In [14]:

```python
import pandas as pd
import numpy as np
```

In [15]:

```python
df = pd.read_csv("imputeddatafinal.csv")
df.head(5)
```

Out[15]:

	PROP_ID	BLD_TYPE	APPRAISER	NBHD	QUAL	COND	KITCHEN_CT	KITCHEN_RATING	FULL_BATI
0	98421.0	01 - Ranch	OLG - Noah Olguin	0040 - 0040	C - C	AV - Average	1	AV - Average	
1	98437.0	01 - Ranch	OLG - Noah Olguin	0040 - 0040	C - C	GD - Good	1	AV - Average	
2	98472.0	01 - Ranch	OLG - Noah Olguin	0040 - 0040	C - C	AV - Average	1	AV - Average	
3	98475.0	04 - Cape Cod	OLG - Noah Olguin	0040 - 0040	C - C	AV - Average	1	AV - Average	
4	98476.0	01 - Ranch	OLG - Noah Olguin	0040 - 0040	C- - C-	FR - Fair	1	AV - Average	

In [16]:

```python
df.shape
```

Out[16]:

```
(122096, 20)
```

In [17]:

```python
df.isnull().sum()
```

Out[17]:

```
PROP_ID                  0
BLD_TYPE                 0
APPRAISER                0
NBHD                     0
QUAL                     0
COND                     0
KITCHEN_CT               0
KITCHEN_RATING           0
FULL_BATH_CT             0
FULL_BATH_RATING         0
HALF_BATH_CT             0
HALF_BATH_RATING         0
YEAR_BUILT               0
FINISHED_AREA            0
LAND_SF                  0
SALE_DATE           107341
SALE_PRICE          107347
APPEALED19          121210
APPEALED20          117704
APPEALED21          121579
dtype: int64
```

In [18]:

```python
df.info
```

Out[18]:

```
<bound method DataFrame.info of          PROP_ID            BLD_TYPE            APPRA
ISER         NBHD  \
0        98421.0           01 - Ranch     OLG - Noah Olguin  0040 - 0040
1        98437.0           01 - Ranch     OLG - Noah Olguin  0040 - 0040
```

```
2        98472.0            01 - Ranch     OLG - Noah Olguin  0040 - 0040
3        98475.0         04 - Cape Cod     OLG - Noah Olguin  0040 - 0040
4        98476.0            01 - Ranch     OLG - Noah Olguin  0040 - 0040
...          ...                   ...                   ...          ...
122091  265626.0  19 - Res O/S A & 1/2     OLG - Noah Olguin  2710 - 2710
122092  265635.0    22 - Dplx Bungalow      KOH - Ben Kohout  2870 - 2870
122093  265637.0  19 - Res O/S A & 1/2  STR - Crystal Strong  2950 - 2950
122094  265645.0       11 - Duplex O/S   KAE - Jody Kaebisch  2910 - 2910
122095  265649.0    22 - Dplx Bungalow      ESS - Mike Esser  4910 - 4910

           QUAL          COND  KITCHEN_CT KITCHEN_RATING  FULL_BATH_CT  \
0         C - C  AV - Average           1   AV - Average             1
1         C - C     GD - Good           1   AV - Average             2
2         C - C  AV - Average           1   AV - Average             1
3         C - C  AV - Average           1   AV - Average             1
4       C- - C-     FR - Fair           1   AV - Average             2
...         ...           ...         ...            ...           ...
122091    C - C  AV - Average           1   AV - Average             2
122092    C - C  AV - Average           2   AV - Average             2
122093  C- - C-     PR - Poor           1      PR - Poor             1
122094    C - C     GD - Good           2      GD - Good             2
122095    C - C  AV - Average           2   AV - Average             2

       FULL_BATH_RATING  HALF_BATH_CT      HALF_BATH_RATING  YEAR_BUILT  \
0          AV - Average             1          AV - Average        1954
1             GD - Good             0  N/A - Not Applicable        1955
2          AV - Average             0  N/A - Not Applicable        1960
3          AV - Average             0  N/A - Not Applicable        1951
4          AV - Average             0  N/A - Not Applicable        1952
...                 ...           ...                   ...         ...
122091     AV - Average             0  N/A - Not Applicable        1916
122092     AV - Average             0  N/A - Not Applicable        1923
122093        PR - Poor             0  N/A - Not Applicable        1895
122094        GD - Good             0  N/A - Not Applicable        1890
122095     AV - Average             0  N/A - Not Applicable        1924

        FINISHED_AREA     LAND_SF SALE_DATE  SALE_PRICE APPEALED19 APPEALED20  \
0              1802.0  38332.8000       NaN         NaN        NaN        NaN
1              1693.0  34848.0000       NaN         NaN        NaN        NaN
2              1174.0  14610.0240       NaN         NaN        NaN        NaN
3              1651.0  38206.0404       NaN         NaN        NaN        NaN
4              1000.0  38215.1880       NaN         NaN        NaN        NaN
...               ...         ...       ...         ...        ...        ...
122091         1661.0   7200.4680       NaN         NaN        NaN        NaN
122092         2238.0   9674.6760       NaN         NaN        NaN        NaN
122093         1461.0   3210.3720       NaN         NaN        NaN        NaN
122094         2000.0   4552.0200       NaN         NaN        NaN        NaN
122095         2377.0  29625.1560       NaN         NaN        NaN        NaN

       APPEALED21
0             NaN
1             NaN
2             NaN
3             NaN
4             NaN
...           ...
122091        NaN
122092        NaN
122093        NaN
122094        NaN
122095        NaN

[122096 rows x 20 columns]>
```

```
In [19]: # Inspect the categorical variables
```

```
df.select_dtypes('object').nunique()
```

Out[19]:

```
BLD_TYPE              20
APPRAISER             15
NBHD                 143
QUAL                  17
COND                   8
KITCHEN_RATING         8
FULL_BATH_RATING       8
HALF_BATH_RATING       9
SALE_DATE            990
APPEALED19             1
APPEALED20             1
APPEALED21             1
dtype: int64
```

In [20]:

```
df.describe()
```

Out[20]:

	PROP_ID	KITCHEN_CT	FULL_BATH_CT	HALF_BATH_CT	YEAR_BUILT	FINISHED_AREA	
count	122096.000000	122096.000000	122096.000000	122096.000000	122096.000000	122096.000000	1
mean	181144.674404	1.286668	1.419596	0.250516	1936.157786	1541.999844	2
std	48086.314750	0.478056	0.568489	0.479401	26.019289	625.540182	3
min	98421.000000	1.000000	1.000000	0.000000	1822.000000	366.000000	6
25%	138501.750000	1.000000	1.000000	0.000000	1918.000000	1083.000000	4
50%	176795.500000	1.000000	1.000000	0.000000	1940.000000	1378.000000	5
75%	226241.250000	2.000000	2.000000	0.000000	1955.000000	1897.000000	6
max	265649.000000	5.000000	13.000000	10.000000	2021.000000	12059.000000	6

In [21]:

```
df = df.drop(['PROP_ID', 'NBHD', 'SALE_DATE', 'SALE_PRICE', "APPEALED19", 'APPEALED20',
```

In [22]:

```
one_hot = pd.get_dummies(df['BLD_TYPE'])
df = df.drop('BLD_TYPE',axis = 1)
df = df.merge(one_hot, how='outer', left_index=True, right_index=True)

one_hot = pd.get_dummies(df['APPRAISER'])
df = df.drop('APPRAISER',axis = 1)
df = df.join(one_hot)

one_hot = pd.get_dummies(df['QUAL'])
df = df.drop('QUAL',axis = 1)
df = df.merge(one_hot, how='outer', left_index=True, right_index=True)

one_hot = pd.get_dummies(df['COND'])
df = df.drop('COND',axis = 1)
df = df.merge(one_hot, how='outer', left_index=True, right_index=True)

one_hot = pd.get_dummies(df['KITCHEN_RATING'])
df = df.drop('KITCHEN_RATING',axis = 1)
df = df.merge(one_hot, how='outer', left_index=True, right_index=True)

one_hot = pd.get_dummies(df['FULL_BATH_RATING'])
```

```
df = df.drop('FULL_BATH_RATING',axis = 1)
df = df.join(one_hot, how='outer')

one_hot = pd.get_dummies(df['HALF_BATH_RATING'])
df = df.drop('HALF_BATH_RATING',axis = 1)
df = df.merge(one_hot, how='outer', left_index=True, right_index=True)
```

In [23]:

```
df.shape
```

Out[23]:

```
(122096, 91)
```

In [24]:

```
from sklearn.preprocessing import Normalizer
transformer = Normalizer().fit(df)
transformer
Normalizer()
tdf = transformer.transform(df)
df = pd.DataFrame(tdf, columns = df.columns)
df
```

Out[24]:

	KITCHEN_CT	FULL_BATH_CT	HALF_BATH_CT	YEAR_BUILT	FINISHED_AREA	LAND_SF	01 - Ranch
0	0.000026	0.000026	0.000026	0.050853	0.046897	0.997604	0.000026
1	0.000029	0.000057	0.000000	0.055947	0.048449	0.997258	0.000029
2	0.000068	0.000068	0.000000	0.132544	0.079391	0.987993	0.000068
3	0.000026	0.000026	0.000000	0.050951	0.043117	0.997770	0.000000
4	0.000026	0.000052	0.000000	0.050995	0.026125	0.998357	0.000026
...	...	...	...	...	...	...	...
122091	0.000131	0.000262	0.000000	0.250985	0.217582	0.943220	0.000000
122092	0.000198	0.000198	0.000000	0.190120	0.221263	0.956502	0.000000
122093	0.000250	0.000250	0.000000	0.473276	0.364885	0.801791	0.000000
122094	0.000376	0.000376	0.000000	0.355322	0.376002	0.855785	0.000000
122095	0.000067	0.000067	0.000000	0.064602	0.079812	0.994714	0.000000

122096 rows × 91 columns

In [25]:

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca.fit(df)
pca_df = pca.transform(df)
pca_df
```

Out[25]:

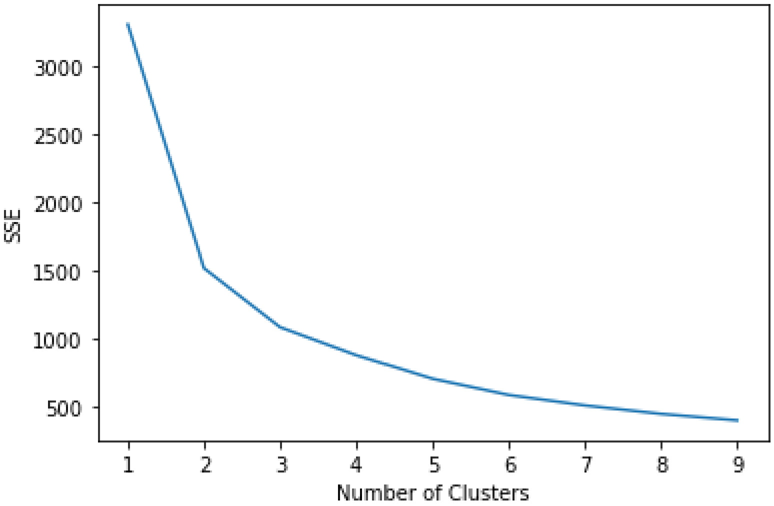
```
array([[-0.34251279, -0.10372781],
       [-0.33865636, -0.10057213],
       [-0.27324179, -0.05731459],
       ...,
       [ 0.19250991,  0.05952918],
       [ 0.11686775, -0.04582888],
       [-0.31009916, -0.11223897]])
```

In [26]:

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

In [27]:

```
sse = {}
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(pca_df)
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of samples to their closest cl
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()
```



In [28]:

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(pca_df)
```

Out[28]:

```
KMeans(n_clusters=3)
```

In [29]:

```
df = pd.read_csv("imputeddatafinal.csv")
```

In [30]:

```
df_final = pd.concat([df.reset_index(drop =True), pd.DataFrame(pca_df)], axis = 1)
```

In [31]:

```
df_final.columns.values[-2: ] = ['Component_1', 'Component_2']
```

In [32]:

```
df_final['Cluster'] = kmeans.labels_
```

In [33]:

```
df_final.shape
```

Out[33]:

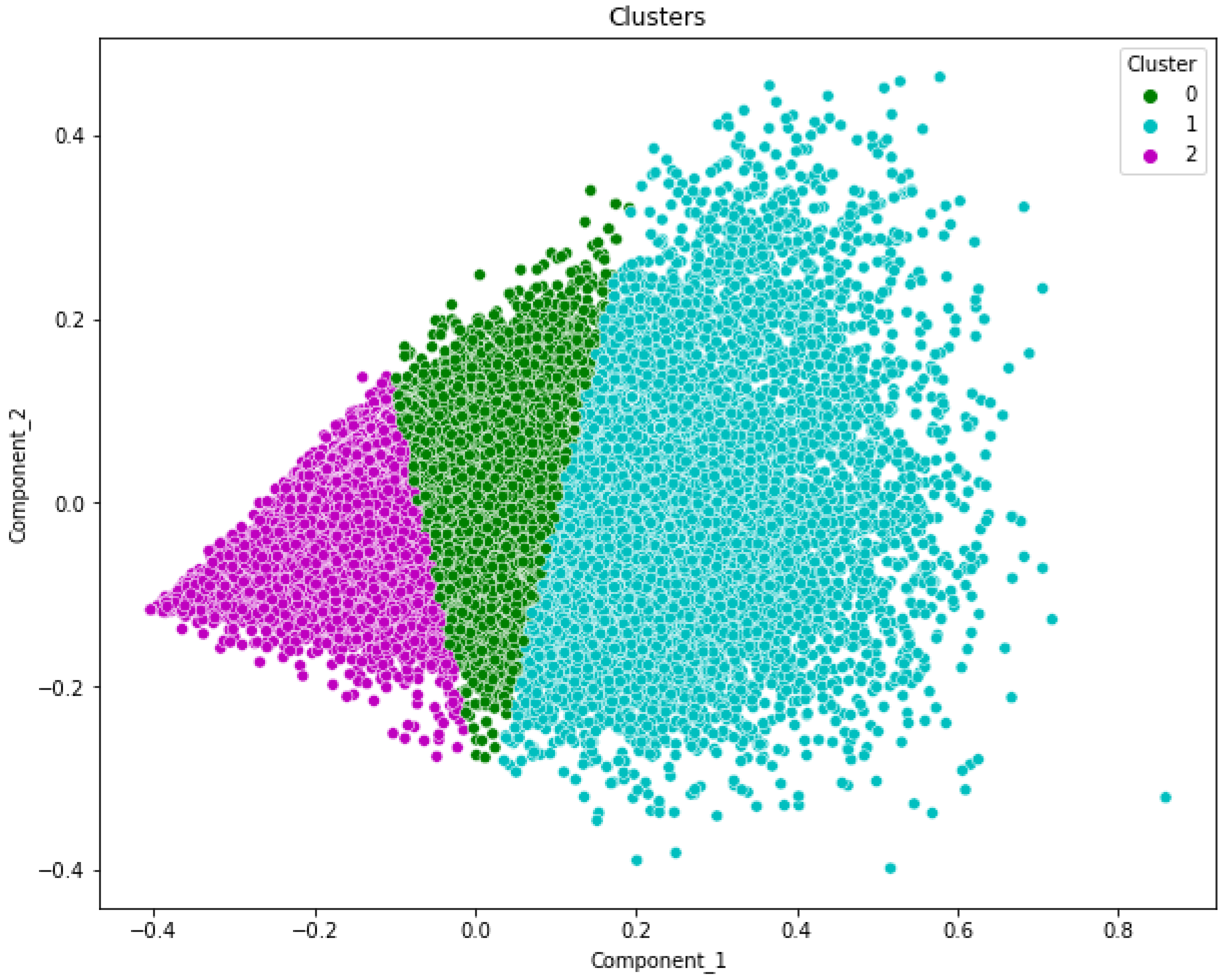
```
(122096, 23)
```

In [34]:

```
x_axis = df_final['Component_1']
y_axis = df_final['Component_2']
plt.figure(figsize = (10,8))
import seaborn as sns
sns.scatterplot(x_axis, y_axis, hue = df_final['Cluster'], palette = ['g', 'c', 'm'])
```

```
plt.title('Clusters')
plt.show()
```

```
C:\Users\yadus\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variables as keyword args: x, y. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```



In [35]:

```
df_Cluster_0 = df_final.loc[df_final['Cluster'] == 0]
df_Cluster_1 = df_final.loc[df_final['Cluster'] == 1]
df_Cluster_2 = df_final.loc[df_final['Cluster'] == 2]
```

In [36]:

```
from scipy import spatial

df_kd_0 = df_Cluster_0[['Component_1', 'Component_2']]
df_kd_1 = df_Cluster_1[['Component_1', 'Component_2']]
df_kd_2 = df_Cluster_2[['Component_1', 'Component_2']]
```

In [47]:

```
tree = spatial.cKDTree(df_kd_1)

# Replace PROP_ID With A property ID number below
distances, indices = tree.query(df_kd_1.loc[PROP_ID].values, k=1+1)
similar_properties = df_kd_1.iloc[indices[1:]].assign(Distance=distances[1:])

print(similar_properties)
```

```
        Component_1  Component_2  Distance
53632      0.117304    -0.045867  0.000438
```

```
53308      0.117435    -0.045976  0.000586
103625     0.117454    -0.045853  0.000587
```

In [51]:

```python
df_final.to_csv('Final.csv', index=False)
```