Program for Task4:

Task-4: Stream twitter data into Spark continuously for 2 hours. After 2 hours the streaming should automatically stop. Identify 2 topic keywords. As the data is streaming in, based on the 2 topic keywords, count the number of times the keywords appear in the tweets. Output the count for each keyword. Caution: Do not use common words. Note: Specify the following in your README file: - The 2 keywords used - The date and time when you started collecting data. (marks will be deducted if the above information is not included in the README file)

Listener:

```
# Importing the required packages
>>> import tweepy
>>> from tweepy import OAuthHandler
>>> from tweepy import Stream
>>> from tweepy.streaming import StreamListener
>>> import socket
>>> import json
>>> from datetime import datetime
>>> now = datetime.now()
>>> print(now)
2021-04-02 02:53:03.315525
# Access tokens and API tokens from Twitter developer Credentials to connect to twitter account
>>> access_token = "1163449936140222464-RtMJqcL1kbvtdOm8uAHTF3TkMp3bLP"
>>> access_secret = "9GFCAce7BMWHfPcUjYkdqomZ9lx1GjLQei4XhpAyvl1Fw"
>>> consumer_key = "U20Ejib46e1uOV2dva3FXkTim"
>>> consumer_secret = “llGqrSuXUNyz3767ZGaQDeJTlr4AcJCZ5lVwUraeDKxiXchxuu"
# Listener class for tweets
>>> class TweetsListener(StreamListener):
        # initialized the constructor
...     def __init__(self, csocket):
...         self.client_socket = csocket
...     def on_data(self, data):
...         try:
                # Twitter data which comes as a JSON format is read
...             msg = json.loads(data)
                # the 'text' in the JSON file contains the tweet.
...             print(msg[‘text'].encode('utf-8'))
                # the tweet data is sent to the client socket
...             self.client_socket.send(msg['text'].encode('utf-8'))
...             return True
...         except BaseException as e:
                # Exception handling
...             print("Something is wrong : %s" % str(e))
...             return True
```

```
...     def on_error(self, status):
...             print(status)
...             return True
...
# Send the tweets to socket port
>>> def sendData(c_socket,Keyword):
        # passing authentication credentials keys
...     auth = OAuthHandler(consumer_key, consumer_secret)
...     auth.set_access_token(access_token, access_secret)
        # twitter_stream will get the actual live tweet data
...     twitter_stream = Stream(auth, TweetsListener(c_socket))
        # filter the tweet feeds related to Keyword and language english
...     twitter_stream.filter(track=Keyword,languages=["en"])
...
# create a socket object
>>> s = socket.socket()
# Get local machine name : host and port, Port numbers start from 5000
>>> host = socket.gethostname()
>>> port = 5566
# Bind port and socket
>>> s.bind((host, port))
>>> print("Listening on port: %s" % str(port))
Listening on port: 5566
# Establish the connection with client.
>>> s.listen(5)
>>> c, addr = s.accept()
# Waits till the sender sends the data
>>> print("Received request from: " + str(addr))
Received request from: ('192.168.122.2', 57122)
# Keep the stream data available
>>> sendData(c,Keyword=[‘NASA','Mars'])
```

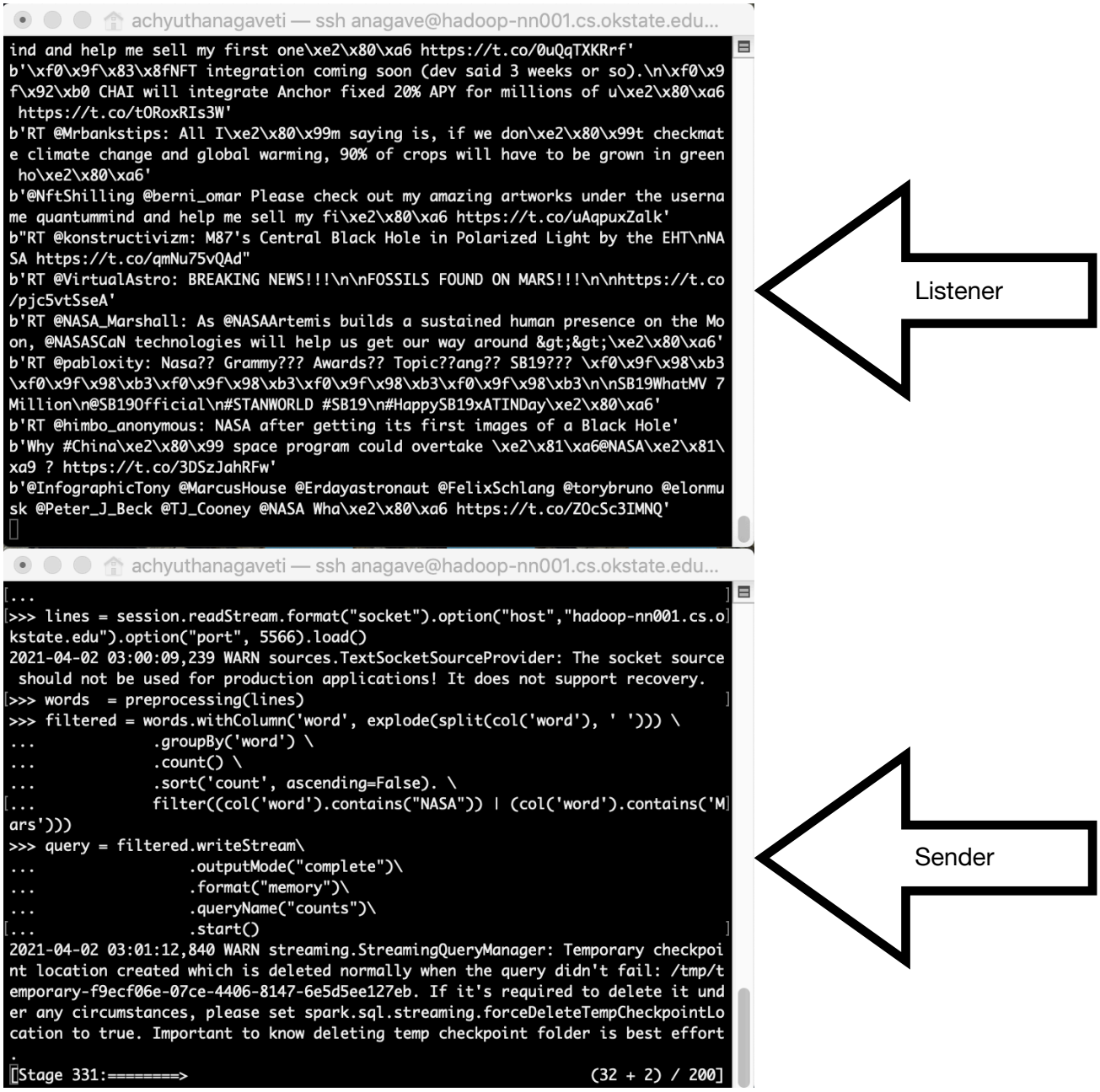
Sender:

```
# Importing the required packages
>>> import pyspark
>>> from pyspark.sql import SparkSession
>>> from pyspark.sql.functions import *
>>> from pyspark.sql.types import *
>>> import pyspark.sql.functions as F
>>> from datetime import datetime
>>> session = SparkSession.builder.appName("Twitter-
stream").master("local[*]").getOrCreate()
>>> now = datetime.now()
# printing the start time of streaming data
>>> print(now)
2021-04-02 02:58:18.963137
# Defining function for Data pre-processing the tweets. Removing special
characters,RT,hashtags and urls
>>> def preprocessing(lines):
...     words = lines.select(explode(split(lines.value, "t_end")).alias("word"))
...     words = words.na.replace('', None)
...     words = words.na.drop()
...     words = words.withColumn('word', F.regexp_replace('word', r'http\S+', ''))
...     words = words.withColumn('word', F.regexp_replace('word', '@\w+', ''))
...     words = words.withColumn('word', F.regexp_replace('word', '#', ''))
...     words = words.withColumn('word', F.regexp_replace('word', 'RT', ''))
...     words = words.withColumn('word', F.regexp_replace('word', ':', ''))
...     words = words.withColumn('word', F.regexp_replace('word', '[^a-z_A-Z ]', ''))
...     words = words.withColumn('word', F.regexp_replace('word', "##$$$123&&'?!!
__!','[^[:alnum:]'' '']", ' '))
...     return words
...
# load the streaming data from socket connection
>>> lines = session.readStream.format("socket").option("host","hadoop-
nn001.cs.okstate.edu").option("port", 5566).load()
2021-04-02 03:00:09,239 WARN sources.TextSocketSourceProvider: The socket source
should not be used for production applications! It does not support recovery.
# applying pre-processing
>>> words  = preprocessing(lines)
# group the tweets and filtering the counts for keyword using for tweets
>>> filtered = words.withColumn('word', explode(split(col('word'), ' '))) \
...             .groupBy('word') \
...             .count() \
...             .sort('count', ascending=False). \
...             filter((col('word').contains("NASA")) | (col(‘word').contains('Mars')))
# query mode of output.
>>> query = filtered.writeStream\
...                 .outputMode("complete")\
...                 .format("memory")\
...                 .queryName("counts")\
...                 .start()
```

2021-04-02 03:01:12,840 WARN streaming.StreamingQueryManager: Temporary checkpoint location created which is deleted normally when the query didn't fail: /tmp/temporary-f9ecf06e-07ce-4406-8147-6e5d5ee127eb. If it's required to delete it under any circumstances, please set spark.sql.streaming.forceDeleteTempCheckpointLocation to true. Important to know deleting temp checkpoint folder is best effort.

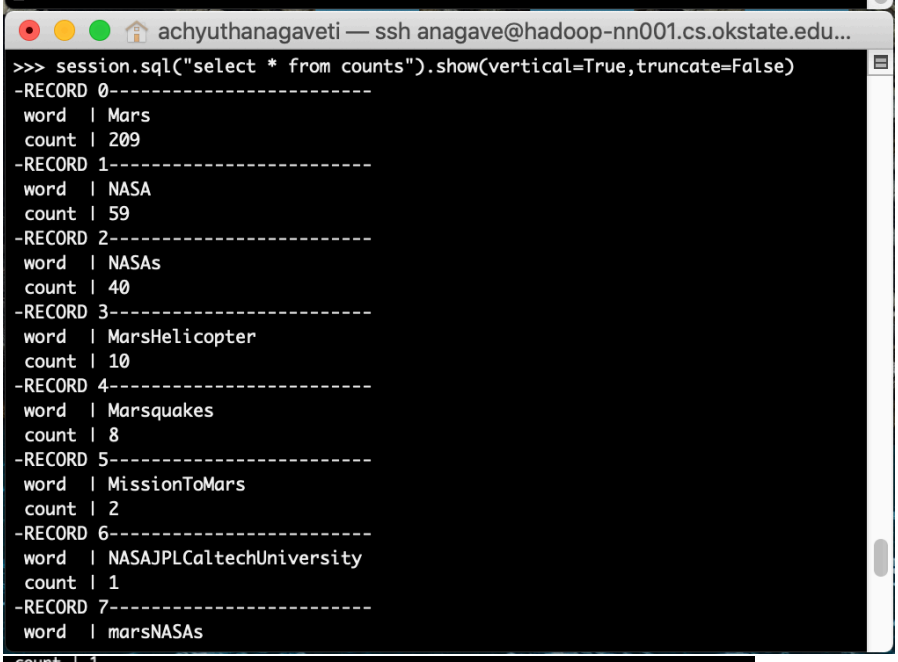
#the result is displayed through sql from data stream

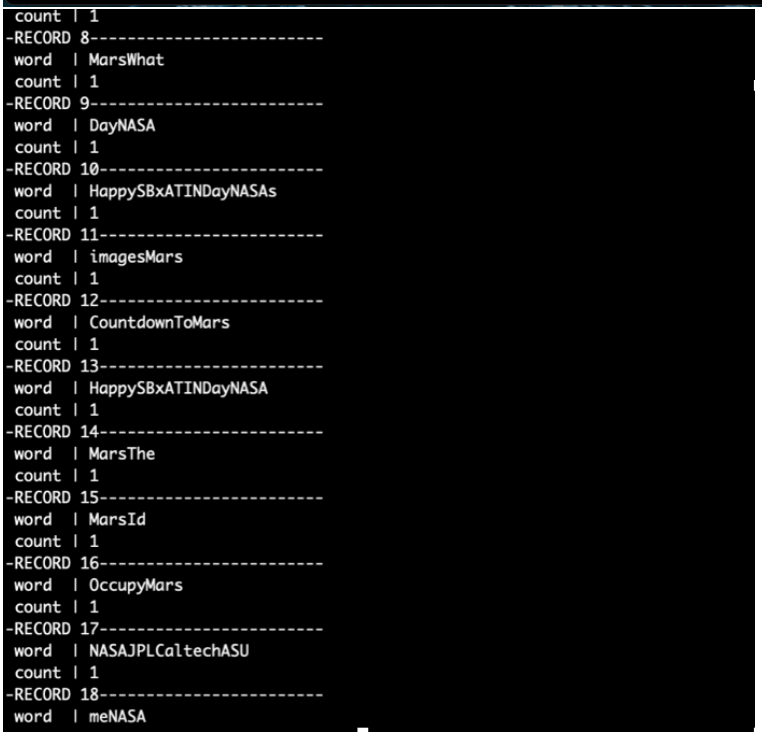
```
>>>session.sql("select * from counts").show(vertical=True,truncate=False)
```



Output:


achyuthanagaveti — ssh anagave@hadoop-nn001.cs.okstate.edu...
r NASA right n\xe2\x80\xa6'
b'Whenever You Call" is a really great song because it\'s a song that can only b
e done at that moment.\n\nREQUEST\xe2\x80\xa6 https://t.co/moKjjqlfTV'
b'Why does https://t.co/Gpk45qAO7G keep redirecting to a GitHub Pages domain?\n\
nRegardless, this particular page appea\xe2\x80\xa6 https://t.co/7Zo97JujNM'
b'RT @MYmilkteh: Hi @elonmusk have you considered Starlink for #Myanmar\nWe know
you wanna go to space or Mars and give them Internet all but\xe2\x80\xa6'
b'RT @MYmilkteh: Hi @elonmusk have you considered Starlink for #Myanmar\nWe know
you wanna go to space or Mars and give them Internet all but\xe2\x80\xa6'
b'RT @pabloxity: Nasa?? Grammy??? Awards?? Topic??ang?? SB19??? \xf0\x9f\x98\xb3
\xf0\x9f\x98\xb3\xf0\x9f\x98\xb3\xf0\x9f\x98\xb3\xf0\x9f\x98\xb3\n\nSB19WhatMV 7
Million\n@SB19Official\n#STANWORLD #SB19\n#HappySB19xATINDay\xe2\x80\xa6'
b'that virgo venus & scorpio mars got me like https://t.co/q0NlfNk7g4'
b"Yahoo News UK: New photos from Mars: NASA's Ingenuity helicopter stretches its
legs, while the Curiosity rover star\xe2\x80\xa6 https://t.co/Yf6y1QXdAv"
b'RT @NASA: Uranus gives off X-rays, astronomers find. \n\nThese @ChandraXray ob
servations may help scientists learn more about the intriguing\xe2\x80\xa6'
b'RT @astro_jaz: FROSTY SAND DUNES ON MARS!! \n\nImage credit: NASA/JPL-Caltech
/University of Arizona https://t.co/uHOnkoR2vU'
b'@Veritatis2021 @SylviaDeeDee Conspiracy theorists and under-educated Boltnista
s tend to believe such silly bs. Mean\xe2\x80\xa6 https://t.co/W1vjCfA5GH'
b'Technology Research Center: Water Suvival Equipment Tested By NASA At Texas A&
amp;M https://t.co/aNA34O0jdh https://t.co/erO4DO1ziB'


achyuthanagaveti — ssh anagave@hadoop-nn001.cs.okstate.edu...
>>> session.sql("select * from counts").show(vertical=True,truncate=False)
-RECORD 0-------------------------
word | Mars
count | 209
-RECORD 1-------------------------
word | NASA
count | 59
-RECORD 2-------------------------
word | NASAs
count | 40
-RECORD 3-------------------------
word | MarsHelicopter
count | 10
-RECORD 4-------------------------
word | Marsquakes
count | 8
-RECORD 5-------------------------
word | MissionToMars
count | 2
-RECORD 6-------------------------
word | NASAJPLCaltechUniversity
count | 1
-RECORD 7-------------------------
word | marsNASAs


count | 1
-RECORD 8-------------------------
word | MarsWhat
count | 1
-RECORD 9-------------------------
word | DayNASA
count | 1
-RECORD 10------------------------
word | HappySBxATINDayNASAs
count | 1
-RECORD 11------------------------
word | imagesMars
count | 1
-RECORD 12------------------------
word | CountdownToMars
count | 1
-RECORD 13------------------------
word | HappySBxATINDayNASA
count | 1
-RECORD 14------------------------
word | MarsThe
count | 1
-RECORD 15------------------------
word | MarsId
count | 1
-RECORD 16------------------------
word | OccupyMars
count | 1
-RECORD 17------------------------
word | NASAJPLCaltechASU
count | 1
-RECORD 18------------------------
word | meNASA

stoping the query

>>>query.stop()