Swisscom Project

December 21, 2022

Swisscom Shop Report

Contents

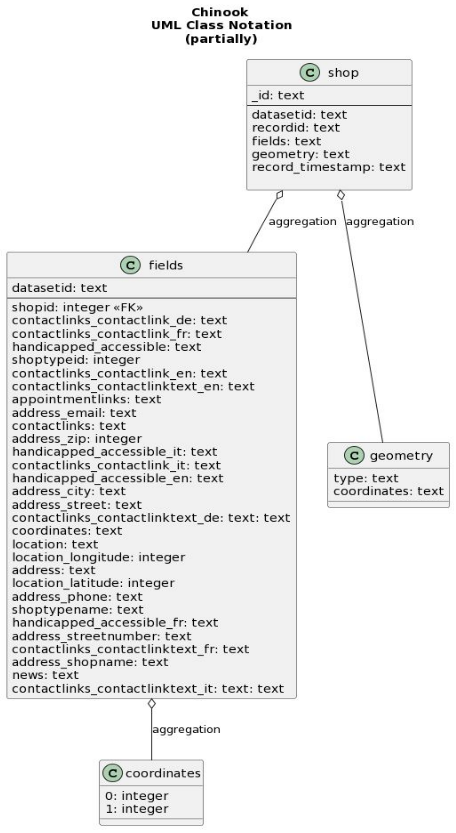
1 Introduction

Swisscom is the leading provider of telecommunication services in Switzerland. It offers a wide range of products and services, including broadband internet, fixed-line telephone service, mobile phone service, digital television and IP-TV. Swisscom also provides IT solutions for businesses as well as cloud computing services. Additionally, it operates 120 retail stores throughout the country where customers can purchase devices such as smartphones, tablets and other products and services. https://www.swisscom.ch/en/about.html

The aim of this report is to demonstrate how to visualise data gathered from the Swisscom Shop API using Python and MongoDB. The Swisscom Shop API used for this project can be found on Swisscom in the *Explore* section. The API contains basic information on Swisscom-owned shops and certified retail partners, these are:

- name
- address
- phone number
- fax
- geographic coordinates for map positioning

Important to know is to define the criteria before one accesses the data. For instance, in the field *row* change 10 to 120, if not only 10 shops will be accessible with the API. Below the structure of the documents in MongoDB are illustrated.



The report will start with setting up the connection requirements of the Mongo Database, followed by the ETL process, after that the data will be visualised and a conclusion will end the report.

2 Requirements & Configuration

```
[2]: ! pip3 list | findstr "pymongo dnspython pandas"
```

```
dnspython                          2.2.1
pandas                             1.5.1
pymongo                            4.3.2
```

```
[5]: import pymongo
     from pprint import pprint
     import pandas as pd
     import requests
     import json
```

```
[6]: # API and Database details
     API_URL = "https://data.swisscom.com/api/records/1.0/search/?
      ↪dataset=swisscom-shops
     -de&q=&rows=121&sort=address_shopname&facet=address_city&facet=address_zip"
     CNX_STR = "mongodb+srv://cluster0:abcD3@cluster0.wwzamb6.mongodb.net/test"
     DB_NAME = "swisscom"
     COLL_NAME = "shop"
```

```
[7]: # connection to MongoDB
     client = pymongo.MongoClient(CNX_STR)
     db = client[DB_NAME]
     shop = db[COLL_NAME]
     dbs = pd.DataFrame(client.list_databases())
     dbs
```

```
[7]:            name  sizeOnDisk  empty
     0       genshin       40960  False
     1      swisscom      159744  False
     2  swisscom_shop       40960  False
     3         admin      344064  False
     4         local  6690467840  False
```

3 ETL

3.1 Remove all existing documents -> Reset collection

```
[8]: shop.drop()
     shop.count_documents({})
     #
```

```
[8]: 0
```

3.2 Fetch data

```
[9]: # fetch JSON from API_URL
     r = requests.get(API_URL)
     data = json.loads(r.text)
```

```
[10]: print(r.text[0:500])
```

```
{"nhits": 6687, "parameters": {"dataset": "swisscom-shops-de", "rows": 121,
"start": 0, "sort": ["address_shopname"], "facet": ["address_city",
"address_zip"], "format": "json", "timezone": "UTC"}, "records": [{"datasetid":
"swisscom-shops-de", "recordid": "c93a94936092144bb1c2082431a570b4ed1ecd05",
"fields": {"shopid": 11063, "contactlinks_contactlink_de":
"http://www.wattcom.ch", "contactlinks_contactlink_fr": "http://www.wattcom.ch",
"handicapped_accessible": "nein", "shoptypeid": 3, "contact
```

3.3 Insert into MongoDB

```
[11]: # insert the list of shops in "records" into MongoDB collection
      shop.insert_many(data['records']);
```

```
[12]: # count number of documents in shop collection
      shop.count_documents({})
```

```
[12]: 121
```

```
[13]: # check one document in shop collection
      pprint(db.shop.find_one())
```

```
{'_id': ObjectId('63a32ad9973d251067cc5a63'),
 'datasetid': 'swisscom-shops-de',
 'fields': {'address': '{}{}',
            'address_city': 'Genève',
            'address_email': '--',
            'address_phone': '+41-22-702 92 62',
            'address_shopname': '1it4u Sàrl',
            'address_street': 'rte de Malagnou',
            'address_streetnumber': '6',
            'address_zip': '1208',
            'appointmentlinks': '{"AppointmentLinkDe": "", '
                                '"AppointmentLinkFr": "", "AppointmentLinkIt": '
                                '"", "AppointmentLinkEn": ""}',
            'contactlinks': '{"ContactLinkDe": "http://www.wattcom.ch", '
                            '"ContactLinkFr": "http://www.wattcom.ch", '
                            '"ContactLinkIt": "http://www.wattcom.ch", '
                            '"ContactLinkEn": "http://www.wattcom.ch", '
                            '"ContactLinkTextDe": "Website öffnen", '
```

```
                              '"ContactLinkTextFr": "Voir le site", '
                              '"ContactLinkTextIt": "Guardare il sito", '
                              '"ContactLinkTextEn": "Open Website"}',
            'contactlinks_contactlink_de': 'http://www.wattcom.ch',
            'contactlinks_contactlink_en': 'http://www.wattcom.ch',
            'contactlinks_contactlink_fr': 'http://www.wattcom.ch',
            'contactlinks_contactlink_it': 'http://www.wattcom.ch',
            'contactlinks_contactlinktext_de': 'Website öffnen',
            'contactlinks_contactlinktext_en': 'Open Website',
            'contactlinks_contactlinktext_fr': 'Voir le site',
            'contactlinks_contactlinktext_it': 'Guardare il sito',
            'coordinates': [46.1984332086195, 6.15739486264927],
            'handicapped_accessible': 'nein',
            'handicapped_accessible_en': 'no',
            'handicapped_accessible_fr': 'non',
            'handicapped_accessible_it': 'no',
            'location': '{"Latitude": 46.1984332086195, "Longitude": '
                        '6.15739486264927}',
            'location_latitude': 46.1984332086195,
            'location_longitude': 6.15739486264927,
            'news': '{"NewsDe": "", "NewsFr": "", "NewsIt": "", "NewsEn": ""}',
            'shopid': 11063,
            'shoptypeid': 3,
            'shoptypename': 'Partner-Shop'},
 'geometry': {'coordinates': [6.15739486264927, 46.1984332086195],
              'type': 'Point'},
 'record_timestamp': '2022-12-21T04:55:39.346Z',
 'recordid': 'c93a94936092144bb1c2082431a570b4ed1ecd05'}
```

So, in the output above one can clearly see, that there are sub-documents *fields* and *geometry*. Apart from that, the output below also confirms the assumption, that those two objects (columns below) hold all important information.

Needless to say the *appointmentlinks* and *contactlinks* objects within the *fields* document are not relevant for this project and therefore, are going to be dropped. Although the coordinates are important, they are present three times, once as a *geometry* sub-document, and twice within the *fields* sub-document, as a value and another as a *location* object array. Once is enough hence only, *location_ latitude* and *location_ longitude* are going to remain.

[14]:
```
#Displaying the shop collection in a dataframe to see which "information" is␣
↪needed
c = shop.aggregate([
      {"$limit": 2},
])

pd.DataFrame(c)
```

```
[14]:                         _id          datasetid  \
      0  63a32ad9973d251067cc5a63  swisscom-shops-de
      1  63a32ad9973d251067cc5a64  swisscom-shops-de

                                 recordid  \
      0  c93a94936092144bb1c2082431a570b4ed1ecd05
      1  818d3a46fe155400bc2402c4755b0c86e1cfcd22

                                                  fields  \
      0  {'shopid': 11063, 'contactlinks_contactlink_de...
      1  {'shopid': 7774, 'contactlinks_contactlink_de'...

                                                geometry          record_timestamp
      0  {'type': 'Point', 'coordinates': [6.1573948626...  2022-12-21T04:55:39.346Z
      1  {'type': 'Point', 'coordinates': [8.3075577523...  2022-12-21T04:55:39.346Z
```

```
[15]: # assign recordid to _id and remove id
      c = shop.aggregate([
          {"$project": {"_id": "$recordid", "datasetid": 1, "fields": 1, "geometry":␣
       ↪1, "record_timestamp": 1}},
      ])
      b = pd.DataFrame(c)
      b.head()
```

```
[15]:           datasetid                                             fields  \
      0  swisscom-shops-de  {'shopid': 11063, 'contactlinks_contactlink_de...
      1  swisscom-shops-de  {'shopid': 7774, 'contactlinks_contactlink_de'...
      2  swisscom-shops-de  {'shopid': 11045, 'contactlinks_contactlink_de...
      3  swisscom-shops-de  {'shopid': 7676, 'contactlinks_contactlink_de'...
      4  swisscom-shops-de  {'shopid': 7931, 'contactlinks_contactlink_de'...

                                                geometry  \
      0  {'type': 'Point', 'coordinates': [6.1573948626...
      1  {'type': 'Point', 'coordinates': [8.3075577523...
      2  {'type': 'Point', 'coordinates': [6.1445328248...
      3  {'type': 'Point', 'coordinates': [8.3921801015...
      4  {'type': 'Point', 'coordinates': [8.6786824883...

                 record_timestamp                                       _id
      0  2022-12-21T04:55:39.346Z  c93a94936092144bb1c2082431a570b4ed1ecd05
      1  2022-12-21T04:55:39.346Z  818d3a46fe155400bc2402c4755b0c86e1cfcd22
      2  2022-12-21T04:55:39.346Z  0ec39d41758ff9ac39a74c1d5d1cf7cf0566ebea
      3  2022-12-21T04:55:39.346Z  d86cba3cca1c270d4ad87af048a13cf500a3b263
      4  2022-12-21T04:55:39.346Z  a9b4c11a56b529082cac76cdef0bad79b7f27c0d
```

3.4 Transform

As mentioned in the introduction the api contains information on Swisscom-owned shops and certified retail partners. Therefore, a list will be created containing the two different types of shops, this will then be presented in a dataframe to then drop the columns which are not needed.

[16]:
```
# Finding Partner Shops nested in shop collection
#for x in shop.find({"fields.shoptypename":"Partner-Shop"}):
    #pprint(x)
```

[17]:
```
#Finding Shops that are not Partner Shops
#for j in shop.find({"fields.shoptypename":{'$ne':"Partner-Shop"}}):
    #pprint(j)   # so it is "Distributor"
```

The two loops above find specific shop types and the aggregation below counts the amount of distinct shop types. As we can see there are three different shop types not two as mentioned in the api information.

[18]:
```
#Count and print how distinct shoptypes
cursor = shop.aggregate([
    { '$unwind':'$fields' },
    { '$group':{'_id':'$fields.shoptypename', 'shop_type':{'$sum':1}}},
    { '$sort':{ "_id": 1 } }
]);
b = pd.DataFrame(cursor)
b
```

[18]:
```
                _id  shop_type
0       Distributor          4
1      Partner-Shop        114
2  SC Swisscom Shop          3
```

3.5 Unwind nested array

Unwind using python First, it was done with python to a dataframe, thanking Mr Fugu Data Science for providing the video and giving a hint on how the data frame should look like.

[19]:
```
# Find and store the 2 different shops in a list
shop_sp = []
for y in shop.find({'$or':[{'fields.shoptypename': 'Distributor'},
                          {'fields.shoptypename': 'Partner-Shop'},
                          {'fields.shoptypename': 'SC Swisscom Shop'}]}) :
    shop_sp.append(y)

# call a news list and iterate through the previous one in order to create the␣
 ↪data frame
nested_fields = []
only_ids =[]
for y in shop_sp:
    nested_fields.append(y["fields"])
    only_ids.append(y["_id"])

shop_fields = pd.DataFrame(nested_fields)

#add a new column with the id
shop_fields['_id'] = only_ids
shop_fields.head(2)
```

[19]:

```
  shopid contactlinks_contactlink_de contactlinks_contactlink_fr  \
0  11063       http://www.wattcom.ch       http://www.wattcom.ch
1   7774    http://www.1solution.ch/    http://www.1solution.ch/

  handicapped_accessible  shoptypeid contactlinks_contactlink_en  \
0                   nein           3       http://www.wattcom.ch
1                   nein           3    http://www.1solution.ch/

  contactlinks_contactlinktext_en  \
0                    Open Website
1                    Open Website

                                   appointmentlinks address_email  \
0  {"AppointmentLinkDe": "", "AppointmentLinkFr":...            --
1  {"AppointmentLinkDe": "", "AppointmentLinkFr":...            --

                                       contactlinks  ...  \
0  {"ContactLinkDe": "http://www.wattcom.ch", "Co...  ...
1  {"ContactLinkDe": "http://www.1solution.ch/", ...  ...

  contactlinks_contactlinktext_fr address_shopname  \
0                    Voir le site       1it4u Sàrl
1                    Voir le site     1solution AG

                                               news  \
0  {"NewsDe": "", "NewsFr": "", "NewsIt": "", "Ne...
1  {"NewsDe": "", "NewsFr": "", "NewsIt": "", "Ne...

  contactlinks_contactlinktext_it       address_fax  \
0                Guardare il sito               NaN
1                Guardare il sito  +41-56-485 76 99

  appointmentlinks_appointmentlink_fr appointmentlinks_appointmentlink_de  \
0                                 NaN                                 NaN
1                                 NaN                                 NaN

  appointmentlinks_appointmentlink_it appointmentlinks_appointmentlink_en  \
0                                 NaN                                 NaN
```

```
1                                 NaN                                  NaN

                         _id
0  63a32ad9973d251067cc5a63
1  63a32ad9973d251067cc5a64

[2 rows x 36 columns]
```

[20]:
```
#32 columns
#shop_fields.info()
```

[21]:
```
#Drop multiple columns
shop_fields.drop(['contactlinks_contactlink_de',␣
 ↪'contactlinks_contactlink_fr','contactlinks_contactlink_en',
               ␣
 ↪'contactlinks_contactlinktext_en','appointmentlinks','handicapped_accessible_it
','contactlinks_contactlink_it',
                 'contactlinks_contactlinktext_de','handicapped_accessible_fr
','contactlinks_contactlinktext_it',
               ␣
 ↪'contactlinks_contactlinktext_fr',"location",'handicapped_accessible',
                 'address', 'news','address_email', 'contactlinks'], axis = 1,␣
 ↪inplace = True)
```

[22]:
```
shop_fields.head(2)
```

[22]:
```
   shopid  shoptypeid address_zip handicapped_accessible_en    address_city  \
0   11063           3        1208                        no          Genève
1    7774           3        5443                        no  Niederrohrdorf

    address_street                               coordinates  location_longitude  \
0  rte de Malagnou  [46.1984332086195, 6.15739486264927]            6.157395
1          Loonstr.  [47.4234399326177, 8.30755775231435]            8.307558

   location_latitude     address_phone  shoptypename address_streetnumber  \
0          46.198433  +41-22-702 92 62  Partner-Shop                    6
1          47.423440  +41-56-485 76 50  Partner-Shop                   6A

  address_shopname       address_fax appointmentlinks_appointmentlink_fr  \
0       1it4u Sàrl               NaN                                 NaN
1     1solution AG  +41-56-485 76 99                                 NaN

  appointmentlinks_appointmentlink_de appointmentlinks_appointmentlink_it  \
0                                 NaN                                  NaN
1                                 NaN                                  NaN

  appointmentlinks_appointmentlink_en                         _id
```

```
0                                  NaN  63a32ad9973d251067cc5a63
1                                  NaN  63a32ad9973d251067cc5a64
```

The *coordinates* column will be kept just in case. Now this dataframe will be saved and uploaded to MongoDB as a new collection called *shopclean*.

```
[245]: #save df to csv
       shop_fields.to_csv('shopclean1.csv', index=False, header=True)
```

```
[27]: #Load csv
      data = pd.read_csv('shopclean1.csv')
```

```
[28]: #Upload the dataframe to mongodb as a new collection
      DB_NAME = "swisscom"
      COLL_NAME = "shopclean"
      shopclean = db[COLL_NAME]
```

```
[29]: shopclean.drop()
      shopclean.count_documents({})
```

```
[29]: 0
```

```
[30]: data.reset_index(inplace=True)

      # Insert collection
      data_dict = data.to_dict("records")
      shopclean.insert_many(data_dict)
```

```
[30]: <pymongo.results.InsertManyResult at 0x1fa6f7378e0>
```

Unwind *fields* sub-document using pymongo

```
[31]: agg_shopfields = shop.aggregate([
          {"$project":{"shopid": "$fields.shopid",
                       "shoptypeid": "$fields.shoptypeid",
                      "address_zip": "$fields.address_zip",
                      "handicapped_accessible_en": "$fields.handicapped_accessible_en",
                      "address_city": "$fields.address_city",
                      "address_street": "$fields.address_street",
                      "coordinates": "$fields.coordinates",
                      "location_longitude": "$fields.location_longitude",
                      "location_latitude": "$fields.location_latitude",
                      "address_phone" : "$fields.address_phone",
                      "shoptypename" : "$fields.shoptypename",
                      "address_streetnumber" : "$fields.address_streetnumber",
                      "address_shopname": "$fields.address_shopname",
                      "address_fax": "$fields.address_fax"
          }}
      ]);
```

```
pyclean = pd.DataFrame(agg_shopfields)
pyclean.head(4)
```

[31]:
```
                        _id  shopid  shoptypeid address_zip  \
0  63a32ad9973d251067cc5a63   11063           3        1208
1  63a32ad9973d251067cc5a64    7774           3        5443
2  63a32ad9973d251067cc5a65   11045           3        1205
3  63a32ad9973d251067cc5a66    7676           3        8966

  handicapped_accessible_en    address_city   address_street  \
0                        no          Genève  rte de Malagnou
1                        no  Niederrohrdorf          Loonstr.
2                        no          Genève   rue SAINT-OURS
3                        no   Oberwil-Lieli          Jurastr.

                              coordinates  location_longitude  \
0  [46.1984332086195, 6.15739486264927]            6.157395
1  [47.4234399326177, 8.30755775231435]            8.307558
2  [46.1977378531502, 6.14453282483435]            6.144533
3  [47.3419177427353, 8.39218010156182]            8.392180

   location_latitude     address_phone  shoptypename address_streetnumber  \
0          46.198433  +41-22-702 92 62  Partner-Shop                    6
1          47.423440  +41-56-485 76 50  Partner-Shop                   6A
2          46.197738  +41-22-752 08 71  Partner-Shop                    4
3          47.341918  +41-56-633 66 16  Partner-Shop                   11

                      address_shopname       address_fax
0                           1it4u Sàrl               NaN
1                        1solution AG  +41-56-485 76 99
2                           1SWISS1 SA               NaN
3  2COM Computer and Communication GmbH  +41-56-633 12 22
```

[ ]:
```
#select all rows with NaN values
pyclean[pyclean.isnull().any(axis=1)]

#_id 63a2dc2c37980972b0c4874e has NaN for coordinates, needs to be droped for␣
 ↪data visualisation
```

[33]:
```
#Dropping the row containing NaN value in longitude and latitude
pyclean = pyclean[pyclean.shopid != 7175]
```

4 Data analysis

In this chapter the basic data analysis will be conducted with the *shopclean* collection. The data visualisation will be done with the *shop* collection.

4.1 Shops per Canton

[34]:
```
c = shopclean.aggregate([
    {"$project": {"shopclean": 0}},
    {"$group": {"_id": "$address_city", "count": {"$sum": 1}}},
    { "$sort": { "count":-1 }},
 ])

cc = pd.DataFrame(c)
cc.head(6)
```

[34]:
```
          _id  count
0      Zürich     12
1  St. Gallen      5
2      Genève      4
3        Bern      4
4        Sion      3
5  Winterthur      3
```

4.2 Shops accessible for handycapped people

[35]:
```
b = shopclean.aggregate([
    {"$project": {"shopclean": 0}},
    {"$group" : {"_id":"$handicapped_accessible_en", "count": {"$sum":1}}},
])

pd.DataFrame(b)
```

[35]:
```
   _id  count
0   no    116
1  yes      5
```

4.3 Data Visualisation

[36]:
```
import folium
from folium import plugins
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

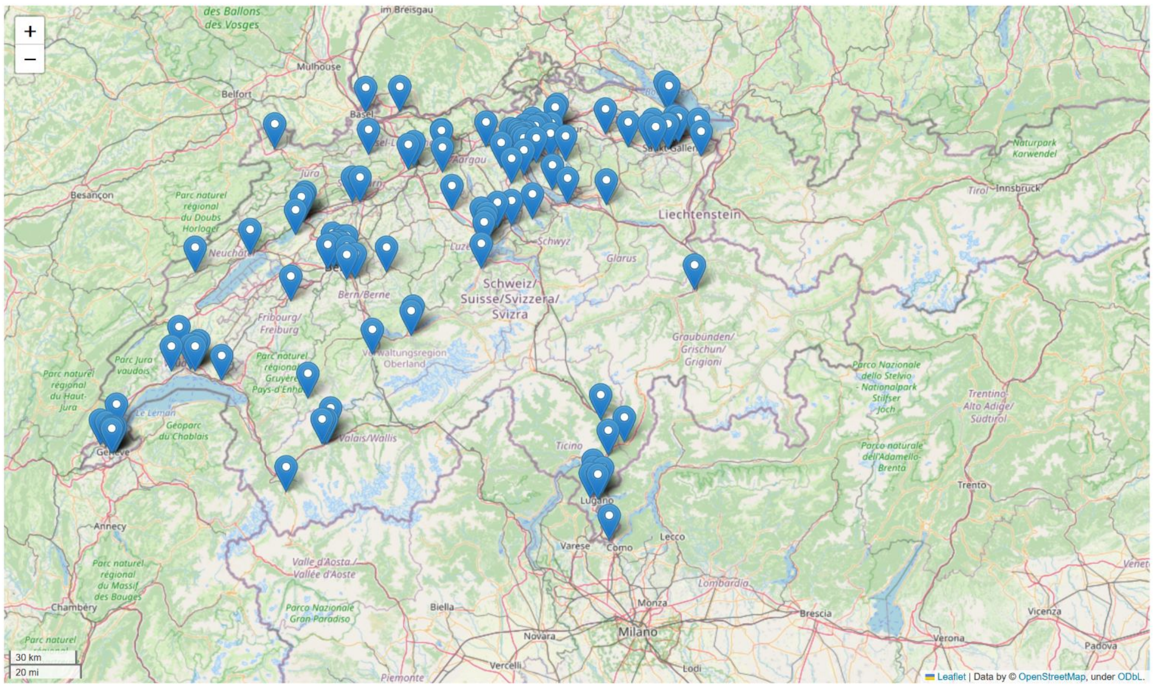
[41]:
```
#Use online specific columns
shop_locations = pyclean[["location_latitude", "location_longitude",␣
↪"shoptypename"]]

#Creating map
map = folium.Map(location=[shop_locations.location_latitude.mean(),␣
↪shop_locations.location_longitude.mean()],
                 zoom_start=7, control_scale=True)

#Adding points to map
for index, location_info in shop_locations.iterrows():
    folium.Marker([location_info["location_latitude"],␣
↪location_info["location_longitude"]],
                  popup=location_info["shoptypename"]).add_to(map)
map
```

[41]:
```
<folium.folium.Map at 0x1fa00dcc4c0>
```

Out[58]:



In the canton Graubünden there is only one shop in Chur and it is a Partner-shop which is not a Swisscom shop rather a certified partner.

5 Conclusions

In total there are 120 Swisscom shops in Switzerland. The location of these shops are well distributed and most of them are in key locations. 12 are in Zürich which was expected surprisingly, there are five in St. Gallen and for both Bern and Geneva there are only four shops. Shockingly, only five stores are accessible for handicapped people. So, either the data in english is not up-to-date or it is true. It would not make sense as most Swisscom shops are usually on ground floors and those should be accessible.

In general it would have been great if they had some sales numbers per shop, to further compare which one is more profitable and what could be the possible reasons behind that.

What was missing on their Swisscom website where the API can be retrieved, is a readme file. Most of the values were clear. However, there was a contradiction on their website regarding the shop type, there the examples Swisscom provided were *consumer electronics, store-in-store, etc.* and within the data it was *Partner-Shop*, *Distributor* and *SC Swisscom shop*. This was a bit confusing at first because I could not find the examples Swisscom provided but there are three shop types. Instead there are a lot of different shop names.

5.1 Learnings

The project taught me how to use the knowledge and skills that have been acquired during the course. I learned and I am still learning to navigate a little better through json data and how to deal with it. I learned how to apply aggregation functions, most offen it does not work but fortunatly, there are many ways on how to get to the next step. Refering to the *Data Transformation* Chapter where I was supposed to *$unwind* the *fields* sub-document but instead I did it with pandas and uploaded it to MongoDB.

In regards to Swisscom, I learned that their own shops are the type *Distributor*. Which makes sense, when they "distribute" their services to their potential clients but it is still a bit confusing.

I learned more about LaTex. Unfortunately, a solution was not found on how to save the geographical map without making a screenshot. Hence, the map was added as a picture. I am thankful to my friend from my Bachelor's studies for providing me guidance.

The course and the materials provided by the lecturer were of great help for the project.