Practica Final

June 11, 2022

1 Practica 3. Cargar y manipular datos

Rodríguez Fitta José Emanuel

1.1 Guardar archivos en HDFS

[19]:
```
!wget  https://raw.githubusercontent.com/omarmendoza564/datos/main/datos/
↪flights-larger.csv
```

```
--2022-06-11 21:10:55--
https://raw.githubusercontent.com/omarmendoza564/datos/main/datos/flights-
larger.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.109.133, 185.199.110.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 9660247 (9.2M) [text/plain]
Saving to: ‘flights-larger.csv.2’

flights-larger.csv. 100%[===================>]   9.21M  --.-KB/s    in 0.09s

2022-06-11 21:10:55 (106 MB/s) - ‘flights-larger.csv.2’ saved [9660247/9660247]
```

Verificamos que los datos se hayan descargados

[23]:
```
!ls
```

```
bin         dev                 hadoop  lib32   lost+found  opt   run   sys  var
boot        etc                 home    lib64   media       proc  sbin  tmp
copyright   flights-larger.csv  lib     libx32  mnt         root  srv   usr
```

Creamos una carpeta para el trabajo que se realizará

[41]:
```
!hdfs dfs -mkdir /tmp/dcd/practica_final
```

Verificamos que la carpeta se haya creado

[46]:
```
!hdfs dfs -ls /tmp/dcd/
```

```
Found 13 items
drwxr-xr-x   - jerf8010 hadoop          0 2022-06-04 17:28 /tmp/dcd/OnTimeDB
drwxr-xr-x   - jerf8010 hadoop          0 2022-06-04 01:01 /tmp/dcd/particion
drwxr-xr-x   - jerf8010 hadoop          0 2022-06-03 21:32 /tmp/dcd/persist
drwxr-xr-x   - root     hadoop          0 2022-06-11 21:22 /tmp/dcd/practica_3
drwxr-xr-x   - root     hadoop          0 2022-06-11 21:24
/tmp/dcd/practica_final
drwxr-xr-x   - jerf8010 hadoop          0 2022-06-03 22:05 /tmp/dcd/pyspark
drwxr-xr-x   - jerf8010 hadoop          0 2022-06-04 16:53 /tmp/dcd/sentimientos
drwxr-xr-x   - jerf8010 hadoop          0 2022-06-03 23:55 /tmp/dcd/sirilo
drwxr-xr-x   - jerf8010 hadoop          0 2022-06-04 00:27 /tmp/dcd/sirilo_avro
drwxr-xr-x   - jerf8010 hadoop          0 2022-06-04 16:34 /tmp/dcd/streamdat
drwxr-xr-x   - jerf8010 hadoop          0 2022-06-04 16:37 /tmp/dcd/streamdata
drwxr-xr-x   - jerf8010 hadoop          0 2022-06-10 20:03 /tmp/dcd/wordcount
drwxr-xr-x   - jerf8010 hadoop          0 2022-05-28 16:45 /tmp/dcd/workcount
```

Guardamos el archivo CSV en la carpeta creada

[43]:
```
!hdfs dfs -put flights-larger.csv /tmp/dcd/practica_final
```

Verificamos que se haya guardado

[47]:
```
!hdfs dfs -ls /tmp/dcd/practica_final
```

```
Found 1 items
-rw-r--r--   2 root hadoop    9660247 2022-06-11 21:24
/tmp/dcd/practica_final/flights-larger.csv
```

1.2 Carga del archivo en notebook de jupyter

[48]:
```
df = spark.read.option('encoding', 'UTF-8').csv('hdfs:///tmp/dcd/
↪practica_final', inferSchema = True, header = True)
```

[49]:
```
df.toPandas()
```

[49]:

	month	dayofmonth	dayofweek	carrier	flight	origin	mile	depart	\
0	10	10	1	OO	5836	ORD	157	8.18	
1	1	4	1	OO	5866	ORD	466	15.50	
2	11	22	1	OO	6016	ORD	738	7.17	
3	2	14	5	B6	199	JFK	2248	21.17	
4	5	25	3	WN	1675	SJC	386	12.92	
…	…	…	…	…	…	…	…	…	…
274995	4	31	6	UA	259	ORD	888	16.75	
274996	3	14	1	UA	119	SFO	337	16.20	
274997	10	4	2	AA	716	ORD	1005	7.00	

```
274998      9          27          1      B6     128    JFK   267   22.50
274999      6          15          2      OH    5552    JFK   301    9.08

        duration delay
0             51    27
1            102    NA
2            127   -19
3            365    60
4             85    22
…            …     …
274995       154    46
274996        84    33
274997       155    -6
274998        86   -19
274999       105   -15

[275000 rows x 10 columns]
```

1.3 Lstar del dataframe

1.3.1 Número de registros

[51]:
```
print(f'Hay {df.count()} registros')
```

```
Hay 275000 registros
```

1.3.2 Estructura

[53]:
```
print('La estructura es: ')
df.printSchema()
```

```
La estructura es:
root
 |-- month: integer (nullable = true)
 |-- dayofmonth: integer (nullable = true)
 |-- dayofweek: integer (nullable = true)
 |-- carrier: string (nullable = true)
 |-- flight: integer (nullable = true)
 |-- origin: string (nullable = true)
 |-- mile: integer (nullable = true)
 |-- depart: double (nullable = true)
 |-- duration: integer (nullable = true)
 |-- delay: string (nullable = true)
```

1.3.3 Nombre de las columnas

[54]:
```
print('Se tienen las siguientes columnas: ')
df.columns
```

```
Se tienen las siguientes columnas:
```

[54]:
```
['month',
 'dayofmonth',
 'dayofweek',
 'carrier',
 'flight',
 'origin',
 'mile',
 'depart',
 'duration',
 'delay']
```

1.3.4 Tipo de datos

[55]:
```
print('Los tipos de datos son: ')
df.dtypes
```

```
Los tipos de datos son:
```

[55]:
```
[('month', 'int'),
 ('dayofmonth', 'int'),
 ('dayofweek', 'int'),
 ('carrier', 'string'),
 ('flight', 'int'),
 ('origin', 'string'),
 ('mile', 'int'),
 ('depart', 'double'),
 ('duration', 'int'),
 ('delay', 'string')]
```

1.3.5 Ver los primeros 20 registros

[61]:
```
print('Los primeros 20 registros son: ')
df.toPandas().head(20)
```

```
Los primeros 20 registros son:
```

[61]:

	month	dayofmonth	dayofweek	carrier	flight	origin	mile	depart	\
0	10	10	1	OO	5836	ORD	157	8.18	
1	1	4	1	OO	5866	ORD	466	15.50	
2	11	22	1	OO	6016	ORD	738	7.17	
3	2	14	5	B6	199	JFK	2248	21.17	

4	5	25	3	WN	1675	SJC	386	12.92
5	3	28	1	B6	377	LGA	1076	13.33
6	5	28	6	B6	904	ORD	740	9.58
7	1	19	2	UA	820	SFO	679	12.75
8	8	5	5	US	2175	LGA	214	13.00
9	5	27	5	AA	1240	ORD	1197	14.42
10	8	20	6	B6	119	JFK	1182	14.67
11	2	3	1	AA	1881	JFK	1090	15.92
12	8	26	5	B6	35	JFK	1028	20.58
13	4	9	5	AA	336	ORD	733	20.50
14	3	8	2	UA	678	ORD	733	10.95
15	8	10	3	OH	6347	LGA	292	11.75
16	8	14	0	UA	624	ORD	612	17.92
17	4	8	4	OH	5585	JFK	301	13.25
18	1	14	4	UA	1524	SFO	414	14.87
19	1	2	6	AA	1341	ORD	1846	7.50

	duration	delay
0	51	27
1	102	NA
2	127	-19
3	365	60
4	85	22
5	182	70
6	130	47
7	123	135
8	71	-10
9	195	-11
10	198	20
11	200	-9
12	193	102
13	125	32
14	129	55
15	102	8
16	109	57
17	88	23
18	91	27
19	275	26

1.3.6 Descripción estadística

[63]:
```
print('Descripción estadística: ')
df.describe().toPandas()
```

```
Descripción estadística:
```

[63]:
	summary	month	dayofmonth	dayofweek	carrier	\
0	count	275000	275000	275000	275000	
1	mean	5.24232	15.71406909090909	2.946090909090909	None	
2	stddev	3.4273573316203576	8.805568383848067	1.9635141531217672	None	
3	min	0	1	0	AA	
4	max	11	31	6	WN	

	flight	origin	mile	depart	\
0	275000	275000	275000	275000	
1	2063.0542763636363	None	881.2222872727273	14.124930981817384	
2	2185.852169684581	None	700.5178890821038	4.683189503417866	
3	1	JFK	11	0.12	
4	6941	TUS	4243	23.98	

	duration	delay
0	275000	275000
1	151.64103636363637	28.34773064280709
2	87.0845640768675	54.01489538326629
3	14	-1
4	605	NA

1.3.7 Descripcion estadistica de una sola columna (delay)

[64]:
```
print('Descripción estadística de la columna delay: ')
df.describe(['delay']).show()
```

```
Descripción estadística de la columna delay:
[Stage 19:============================>                            (1 + 1) / 2]
+-------+-----------------+
|summary|            delay|
+-------+-----------------+
|  count|           275000|
|   mean|28.34773064280709|
| stddev|54.01489538326629|
|    min|               -1|
|    max|               NA|
+-------+-----------------+
```

1.3.8 Realizar un agrupamiento

[84]:
```
print('Agrupamiento: ')
df.groupBy('month').mean().sort('month', ascending=[0]).
 ↪toPandas()['avg(duration)']
```

Agrupamiento:

[84]:
```
0     151.900523
1     149.824350
2     147.295899
3     147.758903
4     152.470290
5     152.742928
6     154.781627
7     153.745812
8     153.272359
9     153.456543
10    150.326092
11    149.848733
Name: avg(duration), dtype: float64
```

1.3.9 Mostrar la filas ordenas por un campo

[86]:
```
print('Filas ordenadas por campo mile: ')
df.orderBy('mile', ascending = [0]).toPandas()
```

Filas ordenadas por campo mile:

[86]:

	month	dayofmonth	dayofweek	carrier	flight	origin	mile	depart	\
0	1	12	2	AA	73	ORD	4243	9.08	
1	2	20	4	UA	1	ORD	4243	10.40	
2	6	22	2	UA	1	ORD	4243	10.65	
3	1	12	2	UA	1	ORD	4243	10.42	
4	4	13	2	AA	73	ORD	4243	9.00	
…	…	…	…	…	…	…	…	…	
274995	10	3	1	OO	5829	ORD	67	17.13	
274996	0	18	5	OO	5826	ORD	67	10.50	
274997	11	31	3	OO	5613	SFO	30	18.00	
274998	4	15	4	OH	4988	JFK	11	18.00	
274999	7	10	0	OH	5572	JFK	11	12.33	

	duration	delay
0	560	11
1	556	21
2	531	8
3	556	10
4	549	-34
…	…	…
274995	47	-7
274996	48	-17
274997	20	NA
274998	60	145

```
274999        60    55

[275000 rows x 10 columns]
```

1.3.10 Generar una consulta SQL desde el DataFrame

[92]:
```
print('Consulta SQL')

df.select('origin', 'duration') \
    .where(df.month == 1) \
    .groupBy('origin').mean() \
    .orderBy('avg(duration)', ascending = 0).toPandas()
```

```
Consulta SQL
```

[92]:

	origin	avg(duration)
0	JFK	214.545632
1	LGA	158.788958
2	ORD	144.134250
3	SFO	144.084703
4	OGG	124.095153
5	TUS	113.829137
6	SJC	108.579057
7	SMF	104.648060

1.3.11 Generar un agrupamiento que muestre funciones de agregacion, minimo tres (sum, max,min, avg)

[94]:
```
from pyspark.sql.functions import col, count, avg, max
```

[125]:
```
print('Agrupamiento con funciones de agregación')


df.groupBy('month', 'origin') \
    .agg(sum('mile').alias("sum_miles"), \
         avg('duration').alias("avg_duration"), \
         count('origin').alias('total_flights'), \
         max('duration').alias("max_duration") \
     ) \
    .filter(col('avg_duration') > 120)  \
    .sort('max_duration') \
    .show(truncate=False)
```

```
Agrupamiento con funciones de agregación
+-----+------+---------+------------------+-------------+------------+
|month|origin|sum_miles|avg_duration      |total_flights|max_duration|
+-----+------+---------+------------------+-------------+------------+
```

```
|4    |LGA   |1919452  |154.77645121449157|2429         |269         |
|6    |LGA   |1968995  |149.22982791586998|2615         |277         |
|5    |LGA   |2085030  |151.86861584011842|2702         |277         |
|8    |LGA   |1414391  |146.89665653495442|1974         |280         |
|7    |LGA   |1911058  |150.52865731462927|2495         |280         |
|11   |LGA   |1577149  |156.00901328273244|2108         |280         |
|10   |LGA   |1174443  |153.4825          |1600         |281         |
|9    |LGA   |1288911  |150.44146202170188|1751         |282         |
|3    |LGA   |2017268  |154.86006957866255|2587         |290         |
|1    |LGA   |1973118  |158.7889584964761 |2554         |290         |
|0    |LGA   |1799748  |159.4808766652342 |2327         |295         |
|11   |SFO   |4039946  |142.8131844064827 |4566         |345         |
|10   |SFO   |3515078  |146.1183047321893 |3846         |346         |
|9    |SFO   |3590421  |147.12682926829268|3895         |346         |
|1    |SFO   |4016697  |144.08470319634702|4380         |348         |
|0    |SFO   |4793136  |143.03595080416272|5285         |348         |
|4    |SFO   |4172776  |150.5621696161802 |4351         |349         |
|3    |SFO   |3881192  |152.93754740834387|3955         |349         |
|6    |SFO   |4750390  |156.45192909280502|4795         |354         |
|8    |SFO   |3706229  |149.5272117616714 |3877         |354         |
+-----+------+---------+------------------+-------------+------------+
only showing top 20 rows
```

1.3.12 Guardar el resultado en una tabla de hive

[126]:
```
df.write.partitionBy('duration').mode("overwrite").saveAsTable('table_hive')
```

1.3.13 Listar las tablas de la base de datos

[127]:
```
!hive -e "show tables;"
```

```
Hive Session ID = 922507d2-c7e4-4abb-9f0e-ffcdc4cd0f93

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-
log4j2.properties Async: true
Hive Session ID = f4344d51-9a96-4e25-8b28-daaf1c70431c
OK
covid_avro
covid_avro_s4s1
covid_parquet
covid_particion
duration_hive
duration_hive_2
sirilo
sirilo_avro
```

```
table_hive
trade_hive
Time taken: 1.655 seconds, Fetched: 10 row(s)
```

[128]:
```
spark.sql('show tables').show()
```

```
+--------+--------------+-----------+
|database|     tableName|isTemporary|
+--------+--------------+-----------+
| default|    covid_avro|      false|
| default|covid_avro_s4s1|      false|
| default| covid_parquet|      false|
| default|covid_particion|      false|
| default| duration_hive|      false|
| default|duration_hive_2|      false|
| default|        sirilo|      false|
| default|   sirilo_avro|      false|
| default|    table_hive|      false|
| default|    trade_hive|      false|
+--------+--------------+-----------+
```

1.3.14 Mostrar el esquema de la nueva tabla

[130]:
```
spark.sql('select * from table_hive').printSchema()
```

```
root
 |-- month: integer (nullable = true)
 |-- dayofmonth: integer (nullable = true)
 |-- dayofweek: integer (nullable = true)
 |-- carrier: string (nullable = true)
 |-- flight: integer (nullable = true)
 |-- origin: string (nullable = true)
 |-- mile: integer (nullable = true)
 |-- depart: double (nullable = true)
 |-- delay: string (nullable = true)
 |-- duration: integer (nullable = true)
```

[ ]:
```

```