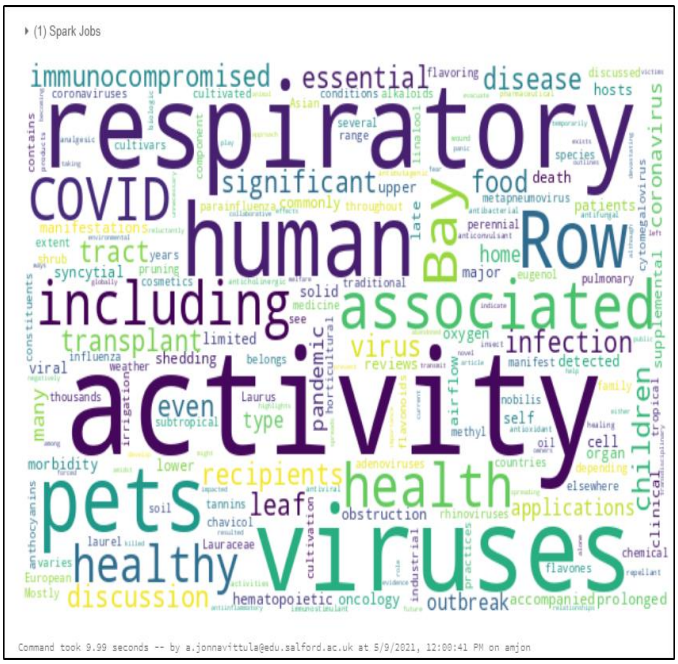


ANALYSIS OF MEDICAL DATA

ABSTRACT

Since the COVID-19 pandemic, there have been many medical journals. This report elucidates the analysis performed on these journals in relation to the widespread.

UNIVERSITY OF SALFORD | Ambareesh Jonnavittula

Big Data Tools & Techniques

Section 1: Setup

1.1 System Configuration

S.No	Environment	Configuration / Version
1	Databricks	Databricks Runtime version 8.1 (Apache Spark 3.1.1, Scala 2.12)
2	Operating System	Ubuntu 18.04.5 LTS
3	Python	Python 3.8.6
4	HiveQL	Apache Spark SQL 3.0 (compatible with Apache Hive)
5	AWS S3	Based on AWS CLI 2
6	AWS Athena	Athena engine version 2 (Presto 0.172)
7	AWS EMR	EMR release 6.0.0 (HBase: 1.4.13, Hadoop 2.10.1, Hive 2.3.7)

1.2 Source Data

- https://ai2-semanticscholar-cord-19.s3-us-west-2.amazonaws.com/2020-07-01/metadata.csv
- https://raw.githubusercontent.com/ambareesh-j/SJR/main/scimagojr%202019.csv

1.3 Initial Prompts

- The keyword 'OVERWRITE' wherever used is to enable reusability.
- Sorting has been done in descending order for all the tasks except for the task 6.

Section 2: Data Cleaning and Pre-processing

2.1 Outline

- **In PySpark implementation** (Dataframes and RDD) – inclusive of exception handling
  i. Libraries are installed – bokeh, seaborn, wordcloud.
  ii. Widgets are created to enable user interaction with the version of the datasets.
  iii. Variables are created using inputs from the widgets given by a user.
  iv. Quality check for website domain presence is performed.
  v. Underlying implementation is validated.
  vi. Datasets are downloaded and copied into DBFS file store to enable utility.
  vii. Availability of the datasets and the underlying data is validated.
  viii. Dataframes for CORD19 dataset are created using
    - (1) .csv file, (2) temporary view, and (3) parquet files (with partition).
  ix. A dataframe for Scimago Journal Rankings (SJR) dataset is created using .csv file

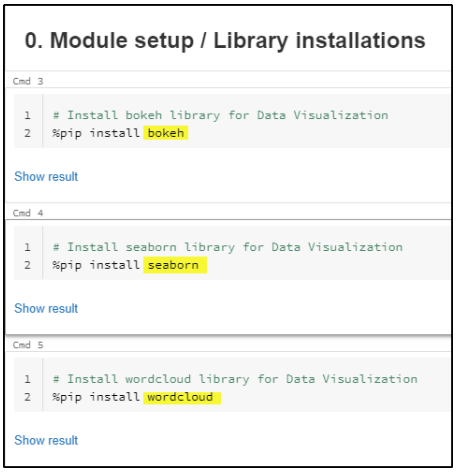
- **In HiveQL implementation**
  i. Hive Input Variables are set to enable version alteration and user interaction.
  ii. Additional hive variables are created from user inputs – database, tables, file path.
  iii. A database is created.
  iv. Hive data structures for CORD19 dataset are created using
    - (1) .csv file to table, (2) temporary view, and (3) parquet files to table
  v. Hive table for SJR dataset is created using .csv file
  vi. The tables / views created are validated

- **In AWS**,
  i. In AWS S3, buckets are created for each of input, output and miscellaneous files.
  ii. Cord-19 and SJR .csv files are manually imported into the input buckets.
  iii. In AWS Athena, a database is first created.
  iv. Tables and views are later created within the database in AWS Athena, and queried.
  v. SQL scripts are converted to .txt format and stored in S3 miscellaneous bucket.
  vi. In AWS EMR, a cluster is created and started. The cluster then enters 'Waiting' state.
  vii. Steps are added & run based on the I/O paths, where input path is where the .txt files are present, and Output path is simply a folder inside / or an S3 bucket.

2.2 PySpark DF & RDD

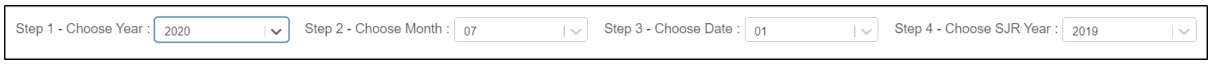
2.2.1 Setting the stage

- Using %pip install command, we install the necessary libraries.



- Using Databricks widgets utilities, dropdown widgets were setup to pick the year, month and date of a month along with SJR_Year using ranged list comprehensions.

```
# Choose Year
dbutils.widgets.dropdown("Step 1 - Choose Year", "2020", [str(x) for x in range(2019, 2026)])

# Choose Month
dbutils.widgets.dropdown("Step 2 - Choose Month", "07", [str(x).zfill(2) for x in range(1, 13)])

# Choose Date
dbutils.widgets.dropdown("Step 3 - Choose Date", "01", [str(x).zfill(2) for x in range(1, 32)])

# Choose ScimagoJr Year
dbutils.widgets.dropdown("Step 4 - Choose SJR Year", "2019", [str(x) for x in range(2019, 2021)])
```



- Later, widget dropdown variables are used to create python variables. Source paths will be part of the input variables besides others.

```
#Main FilePath
filepath = "dbfs:/FileStore/tables/CORD19/"

###################################
# CORD-19 dataset
###################################

# Fragments of the URL from where we will download the dataset from:
url_frag1 = "https://ai2-semanticscholar-cord-19.s3-us-west-2.amazonaws.com/"
url_frag2 = dbutils.widgets.get("Step 1 - Choose Year") + "-" +
dbutils.widgets.get("Step 2 - Choose Month") + "-" + dbutils.widgets.get("Step 3 -
Choose Date")
url_frag3 = "/metadata.csv"

# Combined variable using all the fragments of the URL
metadata_url = url_frag1 + url_frag2 + url_frag3

# Proper filename given to store the data
md_filename = str("metadata_" + url_frag2 + ".csv").replace("-","_")

# Display status :
print("Dataset name : CORD-19")
print("Dataset type : Web-based '.CSV' file (URL)")
print("URL location :", str(metadata_url).strip(), "\n")
```

```
24  ###################################
25  # Scimago Journal Rankings dataset
26  ###################################
27
28  # Instead of manual Import, we uploaded the file into GitHub to use the URL here
    directly
29  # Data Source - https://www.scimagojr.com/journalrank.php?page=1&total_size=30891
30  # GitHub source - https://raw.githubusercontent.com/ambareesh-
    j/SJR/main/scimagojr%202019.csv
31
32  #Fragments of the location of the downloaded/imported file of SJR URL:
33  sjr_frag1 = "https://raw.githubusercontent.com/ambareesh-j/SJR/main/scimagojr%20"
34  sjr_frag2 = dbutils.widgets.get("Step 4 - Choose SJR Year") + ".csv"
35
36  # Combined variable using all the fragments of the file location of SJR URL
37  scimago_loc = sjr_frag1 + sjr_frag2
38
39  # Proper filename given to store the data
40  sjr_filename = str("scimagojr_" + sjr_frag2)
41
42  # Extra variables (during visualization)
43  scimago_year = dbutils.widgets.get("Step 4 - Choose SJR Year")
44
45  # Display status :
46  print("Dataset name : Scimago Journal Rankings")
47  print("Dataset type : Web-based '.CSV' file (URL)")
48  print("URL location: ", str(scimago_loc).strip())
```

2.2.2 Initial Validation

- 'Requests' library is used to validate a website. If the status code equals 200, the website exists.

1.3.1 CORD19 dataset - Quality check

Cmd 13

```
1  import requests
2  request = requests.get(metadata_url)
3  if request.status_code == 200:
4      print('Validation complete, Website exists : ', metadata_url )
5  else:
6      raise NameError('Validation failed: Website does not exist (or) Input Year/Month/Date is invalid')
```

```
Validation complete, Website exists :  https://ai2-semanticscholar-cord-19.s3-us-west-2.amazonaws.com/2020-07-01/metadata.csv
```

Command took 5.61 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/9/2021, 12:00:39 PM on amjon

1.3.2 ScimagoJr dataset - Quality check

Cmd 15

```
1  import requests
2  request = requests.get(scimago_loc)
3  if request.status_code == 200:
4      print('Validation complete, Website exists : ', scimago_loc )
5  else:
6      raise NameError('Validation failed: Website does not exist (or) Input Year ScimagoJr is invalid')
```

```
Validation complete, Website exists :  https://raw.githubusercontent.com/ambareesh-j/SJR/main/scimagojr%202019.csv
```

Command took 1.12 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/9/2021, 12:00:39 PM on amjon

- Python 'imp' library is used to find modules.

2.1 Validate the Underlying implementation

Cmd 18

```
1  # Validate the underlying implementation using imp module in Python
2  import imp
3
4  try:
5      imp.find_module('dbutils')
6  except ImportError:
7      import pyspark
8      sc = pyspark.SparkContext()
9
10 print("Validation complete : Underlying implementation using Databricks")
```

```
<command-858541245727410>:2: DeprecationWarning: the imp module is deprecated in favour of importl
ib; see the module's documentation for alternative uses
  import imp
Validation complete : Underlying implementation using Databricks
```

Command took 0.02 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/9/2021, 12:00:39 PM on amjon

2.2.3 Data Preparation

- Environment variables are set using 'OS' library.

2.2.1 Setup environmental variables to work using Shell script

Cmd 21

```
import os

os.environ['fileurl'] = metadata_url
os.environ['fileroot'] = md_filename

os.environ['fileurl2'] = scimago_loc
os.environ['fileroot2'] = sjr_filename
```

- Using shell script, an existing file is removed within the local file:/tmp/ directory.
- 'curl' command-line utility is used to eliminate the redundancy of manually downloading and importing the file.

2.2.2 Download the file from the URL using path variables and store it in File:/tmp/ folder

Cmd 23

```
%sh

rm /tmp/$fileroot
curl $fileurl > /tmp/$fileroot
```

```
rm: cannot remove '/tmp/metadata_2020_07_01.csv': No such file or directory
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  5  231M    5 12.2M    0     0  45.3M      0  0:00:05 --:--:--  0:00:05 45.3M
 34  231M   34 80.0M    0     0  60.5M      0  0:00:03  0:00:01  0:00:02 60.5M
 72  231M   72  167M    0     0  73.8M      0  0:00:03  0:00:02  0:00:01 73.8M
100  231M  100  231M    0     0  71.9M      0  0:00:03  0:00:03 --:--:-- 72.0M
```

Command took 3.33 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/9/2021, 12:00:39 PM on amjon

```
%sh

rm /tmp/$fileroot2
curl $fileurl2 > /tmp/$fileroot2
```

```
rm: cannot remove '/tmp/scimagojr_2019.csv': No such file or directory
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
 46 7979k   46 3675k    0     0  6315k      0  0:00:01 --:--:--  0:00:01 6305k
100 7979k  100 7979k    0     0  11.0M      0 --:--:-- --:--:-- --:--:-- 11.0M
```

Command took 0.77 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/7/2021, 10:45:15 PM on amjon

- In order to enable utility, we copy these files and store in DBFS.

```
# Copy the files to DBFS:/
dbutils.fs.cp("file:/tmp/"+md_filename, filepath+md_filename)
dbutils.fs.cp("file:/tmp/"+sjr_filename, filepath+sjr_filename)

# Find the files in DBFS:/ path
dbutils.fs.ls(filepath)
```

```
Out[9]: [FileInfo(path='dbfs:/FileStore/tables/CORD19/CountryKeys/', name='CountryKeys/', size=0),
 FileInfo(path='dbfs:/FileStore/tables/CORD19/metadata_2020_03_27.csv', name='metadata_2020_03_27.csv', size=69689204),
 FileInfo(path='dbfs:/FileStore/tables/CORD19/metadata_2020_05_01.csv', name='metadata_2020_05_01.csv', size=89290114),
 FileInfo(path='dbfs:/FileStore/tables/CORD19/metadata_2020_07_01.csv', name='metadata_2020_07_01.csv', size=242867644),
 FileInfo(path='dbfs:/FileStore/tables/CORD19/parq/', name='parq/', size=0),
 FileInfo(path='dbfs:/FileStore/tables/CORD19/prd', name='prd', size=242867644),
 FileInfo(path='dbfs:/FileStore/tables/CORD19/scimagojr_2019.csv', name='scimagojr_2019.csv', size=8170647)]
```

Command took 13.84 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/7/2021, 10:45:15 PM on amjon

2.2.4 Quality check

- Using python, a custom function to throw a FileNotFoundException is created using exception handling techniques.

```
# Check if this file is present in /FileStore/tables/CORD19/

# Custom function to see if the file exists in a given path
def file_exists(path):
    try:
        dbutils.fs.ls(path)
        return True
    except Exception as e:
        if 'java.io.FileNotFoundException' in str(e):
            return False
        else:
            raise

```

- 'assert' is used to check whether the file exists within the try block and an ImportError is thrown if an exception is encountered.

```
####################
# CORD19 Dataset
####################
try:
    imp.find_module('dbutils')
    assert file_exists(filepath+md_filename) == True, "ERROR - The .CSV file was not correctly uploaded for CORD19 metadata - " + md_filename + ", please check again and upload into /FileStore/tables/CORD19/"
except ImportError:
    print("Are you running this on Databricks?")

print("Validation complete, File exists in Databricks :", filepath+md_filename)

####################
# ScimagoJr Dataset
####################
try:
    imp.find_module('dbutils')
    assert file_exists(filepath+sjr_filename) == True, "ERROR - The .CSV file was not correctly uploaded for Scimago Journal Rankings - " + sjr_filename + ", please check again and upload into /FileStore/tables/CORD19/"
except ImportError:
    print("Are you running this on Databricks?")

print("Validation complete, File exists in Databricks :", filepath+sjr_filename)
```

- Before the transformation begins, both the datasets are checked for data availability within.

```
# Temporary RDD created
tmp_cord_rdd = sc.textFile(filepath+md_filename)

# Assert if the number of records present are at least 2 - a header and a line item
assert tmp_cord_rdd.count() >= 2, "Validation failed: Missing data in the uploaded file, please recheck the URL used"

# Validation status - success message
print("Validation complete: There is at least 1 record in the RDD, so expecting successful transformation for the file :", "\n", filepath+md_filename)

# Delete the temporary RDD to conserve memory
del tmp_cord_rdd
```

```
▸ (1) Spark Jobs
Validation complete: There is at least 1 record in the RDD, so expecting successful transformation for the file :
 dbfs:/FileStore/tables/CORD19/metadata_2020_07_01.csv
Command took 12.05 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/7/2021, 10:45:15 PM on amjon
```

2.4.2 ScimagoJr - Underlying data validation

```
Cmd 33

# Temporary RDD created
tmp_scimago_rdd = sc.textFile(filepath+sjr_filename)

# Validate if the number of records present are at least 2 - a header and a line item
assert tmp_scimago_rdd.count() >= 2, "Validation failed: Missing data in the uploaded file, please recheck the URL used"

# Validation status - success message
print("Validation complete: There is at least 1 record in the RDD, so expecting successful transformation for the file :", "\n", filepath+sjr_filename)

# Delete the temporary RDD to conserve memory
del tmp_scimago_rdd
```

```
▸ (1) Spark Jobs
Validation complete: There is at least 1 record in the RDD, so expecting successful transformation for the file :
 dbfs:/FileStore/tables/CORD19/scimagojr_2019.csv
Command took 0.85 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/9/2021, 12:00:39 PM on amjon
```

2.2.5 Dataframes creation

- Spark read options like 'quote' and 'escape' are utilized ("\"") and the delimiter is going to be a comma, since we are using a .csv file.
- A schema is dynamically inferred for the covid19 dataset because the columns may vary in the future. This observation is in accordance with the test datasets given.

```
# File location and type
file_location = filepath+md_filename
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df_cord = spark.read.format(file_type) \
               .option("inferSchema", infer_schema) \
               .option("header", first_row_is_header) \
               .option("quote", "\"") \
               .option("escape", "\"") \
               .option("sep", delimiter) \
               .load(file_location)

# Dataframe df_cord displayed
df_cord.printSchema()
display(df_cord)
```

- Whereas, for ScimagoJr dataset, we create a schema assuming the metadata of the file wouldn't alter in the future. Here the delimiter is a semicolon.

```
from pyspark.sql.types import IntegerType, StringType, StructType, StructField

#Creating a Schema
scimagoSchema = StructType([
                          StructField("Rank", IntegerType(), False),
                          StructField("Sourceid", StringType(), True),
                          StructField("Title", StringType(), True),
                          StructField("Type", StringType(), True),
                          StructField("Issn", StringType(), True),
                          StructField("SJR", IntegerType(), True),
                          StructField("SJR_Best_Quartile", StringType(), True),
                          StructField("H_index", IntegerType(), True),
                          StructField("Total_Docs_"+scimago_year, IntegerType(), True ),
                          StructField("Total_Docs_3years", IntegerType(), True),
                          StructField("Total_Refs", IntegerType(), True ),
                          StructField("Total_Cites_3years", IntegerType(), True ),
                          StructField("Citable_Docs_3years", IntegerType(), True),
                          StructField("Cites/Doc_2years", IntegerType(), True ),
                          StructField("Ref/Doc", StringType(), True),
                          StructField("Country", StringType(), True),
                          StructField("Region", StringType(), True ),
                          StructField("Publisher", StringType(), True),
                          StructField("Coverage", StringType(), True),
                          StructField("Categories", StringType(), True),
                            ])

# File location and type
file_location = filepath+sjr_filename
file_type = "csv"

```

```
# CSV options
infer_schema = "false"
first_row_is_header = "true"
delimiter = ";"

# The applied options are for CSV files. For other file types, these will be ignored.
df_scimagojr = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .schema(scimagoSchema) \
  .load(file_location)

#   .option("schema", scimagoSchema) \

df_scimagojr.printSchema()
display(df_scimagojr)
```

- The implementation is common until here for both Dataframes & RDD.
- In parallel, a temporary view is created which is session-scoped only.

- In addition, a parquet file is created and partitioned using the dataframe. This is to enable the power of chunking and predicate push-down which allows parquet to skip over records that don't match the filter criteria. This partition, however, creates performance issues.
- **df_cord.columns* unpacks all the list of the columns from the selected dataframe, in addition to the custom column – Published_Year
- Substring function is used to take the first 4 characters in the publish_time column to create Published_Year and the parquet file is partitioned using the same.

- Later these parquet files are read back into a new dataframe.

```
df_parq = spark.read.parquet("dbfs:/FileStore/tables/CORD19/parq/t_cord19.parquet")

display(df_parq)
```

(3) Spark Jobs

df_parq: pyspark.sql.dataframe.DataFrame = [cord_uid: string, sha: string ... 18 more fields]

2.2.6 RDD creation

- RDD variables are created from the above created dataframes by using .rdd() method.

2.5.1.2 CORD19 - Create RDD from Dataframe --> cord19_rdd

Cmd 37

```
# RDD Creation from the DF.rdd method
cord19_rdd = df_cord.rdd
cord19_rdd.take(10)
```

(1) Spark Jobs

```
Out[14]: [Row(cord_uid='ug7v899j', sha='d1aafb70c066a2068b02786f8929fd9c900897fb', source_x='PMC',
title='Clinical features of culture-proven Mycoplasma pneumoniae infections at King Abdulaziz Univ
ersity Hospital, Jeddah, Saudi Arabia', doi='10.1186/1471-2334-1-6', pmcid='PMC35282', pubmed_id
='11472636', license='no-cc', abstract='OBJECTIVE: This retrospective chart review describes the e
pidemiology and clinical features of 40 patients with culture-proven Mycoplasma pneumoniae infecti
ons at King Abdulaziz University Hospital, Jeddah, Saudi Arabia. METHODS: Patients with positive
M. pneumoniae cultures from respiratory specimens from January 1997 through December 1998 were ide
ntified through the Microbiology records. Charts of patients were reviewed. RESULTS: 40 patients w
ere identified, 33 (82.5%) of whom required admission. Most infections (92.5%) were community-acqu
ired. The infection affected all age groups but was most common in infants (32.5%) and pre-school
children (22.5%). It occurred year-round but was most common in the fall (35%) and spring (30%). M
ore than three-quarters of patients (77.5%) had comorbidities. Twenty-four isolates (60%) were ass
ociated with pneumonia, 14 (35%) with upper respiratory tract infections, and 2 (5%) with bronchio
```

2.5.2.2 ScimagoJr - Create RDD from Dataframe --> scimagojr_rdd

Cmd 42

```
scimagojr_rdd = df_scimagojr.rdd
scimagojr_rdd.take(10)
```

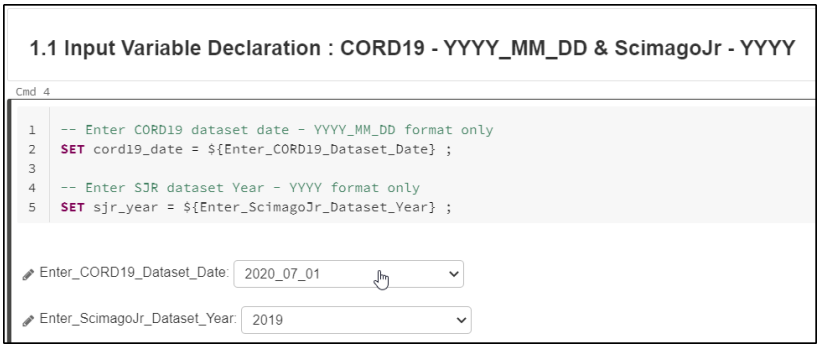
(1) Spark Jobs

```
Out[65]: [Row(Rank=1, Sourceid='28773', Title='CA - A Cancer Journal for Clinicians', Type='journa
l', Issn='15424863, 00079235', SJR=None, SJR_Best_Quartile='Q1', H_index=156, Total_Docs_2019=36,
Total_Docs_3years=129, Total_Refs=2924, Total_Cites_3years=22644, Citable_Docs_3years=89, Cites/Do
c_2years=None, Ref/Doc='81,22', Country='United States', Region='Northern America', Publisher='Wil
ey-Blackwell', Coverage='1950-2020', Categories='Hematology (Q1); Oncology (Q1)'),
 Row(Rank=2, Sourceid='19434', Title='MMWR. Recommendations and reports : Morbidity and mortality
weekly report. Recommendations and reports / Centers for Disease Control', Type='journal', Issn='1
0575987, 15458601', SJR=None, SJR_Best_Quartile='Q1', H_index=138, Total_Docs_2019=4, Total_Docs_3
years=11, Total_Refs=144, Total_Cites_3years=898, Citable_Docs_3years=11, Cites/Doc_2years=None, R
ef/Doc='36,00', Country='United States', Region='Northern America', Publisher='Centers for Disease
Control and Prevention (CDC)', Coverage='1990-2020', Categories='Epidemiology (Q1); Health Informa
tion Management (Q1); Health (social science) (Q1); Health, Toxicology and Mutagenesis (Q1); Medic
ine (miscellaneous) (Q1)'),
```

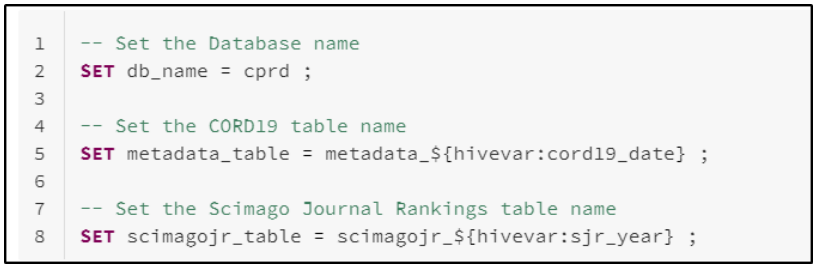
2.3 HiveQL

2.3.1 Setting the stage

- Extract date for CORD19 and year for Scimago datasets are set as hive input variables assuming that the relevant files are present in the Hive metastore file system - "dbfs:/user/hive/warehouse/"



- Hive variables for the names of database, metadata_table, and scimagojr_table are set using the above created input variables.



- Hive path variables have been created for the files present in the hive metastore.

2.3.2 Database Creation

- The database is initially dropped if exists to ensure reusability using CASCADE which would drop the underlying data structures as well.
- Using the 'db_name' variable and DB properties, a database is created and listed.

2.1 Create Database from the Hive variable : db_name

Cmd 12

2.1.1 Drop, Create & Show Database

Cmd 13

```
DROP DATABASE IF EXISTS ${hivevar:db_name} CASCADE;
CREATE DATABASE ${hivevar:db_name} WITH DBPROPERTIES (
  ID = 001,
  Name = 'Ambareesh Jonnavittula',
  Type = 'admin',
  DBType = 'Production'
);
SHOW DATABASES;
```

	databaseName
1	cprd
2	cqua
3	default

Showing all 3 rows.

- A database is properly created with a .db extension as shown in extended properties.

2.1.2 Describe the Database Properties

Cmd 15

```
DESCRIBE DATABASE EXTENDED ${hivevar:db_name}
```

	database_description_item	database_description_value
1	Database Name	cprd
2	Comment	
3	Location	dbfs:/user/hive/warehouse/cprd.db
4	Owner	root
5	Properties	((ID,001), (Name,Ambareesh Jonnavittula), (DBType,Production), (Type,admin))

Showing all 5 rows.

Command took 0.05 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/8/2021, 6:30:15 PM on amjon2

2.3.3 Tables / Views creation

- Using the table variables, again the tables are dropped if exists and created.
- Path is given from the hive path variable created.
- Header is present, schema is inferred, delimiter is a comma.
- Additional arguments for quote and escape are given as “\””, which will clean the raw data.

2.2 Create permanent tables in the Database

Cmd 17

2.2.1 Create a table for cord19 dataset and rename it in the format - 'metadata_YYYY_MM_DD' within the database

Cmd 18

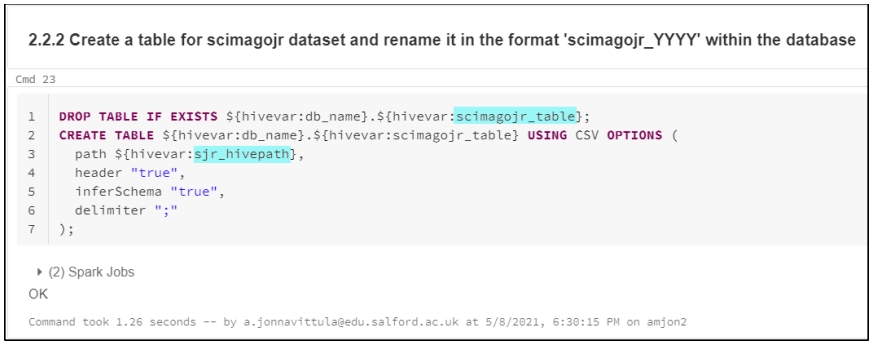
```
DROP TABLE IF EXISTS ${hivevar:db_name}.${hivevar:metadata_table};
CREATE TABLE ${hivevar:db_name}.${hivevar:metadata_table} USING CSV OPTIONS (
  path ${hivevar:cord19_hivepath},
  header "true",
  inferSchema "true",
  delimiter ",",
  quote "\"",
  escape "\""
);
```

(2) Spark Jobs

OK

Command took 7.55 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/8/2021, 6:30:15 PM on amjon2

- For Scimago dataset, header is present, schema is inferred but the delimiter is a semicolon.



2.2.2 Create a table for scimagojr dataset and rename it in the format 'scimagojr_YYYY' within the database

Cmd 23

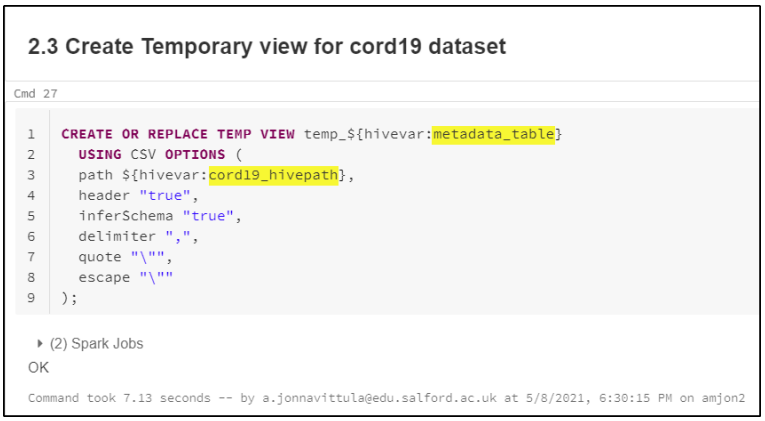
```
DROP TABLE IF EXISTS ${hivevar:db_name}.${hivevar:scimagojr_table};
CREATE TABLE ${hivevar:db_name}.${hivevar:scimagojr_table} USING CSV OPTIONS (
  path ${hivevar:sjr_hivepath},
  header "true",
  inferSchema "true",
  delimiter ";"
);
```

(2) Spark Jobs

OK

Command took 1.26 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/8/2021, 6:30:15 PM on amjon2

- A temporary view is created/replaced using the same method as for the metadata_table which was created above.



2.3 Create Temporary view for cord19 dataset

Cmd 27

```
CREATE OR REPLACE TEMP VIEW temp_${hivevar:metadata_table}
  USING CSV OPTIONS (
  path ${hivevar:cord19_hivepath},
  header "true",
  inferSchema "true",
  delimiter ",",
  quote "\"",
  escape "\""
);
```

(2) Spark Jobs

OK

Command took 7.13 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/8/2021, 6:30:15 PM on amjon2

- A new table stored as parquet is created from the existing metadata_table. Only this time, we create a schema limiting the number of columns. Notice that we don't use partitioning here, which would result in better performance overall.

2.4 Parquet file to Table

Cmd 31

```
DROP TABLE IF EXISTS ${hivevar:db_name}.parq_${hivevar:metadata_table};

CREATE TABLE ${hivevar:db_name}.parq_${hivevar:metadata_table}
   ( cord_uid STRING,
     sha STRING,
     source_x STRING,
     title STRING,
     doi STRING,
     pmcid STRING,
     pubmed_id STRING,
     license STRING,
     abstract STRING,
     publish_time STRING,
     authors STRING,
     journal STRING,
     url STRING
   )
--   LIKE ${hivevar:db_name}.${hivevar:metadata_table}
  STORED AS PARQUET
  LOCATION "dbfs:/user/hive/warehouse/cord19" ;
```

OK

Command took 0.23 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/8/2021, 6:30:15 PM on amjon2

- If we don't use 'overwrite', we create duplicate values.

```
INSERT OVERWRITE TABLE ${hivevar:db_name}.parq_${hivevar:metadata_table}
  SELECT cord_uid,
         sha,
         source_x,
         title,
         doi,
         pmcid,
         pubmed_id,
         license,
         abstract,
         publish_time,
         authors,
         journal,
         url
  FROM ${hivevar:db_name}.${hivevar:metadata_table};
```

(1) Spark Jobs

OK

Command took 19.11 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/8/2021, 6:30:15 PM on amjon2

2.3.4 Quality check

- Count the records and view the top 2 records to see the clean data.

2.2.1.1 Quality Check for 'metadata_table'

Cmd 20

```
SELECT COUNT(*) as `Num_of_Records` FROM ${hivevar:db_name}.${hivevar:metadata_table}
```

(2) Spark Jobs

	Num_of_Records
1	169821

Showing all 1 rows.

Command took 4.68 seconds -- by a.jonnavittula@edu.s

```
SELECT
  *
FROM
  ${hivevar:db_name}.${hivevar:metadata_table}
LIMIT
  2;
```

(1) Spark Jobs

	cord_uid	sha	source_x	title
1	ug7v899j	d1aafb70c066a2068b02786f8929fd9c900897fb	PMC	Clinical features of culture-proven Mycoplasma pneumoniae Arabia
2	02tnwd4m	6b0567729c2143a66d737eb0a2f63f2dce2e5a7d	PMC	Nitric oxide: a pro-inflammatory mediator in lung disease?

Showing all 2 rows.

- Temp view and Parquet table

2.3.1 Quality Check for temp view 'metadata_table'

Cmd 29

```
SELECT COUNT(*) as `Num_of_Records` FROM temp_${hivevar:metadata_table}
```

(2) Spark Jobs

	Num_of_Records
1	169821

Showing all 1 rows.

2.4.1 Quality Check for parquet table created

Cmd 34

```
SELECT COUNT(*) as `Num_of_Records` FROM ${hivevar:db_name}.parq_${hivevar:metadata_table}
```

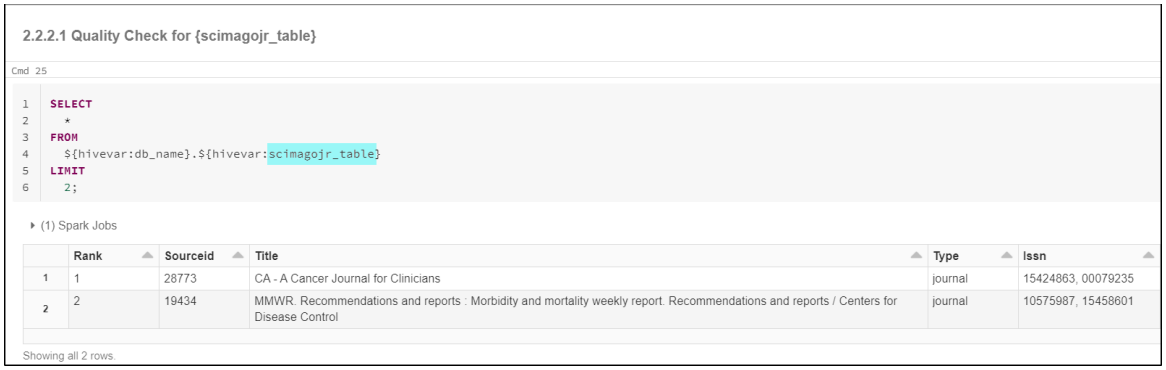
(2) Spark Jobs

	Num_of_Records
1	169821

Showing all 1 rows.

Command took 1.18 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/8/2021, 6:30:15 PM on amjon2

- Scimago table



2.2.2.1 Quality Check for {scimagojr_table}

Cmd 25

```
SELECT
  *
FROM
  ${hivevar:db_name}.${hivevar:scimagojr_table}
LIMIT
  2;
```

(1) Spark Jobs

	Rank	Sourceid	Title	Type	Issn
1	1	28773	CA - A Cancer Journal for Clinicians	journal	15424863, 00079235
2	2	19434	MMWR. Recommendations and reports : Morbidity and mortality weekly report. Recommendations and reports / Centers for Disease Control	journal	10575987, 15458601

Showing all 2 rows.

- Show the tables & views to check availability.

```
SHOW TABLES
FROM
  ${hivevar:db_name};
```

	database	tableName	isTemporary
1	cprd	metadata_2020_07_01	false
2	cprd	parq_metadata_2020_07_01	false
3	cprd	scimagojr_2019	false

Showing all 7 rows.

Command took 0.07 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/8/2021,

```
SHOW VIEWS LIKE 'temp_metadata*'
```

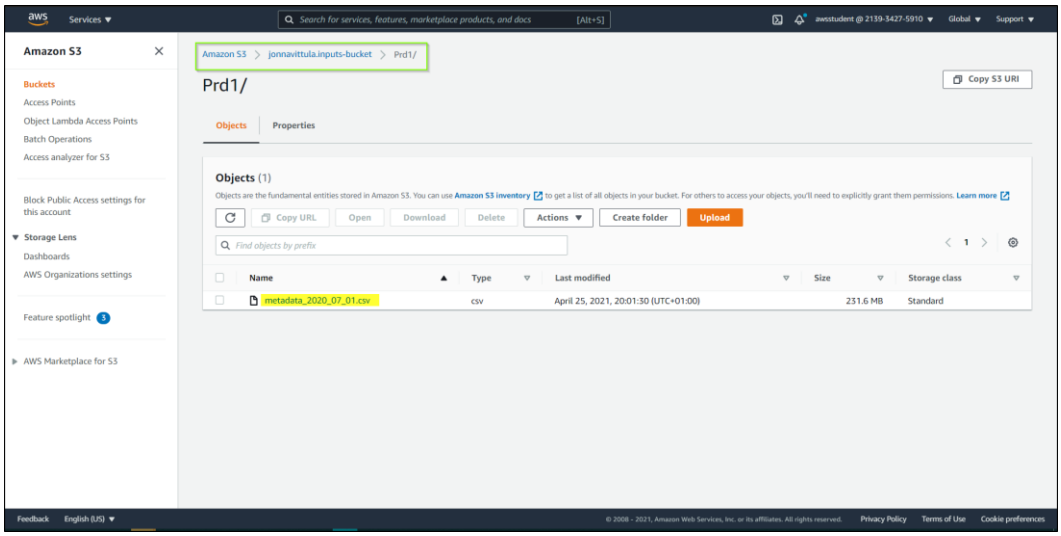
	namespace	viewName	isTemporary
1		temp_metadata_2020_03_27	true
2		temp_metadata_2020_05_01	true
3		temp_metadata_2020_07_01	true

Showing all 3 rows.

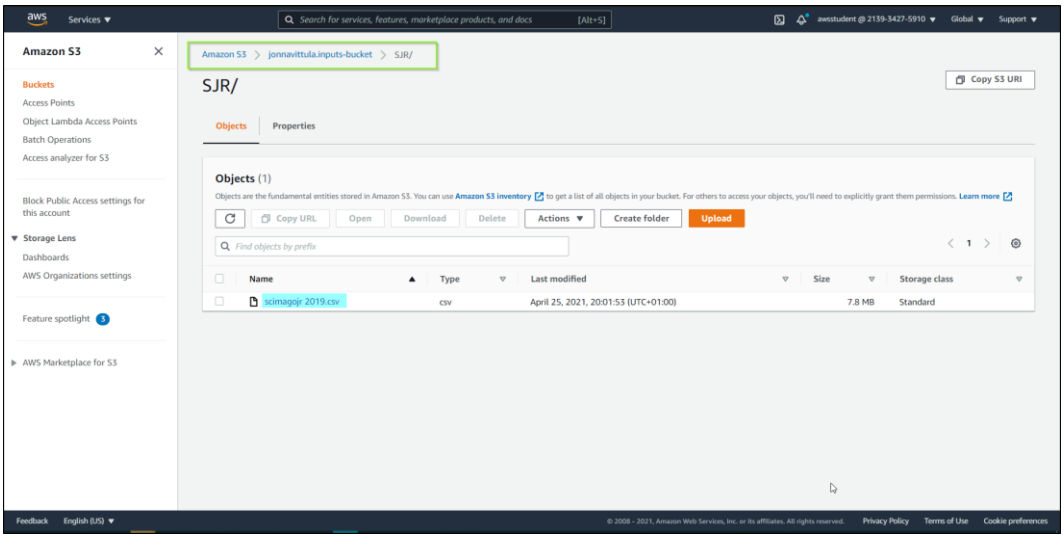
Command took 0.06 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/8/2021, 6:30:15 PM on amjon2

2.4 AWS S3 & Athena

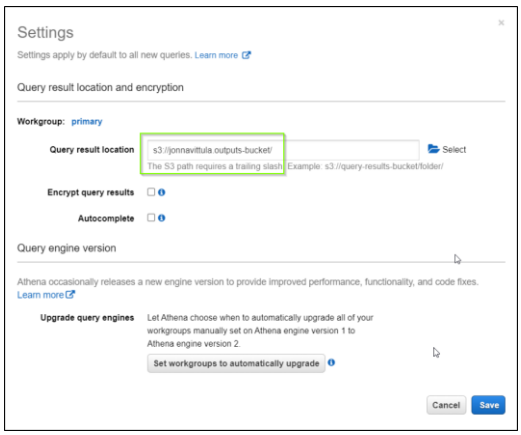
- In AWS S3, a new bucket – Jonnavittula.inputs-bucket is created. A folder ‘Prd1’ contains the cord19 dataset.



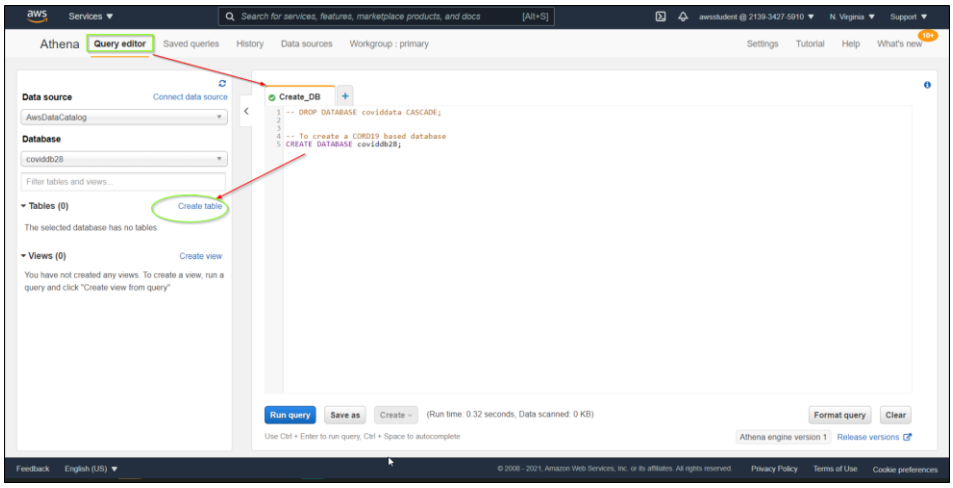
- Similarly, scimago dataset is present within another folder ‘SJR’.



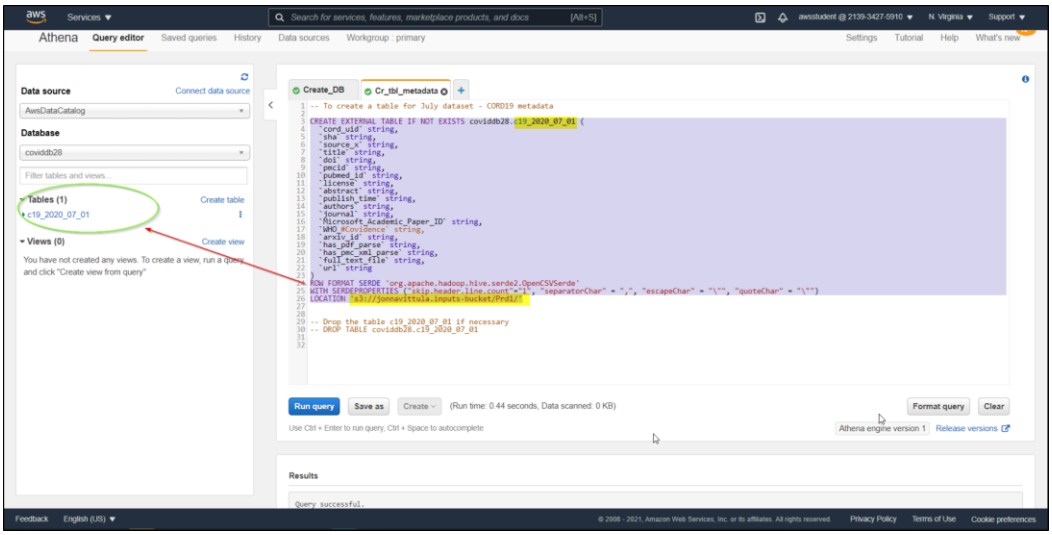
- A new S3 bucket for outputs is created – ‘s3://jonnavittula.outputs-bucket/’ which will store our query results from Athena. Query result location within Athena is given.



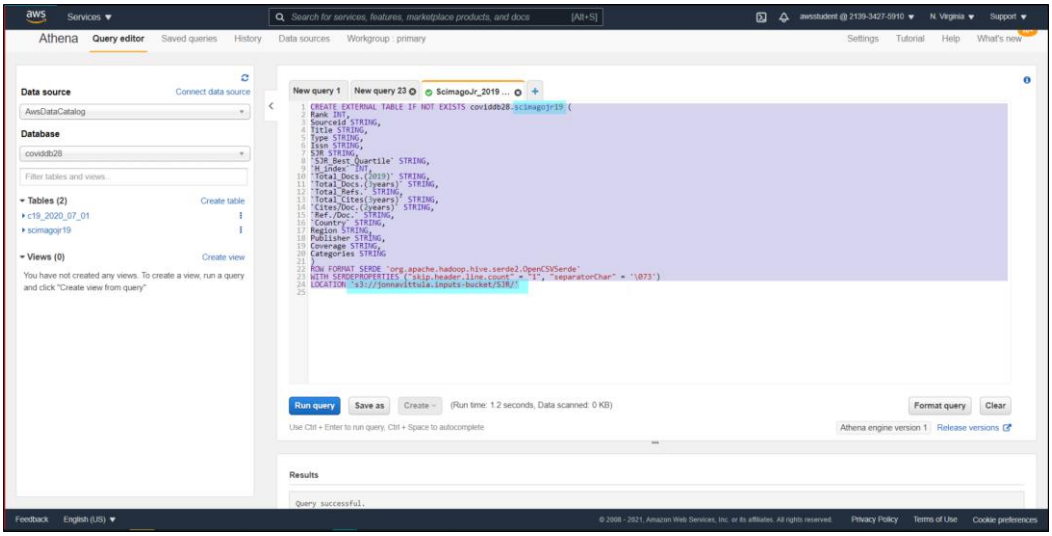
- In Athena Query editor, we proceed with the following steps –
  1. Create Database, tables and views.



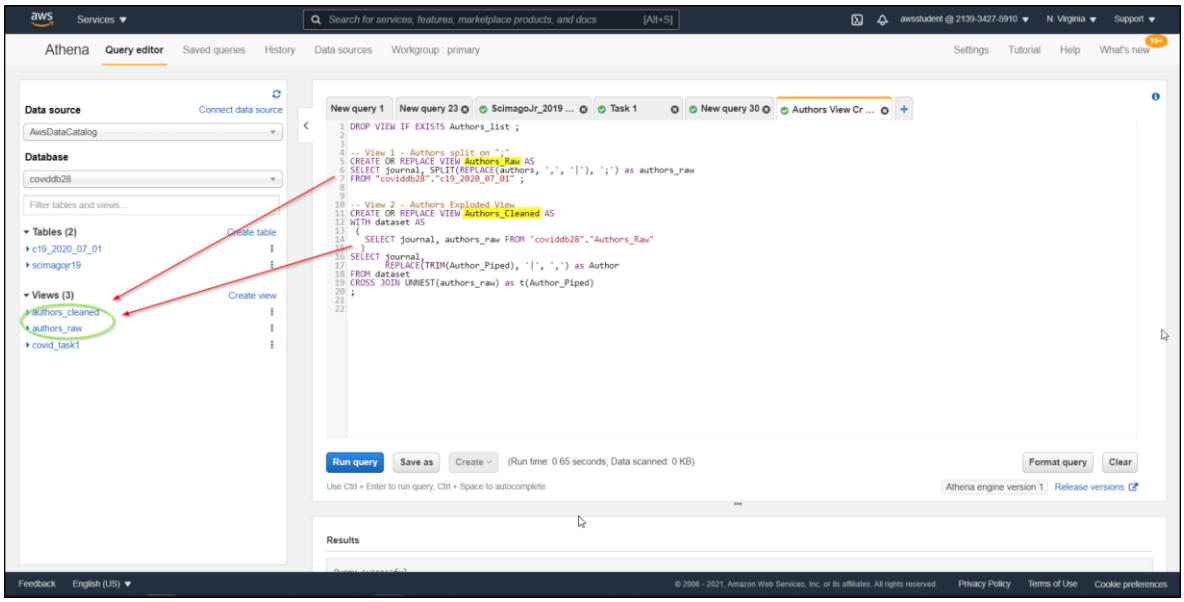
  2. For metadata_table, we skip the header line, separatorChar is a comma, escape and quote characters are “\””



  3. Whereas for scimago, Presto accepts the ascii value of semi-colon ‘\073’ and executes as expected.

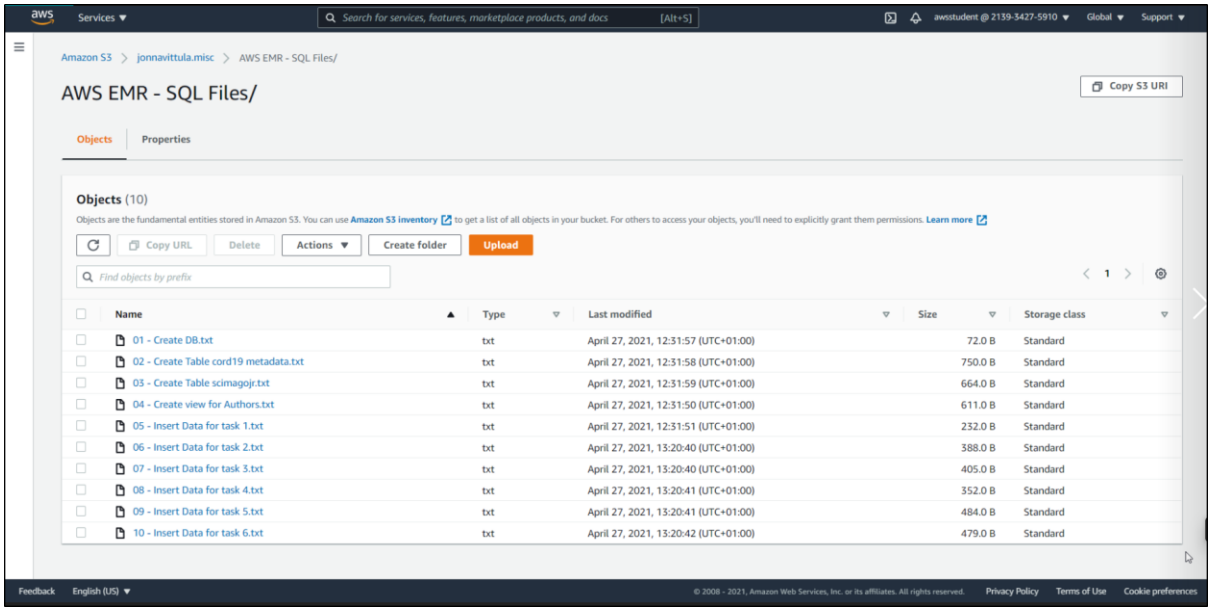


4. A view for raw list of authors is created by replacing the commas with a pipe for data cleaning purposes in Presto SQL which is different to HiveQL, followed by a split on semicolon.
5. Later a cleaned list of authors is created from the raw authors view using Presto's CROSS JOIN UNNEST which is an equivalent to LATERAL VIEW in HiveQL. Here 'WITH dataset as ()' enables the subquery functionality.

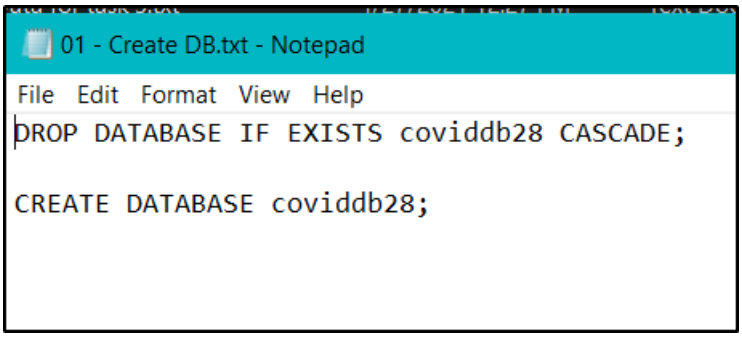


2.5 AWS S3 & EMR

- SQL files converted as .txt files and uploaded into an S3 bucket.



- All the scripts we used in Athena are used here in .txt format.
- Locations ${INPUT} & ${OUTPUT} variables refer to the values given in the EMR cluster steps.



02 - Create Table cord19 metadata.txt - Notepad

File Edit Format View Help

```
DROP TABLE IF EXISTS coviddb28.c19_2020_07_01;

CREATE EXTERNAL TABLE coviddb28.c19_2020_07_01 (
  `cord_uid` string,
  `sha` string,
  `source_x` string,
  `title` string,
  `doi` string,
  `pmcid` string,
  `pubmed_id` string,
  `license` string,
  `abstract` string,
  `publish_time` string,
  `authors` string,
  `journal` string,
  `Microsoft_Academic_Paper_ID` string,
  `WHO_#Covidence` string,
  `arxiv_id` string,
  `has_pdf_parse` string,
  `has_pmc_xml_parse` string,
  `full_text_file` string,
  `url` string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES ("skip.header.line.count"="1", "separatorChar" = ",", "escapeChar" = "\"", "quoteChar" = "\"")
LOCATION '${INPUT}';
```

03 - Create Table scimagojr.txt - Notepad

File Edit Format View Help

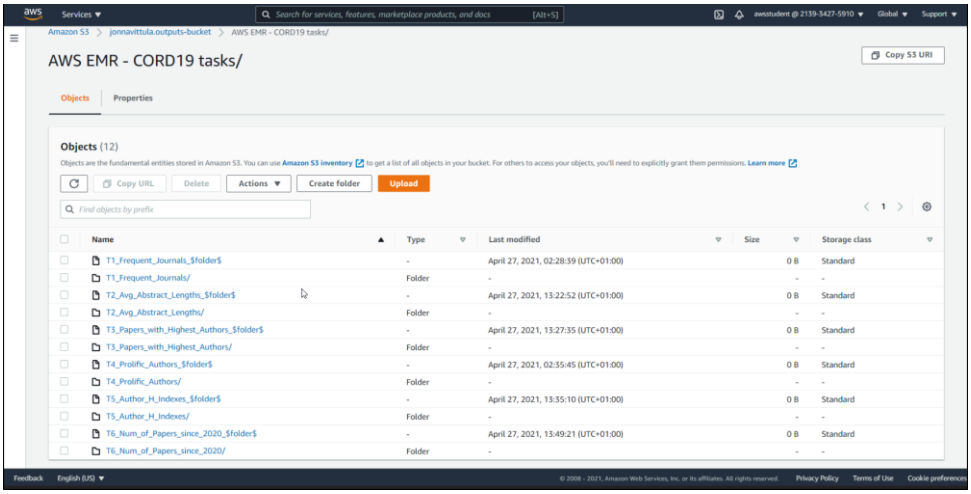
```
DROP TABLE IF EXISTS coviddb28.scimagojr19;

CREATE EXTERNAL TABLE coviddb28.scimagojr19 (
`Rank` INT,
`Sourceid` STRING,
`Title` STRING,
`Type` STRING,
`Issn` STRING,
`SJR` STRING,
`SJR_Best_Quartile` STRING,
`H_index` INT,
`Total_Docs(2019)` STRING,
`Total_Docs(3years)` STRING,
`Total_Refs` STRING,
`Total_Cites(3years)` STRING,
`Cites/Doc(2years)` STRING,
`Ref/Doc` STRING,
`Country` STRING,
`Region` STRING,
`Publisher` STRING,
`Coverage` STRING,
`Categories` STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES ("skip.header.line.count" = "1", "separatorChar" = '\073')
LOCATION '${INPUT}'
```

04 - Create view for Authors.txt - Notepad

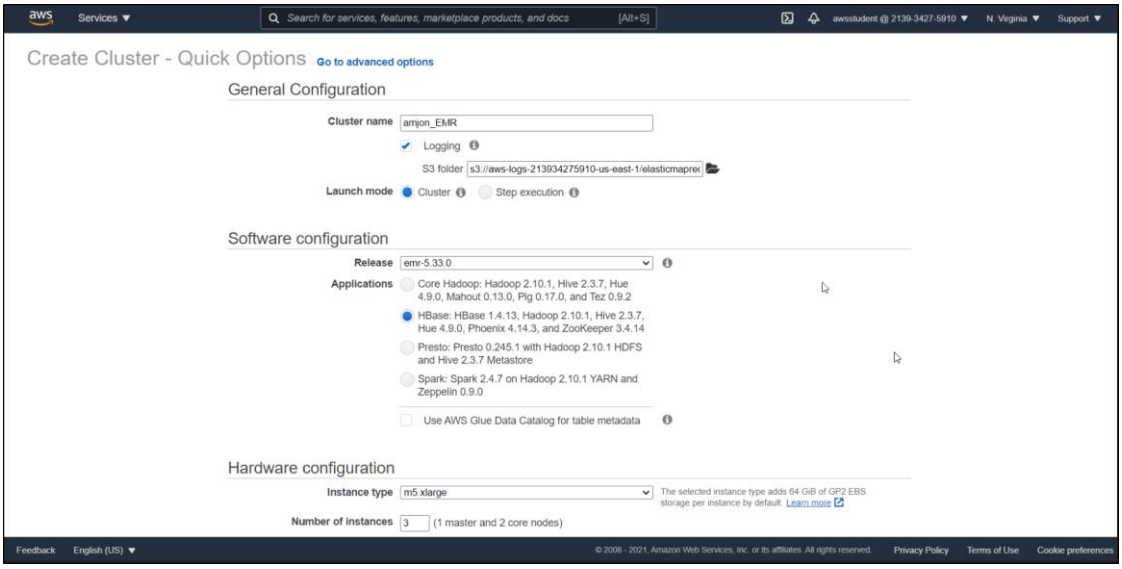
File Edit Format View Help

```
DROP VIEW IF EXISTS coviddb28.Authors_Raw ;
DROP VIEW IF EXISTS coviddb28.Authors_Cleaned ;


-- View 1 - Authors split on ";"
CREATE OR REPLACE VIEW coviddb28.Authors_Raw AS
SELECT journal, SPLIT(REPLACE(authors, ',', '|'), ';') as authors_raw
FROM coviddb28.c19_2020_07_01  ;


-- View 2 - Authors Exploded View
CREATE OR REPLACE VIEW coviddb28.Authors_Cleaned AS
WITH dataset AS
 (
   SELECT journal, authors_raw FROM coviddb28.Authors_Raw
  )
SELECT journal,
       REPLACE(TRIM(Author_Piped), '|', ',') as Author
FROM dataset
LATERAL VIEW EXPLODE(authors_raw) t as Author_Piped
;
```

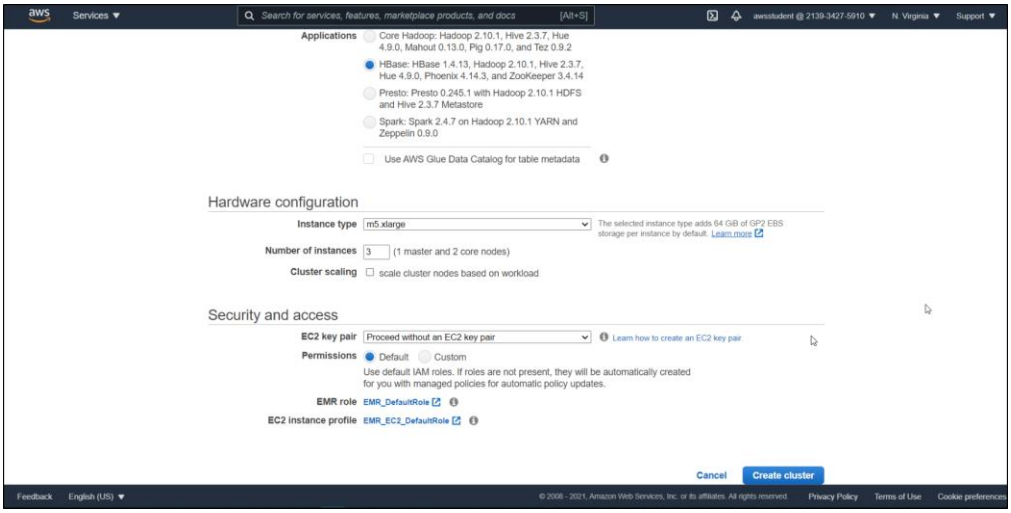
- Output bucket was kept ready with the necessary folders for tasks.



- An EMR cluster was created with the applications – Hive, HBase, Hadoop, and instance type is m5.xlarge.

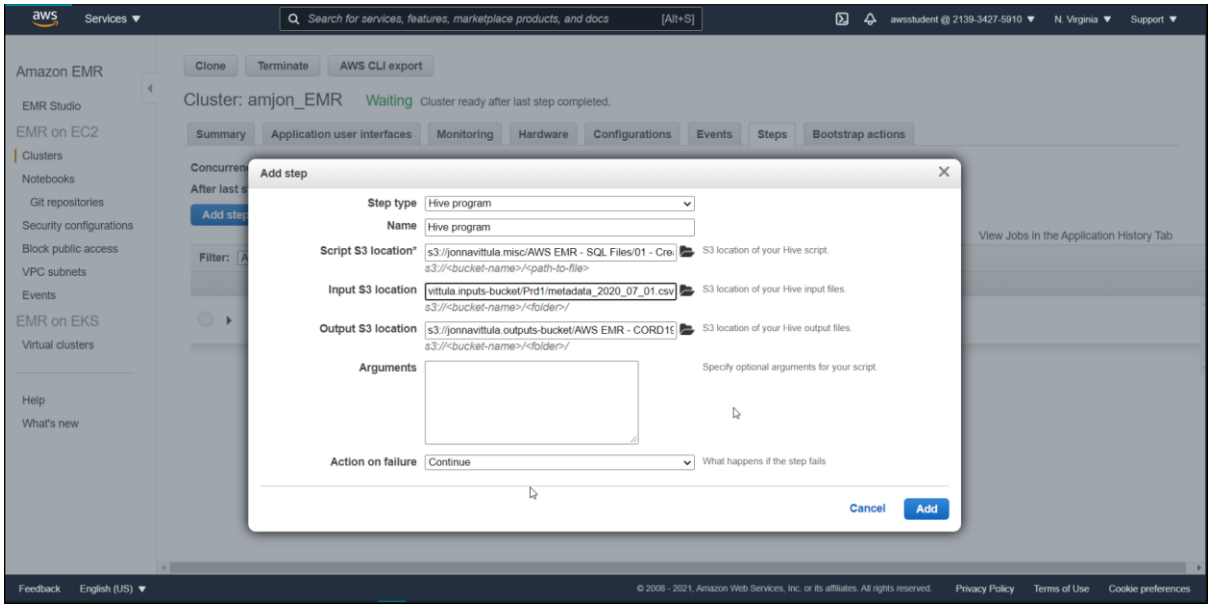


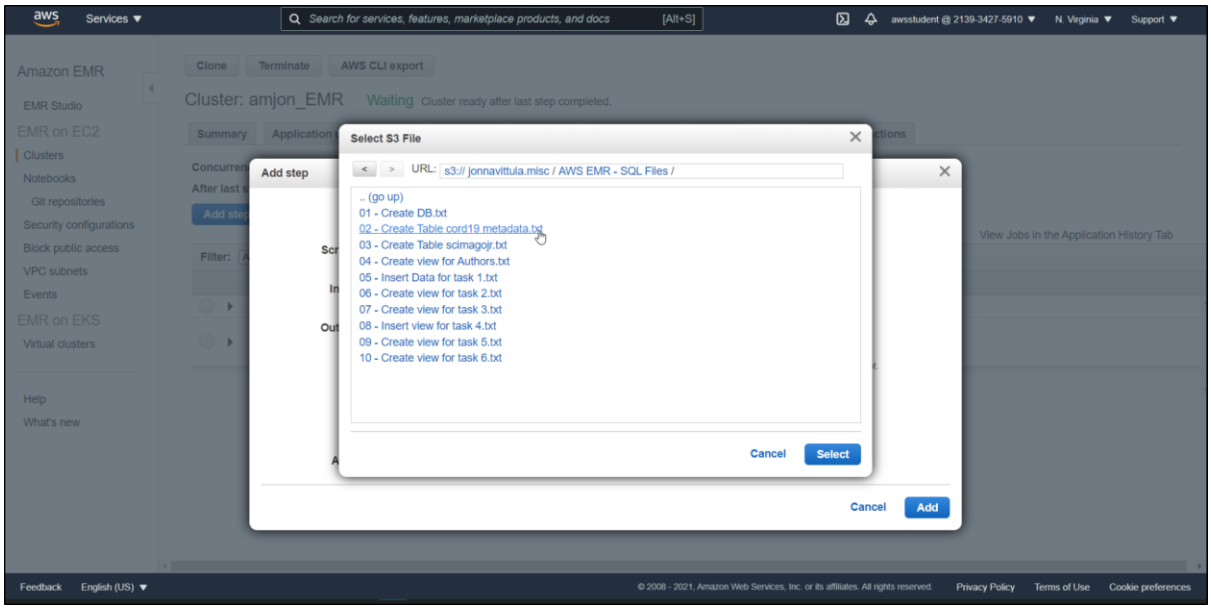
- EC2 key pair is not needed in this case.



- Cluster remains in Waiting state until a step is performed.

- Added steps with Hive program type, script location path and I/O paths.





Section 3: Solutions given for Problems / Tasks

- The implementation was done in 3 different ways for the PySpark dataframes & HiveQL tables created from csv, temp view and parquet file, and all the results match.
- In HiveQL, it is assumed that the .csv files are already uploaded into Databricks file system – Hive metastore. To refer the tables/views, hive variables were used.
- In RDD, Lambda/anonymous functions are used while computing transformations.

3.1: Task 1 - Find the 5 most common journals

3.1.0 Outline

1. Journal column was selected excluding the nulls
2. The journals were grouped by their individual total count and sorted.
3. Aliases were applied and top 5 records were shown.

3.1.1 Assumptions made

- The top 5 journals are read by a readers' market.

3.1.2 PySpark – Dataframes

1. Nulls were dropped using **.na.drop()** method.
2. **.withColumnRenamed()** was used to give alias in Dataframes.

3.1 Using Dataframe created from CSV

Cmd 46

```
task1_df = df_cord.select("journal") \
                  .na.drop() \
                  .groupBy("journal")  \
                  .count()  \
                  .sort('count', ascending=False) \
                  .withColumnRenamed("journal", "Name of the Journal") \
                  .withColumnRenamed("count", "Frequency")

task1_df.show(5)
```

3.2 Using Dataframe created from Temp View

Cmd 48

```
# Perform the task
df_tempt1 = df_tempView.select("journal") \
                       .na.drop() \
                       .groupBy("journal")  \
                       .count()  \
                       .sort('count', ascending=False) \
                       .withColumnRenamed("journal", "Name of the Journal") \
                       .withColumnRenamed("count", "Frequency")

df_tempt1.show(5)
```

3.3 Using Dataframe created from Parquet Files

Cmd 50

```
# Perform the task
df_parqt1 = df_parq.select("journal") \
                       .na.drop() \
                       .groupBy("journal")  \
                       .count()  \
                       .sort('count', ascending=False) \
                       .withColumnRenamed("journal", "Name of the Journal") \
                       .withColumnRenamed("count", "Frequency")

df_parqt1.show(5)
```

3.1.3 HiveQL

1. Aliasing is done in SQL databricks using AS keyword
2. Spaces in the field names are handled using backtick / acute (`)

3.1 Using Permanent Tables

Cmd 44

```
SELECT
  journal AS `Name of the Journal`,
  count(journal) AS Frequency
FROM
  ${hivevar:db_name}.${hivevar:metadata_table}
GROUP BY
  journal
ORDER BY
  Frequency DESC
LIMIT
  5
```

3.2 Using Temp View

Cmd 46

```
SELECT
  journal AS `Name of the Journal`,
  count(journal) AS Frequency
FROM
  temp_${hivevar:metadata_table}
GROUP BY
  journal
ORDER BY
  Frequency DESC
LIMIT
  5
```

3.3 Using Table from Parquet File

Cmd 48

```
SELECT
  journal AS `Name of the Journal`,
  count(journal) AS Frequency
FROM
  ${hivevar:db_name}.parq_${hivevar:metadata_table}
GROUP BY
  journal
ORDER BY
  Frequency DESC
LIMIT
  5
```

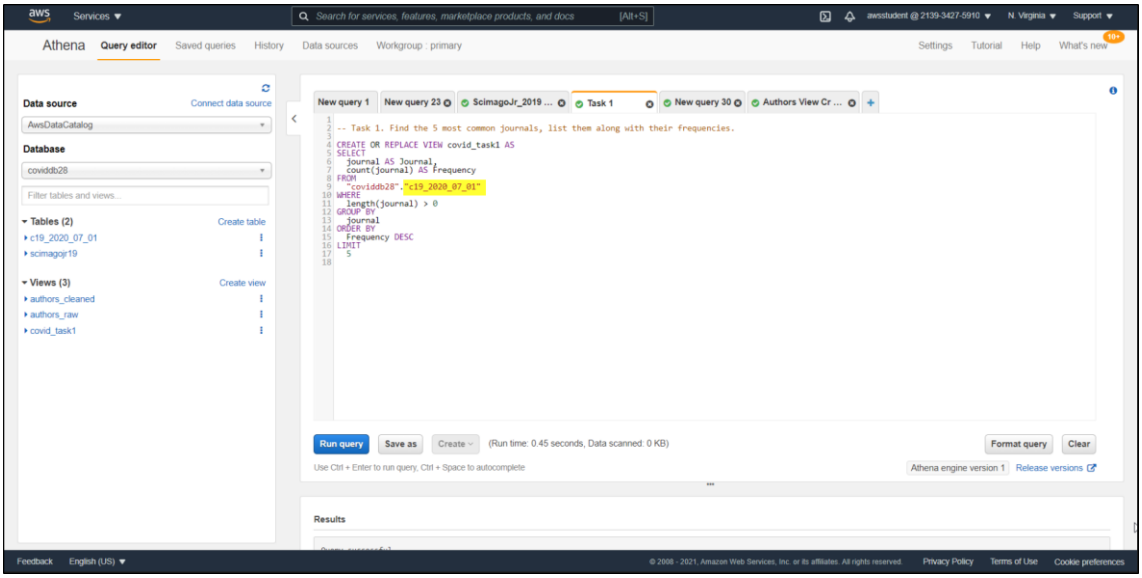
3.1.4 PySpark – RDD

1. Filter out the nulls from Journal column using '**is not None**' condition.
2. Key-value pairs are created for journal and 1, and reduced to count the journal repeats.

```
task1_rdd = cord19_rdd.filter(lambda row: row.journal is not None)  \
                      .map(lambda row: (row.journal,1)) \
                      .reduceByKey(lambda v1,v2 : v1+v2) \
                      .sortBy(lambda line: line[1], ascending=False)

task1_rdd.take(5)
```

3.1.5 AWS

3.1.5.1 AWS S3 & Athena

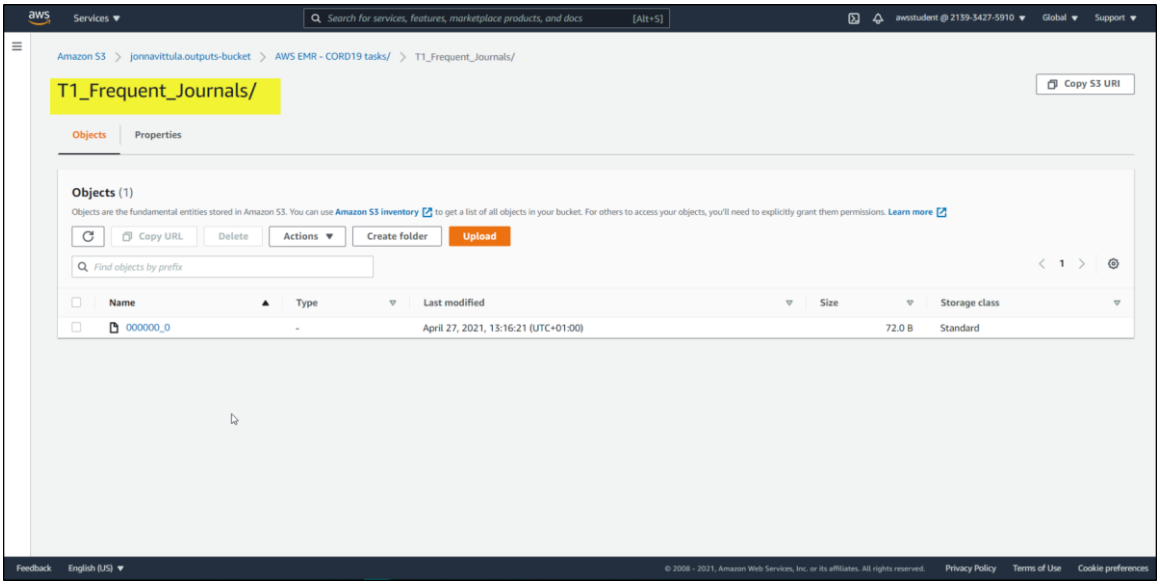


3.1.5.2 AWS S3 & EMR

05 - Insert Data for task 1.txt - Notepad

File Edit Format View Help

```
INSERT OVERWRITE DIRECTORY '${OUTPUT}'
SELECT
  journal AS Journal,
  count(journal) AS Frequency
FROM
  coviddb28.c19_2020_07_01
WHERE
  length(journal) > 0
GROUP BY
  journal
ORDER BY
  Frequency DESC
LIMIT
  5
;
```



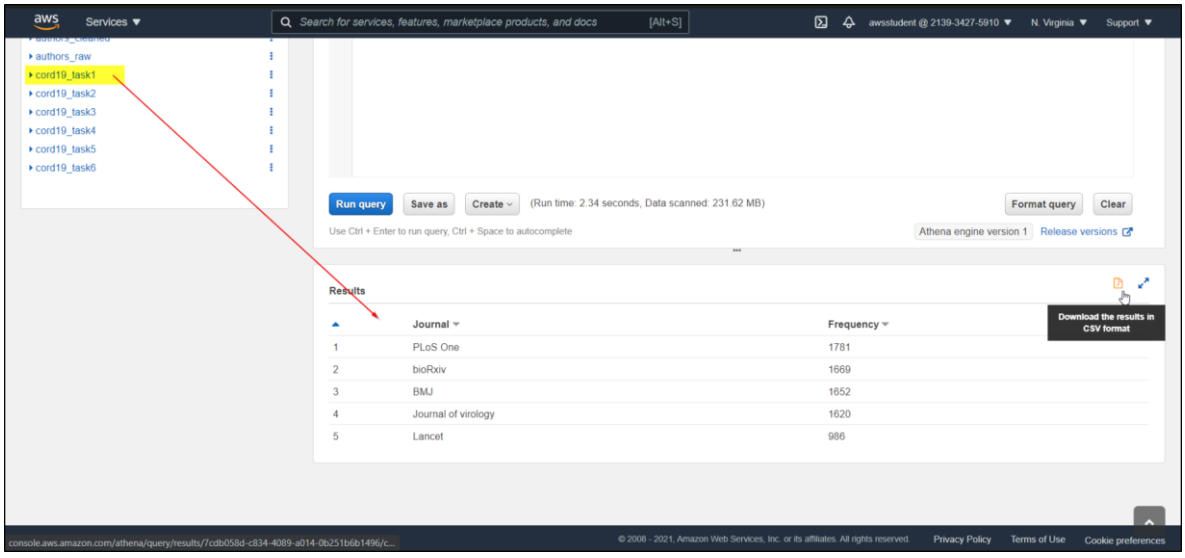
3.1.6 Result

```
+--------------------+---------+
|Name of the Journal|Frequency|
+--------------------+---------+
|            PLoS One|     1781|
|             bioRxiv|     1669|
|                 BMJ|     1652|
|Journal of virology|     1620|
|              Lancet|      986|
+--------------------+---------+
```

	Name of the Journal	Frequency
1	PLoS One	1781
2	bioRxiv	1669
3	BMJ	1652
4	Journal of virology	1620
5	Lancet	986

```
▸ (3) Spark Jobs
Out[59]: [('PLoS One', 1781),
 ('bioRxiv', 1669),
 ('BMJ', 1652),
 ('Journal of virology', 1620),
 ('Lancet', 986)]
```

```
000000_0 - Task 1 - Notepad
File  Edit  Format  View  Help
PLoS One 1781
bioRxiv 1669
BMJ 1652
Journal of virology 1620
Lancet 986
```



3.1.7 Result Review

- 'Journal of virology', which ranked as the most popular journal until May 2020, is now overtaken by 'PLoS One', 'bioRxiv' and 'BMJ' in that order, pushing it to 4th rank, making 'PLoS One' the most popular journal among the readers.
- It seems 'The Lancet' consistently maintained the 5th rank.
- In addition, 'Virology' and 'Emerg Infect Dis', which were on the top 5 until May, now seem to have lost their demand in the readers' market.

3.2: Task 2 - The top 5 average abstract lengths

3.2.0 Outline

1. Journal and abstract columns were selected excluding the nulls
2. Abstract column is customized to eliminate the unnecessary spaces by nullifying them first.
3. Grouping was done on journal column, while the average of the customized abstract column was aggregated and sorted.
4. In PySpark, the intermediate result is persisted to MEMORY_ONLY to optimize.
5. To this persisted result, the average abstract lengths are rounded off to a single decimal point and the columns are aliased.

3.2.1 Assumptions made

- N/A

3.2.2 PySpark – Dataframes

1. **.withColumn()** is used to customize the 'abstract' column to bring the length of each abstract by subtracting the additional spaces which are nullified using regexp_replace().

4.1 Using Dataframe created from CSV

Cmd 53

```
from pyspark.sql.functions import length, regexp_replace, round

df_nospace = df_cord.select("journal", "abstract") \
                    .na.drop() \
                    .withColumn("abstract", 1 + length(df_cord.abstract) - length(regexp_replace(df_cord.abstract, ' ', ''))) \
                    .groupBy("journal") \
                    .avg("abstract").withColumnRenamed("avg(abstract)", "avg_abstract") \
                    .sort("avg_abstract", ascending=False) \
                    .persist()

task2_df = df_nospace.withColumn("avg_abstract", round(df_nospace["avg_abstract"],1)) \
                     .withColumnRenamed("journal", "Name of the Journal") \
                     .withColumnRenamed("avg_abstract", "Avg_Abstract_Length") \

task2_df.show(5)
```

4.2 Using Dataframe created from Temp View

Cmd 55

```
from pyspark.sql.functions import length, regexp_replace, round

df_nospace_temp = df_tempView.select("journal", "abstract") \
                             .na.drop() \
                             .withColumn("abstract", 1 + length(df_tempView.abstract) - length(regexp_replace(df_tempView.abstract, ' ', ''))) \
                             .groupBy("journal") \
                             .avg("abstract").withColumnRenamed("avg(abstract)", "avg_abstract") \
                             .sort("avg_abstract", ascending=False) \
                             .persist()

df_tempt2 = df_nospace_temp.withColumn("avg_abstract", round(df_nospace_temp["avg_abstract"],1)) \
                           .withColumnRenamed("journal", "Name of the Journal") \
                           .withColumnRenamed("avg_abstract", "Avg_Abstract_Length") \

df_tempt2.show(5)
```

4.3 Using Dataframe created from Parquet Files

Cmd 57

```
from pyspark.sql.functions import length, regexp_replace, round

df_nospace_parq = df_parq.select("journal", "abstract") \
                         .na.drop() \
                         .withColumn("abstract", 1 + length(df_parq.abstract) - length(regexp_replace(df_parq.abstract, ' ', ''))) \
                         .groupBy("journal") \
                         .avg("abstract").withColumnRenamed("avg(abstract)", "avg_abstract") \
                         .sort("avg_abstract", ascending=False) \
                         .persist()

df_parqt2 = df_nospace_parq.withColumn("avg_abstract", round(df_nospace_parq["avg_abstract"],1)) \
                           .withColumnRenamed("journal", "Name of the Journal") \
                           .withColumnRenamed("avg_abstract", "Avg_Abstract_Length") \

df_parqt2.show(5)
```

3.2.3 HiveQL

1. In HiveQL, we use REPLACE() function to achieve the same functionality as above.

4.1 Using Permanent Tables

Cmd 51

```
SELECT
  journal AS `Name of the Journal`,
  ROUND(
    AVG(
      LENGTH(abstract) - LENGTH(REPLACE(abstract, ' ', '')) + 1
      ),1) AS `Average Abstract lengths`
FROM
  ${hivevar:db_name}.${hivevar:metadata_table}
GROUP BY
  journal
ORDER BY
  `Average Abstract lengths` DESC
LIMIT
  5
```

4.2 Using Temp View

Cmd 53

```
SELECT
  journal AS `Name of the Journal`,
  ROUND(
    AVG(
      LENGTH(abstract) - LENGTH(REPLACE(abstract, ' ', '')) + 1
      ),1) AS `Average Abstract lengths`
FROM
  temp_${hivevar:metadata_table}
GROUP BY
  journal
ORDER BY
  `Average Abstract lengths` DESC
LIMIT
  5
```

4.3 Using Table from Parquet File

Cmd 55

```
SELECT
  journal AS `Name of the Journal`,
  ROUND(
    AVG(
      LENGTH(abstract) - LENGTH(REPLACE(abstract, ' ', '')) + 1
      ),1) AS `Average Abstract lengths`
FROM
  ${hivevar:db_name}.parq_${hivevar:metadata_table}
GROUP BY
  journal
ORDER BY
  `Average Abstract lengths` DESC
LIMIT
  5
```

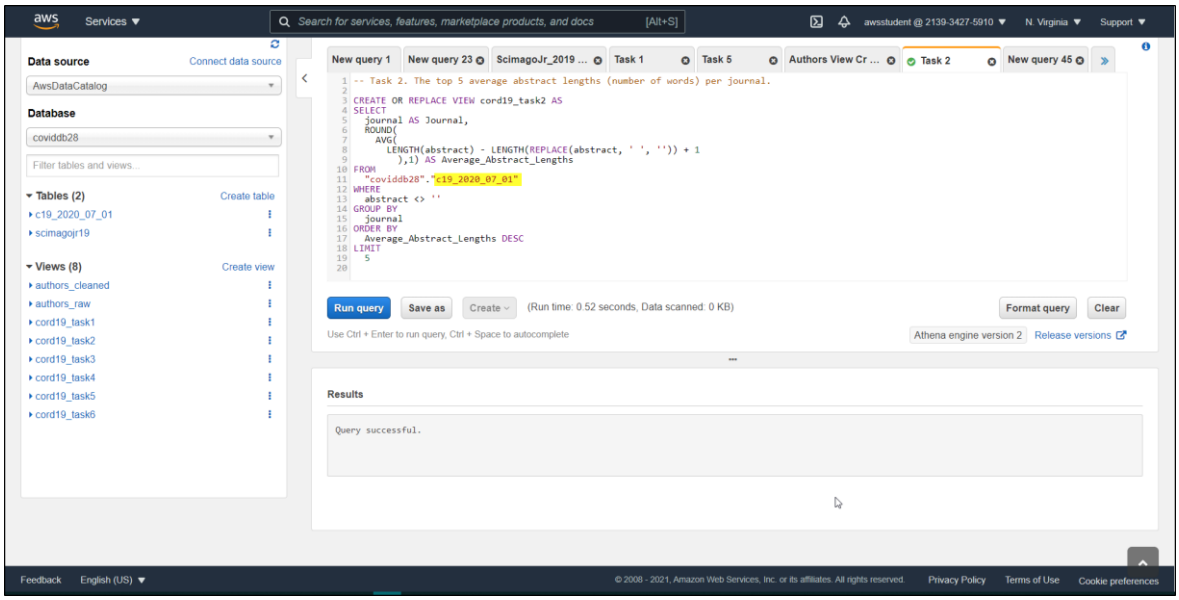
3.2.4 PySpark – RDD

1. Key-value pairs are created for journal and cleaned abstract length after nullifying spaces.
2. mapValues() is used to apply average formula directly on the values in the dictionary.

```
task2_rdd = cord19_rdd.filter(lambda row: row.abstract is not None)  \
                      .filter(lambda row: row.journal is not None)  \
                      .map(lambda row: (row.journal, 1 + len(row.abstract) -
len(row.abstract.replace(" ", "")))) \
                      .groupByKey() \
                      .mapValues(lambda x: round(sum(x) / len(x) ,1)) \
                      .sortBy(lambda line: line[1], ascending=False)

task2_rdd.take(5)
```

3.2.5 AWS

3.2.5.1 AWS S3 & Athena

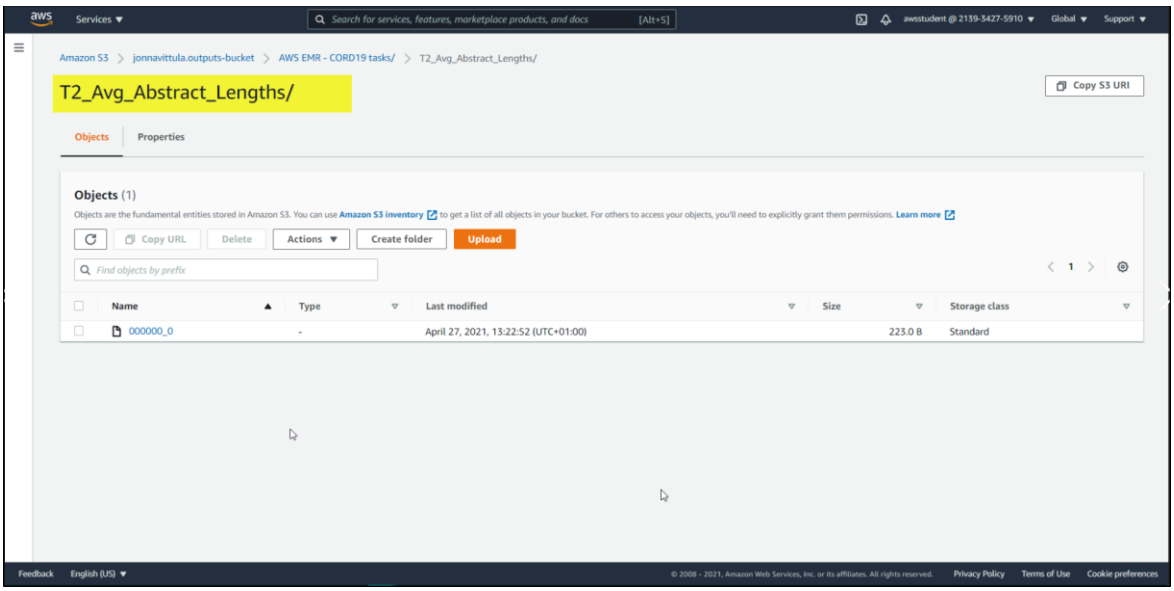


3.2.5.2 AWS S3 & EMR

06 - Insert Data for task 2.txt - Notepad

File Edit Format View Help

```
-- Task 2. The top 5 average abstract lengths (number of words) per journal.

INSERT OVERWRITE DIRECTORY '${OUTPUT}'
SELECT
  journal AS Journal,
  ROUND(
    AVG(
      LENGTH(abstract) - LENGTH(REPLACE(abstract, ' ', '')) + 1
        ),1) AS Average_Abstract_Lengths
FROM
  coviddb28.c19_2020_07_01
WHERE
  abstract<>''
GROUP BY
  journal
ORDER BY
  Average_Abstract_Lengths DESC
LIMIT
  5
```



3.2.6 Result

```
+--------------------+--------------------+
| Name of the Journal|Avg_Abstract_Length|
+--------------------+--------------------+
|Absolute Neurocri...|              2232.0|
|World Allergy Org...|              1677.1|
|Endocrine, metabo...|              1504.0|
|National Toxicolo...|              1301.8|
|       Tob Induc Dis|               969.0|
+--------------------+--------------------+
```

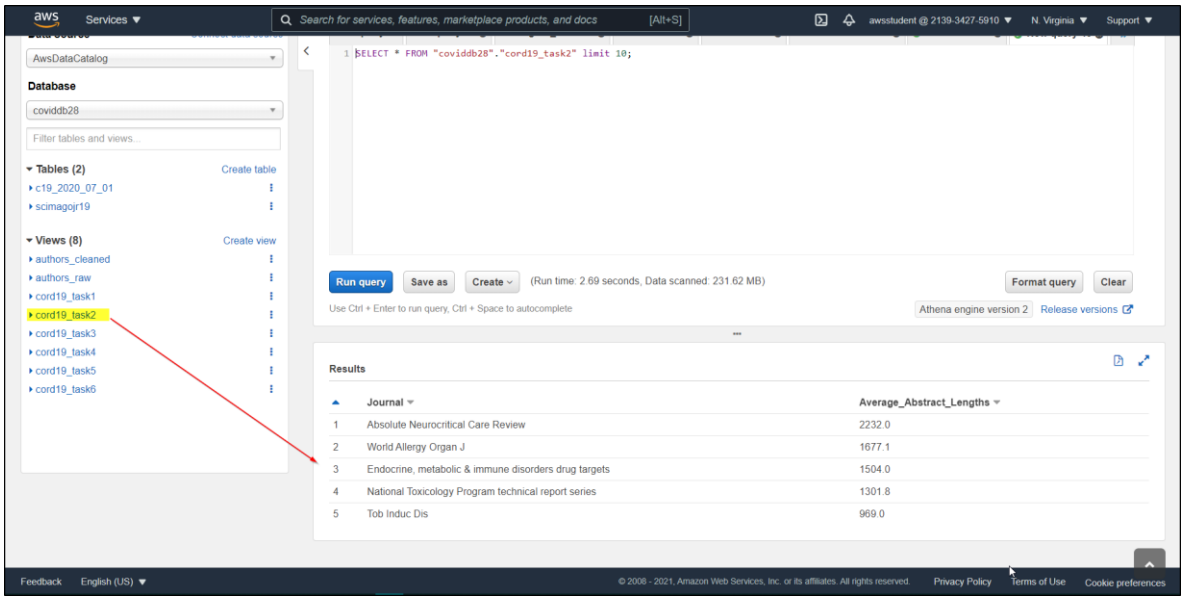
	Name of the Journal	Average Abstract lengths
1	Absolute Neurocritical Care Review	2232
2	World Allergy Organ J	1677.1
3	Endocrine, metabolic & immune disorders drug targets	1504
4	National Toxicology Program technical report series	1301.8
5	Tob Induc Dis	969

```
▸ (3) Spark Jobs
Out[58]: [('Absolute Neurocritical Care Review', 2232.0),
 ('World Allergy Organ J', 1677.1),
 ('Endocrine, metabolic & immune disorders drug targets', 1504.0),
 ('National Toxicology Program technical report series', 1301.8),
 ('Tob Induc Dis', 969.0)]
```

000000_0 - Task 2 - Notepad

File Edit Format View Help

```
Absolute Neurocritical Care Review2232.0
World Allergy Organ J1677.1
Endocrine, metabolic & immune disorders drug targets1504.0
National Toxicology Program technical report series1301.8
Tob Induc Dis969.0
```



3.2.7 Result Review

- Until May, 'World Allergy Orggan J' has an average abstract length of 4658.5, but now in July 2020, not many words are given in these copies of the journals, hence the average length got reduced to 1677.1.
- In July 2020, 'Absolute Neurocritical Care Review' has given the highest amount of words in their copies of abstract, followed by 'Endocrine, metabolic & immune disorders drug targets' and later 'National Toxicology Program technical report series'.
- 'Tob Induc Dis' has consistently remained at 969.0, but lessened its position from 4th to 5th.

3.2.8 Further Analysis 1

- **Question** - From the 3 most recent abstracts, what are the authors trying to lay importance on / stress on?

Steps –

1. Title, Publish_time, abstract columns are selected and the nulls are dropped.
2. Using substring method, 5th and 8th characters of the publish_time column are checked for a hyphen like in a date (YYYY-MM-DD) format. This is done to ensure we have only dates.
3. Abstract column is checked not to be 'Unknown' which is considered as a cliché in the data, and cleaned as part of the transformation.
4. Finally, sorting is done on Publish_time in the descending order and top 3 values are shown.

4.4.1 Dataset preparation

Cmd 60

```
# To create a DF with Abstracts & Journal published date in descending order (Top 3)

df_taskFA1 = df_cord.select("title", "publish_time", "abstract")   \
                    .na.drop()  \
                    .where(substring(df_cord["publish_time"],5,1) == '-') \
                    .where(substring(df_cord["publish_time"],8,1) == '-') \
                    .where(df_cord["abstract"] != 'Unknown') \
                    .sort("publish_time", ascending=False)

df_taskFA1.show(3)
```

```
+--------------------+------------+--------------------+
|               title|publish_time|            abstract|
+--------------------+------------+--------------------+
|21 Respiratory vi...|  2021-12-31|Abstract: Respira...|
|Chapter 27 Emerge...|  2020-12-31|Abstract The unpr...|
|Computer Vision T...|  2020-12-31|Abstract Digitiza...|
+--------------------+------------+--------------------+
only showing top 3 rows
```

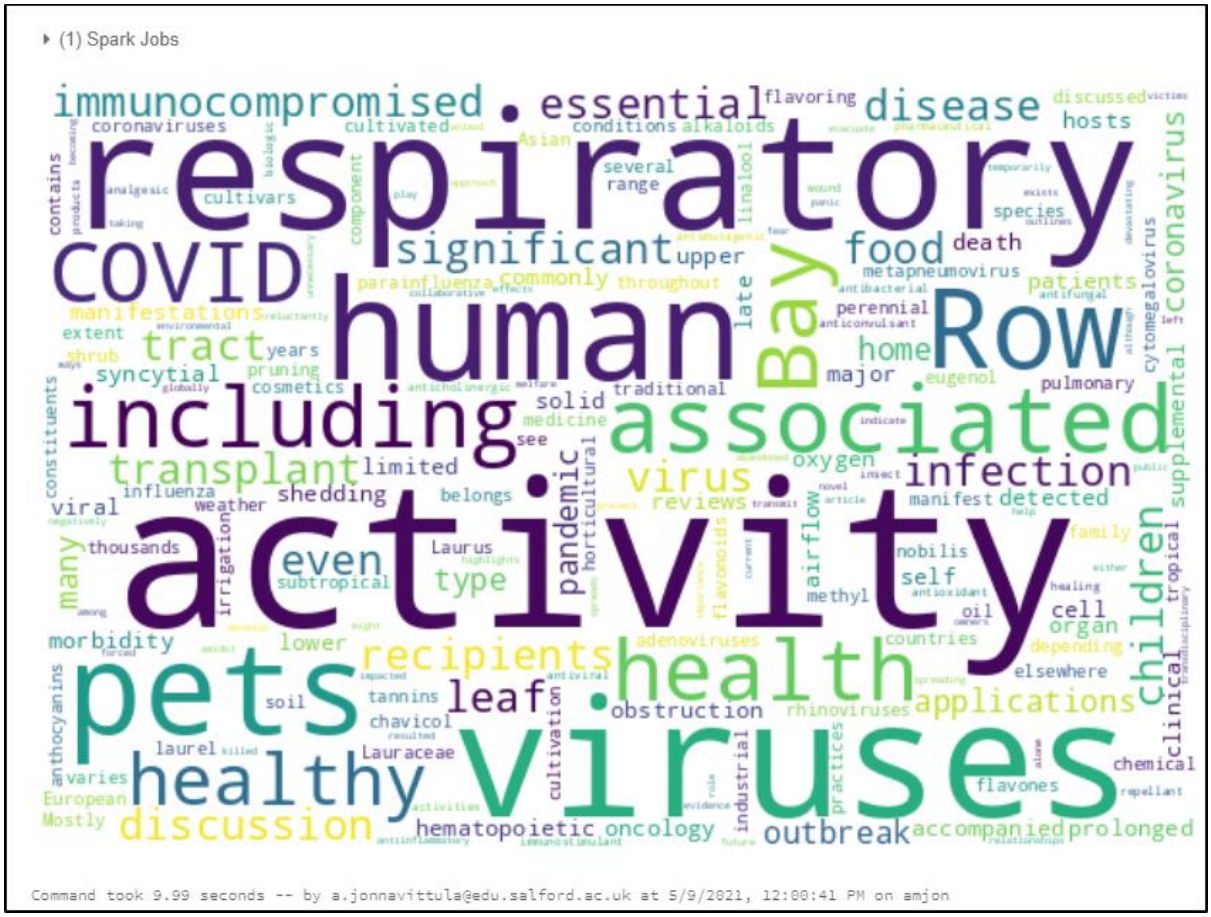
- Stopwords are used to remove words that don't add value to the plot.

4.4.2 Word Cloud plotting

Cmd 62

```
# Use the above created DF in a word cloud

import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
import sys, os
os.chdir(sys.path[0])

# Read the top 3 abstracts into text as a "String"
text = str(df_taskFA1.select("abstract").head(3))

# Custom Stop words
extra_stopwds = ['use', 'used', 'using', 'different', 'Abstract', 'need', 'due', 'method', 'system', 'data',
'machine', 'machining', 'chapter', 'one', 'well', 'will', 'may', 'new', 'two', 'tool', 'based', 'paper',
'material']

# Adding custom stopwords to the original stopwords 'set'
stopwords = STOPWORDS.update(extra_stopwds)

# Defining instance of WordCloud with input arguments
wc = WordCloud(background_color='white',
               stopwords=stopwords,
               height = 400,
               width=600)

# WordCloud Generate
wc.generate(text)

# Plot the figure using MatPlotLib
plt.figure(figsize=[20,10])
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

- **Result -**



- Result Review – By the end of the year, it seems that the journals have emphasized more on covid and associated viruses, along with their effects on the human respiratory system. In addition, they focused on the agents, especially the role of pets, which act as a catalyst sometimes, through which the people could get infected or even immunocompromised. Healthy lifestyle, food, oncology study, parainfluenza and metapneumovirus are the other topics which were accentuated.

3.3: Task 3 – 5 Titles with highest numbers of authors.

3.3.0 Outline

1. Title and Authors are selected from cord19 table
2. An exploded split is performed on Authors using the delimiter – semicolon ';'
3. Grouping is done on Title against the total count of authors, which is later sorted and aliased.

3.3.1 Assumptions made

- N/A

3.3.2 PySpark – Dataframes

1. Explode(split(df_cord.authors, ";")) shown below is used to first split the multiple authors within the same row and later use these authors list as an array within the dataframe.

5.1 Using Dataframe created from CSV

Cmd 65

```
from pyspark.sql.functions import split, explode

task3_df = df_cord.select("title", "authors") \
                  .withColumn("authors", explode(split(df_cord.authors, ";"))) \
                  .groupBy("title") \
                  .count() \
                  .sort("count", ascending=False) \
                  .withColumnRenamed("title", "Title") \
                  .withColumnRenamed("count", "Num_of_Authors")

task3_df.show(5)
```

5.2 Using Dataframe created from Temp View

Cmd 67

```
from pyspark.sql.functions import split, explode

df_tempt3 = df_tempView.select("title", "authors") \
                  .withColumn("authors", explode(split(df_tempView.authors, ";"))) \
                  .groupBy("title") \
                  .count() \
                  .sort("count", ascending=False) \
                  .withColumnRenamed("title", "Title") \
                  .withColumnRenamed("count", "Num_of_Authors")

df_tempt3.show(5)
```

5.3 Using Dataframe created from Parquet Files

Cmd 69

```
from pyspark.sql.functions import split, explode

df_parqt3 = df_parq.select("title", "authors") \
                  .withColumn("authors", explode(split(df_parq.authors, ";"))) \
                  .groupBy("title") \
                  .count() \
                  .sort("count", ascending=False) \
                  .withColumnRenamed("title", "Title") \
                  .withColumnRenamed("count", "Num_of_Authors")

df_parqt3.show(5)
```

3.2.3 HiveQL

5.1 Using Permanent Tables

Cmd 58

```
SELECT
  title as Title,
  COUNT(author) AS `Number of Authors`
FROM
  (
    SELECT
      title,
      explode(split(authors, ";")) as author
    FROM
      ${hivevar:db_name}.${hivevar:metadata_table}
  )
GROUP BY
  title
ORDER BY
  `Number of Authors` DESC
LIMIT
  5
```

5.2 Using Temp View

Cmd 60

```
SELECT
  title as Title,
  COUNT(author) AS `Number of Authors`
FROM
  (
    SELECT
      title,
      explode(split(authors, ";")) as author
    FROM
      temp_${hivevar:metadata_table}
  )
GROUP BY
  title
ORDER BY
  `Number of Authors` DESC
LIMIT
  5
```

5.3 Using Table from Parquet File

Cmd 62

```
SELECT
  title as Title,
  COUNT(author) AS `Number of Authors`
FROM
  (
    SELECT
      title,
      explode(split(authors, ";")) as author
    FROM
      ${hivevar:db_name}.parq_${hivevar:metadata_table}
  )
GROUP BY
  title
ORDER BY
  `Number of Authors` DESC
LIMIT
  5
```

3.3.4 PySpark – RDD

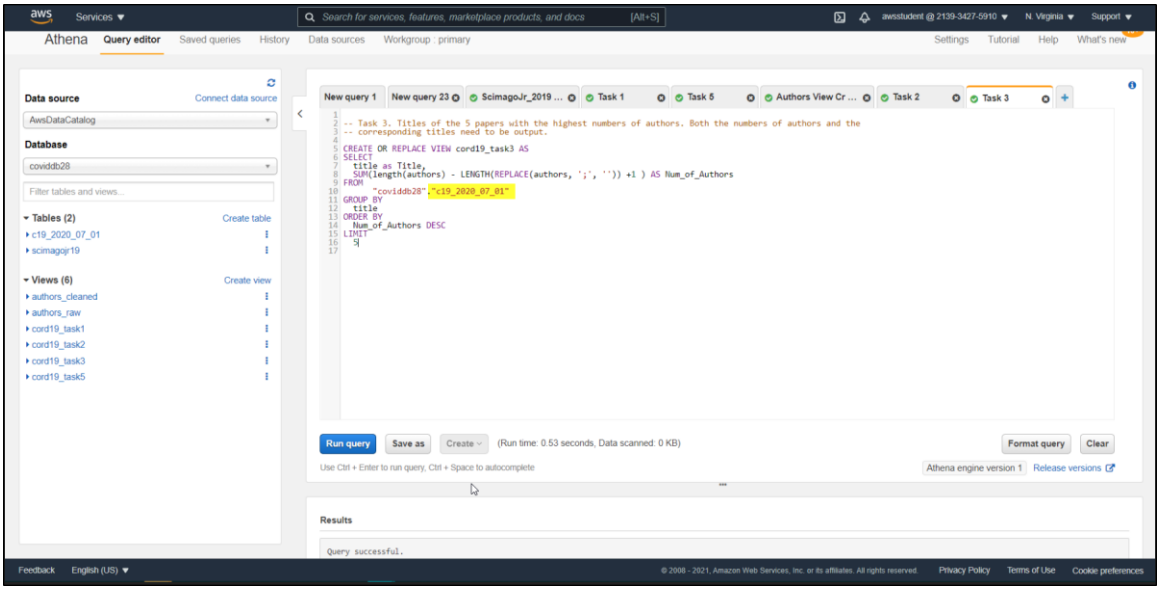
1. Explode is not needed in RDD, since we calculate the length of the author's array after the split at the iteration itself.

```
task3_rdd = cord19_rdd.filter(lambda row: row.title is not None) \
                      .filter(lambda row: row.authors is not None) \
                      .map(lambda row: [row.title,len(row.authors.split(";"))]) \
                      .sortBy(lambda line: line[1], ascending=False)

task3_rdd.take(5)
```

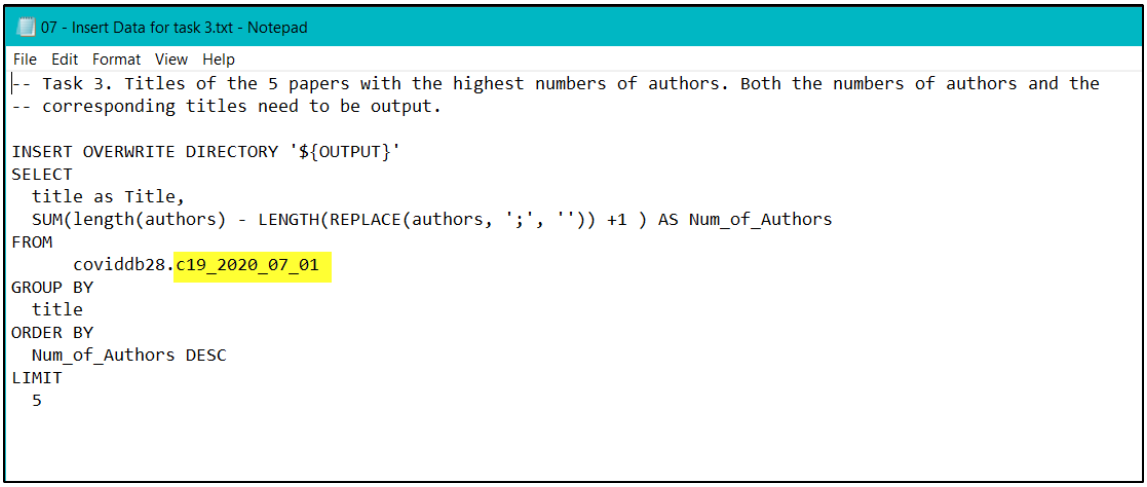
3.3.5 AWS

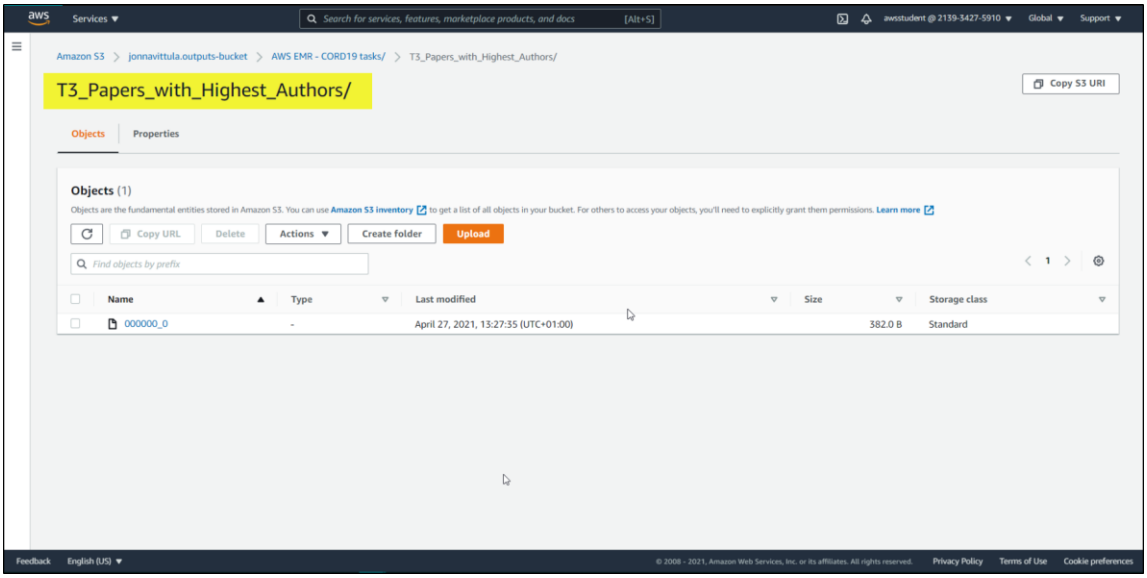
3.3.5.1 AWS S3 & Athena

1. In Presto also, we calculate the average directly at the query level.



3.3.5.2 AWS S3 & EMR



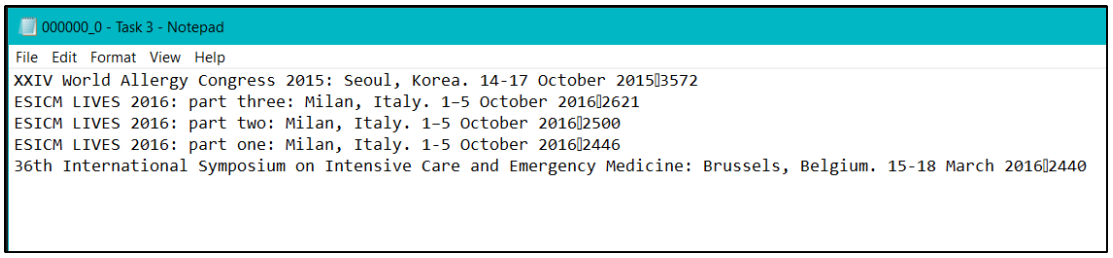


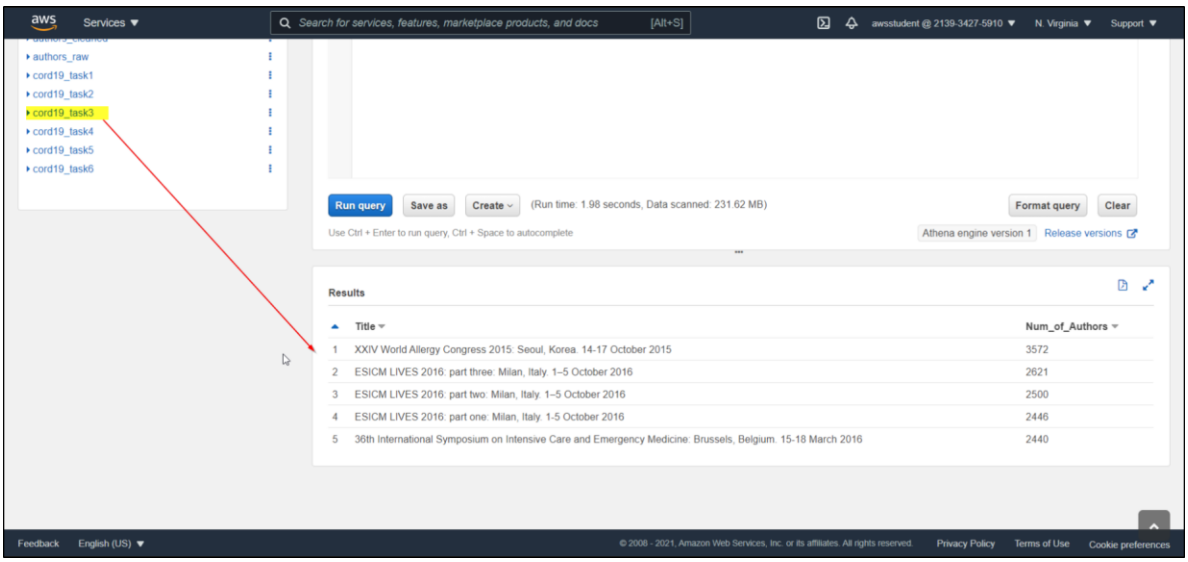
3.3.6 Result

```
+--------------------+--------------+
|               Title|Num_of_Authors|
+--------------------+--------------+
|XXIV World Allerg...|          3572|
|ESICM LIVES 2016:...|          2621|
|ESICM LIVES 2016:...|          2500|
|ESICM LIVES 2016:...|          2446|
|36th Internationa...|          2440|
+--------------------+--------------+
```

	Title	Number of Authors
1	XXIV World Allergy Congress 2015: Seoul, Korea. 14-17 October 2015	3572
2	ESICM LIVES 2016: part three: Milan, Italy. 1–5 October 2016	2621
3	ESICM LIVES 2016: part two: Milan, Italy. 1–5 October 2016	2500
4	ESICM LIVES 2016: part one: Milan, Italy. 1-5 October 2016	2446
5	36th International Symposium on Intensive Care and Emergency Medicine: Brussels, Belgium. 15-18 March 2016	2440

```
▸ (3) Spark Jobs
Out[57]: [['XXIV World Allergy Congress 2015: Seoul, Korea. 14-17 October 2015', 3572],
 ['ESICM LIVES 2016: part three: Milan, Italy. 1-5 October 2016', 2621],
 ['ESICM LIVES 2016: part two: Milan, Italy. 1-5 October 2016', 2500],
 ['ESICM LIVES 2016: part one: Milan, Italy. 1-5 October 2016', 2446],
 ['36th International Symposium on Intensive Care and Emergency Medicine: Brussels, Belgium. 15-18
March 2016',
  2440]]

Command took 39.35 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/8/2021, 7:11:38 PM on amjon2
```





3.3.7 Result Review

- ESICM LIVES 2016: part two and part three have gained more authors following part one from May.
- XXIV World Allergy Congress 2015 and 36th International Symposium journals have remained in the top 5, and the former ranking the first.

3.4: Task 4 – Top 5 authors and papers published

3.4.0 Outline

1. The authors column needs to be selected, split and exploded using a delimiter ';' (semicolon).
2. Then we filter out any authors that have digits in their names using regular expressions.
3. The resulted authors are now grouped and counted, and later sorted in descending order.

3.4.1 Assumptions made

- More often than not, people/authors don't have digits in their names, so this code was written to solve a general problem of removing numbers from names. Example – D039, O039, etc.

3.4.2 PySpark – Dataframes

- Generators are not supported when it's nested in expressions.
- RLIKE is a method used for pattern finding.

6.1 Using Dataframe created from CSV

Cmd 72

```
from pyspark.sql.functions import split, explode, trim

# Explode view is created here
df_notrim = df_cord.select(explode(split("authors", ";")).alias("Author"), "journal").persist()

# Filtering out any author that has numbers in their name
df_notrim = df_notrim.withColumn("Author_alpha", trim(df_notrim["Author"]).rlike("[0-9]")).persist()
df_notrim = df_notrim.filter(df_notrim["Author_alpha"]==False).persist()

# Performing Task 4 now
task4_df = df_notrim.withColumn("Author", trim(df_notrim["Author"])) \
                    .select("Author")  \
                    .groupBy("Author") \
                    .count()   \
                    .withColumnRenamed("count", "Num_of_Papers") \
                    .sort("Num_of_Papers", ascending=False)

task4_df.show(5)

```

6.2 Using Dataframe created from Temp View

Cmd 74

```
from pyspark.sql.functions import split, explode, trim

# Explode view is created here
df_notrim_temp = df_tempView.select(explode(split("authors", ";")).alias("Author"), "journal").persist()

# Filtering out any author that has numbers in their name
df_notrim_temp = df_notrim_temp.withColumn("Author_alpha", trim(df_notrim_temp["Author"]).rlike("[0-9]")).persist()
df_notrim_temp = df_notrim_temp.filter(df_notrim_temp["Author_alpha"]==False).persist()

# Performing Task 4 now
df_tempt4 = df_notrim_temp.withColumn("Author", trim(df_notrim_temp["Author"])) \
                    .select("Author")  \
                    .groupBy("Author") \
                    .count()   \
                    .withColumnRenamed("count", "Num_of_Papers") \
                    .sort("Num_of_Papers", ascending=False)

df_tempt4.show(5)

```

```
from pyspark.sql.functions import split, explode, trim

# Explode view is created here
df_notrim_parq = df_parq.select(explode(split("authors", ";")).alias("Author"), "journal").persist()

# Filtering out any author that has numbers in their name
df_notrim_parq = df_notrim_parq.withColumn("Author_alpha", trim(df_notrim_parq["Author"]).rlike("[0-9]")).persist()
df_notrim_parq = df_notrim_parq.filter(df_notrim_parq["Author_alpha"]==False).persist()

# Performing Task 4 now
df_parqt4 = df_notrim_parq.withColumn("Author", trim(df_notrim_parq["Author"])) \
                    .select("Author")  \
                    .groupBy("Author") \
                    .count()   \
                    .withColumnRenamed("count", "Num_of_Papers") \
                    .sort("Num_of_Papers", ascending=False)

df_parqt4.show(5)

```

3.4.3 HiveQL

6.1 Using Permanent Tables

Cmd 65

```
SELECT
  TRIM(full_list) AS `Name of the Author`,
  COUNT(*) AS `Number of Papers`
FROM
  ${hivevar:db_name}.${hivevar:metadata_table} LATERAL VIEW EXPLODE(SPLIT(authors, ';')) AS full_list
WHERE
 TRIM(full_list) NOT RLIKE '[0-9]'
GROUP BY
  `Name of the Author`
ORDER BY
  `Number of Papers` desc
LIMIT
  5
```

6.2 Using Temp View

Cmd 67

```
SELECT
  TRIM(full_list) AS `Name of the Author`,
  COUNT(*) AS `Number of Papers`
FROM
  temp_${hivevar:metadata_table} LATERAL VIEW EXPLODE(SPLIT(authors, ';')) AS full_list
WHERE
 TRIM(full_list) NOT RLIKE '[0-9]'
GROUP BY
  `Name of the Author`
ORDER BY
  `Number of Papers` desc
LIMIT
  5
```

6.3 Using Table from Parquet File

Cmd 69

```
SELECT
  TRIM(full_list) AS `Name of the Author`,
  COUNT(*) AS `Number of Papers`
FROM
  ${hivevar:db_name}.parq_${hivevar:metadata_table} LATERAL VIEW EXPLODE(SPLIT(authors, ';')) AS full_list
WHERE
 TRIM(full_list) NOT RLIKE '[0-9]'
GROUP BY
  `Name of the Author`
ORDER BY
  `Number of Papers` desc
LIMIT
  5
```

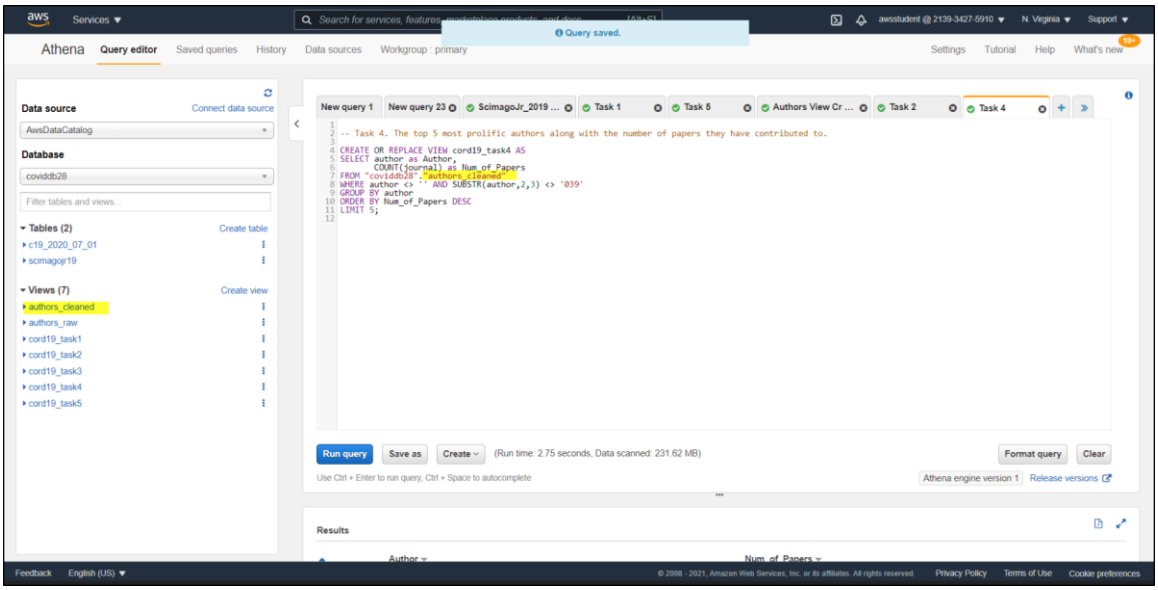
3.4.4 PySpark – RDD

- Re is the regular expressions library from python. Re.match() method does pattern matching.
- .cache() is used to cache the result in memory & .flatMap() is used to flatten the list of authors.
- .subtract() is used to remove the authors that have digits in their names from the final rdd.

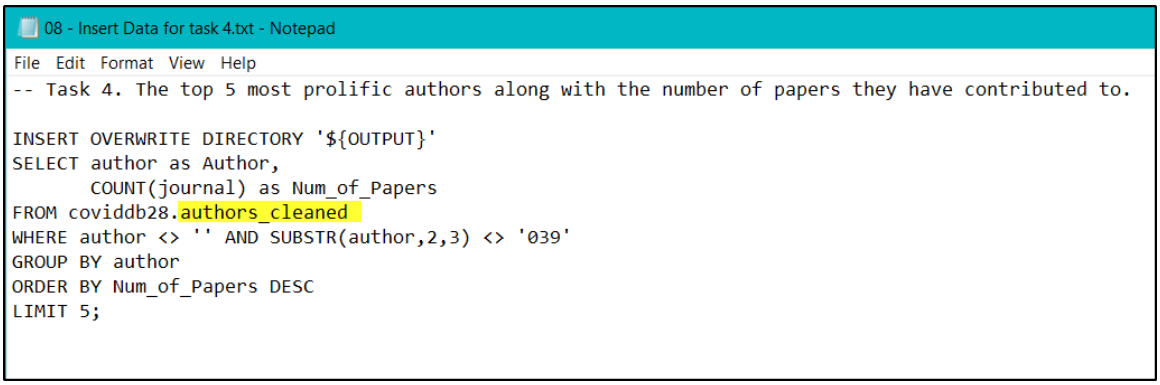
```
import re

task4_junk_rdd = cord19_rdd.filter(lambda row: row.authors is not None) \
                      .map(lambda row: row.authors.split(";")) \
                      .flatMap(lambda line: line) \
                      .filter(lambda line: re.match("^[a-zA-Z][0-9]", line)) \
                      .cache()

task4_rdd = cord19_rdd.filter(lambda row: row.authors is not None) \
                      .map(lambda row: row.authors.split(";")) \
                      .flatMap(lambda line: line) \
                      .cache()

task4_rdd = task4_rdd.subtract(task4_junk_rdd)  \
                     .map(lambda line: (line.strip(),1)) \
                     .reduceByKey(lambda v1,v2: v1+v2, 15) \
                     .sortBy(lambda line: line[1], ascending=False) \

task4_rdd.take(5)
```

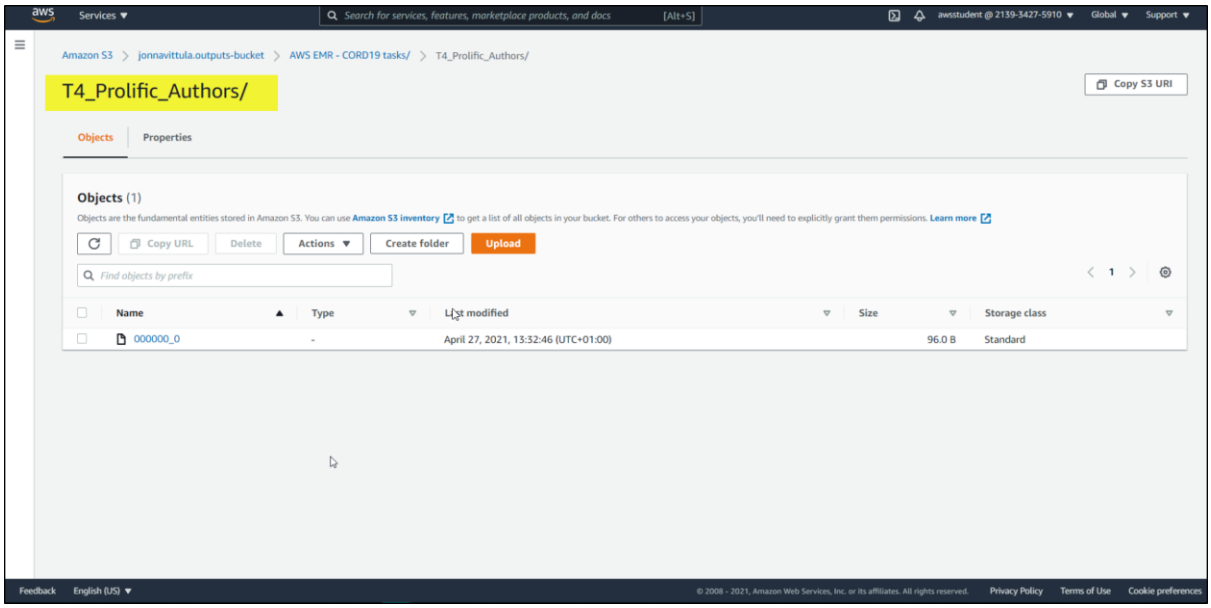
3.4.5 AWS

3.4.5.1 AWS S3 & Athena



3.4.5.2 AWS S3 & EMR



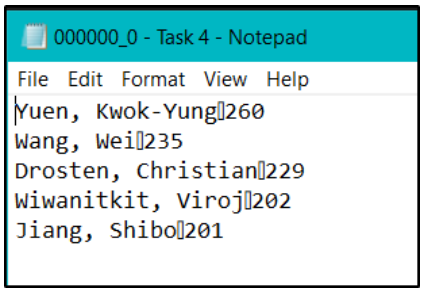


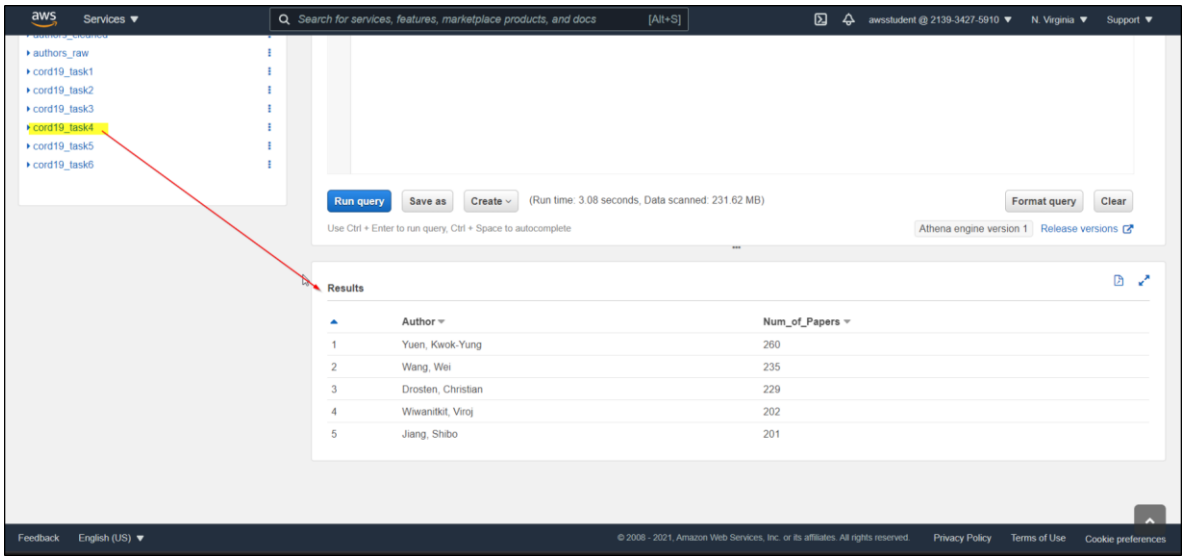
3.4.6 Result

```
+------------------+-------------+
|            Author|Num_of_Papers|
+------------------+-------------+
|   Yuen, Kwok-Yung|          260|
|         Wang, Wei|          235|
|Drosten, Christian|          229|
| Wiwanitkit, Viroj|          202|
|      Jiang, Shibo|          201|
+------------------+-------------+
```

	Name of the Author	Number of Papers
1	Yuen, Kwok-Yung	260
2	Wang, Wei	235
3	Drosten, Christian	229
4	Wiwanitkit, Viroj	202
5	Jiang, Shibo	201

```
▸ (3) Spark Jobs
Out[47]: [('Yuen, Kwok-Yung', 260),
 ('Wang, Wei', 235),
 ('Drosten, Christian', 229),
 ('Wiwanitkit, Viroj', 202),
 ('Jiang, Shibo', 201)]
```





3.4.7 Result Review

- Christian Drosten and Shibo Jiang remained to be on the top 5 authors who contributed to the maximum number of papers, with Kwok-Yung Yuen in the leading position.

3.5: Task 5 – Top 5 authors with H-index

3.5.0 Outline

1. An inner join is established between cord19 dataset (field = journal) and scimago dataset (field = Title) to select the list of authors and H-index.
2. The list of authors needs to be cleaned as we did in the previous step. So we use the same result which persisted in memory.
3. From the joined dataset, authors are grouped against the total H-index value, which is later sorted and aliased.

3.5.1 Assumptions made

- The cleaned list of authors in task 4 is used in task 5 to eliminate redundancy.

3.5.2 PySpark – Dataframes

- .join() is used to join 2 dataframes.
- H_index is a string, so in order to aggregate, it was casted into Integer type.

7.1 Using Dataframe created from CSV

Cmd 79

```
from pyspark.sql.functions import split, explode, trim

df_joined = df_notrim.join(df_scimagojr, df_notrim.journal == df_scimagojr.Title, "inner") \
                     .select(trim("Author").alias("Author"), "H_index") \
                     .persist()

task5_df = df_joined.withColumn("H_index", df_joined["H_index"].cast("Integer")) \
                     .groupBy("Author") \
                     .sum("H_index") \
                     .withColumnRenamed("sum(H_index)", "Total_H_Index") \
                     .sort("Total_H_Index", ascending=False)

task5_df.show(5)
```

7.2 Using Dataframe created from Temp View

Cmd 81

```
from pyspark.sql.functions import split, explode, trim

df_joined_temp = df_notrim_temp.join(df_scimagojr, df_notrim_temp.journal == df_scimagojr.Title, "inner") \
                    .select(trim("Author").alias("Author"), "H_index") \
                    .persist()

df_tempt5 = df_joined_temp.withColumn("H_index", df_joined_temp["H_index"].cast("Integer")) \
                    .groupBy("Author") \
                    .sum("H_index") \
                    .withColumnRenamed("sum(H_index)", "Total_H_Index") \
                    .sort("Total_H_Index", ascending=False)

df_tempt5.show(5)
```

7.3 Using Dataframe created from Parquet Files

Cmd 83

```
from pyspark.sql.functions import split, explode, trim

df_joined_parq = df_notrim_parq.join(df_scimagojr, df_notrim_parq.journal == df_scimagojr.Title, "inner") \
                    .select(trim("Author").alias("Author"), "H_index") \
                    .persist()

df_parqt5 = df_joined_parq.withColumn("H_index", df_joined_parq["H_index"].cast("Integer")) \
                    .groupBy("Author") \
                    .sum("H_index") \
                    .withColumnRenamed("sum(H_index)", "Total_H_Index") \
                    .sort("Total_H_Index", ascending=False)

df_parqt5.show(5)
```

3.5.3 HiveQL

7.1 Using Permanent Tables

Cmd 72

```
SELECT
  TRIM(author_list) AS `Name of the Author`,
  SUM(`H index`) AS `Total H-index`
FROM
  ${hivevar:db_name}.${hivevar:metadata_table} AS A
  INNER JOIN
  ${hivevar:db_name}.${hivevar:scimagojr_table} AS J
  ON A.journal = J.title
  LATERAL VIEW EXPLODE(SPLIT(authors, ';')) AS author_list
GROUP BY
  `Name of the Author`
ORDER BY
  `Total H-index` DESC
LIMIT
  5
```

7.2 Using Temp View

Cmd 74

```
SELECT
  TRIM(author_list) AS `Name of the Author`,
  SUM(`H index`) AS `Total H-index`
FROM
  temp_${hivevar:metadata_table} AS A
  INNER JOIN
  ${hivevar:db_name}.${hivevar:scimagojr_table} AS J
  ON A.journal = J.title
  LATERAL VIEW EXPLODE(SPLIT(authors, ';')) AS author_list
GROUP BY
  `Name of the Author`
ORDER BY
  `Total H-index` DESC
LIMIT
  5
```

7.3 Using Table from Parquet File

Cmd 76

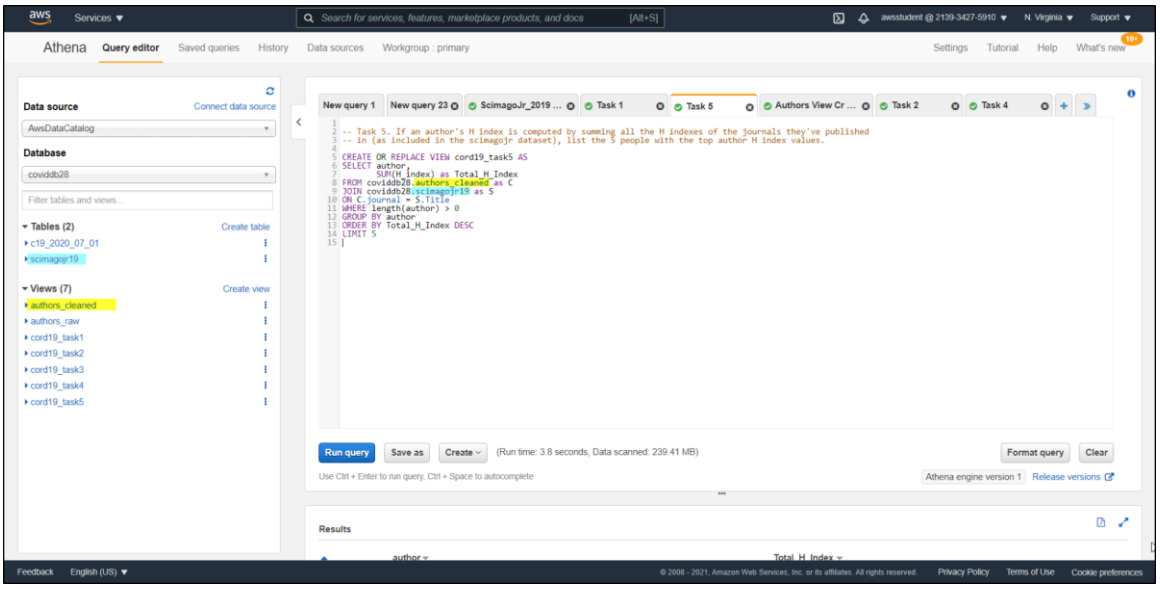
```
SELECT
  TRIM(author_list) AS `Name of the Author`,
  SUM(`H index`) AS `Total H-index`
FROM
  ${hivevar:db_name}.parq_${hivevar:metadata_table} AS A
  INNER JOIN
  ${hivevar:db_name}.${hivevar:scimagojr_table} AS J
  ON A.journal = J.title
  LATERAL VIEW EXPLODE(SPLIT(authors, ';')) AS author_list
GROUP BY
  `Name of the Author`
ORDER BY
  `Total H-index` DESC
LIMIT
  5
```

3.5.4 PySpark – RDD

```
authors_rdd = cord19_rdd.filter(lambda row: row.authors is not None) \
                        .filter(lambda row: row.journal is not None) \
                        .map(lambda row: (row.journal, row.authors))  \
                        .persist()

h_index_rdd = scimagojr_rdd.filter(lambda row: row.Title is not None) \
                           .filter(lambda row: row["H_index"] is not None) \
                           .map(lambda row: (row.Title, row["H_index"]))  \
                           .persist()

task5_rdd = authors_rdd.join(h_index_rdd)  \
                       .map(lambda line: (line[1][1], line[1][0].split(";")))   \
                       .flatMapValues(lambda line: line)       \
                       .map(lambda line: (line[1].strip(), int(line[0])))   \
                       .groupByKey()    \
                       .mapValues(lambda x : sum(x)) \
                       .sortBy(lambda line: line[1], ascending=False) \


task5_rdd.take(5)
```

3.5.5 AWS

3.5.5.1 AWS S3 & Athena

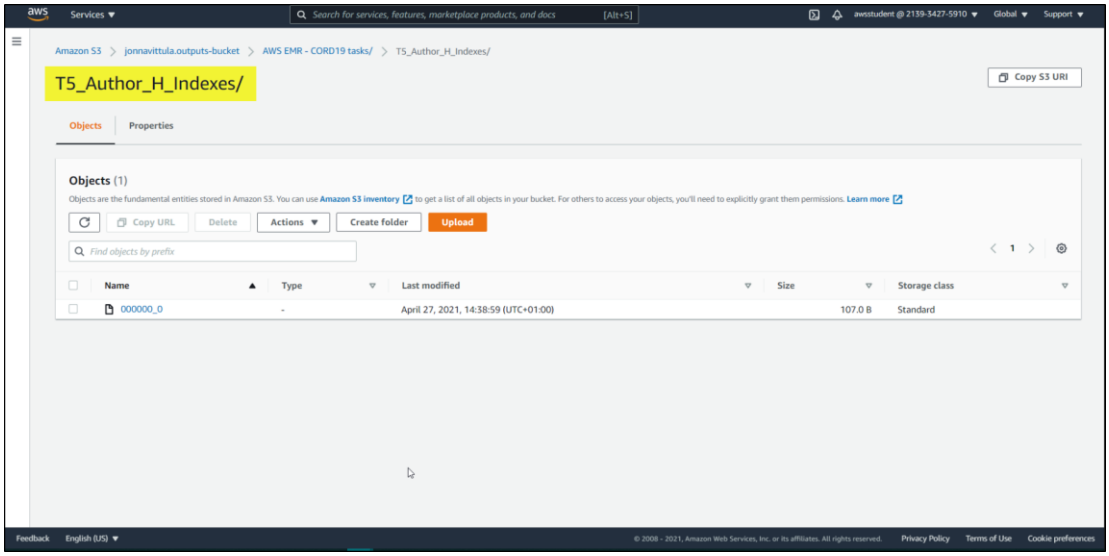


3.5.5.2 AWS S3 & EMR

09 - Insert Data for task 5.txt - Notepad

File Edit Format View Help

```
-- Task 5. If an author's H index is computed by summing all the H indexes of the journals they've published
-- in (as included in the scimagojr dataset), list the 5 people with the top author H index values.

INSERT OVERWRITE DIRECTORY '${OUTPUT}'
SELECT author,
       SUM(H_index) as Total_H_Index
FROM coviddb28.authors_cleaned as C
JOIN coviddb28.scimagojr19 as S
ON C.journal = S.Title
WHERE length(author) > 0
GROUP BY author
ORDER BY Total_H_Index DESC
LIMIT 5
```



3.5.6 Result

```
+------------------+-------------+
|            Author|Total_H_Index|
+------------------+-------------+
|  Cyranoski, David|        53314|
|Kupferschmidt, Kai|        52828|
|        Cohen, Jon|        50580|
|       Maxmen, Amy|        30134|
| Mallapaty, Smriti|        30134|
+------------------+-------------+
```

	Name of the Author	Total H-index
1	Cyranoski, David	53314
2	Kupferschmidt, Kai	52828
3	Cohen, Jon	50580
4	Maxmen, Amy	30134
5	Mallapaty, Smriti	30134

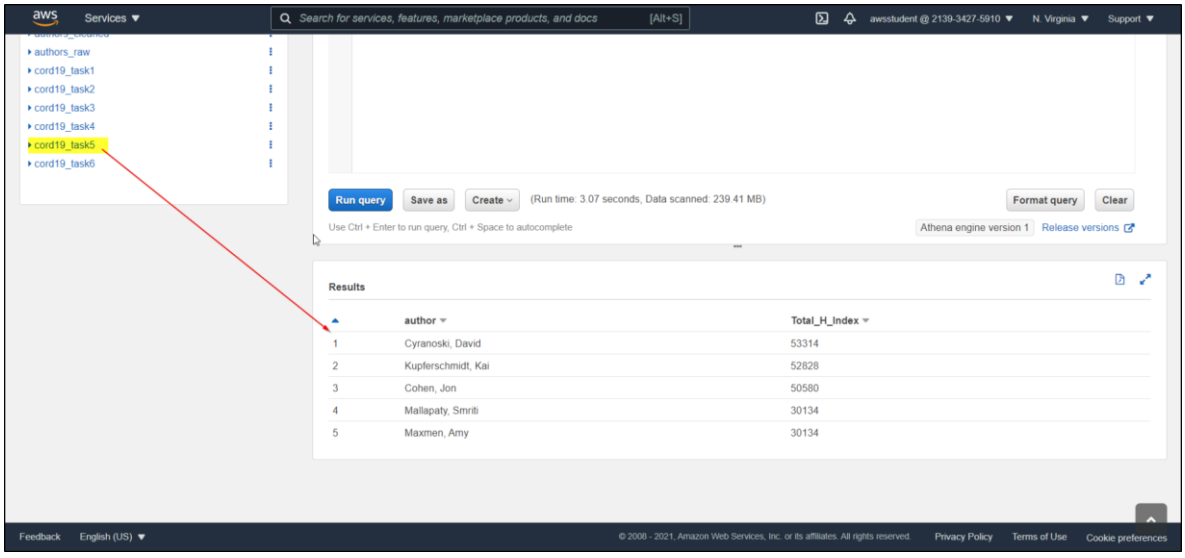
▸ (3) Spark Jobs

```
Out[50]: [('Cyranoski, David', 53314),
 ('Kupferschmidt, Kai', 52828),
 ('Cohen, Jon', 50580),
 ('Maxmen, Amy', 30134),
 ('Mallapaty, Smriti', 30134)]
```

000000_0 - Task 5 - Notepad

File Edit Format View Help

```
Cyranoski, David53314
Kupferschmidt, Kai52828
Cohen, Jon50580
Maxmen, Amy30134
Mallapaty, Smriti30134
```



3.5.7 Result Review

- All the top 5 authors until May are now replaced by new authors measured in terms of H-index.
- David Cyranoski's articles seem to have received the most citations over the period.

3.5.8 Further Analysis 2

- For the COVID-19 based journals, is there a correlation between the number of cites in the last 3 years and H-index?

7.4 Further Analysis 2 - Understanding Scimago Journal Rankings dataset (in relation to CORD19 data)

Question - For the COVID-19 based journals, is there a correlation between the number of cites in the last 3 years and H-index?

Linear Regression - H-index vs. Number of cites in the last 3 years since 2019

Libraries used - NumPy, Bokeh

Cmd 85

7.4.1 Dataset preparation

Cmd 86

```
df_joined = df_cord.join(df_scimagojr, df_cord.journal == df_scimagojr.Title, "inner") \
                   .select(trim("Region").alias("Region"),
                           trim("Country").alias("Country"),
                           trim("journal").alias("Journal"),
                           substring("publish_time",1,7).alias("Year_Month"),
                           "H_index",
                           "Total_Docs_"+scimago_year,
                           "Total_Docs_3years",
                           "Total_Refs",
                           "Total_Cites_3years",
                           "Citable_Docs_3years",
                           regexp_replace("Categories", "[(%Q%1%2%3%4%)]", "").alias("Categories"))  \
                   .where(substring(df_cord["publish_time"],5,1) == '-') \
                   .where(substring(df_cord["publish_time"],8,1) == '-') \
                   .persist()

df_joined.printSchema()
```

7.4.2 Correlation Coefficient & R-squared values are calculated

Cmd 88

```
import numpy as np

# Setting up lists for Data Visualization / Plotting

df_regr = df_joined.select("H_index","Total_Cites_3years")

x_dep = []
y_indep = []

for item in df_regr.rdd.collect():
    x_dep.append(item[0])
    y_indep.append(item[1])

correlation_matrix = np.corrcoef(x_dep, y_indep)
correlation_xy = correlation_matrix[0,1]
r_squared = correlation_xy**2

print("R-Squared value for this regression model: ", r_squared)
if r_squared > 0.8:
  print("Since R-squared value seems to be close to 1, we can proceed with plotting a graph for this correlation.")
```

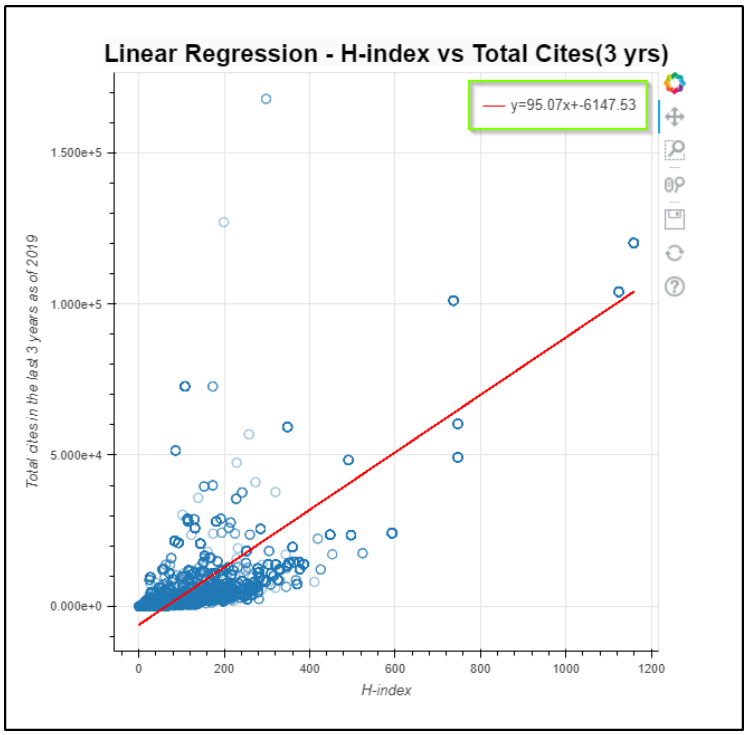
```
▸ (1) Spark Jobs
▸ df_regr: pyspark.sql.dataframe.DataFrame = [H_index: integer, Total_Cites_3years: integer]
R-Squared value for this regression model:  0.8650663993365079
Since R-squared value seems to be close to 1, we can proceed with plotting a graph for this correlation.
Command took 0.49 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/9/2021, 1:53:08 PM on amjon
```

7.4.3 Linear Regression plot

Cmd 90

```
import numpy as np
from bokeh.plotting import figure
from bokeh.io import show
from bokeh.embed import components, file_html
from bokeh.resources import CDN

#the data
x=x_dep
y=y_indep

# determine best fit line
par = np.polyfit(x, y, 1, full=True)
slope=par[0][0]
intercept=par[0][1]
y_predicted = [slope*i + intercept  for i in x]

# Legend
lgd = 'y='+str(np.round(slope,2))+'x+'+str(np.round(intercept,2))

# Plot definition
fig=figure(width=600, height=600, x_axis_label='H-index', y_axis_label='Total cites in the last 3 years as of'+scimago_year)
fig.circle(x,y, size=8, fill_alpha=0, alpha=0.5)
fig.line(x,y_predicted,color='red', legend_label = lgd)
show(fig)

# To configure visual properties on a plot's title attribute
fig.title.text = "Linear Regression - H-index vs Total Cites(3 yrs)"
fig.title.align = "center"
fig.title.text_color = "black"
fig.title.text_font_size = "22px"
fig.title.background_fill_color = "#fafafa"

# To create an html document that embeds the Bokeh plot
html2 = file_html(fig, CDN, "my plot2")

# To display this html
displayHTML(html2)
```

- Result -



- Result Review – There seems to be a linear relationship between the current H-index vs. number of citations from last 3 years. The slope seems to be steeper enough, so the rate of change is greater.

3.6: Task 6 – Number of papers since 2020-01

3.6.0 Outline

1. The substring of first 7 characters for publish_time is selected,
2. To remove jargon values, we select only values that have a hyphen in 5th and 8th characters.
3. We group by Year-Month, and sort in ascending order for the number of occurrences.
4. Arrays are used in matplotlib plot and a data dictionary is created for bokeh / seaborn plots.

3.6.1 Assumptions made

- The column 'publish_time' from covid19 dataset is considered as a string value.

3.6.2 PySpark – Dataframes

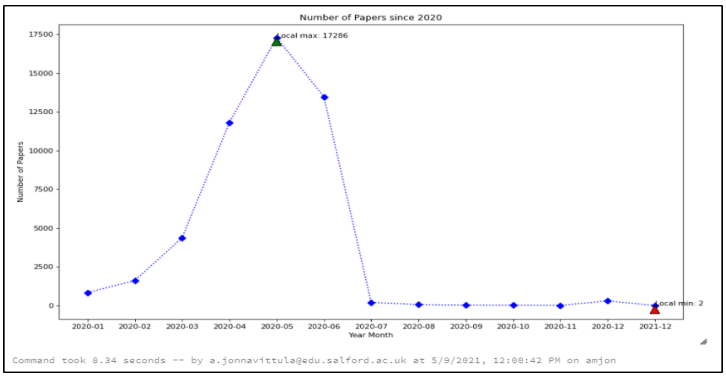
8.1 Using Dataframe created from CSV

Cmd 93

```
from pyspark.sql.functions import substring, length

task6_df = df_cord.select(substring("publish_time",1,7).alias("Year_Month"), "journal") \
                  .where(substring(df_cord["publish_time"],1,4) >= '2020') \
                  .where(substring(df_cord["publish_time"],5,1) == '-') \
                  .groupBy("Year_Month") \
                  .count() \
                  .sort("Year_Month", ascending=True)   \
                  .withColumnRenamed("count", "Freq")

task6_df.show()
```

- Max and min values are calculated for annotations.
- Matplotlib.pyplot -

```
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

figure(figsize=(14, 8), dpi=80)

names = []
values = []

for item in task6_df.rdd.collect():
    names.append(str(item[0]))
    values.append(item[1])

ymax = max(values)
xpos = values.index(ymax)
xmax = names[xpos]

ymin = min(values)
xpos_1 = values.index(ymin)
xmin = names[xpos_1]

plt.plot(names, values, label="Number of Papers", marker = 'D', color='b', linestyle='dotted')

plt.annotate('Local max: ' + str(ymax), xy=(xmax, ymax), xytext=(xmax, ymax+5),
             arrowprops=dict(facecolor='green'),
             )

plt.annotate('Local min: ' + str(ymin), xy=(xmin, ymin), xytext=(xmin, ymin+5),
             arrowprops=dict(facecolor='red'),
             )

plt.xlabel('Year Month')
plt.ylabel('Number of Papers')
plt.title('Number of Papers since 2020')

plt.show()
```



Command took 8.34 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/9/2021, 12:00:42 PM on amjon

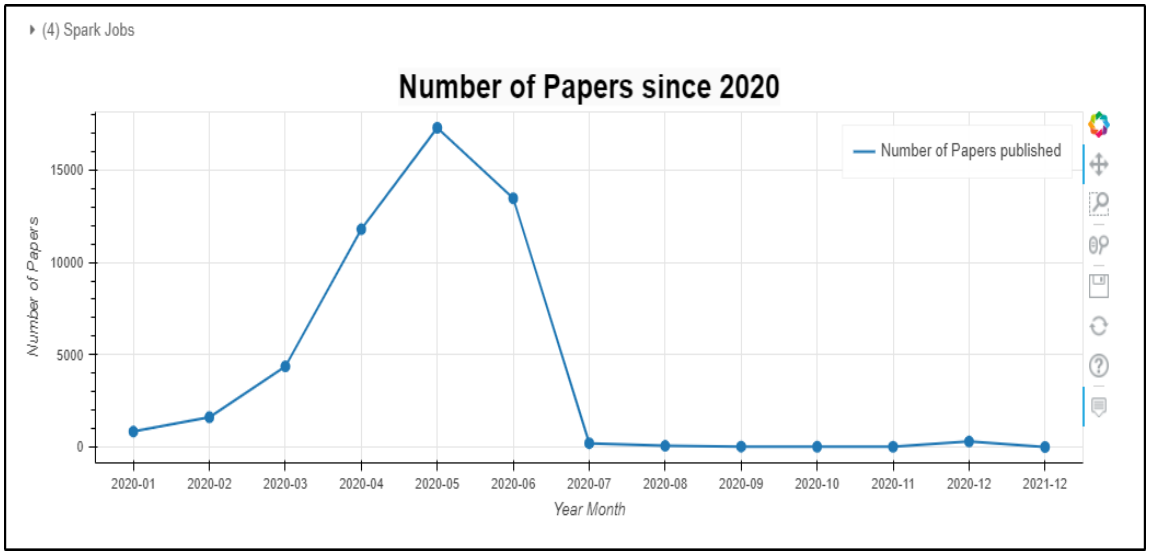
8.2 Using Dataframe created from Temp View

Cmd 97

```
from pyspark.sql.functions import substring, length

df_tempt6 = df_tempView.select(substring("publish_time",1,7).alias("Year_Month"), "journal") \
                       .where(substring(df_tempView["publish_time"],1,4) >= '2020') \
                       .where(substring(df_tempView["publish_time"],5,1) == '-') \
                       .groupBy("Year_Month") \
                       .count() \
                       .sort("Year_Month", ascending=True)   \
                       .withColumnRenamed("count", "Num_of_Papers")

df_tempt6.show()
```

- Bokeh plot -

```
from bokeh.plotting import figure, output_file
from bokeh.models import ColumnDataSource, HoverTool
from bokeh.embed import components, file_html
from bokeh.resources import CDN

names = []
values = []

for item in df_tempt6.rdd.collect():
    names.append(str(item[0]))
    values.append(item[1])

# Creating the Data Dictionary for X & Y (tooltips)
data = {"x": names, "y": values}

# Create a Figure() with necessary attributes
p = figure(x_range=names, plot_height=350, plot_width=1000,
           x_axis_label='Year Month', y_axis_label='Number of Papers', tooltips=None)

# To render the plot into a Line chart, we use a line()
p.line(names, values, legend_label="Number of Papers published", line_width=2)

# To render markers where the datapoints are present
p.circle(names, values, width=5)

# To add HoverTool which would enable hovering across the vertical line & give tooltips
p.add_tools(HoverTool(tooltips=[( 'Year Month',   '@x'),( 'Num of Papers', '@y' ),],
    # display a tooltip whenever the cursor is vertically in line with a glyph
    mode='vline'))

# To configure visual properties on a plot's title attribute
p.title.text = "Number of Papers since 2020"
p.title.align = "center"
p.title.text_color = "black"
p.title.text_font_size = "25px"
p.title.background_fill_color = "#fafafa"

# To create an html document that embeds the Bokeh plot
html = file_html(p, CDN, "my plot1")

# To display this html
displayHTML(html)
```



8.3 Using Dataframe created from Parquet File

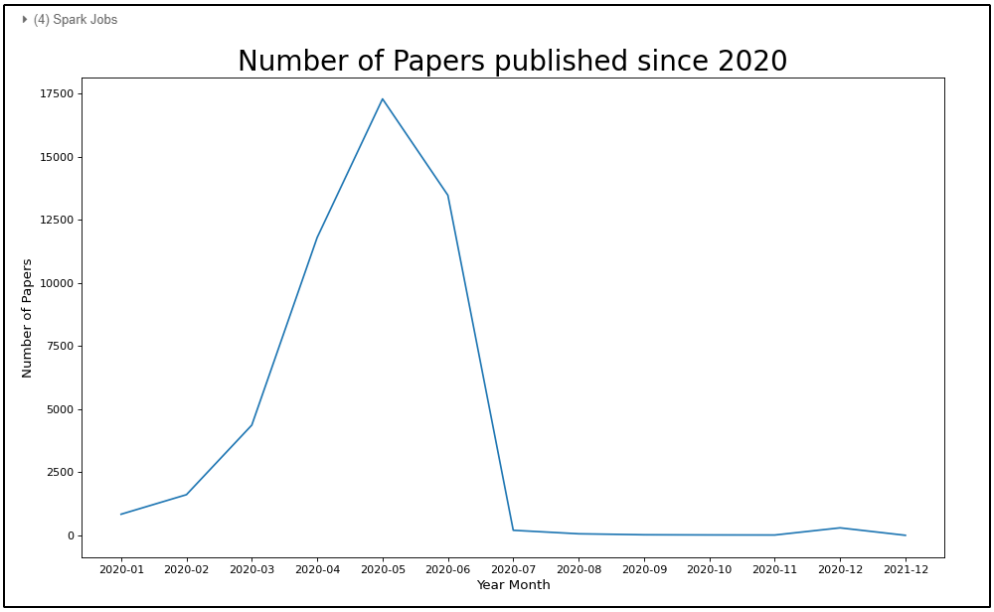
Cmd 101

```
from pyspark.sql.functions import substring, length

df_parqt6 = df_parq.select(substring("publish_time",1,7).alias("Year_Month"), "journal") \
                   .where(substring(df_parq["publish_time"],1,4) >= '2020') \
                   .where(substring(df_parq["publish_time"],5,1) == '-') \
                   .groupBy("Year_Month") \
                   .count() \
                   .sort("Year_Month", ascending=True)   \
                   .withColumnRenamed("count", "Num_of_Papers")

df_parqt6.show()
```

- Seaborn plot -

```
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

figure(figsize=(14, 8), dpi=80)

names = []
values = []

for item in df_parqt6.rdd.collect():
    names.append(str(item[0]))
    values.append(item[1])

# Creating the Data Dictionary for X & Y (tooltips)
data_plot = {
"x": names,
"y": values
}

sns.lineplot(x = 'x', y = 'y', data=data_plot, ci=True)
sns.set_style("darkgrid")

plt.xlabel("Year Month", fontsize= 12)
plt.ylabel("Number of Papers", fontsize= 12)
plt.title("Number of Papers published since 2020", fontsize= 25)

plt.show()
```

▸ (4) Spark Jobs



3.6.3 HiveQL

8.1 Using Permanent Tables

Cmd 79

```
SELECT
  SUBSTR(publish_time, 1, 7) as `Year Month`,
  COUNT(*) as `Number of papers published`
FROM
  ${hivevar:db_name}.${hivevar:metadata_table}
WHERE
  publish_time > '2020'
GROUP BY
  `Year Month`
ORDER BY
  `Year Month` ASC
```

8.2 Using Temp View

Cmd 81

```
SELECT
  SUBSTR(publish_time, 1, 7) as `Year Month`,
  COUNT(*) as `Number of papers published`
FROM
  temp_${hivevar:metadata_table}
WHERE
  publish_time > '2020'
GROUP BY
  `Year Month`
ORDER BY
  `Year Month` ASC
```

8.3 Using Table from Parquet File

Cmd 83

```
SELECT
  SUBSTR(publish_time, 1, 7) as `Year Month`,
  COUNT(*) as `Number of papers published`
FROM
  ${hivevar:db_name}.parq_${hivevar:metadata_table}
WHERE
  publish_time > '2020'
GROUP BY
  `Year Month`
ORDER BY
  `Year Month` ASC
```

3.6.4 PySpark – RDD

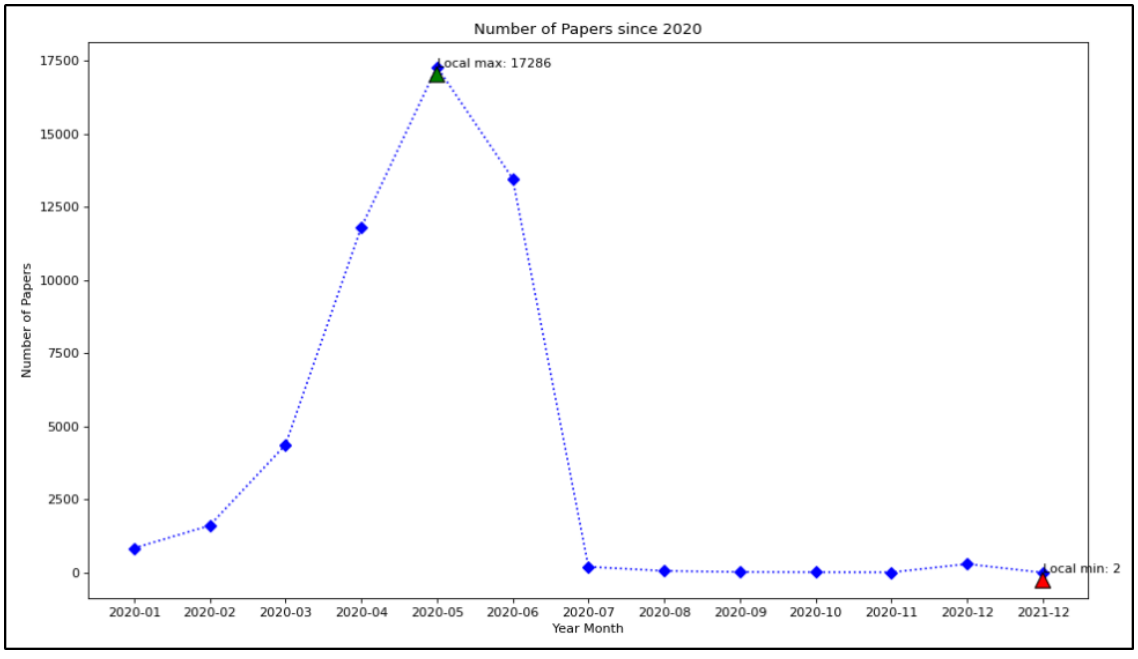
```
task6_rdd = cord19_rdd.filter(lambda row: row.publish_time is not None) \
                      .map(lambda row: (row.publish_time[0:7], 1)) \
                      .reduceByKey(lambda v1,v2: v1+v2) \
                      .filter(lambda line: line[0] > '2020' and line[0][4:5] == '-') \
                      .sortByKey(ascending=True)  \
                      .persist()

task6_rdd.take(30)

```

```
# Setting up fields for Data Visualization / Plotting

names = []
values = []

for item in task6_rdd.collect():
    names.append(str(item[0]))
    values.append(item[1])

ymax = max(values)
xpos = values.index(ymax)
xmax = names[xpos]

ymin = min(values)
xpos_1 = values.index(ymin)
xmin = names[xpos_1]

data = {
"names": names,
"values": values
}
```

8.1 Using Matplotlib to plot the datapoints

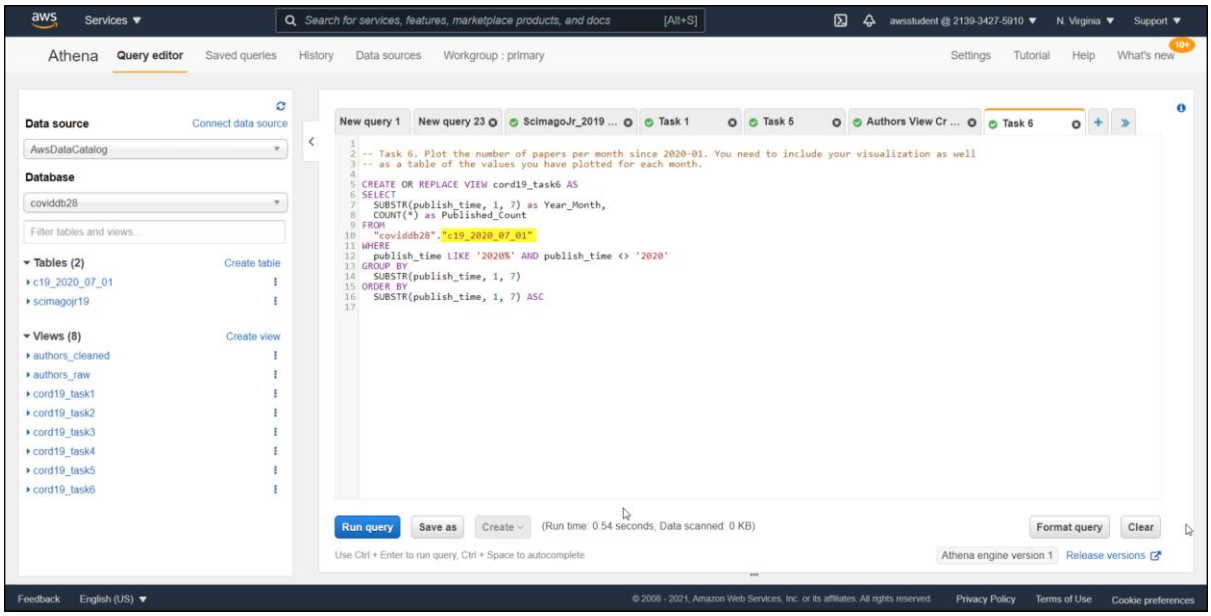
Cmd 57

```
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

figure(figsize=(14, 8), dpi=80)

plt.plot(names, values, label="Number of Papers", marker = 'D', color='b', linestyle='dotted')

plt.annotate('Local max: ' + str(ymax), xy=(xmax, ymax), xytext=(xmax, ymax+5),
             arrowprops=dict(facecolor='green'),
             )

plt.annotate('Local min: ' + str(ymin), xy=(xmin, ymin), xytext=(xmin, ymin+5),
             arrowprops=dict(facecolor='red'),
             )

plt.xlabel('Year Month')
plt.ylabel('Number of Papers')
plt.title('Number of Papers since 2020')

plt.show()
```



3.6.5 AWS

3.6.5.1 AWS S3 & Athena

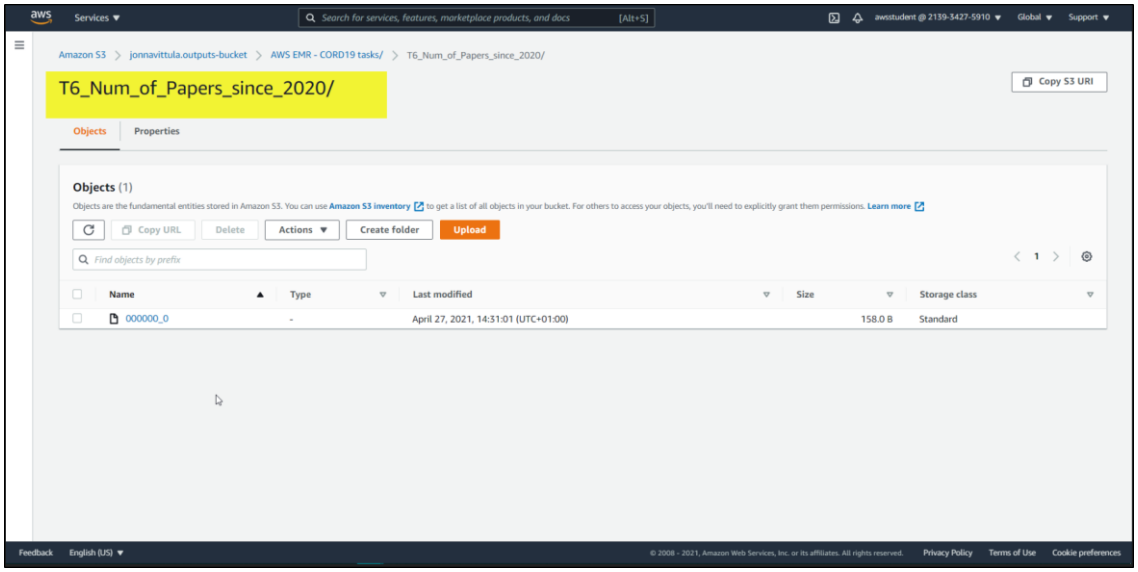


3.6.5.2 AWS S3 & EMR

10 - Insert Data for task 6.txt - Notepad

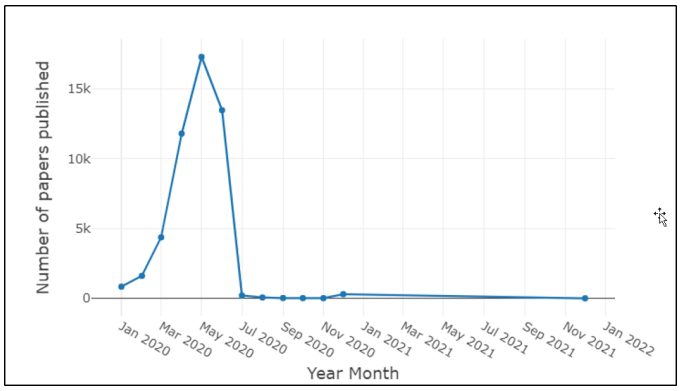
File Edit Format View Help

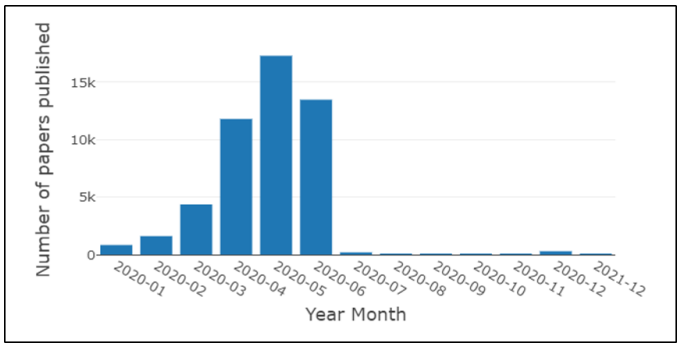
```
-- Task 6. Plot the number of papers per month since 2020-01. You need to include your visualization as well
-- as a table of the values you have plotted for each month.

INSERT OVERWRITE DIRECTORY '${OUTPUT}'
SELECT
  SUBSTR(publish_time, 1, 7) as Year_Month,
  COUNT(*) as Published_Count
FROM
  coviddb28.c19_2020_07_01
WHERE
  publish_time >= '2020-01-01' AND publish_time LIKE '%-%-%'
GROUP BY
  SUBSTR(publish_time, 1, 7)
ORDER BY
  Year_Month ASC
```

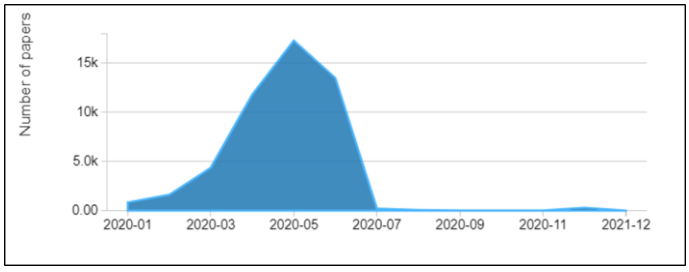


3.6.6 Result

Year_Month	Num_of_Papers
2020-01	837
2020-02	1611
2020-03	4367
2020-04	11796
2020-05	17286
2020-06	13470
2020-07	201
2020-08	62
2020-09	23
2020-10	16
2020-11	13
2020-12	297
2021-12	2

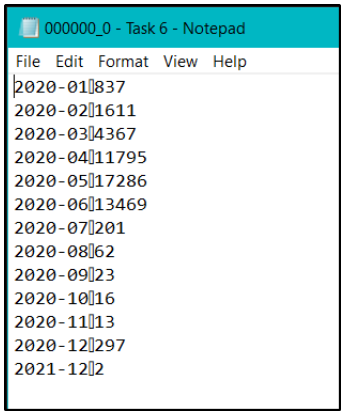


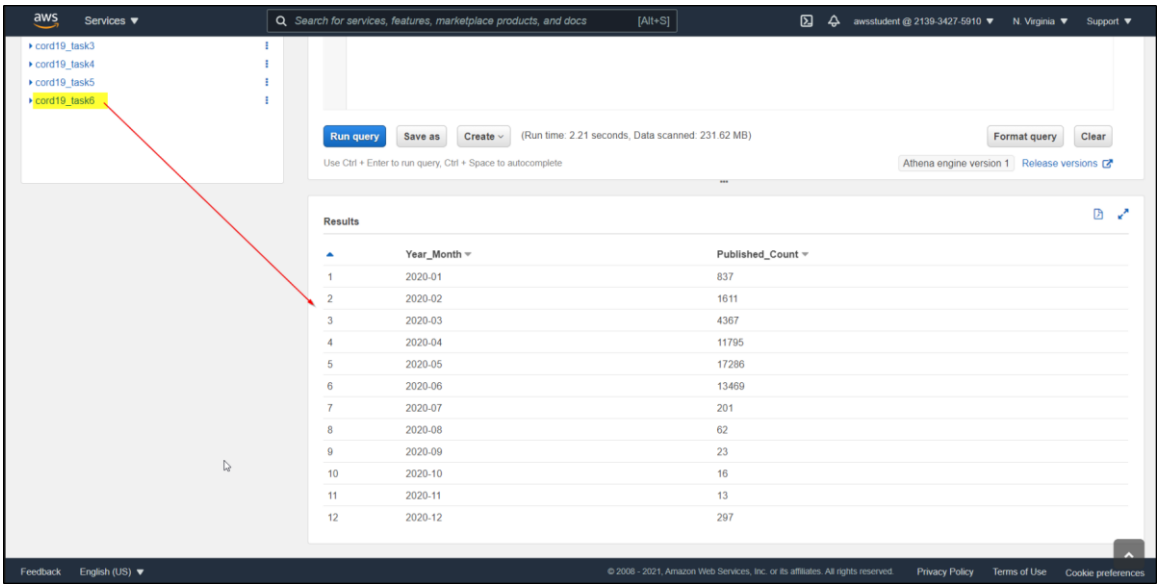




▸ (5) Spark Jobs

```
Out[22]: [('2020-01', 837),
 ('2020-02', 1611),
 ('2020-03', 4367),
 ('2020-04', 11796),
 ('2020-05', 17286),
 ('2020-06', 13470),
 ('2020-07', 201),
 ('2020-08', 62),
 ('2020-09', 23),
 ('2020-10', 16),
 ('2020-11', 13),
 ('2020-12', 297),
 ('2021-12', 2)]
```





3.6.7 Result Review

- The highest number of papers were published in May-2020 (17286) and the lowest were published in 2021-12 (2).
- This dataset is as of Jul-2020, hence we see the sudden plummeting of the graph after Jun-2020.

Section 4: Additional efforts

9.1 Built-in Map chart utilization

Download and use Countrywise Alpha-3 codes from GitHub into the Databricks notebook

Cmd 106

```
%sh
curl https://raw.githubusercontent.com/ambareesh-j/world_countries/master/data/en/countries.json > /tmp/countries.json
ls /tmp/countries.json
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 11684  100 11684    0     0  59612      0 --:--:-- --:--:-- --:--:-- 59612
/tmp/countries.json
```

Command took 0.29 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/9/2021, 12:00:42 PM on amjon

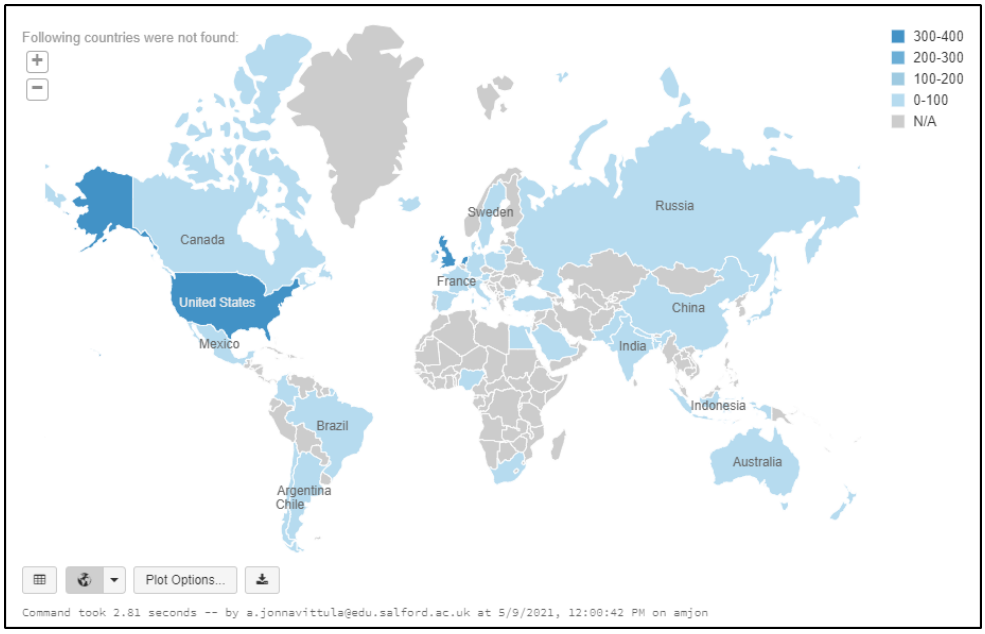
Cmd 107

```
# Moving from File:/ to DBFS:/
dbutils.fs.cp('file:/tmp/countries.json', filepath+"CountryKeys/countries.json")

dbutils.fs.ls(filepath+"CountryKeys/")
```

```
Out[45]: [FileInfo(path='dbfs:/FileStore/tables/CORD19/CountryKeys/countries.json', name='countries.json', size=11684)]
```

Command took 0.48 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/9/2021, 12:00:42 PM on amjon

```
from pyspark.sql.functions import regexp_replace, split, explode, trim

# Read JSON file into dataframe
df_ctry_raw = spark.read.option("multiline","true") \
                    .json(filepath+"CountryKeys/countries.json") \
                    .persist()

df_ctry = df_ctry_raw.select("name", "alpha3").persist()

df_ctry_jrs = df_joined.select("Region","Country", "Journal") \
                       .distinct() \
                       .groupBy("Region","Country") \
                       .count()  \
                       .withColumnRenamed("count", "Total Journals") \
                       .sort("Total Journals", ascending=False) \
                       .persist()

#Using the Country Key for Map charts to bring out Top 10 key players
df_ctry_key = df_ctry_jrs.join(df_ctry, df_ctry_jrs.Country == df_ctry.name, "left") \
                         .select("Country", "alpha3", "Total Journals") \
                         .sort("Total Journals", ascending=False)

# Drop from memory
df_ctry_raw = []

display(df_ctry_key)
```



Command took 2.81 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/9/2021, 12:00:42 PM on amjon

9.2 Pivot Table to see the categories by Region/Country

Cmd 110

```
1  df_cat_count = df_joined.select("Region", "Country", explode(split("Categories", ";")).alias("Category")) \
2                          .groupBy("Region", "Country", "Category") \
3                          .count()
4
5  display(df_cat_count)
```

▸ (10) Spark Jobs

▸ df_cat_count: pyspark.sql.dataframe.DataFrame = [Region: string, Country: string ... 2 more fields]

	Category	Western Europe	Northern America	Asiatic Region	Eastern Europe	Latin America
1	Infectious Diseases	1768	33	18	0	24
2	Virology	1738	212	33	0	0
3	Medicine miscellaneous	1528	317	398	5	61
4	Infectious Diseases	1226	275	4	0	5
5	Medicine miscellaneous	1094	496	75	0	0
6	Public Health, Environmental and Occupational Health	866	175	6	0	0
7	Microbiology medical	740	105	4	0	28

Aggregated (by sum) in the backend.

Plot Options...

Command took 6.83 seconds -- by a.jonnavittula@edu.salford.ac.uk at 5/9/2021, 12:00:42 PM on amjon