Second Exercise: Wines

In this exercise, We work with data about wines and We want to get some information from that data.

Load both files, the .csv as pandas dataframe and the .json as json

The data is stored in two datasets contained in two files.

One is a json file and the other is a csv file.

As We want to get some information about this data, We proceed to create one dataframe for each datasets. To do it we load the json file and obtain a dataframe and We read the csv as dataframe

In [1]:
```
#We import all the modules that We will need to solve the tasks

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import json
```

In [2]:
```
# We open the JSON file using the open() function
with open('Files/winemag-data-130k-v2.json') as f:
    #We load the content of the file
    data = json.load(f)

# We convert it in a dataframe using pandas.DataFrame() function
df = pd.DataFrame(data)
df
```

Out[2]:

	points	title	description	taster_name	taster_twitter_handle	price	designation	variety	region_1	regi
0	87	Nicosia 2013 Vulkà Bianco (Etna)	Aromas include tropical fruit, broom, brimston...	Kerin O'Keefe	@kerinokeefe	NaN	Vulkà Bianco	White Blend	Etna	
1	87	Quinta dos Avidagos 2011 Avidagos Red (Douro)	This is ripe and fruity, a wine that is smooth...	Roger Voss	@vossroger	15.0	Avidagos	Portuguese Red	None	
2	87	Rainstorm 2013 Pinot Gris (Willamette Valley)	Tart and snappy, the flavors of lime flesh and...	Paul Gregutt	@paulgwine	14.0	None	Pinot Gris	Willamette Valley	Willar V
3	87	St. Julian 2013 Reserve Late Harvest Riesling ...	Pineapple rind, lemon pith and orange blossom ...	Alexander Peartree	None	13.0	Reserve Late Harvest	Riesling	Lake Michigan Shore	
4	87	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Much like the regular bottling from 2012, this...	Paul Gregutt	@paulgwine	65.0	Vintner's Reserve Wild Child Block	Pinot Noir	Willamette Valley	Willar V
...	...	...	...	...	...	...	...	...	...	
129966	90	Dr. H. Thanisch (Erben Müller-Burggraef) 2013 ...	Notes of honeysuckle and cantaloupe sweeten th...	Anna Lee C. Iijima	None	28.0	Brauneberger Juffer-Sonnenuhr Spätlese	Riesling	None	
129967	90	Citation 2004 Pinot Noir (Oregon)	Citation is given as much as a decade of bottl...	Paul Gregutt	@paulgwine	75.0	None	Pinot Noir	Oregon	Or (
129968	90	Domaine Gresser 2013 Kritt Gewurztraminer (Als...	Well-drained gravel soil gives this wine its c...	Roger Voss	@vossroger	30.0	Kritt	Gewürztraminer	Alsace	
129969	90	Domaine Marcel Deiss 2012 Pinot Gris (Alsace)	A dry style of Pinot Gris, this is crisp with ...	Roger Voss	@vossroger	32.0	None	Pinot Gris	Alsace	
129970	90	Domaine Schoffit 2012 Lieu-dit Harth Cuvée Car...	Big, rich and off-dry, this is powered by inte...	Roger Voss	@vossroger	21.0	Lieu-dit Harth Cuvée Caroline	Gewürztraminer	Alsace	

129971 rows × 13 columns

In [3]:
```
#Don't forget to ask about what our client wants

#df = pd.read_json('Files2/winemag-data-130k-v2.json')
#df
```

In [4]:
```
df_2 = pd.read_csv('Files/winemag-data_first150k.csv')
df_2
```

Out[4]:

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	variety	winery
0	0	US	This tremendous 100% varietal wine hails from ...	Martha's Vineyard	96	235.0	California	Napa Valley	Napa	Cabernet Sauvignon	Heitz
1	1	Spain	Ripe aromas of fig, blackberry and cassis are ...	Carodorum Selección Especial Reserva	96	110.0	Northern Spain	Toro	NaN	Tinta de Toro	Bodega Carmen Rodríguez
2	2	US	Mac Watson honors the memory of a wine once ma...	Special Selected Late Harvest	96	90.0	California	Knights Valley	Sonoma	Sauvignon Blanc	Macauley
3	3	US	This spent 20 months in 30% new French oak, an...	Reserve	96	65.0	Oregon	Willamette Valley	Willamette Valley	Pinot Noir	Ponzi
4	4	France	This is the top wine from La Bégude, named aft...	La Brûlade	95	66.0	Provence	Bandol	NaN	Provence red blend	Domaine de la Bégude
...	...	...	...	...	...	...	...	...	...	...	...
150925	150925	Italy	Many people feel Fiano represents southern Ita...	NaN	91	20.0	Southern Italy	Fiano di Avellino	NaN	White Blend	Feudi di San Gregorio
150926	150926	France	Offers an intriguing nose with ginger, lime an...	Cuvée Prestige	91	27.0	Champagne	Champagne	NaN	Champagne Blend	H.Germain
150927	150927	Italy	This classic example comes from a cru vineyard...	Terre di Dora	91	20.0	Southern Italy	Fiano di Avellino	NaN	White Blend	Terredora
150928	150928	France	A perfect salmon shade, with scents of peaches...	Grand Brut Rosé	90	52.0	Champagne	Champagne	NaN	Champagne Blend	Gosset
150929	150929	Italy	More Pinot Grigios should taste like this. A r...	NaN	90	15.0	Northeastern Italy	Alto Adige	NaN	Pinot Grigio	Alois Lageder

150930 rows × 11 columns

Once We have both dataframes, We explore the columns of each dataframe

In [5]:
```
df.columns
```

Out[5]:
```
Index(['points', 'title', 'description', 'taster_name',
       'taster_twitter_handle', 'price', 'designation', 'variety', 'region_1',
       'region_2', 'province', 'country', 'winery'],
      dtype='object')
```

In [6]:
```
df_2.columns
```

Out[6]:
```
Index(['Unnamed: 0', 'country', 'description', 'designation', 'points',
       'price', 'province', 'region_1', 'region_2', 'variety', 'winery'],
      dtype='object')
```

Get a single dataframe that is the union of both files that only has the following columns: country, designation, points, price, state, winery.

We get just the columns that the task asks for from the resulting dataframe from the json

In [7]:
```
#List with all the columns that We want
chosen_columns = ['country','designation', 'points', 'price', 'province', 'winery']

#We select those wanted columns and make a copy() so We still have the original first dataframe untouched
df_3 = df.loc[: , chosen_columns].copy()

df_3
```

Out[7]:

	country	designation	points	price	province	winery
0	Italy	Vulkà Bianco	87	NaN	Sicily & Sardinia	Nicosia
1	Portugal	Avidagos	87	15.0	Douro	Quinta dos Avidagos
2	US	None	87	14.0	Oregon	Rainstorm
3	US	Reserve Late Harvest	87	13.0	Michigan	St. Julian
4	US	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Sweet Cheeks
...	...	...	...	...	...	...
129966	Germany	Brauneberger Juffer-Sonnenuhr Spätlese	90	28.0	Mosel	Dr. H. Thanisch (Erben Müller-Burggraef)
129967	US	None	90	75.0	Oregon	Citation
129968	France	Kritt	90	30.0	Alsace	Domaine Gresser
129969	France	None	90	32.0	Alsace	Domaine Marcel Deiss
129970	France	Lieu-dit Harth Cuvée Caroline	90	21.0	Alsace	Domaine Schoffit

129971 rows × 6 columns

We do the same with the second dataframe

In [8]:
```
#We select those wanted columns and make a copy() so We still have the original second dataframe untouched
df_4 = df_2.loc[: , chosen_columns].copy()

df_4
```

Out[8]:

	country	designation	points	price	province	winery
0	US	Martha's Vineyard	96	235.0	California	Heitz
1	Spain	Carodorum Selección Especial Reserva	96	110.0	Northern Spain	Bodega Carmen Rodríguez
2	US	Special Selected Late Harvest	96	90.0	California	Macauley
3	US	Reserve	96	65.0	Oregon	Ponzi
4	France	La Brûlade	95	66.0	Provence	Domaine de la Bégude
...	...	...	...	...	...	...
150925	Italy	NaN	91	20.0	Southern Italy	Feudi di San Gregorio
150926	France	Cuvée Prestige	91	27.0	Champagne	H.Germain
150927	Italy	Terre di Dora	91	20.0	Southern Italy	Terredora
150928	France	Grand Brut Rosé	90	52.0	Champagne	Gosset
150929	Italy	NaN	90	15.0	Northeastern Italy	Alois Lageder

150930 rows × 6 columns

In [9]:
```
#Finally We unify both dataframes.
df_final = pd.concat([df_3, df_4])
df_final
```

Out[9]:

	country	designation	points	price	province	winery
0	Italy	Vulkà Bianco	87	NaN	Sicily & Sardinia	Nicosia
1	Portugal	Avidagos	87	15.0	Douro	Quinta dos Avidagos
2	US	None	87	14.0	Oregon	Rainstorm
3	US	Reserve Late Harvest	87	13.0	Michigan	St. Julian
4	US	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Sweet Cheeks
...	...	...	...	...	...	...
150925	Italy	NaN	91	20.0	Southern Italy	Feudi di San Gregorio
150926	France	Cuvée Prestige	91	27.0	Champagne	H.Germain
150927	Italy	Terre di Dora	91	20.0	Southern Italy	Terredora
150928	France	Grand Brut Rosé	90	52.0	Champagne	Gosset
150929	Italy	NaN	90	15.0	Northeastern Italy	Alois Lageder

280901 rows × 6 columns

We compare if the number of rows of the final dataframe is equal to the rows of both previous dataframes

In [10]:
```
df_final.shape[0] == df_3.shape[0] + df_4.shape[0]
```

Out[10]:
```
True
```

Are there duplicates in the dataframe? Delete all duplicate rows

We see if there is duplicate rows

In [11]:
```
#This is the final version of this exercise

#This variable will be useful later to see if the elimination of duplicate rows is correct
n_rows_total = df_final.shape[0]

#Shows the duplicate rows of the file
duplicateRows = df_final[df_final.duplicated()]

#To see this more clearly We sort by the values in winery column
duplicateRows.sort_values(by="winery")
```

Out[11]:

	country	designation	points	price	province	winery
121401	US	NaN	88	NaN	Washington	'37 Cellars
133867	Spain	Brut	86	13.0	Catalonia	1+1=3
116517	Spain	Cabernet Sauvignon	82	18.0	Catalonia	1+1=3
141687	Spain	Brut	86	13.0	Catalonia	1+1=3
93375	Spain	Brut	87	16.0	Catalonia	1+1=3
...	...	...	...	...	...	...
15282	US	NaN	89	28.0	Washington	àMaurice
120292	US	NaN	92	34.0	Washington	àMaurice
81557	US	NaN	89	34.0	Washington	àMaurice
21901	US	Fred Estate	89	45.0	Washington	àMaurice
91274	Slovenia	Izbrani	88	20.0	Kras	Štoka

77782 rows × 6 columns

In [12]:
```
#Deleting duplicated rows
df_final.drop_duplicates(inplace=True)
df_final
```

Out[12]:

	country	designation	points	price	province	winery
0	Italy	Vulkà Bianco	87	NaN	Sicily & Sardinia	Nicosia
1	Portugal	Avidagos	87	15.0	Douro	Quinta dos Avidagos
2	US	None	87	14.0	Oregon	Rainstorm
3	US	Reserve Late Harvest	87	13.0	Michigan	St. Julian
4	US	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Sweet Cheeks
...	...	...	...	...	...	...
149633	US	NaN	84	40.0	New York	Wölffer
149634	France	NaN	84	15.0	Alsace	W. Gisselbrecht
149635	US	Bungalow Red	84	15.0	California	Casa Barranca
149636	Portugal	30-year old tawny	84	NaN	Port	Casa Santa Eufemia
149638	Argentina	NaN	84	9.0	Mendoza Province	Finca El Portillo

203119 rows × 6 columns

In [13]:
```
#Is the number of rows in the dataframe with just the duplicate rows plus the number of rows without duplicates
#equal to the original dataframe number of rows?


n_rows_total == df_final.shape[0] + duplicateRows.shape[0]
```

Out[13]:
```
True
```

Make a study of the null values of the dataset, assign a zero to all the rows that have a null value in the price, after that, delete the rest of the null values of the dataframe. What is the resulting number of rows?

In [14]:
```
#We use a list because the criteria to order a list is Nan values, alpha uppercase from A-Z,
#alpha lowercase from a-z, and numbers from the littlest one to the biggest one
#If there is an string or a None value, sort() will raise an error
unique_prices = list(df_final.price.unique())
unique_prices.sort()
unique_prices
```

Out[14]:
```
[nan,
 4.0,
 5.0,
 6.0,
 7.0,
 8.0,
 9.0,
 10.0,
 11.0,
 12.0,
 13.0,
 14.0,
 15.0,
 16.0,
 17.0,
 18.0,
 19.0,
 20.0,
 21.0,
 22.0,
 23.0,
 24.0,
 25.0,
 26.0,
 27.0,
 28.0,
 29.0,
 30.0,
 31.0,
 32.0,
 33.0,
 34.0,
 35.0,
 36.0,
 37.0,
 38.0,
 39.0,
 40.0,
 41.0,
 42.0,
 43.0,
 44.0,
 45.0,
 46.0,
 47.0,
 48.0,
 49.0,
 50.0,
 51.0,
 52.0,
 53.0,
 54.0,
 55.0,
 56.0,
 57.0,
 58.0,
 59.0,
 60.0,
 61.0,
 62.0,
 63.0,
 64.0,
 65.0,
 66.0,
 67.0,
 68.0,
 69.0,
 70.0,
 71.0,
 72.0,
 73.0,
 74.0,
 75.0,
 76.0,
 77.0,
 78.0,
 79.0,
 80.0,
 81.0,
 82.0,
 83.0,
 84.0,
 85.0,
 86.0,
 87.0,
 88.0,
 89.0,
 90.0,
 91.0,
 92.0,
 93.0,
 94.0,
 95.0,
 96.0,
 97.0,
 98.0,
 99.0,
 100.0,
 101.0,
 102.0,
 103.0,
 104.0,
 105.0,
 106.0,
 107.0,
 108.0,
 109.0,
 110.0,
 111.0,
 112.0,
 113.0,
 114.0,
 115.0,
 116.0,
 117.0,
 118.0,
 119.0,
 120.0,
 121.0,
 122.0,
 123.0,
 124.0,
 125.0,
 126.0,
 127.0,
 128.0,
 129.0,
 130.0,
 131.0,
 132.0,
 133.0,
 134.0,
 135.0,
 136.0,
 137.0,
 138.0,
 139.0,
 140.0,
 141.0,
 142.0,
 143.0,
 144.0,
 145.0,
 146.0,
 147.0,
 148.0,
 149.0,
 150.0,
 151.0,
 152.0,
 153.0,
 154.0,
 155.0,
 156.0,
 157.0,
 158.0,
 159.0,
 160.0,
 161.0,
 162.0,
 163.0,
 164.0,
 165.0,
 166.0,
 167.0,
 168.0,
 169.0,
 170.0,
 171.0,
 172.0,
 173.0,
 174.0,
 175.0,
 176.0,
 177.0,
 178.0,
 179.0,
 180.0,
 181.0,
 182.0,
 183.0,
 184.0,
 185.0,
 187.0,
 188.0,
 189.0,
 190.0,
 191.0,
 193.0,
 194.0,
 195.0,
 196.0,
 197.0,
 198.0,
 199.0,
 200.0,
 202.0,
 203.0,
 204.0,
 205.0,
 206.0,
 207.0,
 208.0,
 209.0,
 210.0,
 211.0,
 212.0,
 214.0,
 215.0,
 216.0,
 217.0,
 219.0,
 220.0,
 222.0,
 224.0,
 225.0,
 226.0,
 227.0,
 228.0,
 229.0,
 230.0,
 231.0,
 234.0,
 235.0,
 236.0,
 237.0,
 238.0,
 239.0,
 240.0,
 242.0,
 243.0,
 245.0,
 246.0,
 247.0,
 248.0,
 249.0,
 250.0,
 252.0,
 253.0,
 255.0,
 256.0,
 257.0,
 258.0,
 259.0,
 260.0,
 261.0,
 262.0,
 265.0,
 266.0,
 268.0,
 269.0,
 270.0,
 271.0,
 272.0,
 273.0,
 275.0,
 276.0,
 279.0,
 280.0,
 281.0,
 282.0,
 283.0,
 285.0,
 286.0,
 288.0,
 289.0,
 290.0,
 292.0,
 293.0,
 294.0,
 295.0,
 299.0,
 300.0,
 301.0,
 303.0,
 304.0,
 305.0,
 306.0,
 307.0,
 310.0,
 312.0,
 314.0,
 315.0,
 316.0,
 317.0,
 318.0,
 319.0,
 320.0,
 322.0,
 323.0,
 325.0,
 328.0,
 330.0,
 333.0,
 335.0,
 345.0,
 349.0,
 350.0,
 351.0,
 353.0,
 354.0,
 355.0,
 357.0,
 359.0,
 360.0,
 365.0,
 367.0,
 369.0,
 370.0,
 375.0,
 376.0,
 380.0,
 385.0,
 388.0,
 390.0,
 391.0,
 395.0,
 398.0,
 399.0,
 400.0,
 410.0,
 411.0,
 412.0,
 415.0,
 419.0,
 420.0,
 425.0,
 426.0,
 428.0,
 430.0,
 440.0,
 445.0,
 448.0,
 450.0,
 451.0,
 455.0,
 460.0,
 463.0,
 467.0,
 468.0,
 469.0,
 470.0,
 474.0,
 475.0,
 476.0,
 480.0,
 486.0,
 495.0,
 496.0,
 499.0,
 500.0,
 505.0,
 510.0,
 520.0,
 525.0,
 530.0,
 535.0,
 540.0,
 544.0,
 545.0,
 550.0,
 560.0,
 569.0,
 574.0,
 575.0,
 580.0,
 588.0,
 595.0,
 596.0,
 599.0,
 600.0,
 602.0,
 612.0,
 617.0,
 625.0,
 630.0,
 639.0,
 650.0,
 660.0,
 670.0,
 672.0,
 675.0,
 685.0,
 698.0,
 710.0,
 740.0,
 750.0,
 757.0,
 764.0,
 767.0,
 770.0,
 775.0,
 780.0,
 790.0,
 800.0,
 820.0,
 848.0,
 850.0,
 886.0,
 900.0,
 932.0,
 973.0,
 980.0,
 1000.0,
 1100.0,
 1125.0,
 1200.0,
 1300.0,
 1400.0,
 1500.0,
 1900.0,
 2000.0,
 2013.0,
 2300.0,
 2500.0,
 3300.0]
```

That's how we can see that the only null value that there is in the price column is NaN value.

Knowing this We replace all NaN values with zeros:

In [15]:
```
df_final["price"] = df_final.price.replace(np.NaN, 0)
df_final
```

Out[15]:

	country	designation	points	price	province	winery
0	Italy	Vulkà Bianco	87	0.0	Sicily & Sardinia	Nicosia
1	Portugal	Avidagos	87	15.0	Douro	Quinta dos Avidagos
2	US	None	87	14.0	Oregon	Rainstorm
3	US	Reserve Late Harvest	87	13.0	Michigan	St. Julian
4	US	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Sweet Cheeks
...	...	...	...	...	...	...
149633	US	NaN	84	40.0	New York	Wölffer
149634	France	NaN	84	15.0	Alsace	W. Gisselbrecht
149635	US	Bungalow Red	84	15.0	California	Casa Barranca
149636	Portugal	30-year old tawny	84	0.0	Port	Casa Santa Eufemia
149638	Argentina	NaN	84	9.0	Mendoza Province	Finca El Portillo

203119 rows × 6 columns

Testing if there is NaN values in our dataframe

In [16]:
```
np.NaN in list(df_final.price.unique())
```

Out[16]:
```
False
```

In [17]:
```
for col in df_final.columns:
    if(col != "price"):
        print(df_final[col].dtype)
```

```
object
object
object
object
object
```

All other columns contain strings or non-numerical values (object pandas type).

We decided that We will just consider as null values the non-numerical values (None, NaN).

As None is consider a NaN value. We can use dropna to get rid of all rows with a Null value:

In [18]:
```
#Is None considered a NaN value?
pd.isna(None)
```

Out[18]:
```
True
```

In [19]:
```
#delete the rest of the null values of the dataframe
df_final.dropna(axis=0, inplace=True)
df_final
```

Out[19]:

	country	designation	points	price	province	winery
0	Italy	Vulkà Bianco	87	0.0	Sicily & Sardinia	Nicosia
1	Portugal	Avidagos	87	15.0	Douro	Quinta dos Avidagos
3	US	Reserve Late Harvest	87	13.0	Michigan	St. Julian
4	US	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Sweet Cheeks
5	Spain	Ars In Vitro	87	15.0	Northern Spain	Tandem
...	...	...	...	...	...	...
149629	US	Five-O	84	25.0	New York	Martha Clara
149630	US	Crimson Creek	84	27.0	California	Pine Ridge
149631	Portugal	40-year old tawny	84	130.0	Port	Poças
149635	US	Bungalow Red	84	15.0	California	Casa Barranca
149636	Portugal	30-year old tawny	84	0.0	Port	Casa Santa Eufemia

145455 rows × 6 columns

Obtain a subset with all the wines that are from Spain and that also include the word reserve 'reserva' in the designation of origin. Hint: To locate a text string in a column, you can use the str.contains function https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.str.contains.html

In [20]:
```
#Select the rows that has as country Spain and any designation
#with the word "reserva"
mask_reserva_spain =(df_final.country=='Spain') & (df_final.designation.str.contains(pat = "reserva", case = Fa

df_reserva = df_final[mask_reserva_spain].copy()
df_reserva
```

Out[20]:

	country	designation	points	price	province	winery
490	Spain	Reserva Brut	87	19.0	Catalonia	V&N; Cellars
604	Spain	Reserva 1423	87	35.0	Northern Spain	Príncipe de Viana
836	Spain	Cinta Púrpura Brut Reserva	86	13.0	Catalonia	Juvé y Camps
837	Spain	Sweet Reserva	86	15.0	Catalonia	Juvé y Camps
1053	Spain	Reserva	85	30.0	Northern Spain	Finca La Emperatriz
...	...	...	...	...	...	...
145885	Spain	Reserva	88	11.0	Central Spain	Castillo de Almansa
147286	Spain	Ceremonia Reserva De Autor	85	15.0	Levante	Vicente Gandia
147460	Spain	Hoya De Cadenas Reserva	85	9.0	Levante	Gandia
148030	Spain	Reserva	81	18.0	Northern Spain	Señorío de Sarria
148036	Spain	Reserva	81	20.0	Northern Spain	Viñalcasta

1485 rows × 6 columns

Of the previous sub-set with the Spanish reserve wines, do they have a higher average score than all the wines or a lower one?

In [21]:
```
df_final.dtypes
```

Out[21]:
```
country         object
designation     object
points          object
price          float64
province        object
winery          object
dtype: object
```

To be able to calculate the mean of the points column We have to change the data type to a numeric one

In [22]:
```
df_final.points = pd.to_numeric(df_final.points)
df_reserva.points = pd.to_numeric(df_reserva.points)
```

We get the mean points from all the world

In [23]:
```
#Gets the mean of the points with just 2 decimal places
mean_total=round(df_final.points.mean(),2)
mean_total
```

Out[23]:
```
88.55
```

We get the mean points from the Spanish wines with a "reserva" designation

In [24]:
```
mean_reserva_spain = round(df_reserva.points.mean(),2)
mean_reserva_spain
```

Out[24]:
```
88.22
```

In [25]:
```
print( "Are the Spanish Reserva wines better than the average of the world?: ",mean_total<mean_reserva_spain)
```

```
Are the Spanish Reserva wines better than the average of the world?:  False
```

The average from "reserva" Spanish wines is lower than the world average

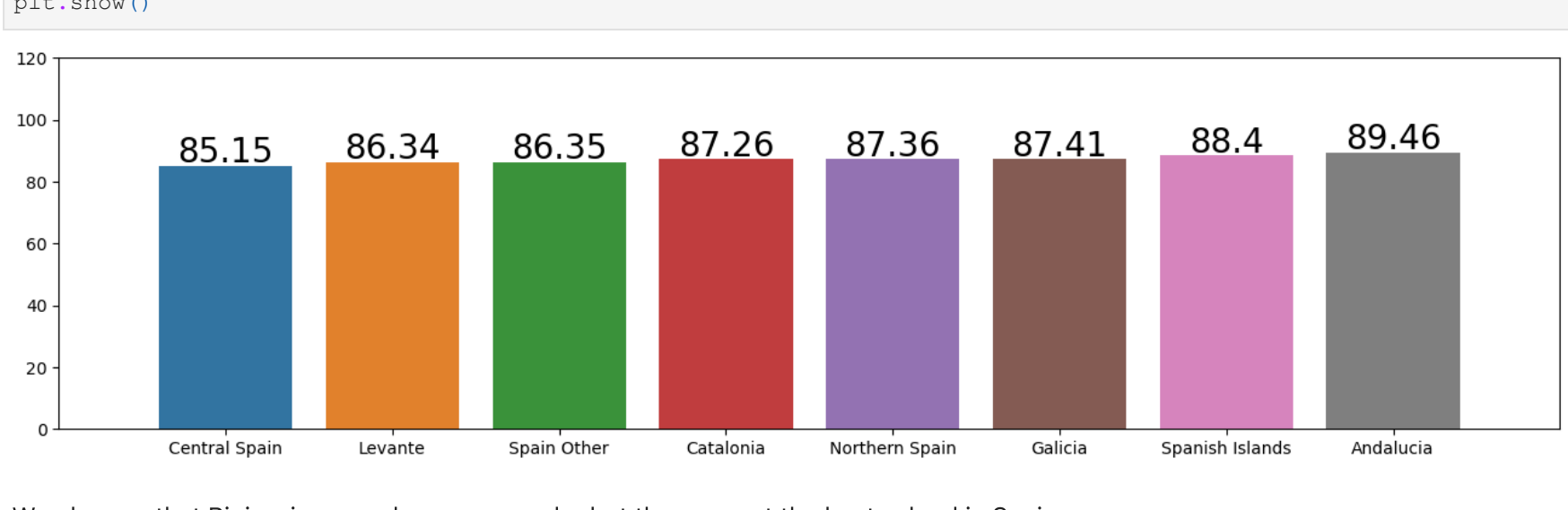
Take again the sub-set of Spanish reserve wines, in general, in Spain the wines from Rioja (a province in the north of Spain, North Spain will appear in the dataset) have great fame, therefore , it is to be expected that this province or region is the one with the best wines in this sub-set, check whether this statement is true graphically.

We obtain the mean for each province in Spain and represent the result so We can more easily view the results.

In [26]:
```
mean_points_by_province =df_final[df_final.country=='Spain'].groupby('province').points.mean()

mean_points_by_province_sorted = mean_points_by_province.sort_values()
mean_points_by_province_sorted
```

Out[26]:
```
province
Central Spain      85.145242
Levante            86.338104
Spain Other        86.346667
Catalonia          87.255867
Northern Spain     87.360326
Galicia            87.414201
Spanish Islands    88.400000
Andalucia          89.455479
Name: points, dtype: float64
```

In [27]:
```
#We obtain both arrays to be used as variables in the barplot
names_spain_province = np.array(mean_points_by_province_sorted.keys())
points_spain_province = np.array(mean_points_by_province_sorted)

#The values are rounded.
points_spain_province = np.around(points_spain_province , decimals=2)

#Determining the size of the figure
fig, ax = plt.subplots(1, 1, figsize=(16,4))

#A barplot is created
sns.barplot(x=names_spain_province, y=points_spain_province)

#The values of each bar are set just above them
for i in range(len(points_spain_province)):
    ax.text(i, points_spain_province[i]+1, points_spain_province[i], horizontalalignment='center', size=20)

#The limits of the axes are set for a better view
ax.set(xlim=(-1,len(points_spain_province)), ylim=(0,120))
plt.show()
```



We observe that Rioja wines can be more popular but they are not the best valued in Spain.

Show the top 10 of the best valued wines.

We get the 10 most valued wines:

In [28]:
```
#The top 10 of the best valued wines
df_top_10_wines = df_final.sort_values("points", ascending=False).head(10)
df_top_10_wines
```

Out[28]:

	country	designation	points	price	province	winery
111754	Italy	Cerretalto	100	270.0	Tuscany	Casanova di Neri
45781	Italy	Riserva	100	550.0	Tuscany	Biondi Santi
42197	Portugal	Barca-Velha	100	450.0	Douro	Casa Ferreirinha
114972	Portugal	Nacional Vintage	100	650.0	Port	Quinta do Noval
98647	US	Litton Estate Vineyard	100	100.0	California	Williams Selyem
89729	France	Le Mesnil Blanc de Blancs Brut	100	617.0	Champagne	Salon
89728	France	Cristal Vintage Brut	100	250.0	Champagne	Louis Roederer
24151	Italy	Masseto	100	460.0	Tuscany	Tenuta dell'Ornellaia
113929	US	Royal City	100	80.0	Washington	Charles Smith
39286	Italy	Masseto	100	460.0	Tuscany	Tenuta dell'Ornellaia

As We noticed there are more than 10 wines but the maxim calification (100 points), We decided to choose the ones with lower price, as an added good feature.

In [29]:
```
#We get a dataframe with all wines with maxim puntuation
#and calculate the number of rows
len(df_final.loc[df_final["points"]==100])
```

Out[29]:
```
22
```

In [30]:
```
#We use the previous dataframe and ordered it by price
#from low to high price
df_top_10_value_price=df_final.loc[df_final["points"]==100]
df_top_10_value_price=df_top_10_value_price.sort_values("price", ascending=True)
#And take the first 10 wines
df_top_10_value_price.head(10)
```

Out[30]:

	country	designation	points	price	province	winery
19354	US	En Chamberlin Vineyard	100	65.0	Oregon	Cayuse
113929	US	Royal City	100	80.0	Washington	Charles Smith
123545	US	Bionic Frog	100	80.0	Washington	Cayuse
98647	US	Litton Estate Vineyard	100	100.0	California	Williams Selyem
28954	Italy	Guado de' Gemoli	100	195.0	Tuscany	Giovanni Chiappini
7335	Italy	Occhio di Pernice	100	210.0	Tuscany	Avignonesi
111087	Italy	Occhio di Pernice	100	210.0	Tuscany	Avignonesi
92916	US	Hillside Select	100	215.0	California	Shafer
114272	US	Red Wine	100	245.0	California	Sloan
89728	France	Cristal Vintage Brut	100	250.0	Champagne	Louis Roederer

Using this criteria it is observed that Portugal doesn't make it to the top 10.

In [31]:
```
#The dessignation and winery are shown alone
df_top_10_value_price.loc[:,["winery", "designation"]].head(10)
```

Out[31]:

	winery	designation
19354	Cayuse	En Chamberlin Vineyard
113929	Charles Smith	Royal City
123545	Cayuse	Bionic Frog
98647	Williams Selyem	Litton Estate Vineyard
28954	Giovanni Chiappini	Guado de' Gemoli
7335	Avignonesi	Occhio di Pernice
111087	Avignonesi	Occhio di Pernice
92916	Shafer	Hillside Select
114272	Sloan	Red Wine
89728	Louis Roederer	Cristal Vintage Brut

Show the average price of wines in each country. Which country has the highest average price? Does this country appear in the top 10 above?

In [32]:
```
df_final.groupby("country").price.mean().head()
```

Out[32]:
```
country
Argentina                 26.268494
Armenia                   14.500000
Australia                 38.229518
Austria                   27.653320
Bosnia and Herzegovina    13.000000
Name: price, dtype: float64
```

We get the mean average price for each country. We order from higher to lower average price. And finally We get the first 5 country with the highest average price.

In [33]:
```
#Get average price for each country
mean_price_by_country= df_final.groupby("country").price.mean()
#We sort that average price in descending order
mean_price_by_country_sorted = mean_price_by_country.sort_values(ascending=False)
#And with round the result with 2 decimal places
round(mean_price_by_country_sorted,2).head()
```

Out[33]:
```
country
Hungary    50.13
England    48.51
Germany    43.41
Canada     39.73
US         39.53
Name: price, dtype: float64
```

In [34]:
```
print("Country with the highest average price:", mean_price_by_country_sorted.keys()[0]," Average price: " ,rou
```

```
Country with the highest average price: Hungary  Average price:  50.13
```

To check if this country is in the top 10 list done above, a conditional is done. Is Hungary in the countries of the top 10 best wines?

In [35]:
```
mean_price_by_country_sorted.keys()[0] in df_top_10_wines.loc[:,["country"]]
```

Out[35]:
```
False
```

This means even wines in Hungary are more expensive in average there is no a top 10 wine. Maybe the average price is higher for other reasons. Maybe is just more expensive to make wine in those countries.

Graphically show the price over the score based on the province of the subset of Spanish reserve wines. Which province has the most expensive wine? Is the north of Spain still the province with the best valued reserve wine?

The dataframe showing the reserve wines in Spain is used again

In [36]:
```
df_reserva
```

Out[36]:

	country	designation	points	price	province	winery
490	Spain	Reserva Brut	87	19.0	Catalonia	V&N; Cellars
604	Spain	Reserva 1423	87	35.0	Northern Spain	Príncipe de Viana
836	Spain	Cinta Púrpura Brut Reserva	86	13.0	Catalonia	Juvé y Camps
837	Spain	Sweet Reserva	86	15.0	Catalonia	Juvé y Camps
1053	Spain	Reserva	85	30.0	Northern Spain	Finca La Emperatriz
...	...	...	...	...	...	...
145885	Spain	Reserva	88	11.0	Central Spain	Castillo de Almansa
147286	Spain	Ceremonia Reserva De Autor	85	15.0	Levante	Vicente Gandia
147460	Spain	Hoya De Cadenas Reserva	85	9.0	Levante	Gandia
148030	Spain	Reserva	81	18.0	Northern Spain	Señorío de Sarria
148036	Spain	Reserva	81	20.0	Northern Spain	Viñalcasta

1485 rows × 6 columns

In [37]:
```
#To show the mean of the points each province has
df_reserva.groupby("province").points.mean()
```

Out[37]:
```
province
Andalucia         91.800000
Catalonia         87.796407
Central Spain     85.000000
Levante           85.567568
Northern Spain    88.612225
Spain Other       89.000000
Name: points, dtype: float64
```

To create the arrray to get the column of points of the dataframe that will be used to plot:

In [38]:
```
points = np.array(df_reserva.groupby("province").points.mean())
points
```

Out[38]:
```
array([91.8       , 87.79640719, 85.        , 85.56756757, 88.61222541,
       89.        ])
```

To create the arrray to get the column of price of the dataframe that will be used to plot:

In [39]:
```
df_reserva.groupby("province").price.mean()
```

Out[39]:
```
province
Andalucia         115.800000
Catalonia          26.086826
Central Spain      14.883333
Levante            20.783784
Northern Spain     36.237822
Spain Other        24.000000
Name: price, dtype: float64
```

To create the arrray to get the column of provinces of the dataframe that will be used to plot.

In [40]:
```
price=np.array(df_reserva.groupby("province").price.mean())

province = np.array(df_reserva.groupby("province").price.mean().keys())
province
```

Out[40]:
```
array(['Andalucia', 'Catalonia', 'Central Spain', 'Levante',
       'Northern Spain', 'Spain Other'], dtype=object)
```

The dataframe to plot is created:

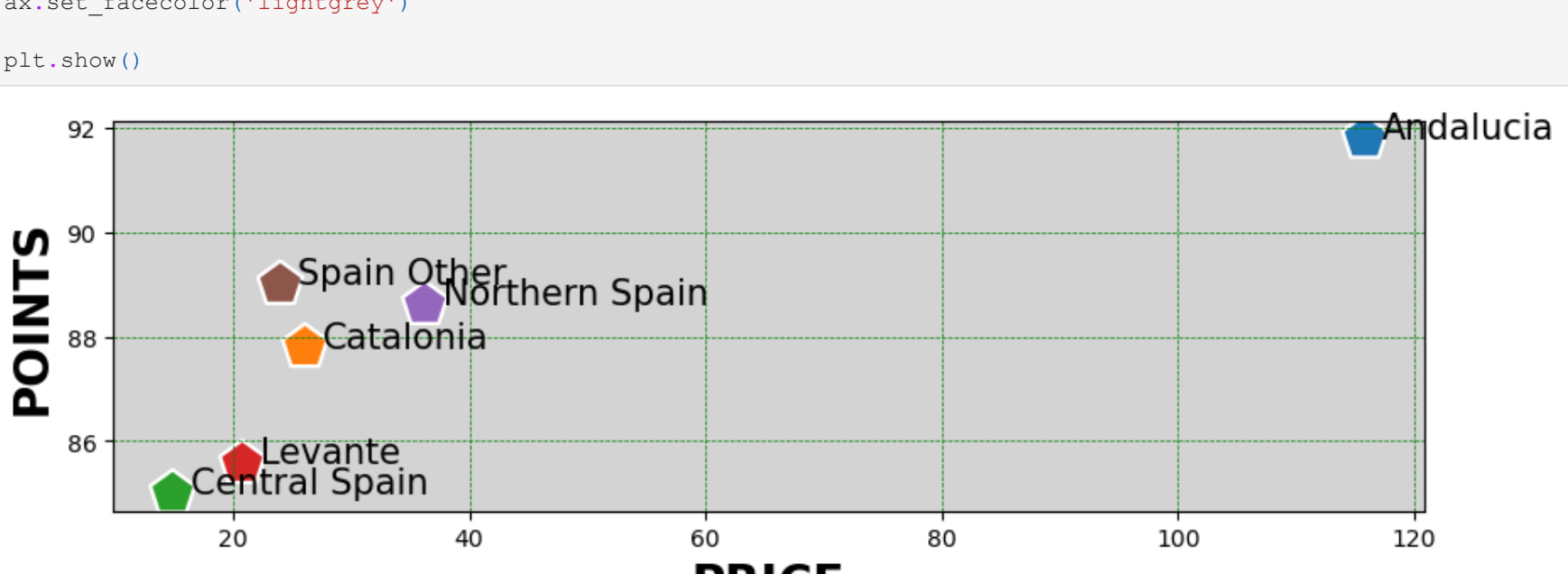
In [41]:
```
df_reserva_price_value =pd.DataFrame({"province": province, "price": price , "points" : points})
df_reserva_price_value
```

Out[41]:

	province	price	points
0	Andalucia	115.800000	91.800000
1	Catalonia	26.086826	87.796407
2	Central Spain	14.883333	85.000000
3	Levante	20.783784	85.567568
4	Northern Spain	36.237822	88.612225
5	Spain Other	24.000000	89.000000

Finally We make the plotting. We plot for each region from Spain the relationship between the mean price and the mean points given to his wines.

In [42]:
```
fig, ax = plt.subplots(1, 1, figsize=(10,3))

# To create the scatter plot
sns.scatterplot(data = df_reserva_price_value , x= price , y= points , hue=province, marker="p" , s = 400, lege

#to make the fontsize of the contry names in the graph bigger
sns.set_context("notebook", font_scale=1.4, rc={"font.size":15})

# To show the values of the x-axis near each point
for i, val in enumerate(price):
    plt.annotate(province[i], (price[i]+1.5, points[i]))

#Labels
ax.set_xlabel('PRICE', fontsize=20, fontweight='bold')
ax.set_ylabel('POINTS', fontsize=20, fontweight='bold')
#To set a grid
plt.grid(color='g', linestyle='--', linewidth=0.5)
#To set a color at the background
ax.set_facecolor('lightgrey')

plt.show()
```



The result tell us that "Spain Other" region has the best relationship, quality-price. Wines from Andalucia are sligtly better but they are far more expensive

This results are quite accurate, because the prices without a value were set as zero at the beginning was just arond 7%. So there is little bias.

In [43]:
```
#The number of nulls in the original merged dataframe
nulos=df_3.price.isna().sum()
```

In [44]:
```
#percentage on NANs at the price column in the original merged dataframe
nulos/len(df_3)*100
```

Out[44]:
```
6.921544036746659
```

Which is the most expensive wine?

In [45]:
```
df_reserva.price.max()
```

Out[45]:
```
600.0
```

In [46]:
```
max_row_index = df_reserva.price.idxmax()
df_reserva.loc[max_row_index]
```

Out[46]:
```
country                       Spain
designation    Unico Reserva Especial
points                           92
price                         600.0
province             Northern Spain
winery                 Vega Sicilia
Name: 97421, dtype: object
```

As we can see, even though this wine is the most expensive, it is not one of the most valued ones.

Questions?