DATABRICKS CAPSTONE PROJECT

This project is divided into two essential parts:

1. Part 1 – Data Ingestion:
    - Extracting and Exploring the Data
    - Defining and Applying a Schema
    - Creating the Tables
    - Loading the Results

Here, in this project we use mount JSON data using DBFS. The dataset we have has nested columns which is difficult to read and understand so here we use different functions and methods to work and read and understand data more easily and efficiently.

2. Part 2 – Transformation and Load
    - Parse tweeted URLs using a custom UDF
    - Compute aggregate statistics of most tweeted websites and hashtags by day
    - Join new data to an existing dataset of malicious users
    - Load records into a target database

Here, we process the old dataset and also read the new dataset as well named badactors and combine them to find more insights about the data.

PART 1: DATA INGESTION

STEP 1: In this step we run the following command.

> "%run" – This command is used to run another notebook withing the current notebook.
>
> "./Includes/Classroom-Setup" – This is the notebook that is going to run

Using the above command we set up the environment, load necessary libraries, or define functions and variables.

```
Mar 20, 2025 (3s)                                    4
%run "./Includes/Classroom-Setup"

Initialized classroom variables & functions...

Datasets are already mounted to /mnt/training from s3a://databricks-corp-training/common

Imported Test Library...

All done!
```

STEP 2: In this step, we read the JSON file from the S3 bucket into a Spark DataFrame and prints the schema of the dataframe.

"S3_file_path" – variable that stores the path

"df = spark.read.json()" – this helps to read the JSON file from the path.

"df.printSchema()" – it displays the schema of the dataset.

```
Mar 20, 2025 (1s)                                    9                                    Python
# TODO
#df = # <<FILL_IN>>
# Define the S3 file path
s3_file_path = 's3a://databricks-corp-training/common/twitter/firehose/2018/01/08/18/
twitterstream-1-2018-01-08-18-48-00-bcf3d615-9c04-44ec-aac9-25f966490aa4'

# Read the file into a Spark DataFrame
df = spark.read.json(s3_file_path)

# Display the DataFrame

#display(df.limit(10))
df.printSchema()
(1) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [contributors: string, coordinates: struct ... 36 more fields]
root
 |-- contributors: string (nullable = true)
 |-- coordinates: struct (nullable = true)
 |    |-- coordinates: array (nullable = true)
 |    |    |-- element: double (containsNull = true)
 |    |-- type: string (nullable = true)
 |-- created_at: string (nullable = true)
 |-- delete: struct (nullable = true)
 |    |-- status: struct (nullable = true)
 |    |    |-- id: long (nullable = true)
 |    |    |-- id_str: string (nullable = true)
```

STEP 3: Here, we use the user define function dbtest that takes three parameters i.e. id, expected and result.

Dbtest works as the test cases, if the values are true it gives the output as “Tests passed”.

```
Mar 20, 2025 (1s)                                   10                                   Python
# TEST - Run this cell to test your solution
cols = df.columns

dbTest("ET1-P-08-02-01", 1744, df.count())
dbTest("ET1-P-08-02-02", True, "id" in cols)
dbTest("ET1-P-08-02-03", True, "text" in cols)

print("Tests passed!")
(2) Spark Jobs
Tests passed!
```

STEP 4: In this step, we use count function to count the number of rows and then by using dbtest I can validate whether the information is right or not. Since, the information is right we get “tests passed” as output.

```
Mar 20, 2025 (1s)                                   14
# TODO
dfCount = df.count()
dfCount

(2) Spark Jobs
Out[9]: 1744
                                                    + Code    + Text
Mar 20, 2025 (<1s)                                  15
# TEST - Run this cell to test your solution
dbTest("ET1-P-08-03-01", 1744, dfCount)

print("Tests passed!")
Tests passed!
```

STEP 5: As we saw the basics of spark, we now need to create a schema for the tweet table that is in our dataset. For this we need to import various libraries:

StructType : helps to create a table

StructField: helps to create the col having three parameter i.e. col_name, data_type and nullable or not

Col: helps to refer to a particular column

To_timestamp: converts the col containing string data into timestamptype

Now we create a tweetdf using the schema that we created. We also modify the table and display the tweetdf.

```python
# TODO
from pyspark.sql.types import StructType, StructField, LongType, StringType, IntegerType
from pyspark.sql.functions import col, to_timestamp

# Define the schema for the Tweet table
tweetSchema = StructType([
    StructField("id", LongType(), True),  # tweetID
    StructField("user", StructType([StructField("id", LongType(), True)]), True),  # userID
    StructField("lang", StringType(), True),  # language
    StructField("text", StringType(), True),  # text
    StructField("created_at", StringType(), True)  # createdAt as string initially
])

# Path to the JSON file
path = 's3a://databricks-corp-training/common/twitter/firehose/2018/01/08/18/twitterstream-1-2018-01-08-18-48-00-bcf3d615-9c04-44ec-aac9-25f966490aa4'

# Read the JSON file into a DataFrame using the defined schema
tweetDF = spark.read.schema(tweetSchema).json(path)

# Rename columns and flatten the structure
tweetDF = tweetDF.withColumnRenamed("id", "tweetID") \
                 .withColumn("userID", col("user.id")) \
                 .drop("user")

# Display the DataFrame
display(tweetDF)
```

Below is the tweetdf.

```python
# Display the DataFrame
display(tweetDF)
```

(2) Spark Jobs

tweetDF: pyspark.sql.dataframe.DataFrame = [tweetID: long, lang: string ... 3 more fields]

Table

	tweetID	l..	text	created_at
1	null	null	null	null
2	null	null	null	null
3	9504389542720962...	en	RT @TheTinaVasquez: Quick facts for the know-nothings who will tweet me today: MS-13 began in Los Angeles. The U.S. helped fund ...	Mon Jan 08 18:47:59 +000
4	9504389542889144...	ja	【太ももを引きしめるエクササイズ】足あげ～1．うつぶせに寝る。2．片方の足先を床から10センチ上にあげてキープ3．2の...	Mon Jan 08 18:47:59 +000
5	9504389542764503...	tr	Ben bir beni bulup icine girip saklanirsam kim beni bulur	Mon Jan 08 18:47:59 +000
6	9504389542804725...	ar	تواصل قالوا عن قطر المتحدث باسم الجيش الصهيوني يشكر قناة الجزيرة https://t.co/j0RgDwS36n ... #صاروخ_سعودي_يرعب_ايران	Mon Jan 08 18:47:59 +000
7	9504389542888898...	en	*Before you argue about your dirty house someone didn't clean or sweep -* *Think of the people who are living in the streets.*	Mon Jan 08 18:47:59 +000
8	9504389542806691...	en	RT @TippyLexx: Bruh you ever accidentally open a message and be like damn now I gotta reply 😂😂	Mon Jan 08 18:47:59 +000
9	9504389542764421...	pt	RT @MorraoTudo2: A liberdade é só questão de tempo, solta os faixa preta 🔐🔐⏳✌️	Mon Jan 08 18:47:59 +000
10	9504389542764789...	en	I just want this all to be over	Mon Jan 08 18:47:59 +000
11	9504389542890332...	ar	RT @Arab_original: قطاع كان ممكن حل ولا اروع للبطاله لكن وزارة النقل قررت ان لا تنظم السوق يحجه السوق الحرة !! اي حريه والشركتين يسحق ...	Mon Jan 08 18:47:59 +000
12	9504389542890332...	ru	RT @craneswordboi: блять мне так смешно от слова срождество	Mon Jan 08 18:47:59 +000
13	9504389542973030...	en	RT @oscahhhhh: THE MOMENT NATALIE PORTMAN SNAPPED #Tea #GoldenGlobes https://t.co/EJbw5q5T2j	Mon Jan 08 18:47:59 +000
14	9504389542972743...	en	RT @troyesivan: ❤️ 'MY MY MY!' OFFICIAL ARTWORK ❤️ JANUARY 10, 9PM PST ❤️ https://t.co/dBNJZTi7AE	Mon Jan 08 18:47:59 +000
15				

1,744 rows | 0.80s runtime

Refreshed 13 days ago

STEP 6: Here we do a bit of transformation to gain different information about the data.

“tweetSchema.fieldNames()”: fieldnames extract the colnames and stores in variable schema

“schema.sort()”: sorts the colnames according to the alphabets.

“tweetDF.filter()”: checks the null values and remove them and get the count of the remaining rows and using assert we check the conditions and also prints “tests passed” if return true.

```
Mar 20, 2025 (1s)                                                    21

# TEST - Run this cell to test your solution
from pyspark.sql.functions import col

schema = tweetSchema.fieldNames()#extracted the column name
schema.sort()#sorted the name in alphabetical order
tweetCount = tweetDF.filter(col("id").isNotNull()).count()#stores the count of the rows that are not null

dbTest("ET1-P-08-04-01", 'created_at', schema[0])
dbTest("ET1-P-08-04-02", 'id', schema[1])
dbTest("ET1-P-08-04-03", 1491, tweetCount)

assert schema[0] == 'created_at' and schema[1] == 'id'
assert tweetCount == 1491#assert checks the conditions and if its true then returns test passed

print("Tests passed!")
(2) Spark Jobs
Tests passed!
```

STEP 7: Now we can create the fulltweetschema and create the ccorresponding DataFrame.

```
Mar 20, 2025 (1s)                                                    23

# Define the fulltweetschema
fullTweetSchema = StructType([
    StructField("id", StringType(), True),
    StructField("created_at", StringType(), True),
    StructField("text", StringType(), True),
    StructField("user", StructType([
        StructField("id", StringType(), True),
        StructField("name", StringType(), True),
        StructField("screen_name", StringType(), True),
        StructField("location", StringType(), True),
        StructField("friends_count", StringType(), True),
        StructField("followers_count", IntegerType(), True)
    ]), True),
    StructField("entities", StructType([
        StructField("hashtags", ArrayType(StructType([
            StructField("text", StringType(), True)
        ])), True),
        StructField("urls", ArrayType(StructType([
            StructField("url", StringType(), True),
            StructField("expanded_url", StringType(), True),
            StructField("display_url", StringType(), True)
        ])), True)
    ]), True)
])

# Read the JSON file
path = "s3a://databricks-corp-training/common/twitter/firehose/2018/01/08/18/twitterstream-1-2018-01-08-18-48-00-bcf3d615-9c04-44ec-aac9-25f966490aa4"
fullTweetDF = spark.read.schema(fullTweetSchema).json(path)
fullTweetDF.show()
(1) Spark Jobs
```

Below is how our fulltweetDF lookslike.

```
fullTweetDF.show()
▸ (1) Spark Jobs
▸ fullTweetDF: pyspark.sql.dataframe.DataFrame = [id: string, created_at: string ... 3 more fields]
+------------------+--------------------+--------------------------------+----------------------+--------------------+
|               id|          created_at|                            text|                  user|            entities|
+------------------+--------------------+--------------------------------+----------------------+--------------------+
|              null|                null|                            null|                  null|                null|
|              null|                null|                            null|                  null|                null|
|950438954272096257|Mon Jan 08 18:47:...|                RT @TheTinaVasque...|   {371607576, Ash, ...|            {[], []}|
|950438954288914432|Mon Jan 08 18:47:...|【太ももを引きしめるエクササイズ】...|{732417055, 美モテDI...|      {[{diet}], []}|
|950438954276450305|Mon Jan 08 18:47:...|                Ben bir beni bulu...|   {235927210, E., m...|            {[], []}|
|950438954280472576|Mon Jan 08 18:47:...|                سعد ,1564880654}  |...تواصل قالوا عن قط...|
|950438954288889856|Mon Jan 08 18:47:...|                *Before you argue...|   {349070364, Kibir...|            {[], []}|
|950438954280669184|Mon Jan 08 18:47:...|                RT @TippyLexx: Br...|   {340482488, the Q...|            {[], []}|
|950438954276442113|Mon Jan 08 18:47:...|                RT @MorraoTudo2: ...|   {4354072997, gabr...|            {[], []}|
|950438954276478976|Mon Jan 08 18:47:...|                I just want this ...|   {7388972250619125...|            {[], []}|
|950438954289033216|Mon Jan 08 18:47:...|                RT @Arab_original...|   {273646363, close...|            {[], []}|
|950438954289033218|Mon Jan 08 18:47:...|                RT @craneswordboi...|   {1541143441, анал...|            {[], []}|
|950438954297303040|Mon Jan 08 18:47:...|                RT @oscahhhhh: TH...|{2937433261, ＺＯＥ,...|{[{Tea}, {GoldenG...|
|950438954297274368|Mon Jan 08 18:47:...|                RT @troyesivan: ❤...|   {3318213908, Stra...|            {[], []}|
|950438954272215041|Mon Jan 08 18:47:...|                @Capital967 @ADie...|   {2296919349, Mirk...|            {[], []}|
|950438954297208832|Mon Jan 08 18:47:...|                RT @KitoriJP: The...|   {7660176254953144...|            {[], []}|
|950438954297249792|Mon Jan 08 18:47:...|                @KAYA_SWG @VNT_SW...|   {9484838904689582...|            {[], []}|
```

STEP 8: This is how our output lookslike after we filter the null values from the columns using filter and store the data into the fulltweetfiltereddf.

```
Mar 20, 2025 (<1s)                                                    26                                                    Python
# TODO
fullTweetFilteredDF = fullTweetDF.filter(fullTweetDF.id.isNotNull() & fullTweetDF.text.isNotNull())#here we filter the data from fulltweet schema
fullTweetFilteredDF.show()
▸ (1) Spark Jobs
▸ fullTweetFilteredDF: pyspark.sql.dataframe.DataFrame = [id: string, created_at: string ... 3 more fields]
+------------------+--------------------+--------------------------------+----------------------+--------------------+
|               id|          created_at|                            text|                  user|            entities|
+------------------+--------------------+--------------------------------+----------------------+--------------------+
|950438954272096257|Mon Jan 08 18:47:...|                RT @TheTinaVasque...|   {371607576, Ash, ...|            {[], []}|
|950438954288914432|Mon Jan 08 18:47:...|【太ももを引きしめるエクササイズ】...|{732417055, 美モテDI...|      {[{diet}], []}|
|950438954276450305|Mon Jan 08 18:47:...|                Ben bir beni bulu...|   {235927210, E., m...|            {[], []}|
|950438954280472576|Mon Jan 08 18:47:...|                سعد ,1564880654}  |...تواصل قالوا عن قط...|
|950438954288889856|Mon Jan 08 18:47:...|                *Before you argue...|   {349070364, Kibir...|            {[], []}|
|950438954280669184|Mon Jan 08 18:47:...|                RT @TippyLexx: Br...|   {340482488, the Q...|            {[], []}|
|950438954276442113|Mon Jan 08 18:47:...|                RT @MorraoTudo2: ...|   {4354072997, gabr...|            {[], []}|
|950438954276478976|Mon Jan 08 18:47:...|                I just want this ...|   {7388972250619125...|            {[], []}|
|950438954289033216|Mon Jan 08 18:47:...|                RT @Arab_original...|   {273646363, close...|            {[], []}|
|950438954289033218|Mon Jan 08 18:47:...|                RT @craneswordboi...|   {1541143441, анал...|            {[], []}|
|950438954297303040|Mon Jan 08 18:47:...|                RT @oscahhhhh: TH...|{2937433261, ＺＯＥ,...|{[{Tea}, {GoldenG...|
|950438954297274368|Mon Jan 08 18:47:...|                RT @troyesivan: ❤...|   {3318213908, Stra...|            {[], []}|
|950438954272215041|Mon Jan 08 18:47:...|                @Capital967 @ADie...|   {2296919349, Mirk...|            {[], []}|
|950438954297208832|Mon Jan 08 18:47:...|                RT @KitoriJP: The...|   {7660176254953144...|            {[], []}|
|950438954297249792|Mon Jan 08 18:47:...|                @KAYA_SWG @VNT_SW...|   {9484838904689582...|            {[], []}|
|950438954276409346|Mon Jan 08 18:47:...|                to com muita saud...|   {2395396997, Indy...|            {[], []}|
|950438954301472768|Mon Jan 08 18:47:...|            RT @bbbbbkim: 냄비받침...| {2791365483, 돌섬, ...|            {[], []}|
|950438954284797958|Mon Jan 08 18:47:...|                you make me un po...|   {3243825293, tori...|{[], [{https://t...|
```

STEP 9: Run the dbtest to validate the data.

```
Mar 20, 2025 (1s)                                                    27
# TEST - Run this cell to test your solution
dbTest("ET1-P-08-06-01", 1491, fullTweetFilteredDF.count())

print("Tests passed!")#we check whether the test case pass for full tweet df
(2) Spark Jobs
Tests passed!
```

STEP 10: We created the tweet table using the normal way now we will use unix_timestamp for efficient use of created_at column in the data set.

“withColumn”: changes in the whole col

“unix_timestamp”: (col_name, format that you defines)

Then we did some modifications and stored it in the tweetDF. Now we use dbtest to validate the datatype and we get the output as “Tests passed”.

```
Mar 20, 2025 (<1s)                                                   29
# TODO creating tweet table again using timestamp so that the created_at col is more efficient
from pyspark.sql.types import TimestampType
from pyspark.sql.functions import unix_timestamp, col

timestampFormat = "EEE MMM dd HH:mm:ss ZZZZZ yyyy"
tweetDF = (
    fullTweetDF
    .withColumn("created_at", unix_timestamp(col("created_at"), timestampFormat).cast(TimestampType()))#withcolumn:changes in whole col, unix_timestamp:
    (col_name, format that u defined)
    .withColumnRenamed("id", "tweet_id")#renames the col
    .withColumnRenamed("user.id", "user_id")
)
tweetDF: pyspark.sql.dataframe.DataFrame = [tweet_id: string, created_at: timestamp ... 3 more fields]
+ Code    + Text
Mar 20, 2025 (<1s)                                                   30
# TEST - Run this cell to test your solution
from pyspark.sql.types import TimestampType

# Access the `created_at` column instead of `createdAt`
t = tweetDF.select("created_at").schema[0]

dbTest("ET1-P-08-07-01", TimestampType(), t.dataType)

print("Tests passed!")
Tests passed!
```

STEP 11: Now same manner as we created the tweetdf we create the accountdf using the fulltweetschema and do the mandatory modifictaions.

Mar 20, 2025 (1s) 32

```
# TODO create account table
accountDF2 = (spark.read
              .schema(fullTweetSchema)
              .json(path)
              .dropna()
              )

#selects all the nested data within user from fulltweetschema and rename few columns
accountDF=accountDF2.select("user.*").withColumnRenamed("screen_name","screenName").withColumnRenamed("friends_count","friendsCount").withColumnRenamed
("followers_count","followersCount").withColumnRenamed("user_id","userID")
display(accountDF)
```

(2) Spark Jobs

accountDF: pyspark.sql.dataframe.DataFrame = [id: string, name: string ... 4 more fields]

accountDF2: pyspark.sql.dataframe.DataFrame = [id: string, created_at: string ... 3 more fields]

Table

	id	name	screenName	location	friendsCount	followersCount
1	371607576	Ash	smileifyou_love	null	473	160
2	732417055	美モテDIET	bw198e18	null	1641	1285
3	235927210	E.	marlascigarette	null	214	223
4	1564880654	سعد الشمري	rebaab_1326	null	45	0
5	349070364	Kibirango Martin	puskine	Kampala, Uganda	5008	4916
6	340482488	the Queen 👑 ❤️	xNina_Beana	the land	1130	1646
7	4354072997	gabrielfrança 🤠	gbfranca22	cpx da congo 🇨🇩	252	632
8	738897225061912 5...	andy	squeeqi	null	213	160
9	273646363	close account	iiijb53	null	631	427

STEP 12: Below we are using dbtest to validate the information and print the output as “Tests passed”.

Mar 20, 2025 (<1s) 33

```
# TEST - Run this cell to test your solution
cols = accountDF.columns

dbTest("ET1-P-08-08-01", True, "friendsCount" in cols)
dbTest("ET1-P-08-08-02", True, "screenName" in cols)
dbTest("ET1-P-08-08-03", 1491, accountDF.count())


print("Tests passed!")
```

(2) Spark Jobs

```
Tests passed!
```

STEP 13: Here we create the remaining two dataframes hashtagdf and urldf and used explode

Explode: this helps to transform the array into individual elements.

Mar 20, 2025 (1s) 35 Python

```python
# TODO
from pyspark.sql.functions import col, explode
hashtagDF2 = (spark.read
              .schema(fullTweetSchema)
              .json(path)
              .filter(col("entities.hashtags").isNotNull())
              )
urlDF2 = (spark.read
              .schema(fullTweetSchema)
              .json(path)
              .filter(col("entities.urls").isNotNull())
              )
hashtagDF = hashtagDF2.select("id", explode("entities.hashtags")).withColumnRenamed("col", "hashtag")\
.withColumnRenamed("id", "tweetID")
urlDF = urlDF2.select("id", explode("entities.urls")).withColumnRenamed("col", "displayURL")\
.withColumnRenamed("id", "tweetID")
display(hashtagDF)
```

(2) Spark Jobs

hashtagDF: pyspark.sql.dataframe.DataFrame = [tweetID: string, hashtag: struct]
hashtagDF2: pyspark.sql.dataframe.DataFrame = [id: string, created_at: string ... 3 more fields]
urlDF: pyspark.sql.dataframe.DataFrame = [tweetID: string, displayURL: struct]
urlDF2: pyspark.sql.dataframe.DataFrame = [id: string, created_at: string ... 3 more fields]

Table

	tweetID	hashtag
1	9504389542889144...	{"text":"diet"}
2	9504389542804725...	{"text":"صاروخ_سعودي_يرعب_ايران"}
3	9504389542973030...	{"text":"Tea"}

STEP 14: Do the required filter and use dbtest to validate them.

Mar 20, 2025 (1s) 36

```python
# TEST - Run this cell to test your solution
hashtagCols = hashtagDF.columns
urlCols = urlDF.columns
hashtagDFCounts = hashtagDF.count()
urlDFCounts = urlDF.count()

dbTest("ET1-P-08-09-01", True, "hashtag" in hashtagCols)
dbTest("ET1-P-08-09-02", True, "displayURL" in urlCols)
dbTest("ET1-P-08-09-03", 394, hashtagDFCounts)
dbTest("ET1-P-08-09-04", 368, urlDFCounts)

print("Tests passed!")
```

(4) Spark Jobs

```
Tests passed!
```

STEP 15: We are using the time parser policy(it’s a parsing behavior used here)and saving the dataframes as parquet files.

Mar 20, 2025 (4s) 38

```python
# TODO
spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")

accountDF.write.parquet("/tmp/" + username + "2/account.parquet")
tweetDF.write.parquet("/tmp/" + username + "2/tweet.parquet")
```

(2) Spark Jobs

Mar 20, 2025 (3s) 39

```python
hashtagDF.write.parquet("/tmp/" + username + "2/hashtag.parquet")
urlDF.write.parquet("/tmp/" + username + "2/url.parquet")
```

(2) Spark Jobs

STEP 16: Here we are importing dataframe and read the parquet files to the dataframes.

Then we run the dbtest to validate the type of each dataframe and printing “Tests passed” as output.

Mar 20, 2025 (3s) 40 Python

```python
# TEST - Run this cell to test your solution
from pyspark.sql.dataframe import DataFrame

accountDF = spark.read.parquet("/tmp/" + username + "2/account.parquet")
tweetDF = spark.read.parquet("/tmp/" + username + "2/tweet.parquet")
hashtagDF = spark.read.parquet("/tmp/" + username + "2/hashtag.parquet")
urlDF = spark.read.parquet("/tmp/" + username + "2/url.parquet")

dbTest("ET1-P-08-10-01", DataFrame, type(accountDF))
dbTest("ET1-P-08-10-02", DataFrame, type(tweetDF))
dbTest("ET1-P-08-10-03", DataFrame, type(hashtagDF))
dbTest("ET1-P-08-10-04", DataFrame, type(urlDF))

print("Tests passed!")
```

(4) Spark Jobs

```
accountDF: pyspark.sql.dataframe.DataFrame = [id: string, name: string ... 4 more fields]
hashtagDF: pyspark.sql.dataframe.DataFrame = [tweetID: string, hashtag: struct]
tweetDF: pyspark.sql.dataframe.DataFrame = [tweet_id: string, created_at: timestamp ... 3 more fields]
urlDF: pyspark.sql.dataframe.DataFrame = [tweetID: string, displayURL: struct]
Tests passed!
```

PART 2: TRANSFORMATION AND LOAD

STEP 1: In this step we run the following command.

> "%run" – This command is used to run another notebook withing the current notebook.
>
> "./Includes/Classroom-Setup" – This is the notebook that is going to run

Using the above command we set up the environment, load necessary libraries, or define functions and variables.

```
Mar 20, 2025 (40s)                                                    4                                    Python
%run "./Includes/Classroom-Setup"

Initialized classroom variables & functions...

Datasets are already mounted to /mnt/training from s3a://databricks-corp-training/common

Imported Test Library...

Created user-specific database

Using the database aditiofficial0020_gmail_com_dbp.

All done!
```

STEP 2: Create the fulltweetschema using the pyspark.sql.types and pyspark.sql.functions.

StructType : helps to create a table

StructField: helps to create the col having three parameter i.e. col_name, data_type and nullable or not

Col: helps to refer to a particular column

```
Mar 20, 2025 (<1s)                                                    7
from pyspark.sql.types import StructField, StructType, ArrayType, StringType, IntegerType, LongType
from pyspark.sql.functions import col

fullTweetSchema = StructType([
  StructField("id", LongType(), True),
  StructField("user", StructType([
    StructField("id", LongType(), True),
    StructField("screen_name", StringType(), True),
    StructField("location", StringType(), True),
    StructField("friends_count", IntegerType(), True),
    StructField("followers_count", IntegerType(), True),
    StructField("description", StringType(), True)
  ]), True),
  StructField("entities", StructType([
    StructField("hashtags", ArrayType(
      StructType([
        StructField("text", StringType(), True)
      ]),
    ), True),
    StructField("urls", ArrayType(
      StructType([
        StructField("url", StringType(), True),
        StructField("expanded_url", StringType(), True),
        StructField("display_url", StringType(), True)
      ]),
    ), True)
  ]), True),
  StructField("lang", StringType(), True),
  StructField("text", StringType(), True),
  StructField("created_at", StringType(), True)
])
```

STEP 3: This time we create the tweetdf using the schema function and display it.

Mar 20, 2025 (16s) 9 Python

```python
# TODO created a tweet df
path = "s3a://databricks-corp-training/common/twitter/firehose/2018/01/08/18/twitterstream-1-2018-01-08-18-48-00-bcf3d615-9c04-44ec-aac9-25f966490aa4"
tweetDF = (spark.read
           .schema(fullTweetSchema)
           .json(path)
           .filter(col("id").isNotNull()))
display(tweetDF)
```

(2) Spark Jobs

tweetDF: pyspark.sql.dataframe.DataFrame = [id: long, user: struct ... 4 more fields]

Table

	id	user	entities
1	9504389542720962...	{"id":371607576,"screen_name":"smileifyou_love","location":null,"friends_count":473,"followers_count":160,"description":"•Psalm 34:18•...	{"hashtags":[],"urls":[]}
2	9504389542889144...	{"id":732417055,"screen_name":"bw198e18","location":null,"friends_count":1641,"followers_count":1285,"description":"【期間限定】今...	{"hashtags":[{"text":"diet"}],"urls":[]}
3	9504389542764503...	{"id":235927210,"screen_name":"marlascigarette","location":null,"friends_count":214,"followers_count":223,"description":"△"}	{"hashtags":[],"urls":[]}
4	9504389542804725...	{"id":1564880654,"screen_name":"rebaab_1326","location":null,"friends_count":45,"followers_count":0,"description":null}	{"hashtags":[{"text":"خرسعودي_يرعب_ايران
5	9504389542888898...	{"id":349070364,"screen_name":"puskine","location":"Kampala, Uganda","friends_count":5008,"followers_count":4916,"description":"Go...	{"hashtags":[],"urls":[]}
6	9504389542806691...	{"id":340482488,"screen_name":"xNina_Beana","location":"the land ","friends_count":1130,"followers_count":1646,"description":"Prince ...	{"hashtags":[],"urls":[]}
7	9504389542764421...	{"id":4354072997,"screen_name":"gbfranca22","location":"cpx da congo 🚫 ","friends_count":252,"followers_count":632,"description":"m...	{"hashtags":[],"urls":[]}
8	9504389542764789...	{"id":"738897225061912576","screen_name":"squeeqi","location":null,"friends_count":213,"followers_count":160,"description":"We are t...	{"hashtags":[],"urls":[]}
9	9504389542890332...	{"id":273646363,"screen_name":"iiiib53","location":null,"friends_count":631,"followers_count":427,"description":null}	{"hashtags":[],"urls":[]}
10	9504389542890332...	{"id":1541143441,"screen_name":"nappo_what","location":"ꜰᴇᴜᴀ","friends_count":297,"followers_count":925,"description":null}	{"hashtags":[],"urls":[]}
11	9504389542973030...	{"id":2937433261,"screen_name":"zooeeen","location":null,"friends_count":220,"followers_count":3165,"description":"✨ 🌵 (/◕ヲ◕)/*:...	{"hashtags":[{"text":"Tea"},{"text":"Golden

STEP 4: Using dbtest to validate the information and get the output as “tests passed”

Mar 20, 2025 (3s) 10

```python
# TEST - Run this cell to test your solution
dbTest("ET2-P-08-01-01", 1491, tweetDF.count())
dbTest("ET2-P-08-01-02", True, "text" in tweetDF.columns and "id" in tweetDF.columns)

print("Tests passed!")
```

(2) Spark Jobs

```
Tests passed!
```

STEP 5: library re allows you to define a set of rules of a string you want to match.

We also define a function getDomain that extracts the domain name from a given url using regular expression. Then demonstrated by extracting and printing the domain name from a sample url.

Mar 20, 2025 (<1s) 12

```python
# TODO runs for url
import re
def getDomain(URL):#helps to match the string u want to
  pattern = re.compile(r"https?://(www\.)?([^/#?]+).*$")
  match = pattern.search(URL)
  return match.group(2)
URL = "https://www.databricks.com/"
print("The string {} matched {}".format(URL, getDomain(URL)))
```

```
The string https://www.databricks.com/ matched databricks.com
```

Mar 20, 2025 (<1s) 13

```python
# TEST - Run this cell to test your solution
dbTest("ET2-P-08-02-01", "databricks.com",  getDomain("https://www.databricks.com/"))

print("Tests passed!")
```

```
Tests passed!
```

STEP 6: The below code demonstrates how the getDomain function consistently extracts the domains name from various URL formats.

```
Mar 20, 2025 (<1s)                                                    16
#for including all the urls
urls = [
  "https://www.databricks.com/",
  "https://databricks.com/",
  "https://databricks.com/training-overview/training-self-paced",
  "http://www.databricks.com/",
  "http://databricks.com/",
  "http://databricks.com/training-overview/training-self-paced",
  "http://www.apache.org/",
  "http://spark.apache.org/docs/latest/"
]

for url in urls:
  print(getDomain(url))
```

```
databricks.com
databricks.com
databricks.com
databricks.com
databricks.com
databricks.com
apache.org
spark.apache.org
```

STEP 7: The UDF, named getDomainUDF, can be used in Spark SQL queries to extract domain names from urls, with the return type specified as StringType. And the dbtest code snippet runs a test to verify that the getDomainUDF function has been successfully registered as a UDF in spark.

```
Mar 20, 2025 (<1s)                                                    18
# TODO here we are registering the udf to spark to apply custom logics
from pyspark.sql.types import StructField, StructType, ArrayType, StringType, IntegerType, LongType
getDomainUDF = spark.udf.register("getDomainUDF",getDomain,StringType())
```

```
Mar 20, 2025 (<1s)                                                    19
# TEST - Run this cell to test your solution
dbTest("ET2-P-08-03-01", True, bool(getDomainUDF))

print("Tests passed!")
```

```
Tests passed!
```

STEP 8: Here we use explode to create the urldf

Mar 20, 2025 (3s) 21

```
# TODO Create a dataframe called urlDF
from pyspark.sql.functions import explode
urlDF = tweetDF.select(explode("entities.urls.expanded_url").alias("URL"),getDomainUDF("URL").alias("parsedURL"),"created_at")
display(urlDF)
```

(2) Spark Jobs

urlDF: pyspark.sql.dataframe.DataFrame = [URL: string, parsedURL: string ... 1 more field]

Table

	URL	parsedURL	created_at
1	https://www.youtube.com/watch?v=b4iz9nZPzAA	youtube.com	Mon Jan 08 18:47:59 +0000 2018
2	https://twitter.com/i/web/status/950438954284797958	twitter.com	Mon Jan 08 18:47:59 +0000 2018
3	http://bit.ly/OYIKII	bit.ly	Mon Jan 08 18:47:59 +0000 2018
4	https://goo.gl/fb/atjACB	goo.gl	Mon Jan 08 18:47:59 +0000 2018
5	https://www.instagram.com/p/BdsvNFABXNL/	instagram.com	Mon Jan 08 18:47:59 +0000 2018
6	https://twitter.com/i/web/status/950438954301644800	twitter.com	Mon Jan 08 18:47:59 +0000 2018
7	http://alathkar.org	alathkar.org	Mon Jan 08 18:47:59 +0000 2018
8	https://open.spotify.com/album/60eldei5NAr7OjPGr3Ei1B	open.spotify.com	Mon Jan 08 18:47:59 +0000 2018
9	https://twitter.com/i/web/status/950438954301616131	twitter.com	Mon Jan 08 18:47:59 +0000 2018
10	https://twitter.com/frann_frann_/status/949080362218606592	twitter.com	Mon Jan 08 18:47:59 +0000 2018
11	https://twitter.com/thisisinsider/status/950398060072460289	twitter.com	Mon Jan 08 18:47:59 +0000 2018
12	http://goo.gl/TU8gBa	goo.gl	Mon Jan 08 18:47:59 +0000 2018
13	https://twitter.com/i/web/status/950438958470545408	twitter.com	Mon Jan 08 18:48:00 +0000 2018

STEP 9: Using the dbtest to verify the structure and content of the urldf and gives the output as “Tests passed”

Mar 20, 2025 (1s) 22

```
# TEST - Run this cell to test your solution
cols = urlDF.columns
sample = urlDF.first()

dbTest("ET2-P-08-04-01", True, "URL" in cols and "parsedURL" in cols and "created_at" in cols)
dbTest("ET2-P-08-04-02", "https://www.youtube.com/watch?v=b4iz9nZPzAA", sample["URL"])
dbTest("ET2-P-08-04-03", "Mon Jan 08 18:47:59 +0000 2018", sample["created_at"])
dbTest("ET2-P-08-04-04", "youtube.com", sample["parsedURL"])

print("Tests passed!")
```

(1) Spark Jobs

```
Tests passed!
```

STEP 10: We created the url table using the normal way now we will use unix_timestamp for efficient use of created_at column in the data set.

“withColumn”: changes in the whole col

“unix_timestamp”: (col_name, format that you defines)

Mar 20, 2025 (2s) 25 Python

```python
from pyspark.sql.functions import unix_timestamp, hour, col
from pyspark.sql.types import TimestampType
spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")

# Define the timestamp format
timestampFormat = "EEE MMM dd HH:mm:ss ZZZZZ yyyy"

# Parse the timestamp and extract the hour
urlWithTimestampDF = urlDF.withColumn(
    "timestamp", unix_timestamp(col("created_at"), timestampFormat).cast(TimestampType())
).withColumn(
    "hour", hour(col("timestamp"))
)

# Select the required columns
urlWithTimestampDF = urlWithTimestampDF.select("URL", "parsedURL", "timestamp", "hour")

# Display the DataFrame
display(urlWithTimestampDF)
```

(2) Spark Jobs

urlWithTimestampDF: pyspark.sql.dataframe.DataFrame = [URL: string, parsedURL: string ... 2 more fields]

Table

	URL	parsedURL	timestamp	hour
1	https://www.youtube.com/watch?v=b4iz9nZPzAA	youtube.com	2018-01-08T18:47:59.000+00:...	18
2	https://twitter.com/i/web/status/950438954284797958	twitter.com	2018-01-08T18:47:59.000+00:...	18
3	http://bit.ly/OYIKII	bit.ly	2018-01-08T18:47:59.000+00:...	18

STEP 11: Then we did some modifications and stored it in the urlwithtimestampdf. Now we use dbtest to validate the datatype and we get the output as “Tests passed”.

Mar 20, 2025 (1s) 26

```python
# TEST - Run this cell to test your solution
cols = urlWithTimestampDF.columns
sample = urlWithTimestampDF.first()

dbTest("ET2-P-08-05-01", True, "URL" in cols and "parsedURL" in cols and "timestamp" in cols and "hour" in cols)
dbTest("ET2-P-08-05-02", 18, sample["hour"])

print("Tests passed!")
```

(1) Spark Jobs

```
Tests passed!
```

STEP 12: Create the urltrenddf using the schema and some sql queries that helps to sort and analyse the data and display.

Mar 20, 2025 (3s) 28

```python
# TODO

from pyspark.sql.functions import count,col, desc
urlTrendsDF = (urlWithTimestampDF
               .select("hour","parsedURL")
               .groupBy("parsedURL","hour")
               .count().alias("count")
               .orderBy(desc("count"))
               .limit(10)
              )
display(urlTrendsDF)
```

(2) Spark Jobs

urlTrendsDF: pyspark.sql.dataframe.DataFrame = [parsedURL: string, hour: integer ... 1 more field]

Table

	parsedURL	hour	count
1	twitter.com	18	159
2	bit.ly	18	25
3	fb.me	18	17
4	youtu.be	18	16
5	du3a.org	18	15
6	goo.gl	18	12
7	instagram.com	18	10
8	curiouscat.me	18	6

STEP 13: Using dbtest validate the information and give the output as “Tests passed”

Mar 20, 2025 (1s) 29

```
# TEST - Run this cell to test your solution
cols = urlTrendsDF.columns
sample = urlTrendsDF.first()

dbTest("ET2-P-08-06-01", True, "hour" in cols and "parsedURL" in cols and "count" in cols)
dbTest("ET2-P-08-06-02", 18, sample["hour"])
dbTest("ET2-P-08-06-03", "twitter.com", sample["parsedURL"])
dbTest("ET2-P-08-06-04", 159, sample["count"])

print("Tests passed!")
```

(2) Spark Jobs

```
Tests passed!
```

STEP 14: Read the data from the parquet file for the new dataset named bad actors and create a dataframe badactorsdf.

Mar 20, 2025 (8s) 32

```
# TODO
badActorsDF = (spark.read
              .parquet("s3a://databricks-corp-training/common/twitter/supplemental/badactors.parquet")
              )
display(badActorsDF)
```

(3) Spark Jobs

badActorsDF: pyspark.sql.dataframe.DataFrame = [userID: long, screenName: string]

Table

	userID	screenName
1	4875602384	cris_silvag1
2	2641769580	therebel1789
3	9325992365160243...	HiLeonyedek88
4	573733345	KristenSel
5	1458487634	ImPostMalone
6	541253458	Santa_Palabra
7	1942419721	fidgazi
8	9502926532074250...	marylinbrown272
9	2333871434	LryManon
10	3519647781	FeatuuedPromos
11	2287969007	Turguthuseyin45
12	76528887	CoteRubioS

STEP 15: Using dbtest again validate the information and give the output as “Tests passed”

Mar 20, 2025 (<1s) 33

```
# TEST - Run this cell to test your solution
cols = badActorsDF.columns
sample = badActorsDF.first()

dbTest("ET2-P-08-07-01", True, "userID" in cols and "screenName" in cols)
dbTest("ET2-P-08-07-02", 4875602384, sample["userID"])
dbTest("ET2-P-08-07-03", "cris_silvag1", sample["screenName"])

print("Tests passed!")
```

(1) Spark Jobs

```
Tests passed!
```

STEP 16: import functions as F and use join and other select query to get the output and create a dataframe for tweetwithmaliciuosdf and name the col as maliciousact.

Mar 20, 2025 (3s) 35 Python

```python
# TODO
from pyspark.sql import functions as F

tweetDF2 = tweetDF.join(badActorsDF,tweetDF.user.id==badActorsDF.userID,"left_outer")
tweetWithMaliciousDF=tweetDF2.select("*",F.when(tweetDF2.screenName.isNull(),False).otherwise(True).alias("maliciousAcct"))


display(tweetWithMaliciousDF)
```

(3) Spark Jobs

tweetDF2: pyspark.sql.dataframe.DataFrame = [id: long, user: struct ... 6 more fields]

tweetWithMaliciousDF: pyspark.sql.dataframe.DataFrame = [id: long, user: struct ... 7 more fields]

Table

	id	user	entities
1	9504389542720962...	{"id":371607576,"screen_name":"smileifyou_love","location":null,"friends_count":473,"followers_count":160,"description":"•Psalm 34:18•...	{"hashtags":[],"urls":[]}
2	9504389542889144...	{"id":732417055,"screen_name":"bw198e18","location":null,"friends_count":1641,"followers_count":1285,"description":"【期間限定】今...	{"hashtags":[{"text":"diet"}],"urls":[]}
3	9504389542764503...	{"id":235927210,"screen_name":"marlascigarette","location":null,"friends_count":214,"followers_count":223,"description":"△"}	{"hashtags":[],"urls":[]}
4	9504389542804725...	{"id":1564880654,"screen_name":"rebaab_1326","location":null,"friends_count":45,"followers_count":0,"description":null}	{"hashtags":[{"text":"خم_سعودي_يرعب_ايران
5	9504389542888898...	{"id":349070364,"screen_name":"puskine","location":"Kampala, Uganda","friends_count":5008,"followers_count":4916,"description":"Go...	{"hashtags":[],"urls":[]}
6	9504389542806691...	{"id":340482488,"screen_name":"xNina_Beana","location":"the land ","friends_count":1130,"followers_count":1646,"description":"Prince ...	{"hashtags":[],"urls":[]}
7	9504389542764421...	{"id":4354072997,"screen_name":"gbfranca22","location":"cpx da congo 🇨🇩","friends_count":252,"followers_count":632,"description":"m...	{"hashtags":[],"urls":[]}

STEP 17: Using dbtest again validate the information and give the output as “Tests passed”

Mar 20, 2025 (1s) 36 Python

```python
# TEST - Run this cell to test your solution
cols = tweetWithMaliciousDF.columns
sample = tweetWithMaliciousDF.first()

dbTest("ET2-P-08-08-01", True, "maliciousAcct" in cols and "id" in cols)
dbTest("ET2-P-08-08-02", 9504389542720962 57, sample["id"])
dbTest("ET2-P-08-08-03", False, sample["maliciousAcct"])

print("Tests passed!")
```

(2) Spark Jobs

```
Tests passed!
```

STEP 18: Create the df urltrenddf and repartition into 4. Using dbtest again validate the information and give the output as “Tests passed”

Mar 20, 2025 (12s) 38

```python
urlTrendsDF.repartition(4).write.mode("OVERWRITE").parquet(userhome + "/tmp/urlTrends.parquet")
tweetWithMaliciousDF.repartition(4).write.mode("OVERWRITE").parquet(userhome + "/tmp/tweetWithMaliciousDF.parquet")
```

(6) Spark Jobs

Mar 20, 2025 (4s) 39

```python
# TEST - Run this cell to test your solution
urlTrendsDFTemp = spark.read.parquet(userhome + "/tmp/urlTrends.parquet")
tweetWithMaliciousDFTemp = spark.read.parquet(userhome + "/tmp/tweetWithMaliciousDF.parquet")

dbTest("ET2-P-08-09-01", 4, urlTrendsDFTemp.rdd.getNumPartitions())
dbTest("ET2-P-08-09-02", 4, tweetWithMaliciousDFTemp.rdd.getNumPartitions())

print("Tests passed!")
```

(2) Spark Jobs

tweetWithMaliciousDFTemp: pyspark.sql.dataframe.DataFrame = [id: long, user: struct ... 7 more fields]

urlTrendsDFTemp: pyspark.sql.dataframe.DataFrame = [parsedURL: string, hour: integer ... 1 more field]

```
Tests passed!
```