

Oppe 2: Stock Analysis Pipeline

Submitted by Shramana Sinha, 23f1002703

Setup

1. Created a gcs bucket for this assignment.

📁 oppe_2

| | | | |
|---------------------------|---------------|---------------|-------------|
| Location | Storage class | Public access | Protection |
| us-east1 (South Carolina) | Standard | Not public | Soft Delete |

Objects Configuration Permissions Protection Lifecycle Observability **New** Inventory Reports

Folder browser [↶](#)

📁 oppe_2

Buckets > oppe_2 📁

Create folder Upload ▾ Transfer data ▾ Other services ▾

Filter by name prefix only ▾ [Filter](#) Filter objects and folders

| <input type="checkbox"/> | Name | Size | Type | Cre |
|--------------------------|--|---------|----------|-----|
| <input type="checkbox"/> | AARTIIND_EQ_NSE_NSE_MIN... | 21.2 MB | text/csv | Apr |
| <input type="checkbox"/> | ABCAPITAL_EQ_NSE_NSE_MI... | 17.4 MB | text/csv | Apr |
| <input type="checkbox"/> | ADANIENT_EQ_NSE_NSE_MIN... | 21.8 MB | text/csv | Apr |
| <input type="checkbox"/> | ADANIPOINTS_EQ_NSE_NSE_... | 21.8 MB | text/csv | Apr |
| <input type="checkbox"/> | APLLTD_EQ_NSE_NSE_MINUT... | 20.9 MB | text/csv | Apr |
| <input type="checkbox"/> | APOLLOHOSP_EQ_NSE_NSE_... | 22.9 MB | text/csv | Apr |
| <input type="checkbox"/> | ASIANPAINT_EQ_NSE_NSE_M... | 23 MB | text/csv | Apr |
| <input type="checkbox"/> | AUBANK_EQ_NSE_NSE_MINU... | 18.6 MB | text/csv | Apr |

Figure: Google Cloud Storage bucket containing stock data CSV files

2. Spun up a vm to run the zookeeper and kafka servers.

VM instances

[Filter](#) Enter property name or value ⓘ

| <input type="checkbox"/> | Status | Name ↑ | Zone | Recommendations | In use by | Internal IP | External IP | Connect |
|--------------------------|--------|--|---------------|-----------------|-----------|--------------------------------------|---|---------|
| <input type="checkbox"/> | ✓ | kafka-vm | us-central1-c | | | 10.128.0.10 (nic0) | 104.154.133.10 (nic0) | SSH ▾ ⋮ |
| <input type="checkbox"/> | ✓ | spark-cluster-m | us-east1-d | | | 10.142.0.3 (nic0) | 34.139.84.214 (nic0) | SSH ▾ ⋮ |
| <input type="checkbox"/> | ✓ | spark-cluster-producer-m | us-east4-a | | | 10.150.0.4 (nic0) | 35.194.93.180 (nic0) | SSH ▾ ⋮ |
| <input type="checkbox"/> | ✓ | spark-cluster-w-0 | us-east1-d | | | 10.142.0.2 (nic0) | 35.237.230.56 (nic0) | SSH ▾ ⋮ |

Figure: Kafka VM instance (kafka-vm) hosting the message broker service

3. Spun up 2 dataproc clusters - 1 for producer and 1 for consumer.

| Clusters | | | | | | | |
|--|--|-----------|----------|------------|--------------------|---------------|-----------|
| + CREATE CLUSTER REFRESH ▶ START ■ STOP 🗑 DELETE REGIONS ▾ | | | | | | | |
| Filter Search cluster by properties, press Enter | | | | | | | |
| Sorry, the server was not able to fulfill your request. | | | | | | | |
| <input type="checkbox"/> | Name ↑ | Status | Region | Zone | Total worker nodes | Flexible VMs? | Scheduled |
| <input type="checkbox"/> | spark-cluster | ✓ Running | us-east1 | us-east1-d | 2 | No | Off |
| <input type="checkbox"/> | spark-cluster-producer | ✓ Running | us-east4 | us-east4-a | 0 | No | Off |

Figure: Dataproc clusters configured for Spark batch and streaming processing

- Created a google pub/sub topic.

| Subscriptions | | | | | |
|--|-------|--|---------------|---|----|
| + CREATE SUBSCRIPTION DELETE | | | | | |
| LIST METRICS | | | | | |
| Filter Filter subscriptions | | | | | |
| <input type="checkbox"/> | State | Subscription ID ↑ | Delivery type | Topic name | Ac |
| <input type="checkbox"/> | ✓ | CountLineSub | Pull | ⚠ Deleted topic | 10 |
| <input type="checkbox"/> | ✓ | eventarc-us-central1-trigger-rambqbj-sub-271 | Push | ⚠ Deleted topic | 10 |
| <input type="checkbox"/> | ✓ | StockVolumeAnomalies-sub | Pull | projects/celtic-guru-448518-... | 10 |

Figure: Google PubSub topic "StockVolumeAnomalies" for anomaly notifications

- Downloaded the files from the google drive folder to the google cloud shell using the below code:

```
Python
import os
from google.oauth2 import service_account
from googleapiclient.discovery import build
from googleapiclient.http import MediaIoBaseDownload
import io
folder_id = "1bt94v6pTR6s58c8HFX1FU_9zMoXFX4cf"
credentials = service_account.Credentials.from_service_account_file(
    'service-account-key.json',
    scopes=['https://www.googleapis.com/auth/drive.readonly']
)
```

```

)
service = build('drive', 'v3', credentials=credentials)
def download_folder(folder_id, output_dir='.'):
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    results = service.files().list(
        q=f"'{folder_id}' in parents",
        fields="files(id, name, mimeType)").execute()
    items = results.get('files', [])
    if not items:
        print(f'No files found in folder {folder_id}')
        return
    for item in items:
        if item['mimeType'] == 'application/vnd.google-apps.folder':
            subfolder_path = os.path.join(output_dir, item['name'])
            print(f"Processing subfolder: {item['name']}")
            download_folder(item['id'], subfolder_path)
        else:
            print(f"Downloading file: {item['name']}")
            request = service.files().get_media(fileId=item['id'])

            file_path = os.path.join(output_dir, item['name'])
            with io.FileIO(file_path, 'wb') as f:
                downloader = MediaIoBaseDownload(f, request)
                done = False
                while not done:
                    status, done = downloader.next_chunk()
                    print(f"Download {int(status.progress() * 100)}%")
download_folder(folder_id, 'downloaded_files')

```

6. Uploaded the csv files from the google cloud shell local storage to the gcs bucket.

Implementation Details

Batch Processing Implementation

1. Data Loading:

- Reads from GCS using the path pattern `gs://oppe_2/*.csv`

Python

```
gcs_path = "gs://oppe_2/*.csv"
```

- Uses Spark's CSV reader with `inferSchema` to automatically detect data types

Python

```
df = spark.read.option("header", "true").option("inferSchema",  
"true").csv(gcs_path)
```

2. Data Cleaning:

- Filters out rows with null values in the columns using boolean expressions

Python

```
df_clean = df.filter(  
    col("timestamp").isNotNull()  
    & col("close").isNotNull()  
    & col("open").isNotNull()  
    & col("high").isNotNull()  
    & col("low").isNotNull()  
    & col("volume").isNotNull()  
)
```

- Explicitly casts columns to appropriate types to ensure data consistency

Python

```
df_clean = (  
    df_clean.withColumn("timestamp", col("timestamp").cast("timestamp"))  
    .withColumn("close", col("close").cast("double"))  
    .withColumn("open", col("open").cast("double"))  
    .withColumn("high", col("high").cast("double"))  
    .withColumn("low", col("low").cast("double"))  
    .withColumn("volume", col("volume").cast("long"))  
)
```

3. Data Enrichment:

- Uses `input_file_name()` function to extract the source filename. This allows tracking which file each stock record came from.

Python

```
df_clean = df_clean.withColumn("stock_ticker", F.input_file_name())
```

4. Data Sorting:

- Orders data by stock ticker and timestamp to ensure time-series analysis accuracy

Python

```
df_sorted = df_clean.orderBy("stock_ticker", "timestamp")
```

5. Kafka Integration:

- Uses `selectExpr` to create the key-value structure needed for Kafka
- Serializes the entire row structure as JSON using `to_json(struct(*))`
- Uses stock ticker as the Kafka message key for partitioning
- Uses Spark's Kafka connector to write directly to the topic

Python

```
df_sorted.selectExpr(  
    "stock_ticker AS key",  
    "to_json(struct(*)) AS value"  
)  
.write(  
    .format("kafka") \\  
    .option("kafka.bootstrap.servers", "104.154.133.10:9092") \\  
    .option("topic", "stock-trades") \\  
    .save())
```

```
op_max_list=2, op_get_file_status_mean=33, op_get_file_status_min=30, op_get_file_status_req=33, op_list_status_duration=63, op_list_status_max=34, op_list_status_mean=31, op_list_status_req=73, stream_write_bytes=116425, uptimeSeconds=35}  
[CONTEXT ratelimit_period="5 MINUTES" ]  
25/04/06 14:54:43 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimit=33764668-1ba4uxrh/a5f684d8-55c3-4202-890c-8ba245fef1b8/spark-job-history/application_1743926...)  
Data successfully loaded to Kafka  
25/04/06 14:55:22 INFO DataprocSparkPlugin: Shutting down driver plugin. metrics={action_http_op_open=0, action_http_delete_request=7, gcs_api_time=5440, gcs_backoff_count=0, gcs_api_client_request_count=0, gs_filesystem_create=4, exception_count=0, gcs_exception_count=0, gcs_api_total_request_count=0, action_http_put_request=8, op_create_non_recursive=0, gcs_api_client_operations=0, gcs_list_dir_request=2, stream_read_operations=0, gcs_api_client_request_time=67, op_xattr_list=0, op_get_delegation_token=0, gcs_api_server_unavailable_count=0, director_gcs_list_file_request=5, op_hsync=0, action_http_get_request=0, stream_read_operations_increased_status=0, gcs_api_client_requested_range_not_satisfiable_count=0, op_hflush=35, op_list_status_error_count=31, op_get_file_checksum=0, gcs_api_server_internal_error_count=0, stream_read...
```

Figure: Producer job execution showing successful data loading to Kafka

```
{  
  "timestamp": "2017-01-02T04:37:00.000Z",  
  "open": 344.0, "high": 344.85, "low": 343.5, "close": 344.5, "volume": 290, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:38:00.000Z",  
  "open": 344.5, "high": 344.5, "low": 344.5, "close": 344.5, "volume": 0, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:39:00.000Z",  
  "open": 344.5, "high": 344.5, "low": 344.5, "close": 344.5, "volume": 5, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:40:00.000Z",  
  "open": 344.5, "high": 344.5, "low": 344.5, "close": 344.5, "volume": 32, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:41:00.000Z",  
  "open": 344.5, "high": 344.5, "low": 344.5, "close": 344.5, "volume": 0, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:42:00.000Z",  
  "open": 344.5, "high": 344.5, "low": 344.5, "close": 344.5, "volume": 147, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:43:00.000Z",  
  "open": 344.5, "high": 345.0, "low": 342.0, "close": 342.0, "volume": 175, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:44:00.000Z",  
  "open": 342.0, "high": 342.0, "low": 341.0, "close": 341.0, "volume": 2, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:45:00.000Z",  
  "open": 341.0, "high": 341.0, "low": 341.0, "close": 341.0, "volume": 0, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:46:00.000Z",  
  "open": 341.0, "high": 341.0, "low": 341.0, "close": 341.0, "volume": 0, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:47:00.000Z",  
  "open": 341.0, "high": 343.0, "low": 341.0, "close": 343.0, "volume": 1, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:48:00.000Z",  
  "open": 343.0, "high": 343.0, "low": 343.0, "close": 343.0, "volume": 0, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:49:00.000Z",  
  "open": 343.0, "high": 343.0, "low": 343.0, "close": 343.0, "volume": 0, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:50:00.000Z",  
  "open": 343.0, "high": 343.45, "low": 343.0, "close": 343.45, "volume": 1, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
{  
  "timestamp": "2017-01-02T04:51:00.000Z",  
  "open": 343.45, "high": 343.45, "low": 343.45, "close": 343.45, "volume": 0, "stock_ticker": "gs://oppe-2/AARTIIND_EQ_NSE_NSE_MINUTE.csv"}  
}
```

Figure: Kafka topic verification showing successful message ingestion from the producer

Streaming Analytics Implementation

1. Configuration and Setup

- Creates Spark session and configure it with graceful shutdown to properly close resources and optimize the shuffle partitions for better parallelism

Python

```
spark = SparkSession \
    .builder \
    .appName("StockAnomalyDetection-Improved") \
    .config("spark.streaming.stopGracefullyOnShutdown", "true") \
    .config("spark.sql.shuffle.partitions", "8") \
    .getOrCreate()
```

- Defines a strict schema to enforce data types during JSON parsing

Python

```
stock_schema = StructType([
    StructField("stock_ticker", StringType(), True),
    StructField("timestamp", TimestampType(), True),
    StructField("open", DoubleType(), True),
    StructField("high", DoubleType(), True),
    StructField("low", DoubleType(), True),
    StructField("close", DoubleType(), True),
    StructField("volume", LongType(), True)
])
```

- Initializes Google PubSub client for publishing alerts

Python

```
PROJECT_ID = "celtic-guru-448518-f8"
TOPIC_NAME = "StockVolumeAnomalies"
publisher = pubsub_v1.PublisherClient()
topic_path = publisher.topic_path(PROJECT_ID, TOPIC_NAME)
```

2. Stream Reading and Parsing

- Reads from Kafka using Structured Streaming with configuration for:
 - Starting from earliest offset to process all historical data
 - Limiting each trigger to 500 records for controlled batch sizes

Python

```
raw_stream = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "104.154.133.10:9092") \
    .option("subscribe", "stock-trades") \
    .option("startingOffsets", "earliest") \
    .option("maxOffsetsPerTrigger", 500) \
    .load()
```

- Parses JSON data with explicit schema for type safety

Python

```
parsed_stream = raw_stream \
    .select(
        col("key").cast("string"),
        from_json(col("value").cast("string"), stock_schema).alias("data")
    ) \
    .select("key", "data.*")
```

- Applies a 15-minute watermark to handle late-arriving data

Python

```
watermarked_stream = parsed_stream \
    .withWatermark("timestamp", "15 minutes")
```

3. PubSub Publishing Function

- Creates a structured message with relevant anomaly details
- Converts to JSON and encodes as bytes for PubSub compatibility
- Returns the publish future for tracking completion

Python

```
def publish_to_pubsub(stock_ticker, timestamp, volume, avg_volume,
    volume_change_pct):
    message = {
        "stock_ticker": stock_ticker,
        "timestamp": str(timestamp),
        "volume": volume,
        "avg_volume": avg_volume,
        "volume_change_pct": volume_change_pct,
        "warning": f"Traded Volume more than 2% of its average"
    }
```

```

message_data = json.dumps(message).encode('utf-8')
future = publisher.publish(topic_path, data=message_data)
return future

```

4. Batch Processing Function with Anomaly Detection

- Performs early exit for empty batches to save processing time

```

Python
if batch_df.count() == 0:
    return

```

- Uses window functions to calculate:
 - Previous minute's close price using **lag**

```

Python
windowSpec = Window.partitionBy("stock_ticker").orderBy("timestamp")
with_prev_close = stock_data \
    .withColumn("prev_minute_close", lag("close", 1).over(windowSpec))

```

- 10-minute average volume using **avg**

```

Python
volume_window = Window.partitionBy("stock_ticker") \
    .orderBy("timestamp") \
    .rowsBetween(-10, -1)
with_metrics = with_prev_close \
    .withColumn("avg_10min_volume", avg("volume").over(volume_window))

```

- Implements anomaly detection logic with multiple conditions (discussed in the section below)
- Collects volume anomalies for PubSub publishing

```

Python
volume_anomalies = anomalies.filter(col("is_volume_anomaly") == True)
if volume_anomalies.count() > 0:
    print(f"Publishing {volume_anomalies.count()} volume anomalies to PubSub")
    volume_anomalies_list = volume_anomalies.select(
        "stock_ticker", "timestamp", "volume",
        "avg_10min_volume", "volume_change_pct"
    )

```



```

        ).collect()
    for row in volume_anomalies_list:
        publish_to_pubsub(
            row["stock_ticker"],
            row["timestamp"],
            row["volume"],
            row["avg_10min_volume"],
            row["volume_change_pct"]
        )

```

- Provides detailed console output of all the anomalies for monitoring

Python

```

if anomalies.count() > 0:
    print(f"Batch ID: {batch_id} - Found {anomalies.count()}
anomalies")
    anomalies.select(
        "stock_ticker", "timestamp", "close", "volume",
"prev_minute_close",
        "avg_10min_volume", "price_change_pct", "volume_change_pct",
"anomaly_type"
    ).show(truncate=False)

```

5. Stream Processing Execution

- Uses `foreachBatch` to apply the custom processing function, discussed above.
- Sets processing time trigger of 1 minute for regular batch execution
- Uses "update" output mode to process only new/updated data

Python

```

streaming_query = watermarked_stream \
    .writeStream \
    .foreachBatch(process_batch) \
    .outputMode("update") \
    .trigger(processingTime="1 minute") \
    .start()

```

- Waits for termination to keep the application running

Python

```

streaming_query.awaitTermination()

```

Anomaly Detection Implementation Breakdown

1. Price Change Calculation:

- Calculates percentage change from previous minute
- Handles null values with default of 0

Python

```
.withColumn("price_change_pct",  
            when(col("prev_minute_close").isNotNull(),  
                (col("close") - col("prev_minute_close")) /  
col("prev_minute_close") * 100)  
            .otherwise(lit(0)))
```

2. Volume Change Calculation:

- Calculates percentage change from 10-minute average
- Includes check for positive average to avoid division by zero
- Handles null values with default of 0

Python

```
.withColumn("volume_change_pct",  
            when(col("avg_10min_volume").isNotNull() &  
col("avg_10min_volume") > 0),  
                (col("volume") - col("avg_10min_volume")) /  
col("avg_10min_volume") * 100)  
            .otherwise(lit(0)))
```

3. Anomaly Classification:

- Price Anomalies (A1): Flags trades where absolute price change from the previous minute's trade close price (**price_change_pct**) exceeds 0.5%.

Python

```
.withColumn("is_price_anomaly",  
            when(col("prev_minute_close").isNotNull(),  
                F.abs(col("price_change_pct")) > 0.5)  
            .otherwise(lit(False)))
```

- Volume Anomalies (A2): Flags trades where volume exceeds average traded volume for the last 10 minutes (**volume_change_pct**) by 2% or more.

Python

```
.withColumn("is_volume_anomaly",
```

```
when(col("avg_10min_volume").isNotNull(),
      col("volume_change_pct") > 2.0)
      .otherwise(lit(False)))
```

- Defines the anomaly type of the data as **price**, **volume**, **price+volume** based on which kind of anomaly is true.

Python

```
.withColumn("anomaly_type",
             when(col("is_price_anomaly") & col("is_volume_anomaly"),
                 lit("Price+Volume"))
             .when(col("is_price_anomaly"), lit("Price"))
             .when(col("is_volume_anomaly"), lit("Volume"))
             .otherwise(lit("None")))
```

- Adds whether the data is anomaly by checking whether the **is_price_anomaly** or **is_volume_anomaly** column value is true or not and filters out the anomaly data based on it.

Python

```
.withColumn("is_anomaly",
            col("is_price_anomaly") | col("is_volume_anomaly"))
anomalies = anomaly_df.filter(col("is_anomaly") == True)
```

only showing top 20 rows

Publishing 138 volume anomalies to PubSub
Batch ID: 3 - Found 141 anomalies

| stock_ticker | timestamp | close | volume | prev_minute_close | avg_10min_volume | price_change_pct | volume_change_pct | anomaly_type |
|--|---------------------|--------|--------|-------------------|------------------|-----------------------|---------------------|--------------|
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-03 09:47:00 | 351.85 | 579 | 351.95 | 210.8 | -0.028413126864601762 | 174.66793168880454 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-03 09:57:00 | 351.1 | 161 | 351.0 | 117.4 | 0.028490028490034965 | 137.137989778534916 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-03 09:58:00 | 352.4 | 173 | 351.1 | 175.6 | 0.37026488180004397 | 128.83597883597886 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 07:38:00 | 353.7 | 161 | 354.7 | 136.0 | -0.28192839018889204 | 69.44444444444444 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 07:44:00 | 354.6 | 151 | 354.6 | 139.25 | 0.0 | 29.936305732484076 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 07:55:00 | 354.7 | 137 | 354.7 | 136.1 | 0.0 | 12.4930747922437635 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:00:00 | 354.5 | 156 | 354.7 | 136.1 | -0.05638567803777521 | 55.124653739612185 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:03:00 | 355.0 | 250 | 354.8 | 138.2 | 0.05636978579481078 | 554.4502617801047 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:05:00 | 355.6 | 1223 | 355.0 | 160.6 | 0.16901408450704866 | 1918.1518151815183 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:18:00 | 355.65 | 151 | 355.65 | 148.3 | 0.0 | 5.590062111801249 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:21:00 | 355.05 | 140 | 355.65 | 138.1 | -0.16870518768451173 | 4.986876640419943 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:30:00 | 354.75 | 287 | 355.65 | 136.5 | -0.25305778152677555 | 686.3013698630137 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:31:00 | 355.35 | 382 | 354.75 | 161.6 | 0.16913319238901275 | 520.12987012987 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:35:00 | 351.65 | 248 | 354.6 | 198.2 | -0.8319232938522406 | 152.54582484725051 | Price+Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:36:00 | 352.5 | 229 | 351.65 | 119.3 | 0.24171761694867702 | 91.95305951383068 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:38:00 | 352.45 | 1648 | 352.5 | 1138.6 | -0.014184397163123794 | 367.53246753246754 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:42:00 | 350.0 | 1734 | 351.5 | 164.4 | -0.42674253200568996 | 346.47201946472023 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:44:00 | 348.0 | 1491 | 349.9 | 1248.5 | -0.5430122892254865 | 197.58551307847083 | Price+Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:45:00 | 341.5 | 12185 | 348.0 | 1294.0 | -1.8678160919540232 | 643.1972789115647 | Price+Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:46:00 | 343.0 | 11376 | 341.5 | 1497.7 | -0.13923865300146414 | 1182.14066024195202 | Volume |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:47:00 | 345.45 | 1313 | 343.0 | 1602.4 | 0.714285714285711 | -48.04116865869854 | Price |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:49:00 | 346.75 | 1368 | 343.75 | 1574.0 | 0.8727272727272728 | -35.88850174216028 | Price |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE.csv | 2017-01-04 08:58:00 | 351.7 | 1255 | 349.15 | 1139.8 | 0.7303451238722645 | 182.40343347639484 | Price+Volume |

only showing top 20 rows

Figure: Consumer streaming execution demonstrating successful anomaly detection

| Publish time | Attribute keys | Message body ↑ |
|-------------------------|----------------|--|
| Apr 6, 2025, 6:10:40 PM | — | {"stock_ticker": "gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE |
| Apr 6, 2025, 6:10:40 PM | — | {"stock_ticker": "gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE |
| Apr 6, 2025, 6:10:40 PM | — | {"stock_ticker": "gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE |
| Apr 6, 2025, 6:10:40 PM | — | {"stock_ticker": "gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE |
| Apr 6, 2025, 6:10:40 PM | — | {"stock_ticker": "gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE |
| Apr 6, 2025, 6:10:40 PM | — | {"stock_ticker": "gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE |
| Apr 6, 2025, 6:10:40 PM | — | {"stock_ticker": "gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE |
| Apr 6, 2025, 6:10:40 PM | — | {"stock_ticker": "gs://oppe_2/AARTIIND_EQ_NSE_NSE_MINUTE |

| body.stock_ticker | body.timestamp | body.volume |
|---------------------------------------|---------------------|-------------|
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MI... | 2019-03-18 04:42:00 | 235 |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MI... | 2019-03-18 04:47:00 | 393 |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MI... | 2019-03-18 04:51:00 | 819 |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MI... | 2019-03-18 04:53:00 | 205 |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MI... | 2019-03-18 05:04:00 | 379 |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MI... | 2019-03-18 05:06:00 | 1043 |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MI... | 2019-03-18 05:08:00 | 812 |
| gs://oppe_2/AARTIIND_EQ_NSE_NSE_MI... | 2019-03-18 05:09:00 | 371 |

| body.volume_change_pct | body.warning |
|------------------------|---|
| 9.864422627395976 | Traded Volume more than 2% of its average |
| 87.58949880668257 | Traded Volume more than 2% of its average |
| 364.5490640952921 | Traded Volume more than 2% of its average |
| 8.868826340945294 | Traded Volume more than 2% of its average |
| 800.2375296912113 | Traded Volume more than 2% of its average |
| 1141.6666666666665 | Traded Volume more than 2% of its average |
| 358.23927765237016 | Traded Volume more than 2% of its average |
| 45.83333333333333 | Traded Volume more than 2% of its average |

Figure: PubSub message dashboard showing captured volume anomalies with parsed payload