databricks ASP 3.2L - Active Users Lab

databricks Academy

Active Users Lab

Plot daily active users and average active users by day of week.

1. Extract timestamp and date of events
2. Get daily active users
3. Get average number of active users by day of week
4. Sort day of week in correct order

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

```
Deleted the working directory dbfs:/user/odl_user_534131@databrickslabs.com/dbacademy/aspwd/asp_3_2l_active_users_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]
```

Setup

Run the cell below to create the starting DataFrame of user IDs and timestamps of events logged on the BedBricks website.

```
from pyspark.sql.functions import col

df = (spark
      .read
      .format("delta")
      .load(events_path)
      .select("user_id", col("event_timestamp").alias("ts"))
     )

display(df)
```

	user_id	ts
1	UA000000107379500	1593878946592107
2	UA000000107359357	1593877011756535
3	UA000000107375547	1593878815459100
4	UA000000107370581	1593878809276923

5	UA000000107377108	1593878628143633
6	UA000000107377161	1593878634344194
7	UA000000107370851	1593877936171803

Truncated results, showing first 1000 rows.

1. Extract timestamp and date of events

- Convert `ts` from microseconds to seconds by dividing by 1 million and cast to timestamp
- Add `date` column by converting `ts` to date

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

```
# TODO
datetime_df = (df.withColumn("ts",(col("ts")/1000000).cast("timestamp")).withColumn("date",
(to_date(col("ts")).cast("date")))
)
display(datetime_df)
```

	user_id	ts	date
1	UA000000106459980	2020-07-01T06:33:33.296+0000	2020-07-01
2	UA000000106546041	2020-07-01T15:38:10.744+0000	2020-07-01
3	UA000000106556702	2020-07-01T16:17:02.994+0000	2020-07-01
4	UA000000106525232	2020-07-01T14:34:49.359+0000	2020-07-01
5	UA000000106502389	2020-07-01T13:13:07.617+0000	2020-07-01
6	UA000000106476093	2020-07-01T10:54:59.397+0000	2020-07-01
7	UA000000106528363	2020-07-01T14:43:56.012+0000	2020-07-01

1.1: CHECK YOUR WORK

```
from pyspark.sql.types import DateType, StringType, StructField, StructType, TimestampType

expected1a = StructType([StructField("user_id", StringType(), True),
                         StructField("ts", TimestampType(), True),
                         StructField("date", DateType(), True)])

result1a = datetime_df.schema

assert expected1a == result1a, "datetime_df does not have the expected schema"
```

```
import datetime

expected1b = datetime.date(2020, 6, 19)
result1b = datetime_df.sort("date").first().date

assert expected1b == result1b, "datetime_df does not have the expected date values"
```

2. Get daily active users

- Group by date
- Aggregate approximate count of distinct `user_id` and alias to "active_users"
  - Recall built-in function to get approximate count distinct
- Sort by date
- Plot as line graph

```
# TODO
active_users_df =
(datetime_df.groupBy("date").agg(approx_count_distinct("user_id").alias("active_users")).sort("date")
)
display(active_users_df)
```

	date	active_users
1	2020-06-19	251573
2	2020-06-20	357215
3	2020-06-21	305055
4	2020-06-22	239094
5	2020-06-23	243117
6	2020-06-24	235205
7	2020-06-25	246548

Showing all 16 rows.

2.1: CHECK YOUR WORK

```
from pyspark.sql.types import LongType

expected2a = StructType([StructField("date", DateType(), True),
                         StructField("active_users", LongType(), False)])

result2a = active_users_df.schema

assert expected2a == result2a, "active_users_df does not have the expected schema"
```

```
expected2b = [(datetime.date(2020, 6, 19), 251573), (datetime.date(2020, 6, 20), 357215), (datetime.date(2020, 6, 21),
305055), (datetime.date(2020, 6, 22), 239094), (datetime.date(2020, 6, 23), 243117)]

result2b = [(row.date, row.active_users) for row in active_users_df.take(5)]

assert expected2b == result2b, "active_users_df does not have the expected values"
```

3. Get average number of active users by day of week

- Add `day` column by extracting day of week from `date` using a datetime pattern string
- Group by `day`
- Aggregate average of `active_users` and alias to "avg_users"

```
# TODO
active_dow_df =
(active_users_df.withColumn("day",date_format("date","EEE").cast("string")).groupBy("day").agg(avg("active_users").ali
as("avg_users"))
)
display(active_dow_df)
```

	day	avg_users
1	Sun	282905.5
2	Mon	238195.5
3	Thu	264620
4	Sat	278482
5	Wed	227214
6	Fri	247180.66666666666

7	Tue	260942.5	

Showing all 7 rows.

3.1: CHECK YOUR WORK

```
from pyspark.sql.types import DoubleType

expected3a = StructType([StructField("day", StringType(), True),
                         StructField("avg_users", DoubleType(), True)])

result3a = active_dow_df.schema

assert expected3a == result3a, "active_dow_df does not have the expected schema"
```

```
expected3b = [("Fri", 247180.66666666666), ("Mon", 238195.5), ("Sat", 278482.0), ("Sun", 282905.5), ("Thu", 264620.0),
("Tue", 260942.5), ("Wed", 227214.0)]

result3b = [(row.day, row.avg_users) for row in active_dow_df.sort("day").collect()]

assert expected3b == result3b, "active_dow_df does not have the expected values"
```

Clean up classroom

```
classroom_cleanup()
```

```
Dropped the database dbacademy_odl_user_534131_databrickslabs_com_aspwd_asp_3_2l_active_users_lab
```