Certainly! Data wrangling involves cleaning and transforming raw data into a more usable format. Here are some common data wrangling steps that can be applied to this dataset:

1. **Remove Rows with Missing Values**: Since there are rows with all `NaN` values, we can remove them.
2. **Convert Data Types**: Ensure that columns have the correct data types. For example, `price`, `acre_lot`, and `house_size` should be numeric, while `prev_sold_date` should be a date.
3. **Split `bedrooms_bathrooms`**: This column contains two pieces of information (bedrooms and bathrooms) separated by a comma. We can split this into two separate columns.
4. **Handle Missing Values**: For columns with missing values that are not entire rows, decide on a strategy to handle them, such as filling with a default value or using interpolation.
5. **Standardize Address Format**: Ensure that the address format is consistent, which might involve trimming whitespace or correcting any known formatting issues.
6. **Check for Duplicates**: Ensure there are no duplicate rows in the dataset.
7. **Normalize Data**: If necessary, normalize or scale numeric data for analysis.

Let's apply these steps to the dataset:

```
import pandas as pd
import numpy as np

# Load the dataset
data = {
    'brokered_by': [np.nan, 109978.0, 109987.0, 53173.0, np.nan, 96014.0, 101497.0, 10726.0, 53016.0, np.nan, 
    'status': [np.nan, 'sold', 'for_sale', 'sold', np.nan, 'sold', 'for_sale', 'for_sale', 'sold', np.nan, np.
    'price': [np.nan, 760000.0, 490000.0, 350000.0, np.nan, 370000.0, 129900.0, 425000.0, 239000.0, np.nan, np
    'acre_lot': [np.nan, 0.36, 0.16, 0.90, np.nan, 0.12, 1.95, 0.83, 0.56, np.nan, np.nan, 0.11, np.nan, 0.14,
    'zip_code': [np.nan, 92026.0, 78418.0, 25425.0, np.nan, 99207.0, 29044.0, 2631.0, 37830.0, np.nan, np.nan,
    'house_size': [np.nan, 1888.0, 2416.0, 3220.0, np.nan, 2084.0, 2128.0, 1368.0, 1724.0, np.nan, np.nan, 117
    'prev_sold_date': [np.nan, '2021-12-22', '2019-04-03', '2021-12-10', np.nan, '2022-04-05', '2021-09-20', '
    'address': [np.nan, '760 Madison Ln, Escondido, California', '229 Broadway Ave, Corpus Christi, Texas', '6
    'bedrooms_bathrooms': [np.nan, '3, 2', '3, 4', '3, 2', np.nan, '4, 2', '4, 2', '3, 2', '3, 2', np.nan, np.
}

df = pd.DataFrame(data)

# Remove rows with all NaN values
df.dropna(how='all', inplace=True)

# Convert data types
df['price'] = df['price'].astype(float)
df['acre_lot'] = df['acre_lot'].astype(float)
df['zip_code'] = df['zip_code'].astype(str)
df['house_size'] = df['house_size'].astype(float)
df['prev_sold_date'] = pd.to_datetime(df['prev_sold_date'], errors='coerce')

# Split 'bedrooms_bathrooms' into two separate columns
df[['bedrooms', 'bathrooms']] = df['bedrooms_bathrooms'].str.split(', ', expand=True)
df['bedrooms'] = df['bedrooms'].astype(float)
df['bathrooms'] = df['bathrooms'].astype(float)

# Drop the original 'bedrooms_bathrooms' column
df.drop(columns=['bedrooms_bathrooms'], inplace=True)

# Handle missing values (example: fill with median for numeric columns)
df.fillna(df.median(numeric_only=True), inplace=True)

# Check for duplicates and remove them
df.drop_duplicates(inplace=True)
```

```
# Display the cleaned DataFrame
print(df.head())
```

This code snippet performs basic data wrangling on the dataset. Depending on the specific requirements of your analysis, you might need to apply additional transformations or handle missing values differently.