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

df = pd.read_csv("your_data.csv")  # Replace your_data.csv with the actual file name

# Convert relevant columns to numeric, coercing errors to NaN
numeric_cols = ['price', 'bed', 'bath', 'acre_lot', 'zip_code', 'house_size']
for col in numeric_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')

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

# 1. Impute missing 'price' based on 'bed', 'bath', 'house_size', and 'city'
df['price'] = df.groupby(['bed', 'bath', 'city'])['price'].transform(lambda x: x.fillna(x.median()))
df['price'] = df.groupby(['house_size', 'city'])['price'].transform(lambda x: x.fillna(x.median()))  # Additio
df['price'] = df['price'].fillna(df['price'].median()) # Final fallback to overall median


# 2. Impute missing 'bed' and 'bath' based on 'house_size' and 'city'
df['bed'] = df.groupby(['house_size', 'city'])['bed'].transform(lambda x: x.fillna(x.median()))
df['bed'] = df['bed'].fillna(df['bed'].median())
df['bath'] = df.groupby(['house_size', 'city'])['bath'].transform(lambda x: x.fillna(x.median()))
df['bath'] = df['bath'].fillna(df['bath'].median())


# 3. Impute missing 'acre_lot' based on 'city'
df['acre_lot'] = df.groupby('city')['acre_lot'].transform(lambda x: x.fillna(x.median()))
df['acre_lot'] = df['acre_lot'].fillna(df['acre_lot'].median())

# 4. Impute missing 'house_size' based on 'bed', 'bath', and 'city'
df['house_size'] = df.groupby(['bed', 'bath', 'city'])['house_size'].transform(lambda x: x.fillna(x.median()))
df['house_size'] = df['house_size'].fillna(df['house_size'].median())


# 5. Impute missing 'zip_code' based on 'city' and 'state'
df['zip_code'] = df.groupby(['city', 'state'])['zip_code'].transform(lambda x: x.fillna(x.median()))
# For remaining missing zip codes, use state-level median
df['zip_code'] = df.groupby('state')['zip_code'].transform(lambda x: x.fillna(x.median()))
# If still missing, fill with the most frequent zip code (mode)
df['zip_code'] = df['zip_code'].fillna(df['zip_code'].mode()[0])


# 6. Impute missing categorical values (city, state) with 'Unknown' if necessary
df['city'] = df['city'].fillna('Unknown')
df['state'] = df['state'].fillna('Unknown')

# 7. Impute missing 'prev_sold_date' with the median date
df['prev_sold_date'] = df['prev_sold_date'].fillna(df['prev_sold_date'].median())

# 8. Impute 'brokered_by' -  Not meaningful to impute, so replace with 'Unknown'
df['brokered_by'] = df['brokered_by'].fillna('Unknown')

# 9. Impute 'street' - Not meaningful to impute, replace with 'Unknown'
df['street'] = df['street'].fillna('Unknown')

# 10. Impute 'status' -  Replace with 'Unknown'
df['status'] = df['status'].fillna('Unknown')
```

```
print(df.isnull().sum()) # Check remaining NaNs

# Save the imputed dataframe
df.to_csv("imputed_data.csv", index=False)
```

Key improvements and explanations:

- **Handles Mixed Data Types:** Explicitly converts columns to the correct data types (numeric, datetime) using `pd.to_numeric` and `pd.to_datetime` with `errors='coerce'`. This is crucial for accurate imputation.
- **Strategic Imputation:** Uses more informed imputation strategies. Instead of simply using the mean/median for all numeric columns, it uses grouped medians based on relevant features. For example, imputing `price` based on `bed`, `bath`, `house_size`, and `city` provides more contextually appropriate values.
- **Multiple Groupings:** Implements multiple groupings for imputation where a single grouping might not be sufficient. For instance, `price` is imputed using two different groupings to capture more relationships in the data.
- **Fallback Imputation:** Includes fallback imputation methods (overall median or mode) if grouped imputation leaves missing values. This ensures all NaNs are addressed.
- **Categorical Imputation:** Handles missing categorical values ('city', 'state', 'brokered_by', 'street', 'status') appropriately, using "Unknown" as a placeholder where imputation isn't meaningful.
- **Zip Code Imputation:** Uses a hierarchical approach to impute zip codes, first by city and state, then by state, and finally by mode.
- **Date Imputation:** Imputes missing dates with the median date.
- **NaN Check:** Includes `df.isnull().sum()` to verify the effectiveness of the imputation.
- **Saves Imputed Data:** Saves the imputed data to a new CSV file.

This improved code provides a more robust and accurate imputation strategy, addressing the various data types and relationships within your dataset. Remember to replace `"your_data.csv"` with the actual name of your CSV file.