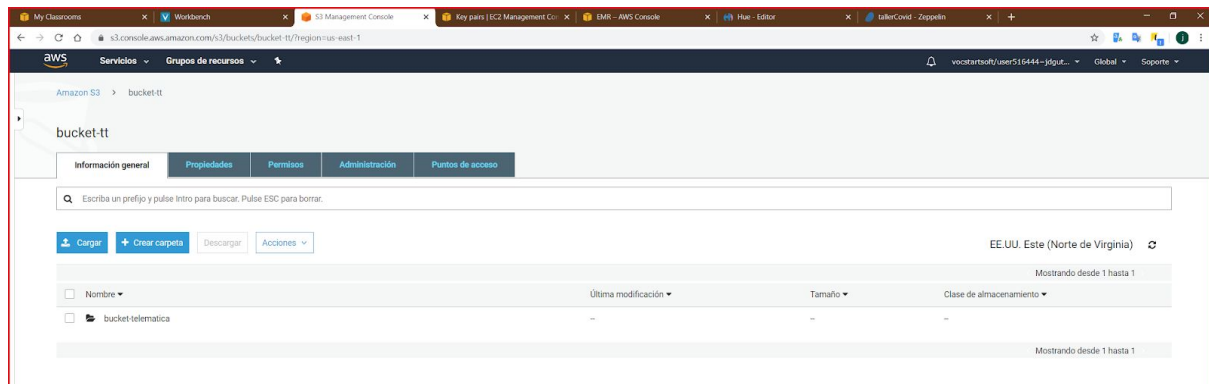


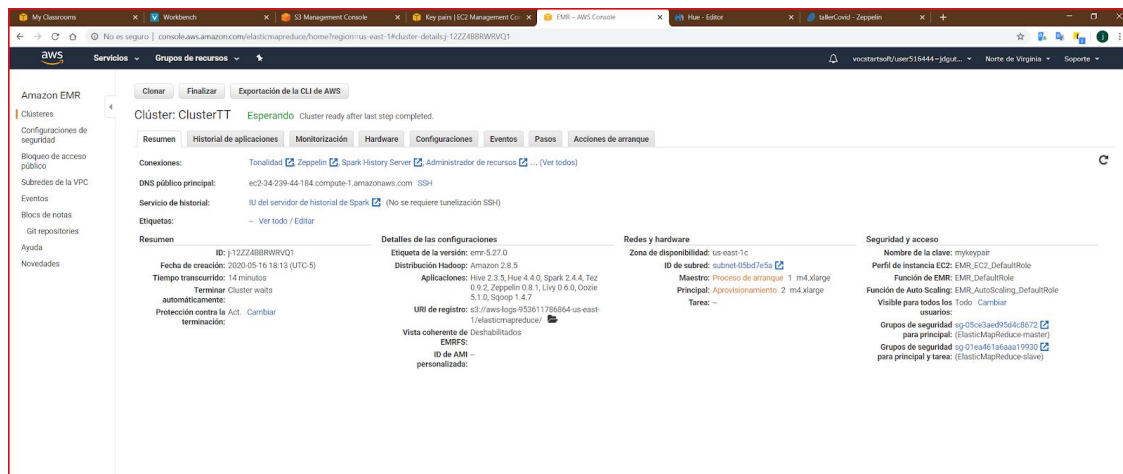
Proyecto BIGDATA Covid-19

Integrantes: Juan Diego Gutiérrez Montoya, Carla Daniela Rendón Baliero.

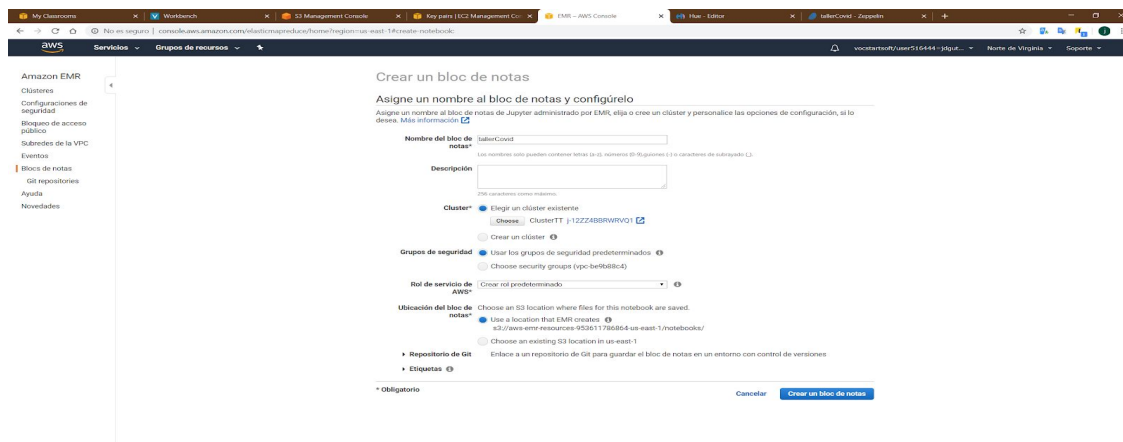
Para la realización del proyecto, es necesaria la creación previa de un Bucket en AMAZON S3, para efectos del proyecto este Bucket es denominado Bucket tt, y crear una carpeta(Bucket telemática) que contendrá los datos proporcionados.



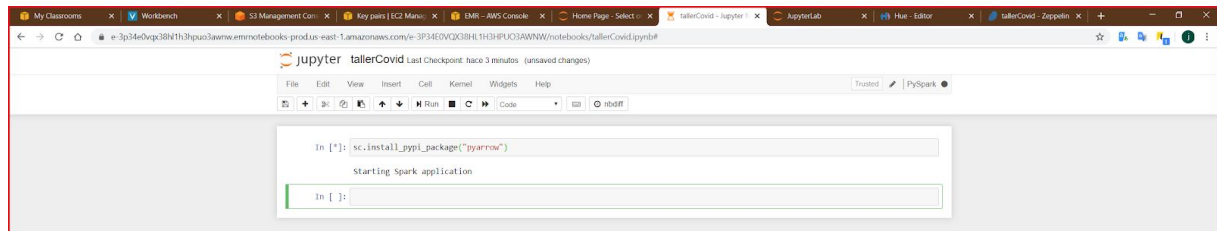
Luego se procederá a la creación de un clúster(Clúster tt), para ello es necesario tener un par de claves creadas con anterioridad.



Dentro de Amazon EMR, procedemos a la creación de un Block de notas(TallerCovid)



Luego, Ingresamos al block de notas, y dentro de jupyter seleccionamos el kernel de pyspark, y ejecutamos el siguiente comando para instalar el paquete pyarrow:

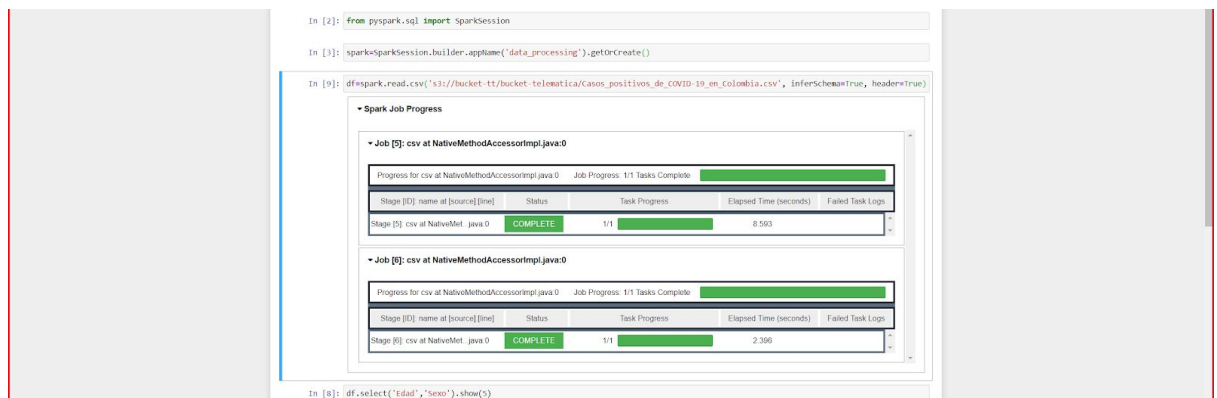


```
In [*]: %pip install pyarrow
```

Starting Spark application

```
In [ ]:
```

Ejecutamos los siguientes comandos para importar los datos proporcionados del covid-19.



```
In [2]: from pyspark.sql import SparkSession
```

```
In [3]: spark=SparkSession.builder.appName('data_processing').getOrCreate()
```

```
In [9]: df=spark.read.csv('s3://bucket-telemtica/casos_positivos_de_COVID-19_en_colombia.csv', inferSchema=True, header=True)
```

Spark Job Progress

Job [5]: csv at NativeMethodAccessorImpl.java:0

Stage [0]: name at [source] [line]	Status	Task Progress	Elapsed Time (seconds)	Failed Task Logs
Stage [5]: csv at NativeMet... java:0	COMPLETE	1/1	8.583	

Job [6]: csv at NativeMethodAccessorImpl.java:0

Stage [0]: name at [source] [line]	Status	Task Progress	Elapsed Time (seconds)	Failed Task Logs
Stage [6]: csv at NativeMet... java:0	COMPLETE	1/1	2.396	

```
In [8]: df.select('edad','sexo').show(5)
```

Para iniciar con el análisis de los datos realizamos un select donde se muestren los primeros 5 valores de los datos correspondientes a las columnas de Edad y Sexo:

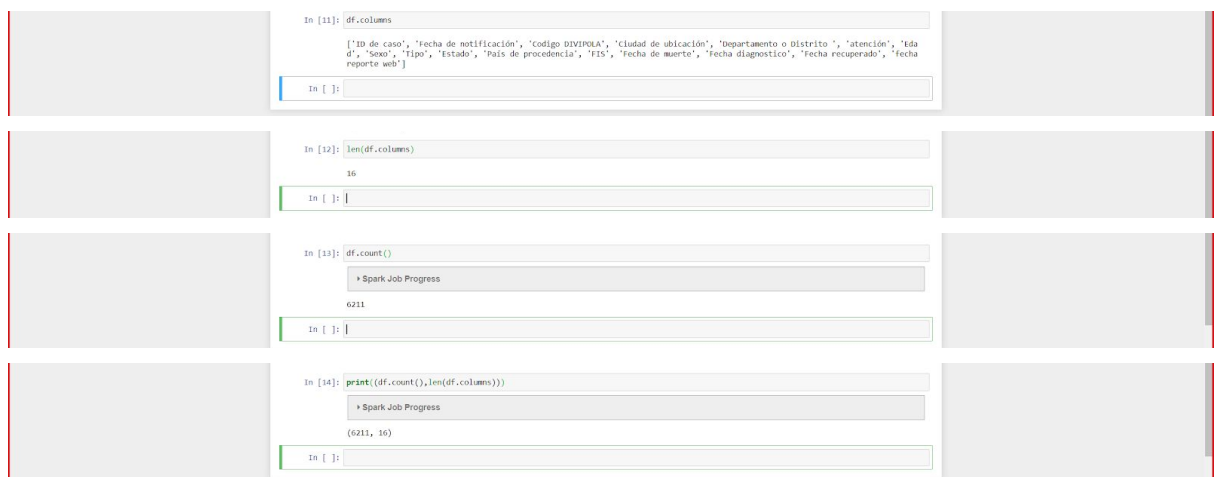


```
In [10]: df.select('edad','sexo').show(5)
```

Spark Job Progress

```
-----+-----+
|edad|sexo|
-----+-----+
| 19|  F|
| 34|  M|
| 50|  F|
| 55|  M|
| 25|  M|
-----+-----+
only showing top 5 rows
```

Luego ejecutamos los siguientes comandos para visualizar las columnas, longitud y el count del schema.



```
In [11]: df.columns
```

```
[ 'ID de caso', 'fecha de notificación', 'codigo DIVUPILA', 'ciudad de ubicación', 'Departamento o Distrito ', 'atención', 'Ida d', 'sexo', 'Tipo', 'Estado', 'Pais de procedencia', 'FIS', 'fecha de muerte', 'fecha diagnostico', 'fecha recuperado', 'fecha reporte web' ]
```

```
In [ ]:
```

```
In [12]: len(df.columns)
```

```
16
```

```
In [ ]:
```

```
In [13]: df.count()
```

Spark Job Progress

```
6211
```

```
In [ ]:
```

```
In [14]: print(df.count(),len(df.columns))
```

Spark Job Progress

```
(6211, 16)
```

```
In [ ]:
```

Con el siguiente comando, imprimimos las columnas del schema, observando el tipo de dato y los valores por defecto de cada una.

```
In [15]: df.printSchema()

root
  |-- id: de cosa: integer (nullable = true)
  |-- fecha de notificación: timestamp (nullable = true)
  |-- Codigo DIVIPOLA: integer (nullable = true)
  |-- Ciudad de ubicación: string (nullable = true)
  |-- departamento o distrito: string (nullable = true)
  |-- atención: string (nullable = true)
  |-- edad: integer (nullable = true)
  |-- Sexo: string (nullable = true)
  |-- Tipo: string (nullable = true)
  |-- Estado: string (nullable = true)
  |-- País de procedencia: string (nullable = true)
  |-- FITS: string (nullable = true)
  |-- fecha de muerte: string (nullable = true)
  |-- fecha diagnostico: timestamp (nullable = true)
  |-- fecha recuperación: string (nullable = true)
  |-- fecha reporte web: timestamp (nullable = true)
```

Con los siguientes comandos mostramos las 5 primeras filas del schema y las describimos.

```
In [16]: df.show(5)
```

```

+ Spark Job Progress
+-----+
ID de caso|fecha de notificación|codigo DIVIPOLA|ciudad de ubicación|departamento o distrito | atención|edad|sexo| Tipo
[Estado|País de procedencia] |Fis|fecha de muerte| Fecha diagnóstico| Fecha recuperado| fecha reporte web|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| level | | | | | | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 2020-03-01 00:00:00 | 11001 | ... | Bogotá D.C. | Recuperado | 19 | F | Importado
| level | | | | | | | | | |
| 1 | ... | Italia | 2020-02-27 00:00:00 | ... | ... | 2020-03-06 00:00:00 | 2020-03-13 00:00:00 | ... | 2020-03-06 00:00:00 |
| level | | | | | | | | |
| 1 | ... | España | 2020-03-04 00:00:00 | 76111 | Guadalupe de Buga | Valle del Cauca | Recuperado | 34 | M | Importado
| level | | | | | | | | |
| 1 | ... | España | 2020-03-04 00:00:00 | 5001 | Medellín | Antioquia | Recuperado | 50 | F | Importado
| level | | | | | | | | |
| 1 | ... | España | 2020-02-29 00:00:00 | 5001 | Medellín | Antioquia | Recuperado | 50 | F | Importado
| level | | | | | | | | |
| 1 | ... | Colombia | 2020-03-06 00:00:00 | 5001 | Medellín | Antioquia | Recuperado | 25 | M | Reporte local
| level | | | | | | | | |
| 1 | ... | Colombia | 2020-03-06 00:00:00 | 5001 | Medellín | Antioquia | Recuperado | 25 | M | Reporte local
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

```
In [18]: df.describe().show()
```

Spark Job Progress

	[summary]	ID de caso	Código DIVOPILA	Ciudad de ubicación	Departamento o Distrito	latencia	Edad	Sexo
tipo	Estado	procedencia	FIS	Fecha de muerte	fecha recuperado			
count	6211	6211	6211	6211	6211	6211	6211	6211
mean	6211	3106.413924	-691515653935	null	null	5952	41.23751987	6.945
stddev	null	1795.1055927263922	27326.15632510684	null	null	null	18.66899156740696	null
min	null	1	Alemania	2020-02-27	Acacias	-	Amazonas	casa
max	fallecido	6211	91081	Zona Bananera	Valle del Cauca	casa	103	mrs
lacionado	leve	Venezuela	Asintomatico	2020-04-29	2010-08-29	...	2020-04-27	23:59:...

Luego, ejecutamos el siguiente comando para importar los tipos de datos de SQL que requerimos para almacenar las consultas:

```
In [19]: from pyspark.sql.types import StringType, DoubleType, IntegerType
```

Con este comando, procedemos a crear una nueva columna dentro del schema, donde almacenamos las personas que son mayores de 10 años:

```
In [20]: df.withColumn("age_after_10_yrs", (df["edad"]>10)).show(10,False)
```

ID de caso	fecha de notificación	codigo DIVIPOLA	ciudad de ubicación	departamento o distrito	atención	edad	sexo	tipo
[estado]	[País de procedencia]	[FIS]	[fecha de muerte]	[fecha diagnostico]	[fecha recuperado]	[fecha reporte web]		
11	2020-03-02 00:00:00	11001	Bogotá D.C.	Bogotá D.C.	[recuperado]	19	F	Importado
6	2020-02-27 00:00:00	-	-	-	2020-03-06 00:00:00	2020-03-13 00:00:00	2020-03-0	
12	2020-03-06 00:00:00	76111	Quindío	Valle del Cauca	[recuperado]	14	M	Importado
11	2020-03-07 00:00:00	5001	Medellín	Antioquia	[recuperado]	50	F	Importado
9	2020-03-07 00:00:00	5001	Medellín	Antioquia	[recuperado]	50	F	Importado
14	2020-03-09 00:00:00	5001	Medellín	Antioquia	[recuperado]	155	M	Relacionado
1	2020-03-09 00:00:00	5001	Medellín	Antioquia	[recuperado]	25	M	Relacionado
15	2020-03-10 00:00:00	5360	Itagüí	Antioquia	[recuperado]	27	F	Relacionado
17	2020-03-11 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	85	F	Importado
12	2020-03-11 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	22	F	Importado
19	2020-03-11 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	28	F	Importado
10	2020-03-12 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	36	F	Importado
2	2020-03-12 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	36	F	Importado

Con este comando, procedemos a filtrar los datos por sexo Femenino.

```
In [21]: df.filter((df["Sexo"]=="F"))[(df["Sexo"]=="F")).show()
```

ID de caso	fecha de notificación	codigo DIVIPOLA	ciudad de ubicación	departamento o distrito	atención	edad	sexo	tipo
[estado]	[País de procedencia]	[FIS]	[fecha de muerte]	[fecha diagnostico]	[fecha recuperado]	[fecha reporte web]		
11	2020-03-02 00:00:00	11001	Bogotá D.C.	Bogotá D.C.	[recuperado]	19	F	Importado
6	2020-02-27 00:00:00	5001	Medellín	Antioquia	[recuperado]	50	F	Importado
11	2020-03-07 00:00:00	5001	Medellín	Antioquia	[recuperado]	50	F	Importado
14	2020-03-09 00:00:00	5001	Medellín	Antioquia	[recuperado]	155	M	Relacionado
1	2020-03-09 00:00:00	5001	Medellín	Antioquia	[recuperado]	25	M	Relacionado
15	2020-03-10 00:00:00	5360	Itagüí	Antioquia	[recuperado]	27	F	Relacionado
17	2020-03-11 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	85	F	Importado
12	2020-03-11 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	22	F	Importado
19	2020-03-11 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	28	F	Importado
10	2020-03-12 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	36	F	Importado
2	2020-03-12 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	36	F	Importado
13	2020-03-13 00:00:00	50001	Medellín	Antioquia	[recuperado]	26	F	Relacionado
16	2020-03-13 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	30	F	Relacionado
17	2020-03-13 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	30	F	Relacionado
19	2020-03-13 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	30	F	Relacionado
20	2020-03-13 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	30	F	Relacionado
24	2020-03-12 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	30	F	Relacionado
26	2020-03-15 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	30	F	Relacionado
27	2020-03-15 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	30	F	Relacionado
29	2020-03-12 00:00:00	13001	Cartagena de Indias	Cartagena D.T. y C.	[recuperado]	30	F	Relacionado
30	2020-03-13 00:00:00	54001	Medellín	Antioquia	[recuperado]	55	F	Relacionado
31	2020-03-13 00:00:00	54001	Medellín	Antioquia	[recuperado]	48	F	Importado

Luego agrupamos por edad y mostramos los primeros 5 resultados:

```
In [24]: df.groupby("edad").count().show(5,False)
```

edad	count
131	161
165	17
165	12
153	12
178	21

Definimos la ruta donde se guardará la consulta, y procedemos a almacenarla con formato csv, para ello ejecutamos los siguientes comandos:

```
In [25]: write_uri='s3://bucket-tt/bucket-telematica/results/df_csv'

In [26]: df.select('edad','sexo').coalesce(1).write.format("csv").option("header","true").save(write_uri)
```

Spark Job Progress

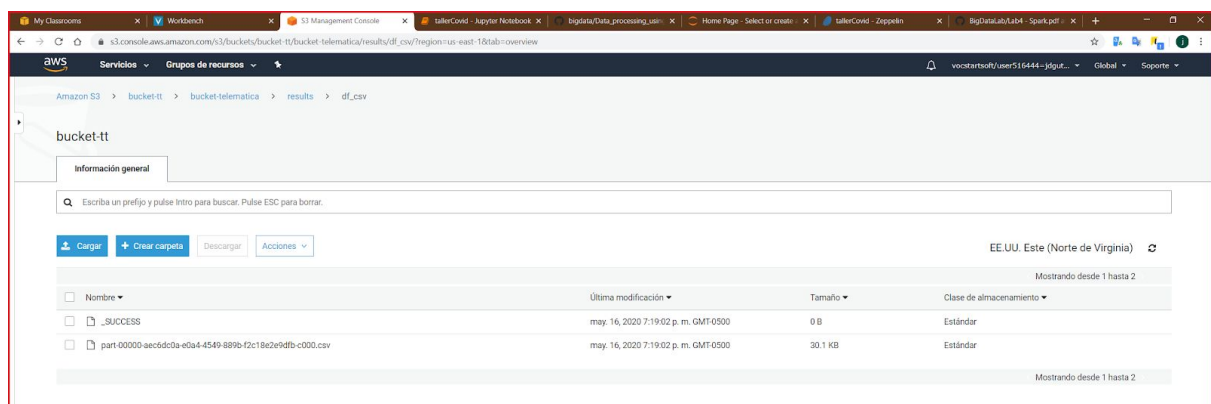
Job [17]: save at NativeMethodAccessorImpl.java:0

Progress for save at NativeMethodAccessorImpl.java:0 Job Progress: 1/1 Tasks Complete

Stage [ID]	name at [source] [line]	Status	Task Progress	Elapsed Time (seconds)	Failed Task Logs
Stage [23]	save at NativeMethodAccessorImpl.java:0	COMPLETE	1/1	9.468	

```
In [ ]:
```

Procedemos a comprobar que los datos se almacenaron correctamente en la ruta especificada.



Definimos la ruta donde se guardará la consulta, y procedemos a almacenarla con formato parquet, para ello ejecutamos los siguientes comandos:

```
In [27]: parquet_uri='s3://bucket-tt/bucket-telematica/results/df_parquet'

In [28]: df.select('edad','sexo').write.format("parquet").save(parquet_uri)
```

Spark Job Progress

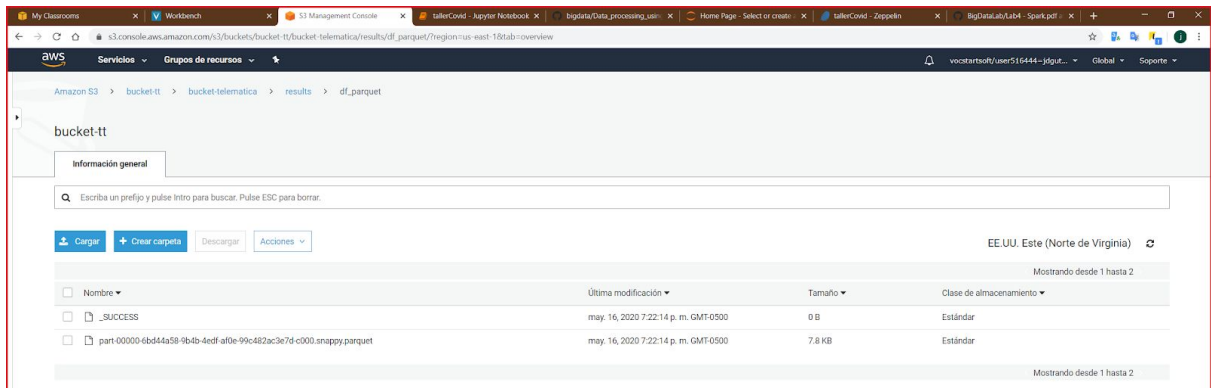
Job [18]: save at NativeMethodAccessorImpl.java:0

Progress for save at NativeMethodAccessorImpl.java:0 Job Progress: 1/1 Tasks Complete

Stage [ID]	name at [source] [line]	Status	Task Progress	Elapsed Time (seconds)	Failed Task Logs
Stage [24]	save at NativeMethodAccessorImpl.java:0	COMPLETE	1/1	9.581	

```
In [ ]:
```

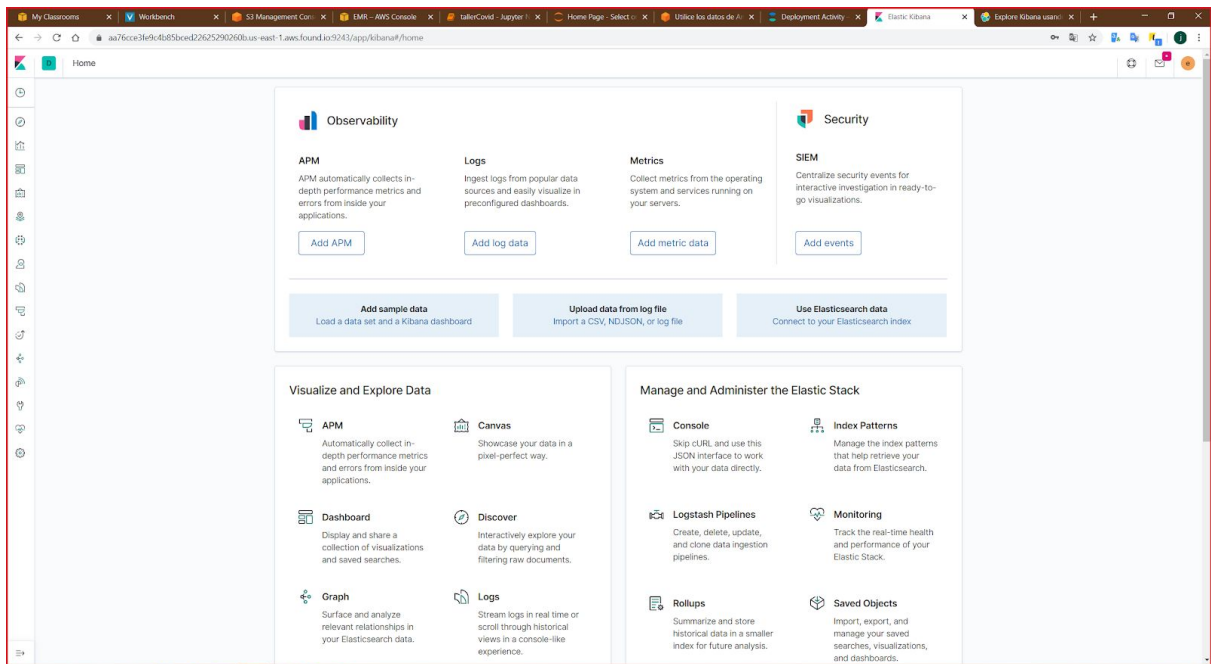
Procedemos a comprobar que los datos se almacenaron correctamente en la ruta especificada.



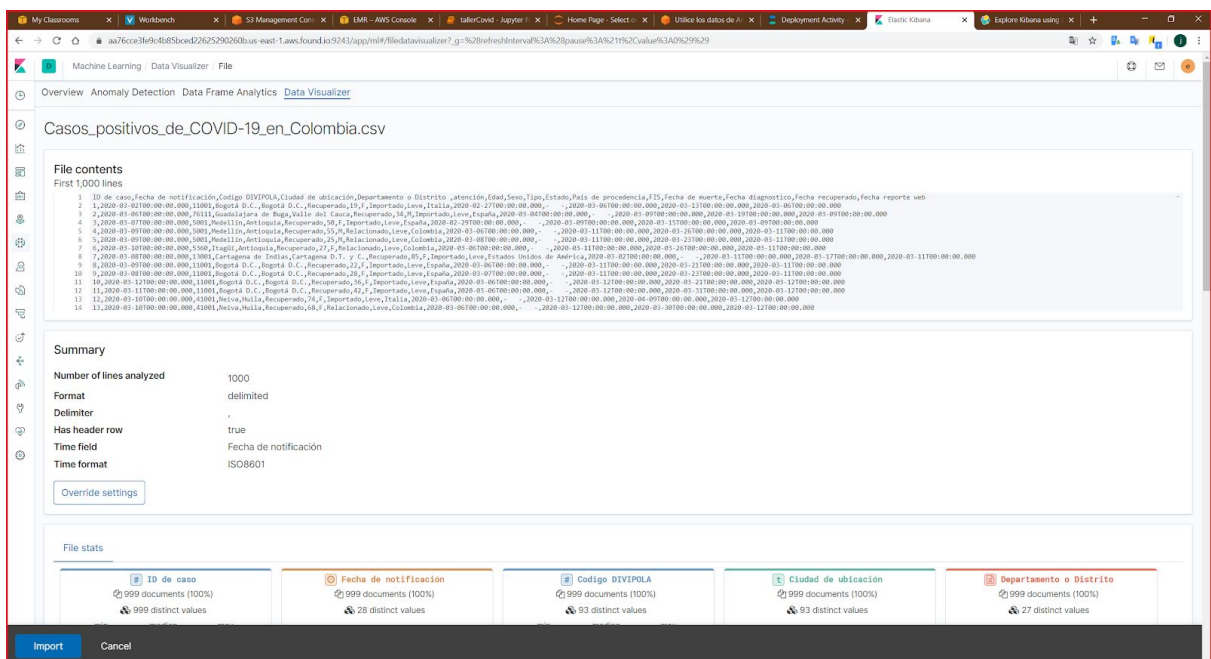
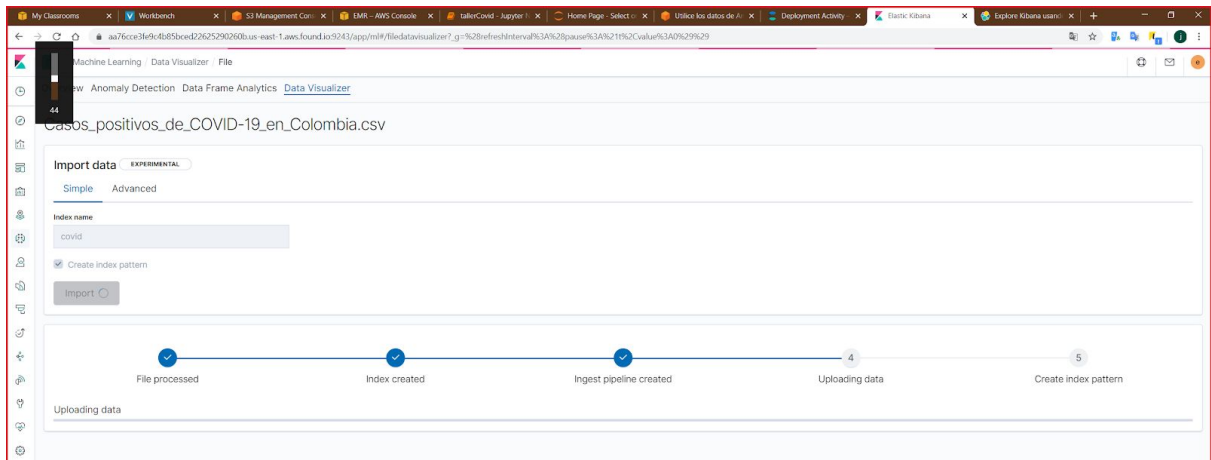
Con esto concluimos con la ingesta, almacenamiento, procesamiento y análisis exploratorio de los datos con pyspark.

Para obtener un entendimiento mayor de la data que estamos almacenando, procederemos a realizar una visualización de la misma mediante Kibana.

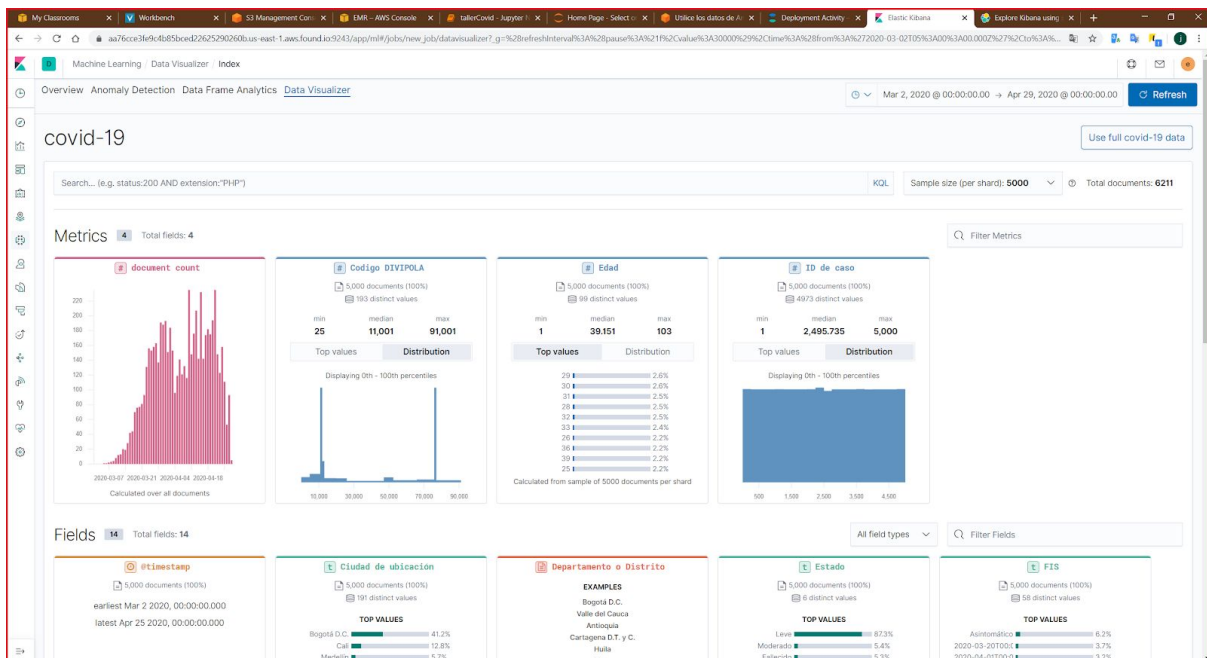
Para ello creamos una cuenta gratuita y accedemos al dashboard, luego seleccionamos la opción “Upload data from log file” para cargar los datasets propuestos:



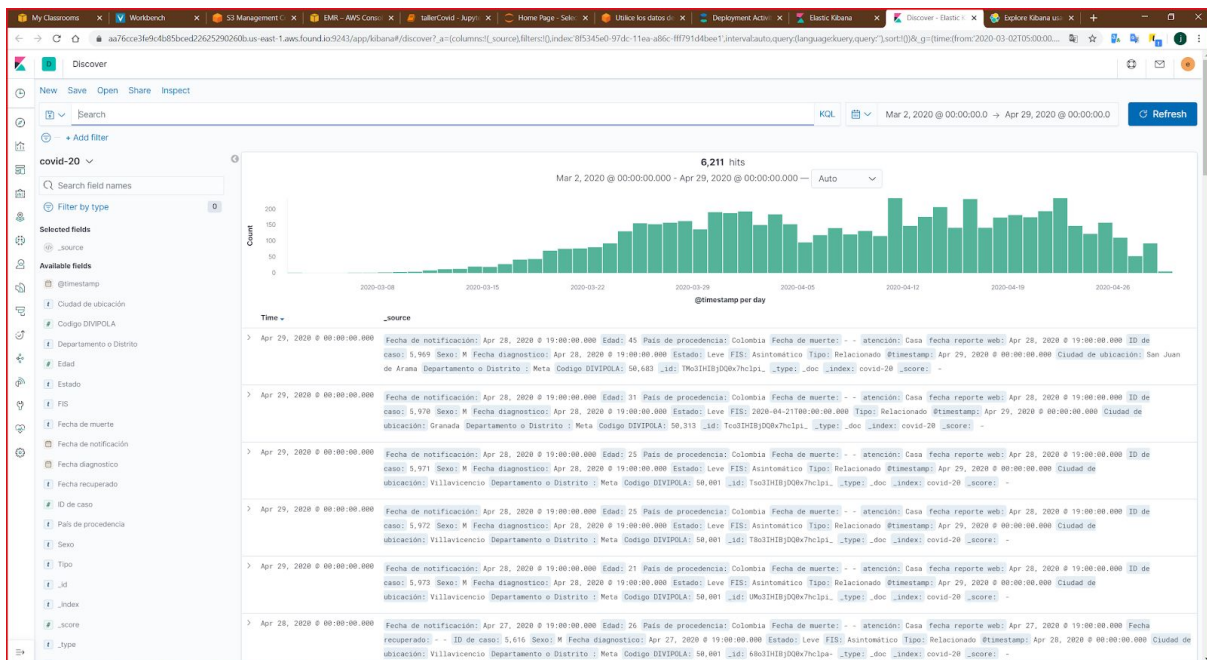
Primero cargamos el dataset que muestra los casos positivos del covid-19 en Colombia y los importamos:



Hacemos un conteo de los campos que posee la tabla y las repeticiones de y contenido, esto se puede visualizar en la sección de Data Visualizer:

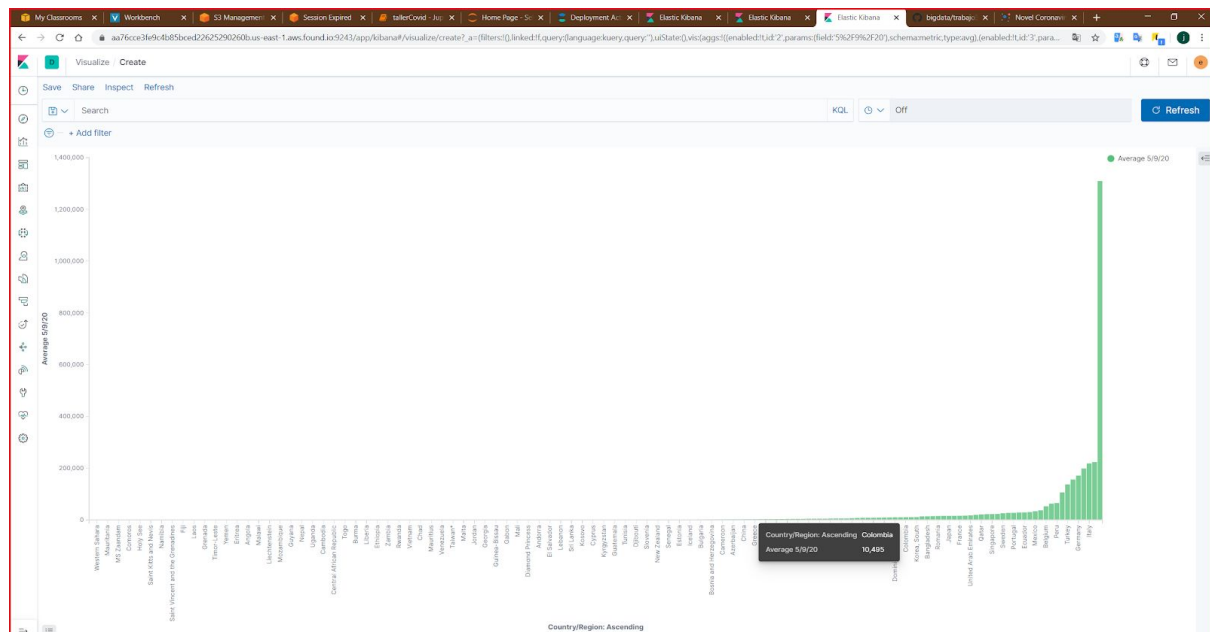


Procedemos a obtener una gráfica de la curva de contagios del Covid-19 en Colombia:



Procedemos a obtener un gráfico que muestra el TOP 10 de las ciudades con más contagios en Colombia y sus cifras correspondientes:

En la Siguiente gráfica podemos observar la posición de Colombia con respecto al mundo, por total de contagios de Covid-19, podemos precisar que se encuentra en el puesto número 20:



Con esto finalizamos con el análisis y la visualización de los datos proporcionados sobre el covid-19.