# TAREA DATA STREAMING

# GRUPO 19:

- Carlos Emanuel Aquino Peralta
- Hugo Alberto Díaz Parra
- Hugo Ismael Gonzalez
- Rodolfo Paredes Monges
- Elio Emanuel Romero López

#### **PRE-REQUISITOS**

- 1. **Python 3.x** con pip actualizado
- 2. **Docker** con docker compose actualizado

#### **INSTRUCCIONES**

- 1. **Crear los contenedores**, para ello utilizar el docker compose utilizando el siguiente comando
  - \$ docker compose -d up
- 2. Comprobar los contenedores, para ello ejecutar el comando
  - \$ docker stats
  - se deben visualizar en ejecucion los contenedores
  - redpanda
  - redpanda-console
  - ksqldb-server
  - ksqldb-cli
- 3. Crear el ambiente de trabajo para Python3, para ello ejecutar:
  - \$ python3 -m venv env
  - \$ source env/bin/activate
  - \$ pip install --upgrade pip
  - \$ pip install kafka-python
- 4. Crear el topic en RedPanda
  - \$ docker exec -it redpanda rpk topic create stock-updates
- 5. **Actualizar el API Key de Finnhub**, para registrarse en https://finnhub.io/ y obtener el API Key, luego editar y actualizar el archivo producer.py en la linea 55 **Finnhub Stock API KEY:** cjfcc21r01qhblojr5s0cjfcc21r01qhblojr5sg
- 6. **Cargar los mensajes a RedPanda**, ejecutando el producer.py \$ python3 stock-monitor/producer.py
- 7. **Verificar la carga de los datos**, para ello ingresar a la consola de RedPanda desde el navegador a la siguiente URL <a href="http://127.0.0.1:8080/overview">http://127.0.0.1:8080/overview</a>, se debe visualizar el topic **stock-updates** y los mensajes.
- 8. Ingresar a KSQL, con el siguiente comando
  - \$ sudo docker exec -it ksqldb-cli ksql http://ksqldb-server:8088
- 9. Crear el stream stock\_updates en KLSQL
  - > CREATE STREAM stock\_updates ( symbol VARCHAR,

price DOUBLE,

volume DOUBLE.

timestamp VARCHAR)

WITH (

kafka topic='stock-updates',

value format='json');

- 10. Consultar los datos del stream,
  - > SELECT \* FROM STOCK UPDATES;
- 11. Crear la tabla de stock\_resume en KLSQL,
  - > CREATE TABLE stock\_resume AS SELECT symbol AS simbolo,

MAX(price) AS maximo,

MIN(price) AS minimo,

SUM(price\*volume)/SUM(volume) AS promedio,

COUNT(\*) AS cantidad

FROM stock updates

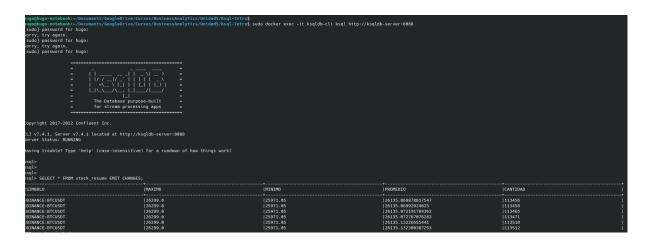
GROUP BY symbol;

- 12. Consultar la tabla de resumen,
  - > SELECT \* FROM STOCK RESUME EMIT CHANGES;

la información resumida va ir actualizando según el productor carga los mensajes a RedPanda.

- 13. Consultar el promedio y la cantidad de transacciones por simbolo
  - > SELECT SIMBOLO, PROMEDIO, CANTIDAD FROM STOCK\_RESUME EMIT CHANGES:
- 14. Consultar el minimo y la cantidad de transacciones por simbolo
  - > SELECT SIMBOLO, MINIMO, CANTIDAD FROM STOCK\_RESUME EMIT CHANGES;
- 15. Consultar el promedio y la cantidad de transacciones por simbolo
  - > SELECT SIMBOLO, MAXIMO, CANTIDAD FROM STOCK\_RESUME EMIT CHANGES;

#### **EVIDENCIAS**



ksql> ksql> SELECT SIMBOLO, PROMEDIO, CANTIDAD FROM STOCK_I	RESUME EMIT CHANGES;	
SIMBOLO	PROMEDIO	CANTIDAD
+		
AMZN	135.08260226066042	17598
AAPL	180.9443361770918	21308
BINANCE:BTCUSDT	26191.716375835982	147158
AAPL	180.94422037199115	21334
AMZN	135.0825797887024	17613
BINANCE:BTCUSDT	26191.758464630595	147214
AMZN	135.0825496205345	17635
AAPL	180.94412031951393	21356

CANTID	
1838   22111   148659   148692   18343   22131   18369	
	22111   148659   148692   18343   22131

ksql> ksql> Select simbolo, maximo, cantidad from stock_resu	ME EMIT CHANGES;	+
SIMBOLO	MAXIMO	CANTIDAD
BINANCE:BTCUSDT	26439.67	149582
AMZN	135.3	18653
AAPL	181.31	22479
BINANCE:BTCUSDT	26439.67	149620
AAPL	181.31	22497
AMZN	135.3	18664
AAPL	181.31	22504
LAMZN	I 135.3	118668

## **REPOSITORIO GIT**

https://github.com/ninjadiaz/redpanda/tree/main/Ksql-Intro

### REFERENCIAS

https://finnhub.io/

https://docs.redpanda.com/docs/get-started/guick-start/

https://redpanda.com/blog/python-redpanda-kafka-api-tutorial

https://redpanda.com/blog/ksqldb-materialized-cache