Big Data Tools and Technique Assignment

By

Syed Ali Murtaza

1. Part-01

STEPS TO IMPLEMENT THE PROJECT

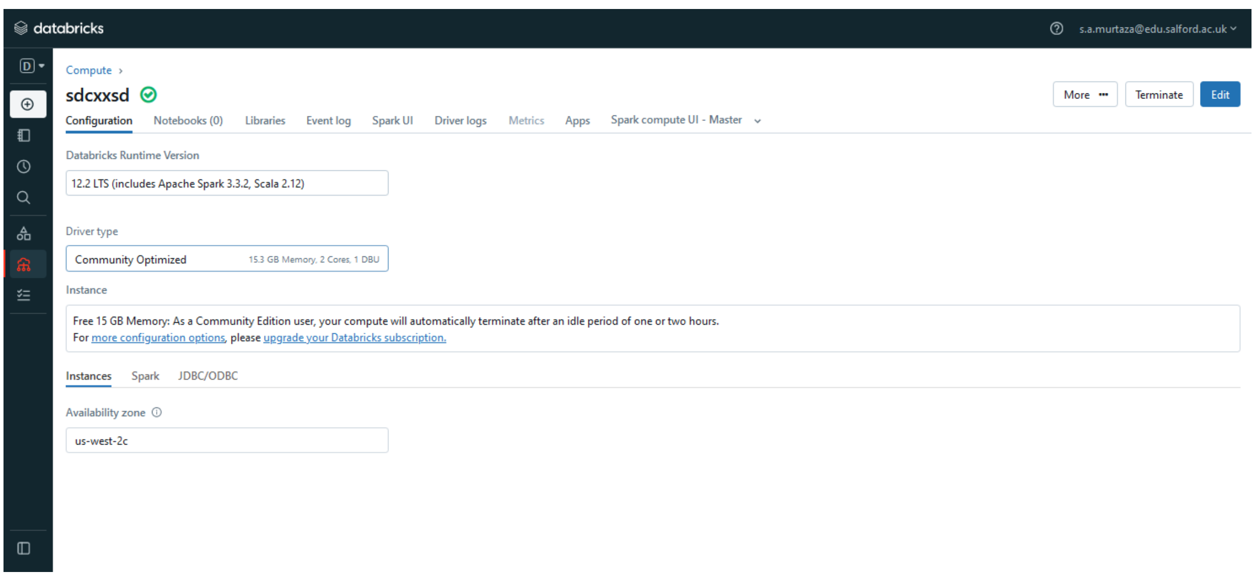
Description of the necessary configuration or prerequisites needed to successfully accomplish this task:-

In this part we have given different files with the name of clinicaltrial-2023, clinicaltrial-2021 and clinicaltrial-2020. So we need databricks platform to perform this task and we have to clean the data and answer the questions given in this task. So I follow several steps to answer the question.

1. **Fire up the Databricks workspace:-**

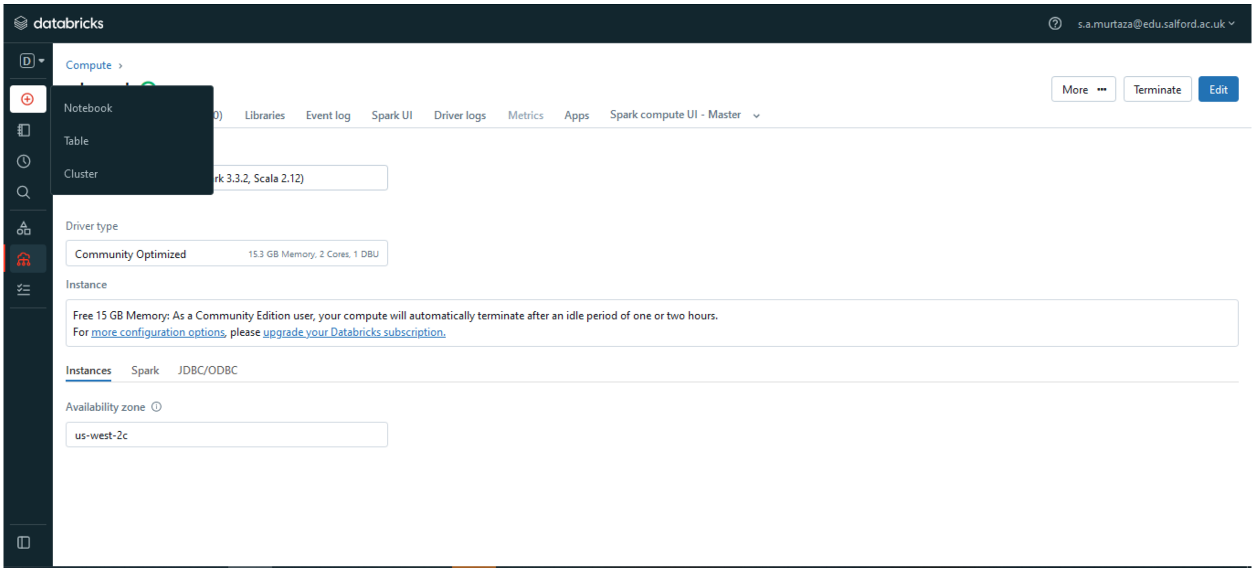
I logged in to the data bricks community edition account.

Than I created a cluster to start my analyses, because a cluster will give me access to use a machine. So for this purpose I clicked on compute and type the name of the cluster and select the most recent runtime as in this screen shot.

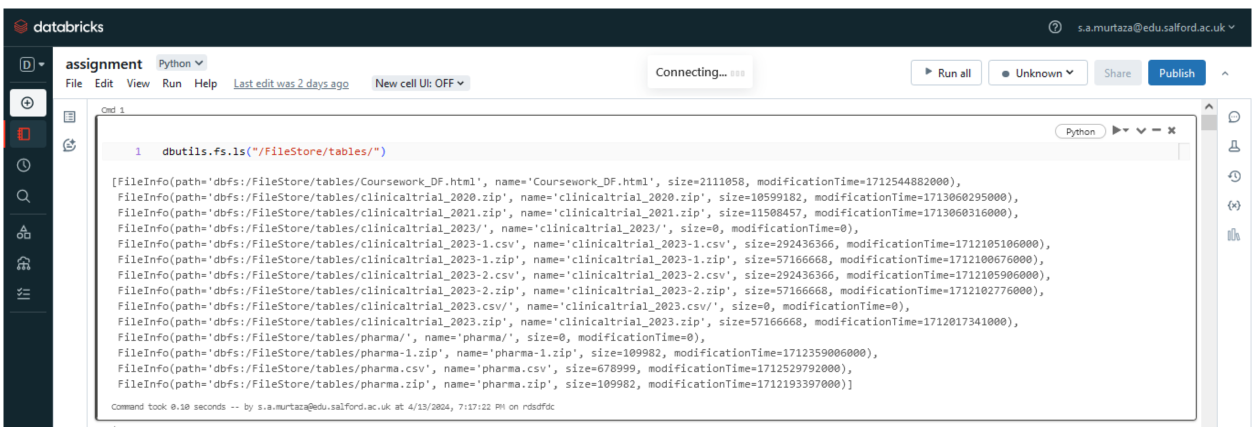


2. **Creating a Notebook:-**

Then I click on “PLUS” icon and select notebook.



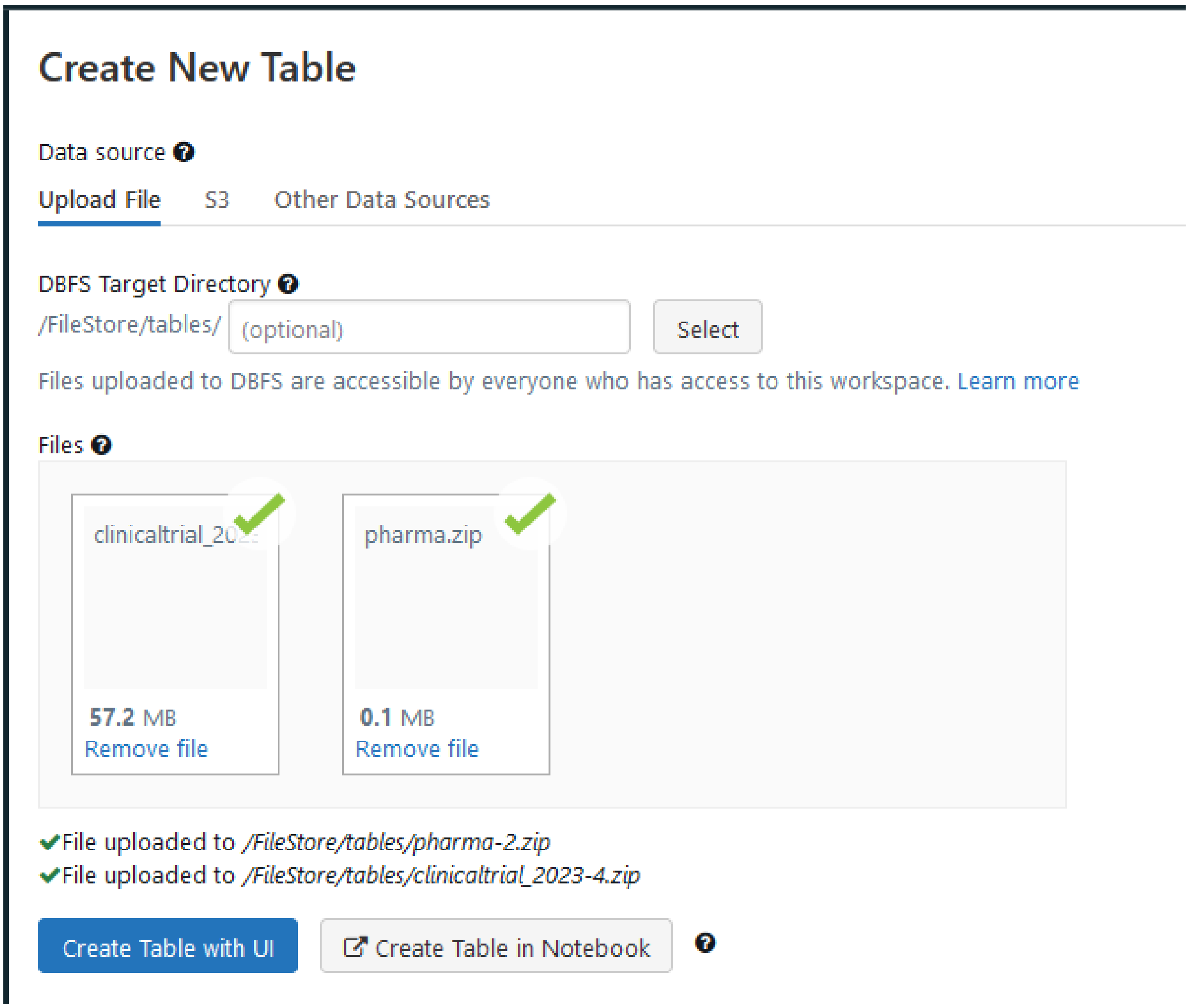
Then I name the notebook and attach the cluster (that I started before) to this notebook.



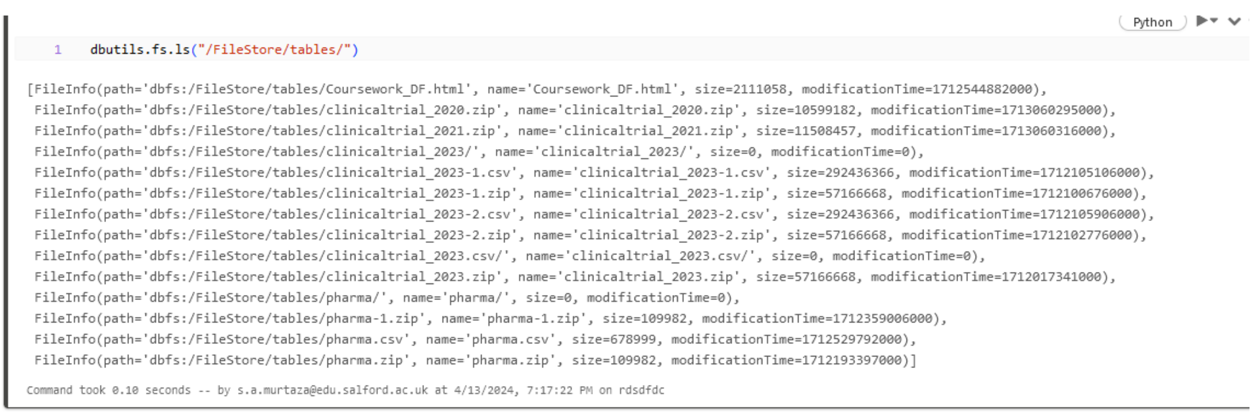
3. Data cleaning and preparation:-

Before answering the questions; we need to clean and prepare the data because the data is not in a correct format it contains all the values in a single row and single column, So I follow various steps to unzip the file and handle missing values, transform the data types and completely ensure data integrity.

1. **Load the dataset:** Load the files into the databricks.



All the files that I uploaded is present in this path /FileStore/tables/ by default. Now to ensure that if the file is uploaded successfully or not I use the dbutils.fs.ls command.



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

[FileInfo(path='dbfs:/FileStore/tables/Coursework_DF.html', name='Coursework_DF.html', size=2111058, modificationTime=1712544882000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2020.zip', name='clinicaltrial_2020.zip', size=10599182, modificationTime=1713060295000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2021.zip', name='clinicaltrial_2021.zip', size=11508457, modificationTime=1713060316000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2023/', name='clinicaltrial_2023/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2023-1.csv', name='clinicaltrial_2023-1.csv', size=292436366, modificationTime=1712105106000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2023-1.zip', name='clinicaltrial_2023-1.zip', size=57166668, modificationTime=1712100676000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2023-2.csv', name='clinicaltrial_2023-2.csv', size=292436366, modificationTime=1712105906000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2023-2.zip', name='clinicaltrial_2023-2.zip', size=57166668, modificationTime=1712102776000),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2023.csv/', name='clinicaltrial_2023.csv/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2023.zip', name='clinicaltrial_2023.zip', size=57166668, modificationTime=1712017341000),
 FileInfo(path='dbfs:/FileStore/tables/pharma/', name='pharma/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/FileStore/tables/pharma-1.zip', name='pharma-1.zip', size=109982, modificationTime=1712359006000),
 FileInfo(path='dbfs:/FileStore/tables/pharma.csv', name='pharma.csv', size=678999, modificationTime=1712529792000),
 FileInfo(path='dbfs:/FileStore/tables/pharma.zip', name='pharma.zip', size=109982, modificationTime=1712193397000)]

Command took 0.10 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/13/2024, 7:17:22 PM on rdsdfdc
```

Now from above we can see that both the files are present.

2. Unzipping the file:-

As the uploaded files are in zip form and dbutils toolkit has no tool to unzip So I copy the files to local file system and extract here using the shell command. After unzipping I put the extracted content back into DBFS.
So first I am going to copy both file from DBFS to local file system using dbutils command.

```
Cmd 3
dbutils.fs.cp("/FileStore/tables/clinicaltrial_2023.zip", "file:/tmp/")

Out[2]: True
Command took 4.17 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/22/2024, 4:14:32 PM on dfvcds
```

```
Cmd 6
dbutils.fs.cp("/FileStore/tables/pharma.zip", "file:/tmp/")

Out[5]: True
Command took 0.69 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/22/2024, 4:15:07 PM on dfvcds
```

Now using ls command I checked that whether it is in shell command or not.

```
Cmd 7
%sh
ls /tmp/

Rserv
RtmpxJ1AIn
chauffeur-daemon-params
chauffeur-daemon.pid
chauffeur-env.sh
clinicaltrial_2021.zip
clinicaltrial_2023.zip
custom-spark.conf
driver-daemon-params
driver-daemon.pid
driver-env.sh
hsperfdata_root
pharma.zip
python_lsp_logs
systemd-private-6271a62ed2b044e18cf155510f6bfbdd-apache2.service-Yz0XGi
systemd-private-6271a62ed2b044e18cf155510f6bfbdd-ntp.service-5eIkfj
systemd-private-6271a62ed2b044e18cf155510f6bfbdd-systemd-logind.service-SqU12g
systemd-private-6271a62ed2b044e18cf155510f6bfbdd-systemd-resolved.service-3fdDWi
tmp.hQysSw0PF6
Command took 0.25 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/22/2024, 4:15:11 PM on dfvcds
```

From above SS we can see that both files are in tmp directory.

Than I have used UNIX command (Unzip) to unzip the files.

```
Cmd 8
%sh
unzip -d /tmp/ /tmp/clinicaltrial_2023.zip

unzip:  cannot find or open /tmp/clinicaltrial_2023.zip, /tmp/clinicaltrial_2023.zip.zip or /tmp/clinicaltrial_2023.zip.ZIP.
Command took 0.12 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/13/2024, 7:19:43 PM on rdsdfdc
```

```
Cmd 11
%sh
unzip -d /tmp/ /tmp/pharma.zip

Archive:  /tmp/pharma.zip
  inflating: /tmp/pharma.csv
Command took 0.19 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/11/2024, 5:11:29 PM on hjnk
```

Now, I aim to transfer the files from the local file system to DFBS. So for this purpose I make the directories.

```
Cmd 17
dbutils.fs.mkdirs("FileStore/tables/clinicaltrial_2023")

Out[36]: True
Command took 0.10 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/11/2024, 5:11:29 PM on hjnk
```

```
Cmd 20
dbutils.fs.mkdirs("FileStore/tables/pharma")

Out[37]: True
Command took 0.19 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/11/2024, 5:11:29 PM on hjnk
```

Now I'm transferring the files from the local file system to DBFS using the command fs.mv.

```
Cmd 21
dbutils.fs.mv("file:/tmp/clinicaltrial_2023.csv", "/FileStore/tables/clinicaltrial_2023", True)

Out[38]: True
Command took 19.53 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/11/2024, 5:11:29 PM on hjnk
```

```
Cmd 24
dbutils.fs.mv("file:/tmp/pharma.csv", "/FileStore/tables/pharma", True)

Out[39]: True
Command took 0.39 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/11/2024, 5:11:29 PM on hjnk
```

I verify the contents of the file located at '/FileStore/tables/FileName' to ensure it contains the expected files

```
Cmd 25
dbutils.fs.ls("FileStore/tables/clinicaltrial_2023/")

Out[40]: [FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2023/cleaned_clinicaltrial_2023.csv/', name='cleaned_clinicaltrial_2023.csv/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/FileStore/tables/clinicaltrial_2023/clinicaltrial_2023.csv', name='clinicaltrial_2023.csv', size=292436366, modificationTime=1712880714000)]
Command took 0.19 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/11/2024, 5:11:29 PM on hjnk
```

```
Cmd 28
dbutils.fs.ls("FileStore/tables/pharma/")

Out[41]: [FileInfo(path='dbfs:/FileStore/tables/pharma/pharma.csv', name='pharma.csv', size=678999, modificationTime=1712880717000)]
Command took 0.19 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/11/2024, 5:11:29 PM on hjnk
```

Then I use the dbutils.fs.head command to see the format of file

1

clinicaltrial-2023. Although it is not well formatted and looks alike as in this screen shot.

```
Python

dbutils.fs.head("FileStore/tables/clinicaltrial_2023/clinicaltrial_2023.csv")

[Truncated to first 65536 bytes]
Out[44]: '"Id\tStudy Title\tAcronym\tStatus\tConditions\tInterventions\tSponsor\tCollaborators\tEnrollment\tFunder Type\tType\tStudy Design\tStart\tCompletio
n",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NCT03630471\tEffectiveness of a Problem-solving Intervention for Common Adolescent Mental Health Problems in Indi
a\tPRIDE\tCOMPLETED\tMental Health Issue (E.G.", Depression, Psychosis, Personality Disorder," Substance Abuse)\tBEHAVIORAL: PRIDE \'Step 1\' problem-solving interve
ntion|BEHAVIORAL: Enhanced usual care\tSangath\tHarvard Medical School (HMS and HSDM)|London School of Hygiene and Tropical Medicine\t250.0\tOTHER\tINTERVENTIONAL\tA
llocation: RANDOMIZED|Intervention Model: PARALLEL|Masking: DOUBLE (INVESTIGATOR"," OUTCOMES_ASSESSOR)|Primary Purpose: TREATMENT\t2018-08-20\t2019-02-2
8",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NCT05992571\tOral Ketone Monoester Supplementation and Resting-state Brain Connectivity\t\tRECRUITING\tCerebrovascular
Function|Cognition\tOTHER: Placebo|DIETARY_SUPPLEMENT: β-OHB\tMcMaster University\tAlzheimer\'s Society of Brant", Haldimand Norfolk," Hamilton Halton\t30.0\tOTHER\t
INTERVENTIONAL\tAllocation: RANDOMIZED|Intervention Model: CROSSOVER|Masking: TRIPLE (PARTICIPANT", INVESTIGATOR," OUTCOMES_ASSESSOR)|Primary Purpose: BASIC_SCIENCE
\t2023-10-25\t2024-0
8",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NCT00237471\tImpact of Tight Glycaemic Control in Acute Myocardial Infarction\t\tTERMINATED\tMyocardial Infarct|Hyper
glycemia\tDRUG: Insulin (tight blood glucose control)\tMelbourne Health\tNational Health and Medical Research Council"," Australia|Bristol-Myers Squibb\t40.0\tOTHER
\tINTERVENTIONAL\tAllocation: RANDOMIZED|Intervention Model: PARALLEL|Masking: NONE|Primary Purpose: TREATMENT\t2005-10\t2006-0
5",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\r\n"NCT03820271\tNew Prognostic Predictive Models of Mortality of Decompensated Cirrhotic Patients Waiting for Liver T
ransplantation\tSUPERMELD\tRECRUITING\tDecompensated Cirrhosis|Liver Transplantation\tOTHER: SuperMELD\tAssistance Publique - Hôpitaux de Paris\t\t500.0\tOTHER\tINTE
RVENTIONAL\tAllocation: NA|Intervention Model: SINGLE_GROUP|Masking: NONE|Primary Purpose: OTHER\t2020-10-01\t2023-10-0
1",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
Command took 0.39 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/11/2024, 5:11:29 PM on hjnk
```

3. Cleaning and Preparation of Data:-

1. Initializing Spark:-

First of all I initialize the spark session.

```
Cmd 2
Python


from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType

# Initialize Spark session
spark = SparkSession.builder.appName("Data Cleaning").getOrCreate()

Command took 0.13 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/23/2024, 6:45:58 PM on fdscsx
```

2. Defining Schema:-

I defined schema using StructType and StructFeild. StructFeild is used to define the column names of the data frame while StructType is used to define the data type of the column. I define all the entries as string. This explicit schema definition ensures data consistency and integrity.

1

Cmd 3

```python

schema = StructType([
    StructField("Id", StringType(), True),
    StructField("Study Title", StringType(), True),
    StructField("Acronym", StringType(), True),
    StructField("Status", StringType(), True),
    StructField("Conditions", StringType(), True),
    StructField("Interventions", StringType(), True),
    StructField("Sponsor", StringType(), True),
    StructField("Collaborators", StringType(), True),
    StructField("Enrollment", StringType(), True),
    StructField("Funder Type", StringType(), True),
    StructField("Type", StringType(), True),
    StructField("Study Design", StringType(), True),
    StructField("Start", StringType(), True),
    StructField("Completion", StringType(), True),
])
```

Command took 0.07 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/23/2024, 4:41:51 PM on fdscsx

3. Load the data:-

Loading data from a file is necessary to perform data cleaning. So I loaded the data into an rdd using spark.sparkContext.textFile(file_path).

Cmd 4

```python

file_path = "/FileStore/tables/clinicaltrial_2023/clinicaltrial_2023.csv"
raw_rdd = spark.sparkContext.textFile(file_path)
```

Command took 0.55 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/23/2024, 4:42:52 PM on fdscsx

4. Deleting a directory and its content:-

In this code I makes a function name delete_dir to recursively delete directories and their contents in Databricks FileStore. Then I specifies directories or files to be deleted and deletes each path using the defined function. Finally, I print a message indicating the completion of cleanup.

```python
# No need to mount DBFS root (it's already mounted at `/FileStore`)
# Function to recursively delete a directory and its contents
def delete_dir(path):
  try:
    for file in dbutils.fs.ls(path):
      if file.isDir():
        delete_dir(file.path)  # Recursive call for subdirectories
      else:
        dbutils.fs.rm(path, recurse=True)  # Use fs.rm for files in FileStore
    dbutils.fs.rm(path, recurse=True)  # Delete the empty directory itself
  except Exception as e:
    print(f"Error deleting directory {path}: {e}")

# Specify directories or files to be deleted (replace with your specific paths)
paths_to_delete = [
  "/FileStore/tables/clinicaltrial_2023-2.csv"
]

```

```
# Delete each path
for path in paths_to_delete:
    delete_dir(path)

# Unmount DBFS (optional, good practice for security)
# Since we didn't mount anything, this line is not necessary

print("Cleanup completed!")

```

```
Cleanup completed!
Command took 0.42 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/29/2024, 5:50:41 AM on dscdsxc
```

4. Processing and Cleaning Data:-

The file contains the delimiter /t between each line and various unnecessary commas, strips leading and trailing quotes, and additional whitespaces that I realize after seeing the file.

So to remove these I defined a UDF which will remove commas, strips leading and trailing quotes, and remove any additional whitespace. This UDF also ensure that each row has same length as the header by padding missing values with empty string.

Than I use a map function on RDD to split each line by a delimeter /t.

```python

def clean_and_pad(parts):
    # Remove commas, strip quotes and whitespace
    cleaned_parts = [part.replace(",", "").strip().strip('"') for part in parts]
    # Pad the row if it has fewer elements than expected
    if len(cleaned_parts) < 14:
        cleaned_parts += [""] * (14 - len(cleaned_parts))
    return cleaned_parts

processed_rdd = raw_rdd.map(lambda line: line.split("\t")).map(clean_and_pad)


# Filter out the header if it's the first row and matches expected headers
header = processed_rdd.first()  # Assuming the first row is the header
data_rdd = processed_rdd.filter(lambda row: row != header and len(row) == 14)  # Ensure all rows have exactly 14 elements

# Create DataFrame
df = spark.createDataFrame(data_rdd, schema=schema)
df.show()

```

(2) Spark Jobs

Output:-

1

```
+-----------+--------------------+----------+------------------+--------------------+--------------------+--------------------+--------------------+----------+
-----------+-------------+--------------------+----------+----------+
|         Id|         Study Title|   Acronym|            Status|          Conditions|       Interventions|             Sponsor|       Collaborators|Enrollment|Funder
Type|         Type|        Study Design|     Start|Completion|
+-----------+--------------------+----------+------------------+--------------------+--------------------+--------------------+--------------------+----------+
-----------+-------------+--------------------+----------+----------+
|NCT03630471|Effectiveness of ...|     PRIDE|         COMPLETED|Mental Health Iss...|BEHAVIORAL: PRIDE...|             Sangath|Harvard Medical S...|     250.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|2018-08-20|2019-02-28|
|NCT05992571|Oral Ketone Monoe...|          |        RECRUITING|Cerebrovascular F...|OTHER: Placebo|DI...| McMaster University|Alzheimer's Socie...|      30.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|2023-10-25|   2024-08|
|NCT00237471|Impact of Tight G...|          |        TERMINATED|Myocardial Infarc...|DRUG: Insulin (ti...|    Melbourne Health|National Health a...|      40.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|   2005-10|   2006-05|
|NCT03820271|New Prognostic Pr...| SUPERMELD|        RECRUITING|Decompensated Cir...|    OTHER: SuperMELD|Assistance Publiq...|                    |     500.0|
OTHER|INTERVENTIONAL|Allocation: NA|In...|2020-10-01|2023-10-01|
|NCT06229171|InTake Care: Deve...|InTakeCare|NOT_YET_RECRUITING|Hypertension|Trea...|OTHER: adherence ...|Istituto Auxologi...|Istituti Clinici ...|     206.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|2024-10-01|2026-04-01|
|NCT02945371|Tailored Inhibito...|       REV|         COMPLETED|Smoking|Alcohol D...|BEHAVIORAL: Perso...|University of Oregon|                    |     103.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|   2014-09|   2016-05|
|NCT01055171|Neuromodulation o...|          |         COMPLETED|Alcohol Dependenc...|DRUG: Propranolol...|Medical Universit...|National Institut...|      44.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|   2010-01|   2012-08|
|NCT01125371|Computerized Brie...|          |         COMPLETED|Alcohol: Harmful ...|BEHAVIORAL: Compu...|Johns Hopkins Uni...|National Institut...|     439.0|
```

Command took 14.71 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/23/2024, 4:43:58 PM on fdscsx

Now from the output we can see that the data is now cleaned and is prepare to analyze.

5. Date format:-

As the file contain various date format in the Start and Completion column of the file. So this correction is achieved by using 2 UDF.

First UDF (delete_day_udf):

This UDF will use a regular expression with the pattern (r'(\b\d(4)-\d(2))-\d(2)\b') to identify date string in the format YYYY-MM-DD and than use replacement pattern r'\1' to remove the day part from the date because it would be easy for me to analyze the dates by removing unnecessary day part.

Second UDF (format_date):

This UDF will attempt to parse the string(date values stored as a string in start and completion column) using datetime.strptime with the format %Y-%m. If the parsing is successful than this UDF create a date_obj and format it using strftime with the format %b%Y .So in this way this UDF standardizes the date formate.

Cmd 6

Python

```python

from pyspark.sql.types import StructType, StructField, StringType, DateType
from pyspark.sql.functions import col, udf, lit
import re
def delete_day(date_str):
    """Removes the day part from a date string (if present) using regular expressions."""
    return re.sub(r'(\b\d(4)-\d(2))-\d(2)\b', r'\1', str(date_str))
def format_date(date_str):
    """Formats a date string (YYYY-MM) into a more human-readable format (Month Year)."""
    if date_str:
        try:
            date_obj = datetime.strptime(date_str, "%Y-%m")
            return date_obj.strftime("%b %Y")
        except ValueError:
            return date_str
    else:
        return date_str

```

Command took 0.11 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/23/2024, 6:40:59 PM on fdscsx

1

Now I am going to apply UDFs in correct order so for this purpose I apply **delete_day_udf to** "Start" and "completion" column, and create a new column with the name Start_Cleaned and Completion_cleaned respectively. Then I Apply the format_date_udf to the 'Start_Cleaned' column and 'Completion_Cleaned' column, attempting to parse and format the date string.

Cmd 6

Python

```python

# Define UDFs
delete_day_udf = udf(delete_day, StringType())
format_date_udf = udf(format_date, StringType())
# Apply UDFs in the correct order
df = df.withColumn("Start_Cleaned", delete_day_udf(col("Start")))
df = df.withColumn("Completion_Cleaned", delete_day_udf(col("Completion")))
df = df.withColumn("Start_Date", format_date_udf(col("Start_Cleaned")))
df.withColumn("Completion_Date", format_date_udf(col("Completion_Cleaned")))
```

Command took 0.11 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/23/2024, 6:40:59 PM on fdscsx

Now in this way we have corrected the date format using user defined function. But the issue is that we have now multiple columns for start and completion, so to resolve this issue I am going to delete the previous(original) start and completion column to avoid the confusion.

Cmd 7

Python

```python

df = df.drop("Start")
df = df.drop("Completion")

```

df: pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 12 more fields]

Command took 0.13 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/23/2024, 6:30:33 PM on fdscsx

6. Temporary View:-

I also define the temporary view of above data frame so that I can view the data clearly and apply SQL querries on this temporary view. I use createOrReplaceTempView command for making this temporary view.

Cmd 8

Python

```python

df.createOrReplaceTempView("clinical_trials")

# Show the cleaned DataFrame
df.show()
```

(1) Spark Jobs

Output:-

1

```
(1) Spark Jobs
+-----------+--------------------+----------+-------------------+--------------------+--------------------+--------------------+--------------------+----------+
-----------+--------------+--------------------+-------------+------------------+
|         Id|         Study Title|   Acronym|             Status|          Conditions|       Interventions|             Sponsor|       Collaborators|Enrollment|Funder
Type|          Type|        Study Design|Start_Cleaned|Completion_Cleaned|
+-----------+--------------------+----------+-------------------+--------------------+--------------------+--------------------+--------------------+----------+
-----------+--------------+--------------------+-------------+------------------+
|NCT03630471|Effectiveness of ...|     PRIDE|          COMPLETED|Mental Health Iss...|BEHAVIORAL: PRIDE...|             Sangath|Harvard Medical S...|     250.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|   2018-08-20|        2019-02-28|
|NCT05992571|Oral Ketone Monoe...|          |         RECRUITING|Cerebrovascular F...|OTHER: Placebo|DI...| McMaster University|Alzheimer's Socie...|      30.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|   2023-10-25|           2024-08|
|NCT00237471|Impact of Tight G...|          |         TERMINATED|Myocardial Infarc...|DRUG: Insulin (ti...|    Melbourne Health|National Health a...|      40.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|      2005-10|           2006-05|
|NCT03820271|New Prognostic Pr...| SUPERMELD|         RECRUITING|Decompensated Cir...|    OTHER: SuperMELD|Assistance Publiq...|                    |     500.0|
OTHER|INTERVENTIONAL|Allocation: NA|In...|   2020-10-01|        2023-10-01|
|NCT06229171|InTake Care: Deve...|InTakeCare|NOT_YET_RECRUITING|Hypertension|Trea...|OTHER: adherence ...|Istituto Auxologi...|Istituti Clinici ...|     206.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|   2024-10-01|        2026-04-01|
|NCT02945371|Tailored Inhibito...|       REV|          COMPLETED|Smoking|Alcohol D...|BEHAVIORAL: Perso...|University of Oregon|                    |     103.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|      2014-09|           2016-05|
|NCT01055171|Neuromodulation o...|          |          COMPLETED|Alcohol Dependenc...|DRUG: Propranolol...|Medical Universit...|National Institut...|      44.0|
OTHER|INTERVENTIONAL|Allocation: RANDO...|      2010-01|           2012-08|
|NCT01125371|Computerized Brie...|          |          COMPLETED|Alcohol: Harmful ...|BEHAVIORAL: Compu...|Johns Hopkins Uni...|National Institut...|     439.0|
Command took 4.57 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/23/2024, 6:30:43 PM on fdscsx
```

Now from above we can see that the data is now completely cleaned is prepared for further analyzing which I am going to do so**.**

Question#1:

The number of studies in the dataset. You must ensure that you explicitly check distinct studies.

Assumptions:-

The assumptions that I made before answering this question are

1. Each row in the dataset is a unique clinical trial.
2. The column name “ID” is used to represent each clinical trial.
3. The dataset is assumed to be properly cleaned and formatted.

SQL implementation:-

Answer:-
I used the count function to count the distinct rows from the temporary view named “Clinical_trial” by applying the distinct keyword. This will ensures that only the rows will be counted which are distinct.

Code:-

1

Result:-

From above code we get the result that there are 483422 distinct rows which means that there are 483422 numbers of distinct studies are present in the dataset.

DF Implementation

Answer:-
To find the distinct number of studies I use df.distinct command to remove all duplicate rows and then I use count function to only count distinct row.

Code:-

Result:-

From above you can see that distinct number of studies is 483422.

RDD implementation

To find the distinct number of studies I use lambda function which takes every row as an input and extract only the first column and then I use distinct() function to eliminate the duplicate elements and then I use count function to count the distinct elements. As in the assumption that ID is used to represent each clinical trial so I did this process on ID which will be equal to distinct number of studies.

1

Cmd 6

Python

```python


# Assuming the first column of your RDD is what you're interested in for distinct counts
distinct_first_elements_count = data_rdd.map(lambda row: row[0]).distinct().count()

print(f"Number of distinct studies: {distinct_first_elements_count}")

```

▸ (1) Spark Jobs

```
Number of distinct studies: 483422
```

Command took 11.52 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/27/2024, 4:10:50 AM on fcxsz

Result:-

Distinct number of study is 483422.

Discussion on results:-

In all three implementations, regardless of whether using SQL, DataFrames, or RDDs, the analysis consistently shows that there are 483,422 distinct studies present in the dataset.

Question#2:

You should list all the types (as contained in the Type column) of studies in the dataset along with the frequencies of each type. These should be ordered from most frequent to least frequent.

Assumptions:-

The assumptions that I made before answering this question are

1. I assumed that fields in data are separated by /t delimeter.
2. The header line is considered to be the initial line of the file.
3. The dataset is assumed to be properly cleaned and formatted.
4. The type of study is contained in the “Type” column of the data set.

Answer:

SQL Implementation

I extract the "Type" column from the temporary view "Clinical_trial" and utilize the count function to determine the frequency of each type. Subsequently, I employ the groupby clause to group the rows by the values in the "Type" column, followed by sorting them in descending order with the OrderBy clause

Code:-

1

Cmd 10

SQL

```sql
SELECT clinical_trials.Type, count(*) as count FROM clinical_trials
GROUP BY clinical_trials.Type
ORDER BY count DESC
```

▸ (2) Spark Jobs

Table

	Type	count
1	INTERVENTIONAL	371382
2	OBSERVATIONAL	110221
3	EXPANDED_ACCESS	928
4		891

4 rows | 18.41 seconds runtime

Refreshed 10 days ago

Command took 18.41 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/14/2024, 3:17:54 PM on gtfdvc

Result:-

From the result set we can see that it showcases the various types of studies present in the dataset, along with their respective frequencies arranged in descending order, as outlined below.

Table

	Type	count
1	INTERVENTIONAL	371382
2	OBSERVATIONAL	110221
3	EXPANDED_ACCESS	928
4		891

DF Implementation

To enumerate all study types, I employ the DataFrame's **groupby()** method on the "Type" column, followed by the **count()** function to tally the occurrences of each type. Finally, I arrange the results in ascending order.

Code:-

Cmd 4

Python

```python
df.groupBy('Type').count().orderBy('count', ascending=False).show(4)
```

▸ (2) Spark Jobs

```
+---------------+------+
|           Type| count|
+---------------+------+
| INTERVENTIONAL|371382|
|  OBSERVATIONAL|110221|
|EXPANDED_ACCESS|   928|
|               |   891|
+---------------+------+
```

Command took 19.60 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/14/2024, 3:20:15 PM on gtfdvc

1

Result:-

From above SS you can see all the type of studies with their respective count.

RDD Implementation

In this code I extracts the “Type” values from each row and removes entries with missing values(None). Then I count how many times each unique type appears and sort them by their frequency. At the end I print a table showing the type and its frequency.

```python
# Extract the "Type" column from the RDD and filter out None values
type_rdd = processed_rdd.map(lambda row: row[10]).filter(lambda x: x is not None)

# Count the occurrences of each type
type_counts_rdd = type_rdd.map(lambda type: (type, 1)).reduceByKey(lambda a, b: a + b)

# Sort the counts in descending order
sorted_type_counts = type_counts_rdd.sortBy(lambda x: x[1], ascending=False)

# Collect the sorted counts
sorted_type_counts_list = sorted_type_counts.collect()

# Print the results
print("{:<30} {:<10}".format('Type', 'Frequency'))
for type, count in sorted_type_counts_list:
    print("{:<30} {:<10}".format(type, count))
```

(3) Spark Jobs

Result:-

```
(3) Spark Jobs
Type                          Frequency
INTERVENTIONAL                371382
OBSERVATIONAL                 110221
EXPANDED_ACCESS               928
                              891
Type                          1
Command took 9.81 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/29/2024, 5:17:14 PM on gfvcx
```

Discussion on result:-

Irrespective of the approach employed (be it SQL, DataFrame, or RDD), the analysis furnishes an exhaustive inventory of the various types of studies present in the dataset, delineating their occurrences from highest to lowest frequency as

```
(3) Spark Jobs
Type                          Frequency
INTERVENTIONAL                371382
OBSERVATIONAL                 110221
EXPANDED_ACCESS               928
                              889
Command took 8.18 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/27/2024, 3:50:26 AM on fcxsz
```

Question#3:

The top 5 conditions (from Conditions) with their frequencies.

Assumptions:-

The assumptions that I made before answering this question are

1. The condition column is assumed to be the 5th column (index 4) in the dataset.
2. Conditions are extracted by splitting the "Conditions" column by the delimeter "|".
3. The dataset is properly cleaned and formatted.

SQL Implementation

I created a temporary view with the name "all conditions" .It has explode function which split the values in the condition column of temporary view "clinical_trial 2023" by a delimeter "|".
Than I querry the data by selecting condition column from temporary view "all condition" and count the occurrences of each condition and then I use 'Where' clause to filter out any empty conditions. Then the results are grouped by the condition and sorted in descending order of count to identify the most common conditions. Then finally I limit the output to the top 5 most conditions.

Code:-

Cmd 12

SQL

```

CREATE OR REPLACE TEMP VIEW all_conditions AS
SELECT explode(split(Conditions, '\\|')) AS condition
FROM clinical_trials;

SELECT TRIM(condition) AS condition, COUNT(*) as count
FROM all_conditions
WHERE condition != ''
GROUP BY condition
ORDER BY count DESC
LIMIT 5;

```

▸ (2) Spark Jobs

Result:-

1

▸ (2) Spark Jobs

Table ⌄ +

	condition	count
1	Healthy	9731
2	Breast Cancer	7502
3	Obesity	6549
4	Stroke	4073
5	Hypertension	4022

5 rows | 20.05 seconds runtime   Refreshed 10 days ago

Command took 20.05 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/14/2024, 3:17:54 PM on gtfdvc

DF Implementation

To identify the top 5 conditions along with their frequencies, I extract the "Condition" column from the DataFrame. After splitting this column by the delimiter "|", I utilize the groupby operation to aggregate the data based on the condition column. By applying the count() function, I determine the frequency of each unique condition. Finally, I sort the results in descending order to obtain the top 5 conditions.

Code:-

Cmd 10

Python

```python
from pyspark.sql.functions import explode, split, col, trim

# Define the delimiter for the conditions column
conditions_delimiter = {
    "clinicaltrial_2023": "\\|",
}

# Replace special characters in the delimiter string
delimiter = conditions_delimiter.get("clinicaltrial_2023", "\\|")

# Explode the Conditions column by the delimiter "|", handle empty conditions, and clean whitespace
exploded_df = df.withColumn('Conditions', explode(split(trim(col('Conditions')), delimiter))) \
                .filter(col('Conditions') != '')

# Group by Conditions and count occurrences
condition_counts_df = exploded_df.groupBy('Conditions').count()

# Order by count in descending order and show the top 5
condition_counts_df.orderBy(condition_counts_df['count'].desc()).show(5, truncate=False)

```

▸ (2) Spark Jobs

▸ exploded_df: pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 14 more fields]

▸ condition_counts_df: pyspark.sql.dataframe.DataFrame = [Conditions: string, count: long]

+-------------+-----+

Result:-

▸ (2) Spark Jobs

▸ exploded_df: pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 14 more fields]

▸ condition_counts_df: pyspark.sql.dataframe.DataFrame = [Conditions: string, count: long]

```
+-------------+-----+
|Conditions   |count|
+-------------+-----+
|Healthy      |9731 |
|Breast Cancer|7502 |
|Obesity      |6549 |
|Stroke       |4073 |
|Hypertension |4022 |
+-------------+-----+
only showing top 5 rows
```

Command took 20.73 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/26/2024, 4:35:14 PM on My Cluster

RDD Implementation

1

In this code I split each condition into separate entries. Then I counts how many times each unique condition appears and then finds the top 5 most frequent conditions. At the end I print the table showing the condition with its respective frequency.

Cmd 6

Python

```python
# Explode the Conditions column and filter out empty conditions
exploded_rdd = processed_rdd.flatMap(lambda row:
  # Assuming Conditions is at index 4, check if it's a string before splitting
  [(tuple(cond.split("|")),) for cond in (row[4] or "").split("|") if cond]) \
.filter(lambda x: x[0] != '')

# Count occurrences of each condition
condition_counts = exploded_rdd.map(lambda x: (x[0], 1)) \
                                .reduceByKey(lambda a, b: a + b)

# Sort by count in descending order and take the top 5
top_5_conditions = condition_counts.sortBy(lambda x: x[1], ascending=False).take(5)

# Print the top 5 conditions and their frequencies
for condition, count in top_5_conditions:
    print(f"Condition: {condition}, Count: {count}")
```

▸ (3) Spark Jobs

Result:-

▸ (3) Spark Jobs

```
Condition: ('Healthy',), Count: 9731
Condition: ('Breast Cancer',), Count: 7502
Condition: ('Obesity',), Count: 6549
Condition: ('Stroke',), Count: 4073
Condition: ('Hypertension',), Count: 4022
```

Command took 13.33 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/29/2024, 5:31:57 PM on gfvcx

Discussion on Result:-

From all three methods RDD, data frame and SQL we got the same answer.

Question#4:

Find the 10 most common sponsors that are not pharmaceutical companies, along with the number of clinical trials they have sponsored. Hint: For a basic implementation, you can assume that the Parent Company column contains all possible pharmaceutical companies.

Answer:-

To answer this question I read the CSV file name pharmaceutical_data.csv into a DF and then I only extract the column name "Parent_Company" and make its temporary view.

Cmd 14

Python

```python
# Assuming your data is stored in a CSV file named "pharmaceutical_data.csv"
file_path = "/FileStore/tables/pharma/pharma.csv"  # Update the file path accordingly

# Read the CSV file into a DataFrame
pharma= spark.read.option("header", "true").csv(file_path)

# Select only the "Parent_Company" column
parent_companies = pharma.select("Parent_Company")

# Show the first few rows of the DataFrame

parent_companies.createOrReplaceTempView("parent_companies_temp")
```

▸ (1) Spark Jobs

▸ pharma: pyspark.sql.dataframe.DataFrame = [Company: string, Parent_Company: string ... 32 more fields]

▸ parent_companies: pyspark.sql.dataframe.DataFrame = [Parent_Company: string]

Command took 4.33 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/26/2024, 5:08:37 AM on gdfcx

Assumptions:-

The assumptions that I made before answering this question are

1. The pharmaceutical dataset has a column name “Parent_company” which include all pharmaceutical companies.
2. The clinical trial data set has a column name “sponsor” which contain the names of all the sponsors of clinical trials.
3. All the data is properly cleaned and well formatted as in schema.
4. The sponsors not present in pharmaceutical companies list are considered to be non-pharmaceutical companies.

SQL Implementation

Cmd 14

SQL

```sql

CREATE OR REPLACE TEMP VIEW non_pharma_sponsor AS SELECT Sponsor FROM clinical_trials WHERE Sponsor NOT IN (SELECT Parent_Company FROM parent_companies_temp);
SELECT Sponsor, count(*) as count FROM non_pharma_sponsor
GROUP BY Sponsor
ORDER BY count DESC
LIMIT 10
```

▸ (3) Spark Jobs

Table +

I created a temporary view with the name non_pharma_sponsor and select the Sponsor column from another temp view “Clinical_trial” using the select command and then use the filter to filter only those Sponsors which are not present in temporary view

1

Parent_Company using the Where clause. Because our aim is to find only non pharmaceutical companies.

Then I count the sponsor column from the temporary view non_pharma_sponsor.

Then I groupby the column with sponsor name.

Then sort the count column in descending order and limit the column to 10.

Result:-

▸ (3) Spark Jobs

Table ⌄ +

	Sponsor	count
1	National Cancer Institute (NCI)	3410
2	Assiut University	3335
3	Cairo University	3023
4	Assistance Publique - Hôpitaux de Paris	2951
5	Mayo Clinic	2766
6	M.D. Anderson Cancer Center	2702
7	Novartis Pharmaceuticals	2393
8	National Institute of Allergy and Infectious Diseases (NIAID)	2340
9	Massachusetts General Hospital	2263
10	National Taiwan University Hospital	2181

10 rows | 20.09 seconds runtime   Refreshed 12 days ago

Command took 20.09 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/14/2024, 3:17:54 PM on gtfdvc

DF Implementation

Now to find the non-pharmaceutical companies I write a code to find the most frequent sponsors in the clinical_trial 2023 data set and the filter out those companies which are present in pharmaceutical companies data.

Cmd 12

Python

```
pharma_list =pharma.select("Parent_Company").rdd.flatMap(lambda x: x).collect()
dfs = df.select("Sponsor")

non_pharma_sponsors = dfs.groupBy("Sponsor").count().orderBy("count", ascending=False).filter(~dfs.Sponsor.isin(pharma_list)).show(10)
```

▸ (3) Spark Jobs

▸ dfs: pyspark.sql.dataframe.DataFrame = [Sponsor: string]

```
+--------------------+-----+
|             Sponsor|count|
+--------------------+-----+
|National Cancer I...| 3410|
|   Assiut University| 3335|
|    Cairo University| 3023|
|Assistance Publiq...| 2951|
|         Mayo Clinic| 2766|
|M.D. Anderson Can...| 2702|
|Novartis Pharmace...| 2393|
|National Institut...| 2340|
|Massachusetts Gen...| 2263|
|National Taiwan U...| 2181|
+--------------------+-----+
only showing top 10 rows
```

Command took 31.29 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/26/2024, 4:40:50 PM on My Cluster

RDD Implementation

To find the non-pharmaceutical sponsor companies, first of all I clean the parent-company column in pharma.csv. Then I apply the logic using rdds that will use clinical trial data set to fetch all the sponsor companies and then subtract those companies which are present in pharma.csv file. In this way we are only left with non-pharma sponsor companies.

Cmd 9

Python

```python
from pyspark.sql import SparkSession

# Initialize Spark session
spark = SparkSession.builder.appName("RDD Example").getOrCreate()

# Define file path
file_path = "/FileStore/tables/pharma/pharma.csv"

# Load the CSV file as an RDD
rdd = spark.sparkContext.textFile(file_path)

# Extract the header
header = rdd.first()

# Split each row by comma
rdd = rdd.map(lambda row: row.split(','))

# Extract the Parent_Company column
parent_companies_rdd = rdd.map(lambda row: (row[1],))  # Assuming Parent_Company is at index 1

# Show the first few rows of the RDD
parent_companies_rdd.take(5)  # You can change 5 to any number to view more or fewer rows

```

(2) Spark Jobs

```
Out[8]: [('"Parent_Company"',)
```

Cmd 10

```python
ct_sponsor_col_index = processed_rdd .first().index('Sponsor')

parent_pharm_comp =rdd.map(lambda x: x[1].replace('"', ''))

processed_rdd.map(lambda x: x[ct_sponsor_col_index]).filter(lambda row: row != 'Sponsor').subtract(rdd.map(lambda x: x[1].replace('"', ''))).map(lambda x:
(x, 1)).reduceByKey(lambda x, y: x + y).sortBy(lambda x: x[1], ascending=False).take(10)
```

(4) Spark Jobs

```
Out[9]: [('National Cancer Institute (NCI)', 3410),
 ('Assiut University', 3335),
 ('Cairo University', 3023),
 ('Assistance Publique - Hôpitaux de Paris', 2951),
 ('Mayo Clinic', 2766),
 ('M.D. Anderson Cancer Center', 2702),
 ('Novartis Pharmaceuticals', 2393),
 ('National Institute of Allergy and Infectious Diseases (NIAID)', 2340),
 ('Massachusetts General Hospital', 2263),
 ('National Taiwan University Hospital', 2181)]
```

Command took 16.70 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/27/2024, 3:55:55 AM on fcxsz

Discussion of result:-

I got the same answer by applying all three implementations (SQL, DataFrame, and RDD) effectively, meeting the requirements of the question.

Question # 05

Plot number of completed studies for each month in 2023. You need to include your visualization as well as a table of all the values you have plotted for each month.

Assumption:-

1. Completion dates of the clinical trial data set are stored in the "Completion_Cleaned" column.
2. The format of completion dates is YY-MM.
3. The status of each clinical trial is stored in "Status" column where "COMPLETED" indicates that the trial has been completed.
4. The data is properly cleaned and well formatted as in schema.

Answer:-

SQL Implementation

First of all I clean up the completion_cleaned date format and extracts the month only, than I only consider those trials with completed status and then group the data by months and count the number of completed trials for each month. I apply the filter using the where clause to only filter the result with year "2023". Then at the end sort the results with months in chronological order.

Cmd 16

SQL

```

-- Use the temporary view "clinical_trials" created by  Python code

SELECT Month, COUNT(*) AS completed_studies
FROM (
  SELECT *,
    SPLIT(Completion_Cleaned, '-')[0] AS Year,
    REPLACE(REPLACE(SPLIT(Completion_Cleaned, '-')[1], ',', ''), '"', '') AS Month
  FROM clinical_trials
  WHERE Status = 'COMPLETED'
) AS processed_data
WHERE Year = '2023'
GROUP BY Month
ORDER BY Month ASC;

```

(2) Spark Jobs

Table

Result:-

1

Table

	Month	completed_studies
1	01	1494
2	02	1272
3	03	1552
4	04	1324
5	05	1415
6	06	1619
7	07	1360
8	08	1230
9	09	1152
10	10	1058
11	11	909
12	12	1082

12 rows | 26.32 seconds runtime   Refreshed 12 days ago

Command took 26.32 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/14/2024, 3:17:54 PM on gtfdvc

Plotting:-

For plotting I convert the cell from SQL to python and read the data from temporary view "clinical_trial" into a spark data frame.

Cmd 19

Python

```
%python
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("ClinicalTrialsPlot").getOrCreate()

# Assuming "clinical_trials" is the temporary view name created by your Spark code

# Read data from the temporary view into a Spark DataFrame
df = spark.sql("SELECT * FROM clinical_trials")
```

df: pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 12 more fields]

Command took 0.09 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/26/2024, 5:18:09 AM on gdfcx

Than I clean the completion_cleaned column and apply filter where status is completed and year is 2023 and order it and group it by months.

Cmd 20

Python

```
%python
from pyspark.sql.functions import split, regexp_replace
import matplotlib.pyplot as plt

# Ensure columns are properly formatted
for col in df.columns:
    df = df.withColumnRenamed(col, col.strip(",").strip('"'))

# Extract data for completed clinical trials in 2023
completed_cd = df.withColumn('Year', split('Completion_Cleaned', "-")[0]) \
                 .withColumn('Month', split('Completion_Cleaned', "-")[1]) \
                 .withColumn('Month', regexp_replace("Month", ",", "")) \
                 .withColumn('Month', regexp_replace("Month", '"', "")) \
                 .filter(df.Status.isin(["COMPLETED"])) \
                 .filter(df.Completion_Cleaned.startswith("2023")) \
                 .select("Month", "Year", "Status")


completed_cd.filter(completed_cd.Year.isin(["2023"])).groupBy("Month").count().orderBy("Month", ascending=True).show()
```

(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 12 more fields]

Output:-

1

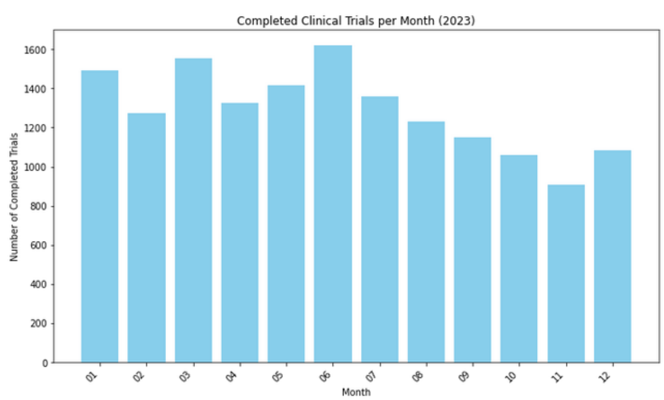
```
▸ (2) Spark Jobs
▸ df: pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 12 more fields]
▸ completed_cd: pyspark.sql.dataframe.DataFrame = [Month: string, Year: string ... 1 more field]
+-----+-----+
|Month|count|
+-----+-----+
|   01| 1494|
|   02| 1272|
|   03| 1552|
|   04| 1324|
|   05| 1415|
|   06| 1619|
|   07| 1360|
|   08| 1230|
|   09| 1152|
|   10| 1058|
|   11|  909|
|   12| 1082|
+-----+-----+

Command took 23.02 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/26/2024, 5:24:48 AM on gdfcx
```

Then I use matplotlib library to plot it.

```python
%python
# Import necessary libraries for plotting
from pyspark.sql.functions import col
import matplotlib.pyplot as plt


# Extract months and counts into separate lists
months = [row["Month"] for row in monthly_counts]
counts = [row["count"] for row in monthly_counts]

# Create the plot
plt.figure(figsize=(10, 6))  # Adjust figure size as desired
plt.bar(months, counts, color='skyblue')
plt.xlabel("Month")
plt.ylabel("Number of Completed Trials")
plt.title("Completed Clinical Trials per Month (2023)")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()

# Display the plot
plt.show()
```

org/firefox/?utm_medium=firefox-des rks-toolbar&utm_campaign=new-users&utm_content=-global

Output:-



Command took 0.41 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/26/2024, 5:43:14 AM on gdfcx

DF Implementation

1

I clean the completion date column using the format YY-MM. Then I filter only those clinical trials whose status is completed in 2023 and group it by “month” and order it by “month” and the plot it using matplotlib library.

```
Cmd 13
Python

from pyspark.sql.functions import split, regexp_replace
import matplotlib.pyplot as plt

# Ensure columns are properly formatted
for col in df.columns:
    df = df.withColumnRenamed(col, col.strip(",").strip('"'))

# Extract data for completed clinical trials in 2023
completed_cd = df.withColumn('Year', split('Completion', "-")[0]) \
                 .withColumn('Month', split('Completion', "-")[1]) \
                 .withColumn('Month', regexp_replace("Month", ",", "")) \
                 .withColumn('Month', regexp_replace("Month", '"', "")) \
                 .filter(df.Status.isin(["COMPLETED"])) \
                 .filter(df.Completion.startswith("2023")) \
                 .select("Month", "Year", "Status")


completed_cd.filter(completed_cd.Year.isin(["2023"])).groupBy("Month").count().orderBy("Month", ascending=True).show()

(2) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 14 more fields]
```

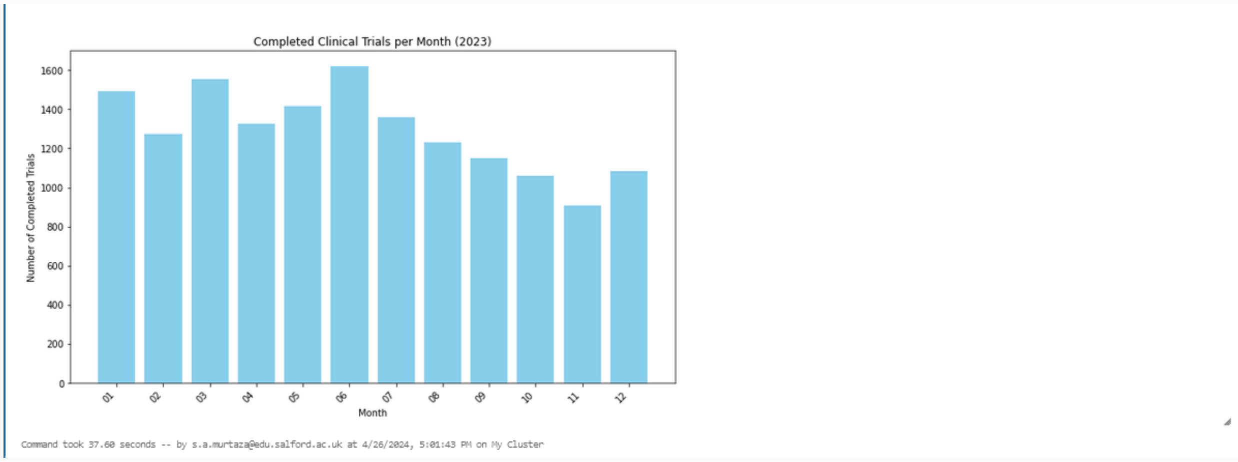
Output:-

```
(2) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [Id: string, Study Title: string ... 14 more fields]
completed_cd: pyspark.sql.dataframe.DataFrame = [Month: string, Year: string ... 1 more field]
+-----+-----+
|Month|count|
+-----+-----+
|   01| 1494|
|   02| 1272|
|   03| 1552|
|   04| 1324|
|   05| 1415|
|   06| 1619|
|   07| 1360|
|   08| 1230|
|   09| 1152|
|   10| 1058|
|   11|  909|
|   12| 1082|
+-----+-----+

Command took 21.29 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/26/2024, 4:54:51 PM on My Cluster
```

Now I plot the above table using matplotlib library.

```
Python

from pyspark.sql.functions import col
import matplotlib.pyplot as plt

# Get monthly counts for completed trials in 2023
monthly_counts = (
    completed_cd.filter(completed_cd.Year.isin(["2023"]))
    .groupBy("Month")
    .count()
    .orderBy("Month", ascending=True)
    .collect()
)
months = [row["Month"] for row in monthly_counts]
counts = [row["count"] for row in monthly_counts]

# Create the plot
plt.figure(figsize=(10, 6))  # Adjust figure size as desired
plt.bar(months, counts, color='skyblue')
plt.xlabel("Month")
plt.ylabel("Number of Completed Trials")
plt.title("Completed Clinical Trials per Month (2023)")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()

# Display the plot
plt.show()

```

Output:-



RDD Implementation

I parse the month, year, and status fields from each record, handling potential errors. Then, I filter for completed studies in 2023, tallying the count of completed studies for each month. Next, I organize the data into a DataFrame with month and count columns, sorting it by month, and present it as a tabular display. Finally, utilizing the matplotlib library, I generate a bar chart illustrating the distribution of completed studies by month.

```python
from pyspark.sql import Row
# Extract necessary fields assuming indexes: Status - 3, Completion - 13
# Completion date expected format "YYYY-MM"
def extract_fields(row):
    try:
        completion = row[13].split("-")
        year = completion[0] if len(completion) > 0 else None
        month = completion[1] if len(completion) > 1 else None
        status = row[3]
        return (month, year, status)
    except IndexError:
        # Handle the error: you can choose to return None or a default value
        return (None, None, None)

extracted_rdd = data_rdd.map(extract_fields).filter(lambda x: None not in x)

# Filter completed studies and specific year
completed_rdd = extracted_rdd.filter(lambda x: x[2] == "COMPLETED" and x[1] == "2023")
```

Command took 0.12 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/29/2024, 6:06:11 PM on gfvcx

Cmd 10

Python

```python


# Count by month
month_counts = completed_rdd.map(lambda x: (x[0], 1)).reduceByKey(lambda a, b: a + b)

# Convert the results into a DataFrame
results_rdd = month_counts.map(lambda x: Row(month=int(x[0]), count=x[1])) if month_counts.first()[0].isdigit() \
                               else month_counts.map(lambda x: Row(month=x[0], count=x[1]))
results_df = spark.createDataFrame(results_rdd)
sorted_results_df = results_df.orderBy("month")

# Show the DataFrame as a table
sorted_results_df.show()



```

(3) Spark Jobs

results_df: pyspark.sql.dataframe.DataFrame = [month: long, count: long]

Output:-

(3) Spark Jobs

results_df: pyspark.sql.dataframe.DataFrame = [month: long, count: long]

sorted_results_df: pyspark.sql.dataframe.DataFrame = [month: long, count: long]

```
+-----+-----+
|month|count|
+-----+-----+
|    1| 1494|
|    2| 1272|
|    3| 1552|
|    4| 1324|
|    5| 1415|
|    6| 1619|
|    7| 1360|
|    8| 1230|
|    9| 1152|
|   10| 1058|
|   11|  909|
|   12| 1082|
+-----+-----+
```

Command took 10.13 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/29/2024, 6:07:30 PM on gfvcx

Plotting

Cmd 11

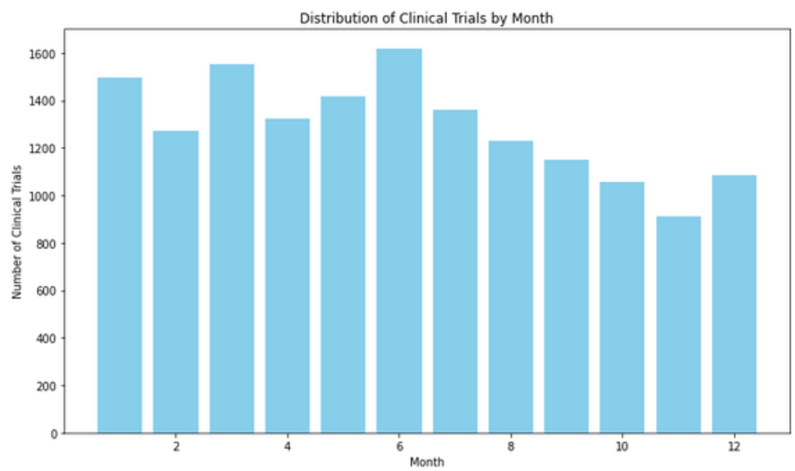
Python

```python
# Plotting (using matplotlib, assumed to be imported)
import matplotlib.pyplot as plt

# Extract month and count from DataFrame
months = [row["month"] for row in sorted_results_df.collect()]  # Collect results as list of dictionaries
counts = [row["count"] for row in sorted_results_df.collect()]

plt.figure(figsize=(10, 6))
plt.bar(months, counts, color='skyblue')
plt.xlabel("Month")
plt.ylabel("Number of Clinical Trials")
plt.title("Distribution of Clinical Trials by Month")
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

(4) Spark Jobs

1

Output:-



Command took 12.06 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/27/2024, 4:05:59 AM on fcxsz

Discussion of Result:-

From above in all three implementations we got the same answer.

Extra features

Question:-

Write a general and reusable code for example for clinicaltrial_2020 and clinicaltrial_2021 datasets.

Answer:-

I write a general code for dataset clinicaltrial_2020 and clinical_trial_2021. In this code I read the data and make its RDD named as raw_rdd. I defined the schema which determine, fields of the data frame along with their data types. Then I define a function which split the data by the delimeter "/|" and pads it with the empty string if the elements are lesser than 9(because 9 elements are in the defined schema) . Then I map that function with raw_rdd. Then I create a data frame using rdd and specified schema.

Cmd 2

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType
import zipfile

# Initialize Spark session
spark = SparkSession.builder.appName("Data Cleaning").getOrCreate()
```

Command took 0.08 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/28/2024, 5:10:07 PM on gfdvdfvc

1

Cmd 3

```python
# Define the schema
schema = StructType([
    StructField("Id", StringType(), True),
    StructField("Sponsor", StringType(), True),
    StructField("Status", StringType(), True),
    StructField("Start", StringType(), True),
    StructField("Completion", StringType(), True),
    StructField("Type", StringType(), True),
    StructField("Submission", StringType(), True),
    StructField("Conditions", StringType(), True),
    StructField("Interventions", StringType(), True),
])

# Load data
# Load data
file_path = "/FileStore/tables/clinicaltrial_2020"
raw_rdd = spark.sparkContext.textFile(file_path)

```

Command took 0.12 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/28/2024, 5:10:48 PM on gfdvdfvc

Cmd 4

Python

```python
# Process and clean data
def clean_and_pad(line):
    # Split line by pipe character
    parts = line.split("|")
    # Pad the row if it has fewer elements than expected
    if len(parts) < 9:
        parts += [""] * (9 - len(parts))
    return parts

processed_rdd = raw_rdd.map(clean_and_pad)
# Filter out the header if it's the first row and matches expected headers
header = processed_rdd.first()  # Assuming the first row is the header
data_rdd = processed_rdd.filter(lambda row: row != header and len(row) == 9)  # Ensure all rows have exactly 9 elements

# Create DataFrame
df = spark.createDataFrame(data_rdd, schema=schema)
# Show the cleaned DataFrame
df.show()

```

(2) Spark Jobs

(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [Id: string, Sponsor: string ... 7 more fields]

```
+-----------+--------------------+--------------------+--------+----------+--------------------+----------+--------------------+--------------------+
|         Id|             Sponsor|              Status|   Start|Completion|                Type|Submission|          Conditions|       Interventions|
+-----------+--------------------+--------------------+--------+----------+--------------------+----------+--------------------+--------------------+
|NCT02758028|The University of...|          Recruiting|Aug 2005|  Nov 2021|      Interventional|  Apr 2016|                    |                    |
|NCT02751957|     Duke University|           Completed|Jul 2016|  Jul 2020|      Interventional|  Apr 2016|Autistic Disorder...|                    |
|NCT02758483|Universidade Fede...|           Completed|Mar 2017|  Jan 2018|      Interventional|  Apr 2016|   Diabetes Mellitus|                    |
|NCT02759848|Istanbul Medeniye...|           Completed|Jan 2012|  Dec 2014|       Observational|  May 2016|Tuberculosis,Lung...|                    |
|NCT02758860|University of Rom...|Active, not recru...|Jun 2016|  Sep 2020|Observational [Pa...|  Apr 2016|Diverticular Dise...|                    |
|NCT02757209|Consorzio Futuro ...|           Completed|Apr 2016|  Jan 2018|      Interventional|  Apr 2016|              Asthma|Fluticasone,Xhanc...|
|NCT02752438|    Ankara University|      Unknown status|May 2016|  Jul 2017|Observational [Pa...|  Apr 2016|      Hypoventilation|                    |
|NCT02753543|     Ruijin Hospital|      Unknown status|Nov 2015|  Nov 2019|      Interventional|  Apr 2016|            Lymphoma|                    |
|NCT02757508|Washington Univer...|           Completed|Mar 2016|  Jul 2017|      Interventional|  Apr 2016|                    |            Vitamins|
|NCT02753530|           Orphazyme|           Completed|Aug 2017|  Jan 2021|      Interventional|  Apr 2016|            Myositis|                    |
|NCT02754817|     Novo Nordisk A/S|           Completed|Apr 2016|  Oct 2016|       Observational|  Apr 2016|   Diabetes Mellitus|Liraglutide,Xultophy|
|NCT02759276|Daniel Alexandre ...|           Completed|May 2015|  Dec 2015|       Observational|  Apr 2016|        Hypertension|                    |
|NCT02750956|Bulent Ecevit Uni...|           Completed|Jun 2015|  Mar 2016|       Observational|  Apr 2016|Periodontal Diseases|                    |
|NCT02752113|Institut für Phar...|           Completed|Apr 2016|  May 2019|      Interventional|  Apr 2016|   Diabetes Mellitus|Metformin,Empagli...|
|NCT02752698|The Third Xiangya...|Active, not recru...|Jan 2015|  Dec 2021|      Interventional|  Jun 2015|Appendicitis,Stom...|                    |
|NCT02755779|Tel Aviv Medical ...|      Unknown status|Jun 2016|  Jun 2017|       Observational|  Apr 2016|                    |                    |
|NCT02750384|Medicines for Mal...|          Terminated|May 2016|  Jul 2016|      Interventional|  Apr 2016|                    |                    |
|NCT02754609|James Cook Univer...|           Completed|Sep 2016|  Oct 2019|      Interventional|  Apr 2016|Hookworm Infectio...|                    |
```

Command took 5.35 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/28/2024, 5:14:59 PM on gfdvdfvc

Additional Analysis 1: Top Sponsors by Enrollment (SQL):

1

In this sql querry I am going to group the data by sponsor column and sum the enrollment values and then order the results in ascending order and shows the only top 10 sponsors.

Cmd 2

SQL

```
SELECT sponsor, SUM(CAST(enrollment AS INT)) AS total_enrollment
FROM clinical_trials
GROUP BY sponsor
ORDER BY total_enrollment DESC
LIMIT 10;
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [sponsor: string, total_enrollment: long]

Table

	sponsor	total_enrollment
1	AstraZeneca	288811380
2	KU Leuven	200238787
3	Center for International Blood and Marrow Transplant Research	200192109
4	RWTH Aachen University	100266992
5	Centre d'études et d'expertise sur les risques" l'environnement" la mobilité et l'aménagement	80000000
6	University Hospital"" Caen	76483961
7	Taipei Medical University WanFang Hospital	67170653
8	ModernaTX"" Inc.	61362977
9	Assistance Publique - Hôpitaux de Paris	43892196
10	University of Pennsylvania	40923451

10 rows | 21.82 seconds runtime

Refreshed 7 minutes ago

Command took 21.82 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/28/2024, 5:40:24 PM on gfdvdfvc

Additional Analysis 2: Find the number of trials initiated by each collaborator.(using df)

Answer:-

In this code I group by the data frame by the collaborator column and count the occurrence of each unique collaborator in descending order and limit the output to 10 to see only top 10 trial by collaborator.

Cmd 2

```python
from pyspark.sql.functions import desc

collaborator_count = df.groupBy("Collaborators").count().orderBy(desc("count")).limit(10)
collaborator_count.show()
```

(2) Spark Jobs

collaborator_count: pyspark.sql.dataframe.DataFrame = [Collaborators: string, count: long]

```
+--------------------+------+
|       Collaborators| count|
+--------------------+------+
|                    |325505|
|National Cancer I...|  7918|
|National Heart" L...|  1689|
|National Institut...|  1590|
|National Institut...|  1394|
|Merck Sharp & Doh...|  1334|
|National Institut...|  1119|
|              Pfizer|   940|
|National Institut...|   872|
|     GlaxoSmithKline|   805|
+--------------------+------+
```

Command took 20.88 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/29/2024, 5:02:03 AM on dscdsxc

Additional Analysis 3: Find the number of clinical trials per year of start date (using RDD)

Answer:-

In this code I calculate the number of clinical trials per year based on the start date. I maps each trial's start year with a value of 1, then aggregates the counts for each year using a reduce operation. Finally, I collects and returns the result as a list of tuples, where each tuple represents a year and the count of trials starting in that year.

Cmd 2

```python
# additional analysis Find the number of clinical trials per year of start date
trials_per_year_rdd = data_rdd.map(lambda row: (row[12].split("-")[0], 1)).reduceByKey(lambda a, b: a + b)
trials_per_year_rdd.collect()
```

(1) Spark Jobs

```
Out[5]: [('2023', 33676),
 ('2011', 18321),
 ('2006', 11314),
 ('', 5146),
 ('2022', 34460),
 ('2008', 14924),
 ('1999', 1730),
 ('2001', 2902),
 ('1996', 647),
 ('1984', 44),
 ('1993', 304),
 ('1992', 224),
 ('1979', 16),
 ('2026', 29),
 ('1977', 24),
 ('1988', 95),
 ('1987', 63),
 ('1980', 20),
 ('1976', 19),
 ('2029', 1),
 ('1963', 1),
```

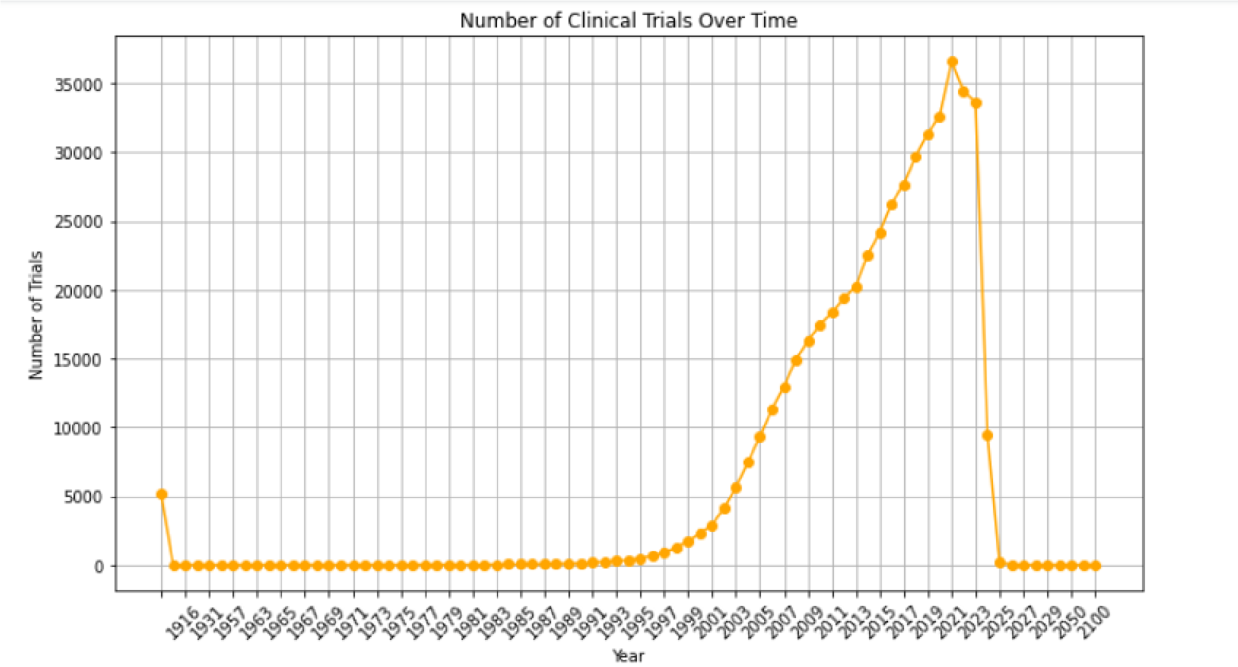
Command took 10.34 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/29/2024, 5:08:33 AM on dscdsxc

Additional analysis:- Creation of additional visualizations presenting useful information based on your own exploration

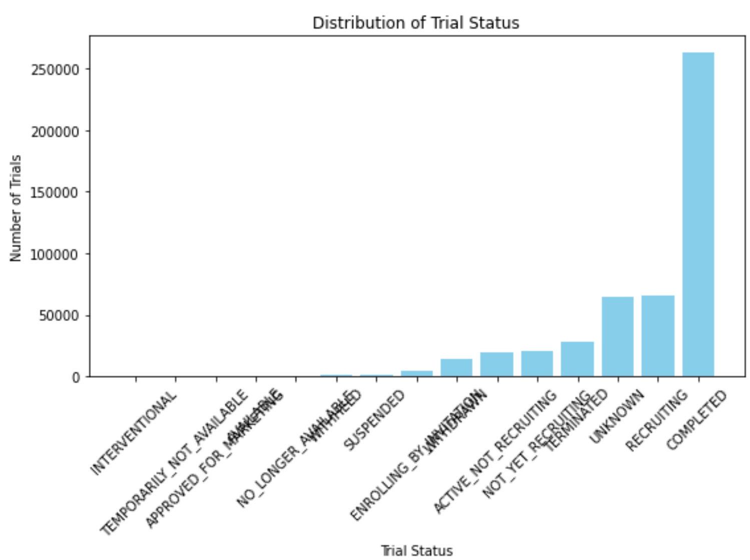
Answer:-

Here are some of my additional visualizations based on my own exploration

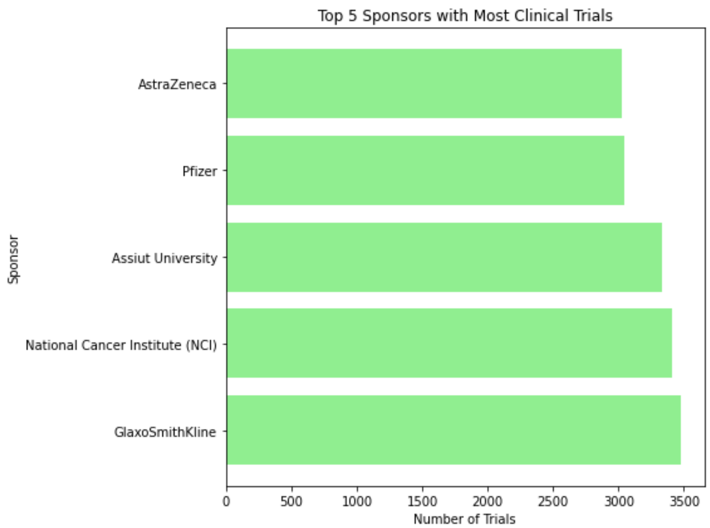
1

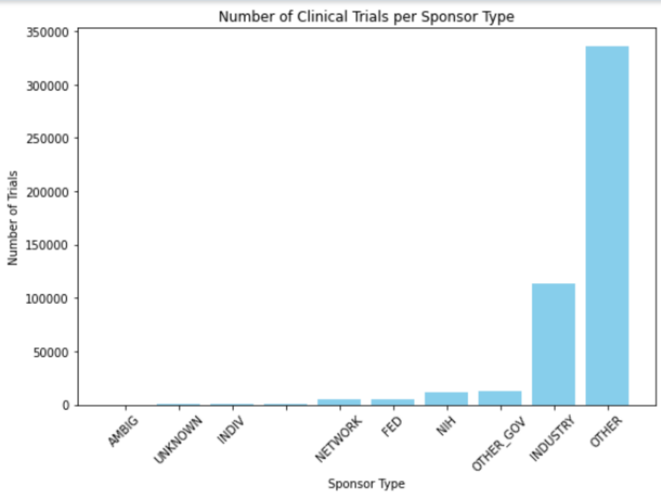



```
Command took 0.64 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/29/2024, 5:39:16 AM on dscdsxc
```



1





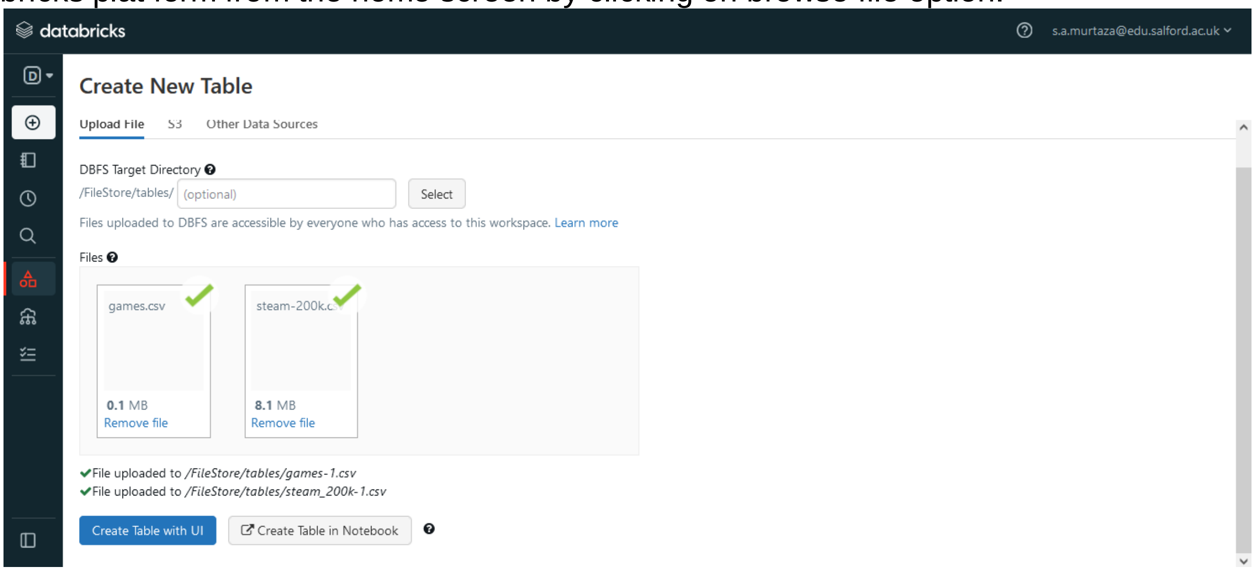
Command took 34.95 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/29/2024, 6:41:09 AM on dscdsxc

1

Assignment Part # 02

1. **Description of the necessary configuration or prerequisites needed to successfully accomplish this task:-**

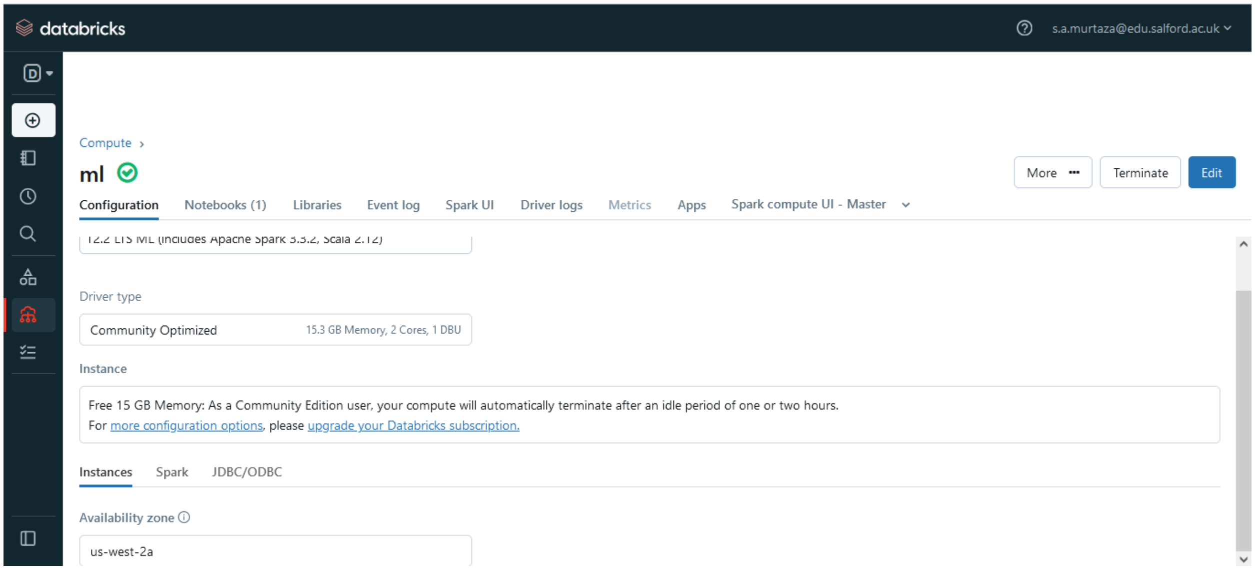
I logged into data bricks community edition account. Then I download the data files which were given for the analysis in this assignment and upload them into data bricks plat form from the home screen by clicking on browse file option.



After uploading the files, I transitioned from the Data Science & Engineering role to the Machine Learning role.

Then I create a cluster using ml runtime by clicking on create compute button.

1



After creating the cluster, I click on "create" and select "Notebook". In this way a python notebook appears to me. I name this notebook and set the default language as python.

2. Preparing data for ingestion into a Spark DataFrame, including any preliminary examination or visualization performed to understand the data structure and characteristics before initiating model training.

Then I use Ls command to check whether the file is uploaded successfully or not

Cmd 1

Python

```
%fs
ls /FileStore/tables/steam_200k.csv
```

Table

	path	name	size	modificationTime
1	dbfs:/FileStore/tables/steam_200k.csv	steam_200k.csv	8059447	1713225488000

1 row | 20.67 seconds runtime   Refreshed 1 hour ago

Command took 20.67 seconds -- by s.a.murtaza@edu.salford.ac.uk at 5/1/2024, 5:05:53 AM on fxdcxd

1

From above we can see that the required file is uploaded successfully and is ready to use for analysis purposes.

Now I am going to import the ML flow library and enable autologging.

```
import mlflow
mlflow.pyspark.ml.autolog()
```

Command took 0.17 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 6:13:35 AM on ml

When I looked into the file I feel its better to define my own schema with my own concept and do further analysis. So I defined schema using StructType and StructFeild. StructFeild is used to define the column names of the data frame while StructType is used to define the data type of the column. I define “member ID” as double, “name of game” as string, “member behaviour” as string and “Status” as double type. I read the data file into spark dataframe using spark.read.csv()

Cmd 9

```
from pyspark.sql.types import *
inputPath = "/FileStore/tables/steam_200k.csv"
myschema = StructType([
    StructField("member Id", DoubleType(), True),
    StructField("Name of the Game", StringType(), True),
    StructField("member behaviour", StringType(), True),
    StructField("Status", DoubleType(), True),
])
# Read the data as a DataFrame with the specified schema
OccupancyDF = (
    spark.read
    .option("header", True)
    .schema(myschema)
    .csv(inputPath)
)

display(OccupancyDF)

```

(1) Spark Jobs

(1) Spark Jobs

OccupancyDF: pyspark.sql.dataframe.DataFrame
- member Id: double
- Name of the Game: string
- member behaviour: string
- Status: double

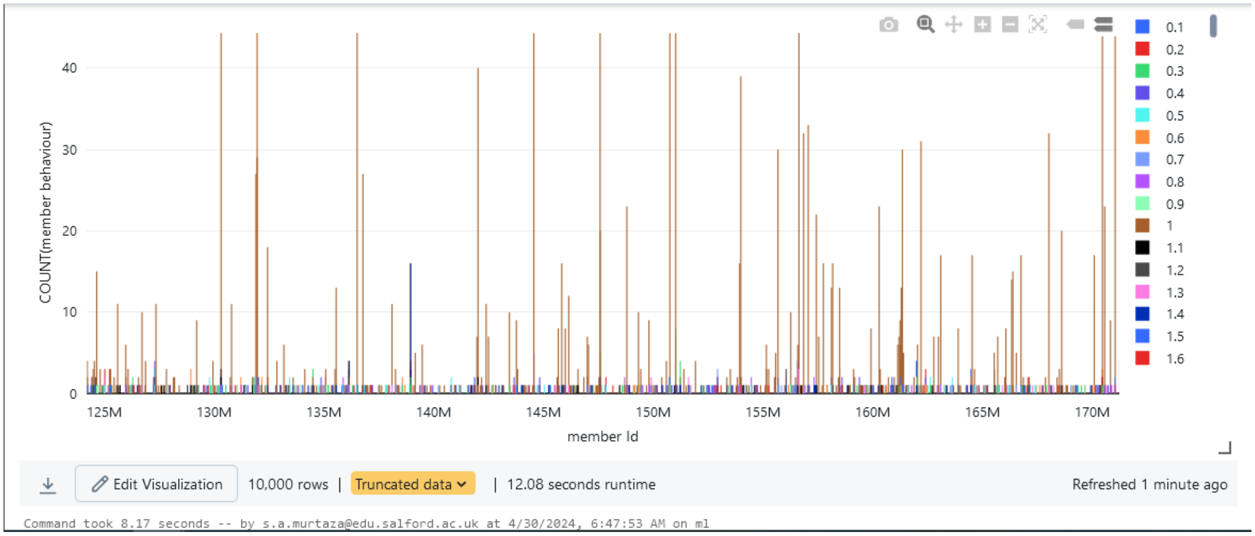
Table | Visualization 1

	member Id	Name of the Game	member behaviour	Status
1	151603712	The Elder Scrolls V Skyrim	play	273
2	151603712	Fallout 4	purchase	1
3	151603712	Fallout 4	play	87
4	151603712	Spore	purchase	1
5	151603712	Spore	play	14.9
6	151603712	Fallout New Vegas	purchase	1
7	151603712	Fallout New Vegas	play	12.1

10,000 rows | Truncated data | 1.28 seconds runtime — Refreshed 30 minutes ago

Command took 1.28 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 6:13:35 AM on ml

For better understanding of data I create visualization by setting member ID on x-axis and member behavior on y-axises and group them by status.



3. Data preprocessing conducted to prepare the data for model training.

For the training of data I have to apply ALS matrix factorization, so I need to have integer ID values for both users and items but the dataset does not contain the ID for the games. So first of all I will create a string Indexer to generate integer IDs for the game names and then fit that string indexer to the data and transform the data frame

Cmd 4

Python

```
from pyspark.ml.feature import StringIndexer

# Create a StringIndexer to generate integer IDs for the game names
indexer = StringIndexer(inputCol="Name of the Game", outputCol="gameId")

# Fit the StringIndexer to the data and transform the DataFrame
indexed_df = indexer.fit(OccupancyDF).transform(OccupancyDF)

# Show the DataFrame with the generated integer IDs
indexed_df.show()

```

▸ (3) Spark Jobs

▾ (1) MLflow run

Logged 1 run to an experiment in MLflow. Learn more

1

```
+---------+--------------------+----------------+------+------+
|member Id|    Name of the Game|member behaviour|Status|gameId|
+---------+--------------------+----------------+------+------+
|151603712|The Elder Scrolls...|            play| 273.0|   8.0|
|151603712|           Fallout 4|        purchase|   1.0| 100.0|
|151603712|           Fallout 4|            play|  87.0| 100.0|
|151603712|               Spore|        purchase|   1.0| 332.0|
|151603712|               Spore|            play|  14.9| 332.0|
|151603712|   Fallout New Vegas|        purchase|   1.0|  29.0|
|151603712|   Fallout New Vegas|            play|  12.1|  29.0|
|151603712|       Left 4 Dead 2|        purchase|   1.0|   4.0|
|151603712|       Left 4 Dead 2|            play|   8.9|   4.0|
|151603712|            HuniePop|        purchase|   1.0| 867.0|
|151603712|            HuniePop|            play|   8.5| 867.0|
|151603712|       Path of Exile|        purchase|   1.0|  39.0|
|151603712|       Path of Exile|            play|   8.1|  39.0|
|151603712|         Poly Bridge|        purchase|   1.0|1347.0|
|151603712|         Poly Bridge|            play|   7.5|1347.0|
|151603712|         Left 4 Dead|        purchase|   1.0|  49.0|
|151603712|         Left 4 Dead|            play|   3.3|  49.0|
|151603712|     Team Fortress 2|        purchase|   1.0|   1.0|
Command took 6.82 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 6:13:35 AM on ml
```

Now my next step is data pre-processing. Data preprocessing mean the data should be in a correct format to train an MLlib model

So particularly in task like recommendation system when we are trying to predict a numeric values we ensures that the model treats all the ratings equally. In this system I considered to include both behavior “play” and “purchase” collectively in the single column name as status. But there is too much variation in status column from 1 to thousands. So I am going to apply normalization on the status column using the formula.

indexed_df = (col("Status") - min_status) / (max_status - min_status))

This formula will ensure that minimum status maps to Zero and maximum status maps to 1.
But if we do not apply normalization to status column than model does not treat all the ratings equally and there would be large value of RMSE(root mean square error) which is not acceptable in our case. So it is important to apply normalization.

So in this code I calculated the minimum value and maximum value of the status column and then apply the normalization formula

```
Cmd 7

from pyspark.sql.functions import col, min as pyspark_min, max as pyspark_max
from pyspark.ml.feature import StringIndexer, MinMaxScaler
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator

# Calculate the minimum and maximum values of the "Status" column
min_status = indexed_df.select(pyspark_min("Status")).first()[0]
max_status = indexed_df.select(pyspark_max("Status")).first()[0]

# Normalize the "Status" column to a range between 0 and 1
indexed_df = indexed_df.withColumn("Normalized_Status", (col("Status") - min_status) / (max_status - min_status))

▸ (4) Spark Jobs
Command took 8.38 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 8:01:54 AM on ml
```

Next, I am going to divide the data into two sets: a training set and a test set because it is crucial for supervised machine learning, where we reserve a portion of the data for evaluating the model's performance. So I split the data randomly, allocating 80% for training and 20% for testing. To ensure reproducibility, I set the seed to 100, ensuring consistent results each time the code is run.

Cmd 8

Python

```
# Split the data into training and test sets
(training, test) = indexed_df.randomSplit([0.8, 0.2], seed=100)
```

Command took 0.59 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 8:07:29 AM on ml

4. Identifying optimal hyperparameters, training the model, evaluating its performance, and utilizing MLflow for experiment tracking

I applied ALS matrix factorization to this recommender system. I used the following parameters

maxIter=10 (maximum number of iterations)
regParam=0.01(for controlling regularization strenght)
userCol: for identifying user
itemCol: for identifying items
ratingCol: for ratings associated with user-item interaction

Cmd 8

Python

```
# Define the ALS model with the required parameters
als = ALS(maxIter=10, regParam=0.01, userCol="member Id", itemCol="gameId", ratingCol="Normalized_Status",
          coldStartStrategy="drop")
```

Command took 0.11 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 10:57:31 AM on dsx

Then I fit the ALS matrix factorization to the training dataset

Cmd 9

Python

```
# Fit the ALS model to the training data
model = als.fit(training)
```

(7) Spark Jobs

(1) MLflow run

Logged 1 run to an experiment in MLflow. Learn more

```
2024/04/30 17:57:50 INFO mlflow.utils.autologging_utils: Created MLflow autologging run with ID '8fb8faa73e874a27914191189355ac88', which will track hyperparameters, performance metrics, model artifacts, and lineage information for the current pyspark.ml workflow
2024/04/30 17:58:18 WARNING mlflow.pyspark.ml: Model ALS_089fa6688ad1 will not be autologged because it is not allowlisted or or because one or more of its nested models are not allowlisted. Call mlflow.spark.log_model() to explicitly log the model, or specify a custom allowlist via the spark.mlflow.pysparkml.autolog.logModelAllowlistFile Spark conf (see mlflow.pyspark.ml.autolog docs for more info).
```

Command took 29.59 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 10:57:49 AM on dsx

1

Then I generate prediction on the test data using the already trained model named as "model" which applies the model to the test dataset and adds a new column named as "Prediction" to the data frame containing the predicted values.

Then I initiate a RegressionEvaluator with the following parameters
metricName="rmse"
labelCol="Normalized_Status"
predictionCol="prediction"
Then I use that regression evaluator to calculate the RMSE value between Normalized_Status values and the predicted values

```python
predictions = model.transform(test)
evaluator = RegressionEvaluator(metricName="rmse", labelCol="Normalized_Status", predictionCol="prediction")
rmse = evaluator.evaluate(predictions)
print("Root Mean Squared Error (RMSE) on test data after normalization = {:.2f}".format(rmse))

# Generate predictions using the trained model
predictions.show()
```

```
(5) Spark Jobs
predictions: pyspark.sql.dataframe.DataFrame = [member Id: integer, Name of the Game: string ... 5 more fields]
2024/04/30 18:00:18 WARNING mlflow.utils.autologging_utils: MLflow autologging encountered a warning: "/databricks/python/lib/python3.10/site-packages/mlflow/data/spark_dataset.py:159: UserWarning: Hint: Inferred schema contains integer column(s). Integer columns in Python cannot represent missing values. If your input data contains missing values at inference time, it will be encoded as floats and will cause a schema enforcement error. The best way to avoid this problem is to infer the model schema based on a realistic data sample (training dataset) that includes missing values. Alternatively, you can declare integer columns as doubles (float64) whenever these columns may have missing values. See `Handling Integers With Missing Values <https://www.mlflow.org/docs/latest/models.html#handling-integers-with-missing-values>`_ for more details."

Root Mean Squared Error (RMSE) on test data after normalization = 0.01
```

```
ains missing values at inference time, it will be encoded as floats and will cause a schema enforcement error. The best way to avoid this problem is to infer the model schema based on a realistic data sample (training dataset) that includes missing values. Alternatively, you can declare integer columns as doubles (float64) whenever these columns may have missing values. See `Handling Integers With Missing Values <https://www.mlflow.org/docs/latest/models.html#handling-integers-with-missing-values>`_ for more details."

Root Mean Squared Error (RMSE) on test data after normalization = 0.01
+---------+--------------------+----------------+------+------+--------------------+------------+
|member Id|    Name of the Game|member behaviour|Status|gameId|   Normalized_Status|  prediction|
+---------+--------------------+----------------+------+------+--------------------+------------+
|     5250|     Cities Skylines|        purchase|   1.0|   158|7.657032984796535E-5|1.0907977E-8|
|     5250|              Dota 2|            play|   0.2|     0|8.507814427551706E-6| 7.487604E-6|
|     5250|Half-Life 2 Death...|        purchase|   1.0|    13|7.657032984796535E-5|1.3883077E-9|
|     5250|              Portal|        purchase|   1.0|    14|7.657032984796535E-5|2.1084414E-9|
|     5250|     Team Fortress 2|        purchase|   1.0|     1|7.657032984796535E-5|5.0977206E-8|
|    76767|         Alien Swarm|            play|   0.8|    32|5.955470099286195E-5|1.6275358E-7|
|    76767|            Banished|            play|  24.0|   214|0.002033367648184...|1.3941788E-7|
|    76767|            Banished|        purchase|   1.0|   214|7.657032984796535E-5|1.3941788E-7|
|    76767|Call of Duty Blac...|        purchase|   1.0|    50|7.657032984796535E-5|2.3781568E-7|
|    76767|Call of Duty Blac...|            play|  12.5|    57|0.001054968989016...|1.4741174E-6|
|    76767|Call of Duty Mode...|            play|  65.0|    25|0.005521571563481058| 4.414076E-7|
|    76767|Call of Duty Mode...|        purchase|   1.0|    25|7.657032984796535E-5| 4.414076E-7|
|    76767|Counter-Strike Gl...|            play|   3.5|     2| 2.89265690536758E-4|2.9329345E-5|
|    76767|           Half-Life|            play|   1.2|    45|9.358595870306876E-5|  6.80719E-8|
|    76767|           Half-Life|        purchase|   1.0|    45|7.657032984796535E-5|  6.80719E-8|
|    76767|Rise of Nations E...|            play|   5.7|   691|4.764376079428956E-4| 5.274042E-8|
|    76767| Thief - Opportunist|        purchase|   1.0|   994|7.657032984796535E-5|2.9432597E-8|
Command took 29.99 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 11:00:05 AM on dsx
```

Result:-

From above we can see that our RMSE value=0.01 which mean that the predicted values are very close to the actual values that's good.

Hyperparameter tunning:-

1

Then I create a parameter grid for hyperparameter tuning. I define different hyperparameters as maximum iterations, Rank, and regularization parameter. Then I uses ParamGridBuilder to create a grid containing combinations of these hyperparameters.

Cmd 11

Python

```
from pyspark.ml.tuning import ParamGridBuilder

# Define the values for each parameter
maxIter_values = [5, 10, 15]
rank_values = [5, 10, 15]
regParam_values = [0.01, 0.1, 0.2]

# Create a parameter grid builder
paramGrid = (ParamGridBuilder()
             .addGrid(als.maxIter, maxIter_values)
             .addGrid(als.rank, rank_values)
             .addGrid(als.regParam, regParam_values)
             .build())

```

Command took 0.10 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 11:15:48 AM on dsx

Then I perform hyperparameter tuning using tvs. I configure TVS with an ALS estimator, a parameter grid, a evaluator for model performance and a specific train-validation split ratio of 80:20

Cmd 12

```
from pyspark.ml.tuning import TrainValidationSplit

# Instantiate TrainValidationSplit
tvs = TrainValidationSplit(estimator=als,
                            estimatorParamMaps=paramGrid,
                            evaluator=evaluator,
                            # 80% of the data will be used for training, 20% for validation
                            trainRatio=0.8)

```

Command took 0.06 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 11:15:52 AM on dsx

Then I train the model using train validation split method.

Cmd 13

Python

```
# Train the model using grid search
model =tvs.fit(training)
```

(5) Spark Jobs

(28) MLflow runs

Logged 28 runs to an experiment in MLflow. Learn more

```
2024/04/30 18:15:54 INFO mlflow.utils.autologging_utils: Created MLflow autologging run with ID '2df3d785fc184fad9019ac871e156c93', which will track hyperparameters, performance metrics, model artifacts, and lineage information for the current pyspark.ml workflow
2024/04/30 18:32:18 WARNING mlflow.pyspark.ml: Model TrainValidationSplitModel_a00c04b16ce1 will not be autologged because it is not allowlisted or or because one or more of its nested models are not allowlisted. Call mlflow.spark.log_model() to explicitly log the model, or specify a custom allowlist via the spark.mlflow.pysparkml.autolog.logModelAllowlistFile Spark conf (see mlflow.pyspark.ml.autolog docs for more info).
```

Command took 16.42 minutes -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 11:15:54 AM on dsx

Best model:-

Then I retrieve the best model and access its parameters using getter method.

1

Cmd 13

Python

```python

# Get the best model from TrainValidationSplit using its getter method
best_model = model.getEstimator()


# Explore the public API of the best_model (usually ALS) to access parameters
print("Parameters for Best Model:")

# Example assuming ALS model:
print("maxIter:", best_model.getMaxIter())
print("rank:", best_model.getRank())
print("regParam:", best_model.getRegParam())
print("RMSE:", rmse)
```

```
Parameters for Best Model:
maxIter: 10
rank: 10
regParam: 0.01
RMSE: 0.010672728321225879
```

Command took 0.11 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 3:39:49 PM on fdc

Multiple Runs:-

Then I uses multiple values of hyperparameters to train the model and check which parameter best fit with the model with lowest RMSE value.

Second run:-

In my 2nd run I defined the hyperparameters as

maxIter_values = [5, 10, 20]
rank_values = [4, 10, 18]
regParam_values = [0.01, 0.001, 0.2]

and then train them by the procedure I explained earlier but here I have not use getter method for getting best model instead I use bestmodel attribute for getting best model and the rest procedure is same as above.

Cmd 14

Python

```python
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit
from pyspark.ml.evaluation import RegressionEvaluator
# Define the values for each parameter
maxIter_values = [5, 10, 20]
rank_values = [4, 10, 18]
regParam_values = [0.01, 0.001, 0.2]
# Create a parameter grid builder
paramGrid = ParamGridBuilder() \
    .addGrid(als.maxIter, maxIter_values) \
    .addGrid(als.rank, rank_values) \
    .addGrid(als.regParam, regParam_values) \
    .build()
```

Command took 0.12 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 4:52:32 PM on fdc

Cmd 15

Python

```python

# Define the evaluator
evaluator = RegressionEvaluator(metricName="rmse", labelCol="Normalized_Status", predictionCol="prediction")
# Create TrainValidationSplit
tvs = TrainValidationSplit(estimator=als,
                           estimatorParamMaps=paramGrid,
                           evaluator=evaluator,
                           trainRatio=0.8)
# Fit TrainValidationSplit to the training data
model = tvs.fit(training)
# Get the best model from TrainValidationSplit
best_model = model.bestModel
# Make predictions on the test data
predictions = best_model.transform(test)
# Evaluate the predictions using the evaluation metric
rmse = evaluator.evaluate(predictions)
print("Best Model RMSE:", rmse)

```

```
Best Model RMSE: 0.010672728321225879
```

Command took 0.08 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 3:44:33 PM on fdc

Result:-

From this result we can see that we got RMSE value of 0.01 from the best hyperparameters which are

Maxiter=10

Rank=10

regParam=0.01

Cmd 16

Python

```python
# Print the hyperparameters and RMSE
print("Parameters for Best Model:")
print("maxIter:", best_model._java_obj.getMaxIter())
print("rank:", best_model._java_obj.getRank())
print("regParam:", best_model._java_obj.getRegParam())
print("Best Model RMSE:", rmse)

```

```
Parameters for Best Model:
maxIter: 10
rank: 10
regParam: 0.01
Best Model RMSE: 0.010672728321225879
```

Command took 0.08 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 3:45:25 PM on fdc

Third Run:-

In my 3rd run I changed the hyperparameters and defined the hyper parameters as

maxIter_values = [10, 20, 30]

rank_values = [8, 13, 20]

regParam_values = [0.1, 0.001, 0.2]

and then train these hyper parameters on the model using the above procedure as in Second run.

Cmd 17

```python
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit
from pyspark.ml.evaluation import RegressionEvaluator

# Define the values for each parameter
maxIter_values = [10, 20, 30]
rank_values = [8, 13, 20]
regParam_values = [0.1, 0.001, 0.2]

# Create a parameter grid builder
paramGrid = ParamGridBuilder() \
    .addGrid(als.maxIter, maxIter_values) \
    .addGrid(als.rank, rank_values) \
    .addGrid(als.regParam, regParam_values) \
    .build()
```

Command took 0.08 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 4:57:02 PM on fdc

```python
# Define the evaluator
evaluator = RegressionEvaluator(metricName="rmse", labelCol="Normalized_Status", predictionCol="prediction")
# Create TrainValidationSplit
tvs = TrainValidationSplit(estimator=als,
                           estimatorParamMaps=paramGrid,
                           evaluator=evaluator,
                           trainRatio=0.8)
# Fit TrainValidationSplit to the training data
model = tvs.fit(training)
# Get the best model from TrainValidationSplit
best_model = model.bestModel
# Make predictions on the test data
predictions = best_model.transform(test)
# Evaluate the predictions using the evaluation metric
rmse = evaluator.evaluate(predictions)
print("Best Model RMSE:", rmse)
```

(6) Spark Jobs

(28) MLflow runs

Logged 28 runs to an experiment in MLflow. Learn more

predictions: pyspark.sql.dataframe.DataFrame = [member Id: integer, Name of the Game: string ... 5 more fields]

```
2024/04/30 22:45:57 INFO mlflow.utils.autologging_utils: Created MLflow autologging run with ID 'd16ef740bbae4de6a545a27699353b62', which will track hyperparameters, performance metrics, model artifacts, and lineage information for the current pyspark.ml workflow
2024/04/30 23:39:36 WARNING mlflow.pyspark.ml: Model TrainValidationSplitModel_6b38c5837902 will not be autologged because it is not allowlisted or or because one or more of its nested models are not allowlisted. Call mlflow.spark.log_model() to explicitly log the model, or specify a custom allowlist via the spark.mlflow.pysparkml.autolog.logModelAllowlistFile Spark conf (see mlflow.pyspark.ml.autolog docs for more info).
Best Model RMSE: 0.010672728321225879
```

Command took 53.80 minutes -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 3:45:56 PM on fdc

Result:-

Here from the result we can see that we got the RMSE value of 0.0106 from the best hyperparameters which are

Maxiter=10
Rank=8
Regparam=0.1

Cmd 19

```python
# Print the parameters of the best model
print("Best Model Parameters:")
print("MaxIter:", best_model._java_obj.parent().getMaxIter())
print("Rank:", best_model._java_obj.parent().getRank())
print("RegParam:", best_model._java_obj.parent().getRegParam())
print("Best Model RMSE:", rmse)
```

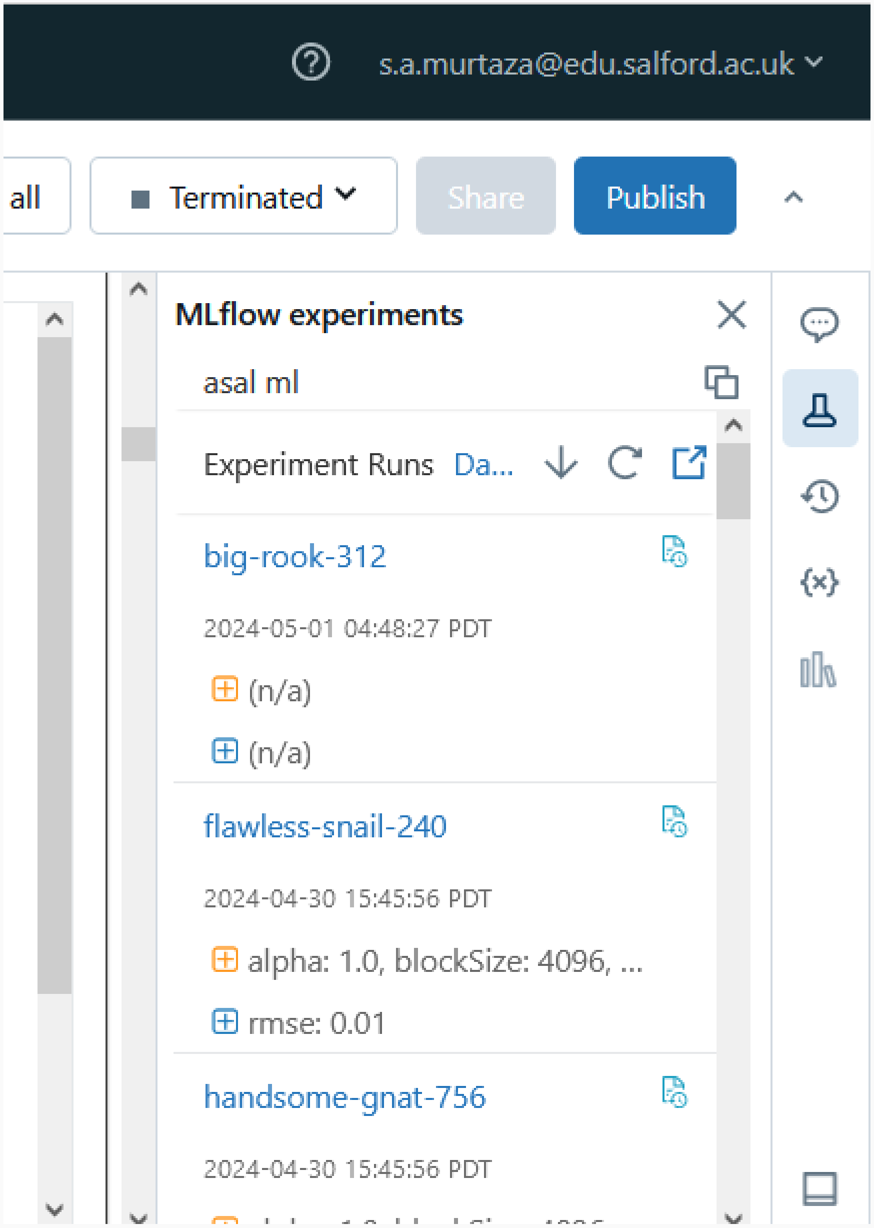
```
Best Model Parameters:
MaxIter: 10
Rank: 8
RegParam: 0.1
Best Model RMSE: 0.010672728321225879
```

Command took 0.06 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 4:47:57 PM on fdc

Using ML flow:-

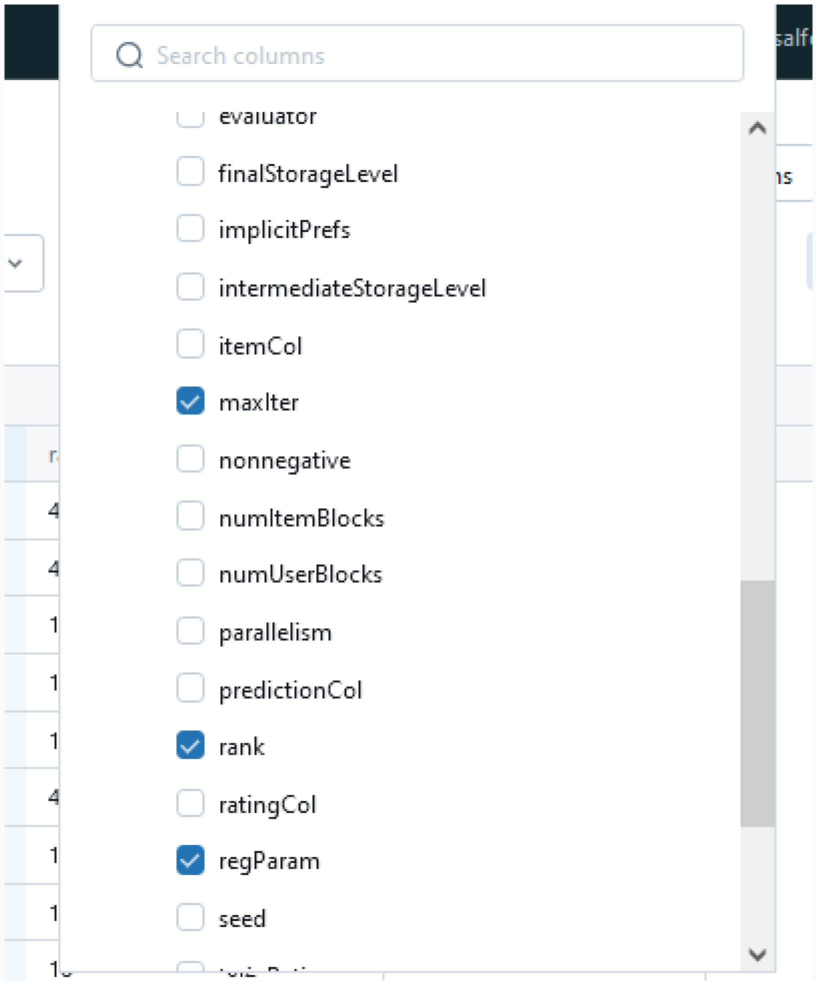
1

Now as we performed multiple runs so I am going to use the experiment UI to view more information on all the runs that I performed earlier as part of the grid search. So for this purpose I clicked on the torch icon and select experiment.



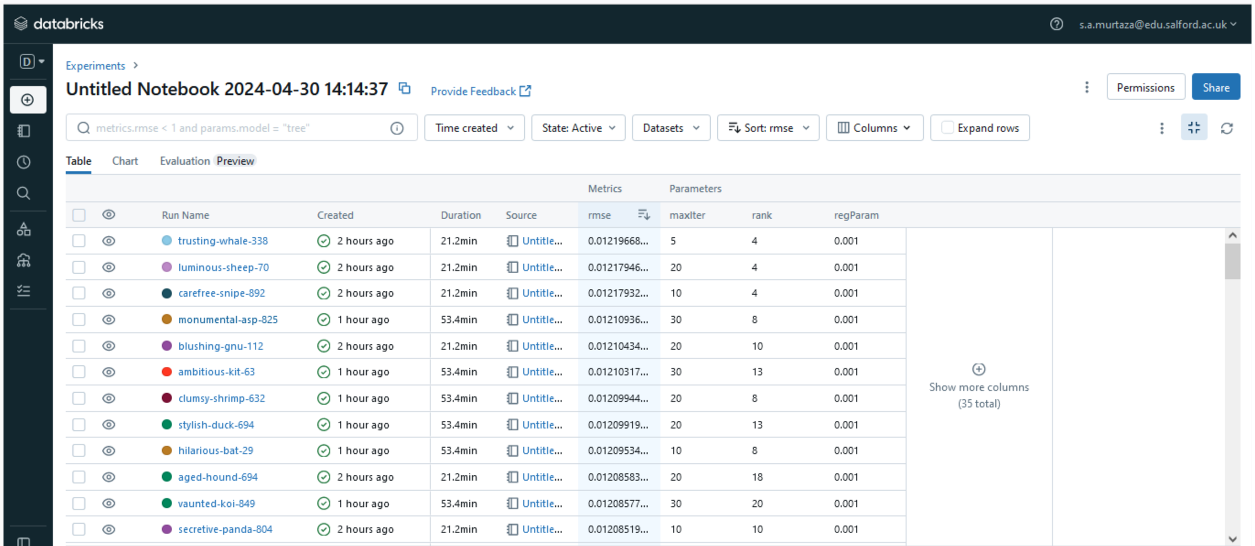
I select all the hyperparameters by clicking on the column button.

1

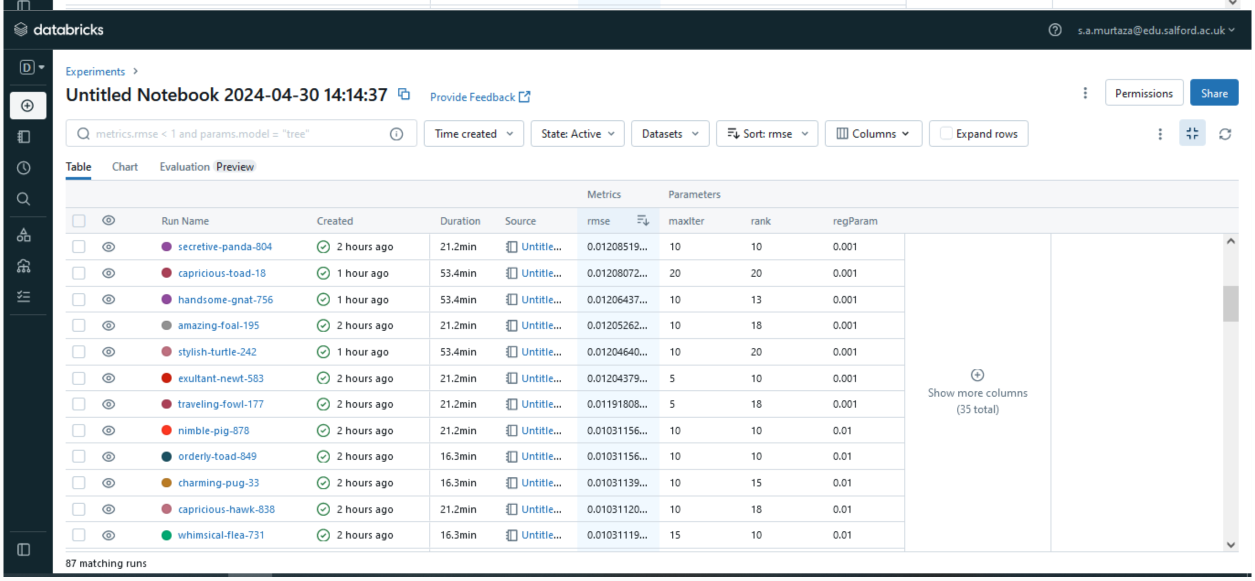


Here you can see that I have performed 87 runs and result of each run is displayed. I clicked on RMSE column to sort the result in ascending order.

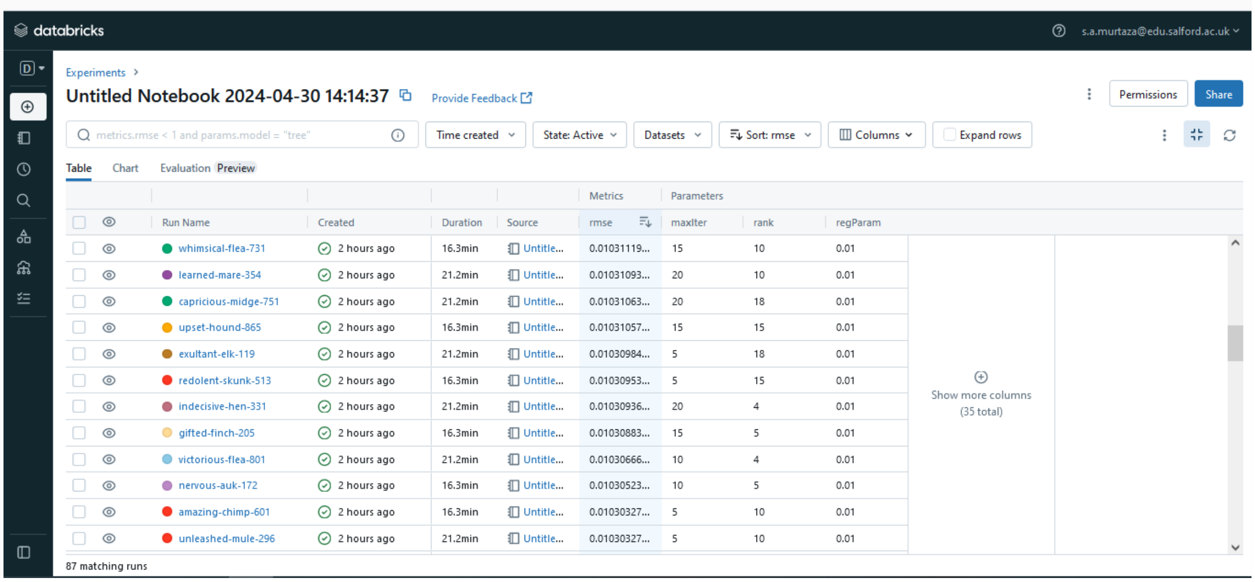
1



Run Name	Created	Duration	Source	rmse	maxIter	rank	regParam
trusting-whale-338	2 hours ago	21.2min	Untitle...	0.01219668...	5	4	0.001
luminous-sheep-70	2 hours ago	21.2min	Untitle...	0.01217946...	20	4	0.001
carefree-snipe-892	2 hours ago	21.2min	Untitle...	0.01217932...	10	4	0.001
monumental-asp-825	1 hour ago	53.4min	Untitle...	0.01210936...	30	8	0.001
blushing-gnu-112	2 hours ago	21.2min	Untitle...	0.01210434...	20	10	0.001
ambitious-kit-63	1 hour ago	53.4min	Untitle...	0.01210317...	30	13	0.001
clumsy-shrimp-632	1 hour ago	53.4min	Untitle...	0.01209944...	20	8	0.001
stylish-duck-694	1 hour ago	53.4min	Untitle...	0.01209919...	20	13	0.001
hilarious-bat-29	1 hour ago	53.4min	Untitle...	0.01209534...	10	8	0.001
aged-hound-694	2 hours ago	21.2min	Untitle...	0.01208583...	20	18	0.001
vaunted-koi-849	1 hour ago	53.4min	Untitle...	0.01208577...	30	20	0.001
secretive-panda-804	2 hours ago	21.2min	Untitle...	0.01208519...	10	10	0.001

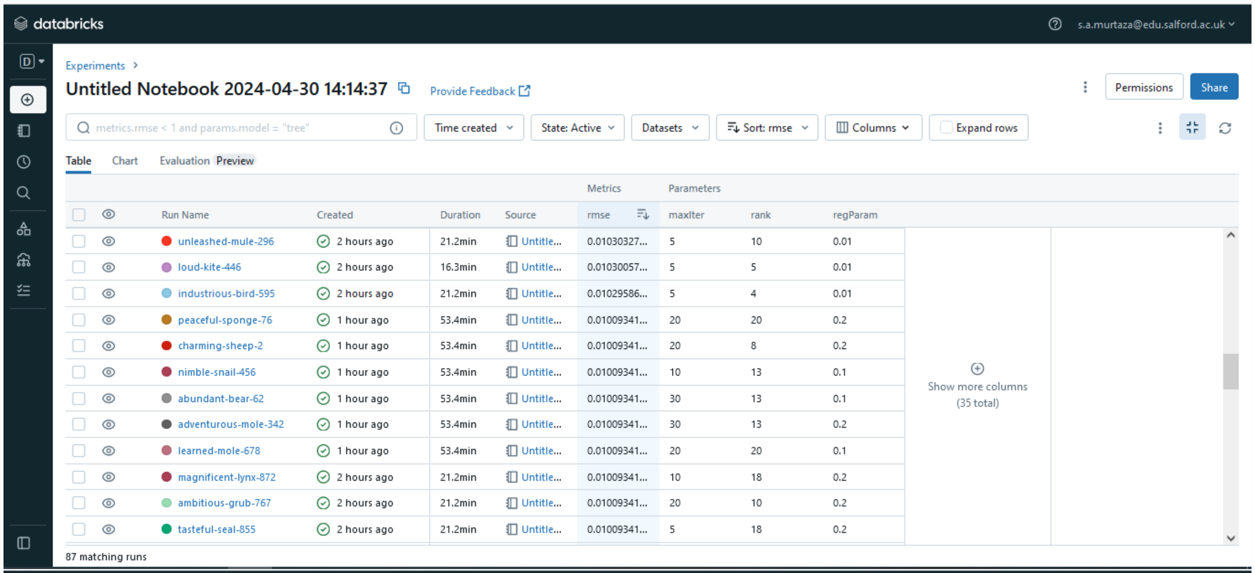


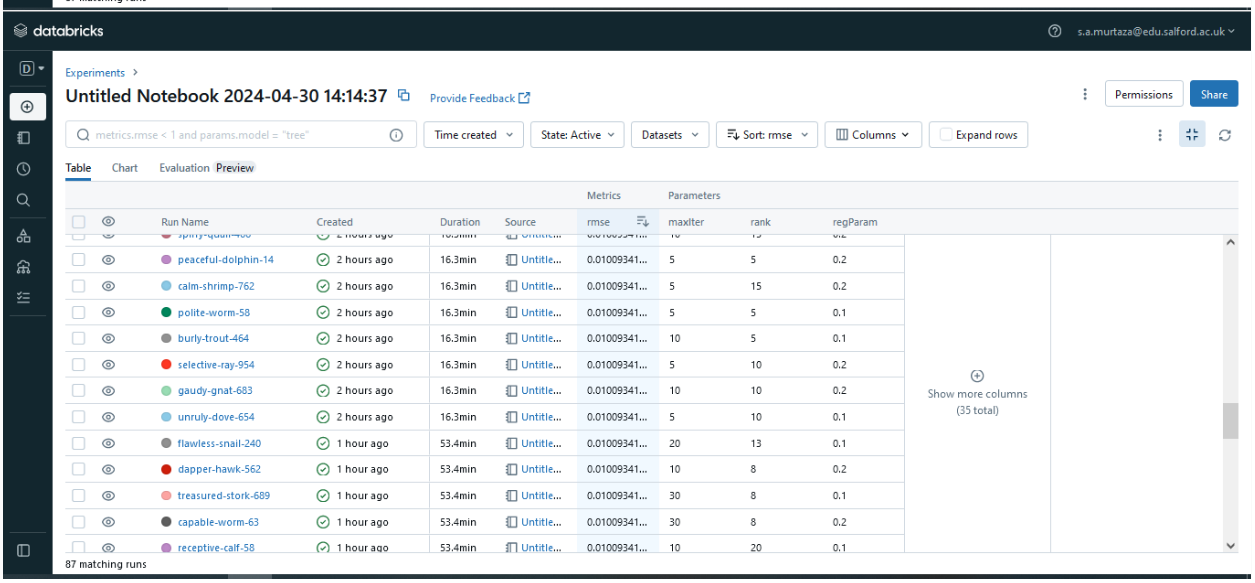
Run Name	Created	Duration	Source	rmse	maxIter	rank	regParam
secretive-panda-804	2 hours ago	21.2min	Untitle...	0.01208519...	10	10	0.001
capricious-toad-18	1 hour ago	53.4min	Untitle...	0.01208072...	20	20	0.001
handsome-gnat-756	1 hour ago	53.4min	Untitle...	0.01206437...	10	13	0.001
amazing-foal-195	2 hours ago	21.2min	Untitle...	0.01205262...	10	18	0.001
stylish-turtle-242	1 hour ago	53.4min	Untitle...	0.01204640...	10	20	0.001
exultant-newt-583	2 hours ago	21.2min	Untitle...	0.01204379...	5	10	0.001
traveling-fowl-177	2 hours ago	21.2min	Untitle...	0.01191808...	5	18	0.001
nimble-pig-878	2 hours ago	21.2min	Untitle...	0.01031156...	10	10	0.01
orderly-toad-849	2 hours ago	16.3min	Untitle...	0.01031156...	10	10	0.01
charming-pug-33	2 hours ago	16.3min	Untitle...	0.01031139...	10	15	0.01
capricious-hawk-838	2 hours ago	21.2min	Untitle...	0.01031120...	10	18	0.01
whimsical-flea-731	2 hours ago	16.3min	Untitle...	0.01031119...	15	10	0.01



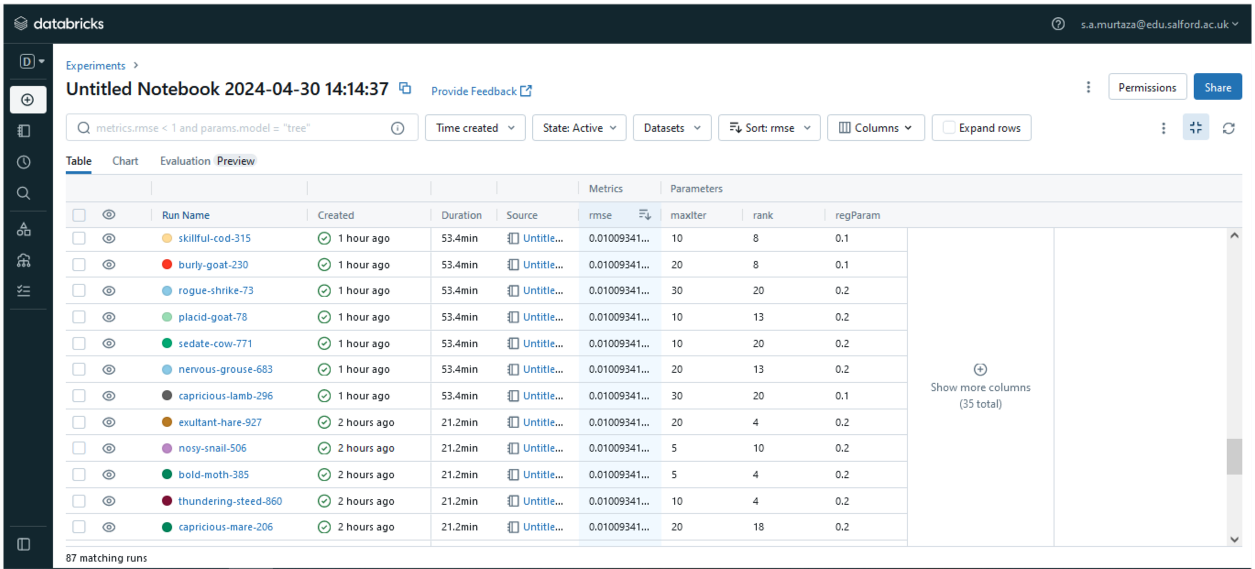
Run Name	Created	Duration	Source	rmse	maxIter	rank	regParam
whimsical-flea-731	2 hours ago	16.3min	Untitle...	0.01031119...	15	10	0.01
learned-mare-354	2 hours ago	21.2min	Untitle...	0.01031093...	20	10	0.01
capricious-midge-751	2 hours ago	21.2min	Untitle...	0.01031063...	20	18	0.01
upset-hound-865	2 hours ago	16.3min	Untitle...	0.01031057...	15	15	0.01
exultant-elk-119	2 hours ago	21.2min	Untitle...	0.01030984...	5	18	0.01
redolent-skunk-513	2 hours ago	16.3min	Untitle...	0.01030953...	5	15	0.01
indecisive-hen-331	2 hours ago	21.2min	Untitle...	0.01030936...	20	4	0.01
gifted-finch-205	2 hours ago	16.3min	Untitle...	0.01030883...	15	5	0.01
victorious-flea-801	2 hours ago	21.2min	Untitle...	0.01030666...	10	4	0.01
nervous-auk-172	2 hours ago	16.3min	Untitle...	0.01030523...	10	5	0.01
amazing-chimp-601	2 hours ago	16.3min	Untitle...	0.01030327...	5	10	0.01
unleashed-mule-296	2 hours ago	21.2min	Untitle...	0.01030327...	5	10	0.01

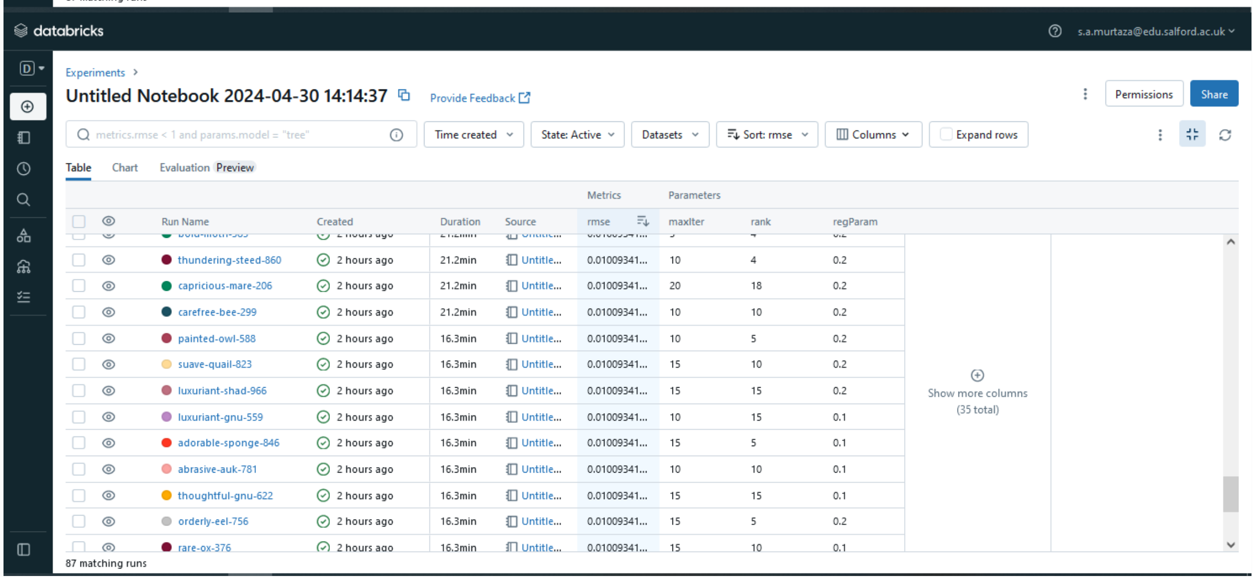
1


databricks
s.a.murtaza@edu.salford.ac.uk
Experiments
Untitled Notebook 2024-04-30 14:14:37
Provide Feedback
Permissions
Share
metrics.rmse < 1 and params.model = "tree"
Time created
State: Active
Datasets
Sort: rmse
Columns
Expand rows
Table
Chart
Evaluation Preview
Metrics
Parameters
Run Name
Created
Duration
Source
rmse
maxIter
rank
regParam
unleashed-mule-296 2 hours ago 21.2min Untitle... 0.01030327... 5 10 0.01
loud-kite-446 2 hours ago 16.3min Untitle... 0.01030057... 5 5 0.01
industrious-bird-595 2 hours ago 21.2min Untitle... 0.01029586... 5 4 0.01
peaceful-sponge-76 1 hour ago 53.4min Untitle... 0.01009341... 20 20 0.2
charming-sheep-2 1 hour ago 53.4min Untitle... 0.01009341... 20 8 0.2
nimble-snail-456 1 hour ago 53.4min Untitle... 0.01009341... 10 13 0.1
abundant-bear-62 1 hour ago 53.4min Untitle... 0.01009341... 30 13 0.1
adventurous-mole-342 1 hour ago 53.4min Untitle... 0.01009341... 30 13 0.2
learned-mole-678 1 hour ago 53.4min Untitle... 0.01009341... 20 20 0.1
magnificent-lynx-872 2 hours ago 21.2min Untitle... 0.01009341... 10 18 0.2
ambitious-grub-767 2 hours ago 21.2min Untitle... 0.01009341... 20 10 0.2
tasteful-seal-855 2 hours ago 21.2min Untitle... 0.01009341... 5 18 0.2
Show more columns
(35 total)
87 matching runs


databricks
s.a.murtaza@edu.salford.ac.uk
Experiments
Untitled Notebook 2024-04-30 14:14:37
Provide Feedback
Permissions
Share
metrics.rmse < 1 and params.model = "tree"
Time created
State: Active
Datasets
Sort: rmse
Columns
Expand rows
Table
Chart
Evaluation Preview
Metrics
Parameters
Run Name
Created
Duration
Source
rmse
maxIter
rank
regParam
peaceful-dolphin-14 2 hours ago 16.3min Untitle... 0.01009341... 5 5 0.2
calm-shrimp-762 2 hours ago 16.3min Untitle... 0.01009341... 5 15 0.2
polite-worm-58 2 hours ago 16.3min Untitle... 0.01009341... 5 5 0.1
burly-trout-464 2 hours ago 16.3min Untitle... 0.01009341... 10 5 0.1
selective-ray-954 2 hours ago 16.3min Untitle... 0.01009341... 5 10 0.2
gaudy-gnat-683 2 hours ago 16.3min Untitle... 0.01009341... 10 10 0.2
unruly-dove-654 2 hours ago 16.3min Untitle... 0.01009341... 5 10 0.1
flawless-snail-240 1 hour ago 53.4min Untitle... 0.01009341... 20 13 0.1
dapper-hawk-562 1 hour ago 53.4min Untitle... 0.01009341... 10 8 0.2
treasured-stork-689 1 hour ago 53.4min Untitle... 0.01009341... 30 8 0.1
capable-worm-63 1 hour ago 53.4min Untitle... 0.01009341... 30 8 0.2
receptive-calf-58 1 hour ago 53.4min Untitle... 0.01009341... 10 20 0.1
Show more columns
(35 total)
87 matching runs

1



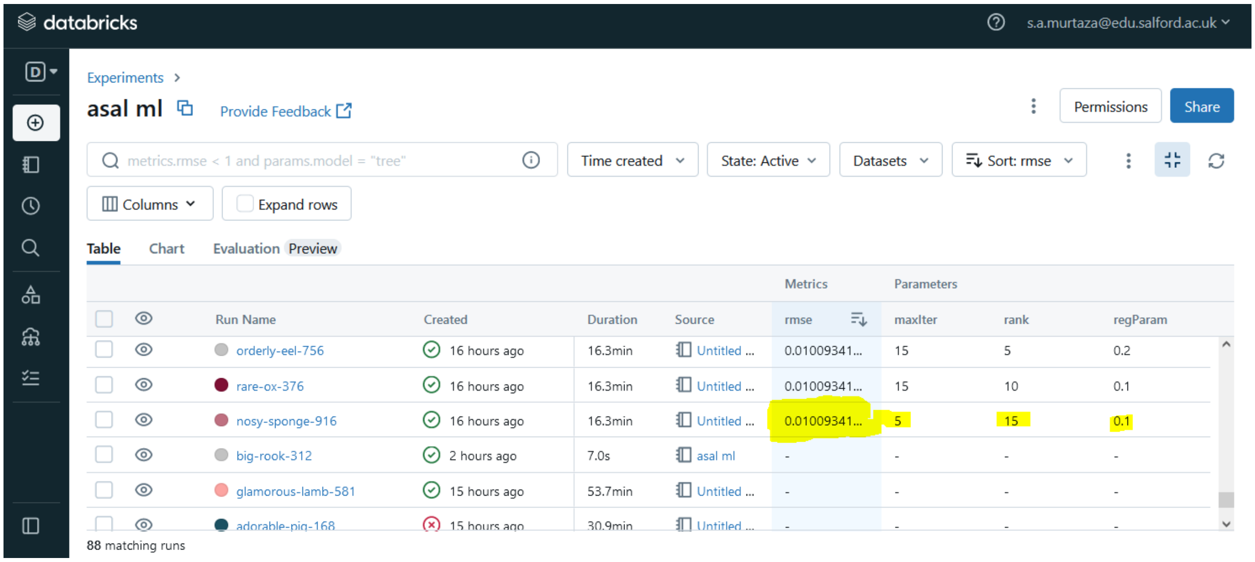


Description of results:-

I have trained the model with different hyperparameters as I described earlier. The RMSE value I got is 0.01 with a little bit difference when changing the hyperparameters value. A low RMSE value of 0.01 indicates that model is performing well and making accurate predictions.

Furthermore I got the lowest RMSE value of 0.010093417523583349 from the hyperparameters as

MaxIter=5
Rank=15
regParam=0.1



Further Analysis

For the further analysis of data I write the following codes.

1. Number of actions per status and member behavior:-

I write a code to find the number of actions per status and member behavior combination by grouping the data frame by "Status," "Member_Behavior," "Name of the Game," and "member id" column in the **OccupancyDF** and count the distinct entries

1

Cmd 21

```python


from pyspark.sql.functions import count

# Group by Status and Member_Behavior and count occurrences
action_counts = OccupancyDF.groupBy("member id","Name of the Game","Status", "member behaviour").count()

# Display the results (number of actions per status and member behavior combination)
action_counts.show()

```

▸ (2) Spark Jobs

▸ action_counts: pyspark.sql.dataframe.DataFrame = [member id: double, Name of the Game: string ... 3 more fields]

```
+------------+-------------------+------+----------------+-----+
|   member id|   Name of the Game|Status|member behaviour|count|
+------------+-------------------+------+----------------+-----+
| 5.9945701E7|The Elder Scrolls...|  58.0|            play|    1|
| 5.9945701E7|Sid Meier's Civil...|  22.0|            play|    1|
| 5.9945701E7|          L.A. Noire|  13.8|            play|    1|
| 5.9945701E7|Company of Heroes...|   1.0|        purchase|    1|
| 5.3875128E7|             Arma 2|   1.0|            play|    1|
| 5.3875128E7|         Transistor|   1.0|        purchase|    1|
| 6.2923086E7|Half-Life Blue Shift|   1.0|        purchase|    1|
```

▸ (2) Spark Jobs

▸ action_counts: pyspark.sql.dataframe.DataFrame = [member id: double, Name of the Game: string ... 3 more fields]

```
+------------+-------------------+------+----------------+-----+
|   member id|   Name of the Game|Status|member behaviour|count|
+------------+-------------------+------+----------------+-----+
| 5.9945701E7|The Elder Scrolls...|  58.0|            play|    1|
| 5.9945701E7|Sid Meier's Civil...|  22.0|            play|    1|
| 5.9945701E7|          L.A. Noire|  13.8|            play|    1|
| 5.9945701E7|Company of Heroes...|   1.0|        purchase|    1|
| 5.3875128E7|             Arma 2|   1.0|            play|    1|
| 5.3875128E7|         Transistor|   1.0|        purchase|    1|
| 6.2923086E7|Half-Life Blue Shift|   1.0|        purchase|    1|
| 6.5117175E7|      BIT.TRIP BEAT|   1.0|        purchase|    1|
| 6.5117175E7|           NightSky|   1.0|        purchase|    1|
| 1.1373749E7|The Beginner's Guide|   1.9|            play|    1|
| 1.1373749E7|               BEEP|   1.0|        purchase|    1|
| 1.1373749E7|         Samorost 2|   1.0|        purchase|    1|
| 5.6038151E7|             Crysis|   1.0|        purchase|    1|
|   9823354.0|             3DMark|   1.0|        purchase|    1|
| 1.9586574E7|Dynasty Warriors ...|  11.0|            play|    1|
|1.98346312E8|  Dragon Age Origins|   1.0|        purchase|    1|
| 2.6802825E7|Prince of Persia ...|   1.0|        purchase|    1|
| 4.2681063E7|Football Manager ...|   3.0|            play|    1|
```

Command took 4.18 seconds -- by s.a.murtaza@edu.salford.ac.uk at 5/1/2024, 4:48:48 AM on fxdcxd

2. Top played games:-

I write a code to define the number of top played games to visualize from the data frame indexed_df and plot them using bar chart

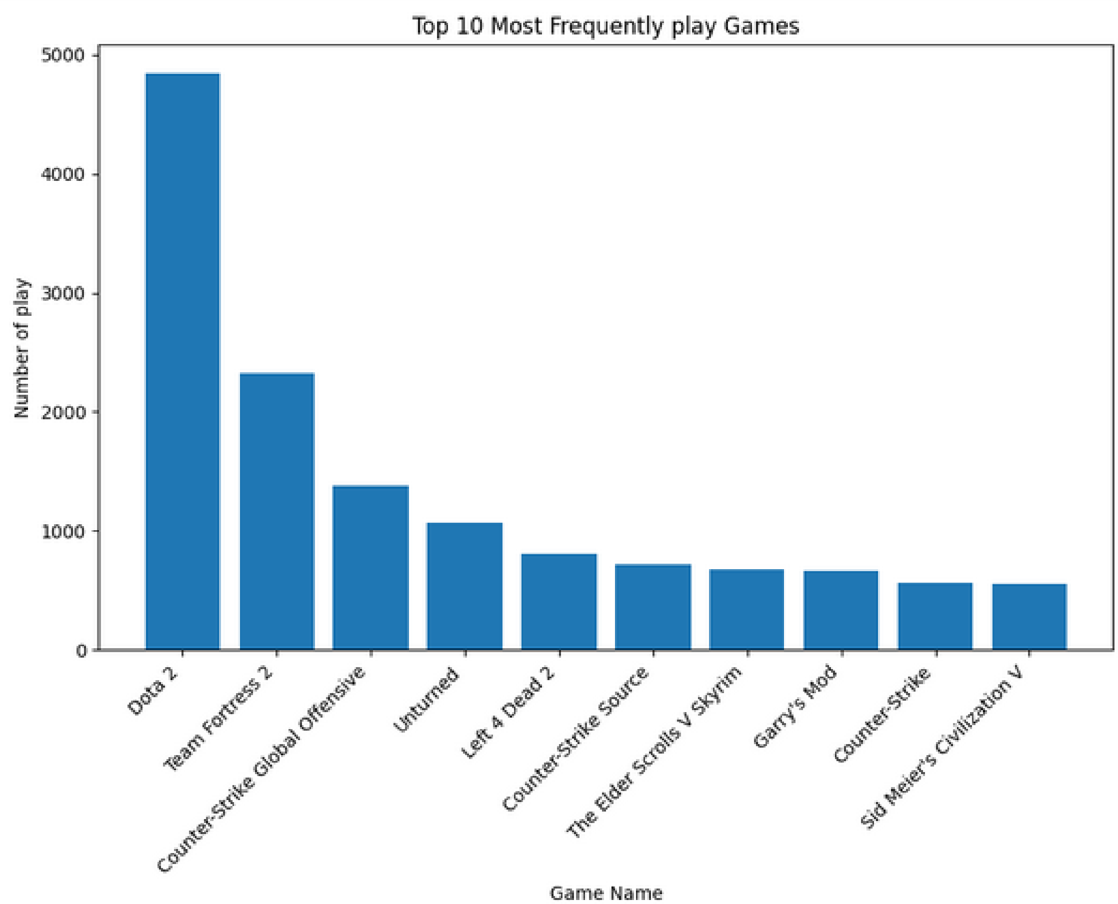
Python

```python
# Import necessary libraries
import matplotlib.pyplot as plt

# Define the number of top games to visualize
N = 10  # You can change this value to visualize a different number of top games

# Summarize the most frequently purchased games
most_purchased = indexed_df.filter(indexed_df['member behaviour'] == 'play') \
    .groupBy('Name of the Game') \
    .count() \
    .orderBy('count', ascending=False)

# Visualize the top N most frequently purchased games
top_n_purchased = most_purchased.limit(N)
top_n_purchased_df = top_n_purchased.toPandas()
plt.figure(figsize=(10, 6))
plt.bar(top_n_purchased_df['Name of the Game'], top_n_purchased_df['count'])
plt.xlabel('Game Name')
plt.ylabel('Number of play')
plt.title('Top {} Most Frequently play Games'.format(N))
plt.xticks(rotation=45, ha='right')
plt.show()
```

▸ (2) Spark Jobs

▸ most_purchased: pyspark.sql.dataframe.DataFrame = [Name of the Game: string, count: long]

Result:-



Command took 4.41 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 5:34:02 PM on fdc

1

I write a code to define the number of top purchased games to visualize from the data frame indexed_df and plot them using bar chart.

Cmd 23

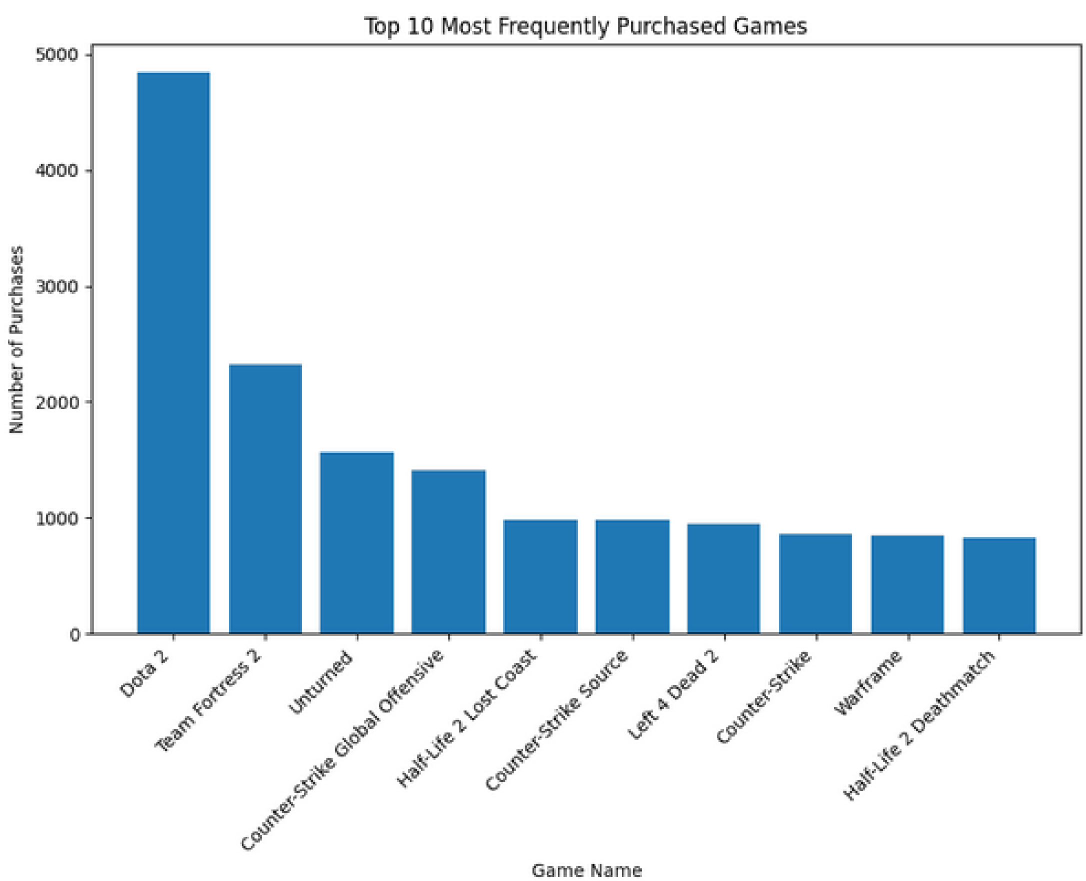
Python

```python
# Import necessary libraries
import matplotlib.pyplot as plt

# Define the number of top games to visualize
N = 10  # You can change this value to visualize a different number of top games

# Summarize the most frequently purchased games
most_purchased = indexed_df.filter(indexed_df['member behaviour'] == 'purchase') \
    .groupBy('Name of the Game') \
    .count() \
    .orderBy('count', ascending=False)

# Visualize the top N most frequently purchased games
top_n_purchased = most_purchased.limit(N)
top_n_purchased_df = top_n_purchased.toPandas()
plt.figure(figsize=(10, 6))
plt.bar(top_n_purchased_df['Name of the Game'], top_n_purchased_df['count'])
plt.xlabel('Game Name')
plt.ylabel('Number of Purchases')
plt.title('Top {} Most Frequently Purchased Games'.format(N))
plt.xticks(rotation=45, ha='right')
plt.show()
```

(2) Spark Jobs

most_purchased: pyspark.sql.dataframe.DataFrame = [Name of the Game: string, count: long]

top_n_purchased: pyspark.sql.dataframe.DataFrame = [Name of the Game: string, count: long]



Command took 1.91 seconds -- by s.a.murtaza@edu.salford.ac.uk at 4/30/2024, 5:34:29 PM on fdc

1

3. Distribution of Play and Purchase actions:-

Then I write a code to calculates the count of actions by member behaviour(“purchase” or “play”) from the data Frame “OccupancyDF” and visualizes the distribution using a bar chart. I first group the dataframe by member behavior and count the occurrences. Then I extract the data for the chart and create a bar chart with appropriate labels and color for each behavior.

Cmd 25

Python

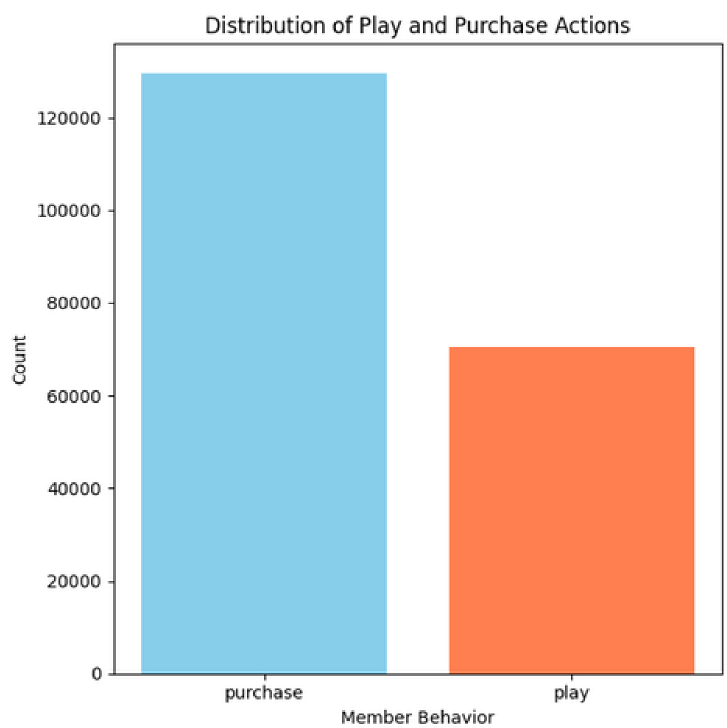
```python
# Assuming you have already loaded your data as 'staticInputDf' with appropriate schema

from pyspark.sql.functions import count

# Option 2: Count Actions by Member Behavior ("purchase" or "play")
behavior_counts = OccupancyDF.groupBy("member behaviour").count()
# Import libraries for plotting
import matplotlib.pyplot as plt

# Assuming 'behavior_counts' contains data for member behavior counts

# Extract data for the chart (assuming columns are named "Member_Behavior" and "count")
labels = list(behavior_counts.select("member behaviour").rdd.flatMap(lambda x: x).collect())
counts = list(behavior_counts.select("count").rdd.flatMap(lambda x: x).collect())

# Create a bar chart
plt.figure(figsize=(6, 6))
plt.bar(labels, counts, color=['skyblue', 'coral'])  # Assign different colors for each behavior
plt.xlabel("Member Behavior")
plt.ylabel("Count")
plt.title("Distribution of Play and Purchase Actions")
plt.xticks(rotation=0)  # No rotation for member behavior labels
plt.tight_layout()
plt.show()
```

(4) Spark Jobs

behavior_counts: pyspark.sql.dataframe.DataFrame = [member behaviour: string, count: long]

Result:-

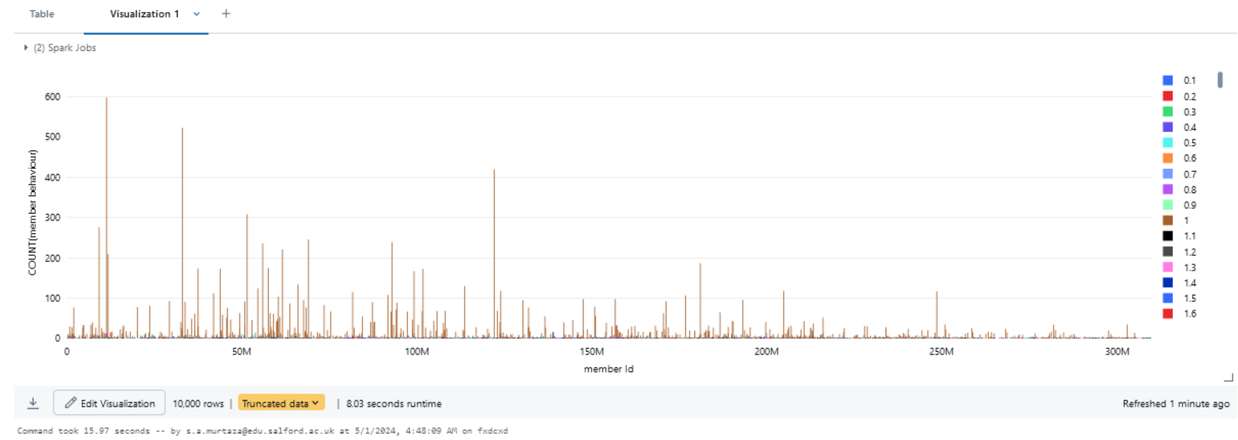
behavior_counts: pyspark.sql.dataframe.DataFrame = [member behaviour: string, count: long]



Command took 6.02 seconds -- by s.a.murtaza@edu.salford.ac.uk at 5/1/2024, 4:49:44 AM on fxdcxd

1

Then I use data bricks inbuilt tool for visiualizing the data frame by maping member id on axises member behavior on y-axises and group by them by status.



1