Q1

1a)

First download the file from my link (same file as stated in the homework) using wget:

wget https://fastupload.io/OcWQo06rvNYK/7jn9QZF7OXg2ova/9AqGQyMegGMn6/Crime_Incidents_in_2013.csv

and put it in hdfs:

`hdfs dfs –put Crime_Incidents_in_2013.csv ./input/`

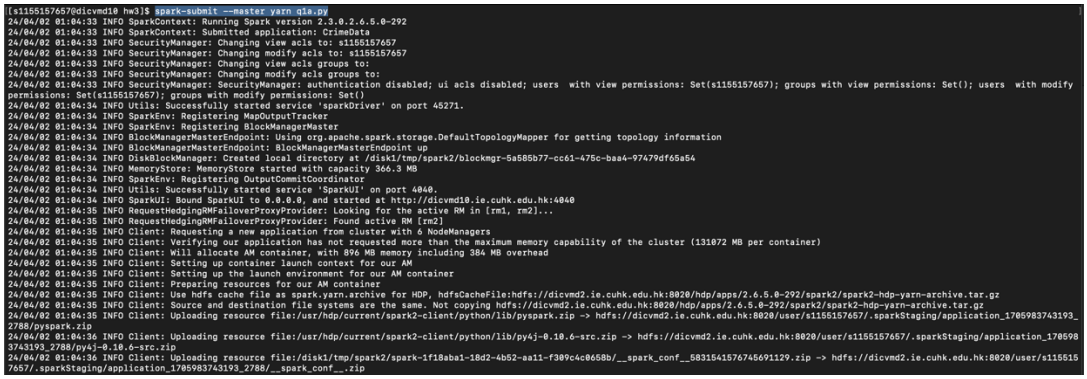
Code (q1a.py):

```python
from pyspark.sql import SparkSession

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

df = spark.read.csv("./input/Crime_Incidents_in_2013.csv", header=True, inferSchema=True)

columns_of_interest = ["CCN", "REPORT_DAT", "OFFENSE", "METHOD", "END_DATE", "DISTRICT"]
df_selected = df.select(columns_of_interest)

df_filtered = df_selected.dropna(subset=columns_of_interest)

df_filtered.write.csv("./output/crime_incidents_filtered.csv", header=True)

spark.stop()
```

In the code, filter the column and drop the row with null value in those columns.

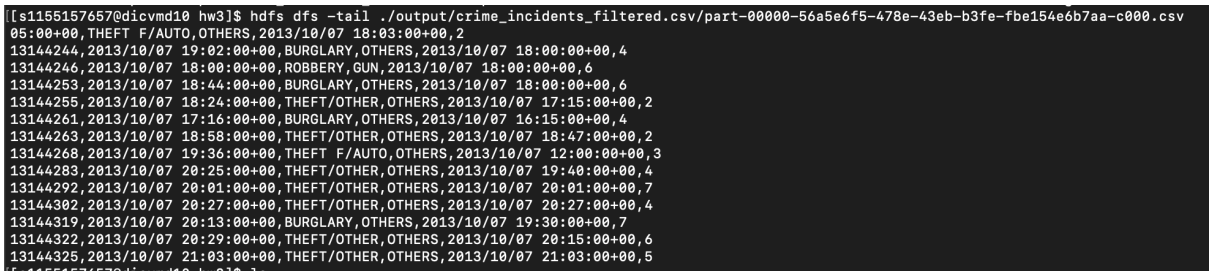
Then, we can use the command:

`spark–submit ––master yarn q1a.py`



See the result (partial):

```
hdfs dfs -tail ./output/crime_incidents_filtered.csv/part-00000-56a5e6f5-478e-43eb-b3fe-fbe154e6b7aa-c000.csv
```



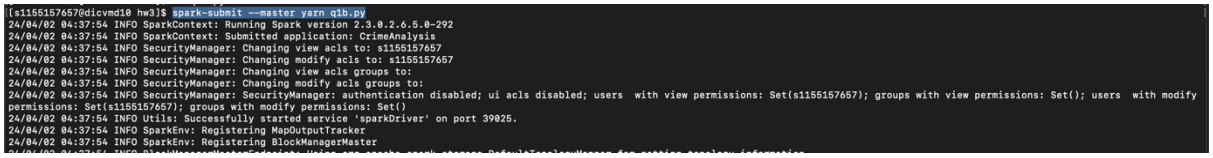
1b)

Code (q1b.py):

```python
from pyspark.sql import SparkSession

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

df = spark.read.csv("./input/Crime_Incidents_in_2013.csv", header=True, inferSchema=True)

df.createOrReplaceTempView("crime_data")

offense_counts_sql = spark.sql("""
SELECT OFFENSE, COUNT(*) as count
FROM crime_data
GROUP BY OFFENSE
ORDER BY count DESC
""")

offense_counts_sql.show()

shift_counts_sql = spark.sql("""
SELECT SHIFT, COUNT(*) as count
FROM crime_data
GROUP BY SHIFT
ORDER BY count DESC
LIMIT 1
""")

shift_counts_sql.show()

spark.stop()
```

Run the code:

```
spark-submit --master yarn q1b.py
```

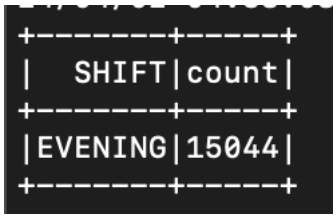


Output:

group by offense:

```
+--------------------+-----+
|            OFFENSE|count|
+--------------------+-----+
|         THEFT/OTHER|12875|
|       THEFT F/AUTO|10159|
|             ROBBERY| 3982|
|            BURGLARY| 3357|
| MOTOR VEHICLE THEFT| 2664|
|ASSAULT W/DANGERO...| 2396|
|           SEX ABUSE|  299|
|            HOMICIDE|  104|
|               ARSON|   35|
+--------------------+-----+
```

Time of most crime occur at evening:



SHIFT	count
EVENING	15044

1c)

First download all other files, unzip it:
`wget https://file.io/Pn91mMIkAfLm.zip`

`unzip Pn91mMIkAfLm.zip`

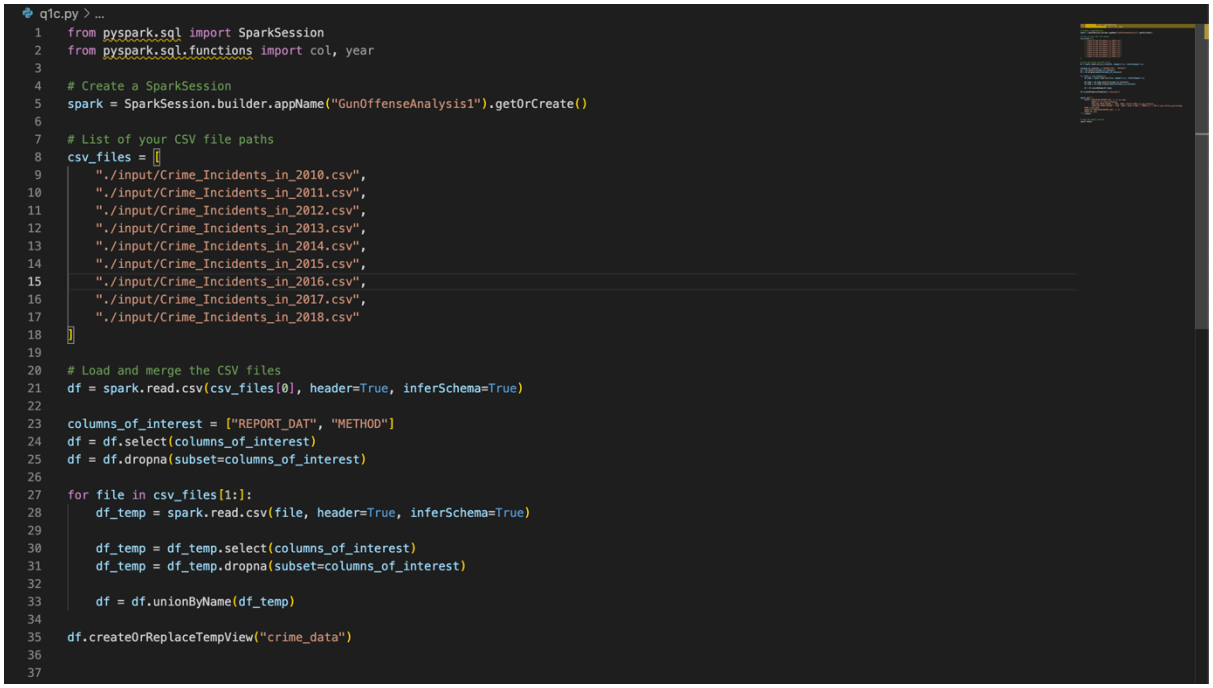
Then put it in hdfs:

`hdfs dfs –put Crime_Incidents_in_201* ./input/`

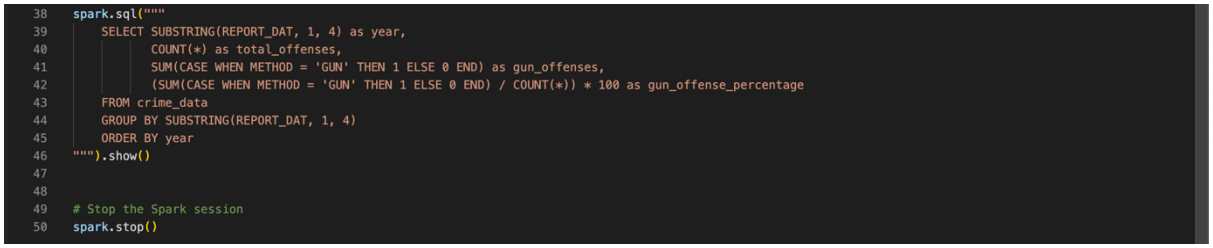
Code (q1c.py):

Explanation:

Get and process each csv and concatenate them by union. Then I used SQL to parse the year REPORT_DAT to select the year, count all the record and the record with method=GUN. Group by year and order in ascending.



```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, year

# Create a SparkSession
spark = SparkSession.builder.appName("GunOffenseAnalysis1").getOrCreate()

# List of your CSV file paths
csv_files = [
    "./input/Crime_Incidents_in_2010.csv",
    "./input/Crime_Incidents_in_2011.csv",
    "./input/Crime_Incidents_in_2012.csv",
    "./input/Crime_Incidents_in_2013.csv",
    "./input/Crime_Incidents_in_2014.csv",
    "./input/Crime_Incidents_in_2015.csv",
    "./input/Crime_Incidents_in_2016.csv",
    "./input/Crime_Incidents_in_2017.csv",
    "./input/Crime_Incidents_in_2018.csv"
]

# Load and merge the CSV files
df = spark.read.csv(csv_files[0], header=True, inferSchema=True)

columns_of_interest = ["REPORT_DAT", "METHOD"]
df = df.select(columns_of_interest)
df = df.dropna(subset=columns_of_interest)

for file in csv_files[1:]:
    df_temp = spark.read.csv(file, header=True, inferSchema=True)

    df_temp = df_temp.select(columns_of_interest)
    df_temp = df_temp.dropna(subset=columns_of_interest)

    df = df.unionByName(df_temp)

df.createOrReplaceTempView("crime_data")
```



```python
spark.sql("""
    SELECT SUBSTRING(REPORT_DAT, 1, 4) as year,
           COUNT(*) as total_offenses,
           SUM(CASE WHEN METHOD = 'GUN' THEN 1 ELSE 0 END) as gun_offenses,
           (SUM(CASE WHEN METHOD = 'GUN' THEN 1 ELSE 0 END) / COUNT(*)) * 100 as gun_offense_percentage
    FROM crime_data
    GROUP BY SUBSTRING(REPORT_DAT, 1, 4)
    ORDER BY year
""").show()


# Stop the Spark session
spark.stop()
```

Run the code:

`spark–submit ––master yarn q1c.py`

Output:

year	total_offenses	gun_offenses	gun_offense_percentage
2010	31675	2022	6.383583267561169
2011	33215	1861	5.602890260424507
2012	35270	2205	6.251772044230224
2013	35874	2201	6.135362658192562
2014	38404	1959	5.101031142589314
2015	37176	2186	5.88013772326232
2016	37199	2122	5.704454420817764
2017	33102	1582	4.779167421908042
2018	33831	1621	4.791463450681328
2019	24	0	0.0

Conclusion:

Obama's executive actions on gun control is on 2016. As we can see the crime percentage according to gun dropped from 5.7 in 2016 to 4.7 in 2017, around 17.5% from 2017.

Q3a

Install java:

sudo apt update

sudo apt install openjdk-11-jdk

Install zookeeper:

sudo apt-get install zookeeperd

Download kafka:

wget https://dlcdn.apache.org/kafka/3.7.0/kafka_2.13-3.7.0.tgz

```
polybuddi@kafka-3:~$ wget https://dlcdn.apache.org/kafka/3.7.0/kafka_2.13-3.7.0.tgz
--2024-04-02 17:26:57--  https://dlcdn.apache.org/kafka/3.7.0/kafka_2.13-3.7.0.tgz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 119028138 (114M) [application/x-gzip]
Saving to: 'kafka_2.13-3.7.0.tgz'

kafka_2.13-3.7.0.tgz  100%[=====================>] 113.51M   254MB/s    in 0.4s

2024-04-02 17:27:00 (254 MB/s) - 'kafka_2.13-3.7.0.tgz' saved [119028138/119028138]
```

Unzip it:

tar -xvzf kafka_2.13-3.7.0.tgz

set config/server.properties for all 3 ndoes:

for example for broker 1:

broker.id=1

listeners=PLAINTEXT://<kafka-broker-1-IP>:9092

log.dirs=/var/lib/kafka/logs

zookeeper.connect=<zookeeper-1-IP>:2181,<zookeeper-2-IP>:2181,<zookeeper-3-IP>:2181

```
############################# Socket Server Settings #############################

# The address the socket server listens on. If not configured, the host name will
be equal to the value of
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and p
ort 9092.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://10.182.0.5:9092
```

```
############################# Log Basics #############################

# A comma separated list of directories under which to store log files
log.dirs=/var/lib/kafka/logs

# The default number of log partitions per topic. More partitions allow greater
# parallelism for consumption, but this will also result in more files across
# the brokers.
num.partitions=2
```

```
############################# Internal Topic Settings  #############################
##
# The replication factor for the group metadata internal topics "__consumer_offset
s" and "__transaction_state"
# For anything other than development testing, a value greater than 1 is recommend
ed to ensure availability such as 3.
offsets.topic.replication.factor=2
transaction.state.log.replication.factor=2
transaction.state.log.min.isr=2
```

```
############################# Zookeeper #############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=10.182.0.5:2181,10.182.0.6:2181,10.182.0.7:2181
```

Then start kafka on each machine:
bin/kafka-server-start.sh config/server.properties

Make repo for logs:

sudo mkdir /var/lib/kafka/

sudo mkdir /var/lib/kafka/logs

Start brokers:

sudo bin/kafka-server-start.sh config/server.properties

(for broker3 example:)

```
[2024-04-02 18:08:06,487] INFO [KafkaServer id=3] started (kafka.server.KafkaServer)
[2024-04-02 18:08:06,640] INFO [zk-broker-3-to-controller-forwarding-channel-manager]:
 Recorded new controller, from now on will use node 10.182.0.7:9092 (id: 3 rack: null)
 (kafka.server.NodeToControllerRequestThread)
[2024-04-02 18:08:06,664] INFO [zk-broker-3-to-controller-alter-partition-channel-mana
ger]: Recorded new controller, from now on will use node 10.182.0.7:9092 (id: 3 rack:
null) (kafka.server.NodeToControllerRequestThread)
```

Then do what the question require:

sudo bin/kafka-topics.sh --create --topic my-test-topic --bootstrap-server 10.182.0.5:9092 --replication-factor 2 --partitions 2

```
polybuddi@kafka-1:~/kafka_2.13-3.7.0$ sudo bin/kafka-topics.sh --create --topic my-test-topic --bootstrap-server 10.182.0.5:9092 --replication-factor 2 --partitions 2
Created topic my-test-topic.
```

send a test message:

sudo bin/kafka-console-producer.sh --broker-list 10.182.0.5:9092 --topic my-test-topic

```
polybuddi@kafka-1:~/kafka_2.13-3.7.0$ sudo bin/kafka-console-producer.sh --broker-list 10.182.0.5:9092 --topic my-test-topic
>my test message
```

sudo bin/kafka-console-consumer.sh --bootstrap-server 10.182.0.5:9092 --topic my-test-topic --from-beginning

```
polybuddi@kafka-1:~/kafka_2.13-3.7.0$ sudo bin/kafka-console-consumer.sh --bootstrap-server 10.182.0.5:9092 --topic my-test-topic --from-beginning
my test message

^CProcessed a total of 1 messages
```

To validate that there's a partition of 2 and replication of 2 using another machine.

```
polybuddi@kafka-2:~/kafka_2.13-3.7.0$ sudo bin/kafka-console-consumer.sh --bootstrap-server 10.182.0.5:9092 --t
opic my-test-topic --from-beginning
my test message
^CProcessed a total of 1 messages
polybuddi@kafka-2:~/kafka_2.13-3.7.0$ sudo bin/kafka-console-consumer.sh --bootstrap-server 10.182.0.7:9092 --t
opic my-test-topic --from-beginning
my test message
^CProcessed a total of 1 messages
```

Q4

Download the dataset:
`wget https://www.dropbox.com/s/jdck5tip9v4tzfw/new_tweets.txt`

Create a topic in DIC:

/usr/hdp/2.6.5.0-292/kafka/bin/kafka-topics.sh --create --zookeeper dicvmd7.ie.cuhk.edu.hk:2181 --replication-factor 2 -partitions 2 --topic bitcoin-1155157657-1

```
[[s1155157657@dicvmd10 hw3]$ /usr/hdp/2.6.5.0-292/kafka/bin/kafka-topics.sh --create --zookeeper dicvm
d7.ie.cuhk.edu.hk:2181 --replication-factor 2 -partitions 2 --topic bitcoin-1155157657-1
Created topic "bitcoin-1155157657-1".
```

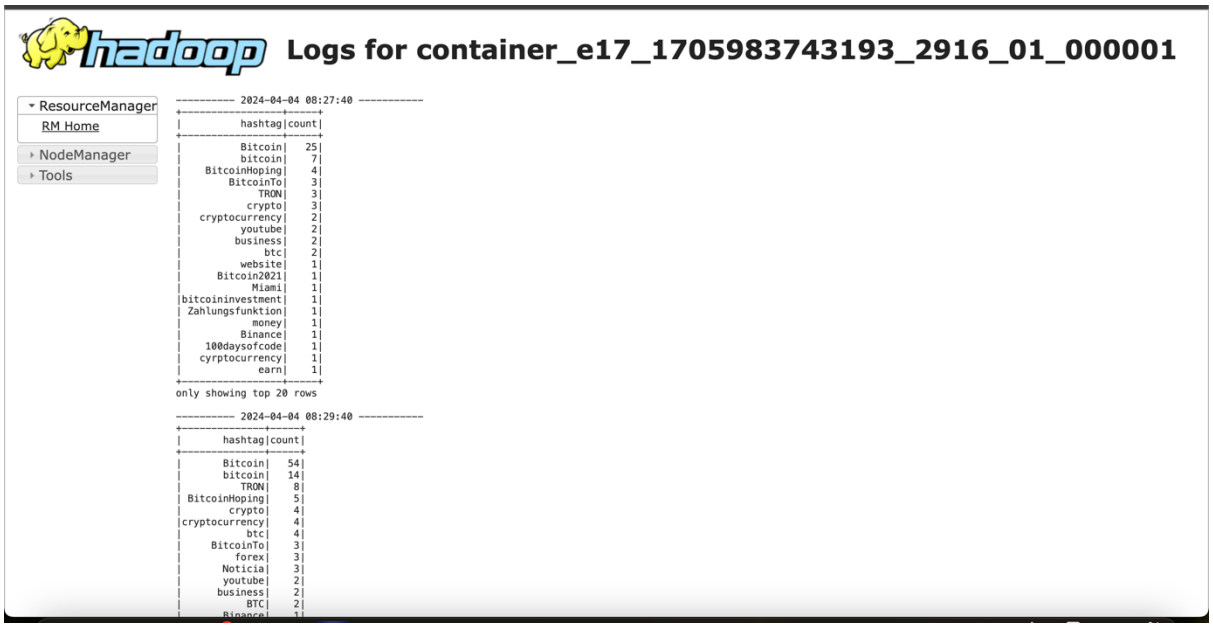
Then use the producer:

```
[[s1155157657@dicvmd10 hw3]$ /usr/hdp/2.6.5.0-292/kafka/bin/kafka-topics.sh --describe --topic bitcoin-1155157657 --zookeeper dicvmd7.ie.cuhk.edu.hk:218
Topic:bitcoin-1155157657        PartitionCount:2        ReplicationFactor:2     Configs:
        Topic: bitcoin-1155157657       Partition: 0    Leader: 1004    Replicas: 1004,1006     Isr: 1004,1006
        Topic: bitcoin-1155157657       Partition: 1    Leader: 1005    Replicas: 1005,1003     Isr: 1005,1003
```

Run the consumer:

spark-submit --master yarn --deploy-mode cluster --packages org.apache.spark:spark-streaming-kafka-0-8_2.11:2.3.0 kafka_consumer.py

Some output example:



some other examples:

```
---------- 2024-04-04 08:39:40 ----------
+---------------+-----+
|        hashtag|count|
+---------------+-----+
|        Bitcoin|   69|
|        bitcoin|   15|
|           TRON|    6|
|     BitcoinTo|    4|
|    blockchain|    3|
| BitcoinHoping|    3|
|       Ethereum|    3|
|cryptocurrency|    2|
|        Binance|    2|
|        Cardano|    2|
|        binance|    2|
|         amazon|    2|
|            ETH|    2|
|        infosec|    2|
|    technology|    2|
|        BITCOIN|    2|
|         Crypto|    2|
|         google|    2|
|       business|    2|
|            BTC|    2|
+---------------+-----+
only showing top 20 rows

---------- 2024-04-04 08:41:40 ----------
+---------------+-----+
|        hashtag|count|
+---------------+-----+
|        Bitcoin|   57|
|        bitcoin|   15|
|           TRON|    7|
| BitcoinHoping|    5|
|     BitcoinTo|    4|
|    blockchain|    4|
|           nfts|    3|
|cryptocurrency|    3|
|         crypto|    3|
|       Ethereum|    2|
|  NFTCommunity|    2|
|            nft|    2|
|            BTC|    2|
|        binance|    2|
|         Crypto|    2|
|         google|    1|
|         c34322|    1|
|            NFT|    1|
```

Code (kafka_producer.py):

kafka_producer.py > random_sleep

```python
import os
import random
import time
from datetime import datetime

def convert_to_seconds(ts):
    dt_obj = datetime.strptime(ts, '%Y-%m-%d %H:%M:%S')
    return int(time.mktime(dt_obj.timetuple()))

# The sleep time should be proportional to the difference in timestamps of the tweets
def random_sleep(current_ts, last_ts):
    if last_ts is None:
        sleep_time = random.randint(1, 4)
    else:
        # Calculate the difference
        sleep_time = current_ts - last_ts

    sleep_time = max(0.1, sleep_time)
    time.sleep(sleep_time)

def main():
    last_ts = None
    with open('new_tweets.txt') as f:
        for line in f:

            # split the text and timestamp
            parts = line.rstrip().rsplit(',', 1)
            text = parts[0]
            ts = parts[-1]

            ts = convert_to_seconds(ts)

            cmd = 'echo "' + text + '" | /usr/hdp/2.6.5.0-292/kafka/bin/kafka-console-producer.sh --broker-list

            print(cmd)
            os.system(cmd)


            if last_ts is not None:
                random_sleep(ts, last_ts)

            # Update the last timestamp
            last_ts = ts


if __name__ == '__main__':
    main()
```

Explain:

I do data processing to extract the text part and the time, then I mimic the time of tweets sent to the broker by adding some random noise and guarantee that it must be larger or equal to the previous tweet. Then echo to the broker.

Code (kafka_consumer.py):

kafka_consumer.py > ...

```python
from pyspark.streaming.kafka import KafkaUtils
from pyspark import SparkConf, SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import Row, SQLContext
import sys
import json
import time
import re

def process_rdd(time, rdd):
    print("---------- %s ----------" % str(time))
    try:
        # Convert RDD[String] to RDD[Row] to DataFrame
        sql_context = SQLContext(rdd.context)
        row_rdd = rdd.map(lambda w: Row(hashtag=w[0], count=w[1]))
        hashtags_df = sql_context.createDataFrame(row_rdd)

        # Register as table
        hashtags_df.registerTempTable("hashtags")

        # Get top 30 hashtags from the table using SQL and print them
        top_hashtags = sql_context.sql(
            "SELECT hashtag, count FROM hashtags ORDER BY count DESC LIMIT 30"
        )
        top_hashtags.show()
    except:
        e = sys.exc_info()
        print("Error: ", e)

def extract_hashtags(text):
    # Extract hashtags from text
    return re.findall(r"#(\w+)", text)

if __name__ == '__main__':
    sc = SparkContext(appName="KafkaHashtagCount")
    sc.setLogLevel("WARN")

    ssc = StreamingContext(sc, 2)  # 2-second batch interval
    ssc.checkpoint('./checkpoint123')  # Set checkpoint directory

    # Create Kafka stream
    kafkaStream = KafkaUtils.createStream(ssc, 'dicvmd7.ie.cuhk.edu.hk:2181', 'test', {'bitcoin-1155157657-1':

    # Extract messages from the stream
    lines = kafkaStream.map(lambda x: x[1])

    # Extract hashtags and count them
    hashtags = lines.flatMap(lambda line: extract_hashtags(line))
    hashtag_counts = hashtags.map(lambda hashtag: (hashtag, 1)).reduceByKeyAndWindow(
        lambda x, y: x + y,
        lambda x, y: x - y,
        300,  # Window length of 300 seconds (5 minutes)
        120   # Slide interval of 120 seconds (2 minutes)
    )

    # Process each RDD generated in each interval
    hashtag_counts.foreachRDD(process_rdd)

    ssc.start()  # Start the computation
    ssc.awaitTermination()  # Wait for the computation to terminate
```

Explain: read the message from broker, extract the hashtags using regex and map it using (text, 1). Then use sliding window of 5 min window and 2 min interval to group the data wanted.

Q5

Command to submit job (`application_1705983743193_3001`):

```
spark-submit --master yarn --deploy-mode cluster --packages
org.apache.spark:spark-sql-kafka-0-10_2.11:2.3.0 q5.py
```

(I changed the producer in this question to make the encoding ascii. Others are the same)

Producer:

kafka_producer.py > main

```python
import os
import random
import time
from datetime import datetime

def convert_to_seconds(ts):
    dt_obj = datetime.strptime(ts, '%Y-%m-%d %H:%M:%S')
    return int(time.mktime(dt_obj.timetuple()))

# The sleep time should be proportional to the difference in timestamps of the tweets
def random_sleep(current_ts, last_ts):
    if last_ts is None:
        sleep_time = random.randint(1, 4)
    else:
        # Calculate the difference
        sleep_time = current_ts - last_ts

    sleep_time = max(0.1, sleep_time)
    time.sleep(sleep_time)

def main():
    last_ts = None
    with open('new_tweets.txt') as f:
        for line in f:

            line = line.decode('ascii', 'ignore')

            # split the text and timestamp
            parts = line.rstrip().rsplit(',', 1)
            text = parts[0]
            ts = parts[-1]

            ts = convert_to_seconds(ts)

            cmd = 'echo "' + text + '" | /usr/hdp/2.6.5.0-292/kafka/bin/kafka-console-producer.sh --broker-list

            print(cmd)
            os.system(cmd)


            if last_ts is not None:
                random_sleep(ts, last_ts)

            # Update the last timestamp
            last_ts = ts


if __name__ == '__main__':
    main()
```

Code:

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, split, window, col
from pyspark.sql.types import StringType
import time

if __name__ == '__main__':
    # Initialize SparkSession
    spark = SparkSession.builder \
        .appName("StructuredKafkaHashtagCount_12345") \
        .getOrCreate()

    # Set log level
    spark.sparkContext.setLogLevel("WARN")

    # Read messages from Kafka
    df = spark \
        .readStream \
        .format("kafka") \
        .option("kafka.bootstrap.servers", "dicvmd7.ie.cuhk.edu.hk:6667") \
        .option("subscribe", "bitcoin-1155157657-1") \
        .load()

    messages = df.selectExpr("CAST(value AS STRING)", "CAST(timestamp AS TIMESTAMP)")

    hashtags = messages.withColumn("word", explode(split(col("value"), " "))) \
    .filter(col("word").rlike("#\\w+")) \
    .select("word", "timestamp")

    # Group by hashtags and window
    hashtag_counts = hashtags.groupBy(
        window(hashtags.timestamp, "300 seconds", "120 seconds"),
        hashtags.word
    ).count().orderBy(col("window.start").desc(), col("count").desc())
```

```python


    # writes the output to memory sink
    query = hashtag_counts \
        .writeStream \
        .outputMode("complete") \
        .queryName("hashtags") \
        .format("memory") \
        .option("checkpointLocation", "./checkpointanpojbfhhdakjfb") \
        .trigger(processingTime='2 minutes') \
        .start()

    # Use this function to periodically query the top hashtags for the top timestamps
    while query.isActive:
        spark.sql("SELECT window, word, count FROM hashtags ORDER BY window.start DESC, count DESC LIMIT 30

        time.sleep(120)  # sleep for 2 minutes before updating the console again


    query.awaitTermination()
```

Explain:

Similar to the one in q4, I try to mimic the effect of the sliding window by taking the word and timestamp of each word in a line. Then use a sql query to get the top 30 count of the recent timeframe.

Output:

```
+------------------------------------------+-----------------+-----+
|window                                    |word             |count|
+------------------------------------------+-----------------+-----+
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#Bitcoin         |16   |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#TRON            |4    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#BitcoinHoping   |2    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#BitcoinTo       |2    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#btc             |2    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#cryptocurrency  |2    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#regulatory      |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|ignore.#bitcoin  |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#Crypto          |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#NewYork         |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#trading         |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#future          |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#block           |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#Noticia         |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#BTC             |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#Zahlungsfunktion|1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#forextrading    |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#cryptocurrency, |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#bitcoin         |1    |
|[2024-04-05 12:40:00, 2024-04-05 12:45:00]|#crypto          |1    |
+------------------------------------------+-----------------+-----+
only showing top 20 rows

+------------------------------------------+------------------+-----+
|window                                    |word              |count|
+------------------------------------------+------------------+-----+
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#Bitcoin          |26   |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#bitcoin          |5    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#TRON             |3    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#Noticia          |3    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#btc              |2    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#Ethereum         |2    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#cryptocurrency   |2    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#Sport            |1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|network.#cryptonews|1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#ShibaSwap        |1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#pit              |1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#BitcoinTo        |1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#investing        |1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#giveaway         |1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#BitcoinThey      |1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#tether           |1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#BTC              |1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|market....#bitcoin|1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#LN               |1    |
|[2024-04-05 12:42:00, 2024-04-05 12:47:00]|#altseason        |1    |
+------------------------------------------+------------------+-----+
only showing top 20 rows
```

Some other:

```
+------------------------------------------+---------------+-----+
|window                                    |word           |count|
+------------------------------------------+---------------+-----+
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#Bitcoin       |20   |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#bitcoin       |6    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#TRON          |3    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#nftartist     |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#Dogeco        |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#Bitcoin.      |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#jobsearch     |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#wallstreetbets|1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#reddit        |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#BitcoinHoping |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#TRX           |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#crypto        |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#nftcommunity  |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#nftart        |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#Teletubbies   |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#eth           |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#xrp           |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#ballsdeepalts |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#yfi           |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#binance       |1    |
+------------------------------------------+---------------+-----+
only showing top 20 rows
```

```
+------------------------------------------+---------------------------+-----+
|window                                    |word                       |count|
+------------------------------------------+---------------------------+-----+
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#Bitcoin                   |21   |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#bitcoin                   |6    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#1SG                       |2    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#BTC                       |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#tron                      |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#Binance                   |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#BitcoinHoping             |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#altcoinsepeti             |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#TRON                      |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#giveaway                  |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#BitcoinTo                 |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#Bitcoin.                  |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#chz                       |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#ethereum                  |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#B                         |1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#Xapohttps://t.co/XHbsROcoye|1    |
|[2024-04-05 12:46:00, 2024-04-05 12:51:00]|#Airdrop                   |1    |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#Bitcoin                   |41   |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#bitcoin                   |12   |
|[2024-04-05 12:44:00, 2024-04-05 12:49:00]|#TRON                      |4    |
+------------------------------------------+---------------------------+-----+
only showing top 20 rows
```