Data imputation is a process of replacing missing or null values in a dataset with statistically estimated values. Here's a step-by-step approach to perform data imputation on the given dataset:

Import necessary libraries

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

Load the dataset

```
# Load the dataset
data = {
    "brokered_by": [np.nan, 109978, 109987, 53173, np.nan, np.nan, 101497, np.nan, 53016, np.nan],
    "status": ["-", "-", "Unknown", "sold", "sold", "for_sale", "Unknown", "for_sale", "-", np.nan],
    "price": [np.nan, 760000, 490000, 350000, np.nan, np.nan, 129900, np.nan, 239000, np.nan],
    "bed": [np.nan, -1, np.nan, 3, -1, np.nan, 4, np.nan, 3, np.nan],
    "bath": [1, 2, 4, np.nan, np.nan, 2, np.nan, 2, np.nan, np.nan],
    "acre_lot": [-1, np.nan, -1, 0.9, 0.06, 0.83, 1.95, 0.83, 0.56, 1.04],
    "street": ["655 First St", np.nan, "229 Broadway Ave", "Unknown", np.nan, np.nan, "575 King Pl", np.nan, n
    "city": [np.nan, np.nan, "Corpus Christi", "Unknown", np.nan, np.nan, "Eastover", np.nan, "Oak Ridge", "Sp
    "state": ["Massachusetts", np.nan, "Texas", np.nan, "Massachusetts", np.nan, "South Carolina", np.nan, np.
    "zip_code": [-1, np.nan, np.nan, 25425, np.nan, 2631, 29044, np.nan, np.nan, -1],
    "house_size": [1015, np.nan, np.nan, np.nan, np.nan, 1368, 2128, np.nan, np.nan, 1876],
    "prev_sold_date": [np.nan, "2021-12-22", "2019-04-03", np.nan, np.nan, np.nan, "2021-09-20", np.nan, "2022
}

df = pd.DataFrame(data)
```

Data Preprocessing

```
# Convert categorical variables to numerical variables
df['status'] = pd.Categorical(df['status']).codes
df['street'] = pd.Categorical(df['street']).codes
df['city'] = pd.Categorical(df['city']).codes
df['state'] = pd.Categorical(df['state']).codes

# Convert date to datetime
df['prev_sold_date'] = pd.to_datetime(df['prev_sold_date'], errors='coerce')
```

Imputation

```
# Impute missing values using mean for numerical columns
numerical_cols = ['price', 'bed', 'bath', 'acre_lot', 'house_size']
imputer = SimpleImputer(strategy='mean')
df[numerical_cols] = imputer.fit_transform(df[numerical_cols])

# Impute missing values using most frequent for categorical columns
categorical_cols = ['status', 'street', 'city', 'state']
for col in categorical_cols:
    df[col] = df[col].fillna(df[col].mode()[0])

# Impute missing values using interpolation for datetime column
df['prev_sold_date'] = df['prev_sold_date'].fillna(pd.to_datetime('2022-01-01'))
```

Model-based Imputation

For more accurate imputation, we can use a model-based approach. We'll use a random forest regressor to predict the missing values.

```
# Split data into training and testing sets
X = df.drop('price', axis=1)
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a random forest regressor
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

# Predict missing values
missing_idx = df['price'].isnull()
df.loc[missing_idx, 'price'] = rf.predict(df.loc[missing_idx].drop('price', axis=1))
```

Note that this is a basic example of data imputation. The choice of imputation method depends on the specific characteristics of the data and the problem you're trying to solve.