databricks ASP 3.1L - Revenue by Traffic Lab

Revenue by Traffic Lab

Get the 3 traffic sources generating the highest total revenue.

1. Aggregate revenue by traffic source
2. Get top 3 traffic sources by total revenue
3. Clean revenue columns to have two decimal places

Methods

- DataFrame (https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.html): `groupBy`, `sort`, `limit`
- Column (https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.Column.html?highlight=column#pyspark.sql.Column): `alias`, `desc`, `cast`, `operators`
- Built-in Functions (https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql.html?#functions): `avg`, `sum`

```
%run ../../Includes/Classroom-Setup
```

```
Deleted the working directory dbfs:/user/ajaypanday678@hotmail.com/dbacademy/aspwd/asp_3_1l_revenue_by_traffic_lab

Your working directory is
dbfs:/user/ajaypanday678@hotmail.com/dbacademy/aspwd

The source for this dataset is
wasbs://courseware@dbacademy.blob.core.windows.net/apache-spark-programming-with-databricks/v02/

Skipping install of existing dataset to
dbfs:/user/ajaypanday678@hotmail.com/dbacademy/aspwd/datasets

Out[5]: DataFrame[key: string, value: string]
```

Setup

Run the cell below to create the starting DataFrame **`df`**.

```
from pyspark.sql.functions import col

# Purchase events logged on the BedBricks website
df = (spark.read.format("delta").load(events_path)
      .withColumn("revenue", col("ecommerce.purchase_revenue_in_usd"))
      .filter(col("revenue").isNotNull())
      .drop("event_name")
     )

display(df)
```

	device	ecommerce	event_previous_timestamp
1	Chrome OS	▶ {"purchase_revenue_in_usd": 595, "total_item_quantity": 1, "unique_items": 1}	1593611100709726
2	Windows	▶ {"purchase_revenue_in_usd": 595, "total_item_quantity": 1, "unique_items": 1}	1593616541455837
3	Windows	▶ {"purchase_revenue_in_usd": 1195, "total_item_quantity": 1, "unique_items": 1}	1593622510420631
4	macOS	▶ {"purchase_revenue_in_usd": 850.5, "total_item_quantity": 1, "unique_items": 1}	1593843139065128
5	Windows	▶ {"purchase_revenue_in_usd": 2240, "total_item_quantity": 2, "unique_items": 2}	1593607132024445
6	Chrome OS	▶ {"purchase_revenue_in_usd": 1195, "total_item_quantity": 1, "unique_items": 1}	1593613298187795
7	macOS	▶ {"purchase_revenue_in_usd": 1045, "total_item_quantity": 1, "unique_items": 1}	1593615168536877

Truncated results, showing first 1000 rows.

1. Aggregate revenue by traffic source

- Group by `traffic_source`
- Get sum of `revenue` as `total_rev`
- Get average of `revenue` as `avg_rev`

Remember to import any necessary built-in functions.

```
# TODO

from pyspark.sql.functions import *

traffic_df = (df.groupBy("traffic_source").agg(sum("revenue").alias("total_rev"),avg("revenue").alias("avg_rev"))
                      )

display(traffic_df)
```

	traffic_source	total_rev	avg_rev
1	instagram	16177893	1083.437784623627
2	direct	12704560	1083.175036234973
3	youtube	8044326	1087.2180024327613
4	email	78800000.29999994	983.2915347084434
5	facebook	24797837	1076.6221074111058
6	google	47218429	1086.8302950789487

Showing all 6 rows.

1.1: CHECK YOUR WORK

```
from pyspark.sql.functions import round

expected1 = [(12704560.0, 1083.175), (78800000.3, 983.2915), (24797837.0, 1076.6221), (47218429.0, 1086.8303),
(16177893.0, 1083.4378), (8044326.0, 1087.218)]
test_df = traffic_df.sort("traffic_source").select(round("total_rev", 4).alias("total_rev"), round("avg_rev",
4).alias("avg_rev"))
result1 = [(row.total_rev, row.avg_rev) for row in test_df.collect()]

assert(expected1 == result1)
```

2. Get top three traffic sources by total revenue

- Sort by `total_rev` in descending order
- Limit to first three rows

```
# TODO
top_traffic_df = (traffic_df.sort(col("total_rev").desc()).limit(3)
)
display(top_traffic_df)
```

	traffic_source	total_rev	avg_rev
1	email	78800000.29999994	983.2915347084434
2	google	47218429	1086.8302950789487
3	facebook	24797837	1076.6221074111058

Showing all 3 rows.

2.1: CHECK YOUR WORK

```
expected2 = [(78800000.3, 983.2915), (47218429.0, 1086.8303), (24797837.0, 1076.6221)]
test_df = top_traffic_df.select(round("total_rev", 4).alias("total_rev"), round("avg_rev", 4).alias("avg_rev"))
result2 = [(row.total_rev, row.avg_rev) for row in test_df.collect()]

assert(expected2 == result2)
```

3. Limit revenue columns to two decimal places

- Modify columns `avg_rev` and `total_rev` to contain numbers with two decimal places
  - Use `withColumn()` with the same names to replace these columns
  - To limit to two decimal places, multiply each column by 100, cast to long, and then divide by 100

```
# TODO
final_df = (top_traffic_df.withColumn("avg_rev",(col("avg_rev")*100).cast("long")/100).withColumn("total_rev",
(col("total_rev")*100).cast("long")/100)
)

display(final_df)
```

	traffic_source	total_rev	avg_rev
1	email	78800000.29	983.29
2	google	47218429	1086.83
3	facebook	24797837	1076.62

Showing all 3 rows.

3.1: CHECK YOUR WORK

```
expected3 = [(78800000.29, 983.29), (47218429.0, 1086.83), (24797837.0, 1076.62)]
result3 = [(row.total_rev, row.avg_rev) for row in final_df.collect()]

assert(expected3 == result3)
```

4. Bonus: Rewrite using a built-in math function

Find a built-in math function that rounds to a specified number of decimal places

```
# TODO
bonus_df = (top_traffic_df.withColumn("total_rev",round("total_rev",2)).withColumn("avg_rev",round("avg_rev",2))
)

display(bonus_df)
```

	traffic_source	total_rev	avg_rev
1	email	78800000.3	983.29
2	google	47218429	1086.83
3	facebook	24797837	1076.62

Showing all 3 rows.

4.1: CHECK YOUR WORK

```
expected4 = [(78800000.3, 983.29), (47218429.0, 1086.83), (24797837.0, 1076.62)]
result4 = [(row.total_rev, row.avg_rev) for row in bonus_df.collect()]

assert(expected4 == result4)
```

5. Chain all the steps above

```
# TODO
chain_df =
(df.groupBy("traffic_source").agg(sum("revenue").alias("total_rev"),avg("revenue").alias("avg_rev")).sort(col("total_r
ev").desc()).limit(3).withColumn("avg_rev",(col("avg_rev")*100).cast("long")/100).withColumn("total_rev",
(col("total_rev")*100).cast("long")/100).withColumn("total_rev",round("total_rev",2)).withColumn("avg_rev",round("avg_
rev",2))
)

display(chain_df)
```

	traffic_source	total_rev	avg_rev
1	email	78800000.29	983.29
2	google	47218429	1086.83
3	facebook	24797837	1076.62

Showing all 3 rows.

5.1: CHECK YOUR WORK

```
expected5 = [(78800000.29, 983.29), (47218429.0, 1086.83), (24797837.0, 1076.62)]
result5 = [(row.total_rev, row.avg_rev) for row in chain_df.collect()]

assert(expected5 == result5)
```

Clean up classroom

```
classroom_cleanup()
```

```
Dropped the database dbacademy_odl_user_534131_databrickslabs_com_aspwd_asp_3_1l_revenue_by_traffic_lab
```