Departamento de Elearning

Facultad Politécnica

Universidad Nacional de Asunción

Trabajo Grupal

Tarea Unida V:
Data Streaming

Identificación

- **Módulo:** Procesamiento de Flujo de Datos Masivos
- **Curso:** Big Data y Business Analytics
- **Integrantes:** Silvia Gabaglio y Stephanie González

1.docker-compose.yml - producer

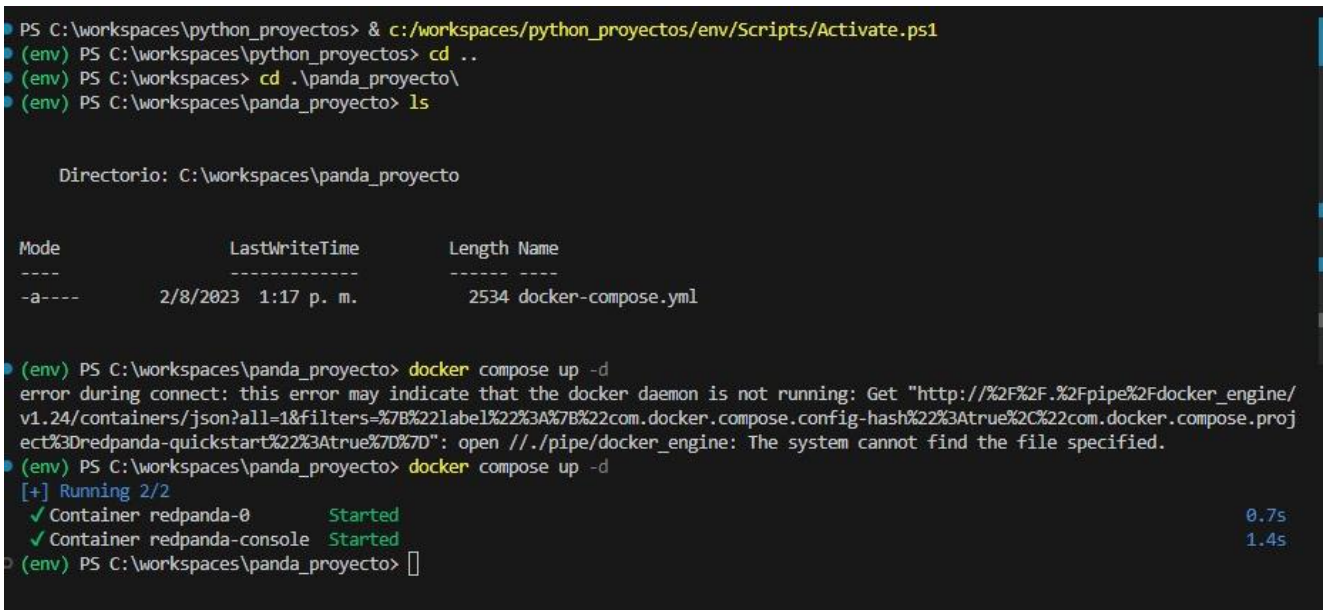
```yaml
---
version: '3.9'
services:

  redpanda:
    command:
      - redpanda
      - start
      - --smp
      - '1'
      - --reserve-memory
      - 0M
      - --overprovisioned
      - --set
      - redpanda.cluster_id=turning-red
      - --set
      - redpanda.enable_idempotence=true
      - --set
      - redpanda.enable_transactions=true
      - --set
      - redpanda.auto_create_topics_enabled=true
      - --node-id
      - '0'
      - --kafka-addr
      - PLAINTEXT://0.0.0.0:29092,OUTSIDE://0.0.0.0:9092
```

```
      - --advertise-kafka-addr
      - PLAINTEXT://redpanda:29092,OUTSIDE://localhost:9092
    image: docker.vectorized.io/vectorized/redpanda:v21.11.11
    container_name: redpanda
    ports:
      - 9092:9092
      - 29092:29092
    networks:
      - redpanda_network

  ksqldb-server:
    image: confluentinc/cp-ksqldb-server:latest
    hostname: ksqldb-server
    container_name: ksqldb-server
    depends_on:
      - redpanda
    ports:
      - "8088:8088"
    environment:
      KSQL_LISTENERS: "http://0.0.0.0:8088"
      KSQL_BOOTSTRAP_SERVERS: "redpanda:29092"
      KSQL_KSQL_SCHEMA_REGISTRY_URL: "http://schema-registry:8081"
      KSQL_KSQL_LOGGING_PROCESSING_STREAM_AUTO_CREATE: "true"
      KSQL_KSQL_LOGGING_PROCESSING_TOPIC_AUTO_CREATE: "true"
    networks:
      - redpanda_network

  ksqldb-cli:
    image: confluentinc/cp-ksqldb-cli:latest
    container_name: ksqldb-cli
    depends_on:
      - redpanda
      - ksqldb-server
    entrypoint: /bin/sh
    tty: true
    networks:
      - redpanda_network

  console:
    container_name: redpanda-console
    image: docker.redpanda.com/vectorized/console:v2.2.4
    entrypoint: /bin/sh
    command: -c 'echo "$$CONSOLE_CONFIG_FILE" > /tmp/config.yml;
/app/console'
    environment:
      CONFIG_FILEPATH: /tmp/config.yml
```

```
      CONSOLE_CONFIG_FILE: |
        kafka:
          brokers: ["redpanda:29092"]
          schemaRegistry:
            enabled: true
            urls: ["http://redpanda:8081"]
        redpanda:
          adminApi:
            enabled: true
            urls: ["http://redpanda:9644"]
    networks:
      - redpanda_network
    ports:
      - 8080:8080
    depends_on:
      - redpanda

networks:
  redpanda_network:
    driver: bridge
```

2.Código para levantar redpanda

Se levanta el redpanda en un puerto.



3. Instalación del KSQLdb-cli y KSQL-server

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

For more help on how to use Docker, head to https://docs.docker.com/go/guides/

PS C:\workspaces\panda_proyecto> docker compose up -d
[+] Running 47/4
 ✓ console 7 layers [⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled                                  194.7s
 ✓ ksqldb-server 12 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled                  215.4s
 ✓ redpanda 7 layers [⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled                               181.5s
 ✓ ksqldb-cli 17 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled          134.6s
[+] Running 5/5
 ✓ Network panda_proyecto_redpanda_network  Created                                    0.3s
 ✓ Container redpanda                       Started                                   4.9s
 ✓ Container ksqldb-server                  Started                                   5.8s
 ✓ Container redpanda-console               Started                                   5.9s
 ✓ Container ksqldb-cli                     Started                                   7.3s
PS C:\workspaces\panda_proyecto>
```

4.consumer

```python
import json
from kafka import KafkaConsumer

consumer = KafkaConsumer(
    bootstrap_servers=["localhost:9092"],
    group_id="demo-group",
    auto_offset_reset="earliest",
    enable_auto_commit=False,
    consumer_timeout_ms=1000,
    value_deserializer=lambda m: json.loads(m.decode('ascii'))
)

consumer.subscribe("CONTAR_SYMBOLOS")

try:
    for message in consumer:
        topic_info = f"topic: {message.partition}|{message.offset})"
        message_info = f"key: {message.key}, {message.value}"
        print(f"{topic_info}, {message_info}")
        print('hola')
except Exception as e:
    print(f"Error occurred while consuming messages: {e}")
finally:
    consumer.close()
```

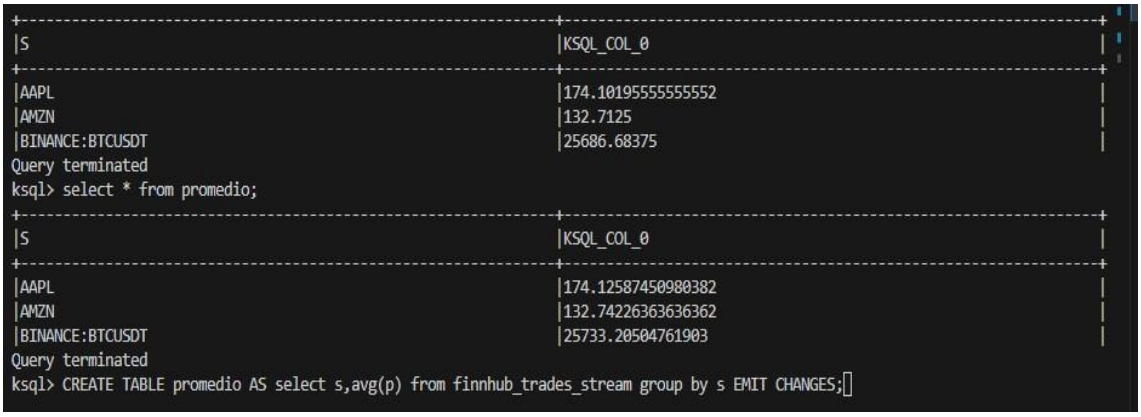
5.Las 4 consultas utilizadas en sql

Promedio de precio ponderado

CREATE TABLE promedio AS select s,avg(p) from finnhub_trades_stream group by s EMIT CHANGES;

Consulta promedio de precio ponderado

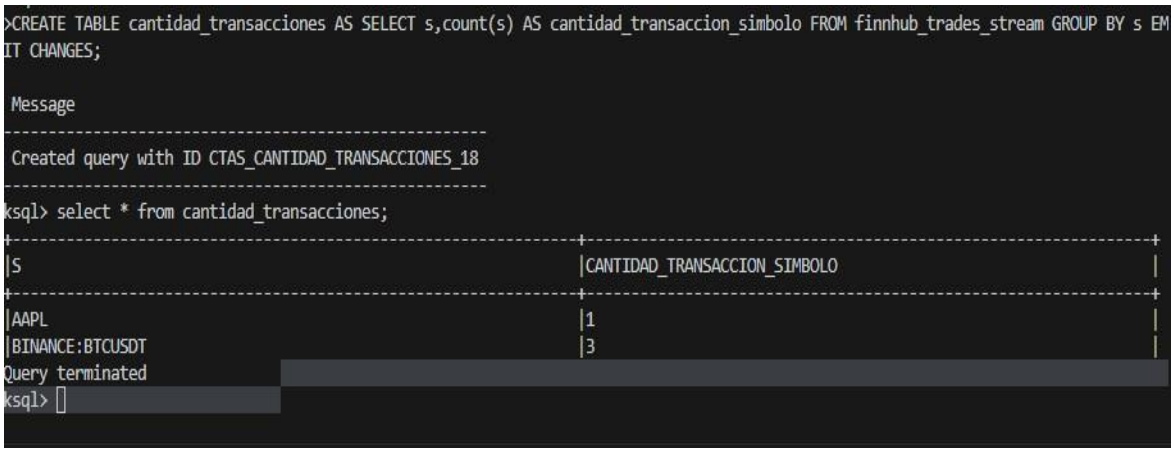
Select * from promedio



Cantidad de transacciones por símbolo

CREATE TABLE cantidad_transacciones AS SELECT s,count(s) AS cantidad_transaccion_simbolo FROM finnhub_trades_stream GROUP BY s EMIT CHANGES;

select * from cantidad_transacciones;



Precio Máximo

CREATE TABLE precio_maximo AS SELECT s,MAX(p) AS MAX_PRICE FROM finnhub_trades_stream GROUP BY s EMIT CHANGES;

select * from precio_maximo;

```
ksql> CREATE TABLE precio_maximo AS SELECT s,MAX(p) AS MAX_PRICE FROM finnhub_trades_stream GROUP BY s EMIT CHANGES;

 Message
----------------------------------------
 Created query with ID CTAS_PRECIO_MAXIMO_14
----------------------------------------
ksql> select * from precio_maximo;
+-------------------------------------------------------------------+-------------------------------------------------------------------+
|S                                                                  |MAX_PRICE                                                          |
+-------------------------------------------------------------------+-------------------------------------------------------------------+
|AAPL                                                               |174.44                                                             |
|AMZN                                                               |133.075                                                            |
Query terminated
ksql>
```

Precio Mínimo

CREATE TABLE precio_minimo AS SELECT s,MIN(p) AS MIN_PRICE FROM finnhub_trades_stream GROUP BY s EMIT CHANGES;

select * from precio_minimo;

```
ksql> select * from precio_minimo;
+-------------------------------------------------------------------+-------------------------------------------------------------------+
|S                                                                  |MIN_PRICE                                                          |
+-------------------------------------------------------------------+-------------------------------------------------------------------+
|AAPL                                                               |174.34                                                             |
|AMZN                                                               |132.9                                                              |
|BINANCE:BTCUSDT                                                    |25982.8                                                            |
Query terminated
ksql>
```