```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color_codes=True)
```

```
data = pd.read_csv('/content/raw_house_data - raw_house_data.csv')
data.head(10)
```

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitchen_f
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1941	13	10.0	10500.0	0.0	Dish Refrigerat
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1997	2	2.0	7300.0	0.0	Dish Garbage I
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	1997	2	3.0	NaN	NaN	Dish Garbage I Ref
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7	5.0	9019.0	4.0	Dishwashe Sink, Pantr
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0	3.0	Dish Garbage I Refrigera
5	21528016	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3	4.0	6842.0	3.0	Dish Garbage I Refrigera
6	21610478	2400000.0	85712	-110.883315	32.261069	2.10	19038.42	2001	9	8.0	12025.0	4.0	Dish Garbage I
7	21211741	2500000.0	85750	-110.861002	32.331603	1.07	21646.00	2011	6	8.0	8921.0	4.0	Co Dish Freezer, I
8	21324646	3700000.0	85718	-110.912156	32.343601	6.73	25094.39	2002	5	7.0	5238.0	3.0	Co Dish Freezer, I
9	21812010	3250000.0	85750	-110.837950	32.327575	3.53	18936.11	2007	5	6.0	6480.0	3.0	Dishwashe Sink Range

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
data.shape
```

```
(5000, 16)
```

```
data.dtypes
```

	0
MLS	int64
sold_price	float64
zipcode	int64
longitude	float64
latitude	float64
lot_acres	float64
taxes	float64
year_built	int64
bedrooms	int64
bathrooms	float64
sqrt_ft	float64
garage	float64
kitchen_features	object
fireplaces	float64
floor_covering	object
HOA	object

dtype: object

```
data.isnull().any()
```

	0
MLS	False
sold_price	False
zipcode	False
longitude	False
latitude	False
lot_acres	True
taxes	False
year_built	False
bedrooms	False
bathrooms	True
sqrt_ft	True
garage	True
kitchen_features	True
fireplaces	True
floor_covering	True
HOA	True

dtype: bool

```
median_garage = data['garage'].median()
data['garage']=data['garage'].fillna(median_garage)
median_garage
```

```
3.0
```

```
data['garage'] = data['garage'].astype(int)
print(data['garage'].dtype)
```

```
int64
```

```
bathrooms = data['bedrooms']
data['bathrooms'] = data['bathrooms'].fillna(bathrooms)
data['bathrooms'] = data.apply(lambda row: row['bedrooms'] if row['bathrooms'] > row['bedrooms'] else row['bathrooms'], axis=1)
```

```
data['bathrooms'] = data['bathrooms'].astype(int)
print(data['bathrooms'].dtype)
```

```
int64
```

```
data['floor_covering'] = data['floor_covering'].fillna(0)
```

```
data['kitchen_features'] = data['kitchen_features'].fillna(0)
```

```
median_sqrt_ft = data['sqrt_ft'].median()
data['sqrt_ft']=data['sqrt_ft'].fillna(median_sqrt_ft)

median_sqrt_ft
```

```
3512.0
```

```
median_lot_acres = data['lot_acres'].median()
data['lot_acres']=data['lot_acres'].fillna(median_lot_acres)

median_lot_acres
```

```
0.99
```

```
median_fireplaces = data['fireplaces'].median()
data['fireplaces'] = data['fireplaces'].fillna(median_fireplaces)
```

```
data['fireplaces'] = pd.to_numeric(data['fireplaces'], errors='coerce').fillna(0).astype(int)
print(data['fireplaces'].dtype)
```

```
int64
```

```
data['HOA'] = pd.to_numeric(data['HOA'], errors='coerce')
median_hoa = data['HOA'].median()
data['HOA'] = data['HOA'].fillna(median_hoa)
```

```
data.isnull().sum()
```

	0
MLS	0
sold_price	0
zipcode	0
longitude	0
latitude	0
lot_acres	0
taxes	0
year_built	0
bedrooms	0
bathrooms	0
sqrt_ft	0
garage	0
kitchen_features	0
fireplaces	0
floor_covering	0
HOA	0

dtype: int64

```
1 data.dtypes
```

	0
MLS	int64
sold_price	float64
zipcode	int64
longitude	float64
latitude	float64
lot_acres	float64
taxes	float64
year_built	int64
bedrooms	int64
bathrooms	int64
sqrt_ft	float64
garage	int64
kitchen_features	object
fireplaces	int64
floor_covering	object
HOA	float64

dtype: object

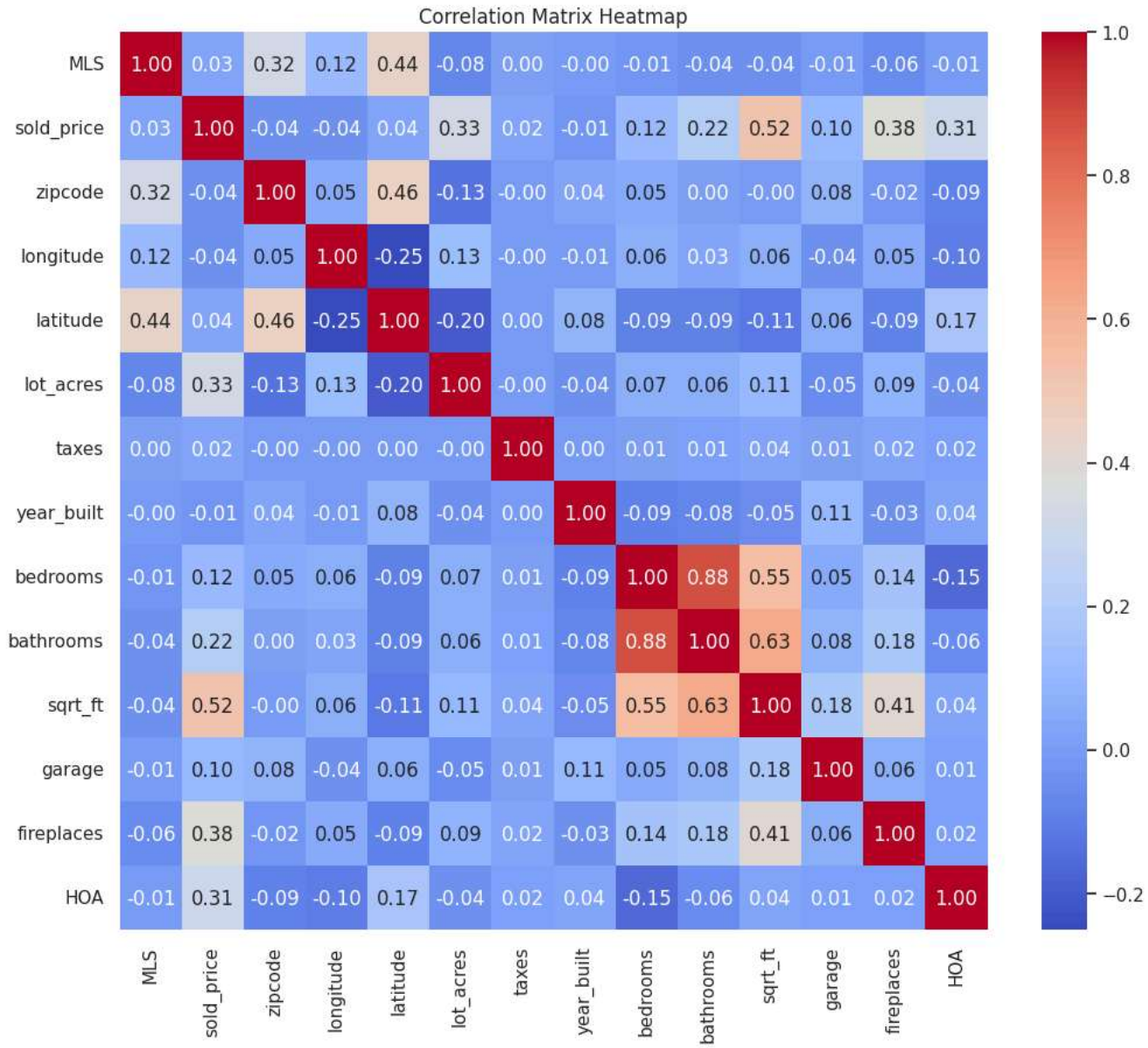
```
1 data.to_csv('file2.csv', header=False, index=False)
2 data.head()
```

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitchen_fe
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1941	13	10	10500.0	0	Dish Refrigerat
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1997	2	2	7300.0	0	Dish Garbage I
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	1997	2	2	3512.0	3	Dish Garbage D Ref
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7	5	9019.0	4	Dishwashe Sink, Pantr
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	4	6396.0	3	Dish Garbage D Refrigera

Next steps: Generate code with `data` View recommended plots New interactive sheet

```
1 numeric_data = data.select_dtypes(include=[np.number])
2 correlation_matrix = numeric_data.corr()
```

```
1   plt.figure(figsize=(12, 10))  # Adjust figure size as needed
2   sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
3   plt.title('Correlation Matrix Heatmap')
4   plt.show()
```




```
1 data['price_per_sqft'] = data['sold_price'] / data['sqrt_ft']
```

```
1 data
```

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitche
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1941	13	10	10500.0	0	Refrig
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1997	2	2	7300.0	0	Garba
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	1997	2	2	3512.0	3	Garba
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7	5	9019.0	4	Dishwa Sink, P
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	4	6396.0	3	Garba Refri
...	...	...	...	...	...	...	...	...	...	...	...	...	
4995	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3	3601.0	3	Dishwa Si Dis
4996	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3	2318.0	3	Dishwa S Rai
4997	21832452	475000.0	85192	-110.755428	32.964708	12.06	1000.00	1969	3	2	1772.0	0	Ele Island,
4998	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4	3724.0	3	Dishwa Si Dis
4999	4111490	450000.0	85621	-110.913054	31.385259	4.16	2814.48	1988	4	4	4317.0	3	Dishwa Sink,

5000 rows × 17 columns

Next steps: Generate code with `data` | View recommended plots | New interactive sheet

```
1 data['Cat_price_per_sqft'] = pd.qcut(data['price_per_sqft'], q=20, labels=False, duplicates='drop')
```

```
1 data
```

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitche
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1941	13	10	10500.0	0	Refrig
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1997	2	2	7300.0	0	Garba
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	1997	2	2	3512.0	3	Garba
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7	5	9019.0	4	Dishwa Sink, P
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	4	6396.0	3	Garba Refri
...	...	...	...	...	...	...	...	...	...	...	...	...	
4995	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3	3601.0	3	Dishwa Si Dis
4996	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3	2318.0	3	Dishwa S Rai
4997	21832452	475000.0	85192	-110.755428	32.964708	12.06	1000.00	1969	3	2	1772.0	0	Ele Island,
4998	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4	3724.0	3	Dishwa Si Dis
4999	4111490	450000.0	85621	-110.913054	31.385259	4.16	2814.48	1988	4	4	4317.0	3	Dishwa Sink,

5000 rows × 18 columns

Next steps: Generate code with data | View recommended plots | New interactive sheet

KNN Classifier

```
# Prepare the Data
X_train = data[['longitude', 'latitude']].values
y_knn = data['Cat_price_per_sqft'].values
```

```
class KNNClassifier:
    def fit(self, X, y):
        self.X = X
        self.y = y

    def predict(self, X, K=3, epsilon=1e-5):
        N = len(X)
        y_hat = np.zeros(N)

        for i in range(N):
            dist2 = np.sum((self.X - X[i]) ** 2, axis=1)
            idxt = np.argsort(dist2)[:K]
            gamma_k = 1 / (np.sqrt(dist2[idxt] + epsilon))
            y_hat[i] = np.bincount(self.y[idxt].astype(int), weights=gamma_k).argmax()

        return y_hat
```

```
def accuracy(y,y_hat):
    return np.mean(y==y_hat)
```

```
knn = KNNClassifier()
```

```
knn.fit(X_train,y_knn)
```

```
y_hat = knn.predict(X_train,K=3)
```

```
accuracy = np.mean(y_hat==y_knn)
accuracy
```

```
0.9078
```

```
data['Category_price_per_sqft'] = y_hat
```

```
data
```

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitche
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1941	13	10	10500.0	0	Refrig
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1997	2	2	7300.0	0	Garba
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	1997	2	2	3512.0	3	Garba
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7	5	9019.0	4	Dishwa Sink, P
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	4	6396.0	3	Garba Refri
...	...	...	...	...	...	...	...	...	...	...	...	...	
4995	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3	3601.0	3	Dishwa S Dis
4996	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3	2318.0	3	Dishwa S Rai
4997	21832452	475000.0	85192	-110.755428	32.964708	12.06	1000.00	1969	3	2	1772.0	0	Ele Island,
4998	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4	3724.0	3	Dishwa S Dis
4999	4111490	450000.0	85621	-110.913054	31.385259	4.16	2814.48	1988	4	4	4317.0	3	Dishwa Sink,

5000 rows × 19 columns

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
data['rooms'] = data['bedrooms'] + data['bathrooms']
```

```
data
```

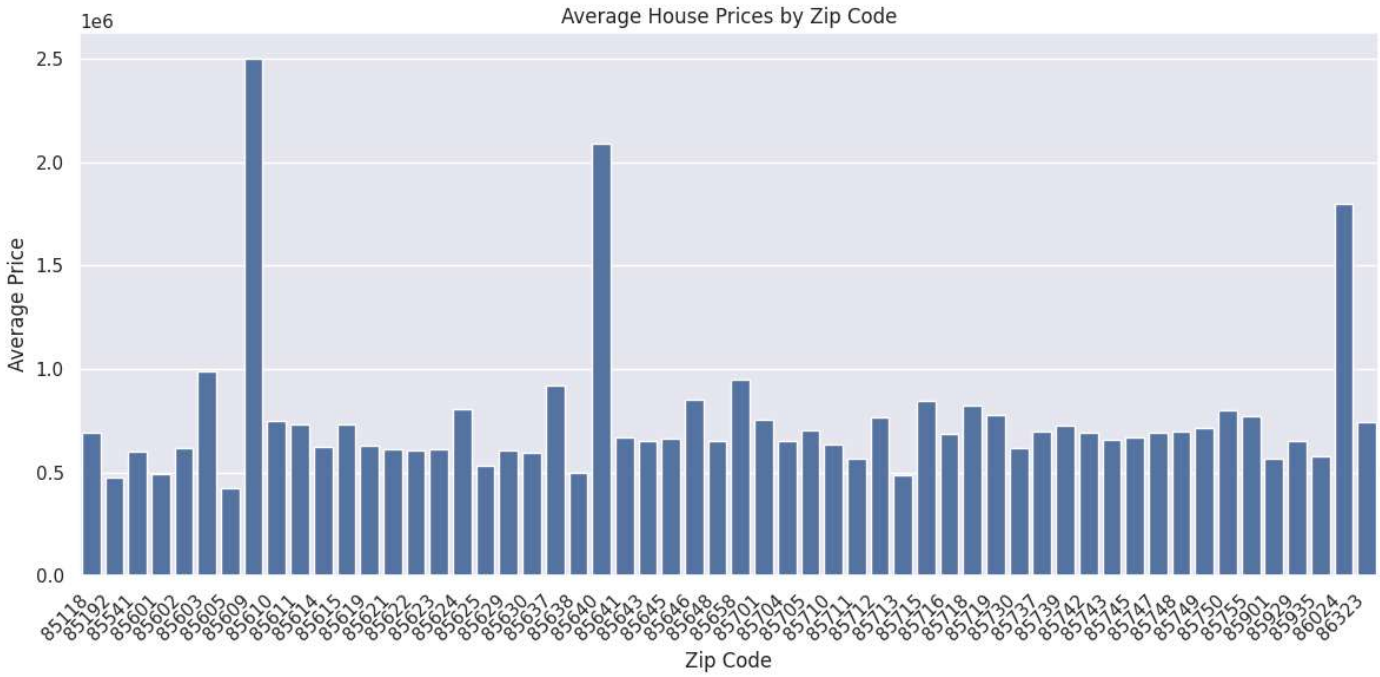
	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitche
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1941	13	10	10500.0	0	Refrig
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1997	2	2	7300.0	0	Garba
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	1997	2	2	3512.0	3	Garba
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7	5	9019.0	4	Dishwa Sink, P
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	4	6396.0	3	Garba Refri
...	...	...	...	...	...	...	...	...	...	...	...	...	
4995	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3	3601.0	3	Dishwa Si Dis
4996	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3	2318.0	3	Dishwa S Rai
4997	21832452	475000.0	85192	-110.755428	32.964708	12.06	1000.00	1969	3	2	1772.0	0	Ele Island,
4998	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4	3724.0	3	Dishwa Si Dis
4999	4111490	450000.0	85621	-110.913054	31.385259	4.16	2814.48	1988	4	4	4317.0	3	Dishwa Sink,

5000 rows × 20 columns

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
1 average_prices = data.groupby('zipcode')['sold_price'].mean().reset_index()
```

```
1 plt.figure(figsize=(12, 6))  # Adjust figure size as needed
2 sns.barplot(x='zipcode', y='sold_price', data=average_prices)
3 plt.title('Average House Prices by Zip Code')
4 plt.xlabel('Zip Code')
5 plt.ylabel('Average Price')
6 plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
7 plt.tight_layout()
8 plt.show()
```




```
#zipcode_of_interest = 85266  # Replace with your desired zip code #85750
#filtered_data = data[data['zipcode'] == zipcode_of_interest]
```

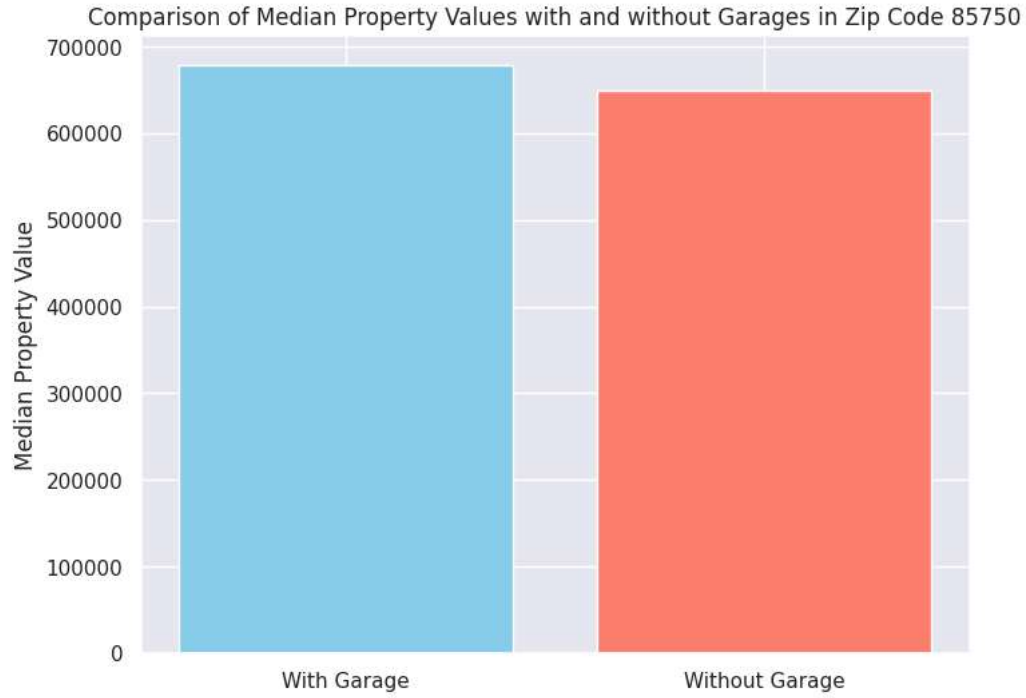
```
zipcode_of_interest = 85750  # Replace with your desired zip code #85750
filtered_data = data[data['zipcode'] == zipcode_of_interest]
```

```
median_with_garage = filtered_data[filtered_data['garage'] > 0]['sold_price'].median()
median_without_garage = filtered_data[filtered_data['garage'] == 0]['sold_price'].median()
```

```
median_with_garage = data[data['garage'] > 0]['sold_price'].median()
median_without_garage = data[data['garage'] == 0]['sold_price'].median()
```

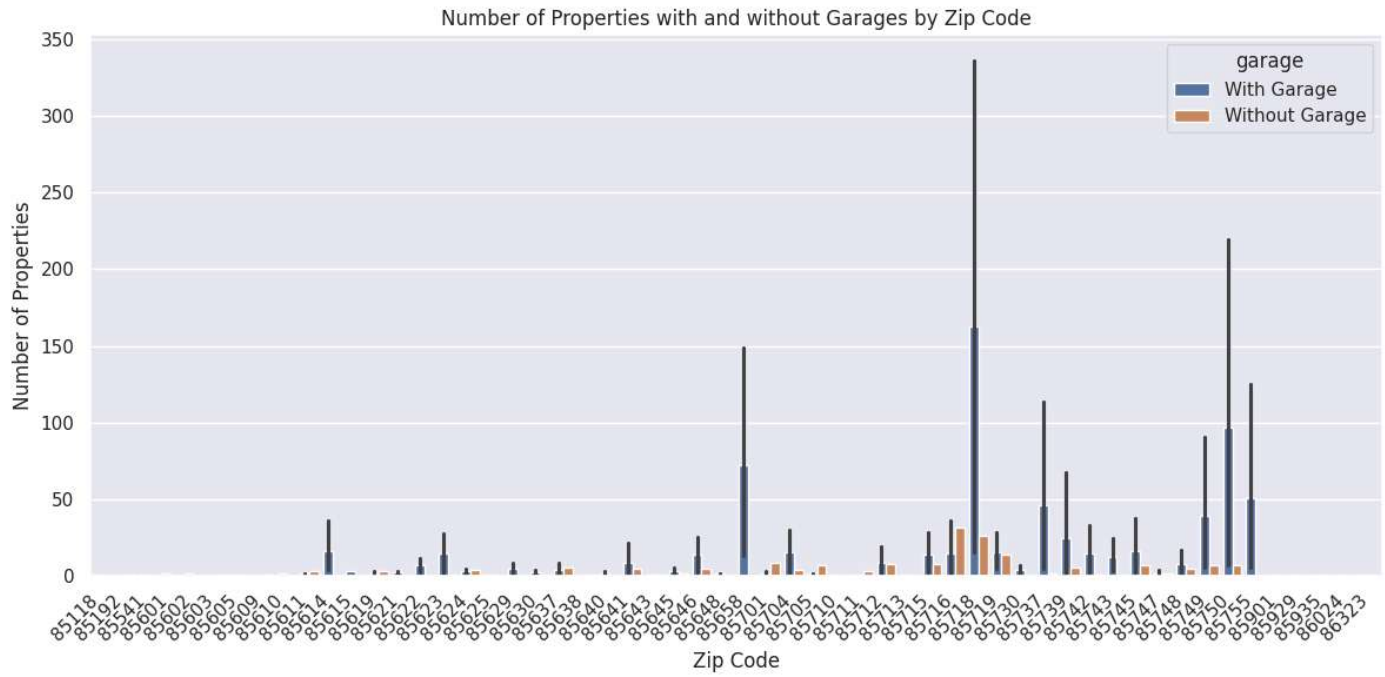
```
categories = ['With Garage', 'Without Garage']
median_values = [median_with_garage, median_without_garage]

plt.figure(figsize=(8, 6))
plt.bar(categories, median_values, color=['skyblue', 'salmon'])
plt.title(f'Comparison of Median Property Values with and without Garages in Zip Code {zipcode_of_interest}')
plt.ylabel('Median Property Value')
plt.show()
```



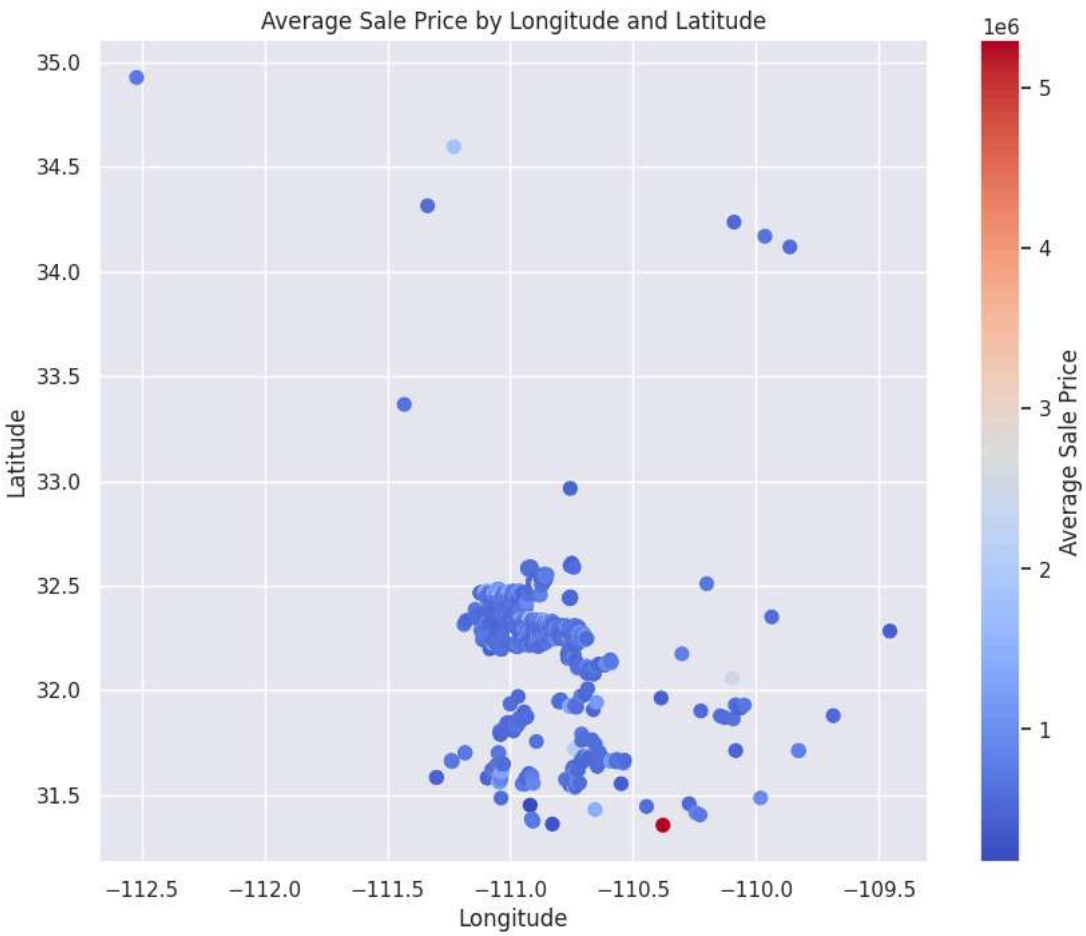

```
garage_counts = data.groupby(['zipcode', 'garage'])['sold_price'].count().reset_index()
garage_counts = garage_counts.rename(columns={'sold_price': 'count'})
garage_counts['garage'] = garage_counts['garage'].apply(lambda x: 'With Garage' if x > 0 else 'Without Garage')
```

```
plt.figure(figsize=(12, 6))
sns.barplot(x='zipcode', y='count', hue='garage', data=garage_counts)
plt.title('Number of Properties with and without Garages by Zip Code')
plt.xlabel('Zip Code')
plt.ylabel('Number of Properties')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



```
average_prices = data.groupby(['longitude', 'latitude'])['sold_price'].mean().reset_index()
```

```
plt.figure(figsize=(10, 8))
plt.scatter(average_prices['longitude'], average_prices['latitude'],
            c=average_prices['sold_price'], cmap='coolwarm', s=50)  # Adjust 's' for marker size
plt.colorbar(label='Average Sale Price')
plt.title('Average Sale Price by Longitude and Latitude')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



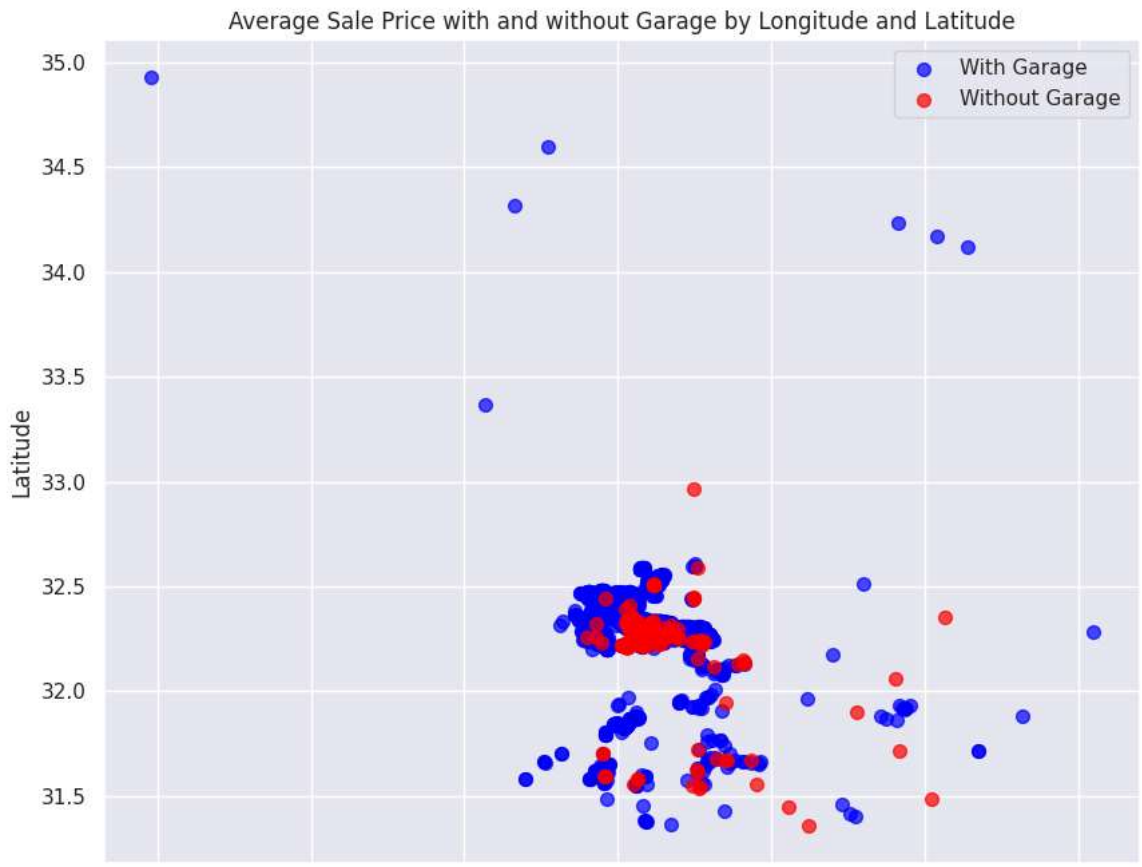

```
average_prices = data.groupby(['longitude', 'latitude', (data['garage'] > 0)])['sold_price'].mean().reset_index()
average_prices.rename(columns={'(data[\'garage\'] > 0)': 'has_garage', 'sold_price': 'avg_price'}, inplace=True)

# Create a scatter plot with color-coded points and legend
plt.figure(figsize=(10, 8))

# Separate data for houses with and without garages
with_garage = average_prices[average_prices['garage'] == True]
without_garage = average_prices[average_prices['garage'] == False]

# Plot points with different colors and labels
plt.scatter(with_garage['longitude'], with_garage['latitude'],
            c='blue', label='With Garage', s=50, alpha=0.7)
plt.scatter(without_garage['longitude'], without_garage['latitude'],
            c='red', label='Without Garage', s=50, alpha=0.7)

plt.legend()  # Add the legend
plt.title('Average Sale Price with and without Garage by Longitude and Latitude')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



KNN Regressor

```
# Prepare the Data
X_train = data[['longitude','latitude','garage','lot_acres','rooms','price_per_sqft','Category_price_per_sqft']].values
y_knnr = data['sold_price'].values
```

```
class KNNRegressor():
    def fit(self,X,y):
        self.X = X
        self.y = y
        self.train_errors = []

    def predict(self,X,K,epsilon = 1e-4):
        N = len(X)
        y_hat = np.zeros(N)

        for i in range(N):
            dist2 = np.sum((self.X-X[i])**2,axis = 1)
            idxt = np.argsort(dist2)[:K]
            gamma_k = np.exp(-dist2[idxt])/(np.exp(-dist2[idxt]).sum()+epsilon)
            y_hat[i] = gamma_k.dot(self.y[idxt])

```