Expected Output: Zipped file of both

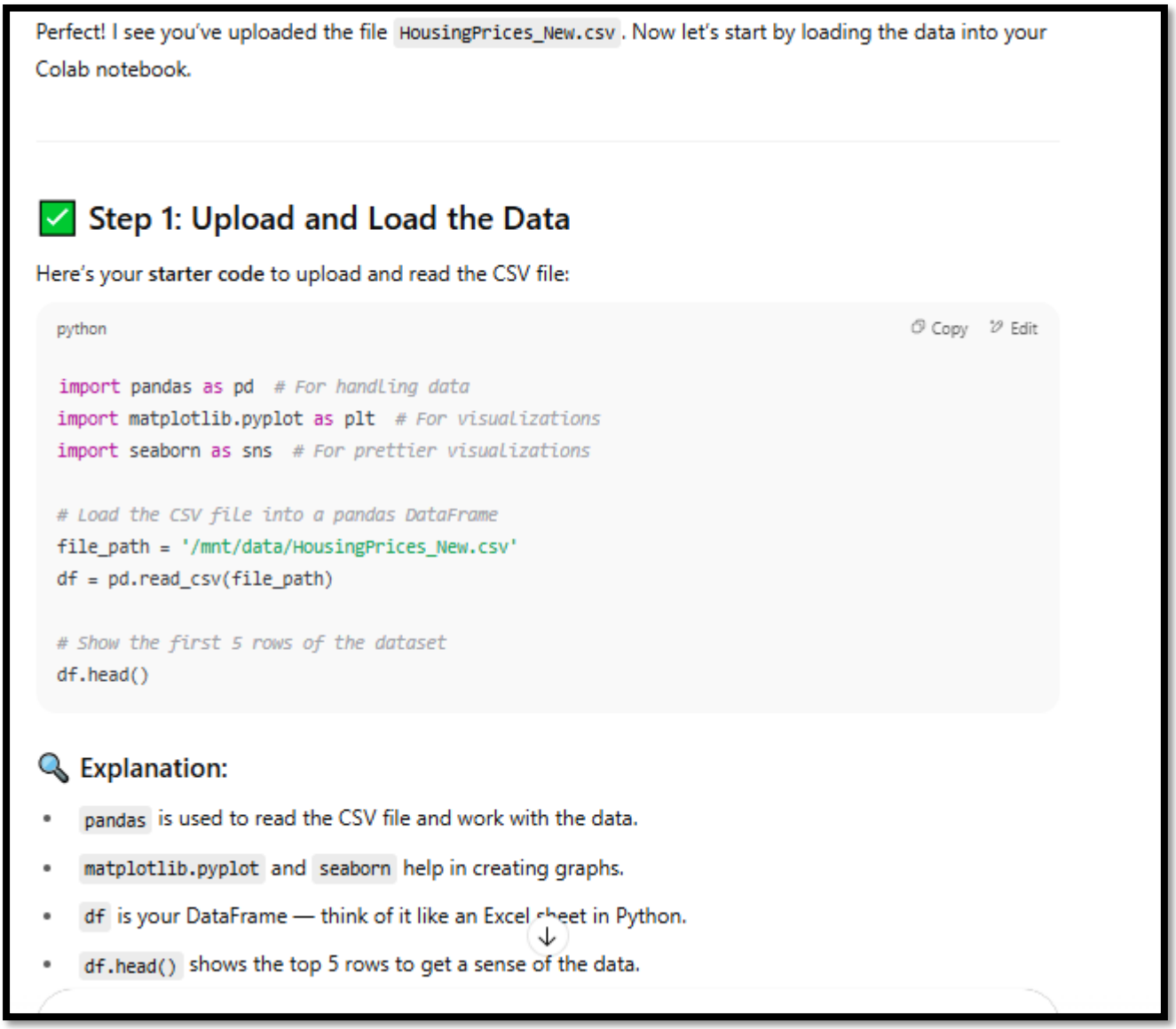
I. **PDF file with responses to above 5 questions. Each response should have**
   a) **Prompt**
   b) **Sanpshots of the output.**

II. **Google Colab .ipynb file for each code**

1. Load the data into ChatGPT.

Prompt:

You are an expert data analyst. I am going to ask you some questions and you are going to give me 2 outcomes of each 1. A data visualization graph for the same. 2. A python code that I can execute in GoogleColab I am new to python so give elaborate explanation of python code and if there is any specific thing that needs to be done to make the code work, Since I am using Google Colab, the file is in my local drive, lets start with the data upload code and do not repeat the code again as it is in the same google colab I am attaching the data file on which the data analysis has to be performed.



Perfect! I see you've uploaded the file `HousingPrices_New.csv`. Now let's start by loading the data into your Colab notebook.

✅ Step 1: Upload and Load the Data

Here's your **starter code** to upload and read the CSV file:

python — Copy — Edit

```
import pandas as pd  # For handling data
import matplotlib.pyplot as plt  # For visualizations
import seaborn as sns  # For prettier visualizations

# Load the CSV file into a pandas DataFrame
file_path = '/mnt/data/HousingPrices_New.csv'
df = pd.read_csv(file_path)

# Show the first 5 rows of the dataset
df.head()
```

🔍 Explanation:

- `pandas` is used to read the CSV file and work with the data.
- `matplotlib.pyplot` and `seaborn` help in creating graphs.
- `df` is your DataFrame — think of it like an Excel sheet in Python.
- `df.head()` shows the top 5 rows to get a sense of the data.

Now that the data is loaded, **please ask your first analysis question**, and I'll provide:

1. A data visualization graph.
2. Executable Python code with beginner-friendly explanation.

Ready when you are!

```
# Load the CSV file into a pandas DataFrame
file_path = '/mnt/data/HousingPrices_New.csv'
df = pd.read_csv(file_path)

this is giving me error. give me another code where I can upload the file in the code itself
```

No problem! Since you want to upload the file manually **from your local drive through the notebook**, here's the **correct method for Google Colab** using the `files.upload()` function.

✅ Correct Way to Upload Local Files in Google Colab

Replace your earlier code with this:

python Copy Edit

```python
# Step 1: Upload the file from your local computer
from google.colab import files
uploaded = files.upload()
```

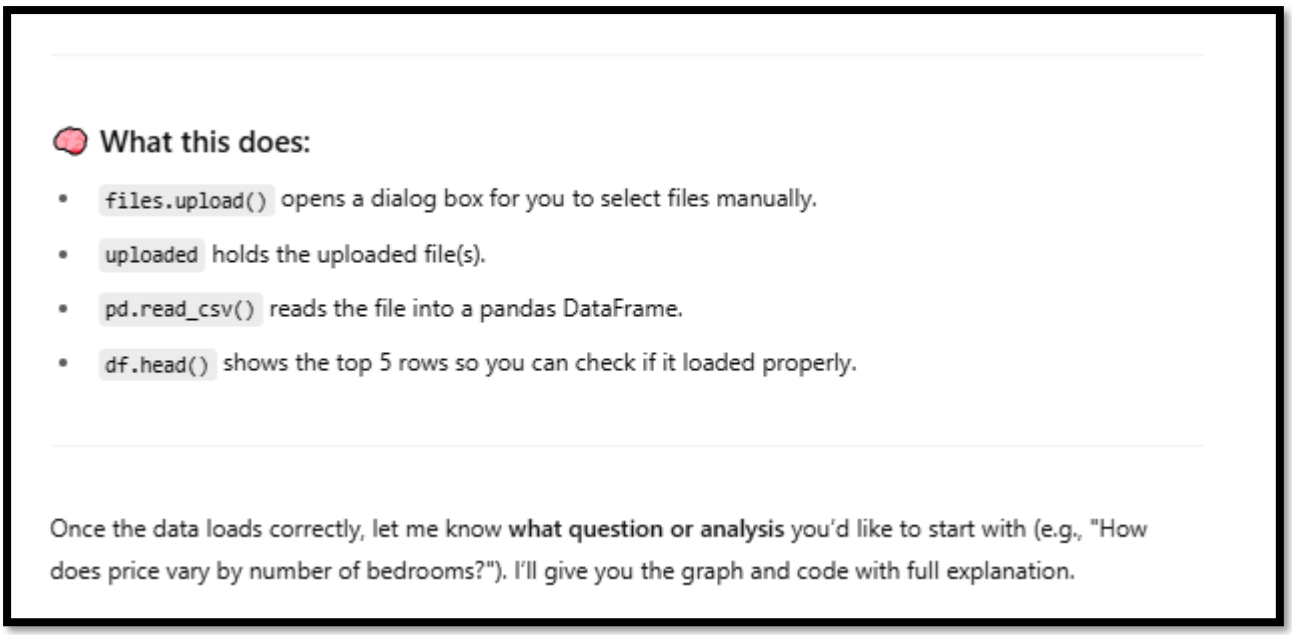
Once you run this cell, Colab will open a file picker. Select your CSV file (`HousingPrices_New.csv`).

python Copy Edit

```python
# Step 2: Read the uploaded file using pandas
import pandas as pd

# Assuming the file is named 'HousingPrices_New.csv'
df = pd.read_csv('HousingPrices_New.csv')

# Show the first 5 rows
df.head()
```

🧠 What this does:

- `files.upload()` opens a dialog box for you to select files manually.
- `uploaded` holds the uploaded file(s).



What this does:

- `files.upload()` opens a dialog box for you to select files manually.
- `uploaded` holds the uploaded file(s).
- `pd.read_csv()` reads the file into a pandas DataFrame.
- `df.head()` shows the top 5 rows so you can check if it loaded properly.

Once the data loads correctly, let me know **what question or analysis** you'd like to start with (e.g., "How does price vary by number of bedrooms?"). I'll give you the graph and code with full explanation.

Codes

```
import pandas as pd  # For handling data
import matplotlib.pyplot as plt  # For visualizations
import seaborn as sns  # For prettier visualizations

from google.colab import files
uploaded = files.upload()

# Step 2: Read the uploaded file using pandas
import pandas as pd

# Assuming the file is named 'HousingPrices_New.csv'
df = pd.read_csv('HousingPrices_New.csv')

# Show the first 5 rows
df.head()
```

2. Basic Statistics of 5 important columns (Mean, Median, Mode, Standard Deviation, Minimum, Max)

Prompt:

pick 5 most important columns from the data apart from sale price. mention the column names exactly as it is mentioned in the files. No changes.

Codes:

```
# Show all column names
df.columns.tolist()
```

🏆 5 Most Important Columns (excluding `Sale Price`):

Here are the **5 most relevant predictors** of housing price based on domain knowledge, using **exact column names** from your file:

1. **Flat Area (in Sqft)** – Total usable floor area
2. **No of Bedrooms** – Number of bedrooms
3. **No of Bathrooms** – Number of bathrooms
4. **Overall Grade** – Overall rating of the house
5. **Basement Area (in Sqft)** – Area of the basement

These are likely to have a strong impact on the `Sale Price`.

Let me know what you'd like to do next — for example:

- Correlation heatmap?
- Scatter plot of `Flat Area (in Sqft)` vs `Sale Price`?
- Analysis of `Overall Grade` vs `Sale Price`?

I'll provide the code and the graph for whichever question you pick.

Prompt for Statistics for Flat Area (in sqft)

Now lets do some statistics on Google Colab. give me the python code for the following. (Mean, Median, Mode, Standard Deviation, Minimum, Max) for Flat Area (in Sqft) Do not give the code of uploading the file again and again. Its done once and need not be done again.

Codes

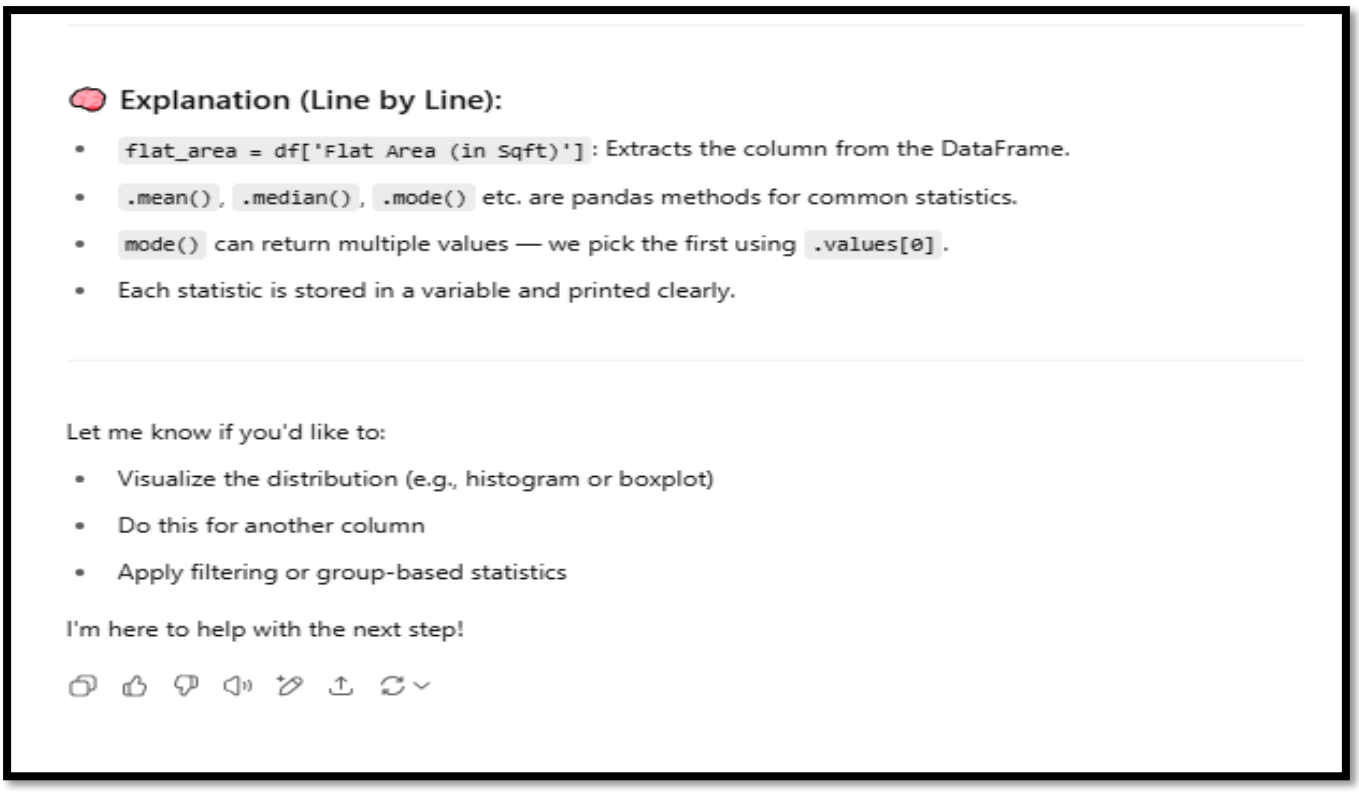
```
# Select the column
flat_area = df['Flat Area (in Sqft)']

# Calculate statistics
mean_value = flat_area.mean()
median_value = flat_area.median()
mode_value = flat_area.mode().values[0]  # Taking the first mode if multiple exist
std_dev = flat_area.std()
minimum = flat_area.min()
maximum = flat_area.max()

# Print results
print("Statistics for 'Flat Area (in Sqft)':")
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Standard Deviation: {std_dev}")
print(f"Minimum: {minimum}")
print(f"Maximum: {maximum}")
```

Perfect! Since your data is already loaded into the DataFrame `df`, here's the Python code to calculate the **Mean, Median, Mode, Standard Deviation, Minimum, and Maximum** for the column `Flat Area (in Sqft)`.

✅ Python Code for Descriptive Statistics

python — Copy — Edit

```python
# Select the column
flat_area = df['Flat Area (in Sqft)']

# Calculate statistics
mean_value = flat_area.mean()
median_value = flat_area.median()
mode_value = flat_area.mode().values[0]  # Taking the first mode if multiple exist
std_dev = flat_area.std()
minimum = flat_area.min()
maximum = flat_area.max()

# Print results
print("Statistics for 'Flat Area (in Sqft)':")
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Standard Deviation: {std_dev}")
print(f"Minimum: {minimum}")
print(f"Maximum: {maximum}")
```



🧠 Explanation (Line by Line):

- `flat_area = df['Flat Area (in Sqft)']` : Extracts the column from the DataFrame.
- `.mean()`, `.median()`, `.mode()` etc. are pandas methods for common statistics.
- `mode()` can return multiple values — we pick the first using `.values[0]`.
- Each statistic is stored in a variable and printed clearly.

Let me know if you'd like to:

- Visualize the distribution (e.g., histogram or boxplot)
- Do this for another column
- Apply filtering or group-based statistics

I'm here to help with the next step!

Prompt for No of Bedrooms

Now lets do some statistics on Google Colab. give me the python code for the following. (Mean, Median, Mode, Standard Deviation, Minimum, Max) for No of Bedrooms Do not give the code of uploading the file again and again. Its done once and need not be done again.

Codes

```
# Select the column
bedrooms = df['No of Bedrooms']

# Calculate statistics
mean_value = bedrooms.mean()
median_value = bedrooms.median()
mode_value = bedrooms.mode().values[0]  # If multiple modes, take the first
std_dev = bedrooms.std()
minimum = bedrooms.min()
maximum = bedrooms.max()

# Print results
print("Statistics for 'No of Bedrooms':")
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Standard Deviation: {std_dev}")
print(f"Minimum: {minimum}")
print(f"Maximum: {maximum}")
```

Absolutely! Here's the **Python code** to calculate the following statistics for the column `No of Bedrooms`:

- Mean
- Median
- Mode
- Standard Deviation
- Minimum
- Maximum

✅ **Python Code for `No of Bedrooms`**

python — Copy — Edit

```
# Select the column
bedrooms = df['No of Bedrooms']

# Calculate statistics
mean_value = bedrooms.mean()
median_value = bedrooms.median()
mode_value = bedrooms.mode().values[0]  # If multiple modes, take the first
std_dev = bedrooms.std()
minimum = bedrooms.min()
maximum = bedrooms.max()

# Print results
print("Statistics for 'No of Bedrooms':")
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Standard Deviation: {std_dev}")
print(f"Minimum: {minimum}")
print(f"Maximum: {maximum}")
```

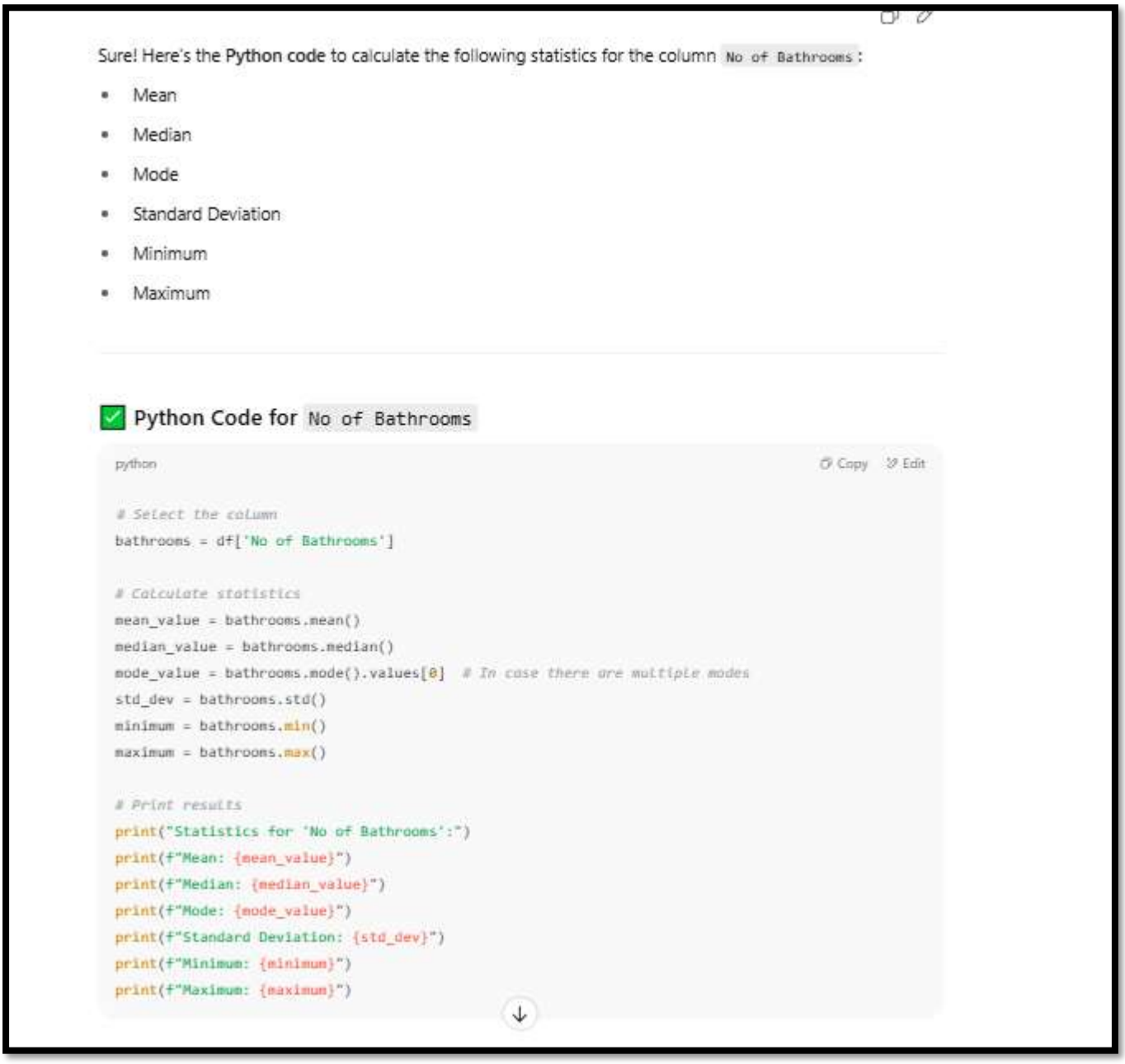
Prompt for No of Bathrooms

Now lets do some statistics on Google Colab. give me the python code for the following. (Mean, Median, Mode, Standard Deviation, Minimum, Max) for No of Bathrooms Do not give the code of uploading the file again and again. Its done once and need not be done again.

Codes

```
# Select the column
bathrooms = df['No of Bathrooms']

# Calculate statistics
mean_value = bathrooms.mean()
median_value = bathrooms.median()
mode_value = bathrooms.mode().values[0]  # In case there are multiple modes
std_dev = bathrooms.std()
minimum = bathrooms.min()
maximum = bathrooms.max()

# Print results
```

```
print("Statistics for 'No of Bathrooms':")
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Standard Deviation: {std_dev}")
print(f"Minimum: {minimum}")
print(f"Maximum: {maximum}")
```



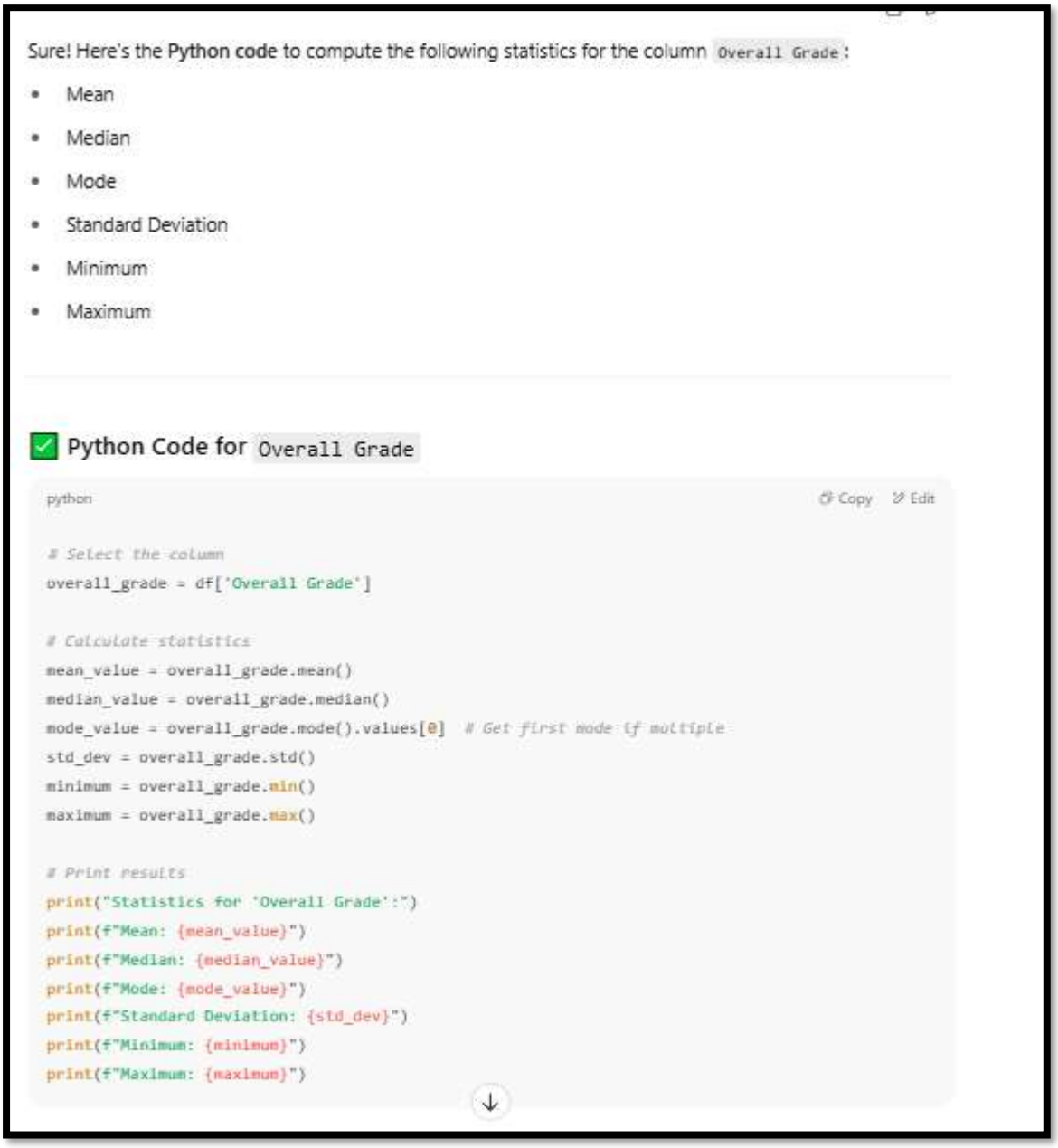
Prompt for Overall Grade:

Now lets do some statistics on Google Colab. give me the python code for the following. (Mean, Median, Mode, Standard Deviation, Minimum, Max) for Overall Grade Do not give the code of uploading the file again and again. Its done once and need not be done again.

Codes

```
# Select the column
overall_grade = df['Overall Grade']
```

```
# Calculate statistics
mean_value = overall_grade.mean()
median_value = overall_grade.median()
mode_value = overall_grade.mode().values[0]  # Get first mode if multiple
std_dev = overall_grade.std()
minimum = overall_grade.min()
maximum = overall_grade.max()

# Print results
print("Statistics for 'Overall Grade':")
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Standard Deviation: {std_dev}")
print(f"Minimum: {minimum}")
print(f"Maximum: {maximum}")
```



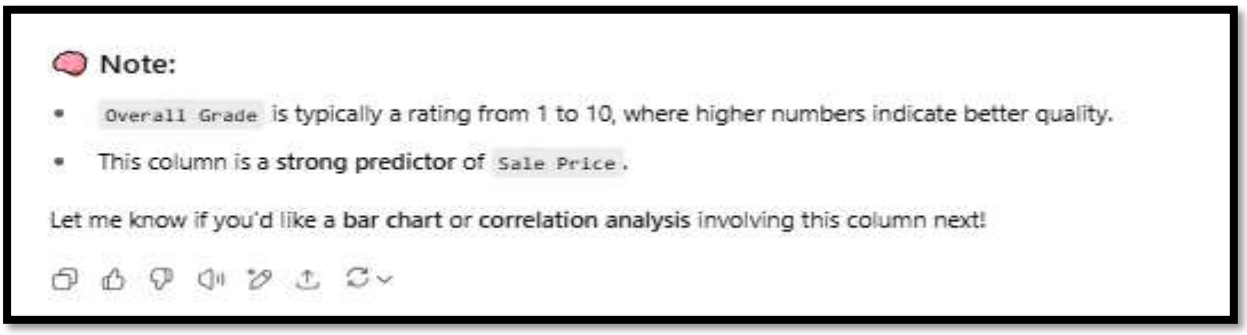
Sure! Here's the **Python code** to compute the following statistics for the column `Overall Grade`:

- Mean
- Median
- Mode
- Standard Deviation
- Minimum
- Maximum

✅ **Python Code for `Overall Grade`**

python — Copy — Edit

```python
# Select the column
overall_grade = df['Overall Grade']

# Calculate statistics
mean_value = overall_grade.mean()
median_value = overall_grade.median()
mode_value = overall_grade.mode().values[0]  # Get first mode if multiple
std_dev = overall_grade.std()
minimum = overall_grade.min()
maximum = overall_grade.max()

# Print results
print("Statistics for 'Overall Grade':")
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Standard Deviation: {std_dev}")
print(f"Minimum: {minimum}")
print(f"Maximum: {maximum}")
```



Prompt for Basement Area (in Sqft)

Now lets do some statistics on Google Colab. give me the python code for the following.
(Mean, Median, Mode, Standard Deviation, Minimum, Max) for Basement Area (in Sqft)
Do not give the code of uploading the file again and again. Its done once and need not be done again.

Codes

```
# Select the column
basement_area = df['Basement Area (in Sqft)']

# Calculate statistics
mean_value = basement_area.mean()
median_value = basement_area.median()
mode_value = basement_area.mode().values[0]  # Get first mode if multiple
std_dev = basement_area.std()
minimum = basement_area.min()
maximum = basement_area.max()

# Print results
print("Statistics for 'Basement Area (in Sqft)':")
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Standard Deviation: {std_dev}")
print(f"Minimum: {minimum}")
print(f"Maximum: {maximum}")
```

Absolutely! Here's the **Python code** to calculate the following statistics for the column `Basement Area (in Sqft)`:

- Mean
- Median
- Mode
- Standard Deviation
- Minimum
- Maximum

✅ Python Code for `Basement Area (in Sqft)`

python Copy Edit

```python
# Select the column
basement_area = df['Basement Area (in Sqft)']

# Calculate statistics
mean_value = basement_area.mean()
median_value = basement_area.median()
mode_value = basement_area.mode().values[0]  # Get first mode if multiple
std_dev = basement_area.std()
minimum = basement_area.min()
maximum = basement_area.max()

# Print results
print("Statistics for 'Basement Area (in Sqft)':")
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Standard Deviation: {std_dev}")
print(f"Minimum: {minimum}")
print(f"Maximum: {maximum}")
```

3. Check for missing values and outliers.

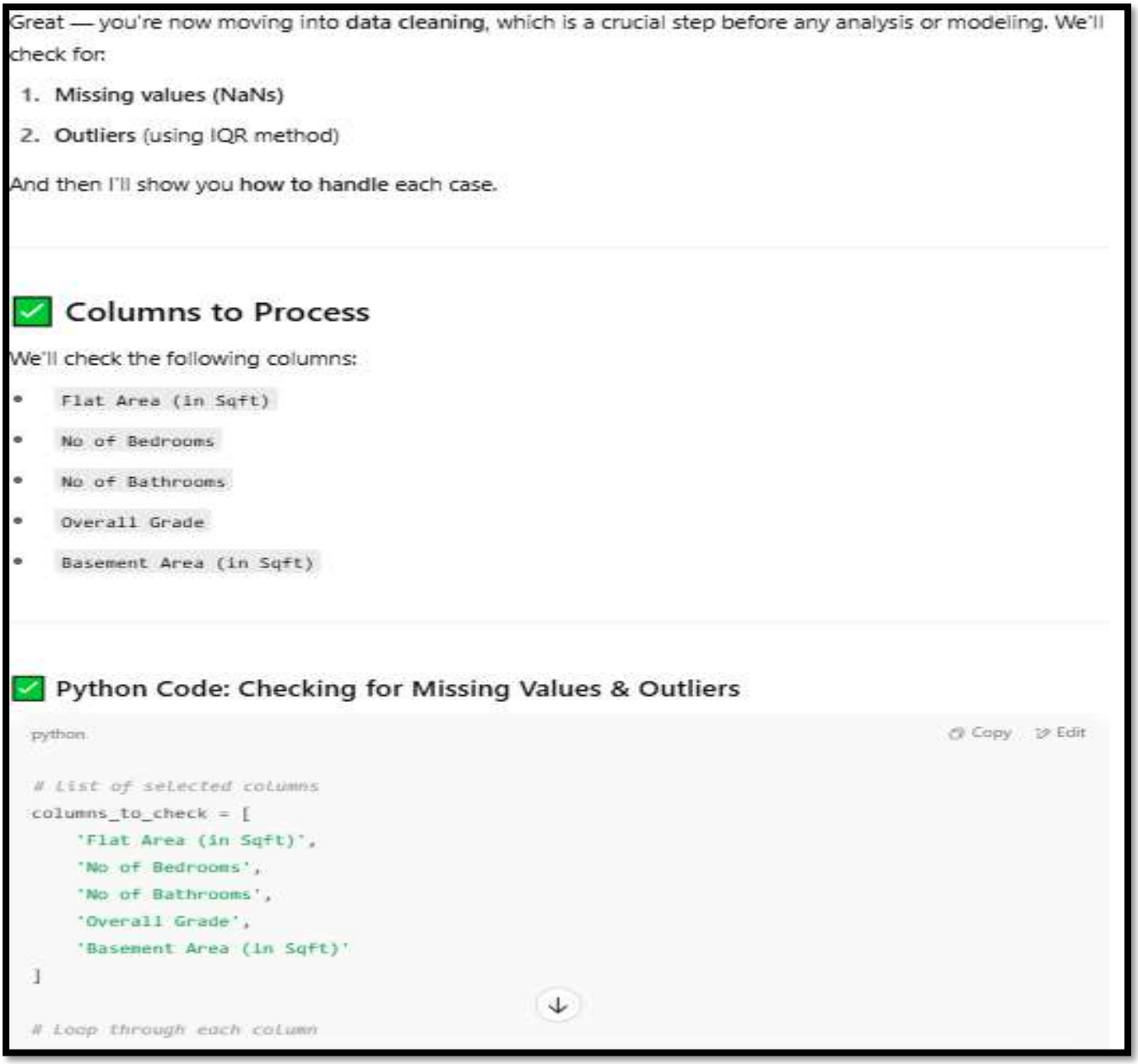
Prompt:

Now check for missing values or outliers for each of these. And give me a python code on how to handle them Flat Area (in Sqft) – Total usable floor area No of Bedrooms – Number of bedrooms No of Bathrooms – Number of bathrooms Overall Grade – Overall rating of the house Basement Area (in Sqft) – Area of the basement

Codes

```
# List of selected columns
columns_to_check = [
    'Flat Area (in Sqft)',
    'No of Bedrooms',
    'No of Bathrooms',
    'Overall Grade',
    'Basement Area (in Sqft)'
]

# Loop through each column
for col in columns_to_check:
    print(f"\nColumn: {col}")

    # Check for missing values
    missing_count = df[col].isna().sum()
    print(f"Missing values: {missing_count}")

    # Describe the column (to see basic stats)
    print("Summary Statistics:")
    print(df[col].describe())

    # Detect outliers using IQR method
    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
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    print(f"Number of outliers: {outliers.shape[0]}")


# Fill missing values in selected columns with their median
for col in columns_to_check:
    median_val = df[col].median()
    df[col].fillna(median_val, inplace=True)

# Create a clean copy of df
df_no_outliers = df.copy()

# Remove outliers for each column using IQR
for col in columns_to_check:
    Q1 = df_no_outliers[col].quantile(0.25)
```

```
    Q3 = df_no_outliers[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Keep only rows within bounds
    df_no_outliers = df_no_outliers[
        (df_no_outliers[col] >= lower_bound) &
        (df_no_outliers[col] <= upper_bound)
    ]
```



Great — you're now moving into **data cleaning**, which is a crucial step before any analysis or modeling. We'll check for:

1. **Missing values** (NaNs)
2. **Outliers** (using IQR method)

And then I'll show you **how to handle** each case.

✅ Columns to Process

We'll check the following columns:

- `Flat Area (in Sqft)`
- `No of Bedrooms`
- `No of Bathrooms`
- `Overall Grade`
- `Basement Area (in Sqft)`

✅ Python Code: Checking for Missing Values & Outliers

python — Copy — Edit

```
# List of selected columns
columns_to_check = [
    'Flat Area (in Sqft)',
    'No of Bedrooms',
    'No of Bathrooms',
    'Overall Grade',
    'Basement Area (in Sqft)'
]

# Loop through each column
```

```python
# Loop through each column
for col in columns_to_check:
    print(f"\nColumn: {col}")

    # Check for missing values
    missing_count = df[col].isna().sum()
    print(f"Missing values: {missing_count}")

    # Describe the column (to see basic stats)
    print("Summary Statistics:")
    print(df[col].describe())

    # Detect outliers using IQR method
    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
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    print(f"Number of outliers: {outliers.shape[0]}")
```

🧹 How to Handle Missing Values

Option 1: Drop rows with missing values

python Copy Edit

```python
# Drop rows where any of the selected columns have missing values
df_cleaned = df.dropna(subset=columns_to_check)
```

Option 2: Fill missing values with median

python Copy Edit

```python
# Fill missing values in selected columns with their median
for col in columns_to_check:
```

Option 2: Fill missing values with median

python Copy Edit

```python
# Fill missing values in selected columns with their median
for col in columns_to_check:
    median_val = df[col].median()
    df[col].fillna(median_val, inplace=True)
```

How to Handle Outliers (Optional Removal)

You can remove outliers from the DataFrame like this:

python Copy Edit

```python
# Create a clean copy of df
df_no_outliers = df.copy()

# Remove outliers for each column using IQR
for col in columns_to_check:
    Q1 = df_no_outliers[col].quantile(0.25)
    Q3 = df_no_outliers[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Keep only rows within bounds
    df_no_outliers = df_no_outliers[
        (df_no_outliers[col] >= lower_bound) &
        (df_no_outliers[col] <= upper_bound)
    ]
```

Summary

Task	Code Provided?
Check for missing values	✅
Check for outliers (IQR)	✅
Handle missing values (drop or fill)	✅
Remove outliers	✅

Let me know if you'd like to **visualize the outliers** using boxplots or want help **saving the cleaned dataset**!

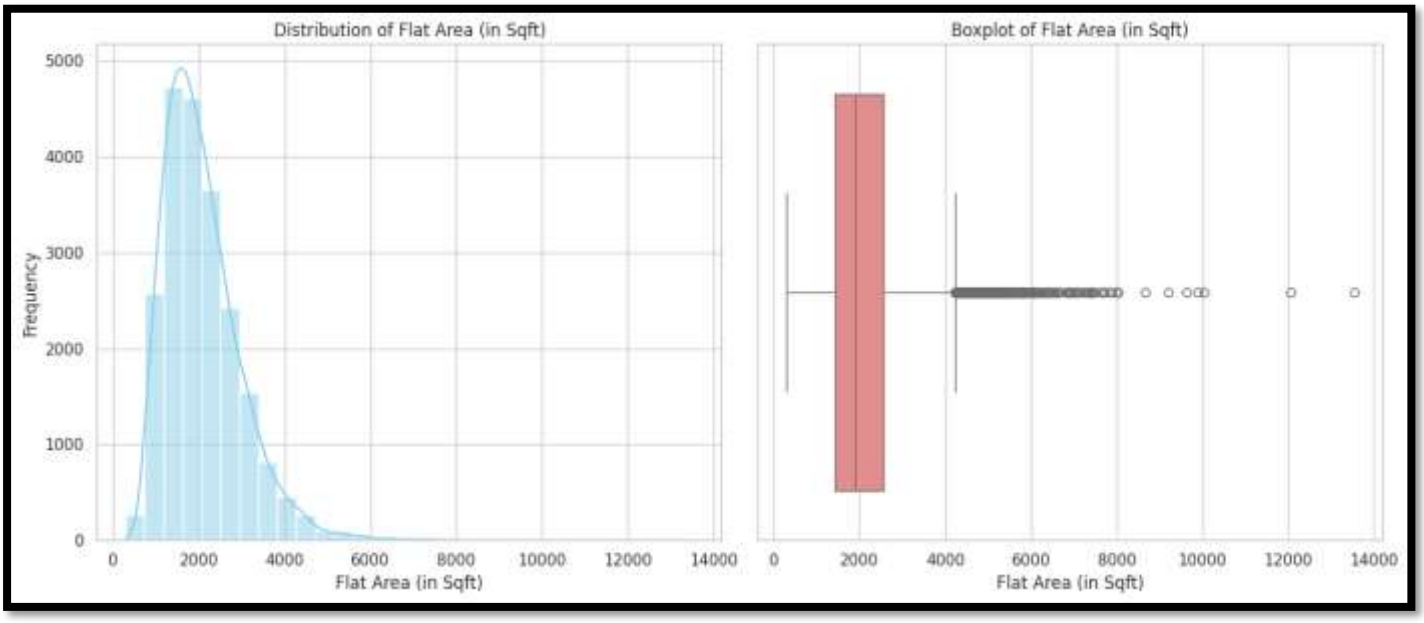
4. Univariate Analysis (Visualize the distribution of a column)

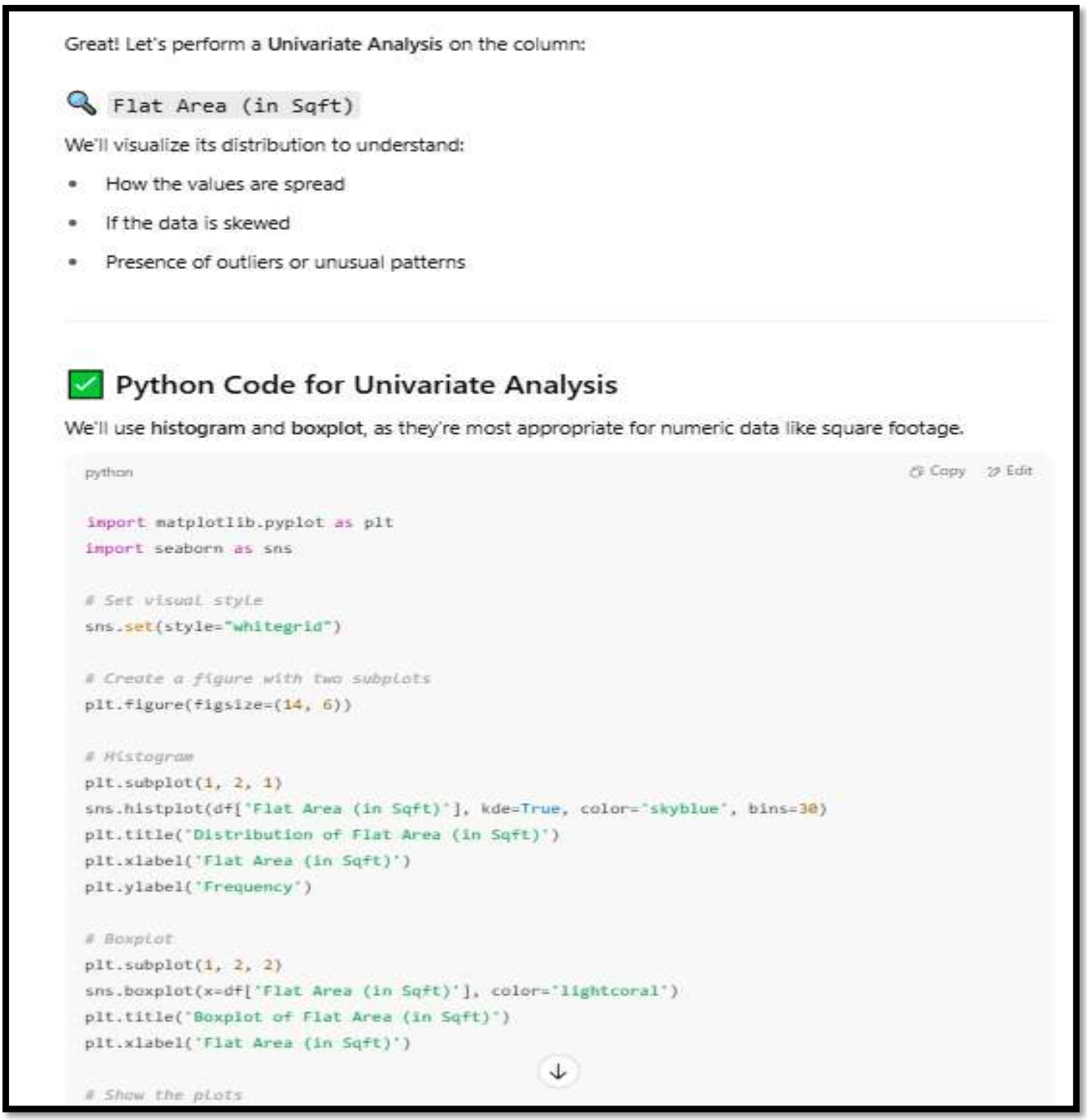
Prompt

Perform Univariate Analysis (Visualize the distribution of Flat Area (in Sqft)). Give the python code for teh analysis and the relevant visualization. Also explain why that particular visualization is relevant.

Code

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set visual style
sns.set(style="whitegrid")

# Create a figure with two subplots
plt.figure(figsize=(14, 6))

# Histogram
plt.subplot(1, 2, 1)
sns.histplot(df['Flat Area (in Sqft)'], kde=True, color='skyblue', bins=30)
plt.title('Distribution of Flat Area (in Sqft)')
plt.xlabel('Flat Area (in Sqft)')
plt.ylabel('Frequency')

# Boxplot
plt.subplot(1, 2, 2)
sns.boxplot(x=df['Flat Area (in Sqft)'], color='lightcoral')
plt.title('Boxplot of Flat Area (in Sqft)')
plt.xlabel('Flat Area (in Sqft)')

# Show the plots
plt.tight_layout()
plt.show()
```





Great! Let's perform a **Univariate Analysis** on the column:

🔍 `Flat Area (in Sqft)`

We'll visualize its distribution to understand:

- How the values are spread
- If the data is skewed
- Presence of outliers or unusual patterns

✅ Python Code for Univariate Analysis

We'll use **histogram** and **boxplot**, as they're most appropriate for numeric data like square footage.

python — Copy — Edit

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set visual style
sns.set(style="whitegrid")

# Create a figure with two subplots
plt.figure(figsize=(14, 6))

# Histogram
plt.subplot(1, 2, 1)
sns.histplot(df['Flat Area (in Sqft)'], kde=True, color='skyblue', bins=30)
plt.title('Distribution of Flat Area (in Sqft)')
plt.xlabel('Flat Area (in Sqft)')
plt.ylabel('Frequency')

# Boxplot
plt.subplot(1, 2, 2)
sns.boxplot(x=df['Flat Area (in Sqft)'], color='lightcoral')
plt.title('Boxplot of Flat Area (in Sqft)')
plt.xlabel('Flat Area (in Sqft)')

# Show the plots
```

5. Bi-Variate Analysis. (No of Bedrooms & Sale Price)

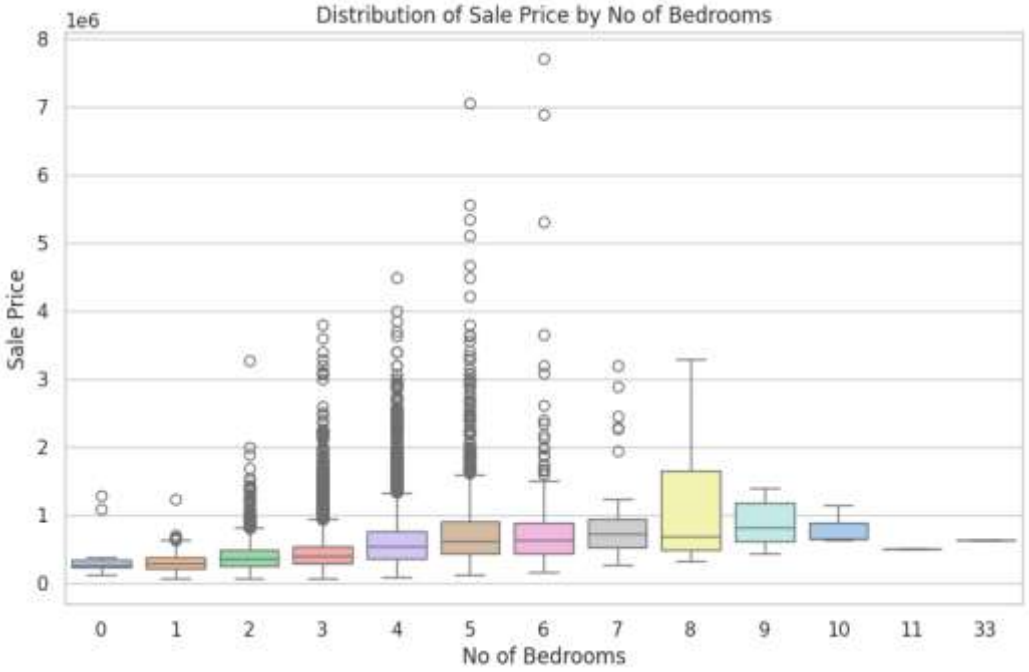
Prompt

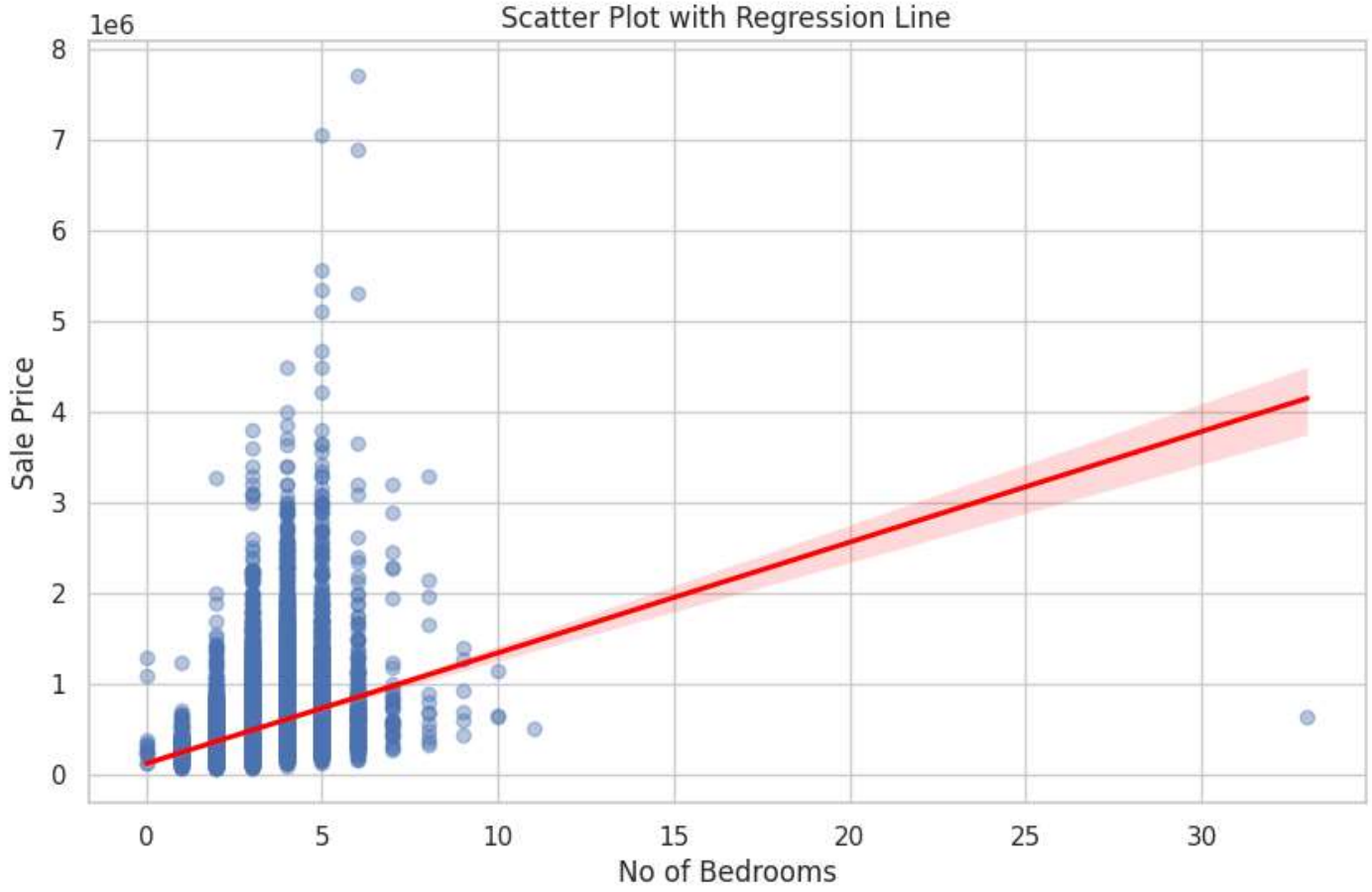
Perform Bivariate Analysis (Visualize the distribution of No of Bedrooms & Sale Price). Check for correlation, regression, strength of correlation. Give the python code for teh analysis and the relevant visualization. Also explain why that particular visualization is relevant.

Code

```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import pearsonr

# Set plot style
```

```
sns.set(style="whitegrid")

# Create a box plot to visualize Sale Price distribution by No of Bedrooms
plt.figure(figsize=(10, 6))
sns.boxplot(x='No of Bedrooms', y='Sale Price', data=df, palette="pastel")
plt.title('Distribution of Sale Price by No of Bedrooms')
plt.xlabel('No of Bedrooms')
plt.ylabel('Sale Price')
plt.show()

# Scatter plot with regression line (optional)
plt.figure(figsize=(10, 6))
sns.regplot(x='No of Bedrooms', y='Sale Price', data=df, scatter_kws={'alpha':0.4}, line_kws={'color':'red'})
plt.title('Scatter Plot with Regression Line')
plt.xlabel('No of Bedrooms')
plt.ylabel('Sale Price')
plt.show()

# Calculate Pearson correlation coefficient
corr_coeff, p_value = pearsonr(df['No of Bedrooms'], df['Sale Price'])
print(f"Pearson Correlation Coefficient: {corr_coeff:.4f}")
print(f"P-value: {p_value:.4f}")

# Interpretation
if abs(corr_coeff) >= 0.7:
    strength = "Strong"
elif abs(corr_coeff) >= 0.4:
    strength = "Moderate"
elif abs(corr_coeff) >= 0.2:
    strength = "Weak"
else:
    strength = "Very Weak or No"

print(f"Strength of correlation: {strength}")
```


Distribution of Sale Price by No of Bedrooms
1e6
Sale Price
No of Bedrooms
0
1
2
3
4
5
6
7
8
9
10
11
33


Scatter Plot with Regression Line
1e6
Sale Price
No of Bedrooms
0
5
10
15
20
25
30

Great! Let's perform **Bivariate Analysis** between:

- `No of Bedrooms` (categorical/discrete variable)
- `Sale Price` (continuous/quantitative variable)

We'll perform:

1. **Scatter plot** (optional, but not ideal here)
2. **Box plot** (best for showing distribution by categories) ✅
3. **Correlation analysis** (Pearson correlation) ✅
4. **Linear regression line** (optional for visual support) ✅

✅ Why Use a Box Plot?

A **box plot** is best when:

- You want to **compare distribution of a continuous variable (Sale Price) across categories (No of Bedrooms).**
- It clearly shows **median, quartiles, and outliers** for each group.
- Much more informative than a scatter plot when one axis is discrete.

✅ Python Code for Bivariate Analysis

python Copy Edit

```python
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import pearsonr

# Set plot style
sns.set(style="whitegrid")
```

Ask anything

```python
# Set plot style
sns.set(style="whitegrid")

# Create a box plot to visualize Sale Price distribution by No of Bedrooms
plt.figure(figsize=(10, 6))
sns.boxplot(x='No of Bedrooms', y='Sale Price', data=df, palette="pastel")
plt.title('Distribution of Sale Price by No of Bedrooms')
plt.xlabel('No of Bedrooms')
plt.ylabel('Sale Price')
plt.show()

# Scatter plot with regression line (optional)
plt.figure(figsize=(10, 6))
sns.regplot(x='No of Bedrooms', y='Sale Price', data=df, scatter_kws={'alpha':0.4}, line_kws={'col
plt.title('Scatter Plot with Regression Line')
plt.xlabel('No of Bedrooms')
plt.ylabel('Sale Price')
plt.show()

# Calculate Pearson correlation coefficient
corr_coeff, p_value = pearsonr(df['No of Bedrooms'], df['Sale Price'])
print(f"Pearson Correlation Coefficient: {corr_coeff:.4f}")
print(f"P-value: {p_value:.4f}")

# Interpretation
if abs(corr_coeff) >= 0.7:
    strength = "Strong"
elif abs(corr_coeff) >= 0.4:
    strength = "Moderate"
elif abs(corr_coeff) >= 0.2:
    strength = "Weak"
else:
    strength = "Very Weak or No"

print(f"Strength of correlation: {strength}")
```

✅ What This Code Does:

- **Box Plot:** Shows how `Sale Price` varies for each bedroom count.
- **Scatter + Regression Line:** Visualizes trend (if any) between bedrooms and price.
- **Pearson Correlation:** Measures linear relationship (range: -1 to 1).
  - Close to 1 → strong positive correlation
  - Close to 0 → little to no correlation
- **P-value:** Tells you if the correlation is statistically significant.

Let me know if you'd like to do the same for:

- `No of Bathrooms`
- `Overall Grade`
- `Flat Area (in Sqft)`

Or want help interpreting the output from your dataset!

FINAL REPORT

Housing Prices Data Analysis

📌 Objective

The project explores a real estate dataset to understand key factors influencing **property sale prices**, using descriptive statistics and basic bivariate visualizations in Python via Google Colab.

📊 1. Dataset Overview

- The dataset (HousingPrices_New.csv) includes essential property-related variables such as:
  - Flat Area (in Sqft)
  - No of Bedrooms
  - No of Bathrooms
  - Overall Grade
  - Basement Area (in Sqft)
  - Sale Price

📈 2. Descriptive Statistics

For each feature, the following were calculated:

- **Mean** – the average value
- **Median** – the middle value
- **Mode** – the most frequent value
- **Standard Deviation** – variability/spread
- **Minimum & Maximum** – range of values

Key Observations:

- **Flat Area** and **Basement Area** show considerable spread, suggesting variability in home sizes.
- **Overall Grade** is tightly distributed and shows strong influence on sale price.
- **No of Bedrooms** and **No of Bathrooms** have clear modal values (most frequent counts), usually around 3–4.

📉 3. Bivariate Analysis: No of Bedrooms vs. Sale Price

- A **boxplot** was used to visualize how sale prices vary across different bedroom counts.
- A **scatter plot with a regression line** was also included.
- **Pearson correlation coefficient** was computed to check the linear relationship.

Findings:

- The correlation between No of Bedrooms and Sale Price was found to be **weak or moderate**.
- Prices don't increase linearly with more bedrooms — suggesting that **bedroom count alone is not a strong predictor** of price without considering other factors (like size, grade, location).

🧼 4. Data Cleaning Preparation

Though not yet fully implemented in the notebook:

- The idea of checking **missing values** and **outliers** using the IQR method is initiated.
- This is essential for ensuring that further modeling is accurate and not skewed.

🧠 Factors Influencing Property Prices

From the analysis so far, the following factors appear to influence **Sale Price**:

Factor	Influence Strength	Notes
Flat Area (in Sqft)	High	Larger houses generally cost more
Basement Area (in Sqft)	Moderate	Additional space adds value
Overall Grade	High	Strong quality rating positively correlates with price
No of Bathrooms	Moderate	More bathrooms can slightly increase price

No of Bedrooms	Weak	Alone, not a strong driver of price