```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://apache.mirror.colo-serv.net/spark/spark-2.4.7/spark-2.4.7-bin-hadoop2.7.t
!tar xf spark-2.4.7-bin-hadoop2.7.tgz
!pip install -q findspark

import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-2.4.7-bin-hadoop2.7"

import findspark
findspark.init("spark-2.4.7-bin-hadoop2.7")# SPARK_HOME

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

sc = SparkContext.getOrCreate()
spark = SparkSession.builder.getOrCreate()
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
medline_raw = sc.textFile("/content/drive/MyDrive/mesh_terms.txt");

medline_lists = medline_raw.map(lambda line: line.split("|"))
print(medline_lists.take(5))

topics = medline_lists.flatMap(lambda topiclist: topiclist)
print(topics.take(5))
```

```
[['Intellectual Disability', 'Maternal-Fetal Exchange', 'Pregnancy Complications'], ['An
['Intellectual Disability', 'Maternal-Fetal Exchange', 'Pregnancy Complications', 'Amnio
```

```
topic_cnt = topics.map(lambda topic: (topic,1))\
                  .reduceByKey(lambda x,y: x+y )
print(topic_cnt.take(5))
```

```
[('Intellectual Disability', 99), ('Maternal-Fetal Exchange', 60), ('Amniocentesis', 24)
```

```
cnt_topicList = topic_cnt.map(lambda tc: (tc[1],tc[0]))\
                         .groupByKey()

print(cnt_topicList.take(5))
```

```
print(cnt_topicList.take(5))
print(cnt_topicList.map(lambda x: (x[0], list(x[1]))).take(5))
```

```
[(60, <pyspark.resultiterable.ResultIterable object at 0x7f836e1ca750>), (24, <pyspark.r
[(60, ['Maternal-Fetal Exchange', 'Urologic Surgical Procedures', 'Accident Prevention',
```

```
# ascending is set to false, so it's descending
cnt_topicList_sorted = cnt_topicList.sortByKey(False)
cnt_topicList_sorted.map(lambda x: (x[0], list(x[1])))\
                    .take(100)
```

```
(322, ['Family Planning Services']),
(318, ['History']),
(316, ['Statistics as Topic']),
(314, ['Agriculture', 'Family Characteristics', 'Pregnancy Complications']),
(311, ['Antibiotics, Antitubercular']),
(310, ['Antibodies']),
(309,
 ['Congenital Abnormalities',
  'Health Knowledge, Attitudes, Practice',
  'Government Regulation']),
(308, ['Communication']),
(307, ['Social Class']),
(306,
 ['Mental Disorders',
  'Anti-Bacterial Agents',
  'Cardiovascular System',
  'Tomography, X-Ray Computed']),
(305, ['Body Fluids']),
(304, ['Urine']),
(302, ['Surgical Procedures, Operative']),
(301, ['Legislation as Topic']),
(299, ['Population Growth']),
(285, ['Respiration', 'Contraception']),
(282, ['Antigens']),
(281, ['Drug Therapy']),
(279, ['Personality', 'Marriage']),
(278, ['Work', 'Radiation Effects', 'Sexual Behavior']),
(277, ['Health Education']),
(273, ['Postoperative Complications']),
(270, ['Military Personnel']),
(268, ['Lipid Metabolism']),
(263, ['Quality of Life', 'Education, Medical']),
(259, ['Pregnancy', 'Population Characteristics']),
(256, ['Magnetic Resonance Imaging']),
(252, ['Ethics, Medical', 'Light']),
(251, ['Bacteriology', 'Transcription, Genetic', 'Urban Population']),
(250, ['Nutritional Physiological Phenomena']),
(248,
 ['Blood Proteins',
  'Transients and Migrants',
  'Skin Diseases',
  'Temperature']),
(247, ['Family']),
(246, ['Histological Techniques', 'Adolescent']),
(245, ['Organizations', 'Blood Pressure', 'Data Collection']),
```

```
(244, ['Attitude of Health Personnel']),
(242, ['Prejudice']),
(241, ['Dermatologic Agents']),
(240, ['Heart']),
(239, ['Diagnosis, Differential']),
(238, ['Angiography']),
(237, ['Niacin', 'Leadership', 'Plants', 'Physician-Patient Relations']),
(236, ['Kidney Transplantation']),
(235, ['Infant Mortality']),
(234, ['Plants, Medicinal']),
(233, ['Prostheses and Implants']),
(232, ['Human Experimentation']),
(231, ['International Cooperation']),
(230, ['Occupational Diseases', 'Geography'])]
```

```
# Let's create a frequency count.
# This is an RDD of integer pairs (cnt, freq), e.g. (5,10),
# meaning that there are 10 topics having a count of 5.
cnt_freq = cnt_topicList.map(lambda x: (x[0], len(x[1])))

cnt_freq.collect()
```

```
(28, 89),
(10, 294),
(176, 4),
(32, 62),
(138, 7),
(96, 9),
(18, 152),
(74, 14),
(90, 7),
(8, 380),
(212, 2),
(20, 132),
(78, 15),
(192, 4),
(22, 106),
(556, 1),
(306, 4),
(166, 2),
(52, 30),
(6, 595),
(110, 4),
(14, 200),
(26, 78),
(228, 1),
(30, 59),
(48, 30),
(40, 44),
(124, 5),
(56, 15),
(142, 4),
(42, 35),
(98, 10),
(44, 17),
(88, 11)
```

```
(88, 11),
(282, 1),
(150, 6),
(248, 4),
(46, 33),
(16, 176),
(104, 7),
(120, 11),
(114, 4),
(252, 2),
(158, 2),
(156, 6),
(94, 15),
(64, 26),
(2, 2185),
(68, 17),
(472, 1),
(122, 9),
(184, 2),
(36, 45),
(164, 6),
(206, 1),
(136, 5),
(50, 35),
(66, 16),
(58, 20),
(70, 15),
```