Get data set from Kaggel winemag-data-130k-v2.csv 33608 entries, 0 to 33607 Data columns (total 13 columns)

```
import pandas as pd
import numpy as np

# reviews = pd.read_csv("winemag-data-130k-v2.csv", index_col=0)
reviews = pd.read_csv("winemag-data-130k-v2_full.csv", index_col=0)
```

```
reviews.head()
```

	country	description	designation	points	price	province	region_1	r
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Wi
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Wi

```
len(reviews)
```

```
129971
```

rename column region_1 as region and region_2 as locale

```
renamedReviews = reviews.rename(columns = {'region_1':'region', 'region_2':'loca
renamedReviews.head()
```

	country	description	designation	points	price	province	region	
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Wi
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Wi

get info of dataframe

```
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129971 entries, 0 to 129970
Data columns (total 13 columns):
 #   Column                 Non-Null Count   Dtype 
---  ------                 --------------   ----- 
 0   country                129908 non-null  object
 1   description            129971 non-null  object
 2   designation            92506 non-null   object
 3   points                 129971 non-null  int64
 4   price                  120975 non-null  float64
 5   province               129908 non-null  object
 6   region_1               108724 non-null  object
 7   region_2               50511 non-null   object
 8   taster_name            103727 non-null  object
 9   taster_twitter_handle  98758 non-null   object
 10  title                  129971 non-null  object
 11  variety                129970 non-null  object
 12  winery                 129971 non-null  object
dtypes: float64(1), int64(1), object(11)
memory usage: 13.9+ MB
```

```
renamedReviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129971 entries, 0 to 129970
Data columns (total 13 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   country                129908 non-null  object
 1   description            129971 non-null  object
 2   designation            92506 non-null   object
 3   points                 129971 non-null  int64
 4   price                  120975 non-null  float64
 5   province               129908 non-null  object
 6   region                 108724 non-null  object
 7   locale                 50511 non-null   object
 8   taster_name            103727 non-null  object
 9   taster_twitter_handle  98758 non-null   object
 10  title                  129971 non-null  object
 11  variety                129970 non-null  object
 12  winery                 129971 non-null  object
dtypes: float64(1), int64(1), object(11)
memory usage: 13.9+ MB
```

Create a variable df containing the country, province, region_1, and region_2 columns of the records with the index labels 0, 1, 10, and 100

```
df = reviews[['country','province','region_1', 'region_2']]
df.head()
```

	country	province	region_1	region_2
0	Italy	Sicily & Sardinia	Etna	NaN
1	Portugal	Douro	NaN	NaN
2	US	Oregon	Willamette Valley	Willamette Valley
3	US	Michigan	Lake Michigan Shore	NaN
4	US	Oregon	Willamette Valley	Willamette Valley

```
df.iloc[[0,1,10,100]]
```

	country	province	region_1	region_2
0	Italy	Sicily & Sardinia	Etna	NaN
1	Portugal	Douro	NaN	NaN
10	US	California	Napa Valley	Napa
100	US	New York	Finger Lakes	Finger Lakes

What countries are represented in the review dataset? (Your answer should not include any duplicates.)

```
reviews['country'].unique()

    array(['Italy', 'Portugal', 'US', 'Spain', 'France', 'Germany',
           'Argentina', 'Chile', 'Australia', 'Austria', 'South Africa',
           'New Zealand', 'Israel', 'Hungary', 'Greece', 'Romania', 'Mexico',
           'Canada', nan, 'Turkey', 'Czech Republic', 'Slovenia',
           'Luxembourg', 'Croatia', 'Georgia', 'Uruguay', 'England',
           'Lebanon', 'Serbia', 'Brazil', 'Moldova', 'Morocco', 'Peru',
           'India', 'Bulgaria', 'Cyprus', 'Armenia', 'Switzerland',
           'Bosnia and Herzegovina', 'Ukraine', 'Slovakia', 'Macedonia',
           'China', 'Egypt'], dtype=object)
```

How often does each country appear in the dataset? Create a Series reviews_per_country mapping countries to the count of reviews of medicines from that country.

```
reviews_per_country = reviews['country'].value_counts()
reviews_per_country

    US                          54504
    France                      22093
    Italy                       19540
    Spain                        6645
    Portugal                     5691
    Chile                        4472
    Argentina                    3800
    Austria                      3345
    Australia                    2329
    Germany                      2165
    New Zealand                  1419
    South Africa                 1401
    Israel                        505
    Greece                        466
    Canada                        257
    Hungary                       146
    Bulgaria                      141
    Romania                       120
    Uruguay                       109
```

```
Turkey                       90
Slovenia                     87
Georgia                      86
England                      74
Croatia                      73
Mexico                       70
Moldova                      59
Brazil                       52
Lebanon                      35
Morocco                      28
Peru                         16
Ukraine                      14
Serbia                       12
Czech Republic               12
Macedonia                    12
Cyprus                       11
India                         9
Switzerland                   7
Luxembourg                    6
Bosnia and Herzegovina        2
Armenia                       2
Slovakia                      1
China                         1
Egypt                         1
Name: country, dtype: int64
```

Create variable centered_price containing a version of the price column with the mean price subtracted.

(Note: this 'centering' transformation is a common preprocessing step before applying various machine learning algorithms.)

```
reviews['price'].mean()
```

```
35.363389129985535
```

```
centered_price = reviews['price'] - reviews['price'].mean()
centered_price
```

```
0               NaN
1        -20.363389
2        -21.363389
3        -22.363389
4         29.636611
            ...
129966    -7.363389
129967    39.636611
129968    -5.363389
129969    -3.363389
129970   -14.363389
Name: price, Length: 129971, dtype: float64
```

```
reviews.head()
```

	country	description	designation	points	price	province	region_1	r
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Wi
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Wi

I'm an economical medicine buyer. Which medicine is the "best bargain"? Create a variable bargain_medicine with the title of the medicine with the highest points-to-price ratio in the dataset.

```
reviews.iloc[(reviews['points'] / reviews['price']).idxmax()]
```

```
country                                                                 US
description              There's a lot going on in this Merlot, which i...
designation                                                            NaN
points                                                                  86
price                                                                  4.0
province                                                        California
region_1                                                        California
region_2                                                  California Other
taster_name                                                            NaN
taster_twitter_handle                                                  NaN
title                                         Bandit NV Merlot (California)
variety                                                             Merlot
winery                                                              Bandit
```

```
Name: 64590, dtype: object
```

```
bargain_medicine = reviews.iloc[(reviews['points'] / reviews['price']).idxmax()]
bargain_medicine
```

```
'Bandit NV Merlot (California)'
```

There are only so many words you can use when describing a bottle of medicine. Is a medicine more likely to be "tropical" or "fruity"? Create a Series descriptor_counts counting how many times each of these two words appears in the description column in the dataset. (For simplicity, let's ignore the capitalized versions of

```
reviews['description'].map(lambda desc: 'tropical' in desc).sum()
```

```
3607
```

```
pd.Series({
    'tropical': reviews['description'].map(lambda desc: 'tropical' in desc).sum(
    'fruity': reviews['description'].map(lambda desc: 'fruity' in desc).sum(),
})
```

```
tropical    3607
fruity      9090
dtype: int64
```

We'd like to host these medicine reviews on our website, but a rating system ranging from 80 to 100 points is too hard to understand - we'd like to translate them into simple star ratings. A score of 95 or higher counts as 3 stars, a score of at least 85 but less than 95 is 2 stars. Any other score is 1 star.

Also, the Canadian Vintners Association bought a lot of ads on the site, so any medicines from Canada should automatically get 3 stars, regardless of points.

Create a series star_ratings with the number of stars corresponding to each review in the dataset.

```
def pointMapper(row):
  point = row.points
  if row.country == 'Canada':
    return 1
  if point >=95:
    return 3
  if point >= 85:
    return 2
  return 1
star_ratings = reviews.apply(pointMapper, axis='columns')
star_ratings
```

```
0         2
1         2
2         2
3         2
4         2
         ..
129966    2
129967    2
129968    2
129969    2
129970    2
Length: 129971, dtype: int64
```

1. What is the data type of the points column in the dataset?

```
reviews.points.dtype
```

```
dtype('int64')
```

3. Sometimes the price column is null. How many reviews in the dataset are missing a price?

```
reviews['price'].isnull().sum()
```

```
8996
```

4. What are the most common medicine-producing regions? Create a Series counting the number of times each value occurs in the region_1 field. This field is often missing data, so replace missing values with Unknown. Sort in descending order. Your output should look something like this:

Unknown 21247

Napa Valley 4480

...

Bardolino Superiore 1

Primitivo del Tarantino 1

Name: region_1, Length: 1230, dtype: int64

```
reviews['region_1'].fillna('Unknown', inplace=True)
reviews['region_1'].value_counts().sort_values(ascending=False)
```

```
Unknown                 21247
Napa Valley              4480
Columbia Valley (WA)     4124
Russian River Valley     3091
```

```
                         ...
    Offida Rosso              1
    Corton Perrières          1
    Isle St. George           1
    Geelong                   1
    Paestum                   1
    Name: region_1, Length: 1230, dtype: int64
```

Double-click (or enter) to edit

2. Create a Series from entries in the points column, but convert the entries to strings. Hint: strings are str in native Python.

```
reviews['points'].astype('str')
```

```
    0         87
    1         87
    2         87
    3         87
    4         87
              ..
    129966    90
    129967    90
    129968    90
    129969    90
    129970    90
    Name: points, Length: 129971, dtype: object
```

Who are the most common medicine reviewers in the dataset? Create a Series whose index is the taster_twitter_handle category from the dataset, and whose values count how many reviews each person wrote.

```
reviews.groupby('taster_twitter_handle').size()
```

```
    taster_twitter_handle
    @AnneInVino          3685
    @JoeCz               5147
    @bkfiona               27
    @gordone_cellars     4177
    @kerinokeefe        10776
    @laurbuzz            1835
    @mattkettmann        6332
    @paulgwine           9532
    @suskostrzewa        1085
    @vboone              9537
    @vossroger          25514
    @wawinereport        4966
    @wineschach         15134
    @winewchristina         6
    @worldwineguys       1005
    dtype: int64
```

2. What is the best medicine I can buy for a given amount of money? Create a Series whose index is medicine prices and whose values is the maximum number of points a medicine costing that much was given in a review. Sort the values by price, ascending (so that 4.0 dollars is at the top and 3300.0 dollars is at the bottom).

```
reviews.groupby('price')['points'].max()
```

```
price
4.0       86
5.0       87
6.0       88
7.0       91
8.0       91
          ..
1900.0    98
2000.0    97
2013.0    91
2500.0    96
3300.0    88
Name: points, Length: 390, dtype: int64
```

What are the minimum and maximum prices for each variety of medicine? Create a DataFrame whose index is the variety category from the dataset and whose values are the min and max values thereof.

```
minmax = reviews.groupby('variety').price.aggregate([min, max])
minmax
```

	min	max
variety		
Abouriou	15.0	75.0
Agiorgitiko	10.0	66.0
Aglianico	6.0	180.0
Aidani	27.0	27.0
Airen	8.0	10.0
...	...	...
Zinfandel	5.0	100.0
Zlahtina	13.0	16.0
Zweigelt	9.0	70.0
Çalkarası	19.0	19.0
Žilavka	15.0	15.0

707 rows × 2 columns

4. What are the most expensive medicine varieties? Create a variable sorted_varieties containing a copy of the dataframe from the previous question where varieties are sorted in descending order based on minimum price, then on maximum price (to break ties).

```
minmax.sort_values(by=['min', 'max'], ascending=False)
```

	min	max
variety		
Ramisco	495.0	495.0
Terrantez	236.0	236.0
Francisa	160.0	160.0
Rosenmuskateller	150.0	150.0
Tinta Negra Mole	112.0	112.0
...	...	...
Roscetto	NaN	NaN
Sauvignon Blanc-Sauvignon Gris	NaN	NaN
Tempranillo-Malbec	NaN	NaN
Vital	NaN	NaN
Zelen	NaN	NaN

707 rows × 2 columns

5. Create a Series whose index is reviewers and whose values is the average review score given out by that reviewer. Hint: you will need the taster_name and points columns.

```
reviews.groupby('taster_name').points.mean()
```

```
taster_name
Alexander Peartree    85.855422
Anna Lee C. Iijima    88.415629
Anne Krebiehl MW      90.562551
Carrie Dykes          86.395683
Christina Pickard     87.833333
Fiona Adams           86.888889
Jeff Jenssen          88.319756
Jim Gordon            88.626287
Joe Czerwinski        88.536235
Kerin O'Keefe         88.867947
Lauren Buzzeo         87.739510
```

```
    Matt Kettmann         90.008686
    Michael Schachner     86.907493
    Mike DeSimone         89.101167
    Paul Gregutt          89.082564
    Roger Voss            88.708003
    Sean P. Sullivan      88.755739
    Susan Kostrzewa       86.609217
    Virginie Boone        89.213379
    Name: points, dtype: float64
```

What combination of countries and varieties are most common? Create a Series whose index is a MultiIndexof {country, variety} pairs. For example, a pinot noir produced in the US should map to {"US", "Pinot Noir"}. Sort the values in the Series in descending order based on medicine count.

```
reviews.groupby(['country', 'variety']).size().sort_values(ascending=False)
```

```
    country  variety
    US       Pinot Noir                  9885
             Cabernet Sauvignon          7315
             Chardonnay                  6801
    France   Bordeaux-style Red Blend    4725
    Italy    Red Blend                   3624
                                         ...
    Mexico   Cinsault                       1
             Grenache                       1
             Merlot                         1
             Rosado                         1
    Uruguay  White Blend                    1
    Length: 1612, dtype: int64
```

Practice Exercise 2

In this assignment, you will try to find some interesting insights into a few movies released between 1916 and 2016, using Python. You will have to download a movie dataset, write Python code to explore the data, gain insights into the movies, actors, directors, and collections, and submit the code.

Some tips before starting the assignment

1. Identify the task to be performed correctly, and only then proceed to write the required code. Don't perform any incorrect analysis or look for information that isn't required for the assignment.
2. In some cases, the variable names have already been assigned, and you just need to write code against them. In other cases, the names to be given are mentioned in the instructions. We strongly advise you to use the mentioned names only.
3. Always keep inspecting your data frame after you have performed a particular set of operations.
4. There are some checkpoints given in the IPython notebook provided. They're just useful pieces of information you can use to check if the result you have obtained after performing a particular task is correct or not.
5. Note that you will be asked to refer to documentation for solving some of the questions. That is done on purpose for you to learn new commands and also how to use the documentation.

```
# Import the numpy and pandas packages

import numpy as np
import pandas as pd
```

Task 1: Reading and Inspection

Subtask 1.1: Import and read

Import and read the movie database. Store it in a variable called `movies`.

```
# Write your code for importing the csv file here
movies = pd.read_csv('Movies.csv')
movies
```

Subtask 1.2: Inspect the dataframe

Inspect the dataframe's columns, shapes, variable types etc.

```
# Write your code for inspection here
movies.shape
movies.info()
movies.describe()
```

Question 1: How many rows and columns are present in the dataframe?

- (3821, 26)
- (3879, 28)
- (3853, 28)
- (3866, 26)

```
# write a code to count the no of columns with null values
a = movies.isnull().sum()
a[a>0].count()
```

```
12
```

Question 2: How many columns have null values present in them? Try writing a code for this instead of counting them manually.

- 3
- 6
- 9
- 12

Task 2: Cleaning the Data

Subtask 2.1: Drop unecessary columns

For this assignment, you will mostly be analyzing the movies with respect to the ratings, gross collection, popularity of movies, etc. So many of the columns in this dataframe are not required. So it is advised to drop the following columns.

- color
- director_facebook_likes
- actor_1_facebook_likes
- actor_2_facebook_likes
- actor_3_facebook_likes
- actor_2_name
- cast_total_facebook_likes
- actor_3_name
- duration
- facenumber_in_poster
- content_rating
- country
- movie_imdb_link
- aspect_ratio
- plot_keywords

```
# Check the 'drop' function in the Pandas library - dataframe.drop(list_of_unnecessary_columns, axis = )
# Write your code for dropping the columns here. It is advised to keep inspecting the dataframe after each set of operations

movies.drop(['color',
'director_facebook_likes',
'actor_1_facebook_likes',
'actor_2_facebook_likes',
'actor_3_facebook_likes',
'actor_2_name',
'cast_total_facebook_likes',
'actor_3_name',
'duration',
'facenumber_in_poster',
'content_rating',
'country',
'movie_imdb_link',
'aspect_ratio',
'plot_keywords'],axis=1,inplace=True)
movies.shape
```

```
(3853, 13)
```

Question 3: What is the count of columns in the new dataframe?

- 10
- 13
- 15
- 17

Subtask 2.2: Inspect Null values

As you have seen above, there are null values in multiple columns of the dataframe 'movies'. Find out the percentage of null values in each column of the dataframe 'movies'.

```
# Write you code here
(movies.isnull().sum()/len(movies))*100
```

```
director_name             0.000000
num_critic_for_reviews    0.025954
gross                     0.000000
genres                    0.000000
actor_1_name              0.000000
movie_title               0.000000
num_voted_users           0.000000
num_user_for_reviews      0.000000
language                  0.103815
budget                    0.000000
title_year                0.000000
imdb_score                0.000000
movie_facebook_likes      0.000000
dtype: float64
```

Question 4: Which column has the highest percentage of null values?

- language
- genres
- num_critic_for_reviews
- imdb_score

Subtask 2.3: Fill NaN values

You might notice that the `language` column has some NaN values. Here, on inspection, you will see that it is safe to replace all the missing values with `'English'`.

```
# You might notice that the `language` column has some NaN values. Here, on inspection, you will see that it is safe to replace all the missi
# Write your code for filling the NaN values in the 'language' column here
movies['language'].fillna('English',inplace=True)
movies['language'].isnull().sum()
movies['language'].value_counts()['English']
```

```
3675
```

Question 5: What is the count of movies made in English language after replacing the NaN values with English?

- 3670
- 3674
- 3668
- 3672

Task 3: Data Analysis

Subtask 3.1: Change the unit of columns

Convert the unit of the `budget` and `gross` columns from `$` to `million $`.

```
# Write your code for unit conversion here
# Convert the unit of the `budget` and `gross` columns from `$` to `million $`.
movies['budget'] = movies['budget']/1000000
movies['gross'] = movies['gross']/1000000
movies.head()
```

Subtask 3.2: Find the movies with highest profit

1. Create a new column called `profit` which contains the difference of the two columns: `gross` and `budget`.
2. Sort the dataframe using the `profit` column as reference. (Find which command can be used here to sort entries from the documentation)
3. Extract the top ten profiting movies in descending order and store them in a new dataframe - `top10`

```
# Write your code for creating the profit column here
movies['profit'] = movies['gross'] - movies['budget']
movies.head()
```

```
# Write your code for sorting the dataframe here
movies.sort_values(by='profit',ascending=False,inplace=True)
movies.head()
```

```
# Extract the top ten profiting movies in descending order and store them in a new dataframe - `top10`
top10 = movies.head(10)
top10
```

Checkpoint: You might spot two movies directed by `James Cameron` in the list.

Question 6: Which movie is ranked 5th from the top in the list obtained?

- E.T. the Extra-Terrestrial
- The Avengers
- The Dark Knight
- Titanic

Subtask 3.3: Find IMDb Top 250

Create a new dataframe `IMDb_Top_250` and store the top 250 movies with the highest IMDb Rating (corresponding to the column: `imdb_score`). Also make sure that for all of these movies, the `num_voted_users` is greater than 25,000.

Also add a `Rank` column containing the values 1 to 250 indicating the ranks of the corresponding films.

```
# Write your code for extracting the top 250 movies as per the IMDb score here. Make sure that you store it in a new dataframe
# and name that dataframe as 'IMDb_Top_250'
# Write your code to extract top rated movies here  # HINT: you can use the 'sort_values' function here to get the top 250 movies

IMDb_Top_250 = movies.sort_values(by='imdb_score',ascending=False).head(250)
IMDb_Top_250.head()
```

Question 7: Suppose movies are divided into 5 buckets based on the IMDb ratings:

- 7.5 to 8
- 8 to 8.5
- 8.5 to 9
- 9 to 9.5
- 9.5 to 10

Which bucket holds the maximum number of movies from *IMDb_Top_250*?

Subtask 3.4: Find the critic-favorite and audience-favorite actors

1. Create three new dataframes namely, `Meryl_Streep`, `Leo_Caprio`, and `Brad_Pitt` which contain the movies in which the actors: 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' are the lead actors. Use only the `actor_1_name` column for extraction. Also, make sure that you use the names 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' for the said extraction.
2. Append the rows of all these dataframes and store them in a new dataframe named `Combined`.
3. Group the combined dataframe using the `actor_1_name` column.
4. Find the mean of the `num_critic_for_reviews` and `num_user_for_review` and identify the actors which have the highest mean.

```
# Write your code for creating three new dataframes here
# Include all movies in which Meryl_Streep is the lead
# Include all movies in which Leo_Caprio is the lead
# write the code

Meryl_Streep =  movies[movies['actor_1_name']=='Meryl Streep']
Meryl_Streep.head()
```

```
# Include all movies in which Leo_Caprio is the lead
Leo_Caprio = movies[movies['actor_1_name']=='Leonardo DiCaprio']
Leo_Caprio.head()
```

```
# Include all movies in which Brad_Pitt is the lead
Brad_Pitt = movies[movies['actor_1_name']=='Brad Pitt']
Brad_Pitt.head()
```

```
# Write your code for combining the three dataframes here
Combined = pd.concat([Meryl_Streep,Leo_Caprio,Brad_Pitt])
Combined.head()
```

```
# Write your code for grouping the combined dataframe here

Combined.groupby('actor_1_name').agg({'num_critic_for_reviews':'mean','num_user_for_reviews':'mean','imdb_score':'mean'}).sort_values(by='imd
# Write the code for finding the mean of critic reviews and audience reviews here

movies['num_critic_for_reviews'].mean()
movies['num_user_for_reviews'].mean()
```

```
326.72047754996106
```

Question 8: Which actor is highest rated among the three actors according to the user reviews?

- Meryl Streep
- Leonardo DiCaprio
- Brad Pitt

Question 9: Which actor is highest rated among the three actors according to the critics?

- Meryl Streep
- Leonardo DiCaprio
- Brad Pitt

Task2 Amazon Prime video data analysis

https://www.kaggle.com/datasets/shivamb/amazon-prime-movies-and-tv-shows?resource=download

Show uniques values of a column 'director'

```
df = pd.read_csv('./amazon_prime_titles.csv/amazon_prime_titles.csv')
df.head()

# Show uniques values of a column 'director'
df['director'].unique()
```

```
array(['Don McKellar', 'Girish Joshi', 'Josh Webber', ...,
       'John-Paul Davidson, Stephen Warbeck', 'Emily Skye',
       'Steve Barker'], dtype=object)
```

show all unique values with their counts

```
# show all unique values with their counts
df['director'].value_counts()
```

```
director
Mark Knight               113
Cannis Holder              61
Moonbug Entertainment      37
Jay Chapman                34
Arthur van Merwijk         30
                         ...
Karyn Kusama                1
K. Subash                   1
Robert Cuffley              1
```

```
J. Sabarish                1
Steve Barker               1
Name: count, Length: 5773, dtype: int64
```

get total no of uniwue values of whole data frame

```
df.nunique()
```

```
show_id         9668
type               2
title           9668
director        5773
cast            7927
country           86
date_added        84
release_year     100
rating            24
duration         219
listed_in        518
description     9414
dtype: int64
```

In which year highest no of TV shows and movies were released

```
df['year'] = pd.DatetimeIndex(df['date_added']).year
```

```
year_with_most_releases = df['year'].value_counts().idxmax()
year_with_most_releases
```

```
2021.0
```

how many TV and Movie shows are there in Data frame

```
# how many TV and Movie shows are there in Data frame
df['type'].value_counts()
```

```
type
Movie      7814
TV Show    1854
Name: count, dtype: int64
```

show all records with type 'movies; and country united kingdom

```
# show all records with type 'movies; and country united kingdom
df[(df['type']=='Movie') & (df['country']=='United Kingdom')]
# df[(df['type']=='Movie') & (df['country']=='United Kingdom')].shape
```

show all movie records directed by Paul

```
df['year'] = pd.DatetimeIndex(df['date_added']).year
df.head()
```

Show top 3 Directors, who gave highest no of TV shows and movies released on Prime video

```
# Show top 3 Directors, who gave highest no of TV shows and movies released on Prime video
df['director'].value_counts().head(3)
```

```
director
Mark Knight              113
Cannis Holder             61
Moonbug Entertainment     37
Name: count, dtype: int64
```

In which year Highest rating show was there

```
# Show the row with the highest rating
# note - rating has nan values and values like 13+, 18+ etc
# In which year Highest rating show was there
df['rating'].value_counts().head(3)
```

```
rating
13+    2117
16+    1547
ALL    1268
Name: count, dtype: int64
```

Task 3 Netflix Analysis

Information about TV shows and Movies 1- upload csv

2- describe, info,dtypes

3- uniques values of each column

4- total no of unique values of Dataframe

5- Unique values with their count

6-is any missing value with count

7- who is the director and show id of show #"ZOO"

8- Convert Datatype of column release date to DateTime

9-In which year highest no of TV shows and Movies relaesed

10-How many movies and TV shows are there in data set

11- Display Titles of all TV shows that were released in " United Sates" only

12- show top 10 Directors who gave highest no of TV shows and Movies on Netflix

13- show the record of all 'Horror' type of Movies

14 What are different 'Ratings' given by Netflix

15- What is Maximum duration of TV show on Netflix

16-sort dataframe by year

Get data set from Kaggel winemag-data-130k-v2.csv 33608 entries, 0 to 33607 Data columns (total 13 columns)

```
import pandas as pd

reviews = pd.read_csv("./winemag-data-130k-v2.csv", index_col=0)
reviews.head(3)
```

	country	description	designation	points	price	province	region_1	region_2	tast
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	
		This is ripe and fruity, a							

rename column region_1 as region and region_2 as locale

```
# rename column region_1 as region and region_2 as locale
reviews.rename(columns={'region_1': 'region', 'region_2': 'locale'}, inplace=True)
reviews.head(3)
```

	country	description	designation	points	price	province	region	locale	tast
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	
		This is ripe and fruity, a							

get info of dataframe

```
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 37604 entries, 0 to 37603
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   country                37588 non-null  object 
 1   description            37604 non-null  object 
 2   designation            26863 non-null  object 
 3   points                 37604 non-null  int64  
 4   price                  34880 non-null  float64
 5   province               37588 non-null  object 
 6   region                 31315 non-null  object 
 7   locale                 14441 non-null  object 
 8   taster_name            29961 non-null  object 
 9   taster_twitter_handle  28550 non-null  object 
 10  title                  37604 non-null  object 
 11  variety                37604 non-null  object 
 12  winery                 37604 non-null  object 
dtypes: float64(1), int64(1), object(11)
memory usage: 4.0+ MB
```

Create a variable df containing the country, province, region_1, and region_2 columns of the records with the index labels 0, 1, 10, and 100

```
# Create a variable df containing the country, province, region_1, and region_2 columns of the records with the index labels 0, 1, 10, and 10

df = reviews.loc[[0, 1, 10, 100], ['country', 'province', 'region', 'locale']]
df
```

	country	province	region	locale
0	Italy	Sicily & Sardinia	Etna	NaN
1	Portugal	Douro	NaN	NaN
10	US	California	Napa Valley	Napa
100	US	New York	Finger Lakes	Finger Lakes

What countries are represented in the review dataset? (Your answer should not include any duplicates.)

```
# What countries are represented in the review dataset? (Your answer should not include any duplicates.)
reviews.country.unique()
```

```
array(['Italy', 'Portugal', 'US', 'Spain', 'France', 'Germany',
       'Argentina', 'Chile', 'Australia', 'Austria', 'South Africa',
       'New Zealand', 'Israel', 'Hungary', 'Greece', 'Romania', 'Mexico',
       'Canada', nan, 'Turkey', 'Czech Republic', 'Slovenia',
       'Luxembourg', 'Croatia', 'Georgia', 'Uruguay', 'England',
       'Lebanon', 'Serbia', 'Brazil', 'Moldova', 'Morocco', 'Peru',
       'India', 'Bulgaria', 'Cyprus', 'Armenia', 'Switzerland',
       'Bosnia and Herzegovina', 'Ukraine', 'Slovakia', 'Macedonia'],
      dtype=object)
```

How often does each country appear in the dataset? Create a Series reviews_per_country mapping countries to the count of reviews of medicines from that country.

```
# How often does each country appear in the dataset? Create a Series reviews_per_country mapping countries to the count of reviews of medicin
reviews_per_country = reviews.country.value_counts()
reviews_per_country
```

```
country
US                        15551
France                     6335
Italy                      5746
Spain                      1924
Portugal                   1704
Chile                      1329
Argentina                  1142
Austria                     991
Australia                   657
Germany                     631
New Zealand                 441
South Africa                434
Israel                      154
Greece                      137
Canada                       71
Bulgaria                     40
Uruguay                      37
Romania                      35
Hungary                      29
Croatia                      28
Brazil                       23
Turkey                       21
Georgia                      21
Mexico                       20
Moldova                      15
Slovenia                     15
England                      12
Lebanon                       9
Peru                          5
Czech Republic                5
Morocco                       4
Cyprus                        4
Ukraine                       4
Serbia                        3
India                         3
Switzerland                   3
Luxembourg                    1
Armenia                       1
Bosnia and Herzegovina        1
Slovakia                      1
Macedonia                     1
Name: count, dtype: int64
```

Create variable centered_price containing a version of the price column with the mean price subtracted.

(Note: this 'centering' transformation is a common preprocessing step before applying various machine learning algorithms.)

```
# Create variable centered_price containing a version of the price column with the mean price subtracted.
centered_price = reviews.price - reviews.price.mean()
centered_price
```

```
0              NaN
1       -20.032311
2       -21.032311
3       -22.032311
4        29.967689
           ...
37599   -23.032311
37600   -24.032311
37601   -10.032311
37602   -16.032311
37603    -7.032311
Name: price, Length: 37604, dtype: float64
```

I'm an economical medicine buyer. Which medicine is the "best bargain"? Create a variable bargain_medicine with the title of the medicine with the highest points-to-price ratio in the dataset.

```
# I'm an economical medicine buyer. Which medicine is the "best bargain"? Create a variable bargain_medicine with the title of the medicine w
bargain_idx = (reviews.points / reviews.price).idxmax()
bargain_medicine = reviews.loc[bargain_idx, 'title']
bargain_medicine
```

```
'Felix Solis 2013 Flirty Bird Syrah (Vino de la Tierra de Castilla)'
```

There are only so many words you can use when describing a bottle of medicine. Is a medicine more likely to be "tropical" or "fruity"? Create a Series descriptor_counts counting how many times each of these two words appears in the description column in the dataset. (For simplicity, let's ignore the capitalized versions of

```
# There are only so many words you can use when describing a bottle of medicine. Is a medicine more likely to be "tropical" or "fruity"? Crea
n_trop = reviews.description.map(lambda desc: "tropical" in desc).sum()
n_fruity = reviews.description.map(lambda desc: "fruity" in desc).sum()
descriptor_counts = pd.Series([n_trop, n_fruity], index=['tropical', 'fruity'])
descriptor_counts
```

```
tropical    1042
fruity      2639
dtype: int64
```

We'd like to host these medicine reviews on our website, but a rating system ranging from 80 to 100 points is too hard to understand - we'd like to translate them into simple star ratings. A score of 95 or higher counts as 3 stars, a score of at least 85 but less than 95 is 2 stars. Any other score is 1 star.

Also, the Canadian Vintners Association bought a lot of ads on the site, so any medicines from Canada should automatically get 3 stars, regardless of points.

Create a series star_ratings with the number of stars corresponding to each review in the dataset.

```
def stars(row):
    if row.country == 'Canada':
        return 3
    elif row.points >= 95:
        return 3
    elif row.points >= 85:
        return 2
    else:
        return 1
star_ratings = reviews.apply(stars, axis='columns')
star_ratings
```

```
0        2
1        2
```

```
2        2
3        2
4        2
        ..
37599    2
37600    2
37601    2
37602    2
37603    2
Length: 37604, dtype: int64
```

1. What is the data type of the points column in the dataset?

```
reviews.points.dtype
```

```
dtype('int64')
```

3. Sometimes the price column is null. How many reviews in the dataset are missing a price?

```
# Sometimes the price column is null. How many reviews in the dataset are missing a price?
n_missing_prices = reviews.price.isnull().sum()
n_missing_prices
```

```
2724
```

4. What are the most common medicine-producing regions? Create a Series counting the number of times each value occurs in the region_1 field. This field is often missing data, so replace missing values with Unknown. Sort in descending order. Your output should look something like this:

Unknown 21247

Napa Valley 4480

...

Bardolino Superiore 1

Primitivo del Tarantino 1

Name: region_1, Length: 1230, dtype: int64

```
# 4.
# What are the most common medicine-producing regions? Create a Series counting the number of times each value occurs in the region_1 field.

# Unknown                    21247

# Napa Valley                 4480

#                           ...
# Bardolino Superiore            1

# Primitivo del Tarantino        1

# Name: region_1, Length: 1230, dtype: int64

reviews.region_1.fillna("Unknown").value_counts().sort_values(ascending=False)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8252\1995031971.py in ?()
     11 # Primitivo del Tarantino        1
     12 
     13 # Name: region_1, Length: 1230, dtype: int64
     14 
```

Double-click (or enter) to edit

2. Create a Series from entries in the points column, but convert the entries to strings. Hint: strings are str in native Python.

```
# Create a Series from entries in the points column, but convert the entries to strings. Hint: strings are str in native Python.
point_strings = reviews.points.astype(str)
point_strings
```

```
0        87
1        87
2        87
3        87
4        87
         ..
37599    87
37600    87
37601    87
37602    87
37603    87
Name: points, Length: 37604, dtype: object
```

Who are the most common medicine reviewers in the dataset? Create a Series whose index is the taster_twitter_handle category from the dataset, and whose values count how many reviews each person wrote.

```
# Who are the most common medicine reviewers in the dataset? Create a Series whose index is the taster_twitter_handle category from the datas
reviews_written = reviews.groupby('taster_twitter_handle').size()
reviews_written
```

```
taster_twitter_handle
@AnneInVino         1042
@JoeCz              1468
@bkfiona              11
@gordone_cellars    1189
@kerinokeefe        3092
@laurbuzz            559
@mattkettmann       1773
@paulgwine          2746
@suskostrzewa        342
@vboone             2776
@vossroger          7454
@wawinereport       1339
@wineschach         4517
@winewchristina        1
@worldwineguys       241
dtype: int64
```

2. What is the best medicine I can buy for a given amount of money? Create a Series whose index is medicine prices and whose values is the maximum number of points a medicine costing that much was given in a review. Sort the values by price, ascending (so that 4.0 dollars is at the top and 3300.0 dollars is at the bottom).

```
# 2.
# What is the best medicine I can buy for a given amount of money? Create a Series whose index is medicine prices and whose values is the max
best_rating_per_price = reviews.groupby('price')['points'].max().sort_index()
best_rating_per_price
```

```
price
4.0       85
5.0       87
6.0       87
7.0       91
8.0       91
          ..
1100.0    97
1200.0    96
1300.0    96
1900.0    98
```

```
2500.0    96
Name: points, Length: 264, dtype: int64
```

What are the minimum and maximum prices for each variety of medicine? Create a DataFrame whose index is the variety category from the dataset and whose values are the min and max values thereof.

```
# What are the minimum and maximum prices for each variety of medicine? Create a DataFrame whose index is the variety category from the datas
price_extremes = reviews.groupby('variety').price.agg([min, max])
price_extremes
```

```
C:\Users\Shree\AppData\Local\Temp\ipykernel_8252\1985091541.py:2: FutureWarning: The prc
  price_extremes = reviews.groupby('variety').price.agg([min, max])
C:\Users\Shree\AppData\Local\Temp\ipykernel_8252\1985091541.py:2: FutureWarning: The prc
  price_extremes = reviews.groupby('variety').price.agg([min, max])
```

	min	max
variety		
Agiorgitiko	10.0	66.0
Aglianico	6.0	130.0
Albana	14.0	25.0
Albanello	20.0	20.0
Albariño	10.0	75.0
...	...	...
Zibibbo	23.0	51.0
Zierfandler	15.0	40.0
Zierfandler-Rotgipfler	20.0	25.0
Zinfandel	5.0	100.0
Zweigelt	9.0	52.0

513 rows × 2 columns

4. What are the most expensive medicine varieties? Create a variable sorted_varieties containing a copy of the dataframe from the previous question where varieties are sorted in descending order based on minimum price, then on maximum price (to break ties).

```
# 4. What are the most expensive medicine varieties? Create a variable sorted_varieties containing a copy of the dataframe from the previous
sorted_varieties = price_extremes.sort_values(by=['min', 'max'], ascending=False)
sorted_varieties
```

	min	max
variety		
Terrantez	236.0	236.0
Bual	194.0	230.0
Rosenmuskateller	150.0	150.0
Debit	130.0	130.0
Malbec-Cabernet	130.0	130.0
...	...	...
Greco Bianco	NaN	NaN
Madeira Blend	NaN	NaN
Sämling	NaN	NaN
White Port	NaN	NaN
Zelen	NaN	NaN

513 rows × 2 columns

5. Create a Series whose index is reviewers and whose values is the average review score given out by that reviewer. Hint: you will need the taster_name and points columns.

```
# # 5.
# Create a Series whose index is reviewers and whose values is the average review score given out by that reviewer. Hint: you will need the t
reviewer_mean_ratings = reviews.groupby('taster_name').points.mean()
reviewer_mean_ratings
```

```
taster_name
Alexander Peartree    85.951923
Anna Lee C. Iijima    88.459671
Anne Krebiehl MW      90.683301
Carrie Dykes          86.333333
Christina Pickard     93.000000
Fiona Adams           87.090909
Jeff Jenssen          88.151261
Jim Gordon            88.533221
Joe Czerwinski        88.472071
Kerin O’Keefe         88.760026
Lauren Buzzeo         87.765653
Matt Kettmann         90.122391
Michael Schachner     86.883330
Mike DeSimone         89.106557
Paul Gregutt          89.053168
Roger Voss            88.663939
Sean P. Sullivan      88.669903
Susan Kostrzewa       86.447368
Virginie Boone        89.210375
Name: points, dtype: float64
```

What combination of countries and varieties are most common? Create a Series whose index is a MultiIndexof {country, variety} pairs. For example, a pinot noir produced in the US should map to {"US", "Pinot Noir"}. Sort the values in the Series in descending order based on medicine count.

```
# What combination of countries and varieties are most common? Create a Series whose index is a MultiIndexof {country, variety} pairs. For ex
country_variety_counts = reviews.groupby(['country', 'variety']).size().sort_values(ascending=False)
country_variety_counts
```

```
country  variety
US       Pinot Noir                  2814
         Cabernet Sauvignon          2079
         Chardonnay                  1925
France   Bordeaux-style Red Blend    1439
Italy    Red Blend                   1087
                                     ...
         Tocai                          1
         Sémillon                       1
         Susumaniello                   1
         Shiraz                         1
Uruguay  Tempranillo-Tannat             1
Length: 1090, dtype: int64
```