

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики, математики и электроники
Факультет информатики
Кафедра технической кибернетики

**Отчет по лабораторной работе № 1
по курсу «Инженерия данных»**

Тема: «Базовый пайплайн работы с данными»

Выполнил Самигуллин Равиль
Группа 6133 – 010402D
Преподаватель Парингер Р.А

Самара

Содержание

Содержание.....	2
Техническое задание.....	3
Подготовка окружения.	4
Выполнение pipeline NIFI-elasticsearch.....	6
Выполнение pipeline airflow-elasticsearch.	20
ЗАКЛЮЧЕНИЕ	23

Техническое задание

В рамках данной лабораторной работы предлагается построить простейший пайплайн, собирающий воедино данные из нескольких файлов, обрабатывающий их и сохраняющий результат в no-sql базу данных.

Для построения пайплайна использовались:

- Apache Airflow
- Apache Nifi
- ElasticSearch

В качестве данных использовал набор из 2х csv файлов полученных из набора данных wine-review

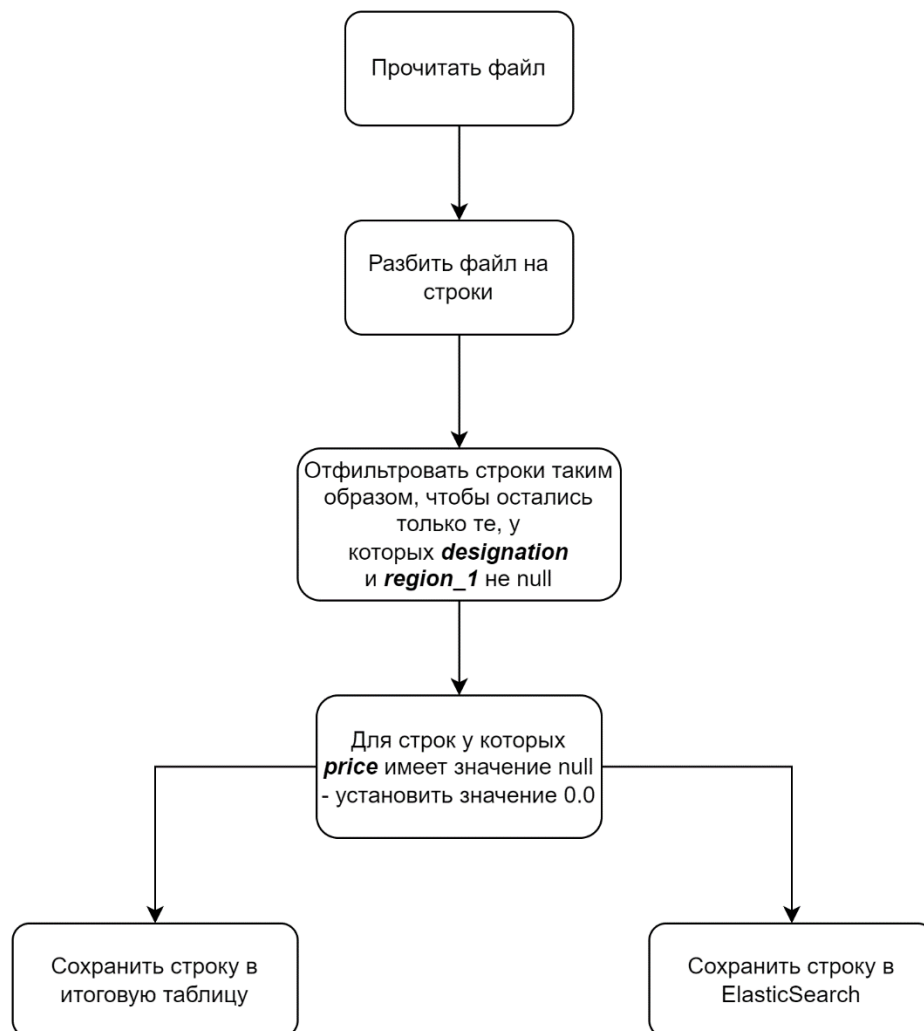


Рис. 1 – Схема пайплайна который необходимо реализовать.

Данный пайплайн должен быть построен дважды: один раз с использованием Apache Nifi и второй раз с использованием Apache Airflow.

Подготовка окружения.

Я использую инструкции и подготовленными docker файлами из (<https://github.com/ssau-data-engineering/Prerequisites/tree/main>), получил сеть data-engineering-labs-network с 5 образами: nifi, airflow, postgresql, elasticsearch, mlflow.

Я использую windows, у меня уже была подготовлена wsl.

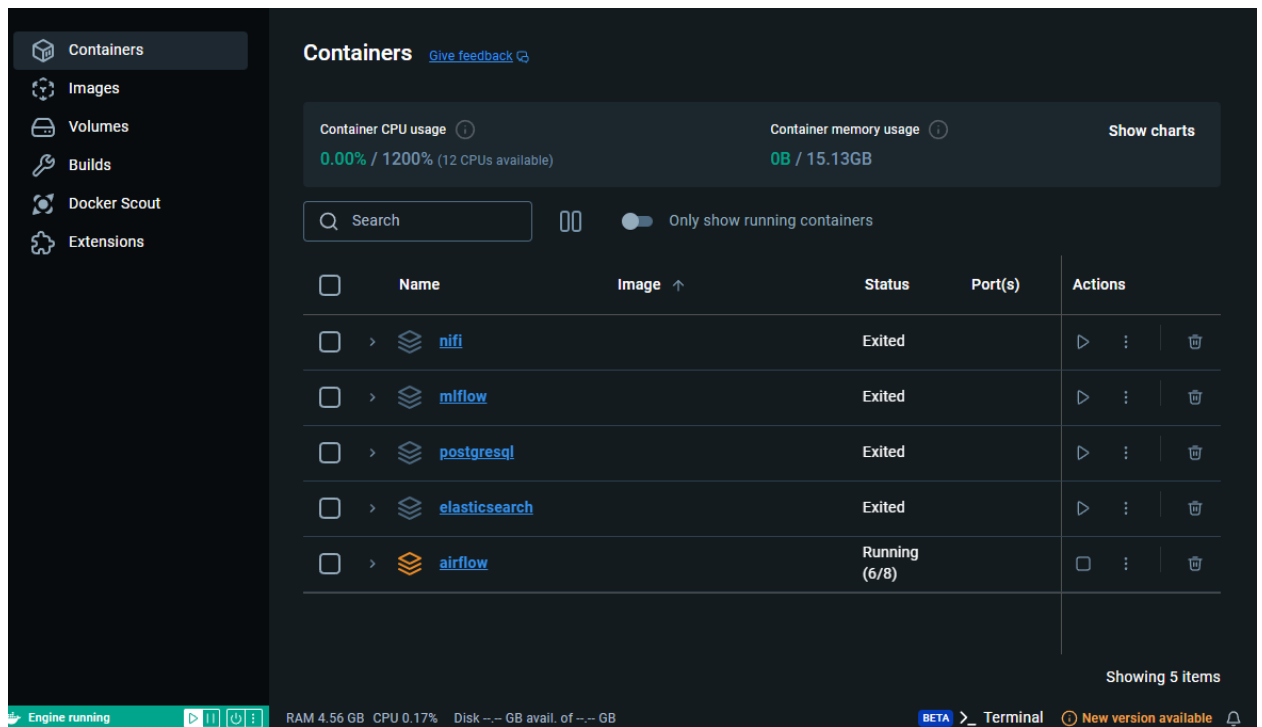


Рис. 2 – Докер образы в docker desktop

```

PS C:\Users\darti> docker network inspect data-engineering-labs-network
[
  {
    "Name": "data-engineering-labs-network",
    "Id": "bdf62ebd8b14fa16bbcb563d151deaeed5314f081cad841639bea87fca",
    "Created": "2024-09-04T10:45:09.148120229Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "0749c3f60318e5b4420a33f4a93504c7e9a7d9267f8fccef49c4478509d9": {
        "Name": "mlflow-db-1",
        "EndpointID": "8d08da827e6df158887cbc253b0b30d8871a3c3c4d",
        "MacAddress": "02:42:ac:12:00:0c",
        "IPv4Address": "172.18.0.12/16",
        "IPv6Address": ""
      },
      "0b264da25bb973a1c177b752604c5ed2003964ddd6209fa0c1a127cc8386": {
        "Name": "airflow-airflow-worker-1",
        "EndpointID": "73f91b0989d021554970fb0fb92c453f9cbac1984d",
        "MacAddress": "02:42:ac:12:00:03",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""
      },
      "1abce7b3861af2bbf5d454ab3718c6d8dbf696035bb98f81ce24d4f03bec": {
        "Name": "airflow-airflow-scheduler-1",
        "EndpointID": "bd7d1ead308789e114a5083dc672edc57b000767ac",
        "MacAddress": "02:42:ac:12:00:05",
        "IPv4Address": "172.18.0.5/16",
        "IPv6Address": ""
      },
      "2277469c9e42ea0ef18f9fa8178838813afd3718ccddd80d06d86b044729": {
        "Name": "nifi-apache-nifi-1",
        "EndpointID": "51d0ab8024736f794e573600a9e9155fcd04ec5959",
        "MacAddress": "02:42:ac:12:00:0d",
        "IPv4Address": "172.18.0.13/16",
        "IPv6Address": ""
      },
      "3d707a47373f1a81b639482e9e6528f6c06416715f39ac1dd995e0fcd4c1": {
        "Name": "elasticsearch-elasticsearch-kibana-1",
        "EndpointID": "b46cc5ca11f03bf110265d8a617e1b662a9a934240",
        "MacAddress": "02:42:ac:12:00:08",
        "IPv4Address": "172.18.0.8/16",
        "IPv6Address": ""
      },
      "41eca7e77bf8e2a21db296c86639933967ab674e219a07c9dfb80c67cd74": {
        "Name": "airflow-airflow-triggerer-1",
        "EndpointID": "774f36c6c44a3bbe740f4b70adebfa7845af085e00",
        "MacAddress": "02:42:ac:12:00:04",
        "IPv4Address": "172.18.0.4/16"
      }
    }
  }
]

```

Рис. 3 – Проверка докер контейнеры запущенные в сети data-engineering-labs-network

Выполнение pipeline NIFI-elasticsearch.

Схема конечного пайплайна:

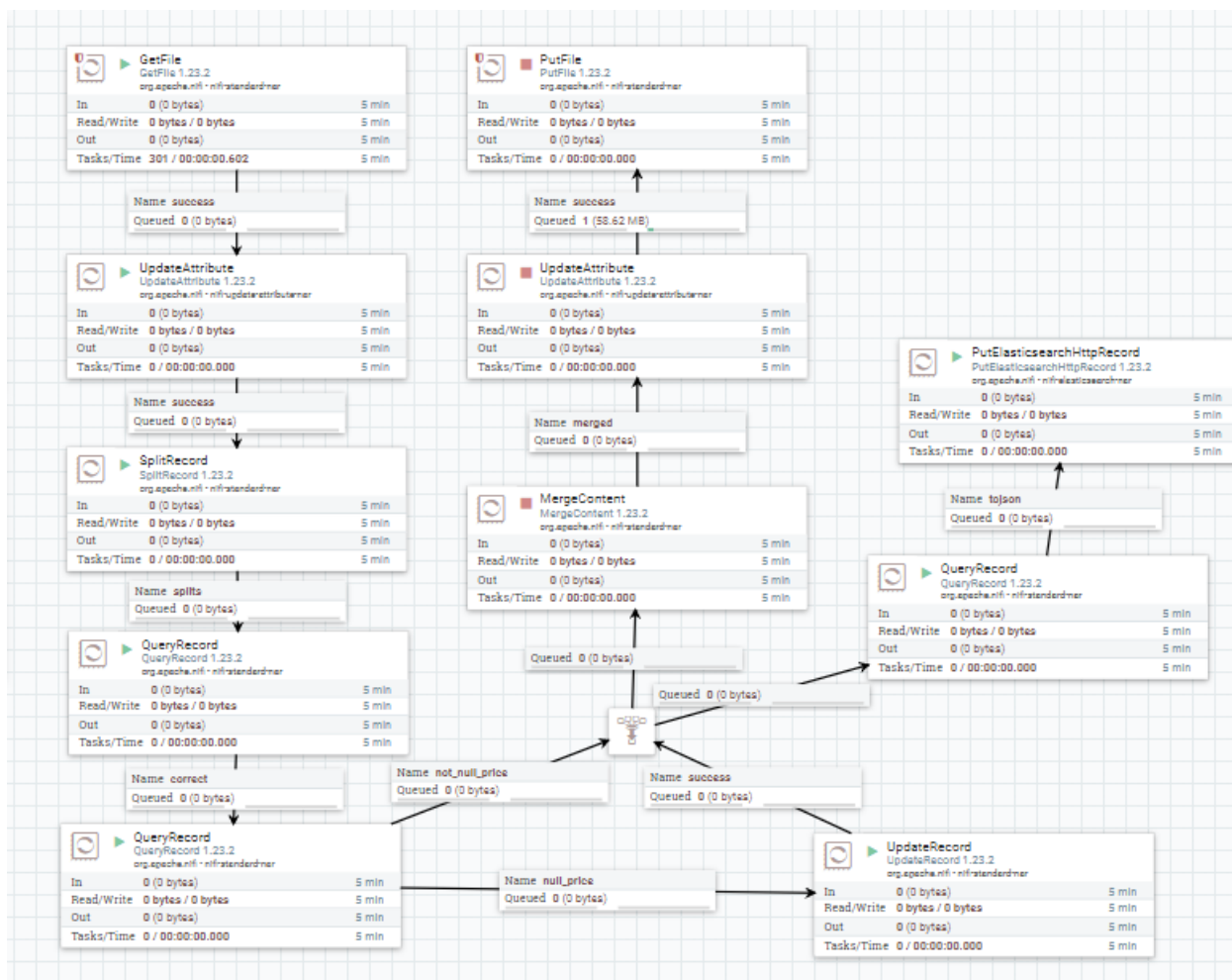


Рис. 4 – Моя конечная схема для Nifi

На самом деле 3 queryRecord не нужен, я использовал его для того что бы трансформировать отправленные файлы в .json, но это не пригодилось.

Для начала создал процессор getFile и выбрал файлы с расширением csv на загрузку.

Configure Processor | GetFile 1.23.2

Invalid

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

✔

+

Property		Value	
Input Directory	?	C:\Users\darti\Desktop\IR1_Pipeline\Prerequisites\ni...	
File Filter	?	[^\.].*\csv	
Path Filter	?	No value set	
Batch Size	?	10	
Keep Source File	?	false	
Recurse Subdirectories	?	true	
Polling Interval	?	0 sec	
Ignore Hidden Files	?	true	
Minimum File Age	?	0 sec	
Maximum File Age	?	No value set	
Minimum File Size	?	0 B	
Maximum File Size	?	No value set	

CANCEL

APPLY

Рис. 5 – Настройки GetFile

Далее настроил подключение на прочтение CSVRecordSetWriter, CSVReader.

NiFi Flow Configuration

✕

GENERAL

CONTROLLER SERVICES

+









	Name	Type	Bundle	State ▾	Scope	
	CSVRecordSetWriter	CSVRecordSetWriter 1.2...	org.apache.nifi - nifi-rec...	⚙ Disabled	NiFi Flow	  
	CSVReader	CSVReader 1.23.2	org.apache.nifi - nifi-rec...	⚙ Disabled	NiFi Flow	  

Рис. 6 –Enable прожал для CSVRecordSetWriter, CSVReader.

Controller Service Details | CSVReader 1.23.2

SETTINGS	PROPERTIES	COMMENTS
Required field		
Property	Value	
Schema Access Strategy	?	Use 'Schema Name' Property
Schema Registry	?	AvroSchemaRegistry →
Schema Name	?	\${schema.name}
Schema Version	?	No value set
Schema Branch	?	No value set
CSV Parser	?	Apache Commons CSV
Date Format	?	No value set
Time Format	?	No value set
Timestamp Format	?	No value set
CSV Format	?	Custom Format
Value Separator	?	,
Record Separator	?	\n
Treat First Line as Header	?	true
Ignore CSV Header Column Names	?	true

Рис. 7 –Настройки CSVreader

Controller Service Details | CSVRecordSetWriter 1.23.2

SETTINGS	PROPERTIES	COMMENTS
Required field		
Property	Value	
Schema Write Strategy	?	Set 'schema.name' Attribute
Schema Cache	?	No value set
Schema Access Strategy	?	Use 'Schema Name' Property
Schema Registry	?	AvroSchemaRegistry →
Schema Name	?	\${schema.name}
Schema Version	?	No value set
Schema Branch	?	No value set
Date Format	?	No value set
Time Format	?	No value set
Timestamp Format	?	No value set
CSV Format	?	Custom Format
Value Separator	?	,
Include Header Line	?	true
Quote Character	?	"

Рис. 8 –Настройки CSVwriter

На рисунках 7,8 конечные настройки, далее по отчету я опишу, как они пришли к этому состоянию.

Далее настроил split, поставил разделение на 1, как я думал по тз, что нужно разделить файл на строчки и каждую строчку последовательно пропускать через очередь, в итоге скорее закономерно получил ошибку с переполнением и нехваткой памяти, и решил, что лучше будет все-таки разделить по 1000 строк в файлах.

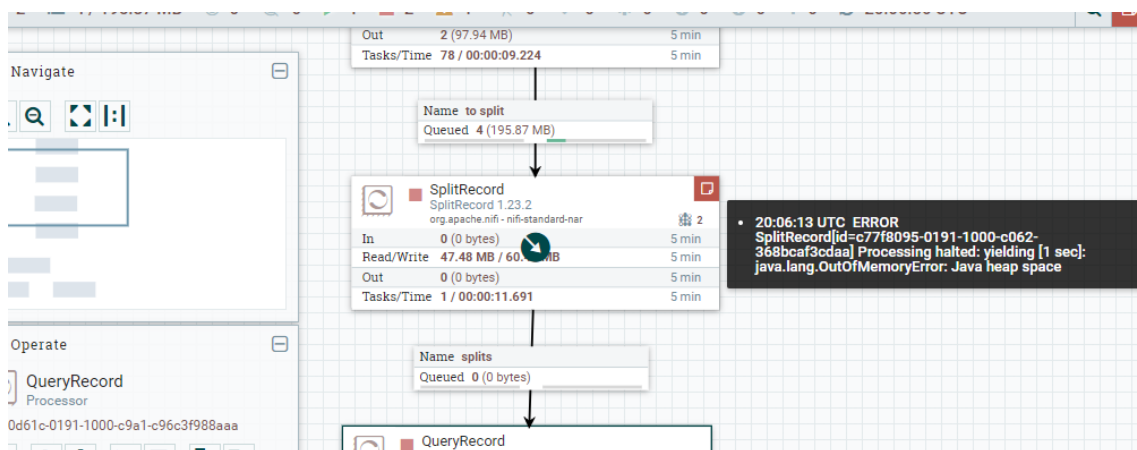


Рис. 9 – Проблема с переполнением

Далее возникла ошибка, которая заставила меня засесть за документацию, но я так и не смог найти конкретного решения. Как я понял, что мой SQL был тут не причем, потому как я использовал `SELECT * FROM FLOWFILE`, как проверку.

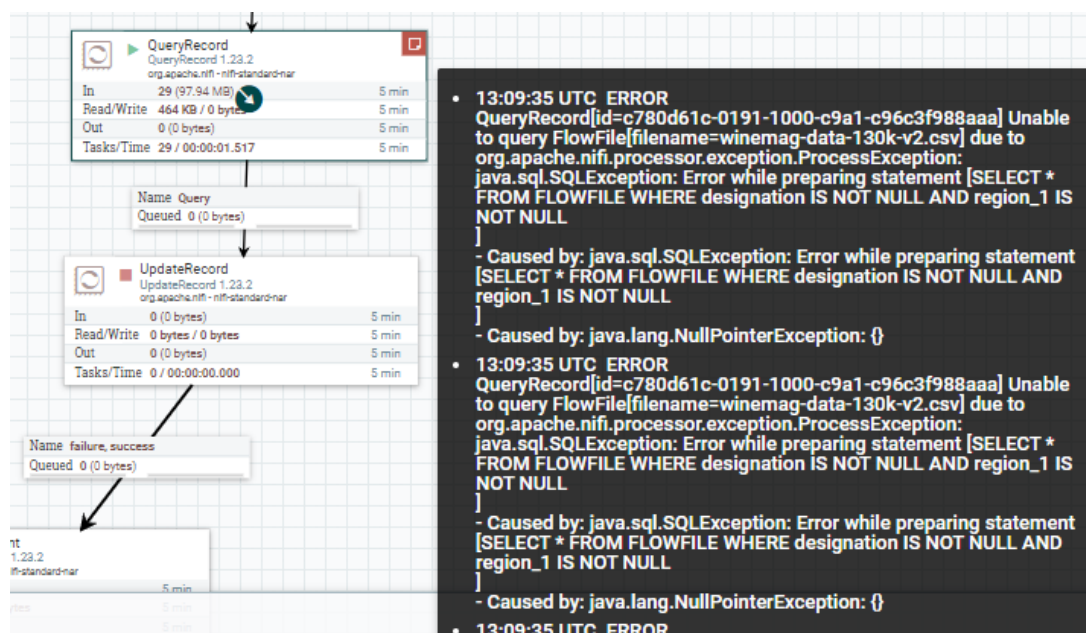


Рис. 11 – Проблема NULL

Проблема скорее заключалась в прочтении csv файлов. И тут я решил сменить, тактику и попробовать так как показано в гайдах на форумах, там для прочтения csv файла использовалась avro schema, которая, конечно, несколько ограничивает нашу гибкость, но пайплайн будет работать.

Controller Service Details | AvroSchemaRegistry 1.23.2

SETTINGS

PROPERTIES

COMMENTS

Required field

Property		Value
Validate Field Names	?	true
wine	?	{...}

```

{
  "type": "record",
  "name": "WineRecord",
  "fields": [
    {"name": "id", "type": ["long"]},
    {"name": "country", "type": ["null", "string"]},
    {"name": "description", "type": ["null", "string"]},
    {"name": "designation", "type": ["null", "string"]},
    {"name": "points", "type": ["null", "int"]},
    {"name": "price", "type": ["null", "float"]},
    {"name": "province", "type": ["null", "string"]},
    {"name": "region_1", "type": ["null", "string"]},
    {"name": "region_2", "type": ["null", "string"]},
    {"name": "taster_name", "type": ["null", "string"]},
    {"name": "taster_twitter_handle", "type": ["null", "string"]},
    {"name": "title", "type": ["null", "string"]},
    {"name": "variety", "type": ["null", "string"]},
    {"name": "winery", "type": ["null", "string"]}
  ]
}

```

OK

updated: 18

OK

ant Processors and services of this Proces

Рис. 12 –Настройки avro

Так же возник ряд ошибок с sql, которые я так и не смог адекватно понять.

И так после перестала работать queryrecord.

Я добавил функцию upgrade которая бы кодировала файл приписывая к нему свойство схемы. И csv начал работать.

Processor Details

UpdateAttribute 1.23.2

Running

STOP & CONFIGURE

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

Property		Value
Delete Attributes Expression	?	No value set
Store State	?	Do not store state
Stateful Variables Initial Value	?	No value set
Cache Value Lookup Cache Size	?	100
schema.name	?	wine

Рис. 12 – Функция update что бы добавить свойство schema.name для пакета данных, которая бы указывала avro как разбивать наши данные.

После появилась неожиданная ошибка pipeline пускал пустые строки.

Так же методом проб и просмотром data provenance, я понял, что ошибка в split, а именно в csvread.

View as: original

1	country,description,designation,points,price,province,region_1,region_2,variety,winery
2	,,,,,,,,,
3	,,,,,,,,,
4	,,,,,,,,,
5	,,,,,,,,,
6	,,,,,,,,,
7	,,,,,,,,,
8	,,,,,,,,,
9	,,,,,,,,,
10	,,,,,,,,,
11	,,,,,,,,,
12	,,,,,,,,,
13	,,,,,,,,,
14	,,,,,,,,,
15	,,,,,,,,,
16	,,,,,,,,,
17	,,,,,,,,,
18	,,,,,,,,,
19	,,,,,,,,,
20	,,,,,,,,,
21	,,,,,,,,,
22	,,,,,,,,,
23	,,,,,,,,,
24	,,,,,,,,,
25	,,,,,,,,,
26	,,,,,,,,,
27	,,,,,,,,,
28	,,,,,,,,,
29	,,,,,,,,,
30	,,,,,,,,,
31	,,,,,,,,,
32	,,,,,,,,,
33	,,,,,,,,,
34	,,,,,,,,,
35	,,,,,,,,,
36	,,,,,,,,,
37	,,,,,,,,,
38	,,,,,,,,,
39	,,,,,,,,,
40	,,,,,,,,,

Рис. 13 – Пропали данные после split

Ошибка быстро решилась разделитель был не тот, то есть была (;) ,а не (,).

(Тут уже можно заметить другую мою ошибку, связанную с кодировкой avro)

Configure Connector Service | CSVReader 1.23.2

SETTINGS

PROPERTIES

COMMENTS

Required field

Property	Value
Schema Branch	No value set
CSV Parser	Apache Commons CSV
Date Format	No value set
Time Format	No value set
Timestamp Format	No value set
CSV Format	Custom Format
Value Separator	;
Record Separator	\n
Treat First Line as Header	true
Ignore CSV Header Column Names	false
Quote Character	"
Escape Character	\
Comment Marker	No value set
Null String	No value set

CANCEL

APPLY

Рис. 14 – Ошибка в Csvread

Processor Details | QueryRecord 1.23.2

Running

STOP & CONFIGURE

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

Property	Value
Record Reader	CSVReader
Record Writer	CSVRecordSetWriter
Include Zero Record FlowFiles	true
Cache Schema	true
Default Decimal Precision	10
Default Decimal Scale	0
correct	SELECT * FROM FLOWFILE WHERE designation <>...

Рис. 15 – Фильтрация с помощью queryRecord

Для этапа фильтрации из условия (designation и region_1 не null) я написал фильтр на sql запросе: `SELECT * FROM FLOWFILE WHERE designation <>`
`" AND region_1 <>"`

Processor Details
QueryRecord 1.23.2

Running

STOP & CONFIGURE

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

Property		Value	
Record Reader	?	CSVReader	→
Record Writer	?	CSVRecordSetWriter	→
Include Zero Record FlowFiles	?	true	
Cache Schema	?	true	
Default Decimal Precision	?	10	
Default Decimal Scale	?	0	
not_null_price	?	SELECT * FROM FLOWFILE WHERE price is not null	
null_price	?	SELECT * FROM FLOWFILE WHERE price is null	

Рис. 16 – Фильтрация для price что бы в последствии изменить данные

Далее я использовал sql функцию для фильтрации на price.

SELECT * FROM FLOWFILE WHERE price is not null

SELECT * FROM FLOWFILE WHERE price is null

Processor Details
UpdateRecord 1.23.2

Running

STOP & CONFIGURE

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

Property		Value	
Record Reader	?	CSVReader	→
Record Writer	?	CSVRecordSetWriter	→
Replacement Value Strategy	?	Literal Value	
/price	?	0.0	

Рис. 17 – Присвоение price с null -> 0.0

Присвоил отфильтрованным значениям price значение 0.0, с помощью процессора upgrade.

Configure Processor
MergeContent 1.23.2

Stopped

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

Property	Value
Merge Strategy	Bin-Packing Algorithm
Merge Format	Binary Concatenation
Attribute Strategy	Keep Only Common Attributes
Correlation Attribute Name	id
Minimum Number of Entries	1
Maximum Number of Entries	300000
Minimum Group Size	0 B
Maximum Group Size	No value set
Max Bin Age	No value set
Maximum number of Bins	5
Delimiter Strategy	Text
Header	No value set

Рис. 18 –Настройки процессора Merge Content

Configure Processor
UpdateAttribute 1.23.2

Stopped

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

Property	Value
Delete Attributes Expression	No value set
Store State	Do not store state
Stateful Variables Initial Value	No value set
Cache Value Lookup Cache Size	100
filename	\$(fragment.identifier)_\${random()}.csv

Рис. 19 – Настройки присвоения Имени Update attribute.

Присвоил имя для конечного merge файла.

Configure Processor
PutFile 1.23.2

Stopped

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

?

+

Property	Value
Directory	/opt/nifi/nifi-current/data/lab_1/output
Conflict Resolution Strategy	fail
Create Missing Directories	true
Maximum File Count	No value set
Last Modified Time	No value set
Permissions	No value set
Owner	No value set
Group	No value set

Рис. 20 –Настройки putfile

Настроил процессор Putfile

Но данные прочитывались неправильно, а именно в строку с string, почему-то декодер хотел прочитать как float.

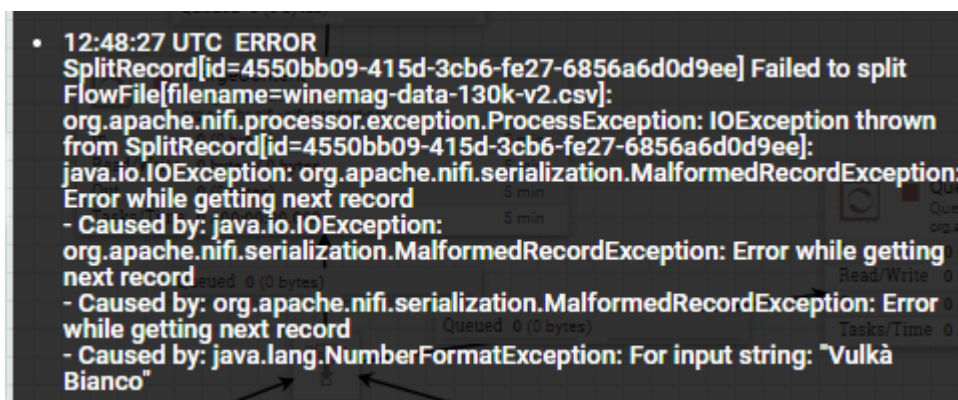


Рис. 21 – Ошибка связанная с настройкой avro

(,country,description,designation,points,price,province,region_1,region_)

Снова мелкая ошибка. Я понял, что id который пропущен первым столбцом прочитывается как str, из-за этого все столбцы смещаются и возникает проблема с чтением и записью данных.

Добавил его в авго. И все заработало.

После возникла другая мелкая ошибка. Связанная с передачей в elasticsearch.

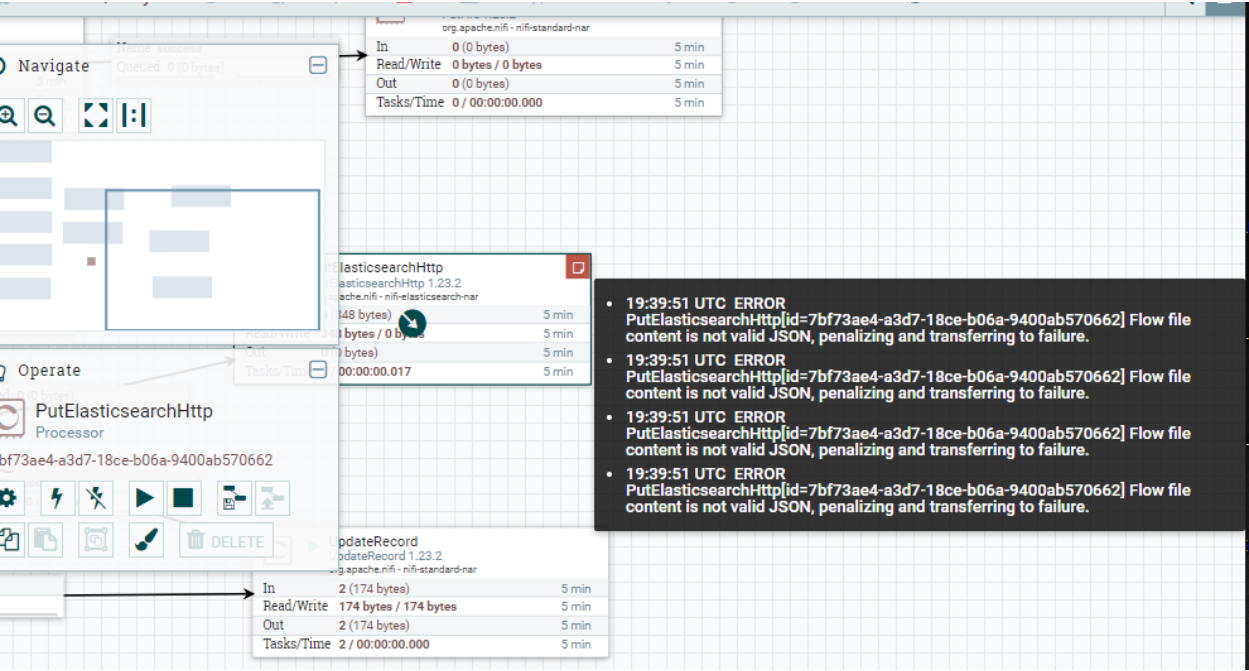


Рис. 22 – Ошибка с файлом данных.

Я решил кодировать пайплайн как json используя авго, данные, которые приходили elasticsearch хоть и видел, но не мог разметить, что было очень странно. И даже моё создание индекса с разметкой в самом elasticsearch не помогло, я решил сменить PutElasticSrachHTTP на PutElasticSrachHTTPRecord.

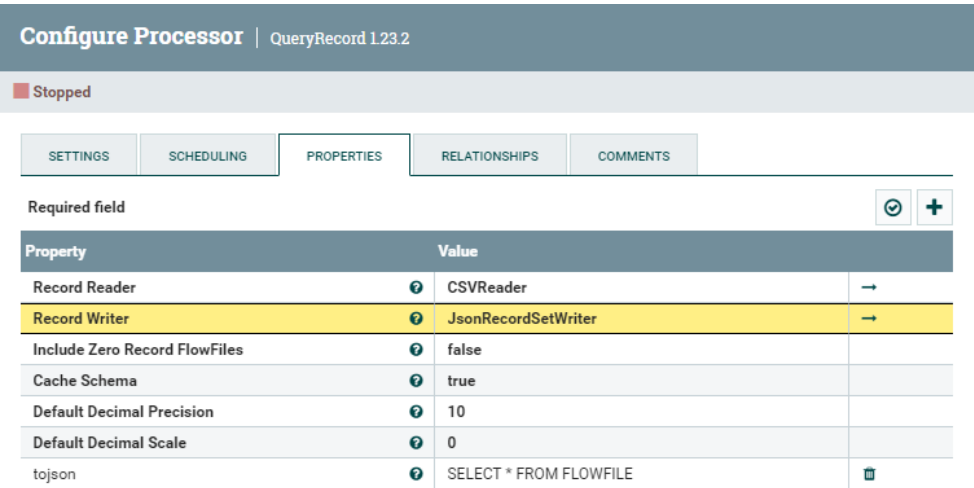


Рис. 23 – Настройки для преобразования .csv в .json

Возможно, я не совсем понял как именно, это работает, но теперь он принимал и csv файл причем не было проблем, с их прочтением.

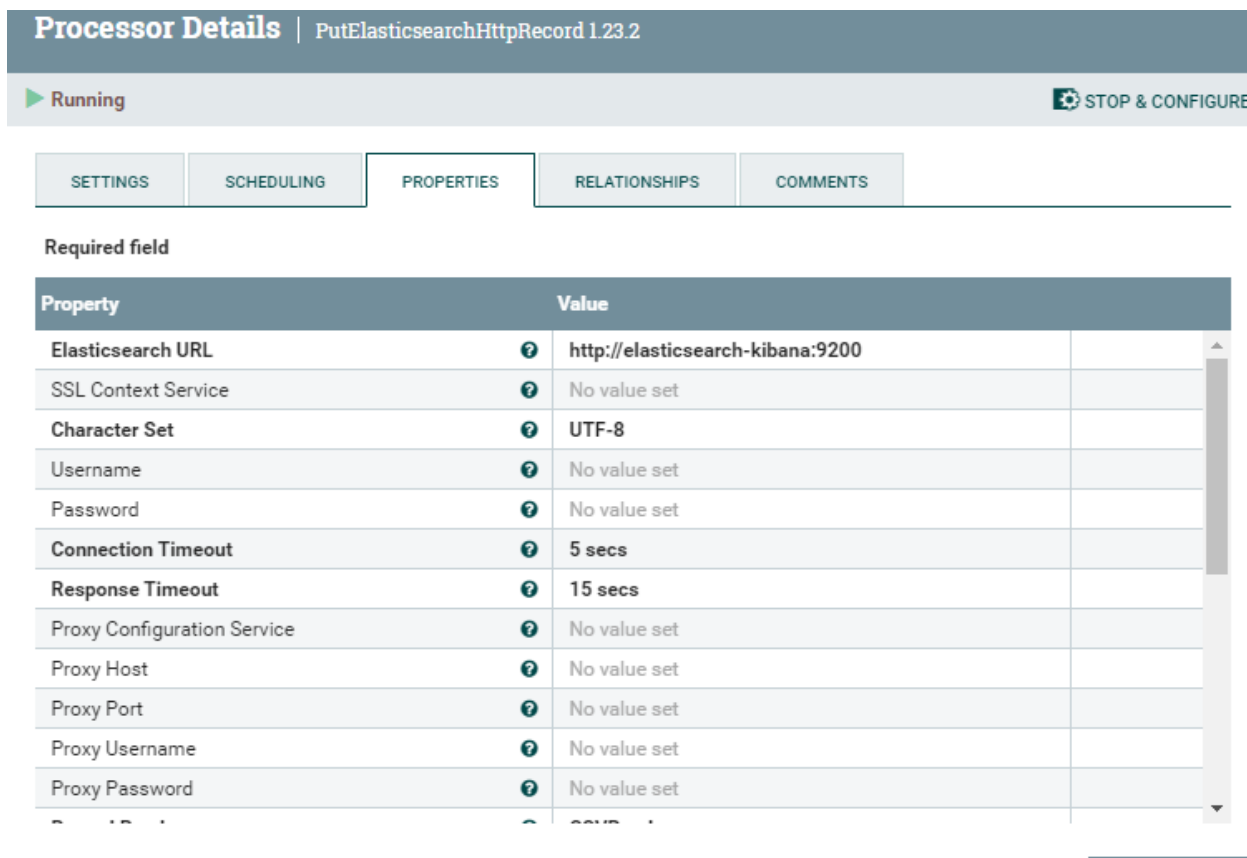


Рис. 24 –Настройки для PutElasticSearchHTTPRecord.

Далее переходим в elasticsearch в index management и просматриваем пришедшие индексы.

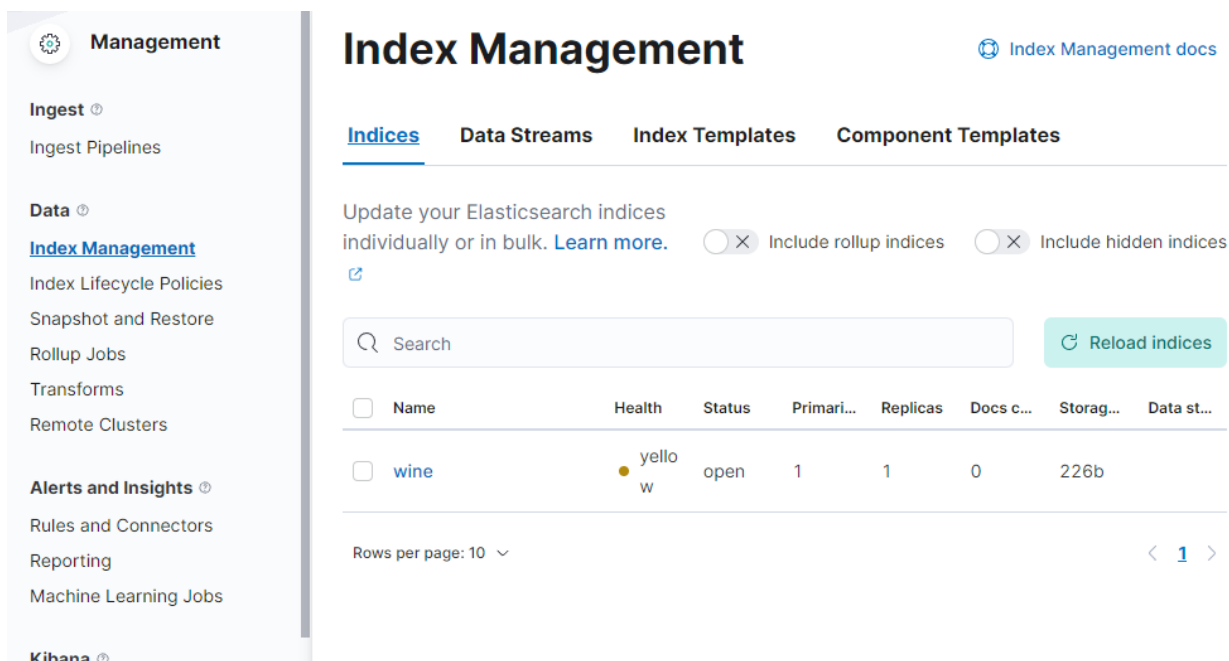


Рис. 25 – Панель индексов elasticsearch.

Далее переходим в панель index pattern и создаем index pattern для построения графика.

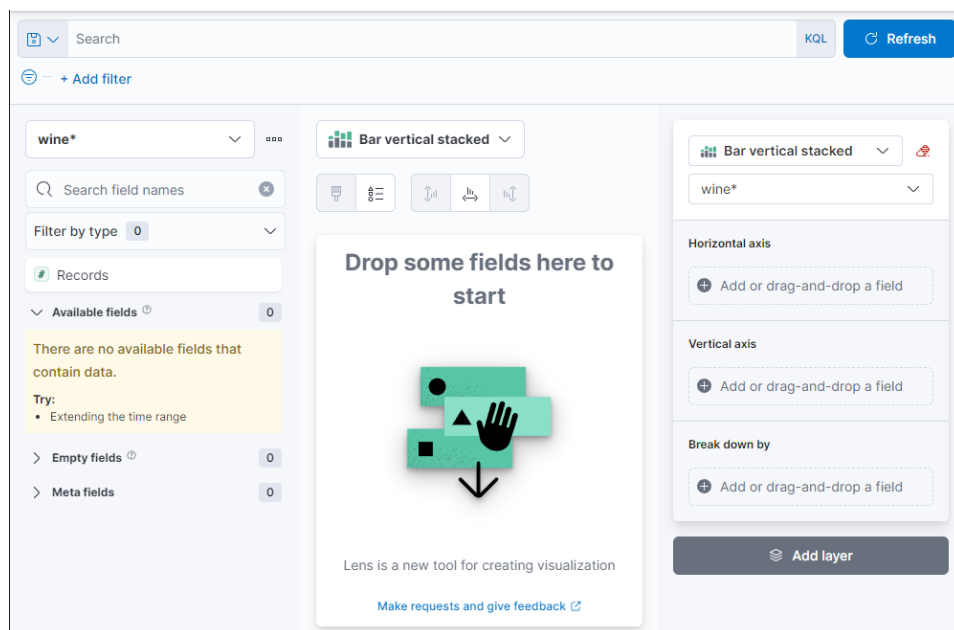


Рис. 26 –Панель создания index pattern

На рисунке 26 индекс, который пришел с json и не прочитался.

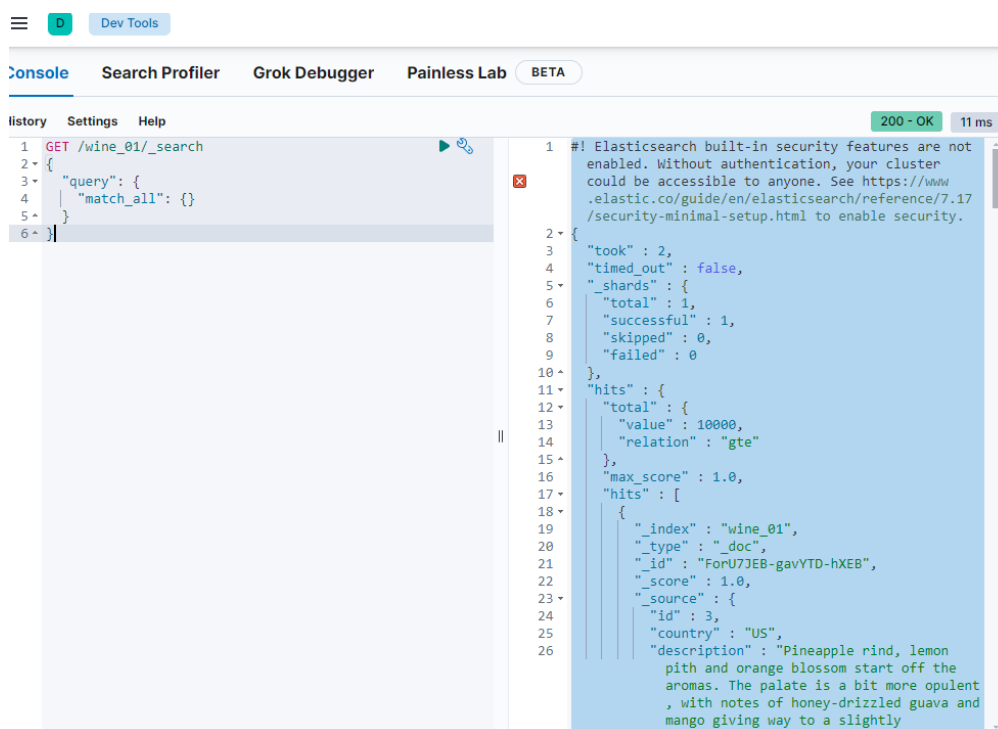


Рис. 27 – Индекс пришедший csv от HTTPRecord

Как видно этот индекс прочитался и выдал всю информацию

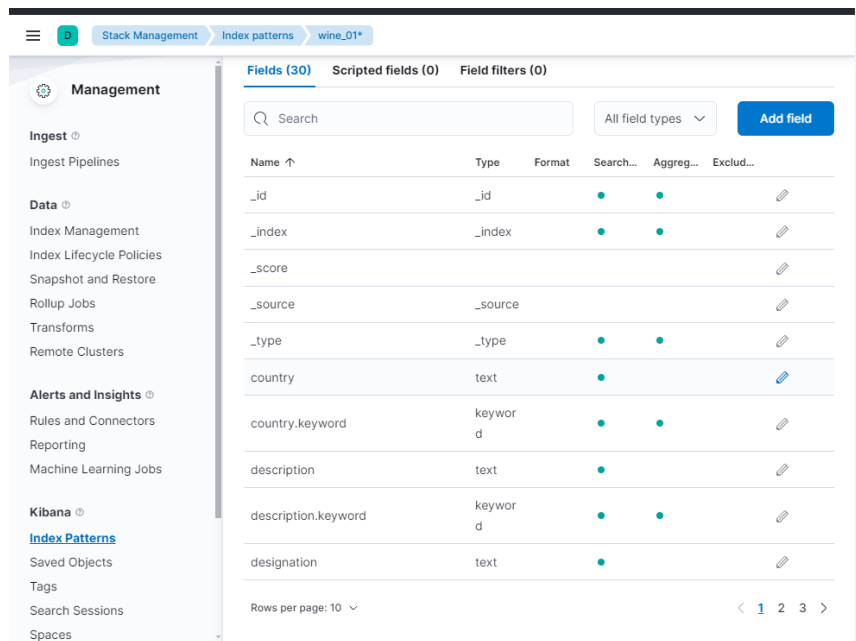


Рис. 28 – index pattern для удачного индекса

В заключении построим график в dash board используя как оси price и point.

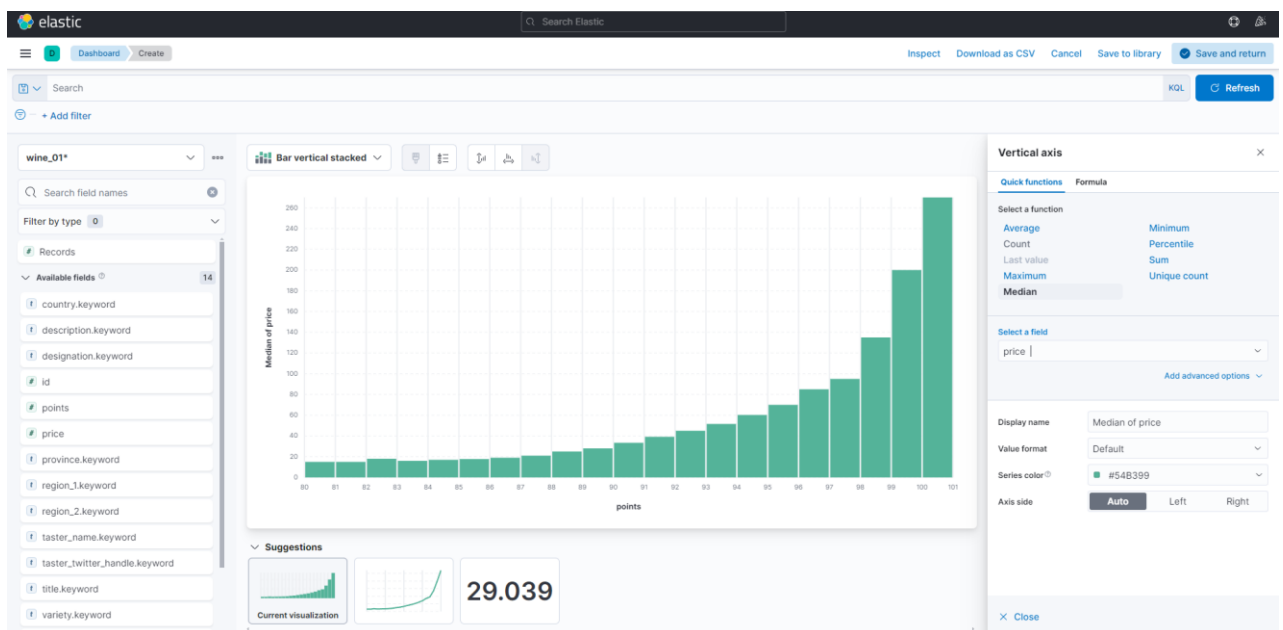
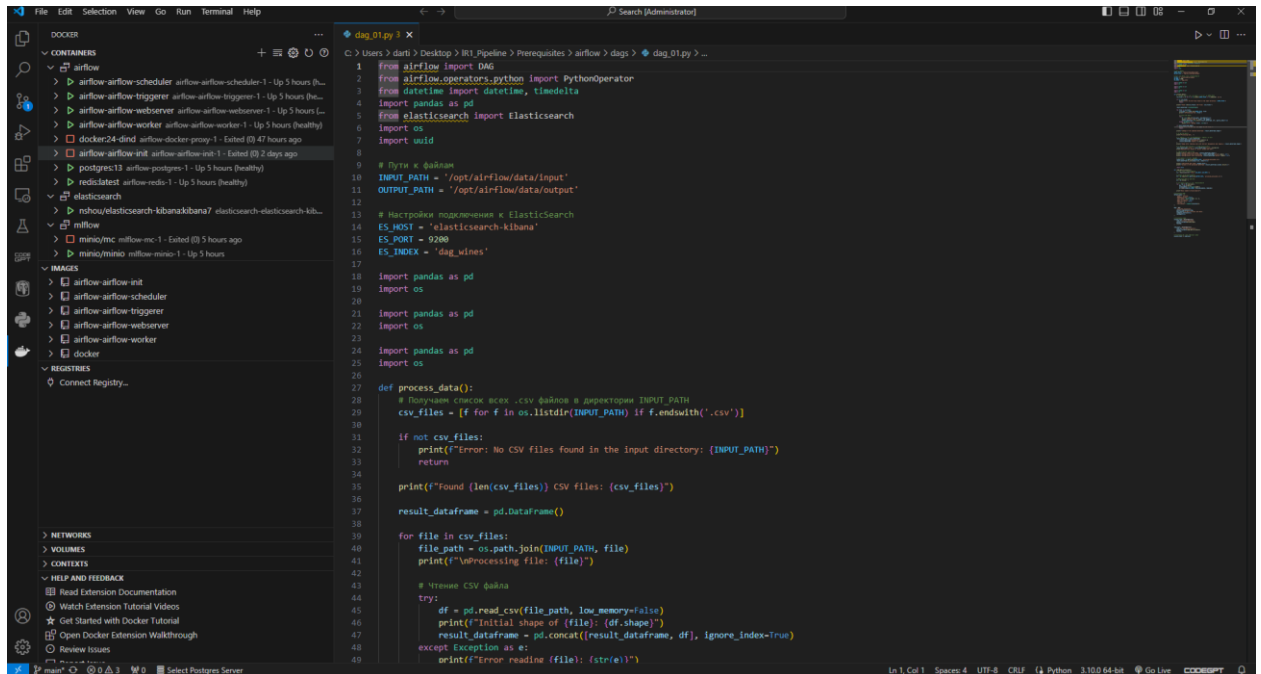


Рис. 29 – График price – point дата сета wine

Я понял что сохранил не тот xml, а после и вовсе потерял схему пришлось её создавать по новой для того что бы отправить в отчете.

Выполнение pipeline airflow-elasticsearch.

С данным ПО все было относительно получше, так как я уже набил шишки на этом датасете, а также логи airflow понятные, и описывают проблему раз в 10 лучше, чем это делают логи nifi. Которые просто показывают исключения java.



```
1 from airflow import DAG
2 from airflow.operators.python import PythonOperator
3 from datetime import datetime, timedelta
4 import pandas as pd
5 from elasticsearch import Elasticsearch
6 import os
7 import sys
8
9 # Путь к файлам
10 INPUT_PATH = '/opt/airflow/data/input'
11 OUTPUT_PATH = '/opt/airflow/data/output'
12
13 # Настройка подключения к Elasticsearch
14 ES_HOST = 'elasticsearch-kibana'
15 ES_PORT = 9200
16 ES_INDEX = 'dag_wines'
17
18 import pandas as pd
19 import os
20
21 import pandas as pd
22 import os
23
24 import pandas as pd
25 import os
26
27 def process_data():
28     # Проверяем наличие csv файлов в директории INPUT_PATH
29     csv_files = [f for f in os.listdir(INPUT_PATH) if f.endswith('.csv')]
30
31     if not csv_files:
32         print(f"Error: No CSV files found in the input directory: {INPUT_PATH}")
33         return
34
35     print(f"Found {len(csv_files)} CSV files: {csv_files}")
36
37     result_dataframe = pd.DataFrame()
38
39     for file in csv_files:
40         file_path = os.path.join(INPUT_PATH, file)
41         print(f"Processing file: {file}")
42
43         # Чтение CSV файла
44         try:
45             df = pd.read_csv(file_path, low_memory=False)
46             print(f"Initial shape of {file}: {df.shape}")
47             result_dataframe = pd.concat([result_dataframe, df], ignore_index=True)
48         except Exception as e:
49             print(f"Error reading {file}: {str(e)}")
```

Рис. 30 – Мой dag file

Для запуска DAG вручную я зашел на airflow, после нашел dag в разделе dags. Первый запуск прошел без ошибок, после я дополнял код новыми блоками.

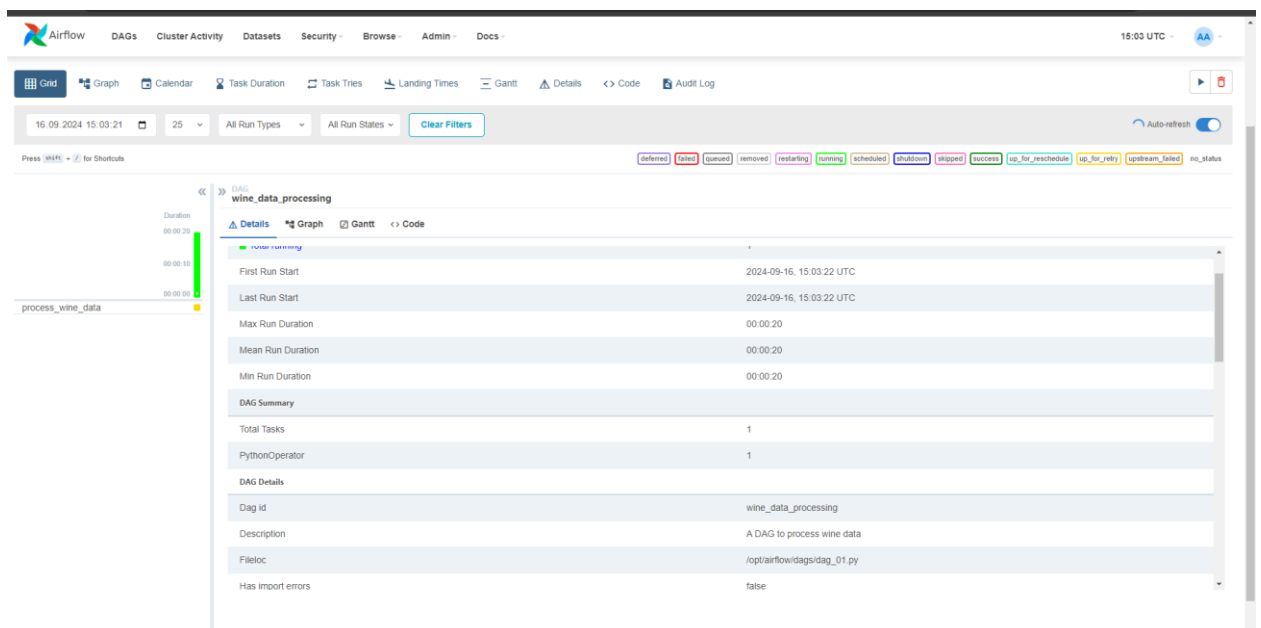


Рис. 31 – Первый запуск

Да я получал несколько ошибок и по большей части они были связаны с проблемой обработки данных, снова дело было в структуре, но проблема быстро решилась.

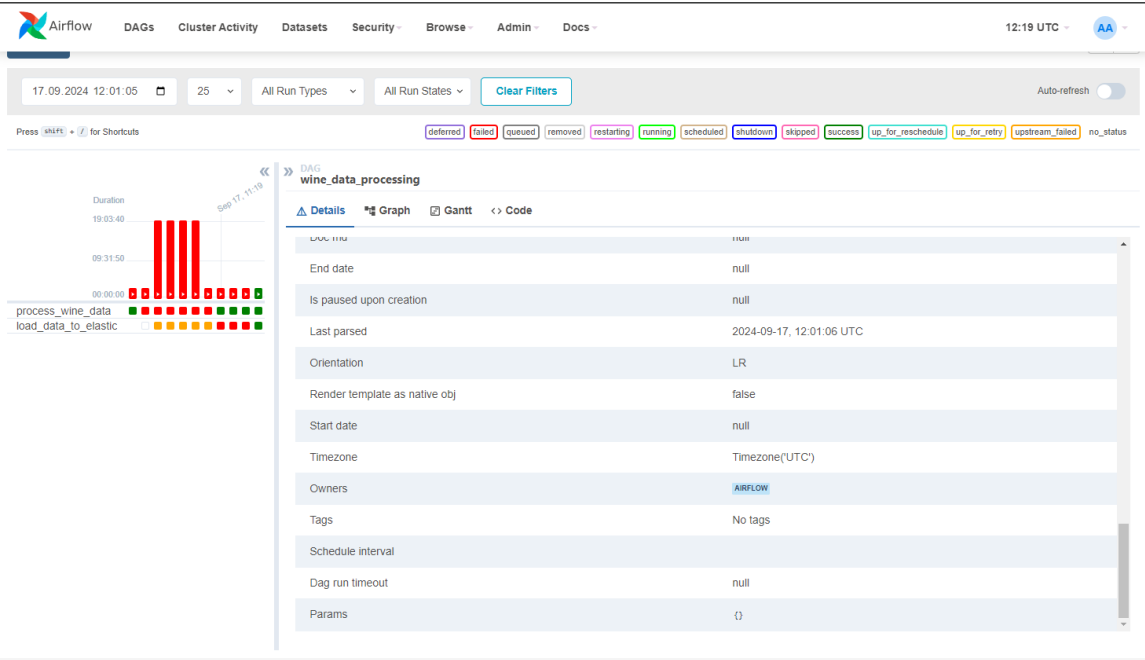


Рис. 32 – Успешный запуск dag после обновления кода

И вот уже мои данные объединены и закинуты на elasticsearch.

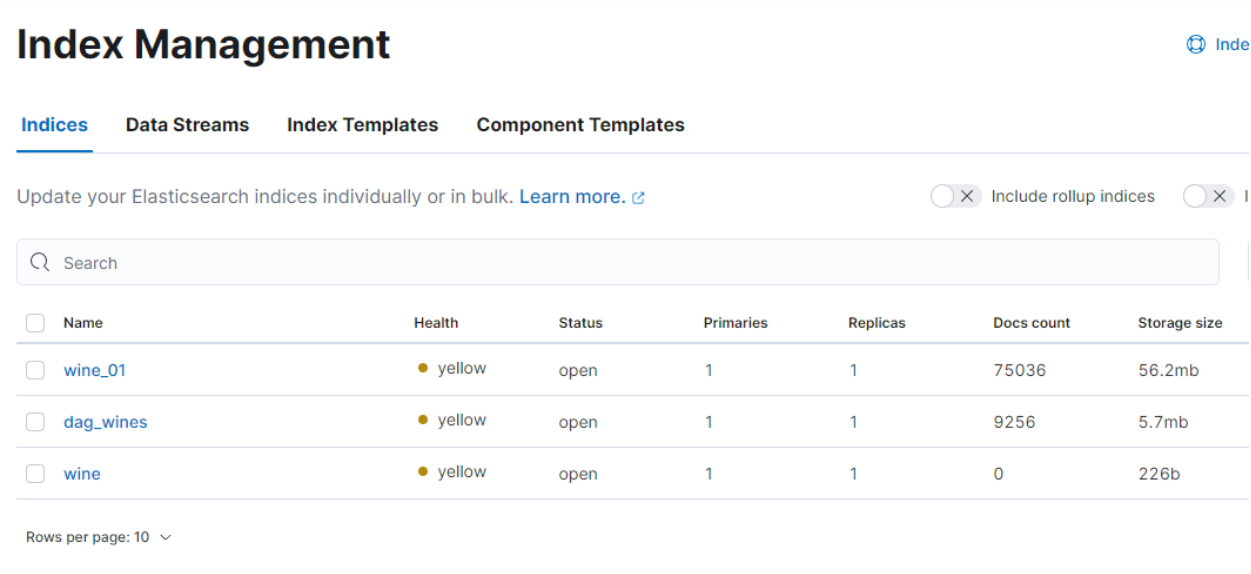


Рис. 33 – Индекс wine_01 дошел до elasticsearch.

Далее, как и с предыдущим индексом я создал pattern index. И построил график.

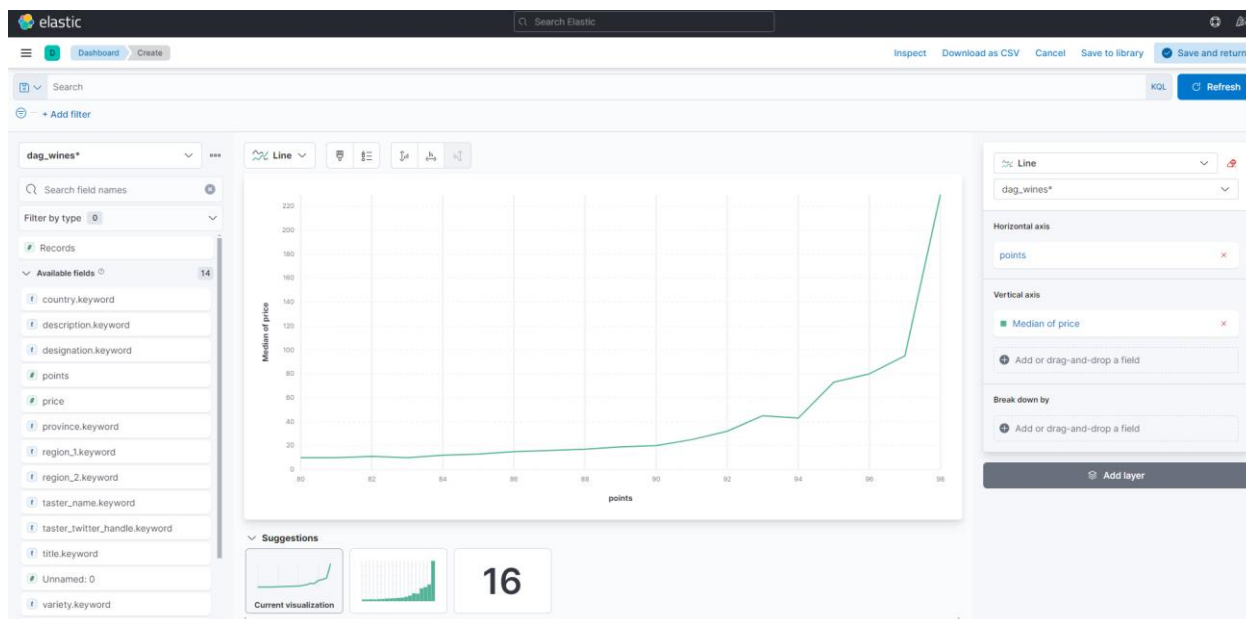


Рис. 34 – график price – point для датасета wine.

ЗАКЛЮЧЕНИЕ

В итоге, конечно, мне бы хотелось отметить, что работать с кодом на питоне гораздо удобнее чем с интерфейсом. Но тем не менее возможность nifi визуализировать pipeline является отличным способом понять куда, как и какие данные мы хотим отправлять, а также возможность создавать sql обработчик прямо в середине pipeline довольно притягательна. Airflow банально удобнее.

Вот + и – как мне кажется этих двух методов.

Apache Airflow	
<div><div>+</div><div>Гибкость Расширяемость Визуализация Логи</div></div>	<div><div>-</div><div>Не оптимален для потоков данных: лучше подходит для пакетной обработки задач, чем для потоковой.</div></div>
Apache NiFi	
<div><div>+</div><div>Потоковая обработка: отлично справляется с реальным временем и потоковыми данными.</div></div>	<div><div>-</div><div>Сложность в масштабировании Сложность починки из-за отсутствия внятных логов</div></div>