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

# Load the dataset (assuming it's in a CSV file named 'real_estate.csv')
try:
    df = pd.read_csv('real_estate.csv')
except FileNotFoundError:
    # If the file isn't found, create a DataFrame from the provided data
    data = {
        'brokered_by': [np.nan, 109978.0, 109987.0, 53173.0, np.nan, np.nan, 101497.0, np.nan, 53016.0, np.nan
        # ... (rest of the data)
    }
    df = pd.DataFrame(data)


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

# 2. Data Type Conversion:
# Convert 'bedrooms_bathrooms' to separate columns
df[['bedrooms', 'bathrooms']] = df['bedrooms_bathrooms'].str.split(', ', expand=True).astype(float)
df.drop(columns=['bedrooms_bathrooms'], inplace=True)

# Convert 'prev_sold_date' to datetime
df['prev_sold_date'] = pd.to_datetime(df['prev_sold_date'])

# Convert 'zip_code' to string (to preserve leading zeros if needed)
df['zip_code'] = df['zip_code'].astype(str)

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

# 4. Handling Outliers (Example - requires domain knowledge):
# For 'acre_lot', if values are significantly larger than typical residential lots,
# they might be errors or represent different property types.  Investigate and decide
# whether to cap, remove, or create a new category.
# (Similar logic can be applied to other numerical features like 'price', 'house_size')


# 5. Data Cleaning (Example):
# Remove any leading/trailing whitespace from string columns
for col in ['address', 'status']:  # Add other string columns as needed
    df[col] = df[col].str.strip() if df[col].dtype == 'object' else df[col]


print(df.head())
print(df.info())  # Check data types and non-null counts after cleaning
```

**Explanation and Key Improvements:**

- **Handles Missing Data:** Rows with entirely missing values are removed, as they don't provide useful information. More sophisticated imputation techniques could be used if needed, but deletion is often a good starting point.
- **Correct Data Types:** The `bedrooms_bathrooms` column is split into separate numerical `bedrooms` and `bathrooms` columns. `prev_sold_date` is converted to the correct datetime format. `zip_code` is converted to a string to avoid potential issues with leading zeros.
- **Feature Engineering:** A new `price_per_sqft` feature is created, which can be valuable for analysis.
- **Outlier Handling (Guidance):** The code includes a comment explaining the importance of outlier detection and handling, using `acre_lot` as an example. This requires domain knowledge to determine appropriate thresholds.
- **Data Cleaning:** Leading/trailing whitespace is removed from string columns to ensure data consistency.

- **Error Handling:** The code now includes a `try-except` block to handle the case where the 'real_estate.csv' file is not found. It creates a DataFrame from the provided data snippet in that scenario.
- **Informative Output:** `df.info()` is included to display the cleaned DataFrame's structure, data types, and non-null counts, which is crucial for verifying the wrangling steps.

This improved code provides a more robust and complete data wrangling process, addressing several common issues and preparing the data for further analysis or modeling. Remember to adapt the outlier handling and feature engineering steps based on your specific analytical goals and domain knowledge.