

Introduction to Big Data

Final Assignment

Name - Avijeet Palit

Roll - 21f1005675

Date - 15/12/2024

Objective.....	1
Breakdown of the Assignment.....	1
Detailed Steps.....	2
Step 1: Create a GCP Project.....	2
Step 2: Download the data set from g-drive.....	2
Step 3: Set Up Google Cloud Storage (GCS).....	3
Step 4: Create a Virtual Machine (VM):.....	4
Step 5: Create a New Firewall Rule.....	5
Step 6: Install Kafka and Dependencies.....	7
Step 7: Set Up and Start Zookeeper.....	8
Step 8: Set Up Kafka.....	9
Step 9: Kafka Producer.....	10
Step 10: Create a Dataproc Cluster.....	12
Step 11: Result and Testing the Pipeline.....	14
Conclusion.....	15

Objective

The goal of the assignment is to identify and emit anomalies in historical stock trading data using Spark Streaming. The anomalies are defined as trades that significantly deviate in price or volume, helping a fraud control unit detect suspicious trading activities for further scrutiny.

Breakdown of the Assignment

1. Data Preparation and Loading

- Load minute-level stock trading data (price and volume) for a 3-year period (2017–2020).
- Stream data by loading into Kafka using Spark Batch.
- Ensure data quality checks to handle bad rows, unsorted timestamps, and data gaps.

2. Real-Time Stream Processing

- Write Spark Streaming code to consume data minute-by-minute from Kafka.
 - Use window functions for anomaly detection:
 - **A1:** Detect price deviations exceeding $\pm 0.5\%$ compared to the previous minute.
 - **A2:** Detect volume spikes exceeding 2% of the average traded volume for the last 10 minutes.
 - Emit detected anomalies with relevant details (trade data and anomaly type).
-

Detailed Steps

Step 1: Create a GCP Project

- Log in to the Google Cloud Console.
- Create a new project or use an existing one.
- Enable the required APIs:
 - **Cloud Storage API**
 - **Dataproc API**
 - **Compute Engine API**

Step 2: Download the data set from g-drive

- Download the dataset
- Unzip the folder
- Manually analyze the dataset for better understanding.

TATACHEM_EQ_NSE_NSE_MINUTE.csv					
A	B	C	D	E	F
timestamp	open	high	low	close	volume
2017-01-02 09:15:00	506.55	509.65	505.45	508.85	12277
2017-01-02 09:16:00	508.35	509.95	508.35	509	8493
2017-01-02 09:17:00	509	509.5	508.75	509.4	4353
2017-01-02 09:18:00	509.4	509.4	508.35	508.4	6385
2017-01-02 09:19:00	508.65	509.45	508.25	508.5	9450
2017-01-02 09:20:00	508.3	509.2	508.3	508.5	10005
2017-01-02 09:21:00	508.5	509	507.9	508.25	5399
2017-01-02 09:22:00	508	508.75	507.9	508.75	4929
2017-01-02 09:23:00	508.75	510.65	508.75	510.25	12241
2017-01-02 09:24:00	509.8	511	509.55	510.85	5535
2017-01-02 09:25:00	511	511.65	510.4	510.4	9080
2017-01-02 09:26:00	509.95	510.8	509.25	510.65	10544
2017-01-02 09:27:00	510.6	510.6	509.9	510.05	13373
2017-01-02 09:28:00	510	510.4	509.7	509.7	4336
2017-01-02 09:29:00	509.7	510.3	509.4	510.3	5295
2017-01-02 09:30:00	510.3	510.45	509.2	509.6	5007
2017-01-02 09:31:00	509.6	509.9	509.05	509.25	3545
2017-01-02 09:32:00	509.25	509.85	509	509	5210
2017-01-02 09:33:00	509	509.85	509	509.45	1279
2017-01-02 09:34:00	509.45	511.6	509.45	511.55	7213
2017-01-02 09:35:00	511.25	511.5	510	510.65	3495
2017-01-02 09:36:00	510.65	511	510.2	510.4	3323
2017-01-02 09:37:00	510.65	510.7	510	510.45	3942
2017-01-02 09:38:00	510.4	511.45	510.4	511	2703
2017-01-02 09:39:00	511	511	510.5	511	2701
2017-01-02 09:40:00	511	511.95	510.5	511.35	4858
2017-01-02 09:41:00	511.35	512	510.6	511.95	3939
2017-01-02 09:42:00	512	513	512	512.6	8538

Step 3: Set Up Google Cloud Storage (GCS)

- Create a Cloud Storage bucket for storing images:
 - Go to **Cloud Storage > Buckets**.
 - Create a bucket (e.g., **stock-oppe**) with the appropriate region and permissions

Machine configuration

Name * stock-oppe

Region * us-central1 (Iowa) Zone * Any

Region is permanent. Google will choose a zone on your behalf, maximizing VM obtainability. Zone is permanent.

☒ General purpose ☐ Compute optimized ☐ Memory optimized ☐ Storage optimized ☐ GPUs

Machine types for common workloads, optimized for cost and flexibility

Series	Description	vCPUs	Memory	Platform
<input type="radio"/> C4	Consistently high performance	2 - 192	4 - 1,488 GB	Intel Emerald Rapids
<input type="radio"/> C4A	Arm-based consistently high performance	1 - 72	2 - 576 GB	Google Axion
<input type="radio"/> N4	Flexible & cost-optimized	2 - 80	4 - 640 GB	Intel Emerald Rapids
<input type="radio"/> C3	Consistently high performance	4 - 192	8 - 1,536 GB	Intel Sapphire Rapids
<input type="radio"/> C3D	Consistently high performance	4 - 360	8 - 2,880 GB	AMD Genoa
<input checked="" type="radio"/> E2	Low cost, day-to-day computing	0.25 - 32	1 - 128 GB	Based on availability
<input type="radio"/> N2	Balanced price & performance	2 - 128	2 - 864 GB	Intel Cascade Lake
<input type="radio"/> N2D	Balanced price & performance	2 - 224	2 - 896 GB	AMD EPYC
<input type="radio"/> T2A	Scale-out workloads	1 - 48	4 - 192 GB	Ampere Altra Arm

Monthly estimate
\$25.46
That's about \$0.03 hourly
Pay for what you use: no upfront costs and per second billing

Item	Monthly estimate
2 vCPU + 4 GB memory	\$24.46
10 GB balanced persistent disk	\$1.00
Total	\$25.46

[Compute Engine pricing](#)
[LESS](#)

- Upload some test images to the bucket.
- Upload `install_packages.sh`

Cloud Storage **Bucket details** **stock-oppe**

Location: us (multiple regions in United States) Storage class: Standard Public access: Not public Protection: Soft Delete

OBJECTS CONFIGURATION PERMISSIONS PROTECTION LIFECYCLE OBSERVABILITY INVENTORY REPORTS OPERATIONS

Folder browser

Buckets > stock-oppe

CREATE FOLDER UPLOAD TRANSFER DATA OTHER SERVICES

Filter by name prefix only Filter objects and folders Show Live objects only

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption
No rows to display								

Uploads and week-6-ibd operations

- Uploading 25 files
 - ABFRL_EQ_NSE_NSE_MI_NUTE.csv
 - ADANIGAS_EQ_NSE_NSE_MINUTE.csv
 - ASHOKLEY_EQ_NSE_NSE_MINUTE.csv

Upload started

Step 4: Create a Virtual Machine (VM):

- Navigate to **Compute Engine > VM Instances**.
- Click **Create Instance** “stock-oppe”
- Choose:
 - Machine type (e.g., e2-medium for basic setups).
 - Operating system (e.g., Ubuntu 22.04 LTS).
- Configure networking to allow external access:
 - Under **Firewall**, check "Allow HTTP traffic" and "Allow HTTPS traffic".

Google Cloud console screenshot showing the 'Create an instance' page. The 'Networking' tab is selected, showing firewall rules configuration. The 'Firewall' section has checkboxes for 'Allow HTTP traffic', 'Allow HTTPS traffic', and 'Allow Load Balancer Health Checks', all of which are checked. The 'Network tags' section shows 'kafka-server' as the selected tag. The 'Hostname' field is empty. The 'IP forwarding' section has an 'Enable' checkbox that is unchecked. The 'Network performance configuration' section has an 'Enable per VM Tier_1 networking performance' checkbox that is unchecked. The 'Monthly estimate' on the right shows a total of \$25.46 for 2 vCPU + 4 GB memory and 10 GB balanced persistent disk.

Step 5: Create a New Firewall Rule

- Click the **"Create Firewall Rule"** button at the top.
- Fill in the details for the new rule:
 - **Name:** Enter a descriptive name, such as **allow-kafka-9092**.
 - **Network:** Select the network your VM instance is part of (usually **default** unless you've set up a custom network).
 - **Priority:** Leave the default value (1000) unless you have specific needs.
 - **Direction of traffic:** Select **Ingress** (incoming traffic).
 - **Action on match:** Select **Allow**.
 - **Targets:**
 - Choose **Specified target tags**.
 - Add a **tag** (e.g., **kafka-server**) that you will assign to your VM later.
- **Source filter:**
 - Choose **IP ranges**.

- Enter the IP range allowed to access your Kafka server:
 - For open access: **0.0.0.0/0** (not recommended for production environments due to security risks).
 - For restricted access: Use a specific IP or range, e.g., **192.168.1.0/24**.
- **Protocols and ports:**
 - Select **Specified protocols and ports**.
 - Check **tcp** and specify port **9092**.
- **Save the Firewall Rule**
 - Click the **Create** button to save the rule.

Google Cloud week-6-ibd Search (/) for resources, docs, products, and more Search

Network Security

Secure Web Proxy

Cloud Armor

- DDoS Dashboard
- Cloud Armor policies
- Adaptive Protection
- Cloud Armor Service Tier

Cloud IDS

- IDS Dashboard
- IDS Endpoints
- IDS Threats

Cloud NGFW

- Dashboard
- Firewall policies**
- Threats

Targets

Target tags kafka-server

Source filters

IP ranges 0.0.0.0/0

Protocols and ports

tcp:9092

Enforcement

Enabled

Insights

None

Hit count monitoring

-

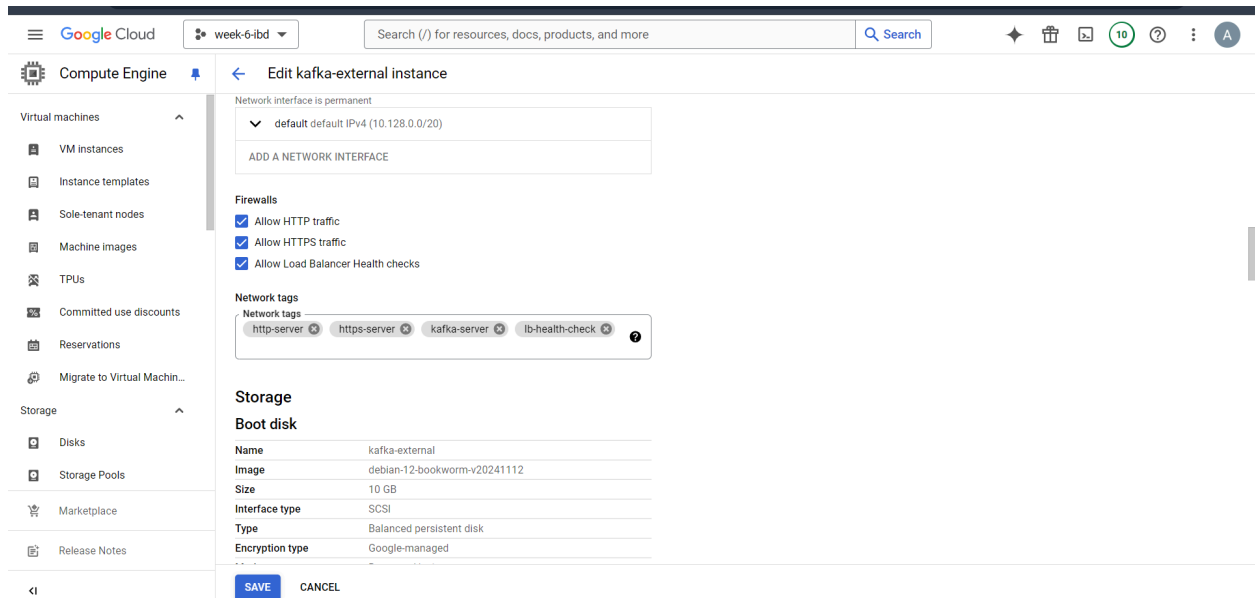
Applicable to instances

The following table does not show any App Engine flexible environment instances

Filter Filter by instance name, project or subnetwork

Name	Subnetwork	Internal IP ranges	External IP ranges	Tags	Service accounts	Project	Labels	Network direction
kafka-external	default	10.128.0.16	34.66.209.84	http-	278755983826-	week-6-ibd		VIEW DE

- **Tag Your VM with the Firewall Rule**
 - Go to **Compute Engine > VM Instances**.
 - Find the VM running Kafka.
 - Click the **Edit** button at the top of the VM details page.
 - In the **Network tags** section, add the same tag you used in the firewall rule (e.g., **kafka-server**).
 - Click **Save**.



Step 6: Install Kafka and Dependencies

- **Connect to the VM: SSH into the VM**
 - Install Java: Kafka requires Java. Install OpenJDK:


```
sudo apt update
sudo apt install -y default-jdk
java -version
```
- **Download and Install Kafka:**
 - Navigate to [Apache Kafka Downloads](https://kafka.apache.org/downloads).
 - Download Kafka:


```
Wget
https://downloads.apache.org/kafka/3.5.1/kafka_2.13-3.5.1.tgz
```
 - Extract the Kafka archive:

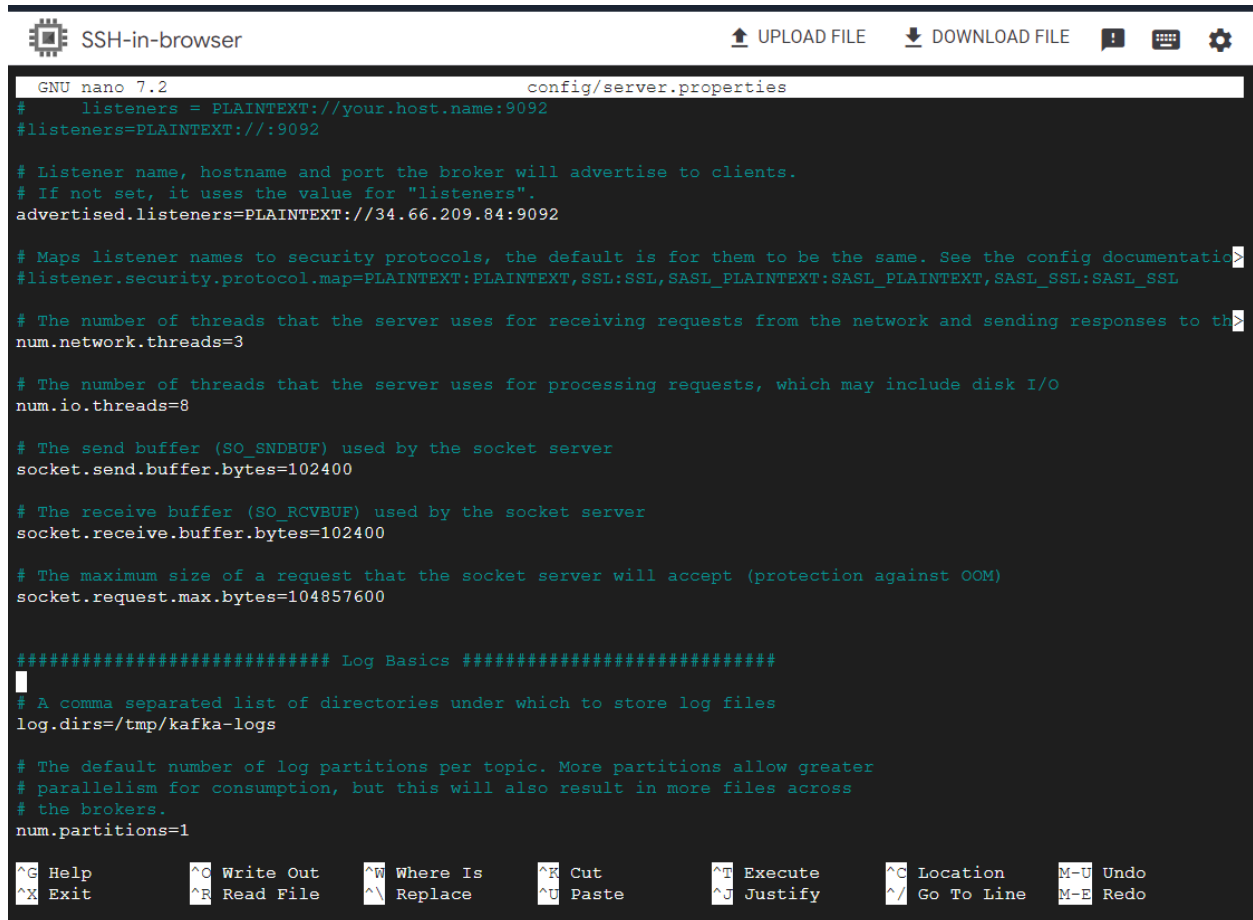

```
tar -xvzf kafka_2.13-3.5.1.tgz
mv kafka_2.13-3.5.1 kafka
```
- **Configure Kafka:**
 - Open Kafka configuration file:


```
nano kafka/config/server.properties
```
 - Set the following:
 - Advertised Hostname:


```
advertised.listeners=PLAINTEXT://34.28.48.166:9092
```
 - Zookeeper (default):


```
zookeeper.connect=localhost:2181
```

- Save and exit.



```
GNU nano 7.2 config/server.properties
# listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092

# Listener name, hostname and port the broker will advertise to clients.
# If not set, it uses the value for "listeners".
advertised.listeners=PLAINTEXT://34.66.209.84:9092

# Maps listener names to security protocols, the default is for them to be the same. See the config documentation
#listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL

# The number of threads that the server uses for receiving requests from the network and sending responses to the
num.network.threads=3

# The number of threads that the server uses for processing requests, which may include disk I/O
num.io.threads=8

# The send buffer (SO_SNDBUF) used by the socket server
socket.send.buffer.bytes=102400

# The receive buffer (SO_RCVBUF) used by the socket server
socket.receive.buffer.bytes=102400

# The maximum size of a request that the socket server will accept (protection against OOM)
socket.request.max.bytes=104857600

##### Log Basics #####
# A comma separated list of directories under which to store log files
log.dirs=/tmp/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=1

^G Help      ^C Write Out  ^W Where Is   ^R Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo
```

Step 7: Set Up and Start Zookeeper

- Kafka requires Zookeeper for coordination.
- Start Zookeeper:
`kafka/bin/zookeeper-server-start.sh`
`kafka/config/zookeeper.properties &`

- Start Kafka Broker:
`kafka/bin/kafka-server-start.sh kafka/config/server.properties &`

```
[2024-12-07 19:31:03,066] INFO [GroupCoordinator 0]: Startup complete. (kafka.coordinator.group.GroupCoordinator)
[2024-12-07 19:31:03,127] INFO [TransactionCoordinator id=0] Starting up. (kafka.coordinator.transaction.TransactionCoordinator)
[2024-12-07 19:31:03,145] INFO [TransactionCoordinator id=0] Startup complete. (kafka.coordinator.transaction.TransactionCoordinator)
[2024-12-07 19:31:03,154] INFO [TxnMarkerSenderThread-0]: Starting (kafka.coordinator.transaction.TransactionMarkerChannelManager)
[2024-12-07 19:31:03,281] INFO [Controller id=0, targetBrokerId=0] Node 0 disconnected. (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,284] WARN [Controller id=0, targetBrokerId=0] Connection to node 0 (/34.66.209.84:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,287] INFO [ExpirationReaper-0-AlterAcls]: Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2024-12-07 19:31:03,292] INFO [Controller id=0, targetBrokerId=0] Client requested connection close from node 0 (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,388] INFO [/config/changes-event-process-thread]: Starting (kafka.common.ZkNodeChangeNotificationListener$ChangeEventProcessThread)
[2024-12-07 19:31:03,399] INFO [Controller id=0, targetBrokerId=0] Node 0 disconnected. (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,401] WARN [Controller id=0, targetBrokerId=0] Connection to node 0 (/34.66.209.84:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,402] INFO [Controller id=0, targetBrokerId=0] Client requested connection close from node 0 (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,421] INFO [SocketServer listenerType=ZK_BROKER, nodeId=0] Enabling request processing. (kafka.network.SocketServer)
[2024-12-07 19:31:03,426] INFO Awaiting socket connections on 0.0.0.0:9092. (kafka.network.DataPlaneAcceptor)
[2024-12-07 19:31:03,439] INFO Kafka version: 3.5.1 (org.apache.kafka.common.utils.AppInfoParser)
[2024-12-07 19:31:03,440] INFO Kafka commitId: 2c6fb6c54472e90a (org.apache.kafka.common.utils.AppInfoParser)
[2024-12-07 19:31:03,440] INFO Kafka startTimeMs: 1733599863433 (org.apache.kafka.common.utils.AppInfoParser)
[2024-12-07 19:31:03,442] INFO [KafkaServer id=0] started (kafka.server.KafkaServer)
[2024-12-07 19:31:03,598] INFO [zk-broker-0-to-controller-forwarding-channel-manager]: Recorded new controller, from now on will use node 34.66.209.84:9092 (id: 0 rack: null) (kafka.server.BrokerToControllerRequestThread)
[2024-12-07 19:31:03,639] INFO [zk-broker-0-to-controller-alter-partition-channel-manager]: Recorded new controller, from now on will use node 34.66.209.84:9092 (id: 0 rack: null) (kafka.server.BrokerToControllerRequestThread)
[2024-12-07 19:31:03,708] INFO [ReplicaFetcherManager on broker 0] Removed fetcher for partitions Set(image-topic-0) (kafka.server.ReplicaFetcherManager)
[2024-12-07 19:31:03,752] INFO [Partition image-topic-0 broker=0] Log loaded for partition image-topic-0 with initial high watermark 815 (kafka.cluster.Partition)
```

- Create a topic:
`kafka/bin/kafka-topics.sh --create --topic stock-oppe --bootstrap-server 34.28.48.166:9092`
- List topics:
`kafka/bin/kafka-topics.sh --list --bootstrap-server 34.28.48.166:9092`

Step 9: Kafka Producer

- Write a **Kafka producer** script and upload it to the VM ssh
- Install the python virtual environment and activate it
 - `sudo apt update`
 - `sudo apt install python3-venv`
 - `python3 -m venv env`
 - `source env/bin/activate`
- Install the Kafka Python client:
`pip install kafka-python google-cloud-storage pandas`
- Run the producer.py
 - `Python3 producer.py`

```

1  from google.cloud import storage
2  import pandas as pd
3  from kafka import KafkaProducer
4  import json
5  import time
6  from io import BytesIO
7
8
9  producer = KafkaProducer(
10     bootstrap_servers='34.28.48.166:9092',
11     acks='all',
12     value_serializer=lambda v: json.dumps(v).encode('utf-8') # Ensure it's properly serialized
13 )
14
15 client = storage.Client()
16 bucket = client.bucket("stock-oppe")
17 blobs = bucket.list_blobs(prefix="NSE_Stocks_Data/")
18
19 csv_files = {}
20 for blob in blobs:
21     if blob.name.endswith(".csv"):
22         print(f"Downloading: {blob.name}")
23         content = blob.download_as_bytes()
24         df = pd.read_csv(BytesIO(content))
25         csv_files[blob.name] = df
26
27 dataframes = {}
28
29 for file_name, df in csv_files.items():
30     company_name = file_name.split("/")[-1]
31     company_name = company_name.split("_")[0]
32     df['company'] = company_name
33     dataframes[file_name] = df
34
35 # Initialize sliding windows for each CSV file
36 windows = {file_name: [] for file_name in csv_files}
37 indices = {file_name: 0 for file_name in csv_files}
38
39 batch_number = 0
40 while True:
41     batch_number += 1
42     print(f"Producing batch #{batch_number}...")
43

```

```

for file_name, df in dataframes.items():
    current_index = indices[file_name]
    if current_index >= len(df):
        continue
    windows[file_name].append(df.iloc[current_index].to_dict())
    if len(windows[file_name]) > 10:
        windows[file_name].pop(0)

    indices[file_name] += 1
    windowed_df = pd.DataFrame(windows[file_name])
    windowed_data = windowed_df.groupby('company').agg(
        avg_volume=('volume', 'mean'),
        last_close=('close', 'max')
    ).reset_index()

    merged_df = windowed_df.merge(windowed_data, on='company', how='left')

    merged_df['A1'] = abs((merged_df['close'] - merged_df['last_close']) / merged_df['last_close']) > 0.005
    merged_df['A2'] = merged_df['volume'] > merged_df['avg_volume'] * 1.02
    merged_df['is_anomaly'] = merged_df['A1'] | merged_df['A2']

    anomalies = merged_df[merged_df['is_anomaly']]

    for index, row in anomalies.iterrows():
        record = {
            "company": row["company"],
            "open": row["open"],
            "high": row["high"],
            "low": row["low"],
            "close": row["close"],
            "volume": row["volume"],
            "avg_volume": row["avg_volume"],
            "last_close": row["last_close"],
            "A1": row["A1"],
            "A2": row["A2"],
            "is_anomaly": row["is_anomaly"]
        }
        print(f"Sending record: {record}")
        producer.send("stock-oppe", record)

if all(indices[file_name] >= len(dataframes[file_name]) for file_name in csv_files):
    print("All data processed. Exiting.")
    break

```

```

if all(indices[file_name] >= len(dataframes[file_name]) for file_name in csv_files):
    print("All data processed. Exiting.")
    break

producer.flush()
time.sleep(20)

producer.close()

```

Step 10: Create a Dataproc Cluster

- Go to **Dataproc > Clusters > Create Cluster**.
- Choose:
 - Cluster type: **Standard (1 master, N workers)**.
 - Machine type: **e2-standard-4** (adjust based on workload).
 - Enable components like **Jupyter Notebook** if needed.
 - Add **Initialization actions** with
`gs://stock-oppe/install_packages.sh`

```
#!/bin/bash
# install_packages.sh

sudo apt-get update
# Install pip (if not already installed)
sudo apt-get install python3-pip -y

# Install required Python packages
pip3 install kafka-python
```

- Check the connection
 - telnet 34.28.48.166 9092

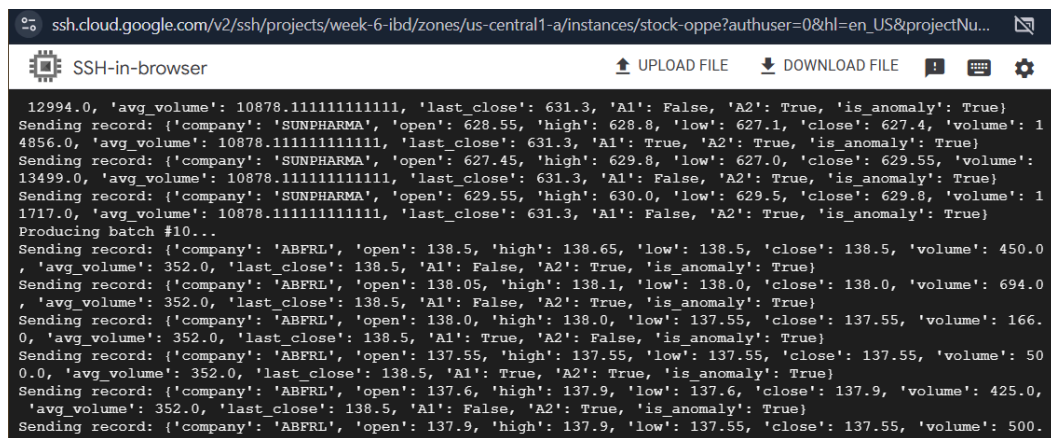
```
avijeetcloud@stock-oppe-m:~$ telnet 34.28.48.166 9092
Trying 34.28.48.166...
Connected to 34.28.48.166.
Escape character is '^]'.
```

- Write a **Kafka consumer “Stream.py”** script and uploads it to the VM ssh.
- Run the stream.py
 - Python3 stream.py

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import from_json, col
3 from pyspark.sql.types import StructType, StructField, StringType, DoubleType, IntegerType, BooleanType
4
5 spark = SparkSession.builder \
6     .appName("StockAnomalyDetection") \
7     .config("spark.jars.packages", "org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.1") \
8     .getOrCreate()
9
10 json_schema = StructType([
11     StructField("company", StringType(), True),
12     StructField("open", DoubleType(), True),
13     StructField("high", DoubleType(), True),
14     StructField("low", DoubleType(), True),
15     StructField("close", DoubleType(), True),
16     StructField("volume", IntegerType(), True),
17     StructField("avg_volume", DoubleType(), True),
18     StructField("last_close", DoubleType(), True),
19     StructField("A1", BooleanType(), True),
20     StructField("A2", BooleanType(), True),
21     StructField("is_anomaly", BooleanType(), True)
22 ])
23
24 stock_data_stream = spark.readStream \
25     .format("kafka") \
26     .option("kafka.bootstrap.servers", "34.28.48.166:9092") \
27     .option("subscribe", "stock-oppe") \
28     .option("startingOffsets", "latest") \
29     .load() \
30     .selectExpr("CAST(value AS STRING) as value")
31
32 anomalies = stock_data_stream \
33     .select(from_json(col("value"), json_schema).alias("data")) \
34     .select("data.*") \
35     .filter(col("data").isNotNull())
36
37 query = anomalies.writeStream \
38     .outputMode("append") \
39     .format("console") \
40     .option("truncate", "false") \
41     .trigger(processingTime="10 seconds") \
42     .start()
43
44 query.awaitTermination()
```

Step 11: Result and Testing the Pipeline

- **Kafka Testing:**
 - Use the Kafka producer script to push test messages to the Kafka topic.
- **Monitor Outputs:**
 - Check the cluster ssh terminal for real-time predictions.
 - Verify the console output manually checking the dataset



The screenshot shows a terminal window titled "SSH-in-browser" with a URL bar indicating a connection to a Google Cloud project. The terminal displays the output of a Kafka producer script, showing a series of JSON records being sent to a Kafka topic. The records include fields for 'company', 'open', 'high', 'low', 'close', 'volume', 'avg_volume', 'last_close', 'A1', 'A2', and 'is_anomaly'. The output is formatted as a continuous stream of text, with each record on a new line. The records are for the company 'SUNPHARMA' and 'ABFRL', with various stock price and volume data points. The 'is_anomaly' field is set to 'True' for all records shown.

```
12994.0, 'avg_volume': 10878.111111111111, 'last_close': 631.3, 'A1': False, 'A2': True, 'is_anomaly': True}
Sending record: {'company': 'SUNPHARMA', 'open': 628.55, 'high': 628.8, 'low': 627.1, 'close': 627.4, 'volume': 1
4856.0, 'avg_volume': 10878.111111111111, 'last_close': 631.3, 'A1': True, 'A2': True, 'is_anomaly': True}
Sending record: {'company': 'SUNPHARMA', 'open': 627.45, 'high': 629.8, 'low': 627.0, 'close': 629.55, 'volume':
13499.0, 'avg_volume': 10878.111111111111, 'last_close': 631.3, 'A1': False, 'A2': True, 'is_anomaly': True}
Sending record: {'company': 'SUNPHARMA', 'open': 629.55, 'high': 630.0, 'low': 629.5, 'close': 629.8, 'volume': 1
1717.0, 'avg_volume': 10878.111111111111, 'last_close': 631.3, 'A1': False, 'A2': True, 'is_anomaly': True}
Producing batch #10...
Sending record: {'company': 'ABFRL', 'open': 138.5, 'high': 138.65, 'low': 138.5, 'close': 138.5, 'volume': 450.0
, 'avg_volume': 352.0, 'last_close': 138.5, 'A1': False, 'A2': True, 'is_anomaly': True}
Sending record: {'company': 'ABFRL', 'open': 138.05, 'high': 138.1, 'low': 138.0, 'close': 138.0, 'volume': 694.0
, 'avg_volume': 352.0, 'last_close': 138.5, 'A1': False, 'A2': True, 'is_anomaly': True}
Sending record: {'company': 'ABFRL', 'open': 138.0, 'high': 138.0, 'low': 137.55, 'close': 137.55, 'volume': 166.
0, 'avg_volume': 352.0, 'last_close': 138.5, 'A1': True, 'A2': False, 'is_anomaly': True}
Sending record: {'company': 'ABFRL', 'open': 137.55, 'high': 137.55, 'low': 137.55, 'close': 137.55, 'volume': 50
0.0, 'avg_volume': 352.0, 'last_close': 138.5, 'A1': True, 'A2': True, 'is_anomaly': True}
Sending record: {'company': 'ABFRL', 'open': 137.6, 'high': 137.9, 'low': 137.6, 'close': 137.9, 'volume': 425.0,
'avg_volume': 352.0, 'last_close': 138.5, 'A1': False, 'A2': True, 'is_anomaly': True}
Sending record: {'company': 'ABFRL', 'open': 137.9, 'high': 137.9, 'low': 137.55, 'close': 137.55, 'volume': 500.
, 'avg_volume': 352.0, 'last_close': 138.5, 'A1': True, 'A2': True, 'is_anomaly': True}
```



SSH-in-browser

↑ UPLOAD FILE

↓ DOWNLOAD FILE



```
|ASHOKLEY|79.6|79.85|79.6|79.65|NULL|27517.0|80.35|true|false|true|
|ASHOKLEY|79.65|79.75|79.55|79.55|NULL|27517.0|80.35|true|false|true|
|BAJAJ|2627.0|2646.1|2612.35|2612.35|NULL|3352.0|2627.0|true|false|true|
|BAJAJ|2614.45|2614.45|2591.3|2596.0|NULL|3352.0|2627.0|true|true|true|
|BAJAJ|2596.0|2596.0|2587.75|2590.8|NULL|3352.0|2627.0|true|false|true|
|BAJAJ|2593.0|2596.95|2584.0|2589.95|NULL|3352.0|2627.0|true|true|true|
```

only showing top 20 rows

Batch: 10

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|company|open|high|low|close|volume|avg_volume|last_close|A1|A2|is_anomaly|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|ABFRL|138.05|138.1|138.0|138.0|NULL|307.5|138.25|false|true|true|
|ABFRL|138.0|138.0|137.55|137.55|NULL|307.5|138.25|true|false|true|
|ABFRL|137.55|137.55|137.55|137.55|NULL|307.5|138.25|true|true|true|
|ABFRL|137.6|137.9|137.6|137.9|NULL|307.5|138.25|false|true|true|
|ABFRL|137.9|137.9|137.55|137.55|NULL|307.5|138.25|true|true|true|
|ASHOKLEY|80.2|80.25|80.0|80.0|NULL|22860.6|80.3|false|true|true|
|ASHOKLEY|79.95|79.95|79.75|79.75|NULL|22860.6|80.3|true|false|true|
|ASHOKLEY|79.75|79.8|79.35|79.35|NULL|22860.6|80.3|true|true|true|
|ASHOKLEY|79.45|79.55|79.35|79.55|NULL|22860.6|80.3|true|true|true|
|ASHOKLEY|79.55|79.65|79.55|79.6|NULL|22860.6|80.3|true|false|true|
|ASHOKLEY|79.6|79.85|79.6|79.65|NULL|22860.6|80.3|true|false|true|
|ASHOKLEY|79.65|79.75|79.55|79.55|NULL|22860.6|80.3|true|false|true|
|ASHOKLEY|79.55|79.7|79.55|79.6|NULL|22860.6|80.3|true|false|true|
|BAJAJ|2614.45|2614.45|2591.3|2596.0|NULL|3261.3|2612.35|true|true|true|
|BAJAJ|2596.0|2596.0|2587.75|2590.8|NULL|3261.3|2612.35|true|false|true|
|BAJAJ|2593.0|2596.95|2584.0|2589.95|NULL|3261.3|2612.35|true|true|true|
|BAJAJ|2587.95|2589.35|2583.0|2583.15|NULL|3261.3|2612.35|true|false|true|
|BAJAJ|2583.0|2596.95|2581.0|2591.4|NULL|3261.3|2612.35|true|true|true|
|BAJAJ|2591.4|2599.0|2591.4|2599.0|NULL|3261.3|2612.35|true|false|true|
|BAJFINANCE|854.05|854.4|851.55|852.0|NULL|5706.6|854.05|false|true|true|
```

only showing top 20 rows

Conclusion

The successful implementation of real-time anomaly detection in historical stock trading data using Spark Streaming and Kafka demonstrates the practical integration of big data technologies to address financial irregularities. By creating a robust data pipeline that ingests, processes, and analyzes minute-level trading data, we effectively identified significant price deviations and volume spikes that could signal suspicious trading activities.

This project not only highlights the capabilities of distributed systems in managing large-scale data but also showcases their potential to enable proactive fraud detection mechanisms. The detailed steps, from setting up infrastructure in Google Cloud Platform to testing and validating the pipeline, ensure a scalable and reliable solution.

Future enhancements could include incorporating machine learning models for anomaly classification, expanding the scope of detection parameters, and automating the pipeline for broader financial applications. This assignment underscores the importance of big data analytics in safeguarding the integrity of financial markets.