

UNIVERSIDAD EAFIT

Tópicos Especiales En Telemática

Laboratorio 3-2 HIVE

Miguel Angel Hoyos Velasquez

Medellín, 18 de Noviembre 2024

Aquí se aprecia la ingesta de datos de manera manual, sobre el mismo cluster.

The screenshot shows the Hue interface with the Hive editor open. The SQL query defines a table 'hdi' with columns: id, country, hdi, l1feex, mysch, eysch, and gni. The data is loaded from a local file 'hdi.txt' located at '/user/hadoop/datasets/ou/hdi.txt'. The query is executed, and the results are displayed in a table with 14 rows.

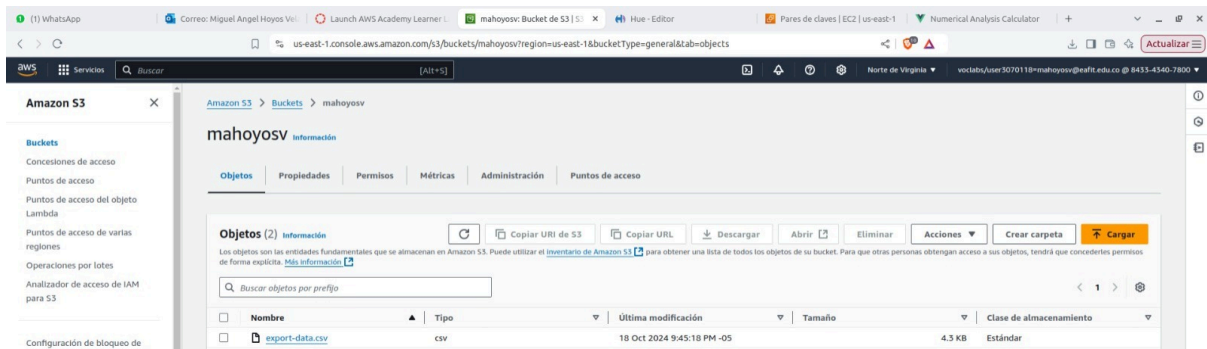
hdi.id	hdi.country	hdi.hdi	hdi.l1feex	hdi.mysch	hdi.eysch	hdi.gni	
1	NULL	NULL	NULL	NULL	NULL	NULL	
2	1	Norway	0.943	81	12	17	47557
3	2	Australia	0.929	81	12	18	34431
4	3	Netherlands	0.91	80	11	16	36402
5	4	United States	0.91	78	12	16	43017
6	5	New Zealand	0.908	80	12	18	23737
7	6	Canada	0.908	81	12	16	35166
8	7	Ireland	0.908	80	11	18	29322
9	8	Liechtenstein	0.905	79	10	14	83717
10	9	Germany	0.905	80	12	15	34854
11	10	Sweden	0.904	81	11	15	35837
12	11	Switzerland	0.903	82	11	15	39924
13	12	Japan	0.901	83	11	15	32295
14	13	Hong Kong China (SAR)	0.898	82	10	15	44865

Posteriormente la definición de la tabla. Esta definirá la estructura de los datos con los que se trabajarán. Se hizo una consulta para validar el correcto funcionamiento.

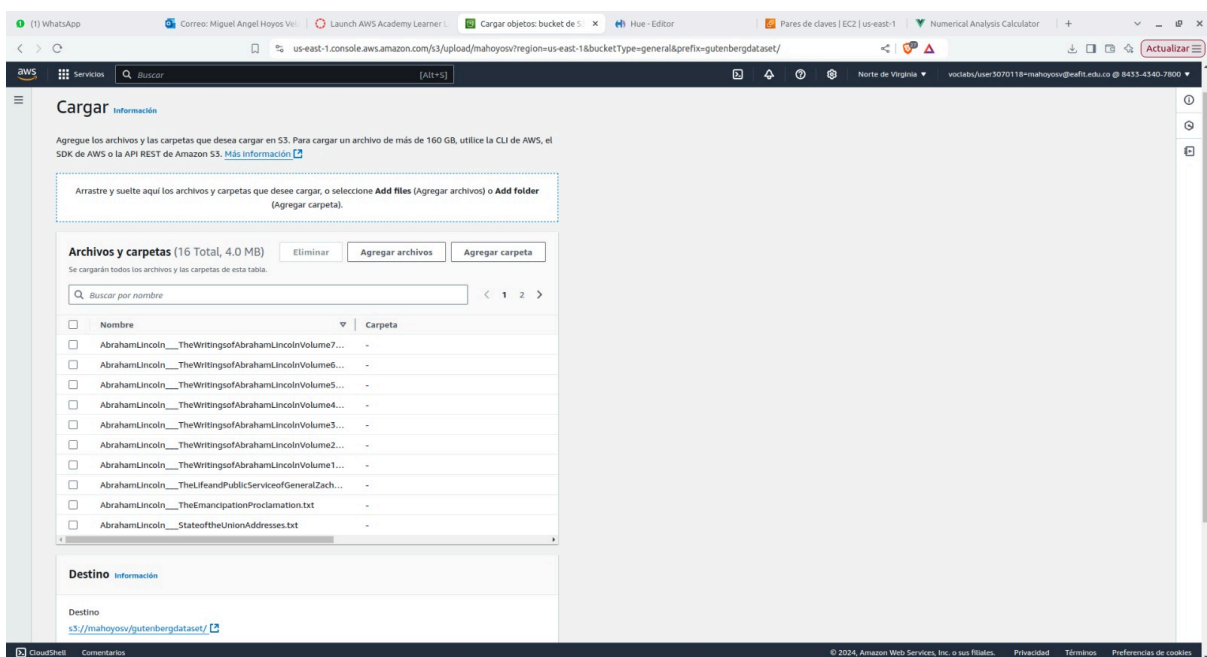
The screenshot shows the Hue interface with the Hive editor open. The SQL query defines a table 'expo' with columns: country, expct, and gni. The data is loaded from S3 using the 's3://mahoyosv' location. The query is executed, and the results are displayed in a table with 16 rows.

h.country	gni	expct	
1	Albania	7803	29.77231
2	Algeria	7658	30.830406
3	Andorra	36095	NULL
4	Angola	4874	56.835884
5	Antigua and Barbuda	15521	44.08267
6	Argentina	14527	21.706469
7	Armenia	5188	20.58361
8	Australia	34431	19.780243
9	Austria	35719	53.971355
10	Azerbaijan	8666	55.125755
11	Bahrain	28169	NULL
12	Barbados	17966	47.34396
13	Belarus	13439	54.822608
14	Belgium	33357	80.00875
15	Belize	5812	NULL
16	Bhutan	5293	NULL

Ahora vemos una consulta, pero en este caso cargando los datos directamente desde S3



Posteriormente se trabajó con el dataset de gutenber small, para realizar el wordcounter.



Aquí se aprecia la carga de los archivos en S3, para su posterior utilización.

The screenshot shows the Hue Editor interface with a SQL query editor on the left and a results pane on the right. The query editor contains the following SQL code:

```

14 use usernamedb;
15 CREATE EXTERNAL TABLE EXPO (country STRING, expct FLOAT)
16 ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
17 STORED AS TEXTFILE
18 LOCATION 's3://mahoyss/gutenbergdataset/';
19
20 SELECT h.country, gni, expct FROM hdi h JOIN EXPO e ON (h.country = e.country) WHERE gni > 2000;
21
22
23
24
25 CREATE EXTERNAL TABLE docs (l1ne STRING)
26 STORED AS TEXTFILE
27 LOCATION 's3://mahoyss/gutenbergdataset/';
28
29 // ordenado por palabra
30
31 SELECT word, count(1) AS count FROM (SELECT explode(split(l1ne, ' ')) AS word FROM docs) w
32 GROUP BY word
33 ORDER BY word DESC LIMIT 10;

```

The results pane shows the output of the second query, displaying a list of words and their counts, ordered by count in descending order:

word	count
1	1
2	1
3	1
4	1
5	1
6	1
7	5
8	3
9	8
10	2

Creación de la tabla y ordenada por palabra.

Adicionalmente, también se trabajó desde jupyterhub con EMR.

The screenshot shows the JupyterLab interface with a Spark session running. The session ID is 4, and the application ID is application_1731543739540_0007. The session is available as 'spark'.

The results pane shows the output of the following SQL query:

```
spark.sql("SELECT * FROM hdi").show()
```

The output displays a table with columns: id, country, hdi, lifeexp, mysch, eysch, gni. The table contains 19 rows of data, showing the top 10 rows of the dataset.

The results pane also shows the output of the following SQL query:

```
spark.sql("select country, gni from hdi where gni > 2000")
```

The output displays a DataFrame with columns: country, gni. The DataFrame contains 10 rows of data, showing the top 10 rows of the dataset where gni is greater than 2000.

The screenshot shows a JupyterLab notebook interface with a browser window at the top displaying the URL `https://ec2-54-80-125-232.compute-1.amazonaws.com:443/user/joyan/notebooks/Untitled.ipynb`. The notebook itself has a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and code execution. The main area contains a code cell with the following content:

```
9 | Germany|0.905| 89| 12| 15|34854|
10| Sweden|0.904| 81| 11| 15|35837|
11| Switzerland|0.903| 82| 11| 15|39924|
12| Japan|0.901| 83| 11| 15|32295|
13| Hong Kong China (...)|0.898| 82| 10| 15|44885|
14| Iceland|0.898| 81| 10| 18|29354|
15| Korea (Republic of)|0.897| 89| 11| 16|28238|
16| Denmark|0.895| 78| 11| 16|34347|
17| Israel|0.888| 81| 11| 15|25849|
18| Belgium|0.886| 89| 10| 16|33357|
19| Austria|0.885| 89| 10| 15|35719|
+-----+
only showing top 20 rows

In [9]: spark.sql("select country, gni from hdi where gni > 2000").show()
```

The output of the query is displayed below the code cell, showing a table with columns `country` and `gni`. The output is truncated with `only showing top 20 rows`.

Data filtrada con una query simple.

The screenshot shows a JupyterLab notebook interface with a browser window at the top displaying the URL `https://ec2-54-80-125-232.compute-1.amazonaws.com:443/user/joyan/notebooks/Untitled.ipynb`. The notebook has a menu bar and a toolbar. The main area contains a code cell with the following content:

```
at org.apache.spark.sql.Dataset.take(Dataset.scala:3583)
at org.apache.spark.sql.Dataset.prRow(Dataset.scala:293)
at org.apache.spark.sql.Dataset.showString(Dataset.scala:318)
at org.apache.spark.sql.Dataset.show(Dataset.scala:883)
at org.apache.spark.sql.Dataset.show(Dataset.scala:842)
at org.apache.spark.sql.Dataset.show(Dataset.scala:851)
... $! elided

In [14]: spark.conf.set("mapreduce.input.fileinputformat.input.dir.recursive", "true")

In [15]: spark.sql("SELECT word, count(1) AS count FROM (SELECT explode(split(line, ' ')) AS word FROM docs) w GROUP BY word ORDER BY count DESC LIMIT 10").show()
```

The output of the query is displayed below the code cell, showing a table with columns `word` and `count`. The output is truncated with `only showing top 20 rows`.

WordCounter. Adicionalmente se adjunta el notebook utilizado, por si se desea replicar.