In this task, we will explore the use of Apache Spark on Databricks to analyse clinical trial data and pharmaceutical company information. The goal is to learn about different facets of clinical trials—such as study types, popular diseases, leading sponsors, and concluded studies.

The following datasets are available for analysis:

Clinical Trial Data: This dataset includes study types, conditions, sponsors, and completion statuses for individual clinical trials.

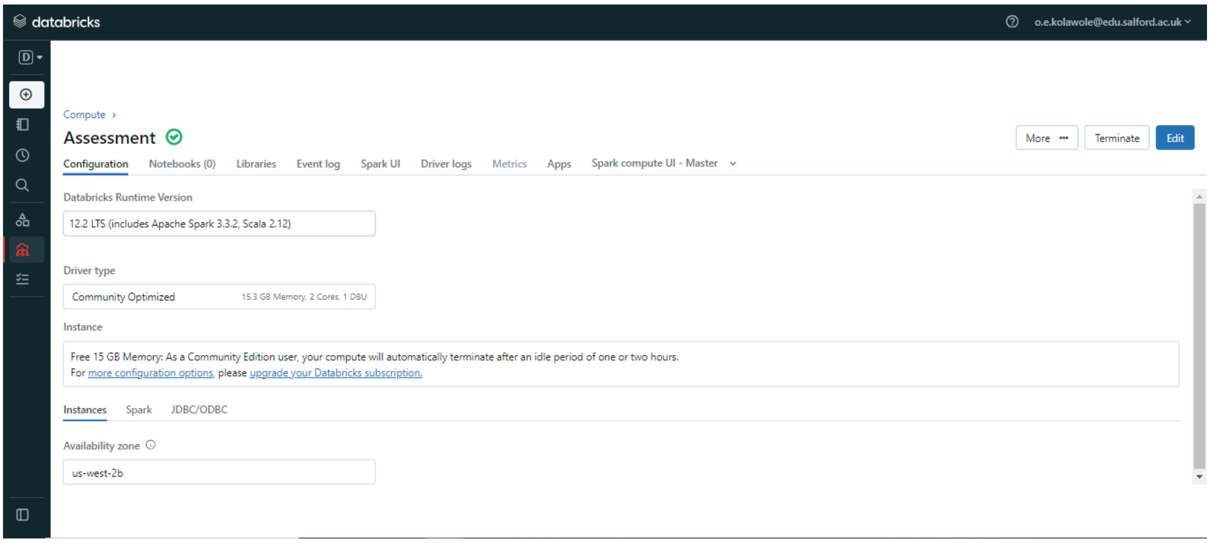
Pharmaceutical Company Information: This dataset offers details on pharmaceutical companies, including parent companies and violations.

DataFrames, Spark SQL, and Spark RDDs will be used in the analysis.

Our goal in performing these analyses is to offer insightful information about the clinical trial landscape and pharmaceutical involvement, information that will help guide future research initiatives and decision-making processes. We will guarantee data accuracy throughout the analysis, address any possible inconsistencies, and clearly present the findings.

SETUP

Using the Databricks community edition and the 12.2 LTS runtime version, the task was accomplished. The "Assessment" compute, which is displayed below, was utilised.



Assessment

Databricks Runtime Version: 12.2 LTS (includes Apache Spark 3.3.2, Scala 2.12)

Driver type: Community Optimized 15.3 GB Memory, 2 Cores, 1 DBU

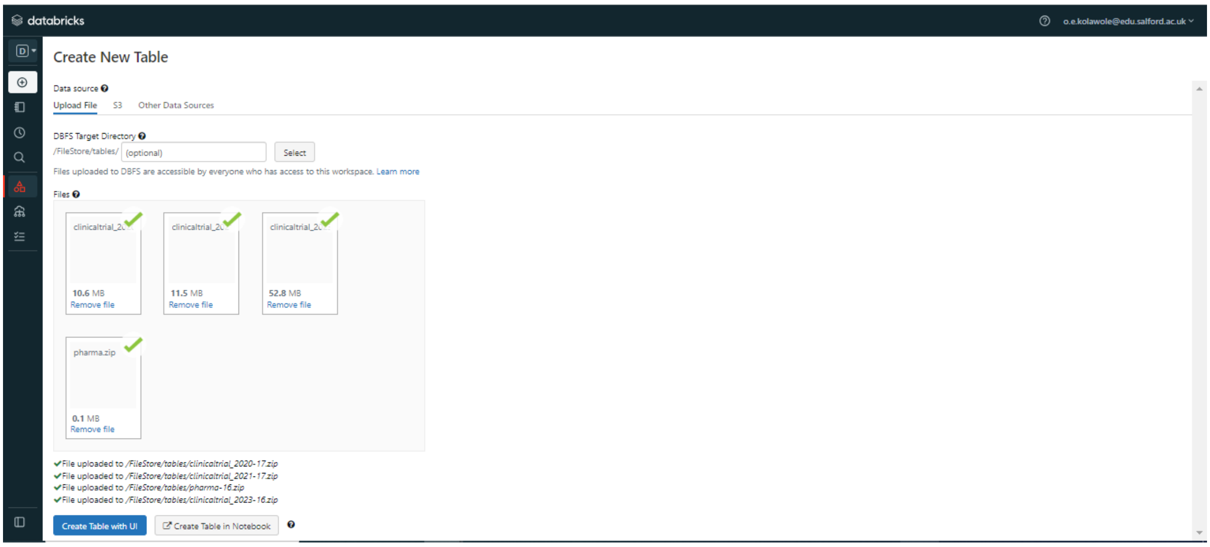
Instance

Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours.
For more configuration options, please upgrade your Databricks subscription.

Availability zone: us-west-2b

Fig 1.0: Compute

The zip files were uploaded in the FileStore/tables directory. A screenshot of this process is shown below.



Create New Table

Data source
Upload File S3 Other Data Sources

DBFS Target Directory
/FileStore/tables/ (optional) Select

Files uploaded to DBFS are accessible by everyone who has access to this workspace. Learn more

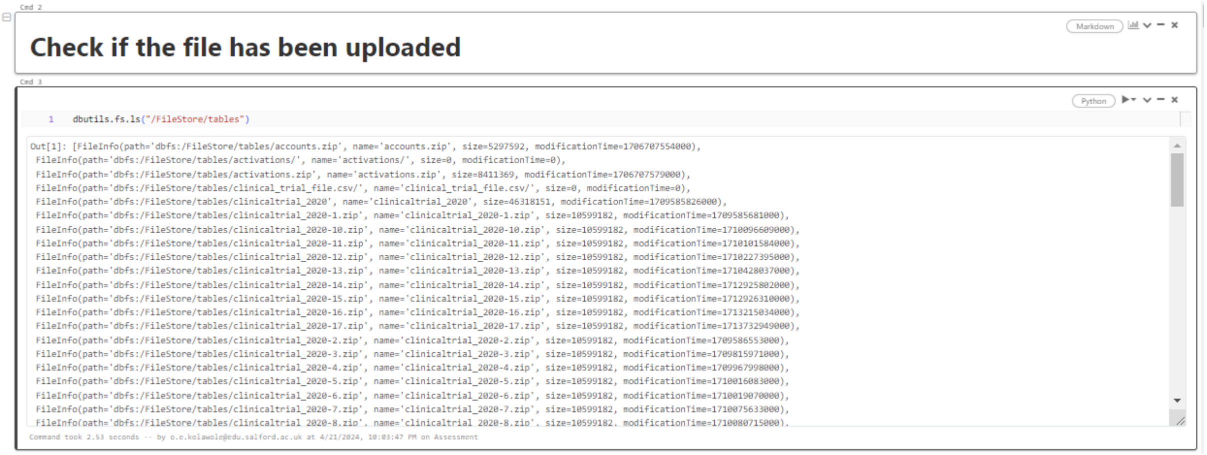
Files

clinicaltrial_20 — 10.6 MB — Remove file
clinicaltrial_20 — 11.5 MB — Remove file
clinicaltrial_20 — 52.8 MB — Remove file
pharma.zip — 0.1 MB — Remove file

✓File uploaded to */FileStore/tables/clinicaltrial_2020-17.zip*
✓File uploaded to */FileStore/tables/clinicaltrial_2021-17.zip*
✓File uploaded to */FileStore/tables/pharma-16.zip*
✓File uploaded to */FileStore/tables/clinicaltrial_2023-16.zip*

Fig 1.1: Dataset uploading

Checking if the file has been uploaded is done below



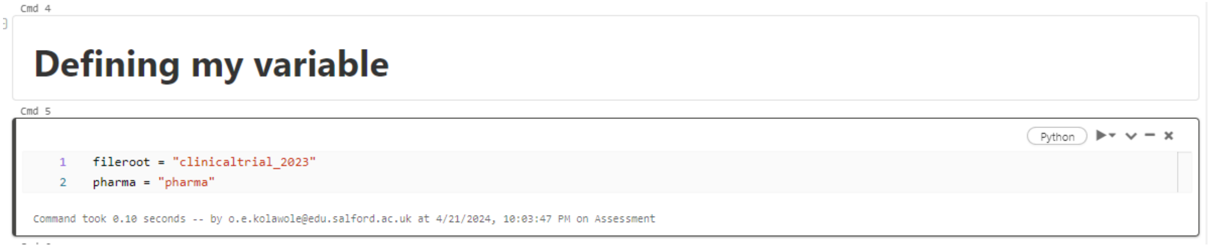
Check if the file has been uploaded

```
dbutils.fs.ls("/FileStore/tables")
```

```
Out[1]: [FileInfo(path='dbfs:/FileStore/tables/accounts.zip', name='accounts.zip', size=5297592, modificationTime=1706707554000),
 FileInfo(path='dbfs:/FileStore/tables/activations/', name='activations/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/FileStore/tables/activations.zip', name='activations.zip', size=8411369, modificationTime=1706707579000),
 FileInfo(path='dbfs:/FileStore/tables/clinical_trial_file.csv/', name='clinical_trial_file.csv/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020', name='clinicaltrial_2020', size=46318151, modificationTime=1709585826000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-1.zip', name='clinicaltrial_2020-1.zip', size=10599182, modificationTime=1709585681000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-10.zip', name='clinicaltrial_2020-10.zip', size=10599182, modificationTime=1710096609000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-11.zip', name='clinicaltrial_2020-11.zip', size=10599182, modificationTime=1710101584000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-12.zip', name='clinicaltrial_2020-12.zip', size=10599182, modificationTime=1710227395000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-13.zip', name='clinicaltrial_2020-13.zip', size=10599182, modificationTime=1710428037000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-14.zip', name='clinicaltrial_2020-14.zip', size=10599182, modificationTime=1712925802000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-15.zip', name='clinicaltrial_2020-15.zip', size=10599182, modificationTime=1712926310000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-16.zip', name='clinicaltrial_2020-16.zip', size=10599182, modificationTime=1713215034000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-17.zip', name='clinicaltrial_2020-17.zip', size=10599182, modificationTime=1713732949000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-2.zip', name='clinicaltrial_2020-2.zip', size=10599182, modificationTime=1709586553000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-3.zip', name='clinicaltrial_2020-3.zip', size=10599182, modificationTime=1709815971000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-4.zip', name='clinicaltrial_2020-4.zip', size=10599182, modificationTime=1709967998000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-5.zip', name='clinicaltrial_2020-5.zip', size=10599182, modificationTime=1710016083000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-6.zip', name='clinicaltrial_2020-6.zip', size=10599182, modificationTime=1710019070000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-7.zip', name='clinicaltrial_2020-7.zip', size=10599182, modificationTime=1710075633000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020-8.zip', name='clinicaltrial_2020-8.zip', size=10599182, modificationTime=1710080715000),
Command took 2.53 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 1.2

Inserting a reusable variable named ‘fileroot’ for the clinical trial data and ‘pharma’ for the pharma file



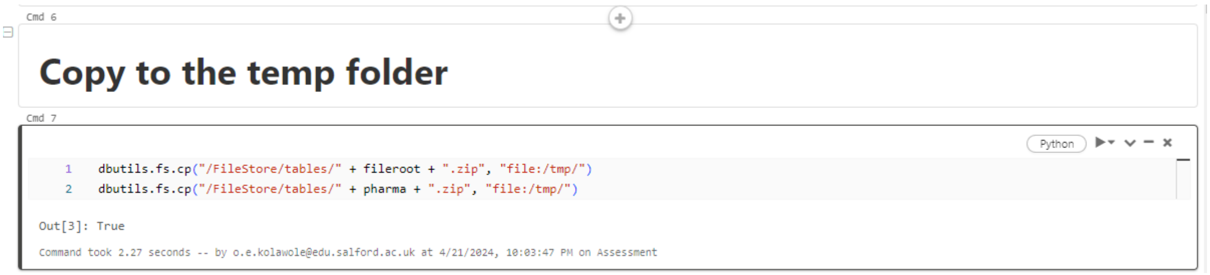
Defining my variable

```
fileroot = "clinicaltrial_2023"
pharma = "pharma"
```

```
Command took 0.10 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 1.3 Variable naming

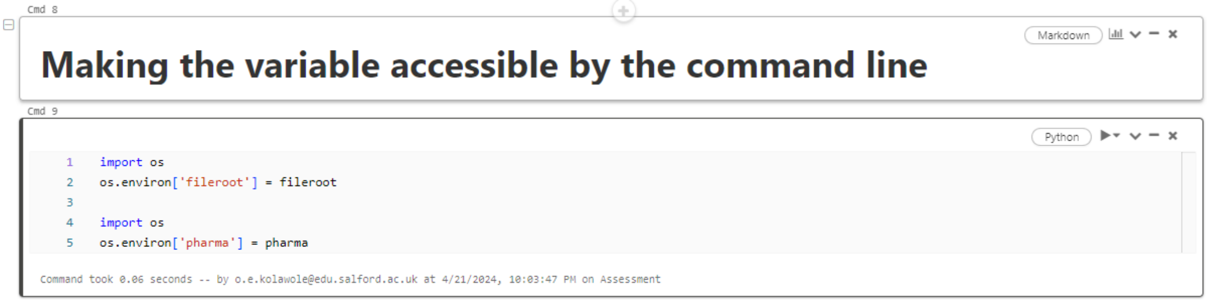
copying the zipped files to the tmp directory on my driver node



Copy to the temp folder

```
dbutils.fs.cp("/FileStore/tables/" + fileroot + ".zip", "file:/tmp/")
dbutils.fs.cp("/FileStore/tables/" + pharma + ".zip", "file:/tmp/")

Out[3]: True
Command took 2.27 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 1.4

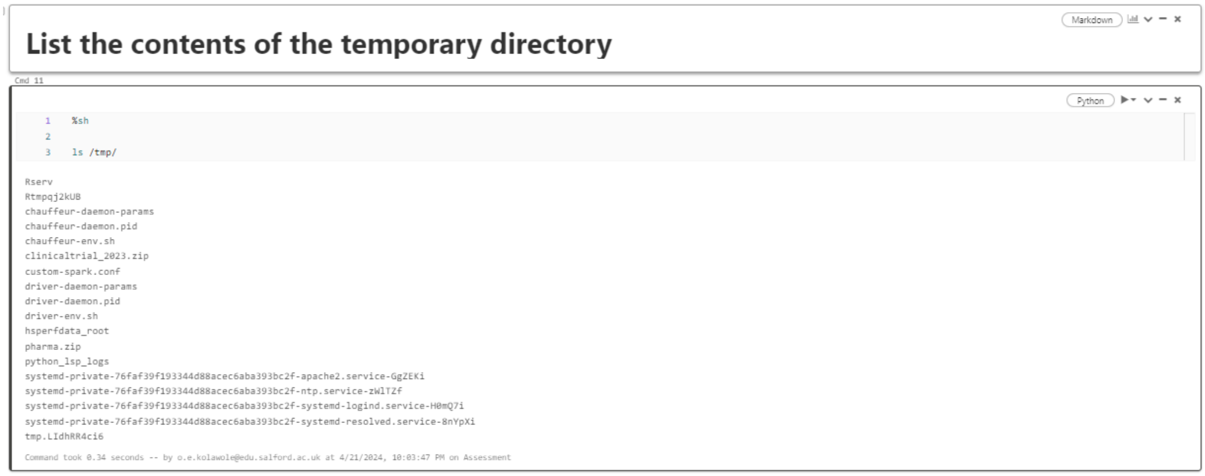


Making the variable accessible by the command line

```
import os
os.environ['fileroot'] = fileroot

import os
os.environ['pharma'] = pharma

Command took 0.06 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 1.5

Check the tmp directory

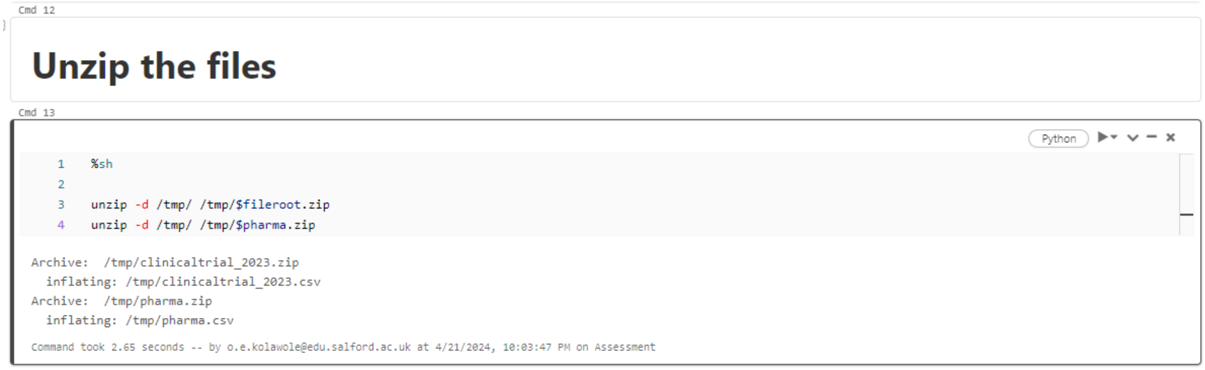


List the contents of the temporary directory

```
%sh

ls /tmp/

Rserv
Rtmpqj2kUB
chauffeur-daemon-params
chauffeur-daemon.pid
chauffeur-env.sh
clinicaltrial_2023.zip
custom-spark.conf
driver-daemon-params
driver-daemon.pid
driver-env.sh
hsperfdata_root
pharma.zip
python_lsp_logs
systemd-private-76faf39f193344d88acec6aba393bc2f-apache2.service-GgZEKi
systemd-private-76faf39f193344d88acec6aba393bc2f-ntp.service-zWlTZf
systemd-private-76faf39f193344d88acec6aba393bc2f-systemd-logind.service-H0mQ7i
systemd-private-76faf39f193344d88acec6aba393bc2f-systemd-resolved.service-8nYpXi
tmp.LIdhRR4ci6
Command took 0.34 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 1.6

Unzipping the zip files into the tmp directory

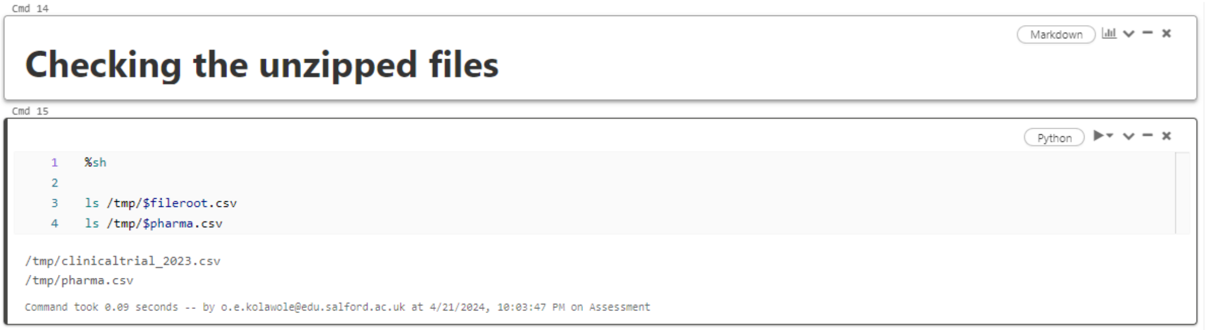


Unzip the files

```
%sh

unzip -d /tmp/ /tmp/$fileroot.zip
unzip -d /tmp/ /tmp/$pharma.zip
```

```
Archive:  /tmp/clinicaltrial_2023.zip
  inflating: /tmp/clinicaltrial_2023.csv
Archive:  /tmp/pharma.zip
  inflating: /tmp/pharma.csv
Command took 2.65 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 1.7



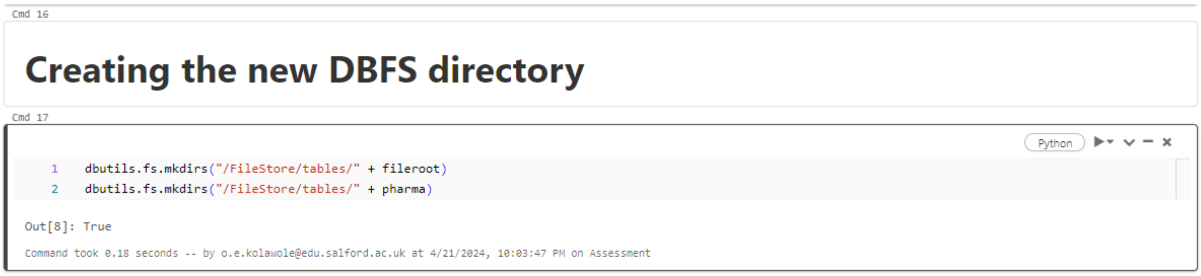
Checking the unzipped files

```
%sh

ls /tmp/$fileroot.csv
ls /tmp/$pharma.csv
```

```
/tmp/clinicaltrial_2023.csv
/tmp/pharma.csv
Command took 0.09 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 1.8

Create a DBFS directory for the files



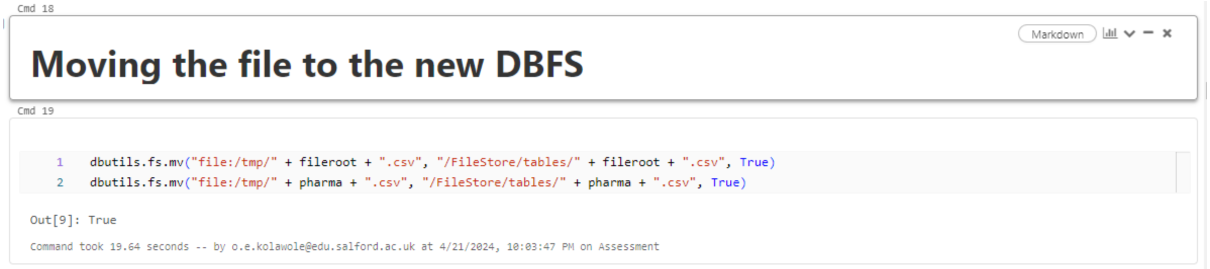
Creating the new DBFS directory

```
dbutils.fs.mkdirs("/FileStore/tables/" + fileroot)
dbutils.fs.mkdirs("/FileStore/tables/" + pharma)
```

```
Out[8]: True
Command took 0.18 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 1.9

Moving the unzipped CSV files from the tmp directory to the DBFS directory



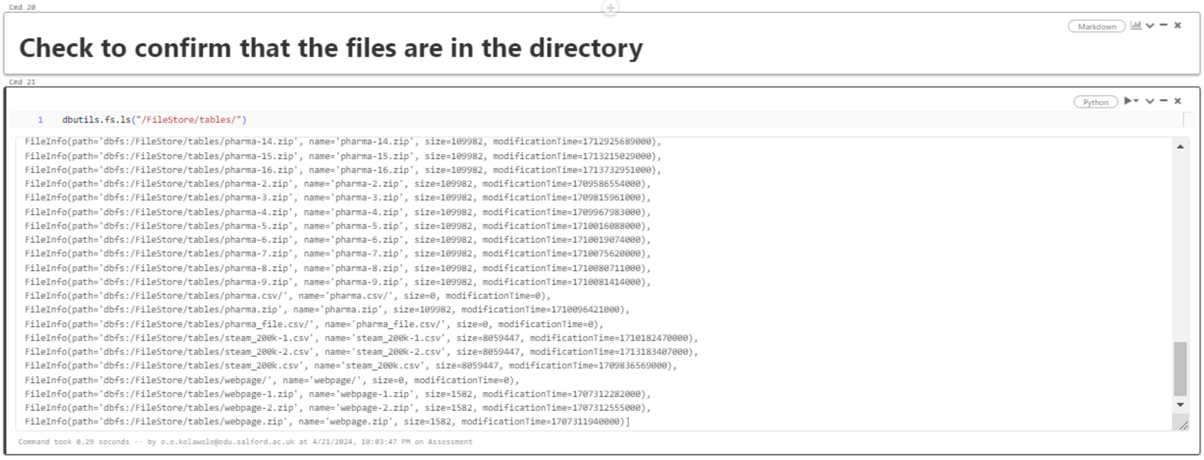
Moving the file to the new DBFS

```
dbutils.fs.mv("file:/tmp/" + fileroot + ".csv", "/FileStore/tables/" + fileroot + ".csv", True)
dbutils.fs.mv("file:/tmp/" + pharma + ".csv", "/FileStore/tables/" + pharma + ".csv", True)
```

```
Out[9]: True
Command took 19.64 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 2.0

Confirming if the CSV files have been moved to the filestore/table



Cmd 20

Check to confirm that the files are in the directory

Cmd 21

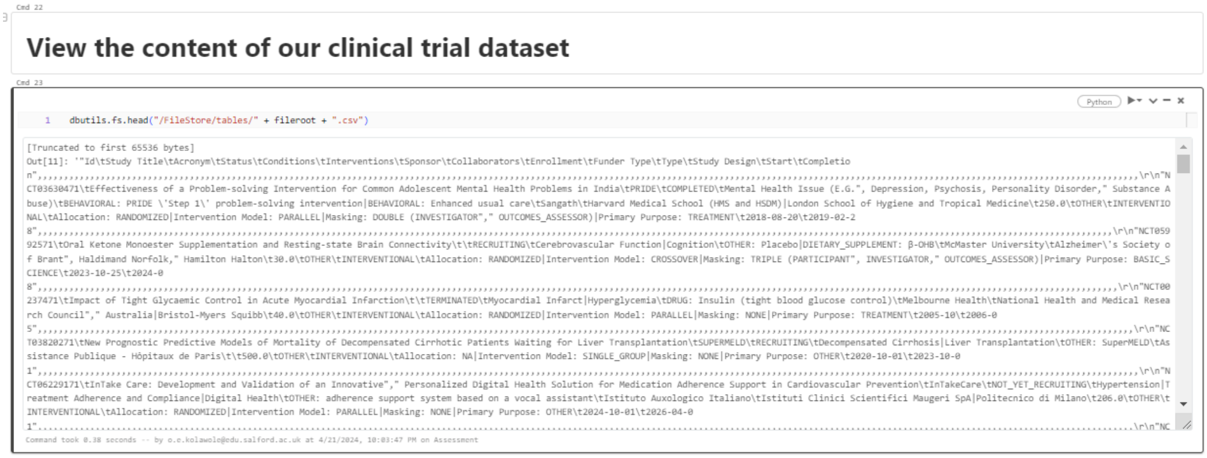
```
dbutils.fs.ls("/FileStore/tables/")
```

```
FileInfo(path='dbfs:/FileStore/tables/pharma-14.zip', name='pharma-14.zip', size=109982, modificationTime=1712925689000),
FileInfo(path='dbfs:/FileStore/tables/pharma-15.zip', name='pharma-15.zip', size=109982, modificationTime=1713215029000),
FileInfo(path='dbfs:/FileStore/tables/pharma-16.zip', name='pharma-16.zip', size=109982, modificationTime=1713732951000),
FileInfo(path='dbfs:/FileStore/tables/pharma-2.zip', name='pharma-2.zip', size=109982, modificationTime=1709586554000),
FileInfo(path='dbfs:/FileStore/tables/pharma-3.zip', name='pharma-3.zip', size=109982, modificationTime=1709815961000),
FileInfo(path='dbfs:/FileStore/tables/pharma-4.zip', name='pharma-4.zip', size=109982, modificationTime=1709967983000),
FileInfo(path='dbfs:/FileStore/tables/pharma-5.zip', name='pharma-5.zip', size=109982, modificationTime=1710016088000),
FileInfo(path='dbfs:/FileStore/tables/pharma-6.zip', name='pharma-6.zip', size=109982, modificationTime=1710019074000),
FileInfo(path='dbfs:/FileStore/tables/pharma-7.zip', name='pharma-7.zip', size=109982, modificationTime=1710075620000),
FileInfo(path='dbfs:/FileStore/tables/pharma-8.zip', name='pharma-8.zip', size=109982, modificationTime=1710080711000),
FileInfo(path='dbfs:/FileStore/tables/pharma-9.zip', name='pharma-9.zip', size=109982, modificationTime=1710081414000),
FileInfo(path='dbfs:/FileStore/tables/pharma.csv/', name='pharma.csv/', size=0, modificationTime=0),
FileInfo(path='dbfs:/FileStore/tables/pharma.zip', name='pharma.zip', size=109982, modificationTime=1710096421000),
FileInfo(path='dbfs:/FileStore/tables/pharma_file.csv/', name='pharma_file.csv/', size=0, modificationTime=0),
FileInfo(path='dbfs:/FileStore/tables/steam_200k-1.csv', name='steam_200k-1.csv', size=8059447, modificationTime=1710182470000),
FileInfo(path='dbfs:/FileStore/tables/steam_200k-2.csv', name='steam_200k-2.csv', size=8059447, modificationTime=1713183407000),
FileInfo(path='dbfs:/FileStore/tables/steam_200k.csv', name='steam_200k.csv', size=8059447, modificationTime=1709836569000),
FileInfo(path='dbfs:/FileStore/tables/webpage/', name='webpage/', size=0, modificationTime=0),
FileInfo(path='dbfs:/FileStore/tables/webpage-1.zip', name='webpage-1.zip', size=1582, modificationTime=1707312282000),
FileInfo(path='dbfs:/FileStore/tables/webpage-2.zip', name='webpage-2.zip', size=1582, modificationTime=1707312555000),
FileInfo(path='dbfs:/FileStore/tables/webpage.zip', name='webpage.zip', size=1582, modificationTime=1707311940000)]
```

Command took 0.29 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment

Fig 2.1

Checking the first few rows of the clinical trial dataset



Cmd 22

View the content of our clinical trial dataset

Cmd 23

```
dbutils.fs.head("/FileStore/tables/" + fileroot + ".csv")
```

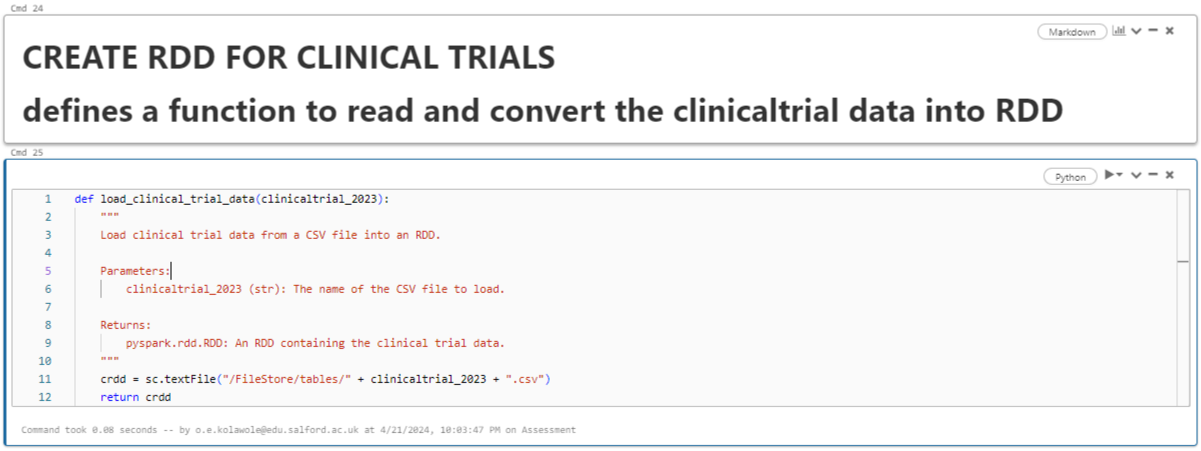
```
[Truncated to first 65536 bytes]
Out[11]: '"Id\tStudy Title\tAcronym\tStatus\tConditions\tInterventions\tSponsor\tCollaborators\tEnrollment\tFunder Type\tType\tStudy Design\tStart\tCompletio
n",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"N
CT03630471\tEffectiveness of a Problem-solving Intervention for Common Adolescent Mental Health Problems in India\tPRIDE\tCOMPLETED\tMental Health Issue (E.G.", Depression, Psychosis, Personality Disorder," Substance A
buse)\tBEHAVIORAL: PRIDE \'Step 1\' problem-solving intervention|BEHAVIORAL: Enhanced usual care\tSangath\tHarvard Medical School (HMS and HSDM)|London School of Hygiene and Tropical Medicine\t250.0\tOTHER\tINTERVENTIO
NAL\tAllocation: RANDOMIZED|Intervention Model: PARALLEL|Masking: DOUBLE (INVESTIGATOR"," OUTCOMES_ASSESSOR)|Primary Purpose: TREATMENT\t2018-08-20\t2019-02-2
8",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NCT059
92571\tOral Ketone Monoester Supplementation and Resting-state Brain Connectivity\t\tRECRUITING\tCerebrovascular Function|Cognition\tOTHER: Placebo|DIETARY_SUPPLEMENT: β-OHB\tMcMaster University\tAlzheimer\'s Society o
f Brant", Haldimand Norfolk," Hamilton Halton\t30.0\tOTHER\tINTERVENTIONAL\tAllocation: RANDOMIZED|Intervention Model: CROSSOVER|Masking: TRIPLE (PARTICIPANT", INVESTIGATOR," OUTCOMES_ASSESSOR)|Primary Purpose: BASIC_S
CIENCE\t2023-10-25\t2024-0
8",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NCT00
237471\tImpact of Tight Glycaemic Control in Acute Myocardial Infarction\t\tTERMINATED\tMyocardial Infarct|Hyperglycemia\tDRUG: Insulin (tight blood glucose control)\tMelbourne Health\tNational Health and Medical Resea
rch Council"," Australia|Bristol-Myers Squibb\t40.0\tOTHER\tINTERVENTIONAL\tAllocation: RANDOMIZED|Intervention Model: PARALLEL|Masking: NONE|Primary Purpose: TREATMENT\t2005-10\t2006-0
5",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NC
T03820271\tNew Prognostic Predictive Models of Mortality of Decompensated Cirrhotic Patients Waiting for Liver Transplantation\tSUPERMELD\tRECRUITING\tDecompensated Cirrhosis|Liver Transplantation\tOTHER: SuperMELD\tAs
sistance Publique - Hôpitaux de Paris\t\t500.0\tOTHER\tINTERVENTIONAL\tAllocation: NA|Intervention Model: SINGLE_GROUP|Masking: NONE|Primary Purpose: OTHER\t2020-10-01\t2023-10-0
1",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"N
CT06229171\tInTake Care: Development and Validation of an Innovative"," Personalized Digital Health Solution for Medication Adherence Support in Cardiovascular Prevention\tInTakeCare\tNOT_YET_RECRUITING\tHypertension|T
reatment Adherence and Compliance|Digital Health\tOTHER: adherence support system based on a vocal assistant\tIstituto Auxologico Italiano\tIstituti Clinici Scientifici Maugeri SpA|Politecnico di Milano\t206.0\tOTHER\t
INTERVENTIONAL\tAllocation: RANDOMIZED|Intervention Model: PARALLEL|Masking: NONE|Primary Purpose: OTHER\t2024-10-01\t2026-04-0
1"...........................................................................................................................................................................................................\r\n"NC
```

Command took 0.38 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment

Fig 2.2

DATA CLEANING AND PREPARATION

RDD



CREATE RDD FOR CLINICAL TRIALS

defines a function to read and convert the clinicaltrial data into RDD

```python
def load_clinical_trial_data(clinicaltrial_2023):
    """
    Load clinical trial data from a CSV file into an RDD.

    Parameters:
        clinicaltrial_2023 (str): The name of the CSV file to load.

    Returns:
        pyspark.rdd.RDD: An RDD containing the clinical trial data.
    """
    crdd = sc.textFile("/FileStore/tables/" + clinicaltrial_2023 + ".csv")
    return crdd
```

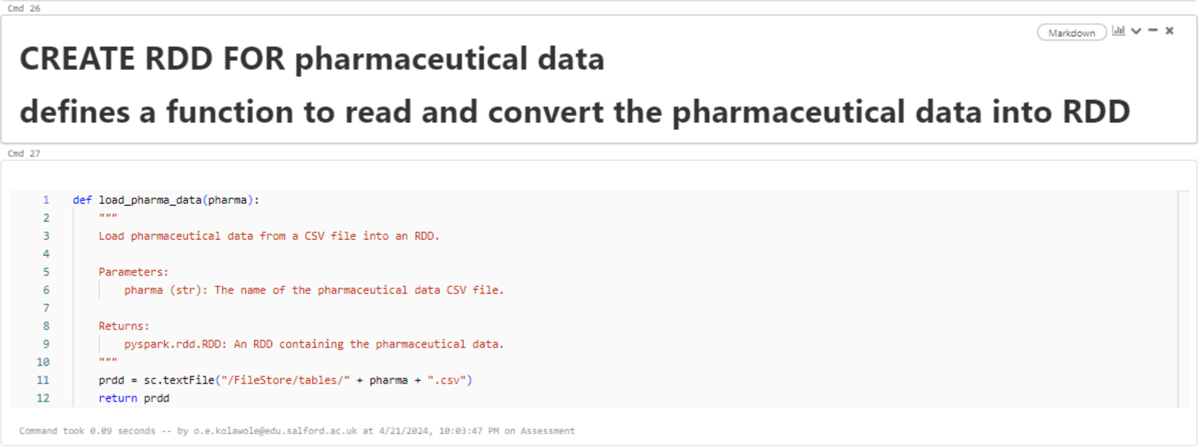
```
Command took 0.08 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 2.3 Clinical file RDD Creation

The code in the figure above creates RDD for clinical trials.

The function reads the data from the CSV file (Clinicaltrial 2023 + ".csv" + "/FileStore/tables/") into an RDD using the sc.textFile() method.

The RDD (crdd) with the clinical trial data is then returned by the function.



CREATE RDD FOR pharmaceutical data

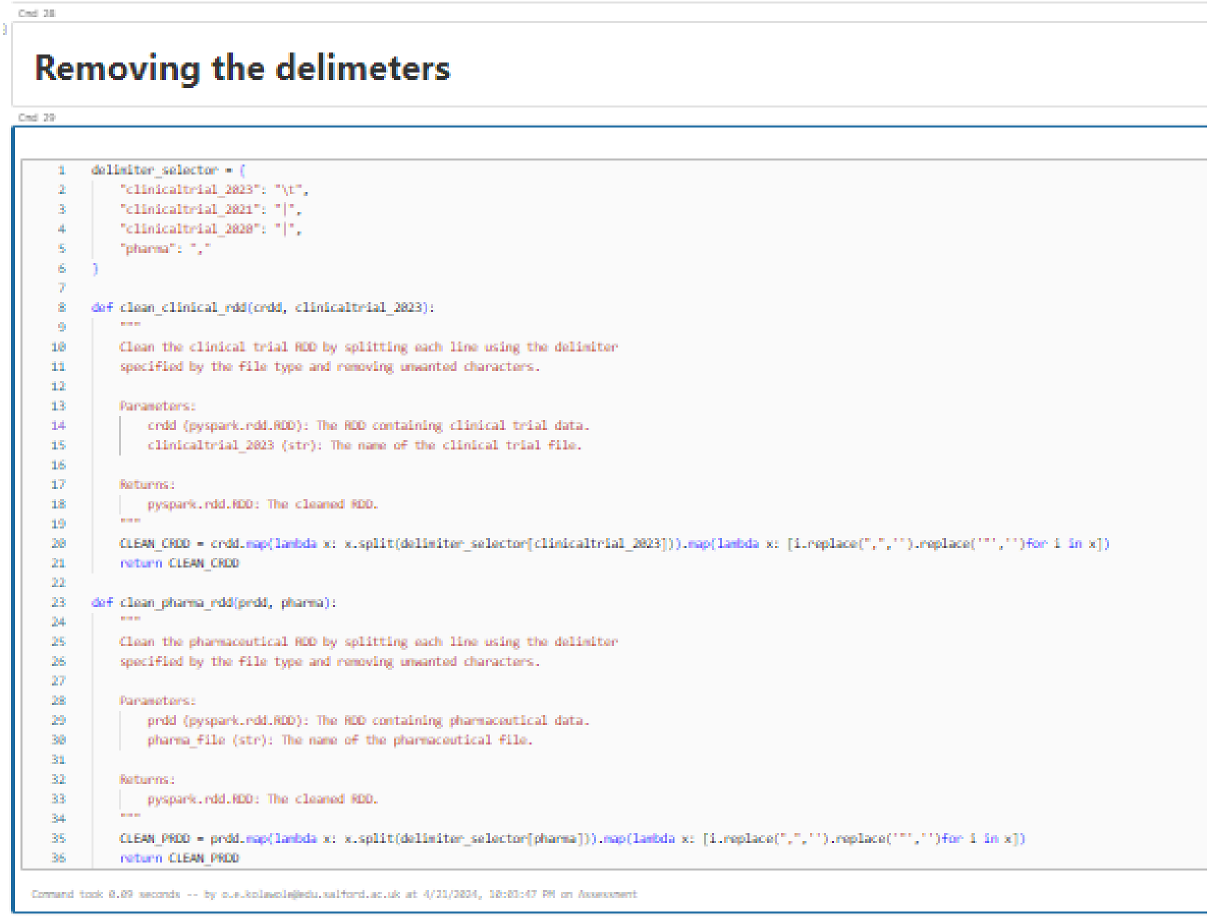
defines a function to read and convert the pharmaceutical data into RDD

```python
def load_pharma_data(pharma):
    """
    Load pharmaceutical data from a CSV file into an RDD.

    Parameters:
        pharma (str): The name of the pharmaceutical data CSV file.

    Returns:
        pyspark.rdd.RDD: An RDD containing the pharmaceutical data.
    """
    prdd = sc.textFile("/FileStore/tables/" + pharma + ".csv")
    return prdd
```

```
Command took 0.09 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 2.4 Pharma file rdd creation

The function reads the contents of the CSV file ("/FileStore/tables/" + pharma + ".csv") supplied by pharma into an RDD using the sc.textFile() method. The RDD with the pharmaceutical data is then returned by the function.



Cmd 28

Removing the delimeters

Cmd 29

```python
delimiter_selector = {
    "clinicaltrial_2023": "\t",
    "clinicaltrial_2021": "|",
    "clinicaltrial_2020": "|",
    "pharma": ","
}

def clean_clinical_rdd(crdd, clinicaltrial_2023):
    """
    Clean the clinical trial RDD by splitting each line using the delimiter
    specified by the file type and removing unwanted characters.

    Parameters:
        crdd (pyspark.rdd.RDD): The RDD containing clinical trial data.
        clinicaltrial_2023 (str): The name of the clinical trial file.

    Returns:
        pyspark.rdd.RDD: The cleaned RDD.
    """
    CLEAN_CRDD = crdd.map(lambda x: x.split(delimiter_selector[clinicaltrial_2023])).map(lambda x: [i.replace(",",'').replace('"','')for i in x])
    return CLEAN_CRDD

def clean_pharma_rdd(prdd, pharma):
    """
    Clean the pharmaceutical RDD by splitting each line using the delimiter
    specified by the file type and removing unwanted characters.

    Parameters:
        prdd (pyspark.rdd.RDD): The RDD containing pharmaceutical data.
        pharma_file (str): The name of the pharmaceutical file.

    Returns:
        pyspark.rdd.RDD: The cleaned RDD.
    """
    CLEAN_PRDD = prdd.map(lambda x: x.split(delimiter_selector[pharma])).map(lambda x: [i.replace(",",'').replace('"','')for i in x])
    return CLEAN_PRDD
```

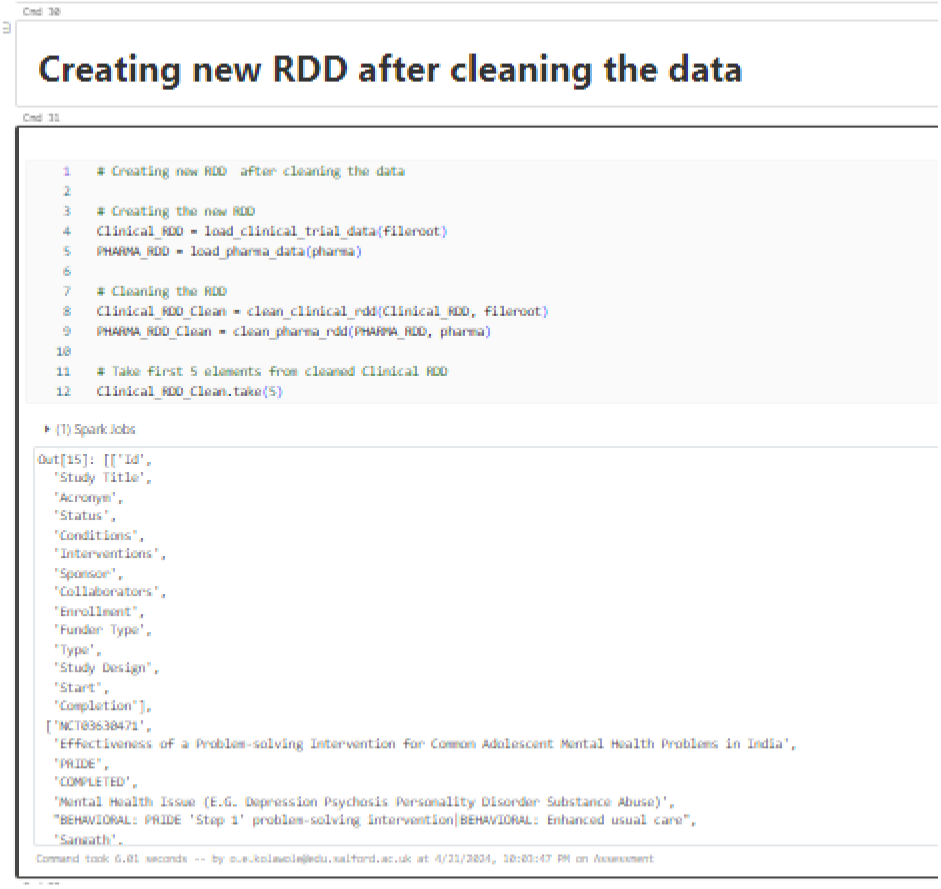
Command took 0.09 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment

Fig 2.5 Removing the delimiters

The two functions clean_pharma_rdd and clean_clinical_rdd, which are defined in this Python code, clean the data in RDDs that contain pharmaceutical and clinical trial data, respectively.

crdd.map(lambda x: x.split(delimiter_selector[clinicaltrial_2023])) splits each line of the RDD (crdd) into a list of elements using the delimiter specified by delimiter. The delimiter is chosen based on the clinical trial year.

.map(lambda x: [i.replace(",",'').replace('"','') for i in x]) removes unwanted characters (commas and double quotes) from each element. This also done for the pharma rdd.



Creating new RDD after cleaning the data

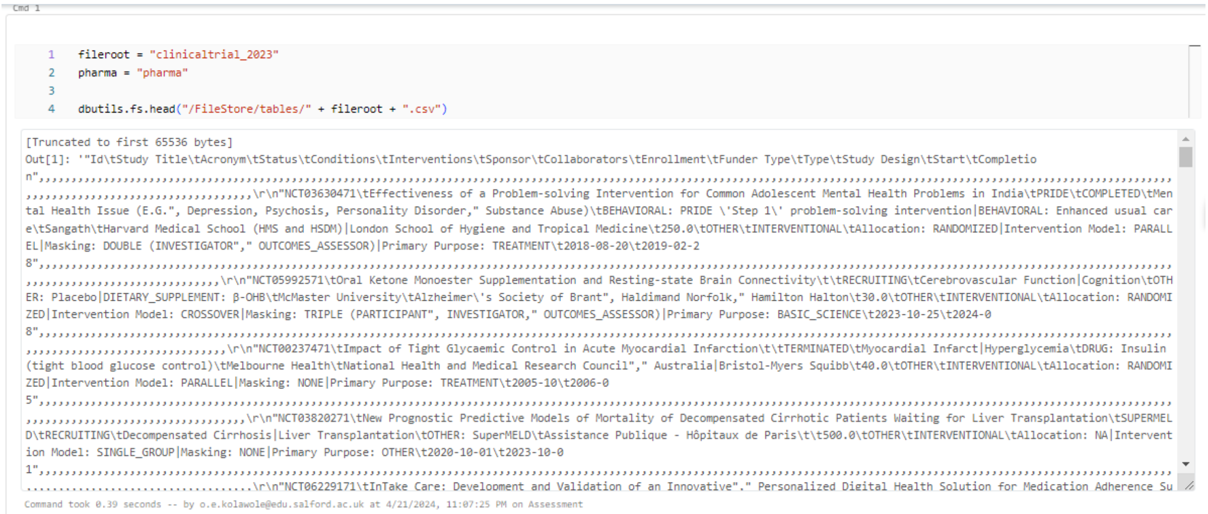
```
# Creating new RDD  after cleaning the data

# Creating the new RDD
Clinical_RDD = load_clinical_trial_data(fileroot)
PHARMA_RDD = load_pharma_data(pharma)

# Cleaning the RDD
Clinical_RDD_Clean = clean_clinical_rdd(Clinical_RDD, fileroot)
PHARMA_RDD_Clean = clean_pharma_rdd(PHARMA_RDD, pharma)

# Take first 5 elements from cleaned Clinical RDD
Clinical_RDD_Clean.take(5)
```

```
Out[15]: [['Id',
  'Study Title',
  'Acronym',
  'Status',
  'Conditions',
  'Interventions',
  'Sponsor',
  'Collaborators',
  'Enrollment',
  'Funder Type',
  'Type',
  'Study Design',
  'Start',
  'Completion'],
 ['NCT03630471',
  'Effectiveness of a Problem-solving Intervention for Common Adolescent Mental Health Problems in India',
  'PRIDE',
  'COMPLETED',
  'Mental Health Issue (E.G. Depression Psychosis Personality Disorder Substance Abuse)',
  "BEHAVIORAL: PRIDE 'Step 1' problem-solving intervention|BEHAVIORAL: Enhanced usual care",
  'Sangath',
Command took 6.01 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

Fig 2.6 Creating new rdd

The load_clinical_trial_data and load_pharma_data is use to load the previously cleaned and then renamed. Thereafter, the cleaned clinical trial rdd is then display to show the first five rows.

DATAFRAME

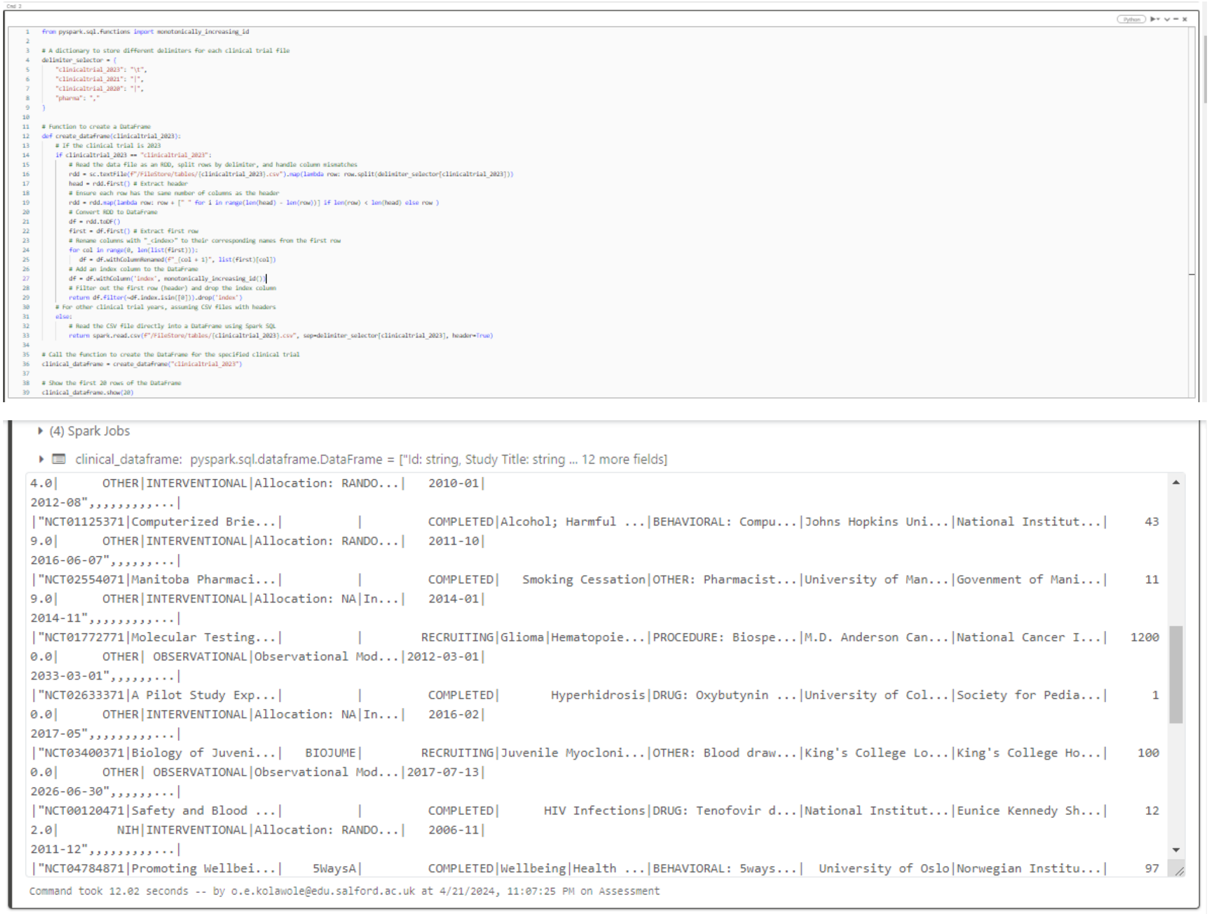


```
fileroot = "clinicaltrial_2023"
pharma = "pharma"

dbutils.fs.head("/FileStore/tables/" + fileroot + ".csv")
```

```
[Truncated to first 65536 bytes]
Out[1]: '"Id\tStudy Title\tAcronym\tStatus\tConditions\tInterventions\tSponsor\tCollaborators\tEnrollment\tFunder Type\tType\tStudy Design\tStart\tCompletio
n",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NCT03630471\tEffectiveness of a Problem-solving Intervention for Common Adolescent Mental Health Problems in India\tPRIDE\tCOMPLETED\tMen
tal Health Issue (E.G.", Depression, Psychosis, Personality Disorder," Substance Abuse)\tBEHAVIORAL: PRIDE \'Step 1\' problem-solving intervention|BEHAVIORAL: Enhanced usual car
e\tSangath\tHarvard Medical School (HMS and HSDM)|London School of Hygiene and Tropical Medicine\t250.0\tOTHER\tINTERVENTIONAL\tAllocation: RANDOMIZED|Intervention Model: PARALL
EL|Masking: DOUBLE (INVESTIGATOR"," OUTCOMES_ASSESSOR)|Primary Purpose: TREATMENT\t2018-08-20\t2019-02-2
8",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NCT05992571\tOral Ketone Monoester Supplementation and Resting-state Brain Connectivity\t\tRECRUITING\tCerebrovascular Function|Cognition\tOTH
ER: Placebo|DIETARY_SUPPLEMENT: β-OHB\tMcMaster University\tAlzheimer\'s Society of Brant", Haldimand Norfolk," Hamilton Halton\t30.0\tOTHER\tINTERVENTIONAL\tAllocation: RANDOMI
ZED|Intervention Model: CROSSOVER|Masking: TRIPLE (PARTICIPANT", INVESTIGATOR," OUTCOMES_ASSESSOR)|Primary Purpose: BASIC_SCIENCE\t2023-10-25\t2024-0
8",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NCT00237471\tImpact of Tight Glycaemic Control in Acute Myocardial Infarction\t\tTERMINATED\tMyocardial Infarct|Hyperglycemia\tDRUG: Insulin
(tight blood glucose control)\tMelbourne Health\tNational Health and Medical Research Council"," Australia|Bristol-Myers Squibb\t40.0\tOTHER\tINTERVENTIONAL\tAllocation: RANDOMI
ZED|Intervention Model: PARALLEL|Masking: NONE|Primary Purpose: TREATMENT\t2005-10\t2006-0
5",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NCT03820271\tNew Prognostic Predictive Models of Mortality of Decompensated Cirrhotic Patients Waiting for Liver Transplantation\tSUPERMEL
D\tRECRUITING\tDecompensated Cirrhosis|Liver Transplantation\tOTHER: SuperMELD\tAssistance Publique - Hôpitaux de Paris\t\t500.0\tOTHER\tINTERVENTIONAL\tAllocation: NA|Intervent
ion Model: SINGLE_GROUP|Masking: NONE|Primary Purpose: OTHER\t2020-10-01\t2023-10-0
1",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NCT06229171\tInTake Care: Development and Validation of an Innovative"." Personalized Digital Health Solution for Medication Adherence Su
Command took 0.39 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:07:25 PM on Assessment
```

Fig 2.7

The first few rows of the clinical trial file or file root was checked.



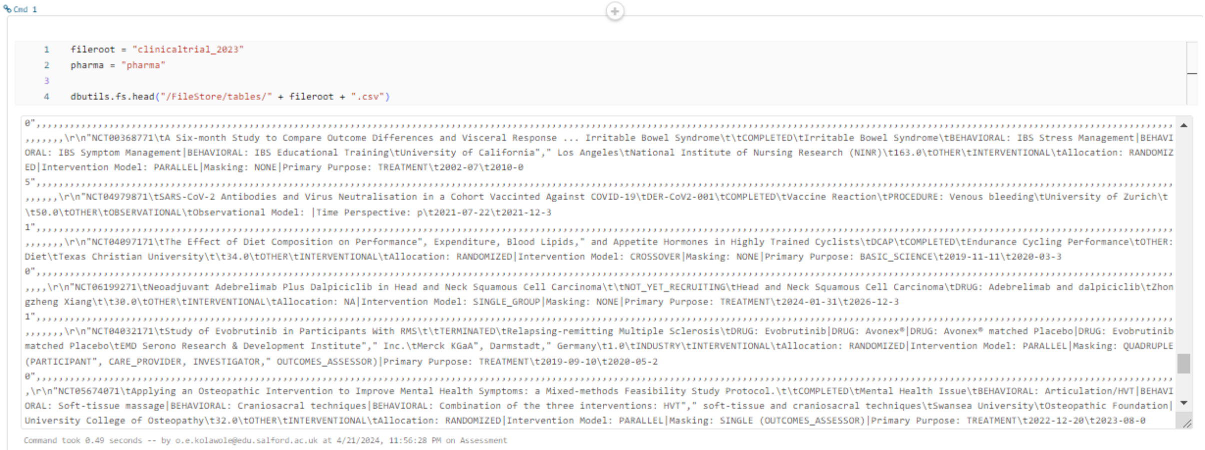
```python
from pyspark.sql.functions import monotonically_increasing_id

# A dictionary to store different delimiters for each clinical trial file
delimiter_selector = {
    "clinicaltrial_2023": "\t",
    "clinicaltrial_2021": "|",
    "clinicaltrial_2020": "|",
    "pharma": ","
}

# Function to create a DataFrame
def create_dataframe(clinicaltrial_2023):
    # If the clinical trial is 2023
    if clinicaltrial_2023 == "clinicaltrial_2023":
        # Read the data file as an RDD, split rows by delimiter, and handle column mismatches
        rdd = sc.textFile(f"/FileStore/tables/{clinicaltrial_2023}.csv").map(lambda row: row.split(delimiter_selector[clinicaltrial_2023]))
        head = rdd.first() # Extract header
        # Ensure each row has the same number of columns as the header
        rdd = rdd.map(lambda row: row + [" " for i in range(len(head) - len(row))] if len(row) < len(head) else row )
        # Convert RDD to DataFrame
        df = rdd.toDF()
        first = df.first() # Extract first row
        # Rename columns with "_<index>" to their corresponding names from the first row
        for col in range(0, len(list(first))):
            df = df.withColumnRenamed(f"_{col + 1}", list(first)[col])
        # Add an index column to the DataFrame
        df = df.withColumn('index', monotonically_increasing_id())
        # Filter out the first row (header) and drop the index column
        return df.filter(~df.index.isin([0])).drop('index')
    # For other clinical trial years, assuming CSV files with headers
    else:
        # Read the CSV file directly into a DataFrame using Spark SQL
        return spark.read.csv(f"/FileStore/tables/{clinicaltrial_2023}.csv", sep=delimiter_selector[clinicaltrial_2023], header=True)

# Call the function to create the DataFrame for the specified clinical trial
clinical_dataframe = create_dataframe("clinicaltrial_2023")

# Show the first 20 rows of the DataFrame
clinical_dataframe.show(20)
```

```
▸ (4) Spark Jobs
▸ clinical_dataframe:  pyspark.sql.dataframe.DataFrame = ["Id: string, Study Title: string ... 12 more fields]
4.0|       OTHER|INTERVENTIONAL|Allocation: RANDO...|   2010-01|
2012-08",,,,,,,,,,,...|
|"NCT01125371|Computerized Brie...|          |         COMPLETED|Alcohol; Harmful ...|BEHAVIORAL: Compu...|Johns Hopkins Uni...|National Institut...|     43
9.0|       OTHER|INTERVENTIONAL|Allocation: RANDO...|   2011-10|
2016-06-07",,,,,,,,,...|
|"NCT02554071|Manitoba Pharmaci...|          |         COMPLETED|   Smoking Cessation|OTHER: Pharmacist...|University of Man...|Government of Mani...|     11
9.0|       OTHER|INTERVENTIONAL|Allocation: NA|In...|   2014-01|
2014-11",,,,,,,,,,,...|
|"NCT01772771|Molecular Testing...|          |        RECRUITING|Glioma|Hematopoie...|PROCEDURE: Biospe...|M.D. Anderson Can...|National Cancer I...|   1200
0.0|       OTHER| OBSERVATIONAL|Observational Mod...|2012-03-01|
2033-03-01",,,,,,,,,...|
|"NCT02633371|A Pilot Study Exp...|          |         COMPLETED|       Hyperhidrosis|DRUG: Oxybutynin ...|University of Col...|Society for Pedia...|      1
0.0|       OTHER|INTERVENTIONAL|Allocation: NA|In...|   2016-02|
2017-05",,,,,,,,,,,...|
|"NCT03400371|Biology of Juveni...|   BIOJUME|        RECRUITING|Juvenile Myocloni...|OTHER: Blood draw...|King's College Lo...|King's College Ho...|    100
0.0|       OTHER| OBSERVATIONAL|Observational Mod...|2017-07-13|
2026-06-30",,,,,,,,,...|
|"NCT00120471|Safety and Blood ...|          |         COMPLETED|      HIV Infections|DRUG: Tenofovir d...|National Institut...|Eunice Kennedy Sh...|     12
2.0|         NIH|INTERVENTIONAL|Allocation: RANDO...|   2006-11|
2011-12",,,,,,,,,,,...|
|"NCT04784871|Promoting Wellbei...|    5WaysA|         COMPLETED|Wellbeing|Health ...|BEHAVIORAL: 5ways...|  University of Oslo|Norwegian Institu...|     97
Command took 12.02 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:07:25 PM on Assessment
```

Fig 2.8

This code defines the create_dataframe function, which takes a CSV file containing clinical trial data and uses it to create a DataFrame. It Import ‘monotonically_increasing_id’ from pyspark.sql.functions which makes every row in a DataFrame has a unique ID. The delimiter selector was then defined which act as a key for each clinical trial file. The dataframe is created in response to the clinicaltrial_2023 by dividing the rows after reading the data as an RDD and ensuring every row has the same number of columns as the header even after extracting the header. Then the first 20 rows of the created dataframe is viewed.

SQL



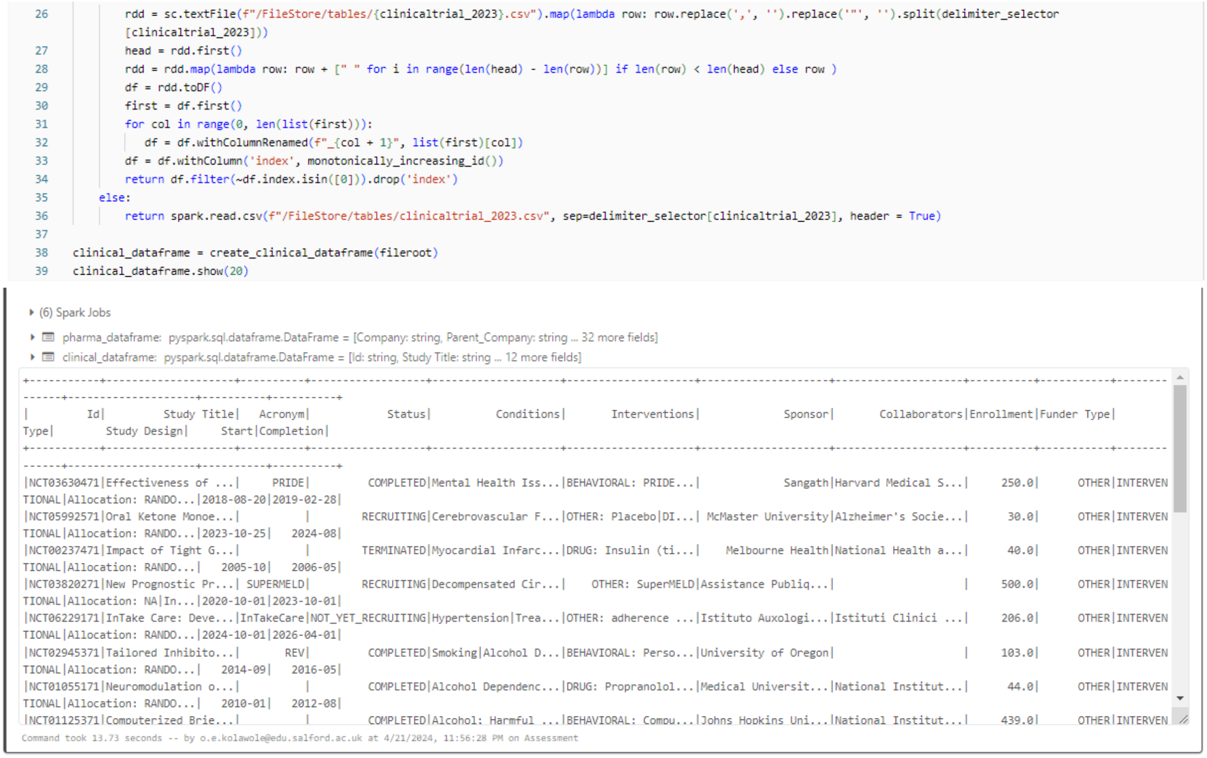
```
fileroot = "clinicaltrial_2023"
pharma = "pharma"

dbutils.fs.head("/FileStore/tables/" + fileroot + ".csv")
```

```
0",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,\r\n"NCT00368771\tA Six-month Study to Compare Outcome Differences and Visceral Response ... Irritable Bowel Syndrome\t\tCOMPLETED\tIrritable Bowel Syndrome\tBEHAVIORAL: IBS Stress Management|BEHAVI
ORAL: IBS Symptom Management|BEHAVIORAL: IBS Educational Training\tUniversity of California"," Los Angeles\tNational Institute of Nursing Research (NINR)\t163.0\tOTHER\tINTERVENTIONAL\tAllocation: RANDOMIZ
ED|Intervention Model: PARALLEL|Masking: NONE|Primary Purpose: TREATMENT\t2002-07\t2010-0
5",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,\r\n"NCT04979871\tSARS-CoV-2 Antibodies and Virus Neutralisation in a Cohort Vaccinted Against COVID-19\tDER-CoV2-001\tCOMPLETED\tVaccine Reaction\tPROCEDURE: Venous bleeding\tUniversity of Zurich\t
\t50.0\tOTHER\tOBSERVATIONAL\tObservational Model: |Time Perspective: p\t2021-07-22\t2021-12-3
1",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,\r\n"NCT04097171\tThe Effect of Diet Composition on Performance", Expenditure, Blood Lipids," and Appetite Hormones in Highly Trained Cyclists\tDCAP\tCOMPLETED\tEndurance Cycling Performance\tOTHER:
Diet\tTexas Christian University\t\t34.0\tOTHER\tINTERVENTIONAL\tAllocation: RANDOMIZED|Intervention Model: CROSSOVER|Masking: NONE|Primary Purpose: BASIC_SCIENCE\t2019-11-11\t2020-03-3
0",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,\r\n"NCT06199271\tNeoadjuvant Adebrelimab Plus Dalpiciclib in Head and Neck Squamous Cell Carcinoma\t\tNOT_YET_RECRUITING\tHead and Neck Squamous Cell Carcinoma\tDRUG: Adebrelimab and dalpiciclib\tZhon
gzheng Xiang\t\t30.0\tOTHER\tINTERVENTIONAL\tAllocation: NA|Intervention Model: SINGLE_GROUP|Masking: NONE|Primary Purpose: TREATMENT\t2024-01-31\t2026-12-3
1",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,\r\n"NCT04032171\tStudy of Evobrutinib in Participants With RMS\t\tTERMINATED\tRelapsing-remitting Multiple Sclerosis\tDRUG: Evobrutinib|DRUG: Avonex®|DRUG: Avonex® matched Placebo|DRUG: Evobrutinib
matched Placebo\tEMD Serono Research & Development Institute"," Inc.\tMerck KGaA", Darmstadt," Germany\t1.0\tINDUSTRY\tINTERVENTIONAL\tAllocation: RANDOMIZED|Intervention Model: PARALLEL|Masking: QUADRUPLE
(PARTICIPANT", CARE_PROVIDER, INVESTIGATOR," OUTCOMES_ASSESSOR)|Primary Purpose: TREATMENT\t2019-09-10\t2020-05-2
0",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,\r\n"NCT05674071\tApplying an Osteopathic Intervention to Improve Mental Health Symptoms: a Mixed-methods Feasibility Study Protocol.\t\tCOMPLETED\tMental Health Issue\tBEHAVIORAL: Articulation/HVT|BEHAVI
ORAL: Soft-tissue massage|BEHAVIORAL: Craniosacral techniques|BEHAVIORAL: Combination of the three interventions: HVT"," soft-tissue and craniosacral techniques\tSwansea University\tOsteopathic Foundation|
University College of Osteopathy\t32.0\tOTHER\tINTERVENTIONAL\tAllocation: RANDOMIZED|Intervention Model: PARALLEL|Masking: SINGLE (OUTCOMES_ASSESSOR)|Primary Purpose: TREATMENT\t2022-12-20\t2023-08-0
```

Command took 0.49 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:56:28 PM on Assessment

Fig 2.9

The first few rows of the clinical trial file or file root were checked.



```
        rdd = sc.textFile(f"/FileStore/tables/{clinicaltrial_2023}.csv").map(lambda row: row.replace(',', '').replace('"', '').split(delimiter_selector
        [clinicaltrial_2023]))
        head = rdd.first()
        rdd = rdd.map(lambda row: row + [" " for i in range(len(head) - len(row))] if len(row) < len(head) else row )
        df = rdd.toDF()
        first = df.first()
        for col in range(0, len(list(first))):
            df = df.withColumnRenamed(f"_{col + 1}", list(first)[col])
        df = df.withColumn('index', monotonically_increasing_id())
        return df.filter(~df.index.isin([0])).drop('index')
    else:
        return spark.read.csv(f"/FileStore/tables/clinicaltrial_2023.csv", sep=delimiter_selector[clinicaltrial_2023], header = True)

clinical_dataframe = create_clinical_dataframe(fileroot)
clinical_dataframe.show(20)
```

▸ (6) Spark Jobs

▸ pharma_dataframe: pyspark.sql.dataframe.DataFrame = [Company: string, Parent_Company: string ... 32 more fields]

▸ clinical_dataframe: pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 12 more fields]

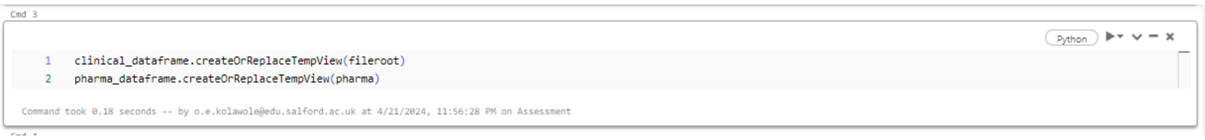
```
+-----------+--------------------+----------+------------------+--------------------+--------------------+--------------------+--------------------+----------+-----------+--------
------+--------------------+----------+----------+
|         Id|         Study Title|   Acronym|            Status|          Conditions|       Interventions|             Sponsor|       Collaborators|Enrollment|Funder Type|
Type|        Study Design|     Start|Completion|
+-----------+--------------------+----------+------------------+--------------------+--------------------+--------------------+--------------------+----------+-----------+--------
------+--------------------+----------+----------+
|NCT03630471|Effectiveness of ...|     PRIDE|         COMPLETED|Mental Health Iss...|BEHAVIORAL: PRIDE...|             Sangath|Harvard Medical S...|     250.0|      OTHER|INTERVEN
TIONAL|Allocation: RANDO...|2018-08-20|2019-02-28|
|NCT05992571|Oral Ketone Monoe...|          |        RECRUITING|Cerebrovascular F...|OTHER: Placebo|DI...| McMaster University|Alzheimer's Socie...|      30.0|      OTHER|INTERVEN
TIONAL|Allocation: RANDO...|2023-10-25|   2024-08|
|NCT00237471|Impact of Tight G...|          |        TERMINATED|Myocardial Infarc...|DRUG: Insulin (ti...|    Melbourne Health|National Health a...|      40.0|      OTHER|INTERVEN
TIONAL|Allocation: RANDO...|   2005-10|   2006-05|
|NCT03820271|New Prognostic Pr...| SUPERMELD|        RECRUITING|Decompensated Cir...|     OTHER: SuperMELD|Assistance Publiq...|                    |     500.0|      OTHER|INTERVEN
TIONAL|Allocation: NA|In...|2020-10-01|2023-10-01|
|NCT06229171|InTake Care: Deve...|InTakeCare|NOT_YET_RECRUITING|Hypertension|Trea...|OTHER: adherence ...|Istituto Auxologi...|Istituti Clinici ...|     206.0|      OTHER|INTERVEN
TIONAL|Allocation: RANDO...|2024-10-01|2026-04-01|
|NCT02945371|Tailored Inhibito...|       REV|         COMPLETED|Smoking|Alcohol D...|BEHAVIORAL: Perso...|University of Oregon|                    |     103.0|      OTHER|INTERVEN
TIONAL|Allocation: RANDO...|   2014-09|   2016-05|
|NCT01055171|Neuromodulation o...|          |         COMPLETED|Alcohol Dependenc...|DRUG: Propranolol...|Medical Universit...|National Institut...|      44.0|      OTHER|INTERVEN
TIONAL|Allocation: RANDO...|   2010-01|   2012-08|
|NCT01125371|Computerized Brie...|          |         COMPLETED|Alcohol: Harmful ...|BEHAVIORAL: Compu...|Johns Hopkins Uni...|National Institut...|     439.0|      OTHER|INTERVEN
```

Command took 13.73 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:56:28 PM on Assessment

Fig 3.0

The “import * from pyspark.sql.types and import * from pyspark.sql.functions” imports the libraries needed to transform and manipulate the data

The Purpose of the Pharmaceutical DataFrame: create_pharma_dataframe and create_clinical_dataframe returns the dataframe after receiving the pharma and clinicaltrial_2023 dataset as input.
Then views are created for each of the dataframe.



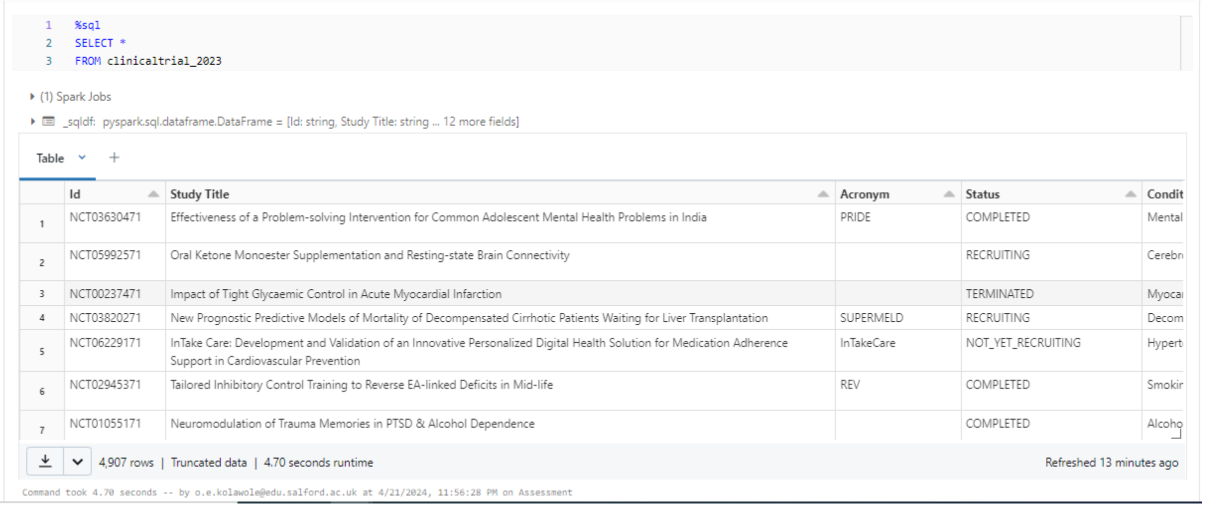
Cmd 3

```
clinical_dataframe.createOrReplaceTempView(fileroot)
pharma_dataframe.createOrReplaceTempView(pharma)
```

```
Command took 0.18 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:56:28 PM on Assessment
```

Fig 3.1

SQL queries can be run against the clinical trial and pharmaceutical datasets now that temporary views have been created for them. It will be simpler to carry out different analyses and respond to the questions as a result.



```
%sql
SELECT *
FROM clinicaltrial_2023
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 12 more fields]

	Id	Study Title	Acronym	Status	Condit
1	NCT03630471	Effectiveness of a Problem-solving Intervention for Common Adolescent Mental Health Problems in India	PRIDE	COMPLETED	Mental
2	NCT05992571	Oral Ketone Monoester Supplementation and Resting-state Brain Connectivity		RECRUITING	Cerebr
3	NCT00237471	Impact of Tight Glycaemic Control in Acute Myocardial Infarction		TERMINATED	Myocai
4	NCT03820271	New Prognostic Predictive Models of Mortality of Decompensated Cirrhotic Patients Waiting for Liver Transplantation	SUPERMELD	RECRUITING	Decom
5	NCT06229171	InTake Care: Development and Validation of an Innovative Personalized Digital Health Solution for Medication Adherence Support in Cardiovascular Prevention	InTakeCare	NOT_YET_RECRUITING	Hypert
6	NCT02945371	Tailored Inhibitory Control Training to Reverse EA-linked Deficits in Mid-life	REV	COMPLETED	Smokir
7	NCT01055171	Neuromodulation of Trauma Memories in PTSD & Alcohol Dependence		COMPLETED	Alcoho

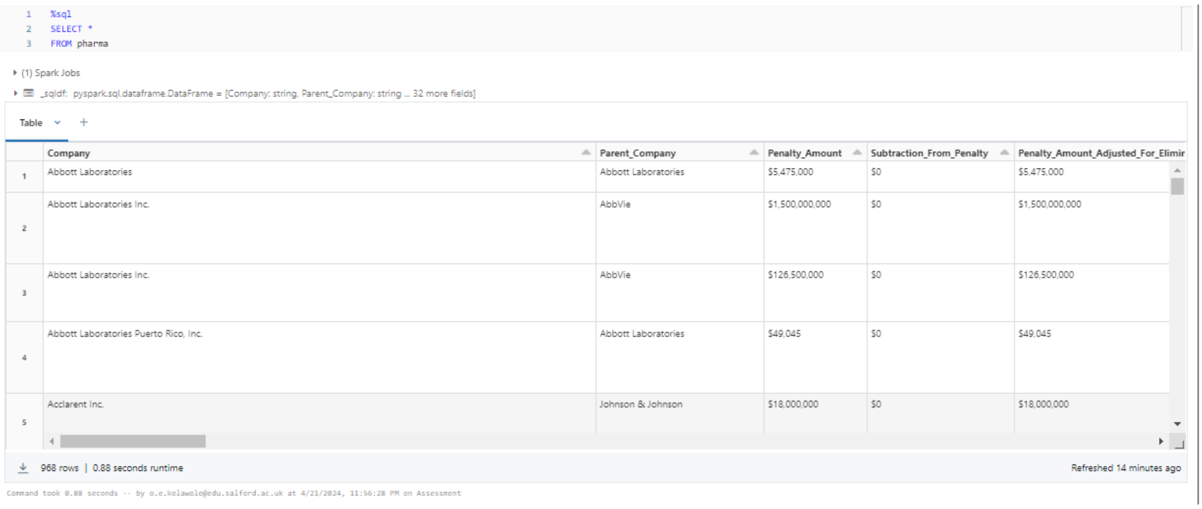
4,907 rows | Truncated data | 4.70 seconds runtime

Refreshed 13 minutes ago

```
Command took 4.70 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:56:28 PM on Assessment
```

Fig 3.2

A SQL query to extract every column from the 2023 clinical trial dataset. This will show every row and every column in the dataset.



```
%sql
SELECT *
FROM pharma
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Company: string, Parent_Company: string ... 32 more fields]

	Company	Parent_Company	Penalty_Amount	Subtraction_From_Penalty	Penalty_Amount_Adjusted_For_Elimir
1	Abbott Laboratories	Abbott Laboratories	$5,475,000	$0	$5,475,000
2	Abbott Laboratories Inc.	AbbVie	$1,500,000,000	$0	$1,500,000,000
3	Abbott Laboratories Inc.	AbbVie	$126,500,000	$0	$126,500,000
4	Abbott Laboratories Puerto Rico, Inc.	Abbott Laboratories	$49,045	$0	$49,045
5	Acclarent Inc.	Johnson & Johnson	$18,000,000	$0	$18,000,000

968 rows | 0.88 seconds runtime

Refreshed 14 minutes ago

```
Command took 0.88 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:56:28 PM on Assessment
```

Fig 3.3

A SQL query to select all columns from the pharmaceutical dataset. This will display all the rows and columns of the dataset.

PROBLEM ANSWERS

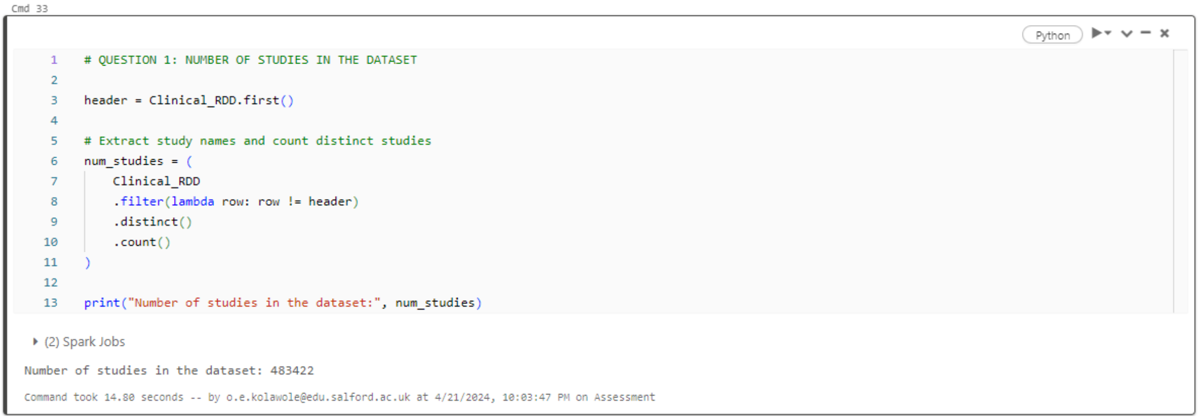
QUESTION 1

Assumptions

- Each row in the study represents a distinct study.
- In the clinical_data, there are no duplicate ID rows.

RDD

The code uses the ".first()" to extract the header from the Clinical_RDD. The header row is then filtered out, duplicate rows are removed using the distinct() transformation, and the number of remaining rows is finally counted to determine the number of distinct studies in the dataset.



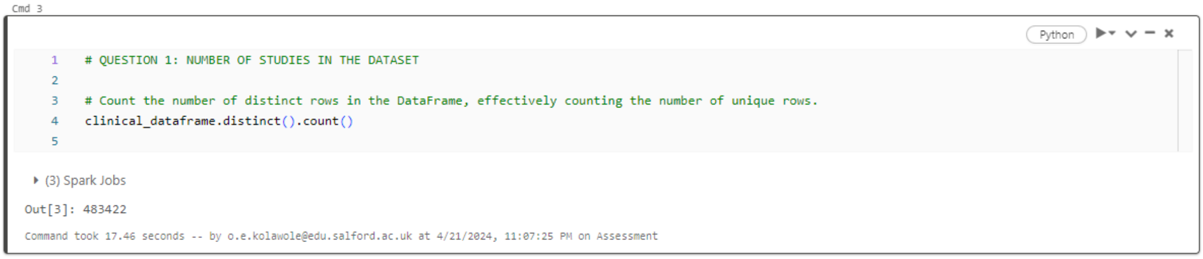
```python
# QUESTION 1: NUMBER OF STUDIES IN THE DATASET

header = Clinical_RDD.first()

# Extract study names and count distinct studies
num_studies = (
    Clinical_RDD
    .filter(lambda row: row != header)
    .distinct()
    .count()
)

print("Number of studies in the dataset:", num_studies)
```

(2) Spark Jobs

```
Number of studies in the dataset: 483422
Command took 14.80 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment
```

DATA FRAME

Duplicate rows are eliminated by using the DataFrame's distinct() function, and the number of unique rows that remain is then counted using the count() function.

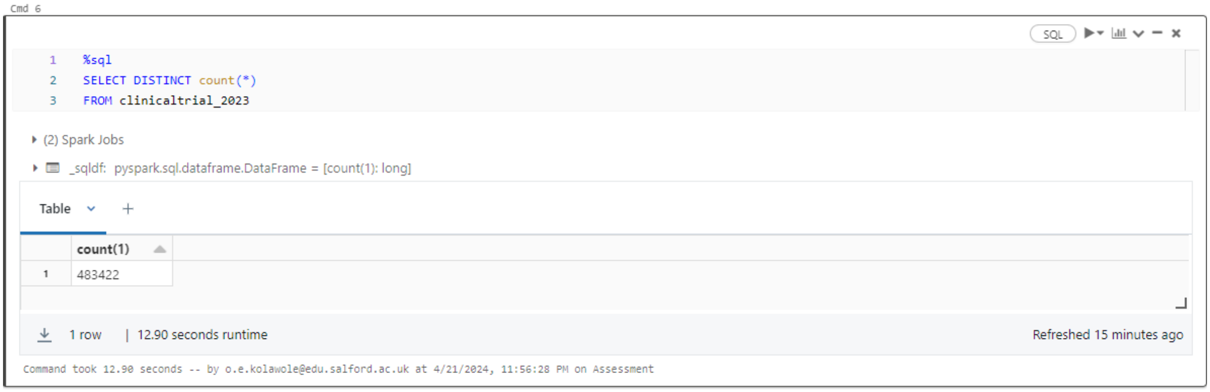


```
# QUESTION 1: NUMBER OF STUDIES IN THE DATASET

# Count the number of distinct rows in the DataFrame, effectively counting the number of unique rows.
clinical_dataframe.distinct().count()

```

```
(3) Spark Jobs
Out[3]: 483422
Command took 17.46 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:07:25 PM on Assessment
```

SQL

The DISTINCT count in the SQL will count the number of unique rows in the clinicaltrial_2023 table using the 'SELECT' and 'FROM' statements.



```
%sql
SELECT DISTINCT count(*)
FROM clinicaltrial_2023
```

```
(2) Spark Jobs
_sqldf: pyspark.sql.dataframe.DataFrame = [count(1): long]
```

	count(1)
1	483422

```
1 row  | 12.90 seconds runtime                                  Refreshed 15 minutes ago
Command took 12.90 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:56:28 PM on Assessment
```

Discussion of Result

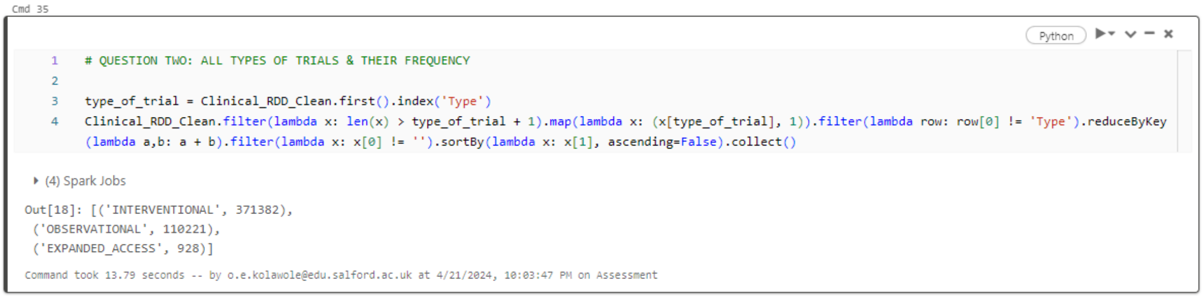
The total number of distinct studies on the 2023 clinical trial is represented by 483,422 rows in the data. This also sheds light on the quantity and range of the clinical trial that were carried out.

QUESTION 2

Assumptions

The "Type" column in the "clinicaltrial_2023" contains the different types of clinical trials performed and are likely to be replicated.

RDD



Cmd 35

```
# QUESTION TWO: ALL TYPES OF TRIALS & THEIR FREQUENCY

type_of_trial = Clinical_RDD_Clean.first().index('Type')
Clinical_RDD_Clean.filter(lambda x: len(x) > type_of_trial + 1).map(lambda x: (x[type_of_trial], 1)).filter(lambda row: row[0] != 'Type').reduceByKey
(lambda a,b: a + b).filter(lambda x: x[0] != '').sortBy(lambda x: x[1], ascending=False).collect()
```

▸ (4) Spark Jobs

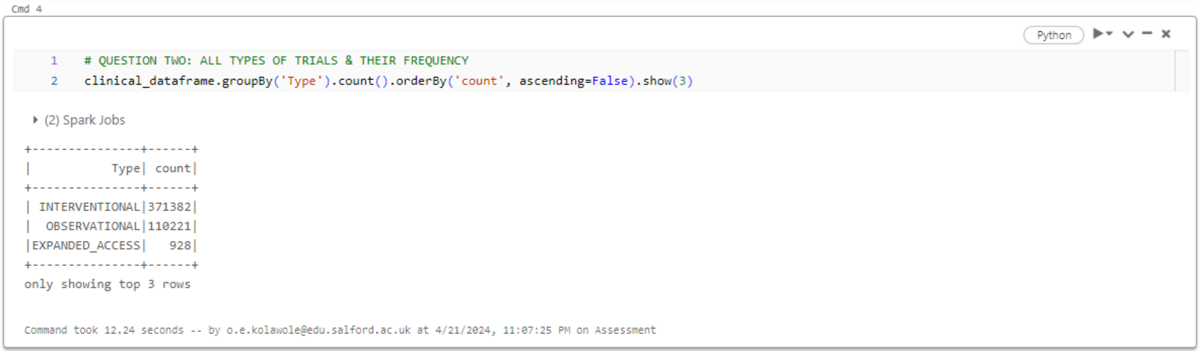
```
Out[18]: [('INTERVENTIONAL', 371382),
 ('OBSERVATIONAL', 110221),
 ('EXPANDED_ACCESS', 928)]
```

Command took 13.79 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment

The code first finds the index of the 'Type' column in your RDD then removes any rows that don't have the 'Type' column then maps each row to a tuple where the first element is the value in the 'Type' column and the second element is 1, indicating a count of one. It then filters out any tuples where the value in the 'Type' column is 'Type'. There sums up the counts for each type and filters out any empty values in the 'Type' column and sorts the results by the count of each type in descending order.

The result is then displayed.

DATA FRAME



Cmd 4

```
# QUESTION TWO: ALL TYPES OF TRIALS & THEIR FREQUENCY
clinical_dataframe.groupBy('Type').count().orderBy('count', ascending=False).show(3)
```

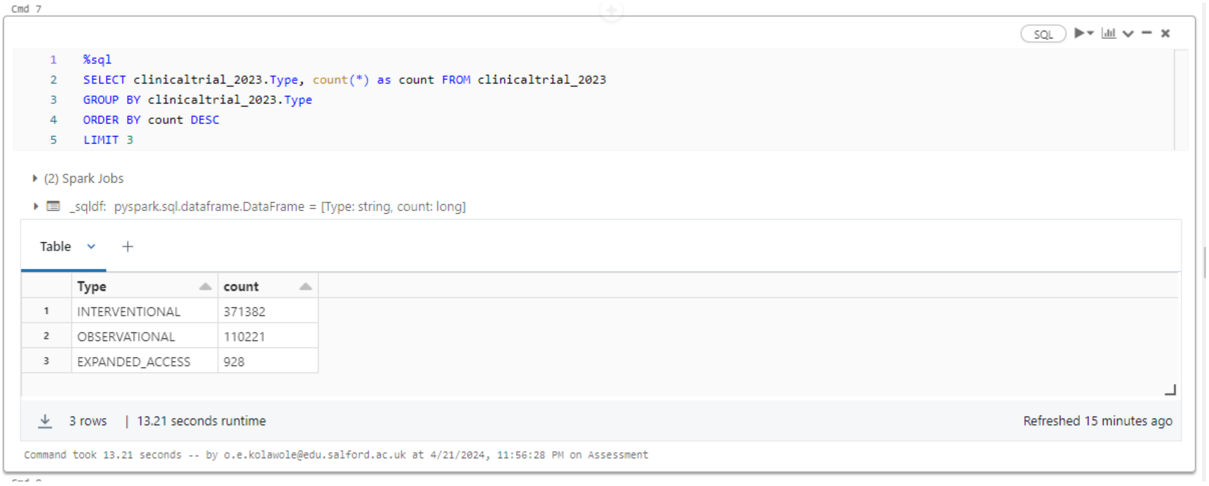
▸ (2) Spark Jobs

```
+---------------+------+
|           Type| count|
+---------------+------+
| INTERVENTIONAL|371382|
|  OBSERVATIONAL|110221|
|EXPANDED_ACCESS|   928|
+---------------+------+
only showing top 3 rows
```

Command took 12.24 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:07:25 PM on Assessment

This query groups the dataframe by the ‘Type’ column , counts the number of occurrences of each distinct value in the column within each group, orders it in descending order then shows the top three rows.

SQL



Cmd 7

```
%sql
SELECT clinicaltrial_2023.Type, count(*) as count FROM clinicaltrial_2023
GROUP BY clinicaltrial_2023.Type
ORDER BY count DESC
LIMIT 3
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Type: string, count: long]

	Type	count
1	INTERVENTIONAL	371382
2	OBSERVATIONAL	110221
3	EXPANDED_ACCESS	928

3 rows | 13.21 seconds runtime

Refreshed 15 minutes ago

```
Command took 13.21 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:56:28 PM on Assessment
```

The query will group the clinicaltrial_2023 file by their type, count the number of trials for each type, order them by count in descending order, and then limit the result to the top 3 types from the select statements.

Discussion of Result

It appears that the most common type of clinical trial is "INTERVENTIONAL," with a frequency of 371,382 trials. This suggests that a significant portion of the dataset consists of trials involving interventions or treatments, where researchers actively intervene to study their effects on participants.

The second most common type is "OBSERVATIONAL," with a frequency of 110,221 trials. Observational trials involve observing participants in their natural environment without any intervention.

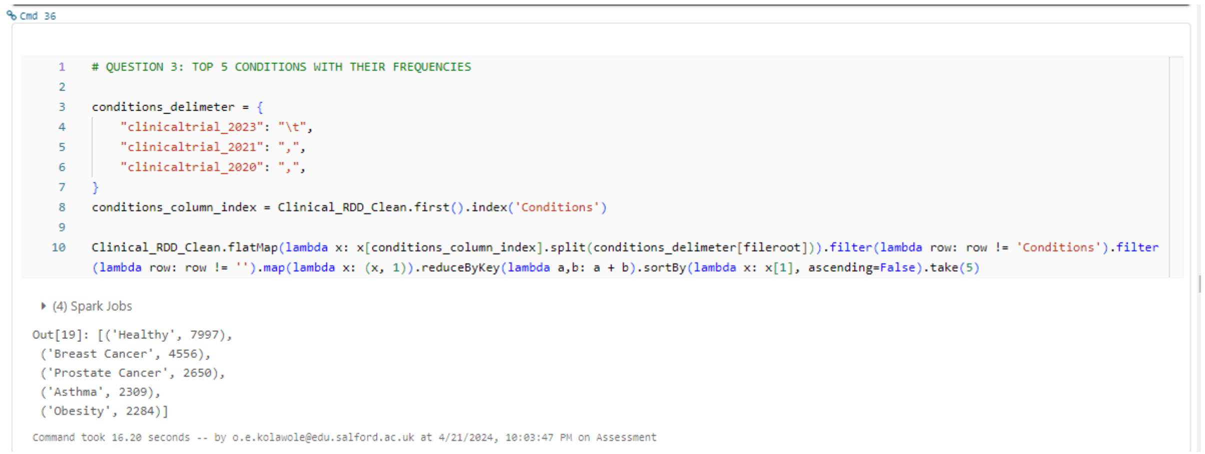
"EXPANDED_ACCESS" trials are the least common, with a frequency of 928 trials. Expanded access trials, also known as compassionate use or early access programs, provide investigational treatments to patients with serious diseases or conditions outside of clinical trials when no satisfactory alternative treatments are available.

QUESTION 3

Assumptions

The “Conditions” column in the “clinicaltrial_2023” contains the different conditions or diseases being studied in each clinical trial performed and are likely to be reoccurring.

RDD



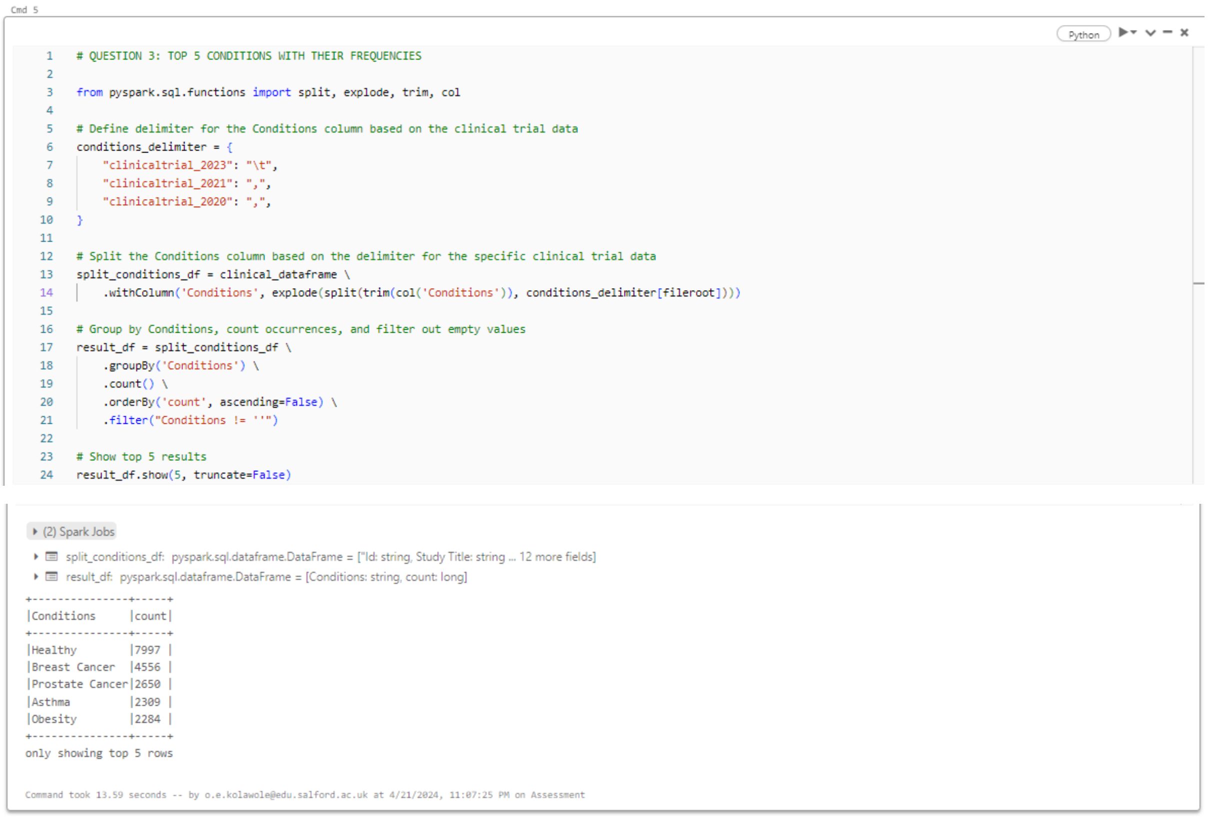
```
# QUESTION 3: TOP 5 CONDITIONS WITH THEIR FREQUENCIES

conditions_delimeter = {
    "clinicaltrial_2023": "\t",
    "clinicaltrial_2021": ",",
    "clinicaltrial_2020": ",",
}
conditions_column_index = Clinical_RDD_Clean.first().index('Conditions')

Clinical_RDD_Clean.flatMap(lambda x: x[conditions_column_index].split(conditions_delimeter[fileroot])).filter(lambda row: row != 'Conditions').filter
(lambda row: row != '').map(lambda x: (x, 1)).reduceByKey(lambda a,b: a + b).sortBy(lambda x: x[1], ascending=False).take(5)
```

```
Out[19]: [('Healthy', 7997),
 ('Breast Cancer', 4556),
 ('Prostate Cancer', 2650),
 ('Asthma', 2309),
 ('Obesity', 2284)]
```

Command took 16.20 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment

In this code, the index of the ‘Conditions’ column was derived from the header of the RDD then the column was split for each row using the appropriate delimiter. The header and empty rows were filtered out then the remaining rows were counted by the occurrence of each status in descending order. The top 5 most frequent conditions of the sorted RDD were then displayed.

DATA FRAME



```
# QUESTION 3: TOP 5 CONDITIONS WITH THEIR FREQUENCIES

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

# Define delimiter for the Conditions column based on the clinical trial data
conditions_delimiter = {
    "clinicaltrial_2023": "\t",
    "clinicaltrial_2021": ",",
    "clinicaltrial_2020": ",",
}

# Split the Conditions column based on the delimiter for the specific clinical trial data
split_conditions_df = clinical_dataframe \
    .withColumn('Conditions', explode(split(trim(col('Conditions')), conditions_delimiter[fileroot])))

# Group by Conditions, count occurrences, and filter out empty values
result_df = split_conditions_df \
    .groupBy('Conditions') \
    .count() \
    .orderBy('count', ascending=False) \
    .filter("Conditions != ''")

# Show top 5 results
result_df.show(5, truncate=False)
```

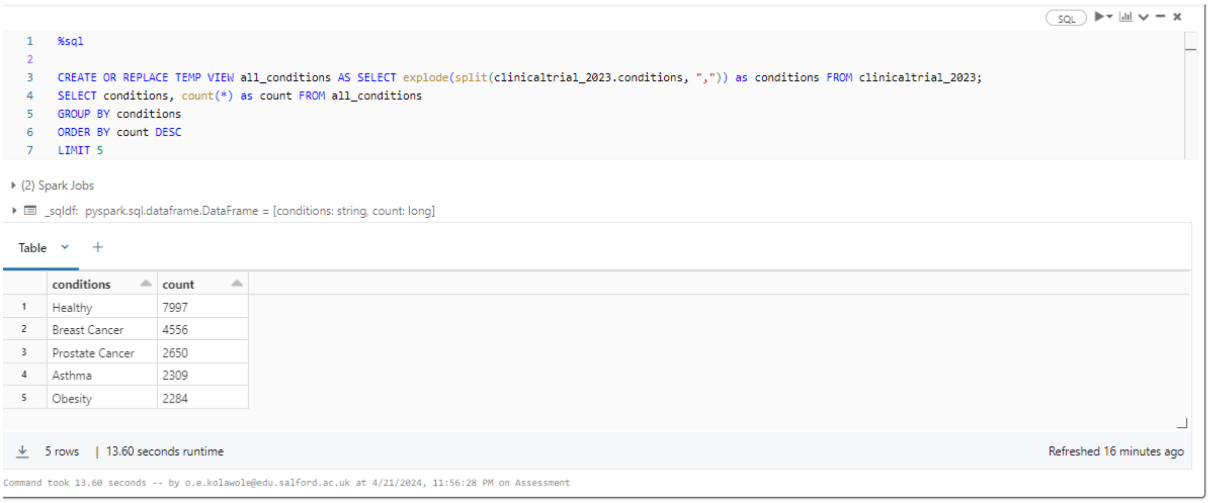
split_conditions_df: pyspark.sql.dataframe.DataFrame = ["Id: string, Study Title: string ... 12 more fields]
result_df: pyspark.sql.dataframe.DataFrame = [Conditions: string, count: long]

```
+---------------+-----+
|Conditions     |count|
+---------------+-----+
|Healthy        |7997 |
|Breast Cancer  |4556 |
|Prostate Cancer|2650 |
|Asthma         |2309 |
|Obesity        |2284 |
+---------------+-----+
only showing top 5 rows
```

Command took 13.59 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:07:25 PM on Assessment

The ‘Conditions’ column was split for each row using the delimiter corresponding to the fileroot variable and a new row was created using the explode function for each condition any empty row is also removed from the condition value. The result was then grouped by ‘Conditions’ column and the number of the occurrences was counted, and it is then ordered in descending order. The top 5 conditions is then displayed with its full name and number of occurrences.

SQL



```
%sql

CREATE OR REPLACE TEMP VIEW all_conditions AS SELECT explode(split(clinicaltrial_2023.conditions, ",")) as conditions FROM clinicaltrial_2023;
SELECT conditions, count(*) as count FROM all_conditions
GROUP BY conditions
ORDER BY count DESC
LIMIT 5
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [conditions: string, count: long]

	conditions	count
1	Healthy	7997
2	Breast Cancer	4556
3	Prostate Cancer	2650
4	Asthma	2309
5	Obesity	2284

5 rows | 13.60 seconds runtime

Refreshed 16 minutes ago

```
Command took 13.60 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:56:28 PM on Assessment
```

Create a temporary view called all_conditions by splitting the clinicaltrial_2023.conditions column by comma delimiter and creating a new row for each condition. Then, select the 'conditions' column and count the number of occurrences using count(*), group results by conditions, order results in descending order, and limit output to top 5 rows.

Discussion of result

With a total of 7,997 occurrences, the results show that the majority of the clinical trial condition is "Healthy." When compared to the total number of trials conducted, the frequency of the top five conditions does not offer significant insights into the clinical trials.

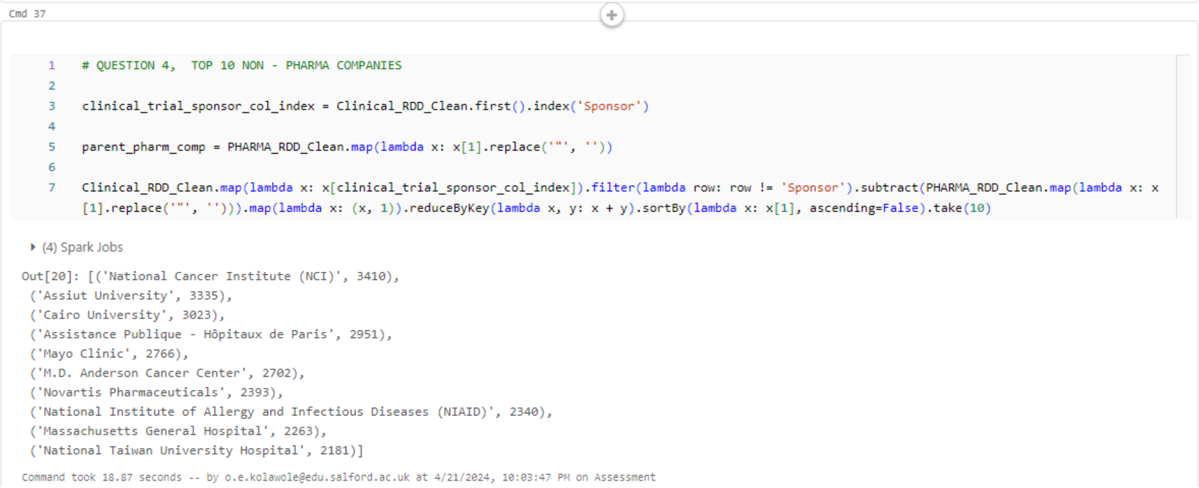
QUESTION 4

Assumptions

Organizations listed in the "Parent Company" column of the "pharma" dataset are assumed to be pharmaceutical companies.

The "Sponsor " column in the "clinicaltrial_2023” comprises the names of companies that are both pharmaceutical and non pharmaceutical.

RDD



Cmd 37

```
# QUESTION 4,  TOP 10 NON - PHARMA COMPANIES

clinical_trial_sponsor_col_index = Clinical_RDD_Clean.first().index('Sponsor')

parent_pharm_comp = PHARMA_RDD_Clean.map(lambda x: x[1].replace('"', ''))

Clinical_RDD_Clean.map(lambda x: x[clinical_trial_sponsor_col_index]).filter(lambda row: row != 'Sponsor').subtract(PHARMA_RDD_Clean.map(lambda x: x
[1].replace('"', ''))).map(lambda x: (x, 1)).reduceByKey(lambda x, y: x + y).sortBy(lambda x: x[1], ascending=False).take(10)
```

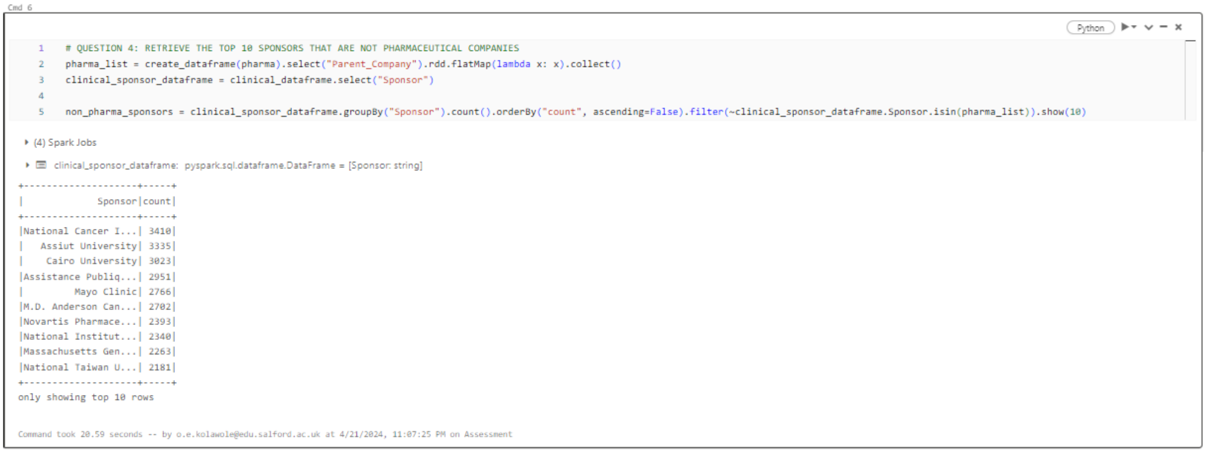
▸ (4) Spark Jobs

```
Out[20]: [('National Cancer Institute (NCI)', 3410),
 ('Assiut University', 3335),
 ('Cairo University', 3023),
 ('Assistance Publique - Hôpitaux de Paris', 2951),
 ('Mayo Clinic', 2766),
 ('M.D. Anderson Cancer Center', 2702),
 ('Novartis Pharmaceuticals', 2393),
 ('National Institute of Allergy and Infectious Diseases (NIAID)', 2340),
 ('Massachusetts General Hospital', 2263),
 ('National Taiwan University Hospital', 2181)]
```

Command took 18.87 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment

The code retrieves the 'Sponsor' column index from the Clinical_RDD_Clean DataFrame, creates an RDD for pharmaceutical companies, and removes double quotes using the replace function. Procedures include mapping the 'Sponsor' column, filtering out 'Sponsor' headers, and subtracting the RDD.

DATA FRAME



Cmd 6

```
# QUESTION 4: RETRIEVE THE TOP 10 SPONSORS THAT ARE NOT PHARMACEUTICAL COMPANIES
pharma_list = create_dataframe(pharma).select("Parent_Company").rdd.flatMap(lambda x: x).collect()
clinical_sponsor_dataframe = clinical_dataframe.select("Sponsor")

non_pharma_sponsors = clinical_sponsor_dataframe.groupBy("Sponsor").count().orderBy("count", ascending=False).filter(~clinical_sponsor_dataframe.Sponsor.isin(pharma_list)).show(10)
```

▸ (4) Spark Jobs

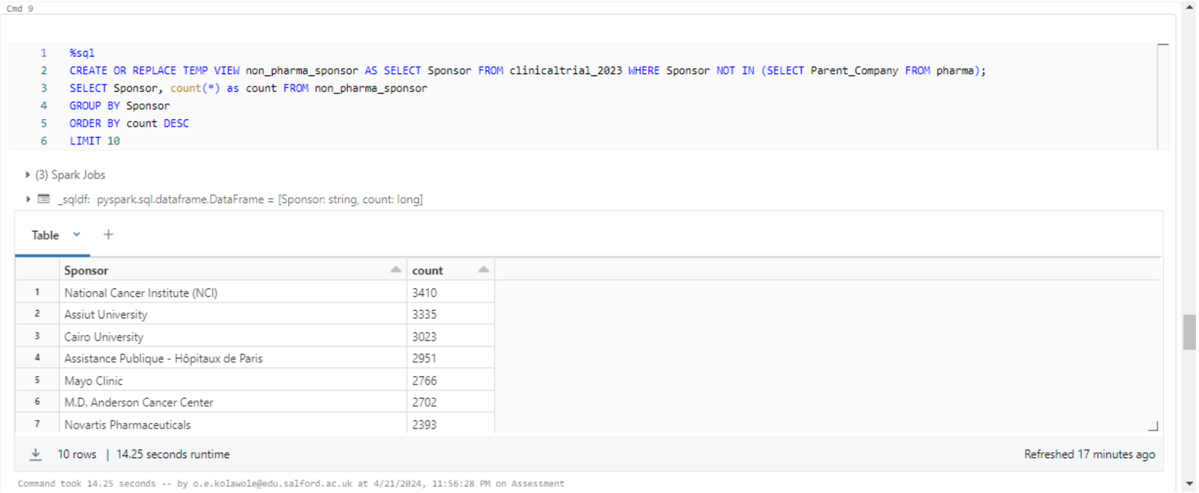
▸ clinical_sponsor_dataframe: pyspark.sql.dataframe.DataFrame = [Sponsor: string]

```
+--------------------+-----+
|             Sponsor|count|
+--------------------+-----+
|National Cancer I...| 3410|
|   Assiut University| 3335|
|    Cairo University| 3023|
|Assistance Publiq...| 2951|
|         Mayo Clinic| 2766|
|M.D. Anderson Can...| 2702|
|Novartis Pharmace...| 2393|
|National Institut...| 2340|
|Massachusetts Gen...| 2263|
|National Taiwan U...| 2181|
+--------------------+-----+
only showing top 10 rows
```

Command took 20.59 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:07:25 PM on Assessment

The pharma dataset is converted into a DataFrame using the create_dataframe(pharma) function. The 'Parent_Company' column is selected and flattened into a single list using the rdd.flatMap(lambda x: x) function. The RDD is materialized and all values are retrieved. A new DataFrame named clinical_sponsor_dataframe is created, containing the 'Sponsor' column. The dataframe is manipulated using groupBy("Sponsor"), count(), orderBy("count"), filter(pharma_list), and show(10).

SQL



```
%sql
CREATE OR REPLACE TEMP VIEW non_pharma_sponsor AS SELECT Sponsor FROM clinicaltrial_2023 WHERE Sponsor NOT IN (SELECT Parent_Company FROM pharma);
SELECT Sponsor, count(*) as count FROM non_pharma_sponsor
GROUP BY Sponsor
ORDER BY count DESC
LIMIT 10
```

_sqldf: pyspark.sql.dataframe.DataFrame = [Sponsor: string, count: long]

	Sponsor	count
1	National Cancer Institute (NCI)	3410
2	Assiut University	3335
3	Cairo University	3023
4	Assistance Publique - Hôpitaux de Paris	2951
5	Mayo Clinic	2766
6	M.D. Anderson Cancer Center	2702
7	Novartis Pharmaceuticals	2393

10 rows | 14.25 seconds runtime

Refreshed 17 minutes ago

Command took 14.25 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:56:28 PM on Assessment

A temporary view named non_pharma_sponsor is created to filter out non-pharmaceutical sponsors from the clinicaltrial_2023 table. The WHERE clause removes sponsors from the pharma table, allowing only non-pharmaceutical sponsors. The SQL query uses the 'Sponsor' column to determine the number of sponsor appearances, group results, sort outcomes in descending order, and restrict output to the top ten rows. The results are sorted in descending order.

Discussion of result

The top non-pharmaceutical company sponsoring clinical trials is National Cancer Institute (NCI) having a frequency of 3410.

These results highlight a diverse range of organizations, including research institutions, hospitals, and government agencies, actively sponsoring clinical trials.

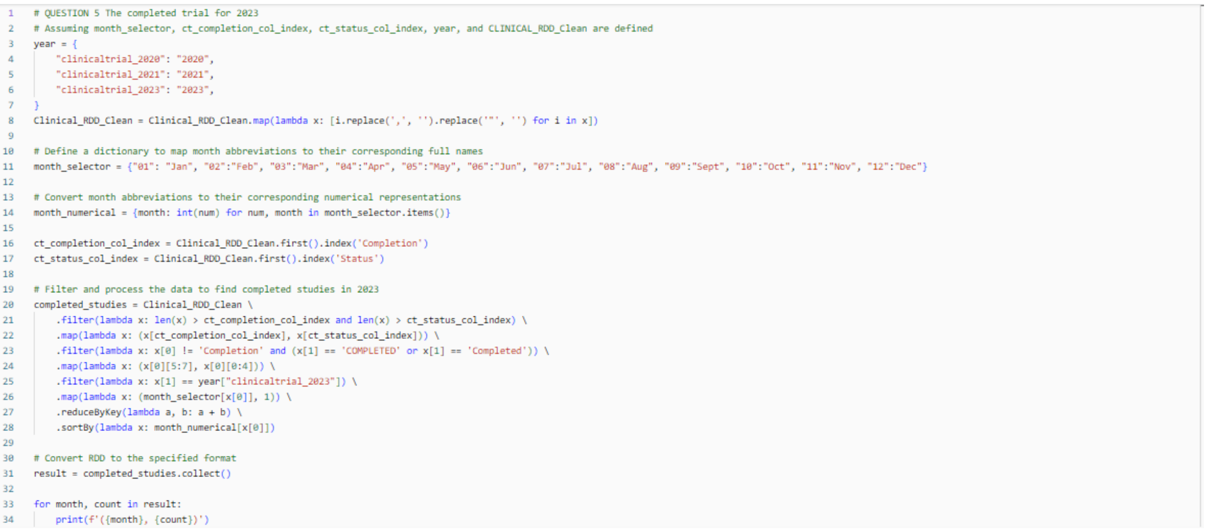
QUESTION 5

Assumptions

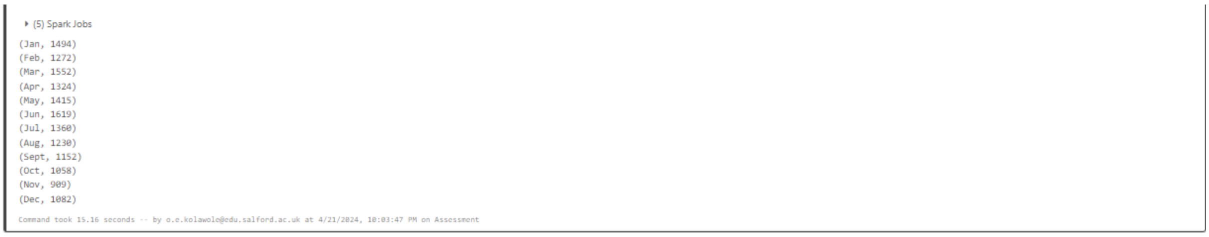
The "Start" column in the "clinicaltrial_2023 " dataset contains the start date of each clinical trial.

The “Completion” column contains the completed studies per month.

RDD



```python
# QUESTION 5 The completed trial for 2023
# Assuming month_selector, ct_completion_col_index, ct_status_col_index, year, and CLINICAL_RDD_Clean are defined
year = {
    "clinicaltrial_2020": "2020",
    "clinicaltrial_2021": "2021",
    "clinicaltrial_2023": "2023",
}
Clinical_RDD_Clean = Clinical_RDD_Clean.map(lambda x: [i.replace(',', '').replace('"', '') for i in x])

# Define a dictionary to map month abbreviations to their corresponding full names
month_selector = {"01": "Jan", "02":"Feb", "03":"Mar", "04":"Apr", "05":"May", "06":"Jun", "07":"Jul", "08":"Aug", "09":"Sept", "10":"Oct", "11":"Nov", "12":"Dec"}

# Convert month abbreviations to their corresponding numerical representations
month_numerical = {month: int(num) for num, month in month_selector.items()}

ct_completion_col_index = Clinical_RDD_Clean.first().index('Completion')
ct_status_col_index = Clinical_RDD_Clean.first().index('Status')

# Filter and process the data to find completed studies in 2023
completed_studies = Clinical_RDD_Clean \
    .filter(lambda x: len(x) > ct_completion_col_index and len(x) > ct_status_col_index) \
    .map(lambda x: (x[ct_completion_col_index], x[ct_status_col_index])) \
    .filter(lambda x: x[0] != 'Completion' and (x[1] == 'COMPLETED' or x[1] == 'Completed')) \
    .map(lambda x: (x[0][5:7], x[0][0:4])) \
    .filter(lambda x: x[1] == year["clinicaltrial_2023"]) \
    .map(lambda x: (month_selector[x[0]], 1)) \
    .reduceByKey(lambda a, b: a + b) \
    .sortBy(lambda x: month_numerical[x[0]])

# Convert RDD to the specified format
result = completed_studies.collect()

for month, count in result:
    print(f'({month}, {count})')
```

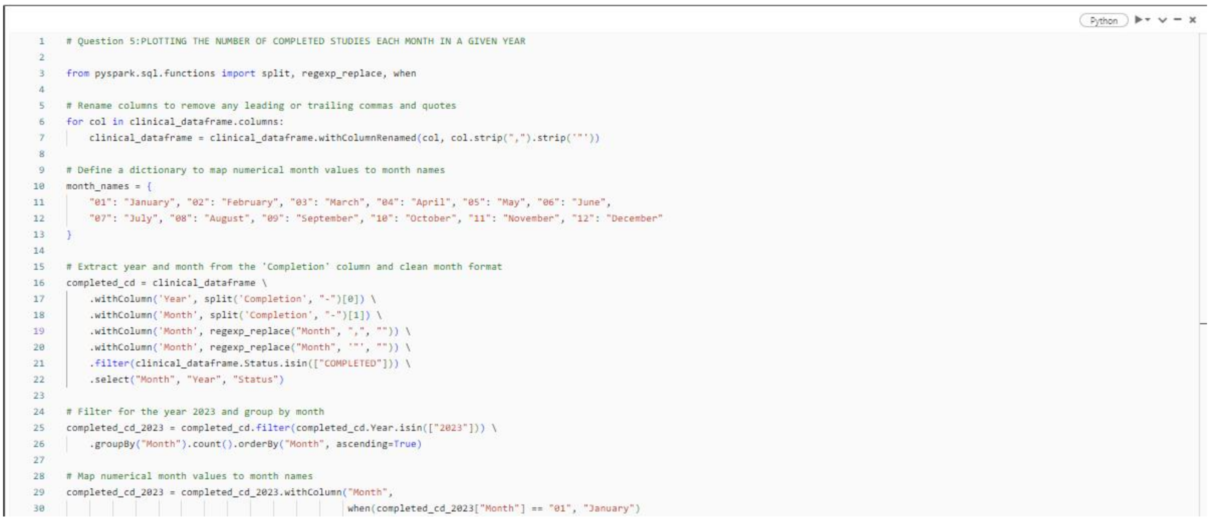


```
(Jan, 1494)
(Feb, 1272)
(Mar, 1552)
(Apr, 1324)
(May, 1415)
(Jun, 1619)
(Jul, 1360)
(Aug, 1230)
(Sept, 1152)
(Oct, 1058)
(Nov, 909)
(Dec, 1082)
```

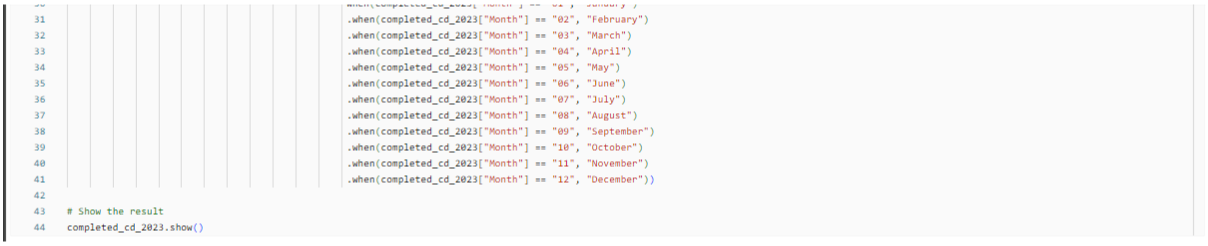
Command took 15.16 seconds -- by o.e.kolawolo@edu.salford.ac.uk at 4/21/2024, 10:03:47 PM on Assessment

The Clinical RDD Clean process maps the year of clinical trial data to the fileroot variable. It removes quotes and commas from values, associates month abbreviations with numerical values, and locates the Clinical RDD_Clean RDD's index for the 'Completion' column. The primary processing involves applying the Clinical RDD Clean filter, extracting the month and year from the 'Completion' column, filtering the data to include records from 2023, and mapping month values to abbreviations. The function reduces and sorts the findings in ascending order, and the resultant (month, count) pairs are printed.

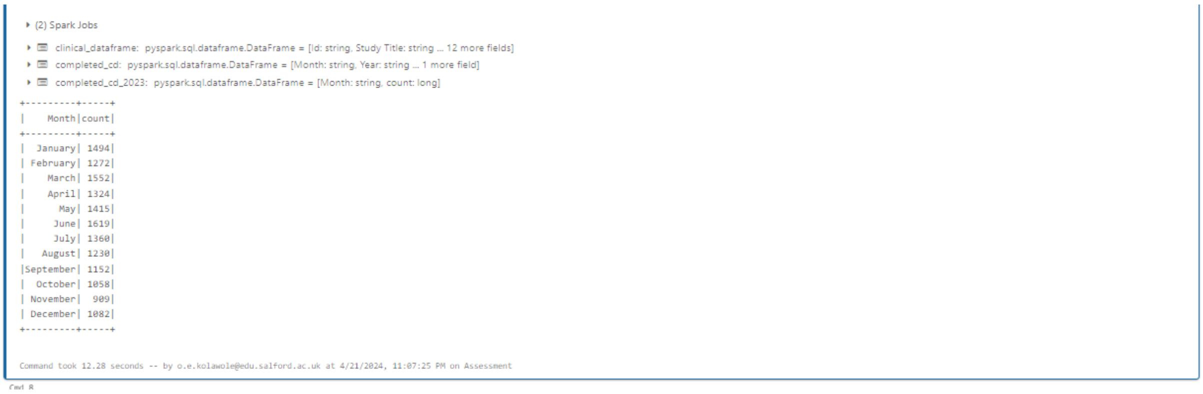
DATAFRAME



```python
# Question 5:PLOTTING THE NUMBER OF COMPLETED STUDIES EACH MONTH IN A GIVEN YEAR

from pyspark.sql.functions import split, regexp_replace, when

# Rename columns to remove any leading or trailing commas and quotes
for col in clinical_dataframe.columns:
    clinical_dataframe = clinical_dataframe.withColumnRenamed(col, col.strip(",").strip('"'))

# Define a dictionary to map numerical month values to month names
month_names = {
    "01": "January", "02": "February", "03": "March", "04": "April", "05": "May", "06": "June",
    "07": "July", "08": "August", "09": "September", "10": "October", "11": "November", "12": "December"
}

# Extract year and month from the 'Completion' column and clean month format
completed_cd = clinical_dataframe \
    .withColumn('Year', split('Completion', "-")[0]) \
    .withColumn('Month', split('Completion', "-")[1]) \
    .withColumn('Month', regexp_replace("Month", ",", "")) \
    .withColumn('Month', regexp_replace("Month", '"', "")) \
    .filter(clinical_dataframe.Status.isin(["COMPLETED"])) \
    .select("Month", "Year", "Status")

# Filter for the year 2023 and group by month
completed_cd_2023 = completed_cd.filter(completed_cd.Year.isin(["2023"])) \
    .groupBy("Month").count().orderBy("Month", ascending=True)

# Map numerical month values to month names
completed_cd_2023 = completed_cd_2023.withColumn("Month",
                                                when(completed_cd_2023["Month"] == "01", "January")
```



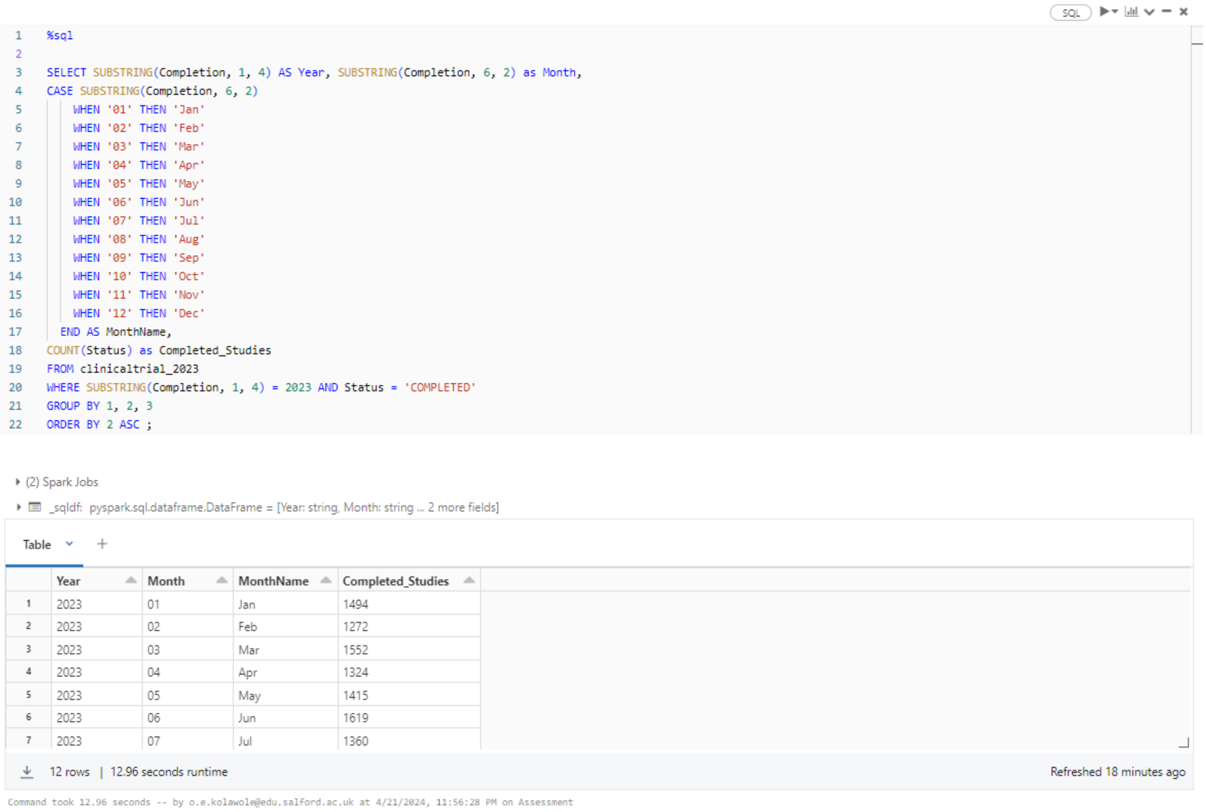
```python
                                                .when(completed_cd_2023["Month"] == "02", "February")
                                                .when(completed_cd_2023["Month"] == "03", "March")
                                                .when(completed_cd_2023["Month"] == "04", "April")
                                                .when(completed_cd_2023["Month"] == "05", "May")
                                                .when(completed_cd_2023["Month"] == "06", "June")
                                                .when(completed_cd_2023["Month"] == "07", "July")
                                                .when(completed_cd_2023["Month"] == "08", "August")
                                                .when(completed_cd_2023["Month"] == "09", "September")
                                                .when(completed_cd_2023["Month"] == "10", "October")
                                                .when(completed_cd_2023["Month"] == "11", "November")
                                                .when(completed_cd_2023["Month"] == "12", "December"))

# Show the result
completed_cd_2023.show()
```



```
▸ (2) Spark Jobs
▸ clinical_dataframe:  pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 12 more fields]
▸ completed_cd:  pyspark.sql.dataframe.DataFrame = [Month: string, Year: string ... 1 more field]
▸ completed_cd_2023:  pyspark.sql.dataframe.DataFrame = [Month: string, count: long]
+---------+-----+
|    Month|count|
+---------+-----+
|  January| 1494|
| February| 1272|
|    March| 1552|
|    April| 1324|
|      May| 1415|
|     June| 1619|
|     July| 1360|
|   August| 1230|
|September| 1152|
|  October| 1058|
| November|  909|
| December| 1082|
+---------+-----+

Command took 12.28 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:07:25 PM on Assessment
```

The query extracts year and month values from the 'Completion' column using the SUBSTRING function. The month name is assigned to the MonthName column. The total number of completed studies for each month is counted. The data is filtered to include records from 2023. Results are grouped by year, month, and month name. The results are sorted by month in ascending order. The table displays year, month, month name, and finalized_studies.

SQL



```
%sql

SELECT SUBSTRING(Completion, 1, 4) AS Year, SUBSTRING(Completion, 6, 2) as Month,
CASE SUBSTRING(Completion, 6, 2)
    WHEN '01' THEN 'Jan'
    WHEN '02' THEN 'Feb'
    WHEN '03' THEN 'Mar'
    WHEN '04' THEN 'Apr'
    WHEN '05' THEN 'May'
    WHEN '06' THEN 'Jun'
    WHEN '07' THEN 'Jul'
    WHEN '08' THEN 'Aug'
    WHEN '09' THEN 'Sep'
    WHEN '10' THEN 'Oct'
    WHEN '11' THEN 'Nov'
    WHEN '12' THEN 'Dec'
  END AS MonthName,
COUNT(Status) as Completed_Studies
FROM clinicaltrial_2023
WHERE SUBSTRING(Completion, 1, 4) = 2023 AND Status = 'COMPLETED'
GROUP BY 1, 2, 3
ORDER BY 2 ASC ;
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Year: string, Month: string ... 2 more fields]

	Year	Month	MonthName	Completed_Studies
1	2023	01	Jan	1494
2	2023	02	Feb	1272
3	2023	03	Mar	1552
4	2023	04	Apr	1324
5	2023	05	May	1415
6	2023	06	Jun	1619
7	2023	07	Jul	1360

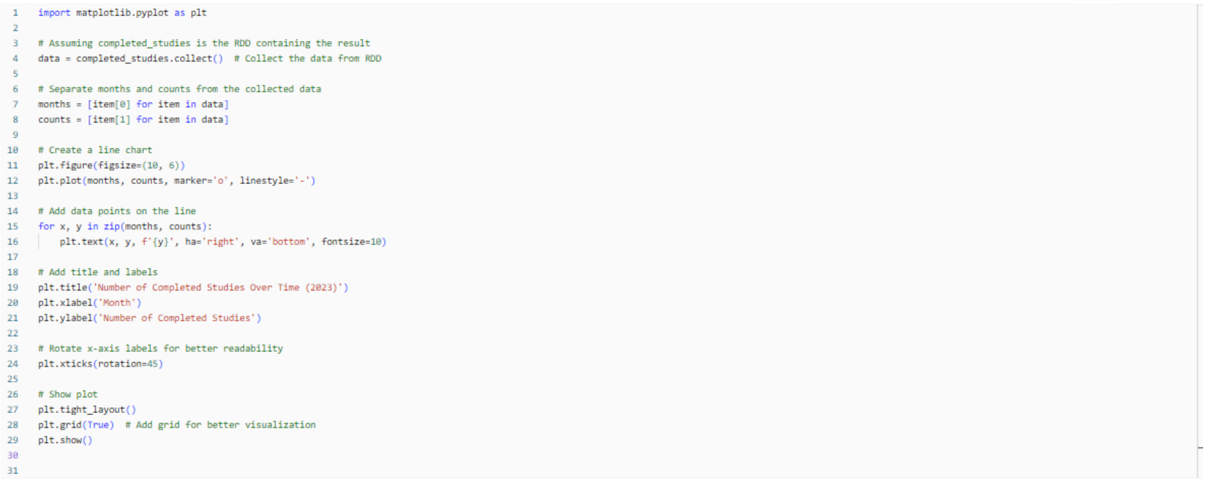
12 rows | 12.96 seconds runtime

Refreshed 18 minutes ago

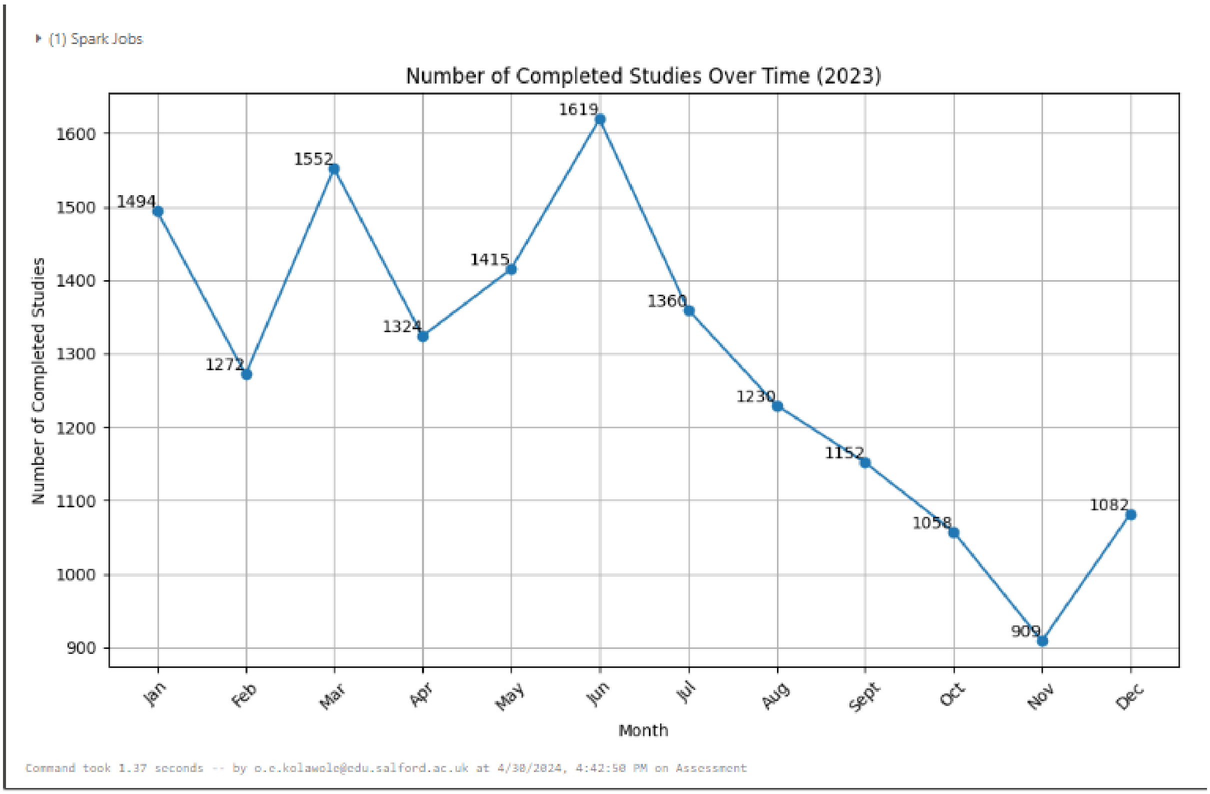
Command took 12.96 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:56:28 PM on Assessment

The query extracts year and month values from the 'Completion' column using the SUBSTRING function. The month name is assigned to the MonthName column. The total number of completed studies for each month is counted. The data is filtered to include records from 2023. Results are grouped by year, month, and month name. The results are sorted by month in ascending order. The table displays year, month, month name, and finalized_studies.

Discussion of Result



```python
import matplotlib.pyplot as plt

# Assuming completed_studies is the RDD containing the result
data = completed_studies.collect()  # Collect the data from RDD

# Separate months and counts from the collected data
months = [item[0] for item in data]
counts = [item[1] for item in data]

# Create a line chart
plt.figure(figsize=(10, 6))
plt.plot(months, counts, marker='o', linestyle='-')

# Add data points on the line
for x, y in zip(months, counts):
    plt.text(x, y, f'{y}', ha='right', va='bottom', fontsize=10)

# Add title and labels
plt.title('Number of Completed Studies Over Time (2023)')
plt.xlabel('Month')
plt.ylabel('Number of Completed Studies')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show plot
plt.tight_layout()
plt.grid(True)  # Add grid for better visualization
plt.show()


```



▸ (1) Spark Jobs

```
Command took 1.37 seconds -- by o.c.kolawole@edu.salford.ac.uk at 4/30/2024, 4:42:50 PM on Assessment
```

The code creates a line chart using **Phyton’s Matplotlib** to visualize the number of completed studies over time. It assumes `completed_studies` is an RDD containing the result and collects data into lists. The chart is created using `plt.plot()` with `months` as the x-axis and `counts` as the y-axis. Data points are added, titles, x-labels, and y-labels set, and the x-axis labels rotated for better readability.

From the result June has the highest number of completed studies of 1,619 clinical trials and the month has the lowest number of completed studies of November is 909 clinical trials. This can be due to factors such as seasonal trends, study duration, funding availability, study duration etc.

FURTHER ANALYSIS

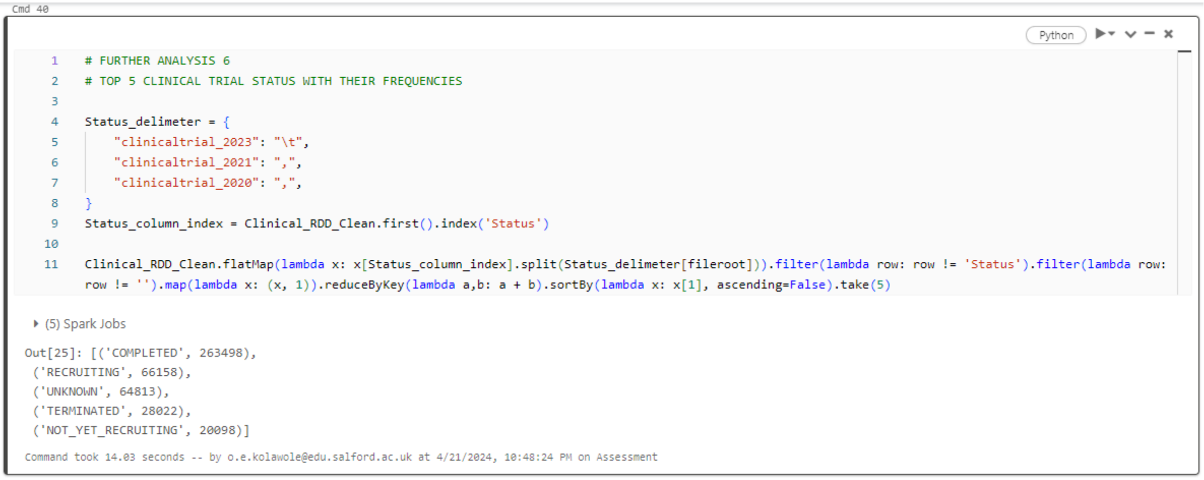
QUESTION 6

The top Status (from Status) with their frequencies.

RDD

Assumptions

The "Status" column in the “clinical trial 2023” contains multiple status values based on the level the trial has progressed.



```python
# FURTHER ANALYSIS 6
# TOP 5 CLINICAL TRIAL STATUS WITH THEIR FREQUENCIES

Status_delimeter = {
    "clinicaltrial_2023": "\t",
    "clinicaltrial_2021": ",",
    "clinicaltrial_2020": ",",
}
Status_column_index = Clinical_RDD_Clean.first().index('Status')

Clinical_RDD_Clean.flatMap(lambda x: x[Status_column_index].split(Status_delimeter[fileroot])).filter(lambda row: row != 'Status').filter(lambda row: row != '').map(lambda x: (x, 1)).reduceByKey(lambda a,b: a + b).sortBy(lambda x: x[1], ascending=False).take(5)
```

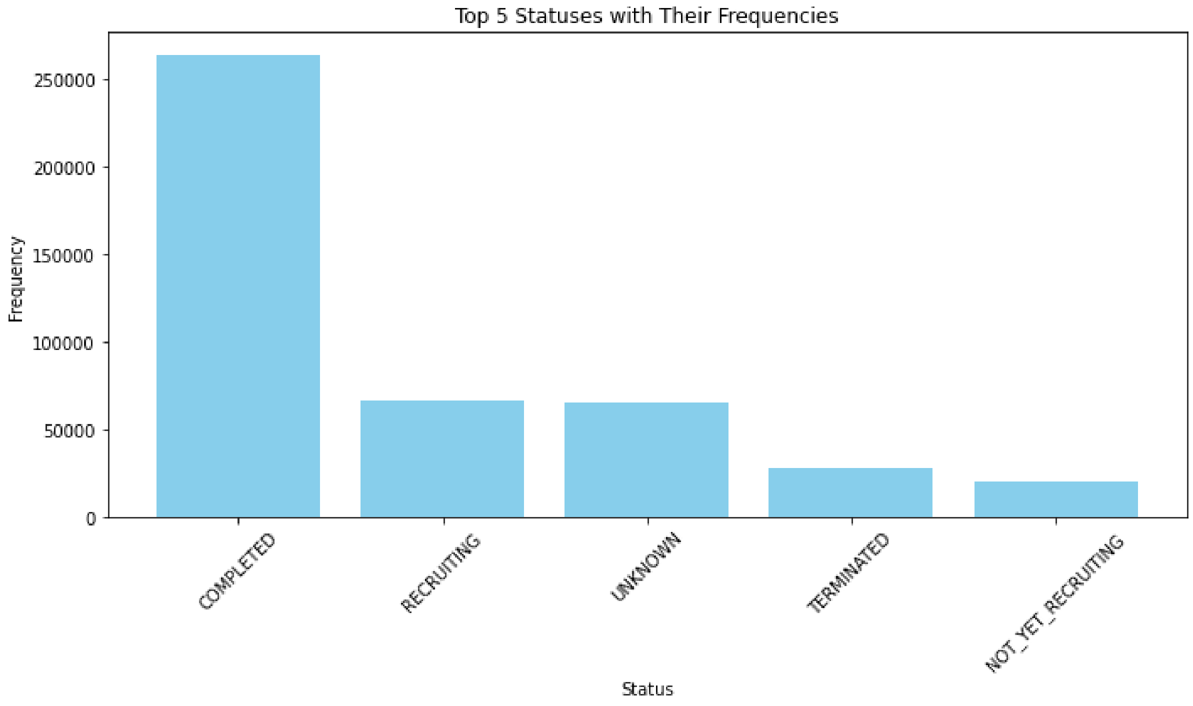
▸ (5) Spark Jobs

```
Out[25]: [('COMPLETED', 263498),
 ('RECRUITING', 66158),
 ('UNKNOWN', 64813),
 ('TERMINATED', 28022),
 ('NOT_YET_RECRUITING', 20098)]
```

Command took 14.03 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 10:48:24 PM on Assessment

In this code, the index of the ‘Status’ column was derived from the header of the RDD then the column is split for each row using the appropriate delimiter. The header and empty rows were filtered out then the remaining rows was counted by the occurrence of each status in descending order. The top 15 most frequent status of the sorted RDD was then displayed.

Discussion of result



The result shows the distribution of different statuses within the clinical trial dataset, along with their respective frequencies. A considerable portion of the trials in the dataset have been completed. These trials have finished their data collection, analysis, and reporting phases.

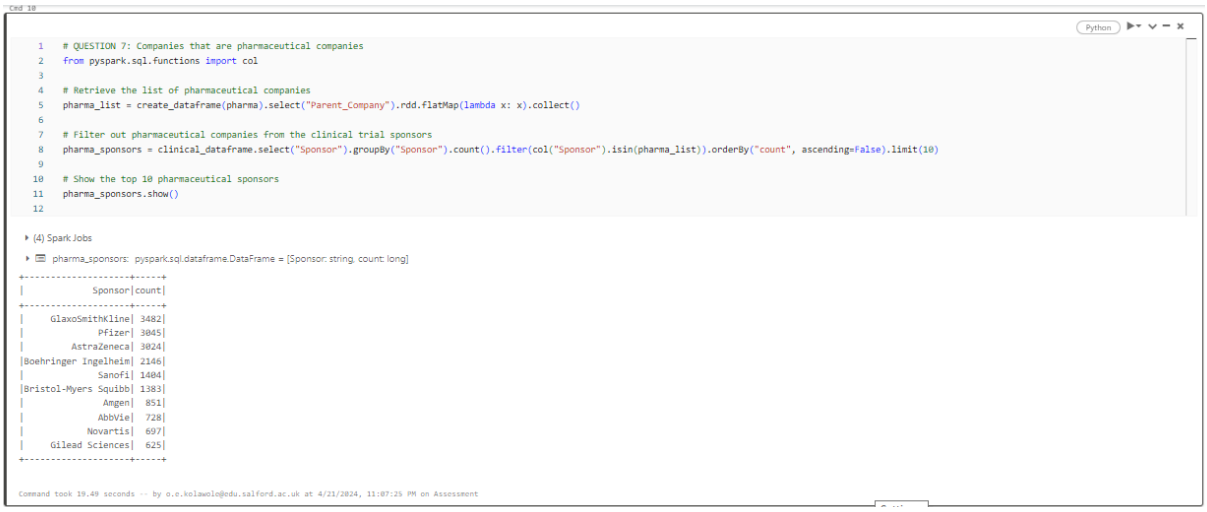
QUESTION 7

Find the 10 most common sponsors that are pharmaceutical companies, along with the number of clinical trials they have sponsored.

DATA FRAME

Assumptions

- Organizations listed in the "Parent Company" column of the "pharma" dataset are assumed to be pharmaceutical companies.
- The "Sponsor " column in the "clinicaltrial_2023” comprises the names of companies that are both pharmaceutical and non-pharmaceutical.



```python
# QUESTION 7: Companies that are pharmaceutical companies
from pyspark.sql.functions import col

# Retrieve the list of pharmaceutical companies
pharma_list = create_dataframe(pharma).select("Parent_Company").rdd.flatMap(lambda x: x).collect()

# Filter out pharmaceutical companies from the clinical trial sponsors
pharma_sponsors = clinical_dataframe.select("Sponsor").groupBy("Sponsor").count().filter(col("Sponsor").isin(pharma_list)).orderBy("count", ascending=False).limit(10)

# Show the top 10 pharmaceutical sponsors
pharma_sponsors.show()

```

```
▸ (4) Spark Jobs
▸ pharma_sponsors:  pyspark.sql.dataframe.DataFrame = [Sponsor: string, count: long]
+--------------------+-----+
|             Sponsor|count|
+--------------------+-----+
|      GlaxoSmithKline| 3482|
|              Pfizer| 3045|
|         AstraZeneca| 3024|
|Boehringer Ingelheim| 2146|
|              Sanofi| 1404|
|Bristol-Myers Squibb| 1383|
|               Amgen|  851|
|              AbbVie|  728|
|            Novartis|  697|
|     Gilead Sciences|  625|
+--------------------+-----+

Command took 19.49 seconds -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:07:25 PM on Assessment
```

A pharma_list is created from the pharma dataset, containing unique parent company names. The dataframe is filtered by the sponsor column, ensuring only sponsors on the pharma list are included, and displayed in descending order.

Discussion of result

The top -pharmaceutical company sponsoring clinical trials is GlaxoSmithKline having a frequency of 3482.

The output can be used to compare the sponsorship efforts of various pharmaceutical companies or to assess how dominant pharmaceutical companies are in the field of clinical trials.

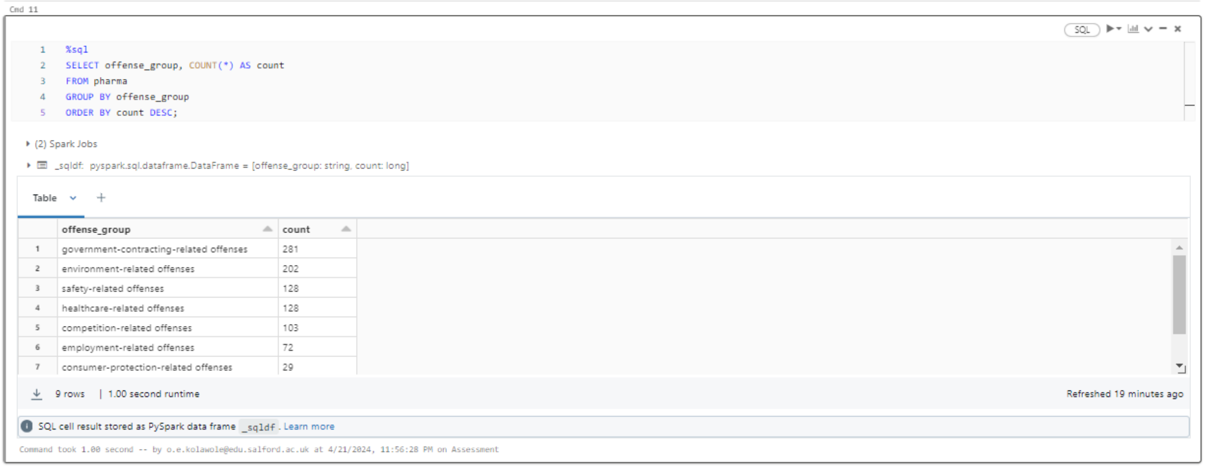
QUESTION 8

Find the top offense commited by the pharmaceutical companies and frequencies

SQL

Assumption

The offense column is in the pharma file.



```
%sql
SELECT offense_group, COUNT(*) AS count
FROM pharma
GROUP BY offense_group
ORDER BY count DESC;
```

_sqldf: pyspark.sql.dataframe.DataFrame = [offense_group: string, count: long]

	offense_group	count
1	government-contracting-related offenses	281
2	environment-related offenses	202
3	safety-related offenses	128
4	healthcare-related offenses	128
5	competition-related offenses	103
6	employment-related offenses	72
7	consumer-protection-related offenses	29

9 rows | 1.00 second runtime

Refreshed 19 minutes ago

SQL cell result stored as PySpark data frame _sqldf. Learn more

```
Command took 1.00 second -- by o.e.kolawole@edu.salford.ac.uk at 4/21/2024, 11:56:28 PM on Assessment
```

The query will group the pharma file by its offense group, count the number for each offense, order them by count in descending order, and show its result using the ‘FROM’ and ‘SELECT’ Statements.

Discussion of result

The most common offense the pharmaceutical companies commit is “government contracting related offenses” having a frequency of 281 occurrences.