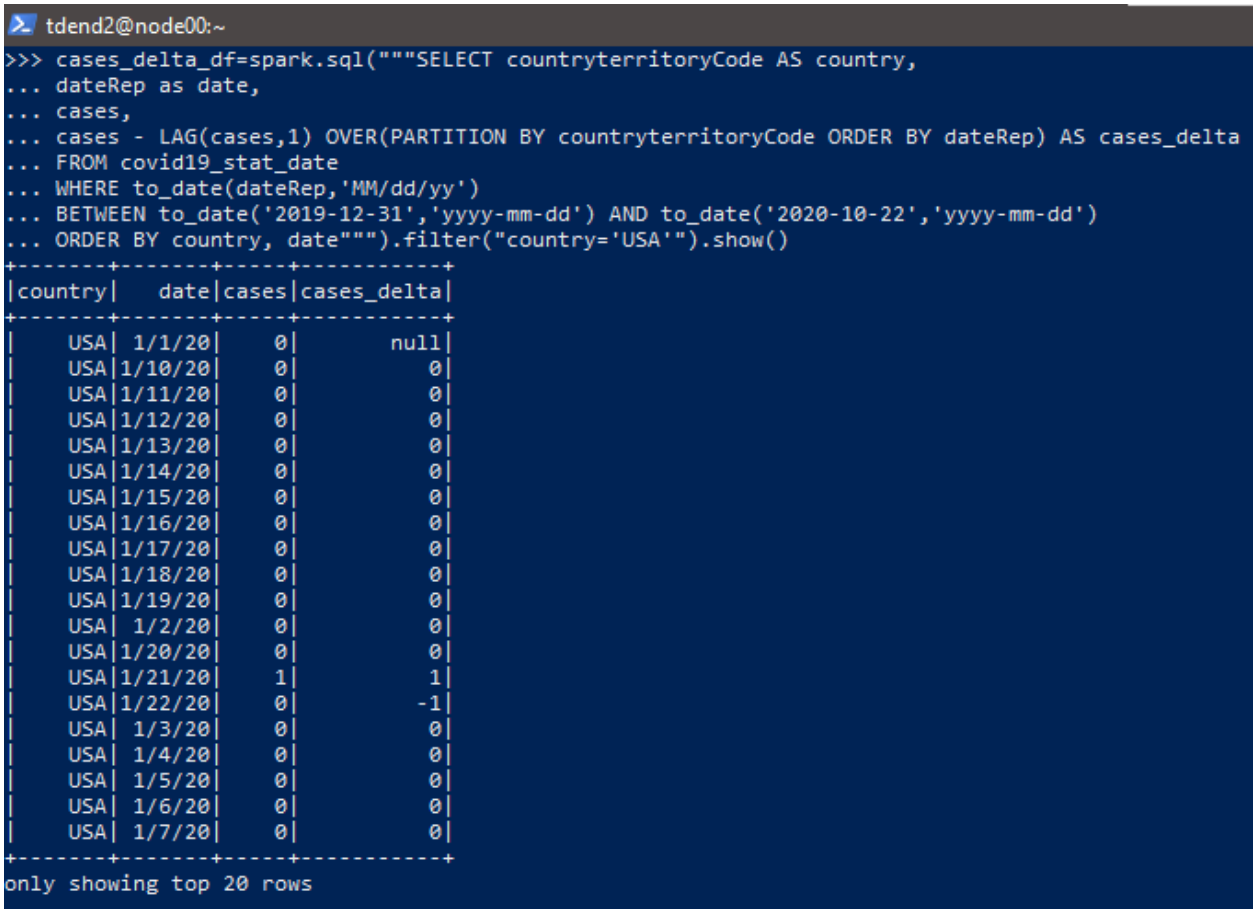
cases_delta_df=spark.sql("""SELECT countryterritoryCode AS country,dateRep as date,cases,cases - LAG(cases,1) OVER(PARTITION BY countryterritoryCode ORDER BY dateRep) AS cases_delta FROM covid19_stat_date ORDER BY country, date""").show()



The above output included all the countries data.

The below output is dated between 12/31/19 and 10/22/20 and filtered USA data.

cases_delta_df=spark.sql("""SELECT countryterritoryCode AS country,dateRep as date,cases,cases - LAG(cases,1) OVER(PARTITION BY countryterritoryCode ORDER BY dateRep) AS cases_delta FROM covid19_stat_date WHERE to_date(dateRep,'MM/dd/yy') BETWEEN to_date('2019-12-31','yyyy-mm-dd') AND to_date('2020-10-22','yyyy-mm-dd') ORDER BY country, date""").filter("country='USA'").show()

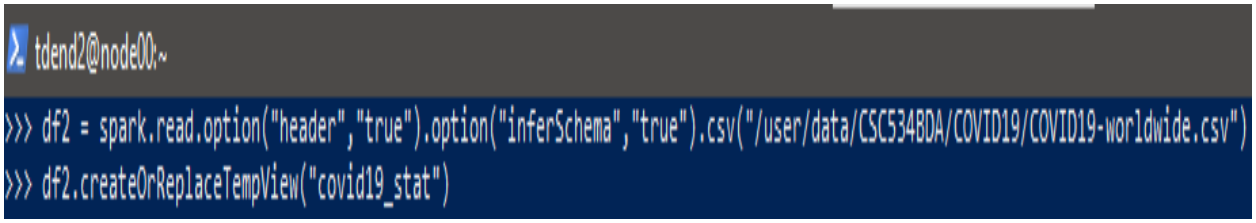


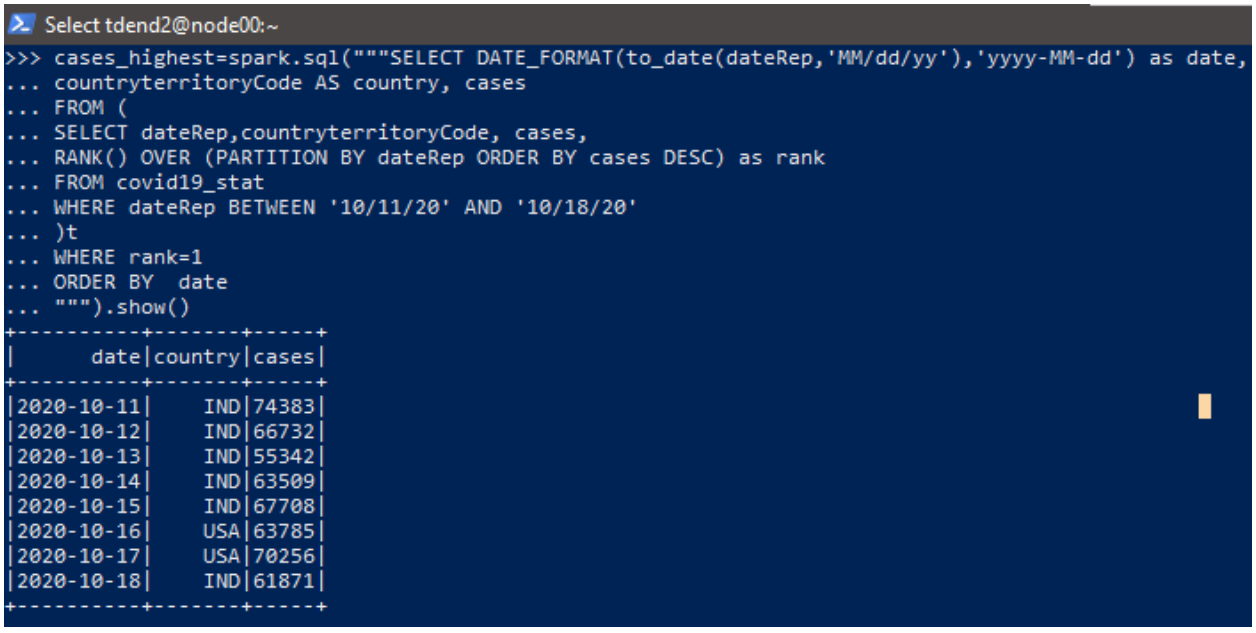
Write and run a Spark SQL query, e.g., spark.sql("""...""") or Spark command to find the countries with the highest number of confirmed cases each day among all countries during the
period from Oct. 11 to Oct. 18, 2020, using 'Rank() OVER(...)'. The output should be like the
one below.

+----------+---------+-----+
|date | country|cases|
+----------+---------+-----+
|2020-10-11| | |

+----------+---------+-----+
|2020-10-12| | |
+----------+---------+-----+
...
Note: Country is an alias of the countryterritoryCode column

```
cases_highest=spark.sql("""SELECT
DATE_FORMAT(to_date(dateRep,'MM/dd/yy'),'yyyy-MM-dd') as date,
countryterritoryCode AS country, cases
FROM (
SELECT dateRep,countryterritoryCode, cases,
RANK() OVER (PARTITION BY dateRep ORDER BY cases DESC) as rank
FROM covid19_stat
WHERE dateRep BETWEEN '10/11/20' AND '10/18/20'
)t
WHERE rank=1
ORDER BY  date
""").show()
```





++++++++++++++++++++++++++++++++THE END++++++++++++++++++++++++++++++++++++