Dubai Rental Dataset Analysis

Purpose of work:

- To investigate trends, patterns and insights from the dataset on rentals in the city of Dubai and
- To predict rental values based on features in the dataset using machine learning models.

Dataset

Dataset Overview

Each entry in the dataset represents a rental property listing with details about the property's features, rental terms, and location specifics. This primary and unique dataset is designed for analysis and can be used to generate insights into the rental market dynamics of the UAE.

Columns Description

Address: Full address of the property.
Rent: The annual rent price in AED.
Beds: Number of bedrooms in the property.
Baths: Number of bathrooms in the property.
Type: Type of property (e.g., Apartment, Villa, Penthouse).
Area_in_sqft: Total area of the property in square feet.
Rent_per_sqft: Rent price per square foot, calculated as Rent divided by Area_in_sqft.
Rent_category: Categorization of the rent price (Low, Medium, High) based on thresholds.
Frequency: Rental payment frequency, which is consistently 'Yearly'.
Furnishing: Furnishing status of the property (Furnished, Unfurnished).
Purpose: The purpose of the listing, typically 'For Rent'.
Posted_date: The date the property was listed for rent.
Age_of_listing_in_days: The number of days the listing has been active since it was posted.
Location: A more specific location within the city where the property is located.
City: City in which the property is situated.
Latitude, Longitude: Geographic coordinates of the property.

1.0 Import Libraries and Data
2.0 Data Integrity Checks
3.0 Exploratory Data Analysis

```
3.1  Property type, rental category, furnishing, bedrooms and
bathrooms
3.2  Rents over time
3.3  Average rents over time
3.4  Age of listing
3.5  Location
3.6  Area
3.7  Relationships
```

4.0 Prediction

```
4.1 Data preparation
4.2 Linear Regression
4.3 Other Models
4.4 New prediction
```

1.0 Import Libraries and Data

In [996…

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy import stats
import folium
import warnings
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import neighbors
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import r2_score
warnings.filterwarnings("ignore")
```

In [287…

```
df = pd.read_csv(r'C:\Users\imoge\AllMLProjects\Data\dubai_properties.csv')
```

In [288…

```
print(df.shape)
```

```
(73742, 17)
```

In [289…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73742 entries, 0 to 73741
Data columns (total 17 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Address                73742 non-null  object 
 1   Rent                   73742 non-null  int64  
 2   Beds                   73742 non-null  int64  
 3   Baths                  73742 non-null  int64  
 4   Type                   73742 non-null  object 
 5   Area_in_sqft           73742 non-null  int64  
 6   Rent_per_sqft          73742 non-null  float64
 7   Rent_category          73742 non-null  object 
 8   Frequency              73742 non-null  object 
 9   Furnishing             73742 non-null  object 
 10  Purpose                73742 non-null  object 
 11  Posted_date            73742 non-null  object 
 12  Age_of_listing_in_days 73742 non-null  int64  
 13  Location               73742 non-null  object 
 14  City                   73742 non-null  object 
 15  Latitude               73023 non-null  float64
 16  Longitude              73023 non-null  float64
dtypes: float64(3), int64(5), object(9)
memory usage: 9.6+ MB
```

2.0 Data Integrity Checks

In [290…

```
# Check the value counts by city
df['City'].value_counts()
```

Out[290…

```
Dubai             34250
Abu Dhabi         23324
Sharjah            9516
Ajman              4704
Al Ain             1040
Ras Al Khaimah      816
Umm Al Quwain        65
Fujairah             27
Name: City, dtype: int64
```

In [291…

```
# Split out just the Dubai results
dubai = df[df['City']=='Dubai']
print(dubai.shape)
```

```
(34250, 17)
```

In [292…

```
dubai.isnull().sum()
```

Out[292…

```
Address                   0
Rent                      0
Beds                      0
Baths                     0
Type                      0
Area_in_sqft              0
Rent_per_sqft             0
Rent_category             0
Frequency                 0
Furnishing                0
Purpose                   0
Posted_date               0
Age_of_listing_in_days    0
Location                  0
City                      0
Latitude                 31
Longitude                31
dtype: int64
```

In [293…

```
# We can see all the missing items relate to one location
dubai[dubai['Latitude'].isnull()]['Location'].value_counts()
```

Out[293…

```
Dubai Science Park    31
Name: Location, dtype: int64
```

In [294…

```
# Lets replace these values with the latitude and longitude for that area
latitude = 25.0745
longitude = 55.2396

dubai['Latitude'].fillna(latitude, inplace = True)
dubai['Longitude'].fillna(longitude, inplace = True)
```

In [295…

```
# Do we have any duplicates?
dubai.duplicated().value_counts()
```

Out[295…

```
False    34250
dtype: int64
```

We don't need the purpose, frequency and city columns have only a single value so are not useful

In [296…

```
# Drop columns we dont need
dubai.drop(columns = ['Purpose','Frequency','City'],axis = 1, inplace = True)
dubai.head()
```

Out[296…

	Address	Rent	Beds	Baths	Type	Area_in_sqft	Rent_per_sqft	Rent_category	Fu
29068	Binghatti Heights, JVC District 10, Jumeirah V...	125000	2	2	Apartment	1145	109.170306	Medium	Unf
29069	Seasons Community, JVC District 15, Jumeirah V...	50000	1	2	Apartment	655	76.335878	Low	Unf
29070	Autumn 1 Block B, Autumn, Seasons Community, J...	90000	1	2	Apartment	896	100.446429	Medium	F
29071	Socio Tower A, Socio, Dubai Hills Estate, Dubai	125000	2	1	Apartment	720	173.611111	Medium	Unf
29072	Eleganz by Danube, JVC District 14, Jumeirah V...	105000	1	2	Apartment	965	108.808290	Medium	F

Rent

In [298…

```
# Descriptive Statistics
dubai.describe()
```

Out[298…

	Rent	Beds	Baths	Area_in_sqft	Rent_per_sqft	Age_of_listing_in_days
count	3.425000e+04	34250.000000	34250.000000	34250.000000	34250.000000	34250.00000
mean	2.133664e+05	1.971212	2.081518	1831.808321	132.253717	63.91226
std	4.272489e+05	1.395483	0.561300	3119.024048	70.329882	55.17620
min	0.000000e+00	0.000000	1.000000	74.000000	0.000000	12.00000
25%	8.500000e+04	1.000000	2.000000	754.000000	86.614173	27.00000

	Rent	Beds	Baths	Area_in_sqft	Rent_per_sqft	Age_of_listing_in_days
50%	1.450000e+05	2.000000	2.000000	1163.000000	120.000000	46.00000C
75%	2.300000e+05	3.000000	2.000000	1930.750000	159.997167	81.00000C
max	5.500000e+07	12.000000	11.000000	210254.000000	2182.044888	1131.00000C

The most expensive rental is 55 million dirham per year (approx $15m). The least expensive is 0. This is a bit odd as we would expect a value for all properties.
There is right skew in the rental data with the mean higher than the median and there are likely outliers (we will look at this again in a bit)

In [299...

```
# Places with zero rent
dubai[dubai['Rent']==0]
```

Out[299...

	Address	Rent	Beds	Baths	Type	Area_in_sqft	Rent_per_sqft	Rent_category	Furni
32240	Al Barsha South 2, Al Barsha South, Al Barsha,...	0	5	2	Villa	10000	0.0	Low	Unfurr
32241	Port Saeed, Deira, Dubai	0	1	2	Apartment	750	0.0	Low	Unfurr
32242	Port Saeed, Deira, Dubai	0	1	2	Apartment	850	0.0	Low	Unfurr
32243	Park Gate Residence A, Park Gate Residence, Al...	0	2	2	Apartment	1554	0.0	Low	Furr
32631	Al Barsha South 2, Al Barsha South, Al Barsha,...	0	5	2	Villa	12500	0.0	Low	Unfurr
32632	Abu Hail, Deira, Dubai	0	5	2	Villa	4500	0.0	Low	Unfurr
32633	Al Barsha 3, Al Barsha, Dubai	0	1	2	Apartment	750	0.0	Low	Furr
34160	Al Khudrawi, Shoreline Apartments, Palm Jumeir...	0	2	2	Apartment	1551	0.0	Low	Unfurr
35389	Serenia Residences West Wing,	0	4	2	Apartment	2885	0.0	Low	Furr

	Address	Rent	Beds	Baths	Type	Area_in_sqft	Rent_per_sqft	Rent_category	Furni
	Serenia Residenc...								
56079	1 Residences, Wasl 1, Al Kifaf, Bur Dubai, Dubai	0	2	2	Apartment	1439	0.0	Low	Unfurr
56080	Abu Keibal, Shoreline Apartments, Palm Jumeira...	0	3	2	Apartment	2138	0.0	Low	Unfurr
56081	Garden Homes Frond K, Garden Homes Palm Jumeir...	0	4	2	Villa	5000	0.0	Low	Unfurr
56082	Marina Residences 5, Marina Residences, Palm J...	0	2	2	Apartment	3996	0.0	Low	Furr
56083	Al Jafiliya, Dubai	0	5	2	Villa	7000	0.0	Low	Unfurr

It is not clear why these properties are listed at zero rent. Perhaps there is some kind of maintenance charge or other charge and/or they are less desirable. There are also some with a peppercorn rent of just 1 AED. We will drop these from our data as we don't know why this is the case and it is not likely to add value to the analysis to include them

In [300...

```
# Drop zero rentals and a few posted at 1 AED
dubai = dubai[dubai['Rent']>1]
dubai.shape
```

Out[300...

```
(34236, 14)
```

We should probably remove the high rent outliers also

In [516...

```
# Create a zscore column and exclude those above and below 3 standard deviations fro
dubai['RentZScore'] = stats.zscore(dubai['Rent'])
```

In [526...

```
# Exclude outliers
dubai = dubai[(dubai['RentZScore']<=3) & (dubai['RentZScore']>-3)]
```

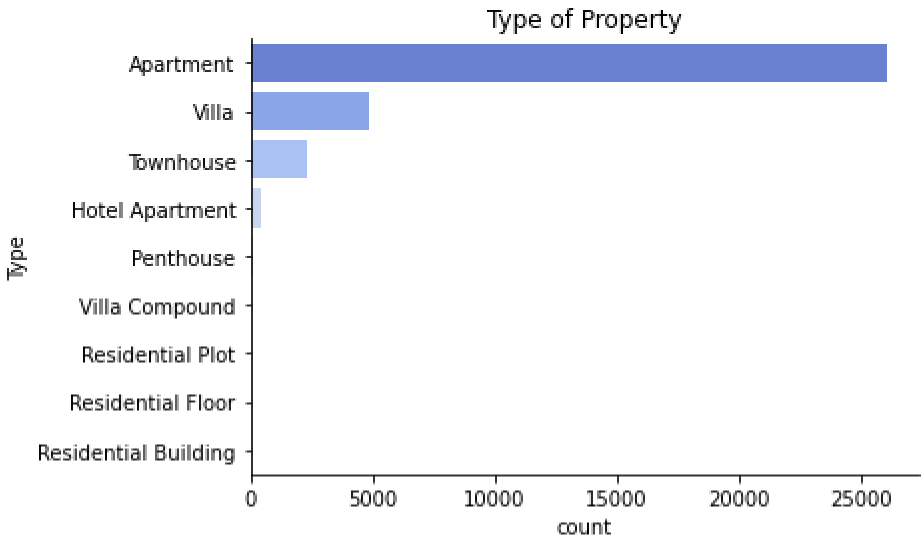
3.0 Exploratory Data Analysis

We have a mix of categorical and numerical variables that we can investigate both separately and in terms of relationships between them.

3.1 Property type, rental category, furnishing, bedrooms and bathrooms

In [527…

```
# Plot the type of property distribution
ax = sns.countplot(data = dubai, y = 'Type',palette="coolwarm" )
plt.title("Type of Property")
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



In [528…

```
dubai['Type'].value_counts(normalize = True)*100
```

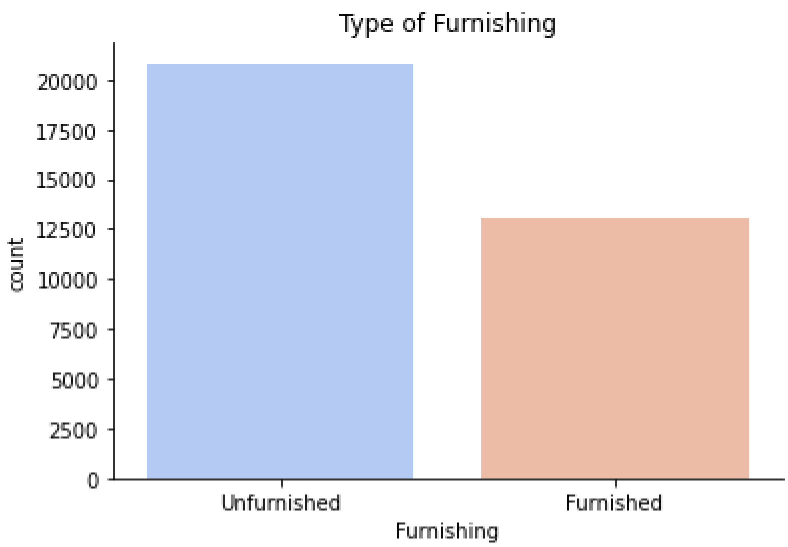
Out[528…

```
Apartment               76.993657
Villa                   14.238088
Townhouse                6.944977
Hotel Apartment          1.327629
Penthouse                0.418941
Villa Compound           0.050155
Residential Building     0.011801
Residential Floor        0.008851
Residential Plot         0.005901
Name: Type, dtype: float64
```

Some 76% of properties are apartments with a further 15% being villas

In [529…

```
# Plot the furnishing distribution
ax = sns.countplot(data = dubai, x = 'Furnishing',palette="coolwarm" )
plt.title("Type of Furnishing")
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```

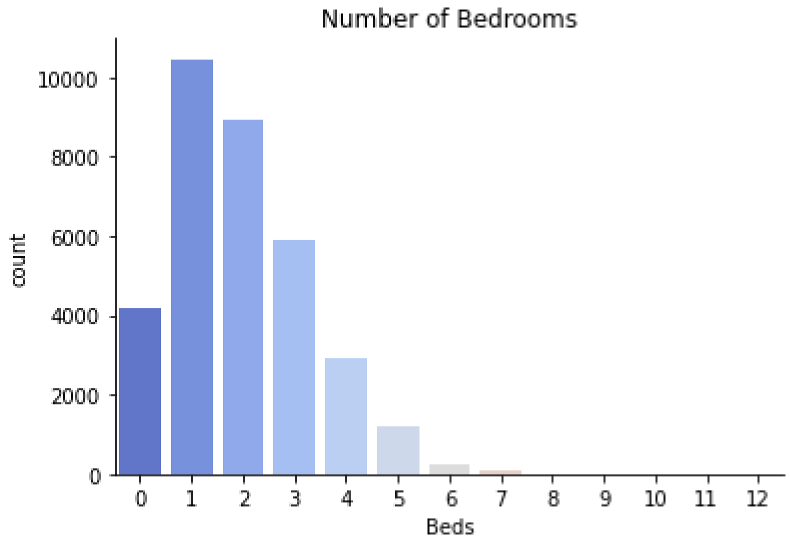


In [530…

```
# Plot the rent category distribution
ax = sns.countplot(data = dubai, x = 'Rent_category',palette="coolwarm" )
plt.title("Rent Category")
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```

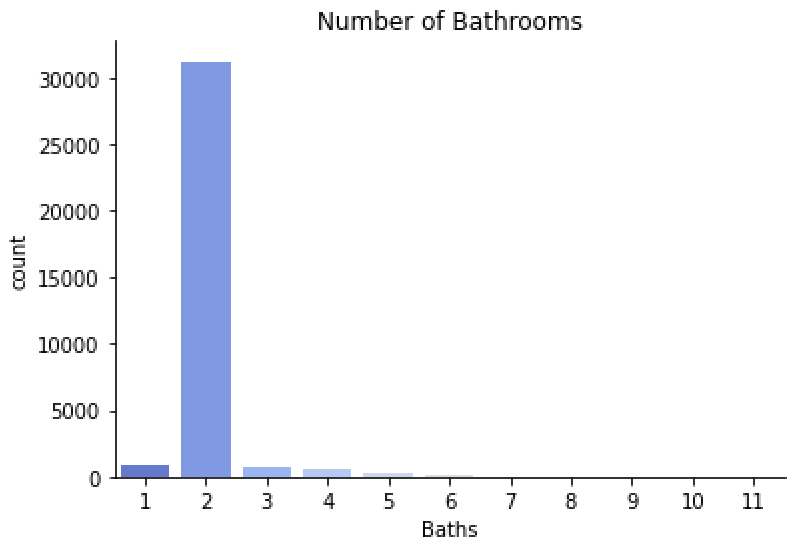
In [531…

```
# Plot the number of bedrooms distribution
ax = sns.countplot(data = dubai, x = 'Beds',palette="coolwarm" )
plt.title("Number of Bedrooms")
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



In [532...

```
# Plot the number of bathrooms distribution
ax = sns.countplot(data = dubai, x = 'Baths',palette="coolwarm" )
plt.title("Number of Bathrooms")
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



Summary

Most properties are high rental unfurnished apartments with one bedroom and two bathrooms.

3.2 Rentals Over Time

In [533...

```
# Convert the posted date to a datetime object
dubai['Posted_date'] = pd.to_datetime(dubai['Posted_date'])

# Split out the year and month for analysis
dubai['Year']= dubai['Posted_date'].dt.year
dubai['Month'] = dubai['Posted_date'].dt.month
dubai['MonthName'] = dubai['Posted_date'].dt.month_name()
```

In [534...

```
# Get the minimum and maximum posting date
print(dubai['Posted_date'].min())
print(dubai['Posted_date'].max())
```

```
2021-03-17 00:00:00
2024-04-09 00:00:00
```

In [535…

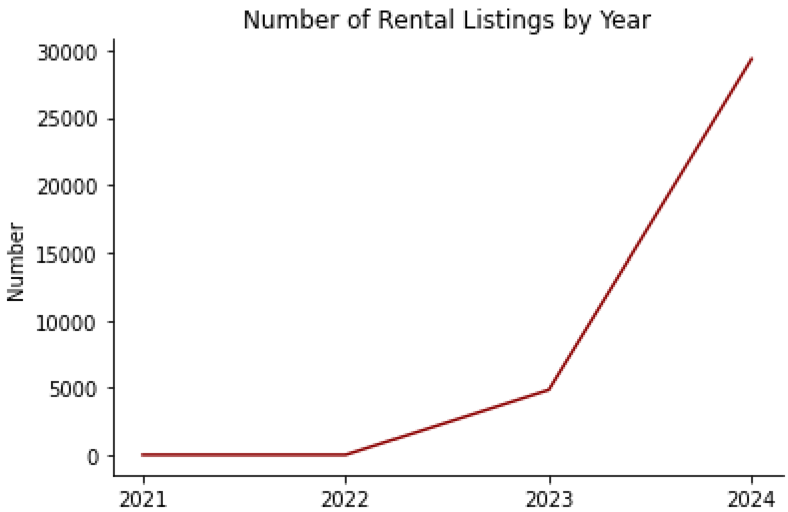
```
# Rentals by year
years2 = dubai.groupby('Year',as_index = False)['Rent'].count()
years2.columns = ['Year','Number']
years2
```

Out[535…

	Year	Number
0	2021	15
1	2022	15
2	2023	4766
3	2024	29099

In [536…

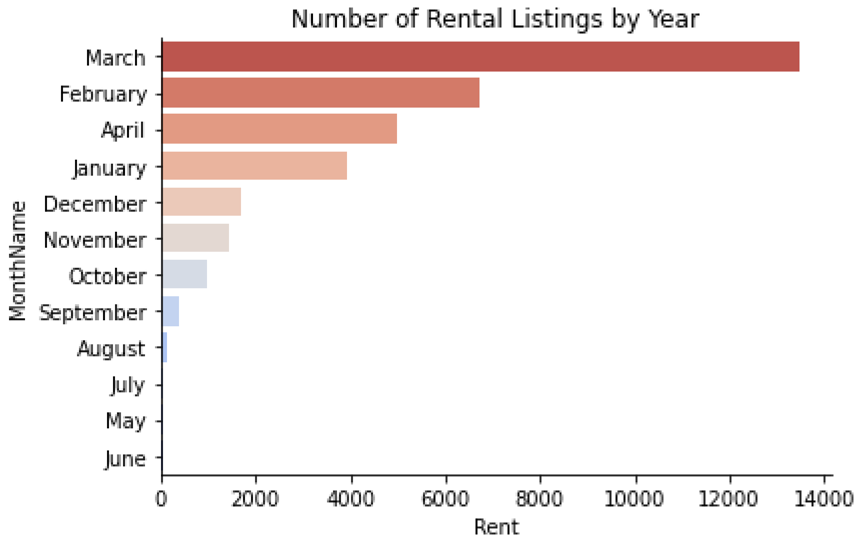
```
# Create a line plot
xvals = ['2021','2022','2023','2024']
ax = sns.lineplot(data=years, x=xvals, y="Number", color = 'darkred')
plt.title("Number of Rental Listings by Year")
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



The data covers just over 3 years from mid March 2021 to just into April 2024. We can see the number of rentals listed has increased significantly between 2023 and 2024 even though only 3 months of the latest year are included.

In [537…

```
# Number of rentals by Month
years3 = dubai.groupby('MonthName',as_index = False)['Rent'].count().sort_values(by

# Create a line plot
ax = sns.barplot(data=years3, y='MonthName', x="Rent",palette = 'coolwarm_r')
plt.title("Number of Rental Listings by Year")
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



More properties are listed in March and February and very few in the summer period of May through to August

3.3 Average Rents over Time

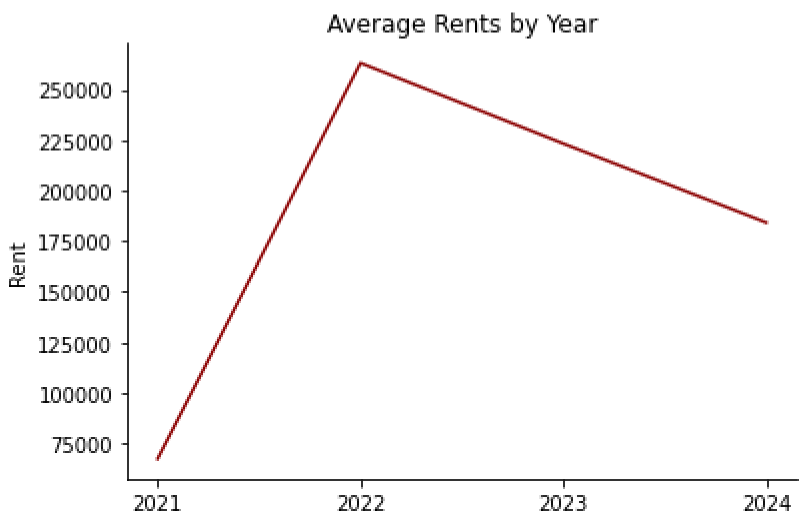
In [538…

```
print(dubai['Rent'].mean())
print(dubai['Rent'].median())
```

```
189587.1648030683
142000.0
```

In [539…

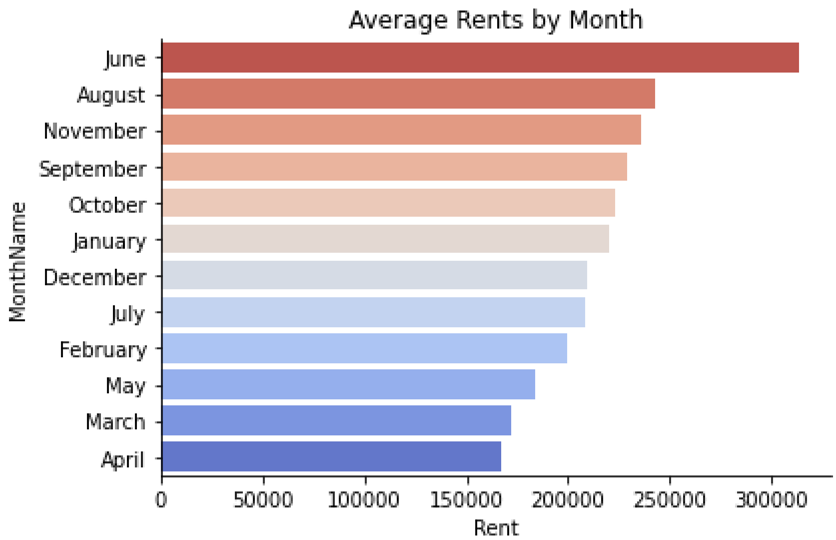
```
# Rents by year
years2 = dubai.groupby('Year',as_index = False)['Rent'].mean()
years2.columns = ['Year','Rent']

# Create a line plot
xvals = ['2021','2022','2023','2024']
ax = sns.lineplot(data=years2, x=xvals, y="Rent", color = 'darkred')
plt.title("Average Rents by Year")
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



Average rents peaked in 2022 where number of listings is low as we might expect. Since then it has declined as listings increase. However, the rates of change (decrease in average rents and increase in properties listed) differ, with average rents declining less rapidly than the number of listings. This might suggest continuing strong demand for rental properties not matched by supply. We can look at the number of days listings over time later which might shed some more light on this.

In [540…

```
# Average rents by Month
years4 = dubai.groupby('MonthName',as_index = False)['Rent'].mean().sort_values(by =

# Create a line plot
ax = sns.barplot(data=years4, y='MonthName', x="Rent",palette = 'coolwarm_r')
plt.title("Average Rents by Month")
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



Average rental prices are high in August and June where the number of rentals being listed are low and demand is strong and low in March and April where there are more listings. The exception is May where the number of listings is low and the rental values are also lowest reflecting lower demand for properties and lower supply on the market.

3.4 Age of Listing

In [541…

```
# Basic stats for age of listing
df2['Age_of_listing_in_days'].describe()
```

Out[541…

```
count    34229.000000
mean        63.917789
std         55.183479
min         12.000000
25%         27.000000
50%         46.000000
75%         81.000000
max       1131.000000
Name: Age_of_listing_in_days, dtype: float64
```

In [542…

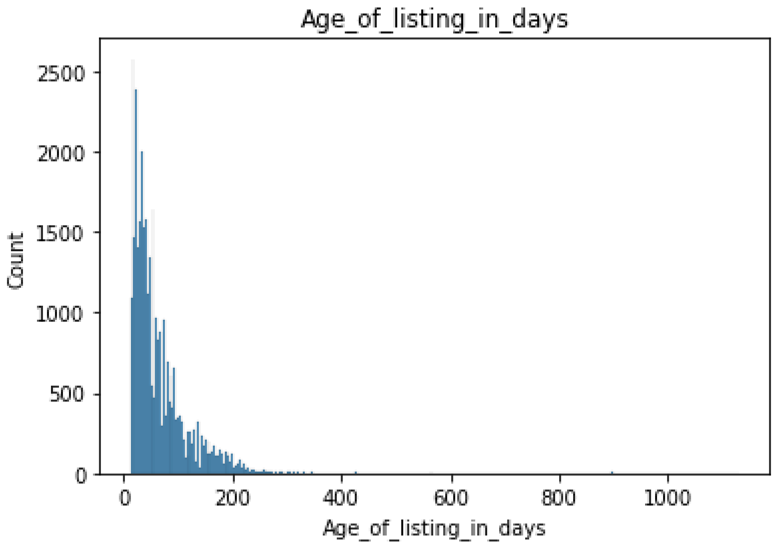
```
# Skew
df2['Age_of_listing_in_days'].skew()
```

Out[542…

```
3.7106414442965345
```

In [543…

```
# Plot the distribution of area in square feet
sns.histplot(data = df2['Age_of_listing_in_days'])
plt.title('Age_of_listing_in_days');
```



In [544…

```
# Boxplot of the data
df2['Age_of_listing_in_days'].plot(kind = 'box')
plt.title('Age of Listing in days');
```

The longest listing is for 1131 days, the shortest for just 12. The median listing time is 46 days and mean of 64 days which is a couple of months. Again the data is right skewed with the median values below the mean.

In [545…

```
# Listings over 1000 days
df2[df2['Age_of_listing_in_days']>1000]
```

Out[545…

	Address	Rent	Beds	Baths	Type	Area_in_sqft	Rent_per_sqft	Rent_category	Fur
32580	CBD-F05, Central Business District, Internatio...	53000	2	2	Apartment	915	57.923497	Low	Unfu
32569	B-12, China Cluster, International City, Dubai	53000	2	2	Apartment	915	57.923497	Low	Unfu
34391	A-09, China Cluster,	53000	2	2	Apartment	915	57.923497	Low	Unfu

	Address	Rent	Beds	Baths	Type	Area_in_sqft	Rent_per_sqft	Rent_category	Fur
	International City, Dubai								
32590	P-01, France Cluster, International City, Dubai	31500	0	2	Apartment	484	65.082645	Low	Unfu

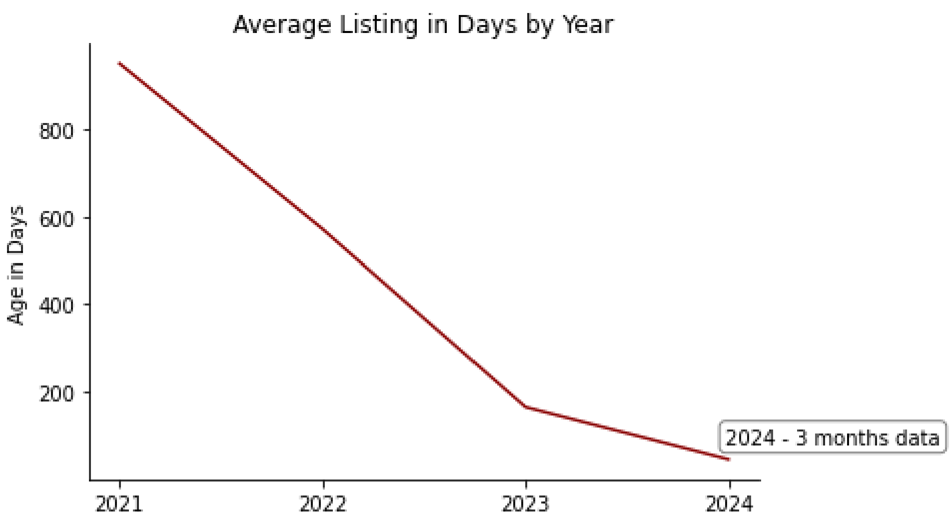
These all seem to be low rent apartments in one location at an address of the China Cluster or France Cluster. There is clearly something about these rentals being in this location that is important so it wouldnt be good idea to just remove them.

In [546...

```
# Change over time?
years5 = dubai.groupby('Year',as_index = False)['Age_of_listing_in_days'].mean()
years5.columns = ['Year','Age in Days']
display(years5)

# Create a line plot
xvals = ['2021','2022','2023','2024']
ax = sns.lineplot(data=years5, x=xvals, y="Age in Days", color = 'darkred')
plt.title("Average Listing in Days by Year")
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False)
props = dict(boxstyle='round', facecolor='White', alpha=0.5)
ax.text(0.95, 0.12, '2024 - 3 months data', transform=ax.transAxes, fontsize=10,
        verticalalignment='top', bbox=props)
;
```

	Year	Age in Days
0	2021	948.933333
1	2022	571.800000
2	2023	165.659463
3	2024	46.344788

Out[546... ''



We can see that average listing times by year have fallen over the period from an average of 949 days in 2021 (9 month average) to 166 days in 2023. We need to consider the huge effect the lockdowns of Covid 19 had on this market and the movement of people when interpreting these figures. The rate of decline slows between 2023 and 2024 but again, and not as sharp as for the rise in number of listings perhaps suggesting that although many more properties are being listed, that demand is still very strong especially into 2024 where properties are on the market for around 6 weeks on average.

In [107…

```
# Age of listing by rental category
dubai.groupby('Rent_category')['Age_of_listing_in_days'].mean()
```

Out[107…

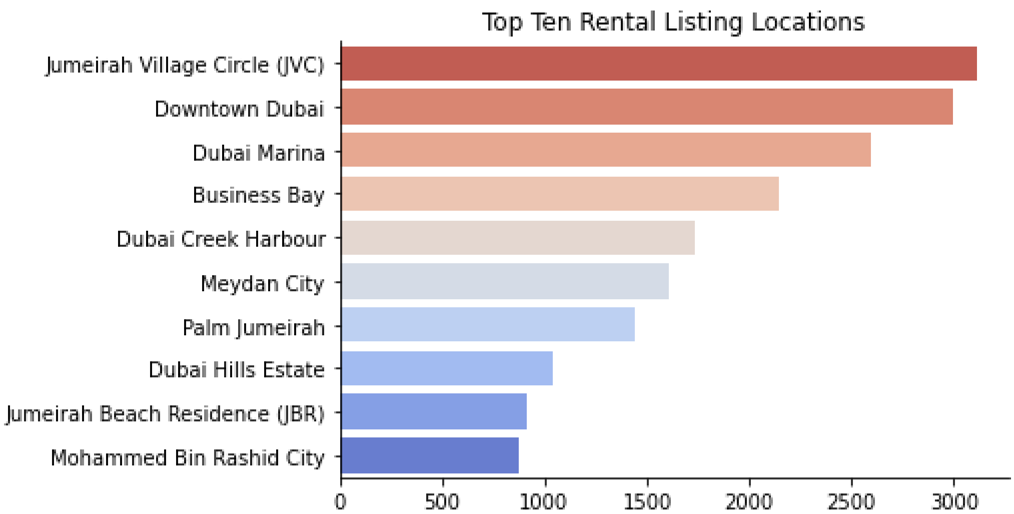
```
Rent_category
High      70.42
Low       55.58
Medium    57.68
Name: Age_of_listing_in_days, dtype: float64
```

As we might expect, higher rental properties have a longer listing age than low or medium rent

3.5 Location

In [547…

```
# Top ten locations being listed
top_10_locations = pd.DataFrame(dubai['Location'].value_counts().head(10).reset_inde

# Plot Top ten locations
ax = sns.barplot(data = top_10_locations, y = 'index', x = 'Location', palette = 'co
plt.title('Top Ten Rental Listing Locations')
plt.ylabel(None)
plt.xlabel(None)
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



Most rentals listed for the period are for Jumeirah Village, Downtown Dubai and the Dubai Marina.

In [131…

```
# Create a map of locations for rentals in Dubai

# Create a base map
m = folium.Map(location = [dubai['Latitude'].mean(), dubai['Longitude'].mean()],zoom

# Add points to the map
for idx, row in dubai.iterrows():
```

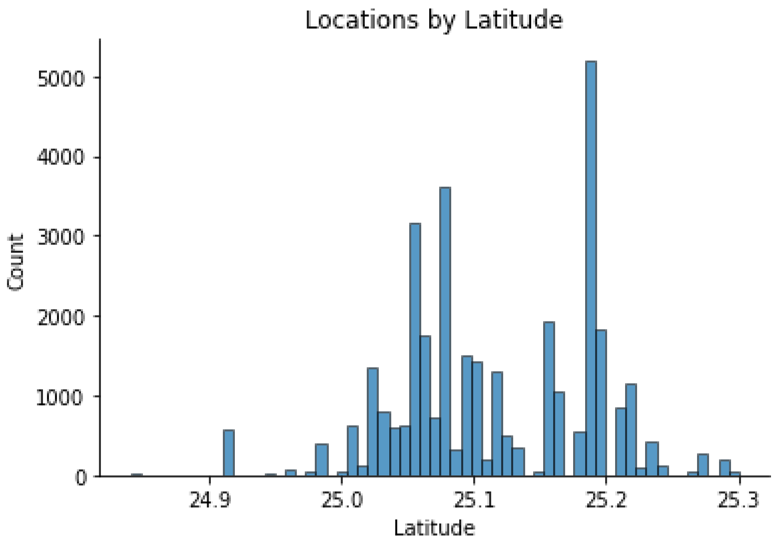
```
    folium.CircleMarker(location = (row['Latitude'], row['Longitude']),
                        radius = 2,
                        weight = 1,
                        fill = True,
                        fill_color = 'red',
                        fill_opacity = 0.7).add_to(m)
m
```

Out[131…

The map shows there are two clusters of properties which we can see if we plot the latitudes and longitudes showing the north south and east west clusters

In [549…

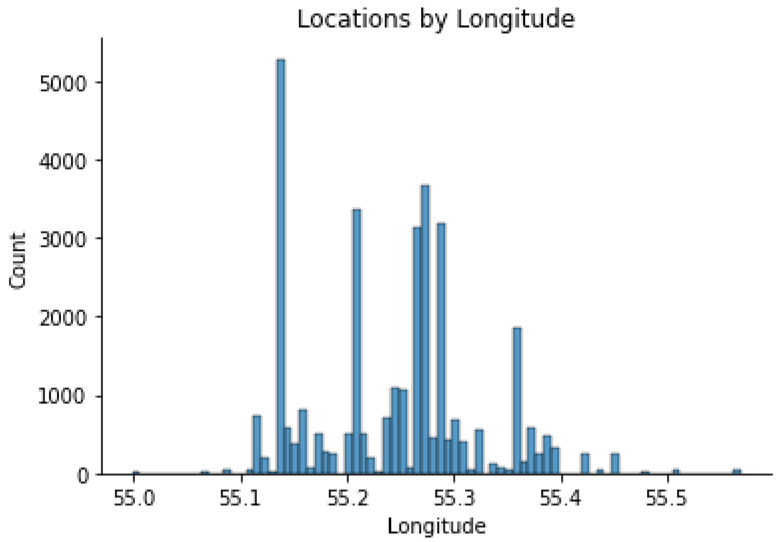
```
# We can plot the latitudes
ax=sns.histplot(dubai, x = 'Latitude')
plt.title('Locations by Latitude')
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



In [550…

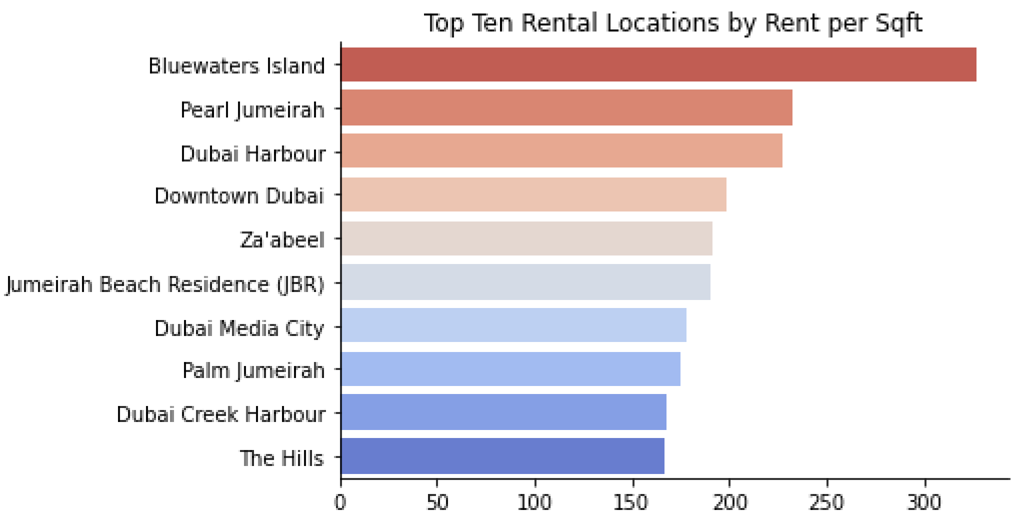
```
# We can plot the longitudes
ax=sns.histplot(dubai, x = 'Longitude')
plt.title('Locations by Longitude')
```

```
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



In [551…

```
# What are the top ten locations by rent per square foot?
top_10_locations_rent  = dubai.groupby('Location',as_index = False)['Rent_per_sqft']

# Plot Top ten locations
ax = sns.barplot(data = top_10_locations_rent, y = 'Location', x = 'Rent_per_sqft',
plt.title('Top Ten Rental Locations by Rent per Sqft')
plt.ylabel(None)
plt.xlabel(None)
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



In [552…

```
dubai[dubai['Location']=='Bluewaters Island']['Rent_per_sqft'].mean()
```

Out[552… 327.33516697397675

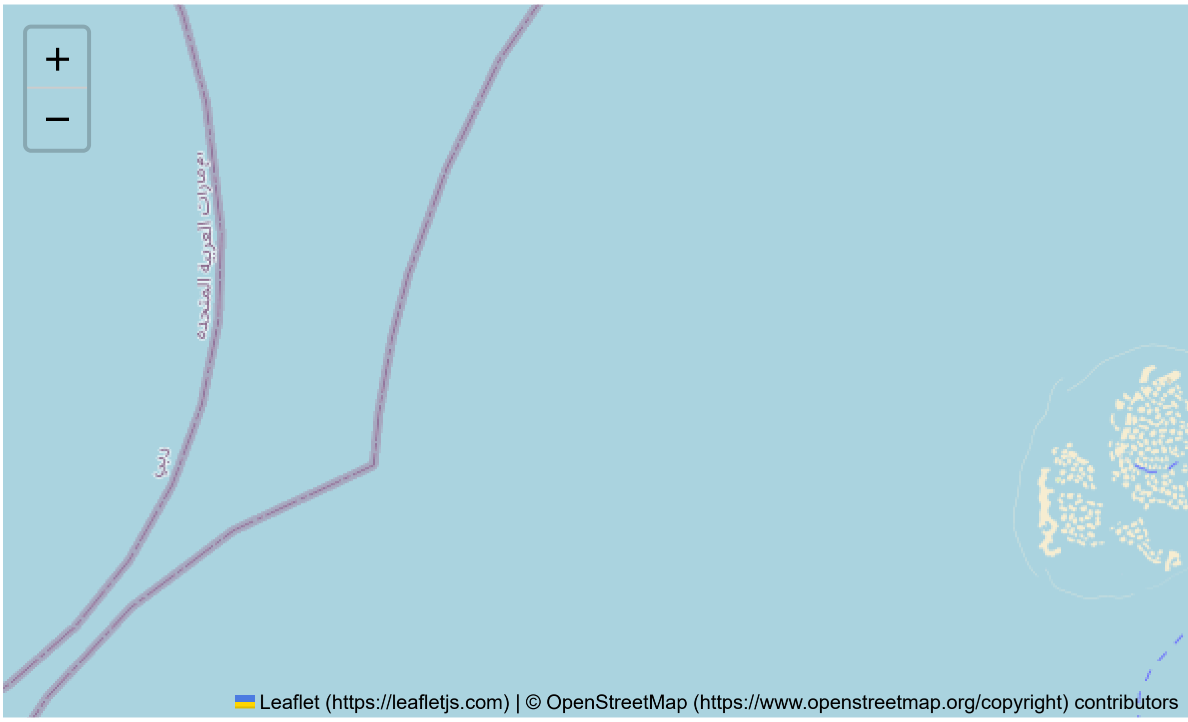
We can see that Bluewaters Island has the highest average rent per square ft of 329 AED per square ft.

In [997…

```
# Lets add a map that shows locations but categorised by the rent category or high,

# Create a base map
m = folium.Map(location = [dubai['Latitude'].mean(), dubai['Longitude'].mean()],zoom
```

```python
# Define categories colour mapping
color_map = {'High':'red', 'Medium':'orange', 'Low':'green'}

# Add points to the map
for idx, row in dubai.iterrows():
    folium.CircleMarker(location = (row['Latitude'], row['Longitude']),
                        radius = 4,
                        color = 'black',
                        weight = 1,
                        color_map = color_map[row['Rent_category']],
                        fill = True,
                        fill_color = color_map[row['Rent_category']],
                        fill_opacity = 0.7).add_to(m)
m
```

Out[997…



The map shows that most of the rentals are high priced and that they are clustered along the water and to the south of the central part of the city. Lower priced rentals are a bit further out, although there are some exceptions. Medium priced rentals are scattered about the city with clusters in the centre north and centre south.

In [554…

```python
dubai.info()
```

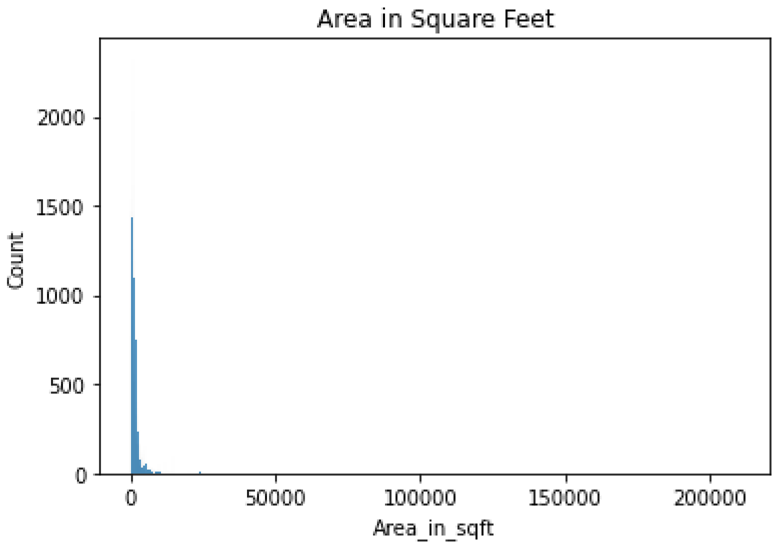
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 33895 entries, 29068 to 63317
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Address                33895 non-null  object
 1   Rent                   33895 non-null  int64
 2   Beds                   33895 non-null  int64
 3   Baths                  33895 non-null  int64
 4   Type                   33895 non-null  object
 5   Area_in_sqft           33895 non-null  int64
 6   Rent_per_sqft          33895 non-null  float64
 7   Rent_category          33895 non-null  object
 8   Furnishing             33895 non-null  object
 9   Posted_date            33895 non-null  datetime64[ns]
 10  Age_of_listing_in_days 33895 non-null  int64
 11  Location               33895 non-null  object
 12  Latitude               33895 non-null  float64
 13  Longitude              33895 non-null  float64
```

```
 14   Year                        33895 non-null   int64
 15   Month                       33895 non-null   int64
 16   MonthName                   33895 non-null   object
 17   RentZScore                  33895 non-null   float64
dtypes: datetime64[ns](1), float64(4), int64(7), object(6)
memory usage: 5.9+ MB
```

3.6 Area

In [555…

```python
# Plot the distribution of area in square feet
sns.histplot(data = dubai['Area_in_sqft'])
plt.title('Area in Square Feet');
```

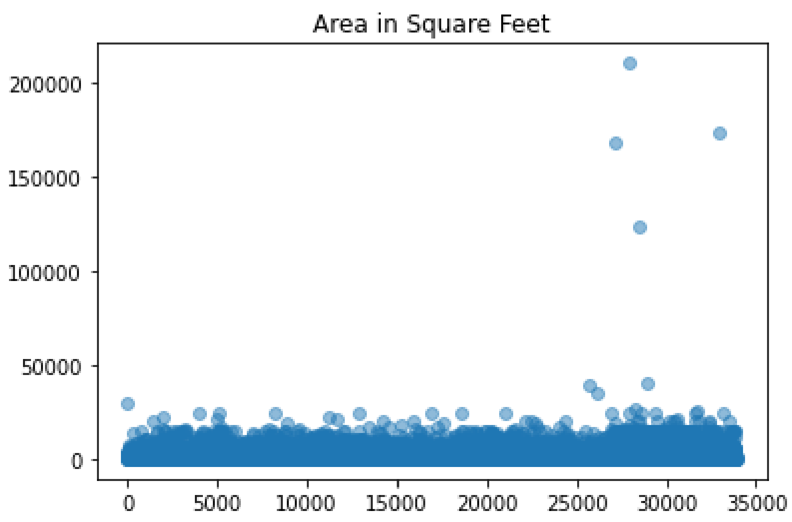


We have a lot of right skew in the data, indicating some very large properties. We can have a look what these are

Outlier Investigation

In [557…

```python
# Scatter plot of the property areas
ind = list(range(0,33895))
vals = list(dubai['Area_in_sqft'])

fig, ax = plt.subplots(figsize = (6,4))
plt.scatter(ind,vals, alpha = 0.5)

plt.title('Area in Square Feet');
```



Several properties are above 100000 square feet

In [558…

```
# Split into those over and those under or equal to 100000 square feet
over = dubai[dubai['Area_in_sqft']>=100000].sort_values(by = 'Area_in_sqft',ascendin
under = dubai[dubai['Area_in_sqft']<100000].sort_values(by = 'Area_in_sqft',ascendin
```

In [559…

```
over
```

Out[559…

	Address	Rent	Beds	Baths	Type	Area_in_sqft	Rent_per_sqft	Rent_category	Fu
57236	Marina Residences 1, Marina Residences, Palm J...	900000	4	2	Penthouse	210254	4.280537	High	Un
62394	One Za'abeel The Residences, One Za'abeel, Za'...	1200000	1	1	Residential Plot	173565	6.913836	High	Un
56445	Jumeirah Village Triangle (JVT), Dubai	299999	6	2	Villa	168046	1.785220	High	
57810	Jumeirah 3, Jumeirah, Dubai	420000	5	2	Villa Compound	123696	3.395421	High	Un

In [560…

```
# Create list of these locations where property is over 100000
locations = over['Location'].tolist()
print(locations)

# Get the mean areas for these locations for the rentals excluding these properties
display(under[under['Location'].isin(locations)].groupby('Location')['Area_in_sqft']

# Get the median areas for these locations for the rentals excluding these propertie
display(under[under['Location'].isin(locations)].groupby('Location')['Area_in_sqft']
```

```
['Palm Jumeirah', "Za'abeel", 'Jumeirah Village Triangle (JVT)', 'Jumeirah']
Location
Jumeirah                           3735.671388
Jumeirah Village Triangle (JVT)    2587.018349
Palm Jumeirah                      1960.056367
Za'abeel                           1319.143820
Name: Area_in_sqft, dtype: float64
Location
Jumeirah                           2907
Jumeirah Village Triangle (JVT)    1850
Palm Jumeirah                      1582
Za'abeel                           1200
Name: Area_in_sqft, dtype: int64
```

Comparing these large properties to the mean and median values for every other property in those respective neighbourhoods, it seems that they are very much outliers in those areas so we will drop them from the data.

In [737…

```
# Create a copy of the revised data
df2 = under.copy()
df2.shape
```

Out[737…

```
(33891, 18)
```

In [107…

```
# Group the areas by rental category
df2.groupby('Rent_category')['Area_in_sqft'].mean()
```

Out[107…

```
Rent_category
High      2579.45
Low        560.65
Medium     978.82
Name: Area_in_sqft, dtype: float64
```

Analysing by rental category, the mean areas for high rentals are two and a half times that of medium rentals and four and a half times that of the low rentals

3.7 Relationships

We can have a look at a few of the many relationships of interest between the different features and rents

Listing Age and Rents

In [738…

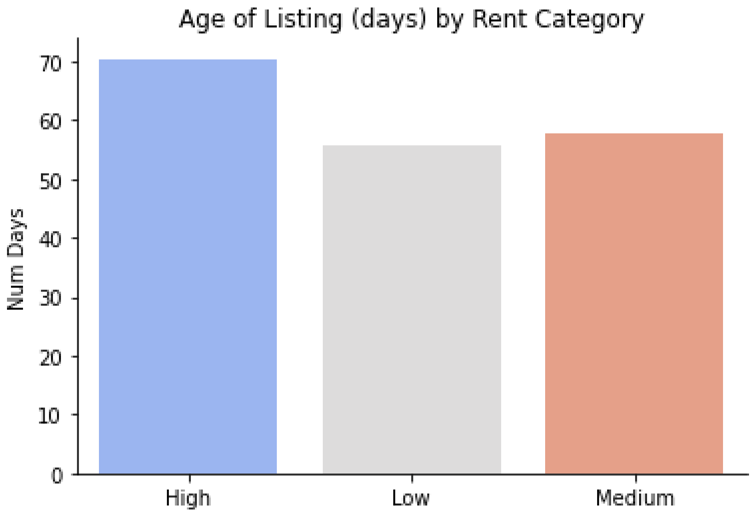
```
# What category of rentals is on the market for the longest and which the shortest?

rent_age = df2.groupby('Rent_category',as_index = False)['Age_of_listing_in_days'].m
rent_age
```

Out[738…

	Rent_category	Age_of_listing_in_days
0	High	70.428125
1	Low	55.581395
2	Medium	57.676539

In [739…

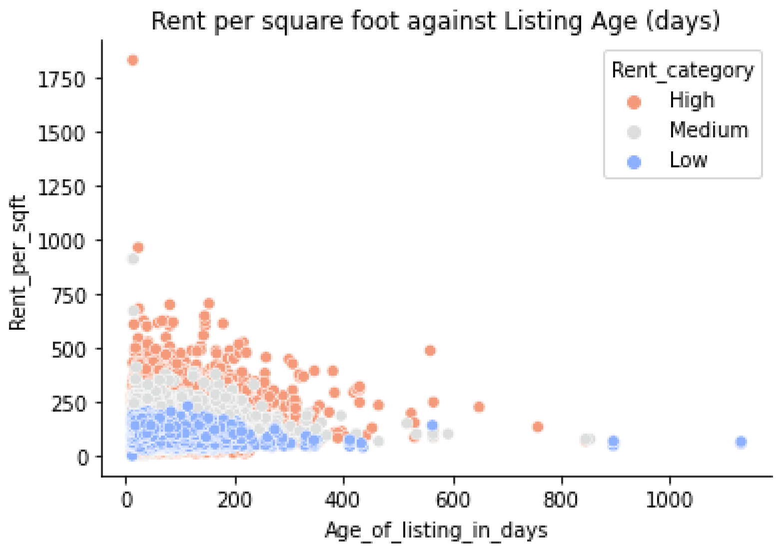
```
# Plot the data
ax = sns.barplot(data = rent_age, x = 'Rent_category', y = 'Age_of_listing_in_days',
plt.title("Age of Listing (days) by Rent Category")
plt.xlabel(None)
plt.ylabel('Num Days')
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



Listing days gives us a sense of demand in the market. High rental properties are on the market an average of 70 days compared to medium priced rentals at 58 and low priced at 58. Therefore there is very little difference between the low priced and medium priced categories in terms of the number of days they are on the market. Our earlier chart showed about double the number of listings for medium priced rentals as low priced and yet they remain on the market for a similar time, suggesting the demand might be primarily for the medium priced rental properties.

In [740…

```
# Listing age against rent per square foot scatterplot
ax = sns.scatterplot(data = df2, x = 'Age_of_listing_in_days', y = 'Rent_per_sqft',
plt.title('Rent per square foot against Listing Age (days)')
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False);
```



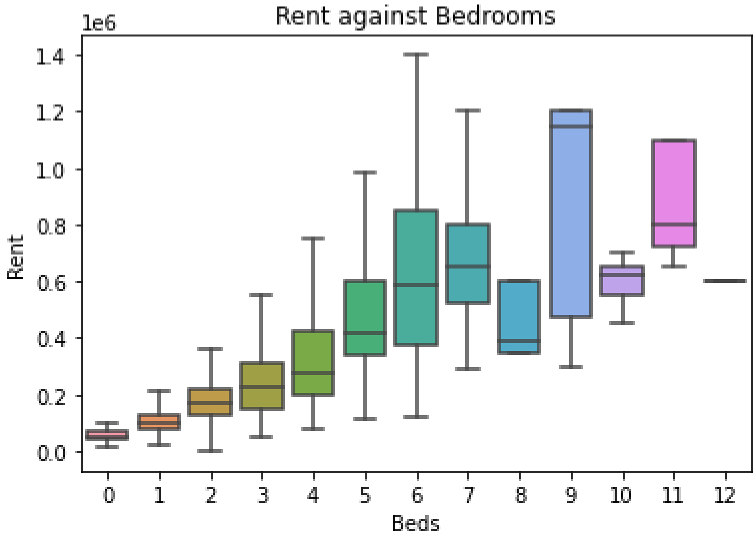
We can see that there is some weak positive correlation here. There are outliers in the data with low rental property on the market for more than 1000 days and higher priced appearing to be snapped up immediatelu. This should probably be the focus of additional work to identify what issues if any relate to their being empty for so long as clearly there are issues other than price affecting the rentability for some of the lower priced ones.

Other Relationships to Rent

The feature we would want to predict would be the rent. We want to identify those other features that have any kind of correlation with rent. For example does the rent value have any correlation with locational data, or with the number of bedrooms?

In [741…

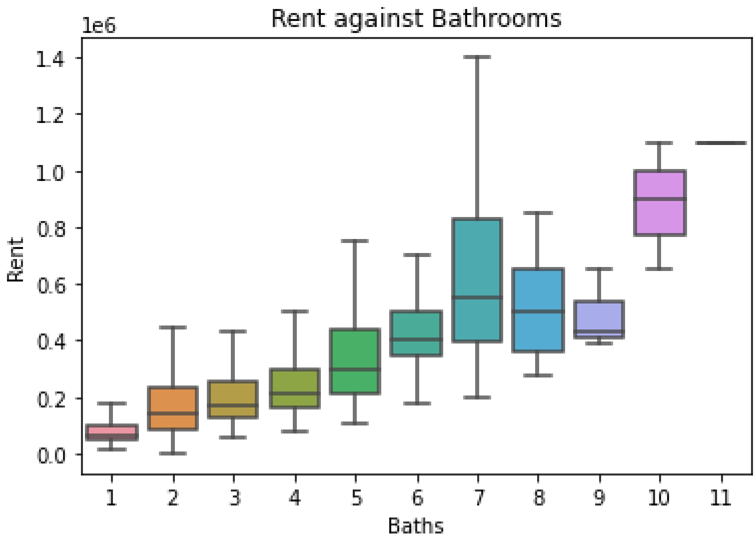
```
# Bedrooms
sns.boxplot(data = df2, x = 'Beds', y = 'Rent', showfliers = False)
plt.title("Rent against Bedrooms");
```



There appears to be a postivie correlation between number of beds and average rental values up to about 7, whereby more bedrooms attracts higher rents. Of interest is the wider range of rental values for properties with large numbers of bedrooms (above 5) from higher rental top-end through to lower rental.

In [742…

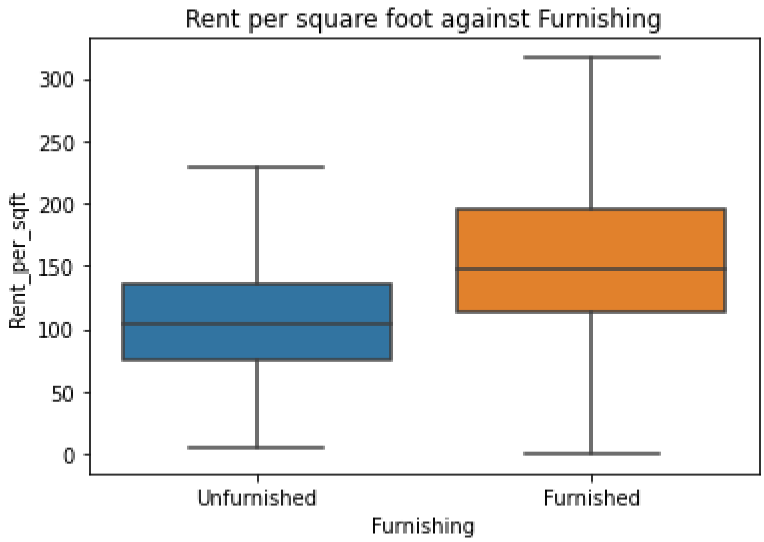
```
# Bathrooms
sns.boxplot(data = df2, x = 'Baths', y = 'Rent', showfliers = False)
plt.title("Rent against Bathrooms");
```



More bathrooms appear to attract higher rents, up to a point. At 7 bedrooms there is again a wider range of rental values

In [743…

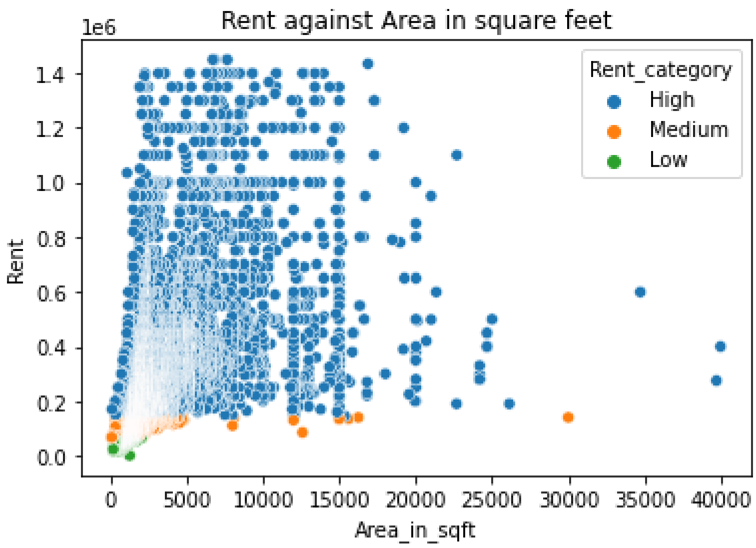
```
# Compare furnishing with rental values excluding outliers
sns.boxplot(data = df2, x = 'Furnishing', y = 'Rent_per_sqft', showfliers = False)
plt.title("Rent per square foot against Furnishing");
```



Furnished properties have a higher rental value than unfurnished

In [744…

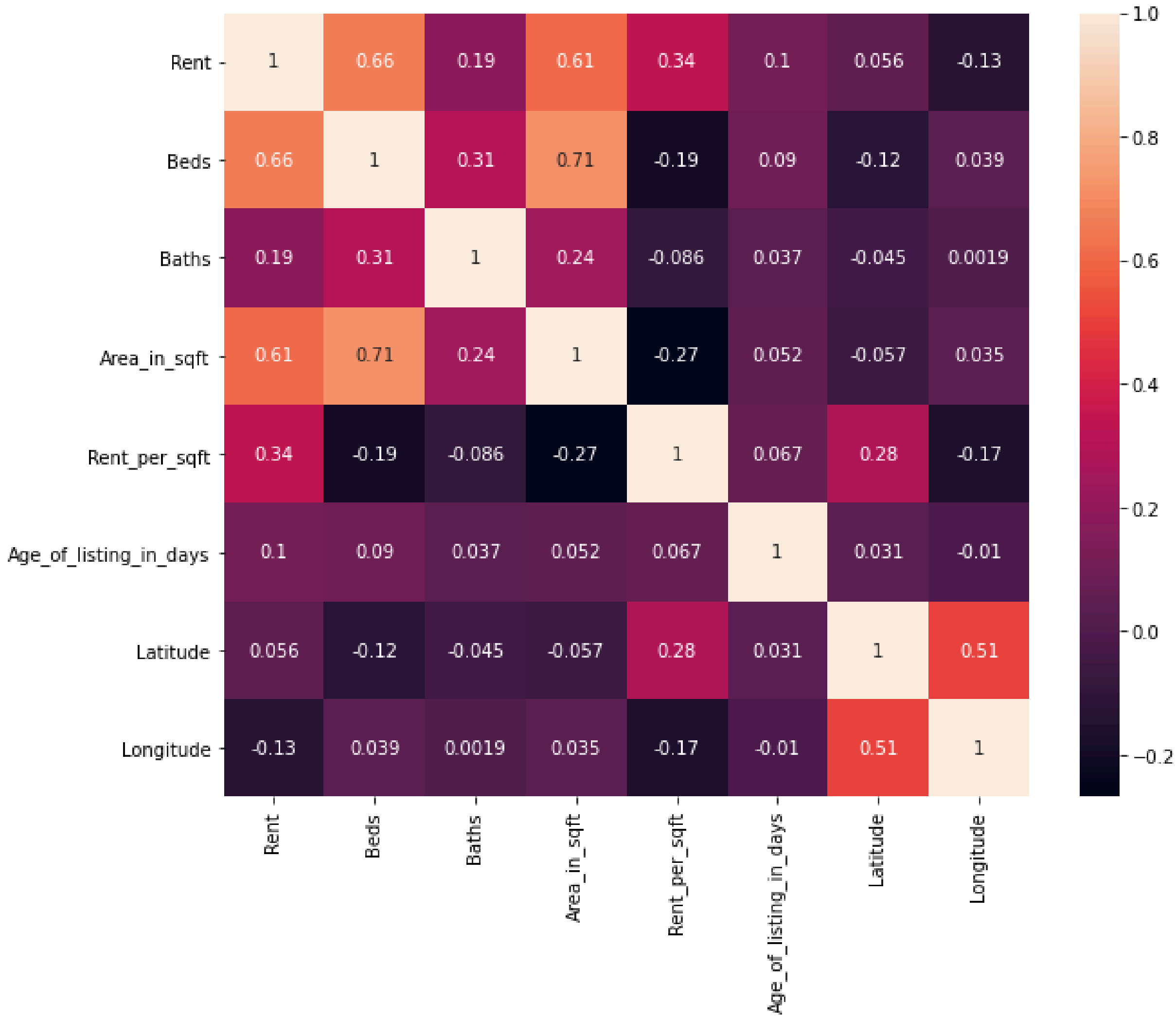
```
# Area against rent
sns.scatterplot(data = df2, x = 'Area_in_sqft', y = 'Rent', hue = 'Rent_category')
plt.title('Rent against Area in square feet');
```



As we might expect there seems to be a positive correlation between the area and rents

In [745…

```
# Correlations in a heatmap
cols = ['Rent','Beds','Baths','Area_in_sqft','Rent_per_sqft','Age_of_listing_in_days
fig, ax = plt.subplots(figsize = (10,8))
ax = sns.heatmap(df2[cols].corr(), annot = True)
```



Rent is strongly positively correlated with area and number of bedrooms, negatively correlated with location (longitude), so that moving inland has lower rents and weakly positively correlated with the number of bathrooms and the age of listing. Location north or south indicated by latitude has a very small positive correlation.

From the analysis, we might look at including most of these features but exclude the geo-cordinates

4.0 Prediction

4.1 Data Preparation

In [883…

```
df3 = df2.copy()
```

Trying to avoid a 'kitchen sink' regression, we will pick some of the features to include. The address has many different values which would complicate the analysis and the information is available in other features. The rent per square foot is contains information already for the target of rent we are trying to predict. The posted date is effectively covered by the year and month and the location will be covered by the latitude and longitude. The monthname copies the month feature and the rentzscore is added earlier and is irrelevant.

In [884…

```
# Drop columns we don't need
df3.drop(columns = ['Address','Rent_per_sqft','Posted_date','Location','MonthName','
```

In [128…

```
# Reset the index so we can manipulate the dataframe
df3.reset_index(inplace = True, drop = True)

# Split dataframe into X and y (features and target)
X = df3.drop(columns = ['Rent'], axis = 1)
y = df3[['Rent']]
```

We have both ordinal and nominal categorical values as well as numerical. For example the rent category is ordered from low to high, however the furnishing and type are NOT ordered so we would not want to apply a sequential ranking to these so would choose a one-hot encoding.

In [128…

```
# Replace categorical ordinal values in rent category with numerical
X['Rent_category'].replace({'High':3,'Medium':2,'Low':1}, inplace = True)

# One hot encoding for the type, furnishing and month (we set drop first = true to a
X = pd.get_dummies(X, prefix=None, columns=['Type', 'Furnishing','Year','Month'], dr
```

We will standardise the numerical features but as we will fit and transform on the training data and then should only transform on the test data, we will need to split the dataframe into training and test sets first

In [128…

```
# Set up the training and test sets
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(27112, 30) (6779, 30) (27112, 1) (6779, 1)
```

In [129…

```
# Split off the numerical features and standardise these
nums = ['Beds','Baths','Area_in_sqft','Age_of_listing_in_days','Latitude','Longitude
X_train_num = X_train[nums]
X_test_num = X_test[nums]

# Standardise the numeric features (Beds, Bath, Area, Age of Listing), fit transform
scale = StandardScaler()
X_train_num = pd.DataFrame(scale.fit_transform(X_train_num))
X_train_num.columns = nums

# Transform the test set
X_test_num = pd.DataFrame(scale.transform(X_test_num))
```

In [131…

```
# Check the scaled values have mean zero and standard deviation of one
X_train_num.describe()
```

Out[131…

	Beds	Baths	Area_in_sqft	Age_of_listing_in_days	Latitude	Longitude
count	27112.00	27112.00	27112.00	27112.00	27112.00	27112.00
mean	0.00	-0.00	0.00	0.00	0.00	0.00
std	1.00	1.00	1.00	1.00	1.00	1.00
min	-1.43	-1.97	-0.81	-0.94	-3.76	-3.08

	Beds	Baths	Area_in_sqft	Age_of_listing_in_days	Latitude	Longitude
25%	-0.70	-0.15	-0.48	-0.67	-0.77	-0.90
50%	0.04	-0.15	-0.28	-0.32	-0.23	0.25
75%	0.78	-0.15	0.08	0.31	0.97	0.56
max	7.40	16.29	18.71	19.36	2.53	4.07

In [889...

```
# Reset the index on each dataframe
X_train.reset_index(drop=True, inplace=True)
X_train_num.reset_index(drop=True, inplace=True)

X_test.reset_index(drop=True, inplace=True)
X_test_num.reset_index(drop=True, inplace=True)

# Drop the numerical columns in the original dataframe
X_train.drop(columns = nums, axis = 1, inplace = True)
X_test.drop(columns = nums, axis = 1, inplace = True)

# Concat the scaled data back to the original dataframe
X_train = pd.concat([X_train, X_train_num], axis = 1)
X_test = pd.concat([X_test, X_test_num], axis = 1)
```

We now have our training and test sets ready for modelling

In [100...

```
X_train.columns
```

Out[100...

```
Index(['Rent_category', 'Type_Hotel Apartment', 'Type_Penthouse',
       'Type_Residential Building', 'Type_Residential Floor',
       'Type_Residential Plot', 'Type_Townhouse', 'Type_Villa',
       'Type_Villa Compound', 'Furnishing_Unfurnished', 'Year_2022',
       'Year_2023', 'Year_2024', 'Month_2', 'Month_3', 'Month_4', 'Month_5',
       'Month_6', 'Month_7', 'Month_8', 'Month_9', 'Month_10', 'Month_11',
       'Month_12', 'Beds', 'Baths', 'Area_in_sqft', 'Age_of_listing_in_days',
       'Latitude', 'Longitude'],
      dtype='object')
```

4.2 Multiple Linear Regression

In [890...

```
# Fit the linear model to the training data and get the training score
lr = LinearRegression()
mod1 = lr.fit(X_train, y_train)
mod1.score(X_train, y_train)
```

Out[890...

```
0.5874595003889362
```

In [891...

```
# Predict on the test set
y_pred = mod1.predict(X_test)
r2_score(y_test, y_pred)
```

Out[891...

```
0.6015289766052676
```

In [100...

```
mod1coeff = pd.DataFrame(mod1.coef_).T
mod1coeff.index = X_train.columns
mod1coeff.columns = ['Coefficient']
mod1coeff
```

Out[100…

	Coefficient
Rent_category	32484.01
Type_Hotel Apartment	17865.10
Type_Penthouse	163714.66
Type_Residential Building	132673.77
Type_Residential Floor	834836.06
Type_Residential Plot	-234762.34
Type_Townhouse	-47623.42
Type_Villa	-7744.18
Type_Villa Compound	-176903.36
Furnishing_Unfurnished	-36753.71
Year_2022	81594.20
Year_2023	80147.39
Year_2024	99747.03
Month_2	-392.66
Month_3	-6031.25
Month_4	-5407.68
Month_5	-22614.23
Month_6	51661.08
Month_7	-7641.79
Month_8	7259.82
Month_9	-4552.35
Month_10	4511.24
Month_11	12619.32
Month_12	4569.17
Beds	76763.38
Baths	-2302.57
Area_in_sqft	44874.64
Age_of_listing_in_days	6633.59
Latitude	31459.44
Longitude	-33590.13

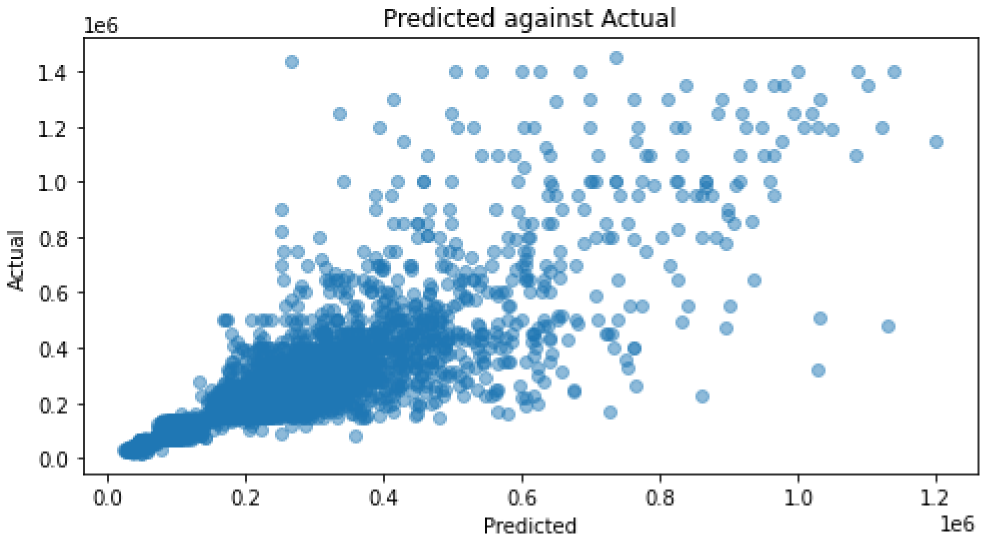
From this first model, 61% of the variation in rental values can be explained by our model. There are clearly other factors at play in the market affecting the price. We should look at the residuals as we have made some assumptions using a linear regression on this data

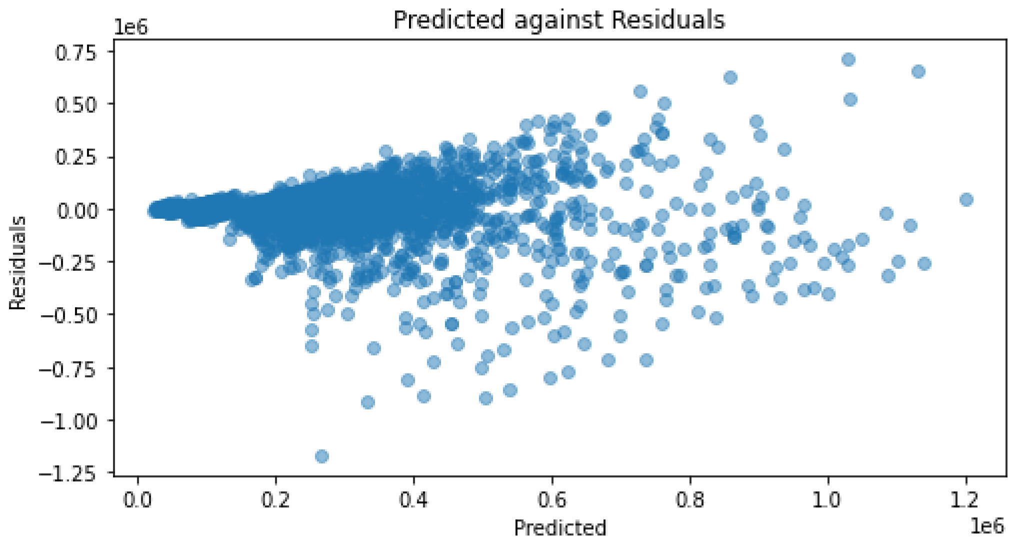
In [934…

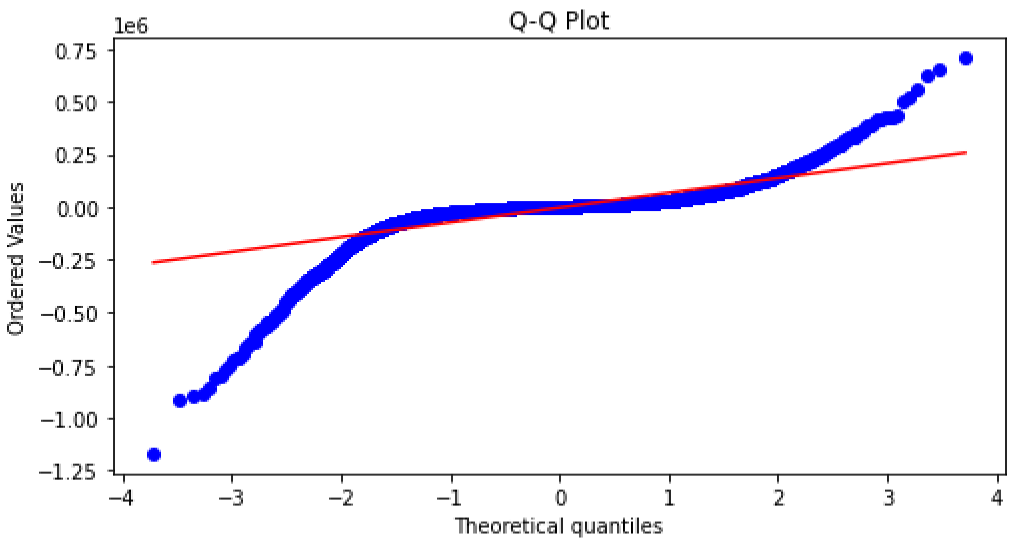
```
# Concat the actual and predicted and calculate residuals
results = pd.concat([y_test, y_pred],axis = 1)
results.columns = ['Actual','Predicted']
results['Res'] = results['Predicted'] - results['Actual']
```

In [948…

```python
# Plot predicted against actual
fig, ax = plt.subplots(figsize = (8,4))
plt.scatter(results['Predicted'], results['Actual'], alpha = 0.5)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Predicted against Actual');

# Predicted against residual
fig, ax = plt.subplots(figsize = (8,4))
plt.scatter(results['Predicted'], results['Res'], alpha = 0.5)
plt.xlabel('Predicted')
plt.ylabel('Residuals')
plt.title('Predicted against Residuals');

# QQ Plot
plt.figure(figsize=(8, 4))
stats.probplot(results['Res'], dist="norm", plot=plt)
plt.title('Q-Q Plot');
```







We can see from these plots that there is increasing variance in the residuals indicating heteroscedasticity with the residuals getting larger across the plot. This reflects the fact that rents can be high or low depending on the other features so we cannot assume they are high or low. The QQ plot shows the blue line varying from the normal red line at both ends. This goes against one of the assumptions of the linear model, so we can either try to adjust this model with maybe a log transform or try different models that do not require these assumptions. We will look at other models.

4.3 Other models

We will look at a couple of other models accepting the default parameters. We are just using a simple train and test split here rather than a cross validation method, as this is just a simple initial analysis

Nearest Neighbours

In [898…

```
# Fit model and get training score
knn = neighbors.KNeighborsRegressor(n_neighbors = 5)
mod2 = knn.fit(X_train, y_train)
mod2.score(X_train, y_train)
```

Out[898…

```
0.8286433707539533
```

In [899…

```
# Predict on the test set
y_pred = mod2.predict(X_test)
r2_score(y_test, y_pred)
```

Out[899…

```
0.7517239157864216
```

This model seems to have performed better than our linear regression model

Random Forest

In [952…

```
# Fit model and get training score
clf = RandomForestClassifier()
mod3 = clf.fit(X_train, y_train)
mod3.score(X_train, y_train)
```

Out[952... 0.9806727648273827

In [953...
```
# Predict on the test set
y_pred = mod3.predict(X_test)
r2_score(y_test, y_pred)
```

Out[953... 0.7349611182082432

The higher training score compared to test score suggests this model might be overfitting.

In [965...
```
# Reduce number of estimators - various values were explored here to see if we could
clf = RandomForestClassifier(n_estimators = 2)
mod3 = clf.fit(X_train, y_train)
print('Training Score:', mod3.score(X_train, y_train))

# Predict on the test set
y_pred = mod3.predict(X_test)
r2_score(y_test, y_pred)
```

```
Training Score: 0.664281498967247
```

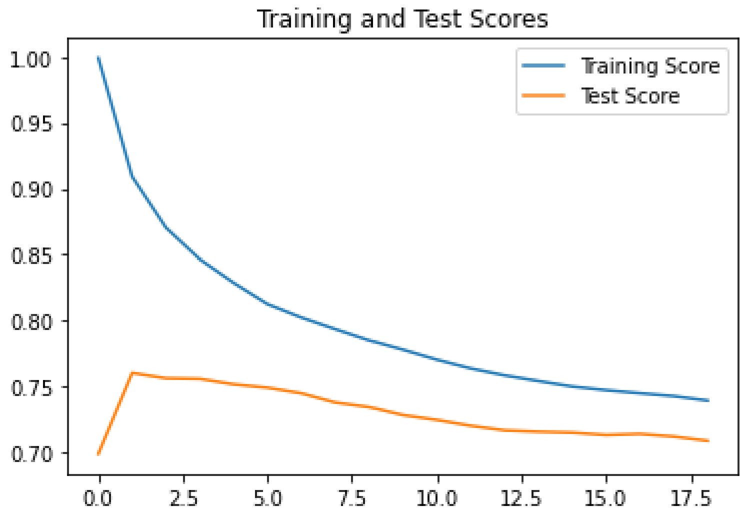
Out[965... 0.6621000484924996

We have reduced overfitting but the training and test scores are not as good as for the nearest neighbours algorithm. We can see if we can improve the results by trying different numbers of neighbours for that model.

Nearest Neighbors - parameter tuning

In [977...
```
# Fit model and get training score
n_neighbors = list(range(1,20,1))
trains = []
tests = []

for i in n_neighbors:
    knn = neighbors.KNeighborsRegressor(n_neighbors = i)
    mod2 = knn.fit(X_train, y_train)
    y_pred = mod2.predict(X_test)

    train_score = mod2.score(X_train, y_train)
    test_score = r2_score(y_test, y_pred)
    trains.append(train_score)
    tests.append(test_score)
```

In [984...
```
# Plot the training and test scores
plt.plot(trains)
plt.plot(tests)
plt.title('Training and Test Scores')
plt.legend(['Training Score','Test Score']);
```



In [993...

```
scores = pd.DataFrame(list(zip(trains, tests)))
scores['Diff'] = scores[0] - scores[1]
scores.sort_values(by = 'Diff',ascending = False)
```

Out[993...

	0	1	Diff
0	1.00	0.70	0.30
1	0.91	0.76	0.15
2	0.87	0.76	0.11
3	0.85	0.76	0.09
4	0.83	0.75	0.08
5	0.81	0.75	0.06
6	0.80	0.74	0.06
7	0.79	0.74	0.06
8	0.79	0.73	0.05
9	0.78	0.73	0.05
10	0.77	0.72	0.05
11	0.76	0.72	0.04
12	0.76	0.72	0.04
13	0.75	0.72	0.04
14	0.75	0.72	0.03
15	0.75	0.71	0.03
16	0.74	0.71	0.03
17	0.74	0.71	0.03
18	0.74	0.71	0.03

Looking a the training and test scores there is no further improvement beyond 14 neighbours

In [994...

```
# Fit model and get training score
knn = neighbors.KNeighborsRegressor(n_neighbors = 14)
mod2 = knn.fit(X_train, y_train)
```

```python
print(mod2.score(X_train, y_train))

# Predict on the test set
y_pred = mod2.predict(X_test)
print(r2_score(y_test, y_pred))
```

```
0.7542007165680242
0.715708109253427
```

There is plenty more we could do with this project but for now we can conclude that with the K Nearest Neighbours Algorithm that we can explain about 72% of the variation in rental prices from the model including the features we have chosen. This leaves another 28% explainable by other factors we have not considered or do not have access to. This might include all kinds of other things such as location close to transport routes, schools, amenities, local conditions, market conditions and government policies. This shows how complicated building a good model to predict rental prices is.

4.4 New prediction

We can use our model to make some predictions on some new data. I am looking for rental values for 1000 square foot apartments in the medium category and in the top five locations for medium priced properties. These properties should be furnished and need 1 bedroom and 2 bathrooms and have been listed for 30 days.

In [121…

```python
# Get the medium rental areas with the most number of properties listed
meds = df2[df2['Rent_category']=='Medium'].groupby('Location',as_index = False)['Add
```

In [122…

```python
# Get the mean latitude and longitude of each of these locations from the original d
lats = []
longs = []

for i in meds.Location:
    lat = df2[df2['Location']==i]['Latitude'].mean()
    long = df2[df2['Location']==i]['Longitude'].mean()
    lats.append(lat)
    longs.append(long)
```

In [127…

```python
# Create a list of the features we will need to copy
r1 = [2,1,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,1,2,1000,30]

# Create a list of 5 independent copies of this list, all the same
lists = [r1[:] for _ in range(5)]

# Add the various latitudes and longitudes we found above to the lists
a = [[*x, y] for x, y in zip(lists,lats)]
b = [[*x, y] for x, y in zip(a,longs)]
```

In [128…

```python
new_row = pd.DataFrame([b[0]], columns=X_train.columns)
new_row2 = pd.DataFrame([b[1]], columns=X_train.columns)
new_row3 = pd.DataFrame([b[2]], columns=X_train.columns)
new_row4 = pd.DataFrame([b[3]], columns=X_train.columns)
new_row5 = pd.DataFrame([b[4]], columns=X_train.columns)

new_data = pd.concat([new_row5, new_row4, new_row3, new_row2, new_row], ignore_index
new_data
```

Out[128...

	Rent_category	Type_Hotel Apartment	Type_Penthouse	Type_Residential Building	Type_Residential Floor	Type_Residential Plot
0	2	1	0	0	0	0
1	2	1	0	0	0	0
2	2	1	0	0	0	0
3	2	1	0	0	0	0
4	2	1	0	0	0	0

5 rows × 30 columns

In [129...

```
# Standardise the numerical features
new_data[['Beds', 'Baths','Area_in_sqft','Age_of_listing_in_days','Latitude','Longit
```

In [131...

```
new_data
```

Out[131...

	Rent_category	Type_Hotel Apartment	Type_Penthouse	Type_Residential Building	Type_Residential Floor	Type_Residential Plot
0	2	1	0	0	0	0
1	2	1	0	0	0	0
2	2	1	0	0	0	0
3	2	1	0	0	0	0
4	2	1	0	0	0	0

5 rows × 30 columns

In [131...

```
# Predict on the new data
y_pred_new = pd.DataFrame(mod2.predict(new_data))
```

In [131...

```
# Rentals for the new locations we picked out
y_pred_new.index = meds.Location

# Name the column and round the entries by setting to integer
y_pred_new.columns = ['Rent']
y_pred_new['Rent'] = y_pred_new['Rent'].astype(int)

# Format the column with commas
y_pred_new['Rent'] = y_pred_new['Rent'].apply(lambda x: f"{x:,}")
y_pred_new
```

Out[131...

	Rent
Location	
Jumeirah Village Circle (JVC)	88,285

	Rent
Location	
Business Bay	119,428
Dubai Marina	110,357
Downtown Dubai	102,035
Meydan City	78,896