Nashville_housing_data_2013_2016

1. Loading and Cleaning the Dataset

```
df = pd.read_csv("Nashville_housing_data_2013_2016.csv")
```

Loads the CSV file into a pandas DataFrame called `df`.

This is your dataset you want to clean and analyze.

```
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_').str.replace('#', 'number')
```

Cleans column names:

`str.strip()`: removes extra spaces from column names.

`str.lower()`: makes all column names lowercase.

`str.replace(' ', '_')`: replaces spaces with underscores to avoid errors in coding.

`str.replace('#', 'number')`: replaces `#` with `number` to avoid syntax issues.

```
df['sale_date'] = pd.to_datetime(df['sale_date'], errors='coerce')
```

Converts the `sale_date` column to datetime format.

`errors='coerce'` converts invalid dates into `NaT` (null), preventing crashes.

---

2. **Handling Missing Data**

```
df = df.dropna(thresh=len(df)*0.4, axis=1)
```

Drops columns that have more than 60% missing values.

`thresh=len(df)*0.4` means keep only those columns that have at least 40% non-null data.

`axis=1` means this operation is done on columns (not rows).

```
df.fillna(df.median(numeric_only=True), inplace=True)
```

Fills missing values in numeric columns with their **median**.

`numeric_only=True` avoids non-numeric columns.

`inplace=True` applies changes directly to `df`.

```
df.fillna(df.mode().iloc[0], inplace=True)
```

Fills missing values in non-numeric (categorical) columns using the **most frequent value** (mode).

`df.mode().iloc[0]` gives the most common value for each column.

```
df.drop_duplicates(inplace=True)
```

Removes any duplicate rows in the dataset to avoid repetition in analysis.

```
df['year'] = df['sale_date'].dt.year
```

Creates a new column `year` by extracting year from the `sale_date`.

---

3. **Boxplot: Sale Price by Land Use**

```
sns.boxplot(x='land_use', y='sale_price', data=df, palette="Set2")
```

Creates a **boxplot** where:

X-axis = land use (e.g., Single Family, Condo)

Y-axis = sale price

`palette="Set2"` adds pleasant multicolors.

```
plt.xticks(rotation=45)
```

Rotates the x-axis labels by 45 degrees for better readability.

---

4. **Line Chart: Median Sale Price by Year**

```
median_prices_by_year = df.groupby('year')['sale_price'].median()
```

Groups the data by `year` and calculates the **median** `sale_price` for each year.

```
sns.lineplot(x=median_prices_by_year.index, y=median_prices_by_year.values, marker='o', color='orange')
```

Plots a **line chart**

X = year

Y = median price

`marker='o'` puts dots on each data point.

`color='orange'` makes the line visually distinct.

---

5. **Correlation Heatmap**

```
sns.heatmap(df.select_dtypes(include='number').corr(), annot=True, cmap='coolwarm', fmt='.2f')
```

Builds a **heatmap of correlations** between all numeric columns.

`annot=True`: shows the correlation values inside each cell.

`cmap='coolwarm'`: uses red/blue gradient for easier pattern recognition.

`fmt='.2f'`: formats the numbers to 2 decimal places.

---

6. **Countplot: Number of Properties by Bedrooms**_

```
sns.countplot(data=df, x='bedrooms', palette='viridis')
```

A **bar chart** showing how many properties have 1, 2, 3... bedrooms.

`palette='viridis'`: applies a stylish color gradient.

---

7. **Histogram: Distribution of Finished Area**_

```
sns.histplot(df['finished_area'], bins=40, kde=True, color='mediumvioletred')
```

A **histogram** showing how many properties fall into area ranges (sq ft).

`bins=40`: divides data into 40 groups.

`kde=True`: adds a smooth line on top showing density.

`color='mediumvioletred'`: gives it a vibrant red-purple color.

Plot Display Settings_

```
plt.title('Title Here')
```

Adds a title to the plot.

_

```
plt.xlabel('X-label') / plt.ylabel('Y-label')
```

Adds labels to X and Y axes.

_

```
plt.tight_layout()
```

Prevents labels from getting cut off.

_

```
plt.show()
```

Displays the plot in your output window.