In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import warnings
from tqdm import tqdm_notebook
from gensim.models import Doc2Vec
import gensim
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from MulticoreTSNE import MulticoreTSNE as TSNE
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [2]:
```
data = pd.read_csv("BeerDataScienceProject.csv",encoding="latin1")
```

In [3]:
```
data['review_time'] = pd.to_datetime(data['review_time'], unit='s')
```

In [4]:
```
total = data.isnull().sum().sort_values(ascending=False)
percent = (data.isnull().sum()/data.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

Out[4]:

	Total	Percent
beer_ABV	20280	0.038346
review_text	119	0.000225
review_profileName	115	0.000217
review_time	0	0.000000
review_aroma	0	0.000000
review_taste	0	0.000000
review_overall	0	0.000000
review_palette	0	0.000000
review_appearance	0	0.000000
beer_style	0	0.000000
beer_name	0	0.000000
beer_beerId	0	0.000000
beer_beerId	0	0.000000

Dropping the null values

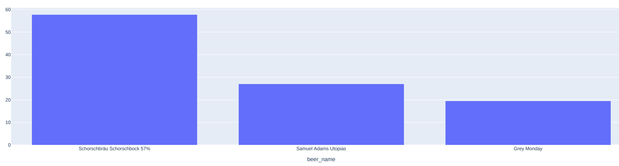
In [5]:
```
data.dropna(inplace=True)
data.reset_index(inplace=True)
```

Question 1

Rank top 3 Breweries which produce the strongest beers?

In [6]:
```
top_brewer_df = data.loc[data.groupby(["beer_brewerId"])["beer_ABV"].idxmax()].sort_values('beer_ABV',ascending=False)[:10]
```

In [7]:
```
fig = px.bar(top_brewer_df['beer_ABV','beer_name']][:3], y='beer_ABV', x='beer_name',
             title="Top 3 Breweries which produce the strongest beers")
fig.show()
```



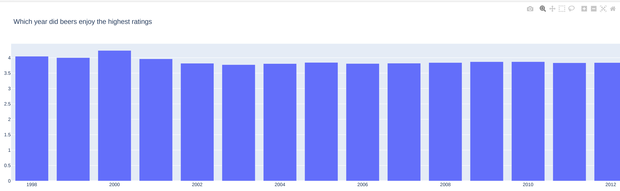
Question 2

Which year did beers enjoy the highest ratings?

In [8]:
```
data['year'] = pd.DatetimeIndex(pd.to_datetime(data['review_time'], unit='s')).year
```

In [9]:
```
top_year = data.sort_values('year',ascending=False).groupby('year') \
    .apply(lambda x: pd.DataFrame({
        'review_overall'          : [x['review_overall'].mean()],
    })
)
top_year.reset_index(inplace=True)
top_year.drop('level_1',axis=1,inplace=True)
top_year.sort_values('review_overall',ascending=False,inplace=True)
```

In [10]:
```
fig = px.bar(top_year, y='review_overall', x='year',
             title="Which year did beers enjoy the highest ratings")
fig.show()
```

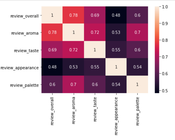


In year 2000 beer rating were highest

Question 3

Based on the user's ratings which factors are important among taste, aroma, appearance, and palette?

In [11]:
```
sns.heatmap(data[['review_overall','review_aroma','review_taste','review_appearance','review_palette']].corr(), annot=True)
plt.show()
```



Aroma is the most important aspect

Question 4

If you were to recommend 3 beers to your friends based on this data which ones will you recommend?

In [12]:
```
beer_recommendation = data.groupby('beer_name') \
    .apply(lambda x: pd.DataFrame({
        'mean_beer_ABV'           : [x['beer_ABV'].mean()],
        'mean_review_overall'     : [x['review_overall'].mean()],
    })
)
beer_recommendation.reset_index(inplace=True)
beer_recommendation.drop('level_1',axis=1,inplace=True)
beer_recommendation.sort_values(['mean_review_overall','mean_beer_ABV'],ascending=False)[:3]
```

Out[12]:

	beer_name	mean_beer_ABV	mean_review_overall
444	AleSmith Speedway Stout - Oak Aged	12.0	5.0
9624	Pilot Series Imperial Sweet Stout - Palm Ridge...	12.0	5.0
1330	Bees Knees Barleywine	11.2	5.0

Question 5

Which Beer style seems to be the favorite based on reviews written by users?

In [13]:
```
def sentiment_scores(sentence):
    sid_obj = SentimentIntensityAnalyzer()
    sentiment_dict = sid_obj.polarity_scores(sentence)
    return(sentiment_dict['compound'])
```

In [ ]:
```
data['polarity'] = ""
```

In [ ]:
```
for row in tqdm_notebook(range(len(data))):
    data['polarity'][row] = sentiment_scores(data['review_text'][row])
```

In [15]:
```
favourite_beer_style = data.groupby('beer_style') \
  .apply(lambda x: pd.DataFrame({
    'mean_polarity_score'     : [x['polarity'].mean()],
    'mean_review_overall'     : [x['review_overall'].mean()],
  })
)
favourite_beer_style.reset_index(inplace=True)
favourite_beer_style.drop('level_1',axis=1,inplace=True)
```

In [16]:
```
favourite_beer_style.sort_values('mean_polarity_score',ascending=False)[:1]
```

Out[16]:

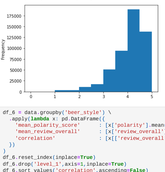
	beer_style	mean_polarity_score	mean_review_overall
86	Quadrupel (Quad)	0.861593	4.052675

Question 6

How does written review compare to overall review score for the beer styles?

In [18]:
```
data['review_overall'].plot.hist()
```

Out[18]:
```
<AxesSubplot:ylabel="Frequency">
```



In [19]:
```
df_6 = data.groupby('beer_style') \
  .apply(lambda x: pd.DataFrame({
    'mean_polarity_score'     : [x['polarity'].mean()],
    'mean_review_overall'     : [x['review_overall'].mean()],
    'correlation'             : [x['review_overall','polarity']].corr()['polarity'][0]
  })
)
df_6.reset_index(inplace=True)
df_6.drop('level_1',axis=1,inplace=True)
df_6.sort_values('correlation',ascending=False)
```

Out[19]:

	beer_style	mean_polarity_score	mean_review_overall	correlation
56	Faro	0.749400	3.820755	0.596201
62	Gose	0.699045	3.558824	0.471446
64	Happoshu	0.522789	2.818182	0.465503
61	German Pilsener	0.754210	3.821913	0.449080
89	Munich Dunkel Lager	0.730196	3.733333	0.449055
...	...	...	...	...
11	American Double / Imperial Stout	0.851921	4.100595	0.266903
58	Flanders Red Ale	0.857309	3.966391	0.260216
86	Quadrupel (Quad)	0.861593	4.052675	0.253458
72	Kvass	0.839459	4.025773	0.226526
41	Eisbock	0.860557	4.082474	0.107067

104 rows × 4 columns

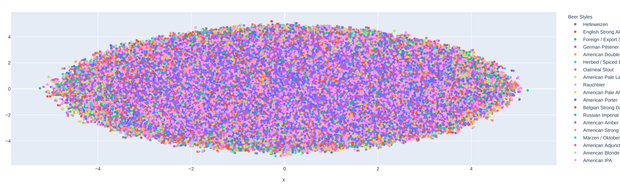
Question 7

How do find similar beer drinkers by using written reviews only?

In [ ]:
```
def tagged_document(list_of_list_of_words):
    for i, list_of_words in enumerate(list_of_list_of_words):
        yield gensim.models.doc2vec.TaggedDocument(list_of_words, [i])
```

In [ ]:
```
tagged_review = tagged_document(list(data['review_text']))
model = gensim.models.doc2vec.Doc2Vec(vector_size=40, min_count=2, epochs=30)
model.build_vocab(tagged_review)
model.train(tagged_review, total_examples=model.corpus_count, epochs=model.epochs)
tsne_model = TSNE(n_jobs=4,n_components=2)
tsne_d2v = tsne_model.fit_transform(model.docvecs.vectors_docs)
```

In [24]:
```
fig = px.scatter(
    tsne_d2v, x='x', y='y',
    color=data.beer_style, labels={'color': 'Beer Styles'}
)
fig.show()
```



Though the above graph doesn't give much insight but upon zooming in it will put similar reviews near by making them small clusters which can then recommended to other similar beer drinkers

Note: Due to high computational time we couldn't use higher embedding size which would have given better insights about similar beer drinkers

In [ ]:
```
```