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.

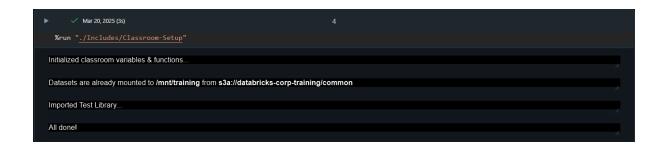
PART 1: DATA INGESTION

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.



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.

```
# TOOO
# # Too
# Define the $3 file path
# S3_file_path = 's3a://databricks-corp-training/common/twitter/firehose/2018/01/08/18/
twitterstream-1-2018-01-08-18-48-00-bcf3d615-9c94-44ec-aac9-25f966490aa4'

# Read the file into a Spark DataFrame
# # spark.read.json($3_file_path)
# Display the DataFrame
# # spark.read.json($3_file_path)
# Display the DataFrame
# # pysparksql.dataframe.DataFrame = [contributors: string. coordinates: struct ... 36 more fields]

root
|-- contributors: string (nullable = true)
|-- coordinates: array (nullable = true)
|-- coordinates: array (nullable = true)
|-- clement: double (containsMull = true)
|-- created_at: 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".

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.



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

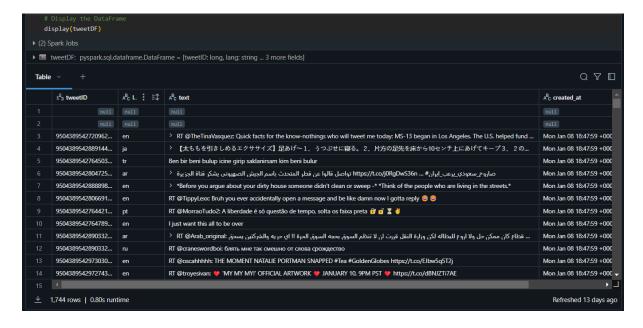
StructField: helps to create the col having three parameter i.e. col_name, data_type and nullable or not

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.

Below is the tweetdf.



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.

```
# 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("ETI-P-08-04-01", 'created_at', schema[0])
dbTest("ETI-P-08-04-02", 'id', schema[1])
dbTest("ETI-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.

```
# Define the fulltweetschems
fullTweetSchema = StructType([

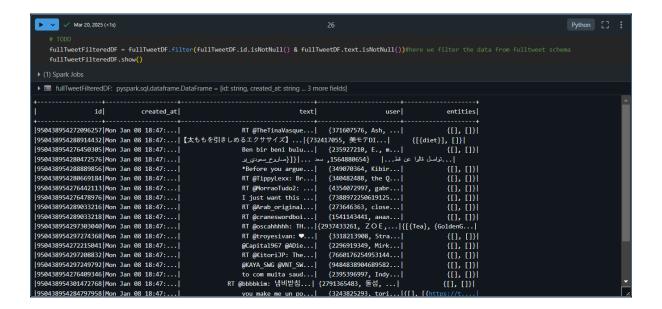
StructField("cated_at", StringType(), True),
StructField("cated_at", StringType(), True),
StructField("cated_at", StringType(), True),
StructField("cated_at", StringType(), True),
StructField("iser", StringType(), True),
StructField("iser", StringType(), True),
StructField("iser", StringType(), True),
StructField("cation", StringType(), True),
StructField("cation", StringType(), True),
StructField("followers_count", IntegerType(), True),
StructField("followers_count", IntegerType(), True)
]), True),
StructField("followers_count", IntegerType(), True)
]), True),
StructField("cation", StringType(), True),
StructField("cation", StringType(), True),
StructField("cut", 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"
fullTweetDE = spark.read.schema(fullTweetSchema).json(path)
fullTweetDE's.show()

) (1) Spark.lobs
```

Below is how our fulltweetDF lookslike.

```
ullTweetDF.show()
 (1) Spark Jobs
   ■ fullTweetDF: pyspark.sql.dataframe.DataFrame = [id: string, created_at: string ... 3 more fields]
                                          created at
                                                                                                                                                                    entities
                       id
                                                                                                         text
                                                                                                                                           user
                    null|
                                                  null|
                                                                                                         nu11
                                                                                                                                           null|
                                                                                                                                                                          nul1
                                                                                                                                                      {[], []}|
{[{diet}], []}|
{[], []}|
|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...|
|950438954276450305|Mon | Jan | 08 | 18:47:...|
                                                                                 Ben bir beni bulu...
                                                                                                                  {235927210, E.,
__,1564880654}
                                                                                                                                                               (13, (13)
|-..ئوام
|{(3, (3)}
|{(3, (3)}
|{(3, (3)}
|{(3, (3)}
950438954280472576|Mon Jan 08 18:47:...|
                                                                                  مد ... [ [ [ [ إصاروخ يسعودي ير
|950438954288889856|Mon Jan 08 18:47:...|
|950438954280669184|Mon Jan 08 18:47:...|
                                                                                 *Before you argue...|
RT @TippyLexx: Br...|
                                                                                                                    {349070364, Kibir...|
{340482488, the Q...|
|950438954276442113|Mon Jan 08 18:47:...|
|950438954276478976|Mon Jan 08 18:47:...|
                                                                                                                    {4354072997, gabr...|
{7388972250619125...|
                                                                                  RT @MorraoTudo2: ...|
                                                                                  I just want this ...
                                                                                                                     {273646363, close...
950438954289033216 Mon Jan 08 18:47:...
                                                                                  RT @craneswordboi...| {1541143441, анал...| {[], []}| RT @oscahhhhh: ТН...|{2937433261, ZОЕ,...|{[{Tea}, {GoldenG...|
950438954289033218 Mon Jan 08 18:47:...
950438954297303040|Mon Jan 08 18:47:...
|950438954297274368|Mon Jan 08 18:47:...
|950438954272215041|Mon Jan 08 18:47:...
                                                                                  RT @troyesivan: ♥...|
@Capital967 @ADie...|
                                                                                                                     {3318213908, Stra...|
{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.



STEP 9: Run the dbtest to validate the data.

```
# 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".

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

•	✓ Mar 20, 2025 (1s)			32			
# TODO create account table accountDF2 = (spark.read .schema(fullTweetSchema) .json(path) .dropna() .dro							
• (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 ∨ +							
	^{AB} c id	A ^B C name	A ^B C screenName	A ^B C location	A ^B C friendsCount	1 ² ₃ followersCount	_
1	371607576	Ash	smileifyou_love	null	473	160	
	732417055	美モテDIET	bw198e18	null	1641	1285	
	235927210	E	marlascigarette	null	214	223	
	1564880654	سعد الشمري	rebaab_1326	null	45	0	
	349070364	Kibirango Martin	puskine	Kampala, Uganda	5008	4916	
	340482488	the Queen 🁑 🧡	xNina_Beana	the land	1130	1646	
	4354072997	gabrielfrança 😇	gbfranca22	cpx da congo 🔞	252	632	
	7388972250619125	andy	squeeqi	null	213	160	
9	273646363	close account	iiib53	null	631	427	

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

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

dbTest("ETI-P-08-08-01", True, "friendsCount" in cols)
dbTest("ETI-P-08-08-02", True, "screenName" in cols)
dbTest("ETI-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.



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

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

dbTest("ETI-P-08-09-01", True, "hashtag" in hashtagCols)
dbTest("ETI-P-08-09-02", True, "displayURL" in urlCols)
dbTest("ETI-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.

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.

```
# 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")
hashtag0f = spark.read.parquet("/tmp/" + username + "2/tweet.parquet")
urlDf = spark.read.parquet("/tmp/" + username + "2/tweet.parquet")

dbTest("ETI-P-08-10-01", DataFrame, type(accountDf))
dbTest("ETI-P-08-10-02", DataFrame, type(tweetDf))
dbTest("ETI-P-08-10-03", DataFrame, type(tweetDf))
dbTest("ETI-P-08-10-04", DataFrame, type(urlDf))

print("Tests passed!")

* (4) Spark Jobs

* accountDf: pysparksql.dataframe.DataFrame = [id: string, name: string ... 4 more fields]

* accountDf: pysparksql.dataframe.DataFrame = [tweet]Dt. string, hashtag2 struct]

* accountDf: pysparksql.dataframe.DataFrame = [tweet]Dt. string, displayURL: struct]

Tests passed!
```

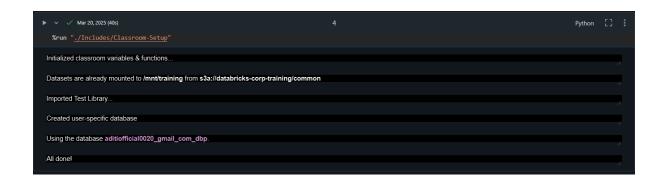
PART 2: TRANSFORMATION AND LOAD

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.



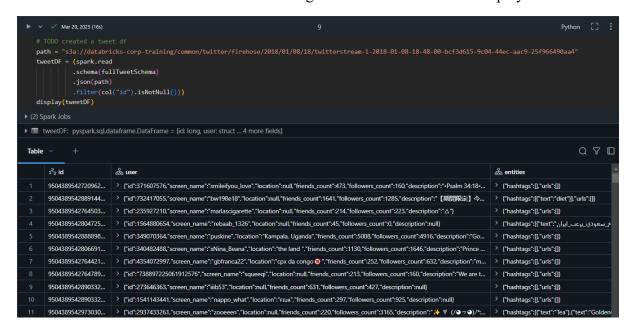
STEP 2: Create the fulltweetschema using the pyspark.sql.types and pyspark.sql.functions.

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

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



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

```
# 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.

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

```
## Mar 20, 2025 (-1s)

#for including all the urls

urls = [

"https://www.databricks.com/",

"https://databricks.com/",

"https://databricks.com/",

"http://databricks.com/",

"http://databricks.com/",

"http://databricks.com/",

"http://databricks.com/",

"http://databricks.com/",

"http://spank.apache.org/",

"http://spank.apache.org/",

"http://spank.apache.org/docs/latest/"

]

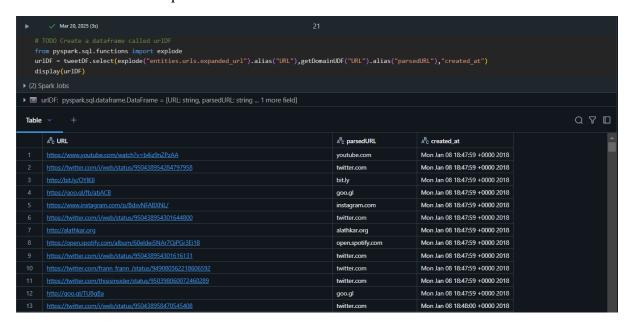
for url in urls:

print(getDomain(url))

databricks.com
databricks.com
databricks.com
databricks.com
databricks.com
databricks.com
databricks.com
databricks.com
databricks.com
gapache.org
spark.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.

STEP 8: Here we use explode to create the urldf



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

```
# 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!")

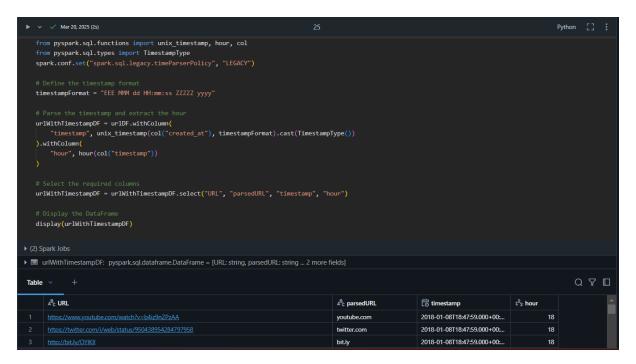
1(1) Spark Jobs

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

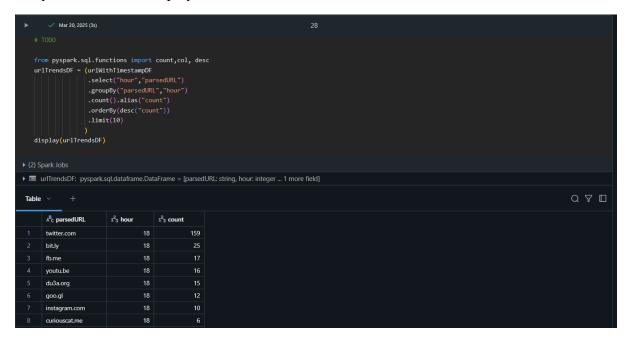
"unix timestamp": (col name, format that you defines)



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".



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



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

```
# 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.



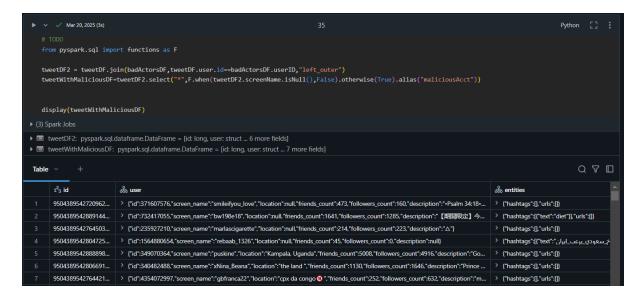
STEP 15: Using dbtest again validate the information and give the output as "Tests passed"

```
# 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 tweetwithmaliciousdf and name the col as maliciousact.



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

```
# 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", 950438954272096257, 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"