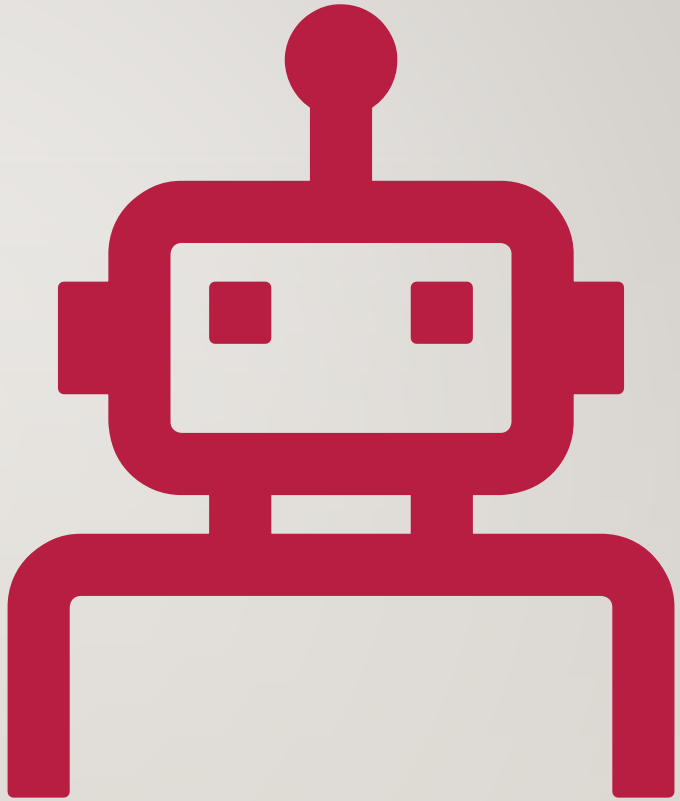
USECASE:

AS THE JUNIOR DATA SCIENTIST, MY TASK IS TO PREPROCESS AND EXPLORE THE HOUSING DATASET THOROUGHLY BEFORE HANDING IT OVER TO THE MODELING TEAM.

PRESENTED BY: LAKSHMI SRAVANI



INDEX

INITIAL DATA LOADING

Imported Libraries:

Pandas: For data loading and Manipulation

NumPy: Handling numeric operations and NAN

Matplotlib.pyplot: Visualizations

Seaborn : Enhanced Plots and EDA Visuals

Loaded Dataset:

Data=pd.read_csv("raw_house_data.csv)

PREVIEWED DATASET STRUCTURE

Data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   MLS               5000 non-null   int64
 1   sold_price        5000 non-null   float64
 2   zipcode           5000 non-null   int64
 3   longitude         5000 non-null   float64
 4   latitude          5000 non-null   float64
 5   lot_acres         4990 non-null   float64
 6   taxes             5000 non-null   float64
 7   year_built        5000 non-null   int64
 8   bedrooms          5000 non-null   int64
 9   bathrooms         4994 non-null   float64
 10  sqrt_ft           4944 non-null   float64
 11  garage            4993 non-null   float64
 12  kitchen_features  4967 non-null   object
 13  fireplaces        5000 non-null   object
 14  floor_covering    4999 non-null   object
 15  HOA               4438 non-null   object
dtypes: float64(8), int64(4), object(4)
memory usage: 625.1+ KB
```

Missing values in features: lot_acres, bathrooms, garage, sqrt_ft, kitchen_features, HOA

Zipcode stored as int should be categorical

HOA and fireplaces stored as object/string, but contain numeric-like values

DATA CLEANING

- ✓ Replaced NON-Standard Null Values:

```
data.replace(['None', 'none', 'n/a', '',' ' ,'NA','NaN'], np.nan, inplace=True)
```

- ✓ Dropped Irrelevant Columns:

```
data.drop(['latitude', 'longitude'], axis=1, inplace=True)
```

- ✓ Converted Data Types:

```
data['zipcode']=data['zipcode'].astype(str)
data['sqrt_ft']=data['sqrt_ft'].astype(float)
data['garage'] = data['garage'].astype(float)
data['bathrooms'] = data['bathrooms'].astype(float)
data['HOA'] = pd.to_numeric(data['HOA'], errors='coerce').fillna(0).astype(int)
data['fireplaces'] = pd.to_numeric(data['fireplaces'], errors='coerce').fillna(0).astype(int)
```

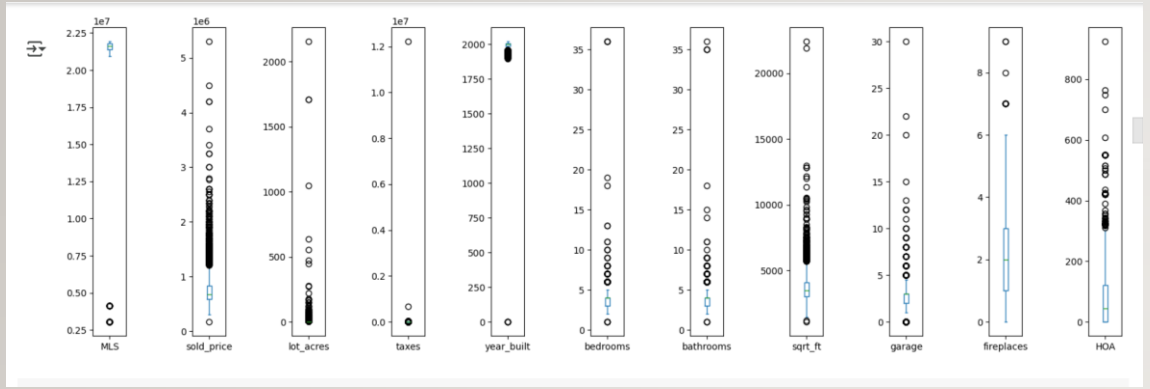
MISSING VALUES IMPUTATION

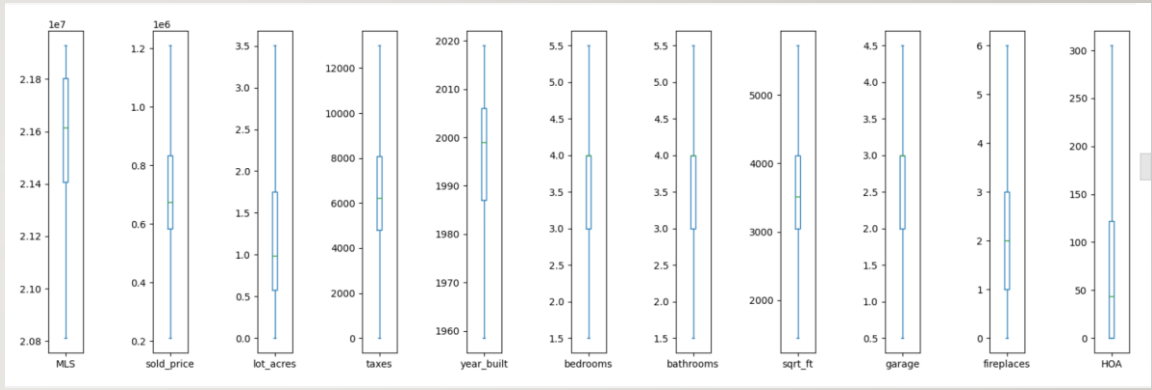
```
for col in ['garage', 'bathrooms', 'sqrt_ft', 'lot_acres']:
    data[col].fillna(data[col].median(), inplace=True)

for col in ['kitchen_features', 'floor_covering']:
    data[col].fillna(data[col].mode()[0], inplace=True)


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

- ✓ Imputed Categorical columns(Using Mode)
- ✓ Imputed Numerical Columns(Using Median)
- ✓ HOA Conversation and Imputation





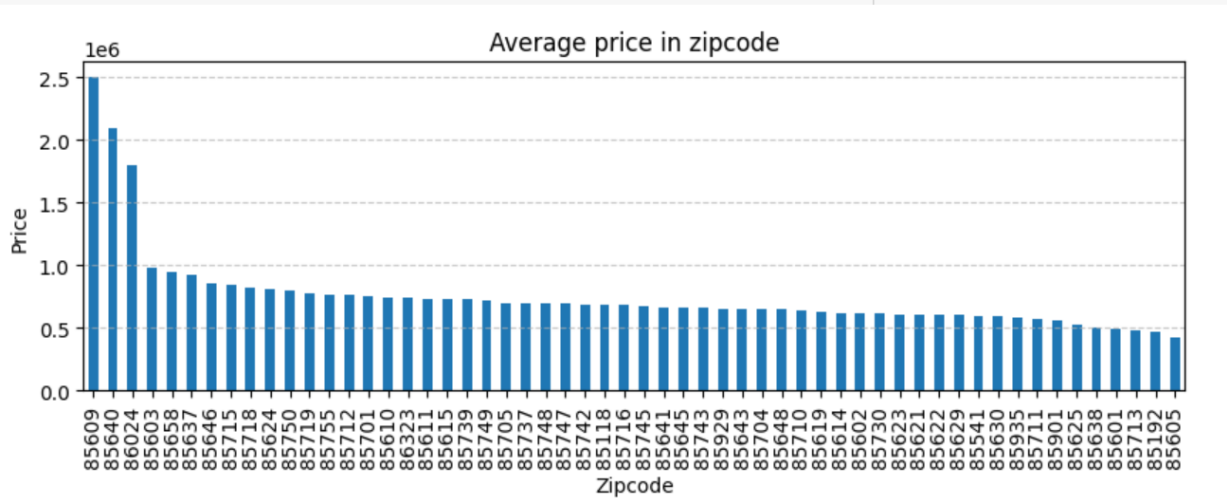
OUTLIERS DETECTION AND HANDLING

USING CLIP(): DATA[COL] = DATA[COL].CLIP(LOWER=LOWER_BOUND, UPPER=UPPER_BOUND)

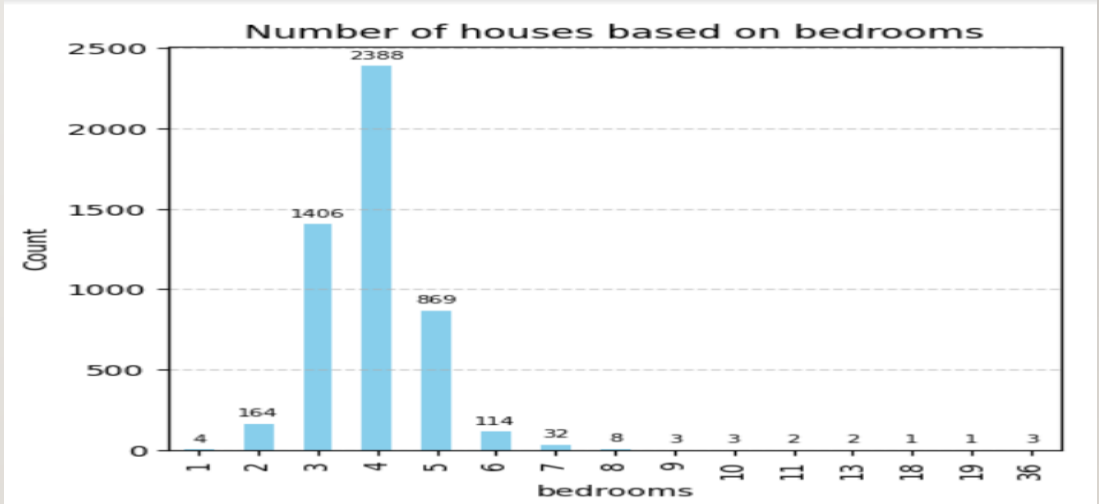
Univariate Analysis: Sold_price, sqrt_ft, lot_acres, taxes using box-plots

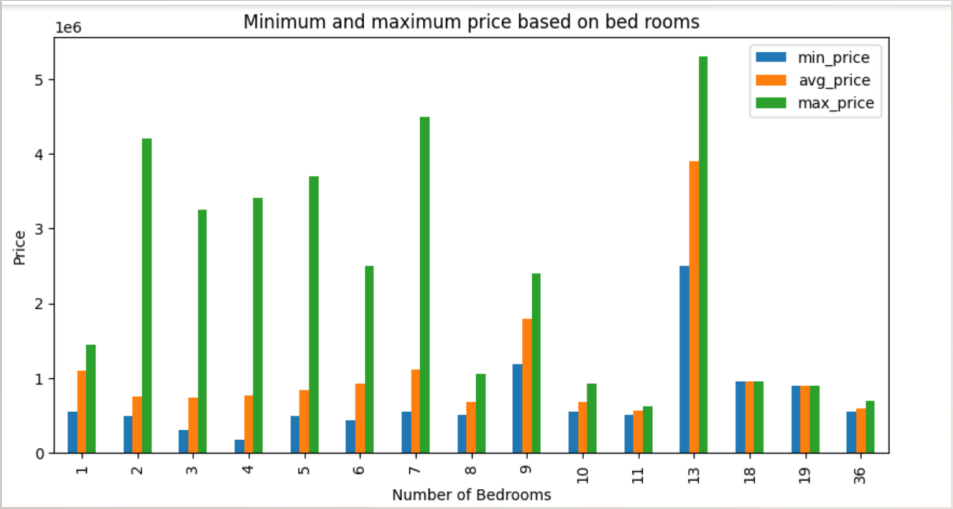
EXPLORATORY DATA ANALYSIS(EDA)

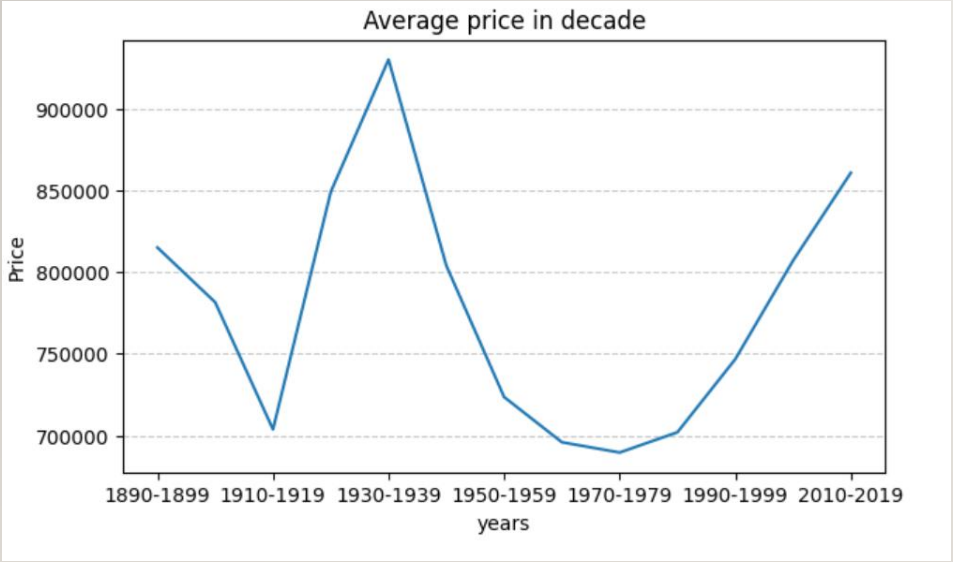
✓ Bivariate Analysis(Sold_price by Zip Code)



✓ Bivariate Analysis(House Count by bedrooms)



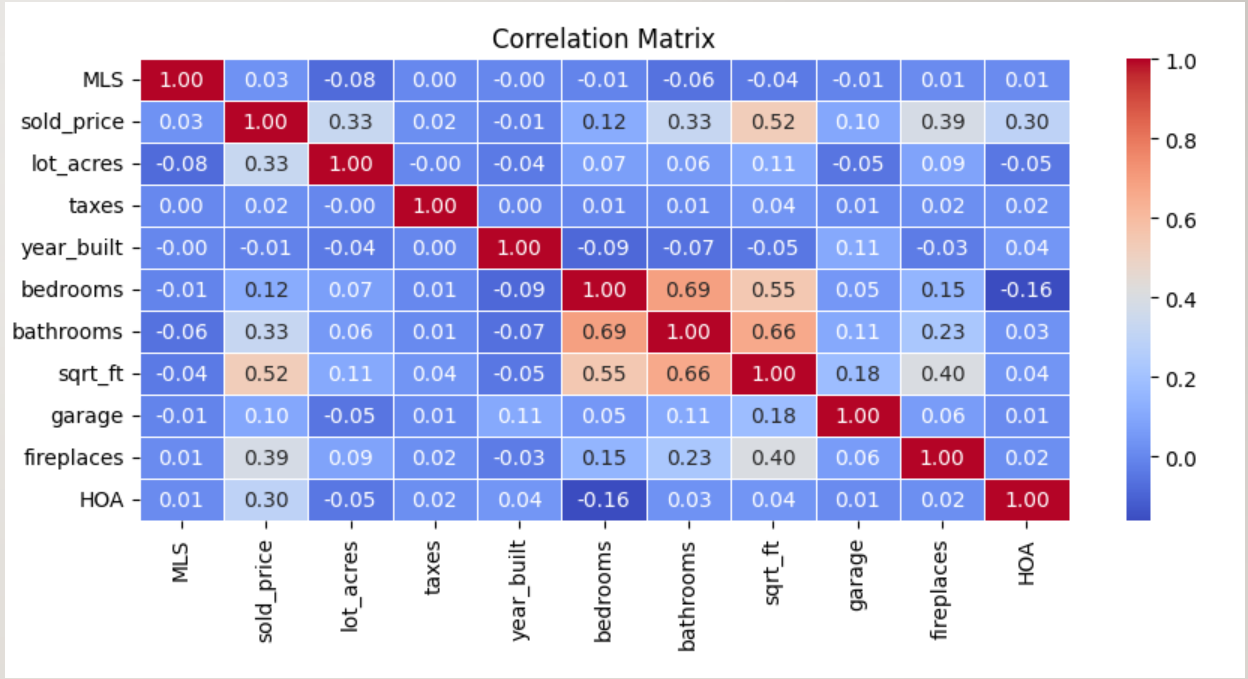




- ✓ Grouped Bar Charts: bedrooms, fireplaces, garage(min,Avg,Max)
- ✓ Trend Analysis : Sold_price by Year_built

CORRELATION ANALYSIS

- ✓ Heatmap generated using data.corr()
- ✓ Sold_price shows string positive correlation with sqrt_ft,bathroom,garage
- ✓ Features with low or no-correlation(like fireplaces,bedrooms) were flagged for lower model priority



FINAL DATASET AND HANDOFF SUMMARY

Final Dataset Summary:

- ✓ Total Records: 5000, with no missing values
- ✓ Outliers handled using IQR clipping
- ✓ Cleaned and transformed features

Model-Ready Features:

- ✓ Key predictors: sqrt_ft, bathrooms, garage, lot_acres, taxes, year_built, HOA, zipcode, price_per_sqft
- ✓ Categorical variables: properly encoded or grouped (e.g., HOA status, kitchen features)

Ready for Modeling:

- ✓ Dataset is clean, structured, and normalized
- ✓ Suitable for regression, tree-based models, or advanced ML pipelines
- ✓ Next step: feature selection, training, and model benchmarking

THANK YOU