Visualisation

November 24, 2024

0.1 Checking the Scala Version

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

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

0.2 Creating Spark Session

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

0.3 Enabling repl.eagerEval

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

0.4 Reading BigQuery table into Spark DataFrame

Use filter() to query data from a partitioned table

[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)
```

Select required columns and apply a filter using where() which is an alias for filter() then cache the table

[6]:
```
df_wiki_en = df_wiki_pageviews \
  .select("datehour", "wiki", "views") \
  .where("views > 1000 AND wiki in ('en', 'en.m')") \
  .cache()

df_wiki_en
```

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

[7]:
```
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)
```

[7]:
```
+-------------------+-----------+
|           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
```

0.5 Convert Spark DataFrame to Pandas DataFrame

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

[8]:
```
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 19.5 ms, sys: 13.6 ms, total: 33.1 ms
Wall time: 3.21 s
```

[8]:

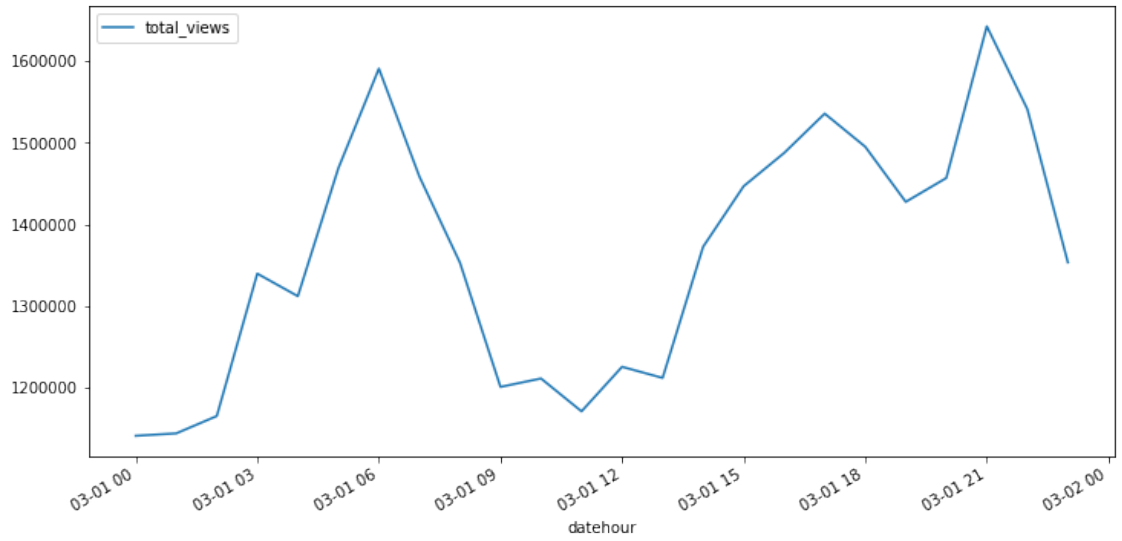
datehour	total_views
2020-03-01 22:00:00	1541455
2020-03-01 09:00:00	1200977
2020-03-01 12:00:00	1225647
2020-03-01 20:00:00	1457051
2020-03-01 10:00:00	1211310

0.6 Plotting Pandas Dataframe

[9]:
```
import matplotlib.pyplot as plt
```

Use the Pandas plot function to create a line chart

[10]:
```
pandas_datehour_totals.plot(kind='line',figsize=(12,6));
```



0.7 Plotting Multiple Columns

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

[11]:
```
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
```

[11]:
```
+-------------------+------+------+
|           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
```

0.8 Converting to Pandas DataFrame

[12]:
```
pandas_wiki_totals = df_wiki_totals.toPandas()

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

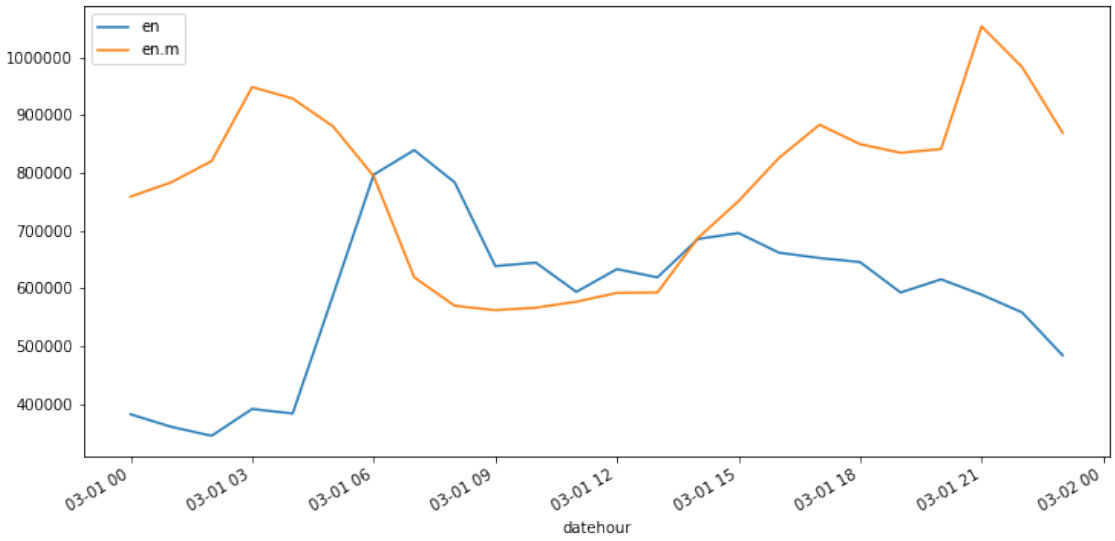
[12]:

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

0.9 Creating plot with line for each column

[13]:
```
pandas_wiki_totals.plot(kind='line',figsize=(12,6))
```

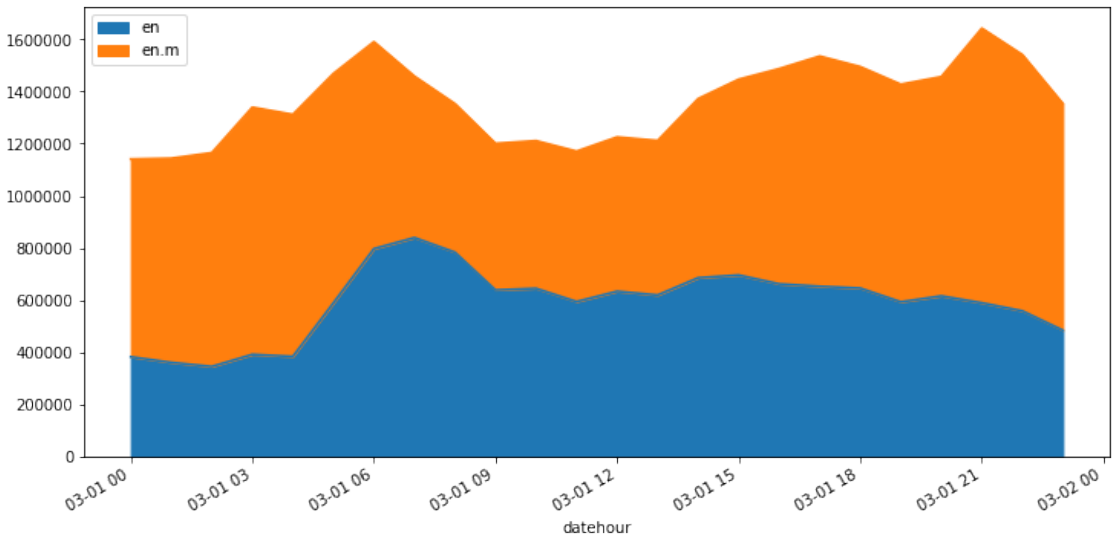
[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd5906c6e10>
```



0.10 Creating stacked area plot

[14]:
```
pandas_wiki_totals.plot.area(figsize=(12,6))
```

[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd57bd5ce10>
```



[ ]:
```

```