Python Exercises

Part 1:

Video Game Sales

We have a file with video game sales data with the following columns:

- Ranking: Ranking of the sale of the video game worldwide
- Name: Name of the video game
- Platform: Platform on which it was launched (Wii, PS4, PC ... et)
- Year: Launch year
- Genre: Video game genre
- Publisher: Company that published the game
- NA_Sales - Sales in North America (in millions)
- EU_Sales - Sales in Europe (in millions)
- JP_Sales - Sales in Japan (in millions)
- Other_Sales - Sales in the rest of the world (in millions)
- Global_Sales - Worldwide Sales (in millions)

Loading the dataset:

In [1]:
```
import pandas as pd
from scipy import stats

games = pd.read_csv('vgsales.csv')

games.head()
```

Out[1]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37

Finding the video game that has generated the most revenue globally

In [2]:
```
games.sort_values('Global_Sales', ascending=False).head(1)["Name"]
```

Out[2]:
```
0    Wii Sports
Name: Name, dtype: object
```

Finding the platform with the most average global sales

In [3]:
```
plataformas = games.groupby('Platform')
plataformas.mean().sort_values('Global_Sales', ascending=False).head(1)
```

Out[3]:

Platform	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
GB	3392.030612	1995.958763	1.166531	0.487959	0.868571	0.083673	2.606633

Finding the publisher with the most global sales

In [4]:
```
empresas = games.groupby('Publisher')
empresas.sum().sort_values('Global_Sales', ascending=False).head(1)
```

Out[4]:

Publisher	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
Nintendo	2714164	1394666.0	816.87	418.74	455.42	95.33	1786.56

Discovering which year had the most global sales

In [5]:
```
años = games.groupby('Year')
años.sum().sort_values('Global_Sales', ascending=False).head(1)
```

Out[5]:

Year	Rank	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
2008.0	12088007	351.44	184.4	60.26	82.39	678.9

Finding the top selling game for each platform

In [6]:
```
años = games.groupby(['Platform', 'Name','Global_Sales'])
años.max().sort_values('Global_Sales', ascending=False).groupby('Platform').head(1)
```

Out[6]:

Platform	Name	Global_Sales	Rank	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales
Wii	Wii Sports	82.74	1	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46
NES	Super Mario Bros.	40.24	2	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77
GB	Pokemon Red/Pokemon Blue	31.37	5	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00
DS	New Super Mario Bros.	30.01	7	2006.0	Platform	Nintendo	11.38	9.23	6.50	2.90
X360	Kinect Adventures!	21.82	16	2010.0	Misc	Microsoft Game Studios	14.97	4.94	0.24	1.67
PS3	Grand Theft Auto V	21.40	17	2013.0	Action	Take-Two Interactive	7.01	9.27	0.97	4.14
PS2	Grand Theft Auto: San Andreas	20.81	18	2004.0	Action	Take-Two Interactive	9.43	0.40	0.41	10.57
SNES	Super Mario World	20.61	19	1990.0	Platform	Nintendo	12.78	3.75	3.54	0.55
GBA	Pokemon Ruby/Pokemon Sapphire	15.85	26	2002.0	Role-Playing	Nintendo	6.06	3.90	5.38	0.50
3DS	Pokemon X/Pokemon Y	14.35	33	2013.0	Role-Playing	Nintendo	5.17	4.05	4.34	0.79
PS4	Call of Duty: Black Ops 3	14.24	34	2015.0	Shooter	Activision	5.77	5.81	0.35	2.31
N64	Super Mario 64	11.89	47	1996.0	Platform	Nintendo	6.91	2.85	1.91	0.23
PS	Gran Turismo	10.95	53	1997.0	Racing	Sony Computer Entertainment	4.02	3.87	2.54	0.52
XB	Halo 2	8.49	80	2004.0	Shooter	Microsoft Game Studios	6.82	1.53	0.05	0.08
PC	The Sims 3	8.11	84	2009.0	Simulation	Electronic Arts	0.98	6.42	0.00	0.71
2600	Pac-Man	7.81	90	1982.0	Puzzle	Atari	7.28	0.45	0.00	0.08
PSP	Grand Theft Auto: Liberty City Stories	7.72	91	2005.0	Action	Take-Two Interactive	2.90	2.83	0.24	1.75
XOne	Call of Duty: Black Ops 3	7.30	102	2015.0	Shooter	Activision	4.52	2.09	0.01	0.67
GC	Super Smash Bros. Melee	7.07	108	2001.0	Fighting	Nintendo	4.41	1.04	1.39	0.22
WiiU	Mario Kart 8	6.96	109	2014.0	Racing	Nintendo	3.13	2.07	1.27	0.49
GEN	Sonic the Hedgehog 2	6.03	144	1992.0	Platform	Sega	4.47	1.20	0.16	0.19
DC	Sonic Adventure	2.42	638	1998.0	Platform	Sega	1.26	0.61	0.46	0.08
PSV	Minecraft	2.25	715	2014.0	Misc	Sony Computer Entertainment Europe	0.28	0.79	0.87	0.32
SAT	Virtua Fighter 2	1.93	890	1995.0	Fighting	Sega	0.34	0.26	1.30	0.03
SCD	Sonic CD	1.50	1263	1993.0	Platform	Sega	1.00	0.36	0.09	0.05
WS	Final Fantasy	0.51	3933	2000.0	Role-Playing	SquareSoft	0.00	0.00	0.51	0.00
NG	Samurai Shodown II	0.25	6683	1994.0	Fighting	SNK	0.00	0.00	0.25	0.00
TG16	Doukyuusei	0.14	9225	1995.0	Adventure	NEC	0.00	0.00	0.14	0.00
3DO	Policenauts	0.06	12637	1995.0	Adventure	Konami Digital Entertainment	0.00	0.00	0.06	0.00
GG	Sonic the Hedgehog 2 (8-bit)	0.04	13527	1992.0	Platform	Sega	0.00	0.00	0.04	0.00
PCFX	Blue Breaker: Ken Yorimo Hohoemi o	0.03	14559	1996.0	Role-Playing	NEC	0.00	0.00	0.03	0.00

Finding the top seller for each genre

In [7]:
```
años = games.groupby(['Genre', 'Name','Global_Sales'])
años.max().sort_values('Global_Sales', ascending=False).groupby('Genre').head(1)
```

Out[7]:

Genre	Name	Global_Sales	Rank	Platform	Year	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales
Sports	Wii Sports	82.74	1	Wii	2006.0	Nintendo	41.49	29.02	3.77	8.46
Platform	Super Mario Bros.	40.24	2	NES	1985.0	Nintendo	29.08	3.58	6.81	0.77
Racing	Mario Kart Wii	35.82	3	Wii	2008.0	Nintendo	15.85	12.88	3.79	3.31
Role-Playing	Pokemon Red/Pokemon Blue	31.37	5	GB	1996.0	Nintendo	11.27	8.89	10.22	1.00
Puzzle	Tetris	30.26	6	GB	1989.0	Nintendo	23.20	2.26	4.22	0.58
Misc	Wii Play	29.02	8	Wii	2006.0	Nintendo	14.03	9.20	2.93	2.85
Shooter	Duck Hunt	28.31	10	NES	1984.0	Nintendo	26.93	0.63	0.28	0.47
Simulation	Nintendogs	24.76	11	DS	2005.0	Nintendo	9.07	11.00	1.93	2.75
Action	Grand Theft Auto V	21.40	17	PS3	2013.0	Take-Two Interactive	7.01	9.27	0.97	4.14
Fighting	Super Smash Bros. Brawl	13.04	40	Wii	2008.0	Nintendo	6.75	2.61	2.66	1.02
Adventure	Super Mario Land 2: 6 Golden Coins	11.18	51	GB	1992.0	Nintendo	6.16	2.04	2.69	0.29
Strategy	Pokemon Stadium	5.45	166	N64	1999.0	Nintendo	3.18	1.24	0.94	0.09

Creating a function that groups video game release dates by decades, and shows how have evolved their global sales

In [8]:
```
import pandas as pd
from scipy import stats

def decades (row):
    year = row ["Year"]
    if 1979 < year < 1990:
        return "1980s"
    elif 1989 < year < 2000:
        return "1990s"
    elif 1999 < year < 2010:
        return "2000s"
    elif 2009 < year < 2018:
        return "2010s"
    else:
        return "Others"

games["Decade"] = games.apply(decades, axis="columns")
games.head(20)
```

Out[8]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Decade
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74	2000s
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24	1980s
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82	2000s
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00	2000s
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37	1990s
5	6	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26	4.22	0.58	30.26	1980s
6	7	New Super Mario Bros.	DS	2006.0	Platform	Nintendo	11.38	9.23	6.50	2.90	30.01	2000s
7	8	Wii Play	Wii	2006.0	Misc	Nintendo	14.03	9.20	2.93	2.85	29.02	2000s
8	9	New Super Mario Bros. Wii	Wii	2009.0	Platform	Nintendo	14.59	7.06	4.70	2.26	28.62	2000s
9	10	Duck Hunt	NES	1984.0	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31	1980s
10	11	Nintendogs	DS	2005.0	Simulation	Nintendo	9.07	11.00	1.93	2.75	24.76	2000s
11	12	Mario Kart DS	DS	2005.0	Racing	Nintendo	9.81	7.57	4.13	1.92	23.42	2000s
12	13	Pokemon Gold/Pokemon Silver	GB	1999.0	Role-Playing	Nintendo	9.00	6.18	7.20	0.71	23.10	1990s
13	14	Wii Fit	Wii	2007.0	Sports	Nintendo	8.94	8.03	3.60	2.15	22.72	2000s
14	15	Wii Fit Plus	Wii	2009.0	Sports	Nintendo	9.09	8.59	2.53	1.79	22.00	2000s
15	16	Kinect Adventures!	X360	2010.0	Misc	Microsoft Game Studios	14.97	4.94	0.24	1.67	21.82	2010s
16	17	Grand Theft Auto V	PS3	2013.0	Action	Take-Two Interactive	7.01	9.27	0.97	4.14	21.40	2010s
17	18	Grand Theft Auto: San Andreas	PS2	2004.0	Action	Take-Two Interactive	9.43	0.40	0.41	10.57	20.81	2000s
18	19	Super Mario World	SNES	1990.0	Platform	Nintendo	12.78	3.75	3.54	0.55	20.61	1990s
19	20	Brain Age: Train Your Brain in Minutes a Day	DS	2005.0	Misc	Nintendo	4.75	9.26	4.16	2.05	20.22	2000s

Part 2

5000 IMD Movies (The Internet Movie Database)

In this project we will focus on the exploratory data analysis of a 5000 movie dataset obtained from the [Internet Movie Database] (https://www.imdb.com).

In [9]:
```
import pandas as pd
import numpy as np
import datetime
```

Loading and viewing the dataset:

In [10]:
```
df = pd.read_csv('movie_metadata.csv')
```

In [11]:
```
df.dtypes
```

Out[11]:
```
color                         object
director_name                 object
num_critic_for_reviews       float64
duration                     float64
director_facebook_likes      float64
actor_3_facebook_likes       float64
actor_2_name                  object
actor_1_facebook_likes       float64
gross                        float64
genres                        object
actor_1_name                  object
movie_title                   object
num_voted_users                int64
cast_total_facebook_likes      int64
actor_3_name                  object
facenumber_in_poster         float64
plot_keywords                 object
movie_imdb_link               object
num_user_for_reviews         float64
language                      object
country                       object
content_rating                object
budget                       float64
title_year                   float64
actor_2_facebook_likes       float64
imdb_score                   float64
aspect_ratio                 float64
movie_facebook_likes           int64
dtype: object
```

In [12]:
```
df.head()
```

Out[12]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0

5 rows × 28 columns

Finding the difference between budget and total box office per director on an annual basis by adding a new column called `diff_gross`.

In [13]:
```
limpio = df.copy()
limpio = limpio.dropna(how='any')
limpio.head()
```

Out[13]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0
5	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	640.0

5 rows × 28 columns

In [14]:
```
limpio["diff_gross"] = limpio["gross"] - limpio["budget"]
limpio.head()
```

Out[14]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0
5	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	640.0

5 rows × 29 columns

In [15]:
```
directors = limpio.groupby('director_name')
directors.sum().sort_values('diff_gross', ascending=False).head()
```

Out[15]:

director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	gross	num_votec
Steven Spielberg	6526.0	3429.0	350000.0	42157.0	271942.0	4.114233e+09	82
George Lucas	1345.0	655.0	0.0	9929.0	82000.0	1.741418e+09	24
James Cameron	1878.0	1098.0	0.0	4303.0	38780.0	1.948126e+09	38
Joss Whedon	2317.0	606.0	0.0	57279.0	80000.0	1.730887e+09	26
Chris Columbus	1567.0	1398.0	0.0	17183.0	167946.0	1.618708e+09	19

Finding which directors have obtained the highest number of critic reviews

In [16]:
```
directors = limpio.groupby('director_name')
directors.sum().sort_values('num_critic_for_reviews', ascending=False).head()
```

Out[16]:

director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	gross	num_votec
Steven Spielberg	6526.0	3429.0	350000.0	42157.0	271942.0	4.114233e+09	82
Ridley Scott	4930.0	2419.0	0.0	10167.0	187713.0	1.337772e+09	46
Tim Burton	4200.0	1759.0	208000.0	46100.0	328705.0	2.071275e+09	35
Clint Eastwood	4172.0	2503.0	304000.0	6753.0	266874.0	1.378321e+09	30
Christopher Nolan	4090.0	1122.0	176000.0	93698.0	150716.0	1.813228e+09	81

IMDB is known for the disproportion between the evaluation of professional critics and that of the users of the platform; High values mean that professional critics value the film more and vice versa. I will be finding this ratio and adding it in a new column called `critic_ratio`.

In [17]:
```
limpio['critic_ratio'] = limpio['num_critic_for_reviews'] / limpio['num_user_for_reviews']
limpio.head()
```

Out[17]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0
5	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	640.0

5 rows × 30 columns

Finding the 20 best directors according to the critic ratio

In [18]:
```
nuevo = limpio.copy()
nuevo.set_index("director_name", inplace=True)

directors_new = nuevo.groupby('director_name')
directors_new.mean().sort_values('critic_ratio', ascending=False).head(20)
```

Out[18]:

director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	gross	num_voted_us
Karim Aïnouz	70.0	106.0	19.0	9.0	585.0	20262.0	173
Hans Canosa	365.0	84.0	3.0	446.0	10000.0	379122.0	1014
Hitoshi Matsumoto	72.0	99.0	17.0	16.0	31.0	22770.0	165
Joseph Dorman	29.0	93.0	0.0	59.0	399.0	906666.0	15
Benjamin Dickinson	36.0	97.0	4.0	150.0	446.0	62480.0	104
Aki Kaurismäki	205.0	93.0	592.0	36.0	232.0	611709.0	1526
Paul Bunnell	94.0	106.0	5.0	356.0	416.0	2436.0	34
Dagur Kári	61.0	99.0	29.0	64.0	117.0	19959.0	357
Anna Muylaert	111.0	112.0	9.0	4.0	61.0	375723.0	702
Michel Leclerc	87.0	100.0	3.0	9.0	74.0	513836.0	630
Hue Rhodes	45.0	85.0	0.0	966.0	22000.0	100669.0	252
Johnnie To	98.0	110.0	143.0	76.0	155.0	49413.0	691
E.L. Katz	193.0	88.0	3.0	307.0	3000.0	59379.0	1279
Tom Putnam	22.0	86.0	15.0	0.0	2.0	111300.0	57
Jirí Menzel	99.0	113.0	44.0	4.0	61.0	617228.0	618
Shari Springer Berman	104.0	108.0	30.0	281.0	474.0	453079.0	412
Huck Botko	46.0	86.0	10.0	126.0	189.0	535249.0	346
Jon Stewart	129.0	103.0	593.0	241.0	374.0	3093491.0	830
Jamie Travis	115.0	88.0	15.0	567.0	938.0	1243961.0	1342
Lena Dunham	113.0	98.0	969.0	433.0	969.0	389804.0	1181

Finding how many movies have been made on each genre (acknowledging that a single movie can belong inside more than a single genre)

First, we need to create a function to translate the genre column, that currently separates genres with the symbol "|"

In [19]:
```
nuevo['genre_list'] = nuevo['genres'].str.split("|", expand = False)

nuevo.head()
```

Out[19]:

director_name	color	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	
James Cameron	Color	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760
Gore Verbinski	Color	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309
Sam Mendes	Color	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200
Christopher Nolan	Color	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448
Andrew Stanton	Color	462.0	132.0	475.0	530.0	Samantha Morton	640.0	73

5 rows × 30 columns

In [20]:
```
from collections import Counter
nuevo.reset_index(drop=False).head()

list_of_genres = nuevo['genre_list'].to_list()
l = []

list(map(l.extend, list_of_genres))
Counter(l)

#import numpy as np
#np.asarray(Counter(l))
```

Out[20]:
```
Counter({'Action': 959,
         'Adventure': 781,
         'Fantasy': 507,
         'Sci-Fi': 496,
         'Thriller': 1117,
         'Romance': 859,
         'Animation': 196,
         'Comedy': 1461,
         'Family': 442,
         'Musical': 96,
         'Mystery': 384,
         'Western': 59,
         'Drama': 1893,
         'History': 149,
         'Sport': 148,
         'Crime': 709,
         'Horror': 392,
         'War': 152,
         'Biography': 239,
         'Music': 151,
         'Documentary': 45,
         'Film-Noir': 1})
```

Part 3

COVID-19 Evolution during the first wave

The data that we are going to explore comes from the dataset available at: https://github.com/datadista/datasets/tree/master/COVID%2019 which contains information on the accumulated daily situation of the SARS-CoV-2 coronavirus disease (COVID-19) in Spain in an accessible and reusable format.

In our case we are going to study the file [ccaa_covid19_datos_isciii.csv] (https://raw.githubusercontent.com/datadista/datasets/master/COVID%2019/ccaa_covid19_datos_isciii.csv), which contains data at the level of cases registered by Autonomous Communities.

Campo	Descripción	Formato	Ejemplo
fecha	Fecha de notificación	YYYY-MM-DD	2020-03-20
cod_ine	Código INE de CCAA	Texto	11
CCAA	Comunidad autónoma	Texto	Madrid
Caso	Casos confirmados acumulados (NaN si no hay casos)	Número	7
Hospitalizados	Casos acumulados que han precisado hospitalización (incluyen UCI)	Número	7
UCI	Casos acumulados que han precisado ingreso en UCI	Número	7
Fallecidos	Personas fallecidas acumuladas	Número	7
Recuperados	Personas curadas acumuladas	Número	7

Loading and printing basic dataset information:

In [21]:
```
import pandas as pd
covid_data= pd.read_csv('ccaa_covid19_datos_isciii.csv')
covid_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 893 entries, 0 to 892
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Fecha           893 non-null    object
 1   cod_ine         893 non-null    int64
 2   CCAA            893 non-null    object
 3   Casos           752 non-null    float64
 4   Hospitalizados  532 non-null    float64
 5   UCI             570 non-null    float64
 6   Fallecidos      570 non-null    float64
 7   Recuperados     549 non-null    float64
dtypes: float64(5), int64(1), object(2)
memory usage: 55.9+ KB
```

In [22]:
```
covid_data.head(5)
```

Out[22]:

	Fecha	cod_ine	CCAA	Casos	Hospitalizados	UCI	Fallecidos	Recuperados
0	2020-02-20	1	Andalucía	NaN	NaN	NaN	NaN	NaN
1	2020-02-20	2	Aragón	NaN	NaN	NaN	NaN	NaN
2	2020-02-20	3	Asturias	NaN	NaN	NaN	NaN	NaN
3	2020-02-20	4	Baleares	1.0	NaN	NaN	NaN	NaN
4	2020-02-20	5	Canarias	1.0	NaN	NaN	NaN	NaN

In [23]:
```
covid_data.tail(5)
```

Out[23]:

	Fecha	cod_ine	CCAA	Casos	Hospitalizados	UCI	Fallecidos	Recuperados
888	2020-04-06	19	Melilla	92.0	40.0	3.0	2.0	11.0
889	2020-04-06	14	Murcia	1283.0	500.0	84.0	78.0	193.0
890	2020-04-06	15	Navarra	3355.0	1488.0	124.0	202.0	421.0
891	2020-04-06	16	País Vasco	9021.0	4856.0	417.0	586.0	3728.0
892	2020-04-06	17	La Rioja	2846.0	918.0	66.0	160.0	994.0

In [24]:
```
print(covid_data.isnull().sum())
covid_data.fillna(0)
```

```
Fecha               0
cod_ine             0
CCAA                0
Casos             141
Hospitalizados    361
UCI               323
Fallecidos        323
Recuperados       344
dtype: int64
```

Out[24]:

	Fecha	cod_ine	CCAA	Casos	Hospitalizados	UCI	Fallecidos	Recuperados
0	2020-02-20	1	Andalucía	0.0	0.0	0.0	0.0	0.0
1	2020-02-20	2	Aragón	0.0	0.0	0.0	0.0	0.0
2	2020-02-20	3	Asturias	0.0	0.0	0.0	0.0	0.0
3	2020-02-20	4	Baleares	1.0	0.0	0.0	0.0	0.0
4	2020-02-20	5	Canarias	1.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
888	2020-04-06	19	Melilla	92.0	40.0	3.0	2.0	11.0
889	2020-04-06	14	Murcia	1283.0	500.0	84.0	78.0	193.0
890	2020-04-06	15	Navarra	3355.0	1488.0	124.0	202.0	421.0
891	2020-04-06	16	País Vasco	9021.0	4856.0	417.0	586.0	3728.0
892	2020-04-06	17	La Rioja	2846.0	918.0	66.0	160.0	994.0

893 rows × 8 columns

Writing a Python program to obtain the total number of confirmed cases, hospitalized, transferred to the ICU, deceased and recovered.

In [25]:
```
covid_data= pd.read_csv('ccaa_covid19_datos_isciii.csv')
covid_data = covid_data.fillna(0)

covid_data['Fecha'] = pd.to_datetime(covid_data['Fecha'])
covid_data = covid_data.astype({"Casos": int, "Hospitalizados": int, "UCI": int, "Fallecidos": int, "Recuperado


totales = covid_data.groupby('Fecha')
totales.sum().sort_values('Fecha', ascending=False).head(1)
```

Out[25]:

Fecha	cod_ine	Casos	Hospitalizados	UCI	Fallecidos	Recuperados
2020-04-06	190	140510	63093	7069	13798	43208

Finding the last number of confirmed, deceased and recovered cases by Community:

In [26]:
```
covid_data= pd.read_csv('ccaa_covid19_datos_isciii.csv')
covid_data = covid_data.fillna(0)
covid_data.set_index(['CCAA'], inplace=True)
covid_data.drop(['UCI', 'Hospitalizados', 'cod_ine'], axis = 1, inplace=True)

ccaa = covid_data.groupby('CCAA')
ccaa.sum().sort_values(['Casos', 'Fallecidos', 'Recuperados'], ascending=False).head(19)
```

Out[26]:

CCAA	Casos	Fallecidos	Recuperados
Madrid	479028.0	59066.0	161945.0
Cataluña	305025.0	25616.0	82249.0
Castilla La Mancha	109078.0	10885.0	7490.0
País Vasco	108397.0	5759.0	30228.0
Castilla y León	102533.0	8236.0	18777.0
Andalucía	98122.0	4529.0	4677.0
C. Valenciana	88299.0	5939.0	5823.0
Galicia	64799.0	1525.0	4279.0
Navarra	39164.0	1762.0	2982.0
Aragón	36484.0	2518.0	3521.0
La Rioja	32989.0	1412.0	7605.0
Extremadura	25751.0	2144.0	1732.0
Asturias	21563.0	856.0	1825.0
Canarias	21072.0	835.0	1181.0
Cantabria	17749.0	665.0	910.0
Baleares	17022.0	709.0	3339.0
Murcia	15746.0	512.0	869.0
Melilla	1062.0	18.0	46.0
Ceuta	655.0	17.0	10.0

Finding the latest number of confirmed cases, hospitalized, transferred to the ICU, deceased and recovered from COVID-19 in Navarra

In [27]:
```
import pandas as pd
covid_data= pd.read_csv('ccaa_covid19_datos_isciii.csv')
covid_data = covid_data.fillna(0)

covid_data.drop(['cod_ine'], axis = 1, inplace=True)

covid_data.loc[covid_data['CCAA'] == "Navarra"].tail(1)
```

Out[27]:

	Fecha	CCAA	Casos	Hospitalizados	UCI	Fallecidos	Recuperados
890	2020-04-06	Navarra	3355.0	1488.0	124.0	202.0	421.0

Adding a new column with active cases:

In [28]:
```
covid_data= pd.read_csv('ccaa_covid19_datos_isciii.csv')
covid_data = covid_data.fillna(0)

covid_data['Fecha'] = pd.to_datetime(covid_data['Fecha'])
covid_data = covid_data.astype({"Casos": int, "Hospitalizados": int, "UCI": int, "Fallecidos": int, "Recuperado
covid_data['Activos'] = covid_data['Casos'] - covid_data['Fallecidos'] + covid_data['Recuperados']
covid_data.head()
```

Out[28]:

	Fecha	cod_ine	CCAA	Casos	Hospitalizados	UCI	Fallecidos	Recuperados	Activos
0	2020-02-20	1	Andalucía	0	0	0	0	0	0
1	2020-02-20	2	Aragón	0	0	0	0	0	0
2	2020-02-20	3	Asturias	0	0	0	0	0	0
3	2020-02-20	4	Baleares	1	0	0	0	0	1
4	2020-02-20	5	Canarias	1	0	0	0	0	1

Finding when did the first COVID-19 cases and deaths occur in Spain

In [29]:
```
covid_data.loc[covid_data['Casos'] == 1].head(1)
```

Out[29]:

	Fecha	cod_ine	CCAA	Casos	Hospitalizados	UCI	Fallecidos	Recuperados	Activos
3	2020-02-20	4	Baleares	1	0	0	0	0	1

In [30]:
```
covid_data.loc[covid_data['Fallecidos'] == 1].head(1)
```

Out[30]:

	Fecha	cod_ine	CCAA	Casos	Hospitalizados	UCI	Fallecidos	Recuperados	Activos
333	2020-03-08	10	C. Valenciana	37	0	1	1	1	37

Finding when did the first deaths occur in each Community

In [31]:
```
muertes = covid_data[covid_data["Fallecidos"] > 0]
ccaa2 = muertes.groupby("CCAA")

ccaa2.head(1)
```

Out[31]:

	Fecha	cod_ine	CCAA	Casos	Hospitalizados	UCI	Fallecidos	Recuperados	Activos
324	2020-03-08	2	Aragón	13	0	3	3	0	10
331	2020-03-08	9	Cataluña	75	0	0	2	0	73
333	2020-03-08	10	C. Valenciana	37	0	1	1	1	37
336	2020-03-08	13	Madrid	469	0	53	16	0	453
340	2020-03-08	16	País Vasco	149	0	5	6	2	145
360	2020-03-09	17	La Rioja	144	0	1	1	0	143
372	2020-03-10	11	Extremadura	9	2	1	1	1	9
382	2020-03-11	3	Asturias	47	12	2	1	1	47
383	2020-03-11	4	Baleares	22	8	2	1	1	22
386	2020-03-11	8	Castilla La Mancha	115	39	9	1	2	116
418	2020-03-13	1	Andalucía	269	92	8	2	0	267
422	2020-03-13	5	Canarias	90	15	6	1	7	96
425	2020-03-13	7	Castilla y León	169	25	8	1	2	170
449	2020-03-14	12	Galicia	195	32	8	2	4	197
472	2020-03-15	15	Navarra	274	58	5	1	0	273
499	2020-03-17	6	Cantabria	68	25	3	1	10	77
566	2020-03-20	14	Murcia	240	61	14	1	1	240
660	2020-03-25	19	Melilla	39	16	2	1	0	38
693	2020-03-27	18	Ceuta	17	2	2	1	0	16