Impact estimation of earthquakes with a Twitter analysis

Final report

School: Lucerne University of Applied Sciences and Arts
Program: Master of Science in Applied Information and Data Science (MScIDS)
Course: Data Warehouse and Data Lake Systems 2
Team: Earthquake
Place/Date: Lucerne, 31.12.2021

Supervisors: José Mancera and Luis Teràn

Authors:

Sandro Huber	Thomas Schwendimann	Lea Senn
Spitalackerstrasse 20a	Spitalstrasse 8b	Treppenweg 19
3013 Bern	6210 Sursee	5300 Turgi
+4178 840 09 09	+41 79 129 19 43	+4179 960 74 79
sandro.huber@stud.hslu.ch	thomas.schwendimann@stud.hslu.ch	lea.senn@stud.hslu.ch

Table of Contents

List of Figures

1 Background

As a global reinsurance company, our client SwissRe is affected by global catastrophes on a broad scale. In 2020, SwissRe estimated the global insured catastrophe losses to be USD 83 billion (SwissRe, 2020). A large amount of these costs are caused by earthquakes. To be aware what costs are going to be expected, it is important for SwissRe to get as much information about the damages as early as possible.

1.1 Problem statement

While the insured losses caused by earthquakes vary from year to year (Figure 1), the influence on both insurance companies as well as their clients is quite large, both in costs as well as number of victims (Figure 2).

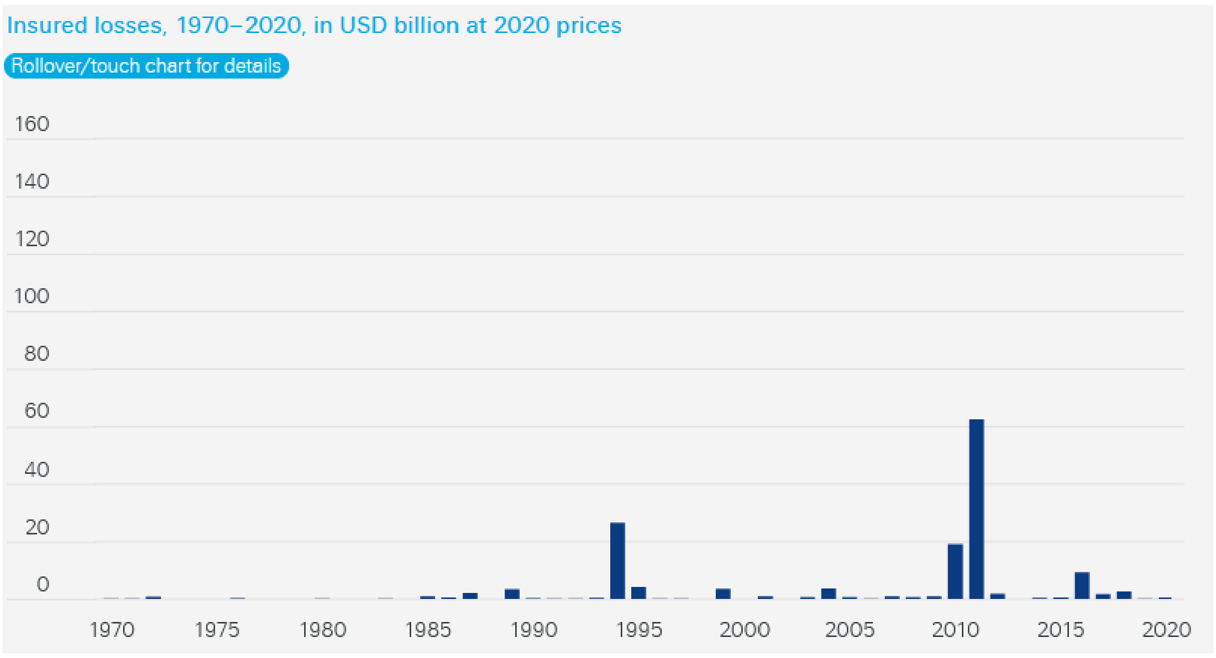


Figure 1 - Insured losses for earthquakes/tsunamis, 1970-2020, in USD billion at 2020 prices (Source: Swiss Re Institute)

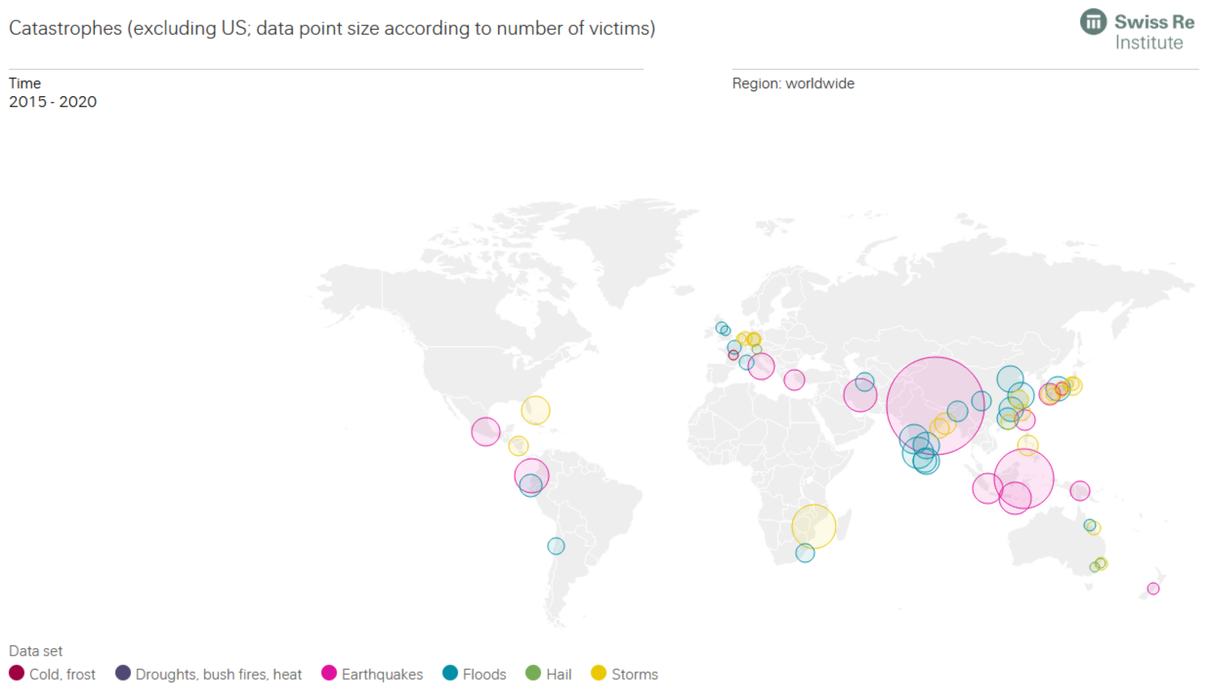


Figure 2 - number of victims depending on catastrophes (excluding US), 2015-2020, (Source Swiss Re Institute)

While seismographs around the world record earthquakes – even if they are barely perceptible to humans – and make the data available in real time, the effects of damage to buildings and people are often only visible later when the damage reports arrive. Additionally, the seismographs are reporting earthquakes all around the world, which means also at very remote places where no costs for SwissRe arise. Therefore, the information from the seismographs should be enriched with qualitative data in order to better estimate an earthquake.

Furthermore, SwissRe has to support their affected clients immediately, improving customer experience and possibly even reducing the damage. In January 2019, SwissRe wrote a lengthy report, indicating how the future business models for insurance companies may change due to digitization. One of these changes is how insurers interact with their clients. As of now, the customer journey is mostly limited to the signing of the contract and the claims, which may or may not arise. In the future, insurance companies might try to be involved more closely with their clients, even utilizing touchpoints such as mobile phone apps, where the insurance company can warn their clients ahead of possibly damaging events (Avramakis, Anchen, & Raverkar, 2019).

1.2 Business questions

Hundreds of earthquakes are registered worldwide every day. Fortunately, most of them cause little damage to buildings and people. But for larger earthquakes that produce a lot of damage, SwissRe's claims experts want to know as soon as possible what costs they need to expect.

We assume that bigger damages also lead to a bigger response in social media. To estimate the damage, we want to try to analyse the "Twitter echo" after big earthquakes and compare the amount of Tweets with the numbers from comparable earthquakes in the past.

We plan to make this comparison available to SwissRe's claims experts on a dashboard in an intuitive way. The claims experts from SwissRe can then use this information and match it with their historical cost data. They might find a relationship between the number of tweets and the costs and can use the information to forecast future costs for earthquakes.

For this comprehensive analysis, we will analyse stronger earthquakes (i.e. magnitude of 6 or more on the Richter scale) from the past. We suspect that relevant Twitter messages can be found for the last five to ten years. In the period before that, Twitter was probably not widespread enough.

Further, a sentiment analysis of the content of the tweets could be interesting to get more insights about how large the damage is. Since NLP is not the main topic of this course, we see this part as rather optional.

We decided to answer the following research questions:

1. *At what maginute do earthquakes trigger a reaction on Twitter?*
2. *Is there a dependency between the strength of the earthquake and the number of tweets sent?*
3. *How did the sentiment in the Tweets change based on the magnitude?*

2 State of the art

The U.S. Geological Survey of the USGS (which also provides the API used in this work) has an article examining the influence of earthquakes on tweets. The researchers come to the conclusion that "the tweet-frequency time series constructed from tweets containing the word 'earthquake' clearly shows large peaks correlated with the origin times of widely felt events." Also important to our work is the USGS's recognition that the response to Twitter comes very quickly. About 75 percent of tweets are registered within two minutes of the quake. The USGS refers to this as "considerably faster than seismographic detections in poorly instrumented regions of the world." (Earle et al., 2011)

The finding that Twitter reports a faster response time than the USGS was also evident in the 2008 Sichuan earthquake, in part because the USGS operated most of its seismographs in the U.S. at the time and areas such as China were poorly equipped. This fact drove the USGS to continue working with data from Twitter. This also created dashboards with various data on earthquakes on Twitter (Figure 3). Details about this project are available on Twitter's blog (Elaine, 2015).

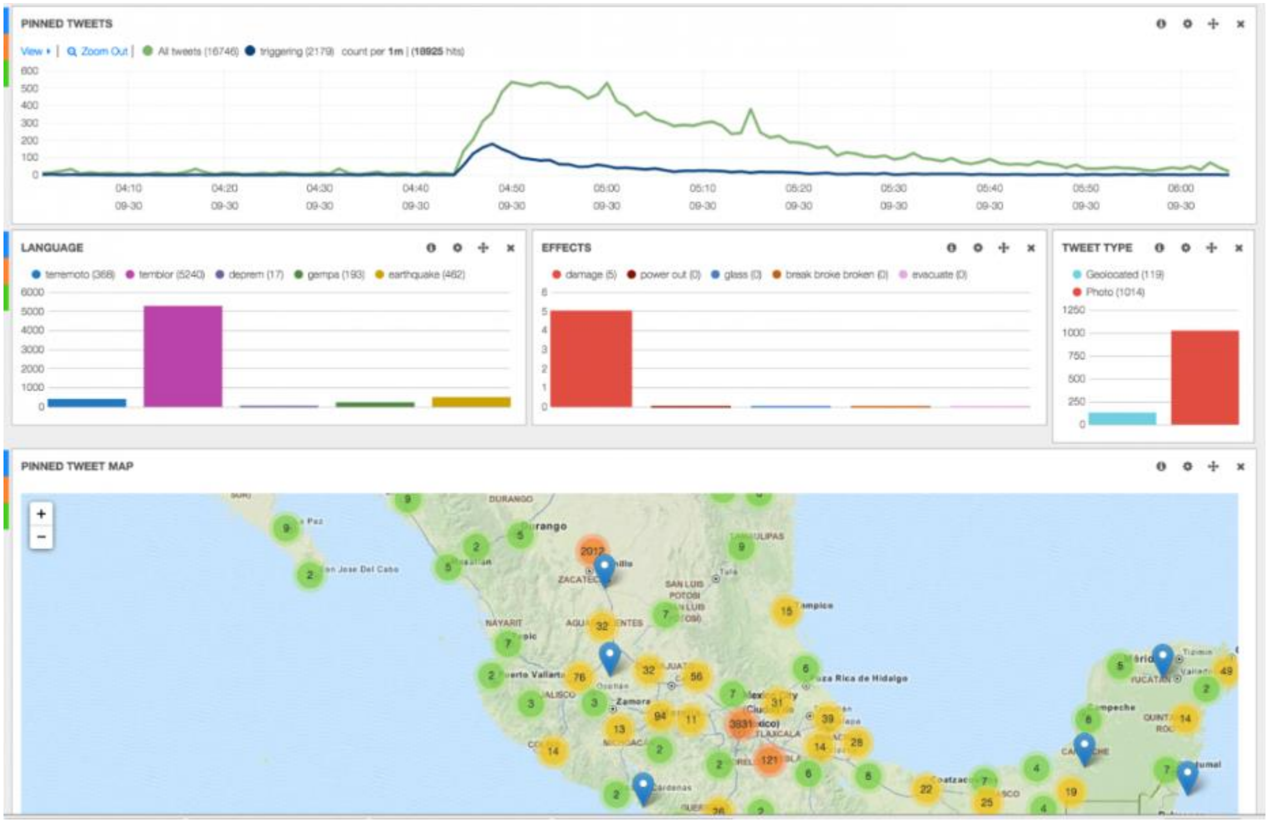

PINNED TWEETS
View | Zoom Out | ● All tweets (16746) ● triggering (2179) count per 1m | (18925 hits)
LANGUAGE
● terremoto (368) ● temblor (5240) ● deprem (17) ● gempa (193) ● earthquake (462)
EFFECTS
● damage (5) ● power out (0) ● glass (0) ● break broke broken (0) ● evacuate (0)
TWEET TYPE
● Geolocated (119) ● Photo (1014)
PINNED TWEET MAP

Figure 3 - Screenshot of the dashboard by USGS (Elaine, 2015)

3 Methodology

The aim is to quantify the twitter echo after a major earthquake by counting the tweets and retweets including the word "earthquake" on twitter within 24 hours after an event. It is planned to do this for a selection of the most severe earthquakes in terms of their magnitude, probably for earthquakes with a magnitude of 6 or more on the Richter scale, as these are described as "Damage to a moderate number of well-built structures in populated areas. Earthquake-resistant structures survive with slight to moderate damage. Poorly designed structures receive moderate to severe damage. Felt in wider areas; up to hundreds of kilometers from the epicenter. Strong to violent shaking in epicentral area." (USGS, 2010). We suspect that the greater the damage and therefore the greater the costs, the more earthquake-related messages are posted on Twitter.

This chapter describes the data sources used in this project and explains the used technology and procedure.

3.1 Data sources

Two different APIs are going to be used for this project.

3.1.1 Twitter API

Twitter is a social media platform with roughly 500 million tweets every day, which accumulates to 6’000 tweets a second (brandwatch, 2020). As such, Twitter is quick to pick up on trends and events that are happening right at this moment, all around the globe. The Twitter API gives access to historic tweets as well as streaming them in real time. To be able to use these services to their full extent, an academic research account is required, for which we applied for.

From our first tests we know that around 10‘000 to 15‘000 tweets including the word „earthquake“ are posted on Twitter every day. A significant part of them are posted by bots, that do not react to human perception or even damages.

Information about the Twitter API are published under
https://developer.twitter.com/en/docs/twitter-api

3.1.2 Earthquake API

The National Institute of Standards and Technology leads the USGS Earthquake Hazards Program of the U.S. Geological Survey (USGS). On their website, they provide an API with relevant earthquake science information and knowledge, including the time, location, magnitude and others. The API is updated every minute.

Details can be found under https://earthquake.usgs.gov/fdsnws/event/1/.

4 Procedure - Data lake

4.1 Twitter Data

In a first step, the Twitter API is used to access the historical data for as far back as possible (max. 10 years) with the limit of 10 million tweets a month. Python's library 'requests' is used for all GET requests to the API. The code is executed in a Directed Acyclic Graph (DAG) from Apache Airflow. This chapter provides an overview of the steps involved in extracting the data and storing it in a postgreSQL database – the data lake – using a DAG.

4.1.1 Preparation

Due to the monthly limit of tweets that can be accessed (10 mio. tweets per month), it was not possible to create a query which returns every tweet containing the word 'earthquake'. A single day already returns roughly over 10'000 tweets, which would limit the time frame that is accessible for this project. In order to increase this time frame, test queries have been executed to optimize the final search criteria. For example, retweets, replies and the word 'minor' are excluded from the search.

Also, a lot of tweets are created by bots which do not deliver insights to our research. Those tweets usually contain no relevant information, such as earthquakes with a low value on the Richter scale. Therefore, the number of tweets a single account posts a day were counted for a short time period. In order to further reduce the number of results we obtain, the accounts with the highest number of tweets were extracted from the data and excluded from the query. Since the length of the query is limited to 1024 characters, only around 40 usernames have been excluded. Once the data is in the data lake, the other bots will be removed when transforming the data before it is loaded into the data warehouse.

4.1.2 Data Lake

For the data storage, a postgreSQL database has been created in Thomas Schwendimann's Amazon Relational Database Service (RDS). This database serves as a data lake, where all the Twitter data will be stored in its raw format. This includes for one the tweets matching the criteria, but also information about the users which posted them. These data will be stored in separate tables.

4.1.3 Extracting the Data

Figure 4 represents the DAG for the process of extracting and loading the Twitter Data into the data lake. Due to the access limitation of the API, such as the number of allowed requests every 15 minutes, this DAG is executed every six hours and will extract half a year's worth of data. The defined time interval is 1 January 2010 to 31 October 2021.

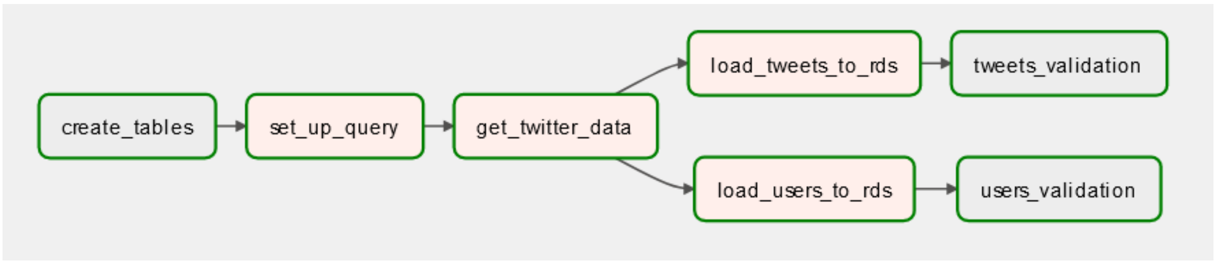


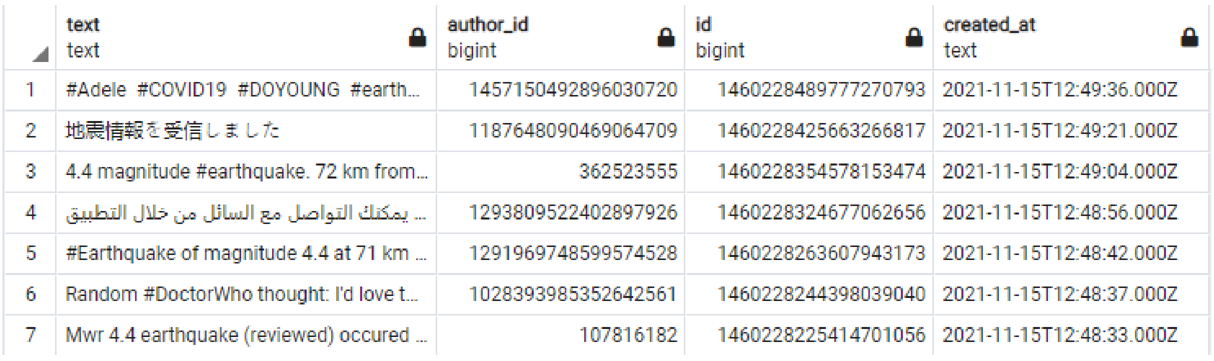
Figure 4 - DAG get_twitter.py

The first task checks if the corresponding tables ('tweets' and 'users') in the data lake exist, and if not creates them. The tables are therefore ideally only created with the first execution. Next up are the query parameters, which are defined in a separate task. Arguably, this step could be combined with the extracting of the data, but was chosen to be separate for a better overview. Two variables were added to the Apache Airflow environment, allowing access to them without writing the values as code. First, the bearer token required for the authentication when making the API requests. Second, the filter for the query, which consists of the prefix '-from:' and the usernames that are excluded from the search. These variables are accessed and used in the setup of the query. The query parameters, URL and headers required for the request are saved using XComs (cross-communications), a mechanism that allows sending data between tasks.

The next step pulls the XComs with the relevant variables and accesses the API and requests the Twitter data matching the query. Only 500 tweets can be accessed per request from the API. With every request the API returns a token in the metadata – assuming more results are there but were not returned due to the limit. Using this token, one can then request again and will get the data beginning from where the previous request left off. This is done in an infinite while loop, which stops if there is no token in the metadata, indicating that all the results have been delivered. For the tweets, the fields 'created_at', 'text', 'author_id' and 'id' are returned as per query defined. For the users, the fields 'name', 'username', 'location' (if available) and 'id' (which matches the author id). All the responses are stored in lists containing a dictionary (json format) for each tweet and user respectively – and then once again saved using XComs.

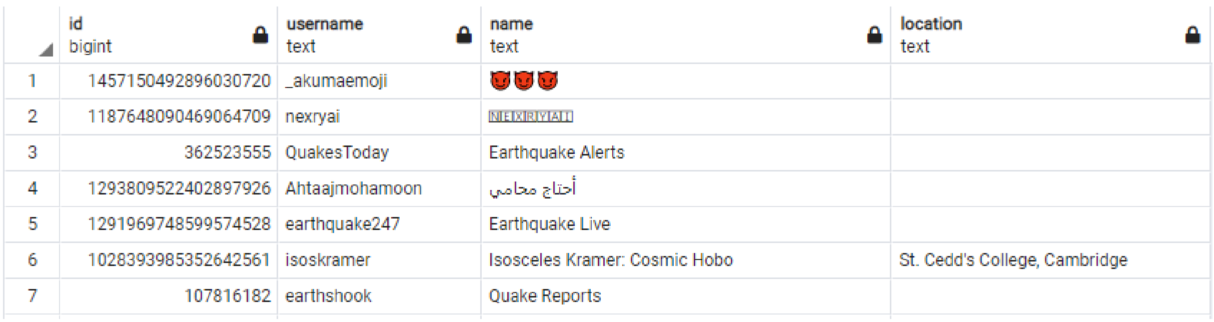
4.1.4 Loading the Data

The loading of the tweets and user information into the database is done in two separate tasks. In both cases, an iterator is created from the list containing the data – in order to save memory. Since not all user entries contain the key 'location', the key and an empty string as value is added in its place if not in the dictionary. This allows for easy insertion into the database. To connect to the database, a PostgresHook accesses the connection details saved in Apache Airflow – once again to not have sensitive data visible in the code. Due to the volume of data, the execute_batch function from psycopg2's library is used to insert the rows into the tables. This requires a separate import psycopg2.extras. If the data volume were even higher, the copy_from method would have been used, which has the fastest execution time when combined with an iterator (Benita, 2019). Finally, once the data has been loaded, a simple SELECT request is done for each table to validate if the process has been successful or not. In the tables below is the output of a simple SELECT statement using pgAdmin4, a software with a GUI for postgreSQL databases.



	text text	author_id bigint	id bigint	created_at text
1	#Adele #COVID19 #DOYOUNG #earth...	1457150492896030720	1460228489777270793	2021-11-15T12:49:36.000Z
2	地震情報を受信しました	1187648090469064709	1460228425663266817	2021-11-15T12:49:21.000Z
3	4.4 magnitude #earthquake. 72 km from...	362523555	1460228354578153474	2021-11-15T12:49:04.000Z
4	... يمكنك التواصل مع السائل من خلال التطبيق	1293809522402897926	1460228324677062656	2021-11-15T12:48:56.000Z
5	#Earthquake of magnitude 4.4 at 71 km ...	1291969748599574528	1460228263607943173	2021-11-15T12:48:42.000Z
6	Random #DoctorWho thought: I'd love t...	1028393985352642561	1460228244398039040	2021-11-15T12:48:37.000Z
7	Mwr 4.4 earthquake (reviewed) occured ...	107816182	1460228225414701056	2021-11-15T12:48:33.000Z

Figure 5 - Abstract table tweets



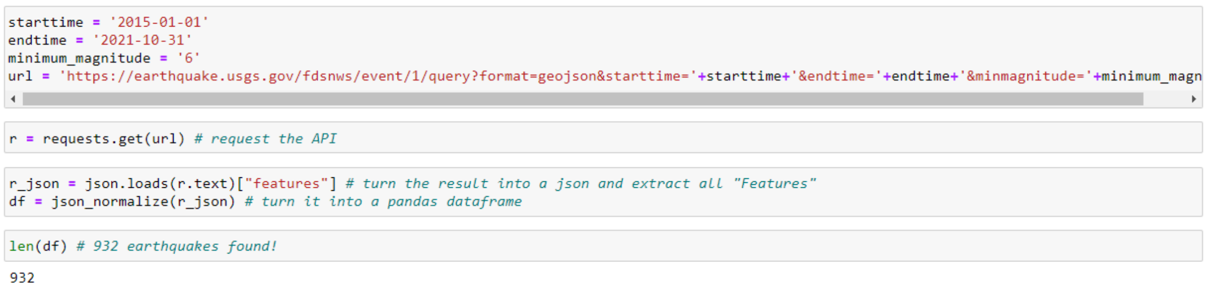
	id bigint	username text	name text	location text
1	1457150492896030720	_akumaemoji	👿👿👿	
2	1187648090469064709	nexryai	🄽🄴🅇🅁🅈🄰🄸	
3	362523555	QuakesToday	Earthquake Alerts	
4	1293809522402897926	Ahtaajmohamoon	أحتاج محامي	
5	1291969748599574528	earthquake247	Earthquake Live	
6	1028393985352642561	isoskramer	Isosceles Kramer: Cosmic Hobo	St. Cedd's College, Cambridge
7	107816182	earthshook	Quake Reports	

Figure 6 - Abstract table tweets_user

4.2 Earthquake data

The earthquake data was retrieved from the earthquake.usgs.gov API. The API is provided by the United States Geological Survey (USGS) and can directly be accessed over the URL without any tokens or other verifications needed. The data can already be filtered via setting of parameters in the URL of the API. To save computational power and space on our web server, this filtering method was used to pre-filter all earthquakes that are not relevant for the project. This was done with the definition of the following parameters:

- Starttime
- Endtime
- Minimum magnitude



```
starttime = '2015-01-01'
endtime = '2021-10-31'
minimum_magnitude = '6'
url = 'https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&starttime='+starttime+'&endtime='+endtime+'&minmagnitude='+minimum_magn
```

```
r = requests.get(url) # request the API
```

```
r_json = json.loads(r.text)["features"] # turn the result into a json and extract all "Features"
df = json_normalize(r_json) # turn it into a pandas dataframe
```

```
len(df) # 932 earthquakes found!
```

```
932
```

Figure 7 - SEQ Figure * ARABIC 7 - Access API

The time frame of the observation is between 1 January 2015 until the 31 October 2021. These dates were chosen in order to have a big sample size but also to limit ourselves to a certain time frame that is possible to cover with the twitter data. Additionally, the minimum magnitude of six has been defined, because of the above already described reasons in the elaboration of the background of the project.

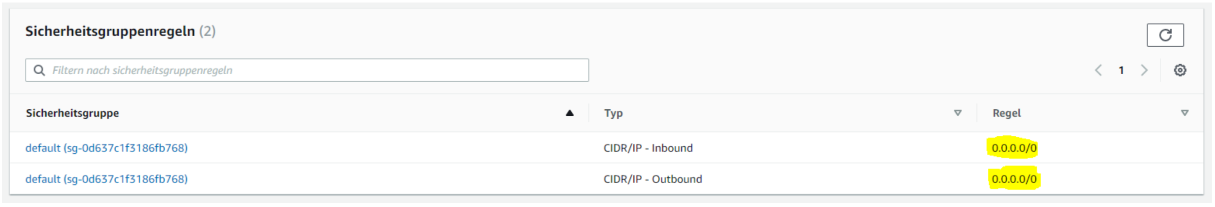
With help of this first filtering of the data, the API delivers 932 relevant earthquakes. Obviously, this list has again to be cleaned, for example in terms of their location, because of the irrelevancy of earthquakes that are happening in the ocean or other remote places. This duty will be done during the next project stage in the second half of the semester.

4.2.1 Data storage

To have a uniform data storage, the earthquake data is stored in a postgreSQL database, similar to the twitter data. For this purpose, the database first needed to be created in the Amazon RDS. When it comes to safety aspects in terms of limited access to the database from specific IP-addresses, our group could not follow the best practice of a limited access. It was necessary to leave the access open to all IP-addresses since no one of our group member has a static IP-address. Therefore, this option was not possible for our team. However, the data is not sensitive, therefore, safety is not a big issue in this case. Nevertheless, the database is secured with help of a password, which is sufficient for this project.

4.2.2 Data loading

This step was first tried to be accomplished with help of a small sample dataset with help of a Jupyter



Sicherheitsgruppenregeln (2)

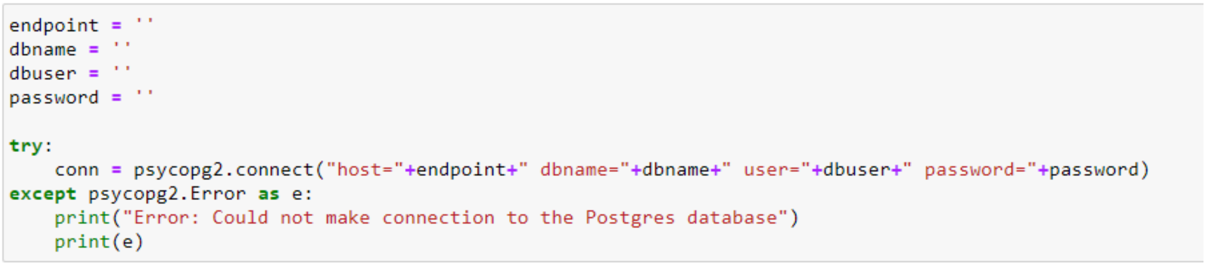
Filtern nach sicherheitsgruppenregeln

Sicherheitsgruppe	Typ	Regel
default (sg-0d637c1f3186fb768)	CIDR/IP - Inbound	0.0.0.0/0
default (sg-0d637c1f3186fb768)	CIDR/IP - Outbound	0.0.0.0/0

Figure 8 - GIVE TITLE

Notebook. This was done with help of the following steps:

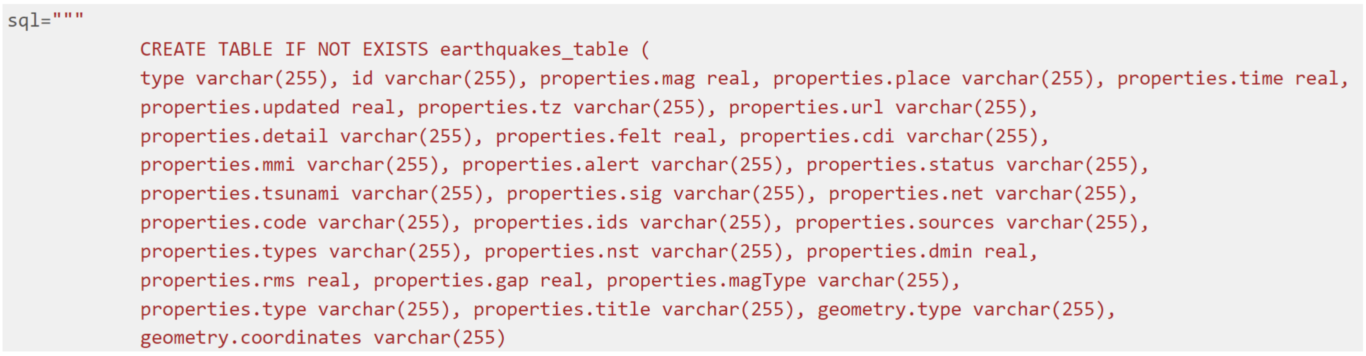
- Downloading the data via USGS API
- Small data preparation step
    - Unfolding of the geodata points
- Connecting to the database with the according credentials



```
endpoint = ''
dbname = ''
dbuser = ''
password = ''

try:
    conn = psycopg2.connect("host="+endpoint+" dbname="+dbname+" user="+dbuser+" password="+password)
except psycopg2.Error as e:
    print("Error: Could not make connection to the Postgres database")
    print(e)
```

Figure 9 - SEQ Figure * ARABIC 9 - Connection via jupyter notebook

- Creation of a table in the PostgreSQL database with the following code snippet



```
sql="""
            CREATE TABLE IF NOT EXISTS earthquakes_table (
            type varchar(255), id varchar(255), properties.mag real, properties.place varchar(255), properties.time real,
            properties.updated real, properties.tz varchar(255), properties.url varchar(255),
            properties.detail varchar(255), properties.felt real, properties.cdi varchar(255),
            properties.mmi varchar(255), properties.alert varchar(255), properties.status varchar(255),
            properties.tsunami varchar(255), properties.sig varchar(255), properties.net varchar(255),
            properties.code varchar(255), properties.ids varchar(255), properties.sources varchar(255),
            properties.types varchar(255), properties.nst varchar(255), properties.dmin real,
            properties.rms real, properties.gap real, properties.magType varchar(255),
            properties.type varchar(255), properties.title varchar(255), geometry.type varchar(255),
            geometry.coordinates varchar(255)
```

Figure 10 - GIVE TITLE

- Implementation of the data
- Close the connection again

After the successful implementation of the data into the PostgreSQL server with help of this little local prototype, the implementation was automated with help of Apache Airflow ().

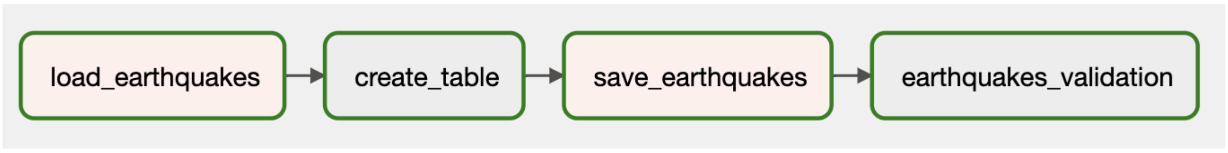


Figure 11 - DAG earthquake_daily.py

4.3 Data lake storage overview

Finally, the created data lake includes the following information:

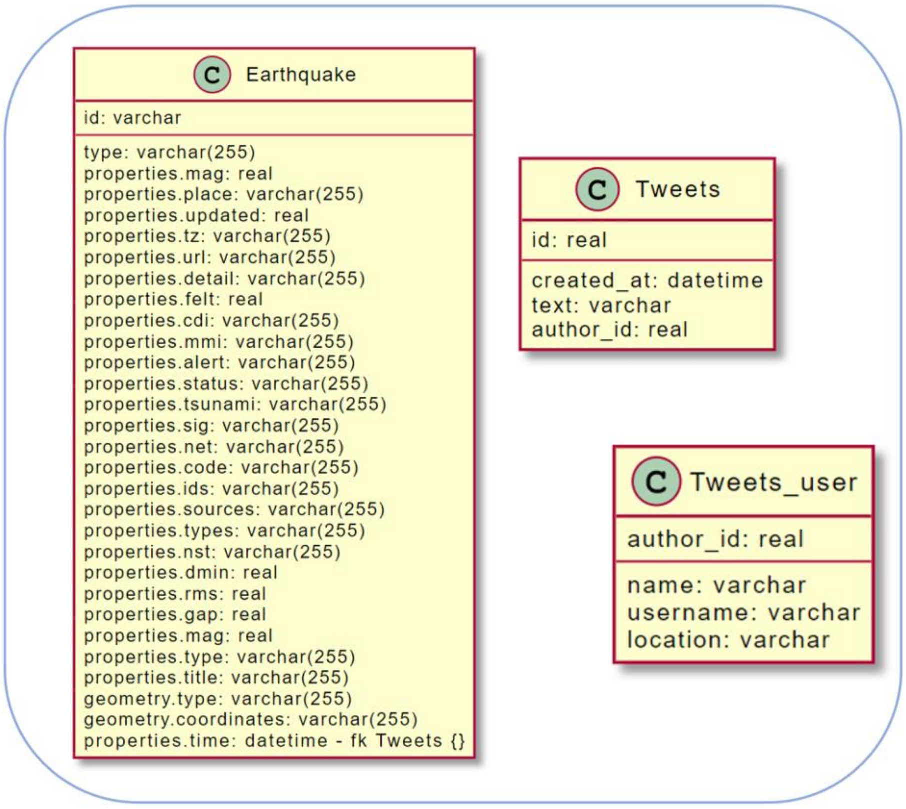

C Earthquake
id: varchar
type: varchar(255)
properties.mag: real
properties.place: varchar(255)
properties.updated: real
properties.tz: varchar(255)
properties.url: varchar(255)
properties.detail: varchar(255)
properties.felt: real
properties.cdi: varchar(255)
properties.mmi: varchar(255)
properties.alert: varchar(255)
properties.status: varchar(255)
properties.tsunami: varchar(255)
properties.sig: varchar(255)
properties.net: varchar(255)
properties.code: varchar(255)
properties.ids: varchar(255)
properties.sources: varchar(255)
properties.types: varchar(255)
properties.nst: varchar(255)
properties.dmin: real
properties.rms: real
properties.gap: real
properties.mag: real
properties.type: varchar(255)
properties.title: varchar(255)
geometry.type: varchar(255)
geometry.coordinates: varchar(255)
properties.time: datetime - fk Tweets {}
C Tweets
id: real
created_at: datetime
text: varchar
author_id: real
C Tweets_user
author_id: real
name: varchar
username: varchar
location: varchar

Figure 12 - SEQ Figure * ARABIC 12 – Final datalake

Until this point, the data is floating in the database in loose tables. This is the goal of the first stage of this project. However, it is important to have an outlook on how the data can be prepared and rearranged to combine these raw data tables. Therefore, the following relationship model has been drawn:

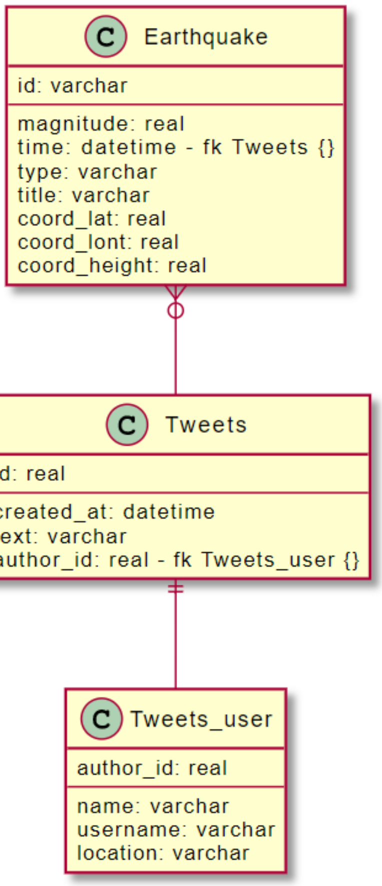


Figure 13 - GIVE TITLE

After the processing of the data, the tweets can be associated with the earthquakes according to the “time” of the earthquake and the “created_at” of the tweets. An earthquake can have zero or multiple tweets. Additionally, a twitter user can directly be assigned to a tweet with help of the “author_id”. One tweet must have exactly one user who created the tweet.

5 Procedure – Data Warehouse

As stated in previous chapters, the extracted data was loaded into the data lake in a mostly raw format, with only slight changes. The data lake serves as the basis for all business questions, as it contains all information available. For our project, only some of this data is useful. As such, we used Apache Airflow to clean and filter the data and load it into the data warehouse (Figure 14).

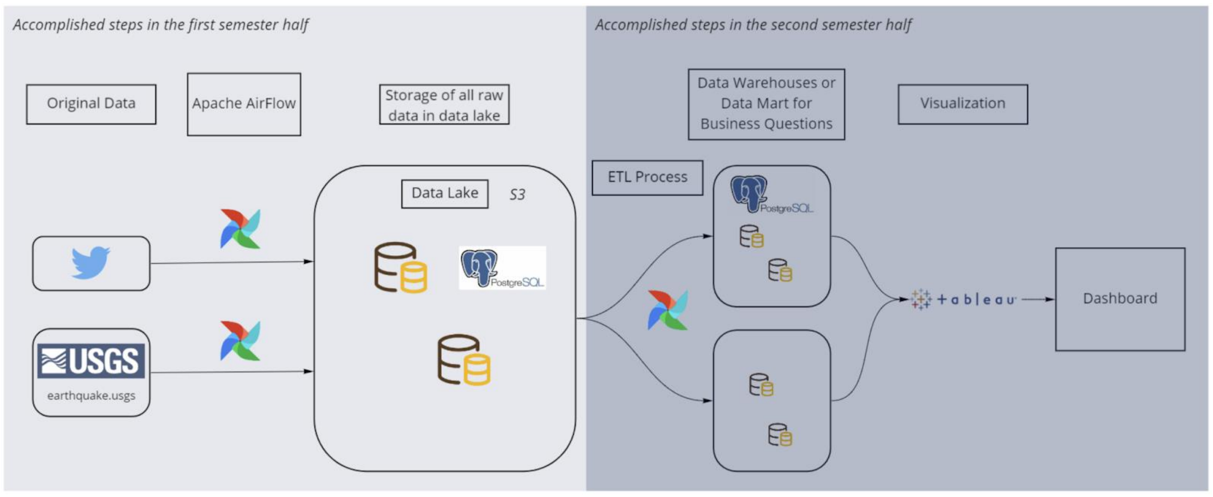

Accomplished steps in the first semester half
Accomplished steps in the second semester half
Original Data
Apache AirFlow
Storage of all raw data in data lake
Data Warehouses or Data Mart for Business Questions
Visualization
Data Lake
S3
ETL Process
PostgreSQL
earthquake.usgs
USGS
tableau
Dashboard

Figure 14 - Architecture

5.1 Twitter Data

Cleaning data is usually an explorative process. Looking at the data reveals insights of what needs to be changed, adjusted or dropped altogether. Once changes have been made, exploring other aspects of the data reveals even more. However, since the cleaning and loading of the data into the warehouse was to be done in a one step process – a DAG using Apache Airflow – a Jupyter notebook was used to explore the data and find what changes need to be done. The resulting transformations were then executed in the respective DAG. In both cases, sensitive data, such as credentials and tokens for accessing APIs, has been hidden from the code. In Apache Airflow, the built-in function was used to store variables. In Jupyter notebook, the dotenv library enabled to store the variables inside an .env file, which is ignored by the git repository (.gitignore). Figure X represents the individual steps of the DAG (clean_datalake.py).

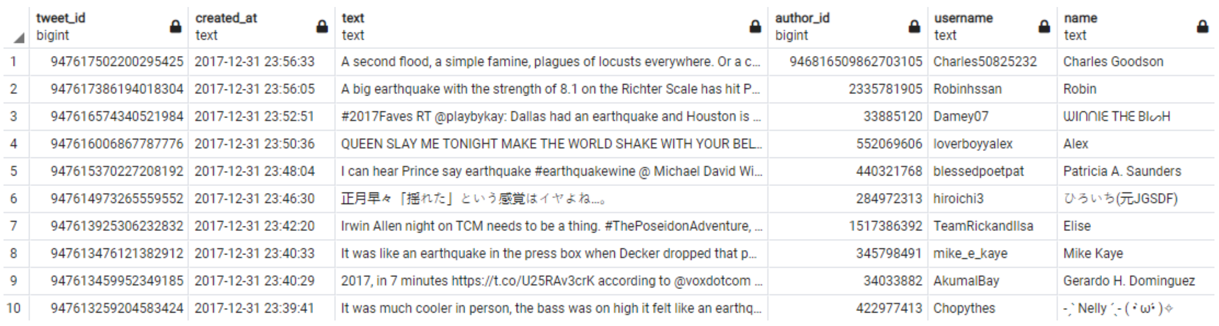
INSERT DIAGRAM

Just as for the data lake, the tables for the data warehouse are created in the first step if they do not already exist. This step could potentially be omitted, since the way the data is later loaded into the data warehouse does not require existing tables. They are instead created dynamically if they do not already exist. In this case, a temporary table is created additionally to the regular data warehouse table to store the data between tasks.

In parallel, the two tables containing information of the users and tweets are combined into one, while at the same time missing data is extracted from the Twitter API. To be more precise, some user data was not loaded into the data lake in the first step due to invalid characters which the PostgreSQL database could not handle. As a result, if one username contained this character, all the usernames remaining in the iterator were not loaded into the table. At the time, this was ignored since there was a save way to find the periods with the usernames missing and later extract them again. However, as the project progressed it was apparent that it would have been easy to simply continue the loading process of the data once the error message had been raised. Since an iterator is used, the username containing the invalid character would have no longer been in said iterator, and the rest could have been inserted safely. This would have ensured that only a few usernames would have been missing, which could have been ignored and the time-consuming step of extracting the data would have not been necessary. Ultimately, the user data did not serve a purpose for the project and would not have been needed in the first place. However, in a real-life scenario the data could still be valuable for different analysis and should at least be stored in the data lake.

As a final step of the cleaning process, the bots which consistently tweeted about every earthquake were removed from the data. To identify these bots, two rules were defined and applied. First, every user containing the string sequence 'quake' or 'bot' were dropped. Second, all users that tweeted over a hundred times were dropped.

Finally, the data was loaded into the data warehouse and the temporary table was dropped. The whole process reduced the number of tweets from around 13.5 million to 6 million.



	tweet_id bigint	created_at text	text text	author_id bigint	username text	name text
1	947617502200295425	2017-12-31 23:56:33	A second flood, a simple famine, plagues of locusts everywhere. Or a c...	946816509862703105	Charles50825232	Charles Goodson
2	947617386194018304	2017-12-31 23:56:05	A big earthquake with the strength of 8.1 on the Richter Scale has hit P...	2335781905	Robinhssan	Robin
3	947616574340521984	2017-12-31 23:52:51	#2017Faves RT @playbykay: Dallas had an earthquake and Houston is ...	33885120	Damey07	ꟺIՈՈIƐ THƐ BI൮H
4	947616006867787776	2017-12-31 23:50:36	QUEEN SLAY ME TONIGHT MAKE THE WORLD SHAKE WITH YOUR BEL...	552069606	loverboyyalex	Alex
5	947615370227208192	2017-12-31 23:48:04	I can hear Prince say earthquake #earthquakewine @ Michael David Wi...	440321768	blessedpoetpat	Patricia A. Saunders
6	947614973265559552	2017-12-31 23:46:30	正月早々「揺れた」という感覚はイヤよね...。	284972313	hiroichi3	ひろいち(元JGSDF)
7	947613925306232832	2017-12-31 23:42:20	Irwin Allen night on TCM needs to be a thing. #ThePoseidonAdventure, ...	1517386392	TeamRickandIlsa	Elise
8	947613476121382912	2017-12-31 23:40:33	It was like an earthquake in the press box when Decker dropped that p...	345798491	mike_e_kaye	Mike Kaye
9	947613459952349185	2017-12-31 23:40:29	2017, in 7 minutes https://t.co/U25RAv3crK according to @voxdotcom ...	34033882	AkumalBay	Gerardo H. Dominguez
10	947613259204583424	2017-12-31 23:39:41	It was much cooler in person, the bass was on high it felt like an earthq...	422977413	Chopythes	-ˏˋ Nelly ˊˎ- (•̀ ω •́)✧

Figure 15 - Abstract data warehouse dw_twitter

5.1.1 Counting Tweets

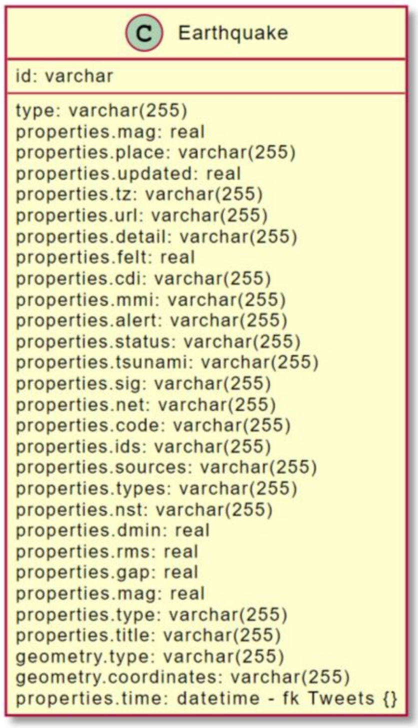
During the later stage of the project, it was apparent that using the extracted data to create the dashboard would be difficult due to the long loading process. As a result, Twitter API was used to simply count the number of tweets matching the defined criteria (DAG get_twitter_count.py). This resulted in one single data point per day, which significantly reduced the data load. Given more time and more computational power, the analysis of the actual tweets could provide better insights, as there is no way to exclude bots from the count of tweets matching the criteria.

5.2 Cleaning earthquakes table

This chapter describes how the earthquake data was transformed into a data warehouse. The data cleaning was not only done in the second part of the project but started already with the data loading into the data lake. What happened in which parts will be described next.

5.2.1 Cleaning during the first stage (loading of the data in the data lake)

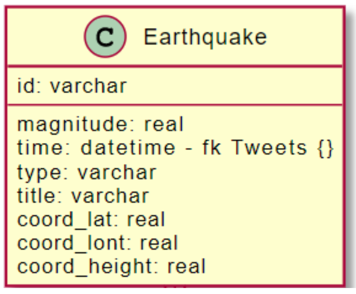
As mentioned above, the earthquake data from the USGS website consists of the following table columns.



```
C Earthquake
id: varchar
type: varchar(255)
properties.mag: real
properties.place: varchar(255)
properties.updated: real
properties.tz: varchar(255)
properties.url: varchar(255)
properties.detail: varchar(255)
properties.felt: real
properties.cdi: varchar(255)
properties.mmi: varchar(255)
properties.alert: varchar(255)
properties.status: varchar(255)
properties.tsunami: varchar(255)
properties.sig: varchar(255)
properties.net: varchar(255)
properties.code: varchar(255)
properties.ids: varchar(255)
properties.sources: varchar(255)
properties.types: varchar(255)
properties.nst: varchar(255)
properties.dmin: real
properties.rms: real
properties.gap: real
properties.mag: real
properties.type: varchar(255)
properties.title: varchar(255)
geometry.type: varchar(255)
geometry.coordinates: varchar(255)
properties.time: datetime - fk Tweets {}
```

Figure 16 - GIVE TITLE

Since our group has done a precise planning of how the data should be visualized and what data is needed to realize it, we were able to already filter out some information from the table from the beginning. Due to this, computational power, time and space on the database could be saved. Additionally, the structure within the data lake was clearer, since only a reduced data set was loaded in it. The relevant columns that were loaded into the data lake were the following.



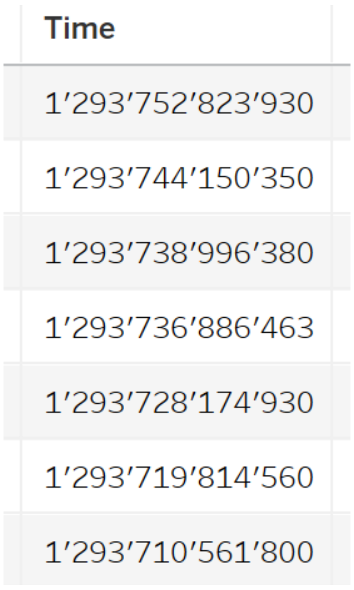
```
C Earthquake
id: varchar
magnitude: real
time: datetime - fk Tweets {}
type: varchar
title: varchar
coord_lat: real
coord_lont: real
coord_height: real
```

Figure 17 - GIVE TITLE

Therefore, at this stage, a big part of data cleaning has already been done. This step was realised with the first DAG of loading the data lake.

5.2.2 Cleaning during the second stage (from data lake into data warehouse)

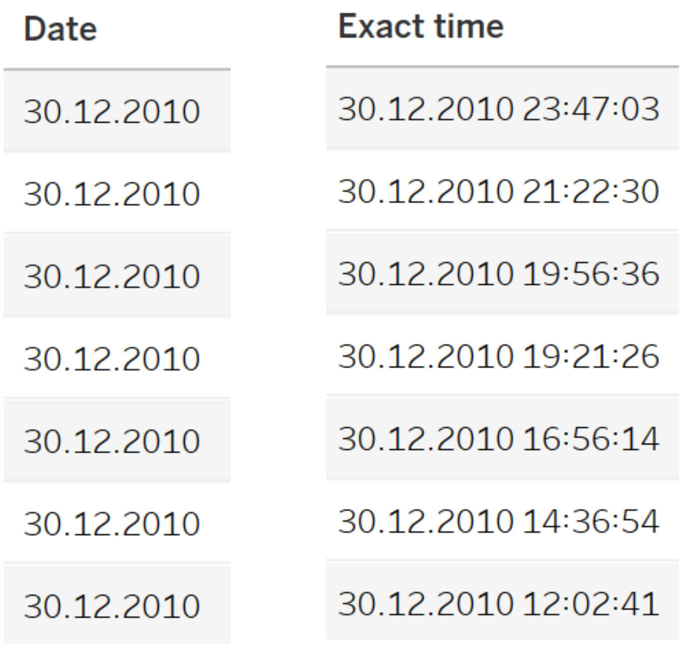
The data actually needed not much cleaning, since it came already very well structured and complete from the api. There were for example neither duplicate entries present nor NaN values. However, there was still an additional task to fulfill. In order to combine the twitter and earthquake datasets, it was necessary to create a new column that could be used to make the join on. It was defined at the start of the process that this will be the time of the occurrence of the earthquake and the tweet. The earthquake data has stored its time information in terms of miliseconds according to the following illustration:



Time
1’293’752’823’930
1’293’744’150’350
1’293’738’996’380
1’293’736’886’463
1’293’728’174’930
1’293’719’814’560
1’293’710’561’800

Figure 18 - GIVE TITLE

Since the time of the tweet is stored in another format, it was decided to transform this column into two new columns. First it was aimed to transform it into a datetime format. Therefore, the exact date as well as the hours, minutes and seconds should be shown. Second, a new column should be created with only the date. This is the column on which the merge of the two tables should happen. This was defined, because a day is an ideal timeframe to aggregate the earthquakes and the twitter data. The target columns can be seen in the following illustration.

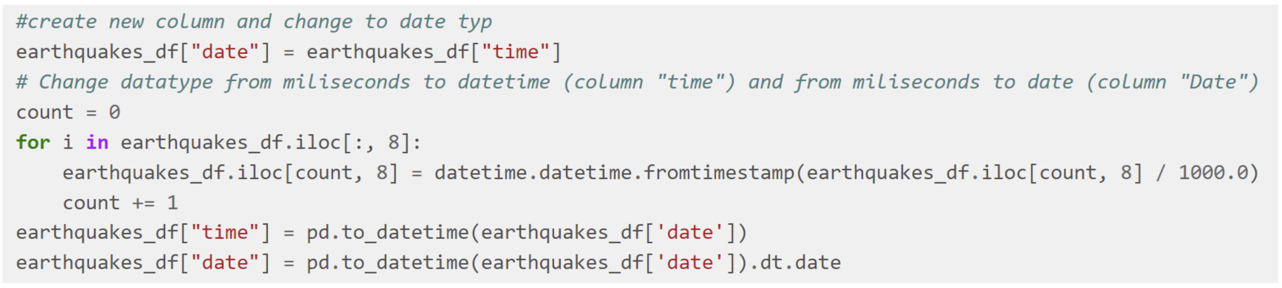


Date	Exact time
30.12.2010	30.12.2010 23:47:03
30.12.2010	30.12.2010 21:22:30
30.12.2010	30.12.2010 19:56:36
30.12.2010	30.12.2010 19:21:26
30.12.2010	30.12.2010 16:56:14
30.12.2010	30.12.2010 14:36:54
30.12.2010	30.12.2010 12:02:41

Figure 19 - GIVE TITLE

For the creation of these columns, the data was loaded again from the data lake and was transformed within another DAG and then loaded into the final data warehouse. Additionally, the original column with the milliseconds was dropped, since this would have been redundant information.

This process was done with the following code:



```
#create new column and change to date typ
earthquakes_df["date"] = earthquakes_df["time"]
# Change datatype from miliseconds to datetime (column "time") and from miliseconds to date (column "Date")
count = 0
for i in earthquakes_df.iloc[:, 8]:
    earthquakes_df.iloc[count, 8] = datetime.datetime.fromtimestamp(earthquakes_df.iloc[count, 8] / 1000.0)
    count += 1
earthquakes_df["time"] = pd.to_datetime(earthquakes_df['date'])
earthquakes_df["date"] = pd.to_datetime(earthquakes_df['date']).dt.date
```

Figure 20 - GIVE TITLE

Within the “time” column, the exact time is listed and in the “date” column, only the date. At the end the “date” column was used for the merge.

5.3 Preparation of the twitter_count table for the merge

For the data warehouse, the twitter_count table was needed as a completion of the earthquake data. The table holds the information about how many tweets were sent per day. The time frame is defined by a start- and endtime of the day. This is fine but it does not fit the requirement for the merge with the earthquake dataset. Only one column with the date is needed. Therefore, the starttime was transformed into the same datatype as in the earthquake table. This was done within the same DAG as the earthquake “date” column was created.

```
##Access twitter data and create df
twitter_df = db_hook.get_pandas_df("SELECT * FROM tweetcount")
#Adjust twitter table
#change type of "starttime"
twitter_df["starttime"] = pd.to_datetime(twitter_df["starttime"]).dt.date
# Rename column "starttime"
twitter_df.rename(columns={"starttime": "date"}, inplace=True)
# Drop Endtime
twitter_df.drop("endtime", axis=1, inplace=True)
```

Figure 21 - GIVE TITLE

After this last preprocessing, the two data sets were ready for the merge.

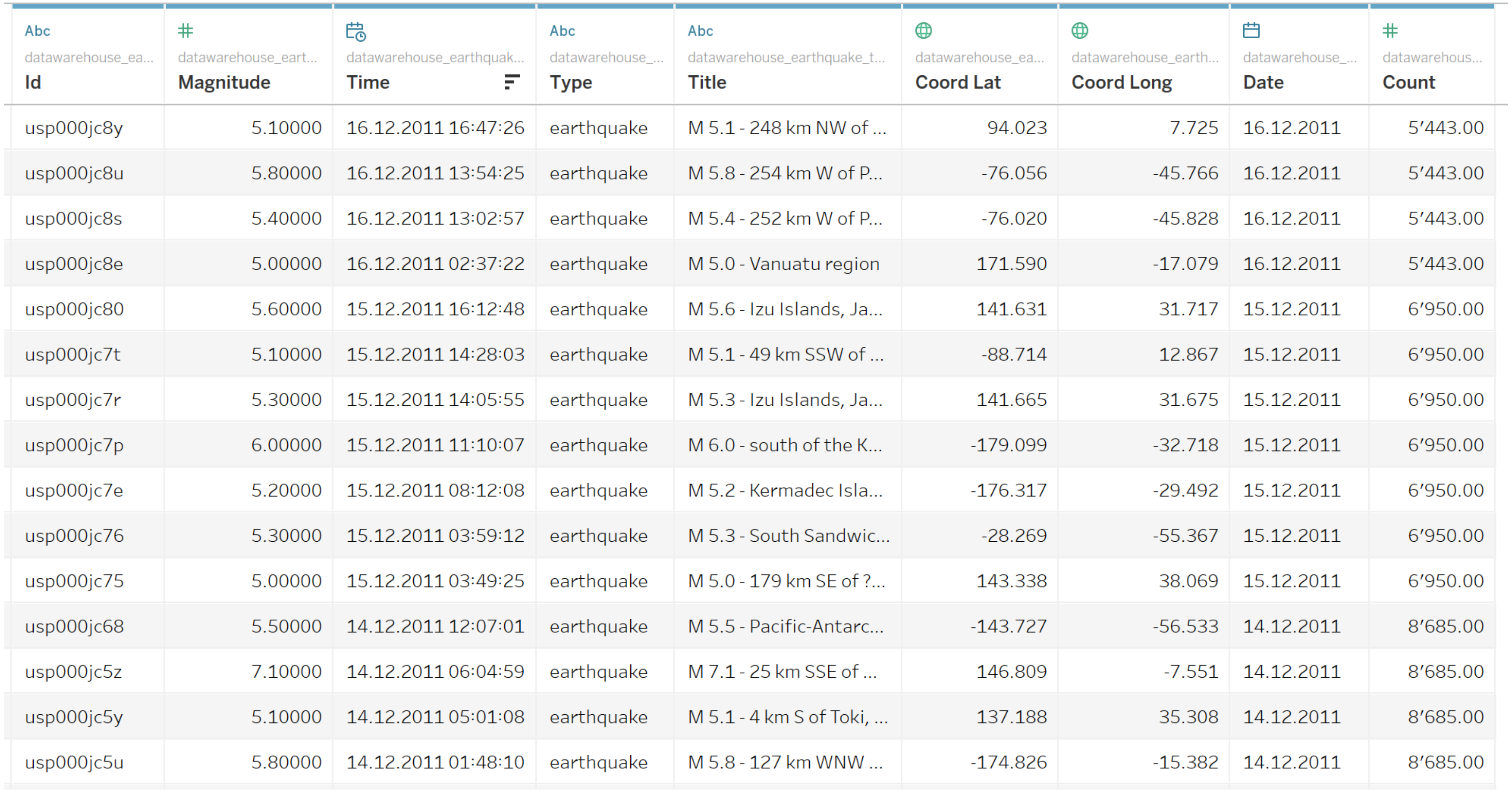
5.4 Final data warehouse table

The final data warehouse for the visualization was created within the same DAG as in the tweet_count table preparation and the earthquake table preparation.

```
    # merge tables
    earthquake_tweetcount = earthquakes_df.merge(twitter_df, on='date', how='left')

    logging.info(f'{earthquake_tweetcount.head(10)} records_quakes')
    logging.info(f'{earthquake_tweetcount.columns} records_quakes')

    # write data to RDS table
    # Now set up insert statement
    sql = 'INSERT INTO datawarehouse_earthquake_tweetcount(id, magnitude, time, type, title, coor
    for index, row in earthquake_tweetcount.iterrows():
        try:
            sql += ("('{}',{},'{}','{}','{}',{},{},'{}',{}), ".format(row['id'],row['magnitude'],
        except:
            logging.info("Problem with earthquake ({}, {})".format(row['id'],row['title']))


dag = DAG(
        'earthquake_tweetcount_datawarehouse',
        schedule_interval='@daily',
        start_date=datetime.datetime.now() - datetime.timedelta(days=30))
```

Figure 22 - GIVE TITLE

The final data warehouse table looks in the end like this:



Abc	#		Abc	Abc				#
datawarehouse_ea...	datawarehouse_eart...	datawarehouse_earthquak...	datawarehouse_...	datawarehouse_earthquake_t...	datawarehouse_ea...	datawarehouse_earth...	datawarehouse_...	datawarehous...
Id	**Magnitude**	**Time**	**Type**	**Title**	**Coord Lat**	**Coord Long**	**Date**	**Count**
usp000jc8y	5.10000	16.12.2011 16:47:26	earthquake	M 5.1 - 248 km NW of ...	94.023	7.725	16.12.2011	5’443.00
usp000jc8u	5.80000	16.12.2011 13:54:25	earthquake	M 5.8 - 254 km W of P...	-76.056	-45.766	16.12.2011	5’443.00
usp000jc8s	5.40000	16.12.2011 13:02:57	earthquake	M 5.4 - 252 km W of P...	-76.020	-45.828	16.12.2011	5’443.00
usp000jc8e	5.00000	16.12.2011 02:37:22	earthquake	M 5.0 - Vanuatu region	171.590	-17.079	16.12.2011	5’443.00
usp000jc80	5.60000	15.12.2011 16:12:48	earthquake	M 5.6 - Izu Islands, Ja...	141.631	31.717	15.12.2011	6’950.00
usp000jc7t	5.10000	15.12.2011 14:28:03	earthquake	M 5.1 - 49 km SSW of ...	-88.714	12.867	15.12.2011	6’950.00
usp000jc7r	5.30000	15.12.2011 14:05:55	earthquake	M 5.3 - Izu Islands, Ja...	141.665	31.675	15.12.2011	6’950.00
usp000jc7p	6.00000	15.12.2011 11:10:07	earthquake	M 6.0 - south of the K...	-179.099	-32.718	15.12.2011	6’950.00
usp000jc7e	5.20000	15.12.2011 08:12:08	earthquake	M 5.2 - Kermadec Isla...	-176.317	-29.492	15.12.2011	6’950.00
usp000jc76	5.30000	15.12.2011 03:59:12	earthquake	M 5.3 - South Sandwic...	-28.269	-55.367	15.12.2011	6’950.00
usp000jc75	5.00000	15.12.2011 03:49:25	earthquake	M 5.0 - 179 km SE of ?...	143.338	38.069	15.12.2011	6’950.00
usp000jc68	5.50000	14.12.2011 12:07:01	earthquake	M 5.5 - Pacific-Antarc...	-143.727	-56.533	14.12.2011	8’685.00
usp000jc5z	7.10000	14.12.2011 06:04:59	earthquake	M 7.1 - 25 km SSE of ...	146.809	-7.551	14.12.2011	8’685.00
usp000jc5y	5.10000	14.12.2011 05:01:08	earthquake	M 5.1 - 4 km S of Toki, ...	137.188	35.308	14.12.2011	8’685.00
usp000jc5u	5.80000	14.12.2011 01:48:10	earthquake	M 5.8 - 127 km WNW ...	-174.826	-15.382	14.12.2011	8’685.00

Figure 23 - GIVE TITLE

The data warehouse is basically the earthquake dataset, that is enriched with the tweet count per day. What can be seen is that the count per day is assigned to every earthquake with the same day.

6 Visualization

The final visualization can be seen in the following illustration:

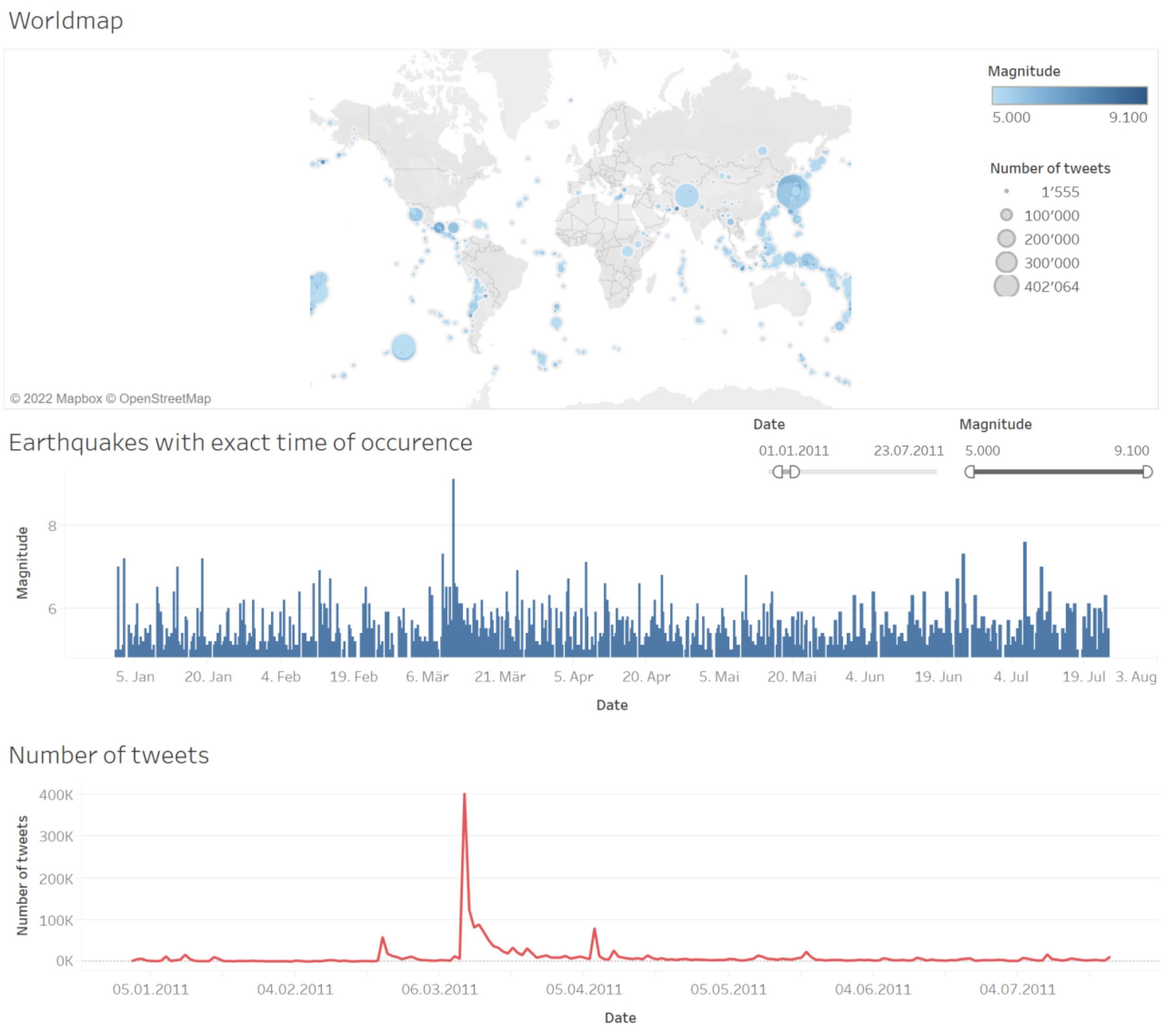


Figure 24 - GIVE TITLE

The dashboards was created on tableau an is showing three graphics. They are briefly described in the following.

The dashboard can also be found online under the following link:

https://public.tableau.com/app/profile/sandro4133/viz/earthquake_twitter_dashboard_final/Dashboard1?publish=yes

6.1 Individual graphics

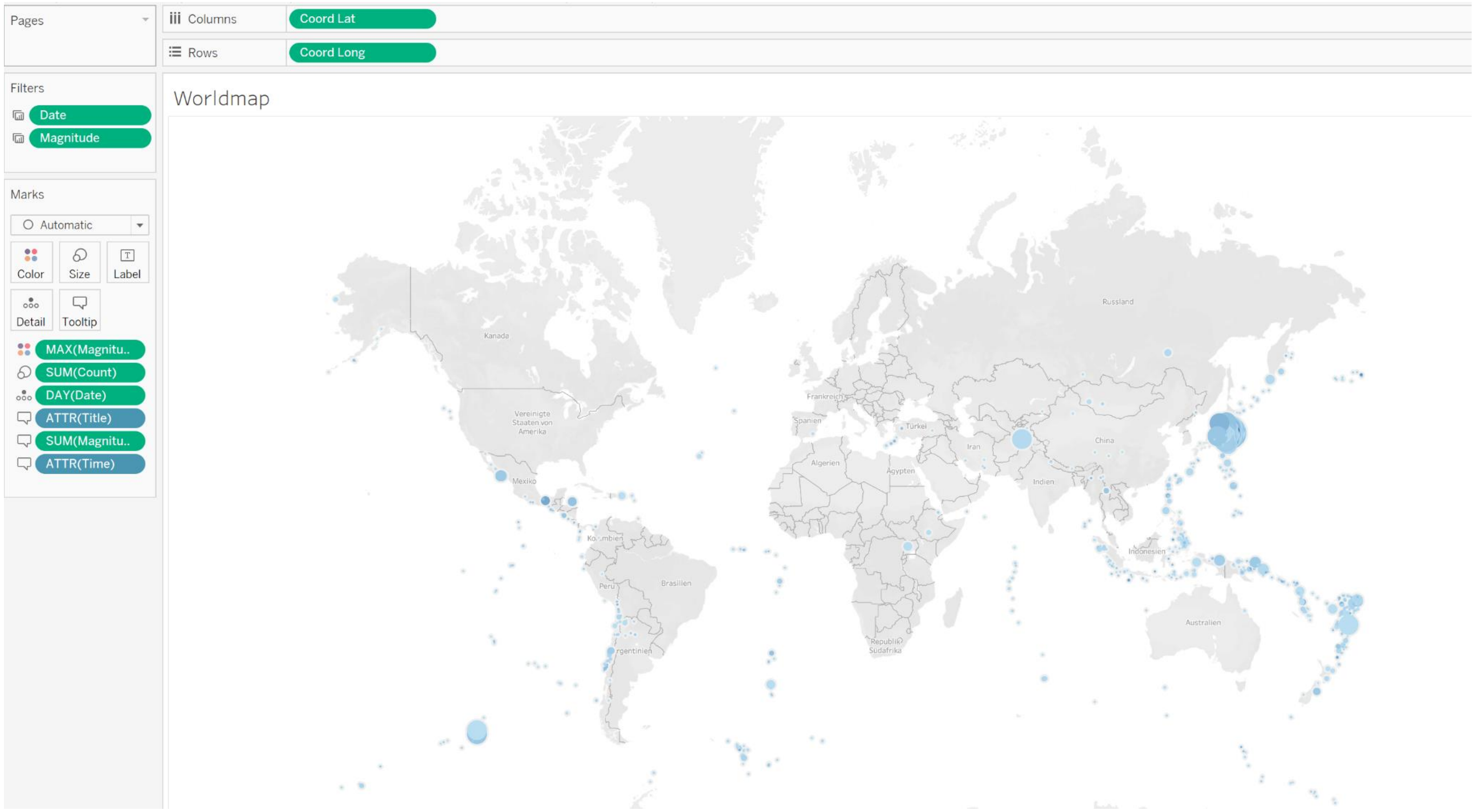
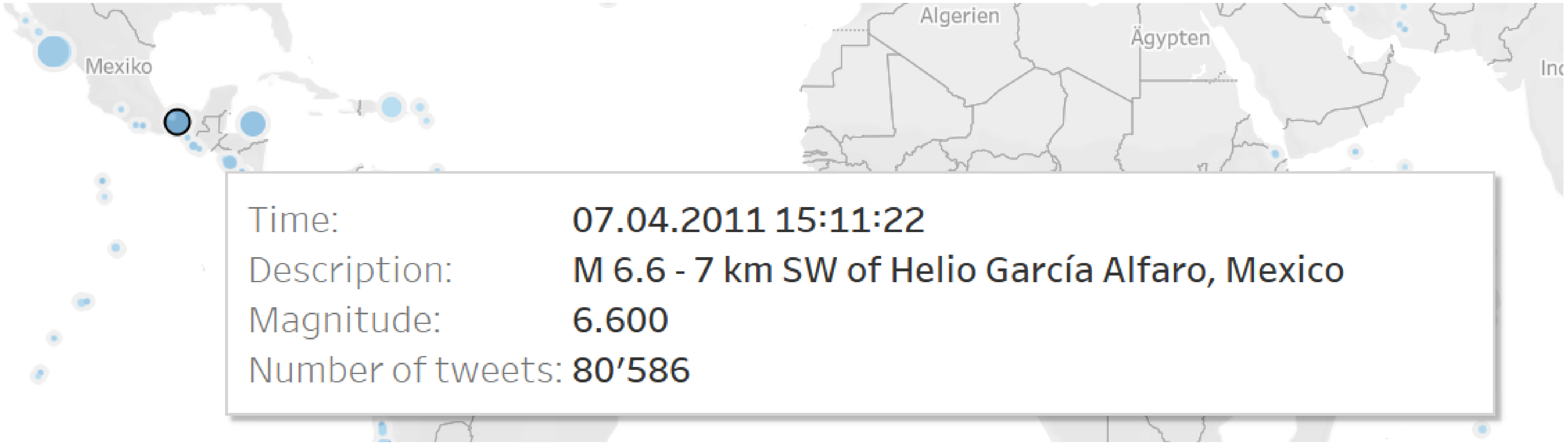


Figure 25 - GIVE TITLE

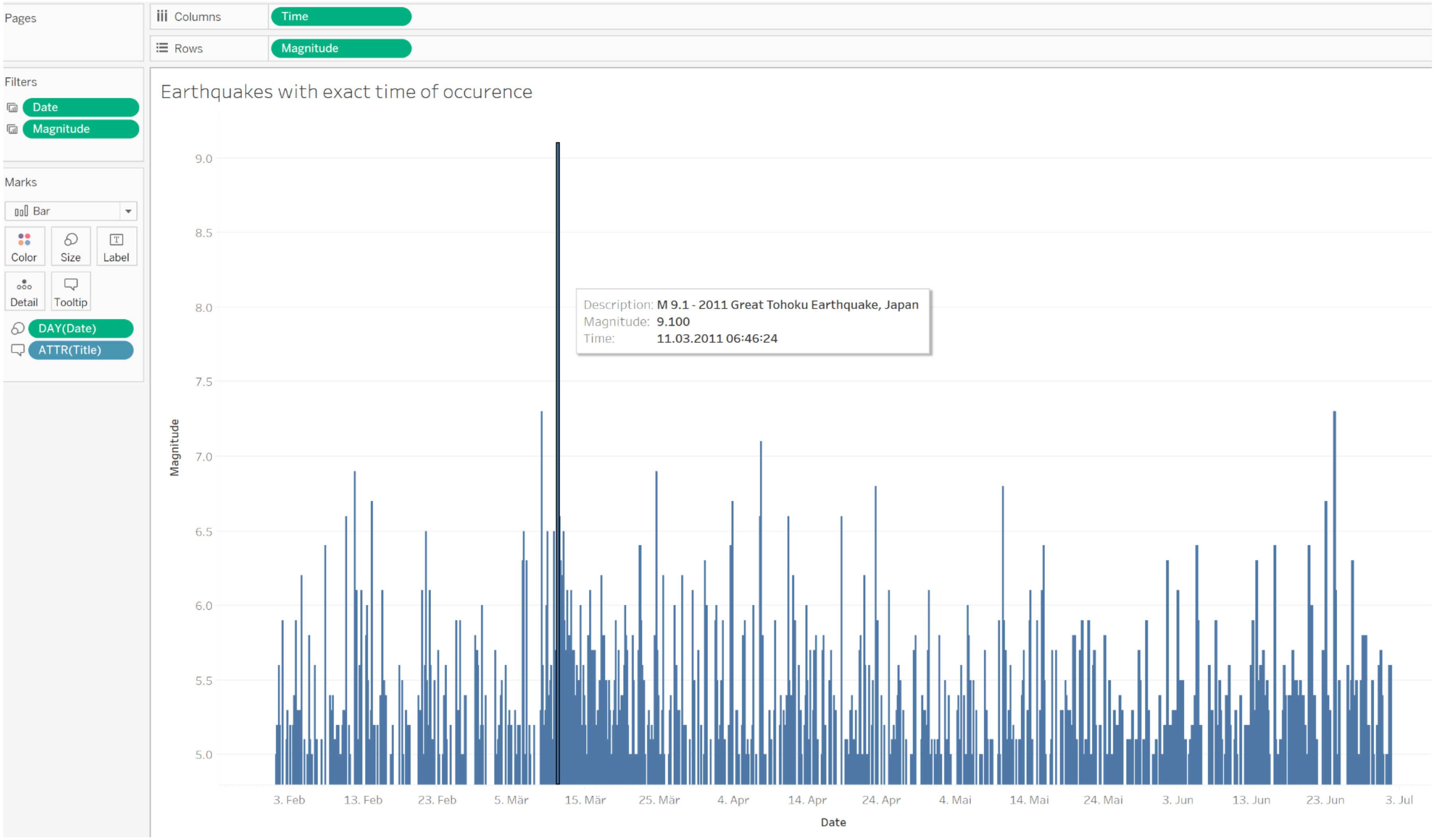
The worldmap show where earthquakes were occuring on a map with help of blue dots. The colour indicates how severe the earthquake was, a light blue indicates a low magnitude and a dark blue indicates a high magnitude. The size of the dots reflect the twitter echo - the bigger the dot, the more tweets has been sent at the respective date. Additionally, when hovering over the dots, additional information is revealed:



Time:	07.04.2011 15:11:22
Description:	M 6.6 - 7 km SW of Helio García Alfaro, Mexico
Magnitude:	6.600
Number of tweets:	80’586

Figure 26 - GIVE TITLE

In the next visualization, each earthquake is listed, structured to the exact time of occurence. Also here when hovering over an earthquake, more information is revealed:



Description: M 9.1 - 2011 Great Tohoku Earthquake, Japan
Magnitude: 9.100
Time: 11.03.2011 06:46:24

Figure 27 - GIVE TITLE

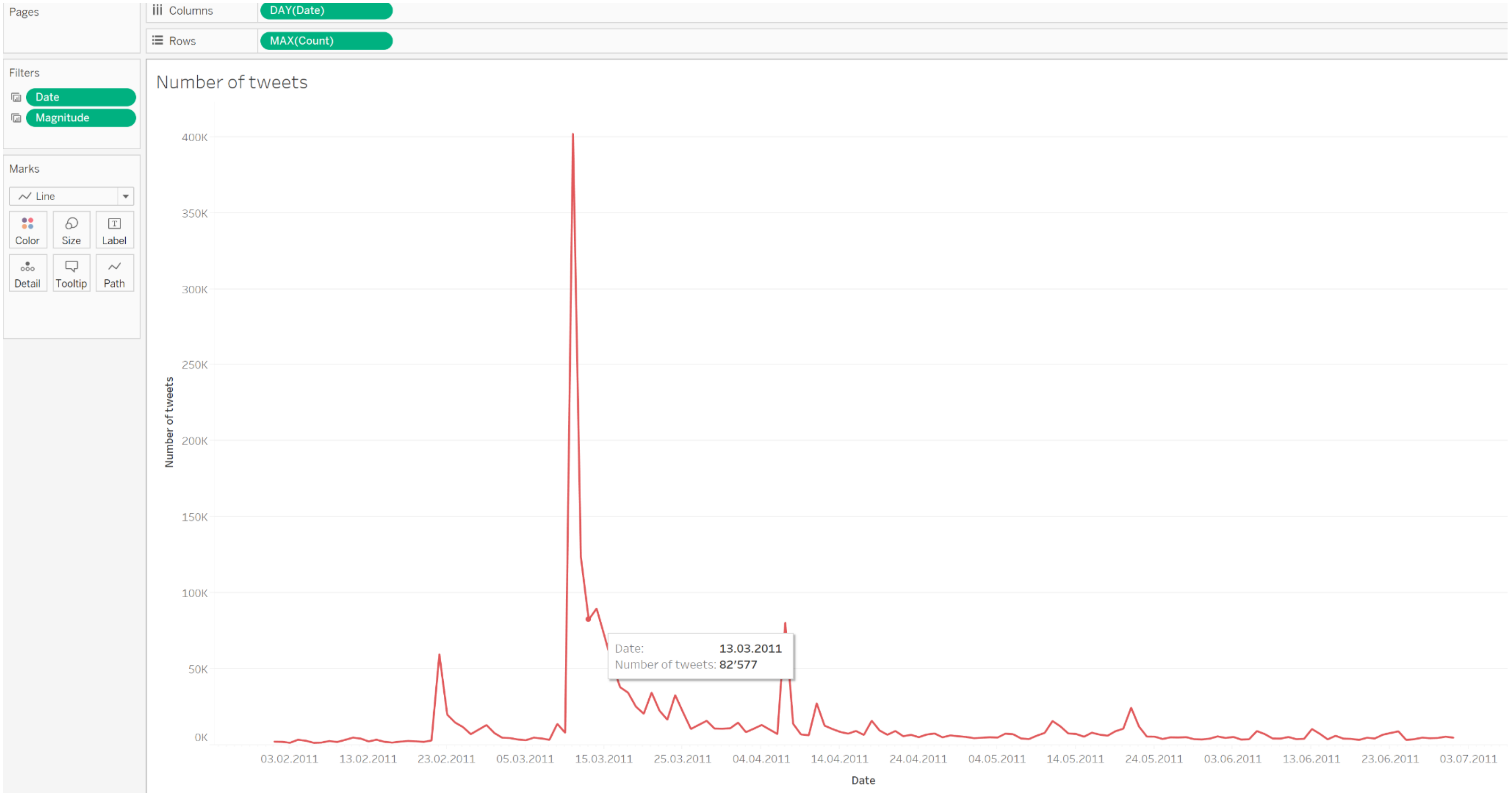


Figure 28 - GIVE TITLE

This chart finally shown the number of tweets sent per day, with numerical information revealed by hovering over it.

6.2 Dashboard filters

The dashboard can be filtered with two main indicators, those are the date and the magnitude:

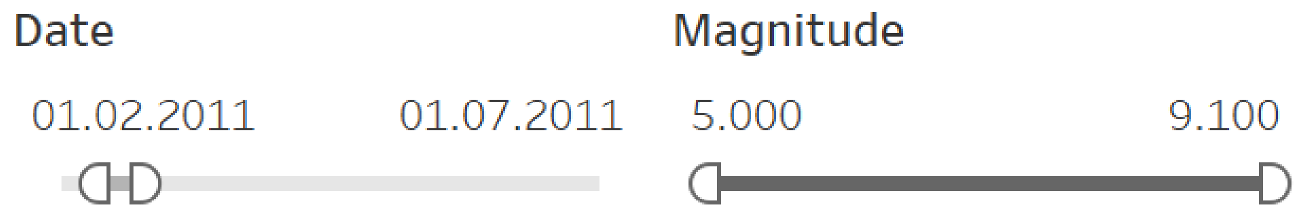
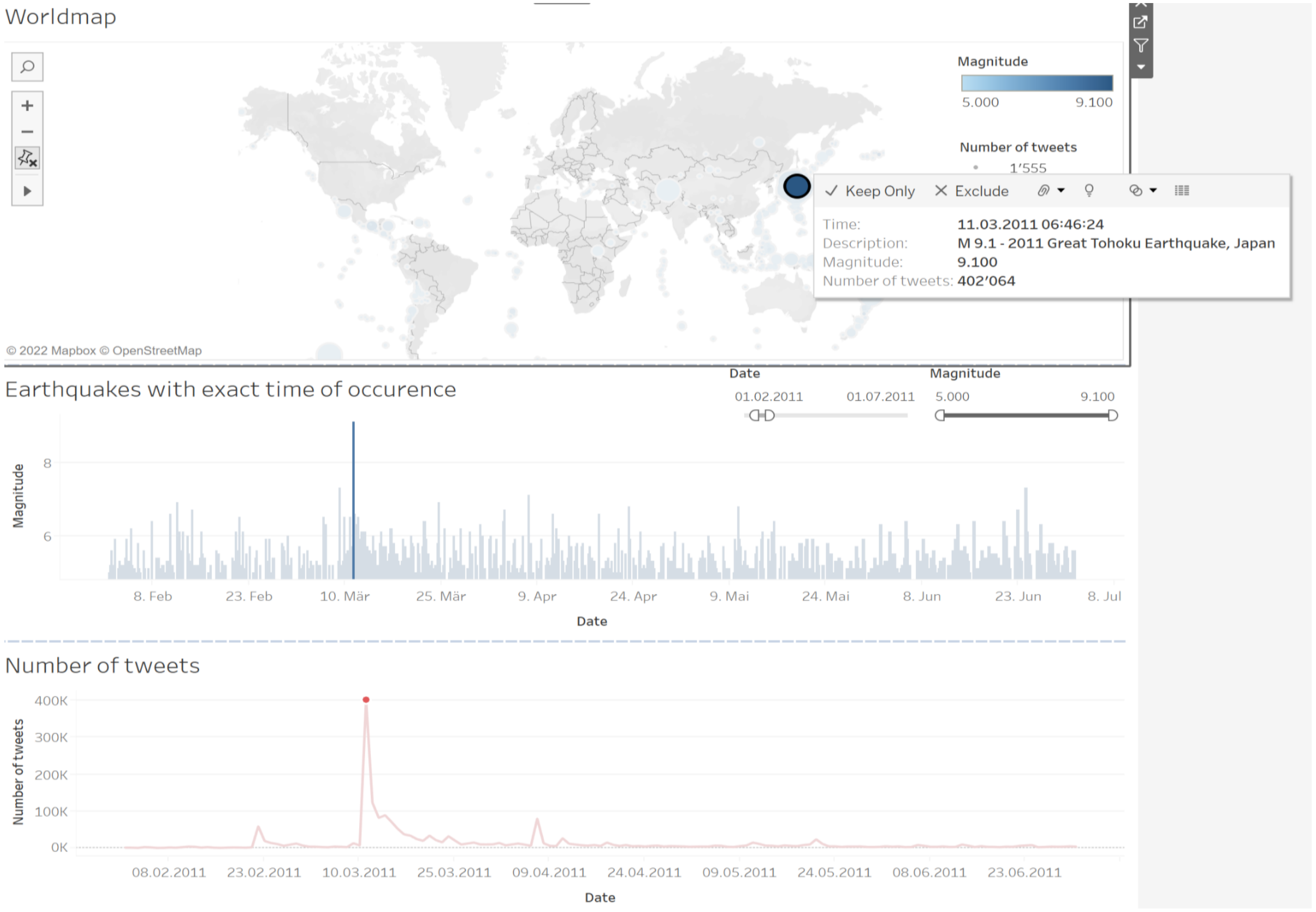


Figure 29 - GIVE TITLE

The reader can easily filter out earthquakes that were happening in a certain timeframe and can for example also analyse how many tweets were sent for earthquakes with a certain magnitude. Also other questions can easily be answered by applying these filters. Additionally, there are other filters and highlighters available on the board.

When clicking on a point on the map, the other graphics get adjusted as well and the concerning earthquake is highlighted.



Time: 11.03.2011 06:46:24
Description: M 9.1 - 2011 Great Tohoku Earthquake, Japan
Magnitude: 9.100
Number of tweets: 402'064

Figure 30 - GIVE TITLE

When clicking on a date from the number of tweets-graphic, this day gets filtered in the other graphics.

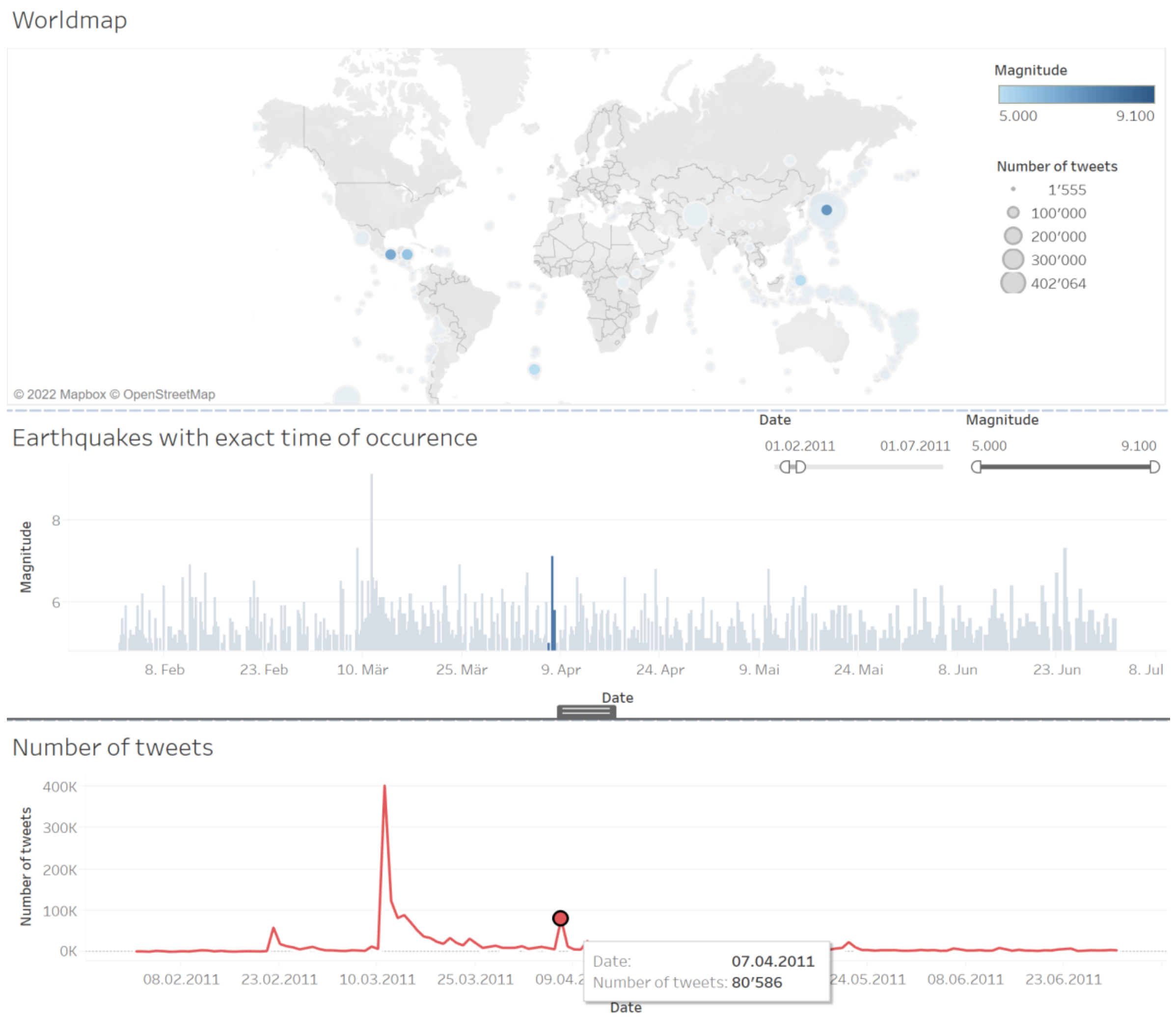


Figure 31 - GIVE TITLE

6.3 Project organization

According to the recommendation of the lecturers, a local prototype in Jupyter Notebook is created first. This allows the students to familiarize themselves with the APIs. In addition, for the time being it is only possible to get a certain extract from the complete data set. This is especially important for Twitter, where the number of tweets per developer account is limited.

The local prototype will later be migrated to Apache Airflow, where it will regularly (presumably once a day) retrieve the latest data from the API and store it in the database on Amazon Web Services.

6.4 Used technology

In the following illustration, the project's high level architecture can be seen. Beside a rough overview of the steps that the data is aiming to take, also the tools that are planned to be applied are visualized.

The highlighted part shows the accomplished steps in the first half of the semester.

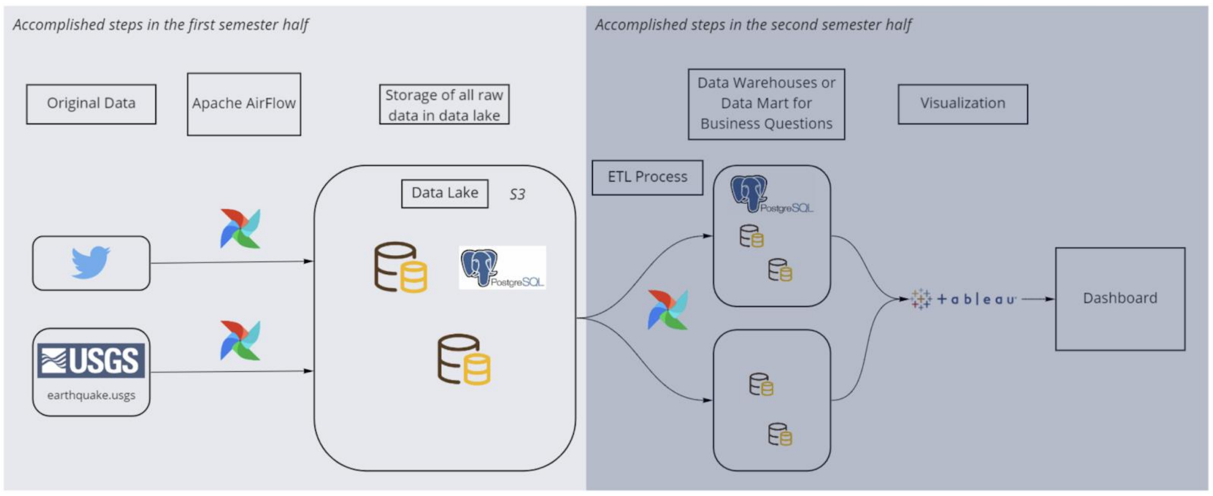

Accomplished steps in the first semester half
Accomplished steps in the second semester half
Original Data
Apache AirFlow
Storage of all raw data in data lake
Data Warehouses or Data Mart for Business Questions
Visualization
ETL Process
Data Lake
S3
PostgreSQL
USGS
earthquake.usgs
PostgreSQL
tableau
Dashboard

Figure 32 - Architecture

6.5 Data visualization

blabla tableau

screenshots tableau

evt. link to interactive dashboard version?

7 Project plan

The following Gantt-chart provides a broad overview about the tasks that have been achieved so far and that are aimed to be accomplished throughout this project. The red points are showing important milestones in this project, such as the finalization of the paper, the creation of data warehouse, dashboards and more.

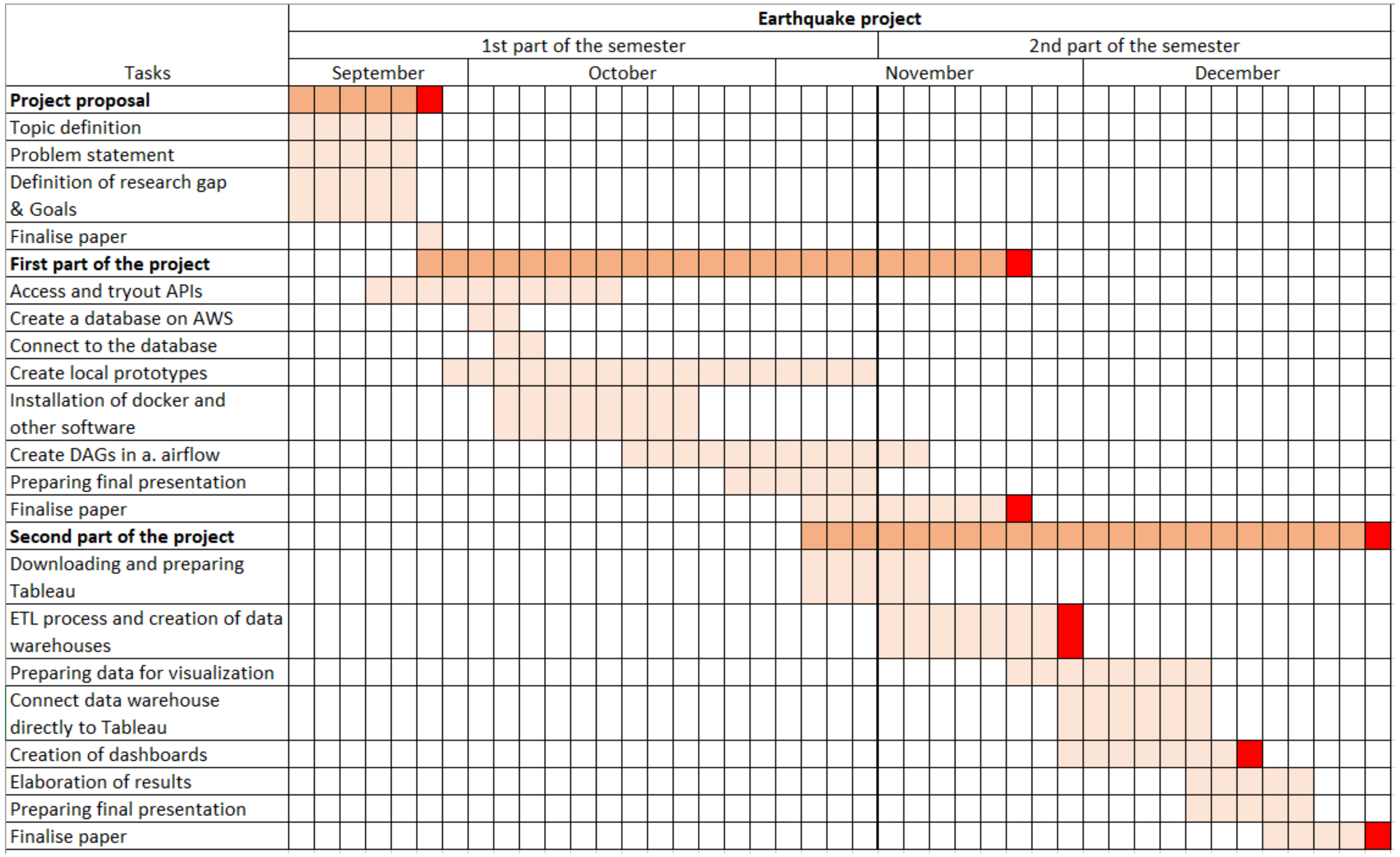


Figure 33 - Project Plan

8 Conclusions

9 Collaboration

The project is driven by the three team members simultaneously. The Scrum framework helps with this: During regular meetings on site or online, current challenges and upcoming steps are discussed. Although everyone is driving their own sub-projects, great importance is attached to ensuring that all members come into contact with all technologies. This is not the most efficient way to reach the goal, but because all team members have little experience with the technologies presented in this module, it seems to be the best learning path.

Project organization files - namely the proposals, documentation and schedules - are stored on a shared folder in the Switch-Drive cloud environment and backed up regularly.

Code and data are stored and versioned in a Github repository. This allows the individual work steps to be documented and tracked as needed. The recommended steps covered in class were included, such as creating a .gitignore file. The README.md file provides a quick overview of the project.

10 Project limitations

In keeping with the curriculum of this course, the focus will be on exploring new technologies, such as Apache Airflow, Amazon Web Services, Lambda Functions, DBT, and so on. In answering the research questions, some limitations will therefore be accepted.

For example, only tweets containing the word "earthquake" will be intercepted. Of course, there are also tweets in languages other than English, which then contain corresponding translations of this term. In addition, there are various synonyms or related terms, such as "shake" or "quake", which are also not included in the analysis. In the future, the search query could be extended with additional keywords.

Furthermore, the focus is not on sentiment analysis and NLP, because that would be out of scope of this course. With a short analysis, the potential of this methodology is hinted at, but further evaluations are postponed to a later date.

11 Sources

Avramakis, E., Anchen, J., & Raverkar, A. K. (2019, January). *Digital ecosystems: extending the boundaries of value creation in insurance.* Swiss Re Management Ltds.

Benita, H. (2019, July 9). *Haki Benita*. Retrieved at 08.11.2021 from https://hakibenita.com/fast-load-data-python-postgresql

Brandwatch (2020, January 2). *60 Incredible and Interesting Twitter Stats and Statistics.* Retrieved at 10.11.2021 from https://www.brandwatch.com/blog/twitter-stats-and-statistics/

Elaine (2015, October 7). *How the USGS uses Twitter data to track earthquakes.* Retrieved at 08.11.2021 from https://blog.twitter.com/en_us/a/2015/usgs-twitter-data-earthquake-detection

Earle, Paul. Bowden, Daniel. Guy, Michelle (2011). *Twitter earthquake detection: Earthquake monitoring in a social world.* Retrieved at 05.11.2021 from https://pubs.er.usgs.gov/publication/70006356

SwissRe. (2020, December 15). *Swiss Re Institute estimates USD 83 billion global insured catastrophe losses in 2020, the fifth-costliest on record* Retrieved at 01.11.2021 from https://www.swissre.com/media/news-releases/nr-20201215-sigma-full-year-2020-preliminary-natcat-loss-estimates.html

USGS (2010, May 24). *Earthquake Facts and Statistics.* Retrieved at 15.10.2021 from https://web.archive.org/web/20100524161817/http://earthquake.usgs.gov/earthquakes/eqarchives/year/eqstats.php

11.1 Images

Figure 1: Swiss Re's insured losses chart: https://www.swissre.com/media/news-releases/nr-20201215-sigma-full-year-2020-preliminary-natcat-loss-estimates.html

Figure 2: Swiss Re's insured losses chart: https://www.swissre.com/media/news-releases/nr-20201215-sigma-full-year-2020-preliminary-natcat-loss-estimates.html

Figure 3: Earthquake dashboard by USGS: https://blog.twitter.com/en_us/a/2015/usgs-twitter-data-earthquake-detection

Figure 15: Wikipedia Map of Earthquakes 1900 – 2017: https://commons.wikimedia.org/wiki/File:Map_of_earthquakes_1900-.svg

Figure 16: Animated timeline of covid infections on «Our world in data»: https://ourworldindata.org/explorers/coronavirus-data-explorer?tab=map&zoomToSelection=true&time=2021-07-09&facet=none&pickerSort=asc&pickerMetric=location&Metric=Confirmed+cases&Interval=7-day+rolling+average&Relative+to+Population=true&Align+outbreaks=false&country=USA~GBR~CAN~DEU~ITA~IND

Tutorial UNIX Timestamp

https://www.youtube.com/watch?v=HpTjmUWW_lk&ab_channel=AlexanderKarsonis