# Question 1

In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```python
ingredient_df = pd.read_csv("ingredient.csv")
ingredient_df
```

Out[2]:

| | a | b | c | d | e | f | g | h | i |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.51735 | 13.02 | 3.54 | 1.69 | 72.73 | 0.54 | 8.44 | 0.00 | 0.07 |
| **1** | 1.53125 | 10.73 | 0.00 | 2.10 | 69.81 | 0.58 | 13.30 | 3.15 | 0.28 |
| **2** | 1.52300 | 13.31 | 3.58 | 0.82 | 71.99 | 0.12 | 10.17 | 0.00 | 0.03 |
| **3** | 1.51768 | 12.56 | 3.52 | 1.43 | 73.15 | 0.57 | 8.54 | 0.00 | 0.00 |
| **4** | 1.51813 | 13.43 | 3.98 | 1.18 | 72.49 | 0.58 | 8.15 | 0.00 | 0.00 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **209** | 1.52152 | 13.12 | 3.58 | 0.90 | 72.20 | 0.23 | 9.82 | 0.00 | 0.16 |
| **210** | 1.51848 | 13.64 | 3.87 | 1.27 | 71.96 | 0.54 | 8.32 | 0.00 | 0.32 |
| **211** | 1.51784 | 12.68 | 3.67 | 1.16 | 73.11 | 0.61 | 8.70 | 0.00 | 0.00 |
| **212** | 1.51841 | 12.93 | 3.74 | 1.11 | 72.28 | 0.64 | 8.96 | 0.00 | 0.22 |
| **213** | 1.51321 | 13.00 | 0.00 | 3.02 | 70.70 | 6.21 | 6.93 | 0.00 | 0.00 |

214 rows × 9 columns

In [3]:
```python
ingredient_df.describe()
```

Out[3]:

| | a | b | c | d | e | f | g | h | i |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 214.000000 | 214.000000 | 214.000000 | 214.000000 | 214.000000 | 214.000000 | 214.000000 | 214.000000 | 214.000000 |
| **mean** | 1.518365 | 13.407850 | 2.684533 | 1.444907 | 72.650935 | 0.497056 | 8.956963 | 0.175047 | 0.057009 |
| **std** | 0.003037 | 0.816604 | 1.442408 | 0.499270 | 0.774546 | 0.652192 | 1.423153 | 0.497219 | 0.097439 |
| **min** | 1.511150 | 10.730000 | 0.000000 | 0.290000 | 69.810000 | 0.000000 | 5.430000 | 0.000000 | 0.000000 |
| **25%** | 1.516522 | 12.907500 | 2.115000 | 1.190000 | 72.280000 | 0.122500 | 8.240000 | 0.000000 | 0.000000 |
| **50%** | 1.517680 | 13.300000 | 3.480000 | 1.360000 | 72.790000 | 0.555000 | 8.600000 | 0.000000 | 0.000000 |
| **75%** | 1.519157 | 13.825000 | 3.600000 | 1.630000 | 73.087500 | 0.610000 | 9.172500 | 0.000000 | 0.100000 |
| **max** | 1.533930 | 17.380000 | 4.490000 | 3.500000 | 75.410000 | 6.210000 | 16.190000 | 3.150000 | 0.510000 |

The scale of the data different a lot between columns. in the following segment, we will rescale and calculate the anova test F score to see the significance of the additives to the formulations of petrol

In [4]:
```python
# first, we will scale the dataframes with the min max scaler functions
from scipy.stats import f_oneway
```

```
import pandas as pd
from sklearn import preprocessing

min_max_scaler = preprocessing.MinMaxScaler()
ingredient_scaled = min_max_scaler.fit_transform(ingredient_df)
df = pd.DataFrame(ingredient_scaled)
df.columns = ingredient_df.columns
```

To calculate the F score, we first want to define the null and alternate hypothesis as follow:

- $H_0$ (Null hypothesis) — that there is no difference among group means.
- $H_1$ (Alternate hypothesis) — that at least one group differs significantly from the overall mean of the dependent variable.

if the F test score is significantly greater than the F table critical value, then we will reject the null hypothesis. this means that the additives does have a significant effect to the formulations of petrol

## Calculate the group means and overall mean between groups

In [5]:
```
group_means = df.mean()
group_means
```

Out[5]:
```
a    0.316744
b    0.402684
c    0.597891
d    0.359784
e    0.507310
f    0.080041
g    0.327785
h    0.055570
i    0.111783
dtype: float64
```

In [6]:
```
overall_mean = group_means.mean()
overall_mean
```

Out[6]:
```
0.30662137944725787
```

## Sum of Squares

In [7]:
```
SS_total = (((df - overall_mean)**2).sum()).sum()
SS_total
```

Out[7]:
```
119.43404733965048
```

In [8]:
```
SS_within = (((df - group_means)**2).sum()).sum()
SS_within
```

Out[8]:
```
57.36422131848856
```

In [9]:
```
SS_between = (df.shape[0] * (group_means - overall_mean)**2).sum()
SS_between
```

Out[9]:
```
62.06982602116198
```

In [10]:
```
# this is to verify that SS_total = SS_within + SS_between
SS_within + SS_between
```

Out[10]:
```
119.43404733965053
```

## Degree of Freedom

In [11]:
```
df_total = df.shape[0] - 1         # 213
df_within = df.shape[0] - df.shape[1]        # 205
df_between = df.shape[1] - 1                          # 8
```

## Calculate the Mean Squares and the F score

In [12]:
```
mean_sq_between = SS_between / (df.shape[1] - 1)
mean_sq_within = \
    SS_within / (df.shape[0] - df.shape[1])
```

In [13]:
```
F = mean_sq_between / mean_sq_within
F
```

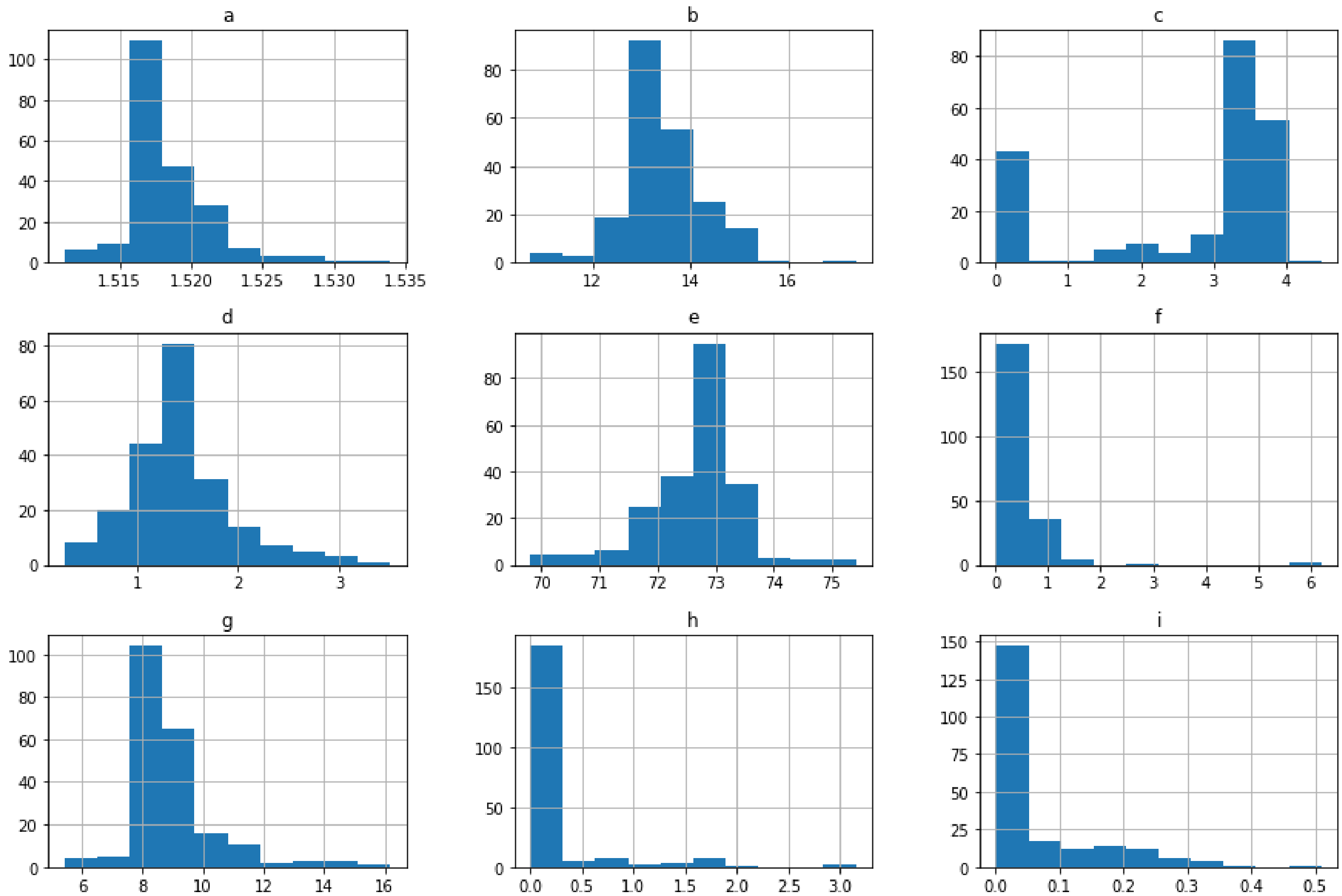
Out[13]:
```
27.727026624514522
```

Since the F value is more than the critical value, which is 27.727 > 1.9384 we can reject the null hypothesis, which means that the value in the additives ingredient type does effect the formulations of petrol and it is significantly different.

## Visualization of distribution and correlations

In [14]:
```
ingredient_df.hist(figsize=(15,10))
```

Out[14]:
```
array([[<AxesSubplot:title={'center':'a'}>,
        <AxesSubplot:title={'center':'b'}>,
        <AxesSubplot:title={'center':'c'}>],
       [<AxesSubplot:title={'center':'d'}>,
        <AxesSubplot:title={'center':'e'}>,
        <AxesSubplot:title={'center':'f'}>],
       [<AxesSubplot:title={'center':'g'}>,
        <AxesSubplot:title={'center':'h'}>,
        <AxesSubplot:title={'center':'i'}>]], dtype=object)
```



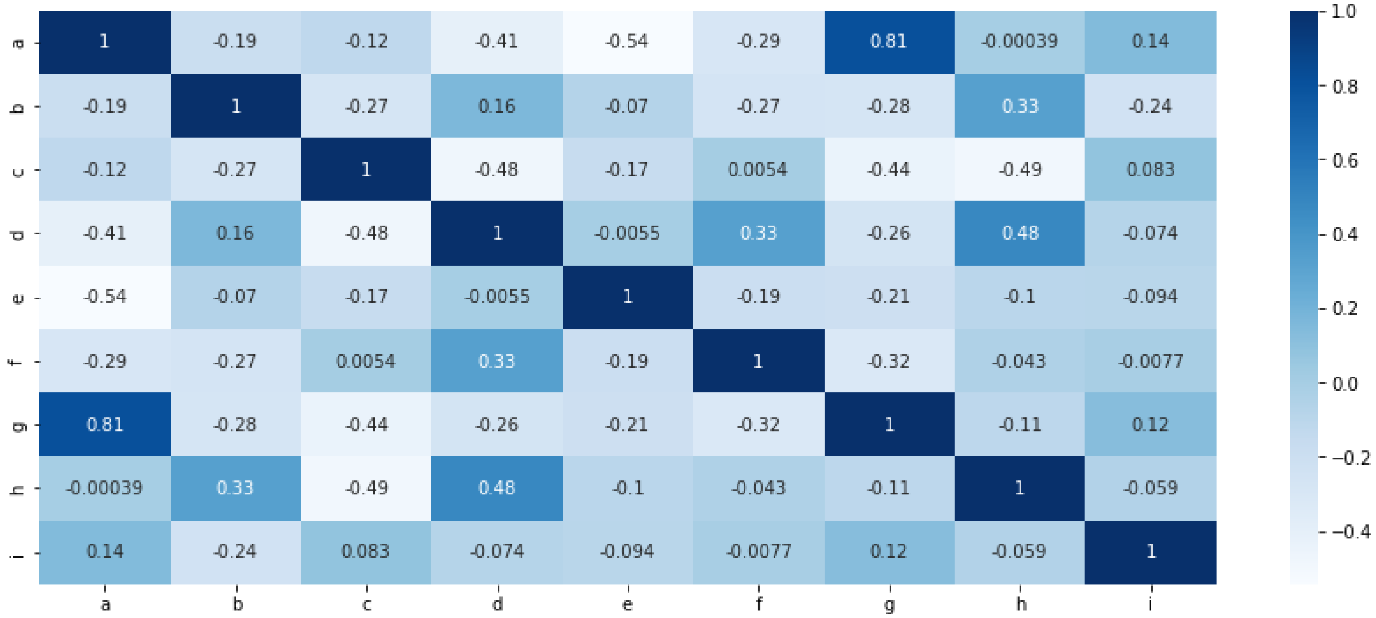

The following segments shows the correlation scatter plot for additive ingredients in regards to other additives ingredient. it seems that we can see some linear correlation from an ingredient to another ingredients. One such example is correlation from ingredient a to ingredients b, d, e, f, and strong linear correlation with ingredient g. this shows that the addition or reduction of the ingridients b, d, e, f, g may affect the value of ingredient a more significantly than other ingredients.

In [15]:

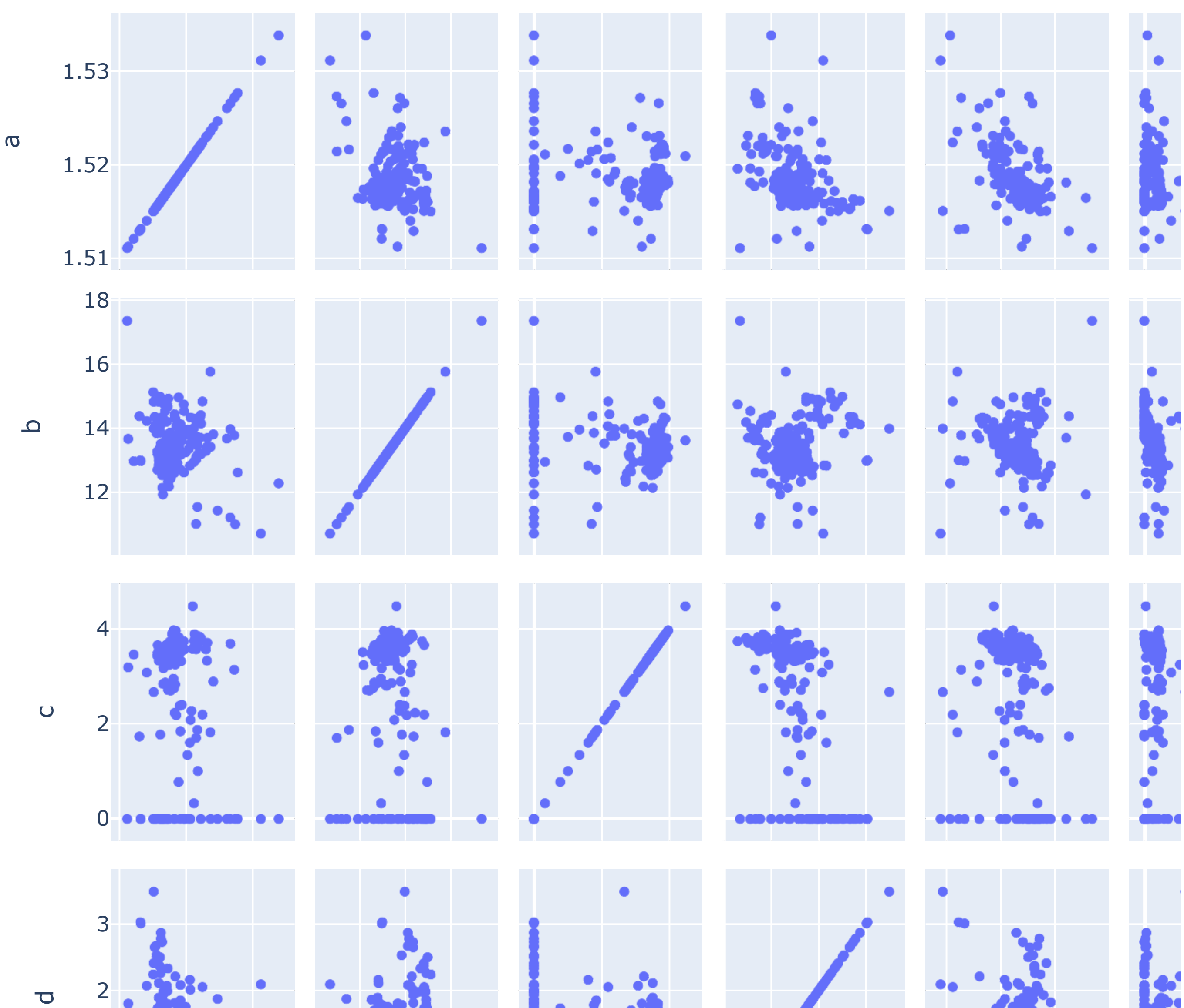
```
plt.figure(figsize=(15,6))
# calculate the correlation matrix
corr = ingredient_df.corr()


# plot the heatmap
sns.heatmap(corr, cmap="Blues", annot=True)
```

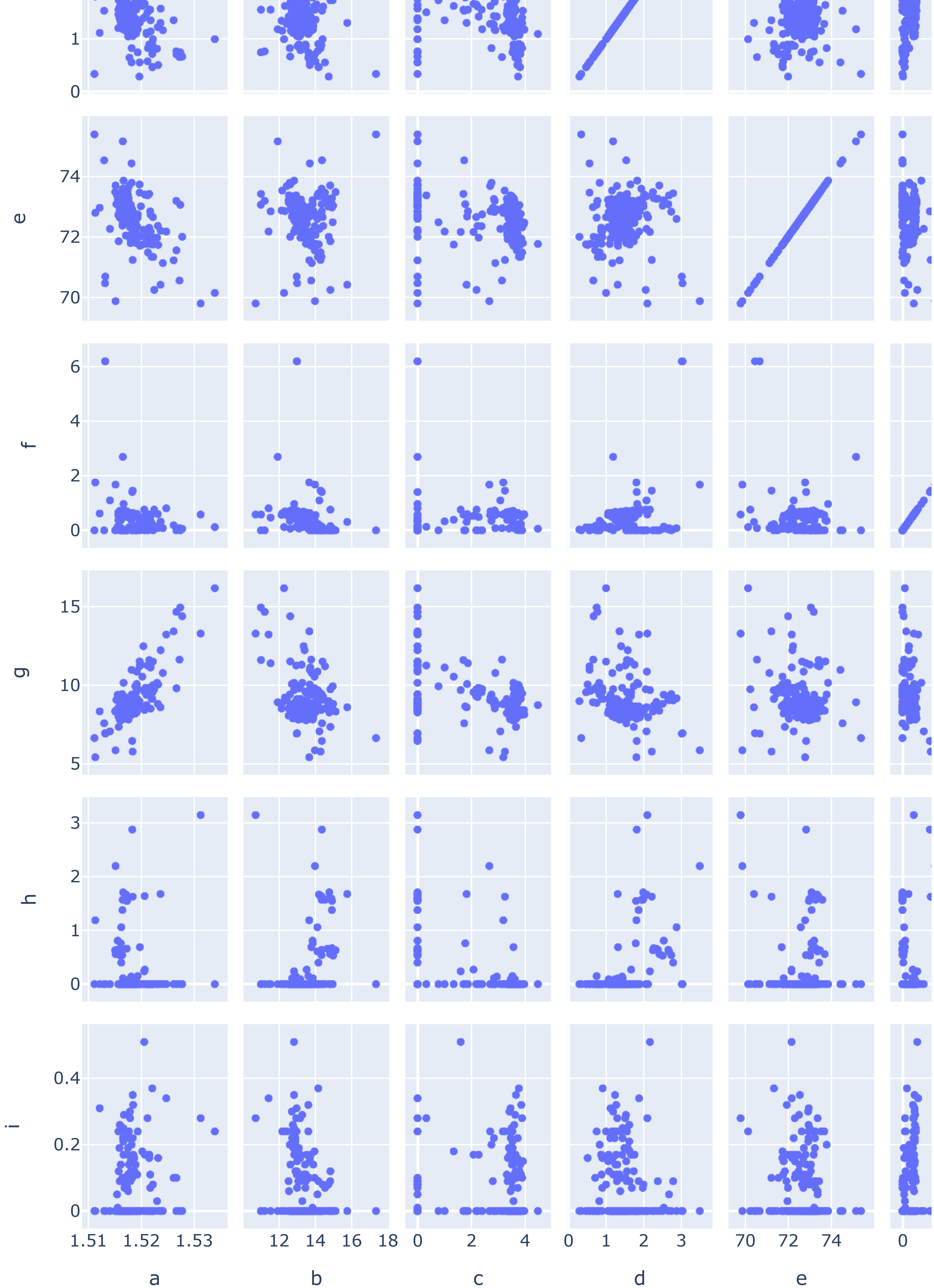
Out[15]:

```
<AxesSubplot:>
```




In [16]:
```python
import plotly.express as px
fig = px.scatter_matrix(ingredient_df,
width=1200, height=1600)
fig.show()
```



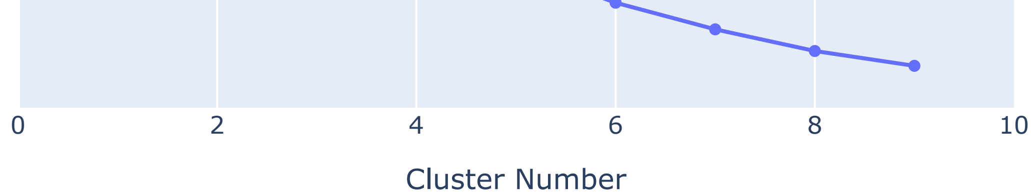



## Clustering analysis

In this segment, we will use K-means clustering algorithm to analyze the cluster of the groups. we already standardize the dataframe with minmax scaler on previous segment, we will next try to feed the dataframe to the kmeans algorithm

In [17]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
import plotly.graph_objects as go
import numpy as np

inertia = []
for i in range(1,10):
    kmeans = KMeans(
        n_clusters=i, init="k-means++"
    )
    kmeans.fit(df)
    inertia.append(kmeans.inertia_)
fig = go.Figure(data=go.Scatter(x=np.arange(1,10),y=inertia))
fig.update_layout(title="Inertia vs Cluster Number",xaxis=dict(range=[0,10],title="Clust
                  yaxis={'title':'Inertia'},
                 annotations=[
        dict(
            x=3,
            y=inertia[2],
            xref="x",
            yref="y",
            text="Elbow",
            showarrow=True,
            arrowhead=7,
            ax=20,
            ay=-40
        )
    ])
```

```
C:\Users\winson121\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarn
ing:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
```

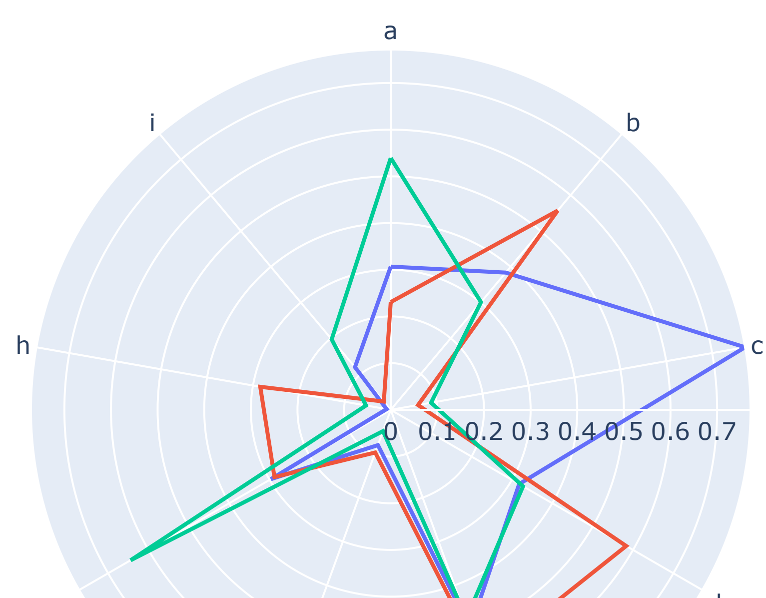


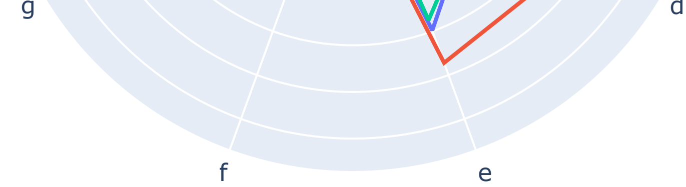

The elbow is between 2 and 3. we will use 3 clusters since it reduce the inertia much more from 2 clusters to 3 clusters compare to the reduce in inertia from 3 clusters to 10 clusters.

For easier visualization, we will use line polar plot to map the cluster and the features.

In [18]:
```
kmeans = KMeans(
        n_clusters=3, init="k-means++",
    )
kmeans.fit(df)
clusters=pd.DataFrame(df)
clusters['label']=kmeans.labels_
polar=clusters.groupby("label").mean().reset_index()
polar=pd.melt(polar,id_vars=["label"])
fig = px.line_polar(polar, r="value", theta="variable", color="label", line_close=True,h
fig.show()
```

```
C:\Users\winson121\anaconda3\lib\site-packages\plotly\express\_core.py:271: FutureWarnin
g:

The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.

C:\Users\winson121\anaconda3\lib\site-packages\plotly\express\_core.py:271: FutureWarnin
g:

The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.

C:\Users\winson121\anaconda3\lib\site-packages\plotly\express\_core.py:271: FutureWarnin
g:

The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
```



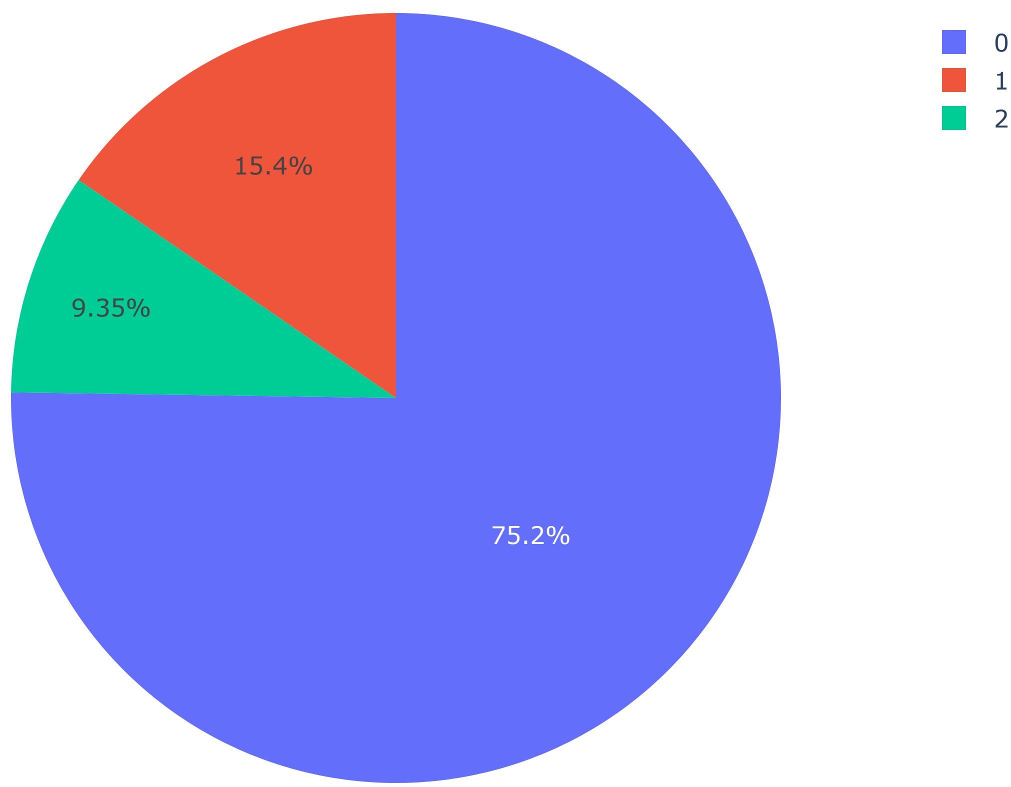



The graph shows that additive e and f is quite similar for every group of the clusters. For each label, we could categorize each segments base on high level of certains additive ingredients type in the clusters:

- label 0 consist of high level of ingredients c-e
- label 1 consist of high level of ingredients b-d-e
- label 2 consist of high level of ingredients a-g-e

now let's see the number of record in percentage for each clusters.

In [19]:
```
pie=clusters.groupby('label').size().reset_index()
pie.columns=['label','value']
px.pie(pie,values='value',names='label',color=['blue','red','green'])
```




The number of clustering indicate that label 1 is the most prevalent cluster among the other clusters. this may tell us that most of the additive ingredients data has the characteristics of label 1, which is high in ingredients b-d-e.