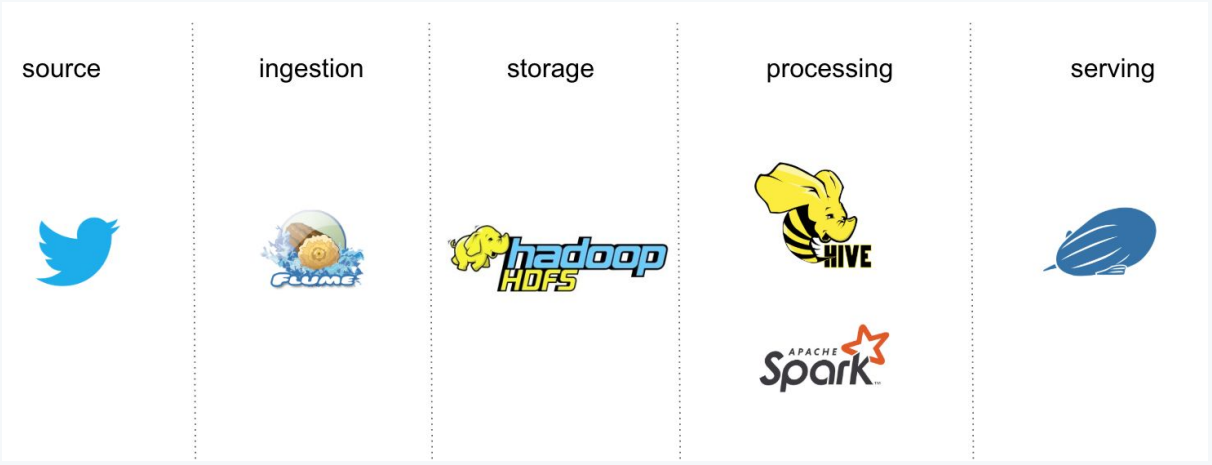
Hadoop Team Assignment

You have joined an inversion startup that is working on a project to predict cryptocurrencies market movements based on social media sentiment analysis.

Business analysts have chosen the following cryptocurrencies to start with:

team	cryptocurrency	ticker	cashtag	logo
A	bitcoin	BTC	$BTC	
B	ethereum	ETH	$ETH	
C	ripple	XRP	$XRP	
D	cardano	ADA	$ADA	
E	litecoin	LTC	$LTC	

Before deciding the final data architecture, data engineering team has designed the following POC (proof of concept) data architecture to start analyzing twitter data:



You have available a dataset located in hdfs path /user/flume/sentiment-dictionary/

The dataset is a tab delimited file containing english words (in lower case) with their sentiment polarity. It has the following columns:

type:string
length:integer
word:string
word_type:string
stemmed:string
polarity:string

This is a sample of the contents

weaksubj	1	abandoned	adj	n	negative
weaksubj	1	abandonment	noun	n	negative
weaksubj	1	abandon verb	y	negative	
strongsubj	1	abase verb	y	negative	
strongsubj	1	abasement	anypos	y	negative
strongsubj	1	abash verb	y	negative	
weaksubj	1	abate verb	y	negative	

The engineering team is currently ingesting cryptocurrency related tweets into hdfs directory /user/flume/crypto-tweets/.

Your task is to start analyzing the available data files in json format containing the tweets (one per line). The following is a sample tweet so that you can understand its data structure:


```
{
  "contributors": null,
  "coordinates": null,
  "created_at": "Sun Nov 18 20:19:30 +0000 2018",
  "entities": {
    "hashtags": [{"indices": [109,117], "text": "Bitcoin"} ], "symbols": [],
    "urls": [{"display_url": "bit.ly/2ON1Mln", "expanded_url": "http://bit.ly/2ON1Mln", "indices": [85,108], "url": "https://t.co/oGkQDBQsH7"} ],
    "user_mentions": []
  },
  "favorite_count": 0,
  "favorited": false,
  "filter_level": "low",
  "geo": null,
  "id": 1064251774738067461,
  "id_str": "1064251774738067461",
  "in_reply_to_screen_name": null,
  "in_reply_to_status_id": null,
  "in_reply_to_status_id_str": null,
  "in_reply_to_user_id": null,
  "in_reply_to_user_id_str": null,
  "is_quote_status": false,
  "lang": "en",
  "place": null,
  "possibly_sensitive": false,
  "quote_count": 0,
  "reply_count": 0,
  "retweet_count": 0,
  "retweeted": false,
  "source": "<a href=\"https://ifttt.com\" rel=\"nofollow\">IFTTT</a>",
  "text": "In 2017 Bitcoin Went From $5.5k to $19k in 33 Days, Not Impossible in 2019 - newsBTC https://t.co/oGkQDBQsH7 #Bitcoin",
  "timestamp_ms": "1542572370596",
  "truncated": false,
  "user": {
    "contributors_enabled": false,
    "created_at": "Sat Dec 25 12:12:16 +0000 2010",
    "default_profile": false,
    "default_profile_image": false,
    "description": "Who is love Computer. #SKIDDOW #CyberSecurity #HackedBy #Anonymous News feed. \n#Bitcoin News",
    "favourites_count": 528,
    "follow_request_sent": null,
    "followers_count": 221,
    "following": null,
    "friends_count": 33,
    "geo_enabled": false,
    "id": 230423908,
    "id_str": "230423908",
    "is_translator": false,
    "lang": "en",
    "listed_count": 177,
    "location": null,
    "name": "SKIDDOW",
    "notifications": null,
    "profile_background_color": "000000",
    "profile_background_image_url": "http://abs.twimg.com/images/themes/theme5/bg.gif",
    "profile_background_image_url_https": "https://abs.twimg.com/images/themes/theme5/bg.gif",
    "profile_background_tile": false,
    "profile_banner_url": "https://pbs.twimg.com/profile_banners/230423908/1462000239",
    "profile_image_url": "http://pbs.twimg.com/profile_images/626267268448522240/H_9Mhamn_normal.png",
    "profile_image_url_https": "https://pbs.twimg.com/profile_images/626267268448522240/H_9Mhamn_normal.png",
    "profile_link_color": "9266CC",
    "profile_sidebar_border_color": "000000",
    "profile_sidebar_fill_color": "000000",
    "profile_text_color": "000000",
    "profile_use_background_image": false,
    "protected": false,
    "screen_name": "SKIDDOW_KIDDO",
    "statuses_count": 63795,
    "time_zone": null,
    "translator_type": "none",
    "url": "https://www.skiddow.net",
    "utc_offset": null,
    "verified": false
  }
}
```

You have additional information about tweet json structure in the following links:

Tweet JSON Introduction

Tweet JSON Object Documentation

Each team should deliver a text file with all the required statements.

Due date: Sunday 2019/07/21 23:59:59

Please zip the file with the name of the team (team-a.zip) and upload it to blackboard before due date.

TEAM A

1. create a database named **crypto_team_a**.

2. select the database you just created so that all the tables you are going to create belong to that database.

3. create an external table named **sentiment_dictionary** over the file located in /user/flume/sentiment-dictionary/ .

4. create an external table named **tweets_json** over the files located in /user/flume/crypto-tweets/ .

   You don't need to reference all the fields in a tweet, just the ones to solve your assignment.
   You can use the table definition we saw during hive lab as a template (you will have to add and remove some fields).

5. write a query that returns the total number of tweets in table **tweets_json**.
   Annotate both the number of records and the amount of seconds that it took.

6. create a managed table **tweets_orc** with same schema as tweets_json but stored in orc format.
   (hint: create table … like)

7. insert all rows from **tweets_json** into **tweets_orc**.
   (hint: insert into ...)

8. write a query that returns the total number of tweets in table **tweets_orc**.
   Annotate both the number of records and the amount of seconds that it took.

9. verify that both tables contain the same number of tweets.
   which of the queries was faster?

10. write a query that returns the total number of users with geolocation enabled from table **tweets_orc**.

11. write a query that returns the total number of tweets per language from table **tweets_orc**.

12. write a query that returns the top 10 users with more tweets published from table **tweets_orc**.

13. write a query that returns the geoname latitude, longitude and timezone of the tweet place by joining **geonames** and **tweets_orc**.
    (hint: there can be places with the same name in different countries
    normalize (upper/lower case) the place name when joining)

14. write a query that returns the total count, total distinct count, maximum, minimum, average, standard deviation and percentiles 25th, 50th, 75th, 100th of **hashtags** in tweets from table **tweets_orc**

15. write a query that returns the top 10 more popular hashtags from table **tweets_orc**

16. create a table **tweet_words** in parquet format exploding the words in the tweets. Also normalize the words to lower case.
    (hint: use lateral view)

    Example

id	text
12345	“This a test”

id	word
12345	this
12345	is
12345	test

17. create a table **tweet_words_sentiment** in parquet format as the result of a query that returns the polarity of each word by left joining **tweet_words** with **sentiment_dictionary.** The polarity for non joining words will be neutral (you can use coalesce function). Also codify the polarity (you can use case when …) as integer in the following way:

positive ->1
neutral -> 0
negative -> -1

Example

id	word
12345	bad
12345	wewew

id	word	polarity
12345	bad	-1
12345	wewew	0

18. create a table **tweets_sentiment** in parquet format as the result of a query that sums the polarity of every tweet so that
    sum(polarity) > 0 -> ‘positive’
    sum(polarity) < 0 -> ‘negative’
    sum(polarity) = 0 -> ‘neutral’

    Example

id	word	polarity
12345	bad	-1
12345	wewew	0

id	polarity
12345	‘negative’

19. write a query that returns the hourly evolution of sentiment of tweets with hashtag BTC or bitcoin
    Example

hour	positive	negative
2019062522	1233	235
2019062523	2355	124

TEAM B

1. create a database named **crypto_team_b**.

2. select the database you just created so that all the tables you are going to create belong to that database.

3. create an external table named **sentiment_dictionary** over the file located in /user/flume/sentiment-dictionary/ .

4. create an external table named **tweets_json** over the files located in /user/flume/crypto-tweets/ .

   You don't need to reference all the fields in a tweet, just the ones to solve your assignment.
   You can use the table definition we saw during hive lab as a template (you will have to add and remove some fields).

5. write a query that returns the total number of tweets in table **tweets_json**.
   Annotate both the number of records and the amount of seconds that it took.

6. create a managed table **tweets_parquet** with same schema as tweets_json but stored in parquet format.
   (hint: create table … like)

7. insert all rows from **tweets_json** into **tweets_parquet**.
   (hint: insert into ...)

8. write a query that returns the total number of tweets in table **tweets_parquet**.
   Annotate both the number of records and the amount of seconds that it took.

9. verify that both tables contain the same number of tweets.
   which of the queries was faster?

10. write a query that returns the total number of users with geolocation enabled from table **tweets_parquet**.

11. write a query that returns the total number of tweets per language from table **tweets_parquet**.

12. write a query that returns the top 10 users with more followers from table **tweets_parquet**.

13. write a query that returns the geoname latitude, longitude and timezone of the tweet place by joining **geonames** and **tweets_parquet**.
    (hint: there can be places with the same name in different countries
    normalize (upper/lower case) the place name when joining)

14. write a query that returns the total count, total distinct count, maximum, minimum, average, standard deviation and percentiles 25th, 50th, 75th, 100th of **cashtags** in tweets from table **tweets_parquet**

15. write a query that returns the top 10 more popular **cashtags** from table **tweets_parquet**

16. create a table **tweet_words** in parquet format exploding the words in the tweets. Also normalize the words to lower case.
    (hint: use lateral view)

    Example

id	text
12345	“This a test”

id	word
12345	this
12345	is
12345	test

17. create a table **tweet_words_sentiment** in parquet format as the result of a query that returns the polarity of each word by left joining **tweet_words** with **sentiment_dictionary.** The polarity for non joining words will be neutral (you can use coalesce function). Also codify the polarity (you can use case when …) as integer in the following way:

positive ->1
neutral -> 0
negative -> -1

Example

id	word
12345	bad
12345	wewew

id	word	polarity
12345	bad	-1
12345	wewew	0

18. create a table **tweets_sentiment** in parquet format as the result of a query that sums the polarity of every tweet so that
 sum(polarity) > 0 -> ‘positive’
 sum(polarity) < 0 -> ‘negative’
 sum(polarity) = 0 -> ‘neutral’

Example

id	word	polarity
12345	bad	-1
12345	wewew	0

id	polarity
12345	‘negative’

19. write a query that returns the hourly evolution of sentiment of tweets with hashtag ETH or ethereum

Example

hour	positive	negative
2019062522	1233	235
2019062523	2355	124

TEAM C

1. create a database named **crypto_team_c**.

2. select the database you just created so that all the tables you are going to create belong to that database.

3. create an external table named **sentiment_dictionary** over the file located in /user/flume/sentiment-dictionary/ .

4. create an external table named **tweets_json** over the files located in /user/flume/crypto-tweets/ .

   You don't need to reference all the fields in a tweet, just the ones to solve your assignment.
   You can use the table definition we saw during hive lab as a template (you will have to add and remove some fields).

5. write a query that returns the total number of tweets in table **tweets_json**.
   Annotate both the number of records and the amount of seconds that it took.

6. create a managed table **tweets_orc** with same schema as tweets_json but stored in orc format.
   (hint: create table … like)

7. insert all rows from **tweets_json** into **tweets_orc**.
   (hint: insert into ...)

8. write a query that returns the total number of tweets in table **tweets_orc**.
   Annotate both the number of records and the amount of seconds that it took.

9. verify that both tables contain the same number of tweets.
   which of the queries was faster?

10. write a query that returns the total number of users with geolocation enabled from table **tweets_orc**.

11. write a query that returns the total number of tweets per language from table **tweets_orc**.

12. write a query that returns the top 10 users with more followers from table **tweets_orc**.

13. write a query that returns the geoname latitude, longitude and timezone of the tweet place by joining **geonames** and **tweets_orc**.
    (hint: there can be places with the same name in different countries
    normalize (upper/lower case) the place name when joining)

14. write a query that returns the total count, total distinct count, maximum, minimum, average, standard deviation and percentiles 25th, 50th, 75th, 100th of **user mentions** in tweets from table **tweets_orc**

15. write a query that returns the top 10 users more mentioned from table **tweets_orc**

16. create a table **tweet_words** in parquet format exploding the words in the tweets. Also normalize the words to lower case.
    (hint: use lateral view)

    Example

id	text
12345	“This a test”

id	word
12345	this
12345	is
12345	test

17. create a table **tweet_words_sentiment** in parquet format as the result of a query that returns the polarity of each word by left joining **tweet_words** with **sentiment_dictionary.** The polarity for non joining words will be neutral (you can use coalesce function). Also codify the polarity (you can use case when …) as integer in the following way:

positive ->1
neutral -> 0
negative -> -1

Example

id	word
12345	bad
12345	wewew

id	word	polarity
12345	bad	-1
12345	wewew	0

18. create a table **tweets_sentiment** in parquet format as the result of a query that sums the polarity of every tweet so that
 sum(polarity) > 0 -> ‘positive’
 sum(polarity) < 0 -> ‘negative’
 sum(polarity) = 0 -> ‘neutral’

Example

id	word	polarity
12345	bad	-1
12345	wewew	0

id	polarity
12345	‘negative’

19. write a query that returns the hourly evolution of sentiment of tweets with hashtag XRP or ripple

Example

hour	positive	negative
2019062522	1233	235
2019062523	2355	124

TEAM D

1. create a database named **crypto_team_d**.

2. select the database you just created so that all the tables you are going to create belong to that database.

3. create an external table named **sentiment_dictionary** over the file located in /user/flume/sentiment-dictionary/ .

4. create an external table named **tweets_json** over the files located in /user/flume/crypto-tweets/ .

   You don't need to reference all the fields in a tweet, just the ones to solve your assignment.
   You can use the table definition we saw during hive lab as a template (you will have to add and remove some fields).

5. write a query that returns the total number of tweets in table **tweets_json**.
   Annotate both the number of records and the amount of seconds that it took.

6. create a managed table **tweets_parquet** with same schema as tweets_json but stored in parquet format.
   (hint: create table … like)

7. insert all rows from **tweets_json** into **tweets_parquet**.
   (hint: insert into ...)

8. write a query that returns the total number of tweets in table **tweets_parquet**.
   Annotate both the number of records and the amount of seconds that it took.

9. verify that both tables contain the same number of tweets.
   which of the queries was faster?

10. write a query that returns the total number of users with geolocation enabled from table **tweets_parquet**.

11. write a query that returns the total number of tweets per language from table **tweets_parquet**.

12. write a query that returns the top 10 users with more followers from table **tweets_parquet**.

13. write a query that returns the geoname latitude, longitude and timezone of the tweet place by joining **geonames** and **tweets_parquet**.
    (hint: there can be places with the same name in different countries
    normalize (upper/lower case) the place name when joining)

14. write a query that returns the total count, total distinct count, maximum, minimum, average, standard deviation and percentiles 25th, 50th, 75th, 100th of **media elements** in tweets from table **tweets_parquet**

15. write a query that returns the top 10 websites whose media contents are being shared from table **tweets_parquet**.

    CREATE TEMPORARY MACRO website(url string) parse_url(url, 'HOST');

16. create a table **tweet_words** in parquet format exploding the words in the tweets. Also normalize the words to lower case.
    (hint: use lateral view)

    Example

id	text
12345	“This a test”

id	word
12345	this
12345	is
12345	test

17. create a table **tweet_words_sentiment** in parquet format as the result of a query that returns the polarity of each word by left joining **tweet_words** with **sentiment_dictionary.** The polarity for non joining words will be neutral (you can use coalesce function). Also codify the polarity (you can use case when …) as integer in the following way:

positive ->1
neutral -> 0
negative -> -1

Example

id	word
12345	bad
12345	wewew

id	word	polarity
12345	bad	-1
12345	wewew	0

18. create a table **tweets_sentiment** in parquet format as the result of a query that sums the polarity of every tweet so that
    sum(polarity) > 0 -> ‘positive’
    sum(polarity) < 0 -> ‘negative’
    sum(polarity) = 0 -> ‘neutral’

    Example

id	word	polarity
12345	bad	-1
12345	wewew	0

id	polarity
12345	‘negative’

19. write a query that returns the hourly evolution of sentiment of tweets with hashtag ADA

    Example

hour	positive	negative
2019062522	1233	235
2019062523	2355	124

TEAM E

1. create a database named **crypto_team_e**.

2. select the database you just created so that all the tables you are going to create belong to that database.

3. create an external table named **sentiment_dictionary** over the file located in /user/flume/sentiment-dictionary/ .

4. create an external table named **tweets_json** over the files located in /user/flume/crypto-tweets/ .

   You don't need to reference all the fields in a tweet, just the ones to solve your assignment.
   You can use the table definition we saw during hive lab as a template (you will have to add and remove some fields).

5. write a query that returns the total number of tweets in table **tweets_json**.
   Annotate both the number of records and the amount of seconds that it took.

6. create a managed table **tweets_orc** with same schema as tweets_json but stored in orc format.
   (hint: create table … like)

7. insert all rows from **tweets_json** into **tweets_orc**.
   (hint: insert into ...)

8. write a query that returns the total number of tweets in table **tweets_orc**.
   Annotate both the number of records and the amount of seconds that it took.

9. verify that both tables contain the same number of tweets.
   which of the queries was faster?

10. write a query that returns the total number of users with geolocation enabled from table **tweets_orc**.

11. write a query that returns the total number of tweets per language from table **tweets_orc**.

12. write a query that returns the top 10 users with more followers from table **tweets_orc**.

13. write a query that returns the geoname latitude, longitude and timezone of the tweet place by joining **geonames** and **tweets_raw_orc**.
    (hint: there can be places with the same name in different countries
    normalize (upper/lower case) the place name when joining)

14. write a query that returns the total count, distinct count, max number of hashtags, min number of hashtags, average number of hashtags, the standard deviation and percentiles 25th, 50th, 75th, 100th of **words** in tweets from table **tweets_raw_orc**

15. write a query that returns the top 10 words with at least 4 letters from table **tweets_raw_orc**

16. create a table **tweet_words** in parquet format exploding the words in the tweets. Also normalize the words to lower case.
    (hint: use lateral view)

    Example

id	text
12345	“This a test”

id	word
12345	this
12345	is
12345	test

17. create a table **tweet_words_sentiment** in parquet format as the result of a query that returns the polarity of each word by left joining **tweet_words** with **sentiment_dictionary.** The polarity for non joining words will be neutral (you can use coalesce function). Also codify the polarity (you can use case when …) as integer in the following way:

positive ->1
neutral -> 0
negative -> -1

Example

id	word
12345	bad
12345	wewew

id	word	polarity
12345	bad	-1
12345	wewew	0

18. create a table **tweets_sentiment** in parquet format as the result of a query that sums the polarity of every tweet so that
    sum(polarity) > 0 -> ‘positive’
    sum(polarity) < 0 -> ‘negative’
    sum(polarity) = 0 -> ‘neutral’

    Example

id	word	polarity
12345	bad	-1
12345	wewew	0

id	polarity
12345	‘negative’

19. write a query that returns the hourly evolution of sentiment of tweets with hashtag LTC or litecoin

    Example

hour	positive	negative
2019062522	1233	235
2019062523	2355	124