Sophie Titlebaum

Objectives: Use Spark to process and perform basic analysis on non-relational data, including its DataFrame and SQL interfaces.

Grading criteria: The tasks should all be completed, and questions should all be answered with Python code, SQL queries, shell commands, and markdown cells. The notebook itself should be completely reproducible (using AWS EC2 instance based on the provided AMI) from start to finish; another person should be able to use the code to obtain the same results as yours. Note that you will receive no more than partial credit if you do not add text/markdown cells explaining your thinking when appropriate.

Deadline: Thursday, November 16 at Midnight

Part 1 - Setup

Begin by setting up Spark and fetching the project data.

Note: you may want to use a larger EC2 instance type than normal. This project was prepared using a `t2.xlarge` instance. Just remember that the larger the instance, the higher the per-hour charge, so be sure to remember to shut your instance down when you're done, as always.

About the data

We will use JSON data from Twitter; we saw an example of this in class. It should parse cleanly, allowing you to focus on analysis.

This data was gathered using GWU Libraries' data sets during a game of the MLB World Series featuring the Los Angeles Dodgers and Houston Astros. This first file tells you a little bit about how it was gathered:

First make sure you are working from the right working directory

In [1]:
```
!pwd
```

```
/home/ubuntu/notebooks/Individual HW3
```

This below file provides context and detail information on the files we are going to work with

in this assignment

In [2]:
```
!wget https://s3.amazonaws.com/2017-dmfa/project-3/9670f3399f774789b7c3e18975d25611-RE
```

```
--2023-11-16 17:44:50--  https://s3.amazonaws.com/2017-dmfa/project-3/9670f3399f77478
9b7c3e18975d25611-README.txt
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.217.84.230, 52.217.126.120, 54.23
1.132.24, ...
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.217.84.230|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1920 (1.9K) [text/plain]
Saving to: ‘9670f3399f774789b7c3e18975d25611-README.txt.5’

9670f3399f774789b7c 100%[===================>]   1.88K  --.-KB/s    in 0s

2023-11-16 17:44:50 (90.9 MB/s) - ‘9670f3399f774789b7c3e18975d25611-README.txt.5’ sav
ed [1920/1920]
```

In [3]:
```
!cat 9670f3399f774789b7c3e18975d25611-README.txt
```

This is an export created with Social Feed Manager.

EXPORT INFORMATION
Selected seeds: All seeds
Export id: 9670f3399f774789b7c3e18975d25611
Export type: twitter_filter
Format: Full JSON
Export completed: Oct. 30, 2017, 11:21:04 p.m. EDT
Deduplicate: No

COLLECTION INFORMATION
Collection name: test set for world series
Collection id: 34e3f7460b5c4df09d64a1e61fd81238
Collection set: mlb-test (collection set id d6e8c27b1bc942e78790aa55a82b3a7a)
Harvest type: Twitter filter
Collection description: running for just one hour, just for fun.

Harvest options:
Media: No
Web resources: No

Seeds:
* Track: dodgers,astros - Active

Change log:

Change to test set for world series (collection) on Oct. 30, 2017, 10:59:56 p.m. EDT by dchud:
* is_active: "True" changed to "False"

Change to test set for world series (collection) on Oct. 30, 2017, 10:58:51 p.m. EDT by dchud:
* is_on: "True" changed to "False"

Change to test set for world series (collection) on Oct. 29, 2017, 8:01:24 p.m. EDT by dchud:
* is_on: "False" changed to "True"

Change to Track: dodgers,astros (seed) on Oct. 29, 2017, 8:01:21 p.m. EDT by dchud:
* token: "blank" changed to "{"track": "dodgers,astros"}"
* is_active: "blank" changed to "True"

Change to test set for world series (collection) on Oct. 29, 2017, 8:01:06 p.m. EDT by dchud:
* credential: "blank" changed to "dchud's twitter credential"
* harvest_type: "blank" changed to "twitter_filter"
* is_on: "blank" changed to "False"
* end_date: "blank" changed to "Oct. 31, 2017, 8:00:38 p.m. EDT"
* description: "blank" changed to "running for just one hour, just for fun."
* collection_set: "blank" changed to "mlb-test"
* is_active: "blank" changed to "True"
* visibility: "blank" changed to "default"
* harvest_options: "blank" changed to "{"media": false, "web_resources": false}"
* name: "blank" changed to "test set for world series"

Created on Oct. 30, 2017, 11:21:04 p.m. EDT

The most important pieces in that metadata are:

- It tracked tweets that mentioned "dodgers" or "astros". Every item in this set should refer to one or the other, or both.
- This data was not deduplicated; we may see individual items more than once.
- Data was collected between October 29 and October 30. Game 5 of the Series was played during this time.

You should not need to know anything about baseball to complete this assignment.

Please note: sometimes social media data contains offensive material. This data set has not been filtered; if you do come across something inappropriate, please do your best to ignore it if you can.

Fetch the data

The following files are available:

- https://s3.amazonaws.com/2017-dmfa/project-3/9670f3399f774789b7c3e18975d25611_003.json
- https://s3.amazonaws.com/2017-dmfa/project-3/9670f3399f774789b7c3e18975d25611_006.json

Q1.1 - Upload the above files to your instance using `wget`. Verify the file sizes using the command line.

Each file should contain exactly 100,000 tweets.

Note: you are required to use all files. It will be easier to process more data if you use a larger EC2 instance type, as suggested above. Use the exact same set of files throughout the assignment.

Answer

In [4]:
```
!wget https://s3.amazonaws.com/2017-dmfa/project-3/9670f3399f774789b7c3e18975d25611_00
```

```
--2023-11-16 17:44:50--  https://s3.amazonaws.com/2017-dmfa/project-3/9670f3399f77478
9b7c3e18975d25611_003.json
Resolving s3.amazonaws.com (s3.amazonaws.com)... 16.182.103.112, 16.182.101.80, 16.18
2.97.16, ...
Connecting to s3.amazonaws.com (s3.amazonaws.com)|16.182.103.112|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 595711407 (568M) [application/json]
Saving to: ‘9670f3399f774789b7c3e18975d25611_003.json.5’

9670f3399f774789b7c 100%[===================>] 568.11M  64.3MB/s    in 9.3s

2023-11-16 17:45:00 (61.0 MB/s) - ‘9670f3399f774789b7c3e18975d25611_003.json.5’ saved
[595711407/595711407]
```

In [5]:
```
!wget https://s3.amazonaws.com/2017-dmfa/project-3/9670f3399f774789b7c3e18975d25611_00
```

```
--2023-11-16 17:45:00--  https://s3.amazonaws.com/2017-dmfa/project-3/9670f3399f77478
9b7c3e18975d25611_006.json
Resolving s3.amazonaws.com (s3.amazonaws.com)... 54.231.232.208, 16.182.64.112, 52.21
6.38.224, ...
Connecting to s3.amazonaws.com (s3.amazonaws.com)|54.231.232.208|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 545081593 (520M) [application/json]
Saving to: ‘9670f3399f774789b7c3e18975d25611_006.json.5’

9670f3399f774789b7c 100%[===================>] 519.83M  56.0MB/s    in 9.0s

2023-11-16 17:45:09 (57.8 MB/s) - ‘9670f3399f774789b7c3e18975d25611_006.json.5’ saved
[545081593/545081593]
```

Check the files have exactly 100,000 tweets using the command line

In [6]:
```
!wc -l 9670f3399f774789b7c3e18975d25611_003.json
```

```
100000 9670f3399f774789b7c3e18975d25611_003.json
```

In [7]:
```
!wc -l 9670f3399f774789b7c3e18975d25611_006.json
```

```
100000 9670f3399f774789b7c3e18975d25611_006.json
```

One can see that both files' word counts verify that each has exactly 100,000 lines, i.e., tweets

For your reference, here is the text of one Tweet, randomly selected from one of these files. You might wish to study its structure and refer to it later.

In [8]:
```
!cat *.json | shuf -n 1 > example-tweet.json
```

In [9]:
```
import json
print(json.dumps(json.load(open("example-tweet.json")), indent=2))
```

```
{
  "quote_count": 0,
  "contributors": null,
  "truncated": false,
  "text": "@LA_Miike When he opts out after next year and dodgers invest that money i
nto someone who shows up in big games&gt;&gt;&gt;&gt;",
  "is_quote_status": false,
  "in_reply_to_status_id": 924824910181969920,
  "reply_count": 0,
  "id": 924825299426168832,
  "favorite_count": 0,
  "entities": {
    "user_mentions": [
      {
        "id": 905124128,
        "indices": [
          0,
          9
        ],
        "id_str": "905124128",
        "screen_name": "LA_Miike",
        "name": "MIKE"
      }
    ],
    "symbols": [],
    "hashtags": [],
    "urls": []
  },
  "retweeted": false,
  "coordinates": null,
  "timestamp_ms": "1509330508463",
  "source": "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter
for iPhone</a>",
  "in_reply_to_screen_name": "LA_Miike",
  "id_str": "924825299426168832",
  "display_text_range": [
    10,
    127
  ],
  "retweet_count": 0,
  "in_reply_to_user_id": 905124128,
  "favorited": false,
  "user": {
    "follow_request_sent": null,
    "profile_use_background_image": true,
    "default_profile_image": false,
    "id": 735783083559903232,
    "default_profile": true,
    "verified": false,
    "profile_image_url_https": "https://pbs.twimg.com/profile_images/9168995631427502
09/nb81U8uo_normal.jpg",
    "profile_sidebar_fill_color": "DDEEF6",
    "profile_text_color": "333333",
    "followers_count": 239,
    "profile_sidebar_border_color": "C0DEED",
    "id_str": "735783083559903232",
    "profile_background_color": "F5F8FA",
```

```
    "listed_count": 3,
    "profile_background_image_url_https": "",
    "utc_offset": -25200,
    "statuses_count": 6439,
    "description": "Intersectional Liberal Imperialist. Cosmopolitan Patriot. Soros f
unded. Undergrad at Frankfurt School of Cultural Marxism.",
    "friends_count": 669,
    "location": "South Cal",
    "profile_link_color": "1DA1F2",
    "profile_image_url": "http://pbs.twimg.com/profile_images/916899563142750209/nb81
U8uo_normal.jpg",
    "following": null,
    "geo_enabled": false,
    "profile_banner_url": "https://pbs.twimg.com/profile_banners/735783083559903232/1
504941323",
    "profile_background_image_url": "",
    "name": "The North Remembers",
    "lang": "en",
    "profile_background_tile": false,
    "favourites_count": 2356,
    "screen_name": "SerellThuggs",
    "notifications": null,
    "url": "http://opensocietyfoundations.org",
    "created_at": "Thu May 26 10:42:12 +0000 2016",
    "contributors_enabled": false,
    "time_zone": "Pacific Time (US & Canada)",
    "protected": false,
    "translator_type": "none",
    "is_translator": false
  },
  "geo": null,
  "in_reply_to_user_id_str": "905124128",
  "lang": "en",
  "created_at": "Mon Oct 30 02:28:28 +0000 2017",
  "filter_level": "low",
  "in_reply_to_status_id_str": "924824910181969920",
  "place": null
}
```


You can find several key elements in this example; the text, time, and language of the tweet, whether it was a reply to another user, the user's screen name along with their primary language and other account information like creation date, follower/friend/tweet counts, and perhaps their location.

If there are hashtags, user mentions, or urls present in their tweet, they will be present in the `entities` section; these are not present in every tweet. If this is a retweet, you will see the original tweet and its information nested within.

Q1.2 - Start up Spark, and verify the file sizes.

We will use our normal startup sequence here:

In [10]:
```
import os
```

In [11]:
```
os.environ['SPARK_HOME'] = '/usr/local/lib/spark'
```

In [12]:
```
import findspark
```

In [13]:
```
findspark.init()
```

In [14]:
```
from pyspark import SparkContext
```

In [15]:
```
spark = SparkContext(appName='project-03')
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(new
Level).
23/11/16 17:45:31 WARN NativeCodeLoader: Unable to load native-hadoop library for you
r platform... using builtin-java classes where applicable
```

In [16]:
```
spark
```

Out[16]:

SparkContext

Spark UI

Version	`v3.3.1`
Master	`local[*]`
AppName	`project-03`

In [17]:
```
from pyspark import SQLContext
```

In [18]:
```
sqlc = SQLContext(spark)
```

```
/usr/local/lib/spark/python/pyspark/sql/context.py:112: FutureWarning: Deprecated in
3.0.0. Use SparkSession.builder.getOrCreate() instead.
  warnings.warn(
```

In [19]:
```
sqlc
```

Out[19]:
```
<pyspark.sql.context.SQLContext at 0x7f522ecada60>
```

In [20]:
```
tweets = sqlc.read.json("*.json")
```

```
23/11/16 17:45:48 WARN package: Truncated the string representation of a plan since i
t was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStrin
gFields'.
```

Verify that Spark has loaded the same number of tweets you saw before:

Answer

In [21]:
```
tweets.count()
```

Out[21]:
```
200001
```

Do you see exactly the same number of tweets in Spark that you saw on the command line?

We see 200001, which is the two files of 100000 plus the example-tweet.json file -- resulting in 100000+100000+1=200001. Because the example-tweet.json file came from one of the two files, we don't want its duplicate to affect our analysis. For this reason, I will delete the example-tweet.json file in the next block of code.

In [22]:
```
!rm example-tweet.json
```

The file is now removed. Now, when we look at tweets.count(), we will see that we get exactly 200000 tweets (the sum of the two main files), as expected.

In [23]:
```
tweets = sqlc.read.json("*.json")
```

In [24]:
```
tweets.count()
```

Out[24]:
```
200000
```

Part 2 - Comparing DataFrames and Spark SQL

For the next three questions, we will look at operations using both DataFrames and SQL queries. Note that `tweets` is already a DataFrame:

To issue SQL queries, we need to register a table based on `tweets`:

This is all well and good, but how well did schema inference work?

In [25]:
```
tweets.createOrReplaceTempView("tweets")
```

Q2.1 - Which 6 languages are most commonly used in tweets? Verify your result by executing it with both the dataframe and with SQL.

Hint: for the dataframe, use `groupBy`, `count`, and `orderBy`. See the documentation at https://spark.apache.org/docs/2.2.0/api/python/pyspark.sql.html for details on these and other functions.

Answer

In [26]:
```
tweets.groupBy("lang").count().orderBy("count", ascending=False).show(6)
```

```
[Stage 8:=================================================>      (8 + 1) / 9]
```

```
+----+------+
|lang| count|
+----+------+
|  en|173906|
|  es| 15666|
| und|  7579|
|  in|   483|
|  pt|   479|
|  fr|   435|
+----+------+
only showing top 6 rows
```

In [45]:
```
sqlc.sql("""
    SELECT lang as Language_Code, count(*) as Count
    FROM tweets
    group by lang
    order by count(*) desc
""").show(6)
```

```
[Stage 59:=================================================>      (8 + 1) / 9]
+-------------+------+
|Language_Code| Count|
+-------------+------+
|           en|173906|
|           es| 15666|
|          und|  7579|
|           in|   483|
|           pt|   479|
|           fr|   435|
+-------------+------+
only showing top 6 rows
```

The count for English is significantly higher than all the rest of the languages. This makes sense, considering that the MLB is a North American league, and in North America, the main language is English.

Note: I considered using user.lang, however, I determined that lang made more sense, as user.lang refers to the user's profile as a whole, but lang refers to the specific tweet itself.

Also note: und means undetermined. For this reason, I re-ran the SQL code without 'und' in order to get a real list of 6 languages.

In [28]:
```
sqlc.sql("""
    SELECT lang as Language_Code, count(*) as Count
    FROM tweets
    WHERE lang!='und'
    group by lang
    order by count(*) desc
""").show(6)
```

```
[Stage 14:=====================================>                  (6 + 3) / 9]
```

```
+-------------+------+
|Language_Code| Count|
+-------------+------+
|           en|173906|
|           es| 15666|
|           in|   483|
|           pt|   479|
|           fr|   435|
|           ht|   201|
+-------------+------+
only showing top 6 rows
```

Q2.2 - Which 5 time zones are most common among users? Verify your result with both the dataframe and SQL.

Note: for this question, you may leave NULL values present in your results, as a way to help you understand what data is present and what is missing.

Answer

In [29]:
```
tweets.groupBy("user.time_zone").count().orderBy("count", ascending=False).show(5)
```

```
[Stage 17:=====================================================>      (8 + 1) / 9]
+--------------------+-----+
|           time_zone|count|
+--------------------+-----+
|                null|84166|
|Pacific Time (US ...|35816|
|Central Time (US ...|32031|
|Eastern Time (US ...|17780|
|             Arizona| 5199|
+--------------------+-----+
only showing top 5 rows
```

In [30]:
```
sqlc.sql("""
    SELECT user.time_zone as Time_Zone, count(*) as Count FROM tweets
    group by user.time_zone
    order by count(*) desc
""").show(5)
```

```
[Stage 20:=======================================>                   (6 + 3) / 9]
```

```
+--------------------+-----+
|           Time_Zone|Count|
+--------------------+-----+
|                null|84166|
|Pacific Time (US ...|35816|
|Central Time (US ...|32031|
|Eastern Time (US ...|17780|
|             Arizona| 5199|
+--------------------+-----+
only showing top 5 rows
```

Write your observations here.

It looks like the top 5 time zones are Pacific Time, Central Time, Eastern Time, and Arizona. However, Arizona is not a time zone, and null is in the top spot, and thus there must be an error in the data. For this reason, I re-ran the SQL code below, without 'Arizona' and without 'null'. That 4th and 5th time zone, seen below, are Mountain Time and Atlantic Time.

In [31]:
```
sqlc.sql("""
    SELECT user.time_zone as Time_Zone, count(*) as Count FROM tweets
    WHERE user.time_zone IS NOT NULL AND user.time_zone!='Arizona'
    group by user.time_zone
    order by count(*) desc
""").show(5)
```

```
[Stage 23:=============================================================>      (8 + 1) / 9]
+--------------------+-----+
|           Time_Zone|Count|
+--------------------+-----+
|Pacific Time (US ...|35816|
|Central Time (US ...|32031|
|Eastern Time (US ...|17780|
|Mountain Time (US...| 4700|
|Atlantic Time (Ca...| 2383|
+--------------------+-----+
only showing top 5 rows
```

These results are as I would expect, considering the fact that the LA Dodgers' LA time zone is Pacific Time, and the Houston Astros' Houston time zone is Central Time. However, what is interesting, as we will see in the following problems, is that even though Pacific Time is the most common for users, the Dodgers are mentioned less than the Astros. So, either a lot of people in Pacific Time are talking badly about the Astros, or there are a lot of Astros fans in Pacific Time. It is possible that some people in Pacific Time are Astros fans, considering that Pacific Time covers California, Washington, Oregon, Nevada, parts of Idaho, parts of Arizona, and parts of Utah. For this reason, the time zone seems to not be an indicator of which team is talked about more or less.

One other thing to point out is the fact that Spanish is the second-most popular language in this database of tweets. Spanish is a heavily used language in southern states of the US, and Central Time covers a decent number of southern states. Again, this follows from the prior code indicating the most-heavily used languages, as well as this code, indicating the most-dominant time zones.

Q2.3 - How many tweets mention the Dodgers? How many mention the Astros? How many mention both?

You may use either the dataframe or SQL to answer. Explain why you have chosen that approach.

Hint: you will want to look at the value of the `text` field.

Answer

In [32]:
```
sqlc.sql("""
    SELECT
    SUM(CASE WHEN LOWER(text) LIKE '%dodger%' AND LOWER(text) NOT LIKE '%astro%' THEN
    SUM(CASE WHEN LOWER(text) LIKE '%astro%' AND LOWER(text) NOT LIKE '%dodger%' THEN
    SUM(CASE WHEN LOWER(text) LIKE '%dodger%astro%' OR LOWER(text) LIKE '%astro%dodger
    FROM tweets
""").show(5)
```

```
[Stage 26:=============================================================>      (8 + 1) / 9]
+-----------------+----------------+--------------------+
|Dodgers_Mentioned|Astros_Mentioned|Both_Teams_Mentioned|
+-----------------+----------------+--------------------+
|            54527|          102741|               24535|
+-----------------+----------------+--------------------+
```

We can see that the dodgers are mentioned 54,527 times, the astros are mentioned 102,741 times, and they are both mentioned together 24,535 times. It is interesting that the astros are mentioned nearly 2x the amount of times as the dodgers, and the dodgers are mentioned nearly 2x the amount as a tweet where they are both mentioned. Considering that the Astros won the World Series in 2017, it makes sense that they were mentioned so much more.

However, we must note that this only adds up to 181,803 tweets. In the below code, I investigate what happened to the remaining 18,197 tweets:

In [33]:
```
sqlc.sql("""
    SELECT text
    FROM tweets
    WHERE LOWER(text) NOT LIKE '%dodger%astro%' AND LOWER(text) NOT LIKE '%astro%dodge
    AND LOWER(text) NOT LIKE '%dodger%' AND LOWER(text) NOT LIKE '%astro%'
""").show(5, False)
```

```
+--------------------------------------------------------------------------------
--------------------------------------------------------+
|text
|
+--------------------------------------------------------------------------------
--------------------------------------------------------+
|RT @SeanTPendergast: DEAL https://t.co/MtcRbtuxhc
|
|Oh, yeah, shocked......I'm not sure shocked is what I'd call it. It's really more of
a duh feeling. https://t.co/aU4jAVpihl                 |
|RT @TeamCJCorrea: I can't recall doing that 😂😂😂 https://t.co/qhpOQoIIon
|
|#HR4HR https://t.co/XVxLF6Tomr
|
|RT @AmandaRTubbs: "Hi, I'm Adrian Gonzalez. We've had a lot of fun this year, but
I'm here to talk about a serious issue: switching to robo…|
+--------------------------------------------------------------------------------
--------------------------------------------------------+
only showing top 5 rows
```

In [34]:
```
sqlc.sql("""
    SELECT
    SUM(CASE WHEN LOWER(text) NOT LIKE '%dodger%' AND (LOWER(text) LIKE '%rt%' OR LOWE
    FROM tweets
""").show()
```

```
[Stage 30:=============================================================>      (8 + 1) / 9]
+----------------------------+
|Retweets_Without_Team_In_Text|
+----------------------------+
|                       18197|
+----------------------------+
```

In the above tweets, it looks like they all are retweets of tweets that mentioned one or both of the teams. However, the text itself does not include 'Astros' or 'Dodgers' and thus these are left to their own category. As we can see, again confirmed above, the 18,197 tweets that are missing from the tweet count either say 'RT', i.e., retweet, or say 'https://t.co', indicating that it is linking to a prior tweet. In the prior tweet, one of the teams is mentioned.

Part 3 - More complex queries

For this section, you may choose to use dataframe queries or SQL. If you wish, you may verify results by using both, as in Part 2, but this is not required for this section.

Q3.1 - Team mentions by location

In which users' locations are the Astros and the Dodgers being mentioned the most? Consider each team separately, one at a time. Discuss your findings. Do not count null time_zones or location.

Hint: you may use either the time zones or user-specified locations for this question.

Answer

In [35]:
```
sqlc.sql("""
    SELECT user.location as Location, count(*) as Count
    FROM tweets
    WHERE LOWER(text) LIKE '%dodger%'
    AND user.location IS NOT NULL
    GROUP BY user.location
    ORDER BY count(user.location) DESC
    """).show(10)
```

```
[Stage 33:=============================================>      (8 + 1) / 9]
+--------------+-----+
|       Location|Count|
+--------------+-----+
|Los Angeles, CA| 2905|
|California, USA| 1275|
|    Houston, TX| 1201|
|    Los Angeles|  999|
|  United States|  812|
|     Texas, USA|  393|
|     California|  383|
|      Venezuela|  362|
|         México|  306|
|  Las Vegas, NV|  272|
+--------------+-----+
only showing top 10 rows
```

In [36]:
```
sqlc.sql("""
    SELECT user.location as Location, count(*) as Count
    FROM tweets
    WHERE LOWER(text) LIKE '%astro%'
    AND user.location IS NOT NULL
    GROUP BY user.location
    ORDER BY count(user.location) DESC
    """).show(10)
```

```
[Stage 36:=============================================>      (8 + 1) / 9]
```

```
+---------------+-----+
|       Location|Count|
+---------------+-----+
|    Houston, TX| 9500|
|     Texas, USA| 2219|
|          Texas| 1311|
|Los Angeles, CA| 1271|
|        Houston| 1153|
|  United States|  971|
| Houston, Texas|  960|
|     Austin, TX|  734|
|California, USA|  672|
|San Antonio, TX|  650|
+---------------+-----+
only showing top 10 rows
```

As we can see in the above codes, the Dodgers are most talked about in Los Angeles, CA and the Astros are most talked about in Houston, TX. This makes sense, considering the fact that the Dodgers are LA's team and the Astros are Houston's team. We can, again, see some issues with the data -- notice that in the 'astros' code, Houston, TX appears in the output and also Houston appears in the output. In reality, the two should be combined. This is the same with Texas, USA vs. Houston, TX -- they should be more consistent as city, state across all outputs. It also makes sense that Venezuela appeared in the output for 'dodgers', considering that baseball is a very popular sport in Venezuela.

Additionally, this is following the same trends we've seen in the prior codes -- Astros are talked about more than the Dodgers, which is evident in the Astros' location, Houston, in this code as well.

Q3.2 - Which Twitter users are being replied to the most?

Discuss your findings.

Hint: use the top-level `in_reply_to_screen_name` for this.

Answer

In [37]:
```
sqlc.sql("""
    SELECT in_reply_to_screen_name as Most_Replied_To_User, count(in_reply_to_screen_n
    FROM tweets
    GROUP BY in_reply_to_screen_name
    ORDER BY count(in_reply_to_screen_name) DESC
    """).show(5)
```

```
[Stage 39:=================================================>      (8 + 1) / 9]
```

```
+-------------------+-----+
|Most_Replied_To_User|Count|
+-------------------+-----+
|              astros| 2061|
|             Dodgers| 1569|
|                 MLB|  464|
|            MLBONFOX|  113|
|       stephenasmith|  109|
+-------------------+-----+
only showing top 5 rows
```

We can see that the account that is replied to the most is the Astros, and the next most is the Dodgers then MLB. This all makes sense, as Twitter users are likely replying to game news tweets or highlight tweets to support or go against the team/post. Also, it makes sense that Stephen A Smith appears in this list, as he is a Sports TV Media Personality.

Q3.3 - Which 12 verified users have the most followers? Which 12 unverified users have the most followers?

Provide both the screen names (screen_name) and follower counts (followers_count) for each. Verified users -- use verified == 't'

Discuss your findings.

Answer

In [38]:
```
sqlc.sql("""
    SELECT
    user.screen_name as Most_Followed_Screen_Names_Verified,
    FIRST(user.followers_count) AS Followers
    FROM tweets
    WHERE user.verified = 't'
    GROUP BY user.screen_name
    ORDER BY Followers DESC
    """).show(12)
```

```
[Stage 42:=============================================>           (7 + 2) / 9]
```

```
+----------------------------------+---------+
|Most_Followed_Screen_Names_Verified|Followers|
+----------------------------------+---------+
|                           Reuters| 18937529|
|                           FoxNews| 16272836|
|                               ABC| 12551437|
|                     washingtonpost| 11417638|
|                               MLB|  7841255|
|                               NPR|  7289492|
|                       BillSimmons|  6000106|
|                         Residente|  5856019|
|                           NBCNews|  5442705|
|                        JohnLegere|  4630104|
|                         ANCALERTS|  4453229|
|                           Milenio|  4007212|
+----------------------------------+---------+
only showing top 12 rows
```

In [39]:
```
sqlc.sql("""
    SELECT
    user.screen_name as Most_Followed_Screen_Names_Unverified,
    FIRST(user.followers_count) AS Followers
    FROM tweets
    WHERE user.verified = 'f'
    GROUP BY user.screen_name
    ORDER BY Followers DESC
    """).show(12)
```

```
[Stage 45:=====================================================>      (8 + 1) / 9]
+------------------------------------+---------+
|Most_Followed_Screen_Names_Unverified|Followers|
+------------------------------------+---------+
|                      Daminous_Purity|   998742|
|                              chochos|   833669|
|                        el_carabobeno|   725952|
|                             PAMsLOvE|   712254|
|                       mlbtraderumors|   659851|
|                              jilevin|   568341|
|                         sun_das_ill|   559669|
|                        DiegoArcos14|   544915|
|                         TVCDeportes|   543095|
|                            EP_Mundo|   538525|
|                              LALATE|   516139|
|                        piercearrow33|   503015|
+------------------------------------+---------+
only showing top 12 rows
```

It is important to note that without the FIRST() clause, the screen_names can repeat. For example, MLB repeated multiple times -- this is due to the fact that at each tweet, the followers_count changes, as people follow and unfollow over time.

We can see that, of the verified accounts, Reuters has the most followers. Of the unverified accounts, Daminous_Purity has the most followers. Many of the verified accounts' screen_names are ones that many may recognize, naturally. The same cannot be said for the unverigied accounts.

Q4 - Analyze common words in tweet text

Following the example in class, use `tweets.rdd` to find the most common interesting words in tweet text. To keep it "interesting", remove at least 12 common stop words found in tweets, like "a", "an", "the", and "RT" (you might want to derive these stop words from initial results). A simple split on text whitespace like we had in class is sufficient; you do not have to account for punctuation.

After you find the most common words, use dataframe or SQL queries to find patterns among how those words are used. For example, are they more frequently used by Dodgers or Astros fans, or by people in one part of the country over another? Explore and see what you can find, and discuss your findings.

You will notice that common words include words like "thisteam" and "earnhistory". I would like you to write two queries to investigate whether those two words are used by the Astros or Dodgers

Hint: don't forget all the word count pipeline steps we used earlier in class.

Answer

In [40]:
```
tweets_sample = tweets.sample(False, 0.01, 12345)
```

In [41]:
```
filter_out = ['RT','the','to','Astros','a','','@astros','in','#WorldSeries','Dodgers',
              ,'@astros:','I','is','of','this','and','The','on','that','it','@Dodgers'
             , '@Dodgers:' '#Dodgers', 'are', 'A', 'was', 'you', 'THE', 'have', 'out',
             , 'one', 'game', 'This', 'win', 'Series']

tweets_sample.rdd.flatMap(lambda r: r['text'].split(' ')) \
    .map(lambda t: (t, 1)) \
    .reduceByKey(lambda a, b: a + b) \
    .filter(lambda pair: pair[0] not in filter_out) \
    .takeOrdered(15, key=lambda pair: -pair[1])
```

Out[41]:
```
[('#EarnHistory', 146),
 ('@Dodgers:', 108),
 ('#Dodgers', 96),
 ('World', 85),
 ('Game', 74),
 ('GAME', 72),
 ("LET'S", 70),
 ('#ThisTeam', 68),
 ('just', 67),
 ('with', 66),
 ('not', 65),
 ('come', 65),
 ('dodgers', 64),
 ('5', 64),
 ('TIE', 63)]
```

We can see that #EarnHistory and #ThisTeam are used quite a bit in tweets. I will now write two queries to investigate whether those two words are used by the Astros or Dodgers.

In [42]:
```
sqlc.sql("""
    SELECT
    SUM(CASE WHEN LOWER(text) LIKE '%dodger%' AND LOWER(text) LIKE '%#earnhistory%' AN
    SUM(CASE WHEN LOWER(text) LIKE '%astro%' AND LOWER(text) LIKE '%#earnhistory%' AND
    SUM(CASE WHEN LOWER(text) LIKE '%dodger%' AND LOWER(text) LIKE '%#earnhistory%' AN
    FROM tweets
""").show(5)
```

```
[Stage 50:=====================================================>      (8 + 1) / 9]
+---------------------------+--------------------------+-------------------------+
|Dodgers_earnhistory_mentions|Astros_earnhistory_mentions|Both_earnhistory_mentions|
+---------------------------+--------------------------+-------------------------+
|                        175|                     20818|                     1210|
+---------------------------+--------------------------+-------------------------+
```

In [43]:
```
sqlc.sql("""
    SELECT
    SUM(CASE WHEN LOWER(text) LIKE '%dodger%' AND LOWER(text) LIKE '%#thisteam%' AND L
    SUM(CASE WHEN LOWER(text) LIKE '%astro%' AND LOWER(text) LIKE '%#thisteam%' AND LO
    SUM(CASE WHEN LOWER(text) LIKE '%dodger%' AND LOWER(text) LIKE '%#thisteam%' AND L
    FROM tweets
""").show(5)
```

```
[Stage 53:=====================================================>      (8 + 1) / 9]
+-----------------------+----------------------+---------------------+
|Dodgers_thisteam_mentions|Astros_thisteam_mentions|Both_thisteam_mentions|
+-----------------------+----------------------+---------------------+
|                  10183|                   426|                 1071|
+-----------------------+----------------------+---------------------+
```

Based on the above codes, it is clear that the #EarnHistory hashtag is most often used by people mentioning the Astros. Additionally, it is clear that the #ThisTeam hashtag is most often

used by people mentioning the Dodgers.

After researching online, it seems that the '#ThisTeam' phrase was used by the Dodgers during the 2017 playoffs and World Series as a signification of spirit and unity. Additionally, it seems that the '#EarnHistory' phrase was used by the Astros during the 2017 playoffs and World Series as a phrase to encompass their journey to championship.