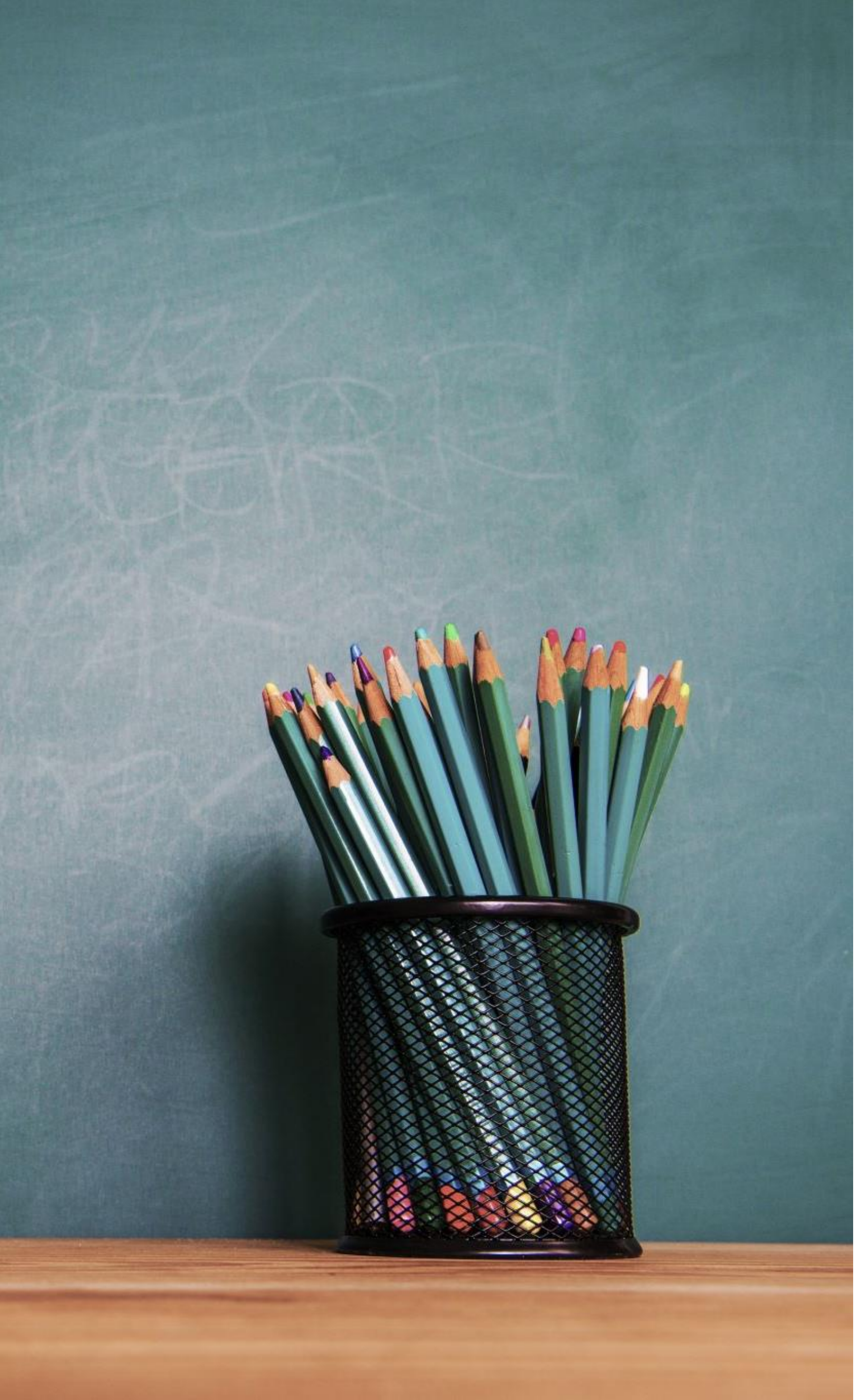
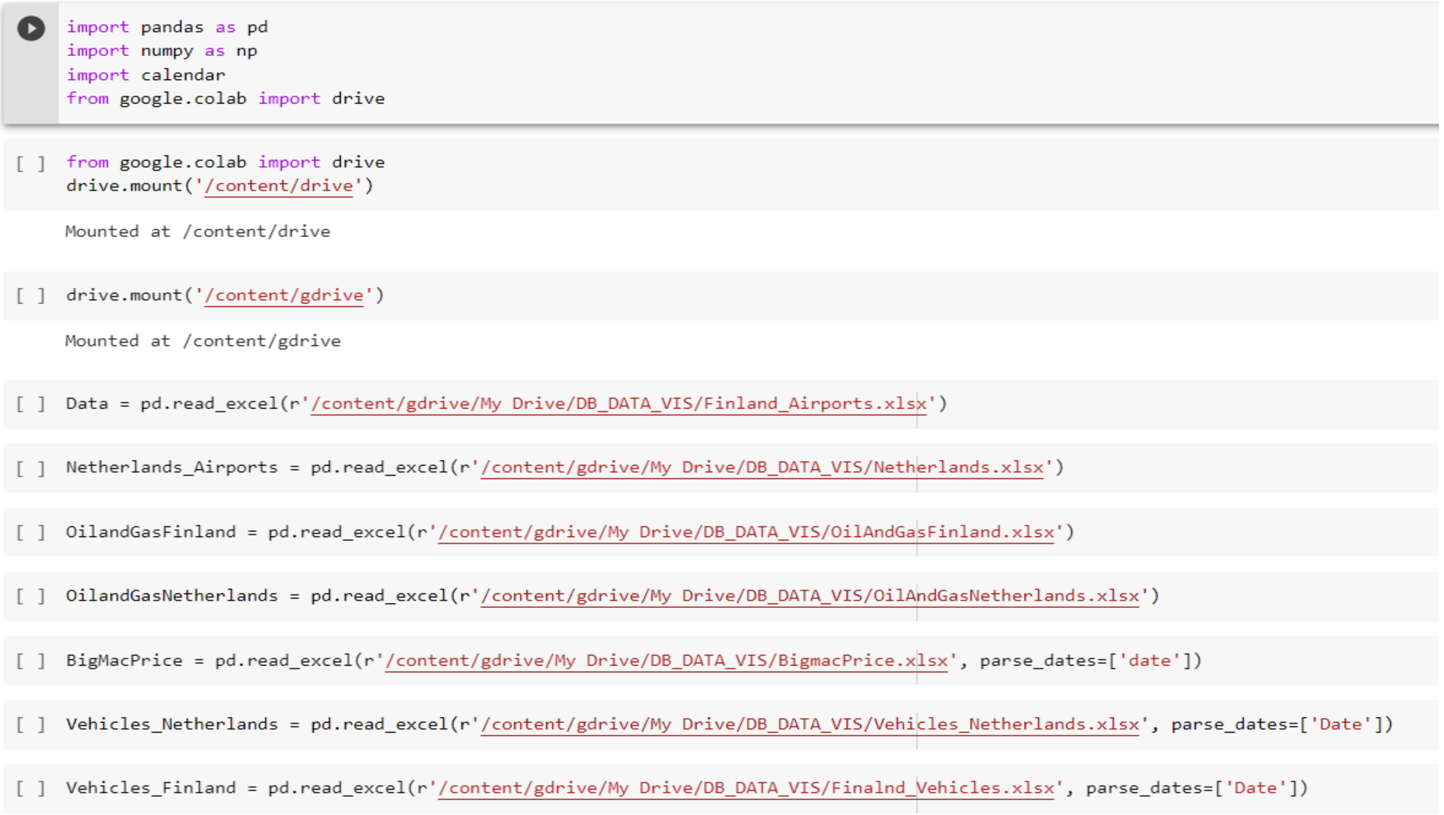


The main aim of the project is to inform the user how the inflation affect goods and services which are inevitable part of our lives. By deeply analyzing different petroleum products prices the project goal is to show if the inflation has an impact on the previously mentioned areas.

Data sources:

- Oil and Gas Data:
  - Nethelrnads : https://countryeconomy.com/energy/prices-gasoline-gas-oil-heating/netherlands
  - Finland :https://countryeconomy.com/energy/prices-gasoline-gas-oil-heating/finland
- New car registrations :
  - Netherlands: https://countryeconomy.com/business/car-registrations/netherlands
  - Finland: https://countryeconomy.com/business/car-registrations/finland
- Airports Data:
  - Netherlands: https://opendata.cbs.nl/#/CBS/en/dataset/37478eng/table
  - Finland: https://www.stat.fi/en/publication/ckfnuwcqw1nc001020brv9ecc
- BigMac Prices:
  - For both countries: https://www.kaggle.com/datasets/vittoriogiatti/bigmacprice
- Latitude and Longitude
  - For both countries: https://www.latlong.net/

1. Import



```
import pandas as pd
import numpy as np
import calendar
from google.colab import drive
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
Data = pd.read_excel(r'/content/gdrive/My Drive/DB_DATA_VIS/Finland_Airports.xlsx')
```

```
Netherlands_Airports = pd.read_excel(r'/content/gdrive/My Drive/DB_DATA_VIS/Netherlands.xlsx')
```

```
OilandGasFinland = pd.read_excel(r'/content/gdrive/My Drive/DB_DATA_VIS/OilAndGasFinland.xlsx')
```

```
OilandGasNetherlands = pd.read_excel(r'/content/gdrive/My Drive/DB_DATA_VIS/OilAndGasNetherlands.xlsx')
```

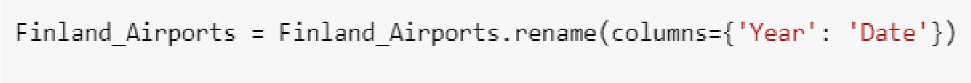
```
BigMacPrice = pd.read_excel(r'/content/gdrive/My Drive/DB_DATA_VIS/BigmacPrice.xlsx', parse_dates=['date'])
```

```
Vehicles_Netherlands = pd.read_excel(r'/content/gdrive/My Drive/DB_DATA_VIS/Vehicles_Netherlands.xlsx', parse_dates=['Date'])
```

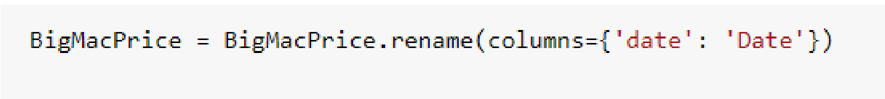
```
Vehicles_Finland = pd.read_excel(r'/content/gdrive/My Drive/DB_DATA_VIS/Finalnd_Vehicles.xlsx', parse_dates=['Date'])
```

2. Data Cleaning Process

- Renaming columns

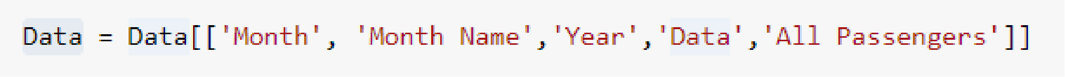


```
Finland_Airports = Finland_Airports.rename(columns={'Year': 'Date'})
```

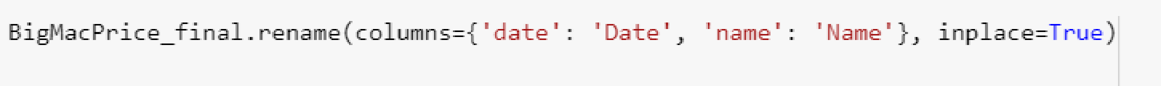


```
BigMacPrice = BigMacPrice.rename(columns={'date': 'Date'})
```

  - Same naming convention

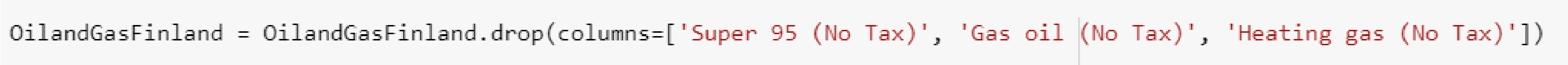


```
Data = Data[['Month', 'Month Name','Year','Data','All Passengers']]
```

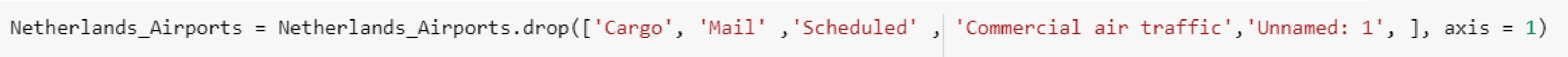


```
BigMacPrice_final.rename(columns={'date': 'Date', 'name': 'Name'}, inplace=True)
```

  - Removing columns

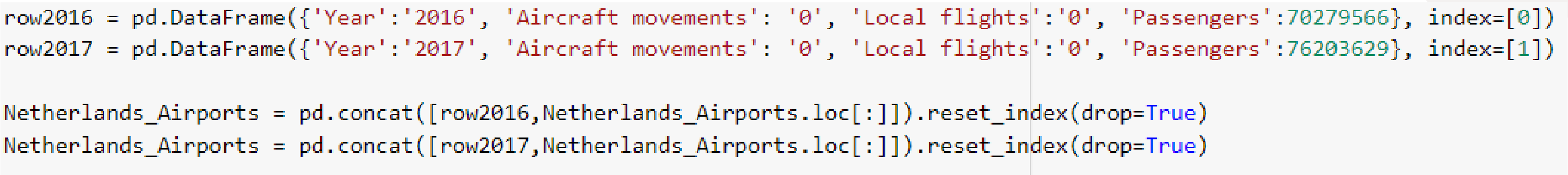


```
OilandGasFinland = OilandGasFinland.drop(columns=['Super 95 (No Tax)', 'Gas oil (No Tax)', 'Heating gas (No Tax)'])
```



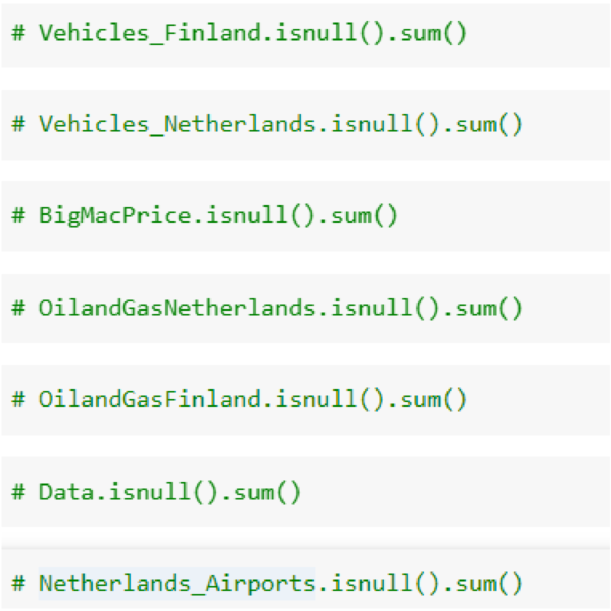
```
Netherlands_Airports = Netherlands_Airports.drop(['Cargo', 'Mail' ,'Scheduled' , 'Commercial air traffic','Unnamed: 1', ], axis = 1)
```

  - Adding new rows
  - Create the rows
  - Add it to the table
  - Reset the index



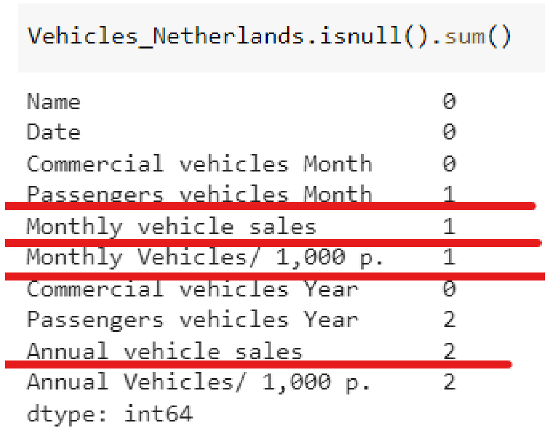
```
row2016 = pd.DataFrame({'Year':'2016', 'Aircraft movements': '0', 'Local flights':'0', 'Passengers':70279566}, index=[0])
row2017 = pd.DataFrame({'Year':'2017', 'Aircraft movements': '0', 'Local flights':'0', 'Passengers':76203629}, index=[1])

Netherlands_Airports = pd.concat([row2016,Netherlands_Airports.loc[:]]).reset_index(drop=True)
Netherlands_Airports = pd.concat([row2017,Netherlands_Airports.loc[:]]).reset_index(drop=True)
```

- Checking for null values



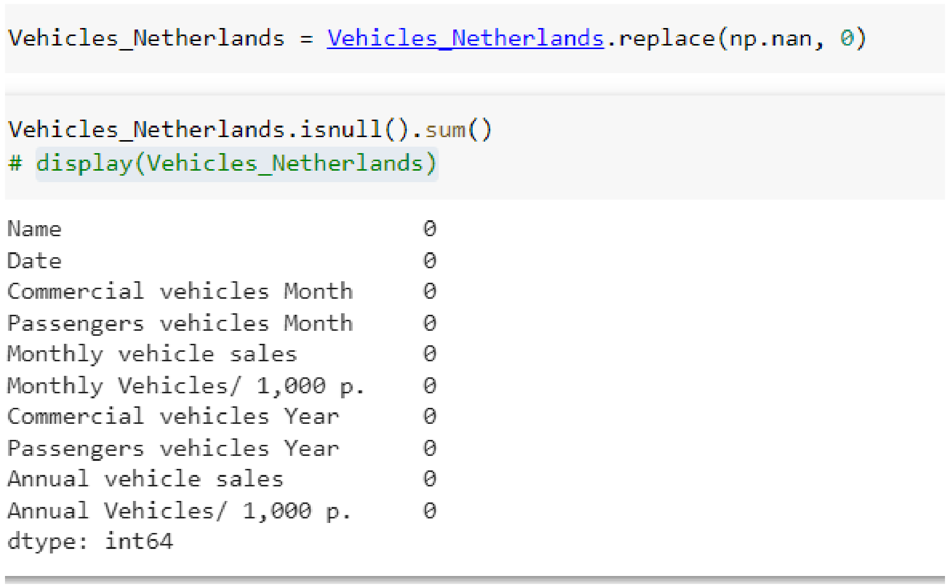
```
# Vehicles_Finland.isnull().sum()

# Vehicles_Netherlands.isnull().sum()

# BigMacPrice.isnull().sum()

# OilandGasNetherlands.isnull().sum()

# OilandGasFinland.isnull().sum()

# Data.isnull().sum()

# Netherlands_Airports.isnull().sum()
```

- If found:



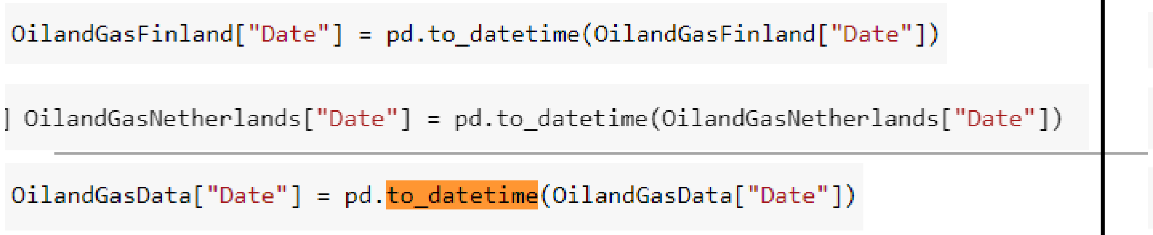
```
Vehicles_Netherlands.isnull().sum()

Name                           0
Date                           0
Commercial vehicles Month      0
Passengers vehicles Month      1
Monthly vehicle sales          1
Monthly Vehicles/ 1,000 p.     1
Commercial vehicles Year       0
Passengers vehicles Year       2
Annual vehicle sales           2
Annual Vehicles/ 1,000 p.      2
dtype: int64
```

- Dealing with it:

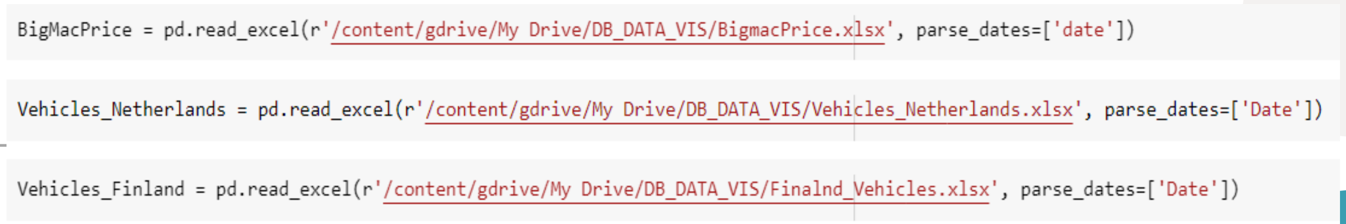


```
Vehicles_Netherlands = Vehicles_Netherlands.replace(np.nan, 0)

Vehicles_Netherlands.isnull().sum()
# display(Vehicles_Netherlands)

Name                          0
Date                          0
Commercial vehicles Month     0
Passengers vehicles Month     0
Monthly vehicle sales         0
Monthly Vehicles/ 1,000 p.    0
Commercial vehicles Year      0
Passengers vehicles Year      0
Annual vehicle sales          0
Annual Vehicles/ 1,000 p.     0
dtype: int64
```

- Converting Date column to DateTime (2 ways)

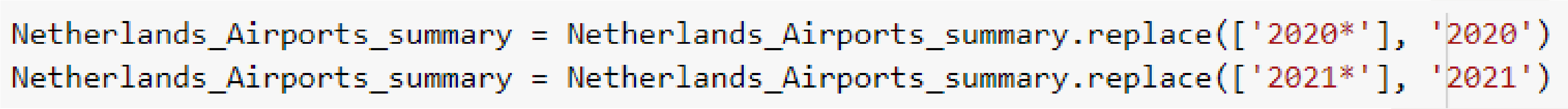


```
OilandGasFinland["Date"] = pd.to_datetime(OilandGasFinland["Date"])

] OilandGasNetherlands["Date"] = pd.to_datetime(OilandGasNetherlands["Date"])

OilandGasData["Date"] = pd.to_datetime(OilandGasData["Date"])
```



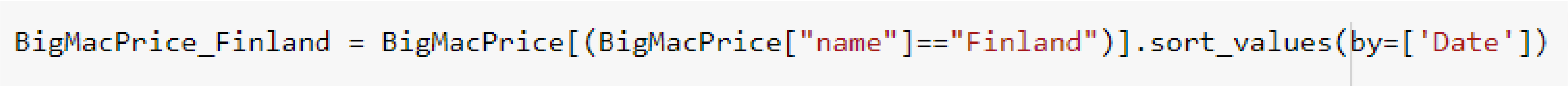
```
BigMacPrice = pd.read_excel(r'/content/gdrive/My Drive/DB_DATA_VIS/BigmacPrice.xlsx', parse_dates=['date'])

Vehicles_Netherlands = pd.read_excel(r'/content/gdrive/My Drive/DB_DATA_VIS/Vehicles_Netherlands.xlsx', parse_dates=['Date'])

Vehicles_Finland = pd.read_excel(r'/content/gdrive/My Drive/DB_DATA_VIS/Finalnd_Vehicles.xlsx', parse_dates=['Date'])
```

- Replacing values

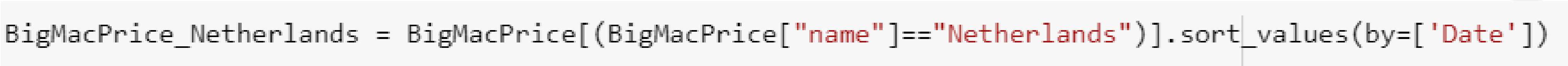


```
Netherlands_Airports_summary = Netherlands_Airports_summary.replace(['2020*'], '2020')
Netherlands_Airports_summary = Netherlands_Airports_summary.replace(['2021*'], '2021')
```

- Extracting data based on a condition

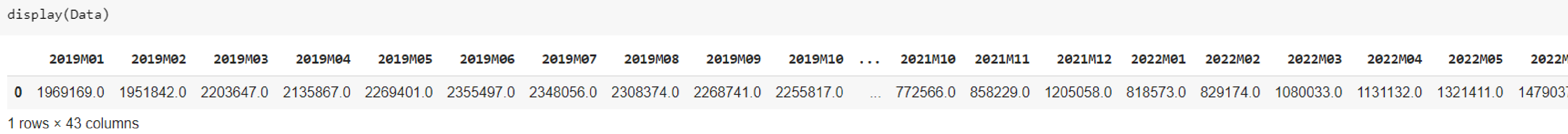


```
BigMacPrice_Finland = BigMacPrice[(BigMacPrice["name"]=="Finland")].sort_values(by=['Date'])
```



```
BigMacPrice_Netherlands = BigMacPrice[(BigMacPrice["name"]=="Netherlands")].sort_values(by=['Date'])
```

3. Table Transforming

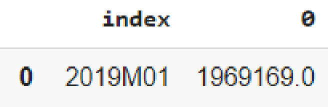


```
display(Data)
```

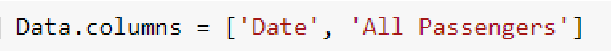
	2019M01	2019M02	2019M03	2019M04	2019M05	2019M06	2019M07	2019M08	2019M09	2019M10	...	2021M10	2021M11	2021M12	2022M01	2022M02	2022M03	2022M04	2022M05	2022M
0	1969169.0	1951842.0	2203647.0	2135867.0	2269401.0	2355497.0	2348056.0	2308374.0	2268741.0	2255817.0	...	772566.0	858229.0	1205058.0	818573.0	829174.0	1080033.0	1131132.0	1321411.0	147903

1 rows × 43 columns

- .transpose()
- Renaming columns



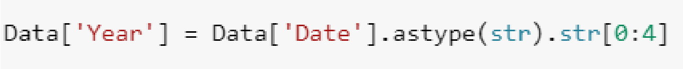
	index	0
0	2019M01	1969169.0



```
Data.columns = ['Date', 'All Passengers']
```

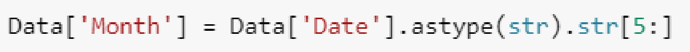
- Reformatting the Date

  Get the year



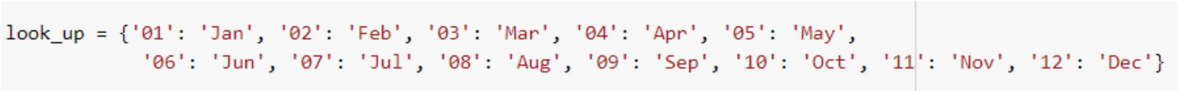
```
Data['Year'] = Data['Date'].astype(str).str[0:4]
```

  Get the month number



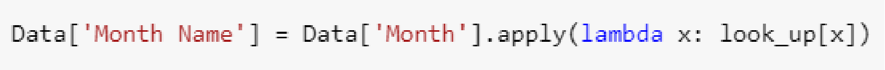
```
Data['Month'] = Data['Date'].astype(str).str[5:]
```

  - Create a dictionary mapping the month number with its name



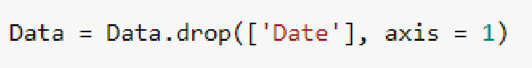
```
look_up = {'01': 'Jan', '02': 'Feb', '03': 'Mar', '04': 'Apr', '05': 'May',
           '06': 'Jun', '07': 'Jul', '08': 'Aug', '09': 'Sep', '10': 'Oct', '11': 'Nov', '12': 'Dec'}
```

  - Use a for loop to populate the table with the month name



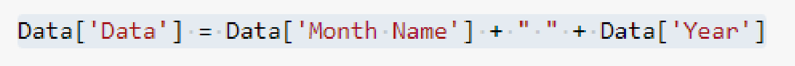
```
Data['Month Name'] = Data['Month'].apply(lambda x: look_up[x])
```

  - Delete the old Date column



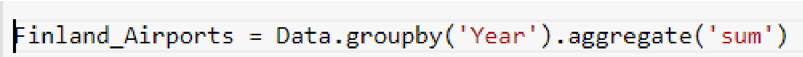
```
Data = Data.drop(['Date'], axis = 1)
```

  - Create a new column combining the Month and Year



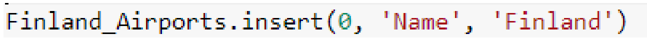
```
Data['Data'] = Data['Month Name'] + " " + Data['Year']
```

- Group by Year and aggregate sum



```
Finland_Airports = Data.groupby('Year').aggregate('sum')
```

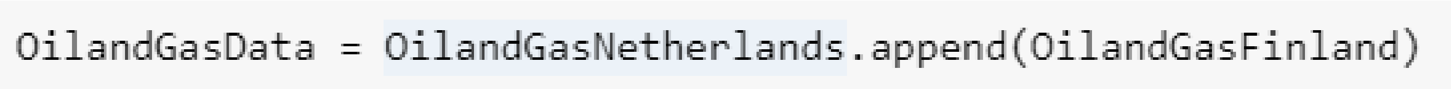
- Add a new column to each table using the same spelling



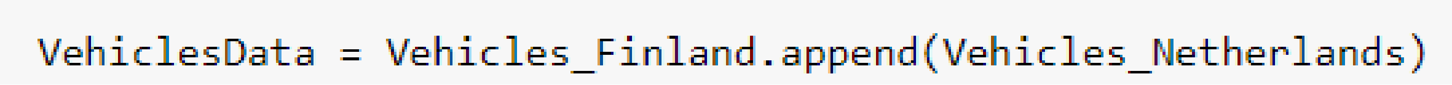
```
Finland_Airports.insert(0, 'Name', 'Finland')
```

4. .append() to combine two tables

- Create a dataframe combining the tables for all countries:



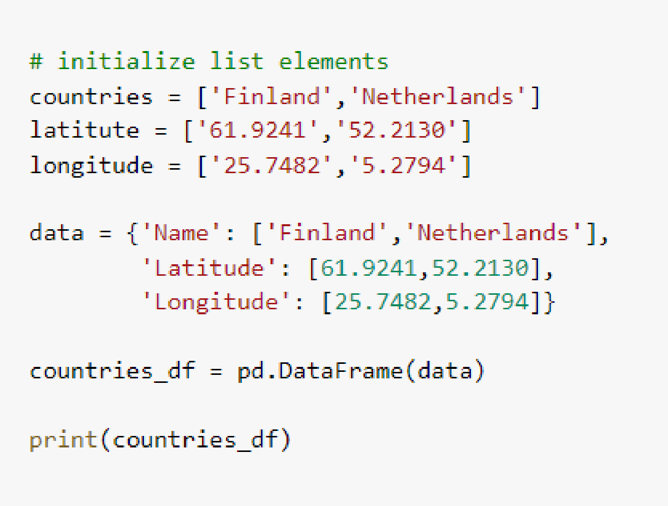
```
OilandGasData = OilandGasNetherlands.append(OilandGasFinland)
```



```
VehiclesData = Vehicles_Finland.append(Vehicles_Netherlands)
```

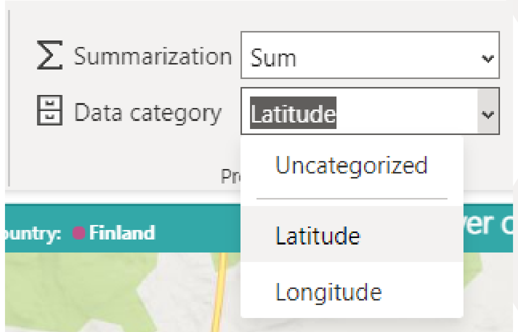
5. Create a new countries_df table. Why?

- The country name will be used a slicer in the dashboard
- It will be used in the map visualization



```
# initialize list elements
countries = ['Finland','Netherlands']
latitute = ['61.9241','52.2130']
longitude = ['25.7482','5.2794']

data = {'Name': ['Finland','Netherlands'],
        'Latitude': [61.9241,52.2130],
        'Longitude': [25.7482,5.2794]}

countries_df = pd.DataFrame(data)

print(countries_df)
```

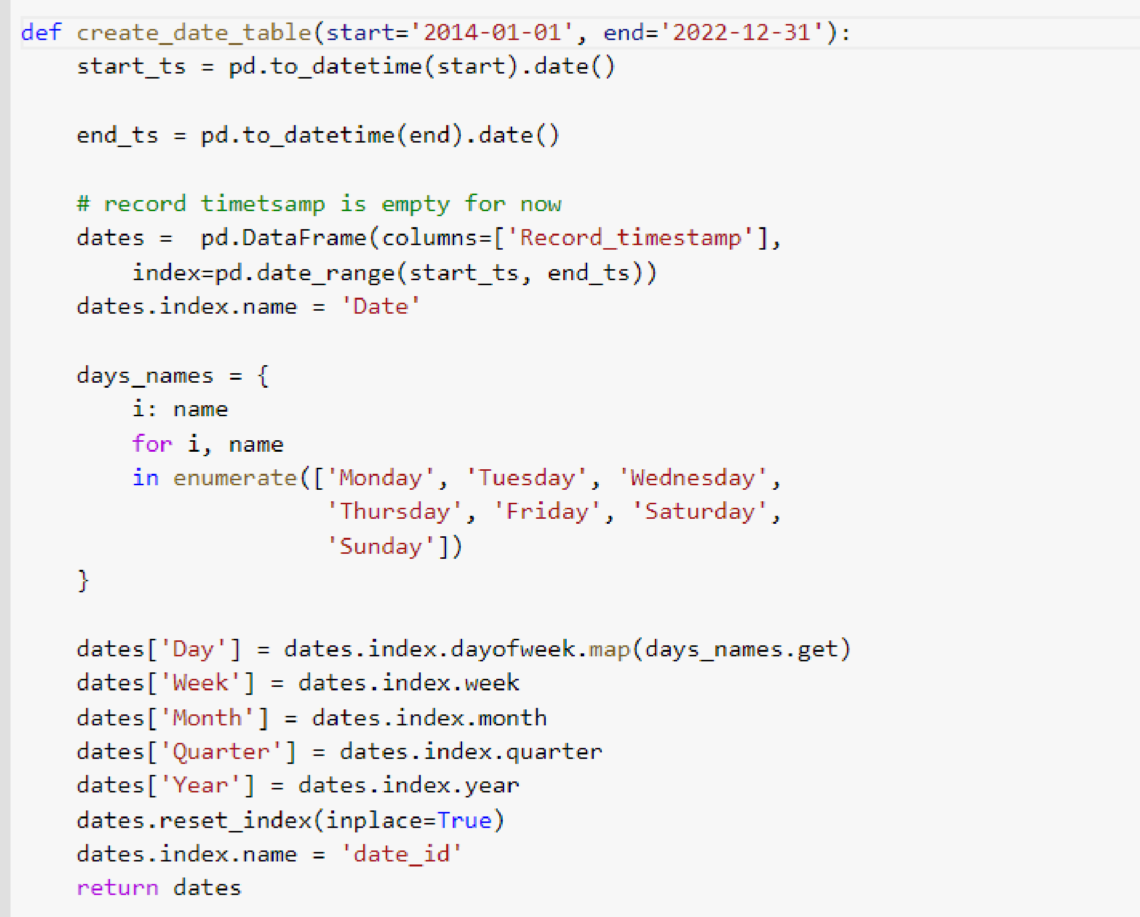
- 5.1 Change Data Category for the map visual



Summarization Sum

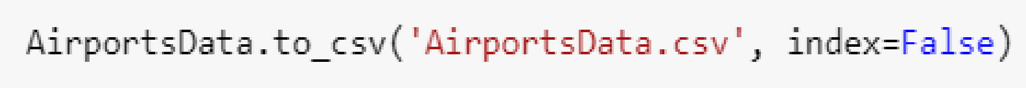
Data category Latitude

6. Create a new Date table

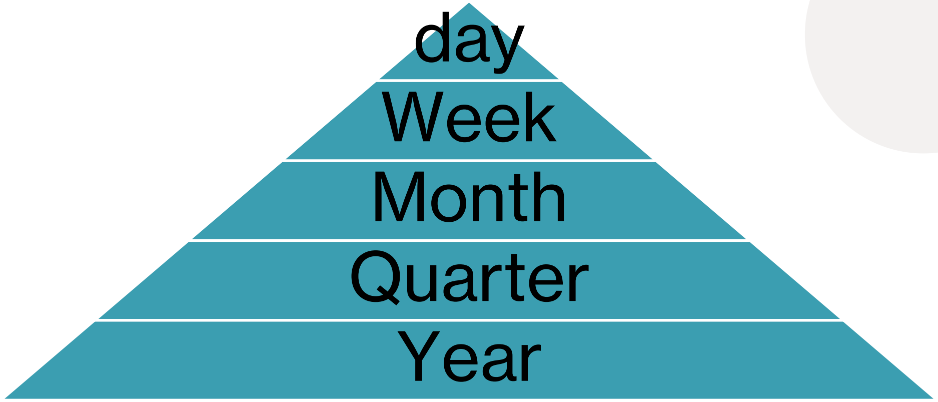


```
def create_date_table(start='2014-01-01', end='2022-12-31'):
    start_ts = pd.to_datetime(start).date()

    end_ts = pd.to_datetime(end).date()

    # record timetsamp is empty for now
    dates =  pd.DataFrame(columns=['Record_timestamp'],
        index=pd.date_range(start_ts, end_ts))
    dates.index.name = 'Date'

    days_names = {
        i: name
        for i, name
        in enumerate(['Monday', 'Tuesday', 'Wednesday',
                      'Thursday', 'Friday', 'Saturday',
                      'Sunday'])
    }

    dates['Day'] = dates.index.dayofweek.map(days_names.get)
    dates['Week'] = dates.index.week
    dates['Month'] = dates.index.month
    dates['Quarter'] = dates.index.quarter
    dates['Year'] = dates.index.year
    dates.reset_index(inplace=True)
    dates.index.name = 'date_id'
    return dates
```

7. Save the tables into .csv file



```
AirportsData.to_csv('AirportsData.csv', index=False)
```



day

Week

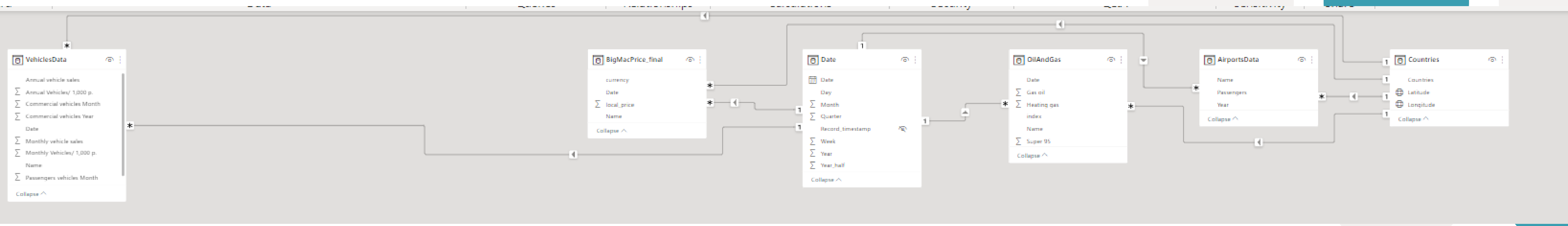
Month

Quarter

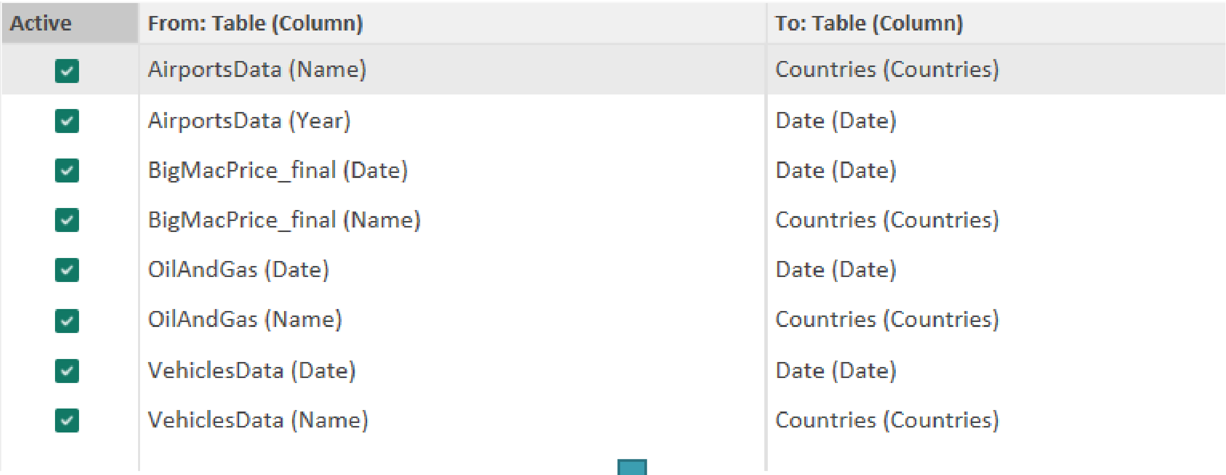
Year

9. PowerBi Relationships

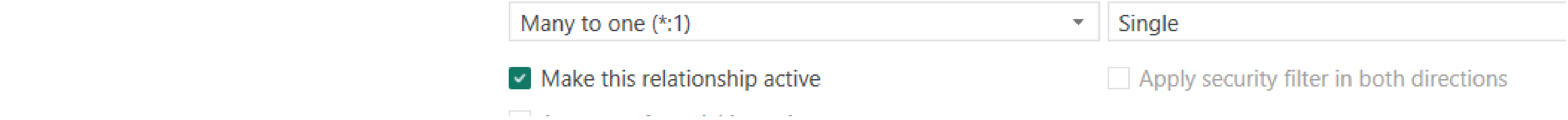
Star schema:



Model:



Active	From: Table (Column)	To: Table (Column)
☑	AirportsData (Name)	Countries (Countries)
☑	AirportsData (Year)	Date (Date)
☑	BigMacPrice_final (Date)	Date (Date)
☑	BigMacPrice_final (Name)	Countries (Countries)
☑	OilAndGas (Date)	Date (Date)
☑	OilAndGas (Name)	Countries (Countries)
☑	VehiclesData (Date)	Date (Date)
☑	VehiclesData (Name)	Countries (Countries)



Cardinality: Many to one (*:1)

Cross filter direction: Single

☑ Make this relationship active

□ Apply security filter in both directions

10. Reflective Statement

- The project was a good exercise to gather information, researching different topics and forming a base for a dashboard. During the process, I was able to experiment with different data cleaning and manipulation techniques which are an inevitable part of every real-world project.
- Using Python and Pandas during the model I sharpened my knowledge. I believe I can apply my awareness in AI and ML for example.
- PowerBi was a great tool to present my findings and learn how to work with multiple tables and make the dashboard more interactive.
- With the newly gained knowledge I believe a have a solid base for my future projects in data science.