databricks ASP 3.4L - Abandoned Carts Lab

Abandoned Carts Lab

Get abandoned cart items for email without purchases.

1. Get emails of converted users from transactions
2. Join emails with user IDs
3. Get cart item history for each user
4. Join cart item history with emails
5. Filter for emails with abandoned cart items

Methods

- DataFrame (https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.html): `join`
- Built-In Functions (https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql.html?#functions): `collect_set`, `explode`, `lit`

- DataFrameNaFunctions (https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrameNaFunctions.html): `fill`

Setup

Run the cells below to create DataFrames **`sales_df`**, **`users_df`**, and **`events_df`**.

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

```
Deleted the working directory dbfs:/user/odl_user_534131@databrickslabs.com/dbacademy/aspwd/asp_3_4l_abandoned_carts_l
ab


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]
```

```
# sale transactions at BedBricks
sales_df = spark.read.format("delta").load(sales_path)
display(sales_df)
```

1	257437	kmunoz@powell-duran.com	1592194221828900	1	1995
2	282611	bmurillo@hotmail.com	1592504237604072	1	940.5
3	257448	bradley74@gmail.com	1592200438030141	1	945
4	257440	jameshardin@campbell-morris.biz	1592197217716495	1	1045
5	283949	whardin@hotmail.com	1592510720760323	1	535.5
6	257444	emily88@cobb.com	1592199040703476	1	1045
7	257449	craig61@luna-oliver.com	1592200459769596	1	1195

Truncated results, showing first 1000 rows.

```
# user IDs and emails at BedBricks
users_df = spark.read.format("delta").load(users_path)
display(users_df)
```

	user_id	user_first_touch_timestamp	email
1	UA000000102357305	1592182691348767	null
2	UA000000102357308	1592183287634953	null
3	UA000000102357309	1592183302736627	null
4	UA000000102357321	1592184604178702	david23@orozco-parker.com
5	UA000000102357325	1592185154063628	null
6	UA000000102357335	1592186122660210	null
7	UA000000102357338	1592186300091435	null

Truncated results, showing first 1000 rows.

```
# events logged on the BedBricks website
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

Truncated results, showing first 1000 rows.

1: Get emails of converted users from transactions

- Select the `email` column in `sales_df` and remove duplicates
- Add a new column `converted` with the value `True` for all rows

Save the result as `converted_users_df`.

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

converted_users_df = (sales_df.select("email").dropDuplicates(["email"]).withColumn("converted",lit(True))
                      )
display(converted_users_df)
```

	email	converted
1	gonzalezphilip@smith.com	true
2	kevin36@hotmail.com	true
3	jennifer11@hotmail.com	true
4	alexandriafoster@coffey-morrow.com	true
5	victoria79@terry.com	true
6	katiepatterson@gmail.com	true
7	ariasjason@nguyen.com	true

Truncated results, showing first 1000 rows.

1.1: Check Your Work

Run the following cell to verify that your solution works:

```
expected_columns = ["email", "converted"]

expected_count = 210370

assert converted_users_df.columns == expected_columns, "converted_users_df does not have the correct columns"

assert converted_users_df.count() == expected_count, "converted_users_df does not have the correct number of rows"

assert converted_users_df.select(col("converted")).first()[0] == True, "converted column not correct"
```

2: Join emails with user IDs

- Perform an outer join on `converted_users_df` and `users_df` with the `email` field
- Filter for users where `email` is not null
- Fill null values in `converted` as `False`

Save the result as `conversions_df`.

```
# TODO
conversions_df = (users_df.join(converted_users_df,"email","outer").filter("email is Not
Null").withColumn("converted",when(col("converted").isNull(),lit(False)).otherwise(col("converted")))
                 )
display(conversions_df)
```

	email	user_id	user_first_touch_timestamp	converted
1	aabbott@fischer-thompson.info	UA000000107293930	1593868005679801	false
2	aacevedo@moss-young.com	UA000000103755561	1592671212475050	false
3	aacosta11@gmail.com	UA000000106362980	1593540790039008	false

4	aadams9@gmail.com	UA000000103384927	1592575968245258	false
5	aadams@coleman.org	UA000000107105749	1593795399348718	false
6	aadams@howard.biz	UA000000104562958	1592928837244180	false
7	aadams@parker.net	UA000000106086190	1593449235669977	false

Truncated results, showing first 1000 rows.

2.1: Check Your Work

Run the following cell to verify that your solution works:

```
expected_columns = ["email", "user_id", "user_first_touch_timestamp", "converted"]

expected_count = 782749

expected_false_count = 572379

assert conversions_df.columns == expected_columns, "Columns are not correct"

assert conversions_df.filter(col("email").isNull()).count() == 0, "Email column contains null"

assert conversions_df.count() == expected_count, "There is an incorrect number of rows"

assert conversions_df.filter(col("converted") == False).count() == expected_false_count, "There is an incorrect number
of false entries in converted column"
```

3: Get cart item history for each user

- Explode the `items` field in `events_df` with the results replacing the existing `items` field

- Group by `user_id`
  - Collect a set of all `items.item_id` objects for each user and alias the column to "cart"

Save the result as `carts_df`.

```
# TODO
carts_df =
(events_df.withColumn("items",explode("items")).groupBy("user_id").agg(collect_set("items.item_id").alias("cart"))
)
display(carts_df)
```

	user_id	cart
1	UA000000102358054	▸ ["M_STAN_T"]
2	UA000000102360011	▸ ["M_STAN_Q"]
3	UA000000102360488	▸ ["M_STAN_Q"]
4	UA000000102360715	▸ ["M_STAN_T"]
5	UA000000102360871	▸ ["M_STAN_T"]
6	UA000000102362166	▸ ["M_STAN_K"]
7	UA000000102362400	▸ ["M_STAN_Q"]

Truncated results, showing first 1000 rows.

3.1: Check Your Work

Run the following cell to verify that your solution works:

```
expected_columns = ["user_id", "cart"]

expected_count = 488403

assert carts_df.columns == expected_columns, "Incorrect columns"

assert carts_df.count() == expected_count, "Incorrect number of rows"

assert carts_df.select(col("user_id")).drop_duplicates().count() == expected_count, "Duplicate user_ids present"
```

4: Join cart item history with emails

- Perform a left join on `conversions_df` and `carts_df` on the `user_id` field

Save result as `email_carts_df`.

```
# TODO
email_carts_df = conversions_df.join(carts_df,"user_id","left")
display(email_carts_df)
```

	user_id	email	user_first_touch_timestamp	converted	cart
1	UA000000102357285	ianortiz@francis.com	1592169133135185	false	null
2	UA000000102357324	mtorres@gmail.com	1592185107111059	false	null
3	UA000000102357348	phillipmorgan@hotmail.com	1592187663145345	true	null
4	UA000000102357373	emorrow@glenn-dorsey.info	1592189303216185	false	null
5	UA000000102357460	michellewilson@simon.info	1592191799247598	true	null
6	UA000000102357468	mccannsarah@stewart.com	1592192054343730	false	null
7					

Truncated results, showing first 1000 rows.

4.1: Check Your Work

Run the following cell to verify that your solution works:

```
expected_columns = ["user_id", "email", "user_first_touch_timestamp", "converted", "cart"]

expected_count = 782749

expected_cart_null_count = 397799

assert email_carts_df.columns == expected_columns, "Columns do not match"

assert email_carts_df.count() == expected_count, "Counts do not match"

assert email_carts_df.filter(col("cart").isNull()).count() == expected_cart_null_count, "Cart null counts incorrect from join"
```

5: Filter for emails with abandoned cart items

- Filter `email_carts_df` for users where `converted` is False
- Filter for users with non-null carts

Save result as `abandoned_carts_df`.

```
# TODO
abandoned_carts_df = (email_carts_df.filter("converted is False").filter("cart is not null")
)
display(abandoned_carts_df)
```

	user_id	email	user_first_touch_timestamp	converted	cart
1	UA000000102358054	markfitzpatrick@hotmail.com	1592198812458125	false	["M_STAN_T"]
2	UA000000102367817	russellpamela@yahoo.com	1592213618560512	false	["M_PREM_K"]
3	UA000000102369539	karenwright@jennings.com	1592214729249771	false	["M_STAN_K"]
4	UA000000102374838	kyle50@huang.com	1592217432667557	false	["M_STAN_Q"]
5	UA000000102376621	ubrown55@yahoo.com	1592218189654409	false	["M_PREM_Q"]
6	UA000000102379071	nelsonchristopher@yahoo.com	1592219147404116	false	["M_PREM_Q"]
7	UA000000102385440	thomaswatkins@yahoo.com	1592221304639231	false	["M_STAN_K"]

Truncated results, showing first 1000 rows.

5.1: Check Your Work

Run the following cell to verify that your solution works:

```
expected_columns = ["user_id", "email", "user_first_touch_timestamp", "converted", "cart"]

expected_count = 204272

assert abandoned_carts_df.columns == expected_columns, "Columns do not match"

assert abandoned_carts_df.count() == expected_count, "Counts do not match"
```

6: Bonus Activity

Plot number of abandoned cart items by product

```
# TODO
abandoned_items_df = (abandoned_carts_df.withColumn("items",explode("cart")).groupBy("items").count()
                      )
display(abandoned_items_df)
```

	items	count
1	P_FOAM_K	5007
2	M_STAN_Q	47008
3	P_FOAM_S	11497
4	M_PREM_Q	11976
5	M_STAN_F	25761
6	M_STAN_T	50315
7	M_PREM_K	9839

Showing all 12 rows.

6.1: Check Your Work

Run the following cell to verify that your solution works:

```
abandoned_items_df.count()
```

```
Out[51]: 12
```

```
expected_columns = ["items", "count"]

expected_count = 12

assert abandoned_items_df.count() == expected_count, "Counts do not match"

assert abandoned_items_df.columns == expected_columns, "Columns do not match"
```

Clean up classroom

```
classroom_cleanup()
```

```
Dropped the database dbacademy_odl_user_534131_databrickslabs_com_aspwd_asp_3_4l_abandoned_carts_lab
```