BigQuery Storage & Spark DataFrames

November 25, 2024

0.1 Scala Version

[8]:
```
!scala -version
```

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

0.2 Creating Spark Session

[9]:
```
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()
```

0.3 Enable repl.eagerEval

[10]:
```
# 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

spark.conf.set("spark.sql.repl.eagerEval.enabled",True)
```

0.4 Reading BigQuery table into Spark DataFrame

[11]:
```
# Filtered for english version of Wikipedia for both desktop and mobile versions

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)
```

0.5 Selecting the required columns and filtering for English version

[12]:
```
df_wiki_en = df_wiki_pageviews \
  .select("title", "wiki", "views") \
  .where("views > 1000 AND wiki in ('en', 'en.m')") \
  .cache()

df_wiki_en
```

[12]:
```
+--------------------+----+------+
|               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
```

0.6 Grouping by title and ordering by page views to see the top pages

[13]:
```
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)
```

```
[13]: +--------------------+-----------+
|               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
```

0.7 Writing Spark Dataframe to BigQuery table

[14]:
```
# Update to your GCS bucket
gcs_bucket = 'andrewmarfo'

# Update to your BigQuery dataset name you created
bq_dataset = 'wikidataset2024'

# 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()
```

0.8 USING SPARK SQL

0.9 Checking the Scala version

[2]:
```
!scala -version
```

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

0.10 Creating Spark Session

[3]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
    .appName('1.2. BigQuery Storage & Spark SQL - Python')\
    .config('spark.jars', 'gs://spark-lib/bigquery/spark-bigquery-latest_2.12.
 ↪jar') \
    .getOrCreate()
```

0.11 Enable repl.eagerEval

[4]:
```
# 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

spark.conf.set("spark.sql.repl.eagerEval.enabled",True)
```

0.12 Reading BigQuery table into Spark Dataframe

[5]:
```
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)
```

0.13 Creating temp table

[6]:
```
# Creating temp table to be used in Spark SQL queries
df_wiki_pageviews.createOrReplaceTempView("wiki_pageviews")

# Selecting required columns and appling a filter using WHERE
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
```

[6]:
```
+--------------------+----+------+
|               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
```

0.14 Creating a wiki en pageviews table

[7]:
```
df_wiki_en.createOrReplaceTempView("wiki_en")
```

0.15 Grouping by title and finding the top pages by page views

[8]:
```
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
```

[8]:
```
+--------------------+-----------+
|               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
```

0.16 Writing Spark Dataframe to BigQuery table

[9]:
```
# Updating GCS bucket
gcs_bucket = 'andrewmarfo'

# Updating dataset
bq_dataset = 'wikidataset2024'

# 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()
```

0.17 Spark DataFrames & Pandas Plotting - Python

0.17.1 Reading BigQuery table into Spark DataFrame. Filtering to include the date-hour

[11]:
```
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)
```

0.18 Selecting required columns and applying a filter using where() which is an alias for filter() then caching the table

[12]:
```
df_wiki_en = df_wiki_pageviews \
  .select("datehour", "wiki", "views") \
  .where("views > 1000 AND wiki in ('en', 'en.m')") \
  .cache()

df_wiki_en
```

[12]:
```
+-------------------+----+------+
|           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
```

0.19 Grouping by title and ordering by page views to see the top pages

[13]:
```
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)
```

```
[13]: +-------------------+-----------+
      |           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
```

0.20 Converting Spark DataFrame to Pandas DataFrame

```
[14]: # Converting the Spark DataFrame to Pandas DataFrame and seting the datehour as␣
       ↪the index
      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()
```

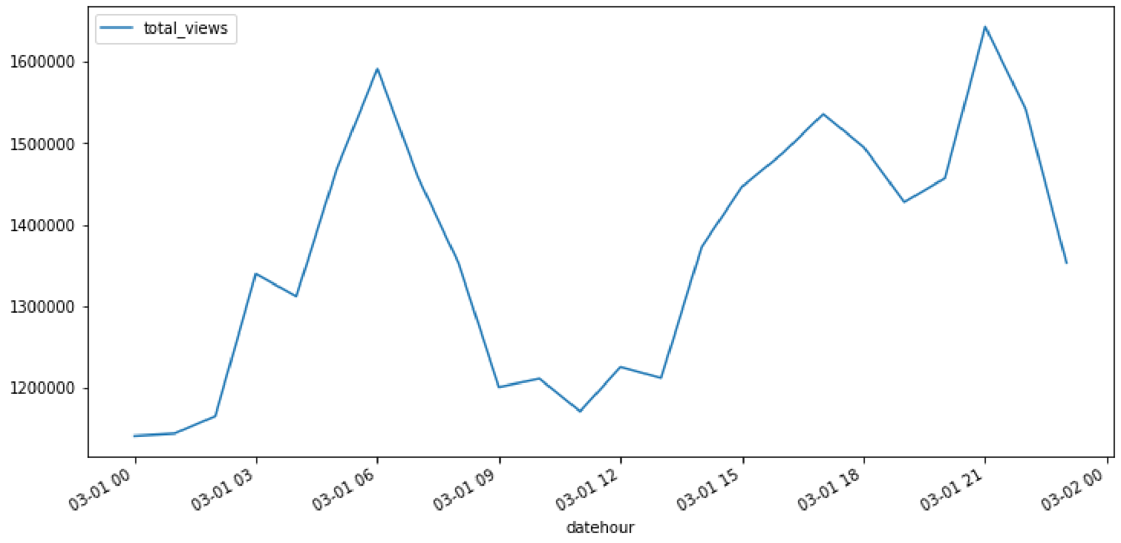
```
CPU times: user 26.3 ms, sys: 11 ms, total: 37.3 ms
Wall time: 1.89 s
```

```
[14]:                      total_views
      datehour
      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
```

0.21 Plotting Pandas Dataframe

[16]:
```python
import matplotlib.pyplot as plt
pandas_datehour_totals.plot(kind='line',figsize=(12,6));
```



0.22 Ploting Multiple Columns

[17]:
```python
# Creating a new Spark DataFrame and pivoting the wiki column to create␣
↪multiple rows for each wiki value

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
```

[17]:
```
+-------------------+------+------+
|           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|
|2020-03-01 05:00:00|588808|880260|
|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
```

0.23 Converting to Pandas Dataframe

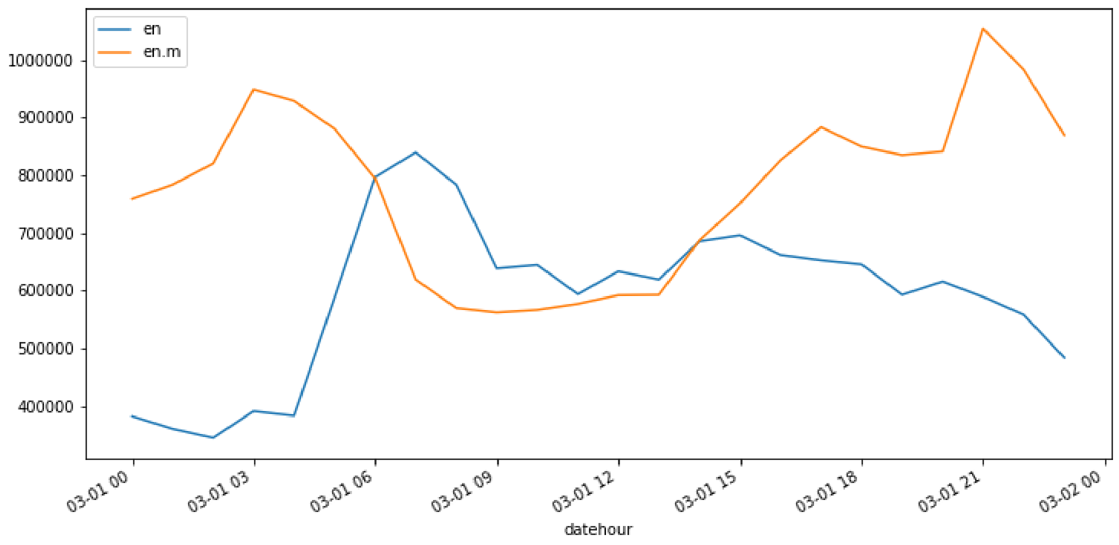
[18]:
```
pandas_wiki_totals = df_wiki_totals.toPandas()

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

[18]:
```
                         en    en.m
datehour
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
```

0.24 Ploting with line for each column

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

[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4d0fafa90>
```


en
en.m
1000000
900000
800000
700000
600000
500000
400000
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

```
[ ]:
```