Melbouerne Housing Price Data Exploration

Analysis performed on the Melbourne Housing Price Data to identify major Suburbs, Number of Rooms, locations and Date of sold. In this analysis, we analysed the prime locations to buy a house, Primitive areas with the Price factors anamolies, statistics of Price variation trend.

Visualization performed to analyse the data is represented on a poster. We have utilised python to perform analyses

This dataset provides us with several kinds of insights and if we will be able to dig deeper we will be able to predict the price of the houses depending on the data provided. In this Explainatory Data Analysis we will try to answer some few kquestions to get the idea of the data.

Using the dataset some objectives that can be achieve are the following:

- Is the dataset enough to predict whether a housing bubble exists.
- Which suburbs are the best to buy in?
- Where's the expensive side of town?
- Where should one buy a 2 bedroom unit for best returns?

For our EDA we are going to use few libraries such as:

- Pandas
- Numpy
- MatplotLib
- Seaborn
- Folium

Importing all the required libraries

In [1]:

```
import pandas as pd
import numpy as np
import pandas_profiling
from pandas import Series, DataFrame
import matplotlib.pyplot as plt
import os
from wordcloud import WordCloud ,STOPWORDS, ImageColorGenerator
import seaborn as sns
%matplotlib inline
import folium
from folium.plugins import HeatMap
p = "YlGnBu"
p2 = "YlGn"
p3 = "Greys"
p4="viridis"
p5="coolwarm"
import datetime as dt
from sklearn.preprocessing import Imputer
from sklearn import preprocessing as pro
from pandas.plotting import scatter_matrix
import warnings
warnings.filterwarnings('ignore')
```

Reading the Dataset

In [3]:

```
df =
pd.read_csv(r"C:\Users\SA20020888\Desktop\Projects\MELBOURNE_HOUSE_PRICES_LESS.csv\Melbourne_housir
ULL.csv")
```

Display Information about the data

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34857 entries, 0 to 34856
Data columns (total 21 columns):
Suburb           34857 non-null object
Address          34857 non-null object
Rooms            34857 non-null int64
Type             34857 non-null object
Price            27247 non-null float64
Method           34857 non-null object
SellerG          34857 non-null object
Date             34857 non-null object
Distance         34856 non-null float64
Postcode         34856 non-null float64
Bedroom2         26640 non-null float64
Bathroom         26631 non-null float64
Car              26129 non-null float64
Landsize         23047 non-null float64
BuildingArea     13742 non-null float64
YearBuilt        15551 non-null float64
CouncilArea      34854 non-null object
Lattitude        26881 non-null float64
Longtitude       26881 non-null float64
Regionname       34854 non-null object
Propertycount    34854 non-null float64
dtypes: float64(12), int64(1), object(8)
memory usage: 5.6+ MB
```

Column Labels

In [5]:

```
df.columns
```

Out[5]:

```
Index(['Suburb', 'Address', 'Rooms', 'Type', 'Price', 'Method', 'SellerG',
       'Date', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom', 'Car',
       'Landsize', 'BuildingArea', 'YearBuilt', 'CouncilArea', 'Lattitude',
       'Longtitude', 'Regionname', 'Propertycount'],
      dtype='object')
```

Data Wrangling and Cleaning

As we can see there are many categorical values which are defined as object we need to convert its data type to "Category", there is another column which contains dates of the property sold which are also not in datetime format.

So we first convert the date column to datetime data type then we will fetch year and month from the same

In [6]:

```
df['Date'] = pd.to_datetime(df['Date'])
df['Month'] = df['Date'].dt.month
df['Year'] =  df['Date'].dt.year
```

In [7]:

```
category_cols = ['Type', 'Method', 'CouncilArea', 'Regionname', 'SellerG']
for i in category_cols:
    df[i] = df[i].astype('category')
```

Handling Missing Values

In [8]:

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

Out[8]:

```
Suburb               0
Address              0
Rooms                0
Type                 0
Price             7610
Method               0
SellerG              0
Date                 0
Distance             1
Postcode             1
Bedroom2          8217
Bathroom          8226
Car               8728
Landsize         11810
BuildingArea     21115
YearBuilt        19306
CouncilArea          3
Lattitude         7976
Longtitude        7976
Regionname           3
Propertycount        3
Month                0
Year                 0
dtype: int64
```

As we can see there are many columns which contains null values and most importantly price column also has null values

we handle these value in many ways but the simplest one is to drop all the rows corresponding to it

In [9]:

```
data_priced  = df[df.Price.notnull()].drop('Address', axis = 1)
```

Now as we looked closely we could see that there are some insignificant columns which doesnt provide any help in gettin the insights and moreover it also contains null values, so lets go ahead and drop them as well

In [10]:

```
data_priced.drop(columns = ['Distance', 'Propertycount', 'YearBuilt', 'BuildingArea', 'Car', 'Lands
ize', 'Bedroom2', 'Bathroom'], inplace=True)
```

In [11]:

```
data_priced.dropna(axis = 0, subset=['Postcode','CouncilArea','Lattitude','Longtitude'], inplace=Tr
ue)
```

After Handling all the missing values, lets move forward with the data cleaning.

One of the major data cleaning process is to detect all the outliers existing in the data.

- Q1 is the 25% value of all the prices dristributed over the data
- Q3 is the 75% value of all the prices dristributed over the data

IQR is the inter quartile range which contains the 50% amount of data. For detecting Outliers we use two formulas:

- Q1 - 1.5 * IQR to detect all the outliers in the lower range
- Q1 + 1.5 * IQR to detect all the outliers in the higher range

In [12]:

```
Q1 = data_priced['Price'].quantile(0.25)
```

```
Q3 = data_priced['Price'].quantile(0.75)
IQR = Q3 - Q1
filtered = data_priced.query('(@Q1 - 1.5 * @IQR) <= Price <= (@Q3 + 1.5 * @IQR)')
filtered['PriceMillions'] = filtered['Price'].apply(lambda x: x/1000000)
```

filtered dataframe contains the data excluding all the outliers

Lets compare the mean before and after removing Outliers

In [13]:

```
data_priced['Price'].mean() , filtered['Price'].mean()
```

Out[13]:

```
(1089746.1750583528, 991633.7877728385)
```

We can clearly see the difference between the mean

In [14]:

```
filtered.head()
```

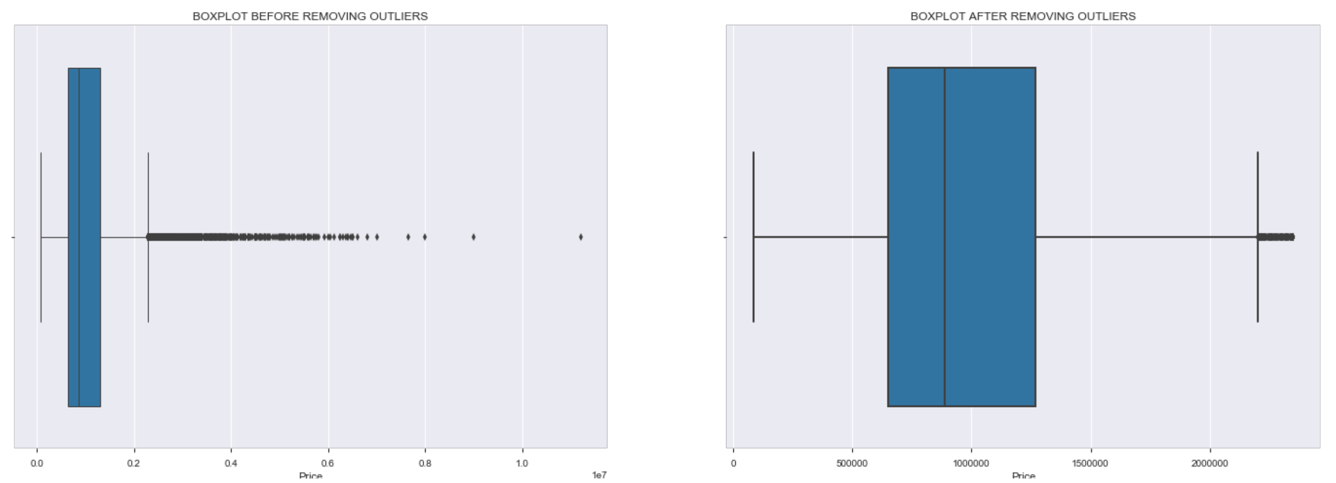
Out[14]:

	Suburb	Rooms	Type	Price	Method	SellerG	Date	Postcode	CouncilArea	Lattitude	Longtitude	Regionname	Month	Y
1	Abbotsford	2	h	1480000.0	S	Biggin	2016-03-12	3067.0	Yarra City Council	-37.7996	144.9984	Northern Metropolitan	3	2
2	Abbotsford	2	h	1035000.0	S	Biggin	2016-04-02	3067.0	Yarra City Council	-37.8079	144.9934	Northern Metropolitan	4	2
4	Abbotsford	3	h	1465000.0	SP	Biggin	2017-04-03	3067.0	Yarra City Council	-37.8093	144.9944	Northern Metropolitan	4	2
5	Abbotsford	3	h	850000.0	PI	Biggin	2017-04-03	3067.0	Yarra City Council	-37.7969	144.9969	Northern Metropolitan	4	2
6	Abbotsford	4	h	1600000.0	VB	Nelson	2016-04-06	3067.0	Yarra City Council	-37.8072	144.9941	Northern Metropolitan	4	2

Lets Plot the box plot to verify our assumption

In [15]:

```
fig, axs = plt.subplots(ncols=2, figsize=(24,8))
# fig = plt.figure(figsize=(20,6), dpi=105)
sns.boxplot(df['Price'], linewidth=1, ax=axs[0])
axs[0].title.set_text("BOXPLOT BEFORE REMOVING OUTLIERS")
sns.boxplot(filtered['Price'], linewidth=2)
axs[1].title.set_text("BOXPLOT AFTER REMOVING OUTLIERS")
plt.show()
```



Plotting the Correlation matrix to find all the relations between various variables

In [16]:

```
corr_matrix = filtered.corr()
corr_matrix
```

Out[16]:

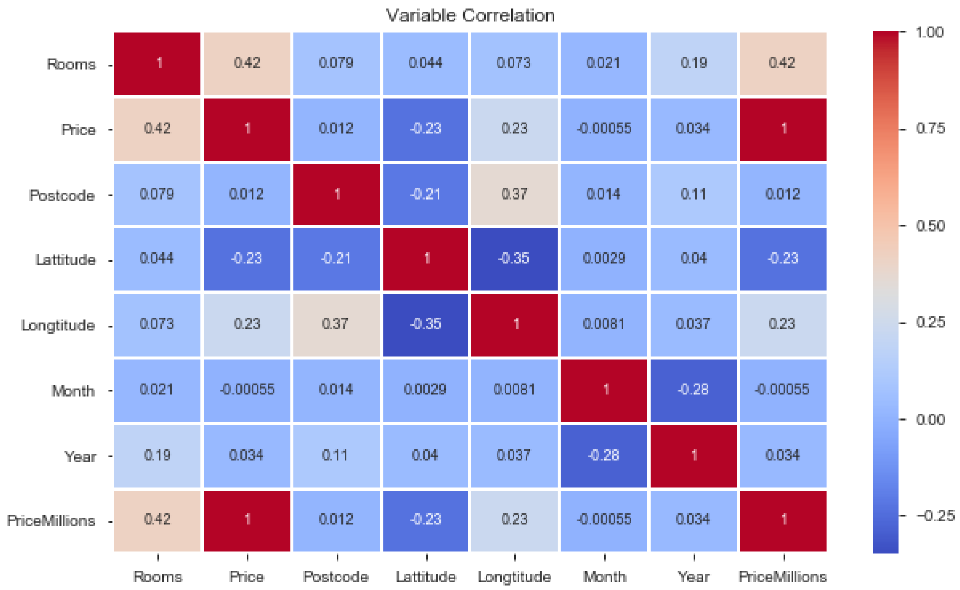
	Rooms	Price	Postcode	Lattitude	Longtitude	Month	Year	PriceMillions
Rooms	1.000000	0.418845	0.079192	0.044351	0.072863	0.021071	0.188223	0.418845
Price	0.418845	1.000000	0.011698	-0.229351	0.233019	-0.000551	0.034162	1.000000
Postcode	0.079192	0.011698	1.000000	-0.211213	0.366581	0.013545	0.112738	0.011698
Lattitude	0.044351	-0.229351	-0.211213	1.000000	-0.348788	0.002883	0.039637	-0.229351
Longtitude	0.072863	0.233019	0.366581	-0.348788	1.000000	0.008061	0.036644	0.233019
Month	0.021071	-0.000551	0.013545	0.002883	0.008061	1.000000	-0.284837	-0.000551
Year	0.188223	0.034162	0.112738	0.039637	0.036644	-0.284837	1.000000	0.034162
PriceMillions	0.418845	1.000000	0.011698	-0.229351	0.233019	-0.000551	0.034162	1.000000

In [17]:

```
plt.figure(figsize=(10,6))
sns.heatmap(corr_matrix,cmap = 'coolwarm',linewidth = 1,annot= True, annot_kws={"size": 9})
plt.title('Variable Correlation')
```

Out[17]:

```
Text(0.5, 1.0, 'Variable Correlation')
```

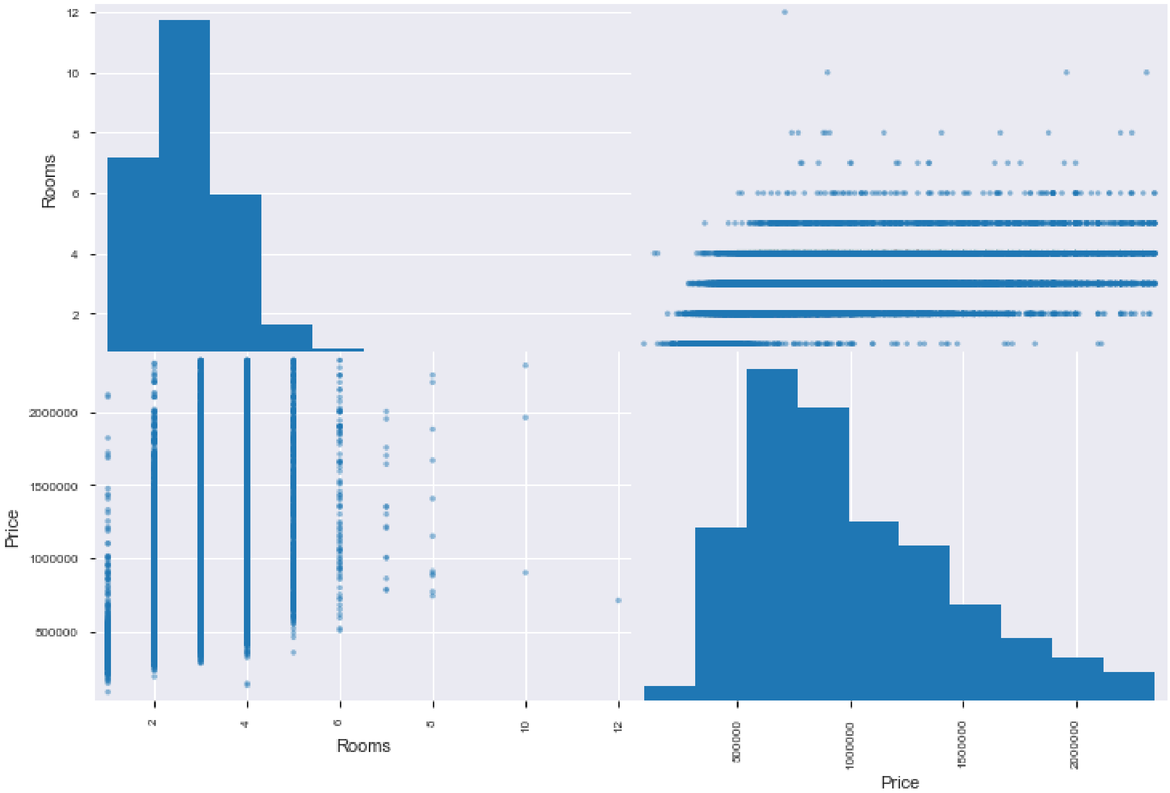


As we can see there is not so many strong correlations between any of the variables apart from Number of Rooms and Prices,

So Lets Plot a Scatter matrix to visualize the relation between "Rooms" and "Price"

In [18]:

```
attributes = ["Rooms","Price"]
scatter_matrix(filtered[attributes], figsize=(12, 8))
#plt.savefig('matrix.png')
plt.show()
```

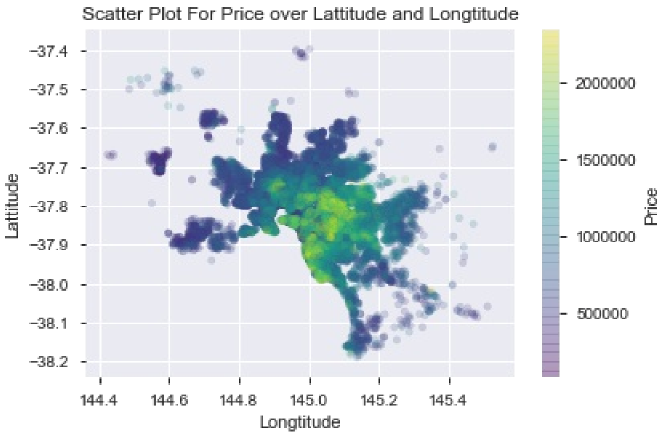


Lets Plot Scatter Plot For Price over Lattitude and Longtitude to get the idea of the shape of distribution of houses over melbourne

In [19]:

```
fig = plt.figure(figsize=(30,10))
filtered.plot(kind="scatter", x="Longtitude", y="Lattitude", c = "Price", alpha=0.2, colorbar=True,
sharex=False, cmap=plt.get_cmap("viridis"))
plt.title('Scatter Plot For Price over Lattitude and Longtitude')
plt.show()
```

```
<Figure size 2160x720 with 0 Axes>
```



More Zoomed in version of the same

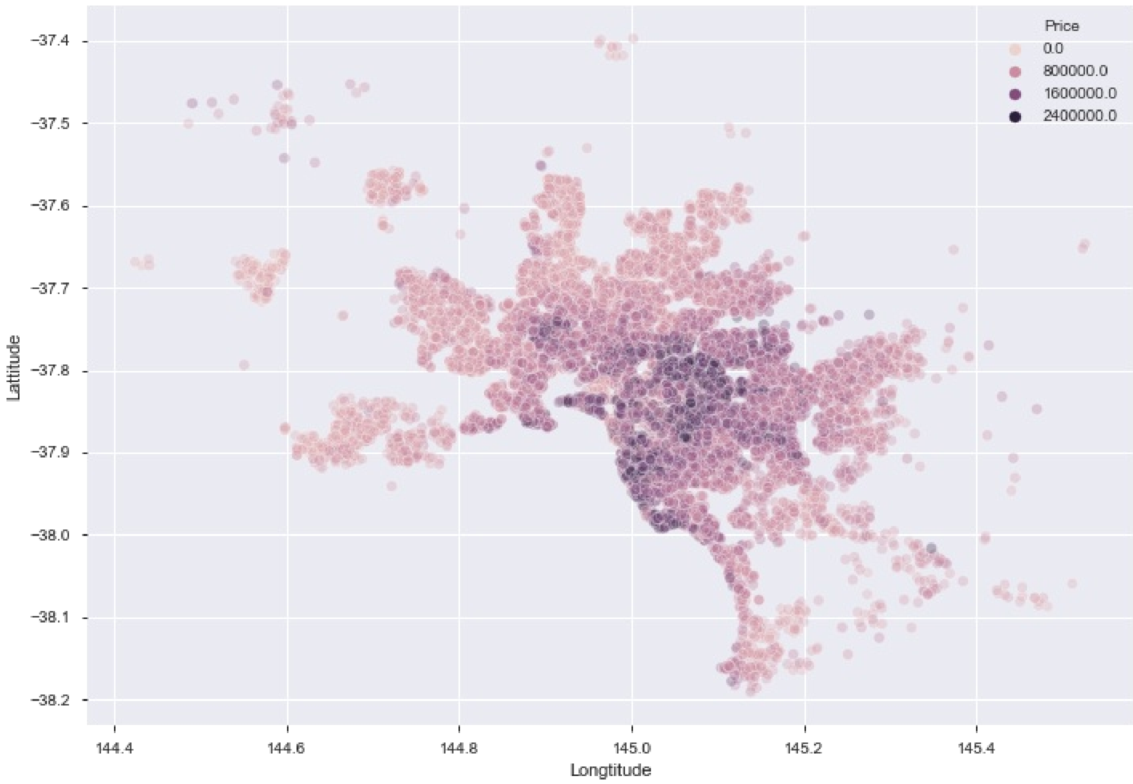
In [20]:

```
# Simple scatterplot
a4_dims = (11.7, 8.27)
fig, ax = plt.subplots(figsize=a4_dims)
sns.scatterplot(y='Lattitude',
                x='Longtitude',
                alpha=0.3,
```

```
                 alpha=0.5,
                 data=filtered, ax=ax, cmap=plt.get_cmap("viridis"), hue='Price')
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15dbf052e48>
```



Lets Try to answer the first Question:

- Which suburbs are the best to buy in?

"Best" is a very vague term and also creates the ambiguity in the buyers head as the 'Best' can be defined on the basis of the price of the house or maybe at the basis of the number of houses in the suburbs

Now lets dig deeper into the concept of "Best" suburb on the basis of average price of houses of the respective Suburb

Lets get the Top 10 suburbs on the basis of average price of houses of the respective Suburb

In [21]:

```
suburb_group_mean = filtered.groupby(by = 'Suburb', as_index=False).mean()
suburb_group_mean = suburb_group_mean.reset_index()
suburb_group_mean = suburb_group_mean.sort_values(by =['Price'], ascending=False, ).head(10)
# g.drop(columns=['index'], inplace = True)
suburb_group_mean.reset_index(drop=True, inplace=True)
suburb_group_mean.drop(columns=['index'], inplace = True)
lis_sub = list(suburb_group_mean['Suburb'])
```

In [22]:

```
suburb_group_mean
```

Out[22]:

	Suburb	Rooms	Price	Postcode	Lattitude	Longtitude	Month	Year	PriceMillions
0	Kew East	3.411765	1.678456e+06	3102.0	-37.793881	145.053080	7.073529	2016.794118	1.678456
1	Albert Park	2.590909	1.667371e+06	3206.0	-37.844236	144.953147	6.666667	2016.621212	1.667371

	Suburb	Rooms	Price	Postcode	Lattitude	Longtitude	Month	Year	PriceMillions
2	Balwyn North	3.584615	1.652440e+06	3104.0	-37.792707	145.085839	6.794872	2016.676923	1.652440
3	Beaumaris	3.642857	1.644420e+06	3193.0	-37.982492	145.039813	6.857143	2017.250000	1.644420
4	Sandringham	3.510204	1.638153e+06	3191.0	-37.952744	145.015562	6.959184	2017.163265	1.638153
5	Ashburton	3.283582	1.619731e+06	3147.0	-37.867235	145.080159	7.268657	2016.597015	1.619731
6	McKinnon	3.521739	1.602587e+06	3204.0	-37.909779	145.038948	7.434783	2017.173913	1.602587
7	Eaglemont	3.392857	1.590500e+06	3084.0	-37.763034	145.059754	7.928571	2016.642857	1.590500
8	Deepdene	3.000000	1.546667e+06	3103.0	-37.810400	145.066667	10.000000	2017.000000	1.546667
9	Middle Park	2.516129	1.529355e+06	3206.0	-37.850536	144.962965	5.870968	2016.677419	1.529355

In [23]:

```
suburb_group_mean = filtered[filtered["Suburb"].isin(lis_sub)]
```

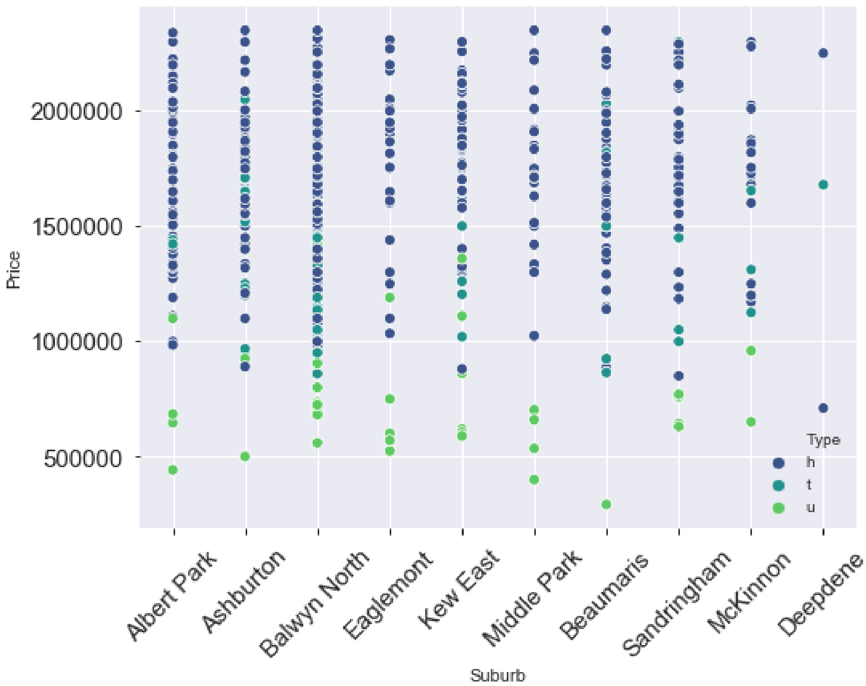
Scatter Plot for Suburb vs average price having type as a hue

In [24]:

```
fig = plt.figure(figsize=(8,6))
sns.scatterplot(x=suburb_group_mean['Suburb'],y = suburb_group_mean['Price'], palette=p4, hue=subur
b_group_mean['Type'])
plt.xticks(rotation = 45)
plt.xticks(size=15)
plt.yticks(size=15)
```

Out[24]:

```
(array([      0.,  500000., 1000000., 1500000., 2000000., 2500000.]),
 <a list of 6 Text yticklabel objects>)
```



as you can see when it comes to type **h(Houses)** are the most common and famous ones then comes t and then comes u.

Scatter Plot for Suburb vs average price having Rooms as a hue

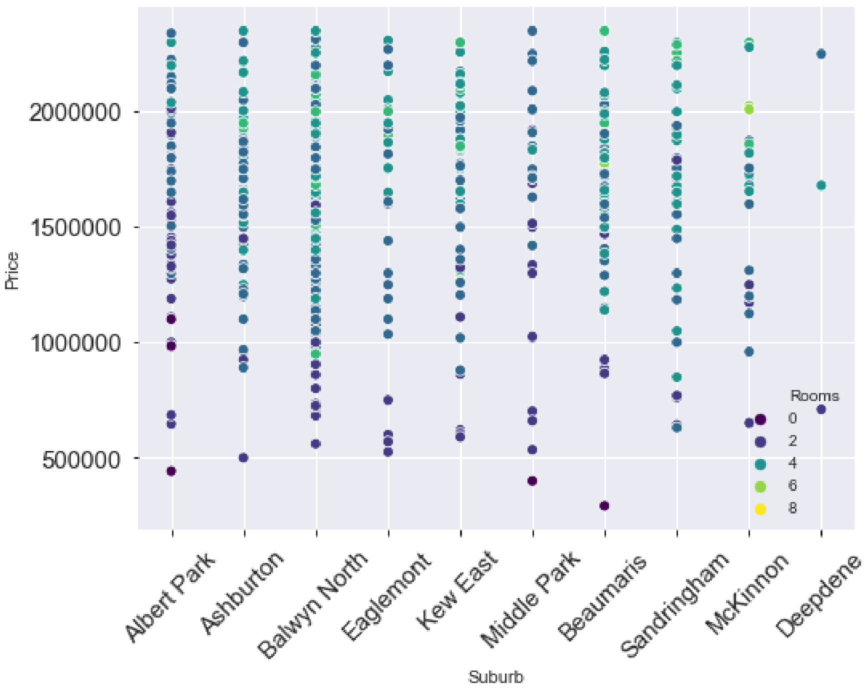
In [25]:

```
fig = plt.figure(figsize=(8,6))
sns.scatterplot(x=suburb_group_mean['Suburb'],y = suburb_group_mean['Price'], palette=p4, hue=subur
b_group_mean['Rooms'])
plt.xticks(rotation = 45)
```

```
plt.xticks(rotation=45)
plt.xticks(size=15)
plt.yticks(size=15)
```

Out[25]:

```
(array([      0.,  500000., 1000000., 1500000., 2000000., 2500000.]),
 <a list of 6 Text yticklabel objects>)
```



as you can see when it comes to type **2 Rooms** are one of the most common but not the most famous ones there are very 8 rooms houses and the most common is 4 rooms houses

Scatter Plot for Suburb vs average price on the basis of lattitude and longtitude

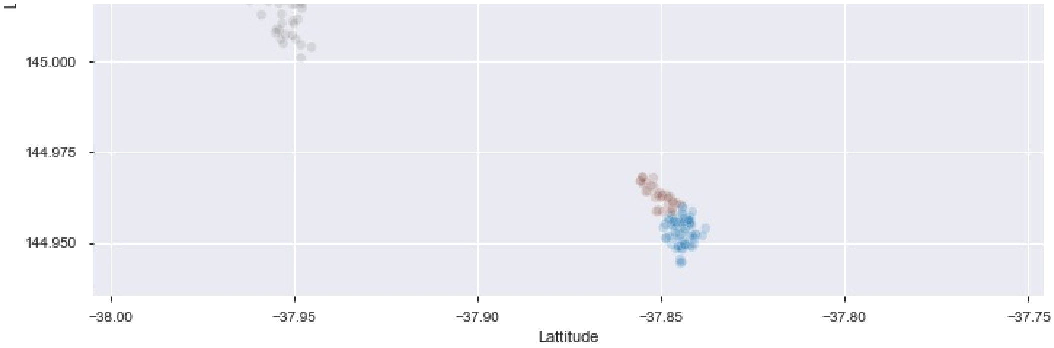
In [26]:

```
# Plot districts

a4_dims = (11.7, 8.27)
fig, ax = plt.subplots(figsize=a4_dims)
sns.scatterplot(x='Lattitude',
                y='Longtitude',
                 hue='Suburb',
                 alpha=0.2,
                data=suburb_group_mean, ax=ax)
plt.title("TOP 10 SUBURBS ACCORDING TO THE MEAN PRICE OF THE HOUSES")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, fontsize=15)
```

Out[26]:

```
<matplotlib.legend.Legend at 0x15dbee67358>
```



Now lets dig deeper into the concept of "Best" suburb on the basis of Number of houses of the respective Suburb

Lets get the Top 10 suburbs on the basis of Number of price of houses of the respective Suburb

In [27]:

```
suburb_group_count= filtered.groupby(by = 'Suburb', as_index=False).count()
suburb_group_count = suburb_group_count.reset_index()
suburb_group_count = suburb_group_count.sort_values(by =['Price'], ascending=False, ).head(10)
# g.drop(columns=['index'], inplace = True)
suburb_group_count.reset_index(drop=True, inplace=True)
suburb_group_count.drop(columns=['index'], inplace = True)
lis_sub = list(suburb_group_count['Suburb'])
```

In [28]:

```
suburb_group_count
```

Out[28]:

	Suburb	Rooms	Type	Price	Method	SellerG	Date	Postcode	CouncilArea	Lattitude	Longtitude	Regionname	Month	Year	F
0	Reservoir	500	500	500	500	500	500	500	500	500	500	500	500	500	
1	Bentleigh East	353	353	353	353	353	353	353	353	353	353	353	353	353	
2	Richmond	325	325	325	325	325	325	325	325	325	325	325	325	325	
3	Preston	324	324	324	324	324	324	324	324	324	324	324	324	324	
4	Brunswick	308	308	308	308	308	308	308	308	308	308	308	308	308	
5	Essendon	276	276	276	276	276	276	276	276	276	276	276	276	276	
6	Coburg	268	268	268	268	268	268	268	268	268	268	268	268	268	
7	Northcote	265	265	265	265	265	265	265	265	265	265	265	265	265	
8	South Yarra	232	232	232	232	232	232	232	232	232	232	232	232	232	
9	Glenroy	226	226	226	226	226	226	226	226	226	226	226	226	226	

In [29]:

```
suburb_group_count = filtered[filtered['Suburb'].isin(lis_sub)]
```

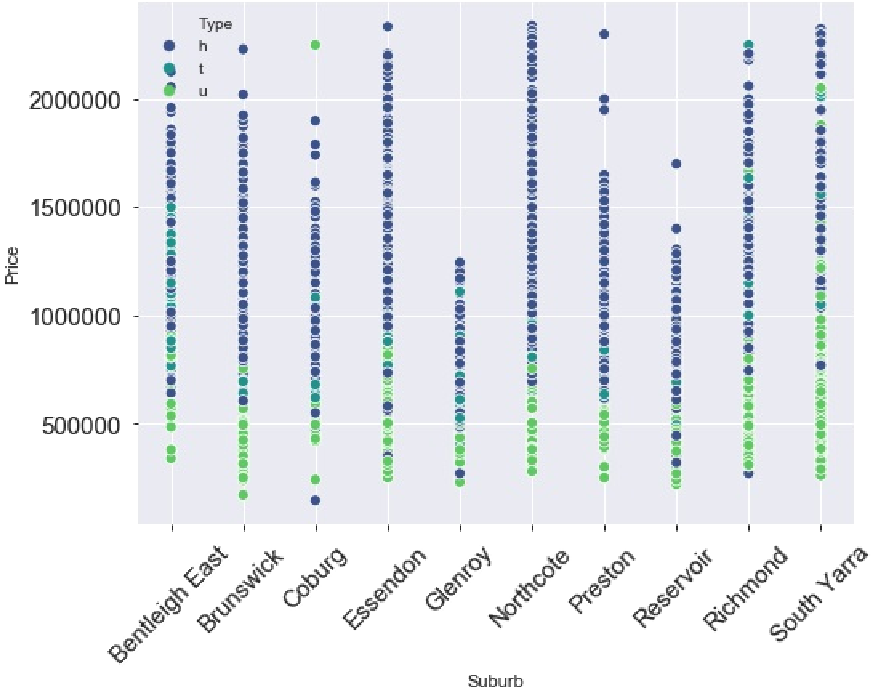
Scatter Plot for Suburb vs Number of price having type as a hue

In [30]:

```
fig = plt.figure(figsize=(8,6))
sns.scatterplot(x=suburb_group_count['Suburb'],y = suburb_group_count['Price'], palette=p4, hue=sub
urb_group_count['Type'])
plt.xticks(rotation = 45)
plt.xticks(size=15)
plt.yticks(size=15)
```

Out[30]:

```
(array([      0.,  500000., 1000000., 1500000., 2000000., 2500000.]),
 <a list of 6 Text yticklabel objects>)
```



as you can see when it comes to type h(Houses) are the most common and famous ones then comes t and then comes u.

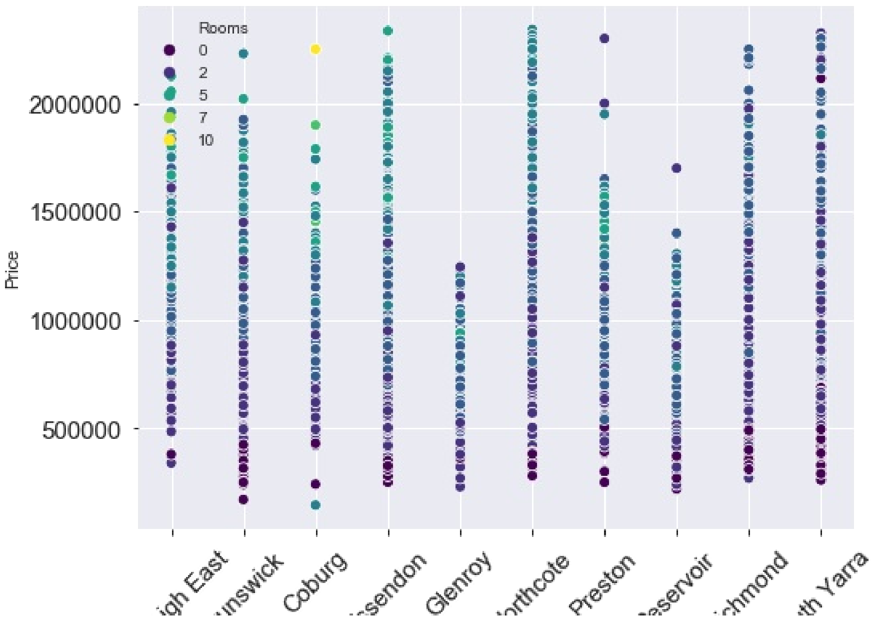
Scatter Plot for Suburb vs Number of price having Rooms as a hue

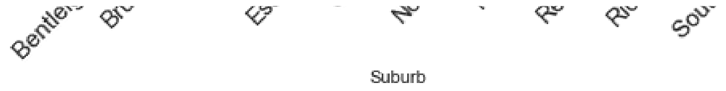
In [31]:

```
fig = plt.figure(figsize=(8,6))
sns.scatterplot(x=suburb_group_count['Suburb'],y = suburb_group_count['Price'], palette=p4, hue=sub
urb_group_count['Rooms'])
plt.xticks(rotation = 45)
plt.xticks(size=15)
plt.yticks(size=15)
```

Out[31]:

```
(array([      0.,  500000., 1000000., 1500000., 2000000., 2500000.]),
 <a list of 6 Text yticklabel objects>)
```





Hmm...Interesting, as we can there are noticable amount of houses which has 0 rooms in them. And only few suburbs contains houses with 8 rooms.

Scatter Plot for Suburb vs Number of price on the basis of lattitude and longtitude

In [32]:

```
# Plot districts

a4_dims = (11.7, 8.27)
fig, ax = plt.subplots(figsize=a4_dims)
sns.scatterplot(x='Lattitude',
                y='Longtitude',
                 hue='Suburb',
                 alpha=0.2,
                data=suburb_group_count, ax=ax)
plt.title("TOP 10 SUBURBS ACCORDING TO THE NUMBER OF HOUSES")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, fontsize=15)
```

Out[32]:

```
<matplotlib.legend.Legend at 0x15dc0586978>
```

Great, i hope that helped you to get some insights regarding the Suburbs, Now lets plot these suburbs assumptions geographically using folium to help view things better

TOP 10 Suburbs vs Average Price Geographical Representation

In [33]:

```
# Create basic Folium crime map
melbourne_map = folium.Map(location=[-37.840935,144.946457],
                        tiles = "Stamen Toner",
```

```
                      zoom_start = 12)

# Add data for heatmp
# data_heatmap = g[g.Year == 2017]
data_heatmap = suburb_group_mean[['Lattitude','Longtitude']]
data_heatmap = suburb_group_mean.dropna(axis=0, subset=['Lattitude','Longtitude'])
data_heatmap = [[row['Lattitude'],row['Longtitude']] for index, row in data_heatmap.iterrows()]
HeatMap(data_heatmap, radius=10, name = 'Sajal').add_to(melbourne_map)
# data_heatmap
# Plot!
melbourne_map
```

Out[33]:

TOP 10 Suburbs vs Number of Houses Geographical Representation

In [34]:

```
# Create basic Folium crime map
melbourne_map = folium.Map(location=[-37.840935,144.946457],
                       tiles = "Stamen Toner",
                      zoom_start = 11)

# Add data for heatmp
# data_heatmap = g[g.Year == 2017]
data_heatmap = suburb_group_count[['Lattitude','Longtitude']]
data_heatmap = suburb_group_count.dropna(axis=0, subset=['Lattitude','Longtitude'])
data_heatmap = [[row['Lattitude'],row['Longtitude']] for index, row in data_heatmap.iterrows()]
HeatMap(data_heatmap, radius=10).add_to(melbourne_map)

# Plot!
melbourne_map
```

Out[34]:

All the suburbs Geographiacal representation

In [35]:

```
# Create basic Folium crime map
melbourne_map = folium.Map(location=[-37.840935,144.946457],
                        tiles = "Stamen Toner",
                       zoom_start = 11)

# Add data for heatmp
# data_heatmap = g[g.Year == 2017]
data_heatmap = filtered[['Lattitude','Longtitude']]
data_heatmap = filtered.dropna(axis=0, subset=['Lattitude','Longtitude'])
data_heatmap = [[row['Lattitude'],row['Longtitude']] for index, row in data_heatmap.iterrows()]
HeatMap(data_heatmap, radius=10).add_to(melbourne_map)

# Plot!
melbourne_map
```

Out[35]:

REPRESENTATION OF THE SUBURBS HAING TYPE AS 'H'

In [36]:

```
import folium
from folium.plugins import HeatMap

map_hooray = folium.Map(location=[-37.840935,144.946457],
                    zoom_start = 12, min_zoom=12, tiles= "Stamen Toner" ) #Giving the location just
write boston coordinat to google

heat_df = filtered[filtered['Type']=='h']# I take 2017 cause there is more crime against to other y
ears
# heat_df = data[data['Group']=='Larceny']
heat_df = heat_df[['Lattitude', 'Longtitude']] #giving only latitude and longitude now in heat_df
just latitude and longitude
                                        #from 2017 larceny responde
heat_df=heat_df.dropna()
folium.CircleMarker([-37.840935,144.946457],
                    radius=100,
                    popup='Homicide',
                    color='red',
                    ).add_to(map_hooray) #Adding mark on the map but it's hard to find correct plac
e.
                                         #it's take to muhc time


heat_data = [[row['Lattitude'],row['Longtitude']] for index, row in heat_df.iterrows()]
#We have to give latitude and longitude like this [[lat, lon],[lat, lon],[lat, lon],[lat, lon],[la
t, lon]]

HeatMap(heat_data, radius=10).add_to(map_hooray) #Adding map_hooray to HeatMap
map_hooray #Plotting
```

Out[36]:

REPRESENTATION OF THE SUBURBS HAING TYPE AS 'T'

In [37]:

```
import folium
from folium.plugins import HeatMap

map_hooray = folium.Map(location=[-37.840935,144.946457],
                    zoom_start = 12, min_zoom=12, tiles= "Stamen Toner" ) #Giving the location just
write boston coordinat to google

heat_df = data_priced[data_priced['Type']=='t']# I take 2017 cause there is more crime against to c
ther years
```

```
ther years
# heat_df = data[data['Group']=='Larceny']
heat_df = heat_df[['Lattitude', 'Longtitude']] #giving only latitude and longitude now in heat_df
just latitude and longitude
                                        #from 2017 larceny responde
heat_df=heat_df.dropna()
# folium.CircleMarker([-37.840935,144.946457],
#                      radius=50,
#                      popup='Homicide',
#                      color='red',
#                      ).add_to(map_hooray) #Adding mark on the map but it's hard to find correct pl
ace.
                                         #it's take to muhc time


heat_data = [[row['Lattitude'],row['Longtitude']] for index, row in heat_df.iterrows()]
#We have to give latitude and longitude like this [[lat, lon],[lat, lon],[lat, lon],[lat, lon],[la
t, lon]]

HeatMap(heat_data, radius=10).add_to(map_hooray) #Adding map_hooray to HeatMap
map_hooray #Plotting
```

Out[37]:

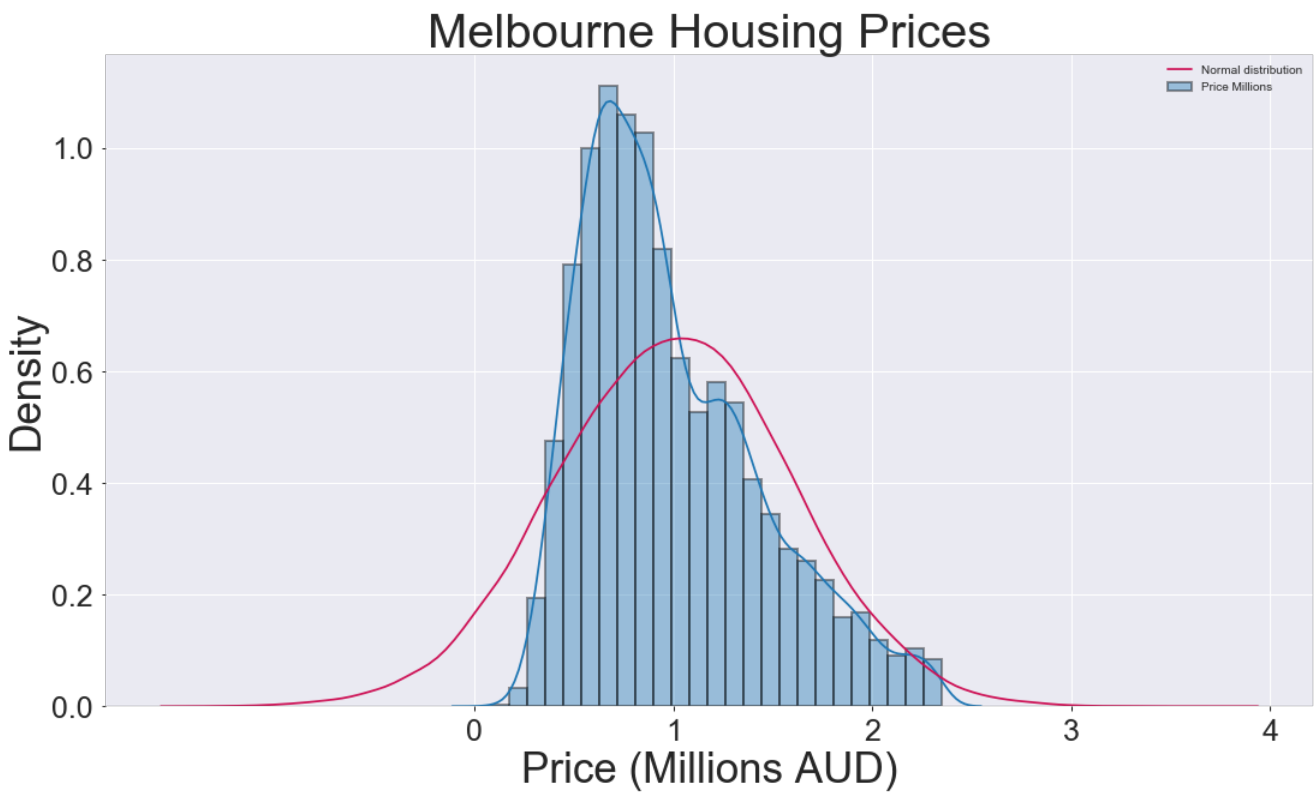
Enough with the suburbs, lets get to some statistical and categorical graphs

In [38]:

```
# Normal distribution with same std and mean as price data
normal = np.random.normal(loc=.9978982,scale=.5934989,size=48433)

# Histogram settings
plt.figure(figsize = (18, 10))
plt.rcParams["axes.labelsize"] = 35
plt.title('Melbourne Housing Prices', fontsize=40)
plt.ylabel('Density')

plt.xticks(np.arange(12), ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11'))
price = sns.distplot(filtered['PriceMillions'], bins=25, label='Price Millions', axlabel='Price (Mi
llions AUD)', hist_kws=dict(edgecolor="k", linewidth=2))
normalprice = sns.distplot(normal, hist=False, bins=25, color='#ce1256', label='Normal distribution
')
normalprice.tick_params(labelsize=25)
plt.legend(fontsize = 10)
plt.show()
```

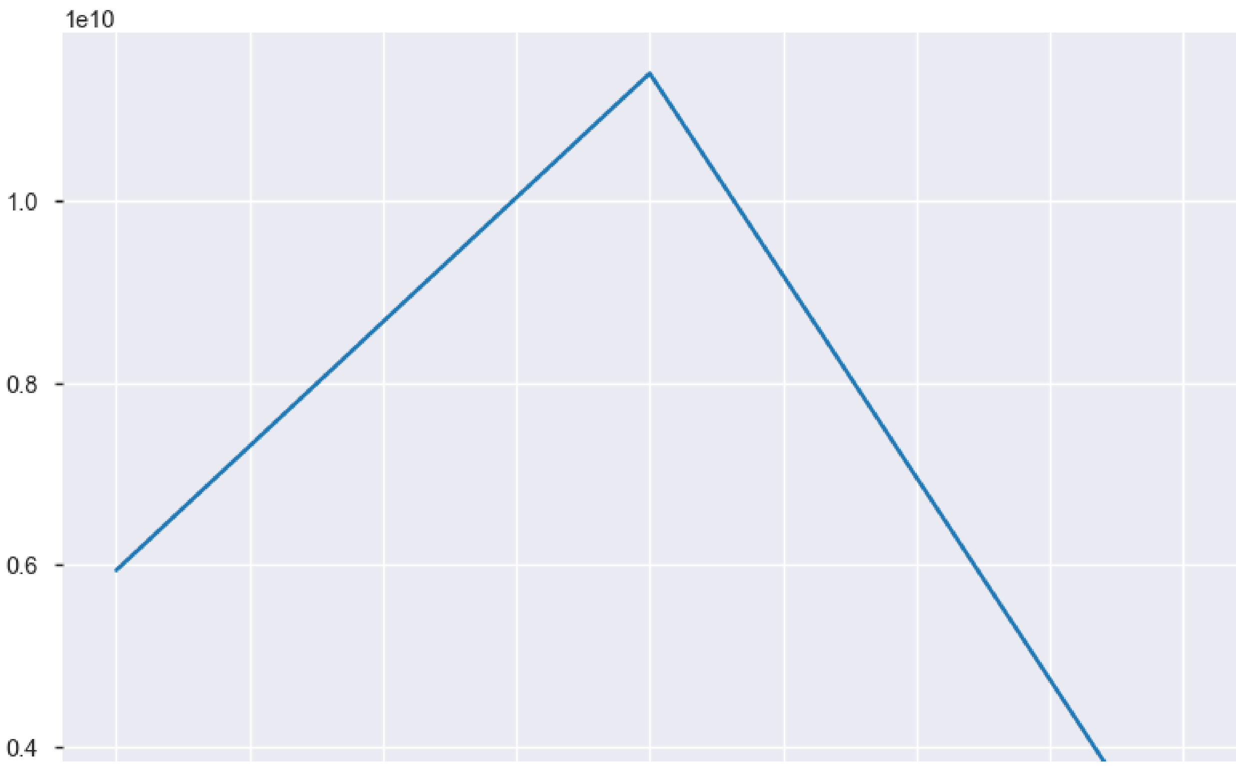


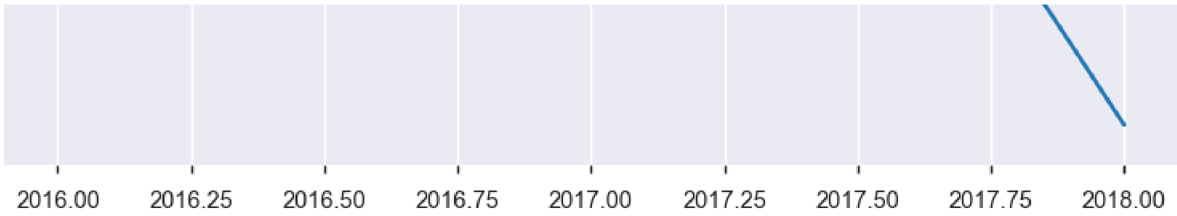
This graph shows the Histogram plot of the Price in Millions having KDE(Kernel Distribution Estimatio) plot within which best represent the data distribustion and moreover the red line is of normal distribution

Change in the seling Prices over the span of 2 years from 2016-2018

In [39]:

```
sell_price_each_year = filtered.groupby(by = ['Year'])['Price'].sum()
plt.figure(figsize=(9,7), dpi=105)
# plt.xticks(ticks=['2016', '2017', '2018'], ('2016', '2017', '2018'))
plt.plot(sell_price_each_year)
plt.show()
```





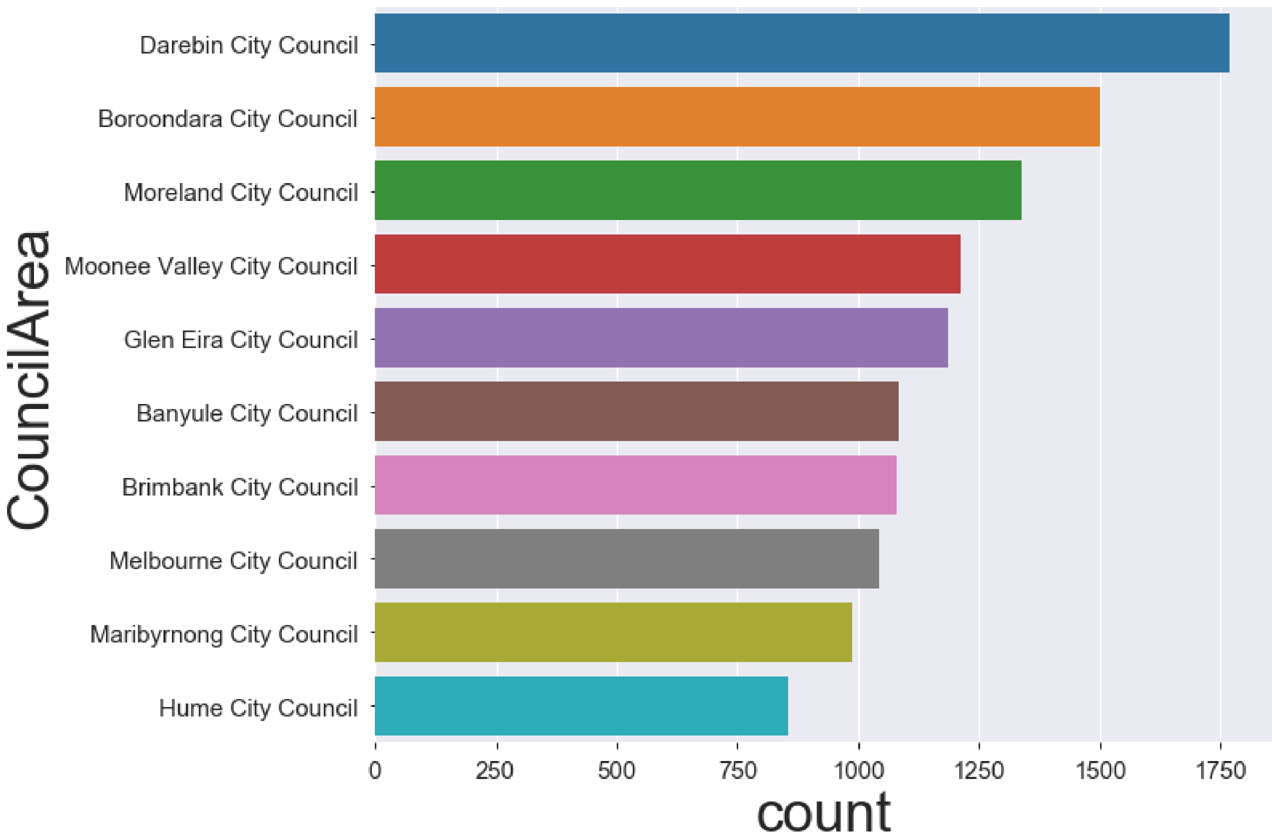
Lets do some analysis with the coucil area and region name

In [40]:

```
sns.catplot(y='CouncilArea', kind='count', height=7, aspect=1.5, order=filtered.CouncilArea.value_c
ounts().iloc[:10].index,  data=filtered)
plt.xticks(size=15)
plt.yticks(size=15)
```

Out[40]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]), <a list of 10 Text yticklabel objects>)
```



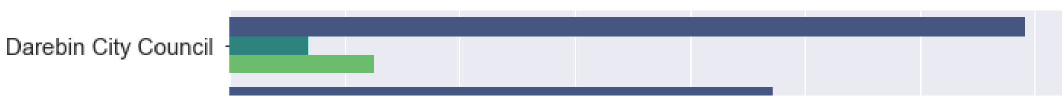
The Above graph shows the top 10 council area having the most number of homes

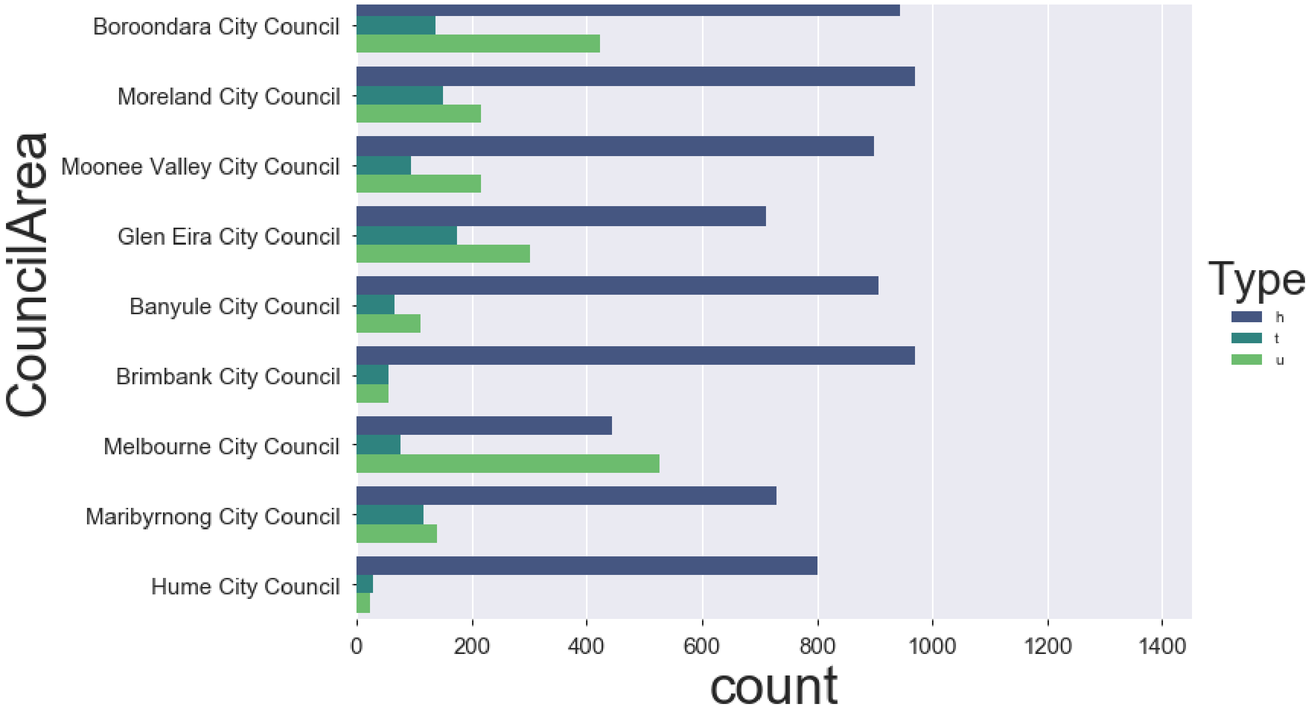
In [41]:

```
sns.catplot(y='CouncilArea',hue = ('Type'), kind='count', height=7, aspect=1.5, order=filtered.Coun
cilArea.value_counts().iloc[:10].index,  data=filtered, palette=p4)
plt.xticks(size=15)
plt.yticks(size=15)
```

Out[41]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]), <a list of 10 Text yticklabel objects>)
```





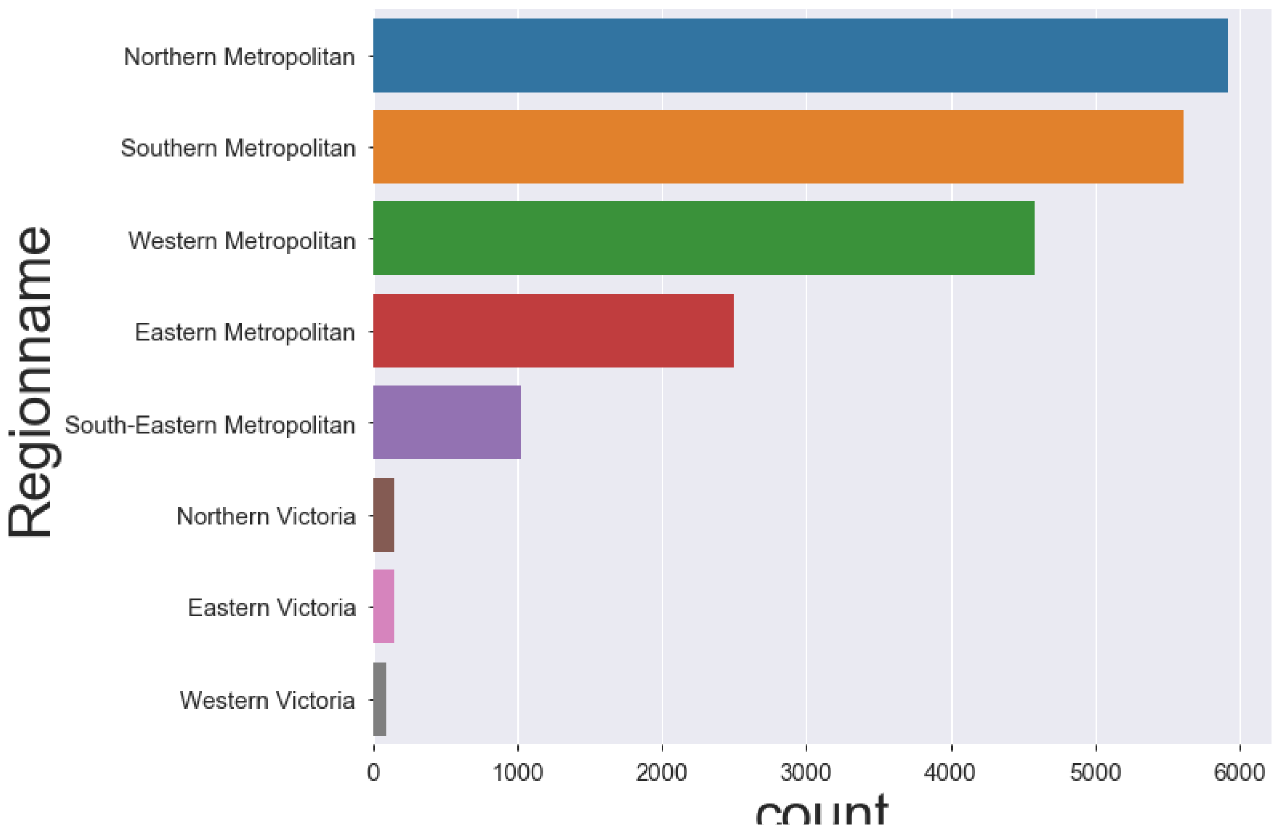
The Above graph shows the same but this time having Type as the hue, as you can see
Darebin City Council has the most houses but when it comes to the units or townhouse the conclusion isnt the same.
Melbourne City Council has the most units
*** Glen Eira and Moreland City Council has the same amout of toenhouses in the Melbourne**

In [42]:

```
sns.catplot(y='Regionname', kind='count', height=7, aspect=1.5, order=filtered.Regionname.value_cou
nts().index,  data=filtered)
plt.xticks(size=15)
plt.yticks(size=15)
```

Out[42]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7]), <a list of 8 Text yticklabel objects>)
```



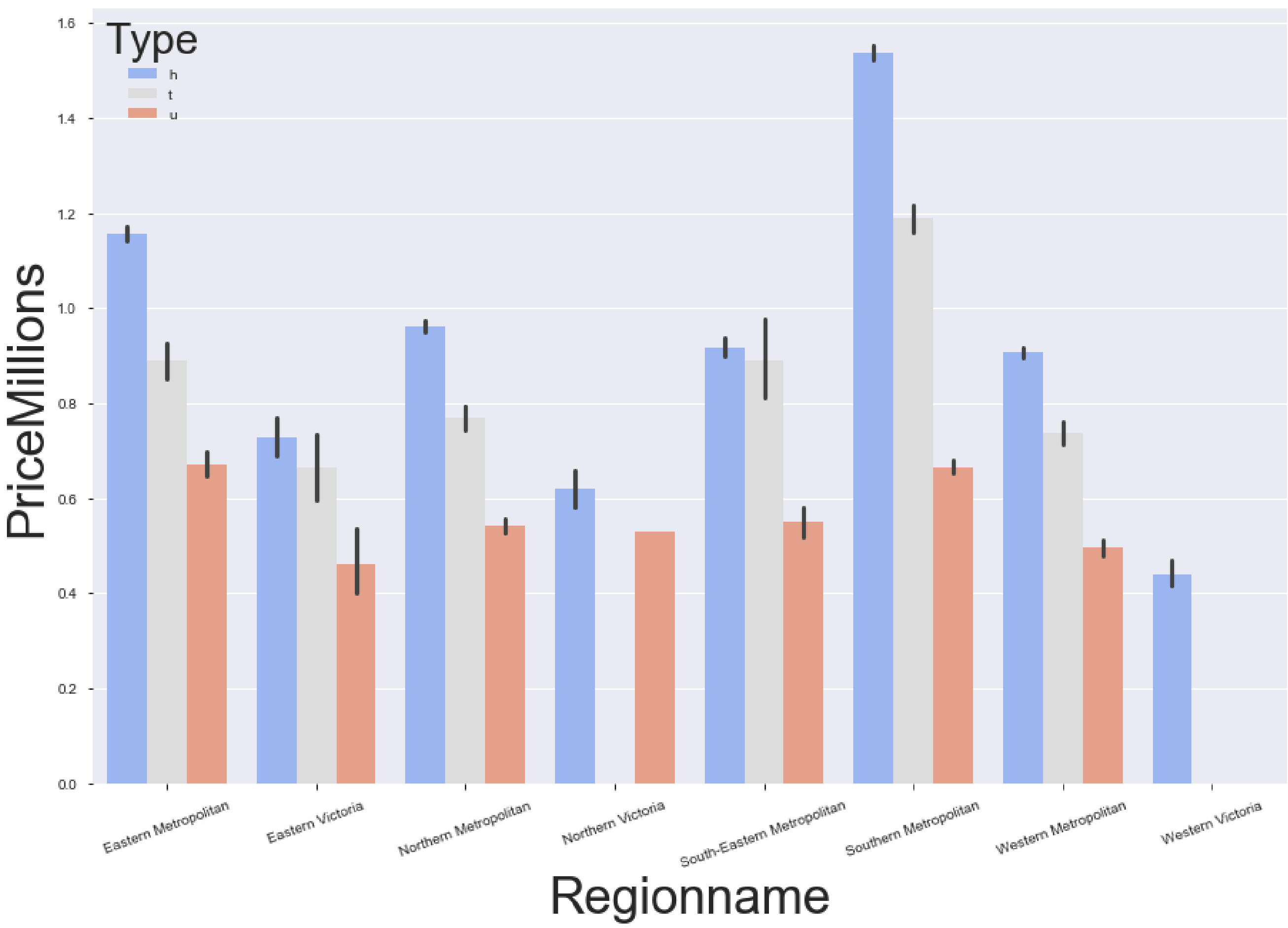
The Above Plot Shows All the Regioname In ordered form Having most number of houses from most to least

In [43]:

```
fig = plt.figure(figsize=(15,10))
ax = sns.barplot(x="Regionname", y="PriceMillions", data=filtered, estimator=np.mean, hue='Type', p
alette=p5)
plt.xticks(rotation = 20)
```

Out[43]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7]), <a list of 8 Text xticklabel objects>)
```



The Above graph shows the same but this time having Type as the hue, as you can see
Southern Melbourne has the most houses but when it comes to the units or townhouse the conclusion isnt the same.
Eastern Melbourne has the most units
*** Southern Melbourne has the most amout of townhouses in the Melbourne**

Lets Talk about rooms how the price gets affected with number of rooms

In [44]:

```
room_price = filtered.groupby(by = 'Rooms', as_index=False).mean()
room_price = room_price.reset_index()
room_price = room_price.sort_values(by =['Price'], ascending=False, ).head(10)
# g.drop(columns=['index'], inplace = True)
room_price.reset_index(drop=True, inplace=True)
room_price.drop(columns=['index'], inplace = True)
```

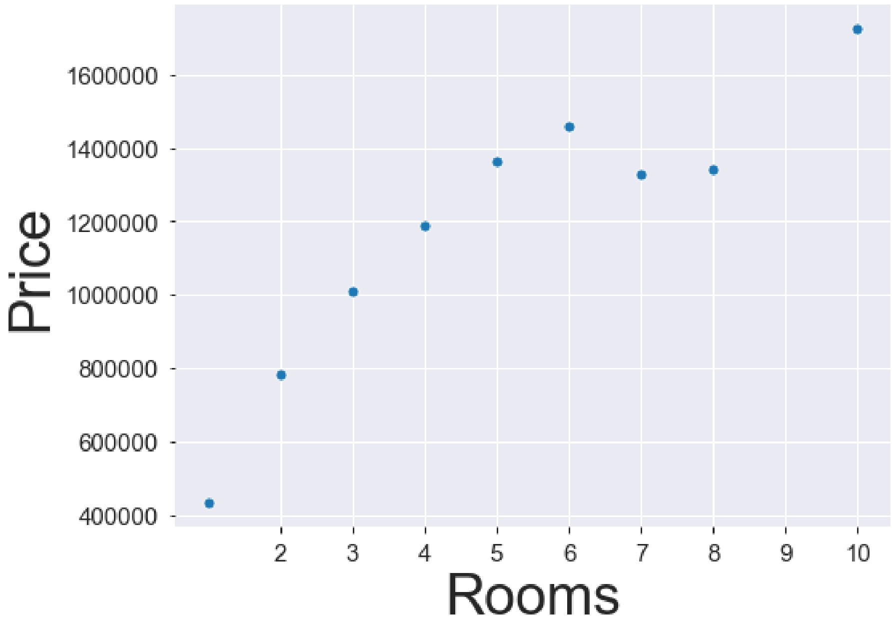
SCATER PLOT OF ROOMS VS PRICE

In [45]:

```
room_price =room_price[room_price['Rooms']<=10]
fig = plt.figure(figsize=(8,6))
sns.scatterplot(x=room_price['Rooms'],y = room_price['Price'], palette=p4)
plt.xticks(np.arange(2,11))
plt.xticks(size=15)
plt.yticks(size=15)
```

Out[45]:

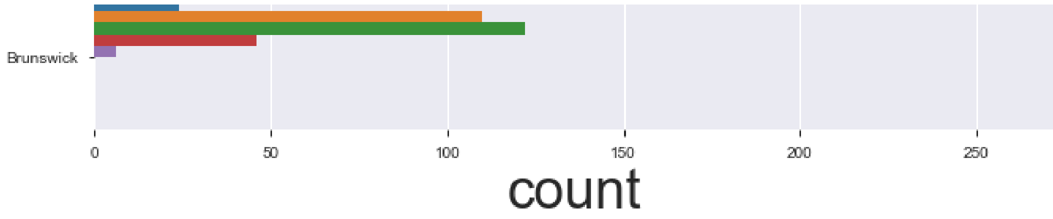
```
(array([ 200000.,  400000.,  600000.,  800000., 1000000., 1200000.,
        1400000., 1600000., 1800000.]), <a list of 9 Text yticklabel objects>)
```



In [46]:

```
# fig
fig = plt.figure(figsize=(10,8))
# plt.barh(y = f['Suburb'], width = f['Price'])
sns.catplot(y='Suburb', kind='count', height=7, aspect=1.5, order=filtered.Suburb.value_counts().il
oc[:5].index,  data=filtered, hue = 'Rooms')
plt.title("TOP 5 SUBURBS WITH ROOMS AS HUE")
plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```



Below Graph shows the number of houses sold in each month in the span of 2 years

In [47]:

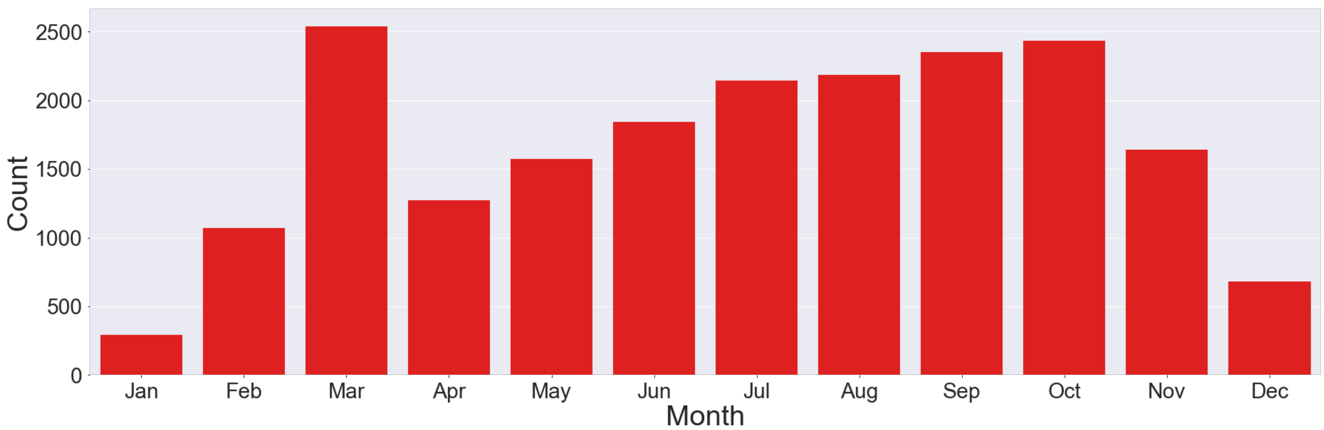
```
Months = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']

sns.catplot(x='Month',
           kind='count',
            height=8.27,
            aspect=3,
            color='red',
           data=filtered)
plt.xticks(np.arange(12), Months, size=30)

# plt.xticks(size=30)
plt.yticks(size=30)
plt.xlabel('Month', fontsize=40)
plt.ylabel('Count', fontsize=40)
```

Out[47]:

```
Text(-1.4500000000000028, 0.5, 'Count')
```



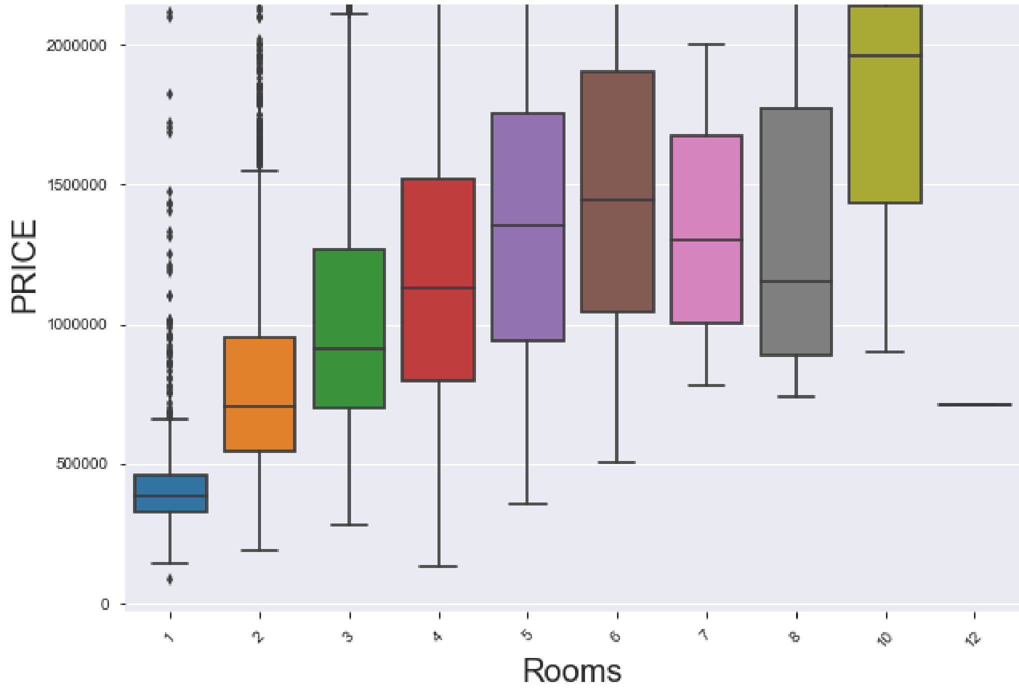
As you can see that the most houses were sold in the month of March followed by september and october later in those years.

In [48]:

```
fig = plt.figure(figsize=(10,8))
sns.boxplot(x = 'Rooms', y = 'Price', data = filtered)
plt.xlabel('Rooms', fontsize = 20)
plt.ylabel('PRICE', fontsize = 20)
plt.xticks(rotation = 45)
#axes[1,0].set_ylabel('Price')
plt.title('Number of Rooms v Price',  fontsize = 20)
```

Out[48]:

```
Text(0.5, 1.0, 'Number of Rooms v Price')
```

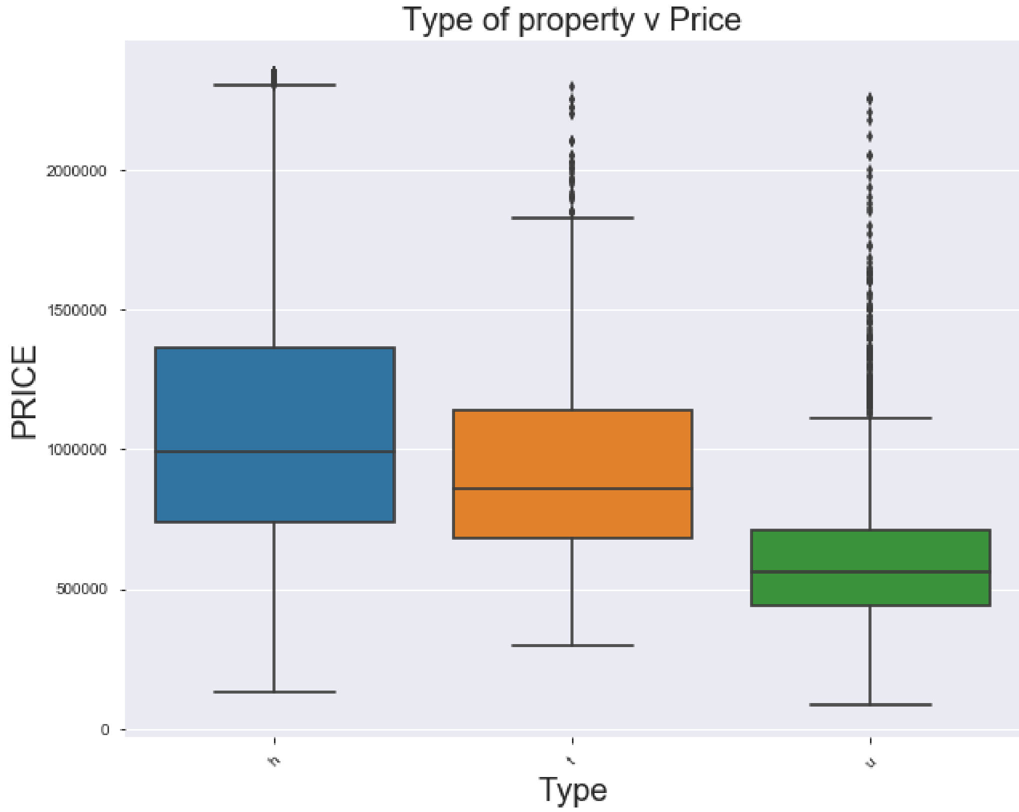


In [49]:

```
# fig, ax = plt.subplots(1, 2, figsize=(10, 4))
# ax = plt.axes()
fig = plt.figure(figsize=(10,8))
sns.boxplot(x = 'Type', y = 'Price', data = filtered)
plt.xlabel('Type', fontsize = 20)
plt.ylabel('PRICE', fontsize = 20)
plt.xticks(rotation = 45)
# ax.set_xticklabels(fontsize=12)
#axes[1,0].set_ylabel('Price')
plt.title('Type of property v Price',  fontsize = 20)
```

Out[49]:

```
Text(0.5, 1.0, 'Type of property v Price')
```

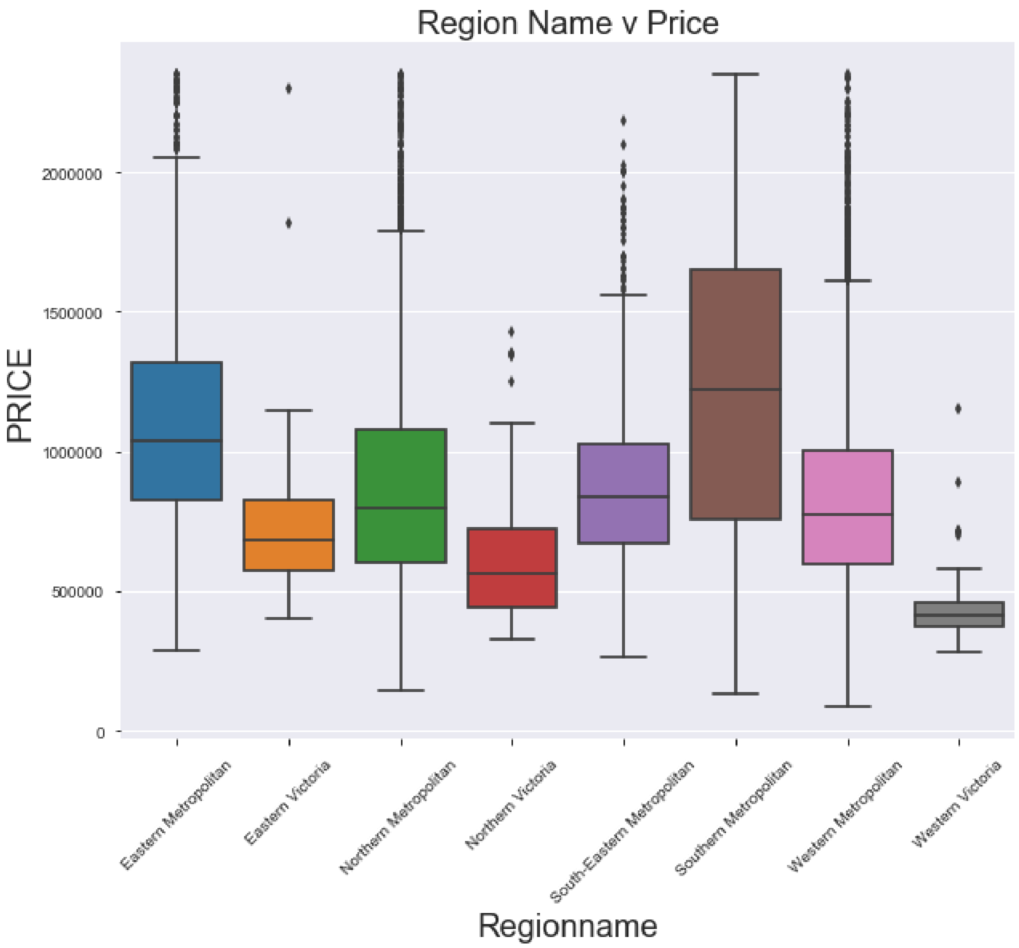


In [50]:

```
fig = plt.figure(figsize=(10,8))
sns.boxplot(x = 'Regionname', y = 'Price', data = filtered)
plt.xlabel('Regionname', fontsize = 20)
plt.ylabel('PRICE', fontsize = 20)
plt.xticks(rotation = 45)
#axes[1,0].set_ylabel('Price')
plt.title('Region Name v Price',  fontsize = 20)
```

Out[50]:

```
Text(0.5, 1.0, 'Region Name v Price')
```

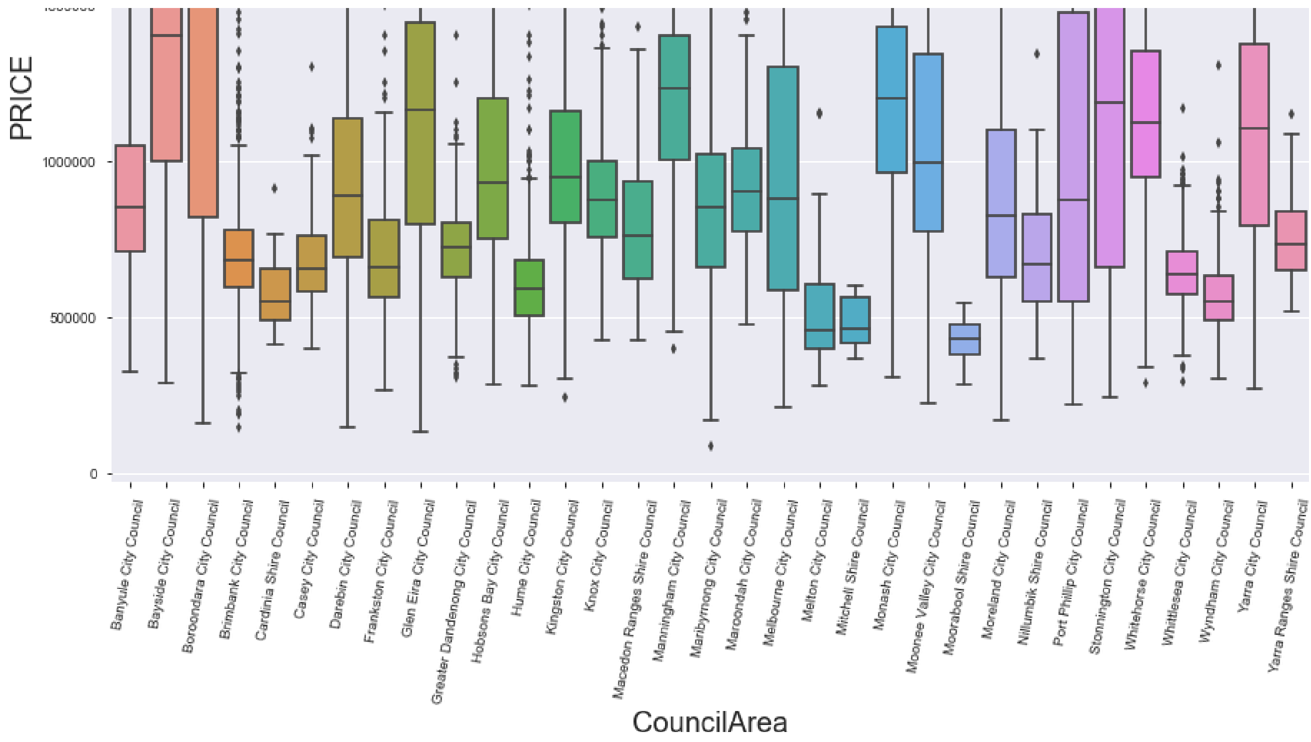


In [51]:

```
fig = plt.figure(figsize=(15,10))
sns.boxplot(x = 'CouncilArea', y = 'Price', data = filtered)
plt.xlabel('CouncilArea', fontsize = 20)
plt.ylabel('PRICE', fontsize = 20)
plt.xticks(rotation = 80)
#axes[1,0].set_ylabel('Price')
plt.title('Council Area v Price',  fontsize = 20)
```

Out[51]:

```
Text(0.5, 1.0, 'Council Area v Price')
```



Insights

- Median prices for houses are over $1M, townhomes are $800k - $900k and units are approx $500k.
- Median prices in the Metropolitan Region are higher than than that of Victoria Region - with Southern Metro being the area with the highest median home price ~$1.3M

- Median Prices in the Boroondara City Council are higher than the others

Conclusions:

In Summary, This EDA shows:

- The dataset has some some analomies and lots of missing which were handeled by simply dropping though there are many advance way to deal with them
- we have detected all the outliers using the concept of Inter Quartile Range and the distribution of the data consistent.
- There are not many correlations between the variables of the dataset apart from number of rooms and prices
- Kew East and Albert Park are two of the most expensive suburbs when it comes to the average prices of houses.
- Reservoir and Betleigh East are two of the most common suburbs havingv the maximum number of houses.
- The Prices of the House increased linearly from 2016-2017 and decreased linearly from 2017-2018 with around same slope.
- When it comes to the council areas Darebin and Boroondara are the most common.
- And When it comes to the Nothern and southern Metropolitan are the most common.
- The scatter plot between rooms and prices shows that the prices of a house increase with increase in the number of rooms until 6 rooms and then we noticed that there is a price decreament for 7 and 8 rooms, 10 rooms being an exception.
- Most houses were sold in the month of March followed by september and october later in those years.
- Median prices for houses are over $1M, townhomes are $800k - $900k and units are approx $500k.
- Median Prices in the Boroondara City Council are higher than the others

This EDA just scratches the surface of the dataset. Further analyses could explore how the pricesof different types of Houses vary in time and space.