University of Salford
MANCHESTER

University of Salford
Big Data Tools and Techniques

MICHAEL IFECHUKWU OLU
@00755387

TASK 1

1. DESCRIPTION OF SETUP

The general aim of this task is to prepare, clean and perform various analysis on the clinical trial dataset of the year 2023 (clinicaltrial_2023). We will also perform these tasks on the datasets of the previous year's (2021 and 2020).

These tasks will be accomplished using the **Databricks** platform. There will be some initial set up required before the platform can be properly utilized for tasks. The initial setup are as follows:

- Creation of a **compute cluster** (Figure 1.1), which will provide us some processing power to be able to run our codes.

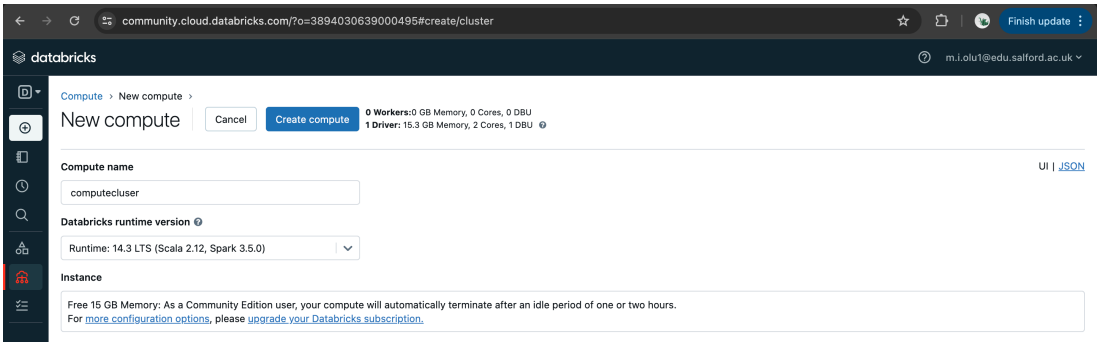


Figure 1.1

- Creation of a **Databricks notebook** (Figure 1.2). This is an IDE (Integrated development environment) where we will be able to write our codes and run them.

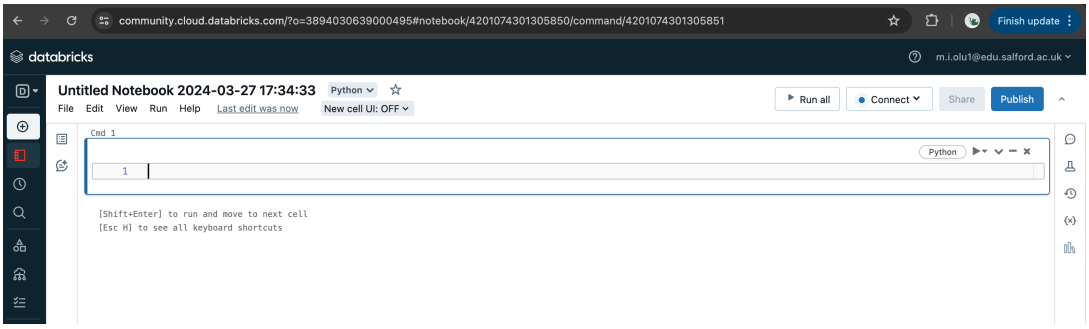


Figure 1.2

After the creation of the cluster and notebooks, we can go ahead and upload our datasets in a zip file format into our DBFS (Databricks file system) as shown in Figure 1.3.

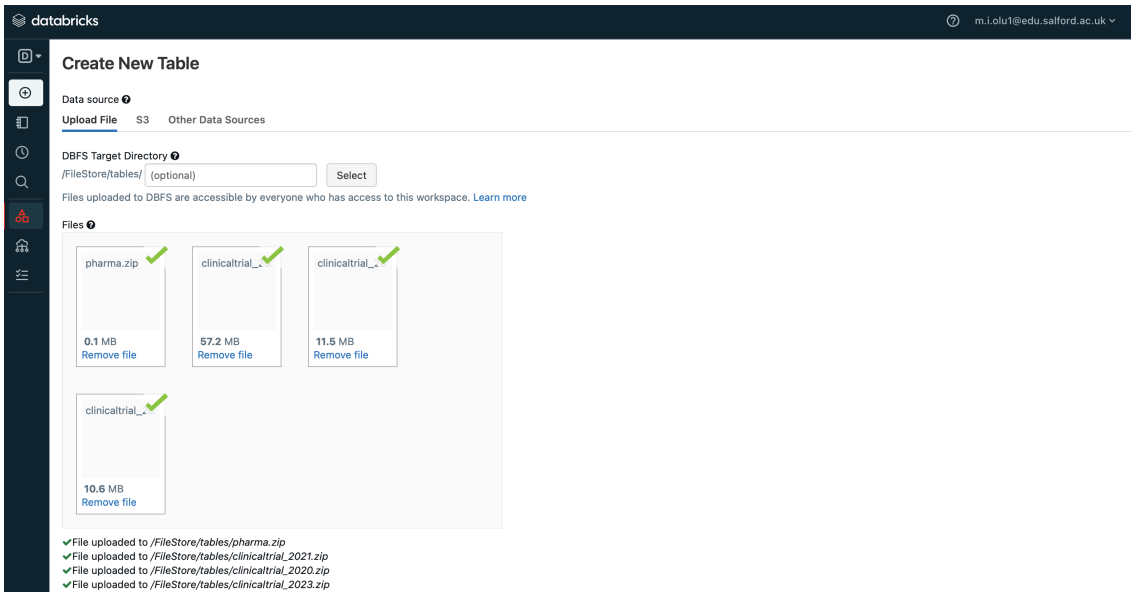


Figure 1.3

The following files were uploaded into the DBFS as shown in figure 1.3

- clinicaltiral_2023.zip
- clinicaltiral_2021.zip
- clinicaltiral_2020.zip
- pharma.zip

2. DATA CLEANING AND PREPARATION

The files in the previous section are in zip formats, which will be unzipped using our Databricks notebook using the following processes.

i. Confirmation of files in the dbfs

The initial step involves viewing the Databricks file system to verify the presence of our files using the built-in library called **dbutils.**

Figure 2.1

Figure 2.1 reveals that all the uploaded zip files (clinicaltrial_2023.zip, clinicaltrial_2021.zip, clinicaltrial_2020.zip, and pharma.zip) are present in the file system.

Figure 2.2

The image displays variable assignment to zip file clinicaltrial_2023.zip and pharma.zip, allowing for file analysis of files like "clinicaltrial_2020.zip" and "clinicaltrial_2020.zip".

We also made use of a **try-except** block to skip this step incase our zip file does not exist or has been unzipped already.

ii. Copying files from the dbfs to the local file system

In Figure 2.3, the required files were copied to the local file system for unzipping using Linux commands, as Linux commands cannot be executed on the DBFS.

Figure 2.3

iii. Unzipping the files in the local file system using Linux commands.

For this step, as seen in *Figure 2.4*, we created variables in the operating system environment which can be accessed by the Linux command line.

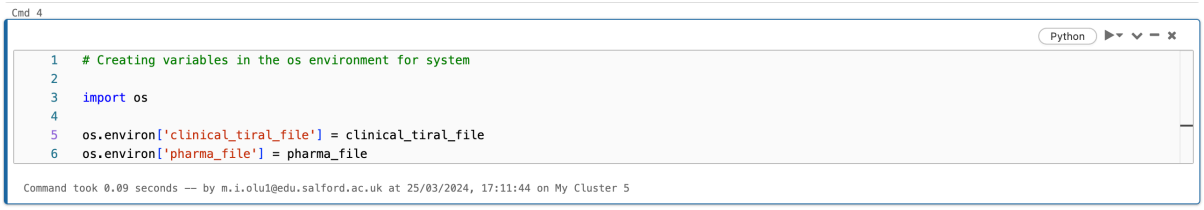


Figure 2.4

After creating our variables, we used this variables in place of file paths when unzipping in the Linux command line as seen in *Figure 2.5*.

Figure 2.5

iv. Moving the unzipped files from the local file system back to dbfs

The files were extracted and stored in the local file system after unzipping, and then moved back to the DBFS as shown in *Figure 2.6.*

Cmd 6

```python
# Moving unzipped files back to the Databricks file system

def move_file_to_dbfs(_file):
    try:
        dbutils.fs.mv(f"file:/tmp/{_file}.csv", f"/FileStore/tables/{_file}.csv", )
    except Exception:
        pass

move_file_to_dbfs(clinical_tiral_file)
move_file_to_dbfs(pharma_file)
```

Command took 0.76 seconds -- by m.i.olu1@edu.salford.ac.uk at 25/03/2024, 17:11:44 on My Cluster 5

Figure 2.6

v. Deleting the zip files from the dbfs after unzipping has been completed

After we have unzipped and moved our csv files to the appropriate location, we can get rid of the unnecessary zip files which will no longer be required going forward. See Figure 2.7

Cmd 7

```python
# Removing the zip files from the directory after it has served its purpose

def clean_directory(_file):
    try:
        dbutils.fs.rm(f"/FileStore/tables/{_file}.zip")
    except Exception:
        pass

clean_directory(clinical_tiral_file)
clean_directory(pharma_file)
```

Command took 0.29 seconds -- by m.i.olu1@edu.salford.ac.uk at 27/03/2024, 22:59:36 on My Cluster 5

Figure 2.7

FILE SAMPLE VIEWING

We have prepared our csv files in DBFS and can now create dataset objects. Before proceeding, we should review the file contents to understand their structure.

We can accomplish this by using **dbutils** to view the head of the file. See Figure 2.8.

clinicaltrial_2023.csv

```python
dbutils.fs.head(f"/FileStore/tables/{clinical_tiral_file}.csv")
```

```
[Truncated to first 65536 bytes]
'"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\tMental Health Issue
(E.G.", Depression, Psychosis, Personality Disorder," Substance Abuse)\tBEHAVIORAL: PRIDE \'Step 1\' problem-solving intervention|BEHAVIORAL: Enhanced usual c
are\tSangath\tHarvard Medical School (HMS and HSDM)|London School of Hygiene and Tropical Medicine\t250.0\tOTHER\tINTERVENTIONAL\tAllocation: RANDOMIZED|Inter
vention Model: PARALLEL|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\tOTHER: Placebo
|DIETARY_SUPPLEMENT: β-OHB\tMcMaster University\tAlzheimer\'s Society of Brant", Haldimand Norfolk," Hamilton Halton\t30.0\tOTHER\tINTERVENTIONAL\tAllocation:
RANDOMIZED|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: R
ANDOMIZED|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\tSUPERMELD\tRECRUITING\t
Decompensated Cirrhosis|Liver Transplantation\tOTHER: SuperMELD\tAssistance Publique - Hôpitaux de Paris\t\t500.0\tOTHER\tINTERVENTIONAL\tAllocation: NA|Inter
```

Command took 0.30 seconds -- by m.i.olu1@edu.salford.ac.uk at 27/03/2024, 22:59:36 on My Cluster 5

The clinical trial datasets for 2021, 2020 are analyzed, providing details on the file's nature, delimiter, and column count.

clinicaltrial_2021.csv

```
Cmd 8
                                                                                    Python
1   dbutils.fs.head(f"/FileStore/tables/{clinical_tiral_file}.csv")

[Truncated to first 65536 bytes]
"Id|Sponsor|Status|Start|Completion|Type|Submission|Conditions|Interventions\r\nNCT02758028|The University of Hong Kong|Recruiting|Aug 2005|Nov 2021|Intervent
ional|Apr 2016||\r\nNCT02751957|Duke University|Completed|Jul 2016|Jul 2020|Interventional|Apr 2016|Autistic Disorder,Autism Spectrum Disorder|\r\nNCT02758483
|Universidade Federal do Rio de Janeiro|Completed|Mar 2017|Jan 2018|Interventional|Apr 2016|Diabetes Mellitus|\r\nNCT02759848|Istanbul Medeniyet University|Co
mpleted|Jan 2012|Dec 2014|Observational|May 2016|Tuberculosis,Lung Diseases,Pulmonary Disease|\r\nNCT02758860|University of Roma La Sapienza|Active, not recru
iting|Jun 2016|Sep 2020|Observational [Patient Registry]|Apr 2016|Diverticular Diseases,Diverticulum,Diverticulosis|\r\nNCT02757209|Consorzio Futuro in Ricerc
a|Completed|Apr 2016|Jan 2018|Interventional|Apr 2016|Asthma|Fluticasone,Xhance,Budesonide,Formoterol Fumarate,Salmeterol Xinafoate\r\nNCT02752438|Ankara Univ
ersity|Unknown status|May 2016|Jul 2017|Observational [Patient Registry]|Apr 2016|Hypoventilation|\r\nNCT02753543|Ruijin Hospital|Unknown status|Nov 2015|Nov
2019|Interventional|Apr 2016|Lymphoma|\r\nNCT02757508|Washington University School of Medicine|Completed|Mar 2016|Jul 2017|Interventional|Apr 2016||Vitamins\r
\nNCT02753530|Orphazyme|Completed|Aug 2017|Jan 2021|Interventional|Apr 2016|Myositis|\r\nNCT02754817|Novo Nordisk A/S|Completed|Apr 2016|Oct 2016|Observationa
l|Apr 2016|Diabetes Mellitus|Liraglutide,Xultophy\r\nNCT02759276|Daniel Alexandre Bottino|Completed|May 2015|Dec 2015|Observational|Apr 2016|Hypertension|\r\n
NCT02750956|Bulent Ecevit University|Completed|Jun 2015|Mar 2016|Observational|Apr 2016|Periodontal Diseases|\r\nNCT02752113|Institut für Pharmakologie und Pr
äventive Medizin|Completed|Apr 2016|May 2019|Interventional|Apr 2016|Diabetes Mellitus|Metformin,Empagliflozin,Linagliptin\r\nNCT02752698|The Third Xiangya Ho
spital of Central South University|Active, not recruiting|Jan 2015|Dec 2021|Interventional|Jun 2015|Appendicitis,Stomach Ulcer,Cholecystolithiasis,Cholelithia
sis,Gallstones|\r\nNCT02755779|Tel Aviv Medical Center|Unknown status|Jun 2016|Jun 2017|Observational|Apr 2016||\r\nNCT02750384|Medicines for Malaria Venture|
Terminated|May 2016|Jul 2016|Interventional|Apr 2016||\r\nNCT02754609|James Cook University, Queensland, Australia|Completed|Sep 2016|Oct 2019|Interventional|
Apr 2016|Hookworm Infections,Celiac Disease|\r\nNCT02755701|Soonchunhyang University Hospital|Unknown status|Jul 2016|Dec 2018|Interventional|Apr 2016|Ascites
|\r\nNCT02751762|Member Companies of the Opioid PMR Consortium|Recruiting|Nov 2017|Oct 2022|Observational|Apr 2016|Chronic Pain,Substance-Related Disorders,Op
ioid-Related Disorders,Narcotic-Related Disorders,Behavior|\r\nNCT02756299|Marmara University|Completed|Jun 2014|Apr 2015|Interventional|Apr 2016|Sleep Apnea
Syndromes,Sleep Apnea|\r\nNCT02750709|Cycle Pharmaceuticals Ltd.|Completed|Oct 2015|Jan 2016|Interventional|Apr 2016|Tyrosinemias|Nitisinone\r\nNCT02753907|Yo
nsei University|Completed|Jun 2015||Interventional|Apr 2016||\r\nNCT02755467|Cutera Inc.|Completed|May 2016|Apr 2017|Interventional|Apr 2016|Hemangioma|\r\nNC
T02755298|University of Zurich|Completed|Oct 2016|Nov 2020|Interventional|Mar 2016|Hypertension|Acetazolamide\r\nNCT02759614|Guangdong Association of Clinical
Command took 0.20 seconds -- by m.i.olu1@edu.salford.ac.uk at 27/03/2024, 23:39:13 on My Cluster 5
```

Figure 2.9

clinicaltrial_2020.csv

```
Cmd 8
                                                                                    Python
1   dbutils.fs.head(f"/FileStore/tables/{clinical_tiral_file}.csv")

[Truncated to first 65536 bytes]
"Id|Sponsor|Status|Start|Completion|Type|Submission|Conditions|Interventions\r\nNCT02758028|The University of Hong Kong|Recruiting|Aug 2005|Nov 2021|Intervent
ional|Apr 2016||\r\nNCT02751957|Duke University|Completed|Jul 2016|Jul 2020|Interventional|Apr 2016|Autistic Disorder,Autism Spectrum Disorder|\r\nNCT02758483
|Universidade Federal do Rio de Janeiro|Completed|Mar 2017|Jan 2018|Interventional|Apr 2016|Diabetes Mellitus|\r\nNCT02759848|Istanbul Medeniyet University|Co
mpleted|Jan 2012|Dec 2014|Observational|May 2016|Tuberculosis,Lung Diseases,Pulmonary Disease|\r\nNCT02758860|University of Roma La Sapienza|Active, not recru
iting|Jun 2016|Sep 2020|Observational [Patient Registry]|Apr 2016|Diverticular Diseases,Diverticulum,Diverticulosis|\r\nNCT02757209|Consorzio Futuro in Ricerc
a|Completed|Apr 2016|Jan 2018|Interventional|Apr 2016|Asthma|Fluticasone,Xhance,Budesonide,Formoterol Fumarate,Salmeterol Xinafoate\r\nNCT02752438|Ankara Univ
ersity|Unknown status|May 2016|Jul 2017|Observational [Patient Registry]|Apr 2016|Hypoventilation|\r\nNCT02753543|Ruijin Hospital|Unknown status|Nov 2015|Nov
2019|Interventional|Apr 2016|Lymphoma|\r\nNCT02757508|Washington University School of Medicine|Completed|Mar 2016|Jul 2017|Interventional|Apr 2016||Vitamins\r
\nNCT02753530|Orphazyme|Completed|Aug 2017|Jan 2021|Interventional|Apr 2016|Myositis|\r\nNCT02754817|Novo Nordisk A/S|Completed|Apr 2016|Oct 2016|Observationa
l|Apr 2016|Diabetes Mellitus|Liraglutide,Xultophy\r\nNCT02759276|Daniel Alexandre Bottino|Completed|May 2015|Dec 2015|Observational|Apr 2016|Hypertension|\r\n
NCT02750956|Bulent Ecevit University|Completed|Jun 2015|Mar 2016|Observational|Apr 2016|Periodontal Diseases|\r\nNCT02752113|Institut für Pharmakologie und Pr
äventive Medizin|Completed|Apr 2016|May 2019|Interventional|Apr 2016|Diabetes Mellitus|Metformin,Empagliflozin,Linagliptin\r\nNCT02752698|The Third Xiangya Ho
spital of Central South University|Active, not recruiting|Jan 2015|Dec 2021|Interventional|Jun 2015|Appendicitis,Stomach Ulcer,Cholecystolithiasis,Cholelithia
sis,Gallstones|\r\nNCT02755779|Tel Aviv Medical Center|Unknown status|Jun 2016|Jun 2017|Observational|Apr 2016||\r\nNCT02750384|Medicines for Malaria Venture|
Terminated|May 2016|Jul 2016|Interventional|Apr 2016||\r\nNCT02754609|James Cook University, Queensland, Australia|Completed|Sep 2016|Oct 2019|Interventional|
Apr 2016|Hookworm Infections,Celiac Disease|\r\nNCT02755701|Soonchunhyang University Hospital|Unknown status|Jul 2016|Dec 2018|Interventional|Apr 2016|Ascites
|\r\nNCT02751762|Member Companies of the Opioid PMR Consortium|Recruiting|Nov 2017|Oct 2022|Observational|Apr 2016|Chronic Pain,Substance-Related Disorders,Op
ioid-Related Disorders,Narcotic-Related Disorders,Behavior|\r\nNCT02756299|Marmara University|Completed|Jun 2014|Apr 2015|Interventional|Apr 2016|Sleep Apnea
Syndromes,Sleep Apnea|\r\nNCT02750709|Cycle Pharmaceuticals Ltd.|Completed|Oct 2015|Jan 2016|Interventional|Apr 2016|Tyrosinemias|Nitisinone\r\nNCT02753907|Yo
nsei University|Completed|Jun 2015||Interventional|Apr 2016||\r\nNCT02755467|Cutera Inc.|Completed|May 2016|Apr 2017|Interventional|Apr 2016|Hemangioma|\r\nNC
T02755298|University of Zurich|Completed|Oct 2016|Nov 2020|Interventional|Mar 2016|Hypertension|Acetazolamide\r\nNCT02759614|Guangdong Association of Clinical
Command took 0.19 seconds -- by m.i.olu1@edu.salford.ac.uk at 27/03/2024, 23:45:16 on My Cluster 5
```

Figure 2.10

pharma.csv

```
Cmd 9
                                                                                    Python
1   dbutils.fs.head(f"/FileStore/tables/{pharma_file}.csv")

[Truncated to first 65536 bytes]
'"Company","Parent_Company","Penalty_Amount","Subtraction_From_Penalty","Penalty_Amount_Adjusted_For_Eliminating_Multiple_Counting","Penalty_Year","Penalty_Da
te","Offense_Group","Primary_Offense","Secondary_Offense","Description","Level_of_Government","Action_Type","Agency","Civil/Criminal","Prosecution_Agreemen
t","Court","Case_ID","Private_Litigation_Case_Title","Lawsuit_Resolution","Facility_State","City","Address","Zip","NAICS_Code","NAICS_Translation","HQ_Country
_of_Parent","HQ_State_of_Parent","Ownership_Structure","Parent_Company_Stock_Ticker","Major_Industry_of_Parent","Specific_Industry_of_Parent","Info_Source","N
otes"\n"Abbott Laboratories","Abbott Laboratories","$5,475,000","$0","$5,475,000","2013","20131227","government-contracting-related offenses","False Claims Ac
t and related","kickbacks and bribery","Abbott Laboratories agreed to $5.475 million to resolve allegations that it violated the False Claims Act by paying ki
ckbacks to induce doctors to implant the company\'s carotid, biliary and peripheral vascular products.","federal","agency action","Justice Department Civil Di
vision","civil","","","","","","","","","","","","USA","Illinois","publicly traded","ABT","pharmaceuticals","pharmaceuticals","https://www.justice.gov/opa/pr/
abbott-laboratories-pays-us-5475-million-settle-claims-company-paid-kickbacks-physicians",""\n"Abbott Laboratories Inc.","AbbVie","$1,500,000,000","$0","$1,50
0,000,000","2012","20120507","healthcare-related offenses","off-label or unapproved promotion of medical products","","Global Health Care Company Abbott Labor
atories Inc. has pleaded guilty and agreed to pay $1.5 billion to resolve its criminal and civil liability arising from the company\'s unlawful promotion of t
he prescription drug Depakote for uses not approved as safe and effective by the Food and Drug Administration. The resolution - the second largest payment by
a drug company - includes a criminal fine and forfeiture totaling $700 million and civil settlements with the federal government and the states totaling $800
million. Abbott also will be subject to court-supervised probation and reporting obligations for Abbotts CEO and Board of Directors.","federal","agency actio
n","Food and Drug Administration referral to the Justice Department","civil and criminal","","","","","","","","","","","","USA","Illinois","publicly trade
d","ABBV","pharmaceuticals","pharmaceuticals","http://www.justice.gov/opa/pr/abbott-labs-pay-15-billion-resolve-criminal-civil-investigations-label-promotion-
depakote",""\n"Abbott Laboratories Inc.","AbbVie","$126,500,000","$0","$126,500,000","2010","20101207","government-contracting-related offenses","False Claims
Act and related","","Abbott Laboratories Inc., B. Braun Medical Inc. and Roxane Laboratories Inc. n/k/a Boehringer Ingelheim Roxane Inc. and affiliated entiti
es agreed to pay $421 million to settle False Claims Act allegations, that they engaged in a scheme to report false and inflated prices for numerous pharmaceu
tical products knowing that federal healthcare programs relied on those reported prices to set payment rates.","federal","agency action","Justice Department C
ivil Division","civil","","","","","","","","","","","","USA","Illinois","publicly traded","ABBV","pharmaceuticals","pharmaceuticals","https://www.justice.go
Command took 0.27 seconds -- by m.i.olu1@edu.salford.ac.uk at 27/03/2024, 23:51:12 on My Cluster 5
```

Figure 2.11

From the results in the above images, we can conclude some general information about the nature of all four files which will be summarized in the table below.

Dataset	Delimiter	Number of columns
clinicaltrial_2023.csv	\t	14
clinicaltrial_2021.csv	|	9
clinicaltrial_2020.csv	|	9
pharma.csv	,	34

CREATING AND CLEANING OF RDD, DATAFRAME AND SQL TABLES

After studying our datasets, we can go ahead to create our tables.

i. Creating an RDD

Cmd 10

Python

```python
# Function to create

def create_rdd(_file):
    RDD = sc.textFile(f"/FileStore/tables/{_file}.csv")
    return RDD

CT_RDD_MAIN = create_rdd(clinical_tiral_file) # Calling the function for creating rdd with the clinical trial file as the parameter
PHARMA_RDD_MAIN = create_rdd(pharma_file)  # Calling the function for creating rdd with the pharma file as the parameter

CT_RDD_MAIN.take(10)
```

(1) Spark Jobs

```
['"Id\tStudy Title\tAcronym\tStatus\tConditions\tInterventions\tSponsor\tCollaborators\tEnrollment\tFunder Type\tType\tStudy Design\tStart\tCompletio
n
",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
 '"NCT03630471\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 Abuse)\tBEHAVIORAL: PRIDE \'Step 1\' problem-solving intervention|BEHAVIORAL: Enhanced usual ca
re\tSangath\tHarvard Medical School (HMS and HSDM)|London School of Hygiene and Tropical Medicine\t250.0\tOTHER\tINTERVENTIONAL\tAllocation: RANDOMIZED|Interv
ention Model: PARALLEL|Masking: DOUBLE (INVESTIGATOR"," OUTCOMES_ASSESSOR)|Primary Purpose: TREATMENT\t2018-08-20\t2019-02-2
8
",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
 '"NCT05992571\tOral Ketone Monoester Supplementation and Resting-state Brain Connectivity\t\tRECRUITING\tCerebrovascular Function|Cognition\tOTHER: Placebo|D
IETARY_SUPPLEMENT: β-OHB\tMcMaster University\tAlzheimer\'s Society of Brant", Haldimand Norfolk," Hamilton Halton\t30.0\tOTHER\tINTERVENTIONAL\tAllocation: R
ANDOMIZED|Intervention Model: CROSSOVER|Masking: TRIPLE (PARTICIPANT", INVESTIGATOR," OUTCOMES_ASSESSOR)|Primary Purpose: BASIC_SCIENCE\t2023-10-25\t2024-0
8
",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
 '"NCT00237471\tImpact of Tight Glycaemic Control in Acute Myocardial Infarction\t\tTERMINATED\tMyocardial Infarct|Hyperglycemia\tDRUG: Insulin (tight blood g
lucose control)\tMelbourne Health\tNational Health and Medical Research Council"," Australia|Bristol-Myers Squibb\t40.0\tOTHER\tINTERVENTIONAL\tAllocation: RA
NDOMIZED|Intervention Model: PARALLEL|Masking: NONE|Primary Purpose: TREATMENT\t2005-10\t2006-0
5
",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
 '"NCT03820271\tNew Prognostic Predictive Models of Mortality of Decompensated Cirrhotic Patients Waiting for Liver Transplantation\tSUPERMELD\tRECRUITING\tDe
compensated Cirrhosis|Liver Transplantation\tOTHER: SuperMELD\tAssistance Publique - Hôpitaux de Paris\t\t500.0\tOTHER\tINTERVENTIONAL\tAllocation: NA|Interve
```

Command took 1.62 seconds -- by m.i.olu1@edu.salford.ac.uk at 28/03/2024, 00:24:08 on My Cluster 5

Figure 2.12

The image shows an initial RDD created using the textFile module in the SC library, using a function called create_rdd() that constructs a file path using string formatting and takes an argument as the file name.

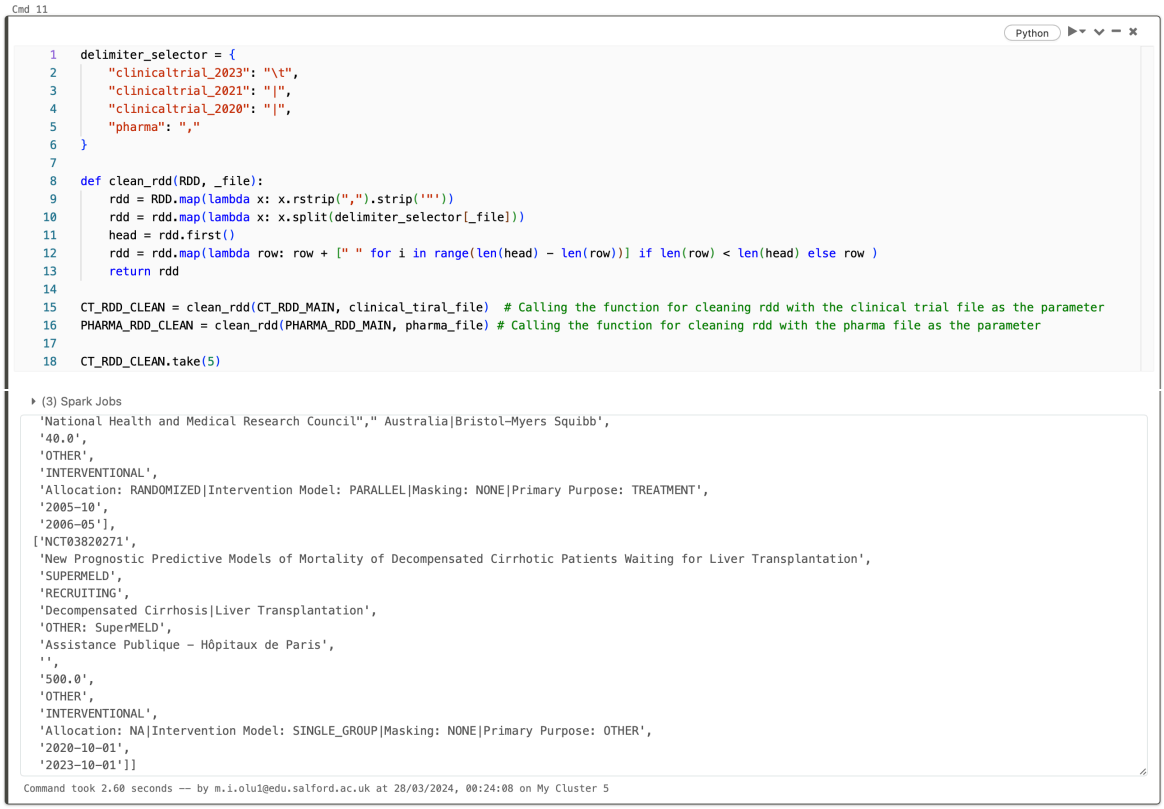


Figure 2.13

In figure 2.13, We created a function called clean_rdd, which removes unnecessary commas and quotation marks from the initial RDD and its file name, and then splits it using its delimiter.

Justification: This is to enable efficient processing of dataset and also for more readability.

The delimiter of each file is automatically extracted from the **delimiter_selector** dictionary using its file name as the key to extract the associated value in the dictionary.

Justification: The reason for this is to allow for code reusability for all versions of file where each file contains different types of delimiter.

The next line of code (line 12) filled in empty strings for rows where the number of columns are incomplete or are not equal to the first row in the dataset which is the header.

Justification: This is to avoid index errors when performing analysis on specific columns on the dataset.

ii. Creating a Dataframe

Cmd 2

```python
# CREATING DATAFRAME
from pyspark.sql.types import *
from pyspark.sql.functions import *

delimiter = {
    "clinicaltrial_2023": "\t",
    "clinicaltrial_2021": "|",
    "clinicaltrial_2020": "|",
    "pharma": ","
}

# Function for creating the dataframe using using different methods for different datasets
def create_dataframe(_file):
    if _file == "clinicaltrial_2023":
        rdd = sc.textFile(f"/FileStore/tables/{_file}.csv").map(lambda x: x.rstrip(",").strip('"')).map(lambda row: row.split(delimiter[_file]))
        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/{_file}.csv", sep=delimiter[_file], header = True)
```

Command took 0.16 seconds -- by m.i.olu1@edu.salford.ac.uk at 27/03/2024, 10:02:50 on hgdgwy

The dataframe was created using the rdd.toDF() method, converting a clean RDD into a dataframe. A loop was used to manually rename the schema before removing the first row containing the header.

Figure 2.15 shows the result of the 2023 dataframe. The value of the variable **clinical_trial_file** can also be changed to 2020 or 2021 dataset to view the dataframe of each dataset.

Cmd 3

```python
# Creating the clinical trial dataframe using the function from previous cell

ct_dataframe = create_dataframe(clinical_tiral_file)
ct_dataframe.show(20)
```

(4) Spark Jobs

ct_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	INTERVENTIONAL	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	INTERVENTIONAL	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	INTERVENTIONAL	Allocation: RANDO...	2005-10	2006-05
NCT03820271	New Prognostic Pr...	SUPERMELD	RECRUITING	Decompensated Cir...	OTHER: SuperMELD	Assistance Publiq...		500.0	OTHER	INTERVENTIONAL	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	INTERVENTIONAL	Allocation: RANDO...	2024-10-01	2026-04-01
NCT02945371	Tailored Inhibito...	REV	COMPLETED	Smoking|Alcohol D...	BEHAVIORAL: Perso...	University of Oregon		103.0	OTHER	INTERVENTIONAL	Allocation: RANDO...	2014-09	2016-05
NCT01055171	Neuromodulation o...		COMPLETED	Alcohol Dependenc...	DRUG: Propranolol...	Medical Universit...	National Institut...	44.0	OTHER	INTERVENTIONAL	Allocation: RANDO...	2010-01	2012-08
NCT01125371	Computerized Brie...		COMPLETED	Alcohol; Harmful ...	BEHAVIORAL: Compu...	Johns Hopkins Uni...	National Institut...	439.0					

Command took 22.30 seconds -- by m.i.olu1@edu.salford.ac.uk at 27/03/2024, 10:02:50 on hgdgwy

Figure 2.15

iii. SQL TABLE CREATION

Cmd 4

```
%python
# Creating an information schema for the tables

i_schema = StructType([StructField("info_type", StringType()), StructField("value", StringType())])
conditions_delimiter = { "clinicaltrial_2023": "\|", "clinicaltrial_2021": ",", "clinicaltrial_2020": ","}

date_delimiter = {"clinicaltrial_2023": "-", "clinicaltrial_2021": " ", "clinicaltrial_2020": " "}

rdd = sc.parallelize([("filename", clinical_trial_file), ("conditions_delimiter", conditions_delimiter[clinical_trial_file]), ("date_delimiter",
date_delimiter[clinical_trial_file]), ("year", clinical_trial_file[-4:])])
infos = spark.createDataFrame(rdd, schema=i_schema)


infos.show()
```

(3) Spark Jobs

infos: pyspark.sql.dataframe.DataFrame = [info_type: string, value: string]

```
+--------------------+--------------------+
|           info_type|               value|
+--------------------+--------------------+
|            filename|clinicaltrial_2023|
|conditions_delimiter|                 \||
|      date_delimiter|                  -|
|                year|                2023|
+--------------------+--------------------+
```

Command took 0.70 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/05/2024, 21:34:35 on cluster

I created an info table with values which will be useful for some analysis in some questions.

Cmd 5

```
%python
ct_dataframe.createOrReplaceTempView('clinical_trial_table')
pharma_dataframe.createOrReplaceTempView('pharma')
infos.createOrReplaceTempView('infos')
```

Command took 0.09 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/05/2024, 18:46:28 on cluster

Cmd 6

```
CREATE DATABASE IF NOT EXISTS bdtt_db;
CREATE OR REPLACE TABLE bdtt_db.infos AS SELECT * FROM infos;
```

(6) Spark Jobs

Query returned no results

Command took 5.27 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/05/2024, 18:46:28 on cluster

Cmd 7

```
SELECT * FROM clinical_trial_table
```

(1) Spark Jobs

Table

	Id	Study Title	Acronym	Status
1	NCT03630471	Effectiveness of a Problem-solving Intervention for Common Adolescent Mental Health Problems in India	PRIDE	COMPLETED
2	NCT05992571	Oral Ketone Monoester Supplementation and Resting-state Brain Connectivity		RECRUITING
3	NCT00237471	Impact of Tight Glycaemic Control in Acute Myocardial Infarction		TERMINATED
4	NCT03820271	New Prognostic Predictive Models of Mortality of Decompensated Cirrhotic Patients Waiting for Liver Transplantation	SUPERMELD	RECRUITING
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

4,862 rows | Truncated data | 1.14 seconds runtime — Refreshed 6 minutes ago

Command took 1.14 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/05/2024, 19:05:27 on cluster

Figure 2.16

The SQL table was created from a dataframe using the `createOrReplaceTempView` function, displaying the 2023 dataset, which can be switched between 2020 and 2021.

3. PROBLEM ANSWERS

QUESTION 1: NUMBER OF STUDIES IN THE DATASET

ASSUMPTIONS:

I assumed that each row in the dataset represents each study carried out for each year. Therefore, counting the number of rows gives us the number of studies in the dataset.

QUESTION 1 RDD SOLUTIONS

clinicaltrial_2023 Question 1 (RDD)

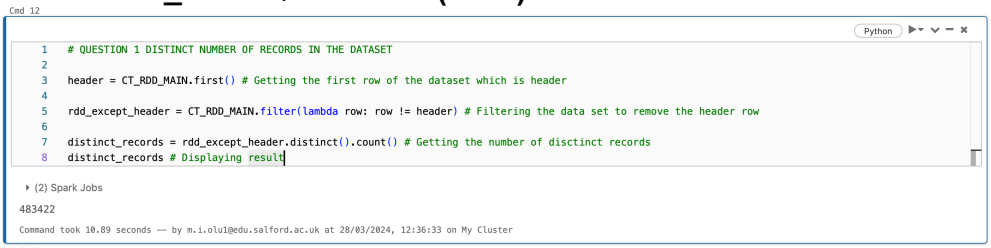


Figure 3.1

clinicaltrial_2021 Question 1 (RDD)

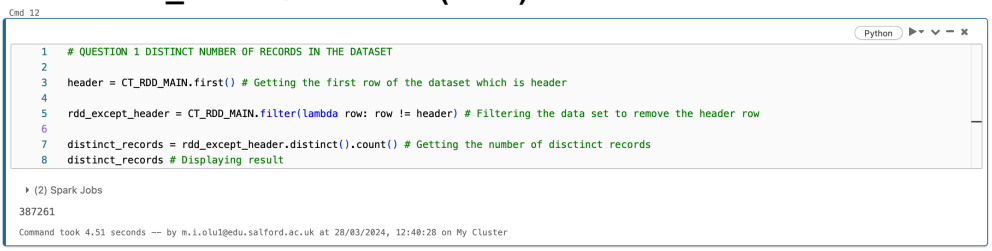


Figure 3.2

clinicaltrial_2020 Question 1 (RDD)

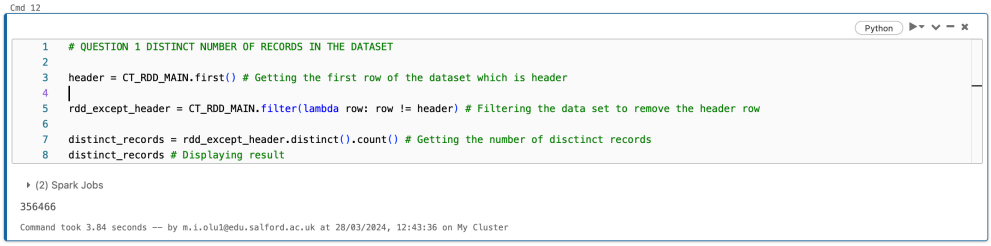


Figure 3.3

Description of main ideas for question 1 (RDD)

In order to get the actual number of unique studies in the dataset, we must use the **distinct()** function on our rdd dataset followed by the **count()** function to count the distinct dataset. The header was filtered out because the header isn't a part of the records.

QUESTION 1 DATAFRAME SOLUTIONS

clinicaltrial_2023 Question 1 (Dataframe)

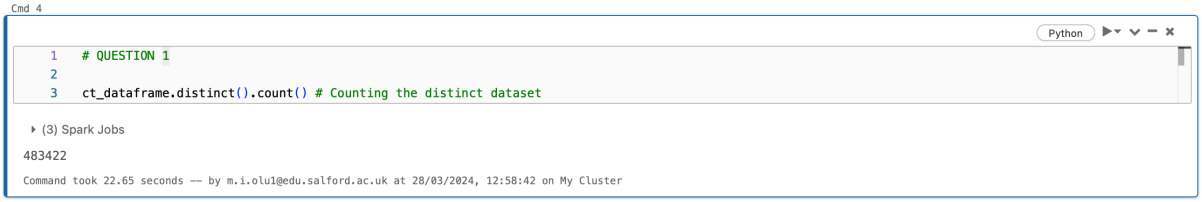


Figure 3.4

clinicaltrial_2021 Question 1 (Dataframe)

Figure 3.5

clinicaltrial_2020 Question 1 (Dataframe)

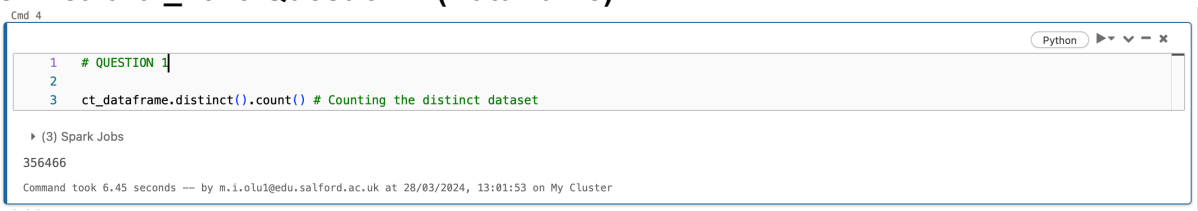


Figure 3.6

Description of main ideas for question 1 (Dataframe)

In order to get the actual number of unique studies in the dataset, we must use the **distinct()** function on our dataframe dataset followed by the **count()** function to count the distinct dataset.

QUESTION 1 SQL SOLUTIONS

clinicaltrial_2023 Question 1 (SQL)

Figure 3.7

clinicaltrial_2021 Question 1 (SQL)

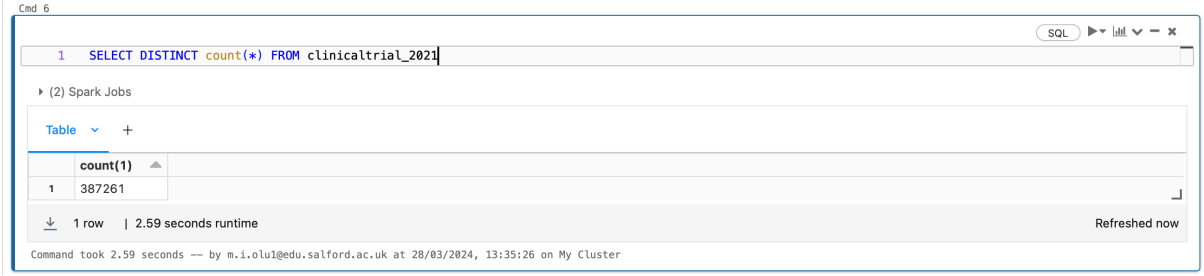


Figure 3.8

clinicaltrial_2020 Question 1 (SQL)

Figure 3.9

Description of main ideas for question 1 (SQL)

The "SELECT DISTINCT" command in SQL extracts a dataset with unique records. The count() function, with the "*" parameter, counts all records extracted by the "SELECT DISTINCT" command, providing the distinct number of studies in the dataset.

Discussion of results for question 1

From the results, we were able to know how many unique clinical trials which was carried out in the years 2023, 2021 and 2020. Clinical trial 2023 had the highest number of studies with **483,422** unique studies, followed by 2021 with **387,261** and then 2020 with **356,466** studies.

QUESTION 2: LIST ALL THE TYPES (AS CONTAINED IN THE TYPE COLUMN) OF STUDIES IN THE DATASET ALONG WITH THE FREQUENCIES OF EACH TYPE.

ASSUMPTIONS:

In this analysis, I assumed that all the “Types” has same index as the header. Or all types are under the Type Column.

QUESTION 2 RDD SOLUTIONS

clinicaltrial_2023 Question 2 (RDD)

Cmd 13

```
# QUESTION TWO: ALL TYPES OF TRIALS IN THE TYPE COLUMN

typ_inx = CT_RDD_CLEAN.first().index('Type') # Finding the index of the "Type" Column

typeColumn = CT_RDD_CLEAN.map(lambda x: (x[typ_inx], 1)) # Creating a new rdd from the "Type" column using its index

typeColumn = typeColumn.filter(lambda row: row[0] not in ("Type", " ", "")) # Removing the column header and empty columns

typeColumnFrequency = typeColumn.reduceByKey(lambda a,b: a + b) # Reducing by key to get frequency of each Type

typeColumnFrequencySorted = typeColumnFrequency.sortBy(lambda x: x[1], ascending=False) # Sorting accoringing to frequency of each type (x[1])

typeColumnFrequencySorted.collect() # Viewing results
```

(4) Spark Jobs

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

Command took 13.67 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/04/2024, 18:15:05 on My Cluster 5

Figure 3.10

clinicaltrial_2021 Question 2 (RDD)

Cmd 13

```
# QUESTION TWO: ALL TYPES OF TRIALS IN THE TYPE COLUMN

typ_inx = CT_RDD_CLEAN.first().index('Type') # Finding the index of the "Type" Column

typeColumn = CT_RDD_CLEAN.map(lambda x: (x[typ_inx], 1)) # Creating a new rdd from the "Type" column using its index

typeColumn = typeColumn.filter(lambda row: row[0] not in ("Type", " ", "")) # Removing the column header and empty columns

typeColumnFrequency = typeColumn.reduceByKey(lambda a,b: a + b) # Reducing by key to get frequency of each Type

typeColumnFrequencySorted = typeColumnFrequency.sortBy(lambda x: x[1], ascending=False) # Sorting accoringing to frequency of each type (x[1])

typeColumnFrequencySorted.collect() # Viewing results
```

(4) Spark Jobs

```
[('Interventional', 301472),
 ('Observational', 77540),
 ('Observational [Patient Registry]', 8180),
 ('Expanded Access', 69)]
```

Command took 3.92 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/04/2024, 18:48:52 on My Cluster 5

Figure 3.11

clinicaltrial_2020 Question 2 (RDD)

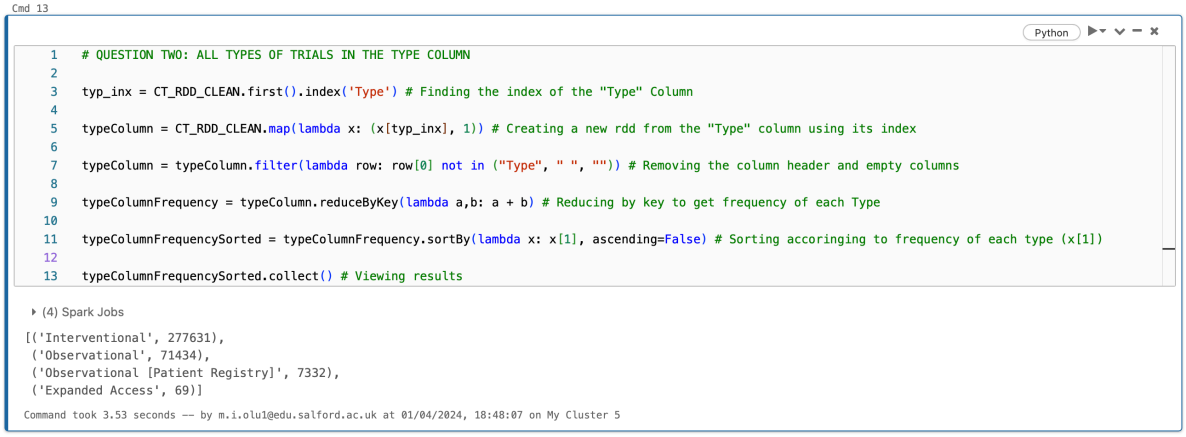


Figure 3.12

Description of main ideas for question 2 (RDD)

The code extracts the "Type" column index from the first row of the dataframe, creates a key-value pair, filters out empty rows, and performs a reducing transformation on the rdd to find the sum of each distinct key, resulting in the final answer.

QUESTION 2 DATAFRAME SOLUTIONS

clinicaltrial_2023 Question 2 (Dataframe)

Figure 3.13

clinicaltrial_2021 Question 2 (Dataframe)

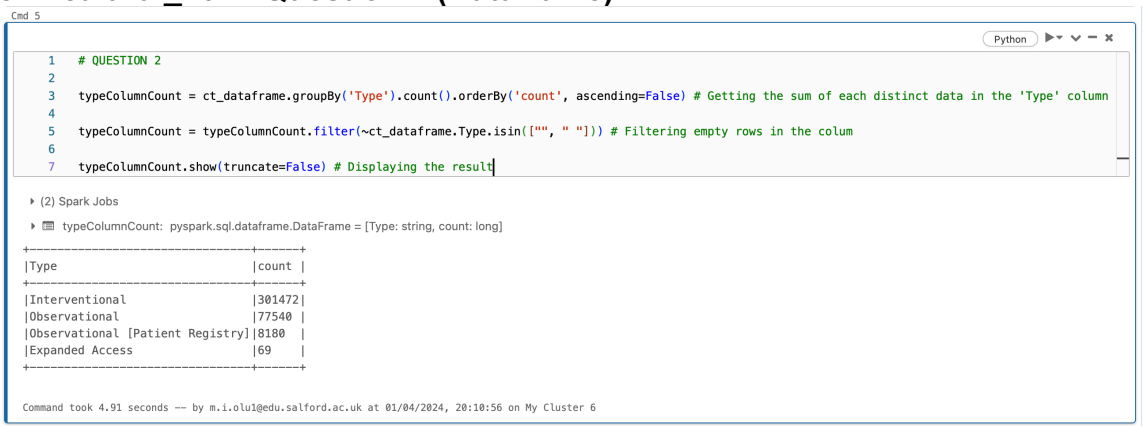


Figure 3.14

clinicaltrial_2020 Question 2 (Dataframe)

Figure 3.15

Description of main ideas for question 2 (Dataframe)

The first step involved summarizing distinct values in the Type column using groupBy(), count(), orderBy() functions, similar to a map-reducing transformation in an rdd. A filtering function was used to remove empty rows, and results were displayed in line 7.

QUESTION 2 SQL SOLUTIONS

clinicaltrial_2023 Question 2 (SQL)

Figure 3.16

clinicaltrial_2021 Question 2 (SQL)

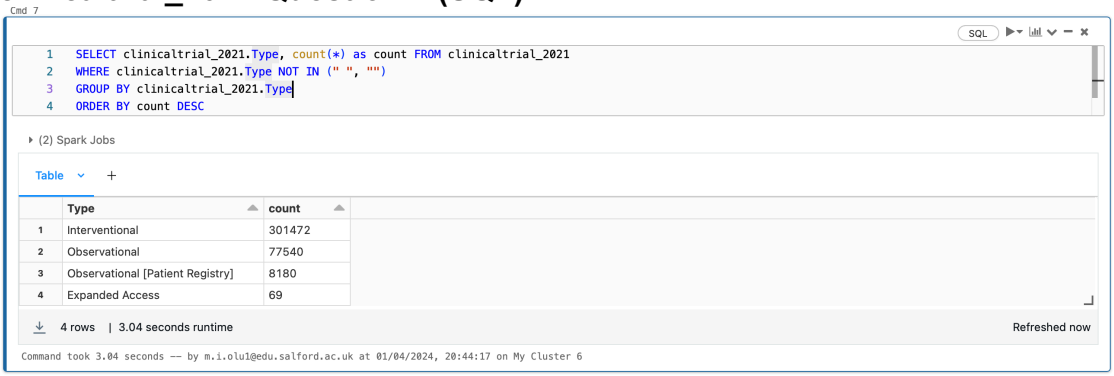


Figure 3.17

clinicaltrial_2020 Question 2 (SQL)

Figure 3.18

Description of main ideas for question 2 (SQL)

The type column was selected and counted using the count function, which requires a group command. The query was filtered to remove empty rows and ordered in descending order.

Discussion of results for question 2

The interventional study is the most common type of study across all three years of clinical trials, with 2023 having 371,382, 2021 at 301,472, and 2021 at 277,631, followed by the observational study, which is the second most common.

Dataset	Interventional	Observational	Observational [Patient Registry]	Expanded Access
Clinical trials 2023	371,382	110,221	0	928
Clinical trials 2021	301,472	77,540	8,180	69
Clinical trials 2020	277,631	71,434	7,332	69

QUESTION 3: THE TOP 5 CONDITIONS (FROM CONDITIONS) WITH THEIR FREQUENCIES.

ASSUMPTIONS:

In this analysis, I assumed that all the “Conditions” has same index as the header. Furthermore, some rows contain multiple conditions split by a delimiter.

QUESTION 3 RDD SOLUTIONS

clinicaltrial_2023 Question 3 (RDD)

```
Python
# QUESTION 3: ALL CONDITIONS

conditions_delimeter = {       # Creating a dictionary for delimiter mapping using the file names for each year
    "clinicaltrial_2023": "|",
    "clinicaltrial_2021": ",",
    "clinicaltrial_2020": ",",
}
con_col_indx = CT_RDD_CLEAN.first().index('Conditions') #Getting the index of the Conditions column

flatMapConditionsCol = CT_RDD_CLEAN.flatMap(lambda x: x[con_col_indx].split(conditions_delimeter[clinical_tiral_file])) # Flatmaping to destructure

filteredConditionsCol = flatMapConditionsCol.filter(lambda row: row not in ('Conditions', "", " ")) # Filtering to remove header and empty rows

valueKeyPairConditionsCol = filteredConditionsCol.map(lambda x: (x, 1)) # Creating a key-value pair tuple for each item in the column

valueSumsConditionCol = valueKeyPairConditionsCol.reduceByKey(lambda a,b: a + b) # Getting the individual sum of distinct data by reducing by key

sortedConditionsCol = valueSumsConditionCol.sortBy(lambda x: x[1], ascending=False) #Sorting our rdd by the value of the sums

sortedConditionsCol.take(5) # Viewing results

▸ (4) Spark Jobs
[('Healthy', 9731),
 ('Breast Cancer', 7502),
 ('Obesity', 6549),
 ('Stroke', 4071),
 ('Hypertension', 4020)]
Command took 11.16 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/04/2024, 20:58:50 on My Cluster 6
```

Figure 3.19

clinicaltrial_2021 Question 3 (RDD)

```
Python
# QUESTION 3: ALL CONDITIONS

conditions_delimeter = {       # Creating a dictionary for delimiter mapping using the file names for each year
    "clinicaltrial_2023": "|",
    "clinicaltrial_2021": ",",
    "clinicaltrial_2020": ",",
}
con_col_indx = CT_RDD_CLEAN.first().index('Conditions') #Getting the index of the Conditions column

flatMapConditionsCol = CT_RDD_CLEAN.flatMap(lambda x: x[con_col_indx].split(conditions_delimeter[clinical_tiral_file])) # Flatmaping to destructure

filteredConditionsCol = flatMapConditionsCol.filter(lambda row: row not in ('Conditions', "", " ")) # Filtering to remove header and empty rows

valueKeyPairConditionsCol = filteredConditionsCol.map(lambda x: (x, 1)) # Creating a key-value pair tuple for each item in the column

valueSumsConditionCol = valueKeyPairConditionsCol.reduceByKey(lambda a,b: a + b) # Getting the individual sum of distinct data by reducing by key

sortedConditionsCol = valueSumsConditionCol.sortBy(lambda x: x[1], ascending=False) #Sorting our rdd by the value of the sums

sortedConditionsCol.take(5) # Viewing results

▸ (4) Spark Jobs
[('Carcinoma', 13389),
 ('Diabetes Mellitus', 11080),
 ('Neoplasms', 9371),
 ('Breast Neoplasms', 8640),
 ('Syndrome', 8032)]
Command took 4.00 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/04/2024, 21:07:50 on My Cluster 6
```

Figure 3.20

clinicaltrial_2020 Question 3 (RDD)

Cmd 14

```python
# QUESTION 3: ALL CONDITIONS

conditions_delimeter = {       # Creating a dictionary for delimiter mapping using the file names for each year
    "clinicaltrial_2023": "|",
    "clinicaltrial_2021": ",",
    "clinicaltrial_2020": ",",
}
con_col_indx = CT_RDD_CLEAN.first().index('Conditions') #Getting the index of the Conditions column

flatMapConditionsCol = CT_RDD_CLEAN.flatMap(lambda x: x[con_col_indx].split(conditions_delimeter[clinical_tiral_file])) # Flatmaping to destructure

filteredConditionsCol = flatMapConditionsCol.filter(lambda row: row not in ('Conditions', "", " ")) # Filtering to remove header and empty rows

valueKeyPairConditionsCol = filteredConditionsCol.map(lambda x: (x, 1)) # Creating a key-value pair tuple for each item in the column

valueSumsConditionCol = valueKeyPairConditionsCol.reduceByKey(lambda a,b: a + b) # Getting the individual sum of distinct data by reducing by key

sortedConditionsCol = valueSumsConditionCol.sortBy(lambda x: x[1], ascending=False) #Sorting our rdd by the value of the sums

sortedConditionsCol.take(5) # Viewing results
```

▸ (4) Spark Jobs

```
[('Carcinoma', 12245),
 ('Diabetes Mellitus', 10425),
 ('Neoplasms', 8534),
 ('Breast Neoplasms', 8009),
 ('Syndrome', 7419)]
```

Command took 3.50 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/04/2024, 21:13:21 on My Cluster 6

Figure 3.21

Description of main ideas for question 3 (RDD)

Some conditions column contains multiple conditions separated by a delimiter. A flatmap transformation was used to split the column, selecting the delimiter based on the dataset. A key-value pair was created, followed by a reducing transformation for the solution.

QUESTION 3 DATAFRAME SOLUTIONS

clinicaltrial_2023 Question 3 (Dataframe)

```
# QUESTION 3
conditions_delimeter = {
    "clinicaltrial_2023": "\|",
    "clinicaltrial_2021": ",",
    "clinicaltrial_2020": ",",
}

# Spliting the rows in the Conditions column using the delimeter from "conditions_delimeter" dictionary and exploding to destructure grouped data
into individual rows.
explodedConditionsColumn = ct_dataframe.withColumn('Conditions', explode(split('Conditions', conditions_delimeter[clinical_tiral_file])))

# Getting the sum of each distinct data in the exploded Conditions column.
ConditionsColumnCount = explodedConditionsColumn.groupBy('Conditions').count().orderBy('count', ascending=False)

# Filtering to remove empty rows
ConditionsColumnCount = ConditionsColumnCount.filter("Conditions != ''")

# Displaying the results
ConditionsColumnCount.show(5, truncate=False)
```

```
(2) Spark Jobs
explodedConditionsColumn:  pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 12 more fields]
ConditionsColumnCount:  pyspark.sql.dataframe.DataFrame = [Conditions: string, count: long]
+-------------+-----+
|Conditions   |count|
+-------------+-----+
|Healthy      |9731 |
|Breast Cancer|7502 |
|Obesity      |6549 |
|Stroke       |4071 |
|Hypertension |4020 |
+-------------+-----+
only showing top 5 rows

Command took 13.53 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/04/2024, 22:26:27 on My Cluster 6
```

Figure 3.22

clinicaltrial_2021 Question 3 (Dataframe)

```
# QUESTION 3
conditions_delimeter = {
    "clinicaltrial_2023": "\|",
    "clinicaltrial_2021": ",",
    "clinicaltrial_2020": ",",
}

# Spliting the rows in the Conditions column using the delimeter from "conditions_delimeter" dictionary and exploding to destructure grouped data
into individual rows.
explodedConditionsColumn = ct_dataframe.withColumn('Conditions', explode(split('Conditions', conditions_delimeter[clinical_tiral_file])))

# Getting the sum of each distinct data in the exploded Conditions column.
ConditionsColumnCount = explodedConditionsColumn.groupBy('Conditions').count().orderBy('count', ascending=False)

# Filtering to remove empty rows
ConditionsColumnCount = ConditionsColumnCount.filter("Conditions != ''")

# Displaying the results
ConditionsColumnCount.show(5, truncate=False)
```

```
(2) Spark Jobs
explodedConditionsColumn:  pyspark.sql.dataframe.DataFrame = [Id: string, Sponsor: string ... 7 more fields]
ConditionsColumnCount:  pyspark.sql.dataframe.DataFrame = [Conditions: string, count: long]
+------------------+-----+
|Conditions        |count|
+------------------+-----+
|Carcinoma         |13389|
|Diabetes Mellitus|11080|
|Neoplasms         |9371 |
|Breast Neoplasms  |8640 |
|Syndrome          |8032 |
+------------------+-----+
only showing top 5 rows

Command took 5.97 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 12:44:41 on My Cluster 7
```

Figure 3.23

clinicaltrial_2020 Question 3 (Dataframe)

```python
# QUESTION 3
conditions_delimeter = {
    "clinicaltrial_2023": "\|",
    "clinicaltrial_2021": ",",
    "clinicaltrial_2020": ",",
}

# Spliting the rows in the Conditions column using the delimeter from "conditions_delimeter" dictionary and exploding to destructure grouped data into individual rows.
explodedConditionsColumn = ct_dataframe.withColumn('Conditions', explode(split('Conditions', conditions_delimeter[clinical_tiral_file])))

# Getting the sum of each distinct data in the exploded Conditions column.
ConditionsColumnCount = explodedConditionsColumn.groupBy('Conditions').count().orderBy('count', ascending=False)

# Filtering to remove empty rows
ConditionsColumnCount = ConditionsColumnCount.filter("Conditions != ''")

# Displaying the results
ConditionsColumnCount.show(5, truncate=False)
```

```
(2) Spark Jobs
explodedConditionsColumn:  pyspark.sql.dataframe.DataFrame = [Id: string, Sponsor: string ... 7 more fields]
ConditionsColumnCount:  pyspark.sql.dataframe.DataFrame = [Conditions: string, count: long]
+----------------+-----+
|Conditions      |count|
+----------------+-----+
|Carcinoma       |12245|
|Diabetes Mellitus|10425|
|Neoplasms       |8534 |
|Breast Neoplasms|8009 |
|Syndrome        |7419 |
+----------------+-----+
only showing top 5 rows

Command took 3.46 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/04/2024, 20:15:26 on My Cluster 6
```

Figure 3.24

Description of main ideas for question 3 (Dataframe)

The conditions column was split using a delimiter selector dictionary, resulting in new rows, counted, grouped, and ordered by sum. The dataframe was filtered to remove empty rows, and the top 5 results were displayed.

QUESTION 3 SQL SOLUTIONS

clinicaltrial_2023 Question 3 (SQL)

```
Cmd 14
```

```sql
-- The top 5 conditions (from Conditions) with their frequencies.
CREATE OR REPLACE FUNCTION conditions_delimiter()
  RETURNS STRING
  RETURN SELECT FIRST(value) FROM bdtt_db.infos WHERE info_type == 'conditions_delimiter';

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

```
(5) Spark Jobs
```

Table

	conditions	count
1	Healthy	9731
2	Breast Cancer	7502
3	Obesity	6549
4	Stroke	4071
5	Hypertension	4020

5 rows | 14.60 seconds runtime

Refreshed 4 minutes ago

```
Command took 14.60 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/05/2024, 21:34:35 on cluster
```

Figure 3.25

clinicaltrial_2021 Question 3 (SQL)

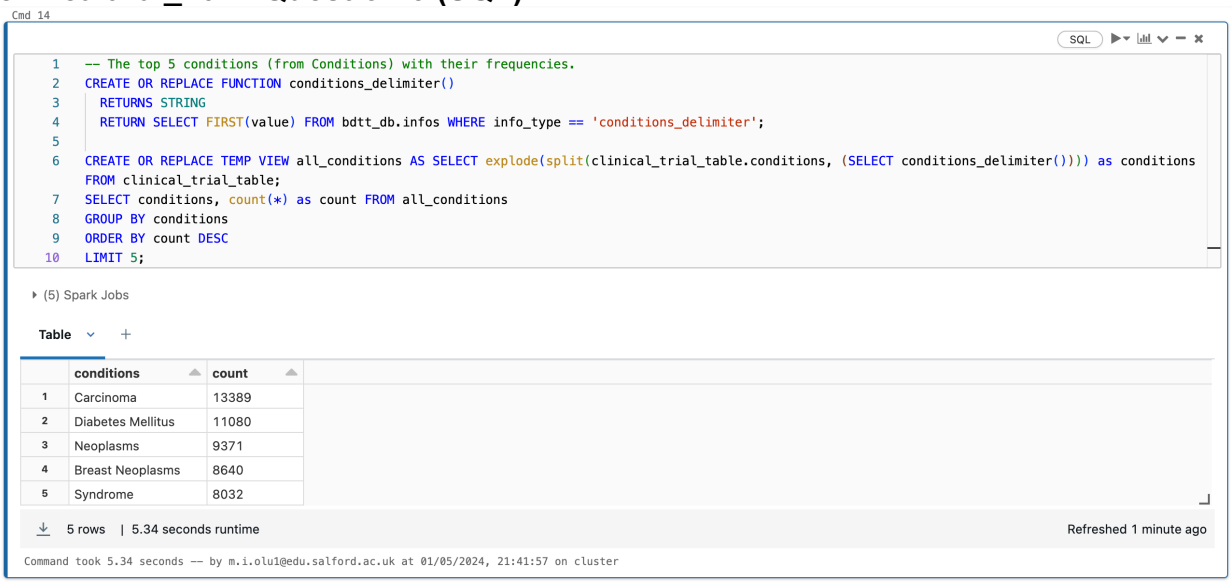


Figure 3.25

clinicaltrial_2020 Question 3 (SQL)

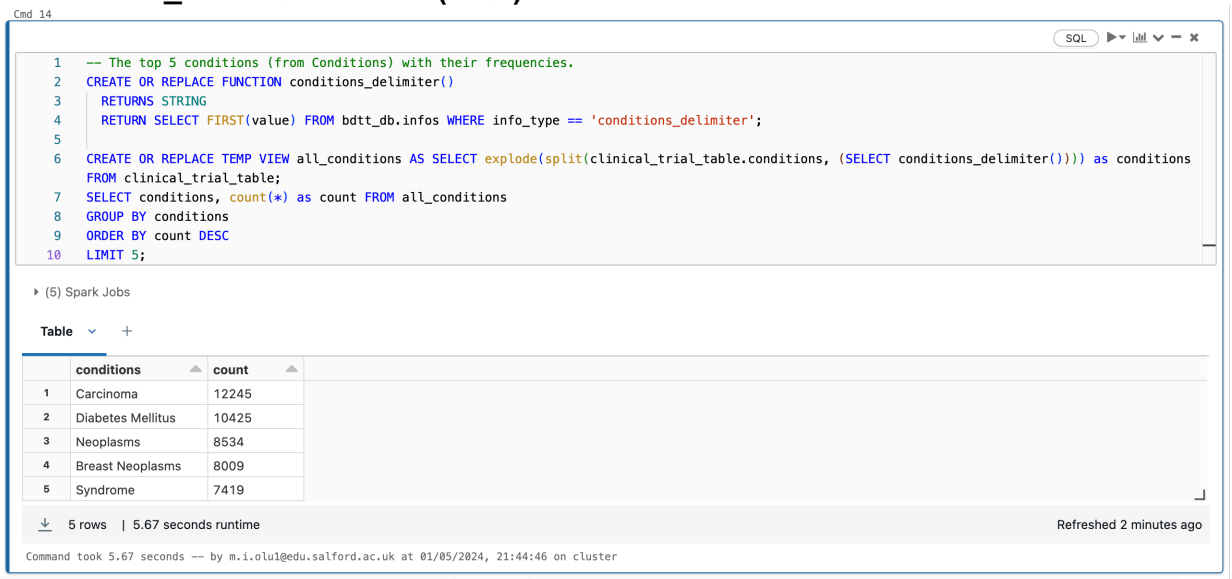


Figure 3.25

Description of main ideas for question 3 (SQL)

A new temporary table was created from and exploded conditions column using a dynamically extracted delimiter from the conditions_delimiter udf, the resulting column was then counted, grouped by conditions column to get the sum of each condition, ordered by the count in descending order and finally limited by 5 to show the top 5 results.

Discussion of results for question 3

The results from clinical trials of 2023 shows the most common condition for the year as **Healthy** with a frequency of **9,731**, followed by **breast cancer** with **7,502** studies, followed by **Obesity** with **6,549**, followed by **Stroke** with **4,701** cases and finally **hypertension** as the least common with **4,020** studies.

QUESTION 4: FIND THE 10 MOST COMMON SPONSORS THAT ARE NOT PHARMACEUTICAL COMPANIES, ALONG WITH THE NUMBER OF CLINICAL TRIALS THEY HAVE SPONSORED.

ASSUMPTIONS:

In this analysis, I assumed that the Parent Company column inside the pharma.csv dataset contains all possible pharmaceutical companies in question.

QUESTION 4 RDD SOLUTIONS

clinicaltrial_2023 Question 4 (RDD)

Cmd 15

```python
# QUESTION 4,  TOP 10 NON - PHARMA COMPANIES
ct_sponsor_col_index = CT_RDD_CLEAN.first().index('Sponsor') # Finding the index of the Sponsor column

parent_pharm_comp = PHARMA_RDD_CLEAN.map(lambda x: x[1].replace('"', '')) # Extracting the parent pharma company column and also cleaning the data

sponsorColumn = CT_RDD_CLEAN.map(lambda x: x[ct_sponsor_col_index]) # Extracting the Sponsor column from the main RDD

sponsorColumn = sponsorColumn.filter(lambda row: row != 'Sponsor') # Removing the Header row from the RDD

sponsorColumnFiltered = sponsorColumn.subtract(parent_pharm_comp) # Finding the difference between the parent pharma company column and Sponsor column

keyValuePairSponsorColumn = sponsorColumnFiltered.map(lambda x: (x, 1)) # Creating a key value pair for the reducing function

valueSumSponsorColumn = keyValuePairSponsorColumn.reduceByKey(lambda x, y: x + y) # Reducing by Key to get sum of distinct data

sortedSponsorColumn = valueSumSponsorColumn.sortBy(lambda x: x[1], ascending=False) # Sorting rdd by sum in decending order

sortedSponsorColumn.take(10) # Veiwing top 10 results
```

▸ (4) Spark Jobs

```
[('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 13.90 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 13:18:14 on My Cluster 7

Figure 3.26

clinicaltrial_2021 Question 4 (RDD)

Cmd 15

```python
# QUESTION 4,  TOP 10 NON - PHARMA COMPANIES
ct_sponsor_col_index = CT_RDD_CLEAN.first().index('Sponsor') # Finding the index of the Sponsor column

parent_pharm_comp = PHARMA_RDD_CLEAN.map(lambda x: x[1].replace('"', '')) # Extracting the parent pharma company column and also cleaning the data

sponsorColumn = CT_RDD_CLEAN.map(lambda x: x[ct_sponsor_col_index]) # Extracting the Sponsor column from the main RDD

sponsorColumn = sponsorColumn.filter(lambda row: row != 'Sponsor') # Removing the Header row from the RDD

sponsorColumnFiltered = sponsorColumn.subtract(parent_pharm_comp) # Finding the difference between the parent pharma company column and Sponsor column

keyValuePairSponsorColumn = sponsorColumnFiltered.map(lambda x: (x, 1)) # Creating a key value pair for the reducing function

valueSumSponsorColumn = keyValuePairSponsorColumn.reduceByKey(lambda x, y: x + y) # Reducing by Key to get sum of distinct data

sortedSponsorColumn = valueSumSponsorColumn.sortBy(lambda x: x[1], ascending=False) # Sorting rdd by sum in decending order

sortedSponsorColumn.take(10) # Veiwing top 10 results
```

▸ (4) Spark Jobs

```
[('National Cancer Institute (NCI)', 3218),
 ('M.D. Anderson Cancer Center', 2414),
 ('Assistance Publique - Hôpitaux de Paris', 2369),
 ('Mayo Clinic', 2300),
 ('Merck Sharp & Dohme Corp.', 2243),
 ('Assiut University', 2154),
 ('Novartis Pharmaceuticals', 2088),
 ('Massachusetts General Hospital', 1971),
 ('Cairo University', 1928),
 ('Hoffmann-La Roche', 1828)]
```

Command took 6.82 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 13:17:29 on My Cluster 7

Figure 3.27

clinicaltrial_2020 Question 4 (RDD)

Cmd 15

```python
# QUESTION 4,  TOP 10 NON - PHARMA COMPANIES
ct_sponsor_col_index = CT_RDD_CLEAN.first().index('Sponsor') # Finding the index of the Sponsor column

parent_pharm_comp = PHARMA_RDD_CLEAN.map(lambda x: x[1].replace('"', '')) # Extracting the parent pharma company column and also cleaning the data

sponsorColumn = CT_RDD_CLEAN.map(lambda x: x[ct_sponsor_col_index]) # Extracting the Sponsor column from the main RDD

sponsorColumn = sponsorColumn.filter(lambda row: row != 'Sponsor') # Removing the Header row from the RDD

sponsorColumnFiltered = sponsorColumn.subtract(parent_pharm_comp) # Finding the difference between the parent pharma company column and Sponsor column

keyValuePairSponsorColumn = sponsorColumnFiltered.map(lambda x: (x, 1)) # Creating a key value pair for the reducing function

valueSumSponsorColumn = keyValuePairSponsorColumn.reduceByKey(lambda x, y: x + y) # Reducing by Key to get sum of distinct data

sortedSponsorColumn = valueSumSponsorColumn.sortBy(lambda x: x[1], ascending=False) # Sorting rdd by sum in decending order

sortedSponsorColumn.take(10) # Veiwing top 10 results
```

▸ (4) Spark Jobs

```
[('National Cancer Institute (NCI)', 3100),
 ('M.D. Anderson Cancer Center', 2238),
 ('Merck Sharp & Dohme Corp.', 2184),
 ('Mayo Clinic', 2097),
 ('Assistance Publique - Hôpitaux de Paris', 2043),
 ('Novartis Pharmaceuticals', 1962),
 ('Massachusetts General Hospital', 1823),
 ('Assiut University', 1806),
 ('Hoffmann-La Roche', 1761),
 ('National Taiwan University Hospital', 1720)]
```

Command took 7.82 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 13:16:07 on My Cluster 7

Figure 3.28

Description of main ideas for question 4 (RDD)

I extracted the Sponsor and Parent_Company columns from a clinical trial dataset and used the subtract function to identify companies in the Sponsors column but not in the Parent _Company column. I then created a key-value pair, reduced by key, and sorted in descending order to view top 10.

QUESTION 4 DATAFRAME SOLUTIONS

clinicaltrial_2023 Question 4 (Dataframe)

Cmd 7

```python
# QUESTION 4

# Extracting the "Parent_Company" column from the pharma file and converting to a list
pharma_list = create_dataframe(pharma_file).select("Parent_Company").rdd.flatMap(lambda x: x).collect()

# Extracting the "Sponsor" column from the clinical trial file
ct_sponsor_dataframe = ct_dataframe.select("Sponsor")

# Getting the sum of each distinct data in the "Sponsor" column
sponsorColumnCount = ct_sponsor_dataframe.groupBy("Sponsor").count().orderBy("count", ascending=False)

# Filtering the Sponsor column using the pharma list to find the difference between both columns
nonPharmaSponsors = sponsorColumnCount.filter(~ct_sponsor_dataframe.Sponsor.isin(pharma_list))

# Displaying the result
nonPharmaSponsors.show(10, truncate=False)
```

▸ (4) Spark Jobs

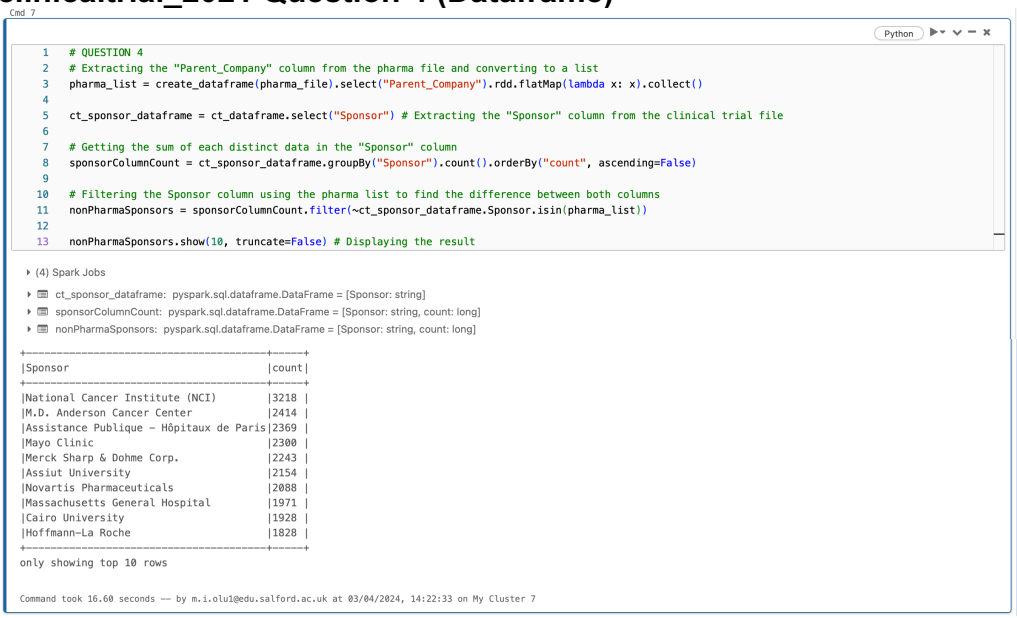
▸ ct_sponsor_dataframe: pyspark.sql.dataframe.DataFrame = [Sponsor: string]
▸ sponsorColumnCount: pyspark.sql.dataframe.DataFrame = [Sponsor: string, count: long]
▸ nonPharmaSponsors: pyspark.sql.dataframe.DataFrame = [Sponsor: string, count: long]

```
+------------------------------------------------------------+-----+
|Sponsor                                                     |count|
+------------------------------------------------------------+-----+
|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 |
+------------------------------------------------------------+-----+
only showing top 10 rows
```

Command took 23.66 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 14:00:37 on My Cluster 7

Figure 3.26

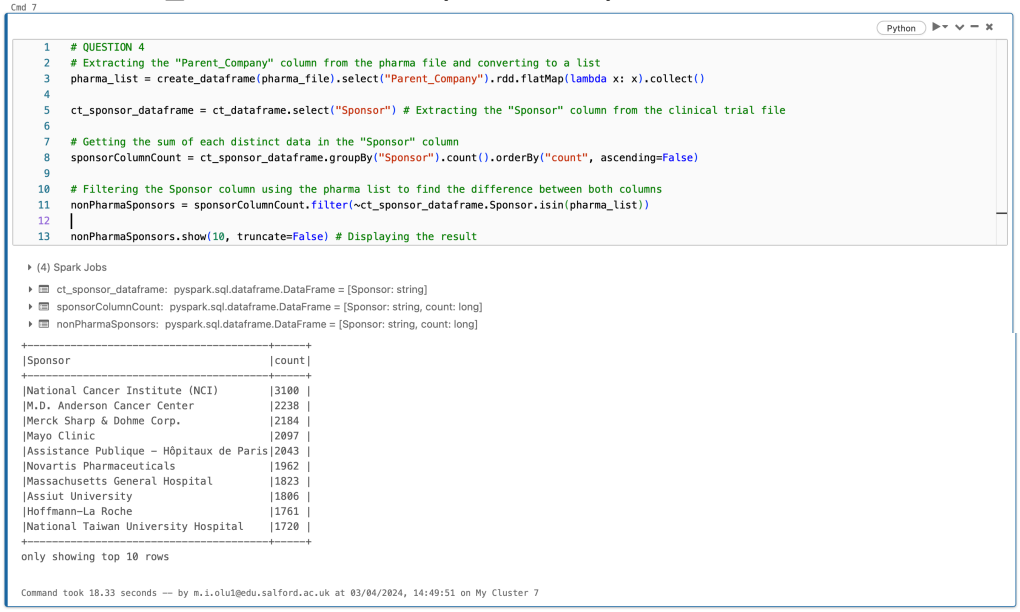
clinicaltrial_2021 Question 4 (Dataframe)



```
Cmd 7
Python
1  # QUESTION 4
2  # Extracting the "Parent_Company" column from the pharma file and converting to a list
3  pharma_list = create_dataframe(pharma_file).select("Parent_Company").rdd.flatMap(lambda x: x).collect()
4
5  ct_sponsor_dataframe = ct_dataframe.select("Sponsor") # Extracting the "Sponsor" column from the clinical trial file
6
7  # Getting the sum of each distinct data in the "Sponsor" column
8  sponsorColumnCount = ct_sponsor_dataframe.groupBy("Sponsor").count().orderBy("count", ascending=False)
9
10 # Filtering the Sponsor column using the pharma list to find the difference between both columns
11 nonPharmaSponsors = sponsorColumnCount.filter(~ct_sponsor_dataframe.Sponsor.isin(pharma_list))
12
13 nonPharmaSponsors.show(10, truncate=False) # Displaying the result

▸ (4) Spark Jobs
▸ ct_sponsor_dataframe:  pyspark.sql.dataframe.DataFrame = [Sponsor: string]
▸ sponsorColumnCount:  pyspark.sql.dataframe.DataFrame = [Sponsor: string, count: long]
▸ nonPharmaSponsors:  pyspark.sql.dataframe.DataFrame = [Sponsor: string, count: long]

+----------------------------------------+-----+
|Sponsor                                 |count|
+----------------------------------------+-----+
|National Cancer Institute (NCI)         |3218 |
|M.D. Anderson Cancer Center             |2414 |
|Assistance Publique - Hôpitaux de Paris|2369 |
|Mayo Clinic                             |2300 |
|Merck Sharp & Dohme Corp.               |2243 |
|Assiut University                       |2154 |
|Novartis Pharmaceuticals                |2088 |
|Massachusetts General Hospital          |1971 |
|Cairo University                        |1928 |
|Hoffmann-La Roche                       |1828 |
+----------------------------------------+-----+
only showing top 10 rows

Command took 16.60 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 14:22:33 on My Cluster 7
```

Figure 3.27

clinicaltrial_2020 Question 4 (Dataframe)



```
Cmd 7
Python
1  # QUESTION 4
2  # Extracting the "Parent_Company" column from the pharma file and converting to a list
3  pharma_list = create_dataframe(pharma_file).select("Parent_Company").rdd.flatMap(lambda x: x).collect()
4
5  ct_sponsor_dataframe = ct_dataframe.select("Sponsor") # Extracting the "Sponsor" column from the clinical trial file
6
7  # Getting the sum of each distinct data in the "Sponsor" column
8  sponsorColumnCount = ct_sponsor_dataframe.groupBy("Sponsor").count().orderBy("count", ascending=False)
9
10 # Filtering the Sponsor column using the pharma list to find the difference between both columns
11 nonPharmaSponsors = sponsorColumnCount.filter(~ct_sponsor_dataframe.Sponsor.isin(pharma_list))
12
13 nonPharmaSponsors.show(10, truncate=False) # Displaying the result

▸ (4) Spark Jobs
▸ ct_sponsor_dataframe:  pyspark.sql.dataframe.DataFrame = [Sponsor: string]
▸ sponsorColumnCount:  pyspark.sql.dataframe.DataFrame = [Sponsor: string, count: long]
▸ nonPharmaSponsors:  pyspark.sql.dataframe.DataFrame = [Sponsor: string, count: long]

+----------------------------------------+-----+
|Sponsor                                 |count|
+----------------------------------------+-----+
|National Cancer Institute (NCI)         |3100 |
|M.D. Anderson Cancer Center             |2238 |
|Merck Sharp & Dohme Corp.               |2184 |
|Mayo Clinic                             |2097 |
|Assistance Publique - Hôpitaux de Paris|2043 |
|Novartis Pharmaceuticals                |1962 |
|Massachusetts General Hospital          |1823 |
|Assiut University                       |1806 |
|Hoffmann-La Roche                       |1761 |
|National Taiwan University Hospital     |1720 |
+----------------------------------------+-----+
only showing top 10 rows

Command took 18.33 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 14:49:51 on My Cluster 7
```

Figure 3.28

Description of main ideas for question 4 (Dataframe)

The **Parent_Company** column was extracted from the **pharma** dataframe and collected to create a list. This list was used in line 11 to filter the **Sponsor** column which was already counted by the **Sponsor** column and ordered by count. The first 10 results were shown to reveal the top 10 results.

QUESTION 4 SQL SOLUTIONS

clinicaltrial_2023 SQL

Cmd 9

```
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
```

(3) Spark Jobs

Table

	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

10 rows | 14.50 seconds runtime — Refreshed 9 minutes ago

Command took 14.50 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 15:22:09 on My Cluster 7

Figure 3.29

clinicaltrial_2021 SQL

Cmd 9

```
CREATE OR REPLACE TEMP VIEW non_pharma_sponsor AS SELECT Sponsor FROM clinicaltrial_2021 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
```

(3) Spark Jobs

Table

	Sponsor	count
1	National Cancer Institute (NCI)	3218
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2369
4	Mayo Clinic	2300
5	Merck Sharp & Dohme Corp.	2243
6	Assiut University	2154

10 rows | 5.71 seconds runtime — Refreshed 1 minute ago

Command took 5.71 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 15:38:46 on My Cluster 7

Figure 3.30

clinicaltrial_2020 SQL

Cmd 9

```
CREATE OR REPLACE TEMP VIEW non_pharma_sponsor AS SELECT Sponsor FROM clinicaltrial_2020 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
```

(3) Spark Jobs

Table

	Sponsor	count
1	National Cancer Institute (NCI)	3100
2	M.D. Anderson Cancer Center	2238
3	Merck Sharp & Dohme Corp.	2184
4	Mayo Clinic	2097
5	Assistance Publique - Hôpitaux de Paris	2043
6	Novartis Pharmaceuticals	1962

10 rows | 5.04 seconds runtime — Refreshed now

Command took 5.04 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 15:41:21 on My Cluster 7

Figure 3.29

Description of main ideas for question 4 (SQL)

A temporary view was created from the clinical trial table with a clause where the sponsor column does not contain items from the Parent_Company column in the pharma file. Then a query was written to count all the items grouped by the Sponsor column in order to get the frequency of each unique sponsor.

Discussion of results

According to the results from the 2023 dataset, the top 10 non-parasitical sponsors are:

- National Cancer Institute (NCI)
- M.D. Anderson Cancer Centre
- Merck Sharp & Dohme Corp.
- Mayo Clinic
- Assistance Publique - Hôpitaux de Paris
- Novartis Pharmaceuticals
- Massachusetts General Hospital
- Assiut University
- Hoffmann-La Roche
- National Taiwan University Hospital

The most common sponsor in the 2023 dataset is the National Cancer Institute (NCI) with 3,410 studies, followed by Assuit University with 3,335 studies.

QUESTION 5: PLOT NUMBER OF COMPLETED STUDIES FOR EACH MONTH IN 2023.

ASSUMPTIONS:

In this analysis, I assumed that all completed studies carry “completed” or “COMPLETED” in the status column depending on the dataset.

QUESTION 5 RDD SOLUTIONS

The visualizations for the solutions are also plotted using Python’s **matplotlib** library in *Figure 3.27* for 2023, *Figure 3.29* for 2021 and *Figure 3.31*

clinicaltrial_2023 RDD

Cmd 16

```python
#number of completed studies for each month in 2023
status_column_index = CT_RDD_CLEAN.first().index('Status') # Getting index of "Status" column
completion_column_index = CT_RDD_CLEAN.first().index('Completion') # Getting index of "Completion" column

dateDelimeter = {               # Creating dictionary for selection of delimeter using file names and keys
    "clinicaltrial_2023": "–",
    "clinicaltrial_2021": " ",
    "clinicaltrial_2020": " ",
}

orderedMonths = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"] # Defining an ordered list of months

completedStudies = CT_RDD_CLEAN.filter(lambda row: row[status_column_index] in ("COMPLETED", "Completed")) # Filtering rows with "Status" column as completed

completionColumn = completedStudies.map(lambda row: row[completion_column_index]).filter(lambda x: x not in ("", " ")) # Extracting "Completion" column and filtering the empty rows

extractedDates = completionColumn.map(lambda x: x.split(dateDelimeter[clinical_tiral_file])) # Spliting the date data by delimeter from the dateDelimeter dictionary

extractedDates = extractedDates.map(lambda x: (x[1], orderedMonths.index(x[0]) + 1) if x[1].startswith('20') else (x[0], int(x[1]))) #Creating Key value pairs with specific conditions

extractedDates = extractedDates.filter(lambda x: x[0] == clinical_tiral_file.split("_")[1]) # Filtering dates by year. eg 2023, 2021, 2020

extractedDates = extractedDates.map(lambda x: (x[1], 1)).reduceByKey(lambda a, b: a + b) # Getting sums of distinct data using the reducing function

extractedDates = extractedDates.sortBy(lambda x: x[0]).map(lambda x: (orderedMonths[x[0] - 1], x[1])) # Sorting by months an

extractedDates.collect() # Displaying results
```

(5) Spark Jobs

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

Command took 10.68 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 16:25:59 on My Cluster 7

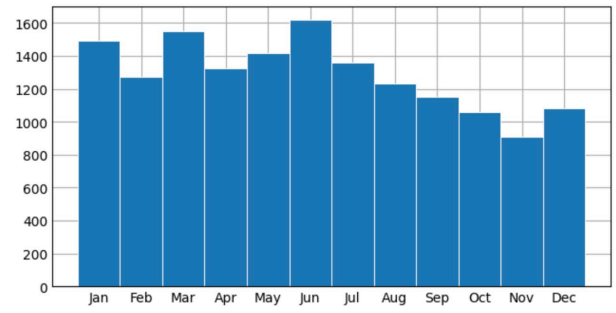
Figure 3.26

Matplotlib visualization for clinicaltrial_2023

Cmd 18

```python
import matplotlib.pyplot as plt # Importing pyplot module in the matplotlib library

x = extractedDates.map(lambda x: x[0]).collect() # Extracting the months column for the x axis
y = extractedDates.map(lambda x: x[1]).collect() # Extracting the frequency column for the y axis

plt.figure(figsize=(6, 3)) # Setting the dimensions of the graph frame

plt.bar(x, y, width=1, edgecolor="white", linewidth=0.7) # Passing the x and y axis to the bar method in the pyplot object

plt.show() # Displaying the graph
```

(2) Spark Jobs



Command took 0.79 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 16:25:59 on My Cluster 7

Figure 2.27

clinicaltrial_2021 RDD

Cmd 16

```python
#number of completed studies for each month in 2023
status_column_index = CT_RDD_CLEAN.first().index('Status') # Getting index of "Status" column
completion_column_index = CT_RDD_CLEAN.first().index('Completion') # Getting index of "Completion" column

dateDelimeter = {               # Creating dictionary for selection of delimeter using file names and keys
    "clinicaltrial_2023": "-",
    "clinicaltrial_2021": " ",
    "clinicaltrial_2020": " ",
}

orderedMonths = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"] # Defining an ordered list of months

completedStudies = CT_RDD_CLEAN.filter(lambda row: row[status_column_index] in ("COMPLETED", "Completed")) # Filtering rows with "Status" column as completed

completionColumn = completedStudies.map(lambda row: row[completion_column_index]).filter(lambda x: x not in ("", " ")) # Extracting "Completion" column and filtering the empty rows

extractedDates = completionColumn.map(lambda x: x.split(dateDelimeter[clinical_tiral_file])) # Spliting the date data by delimeter from the dateDelimeter dictionary

extractedDates = extractedDates.map(lambda x: (x[1], orderedMonths.index(x[0]) + 1) if x[1].startswith('20') else (x[0], int(x[1]))) #Creating Key value pairs with specific conditions

extractedDates = extractedDates.filter(lambda x: x[0] == clinical_tiral_file.split("_")[1]) # Filtering dates by year. eg 2023, 2021, 2020

extractedDates = extractedDates.map(lambda x: (x[1], 1)).reduceByKey(lambda a, b: a + b) # Getting sums of distinct data using the reducing function

extractedDates = extractedDates.sortBy(lambda x: x[0]).map(lambda x: (orderedMonths[x[0] - 1], x[1])) # Sorting by months an

extractedDates.collect() # Displaying results
```

(5) Spark Jobs

```
[('Jan', 1131),
 ('Feb', 934),
 ('Mar', 1227),
 ('Apr', 967),
 ('May', 984),
 ('Jun', 1094),
 ('Jul', 819),
 ('Aug', 700),
 ('Sep', 528),
 ('Oct', 187)]
```

Command took 5.49 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 19:21:25 on My Cluster 8

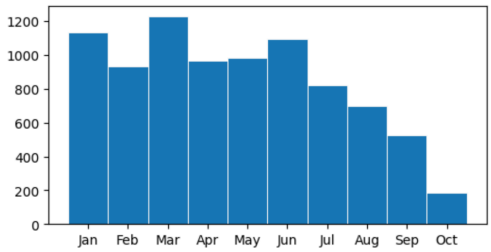
Figure 3.28

Matplotlib visualization for clinicaltrial_2021

Cmd 18

```python
import matplotlib.pyplot as plt # Importing pyplot module in the matplotlib library

x = extractedDates.map(lambda x: x[0]).collect() # Extracting the months column for the x axis
y = extractedDates.map(lambda x: x[1]).collect() # Extracting the frequency column for the y axis

plt.figure(figsize=(6, 3)) # Setting the dimensions of the graph frame

plt.bar(x, y, width=1, edgecolor="white", linewidth=0.7) # Passing the x and y axis to the bar method in the pyplot object

plt.show() # Displaying the graph
```

(2) Spark Jobs



Command took 1.87 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 19:21:25 on My Cluster 8

Figure 2.29

clinicaltrial_2020 RDD

Cmd 16

```python
#number of completed studies for each month in 2023
status_column_index = CT_RDD_CLEAN.first().index('Status') # Getting index of "Status" column
completion_column_index = CT_RDD_CLEAN.first().index('Completion') # Getting index of "Completion" column

dateDelimeter = {               # Creating dictionary for selection of delimeter using file names and keys
    "clinicaltrial_2023": "-",
    "clinicaltrial_2021": " ",
    "clinicaltrial_2020": " ",
}

orderedMonths = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"] # Defining an ordered list of months

completedStudies = CT_RDD_CLEAN.filter(lambda row: row[status_column_index] in ("COMPLETED", "Completed")) # Filtering rows with "Status" column as completed

completionColumn = completedStudies.map(lambda row: row[completion_column_index]).filter(lambda x: x not in ("", " ")) # Extracting "Completion" column and filtering the empty rows

extractedDates = completionColumn.map(lambda x: x.split(dateDelimeter[clinical_tiral_file])) # Spliting the date data by delimeter from the dateDelimeter dictionary

extractedDates = extractedDates.map(lambda x: (x[1], orderedMonths.index(x[0]) + 1) if x[1].startswith('20') else (x[0], int(x[1]))) #Creating Key value pairs with specific conditions

extractedDates = extractedDates.filter(lambda x: x[0] == clinical_tiral_file.split("_")[1]) # Filtering dates by year. eg 2023, 2021, 2020

extractedDates = extractedDates.map(lambda x: (x[1], 1)).reduceByKey(lambda a, b: a + b) # Getting sums of distinct data using the reducing function

extractedDates = extractedDates.sortBy(lambda x: x[0]).map(lambda x: (orderedMonths[x[0] - 1], x[1])) # Sorting by months an

extractedDates.collect() # Displaying results
```

(5) Spark Jobs

```
[('Jan', 1544),
 ('Feb', 1286),
 ('Mar', 1740),
 ('Apr', 1080),
 ('May', 1176),
 ('Jun', 1424),
 ('Jul', 1237),
 ('Aug', 1126),
 ('Sep', 1167),
 ('Oct', 1176),
 ('Nov', 1078),
 ('Dec', 2084)]
```

Command took 3.70 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 19:27:29 on My Cluster 8

Figure 3.26

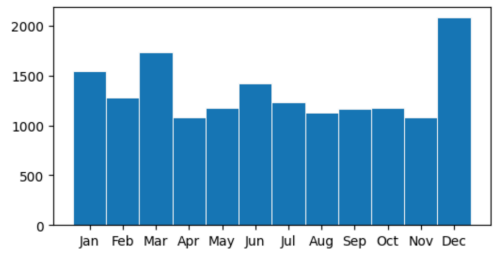
Matplotlib visualization for clinicaltrial_2020

Cmd 18

Python

```
1  import matplotlib.pyplot as plt # Importing pyplot module in the matplotlib library
2
3  x = extractedDates.map(lambda x: x[0]).collect() # Extracting the months column for the x axis
4  y = extractedDates.map(lambda x: x[1]).collect() # Extracting the frequency column for the y axis
5
6  plt.figure(figsize=(6, 3)) # Setting the dimensions of the graph frame
7
8  plt.bar(x, y, width=1, edgecolor="white", linewidth=0.7) # Passing the x and y axis to the bar method in the pyplot object
9
10 plt.show() # Displaying the graph
```

▸ (2) Spark Jobs



Command took 0.69 seconds -- by m.i.olu1@edu.salford.ac.uk at 03/04/2024, 19:27:29 on My Cluster 8

Figure 2.31

Description of main ideas Question 5 (RDD)

The code extracts completed studies by month for each dataset by filtering by status column ('COMPLETED' for 2023) and 'completed' for all others. It transforms RDD by extracting 'Completion' column and splitting them by the dates delimiter to extract the year and month components of the dates. Next it filters by year 2023, 2020, or 2021, and creates a key-value pair with months. It then reduces by key to get frequency.

The graph was plotted using matplotlib with X axis as months and Y axis as frequency.

QUESTION 5 DATAFRAME SOLUTIONS

clinicaltrial_2023 Dataframe

Cmd 8

Python

```
orderedMonths = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"] # Defining an ordered list of months

# Conditional loop to split column based on file because of different delimiters
if clinical_tiral_file.endswith("2023"):
    # Splits the completion column based on the "-" delimeter and creates new columns from the extracted values
    dateDataframe = ct_dataframe.withColumn('Year', split('Completion', "-")[0]).withColumn('Month', split('Completion', "-")[1])
else:
    # Splits the completion column based on the " " delimeter and creates new columns from the extracted values
    dateDataframe = ct_dataframe.withColumn('Year', split('Completion', " ")[1]).withColumn('Month', split('Completion', " ")[0])

    # Removes rows with empty values in month column by filtering based on the orderedMonth list which was initialy defined
    dateDataframe = dateDataframe.filter(dateDataframe.Month.isin(orderedMonths))

    month_index_udf = udf(lambda x: orderedMonths.index(x) + 1) # Udf to convert month name to corresponding numbers using the orderedMonth

    dateDataframe = dateDataframe.withColumn('Month', month_index_udf(col('Month'))) # Udf to convert month name to corresponding numbers using the
    orderedMonth

# Selecting rows with Status column as "COMPLETED" or "Completed".
completionDateDataframe = dateDataframe.filter(dateDataframe.Status.isin(["COMPLETED", "Completed"])).select("Month", "Year", "Status")

# Converting Month column data type to integer to run arithemetic operations on it.
completionDateDataframe = completionDateDataframe.withColumn('Month', col('Month').cast('int'))

month_name_udf = udf(lambda x: orderedMonths[x - 1]) # Udf to replace month number to month name using the month number as index

# Filtering dates by year (2023, 2021, 2020), then counting dataframe grouped by month. then using UDF to convert month number to month name.
completionDateDataframe = completionDateDataframe.filter(completionDateDataframe.Year.isin([clinical_tiral_file.split("_")[1]])).groupBy("Month").
count().orderBy("Month", ascending=True).withColumn('Month', month_name_udf(col('Month')))

completionDateDataframe.show() # Displaying final results
```

▸ (4) Spark Jobs

▸ dateDataframe: pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 14 more fields]

▸ completionDateDataframe: pyspark.sql.dataframe.DataFrame = [Month: string, count: long]

```
+-----+-----+
|Month|count|
+-----+-----+
|  Jan| 1494|
|  Feb| 1272|
|  Mar| 1552|
|  Apr| 1324|
|  May| 1415|
|  Jun| 1619|
|  Jul| 1360|
|  Aug| 1230|
|  Sep| 1152|
|  Oct| 1058|
|  Nov|  909|
|  Dec| 1082|
+-----+-----+
```

Command took 16.56 seconds -- by m.i.olu1@edu.salford.ac.uk at 04/04/2024, 09:48:12 on My Cluster 10

Figure 3.32

clinicaltrial_2021 Dataframe

Cmd 8

Python

```python
orderedMonths = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"] # Defining an ordered list of months

# Conditional loop to split column based on file because of different delimiters
if clinical_tiral_file.endswith("2023"):
    # Splits the completion column based on the "-" delimeter and creates new columns from the extracted values
    dateDataframe = ct_dataframe.withColumn('Year', split('Completion', "-")[0]).withColumn('Month', split('Completion', "-")[1])
else:
    # Splits the completion column based on the " " delimeter and creates new columns from the extracted values
    dateDataframe = ct_dataframe.withColumn('Year', split('Completion', " ")[1]).withColumn('Month', split('Completion', " ")[0])

    # Removes rows with empty values in month column by filtering based on the orderedMonth list which was initialy defined
    dateDataframe = dateDataframe.filter(dateDataframe.Month.isin(orderedMonths))

    month_index_udf = udf(lambda x: orderedMonths.index(x) + 1) # Udf to convert month name to corresponding numbers using the orderedMonth

    dateDataframe = dateDataframe.withColumn('Month', month_index_udf(col('Month'))) # Udf to convert month name to corresponding numbers using the orderedMonth

# Selecting rows with Status column as "COMPLETED" or "Completed".
completionDateDataframe = dateDataframe.filter(dateDataframe.Status.isin(["COMPLETED", "Completed"])).select("Month", "Year", "Status")

# Converting Month column data type to integer to run arithemetic operations on it.
completionDateDataframe = completionDateDataframe.withColumn('Month', col('Month').cast('int'))

month_name_udf = udf(lambda x: orderedMonths[x - 1]) # Udf to replace month number to month name using the month number as index

# Filtering dates by year (2023, 2021, 2020), then counting dataframe grouped by month. then using UDF to convert month number to month name.
completionDateDataframe = completionDateDataframe.filter(completionDateDataframe.Year.isin([clinical_tiral_file.split("_")[1]])).groupBy("Month").count().orderBy("Month", ascending=True).withColumn('Month', month_name_udf(col('Month')))

completionDateDataframe.show() # Displaying final results
```

▸ (4) Spark Jobs

▸ dateDataframe: pyspark.sql.dataframe.DataFrame = [Id: string, Sponsor: string ... 9 more fields]

▸ completionDateDataframe: pyspark.sql.dataframe.DataFrame = [Month: string, count: long]

```
+-----+-----+
|Month|count|
+-----+-----+
|  Jan| 1131|
|  Feb|  934|
|  Mar| 1227|
|  Apr|  967|
|  May|  984|
|  Jun| 1094|
|  Jul|  819|
|  Aug|  700|
|  Sep|  528|
|  Oct|  187|
+-----+-----+
```

1

Command took 6.69 seconds -- by m.i.olu1@edu.salford.ac.uk at 04/04/2024, 10:41:06 on My Cluster 10

Figure 3.32

clinicaltrial_2020 Dataframe

Cmd 8

```python
orderedMonths = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"] # Defining an ordered list of months

# Conditional loop to split column based on file because of different delimiters
if clinical_tiral_file.endswith("2023"):
    # Splits the completion column based on the "-" delimeter and creates new columns from the extracted values
    dateDataframe = ct_dataframe.withColumn('Year', split('Completion', "-")[0]).withColumn('Month', split('Completion', "-")[1])
else:
    # Splits the completion column based on the " " delimeter and creates new columns from the extracted values
    dateDataframe = ct_dataframe.withColumn('Year', split('Completion', " ")[1]).withColumn('Month', split('Completion', " ")[0])

    # Removes rows with empty values in month column by filtering based on the orderedMonth list which was initialy defined
    dateDataframe = dateDataframe.filter(dateDataframe.Month.isin(orderedMonths))

    month_index_udf = udf(lambda x: orderedMonths.index(x) + 1) # Udf to convert month name to corresponding numbers using the orderedMonth

    dateDataframe = dateDataframe.withColumn('Month', month_index_udf(col('Month'))) # Udf to convert month name to corresponding numbers using the
    orderedMonth

# Selecting rows with Status column as "COMPLETED" or "Completed".
completionDateDataframe = dateDataframe.filter(dateDataframe.Status.isin(["COMPLETED", "Completed"])).select("Month", "Year", "Status")

# Converting Month column data type to integer to run arithemetic operations on it.
completionDateDataframe = completionDateDataframe.withColumn('Month', col('Month').cast('int'))

month_name_udf = udf(lambda x: orderedMonths[x - 1]) # Udf to replace month number to month name using the month number as index

# Filtering dates by year (2023, 2021, 2020), then counting dataframe grouped by month. then using UDF to convert month number to month name.
completionDateDataframe = completionDateDataframe.filter(completionDateDataframe.Year.isin([clinical_tiral_file.split("_")[1]])).groupBy("Month").
count().orderBy("Month", ascending=True).withColumn('Month', month_name_udf(col('Month')))

completionDateDataframe.show() # Displaying final results
```

```
▸ (4) Spark Jobs
▸ dateDataframe:  pyspark.sql.dataframe.DataFrame = [Id: string, Sponsor: string ... 9 more fields]
▸ completionDateDataframe:  pyspark.sql.dataframe.DataFrame = [Month: string, count: long]
+-----+-----+
|Month|count|
+-----+-----+
|  Jan| 1544|
|  Feb| 1286|
|  Mar| 1740|
|  Apr| 1080|
|  May| 1176|
|  Jun| 1424|
|  Jul| 1237|
|  Aug| 1126|
|  Sep| 1167|
|  Oct| 1176|
|  Nov| 1078|
|  Dec| 2084|
+-----+-----+

Command took 5.56 seconds -- by m.i.olu1@edu.salford.ac.uk at 04/04/2024, 10:49:07 on My Cluster 10
```

Figure 3.32

Description of main ideas Question 5 (Dataframe)

The dataframe was filtered by rows with 'COMPLETED' status for 2023 or 'Completed' status for 2021 and 2020, then split Completion column into year and month components, counted and grouped by month, and then used the month_name_udf function to replace month numbers.

QUESTION 5 SQL SOLUTIONS.

Some UDFs are created to dynamically extract dates delimiter and also ‘Year’ and ‘month’ components of the Completion dates for various dates formats.

Cmd 18

```sql
-- Creating user defined functions for extraction of dates components dynamically for all years
CREATE OR REPLACE FUNCTION date_delimiter()
  RETURNS STRING
  RETURN SELECT FIRST(value) FROM bdtt_db.infos WHERE info_type == 'date_delimiter';

CREATE OR REPLACE FUNCTION date_index_month()
  RETURNS INT
  RETURN SELECT CASE FIRST(value) WHEN ' ' THEN 0 ELSE 1 END AS index FROM bdtt_db.infos WHERE info_type == 'date_delimiter';

CREATE OR REPLACE FUNCTION date_index_year()
  RETURNS INT
  RETURN SELECT CASE FIRST(value) WHEN ' ' THEN 1 ELSE 0 END AS index FROM bdtt_db.infos WHERE info_type == 'date_delimiter';

CREATE OR REPLACE FUNCTION file_year()
  RETURNS INT
  RETURN SELECT FIRST(value) FROM bdtt_db.infos WHERE info_type == 'year';
```

OK

Command took 0.74 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/05/2024, 21:44:46 on cluster

The questions are then answered below using the UDFs created in the above image.

clinicaltrial_2023 SQL

Cmd 19

```sql
-- Plot number of completed studies for each month in 2023

create or replace temp view extractedDates as SELECT split(Completion, (select date_delimiter()))[(select date_index_month())] as Month, count(*)
as Count
From clinical_trial_table
where clinical_trial_table.Status in ('COMPLETED', 'Completed') and split(clinical_trial_table.Completion, (select date_delimiter()))[(select
date_index_year())] == (select file_year())
group by Month
order by Month;

Select
CASE Month
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
        ELSE Month
    END AS Month, Count from extractedDates
```

(6) Spark Jobs

Table

	Month	Count
1	January	1494
2	February	1272
3	March	1552
4	April	1324
5	May	1415
6	June	1619

12 rows | 13.03 seconds runtime

Refreshed 4 minutes ago

Command took 13.03 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/05/2024, 22:14:12 on cluster

Figure 3.32

clinicaltrial_2021 SQL

Cmd 19

SQL

```
-- Plot number of completed studies for each month in 2023

create or replace temp view extractedDates as SELECT split(Completion, (select date_delimiter()))[(select date_index_month())] as Month, count(*)
as Count
From clinical_trial_table
where clinical_trial_table.Status in ('COMPLETED', 'Completed') and split(clinical_trial_table.Completion, (select date_delimiter()))[(select
date_index_year())] == (select file_year())
group by Month
order by Month;

Select
CASE Month
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
        ELSE Month
    END AS Month, Count from extractedDates
```

(6) Spark Jobs

Table +

	Month	Count
1	Apr	967
2	Aug	700
3	Feb	934
4	Jan	1131
5	Jul	819
6	Jun	1094

10 rows | 5.47 seconds runtime — Refreshed 1 minute ago

1

Command took 5.47 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/05/2024, 22:24:32 on cluster

Figure 3.32

clinicaltrial_2020 SQL

Cmd 19

SQL

```
-- Plot number of completed studies for each month in 2023

create or replace temp view extractedDates as SELECT split(Completion, (select date_delimiter()))[(select date_index_month())] as Month, count(*)
as Count
From clinical_trial_table
where clinical_trial_table.Status in ('COMPLETED', 'Completed') and split(clinical_trial_table.Completion, (select date_delimiter()))[(select
date_index_year())] == (select file_year())
group by Month
order by Month;

Select
CASE Month
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
        ELSE Month
    END AS Month, Count from extractedDates
```

(6) Spark Jobs

Table +

	Month	Count
1	Apr	1080
2	Aug	1126
3	Dec	2084
4	Feb	1286
5	Jan	1544
6	Jul	1237

12 rows | 4.87 seconds runtime — Refreshed now

Command took 4.87 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/05/2024, 22:27:28 on cluster

Figure 3.32

Description of main ideas Question 5 (SQL)

An SQL view was created to extract months and years from completion dates by splitting with delimiters and taking index of the year and month using various UDFs, counting by month, and filtering by status ('COMPLETED' for 2023 or 'Completed' for 2021 and 2020). The resulting view was queried using a CASE block to exchange month numbers for corresponding days.

Discussion of result for Question 5

The results from 2023 shows that the highest number of studies were completed in June with 1,619 completed clinical trials, closely followed by the month of March with 1,552 clinical trials and the lowest being the month of November with 909 clinical trials.

4. FURTHER ANALYSIS 1 (RDD): 10 most common sponsors that are pharmaceutical companies, along with the number of clinical trials they have sponsored

Assumptions:

I assumed that all the pharmaceutical companies are in the Parent_Company column in the pharma file.

clinicaltrial_2023 RDD

Cmd 32

```
# Further analysis 1: Find the 10 most common sponsors that are pharmaceutical companies, along with the number of clinical trials they have sponsored
ct_sponsor_col_index = CT_RDD_CLEAN.first().index('Sponsor') # Finding the index of the Sponsor column

parent_pharm_comp = PHARMA_RDD_CLEAN.map(lambda x: x[1].replace('"', '')) # Extracting the parent pharma company column and also cleaning the data

sponsorColumn = CT_RDD_CLEAN.map(lambda x: x[ct_sponsor_col_index]) # Extracting the Sponsor column from the main RDD

sponsorColumn = sponsorColumn.filter(lambda row: row != 'Sponsor') # Removing the Header row from the RDD

sponsorColumnFiltered = sponsorColumn.subtract(parent_pharm_comp) # Finding the difference between the parent pharma company column and Sponsor column

sponsorColumnRefiltered = sponsorColumn.subtract(sponsorColumnFiltered) # Finding difference between the Sponsor colum and the non-pharma Sponsor column, creating a column of only pharma Sponsor companies

keyValuePairSponsorColumn = sponsorColumnRefiltered.map(lambda x: (x, 1)) # Creating a key value pair for the reducing function

valueSumSponsorColumn = keyValuePairSponsorColumn.reduceByKey(lambda x, y: x + y) # Reducing by Key to get sum of distinct data

sortedSponsorColumn = valueSumSponsorColumn.sortBy(lambda x: x[1], ascending=False) # Sorting rdd by sum in decending order

sortedSponsorColumn.take(10) # Veiwing top 10 results
```

(6) Spark Jobs

```
[('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 27.36 seconds -- by m.i.olu1@edu.salford.ac.uk at 30/04/2024, 14:38:06 on cmpt

Figure 4.1

Discussion of main ideas for further analysis 1

The sponsor column was extracted, subtracted from the Parent_Company column to obtain non-pharma companies, and then subtracted from the Sponsor column to obtain all pharma companies, and frequency was calculated.

Discussion of result for further analysis 1

The result shows that in 2023, `GlaxoSmithKline` with 3,482 trials was the top common pharmaceutical sponsor in 2023 clinical trials, followed by `Pfizer` with 3,045 trials and then `Gilead Science` as the 10th with 625 trials.

5. FURTHER ANALYSIS 2 (DATAFRAME): Number of studies which have been terminated.

Assumptions:

I assumed that all terminated studies carry ‘TERMINATED’ or ‘Terminated’ in the status column.

clinicaltrial_2023 DATAFRAME

Figure 5.1

Discussion of main ideas for further analysis 2

I filtered the rows from the dataframe where Status is ‘Terminated or TERMINATED’. Then I counted the resulting rows distinctly to get the final number.

Discussion of result for further analysis 2

The result shows that there were 28,022 clinical trials which were terminated, making up about 5.8% of the 483,422 total trials conducted in 2023.

6. FURTHER ANALYSIS 3 (SQL): Top 5 conditions sponsored by non-pharmaceutical companies.

Assumptions:

I assumed that all the pharmaceutical companies are in the Parent_Company column in the pharma file.

clinicaltrial_2023 SQL

Cmd 21

```sql
-- FURTHER ANALYSIS: Top 5 conditions sponsored by non-pharmaceutical companies

CREATE OR REPLACE TEMP VIEW non_pharma_sponsor_trials AS SELECT * FROM clinical_trial_table WHERE Sponsor NOT IN (SELECT Parent_Company FROM pharma);
CREATE OR REPLACE TEMP VIEW non_pharma_trial_conditions AS SELECT explode(split(non_pharma_sponsor_trials.conditions, (SELECT conditions_delimiter ()))) as conditions FROM non_pharma_sponsor_trials;
SELECT conditions, count(*) as count FROM non_pharma_trial_conditions
GROUP BY conditions
ORDER BY count DESC
LIMIT 5
```

(5) Spark Jobs

Table

	conditions	count
1	Healthy	8391
2	Breast Cancer	7275
3	Obesity	6445
4	Stroke	4043
5	Depression	3884

5 rows | 13.65 seconds runtime Refreshed now

Command took 13.65 seconds -- by m.i.olu1@edu.salford.ac.uk at 01/05/2024, 22:30:18 on cluster

Figure 6.1

Discussion of main ideas for further analysis 3

A temp view was created by filtering by the sponsor column nested query where sponsor isn't in the Parent_Company column. This view was used to create another view of the conditions column which was split and exploded, finally a select query was performed on this view to count the conditions column, grouped by conditions and ordered by the count

Discussion of result for further analysis 3

The result shows that there the top 5 conditions sponsored by non-pharmaceutical companies in 2023 are 'Healthy, Breast Cancer, Obesity, Stroke and Depression'.