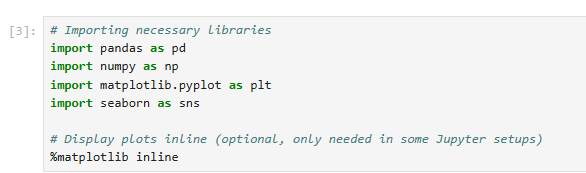
**Python-Case Study**

**Submitted By-**

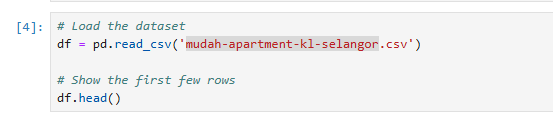
**Subrat Shukla, DE Batch1**

Dataset: mudah-apartment-kl-selangor

**1. Importing necessary libraries:**



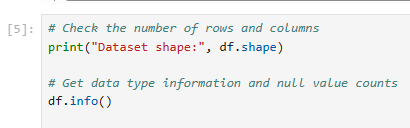
**2. Loading the given Dataset:**

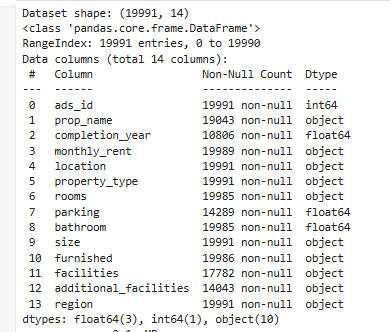




**Explanation:** pd.read\_csv() is a function in pandas used to read a CSV (Comma-Separated Values) file. The result is a DataFrame, a 2D structure similar to a table in Excel, with rows and columns.

**3. Number of rows and columns, data type information:**

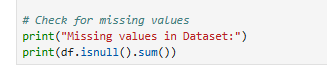




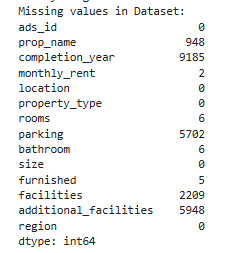
**Explanation:** df.shape,returns a tuple representing the DataFrame's dimensions: (number of rows, number of columns).

df.info(), summarizes the DataFrame, including column names, data types, non-null value counts, and memory usage.

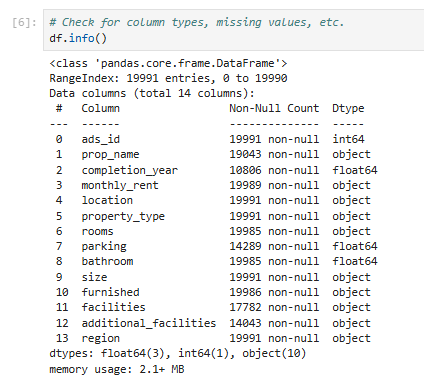
**4. Checking missing values:**



**Explanation:** ‘df.isnull()’ creates a DataFrame of the same shape, marking True for null values and False otherwise. ‘sum()’ calculates the total count of True values (null entries) for each column, providing a summary of missing data.



**5. Checking column types, missing values, etc.:**

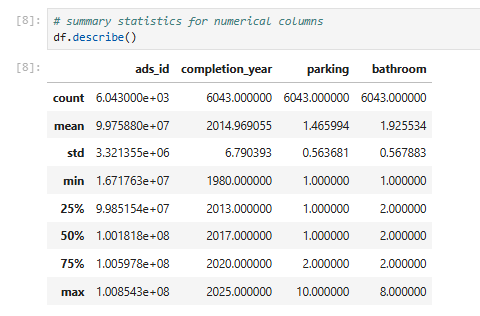


**6. Drop missing values rows:**



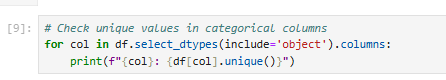
**Explanation:** Above formula removes all rows from the DataFrame that contain any missing (null) values. It is useful for cleaning the dataset by eliminating incomplete records.

**7. Summary statistics for numerical columns:**

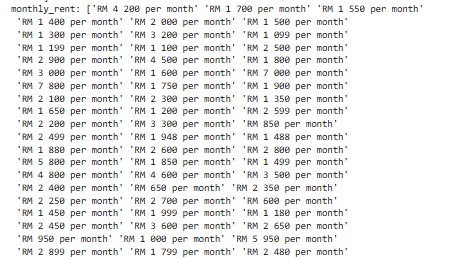


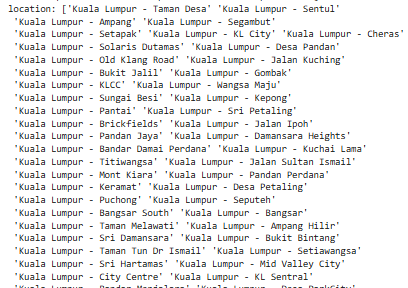
**Explanation:** The ‘df.describe()’ function generates summary statistics for all numerical columns in the DataFrame. This function helps in understanding the distribution, central tendency, and variability of the numerical data in the dataset.

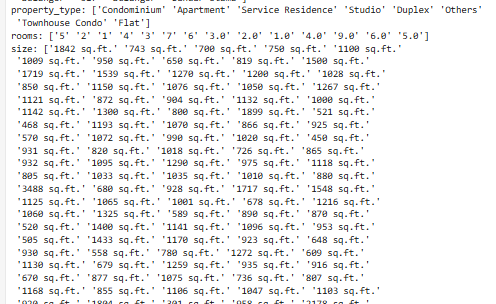
**8. Checking unique values in categorical columns:**

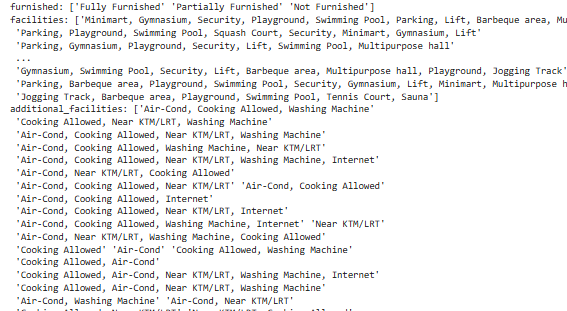








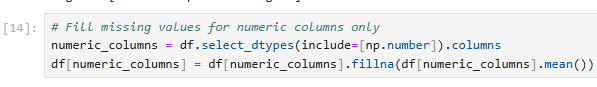




--and so on

**Explanation:** The above code snippet iterates through all columns in the DataFrame with a data type of object (typically used for categorical or string data). For each column, it prints the column name along with its unique values. This is helpful for exploring categorical data, identifying distinct categories, or detecting anomalies in the dataset.

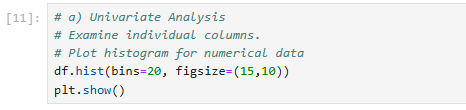
**9. Fill missing values for numeric columns only:**

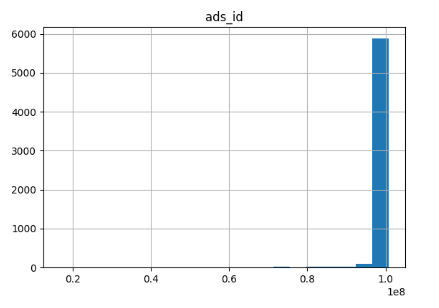
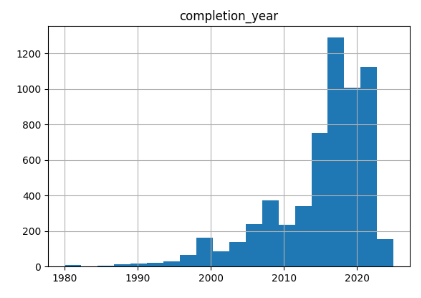


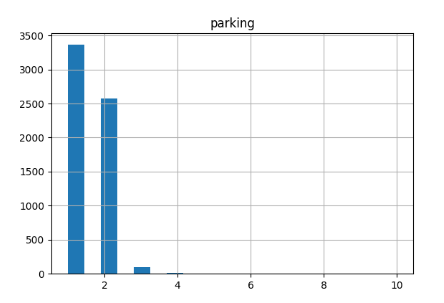
**Explanation:** This code identifies all numeric columns in the DataFrame and fills any missing (null) values in these columns with their respective mean values. The ‘df.select\_dtypes(include=[np.number]).columns’ extracts the names of numeric columns, and ‘df[numeric\_columns].fillna(df[numeric\_columns].mean())’ replaces null values with column-wise means.

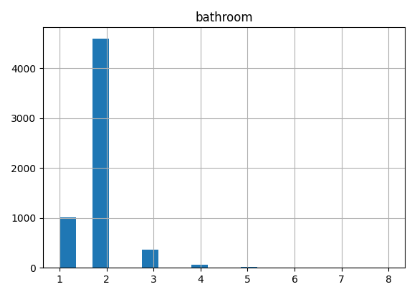
This technique ensures missing data in numerical columns is handled without removing rows.

**10. Univariate Analysis, plotting histogram for numerical data:**

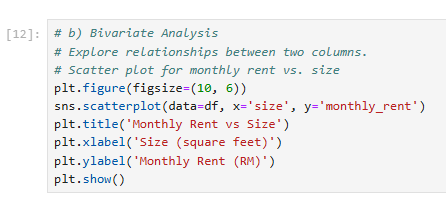


**Explanation:** This code generates histograms for all numerical columns in the DataFrame. The figsize parameter adjusts the overall size of the plot, and plt.show() displays the generated plots. This is a quick way to explore the data's distribution and detect patterns or outliers.  

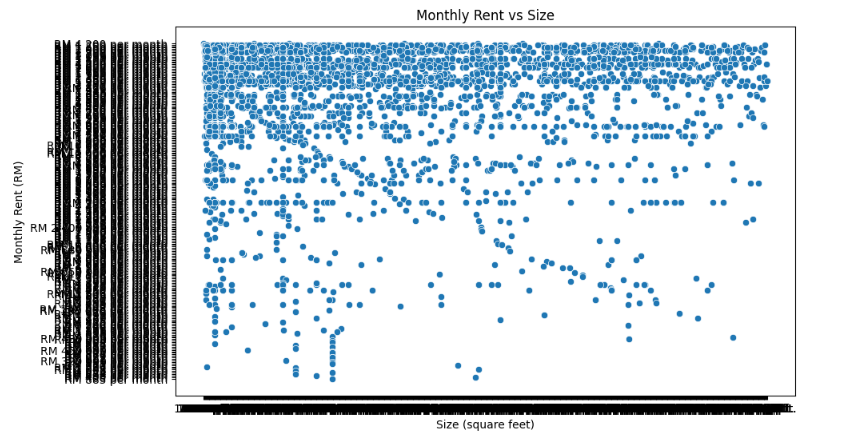




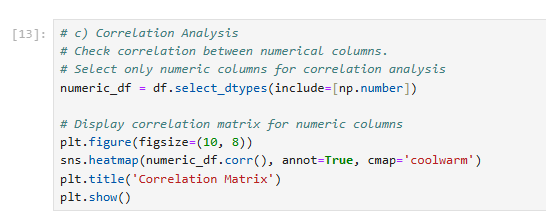
**11. Bivariate Analysis, plotting scatter for monthly rent vs. size:**



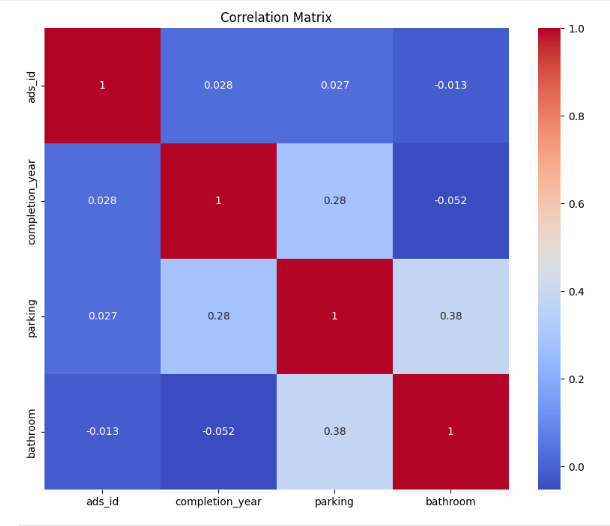
**Explanation:** This code creates a scatter plot to visualize the relationship between the size and monthly\_rent columns in the DataFrame. The plt.figure(figsize=(10, 6)) sets the plot's dimensions, while sns.scatterplot() from the seaborn library plots the data points. The plt.title(), plt.xlabel(), and plt.ylabel() functions add a title and labels to the axes for better readability. Finally, plt.show() displays the plot, helping to identify trends or correlations between property size and rent.



**12. Correlation Analysis, checking the correlation between numerical columns:**



**Explanation:** This code computes and visualizes the correlation matrix for all numeric columns in the DataFrame. The `numeric\_df = df.select\_dtypes(include=[np.number])` filters the DataFrame to include only numeric columns. The `numeric\_df.corr()` calculates the pairwise correlation coefficients. The `sns.heatmap()` creates a heatmap to visualize these correlations, with annotations (`annot=True`) showing the correlation values and a color gradient (`cmap='coolwarm'`) for better differentiation. This plot helps identify strong positive or negative relationships between numerical variables.



**--Thank You!**