ASP 2.3L - Purchase Revenues Lab

Purchase Revenues Lab

Prepare dataset of events with purchase revenue.

Tasks

1. Extract purchase revenue for each event
2. Filter events where revenue is not null
3. Check what types of events have revenue
4. Drop unneeded column

Methods

- DataFrame: `select`, `drop`, `withColumn`, `filter`, `dropDuplicates`
- Column: `isNotNull`

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

```
Deleted the working directory dbfs:/user/odl_user_534131@databrickslabs.com/dbacademy/aspwd/asp_2_3l_purchase_revenues
_lab


Your working directory is
dbfs:/user/odl_user_534131@databrickslabs.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/odl_user_534131@databrickslabs.com/dbacademy/aspwd/datasets

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

```
events_df = spark.read.format("delta").load(events_path)
display(events_df)
```

	device	ecommerce	event_name	event_previous_timestam
1	macOS	▸ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	warranty	1593878899217692
2	Windows	▸ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	press	1593876662175340
3	macOS	▸ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	add_item	1593878792892652
4	iOS	▸ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	mattresses	1593878178791663
5	Windows	▸ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	mattresses	null
6	Windows	▸ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	main	null
7	iOS	▸ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	main	null

1. Extract purchase revenue for each event

Add new column `revenue` by extracting `ecommerce.purchase_revenue_in_usd`

```
from pyspark.sql.functions import *
```

```
# TODO
revenue_df = events_df.withColumn("revenue",col("ecommerce.purchase_revenue_in_usd"))
display(revenue_df)
```

	device	ecommerce	event_name	event_previous_timestam
1	macOS	{"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	warranty	1593878899217692
2	Windows	{"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	press	1593876662175340
3	macOS	{"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	add_item	1593878792892652
4	iOS	{"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	mattresses	1593878178791663
5	Windows	{"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	mattresses	null
6	Windows	{"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	main	null
7	iOS	{"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items": null}	main	null

Truncated results, showing first 1000 rows.

1.1: CHECK YOUR WORK

```
expected1 = [5830.0, 5485.0, 5289.0, 5219.1, 5180.0, 5175.0, 5125.0, 5030.0, 4985.0, 4985.0]
result1 = [row.revenue for row in revenue_df.sort(col("revenue").desc_nulls_last()).limit(10).collect()]

assert(expected1 == result1)
```

2. Filter events where revenue is not null

Filter for records where `revenue` is not `null`

```
# TODO
purchases_df = revenue_df.filter(col("revenue").isNotNull())
display(purchases_df)
```

	device	ecommerce	event_name	event_previ
1	Chrome OS	{"purchase_revenue_in_usd": 595, "total_item_quantity": 1, "unique_items": 1}	finalize	1593611100
2	Windows	{"purchase_revenue_in_usd": 595, "total_item_quantity": 1, "unique_items": 1}	finalize	1593616541
3	Windows	{"purchase_revenue_in_usd": 1195, "total_item_quantity": 1, "unique_items": 1}	finalize	1593622510
4	macOS	{"purchase_revenue_in_usd": 850.5, "total_item_quantity": 1, "unique_items": 1}	finalize	1593843139
5	Windows	{"purchase_revenue_in_usd": 2240, "total_item_quantity": 2, "unique_items": 2}	finalize	1593607132
6	Chrome OS	{"purchase_revenue_in_usd": 1195, "total_item_quantity": 1, "unique_items": 1}	finalize	1593613298

7	macOS	{"purchase_revenue_in_usd": 1045, "total_item_quantity": 1, "unique_items": 1}	finalize	1593615168

Truncated results, showing first 1000 rows.

2.1: CHECK YOUR WORK

```
assert purchases_df.filter(col("revenue").isNull()).count() == 0, "Nulls in 'revenue' column"
```

3. Check what types of events have revenue

Find unique `event_name` values in `purchases_df` in one of two ways:

- Select "event_name" and get distinct records
- Drop duplicate records based on the "event_name" only

There's only one event associated with revenues

```
# TODO
distinct_df = purchases_df.dropDuplicates(["event_name"])
display(distinct_df)
```

	device	ecommerce	event_name	event_previous_timestamp
1	Chrome OS	{"purchase_revenue_in_usd": 595, "total_item_quantity": 1, "unique_items": 1}	finalize	1593611100709726

Showing all 1 rows.

4. Drop unneeded column

Since there's only one event type, drop `event_name` from `purchases_df`.

```
# TODO
final_df = purchases_df.drop("event_name")
display(final_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.

4.1: CHECK YOUR WORK

```
expected_columns = {"device", "ecommerce", "event_previous_timestamp", "event_timestamp",
                    "geo", "items", "revenue", "traffic_source",
                    "user_first_touch_timestamp", "user_id"}
assert(set(final_df.columns) == expected_columns)
```

5. Chain all the steps above excluding step 3

```
# TODO
final_df = (events_df
  .withColumn("revenue",col("ecommerce.purchase_revenue_in_usd"))
  .filter(col("revenue").isNotNull())
  .drop("event_name")
)

display(final_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.

5.1: CHECK YOUR WORK

```
assert(final_df.count() == 180678)
```

```
expected_columns = {"device", "ecommerce", "event_previous_timestamp", "event_timestamp",
                    "geo", "items", "revenue", "traffic_source",
                    "user_first_touch_timestamp", "user_id"}
assert(set(final_df.columns) == expected_columns)
```

Clean up classroom

```
classroom_cleanup()
```

```
Dropped the database dbacademy_odl_user_534131_databrickslabs_com_aspwd_asp_2_3l_purchase_revenues_lab
```