CS483 - Homework 1

Spark Tutorial

Complete you student information here:

You Name: Aqsa Arif

UIN: 677417263

Net ID: aarif20

In this tutorial, you will learn how to use Apache Spark in local mode on a Colab enviroment. Credits to Stanford CS246.

The tutorial is worth 5 points. The bonus question is worth extra 3 points, which can be added to the total score of this course. After you complete this homework, submit pdf file to gradescope AND ipynb file to blackboard.

Setup

Let's set up Spark on your Colab environment. Run the cell below!

```
!pip install pyspark
!pip install -U -q PyDrive
!apt install openjdk-8-jdk-headless -qq
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting pyspark
  Downloading pyspark-3.3.1.tar.gz (281.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 281.4/281.4 MB 4.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting py4j==0.10.9.5
  Downloading py4j-0.10.9.5-py2.py3-none-any.whl (199 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 199.7/199.7 KB 9.3 MB/s eta 0:00:00
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.3.1-py2.py3-none-any.whl size=281845512
  Stored in directory: /root/.cache/pip/wheels/43/dc/11/ec201cd671da62fa9c5cc77078235e4
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.5 pyspark-3.3.1
The following additional packages will be installed:
  openjdk-8-jre-headless
Suggested packages:
  openjdk-8-demo openjdk-8-source libnss-mdns fonts-dejavu-extra
```

```
  fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei
  fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  openjdk-8-jdk-headless openjdk-8-jre-headless
0 upgraded, 2 newly installed, 0 to remove and 27 not upgraded.
Need to get 36.5 MB of archives.
After this operation, 143 MB of additional disk space will be used.
Selecting previously unselected package openjdk-8-jre-headless:amd64.
(Reading database ... 129499 files and directories currently installed.)
Preparing to unpack .../openjdk-8-jre-headless_8u352-ga-1~20.04_amd64.deb ...
Unpacking openjdk-8-jre-headless:amd64 (8u352-ga-1~20.04) ...
Selecting previously unselected package openjdk-8-jdk-headless:amd64.
Preparing to unpack .../openjdk-8-jdk-headless_8u352-ga-1~20.04_amd64.deb ...
Unpacking openjdk-8-jdk-headless:amd64 (8u352-ga-1~20.04) ...
Setting up openjdk-8-jre-headless:amd64 (8u352-ga-1~20.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to provide /u
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool to prov
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/tnameserv to provi
Setting up openjdk-8-jdk-headless:amd64 (8u352-ga-1~20.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/idlj to provide /usr/b
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/wsimport to provide /u
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jsadebugd to provide /
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/native2ascii to provid
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/javah to provide /usr/
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/clhsdb to provide /usr
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/xjc to provide /usr/bi
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/hsdb to provide /usr/b
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/schemagen to provide /
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/extcheck to provide /u
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jhat to provide /usr/b
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/wsgen to provide /usr/
```

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
```

Now, we authenticate a Google Drive client to download the file we will be processing in our Spark job.

Make sure to follow the interactive instructions.

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# Authenticate and create the PyDrive client
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

```
id='1L6pCQkldvdBoaEhRFzL0VnrggEFvqON4'
downloaded = drive.CreateFile({'id': id})
downloaded.GetContentFile('Bombing_Operations.json.gz')

id='14dyBmcTBA32uXPxDbqr0bFDIzGxMTWwl'
downloaded = drive.CreateFile({'id': id})
downloaded.GetContentFile('Aircraft_Glossary.json.gz')
```

If you executed the cells above, you should be able to see the files *Bombing_Operations.json.gz* and *Aircraft_Glossary.json.gz* under the "Files" tab on the left panel.

```
# Let's import the libraries we will need
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import pyspark
from pyspark.sql import *
from pyspark.sql.functions import *
from pyspark import SparkContext, SparkConf
```

Let's initialize the Spark context.

```
# create the session
conf = SparkConf().set("spark.ui.port", "4050")

# create the context
sc = pyspark.SparkContext(conf=conf)
spark = SparkSession.builder.getOrCreate()
```

You can easily check the current version and get the link of the web interface. In the Spark UI, you can monitor the progress of your job and debug the performance bottlenecks (if your Colab is running with a **local runtime**).

```
spark
```

SparkSession - in-memory

SparkContext

Spark UI

If you are running this Colab on the Google hosted runtime, the cell below will create a *ngrok* tunnel which will allow you to check the Spark UI.

To facilitate setting up the ngrok tunnel, please do the following steps:

1. First, navigate to the ngrok website and create your account: https://dashboard.ngrok.com/login
2. Then, obtain your authentication token from https://dashboard.ngrok.com/get-started/your-authtoken
3. Replace <YOUR_AUTH_TOKEN_HERE> in the following cell with your auth token.

You can then proceed to run the following cells, and you should be able to see the Spark UI as a separate web page.

(If you see a security warning, "Deceptive site ahead", please click "Details -> visit this unsafe site" to view the Spark UI.)

```
!wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
!unzip ngrok-stable-linux-amd64.zip
!./ngrok authtoken <YOUR_AUTH_TOKEN_HERE>
!cat ~/.ngrok2/ngrok.yml
get_ipython().system_raw('./ngrok http 4050 &')
```

```
--2023-01-27 23:29:37--  https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.
Resolving bin.equinox.io (bin.equinox.io)... 18.205.222.128, 54.237.133.81, 52.202.168.
Connecting to bin.equinox.io (bin.equinox.io)|18.205.222.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13832437 (13M) [application/octet-stream]
Saving to: ‘ngrok-stable-linux-amd64.zip’

ngrok-stable-linux- 100%[===================>]  13.19M  13.8MB/s    in 1.0s

2023-01-27 23:29:39 (13.8 MB/s) - ‘ngrok-stable-linux-amd64.zip’ saved [13832437/138324

Archive:  ngrok-stable-linux-amd64.zip
  inflating: ngrok
/bin/bash: -c: line 0: syntax error near unexpected token `newline'
/bin/bash: -c: line 0: `./ngrok authtoken <YOUR_AUTH_TOKEN_HERE>'
cat: /root/.ngrok2/ngrok.yml: No such file or directory
```

```
!curl -s http://localhost:4040/api/tunnels | python3 -c \
    "import sys, json; print(json.load(sys.stdin)['tunnels'][0]['public_url'])"
```

https://c754-35-204-25-255.ngrok.io

Vietnam War

Pres. Johnson: *What do you think about this Vietnam thing? I'd like to hear you talk a little bit.*

Sen. Russell: *Well, frankly, Mr. President, it's the damn worse mess that I ever saw, and I don't like to brag and I never have been right many times in my life, but I knew that we were going to get into this sort of mess when we went in there.*

May 27, 1964

The Vietnam War, also known as the Second Indochina War, and in Vietnam as the Resistance War Against America or simply the American War, was a conflict that occurred in Vietnam, Laos, and Cambodia from 1 November 1955 to the fall of Saigon on 30 April 1975. It was the second of the Indochina Wars and was officially fought between North Vietnam and the government of South Vietnam.

The dataset describes all the air force operation in during the Vietnam War.

Bombing_Operations Get the dataset here

- AirCraft: *Aircraft model (example: EC-47)*
- ContryFlyingMission: *Country*
- MissionDate: *Date of the mission*
- OperationSupported: *Supported War operation* (example: Operation Rolling Thunder)
- PeriodOfDay: *Day or night*
- TakeoffLocation: *Take off airport*
- TimeOnTarget
- WeaponType
- WeaponsLoadedWeight

Aircraft_Glossary Get the dataset here

- AirCraft: *Aircraft model (example: EC-47)*
- AirCraftName
- AirCraftType

Dataset Information:

THOR is a painstakingly cultivated database of historic aerial bombings from World War I through Vietnam. THOR has already proven useful in finding unexploded ordnance in Southeast Asia and improving Air Force combat tactics: https://www.kaggle.com/usaf/vietnam-war-bombing-operations

Load the datasets:

```
Bombing_Operations = spark.read.json("Bombing_Operations.json.gz")
Aircraft_Glossary = spark.read.json("Aircraft_Glossary.json.gz")
```

Check the schema:

```
Bombing_Operations.printSchema()
```

```
root
 |-- AirCraft: string (nullable = true)
 |-- ContryFlyingMission: string (nullable = true)
 |-- MissionDate: string (nullable = true)
 |-- OperationSupported: string (nullable = true)
 |-- PeriodOfDay: string (nullable = true)
 |-- TakeoffLocation: string (nullable = true)
 |-- TargetCountry: string (nullable = true)
 |-- TimeOnTarget: double (nullable = true)
 |-- WeaponType: string (nullable = true)
 |-- WeaponsLoadedWeight: long (nullable = true)
```

```
Aircraft_Glossary.printSchema()
```

```
root
 |-- AirCraft: string (nullable = true)
 |-- AirCraftName: string (nullable = true)
 |-- AirCraftType: string (nullable = true)
```

Get a sample with `take()`:

```
Bombing_Operations.take(3)
```

```
[Row(AirCraft='EC-47', ContryFlyingMission='UNITED STATES OF AMERICA',
MissionDate='1971-06-05', OperationSupported=None, PeriodOfDay='D',
TakeoffLocation='TAN SON NHUT', TargetCountry='CAMBODIA', TimeOnTarget=1005.0,
WeaponType=None, WeaponsLoadedWeight=0),
 Row(AirCraft='EC-47', ContryFlyingMission='UNITED STATES OF AMERICA',
MissionDate='1972-12-26', OperationSupported=None, PeriodOfDay='D',
TakeoffLocation='NAKHON PHANOM', TargetCountry='SOUTH VIETNAM', TimeOnTarget=530.0,
WeaponType=None, WeaponsLoadedWeight=0),
 Row(AirCraft='RF-4', ContryFlyingMission='UNITED STATES OF AMERICA',
MissionDate='1973-07-28', OperationSupported=None, PeriodOfDay='D',
TakeoffLocation='UDORN AB', TargetCountry='LAOS', TimeOnTarget=730.0, WeaponType=None,
WeaponsLoadedWeight=0)]
```

Get a formatted sample with `show()`:

```
Aircraft_Glossary.show()
```

```
+--------+--------------------+--------------------+
|AirCraft|        AirCraftName|        AirCraftType|
+--------+--------------------+--------------------+
|     A-1|Douglas A-1 Skyra...|         Fighter Jet|
|    A-26|Douglas A-26 Invader|        Light Bomber|
|    A-37|Cessna A-37 Drago...|Light ground-atta...|
|     A-4|McDonnell Douglas...|         Fighter Jet|
|     A-5|North American A-...|          Bomber Jet|
|     A-6|Grumman A-6 Intruder|     Attack Aircraft|
|     A-7|  LTV A-7 Corsair II|     Attack Aircraft|
|  AC-119|Fairchild AC-119 ...|Military Transpor...|
|  AC-123|Fairchild C-123 P...|Military Transpor...|
|  AC-130|Lockheed AC-130 S...|Fixed wing ground...|
|   AC-47|Douglas AC-47 Spooky|Ground attack air...|
|    AH-1| Bell AH-1 HueyCobra|          Helicopter|
|     B-1| Rockwell B-1 Lancer|Heavy strategic b...|
|    B-52| B-52 Stratofortress|    Strategic bomber|
|    B-57|Martin B-57 Canberra|     Tactical Bomber|
|    B-66|Douglas B-66 Dest...|        Light Bomber|
|     C-1| Grumman C-1A Trader|           Transport|
|   C-117|     C-117D Skytrain|           Transport|
|   C-119|Fairchild C-119 F...|Military Transpor...|
|   C-123|Fairchild C-123 P...|Military Transpor...|
+--------+--------------------+--------------------+
only showing top 20 rows
```

```
print("In total there are {0} operations".format(Bombing_Operations.count()))
```

```
In total there are 4400775 operations
```

Question 1: Which countries are involved and in how many missions?

Keywords: `Dataframe API`, `SQL`, `group by`, `sort`

Let's group the missions by `ContryFlyingMission` and count how many records exist:

```
Bombing_Operations.show()
```

```
+--------+--------------------+-----------+-----------------+-----------+-------------
|AirCraft| ContryFlyingMission|MissionDate|OperationSupported|PeriodOfDay|TakeoffLocati
+--------+--------------------+-----------+-----------------+-----------+-------------
|   EC-47|UNITED STATES OF ...| 1971-06-05|             null|          D|   TAN SON NH
|   EC-47|UNITED STATES OF ...| 1972-12-26|             null|          D|  NAKHON PHAN
|    RF-4|UNITED STATES OF ...| 1973-07-28|             null|          D|       UDORN
|     A-1|UNITED STATES OF ...| 1970-02-02|             null|          N|  NAKHON PHAN
|    A-37|     VIETNAM (SOUTH)| 1970-10-08|             null|          D|         DANA
|     F-4|UNITED STATES OF ...| 1970-11-25|             null|          D|        UBON
|     A-4|UNITED STATES OF ...| 1972-03-08|             null|          D|    TONKIN GU
|     F-4|UNITED STATES OF ...| 1971-12-27|             null|       null|       UDORN
|     A-7|UNITED STATES OF ...| 1972-05-24|             null|       null|    TONKIN GU
|   EC-47|UNITED STATES OF ...| 1972-09-12|             null|          D|   TAN SON NH
|   CH-53|UNITED STATES OF ...| 1974-06-13|             null|          N|  NAKHON PHAN
|   CH-53|UNITED STATES OF ...| 1974-12-19|             null|          D|  NAKHON PHAN
|     O-1|     VIETNAM (SOUTH)| 1973-10-24|             null|          D|      NHA TRA
|    UH-1|     VIETNAM (SOUTH)| 1974-03-19|             null|          D|        PHU C
|     C-7|UNITED STATES OF ...| 1970-05-08|             null|          D|   TAN SON NH
|     A-6|UNITED STATES OF ...| 1971-05-12|             null|          N|    TONKIN GU
|   EB-66|UNITED STATES OF ...| 1971-12-03|             null|          N|          KOR
|    T-28|                LAOS| 1971-12-19|             null|          D|     SAVANAKH
|     A-6|UNITED STATES OF ...| 1972-08-18|             null|       null|    TONKIN GU
|     A-7|UNITED STATES OF ...| 1972-10-15|             null|          D|    TONKIN GU
+--------+--------------------+-----------+-----------------+-----------+-------------
only showing top 20 rows
```

```
tmp1 = Bombing_Operations.groupBy("ContryFlyingMission").count().sort(desc("count"))
tmp1.show()
```

```
+--------------------+-------+
| ContryFlyingMission|  count|
+--------------------+-------+
|UNITED STATES OF ...|3708997|
|     VIETNAM (SOUTH)| 622013|
|                LAOS|  32777|
|       KOREA (SOUTH)|  24469|
|           AUSTRALIA|  12519|
+--------------------+-------+
```

```
missions_counts = Bombing_Operations.groupBy("ContryFlyingMission")\
                                    .agg(count("*").alias("MissionsCount"))\
                                    .sort(desc("MissionsCount"))
missions_counts.show()
```

```
+--------------------+-------------+
| ContryFlyingMission|MissionsCount|
+--------------------+-------------+
|UNITED STATES OF ...|      3708997|
|     VIETNAM (SOUTH)|       622013|
|                LAOS|        32777|
|       KOREA (SOUTH)|        24469|
|           AUSTRALIA|        12519|
+--------------------+-------------+
```

In this case we used the DataFrame API, but we could rewite the `groupBy` using pure SQL:

```
Bombing_Operations.registerTempTable("Bombing_Operations")

query = """
SELECT ContryFlyingMission, count(*) as MissionsCount
FROM Bombing_Operations
GROUP BY ContryFlyingMission
ORDER BY MissionsCount DESC
"""

missions_counts = spark.sql(query)
missions_counts.show()
```

```
/usr/local/lib/python3.8/dist-packages/pyspark/sql/dataframe.py:229: FutureWarning: Dep
  warnings.warn("Deprecated in 2.0, use createOrReplaceTempView instead.", FutureWarnin
+--------------------+-------------+
| ContryFlyingMission|MissionsCount|
+--------------------+-------------+
|UNITED STATES OF ...|      3708997|
|     VIETNAM (SOUTH)|       622013|
|                LAOS|        32777|
|       KOREA (SOUTH)|        24469|
|           AUSTRALIA|        12519|
+--------------------+-------------+
```

The Dataframe is small enough to be moved to Pandas:

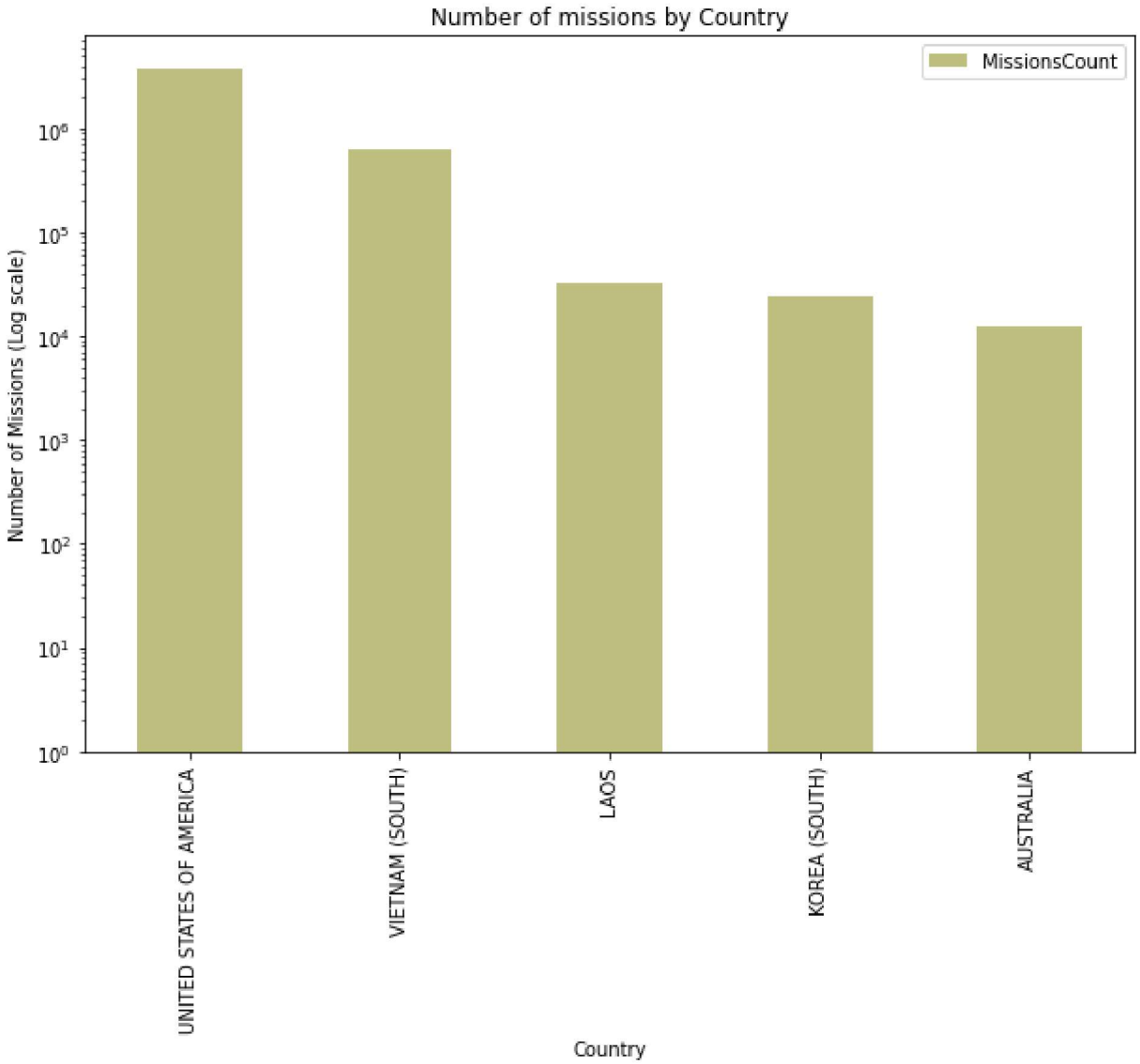
```
missions_count_pd = missions_counts.toPandas()
missions_count_pd.head()
```

	ContryFlyingMission	MissionsCount
0	UNITED STATES OF AMERICA	3708997
1	VIETNAM (SOUTH)	622013

Let's plot a barchart with the number of missions by country:

```
pl = missions_count_pd.plot(kind="bar",
                            x="ContryFlyingMission", y="MissionsCount",
                            figsize=(10, 7), log=True, alpha=0.5, color="olive")
pl.set_xlabel("Country")
pl.set_ylabel("Number of Missions (Log scale)")
pl.set_title("Number of missions by Country")
```

```
Text(0.5, 1.0, 'Number of missions by Country')
```



Questions 2: Show the number of missions in time for each of the countries involved.

Keywords: `group by`, `parse date`, `plot`

Let's select the relevant columns:

```
missions_countries = Bombing_Operations.selectExpr(["to_date(MissionDate) as MissionDate", "C
missions_countries.show()
```

```
+-----------+--------------------+
|MissionDate| ContryFlyingMission|
+-----------+--------------------+
| 1971-06-05|UNITED STATES OF ...|
| 1972-12-26|UNITED STATES OF ...|
| 1973-07-28|UNITED STATES OF ...|
| 1970-02-02|UNITED STATES OF ...|
| 1970-10-08|     VIETNAM (SOUTH)|
| 1970-11-25|UNITED STATES OF ...|
| 1972-03-08|UNITED STATES OF ...|
| 1971-12-27|UNITED STATES OF ...|
| 1972-05-24|UNITED STATES OF ...|
| 1972-09-12|UNITED STATES OF ...|
| 1974-06-13|UNITED STATES OF ...|
| 1974-12-19|UNITED STATES OF ...|
| 1973-10-24|     VIETNAM (SOUTH)|
| 1974-03-19|     VIETNAM (SOUTH)|
| 1970-05-08|UNITED STATES OF ...|
| 1971-05-12|UNITED STATES OF ...|
| 1971-12-03|UNITED STATES OF ...|
| 1971-12-19|                LAOS|
| 1972-08-18|UNITED STATES OF ...|
| 1972-10-15|UNITED STATES OF ...|
+-----------+--------------------+
only showing top 20 rows
```

The filed MissionDate is converted to a Python `date` object.

Now we can group by `MissionDate` and `ContryFlyingMission` to get the count:

```
missions_by_date = missions_countries\
                   .groupBy(["MissionDate", "ContryFlyingMission"])\
                   .agg(count("*").alias("MissionsCount"))\
                   .sort(asc("MissionDate"))
missions_by_date.show()
```

```
+-----------+--------------------+-------------+
|MissionDate| ContryFlyingMission|MissionsCount|
```

```
+-----------+--------------------+-------------+
| 1965-10-01|UNITED STATES OF ...|          447|
| 1965-10-02|UNITED STATES OF ...|          652|
| 1965-10-03|UNITED STATES OF ...|          608|
| 1965-10-04|UNITED STATES OF ...|          532|
| 1965-10-05|UNITED STATES OF ...|          697|
| 1965-10-05|     VIETNAM (SOUTH)|           72|
| 1965-10-06|     VIETNAM (SOUTH)|           49|
| 1965-10-06|UNITED STATES OF ...|          689|
| 1965-10-07|UNITED STATES OF ...|          605|
| 1965-10-07|     VIETNAM (SOUTH)|           50|
| 1965-10-08|     VIETNAM (SOUTH)|           64|
| 1965-10-08|UNITED STATES OF ...|          700|
| 1965-10-09|     VIETNAM (SOUTH)|           69|
| 1965-10-09|UNITED STATES OF ...|          677|
| 1965-10-10|     VIETNAM (SOUTH)|           59|
| 1965-10-10|UNITED STATES OF ...|          789|
| 1965-10-11|UNITED STATES OF ...|          455|
| 1965-10-11|     VIETNAM (SOUTH)|           48|
| 1965-10-12|UNITED STATES OF ...|          449|
| 1965-10-12|     VIETNAM (SOUTH)|           62|
+-----------+--------------------+-------------+
only showing top 20 rows
```

Now we can plot the content with a different series for each country:

```
fig = plt.figure(figsize=(10, 6))

countries = missions_by_date.select("ContryFlyingMission").distinct().toPandas().values.squee
for country in countries:
    missions = missions_by_date.where(missions_by_date.ContryFlyingMission==country).toPandas
    plt.plot(missions["MissionDate"], missions["MissionsCount"], label=country)

plt.legend(loc='best')
```

```
<matplotlib.legend.Legend at 0x7fdfc79c4670>
```

We can observe how South Vietnam increased its missions starting from 1970. The drop in 1973 is motivated by the Paris Peace Accords that took place on January 27th, 1973, to establish peace in Vietnam and end the war.

Question 3: Who bombed this location?

Keywords: `RDD map reduce cache save results`

This picture is the Hanoi POL facility (North Vietnam) burning after it was attacked by the U.S. Air Force on 29 June 1966 in the context of the Rolling Thunder operation.

We are interested in discovering what was the most common take-off location during that day.

```
jun_29_operations = Bombing_Operations.where("MissionDate = '1966-06-29' AND TargetCountry='N
jun_29_operations.show()
```

```
+--------+--------------------+-----------+------------------+-----------+-------------
|AirCraft| ContryFlyingMission|MissionDate|OperationSupported|PeriodOfDay|TakeoffLocati
+--------+--------------------+-----------+------------------+-----------+-------------
|   F-105|UNITED STATES OF ...| 1966-06-29|       STEEL TIGER|          D|          TAKH
|   C-130|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          N|          DANA
|     F-4|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          D|  CONSTELLATI
|   C-130|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          N|          UBON
|     A-1|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          D|         UDORN
|     A-4|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          E|  CONSTELLATI
|   F-105|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          E|          KOR
|     A-4|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          D|  CONSTELLATI
|   F-105|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          D|          KOR
|     F-4|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          D|          RANG
|    RA-5|UNITED STATES OF ...| 1966-06-29|              null|          D|  CONSTELLATI
```

```
|   F-105|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          D|          TAKH
|   RB-66|UNITED STATES OF ...| 1966-06-29|         FOG BOUND|          D|          TAKH
|     F-4|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          D|  CONSTELLATI
|     F-4|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          N|        UBON
|     F-4|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          N|        UBON
|     A-4|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          D|  CONSTELLATI
|     A-4|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          D|  CONSTELLATI
|     E-2|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          D|  CONSTELLATI
|    B-66|UNITED STATES OF ...| 1966-06-29|      ROLLING THUN|          N|          TAKH
+--------+--------------------+-----------+------------------+-----------+-------------
only showing top 20 rows
```

Which coutries scheduled missions that day?

```
jun_29_operations.groupBy("ContryFlyingMission").agg(count("*").alias("MissionsCount")).toPar
```

	ContryFlyingMission	MissionsCount
0	VIETNAM (SOUTH)	6
1	UNITED STATES OF AMERICA	389

Most of the operation that day were performed by USA airplanes.

```
jun_29_operations.take(1)
```

```
[Row(AirCraft='F-105', ContryFlyingMission='UNITED STATES OF AMERICA',
MissionDate='1966-06-29', OperationSupported='STEEL TIGER', PeriodOfDay='D',
TakeoffLocation='TAKHLI', TargetCountry='NORTH VIETNAM', TimeOnTarget=310.0,
WeaponType='1000LB MK-83', WeaponsLoadedWeight=-1)]
```

You can specify to cache the content in memory:

```
jun_29_operations.cache()
```

```
DataFrame[AirCraft: string, ContryFlyingMission: string, MissionDate: string,
OperationSupported: string, PeriodOfDay: string, TakeoffLocation: string,
TargetCountry: string, TimeOnTarget: double, WeaponType: string, WeaponsLoadedWeight:
bigint]
```

Now you can count the number of rows and move the content to the cache:

```
%time jun_29_operations.count()
```

```
CPU times: user 51.7 ms, sys: 9.29 ms, total: 61 ms
```

```
Wall time: 10.5 s
395
```

The second time the content is cached and the operation is much faster:

```
%time jun_29_operations.count()
```

```
CPU times: user 1.61 ms, sys: 0 ns, total: 1.61 ms
Wall time: 65.6 ms
395
```

You can also save the results on a file...

```
jun_29_operations.write.mode('overwrite').json("jun_29_operations.json")
```

... and read from the file:

```
jun_29_operations = spark.read.json("jun_29_operations.json")
```

We can use the simple DataFrame API...

```
TakeoffLocationCounts = jun_29_operations\
                            .groupBy("TakeoffLocation").agg(count("*").alias("MissionsCount")
                            .sort(desc("MissionsCount"))
TakeoffLocationCounts.show()
```

```
+----------------+-------------+
| TakeoffLocation|MissionsCount|
+----------------+-------------+
|   CONSTELLATION|           87|
|           TAKHLI|           56|
|            KORAT|           55|
|         UDORN AB|           44|
|          UBON AB|           44|
|           RANGER|           35|
|           DANANG|           35|
|    TAN SON NHUT|           26|
|HANCOCK (CVA-19)|           10|
|    CAM RANH BAY|            2|
|          CUBI PT|            1|
+----------------+-------------+
```

... or the explicit Map/Reduce format with RDDs.

First we emit a pair in the format (Location, 1):

```
all_locations = jun_29_operations.rdd.map(lambda row: (row.TakeoffLocation, 1))
all_locations.take(3)
```

```
[('TAKHLI', 1), ('DANANG', 1), ('CONSTELLATION', 1)]
```

Then, we sum counters in the reduce step, and we sort by count:

```
locations_counts_rdd = all_locations.reduceByKey(lambda a, b: a+b).sortBy(lambda r: -r[1])
locations_counts_rdd.take(3)
```

```
[('CONSTELLATION', 87), ('TAKHLI', 56), ('KORAT', 55)]
```

Now we can convert the RDD in dataframe by mapping the pairs to objects of type `Row`

```
locations_counts_with_schema = locations_counts_rdd.map(lambda r: Row(TakeoffLocation=r[0], M
locations_counts = spark.createDataFrame(locations_counts_with_schema)
locations_counts.show()
```

```
+----------------+-------------+
| TakeoffLocation|MissionsCount|
+----------------+-------------+
|   CONSTELLATION|           87|
|          TAKHLI|           56|
|           KORAT|           55|
|         UBON AB|           44|
|        UDORN AB|           44|
|          DANANG|           35|
|          RANGER|           35|
|    TAN SON NHUT|           26|
|HANCOCK (CVA-19)|           10|
|    CAM RANH BAY|            2|
|         CUBI PT|            1|
+----------------+-------------+
```

That day the most common take-off location was the ship USS Constellation (CV-64). We cannot univocally identify one take off location, but we can reduce the possible candidates. Next steps: explore TimeOnTarget feature.

USS Constellation (CV-64), a Kitty Hawk-class supercarrier, was the third ship of the United States Navy to be named in honor of the "new constellation of stars" on the flag of the United States. One of the fastest ships in the Navy, as proven by her victory during a battlegroup race held in 1985, she was nicknamed "Connie" by her crew and officially as "America's Flagship"

Questions 4: What is the most used aircraft type during the Vietnam war (number of missions)?

Keywords: `join group by`

Let's check the content of `Aircraft_Glossary` :

```
Aircraft_Glossary.show(5)
```

```
+--------+--------------------+--------------------+
|AirCraft|        AirCraftName|        AirCraftType|
+--------+--------------------+--------------------+
|     A-1|Douglas A-1 Skyra...|         Fighter Jet|
|    A-26|Douglas A-26 Invader|        Light Bomber|
|    A-37|Cessna A-37 Drago...|Light ground-atta...|
|     A-4|McDonnell Douglas...|         Fighter Jet|
|     A-5|North American A-...|          Bomber Jet|
+--------+--------------------+--------------------+
only showing top 5 rows
```

We are interested in the filed `AirCraftType` .

```
Bombing_Operations.select("AirCraft").show(5)
```

```
+--------+
|AirCraft|
+--------+
|   EC-47|
|   EC-47|
|    RF-4|
|     A-1|
|    A-37|
+--------+
```

```
only showing top 5 rows
```

We can join on the column `AirCraft` of both dataframes.

With Dataframe API:

```
missions_joined = Bombing_Operations.join(Aircraft_Glossary,
                                          Bombing_Operations.AirCraft == Aircraft_Glossary.Ai
missions_joined
```

```
DataFrame[AirCraft: string, ContryFlyingMission: string, MissionDate: string,
OperationSupported: string, PeriodOfDay: string, TakeoffLocation: string,
TargetCountry: string, TimeOnTarget: double, WeaponType: string, WeaponsLoadedWeight:
bigint, AirCraft: string, AirCraftName: string, AirCraftType: string]
```

We can select only the field we are interested in:

```
missions_aircrafts = missions_joined.select("AirCraftType")
missions_aircrafts.show(5)
```

```
+--------------------+
|        AirCraftType|
+--------------------+
|Military Transpor...|
|Military Transpor...|
|  Fighter bomber jet|
|         Fighter Jet|
|Light ground-atta...|
+--------------------+
only showing top 5 rows
```

And finally we can group by `AirCraftType` and count:

```
missions_aircrafts.groupBy("AirCraftType").agg(count("*").alias("MissionsCount"))\
                  .sort(desc("MissionsCount"))\
                  .show()
```

```
+--------------------+-------------+
|        AirCraftType|MissionsCount|
+--------------------+-------------+
|  Fighter Jet Bomber|      1073126|
|         Fighter Jet|       882594|
|  Jet Fighter Bomber|       451385|
|     Attack Aircraft|       315246|
|Light ground-atta...|       267457|
|  Fighter bomber jet|       242231|
|Military Transpor...|       228426|
|  Utility Helicopter|       146653|
```

```
|    Strategic bomber|        99100|
|     Tactical Bomber|        82219|
|Observation Aircraft|        81820|
|Fixed wing ground...|        75058|
|Ground attack air...|        73843|
|Carrier-based Fig...|        58691|
|   Training Aircraft|        48435|
|       Light fighter|        39999|
|        Light Bomber|        39262|
|Light Tactical Bo...|        34738|
| Light Utility Plane|        28582|
|Observation/ Ligh...|        24491|
+--------------------+-------------+
only showing top 20 rows
```

In alternative we can rewrite this in pure SQL:

```
Bombing_Operations.registerTempTable("Bombing_Operations")
Aircraft_Glossary.registerTempTable("Aircraft_Glossary")

query = """
SELECT AirCraftType, count(*) MissionsCount
FROM Bombing_Operations bo
JOIN Aircraft_Glossary ag
ON bo.AirCraft = ag.AirCraft
GROUP BY AirCraftType
ORDER BY MissionsCount DESC
"""

spark.sql(query).show()
```

```
/usr/local/lib/python3.8/dist-packages/pyspark/sql/dataframe.py:229: FutureWarning: Dep
  warnings.warn("Deprecated in 2.0, use createOrReplaceTempView instead.", FutureWarnin
+--------------------+-------------+
|        AirCraftType|MissionsCount|
+--------------------+-------------+
|  Fighter Jet Bomber|      1073126|
|         Fighter Jet|       882594|
|  Jet Fighter Bomber|       451385|
|     Attack Aircraft|       315246|
|Light ground-atta...|       267457|
|  Fighter bomber jet|       242231|
|Military Transpor...|       228426|
|  Utility Helicopter|       146653|
|    Strategic bomber|        99100|
|     Tactical Bomber|        82219|
|Observation Aircraft|        81820|
|Fixed wing ground...|        75058|
|Ground attack air...|        73843|
|Carrier-based Fig...|        58691|
|   Training Aircraft|        48435|
|       Light fighter|        39999|
```

```
|        Light Bomber|        39262|
|Light Tactical Bo...|        34738|
| Light Utility Plane|        28582|
|Observation/ Ligh...|        24491|
+--------------------+-------------+
only showing top 20 rows
```

The aircrafts of type `Fighter Jet Bomber` participated in most of the missions in the Vietnam war.

Note: This dataset would require further cleaning and normalization. See `Fighter Jet Bomber`, `Jet Fighter Bomber`, `Fighter bomber jet`

Bonus Question: Word Count in Spark

If you executed the cells below, you should be able to see the file pg100.txt under the "Files" tab on the left panel.

```
id='1SE6k_0YukzGd5wK-E4i6mG83nydlfvSa'
downloaded = drive.CreateFile({'id': id})
downloaded.GetContentFile('pg100.txt')
```

If you successfully run the setup stage, you are ready to work on the *pg100.txt* file which contains a copy of the complete works of Shakespeare.

Write a Spark application which outputs the number of words that start with each letter. This means that for every letter, we want to count the total number of (non-unique) words that start with a specific letter.

In your implementation, **ignore the letter case**, i.e., consider all words as lower case. Also, you can ignore all words that **start** with a non-alphabetic character. You should output word counts for the **entire document**, inclusive of the title, author, and the main texts. If you encounter words broken as a result of new lines, e.g. "pro-ject" where the segment after the dash sign is on a new line, no special processing is needed and you can safely consider it as two words.

Your outputs will be graded on a range -- if your differences from the ground-truths are within an error threshold of 5, you'll be considered correct.

```
from pyspark.sql import *
from pyspark.sql.functions import *
from pyspark import SparkContext
import pandas as pd
import re
```

```
# create the Spark Session
spark = SparkSession.builder.getOrCreate()

# create the Spark Context
sc = spark.sparkContext


from IPython.utils import tempdir

# YOUR
txt = spark.read.text("pg100.txt")

txt.printSchema()

#txt.show()
print("In total there are {0} rows".format(txt.count()))
data = txt.toDF("txt")

temp = data.rdd.flatMap(lambda row: row[0].split(" "))
temp = temp.filter(lambda x: re.match("^[a-z]", x))

print("In total there are {0} words".format(temp.count()))

row = Row("word")
temp2 = temp.map(row).toDF()

temp3 = temp2.withColumn("alphabet", lower(temp2.word.substr(1, 1)))

temp4 = temp3.groupby("alphabet").agg(count("alphabet").alias("alphaCount")).sort(asc("alphab
makePanda = temp4.toPandas()
makePanda.head(30)
```

```
root
 |-- value: string (nullable = true)

In total there are 124787 rows
In total there are 678107 words
```

	alphabet	alphaCount
0	a	63748
1	b	34561
2	c	23496
3	d	23531
4	e	10431
5	f	28819
6	g	14703
7	h	50511
8	i	32292
9	j	1593
10	k	5789
11	l	22353
12	m	46233
13	n	21813
14	o	34201
15	p	19344
16	q	1332
17	r	10400
18	s	52643
19	t	101603
20	u	7667
21	v	4131
22	w	44981
23	y	21879
24	z	53

Colab paid products - Cancel contracts here

✓ 3s completed at 9:46 PM