Table of Contents

Understanding Paris's Housing Market Through Data

Introduction

Paris's housing market is known for steep prices and diverse property features, making it ideal for exploring how different variables affect value. This report presents an exploratory data analysis (EDA) to reveal trends and anomalies, aiding in strategic real-estate decisions. We import, clean, and visualize data, employing correlation and regression techniques to quantify relationships. By critically examining assumptions, we aim to guide the company on which factors most drive property prices, informing more robust strategies.

Methodology

The analysis was conducted using Python, the preferred language for exploratory data analysis (EDA) due to its versatility and data handling capabilities. The process followed a structured workflow, as outlined below:

1. Data Import and Initial Inspection

The necessary Python libraries—Pandas (for data manipulation), Matplotlib, and Seaborn (for visualization)—were imported to facilitate the analysis as seen in figure A1. The dataset, stored as a CSV file, was loaded into a Pandas DataFrame (a tabular data structure consisting of rows and columns; McKinney, 2022). See figure A2.

We started by performing some initial checks which included: checking the shape of the dataset. The DataFrame contained 19,999 rows and 11 columns which provided a substantial sample for analysis, as seen in figure A3. The first five rows of the DataFrame were examined to assess the structure of the data and to detect immediate anomalies and outliers. See figure A4.

Following this a check was done on the data types of the variables as seen in figure A5. The variables were primarily numerical (floats and integers), though it was noted that some variables were incorrectly classified. Next, a check for missing values revealed that all columns except floors, grade and renovated contained missing values. See figure A6. This issue will be addressed in the pre-processing stage. These issues will be addressed in the pre-processing stage.

2. Basic Statistical test

Basic statistical measures were calculated to summarize the dataset as seen in figure A7. By generating these key metrics—count, mean, standard deviation, min/max values, and percentiles—for all numeric columns in a dataset, we can get a quick yet powerful snapshot of the data distribution and quality (Mukhiya and Ahmed, 2020). From this statistical test, we were able to conclude that:

- The houses are typically 3 bedrooms but range from 1 to 33, which is an outlier.
- There is a high variability (€365,922) in prices likely due to differences in property features or locations.
- Most properties have 1 floor, with only a few reaching 3.5 floors.

To dig even deeper into the data, as seen in figure A8, a correlation matrix was generated to quantify relationships between variables, identifying potential predictors for further investigation. According to Wienclaw (2021), "...correlation mathematically express the degree of relationship between two events or variables on a scale of 0.0 (demonstrating no relationship between the two variables) to 1.0 (demonstrating a perfect relationship between the variables)". From this, we deduced that: living room square footage (r=0.70) has the strongest correlation with price, grade(r=0.67) has a positive effect on price and condition (r=0.14) has a surprisingly weak effect on prices.

These findings are supported by Musa & Yusoff (2015) who posited that factors such as the size of rooms have a profound influence on price. This means buyers are willing to pay significantly more for spacious, high-quality homes. Renovations also had a modest positive effect on prices, but surprisingly, the overall condition of a home—rated on a scale from 1 to 5—had almost no impact. This suggests that buyers in Paris may prioritize potential over perfection, seeing value in homes that could be improved rather than those already in great shape.

3. Data Preprocessing

The next step is preparing the data, a crucial step to ensure accuracy. According to Mukhiya & Ahmed (2020), before data can be properly analyzed, it needs to go through a process where we fix incomplete records, remove duplicates, correct errors, and fill missing values. Three primary issues were addressed:

A. Handling Missing Values:

   Only 37 rows in our data (<0.2%) had missing values. Given their perceived negligible impact on the overall dataset, they were excluded. In cases where missing data is more extensive, other techniques could be considered (Mukhiya & Ahmed, 2020), but given the small scale here, removing them was the simplest option. See figure A9.

B. Checking for Duplicates:

A custom function revealed four duplicated rows in the dataset (Figure A10). After visually confirming these were genuine duplicates, the first occurrence was retained and the others removed (Figure A11) to retain the quality of our data.

C. Data Type Standardization (as seen in figure A12):

As seen in figure A5, variables such as “built” were stored as floats, implying fractional years. Since that makes little sense for years, they were recategorized as integers to prevent errors as seen in figure A12. Ensuring correct data types helps prevent computational or interpretive errors—especially when generating summary statistics, where integer versus float differences can distort results. (McKinney, 2022).

4. Further Statistical test

Once the pre-processing was completed, we performed basic statistical tests again to see how the changes affected the dataset. As shown in Figure A13, preprocessing had a minimal impact on the dataset: the mean price increased only slightly from €535,394.40 in Figure A7 to €535,529.74.

5. Investigation using graphical representations

Price and Number of bedrooms

To explore the relationship between price and number of bedrooms, a linear regression model was created which plotted the variables against each other as seen in figure A14. For each additional bedroom, the model suggests an estimated price increase of €119,125.83, as expressed by the equation **price = 119,125.83 × bedrooms + 134,439.36**, with an R^2 of 0.916. See figure A15. R^2 represents the proportion (or percentage) of the total variation in the variable that is accounted for by the model (*Montgomery, Peck & Vining, 2012, p. 29).* With an R^2 of 0.0916, bedrooms alone explain only 9.16% of price variation, suggesting that other factors—like square footage or condition—must significantly influence property values.

Price and Grade

A second linear regression model (Figure A16) predicts price from grade, producing the equation **Price = €208,280 x grade + -1,048,759.15** (Figure A17), where each additional grade point corresponds to €208,281 increase in price. This suggests that buyers are willing to pay significantly more for higher-quality finishes and design standards. The model had an R^2 of 0.4433 which indicates that grade is moderately important in predicting price (44.33%), though other factors still exert influence. Nonetheless, it is stronger than the bedroom variable, which explained just 9.16% of price variation as seen in figure A14.

Price and Square Footage of the Living room

A regression model developed to predict price based on the square footage of the living room (see figure A17), revealed that for every additional square foot of living space, the home's price increases by $283.42, on average. As seen in figure A18, the equation for the model is: **price = 283.42 x sqftliving - 47,58907.**

Number of Floors and Total Square Footage

Another model was developed to examine the correlation between the number of floors and the total square footage using Pearson correlation coefficient. Pearson correlation coefficient measures the strength of a relationship between two variables Sheposh (2025).As seen in figure A19, the Pearson correlation coefficient between the two variables is 0.2, indicating the relationship between the two variables is not linear as the closer a number is to 1 the stronger the relationship. A visual representation of this correlation can be seen in figure A20.

Key Findings

Our findings underscore the importance of highlighting living space and property quality rather than focusing solely on the number of bedrooms. A well-designed, spacious home in a desirable location tends to attract more buyers than a larger property with a poor layout. In Paris's housing market, square footage and quality appear to drive prices more strongly than bedroom counts or generalized condition. For sellers, this means emphasizing roomy layouts and premium finishes. For buyers, it suggests targeting well-located homes with the potential for upgrades. For developers, it underscores the value of creating open, high-quality living spaces that align with modern preferences. Grounding our strategies in these insights will enable more informed decisions that meet genuine buyer needs—and foster greater success in Paris's competitive real estate landscape.

References

Downey, A.B. (2014) *Think Stats* [online]. 2nd edition. O'Reilly Media Inc.

Harper-Scott, L. (2025) *European housing market has turned a corner* [online]. Oxford Economics. Available from: https://www.oxfordeconomics.com/resource/european-housing-market-has-turned-a-corner/ [Accessed 31 March 2025].

McKinney, W. (2022) *Python for Data Analysis* [online]. 3rd edition. O'Reilly Media Inc.

Montgomery, D.C., Peck, E.A. and Vining, G.G. (2012) Introduction to Linear Regression Analysis. 5th edn. Hoboken, NJ: Wiley

Mukhiya, S.K. & Ahmed, U. (2020) *Hands-On Exploratory Data Analysis with Python: Perform EDA Techniques to Understand, Summarize, and Investigate Your Data*. Packt Publishing.

Musa, U. & Yusoff, W.Z.W. (n.d.) *The Influence of Housing Components on Prices of Residential Houses: A Review of Literature* [online]. Available from: https://core.ac.uk/download/pdf/42955865.pdf [Accessed 31 March 2025].

Notaires de France (2025) *Marché de l'immobilier : tendances et évolutions des prix de l'immobilier - janvier 2025* [online]. Available from: https://www.notaires.fr/fr/article/marche-de-limmobilier-tendances-et-evolutions-des-prix-de-limmobilier-janvier-2025 [Accessed 31 March 2025].

Sheposh, R. (2025) ‘Pearson correlation coefficient (PCC)’, *Salem Press Encyclopedia of Science* [Preprint]. Available at: https://research.ebsco.com/linkprocessor/plink?id=e307f95e-97f8-369c-83ef-3c8a38bbba87 (Accessed: 11 April 2025).

Wienclaw, R.A. (2021) *Correlation*. Salem Press Encyclopedia [online]. Available from: https://research.ebsco.com/linkprocessor/plink?id=fef02228-0378-3d23-a80f-4fb6e0b2b4cb [Accessed 31 March 2025].

Appendix A

[90]:
```
#Importing the necessary libraries needed for the EDA. NumPy was used for numerical data manipulation in Python, while Pandas was employed
#for handling structured or tabular data, such as our dataset.Additionally, Pandas facilitated data preprocessing, including managing missing
#values. For data visualization, Matplotlib and Seaborn were utilized to generate graphical representations,aiding in the exploration and
#analysis of the dataset. Sklearn is a machine learning module that is built into Python that will be used to create the models that will idetify
#trends in the data
import pandas as pd
from scipy import stats
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

Figure A1: Importing the Python libraries needed for the EDA.

[13]:
```
#importing the dataset into a pandas DataFrame called 'Housing_Data'
Housing_Data= pd.read_excel("Paris_housing_Data_Set.xlsx", engine="openpyxl")
```

Figure A2: Importing the dataset into a Pandas DataFrame

[91]:
```
#Checking the initial shape of the data to get an understanding of the magnitude of the data that we are working with.
Housing_Data.shape
```

[91]: (19999, 11)

Figure A3: showing the shape of the DataFrame

[103]:
```
#Checking the first five rows of the data to get an idea of what the data looks like. From this peek of the data,errors were identified that will be
# addressed in the pre-processing stage
Housing_Data.head()
```

[103]:

	price	bedrooms	bathrooms	sqft_living	sqft_total	floors	condition	grade	built	renovated	living_area_sqft
0	221900.0	3.0	1.0	1180.0	5650.0	1.0	3.0	7	1955.0	0	1340.0
1	538000.0	3.0	NaN	2570.0	7242.0	2.0	3.0	7	1951.0	1991	1690.0
2	180000.0	2.0	1.0	770.0	10000.0	1.0	3.0	6	1933.0	0	2720.0
3	604000.0	4.0	3.0	1960.0	5000.0	1.0	5.0	7	1965.0	0	1360.0
4	510000.0	3.0	2.0	1680.0	8080.0	1.0	3.0	8	1987.0	0	1800.0

Figure A4: showing the first 5 columns of the DataFrame

[133]:
```
#Checking the data types of the variables to ensure that all variables are assigned the correct data type. Variables such as built, should be integers
#and not floats to accurately reflect the nature of the data as years cannot be fractional.
Housing_Data.dtypes
```

[133]:
```
price               float64
bedrooms            float64
bathrooms           float64
sqft_living         float64
sqft_total          float64
floors              float64
condition           float64
grade                 int64
built               float64
renovated             int64
living_area_sqft    float64
dtype: object
```

Figure A5: showing the date type of the variables in the DataFrame

```
[17]: #Checking for missing values within the dataframe
      Housing_Data.isnull().sum()
```

```
[17]: price              4
      bedrooms          15
      bathrooms         11
      sqft_living        1
      sqft_total         5
      floors             0
      condition          1
      grade              0
      built              1
      renovated          0
      living_area_sqft   6
      dtype: int64
```

Figure A6: showing the missing values in the dataset

```
[134]: #performing a basic statistical test on the dataset to get some insights into the data. Based on this test, we can see that the houses in our dataset
       #typically have 3 bedrooms but ranges from 1 to 33 which is an outlier. The 33-bedroom property, I believe, points to the existence of luxury propertie
       #I've chosen not to remove this datapoint because it represents real work scenarios because Paris does have luxury properties. So, while it may skew
       #some statistics such as the mean, it provides real word insight into the luxury market in Paris.
       #The houses have between 1.5 to 2.5 bathrooms. Also, most properties have 1 floor but a few have 3.5 floors.
       #The column sqft_living closely resembles living_area_sqft which raises the question about if the data is redundant.
       Housing_Data.describe().round(2)
```

[134]:

	price	bedrooms	bathrooms	sqft_living	sqft_total	floors	condition	grade	built	renovated	living_area_sqft
count	19995.00	19984.00	19988.00	19998.00	19994.00	19999.00	19998.00	19999.00	19998.00	19999.00	19993.0
mean	535394.40	3.37	2.07	2057.94	15606.96	1.44	3.44	7.61	1967.95	90.81	1974.2
std	365921.66	0.93	0.76	905.64	41775.76	0.52	0.67	1.17	28.32	415.95	675.2
min	75000.00	1.00	0.50	290.00	520.00	1.00	1.00	1.00	1900.00	0.00	399.0
25%	317000.00	3.00	1.50	1420.00	5350.00	1.00	3.00	7.00	1950.00	0.00	1490.0
50%	449900.00	3.00	2.00	1900.00	7817.50	1.00	3.00	7.00	1969.00	0.00	1830.0
75%	640000.00	4.00	2.50	2510.00	11000.00	2.00	4.00	8.00	1991.00	0.00	2336.0
max	7700000.00	33.00	8.00	13540.00	1651359.00	3.50	5.00	13.00	2015.00	2015.00	6210.0

Figure A7: showing the results of the statistical test performed on the DataFrame

```
•[20]: #performing an initial correlation matrix. This test shows that the strongest predictor of price re sqft_living(0.70), grade(0.67) and bathrooms(0.52) while sqft_toal(0.09)
       #has minimal impact on price. Other notable mentions are bathrooms and sqft_living which have a correlation of 0.76 and grade and sqft_living which have a correlation of 0.77.
       Housing_Data.corr()
```

[20]:

	price	bedrooms	bathrooms	sqft_living	sqft_total	floors	condition	grade	built	renovated	living_area_sqft
price	1.000000	0.302715	0.524730	0.701247	0.085870	0.278690	0.047269	0.665259	0.040088	0.135701	0.595678
bedrooms	0.302715	1.000000	0.515724	0.568133	0.030603	0.204928	0.033419	0.354848	0.159865	0.021052	0.376581
bathrooms	0.524730	0.515724	1.000000	0.761640	0.092467	0.505132	-0.096782	0.661418	0.488162	0.066027	0.576770
sqft_living	0.701247	0.568133	0.761640	1.000000	0.171361	0.391976	-0.046014	0.766483	0.321674	0.064575	0.753186
sqft_total	0.085870	0.030603	0.092467	0.171361	1.000000	0.015878	-0.017107	0.117585	0.076945	0.005526	0.151518
floors	0.278690	0.204928	0.505132	0.391976	0.015878	1.000000	-0.227628	0.464452	0.431733	0.027793	0.318028
condition	0.047269	0.033419	-0.096782	-0.046014	-0.017107	-0.227628	1.000000	-0.123643	-0.323934	-0.071889	-0.086868
grade	0.665259	0.354848	0.661418	0.766483	0.117585	0.464452	-0.123643	1.000000	0.435028	0.024937	0.729372
built	0.040088	0.159865	0.488162	0.321674	0.076945	0.431733	-0.323934	0.435028	1.000000	-0.219860	0.341028
renovated	0.135701	0.021052	0.066027	0.064575	0.005526	0.027793	-0.071889	0.024937	-0.219860	1.000000	0.001815
living_area_sqft	0.595678	0.376581	0.576770	0.753186	0.151518	0.318028	-0.086868	0.729372	0.341028	0.001815	1.000000

Figure A8: showing the relationship between the variables in the DataFrame.

```
[139]: #Checking for rows with missing values in the dataset have missing values to determine how to handle them.The dataset contained a minimal number of
       #null entries (37 rows, representing less than 0.2% of total observations). Given their negligible impact on the overall dataset (19,999 rows) and
       #the preservation of key statistical properties post-removal, they were removed from the dataset and the analysis.
       num_of_na= Housing_Data.isna().any(axis=1).sum()
       print(num_of_na)

       37

[140]: #Since there are 37 rows with missing values they can be dropped because they are not material when compared to the number of data points
       #that are in our dataset.
       Cleaned_Data=Housing_Data.dropna().copy()
```

Figure A9: showing the number of missing values in the dataset and their deletion

```
[112]: # Checking for duplicates across all columns
       duplicates = Cleaned_Data[Cleaned_Data.duplicated(keep=False)]

       if not duplicates.empty:
           print(f"Found {len(duplicates)} duplicate rows:")
           print(duplicates.sort_values(list(Cleaned_Data.columns)))
       else:
           print("No duplicates found in Cleaned_Data")

       Found 8 duplicate rows:
                 price  bedrooms  bathrooms  sqft_living  sqft_total  floors  \
       547    259950.0       2.0       2.00       1070.0       649.0     2.0
       4352   259950.0       2.0       2.00       1070.0       649.0     2.0
       3950   550000.0       4.0       1.75       2410.0      8447.0     2.0
       3951   550000.0       4.0       1.75       2410.0      8447.0     2.0
       14982  585000.0       3.0       2.50       2290.0      5089.0     2.0
       14983  585000.0       3.0       2.50       2290.0      5089.0     2.0
       16380  629950.0       3.0       2.50       1680.0      1683.0     2.0
       17242  629950.0       3.0       2.50       1680.0      1683.0     2.0

              condition  grade   built  renovated  living_area_sqft
       547          3.0      9  2008.0          0            1070.0
       4352         3.0      9  2008.0          0            1070.0
       3950         4.0      8  1936.0       1980            2520.0
       3951         4.0      8  1936.0       1980            2520.0
       14982        3.0      9  2001.0          0            2290.0
       14983        3.0      9  2001.0          0            2290.0
```

Figure A10: showing the duplicate records

```
•[113]: # After visualling confirming that the data point were duplicate the duplicates were dropped but the first occurence was kept.
        Cleaned_Data = Cleaned_Data.drop_duplicates()

        # Verify duplicates were removed
        remaining_dupes = Cleaned_Data[Cleaned_Data.duplicated()]
        print(f"Remaining duplicates after dropping: {len(remaining_dupes)}")

        Remaining duplicates after dropping: 0
```

Figure A11: showing the check for duplicates and then their removal from the DataFrame

```
•[114]: ## Filling missing values with 0 and converting columns to appropriate data types:
        # - Integer columns:for discrete variables
        # - Float columns: for continous variables
        #Some variables, like the "built" year, were incorrectly stored as floats, implying fractional years. Since fractional values have no practical meaning
        #for the construction year, they were recategorized to integers.
        Cleaned_Data['bedrooms'] = Cleaned_Data['bedrooms'].fillna(0).astype(float)
        Cleaned_Data['floors'] = Cleaned_Data['floors'].fillna(0).astype(float)
        Cleaned_Data['condition'] = Cleaned_Data['condition'].fillna(0).astype(float)
        Cleaned_Data['grade'] = Cleaned_Data['grade'].fillna(0).astype(float)
        Cleaned_Data['built'] = Cleaned_Data['built'].fillna(0).astype(int)
        Cleaned_Data['renovated'] = Cleaned_Data['renovated'].fillna(0).astype(int)
        Cleaned_Data['price'] = Cleaned_Data['price'].fillna(0).astype(float)
        Cleaned_Data['bathrooms'] = Cleaned_Data['bathrooms'].fillna(0).astype(float)
        Cleaned_Data['sqft_living'] = Cleaned_Data['sqft_living'].fillna(0).astype(float)
        Cleaned_Data['sqft_total'] = Cleaned_Data['sqft_total'].fillna(0).astype(float)
        Cleaned_Data['living_area_sqft'] = Cleaned_Data['living_area_sqft'].fillna(0).astype(float)
```

Figure A12: showing the recategorization of some variables

[28]:
```
#Doing a statistical test on the cleaned data
Cleaned_Data.describe().round(2)
```

[28]:

	price	bedrooms	bathrooms	sqft_living	sqft_total	floors	condition	grade	built	renovated	living_area_sqft
count	19962.00	19962.00	19962.00	19962.00	19962.00	19962.00	19962.00	19962.00	19962.00	19962.00	19962.00
mean	535529.74	3.37	2.07	2058.37	15602.44	1.44	3.44	7.61	1967.94	90.88	1974.32
std	366026.94	0.93	0.76	905.43	41783.60	0.52	0.67	1.17	28.32	416.10	675.07
min	78000.00	1.00	0.50	370.00	520.00	1.00	1.00	3.00	1900.00	0.00	399.00
25%	317500.00	3.00	1.50	1420.00	5350.00	1.00	3.00	7.00	1950.00	0.00	1490.00
50%	449950.00	3.00	2.00	1900.00	7815.50	1.00	3.00	7.00	1969.00	0.00	1830.00
75%	640000.00	4.00	2.50	2510.00	11000.00	2.00	4.00	8.00	1991.00	0.00	2330.00
max	7700000.00	33.00	8.00	13540.00	1651359.00	3.50	5.00	13.00	2015.00	2015.00	6210.00

Figure A13: showing statistical test performed on the cleaned DataFrame

[150]:
```
#Creating the model to predict price using number of bedrooms with linear regression.The model indicated that the equation for
#the relationship between the two variables is: price=119,125.83×bedrooms+134439.357 which means, for each additional bedroom, the Price increases
#by €119,125.83, on average.

model_bedroom = LinearRegression()
model_bedroom.fit(Cleaned_Data[['bedrooms']], Cleaned_Data['price'])
bedroom_pred = model_bedroom.predict(Cleaned_Data[['bedrooms']])

r2 = r2_score(Cleaned_Data['price'], bedroom_pred)

#Printing the equation for the model
print("Bedroom Model:") #For each additional bedroom, the Price increases by $119,125.83, on average
print(f"Price = {model_bedroom.coef_[0]:.2f} × bedrooms + {model_bedroom.intercept_:.2f}\n")
print(f"R-squared: {r2:.4f}")

#Plotting
fig, ax = plt.subplots(figsize=(8, 6))

#formatting price as a currency
ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '€{:,.0f}'.format(x)))

ax.scatter(Cleaned_Data['bedrooms'], Cleaned_Data['price'], alpha=0.5, label='Actual')
ax.plot(Cleaned_Data['bedrooms'], bedroom_pred, 'r-', label='Predicted')

# Labels and title
ax.set_xlabel('bedrooms')
ax.set_ylabel('Price')
ax.set_title('Price Prediction by Number of Bedrooms')
ax.legend()

plt.tight_layout()
plt.show()
```

Figure A14: showing model that can be used to predict the price using the number of bedrooms

```
Bedroom Model:
Price = 119125.83 × bedrooms + 134439.35

R-squared: 0.0916
```

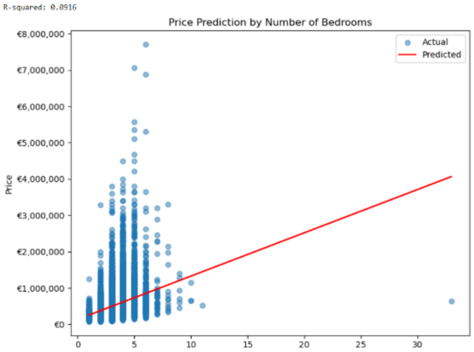

Figure A15: showing the model equation for predicting price based on the number of bedrooms

```
[72]: #PRICE VS GRADE

[74]: #Creating the model to predict price using grade with linear regression
model_grade = LinearRegression()
model_grade.fit(Cleaned_Data[['grade']], Cleaned_Data['price'])
grade_pred = model_grade.predict(Cleaned_Data[['grade']])

#Calculating the R squared which looks at the proportion of price that can be explained by grade.In this case,44% of the price can be explained by the grade of a property.
r2 = r2_score(Cleaned_Data['price'], grade_pred)

# Printing equation for the grade model
print("Grade Model:") #For every 1 unit that grade increases, price increases by $208,280.87
print(f"Price = {model_grade.coef_[0]:.2f} × grade + {model_grade.intercept_:.2f}\n")
print(f"R-squared: {r2:.4f}")

# Plotting the grade model
plt.subplot(1, 2, 1)
plt.scatter(Cleaned_Data['grade'], Cleaned_Data['price'], alpha=0.5, label='Actual')
plt.plot(Cleaned_Data['grade'], grade_pred, 'r-', label='Predicted')
plt.xlabel('Grade')
plt.ylabel('Price')
plt.title('Price Prediction by Grade')
plt.legend()
```

Figure A16: showing the regression model developed to predict price based on the grade of a house.

```
Grade Model:
Price = 208332.97 × grade + -1049101.88

R-squared: 0.4434
[30]: <matplotlib.legend.Legend at 0x1fa42300110>
```

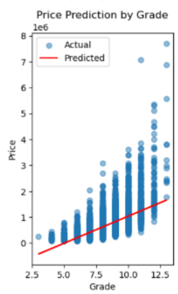

Figure A17: showing the equation and a graphical representation of the grade model

```
[31]: #PRICE VS SQFT_LIVING

[32]: #Creating a model to predict price using sqft_living using linear regression
model_sqft = LinearRegression()
model_sqft.fit(Cleaned_Data[['sqft_living']], Cleaned_Data['price'])
sqft_pred = model_sqft.predict(Cleaned_Data[['sqft_living']])

#calculating the R2 which shows that 49% of the price can be explained by sqft_living
r2 = r2_score(Cleaned_Data['price'], sqft_pred)

# Printing equation for the grade model
print("Square Footage Model:")#For every additional square foot of living space, the Price increases by $283.42, on average.
print(f"Price = {model_sqft.coef_[0]:.2f} x sqft_living + {model_sqft.intercept_:.2f}")
print(f"R-squared: {r2:.4f}")

# Plot for the sqft_living room vs price model
plt.subplot(1, 2, 2)
plt.scatter(Cleaned_Data['sqft_living'], Cleaned_Data['price'], alpha=0.5, label='Actual')
plt.plot(Cleaned_Data['sqft_living'], sqft_pred, 'r-', label='Predicted')
plt.xlabel('Living Area (sqft)')
plt.ylabel('Price')
plt.title('Price Prediction by Square Footage')
plt.legend()
```

Figure A18: showing the linear regression model to predict price based on the square footage of the living room of a property.

```
Square Footage Model:
Price = 283.42 × sqft_living + -47859.07
[167]: <matplotlib.legend.Legend at 0x1035ca5eed0>
```

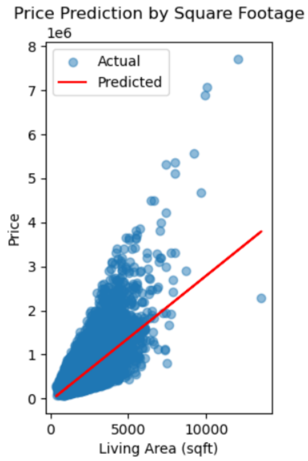

Figure A19: showing the equation for the model and a graphical representation.

```
[ ]: #FLOORS VS SQFT_TOTAL
```

```
•[180]: #calculating the correlation between floors and sqft_total
correlation = Cleaned_Data['floors'].corr(Cleaned_Data['sqft_total'])
print(f"Pearson Correlation (r): {correlation:.2f}")
# Preparing the data to create the model
X = Cleaned_Data[['floors']]  # Feature
y = Cleaned_Data['sqft_total']  # Target

# Fit regression
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)

# Plotting the model. There is a correlation of 0.02 which is close to zero which indicates that no linear relationship exists between the two variables.
#The scatter shows that square footage does not increase with the number of floors. It can be observed that 1 floor homes have a range of square footage and 3-floor homes are
#potentially smaller per-square footage.
plt.figure(figsize=(8, 5))
plt.scatter(X, y, alpha=0.6, color='blue', label='Data Points')
plt.plot(X, y_pred, color='red', linewidth=2, label='Regression Line')
plt.title(f"Floors vs. Total Sqft (r = {correlation:.2f})", fontsize=14)
plt.xlabel("Number of Floors", fontsize=12)
plt.ylabel("Total Square Footage", fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```

Figure A20: showing the model used to examine the correlation between number of floors and total square footage.

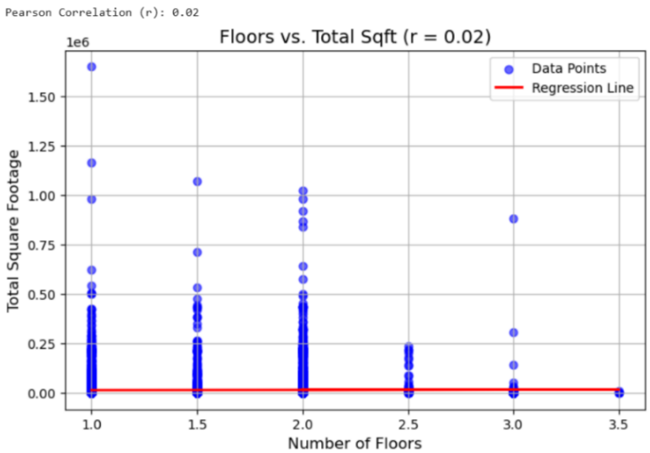

Figure A21: showing the correlation between number of floors and the total square footage.

Appendix B

Link to the full Python Code:

Mathematics_for Data_Science_1.1_STU24110970.ipynb