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

In [ ]:
```
import pandas as pd
```

In [ ]:
```
reviews = pd.read_csv("winemag-data-130k-v2.csv", index_col=0)
```

rename column region_1 as region and region_2 as locale

In [ ]:
```
reviews.rename(columns = {'region_1':'region','region_2':'locale'})
```

Out[ ]:

	country	description	designation	points	price	province	region	locale	taster_name	taster_twitter_handle	
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	Nicosia Vulkà B (
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Quint Avidagos Avidago (D
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Rainstorm Pino (Willar V
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN	St. Julian Reserve Harvest Rie
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Sweet Ch 2012 Vint Reserve C
...	...	...	...	...	...	...	...	...	...	...	
129966	Germany	Notes of honeysuckle and cantaloupe sweeten th...	Brauneberger Juffer-Sonnenuhr Spätlese	90	28.0	Mosel	NaN	NaN	Anna Lee C. Iijima	NaN	Dr. H. Tha (Erben M Burgg 2(

	country	description	designation	points	price	province	region	locale	taster_name	taster_twitter_handle	
129967	US	Citation is given as much as a decade of bottl...	NaN	90	75.0	Oregon	Oregon	Oregon Other	Paul Gregutt	@paulgwine	Citation Pinot (Ore
129968	France	Well-drained gravel soil gives this wine its c...	Kritt	90	30.0	Alsace	Alsace	NaN	Roger Voss	@vossroger	Dor Gresser Gewurztrar
129969	France	A dry style of Pinot Gris, this is crisp with ...	NaN	90	32.0	Alsace	Alsace	NaN	Roger Voss	@vossroger	Dor Marcel 2012 Pino (Al
129970	France	Big, rich and off-dry, this is powered by inte...	Lieu-dit Harth Cuvée Caroline	90	21.0	Alsace	Alsace	NaN	Roger Voss	@vossroger	Dor Schoffit Lieu-dit H Cuvée

129971 rows × 13 columns

get info of dataframe

In [ ]:
```
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 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
```

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

In [ ]:
```
df = reviews[['country','province','region_1','region_2']].iloc[:4,:]
df.rename(index = {1:1,2:10,3:100})
```

Out[ ]:

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

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

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

Out[ ]:
```
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.

In [ ]:
```
reviews_per_country = pd.Series(reviews['country'].value_counts())
reviews_per_country
```

Out[ ]:

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: 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.)

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

Out[ ]:
```
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
```

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.

In [ ]:
```
bargain_medicine = pd.DataFrame({'Title':reviews['title'],'ratio':(reviews['points']/reviews['price'])})
bargain_medicine.sort_values(by='ratio',ascending=False)
```

Out[ ]:

	Title	ratio
64590	Bandit NV Merlot (California)	21.50
126096	Cramele Recas 2011 UnWineD Pinot Grigio (Viile...	21.50
20484	Dancing Coyote 2015 White (Clarksburg)	21.25
1987	Felix Solis 2013 Flirty Bird Syrah (Vino de la...	21.25
110255	Bandit NV Merlot (California)	21.00
...	...	...
129844	Caparzo 2006 Doga delle Clavule (Morellino di...	NaN
129860	Quinta da Pacheca 2013 Pacheca Superior Red (D...	NaN
129863	Seacampo 2011 Reserva Red (Dão)	NaN
129893	Le Mandolare 2015 Corte Menini (Soave Classico)	NaN
129964	Domaine Ehrhart 2013 Domaine Saint-Rémy Herren...	NaN

129971 rows × 2 columns

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

In [ ]:

```
tropical = 0
fruity = 0
for sent in reviews['description']:
    if("tropical") in sent:
        tropical = tropical+1
    elif("fruity") in sent:
        fruity = fruity+1
descriptor_counts = pd.Series({'Tropical':tropical,'Fruity':fruity})
descriptor_counts
```

Out[ ]:
```
Tropical    3607
Fruity      8880
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.

In [ ]:
```
reviews['star rating'] = pd.cut(x=reviews['points'],bins=[0,85,94,100],labels=['1 star','2 star','3 star',])
canada = reviews[reviews['country'] =='Canada']
canada.loc[:,'star_rating'] = '3 star'
star_rating = pd.Series(reviews['star rating'])
star_rating
```

```
C:\Users\Rommel\AppData\Local\Temp\ipykernel_13564\19921504.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  canada.loc[:,'star_rating'] = '3 star'
```

Out[ ]:
```
0         2 star
1         2 star
2         2 star
3         2 star
4         2 star
           ...
129966    2 star
129967    2 star
129968    2 star
129969    2 star
129970    2 star
Name: star rating, Length: 129971, dtype: category
Categories (3, object): ['1 star' < '2 star' < '3 star']
```

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

In [ ]:
```
reviews['points'].dtype
```

Out[ ]:
```
dtype('int64')
```

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

In [ ]:
```
reviews['price'].isna().sum()
```

Out[ ]:
```
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

In [ ]:
```
reviews['region_1'].fillna('Unknown',inplace=True)
reviews['region_1'].value_counts()
```

Out[ ]:
```
region_1
Unknown                    21247
Napa Valley                 4480
Columbia Valley (WA)        4124
Russian River Valley        3091
California                  2629
                           ...
Lamezia                        1
Trentino Superiore             1
Grave del Friuli               1
Vin Santo di Carmignano        1
Paestum                        1
Name: count, Length: 1230, dtype: int64
```

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

In [ ]:
```
str_points = pd.Series(reviews['points'].astype('str'))
str_points
```

Out[ ]:
```
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.

In [ ]:
```
taster = pd.Series(reviews['taster_twitter_handle'].value_counts())
taster
```

Out[ ]:
```
taster_twitter_handle
@vossroger          25514
@wineschach         15134
@kerinokeefe        10776
@vboone              9537
@paulgwine           9532
@mattkettmann        6332
@JoeCz               5147
@wawinereport        4966
@gordone_cellars     4177
@AnneInVino          3685
@laurbuzz            1835
@suskostrzewa        1085
@worldwineguys       1005
@bkfiona               27
@winewchristina         6
Name: count, 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).

In [ ]:
```
best_rating_price = reviews.groupby('price')['points'].max().sort_index()
rating_price = pd.Series(best_rating_price)
rating_price
```

Out[ ]:
```
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.

In [ ]:
```
min_max = reviews.groupby('variety')['price'].agg(['min','max'])
min_max
```

Out[ ]:

variety	min	max
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).

In [ ]:
```
min_max_copy = min_max.copy()
min_max_copy.sort_values(by=['min','max'],ascending=False)
```

Out[ ]:

variety	min	max
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.

In [ ]:
```
avg_taster_pts = reviews.groupby('taster_name')['points'].mean()
ans = pd.Series(avg_taster_pts)
ans
```

Out[ ]:
```
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.

In [ ]:
```
country_var_num = reviews.groupby(['country','variety']).size().sort_values(ascending=False)
country_var_num = pd.Series(country_var_num)
country_var_num
```

Out[ ]:

```
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
```