

Q1

1a)

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

wget

https://fastupload.io/0cW0o06rvNYK/7jn9QZF70Xg2ova/9AqG0yMegGMn6/Crime_Incidents_in_2013.csv

and put it in hdfs:

hdfs dfs -put Crime_Incidents_in_2013.csv ./input/

Code (q1a.py):

```
q1a.py 1 X
q1a.py > ...
1 from pyspark.sql import SparkSession
2
3 spark = SparkSession.builder.appName("CrimeData").getOrCreate()
4
5 df = spark.read.csv("./input/Crime_Incidents_in_2013.csv", header=True, inferSchema=True)
6
7 columns_of_interest = ["CCN", "REPORT_DAT", "OFFENSE", "METHOD", "END_DATE", "DISTRICT"]
8 df_selected = df.select(columns_of_interest)
9
10 df_filtered = df_selected.dropna(subset=columns_of_interest)
11
12 df_filtered.write.csv("./output/crime_incidents_filtered.csv", header=True)
13
14 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

```
[s1155157657@dicvmd10 hw3]$ spark-submit --master yarn q1a.py
24/04/02 01:04:33 INFO SparkContext: Running Spark version 2.3.0
24/04/02 01:04:33 INFO SparkContext: Submitted application: CrimeData
24/04/02 01:04:33 INFO SecurityManager: Changing view acls to: s1155157657
24/04/02 01:04:33 INFO SecurityManager: Changing modify acls to: s1155157657
24/04/02 01:04:33 INFO SecurityManager: Changing view acls groups to:
24/04/02 01:04:33 INFO SecurityManager: Changing modify acls groups to:
24/04/02 01:04:33 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(s1155157657); groups with view permissions: Set(); users with modify permissions: Set(s1155157657); groups with modify permissions: Set()
24/04/02 01:04:34 INFO Utils: Successfully started service 'sparkDriver' on port 45271.
24/04/02 01:04:34 INFO SparkEnv: Registering MapOutputTracker
24/04/02 01:04:34 INFO SparkEnv: Registering BlockManagerMaster
24/04/02 01:04:34 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
24/04/02 01:04:34 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
24/04/02 01:04:34 INFO DiskBlockManager: Created local directory at /disk1/tmp/spark2/blockmgr-5a585b77-cc61-476c-baa4-97479df65a54
24/04/02 01:04:34 INFO MemoryStore: MemoryStore started with capacity 366.3 MB
24/04/02 01:04:34 INFO SparkEnv: Registering OutputCommitCoordinator
24/04/02 01:04:34 INFO Utils: Successfully started service 'SparkUI' on port 4040.
24/04/02 01:04:34 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://dicvmd10.ie.cuhk.edu.hk:4040
24/04/02 01:04:35 INFO RequestHedgingRMFailoverProxyProvider: Looking for the active RM in [rm1, rm2]...
24/04/02 01:04:35 INFO RequestHedgingRMFailoverProxyProvider: Found active RM [rm2]
24/04/02 01:04:35 INFO Client: Requesting a new application from cluster with 6 NodeManagers
24/04/02 01:04:35 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster (131872 MB per container)
24/04/02 01:04:35 INFO Client: Will allocate AM container, with 896 MB memory including 384 MB overhead
24/04/02 01:04:35 INFO Client: Setting up container launch context for our AM
24/04/02 01:04:35 INFO Client: Setting up the launch environment for our AM container
24/04/02 01:04:35 INFO Client: Preparing resources for our AM container
24/04/02 01:04:35 INFO Client: Use hdfs.cache file as spark.yarn.archive for HDP, hdfs.cache file as spark.yarn.archive for HDP, hdfs.cache file as spark.yarn.archive for HDP
24/04/02 01:04:35 INFO Client: Source and destination file systems are the same. Not copying hdfs://dicvmd2.ie.cuhk.edu.hk:8020/hdp/apps/2.6.5.0-292/spark2/spark2-hdp-yarn-archive.tar.gz
24/04/02 01:04:35 INFO Client: Unloading resource file /usr/hdp/current/spark2-client/python/lib/pyspark.zip -> hdfs://dicvmd2.ie.cuhk.edu.hk:8020/user/s1155157657/.sparkStaging/application_1705983743193_2788/pyspark.zip
24/04/02 01:04:35 INFO Client: Unloading resource file /usr/hdp/current/spark2-client/python/lib/py4j-0.10.6-src.zip -> hdfs://dicvmd2.ie.cuhk.edu.hk:8020/user/s1155157657/.sparkStaging/application_1705983743193_2788/py4j-0.10.6-src.zip
24/04/02 01:04:35 INFO Client: Unloading resource file /disk1/tmp/spark2/spark-1f18ab1-18d2-4b52-a511-f309c4c8658b/_spark_conf_5831541576745691129.zip -> hdfs://dicvmd2.ie.cuhk.edu.hk:8020/user/s1155157657/.sparkStaging/application_1705983743193_2788/_spark_conf_.zip
```

See the result (partial):

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

```
[s1155157657@dicvmd10 hw3]$ hdfs dfs -tail ./output/crime_incidents_filtered.csv/part-00000-56a5e6f5-478e-43eb-b3fe-fbe154e6b7aa-c000.csv
05:00+00,THEFT F/AUTO,OTHERS,2013/10/07 18:03:00+00,2
13144244,2013/10/07 19:02:00+00,BURGLARY,OTHERS,2013/10/07 18:00:00+00,4
13144246,2013/10/07 18:00:00+00,ROBBERY,GUN,2013/10/07 18:00:00+00,6
13144253,2013/10/07 18:44:00+00,BURGLARY,OTHERS,2013/10/07 18:00:00+00,6
13144255,2013/10/07 18:24:00+00,THEFT/OTHER,OTHERS,2013/10/07 17:15:00+00,2
13144261,2013/10/07 17:16:00+00,BURGLARY,OTHERS,2013/10/07 16:15:00+00,4
13144263,2013/10/07 18:58:00+00,THEFT/OTHER,OTHERS,2013/10/07 18:47:00+00,2
13144268,2013/10/07 19:36:00+00,THEFT F/AUTO,OTHERS,2013/10/07 12:00:00+00,3
13144283,2013/10/07 20:25:00+00,THEFT/OTHER,OTHERS,2013/10/07 19:40:00+00,4
13144292,2013/10/07 20:01:00+00,THEFT/OTHER,OTHERS,2013/10/07 20:01:00+00,7
13144302,2013/10/07 20:27:00+00,THEFT/OTHER,OTHERS,2013/10/07 20:27:00+00,4
13144319,2013/10/07 20:13:00+00,BURGLARY,OTHERS,2013/10/07 19:30:00+00,7
13144322,2013/10/07 20:29:00+00,THEFT/OTHER,OTHERS,2013/10/07 20:15:00+00,6
13144325,2013/10/07 21:03:00+00,THEFT/OTHER,OTHERS,2013/10/07 21:03:00+00,5
```

1b)

Code (q1b.py):

```
q1b_offense.py > ...
1  from pyspark.sql import SparkSession
2
3  spark = SparkSession.builder.appName("CrimeAnalysis").getOrCreate()
4
5  df = spark.read.csv("./input/Crime_Incidents_in_2013.csv", header=True, inferSchema=True)
6
7  df.createOrReplaceTempView("crime_data")
8
9  offense_counts_sql = spark.sql("""
10 SELECT OFFENSE, COUNT(*) as count
11 FROM crime_data
12 GROUP BY OFFENSE
13 ORDER BY count DESC
14 """)
15
16 offense_counts_sql.show()
17
18 shift_counts_sql = spark.sql("""
19 SELECT SHIFT, COUNT(*) as count
20 FROM crime_data
21 GROUP BY SHIFT
22 ORDER BY count DESC
23 LIMIT 1
24 """)
25
26 shift_counts_sql.show()
27
28 spark.stop()
```

Run the code:

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

```
[s1155157657@mdc10 hwc]$ spark-submit --master yarn q1b.py
24/04/02 04:37:54 INFO SparkContext: Running Spark version 2.3.0-2.6.5.0-292
24/04/02 04:37:54 INFO SparkContext: Submitted application: CrimeAnalysis
24/04/02 04:37:54 INFO SecurityManager: Changing view acls to: s1155157657
24/04/02 04:37:54 INFO SecurityManager: Changing modify acls to: s1155157657
24/04/02 04:37:54 INFO SecurityManager: Changing view acls groups to:
24/04/02 04:37:54 INFO SecurityManager: Changing modify acls groups to:
24/04/02 04:37:54 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(s1155157657); groups with view permissions: Set(); users with modify
permissions: Set(s1155157657); groups with modify permissions: Set()
24/04/02 04:37:54 INFO Utils: Successfully started service 'sparkDriver' on port 39025.
24/04/02 04:37:54 INFO SparkEnv: Registering MapOutputTracker
24/04/02 04:37:54 INFO SparkEnv: Registering BlockManagerMaster
```

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.

```
q1c.py > ...
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import col, year
3
4 # Create a SparkSession
5 spark = SparkSession.builder.appName("GunOffenseAnalysis1").getOrCreate()
6
7 # List of your CSV file paths
8 csv_files = [
9     "./input/Crime_Incidents_in_2010.csv",
10    "./input/Crime_Incidents_in_2011.csv",
11    "./input/Crime_Incidents_in_2012.csv",
12    "./input/Crime_Incidents_in_2013.csv",
13    "./input/Crime_Incidents_in_2014.csv",
14    "./input/Crime_Incidents_in_2015.csv",
15    "./input/Crime_Incidents_in_2016.csv",
16    "./input/Crime_Incidents_in_2017.csv",
17    "./input/Crime_Incidents_in_2018.csv"
18 ]
19
20 # Load and merge the CSV files
21 df = spark.read.csv(csv_files[0], header=True, inferSchema=True)
22
23 columns_of_interest = ["REPORT_DAT", "METHOD"]
24 df = df.select(columns_of_interest)
25 df = df.dropna(subset=columns_of_interest)
26
27 for file in csv_files[1:]:
28     df_temp = spark.read.csv(file, header=True, inferSchema=True)
29
30     df_temp = df_temp.select(columns_of_interest)
31     df_temp = df_temp.dropna(subset=columns_of_interest)
32
33     df = df.unionByName(df_temp)
34
35 df.createOrReplaceTempView("crime_data")
36
37
38 spark.sql("""
39     SELECT SUBSTRING(REPORT_DAT, 1, 4) as year,
40            COUNT(*) as total_offenses,
41            SUM(CASE WHEN METHOD = 'GUN' THEN 1 ELSE 0 END) as gun_offenses,
42            (SUM(CASE WHEN METHOD = 'GUN' THEN 1 ELSE 0 END) / COUNT(*)) * 100 as gun_offense_percentage
43     FROM crime_data
44     GROUP BY SUBSTRING(REPORT_DAT, 1, 4)
45     ORDER BY year
46 """).show()
47
48 # Stop the Spark session
49 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-manager]:
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:

```
Created topic "bitcoin-1155157657-1".  
[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:2181  
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:

The screenshot displays the Hadoop web interface with the title "Logs for container_e17_1705983743193_2916_01_000001". On the left, a sidebar contains links for "ResourceManager", "RM Home", "NodeManager", and "Tools". The main content area shows two log snippets. The first snippet, dated "2024-04-04 08:27:40", lists hashtag counts for various topics. The second snippet, dated "2024-04-04 08:29:40", shows the top 20 rows of the same data.

hashtag	count
Bitcoin	25
Bitcoin	7
BitcoinHoping	4
BitcoinTo	3
TRON	3
crypto	3
cryptocurrency	2
youtube	2
business	2
btci	2
website	1
Bitcoin2021	1
Risai	1
bitcoininvestment	1
Zahlungsfunktion	1
money	1
Binance	1
100daysofcode	1
cryptocurrency	1
earn	1

only showing top 20 rows

hashtag	count
Bitcoin	54
Bitcoin	14
TRON	8
BitcoinHoping	5
crypto	4
cryptocurrency	4
btci	4
BitcoinTo	3
forex	3
Noticia	3
youtube	2
business	2
BTIC	2
Binance	1

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
1  import os
2  import random
3  import time
4  from datetime import datetime
5
6  def convert_to_seconds(ts):
7      dt_obj = datetime.strptime(ts, '%Y-%m-%d %H:%M:%S')
8      return int(time.mktime(dt_obj.timetuple()))
9
10 # The sleep time should be proportional to the difference in timestamps of the tweets
11 def random_sleep(current_ts, last_ts):
12     if last_ts is None:
13         sleep_time = random.randint(1, 4)
14     else:
15         # Calculate the difference
16         sleep_time = current_ts - last_ts
17
18     sleep_time = max(0.1, sleep_time)
19     time.sleep(sleep_time)
20
21 def main():
22     last_ts = None
23     with open('new_tweets.txt') as f:
24         for line in f:
25
26             # split the text and timestamp
27             parts = line.rstrip().rsplit(',', 1)
28             text = parts[0]
29             ts = parts[-1]
30
31             ts = convert_to_seconds(ts)
32
33             cmd = 'echo "' + text + '" | /usr/hdp/2.6.5.0-292/kafka/bin/kafka-console-producer.sh --broker-list
34
35             print(cmd)
36             os.system(cmd)
37
38
39             if last_ts is not None:
40                 random_sleep(ts, last_ts)
41
42             # Update the last timestamp
43             last_ts = ts
44
45
46 if __name__ == '__main__':
47     main()
48
```

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 > ...
1  from pyspark.streaming.kafka import KafkaUtils
2  from pyspark import SparkConf, SparkContext
3  from pyspark.streaming import StreamingContext
4  from pyspark.sql import Row, SQLContext
5  import sys
6  import json
7  import time
8  import re
9
10 def process_rdd(time, rdd):
11     print("----- %s -----" % str(time))
12     try:
13         # Convert RDD[String] to RDD[Row] to DataFrame
14         sql_context = SQLContext(rdd.context)
15         row_rdd = rdd.map(lambda w: Row(hashtag=w[0], count=w[1]))
16         hashtags_df = sql_context.createDataFrame(row_rdd)
17
18         # Register as table
19         hashtags_df.registerTempTable("hashtags")
20
21         # Get top 30 hashtags from the table using SQL and print them
22         top_hashtags = sql_context.sql(
23             "SELECT hashtag, count FROM hashtags ORDER BY count DESC LIMIT 30"
24         )
25         top_hashtags.show()
26     except:
27         e = sys.exc_info()
28         print("Error: ", e)
29
30 def extract_hashtags(text):
31     # Extract hashtags from text
32     return re.findall(r"#(\w+)", text)
33
34 if __name__ == '__main__':
35     sc = SparkContext(appName="KafkaHashtagCount")
36     sc.setLogLevel("WARN")
37
38     ssc = StreamingContext(sc, 2) # 2-second batch interval
39     ssc.checkpoint('./checkpoint123') # Set checkpoint directory
40
41     # Create Kafka stream
42     kafkaStream = KafkaUtils.createStream(ssc, 'dicvmd7.ie.cuhk.edu.hk:2181', 'test', {'bitcoin-1155157657-1':
43
44     # Extract messages from the stream
45     lines = kafkaStream.map(lambda x: x[1])
46
47     # Extract hashtags and count them
48     hashtags = lines.flatMap(lambda line: extract_hashtags(line))
49     hashtag_counts = hashtags.map(lambda hashtag: (hashtag, 1)).reduceByKeyAndWindow(
50         lambda x, y: x + y,
51         lambda x, y: x - y,
52         300, # Window length of 300 seconds (5 minutes)
53         120 # Slide interval of 120 seconds (2 minutes)
54     )
55
56     # Process each RDD generated in each interval
57     hashtag_counts.foreachRDD(process_rdd)
58
59     ssc.start() # Start the computation
60     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
1  import os
2  import random
3  import time
4  from datetime import datetime
5
6  def convert_to_seconds(ts):
7      dt_obj = datetime.strptime(ts, '%Y-%m-%d %H:%M:%S')
8      return int(time.mktime(dt_obj.timetuple()))
9
10 # The sleep time should be proportional to the difference in timestamps of the tweets
11 def random_sleep(current_ts, last_ts):
12     if last_ts is None:
13         sleep_time = random.randint(1, 4)
14     else:
15         # Calculate the difference
16         sleep_time = current_ts - last_ts
17
18     sleep_time = max(0.1, sleep_time)
19     time.sleep(sleep_time)
20
21 def main():
22     last_ts = None
23     with open('new_tweets.txt') as f:
24         for line in f:
25
26             line = line.decode('ascii', 'ignore')
27
28             # split the text and timestamp
29             parts = line.rstrip().rsplit(',', 1)
30             text = parts[0]
31             ts = parts[-1]
32
33             ts = convert_to_seconds(ts)
34
35             cmd = 'echo "' + text + '" | /usr/hdp/2.6.5.0-292/kafka/bin/kafka-console-producer.sh --broker-list
36
37             print(cmd)
38             os.system(cmd)
39
40
41             if last_ts is not None:
42                 random_sleep(ts, last_ts)
43
44             # Update the last timestamp
45             last_ts = ts
46
47
48 if __name__ == '__main__':
49     main()
50
```

Code:

```
1  from pyspark.sql import SparkSession
2  from pyspark.sql.functions import explode, split, window, col
3  from pyspark.sql.types import StringType
4  import time
5
6  if __name__ == '__main__':
7      # Initialize SparkSession
8      spark = SparkSession.builder \
9          .appName("StructuredKafkaHashtagCount_12345") \
10         .getOrCreate()
11
12     # Set log level
13     spark.sparkContext.setLogLevel("WARN")
14
15     # Read messages from Kafka
16     df = spark \
17         .readStream \
18         .format("kafka") \
19         .option("kafka.bootstrap.servers", "dicvmd7.ie.cuhk.edu.hk:6667") \
20         .option("subscribe", "bitcoin-1155157657-1") \
21         .load()
22
23     messages = df.selectExpr("CAST(value AS STRING)", "CAST(timestamp AS TIMESTAMP)")
24
25     hashtags = messages.withColumn("word", explode(split(col("value"), " "))) \
26         .filter(col("word").rlike("#\\w+")) \
27         .select("word", "timestamp")
28
29     # Group by hashtags and window
30     hashtag_counts = hashtags.groupBy(
31         window(hashtags.timestamp, "300 seconds", "120 seconds"),
32         hashtags.word
33     ).count().orderBy(col("window.start").desc(), col("count").desc())
34
35
36     # writes the output to memory sink
37     query = hashtag_counts \
38         .writeStream \
39         .outputMode("complete") \
40         .queryName("hashtags") \
41         .format("memory") \
42         .option("checkpointLocation", "./checkpointanpojbfhdakjfb") \
43         .trigger(processingTime='2 minutes') \
44         .start()
45
46     # Use this function to periodically query the top hashtags for the top timestamps
47     while query.isActive:
48         spark.sql("SELECT window, word, count FROM hashtags ORDER BY window.start DESC, count DESC LIMIT 30")
49
50         time.sleep(120) # sleep for 2 minutes before updating the console again
51
52
53     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/XHbsR0coye	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