**Hadoop-Group B assignment**

TEAM B

\_\_\_\_ > unzipping

unzip -x group-assignment-resources.zip

Results

------

[Mohamed.Amer@ip-172-31-39-113 ~]$ unzip -x group-assignment-resources.zip

Archive: group-assignment-resources.zip

creating: crypto-tweets/

inflating: crypto-tweets/tweets.1575228274215.json

inflating: crypto-tweets/tweets.1575231282880.json

inflating: crypto-tweets/tweets.1575225869134.json

inflating: crypto-tweets/tweets.1575224668061.json

inflating: crypto-tweets/tweets.1575230080835.json

inflating: crypto-tweets/tweets.1575230681766.json

inflating: crypto-tweets/tweets.1575227672756.json

inflating: crypto-tweets/tweets.1575224066922.json

inflating: crypto-tweets/tweets.1575226470158.json

inflating: crypto-tweets/tweets.1575223465891.json

inflating: crypto-tweets/tweets.1575225268613.json

inflating: crypto-tweets/tweets.1575227071220.json

inflating: crypto-tweets/tweets.1575229479047.json

inflating: crypto-tweets/tweets.1575228877124.json

creating: sentiment-dictionary/

inflating: sentiment-dictionary/dictionary.tsv

------

1. create a database named 2019o1\_team\_b .

create database 2019o1\_team\_b;

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

create belong to that database.

use 2019o1\_team\_b;

3. create an external table named sentiment\_dictionary with the files provided .

create external table sentiment\_dictionary

(Type string,

Length int,

Word string,

Word\_Type string,

Stemmed string,

Polarity string)

row format delimited

fields terminated by '\t'

stored as textfile

location '/user/Mohamed.Amer/sentiment-dictionary';

load data local inpath '/data/home/Mohamed.Amer/sentiment-dictionary' into table sentiment\_dictionary;

select count(\*) as total from sentiment\_dictionary;

-------------

hive> select count(\*) as total from sentiment\_dictionary;

Query ID = Mohamed.Amer\_20191211145441\_e1bfedf3-d57c-44f5-9dc2-3c5bc65a316b

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks determined at compile time: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1575415580694\_0408, Tracking URL = http://ip-172-31-44-81.eu-west-1.compute.internal:8088/proxy/application\_1575415580694\_0408/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1575415580694\_0408

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2019-12-11 14:54:49,833 Stage-1 map = 0%, reduce = 0%

2019-12-11 14:54:56,127 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.0 sec

2019-12-11 14:55:02,400 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.08 sec

MapReduce Total cumulative CPU time: 7 seconds 80 msec

Ended Job = job\_1575415580694\_0408

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.08 sec HDFS Read: 317342 HDFS Write: 5 SUCCESS

Total MapReduce CPU Time Spent: 7 seconds 80 msec

OK

total

8221

Time taken: 22.349 seconds, Fetched: 1 row(s)

-------------

4. create an external table named tweets\_json with the files provided .

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).

New columns to add Followers and timezone

create external table tweets\_json( created\_at string, id bigint, id\_str string, text string, source string, truncated boolean, in\_reply\_to\_status\_id bigint, in\_reply\_to\_status\_id\_str string, in\_reply\_to\_user\_id bigint, in\_reply\_to\_user\_id\_str string, in\_reply\_to\_screen\_name string,

`user` struct<id:bigint,id\_str:string,name:string,screen\_name:string,location:string,url:string,description:string ,protected:boolean,verified:boolean,followers\_count:int,friends\_count:int,listed\_count:int,favourites\_count:int,statuses\_count:int,utc\_offset:int,time\_zone:string,geo\_enabled:boolean,lang:string>,coordinates struct <coordinates:array<float>>,place struct<country:string,country\_code:string,full\_name:string,name:string,place\_type:string,url:string>, quoted\_status\_id bigint, quoted\_status\_id\_str string, is\_quote\_status boolean, quote\_count int, reply\_count int, retweet\_count int, favorite\_count int, favorited boolean, retweeted boolean, possibly\_sensitive boolean, filter\_level string, lang string, entities struct <

symbols:array<struct < text: string >>,user\_mentions: array < struct <screen\_name: string,name: string >>,

hashtags: array < struct <text: string >>>,year int,month int,day int,hour int) row format serde 'org.apache.hive.hcatalog.data.JsonSerDe' stored as textfile location '/user/Mohamed.Amer/hive/tweets\_json';

symbols

load data local inpath '/data/home/Mohamed.Amer/crypto-tweets/' into table tweets\_json;

-----------------

hive> select count(\*) from tweets\_json;

Query ID = Mohamed.Amer\_20191211145646\_6cce30e9-78a1-4b9e-945e-dfe7cffc3cbc

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks determined at compile time: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1575415580694\_0410, Tracking URL = http://ip-172-31-44-81.eu-west-1.compute.internal:8088/proxy/application\_1575415580694\_0410/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1575415580694\_0410

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2019-12-11 14:56:52,544 Stage-1 map = 0%, reduce = 0%

2019-12-11 14:57:01,978 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 8.81 sec

2019-12-11 14:57:08,196 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 11.78 sec

MapReduce Total cumulative CPU time: 11 seconds 780 msec

Ended Job = job\_1575415580694\_0410

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 11.78 sec HDFS Read: 52974933 HDFS Write: 5 SUCCESS

Total MapReduce CPU Time Spent: 11 seconds 780 msec

OK

\_c0

9639

Time taken: 22.684 seconds, Fetched: 1 row(s)

-------------------------------

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.

create temporary macro hasEmoji(text string) text rlike '🚀|💰|💎|🌝' ;

select `user`.name,text from tweets\_json where hasEmoji(text) limit 5;

select count(\*) as No\_Tweets from tweets\_json;

no\_tweets

9639

Time taken: 24.288 seconds, Fetched: 1 row(s)

6. create a managed table tweets\_parquet with same schema as tweets\_json

but stored in parquet format.

(hint: create table … like)

create table tweets\_parquet stored as parquet as select \* from tweets\_json;

select count(\*) as No\_parquet from tweets\_parquet;

create table tweets\_parquet like tweets\_json stored as parquet;

7. insert all rows from tweets\_json into tweets\_parquet .

(hint: insert into ...)

insert into tweets\_parquet select \* from tweets\_json;

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.

select count(\*) as No\_parquet from tweets\_parquet;

no\_parquet

9639

Time taken: 0.049 seconds, Fetched: 1 row(s)

9. verify that both tables contain the same number of tweets.

which of the queries was faster?

Done

hive> select count(\*) as No\_parquet from tweets\_parquet;

OK

no\_parquet

9639

Time taken: 0.066 seconds, Fetched: 1 row(s)

------

hive> select count(\*) as tweets\_no from tweets\_json;

Query ID = Mohamed.Amer\_20191209143451\_6d788ca7-a870-4d6b-b589-3ded1e3e42ac

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks determined at compile time: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1575415580694\_0046, Tracking URL = http://ip-172-31-44-81.eu-west-1.compute.internal:8088/proxy/application\_1575415580694\_0046/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1575415580694\_0046

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2019-12-09 14:34:57,922 Stage-1 map = 0%, reduce = 0%

2019-12-09 14:35:06,214 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 8.52 sec

2019-12-09 14:35:13,468 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 11.66 sec

MapReduce Total cumulative CPU time: 11 seconds 660 msec

Ended Job = job\_1575415580694\_0046

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 11.66 sec HDFS Read: 52974964 HDFS Write: 5 SUCCESS

Total MapReduce CPU Time Spent: 11 seconds 660 msec

OK

tweets\_no

9639

Time taken: 22.602 seconds, Fetched: 1 row(s)

-----

Parquet is way faster

10.write a query that returns the total number of users with geolocation enabled

from table tweets\_parquet .

select count(`user`.id) as num\_user\_geo\_enabled from tweets\_json where `user`.geo\_enabled = TRUE;

num\_user\_geo\_enabled

1828

Time taken: 22.51 seconds, Fetched: 1 row(s)

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

select lang, count(id) as lang\_count from tweets\_parquet group by lang order by lang\_count desc;

lang lang\_count

en 8124

und 437

es 310

tr 163

ja 144

pt 99

fr 73

de 59

it 36

in 33

nl 22

ru 12

da 12

pl 11

ar 10

tl 10

ht 10

et 8

el 8

vi 8

th 8

cs 5

cy 5

zh 5

ca 5

ro 4

lt 3

eu 3

sv 3

hu 2

ko 2

sl 1

fi 1

no 1

is 1

lv 1

Time taken: 43.179 seconds, Fetched: 36 row(s)

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

tweets\_parquet .

With mygroup as (select `user`.name as User\_Name, `user`.followers\_count As Num\_Followers, concat(substr(created\_at,27,4),CASE WHEN lower(substr(created\_at,5,3))= "dec" THEN 12 WHEN lower(substr(created\_at,5,3))= "nov" THEN 11 WHEN lower(substr(created\_at,5,3))= "oct" THEN 10 WHEN lower(substr(created\_at,5,3))= "sep" THEN 09 WHEN lower(substr(created\_at,5,3))= "aug" THEN 08 WHEN lower(substr(created\_at,5,3))= "jul" THEN 07 WHEN lower(substr(created\_at,5,3))= "jun" THEN 06 WHEN lower(substr(created\_at,5,3))= "may" THEN 05 WHEN lower(substr(created\_at,5,3))= "apr" THEN 04 WHEN lower(substr(created\_at,5,3))= "mar" THEN 03 WHEN lower(substr(created\_at,5,3))= "feb" THEN 02 ELSE "01" END, substr(created\_at,9,2), substr(created\_at,12,2)) as Hourly\_Evolution, ROW\_NUMBER() OVER ( PARTITION BY (`user`.name) ORDER BY (`user`.followers\_count) DESC ) AS (ROW\_NUM)

from tweets\_parquet order by Num\_Followers desc , Hourly\_Evolution desc )

Select \* from mygroup where mygroup.ROW\_Num = 1 limit 10;

-------

mygroup.user\_name mygroup.num\_followers mygroup.hourly\_evolution mygroup.row\_num

CNBC 3320803 2019120120 1

Estado de Minas 507521 2019120118 1

Pure Michigan 497994 2019120118 1

Rich Simmonds 485857 2019120118 1

Cointelegraph 484874 2019120120 1

Vegas 461802 2019120119 1

Pomp 🌪 296365 2019120120 1

Ｍｉｃｈａｅｌ 288249 2019120118 1

𝗣𝗼®️𝗻𝗧Ⓤ𝗯𝗯𝗲 ™🔞🐖 283785 2019120119 1

Joshuwa Roomsburg 🌐 270664 2019120118 1

Time taken: 42.773 seconds, Fetched: 10 row(s)

hive>

-------

13.write a query that returns the geoname latitude, longitude and timezone of the

tweet place by joining geonames and tweets\_parquet .

**Answer:**

create external table geonames (id bigint,name string,ascii\_name string,alternate\_names array<string>,latitude float,longitude float,feature\_class string,feature\_code string,country\_code string,country\_code2 array<string>,admin1\_code string,admin2\_code string,admin3\_code string,admin4\_code string,population bigint,elevation int,dem int,timezone string,modification\_date date)

row format delimited fields terminated by '\t' collection items terminated by ',' stored as textfile location '/user/Mohamed.Amer/hive/geonames';

load data local inpath '/data/home/Mohamed.Amer/cities15000.txt' into table geonames;

select distinct t.id,g.latitude,g.longitude,g.timezone, t.place.name, g.name from tweets\_parquet t left join geonames g on upper(t.place.country\_code) = upper(g.country\_code) where g.latitude is not null

and ((upper(t.place.name) like upper(g.name)) or (upper(g.name) like upper(t.place.name)));

-----

col\_name data\_type comment

id bigint

name string

ascii\_name string

alternate\_names array<string>

latitude float

longitude float

feature\_class string

feature\_code string

country\_code string

country\_code2 array<string>

admin1\_code string

admin2\_code string

admin3\_code string

admin4\_code string

population bigint

elevation int

dem int

timezone string

modification\_date date

Time taken: 0.34 seconds, Fetched: 19 row(s)

hive>

-----

(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

select sum(size(`entities`.symbols.text)) as Total\_Count,T.Total\_Distinct\_Count, max(size(`entities`.symbols.text)) as maxim, min(size(`entities`.symbols.text)) as minim, avg(size(`entities`.symbols.text)) as avgim, std(size(`entities`.symbols.text)) as stdim,

percentile(size(`entities`.symbols.text),0.25) perc\_25th, percentile(size(`entities`.symbols.text),0.50) perc\_50th, percentile(size(`entities`.symbols.text),0.75) perc\_75th, percentile(size(`entities`.symbols.text),0.97) perc\_97th, percentile(size(`entities`.symbols.text),1) perc\_100th from tweets\_parquet J,(select count(distinct cashtag) Total\_Distinct\_Count from tweets\_parquet lateral view explode(`entities`.symbols.text) w as cashtag) T group by T. Total\_Distinct\_Count;

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

tweets\_parquet

**Answer:**

select upper(cashtag) as Cashtag,count(upper(cashtag)) Total\_Count from tweets\_parquet lateral view explode(`entities`.symbols.text) w as cashtag group by upper(cashtag) order by Total\_Count desc limit 10;

**Results:**

cashtag total\_count

BTC 1129

ETH 303

XRP 236

XMR 172

LTC 171

VET 125

RVN 94

TRX 77

XLM 72

EOS 49

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

**Answer:**

create table word\_split as select id,split(lower(text),' ') as words from tweets\_json;

create table tweet\_words stored as parquet as select id,word from word\_split LATERAL VIEW explode(words) w as word;

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

**Answer:**

Create table tweet\_words\_sentiment stored as parquet as select id, WS.word,

case SD.polarity

when 'negative' then -1

when 'positive' then 1

else 0 end as polarity

from tweet\_words WS left outer join sentiment\_dictionary SD on WS.word = SD.word;

**Results:**

netural:

hive> select \* from tweet\_words\_sentiment where polarity=0 limit 20;

OK

tweet\_words\_sentiment.id tweet\_words\_sentiment.word tweet\_words\_sentiment.polarity

1201200372443951104 let's 0

1201200372443951104 talk 0

1201200372443951104 bitcoin! 0

Positive:

tweet\_words\_sentiment.id tweet\_words\_sentiment.word tweet\_words\_sentiment.polarity

1201200372443955202 interest 1

1201200372443955202 interest 1

1201200372443955202 interest 1

Negative:

hive> select \* from tweet\_words\_sentiment where polarity=-1 limit 20;

OK

tweet\_words\_sentiment.id tweet\_words\_sentiment.word tweet\_words\_sentiment.polarity

1201200394652786688 limited -1

1201200394652786688 moon -1

1201200440211394560 block -1

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’

**Answer:**

create table tweets\_sentiment stored as parquet as select

id,

case

when sum( polarity ) < 0 then 'negative'

when sum( polarity ) > 0 then 'positive'

else 'neutral' end as polarity\_sum

from tweet\_words\_sentiment group by id;

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

hive> select created\_at from tweets\_parquet limit 10;

OK

created\_at

Sun Dec 01 18:04:20 +0000 2019

Sun Dec 01 18:04:20 +0000 2019

Sun Dec 01 18:04:20 +0000 2019

Sun Dec 01 18:04:21 +0000 2019

Sun Dec 01 18:04:22 +0000 2019

Sun Dec 01 18:04:22 +0000 2019

Sun Dec 01 18:04:25 +0000 2019

Sun Dec 01 18:04:25 +0000 2019

Sun Dec 01 18:04:26 +0000 2019

Sun Dec 01 18:04:26 +0000 2019

Time taken: 0.028 seconds, Fetched: 10 row(s)

hive>

**Answer:**

select P. Hourly\_Evolution, sum(negative) as Negative, sum(positive) as Positive from (select concat(substr(TP.created\_at,27,4),CASE WHEN lower(substr(created\_at,5,3))= "dec" THEN 12 WHEN lower(substr(created\_at,5,3))= "nov" THEN 11 WHEN lower(substr(created\_at,5,3))= "oct" THEN 10 WHEN lower(substr(created\_at,5,3))= "sep" THEN 09 WHEN lower(substr(created\_at,5,3))= "aug" THEN 08 WHEN lower(substr(created\_at,5,3))= "jul" THEN 07 WHEN lower(substr(created\_at,5,3))= "jun" THEN 06 WHEN lower(substr(created\_at,5,3))= "may" THEN 05 WHEN lower(substr(created\_at,5,3))= "apr" THEN 04 WHEN lower(substr(created\_at,5,3))= "mar" THEN 03 WHEN lower(substr(created\_at,5,3))= "feb" THEN 02 ELSE "01" END, substr(created\_at,9,2), substr(created\_at,12,2)) as Hourly\_Evolution, case TS.polarity\_sum when "negative" then 1 else 0 end as negative, case TS.polarity\_sum when "positive" then 1 else 0 end as positive from tweets\_parquet TP inner join tweets\_sentiment TS on TS.ID= TP.id

inner join (select Cashtag,id,total\_count from (select upper(cashtag) as Cashtag,count(upper(cashtag)) Total\_Count,id from tweets\_parquet lateral view explode(`entities`.symbols.text) w as cashtag group by id,upper(cashtag)) Cash where upper(cashtag)='ETH')temp\_eth on temp\_eth.ID= TP.id ) P group by P. Hourly\_Evolution;

**Results**

p.hourly\_evolution negative positive

2019120118 3 26

2019120119 7 50

2019120120 0 15

Time taken: 49.527 seconds, Fetched: 3 row(s)