Data imputation involves filling in missing or incomplete data in a dataset. Given the dataset you provided, we can apply several strategies to handle missing values. Here are some common imputation techniques:

1. **Mean/Median/Mode Imputation**: For numerical columns, you can fill missing values with the mean, median, or mode of the column. For categorical columns, you can use the mode.
2. **Forward/Backward Fill**: This method fills missing values using the previous or next value in the column. It is useful for time-series data.
3. **Constant Value Imputation**: Fill missing values with a constant value, such as 0 or "Unknown".
4. **Predictive Imputation**: Use machine learning models to predict missing values based on other features.
5. **Drop Missing Values**: If a column or row has too many missing values, it might be best to drop it entirely.

Let's apply some of these techniques to your dataset:

Imputation Strategy

1. **Numerical Columns**:
   - For `price`, `bed`, `bath`, `acre_lot`, and `house_size`, use the median to fill missing values, as it is less sensitive to outliers than the mean.
   - For `prev_sold_date`, use a constant value like "Unknown" if the date is missing.
2. **Categorical Columns**:
   - For `brokered_by`, `status`, `street`, `city`, `state`, and `zip_code`, use the mode to fill missing values.
   - For `prev_sold_date`, use a constant value like "Unknown" if the date is missing.
3. **Special Cases**:
   - If `state` or `city` is missing, and the `zip_code` is available, you might use external data to fill these values based on the `zip_code`.
   - If `zip_code` is missing, and `city` and `state` are available, you might use external data to fill these values based on the `city` and `state`.

Example Imputation

Here's a simplified example of how you might perform imputation on this dataset using Python and pandas:

```python
import pandas as pd
import numpy as np

# Load the dataset
data = pd.read_csv('your_dataset.csv')

# Fill numerical columns with median
numerical_cols = ['price', 'bed', 'bath', 'acre_lot', 'house_size']
for col in numerical_cols:
    data[col] = data[col].replace(-1, np.nan)  # Replace -1 with NaN if used as a placeholder
    data[col].fillna(data[col].median(), inplace=True)

# Fill categorical columns with mode
categorical_cols = ['brokered_by', 'status', 'street', 'city', 'state', 'zip_code']
for col in categorical_cols:
    data[col].fillna(data[col].mode()[0], inplace=True)

# Fill 'prev_sold_date' with a constant value
data['prev_sold_date'].fillna('Unknown', inplace=True)

# Display the imputed dataset
print(data.head())
```

This code provides a basic framework for data imputation. Depending on the specific characteristics and requirements of your dataset, you might need to adjust the imputation strategy. Additionally, for more complex imputation tasks, you might consider using libraries like `sklearn` for predictive imputation or `fancyimpute` for advanced techniques like matrix factorization.