GENERATE A WORD CLOUD USING WINE DATASET

In [1]:

```
# Start with loading all necessary libraries
import numpy as np
import pandas as pd
from os import path
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

In [2]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [5]:

```
df = pd.read_csv(r"winemag-data-130k-v2.csv", index_col=0)
```

In [6]:

```
df.head()
```

Out[6]:

	country	description	designation	points	price	province	region_1	region_2	taster_
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	O
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roge
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul G
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alex Pe
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 G

In [7]:

```
print("There are {} observations and {} features in this dataset. \n".format(df.shape[

print("There are {} types of wine in this dataset such as {}... \n".format(len(df.vari
                                                                            ", ".join(d

print("There are {} countries producing wine in this dataset such as {}... \n".format(
```

```
There are 129971 observations and 13 features in this dataset.

There are 708 types of wine in this dataset such as White Blend, Portugu
ese Red, Pinot Gris, Riesling, Pinot Noir...

There are 44 countries producing wine in this dataset such as Italy, Por
tugal, US, Spain, France...
```

In [9]:

```
df[["country", "description","points"]].head()
```

Out[9]:

	country	description	points
0	Italy	Aromas include tropical fruit, broom, brimston...	87
1	Portugal	This is ripe and fruity, a wine that is smooth...	87
2	US	Tart and snappy, the flavors of lime flesh and...	87
3	US	Pineapple rind, lemon pith and orange blossom ...	87
4	US	Much like the regular bottling from 2012, this...	87

In [10]:

```
# Groupby by country
country = df.groupby("country")

# Summary statistic of all countries
country.describe().head()
```

Out[10]:

	points									
	count	mean	std	min	25%	50%	75%	max	count	mea
country										
Argentina	3800.0	86.710263	3.179627	80.0	84.00	87.0	89.00	97.0	3756.0	24.5101
Armenia	2.0	87.500000	0.707107	87.0	87.25	87.5	87.75	88.0	2.0	14.5000
Australia	2329.0	88.580507	2.989900	80.0	87.00	89.0	91.00	100.0	2294.0	35.4376
Austria	3345.0	90.101345	2.499799	82.0	88.00	90.0	92.00	98.0	2799.0	30.7627
Bosnia and Herzegovina	2.0	86.500000	2.121320	85.0	85.75	86.5	87.25	88.0	2.0	12.5000

In [11]:

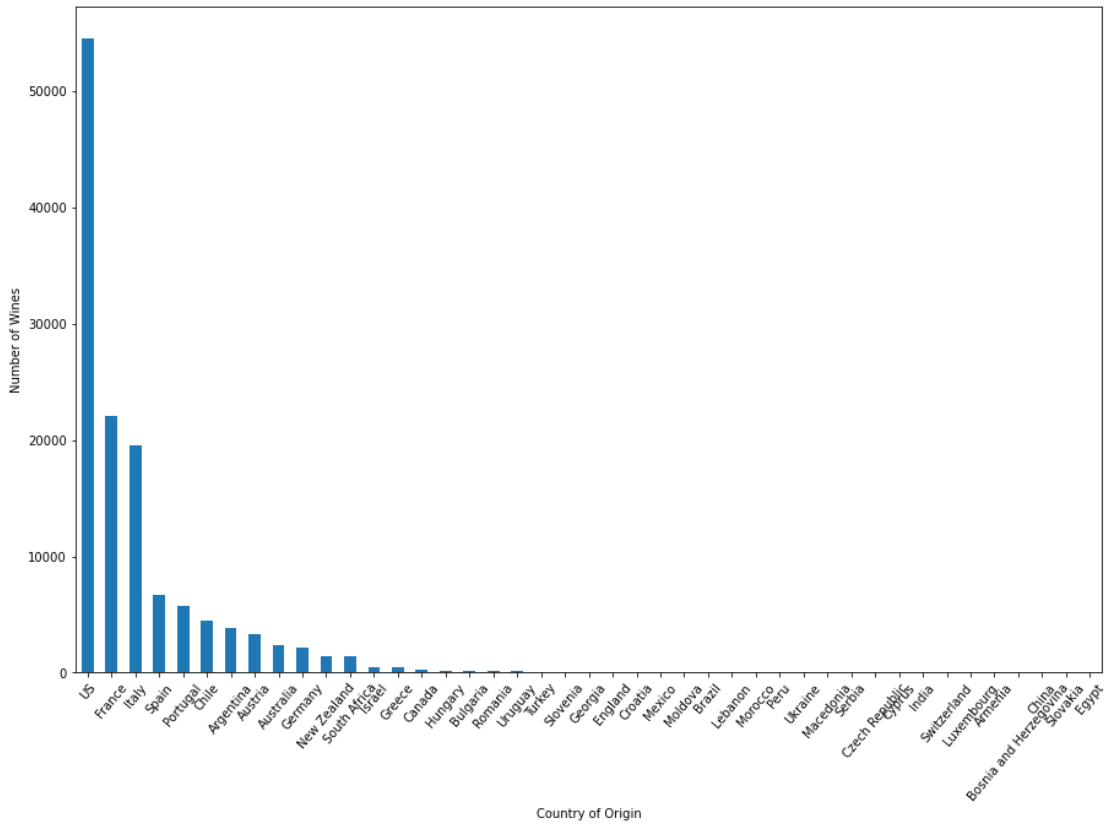
```
country.mean().sort_values(by="points",ascending=False).head()
```

Out[11]:

country	points	price
England	91.581081	51.681159
India	90.222222	13.333333
Austria	90.101345	30.762772
Germany	89.851732	42.257547
Canada	89.369650	35.712598

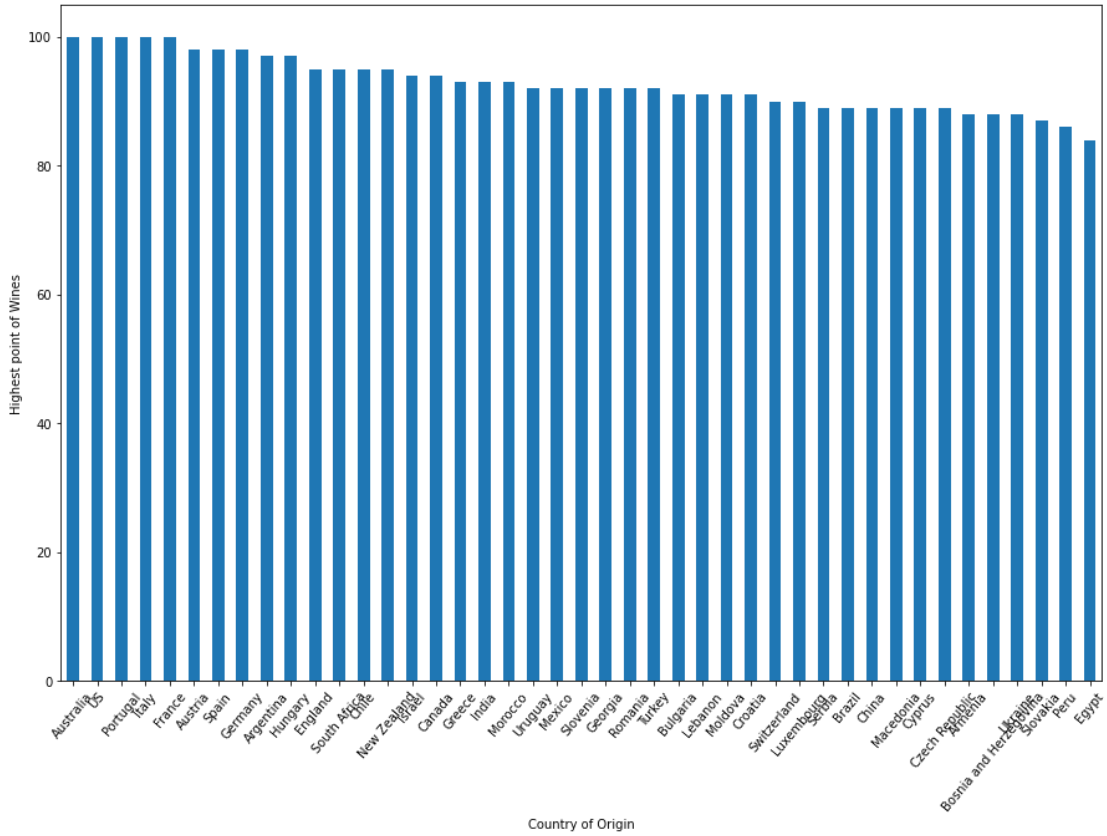
In [12]:

```
plt.figure(figsize=(15,10))
country.size().sort_values(ascending=False).plot.bar()
plt.xticks(rotation=50)
plt.xlabel("Country of Origin")
plt.ylabel("Number of Wines")
plt.show()
```



In [15]:

```
plt.figure(figsize=(15,10))
country.max().sort_values(by="points",ascending=False)["points"].plot.bar()
plt.xticks(rotation=50)
plt.xlabel("Country of Origin")
plt.ylabel("Highest point of Wines")
plt.show()
```

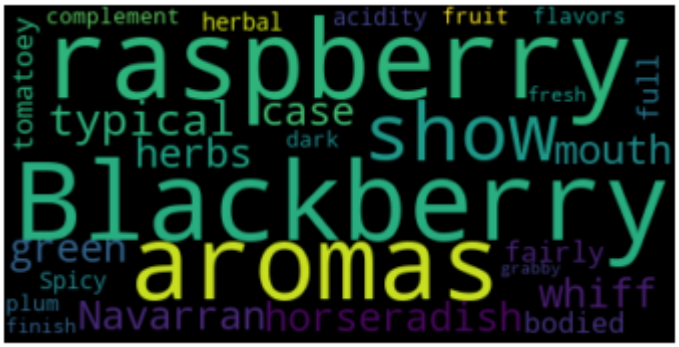


In [21]:

```
# Start with one review:
text = df.description[5]

# Create and generate a word cloud image:
wordcloud = WordCloud().generate(text)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



Changing optional word cloud arguments

In [18]:

```
# lower max_font_size, change the maximum number of word and lighten the background:
wordcloud = WordCloud(max_font_size=50, max_words=100, background_color="white").gener
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```


```
Using max_font_size here might not be a good idea.
It makes it more difficult to see the differences between word frequencies.
However, brightening the background makes the cloud easier to read
```

Combining data

In [19]:

```
text = " ".join(review for review in df.description)
print ("There are {} words in the combination of all review.".format(len(text)))
```

```
There are 31661073 words in the combination of all review.
```

In [20]:

```
# Create stopword list:
stopwords = set(STOPWORDS)
stopwords.update(["drink", "now", "wine", "flavor", "flavors"])

# Generate a word cloud image
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)

# Display the generated image:
# the matplotlib way:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

In [36]:

```
from PIL import Image
im = Image.open(r'wine.webp')
im
```

Out[36]:

In [37]:

```
wine_mask = np.array(im)
wine_mask
```

Out[37]:

```
array([[[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       ...,

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]]], dtype=uint8)
```

In [38]:

```
def transform_format(val):
    if val == 0:
        return 255
    else:
        return val
```

Named Entity Recognition using

1. POS
2. Text hero

In [25]:

```
def word_count_func(text):
    return len(text.split())
```

In [26]:

```
def norm_stemming_func(text):
    words = word_tokenize(text)
    text = ' '.join([PorterStemmer().stem(word) for word in words])
    return text
```

In [27]:

```
def norm_lemm_func(text):
    words = word_tokenize(text)
    text = ' '.join([WordNetLemmatizer().lemmatize(word) for word in words])
    return text
```

In [28]:

```
def norm_lemm_v_func(text):
    words = word_tokenize(text)
    text = ' '.join([WordNetLemmatizer().lemmatize(word, pos='v') for word in words])
    return text
```

In [29]:

```
def norm_lemm_a_func(text):
    words = word_tokenize(text)
    text = ' '.join([WordNetLemmatizer().lemmatize(word, pos='a') for word in words])
    return text
```

In [30]:

```
def get_wordnet_pos_func(word):
    tag = pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)
```

In [31]:

```
def norm_lemm_POS_tag_func(text):
    words = word_tokenize(text)
    text = ' '.join([WordNetLemmatizer().lemmatize(word, get_wordnet_pos_func(word)) f
    return text
```

In [32]:

```
def norm_lemm_v_a_func(text):
    words1 = word_tokenize(text)
    text1 = ' '.join([WordNetLemmatizer().lemmatize(word, pos='v') for word in words1]
    words2 = word_tokenize(text1)
    text2 = ' '.join([WordNetLemmatizer().lemmatize(word, pos='a') for word in words2]
    return text2
```

In [33]:

```
pos_ner_text = "Bill Gates founded Microsoft Corp. together with Paul Allen in 1975."
pos_ner_text
```

Out[33]:

```
'Bill Gates founded Microsoft Corp. together with Paul Allen in 1975.'
```