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

# Load the data (assuming it's in a CSV file named 'real_estate.csv')
df = pd.read_csv('real_estate.csv')

# 1. Handle Missing Values:
# Remove rows with all NaN values
df.dropna(how='all', inplace=True)

# For remaining NaNs, consider strategies based on the column:
# - brokered_by:  Impute with a placeholder like -1 (assuming it's an ID)
df['brokered_by'].fillna(-1, inplace=True)

# - status:  Categorical, so impute with 'unknown'
df['status'].fillna('unknown', inplace=True)

# - Numerical columns (price, acre_lot, zip_code, house_size):
#   * Impute with the median (more robust to outliers than mean)
for col in ['price', 'acre_lot', 'house_size']:
    df[col].fillna(df[col].median(), inplace=True)

# - zip_code:  Important for location-based analysis.  Don't impute.
#               Rows with missing zip codes might need to be excluded depending on your analysis.
# - prev_sold_date:  Impute with a placeholder date or handle based on your analysis needs.
df['prev_sold_date'].fillna('1900-01-01', inplace=True)  # Placeholder
df['prev_sold_date'] = pd.to_datetime(df['prev_sold_date']) # Convert to datetime

# - address:  Potentially valuable, but requires more complex handling (geocoding, etc.).
#              For now, leave as is.
# - bedrooms_bathrooms:  Extract bedrooms and bathrooms into separate columns.
df[['bedrooms', 'bathrooms']] = df['bedrooms_bathrooms'].str.split(',', expand=True)
df['bedrooms'] = pd.to_numeric(df['bedrooms'].str.strip(), errors='coerce')
df['bathrooms'] = pd.to_numeric(df['bathrooms'].str.strip(), errors='coerce')
df.drop('bedrooms_bathrooms', axis=1, inplace=True)  # Remove original column

# Impute missing bedrooms and bathrooms with median
df['bedrooms'].fillna(df['bedrooms'].median(), inplace=True)
df['bathrooms'].fillna(df['bathrooms'].median(), inplace=True)


# 2. Data Type Conversion:
# Ensure correct data types:
df['zip_code'] = df['zip_code'].astype(str).str.replace('.0', '', regex=False) # Remove .0 from zip codes
df['brokered_by'] = df['brokered_by'].astype(int)


# 3. Feature Engineering (Example):
# Calculate price per square foot
df['price_per_sqft'] = df['price'] / df['house_size']

# Calculate years since last sold (if needed)
df['years_since_sold'] = (pd.to_datetime('today') - df['prev_sold_date']).dt.days / 365.25


# 4. Outlier Detection and Handling (Example):
# Identify and handle outliers in 'price' and 'acre_lot' (using IQR method)
for col in ['price', 'acre_lot', 'house_size', 'price_per_sqft']:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]  # Remove outliers


# 5. Data Consistency Checks:
# Check for duplicates (and remove if necessary)
df.drop_duplicates(inplace=True)

# ... other consistency checks as needed ...

print(df.info())  # Check the cleaned data
# Save the cleaned data
df.to_csv('cleaned_real_estate.csv', index=False)
```

Key Improvements and Explanations:

- **More Robust NaN Handling:** Using median imputation for numerical features is generally better than mean when outliers might be present. A placeholder (-1) is used for `brokered_by` assuming it's an ID, and "unknown" for the categorical `status`.
- **Zip Code Handling:** Zip codes are treated as strings to preserve leading zeros and avoid misinterpretation as numbers. The `.0` is removed. No imputation is done as it's crucial for location analysis.
- **Date Handling:** `prev_sold_date` is converted to datetime objects for proper date calculations. A placeholder date is used for missing values. You can modify this based on your analysis needs (e.g., impute with the median date).
- **Bedrooms/Bathrooms Extraction:** The `bedrooms_bathrooms` column is split into separate numerical `bedrooms` and `bathrooms` columns.
- **Feature Engineering:** `price_per_sqft` and `years_since_sold` are calculated as examples. You can add more features relevant to your analysis.
- **Outlier Handling:** The IQR method is used to detect and remove outliers in key numerical columns. This helps improve model performance and the reliability of your analysis.
- **Data Consistency:** Duplicate rows are removed.
- **Clearer Code and Comments:** The code is better structured and commented for readability and maintainability.
- **Type Conversions:** Ensures data is in the correct format (e.g., zip code as string, brokered_by as integer).

This improved code provides a more comprehensive and robust data wrangling process, addressing missing values, data types, feature engineering, outlier handling, and data consistency. Remember to adapt the outlier handling and imputation strategies based on the specific requirements of your analysis. The example outlier removal might be too aggressive depending on your data and goals. Consider visualizing the data to make informed decisions about outlier handling.