BIG DATA PROJECT
ANALYSIS ON CO2 EMISSION

Sonal Bisla, Vaidehi Atodaria, Lakshmi Sravanthi Naupada under supervision of Prof.Roy Kucukates

TORONTO METROPOLITAN UNIVERSITY MSc. in Data Science & Analytics

Table of Contents

1. Introduction

Global warming is the long-term heating of earth's surface due to raise in temperatures. In the 19th century there is a steep raise in greenhouse gases due to burning of fossil fuels and petroleum[2]. Each year there is raise in human activities and advancements in technology industries leading to raise of carbon dioxide especially in developed and developing countries. Among these green house gases carbon dioxide is the most dangerous gas leading to ocean acidification. Although many countries are taking steps to reduce carbon footprint there is still more to do. So, it is important to analyze the scenarios responsible for saving the world. This project is all about analyzing carbon dioxide emission factors and its impact on the world using big data tools.

1.1 Problem Definition

The project problem definition is:

- To analyze different countries for the carbon dioxide emission and knowing the top countries with highest emission.
- To analyze the different sources responsible for the co2 emission.
- To analyze if there is any relation between the co2 emission and the life expectancy for the recent years.

2. Dataset Description

The dataset that we selected is from world development indicators (DATA WORLD BANK WEBSITE). The variables are 52 countries and the carbon dioxide emission levels for the years between 1990 and 2019. Based on phase 1 analysis, further variables are taken up for future analysis. Detailed description is as follows.

Dataset link: https://databank.worldbank.org/source/world-development-indicators

2.1 Attribute Description

Phase 1: Cleaning and analysing 52 countries for the levels of emission and knowing the top 5 countries with highest emission of carbon dioxide.

- **Country name**: This column has all the 52 countries that are taken for analysis
- **Country Code**: Code for specific country.
- **Series code**: It is a related abbreviated column for the factors. It has short descriptions for the factors
- **Series name**: This is our factors column that has all the factors which we have taken into consideration for our analysis
- **Years :1990 to 2019 (CO_2 emission):** It's the overall total carbon dioxide emission in Kt (Kilotons) for the respective years.

Phase 2: For the top country with highest emission looking at the sources responsible and predicting which source is highly responsible.

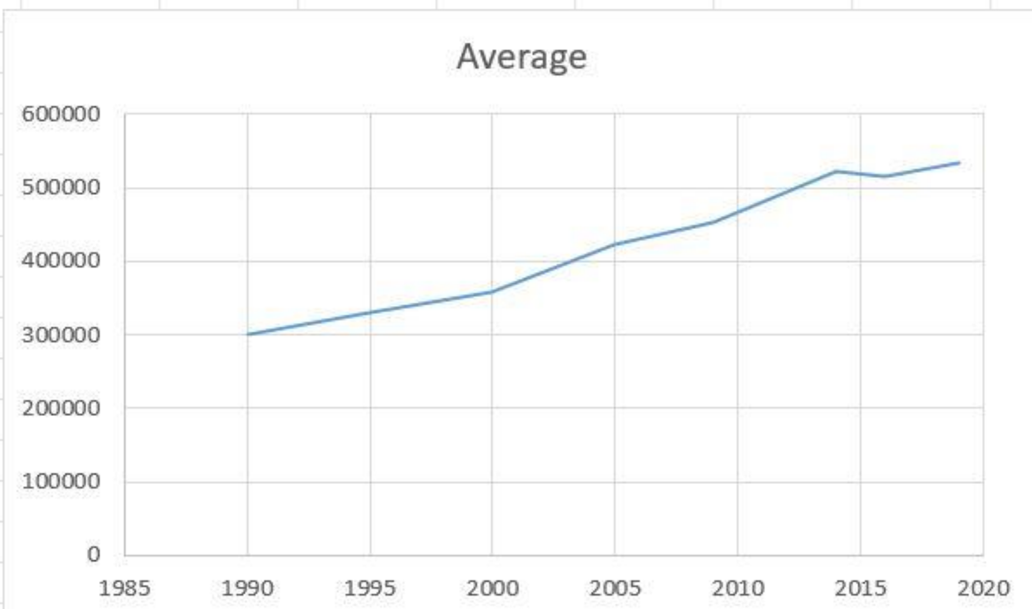
- **Co2 emission from Gaseous fuel:** co2 emitted from burning of natural gas or propane.
- **Co2 emission from Liquid fuel:** It is released by burning of liquid fuels such as gasoline.
- **Co2 emission from Solid fuel:** The emission from solid state is from different sources like burning of fossil fuels, production and deforestation.
- **Years selected**: 1980,2014,2018 & **Country:** USA

Phase 3: To detect the patterns of relation between the co2 emission and the life expectancy for the recent years.

- **Country_code**: It contains the country code for instance here we have considered the top 2 countries on our file i.e., CHN and USA
- **Country_name**: It contains the country names i.e., China and United States
- **Series_code**: It is an abbreviated column for the factors. It has short descriptions for the factors
- **Series_name**: This is our factors column that has all the factors which we have taken into consideration for our analysis
- **y_2014, y_2016, y_2019**

2.2 Statistics of data

Below graph shows there is a steep raise of co2 emission over the period.



3. Solution Description

PHASE 1: The phase 1 analysis is about cleaning the dataset that involves correcting the column name, removing null values, removing any duplicate values. Once we have our clean data, we worked on the summary statistics of all the columns to determine the mean and min max values.

At this stage we will be working on analysing the trends in emission from year 1990 to 2019 and determine the top countries for emission.

Tools used: Tools used for this analysis are **HDFS, PySpark** and **Tableau** and parallel analysis using **Hive.**

How they are used: HDFS system is used to store the data set. We used Pyspark for data preparation, cleaning and analysis as shown in the snippets below. And use Tableau for the visualisation graphs

WHY: Initially we started working on the data cleaning part using HIVE and Pyspark. Further analysis showed that Pyspark is giving quick results and cleaning using hive is a little challenging especially while removing the rows that are unnecessary. And Pyspark can be implemented using multiple languages Python, R, Scala and java. And the computational speed is high in Pyspark compared to hive and while working with more variables Pyspark is giving quick results.

Code snippets:

DATA PREPERATION:

Using HDFS as storage container

```
[root@sandbox-hdp ~]# cd olm
[root@sandbox-hdp olm]# hadoop fs -put Countries.csv /user/root/olm
[root@sandbox-hdp olm]# pyspark
```

This data frame df shows the initial dataset with the variables of 52 countries and the co2 emission for all the years 1990 to 2019.

```
ortonworks.com:8020/user/root/Countries.csv;
>>> df = spark.read.option("header",True) \
...     .csv("olm/Countries.csv")
>>> df.show(6)
+-----------+------------+-------------------+--------------+------------+------------+------------+----------------+----------------+----------------+----------------+----------------+
|Country Name|Country Code|      Series Name|   Series Code|1990 [YR1990]|1995 [YR1995]|2000 [YR2000]|   2005 [YR2005]|   2009 [YR2009]|   2014 [YR2014]|   2016 [YR2016]|   2019 [YR2019]|
+-----------+------------+-------------------+--------------+------------+------------+------------+----------------+----------------+----------------+----------------+----------------+
| Afghanistan|         AFG|CO2 emissions (kt)|EN.ATM.CO2E.KT|        2380|        1240|         760|1549.99995231628|4880.00011444092|4880.00011444092|5300.00019073486|6079.99992370605|
|     Algeria|         DZA|CO2 emissions (kt)|EN.ATM.CO2E.KT|       62940|       76440|       80050|94190.0024414063|112169.998168945|147740.005493164|154910.003662109|          171250|
|   Australia|         AUS|CO2 emissions (kt)|EN.ATM.CO2E.KT|      263630|      290180|      339450|370089.996337891|395290.008544922|371630.004882813|384989.990234375|386529.998779297|
|     Austria|         AUT|CO2 emissions (kt)|EN.ATM.CO2E.KT|       58270|       61180|       63530|76239.9978637695|64419.9981689453|62049.9992370605|63680.0003051758|64769.9966430664|
|Bahamas, The|         BHS|CO2 emissions (kt)|EN.ATM.CO2E.KT|        1960|        2200|        2230|2109.99989509583|6170.00007629395|2519.99998092651|2039.99996185303|2839.99991416931|
|  Bangladesh|         BGD|CO2 emissions (kt)|EN.ATM.CO2E.KT|       11520|       16550|       21650|32709.9990844727|           44750|63830.0018310547| 74379.997253418|90739.9978637695|
+-----------+------------+-------------------+--------------+------------+------------+------------+----------------+----------------+----------------+----------------+----------------+
```

This schema shows the datatype of the variables

```
>>> df.printSchema()
root
 |-- Country Name: string (nullable = true)
 |-- Country Code: string (nullable = true)
 |-- Series Name: string (nullable = true)
 |-- Series Code: string (nullable = true)
 |-- 1990 [YR1990]: string (nullable = true)
 |-- 1995 [YR1995]: string (nullable = true)
 |-- 2000 [YR2000]: string (nullable = true)
 |-- 2005 [YR2005]: string (nullable = true)
 |-- 2009 [YR2009]: string (nullable = true)
 |-- 2014 [YR2014]: string (nullable = true)
 |-- 2016 [YR2016]: string (nullable = true)
 |-- 2019 [YR2019]: string (nullable = true)
```

To get the count of variables we use the following code

```
>>> from pyspark .sql.functions import count
>>> rows = df.count()
```

```
>>> print(rows)
57
```

CLEANING:

Correcting the column name to correct format

```
>>> df =df.withColumnRenamed("Country Name","Country_name")
>>> df=df.withColumnRenamed("Country code", "country_code")
>>> df=df.withColumnRenamed("Series Name", "series_name")
>>> df=df.withColumnRenamed("Series Code", "series_code")
>>> df=df.withColumnRenamed("2009 [YR2009]", "y_2009")
>>> df=df.withColumnRenamed("1990 [YR1990]", "y_1990")
>>> df=df.withColumnRenamed("1995 [YR1995]", "y_1995")
>>> df=df.withColumnRenamed("2000 [YR2000]", "y_2000")
>>> df=df.withColumnRenamed("2005 [YR2005]", "y_2005")
>>> df=df.withColumnRenamed("2014 [YR2014]", "y_2014")
>>> df=df.withColumnRenamed("2016 [YR2016]", "y_2016")
>>> df=df.withColumnRenamed("2019 [YR2019]", "y_2019")
>>> df.printSchema()
root
 |-- Country_name: string (nullable = true)
 |-- country_code: string (nullable = true)
 |-- series_name: string (nullable = true)
 |-- series_code: string (nullable = true)
 |-- y_1990: string (nullable = true)
 |-- y_1995: string (nullable = true)
 |-- y_2000: string (nullable = true)
 |-- y_2005: string (nullable = true)
 |-- y_2009: string (nullable = true)
 |-- y_2014: string (nullable = true)
 |-- y_2016: string (nullable = true)
 |-- y_2019: string (nullable = true)
```

Then we have to change the type casting for the variables changing the string type to integer for all the years.

```
>>> from pyspark.sql.types import IntegerType
>>> df = df.withColumn("y_1990", df["y_1990"].cast(IntegerType()))
>>> df = df.withColumn("y_1995", df["y_1995"].cast(IntegerType()))
>>> df = df.withColumn("y_2000", df["y_2000"].cast(IntegerType()))
>>> df = df.withColumn("y_2005", df["y_2005"].cast(IntegerType()))
>>> df = df.withColumn("y_2009", df["y_2009"].cast(IntegerType()))
>>> df = df.withColumn("y_2014", df["y_2014"].cast(IntegerType()))
>>> df = df.withColumn("y_2016", df["y_2016"].cast(IntegerType()))
>>> df = df.withColumn("y_2019", df["y_2019"].cast(IntegerType()))
>>> df.printSchema()
root
 |-- Country_name: string (nullable = true)
 |-- country_code: string (nullable = true)
 |-- series_name: string (nullable = true)
 |-- series_code: string (nullable = true)
 |-- y_1990: integer (nullable = true)
 |-- y_1995: integer (nullable = true)
 |-- y_2000: integer (nullable = true)
 |-- y_2005: integer (nullable = true)
 |-- y_2009: integer (nullable = true)
 |-- y_2014: integer (nullable = true)
 |-- y_2016: integer (nullable = true)
 |-- y_2019: integer (nullable = true)
```

ANALYSIS:

```
>>> df_numeric = df.select('1990','1995','2000','2005',' 2009',' 2014','2019')
Traceback (most recent call last):
```

```
>>> df_numeric.show(3)
+------+------+------+------+------+------+------+
|y_1990|y_1995|y_2000|y_2005|y_2009|y_2014|y_2019|
+------+------+------+------+------+------+------+
|  2380|  1240|   760|  1549|  4880|  4880|  6079|
| 62940| 76440| 80050| 94190|112169|147740|171250|
|263630|290180|339450|370089|395290|371630|386529|
+------+------+------+------+------+------+------+
only showing top 3 rows
```

The summary statistics of the variables and the years are shown below

```
>>> df_numeric.describe().show()
+-------+-----------------+-----------------+-----------------+------------------+------------------+-----------------+-----------------+
|summary|           y_1990|           y_1995|           y_2000|            y_2005|            y_2009|           y_2014|           y_2019|
+-------+-----------------+-----------------+-----------------+------------------+------------------+-----------------+-----------------+
|  count|               51|               51|               51|                51|                51|               51|               51|
|   mean|299413.92156862747|329594.70588235295|358282.1568627451|422948.09803921566| 453194.5882352941|521618.09803921566| 533992.8039215687|
| stddev| 746245.8989504154| 830384.134656613|925095.3263687232|1126740.8466374378|1280758.0224296267| 1558597.303681375|1633926.1305377674|
|    min|               80|              160|              210|               319|               319|              649|              689|
|    max|          4844520|          5117040|          5775810|           5824629|           7719069|         10006669|         10707219|
+-------+-----------------+-----------------+-----------------+------------------+------------------+-----------------+-----------------+
```

Here we see the average value for emission is highest for the year 2019 compared to all other years.

```
>>> from pyspark.sql.functions import col
>>> df.select(col("Country_name"),col("y_2019")).show()
+--------------+--------+
|  Country_name|  y_2019|
+--------------+--------+
|   Afghanistan|    6079|
|       Algeria|  171250|
|     Australia|  386529|
|       Austria|   64769|
|  Bahamas, The|    2839|
|    Bangladesh|   90739|
|       Bahrain|   33259|
|       Belgium|   93010|
|       Bolivia|   22340|
|        Brazil|  434299|
|        Canada|  580210|
|         China|10707219|
|      Bulgaria|   39139|
|United Kingdom|  348920|
| United States| 4817720|
|       Ukraine|  174729|
|      Thailand|  267089|
|   Switzerland|   37380|
|        Sweden|   35000|
|  South Africa|  439640|
+--------------+--------+
only showing top 20 rows
```

```
>>> df.sort(col("y_2019").desc()).show(truncate=False)
+--------------+------------+------------------+-------------+-------+-------+-------+-------+-------+--------+-------+--------+
|Country_name  |country_code|series_name       |series_code  |y_1990 |y_1995 |y_2000 |y_2005 |y_2009 |y_2014  |y_2016 |y_2019  |
+--------------+------------+------------------+-------------+-------+-------+-------+-------+-------+--------+-------+--------+
|China         |CHN         |CO2 emissions (kt)|EN.ATM.CO2E.KT|2173360|3088620|3346530|5824629|7719069|10006669|9874660|10707219|
|United States |USA         |CO2 emissions (kt)|EN.ATM.CO2E.KT|4844520|5117040|5775810|5753490|5156430|5107209 |4894500|4817720 |
|India         |IND         |CO2 emissions (kt)|EN.ATM.CO2E.KT|563580 |737860 |937860 |1136469|1564880|2147110 |2195250|2456300 |
|Japan         |JPN         |CO2 emissions (kt)|EN.ATM.CO2E.KT|1090530|1171010|1182610|1212819|1100979|1217119 |1167790|1081569 |
|Germany       |DEU         |CO2 emissions (kt)|EN.ATM.CO2E.KT|955310 |874660 |830280 |802380 |734809 |736010  |747150 |657400  |
|Indonesia     |IDN         |CO2 emissions (kt)|EN.ATM.CO2E.KT|148530 |223680 |280650 |342149 |391079 |483910  |482510 |619840  |
|Canada        |CAN         |CO2 emissions (kt)|EN.ATM.CO2E.KT|419490 |448050 |514220 |549030 |521320 |561679  |556830 |580210  |
|Saudi Arabia  |SAU         |CO2 emissions (kt)|EN.ATM.CO2E.KT|171410 |204830 |249660 |315290 |406529 |540520  |561229 |523780  |
|Mexico        |MEX         |CO2 emissions (kt)|EN.ATM.CO2E.KT|269580 |306840 |379180 |432190 |448369 |462239  |473309 |449269  |
|South Africa  |ZAF         |CO2 emissions (kt)|EN.ATM.CO2E.KT|247660 |264310 |284660 |377649 |404200 |447929  |425140 |439640  |
|Brazil        |BRA         |CO2 emissions (kt)|EN.ATM.CO2E.KT|197900 |241280 |313670 |331690 |350000 |511619  |447079 |434299  |
|Australia     |AUS         |CO2 emissions (kt)|EN.ATM.CO2E.KT|263630 |290180 |339450 |370089 |395290 |371630  |384989 |386529  |
|United Kingdom|GBR         |CO2 emissions (kt)|EN.ATM.CO2E.KT|561770 |526810 |530890 |540919 |466489 |415600  |380809 |348920  |
|Italy         |ITA         |CO2 emissions (kt)|EN.ATM.CO2E.KT|405260 |416420 |436300 |473829 |397059 |327500  |333339 |317239  |
|France        |FRA         |CO2 emissions (kt)|EN.ATM.CO2E.KT|356240 |352240 |373120 |380660 |343730 |306100  |313920 |300519  |
|Poland        |POL         |CO2 emissions (kt)|EN.ATM.CO2E.KT|350210 |340000 |295770 |301350 |297260 |285730  |299799 |295130  |
|Thailand      |THA         |CO2 emissions (kt)|EN.ATM.CO2E.KT|89220  |155780 |164490 |217770 |220259 |256799  |261600 |267089  |
|Malaysia      |MYS         |CO2 emissions (kt)|EN.ATM.CO2E.KT|54620  |86310  |124360 |167419 |181929 |236649  |235960 |253270  |
|Pakistan      |PAK         |CO2 emissions (kt)|EN.ATM.CO2E.KT|59030  |82740  |98370  |121669 |145139 |154240  |181110 |190570  |
|Ukraine       |UKR         |CO2 emissions (kt)|EN.ATM.CO2E.KT|688620 |399250 |297380 |295410 |251619 |237729  |201660 |174729  |
+--------------+------------+------------------+-------------+-------+-------+-------+-------+-------+--------+-------+--------+
only showing top 20 rows
```

Phase 2: From the phase1 analysis we have found out that China and USA are the most affected country by co2 emission. In phase2, we will analyze which factors are responsible for c02 emission. Although, there are so many factors responsible for it, we have chosen the below sources

co2 emission from gaseous fuel,

C02 emission from liquid fuel,

Co2 emission from solid fuel

For the years 1980,2014, 2016.

Tools used: Data Analysis tools used are **Apache hive cluster** on **Databricks cloud** platform, and visualization tool **Ambari** is used for detailed data analysis for yearly records

Why used: Hive is built on top of Apache Hadoop, which is an open-source framework used to efficiently store and process large datasets.

How: Hive organizes tables into partitions based on partition keys for grouping similar data together.

The partitions are further categorized into buckets based on the hash function of a column in the table. These buckets are stored as a file in the partition directory.

It makes MapReduce programming easier because you don't have to know and write lengthy Java code

Each table in the hive can have one or more partition keys to identify a particular partition. Using partition, it is easy to do queries on slices of the data.

Hive Partitioning Advantages:

Partitioning in Hive distributes execution load horizontally.

In partition faster execution of queries with the low volume of data takes place. For example, the search population from USA returns very fast instead of searching every country.

Code snippets:

Non partition table and Partition(country) table:

```
hive> create table phasetb(country_name string,country_code string,
    > series_name string,series_code string,
    > y_1980 string,y_2014 string,y_2016 string)
    > row format delimited fields terminated by ',';
OK
Time taken: 1.594 seconds
hive> create table phase_co(country_name string,country_code string,
    > series_name string,series_code string,
    > y_1980 string,y_2014 string,y_2016 string)
    > partitioned by(country string)
    > clustered by(country_code) into 2 buckets;
OK
Time taken: 1.3 seconds
```

Insert into table:

```
hive> insert overwrite table phase_co partition(country)
    > select country_name,country_code,series_name,series_code,y_1980,
    > y_2014,y_2016,(case when country_name like 'none' then 'none'
    > when country_name like '%(%' then regexp_extract(country_name,'\\(.*?)\\)',1) else 'United States' end) country
    > from phasetb;
Query ID = root_20221203232050_a9d7b085-91dd-4349-9abc-917f6d720e94
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1670078264189_0014)

----------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      2          2        0        0       0       0
----------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 26.18 s
----------------------------------------------------------------------------------
Loading data to table phase2.phase_co partition (country=null)
         Time taken to load dynamic partitions: 1.326 seconds
        Loading partition {country=United States}
         Time taken for adding to write entity : 3
Partition phase2.phase_co{country=United States} stats: [numFiles=1, numRows=7, totalSize=758, rawDataSize=751]
OK
Time taken: 51.096 seconds
```

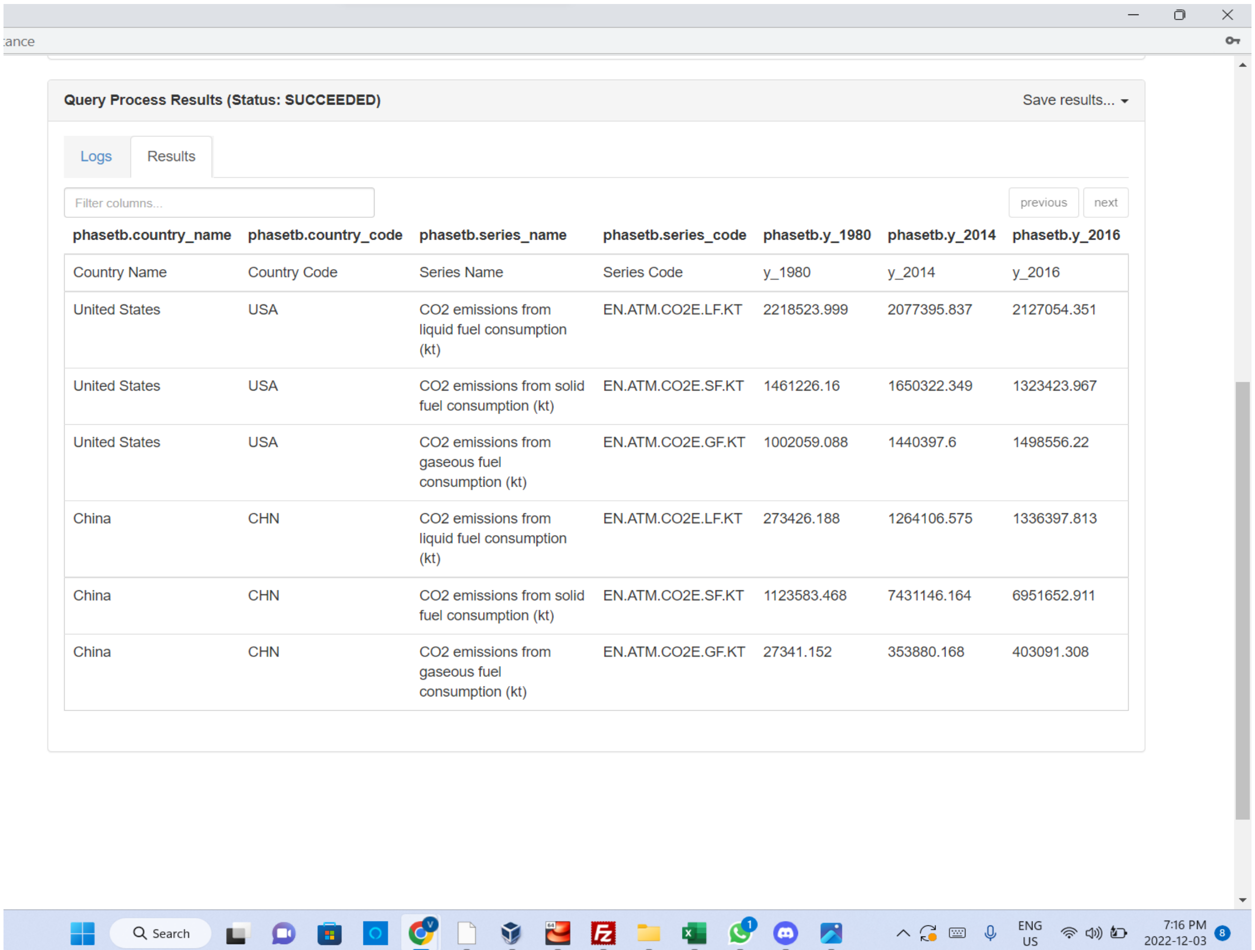
Partition on Hadoop cluster:

```
[root@sandbox-hdp ~]# hadoop dfs -ls /apps/hive/warehouse/phase2.db/phase_co
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Found 1 items
drwxrwxrwx   - root hadoop          0 2022-12-03 23:21 /apps/hive/warehouse/phase2.db/phase_co/country=United States
[root@sandbox-hdp ~]#
```

```
hive> show partitions phase_co;
OK
country=United States
Time taken: 1.134 seconds, Fetched: 1 row(s)
```

Database:



phasetb.country_name	phasetb.country_code	phasetb.series_name	phasetb.series_code	phasetb.y_1980	phasetb.y_2014	phasetb.y_2016
Country Name	Country Code	Series Name	Series Code	y_1980	y_2014	y_2016
United States	USA	CO2 emissions from liquid fuel consumption (kt)	EN.ATM.CO2E.LF.KT	2218523.999	2077395.837	2127054.351
United States	USA	CO2 emissions from solid fuel consumption (kt)	EN.ATM.CO2E.SF.KT	1461226.16	1650322.349	1323423.967
United States	USA	CO2 emissions from gaseous fuel consumption (kt)	EN.ATM.CO2E.GF.KT	1002059.088	1440397.6	1498556.22
China	CHN	CO2 emissions from liquid fuel consumption (kt)	EN.ATM.CO2E.LF.KT	273426.188	1264106.575	1336397.813
China	CHN	CO2 emissions from solid fuel consumption (kt)	EN.ATM.CO2E.SF.KT	1123583.468	7431146.164	6951652.911
China	CHN	CO2 emissions from gaseous fuel consumption (kt)	EN.ATM.CO2E.GF.KT	27341.152	353880.168	403091.308

Phase 3: In continuation of phase 1 and phase 2, here we are using the already cleaned data file. For phase 3, we have used elastic search tool.

Tools used: **Elastic Search, Kibana**

Elasticsearch is a distributed, free and open search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured. Elasticsearch is built on Apache Lucene and was first released in 2010 by Elasticsearch N.V. (now known as Elastic). Known for its simple REST APIs, distributed nature, speed, and scalability, Elasticsearch is the central component of the Elastic Stack, a set of free and open tools for data ingestion, enrichment, storage, analysis, and visualization. Commonly referred to as the ELK Stack (after Elasticsearch, Logstash, and Kibana), the Elastic Stack now includes a rich collection of lightweight shipping agents known as Beats for sending data to Elasticsearch.

Why used: It runs perfectly fine on any machine or in a cluster containing hundreds of nodes, and the experience is almost identical.

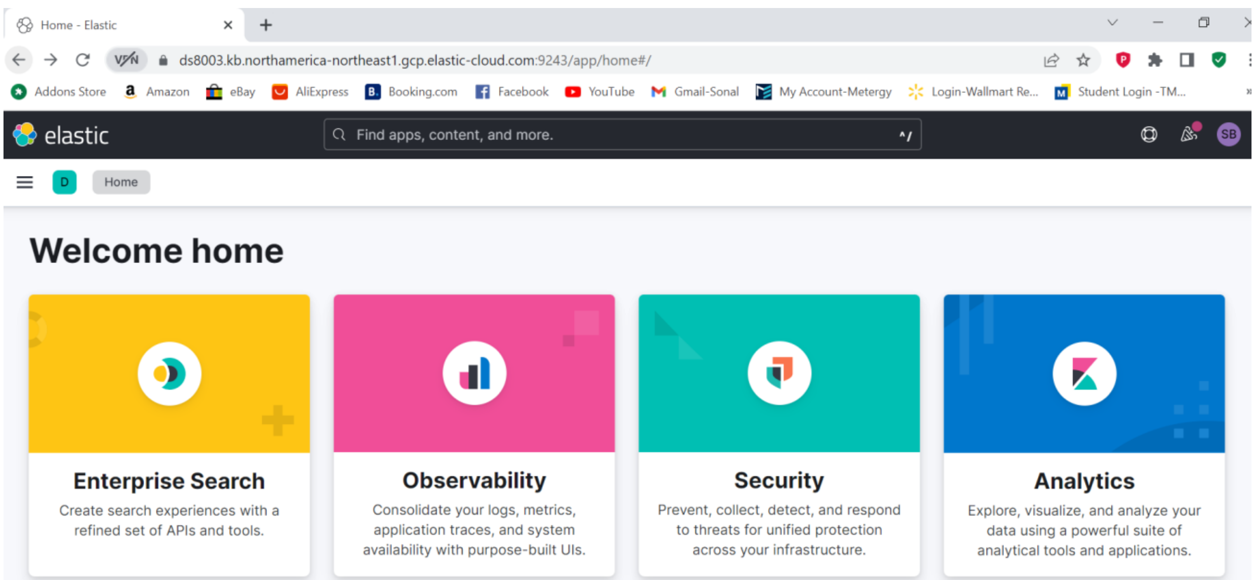
One can perform and combine various kind of searches irrespective of their data type which included structured, unstructured, geo and metrics data type.

Fast performance: By using distributed inverted indices, Elasticsearch quickly finds the best matches for your full-text searches from even very large data sets.

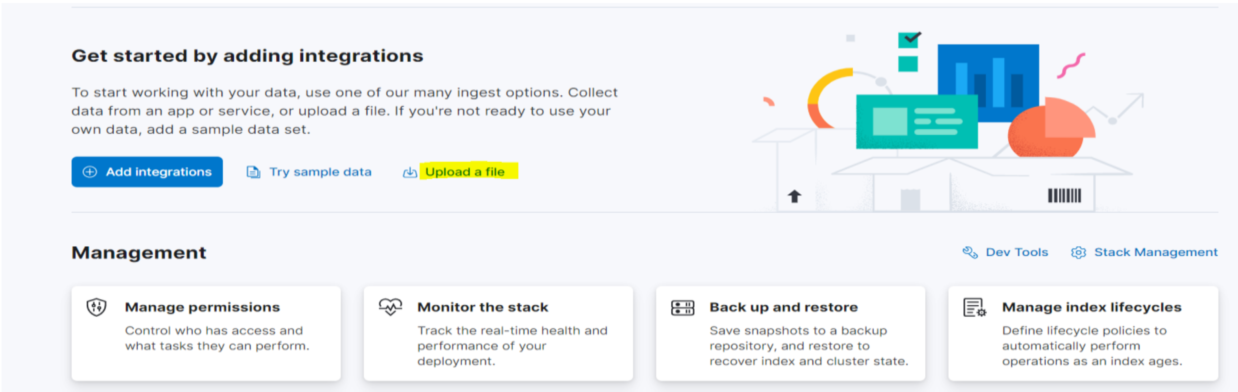
It also provides aggregations which can explore trends and patterns of data.

Kibana: Kibana is a data visualization and exploration tool. It offers powerful and easy-to-use features such as histograms, line graphs, pie charts, heat maps, and built-in geospatial support.

How: Create a free account and the first deployment with the deployment name ds8003 on the website https://www.elastic.co



Once we have created the deployment, we enter the elastic search page. We now will upload the csv file.

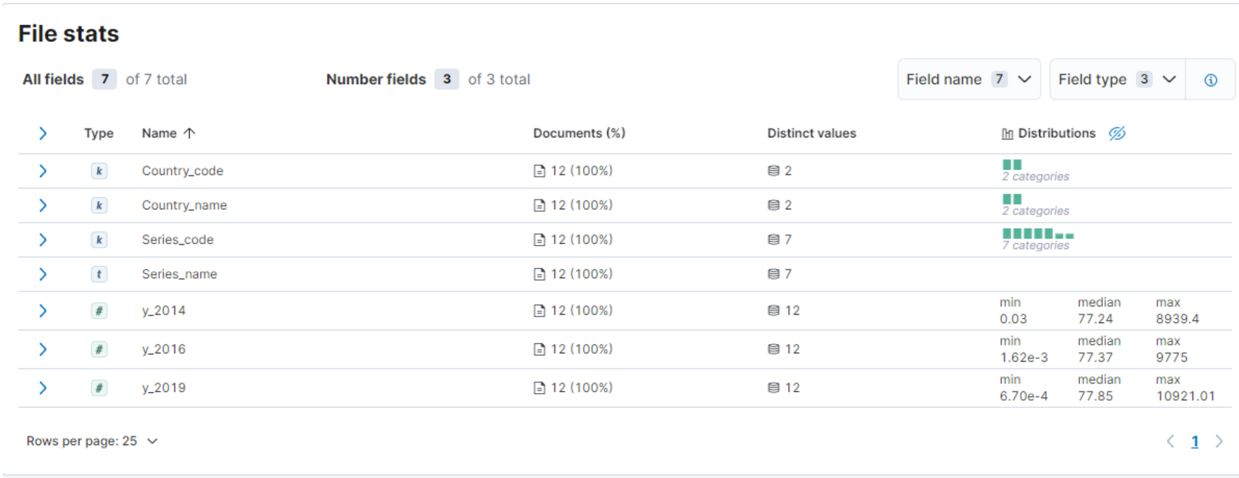


Once this part is done, we do the further steps as follows:

Code snippets:

Upload our file name life_expectancy.csv

Now here we can see our file was uploaded and it shows the stats which includes all the 7 column names/fields with the distinct values and distributions along with the min, max and median for the year columns respectively beside them.



File stats

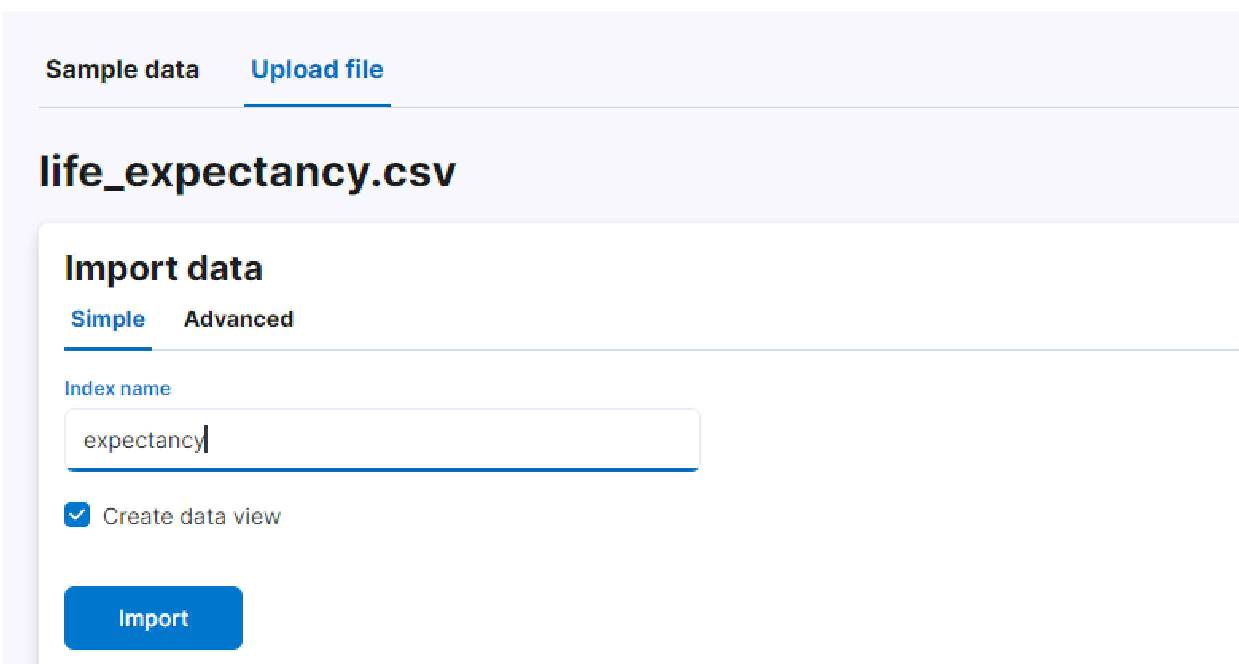
All fields 7 of 7 total | Number fields 3 of 3 total | Field name 7 | Field type 3

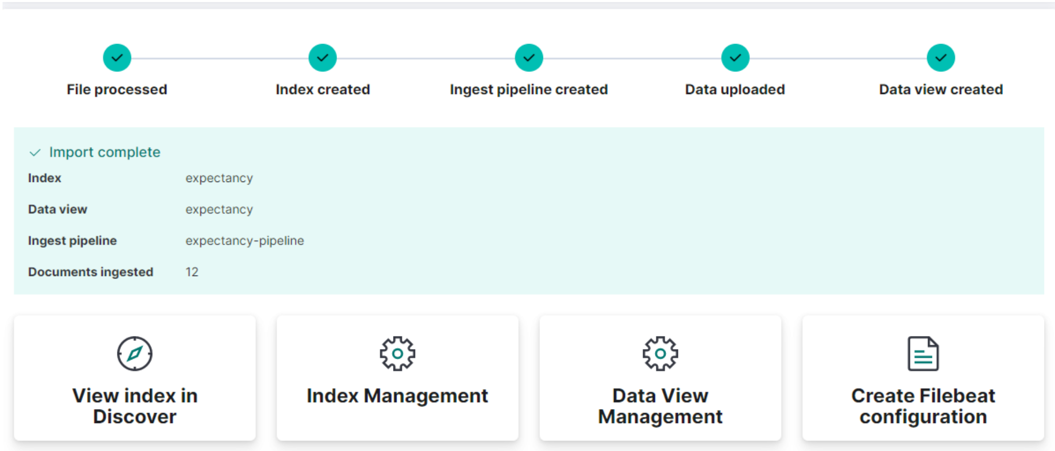
Type	Name	Documents (%)	Distinct values	Distributions
k	Country_code	12 (100%)	2	2 categories
k	Country_name	12 (100%)	2	2 categories
k	Series_code	12 (100%)	7	7 categories
t	Series_name	12 (100%)	7	
#	y_2014	12 (100%)	12	min 0.03, median 77.24, max 8939.4
#	y_2016	12 (100%)	12	min 1.62e-3, median 77.37, max 9775
#	y_2019	12 (100%)	12	min 6.70e-4, median 77.85, max 10921.01

Rows per page: 25

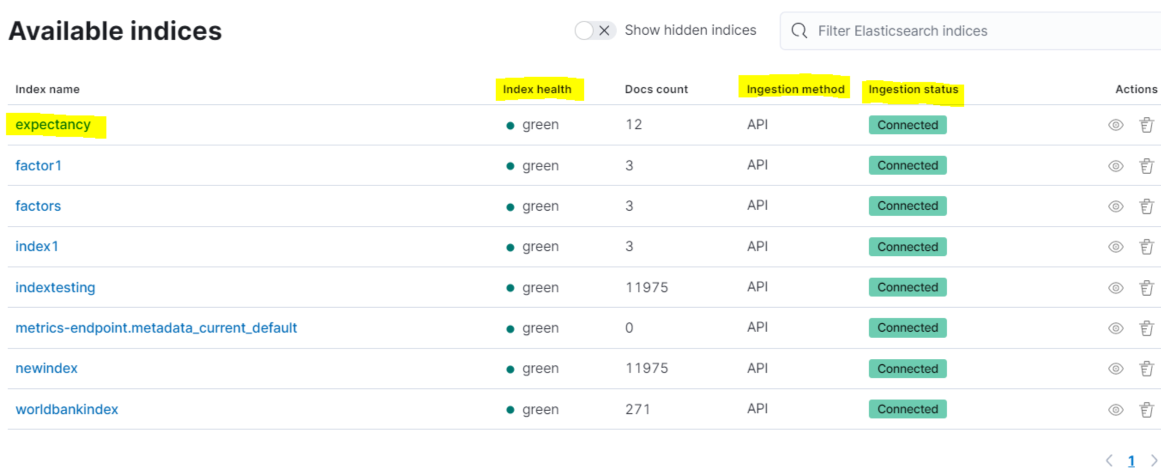
Note: In Elasticsearch, an index (plural: indices) contains a schema and can have one or more shards and replicas. An Elasticsearch index is divided into shards and each shard is an instance of a Lucene index. Indices are used to store the documents in dedicated data structures corresponding to the data type of fields.

Here we have created the index with the name expectancy:



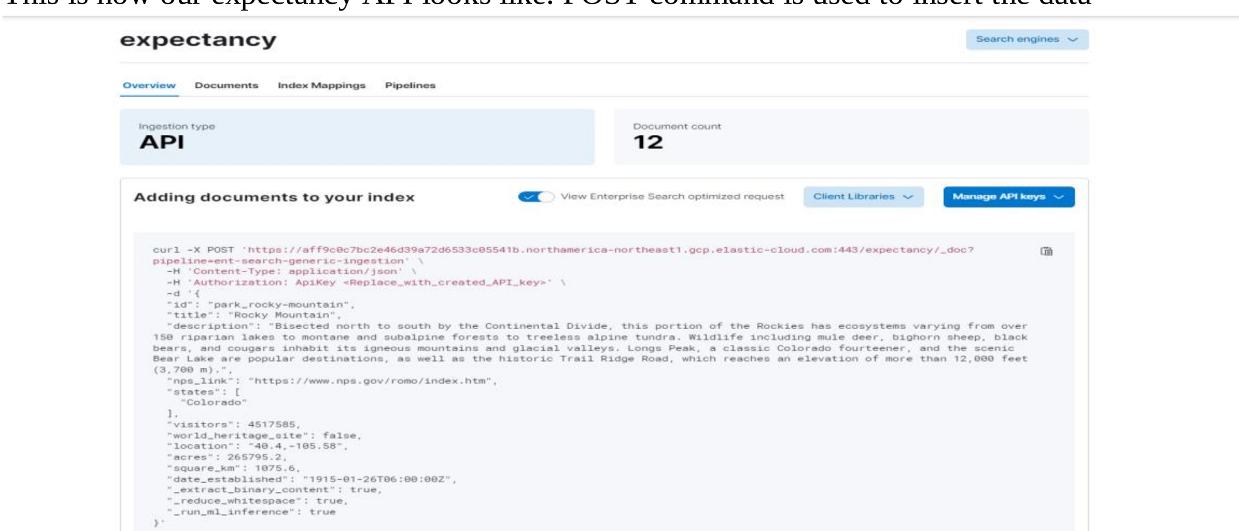


This is how our index looks below. Index health is green and is connected to the API. Ingestion status is also showing connected.

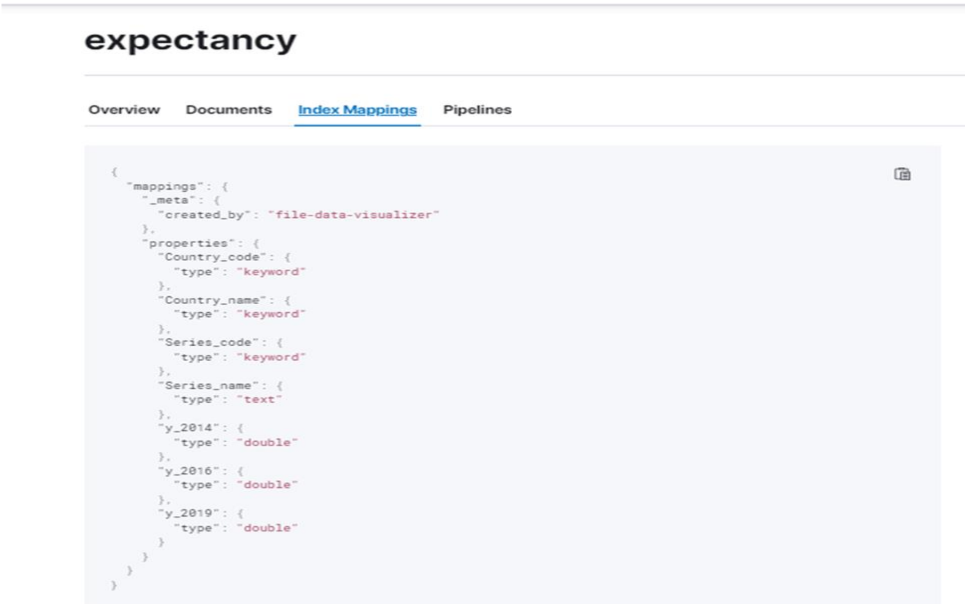


Index name	Index health	Docs count	Ingestion method	Ingestion status	Actions
expectancy	green	12	API	Connected	
factor1	green	3	API	Connected	
factors	green	3	API	Connected	
index1	green	3	API	Connected	
indextesting	green	11975	API	Connected	
metrics-endpoint.metadata_current_default	green	0	API	Connected	
newindex	green	11975	API	Connected	
worldbankindex	green	271	API	Connected	

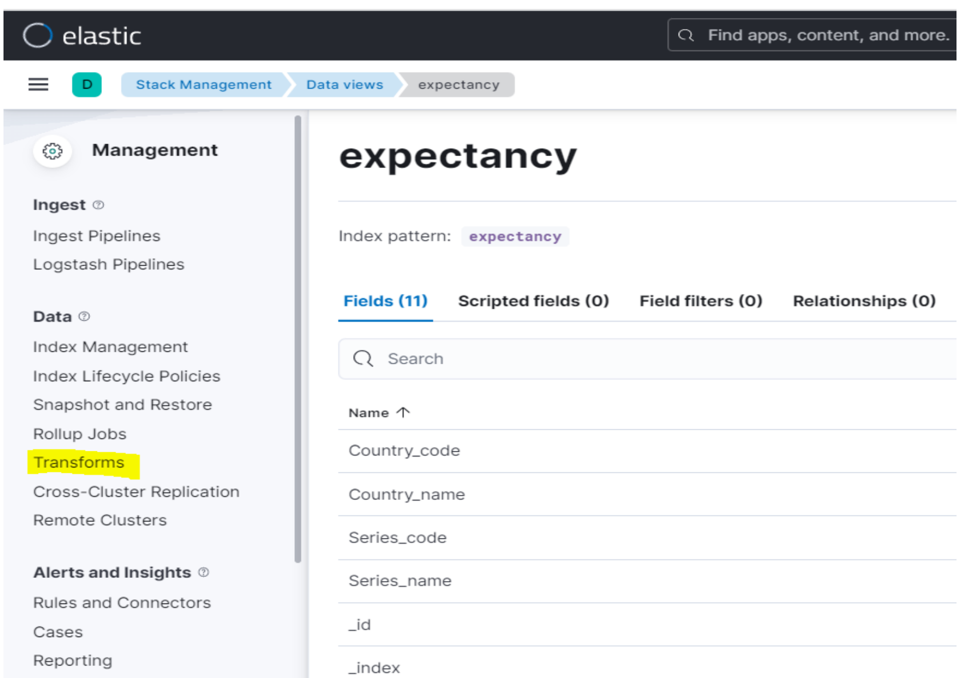
This is how our expectancy API looks like. POST command is used to insert the data



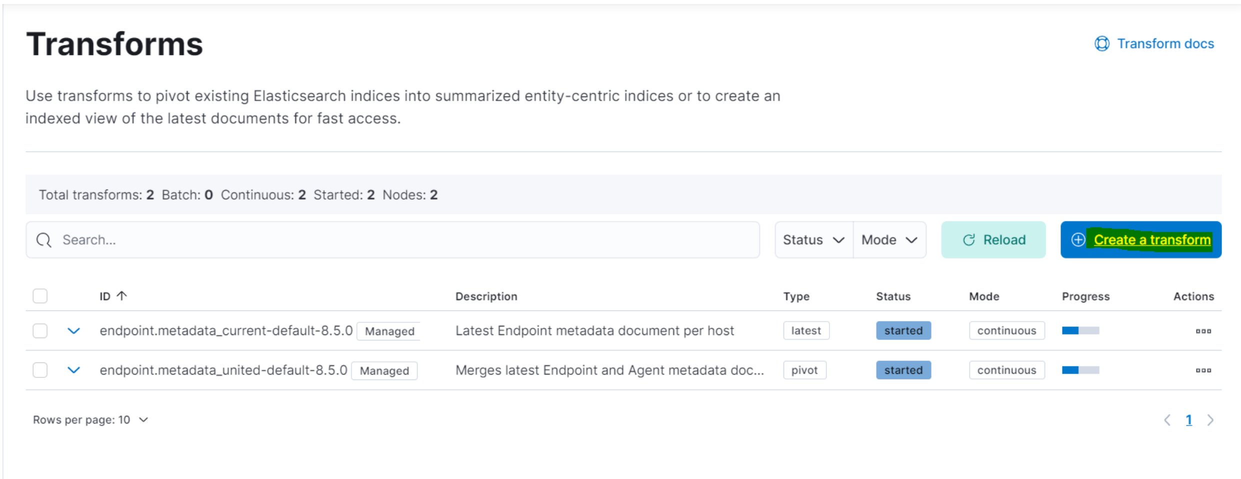
Index mapping:



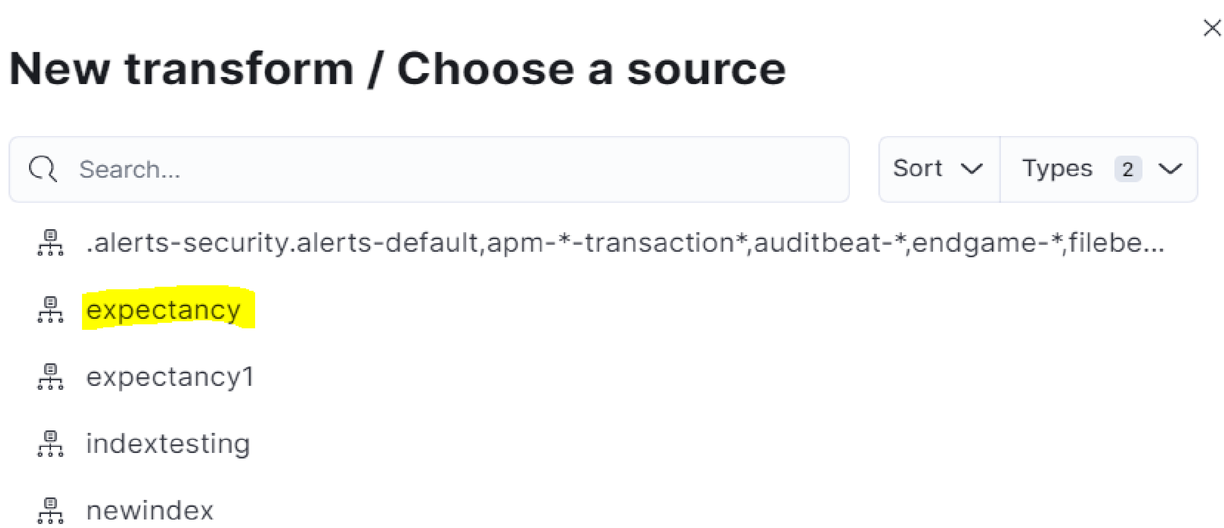
Transforms enable you to convert existing Elasticsearch indices into summarized indices, which provide opportunities for new insights and analytics. For example, you can use transforms to pivot your data into entity-centric indices that summarize the behavior of users or sessions or other entities in your data.



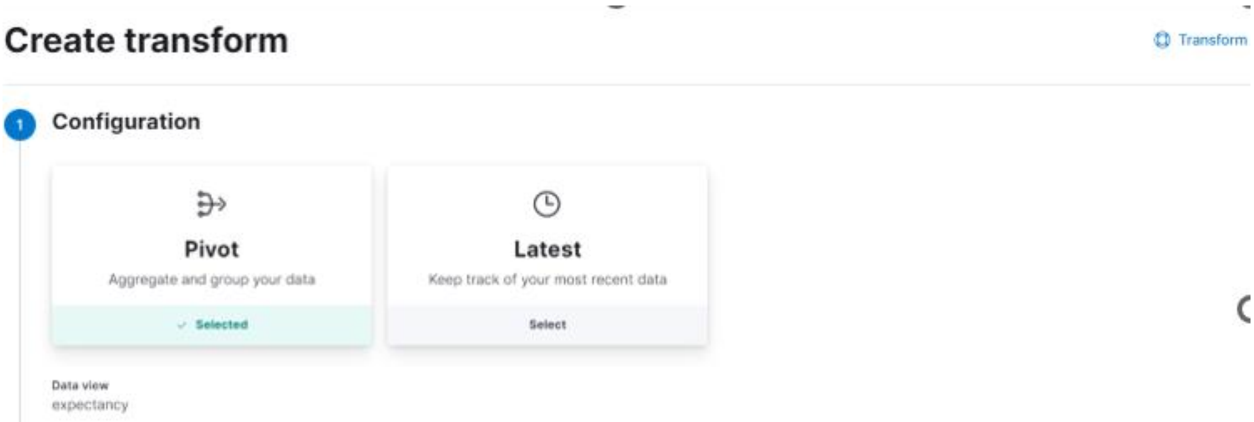
By clicking on a create a transform tab we will land up to a page where we can transform our data as per our requirements that we need for the better and clear visualizations further

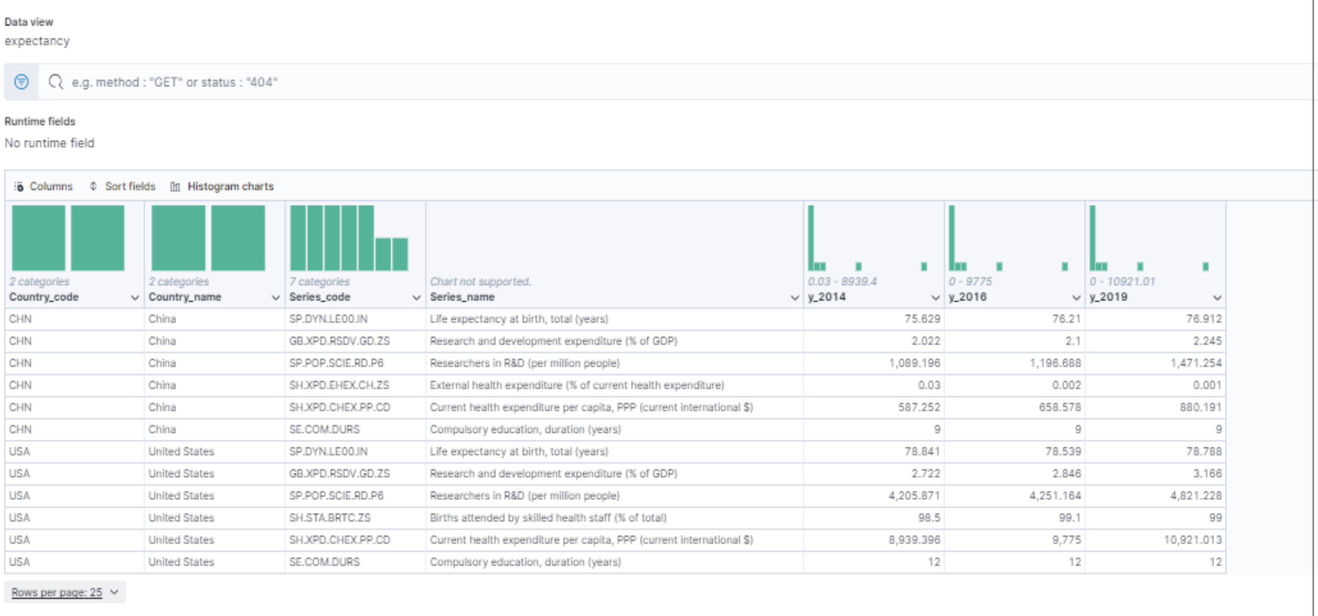


Select index:



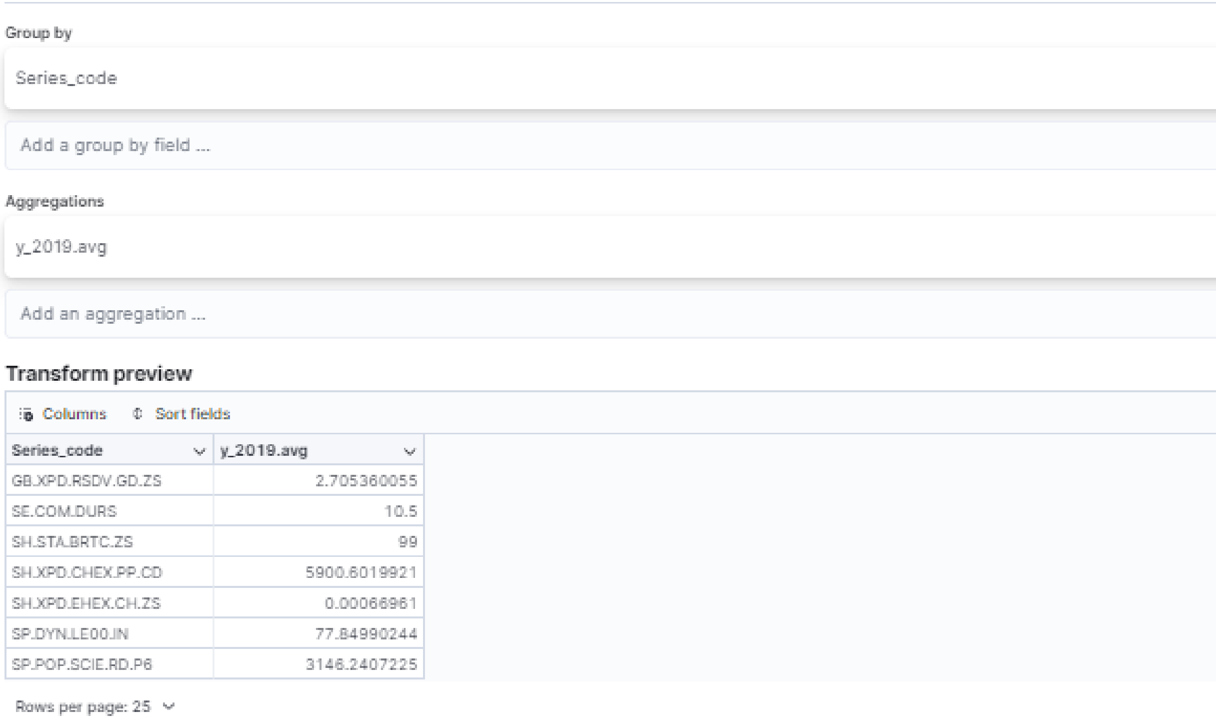
This gave us a pivot and group by options. Also, as we can see we are able to see our columns data without creating any tables. It generated histogram charts as well for each column.



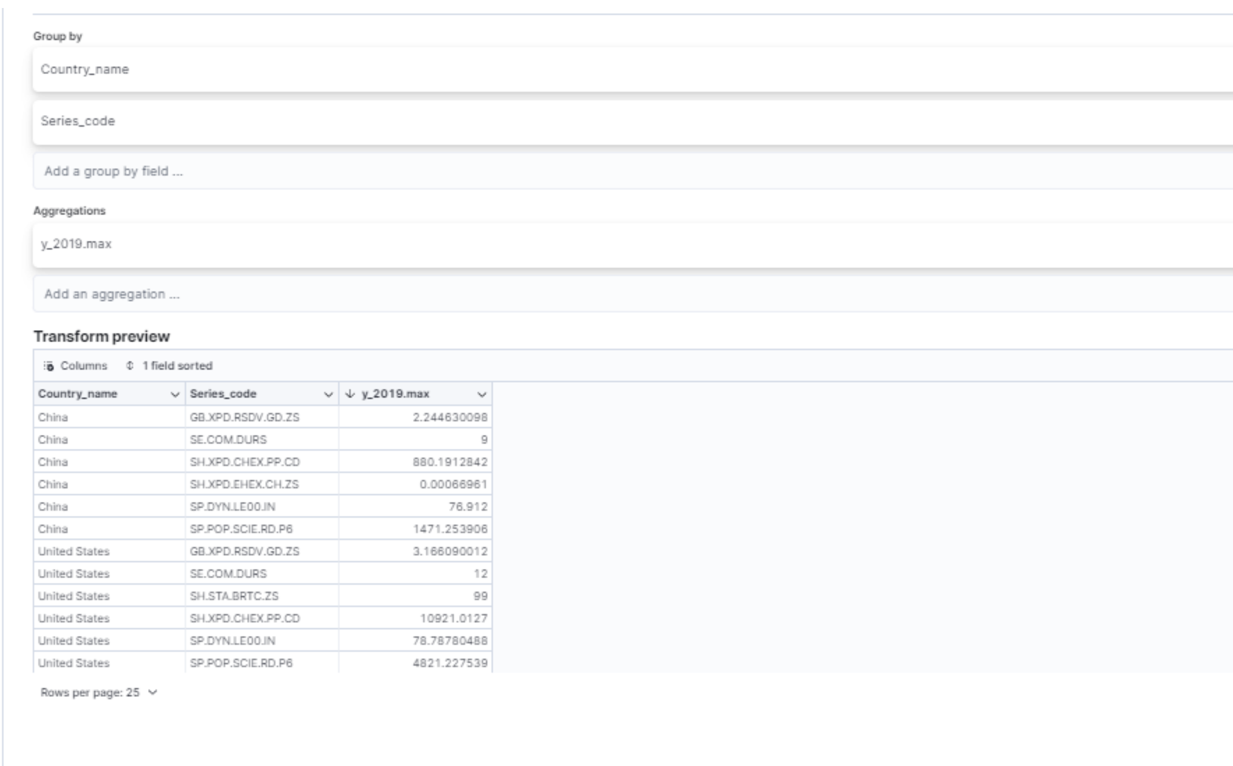


Country_code	Country_name	Series_code	Series_name	y_2014	y_2016	y_2019
CHN	China	SP.DYN.LE00.IN	Life expectancy at birth, total (years)	75.629	76.21	76.912
CHN	China	GB.XPD.RSDV.GD.ZS	Research and development expenditure (% of GDP)	2.022	2.1	2.245
CHN	China	SP.POP.SCIE.RD.P6	Researchers in R&D (per million people)	1,089.196	1,196.688	1,471.254
CHN	China	SH.XPD.EHEX.CH.ZS	External health expenditure (% of current health expenditure)	0.03	0.002	0.001
CHN	China	SH.XPD.CHEX.PP.CD	Current health expenditure per capita, PPP (current international $)	587.252	658.578	880.191
CHN	China	SE.COM.DURS	Compulsory education, duration (years)	9	9	9
USA	United States	SP.DYN.LE00.IN	Life expectancy at birth, total (years)	78.841	78.539	78.788
USA	United States	GB.XPD.RSDV.GD.ZS	Research and development expenditure (% of GDP)	2.722	2.846	3.166
USA	United States	SP.POP.SCIE.RD.P6	Researchers in R&D (per million people)	4,205.871	4,251.164	4,821.228
USA	United States	SH.STA.BRTC.ZS	Births attended by skilled health staff (% of total)	98.5	99.1	99
USA	United States	SH.XPD.CHEX.PP.CD	Current health expenditure per capita, PPP (current international $)	8,939.396	9,775	10,921.013
USA	United States	SE.COM.DURS	Compulsory education, duration (years)	12	12	12

Now in transforms tab it also gave us an option where we can get a better result by using group by and aggregations operations for any desired column we want.



Series_code	y_2019.avg
GB.XPD.RSDV.GD.ZS	2.705360055
SE.COM.DURS	10.5
SH.STA.BRTC.ZS	99
SH.XPD.CHEX.PP.CD	5900.6019921
SH.XPD.EHEX.CH.ZS	0.00066961
SP.DYN.LE00.IN	77.84990244
SP.POP.SCIE.RD.P6	3146.2407225



Country_name	Series_code	y_2019.max
China	GB.XPD.RSDV.GD.ZS	2.244630098
China	SE.COM.DURS	9
China	SH.XPD.CHEX.PP.CD	880.1912842
China	SH.XPD.EHEX.CH.ZS	0.00066961
China	SP.DYN.LE00.IN	76.912
China	SP.POP.SCIE.RD.P6	1471.253906
United States	GB.XPD.RSDV.GD.ZS	3.166090012
United States	SE.COM.DURS	12
United States	SH.STA.BRTC.ZS	99
United States	SH.XPD.CHEX.PP.CD	10921.0127
United States	SP.DYN.LE00.IN	78.78780488
United States	SP.POP.SCIE.RD.P6	4821.227539

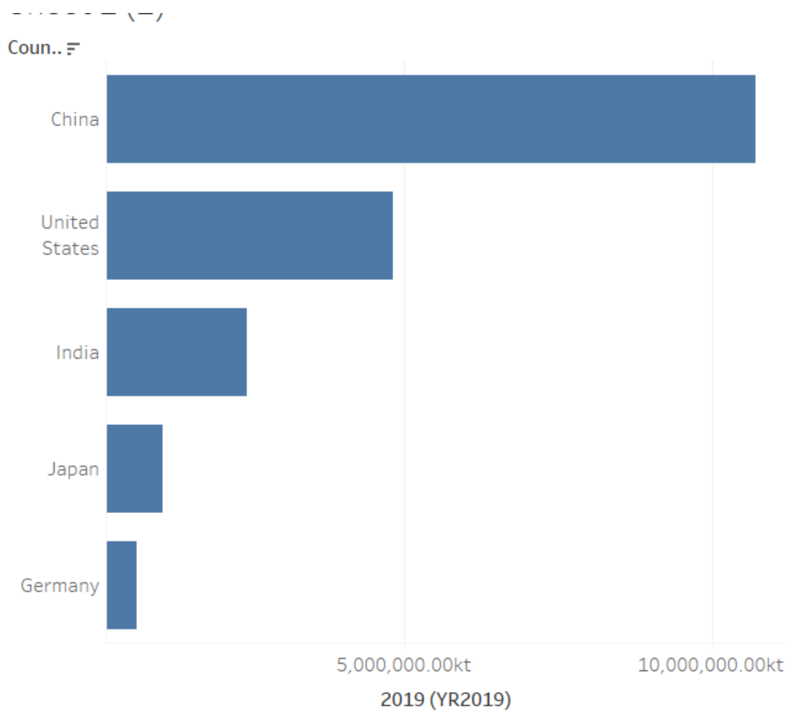
4. Insights

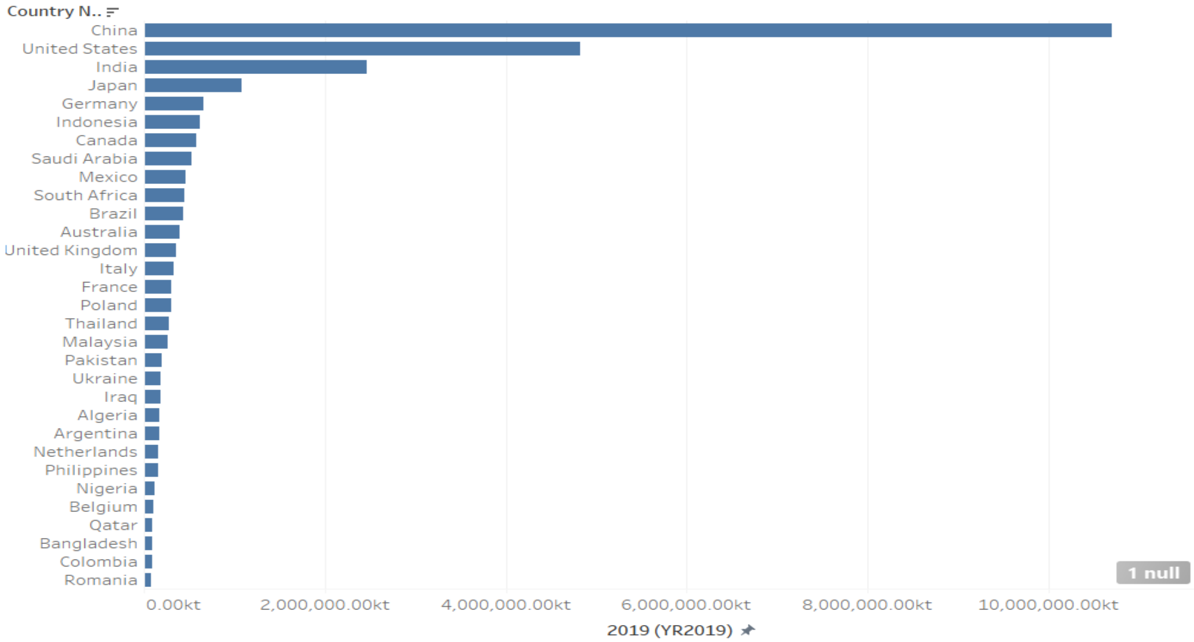
Phase 1: After cleaning and analyzing each of the variables and performing exploratory analysis we have found two insights from phase 1 are as follows:

- **There is a steep raise in co2 emission from the year 1990 to 2021**
- **Among all countries the top countries with highest emission are USA and China.**

The visualisations using **Tableau** are below:

Co2 emission for the countries for the year 2019

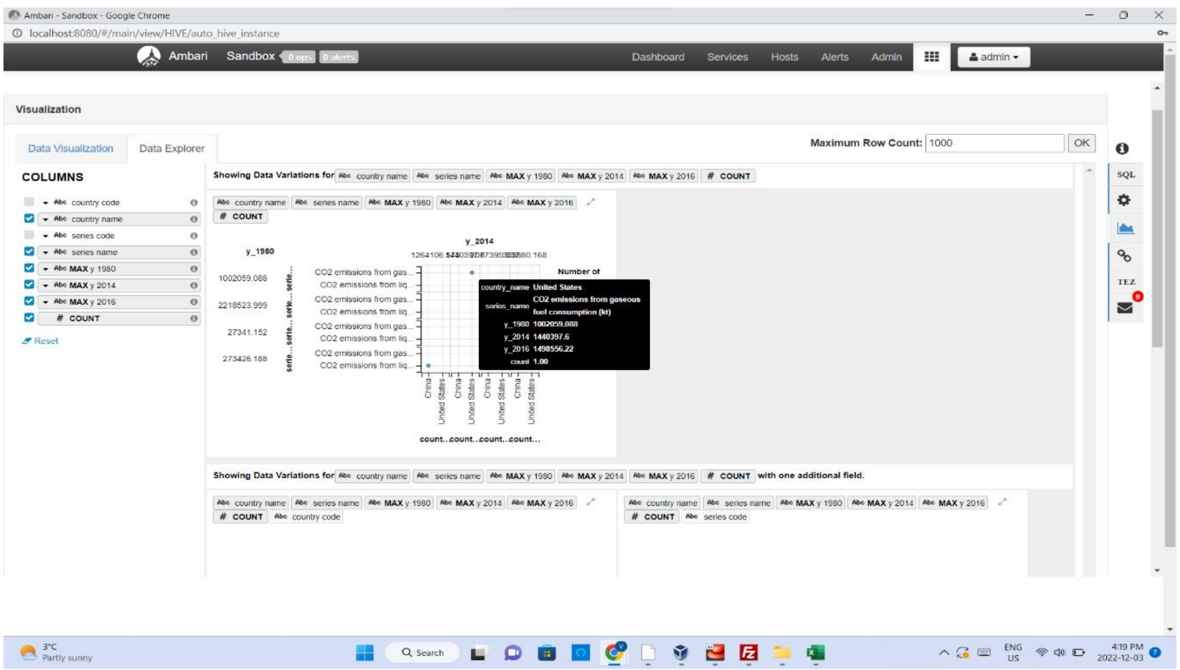




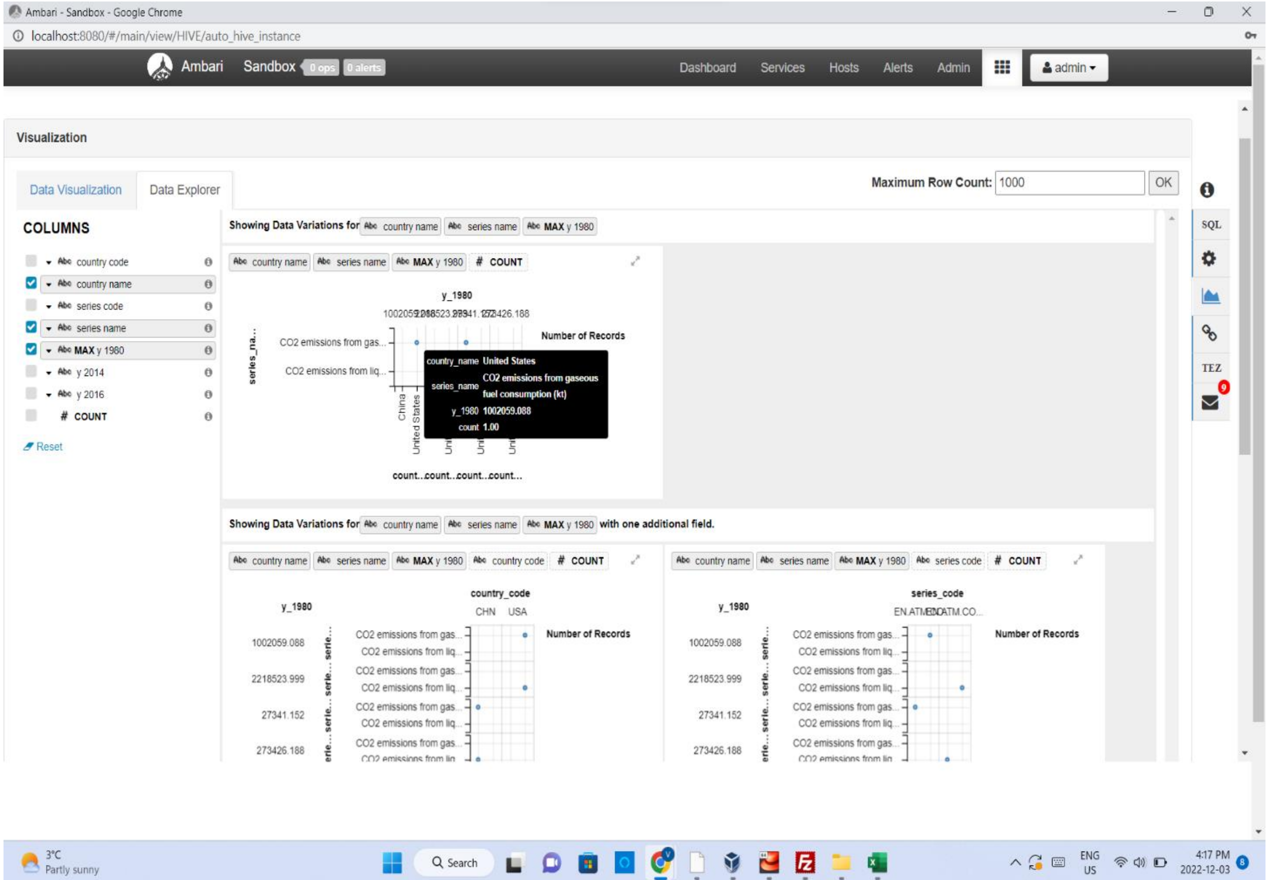
Phase 2: INSIGHTS FROM THIS PHASE 2 ARE AS FOLLOWS:

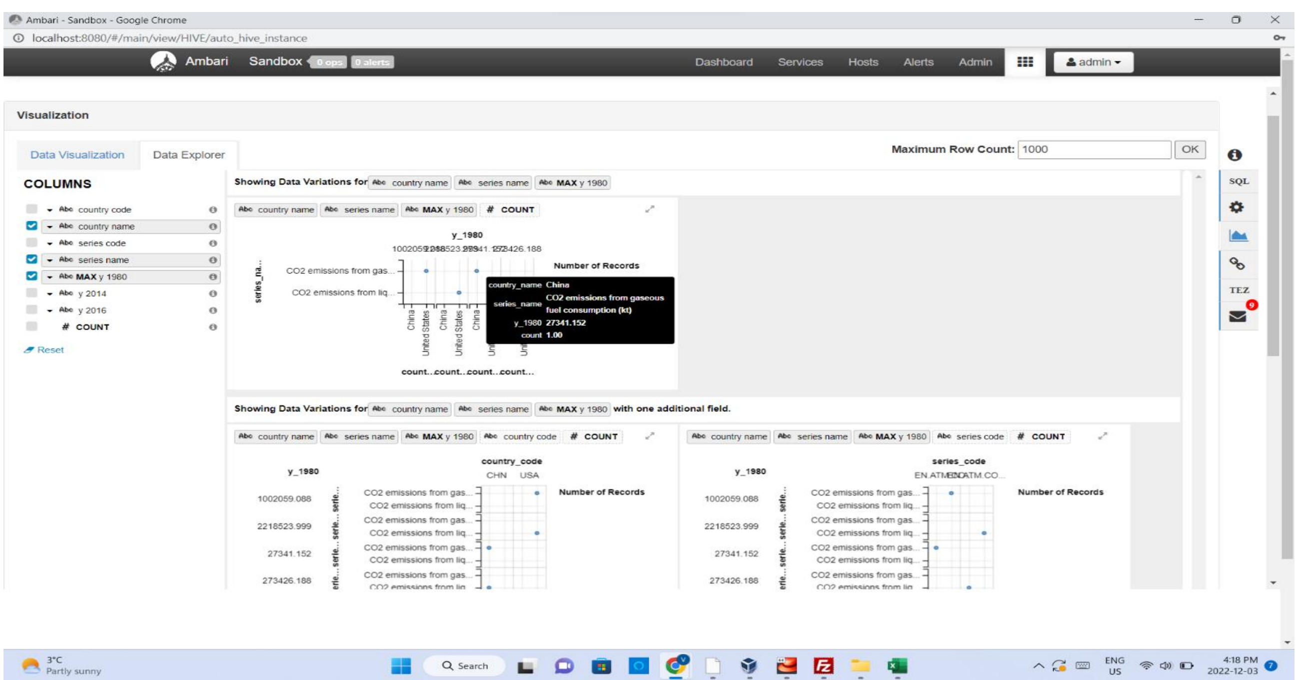
- **Among all sources, gaseous source is most responsible**

DATA SHOWING THE FACTORS WITH THE MAX CO2 EMISSION OUT OF LIQUID, GASEOUS AND SOLID FOR THE YEARS_1980,2014 AND 2016



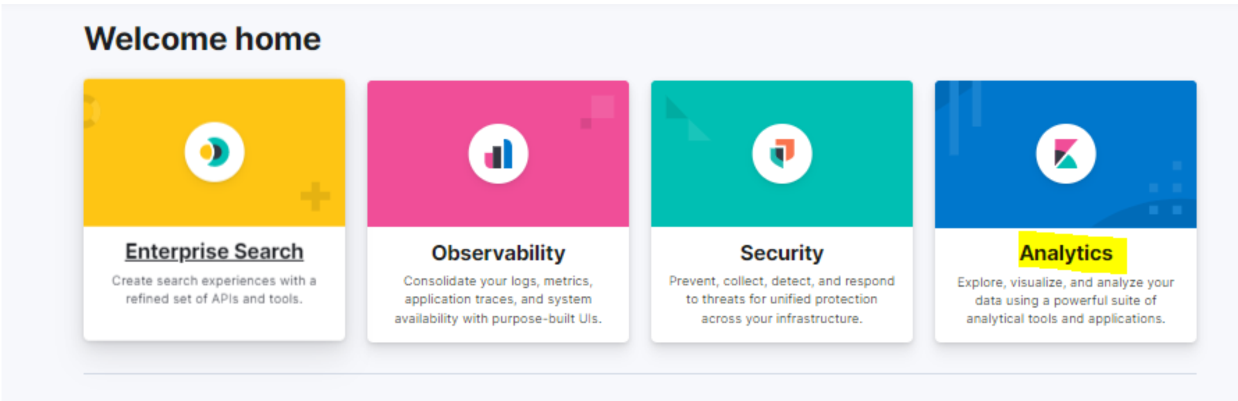
Data showing the gaseous factor with the max Co2 emission for the year_1980 in China and USA



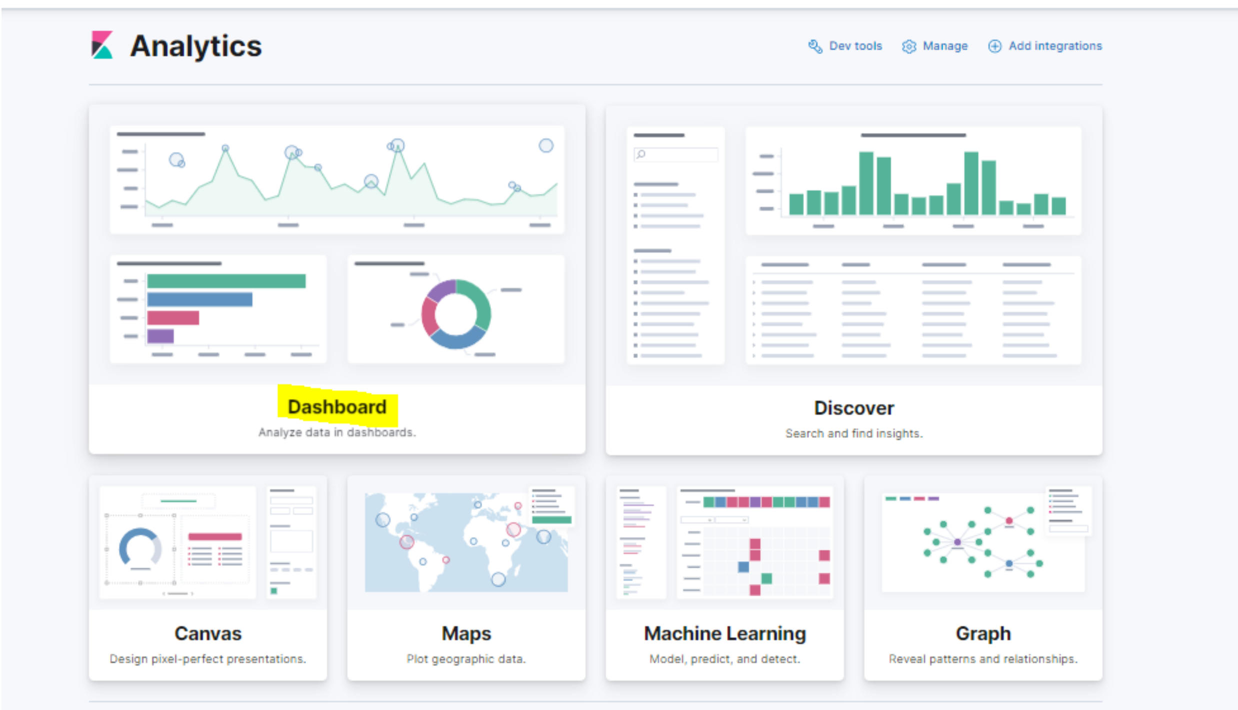


Phase 3: In continuation of phase 1 and phase 2, here we are using the already cleaned data file. For phase 3, we have used elastic search tool.

Kibana is the the most effective elastic search interface for discovering data insights and performing active management of the health of the Elastic Stack. As we can see, it gives an option for Analytics. We clicked on that.



Inbuilt options for the visualization like dashboards, maps, graphs:



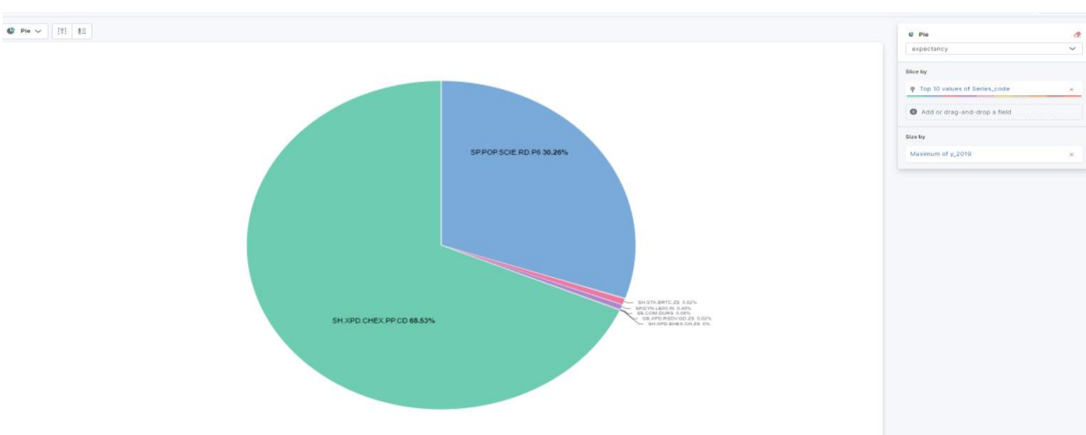
This is the bar graph we generated using inbuild Kibana dashboards that depicts us the below result.

Result: SH.XPD.CHEX.PP.CD (Current health expenditure per capita, PPP, international $) for US is the most dominating factor which is directly related to improving the life expectancy. While the second most responsible factor to improve the life expectancy is SP.POP.SCIE. RD. P6 (Researchers in R&D, per million people)

y_2019, series_code and country_name columns being used in this.

This is another visualization below using pie chart for the same result that we saw above regarding the contribution of the factors for improving the life expectancy.

y_2019 and series_code columns being used



5. Future Work

- We will be focusing on the data for the years 2020 and 2022 and using more big data tools and cloud computing tools as well.
- We will be considering the topmost countries only which are USA and China in order to focus on the most relevant factors to reduce the greenhouse effect.
- Apart from the Co2 emission, we will be working on other factors which are directly or indirectly affecting the world and we would like to consider the pandemic phase.

6. References

- *https://climate.nasa.gov/global-warming-vs-climate-change/*
- *https://www.encyclopedia.com/environment/energy-government-and-defense-magazines/carbon-dioxide-co2-emissions*
- *https://www.climate.gov/news-features/understanding-climate/climate-change-atmospheric-carbon-dioxide*
- *https://medium.com/bazaar-tech/apache-spark-data-cleaning-using-pyspark-for-beginners-eeeced351ebf*
- *https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0262802#:~:text=Carbon%20emissions%20have%20a%20significantly,decreases%20life%20expectancy%20by%200.012%25*
- *blogs.sap.com*
- *Lecture_10 & Lab 8_Elastic_Search*
- *https://www.elastic.co/elasticsearch*
- *https://severalnines.com/blog/what-is-elasticsearch-and-why-use-it/*
- *https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0262802*