Wikipedia_Lab

November 25, 2024

[1]:
```
!scala -version
```

```
Scala code runner version 2.12.10 -- Copyright 2002-2019, LAMP/EPFL and
Lightbend, Inc.
```

1 INITIALIZING SESSION

SparkSession.builder: Used to create a new Spark session. appName(): Sets a name for the application (useful for identifying Spark jobs in logs/UI). config(): Configures the session, specifying the location of the BigQuery connector jar. getOrCreate(): Creates a new session or retrieves an existing one.

[2]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
  .appName('1.1. BigQuery Storage & Spark DataFrames - Python')\
  .config('spark.jars', 'gs://spark-lib/bigquery/spark-bigquery-latest_2.12.
↪jar') \
  .getOrCreate()
```

1.1 Enable repl.eagerEval

This will output the results of DataFrames in each step without the new need to show df.show() and also improves the formatting of the output

[3]:
```
spark.conf.set("spark.sql.repl.eagerEval.enabled",True)
```

1.1.1 Reading Data from BigQuery and Displaying the Schema

The following code reads data from a BigQuery table, applies a filter to limit the data, and displays the schema of the resulting DataFrame.

[4]:
```
table = "bigquery-public-data.wikipedia.pageviews_2020"
df_wiki_pageviews = spark.read \
  .format("bigquery") \
  .option("table", table) \
  .option("filter", "datehour >= '2020-03-01' AND datehour < '2020-03-02'") \
  .load()
```

```
df_wiki_pageviews.printSchema()
```

```
root
 |-- datehour: timestamp (nullable = true)
 |-- wiki: string (nullable = true)
 |-- title: string (nullable = true)
 |-- views: long (nullable = true)
```

1.1.2 Filtering and Caching Wikipedia Pageviews Data

filtering the Wikipedia pageviews data to include only records with more than 1000 views and English language wikis, selects specific columns, and caches the resulting DataFrame.

[5]:
```
df_wiki_en = df_wiki_pageviews \
  .select("title", "wiki", "views") \
  .where("views > 1000 AND wiki in ('en', 'en.m')") \
  .cache()

df_wiki_en
```

[5]:
```
+--------------------+----+------+
|               title|wiki| views|
+--------------------+----+------+
|                   -|  en|143159|
|                   -|  en| 14969|
|                   -|  en|186802|
|                   -|  en|131686|
|                   -|  en|213787|
|                   -|  en|211910|
|                   -|  en|186675|
|                   -|  en| 21901|
|                   -|  en|163710|
|                   -|  en| 23527|
|                   -|  en|202621|
|                   -|  en|110524|
|                   -|  en|220543|
|12_Angry_Men_(195…|  en|  1124|
|                   -|  en|195339|
|                   -|  en|151283|
|                   -|  en| 22490|
|                   -|  en|182985|
|                   -|  en| 45182|
|                   -|  en|153327|
+--------------------+----+------+
only showing top 20 rows
```

1.1.3 Aggregating and Ordering Total Pageviews

Grouping the filtered Wikipedia pageviews data by `title`, calculates the total views for each title, and orders the result by the total views in descending order.

[6]:
```
import pyspark.sql.functions as F

df_wiki_en_totals = df_wiki_en \
.groupBy("title") \
.agg(F.sum('views').alias('total_views'))

df_wiki_en_totals.orderBy('total_views', ascending=False)
```

[6]:
```
+--------------------+-----------+
|               title|total_views|
+--------------------+-----------+
|           Main_Page|   10939337|
|United_States_Senate|    5619797|
|                   -|    3852360|
|      Special:Search|    1538334|
|2019-20_coronavir…|     407042|
|2020_Democratic_P…|     260093|
|         Coronavirus|     254861|
|The_Invisible_Man…|     233718|
|       Super_Tuesday|     201077|
|         Colin_McRae|     200219|
|         David_Byrne|     189989|
|2019-20_coronavir…|     156803|
|        John_Mulaney|     155605|
|2020_South_Caroli…|     152137|
|      AEW_Revolution|     140503|
|       Boris_Johnson|     120957|
|          Tom_Steyer|     120926|
|Dyatlov_Pass_inci…|     117704|
|         Spanish_flu|     108335|
|2020_coronavirus_…|     107653|
+--------------------+-----------+
only showing top 20 rows
```

1.1.4 Writing Data to BigQuery from Spark

Writing the Spark Dataframe to BigQuery table using BigQuery Storage connector. This will also create the table if it does not exist. The GCS bucket and BigQuery dataset must already exist.

[7]:
```
gcs_bucket = 'lab_amali_4'

bq_dataset = 'wiki'
```

```
bq_table = 'wiki_total_pageviews'

df_wiki_en_totals.write \
  .format("bigquery") \
  .option("table","{}.{}".format(bq_dataset, bq_table)) \
  .option("temporaryGcsBucket", gcs_bucket) \
  .mode('overwrite') \
  .save()
```

[8]:
```
df_wiki_pageviews.createOrReplaceTempView("wiki_pageviews")
```

1.1.5 Querying a Temporary View with Spark SQL

Using Spark SQL to query the `wiki_pageviews` temporary view and filter the results based on specific conditions:

[9]:
```
df_wiki_en = spark.sql("""
SELECT
 title, wiki, views
FROM wiki_pageviews
WHERE views > 1000 AND wiki in ('en', 'en.m')
""").cache()

df_wiki_en
```

[9]:
```
+--------------------+----+------+
|               title|wiki| views|
+--------------------+----+------+
|                   -|  en|143159|
|                   -|  en| 14969|
|                   -|  en|186802|
|                   -|  en|131686|
|                   -|  en|213787|
|                   -|  en|211910|
|                   -|  en|186675|
|                   -|  en| 21901|
|                   -|  en|163710|
|                   -|  en| 23527|
|                   -|  en|202621|
|                   -|  en|110524|
|                   -|  en|220543|
|12_Angry_Men_(195…|  en|  1124|
|                   -|  en|195339|
|                   -|  en|151283|
|                   -|  en| 22490|
|                   -|  en|182985|
|                   -|  en| 45182|
```

```
|                   -|  en|153327|
+--------------------+----+------+
only showing top 20 rows
```

[10]:
```
df_wiki_en.createOrReplaceTempView("wiki_en")
```

1.1.6 Running an SQL Query on the Temporary View

Query to group the `views` by `title` from the `wiki_en` temporary view and orders the results by the total views in descending order

[11]:
```
df_wiki_en_totals = spark.sql("""
SELECT
 title,
 SUM(views) as total_views
FROM wiki_en
GROUP BY title
ORDER BY total_views DESC
""")

df_wiki_en_totals
```

[11]:
```
+--------------------+-----------+
|               title|total_views|
+--------------------+-----------+
|           Main_Page|   10939337|
|United_States_Senate|    5619797|
|                   -|    3852360|
|      Special:Search|    1538334|
|2019-20_coronavir…|     407042|
|2020_Democratic_P…|     260093|
|         Coronavirus|     254861|
|The_Invisible_Man…|     233718|
|       Super_Tuesday|     201077|
|         Colin_McRae|     200219|
|         David_Byrne|     189989|
|2019-20_coronavir…|     156803|
|        John_Mulaney|     155605|
|2020_South_Caroli…|     152137|
|      AEW_Revolution|     140503|
|       Boris_Johnson|     120957|
|          Tom_Steyer|     120926|
|Dyatlov_Pass_inci…|     117704|
|         Spanish_flu|     108335|
|2020_coronavirus_…|     107653|
+--------------------+-----------+
only showing top 20 rows
```

[13]:
```
# Update to your GCS bucket
gcs_bucket = 'lab_amali_4'

# Update to your BigQuery dataset name you created
bq_dataset = 'wiki'

# Enter BigQuery table name you want to create or overwite.
# If the table does not exist it will be created when you run the write function
bq_table = 'wiki_total_pageviews'

df_wiki_en_totals.write \
  .format("bigquery") \
  .option("table","{}.{}".format(bq_dataset, bq_table)) \
  .option("temporaryGcsBucket", gcs_bucket) \
  .mode('overwrite') \
  .save()
```

1.1.7 Filtering the Data for Views Greater Than 1000

Filtering the `df_wiki_pageviews` DataFrame to include only the rows where `views` are greater than 1000 and the `wiki` is either 'en' or 'en.m'. It also selects the columns `datehour`, `wiki`, and `views`, and caches the DataFrame for optimization:

[15]:
```
df_wiki_en = df_wiki_pageviews \
  .select("datehour", "wiki", "views") \
  .where("views > 1000 AND wiki in ('en', 'en.m')") \
  .cache()

df_wiki_en
```

[15]:
```
+-------------------+----+------+
|           datehour|wiki| views|
+-------------------+----+------+
|2020-03-01 16:00:00|  en|143159|
|2020-03-01 02:00:00|  en| 14969|
|2020-03-01 13:00:00|  en|186802|
|2020-03-01 10:00:00|  en|131686|
|2020-03-01 21:00:00|  en|213787|
|2020-03-01 07:00:00|  en|211910|
|2020-03-01 18:00:00|  en|186675|
|2020-03-01 04:00:00|  en| 21901|
|2020-03-01 15:00:00|  en|163710|
|2020-03-01 01:00:00|  en| 23527|
|2020-03-01 12:00:00|  en|202621|
|2020-03-01 09:00:00|  en|110524|
|2020-03-01 20:00:00|  en|220543|
|2020-03-01 20:00:00|  en|  1124|
```

```
|2020-03-01 06:00:00|  en|195339|
|2020-03-01 17:00:00|  en|151283|
|2020-03-01 03:00:00|  en| 22490|
|2020-03-01 14:00:00|  en|182985|
|2020-03-01 00:00:00|  en| 45182|
|2020-03-01 11:00:00|  en|153327|
+-------------------+----+------+
only showing top 20 rows
```

1.1.8 Aggregating Views by Datehour

Grouping the total views for each `datehour` by summing up the `views` column. It then orders the results by `total_views` in descending order:

[16]:
```
import pyspark.sql.functions as F

df_datehour_totals = df_wiki_en \
.groupBy("datehour") \
.agg(F.sum('views').alias('total_views'))

df_datehour_totals.orderBy('total_views', ascending=False)
```

[16]:
```
+-------------------+-----------+
|           datehour|total_views|
+-------------------+-----------+
|2020-03-01 21:00:00|    1642981|
|2020-03-01 06:00:00|    1591160|
|2020-03-01 22:00:00|    1541455|
|2020-03-01 17:00:00|    1535983|
|2020-03-01 18:00:00|    1495387|
|2020-03-01 16:00:00|    1487786|
|2020-03-01 05:00:00|    1469068|
|2020-03-01 07:00:00|    1458756|
|2020-03-01 20:00:00|    1457051|
|2020-03-01 15:00:00|    1446984|
|2020-03-01 19:00:00|    1427811|
|2020-03-01 14:00:00|    1372760|
|2020-03-01 23:00:00|    1353548|
|2020-03-01 08:00:00|    1353292|
|2020-03-01 03:00:00|    1339853|
|2020-03-01 04:00:00|    1312186|
|2020-03-01 12:00:00|    1225647|
|2020-03-01 13:00:00|    1212003|
|2020-03-01 10:00:00|    1211310|
|2020-03-01 09:00:00|    1200977|
+-------------------+-----------+
only showing top 20 rows
```

1.1.9 Convert Spark DataFrame to Pandas DataFrame

Converting the Spark DataFrame to Pandas DataFrame and set the datehour as the index

[17]:
```
spark.conf.set("spark.sql.execution.arrow.enabled", "true")
%time pandas_datehour_totals = df_datehour_totals.toPandas()

pandas_datehour_totals.set_index('datehour', inplace=True)
pandas_datehour_totals.head()
```

```
CPU times: user 72.9 ms, sys: 19.1 ms, total: 92 ms
Wall time: 1.74 s
```

[17]:

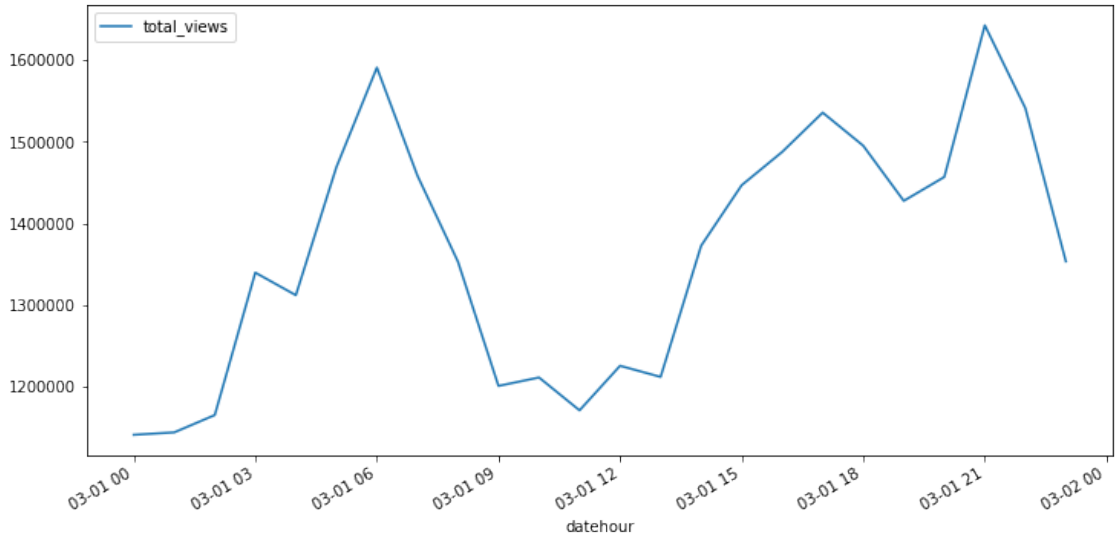
datehour	total_views
2020-03-01 22:00:00	1541455
2020-03-01 09:00:00	1200977
2020-03-01 12:00:00	1225647
2020-03-01 20:00:00	1457051
2020-03-01 10:00:00	1211310

[18]:
```
import matplotlib.pyplot as plt
```

1.1.10 Plotting the Total Views by Datehour Using Pandas

Creating a line plot of the total views by `datehour` from the Pandas DataFrame `pandas_datehour_totals`. The plot is displayed with a size of 12x6 inches.

[19]:
```
pandas_datehour_totals.plot(kind='line',figsize=(12,6));
```



1.1.11 Plot Multiple Columns

Create a new Spark DataFrame and pivot the wiki column to create multiple rows for each wiki value

[20]:
```
import pyspark.sql.functions as F

df_wiki_totals = df_wiki_en \
.groupBy("datehour") \
.pivot("wiki") \
.agg(F.sum('views').alias('total_views'))

df_wiki_totals
```

[20]:
```
+-------------------+------+------+
|           datehour|    en|  en.m|
+-------------------+------+------+
|2020-03-01 22:00:00|558358|983097|
|2020-03-01 09:00:00|638692|562285|
|2020-03-01 12:00:00|633432|592215|
|2020-03-01 20:00:00|615714|841337|
|2020-03-01 05:00:00|588808|880260|
|2020-03-01 10:00:00|644680|566630|
|2020-03-01 14:00:00|685500|687260|
|2020-03-01 19:00:00|592967|834844|
|2020-03-01 03:00:00|391300|948553|
|2020-03-01 01:00:00|360511|783510|
|2020-03-01 04:00:00|383489|928697|
|2020-03-01 18:00:00|645590|849797|
|2020-03-01 00:00:00|382154|758920|
|2020-03-01 07:00:00|839531|619225|
|2020-03-01 08:00:00|783419|569873|
|2020-03-01 13:00:00|619111|592892|
|2020-03-01 11:00:00|594027|577016|
|2020-03-01 15:00:00|695881|751103|
|2020-03-01 16:00:00|661878|825908|
|2020-03-01 23:00:00|484077|869471|
+-------------------+------+------+
only showing top 20 rows
```

1.1.12 Convert to Pandas DataFrame

[21]:
```
pandas_wiki_totals = df_wiki_totals.toPandas()

pandas_wiki_totals.set_index('datehour', inplace=True)
pandas_wiki_totals.head()
```

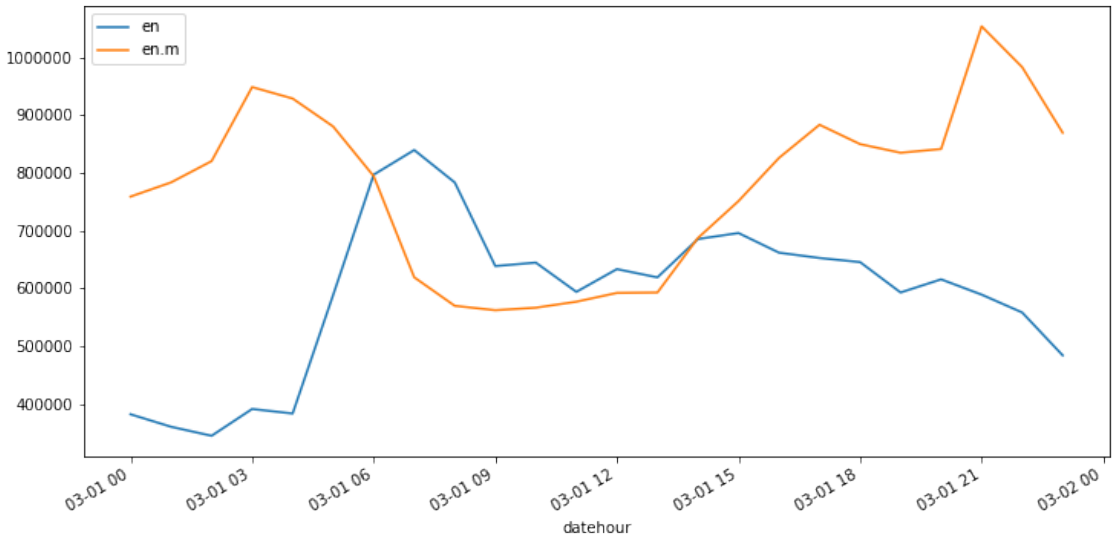
[21]:

datehour	en	en.m
2020-03-01 22:00:00	558358	983097
2020-03-01 09:00:00	638692	562285
2020-03-01 12:00:00	633432	592215
2020-03-01 20:00:00	615714	841337
2020-03-01 10:00:00	644680	566630

1.1.13 Create plot with line for each column

[22]:
```
pandas_wiki_totals.plot(kind='line',figsize=(12,6))
```

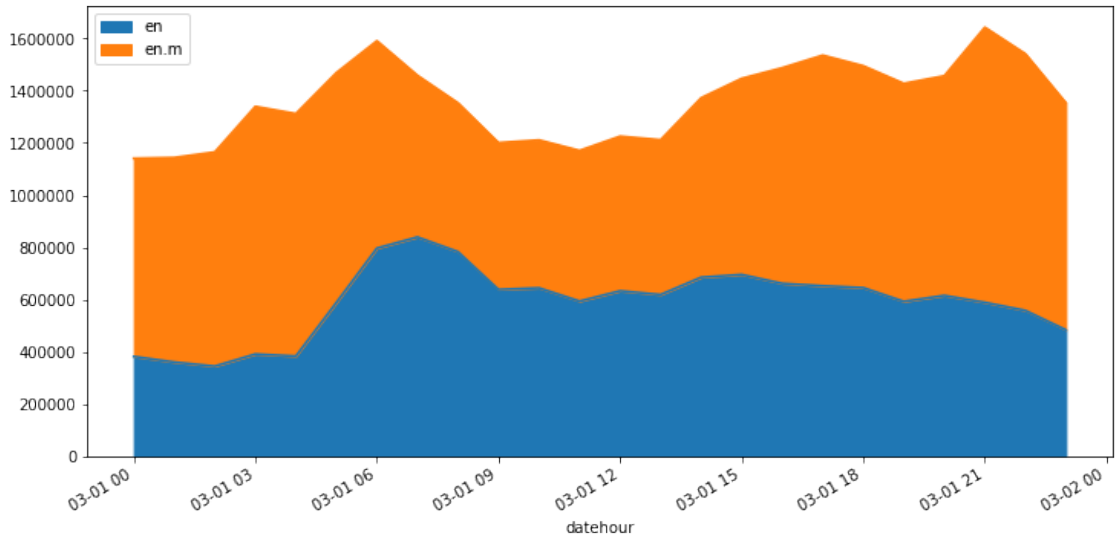
[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcf530d97d0>



1.1.14 Create stacked area plot

[23]:
```
pandas_wiki_totals.plot.area(figsize=(12,6))
```

[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcf533dced0>


en
en.m
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
03-01 00
03-01 03
03-01 06
03-01 09
03-01 12
03-01 15
03-01 18
03-01 21
03-02 00
datehour

[ ]: