DATA-228 BIG DATA TECHNOLOGIES AND APPLICATIONS

HOMEWORK-2

Group-9

Aarthika Teegala (017047589)

Harshitha Boddu (017046913)

Nivegna Lagadapati (016980626)

Leela Sai Rahul Suryadevara (017440735)

Venkata Sai Krishna Velamala (017422366)

Question-1(a): Examine the Hadoop source code at https://github.com/apache/hadoop/ to see how map-reduce and similar paradigms are used to run the jobs. Checkout the map reduce examples at https://github.com/apache/hadoop/tree/trunk/hadoop-mapreduce-project/hadoop-mapreduce-examples. List 10 key findings from your analysis.

Solution-1(a):

Here are the 10 key findings:

1. Hadoop's design follows a modular pattern, with distinct components handling tasks like distributed storage (HDFS), resource management (YARN), and data processing (MapReduce).
2. Hadoop is predominantly written in Java, ensuring the language's strengths for cross-platform compatibility and fostering a robust developer community.
3. HDFS stores and manages data across clusters, with source code revealing insights into data storage and replication strategies.
4. YARN acts like a traffic manager, making sure everyone gets their fair share of the resources.
5. Analysis of MapReduce internals reveals mechanisms for task scheduling, fault tolerance, and data processing across nodes.
6. Hadoop incorporates various mechanisms to handle node failures and data corruption, ensuring system reliability.
7. Hadoop's design facilitates horizontal scaling, distributing data and processing tasks across multiple nodes.
8. The source code may contain optimizations and enhancements focused on improving data processing speed, resource utilization, and overall system efficiency.
9. Hadoop integrates security measures such as authentication, authorization, and encryption.

10. The Hadoop source code repository reflects a collaborative developer community, with insights into code contributions, reviews, and discussions, providing a glimpse into best practices in software development.

Question-1(b): List 10 key findings from your analysis of the source code at https://github.com/Yelp/mrjob/

Solution-1(b):

Here are the 10 key findings:

1. **Abstracted Functionality**: The code simplifies complexities by presenting a Python interface for MapReduce tasks.
2. **Map and Reduce Logic**: Inside the codebase, you'll likely find implementations of map and reduce functions, crucial for data processing.
3. **Job Customization**: It offers mechanisms for adjusting job parameters, input/output paths, and other configurations.
4. **Integration Capabilities**: There are features enabling seamless interaction with Hadoop clusters or AWS Elastic MapReduce instances.
5. **Data Handling Utilities**: Utilities are available for efficiently managing various input data sources and formats for output.
6. **Optimization Support**: The code may include support for optimizing performance through combiners and partitioners.
7. **Error Management**: Expect provisions for managing errors during job execution, ensuring robustness.
8. **Scalability Enhancements**: Features are present to facilitate the distribution of computation across nodes, catering to large datasets.
9. **Testing Frameworks**: You might encounter testing tools or frameworks to verify the accuracy of job implementations.
10. **Documentation Resources**: Inline comments and external documentation serve to aid users in understanding and contributing to the project effectively.

Question-2(a): Document technical difficulties and possible best practices you discovered in the process of setting up your environment.

Solution-2(a):

Below are the technical difficulties discovered in the process of setting up the environment:

1. In Windows, though the installation of VirtualBox was easy, the setup of Hortonworks Sandbox was difficult. After starting the VM, initialization of the VM took a lot of time.

2. The Ambari Sandbox page was not accessible initially, so we had to restart the VM and try again using multiple ways like using ip address and port.
3. Once the Ambari Sandbox was opened, it took a lot of time to resolve the build issues.
4. On Mac, there are no builds for apple silicon chips, though there are a few for intel-based macs. The latest version is Beta version.
5. Support for HDP Sandbox has been stopped, no proper documentation and no community support.
6. Docker application installation was easy on windows and mac. The problem with the windows version is though there are a few docker images available, the minimum RAM requirement is 32 GB, due to which starting the instance itself took a lot of time. Once the instance was started, the instance always crashed within 2 minutes.
7. The problem with the mac version is docker images relating to HDP Sandbox are designed to work with AMD architecture, and not ARM architecture. We could not find ARM supported instances.
8. On windows, we utilized the VMWare Player application to setup and run the HDP sandbox as it required less resources to run the sandbox such as 8GB of RAM and 20 GB of storage.
9. The version of python supported by HDP Sandbox is 2.7, therefore, when running python programs, too many errors surrounding the syntax of python libraries were raised, which caused the execution of python programs to fail.
10. The mrjob library installation was not straightforward, there were too many installation errors, and the library was not found in the first place to be installed. We then had to manually find the suitable version and use wget command to install the library.
11. The mrjob library is supported by python versions which are greater than 3.6, which is why there were many compatibility issues during the execution of python programs.

Question-2(b): Choose an innovative application and explain the motivation.

i. Identify an interesting project that involves processing large datasets. Consider topics like extracting keywords and analyzing trends from news articles or product reviews, processing geospatial data for weather forecasting or traffic analysis, etc.
ii. Data Acquisition: Find publicly available datasets related to your chosen application. Explore resources like:
    a) Kaggle: https://www.kaggle.com/
    b) UCI Machine Learning Repository: https://archive.ics.uci.edu/
    c) Google Dataset Search: https://datasetsearch.research.google.com/

Solution-2(b):

i. The dataset we chose is Twitter Dataset.
ii. The dataset is taken from Kaggle.

Source: https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis

Question-2(c): Implement your MapReduce job.

i. Write and document Python code using the mrjob library to implement your chosen application's logic. Utilize MapReduce functions like mapper and reducer to process your dataset.
ii. Thoroughly test your script locally before running it on the Hadoop cluster. Ensure it produces the desired results correctly. Explain how you tested the code.
iii. Hadoop Job Submission: Submit your MapReduce job to the Single node Hadoop cluster using the mrjob command-line interface. Analyze the output and refine your script as needed. List 3 key findings.

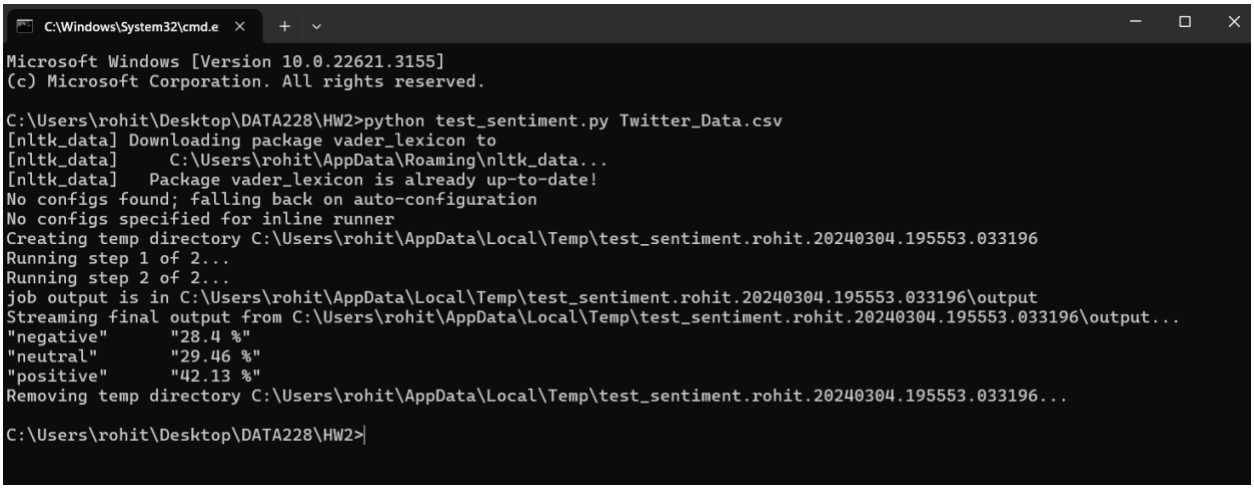
Solution-2(c):

i. Code:

```
import nltk
from mrjob.job import MRJob
from mrjob.step import MRStep
from nltk.tokenize import word_tokenize
from nltk.sentiment import SentimentIntensityAnalyzer
import re
sentiAna=SentimentIntensityAnalyzer()

nltk.download('vader_lexicon')


class TwitterTest(MRJob):

    def steps(self):
        return [
            MRStep(mapper=self.mapper_tweets,
                   reducer=self.reducer_tweets),
            MRStep(reducer=self.reducer_percent_tweets)

        ]


    def clean_tweets(self,tweet):
        pattern = r',-?\d+$'# Remove the matched pattern from the text
```

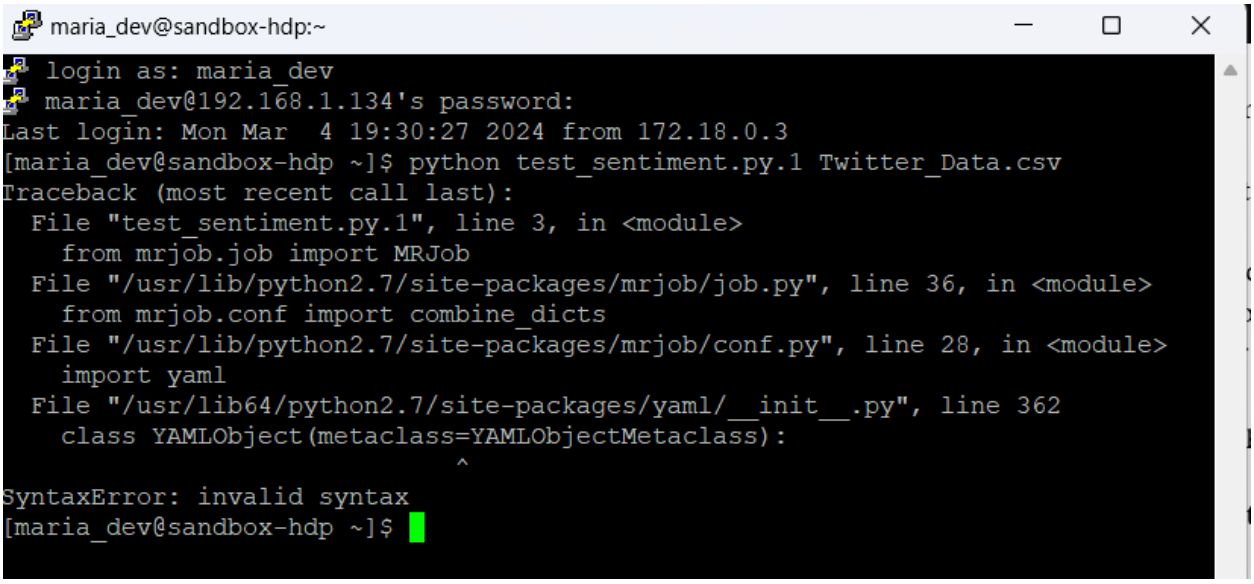
```
        modified_text = re.sub(pattern, '', tweet)
        return modified_text

    def sentiment_analyzer(self,tweet):
        sentiment=sentiAna.polarity_scores(tweet)
        compound=sentiment['compound']
        if compound >= 0.05:
            return "positive"
        elif compound<= -0.05:
            return "negative"
        else:
            return "neutral"


    def mapper_tweets(self,key,tweets):
        tweets=self.clean_tweets(tweets)
        sentiment=self.sentiment_analyzer(tweets)
        yield sentiment , 1

    def reducer_tweets(self,key,value):
        yield "total counts",(key, sum(value))

    def reducer_percent_tweets(self,key,values):
        sentiment_scores=dict(values)
        total_tweets=sum(sentiment_scores.values())
        for sentiment,count in sentiment_scores.items():
            percent=(count/total_tweets)*100
            yield sentiment,f"{round(percent,ndigits=2)} %"


if __name__ == '__main__':
    TwitterTest.run()
```

ii. **Run:** Testing the above script locally before running it on the Hadoop cluster:

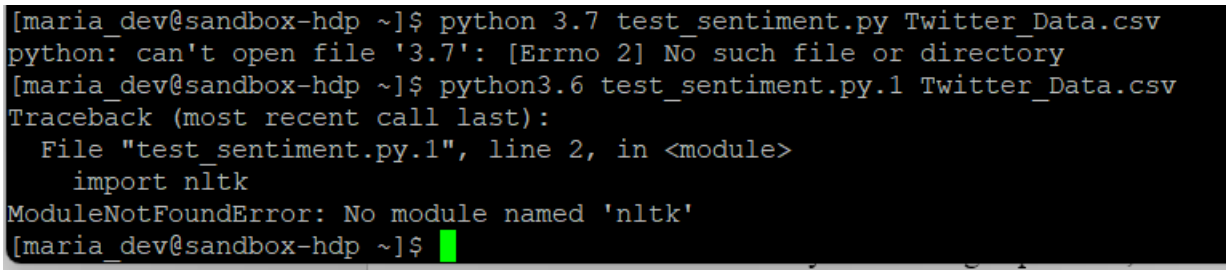


- From the twitter dataset, we took each tweet, removed urls, special characters and other unnecessary data.
- We then passed the cleaned data to the sentiment analyzer function. The output of the function is the sentiment score for each tweet.
- We set a sentiment score threshold of 0.05. Based on this threshold, the tweets with sentiment score >=0.05 (upper bound) are categorized into positive tweets.
- The tweets with sentiment score <= -0.05 (lower bound) are categorized into negative tweets.
- The tweets with a sentiment score between the upper bound and lower bound are categorized into neutral tweets.
- As seen from the output, negative tweets account to 28.4%, positive tweets account to 42.13%, and neutral tweets account to 29.46%.

iii. We were unable to execute the mrjob on command-line interface due to version compatibility issues as seen below. The code in mrjob library is written in python3 format, whereas the HDP Sandbox supports python2.



We tried to execute our program using python 3.6, the mrjob issue was resolved, but we were facing issues with nltk.



To execute our python program, we need the nltk library to perform sentiment analysis. Since the minimum python version required to run nltk library is 3.7, the HDP Sandbox does not support it. Though we tried to install the nltk library using wget command, we were still facing similar compatibility issues.

Question-2(d): Reflect and Share.

i. Summarize your learning experience, challenges faced, and insights gained.
ii. Write a blog post, create a video presentation, or share your work on GitHub to build your brand and engage with others.

Solution-2(d) (i);

Learning experience:

1. Installing and configuring complex software like Hadoop and Hive improved our technical proficiency. It included understanding of system requirements, moving through cofig files and resolving any issues.

2. As we familiarized ourselves with the Sandbox environment, we were able to understand how Hadoop, Hive, and HDFS interact with each other and with data, which further helped us in data processing and analysis.
3. Using mrjob and Parquet increased the range of tasks that can be done with improved efficiency even while using different file formats.
4. Working with Hive, Hadoop and HDFS improved our logical thinking and problem-solving skills which we were able to troubleshoot any issues.

Challenges faced:

1. Setting up Hadoop and Hive meant carefully adjusting settings and files to fit the system in which they are being used.
2. Learning about HDFS was complex at first and it took meticulous practice.
3. Writing MapReduce functions, creating efficient mappers, reducers, and combiners required sound knowledge of MapReduce concepts.
4. Debugging and fixing errors was tough.

Insights gained:

1. Configuring Hadoop, Hive, and HDFS involved different configuration files. Throughout this process, we managed configuration efficiently and understood how it impacts the system's stability and performance.
2. Delving deeper into HDFS and distributed file systems, we were able to gain insights into data storage and retrieval mechanisms across multiple nodes. We also learnt the importance of fault tolerance, scalability and integration in distributed environments.
3. We were able to practically apply, test the MapReduce scripts and troubleshoot effectively.

Solution-2(d) (ii):

GitHub Link:
https://github.com/VenkataSaiKrishnaVelamala/Sentiment_analysis_using_MRJOB

Question-3 (1): Document technical difficulties and possible best practices you discovered in the process of setting up your environment.

Solution-3 (1):

Challenges Faced and Resolutions: Installing Hadoop on macOS

Initiating Setup:

Upon commencing the configuration of the macOS environment for Hadoop installation, several hurdles impeded our progress. Initially, we encountered significant complications stemming from Hadoop's incompatibility with Java versions 11 and beyond. Consequently, we explored alternative avenues, contemplating the installation of Homebrew as a viable substitute.

Encountered Challenges:

1. **Java Compatibility with Hadoop:** The foremost challenge arose from the incongruity between Hadoop and Java versions 11 or higher on macOS.
2. **HDFS Formatting Error:** An obstacle surfaced when executing the HDFS formatting command `hdfs namenode -format`, leading to the error message: "Cannot execute /usr/local/Cellar/hadoop/3.0.0/libexec/hdfs-config.sh."
3. **Homebrew Installation Issues:** Subsequent to considering Homebrew as an installation method, further complexities arose during the installation process of Hadoop via this platform.
4. **HADOOP_HOME Configuration:** Additional impediments manifested due to inadequacies in configuring the HADOOP_HOME environment variable, hindering seamless operation.
5. **Permission Challenges:** Lastly, potential permission issues posed a hindrance to the proper functioning of Hadoop on local host, exacerbating the installation process.

Addressing these challenges required meticulous troubleshooting and strategic adjustments to ensure a successful installation of Hadoop on the macOS environment.

Changing to Windows and Its Difficulties:

Owing to the previously described difficulties with macOS, we decided to move the installation process over to Windows. Even though the setup was more straightforward overall, manual configuration was still required:

1. **Creating Folders for DataNodes and NameNodes:** The DataNode and NameNode each have their own folder that we made.
2. **Making Changes to Configuration Files:** Several configuration files were modified by us, including:
   *{hdfs-site.xml}: Changing HDFS-related attributes.
   * {`yarn-site.xml}: Setting up the Yarn configuration.
   * `hadoop-env.cmd`: Changing `%JAVA_HOME%} to the installation directory path for the Java JDK 1.8.
3. **An additional problem was encountered:** The "Program Files" folder was the default location for the program during the Java JDK installation. The folder name with a space in it produced problems for Hadoop. To fix this, we
   * Relocated the Java JDK folder to a path that is space-free.

* Relocated the Java folder to the new location and updated the system path and `JAVA_HOME} variables accordingly.

Question-3 (2): Develop the application.

i. **Real-time Stream Consumer:** Write a Python script using APIs to consume posts (data) from a specific keyword in real-time. Filter and pre-process the data (remove URLs, etc.)

ii. **Real-time Streaming Data Analysis with Spark:**
For Sentiment Analysis - Leverage a pre-trained sentiment analysis model (e.g., VADER) within Spark. Use PySpark to analyze the sentiment (positive, negative, neutral) of each post in the stream.
(Or) For Trend Analysis with Spark – Use PySpark to analyze the market trends and get some meaningful and useful insights if the choice of API is Alpaca API.
(Or) Perform something similar for a different application.

iii. **Store and Access Data:** Write the analyzed data to a Kafka topic. Use Spark Streaming to read data from Kafka and store it in a Hive table in Parquet format.

iv. **Visualization:** Create a simple dashboard using tools like Kibana or Grafana to visualize real-time data analysis.

v. Summarize your learning experience, challenges faced, and insights gained.

Solution-3 (2):

i)

```
import praw
from kafka import KafkaProducer

import json
import time


reddit = praw.Reddit(
    client_id='PKIShrCUOkF56laMZtAUTQ',
    client_secret='i-lygrIraIzh85Fvn1RG5xLFcZEGbw',
    user_agent='my_sentiment_analysis by u/SuperWishbone6330',
    #username='u/SuperWishbone6330',
    #password='Sujani@69'
)
```

Praw is a Python Reddit API Wrapper that allows access to Reddit's API easily. To the Jupyter notebook, we imported Praw and then we imported KafkaProducer from Kafka to produce the data in kafka that is being read from the redditt API.

Establishing a connetion

- `reddit = praw.Reddit`: This line creates a Reddit API connection object. It uses the `praw.Reddit` class and passes a dictionary containing authentication and user agent information.

Authentication Details

- `client_id='PKIShrCUOkF56laMZtAUTQ'`: This is the client ID associated with the Reddit API app. The client ID is obtained when you create a Reddit app on the Reddit developer platform.
- `client_secret='i-lygrIraIzh85Fvn1RG5xLFcZEGbw'`: This is the client secret associated with the Reddit API app. Like the client ID, the client secret is obtained when you create a Reddit app.
- `user_agent='my_sentiment_analysis by u/SuperWishbone6330'`: The user agent is a string that helps Reddit identify the source of the API requests. It should be unique and descriptive. In this case, it includes the name of the project ("my_sentiment_analysis") and the Reddit username of the app creator ("u/SuperWishbone6330").

```
from confluent_kafka import Producer

config = {
    'bootstrap.servers': 'pkc-56d1g.eastus.azure.confluent.cloud:9092',
    'security.protocol': 'SASL_SSL',
    'sasl.mechanism': 'PLAIN',
    'sasl.username': 'H35D7ED7PN74YGWT',
    'sasl.password': 'DMrW+RwSTk5YVnYvGufbv0K6fSCn0TF9ODJr4TfA9l4FTF3slrs3zUDYTUmDir2i',
    'ssl.endpoint.identification.algorithm': 'https',
}

producer = Producer(**config)
```

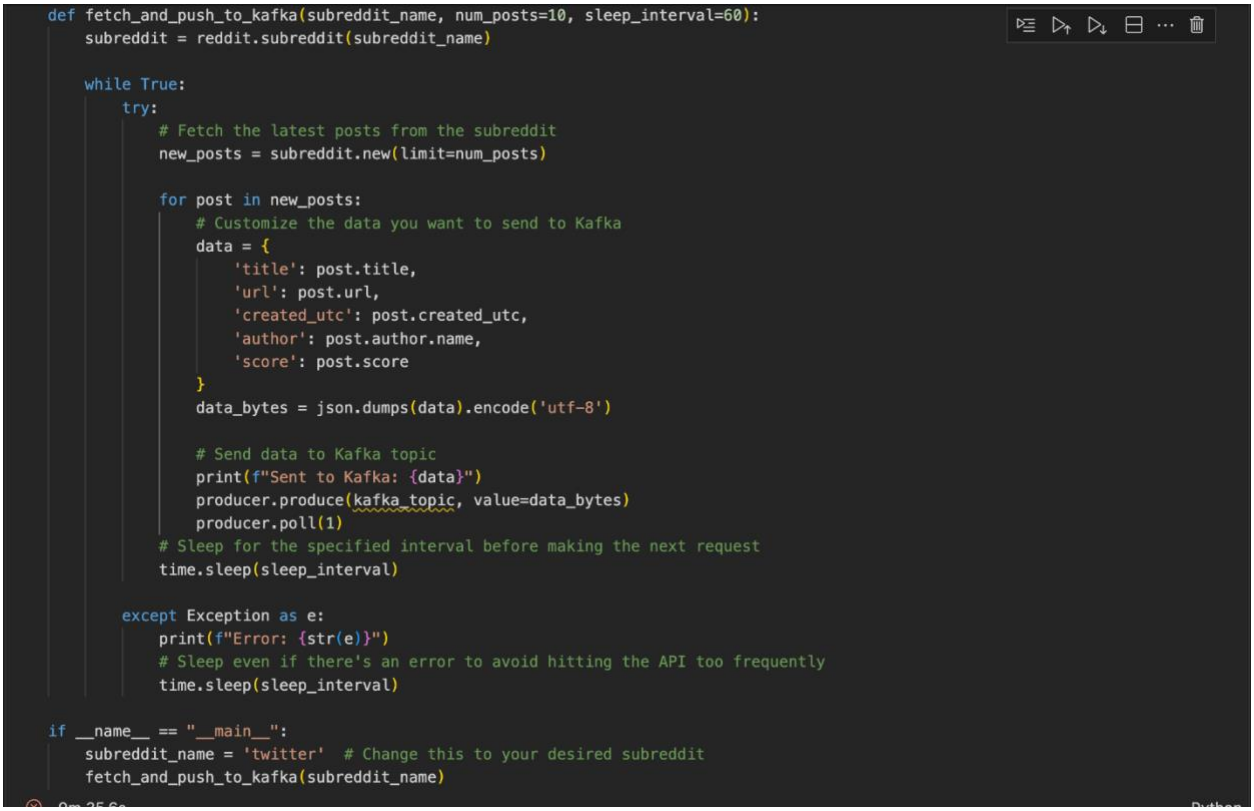
✓ 0.0s

```
from confluent_kafka import Producer
```

✓ 0.1s

- `'bootstrap.servers'`: The address of the Kafka bootstrap servers. In this case, it points to a Confluent Cloud Kafka cluster.
- `'security.protocol'`: The security protocol used for communication. Here, it's set to `'SASL_SSL'`, indicating a combination of SASL authentication and SSL encryption.
- `'sasl.mechanism'`: The Simple Authentication and Security Layer (SASL) mechanism used for authentication. Here, it's set to `'PLAIN'`, which is a widely used mechanism.
- `'sasl.username'` and `'sasl.password'`: The username and password for SASL authentication. These are specific to your Confluent Cloud account.
- `'ssl.endpoint.identification.algorithm'`: The SSL endpoint identification algorithm. Setting it to `'https'` means that the Kafka broker's hostname will be verified using HTTPS-style certificate matching.

The line `producer = Producer(**config)` creates a Kafka producer instance using the provided configuration settings. The double-asterisk (`**`) unpacks the dictionary (`config`) to pass its key-value pairs as keyword arguments to the producer constructor.



```python
def fetch_and_push_to_kafka(subreddit_name, num_posts=10, sleep_interval=60):
    subreddit = reddit.subreddit(subreddit_name)

    while True:
        try:
            # Fetch the latest posts from the subreddit
            new_posts = subreddit.new(limit=num_posts)

            for post in new_posts:
                # Customize the data you want to send to Kafka
                data = {
                    'title': post.title,
                    'url': post.url,
                    'created_utc': post.created_utc,
                    'author': post.author.name,
                    'score': post.score
                }
                data_bytes = json.dumps(data).encode('utf-8')

                # Send data to Kafka topic
                print(f"Sent to Kafka: {data}")
                producer.produce(kafka_topic, value=data_bytes)
                producer.poll(1)
            # Sleep for the specified interval before making the next request
            time.sleep(sleep_interval)

        except Exception as e:
            print(f"Error: {str(e)}")
            # Sleep even if there's an error to avoid hitting the API too frequently
            time.sleep(sleep_interval)

if __name__ == "__main__":
    subreddit_name = 'twitter'  # Change this to your desired subreddit
    fetch_and_push_to_kafka(subreddit_name)
```

The Python script establishes a continuous data ingestion process from a specific subreddit on Reddit to a Kafka topic. The `fetch_and_push_to_kafka` function, utilizing the `praw` library for Reddit API interaction and the `kafka-python` library for Kafka integration, fetches a defined number of latest posts from the specified subreddit. Each post's relevant information, including title, URL, creation timestamp, author, and score, is customized into a dictionary, serialized to bytes using JSON, and then sent to a Kafka topic. The script includes error handling to manage exceptions during the data fetching process, and it adheres to a specified sleep interval to avoid excessive API requests and potential errors. The main block initializes the subreddit ('twitter' in this instance) and triggers the continuous data retrieval and Kafka publishing process.

ii)

In the above code we initialize a Spark session using PySpark, setting up configurations for Kafka integration. Here's a brief explanation:The code's primary purpose is to create a Spark session named "ConfluentKafkaConsumerExample" with the necessary configurations for interacting with Kafka.

- `SparkSession.builder`: Initiates the creation of a Spark session using the builder pattern.
- `.appName("ConfluentKafkaConsumerExample")`: Sets a human-readable name for the Spark application, aiding identification in the Spark UI.
- `.config("spark.jars.packages", "org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2")`: Configures Spark to include the required package (`org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2`) for Kafka integration. This package provides Spark's Kafka connector for version 0.10 of Kafka.
- `.getOrCreate()`: Attempts to get an existing Spark session or creates a new one if none exists, ensuring only one Spark session per application.

`.readStream`: Specifies that the DataFrame should be created for streaming data.

`.format("kafka")`: Specifies that the source is a Kafka topic.

Connection Settings

- `.option("kafka.bootstrap.servers", "...")`: Sets the Kafka bootstrap servers to establish the initial connection.
- `.option("subscribe", "reddit_data")`: Specifies the Kafka topic ("reddit_data") to subscribe to.
- `.option("kafka.security.protocol", "SASL_SSL")`: Configures the security protocol for Kafka.
- `.option("kafka.sasl.mechanism", "PLAIN")`: Specifies the SASL mechanism for authentication.
- `.option("kafka.sasl.jaas.config", "...")`: Provides the JAAS configuration for SASL authentication with the required username and password.
- `.option("kafka.ssl.endpoint.identification.algorithm", "https")`: Configures the SSL endpoint identification algorithm.
- `.option("startingOffsets", "earliest")`: Specifies that the stream should start reading from the earliest available offset in the Kafka topic.
- `.load()`: Loads the streaming data from the specified Kafka topic and configurations into a DataFrame (`kafkaDataFrame`)

The code processes data from a Kafka topic in PySpark Structured Streaming. It selects and casts specific columns of interest from the Kafka data, creating a new DataFrame named `processedDataFrame`. The next step, not shown, involves setting up a streaming query to write the processed data to an output sink, such as the console or a database.

```
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType
import re

def clean_text(text):
    # Remove URLs
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)
    # Remove special characters and digits
    text = re.sub(r'\W+|\d+', ' ', text)
    # Optionally, remove single characters and extra spaces, convert to lowercase
    text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text).strip()
    return text.lower()

# Register the function as a UDF
clean_text_udf = udf(clean_text, StringType())
```

[6] ✓ 0.0s

This code defines a PySpark user-defined function (UDF) named `clean_text` to preprocess text data. The function, applied using regular expressions, removes URLs, special characters, and digits. Optionally, it handles single characters and extra spaces, converting the text to lowercase. The UDF is registered as `clean_text_udf` and can be employed in PySpark DataFrames for efficient text cleaning operations.

```
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType
```

✓ 0.0s

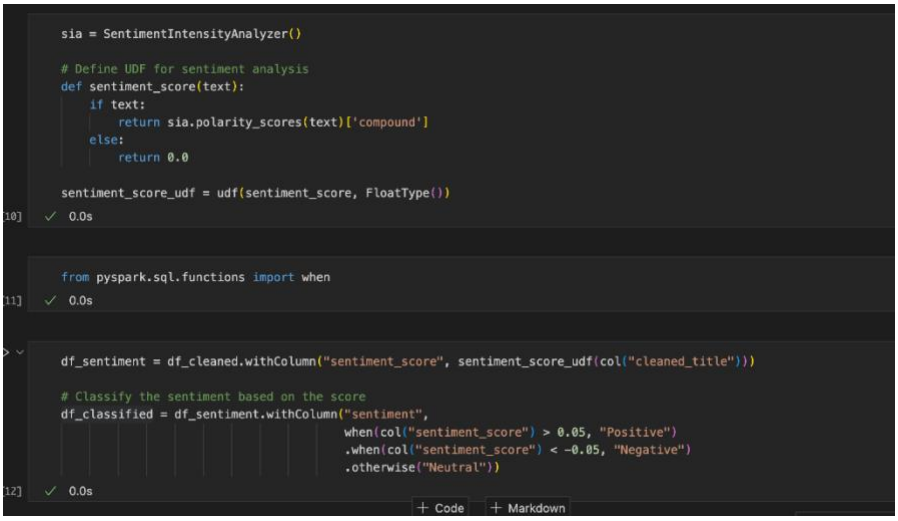
```
schema = StructType([
    StructField("title", StringType()),
    StructField("url", StringType()),
    StructField("created_utc", FloatType()),
    StructField("author", StringType()),
    StructField("score", IntegerType()),
])

# Parse the JSON from the 'value' column
processedDataFramedf_parsed = processedDataFrame.withColumn("value", from_json("value", schema))
df_cleaned = processedDataFrame.withColumn("cleaned_title", clean_text_udf(col("value")))
```

✓ 0.1s

```
from nltk.sentiment import SentimentIntensityAnalyzer
from pyspark.sql.functions import udf
from pyspark.sql.types import FloatType
```

✓ 0.6s

The code defines a PySpark StructType schema named `schema`, specifying the structure of the data to be parsed. It includes fields like "title," "url," "created_utc," "author," and "score" with their respective data types. The `processedDataFrame` is then transformed using the `withColumn` method to parse JSON data from the 'value' column using the specified schema, resulting in `processedDataFrame_parsed`. Subsequently, the `clean_text_udf` user-defined function is applied to the 'value' column, and the cleaned text is stored in a new column named "cleaned_title" within the DataFrame `df_cleaned`.



```
sia = SentimentIntensityAnalyzer()

# Define UDF for sentiment analysis
def sentiment_score(text):
    if text:
        return sia.polarity_scores(text)['compound']
    else:
        return 0.0

sentiment_score_udf = udf(sentiment_score, FloatType())
```

[10] ✓ 0.0s

```
from pyspark.sql.functions import when
```

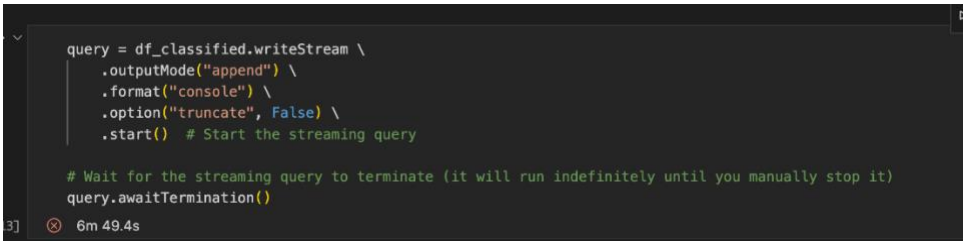
[11] ✓ 0.0s

```
df_sentiment = df_cleaned.withColumn("sentiment_score", sentiment_score_udf(col("cleaned_title")))

# Classify the sentiment based on the score
df_classified = df_sentiment.withColumn("sentiment",
                                        when(col("sentiment_score") > 0.05, "Positive")
                                        .when(col("sentiment_score") < -0.05, "Negative")
                                        .otherwise("Neutral"))
```

[12] ✓ 0.0s

+ Code + Markdown

The provided code initializes a SentimentIntensityAnalyzer from the `nltk` library, typically used for sentiment analysis. It then defines a user-defined function (UDF) named `sentiment_score` that calculates the sentiment compound score using the SentimentIntensityAnalyzer. The UDF is registered as `sentiment_score_udf` with a return type of FloatType. This UDF can be applied to PySpark DataFrames to analyze sentiment scores based on text data. If the text is non-empty, the compound sentiment score is calculated using the SentimentIntensityAnalyzer; otherwise, a default score of 0.0 is returned.



```
query = df_classified.writeStream \
    .outputMode("append") \
    .format("console") \
    .option("truncate", False) \
    .start()  # Start the streaming query

# Wait for the streaming query to terminate (it will run indefinitely until you manually stop it)
query.awaitTermination()
```

[13] ⊗ 6m 49.4s

This code augments a PySpark DataFrame (`df_cleaned`) with sentiment analysis results. It uses the `withColumn` method to add a new column named "sentiment_score" to the DataFrame, applying the previously defined `sentiment_score_udf` user-defined function to the "cleaned_title" column. Subsequently, it creates another new column, "sentiment," based on the calculated sentiment scores. The sentiment is classified as "Positive" if the score is greater than 0.05, "Negative" if less than -0.05, and "Neutral" otherwise. This DataFrame (`df_classified`) now includes sentiment scores and classifications for further analysis of sentiment in the processed data.

Sending all the twitter data collected as key value pairs to kafka.

Confluent platform:

Confluent Platform is a specialized distribution of Kafka that includes additional features and APIs.

Source: https://docs.confluent.io/platform/current/platform.html

In the below screenshots, the functioning of Confluent cluster can be seen. And the number of bytes per second for both Production and Consumption can be inferred.

Live (1)

cluster_0

Running

Metrics

Production	Consumption	Storage
0B/s	0B/s	31.03KB

Resources

ksqlDB	Connectors	Clients
0	0	0

Overview

ID	lkc-xx28jx
Type	Basic
Provider & region	AZURE | eastus

cluster_0

Running

Metrics

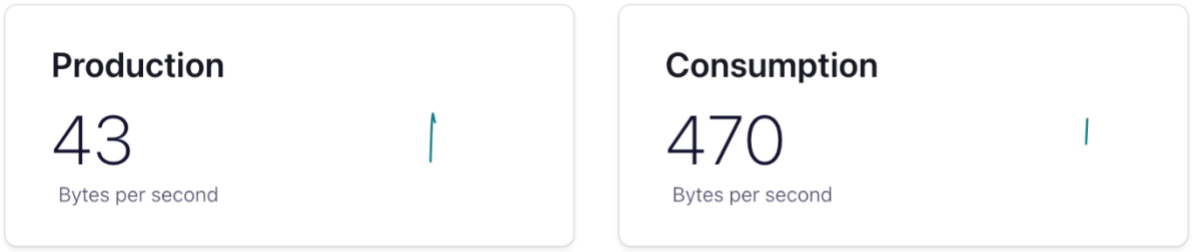
Production	Consumption	Storage
18B/s	31B/s	39.95KB

Resources

ksqlDB	Connectors	Clients
0	0	3

Overview

ID	lkc-xx28jx
Type	Basic
Provider & region	AZURE | eastus



reddit_data

Actions

Overview | Messages | Schema | Configuration

Production in last hour	Consumption in last hour	Total messages	Retention time
38 messages	152 messages	165	1 week

Filter by timestamp, offset, key or value | All partitions | Latest | Max 50 results

50 messages shown | Auto-refresh on | CSV | JSON

Timestamp	Offset	Partition	Key	Value
1709591945500	26	3		{"title":"Talking to someone in in DM's, before suddenly unable to?","url":"https://www.reddit.com/r/Tw
1709591945298	28	5		{"title":"X / Twitter allows users to suggest a coup if they don't agree with government","url":"https://w
1709591945095	27	5		{"title":"Girls asking questions?","url":"https://www.reddit.com/r/Twitter/comments/1b6dkuz/girls_askir
1709591944879	25	1		{"title":"[Help Needed] Growth stalled on twitter after changing profile to personal brand","url":"https:/
1709591944679	26	5		{"title":"Why do people say Twitter/X sucks?","url":"https://www.reddit.com/r/Twitter/comments/1b6hlr

Identification

Name	cluster_0
Cluster ID	lkc-xx28jx

Endpoints

Bootstrap server	pkc-56d1g.eastus.azure.confluent.cloud:9092
REST endpoint	https://pkc-56d1g.eastus.azure.confluent.cloud:443

Use the Kafka REST API to interact with your cluster and produce records.

The data that is being fetched is represented as the key value pairs and displayed.

```
-----+-----------+---------+------+-----------------------+------------------------------------------------------------------------
     |topic      |partition|offset|timestamp              |cleaned_title
-----+-----------+---------+------+-----------------------+------------------------------------------------------------------------
': 1}|reddit_data|2        |0     |2024-03-04 01:50:02.846|title if an account is shadowbanned temporary label that won t go
     |reddit_data|2        |1     |2024-03-04 01:50:03.686|title what does this error message mean url created_utc     author
     |reddit_data|2        |2     |2024-03-04 01:51:04.684|title tweets going straight to drafts can t follow anyone url crea
     |reddit_data|2        |3     |2024-03-04 01:51:05.073|title social media platform without political discussions or recom
     |reddit_data|2        |4     |2024-03-04 01:51:07.064|title twitter automatically makes me leave groupchats url created_
     |reddit_data|2        |5     |2024-03-04 01:51:07.263|title help pls can t change url created_utc     author rudolphfn s
     |reddit_data|2        |6     |2024-03-04 02:15:37.661|title social media platform without political discussions or recom
     |reddit_data|2        |7     |2024-03-04 02:16:41.911|title how to gain followers url created_utc     author jjcalifajoy
     |reddit_data|2        |8     |2024-03-04 02:17:43.58 |title tweets going straight to drafts can t follow anyone url crea
     |reddit_data|2        |9     |2024-03-04 02:55:00.483|title what does this error message mean url created_utc     author
': 1}|reddit_data|2        |10    |2024-03-04 02:56:43.055|title if an account is shadowbanned temporary label that won t go
     |reddit_data|2        |11    |2024-03-04 03:03:04.484|title tweets going straight to drafts can t follow anyone url crea
     |reddit_data|2        |12    |2024-03-04 03:03:06.337|title went to try to change my timeline settings and now it just s
     |reddit_data|3        |0     |2024-03-03 22:40:09.356|all that is gold
     |reddit_data|3        |1     |2024-03-04 01:50:00.852|title tweets going straight to drafts can t follow anyone url crea
     |reddit_data|3        |2     |2024-03-04 01:50:04.276|title can login to my account url created_utc     author adasstrap
     |reddit_data|3        |3     |2024-03-04 02:16:42.709|title can login to my account url created_utc     author adasstrap
     |reddit_data|3        |4     |2024-03-04 02:17:44.155|title went to try to change my timeline settings and now it just s
     |reddit_data|3        |5     |2024-03-04 02:17:45.151|title help pls can t change url created_utc     author rudolphfn s
     |reddit_data|3        |6     |2024-03-04 02:54:57.125|title social media platform without political discussions or recom
-----+-----------+---------+------+-----------------------+------------------------------------------------------------------------
```

Now, we have tried to separate the key value pairs and put them under different columns and cleaned the data.

sentiment_score	sentiment
0.5994	Positive
-0.481	Negative
0.2263	Positive
0.0	Neutral
-0.0516	Negative
0.4588	Positive
0.0	Neutral
0.5267	Positive
0.2263	Positive
-0.481	Negative
0.5994	Positive
0.2263	Positive
0.0	Neutral
0.0	Neutral
0.2263	Positive
0.0	Neutral
0.0	Neutral
0.0	Neutral
0.4588	Positive
0.0	Neutral

Output after performing the sentiment analysis using PySpark. The senitment is either Positive, negative or Neutral.

Question-3(3): Implement any one of the following:

- Analyze variations over time and identify trends.
- Incorporate geographical information and analyze sentiment by location.
- Use more advanced NLP techniques for improved sentiment accuracy.
- A comparative market trend analyzer with historic and real-time data using Alpaca API

- Write a blog post, create a video presentation, and share your work on GitHub to build your brand and engage with others

Solution-3(3):

Blog Post: https://medium.com/@taarthika21/exploring-real-time-data-analysis-with-kafka-spark-and-hive-b2dfb99e71dd

GitHub Link: https://github.com/VenkataSaiKrishnaVelamala/RealTime_Reddit_Data_Stremaing_Pipeline

Note:

The video presentation is uploaded in the GitHub link.