Certainly! Data wrangling involves cleaning and transforming raw data into a more usable format. Here are some common steps we can take to clean and prepare this dataset:

1. **Remove Rows with Missing Values**: Since there are many rows with all NaN values, we can remove those.
2. **Convert Data Types**: Ensure that columns have the correct data types, such as converting `prev_sold_date` to a datetime format.
3. **Split `bedrooms_bathrooms`**: Split this column into two separate columns: `bedrooms` and `bathrooms`.
4. **Handle Missing Values**: Decide on a strategy for handling any remaining missing values, such as filling them with a default value or using interpolation.
5. **Standardize Address Format**: Ensure that addresses are consistently formatted, though this might already be the case.
6. **Remove Duplicates**: Check for and remove any duplicate rows.

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, np.nan, 101497.0, np.nan, 53016.0, np.nan, np
    'status': [np.nan, 'sold', 'for_sale', 'sold', np.nan, np.nan, 'for_sale', np.nan, 'sold', np.nan, np.nan,
    'price': [np.nan, 760000.0, 490000.0, 350000.0, np.nan, np.nan, 129900.0, np.nan, 239000.0, np.nan, np.nan
    'acre_lot': [np.nan, 0.36, 0.16, 0.9, np.nan, np.nan, 1.95, np.nan, 0.56, np.nan, np.nan, np.nan, np.nan, 
    'zip_code': [np.nan, 92026.0, 78418.0, 25425.0, np.nan, np.nan, 29044.0, np.nan, 37830.0, np.nan, np.nan, 
    'house_size': [np.nan, 1888.0, 2416.0, 3220.0, np.nan, np.nan, 2128.0, np.nan, 1724.0, np.nan, np.nan, np.
    'prev_sold_date': [np.nan, '2021-12-22', '2019-04-03', '2021-12-10', np.nan, np.nan, '2021-09-20', np.nan,
    '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, np.nan, '4, 2', np.nan, '3, 2', np.nan, np.
}

df = pd.DataFrame(data)

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

# Convert 'prev_sold_date' to datetime
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)

# Convert 'bedrooms' and 'bathrooms' to numeric
df['bedrooms'] = pd.to_numeric(df['bedrooms'], errors='coerce')
df['bathrooms'] = pd.to_numeric(df['bathrooms'], errors='coerce')

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

# Handle remaining missing values (e.g., fill with a default value or drop)
# Here, we'll fill numeric columns with 0 and categorical with 'Unknown'
df.fillna({'brokered_by': 0, 'status': 'Unknown', 'price': 0, 'acre_lot': 0, 'zip_code': 0, 'house_size': 0, '

# Remove duplicates
df.drop_duplicates(inplace=True)

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

This code will clean the dataset by removing rows with all NaN values, converting data types, splitting columns, handling missing values, and removing duplicates. Adjust the handling of missing values as needed based on the context of your analysis.