Analysis of Clinical Trial Data

University of Salford MANCHESTER

ASIF JAVED

@00653639

Contents

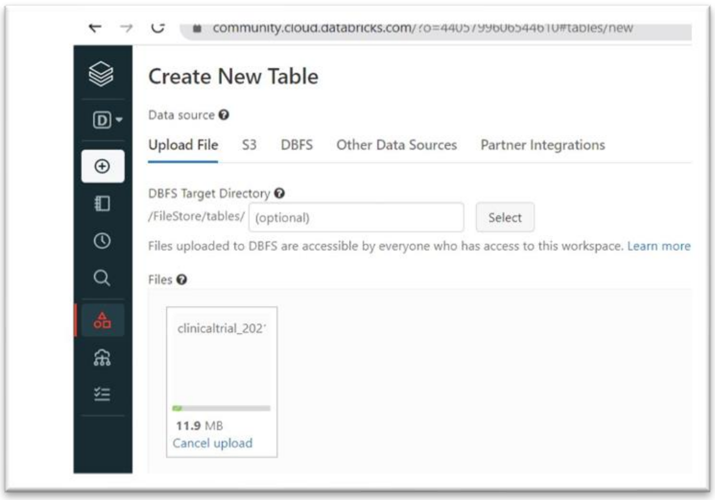
Analysis of clinical trial data

Part 1 – Setup required for analysis

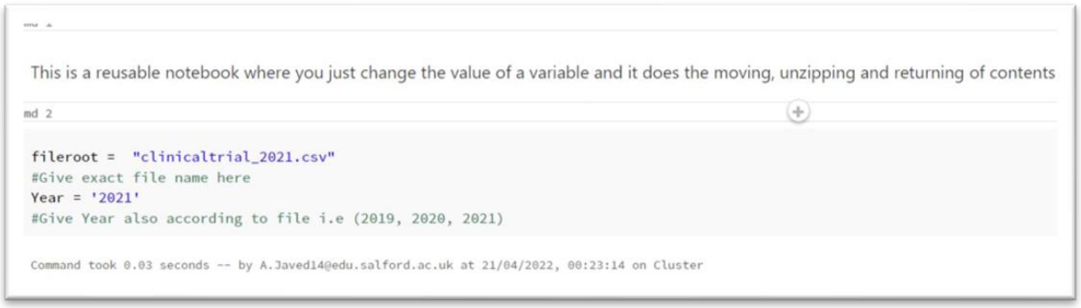
To analyze the clinical trial dataset, I have used Databricks to work with Pyspark and HQL using the Python & SQL notebooks to execute the code in a web based interactive interface. Different python libraries are used to extract the conclusions. I have also attached my code for each answer from Q1 to Q6. Each code segment is written with comments to explain the answers.

Part 2 – Data Cleaning and Preparation

Dataset for clinical trial was provided in compressed csv file. To use the dataset in databricks environment I have uploaded the dataset in databricks FileStore>tables directory.

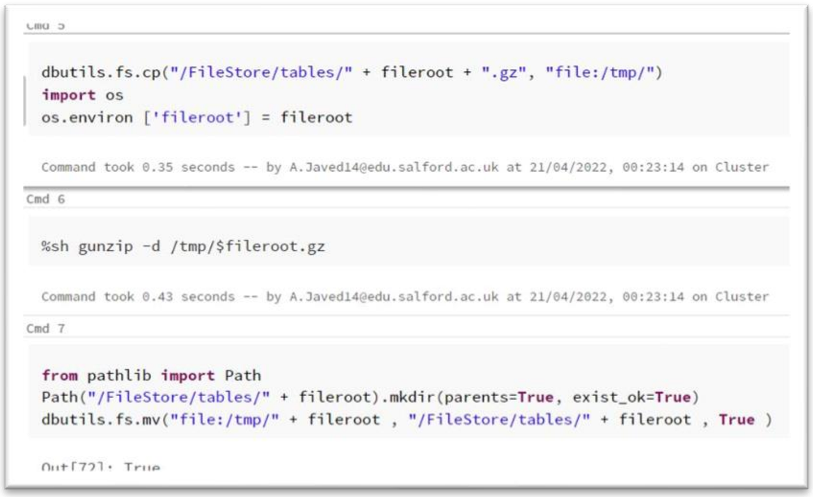


After data uploading, I have uncompressed the dataset to csv format for performing the actions. To uncompressing the data I have used the unix command in python notebook and first I moved the compressed file to databricks FileStore/tmp directory and after uncompressing I again moved the file to its original directory dbfs/tables directory. For this whole process I have I have used a fileroot variable which performed the taking file path, moving and uncompressing actions.


This is a reusable notebook where you just change the value of a variable and it does the moving, unzipping and returning of contents

```
fileroot =  "clinicaltrial_2021.csv"
#Give exact file name here
Year = '2021'
#Give Year also according to file i.e (2019, 2020, 2021)
```

```
Command took 0.03 seconds -- by A.Javed14@edu.salford.ac.uk at 21/04/2022, 00:23:14 on Cluster
```


```
Cmd 5

dbutils.fs.cp("/FileStore/tables/" + fileroot + ".gz", "file:/tmp/")
import os
os.environ ['fileroot'] = fileroot

Command took 0.35 seconds -- by A.Javed14@edu.salford.ac.uk at 21/04/2022, 00:23:14 on Cluster
Cmd 6

%sh gunzip -d /tmp/$fileroot.gz

Command took 0.43 seconds -- by A.Javed14@edu.salford.ac.uk at 21/04/2022, 00:23:14 on Cluster
Cmd 7

from pathlib import Path
Path("/FileStore/tables/" + fileroot).mkdir(parents=True, exist_ok=True)
dbutils.fs.mv("file:/tmp/" + fileroot , "/FileStore/tables/" + fileroot , True )

Out[72]: True
```

These steps are done in PySpark both RDD & Dataframe. To abstract the data, data cleaning of the csv file was compulsory. So, I have I went through the dataset and removed/skipped the header and split the dataset using delimiters pipe.
For running the dataset throughout the file I have declared two variables one (fileroot) for file name and other (Year) for year as dataset year for relevant and accurate results.

Part 3 – Problem Answers

1) Problem Statement I :

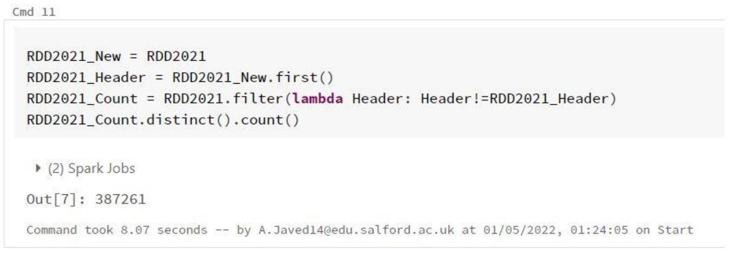
Answer:
(dataset used to produce answer: clinicaltrial_2021.csv)
Count: 387261 #Total number of distinct studies

Code Segments Used in RDD Implementation:

According to problem statement dataset has studies which involves the clinical trial analysis. To extract the distinct studies from the dataset I have assigned the dataset to the RDD2021 and for accurate results I have skipped the first row as it's the header by filtering the RDD2021. Then I have taken the distinct count of the dataset studies.

```
RDD2021= sc.textFile('FileStore/tables/'+fileroot)
RDD2021_New = RDD2021
RDD2021_Header = RDD2021_New.first()            #assiging first row to RDD
RDD2021_Count = RDD2021.filter(lambda Header: Header!=RDD2021_Header) #skipping the header by filtering RDD
RDD2021_Count.distinct().count()                #taking distinct count from the RDD data
```

Screenshot:

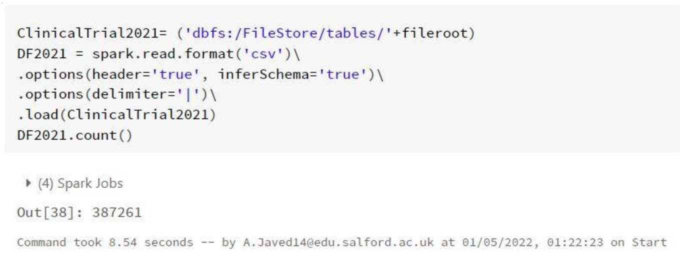

```
Cmd 11

RDD2021_New = RDD2021
RDD2021_Header = RDD2021_New.first()
RDD2021_Count = RDD2021.filter(lambda Header: Header!=RDD2021_Header)
RDD2021_Count.distinct().count()

▸ (2) Spark Jobs
Out[7]: 387261
Command took 8.07 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:24:05 on Start
```

Code Segments Used in DF Implementation:

I have loaded the file in DF2021 and made further steps for header, schema and delimiter pipe. Then took the count for studies and results are accurate as per RDD.

```
ClinicalTrial2021= ('dbfs:/FileStore/tables/'+fileroot)
DF2021 = spark.read.format('csv')\
.options(header='true', inferSchema='true')\
.options(delimiter='|')\
.load(ClinicalTrial2021)                    #loading and ready the dataset in dataframe
DF2021.count()                              #taking the count of dataset
```

Screenshot:



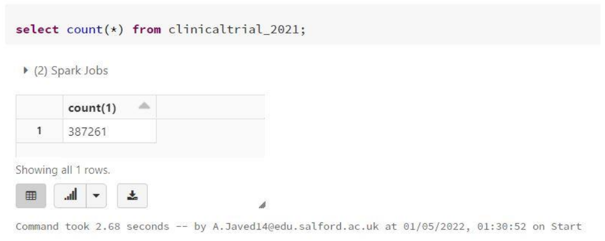
```
ClinicalTrial2021= ('dbfs:/FileStore/tables/'+fileroot)
DF2021 = spark.read.format('csv')\
.options(header='true', inferSchema='true')\
.options(delimiter='|')\
.load(ClinicalTrial2021)
DF2021.count()

(4) Spark Jobs
Out[38]: 387261
Command took 8.54 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:22:23 on Start
```

Code Segments Used in HQL Implementation:

I have made an external table clinicaltrial_2021 by specifying the schema and inserted the data from the dataset. And choose the studies count from the table.

```
CREATE EXTERNAL TABLE IF NOT EXISTS clinicaltrial_2021   #Creating table using the clinicaltrial_2021 datset
( Id STRING ,
Sponsor STRING ,
Status STRING ,
Start STRING ,
Completion STRING ,
Type STRING ,
Submission STRING ,
Conditions STRING ,
Interventions STRING )
USING CSV
OPTIONS (path 'dbfs:/FileStore/tables/$fileroot',
delimiter "|",
header "true");

select Distinct count(*) from clinicaltrial_2021;                    #Showing count of dataset
```

Screenshot:



```
select count(*) from clinicaltrial_2021;

(2) Spark Jobs
```

	count(1)
1	387261

```
Showing all 1 rows.
Command took 2.68 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:30:52 on Start
```

2) Problem Statement II :

<u>Answer</u>:

(dataset used to produce answer: clinicaltrial_2021.csv)

[('Interventional', **301472**),

('Observational', **77540**),

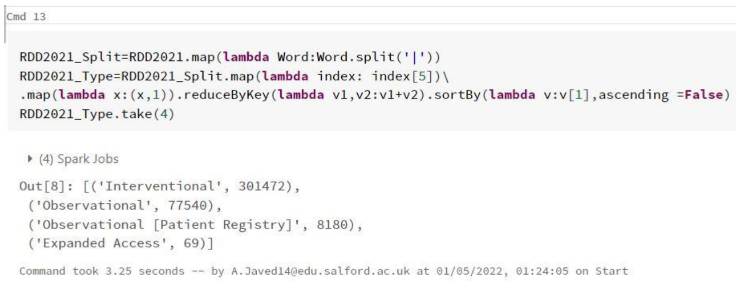
('Observational [Patient Registry]', **8180**),
('Expanded Access', **69**)]

Code Segments Used in RDD Implementation:

I have splitted the RDD2021_Split by pipe '|'. Then I have taken the 6th index column name 'type' to check the types with frequencies. Then I used reduceByKey function for grouping the same keys and adding their values. As per requirement I sorted the final results in descending order for most frequent types at top.

```
RDD2021_Split=RDD2021.map(lambda Word:Word.split('|'))            #RDD Splitting on pipe
RDD2021_Type=RDD2021_Split.map(lambda index: index[5])\
.map(lambda x:(x,1)).reduceByKey(lambda v1,v2:v1+v2)\             #reducing same keys by adding their values
.sortBy(lambda v:v[1],ascending =False)                           #sorting RDD
RDD2021_Type.take(4)
```

Screenshot:



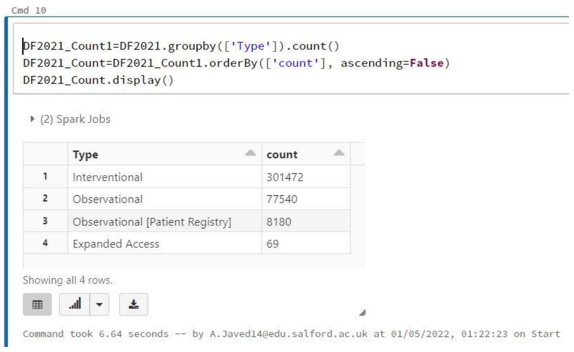
```
Cmd 13

RDD2021_Split=RDD2021.map(lambda Word:Word.split('|'))
RDD2021_Type=RDD2021_Split.map(lambda index: index[5])\
.map(lambda x:(x,1)).reduceByKey(lambda v1,v2:v1+v2).sortBy(lambda v:v[1],ascending =False)
RDD2021_Type.take(4)

▸ (4) Spark Jobs

Out[8]: [('Interventional', 301472),
 ('Observational', 77540),
 ('Observational [Patient Registry]', 8180),
 ('Expanded Access', 69)]

Command took 3.25 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:24:05 on Start
```

Code Segments Used in DF Implementation:

I grouped the Type column by taking their count and then ordered it in descending.

```
DF2021_Count1=DF2021.groupby(['Type']).count()                          #grouping the same keys and taking count
DF2021_Count=DF2021_Count1.orderBy(['count'], ascending=False)          #sorting the dataframe
DF2021_Count.display()
```

Screenshot:



```
Cmd 10

DF2021_Count1=DF2021.groupby(['Type']).count()
DF2021_Count=DF2021_Count1.orderBy(['count'], ascending=False)
DF2021_Count.display()

▸ (2) Spark Jobs
```

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

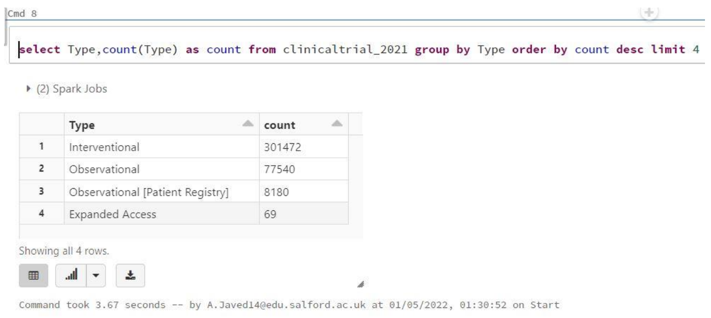
```
Showing all 4 rows.

Command took 6.64 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:22:23 on Start
```

Code Segments Used in HQL Implementation:

I made a query of clinicaltrial_2021 table and grouped all the types from type column by taking the count and then ordered it as per desired output.

```
select Type, count(Type) as count from clinicaltrial_2021          #selecting the type column and its count
group by Type
order by count
desc limit 4
```

Screenshot:



```
Cmd 8

select Type,count(Type) as count from clinicaltrial_2021 group by Type order by count desc limit 4

(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 3.67 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:30:52 on Start
```

3) Problem Statement III :

<u>Answer</u>:

(dataset used to produce answer: clinicaltrial_2021.csv)

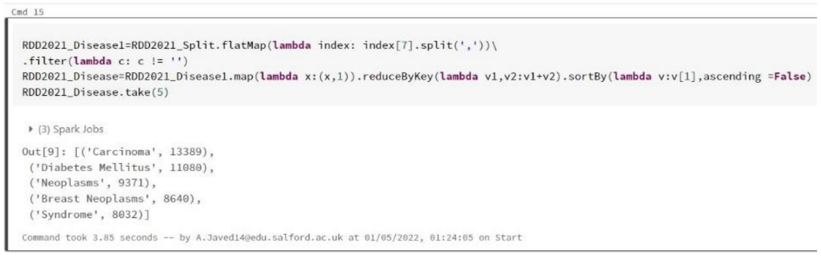
[('Carcinoma', 13389),
('Diabetes Mellitus', 11080),
('Neoplasms', 9371),
('Breast Neoplasms', 8640),
('Syndrome', 8032)]

Code Segments Used in RDD Implementation:

For problem statement 3 I have mapped the each value from conditions column and splitted it against comma delimiter and skipped the nulled values by filtering the RDD. The I have mapped the RDD and assigned 1 to each key. reduceByKey function is used for grouping the same keys and adding their values and sorted the RDD on values in descending order.

```
RDD2021_Disease1=RDD2021_Split.flatMap(lambda index: index[7].split(','))\      #splitting 8th column on comma
.filter(lambda c: c != '')                                                     #skipping null values
RDD2021_Disease=RDD2021_Disease1.map(lambda x:(x,1))\                          #assigning 1 value to each key
.reduceByKey(lambda v1,v2:v1+v2)\                                              #reducing keys by adding their values
.sortBy(lambda v:v[1],ascending =False)                                        #sorting the RDD
RDD2021_Disease.take(5)
```

Screenshot:



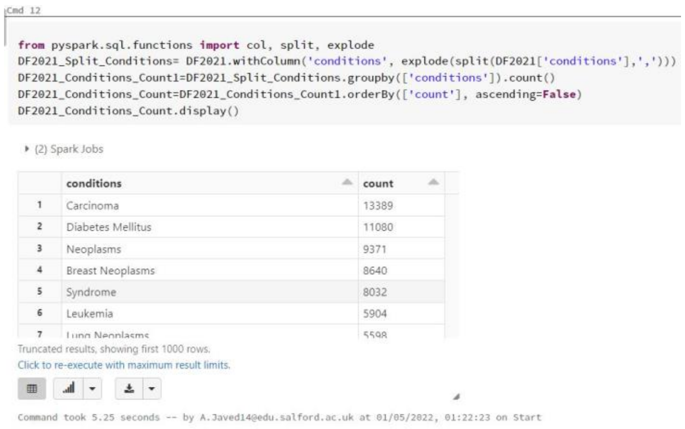
```
Cmd 15

RDD2021_Disease1=RDD2021_Split.flatMap(lambda index: index[7].split(','))\
.filter(lambda c: c != '')
RDD2021_Disease=RDD2021_Disease1.map(lambda x:(x,1)).reduceByKey(lambda v1,v2:v1+v2).sortBy(lambda v:v[1],ascending =False)
RDD2021_Disease.take(5)

(3) Spark Jobs

Out[9]: [('Carcinoma', 13389),
 ('Diabetes Mellitus', 11080),
 ('Neoplasms', 9371),
 ('Breast Neoplasms', 8640),
 ('Syndrome', 8032)]

Command took 3.85 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:24:05 on Start
```

Code Segments Used in DF Implementation:

I splitted the column condition on comma and used the explode function to list all the items on row from the column to new column named conditions. Then I grouped the conditions and make the order descending of the conditions count.

```
from pyspark.sql.functions import col, split, explode
DF2021_Split_Conditions= DF2021.withColumn('conditions', explode(split(DF2021['conditions'],',')))
#splitting the column and exploding the values
DF2021_Conditions_Count1=DF2021_Split_Conditions.groupby(['conditions']).count()
#grouping by same type and taking count
DF2021_Conditions_Count=DF2021_Conditions_Count1.orderBy(['count'], ascending=False) #sorting/ordering
DF2021_Conditions_Count.display()
```

Screenshot:



Cmd 12

```
from pyspark.sql.functions import col, split, explode
DF2021_Split_Conditions= DF2021.withColumn('conditions', explode(split(DF2021['conditions'],',')))
DF2021_Conditions_Count1=DF2021_Split_Conditions.groupby(['conditions']).count()
DF2021_Conditions_Count=DF2021_Conditions_Count1.orderBy(['count'], ascending=False)
DF2021_Conditions_Count.display()
```

(2) Spark Jobs

	conditions	count
1	Carcinoma	13389
2	Diabetes Mellitus	11080
3	Neoplasms	9371
4	Breast Neoplasms	8640
5	Syndrome	8032
6	Leukemia	5904
7	Lung Neoplasms	5598

Truncated results, showing first 1000 rows.

Click to re-execute with maximum result limits.

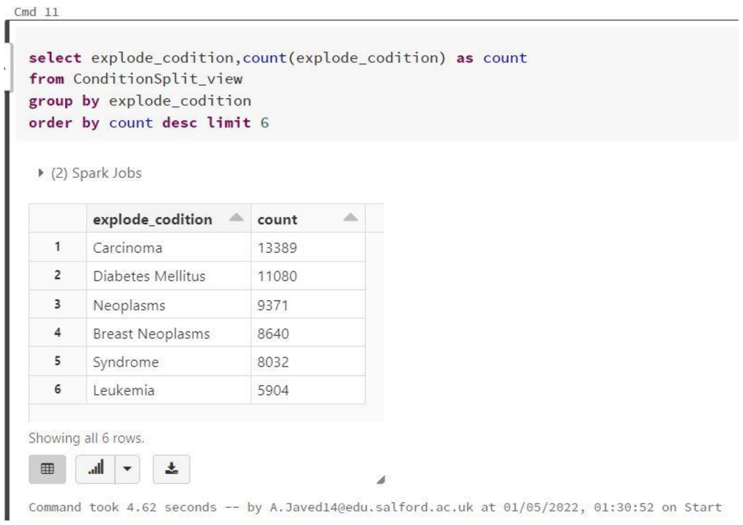
```
Command took 5.25 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:22:23 on Start
```

Code Segments Used in HQL Implementation:

I created a view in which I splitted the column and used explode function for converting the list into row in a new column named explode_condition. I skipped the nulled values by checking the length of each row and make a count by grouping the conditions and sorted it as per desired results.

```
CREATE OR REPLACE VIEW  ConditionSplit_view
AS
SELECT id, explode_codition
FROM clinicaltrial_2021
lateral VIEW explode(split(Conditions,",")) condition_view AS explode_codition
where   length(explode_codition) > 0 ;

select explode_codition,count(explode_codition) as count
from ConditionSplit_view
group by explode_codition
order by count desc limit 6
```

Screenshot:



Cmd 11

```
select explode_codition,count(explode_codition) as count
from ConditionSplit_view
group by explode_codition
order by count desc limit 6
```

(2) Spark Jobs

	explode_codition	count
1	Carcinoma	13389
2	Diabetes Mellitus	11080
3	Neoplasms	9371
4	Breast Neoplasms	8640
5	Syndrome	8032
6	Leukemia	5904

Showing all 6 rows.

```
Command took 4.62 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:30:52 on Start
```

4) Problem Statement IV :

<u>Answer</u>:

(dataset used to produce answer: clinicaltrial_2021.csv and Mesh.csv)

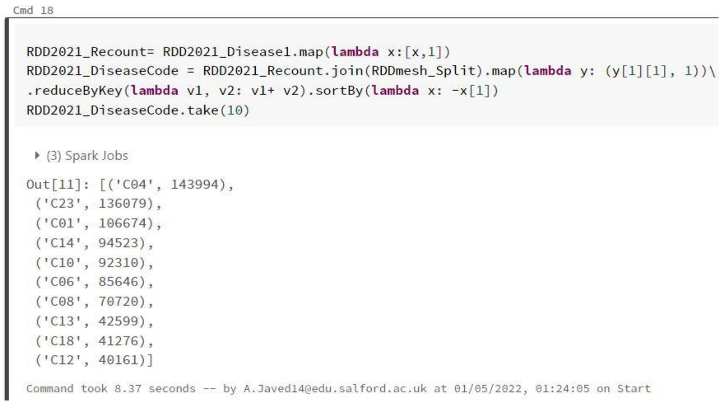
```
[('C04', 143994),
('C23', 136079),
('C01', 106674),
('C14', 94523),
('C10', 92310),
('C06', 85646),
('C08', 70720),
('C13', 42599),
('C18', 41276),
('C12', 40161)]
```

Code Segments Used in RDD Implementation:

Each condition has its own hierarchy code. To get access with it I have used mesh file and cleaned it by skipping the header and splitting it on comma and full stop to get the desired part of the code related to condition. Then I joined the clinicaltrial_20211's conditions with cleaned mesh file to join conditions and codes. Then I used reducebykey to group the similar keys and added their values. And sorted it in descending order to show the most frequent at top.

```
RDDmesh_Header = RDDmesh.first()
RDDmesh_Header_Remove = RDDmesh.filter(lambda Header: Header!=RDDmesh_Header)
RDDmesh_Split = RDDmesh_Header_Remove.map(lambda x:(x.split(",")[0], x.split(",")[1].split(".")[0]))
RDDmesh_Split.take(2)

RDD2021_Recount= RDD2021_Disease1.map( lambda x:[x,1] )
RDD2021_DiseaseCode = RDD2021_Recount.join(RDDmesh_Split).map(lambda y: (y[1][1], 1))\
.reduceByKey(lambda v1, v2: v1+ v2).sortBy( lambda x: -x[1] )
RDD2021_DiseaseCode.take(10)
```

Screenshot:



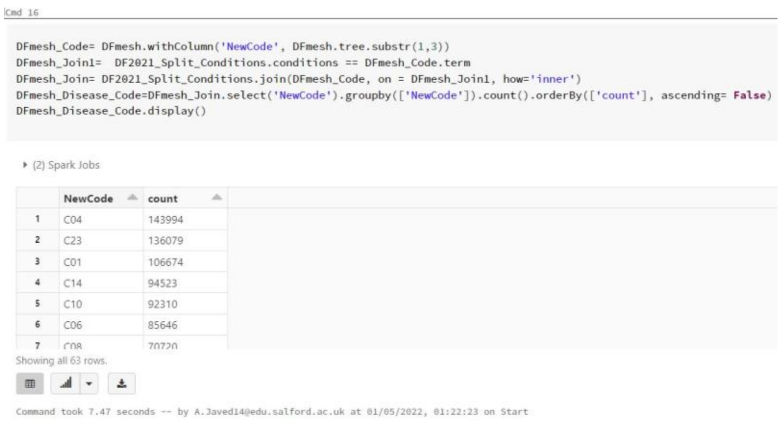
```
Cmd 18

RDD2021_Recount= RDD2021_Disease1.map(lambda x:[x,1])
RDD2021_DiseaseCode = RDD2021_Recount.join(RDDmesh_Split).map(lambda y: (y[1][1], 1))\
.reduceByKey(lambda v1, v2: v1+ v2).sortBy(lambda x: -x[1])
RDD2021_DiseaseCode.take(10)

▸ (3) Spark Jobs

Out[11]: [('C04', 143994),
 ('C23', 136079),
 ('C01', 106674),
 ('C14', 94523),
 ('C10', 92310),
 ('C06', 85646),
 ('C08', 70720),
 ('C13', 42599),
 ('C18', 41276),
 ('C12', 40161)]

Command took 8.37 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:24:05 on Start
```

Code Segments Used in DF Implementation:

I have loaded the mesh file into DFmesh by defining the schema and delimiter. Then to take code part from mesh file I have used substr function by taking first three letters from the code with a new column name NewCode. Then I compared the condition file and mesh file on the basis of conditions and joined both dataframes. Now each condition has its own code and then I have group them and made the count and sorted it in descending order.

```
mesh=('dbfs:/FileStore/tables/mesh.csv')
DFmesh = spark.read.format('csv')\
.options(header='true', inferSchema='true')\
.options(delimiter=',')\
.load(mesh)
display(DFmesh)

DFmesh_Code= DFmesh.withColumn('NewCode', DFmesh.tree.substr(1,3))
DFmesh_Join1=  DF2021_Split_Conditions.conditions == DFmesh_Code.term
DFmesh_Join= DF2021_Split_Conditions.join(DFmesh_Code, on = DFmesh_Join1, how='inner')
#joining 2 DFs to get code
DFmesh_Disease_Code=DFmesh_Join.select('NewCode').groupby(['NewCode']).count().orderBy(['count'],
ascending= False)
DFmesh_Disease_Code.display()
```

Screenshot:



Cmd 16

```
DFmesh_Code= DFmesh.withColumn('NewCode', DFmesh.tree.substr(1,3))
DFmesh_Join1=  DF2021_Split_Conditions.conditions == DFmesh_Code.term
DFmesh_Join= DF2021_Split_Conditions.join(DFmesh_Code, on = DFmesh_Join1, how='inner')
DFmesh_Disease_Code=DFmesh_Join.select('NewCode').groupby(['NewCode']).count().orderBy(['count'], ascending= False)
DFmesh_Disease_Code.display()
```

▸ (2) Spark Jobs

	NewCode	count
1	C04	143994
2	C23	136079
3	C01	106674
4	C14	94523
5	C10	92310
6	C06	85646
7	C08	70720

Showing all 63 rows.

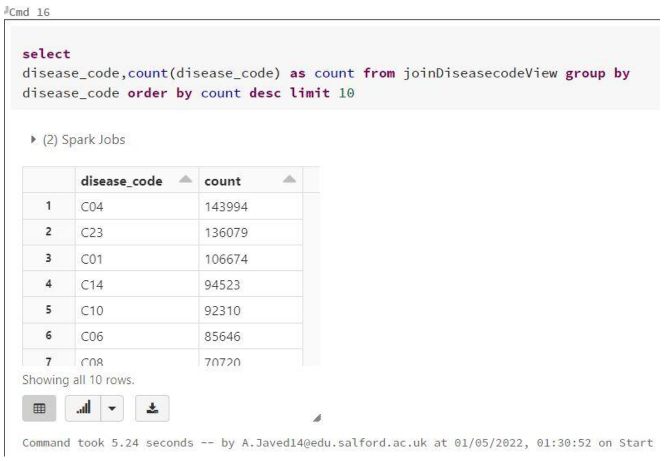
```
Command took 7.47 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:22:23 on Start
```

Code Segments Used in HQL Implementation:

I have created table using the mesh file. Then I created a view from code by using substring function . then I joined the condition column and disease code where both conditions are same. Then I made a count, grouped by same keys and sorted it according to requirement.

```
CREATE External TABLE IF NOT EXISTS mesh (     #creating table from mesh file
term STRING ,
tree STRING )
USING CSV
OPTIONS ( path "dbfs:/FileStore/tables/mesh.csv",
delimiter ",",
header "true" );

CREATE OR REPLACE VIEW  disease_code As
select term,tree,substr( tree,1,3 ) as disease_code from mesh     #using substring function to get first 3 letters of code

CREATE OR REPLACE VIEW  joinDiseasecodeView As
SELECT c.id,c.explode_codition,d.disease_code
FROM ConditionSplit_view as c
JOIN disease_code as d ON c.explode_codition=d.term

select disease_code, count(disease_code) as count
from joinDiseasecodeView
group by disease_code
order by count desc limit 10
```

Screenshot:



Cmd 16

```
select
disease_code,count(disease_code) as count from joinDiseasecodeView group by
disease_code order by count desc limit 10
```

▸ (2) Spark Jobs

	disease_code	count
1	C04	143994
2	C23	136079
3	C01	106674
4	C14	94523
5	C10	92310
6	C06	85646
7	C08	70720

Showing all 10 rows.

```
Command took 5.24 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:30:52 on Start
```

5) Problem Statement V

Answer:

(dataset used to produce answer: clinicaltrial_2021.csv and Pharma.csv)

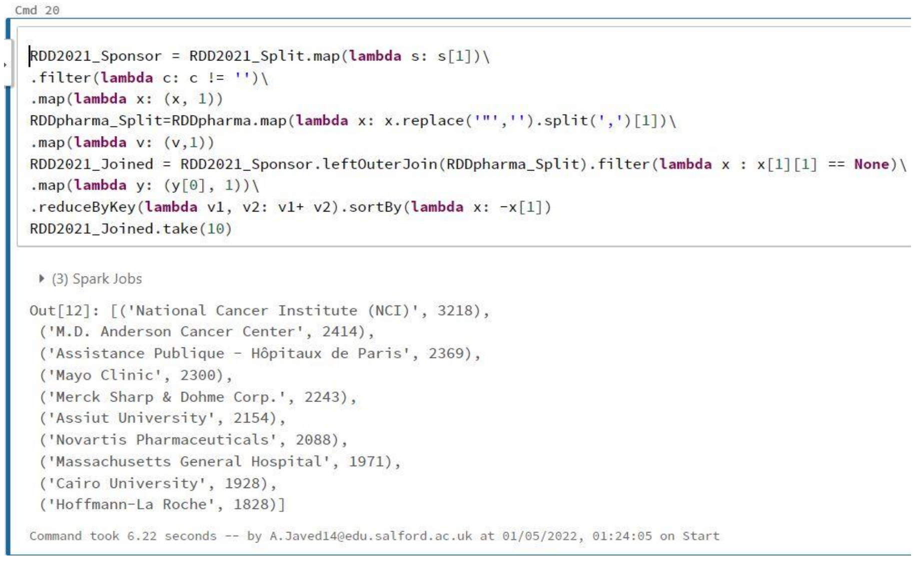
```
[('National Cancer Institute (NCI)', 3218),
('M.D. Anderson Cancer Center', 2414),
('Assistance Publique - Hôpitaux de Paris', 2369),
('Mayo Clinic', 2300),
('Merck Sharp & Dohme Corp.', 2243),
('Assiut University', 2154),
('Novartis Pharmaceuticals', 2088),
('Massachusetts General Hospital', 1971),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

Code Segments Used in RDD Implementation:

In this implementation Pharma file has all the pharmaceutical companies list but in clinical trial file there are some sponsors which were not pharmaceutical company. So I used the sponsor column from clinicaltrial file and skipped the nulled fields by filtering the RDD. I used leftouterjoin function to pick the unmatched sponsor companies from clinicaltrial file which were not in Pharma file. Then I again filtered the resulted RDD to skipped the null fields. Reducebykey function and count function used to find the frequency of each sponsor for trial study.

```
RDD2021_Sponsor = RDD2021_Split.map(lambda s: s[1]) \
.filter(lambda c: c != '') \
.map(lambda x: (x, 1))
RDDpharma_Split=RDDpharma.map(lambda x: x.replace('"','').split(',')[1]) \ #splitting the rdd's 2nd index on comma
.map(lambda v: (v,1))
RDD2021_Joined = RDD2021_Sponsor.leftOuterJoin(RDDpharma_Split).filter(lambda x : x[1][1] == None) \
.map(lambda y: (y[0], 1)) \        #joining 2 RDDs by left out join function to get values of 1st Rdd which are not in 2nd
.reduceByKey(lambda v1, v2: v1+ v2).sortBy(lambda x: -x[1]) #reducing keys by adding its values and sorting
RDD2021_Joined.take(10)
```

Screenshot:



Cmd 20

```
RDD2021_Sponsor = RDD2021_Split.map(lambda s: s[1])\
.filter(lambda c: c != '')\
.map(lambda x: (x, 1))
RDDpharma_Split=RDDpharma.map(lambda x: x.replace('"','').split(',')[1])\
.map(lambda v: (v,1))
RDD2021_Joined = RDD2021_Sponsor.leftOuterJoin(RDDpharma_Split).filter(lambda x : x[1][1] == None)\
.map(lambda y: (y[0], 1))\
.reduceByKey(lambda v1, v2: v1+ v2).sortBy(lambda x: -x[1])
RDD2021_Joined.take(10)
```

▸ (3) Spark Jobs

```
Out[12]: [('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)]
```

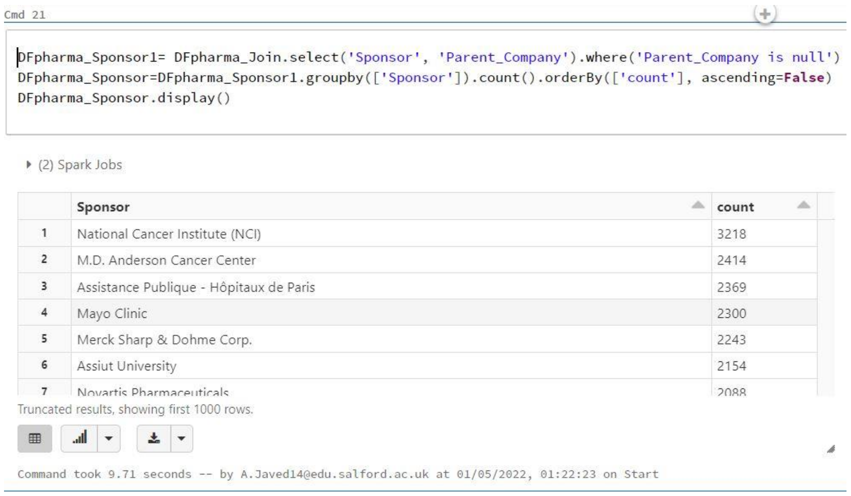
Command took 6.22 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:24:05 on Start

Code Segments Used in DF Implementation:

I loaded the pharma file and read it as csv file by defining the schema and delimiter.Then compared and joined the clinicaltrial's sponsor column and pharma file's parent company column. By using the where statement I checked the values where sponsor is not matched with parents company column. Then I grouped by sponsor and order it to extract the desired results.

```
pharma=('dbfs:/FileStore/tables/pharma.csv')
DFpharma=spark.read.format('csv') \
.options(header='true', inferSchema='true') \
.options(delimiter=',') \
.load(pharma)
DFpharma.display()

DFpharma_Join1=DF2021.Sponsor == DFpharma.Parent_Company
DFpharma_Join= DF2021.join(DFpharma, on = DFpharma_Join1, how='left')
DFpharma_Join.display()

DFpharma_Sponsor1= DFpharma_Join.select('Sponsor', 'Parent_Company').where('Parent_Company is null')
DFpharma_Sponsor=DFpharma_Sponsor1.groupby(['Sponsor']).count().orderBy(['count'], ascending=False)
DFpharma_Sponsor.display()
```

Screenshot:



Cmd 21

```
DFpharma_Sponsor1= DFpharma_Join.select('Sponsor', 'Parent_Company').where('Parent_Company is null')
DFpharma_Sponsor=DFpharma_Sponsor1.groupby(['Sponsor']).count().orderBy(['count'], ascending=False)
DFpharma_Sponsor.display()
```

(2) Spark Jobs

	Sponsor	count
1	National Cancer Institute (NCI)	3218
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2369
4	Mayo Clinic	2300
5	Merck Sharp & Dohme Corp.	2243
6	Assiut University	2154
7	Novartis Pharmaceuticals	2088

Truncated results, showing first 1000 rows.

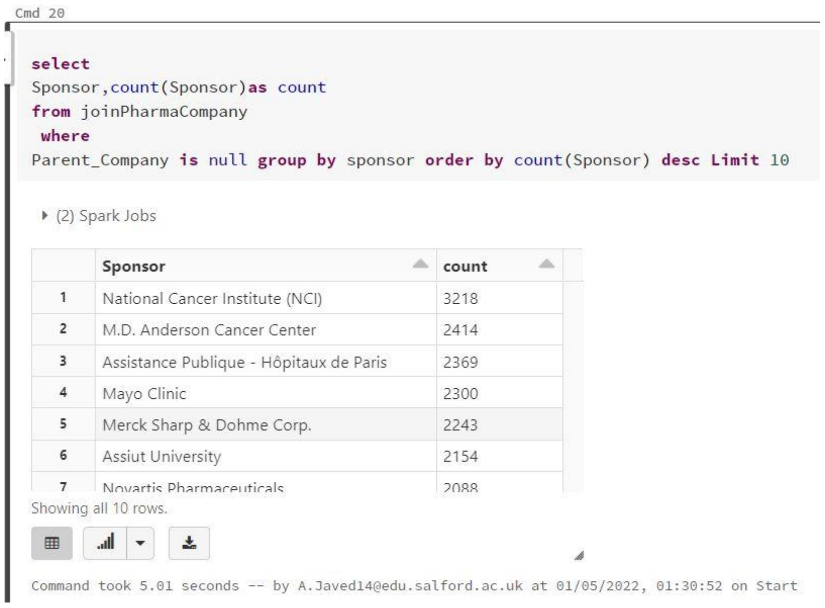
```
Command took 9.71 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:22:23 on Start
```

Code Segments Used in HQL Implementation:

I have created table using the pharma file. Then I created to view to check the different values by using the leftouterjoin function which were not in pharma file but in clinicaltrial file’s sponsor column. By grouping the sponsor and making the count I got the desired output as per client’s requirement.

```
CREATE External TABLE IF NOT EXISTS pharma (            #Creating the 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 )
USING CSV
OPTIONS (path "dbfs:/FileStore/tables/pharma.csv",
delimiter ",",
header "true" ) ;

CREATE OR REPLACE VIEW  joinPharmaCompany           #creating view
AS
select c.*,p.Parent_Company
from clinicaltrial_2021 as c LEFT OUTER JOIN pharma p
#by using leftouterjoin function taking first table's column values which are not in 2nd table's column

ON (c.Sponsor=p.Parent_Company)      #condition for joining

select
Sponsor,count(Sponsor)as count
from joinPharmaCompany
where
Parent_Company is null group by sponsor order by count(Sponsor) desc Limit 10
```

Screenshot:


Cmd 20

```
select
Sponsor,count(Sponsor)as count
from joinPharmaCompany
 where
Parent_Company is null group by sponsor order by count(Sponsor) desc Limit 10
```

(2) Spark Jobs

	Sponsor	count
1	National Cancer Institute (NCI)	3218
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2369
4	Mayo Clinic	2300
5	Merck Sharp & Dohme Corp.	2243
6	Assiut University	2154
7	Novartis Pharmaceuticals	2088

Showing all 10 rows.

Command took 5.01 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:30:52 on Start

6) Problem Statement VI :

Answer:

(dataset used to produce answer: clinicaltrial_2021.csv, Mesh.csv and Pharma.csv)

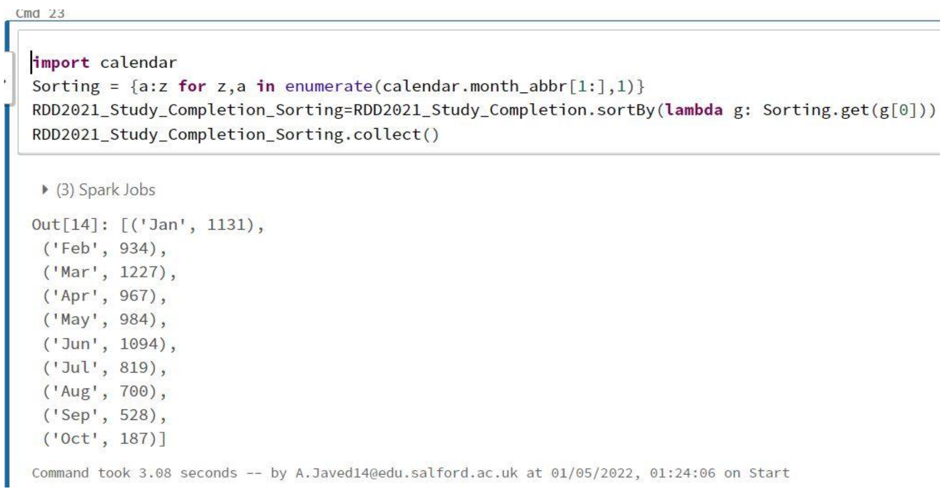
```
[('Jan', 1131),
('Feb', 934),
('Mar', 1227),
('Apr', 967),
('May', 984),
('Jun', 1094),
('Jul', 819),
('Aug', 700),
('Sep', 528),
('Oct', 187)]
```

Code Segments Used in RDD Implementation:

For this problem statement in RDD implementation I have used the filtered RDD without header and splitted the completion column to separate the month and year. Also I skipped the nulled values. Then I filtered the resulted RDD to take the data where study status was completed and year were 2021. The used the reducebykey function to take the completed study count.
I have used the pythons library calendar to sort the months from January to December.

```
RDD1 = RDD2021_Split.first()
RDD2021_Count = RDD2021_Split.filter(lambda Header: Header!=RDD1)    #skipping header
RDD2021_Study_Completion=RDD2021_Count.map(lambda x: (x[4].split(),x[2]))\
.filter(lambda x: len(x[0])!=0)\                                          #skipping null values/index
.filter(lambda y:(y[1]=='Completed'))\
.filter(lambda z: (z[0][1]==Year))\
.map(lambda a: (a[0][0],1))\
.reduceByKey(lambda v1, v2: v1+ v2)
RDD2021_Study_Completion.take(12)

import calendar                                        #python library calender to sort the months/keys sorting
Sorting = {a:z for z,a in enumerate(calendar.month_abbr[1:],1)}
RDD2021_Study_Completion_Sorting=RDD2021_Study_Completion.sortBy(lambda g: Sorting.get(g[0]))
RDD2021_Study_Completion_Sorting.collect()
```

Screenshot:



```
Cmd 23

import calendar
Sorting = {a:z for z,a in enumerate(calendar.month_abbr[1:],1)}
RDD2021_Study_Completion_Sorting=RDD2021_Study_Completion.sortBy(lambda g: Sorting.get(g[0]))
RDD2021_Study_Completion_Sorting.collect()

▸ (3) Spark Jobs

Out[14]: [('Jan', 1131),
 ('Feb', 934),
 ('Mar', 1227),
 ('Apr', 967),
 ('May', 984),
 ('Jun', 1094),
 ('Jul', 819),
 ('Aug', 700),
 ('Sep', 528),
 ('Oct', 187)]

Command took 3.08 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:24:06 on Start
```

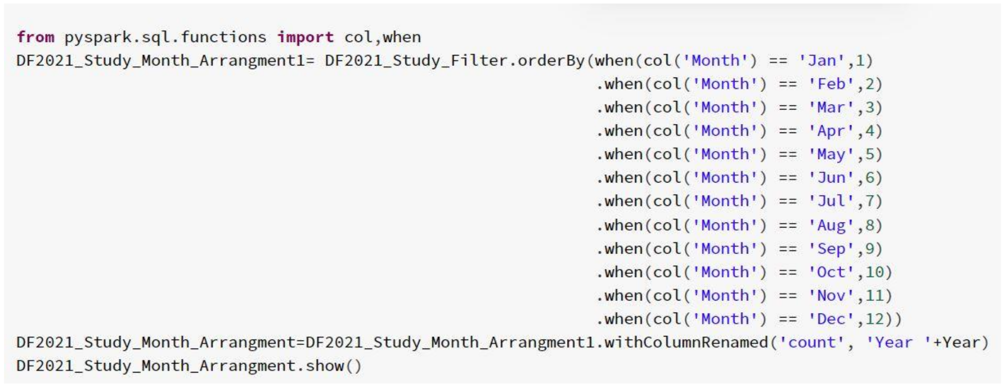
Code Segments Used in DF Implementation:

I have splitted the column completion to get the month and year and took the data from status column where study was completed. Then I used groupby function to take the count of completed study count month wise in 2021.
To sort the month from January to December I used the sql function for custom sorting to get the desired output.

DF2021_Study1=DF2021.withColumn('Month', split(DF2021['Completion'], ' ').getItem(0))\
.withColumn('year', split(DF2021['Completion'], ' ').getItem(1))
DF2021_Study1.show()

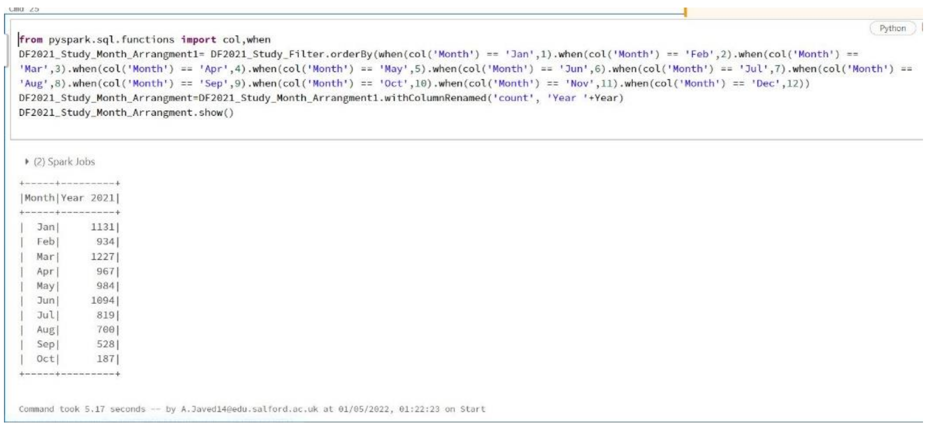
DF2021_Study_Filter=DF2021_Study1.filter((DF2021_Study1.Status=='Completed') & (DF2021_Study1.year==Year)).groupby('Month').count()
DF2021_Study_Filter.show()

#importing sql function to sort months from Jan to Dec



```
from pyspark.sql.functions import col,when
DF2021_Study_Month_Arrangment1= DF2021_Study_Filter.orderBy(when(col('Month') == 'Jan',1)
                                                             .when(col('Month') == 'Feb',2)
                                                             .when(col('Month') == 'Mar',3)
                                                             .when(col('Month') == 'Apr',4)
                                                             .when(col('Month') == 'May',5)
                                                             .when(col('Month') == 'Jun',6)
                                                             .when(col('Month') == 'Jul',7)
                                                             .when(col('Month') == 'Aug',8)
                                                             .when(col('Month') == 'Sep',9)
                                                             .when(col('Month') == 'Oct',10)
                                                             .when(col('Month') == 'Nov',11)
                                                             .when(col('Month') == 'Dec',12))
DF2021_Study_Month_Arrangment=DF2021_Study_Month_Arrangment1.withColumnRenamed('count', 'Year '+Year)
DF2021_Study_Month_Arrangment.show()
```

Screenshot:



```
from pyspark.sql.functions import col,when
DF2021_Study_Month_Arrangment1= DF2021_Study_Filter.orderBy(when(col('Month') == 'Jan',1).when(col('Month') == 'Feb',2).when(col('Month') ==
'Mar',3).when(col('Month') == 'Apr',4).when(col('Month') == 'May',5).when(col('Month') == 'Jun',6).when(col('Month') == 'Jul',7).when(col('Month') ==
'Aug',8).when(col('Month') == 'Sep',9).when(col('Month') == 'Oct',10).when(col('Month') == 'Nov',11).when(col('Month') == 'Dec',12))
DF2021_Study_Month_Arrangment=DF2021_Study_Month_Arrangment1.withColumnRenamed('count', 'Year '+Year)
DF2021_Study_Month_Arrangment.show()
```

```
+-----+---------+
|Month|Year 2021|
+-----+---------+
|  Jan|     1131|
|  Feb|      934|
|  Mar|     1227|
|  Apr|      967|
|  May|      984|
|  Jun|     1094|
|  Jul|      819|
|  Aug|      700|
|  Sep|      528|
|  Oct|      187|
+-----+---------+

Command took 5.17 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:22:23 on Start
```

Code Segments Used in HQL Implementation:

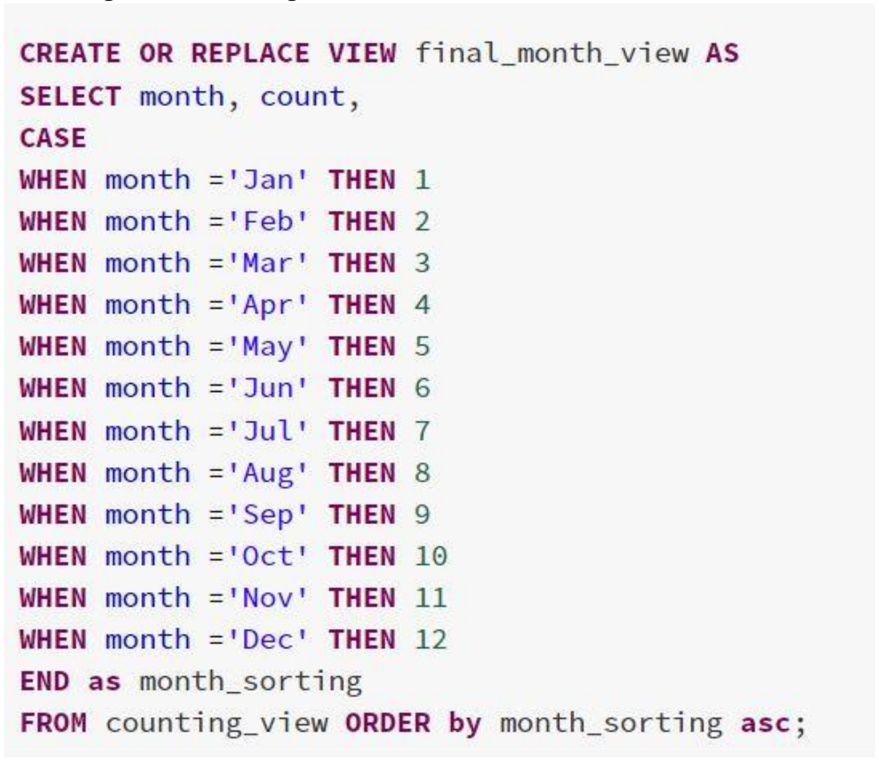
I have created a view where I get the month and year from a column completion by splitting the values. Again I created a view to get the status completed and I used a global variable to get the year. Then I grouped the keys to get the count of completed studies.

Case function was used to sort the months.

CREATE OR REPLACE VIEW month_year
AS
select *, split(completion,' ')[0] as Month, split(completion,' ')[1] as Year from clinicaltrial_2021 where completion is not null

CREATE OR REPLACE VIEW counting_view
AS
select month,count(month)as count from month_year where Status=='Completed' and Year=='$Year' group by month

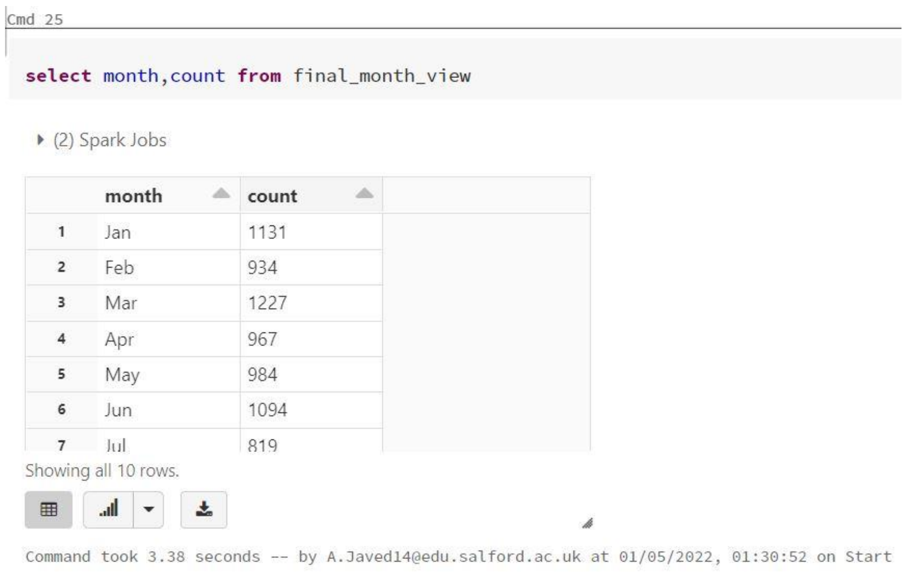
#creating view and using CASE function to sort months



```
CREATE OR REPLACE VIEW final_month_view AS
SELECT month, count,
CASE
WHEN month ='Jan' THEN 1
WHEN month ='Feb' THEN 2
WHEN month ='Mar' THEN 3
WHEN month ='Apr' THEN 4
WHEN month ='May' THEN 5
WHEN month ='Jun' THEN 6
WHEN month ='Jul' THEN 7
WHEN month ='Aug' THEN 8
WHEN month ='Sep' THEN 9
WHEN month ='Oct' THEN 10
WHEN month ='Nov' THEN 11
WHEN month ='Dec' THEN 12
END as month_sorting
FROM counting_view ORDER by month_sorting asc;
```

select month,count from final_month_view

Screenshot:



```
Cmd 25

select month,count from final_month_view
```

(2) Spark Jobs

	month	count
1	Jan	1131
2	Feb	934
3	Mar	1227
4	Apr	967
5	May	984
6	Jun	1094
7	Jul	819

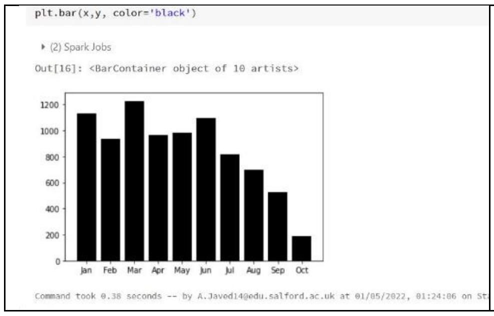
Showing all 10 rows.

```
Command took 3.38 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:30:52 on Start
```

Visualization:

Visualization in RDD implementation:

I have used the matplotlib library of python. To show months and study count on axis I have mapped the final RDD and collected its each value. Multiple graphs for visualization is used to show the analysis.

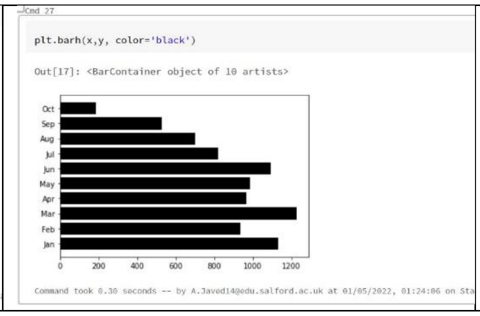


```
plt.bar(x,y, color='black')
```

(2) Spark Jobs

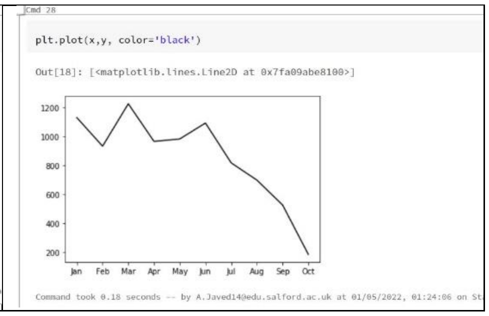
```
Out[16]: <BarContainer object of 10 artists>
```

```
Command took 0.38 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:24:06 on St
```



```
Cmd 27

plt.barh(x,y, color='black')

Out[17]: <BarContainer object of 10 artists>
```

```
Command took 0.30 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:24:06 on Sta
```

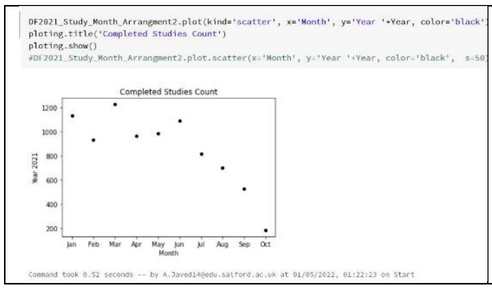


```
Cmd 28

plt.plot(x,y, color='black')

Out[18]: [<matplotlib.lines.Line2D at 0x7fa09abe8100>]
```

```
Command took 0.18 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:24:06 on St
```

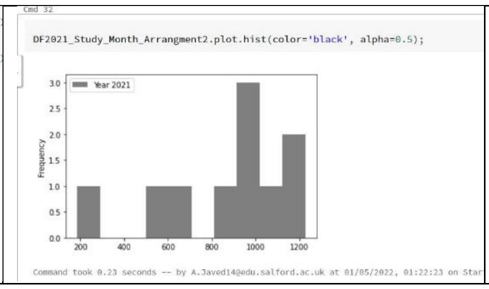
Visualization in DF implementation:

I used matplot library of python here also. And different settings and configuration is used to visualize the final dataframe.



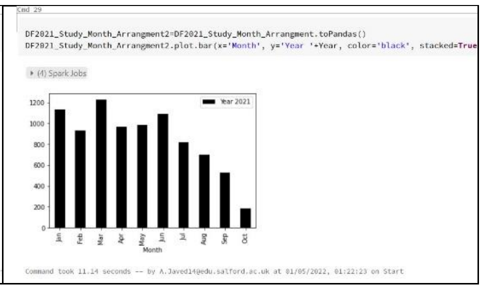
```
DF2021_Study_Month_Arrangment2.plot(kind='scatter', x='Month', y='Year '+Year, color='black')
ploting.title('Completed Studies Count')
ploting.show()
#DF2021_Study_Month_Arrangment2.plot.scatter(x='Month', y='Year '+Year, color='black',  s=50)
```

```
Command took 0.52 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:22:23 on Start
```



```
Cmd 32

DF2021_Study_Month_Arrangment2.plot.hist(color='black', alpha=0.5);
```

```
Command took 0.23 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:22:23 on Star
```



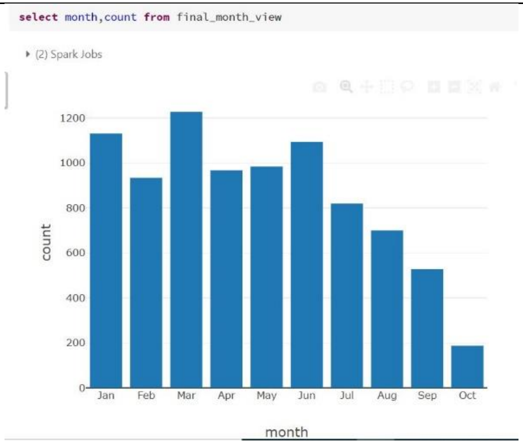
```
Cmd 29

DF2021_Study_Month_Arrangment2=DF2021_Study_Month_Arrangment.toPandas()
DF2021_Study_Month_Arrangment2.plot.bar(x='Month', y='Year '+Year, color='black', stacked=True
```

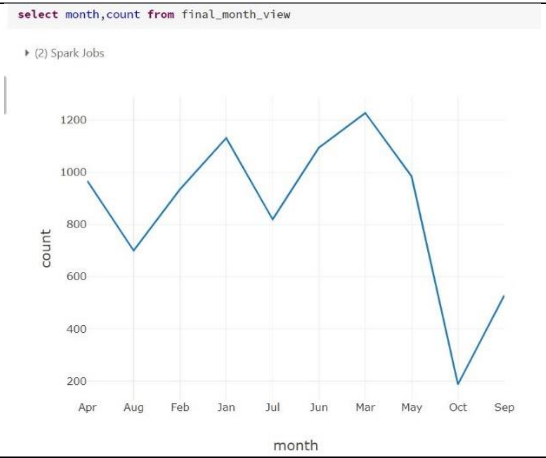
(4) Spark Jobs

```
Command took 11.14 seconds -- by A.Javed14@edu.salford.ac.uk at 01/05/2022, 01:22:23 on Start
```

Visualization in HQL implementation:

For visualization in Hive SQL I have used the built available plotting options. Just selecting the multiple plot layouts I represent the data.


```
select month,count from final_month_view
```
▸ (2) Spark Jobs


```
select month,count from final_month_view
```
▸ (2) Spark Jobs

Instructions to Run:

Use web based cloud platform Databricks for running the codes. This has the python and sql notebook interface. Upload/import the .ipynb file into databricks workspace and execute it.