# USA Real Estate: Analysis and Prediction House Price in the US

*GROUP 2*

# OUTLINE

1. Research Purpose
2. Data Processing with PySpark
3. Exploratory Data Analysis
4. Predictive Modeling
5. Conclusion

# I. Research Purpose

# Research Purpose

## Understanding the U.S. Real Estate Market

The U.S. housing market is a complex, data-rich environment shaped by wide range of factors

Understanding how these factors affect **housing prices** is crucial for:

- 🏠 **Homebuyers**
- 💼 **Investors & Developers**
- 🏦 **Financial institutions**

# Research Purpose

## FOCUS QUESTION OF THIS NOTEBOOK

1. What is the overall picture of the USA housing prices w.r.t. locations?
2. Do house attributes (bedroom, bathroom count) strongly correlate with the price? Are there any hidden patterns?
3. How are housing price and location attributes correlated?
4. Can we predict housing prices based on the features?

# Research Purpose

Supports investors in identifying undervalued or overvalued markets.

Helps buyers and sellers make informed decisions on pricing.

Understand the key factors influencing home prices - predictive machine learning models

Serves as a baseline for automated real estate valuation systems.

# II. Data Processing with PySpark

# 1. Data Collection & Import

## Step 1: Initialize SparkSession

```
In [ ]: spark = SparkSession.builder.appName("RealtorDataAnalysis").getOrCreate()
```

- Starts a new Spark session named "RealtorDataAnalysis" so we can work with Spark DataFrames.
- SparkSession is the entry point to use Spark SQL and DataFrame operations.

---

## Step 2: Data Import

```
In [ ]: df = spark.read.csv("realtor-data.csv", header=True, inferSchema=True)
        df.printSchema()
        df.show(5)
```

To load the real estate dataset (realtor-data.csv) into PySpark, automatically detect its structure, and preview its contents before performing data cleaning or analysis.

# 1. Data Collection & Import

## Schema

```
root
 |-- brokered_by: double (nullable = true)
 |-- status: string (nullable = true)
 |-- price: double (nullable = true)
 |-- bed: integer (nullable = true)
 |-- bath: integer (nullable = true)
 |-- acre_lot: double (nullable = true)
 |-- street: double (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- zip_code: integer (nullable = true)
 |-- house_size: double (nullable = true)
 |-- prev_sold_date: date (nullable = true)
```

Nullable = true means the column can contain null values, i.e., missing data.

To inspect the data structure before processing (ETL) or performing analysis.

# 1. Data Collection & Import

## The dataset contains Real Estate listings in the US (by State and Zip code)

EXAMPLE

| brokered_by | status | price | bed | bath | acre_lot | street | city | state | zip_code | house_size | prev_sold_date |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 103378.0 | for_sale | 105000.0 | 3 | 2 | 0.12 | 1962661.0 | Adjuntas | Puerto Rico | 601 | 920.0 | NULL |
| 52707.0 | for_sale | 80000.0 | 4 | 2 | 0.08 | 1902874.0 | Adjuntas | Puerto Rico | 601 | 1527.0 | NULL |
| 103379.0 | for_sale | 67000.0 | 2 | 1 | 0.15 | 1404990.0 | Juana Diaz | Puerto Rico | 795 | 748.0 | NULL |
| 31239.0 | for_sale | 145000.0 | 4 | 2 | 0.1 | 1947675.0 | Ponce | Puerto Rico | 731 | 1800.0 | NULL |
| 34632.0 | for_sale | 65000.0 | 6 | 2 | 0.05 | 331151.0 | Mayaguez | Puerto Rico | 680 | NULL | NULL |

only showing top 5 rows

## Source: Realtor.com

**Full data can be seen in the report*

# 2. Data Cleaning

## Step 1 : Casting Column Data Types

In [ ]:

```
from pyspark.sql.functions import col
from pyspark.sql.types import IntegerType

df = df.withColumn("brokered_by", col("brokered_by").cast(StringType()))
df = df.withColumn("street", col("street").cast(StringType()))
df = df.withColumn("zip_code", col("zip_code").cast(StringType()))
```

Ensure that these columns are treated as strings (categorical).
This is useful for correct null handling or further processing like mode imputation.

---

## Step 2 : Importing Required Libraries

```
from pyspark.sql import Row
```

Importing Row to help convert Python objects into Spark DataFrame rows.

# 2. Data Cleaning

### Step 3: Count Total Rows in the Dataset

```
total_rows = df.count()
```

Stores the number of total records in the DataFrame, used later to compute percentages.

---

### Step 4: Count Null Values in Each Column

```
null_counts = df.select([
    (count(when(col(c).isNull(), c)).alias(c)) for c in df.columns
]).collect()[0].asDict()
```

- Loops over each column in the DataFrame.
- Uses when(...).isNull() to count how many nulls are in each column.
- Stores the result as a dictionary with format {column_name: null_count}.

# 2. Data Cleaning

## Step 5: Calculate Null Percentage Per Column

```
null_percentage_rows = [
    Row(column_name=col_name, null_pct=null_count / total_rows * 100)
    for col_name, null_count in null_counts.items()
]
```

For every column, calculate the null percentage:

null_pct = (null_count / total_rows) * 100

Save it as a list of Row objects.

---

## Step 6: Create a New DataFrame to Display Results

```
pivoted_df = spark.createDataFrame(null_percentage_rows)

# Show the result
pivoted_df.show()
```

# 2. Data Cleaning

| column_name | null_pct |
|---|---|
| brokered_by | 0.20360387390843082 |
| status | 0.0 |
| price | 0.06921543562605159 |
| bed | 21.618796774318152 |
| bath | 22.986666259428976 |
| acre_lot | 14.624130090882876 |
| street | 0.48805640721134114 |
| city | 0.063196702093351 46 |
| state | 3.593273750865754... |
| zip_code | 0.013429860643860756 |
| house_size | 25.533982937339594 |
| prev_sold_date | 32.98162669299339 |

Almost all columns get **null** value. The column gets **highest** null percentage is **prev_sold_date** *(around 33%)*.

Suggested ways to deal with null value:

1. Drop NaN for columns that has null **> 5%**.
2. If **0% <= null percentage <= 5%** and variable is **numeric** then fill null value with **mean value.**
3. If **0% <= null percentage <= 5%** and variable is **categorical** then fill null value with **mode value.**
4. Remove columns with **highest null value** & **insignificant** to the objective of this topic which is **prev_sold_date.**

# 2. Data Cleaning

| brokered_by | status | price | bed | bath | acre_lot | street | city | state | zip_code | house_size |
|---|---|---|---|---|---|---|---|---|---|---|
| 103378.0 | for_sale | 105000.0 | 3 | 2 | 0.12 | 1962661.0 | Adjuntas | Puerto Rico | 601 | 920.0 |
| 52707.0 | for_sale | 80000.0 | 4 | 2 | 0.08 | 1902874.0 | Adjuntas | Puerto Rico | 601 | 1527.0 |
| 103379.0 | for_sale | 67000.0 | 2 | 1 | 0.15 | 1404990.0 | Juana Diaz | Puerto Rico | 795 | 748.0 |
| 31239.0 | for_sale | 145000.0 | 4 | 2 | 0.1 | 1947675.0 | Ponce | Puerto Rico | 731 | 1800.0 |
| 103378.0 | for_sale | 179000.0 | 4 | 3 | 0.46 | 1850806.0 | San Sebastian | Puerto Rico | 612 | 2520.0 |
| 1205.0 | for_sale | 50000.0 | 3 | 1 | 0.2 | 1298094.0 | Ciales | Puerto Rico | 639 | 2040.0 |
| 50739.0 | for_sale | 71600.0 | 3 | 2 | 0.08 | 1048466.0 | Ponce | Puerto Rico | 731 | 1050.0 |
| 81909.0 | for_sale | 100000.0 | 2 | 1 | 0.09 | 734904.0 | Ponce | Puerto Rico | 730 | 1092.0 |
| 65672.0 | for_sale | 300000.0 | 5 | 3 | 7.46 | 1946226.0 | Las Marias | Puerto Rico | 670 | 5403.0 |
| 52707.0 | for_sale | 89000.0 | 3 | 2 | 13.39 | 1902814.0 | Isabela | Puerto Rico | 662 | 1106.0 |
| 52707.0 | for_sale | 150000.0 | 3 | 2 | 0.08 | 1773902.0 | Juana Diaz | Puerto Rico | 795 | 1045.0 |
| 46019.0 | for_sale | 155000.0 | 3 | 2 | 0.1 | 1946165.0 | Lares | Puerto Rico | 669 | 4161.0 |
| 52707.0 | for_sale | 79000.0 | 5 | 2 | 0.12 | 1761024.0 | Utuado | Puerto Rico | 641 | 1620.0 |
| 88441.0 | for_sale | 649000.0 | 5 | 5 | 0.74 | 1879215.0 | Ponce | Puerto Rico | 731 | 2677.0 |
| 50739.0 | for_sale | 120000.0 | 3 | 2 | 0.08 | 17854.0 | Yauco | Puerto Rico | 698 | 1100.0 |
| 51202.0 | for_sale | 235000.0 | 4 | 4 | 0.22 | 13687.0 | Mayaguez | Puerto Rico | 680 | 3450.0 |
| 12876.0 | for_sale | 105000.0 | 3 | 2 | 0.08 | 1868721.0 | Ponce | Puerto Rico | 728 | 1500.0 |
| 109906.0 | for_sale | 575000.0 | 3 | 2 | 3.88 | 1312671.0 | San Sebastian | Puerto Rico | 685 | 4000.0 |
| 46019.0 | for_sale | 140000.0 | 6 | 3 | 0.25 | 6710.0 | Anasco | Puerto Rico | 610 | 1230.0 |
| 52707.0 | for_sale | 50000.0 | 2 | 1 | 0.23 | 1902835.0 | Yauco | Puerto Rico | 698 | 621.0 |

*Only showing top 20 rows*

*Rows: 1361241, Columns: 11*

# 2. Data Cleaning

The dataset snapshot provides a **clear view** of real estate listings with diverse features.

The accompanying null counts confirm that all missing data has been **handled properly**, making this dataset **reliable** for further exploratory data analysis or predictive modeling.

# 3. Remove Outlier

## Why Remove Outliers?

- Can distort statistical measures and skew model performance.
- Often originate from data entry errors or rare cases.

## Outcome

Clean data to improve reliability and modeling accuracy.

## Steps

1. Detecting outliers with boxplots
2. Define outlier removal function
3. Apply the outlier removal function
4. Check distribution after cleaning

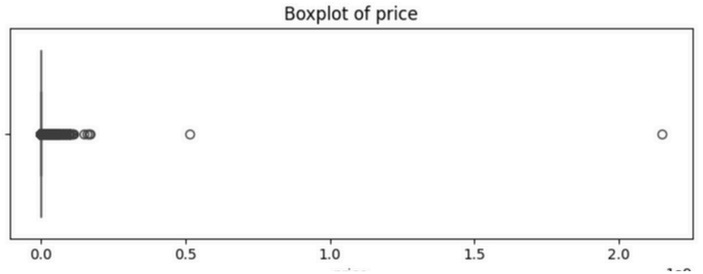
# 3. Remove Outlier

## Step 1: Detecting outliers with boxplots

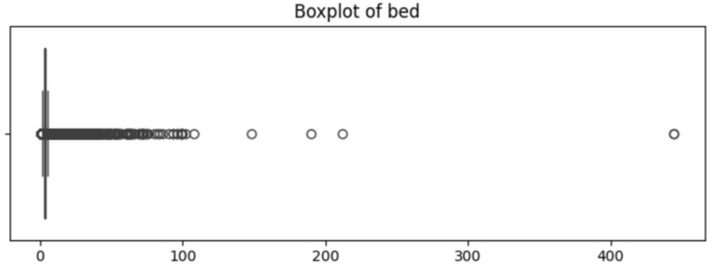
- **Goal:** Detect extreme values in key numeric columns.
- **Method:** Use boxplots to visualize data distribution.
- **Columns analyzed:** price, bed, bath, acre_lot, house_size.

**Comment:**

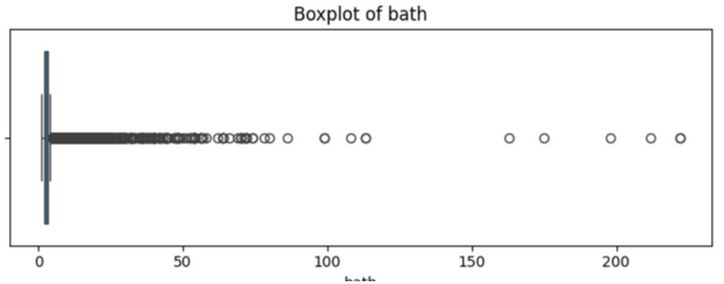
Most numerical features have **extreme outliers**, leading to skewed distributions.



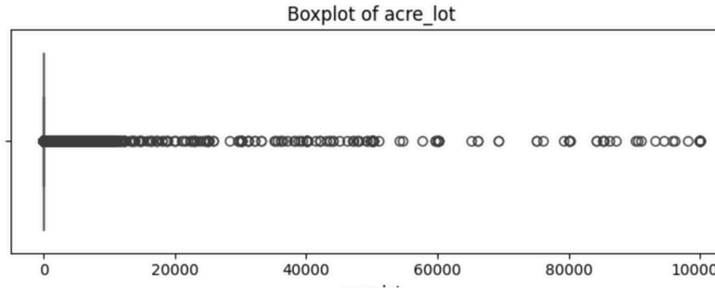

Price



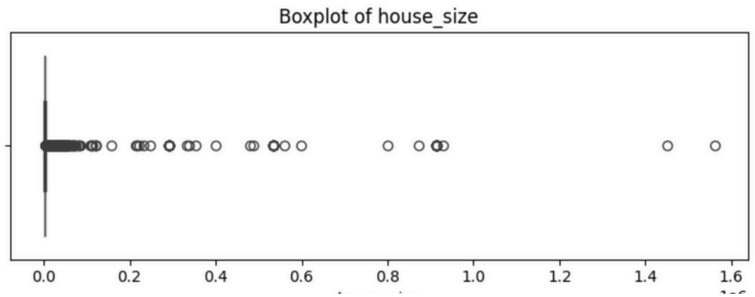

Bed




Bath




Acre_lot




House's price

# 3. Remove Outlier

## 📌 Step 2: Define outlier removal function

- **Goal:** Remove those outliers based on a statistical rule (IQR) to improve data quality.
- **Method:** Use Interquartile Range (IQR) to define acceptable bounds.

## 📌 Step 3: Apply outlier removal to multiple columns

- **Goal:** Remove those outliers by columns in order to save time.
- **Method:** Loop through each column and apply the remove_outliers() function

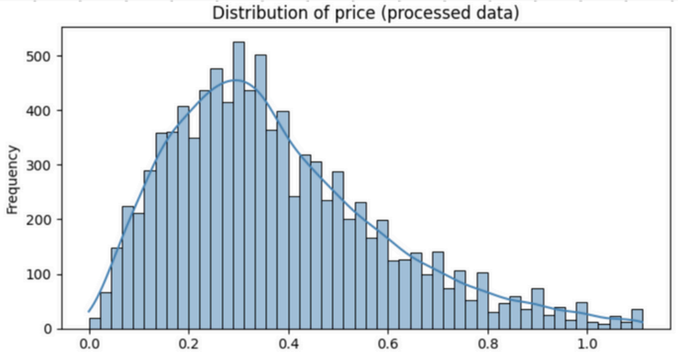
# 3. Remove Outlier

## Step 4: Check distribution after cleaning

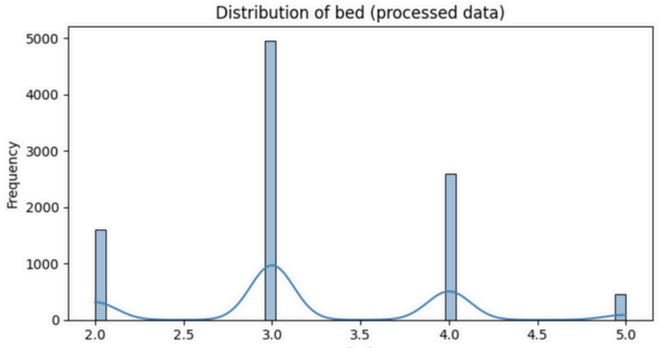
- **Goal:** Verify the effectiveness of outlier removal by visualizing the cleaned data.
- **Method:** Use histogram or KDE plots to compare distributions before and after cleaning.

**Comment:**

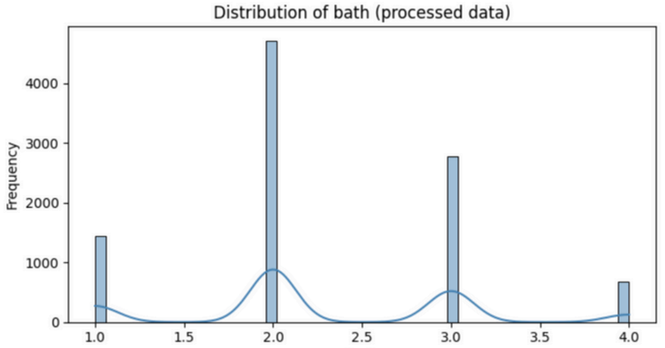
Data is smoother, more centered distributions => Ready for modeling.



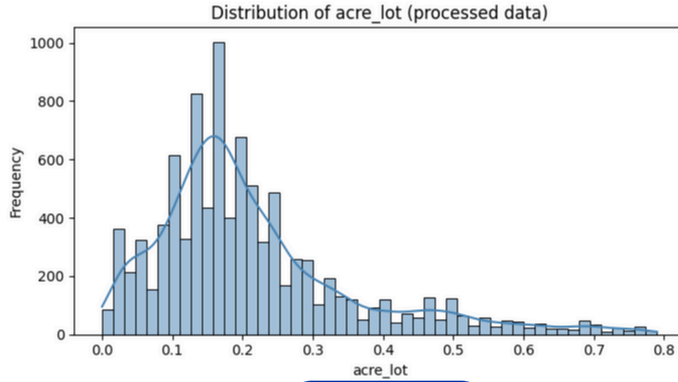

Price



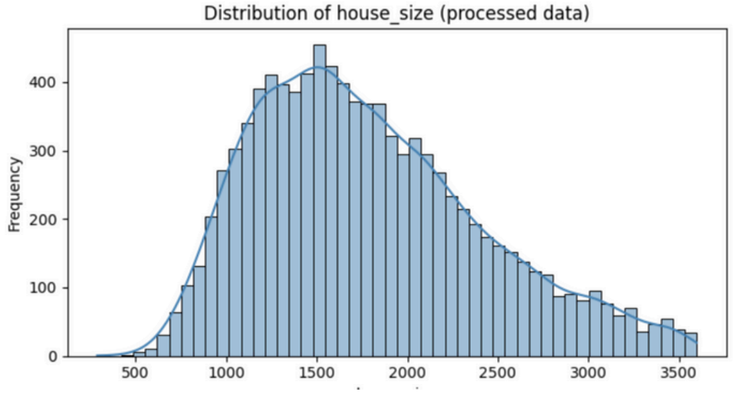

Bed




Bath




Acre_lot



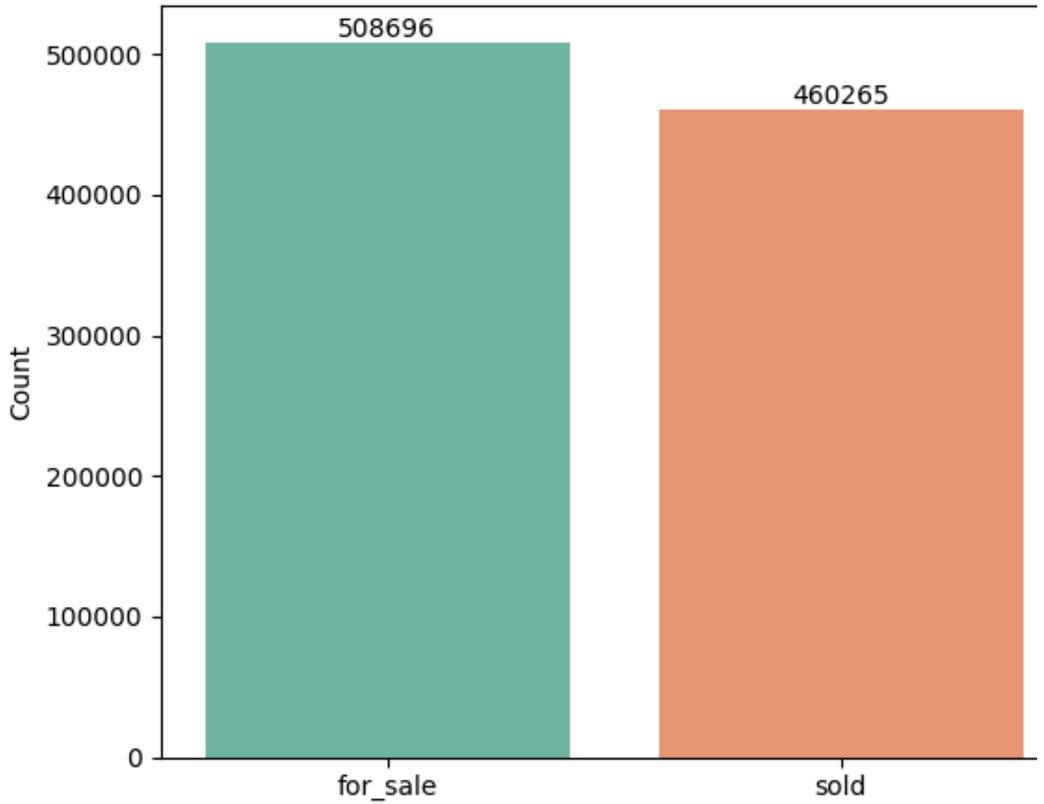

House's price

# III. Explanatory Data Analysis (EDA)

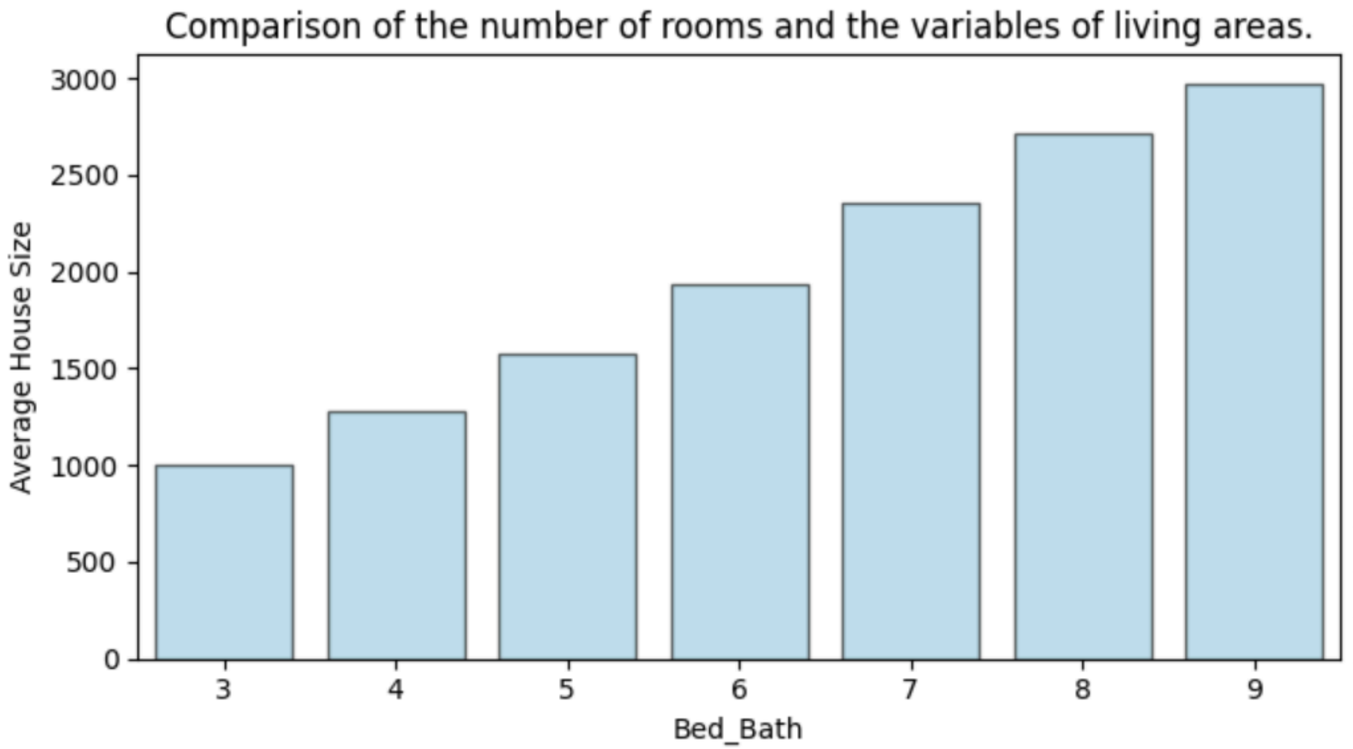
# 1. Status variable

- The status variable shows the transaction stage *(e.g., for sale, pending, sold)*.

- Most listings are in the for sale stage, indicating high market activity.



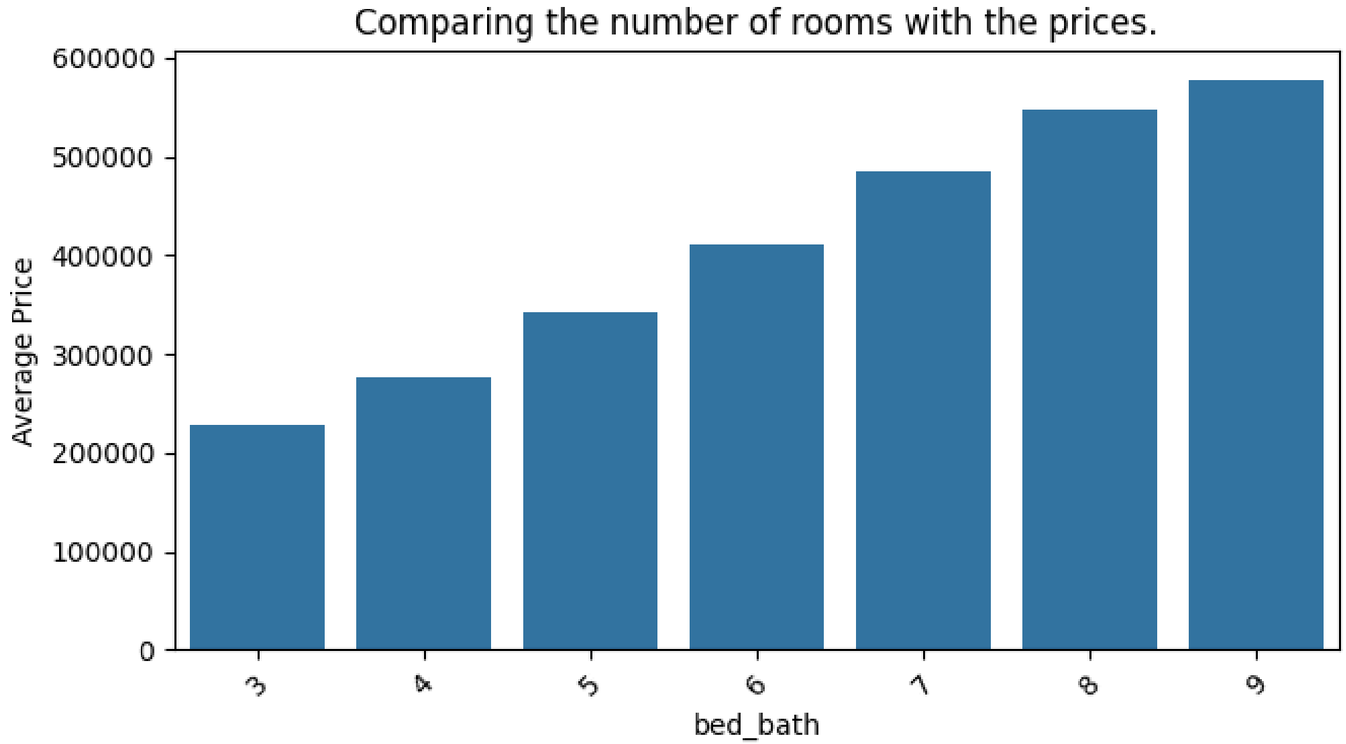
# 2. Bed-Bath vs House Size Comparison




- The average house size **increases consistently** with the number of rooms.
- Homes with **9** total rooms have nearly triple the space compared to those with only **3**.

# 3. Bed - Bath vs Price Comparison

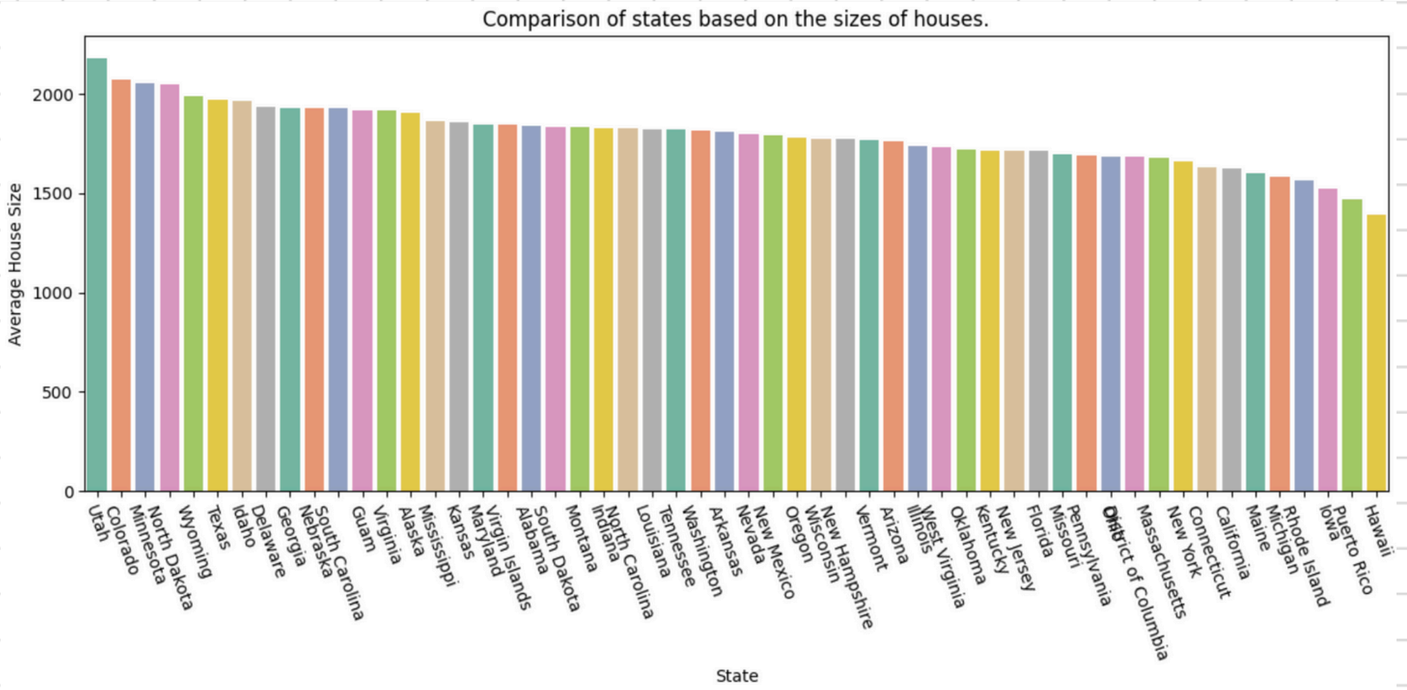
- There is a **clear positive relationship** between the number of rooms and average house prices.
- Larger homes with more rooms tend to command significantly higher market values.



# 4. State - House Size Comparison

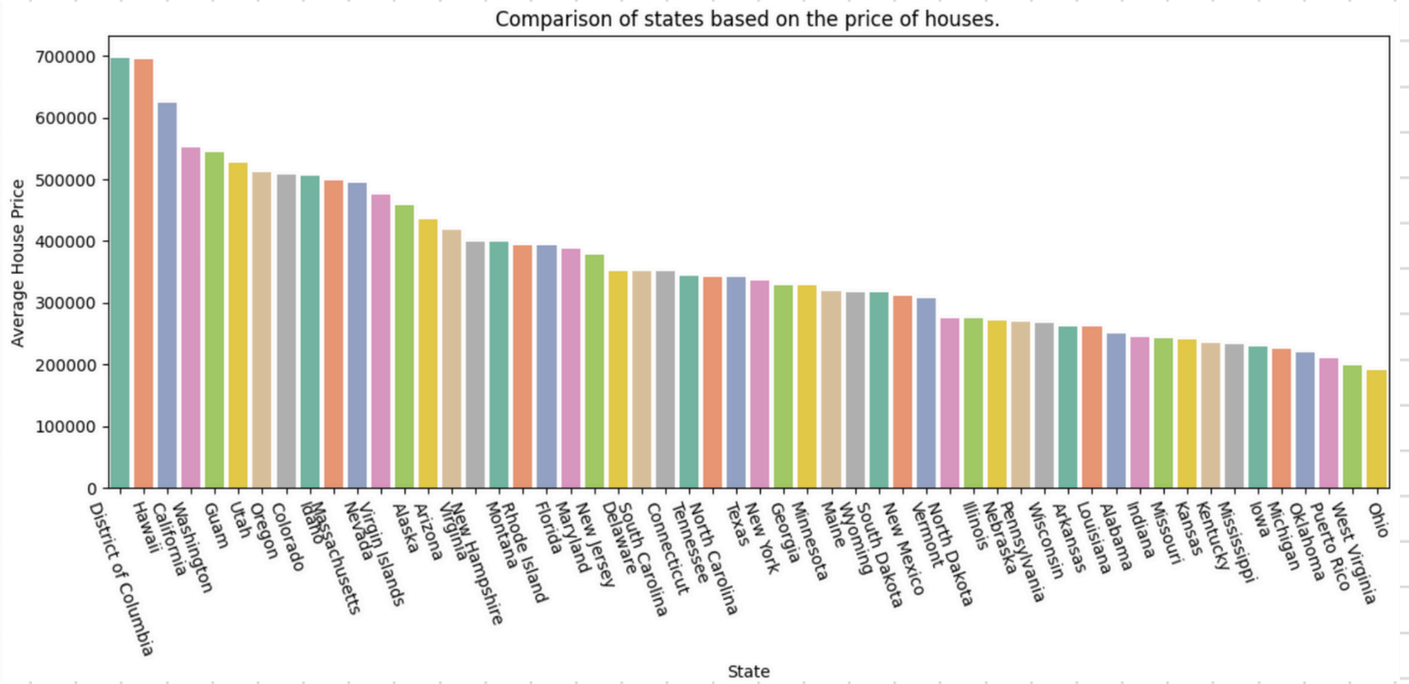
- In **Utah**, the living areas of houses are **generally larger than** in other states → **more spacious homes.**

- In contrast, **Hawaii** has the smallest living areas among the states → homes there are **less spacious.**

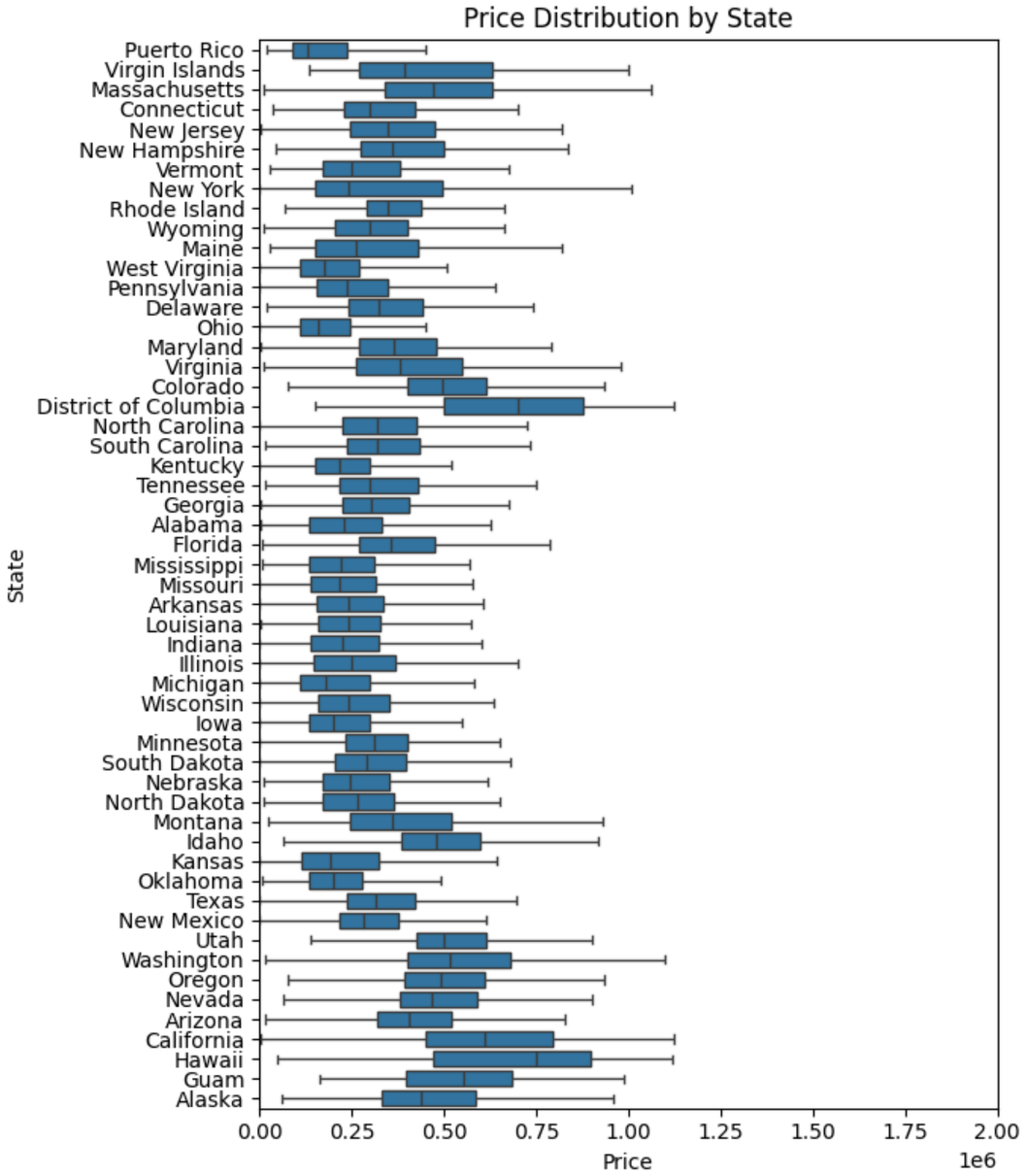


# 5. State vs Price Comparison

- **Average home prices** vary by state, but **larger living spaces** isn't synonymous with **higher prices**.
- The highest prices are in **DC, Hawaii,** and **California**, highlighting location's stronger impact than size.



# 6. Price Distribution for Each State



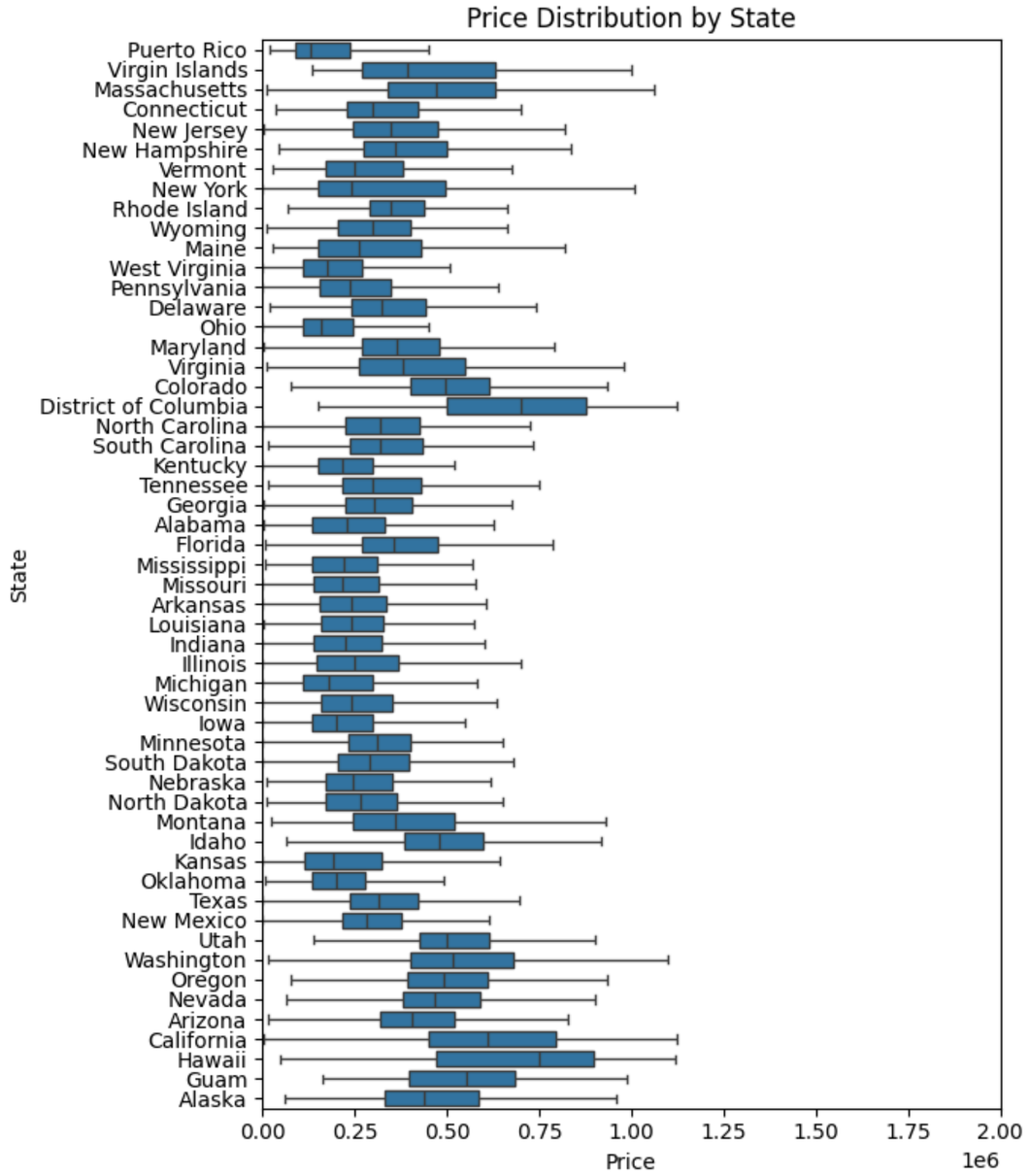

- District of Columbia, Hawaii, and California: highest median house prices

➤ The most expensive markets in the U.S.

- Puerto Rico, Mississippi, and West Virginia: the lowest price medians

➤ More affordable housing markets

# 6. Price Distribution for Each State



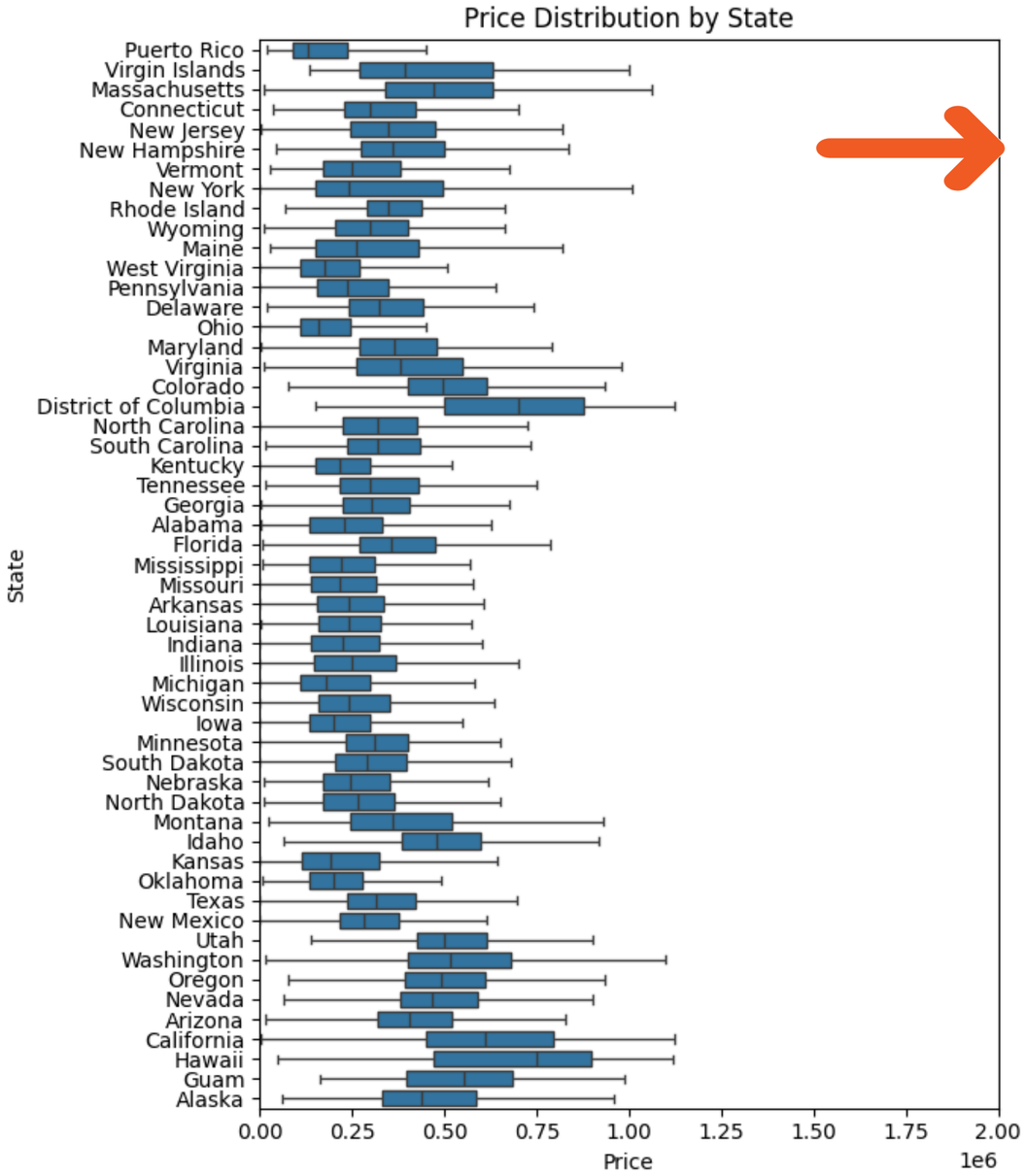

- Some states such as Alaska, Guam, and Hawaii show wide price ranges

➤ Greater price variability — possibly due to differing regions (urban vs. rural) or property types.

- States such as Texas, Florida, and Arizona: moderate prices with broader variability

➤ A diverse housing market

# 6. Price Distribution for Each State

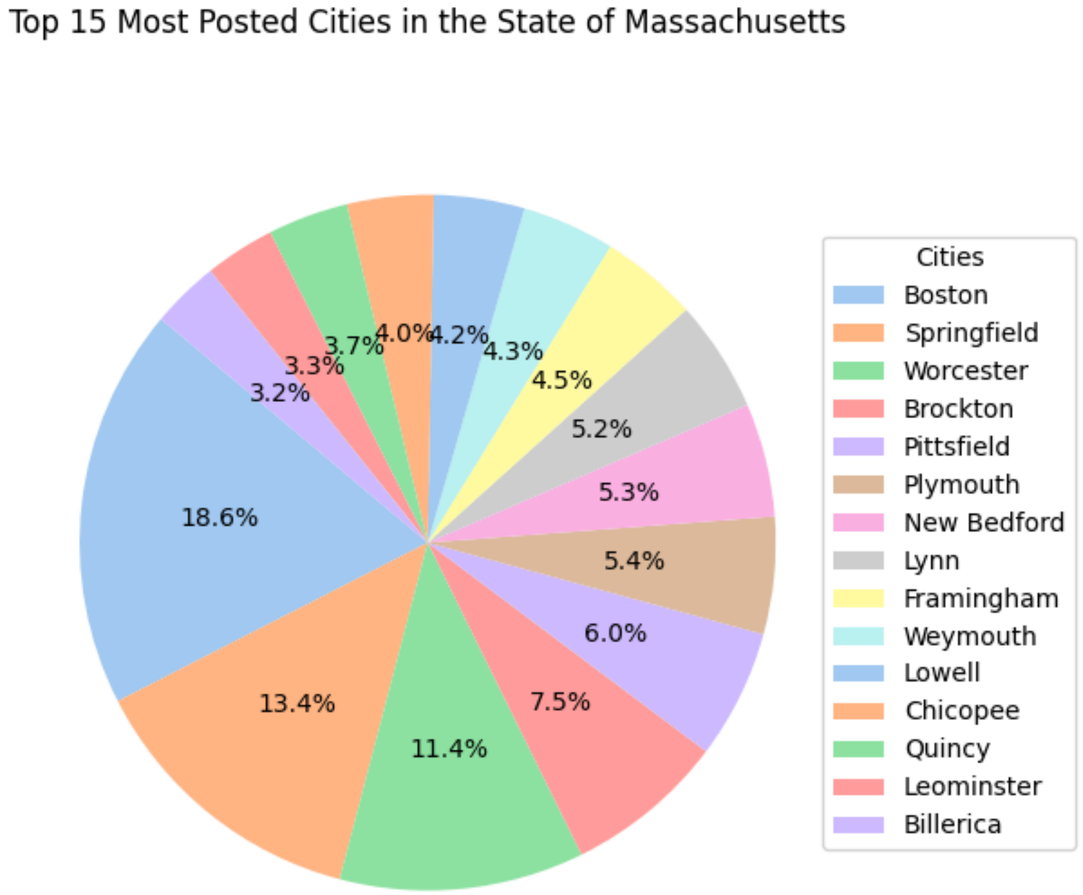



The U.S. housing market shows **significant regional variation** in price, influenced by location, demand, urbanization, and economic conditions.

Understanding state-level price distributions helps in:

- **Identifying** high-cost vs. affordable regions,
- **Targeting** investments geographically,
- **Adjusting models** for regional features in pricing predictions.

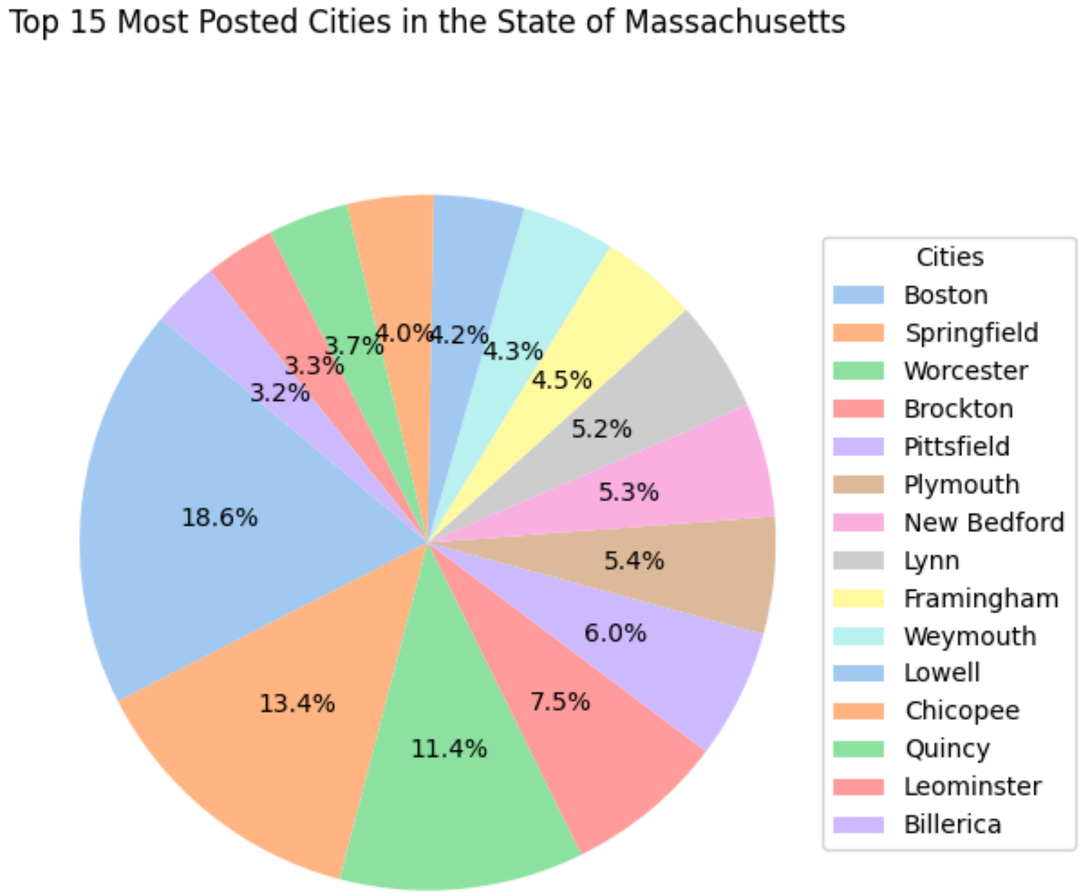
# 7. City Proportions By The State




## E.g. State of Massachusetts

- Boston (18.6%) - the most active market in the state.
- Springfield (13.4%) & Worcester (11.4%) follow, together comprising over 40% of listings.
- There is still a healthy breadth of activity across many smaller communities.

# 7. City Proportions By The State



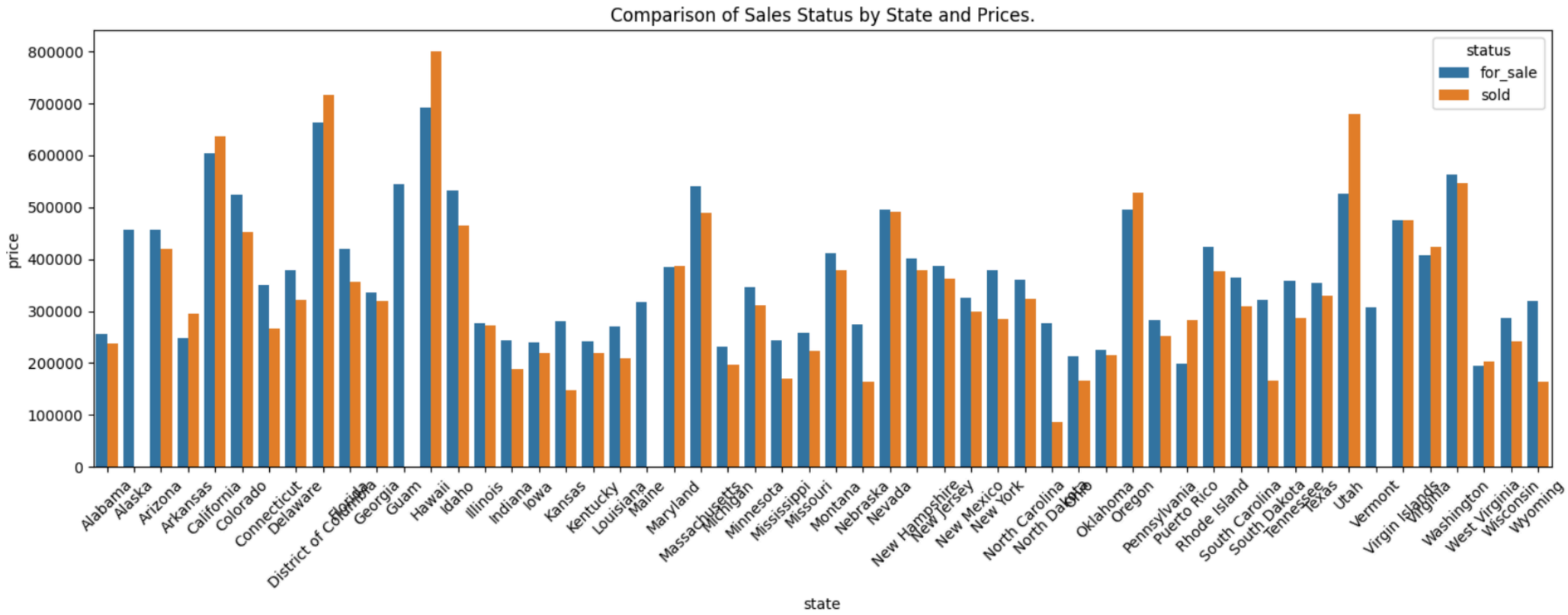

👉 **Market focus** for investors and agents: Boston, Springfield, and Worcester for volume

👉 But opportunities also exist in the **mid-tier cities.**

👉 A long tail of smaller markets implies **potential for niche strategies** (e.g., targeting underserved suburban or rural areas).

# 8. Grouped States, Price and Status of Sale Comparison

The price of status (sold and for_sale) for each State are shown in the graph below:




→ Grouping by state and sale status to calculate average prices.

# 8. Grouped States, Price and Status of Sale Comparison

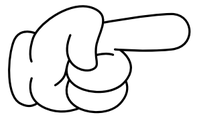
- Most U.S. states show a consistent gap between listed prices and actual sold prices -> possible bidding wars or seller pricing below market value to attract buyers.

- Large price gaps in California, Hawaii, D.C. -> strong market expectations vs. buyer reality.

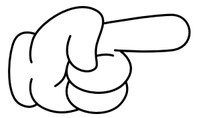
- In some states, ‘for_sale’ price is lower than sold -> underpricing or bidding wars.

# 8. Grouped States, Price and Status of Sale Comparison

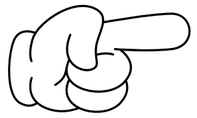
The comparison between listing and actual sale prices reveals market competitiveness and negotiation behavior:



**Higher sold than listed: Hot market, bidding activity.**



**Lower sold than listed: Buyer resistance or overvaluation.**



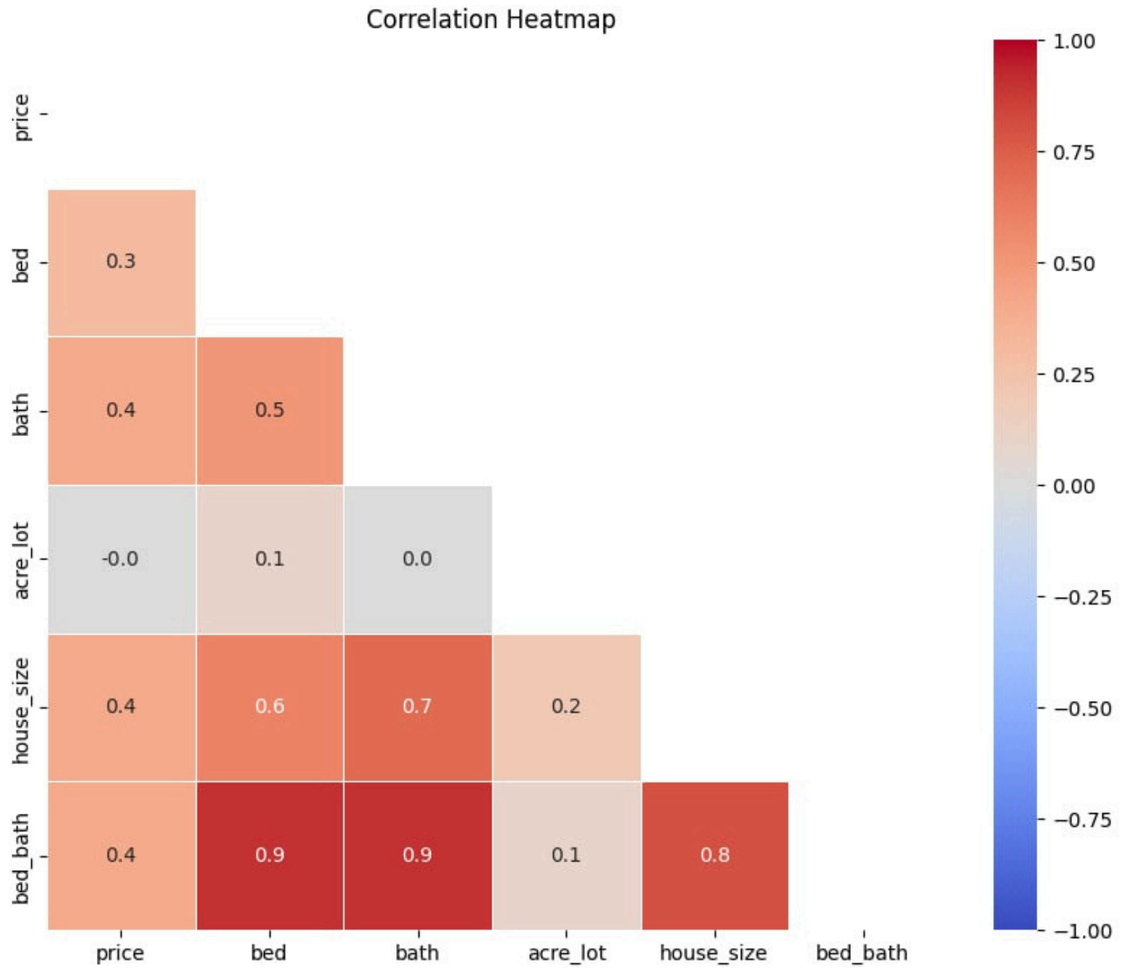
**Narrow gaps: Stable or well-priced markets.**

# 9. Data Correlation

## 9.1. Relationships between features:

- bath and house_size: 0.7
- bed and bath: 0.5

These features often **increase together**, reflecting **common housing designs** (more bedrooms & larger homes usually imply more bathrooms).



# 9. Data Correlation

## 9.2. Correlation with Price:

- **bath (0.4) and house_size (0.4)**

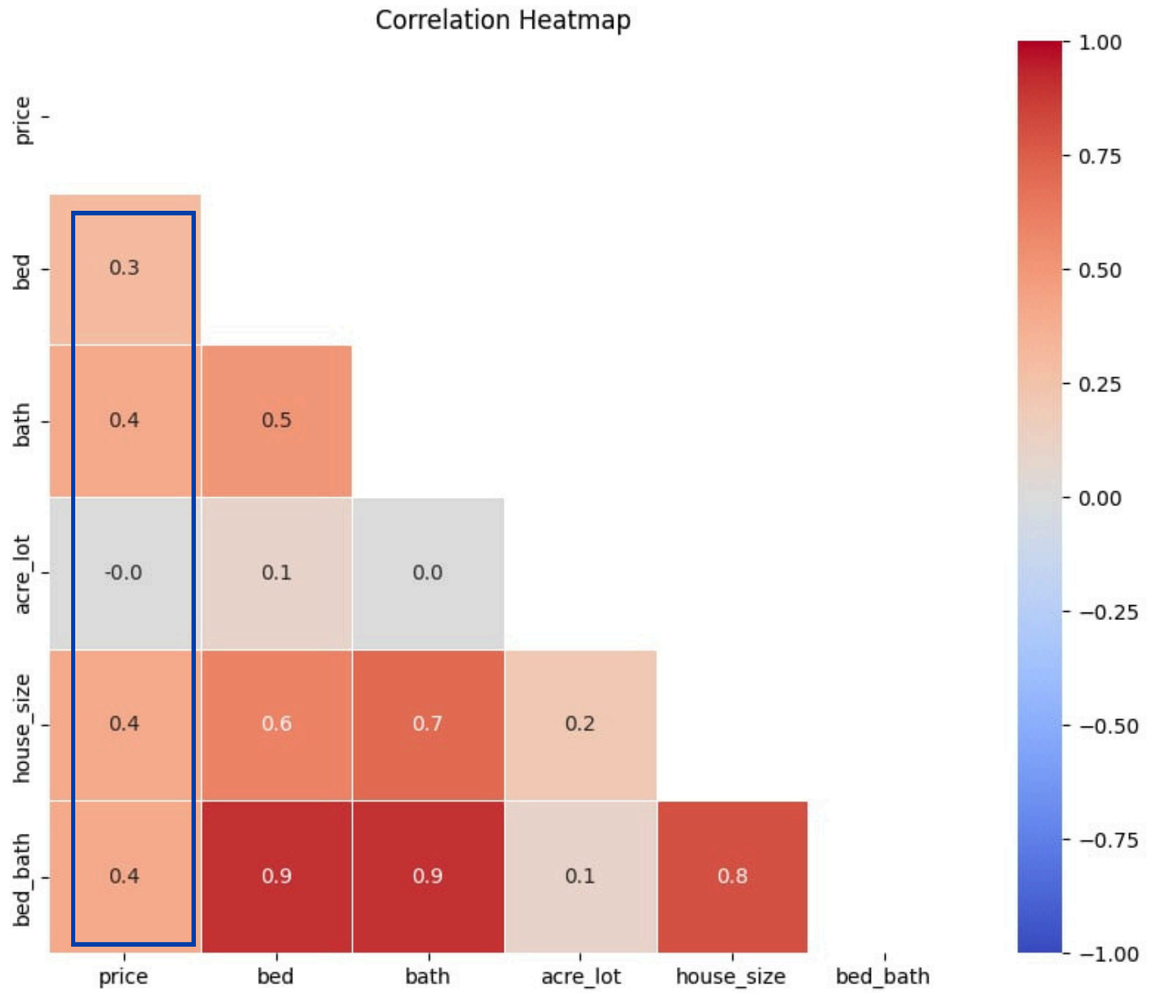
➤ A moderate **positive** correlation

➤ More bathrooms and larger houses tend to increase property value.

- **bed (0.3)**

➤ A weaker **positive** correlation

➤ The number of bedrooms impacts price, but less significantly.



# 9. Data Correlation

## 9.2. Correlation with Price:

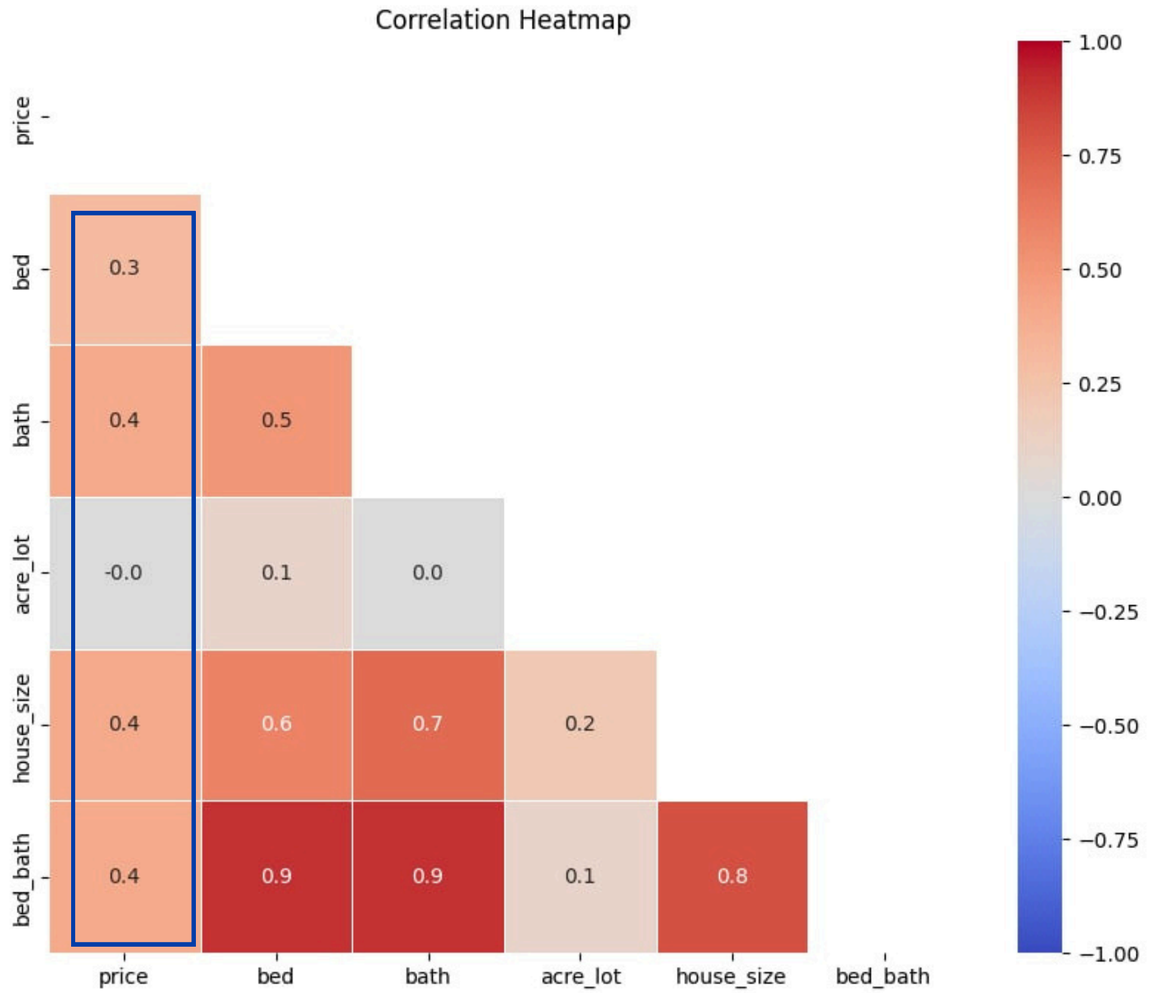
- **bed_bath (0.4)**

➤ Correlates **similarly** to bath and house_size

➤ It can be a useful composite feature

- **acre_lot has a near-zero correlation with price (0.0)**

➤ Lot size appears to have **minimal influence** on house value in this dataset.



# 9. Data Correlation

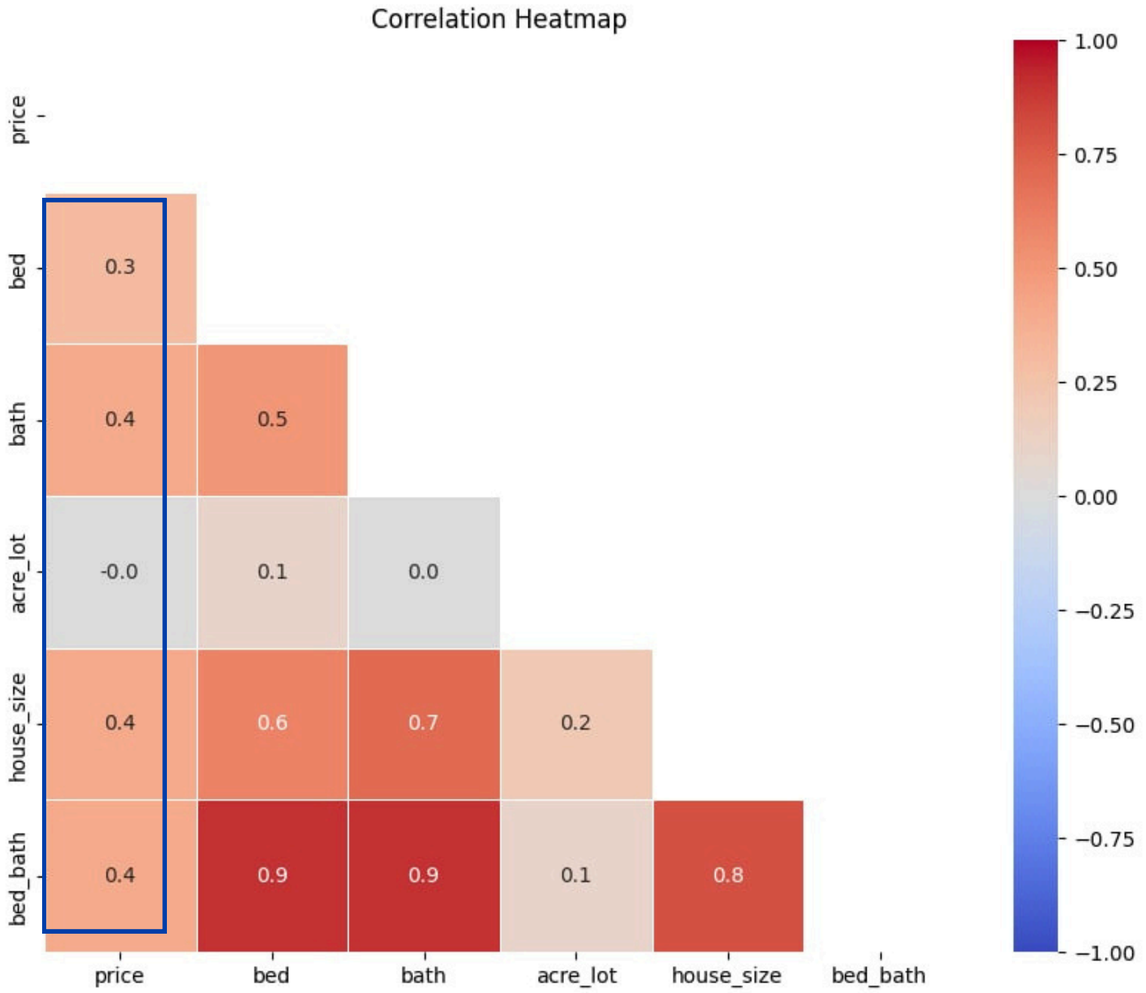
## 9.2. Correlation with Price:

Features like house_size, bath, and bed_bath have the **strongest correlations** with price

→ They are valuable for building prediction models.

acre_lot contributes **little explanatory power**

→ Location or other factors may outweigh land area in determining value.



# IV. Model Prediction

# Model Prediction

```
In [ ]:
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression, RandomForestRegressor
from pyspark.ml.evaluation import RegressionEvaluator
```

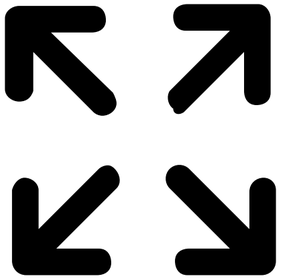
→ **Purpose**: Import essential machine learning components:

- **VectorAssembler**: Combine multiple feature columns into a single vector.
- **Linear Regression and Random Forest Regression**: Build regression models
- **RegressionEvaluator**: Evaluate the performance of regression models using metrics like RMSE or $R^2$.

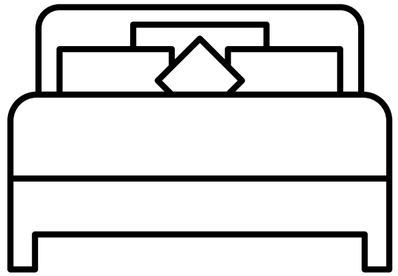
# Model Prediction

Developed machine learning models to predict housing prices based on features:

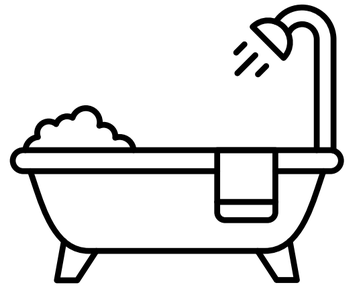
House Size



Lot Size



Number of Bedrooms



Number of Bathrooms

→ **Result**

**Linear Regression RMSE:** 190395.38 **$R^2$:** 0.23

**Random Forest RMSE:** 189370.70 **$R^2$**: 0.24

→ **Conclusion**

Random Forest **slightly outperforms** Linear Regression in terms of **accuracy**, with lower error rates and higher $R^2$ scores.

# Model Prediction

| price | lr_prediction | rf_prediction |
|---|---|---|
| 118000.0 | 194346.6300546726 | 241500.41325955465 |
| 185000.0 | 216244.18185334973 | 241500.41325955465 |
| 185000.0 | 216244.18185334973 | 241500.41325955465 |
| 158000.0 | 216244.18185334973 | 241500.41325955465 |
| 158000.0 | 216244.18185334973 | 241500.41325955465 |
| 165000.0 | 227395.71286193532 | 241500.41325955465 |
| 29900.0 | 230031.52928214642 | 241500.41325955465 |
| 29900.0 | 230031.52928214642 | 241500.41325955465 |
| 29900.0 | 230031.52928214642 | 241500.41325955465 |
| 29900.0 | 230031.52928214642 | 241500.41325955465 |

*Only showing top 10 rows*

# Model Prediction

The models provide a basic estimate of housing prices based on physical characteristics such as house size, number of rooms, and lot size. While they offer some insight into price trends, their predictive accuracy remains **limited.**

By comparing actual prices with predicted prices, we can identify cases of **mispricing**

—EX: properties listed far above their expected value, or vice versa.

# Model Prediction

→ While the models have limited predictive power *($R^2$ = 0.24 & 0.23)* in their current form, they still provide meaningful insights when analyzing historical housing data.

- Supporting homebuyers in making informed decisions,
- Helping investors identify undervalued opportunities
- Assisting lenders in evaluating collateral for loans.

# Model Prediction

However, the models still have **limitations.**

Given the model's limited accuracy, these results should be interpreted with caution and ideally combined with additional market context.

Future iterations could integrate **geospatial data, time-based trends, and qualitative features** to improve accuracy and applicability.

# V. Final Conclusion

# Conclusion

## Exploratory Analysis revealed:

1. **Strong price differences** across states (e.g., D.C., California, Hawaii have highest prices)
2. **Moderate correlations** between price and features (house size, number of bathrooms/bedrooms)
3. **Very weak correlation** between price and lot size → land area may not be a major determinant of property value

# Conclusion

## Predictive Modeling Results

Linear Regression **$R^2 \approx 0.23$**

Random Forest **$R^2 \approx 0.24$**

→ **Limited predictive power**

While basic physical features such as house size, lot size, and room count carry some predictive value - **not sufficient alone to fully explain housing prices.**

# Conclusion

## Future improvements

Focus on feature enrichment:

- **Geographic layers (city/neighborhood)**
- **Temporal trends**
- **Socio-economic and location-based context**

→ **Enhance model performance and real-world applicability.**

# THANK YOU FOR LISTENING