# HADOOP GROUP ASSIGNMENT

# TEAM D: Tarek El Noury, Tommaso Fantin,

# Louise Amélie Leuchtenberger, Dennis Pedersen, Rebecca Rosser, Vasileios Sagiannos

1. Create a database named ​**2020a1\_team\_d**​.

create database 2020a1\_team\_d

1. Select the database you just created so that all the tables you are going to create belong to that database.

Browse & select database “2020a1\_team\_d”

1. 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 serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'   
with serdeproperties ("separatorChar" = "\t")  
stored as textfile   
location '/user/rebeccarosser/team/sentiment-dictionary/';

1. Create an external table named ​**tweets\_json**​ with the files provided.

create external table tweets\_json

(id bigint,

text string,

created\_at string,

`user` struct<id:bigint, name:string,screen\_name:string,geo\_enabled:boolean,followers\_count:int,url:string>,

lang string,

place struct<name:string, country\_code:string>,

coordinates struct<coordinates:array<float>>)

row format serde 'org.apache.hive.hcatalog.data.JsonSerDe'

stored as textfile

location '/user/rebeccarosser/raw/twitter/tweets';

1. 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.

select count(\*) from tweets\_json;

In terminal:

9639 records

23.564 seconds

1. Create a managed table ​**tweets\_parquet** with the same schema as tweets\_json but stored in parquet format.  
   (hint: create table ... like)

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

1. Insert all rows from ​**tweets\_json**​ into ​**tweets\_parquet**​. (hint: insert into ...).

insert into table tweets\_parquet

select \* from tweets\_json;

1. 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 (\*) from tweets\_parquet;

9639 records

0.037 seconds

1. Verify that both tables contain the same number of tweets. Which of the queries was faster?

They give us different numbers. Tweets\_parquet is faster. This is why storing data in the right format is so important.

1. Write a query that returns the total number of users with geolocation enabled from table ​**tweets\_parquet**​.

Select count (distinct id)

From tweets\_parquet

Where `user`.geo\_enabled = TRUE;

1828 users with geolocation enabled

22.076 seconds

1. Write a query that returns the total number of tweets per language from table **tweets\_parquet**​.

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

1. Write a query that returns the top 10 users with more followers from table **tweets\_parquet**​.

select `user`.screen\_name, max(`user`.followers\_count) as total   
from tweets\_parquet   
group by `user`.screen\_name   
order by total desc   
limit 10;

1. Write a query that returns the total count, total distinct count, maximum, minimum, average, standard deviation and percentiles 25th, 50th, 75th of **media elements** ​in tweets from table ​**tweets\_parquet**

select sum(media\_count) as count,   
count(distinct media\_count) as count\_distinct,  
max(media\_count) as max,   
min(media\_count) as min,  
avg(media\_count) as average,  
stddev(media\_count) as stddev,  
percentile(media\_count,0.25) as p1,  
percentile(media\_count,0.5) as p2,  
percentile(media\_count,0.75) as p3

from (select id, size(entities.media) as media\_count from tweets\_parquet) t;

1. Write a query that returns the top 10 websites whose media contents are being shared from table ​**tweets\_parquet**​.

select `user`.url, count(\*) as total

from tweets\_parquet

group by `user`.url

order by total desc

limit 10;

1. Create a table ​**tweet\_words** ​in parquet format exploding the words in the tweets. Also normalize the words to lowercase.  
   (hint: use lateral view)

create table tweets\_words

stored as parquet as

select id, word

from tweets\_parquet

lateral view explode (split(lower(text), ' ')) t as word;

1. 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 l(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

create table tweets\_words\_sentiment

stored as parquet

as select t1.id, t1.word,

case when t2.polarity = 'negative' then -1

when t2.polarity = 'positive' then 1

else 0

end as polarity

from tweets\_words t1

left outer join sentiment\_dictionary t2 on t1.word = t2.word;

1. 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’

### create table tweets\_sentiment

### stored as parquet as

### select id,

### case when sum(polarity) > 0 then 'positive'

### when sum(polarity) < 0 then 'negative'

### else 'neutral'

### end as polarity

### from tweets\_words\_sentiment

### group by id;

1. Write a query that returns the hourly evolution of sentiment of tweets with hashtag ADA.

create temporary MACRO tweet\_hour(created\_at string) from\_unixtime(unix\_timestamp(created\_at, 'EEE MMM dd HH:mm:ss ZZZZZ yyyy'),'yyyyMMddHH');

select

from\_unixtime(unix\_timestamp(t1.created\_at, 'EEE MMM dd HH:mm:ss ZZZZZ yyyy'),'yyyyMMddHH') as hour,

sum(case when t2.polarity= 'positive' then 1 else 0 end) as positive,

sum(case when t2.polarity = 'negative' then 1 else 0 end) as negative

from tweets\_parquet t1

inner join tweets\_sentiment t2

on t1.id = t2.id

where lower(text) rlike '#ada'

group by from\_unixtime(unix\_timestamp(t1.created\_at, 'EEE MMM dd HH:mm:ss ZZZZZ yyyy'),'yyyyMMddHH')

order by hour;