Name: **Lateefat Amuda**

Student ID: **@00651625**

Module Title: Big Data Tools & Techniques for MSc

Assessment Title: Analysis of clinical trial data

Description of any setup required

The clinical trial, mesh and pharma files should be uploaded to the **tables** folder in the Databricks **FileStore**. There is a **Scripts** Notebook attached as part of this submission, it contains the scripts needed for file loading. A cell has been added to each Notebook to ensure that the *Scripts* Notebook is executed first.

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

For Hive implementation, after loading the data, the source file will be deleted from the source location. Hence, the files were copied into another location and then loaded from there.

Data cleaning and preparation

- **PySpark DataFrame**

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 the *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 and pharma csv files, respectively.

```
from pyspark.sql.functions import *
```

Command took 0.05 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:09:58 on Cluster1

Cmd 5

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

▸ (3) Spark Jobs

```
+-----------+--------------------+--------------------+--------+----------+--------------------+----------+--------------------+--------------------+
|         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 18.46 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:09:58 on Cluster1

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

▸ (3) Spark Jobs

```
+-----------+--------------------+
|       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 3.77 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:09:58 on Cluster1

```
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 2.91 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:09:58 on Cluster1

Images: Data cleaning and preparation for Dataframe implementation.

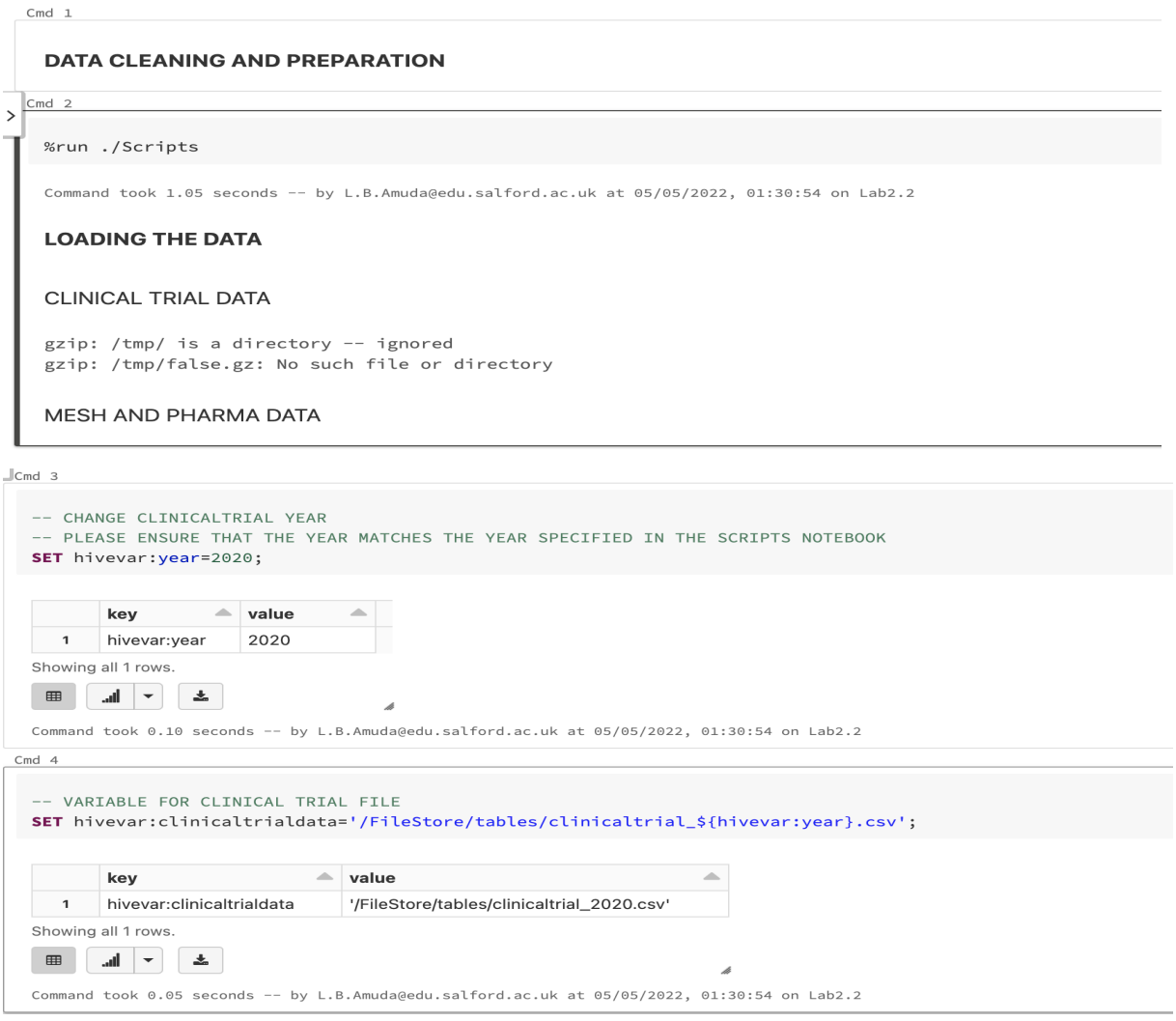
- **HiveQL**

To clean up and prevent data from being *cached* in the table or hive location, existing tables and views were removed. This helps for cases where the notebook commands would be executed multiple times for different years.

After removing the existing tables, new tables were created based on defined schemas, with the location for storing each table specified. Then data was loaded into each table, specifying the **overwrite** keyword, to prevent the data from being appended to the table every time the command is executed.

However, for the pharma data, another table was created, containing only the two columns needed (*Company* and *Parent_Company* columns). The data for the two columns was formatted to remove the double quotation marks that were appended to the data after loading.

Then the views were generated from each table, removing the *header* that was included when loading the csv files.



Cmd 5

```
%python
# DO NOT CHANGE THE CONTENT OF THIS CELL
# THE MESH AND PHARMA FILES GET REMOVED AFTER THE DATA IS LOADED, THIS WOULD HELP FIX THE ISSUE, BY COPYING THE FILES BACK TO FILESTORE
if 'dbruntime.dbutils' in sys.modules.keys():
    try:
        dbutils.fs.ls("/FileStore/tables/" + mesh_csv)
        dbutils.fs.ls("/FileStore/tables/" + pharma_csv)
    except:
        dbutils.fs.cp("file:/tmp/" + mesh_csv, "/FileStore/tables/" + mesh_csv)
        dbutils.fs.cp("file:/tmp/" + pharma_csv, "/FileStore/tables/" + pharma_csv)
```

Command took 0.10 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 6

REMOVING THE EXISTING TABLES AND VIEWS

Cmd 7

```
%python
clinicaltrial_tables = "/FileStore/clinicaltrials/"

try:
    if len(dbutils.fs.ls(clinicaltrial_tables)) > 0:
        dbutils.fs.rm(clinicaltrial_tables, True)
    dbutils.fs.rm('dbfs:/user/hive/warehouse/pharma_formatted_table/', True)
    dbutils.fs.rm('dbfs:/user/hive/warehouse/mesh/', True)
except:
    print("There's nothing to see here!")
```

Command took 0.25 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 8

```
DROP TABLE IF EXISTS clinicaltrial_${hivevar:year}_table;
DROP VIEW IF EXISTS clinicaltrial_${hivevar:year}_view;
DROP VIEW IF EXISTS explodedclinical_view;
DROP VIEW IF EXISTS completedstudies_view;
```

OK

Command took 0.31 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 9

```
DROP TABLE IF EXISTS mesh_table;
DROP VIEW IF EXISTS mesh_view;
```

OK

Command took 0.18 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 10

```
DROP TABLE IF EXISTS pharma_table;
DROP TABLE IF EXISTS pharma_formatted_table;
DROP VIEW IF EXISTS pharma_view;
```

OK

Command took 0.21 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 11

CREATING TABLES AND LOADING DATA INTO EACH ONE

Cmd 12

```
-- CREATE CLINICALTRIAL TABLE
CREATE TABLE IF NOT EXISTS clinicaltrial_${hivevar:year}_table(
  Id STRING,
  Sponsor STRING,
  Status STRING,
  Start STRING,
  Completion STRING,
  Type STRING,
  Submission STRING,
  Conditions STRING,
  Interventions STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
LOCATION '/FileStore/clinicaltrials';
```

OK

Command took 0.43 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 13

```
-- LOAD DATA INTO CLINICALTRIAL TABLE
LOAD DATA INPATH ${hivevar:clinicaltrialdata} OVERWRITE INTO TABLE clinicaltrial_${hivevar:year}_table;
```

OK

Command took 3.51 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 14

```
-- CREATE MESH TABLE
CREATE TABLE IF NOT EXISTS mesh_table(
  term STRING,
  tree STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/FileStore/mesh';
```

OK

Command took 0.31 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 15

```
-- LOAD DATA INTO MESH TABLE
LOAD DATA INPATH '/FileStore/tables/mesh.csv' OVERWRITE INTO TABLE mesh_table;
```

OK

Command took 2.27 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 16

```
-- CREATE PHARMA TABLE
CREATE TABLE IF NOT EXISTS pharma_table(
  Company STRING, Parent_Company STRING, Penalty_Amount STRING,
  Subtraction_From_Penalty STRING, Penalty_Amount_Adjusted_For_Eliminating_Multiple_Counting STRING,
  Penalty_Year STRING, Penalty_Date STRING, Offense_Group STRING, Primary_Offense STRING,
  Secondary_Offense STRING, Description STRING, Level_of_Government STRING, Action_Type STRING,
  Agency STRING, `Civil/Criminal` STRING, Prosecution_Agreement STRING, Court STRING,
  Case_ID STRING, Private_Litigation_Case_Title STRING, Lawsuit_Resolution STRING,
  Facility_State STRING, City STRING, Address STRING, Zip STRING, NAICS_Code STRING,
  NAICS_Translation STRING, HQ_Country_of_Parent STRING, HQ_State_of_Parent STRING,
  Ownership_Structure STRING, Parent_Company_Stock_Ticker STRING, Major_Industry_of_Parent STRING,
  Specific_Industry_of_Parent STRING, Info_Source STRING, Notes STRING
  )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/FileStore/pharma';
```

OK

Command took 0.34 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 17

```
-- LOAD DATA INTO PHARMA TABLE
LOAD DATA INPATH '/FileStore/tables/pharma.csv' OVERWRITE INTO TABLE pharma_table;
```

OK

Command took 2.04 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 18

```
-- FORMAT PHARMA TABLE TO REMOVE DOUBLE QUOTATIONS AND FIELDS THAT ARE NOT NEEDED
CREATE TABLE IF NOT EXISTS pharma_formatted_table
AS SELECT REGEXP_REPLACE(Company, '["]','') AS Company, REGEXP_REPLACE(Parent_Company, '["]','') AS Parent_Company
FROM pharma_table;
```

▸ (4) Spark Jobs

Query returned no results

Command took 19.19 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 19

CREATING VIEWS FROM TABLES

Cmd 20

```
-- CREATE A TEMPORARY VIEW FROM CLINICALTRIAL TABLE
CREATE TEMPORARY VIEW clinicaltrial_${hivevar:year}_view
AS SELECT * FROM clinicaltrial_${hivevar:year}_table WHERE Id!='Id';
```

OK

Command took 0.35 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 21

```
-- CREATE A TEMPORARY VIEW FROM MESH TABLE
CREATE TEMPORARY VIEW mesh_view
AS SELECT * FROM mesh_table WHERE term!='term';
```

OK

Command took 0.08 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 22

```
-- CREATE A TEMPORARY VIEW FROM PHARMA FORMATTED TABLE
CREATE TEMPORARY VIEW pharma_view
AS SELECT * FROM pharma_formatted_table WHERE Company!='Company';
```

OK

Command took 0.35 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Images: Data cleaning and preparation for HiveQL implementation.

- **PySpark RDD**

For reusability, a function was defined to create an RDD from a file and remove the first row in the RDD, which is the *header* from the csv file that was loaded.

This function was used to create clinical trials and mesh RDDs. However, this function could not be used on the pharma data as removing the double quotation from the dataset proved problematic. Hence, a dataframe was created from the pharma csv file and converted to RDD, using the *rdd* method.

PREPARE THE DATA

Cmd 2

RUN THE SCRIPT NOTEBOOK

Cmd 3

```
%run ./Scripts
```

```
Command took 6.47 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 00:58:56 on Lab2.2
```

LOADING THE DATA

CLINICAL TRIAL DATA

```
gzip: /tmp/ is a directory -- ignored
```

MESH AND PHARMA DATA

```
# 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 05/05/2022, 00:58:56 on Lab2.2
```

Cmd 6

```
# 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 0.03 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 00:58:56 on Lab2.2
```

Cmd 7

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

▸ (2) Spark Jobs

```
Out[42]: ['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,Pulmona
 'NCT02758860|University of Roma La Sapienza|Active, not recruiting|Jun 2016|Sep 2020|Observational [Patient Registry]|Apr 2016|D
iculosis|']
```

```
Command took 2.38 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 00:58:56 on Lab2.2
```

Cmd 8

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

▸ (2) Spark Jobs

```
Out[43]: ['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 1.11 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 00:58:56 on Lab2.2

Cmd 9

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

▸ (3) Spark Jobs

```
Out[44]: [Row(Company='Abbott Laboratories', Parent_Company='Abbott Laboratories', Penalty_Amount='$5,475,000', Subtraction_From_Pen
r_Eliminating_Multiple_Counting='$5,475,000', Penalty_Year=2013, Penalty_Date=20131227, Offense_Group='government-contracting-relate
laims Act and related', Secondary_Offense='kickbacks and bribery', Description="Abbott Laboratories agreed to $5.475 million to reso
False Claims Act by paying kickbacks to induce doctors to implant the company's carotid, biliary and peripheral vascular products."
n_Type='agency action', Agency='Justice Department Civil Division', Civil/Criminal='civil', Prosecution_Agreement=None, Court=None,
_Title=None, Lawsuit_Resolution=None, Facility_State=None, City=None, Address=None, Zip=None, NAICS_Code=None, NAICS_Translation=Non
te_of_Parent='Illinois', Ownership_Structure='publicly traded', Parent_Company_Stock_Ticker='ABT', Major_Industry_of_Parent='pharmac
t='pharmaceuticals', Info_Source='https://www.justice.gov/opa/pr/abbott-laboratories-pays-us-5475-million-settle-claims-company-paid
 Row(Company='Abbott Laboratories Inc.', Parent_Company='AbbVie', Penalty_Amount='$1,500,000,000', Subtraction_From_Penalty='$0', Pe
_Multiple_Counting='$1,500,000,000', Penalty_Year=2012, Penalty_Date=20120507, Offense_Group='healthcare-related offenses', Primary_
otion of medical products', Secondary_Offense=None, Description="Global Health Care Company Abbott Laboratories Inc. has pleaded gu
resolve its criminal and civil liability arising from the company's unlawful promotion of the prescription drug Depakote for uses no
he Food and Drug Administration. The resolution - the second largest payment by a drug company - includes a criminal fine and forfe
settlements with the federal government and the states totaling $800 million. Abbott also will be subject to court-supervised probat
otts CEO and Board of Directors.", Level_of_Government='federal', Action_Type='agency action', Agency='Food and Drug Administration
Civil/Criminal='civil and criminal', Prosecution_Agreement=None, Court=None, Case_ID=None, Private_Litigation_Case_Title=None, Lawsu
ne, City=None, Address=None, Zip=None, NAICS_Code=None, NAICS_Translation=None, HQ_Country_of_Parent='USA', HQ_State_of_Parent='Ill
traded', Parent_Company_Stock_Ticker='ABBV', Major_Industry_of_Parent='pharmaceuticals', Specific_Industry_of_Parent='pharmaceutica
ov/opa/pr/abbott-labs-pay-15-billion-resolve-criminal-civil-investigations-label-promotion-depakote', Notes=None)]
```

Command took 2.36 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 00:58:56 on Lab2.2

Images: Data cleaning and preparation for RDD implementation.

Problem answers

Question 1

- **Assumptions made**

For this task, it was assumed that the clinical trial data contained some duplicate data.

- **PySpark implementation outline: DataFrame**

A distinct count operation was performed to determine the total number of studies in the clinical trial, excluding the duplicate records.

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

▸ (3) Spark Jobs

```
Out[8]: 387261
```

```
Command took 11.60 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:09:59 on Cluster1
```

- **HiveQL implementation outline**

The *Distinct* keyword was used with the *Select* statement to fetch unique rows in the table and then count them.

```
-- QUESTION1: The number of distinct studies in the clinical trial dataset
SELECT DISTINCT COUNT(*) AS Count FROM clinicaltrial_${hivevar:year}_view;
```

▸ (2) Spark Jobs

	Count
1	387261

Showing all 1 rows.

```
Command took 1.51 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:37:59 on Cluster1
```

- **PySpark implementation outline: RDD**

This is the same as the DataFrame implementation stated above. The *Distinct* and *Count* functions were used to compute the number of unique records present in the clinical trial RDD.

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

- **Result on the submission**

Below are the results for each implementation.

ANALYSE THE DATA

Cmd 9

```
# The total number of clinical trial studies
clinicaltrialDF.count()
```

▸ (2) Spark Jobs

Out[7]: 387261

Command took 5.54 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:09:59 on Cluster1

Cmd 10

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

▸ (3) Spark Jobs

Out[8]: 387261

Command took 11.60 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:09:59 on Cluster1

Cmd 21

ANALYSING THE DATA

Cmd 22

```
-- Total number of studies in the clinical trial dataset
SELECT COUNT(*) AS Count FROM clinicaltrial_${hivevar:year}_view;
```

▸ (2) Spark Jobs

	Count
1	387261

Showing all 1 rows.

Command took 2.39 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:37:59 on Cluster1

Cmd 23

```
-- QUESTION1: The number of distinct studies in the clinical trial dataset
SELECT DISTINCT COUNT(*) AS Count FROM clinicaltrial_${hivevar:year}_view;
```

▸ (2) Spark Jobs

	Count
1	387261

Showing all 1 rows.

Command took 1.51 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:37:59 on Cluster1

ANALYSING THE DATA

Cmd 9

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

▸ (1) Spark Jobs

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

Images of Dataframe, HiveQL, and RDD results.

- **Discussion of result**

Based on the result, there are **387261** distinct studies in the 2021 clinical trials. This is the same as the total number of data in the dataset, with this, we can infer that the dataset did not contain duplicate data.

It was also observed that the number of studies in **2021** was higher than in **2019** and **2020**, which have **326348** and **356466** studies, respectively. From this observation, we can conclude 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 is a yearly increase in the number of people affected by the Conditions of these trials.

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 clinical trial has a Type.

- **PySpark implementation outline: DataFrame**

The clinical trial data was grouped by the Type column and counted, to obtain the frequency of each Type of study in the dataset. The resulting aggregate was then sorted in descending order from the most to the least frequent Type.

```
# 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)
```

- **HiveQL implementation outline**

To determine the frequencies of the Type of studies, the clinical trial data were grouped based on the Type column and counted. Then the result was ordered by the Count column in descending order.

```
-- QUESTION2: All the Type of studies in the dataset along with the frequencies of each one
SELECT Type,COUNT(*) AS Count FROM clinicaltrial_${hivevar:year}_view GROUP BY Type ORDER BY Count DESC;
```

- **PySpark implementation outline: RDD**

The clinical trial data was split and mapped to remove the delimiter and generate a pair RDD, which was then reduced by key to get a cumulation of the value of each key. The resulting aggregate was sorted in descending order.

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

```
Command took 0.03 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 10:04:33 on Cluster1
```

```
Cmd 11
```

```
# 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()
```

- **Result on the submission**

Below are the results for each implementation.

```
# 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)
```

```
▸ (2) Spark Jobs
+---------------------------------+------+
|Type                             |count |
+---------------------------------+------+
|Interventional                   |301472|
|Observational                    |77540 |
|Observational [Patient Registry]|8180  |
|Expanded Access                  |69    |
+---------------------------------+------+

Command took 6.26 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:09:59 on Cluster1
```

```
-- QUESTION2: All the Type of studies in the dataset along with the frequencies of each one
SELECT Type,COUNT(*) AS Count FROM clinicaltrial_${hivevar:year}_view GROUP BY Type ORDER BY Count DESC;
```

▸ (2) Spark Jobs

	Type	Count
1	Interventional	301472
2	Observational	77540
3	Observational [Patient Registry]	8180
4	Expanded Access	69

Showing all 4 rows.

```
Command took 2.39 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:37:59 on Cluster1
```

```
# 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()
```

▸ (3) Spark Jobs

```
Out[13]: [('Interventional', 301472),
 ('Observational', 77540),
 ('Observational [Patient Registry]', 8180),
 ('Expanded Access', 69)]
```

Command took 3.08 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 10:04:33 on Cluster1

Images of Dataframe, HiveQL, and RDD results.

- **Discussion of result**

From the result, *Interventional* was the most frequent type and *Expanded Access* was the least frequent type of study 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 were 255945 and 277631 respectively), while Expanded Access trials remained at 69 over these 3 years.

With this result, we can deduce that people are more interested in Interventional trials, where they are part of treatment groups, receiving little or no intervention so that researchers can evaluate outcomes. This approach is more predictive. We can also infer that people 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 comma is the only delimiter in the Conditions column. It was also assumed that the column is not null.

- **PySpark implementation outline: DataFrame**

The clinical trial data was exploded, with each comma-separated Condition moved to separate rows.

The exploded data were grouped and counted, to determine the frequency of each Condition. Then, the resulting data was ordered by the count, in descending order, from highest to lowest, and the top five most frequent Conditions were 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)
```

- **HiveQL implementation outline**

This implementation consists of two queries, nested. In the inner query, the Conditions were split and exploded into separate rows, where the Conditions are not empty. In the outer query, the exploded Conditions were selected and grouped, then ordered in descending order. And the top five Conditions were returned.

```
-- QUESTION3: The top 5 Conditions with their frequencies
SELECT Conditions,COUNT(Conditions) AS Counts
FROM (
  SELECT explode(split(Conditions, ','))
  AS Conditions
  FROM clinicaltrial_${hivevar:year}_view
  WHERE Conditions!=''
  )
GROUP BY Conditions ORDER BY Counts DESC LIMIT 5;
```

- **PySpark implementation outline: RDD**

In this implementation, the clinical trial data were split, 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. The resulting data was then reduced by key and sorted in descending order. And 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)
```

- **Result on the submission**

Below are the results for each implementation.

```
# 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)
```

▸ (2) Spark Jobs

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

```
-- QUESTION3: The top 5 Conditions with their frequencies
SELECT Conditions,COUNT(Conditions) AS Counts
FROM (
  SELECT explode(split(Conditions, ','))
  AS Conditions
  FROM clinicaltrial_${hivevar:year}_view
  WHERE Conditions!=''
  )
GROUP BY Conditions ORDER BY Counts DESC LIMIT 5;
```

▸ (2) Spark Jobs

	Conditions	Counts
1	Carcinoma	13389
2	Diabetes Mellitus	11080
3	Neoplasms	9371
4	Breast Neoplasms	8640
5	Syndrome	8032

Showing all 5 rows.

Command took 3.11 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 09:37:59 on Cluster1

```
# 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)
```

▸ (3) Spark Jobs

```
Out[14]: [('Carcinoma', 13389),
 ('Diabetes Mellitus', 11080),
 ('Neoplasms', 9371),
 ('Breast Neoplasms', 8640),
 ('Syndrome', 8032)]
```

Command took 3.75 seconds -- by L.B.Amuda@edu.salford.ac.uk at 04/05/2022, 10:04:33 on Cluster1

Images of Dataframe, HiveQL, and RDD results.

- **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* remaining the least common, in the top 5.

Question 4

- **Assumptions made**

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

- **PySpark implementation outline: DataFrame**

The mesh data and exploded clinical trial data were joined, and the required columns were selected. Then the first 3 characters of the tree column (the root of the Conditions) was extracted and grouped. The resulting aggregate was then ordered in descending order with the five most frequent roots 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)
```

- **HiveQL implementation outline**

In this implementation, a temporary view was created for the exploded clinical trial data from question 3. Then, the mesh and exploded clinical trial views were joined, and the first 3 characters of the tree column were selected, grouped and ordered in descending order. And the 5 most frequent roots were returned.

```
-- CREATE A TEMPORARY VIEW FROM THE EXPLODED CLINICALTRIAL DATA
CREATE TEMPORARY VIEW explodedclinical_view
AS
SELECT *,explode(split(Conditions, ',')) AS ExplodedConditions
FROM clinicaltrial_${hivevar:year}_view WHERE Conditions!='';
```

OK

Command took 0.24 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 01:30:54 on Lab2.2

Cmd 29

```
-- QUESTION 4: The 5 most frequent roots from the hierarchy codes
SELECT tree,COUNT(tree) counts FROM
(
SELECT LEFT(m.tree,3) AS tree FROM mesh_view AS m
JOIN explodedclinical_view AS e
ON (m.term=e.ExplodedConditions)
)
GROUP BY tree ORDER BY counts DESC LIMIT 5;
```

- **PySpark implementation outline: RDD**

The mesh data was split by comma, to create a new RDD which is a pair of tree and term values. Also, a pair RDD was created from the exploded clinical trial data, setting the value to zero (0). The two RDDs were joined, and then split and mapped to return a pair RDD, containing the first 3 characters of the tree values as the key.

This resulting pair RDD was reduced by key to get a cumulation of the value and then sorted in descending order. The top 5 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)
```

- **Result on the submission**

Below are the results for each implementation.

```
# 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
```

```
# 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[49]: [('C04', 133091),
 ('C23', 124589),
 ('C01', 94293),
 ('C14', 88065),
 ('C10', 83894)]
```

Command took 8.85 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 00:58:57 on Lab2.2

```
-- CREATE A TEMPORARY VIEW FROM THE EXPLODED CLINICALTRIAL DATA
CREATE TEMPORARY VIEW explodedclinical_view
AS
SELECT *,explode(split(Conditions, ',')) AS ExplodedConditions
FROM clinicaltrial_${hivevar:year}_view WHERE Conditions!='';
```

OK

Command took 0.23 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 04:43:06 on Coursework Cluster

Cmd 29

```
-- QUESTION 4: The 5 most frequent roots from the hierarchy codes
SELECT tree,COUNT(tree) counts FROM
(
SELECT LEFT(m.tree,3) AS tree FROM mesh_view AS m
JOIN explodedclinical_view AS e
ON (m.term=e.ExplodedConditions)
)
GROUP BY tree ORDER BY counts DESC LIMIT 5;
```

▸ (4) Spark Jobs

	tree	counts
1	C04	143994
2	C23	136079
3	C01	106674
4	C14	94523
5	C10	92310

Showing all 5 rows.

Images of Dataframe, HiveQL, and RDD results.

- **Discussion of result**

Of the 5 most frequent roots in 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**. The year 2021 has roots with higher frequencies compared to the 2019 and 2020 data which also had the most common root **C04** at **123221** and **133091**, respectively.

With this information, we can predict a steady increase in the frequency of **C04** root causes in the following years. Based on these data, preventive or remedial methods can be developed to help reduce the occurrence of the Conditions caused by this root.

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**

The clinical trial and pharma dataframes were joined and the *leftanti* operation was performed, to return only columns in the clinical trial data that do not have matching records in the pharma data. The resulting data was filtered to remove the studies that have their *Status* value as **Active**, then grouped, counted and ordered by the *Sponsor* column in descending order, returning the top 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
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)
```

- **HiveQL implementation outline**

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 along with the matching rows found in the pharma view, where the *Parent_Company* column is not null and the *Status* is not **Active**.

In the outer query, the *Sponsor* column was selected, grouped and counted and then ordered in descending order, returning the top 10 most common Sponsors.

```
-- QUESTION 5: The 10 most common sponsors that are not pharmaceutical companies with the number of clinical trials they have sponsored
SELECT Sponsor,COUNT(Sponsor) AS counts FROM
(
SELECT * FROM clinicaltrial_${hivevar:year}_view c
LEFT OUTER JOIN pharma_view p
ON c.Sponsor=p.Parent_Company
WHERE p.Parent_Company IS NULL AND c.Status <> "Active"
)
GROUP BY Sponsor ORDER BY counts DESC LIMIT 10;
```

- **PySpark implementation outline: RDD**

The clinical trial data was mapped to return a pair RDD containing the *Sponsor* and *Status* values. The pharma data was split and mapped to create another pair RDD, to make sure both RDDs are in the form of (key, value) for each element of the RDDs when joining.

The two RDDs were joined, with *Left Outer join*, to return all the rows from the clinical trial data where the *Sponsor* values matched the *Parent_Companny* 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 resulting data is then filtered to return a pair RDD containing records where *Parent_Company* is **None** and *Status* is not **Active**.

This pair RDD was reduced by key to get a cumulation of the value 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)
```

- **Result on the submission**

Below are the results for each implementation.

```
# 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)
```

▸ (2) Spark Jobs

```
+----------------------------------------+-----+
|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 4.82 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 05:07:04 on Coursework Cluster

```
-- QUESTION 5: The 10 most common sponsors that are not pharmaceutical companies with the number of clinical trials they have sponsored
SELECT Sponsor,COUNT(Sponsor) AS counts FROM
(
SELECT * FROM clinicaltrial_${hivevar:year}_view c
LEFT OUTER JOIN pharma_view p
ON c.Sponsor=p.Parent_Company
WHERE p.Parent_Company IS NULL AND c.Status <> "Active"
)
GROUP BY Sponsor ORDER BY counts DESC LIMIT 10;
```

▸ (2) Spark Jobs

	Sponsor	counts
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

Showing all 10 rows.

Command took 3.73 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 04:43:06 on Coursework Cluster

```
# 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[18]: [('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 6.00 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 05:14:11 on Coursework Cluster

Images of Dataframe, HiveQL, and RDD results.

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

- **PySpark implementation outline: DataFrame**

The clinical trial data was filtered to return completed trials for the specified trial year (2021), and then the months were extracted from the *Completion* column, grouped and counted. The resulting aggregate was then ordered chronologically, based on the months in the year.

To visualize the result, the *matplotlib* library was used to create a bar chart that shows 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()
```

- **HiveQL implementation outline**

From the clinical trials data, the *Completion* column was selected and the first 3 characters (month) were extracted for completed trials within the specified clinical trial year, then they were grouped, counted, and ordered to give the number of the completed studies in each month, sorted chronologically.

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

```
-- QUESTION 6: Number of completed studies each month in a given year
-- LEFT(m.tree,3)
SELECT LEFT(Completion,3) AS Completion,
       COUNT(Completion) AS counts
FROM clinicaltrial_${hivevar:year}_view
WHERE Status=="Completed" AND Completion LIKE '%${hivevar:year}'
GROUP BY Completion
ORDER BY (unix_timestamp(Completion,'MMM'),'MM');
```

- **PySpark implementation outline: RDD**

For this implementation, the clinical trial data was filtered to return completed trials for the given clinical trial year. The completed trials were then split to extract the months from the *Completion* values and a pair RDD was created. This pair RDD was then reduced by key to return the number of completed trials for each month.

To sort the months chronologically, a month's dictionary was created to convert month names to month numbers. Each month from the resulting data was mapped to the corresponding name in the dictionary to obtain their expected month number and then sorted.

```
# 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()
```

- **Result on the submission**

Below are the results for each implementation.

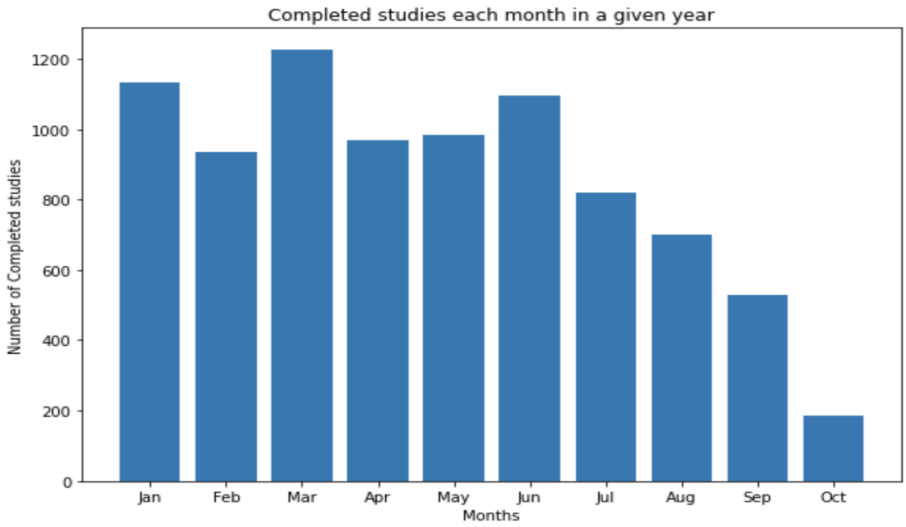
```
# QUESTION 6 - Number of completed studies each month in a given year
completedStudiesDF = clinicaltrialDF.filter(clinicaltrialDF.Status=="Completed").filter(col("Completion").endswith(clinicaltrial_year)).\
                                withColumn('Completion', substring('Completion', 1,3)).groupBy("Completion").count()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

```
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



Command took 6.89 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 05:07:04 on Coursework Cluster

Images of the Dataframe result

```
-- QUESTION 6: Number of completed studies each month in a given year
-- LEFT(m.tree,3)
SELECT LEFT(Completion,3) AS Completion,
       COUNT(Completion) AS counts
FROM clinicaltrial_${hivevar:year}_view
WHERE Status=="Completed" AND Completion LIKE '%${hivevar:year}'
GROUP BY Completion
ORDER BY (unix_timestamp(Completion,'MMM'),'MM');
```

▸ (2) Spark Jobs

	Completion	counts
1	Jan	1131
2	Feb	934
3	Mar	1227
4	Apr	967
5	May	984
6	Jun	1094

Showing all 10 rows.

```
-- Create a temporary view from exploded clinical trial data
CREATE TEMP VIEW completedstudies_view
AS

SELECT SUBSTRING(Completion,1,3) AS Completion,
       COUNT(Completion) AS counts
FROM clinicaltrial_${hivevar:year}_view
WHERE Status=="Completed" AND Completion LIKE concat("%",${hivevar:year})
GROUP BY Completion
ORDER BY (unix_timestamp(Completion,'MMM'),'MM');
```

OK

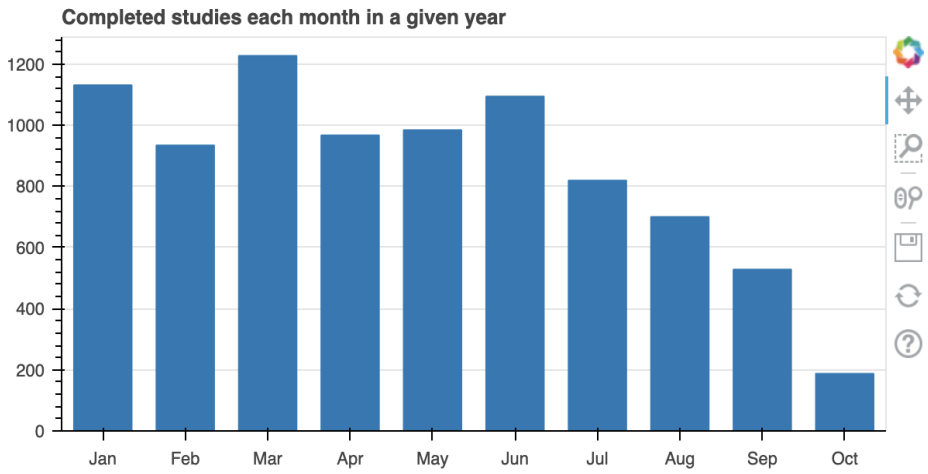
Command took 0.10 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 04:43:06 on Coursework Cluster

Cmd 34

```
%python
%pip install bokeh
```

```
%python
# QUESTION 6: Plot number of completed studies each month in a given year

from bokeh.io import output_file, show
from bokeh.plotting import figure
from bokeh.embed import file_html
from bokeh.resources import CDN

months = spark.sql("SELECT * FROM completedstudies_view").select("Completion").rdd.map(lambda row: row["Completion"]).collect()
counts = spark.sql("SELECT * FROM completedstudies_view").select("counts").rdd.map(lambda row: row["counts"]).collect()

p = figure(x_range=months, height=300, title="Completed studies each month in a given year")

p.vbar(x=months, top=counts, width=0.7)

p.xgrid.grid_line_color = None
p.y_range.start = 0

html = file_html(p, CDN, "plot")
displayHTML(html)
```

▸ (8) Spark Jobs



Command took 5.75 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 04:43:06 on Coursework Cluster

Images of HiveQL result

```
# 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()
```

▸ (3) Spark Jobs

```
Out[19]: [('Jan', 1131),
 ('Feb', 934),
 ('Mar', 1227),
 ('Apr', 967),
 ('May', 984),
 ('Jun', 1094),
 ('Jul', 819),
 ('Aug', 700),
 ('Sep', 528),
 ('Oct', 187)]
```

Command took 2.63 seconds -- by L.B.Amuda@edu.salford.ac.uk at 05/05/2022, 05:14:11 on Coursework Cluster

Images of RDD result

- **Discussion of result**

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

Based on the data obtained from the previous years, 2019 and 2020, *December* had the highest number of studies (**2690** and **2084**, respectively). Hence, we can predict that the number of studies for the year 2021 would increase and be highest by December.

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