University of Salford MANCHESTER

Big Data Tools & Development

Assignment 2022

Sophie Mannion

@00641004

Data preparation and exploration

Files “clinicaltrial_2021.csv.gz”, “mesh.csv” and “pharma.csv” were uploaded to Databricks using the UI interface and placed in directory “dbfs:/FileStore/tables/”. Clinical trial data required unzipping. Dataset year was saved as a variable in string format and used to check whether the unzipped csv file for that year is present in the directory, otherwise the gzip file is copied to “file:/tmp/”, unzipped and moved back to “dbfs:/FileStore/tables/” as a csv file.

Insert year for clinical trial dataset

```
year="2021"
```

Required additional packages

```
import os
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import seaborn as sns
sns.set()
from pandas import DataFrame
from pyspark.sql.functions import isnan, count, col, when,split, explode,from_unixtime, unix_timestamp, regexp_replace, months_between, countDistinct,
when,collect_set
from pyspark.sql.types import TimestampType, DoubleType
```

Unzipping clinical trial data

```
#check year variable inputted and is a string
try:
    filename = "clinicaltrial_"+year
except:
    dbutils.notebook.exit("ERROR: Year variable must be string - please correct input in Cmd 1")

unzipFile = "false"
#if clinical trial file is not in database then change unzipFile to true
try:
    dbutils.fs.ls("/FileStore/tables/"+ filename +".csv")
except:
    unzipFile = "true"
#if true then try to move file to /tmp/ else gzip file needs importing and exit notebook
if unzipFile == "true":
    try:
        dbutils.fs.cp("FileStore/tables/"+filename+"_csv.gz", "file:/tmp")
        os.environ["unzipFiles"] = "true"
        os.environ["file"] = filename+"_csv.gz"
    except:
         dbutils.notebook.exit("import" + filename + "gzip file")
```

```
%%bash
#if unzip file true then unzip file in tmp
if [[ $unzipFiles = "true" ]]; then
    gzip -d /tmp/$file

fi
```

```
#if unzip file true then try to move file into /tables/ else exit notebook
if unzipFile == "true":
    try:
        dbutils.fs.mv("file:/tmp/"+filename+"_csv", "/FileStore/tables/"+filename+".csv", True)
    except:
        dbutils.notebook.exit("csv file not in /tmp/")
```

For data exploration, “clinicaltrial_2021.csv” was read into the notebook as a dataframe. The data types were all strings. Although “Start”, “Completion” and “Submission” are dates these were left as strings for the time being. Conditions and interventions were recorded as lists of all studied in each trial.

```
#import clinical trial data
clinicaltrial = filename + ".csv"
trialDF = spark.read.options(inferSchema="True",delimiter="|",header="True").csv("/FileStore/tables/"+clinicaltrial)
trialDF.display()
```

	Id	Sponsor	Status	Start	Completion	Type
1	NCT02758028	The University of Hong Kong	Recruiting	Aug 2005	Nov 2021	Interventiona
2	NCT02751957	Duke University	Completed	Jul 2016	Jul 2020	Interventiona
3	NCT02758483	Universidade Federal do Rio de Janeiro	Completed	Mar 2017	Jan 2018	Interventiona
4	NCT02759848	Istanbul Medeniyet University	Completed	Jan 2012	Dec 2014	Observationa
5	NCT02758860	University of Roma La Sapienza	Active, not recruiting	Jun 2016	Sep 2020	Observationa
6	NCT02757209	Consorzio Futuro in Ricerca	Completed	Apr 2016	Jan 2018	Interventiona
7	NCT02752438	Ankara University	Unknown status	May 2016	Jul 2017	Observationa

Truncated results, showing first 1000 rows.

```
#dataframe schema
trialDF.printSchema()
```

```
root
 |-- Id: string (nullable = true)
 |-- Sponsor: string (nullable = true)
 |-- Status: string (nullable = true)
 |-- Start: string (nullable = true)
 |-- Completion: string (nullable = true)
 |-- Type: string (nullable = true)
 |-- Submission: string (nullable = true)
 |-- Conditions: string (nullable = true)
 |-- Interventions: string (nullable = true)
```

The dataset was checked for duplicates. There were not any duplicate rows when considering entire rows or just trial ‘Id’ column. When not including ‘Id’ 1502 rows appeared to be duplicated, likely the same trials recorded multiple times under different trial IDs, so were removed from the dataset.

```
#number of rows
trialDF.distinct().count()

Out[80]: 387261
```

```
#check for duplicate rows
trialDF.groupBy("Id","Sponsor","Status", "Start", "Completion", "Type", "Submission", "Conditions", "Interventions").count().filter("count > 1").count()

Out[81]: 0
```

```
#check for duplicate Id
trialDF.groupBy("Id").count().filter("count > 1").count()

Out[82]: 0
```

```
#some trials repeated under different IDs
trialDF.groupBy("Sponsor","Status", "Start", "Completion", "Type", "Submission", "Conditions", "Interventions").count().filter("count >
1").orderBy("count").count()

Out[83]: 1502
```

```
#remove duplicated data
trialDF=trialDF.dropDuplicates(["Sponsor","Status", "Start", "Completion", "Type", "Submission", "Conditions", "Interventions"])
```

```
#number of distinct trials
trialDF.distinct().count()

Out[85]: 385562
```

Number of distinct values and of missing data for each column was produced. The completion date, conditions, and interventions for a number of studies are missing. Interventions and completion date could be explained if not using interventions or being incomplete, however, it is expected all clinical trials would investigate a condition. It is unclear whether the dataset is incomplete or not.

```
#number of distinct values for each column
trialDF.select(countDistinct("Id").alias("Id"),countDistinct("Sponsor").alias("Sponsor"), countDistinct("Status").alias("Status"),
countDistinct("Start").alias("Start"), countDistinct("Completion").alias("Completion"), countDistinct("Type").alias("Type"),
countDistinct("Submission").alias("Submission"), countDistinct("Conditions").alias("Conditions"),
countDistinct("Interventions").alias("Interventions")).display()
```

	Id	Sponsor	Status	Start	Completion	Type	Submission	Conditions	Interventions
1	385562	34442	13	556	761	4	266	46769	29177

Showing all 1 rows.

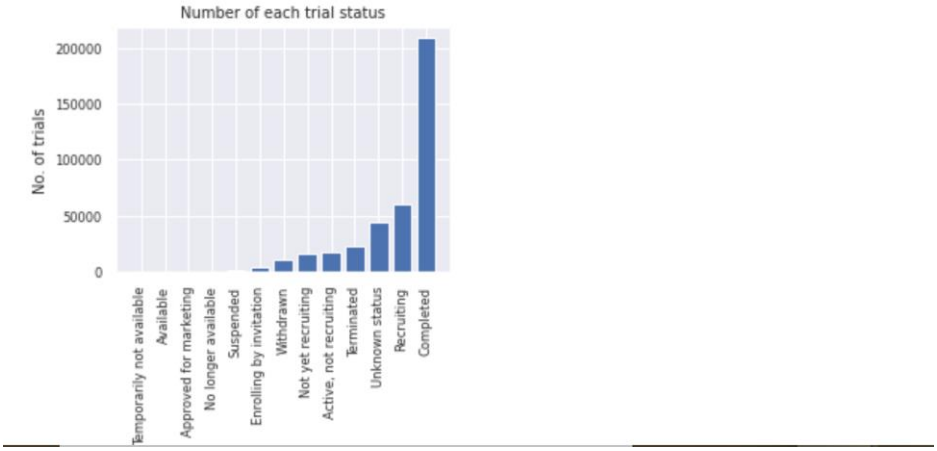
```
#display number of missing data for each column
trialDF.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in trialDF.columns]).display()
```

	Id	Sponsor	Status	Start	Completion	Type	Submission	Conditions	Interventions
1	0	0	0	0	13089	0	0	64509	252732

Showing all 1 rows.

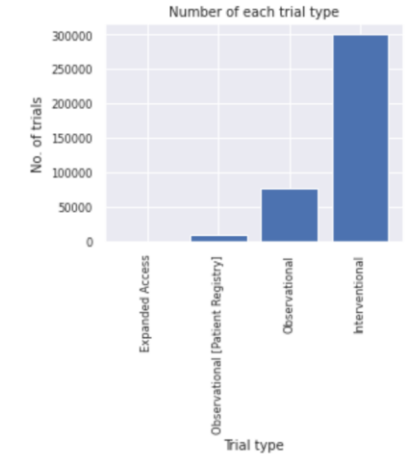
The number of completed trials is higher than any of the other trial types, so trials could be simplified to completed or incomplete. It is unclear whether other trial statuses also suggest a trial is competed, such as “Available” and “Approved for marketing”, so in future analysis only trials officially recorded as “Completed” will be considered as completed trials.

```
#number of trials by status
statusDF=trialDF.groupBy("Status").count().orderBy("count").toPandas()

fig = plt.figure(figsize = (4,3))

plt.bar(statusDF["Status"], statusDF["count"])

plt.xlabel("Trial status", fontsize="small")
plt.xticks(fontsize="x-small",rotation=90)
plt.ylabel("No. of trials", fontsize="small")
plt.yticks(fontsize="x-small")
plt.title("Number of each trial status", fontsize="small")
plt.show()
```



Interventional trials are most frequent, followed by observational. Observational using patient registry could be combined with observational studies, however the number is relatively small, so effect of combining would likely be negligible. Expanded access studies are so few they are mostly negligible.

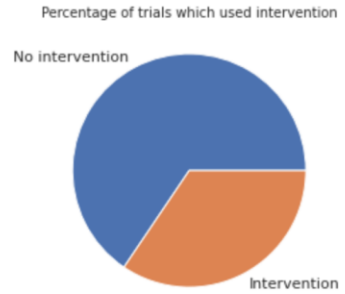
```
#number of trials by type
typeDF=trialDF.groupBy("Type").count().orderBy("count").toPandas()
fig2 = plt.figure(figsize = (4,3))

plt.bar(typeDF["Type"], typeDF["count"])

plt.xlabel("Trial type", fontsize="small")
plt.xticks(fontsize="x-small",rotation=90)
plt.ylabel("No. of trials", fontsize="small")
plt.yticks(fontsize="x-small")
plt.title("Number of each trial type", fontsize="small")
plt.show()
```



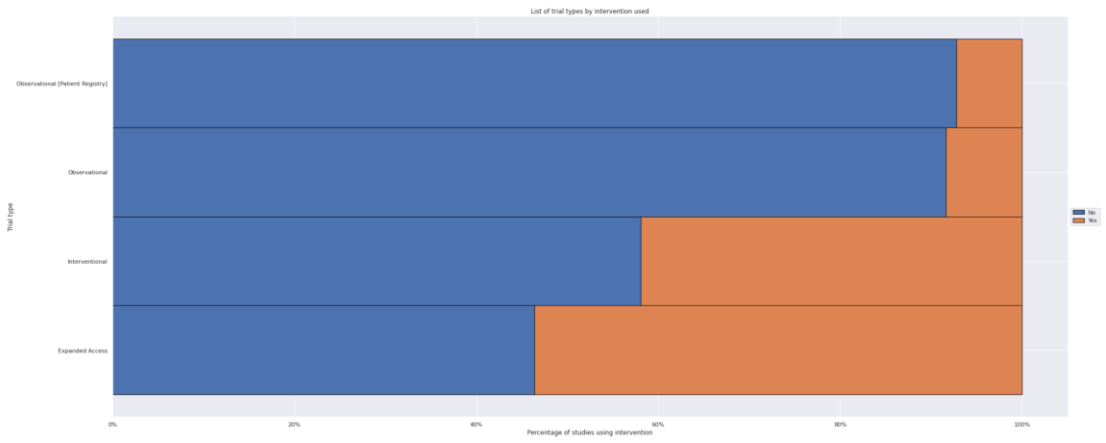
When plotting the percentage of trials with or without an intervention recorded for each trial type, around 60% of interventional studies did not have an intervention recorded. This could suggest the clinical trial data is incomplete.

```
#percentage of trials which used an intervention verses those who did not
no_intervention=trialDF.select("Interventions").filter("Interventions is null").count()
intervention=trialDF.select("Interventions").filter("Interventions is not null").count()
total_trials=trialDF.count()
no_intervention/total_trials *100
interventions = {
    "Label":["No intervention", "Intervention"],
    "Total": [no_intervention, intervention],
    "Percentage": [(no_intervention/total_trials *100), (intervention/total_trials *100)]
}
```

```
#pie chart for percentage of trials which used/did not use intervention
fig3 = plt.figure(figsize = plt.figaspect(1))
plt.pie(interventions["Percentage"], labels=interventions["Label"])
plt.title("Percentage of trials which used intervention", fontsize="small")
plt.show()
```




```
#interventions by type
conditions = when(col("Interventions").isNull(), "No").otherwise("Yes")
intervention=trialDF.withColumn("Intervention", conditions).groupBy("Type", "Intervention").count().toPandas()

fig = plt.figure(figsize=(30, 12))
grid_size = (1,1)
ax = plt.subplot2grid(grid_size, (0,0), colspan=1, rowspan=1)
plot = intervention.groupby(["Type", "Intervention"]).agg(np.mean).groupby(level=0).apply(lambda x: 100 * x / x.sum()).unstack().plot(kind="barh"
                    , stacked=True, width=1, edgecolor="black", ax=ax, title="List of trial types by intervention used")
ylabel = plt.ylabel("Trial type");
xlabel = plt.xlabel("Percentage of studies using intervention");
legend = plt.legend(sorted(intervention["Intervention"].unique()), loc="center left", bbox_to_anchor=(1.0, 0.5))
param_update = plt.rcParams.update({"font.size": 16});
ax = plt.gca()
formatter = ax.xaxis.set_major_formatter(mtick.PercentFormatter());
a = fig.tight_layout()
plt.show()
```



Looking at recorded interventions, they appear to mostly be drug related.

```
#interventions used in trials by frequency
trialDF.select("Interventions").withColumn("Interventions",explode(split("Interventions",","))).groupBy("Interventions").count().orderBy("count", ascending=False).display()
```

	Interventions	count
1	Paclitaxel	3218
2	Cyclophosphamide	3003
3	Dexamethasone	2515
4	Carboplatin	2388
5	Antibodies	2331
6	Gemcitabine	2195
7	Vaccines	2194

Truncated results, showing first 1000 rows.

The top five sponsors are either pharmaceutical companies or cancer research centres.

```
#number of sponsors
trialDF.select("Sponsor").distinct().count()

Out[95]: 34442
```

```
#top 10 sponsors
trialDF.groupBy("Sponsor").count().orderBy("count", ascending=False).limit(10).display()
```

	Sponsor	count
1	GlaxoSmithKline	3343
2	National Cancer Institute (NCI)	3212
3	AstraZeneca	2660
4	Pfizer	2628
5	M.D. Anderson Cancer Center	2414
6	Assistance Publique - Hôpitaux de Paris	2356
7	Mayo Clinic	2298

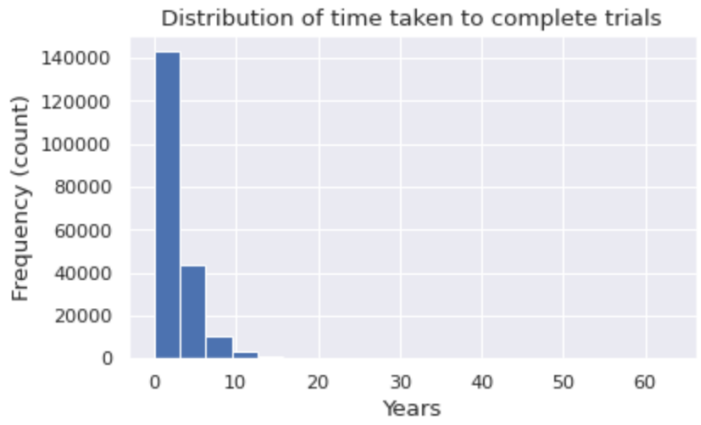
Showing all 10 rows.

The time taken for trials to be completed (in years) was worked out by converting the dates into timestamps and then calculating the months between “Start” and “Completion” for completed studies and dividing this number by twelve. 75% of the studies appeared to take less than 3.50 years, with some outliers, including a study which took 63.08 years.

```
#years taken to complete studies
spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
#create timestamps for each date column
trialDF=trialDF.withColumn("Start-timestamp", (unix_timestamp(trialDF["Start"],"MMM yyyy"))).withColumn("Completion_timestamp", (unix_timestamp(trialDF["Completion"],"MMM
yyyy"))).withColumn("Submission_timestamp", (unix_timestamp(trialDF["Submission"],"MMM yyyy")))
#filter for completed trials and create years column to work out difference in years between start and completion timestamp
timeDF=trialDF.filter(trialDF.Status=="Completed").select(trialDF["Start-timestamp"].cast(TimestampType()), trialDF["Completion_timestamp"].cast(TimestampType())).withColumn("Years",
(months_between(col("Completion_timestamp"),col("Start-timestamp"))/12).cast(DoubleType()))

timeDF.select("Years").toPandas().describe()
```

	Years
count	202448.000000
mean	2.605998
std	2.634629
min	0.000000
25%	0.833333
50%	1.916667
75%	3.500000
max	63.083333

```
#distribution of years taken to complete trials
years = timeDF.select(["Years"]).filter(col("Years").isNotNull()).toPandas()
fig4 = plt.figure(figsize = (6,3.5))
plt.hist(years, bins=20)
plt.xlabel("Years", fontsize="small")
plt.xticks(fontsize="x-small")
plt.ylabel("Frequency (count)", fontsize="small")
plt.yticks(fontsize="x-small")
plt.title("Distribution of time taken to complete trials", fontsize="small")
plt.show()
```



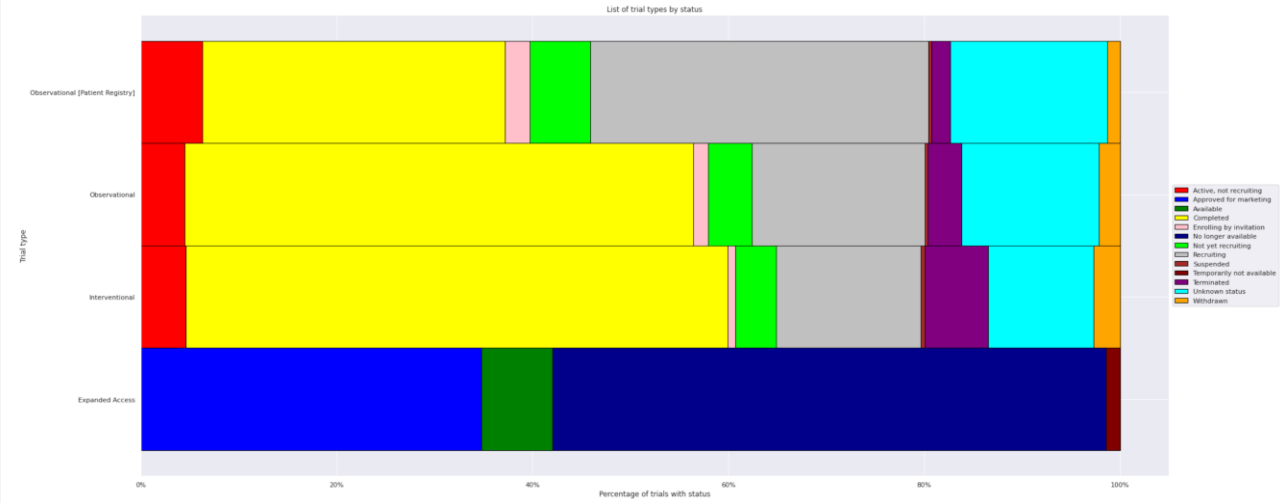
None of expanded access trials have the status “Completed”. Further investigation shows expanded access trials have “Available”, “Approved for marketing”, “Withdrawn” or “Suspended” statuses. As this type of trial involves monitoring experimental drugs on patients, it is possible completed trials of this type are either “Available” or “Approved for marketing”.

```
#number of completed trials for each trial type
trialDF.filter(trialDF.Status=="Completed").groupBy("Type").count().orderBy("count", ascending=False).display()
```

	Type	count
1	Interventional	166083
2	Observational	40120
3	Observational [Patient Registry]	2519

Showing all 3 rows.

```
#trial types by status
type_by_status = trialDF.groupBy("Type", "Status").count().toPandas()

colours = ["red", "blue", "green", "yellow", "pink", "darkblue", "lime", "silver", "brown", "maroon", "purple", "cyan", "orange"]

fig = plt.figure(figsize=(30, 12))
grid_size = (1,1)
ax = plt.subplot2grid(grid_size, (0,0), colspan=1, rowspan=1)
plot = type_by_status.groupby(["Type", "Status"]).agg(np.mean).groupby(level=0).apply(
    lambda x: 100 * x / x.sum()).unstack().plot(kind="barh", stacked=True, width=1
                , edgecolor="black", color = colours,  ax=ax, title="List of trial types by status")
ylabel = plt.ylabel("Trial type");
xlabel = plt.xlabel("Percentage of trials with status");
legend = plt.legend(
    sorted(type_by_status["Status"].unique()), loc="center left", bbox_to_anchor=(1.0, 0.5)
)
param_update = plt.rcParams.update({"font.size": 16});
ax = plt.gca()
formatter = ax.xaxis.set_major_formatter(mtick.PercentFormatter());
a = fig.tight_layout()
plt.show()
```



The number of distinct rows in the mesh dataset was 124,282, with the number of distinct terms being 58,804 and distinct trees being 61,314. There are a number of terms which share the same tree and a number of terms which have multiple trees.

```
#create mesh dataframe
meshDF = spark.read.options(inferSchema="True",delimiter=",",header="True").csv("/FileStore/tables/mesh.csv")
meshDF.display()
```

	term	tree
1	Calcimycin	D03.633.100.221.173
2	A-23187	D03.633.100.221.173
3	Temefos	D02.705.400.625.800
4	Temefos	D02.705.539.345.800
5	Temefos	D02.886.300.692.800
6	Abate	D02.705.400.625.800
7	Abate	D02.705.539.345.800

Truncated results, showing first 1000 rows.

```
#Mesh dataframe schema
meshDF.printSchema()

root
 |-- term: string (nullable = true)
 |-- tree: string (nullable = true)
```

```
#drop any duplicate mesh values
meshDF.dropDuplicates()

Out[103]: DataFrame[term: string, tree: string]
```

```
#number of distinct rows
meshDF.distinct().count()

Out[104]: 124282
```

```
#number of distinct term values
meshDF.select("term").distinct().count()

Out[105]: 58804
```

```
#number of distinct tree values
meshDF.select("tree").distinct().count()

Out[106]: 61314
```

```
#terms associated with specific trees
duplicateTree=meshDF.groupBy("tree").agg(collect_set("term").alias("terms")).withColumnRenamed("tree","tree_name")
meshDF.groupBy("tree").count().join(duplicateTree, duplicateTree["tree_name"]==meshDF["tree"]).select("tree","count","terms").orderBy("count",ascending=False).display()
```

	tree	count	terms
1	D03.633.100.150.909.750	55	["Snow-E Muscle, Energy & Feritility", "Eplonat", "Vitamin E Sanum", "Vitamin-E EVI-MIRALE", "Hydrovit E", "Vitazell", "Richtavit E", "Ecoro", "Togasan Vitamin E", "E-ferol", "EUNOVA Vitamin E", "Uno-Vit", "Tocolion", "Tocopherol Bayer", "E-Vitamin-Ratiopharm", "E Vitamin E", "Bio E", "Detulin", "Ephynal", "Sanavitan S", "Biopto-E", "Mowivit Vitamin E", "Spondyvit", "Davitamon", "Elex Verla", "Puncto E", "Biosan", "Vitamin-E Dragees", "Aquasol E", "Vit E hydrosol", "Vitagutt Vitamin E", "Vibolex", "E-Mulsin", "Vitamin E AL", "Vitamin E Natur", "Dermorelle", "Tocopa", "Tocopherols", "Embial", "Vita-E", "Equivit E", "Unique E", "Vita-Plus E", "Dal-E", "E-Vicotrat", "Vitamine E GNR", "Bioweyxin", "Tocovital", "Malton E", "Abortosan", "Vit. E Stada", "Auxina E", "Tocopharm", "Vitamin E Suspension", "Eusovit"]
2	D02.092.668.387.750	55	["Ultram", "Tramadol-Hameln", "Tramadin", "Tramal", "Trama KD", "Tramadol Lichtenstein", "Tramadoc", "Tramadol Basics", "Nobligan", "Zamudol", "Xymel 50", "Tramadol", "Tradol", "Topalgic", "Tramadol Cinfa", "Tramadol Hydrochloride", "Tramadol Bexal", "Tramadol AL", "Trama-Dorsch", "Tramadol Mabo", "Tramadol Kern", "Zumalgic", "Zytram", "Tramadol acis", "Tramake", "Tramadol Bayvit", "Zydol", "Jutadol", "Tramadol Stada", "Tradol-Puren", "Trama AbZ", "Prontofort", "Tramadol 1A", "Tramex", "Tiral", "Tramadol Heumann", "Theradol", "Tramagit", "Tramadol Normon", "Trasedal", "Tramabeta", "Tramadol-Dolgit", "Tradonal", "Amadol", "Adolonta", "Tralgiol", "Tramadol Edigen", "Biodalgic", "Tramadolor", "MTW-Tramadol", "K-315", "Tramundin", "Tramadura", "Takadol", "Tramagetic"]
3	D02.033.415.510.500.802	55	["Ultram", "Tramadol-Hameln", "Tramadin", "Tramal", "Trama KD", "Tramadol Lichtenstein", "Tramadoc", "Tramadol Basics", "Nobligan", "Zamudol", "Xymel 50", "Tramadol", "Tradol", "Topalgic", "Tramadol Cinfa", "Tramadol Hydrochloride", "Tramadol Bexal", "Tramadol AL", "Trama-Dorsch", "Tramadol Mabo", "Tramadol Kern", "Zumalgic", "Zytram", "Tramadol acis", "Tramake", "Tramadol Bayvit", "Zydol", "Jutadol", "Tramadol Stada", "Tradol-Puren", "Trama AbZ", "Prontofort", "Tramadol 1A", "Tramex", "Tiral", "Tramadol Heumann", "Theradol", "Tramagit", "Tramadol Normon", "Trasedal", "Tramabeta", "Tramadol-Dolgit", "Tradonal", "Amadol", "Adolonta", "Tralgiol", "Tramadol Edigen", "Biodalgic", "Tramadolor", "MTW-Tramadol", "K-315", "Tramundin", "Tramadura", "Takadol", "Tramagetic"]

Truncated results, showing first 1000 rows.

```
#trees associated with terms
duplicateTerm=meshDF.groupBy("term").agg(collect_set("tree").alias("trees")).withColumnRenamed("term","term_name")
meshDF.groupBy("term").count().join(duplicateTerm, duplicateTerm["term_name"]==meshDF["term"]).select("term","count","trees").orderBy("count",ascending=False).display()
```

	term	count	trees
1	WAGR Syndrome	20	["C16.131.384.079.950", "C16.131.260.940", "C13.351.875.253.096.875", "C11.941.375.060.950", "C10.597.606.360.969", "C16.320.290.078.950", "C04.588.945.947.535.585.950", "C11.250.060.950", "C12.777.419.473.585.950", "C19.391.119.096.875", "C13.351.968.419.473.585.950", "C13.351.937.820.535.585.950", "C12.758.820.750.585.950", "C16.131.939.316.096.875", "C04.557.435.595.950", "C16.320.700.900.950", "C12.706.316.096.875", "C16.320.180.940", "C04.700.900.950", "C11.270.060.950"]
2	Wolfram Syndrome 1	16	["C16.131.077.299.750", "C11.270.564.980", "C12.777.419.135.875", "C16.320.400.630.980", "C18.452.394.750.124.960", "C11.966.075.375.750", "C13.351.968.419.135.875", "C10.292.700.225.500.980", "C19.700.159.875", "C10.574.500.662.980", "C11.640.451.451.980", "C19.246.267.960", "C10.597.751.941.162.625.750", "C10.597.751.418.341.186.500.750", "C09.218.458.341.186.500.750", "C16.320.290.564.980"]
3	Adrenomyeloneuropathy	16	["C18.452.132.100.084", "C16.320.565.189.362.250", "C16.320.400.525.124", "C18.452.648.189.084", "C16.320.565.663.100", "C18.452.648.663.100", "C18.452.132.100.362.250", "C18.452.648.189.362.250", "C10.228.140.163.100.084", "C10.228.140.695.625.250", "C10.597.606.360.455.124", "C16.320.565.189.084", "C10.314.400.250", "C16.320.322.500.124", "C19.053.500.270", "C10.228.140.163.100.362.250"]
4	Endoscopic Ultrasound-Guided Fine Needle Aspiration	16	["E05.200.500.384.100.119.500.500", "E05.242.384.100.119.500.500", "E01.370.388.100.100.500.500", "E04.074.370.500", "E01.370.350.850.855.500", "E04.074.119.500.500", "E01.370.388.100.370.500", "E01.370.225.998.054.119.500.500", "E04.502.890.500", "E05.200.500.384.100.370.500", "E01.370.225.500.384.100.370.500", "E01.370.225.998.054.370.500", "E01.370.225.500.384.100.119.500.500", "E05.200.998.054.119.500.500", "E05.242.384.100.370.500", "E05.200.998.054.370.500"]
5	Adrenoleukodystrophy	16	["C18.452.132.100.084", "C16.320.565.189.362.250", "C16.320.400.525.124", "C18.452.648.189.084", "C16.320.565.663.100", "C18.452.648.663.100", "C18.452.132.100.362.250", "C18.452.648.189.362.250", "C10.228.140.163.100.084", "C10.228.140.695.625.250", "C10.597.606.360.455.124", "C16.320.565.189.084", "C10.314.400.250", "C16.320.322.500.124", "C19.053.500.270", "C10.228.140.163.100.362.250"]

Truncated results, showing first 1000 rows.

There are 608 distinct companies in pharma dataset and 72 parent companies. These companies are from 13 different countries. When looking at the specific industry and the major industry of parent companies, columns do not appear to be split correctly with some columns containing states and countries. These columns are unlikely to be

used, and overall, columns support this dataset containing pharmaceutical companies only.

```
#pharma dataframe
pharmaDF = spark.read.options(inferSchema="True",delimiter=",",header="True").csv("/FileStore/tables/pharma.csv")
pharmaDF.display()
```

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

Showing all 968 rows.

```
#Pharma dataframe schema
pharmaDF.printSchema()
```

```
root
 |-- Company: string (nullable = true)
 |-- Parent_Company: string (nullable = true)
 |-- Penalty_Amount: string (nullable = true)
 |-- Subtraction_From_Penalty: string (nullable = true)
 |-- Penalty_Amount_Adjusted_For_Eliminating_Multiple_Counting: string (nullable = true)
 |-- Penalty_Year: integer (nullable = true)
 |-- Penalty_Date: integer (nullable = true)
 |-- Offense_Group: string (nullable = true)
 |-- Primary_Offense: string (nullable = true)
 |-- Secondary_Offense: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Level_of_Government: string (nullable = true)
 |-- Action_Type: string (nullable = true)
 |-- Agency: string (nullable = true)
 |-- Civil/Criminal: string (nullable = true)
 |-- Prosecution_Agreement: string (nullable = true)
 |-- Court: string (nullable = true)
 |-- Case_ID: string (nullable = true)
 |-- Private_Litigation_Case_Title: string (nullable = true)
 |-- Lawsuit_Resolution: string (nullable = true)
```

```
#number of distinct values for each column
pharmaDF.select(countDistinct("Company").alias("Company"),countDistinct("Parent_Company").alias("Parent_Company"),countDistinct("HQ_Country_of_Parent").alias("HQ_Country_of_Parent"),
countDistinct("Major_Industry_of_Parent").alias("Major_Industry_of_Parent"), countDistinct("Specific_Industry_of_Parent").alias("Specific_Industry_of_Parent")).display()
```

	Company	Parent_Company	HQ_Country_of_Parent	Major_Industry_of_Parent	Specific_Industry_of_Parent
1	608	72	13	4	10

Showing all 1 rows.

```
#parent company country
pharmaDF.select("HQ_Country_of_Parent").distinct().display()
```

	HQ_Country_of_Parent
1	Germany
2	France
3	null
4	Belgium
5	India
6	Denmark
7	Ireland
8	Israel
9	USA
10	Switzerland
11	Canada
12	Japan
13	Australia
14	United Kingdom

```
#state for parents
pharmaDF.select("HQ_State_of_Parent").distinct().display()
```

	HQ_State_of_Parent
1	Utah
2	null
3	Pennsylvania
4	Connecticut
5	Illinois
6	Delaware
7	Ireland
8	Missouri
9	Georgia
10	Virginia
11	North Carolina
12	New Jersey
13	Maryland
14	Arizona
15	Massachusetts
16	Indiana
17	California
18	New York
19	Colorado
20	Maine

```
#industry of parent
pharmaDF.select("Specific_Industry_of_Parent").distinct().display()
```

	Specific_Industry_of_Parent
1	pharmaceuticals
2	nutritional supplements
3	biopharmaceuticals
4	generic drugs
5	JNJ
6	animal health products
7	Missouri
8	pharmaceutical manufacturing services
9	non-opioid pain management
10	veterinary medical products

Showing all 10 rows.

```
#major industry of parent
pharmaDF.select("Major_Industry_of_Parent").distinct().display()
```

	Major_Industry_of_Parent
1	pharmaceuticals
2	USA
3	publicly traded
4	ENDP

Pyspark – creating Dataframes

“mesh.csv”, “pharma.csv” and clinical trial data were read into dataframes, with the schema automatically inferred. If datasets fail to be read into dataframes, the notebook will exit at that cell. An additional dataframe with the duplicate trials removed was also created.

```
#import functions
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import seaborn as sns
sns.set()
from pyspark.sql.functions import split, explode, col,from_unixtime, unix_timestamp, regexp_replace,lower, length, collect_set,arrays_zip,monotonically_increasing_id, months_between, to_timestamp,when
from pyspark.sql.types import TimestampType, DoubleType
```

```
#Select year as string
year="2021"
```

```
#clinical trial dataframe
#attempt to read in dataframe otherwise exit notebook
try:
    trial = spark.read.options(inferSchema="True",delimiter="|",header="True").csv("/FileStore/tables/clinicaltrial_"+year+".csv")
    trial.display()
except:
    dbutils.notebook.exit("File not found - check trial data has been imported or check year string variable")
```

	Id	Sponsor	Status	Start	Completion	Type	Subn
1	NCT02758028	The University of Hong Kong	Recruiting	Aug 2005	Nov 2021	Interventional	Apr 2
2	NCT02751957	Duke University	Completed	Jul 2016	Jul 2020	Interventional	Apr 2
3	NCT02758483	Universidade Federal do Rio de Janeiro	Completed	Mar 2017	Jan 2018	Interventional	Apr 2
4	NCT02759848	Istanbul Medeniyet University	Completed	Jan 2012	Dec 2014	Observational	May
5	NCT02758860	University of Roma La Sapienza	Active, not recruiting	Jun 2016	Sep 2020	Observational [Patient Registry]	Apr 2
6	NCT02757209	Consorzio Futuro in Ricerca	Completed	Apr 2016	Jan 2018	Interventional	Apr 2
7	NCT02752438	Ankara University	Unknown status	May 2016	Jul 2017	Observational [Patient Registry]	Apr 2

Truncated results, showing first 1000 rows.

```
#pharma dataframe
#attempt to read in dataframe otherwise exit notebook
try:
    pharma = spark.read.options(inferSchema="True",delimiter=",",header="True").csv("/FileStore/tables/pharma.csv")
    pharma.display()
except:
    dbutils.notebook.exit("File not found")
```

	Company	Parent_Company	Penalty_Amount	Subtraction_From_Penalty
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.

```
#mesh dataframe
#attempt to read in dataframe otherwise exit notebook
try:
    mesh = spark.read.options(inferSchema="True",delimiter=",",header="True").csv("/FileStore/tables/mesh.csv")
    mesh.display()
except:
    dbutils.notebook.exit("File not found")
```

	term	tree
1	Calcimycin	D03.633.100.221.173
2	A-23187	D03.633.100.221.173
3	Temefos	D02.705.400.625.800
4	Temefos	D02.705.539.345.800
5	Temefos	D02.886.300.692.800
6	Abate	D02.705.400.625.800
7	Abate	D02.705.539.345.800

Truncated results, showing first 1000 rows.

```
#dataset with removed duplicates
trial_nodupl= trial.dropDuplicates(["Sponsor", "Status", "Start", "Completion", "Type", "Submission", "Conditions", "Interventions"])
trial_nodupl.display()
```

	Id	Sponsor	Status	Start	Completion	Type
1	NCT00001540	National Institute of Allergy and Infectious Diseases (NIAID)	Completed	Apr 1996	Mar 2000	Obser
2	NCT00005048	NYU Langone Health	Completed	Apr 1997	null	Interve
3	NCT00005570	National Heart, Lung, and Blood Institute (NHLBI)	Completed	Apr 2000	Jul 2000	Obser
4	NCT00005111	National Institute of Neurological Disorders and Stroke (NINDS)	Completed	Apr 2000	Mar 2001	Obser
5	NCT00005120	Sarawak MediChem Pharmaceuticals	Unknown status	Apr 2000	null	Interve
6	NCT00005088	Radiation Therapy Oncology Group	Completed	Apr 2000	null	Interve
7	NCT00005090	Southwest Oncology Group	Terminated	Apr 2000	May 2005	Interve

Truncated results, showing first 1000 rows.

Pyspark – creating RDDs

"mesh.csv", "pharma.csv" and clinical trial data were read into RDDs, with function "split" being used to separate the columns by their respective delimiter. The first row contained headers so was placed into its own variable and then filtered out of the RDDs. If the datasets fail to be read into RDDs the notebook will exit at that cell.

```
#import packages
import numpy as np
import matplotlib.pyplot as plt
import calendar
```

```
#set year value
year="2021"
```

```
#create clinical trials RDD otherwise exit notebook
try:
    RDD = sc.textFile("/FileStore/tables/clinicaltrial_"+year+".csv").map(lambda x: x.split("|"))
    header = RDD.first()
    trialsRDD = RDD.filter(lambda x: x!=header)
except:
    dbutils.notebook.exit("File not found - check trial data has been imported or check year string variable")
```

```
#create mesh RDD otherwise exit notebook
try:
    RDD = sc.textFile("/FileStore/tables/mesh.csv")
    meshHeader = RDD.first()
    meshRDD = RDD.filter(lambda x: x!=meshHeader).map(lambda x: x.replace('"', "")).map(lambda x: x.replace(", ", " ")).map(lambda x: x.split(","))
    rootRDD = meshRDD.map(lambda x: (x[0], x[1].split("."))).map(lambda x: (x[0],x[1][0]))
except:
    dbutils.notebook.exit("File not found")
```

```
#create pharma RDD otherwise exit notebook
try:
    RDD = sc.textFile("/FileStore/tables/pharma.csv").map(lambda x: x.split(","))
    pharmaHeader = RDD.first()
    pharmaRDD = RDD.filter(lambda x: x!=pharmaHeader)
    parentCompany=pharmaRDD.map(lambda x: x[1]).map(lambda x: (x,1))
except:
    dbutils.notebook.exit("File not found")
```

An additional dataframe with the duplicate trials removed was also created. This was achieved by creating a function which joined the columns except “Id” to create a key which could be used to remove duplicates by using “reduceByKey”.

```
#remove duplicates
def get_key(x):
    return ",".join(x[1:9])

trialsRDD_nodupl=trialsRDD.map(lambda x: (get_key(x),x)).reduceByKey(lambda x,y: (x)).map(lambda x: x[1])
```

HiveQL – creating tables

A python definition was created to produce a table from a csv and save it in parquet form. If the table was not listed in the expected directory, this function was applied to each csv (“mesh.csv”, “pharma.csv” and the clinical trial dataset). The tables were then created using HiveQL. The table without duplicated trials was created by creating a column which counted duplicates.

```
%python
#function to create table from csv
def table_from_csv(filename, schema, first_row_header, delimiter):
    file_location = "/FileStore/tables/"+filename+".csv"
    df = spark.read.csv(file_location,inferSchema=schema, header = first_row_header, sep=delimiter)
    permanent_table_name = filename
    df.write.format("parquet").saveAsTable(permanent_table_name)
```

```python
%python
#input selected year
year="2021"

filename="clinicaltrial_"+year


#clinical trial
try:
    dbutils.fs.ls("dbfs:/user/hive/warehouse/"+filename)
except:
    table_from_csv(filename, True, True, "|")

#mesh
try:
    dbutils.fs.ls("dbfs:/user/hive/warehouse/mesh")
except:
    table_from_csv("mesh", True, True, ",")

#pharma
try:
    dbutils.fs.ls("dbfs:/user/hive/warehouse/pharma")
except:
    table_from_csv("pharma", True, True, ",")

#remove table without duplicates
try:
    dbutils.fs.rm("dbfs:/user/hive/warehouse/clinicaltrial_nodupl", True)
except:
    pass
```

```
--Set year variable in HIVE
SET hivevar:year=2021;
```

	key	value
1	hivevar:year	2021

Showing all 1 rows.

```
--create clinical trial table
CREATE TABLE IF NOT EXISTS clinicaltrial
    USING org.apache.spark.sql.parquet
    OPTIONS (PATH 'dbfs:/user/hive/warehouse/clinicaltrial_${hivevar:year}');
```

OK

```
--create mesh table
CREATE TABLE IF NOT EXISTS mesh
    USING org.apache.spark.sql.parquet
    OPTIONS (PATH 'dbfs:/user/hive/warehouse/mesh')
```

OK

```
--create pharma table
CREATE TABLE IF NOT EXISTS pharma
    USING org.apache.spark.sql.parquet
    OPTIONS (PATH 'dbfs:/user/hive/warehouse/pharma')
```

OK

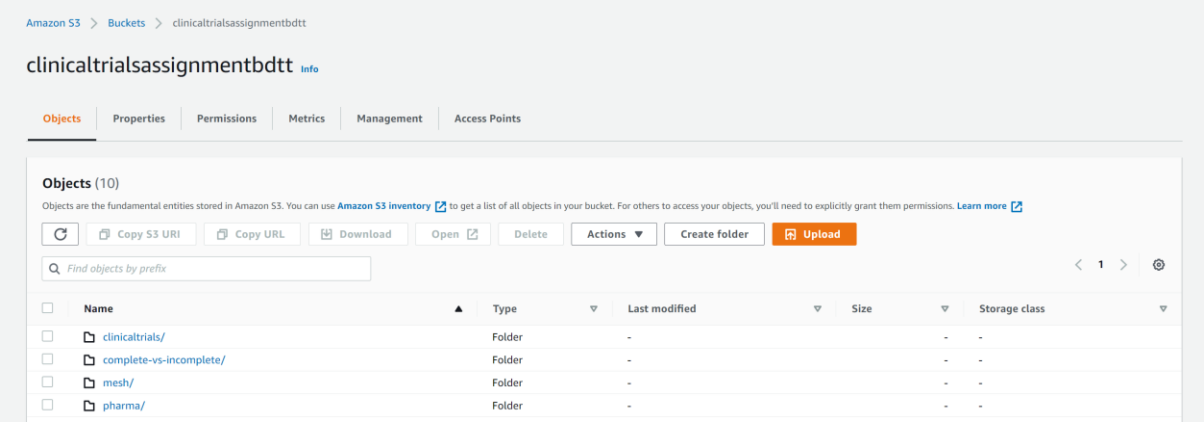
```
--removing duplicates
CREATE OR REPLACE TABLE clinicaltrial_nodupl
AS
SELECT Id,Sponsor, Status, Start, Completion, Type, Submission, Conditions, Interventions,
       ROW_NUMBER() OVER (PARTITION BY Sponsor, Status, Start, Completion, Type, Submission, Conditions, Interventions
           ORDER BY Id) AS DuplicateCount
    FROM clinicaltrial;
DELETE FROM clinicaltrial_nodupl
WHERE DuplicateCount > 1;
```

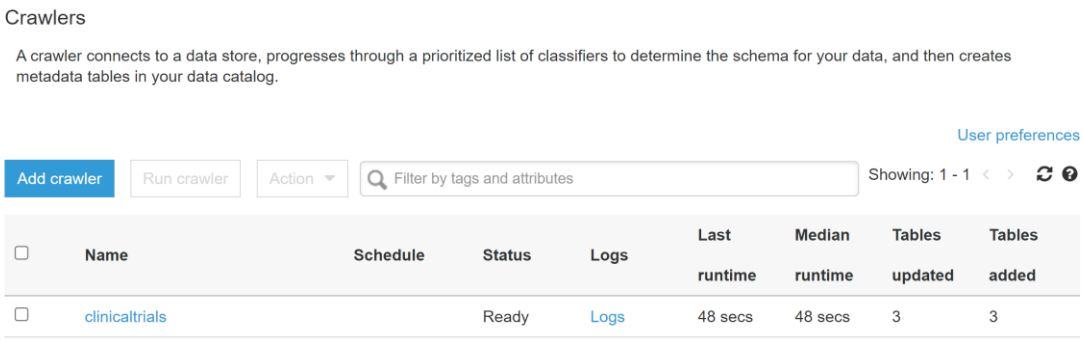
	num_affected_rows
1	1699

Showing all 1 rows.

AWS – using Glue to create tables

An S3 bucket was created containing the “mesh.csv”, “pharma.csv” and the clinical trial dataset. A crawler was used to infer the schema and create the tables. As mesh and pharma contained commas within quotations, additional changes to serde parameters were implemented to prevent additional splitting. These tables were then available in Athena. A table created with a column which counted duplicates and then a view was used to remove the duplicated trials.





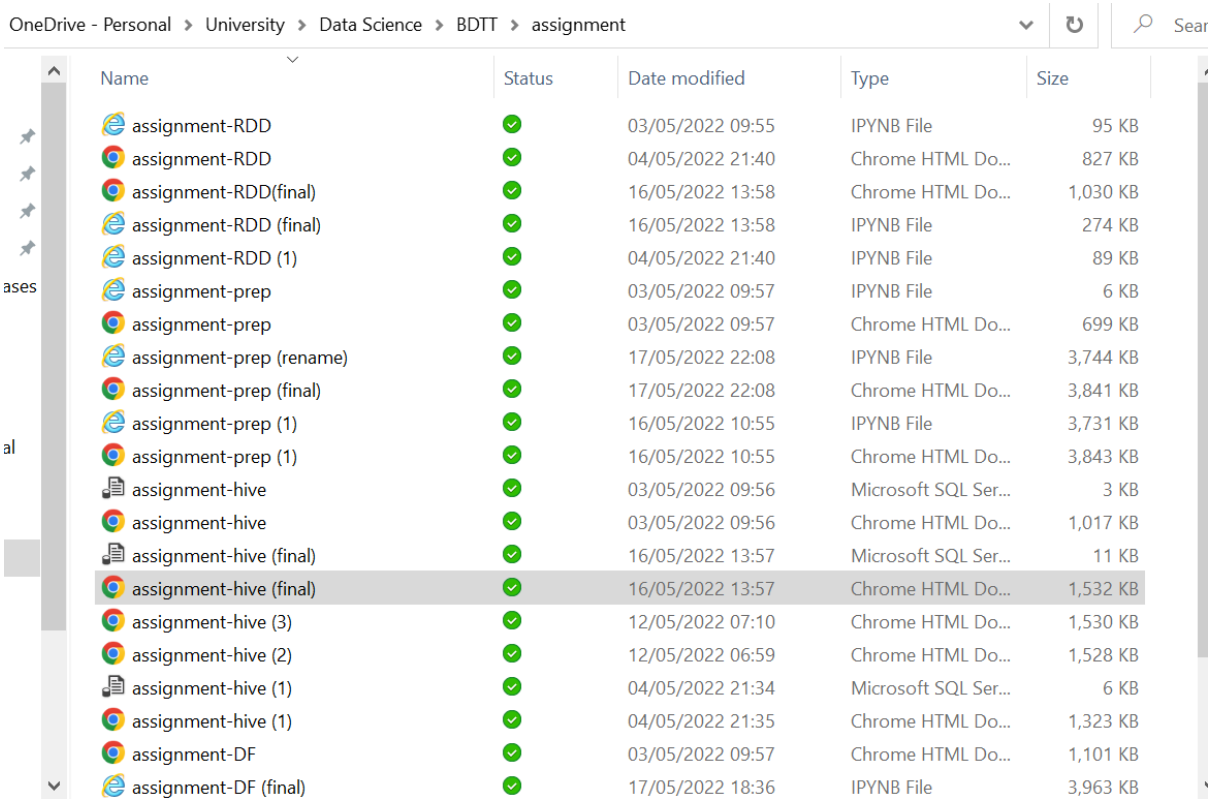
Name	mesh
Description	
Database	clinical trials
Classification	csv
Location	s3://clinicaltrialsassignmentbdtt/mesh/
Connection	
Deprecated	No
Last updated	Wed May 11 14:25:10 GMT+100 2022
Input format	org.apache.hadoop.mapred.TextInputFormat
Output format	org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Serde serialization lib	org.apache.hadoop.hive.serde2.OpenCSVSerde
Serde parameters	quoteChar " field.delim , separatorChar ,
	skip.header.line.count 1 sizeKey 5295548 objectCount 1
	UPDATED_BY_CRAWLER clinicaltrials CrawlerSchemaSerializerVersion 1.0

Name	pharma
Description	
Database	clinical trials
Classification	csv
Location	s3://clinicaltrialsassignmentbdtt/pharma/
Connection	
Deprecated	No
Last updated	Wed May 11 15:00:07 GMT+100 2022
Input format	org.apache.hadoop.mapred.TextInputFormat
Output format	org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Serde serialization lib	org.apache.hadoop.hive.serde2.OpenCSVSerde
Serde parameters	quoteChar " field.delim , separatorChar ,
	skip.header.line.count 1 sizeKey 678999 objectCount 1
	UPDATED_BY_CRAWLER clinicaltrials CrawlerSchemaSerializerVersion 1.0

```
--create table with duplicate counter
CREATE TABLE "AwsDataCatalog"."clinical trials"."clinicaltrial_duplcount"
AS
SELECT Id,Sponsor, Status, Start, Completion, Type, Submission, Conditions, Interventions,
       ROW_NUMBER() OVER (PARTITION BY Sponsor, Status, Start, Completion, Type,
           Submission, Conditions, Interventions
           ORDER BY Id) AS DuplicateCount
    FROM "AwsDataCatalog"."clinical trials"."clinicaltrials";

```

```
--remove duplicates
CREATE OR REPLACE VIEW "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl"
AS
SELECT *
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_duplcount"
WHERE DuplicateCount < 2;
```

Code was backed up using the OneDrive.



Problem Answers

1. Number of studies

The implementation for the number of studies requires the counting of distinct rows to ensure any duplicates are not counted. For Pyspark's dataframe implementation this simply requires counting distinct rows. However, for RDD implementation, the columns are joined into a key and then reduced by key, allowing any duplicate rows to be combined and then counted. In HiveQL and AWS Athena, the count function was applied to all distinct trial IDs in the table to produce the number of rows.

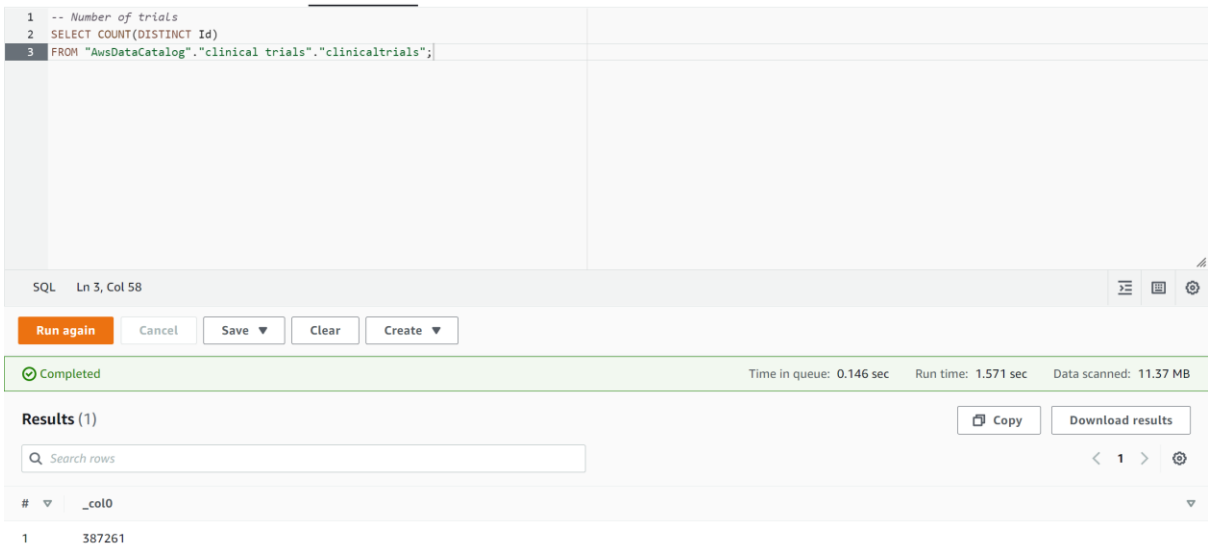
```
#initial implementation
trial.distinct().count()

Out[8]: 387261
```

```
#initial implementation
trialsRDD.map(lambda x: ",".join(x[0:9])).map(lambda x: (x,1)).reduceByKey(lambda a,b: a+b).count()

Out[8]: 387261
```

```
--initial implementation
SELECT COUNT(*) as Count
FROM clinicaltrial;
```

	Count
1	387261

Showing all 1 rows.



This implementation assumes the trial ID ('Id') indicates a unique trial, so if all ID values are unique in the data, then there are no trials recorded more than once. From data exploration, this does not appear to be the case, with trials appearing to be recorded multiple times under different ID values. These were removed and the number of distinct trials was applied again.

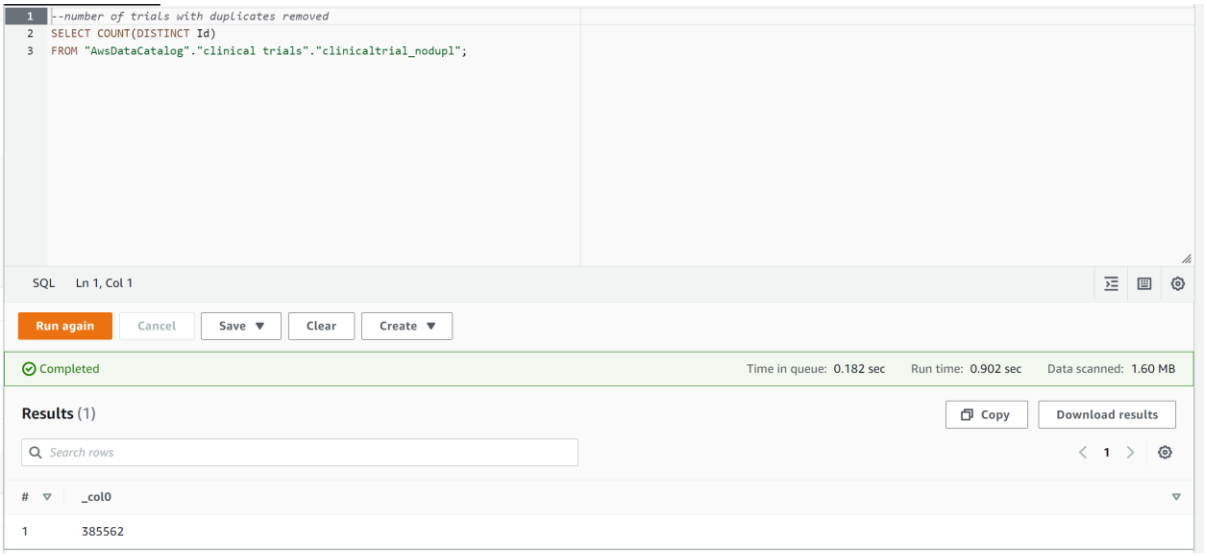
```
#without duplicates
trial_nodupl.distinct().count()

Out[9]: 385562
```

```
#without duplicates
trialsRDD_nodupl.map(lambda x: ",".join(x[0:9])).map(lambda x: (x,1)).reduceByKey(lambda a,b: a+b).count()

Out[9]: 385562
```

```
--no duplicates
SELECT COUNT(*) as Count
FROM clinicaltrial_nodupl;
```

	Count
1	385562

Showing all 1 rows.



In the dataset there are 387,361 trials recorded using the initial implementation. With potential duplicate trials removed there are 385,562 unique trials recorded.

However, if only considering trials with the status of completed, this changes to 208,722 trials. This is a large enough dataset to analyse.

```
#number of completed verse uncompleted trials
print("Total trials: "+ str(trial_nodupl.distinct().count()))
print("Completed trials: " + str(trial_nodupl.filter(col("Status")=="Completed").distinct().count()))
print("Incomplete trials: "+ str(trial_nodupl.filter(col("Status")!="Completed").distinct().count()))
```

```
Total trials: 385562
Completed trials: 208722
Incomplete trials: 176840
```

```
#number of completed verse uncompleted trials
total=trialsRDD_nodupl.map(lambda x: ",".join(x[0:9])).map(lambda x: (x,1)).reduceByKey(lambda a,b: a+b).count()
incomplete=trialsRDD_nodupl.filter(lambda x: x[2]!="Completed").map(lambda x: ",".join(x[0:9])).map(lambda x: (x,1)).reduceByKey(lambda a,b: a+b).count()
complete=trialsRDD_nodupl.filter(lambda x: x[2]=="Completed").map(lambda x: ",".join(x[0:9])).map(lambda x: (x,1)).reduceByKey(lambda a,b: a+b).count()
sc.parallelize([("Total",total),("Complete",complete),("Incomplete",incomplete)]).collect()
```

```
Out[10]: [('Total', 385562), ('Complete', 208722), ('Incomplete', 176840)]
```

```
--number of completed verse uncompleted trials
SELECT COUNT(*) as Total, (SELECT COUNT(Status)
FROM clinicaltrial_nodupl
WHERE Status=='Completed')as Completed, (SELECT COUNT(Status)
FROM clinicaltrial_nodupl
WHERE Status!='Completed') as Incomplete
FROM clinicaltrial_nodupl;
```

	Total	Completed	Incomplete
1	385562	208722	176840

Showing all 1 rows.

```
--number of completed verse uncompleted trials
SELECT COUNT(*) as Total, (SELECT COUNT(Status)
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl"
WHERE Status='Completed')as Completed, (SELECT COUNT(Status)
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl"
WHERE Status!='Completed') as Incomplete
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl";
```

SQL Ln 6, Col 26

Run again | Cancel | Save | Clear | Create

Completed — Time in queue: 0.123 sec — Run time: 0.906 sec — Data scanned: 134.28 KB

Results (1) — Copy | Download results

Search rows — 1

#	Total	Completed	Incomplete
1	385562	208722	176840

2. Types of studies with their frequencies

For the initial implementation in dataframes, HiveQL and AWS Athena, the column ‘Type’ was selected and counted by grouping types. For RDD, type was placed into a pair-key and then reduced by key to count the frequency of each.

```
#initial implementation
typeFreq = trial.select(trial.Type).groupBy("Type").count().orderBy("count", ascending=False)
typeFreq.display()
```

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

Showing all 4 rows.

```
#initial implementation
type = trialsRDD.map(lambda x: (x[5], 1)).reduceByKey(lambda a,b: a+b).sortBy(lambda x: x[1], False).collect()
type

Out[11]: [('Interventional', 301472),
 ('Observational', 77540),
 ('Observational [Patient Registry]', 8180),
 ('Expanded Access', 69)]
```

```
--initial implementation
SELECT Type, COUNT(Type) AS Count
FROM clinicaltrial
GROUP BY Type
ORDER BY Count DESC;
```

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

Showing all 4 rows.

```sql
--number of each type of trials
--initial implementation
SELECT Type, COUNT(Type) AS Count
FROM "AwsDataCatalog"."clinical trials"."clinicaltrials"
GROUP BY Type
ORDER BY Count DESC;
```

SQL Ln 1, Col 1

Run again | Cancel | Save | Clear | Create

Completed — Time in queue: 0.192 sec — Run time: 1.372 sec — Data scanned: 11.37 MB

Results (4) — Copy | Download results

Search rows — 1

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

This was also repeated for the dataset without duplicates, which affected all trial types, except ‘Expanded Access’, but did not change the order of frequency.

```
#without duplicates
typeFreq2 = trial_nodupl.select("Type").groupBy("Type").count().orderBy("count", ascending=False)
typeFreq2.display()
```

	Type	count
1	Interventional	300082
2	Observational	77249
3	Observational [Patient Registry]	8162
4	Expanded Access	69

Showing all 4 rows.

```
#without duplicates
type2 = trialsRDD_nodupl.map(lambda x: (x[5], 1)).reduceByKey(lambda a,b: a+b).sortBy(lambda x: x[1], False)
type2.collect()

Out[12]: [('Interventional', 300082),
 ('Observational', 77249),
 ('Observational [Patient Registry]', 8162),
 ('Expanded Access', 69)]
```

```sql
--without duplicates
SELECT Type, COUNT(Type) AS Count
FROM clinicaltrial_nodupl
GROUP BY Type
ORDER BY Count DESC;
```

	Type	Count
1	Interventional	300082
2	Observational	77249
3	Observational [Patient Registry]	8162
4	Expanded Access	69

Showing all 4 rows.

```sql
--number of each type of trials
--without duplicates
SELECT Type, COUNT(Type) AS Count
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl"
GROUP BY Type
ORDER BY Count DESC;
```

SQL Ln 1, Col 1

Run again | Cancel | Save | Clear | Create

Completed — Time in queue: 0.186 sec — Run time: 0.894 sec — Data scanned: 75.37 KB

Results (4) — Copy | Download results

Search rows — 1

#	Type	Count
1	Interventional	300082
2	Observational	77249
3	Observational [Patient Registry]	8162
4	Expanded Access	69

There are four types of trials in this dataset: observational, observational [patient registry], interventional and expanded access. Interventional studies are the most popular trial type and expanded access studies are least used. Observational [Patient registry] is a type of observational study; depending on particular interests, these two columns could be combined into one observational column (which would involve changing rows containing ‘Observation [Patient registry]’ to ‘Observation’). However, if interested in compared conditions observed using the patient registry verses in an observational study with participants, they should be kept separate.

Despite interventional being the most popular, approximately 58% of interventional studies do not have interventions recorded. A noted previously, recorded interventions are mostly drug related. It is possible for interventions to be non-drug related, such as education or lifestyle changes. It is unclear whether the missing interventions is due to incomplete records or involved non-drug related interventions. This is particularly true for observational studies, which can have interventions or not. Any investigation into relating conditions to their intervention would not be reliable, therefore.

```
#trials with no interventions
no_interventions = trial_nodupl.select("Type", "Interventions").filter(col("Interventions").isNull()).groupBy("Type").count().withColumnRenamed("count","no intervention").withColumnRenamed("Type","Type2")
#trials with interventions
interventions = trial_nodupl.select("Type", "Interventions").filter(col("Interventions").isNotNull()).groupBy("Type").count().withColumnRenamed("count","intervention").withColumnRenamed("Type","Type3")
#join and get percentage
typeFreq3 = typeFreq2.join(no_interventions,no_interventions["Type2"]==typeFreq2["Type"]).join(interventions,interventions["Type3"]==typeFreq2["Type"]).withColumn("intervention (percentage)",
(col("intervention")/col("count")*100)).withColumn("no intervention (percentage)",(col("no intervention")/col("count")*100)).select("Type","count","no intervention","no intervention (percentage)","intervention",
"intervention (percentage)").withColumnRenamed("count","total").orderBy("total", ascending=False)

typeFreq3.display()
```

	Type	total	no intervention	no intervention (percentage)	intervention	intervention (percentage)
1	Interventional	300082	174363	58.10511793443126	125719	41.894882065568744
2	Observational	77249	70768	91.61024738184314	6481	8.38975261815687
3	Observational [Patient Registry]	8162	7569	92.73462386669934	593	7.265376133300662
4	Expanded Access	69	32	46.3768115942029	37	53.62318840579711

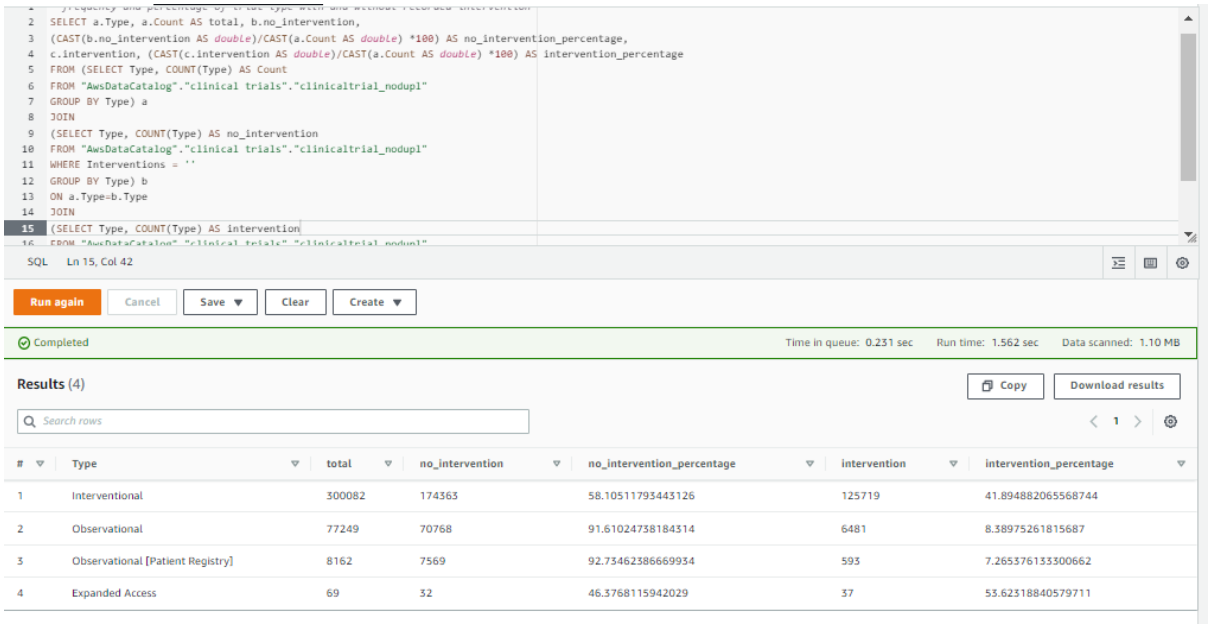
```
#percentage of trials with no interventions
trialsRDD_nodupl.filter(lambda x: x[8]=="").map(lambda x: (x[5], 1)).reduceByKey(lambda a,b: a+b).join(type2).sortBy(lambda x: x[1][1], False).map(lambda x: (x[0],(x[1][0]/x[1][1]*100))).collect()

Out[13]: [('Interventional', 58.10511793443126),
 ('Observational', 91.61024738184314),
 ('Observational [Patient Registry]', 92.73462386669934),
 ('Expanded Access', 46.3768115942029)]
```

```
#percentage of trials with interventions
trialsRDD_nodupl.filter(lambda x: x[8]!="").map(lambda x: (x[5], 1)).reduceByKey(lambda a,b: a+b).join(type2).sortBy(lambda x: x[1][1], False).map(lambda x: (x[0],(x[1][0]/x[1][1]*100))).collect()

Out[14]: [('Interventional', 41.894882065568744),
 ('Observational', 8.38975261815687),
 ('Observational [Patient Registry]', 7.265376133300662),
 ('Expanded Access', 53.62318840579711)]
```

```sql
--trial type with and without interventions
SELECT a.Type, a.Count AS total, b.no_intervention,
(b.no_intervention/a.Count*100) AS no_intervention_percentage,
c.intervention, (c.intervention/a.Count*100) AS intervention_percentage
FROM (SELECT Type, COUNT(Type) AS Count
FROM clinicaltrial_nodupl
GROUP BY Type) a
JOIN
(SELECT Type, COUNT(Type) AS no_intervention
FROM clinicaltrial_nodupl
WHERE Interventions IS NULL
GROUP BY Type) b
ON a.Type==b.Type
JOIN
(SELECT Type, COUNT(Type) AS intervention
FROM clinicaltrial_nodupl
WHERE Interventions IS NOT NULL
GROUP BY Type) c
ON a.Type==c.Type
ORDER BY total DESC;
```

	Type	total	no_intervention	no_intervention_percentage	intervention	intervention_percentage
1	Interventional	300082	174363	58.10511793443126	125719	41.894882065568744
2	Observational	77249	70768	91.61024738184314	6481	8.38975261815687
3	Observational [Patient Registry]	8162	7569	92.73462386669934	593	7.265376133300662
4	Expanded Access	69	32	46.3768115942029	37	53.62318840579711

Showing all 4 rows.



3. Top five conditions with their frequencies

For initial implementation in dataframes and HiveQL, the ‘Conditions’ column was selected and split into separate rows using ‘explode’, then counted by grouping the conditions. In AWS Athena cross joins and unnesting was applied to splitting conditions to record individual conditions on separate rows. In RDD, conditions were split by ‘,’ and then flatmapped to separate them, they were placed into pair-keys and reduced by key to obtain the frequency. These were all sorted by frequency and the top five selected.

```
#initial implementation
conditions = trial.select("Conditions").withColumn("Conditions",explode(split("Conditions",","))).groupBy("Conditions").count().orderBy("count", ascending=False)
conditions.limit(5).display()
```

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

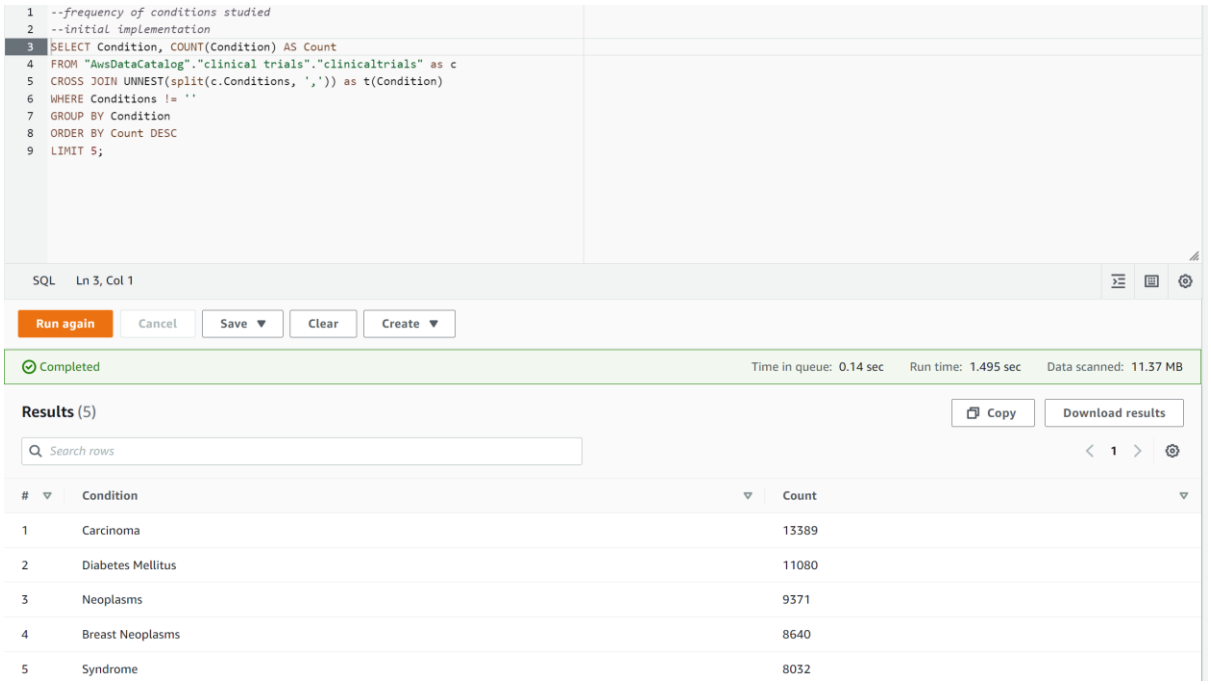
Showing all 5 rows.

```
#initial implementation
condition = trialsRDD.map(lambda x: x[7].split(",")).flatMap(lambda x: x).filter(lambda x: x!= "").map(lambda x: (x,1)).reduceByKey(lambda a,b: a+b).sortBy(lambda x: x[1], False)
condition.take(5)

Out[16]: [('Carcinoma', 13389),
 ('Diabetes Mellitus', 11080),
 ('Neoplasms', 9371),
 ('Breast Neoplasms', 8640),
 ('Syndrome', 8032)]
```

```
--initial implementation
SELECT Condition, COUNT(Condition) AS Count
FROM clinicaltrial
LATERAL VIEW explode(split(Conditions, ',')) Conditions as Condition
GROUP BY Condition
ORDER BY Count DESC
LIMIT 5;
```

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

Showing all 5 rows.



This was repeated for the dataset without duplicated trials; however, the top five conditions did not change. In HiveQL and Athena, these were saved as views for further implementation.

```
#without duplicates
conditions2 = trial_nodupl.select("Conditions").withColumn("Conditions",explode(split("Conditions",","))).groupBy("Conditions").count().orderBy("count", ascending=False)
conditions2.limit(5).display()
```

	Conditions	count
1	Carcinoma	13360
2	Diabetes Mellitus	11029
3	Neoplasms	9349
4	Breast Neoplasms	8629
5	Syndrome	8018

Showing all 5 rows.

```
#withoutduplicates
condition2 = trialsRDD_nodupl.map(lambda x: x[7].split(",")).flatMap(lambda x: x).filter(lambda x: x!= "").map(lambda x: (x,1)).reduceByKey(lambda a,b: a+b).sortBy(lambda x: x[1], False)
condition2.take(5)

Out[17]: [('Carcinoma', 13360),
 ('Diabetes Mellitus', 11029),
 ('Neoplasms', 9349),
 ('Breast Neoplasms', 8629),
 ('Syndrome', 8018)]
```

```
--without duplicates
CREATE OR REPLACE VIEW ConditionCount AS
SELECT Condition, COUNT(Condition) AS Count
FROM clinicaltrial_nodupl
LATERAL VIEW explode(split(Conditions, ',')) Conditions as Condition
GROUP BY Condition;

SELECT * FROM ConditionCount
ORDER BY Count DESC
LIMIT 5;
```

	Condition	Count
1	Carcinoma	13360
2	Diabetes Mellitus	11029
3	Neoplasms	9349
4	Breast Neoplasms	8629
5	Syndrome	8018

Showing all 5 rows.

```
--frequency of conditions studied
--without duplicates
CREATE OR REPLACE VIEW ConditionCount AS
SELECT Condition, COUNT(Condition) AS Count
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl" as c
CROSS JOIN UNNEST(split(c.Conditions, ',')) as t(Condition)
WHERE Conditions != ''
GROUP BY Condition
ORDER BY Count DESC;
```

```
1 SELECT * FROM "clinical trials"."conditioncount" limit 5;
```

SQL Ln 1, Col 58

Run again | Cancel | Save | Clear | Create

Completed — Time in queue: 0.156 sec — Run time: 1.938 sec — Data scanned: 3.45 MB

Results (5) — Copy | Download results

Search rows — 1

#	Condition	Count
1	Carcinoma	13360
2	Diabetes Mellitus	11029
3	Neoplasms	9349
4	Breast Neoplasms	8629
5	Syndrome	8018

Carcinomas are the most studied condition, followed by Diabetes Mellitus. Neoplasms appears twice, being the third most studied condition and fourth with ‘Breast Neoplasms’. It is possible conditions may be similar to be grouped, such as the neoplasms. Without medical knowledge this cannot be concluded based on name alone. The mesh data contains some terms which contain the root as their tree value (e.g., the term ‘Cancer’ having tree ‘C04’). These could be overall terms which conditions can be grouped under due to relatedness and used to group individual conditions together. A new column containing the root for each tree was added to the mesh data and joined to the table of counted conditions based on matching conditions and terms. The table of counted conditions was joined to the original mesh table using the root column and tree column. There appeared to be multiple terms with the same tree value so were aggregated into one row.

```
#include groups based on root
rootdf = mesh.withColumn("root", split(mesh["tree"], "\.").getItem(0))
conditionRoot = conditions2.join(rootdf,rootdf["term"]==conditions2["Conditions"], "left").select("Conditions","count","root")
conditionGroup = conditionRoot.join(mesh,mesh["tree"]==conditionRoot["root"], "left").groupBy("Conditions","count").agg(collect_set("term").alias("group")).orderBy("count", ascending=False)
conditionGroup.limit(5).display()
```

	Conditions	count	group
1	Carcinoma	13360	["Neoplasms", "Benign Neoplasms", "Cancer"]
2	Diabetes Mellitus	11029	["Nutritional and Metabolic Diseases", "Endocrine System Diseases"]
3	Neoplasms	9349	["Neoplasms", "Benign Neoplasms", "Cancer"]
4	Breast Neoplasms	8629	["Skin and Connective Tissue Diseases", "Neoplasms", "Benign Neoplasms", "Cancer"]
5	Syndrome	8018	["Pathological Conditions, Signs and Symptoms"]

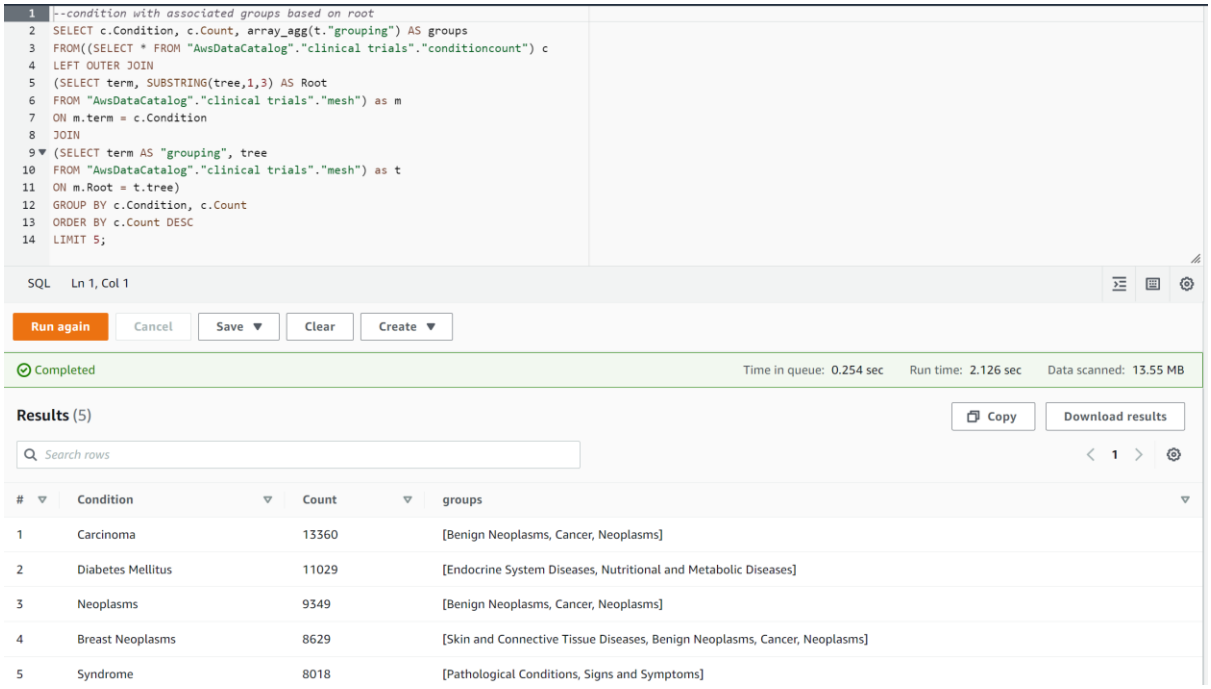
Showing all 5 rows.

```
#include groups based on root
conditionRoot = rootRDD.join(condition2).map(lambda x: (x[1][0],(x[0],x[1][1])))
groupRDD = meshRDD.map(lambda x: (x[1],x[0])).join(conditionRoot).map(lambda x: (x[1][1][0],x[1][0])).reduceByKey(lambda a,b: str(a)+", "+str(b)).join(condition2).sortBy(lambda x: x[1][1], False)
groupRDD.take(5)

Out[18]: [('Carcinoma', ('Neoplasms, Cancer, Benign Neoplasms', 13360)),
 ('Diabetes Mellitus',
  ('Endocrine System Diseases, Nutritional and Metabolic Diseases', 11029)),
 ('Neoplasms', ('Neoplasms, Cancer, Benign Neoplasms', 9349)),
 ('Breast Neoplasms',
  ('Neoplasms, Cancer, Benign Neoplasms, Skin and Connective Tissue Diseases',
   8629)),
 ('Syndrome', ('Pathological Conditions Signs and Symptoms', 8018))]
```

```sql
--condition with associated groups based on root
SELECT Condition, Count, collect_set(group) AS groups
FROM((SELECT * FROM ConditionCount) c
LEFT OUTER JOIN
(SELECT term, SUBSTRING(tree,1,3) AS Root
FROM mesh) as m
ON m.term == c.Condition
JOIN
(SELECT term AS group, tree
FROM mesh) as t
ON m.Root == t.tree)
GROUP BY Condition, Count
ORDER BY Count DESC
LIMIT 5;
```

	Condition	Count	groups
1	Carcinoma	13360	["Neoplasms", "Benign Neoplasms", "Cancer"]
2	Diabetes Mellitus	11029	["Nutritional and Metabolic Diseases", "Endocrine System Diseases"]
3	Neoplasms	9349	["Neoplasms", "Benign Neoplasms", "Cancer"]
4	Breast Neoplasms	8629	["Skin and Connective Tissue Diseases", "Neoplasms", "Benign Neoplasms", "Cancer"]
5	Syndrome	8018	["Pathological Conditions, Signs and Symptoms"]

Showing all 5 rows.



It appears Carcinomas, Neoplasms and Breast Neoplasms belong to the group of Neoplasms, Benign neoplasms, and Cancer. Therefore, cancers are the most studied conditions overall due three of the top conditions belonging to this group. Conditions could instead be represented by groupings and counted by similar groupings. This may be slightly complicated to implement, as some conditions overlap in groups they belong to (Breast Neoplasms contains the additional grouping of ‘Skin and Connective Tissue Diseases’) and so domain expertise would be required to decide which grouping is more important to be assigned to. Additionally, conditions with the same grouping studied in the same trial can be counted as one incidence rather than counted separately.

The most frequent conditions in completed trials may be different. The same implementation was applied with trials containing the status “Completed”. This

revealed Diabetes Mellitus is the most popular condition in completed trials, followed by Carcinoma and then Syndrome. Perhaps trials involving cancers take longer to complete or more frequently withdrawn/cancelled.

```
#top 5 conditions with group for completed trials only
conditions3=trial_nodupl.filter(col("Status")=="Completed").select("Conditions").withColumn("Conditions",explode(split("Conditions",",").groupBy("Conditions").count().orderBy("count", ascending=False)
conditionRoot2 = conditions3.join(rootdf,rootdf["term"]==conditions3["Conditions"], "left").select("Conditions","count","root")
conditionCompleted = conditionRoot2.join(mesh,mesh["tree"]==conditionRoot2["root"], "left").groupBy("Conditions","count").agg(collect_set("term").alias("group")).orderBy("count", ascending=False)
conditionCompleted.limit(5).display()
```

	Conditions	count	group
1	Diabetes Mellitus	7241	["Nutritional and Metabolic Diseases", "Endocrine System Diseases"]
2	Carcinoma	4903	["Neoplasms", "Benign Neoplasms", "Cancer"]
3	Syndrome	4181	["Pathological Conditions, Signs and Symptoms"]
4	Breast Neoplasms	3894	["Skin and Connective Tissue Diseases", "Neoplasms", "Benign Neoplasms", "Cancer"]
5	Neoplasms	3814	["Neoplasms", "Benign Neoplasms", "Cancer"]

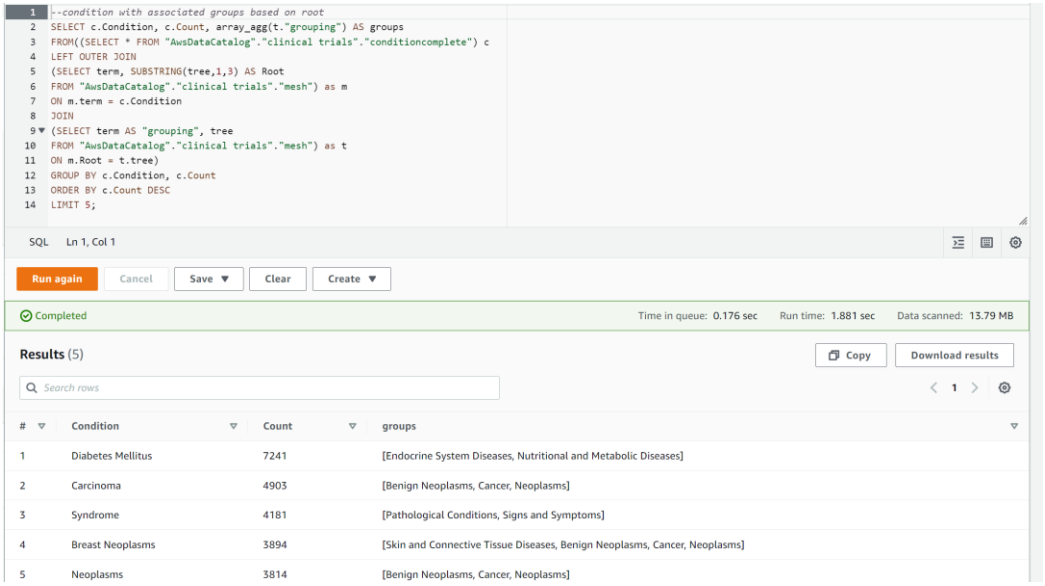
Showing all 5 rows.

```
#top 5 conditions for completed trials with groups
conditionComplete = trialsRDD_nodupl.filter(lambda x: x[2]=="Completed").map(lambda x: x[7].split(",")).flatMap(lambda x: x).filter(lambda x: x!= "").map(lambda x: (x,1)).reduceByKey(lambda a,b:
a+b).sortBy(lambda x: x[1], False)
conditionRoot = rootRDD.join(conditionComplete).map(lambda x: (x[1][0],(x[0],x[1][1])))
groupComplete = meshRDD.map(lambda x: (x[1],x[0])).join(conditionRoot).map(lambda x: (x[1][1][0],x[1][0])).reduceByKey(lambda a,b: str(a)+", "+str(b)).join(conditionComplete).sortBy(lambda x: x[1][1], False)
groupComplete.take(5)

Out[19]: [('Diabetes Mellitus',
  ('Endocrine System Diseases, Nutritional and Metabolic Diseases', 7241)),
 ('Carcinoma', ('Neoplasms, Cancer, Benign Neoplasms', 4903)),
 ('Syndrome', ('Pathological Conditions Signs and Symptoms', 4181)),
 ('Breast Neoplasms',
  ('Neoplasms, Cancer, Benign Neoplasms, Skin and Connective Tissue Diseases',
   3894)),
 ('Neoplasms', ('Neoplasms, Cancer, Benign Neoplasms', 3814))]
```

```
--- top 5 conditions with group for completed trials only
CREATE OR REPLACE VIEW ConditionComplete AS
SELECT Condition, COUNT(Condition) AS Count
FROM clinicaltrial_nodupl
LATERAL VIEW explode(split(Conditions, ',')) Conditions as Condition
WHERE Status=='Completed'
GROUP BY Condition;

SELECT Condition, Count, collect_set(group) AS groups
FROM((SELECT * FROM ConditionComplete) c
LEFT OUTER JOIN
(SELECT term, SUBSTRING(tree,1,3) AS Root
FROM mesh) as m
ON m.term == c.Condition
JOIN
(SELECT term AS group, tree
FROM mesh) as t
ON m.Root == t.tree)
GROUP BY Condition, Count
ORDER BY Count DESC
LIMIT 5;
```

	Condition	Count	groups
1	Diabetes Mellitus	7241	["Nutritional and Metabolic Diseases", "Endocrine System Diseases"]
2	Carcinoma	4903	["Neoplasms", "Benign Neoplasms", "Cancer"]
3	Syndrome	4181	["Pathological Conditions, Signs and Symptoms"]
4	Breast Neoplasms	3894	["Skin and Connective Tissue Diseases", "Neoplasms", "Benign Neoplasms", "Cancer"]
5	Neoplasms	3814	["Neoplasms", "Benign Neoplasms", "Cancer"]

```
--frequency of conditions studied
--completed trials
CREATE OR REPLACE VIEW ConditionComplete AS
SELECT Condition, COUNT(Condition) AS Count
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl" as c
CROSS JOIN UNNEST(split(c.Conditions, ',')) as t(Condition)
WHERE Conditions != '' AND Status = 'Completed'
GROUP BY Condition
ORDER BY Count DESC
LIMIT 5;
```



```
--condition with associated groups based on root
SELECT c.Condition, c.Count, array_agg(t."grouping") AS groups
FROM((SELECT * FROM "AwsDataCatalog"."clinical trials"."conditioncomplete") c
LEFT OUTER JOIN
(SELECT term, SUBSTRING(tree,1,3) AS Root
FROM "AwsDataCatalog"."clinical trials"."mesh") as m
ON m.term = c.Condition
JOIN
(SELECT term AS "grouping", tree
FROM "AwsDataCatalog"."clinical trials"."mesh") as t
ON m.Root = t.tree)
GROUP BY c.Condition, c.Count
ORDER BY c.Count DESC
LIMIT 5;
```

SQL Ln 1, Col 1

Run again | Cancel | Save | Clear | Create

Completed — Time in queue: 0.176 sec — Run time: 1.881 sec — Data scanned: 13.79 MB

Results (5) — Copy | Download results

Search rows — 1

#	Condition	Count	groups
1	Diabetes Mellitus	7241	[Endocrine System Diseases, Nutritional and Metabolic Diseases]
2	Carcinoma	4903	[Benign Neoplasms, Cancer, Neoplasms]
3	Syndrome	4181	[Pathological Conditions, Signs and Symptoms]
4	Breast Neoplasms	3894	[Skin and Connective Tissue Diseases, Benign Neoplasms, Cancer, Neoplasms]
5	Neoplasms	3814	[Benign Neoplasms, Cancer, Neoplasms]

It is important to note 16.73% of trials did not have conditions recorded, so the dataset may not be complete and not all conditions represented.

```
#no condition percentage
no_condition = trial_nodupl.filter(col("Conditions").isNull()).count()
total=trial_nodupl.distinct().count()
no_condition/total*100
```

```
Out[19]: 16.731161265892386
```

```
#percentage of data with no condition recorded
(trialsRDD_nodupl.filter(lambda x: x[7]=="").map(lambda x: ",".join(x[0:9])).map(lambda x: (x,1)).reduceByKey(lambda a,b: a+b).count())/(trialsRDD_nodupl.map(lambda x: ",".join(x[0:9])).map(lambda x: (x,1)).reduceByKey(lambda a,b: a+b).count())*100
```

```
Out[20]: 16.731161265892386
```

```
--percentage of trials without recorded conditions
SELECT (b.no_condition/a.Count*100) AS no_condition_percentage
FROM (SELECT COUNT(*) AS Count
FROM clinicaltrial_nodupl) a
JOIN
(SELECT COUNT(*) AS no_condition
FROM clinicaltrial_nodupl
WHERE Conditions IS NULL) b ;
```

	no_condition_percentage
1	16.731161265892386

Showing all 1 rows.

```sql
--percentage of trials without recorded conditions
SELECT (CAST(no_condition as double)/CAST("Count" as double )*100) AS no_condition_percentage
FROM (SELECT COUNT(*) AS Count
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl") a
CROSS JOIN
(SELECT COUNT(*) AS no_condition
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl"
WHERE Conditions = '') b;
```

SQL Ln 1, Col 1

Run again | Cancel | Save | Clear | Create

Completed Time in queue: 0.262 sec

Results (1)

Search rows

#	no_condition_percentage
1	16.731161265892386

4. Five most frequent roots

The same process used previously to split the conditions column into one condition per row was repeated. The root of the tree column in the mesh dataset was separated into its own column either by selecting the first three characters or splitting the string using '.'. These datasets/tables were joined using left join on the condition column in trial data and the term column in mesh data. Conditions with null terms were removed and the rest were counted according to their root.

```
#initial implementation
rootdf = mesh.withColumn("root", split(mesh["tree"], "\.").getItem(0))

condition = trial.select("Conditions").withColumn("Conditions",explode(split("Conditions",",")))

root = condition.join(rootdf,rootdf["term"]==condition["Conditions"], "left").filter(col("term").isNotNull()).groupBy("root").count().orderBy("count", ascending=False)
root.limit(5).display()
```

	root	count
1	C04	143994
2	C23	136079
3	C01	106674
4	C14	94523
5	C10	92310

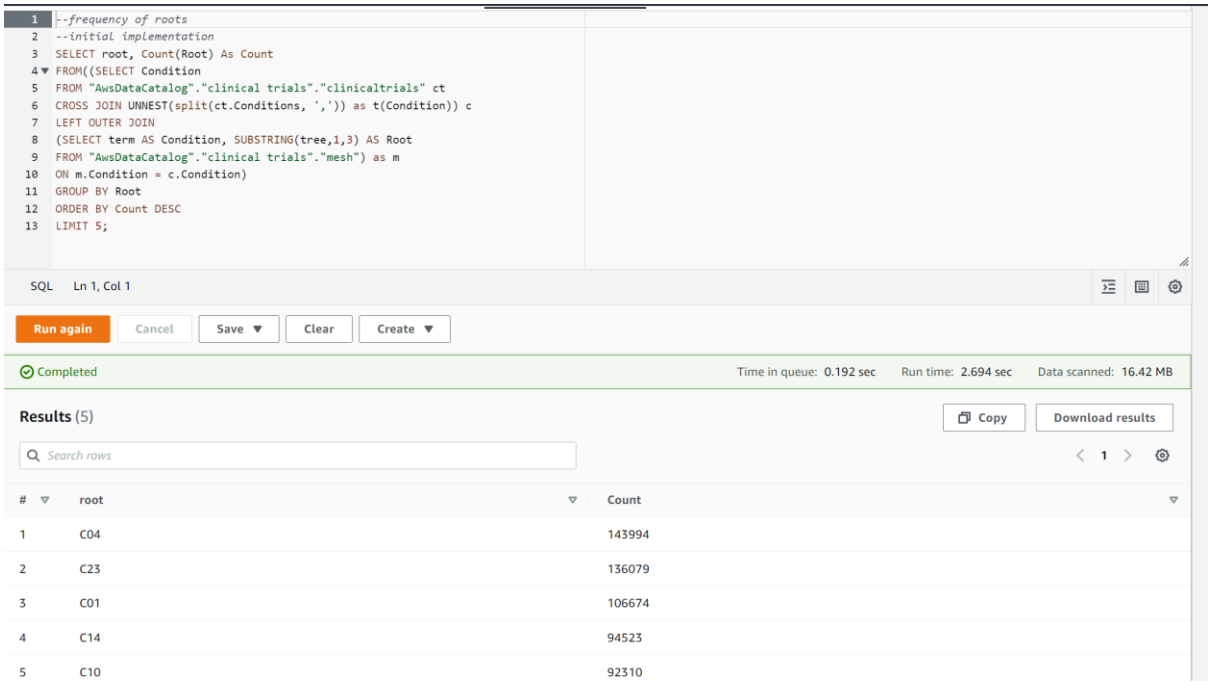
Showing all 5 rows.

```
#initial implementation
conditionRDD = trialsRDD.map(lambda x: x[7].split(",")).flatMap(lambda x: x).filter(lambda x: x!= "").map(lambda x: (x,1))
joinRDD = rootRDD.join(conditionRDD).map(lambda x: (x[1][0],x[1][1])).filter(lambda x: x is not None).reduceByKey(lambda a,b: a+b).sortBy(lambda x: x[1], False)
joinRDD.take(5)

Out[21]: [('C04', 143994),
 ('C23', 136079),
 ('C01', 106674),
 ('C14', 94523),
 ('C10', 92310)]
```

```
--initial implementation
SELECT root, Count(Root) As Count
FROM((SELECT Condition
FROM clinicaltrial
LATERAL VIEW explode(split(Conditions, ',')) Conditions as Condition) c
LEFT OUTER JOIN
(SELECT term AS Condition, SUBSTRING(tree,1,3) AS Root
FROM mesh) as m
ON m.Condition == c.Condition)
GROUP BY Root
ORDER BY Count DESC
LIMIT 5;
```

	root	Count
1	C04	143994
2	C23	136079
3	C01	106674
4	C14	94523
5	C10	92310

Showing all 5 rows.



This was repeated using the dataset without duplicates, but the top five roots did not change. In HiveQL and Athena, these were saved as views for further implementation.

```
#without duplicates
condition2 = trial_nodupl.select("Conditions").withColumn("Conditions",explode(split("Conditions",",")))
root2 = condition2.join(rootdf,rootdf["term"]==condition2["Conditions"], "left").filter(col("term").isNotNull()).groupBy("root").count().orderBy("count", ascending=False)
root2.limit(5).display()
```

	root	count
1	C04	143723
2	C23	135748
3	C01	106190
4	C14	94252
5	C10	92096

Showing all 5 rows.

```
#without duplicates
conditionRDD2 = trialsRDD_nodupl.map(lambda x: x[7].split(",")).flatMap(lambda x: x).filter(lambda x: x!= "").map(lambda x: (x,1))
joinRDD2 = rootRDD.join(conditionRDD2).map(lambda x: (x[1][0],x[1][1])).filter(lambda x: x is not None).reduceByKey(lambda a,b: a+b).sortBy(lambda x: x[1], False)
joinRDD2.take(5)

Out[22]: [('C04', 143723),
 ('C23', 135748),
 ('C01', 106190),
 ('C14', 94252),
 ('C10', 92096)]
```

```
--without duplicates
CREATE OR REPLACE VIEW CountRoot AS
SELECT root, Count(Root) As count
FROM((SELECT Condition
FROM clinicaltrial_nodupl
LATERAL VIEW explode(split(Conditions, ',')) Conditions as Condition) c
LEFT OUTER JOIN
(SELECT term AS Condition, SUBSTRING(tree,1,3) AS Root
FROM mesh) as m
ON m.Condition == c.Condition)
GROUP BY Root;

SELECT * FROM CountRoot
ORDER BY Count DESC
LIMIT 5;
```

	root	count
1	C04	143723
2	C23	135748
3	C01	106190
4	C14	94252
5	C10	92096

Showing all 5 rows.

```
--frequency of roots
--without duplicates
CREATE OR REPLACE VIEW RootCount AS
SELECT root, Count(Root) As Count
FROM((SELECT Condition
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl" ct
CROSS JOIN UNNEST(split(ct.Conditions, ',')) as t(Condition)) c
LEFT OUTER JOIN
(SELECT term AS Condition, SUBSTRING(tree,1,3) AS Root
FROM "AwsDataCatalog"."clinical trials"."mesh") as m
ON m.Condition = c.Condition)
GROUP BY Root
ORDER BY Count DESC;
```

```
SELECT * FROM "clinical trials"."rootcount" limit 5;
```

SQL Ln 1, Col 52

Run again | Cancel | Save | Clear | Create

Completed Time in queue: 0.196 sec

Results (5)

Search rows

#	root	Count
1	C04	143723
2	C23	135748
3	C01	106190
4	C14	94252
5	C10	92096

The top five roots are C04, C23, C01, C14 and C10. This does not tell us the meaning of the roots. It was attempted to include the conditions in the trials data which are associated with these roots, however there were too many.

```
#root with associated conditions recorded for each trial
condition3 = trial_nodupl.select("Id","Conditions").withColumn("Conditions",explode(split("Conditions",",")))
root3 = condition3.join(rootdf,rootdf["term"]==condition3["Conditions"], "left").filter(col("term").isNotNull())
root4=root3.groupBy("root").count()
groupConditions = root3.groupBy("root").agg(collect_set("conditions").alias("conditions")).withColumnRenamed("root","root_code")
root5 = root4.join(groupConditions, groupConditions["root_code"]==root4["root"], "left").select("root","count","conditions").orderBy("count", ascending=False)

root5.limit(5).display()
```

root	count	conditions
C04	143723	["Breast Neoplasms", "Ganglion Cysts", "Lymphangioma", "Laryngeal Neoplasms", "Follicular Cyst", "Sturge-Weber Syndrome", "Muscle Neoplasms", "Syringoma", "Pulmonary Blastoma", "Pharyngeal Neoplasms", "Mesothelioma", "Intestinal Neoplasms", "Granulosa Cell Tumor", "Squamous Cell Carcinoma of Head and Neck", "Myofibroma", "Lymphoma", "Wilms Tumor", "Cerebellar Neoplasms", "Leydig Cell Tumor", "Neuroblastoma", "Hypothalamic Neoplasms", "Hemangiopericytoma", "Radicular Cyst", "Mandibular Neoplasms", "Atypical Squamous Cells of the Cervix", "Diffuse Intrinsic Pontine Glioma", "Endocrine Gland Neoplasms", "WAGR Syndrome", "Adenoma", "Paraneoplastic Polyneuropathy", "Lymphangioleiomyomatosis", "Sex Cord-Gonadal Stromal Tumors", "Thymoma", "Pallister-Hall Syndrome", "Malignant Carcinoid Syndrome", "Kidney Neoplasms", "Adenocarcinoma of Lung", "Histiocytoma", "Anaplasia", "Nasopharyngeal Neoplasms", "Klatskin Tumor", "Neurilemmoma", "Neurofibromatosis 2", "Intraocular Lymphoma", "Synovial Cyst", "Chondroma", "Osteochondroma", "Retinal Neoplasms", "Popliteal Cyst", "Peripheral Nervous System Neoplasms", "Vipoma", "Eye Neoplasms", "Soft Tissue Neoplasms", "Brain Neoplasms", "Anti-N-Methyl-D-Aspartate Receptor Encephalitis", "Infratentorial Neoplasms", "Pheochromocytoma", "Palatal Neoplasms", "Precancerous Conditions", "Rhabdoid Tumor", "Thyroid Nodule", "Pancreatic Cyst", "Urticaria Pigmentosa", "Oropharyngeal Neoplasms", "Ganglioneuroblastoma", "Leiomyomatosis", "Uterine Cervical Dysplasia", "Growth Hormone-Secreting Pituitary Adenoma", "Nervous System Neoplasms", "Carcinoid Tumor", "Lymphocele", "Oligodendroglioma", "Medulloblastoma", "Adenolymphoma", "Hemangioma", "Precursor T-Cell Lymphoblastic Leukemia-Lymphoma", "Ear Neoplasms", "Bronchial Neoplasms", "Myofibromatosis", "Myasthenia Gravis", "Pinealoma", "Common Bile Duct Neoplasms", "Lymphangiosarcoma", "Neoplasms by Site", "Fibromatosis", "Adenocarcinoma in Situ", "Paraganglioma", "Multiple Pulmonary Nodules", "Fibroadenoma", "Sigmoid Neoplasms", "Eyelid Neoplasms", "Precursor B-Cell Lymphoblastic Leukemia-Lymphoma", "Brenner Tumor", "Carcinoid Heart Disease", "Lymphomatoid Granulomatosis", "Xeroderma Pigmentosum", "Parotid Neoplasms", "Glioblastoma", "Pancoast Syndrome", "Esophageal Squamous Cell Carcinoma", "Parovarian Cyst", "Prolactinoma", "Neoplastic Processes", "Tongue Neoplasms", "Bone Cysts", "Paranasal Sinus Neoplasms", "Prostatic Neoplasms", "Brain Stem Neoplasms", "Neurocytoma", "Leiomyoma", "Carcinoma", "Solitary Fibrous Tumors", "Proteus Syndrome", "Stomach Neoplasms", "Arachnoid

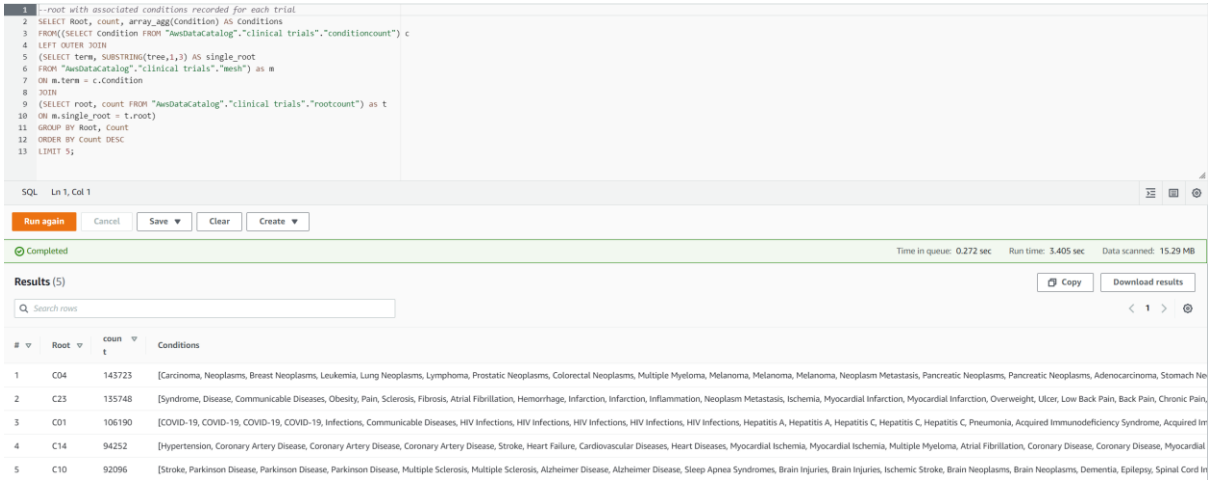
Showing all 5 rows.

```
#root with associated conditions recorded for each trial
conditionRoot = trialsRDD_nodupl.map(lambda x: (x[0],x[7].split(","))).flatMap(lambda x: [(x[0], v) for v in x[1]]).filter(lambda x: x[1]!= "").map(lambda x: (x[1],x[0])).join(rootRDD).map(lambda x: ((x[1]
[1],x[0]),1)).reduceByKey(lambda a,b: a+b).map(lambda x: (x[0][0],x[0][1])).reduceByKey(lambda a,b: str(a)+", "+str(b)).join(joinRDD2).sortBy(lambda x: x[1][1], False)
conditionRoot.take(5)
```

```
Out[23]: [('C04',
  ('Lymphoma, Bowens Disease, Lipoma, Nasopharyngeal Carcinoma, Thyroid Nodule, Melanoma, Peritoneal Neoplasms, ACTH-Secreting Pituitary Adenoma, Laryngeal Neoplasms, Mastocytosis, Hepatoblastoma, Carcinogenesi
s, Biliary Tract Neoplasms, Head and Neck Neoplasms, Adenoma, Esophageal Neoplasms, Oropharyngeal Neoplasms, Meningeal Neoplasms, Neuroma, Hemangiosarcoma, Pituitary Neoplasms, Carcinoid Tumor, Esophageal Squamo
us Cell Carcinoma, Pheochromocytoma, Bone Neoplasms, Carcinosarcoma, Li-Fraumeni Syndrome, Adenomyoepithelioma, Oligodendroglioma, Pancreatic Pseudocyst, Composite Lymphoma, Cervical Intraepithelial Neoplasia, P
olycythemia Vera, Rhabdomyosarcoma, Neurofibroma, Neurofibromatoses, Neurofibromatosis 1, Neurofibrosarcoma, Optic Nerve Glioma, Meningioma, Leukoplakia, Precursor T-Cell Lymphoblastic Leukemia-Lymphoma, Gliosar
coma, Diffuse Intrinsic Pontine Glioma, Ganglion Cysts, Synovial Cyst, Spinal Cord Neoplasms, Germinoma, Wilms Tumor, Malignant Carcinoid Syndrome, Kasabach-Merritt Syndrome, Hemangioblastoma, Angiofibroma, Chor
oid Plexus Neoplasms, Liposarcoma, Ovarian Cysts, Ganglioglioma, Adrenocortical Carcinoma, Follicular Cyst, Bone Cysts, Smoldering Multiple Myeloma, Prolactinoma, Abdominal Neoplasms, Angiomyolipoma, Growth Horm
one-Secreting Pituitary Adenoma, Hydatidiform Mole, Basal Cell Nevus Syndrome, Parotid Neoplasms, Retinal Neoplasms, Neurofibromatosis 2, Histiocytic Sarcoma, Anti-N-Methyl-D-Aspartate Receptor Encephalitis, Fib
roadenoma, Common Bile Duct Neoplasms, Angiolipoma, Adenofibroma, Tarlov Cysts, Pulmonary Blastoma, Endodermal Sinus Tumor, Retroperitoneal Neoplasms, Chondroma, Popliteal Cyst, Conjunctival Neoplasms, Choroid N
eoplasms, Lambert-Eaton Myasthenic Syndrome, Phyllodes Tumor, Opsoclonus-Myoclonus Syndrome, Cerebellar Neoplasms, Krukenberg Tumor, Urachal Cyst, Xeroderma Pigmentosum, Eccrine Porocarcinoma, Osteochondroma, Gl
omus Tumor, Atypical Squamous Cells of the Cervix, Perivascular Epithelioid Cell Neoplasms, Nevus of Ota, Erythroplasia, Submandibular Gland Neoplasms, Struma Ovarii, Neurocytoma, Ear Neoplasms, Linitis Plastic
a, Hemangioma, Carcinoma, Brain Neoplasms, Triple Negative Breast Neoplasms, Urinary Bladder Neoplasms, Multiple Myeloma, Thyroid Neoplasms, Prostatic Neoplasms, Nasopharyngeal Neoplasms, Carcinoma in Situ, Pros
tatic Intraepithelial Neoplasia, Polycystic Ovary Syndrome, Neuroendocrine Tumors, Rectal Neoplasms, Dermatofibrosarcoma, Hodgkin Disease, Preleukemia, Adrenocortical Adenoma, Leiomyoma, Paranasal Sinus Neoplasm
s, Proteus Syndrome, Adenomatous Polyposis Coli, Nevus, Adenocarcinoma of Lung, Myofibroma, Myasthenia Gravis, Plasmablastic Lymphoma, Waldenstrom Macroglobulinemia, Neuroblastoma, Anus Neoplasms, Vaginal Neopla
sms, Colonic Neoplasms, Gastrinoma, Somatostatinoma, Precursor B-Cell Lymphoblastic Leukemia-Lymphoma, Inflammatory Breast Neoplasms, Fibroma, Multiple Pulmonary Nodules, Osteosarcoma, Skull Base Neoplasms, Bile
Duct Neoplasms, Gastrointestinal Stromal Tumors, Vulvar Neoplasms, Intestinal Neoplasms, Pelvic Neoplasms, Pinealoma, Retinoblastoma, Breast Carcinoma In Situ, Ependymoma, Soft Tissue Neoplasms, Klatskin Tumor,
Adrenal Cortex Neoplasms, Craniopharyngioma, Parathyroid Neoplasms, Pseudomyxoma Peritonei, Ganglioneuroma, Hemangiopericytoma, Lymphomatoid Papulosis, Neoplasms by Site, Histiocytoma, Heart Neoplasms, Giant Cel
l Tumor of Tendon Sheath, Squamous Intraepithelial Lesions of the Cervix, Bronchial Neoplasms, Odontogenic Tumors, Central Nervous System Cysts, Sertoli-Leydig Cell Tumor, Osteoma, Choriocarcinoma, Hutchinsons M
elanotic Freckle, Maxillary Neoplasms, Granulosa Cell Tumor, Urticaria Pigmentosa, Multiple Endocrine Neoplasia Type 2a, Neoplasm Seeding, Pleural Neoplasms, Radicular Cyst, Denys-Drash Syndrome, Mastocytoma, Se
x Cord-Gonadal Stromal Tumors, Angiokeratoma, Parovarian Cyst, Lymphangiomyoma, Mucocele, Skull Neoplasms, Gardner Syndrome, Breast Neoplasms, Pancreatic Neoplasms, Colorectal Neoplasms, Neoplasm Metastasis, Liv
er Neoplasms, Fallopian Tube Neoplasms, Precursor Cell Lymphoblastic Leukemia-Lymphoma, Mycosis Fungoides, Sezary Syndrome, Mouth Neoplasms, Glioblastoma, Brain Stem Neoplasms, Hypopharyngeal Neoplasms, Precanc
```

```
--root with associated conditions recorded for each trial
SELECT Root, count, collect_set(Condition) AS Conditions
FROM((SELECT Condition FROM ConditionCount) c
LEFT OUTER JOIN
(SELECT term, SUBSTRING(tree,1,3) AS single_root
FROM mesh) as m
ON m.term == c.Condition
JOIN
(SELECT root, count FROM CountRoot) as t
ON m.single_root == t.root)
GROUP BY Root, Count
ORDER BY Count DESC
LIMIT 5;
```

Root	count	Conditions
C04	143723	["Follicular Cyst", "Ganglion Cysts", "Lymphangioma", "Laryngeal Neoplasms", "Breast Neoplasms", "Sturge-Weber Syndrome", "Muscle Neoplasms", "Syringoma", "Pulmonary Blastoma", "Pharyngeal Neoplasms", "Mesothelioma", "Intestinal Neoplasms", "Granulosa Cell Tumor", "Squamous Cell Carcinoma of Head and Neck", "Myofibroma", "Radicular Cyst", "Wilms Tumor", "Mandibular Neoplasms", "Lymphoma", "Neuroblastoma", "Leydig Cell Tumor", "Hypothalamic Neoplasms", "Hemangiopericytoma", "Cerebellar Neoplasms", "Atypical Squamous Cells of the Cervix", "Diffuse Intrinsic Pontine Glioma", "Endocrine Gland Neoplasms", "WAGR Syndrome", "Adenoma", "Paraneoplastic Polyneuropathy", "Lymphangioleiomyomatosis", "Sex Cord-Gonadal Stromal Tumors", "Thymoma", "Pallister-Hall Syndrome", "Malignant Carcinoid Syndrome", "Kidney Neoplasms", "Adenocarcinoma of Lung", "Histiocytoma", "Anaplasia", "Nasopharyngeal Neoplasms", "Klatskin Tumor", "Neurofibromatosis 2", "Synovial Cyst", "Intraocular Lymphoma", "Osteochondroma", "Retinal Neoplasms", "Popliteal Cyst", "Peripheral Nervous System Neoplasms", "Neurilemmoma", "Chondroma", "Vipoma", "Eye Neoplasms", "Anti-N-Methyl-D-Aspartate Receptor Encephalitis", "Soft Tissue Neoplasms", "Brain Neoplasms", "Infratentorial Neoplasms", "Pheochromocytoma", "Palatal Neoplasms", "Thyroid Nodule", "Rhabdoid Tumor", "Precancerous Conditions", "Pancreatic Cyst", "Urticaria Pigmentosa", "Oropharyngeal Neoplasms", "Ganglioneuroblastoma", "Leiomyomatosis", "Uterine Cervical Dysplasia", "Growth Hormone-Secreting Pituitary Adenoma", "Nervous System Neoplasms", "Carcinoid Tumor", "Lymphocele", "Oligodendroglioma", "Medulloblastoma", "Ear Neoplasms", "Precursor T-Cell Lymphoblastic Leukemia-Lymphoma", "Hemangioma", "Adenolymphoma", "Bronchial Neoplasms", "Myofibromatosis", "Myasthenia Gravis", "Pinealoma", "Common Bile Duct Neoplasms", "Neoplasms by Site", "Lymphangiosarcoma", "Fibromatosis", "Adenocarcinoma in Situ", "Paraganglioma", "Multiple Pulmonary Nodules", "Fibroadenoma", "Sigmoid Neoplasms", "Eyelid Neoplasms", "Precursor B-Cell Lymphoblastic Leukemia-Lymphoma", "Brenner Tumor", "Carcinoid Heart Disease", "Lymphomatoid Granulomatosis", "Xeroderma Pigmentosum", "Parotid Neoplasms", "Glioblastoma", "Pancoast Syndrome", "Esophageal Squamous Cell Carcinoma", "Parovarian Cyst", "Prolactinoma", "Neoplastic Processes", "Tongue Neoplasms", "Bone Cysts", "Paranasal Sinus Neoplasms", "Brain Stem Neoplasms", "Prostatic



Instead, roots were joined to corresponding groups, as in the conditions implementation. Cancers were the most popular root studied, followed by pathological conditions, infections, cardiovascular diseases, and finally nervous system diseases. This follows a similar trend to the top conditions studied, with cancers the most studied overall.

```
#root with associated groups recorded for each trial
rootGroup2 = root2.join(mesh,mesh["tree"]==root2["root"], "left").select("root","count","term").groupBy("root","count").agg(collect_set("term").alias("term")).orderBy("count", ascending=False)
rootGroup2.limit(5).display()
```

	root	count	term
1	C04	143723	["Neoplasms", "Benign Neoplasms", "Cancer"]
2	C23	135748	["Pathological Conditions, Signs and Symptoms"]
3	C01	106190	["Infections"]
4	C14	94252	["Cardiovascular Diseases"]
5	C10	92096	["Nervous System Diseases"]

Showing all 5 rows.

```
#root with associated groups recorded for each trial
rootGroup = meshRDD.map(lambda x: (x[1],x[0])).join(joinRDD2).map(lambda x: (x[0],x[1][0])).reduceByKey(lambda a,b: str(a)+", "+str(b))
rootGroup.join(joinRDD2).map(lambda x: (x[0],(x[1][1],x[1][0]))).sortBy(lambda x: x[1][0], False).take(5)
```

```
Out[24]: [('C04', (143723, 'Neoplasms, Cancer, Benign Neoplasms')),
 ('C23', (135748, 'Pathological Conditions Signs and Symptoms')),
 ('C01', (106190, 'Infections')),
 ('C14', (94252, 'Cardiovascular Diseases')),
 ('C10', (92096, 'Nervous System Diseases'))]
```

```
--- root with corresponding group
SELECT root, count, collect_set(term) AS groups
FROM((SELECT root, count FROM CountRoot) c
LEFT OUTER JOIN
(SELECT term, tree
FROM mesh) as m
ON m.tree == c.root)
GROUP BY root, count
ORDER BY count DESC
LIMIT 5;
```

	root	count	groups
1	C04	143723	["Neoplasms", "Benign Neoplasms", "Cancer"]
2	C23	135748	["Pathological Conditions, Signs and Symptoms"]
3	C01	106190	["Infections"]
4	C14	94252	["Cardiovascular Diseases"]
5	C10	92096	["Nervous System Diseases"]

Showing all 5 rows.

```sql
--root frequency with groups
SELECT root, count, array_agg(term) AS groups
FROM((SELECT root, count FROM "AwsDataCatalog"."clinical trials"."rootcount") c
LEFT OUTER JOIN
(SELECT term, tree
FROM "AwsDataCatalog"."clinical trials"."mesh") as m
ON m.tree = c.root)
GROUP BY root, count
ORDER BY count DESC
LIMIT 5;
```

SQL Ln 1, Col 1

Run again | Cancel | Save | Clear | Create

Completed Time in queue: 0.168 sec Run time: 2.029 sec

Results (5) Copy

Search rows

#	root	count	groups
1	C04	143723	[Benign Neoplasms, Cancer, Neoplasms]
2	C23	135748	[Pathological Conditions, Signs and Symptoms]
3	C01	106190	[Infections]
4	C14	94252	[Cardiovascular Diseases]
5	C10	92096	[Nervous System Diseases]

Similar to the top conditions studied, there may be a different order of frequency when only considering trials with the status ‘Completed’. This also found Cancers (C04) are less frequently found in completed trials compared to the full set of trials and so could take longer to complete these trials.

```
#top 5 roots for complete trials only
conditionComplete=trial_nodupl.filter(col("Status")=="Completed").select("Id","Conditions").withColumn("Conditions",explode(split("Conditions",",")))
root6 = conditionComplete.join(rootdf,rootdf["term"]==conditionComplete["Conditions"], "left").filter(col("term").isNotNull()).groupBy("root").count().orderBy("count", ascending=False)
rootGroup3 = root6.join(mesh,mesh["tree"]==root6["root"], "left").select("root","count","term").groupBy("root","count").agg(collect_set("term").alias("term")).orderBy("count", ascending=False)
rootGroup3.limit(5).display()
```

	root	count	term
1	C23	70517	["Pathological Conditions, Signs and Symptoms"]
2	C04	59623	["Neoplasms", "Benign Neoplasms", "Cancer"]
3	C01	57888	["Infections"]
4	C14	47555	["Cardiovascular Diseases"]
5	C10	45319	["Nervous System Diseases"]

Showing all 5 rows.

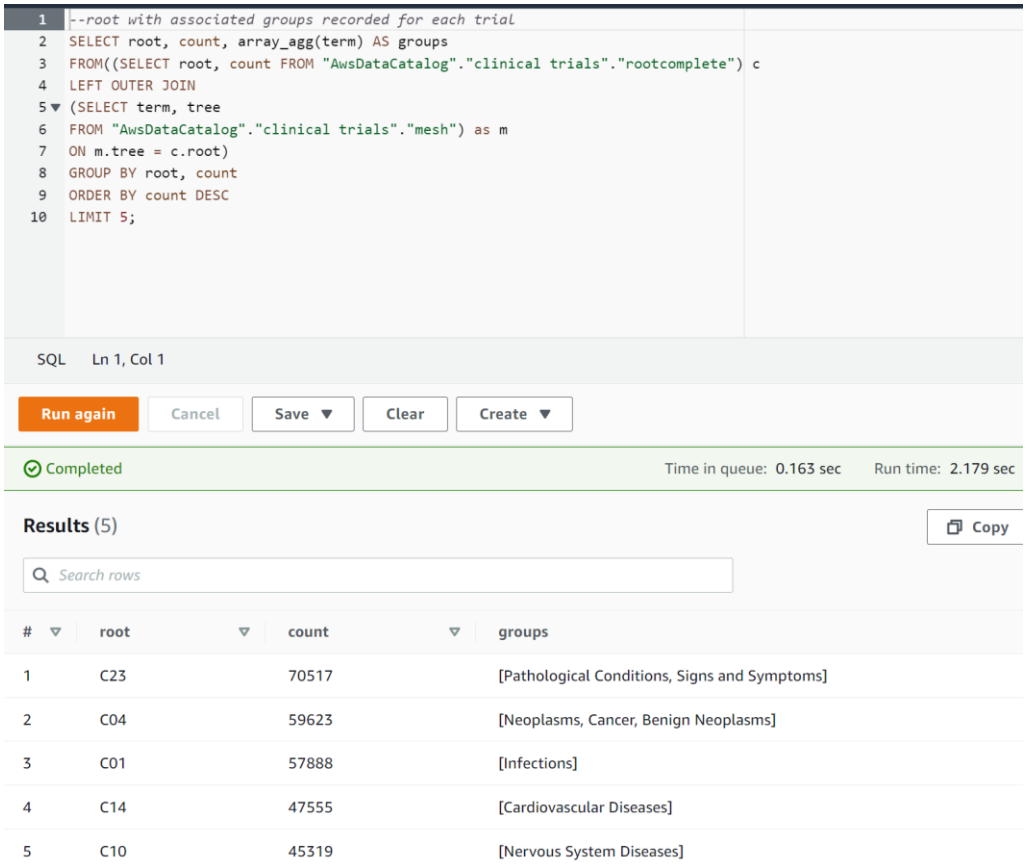
```
#roots for completed trials only
conditionRDD2 = trialsRDD_nodupl.filter(lambda x: x[2]=="Completed").map(lambda x: x[7].split(",")).flatMap(lambda x: x).filter(lambda x: x!= "").map(lambda x: (x,1))
joinRDD2 = rootRDD.join(conditionRDD2).map(lambda x: (x[1][0],x[1][1])).filter(lambda x: x is not None).reduceByKey(lambda a,b: a+b).sortBy(lambda x: x[1], False)
rootGroup = meshRDD.map(lambda x: (x[1],x[0])).join(joinRDD2).map(lambda x: (x[0],x[1][0])).reduceByKey(lambda a,b: str(a)+", "+str(b))
rootGroup.join(joinRDD2).map(lambda x: (x[0],(x[1][1],x[1][0]))).sortBy(lambda x: x[1][0], False).take(5)

Out[25]: [('C23', (70517, 'Pathological Conditions Signs and Symptoms')),
 ('C04', (59623, 'Neoplasms, Cancer, Benign Neoplasms')),
 ('C01', (57888, 'Infections')),
 ('C14', (47555, 'Cardiovascular Diseases')),
 ('C10', (45319, 'Nervous System Diseases'))]
```

```
--- top 5 roots for completed trials only
CREATE OR REPLACE VIEW RootComplete AS
SELECT root, Count(Root) As count
FROM((SELECT Condition
FROM clinicaltrial_nodupl
LATERAL VIEW explode(split(Conditions, ',')) Conditions as Condition
WHERE Status=='Completed') c
LEFT OUTER JOIN
(SELECT term AS Condition, SUBSTRING(tree,1,3) AS Root
FROM mesh) as m
ON m.Condition == c.Condition)
GROUP BY Root;
--root with associated groups recorded for each trial
SELECT root, count, collect_set(term) AS groups
FROM((SELECT root, count FROM RootComplete) c
LEFT OUTER JOIN
(SELECT term, tree
FROM mesh) as m
ON m.tree == c.root)
GROUP BY root, count
ORDER BY count DESC
LIMIT 5;
```

	root	count	groups
1	C23	70517	["Pathological Conditions, Signs and Symptoms"]
2	C04	59623	["Neoplasms", "Benign Neoplasms", "Cancer"]
3	C01	57888	["Infections"]
4	C14	47555	["Cardiovascular Diseases"]
5	C10	45319	["Nervous System Diseases"]

Showing all 5 rows.

```
--create root complete view
CREATE OR REPLACE VIEW RootComplete AS
SELECT root, Count(Root) As count
FROM((SELECT Condition
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl" n
CROSS JOIN UNNEST(split(n.Conditions, ',')) AS t(Condition)
WHERE Status='Completed') c
LEFT OUTER JOIN
(SELECT term AS Condition, SUBSTRING(tree,1,3) AS Root
FROM "AwsDataCatalog"."clinical trials"."mesh") as m
ON m.Condition = c.Condition)
GROUP BY Root;
```



```
--root with associated groups recorded for each trial
SELECT root, count, array_agg(term) AS groups
FROM((SELECT root, count FROM "AwsDataCatalog"."clinical trials"."rootcomplete") c
LEFT OUTER JOIN
(SELECT term, tree
FROM "AwsDataCatalog"."clinical trials"."mesh") as m
ON m.tree = c.root)
GROUP BY root, count
ORDER BY count DESC
LIMIT 5;
```

SQL Ln 1, Col 1

Run again | Cancel | Save | Clear | Create

Completed Time in queue: 0.163 sec Run time: 2.179 sec

Results (5) Copy

Search rows

#	root	count	groups
1	C23	70517	[Pathological Conditions, Signs and Symptoms]
2	C04	59623	[Neoplasms, Cancer, Benign Neoplasms]
3	C01	57888	[Infections]
4	C14	47555	[Cardiovascular Diseases]
5	C10	45319	[Nervous System Diseases]

The second most studied condition found previously was Diabetes Mellititus, belonging to the group containing Nutritional and Metabolic Diseases and Endocrine System Diseases, which did not feature in the most studied roots. Individual conditions may be studied more frequently than others within their group or there are less conditions within their group, so their root is represented less compared to those which contain many conditions or more equally represented conditions. It may be more informative to consider individual conditions rather than groups unlike previously suggested.

5. Ten most common non-pharmaceutical sponsors

Initial implementation involved joining the sponsor column in the clinical trial data with the parent company column in the pharma data using left join. Pharmaceutical companies in the parent column which were not null were removed, leaving sponsors which did not match parent companies. These were counted as previously done before and the top ten selected.

```
#parent companies only
pharm_only = pharma.select(pharma.Parent_Company).distinct()
```

```
#initial implementation
sponsor = trial.join(pharm_only, trial.Sponsor == pharm_only.Parent_Company, "left").filter(col("Parent_Company").isNull()).groupBy("Sponsor").count().orderBy("count", ascending=False)
sponsor.limit(10).display()
```

	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
8	Massachusetts General Hospital	1971
9	Cairo University	1928
10	Hoffmann-La Roche	1828

Showing all 10 rows.

```
#initial implementation
sponsorRDD = trialsRDD.map(lambda x: (x[1],1))
nonpharma=sponsorRDD.leftOuterJoin(parentCompany).filter(lambda x: x[1][1] !=1).map(lambda x: (x[0],x[1][0])).reduceByKey(lambda a,b: a+b).sortBy(lambda x: x[1], False)
nonpharma.take(10)

Out[26]: [('GlaxoSmithKline', 3378),
 ('National Cancer Institute (NCI)', 3218),
 ('AstraZeneca', 2691),
 ('Pfizer', 2645),
 ('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)]
```

```
--initial implementation
SELECT Sponsor AS Sponsor, count(sponsor) AS Count
FROM clinicaltrial
LEFT ANTI JOIN pharma ON Sponsor == Parent_Company
GROUP BY sponsor
ORDER BY Count DESC
LIMIT 10;
```

	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
8	Massachusetts General Hospital	1971
9	Cairo University	1928
10	Hoffmann-La Roche	1828

Showing all 10 rows.

```
--frequency of non-pharmaceutical sponsors
--initial implementation
SELECT c.Sponsor, COUNT(c.Sponsor) AS count
FROM "AwsDataCatalog"."clinical trials"."clinicaltrials" c
WHERE NOT EXISTS (
SELECT * FROM "AwsDataCatalog"."clinical trials"."pharma"
WHERE Parent_Company= c.Sponsor)
GROUP BY Sponsor
ORDER BY count DESC
LIMIT 10;
```

Results (10)

Copy

Search rows

#	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
8	Massachusetts General Hospital	1971
9	Cairo University	1928
10	Hoffmann-La Roche	1828

The same implementation was also repeated using the data without duplicate trials, but this did not alter the sponsor order.

```
#without duplicates
sponsor2 = trial_nodupl.join(pharm_only, trial_nodupl.Sponsor == pharm_only.Parent_Company, "left").filter(col("Parent_Company").isNull()).groupBy("Sponsor").count().orderBy("count", ascending=False)
sponsor2.limit(10).display()
```

	Sponsor	count
1	National Cancer Institute (NCI)	3212
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2356
4	Mayo Clinic	2298
5	Merck Sharp & Dohme Corp.	2230
6	Assiut University	2146
7	Novartis Pharmaceuticals	2080
8	Massachusetts General Hospital	1966
9	Cairo University	1895
10	Hoffmann-La Roche	1822

```
#removing companies containing Pharma and Corp
nonpharma2=sponsorRDD2.leftOuterJoin(parentCompany).filter(lambda x: x[1][1] !=1).map(lambda x: (x[0],x[1][0])).filter(lambda x: "Pharma" not in x[0]).filter(lambda x: "Corp" not in x[0]).reduceByKey(lambda a,b: a+b).sortBy(lambda x: x[1], False)
nonpharma2.take(10)

Out[28]: [('GlaxoSmithKline', 3343),
 ('National Cancer Institute (NCI)', 3212),
 ('AstraZeneca', 2660),
 ('Pfizer', 2628),
 ('M.D. Anderson Cancer Center', 2414),
 ('Assistance Publique - Hôpitaux de Paris', 2356),
 ('Mayo Clinic', 2298),
 ('Assiut University', 2146),
 ('Massachusetts General Hospital', 1966),
 ('Boehringer Ingelheim', 1906)]
```

```
--without duplicates
SELECT Sponsor AS Sponsor, count(sponsor) AS Count
FROM clinicaltrial_nodupl
LEFT ANTI JOIN pharma ON Sponsor == Parent_Company
GROUP BY sponsor
ORDER BY Count DESC
LIMIT 10;
```

	Sponsor	Count
1	National Cancer Institute (NCI)	3212
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2356
4	Mayo Clinic	2298
5	Merck Sharp & Dohme Corp.	2230
6	Assiut University	2146
7	Novartis Pharmaceuticals	2080
8	Massachusetts General Hospital	1966
9	Cairo University	1895
10	Hoffmann-La Roche	1822

Showing all 10 rows.

```sql
--frequency of non-pharmaceutical sponsors
--without duplicates
SELECT c.Sponsor, COUNT(c.Sponsor) AS count
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl" c
WHERE NOT EXISTS (
SELECT * FROM "AwsDataCatalog"."clinical trials"."pharma"
WHERE Parent_Company= c.Sponsor)
GROUP BY Sponsor
ORDER BY count DESC
LIMIT 10;
```

Results (10) Copy

Search rows

#	Sponsor	count
1	National Cancer Institute (NCI)	3212
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2356
4	Mayo Clinic	2298
5	Merck Sharp & Dohme Corp.	2230
6	Assiut University	2146
7	Novartis Pharmaceuticals	2080
8	Massachusetts General Hospital	1966
9	Cairo University	1895
10	Hoffmann-La Roche	1822

This assumes parent company column contains all possible pharmaceutical companies. However, this is not the case with ‘Novartis Pharmaceuticals’ and ‘Merck Sharp & Dohme Corp’ being present in the top ten. These can be removed by removing rows which contain ‘Pharma’ and ‘Corp’.

```
#removing companies containing pharma and corp
sponsor3=trial_nodupl.join(pharm_only, trial_nodupl.Sponsor == pharm_only.Parent_Company,
"left").filter(col("Parent_Company").isNull()).withColumn("Sponsor_lower",lower("Sponsor")).filter(~col("Sponsor_lower").contains("pharma"))\
.filter(~col("Sponsor_lower").contains("corp")).groupBy("Sponsor").count().orderBy("count", ascending=False)
sponsor3.limit(10).display()
```

	Sponsor	count
1	National Cancer Institute (NCI)	3212
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2356
4	Mayo Clinic	2298
5	Assiut University	2146
6	Massachusetts General Hospital	1966
7	Cairo University	1895
8	Hoffmann-La Roche	1822
9	National Taiwan University Hospital	1804
10	Eli Lilly and Company	1633

```
#removing companies containing Pharma and Corp
nonpharma2=sponsorRDD2.leftOuterJoin(parentCompany).filter(lambda x: x[1][1] !=1).map(lambda x: (x[0],x[1][0])).filter(lambda x: "Pharma" not in x[0]).filter(lambda x: "Corp" not in x[0]).reduceByKey(lambda a,b:
a+b).sortBy(lambda x: x[1], False)
nonpharma2.take(10)

Out[28]: [('GlaxoSmithKline', 3343),
 ('National Cancer Institute (NCI)', 3212),
 ('AstraZeneca', 2660),
 ('Pfizer', 2628),
 ('M.D. Anderson Cancer Center', 2414),
 ('Assistance Publique - Hôpitaux de Paris', 2356),
 ('Mayo Clinic', 2298),
 ('Assiut University', 2146),
 ('Massachusetts General Hospital', 1966),
 ('Boehringer Ingelheim', 1906)]
```

```sql
--removing companies containing pharma and corp
SELECT Sponsor AS Sponsor, count(sponsor) AS Count
FROM clinicaltrial_nodupl
LEFT ANTI JOIN pharma ON Sponsor == Parent_Company
WHERE Sponsor NOT LIKE '%pharma%' AND Sponsor NOT LIKE '%Pharma%'
AND Sponsor NOT LIKE '%corp%' AND Sponsor NOT LIKE '%Corp%'
GROUP BY sponsor
ORDER BY Count DESC
LIMIT 10;
```

	Sponsor	Count
1	National Cancer Institute (NCI)	3212
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2356
4	Mayo Clinic	2298
5	Assiut University	2146
6	Massachusetts General Hospital	1966
7	Cairo University	1895
8	Hoffmann-La Roche	1822
9	National Taiwan University Hospital	1804
10	Eli Lilly and Company	1633

```sql
--frequency of non-pharmaceutical sponsors
--without 'pharma' and 'corp'
SELECT c.Sponsor, COUNT(c.Sponsor) AS count
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl" c
WHERE NOT EXISTS (
SELECT * FROM "AwsDataCatalog"."clinical trials"."pharma"
WHERE Parent_Company= c.Sponsor) AND Sponsor NOT LIKE '%pharma%' AND Sponsor NOT LIKE '%Pharma%' AND Sponsor NOT LIKE '%corp%' AND Sponsor NOT LIKE '%Corp%'
GROUP BY Sponsor
ORDER BY count DESC
LIMIT 10;
```

Results (10)

Copy

Search rows

#	Sponsor	count
1	National Cancer Institute (NCI)	3212
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2356
4	Mayo Clinic	2298
5	Assiut University	2146
6	Massachusetts General Hospital	1966
7	Cairo University	1895
8	Hoffmann-La Roche	1822
9	National Taiwan University Hospital	1804
10	Eli Lilly and Company	1633

This improves the current top ten, however does not necessarily remove all pharmaceutical companies from the full list in general. Using the Company column in the pharma dataset seems to yield better results.

```
#companies only
pharm_only2 = pharma.select(pharma.Company).distinct()
#using company column
sponsor4 = trial_nodupl.join(pharm_only2, trial_nodupl.Sponsor == pharm_only2.Company, "left").filter(col("Company").isNull()).groupBy("Sponsor").count().orderBy("count", ascending=False)
sponsor4.limit(10).display()
```

	Sponsor	count
1	National Cancer Institute (NCI)	3212
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2356
4	Mayo Clinic	2298
5	Assiut University	2146
6	Massachusetts General Hospital	1966
7	Cairo University	1895
8	Hoffmann-La Roche	1822
9	National Taiwan University Hospital	1804
10	Memorial Sloan Kettering Cancer Center	1601

```
#companies only
companyRDD=pharmaRDD.map(lambda x: x[0]).map(lambda x: x.replace('"', "")).map(lambda x: (x,1))
nonpharma4=sponsorRDD2.leftOuterJoin(companyRDD).filter(lambda x: x[1][1] !=1).map(lambda x: (x[0],x[1][0])).reduceByKey(lambda a,b: a+b).sortBy(lambda x: x[1], False)
nonpharma4.take(10)
```

```
Out[29]: [('National Cancer Institute (NCI)', 3212),
 ('M.D. Anderson Cancer Center', 2414),
 ('Assistance Publique - Hôpitaux de Paris', 2356),
 ('Mayo Clinic', 2298),
 ('Assiut University', 2146),
 ('Massachusetts General Hospital', 1966),
 ('Cairo University', 1895),
 ('National Taiwan University Hospital', 1804),
 ('Memorial Sloan Kettering Cancer Center', 1601),
 ('University of California, San Francisco', 1597)]
```

```
--companies only
SELECT Sponsor AS Sponsor, count(sponsor) AS Count
FROM clinicaltrial_nodupl
LEFT ANTI JOIN pharma ON Sponsor == Company
GROUP BY sponsor
ORDER BY Count DESC
LIMIT 10;
```

	Sponsor	Count
1	National Cancer Institute (NCI)	3212
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2356
4	Mayo Clinic	2298
5	Assiut University	2146
6	Massachusetts General Hospital	1966
7	Cairo University	1895
8	Hoffmann-La Roche	1822
9	National Taiwan University Hospital	1804
10	Memorial Sloan Kettering Cancer Center	1601

```
1  --frequency of non-pharmaceutical sponsors
2  --using company
3  SELECT c.Sponsor, COUNT(c.Sponsor) AS count
4  FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl" c
5  WHERE NOT EXISTS (
6  SELECT * FROM "AwsDataCatalog"."clinical trials"."pharma"
7  WHERE Company= c.Sponsor)
8  GROUP BY Sponsor
9  ORDER BY count DESC
10 LIMIT 10;
```

Results (10)

Copy Downl

Search rows

#	Sponsor	count
1	National Cancer Institute (NCI)	3212
2	M.D. Anderson Cancer Center	2414
3	Assistance Publique - Hôpitaux de Paris	2356
4	Mayo Clinic	2298
5	Assiut University	2146
6	Massachusetts General Hospital	1966
7	Cairo University	1895
8	Hoffmann-La Roche	1822
9	National Taiwan University Hospital	1804
10	Memorial Sloan Kettering Cancer Center	1601

The most popular non-pharmaceutical sponsor is the National Cancer Institute, followed by the M.D. Anderson Cancer Center and then the Memorial Sloan Kettering Cancer Centre is tenth. This makes sense with cancers being the most popular conditions studied in the trials data. Other popular non-pharmaceutical sponsors appear to be mainly hospitals and universities.

6. Number of completed studies each month in 2021

Each implementation involved filtering status ‘Completed’ then filtering those which contained the inputted year (2021) in the ‘Completed’ column and removing the year value so only the month remained, which was renamed to ‘Month Completed’. These were then counted as previously described. To sort by month in dataframe and HiveQL, the months were sorted by converting them to a timestamp using Unix-timestamp and then converting back into the month. For RDD, a loop which would give the inputted month its corresponding number was created and applied, allowing the month key to be sorted according to these number values. In Athena, the function ‘date_parse’ was used.

```
spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
#initial implementation
completed = trial.filter(col("Status")=="Completed").filter(col("Completion").contains(year)).select(regexp_replace("Completion", year, "").alias("Month Completed")).groupBy("Month
Completed").count().sort(from_unixtime(unix_timestamp(col("Month Completed"),"MMM"),"MM"))
completed.display()
```

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

```
#initial implementation
order_by_month = {i:e for e,i in enumerate(calendar.month_abbr[1:],1)}

completedRDD = trialsRDD.filter(lambda x: x[2]=="Completed").filter(lambda x: year in x[4]).map(lambda x: x[4].split()).map(lambda x: (x[0],1)).reduceByKey(lambda a,b: a+b).sortBy(keyfunc=lambda x:
order_by_month.get(x[0]))
completedRDD.collect()

Out[30]: [('Jan', 1131),
 ('Feb', 934),
 ('Mar', 1227),
 ('Apr', 967),
 ('May', 984),
 ('Jun', 1094),
 ('Jul', 819),
 ('Aug', 700),
 ('Sep', 528),
 ('Oct', 187)]
```

```
--initial implementations
SELECT TRIM(' ${hivevar:year}' FROM Completion) AS Month, Count(Completion) AS Count
FROM clinicaltrial
WHERE Status = 'Completed' AND Completion LIKE '%${hivevar:year}'
GROUP BY Completion
ORDER BY from_unixtime(unix_timestamp(concat(Month,'-2000'),'MMM-yyyy'),'MM-yyyy');
```

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

```
1  --frequency of completed trials per month
2  --initial implementations
3  SELECT SUBSTRING(Completion,1,3) AS Month, Count(Completion) AS Count
4  FROM "AwsDataCatalog"."clinical trials"."clinicaltrials"
5  WHERE Status = 'Completed' AND Completion LIKE '%2021%'
6  GROUP BY Completion
7  ORDER BY date_parse(Month, '%b');
```

Results (10)

Search rows

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

This was repeated using the dataset which did not contain duplicate data.

```
#without duplicates
completed2 = trial_nodupl.filter(col("Status")=="Completed").filter(col("Completion").contains(year)).select(regexp_replace("Completion", year, "").alias("Month Completed")).groupBy("Month Completed").count().sort(from_unixtime(unix_timestamp(col("Month Completed"),"MMM"),"MM"))
completed2.display()
```

	Month Completed	count
1	Jan	1127
2	Feb	933
3	Mar	1222
4	Apr	962
5	May	984
6	Jun	1093
7	Jul	814
8	Aug	698
9	Sep	526
10	Oct	186

Showing all 10 rows.

```
#without duplicates
completedRDD2 = trialsRDD_nodupl.filter(lambda x: x[2]=="Completed").filter(lambda x: year in x[4]).map(lambda x: x[4].split()).map(lambda x: (x[0],1)).reduceByKey(lambda a,b: a+b).sortBy(keyfunc=lambda x: order_by_month.get(x[0]))
completedRDD.collect()
```

```
Out[31]: [('Jan', 1131),
 ('Feb', 934),
 ('Mar', 1227),
 ('Apr', 967),
 ('May', 984),
 ('Jun', 1094),
 ('Jul', 819),
 ('Aug', 700),
 ('Sep', 528),
 ('Oct', 187)]
```

```
--without duplications
SELECT TRIM(' ${hivevar:year}' FROM Completion) AS Month, Count(Completion) AS Count
FROM clinicaltrial_nodupl
WHERE Status = 'Completed' AND Completion LIKE '%${hivevar:year}'
GROUP BY Completion
ORDER BY from_unixtime(unix_timestamp(concat(Month,'-2000'),'MMM-yyyy'),'MM-yyyy');
```

	Month	Count
1	Jan	1127
2	Feb	933
3	Mar	1222
4	Apr	962
5	May	984
6	Jun	1093
7	Jul	814
8	Aug	698
9	Sep	526
10	Oct	186

Showing all 10 rows.

```
--frequency of completed trials per month
--without duplicates
SELECT SUBSTRING(Completion,1,3) AS Month, Count(Completion) AS Count
FROM "AwsDataCatalog"."clinical trials"."clinicaltrial_nodupl"
WHERE Status = 'Completed' AND Completion LIKE '%2021%'
GROUP BY Completion
ORDER BY date_parse(Month, '%b');
```

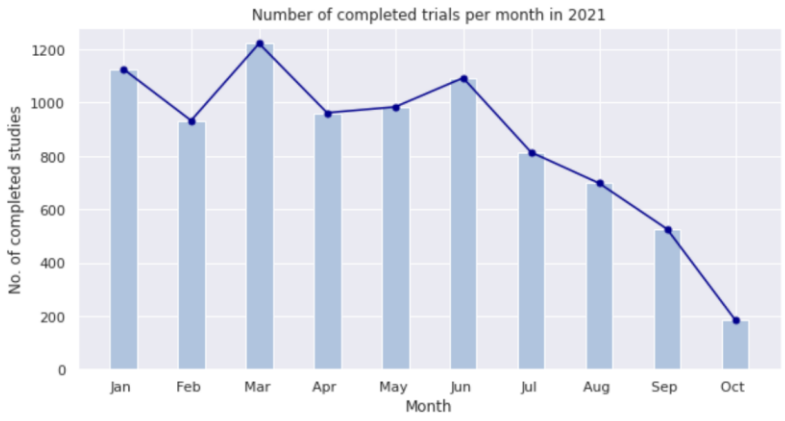
Results (10)

Search rows

#	Month	Count
1	Jan	1127
2	Feb	933
3	Mar	1222
4	Apr	962
5	May	984
6	Jun	1093
7	Jul	814
8	Aug	698
9	Sep	526
10	Oct	186

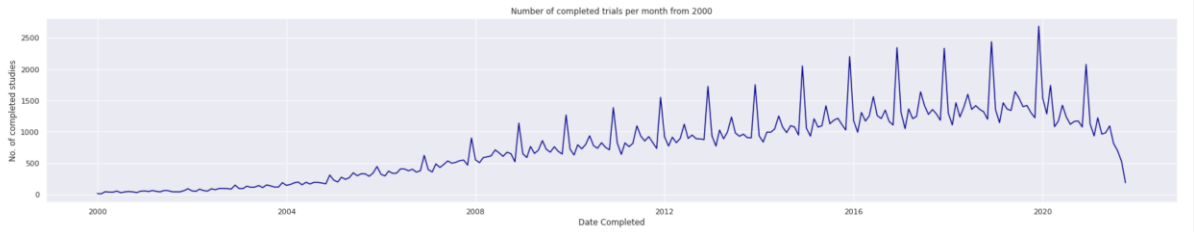
The bar graph was produced using Python packages Matplotlib and Seaborn. A line connecting the points of the bars was included to allow better visualisation of the overall trend. There is an overall decline in studies completed throughout 2021. The final month is October, which could be because the dataset is not up to date to include the final months or there may not have been any trials completed in November or December. As it is not known which is the case, these months not included in the graph as to not mislead the user into thinking no trials were completed in these months.

```
#visualisation using Matplotlib
completedPanda=completed2.toPandas()
fig = plt.figure(figsize = (10, 5))

plt.bar(completedPanda["Month Completed"], completedPanda["count"], color ="#B0C4DE",width = 0.4)
plt.plot(completedPanda["Month Completed"], completedPanda["count"], marker=".", ms=10, color="#00008B")

plt.xlabel("Month")
plt.ylabel("No. of completed studies")
plt.title("Number of completed trials per month in "+year)
plt.show()
```



The pattern seen in 2021 could occur every year or be abnormal. Looking at the number of completed trials per month from 2000 onwards, there is a decline from 2020 onwards and change in pattern so is likely abnormal.

```
completedFull=trial_nodupl.filter(col("Status")=="Completed").filter(col("Completion").contains("20")).filter(~col("Completion").contains("2031")).groupBy("Completion").count().withColumn("Completion",
to_timestamp(unix_timestamp(col("Completion"),"MMM yyyy"))).orderBy("Completion").toPandas()

fig = plt.figure(figsize = (30, 5))

plt.plot(completedFull["Completion"], completedFull["count"], color="#00008B")

plt.xlabel("Date Completed")
plt.ylabel("No. of completed studies")
plt.title("Number of completed trials per month from 2000")
plt.show()
```



Further analysis

For further analysis I was interested in whether the different study types took different lengths of time to complete. The method used in data exploration to produce the years taken for each trial to complete was applied.

```
#Years taken to complete studies
spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
#create timestamps for each date column
trialDF=trial_nodupl.withColumn("Start-timestamp", (unix_timestamp(trial_nodupl["Start"],"MMM yyyy"))).withColumn("Completion_timestamp", (unix_timestamp(trial_nodupl["Completion"],"MMM
yyyy"))).withColumn("Submission_timestamp", (unix_timestamp(trial_nodupl["Submission"],"MMM yyyy")))
#filter for completed trials and create years column to work out difference in years between start and completion timestamp
timeDF=trialDF.filter(trialDF.Status=="Completed").filter(trialDF["Completion"].isNotNull()).withColumn("Years", (months_between(col("Completion_timestamp").cast(TimestampType()),col("Start-
timestamp").cast(TimestampType()))/12).cast(DoubleType())).na.fill(value=(1/12),subset=["Years"])
```

When plotting these results as boxplots and violin plots using matplotlib and seaborn, it did not appear there was much difference between the average time taken for each trial type to be complete. There was also large variation in the time taken to complete the trials. I then decided to see whether there was any change over time in how quickly trials have been completed.

```
#how long each type took to complete on average
timeDF.groupBy("Type").agg({"Years": "avg"}).display()
```

	Type	avg(Years)
1	Observational [Patient Registry]	2.659744515783841
2	Interventional	2.510730067183325
3	Observational	2.995081470480279

Showing all 3 rows.

```
#minimum years to complete for type
timeDF.groupBy("Type").agg({"Years": "min"}).display()
```

	Type	min(Years)
1	Observational [Patient Registry]	0
2	Interventional	0
3	Observational	0

Showing all 3 rows.

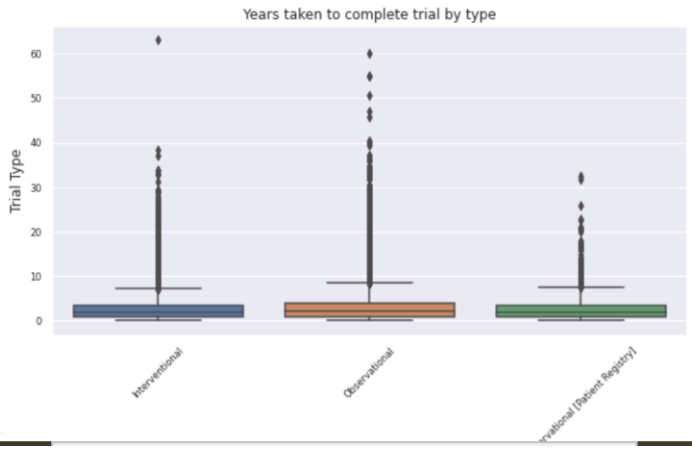
```
#maximum years to complete each type
timeDF.groupBy("Type").agg({"Years": "max"}).display()
```

	Type	max(Years)
1	Observational [Patient Registry]	32.416666666666664
2	Interventional	63.083333333333336
3	Observational	60.166666666666664

Showing all 3 rows.

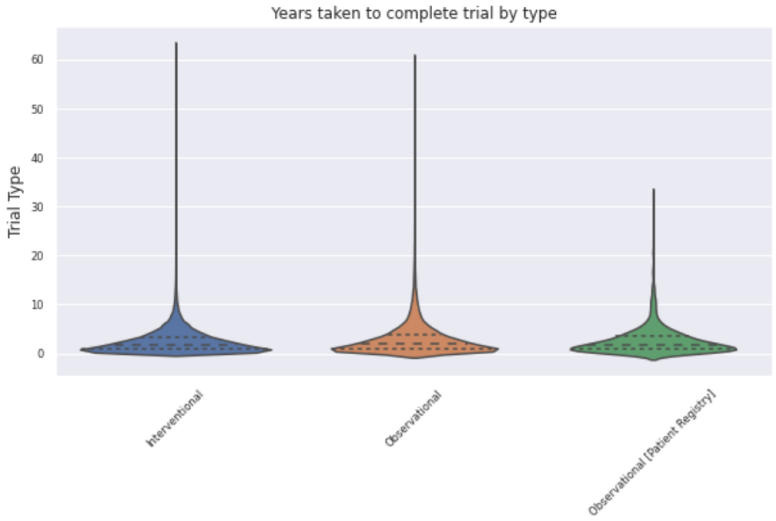
```
#boxplots for each type
timePandas = timeDF.toPandas()

plt.figure(figsize = (10, 5))

fig = sns.boxplot(timePandas["Type"], timePandas["Years"])

plt.xlabel("Years")
plt.xticks(fontsize="x-small",rotation=45)
plt.ylabel("Trial Type")
plt.yticks(fontsize="x-small")
plt.title("Years taken to complete trial by type")
plt.show()
```

```
/databricks/python/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



```
#violin plots for each type
timePandas = timeDF.toPandas()

plt.figure(figsize = (10, 5))

fig = sns.violinplot(timePandas["Type"], timePandas["Years"],
               inner="quartile")

plt.xlabel("Years")
plt.xticks(fontsize="x-small",rotation=45)
plt.ylabel("Trial Type")
plt.yticks(fontsize="x-small")
plt.title("Years taken to complete trial by type")
plt.show()
```

```
/databricks/python/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



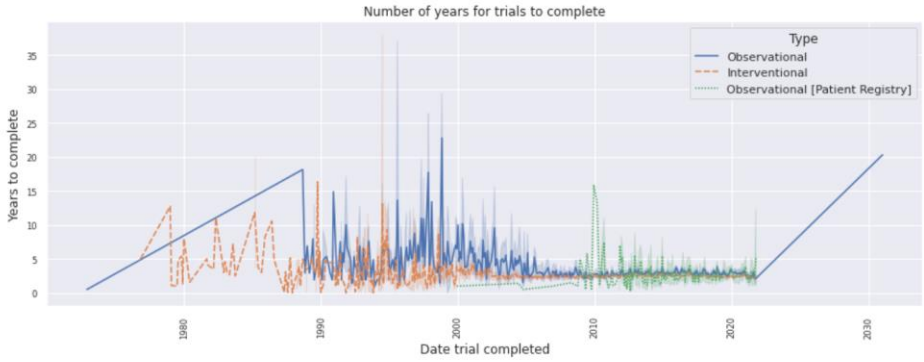
I decided to plot the years taken to complete against the trial's completion date in an effort to reduce the obvious potential relationship between earlier conducted research also having large number of years to complete due to them being completed long-term research, whereas more recent long-term trials may take as long but not have been completed yet. Looking at the plot produced, it does appear over time there has become less variation in the number of years taken to complete trials, trials appear to overall be taking less time. Although there is still more variation in the years to complete for observational trials using the patient registry. Additionally, if the dataset is complete, it appears interventional trials were used more earlier on in time, followed by observational, and using the patient registry has only recently begun being used.

```
#years taken to complete with date of completion of trials
timeSeriesCompleted=timeDF.select("Type","Completion","Years").withColumn("Completion", to_timestamp(unix_timestamp(col("Completion"),"MMM yyyy"))).orderBy("Completion").toPandas()
timeSeriesCompleted
```

	Type	Completion	Years
0	Observational	1972-12-01	0.500000
1	Interventional	1976-11-01	5.000000
2	Interventional	1979-01-01	12.750000
3	Interventional	1979-02-01	1.083333
4	Interventional	1979-07-01	1.000000
...	...	...	...
202443	Interventional	2021-10-01	0.666667
202444	Interventional	2021-10-01	3.333333
202445	Interventional	2021-10-01	1.000000
202446	Interventional	2021-10-01	5.583333
202447	Observational	2031-01-01	20.333333

202448 rows × 3 columns

```
#years taken to complete trial according to time reported completed
#each type of trial
plt.figure(figsize = (15, 5))
sns.lineplot(data = timeSeriesCompleted, x="Completion", y="Years",hue="Type", style="Type")
plt.xlabel("Date trial completed")
plt.xticks(fontsize="x-small",rotation=90)
plt.ylabel("Years to complete")
plt.yticks(fontsize="x-small")
plt.title("Number of years for trials to complete")
plt.show()
```



I then wanted to look into the top conditions for trials completed in 2021, where the top condition was unsurprisingly COVID-19 followed by the previously seen popular conditions, including cancers. This was also true for the studies started from 2020 onwards. However, when looking at those which were started from 2020 onwards which have the status “Completed”, the cancer-related conditions are no longer in the top five. As suggested earlier, cancer trials (those with the root C04) may take longer than non-cancer trials to complete.

```
#conditions for completed trials in 2021
trial_nodupl.filter(col("Status")=="Completed").filter(col("Completion").contains(year)).select("Conditions").withColumn("Conditions",explode(split("Conditions",","))).groupBy("Conditions").count().orderBy("count",
ascending=False).limit(5).display()
```

	Conditions	count
1	COVID-19	658
2	Diabetes Mellitus	222
3	Carcinoma	205
4	Syndrome	155
5	Neoplasms	146

Showing all 5 rows.

```
# top 5 conditions in trials submitted from 2020 onwards
trial_nodupl.filter(col("Start").contains("202")).select("Conditions").withColumn("Conditions",explode(split("Conditions",","))).groupBy("Conditions").count().orderBy("count", ascending=False).limit(5).display()
```

	Conditions	count
1	COVID-19	5481
2	Carcinoma	2222
3	Neoplasms	1555
4	Diabetes Mellitus	1245
5	Syndrome	1187

Showing all 5 rows.

```
#top 5 conditions in trials submitted in 2020 which were completed
trial_nodupl.filter(col("Start").contains("202")).filter(col("Status")=="Completed").select("Conditions").withColumn("Conditions",explode(split("Conditions",","))).groupBy("Conditions").count().orderBy("count",
ascending=False).limit(5).display()
```

	Conditions	count
1	COVID-19	1379
2	Coronavirus Infections	183
3	Diabetes Mellitus	156
4	Infections	148
5	Syndrome	120

When plotting the number of years taken to complete against the date the trial was completed, it appears once the noise is reduced after 2000 cancer trials appear to take longer to complete than non-cancer trials.

```
#years taken to complete non-cancer and cancer trials with the date completed
timeC04=timeConditions.join(rootdf,rootdf["term"]==timeConditions["Conditions"], "left").withColumn("Condition", when(col("root")=="C04", "Cancer").otherwise("Non-cancer")).dropDuplicates(["Id","root"]).select("Condition","Completion",
"Years").withColumn("Completion", to_timestamp(unix_timestamp(col("Completion"),"MMM yyyy"))).orderBy("Years").toPandas()
timeC04
```

	Condition	Completion	Years
0	Non-cancer	1996-08-01	0.000000
1	Cancer	2004-01-01	0.000000
2	Non-cancer	2006-07-01	0.000000
3	Non-cancer	2006-11-01	0.000000
4	Non-cancer	2006-12-01	0.000000
...	...	...	...
336655	Non-cancer	2008-09-01	60.166667
336656	Non-cancer	2008-09-01	60.166667
336657	Non-cancer	1994-07-01	63.083333
336658	Non-cancer	1994-07-01	63.083333
336659	Non-cancer	1994-07-01	63.083333

336660 rows × 3 columns

```
#plotting years taken to complete trials according to completion date
#cancer vs non-cancer trials
plt.figure(figsize = (15, 5))
sns.lineplot(data = timeC04, x="Completion", y="Years", hue="Condition", style="Condition")
plt.xlabel("Date trial completed")
plt.xticks(fontsize="x-small",rotation=90)
plt.ylabel("Years to complete")
plt.yticks(fontsize="x-small")
plt.title("Number of years for trials to complete")
plt.show()
```