Contents

1 SQUAD FORECASTING

ETL Project

> Helen Navarro
> Lien Chin
> Sergio Salvador
> Rubén Tenreiro

2 Importing Libraries

In [1]:

```python
import pandas as pd
import requests
import time
import os
import json
```

3 Extraction

3.1 Postal Codes csv

In [2]:

```python
# opening csv file

ExtracionCSV1 = pd.read_csv("RAW/ETL_project-postal_codes.csv", sep=';', encoding= "utf-8")
```

In [3]:

```python
ExtracionCSV1
```

Out[3]:

	provincia	poblacion	Código Postal
0	Araba/Álava	Alegría-Dulantzi	240
1	Ávila	Candeleda	548
2	Araba/Álava	Vitoria-Gasteiz	1001
3	Araba/Álava	Vitoria-Gasteiz	1002
4	Araba/Álava	Vitoria-Gasteiz	1003
...	...	...	...
14660	Melilla	Melilla	52004
14661	Melilla	Melilla	52005
14662	Melilla	Melilla	52006
14663	Tarragona	Calafell	73820
14664	Zaragoza	Zaragoza	90007

14665 rows × 3 columns

Contents

3.2 Girona

In [4]:

```
def datos_Girona():

    print("Getting Girona data from the INE.",
          "\n", "Connecting...", "\n", sep = "")

    inicio = time.time() # starts a timer

    basic_headers = {'User-Agent': 'Mozilla/5.0'} # sets up basic headers for the request

    url = 'https://servicios.ine.es/wstempus/js/es/DATOS_TABLA/33791?tip=AM' # assign the url for G

    payload = requests.get(url, headers = basic_headers) # requests data from the INE website

    _JSON = payload.json() # converts data to json format

    with open("RAW/Girona.json", "w+") as f: # open "Girona.json" file in "RAW" directory

        json.dump(_JSON, f) # dumps data into the file

    op_time = time.time()-inicio # calculates operation time

    print("Data saved in ~/RAW/Girona.json.", "\n",

          f"This operation took {round(op_time, 1)} seconds", sep = "") # prints message with save
```

In [5]:

```
datos_Girona()
```

```
Getting Girona data from the INE.
Connecting...

Data saved in ~/RAW/Girona.json.
This operation took 95.3 seconds
```

3.3 Censo general csv

In [6]:

```
# importing file from the folder RAW

ExtracionCSV2 = pd.read_csv("RAW/55244 (1).csv", sep=';', low_memory=False)

ExtracionCSV2
```

	Nacionalidad (grandes grupos)	Total Nacional	Municipios	Sexo	Unidades de medida	Total
0	TOTAL	Total Nacional	NaN	Ambos sexos	Personas	47.400.798
1	TOTAL	Total Nacional	NaN	Hombres	Personas	23.248.611
2	TOTAL	Total Nacional	NaN	Mujeres	Personas	24.152.187
3	TOTAL	Total Nacional	01051 Agurain/Salvatierra	Ambos sexos	Personas	5.022
4	TOTAL	Total Nacional	01051 Agurain/Salvatierra	Hombres	Personas	2.525
...	...	...	...	...	...	...
268351	Apátrida	Total Nacional	51001 Ceuta	Hombres	Personas	0
268352	Apátrida	Total Nacional	51001 Ceuta	Mujeres	Personas	0
268353	Apátrida	Total Nacional	52001 Melilla	Ambos sexos	Personas	1
268354	Apátrida	Total Nacional	52001 Melilla	Hombres	Personas	1
268355	Apátrida	Total Nacional	52001 Melilla	Mujeres	Personas	0

4 Standardization

4.1 Postal Codes csv

Use the CSV named “ETL_project-postal_codes.csv” and clean it. Pay attention to the final result that should be a table with the F ciudad, provincia_local (with the local name), ciudad_local (with the local language name)).

In [7]:

```
ExtracionCSV1.head(10)
```

Out[7]:

	provincia	poblacion	Código Postal
0	Araba/Álava	Alegría-Dulantzi	240
1	Ávila	Candeleda	548
2	Araba/Álava	Vitoria-Gasteiz	1001
3	Araba/Álava	Vitoria-Gasteiz	1002
4	Araba/Álava	Vitoria-Gasteiz	1003
5	Araba/Álava	Vitoria-Gasteiz	1004
6	Araba/Álava	Vitoria-Gasteiz	1005
7	Araba/Álava	Vitoria-Gasteiz	1006
8	Araba/Álava	Vitoria-Gasteiz	1007
9	Araba/Álava	Vitoria-Gasteiz	1008

4.1.1 Lists with provinces in Spanish and local language

In [8]:

```
# getting the unique values of the column provincia

ExtracionCSV1["provincia"].unique()
```

Out[8]:

```
array(['Araba/Álava', 'Ávila', 'Burgos', 'Albacete', 'Gipuzkoa', 'Huelva',
       'Murcia', 'Cuenca', 'Alicante/Alacant', 'Asturias', 'Almería',
       'Badajoz', 'Illes Balears', 'Barcelona', 'Lleida', 'Cáceres',
       'Sevilla', 'Cádiz', 'Castellón/Castelló', 'Ciudad Real', 'Córdoba',
       'A Coru?a', 'Granada', 'Guadalajara', 'Girona', 'León', 'Huesca',
       'Zaragoza', 'Jaén', 'La Rioja', 'Lugo', 'Madrid', 'Málaga',
       'Navarra', 'Ourense', 'Palencia', 'Las Palmas', 'Pontevedra',
       'Salamanca', 'Santa Cruz de Tenerife', 'Cantabria', 'Segovia',
       'Soria', 'Tarragona', 'Valencia/Val?ncia', 'Teruel', 'Toledo',
       'Valladolid', 'Bizkaia', 'Zamora', 'Ceuta', 'Melilla'],
      dtype=object)
```

In [9]:

```
# Filling out lists with the names of the provinces

provincia_castellano = []
provincia_idlocal = []

for row in ExtracionCSV1.iterrows():

    # splits the content of the cell by the / delimiter

    provincia_raw = row[1][0].split("/")

    # if there's only 1 item in the resulting element, fills both lists with the same element.

    if len(provincia_raw) == 1:

        provincia_raw.append(provincia_raw[0])

    # adds each element to the corresponding list

    provincia_castellano.append(provincia_raw[0])
    provincia_idlocal.append(provincia_raw[1])
```

In [10]:

```
# sanity check

provincia_castellano
```

```
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
 'Alicante',
```

In [11]:

```
# sanity check

provincia_idlocal
```

Out[11]:

```
['Álava',
 'Ávila',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
```

In [12]:

```
# Araba and Alava are in the wrong lists, so we replace them.

for i in range(0, len(provincia_castellano)):

    if provincia_castellano[i] == "Araba":

        provincia_castellano[i] = "Álava"

for i in range(0, len(provincia_idlocal)):

    if provincia_idlocal[i] == "Álava":

        provincia_idlocal[i] = "Araba"
```

In [13]:

```
provincia_castellano
```

Out[13]:

```
['Álava',
 'Ávila',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
 'Álava',
```

In [14]:

```
provincia_idlocal
```

```
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
 'Araba',
```

4.1.2 Lists with cities in Spanish and local language

In [15]:

```
ciudad_castellano = []
ciudad_idlocal = []

# Filling out lists with the names of the cities. Everything else is the same as before

for row in ExtracionCSV1.iterrows():

    ciudad_raw = row[1][1].split("/")

    if len(ciudad_raw) == 1:

        ciudad_raw.append(ciudad_raw[0])

    ciudad_castellano.append(ciudad_raw[0])
    ciudad_idlocal.append(ciudad_raw[1])
```

In [16]:

```
ciudad_castellano
```

Out[16]:

```
['Alegría-Dulantzi',
 'Candeleda',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Labastida',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Campezo',
 'Campezo',
```

In [17]:

```
ciudad_idlocal
```

Out[17]:

```
['Alegría-Dulantzi',
 'Candeleda',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Bastida',
 'Vitoria-Gasteiz',
 'Vitoria-Gasteiz',
 'Kanpezu',
 'Kanpezu',
```

4.1.3 Lists with postal code and index

In [18]:

```
# We create lists for the remaining columns
```

In [19]:

```
cod_postal = list(ExtracionCSV1["Código Postal"])
```

In [20]:

```
cod_postal
```

```
 1004,
 1005,
 1006,
 1007,
 1008,
 1009,
 1010,
 1012,
 1012,
 1013,
 1015,
 1110,
 1117,
 1117,
 1118,
 1118,
 1118,
 1120,
 1128,
```

In [21]:

```
index_ej1 = list(ExtracionCSV1.index)
```

In [22]:

```
index_ej1
```

Out[22]:

```
[0,
 1,
 2,
 3,
 4,
 5,
 6,
 7,
 8,
 9,
 10,
 11,
 12,
 13,
 14,
 15,
 16,
 17.
```

4.1.4 Creation of the standardized dataframe

columns:

> id,
> cp,
> provincia,
> ciudad,
> provincia_local (with the local name),
> ciudad_local (with the local language name)

In [23]:

```
# merging all the lists into a single dataframe

standardized_postal_codes = pd.DataFrame(zip(index_ej1,
                                             cod_postal,
                                             provincia_castellano,
                                             ciudad_castellano,
                                             provincia_idlocal,
                                             ciudad_idlocal),

                                         columns = ["id", "cp", "provincia", "ciudad",
                                                    "provincia_local", "ciudad_local"]
                                        )

standardized_postal_codes = standardized_postal_codes.set_index("id")
```

In [24]:

```
standardized_postal_codes
```

	cp	provincia	ciudad	provincia_local	ciudad_local
id					
0	240	Álava	Alegría-Dulantzi	Araba	Alegría-Dulantzi
1	548	Ávila	Candeleda	Ávila	Candeleda
2	1001	Álava	Vitoria-Gasteiz	Araba	Vitoria-Gasteiz
3	1002	Álava	Vitoria-Gasteiz	Araba	Vitoria-Gasteiz
4	1003	Álava	Vitoria-Gasteiz	Araba	Vitoria-Gasteiz
...	...	...	...	...	...
14660	52004	Melilla	Melilla	Melilla	Melilla
14661	52005	Melilla	Melilla	Melilla	Melilla
14662	52006	Melilla	Melilla	Melilla	Melilla
14663	73820	Tarragona	Calafell	Tarragona	Calafell
14664	90007	Zaragoza	Zaragoza	Zaragoza	Zaragoza

4.1.5 Creation CSV

In [25]:

```python
# saving the dataframe in the folder STANDARISED

with open('STANDARISED/Postal_Codes.csv', "w+") as f:
    standardized_postal_codes.to_csv(f)
```

4.2 Data INE Girona

They want to focus on Girona, so it's necessary to get the population of all the cities in the Region. This data text = is publicly ava (https://servicios.ine.es/wstempus/js/es/DATOS_TABLA/33791?tip=AM):
You must use the JSON format Datasource. You need to create a table containing the following columns:

> id,
> poblacion,
> origen,
> and a column for each year of data.

In [26]:

```python
# opening the file

with open("RAW/Girona.json") as f:
    girona_data = json.load(f)
```

In [27]:

```python
girona_data
```

```
 'Nombre': 'Dato base. Total. Girona. Total. ',
 'T3_Unidad': 'Personas',
 'T3_Escala': ' ',
 'MetaData': [{'Id': 72,
   'Variable': {'Id': 3, 'Nombre': 'Tipo de dato', 'Codigo': ''},
   'Nombre': 'Dato base',
   'Codigo': ''},
  {'Id': 451,
   'Variable': {'Id': 18, 'Nombre': 'Sexo', 'Codigo': ''},
   'Nombre': 'Total',
   'Codigo': ''},
  {'Id': 18,
   'Variable': {'Id': 115, 'Nombre': 'Provincias', 'Codigo': 'PROV'},
   'Nombre': 'Girona',
   'Codigo': '17'},
  {'Id': 16420,
   'Variable': {'Id': 431, 'Nombre': 'Países y Continentes', 'Codigo': ''},
   'Nombre': 'Total',
   'Codigo': ''}],
 'Data': [{'Fecha': '2022-01-01T00:00:00.000+01:00',
```

In [28]:

```python
# these are the keys that we're intersted in

keys_girona = ['COD', 'MetaData', 'Data']
```

In [29]:

```python
# loop to better see the contents of the file

for index in range(len(girona_data)):

    print(f"elem{index} in girona_data:", "\n", sep = "")

    for key in keys_girona:

        print(f"key {key} contains:", "\n",
              girona_data[index][key], "\n", sep = "")
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

Contents

In [30]:

```python
# identifying where nationalities are stored in the data

for index in range(1000):

    if girona_data[index]['MetaData'][1]['Variable']['Codigo'] == 'MUN':

        print(girona_data[index]['MetaData'][2]['Nombre'])
```

```
Total
España
Extranjero
Europa (sin España)
UE28 sin España
Alemania
Bulgaria
Francia
Italia
Polonia
Portugal
Reino Unido
Rumanía
Europa menos UE28
Rusia
Ucrania
África
Argelia
Marruecos
```

In [31]:

```python
girona_std_list = []

# Bucle que coge los índices de los elementos que tengan dentro del key MetaData el Código = MUN

for index in range(len(girona_data)):

    if girona_data[index]['MetaData'][1]['Variable']['Codigo'] == 'MUN':

        municipio = girona_data[index]['MetaData'][1]['Nombre']

        id_COD = girona_data[index]['COD']

        origen = girona_data[index]['MetaData'][2]['Nombre']

        fila = [id_COD, municipio, origen]

        for i in range(len(girona_data[index]['Data'])):

            n_habitantes = int(girona_data[index]['Data'][i]['Valor'])

            fila.append(n_habitantes)

        girona_std_list.append(fila)
```

Contents

In [32]:

```
# creating a pandas dataframe

girona_std = pd.DataFrame(girona_std_list, columns = ["id", "Municipio", "Nacionalidad", "2022", "2

girona_std
```

Out[32]:

	id	Municipio	Nacionalidad	2022	2021	2020	2019	2018	2017	2016	...	2012	2011	2010	2009	2008
0	PC5662094	Agullana	Total	903	885	863.0	831.0	841.0	831.0	841.0	...	858.0	839.0	840.0	812.0	806.0
1	PC5662093	Agullana	España	735	711	704.0	689.0	697.0	692.0	704.0	...	678.0	657.0	651.0	645.0	648.0
2	PC5662092	Agullana	Extranjero	168	174	159.0	142.0	144.0	139.0	137.0	...	180.0	182.0	189.0	167.0	158.0
3	PC5662063	Agullana	Europa (sin España)	71	79	67.0	56.0	57.0	53.0	55.0	...	94.0	103.0	103.0	92.0	84.0
4	PC5662062	Agullana	UE28 sin España	57	50	54.0	49.0	51.0	59.0	63.0	...	96.0	87.0	79.0	56.0	48.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26515	PC4743757	Vilopriu	China	0	0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0
26516	PC4743756	Vilopriu	Pakistán	0	0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
26517	PC22759807	Vilopriu	UE27_2020 sin España	10	9	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
26518	PC4743749	Vilopriu	Oceanía	0	0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
26519	PC22759806	Vilopriu	Europa menos UE27_2020	1	1	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN

26520 rows × 23 columns

In [33]:

```
# saving the dataframe in a folder

with open('STANDARISED/Girona.csv', "w+") as f:
    girona_std.to_csv(f)
```

4.3 Data INE Censo Municipal

In [34]:

```
# opening the file

censo_municipal_data = pd.read_csv("RAW/55244 (1).csv", sep= ";", low_memory= False)

censo_municipal_data
```

Out[34]:

	Nacionalidad (grandes grupos)	Total Nacional	Municipios	Sexo	Unidades de medida	Total
0	TOTAL	Total Nacional	NaN	Ambos sexos	Personas	47.400.798
1	TOTAL	Total Nacional	NaN	Hombres	Personas	23.248.611
2	TOTAL	Total Nacional	NaN	Mujeres	Personas	24.152.187
3	TOTAL	Total Nacional	01051 Agurain/Salvatierra	Ambos sexos	Personas	5.022
4	TOTAL	Total Nacional	01051 Agurain/Salvatierra	Hombres	Personas	2.525
...	...	...	...	...	...	...
268351	Apátrida	Total Nacional	51001 Ceuta	Hombres	Personas	0
268352	Apátrida	Total Nacional	51001 Ceuta	Mujeres	Personas	0
268353	Apátrida	Total Nacional	52001 Melilla	Ambos sexos	Personas	1
268354	Apátrida	Total Nacional	52001 Melilla	Hombres	Personas	1
268355	Apátrida	Total Nacional	52001 Melilla	Mujeres	Personas	0

268356 rows × 6 columns

In [35]:

```
# loop to remove the postal codes from the dataframe

for i in range(len(censo_municipal_data.Municipios.values)):

    if type(censo_municipal_data.Municipios.values[i]) !=  float:

        censo_municipal_data.Municipios.values[i] = censo_municipal_data.Municipios.values[i].split
```

Contents

In [36]:

```python
# store the data in a csv file

censo_municipal_data[['Nacionalidad (grandes grupos)', 'Municipios', 'Sexo', 'Total']]

with open('STANDARISED/Datos_Censo.csv', "w+") as f:
    censo_municipal_data[['Nacionalidad (grandes grupos)', 'Municipios', 'Sexo', 'Total']].to_csv(f
```

5 Trusted

5.1 Postal Codes csv

In [37]:

```python
# opening file

postal_codes = pd.read_csv('STANDARISED/Postal_Codes.csv', encoding= "latin1")

postal_codes
```

Out[37]:

	id	cp	provincia	ciudad	provincia_local	ciudad_local
0	0	240	Álava	Alegría-Dulantzi	Araba	Alegría-Dulantzi
1	1	548	Ávila	Candeleda	Ávila	Candeleda
2	2	1001	Álava	Vitoria-Gasteiz	Araba	Vitoria-Gasteiz
3	3	1002	Álava	Vitoria-Gasteiz	Araba	Vitoria-Gasteiz
4	4	1003	Álava	Vitoria-Gasteiz	Araba	Vitoria-Gasteiz
...	...	...	...	...	...	...
14660	14660	52004	Melilla	Melilla	Melilla	Melilla
14661	14661	52005	Melilla	Melilla	Melilla	Melilla
14662	14662	52006	Melilla	Melilla	Melilla	Melilla
14663	14663	73820	Tarragona	Calafell	Tarragona	Calafell
14664	14664	90007	Zaragoza	Zaragoza	Zaragoza	Zaragoza

14665 rows × 6 columns

In [38]:

```python
# getting unique values for the column provincia

postal_codes.provincia.unique()
```

Out[38]:

```
array(['Álava', 'Ávila', 'Burgos', 'Albacete', 'Gipuzkoa', 'Huelva',
       'Murcia', 'Cuenca', 'Alicante', 'Asturias', 'Almería', 'Badajoz',
       'Illes Balears', 'Barcelona', 'Lleida', 'Cáceres', 'Sevilla',
       'Cádiz', 'Castellón', 'Ciudad Real', 'Córdoba', 'A Coru?a',
       'Granada', 'Guadalajara', 'Girona', 'León', 'Huesca', 'Zaragoza',
       'Jaén', 'La Rioja', 'Lugo', 'Madrid', 'Málaga', 'Navarra',
       'Ourense', 'Palencia', 'Las Palmas', 'Pontevedra', 'Salamanca',
       'Santa Cruz de Tenerife', 'Cantabria', 'Segovia', 'Soria',
       'Tarragona', 'Valencia', 'Teruel', 'Toledo', 'Valladolid',
       'Bizkaia', 'Zamora', 'Ceuta', 'Melilla'], dtype=object)
```

In [39]:

```
# replacing the values with spelling mistakes

for i in range(len(postal_codes.provincia.values)):

    if postal_codes.provincia[i] == "A Coru?a":

        postal_codes.provincia[i] = "A Coruña"

        postal_codes.provincia_local[i] = "A Coruña"
```

```
C:\Users\Ruben\AppData\Local\Temp\ipykernel_7192\4089188941.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
y)
  postal_codes.provincia[i] = "A Coruña"
C:\Users\Ruben\AppData\Local\Temp\ipykernel_7192\4089188941.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
y)
  postal_codes.provincia_local[i] = "A Coruña"
```

In [40]:

```
# checking whether the values have been changed or not

postal_codes.provincia_local.unique()
```

Out[40]:

```
array(['Araba', 'Ávila', 'Burgos', 'Albacete', 'Gipuzkoa', 'Huelva',
       'Murcia', 'Cuenca', 'Alacant', 'Asturias', 'Almería', 'Badajoz',
       'Illes Balears', 'Barcelona', 'Lleida', 'Cáceres', 'Sevilla',
       'Cádiz', 'Castelló', 'Ciudad Real', 'Córdoba', 'A Coruña',
       'Granada', 'Guadalajara', 'Girona', 'León', 'Huesca', 'Zaragoza',
       'Jaén', 'La Rioja', 'Lugo', 'Madrid', 'Málaga', 'Navarra',
       'Ourense', 'Palencia', 'Las Palmas', 'Pontevedra', 'Salamanca',
       'Santa Cruz de Tenerife', 'Cantabria', 'Segovia', 'Soria',
       'Tarragona', 'Val?ncia', 'Teruel', 'Toledo', 'Valladolid',
       'Bizkaia', 'Zamora', 'Ceuta', 'Melilla'], dtype=object)
```

In [41]:

```
# replacing the values with spelling mistakes in the other column

for i in range(len(postal_codes.provincia_local.values)):

    if postal_codes.provincia_local[i] == 'Val?ncia':

        postal_codes.provincia_local[i] = "València"
```

```
C:\Users\Ruben\AppData\Local\Temp\ipykernel_7192\1503452087.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
y)
  postal_codes.provincia_local[i] = "València"
```

In [42]:

```
# checking whether the values have been changed or not

postal_codes.provincia_local.unique()
```

Out[42]:

```
array(['Araba', 'Ávila', 'Burgos', 'Albacete', 'Gipuzkoa', 'Huelva',
       'Murcia', 'Cuenca', 'Alacant', 'Asturias', 'Almería', 'Badajoz',
       'Illes Balears', 'Barcelona', 'Lleida', 'Cáceres', 'Sevilla',
       'Cádiz', 'Castelló', 'Ciudad Real', 'Córdoba', 'A Coruña',
       'Granada', 'Guadalajara', 'Girona', 'León', 'Huesca', 'Zaragoza',
       'Jaén', 'La Rioja', 'Lugo', 'Madrid', 'Málaga', 'Navarra',
       'Ourense', 'Palencia', 'Las Palmas', 'Pontevedra', 'Salamanca',
       'Santa Cruz de Tenerife', 'Cantabria', 'Segovia', 'Soria',
       'Tarragona', 'València', 'Teruel', 'Toledo', 'Valladolid',
       'Bizkaia', 'Zamora', 'Ceuta', 'Melilla'], dtype=object)
```

In [43]:

```
# storing the dataframe in the file

with pd.ExcelWriter("TRUSTED/postal_codes_trusted.xlsx") as f:
    postal_codes.to_excel(f, sheet_name = "postal_codes_trusted")
```

5.2 Girona csv

In [44]:

```
# opening file

girona_std = pd.read_csv("STANDARISED/Girona.csv", encoding = "latin1", index_col=[0])
```

In [45]:

```
# replacing nas with 0

girona_trusted = girona_std.fillna(0)
```

In [46]:

```
# creating a list with the column names that need to be changed

cambiar_a_int = list(girona_trusted.columns)
```

In [47]:

```
cambiar_a_int[3:]
```

Out[47]:

```
['2022',
 '2021',
 '2020',
 '2019',
 '2018',
 '2017',
 '2016',
 '2015',
 '2014',
 '2013',
 '2012',
 '2011',
 '2010',
 '2009',
 '2008',
 '2007',
 '2006',
 '2005',
 '2004',
 '2003']
```

In [48]:

```
# changing data type to int

for i in cambiar_a_int[3:]:

    girona_trusted[i] = girona_trusted[i].astype("int64", copy = True, errors = "raise")
```

In [49]:

```
# checking whether the changes have taken effect

girona_trusted
```

1	PC5662093	Agullana	España	735	711	704	689	697	692	704	...	678	657	651	645	648	645
2	PC5662092	Agullana	Extranjero	168	174	159	142	144	139	137	...	180	182	189	167	158	108
3	PC5662063	Agullana	Europa (sin España)	71	79	67	56	57	53	55	...	94	103	103	92	84	59
4	PC5662062	Agullana	UE28 sin España	57	50	54	49	51	59	63	...	96	87	79	56	48	49
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
26515	PC4743757	Vilopriu	China	0	0	0	0	0	0	0	...	0	0	0	0	1	0
26516	PC4743756	Vilopriu	Pakistán	0	0	0	0	0	0	0	...	0	0	0	0	0	0
26517	PC22759807	Vilopriu	UE27_2020 sin España	10	9	0	0	0	0	0	...	0	0	0	0	0	0
26518	PC4743749	Vilopriu	Oceanía	0	0	0	0	0	0	0	...	0	0	0	0	0	0
26519	PC22759806	Vilopriu	Europa menos UE27_2020	1	1	0	0	0	0	0	...	0	0	0	0	0	0

In [50]:

```
# storing the cleaned up data in a xlsx file as requested

with pd.ExcelWriter("TRUSTED/girona_trusted.xlsx") as f:
    girona_trusted.to_excel(f, sheet_name = "girona_trusted")
```

5.3 Datos_censo csv

In [51]:

```
# opening the file

datos_censo_trusted = pd.read_csv("STANDARISED/Datos_Censo.csv",
                                  encoding = "latin1", index_col=[0],
                                 low_memory = False)
```

In [52]:

```
# checking which columns have null values

datos_censo_trusted.isna().sum()
```

Out[52]:

```
Nacionalidad (grandes grupos)     0
Municipios                       33
Sexo                              0
Total                             0
dtype: int64
```

Contents

In [53]:

```
# getting the rows with Nulls to understand what they mean

datos_censo_trusted[datos_censo_trusted['Municipios'].isna()]
```

Out[53]:

	Nacionalidad (grandes grupos)	Municipios	Sexo	Total
0	TOTAL	NaN	Ambos sexos	47.400.798
1	TOTAL	NaN	Hombres	23.248.611
2	TOTAL	NaN	Mujeres	24.152.187
24396	Española	NaN	Ambos sexos	41.998.096
24397	Española	NaN	Hombres	20.534.537
24398	Española	NaN	Mujeres	21.463.559
48792	Unión Europea (sin España)	NaN	Ambos sexos	1.627.751
48793	Unión Europea (sin España)	NaN	Hombres	814.031
48794	Unión Europea (sin España)	NaN	Mujeres	813.720
73188	Resto de Europa	NaN	Ambos sexos	565.378
73189	Resto de Europa	NaN	Hombres	262.489
73190	Resto de Europa	NaN	Mujeres	302.889
97584	África	NaN	Ambos sexos	1.179.963
97585	África	NaN	Hombres	704.919
97586	África	NaN	Mujeres	475.044
121980	América del Norte	NaN	Ambos sexos	69.029
121981	América del Norte	NaN	Hombres	29.846
121982	América del Norte	NaN	Mujeres	39.183
146376	Centro América y Caribe	NaN	Ambos sexos	356.411
146377	Centro América y Caribe	NaN	Hombres	133.671
146378	Centro América y Caribe	NaN	Mujeres	222.740
170772	Sudamérica	NaN	Ambos sexos	1.115.107
170773	Sudamérica	NaN	Hombres	497.715
170774	Sudamérica	NaN	Mujeres	617.392
195168	Asia	NaN	Ambos sexos	482.413
195169	Asia	NaN	Hombres	267.599
195170	Asia	NaN	Mujeres	214.814
219564	Oceanía	NaN	Ambos sexos	3.538
219565	Oceanía	NaN	Hombres	1.922
219566	Oceanía	NaN	Mujeres	1.616
243960	Apátrida	NaN	Ambos sexos	3.112
243961	Apátrida	NaN	Hombres	1.882
243962	Apátrida	NaN	Mujeres	1.230

In [54]:

```
# removing nulls

datos_censo_trusted.dropna(inplace = True)
```

In [55]:

```
datos_censo_trusted
```

Out[55]:

	Nacionalidad (grandes grupos)	Municipios	Sexo	Total
3	TOTAL	Agurain/Salvatierra	Ambos sexos	5.022
4	TOTAL	Agurain/Salvatierra	Hombres	2.525
5	TOTAL	Agurain/Salvatierra	Mujeres	2.497
6	TOTAL	Alegría-Dulantzi	Ambos sexos	2.924
7	TOTAL	Alegría-Dulantzi	Hombres	1.518
...	...	...	...	...
268351	Apátrida	Ceuta	Hombres	0
268352	Apátrida	Ceuta	Mujeres	0
268353	Apátrida	Melilla	Ambos sexos	1
268354	Apátrida	Melilla	Hombres	1

In [56]:

```
# storing dataframe in a xlsx file as requested

with pd.ExcelWriter("TRUSTED/datos_censo_trusted.xlsx") as f:
    datos_censo_trusted.to_excel(f, sheet_name = "datos_censo_trusted")
```