



University of
Salford
MANCHESTER

Big Data Tools and Techniques Assignment

Name: Ben Miller (@00498625)

Submitted to: University of Salford, Manchester

Module: Big Data Tools and Techniques

Qualification: Master's in data science

2021/22

Preparation:

During this assignment, 3 datafiles will be used. The 3 files are pharma.csv, mesh.csv and clinicaltrial_2021.csv.gz. The description of the file content are as follows,

Clinicaltrial_2021.csv.gz: Each row represents an individual clinical trial containing 7 column variables. The columns are as follows,

- ID – The identifier for the specific trail
- Sponsor – The sponsor for the trail
- Status – The status of the study at that current time of the file being accessed.
- Start Date – Start date of the trail
- Completion Date – The trail completion date
- Conditions – The list of conditions relating to the trail
- Interventions – The list of interventions explored during the trail

mesh.csv: Each row represents a condition found in the clinical trial list and also its respective hierarchy identifiers in the format [A-Z] [0-9] + (' [A-Z] [0-9] +) * where the initial letter and number combination designates the root of this particular hierarchy and each "." descends a level down the hierarchy. The rows of this file contain condition (term), hierarchy identifier (tree) pairs.

pharma.csv: The file contains a minor number of pharmaceutical breaches from a publicly available list. We are interested in the second column, Parent_Company, which provides the name of the company in issue.

Firstly, all these files are imported into data bricks. This is performed by clicking the burger and selecting the data tab and selecting create table. Once this step is complete, the 3 files are uploaded by clicking the click to browse section and selecting the respective files. After the 3 files are uploaded, they can be located in the file path "dbfs: FileStore/Tables/". Furthermore, before any task are performed on these data sources, a cluster needs to be created. A Databricks cluster is a collection of processing resources and settings which you may use to conduct data engineering, data science, and data analytics workloads (Microsoft, 2022). The cluster is created by going to the burger on Databricks and selecting compute and create cluster where you can name and run it. Once the cluster is operational, a notebook can be started. A notebook is created by again clicking the burger, selecting create and selecting notebook in which you can give it a name, what language you would like to choose and the cluster in which you would like to work on. The languages used for this assignment are Python and SQL.

Once the notebook is open, the files' location is inspected to ensure the files have been uploaded correctly. This is completed using the following code,

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

Out[1]: [FileInfo(path='dbfs:/FileStore/tables/accounts.zip', name='accounts.zip', size=5297592, modificationTime=1647871817000),
FileInfo(path='dbfs:/FileStore/tables/clinicaltrial19/', name='clinicaltrial19/', size=0, modificationTime=0),
FileInfo(path='dbfs:/FileStore/tables/clinicaltrial20/', name='clinicaltrial20/', size=0, modificationTime=0),
FileInfo(path='dbfs:/FileStore/tables/clinicaltrial21/', name='clinicaltrial21/', size=0, modificationTime=0),
FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2019_csv.gz', name='clinicaltrial_2019_csv.gz', size=10060669, modificationTime=1647427233000),
FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020_csv.gz', name='clinicaltrial_2020_csv.gz', size=10981608, modificationTime=1647434232000),
FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2021_csv.gz', name='clinicaltrial_2021_csv.gz', size=11921810, modificationTime=1647442181000),
FileInfo(path='dbfs:/FileStore/tables/mesh/', name='mesh/', size=0, modificationTime=0),
FileInfo(path='dbfs:/FileStore/tables/mesh.csv', name='mesh.csv', size=5295548, modificationTime=1647426157000),
FileInfo(path='dbfs:/FileStore/tables/pharma/', name='pharma/', size=0, modificationTime=0),
FileInfo(path='dbfs:/FileStore/tables/pharma.csv', name='pharma.csv', size=678999, modificationTime=1647425577000)]

Command took 0.21 seconds -- by b.miller2@edu.salford.ac.uk at 20/04/2022, 17:03:23 on Assignment
```

All 3 files have been imported correctly into the FileStore. Next the Clinicaltrial_2021.csv.gz needs to be unzipped. This is completed by first moving the file into the local drive and then unzipping it. The file needs to be relocated into the local since it can't be unzipped in Databricks. The file is then unzipped using a gun zip command in a shell command cell.

```
1 FilePath = "/FileStore/tables/clinicaltrial_2019_csv.gz"
2 FilePath2= "/FileStore/tables/clinicaltrial_2020_csv.gz"
3 FilePath3= "/FileStore/tables/clinicaltrial_2021_csv.gz"
4 dbutils.fs.cp(FilePath, "file:/tmp/")
5 dbutils.fs.cp(FilePath2, "file:/tmp/")
6 dbutils.fs.cp(FilePath3, "file:/tmp/")

Out[3]: True

Command took 1.23 seconds -- by b.miller2@edu.salford.ac.uk at 19/04/2022, 17:15:41 on Assi

Cmd 4

1 %sh
2 gunzip /tmp/clinicaltrial_2019_csv.gz
3 gunzip /tmp/clinicaltrial_2020_csv.gz
4 gunzip /tmp/clinicaltrial_2021_csv.gz

gzip: /tmp/clinicaltrial_2019_csv already exists;      not overwritten
gzip: /tmp/clinicaltrial_2020_csv already exists;      not overwritten
gzip: /tmp/clinicaltrial_2021_csv already exists;      not overwritten

Command took 0.05 seconds -- by b.miller2@edu.salford.ac.uk at 19/04/2022, 17:15:41 on Assi
```

The file will be unzipped after this therefore, the file needs to be moved back into the Databricks FileStore. This is completed however before so, the file is renamed to ensure it in the right format first. This is done by copying the file and renaming it to what is required. The code to rename and move is seen below.

```
1 dbutils.fs.cp("file:/tmp/clinicaltrial_2019_csv", "file:/tmp/clinicaltrial_2019.csv")
2 dbutils.fs.mv("file:/tmp/clinicaltrial_2019.csv", "FileStore/tables/")
3
4 dbutils.fs.cp("file:/tmp/clinicaltrial_2020_csv", "file:/tmp/clinicaltrial_2020.csv")
5 dbutils.fs.mv("file:/tmp/clinicaltrial_2020.csv", "FileStore/tables/")
6
7 dbutils.fs.cp("file:/tmp/clinicaltrial_2021_csv", "file:/tmp/clinicaltrial_2021.csv")
8 dbutils.fs.mv("file:/tmp/clinicaltrial_2021.csv", "FileStore/tables/")

Out[6]: True

Command took 8.15 seconds -- by b.miller2@edu.salford.ac.uk at 19/04/2022, 17:15:41 on Assi
```

Now, we have all files in the correct format and in the correct location. The mesh.csv and pharma.csv file needed no change as they were already in the correct location and format from the beginning however they are given their respective directories when using HIVE SQL to create their respective tables.

Now the preparation of the HIVE SQL method. All files are imported the exact same way as the python method. The same code is used. The clinical trial data is created into a table. When creating a table in HIVE SQL, each column data type must be included, if it's a string or integer for example. Furthermore, the delimiter to separate the data into the correct columns and the location of the data source must be included. The code and to create the table is the following.

```
1 CREATE TABLE clinicaltrial21 (
2 ID STRING,
3 Sponser STRING,
4 Status STRING,
5 StartDate STRING,
6 EndDate STRING,
7 Type STRING,
8 Submission STRING,
9 Conditions STRING,
10 Interventions STRING)
11 ROW FORMAT DELIMITED
12 FIELDS TERMINATED BY '|'
13 LOCATION 'dbfs:/FileStore/tables/clinicaltrial21/';

Error in SQL statement: TableAlreadyExistsException: Table or view 'clinicaltrial21' already exists in database 'default'
Command took 0.09 seconds -- by b.miller2@edu.salford.ac.uk at 26/04/2022, 22:22:03 on abjdba
```

1 SELECT * FROM clinicaltrial21

(1) Spark Jobs

	ID	Sponser	Status	StartDate	EndDate	Type
1	Id	Sponsor	Status	Start	Completion	Type
2	NCT02758028	The University of Hong Kong	Recruiting	Aug 2005	Nov 2021	Interventional
3	NCT02751957	Duke University	Completed	Jul 2016	Jul 2020	Interventional
4	NCT02758483	Universidade Federal do Rio de Janeiro	Completed	Mar 2017	Jan 2018	Interventional
5	NCT02759848	Istanbul Medeniyet University	Completed	Jan 2012	Dec 2014	Observational
6	NCT02758860	University of Roma La Sapienza	Active, not recruiting	Jun 2016	Sep 2020	Observational [Patient I

Truncated results, showing first 1000 rows.

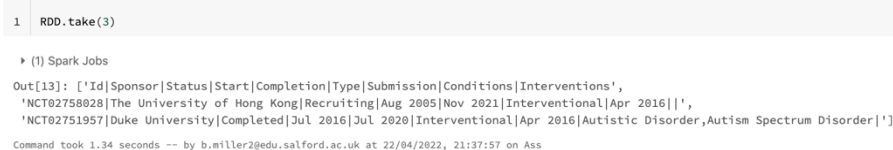
Problem 1:

RDD:

The question for this problem was to find the number of studies in the dataset ensuring to explicitly check distinct studies. This was first completed using PySpark RDDs. This was done by first reading the data into and creating an RDD from our previously imported files in the FileStore using the following code.

```
RDD = sc.textFile("dbfs:/FileStore/tables/clinicaltrial_2021.csv")
```

Firstly, spark is used by using the sc. Then the file was read using textFile, where the text in the brackets is the location of the data. using .take(3) after the created RDD, the first 3 rows of the imported data can be observed.



```
1 RDD.take(3)
```

▶ (1) Spark Jobs

```
Out[13]: ['Id|Sponsor|Status|Start|Completion|Type|Submission|Conditions|Interventions',
'NCT02758028|The University of Hong Kong|Recruiting|Aug 2005|Nov 2021|Interventional|Apr 2016|',
'NCT02751957|Duke University|Completed|Jul 2016|Jul 2020|Interventional|Apr 2016|Autistic Disorder, Autism Spectrum Disorder|']
```

Command took 1.34 seconds -- by b.miller2@edu.salford.ac.uk at 22/04/2022, 21:37:57 on Ass

As we can see from the screenshot above, the first row of the clinical trial data contains names for each of the columns and the other rows contain individual trails. However, it is assumed if we want to calculate the number of trails, it is done by counting the number of rows and therefore, the header must not be included. The header needs to be removed. This is completed using the following code.

```
header = RDD.take(1)[0]
RDD2021 = RDD.filter(lambda x: x != header)
```

The first line of code above takes the header row and returns it as a string. Therefore, seen in the second line of code, this header row can be filtered out. This is done using the filter command. In the bracket, lambda x: x != header is used to filter out the header by going line by line in the RDD. This results in the first 3 rows of the RDD looking like,



```
1 RDD2021.take(3)
```

▶ (1) Spark Jobs

```
Out[16]: ['NCT02758028|The University of Hong Kong|Recruiting|Aug 2005|Nov 2021|Interventional|Apr 2016|',
'NCT02751957|Duke University|Completed|Jul 2016|Jul 2020|Interventional|Apr 2016|Autistic Disorder, Autism Spectrum Disorder|',
'NCT02758483|Universidade Federal do Rio de Janeiro|Completed|Mar 2017|Jan 2018|Interventional|Apr 2016|Diabetes Mellitus|']
```

Command took 2.23 seconds -- by b.miller2@edu.salford.ac.uk at 22/04/2022, 21:49:54 on Ass

The Header has been removed. Next the number of rows and hence the number of trails can be calculated using the following code,

```
RDD2021.count()
```

This code returns 387621 number of studies in the dataset however, the studies have not been checked to be distinct. This is completed using the .distinct() command after the initial RDD once the data has been read. As a result, the final code,

```
1 RDD = sc.textFile("dbfs:/FileStore/tables/clinicaltrial_2021.csv").distinct()
2 header = RDD.take(1)[0]
3 RDD2021 = RDD.filter(lambda x: x != header)
4 RDD2021.count()
```

▶ (2) Spark Jobs

Out[9]: 387261

Command took 6.71 seconds -- by b.miller2@edu.salford.ac.uk at 01/05/2022, 01:05:12 on gvh

This returns 387621 numbers of studies. There are no studies that are the same, they are all unique.

Data Frame:

```
1 DF21 = spark.read.csv("dbfs:/FileStore/tables/clinicaltrial_2021.csv", sep = "|", header=True).distinct()
2 DF21.show()
```

▶ (3) Spark Jobs

	Id	Sponsor	Status	Start	Completion	Type	Submission	Conditions	Interventions
	NCT02757209	Consorzio Futuro ...	Completed	Apr 2016	Jan 2018	Interventional	Apr 2016	Asthma	Fluticasone,Xhanc...
	NCT02751762	Member Companies ...	Recruiting	Nov 2017	Oct 2022	Observational	Apr 2016	Chronic Pain,Subs...	null
	NCT02751957	Duke University	Completed	Jul 2016	Jul 2020	Interventional	Apr 2016	Autistic Disorder...	null
	NCT02758860	University of Rom...	Active, not recru...	Jun 2016	Sep 2020	Observational [Pa...	Apr 2016	Diverticular Dise...	null
	NCT02752113	Institut für Phar...	Completed	Apr 2016	May 2019	Interventional	Apr 2016	Diabetes Mellitus	Metformin,Empagli...
	NCT02755779	Tel Aviv Medical ...	Unknown status	Jun 2016	Jun 2017	Observational	Apr 2016	null	null
	NCT02752438	Ankara University	Unknown status	May 2016	Jul 2017	Observational [Pa...	Apr 2016	Hypoventilation	null
	NCT02752698	The Third Xiangya...	Active, not recru...	Jan 2015	Dec 2021	Interventional	Jun 2015	Appendicitis,Stom...	null
	NCT02756299	Marmara University	Completed	Jun 2014	Apr 2015	Interventional	Apr 2016	Sleep Apnea Syndr...	null
	NCT02759848	Istanbul Medeniye...	Completed	Jan 2012	Dec 2014	Observational	May 2016	Tuberculosis,Lung...	null
	NCT02753530	Orphazyme	Completed	Aug 2017	Jan 2021	Interventional	Apr 2016	Myositis	null
	NCT02754609	James Cook Univer...	Completed	Sep 2016	Oct 2019	Interventional	Apr 2016	Hookworm Infectio...	null
	NCT02755701	Soonchunhyang Uni...	Unknown status	Jul 2016	Dec 2018	Interventional	Apr 2016	Ascites	null
	NCT02753543	Ruijin Hospital	Unknown status	Nov 2015	Nov 2019	Interventional	Apr 2016	Lymphoma	null
	NCT02750956	Bulent Ecevit Uni...	Completed	Jun 2015	Mar 2016	Observational	Apr 2016	Periodontal Diseases	null
	NCT02754817	Novo Nordisk A/S	Completed	Apr 2016	Oct 2016	Observational	Apr 2016	Diabetes Mellitus	Liraglutide,Xultophy
	NCT02758028	The University of...	Recruiting	Aug 2005	Nov 2021	Interventional	Apr 2016	null	null
	NCT02757588	Washington Univer...	Completed	Mar 2016	Jul 2017	Interventional	Apr 2016	null	Vitamins

Command took 21.74 seconds -- by b.miller2@edu.salford.ac.uk at 30/04/2022, 20:33:10 on vgvv

```
1 DF21.count()
```

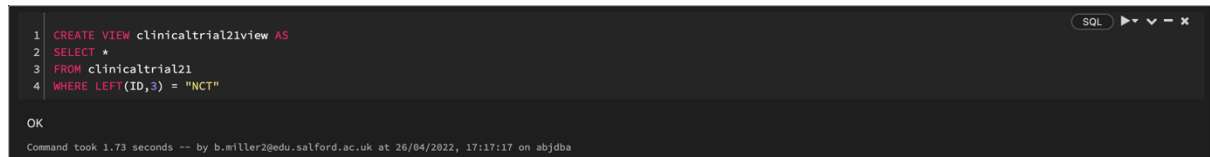
▶ (2) Spark Jobs

Out[8]: 387261

Command took 3.14 seconds -- by b.miller2@edu.salford.ac.uk at 27/04/2022, 10:25:25 on lab10

HIVE SQL:

Firstly, observing the created table seen in the preparation step, the names of the columns are included as the first row of the table. These need to be removed as the table already give the columns the names and if we want to calculate the number of trails which is done by counting the number of rows, the header must not be included. The header is removed by using the following code,



```
1 CREATE VIEW clinicaltrial21view AS
2 SELECT *
3 FROM clinicaltrial21
4 WHERE LEFT(ID,3) = "NCT"
```

OK

Command took 1.73 seconds -- by b.miller2@edu.salford.ac.uk at 26/04/2022, 17:17:17 on abjdba

This creates a new table as a new view based of the original that include all trails with the correct ID codes. Next, the distinct number of studies can be calculated by using the following code on the newly created view,

```
SELECT DISTINCT COUNT(ID) FROM clinicaltrial21view;
```

The above code first starts with SELECT command which decides which columns are selected from the created table. For this problem the COUNT(ID) is selected. Next, The DISTINCT command is used after the SELECT which only output distinct values. COUNT(ID) counts the number of rows. Lastly, FROM is used to specify which table is used which is the clinicaltrial21view. This line of HIVE SQL code returns 387621 studies.

Over all the solutions give a total number of studies of 387621 over the year of 2021. This is over 1000 studies a day. Over 1000 trails are being performed per day.

Problem 2:

RDD:

The question for this problem was to list all the types (as contained in the Type column) of studies in the dataset along with the frequencies of each type. These should be ordered from most frequent to least frequent. It is assumed that all studies contain a type and therefore no filtering is required. This task was first started by splitting the created RDD2021 in the previous problem by the delimiter “|” and then the type of column was separated and paired with the integer 1 for each row split. This pairing creates a dual RDD. The following code and outputted first 3 rows are,

```
SplitRDD = RDD2021.map(lambda s: (s.split("|")[5],1))
```

```
1 SplitRDD = RDD2021.map(lambda s: (s.split("|")[5],1))
2 SplitRDD.take(3)
```

▶ (1) Spark Jobs

```
Out[6]: [('Interventional', 1), ('Interventional', 1), ('Interventional', 1)]
```

Command took 0.56 seconds -- by b.miller2@edu.salford.ac.uk at 24/04/2022, 21:15:13 on msk4

Next, the Split dual RDD is grouped by the Type. The type is the key. The second variable in the RDD increases every time the key appears. The grouping is completed using the reduceByKey function. Lastly the frequencies of the types are ordered from most to least using the sortBy function. The following code and outputted result are,

```
1 SplitRDD = RDD2021.map(lambda s: (s.split("|")[5],1))
2 FreqRDD = SplitRDD.reduceByKey(lambda x,y: x+y)
3 SortedRDD = FreqRDD.sortBy(lambda x: x[1], ascending= False)
4 SortedRDD.collect()
```

▶ (3) Spark Jobs

```
Out[14]: [('Interventional', 301472),
('Observational', 77540),
('Observational [Patient Registry]', 8180),
('Expanded Access', 69)]
```

Command took 1.80 seconds -- by b.miller2@edu.salford.ac.uk at 19/04/2022, 17:15:42 on Assi

Data Frame:

```
1 DF21Type = DF21.select("Type")
2 DF21Type.show()
```

▶ (1) Spark Jobs

Type
Interventional
Interventional
Interventional
Observational
Observational [Pa...]
Interventional
Observational [Pa...]
Interventional
Interventional
Interventional
Observational
Observational
Observational
Interventional
Interventional
Observational
Interventional
Interventional

Command took 0.46 seconds -- by b.miller2@edu.salford.ac.uk at 27/04/2022, 10:54:29 on lab10


```

1 DF21TypeFreq = DF21Type.groupBy(['Type']).count()
2 DF21TypeFreq.sort("count", ascending=False).show()

```

(2) Spark Jobs

Type	count
Interventional	301472
Observational	77540
Observational [Patient Registry]	8180
Expanded Access	69

Command took 4.16 seconds -- by b.miller2@edu.salford.ac.uk at 27/04/2022, 11:13:08 on lab10

HIVE SQL:

In Hive SQL, based on the view created in problem one for the studies, the type of column is selected, and a frequency column is created using the COUNT function which counts the number of times the type appears in that row. The frequency for each row should be 1 since 1 type appears only for each row. Then like the RDD solution, the rows are grouped by the type of key using the GROUP BY function. Lastly, the most types are ordered by the frequency from most to least. The code and the results are as follows,

```

1 SELECT Type, COUNT(Type) AS Freq
2 FROM clinicaltrial21view
3 GROUP BY Type
4 ORDER BY Freq desc;

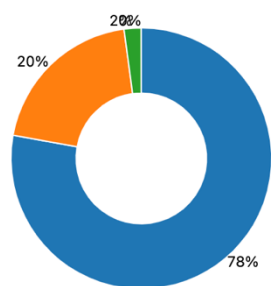
```

(2) Spark Jobs

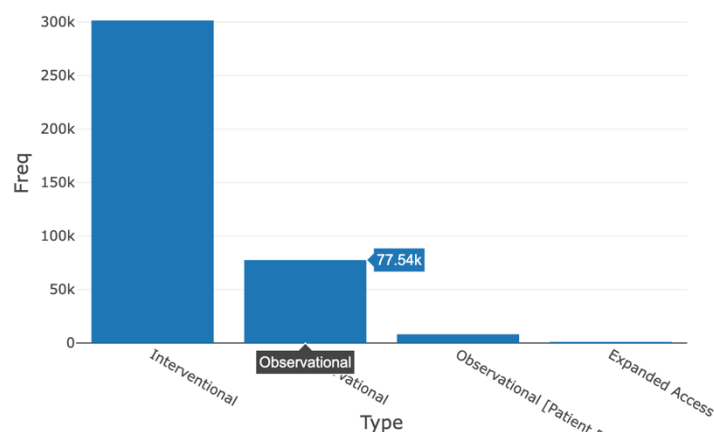
Type	Freq
Interventional	301472
Observational	77540
Observational [Patient Registry]	8180
Expanded Access	69

Showing all 4 rows.

Command took 8.22 seconds -- by b.miller2@edu.salford.ac.uk at 24/04/2022, 21:33:08 on msk4



■ Interventional
■ Observational
■ Observational [Patient Registry]
■ Expanded Access



The solutions produce that the Interventional type is most frequent and Expanded Access is the least. The solutions show that 78% of the studies have the type as interventional and very little, near 0% have the type as Expanded Access. Interventional is the dominant type. Out of all the studies, 301472 have the type interventional. 77540 as observational, 8180 as observational [Patient Registry] and 69 as expanded access. 69 is an extremely small value compared to the over database indicating this type is a very special situation. By filtering the database to observe the conditions of only the expanded access type using RDDs. It is seen that

most of the conditions relating to the trail are life threatening such as different types of cancers, hepatitis and HIV. The code to observe the expanded type and its respective conditions is as follows,

```
1 EA_RDD = RDD2021.map(lambda s: s.split("|")).map(lambda x: (x[5],x[7])).filter(lambda s: s[0] == "Expanded Access")
2 EA_RDD.take(69)
```

- ▶ (2) Spark Jobs

```

Out[19]: [('Expanded Access', 'Arthritis,Emergencies'),
('Expanded Access', ''),
('Expanded Access', 'Hepatitis A,Hepatitis B,Hepatitis,Chronic Disease'),
('Expanded Access', 'Infections,Communicable Diseases'),
('Expanded Access', ''),
('Expanded Access', 'Infectious,Communicable Diseases,Clostridium Infections,Enterocolitis'),
('Expanded Access', 'Renal Insufficiency,Kidney Failure'),
('Expanded Access', 'Hepatic Veno-Occlusive Disease'),
('Expanded Access', 'Breast Neoplasms'),
('Expanded Access', ''),
('Expanded Access', 'Leukemia,Blast Crisis'),
('Expanded Access', 'Neuroendocrine Tumors'),
('Expanded Access', 'Epilepsy'),
('Expanded Access', 'Kidney Diseases,Renal Insufficiency,Hypertension'),
('Expanded Access', 'Carcinoma'),
('Expanded Access', ''),
('Expanded Access', 'Lambert-Eaton Myasthenic Syndrome,Syndrome'),
('Expanded Access', 'Lambert-Eaton Myasthenic Syndrome,Syndrome'),
('Expanded Access', 'Neoplasms'),
('Expanded Access', 'Osteoporosis'),
('Expanded Access', 'Leukemia')]

```

Problem 3: RDD:

The question for this problem was to find the top 5 conditions (from Conditions) with their frequencies. It is assumed that individuals can have multiple conditions and also that a condition row that looks like ' ' suggest there is zero conditions for that trail and can be removed. This was started by first splitting the studies by the same delimiter used in the previous problem in "|" and separating the 7th columns, the conditions column. This created each of conditions for the studies into individual rows however, each condition regardless of the study needs to be in its own individual row. This is done by using the flatMap with an additional split. flatMap is different to map since it explodes each row and separates each element split. After this, the resulting RDD is filtered using the filter function removing any row containing no conditions. Lastly like the previous problem, the conditions are paired with the integer 1 and grouped to find the 5 most frequent conditions in the studies. The code and results are the following,

```
1 SplitRDD = RDD2021.map(lambda s: s.split("|")[7]).flatMap(lambda s: s.split(","))
2 FiltRDD = SplitRDD.filter(lambda x: x != " ").map(lambda x: (x,1))
3 FreqRDD = FiltRDD.reduceByKey(lambda x,y: x+y)
4 SortedRDD2021 = FreqRDD.sortBy(lambda x: x[1], ascending=False)
5 SortedRDD2021.take(5)
```

Python ▶ ▾ ▹ ▸ ✕

▶ (3) Spark Jobs

Out[8]: [('Carcinoma', 13389),
(('Diabetes Mellitus', 11080),
(('Neoplasms', 9371),
(('Breast Neoplasms', 8640),
(('Syndrome', 8032)]

Command took 3.48 seconds -- by b.miller2@edu.salford.ac.uk at 24/04/2022, 22:03:12 on msk4

DATAFRAMES:

```
1 from pyspark.sql.functions import split, explode
2 DF21Con = DF21.select("Conditions").withColumn("Conditions", explode(split("Conditions", ",")))
3 DF21ConFreq = DF21Con.groupBy("Conditions").count()
4 DF21ConFreq.sort("count", ascending=False).show(5)
```

Python ▶ ▾ ▹ ▸ ✕

▶ (3) Spark Jobs

Conditions	count
Carcinoma	13389
Diabetes Mellitus	11080
Neoplasms	9371
Breast Neoplasms	8640
Syndrome	8032

only showing top 5 rows

Command took 8.97 seconds -- by b.miller2@edu.salford.ac.uk at 11/05/2022, 23:08:24 on dc

HIVE SQL:

For this problem in HIVE SQL, a new table is created from the study data with the columns, Split_Conditions and a frequency column named as Freq. The Split_Conditions column is created by splitting the original conditions and then separating each condition the study contains if there are multiple using the explode and lateral view functions. Next the Split_Conditions column is filtered to remove any rows containing no conditions. Lastly, the conditions are grouped like the previous problem and ordered from most frequent to least. The code and results are the following,

SQL ▶ ▼ - ✕

```
1 Create Table Table4 As (  
2 Select Split_Conditions, Count(Split_Conditions) as Freq  
3 FROM clinicaltrial21 lateral view explode(split(Conditions, ',')) Conditions AS Split_Conditions  
4 WHERE Split_Conditions != ""  
5 Group BY Split_Conditions  
6 Order BY Freq Desc  
7 );
```

▶ (7) Spark Jobs

Query returned no results

Command took 16.38 seconds -- by b.miller2@edu.salford.ac.uk at 24/04/2022, 22:29:49 on msk4

Cmd 20

```
1 select * from table4
```

▶ (1) Spark Jobs

	Split_Conditions ▲	Freq ▲
1	Carcinoma	13389
2	Diabetes Mellitus	11080
3	Neoplasms	9371
4	Breast Neoplasms	8640
5	Syndrome	8032
6	Leukemia	5904

Truncated results, showing first 1000 rows.
[Click to re-execute with maximum result limits.](#)

All solution produce that Carcinoma and Diabetes Mellitus are the two most frequent condition found in the clinical trials. Furthermore, using the count function in HIVE SQL and Pyspark it found there are a total of 3437 different conditions. The least frequent condition is Pelvic Neoplasms with a frequency of 71.

Problem 4:

RDD:

The client wishes to know the 5 most frequent roots. Each condition can be mapped to one or more hierarchy codes. These hierarchy codes can be found in the mesh.csv data source for each of the conditions. First, the mesh data is created into a RDD by using the `textFile` function. The following mesh RDD looks like the following,

```
1 MeshRDD = sc.textFile("dbfs:/FileStore/tables/mesh.csv")
2 MeshRDD.take(10)

▶ (1) Spark Jobs
Out[28]: ['term,tree',
'Calcimycin,D03.633.100.221.173',
'A-23187,D03.633.100.221.173',
'Temefos,D02.705.400.625.800',
'Temefos,D02.705.539.345.800',
'Temefos,D02.886.300.692.800',
'Abate,D02.705.400.625.800',
'Abate,D02.705.539.345.800',
'Abate,D02.886.300.692.800',
'Difos,D02.705.400.625.800']
Command took 0.57 seconds -- by b.miller2@edu.salford.ac.uk at 25/04/2022, 14:18:06 on xbczxc
```

The condition and its matching hierarchy code. We want to extract and separate these two elements. This is done by making a pair RDD with the first element being the condition and the second element the root. This is done by using the `split` and then `map` function to rearrange. Furthermore, the whole of the hierarchy code is not required and only the first 3 sequence of letters. This again is done with a `split` and rearrangement using the `map` function. Lastly, like the previous problem, the condition and root are paired with the integer 1 and a `reduceByKey` is used to see how many time the condition and respective code appears in the Mesh Data. The code and result are seen in the following,

```
1 MeshRDD = sc.textFile("dbfs:/FileStore/tables/mesh.csv")
2 MeshRDDF = MeshRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1]))
3 MeshRDDS = MeshRDDF.map(lambda x: (x[0],x[1].split("."))).map(lambda x: ((x[0],x[1][0]),1)).reduceByKey(lambda x,y: x+y)
4 MeshRDDS.take(10)

▶ (1) Spark Jobs
Out[40]: [(['term', 'tree'), 1],
(['Calcimycin', 'D03'], 1),
(['A-23187', 'D03'], 1),
(['Abate', 'D02'], 3),
(['Difos', 'D02'], 3),
(['Abattoirs', 'J01'], 1),
(['Abbreviations as Topic', 'L01'], 1),
(['Abdomen', 'A01'], 1),
(['Abdominal Neoplasms', 'C04'], 1),
(['Cremaster Muscle', 'A02'], 1)]
Command took 1.64 seconds -- by b.miller2@edu.salford.ac.uk at 25/04/2022, 14:36:13 on xbczxc
```

Now we have the condition, root and frequency that appears in the mesh data, it can be joined to original clinical trial data by matching by condition. This is done as the question required to find the most frequent roots. However, before the joining, the mesh RDD must be transformed so it can be joined by condition (the key).

```

1 MeshRDD = sc.textFile("dbfs:/FileStore/tables/mesh.csv")
2 MeshRDDF = MeshRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1]))
3 MeshRDDS = MeshRDDF.map(lambda x: (x[0],x[1].split("."))).map(lambda x: ((x[0],x[1][0]),1)).reduceByKey(lambda x,y: x+y)
4 MeshRDDP = MeshRDDS.map(lambda x: (x[0][0],x[0][1], x[1]))
5 MeshRDDP.take(10)

```

▶ (1) Spark Jobs

```

Out[41]: [('term', ('tree', 1)),
('Calcimycin', ('D03', 1)),
('A-23187', ('D03', 1)),
('Abate', ('D02', 3)),
('Difos', ('D02', 3)),
('Abattoirs', ('J01', 1)),
('Abbreviations as Topic', ('L01', 1)),
('Abdomen', ('A01', 1)),
('Abdominal Neoplasms', ('C04', 1)),
('Cremaster Muscle', ('A02', 1))]

```

Command took 1.64 seconds -- by b.miller2@edu.salford.ac.uk at 25/04/2022, 14:49:56 on xbczxc

After this, joining the two RDDs by using the join function and multiplying the frequency found in problem 3 for the conditions and then grouping, the most frequent root can be discovered. The code and following results are the following,

```

1 MeshRDD = sc.textFile("dbfs:/FileStore/tables/mesh.csv")
2 MeshRDDF = MeshRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1]))
3 MeshRDDS = MeshRDDF.map(lambda x: (x[0],x[1].split("."))).map(lambda x: ((x[0],x[1][0]),1)).reduceByKey(lambda x,y: x+y)
4 MeshRDDP = MeshRDDS.map(lambda x: (x[0][0],x[0][1], x[1]))
5 RDDJoin = SortedRDD2021.join(MeshRDDP).map(lambda x: (x[0],x[1][1][0],x[1][0],x[1][1][1]))
6 RDDJ1 = RDDJoin.map(lambda x: (x[1], x[2]*x[3])).reduceByKey(lambda x,y:x*y).sortBy(lambda x: x[1], ascending= False)
7 RDDJ1.take(5)

```

▶ (3) Spark Jobs

```

Out[35]: [('C04', 143994),
('C23', 136079),
('C01', 106674),
('C14', 94523),
('C10', 92310)]

```

Command took 3.19 seconds -- by b.miller2@edu.salford.ac.uk at 25/04/2022, 14:20:51 on xbczxc

HIVE SQL:

Like the clinical trial data, a table must be created for the mesh data. Furthermore, the delimiter to separate the data into the correct columns and the location of the data source must be included. To create this table, the code looks and creates the table like the following,

```

1 CREATE TABLE Mesh(
2 Condition STRING,
3 hierarchy_identifiers STRING)
4 ROW FORMAT DELIMITED
5 FIELDS TERMINATED BY ','
6 LOCATION 'dbfs:/FileStore/tables/mesh';

```

OK

Command took 0.34 seconds -- by b.miller2@edu.salford.ac.uk at 20/04/2022, 13:20:09 on Assignment

Next based on this table a view is created to extract only the part of the root that is needed, the first 3 letters. The following code and view looks like,

```

CREATE VIEW MeshIdentifiers AS (
SELECT Condition, LEFT(hierarchy_identifiers,3) AS H_I FROM Mesh);

```

1 `select * from MeshIdentifiers`

(1) Spark Jobs

	Condition	H_I
1	term	tre
2	Calcimycin	D03
3	A-23187	D03
4	Temefos	D02
5	Temefos	D02
6	Temefos	D02

Truncated results, showing first 1000 rows.

Command took 1.04 seconds -- by b.miller2@edu.salford.ac.uk at 20/04/2022, 13:20:19 on Assignment

After this view is created, another table is made, which merges our studies to the mesh code. The conditions, root and frequency of the conditions appearing in the study are all merged into one table. This is done by matching the Condition column found in our view created above and the Split_conditions column created in problem 3. The code to do this and resulting table is the following,

1 `Create table Con_HI_Freq2 AS`
 2 `SELECT Condition, H_I, Freq`
 3 `FROM table4, MeshIdentifiers`
 4 `WHERE Split_Conditions = Condition;`

(4) Spark Jobs

Query returned no results

Command took 8.42 seconds -- by b.miller2@edu.salford.ac.uk at 20/04/2022, 13:20:53 on Assignment

1 `Select * From Con_HI_Freq2`

(1) Spark Jobs

	Condition	H_I	Freq
1	COVID-19	C08	5567
2	COVID-19	C08	5567
3	COVID-19	C08	5567
4	COVID-19	C01	5567
5	COVID-19	C01	5567
6	COVID-19	C01	5567

Truncated results, showing first 1000 rows.

Command took 1.14 seconds -- by b.miller2@edu.salford.ac.uk at 20/04/2022, 13:21:01 on Assignment

Lastly, the rows are all grouped by the root (H_I). This is done by the GROUP BY function in HIVE SQL and then ordered from most frequent to least. The code and results are the following,

1 `Select H_I, Sum(Freq) as amount from Con_HI_Freq2`
 2 `group by H_I`
 3 `order by amount desc;`

(2) Spark Jobs

	H_I	amount
1	C04	143994
2	C23	136079
3	C01	106674
4	C14	94523
5	C10	92310
6	C06	85646

Showing all 63 rows.

Command took 1.27 seconds -- by b.miller2@edu.salford.ac.uk at 20/04/2022, 13:21:10 on Assignment

The top 5 most frequent root s found in the studies are C04, C23, C01, C14 and C10. These top frequent, are the same seen in the all the solutions created. C04 is the most

frequent root found in the studies being associated 143994 times. These 5 roots appear in the studies a total of 573580 times which is a 48% of all recorded roots.

Problem 5: RDD:

For this question, the 10 most common sponsors that are not pharmaceutical companies, along with the number of clinical trials they have sponsored is to be found. Like the previous question for the mesh data, another the Pharma.csv is read into a RDD to access the Pharmaceutical Company data. This is required as it is assumed that the Parent Company column contains all possible pharmaceutical companies therefore this will be filtered out of the clinical trial data. The code to create the new RDD is as follows,

```
1 PharmaRDD = sc.textFile("dbfs:/FileStore/tables/pharma.csv")
2 PharmaRDD1 = PharmaRDD.map(lambda x: x.split(",")).map(lambda x: x[1])
3 headerPharma = PharmaRDD1.take(1)[0]
4 PhRDD = PharmaRDD1.filter(lambda x: x != headerPharma)

(1) Spark Jobs
Command took 9.79 seconds -- by b.miller2@edu.salford.ac.uk at 26/04/2022, 16:54:16 on abjdba
```

This code creates a RDD which contains only the Parent company column and also removes the header. Lastly, collecting just the sponsors from the clinical trial data for all the studies and finding their respective frequency for how many times they appear in the studies, it can be filtered out using the FILTER function to remove any sponsors that are a pharmaceutical company found in PhRDD. The code to create and find the 10 most common sponsors is as follows. However, before the filter can be performed, the PhRDD must be created into a list of strings as a RDD can't be filtered by another RDD. This is simply done by collecting all elements from the RDD using the .collect() function. The 10 most common sponsors are,

```
1 RDD21 = RDD2021.map(lambda s: (s.split("|")[1],1)).reduceByKey(lambda x,y: x+y).sortBy(lambda x: x[1], ascending= False)
2
3 PhRDDlist = PhRDD.collect()
4
5 RDD_Sponsor21 = RDD21.filter(lambda s: s[0] not in PhRDDlist)
6 RDD_Sponsor21.take(10)

(4) Spark Jobs
Out[4]: [('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 4.67 seconds -- by b.miller2@edu.salford.ac.uk at 26/04/2022, 16:55:09 on abjdba
```

HIVE SQL:

Like the clinical trial data, a table must be created for the Pharma data. Furthermore, the delimiter to separate the data into the correct columns and the location of the data source must be included. To create this table, the code looks and creates the table like the following,

```
1 CREATE TABLE pharma(  
2 Company string,  
3 Parent_Company string)  
4 ROW FORMAT DELIMITED  
5 FIELDS TERMINATED BY ','  
6 LOCATION 'dbfs:/FileStore/tables/pharma';  
  
OK  
Command took 0.68 seconds -- by b.miller2@edu.salford.ac.uk at 26/04/2022, 17:17:32 on abjdba
```

Only the first two columns of the data are extracted and inputted in the table as only the Parent_Company column is required. Next a view is created to remove the names of the columns as they appear as rows. After this, using the SELECT command seen in previous problems and newly used NOT IN function which filters out rows specified. The code and results are as the following,

```
1 create view PharmaCom AS (  
2 select * from pharma  
3 where Company != "Company");  
  
OK  
Command took 0.98 seconds -- by b.miller2@edu.salford.ac.uk at 26/04/2022, 17:17:36 on abjdba
```

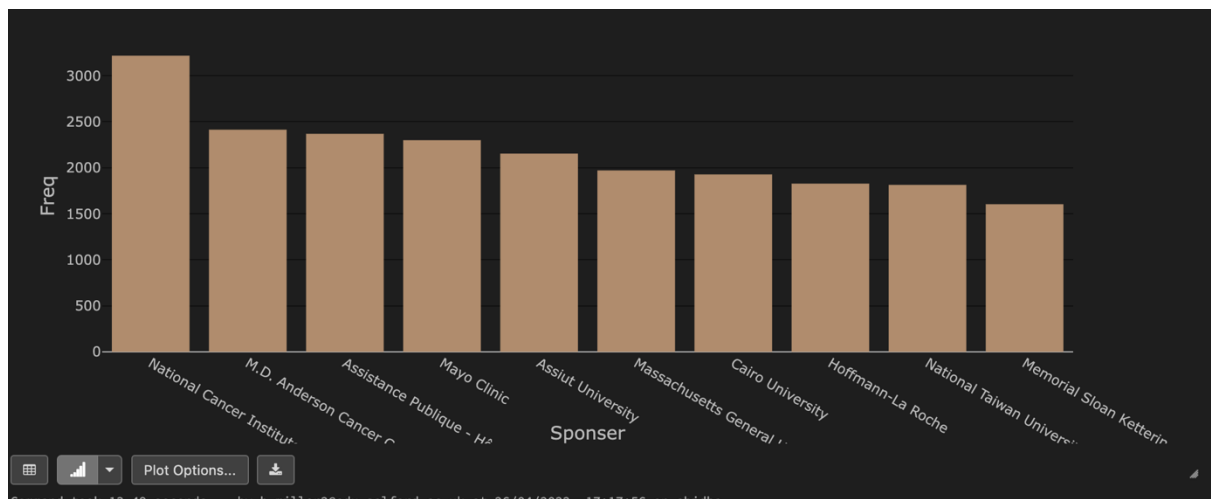
Cmd 31

```
1 SELECT Sponser, COUNT(Sponser) AS Freq  
2 FROM clinicaltrial21view  
3 WHERE Sponser  
4 NOT IN (SELECT Parent_Company  
5 FROM PharmaCom)  
6 GROUP BY Sponser  
7 SORT BY Freq DESC  
8 LIMIT 10;
```

▶ (2) Spark Jobs

	Sponser	Freq
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	Assiut University	2154
6	Massachusetts General Hospital	1971
7	Cairo University	1928
8	Hoffmann-La Roche	1828
9	National Taiwan University Hospital	1814
10	Memorial Sloan Kettering Cancer Center	1604

From all the solutions, these are the top 10 most common sponsors with the National Cancer institute is the most frequent sponsor with a frequency of 3218.



Problem 6:

RDD:

For this problem, the task was to Plot the number of completed studies each month in the year 2021. In RDDs, the clinical data is first split to get the status and completion columns and then filtered. The status column is filtered to only return when the status is Completed, and the Completion column is filtered to only return when the date of completion is in the year of 2021. Therefore, the only studies that remain in the RDD are studies that are completed and in the year of 2021. Furthermore, the studies are then grouped by month to return the number of studies that are completed in each month.

```
1 Completed_RDD2021 = RDD2021.map(lambda s: (s.split("|"))).map(lambda f: (f[2], f[4])).filter(lambda f: "2021" in f[1] and "Completed" in f[0]).map(lambda x: (x[1],1)).reduceByKey(lambda x,y: x+y).sortBy(lambda x: x[1], ascending=False)
2 Completed_RDD2021.collect()

(3) Spark Jobs
Out[68]: [('Mar 2021', 1227),
('Jan 2021', 1131),
('Jun 2021', 1094),
('May 2021', 984),
('Apr 2021', 967),
('Feb 2021', 934),
('Jul 2021', 819),
('Aug 2021', 780),
('Sep 2021', 528),
('Oct 2021', 187)]

Command took 2.96 seconds -- by b.miller2@edu.salford.ac.uk at 26/04/2022, 21:06:29 on abjdba
```

To start the visualization, firstly the above RDD is created to a dataframe then to a panda dataframe where the package matplotlib can be used to create and customise the plot. In addition, the index of the dataframe is reordered so the months are in order. This is done by using a toDF(), toPandas() and reindex() functions respectively. Next the plot() function is used on the panda dataframe to create the plot. A bar plot is used. In addition, the code and resulting plots are the following,

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 Completed_RDD2021DF = Completed_RDD2021.toDF().toPandas().reindex([1,5,0,4,3,2,6,7,8,9])
6 first_column = Completed_RDD2021DF.iloc[:, 0].astype('string')
7
8 Completed_RDD2021DF.plot(kind='bar', colormap='BrBG').set_xticklabels(Completed_RDD2021DF._1)
9 plt.xticks(rotation=30, horizontalalignment="center")
10 plt.title("Completed Studies Each Month In 2021")
11 plt.ylabel("Completed Studies")
12 plt.xlabel("Month")

(2) Spark Jobs
Out[69]: Text(0.5, 0, 'Month')

Completed Studies Each Month In 2021
1200
1000
800
600
400
200
0
Jan 2021 Feb 2021 Mar 2021 Apr 2021 May 2021 Jun 2021 Jul 2021 Aug 2021 Sep 2021 Oct 2021
Month

Command took 0.79 seconds -- by b.miller2@edu.salford.ac.uk at 26/04/2022, 21:29:53 on abjdba
```

HIVE SQL:

In Hive SQL, a view is created to find the frequency of completed studies for each month in the year of 2021. The view is created off the clinicaltrial21 table created and extracts the EndDate. This view is filtered to only include completed statuses and studies completed in the year of 2021. The code and resulting table are as follows,

```
1 Create view CompletedDate as
2 select LEFT(EndDate, 3) as Month, COUNT(EndDate) as Freq
3 From clinicaltrial21
4 where Status = "Completed" and EndDate LIKE '%2021%'
5 Group by EndDate
6 ORDER BY Freq desc;
```

OK

Command took 0.16 seconds -- by b.miller2@edu.salford.ac.uk at 26/04/2022, 22:46:00 on abjdba

Cmd 34

```
1 Select * from CompletedDate
```

(2) Spark Jobs

	Month	Freq
1	Mar	1227
2	Jan	1131
3	Jun	1094
4	May	984
5	Apr	967
6	Feb	934
7	Jul	819
8	Aug	700
9	Sep	528
10	Oct	187

Showing all 10 rows.

The most frequent month for completed studies is in March then January and then June with a respective 1227, 1131 and 1094 number of completed studies. The month of October has the least number of completed studies at 187. The months November and December do not appear in the studies for the year of 2021 suggesting the months have zero completed in them. All the trails are completed before the months of November. Looking at the graph created in the RDD method shows in the latter months, the completed studies start to decrease. Most studies are completed from the months January to June. Having a look at the at the different type of statuses some of the trails can have, the percentage that a trail is either terminated, suspended, withdrawn or no longer available is 8.75%. Therefore 8.75% of trails are not completed. Furthermore, plotting the trails that are recruiting suggest that most of these trails occurs in December. This is complimentary to the completed results since as all the trails are completed in the year, recruiting for more trails begin which indicates the spike in December.



References:

Microsoft. (2022, April 13). *Clusters*. Retrieved from Microsoft:

[https://docs.microsoft.com/en-](https://docs.microsoft.com/en-us/azure/databricks/clusters/#:~:text=An%20Azure%20Databricks%20cluster%20is,hoc%20analytics%2C%20and%20machine%20learning.)

[us/azure/databricks/clusters/#:~:text=An%20Azure%20Databricks%20cluster%20is,hoc%20analytics%2C%20and%20machine%20learning.](https://docs.microsoft.com/en-us/azure/databricks/clusters/#:~:text=An%20Azure%20Databricks%20cluster%20is,hoc%20analytics%2C%20and%20machine%20learning.)