Personal Project Bernardo Freire

April 7, 2022

HSLU Hochschule Luzern

Table of Contents

[1]:
```
!pip freeze > requirements.txt # get's the libraries of this environment
!'brew services run mongodb-community' # turns-on mongodb
```

[2]:
```
# Requirements & Configuration

##############################################
# Built-in imports
import os, sys, time, requests, json, pprint, re
##############################################
# Data imports
import numpy as np
import pandas as pd
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
##############################################
# Mongodb import
import pymongo
from pymongo import MongoClient
from bson import Regex
##############################################
# Plot imports
import matplotlib.pyplot as plt
import seaborn as sns
from pprint import pprint
from IPython.display import HTML
##############################################
# Setup preferences
# pd.set_option('max_rows', None)
# pd.set_option('max_colwidth', None)
plt.rcParams['figure.figsize'] = (16, 9)
sns.set_style('whitegrid')
sns.set_theme()# Table of Contents
```

1 Overview

In the course of this semester the war in Ukraine has changed my perception on fossil-fuels and how we as society are dependent from it. For the transition of all economies from a fossil-fuel-based economy to an renewable-energy based economy, it is my opinion, mobility is one of the most controvesial discussed and important topics. From the technical point of view, electric engines are efficient, and using electricity means that renewable energies can also be used. Electric mobility is for that reason, a key technology for more sustainable mobility and is one way of achieving ambitious energy and climate policy goals. Everything sounds amazing and fun, but, the central question when buying an electric vehicle (at least for me) are:

- Who are the best operators?
- Which city/region is best for electric vehicle users
- Are there any restrictions in terms of pluging devices depending on which city/canton?
- Do I still need special accessories, depending on the charging stations?

This question are answered using two APIs, a MongoDB database and graphs in order to vizualize the data. The used API is based on the page `recharge-my-car.ch` and is the face of the **National Data Infrastructure For Electromobility (DIEMO)** and shows where charging points for electric vehicles are and provides specific detailed information, everything available in real time.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Federal Office of Energy SFOE

At this point it should be mentioned, that not all operators are registered in this database.

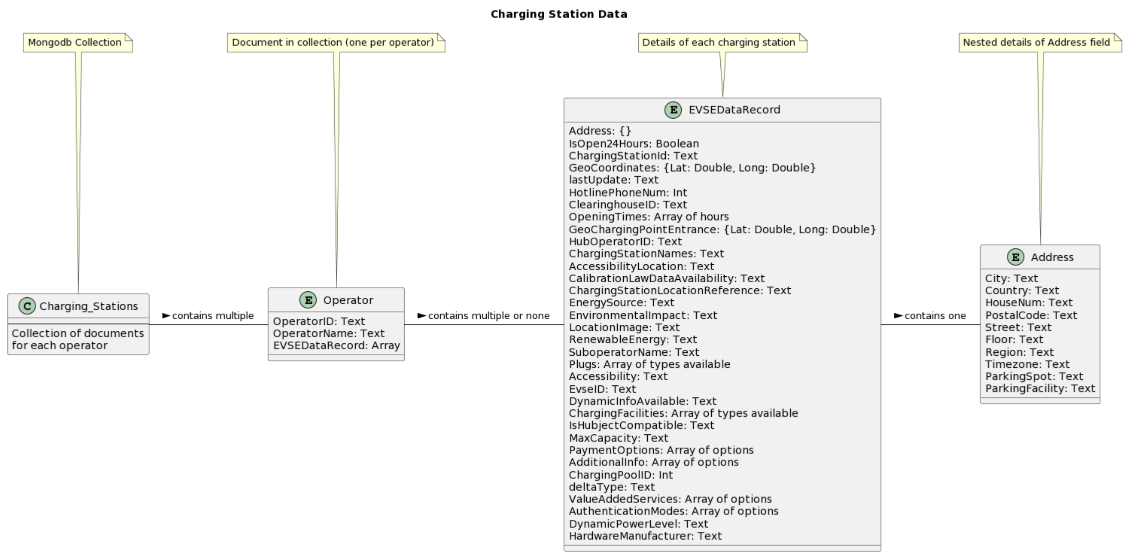
The correct assignment of postal codes and cities has proven to be extremely difficult, since a city can have several postal codes and several municipalities can share one postal code. At the same time, the naming of the city is not consistance as well. To reduce the assignment of cities to a single number, a second API was needed, directly from the Swiss Post. Using this second database, the entries of the DIEMO database were cleaned and enriched.

Last but not least, the MongoDB database is hosted on MongoDB Atlas and available via URL.

2 Document Structure: DIEMO API

The documents returned by the SFOE API were converted to JSON format and straight away imported into their respective collection in the SFOE database (`ChargingStations` collection). The class diagram below represents what a *single document* looks like. However, all collections have the same structure.

The size of all arrays **EVSEDataRecord** is the number of charging stations accross Switzerland. The array **EVSEDataRecord** contains further sub-arrays with the relevant information of each charging station. The '{}' indicates a nested substructure where the additional data is found linked below the main document. Other fields have not been presented in a separate entity in the diagramm.


Charging Station Data
Mongodb Collection
Document in collection (one per operator)
Details of each charging station
Nested details of Address field
C Charging_Stations
Collection of documents for each operator
► contains multiple
E Operator
OperatorID: Text
OperatorName: Text
EVSEDataRecord: Array
► contains multiple or none
E EVSEDataRecord
Address: {}
IsOpen24Hours: Boolean
ChargingStationId: Text
GeoCoordinates: {Lat: Double, Long: Double}
lastUpdate: Text
HotlinePhoneNum: Int
ClearinghouseID: Text
OpeningTimes: Array of hours
GeoChargingPointEntrance: {Lat: Double, Long: Double}
HubOperatorID: Text
ChargingStationNames: Text
AccessibilityLocation: Text
CalibrationLawDataAvailability: Text
ChargingStationLocationReference: Text
EnergySource: Text
EnvironmentalImpact: Text
LocationImage: Text
RenewableEnergy: Text
SuboperatorName: Text
Plugs: Array of types available
Accessibility: Text
EvseID: Text
DynamicInfoAvailable: Text
ChargingFacilities: Array of types available
IsHubjectCompatible: Text
MaxCapacity: Text
PaymentOptions: Array of options
AdditionalInfo: Array of options
ChargingPoolID: Int
deltaType: Text
ValueAddedServices: Array of options
AuthenticationModes: Array of options
DynamicPowerLevel: Text
HardwareManufacturer: Text
► contains one
E Address
City: Text
Country: Text
HouseNum: Text
PostalCode: Text
Street: Text
Floor: Text
Region: Text
Timezone: Text
ParkingSpot: Text
ParkingFacility: Text

- The collection **ChargingStations** contains the fields **OperatorID**, **OperatorName** and **EVSEDataRecord**.

- The **EVSEDataRecord** itself contains for each charging station a separate array containing the **Address**.
  - Within **Address** there are several sub-fields, the most important one is the **Address** array, which contains the details, such as City, Street, StreetNumber, PostalCode etc., of each charging station.

3 Extract & Load

In the ETL phase, data is extracted from the SFOE database using their API. Then, the data is transformed into a JSON format and is then loaded into our MongoDB database.

[3]:
```
###### Fetch data ######
# URL for data
url_data = 'https://data.geo.admin.ch/ch.bfe.ladestellen-elektromobilitaet/data/
↪oicp/ch.bfe.ladestellen-elektromobilitaet.json'
# "Get" data from URL
response = requests.get(url_data)
# Convert json to python dictionary
data = response.json()
# Store keys of API response for text output
keys = list(data.keys())

# URL of the mongodb cluster on the web
cnx = 'mongodb+srv://dbUser:hallo@cluster0.kiuex.mongodb.net/test'
# Client connects to local host
client = MongoClient(cnx)
# Connect to (new) db
db = client['PersonalProject']
# Regional data collection
collection_name = 'ChargingStations'

###### Create a document for each operator ######
for operator in data['EVSEData']:
    operator_name = operator['OperatorName']
    if not list(db[collection_name].find({"OperatorName": operator_name})):
        db[collection_name].insert_one(operator)
        print(f'{operator_name} inserted')
    else:
        print(f'INFO: {operator_name} already exists')

docs_in_collection = db[collection_name].count_documents({})
```

```
INFO: EWD Elektrizitätswerk Davos AG already exists
INFO: Tesla already exists
INFO: Martin Hertach already exists
INFO: Chocolat Frey AG already exists
INFO: PLUG’N ROLL already exists
INFO: Elektrizitätswerk Herrliberg already exists
INFO: Backpackers Villa Sonnenhof already exists
INFO: ebs Energie AG already exists
INFO: Stadt Dietikon already exists
INFO: Lidl Schweiz already exists
INFO: Elektrizitätswerk Obwalden already exists
INFO: AVIA already exists
```

```
INFO: IONITY already exists
INFO: mobilecharge already exists
INFO: Swisscharge already exists
INFO: Bgdi_Admin already exists
INFO: Test Operator already exists
INFO: en mobilecharge  already exists
INFO: EWAcharge already exists
INFO: Mobimo emobility already exists
INFO: S-Charge already exists
INFO: IBC already exists
INFO: PARK & CHARGE already exists
INFO: Migrol Fast Charging already exists
INFO: PAC e-moblity already exists
INFO: Move already exists
INFO: AIL already exists
INFO: EVA E-Mobilität already exists
INFO: eCarUp already exists
INFO: evpass already exists
INFO: Fastned already exists
INFO: Eniwa already exists
INFO: PLUG'N ROLL already exists
```

3.1 Connect to MongoDB and create database

Now, since the data is fetched from the API, we need to connect to the MongoDB database and load the data. Only the data **EVSEData** is of interest will be loaded and which operator will be save as own document within the collection **ChargingStations**.

33 operators have so far registered in DIEMO.

Each of the operators have a different number of charging points, adresses etc. Each operator is stored in a separate document within the collection called `ChargingStations`.

[4]:
```
# Pipeline 0: Check missing values of operator's name
pipeline = [
    # Match only operators with invalid name or PLZ
    {'$match': {'$or': [
            {'OperatorName': {'$exists': False}},
            {'EVSEDataRecord.Address.PostalCode': {'$exists': False}}
            ]
        }
    },
    # Project only the name of the operator and remove the _id
    {'$project': {
        '_id': 0,
        'OperatorName': 1
        }
    },
    # Sort by name - Alphabetical order
    {'$sort': {
        'OperatorName': 1
        }
    },
]

# Save to dataframe
missing_operators =  pd.DataFrame(db.ChargingStations.aggregate(pipeline))

# Pipeline 00: Check missing values post api
pipeline = [
    {'$match': {
        '$and': [
            {'postleitzahl': {'$eq': None}},
            {'gplz': {'$eq': None}}
            ]
        }
    }
]
# Save to dataframe
missing_PLZ = pd.DataFrame(db.PLZ.aggregate(pipeline))
```

The API returned the following operators with empty values:

Bgdi_Admin , IBC , Martin Hertach , S-Charge , Test Operator , evpass

4 Transform & Analysis

4.1 Best operators by number of stations

The best operators, in terms of number of charging stations, are as follows:

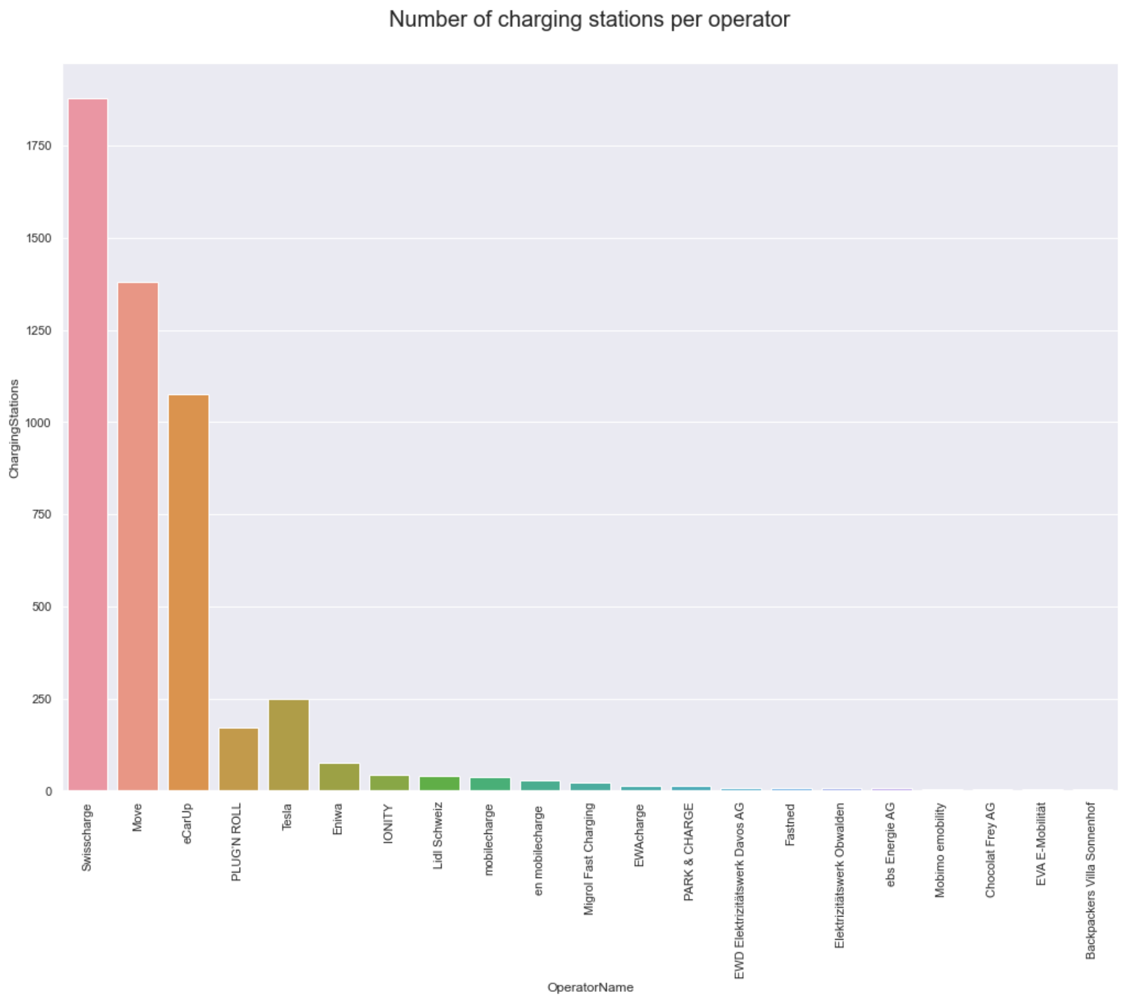
[5]:
```
# Pipeline 1: Get all operators

# Create a pipeline to aggregate the following: Per Operator, under␣
↪OperatorName, the number of charging stations given by the length of␣
↪EVSEDataRecord-array.
pipeline = [
    # Match only operators with more than 0 charging stations
    {"$match": {"EVSEDataRecord.1": {"$exists": True}} },
    # Project only the name of the operator, remove the _id and create a new␣
↪field with the length of the EVSEDataRecord-array.
    {"$project":{"_id":0, "OperatorName":1, "ChargingStations": {"$size":␣
↪"$EVSEDataRecord"}}},
    # Sort by the number of charging stations - Descending order
    {"$sort":{"ChargingStations":-1}},
]

# Save query as cursor
cursor_operators = db.ChargingStations.aggregate(pipeline)

# Store data in pandas dataframe
df = pd.DataFrame(cursor_operators)

# Create a overview plot of the data
sns.catplot(
    x = 'OperatorName',
    y = 'ChargingStations',
    data = df,
    kind = 'bar',
    height = 10,
    aspect = 1.5,
    ci = None,
).set_xticklabels(rotation = 90).fig.suptitle('Number of charging stations per␣
↪operator', fontsize = 20, y = 1.05)
plt.show()
```



The companies **Swisscharge** (1878), **Move** (1379), **eCarUp** (1077), **Plug'n'roll** (331) and **Tesla** (250) have the highest number of charging stations.

The companies **ebs Energie AG** (6), **Mobimo emobility** (5), **Chocolat Frey AG** (4), **EVA E-Mobilität** (4) and **Backpackers Villa Sonnenhof** (3) have the lowest number of charging stations.

4.2 Best cities by number of stations

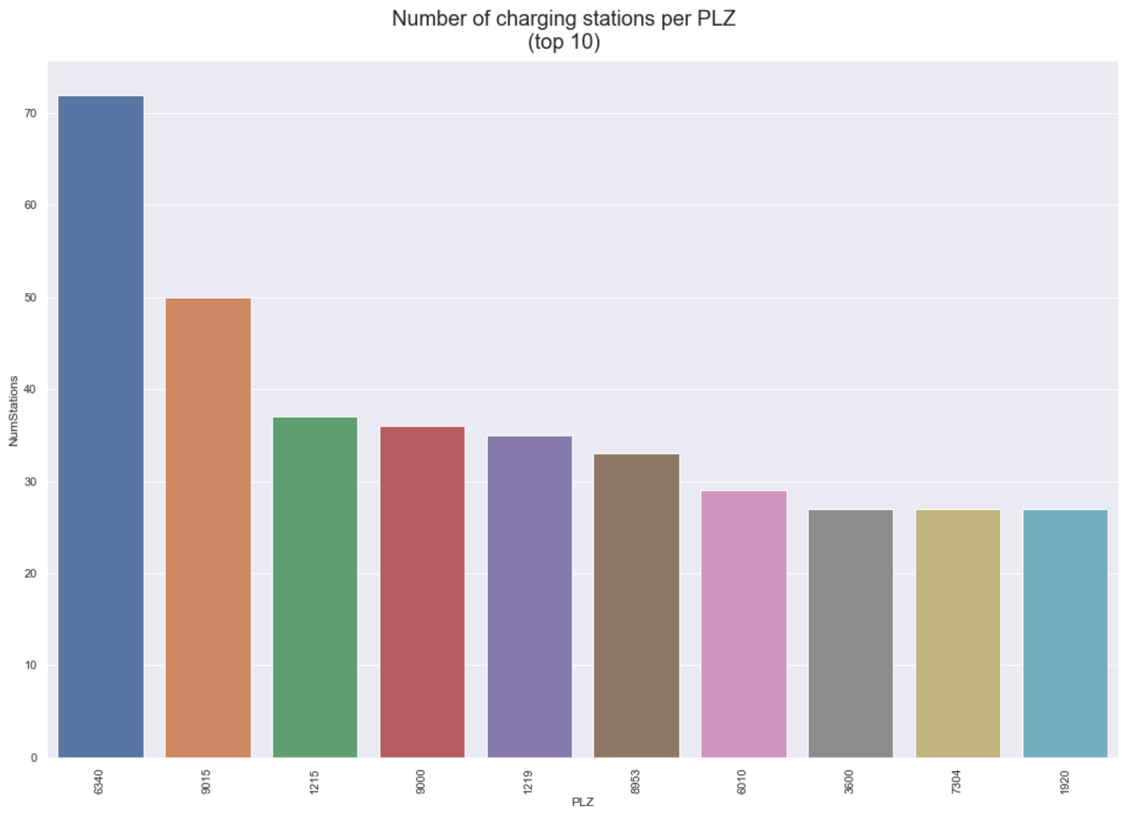
[6]:
```python
# Pipeline 2: Operators by PLZ
regx = re.compile(r"^[0-9]*$")

pipeline = [
    # Unwind the EVSEDataRecord-array
    {"$unwind":"$EVSEDataRecord"},
    # Match only existing PLZ (Numbers only)
    {"$match":{"EVSEDataRecord.Address.PostalCode": regx}},
    # Project only the cities
    {"$project":{
        "_id":0,
        "EVSEDataRecord.Address.PostalCode":1,
        "EVSEDataRecord.Address.City":1
        }},
    # Group by the PostalCode and sum the number of charging stations
    {"$group":{
        "_id": "$EVSEDataRecord.Address.PostalCode",
        "Cities": {"$first":"$EVSEDataRecord.Address.City"},
        "NumStations": {"$sum": 1}
        }},
    # Rename using project
    {"$project":{
        "_id":0,
        "PLZ": "$_id",
        "City":"$Cities",
        "NumStations":"$NumStations",
        }},
    # Sort by the number of charging stations, descending
    {"$sort": {"NumStations": -1}},
    # Convert types of PLZ
    {"$addFields":{"PLZ": {"$toInt":"$PLZ"}}},
]

# Save query as pandas dataframe
PLZ = pd.DataFrame(db.ChargingStations.aggregate(pipeline))

# Create a overview plot of the data
sns.catplot(
    x = 'PLZ',
    y = 'NumStations',
    data = PLZ.assign(PLZ= lambda x: x.PLZ.astype(str)).head(10),
    ci = None,
    kind = 'bar', height = 10, aspect = 1.5,
    ).set_xticklabels(rotation = 90).fig.suptitle('Number of charging stations␣
 ↪per PLZ\n(top 10)', fontsize = 20, y = 1.05)
```

```
plt.show()
```



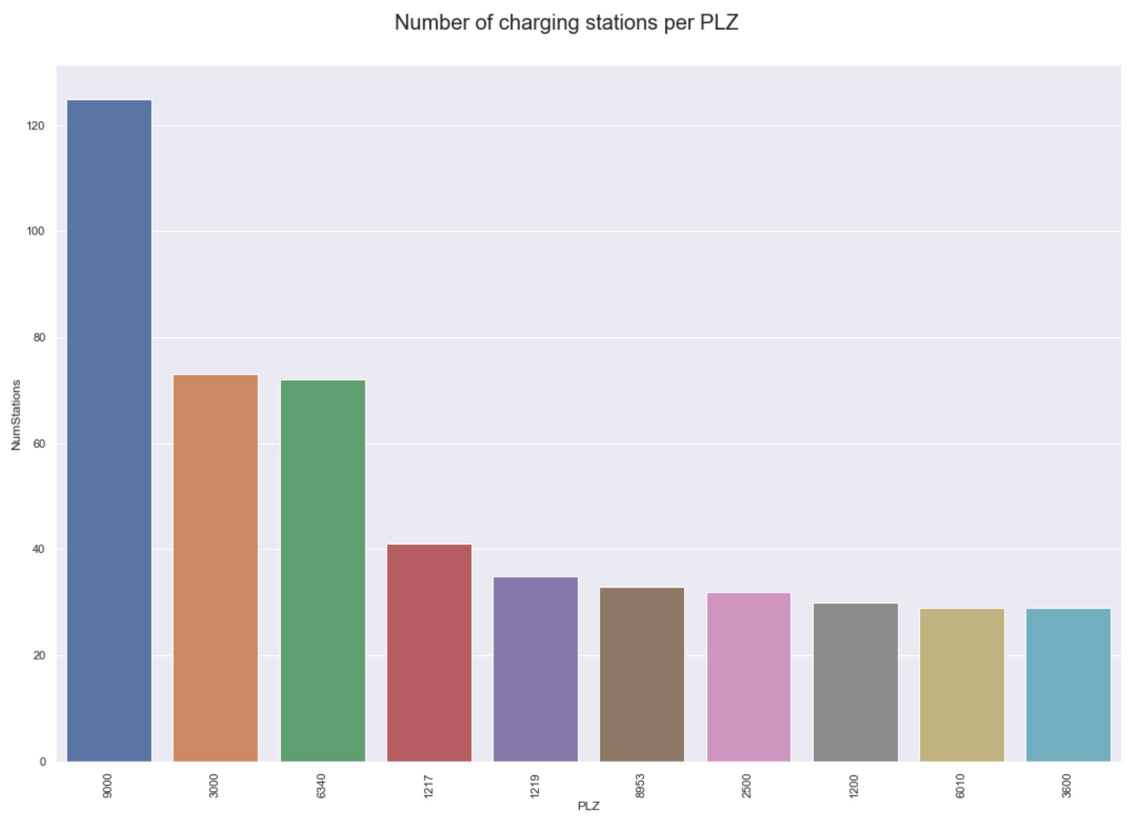
The most attractive cities by postal code (Postleitzahl - PLZ) are shown above. As mentioned in the introduction, the problem with the postal code is that this number does not necessary correspond to the city. Some cities in Switzerland have more than one postal code. For example, the city **Zurich** has two postal codes: **8050**, **8056** etc.

For that reason a second data set is used, which contains the postal code, city and corresponding cantons - this dataset is provided by `Swiss Post`. First, the same analysis is repeated, but this time, the postal code is merged and only the `GPLZ` (Gemeindepostleitzahl - City Postal Code) is than used. Further, due to the manageable distances within the cantons, the summary is provided at cantonal level as well.

[7]:
```
# Pipeline 3: Operators by City using second API and lookup
pipeline = [
    {"$match":{
         "$and":[
            {"EVSEDataRecord.0":{"$exists":True}},
            {"EVSEDataRecord.Address.PostalCode":{"$exists":True}}
            ]
        }
    },
    {"$project":{
        "_id":0,
        "PLZ":"$EVSEDataRecord.Address.PostalCode",
        }
    },
    {"$unwind":"$PLZ"},
    {"$lookup":{
        "from":'PLZ',
        "localField":'PLZ',
        "foreignField":'postleitzahl',
        "as":'POSTAPI'
        }
    },
    {"$project":{
        "_id":0,
        "PLZ":1,
        "GPLZ":{"$toInt":{"$first":"$POSTAPI.gplz"}},
        }
    },
    {"$match":{
        "GPLZ":{"$ne":None}
        }
    },
    {"$group":{
        "_id": "$GPLZ",
        "Count":{"$sum":1},
        }
    },
    {"$sort":{
        "Count":-1
        }
    },
    {"$project":{
        "_id":0,
        "PLZ": "$_id",
        "NumStations": "$Count",
        }
    },
]

# Save query as pandas dataframe
PLZ = pd.DataFrame(db.ChargingStations.aggregate(pipeline))

# Create a overview plot of the data
sns.catplot(
    x = 'PLZ',
    y = 'NumStations',
    data = PLZ.assign(PLZ= lambda x: x.PLZ.astype(str)).head(10),
    ci = None,
    kind = 'bar', height = 10, aspect = 1.5,
    ).set_xticklabels(rotation = 90).fig.suptitle('Number of charging stations␣
 ↪per PLZ', fontsize = 20, y = 1.05)
plt.show()
```



One can see now, that the city of Baar (6340) is “only” in the third position and St. Gallen is on the pole-position followed by the capital Bern.

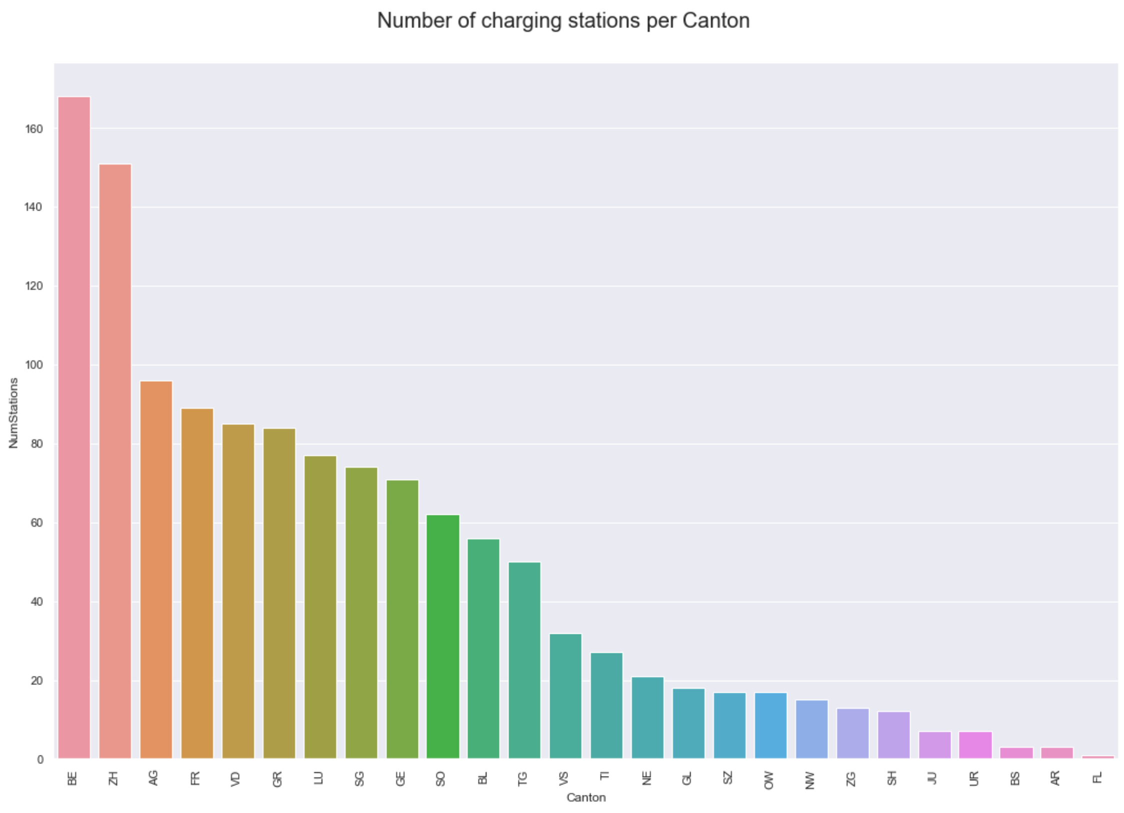
4.3 Best cantons by number of stations

As previously mentioned, the distribution of charging stations is also of interest. Therefore, the analysis is repeated and this time presented on cantonal level as follows.

```
[8]: # Pipeline 4: Operators by Canton
pipeline = [
    {"$lookup":{
        'from':'ChargingStations',
        'localField':'postleitzahl',
        'foreignField':'EVSEDataRecord.Address.PostalCode',
        'as':'ChargingStations'
        }
    },
    {"$match":{"ChargingStations.0":{'$exists':True}}},
    {"$project":{
        "_id":0,
        "Canton":"$kanton",
        "ChargingStations": {"$size": "$ChargingStations.EVSEDataRecord"}
        }
    },
    {"$group":{
        "_id": "$Canton",
        "NumStations": {"$sum": 1}
        }
    },
    {"$sort": {"NumStations": -1}},
    {"$project":{"Canton":"$_id", "NumStations":1, "_id":0}},
]

gplz = pd.DataFrame(db.PLZ.aggregate(pipeline))

# Create a overview plot of the data
sns.catplot(
    x='Canton',
    y='NumStations',
    data=gplz,
    kind='bar',
    height=10,
    aspect=1.5,
).set_xticklabels(rotation = 90).fig.suptitle('Number of charging stations per␣
 ↪Canton', fontsize = 20, y = 1.05)
plt.show()
```



The best canton in Switzerland is Bern, followed by Zurich and Aargau, just to mention a few. Basel-Stadt, Appenzell-Ausserrhoden and Jura come in last among the cantons.

4.4 Best cities by number of plug types

Imagine you are driving around Switzerland and suddenly you need to charge your car. Where do you go if you have to assume that you will not be able to plug in your car.

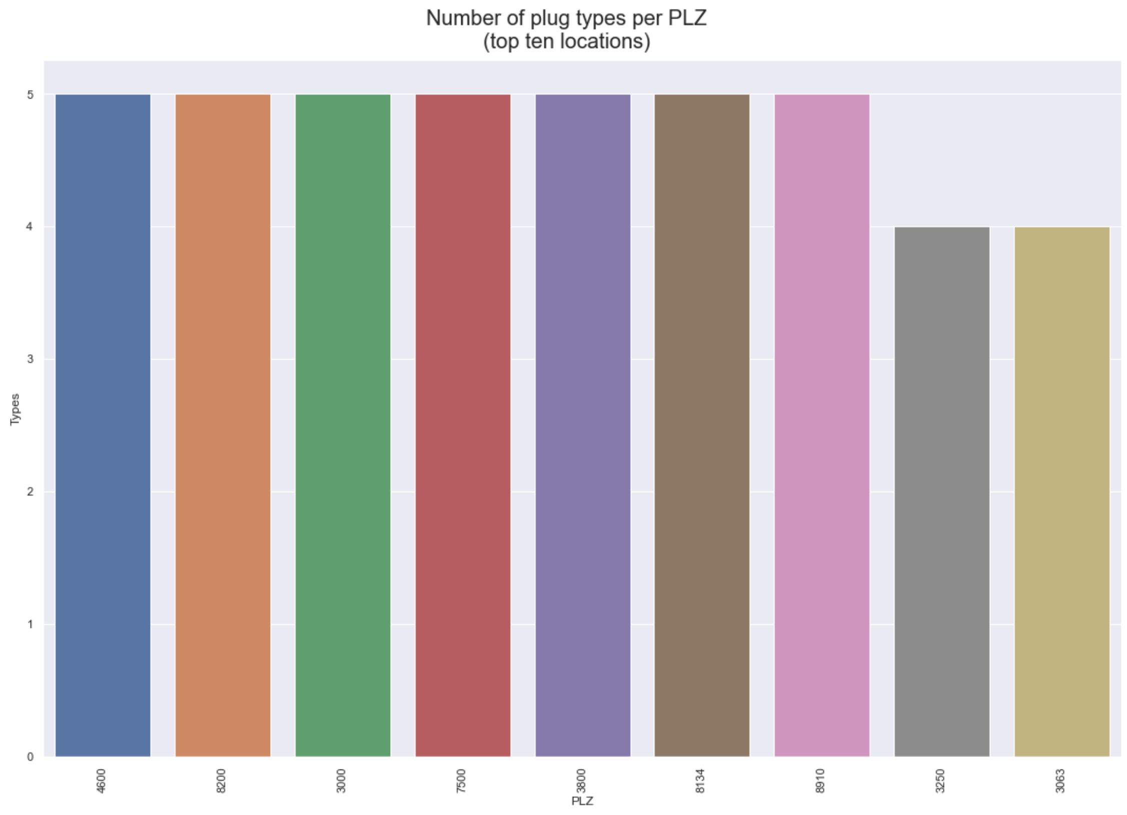
You drive to where the most plug-in types are available, right?

```
[9]: pipeline = [
    {
        '$match': {
            'EVSEDataRecord.Address.PostalCode': {
                '$ne': None
            }
        }
    }, {
        '$unwind': {
            'path': '$EVSEDataRecord'
        }
    }, {
        '$project': {
            '_id': 0,
            'OperatorName': 1,
            'PLZ': '$EVSEDataRecord.Address.PostalCode',
            'Plugs': '$EVSEDataRecord.Plugs',
            'ChargingFacilities': '$EVSEDataRecord.ChargingFacilities'
        }
    }, {
        '$unwind': {
            'path': '$Plugs'
        }
    }, {
        '$group': {
            '_id': '$Plugs',
            'PLZs': {
                '$push': '$PLZ'
            }
        }
    }, {
        '$unwind': {
            'path': '$PLZs'
        }
    }, {
        '$lookup': {
            'from': 'PLZ',
            'localField': 'PLZs',
            'foreignField': 'postleitzahl',
            'as': 'POSTAPI'
        }
    }, {
        '$project': {
            '_id': 0,
            'PlugType': '$_id',
            'Canton': {
                '$first': '$POSTAPI.kanton'
            },
            'PLZ': {
                '$toString': {
                    '$first': '$POSTAPI.gplz'
                }
            }
        }
    }, {
        '$group': {
            '_id': '$PLZ',
            'Plugs': {
                '$addToSet': '$PlugType'
            }
        }
    }, {
        '$project': {
            '_id': 0,
            'PLZ': '$_id',
            'Types': {
                '$size': '$Plugs'
            }
        }
    }, {
        '$sort': {
            'Types': -1
        }
    }
]

plugs = pd.DataFrame(db.ChargingStations.aggregate(pipeline))

# Create a overview plot of the data
sns.catplot(
    x='PLZ',
    y='Types',
    data=plugs.head(10),
    kind='bar',
    height=10,
    aspect=1.5,
).set_xticklabels(rotation = 90).fig.suptitle('Number of plug types per␣
 ↪PLZ\n(top ten locations)', fontsize = 20, y = 1.05)
```

```
plt.show()
```



The best locations are: Adliswil, Interlaken, Bern, Olten St. Moritz, Schaffhausen and Affoltern am Albis: All those cities would have five different kind of plugs available for your car.

4.5 Best cities by Power, Amperage & Voltage

Last but not least, we are now going to analyze the power (and type), amperage and voltage of each location (again, with the city-wide postal code). The power type describes the connection type in terms of AC and DC power supply (e.g. number of phases). The voltage describes the voltage on the supply station, the amperage the maximal amount of amperes and last but not least the power (basically the product of voltage and amperage).

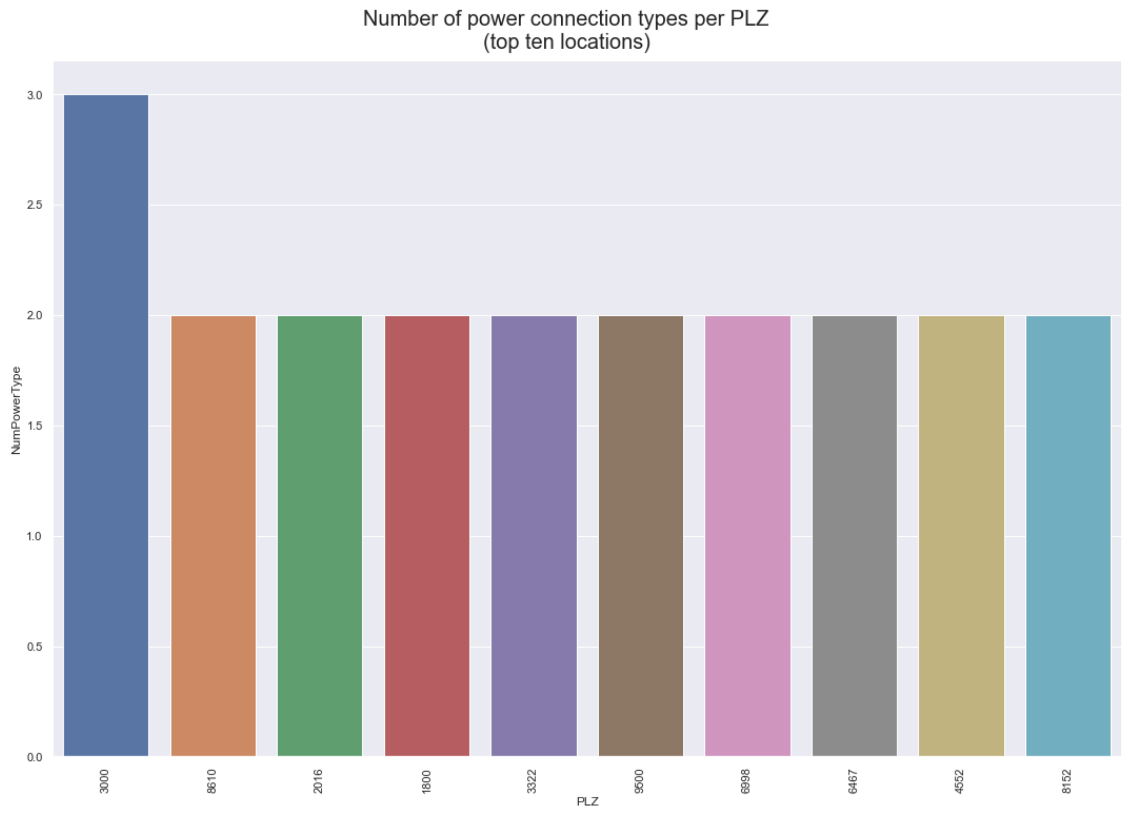
Why is this important? The charging time of electrical cars is exactly from those parameters dependent.

Let now start with the power connection type per city:

[10]:
```
pipeline = [
    {
        '$match': {
            'EVSEDataRecord.Address.PostalCode': {
                '$ne': None
            }
        }
    }, {
        '$unwind': {
            'path': '$EVSEDataRecord'
        }
    }, {
        '$project': {
            '_id': 0,
            'OperatorName': 1,
            'PLZ': '$EVSEDataRecord.Address.PostalCode',
            'Plugs': '$EVSEDataRecord.Plugs',
            'ChargingFacilities': '$EVSEDataRecord.ChargingFacilities'
        }
    }, {
        '$unwind': {
            'path': '$ChargingFacilities'
        }
    }, {
        '$group': {
            '_id': '$ChargingFacilities',
            'PLZs': {
                '$push': '$PLZ'
            }
        }
    }, {
        '$unwind': {
            'path': '$PLZs'
        }
    }, {
        '$lookup': {
            'from': 'PLZ',
            'localField': 'PLZs',
            'foreignField': 'postleitzahl',
            'as': 'POSTAPI'
        }
    }, {
        '$project': {
            '_id': 0,
            'Type': '$_id',
            'Canton': {
                '$first': '$POSTAPI.kanton'
            },
            'PLZ': {
                '$toString': {
                    '$first': '$POSTAPI.gplz'
                }
            }
        }
    }, {
        '$group': {
            '_id': '$PLZ',
            'Types_': {
                '$addToSet': '$Type'
            }
        }
    }, {
        '$project': {
            '_id': 0,
            'PLZ': '$_id',
            'Types_': 1
        }
    }, {
        '$unwind': {
            'path': '$Types_'
        }
    }, {
        '$group': {
            '_id': '$PLZ',
            'PowerType': {
                '$addToSet': '$Types_.powertype'
            },
            'Voltage': {
                '$addToSet': '$Types_.voltage'
            },
            'Amperage': {
                '$addToSet': '$Types_.amperage'
            },
            'Power': {
                '$addToSet': '$Types_.power'
            }
        }
    }, {
        '$project': {
            "_id":0,
            "PLZ":"$_id",
            "PowerType":"$PowerType",
            "Voltage":"$Voltage",
            "Amperage":"$Amperage",
            "Power":"$Power",
            "NumPowerType":{"$size":"$PowerType"},
            "NumVoltage":{"$size":"$Voltage"},
            "NumAmperage":{"$size":"$Amperage"},
            "NumPower":{"$size":"$Power"},
        }
    }
]

power = pd.DataFrame(db.ChargingStations.aggregate(pipeline))
```

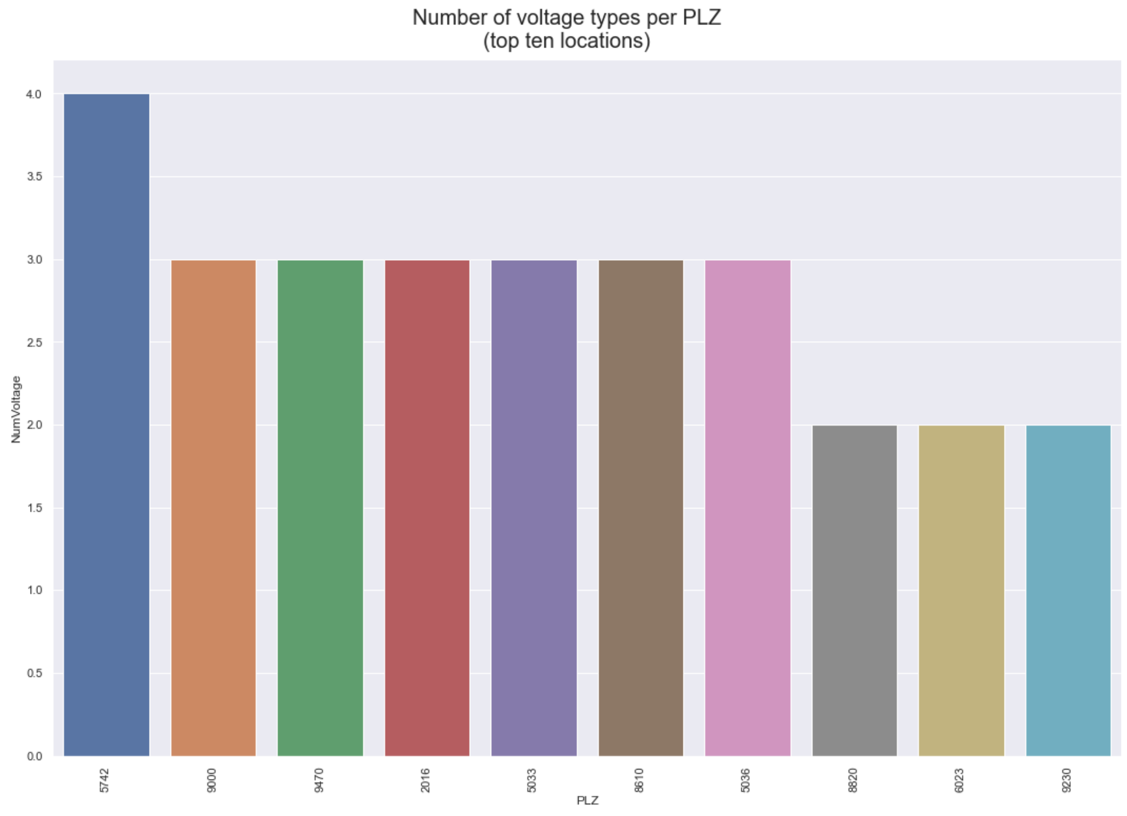
[11]:
```
# Create a overview plot of the data
sns.catplot(
    x='PLZ',
    y='NumPowerType',
    data=power.sort_values(by = ['NumPowerType'], ascending=False).head(10),
    kind='bar',
    height=10,
    aspect=1.5,
).set_xticklabels(rotation = 90).fig.suptitle('Number of power connection types␣
 ↪per PLZ\n(top ten locations)', fontsize = 20, y = 1.05)
plt.show()
```



Again, as one of the top-scorers is the city of Bern, with 3 different available types. The following cities have all only two different kind of power connections available. In particular, the city of Bern has the following power types available:
AC 1 PHASE , AC 3 PHASE , DC

Proceeding to the next question, what is the number of available voltage per PLZ? Let’s check:

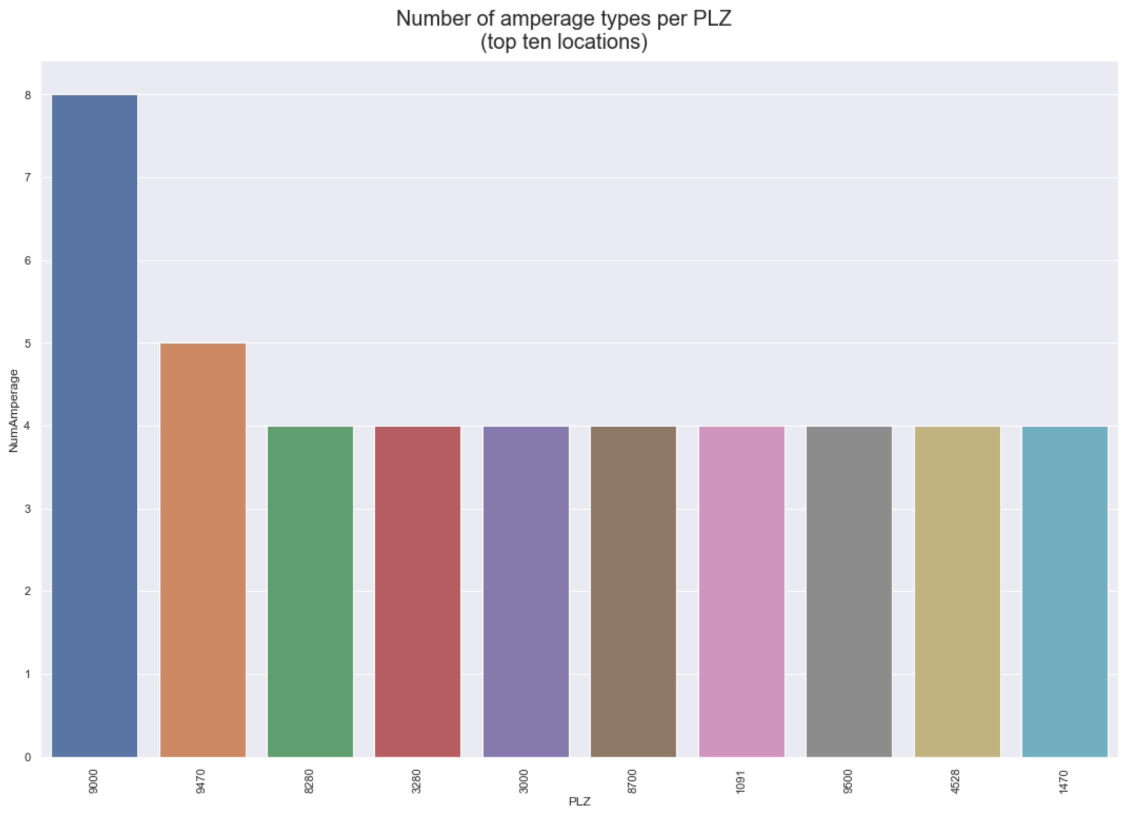
```
[12]: # Create a overview plot of the data
sns.catplot(
    x='PLZ',
    y='NumVoltage',
    data=power.sort_values(by = ['NumVoltage'], ascending=False).head(10),
    kind='bar',
    height=10,
    aspect=1.5,
).set_xticklabels(rotation = 90).fig.suptitle('Number of voltage types per␣
 ↪PLZ\n(top ten locations)', fontsize = 20, y = 1.05)
plt.show()
```



Again, as one of the top-scorers is the city of Bern, but the **best city is this time Kölliken**. The following cities have all only 3 or 2 different kind of connections available. In particular, the city of Kölliken has the following voltage types available:
230 , 400 , 500 , 920 [**Volts**]

Proceeding to the next question, what is the number of available amperage per PLZ? Let's check:

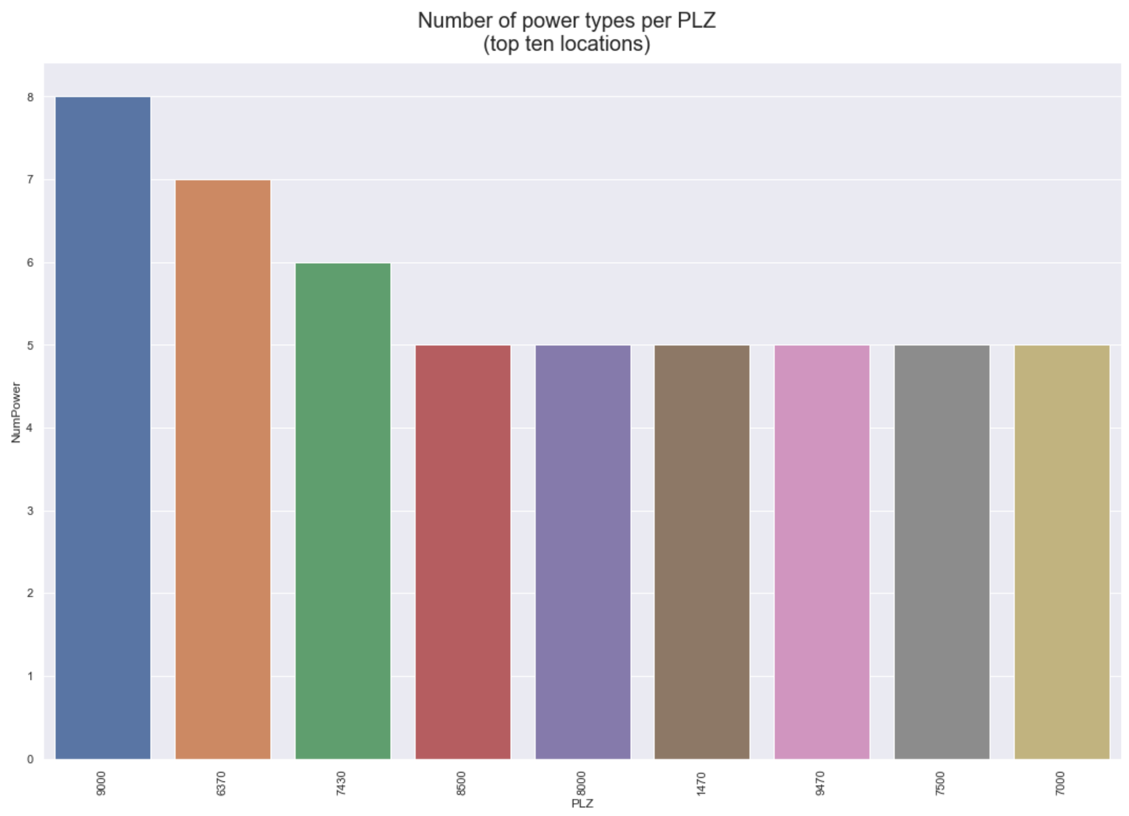
```
[13]: # Create a overview plot of the data
sns.catplot(
    x='PLZ',
    y='NumAmperage',
    data=power.sort_values(by = ['NumAmperage'], ascending=False).head(10),
    kind='bar',
    height=10,
    aspect=1.5,
).set_xticklabels(rotation = 90).fig.suptitle('Number of amperage types per␣
 ↪PLZ\n(top ten locations)', fontsize = 20, y = 1.05)
plt.show()
```



The **best city is this time St. Gallen**. The following cities have all only 5 or 4 different kind of connections available. In particular, the city of St. Gallen has the following amperage types available:
15.0, 16.0, 31.0, 32.0, 60.0, 63.0, 218.0 [Amperes]

Proceeding to the last question, what is the number of available power per PLZ? Let's check:

```
[14]: # Create a overview plot of the data
      sns.catplot(
          x='PLZ',
          y='NumPower',
          data=power.sort_values(by = ['NumPower'], ascending=False).head(10),
          kind='bar',
          height=10,
          aspect=1.5,
      ).set_xticklabels(rotation = 90).fig.suptitle('Number of power types per␣
       ↪PLZ\n(top ten locations)', fontsize = 20, y = 1.05)
      plt.show()
```



The **best city is this time St. Gallen**. The following cities have all only 5 or 4 different kind of connections available. In particular, the city of St. Gallen has the following types available: **11.0, 22.0, 22.0, 22.08, 24.0, 50.0, 50.0, 120.0** [**Kilo Watts**]

5 Conclusion

In regard of the two used APIs, I have learned new insights such as the most relevant operators in the field, the best cities and as cantons for electrical mobility. This report answers the following questions:

- Best operator by number of stations

```
[15]: df.head()
```

```
[15]:   OperatorName  ChargingStations
      0  Swisscharge              1878
      1         Move              1379
      2       eCarUp              1077
      3  PLUG'N ROLL               331
      4        Tesla               250
```

- Best cities by number of stations

```
[16]: gplz.head()
```

```
[16]:    NumStations Canton
      0          168     BE
      1          151     ZH
      2           96     AG
      3           89     FR
      4           85     VD
```

- Best cantons number of stations

```
[17]: gplz.head()
```

```
[17]:    NumStations Canton
      0          168     BE
      1          151     ZH
      2           96     AG
      3           89     FR
      4           85     VD
```

- Best cities by number of plug types

```
[18]: plugs.head()
```

```
[18]:     PLZ  Types
      0  4600      5
      1  8200      5
      2  3000      5
      3  7500      5
      4  3800      5
```

- Best cities by Power, Amperage & Voltage

```
[34]: power.head()
```

```
[34]:    PLZ          PowerType        Voltage     Amperage  \
      0  8610  [DC, AC_3_PHASE]  [0, 400, 230]  [60, 0, 32]
      1  1216  [AC_3_PHASE, DC]       [0, 230]      [0, 63]
      2  1213      [AC_3_PHASE]          [230]         [32]
      3  1796      [AC_3_PHASE]          [230]         [32]
      4  8953  [DC, AC_3_PHASE]       [0, 230]  [0, 32, 16]

                                 Power  NumPowerType  NumVoltage  NumAmperage  \
      0    [20.0, 22.0, 50.0, 22.08]             2           3            3
      1                 [50.0, 22.0]             2           2            2
      2                   [22, 22.0]             1           1            1
      3               [50, 22, 22.0]             1           1            1
      4  [150, 22, 11.0, 50.0, 22.0]             2           2            3

         NumPower
      0         4
      1         2
      2         2
      3         3
      4         5
```

6 Learnings

In this project, I have learned, in more depth, about MongoDB and its document based schema, querying syntax, importing data and aggregation pipelines. In the beginning I was very unsure how to handle MongoDB and especially the aggregation pipelines, but after this project I even started to enjoy creating pipelines. The application of lookup and unpacking of arrays were a bit difficult to start with, but after practicing it, it already started to feel kind of natural. Furthermore, I also learned, that the benefits of MongoDB are not limited to its schema, but also to its ability to scale horizontally, also known as “scale-out”, which is primarily achieved through sharding - something that is not possible with traditional relational database management systems (RDBMS).