In [4]:
```
import os
import pandas as pd
import numpy as np
import pandas as pd
from pandas_profiling import ProfileReport
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import seaborn as sns
```

In [65]:
```
import plotly.io as pio
pio.renderers.default='notebook'
```

In [5]:
```
beerData = pd.read_csv("BeerDataScienceProject.csv",encoding='latin-1')
```

In [6]:
```
beerData
```

Out[6]:

	beer_ABV	beer_beerId	beer_brewerId	beer_name	beer_style	review_appearance	review_palette	review_overall	review_taste	review_pr
0	5.0	47986	10325	Sausa Weizen	Hefeweizen	2.5	2.0	1.5	1.5	
1	6.2	48213	10325	Red Moon	English Strong Ale	3.0	2.5	3.0	3.0	
2	6.5	48215	10325	Black Horse Black Beer	Foreign / Export Stout	3.0	2.5	3.0	3.0	
3	5.0	47969	10325	Sausa Pils	German Pilsener	3.5	3.0	3.0	2.5	
4	7.7	64883	1075	Cauldron DIPA	American Double / Imperial IPA	4.0	4.5	4.0	4.0	johnr
...	...	...	...	...	...	...	...	...	...	
528865	NaN	4032	3340	Dinkel Acker Dark	Munich Dunkel Lager	4.0	3.0	4.0	3.5	orangei
528866	NaN	4032	3340	Dinkel Acker Dark	Munich Dunkel Lager	4.0	3.5	3.0	3.0	I
528867	NaN	4032	3340	Dinkel Acker Dark	Munich Dunkel Lager	4.0	4.0	4.5	4.0	
528868	NaN	4032	3340	Dinkel Acker Dark	Munich Dunkel Lager	4.0	3.0	4.0	4.0	
528869	NaN	4032	3340	Dinkel Acker Dark	Munich Dunkel Lager	4.0	4.0	4.0	4.0	j

528870 rows × 13 columns

In [7]:
```
beerData.shape
```

Out[7]:
```
(528870, 13)
```

In [8]:
```
profile = ProfileReport(beerData, title="Beer Data Feature Profiling",explorative=True)
```

In [9]:
```
profile.to_notebook_iframe()
```

`beer_style` has a high cardinality: 104 distinct values	**High cardinality**
`review_profileName` has a high cardinality: 22800 distinct values	**High cardinality**
`review_text` has a high cardinality: 528371 distinct values	**High cardinality**
`beer_ABV` has 20280 (3.8%) missing values	**Missing**
`review_text` is uniformly distributed	**Uniform**

Reproduction

Analysis started	2021-09-08 09:29:56.062606
Analysis finished	2021-09-08 09:31:32.777834
Duration	1 minute and 36.72 seconds
Software version	pandas-profiling v2.10.0 (https://github.com/pandas-profiling/pandas-profiling)
Download configuration	config.yaml (data:text/plain;charset=utf-8,title%3A%20Beer%20Data%20Feature%20Profiling%0Amemory_de %20%5B%27true%27%2C%20%27false%27%5D%0Adataset%3A%0A%20%20%20%20description%3A%2(

Variables

beer_ABV
Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	283
Distinct (%)	0.1%

Mean	7.017441593
Minimum	0.01

MISSING

Missing	20280	**Maximum**	57.7
Missing (%)	3.8%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%

In [10]:
```
profile.to_file("BeerReviewFeatureProfiling.html")
```

In [11]:
```
beerData.describe()
```

Out[11]:

	beer_ABV	beer_beerId	beer_brewerId	review_appearance	review_palette	review_overall	review_taste	review_aroma	review_tir
count	508590.000000	528870.000000	528870.000000	528870.000000	528870.000000	528870.000000	528870.000000	528870.000000	5.288700e+
mean	7.017442	22098.466016	2598.423429	3.864522	3.758926	3.833197	3.765993	3.817350	1.224885e+
std	2.204460	22158.284352	5281.805350	0.604010	0.685335	0.709962	0.669018	0.718903	7.605600e+
min	0.010000	3.000000	1.000000	0.000000	1.000000	0.000000	1.000000	1.000000	8.843904e+
25%	5.300000	1745.000000	132.000000	3.500000	3.500000	3.500000	3.500000	3.500000	1.174613e+
50%	6.500000	14368.000000	394.000000	4.000000	4.000000	4.000000	4.000000	4.000000	1.240366e+
75%	8.500000	40528.000000	1475.000000	4.000000	4.000000	4.500000	4.000000	4.500000	1.288560e+
max	57.700000	77310.000000	27980.000000	5.000000	5.000000	5.000000	5.000000	5.000000	1.326277e+

Null Values Counts

In [12]:
```
#count null values
beerData.isna().sum()
```

Out[12]:
```
beer_ABV              20280
beer_beerId               0
beer_brewerId             0
beer_name                 0
beer_style                0
review_appearance         0
review_palette            0
review_overall            0
review_taste              0
review_profileName      115
review_aroma              0
review_text             119
review_time               0
dtype: int64
```

In [13]:
```
# Percent of data missing
print("Percent Null Values from Total", round(beerData.isna().sum().max() / len(beerData) * 100, 2),"%")
```

```
Percent Null Values from Total 3.83 %
```

Most of the null value comes from beer_ABV column which is only 3.8%. We can drop all the Null Values

In [14]:
```
beerData = beerData.dropna()
beerData.isna().sum()
```

Out[14]:
```
beer_ABV              0
beer_beerId           0
beer_brewerId         0
beer_name             0
beer_style            0
review_appearance     0
review_palette        0
review_overall        0
review_taste          0
review_profileName    0
review_aroma          0
review_text           0
review_time           0
dtype: int64
```

In [15]:
```
beerData.shape
```

Out[15]:
```
(508358, 13)
```

There are 3 records values having 0 in review_appearance columns and 3 records having 0 in review_overall columns

Ideally rating should lie between 1 and 5 so droping such instances having review ratings less than 0

In [16]:
```
beerData = beerData[(beerData['review_overall'] >= 1) | (beerData['review_appearance'] >=1)]
```

In [17]:
```
beerData.describe()
```

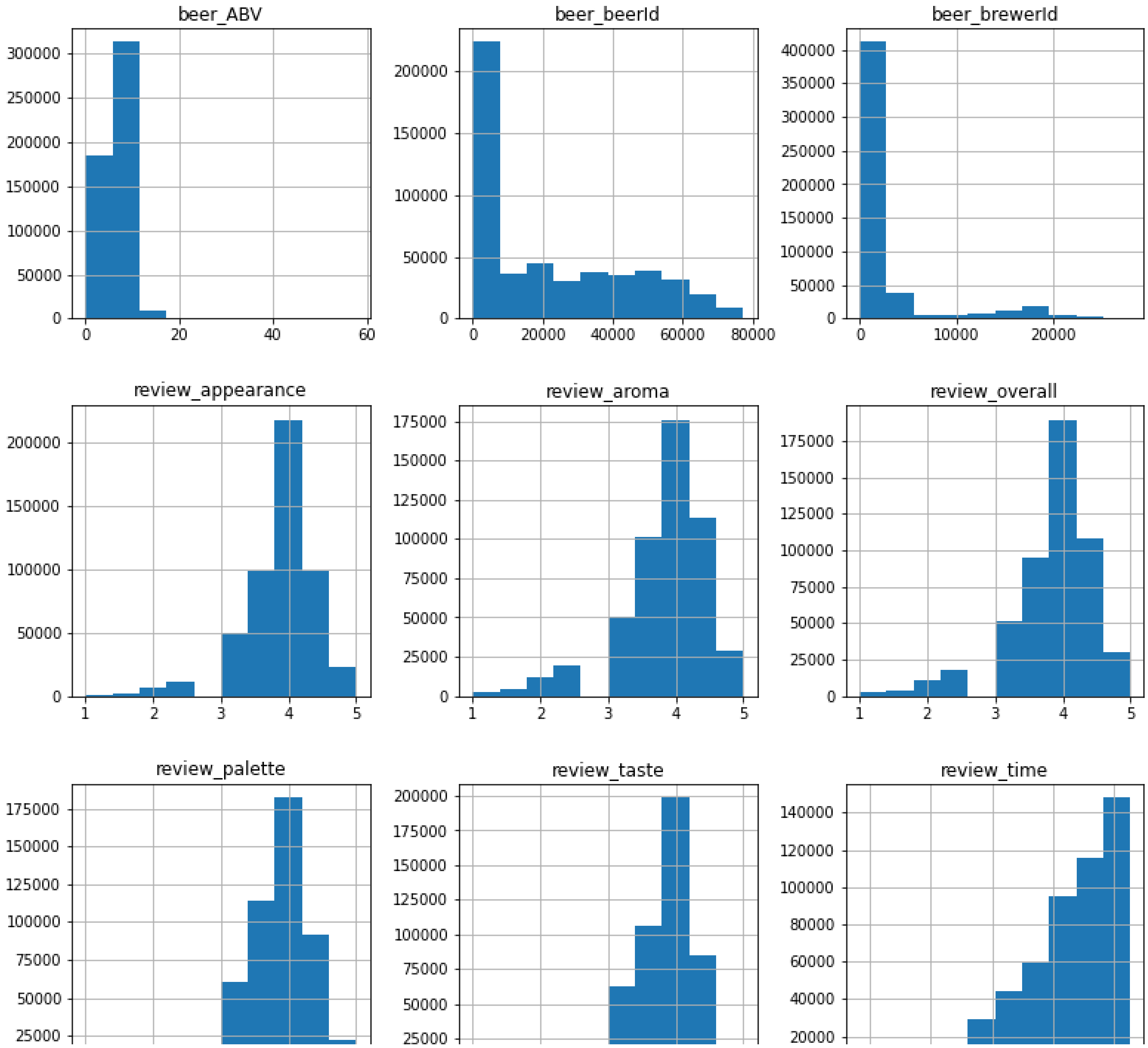
Out[17]:

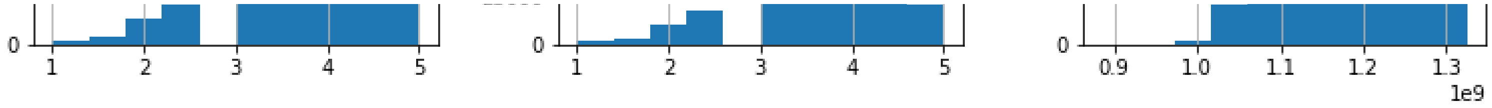
	beer_ABV	beer_beerId	beer_brewerId	review_appearance	review_palette	review_overall	review_taste	review_aroma	review_tir
count	508355.000000	508355.000000	508355.000000	508355.000000	508355.000000	508355.000000	508355.000000	508355.000000	5.083550e+
mean	7.017418	21824.227168	2534.279824	3.872699	3.768997	3.840828	3.775335	3.827657	1.226176e+
std	2.204522	22124.991097	5237.858572	0.601692	0.682351	0.706348	0.665578	0.715110	7.530715e+
min	0.010000	5.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	8.843904e+
25%	5.300000	1673.000000	132.000000	3.500000	3.500000	3.500000	3.500000	3.500000	1.177202e+
50%	6.500000	13850.000000	392.000000	4.000000	4.000000	4.000000	4.000000	4.000000	1.241505e+
75%	8.500000	40418.000000	1315.000000	4.000000	4.000000	4.500000	4.000000	4.500000	1.289074e+
max	57.700000	77310.000000	27980.000000	5.000000	5.000000	5.000000	5.000000	5.000000	1.326277e+

In [18]:
```
beerData.shape
```

Out[18]: (508355, 13)

In [19]:
```
# Histogram of all numeric features
beerData.hist(figsize=(13,13))
plt.show()
```


beer_ABV
300000
250000
200000
150000
100000
50000
0
0
20
40
60
beer_beerId
200000
150000
100000
50000
0
0
20000
40000
60000
80000
beer_brewerId
400000
350000
300000
250000
200000
150000
100000
50000
0
0
10000
20000
review_appearance
200000
150000
100000
50000
0
1
2
3
4
5
review_aroma
175000
150000
125000
100000
75000
50000
25000
0
1
2
3
4
5
review_overall
175000
150000
125000
100000
75000
50000
25000
0
1
2
3
4
5
review_palette
175000
150000
125000
100000
75000
50000
25000
review_taste
200000
175000
150000
125000
100000
75000
50000
25000
review_time
140000
120000
100000
80000
60000
40000
20000



beer_abv - Right Skewed - Most of beers have less than 20% ABV

review_appearance - Normal Distribution - Most beers are rated between 3.5 and 4.5

review_aroma - Normal Distribution - Most beers are rated between 3.5 and 4.5

review_overall - Normal Distribution - Most beers are rated between 3.5 and 4.5

review_palette - Normal Distribution - Most beers rated between 3.5 and 4.5

review_taste - Normal Distribution - Most beers rated between 3.5 and 4.5

Correlation Analysis

In [20]:
```
beerData.corr()
```

Out[20]:

	beer_ABV	beer_beerId	beer_brewerId	review_appearance	review_palette	review_overall	review_taste	review_aroma	review_
beer_ABV	1.000000	0.218121	0.078576	0.252458	0.319381	0.119462	0.269098	0.271610	0.14
beer_beerId	0.218121	1.000000	0.460927	0.051194	0.061337	-0.011662	0.036274	0.015108	0.46
beer_brewerId	0.078576	0.460927	1.000000	-0.004623	0.017379	-0.014272	-0.002699	-0.009604	0.24
review_appearance	0.252458	0.051194	-0.004623	1.000000	0.544752	0.482986	0.551972	0.531182	0.04
review_palette	0.319381	0.061337	0.017379	0.544752	1.000000	0.598069	0.600845	0.703430	0.05
review_overall	0.119462	-0.011662	-0.014272	0.482986	0.598069	1.000000	0.689277	0.780311	0.01
review_taste	0.269098	0.036274	-0.002699	0.551972	0.600845	0.689277	1.000000	0.722729	0.04
review_aroma	0.271610	0.015108	-0.009604	0.531182	0.703430	0.780311	0.722729	1.000000	0.02
review_time	0.143439	0.460089	0.245981	0.046440	0.050506	0.018669	0.044155	0.029316	1.00

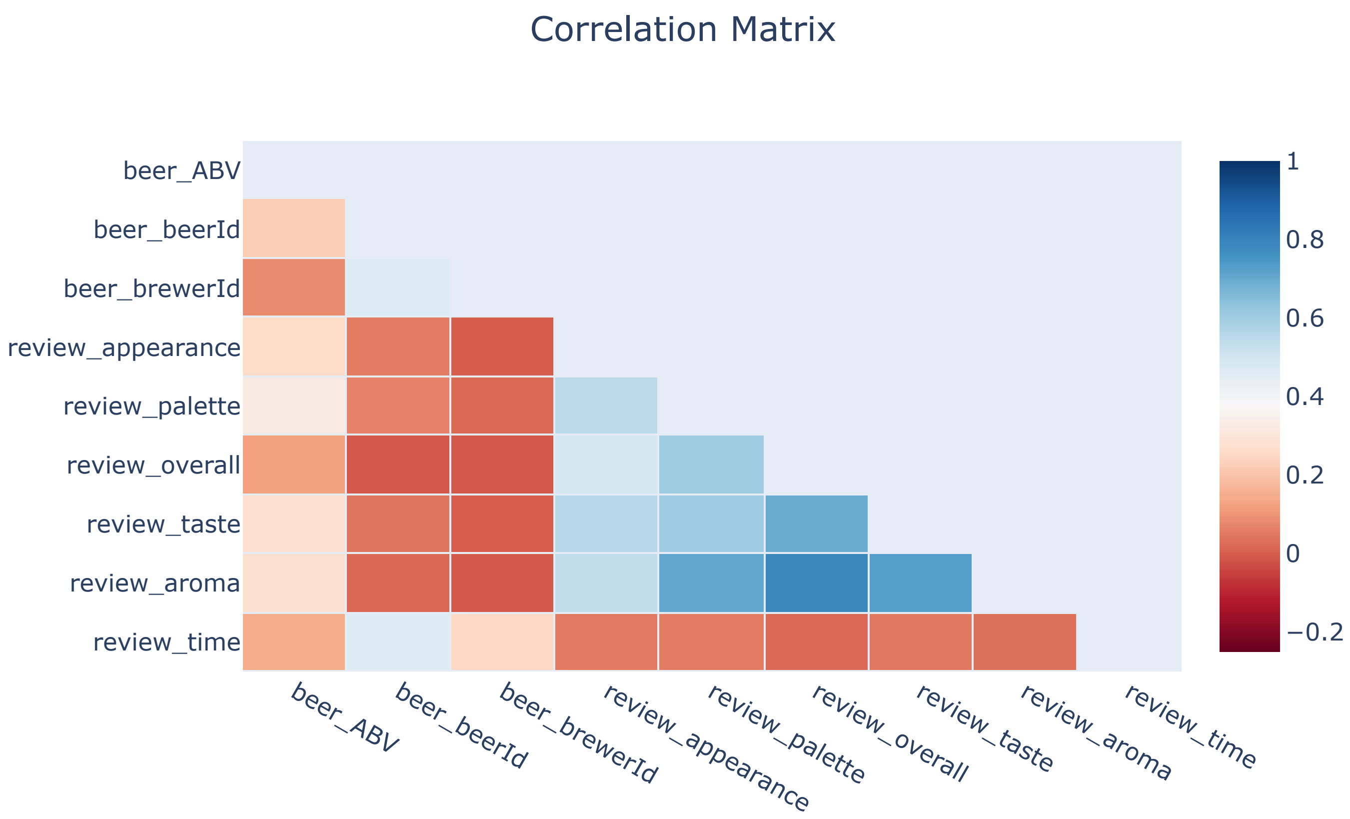
In [21]:
```
r = beerData.corr()
mask = np.triu(np.ones_like(r, dtype=bool))
rLT = r.mask(mask)

heat = go.Heatmap(
    z = rLT,
    x = rLT.columns.values,
    y = rLT.columns.values,
    zmin = - 0.25, # Sets the lower bound of the color domain
    zmax = 1,
    xgap = 1, # Sets the horizontal gap (in pixels) between bricks
    ygap = 1,
    colorscale = 'RdBu'
)

title = 'Correlation Matrix'

layout = go.Layout(
    title_text=title,
    title_x=0.5,
    autosize=False,
    xaxis_showgrid=False,
    yaxis_showgrid=False,
    yaxis_autorange='reversed'
)

fig=go.Figure(data=[heat], layout=layout)
fig.show()
```



review_aroma have high positive correlation with review_overall(0.78), review_taste(0.72) and review_palette(0.70)

Question 1: Rank top 3 Breweries which produce the strongest beers?

In [22]:
```
data = beerData.groupby(['beer_brewerId'])['beer_ABV'].mean()
strong = pd.DataFrame(data.reset_index())

strong.columns = ['beer_brewerId','beer_abv_mean']

strongestMean = strong.sort_values(by=['beer_abv_mean'], ascending=False).head(5)

strongestMean
```

Out[22]:

	beer_brewerId	beer_abv_mean
699	6513	19.228824
165	736	13.750000
1466	24215	12.466667
8	36	12.445860
789	8540	11.750000

In [23]:
```
data = beerData.groupby(['beer_brewerId'])['beer_ABV'].median()
strong = pd.DataFrame(data.reset_index())

strong.columns = ['beer_brewerId','beer_abv_median']

strongestMedian = strong.sort_values(by=['beer_abv_median'], ascending=False).head(5)

strongestMedian
```

Out[23]:

	beer_brewerId	beer_abv_median
165	736	14.0
636	5562	13.2
699	6513	13.0
8	36	13.0
435	2830	12.0

Since the distribution of beer_ABV is Right Skewed median would give us better insights as compared Average or mean as mean is very sensitive to skewness of the data

Top 3 Breweries which produce strongest beers are

- ### 1. brewer_id = 736
- ### 2. brewer_id = 5562
- ### 3. brewer_id = 6513 and 36

Question 2: Which year did beers enjoy the highest ratings?

In [24]:
```
beerData['review_year'] = pd.DatetimeIndex(pd.to_datetime(beerData['review_time'], unit='s')).year
```

In [25]:
```
ratings = beerData.groupby('review_year').agg({
                                            'review_overall': np.mean,
                                            'review_aroma': np.mean,
                                            'review_appearance': np.mean,
                                            'review_palette': np.mean,
                                            'review_taste': np.mean
                                        })
```

In [26]:
```
beerData.groupby("review_year").size()
```

Out[26]:
```
review_year
1998        11
1999        10
2000        30
2001       538
2002      6840
2003     16584
2004     21291
2005     27803
2006     40708
2007     44484
2008     66578
2009     81192
2010     91342
2011    107871
2012      3073
dtype: int64
```

In [27]:
```
hightRatings = beerData[(beerData["review_overall"] == 5) & (beerData["review_appearance"] ==5) &  (beerData["review_p
alette"] ==5) & (beerData["review_taste"] ==5)  & (beerData["review_aroma"] ==5)]
```

In [28]:
```
hightRatings = hightRatings.reset_index()
```

In [29]:
```
hightRatings
```

Out[29]:

	index	beer_ABV	beer_beerId	beer_brewerId	beer_name	beer_style	review_appearance	review_palette	review_overall	review_taste	rev
0	433	6.1	10784	1075	Caldera IPA	American IPA	5.0	5.0	5.0	5.0	
1	1712	5.3	16491	1454	T.J.'s Best Bitter	English Bitter	5.0	5.0	5.0	5.0	
2	1751	5.5	15660	1454	Wobbly Bob APA	American Pale Ale (APA)	5.0	5.0	5.0	5.0	
3	2113	4.5	1557	577	Black Cuillin	Scottish Ale	5.0	5.0	5.0	5.0	
4	2380	4.8	61800	16859	Blonde Ambition	American Blonde Ale	5.0	5.0	5.0	5.0	
...	...	...	...	...	...	...	...	...	...	...	
1968	526508	8.0	773	283	Goudenband	Flanders Oud Bruin	5.0	5.0	5.0	5.0	
1969	526584	8.0	773	283	Goudenband	Flanders Oud Bruin	5.0	5.0	5.0	5.0	
1970	526594	8.0	773	283	Goudenband	Flanders Oud Bruin	5.0	5.0	5.0	5.0	

	index	beer_ABV	beer_beerId	beer_brewerId	beer_name	beer_style	review_appearance	review_palette	review_overall	review_taste	rev
1971	527186	4.3	1751	646	O'Hara's Irish Stout	Irish Dry Stout	5.0	5.0	5.0	5.0	
1972	527270	4.3	1751	646	O'Hara's Irish Stout	Irish Dry Stout	5.0	5.0	5.0	5.0	

1973 rows × 15 columns

In [30]:

```
hightRatings.groupby("review_year").size()
```

Out[30]:

```
review_year
2000      1
2001     37
2002     60
2003     65
2004     75
2005     93
2006    136
2007    142
2008    254
2009    285
2010    333
2011    469
2012     23
dtype: int64
```

In [31]:
```
ratings.sort_values("review_overall",ascending = False)
```

Out[31]:

review_year	review_overall	review_aroma	review_appearance	review_palette	review_taste
2000	4.233333	4.233333	3.916667	3.933333	4.000000
1998	4.045455	4.090909	3.500000	3.681818	3.818182
1999	4.000000	4.050000	3.650000	3.800000	3.900000
2001	3.961896	3.966543	3.907063	3.717472	3.804833
2010	3.869430	3.854322	3.902159	3.803015	3.812573
2009	3.868749	3.856242	3.898309	3.797874	3.805344
2005	3.844657	3.825235	3.858612	3.753624	3.766500
2008	3.840345	3.830875	3.863904	3.765335	3.768820
2012	3.839082	3.847543	3.904328	3.805402	3.803287
2011	3.833394	3.833533	3.896038	3.795381	3.791339
2007	3.819879	3.798680	3.824251	3.721338	3.737310
2002	3.819225	3.785234	3.818567	3.690205	3.705117
2006	3.809104	3.785177	3.834996	3.716923	3.725877
2004	3.806632	3.784956	3.823940	3.713752	3.714316
2003	3.772793	3.738845	3.792933	3.662868	3.684244

- ### Year 2000 has highest mean values of all the rating parameters
- ### Year 2011 has highest number(469) of beers having 5 rating in all the parameters

Question 3 : Based on the user’s ratings which factors are important among taste, aroma, appearance, and palette?

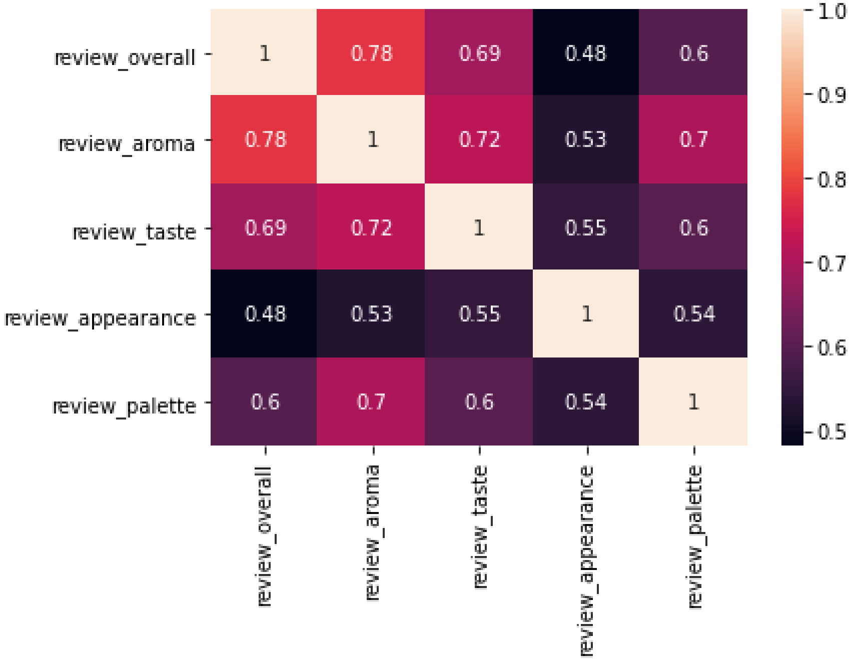
Here we can say that what factors amongst taste, aroma, appearance and palette would influence overall ratings

In [32]:
```
factors = beerData[['review_overall','review_aroma','review_taste','review_appearance','review_palette']]
factors.head()
```

Out[32]:

	review_overall	review_aroma	review_taste	review_appearance	review_palette
0	1.5	1.5	1.5	2.5	2.0
1	3.0	3.0	3.0	3.0	2.5
2	3.0	3.0	3.0	3.0	2.5
3	3.0	3.0	2.5	3.5	3.0
4	4.0	4.5	4.0	4.0	4.5

In [33]:
```
sns.heatmap(factors.corr(), annot=True)
plt.show()
```



Overall Ratings has high correlation with Aroma, Taste and Palette

Lets check feature importance by running Random Forest Regressor

In [34]:
```
features = factors.drop(["review_overall"], axis=1).columns
features
```

Out[34]:
```
Index(['review_aroma', 'review_taste', 'review_appearance', 'review_palette'], dtype='object')
```

In [29]:
```
from sklearn.ensemble import RandomForestRegressor
rnd_clf = RandomForestRegressor(n_estimators=500, n_jobs=-1, random_state=42)
rnd_clf.fit(factors[features].values, factors["review_overall"].values)
feature_importance = rnd_clf.feature_importances_
```

In [30]:
```
for name, score in zip(features, feature_importance):
    print(name, score)
```

```
review_aroma 0.9250195544500398
review_taste 0.05631101139465394
review_appearance 0.009838166457482964
review_palette 0.00883126769782329
```

Aroma is the most important factor which would influence the overall ratings

Question 4: If you were to recommend 3 beers to your friends based on this data which ones will you recommend?

If I have to recommend a beer to someone who is new to the taste of beer I would first find out which beers are popular in the industry number of reviews recieved can be good parameter to find out the popularity of the beer. After finding the popular beers I would pick those beer which are having highest overall ratings. Mass pleasing choice can be good choice to do a cold start.

In [35]:
```
# Pivot table with overall rating
top_3_rec = beerData[['beer_name','review_overall']]\
    .pivot_table(index="beer_name", aggfunc=("count",'mean','median'))\
    .dropna()

# Rename columns and flatten pivot table
top_3_rec.columns = top_3_rec.columns.to_series().str.join('_')
top_3_rec.reset_index(inplace=True)

# Filter for highest rated beers
top_3_rec = top_3_rec.query('review_overall_count >= 2000')\
  .sort_values('review_overall_mean', ascending=False)\
  .head(3)

# # Check it out
top_3_rec
```

Out[35]:

	beer_name	review_overall_count	review_overall_mean	review_overall_median
4614	Founders Breakfast Stout	2501	4.354658	4.5
12767	Trappistes Rochefort 10	2170	4.339401	4.5
7206	La Fin Du Monde	2480	4.297581	4.5

Question 5 : Which Beer style seems to be the favorite based on reviews written by users?

In [36]:
```
import spacy
from spacy.lang.en.stop_words import STOP_WORDS
import re
```

In [37]:
```
import contractions

print(contractions.fix("that'd"))
```

```
that would
```

Review Text data preprocessing

In [38]:
```
#convert to lowercase
beerData['review_text'] = beerData['review_text'].apply(lambda x: x.lower())

# Contraction to Expansion using contractions library

def cont_to_exp(x):
    if type(x) is str:
        x = contractions.fix(x)
        return x
    else:
        return x

beerData['review_text'] = beerData['review_text'].apply(lambda x: cont_to_exp(x))

#Special Chars removal or punctuation removal
beerData['review_text'] = beerData['review_text'].apply(lambda x: re.sub('[^a-z ]+', '', x))

#Remove multiple spaces
beerData['review_text'] = beerData['review_text'].apply(lambda x: " ".join(x.split()))

#Remove Stop Words using Spacy library
beerData['review_text'] = beerData['review_text'].apply(lambda x: " ".join([t for t in x.split() if t not in STOP_WORD
S]))
```

In [39]:
```
beerData.head()
```

Out[39]:

	beer_ABV	beer_beerId	beer_brewerId	beer_name	beer_style	review_appearance	review_palette	review_overall	review_taste	review_profileN
0	5.0	47986	10325	Sausa Weizen	Hefeweizen	2.5	2.0	1.5	1.5	stc
1	6.2	48213	10325	Red Moon	English Strong Ale	3.0	2.5	3.0	3.0	stc
2	6.5	48215	10325	Black Horse Black Beer	Foreign / Export Stout	3.0	2.5	3.0	3.0	stc
3	5.0	47969	10325	Sausa Pils	German Pilsener	3.5	3.0	3.0	2.5	stc
4	7.7	64883	1075	Cauldron DIPA	American Double / Imperial IPA	4.0	4.5	4.0	4.0	johnmichae

Using nltk package for sentiment analysis on review text data

- #### Polarity close to 1 means positive review
- #### Polarity close to -1 means negative review

In [40]:
```
import nltk
nltk.download("vader_lexicon")
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\abhishek.jadhav1\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [41]:
```
beerData['polarity_score'] = beerData['review_text'].apply(lambda x: sia.polarity_scores(x)['compound'])
```

In [42]:
```
beerData.head()
```

Out[42]:

	beer_ABV	beer_beerId	beer_brewerId	beer_name	beer_style	review_appearance	review_palette	review_overall	review_taste	review_profileN
0	5.0	47986	10325	Sausa Weizen	Hefeweizen	2.5	2.0	1.5	1.5	stc
1	6.2	48213	10325	Red Moon	English Strong Ale	3.0	2.5	3.0	3.0	stc
2	6.5	48215	10325	Black Horse Black Beer	Foreign / Export Stout	3.0	2.5	3.0	3.0	stc
3	5.0	47969	10325	Sausa Pils	German Pilsener	3.5	3.0	3.0	2.5	stc
4	7.7	64883	1075	Cauldron DIPA	American Double / Imperial IPA	4.0	4.5	4.0	4.0	johnmichae

In [43]:
```
beerData.groupby('beer_style')['polarity_score'].mean().sort_values(ascending=False)[0:10]
```

Out[43]:
```
beer_style
Eisbock                             0.898516
Braggot                             0.888858
Quadrupel (Quad)                    0.887129
Flanders Red Ale                    0.884769
Wheatwine                           0.880095
Dortmunder / Export Lager           0.877960
American Double / Imperial Stout    0.874006
Roggenbier                          0.869019
American Wild Ale                   0.868759
Old Ale                             0.865855
Name: polarity_score, dtype: float64
```

Eisbock Beer style has hightest polarity score.

In [44]:
```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

In [45]:
```
EisbockTxt = beerData[beerData.beer_style == "Eisbock"].review_text
```

In [46]:
```
text = " ".join(review for review in EisbockTxt)
print ("There are {} words in the combination of all review.".format(len(text)))
```

```
There are 92146 words in the combination of all review.
```

In [47]:
```
stopwords = set(STOPWORDS)
stopwords.update(["drink", "alcohol", "wine", "beer","eisbock"])
```

In [112]:
```
stopwords
```

```
Out[112]: {'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'alcohol',
 'all',
 'also',
 'am',
 'an',
 'and',
 'any',
 'are',
 "aren't",
 'as',
 'at',
 'be',
 'because',
 'been',
 'beer',
 'before',
 'being',
 'below',
 'between',
 'both',
 'but',
 'by',
 'can',
 "can't",
 'cannot',
 'com',
 'could',
 "couldn't",
 'did',
 "didn't",
 'do',
 'does',
 "doesn't",
 'doing',
 "don't",
```

```
'down',
'drink',
'during',
'each',
'eisbock',
'else',
'ever',
'few',
'for',
'from',
'further',
'get',
'had',
"hadn't",
'has',
"hasn't",
'have',
"haven't",
'having',
'he',
"he'd",
"he'll",
"he's",
'hence',
'her',
'here',
"here's",
'hers',
'herself',
'him',
'himself',
'his',
'how',
"how's",
'however',
'http',
'i',
"i'd",
"i'll",
"i'm",
"i've",
'if',
```

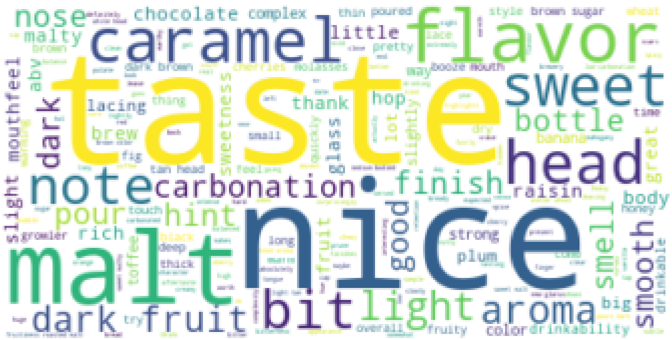
```
             'in',
             'into',
             'is',
             "isn't",
             'it',
             "it's",
             'its',
             'itself',
             'just',
             'k',
             "let's",
             'like',
             'me',
             'more',
             'most',
             "mustn't",
             'my',
             'myself',
             'no',
             'nor',
             'not',
             'of',
             'off',
             'on',
             'once',
             'only',
             'or',
             'other',
             'otherwise',
             'ought',
             'our',
             'ours',
             'ourselves',
             'out',
             'over',
             'own',
             'r',
             'same',
             'shall',
             "shan't",
             'she',
             "she'd",
```

```
                     "she'll",
                     "she's",
                     'should',
                     "shouldn't",
                     'since',
                     'so',
                     'some',
                     'such',
                     'than',
                     'that',
                     "that's",
                     'the',
                     'their',
                     'theirs',
                     'them',
                     'themselves',
                     'then',
                     'there',
                     "there's",
                     'therefore',
                     'these',
                     'they',
                     "they'd",
                     "they'll",
                     "they're",
                     "they've",
                     'this',
                     'those',
                     'through',
                     'to',
                     'too',
                     'under',
                     'until',
                     'up',
                     'very',
                     'was',
                     "wasn't",
                     'we',
                     "we'd",
                     "we'll",
                     "we're",
                     "we've",
```

```
 'were',
 "weren't",
 'what',
 "what's",
 'when',
 "when's",
 'where',
 "where's",
 'which',
 'while',
 'who',
 "who's",
 'whom',
 'why',
 "why's",
 'wine',
 'with',
 "won't",
 'would',
 "wouldn't",
 'www',
 'you',
 "you'd",
 "you'll",
 "you're",
 "you've",
 'your',
 'yours',
 'yourself',
 'yourselves'}
```

In [63]:
```
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)
```

In [64]:
```
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.figure(figsize=[20, 20])
plt.show()
```



<Figure size 1440x1440 with 0 Axes>

Clearly Eisbock beer style has highest positive polarity which means it has received very good reviews even the wordcloud for Eisbock has some positive words poping out .

Question 6.How does written review compare to overall review score for the beer styles?

In [50]:
```
reviews_all = beerData.groupby('beer_style').agg({
                                            'polarity_score': np.mean,
                                            'review_overall': np.mean
                                        })
review_score_beerstyle = pd.DataFrame(reviews_all.reset_index()).sort_values(['polarity_score','review_overall'], asce
nding=[False,False])
review_score_beerstyle
```

Out[50]:

	beer_style	polarity_score	review_overall
41	Eisbock	0.898516	4.082474
32	Braggot	0.888858	3.648990
86	Quadrupel (Quad)	0.887129	4.052675
58	Flanders Red Ale	0.884769	3.966391
101	Wheatwine	0.880095	3.816327
...	...	...	...
69	Japanese Rice Lager	0.621147	3.028398
64	Happoshu	0.619300	2.818182
13	American Malt Liquor	0.590011	2.724702
76	Light Lager	0.564712	2.921185
77	Low Alcohol Beer	0.537040	2.582759

104 rows × 3 columns

Lets do percentile bucketing on both overall ratings and polarity score (Bucket 10 have higher values and Bucket 1 have lower values)

Percentile Bucketing on Polarity Score.

In [51]:
```
review_score_beerstyle['polarityQuantile'] = pd.qcut(review_score_beerstyle['polarity_score'], q=10, precision=0)
bin_labels = ['1', '2', '3', '4', '5','6','7','8','9','10']
review_score_beerstyle['polarityQuantileBucket'] = pd.qcut(review_score_beerstyle['polarity_score'], q=10, precision=0
,labels=bin_labels)
review_score_beerstyle
```

Out[51]:

	beer_style	polarity_score	review_overall	polarityQuantile	polarityQuantileBucket
41	Eisbock	0.898516	4.082474	(0.86, 0.9]	10
32	Braggot	0.888858	3.648990	(0.86, 0.9]	10
86	Quadrupel (Quad)	0.887129	4.052675	(0.86, 0.9]	10
58	Flanders Red Ale	0.884769	3.966391	(0.86, 0.9]	10
101	Wheatwine	0.880095	3.816327	(0.86, 0.9]	10
...	...	...	...	...	...
69	Japanese Rice Lager	0.621147	3.028398	(0.53, 0.7]	1
64	Happoshu	0.619300	2.818182	(0.53, 0.7]	1
13	American Malt Liquor	0.590011	2.724702	(0.53, 0.7]	1
76	Light Lager	0.564712	2.921185	(0.53, 0.7]	1
77	Low Alcohol Beer	0.537040	2.582759	(0.53, 0.7]	1

104 rows × 5 columns

Percentile Bucketing on Overall Rating.

In [52]:
```
review_score_beerstyle['ratingsQuantile'] = pd.qcut(review_score_beerstyle['review_overall'], q=10, precision=0)
bin_labels = ['1', '2', '3', '4', '5','6','7','8','9','10']
review_score_beerstyle['ratingsQuantileBucket'] = pd.qcut(review_score_beerstyle['review_overall'], q=10, precision=0,
labels=bin_labels)
review_score_beerstyle
```

Out[52]:

	beer_style	polarity_score	review_overall	polarityQuantile	polarityQuantileBucket	ratingsQuantile	ratingsQuantileBucket
41	Eisbock	0.898516	4.082474	(0.86, 0.9]	10	(4.03, 4.14]	10
32	Braggot	0.888858	3.648990	(0.86, 0.9]	10	(3.64, 3.73]	3
86	Quadrupel (Quad)	0.887129	4.052675	(0.86, 0.9]	10	(4.03, 4.14]	10
58	Flanders Red Ale	0.884769	3.966391	(0.86, 0.9]	10	(3.95, 4.03]	9
101	Wheatwine	0.880095	3.816327	(0.86, 0.9]	10	(3.79, 3.82]	5
...	...	...	...	...	...	...	...
69	Japanese Rice Lager	0.621147	3.028398	(0.53, 0.7]	1	(2.5700000000000003, 3.37]	1
64	Happoshu	0.619300	2.818182	(0.53, 0.7]	1	(2.5700000000000003, 3.37]	1
13	American Malt Liquor	0.590011	2.724702	(0.53, 0.7]	1	(2.5700000000000003, 3.37]	1
76	Light Lager	0.564712	2.921185	(0.53, 0.7]	1	(2.5700000000000003, 3.37]	1
77	Low Alcohol Beer	0.537040	2.582759	(0.53, 0.7]	1	(2.5700000000000003, 3.37]	1

104 rows × 7 columns

In [53]:
```
review_score_beerstyle['ratingsQuantile'].value_counts()
```

Out[53]:
```
(4.03, 4.14]                11
(3.86, 3.9]                 11
(3.73, 3.79]                11
(2.5700000000000003, 3.37]  11
(3.95, 4.03]                10
(3.9, 3.95]                 10
(3.82, 3.86]                10
(3.79, 3.82]                10
(3.64, 3.73]                10
(3.37, 3.64]                10
Name: ratingsQuantile, dtype: int64
```

In [54]:
```
beerstylePolarityBucketRatingBucket = review_score_beerstyle.groupby(["ratingsQuantileBucket","polarityQuantileBucket"
]).size().to_frame('CntOfBeerStyles').\
reset_index()
```

In [55]:
```
beerstylePolarityBucketRatingBucket[beerstylePolarityBucketRatingBucket["polarityQuantileBucket"] == '1']
```

Out[55]:

	ratingsQuantileBucket	polarityQuantileBucket	CntOfBeerStyles
0	1	1	8
10	2	1	0
20	3	1	0
30	4	1	2
40	5	1	1
50	6	1	0
60	7	1	0
70	8	1	0
80	9	1	0
90	10	1	0

From the above table we can infer that if any particular Beer Style receives poor written reviews (low polarity score) then it will have poor overall rating

In [56]:
```
beerstylePolarityBucketRatingBucket[beerstylePolarityBucketRatingBucket["polarityQuantileBucket"] == '10']
```

Out[56]:

	ratingsQuantileBucket	polarityQuantileBucket	CntOfBeerStyles
9	1	10	0
19	2	10	0
29	3	10	1
39	4	10	0
49	5	10	1
59	6	10	0
69	7	10	1
79	8	10	0
89	9	10	2
99	10	10	6

From the above table we can infer that if any particular Beer Style receives good written reviews (high polarity score) then it generally have good overall ratings