4_data_wrangling

March 27, 2022

1 Data Wrangling with Spark

This is the code used in the previous screencast. Run each code cell to understand what the code does and how it works.

These first three cells import libraries, instantiate a SparkSession, and then read in the data set

```
In [1]: from pyspark.sql import SparkSession
        from pyspark.sql.functions import udf
        from pyspark.sql.types import StringType
        from pyspark.sql.types import IntegerType
        from pyspark.sql.functions import desc
        from pyspark.sql.functions import asc
        from pyspark.sql.functions import sum as Fsum

        import datetime

        import numpy as np
        import pandas as pd
        %matplotlib inline
        import matplotlib.pyplot as plt
```

```
In [2]: spark = SparkSession \
            .builder \
            .appName("Wrangling Data") \
            .getOrCreate()
```

```
In [3]: path = "data/sparkify_log_small.json"
        user_log = spark.read.json(path)
```

2 Data Exploration

The next cells explore the data set.

```
In [4]: user_log.take(5)
```

```
Out[4]: [Row(artist='Showaddywaddy', auth='Logged In', firstName='Kenneth', gender='M', itemInSe
         Row(artist='Lily Allen', auth='Logged In', firstName='Elizabeth', gender='F', itemInSes
```

```
Row(artist='Cobra Starship Featuring Leighton Meester', auth='Logged In', firstName='Ve
Row(artist='Alex Smoke', auth='Logged In', firstName='Sophee', gender='F', itemInSessio
Row(artist=None, auth='Logged In', firstName='Jordyn', gender='F', itemInSession=0, las
```

In [5]:
```
user_log.printSchema()
```

```
root
 |-- artist: string (nullable = true)
 |-- auth: string (nullable = true)
 |-- firstName: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- itemInSession: long (nullable = true)
 |-- lastName: string (nullable = true)
 |-- length: double (nullable = true)
 |-- level: string (nullable = true)
 |-- location: string (nullable = true)
 |-- method: string (nullable = true)
 |-- page: string (nullable = true)
 |-- registration: long (nullable = true)
 |-- sessionId: long (nullable = true)
 |-- song: string (nullable = true)
 |-- status: long (nullable = true)
 |-- ts: long (nullable = true)
 |-- userAgent: string (nullable = true)
 |-- userId: string (nullable = true)
```

In [6]:
```
user_log.describe().show()
```

```
+-------+-----------------+----------+---------+------+-----------------+--------+------------
|summary|           artist|      auth|firstName|gender|    itemInSession|lastName|          le
+-------+-----------------+----------+---------+------+-----------------+--------+------------
|  count|             8347|     10000|     9664|  9664|            10000|    9664|
|   mean|            461.0|      null|     null|  null|          19.6734|    null|249.648658749
| stddev|            300.0|      null|     null|  null|25.382114916132597|    null|95.0043713078
|    min|              !!!|     Guest|   Aakash|     F|                0| Acevedo|          1.1
|    max|ÃČÂŞlafur Arnalds|Logged Out|     Zoie|     M|              163|  Zuniga|        1806.
+-------+-----------------+----------+---------+------+-----------------+--------+------------
```

In [7]:
```
user_log.describe("artist").show()
```

```
+-------+-----------------+
|summary|           artist|
+-------+-----------------+
|  count|             8347|
|   mean|            461.0|
```

```
| stddev|            300.0|
|    min|              !!!|
|    max|ÃČÂŞlafur Arnalds|
+-------+-----------------+
```

```
In [8]: user_log.describe("sessionId").show()
```

```
+-------+------------------+
|summary|         sessionId|
+-------+------------------+
|  count|             10000|
|   mean|         4436.7511|
| stddev|2043.1281541827557|
|    min|                 9|
|    max|              7144|
+-------+------------------+
```

```
In [9]: user_log.count()
```

```
Out[9]: 10000
```

```
In [ ]: user_log.select("page").dropDuplicates().sort("page").show()
```

```
In [ ]: user_log.select(["userId", "firstname", "page", "song"]).where(user_log.userId == "1046"
```

3 Calculating Statistics by Hour

```
In [ ]: get_hour = udf(lambda x: datetime.datetime.fromtimestamp(x / 1000.0). hour)
```

```
In [ ]: user_log = user_log.withColumn("hour", get_hour(user_log.ts))
```

```
In [ ]: user_log.head()
```

```
In [ ]: songs_in_hour = user_log.filter(user_log.page == "NextSong").groupby(user_log.hour).coun
```

```
In [ ]: songs_in_hour.show()
```

```
In [ ]: songs_in_hour_pd = songs_in_hour.toPandas()
        songs_in_hour_pd.hour = pd.to_numeric(songs_in_hour_pd.hour)
```

```
In [ ]: plt.scatter(songs_in_hour_pd["hour"], songs_in_hour_pd["count"])
        plt.xlim(-1, 24);
        plt.ylim(0, 1.2 * max(songs_in_hour_pd["count"]))
        plt.xlabel("Hour")
        plt.ylabel("Songs played");
```

4 Drop Rows with Missing Values

As you'll see, it turns out there are no missing values in the userID or session columns. But there are userID values that are empty strings.

```
In [ ]: user_log_valid = user_log.dropna(how = "any", subset = ["userId", "sessionId"])

In [ ]: user_log_valid.count()

In [ ]: user_log.select("userId").dropDuplicates().sort("userId").show()

In [ ]: user_log_valid = user_log_valid.filter(user_log_valid["userId"] != "")

In [ ]: user_log_valid.count()
```

5 Users Downgrade Their Accounts

Find when users downgrade their accounts and then flag those log entries. Then use a window function and cumulative sum to distinguish each user's data as either pre or post downgrade events.

```
In [ ]: user_log_valid.filter("page = 'Submit Downgrade'").show()

In [ ]: user_log.select(["userId", "firstname", "page", "level", "song"]).where(user_log.userId

In [ ]: flag_downgrade_event = udf(lambda x: 1 if x == "Submit Downgrade" else 0, IntegerType())

In [ ]: user_log_valid = user_log_valid.withColumn("downgraded", flag_downgrade_event("page"))

In [ ]: user_log_valid.head()

In [ ]: from pyspark.sql import Window

In [ ]: windowval = Window.partitionBy("userId").orderBy(desc("ts")).rangeBetween(Window.unbound

In [ ]: user_log_valid = user_log_valid.withColumn("phase", Fsum("downgraded").over(windowval))

In [ ]: user_log_valid.select(["userId", "firstname", "ts", "page", "level", "phase"]).where(use
```