NAME: AMUDA LATEEFAT

STUDENT ID: @00651625

Description of any setup required

The clinical trial, mesh and pharma files should be uploaded to the **tables** folder in the Databricks **FileStore**. Then **Scripts** Notebook should be executed first, to load the files.

By default, the clinical trial year has been set to 2021, you can change this in the Scripts Notebook before running it. However, for HiveQL implementation, you would still need to change/specify the clinical trial year in the first cell of the HiveQL Notebook. Please ensure that the year matches what was specified in the Scripts Notebook, to avoid getting wrong results.

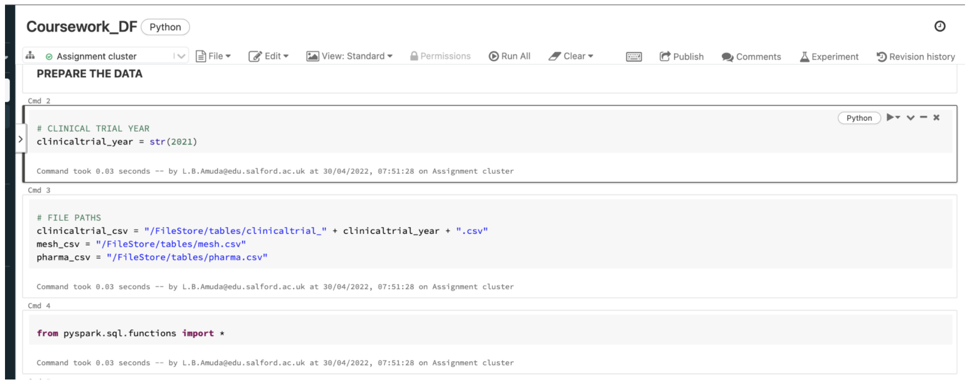
Data cleaning and preparation

In all 3 implementations, the first cell holds the clinical trial year variable, which has been set to 2021 by default and can be changed if needed.

- **PySpark DataFrame**

Firstly, Pyspark functions were imported for the entire notebook. Using the csv method of the dataframe reader(*spark.read*), the clinical trial dataframe was created from the csv file. The *delimiter* option was used to specify that the pipe delimiter is used, the *header* option was used to specify that the file contained a header and should be used as the dataframe header and *inferSchema* option specified that the schema should be inferred based on the data in the dataset.

The mesh and pharma dataframes were created in the same way, from the mesh csv and pharma csv files, respectively.



PREPARE THE DATA

Cmd 2

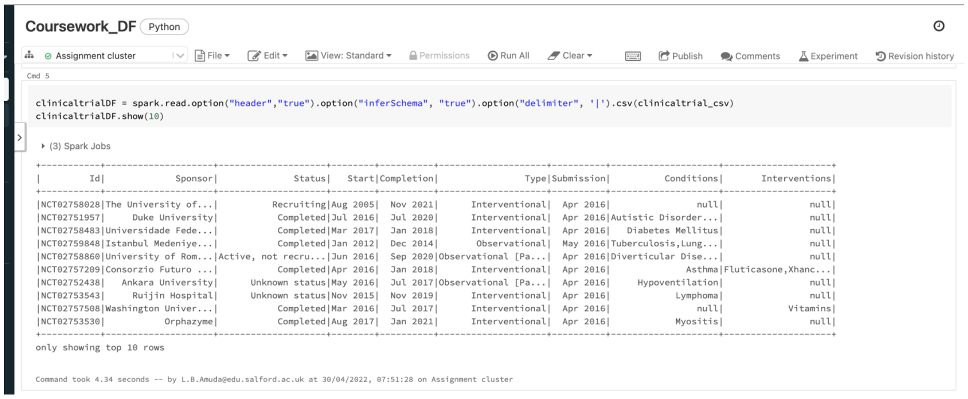
```
# CLINICAL TRIAL YEAR
clinicaltrial_year = str(2021)

Command took 0.03 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster
```

Cmd 3

```
# FILE PATHS
clinicaltrial_csv = "/FileStore/tables/clinicaltrial_" + clinicaltrial_year + ".csv"
mesh_csv = "/FileStore/tables/mesh.csv"
pharma_csv = "/FileStore/tables/pharma.csv"

Command took 0.03 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster
```

Cmd 4

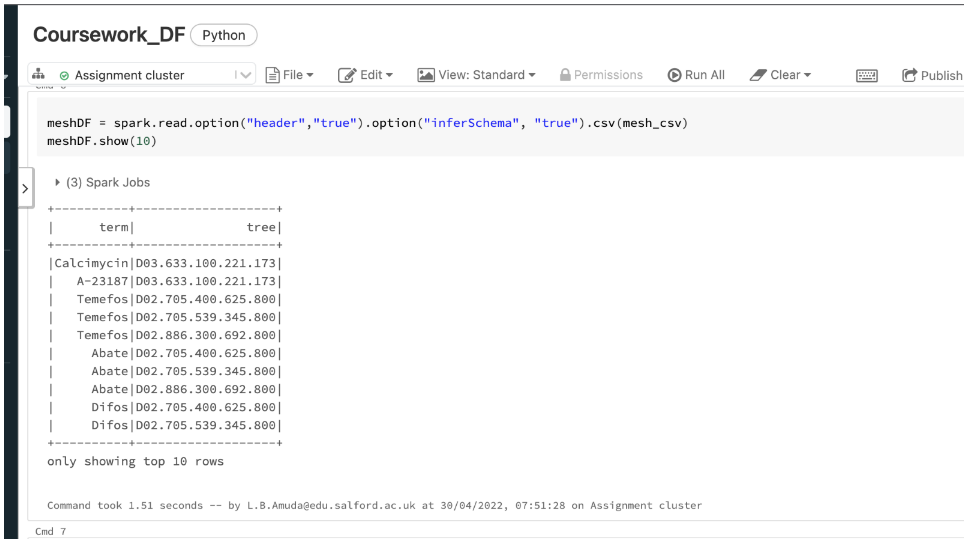
```
from pyspark.sql.functions import *

Command took 0.03 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster
```



```
clinicaltrialDF = spark.read.option("header","true").option("inferSchema", "true").option("delimiter", '|').csv(clinicaltrial_csv)
clinicaltrialDF.show(10)
```

```
+-----------+--------------------+--------------------+--------+----------+--------------------+----------+--------------------+--------------------+
|         Id|             Sponsor|              Status|   Start|Completion|                Type|Submission|          Conditions|       Interventions|
+-----------+--------------------+--------------------+--------+----------+--------------------+----------+--------------------+--------------------+
|NCT02758028|The University of...|          Recruiting|Aug 2005|  Nov 2021|      Interventional|  Apr 2016|                null|                null|
|NCT02751957|     Duke University|           Completed|Jul 2016|  Jul 2020|      Interventional|  Apr 2016|Autistic Disorder...|                null|
|NCT02758483|Universidade Fede...|           Completed|Mar 2017|  Jan 2018|      Interventional|  Apr 2016|   Diabetes Mellitus|                null|
|NCT02759848|Istanbul Medeniye...|           Completed|Jan 2012|  Dec 2014|       Observational|  May 2016|Tuberculosis,Lung...|                null|
|NCT02758860|University of Rom...|Active, not recru...|Jun 2016|  Sep 2020|Observational [Pa...|  Apr 2016|Diverticular Dise...|                null|
|NCT02757209|Consorzio Futuro ...|           Completed|Apr 2016|  Jan 2018|      Interventional|  Apr 2016|              Asthma|Fluticasone,Xhanc...|
|NCT02752438|   Ankara University|      Unknown status|May 2016|  Jul 2017|Observational [Pa...|  Apr 2016|     Hypoventilation|                null|
|NCT02753543|     Ruijin Hospital|      Unknown status|Nov 2015|  Nov 2019|      Interventional|  Apr 2016|            Lymphoma|                null|
|NCT02757508|Washington Univer...|           Completed|Mar 2016|  Jul 2017|      Interventional|  Apr 2016|                null|            Vitamins|
|NCT02753530|           Orphazyme|           Completed|Aug 2017|  Jan 2021|      Interventional|  Apr 2016|            Myositis|                null|
+-----------+--------------------+--------------------+--------+----------+--------------------+----------+--------------------+--------------------+
only showing top 10 rows
```

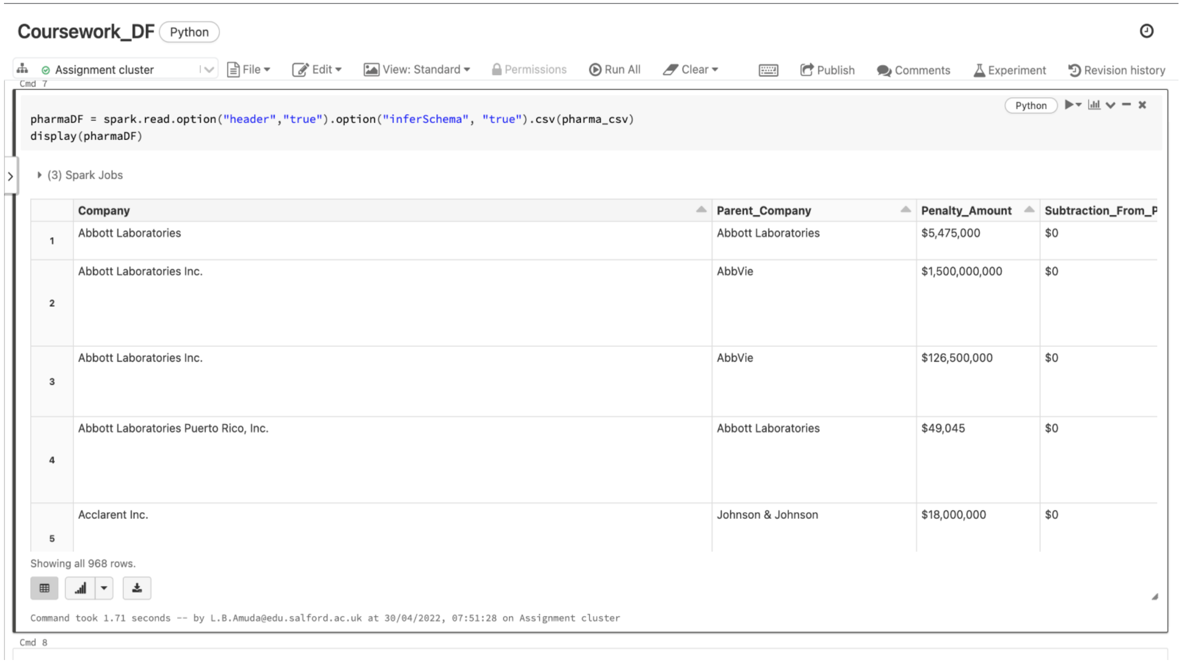
```
Command took 4.34 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster
```



```
meshDF = spark.read.option("header","true").option("inferSchema", "true").csv(mesh_csv)
meshDF.show(10)
```

```
+----------+-------------------+
|      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|
|     Difos|D02.705.539.345.800|
+----------+-------------------+
only showing top 10 rows
```

```
Command took 1.51 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster
```


Coursework_DF Python

```
pharmaDF = spark.read.option("header","true").option("inferSchema", "true").csv(pharma_csv)
display(pharmaDF)
```

(3) Spark Jobs

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

Showing all 968 rows.

```
Command took 1.71 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster
```

- **HiveQL**

Existing tables' folders were removed from the hive location, and external tables and views were dropped/removed as well - **if they exist**. This helps to clean and prevent data from being *cached* in the table or hive location, for cases where the notebook commands would be executed multiple times for different years.

After removing existing tables, the clinical trial table was created based on its schema and the location for storing the table was specified. Then a query was written to load the data from the filepath into the created table, and the **overwrite** keyword was used so as to replace the existing data (if any) in the table, otherwise, the data will be appended to the table every time the command is executed. This was repeated for the mesh and pharma datasets, to create mesh and pharma tables, respectively.

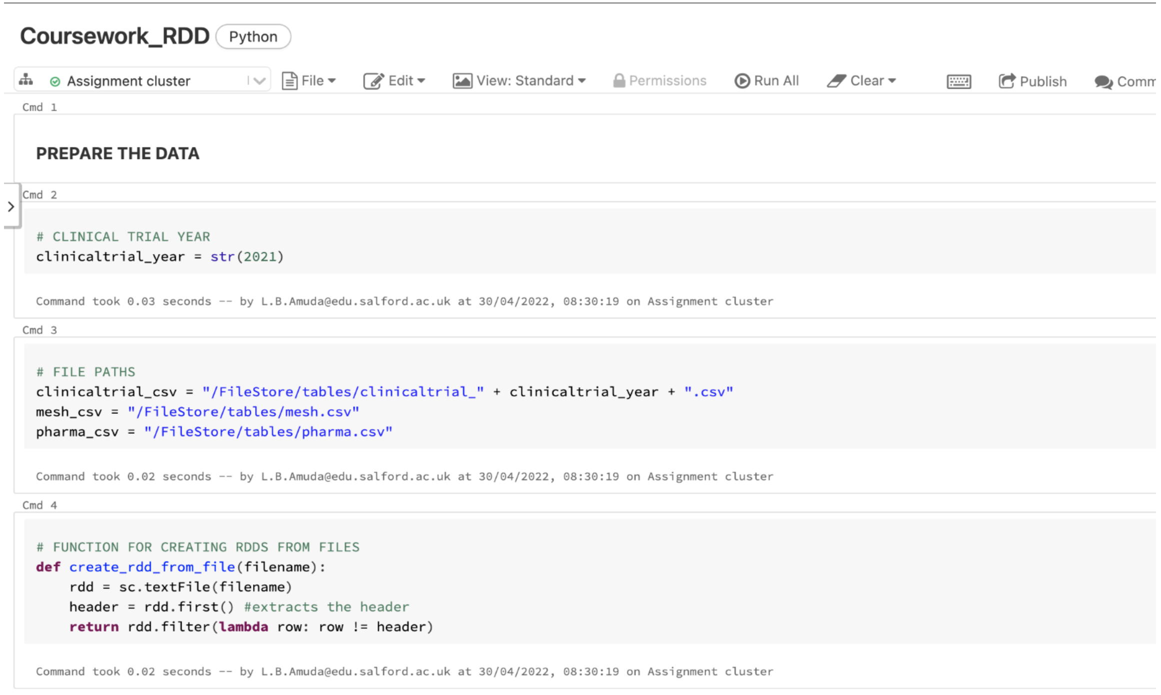
However, another table was created for the pharma table, named *pharma_formatted_table*, which contained only the two columns needed (Company and Parent_Company columns) and the data was also formatted using the Regex Replace function, to remove the double quotation marks that were appended to the column values.

From the clinical trial table created, a temporary view was created and a Where clause was used to remove the header that was included from the csv file when loading the data. This was also repeated for the mesh table and the pharma formatted table, to generate mesh view and pharma view, respectively.

- **PySpark RDD**

A function (named **create_rdd_from_file**) was defined to create an RDD from a file for reusability. This function creates an RDD from a file, using the spark context's *textFile* method, then a variable is created for the first row in the RDD which is the header from the csv file, this header is then filtered out from the RDD, and the result is returned.

This *create_rdd_from_file* function was used to create the *clinical trial* and *mesh* RDDs. However, this function could not be used for the pharma dataset as removing the double quotation from the dataset proved problematic. Hence, the dataframe reader(*spark.read*) and csv method were used to create a dataframe from the pharma csv file and then the *.rdd* method was used to convert it to an RDD.



Cmd 1

PREPARE THE DATA

Cmd 2

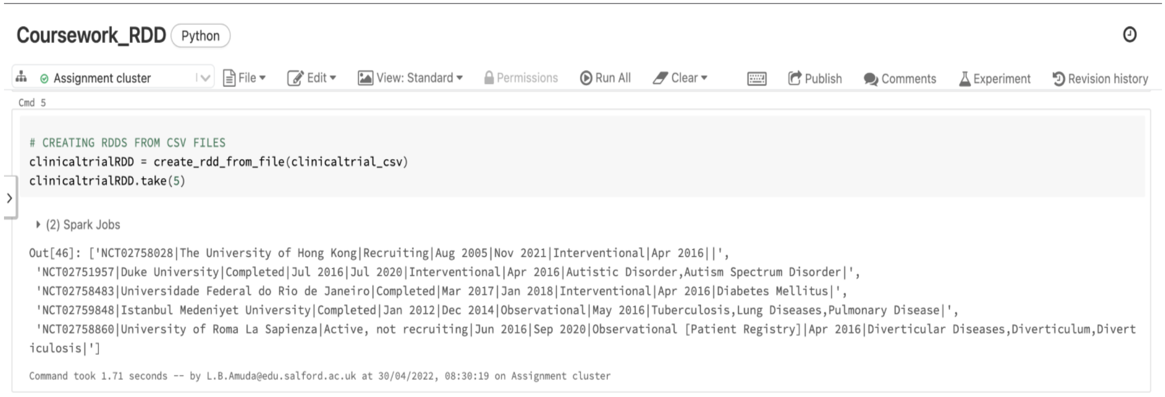
```
# CLINICAL TRIAL YEAR
clinicaltrial_year = str(2021)

Command took 0.03 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:19 on Assignment cluster
```

Cmd 3

```
# FILE PATHS
clinicaltrial_csv = "/FileStore/tables/clinicaltrial_" + clinicaltrial_year + ".csv"
mesh_csv = "/FileStore/tables/mesh.csv"
pharma_csv = "/FileStore/tables/pharma.csv"

Command took 0.02 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:19 on Assignment cluster
```

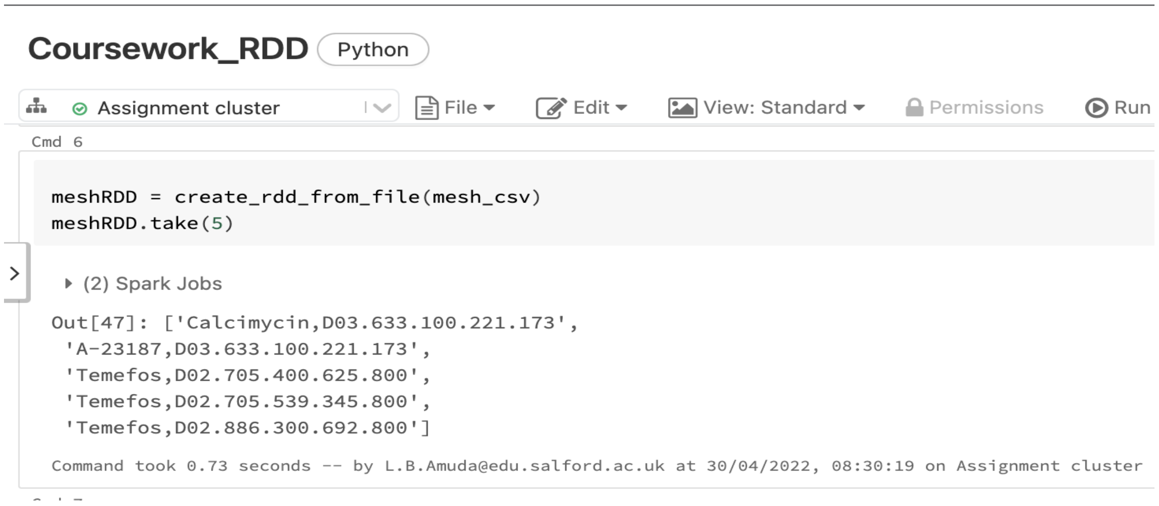
Cmd 4

```
# FUNCTION FOR CREATING RDDS FROM FILES
def create_rdd_from_file(filename):
    rdd = sc.textFile(filename)
    header = rdd.first() #extracts the header
    return rdd.filter(lambda row: row != header)

Command took 0.02 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:19 on Assignment cluster
```



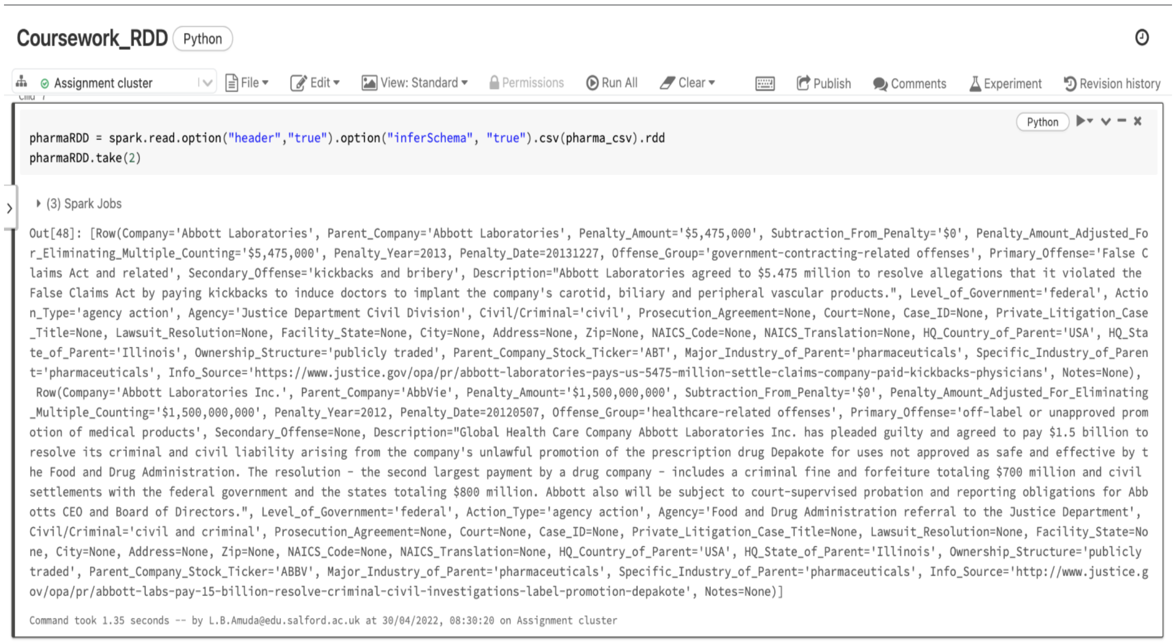
Cmd 5

```
# CREATING RDDS FROM CSV FILES
clinicaltrialRDD = create_rdd_from_file(clinicaltrial_csv)
clinicaltrialRDD.take(5)

(2) Spark Jobs

Out[46]: ['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|',
 'NCT02759848|Istanbul Medeniyet University|Completed|Jan 2012|Dec 2014|Observational|May 2016|Tuberculosis,Lung Diseases,Pulmonary Disease|',
 'NCT02758860|University of Roma La Sapienza|Active, not recruiting|Jun 2016|Sep 2020|Observational [Patient Registry]|Apr 2016|Diverticular Diseases,Diverticulum,Divert
iculosis|']

Command took 1.71 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:19 on Assignment cluster
```



```
meshRDD = create_rdd_from_file(mesh_csv)
meshRDD.take(5)
```

```
▸ (2) Spark Jobs
Out[47]: ['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']

Command took 0.73 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:19 on Assignment cluster
```



```
pharmaRDD = spark.read.option("header","true").option("inferSchema", "true").csv(pharma_csv).rdd
pharmaRDD.take(2)
```

```
▸ (3) Spark Jobs
Out[48]: [Row(Company='Abbott Laboratories', Parent_Company='Abbott Laboratories', Penalty_Amount='$5,475,000', Subtraction_From_Penalty='$0', Penalty_Amount_Adjusted_Fo
r_Eliminating_Multiple_Counting='$5,475,000', Penalty_Year=2013, Penalty_Date=20131227, Offense_Group='government-contracting-related offenses', Primary_Offense='False C
laims Act and related', Secondary_Offense='kickbacks and bribery', Description="Abbott Laboratories agreed to $5.475 million to resolve allegations that it violated the
False Claims Act by paying kickbacks to induce doctors to implant the company's carotid, biliary and peripheral vascular products.", Level_of_Government='federal', Actio
n_Type='agency action', Agency='Justice Department Civil Division', Civil/Criminal='civil', Prosecution_Agreement=None, Court=None, Case_ID=None, Private_Litigation_Case
_Title=None, Lawsuit_Resolution=None, Facility_State=None, City=None, Address=None, Zip=None, NAICS_Code=None, NAICS_Translation=None, HQ_Country_of_Parent='USA', HQ_Sta
te_of_Parent='Illinois', Ownership_Structure='publicly traded', Parent_Company_Stock_Ticker='ABT', Major_Industry_of_Parent='pharmaceuticals', Specific_Industry_of_Paren
t='pharmaceuticals', Info_Source='https://www.justice.gov/opa/pr/abbott-laboratories-pays-us-5475-million-settle-claims-company-paid-kickbacks-physicians', Notes=None),
 Row(Company='Abbott Laboratories Inc.', Parent_Company='AbbVie', Penalty_Amount='$1,500,000,000', Subtraction_From_Penalty='$0', Penalty_Amount_Adjusted_For_Eliminating
_Multiple_Counting='$1,500,000,000', Penalty_Year=2012, Penalty_Date=20120507, Offense_Group='healthcare-related offenses', Primary_Offense='off-label or unapproved prom
otion of medical products', Secondary_Offense=None, Description="Global Health Care Company Abbott Laboratories 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 the prescription drug Depakote for uses not approved as safe and effective by t
he 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 Abb
otts CEO and Board of Directors.", Level_of_Government='federal', Action_Type='agency action', Agency='Food and Drug Administration referral to the Justice Department',
Civil/Criminal='civil and criminal', Prosecution_Agreement=None, Court=None, Case_ID=None, Private_Litigation_Case_Title=None, Lawsuit_Resolution=None, Facility_State=No
ne, City=None, Address=None, Zip=None, NAICS_Code=None, NAICS_Translation=None, HQ_Country_of_Parent='USA', HQ_State_of_Parent='Illinois', Ownership_Structure='publicly
traded', Parent_Company_Stock_Ticker='ABBV', Major_Industry_of_Parent='pharmaceuticals', Specific_Industry_of_Parent='pharmaceuticals', Info_Source='http://www.justice.g
ov/opa/pr/abbott-labs-pay-15-billion-resolve-criminal-civil-investigations-label-promotion-depakote', Notes=None)]

Command took 1.35 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:20 on Assignment cluster
```

Problem answers

Question 1

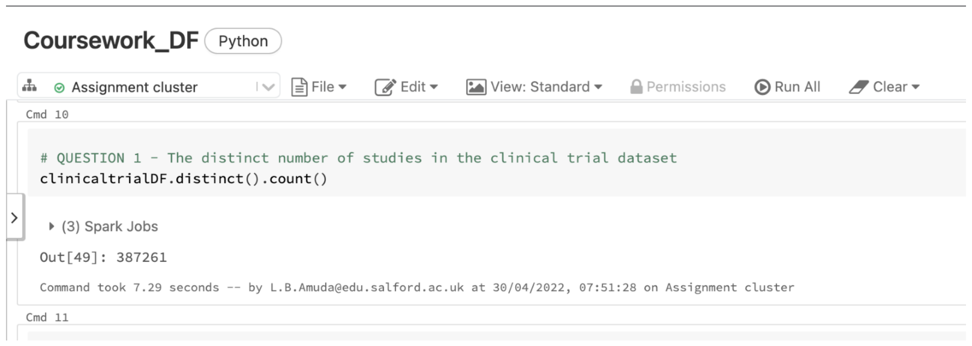
- **Assumptions made**

For this task, it is assumed that the clinical trial dataset contains some duplicate data. Duplicate data is not accepted much in Analytics, as it makes it difficult to analyse data easily.

This helps in avoiding duplicates of the data, making the data analysis easier.

- **PySpark implementation outline: DataFrame**

The Distinct function was used along with the Count function, to find the number of unique records present in the dataframe.



```
# QUESTION 1 - The distinct number of studies in the clinical trial dataset
clinicaltrialDF.distinct().count()
```

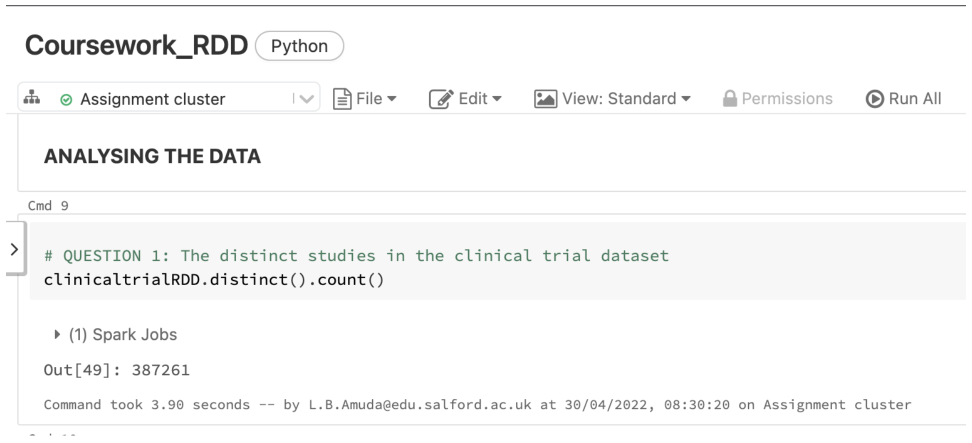
```
Out[49]: 387261
Command took 7.29 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster
```

- **HiveQL implementation outline**

The Distinct keyword was used in the Select statement to fetch unique rows in the table, and then the Count keyword counts it.

- **PySpark implementation outline: RDD**

This is the same as the DataFrame implementation stated above. The **Distinct and Count func**tions were used to count the number of unique records present in the RDD.



```
# QUESTION 1: The distinct studies in the clinical trial dataset
clinicaltrialRDD.distinct().count()
```

```
Out[49]: 387261
Command took 3.90 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:20 on Assignment cluster
```

- **Result on the submission for final data**

Below are the results for each implementation:

- **Discussion of result**

Based on the result, there are **387261** distinct studies in the clinical trial datasets for the year **2021**, which is the same as the total amount of data in the dataset. With this, we can infer that the dataset did not contain duplicate data.

Also, by observation, the 2021 studies were higher than the **2019** and **2020** studies, which have **326348** and **356466** studies**,** respectively. With this we can deduce that people are becoming more open to or interested in clinical trials with each year and hence we can predict that there would be an increase in clinical trial studies in the year 2022.

However, another explanation for this increase would be that there has been an increase in the number of people affected by the Conditions.

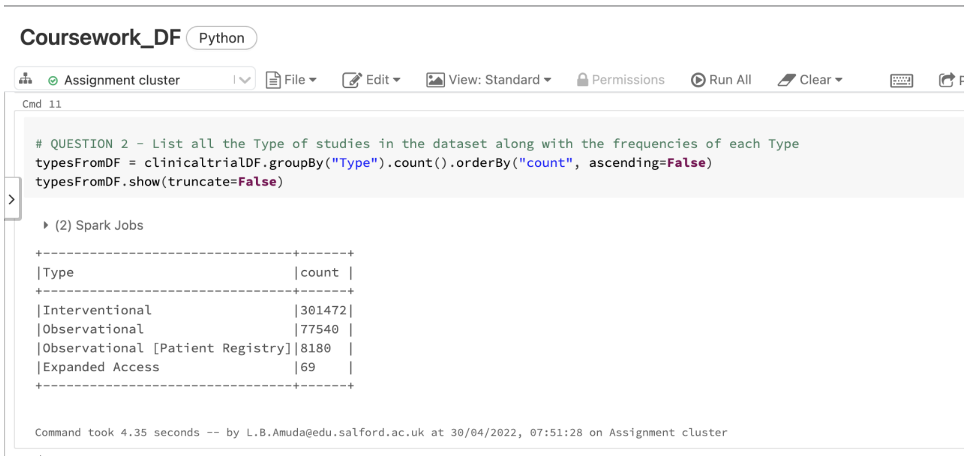
Question 2

- **Assumptions made**

It was assumed that the clinical trial data did not contain duplicate data. It was also assumed that the **Type** column is not null, that is, every record has a **Type** value.

- **PySpark implementation outline: DataFrame**

The clinical trial data was grouped by the Type column; this simply groups the rows based on columnar value (Type). Then the count function was used to count the value and return the count of rows for each group. The resulting count of the aggregated data is then sorted/ordered in descending order, from most frequent to least frequent.



```
# QUESTION 2 - List all the Type of studies in the dataset along with the frequencies of each Type
typesFromDF = clinicaltrialDF.groupBy("Type").count().orderBy("count", ascending=False)
typesFromDF.show(truncate=False)
```

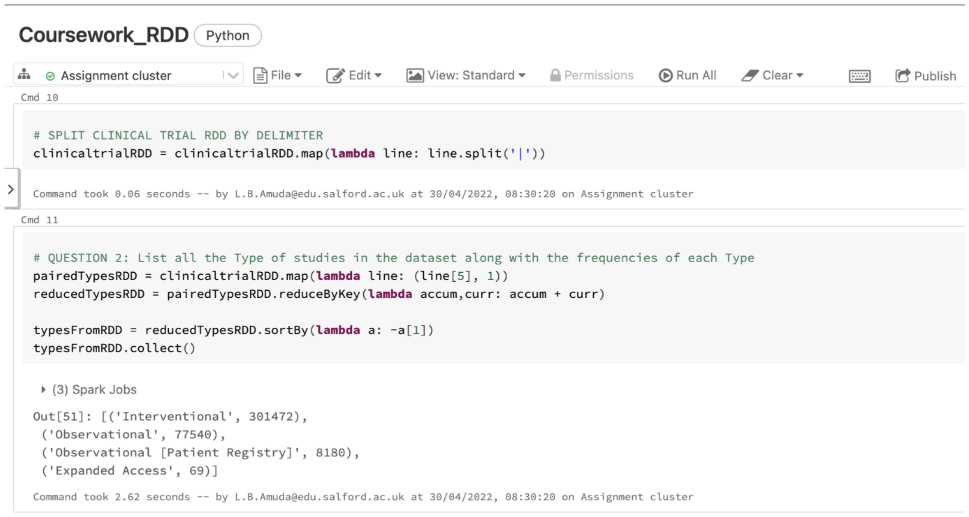
```
+--------------------------------+------+
|Type                            |count |
+--------------------------------+------+
|Interventional                  |301472|
|Observational                   |77540 |
|Observational [Patient Registry]|8180  |
|Expanded Access                 |69    |
+--------------------------------+------+

Command took 4.35 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster
```

- **HiveQL implementation outline**

Using the Select statement, Count and Group By clause, the clinical trial data was grouped based on the Type column, to produce a result of each Type and their frequency. The resulting data was then ordered by the Count column in descending order, to show the frequency from most to least.

- **PySpark implementation outline: RDD**

The first thing I did was to split the clinical trial RDD by the pipe delimiter (|). Then a map operation was performed to generate a pair RDD from the Type column (index 5 of the RDD). The values of each key in the pair RDD were merged using the reduceByKey function. The resulting aggregate was then sorted from most frequent to least frequent.



```
# SPLIT CLINICAL TRIAL RDD BY DELIMITER
clinicaltrialRDD = clinicaltrialRDD.map(lambda line: line.split('|'))

Command took 0.06 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:20 on Assignment cluster
```

```
# QUESTION 2: List all the Type of studies in the dataset along with the frequencies of each Type
pairedTypesRDD = clinicaltrialRDD.map(lambda line: (line[5], 1))
reducedTypesRDD = pairedTypesRDD.reduceByKey(lambda accum,curr: accum + curr)

typesFromRDD = reducedTypesRDD.sortBy(lambda a: -a[1])
typesFromRDD.collect()

Out[51]: [('Interventional', 301472),
 ('Observational', 77540),
 ('Observational [Patient Registry]', 8180),
 ('Expanded Access', 69)]

Command took 2.62 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:20 on Assignment cluster
```

- **Result on the submission for final data**

Below are the results for each implementation:

- **Discussion of result**

From the result, *Interventional* was the most frequent type and *Expanded Access* was the least frequent type of studies in the dataset for the year 2021. This is the same for the years 2019 and 2020. Interventional trials were higher in 2021 (value of **301472**) compared to 2019 and 2020 (which was 255945 and 277631 respectively), while Expanded Access trials remained at 69 over these 3-year period.

With this data we can deduce that people are more interested in Interventional clinical trials where they are part of treatment groups, receiving little or no intervention so that researchers can evaluate outcomes. This approach is more predictive and safer. While they are less interested in the Expanded Access trials, as this involves an investigational approach with unapproved medication, devices, and treatment methods. This approach is risky, and safety is not guaranteed.

Question 3

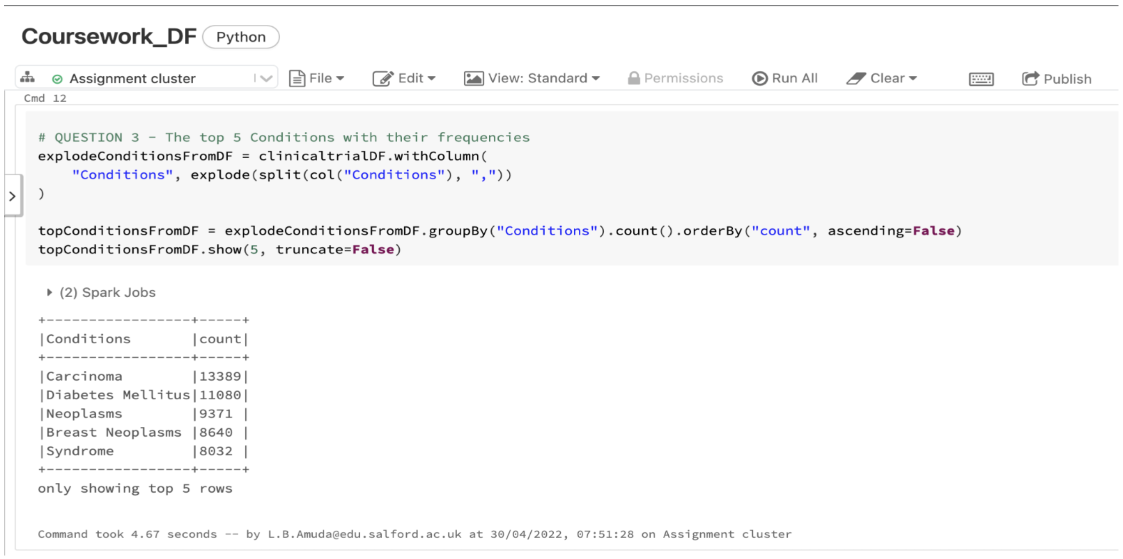
- **Assumptions made**

It was assumed that all the values of the Conditions column are comma-separated only. It was also assumed that the column is not null.

- **PySpark implementation outline: DataFrame**

Using the split function, the Conditions column was split by comma into a list of Conditions. Then the explode function was used to flatten/break each Condition in the list into separate rows.

The exploded data was then grouped and counted, to determine the frequency of each Condition, and ordered by the count in descending order, from highest to lowest. And then the top five most frequent Conditions were selected and returned.



```
# QUESTION 3 - The top 5 Conditions with their frequencies
explodeConditionsFromDF = clinicaltrialDF.withColumn(
    "Conditions", explode(split(col("Conditions"), ","))
)

topConditionsFromDF = explodeConditionsFromDF.groupBy("Conditions").count().orderBy("count", ascending=False)
topConditionsFromDF.show(5, truncate=False)
```

```
+-----------------+-----+
|Conditions       |count|
+-----------------+-----+
|Carcinoma        |13389|
|Diabetes Mellitus|11080|
|Neoplasms        |9371 |
|Breast Neoplasms |8640 |
|Syndrome         |8032 |
+-----------------+-----+
only showing top 5 rows

Command took 4.67 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster
```

- **HiveQL implementation outline**

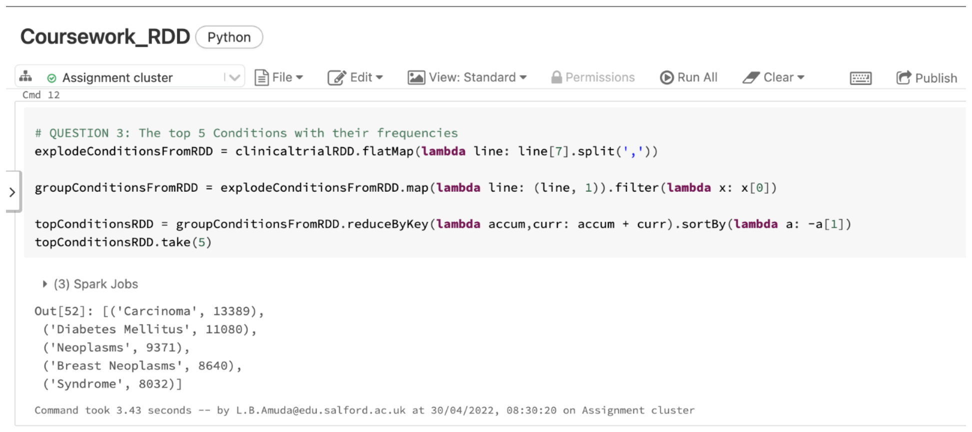
This implementation consists of nested queries. In the inner query, the Select statement was used to select the Conditions column from the clinical trial view, and the Where clause was used to ensure that only Conditions that are not empty were selected. Then the split and explode functions were applied on the column to split the comma-separated column and break each Condition to a separate row.

The outer query selects the already exploded Conditions column from the inner query, and the column was grouped and counted using the Group by clause and the Count function and then the Order by the command was used to sort the grouped Conditions in descending order and the Limit clause was used to select the top 5 conditions.

- **PySpark implementation outline: RDD**

In this implementation, the first thing I did was to split each element in the RDD by comma and flatten them using the flatMap transformation, to return an RDD which has a single Condition per row.

Then a pair RDD was generated and filtered to return only Conditions that are not empty. Using the reduceByKey transformation, the Conditions were grouped and then data is transformed with sort by function to arrange the Conditions in order of their frequencies, from highest to lowest. The top five conditions were selected and returned.



```
# QUESTION 3: The top 5 Conditions with their frequencies
explodeConditionsFromRDD = clinicaltrialRDD.flatMap(lambda line: line[7].split(','))

groupConditionsFromRDD = explodeConditionsFromRDD.map(lambda line: (line, 1)).filter(lambda x: x[0])

topConditionsRDD = groupConditionsFromRDD.reduceByKey(lambda accum,curr: accum + curr).sortBy(lambda a: -a[1])
topConditionsRDD.take(5)
```

```
Out[52]: [('Carcinoma', 13389),
 ('Diabetes Mellitus', 11080),
 ('Neoplasms', 9371),
 ('Breast Neoplasms', 8640),
 ('Syndrome', 8032)]
```

```
Command took 3.43 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:20 on Assignment cluster
```

- **Result on the submission for final data**

Below are the results for each implementation:

- **Discussion of result**

The result shows that the top 5 most frequent Conditions in 2021 studies are the same as 2019 and 2021 studies, with *Carcinoma* being the most common Condition in the studies, and *Syndrome* remains the least common, in the top 5.

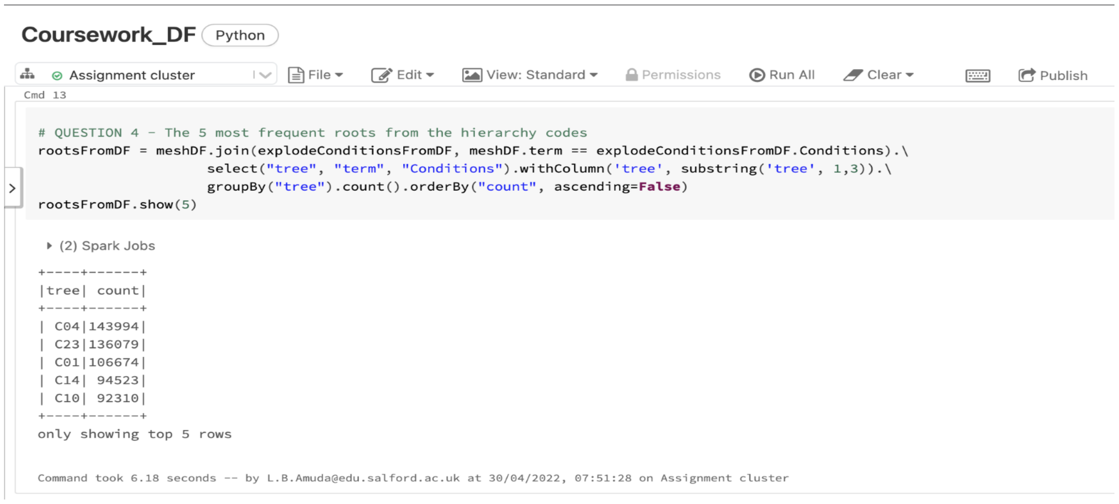
Question 4

- **Assumptions made**

It was assumed the **term** column in the *mesh* dataset is similar to the **Conditions** column of the *clinical trial* dataset, such that each Condition can be mapped to one or more hierarchy codes (**tree** column) in the *mesh* dataset.

- **PySpark implementation outline: DataFrame**

The *mesh* and the *exploded clinical trial* data were joined, on the basis that the *term* and *Conditions* columns are the same, then the required columns were selected, and the substring function was applied to the tree column to extract the first 3 characters of the column which is the root of the Conditions. Then the tree column was grouped, counted, and ordered in descending order, and the 5 most frequent roots were returned.



```
# QUESTION 4 - The 5 most frequent roots from the hierarchy codes
rootsFromDF = meshDF.join(explodeConditionsFromDF, meshDF.term == explodeConditionsFromDF.Conditions).\
                    select("tree", "term", "Conditions").withColumn('tree', substring('tree', 1,3)).\
                    groupBy("tree").count().orderBy("count", ascending=False)
rootsFromDF.show(5)
```

(2) Spark Jobs

```
+----+------+
|tree| count|
+----+------+
| C04|143994|
| C23|136079|
| C01|106674|
| C14| 94523|
| C10| 92310|
+----+------+
only showing top 5 rows
```

Command took 6.18 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster

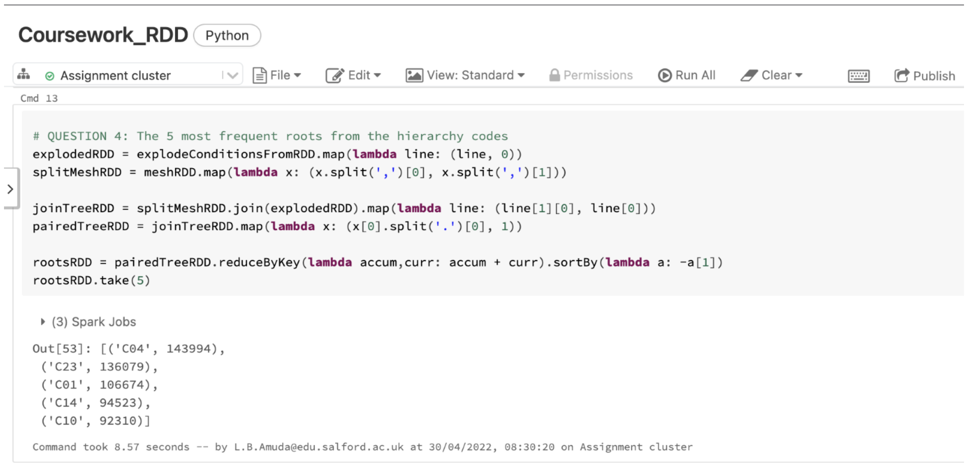
- **HiveQL implementation outline**

Using the ***Join*** command, the mesh and exploded clinical trial views were joined where the *term* and *Conditions* columns are the same, the ***Left*** function was used to select the first 3 characters of the hierarchy codes(roots) as tree column. This tree column was then grouped, counted, and ordered in descending order and the Limit clause was used to return the 5 top roots.

- **PySpark implementation outline: RDD**

The mesh RDD was split by comma to create a new RDD that contained both the *tree* values and *term* values. A pair RDD was created from the *exploded clinical trial RDD*, with the value of the pair set to zero (0) and the key being the *Conditions* value, this is to make sure both RDDs are in the form of (key, value) for each element of the RDDs especially if they are to be joined. The two RDDs were joined with the join function and mapped to return an RDD that contained the *tree* and the *Conditions* values.

A pair RDD was generated, and this contained the first 3 characters of the *tree* values, which were selected using the split function. This pair RDD was reduced by key to get a cumulation of the value of each key and then sorted in descending order. The first 5 top values were returned, to reveal the most frequent roots from the hierarchy codes.



```
# QUESTION 4: The 5 most frequent roots from the hierarchy codes
explodedRDD = explodeConditionsFromRDD.map(lambda line: (line, 0))
splitMeshRDD = meshRDD.map(lambda x: (x.split(',')[0], x.split(',')[1]))

joinTreeRDD = splitMeshRDD.join(explodedRDD).map(lambda line: (line[1][0], line[0]))
pairedTreeRDD = joinTreeRDD.map(lambda x: (x[0].split('.')[0], 1))

rootsRDD = pairedTreeRDD.reduceByKey(lambda accum,curr: accum + curr).sortBy(lambda a: -a[1])
rootsRDD.take(5)
```

```
▸ (3) Spark Jobs
Out[53]: [('C04', 143994),
 ('C23', 136079),
 ('C01', 106674),
 ('C14', 94523),
 ('C10', 92310)]
Command took 8.57 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:20 on Assignment cluster
```

- **Result on the submission for final data**

Below are the results for each implementation:

- **Discussion of result**

Of the 5 most frequent roots in the year 2021, the leading root with the highest frequency was the **C04** which had a value of **143994** and the least was **C10** with a value of **92310**. Compared to the 2019 and 2020 datasets which also had the most common root **C04** at **123221** and **133091**, respectively.

Although similar to the previous years, the 2021 roots had a higher value. With this information, it is safe to infer that these top 5 roots (the root causes of the Conditions) are constantly increasing and an increase in the values can be expected in the following years. Based on these data, the most frequent root causes of various Conditions have been identified and this can be used to provide a means of educating people about these root causes, to help reduce and even prevent the occurrence of these Conditions.

Question 5

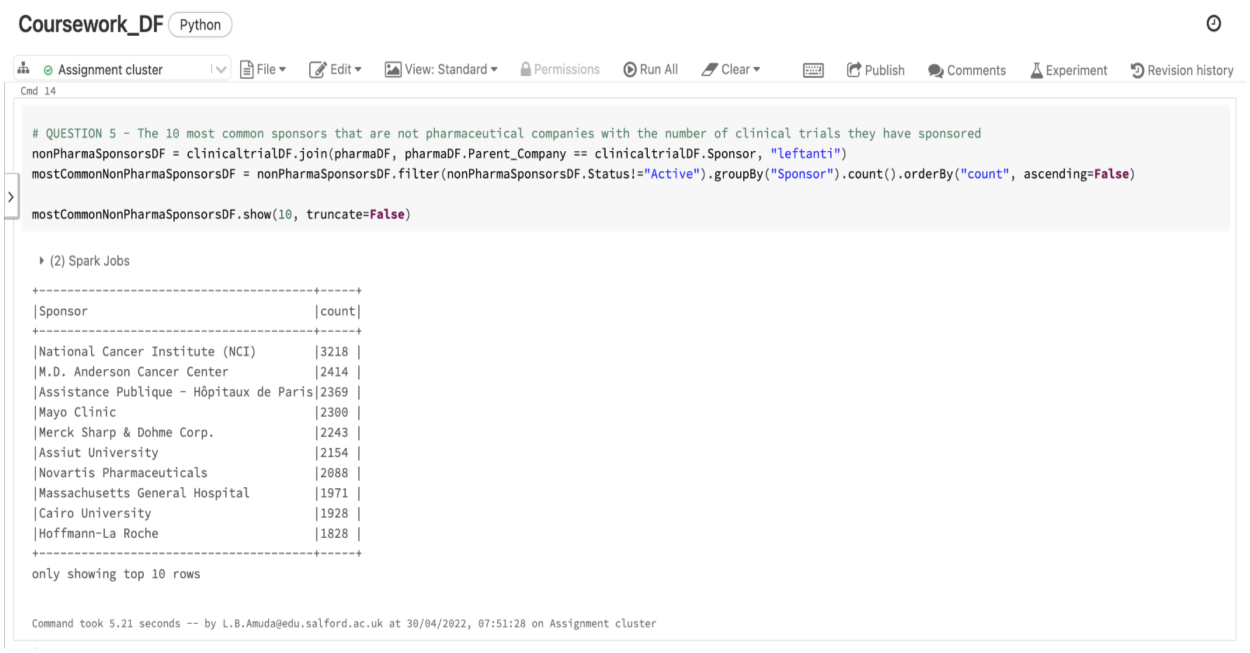
- **Assumptions made**

It was assumed that the *Parent Company* column of the pharma data only contained pharmaceutical companies, and based on this, it was also assumed that the non-pharmaceutical Sponsors are those that do not exist in the *Parent Company* column.

It was assumed that only non-active trials should be considered, hence, all trials with *Status* **Active** should be excluded.

- **PySpark implementation outline: DataFrame** (description of main ideas in words and screenshot of code)

The clinical trial and pharma dataframes were joined and the *leftanti* operation was performed, to return only columns in the clinical trial data for non-matched records in the pharma data. Then the resulting data was filtered to remove the studies that have their Status value as **Active**. The non-active non-pharmaceutical sponsored data was then grouped by the Sponsor column and then ordered in descending order, with the top 10 values returned.



```
# QUESTION 5 - The 10 most common sponsors that are not pharmaceutical companies with the number of clinical trials they have sponsored
nonPharmaSponsorsDF = clinicaltrialDF.join(pharmaDF, pharmaDF.Parent_Company == clinicaltrialDF.Sponsor, "leftanti")
mostCommonNonPharmaSponsorsDF = nonPharmaSponsorsDF.filter(nonPharmaSponsorsDF.Status!="Active").groupBy("Sponsor").count().orderBy("count", ascending=False)

mostCommonNonPharmaSponsorsDF.show(10, truncate=False)
```

```
+---------------------------------------+-----+
|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 5.21 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster

- **HiveQL implementation outline** (description of main ideas in words and screenshot of code)

This implementation consists of two nested queries. In the inner query, a Left Outer Join was performed on the clinical trial and pharma views, to return all rows in the clinical trial view and all the matching rows found in the pharma view, with a Where clause to select the rows only when the Parent Company column is not null and the Status in not Active.

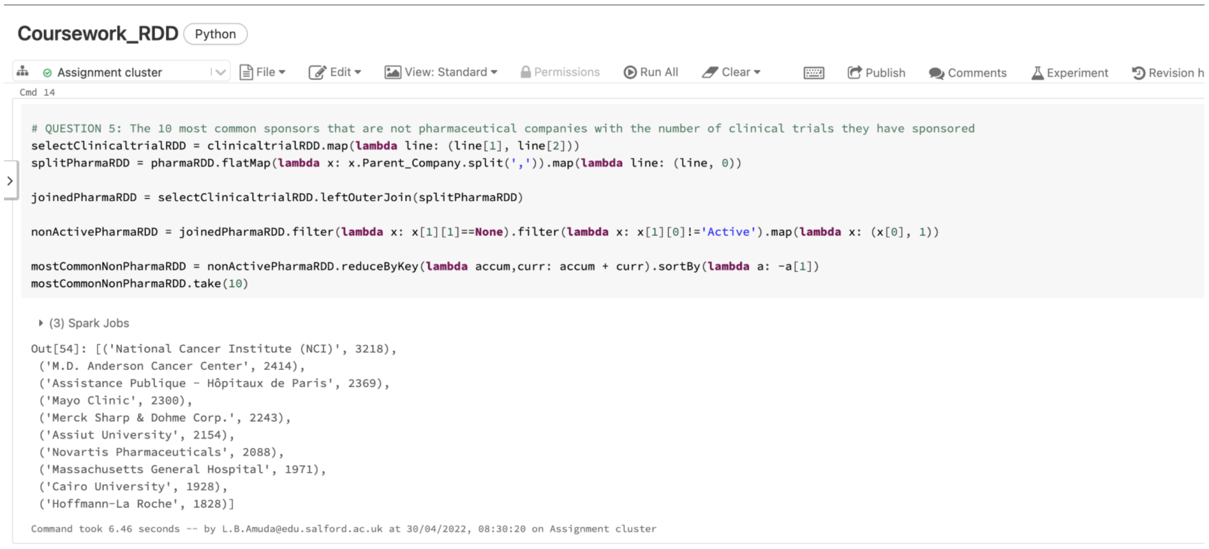
The outer query then selected and counted the rows of the Sponsor column, grouped and ordered the result in descending order and using the Limit clause to pick the top 10 Sponsors.

- **PySpark implementation outline: RDD** (description of main ideas in words and screenshot of code)

The clinical trial data was mapped to return a list of rows containing 2 values each, the Sponsor and the Status of the trial, which are in index 1 and 2, respectively. The flat map function was used to split and map the Parent Company column of the pharma data and a pair RDD was generated, to make sure both RDDs are in the form of (key, value) for each element of the RDDs since they would be joined.

Using the left outer join function, these 2 RDDs were joined to return all the rows from the clinical trial data where the Sponsor values matched the Parent Company values of the pharma data, and null values were assigned for the Parent Company records that did not match. Since we only want the *Sponsors* that do not exist in the *Parent Company*, the filter function was used to select and return only records where Parent Company values as *None* and Status values that are not *Active*.

A pair RDD was generated and reduced by key function was invoked to get a cumulation of the value of each key and then sorted in descending order. The 10 most common sponsors that are not pharmaceutical companies were returned along with their frequencies.



```
# QUESTION 5: The 10 most common sponsors that are not pharmaceutical companies with the number of clinical trials they have sponsored
selectClinicaltrialRDD = clinicaltrialRDD.map(lambda line: (line[1], line[2]))
splitPharmaRDD = pharmaRDD.flatMap(lambda x: x.Parent_Company.split(',')).map(lambda line: (line, 0))

joinedPharmaRDD = selectClinicaltrialRDD.leftOuterJoin(splitPharmaRDD)

nonActivePharmaRDD = joinedPharmaRDD.filter(lambda x: x[1][1]==None).filter(lambda x: x[1][0]!='Active').map(lambda x: (x[0], 1))

mostCommonNonPharmaRDD = nonActivePharmaRDD.reduceByKey(lambda accum,curr: accum + curr).sortBy(lambda a: -a[1])
mostCommonNonPharmaRDD.take(10)
```

(3) Spark Jobs

```
Out[54]: [('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 6.46 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:20 on Assignment cluster

- **Result on the submission for final data** – results for all implementations must be included (screenshots are fine)

Below are the results for each implementation:

- **Discussion of result**

From the result, the *National Cancer Institute (NCI)*, the leading Sponsor, which is not a pharmaceutical company, had **3218** sponsored trials in the year 2021. They were also the leading Sponsor in the year 2019 and 2020, with **3003** and **3100** sponsored trials, respectively.

Of the top 10 most common non-pharmaceutical company sponsors in the year 2021, *Hoffmann-La Roche* had the least number of trials (**1828** trials). However, this value is still higher than 2019 and 2020 values which are **1500** trials for *Eli Lilly and Company* and **1720** for *National Taiwan University Hospital*, respectively.

Question 6

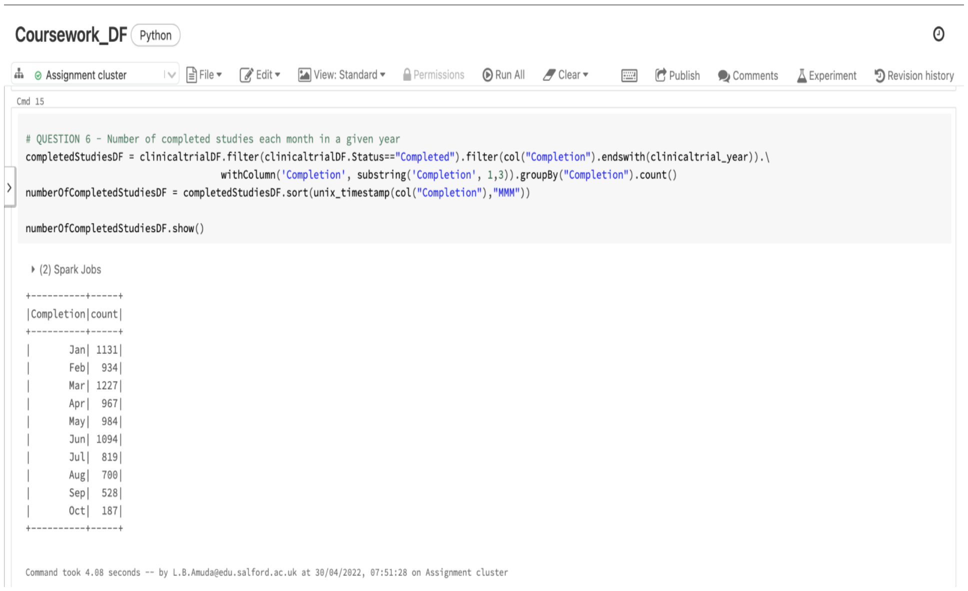
- **Assumptions made**

It was assumed that only clinical trials with ***Status*** *Completed* are completed, other Statuses such as *Terminated* or *Unknown status* were not categorized as completed. It was also assumed that all the months should be ordered/sorted chronologically and not based on their number of completed studies.

- **PySpark implementation outline: DataFrame**

The clinical trial data was filtered where the *Status* is *Completed* and where the Completion year is the current clinical trial year. The *with column* function was used with the *substring* function to extract the months from the *Completion* column. The resulting data were then grouped by *Completion* and counted. Using the *sort* function and the pyspark's *unix timestamp*, the months were ordered chronologically.

To visualize the result, the *matplotlib* library was used to create bar charts to show the number of completed studies for each month in the specified clinical trial year.

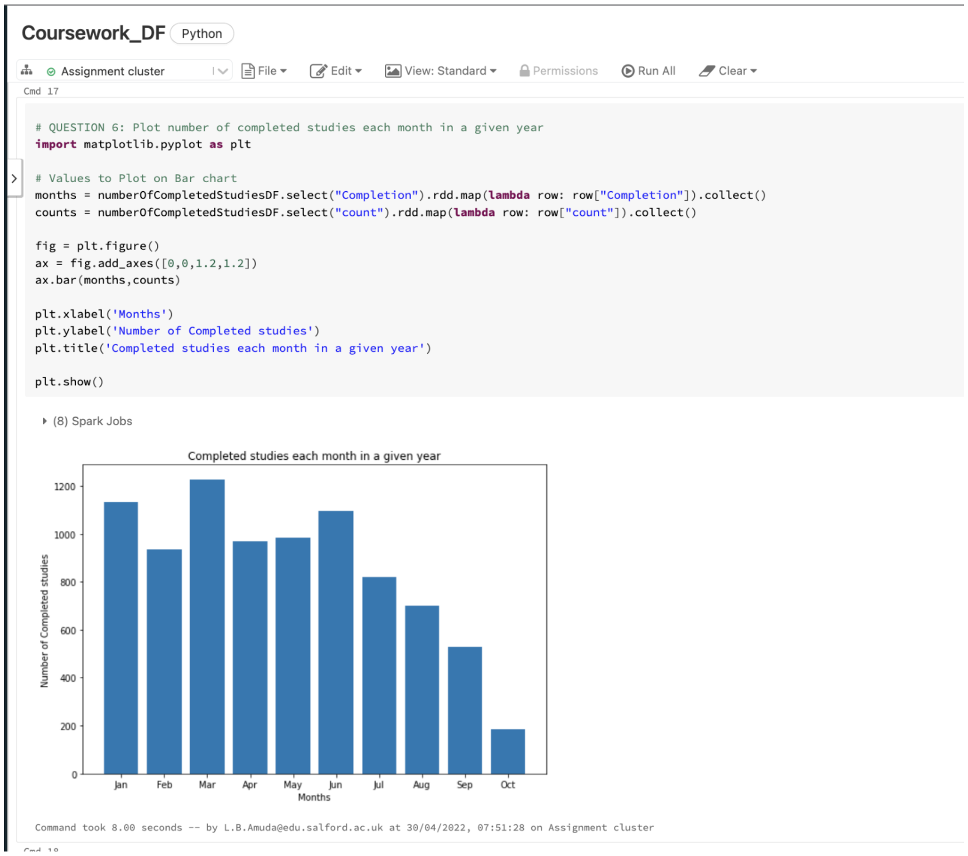


```
# QUESTION 6 - Number of completed studies each month in a given year
completedStudiesDF = clinicaltrialDF.filter(clinicaltrialDF.Status=="Completed").filter(col("Completion").endswith(clinicaltrial_year)).\
                                withColumn('Completion', substring('Completion', 1,3)).groupBy("Completion").count()
numberOfCompletedStudiesDF = completedStudiesDF.sort(unix_timestamp(col("Completion"),"MMM"))

numberOfCompletedStudiesDF.show()
```

(2) Spark Jobs

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

```
Command took 4.08 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster
```



```
# QUESTION 6: Plot number of completed studies each month in a given year
import matplotlib.pyplot as plt

# Values to Plot on Bar chart
months = numberOfCompletedStudiesDF.select("Completion").rdd.map(lambda row: row["Completion"]).collect()
counts = numberOfCompletedStudiesDF.select("count").rdd.map(lambda row: row["count"]).collect()

fig = plt.figure()
ax = fig.add_axes([0,0,1.2,1.2])
ax.bar(months,counts)

plt.xlabel('Months')
plt.ylabel('Number of Completed studies')
plt.title('Completed studies each month in a given year')

plt.show()
```

(8) Spark Jobs
Completed studies each month in a given year
Number of Completed studies
0
200
400
600
800
1000
1200
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Months

```
Command took 8.00 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 07:51:28 on Assignment cluster
```

- **HiveQL implementation outline**

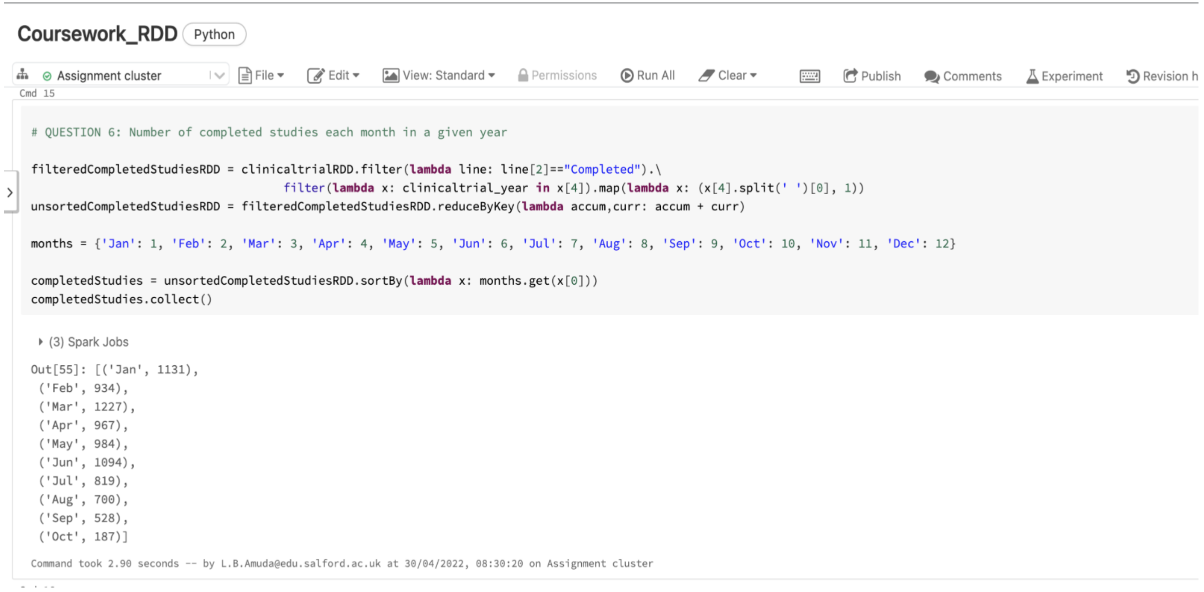
The select statement is used with the Left function to extract the month name from the Completion column, where the trial Status is Completed, and the Completion year is the specified trial year. Then, the result was grouped by the Completion column and counted, and then ordered by using the Hive's *unix timestamp* function to sort the months in chronological order.

The *bokeh* library was used to create bar charts to plot the number of completed studies for each month in the specified clinical trial year.

- **PySpark implementation outline: RDD**

The clinical trial data was filtered where the Status is Completed and the year from the Completion column is the current clinical trial year to return only completed trials. Then a map operation was performed, to extract only months from the Completion column using the split function, and to create a pair RDD. The value of this pair RDD was cumulated using reduce by key function.

To sort the resulting RDD in chronological order of months in a year, a month's dictionary was created. The dictionary had a *get* method which was used to obtain the value of each month based on the corresponding RDD key and the sort by operation was performed on it.



```
# QUESTION 6: Number of completed studies each month in a given year

filteredCompletedStudiesRDD = clinicaltrialRDD.filter(lambda line: line[2]=="Completed").\
                                    filter(lambda x: clinicaltrial_year in x[4]).map(lambda x: (x[4].split(' ')[0], 1))
unsortedCompletedStudiesRDD = filteredCompletedStudiesRDD.reduceByKey(lambda accum,curr: accum + curr)

months = {'Jan': 1, 'Feb': 2, 'Mar': 3, 'Apr': 4, 'May': 5, 'Jun': 6, 'Jul': 7, 'Aug': 8, 'Sep': 9, 'Oct': 10, 'Nov': 11, 'Dec': 12}

completedStudies = unsortedCompletedStudiesRDD.sortBy(lambda x: months.get(x[0]))
completedStudies.collect()
```

```
Out[55]: [('Jan', 1131),
 ('Feb', 934),
 ('Mar', 1227),
 ('Apr', 967),
 ('May', 984),
 ('Jun', 1094),
 ('Jul', 819),
 ('Aug', 700),
 ('Sep', 528),
 ('Oct', 187)]

Command took 2.90 seconds -- by L.B.Amuda@edu.salford.ac.uk at 30/04/2022, 08:30:20 on Assignment cluster
```

- **Result on the submission for final data**

Below are the results for each implementation:

- **Discussion of result**

From the result, the highest number of studies in the year **2021** occurred in March, with the value at **1227**. However, the months November and December were not included in the 2021 dataset as at the time of this report.

However, based on the data obtained from the previous years, 2019 and 2020, December had the highest number of studies (**2690** and **2084**, respectively).

References

- https://clinicaltrials.gov/ct2/about-studies/glossary#:~:text=Interventional%20study%20(clinical%20trial),biomedical%20or%20health%2Drelated%20outcomes.
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5443564/
- https://sparkbyexamples.com/pyspark-tutorial/
- https://www.tutorialspoint.com/hive/hiveql_select_order_by.htm
- https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML
- https://www.educba.com/data-science/data-science-tutorials/spark-tutorial/
- https://stackoverflow.com/questions/71768512/sort-by-key-month-using-rdds-in-pyspark
- https://www.tutorialspoint.com/matplotlib/matplotlib_bar_plot.htm
-
-