

Start coding or [generate](#) with AI.

↵ New Section

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
import math

warnings.filterwarnings("ignore")
```

```
df = pd.read_csv('/content/drive/MyDrive/Saleh-Project1/Apartment_Market_Prices_missing.csv')
```

```
df.head()
```

	Year	OBJECTID	GEOID	Tract Label	Tract Name	Community Reporting Area Name	Community Reporting Area ID	Tract Median Apartment Contract Rent per Square Foot	Tract Median Apartment Contract Rent per Unit	Year over Year Change in Rent per Square Foot	Year over Change in per
0	2001	1	53033001201	12.01	Census Tract 12.01	Northgate/Maple Leaf	8.1	1.52	1085.0	482.758.620.689.656	513.565.891.4
1	2001	2	53033000404	4.04	Census Tract 4.04	Broadview/Bitter Lake	9.1	1.29	1042.0	403.225.806.451.613	
2	2001	3	53033004901	49.01	Census Tract 49.01	Fremont	11.1	1.79	1042.0	529.411.764.705.883	525.252.525.0
3	2001	4	53033003601	36.01	Census Tract 36.01	Green Lake	9.4	2.10	1197.0	552.763.819.095.478	555.555.555.5
4	2001	5	53033002800	28.00	Census Tract 28	Greenwood/Phinney Ridge	9.3	1.60	NaN	526.315.789.473.685	521.276.595.7

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
len(df)
```

4320

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

	0
Year	0
OBJECTID	0
GEOID	0
Tract Label	0
Tract Name	0
Community Reporting Area Name	0
Community Reporting Area ID	0
Tract Median Apartment Contract Rent per Square Foot	0
Tract Median Apartment Contract Rent per Unit	418
Year over Year Change in Rent per Square Foot	0
Year over Year Change in Rent per Unit	0
Cost Category	0
Year over Year Change in Rent Category	0
Mixed-Rate or Mixed-Income Apartments in Tract	0
PROPERTIES	0
Shape__Area	0
Shape__Length	0

dtype: int64

```
# Calculate the percentage of missing values in each column
missing_percentage = df.isnull().mean() * 100
# Select columns with more than 30% missing values
columns_with_missing_30 = missing_percentage[missing_percentage > 30]
# Print the result
print("Columns with more than 30% missing values:")
print(columns_with_missing_30)
```

Columns with more than 30% missing values:
Series([], dtype: float64)

df.describe()

	Year	OBJECTID	GEOID	Tract Label	Community Reporting Area ID	Tract Median Apartment Contract Rent per Square Foot	Tract Median Apartment Contract Rent per Unit	Mixed-Rate or Mixed-Income Apartments in Tract	PROPERTIES
count	4320.000000	4320.000000	4.320000e+03	4320.000000	4320.000000	4320.000000	3902.000000	4320.000000	4320.000000
mean	2012.500000	2160.500000	5.303301e+10	66.461500	7.476111	1.956067	1291.778831	547.108565	22.421296
std	6.922988	1247.220911	4.255099e+03	42.550986	3.471062	0.932163	622.979289	528.865123	21.058124
min	2001.000000	1.000000	5.303300e+10	1.010000	1.100000	0.000000	0.000000	0.000000	0.000000
25%	2006.750000	1080.750000	5.303300e+10	37.505000	4.300000	1.510000	1050.000000	122.000000	6.000000
50%	2012.500000	2160.500000	5.303301e+10	67.015000	7.400000	2.100000	1355.000000	396.000000	16.000000

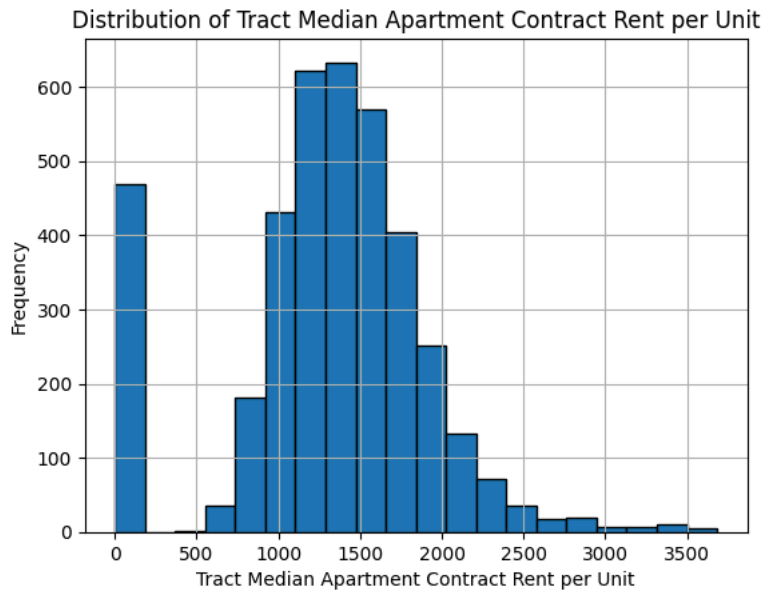
```
print(df['Tract Median Apartment Contract Rent per Unit'].mean())
```

1291.778831368529

```
print(df['Tract Median Apartment Contract Rent per Unit'].median())
```

1355.0

```
# Histogram for the column
column_name = 'Tract Median Apartment Contract Rent per Unit'
# Plot histogram
df[column_name].hist(bins=20, edgecolor='black') # More bins give better detail
plt.title('Distribution of Tract Median Apartment Contract Rent per Unit')
plt.xlabel(column_name)
plt.ylabel('Frequency')
plt.show()
```

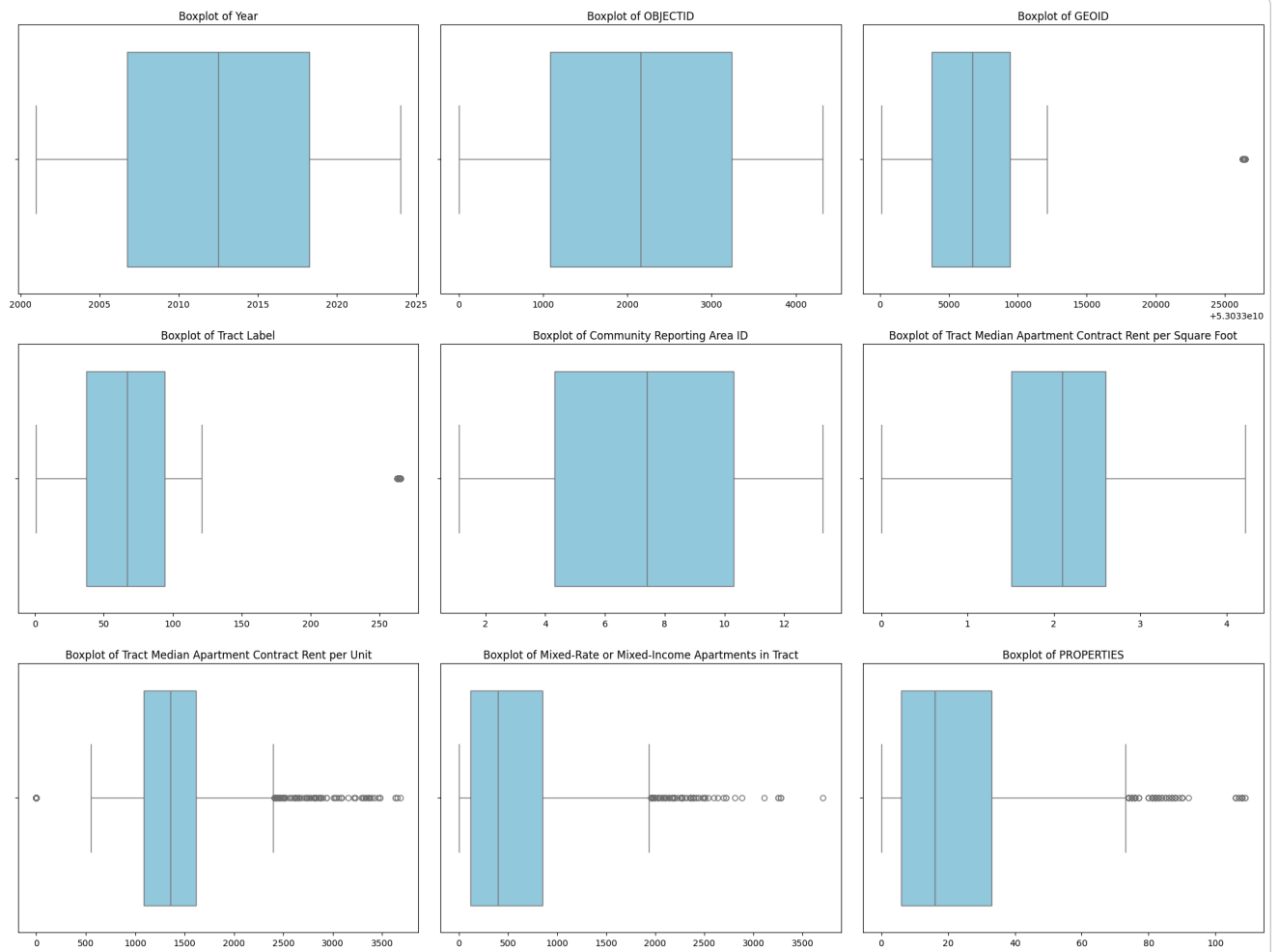


```
# Replace missing values in a column with the column median
column_name = 'Tract Median Apartment Contract Rent per Unit' # your actual column
df[column_name] = df[column_name].fillna(df[column_name].median())
```

```
print(df['Tract Median Apartment Contract Rent per Unit'].isna().sum())
```

```
0
```

```
# Select numeric columns
numerical_df = df.select_dtypes(include=['number'])
num_cols = numerical_df.columns
# Calculate number of rows (3 plots per row)
rows = math.ceil(len(num_cols) / 3)
plt.figure(figsize=(20, 5 * rows))
for i, col in enumerate(num_cols):
    plt.subplot(rows, 3, i + 1)
    sns.boxplot(x=numerical_df[col].dropna(), color='skyblue')
    plt.title(f"Boxplot of {col}")
    plt.xlabel('') # optional
plt.tight_layout()
plt.show() # no extra spaces or parentheses
```



```
def remove_outliers_iqr(data):
    # Select numeric columns
    numeric_data = data.select_dtypes(include='number')

    # Calculate Q1 (25th percentile) and Q3 (75th percentile)
    Q1 = numeric_data.quantile(0.25)
    Q3 = numeric_data.quantile(0.75)
    IQR = Q3 - Q1

    # Define lower and upper bounds
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Filter data: keep only rows where all numeric columns are within bounds
    filtered_data = data[~((numeric_data < lower_bound) | (numeric_data > upper_bound)).any(axis=1)]

    return filtered_data

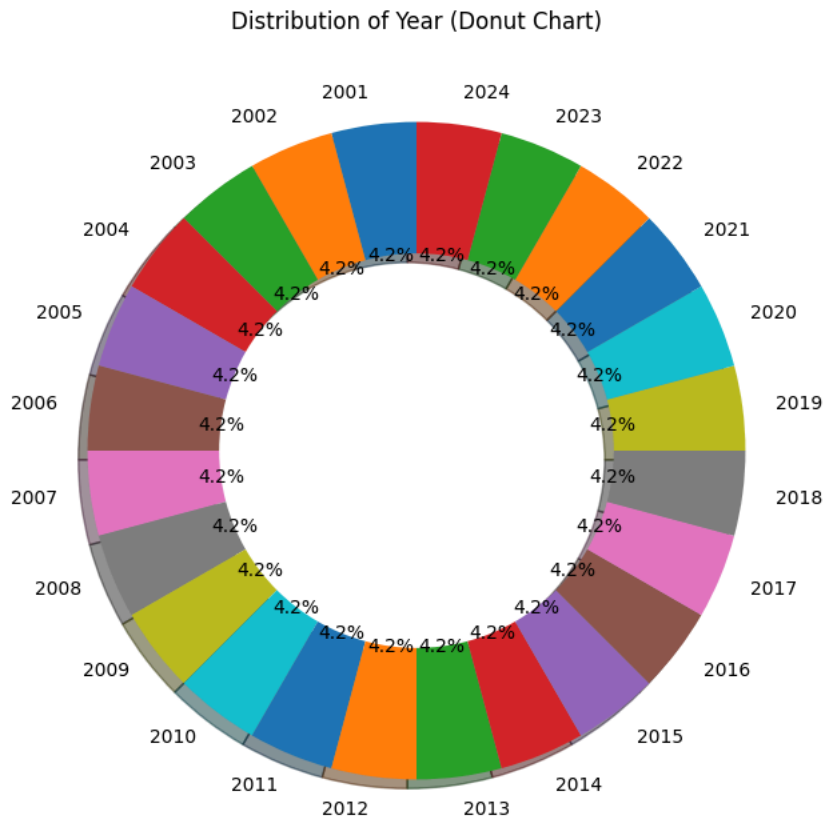
# Apply the function
```

```
cleaned_data = remove_outliers_iqr(df)
print(f"Original rows: {len(df)}, After removing outliers: {len(cleaned_data)}")
```

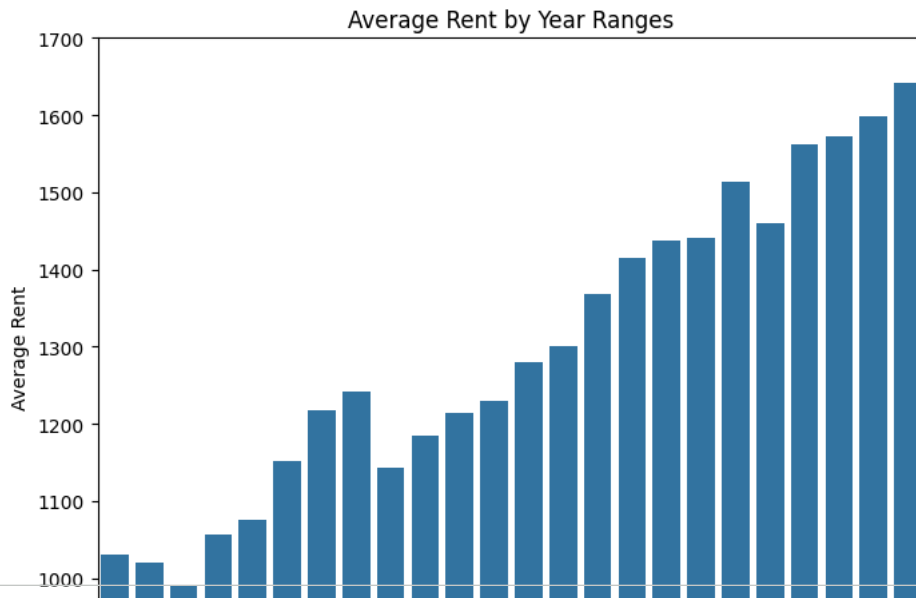
Original rows: 4320, After removing outliers: 3536

```
# Count values in PROPERTIES
category_counts = df['Year'].value_counts()

plt.figure(figsize=(8,8))
plt.pie(
    category_counts,
    labels=category_counts.index,
    autopct='%1.1f%%',
    startangle=90,
    shadow=True,
    wedgeprops={'width': 0.4} # Donut hole
)
plt.title('Distribution of Year (Donut Chart)')
plt.show()
```



```
plt.figure(figsize=(8,6))
sns.barplot(
    x=df['Year'],
    y=df['Tract Median Apartment Contract Rent per Unit'],
    estimator='mean',
    errorbar=None
)
plt.xticks(rotation=90)
plt.title('Average Rent by Year Ranges')
plt.ylabel('Average Rent')
plt.xlabel('Year')
plt.ylim(900, 1700)
plt.show()
```



```
def clean_dotted_number(s):
    try:
        if isinstance(s, float) or s is None:
            return s
        s_clean = s.replace('.', '')
        return float(s_clean) / 1e9
    except Exception:
        return np.nan

df['Tract Median Apartment Contract Rent per Unit'] = df['Tract Median Apartment Contract Rent per Unit'].apply(clean_dotted_nu

df['Tract Median Apartment Contract Rent per Square Foot'] = pd.to_numeric(df['Tract Median Apartment Contract Rent per Square

df = df[df['Cost Category'] != '0'] # remove invalid rows if any

plt.figure(figsize=(8,5))
sns.boxplot(data=df, x='Cost Category', y='Tract Median Apartment Contract Rent per Square Foot')
plt.title('Apartment Rent per Square Foot by Cost Category')
plt.ylabel('Median Contract Rent per Sq Ft ($)')
plt.xlabel('Cost Category')
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```

