databricks Spark Project

(https://databricks.com)

1

```
pip install fuzzywuzzy
```

```
Python interpreter will be restarted.
Collecting fuzzywuzzy
  Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl (18 kB)
Installing collected packages: fuzzywuzzy
Successfully installed fuzzywuzzy-0.18.0
Python interpreter will be restarted.
```

2

```
from pyspark.sql import SparkSession
from fuzzywuzzy import process
from pyspark.sql.functions import col, lit
from pyspark.sql.types import StringType
from pyspark.sql import Row
from pyspark.sql.functions import trim, lower, col, sum, count, lit
```

3

```
spark = SparkSession.builder.getOrCreate()
```

4

```
from pyspark.sql.types import StructType, StructField, StringType, IntegerType

# import and clean data
schema = StructType([
    StructField("sentiment", StringType(), True),
    StructField("publication_URL", StringType(), True),
    StructField("product_URL", StringType(), True),
    StructField("got_click", IntegerType(), True),
    StructField("gender", StringType(), True),
    StructField("age_group", StringType(), True)
])

log = spark.read.csv("/FileStore/tables/log.csv", schema=schema, header=False)
log = log.withColumn("product_URL", trim(lower(col("product_URL"))))
log = log.withColumn("publication_URL",trim(lower(col("publication_URL"))))
log = log.withColumn("got_click", trim(col("got_click")))
log = log.withColumn("gender", trim(lower(col("gender"))))
log = log.withColumn("age_group", trim(lower(col("age_group"))))

product = spark.read.csv("/FileStore/tables/products.csv", header=True)
product = product.withColumn("product_type", trim(lower(col("product_type"))))
product = product.withColumn("product", trim(lower(col("product"))))
product = product.withColumn("product_URL", trim(lower(col("product_URL"))))
product = product.dropDuplicates()

schema = StructType([
    StructField("product_type", StringType(), True),
    StructField("category", StringType(), True)
])
product_cat = spark.read.csv("/FileStore/tables/product_categories.csv", schema=schema, header=True)
product_cat = product_cat.withColumn("category", trim(col("category")))
product_cat = product_cat.withColumn("product_type", trim(col("product_type")))
product_cat = product_cat.dropDuplicates()

product_urls = [row.product_URL for row in product.select("product_URL").collect()]
def find_best_match(query, choices):
    if not query:
        return None
    match, score = process.extractOne(query, choices)
    return match

sc = spark.sparkContext
broadcast_urls = sc.broadcast(product_urls)

@udf(StringType())
def best_match_udf(query):
    return find_best_match(query, broadcast_urls.value)

log = log.withColumn("product_URL", best_match_udf(col("product_URL")))
log.show(truncate=False)
```

5

```
from pyspark.sql.functions import trim, lower, col, sum, count, lit

log.show(5)
product.show(5)
product_cat.show(5)
```

```
|      lavazza coffee|https://lavazza.c...| (https://lavazza.c...|)         coffee|
|       nemok blender|http://nemok.co/b...| (http://nemok.co/b...|)        blender|
|     vitamix blender|https://vitamix.c...| (https://vitamix.c...|)        blender|
|hamilton beach bl...|https://hamiltonb...| (https://hamiltonb...|)        blender|
|instantpot pressu...|https://instantpo...|pressure (https://instantpo...|pressure) cooker|
+--------------------+--------------------+--------------+
only showing top 5 rows

+------------+--------------------+
|product_type|            category|
+------------+--------------------+
|     blender|small kitchen app...|
|    computer|consumer electronics|
|   treadmill|   fitness equipment|
```

Q1 For each product, compute all the Publication_URLs containing an ad for that product.

7

```
from pyspark.sql.functions import col, collect_list

joindf_1 = log.join(product, on = log.product_URL == product.product_URL, how = "right")
result_1 = joindf_1.groupBy("product").agg(collect_list("publication_URL").alias("Publication_URLs"))
result_1.show(n=result_1.count())
print(result_1.count())
```

```
|          samsung tv|[https://www.theg...| (https://www.theg...|)
|       dell computer|[https://www.enga...| (https://www.enga...|)
|         coach purse|[https://www.al.c...| (https://www.al.c...|)
|           lg washer|[https://www.npr....| (https://www.npr....|)
|      lavazza coffee|[https://www.thed...| (https://www.thed...|)
|         dell laptop|[https://www.dall...| (https://www.dall...|)
|        docker pants|[https://www.bost...| (https://www.bost...|)
|           ikea sofa|[https://www.cbsn...| (https://www.cbsn...|)
|hamilton beach bl...|[https://www.nbcn...| (https://www.nbcn...|)
|centrum multivita...|[https://mashable...| (https://mashable...|)
|        apple laptop|[https://www.usat...| (https://www.usat...|)
|    remington shaver|[https://www.vox....| (https://www.vox....|)
|instantpot pressu...|[https://www.theg...| (https://www.theg...|)
|           lee jeans|[https://www.nyti...| (https://www.nyti...|)
|     vitamix blender|[https://www.vox....| (https://www.vox....|)
|      jaguar perfume|[https://www.huff...| (https://www.huff...|)
|      samsung washer|[https://www.dail...| (https://www.dail...|)
|  soundwave speakers|[https://www.nyda...| (https://www.nyda...|)
+--------------------+--------------------+

50
```

Q2 For each product type, compute all the Publication_URLs containing an ad for that product type.

9

```
result_2 = joindf_1.groupBy("product_type").agg(collect_list(joindf_1.publication_URL).alias('publication_URL'))
result_2 = result_2.join(product_cat, on = 'product_type', how = 'right')
result_2.show(n=result_2.count())
print(result_2.count())
```

```
|         furniture|[https://www.cdsn...| (https://www.cdsn...|)  household durables|
|            dryer|[https://www.busi...|large (https://www.busi...|large) kitchen app...|
|elliptical trainer|[https://abcnews....| (https://abcnews....|)   fitness equipment|
|          computer|[https://www.dall...|consumer (https://www.dall...|consumer) electronics|
|            coffee|[https://www.thed...| (https://www.thed...|)       packaged food|
|          speakers|[https://www.nyda...|consumer (https://www.nyda...|consumer) electronics|
|               car|[https://www.foxn...| (https://www.foxn...|)      transportation|
|         treadmill|[https://www.nyti...| (https://www.nyti...|)   fitness equipment|
|           perfume|[https://www.exam...| (https://www.exam...|)     beauty products|
|           blender|[https://www.sfga...|small (https://www.sfga...|small) kitchen app...|
|             pants|[https://www.bost...| (https://www.bost...|)            apparel|
+------------------+--------------------+--------------------+

35
```

Q3 For each product, compute the click rate for it. Click rate is the number of times a display of an ad was clicked on (by any user) divided by the number of times it was displayed (to any user). Note the click rate is not specific to each user.

- For each product, compute the click rate for each sentiment type

11

```
from pyspark.sql.functions import sum, count, col

joindf_2 = log.join(product, on = log.product_URL == product.product_URL, how = 'right')
result_3 =
joindf_2.groupBy("product").agg((sum(col("got_click"))/count(col("got_click"))).alias("click_rate")).orderBy("prod
uct")
result_3.show(truncate=False, n=result_3.count())
print(result_3.count())
```

```
|lg washer                 |0.4953271028037383 |
|maybelline lipstick       |0.5609756097560976 |
|maytag dryer              |0.3448275862068966 |
|maytag refrigerator       |0.39655172413793105|
|maytag washer             |0.5069444444444444 |
|nemok blender             |0.569672131147541  |
|nordictrack elliptical    |0.5284090909090909 |
|nordictrack rower         |0.22340425531914893|
|nordictrack treadmill     |0.4897119341563786 |
|remington shaver          |0.34965034965034963|
|samsung dryer             |0.4358974358974359 |
|samsung tv                |0.73125            |
|samsung washer            |0.5509433962264151 |
|sony tv                   |0.39285714285714285|
|soundwave speakers        |0.5458937198067633 |
|starbucks coffee          |0.275974025974026  |
|tesla                     |0.5916666666666667 |
|vitamix blender           |0.5073170731707317 |
+--------------------------+-------------------+

50
```

12

```
result_4 =
joindf_2.groupBy("product","sentiment").agg((sum(col("got_click"))/count(col("got_click"))).alias("click_rate")).o
rderBy("product")
result_4.show(n=result_4.count())
print(result_4.count())
```

```
|            sony tv| negative| 0.0555555555555555|
|  soundwave speakers|  neutral|  0.9552238805970149|
|  soundwave speakers| negative| 0.11940298507462686|
|  soundwave speakers| positive|  0.5616438356164384|
|    starbucks coffee| negative|                 0.3|
|    starbucks coffee|  neutral|  0.3235294117647059|
|    starbucks coffee| positive| 0.20754716981132076|
|              tesla| negative|  0.7916666666666666|
|              tesla| positive| 0.05747126436781609|
|              tesla|  neutral|  0.9876543209876543|
|     vitamix blender| positive|   0.639344262295082|
|     vitamix blender|  neutral|  0.3387096774193548|
|     vitamix blender| negative|  0.5365853658536586|
+--------------------+---------+--------------------+

150
```

Q4 For each product type, compute the click rate for it.

- For each product type, compute the click rate for each sentiment type

14

```
result_5 =
joindf_2.groupBy("product_type").agg((sum(col("got_click"))/count(col("got_click"))).alias("click_rate")).orderBy(
"product_type")
result_5.show(n=result_5.count())
print(result_5.count())
```

```
|        face cream| 0.8055555555555556|
|         furniture| 0.5549738219895288|
|             jeans|0.45147679324894513|
|          lipstick| 0.6656976744186046|
|            makeup| 0.2524752475247525|
|             pants| 0.6858974358974359|
|           perfume| 0.5668934240362812|
|   pressure cooker|                0.5|
|      refrigerator| 0.2872727272727273|
|    rowing machine|0.22340425531914893|
|            shaver|               0.54|
|          speakers| 0.5359801488833746|
|            tablet| 0.5037878787878788|
|        television| 0.5330578512396694|
|         treadmill| 0.4897119341563786|
|           vitamin| 0.6265560165975104|
|            washer| 0.5189048239895697|
|    women's purse| 0.5151515151515151|
+------------------+-------------------+

24
```

15

```
result_6 =
joindf_2.groupBy("product_type","sentiment").agg((sum(col("got_click"))/count(col("got_click"))).alias("click_rate
")).orderBy("product_type")
result_6.show(n=result_6.count())
print(result_6.count())
```

```
|         treadmill|  neutral| 0.5842696629213483|
|         treadmill| negative|0.31645569620253167|
|         treadmill| positive|               0.56|
|           vitamin| negative| 0.8352941176470589|
|           vitamin| positive|0.18666666666666668|
|           vitamin|  neutral| 0.8148148148148148|
|            washer| negative|  0.376984126984127|
|            washer|  neutral| 0.5189873417721519|
|            washer| positive| 0.6474820143884892|
|     women's purse| positive|0.45751633986928103|
|     women's purse| negative|0.38636363636363635|
|     women's purse|  neutral| 0.6944444444444444|
+------------------+---------+-------------------+

72
```

Q5 For each category, compute the click rate for it.

- Similar to above, for each category compute the click rate for each sentiment type

17

```
joindf_3 = joindf_2.join(product_cat, on = joindf_2.product_type == product_cat.product_type, how = "right")
#joindf_3.show(truncate=False)
#joindf_2.show(truncate=False)

result_7 =
joindf_3.groupBy("category").agg((sum(col("got_click"))/count(col("got_click"))).alias("click_rate")).orderBy("cat
egory")
result_7.show(n=result_7.count())
print(result_7.count())

result_8 =
joindf_3.groupBy("category","sentiment").agg((sum(col("got_click"))/count(col("got_click"))).alias("click_rate")).
orderBy("category")
result_8.show(n=result_8.count())
print(result_8.count())
```

```
|              health| negative| 0.8352941176470589|
|              health| positive|0.18666666666666668|
|              health|  neutral| 0.8148148148148148|
|  household durables| positive|0.27906976744186046|
|  household durables|  neutral| 0.5789473684210527|
|  household durables| negative|              0.825|
|large kitchen app...| positive|  0.491588785046729|
|large kitchen app...| negative| 0.3145631067961165|
|large kitchen app...|  neutral| 0.5664621676891616|
|       packaged food| positive|0.24113475177304963|
|       packaged food|  neutral|0.35714285714285715|
|       packaged food| negative| 0.4652777777777778|
|small kitchen app...|  neutral| 0.4509090909090909|
|small kitchen app...| positive|0.45977011494252873|
|small kitchen app...| negative| 0.5755627009646302|
|      transportation|  neutral| 0.5816993464052288|
|      transportation| negative| 0.3918918918918919|
|      transportation| positive| 0.1566265060240964|
+--------------------+---------+-------------------+

34
```

Q6 Choose a product randomly; determine if there are any 'significant' differences in the click rate between positive and negative sentiment type of the ad context for that product type given the gender of the viewer.

- You can use any form of statistical testing to calculate "significance" that you like. Eg. you assume a binomial distribution for click rate and use a z-test. Compute these figures for all 3 genders.

19

```
+---------+----------+------------+-----------+-------------------+
|neutral  |female    |63.0        |101        |0.6237623762376238 |
|positive |male      |62.0        |115        |0.5391304347826087 |
|positive |non-binary|1.0         |1          |1.0                |
|negative |female    |36.0        |139        |0.2589928057553957 |
|negative |male      |31.0        |114        |0.2719298245614035 |
|neutral  |male      |62.0        |104        |0.5961538461538461 |
|positive |female    |66.0        |137        |0.48175182481751827|
+---------+----------+------------+-----------+-------------------+

Analysis for Product Type: jeans

Gender: male
Z-statistic: 4.1167, P-value: 0.0000
Significant Difference: Yes

Gender: female
Z-statistic: 3.8334, P-value: 0.0001
Significant Difference: Yes

Not enough data for gender: non-binary
```