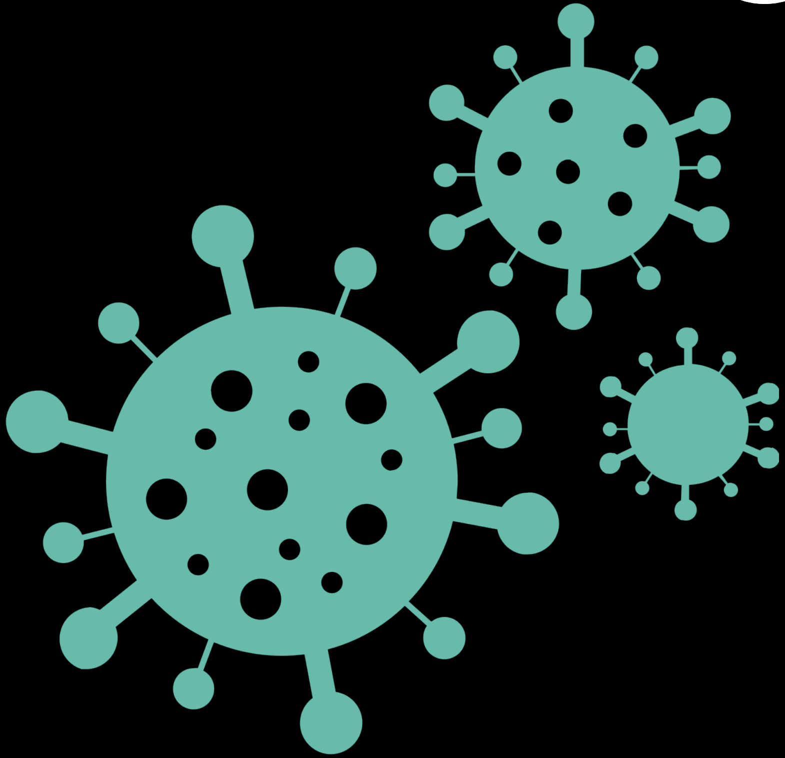


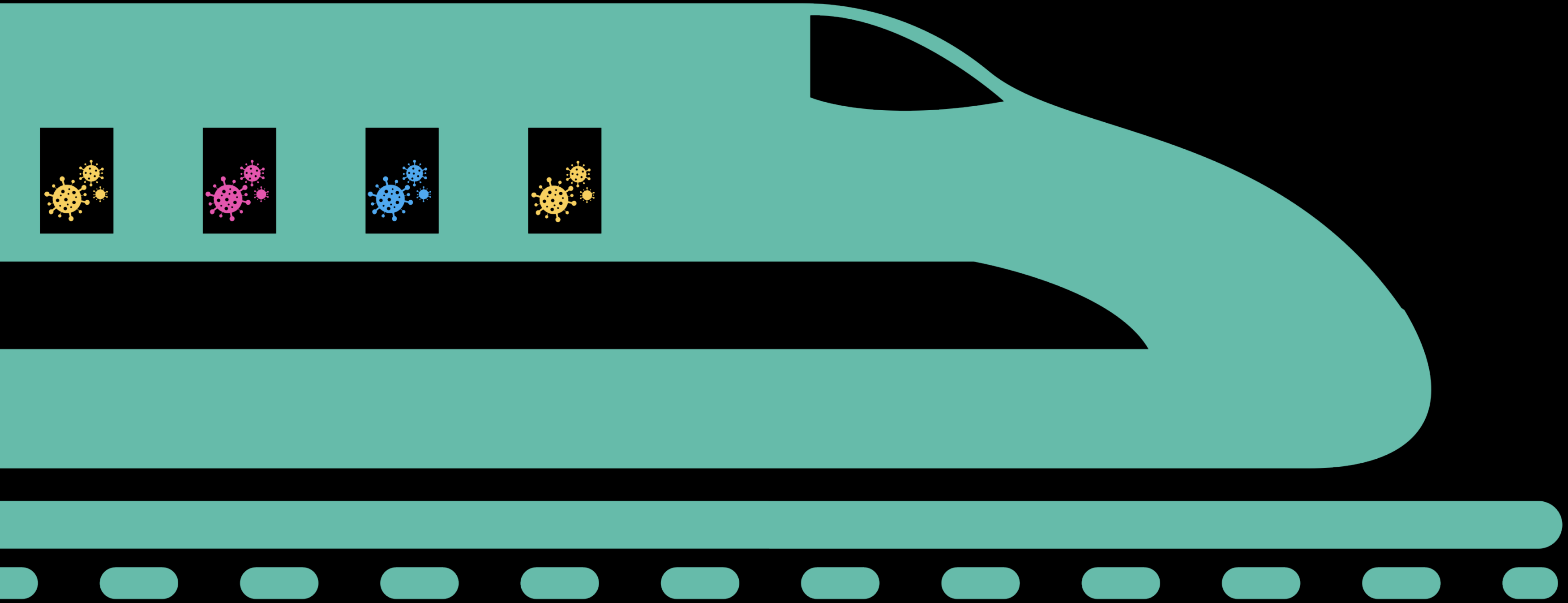
DS II SEMINAR

MARCEL OCHSENDORF

INTERMEDIATE PRESENTATION

RESEARCH QUESTION

IS IT POSSIBLE TO IDENTIFY COVID SPREADS DUE TO PUBLIC TRANSPORTATION SERVICE?*



* ASSUMING 100% RAILWAY TRAFFIC [NO CARS,...]

TIMELINE

PREPARE DATA [DB | RKI]

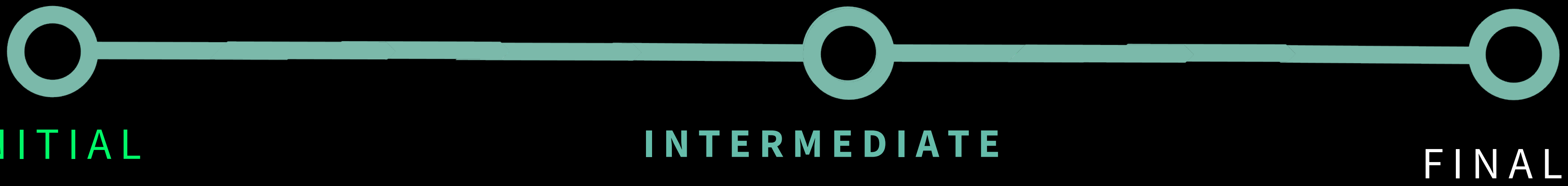
BUILD MAINSTATION AND DEPATURE DATASET

FETCH TRAIN DATA FOR SPECIFIED INTERVALS

COMBINE PROCESSED DATA

GENERATE HEATMAPS

BUILD AN INTERACTIVE MAP



TRAIN DATA ACQUISITION

HAVAS | ~~IRIS~~ | ~~DB API~~

[RAILWAY CONNECTIONS, CAPACITY, FILL RATE, DEPARTURE TABLE]

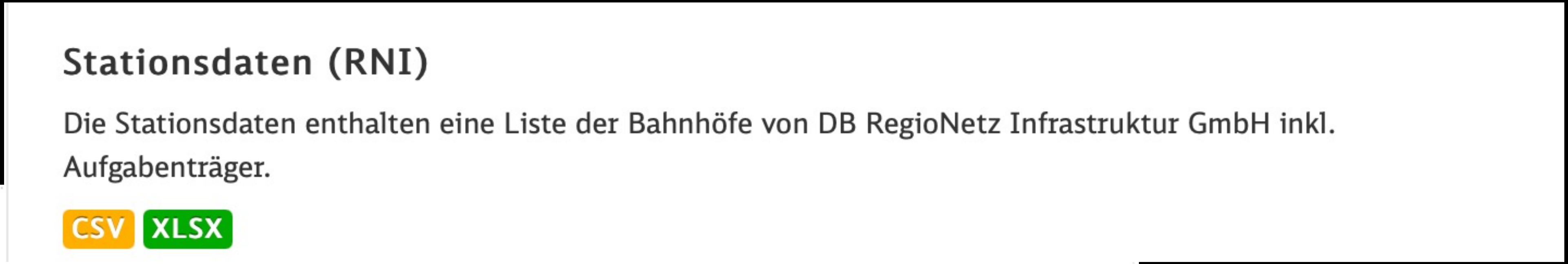
- DIFFICULT DATA EXTRACTION

- MANUAL PDF PARSING OF TRAIN TYPES
- GERMAN CSV/EXCEL FILES
- ENCODING ISSUES
- NOT MATCHING GEO COORDINATE SYSTEMS
- **NO DB HISTORY DATA**

STATION DATA

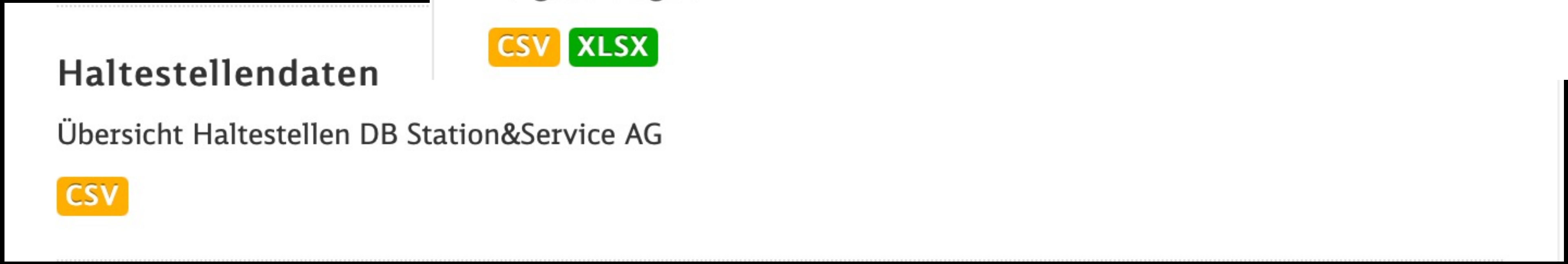
DB API

[**RAILWAY STATIONS**, CAPACITY]



Stationsdaten (RNI)

Die Stationsdaten enthalten eine Liste der Bahnhöfe von DB RegioNetz Infrastruktur GmbH inkl. Aufgabenträger.



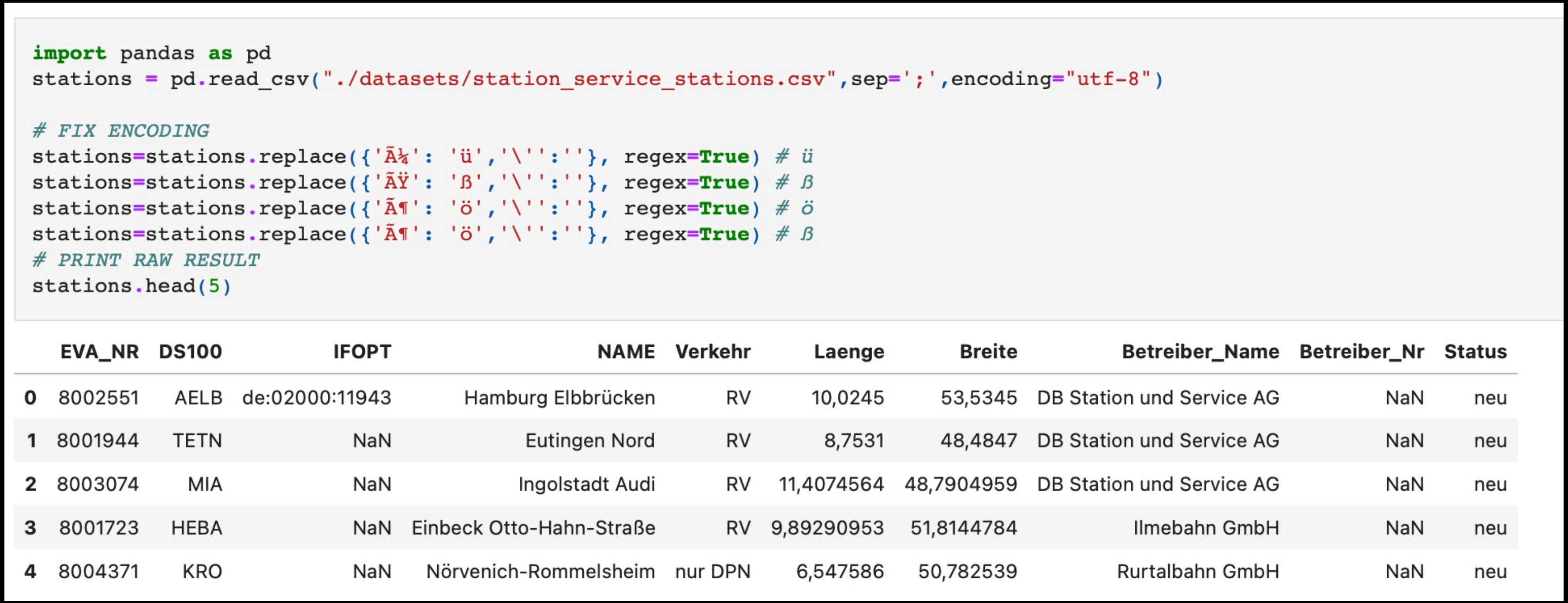
Haltestellendaten

Übersicht Haltestellen DB Station&Service AG

STATION DATA

DB API

[RAILWAY STATIONS]



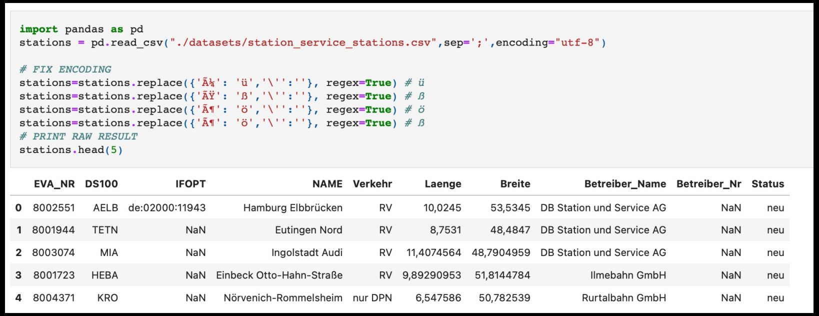
```python
import pandas as pd
stations = pd.read_csv("./datasets/station_service_stations.csv",sep=';',encoding="utf-8")

# FIX ENCODING
stations=stations.replace({'Ã¼': 'ü','\'':''}, regex=True) # ü
stations=stations.replace({'ÃŸ': 'ß','\'':''}, regex=True) # ß
stations=stations.replace({'Ã¶': 'ö','\'':''}, regex=True) # ö
stations=stations.replace({'Ã¶': 'ö','\'':''}, regex=True) # ß
# PRINT RAW RESULT
stations.head(5)
```

	EVA_NR	DS100	IFOPT	NAME	Verkehr	Laenge	Breite	Betreiber_Name	Betreiber_Nr	Status
0	8002551	AELB	de:02000:11943	Hamburg Elbbrücken	RV	10,0245	53,5345	DB Station und Service AG	NaN	neu
1	8001944	TETN	NaN	Eutingen Nord	RV	8,7531	48,4847	DB Station und Service AG	NaN	neu
2	8003074	MIA	NaN	Ingolstadt Audi	RV	11,4074564	48,7904959	DB Station und Service AG	NaN	neu
3	8001723	HEBA	NaN	Einbeck Otto-Hahn-Straße	RV	9,89290953	51,8144784	Ilmebahn GmbH	NaN	neu
4	8004371	KRO	NaN	Nörvenich-Rommelsheim	nur DPN	6,547586	50,782539	Rurtalbahn GmbH	NaN	neu

STATION DATA

DB API

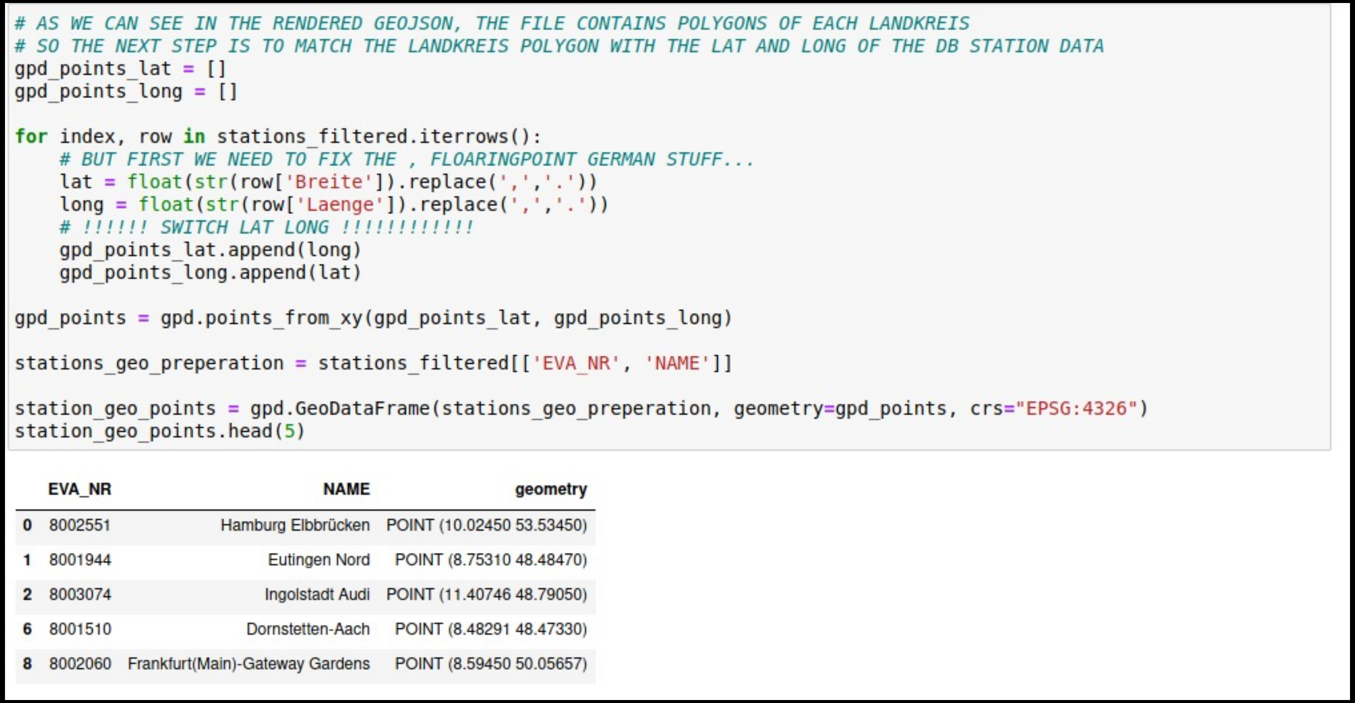
[RAILWAY STATIONS]



```python
import pandas as pd
stations = pd.read_csv("./datasets/station_service_stations.csv",sep=';',encoding="utf-8")

# FIX ENCODING
stations=stations.replace({'Ã¼': 'ü','\'':''}, regex=True) # ü
stations=stations.replace({'ÃŸ': 'ß','\'':''}, regex=True) # ß
stations=stations.replace({'Ã¶': 'ö','\'':''}, regex=True) # ö
stations=stations.replace({'Ã¶': 'ö','\'':''}, regex=True) # ß
# PRINT RAW RESULT
stations.head(5)
```

	EVA_NR	DS100	IFOPT	NAME	Verkehr	Laenge	Breite	Betreiber_Name	Betreiber_Nr	Status
0	8002551	AELB	de:02000:11943	Hamburg Elbbrücken	RV	10,0245	53,5345	DB Station und Service AG	NaN	neu
1	8001944	TETN	NaN	Eutingen Nord	RV	8,7531	48,4847	DB Station und Service AG	NaN	neu
2	8003074	MIA	NaN	Ingolstadt Audi	RV	11,4074564	48,7904959	DB Station und Service AG	NaN	neu
3	8001723	HEBA	NaN	Einbeck Otto-Hahn-Straße	RV	9,89290953	51,8144784	Ilmebahn GmbH	NaN	neu
4	8004371	KRO	NaN	Nörvenich-Rommelsheim	nur DPN	6,547586	50,782539	Rurtalbahn GmbH	NaN	neu

- PANDAS
- GEOPANDAS
- GEOPLOT



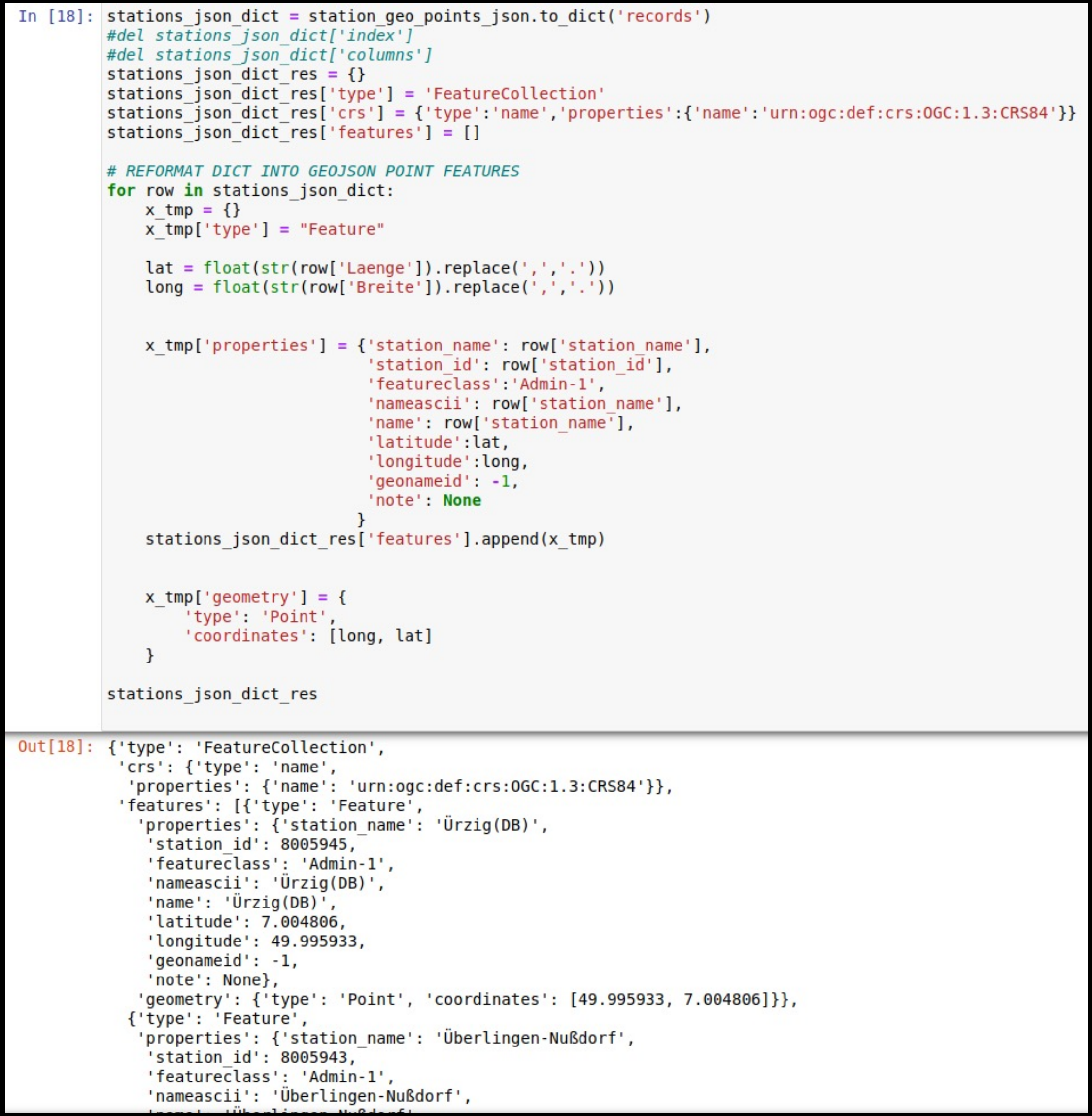
```python
# AS WE CAN SEE IN THE RENDERED GEOJSON, THE FILE CONTAINS POLYGONS OF EACH LANDKREIS
# SO THE NEXT STEP IS TO MATCH THE LANDKREIS POLYGON WITH THE LAT AND LONG OF THE DB STATION DATA
gpd_points_lat = []
gpd_points_long = []

for index, row in stations_filtered.iterrows():
    # BUT FIRST WE NEED TO FIX THE , FLOARINGPOINT GERMAN STUFF...
    lat = float(str(row['Breite']).replace(',','.'))
    long = float(str(row['Laenge']).replace(',','.'))
    # !!!!!! SWITCH LAT LONG !!!!!!!!!!!!
    gpd_points_lat.append(long)
    gpd_points_long.append(lat)

gpd_points = gpd.points_from_xy(gpd_points_lat, gpd_points_long)

stations_geo_preperation = stations_filtered[['EVA_NR', 'NAME']]

station_geo_points = gpd.GeoDataFrame(stations_geo_preperation, geometry=gpd_points, crs="EPSG:4326")
station_geo_points.head(5)
```

	EVA_NR	NAME	geometry
0	8002551	Hamburg Elbbrücken	POINT (10.02450 53.53450)
1	8001944	Eutingen Nord	POINT (8.75310 48.48470)
2	8003074	Ingolstadt Audi	POINT (11.40746 48.79050)
6	8001510	Dornstetten-Aach	POINT (8.48291 48.47330)
8	8002060	Frankfurt(Main)-Gateway Gardens	POINT (8.59450 50.05657)

STATION DATA

DB API

[RAILWAY STATIONS]



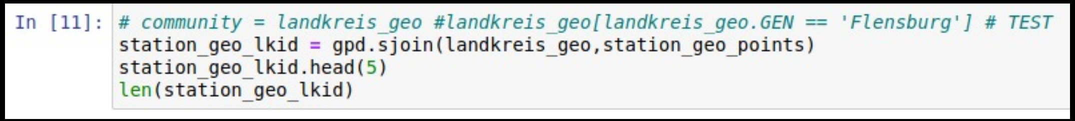
```
In [18]: stations_json_dict = station_geo_points_json.to_dict('records')
         #del stations_json_dict['index']
         #del stations_json_dict['columns']
         stations_json_dict_res = {}
         stations_json_dict_res['type'] = 'FeatureCollection'
         stations_json_dict_res['crs'] = {'type':'name','properties':{'name':'urn:ogc:def:crs:OGC:1.3:CRS84'}}
         stations_json_dict_res['features'] = []

         # REFORMAT DICT INTO GEOJSON POINT FEATURES
         for row in stations_json_dict:
             x_tmp = {}
             x_tmp['type'] = "Feature"

             lat = float(str(row['Laenge']).replace(',','.'))
             long = float(str(row['Breite']).replace(',','.'))


             x_tmp['properties'] = {'station_name': row['station_name'],
                                    'station_id': row['station_id'],
                                    'featureclass':'Admin-1',
                                    'nameascii': row['station_name'],
                                    'name': row['station_name'],
                                    'latitude':lat,
                                    'longitude':long,
                                    'geonameid': -1,
                                    'note': None
                                   }
             stations_json_dict_res['features'].append(x_tmp)


             x_tmp['geometry'] = {
                 'type': 'Point',
                 'coordinates': [long, lat]
             }

         stations_json_dict_res
```

```
Out[18]: {'type': 'FeatureCollection',
          'crs': {'type': 'name',
           'properties': {'name': 'urn:ogc:def:crs:OGC:1.3:CRS84'}},
          'features': [{'type': 'Feature',
            'properties': {'station_name': 'Ürzig(DB)',
             'station_id': 8005945,
             'featureclass': 'Admin-1',
             'nameascii': 'Ürzig(DB)',
             'name': 'Ürzig(DB)',
             'latitude': 7.004806,
             'longitude': 49.995933,
             'geonameid': -1,
             'note': None},
            'geometry': {'type': 'Point', 'coordinates': [49.995933, 7.004806]}},
           {'type': 'Feature',
            'properties': {'station_name': 'Überlingen-Nußdorf',
             'station_id': 8005943,
             'featureclass': 'Admin-1',
             'nameascii': 'Überlingen-Nußdorf',
```

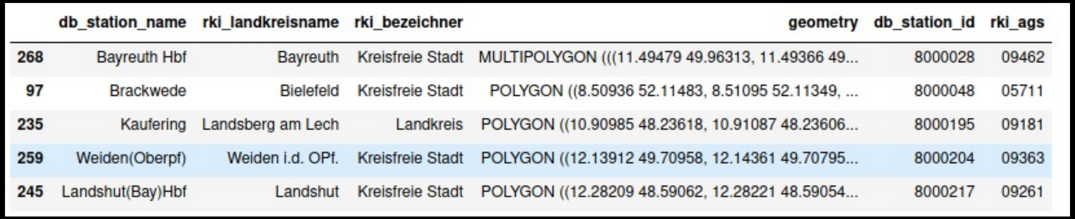
STATION DATA

DB API

[RAILWAY STATIONS]

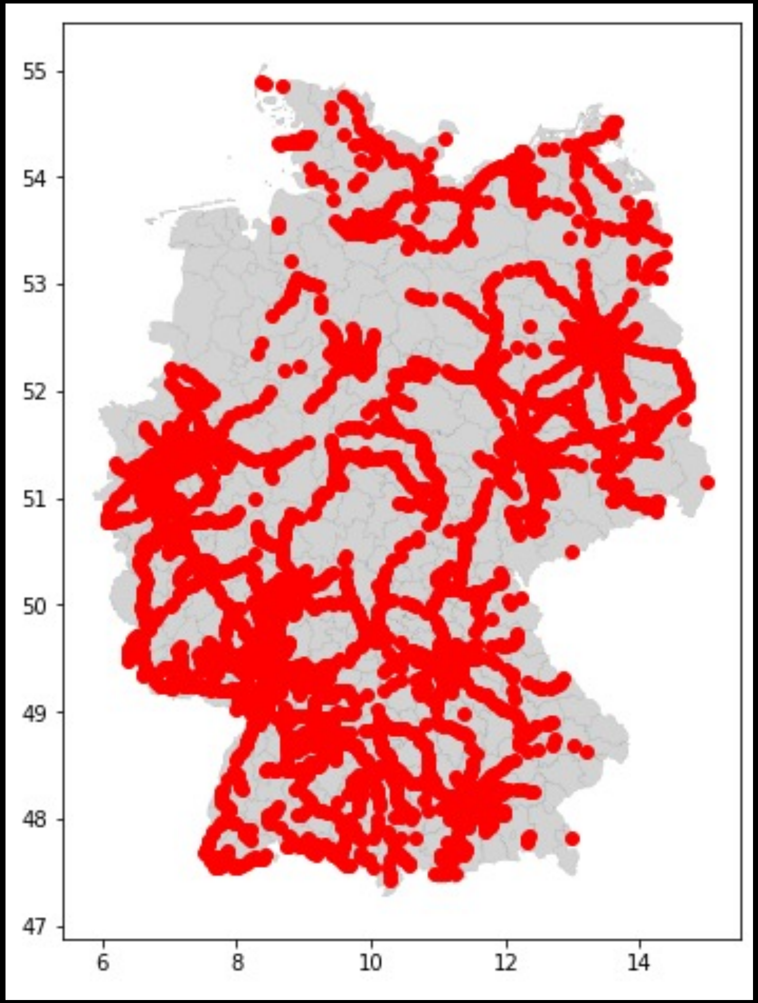


```
In [11]: # community = landkreis_geo #landkreis_geo[landkreis_geo.GEN == 'Flensburg'] # TEST
         station_geo_lkid = gpd.sjoin(landkreis_geo,station_geo_points)
         station_geo_lkid.head(5)
         len(station_geo_lkid)
```



	db_station_name	rki_landkreisname	rki_bezeichner	geometry	db_station_id	rki_ags
268	Bayreuth Hbf	Bayreuth	Kreisfreie Stadt	MULTIPOLYGON (((11.49479 49.96313, 11.49366 49...	8000028	09462
97	Brackwede	Bielefeld	Kreisfreie Stadt	POLYGON ((8.50936 52.11483, 8.51095 52.11349, ...	8000048	05711
235	Kaufering	Landsberg am Lech	Landkreis	POLYGON ((10.90985 48.23618, 10.91087 48.23606...	8000195	09181
259	Weiden(Oberpf)	Weiden i.d. OPf.	Kreisfreie Stadt	POLYGON ((12.13912 49.70958, 12.14361 49.70795...	8000204	09363
245	Landshut(Bay)Hbf	Landshut	Kreisfreie Stadt	POLYGON ((12.28209 48.59062, 12.28221 48.59054...	8000217	09261

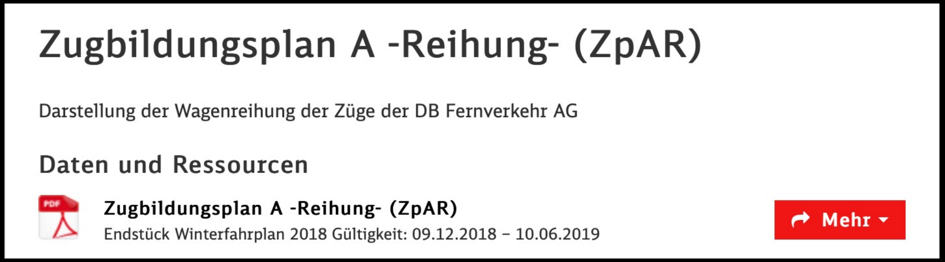
- ONLY DB STATIONS
- NO S-BAHN
- NO 3RD PARTY COMPANIES



TRAIN CAPACITY DATA

DB API

[CAPACITY]

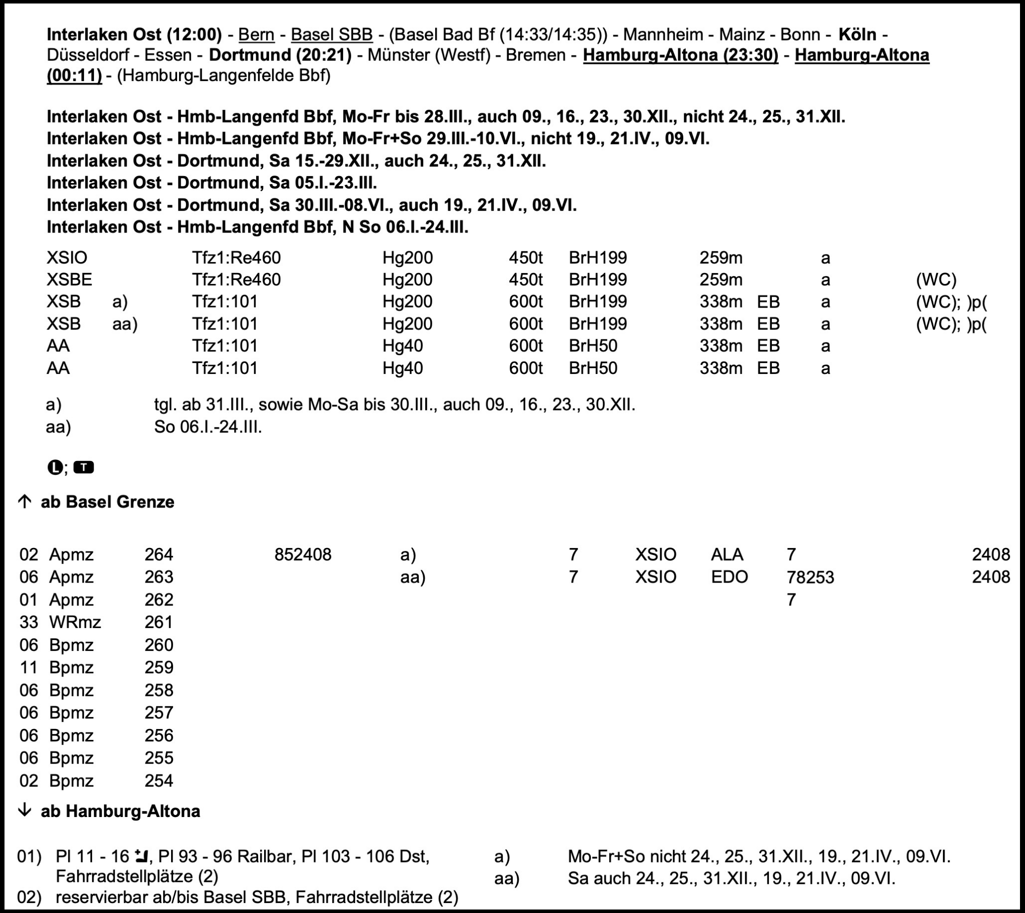


Zugbildungsplan A -Reihung- (ZpAR)

Darstellung der Wagenreihung der Züge der DB Fernverkehr AG

Daten und Ressourcen

Zugbildungsplan A -Reihung- (ZpAR)
Endstück Winterfahrplan 2018 Gültigkeit: 09.12.2018 – 10.06.2019



Interlaken Ost (12:00) - Bern - Basel SBB - (Basel Bad Bf (14:33/14:35)) - Mannheim - Mainz - Bonn - **Köln** - Düsseldorf - Essen - **Dortmund (20:21)** - Münster (Westf) - Bremen - **Hamburg-Altona (23:30)** - **Hamburg-Altona (00:11)** - (Hamburg-Langenfelde Bbf)

Interlaken Ost - Hmb-Langenfd Bbf, Mo-Fr bis 28.III., auch 09., 16., 23., 30.XII., nicht 24., 25., 31.XII.
Interlaken Ost - Hmb-Langenfd Bbf, Mo-Fr+So 29.III.-10.VI., nicht 19., 21.IV., 09.VI.
Interlaken Ost - Dortmund, Sa 15.-29.XII., auch 24., 25., 31.XII.
Interlaken Ost - Dortmund, Sa 05.I.-23.III.
Interlaken Ost - Dortmund, Sa 30.III.-08.VI., auch 19., 21.IV., 09.VI.
Interlaken Ost - Hmb-Langenfd Bbf, N So 06.I.-24.III.

XSIO		Tfz1:Re460	Hg200	450t	BrH199	259m		a	
XSBE		Tfz1:Re460	Hg200	450t	BrH199	259m		a	(WC)
XSB	a)	Tfz1:101	Hg200	600t	BrH199	338m	EB	a	(WC);)p(
XSB	aa)	Tfz1:101	Hg200	600t	BrH199	338m	EB	a	(WC);)p(
AA		Tfz1:101	Hg40	600t	BrH50	338m	EB	a	
AA		Tfz1:101	Hg40	600t	BrH50	338m	EB	a	

a) tgl. ab 31.III., sowie Mo-Sa bis 30.III., auch 09., 16., 23., 30.XII.
aa) So 06.I.-24.III.

Ⓛ; 🅃

↑ ab Basel Grenze

02	Apmz	264	852408	a)	7	XSIO	ALA	7	2408
06	Apmz	263		aa)	7	XSIO	EDO	78253	2408
01	Apmz	262						7	
33	WRmz	261							
06	Bpmz	260							
11	Bpmz	259							
06	Bpmz	258							
06	Bpmz	257							
06	Bpmz	256							
06	Bpmz	255							
02	Bpmz	254							

↓ ab Hamburg-Altona

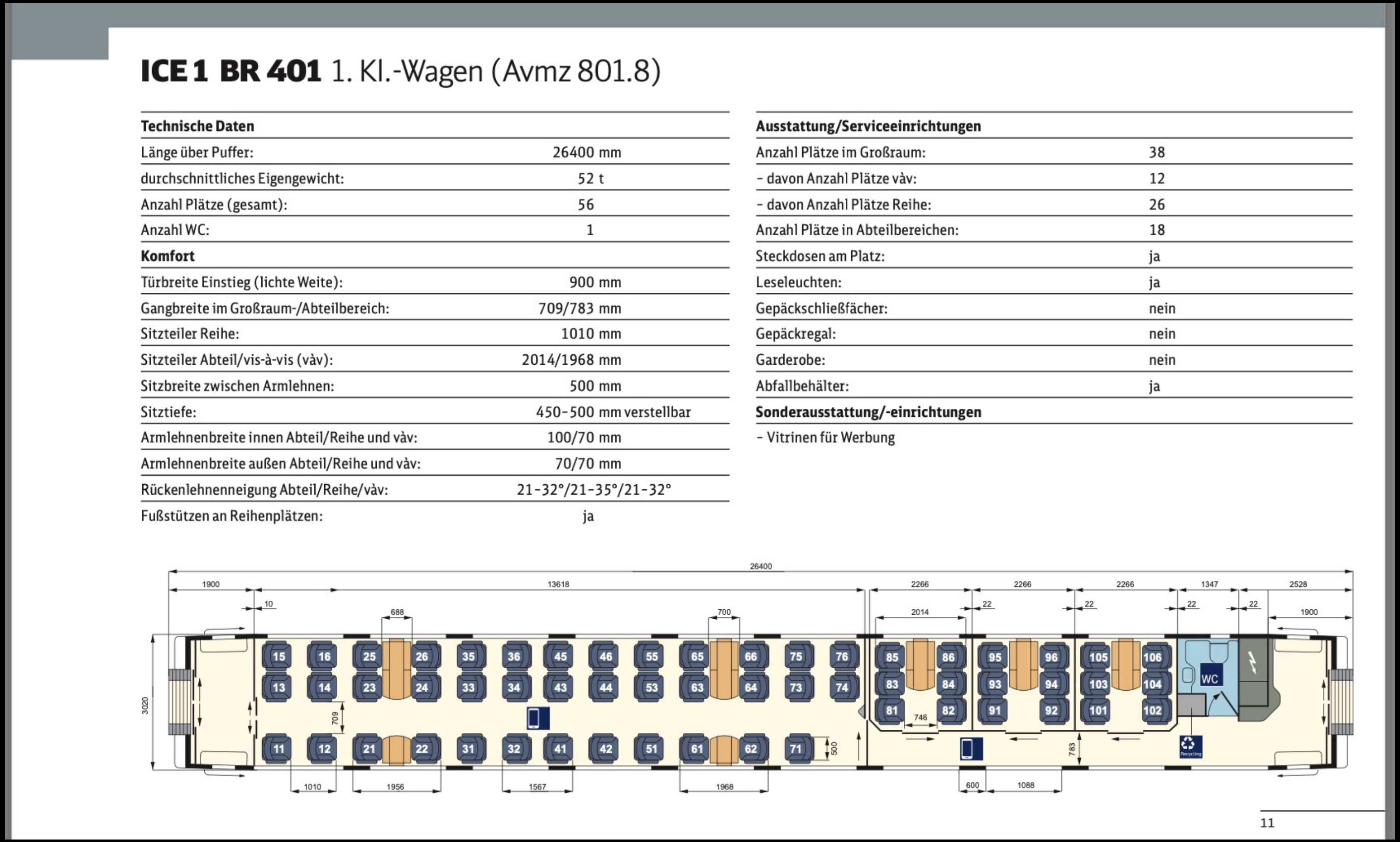
01) Pl 11 - 16, Pl 93 - 96 Railbar, Pl 103 - 106 Dst, Fahrradstellplätze (2)
02) reservierbar ab/bis Basel SBB, Fahrradstellplätze (2)

a) Mo-Fr+So nicht 24., 25., 31.XII., 19., 21.IV., 09.VI.
aa) Sa auch 24., 25., 31.XII., 19., 21.IV., 09.VI.

TRAIN CAPACITY DATA

DB API

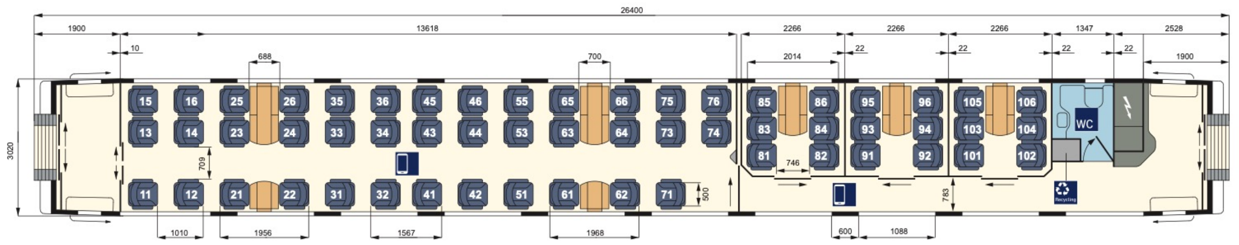
[CAPACITY]



ICE 1 BR 401 1. Kl.-Wagen (Avmz 801.8)

Technische Daten	
Länge über Puffer:	26400 mm
durchschnittliches Eigengewicht:	52 t
Anzahl Plätze (gesamt):	56
Anzahl WC:	1
Komfort	
Türbreite Einstieg (lichte Weite):	900 mm
Gangbreite im Großraum-/Abteilbereich:	709/783 mm
Sitzteiler Reihe:	1010 mm
Sitzteiler Abteil/vis-à-vis (vàv):	2014/1968 mm
Sitzbreite zwischen Armlehnen:	500 mm
Sitztiefe:	450-500 mm verstellbar
Armlehnenbreite innen Abteil/Reihe und vàv:	100/70 mm
Armlehnenbreite außen Abteil/Reihe und vàv:	70/70 mm
Rückenlehnenneigung Abteil/Reihe/vàv:	21-32°/21-35°/21-32°
Fußstützen an Reihenplätzen:	ja

Ausstattung/Serviceeinrichtungen	
Anzahl Plätze im Großraum:	38
– davon Anzahl Plätze vàv:	12
– davon Anzahl Plätze Reihe:	26
Anzahl Plätze in Abteilbereichen:	18
Steckdosen am Platz:	ja
Leseleuchten:	ja
Gepäckschließfächer:	nein
Gepäckregal:	nein
Garderobe:	nein
Abfallbehälter:	ja
Sonderausstattung/-einrichtungen	
– Vitrinen für Werbung	



TRAIN DATA

HAVAS | DB API

[DEPARTURE TABLE]

- OFFICIAL DATA PROVIDERS DO NOT OFFER HISTORICAL DATA ON TRAIN DEPARTURES THAT GOES BACK MORE THAN ONE DAY.

- AND IT IS POSSIBLE TO GET AN IP BAN ALREADY AFTER A FEW REQUESTS PER MINUTE.

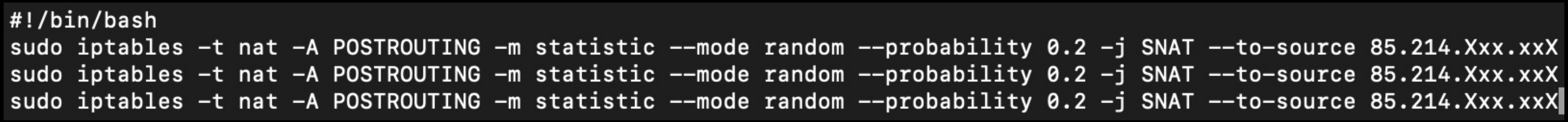
TRAIN DATA

DB API

[DEPARTURE TABLE]

- THE GOAL IS TO AUTOMATICALLY STORE ALL REQUIRED STATIONS AND THEIR DEPARTURE BOARDS BY USING OF SEVERAL SEPARATE OUTBOUND IP ADDRESSES.

- AWS INSTANZ WITH 10 DIFFERENT ADRESSES

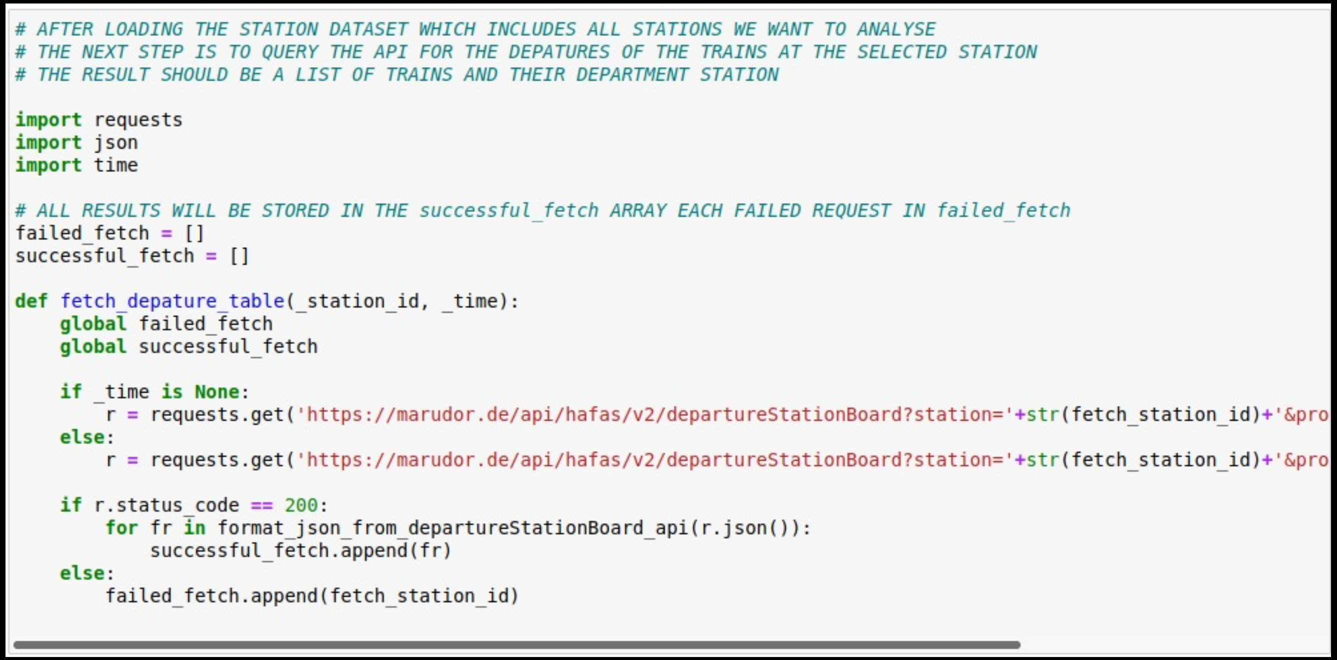


```
#!/bin/bash
sudo iptables -t nat -A POSTROUTING -m statistic --mode random --probability 0.2 -j SNAT --to-source 85.214.Xxx.xxX
sudo iptables -t nat -A POSTROUTING -m statistic --mode random --probability 0.2 -j SNAT --to-source 85.214.Xxx.xxX
sudo iptables -t nat -A POSTROUTING -m statistic --mode random --probability 0.2 -j SNAT --to-source 85.214.Xxx.xxX
```

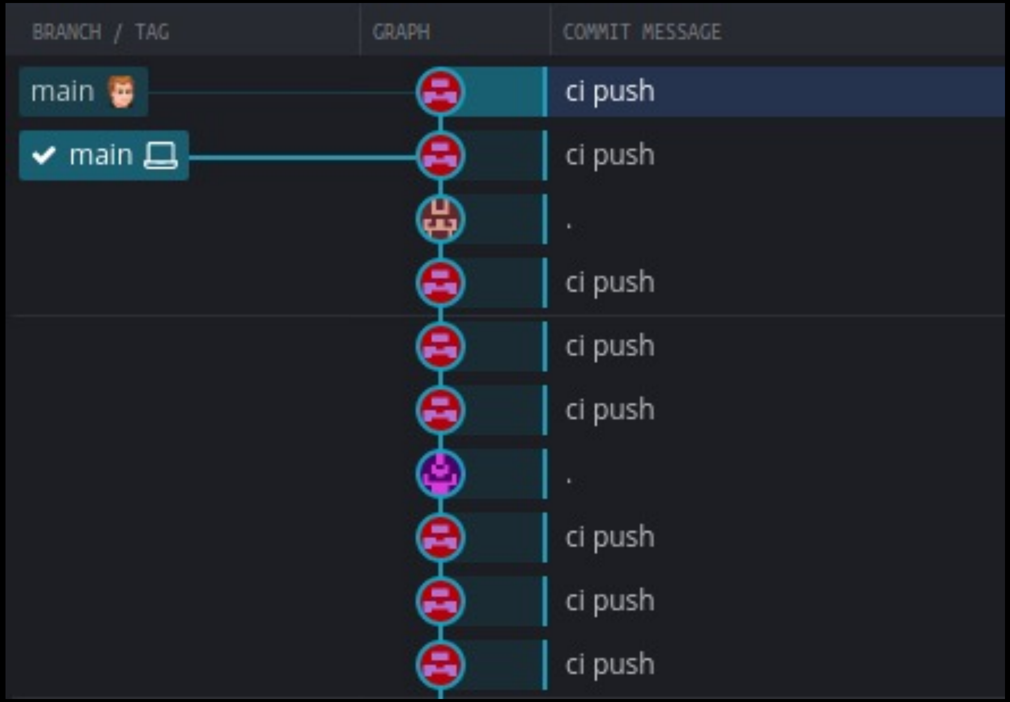
TRAIN DATA

DB API

[DEPARTURE TABLE]



```python
# AFTER LOADING THE STATION DATASET WHICH INCLUDES ALL STATIONS WE WANT TO ANALYSE
# THE NEXT STEP IS TO QUERY THE API FOR THE DEPATURES OF THE TRAINS AT THE SELECTED STATION
# THE RESULT SHOULD BE A LIST OF TRAINS AND THEIR DEPARTMENT STATION

import requests
import json
import time

# ALL RESULTS WILL BE STORED IN THE successful_fetch ARRAY EACH FAILED REQUEST IN failed_fetch
failed_fetch = []
successful_fetch = []

def fetch_depature_table(_station_id, _time):
    global failed_fetch
    global successful_fetch

    if _time is None:
        r = requests.get('https://marudor.de/api/hafas/v2/departureStationBoard?station='+str(fetch_station_id)+'&pro
    else:
        r = requests.get('https://marudor.de/api/hafas/v2/departureStationBoard?station='+str(fetch_station_id)+'&pro

    if r.status_code == 200:
        for fr in format_json_from_departureStationBoard_api(r.json()):
            successful_fetch.append(fr)
    else:
        failed_fetch.append(fetch_station_id)
```



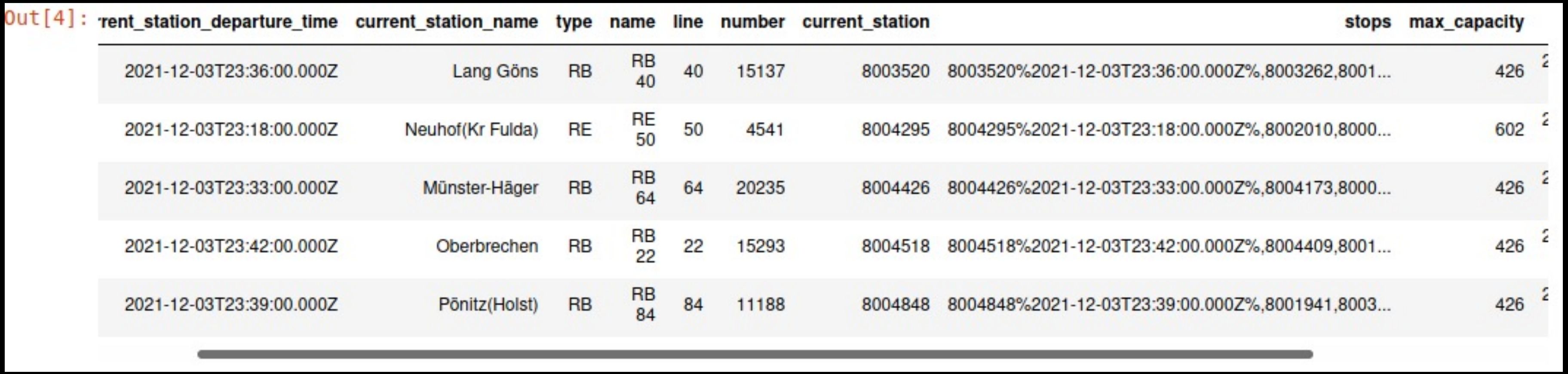
BRANCH / TAG	GRAPH	COMMIT MESSAGE
main		ci push
✓ main		ci push
		.
		ci push
		ci push
		ci push
		.
		ci push
		ci push
		ci push

TRAIN DATA

DB API

[DEPARTURE TABLE]

- CUSTOM DATASET FOR ALL DB TRAINS
- 10 MINUTE SNAPSHOT INTERVAL
- INCLUDING ARRIVAL | DEPARTURE DELAYS



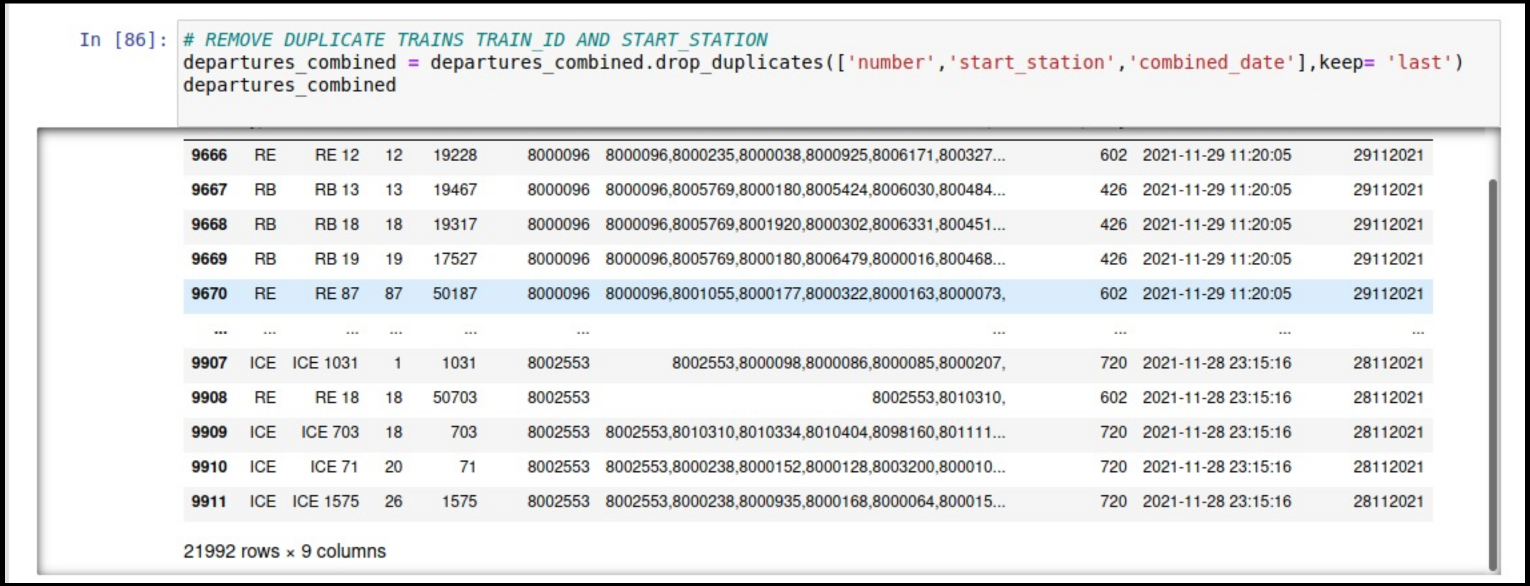
Out[4]:

rent_station_departure_time	current_station_name	type	name	line	number	current_station	stops	max_capacity	
2021-12-03T23:36:00.000Z	Lang Göns	RB	RB 40	40	15137	8003520	8003520%2021-12-03T23:36:00.000Z%,8003262,8001...	426	2
2021-12-03T23:18:00.000Z	Neuhof(Kr Fulda)	RE	RE 50	50	4541	8004295	8004295%2021-12-03T23:18:00.000Z%,8002010,8000...	602	2
2021-12-03T23:33:00.000Z	Münster-Häger	RB	RB 64	64	20235	8004426	8004426%2021-12-03T23:33:00.000Z%,8004173,8000...	426	2
2021-12-03T23:42:00.000Z	Oberbrechen	RB	RB 22	22	15293	8004518	8004518%2021-12-03T23:42:00.000Z%,8004409,8001...	426	2
2021-12-03T23:39:00.000Z	Pönitz(Holst)	RB	RB 84	84	11188	8004848	8004848%2021-12-03T23:39:00.000Z%,8001941,8003...	426	2

TRAIN DATA

DB API

[DEPARTURE TABLE]



In [86]:
```
# REMOVE DUPLICATE TRAINS TRAIN_ID AND START_STATION
departures_combined = departures_combined.drop_duplicates(['number','start_station','combined_date'],keep= 'last')
departures_combined
```

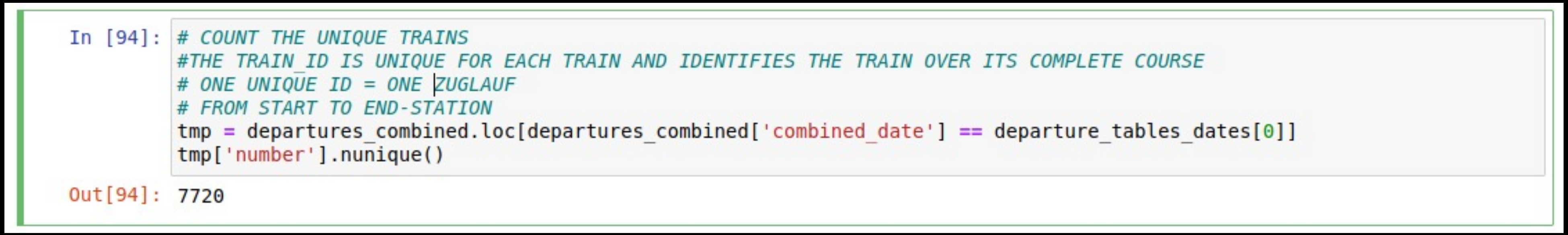
9666	RE	RE 12	12	19228	8000096	8000096,8000235,8000038,8000925,8006171,800327...	602	2021-11-29 11:20:05	29112021
9667	RB	RB 13	13	19467	8000096	8000096,8005769,8000180,8005424,8006030,800484...	426	2021-11-29 11:20:05	29112021
9668	RB	RB 18	18	19317	8000096	8000096,8005769,8001920,8000302,8006331,800451...	426	2021-11-29 11:20:05	29112021
9669	RB	RB 19	19	17527	8000096	8000096,8005769,8000180,8006479,8000016,800468...	426	2021-11-29 11:20:05	29112021
9670	RE	RE 87	87	50187	8000096	8000096,8001055,8000177,8000322,8000163,8000073,	602	2021-11-29 11:20:05	29112021
...	...	...	...	...	...	...	...	...	...
9907	ICE	ICE 1031	1	1031	8002553	8002553,8000098,8000086,8000085,8000207,	720	2021-11-28 23:15:16	28112021
9908	RE	RE 18	18	50703	8002553	8002553,8010310,	602	2021-11-28 23:15:16	28112021
9909	ICE	ICE 703	18	703	8002553	8002553,8010310,8010334,8010404,8098160,801111...	720	2021-11-28 23:15:16	28112021
9910	ICE	ICE 71	20	71	8002553	8002553,8000238,8000152,8000128,8003200,800010...	720	2021-11-28 23:15:16	28112021
9911	ICE	ICE 1575	26	1575	8002553	8002553,8000238,8000935,8000168,8000064,800015...	720	2021-11-28 23:15:16	28112021

21992 rows × 9 columns

TRAIN DATA

DB API

[DEPARTURE TABLE]

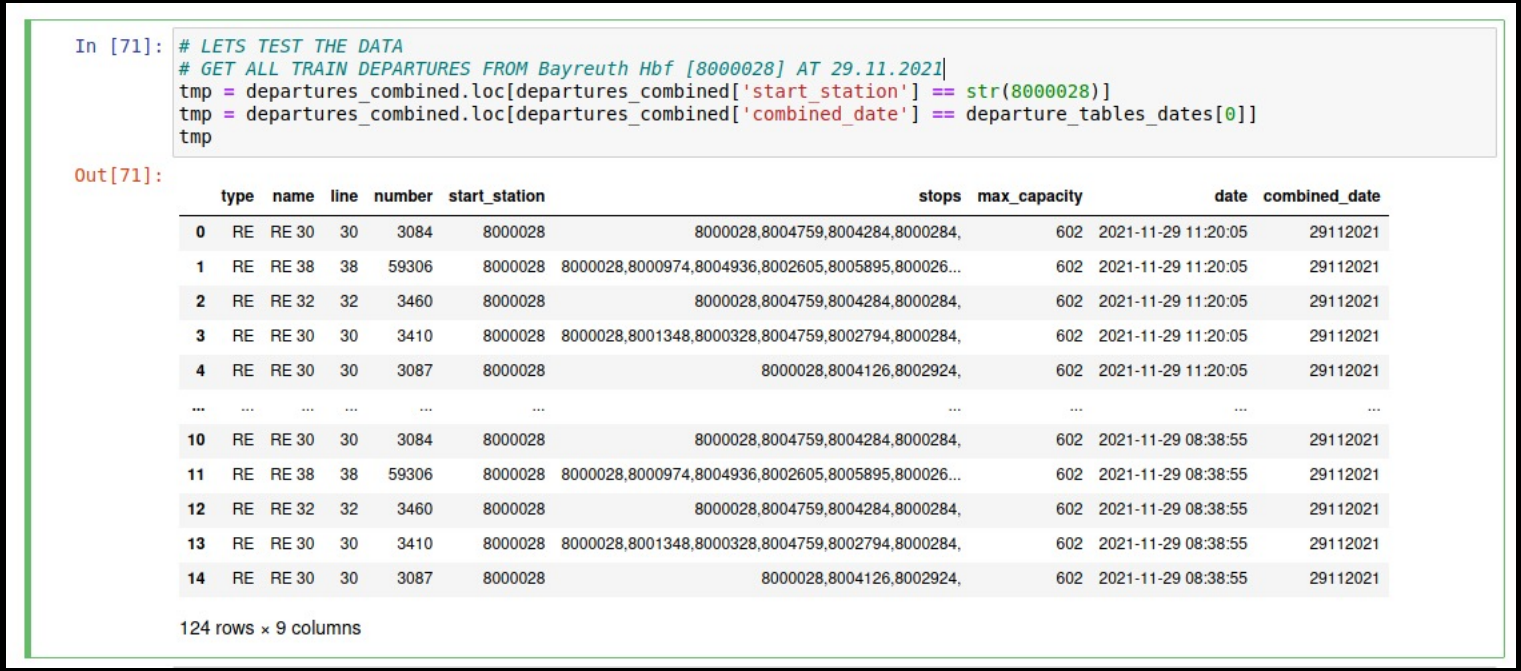


```
In [94]: # COUNT THE UNIQUE TRAINS
         #THE TRAIN_ID IS UNIQUE FOR EACH TRAIN AND IDENTIFIES THE TRAIN OVER ITS COMPLETE COURSE
         # ONE UNIQUE ID = ONE ZUGLAUF
         # FROM START TO END-STATION
         tmp = departures_combined.loc[departures_combined['combined_date'] == departure_tables_dates[0]]
         tmp['number'].nunique()

Out[94]: 7720
```

TRAIN DATA

DB API

[DEPARTURE TABLE]



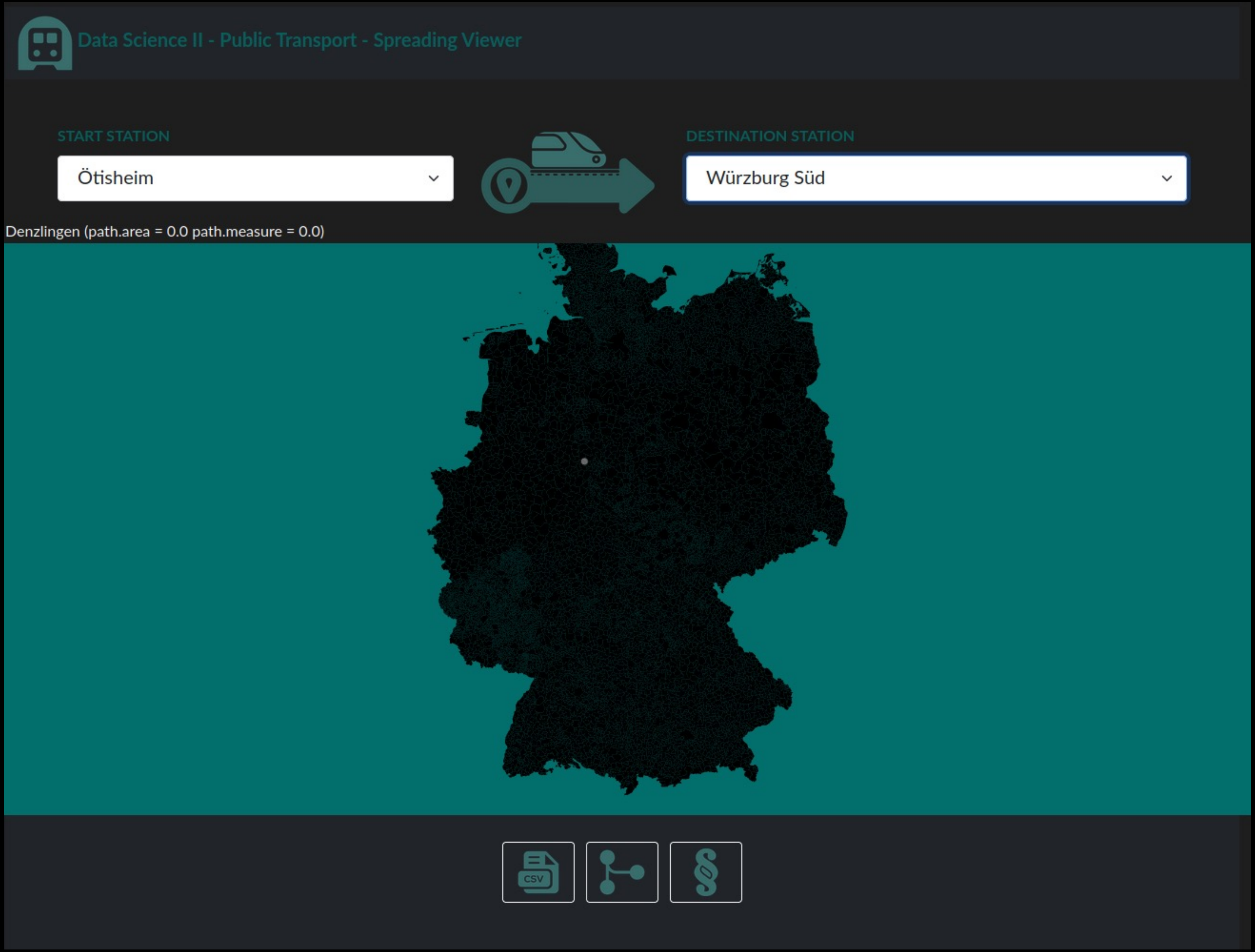
In [71]:
```
# LETS TEST THE DATA
# GET ALL TRAIN DEPARTURES FROM Bayreuth Hbf [8000028] AT 29.11.2021
tmp = departures_combined.loc[departures_combined['start_station'] == str(8000028)]
tmp = departures_combined.loc[departures_combined['combined_date'] == departure_tables_dates[0]]
tmp
```

Out[71]:

	type	name	line	number	start_station	stops	max_capacity	date	combined_date
0	RE	RE 30	30	3084	8000028	8000028,8004759,8004284,8000284,	602	2021-11-29 11:20:05	29112021
1	RE	RE 38	38	59306	8000028	8000028,8000974,8004936,8002605,8005895,800026...	602	2021-11-29 11:20:05	29112021
2	RE	RE 32	32	3460	8000028	8000028,8004759,8004284,8000284,	602	2021-11-29 11:20:05	29112021
3	RE	RE 30	30	3410	8000028	8000028,8001348,8000328,8004759,8002794,8000284,	602	2021-11-29 11:20:05	29112021
4	RE	RE 30	30	3087	8000028	8000028,8004126,8002924,	602	2021-11-29 11:20:05	29112021
...	...	...	...	...	...	...	...	...	...
10	RE	RE 30	30	3084	8000028	8000028,8004759,8004284,8000284,	602	2021-11-29 08:38:55	29112021
11	RE	RE 38	38	59306	8000028	8000028,8000974,8004936,8002605,8005895,800026...	602	2021-11-29 08:38:55	29112021
12	RE	RE 32	32	3460	8000028	8000028,8004759,8004284,8000284,	602	2021-11-29 08:38:55	29112021
13	RE	RE 30	30	3410	8000028	8000028,8001348,8000328,8004759,8002794,8000284,	602	2021-11-29 08:38:55	29112021
14	RE	RE 30	30	3087	8000028	8000028,8004126,8002924,	602	2021-11-29 08:38:55	29112021

124 rows × 9 columns

VISUALISATION



Denzlingen (path.area = 0.0 path.measure = 0.0)

- TS + NODEJS BACKEND
- D3 MAP RENDERING
- WEBSOCKETS FOR REALTIME UPDATES

TIMELINE

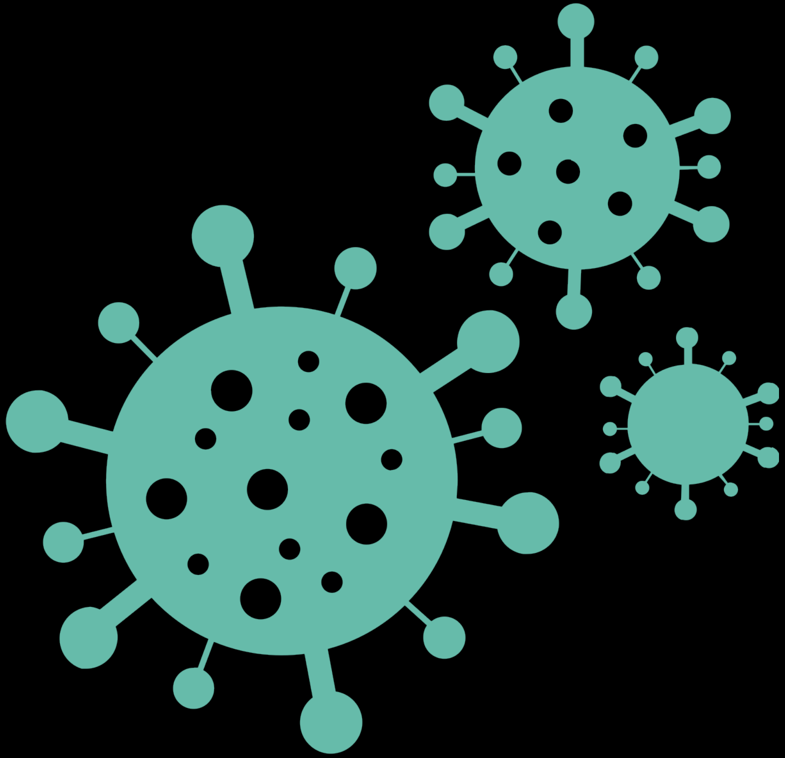
GENERATE HEATMAPS FOR EACH DAY IN HISTORY

BUILD AN INTERACTIVE MAP

CREATE VIRUS SPREAD ANIMATION

INTERMEDIATE

FINAL



QUESTIONS ?