5/6/2025

# Tidying Up the Dataset “California Housing Prices”

Assignment task 1 for the course: Data quality and data wrangling

Alvine Gadeu Nee Djomatchie Touko
IU AKADEMIE

# Content



Alvine Gadeu, Student number: UPS10745343

# Introduction

Data analysis is especially important in today's world, where we are constantly surrounded by vast amounts of data. However, data alone is not valuable unless it can be understood and utilized effectively and that's where data analysis becomes essential. The first step in working with data is data wrangling, which involves cleaning and preparing the data for analysis. In the analysis phase, the data is inspected, transformed, and modeled to extract meaningful insights. Once this is done, the information can be used to drive progress and support data-driven decision-making.

Python has become the leading programming language for data analysis and is widely used in data science due to its simplicity and powerful libraries such as Pandas and Numpy for data extraction and Matplotlib, Seaborn for the visualization and much more. In this assignment, we will use Jupyter notebook to write the code and analyze our dataset effectively.

As part of the course *Data Quality and Data Wrangling*, I will practice analyzing a dataset, summarizing its main characteristics, and applying visualization techniques to uncover patterns, relationships, and insights. For this purpose, I have selected the regression dataset **"**California Housing Prices" from Kaggle, which explores the factors influencing housing prices in California.

To begin, I will examine and clean the dataset to ensure its quality for analysis. Once the data is prepared, I will perform an in-depth analysis and create visualizations to identify trends and explore the relationships between various features. Python will serve as the primary tool for this process, enabling the use of visualization libraries such as Matplotlib, Seaborn, and Plotly to produce clear, informative, and engaging charts that reveal key patterns and insights within the data.

Please note that the full project and code can be found on my GitHub repository using the following link: https://github.com/toukoalvine/Tidying-up-the-dataset-California-Housing-Prices-

# 1. Data Analysis

## 1.1. Data information

To understand the dataset, it was loaded and its structure inspected using *df.head(), df.dtypes*, and *df.info()*. The dataset consists of 22 columns and a total of 20,640 rows. The data types include 9 float features and object-type features.

The dataset was checked for data formats, data types, and column names, all of which were found to be correct. The column names are self-explanatory, and the following are the features included in the dataset:

- **Longitude**: a measure of how far west a house is; a higher value is farther west
- **latitude**: A measure of how far north a house is; a higher value is farther north
- **housing Median Age**: Median age of a house within a block; a lower number is a newer building
- **total Rooms**: Total number of rooms within a block
- **total Bedrooms**: Total number of bedrooms within a block
- **population**: Total number of people residing within a block
- **households**: Total number of households, a group of people residing within a home unit, for a block
- **median Income**: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
- **median House Value**: Median house value for households within a block (measured in US Dollars)
- **ocean Proximity**: Location of the house with respect to ocean/sea.

## 1.2. Location and variability

For the numeric features descriptive statistics can easily be obtained with the following code *df.describe()*. With this, we have the descriptive statistics that summarize the central tendency, dispersion of a dataset distribution, excluding NaN values. For our data, the result are summarized in the table 1 below.

***Table 1:*** *descriptive statistics of the dataset*

| | longitude | latitude | housing_median_age | total_rooms | total_bedrooms | population | households | median_income | median_house_value |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 20640.000000 | 20640.000000 | 20640.000000 | 20640.000000 | 20433.000000 | 20640.000000 | 20640.000000 | 20640.000000 | 20640.000000 |
| **mean** | -119.569704 | 35.631861 | 28.639486 | 2635.763081 | 537.870553 | 1425.476744 | 499.539680 | 3.870671 | 206855.816909 |
| **std** | 2.003532 | 2.135952 | 12.585558 | 2181.615252 | 421.385070 | 1132.462122 | 382.329753 | 1.899822 | 115395.615874 |
| **min** | -124.350000 | 32.540000 | 1.000000 | 2.000000 | 1.000000 | 3.000000 | 1.000000 | 0.499900 | 14999.000000 |
| **25%** | -121.800000 | 33.930000 | 18.000000 | 1447.750000 | 296.000000 | 787.000000 | 280.000000 | 2.563400 | 119600.000000 |
| **50%** | -118.490000 | 34.260000 | 29.000000 | 2127.000000 | 435.000000 | 1166.000000 | 409.000000 | 3.534800 | 179700.000000 |
| **75%** | -118.010000 | 37.710000 | 37.000000 | 3148.000000 | 647.000000 | 1725.000000 | 605.000000 | 4.743250 | 264725.000000 |
| **max** | -114.310000 | 41.950000 | 52.000000 | 39320.000000 | 6445.000000 | 35682.000000 | 6082.000000 | 15.000100 | 500001.000000 |

The table provides information about the number of values (count), the mean, variance, minimum and maximum values, as well as the quartiles (25%, 50%, and 75%) for each numerical feature of the dataset. The mean represents the center of the distribution and takes all observations of the feature into account. For example, the mean of the median_house_value in California is 206855.816 US dollars. The relatively larger standard deviation in the data set indicates that the data points are widely spread out around the mean. Conversely, a smaller standard deviation suggests that the data are tightly clustered around the mean. Quartiles divide the data into four equal parts. The median, also known as the second quartile, corresponds to the 50th percentile and is shown as the 50% value in Table above.

## 1.3. Remove duplicates

Removing duplicates from a dataset is crucial to ensure a more accurate data representation, which is essential for effective data analysis. The code line *print(df.drop_duplicates())* did not remove any rows from the dataset, indicating that there were no duplicates present.

## 1.4. Handle Missing Data

Properly handling missing values helps to maintain the integrity and representativeness of the dataset. I used the codes *print(df.dropna()), print(df.dropna(how='all')) and print(df.dropna(thresh=2))* to investigate missing values in the data. There were 207 missing values in the columns "total_bedrooms". I decided to replace these missing values by the mean using the code line *df1["total_bedrooms"] = df["total_bedrooms"].fillna(df["total_bedrooms"].mean())* in a copy df1 of the original dataset df, to avoid any modification of the original data.

## 1.5. Outliers detection

Knowing how to identify and handle outliers is an important part of data cleaning phase. There are various methods available for this task. For the visual inspection I decided to use box plots, which ensure robust outliers' detection.

The box plot with the following code:

```
plt.figure(figsize=(15, 10))
for i, col in enumerate(df1.columns):
    plt.subplot(5, 5, i + 1)
    sns.boxplot(y=df1[col], data = df1)
    plt.title(col)
plt.tight_layout()
plt.show()
```

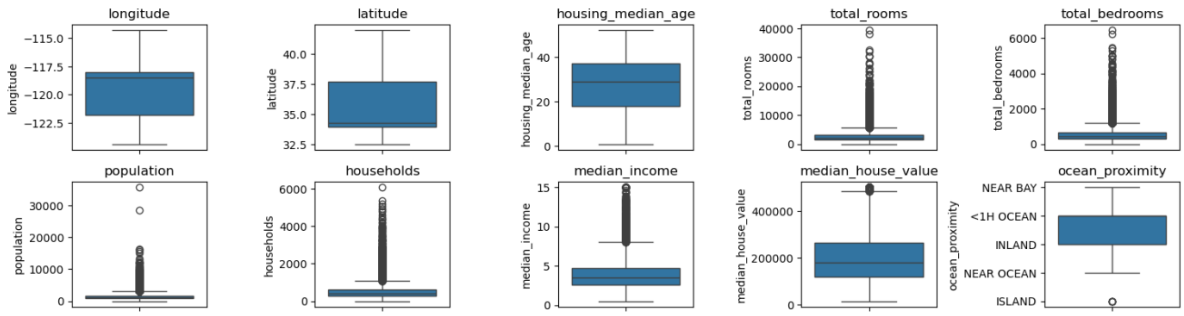


***Figure 1:*** *boxplots of the features*

These plots show that the features: total_rooms, total_bedrooms, population, households, median_income, median_house_value and ocean proximity have outliers. The outliers treatment has been done for each column separately. First, I wrote the code to calculate the first (Q1) and third (Q3) quartiles. The interquartile range (IQR) is then determined as the difference between Q3 and Q1 (IQR = Q3 - Q1). Next, I calculated the upper and lower bounds, and all values outside these fences are considered outliers.

Following is the code for outliers treatment for the feature total_rooms:

```
column_to_analyse = "total_rooms"
#calculate Q1 and Q3
Q1 = df1[column_to_analyse].quantile(0.25)
Q2 = df1[column_to_analyse].quantile(0.5)
Q3 = df1[column_to_analyse].quantile(0.75)
# Calculation of the interquartile range
IQR = Q3 - Q1
#calculate the upper and lower bound for outliers
lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR
print(lower_bound, upper_bound)
#detect outliers
outliers=df1[(df1[column_to_analyse]<lower_bound)|(df1[column_to_analyse]>upper_bound]
#print the number of outliers
print(outliers.shape[0])
```

This has been repeated for all the features with outliers and following is their list and the corresponding number of outliers calculated using the code above:

- total_rooms: 1287 outliers
- total_bedrooms: 1306 outliers
- population: 1196 outliers
- households: 1220 outliers
- median_income: 681outliers
- median_house_value: 1071 outliers

This results in a total of 6761 outliers in the dataset.

## 1.6. Outliers treatment

Since removing all rows with outliers would lead to significant data loss. To handle the outliers, I applied the winsorisation technique using the code below. This approach replaces outlier values with the nearest acceptable limits (lower and upper bounds).

```
column_to_analyse = "total_rooms"
df1["total_rooms"] = df1["total_rooms"].clip(lower=lower_bound, upper=upper_bound)
```

After the transformation, the outlier's number was calculated again with the code below, to make sure that the transformation was effektiv.

```
outliers=df1[(df1["total_rooms"]<lower_bound)|(df1["total_rooms"]>upper_bound)]
print(outliers.shape[0])
```

Also a visual inspection of the data after the transformation has been done by plotting a histogram and a boxplot (see figure 2), using the code:

```
fig, axes = plt.subplots(1, 2)
sns.histplot(x = "total_rooms", data = df1, bins=40, ax=axes[0])
axes[0].set_title("Histogramm of total_rooms")
sns.boxplot(x = "total_rooms", data = df1, ax=axes[1])
axes[1].set_title("Boxplot of total_rooms")
plt.show()
```

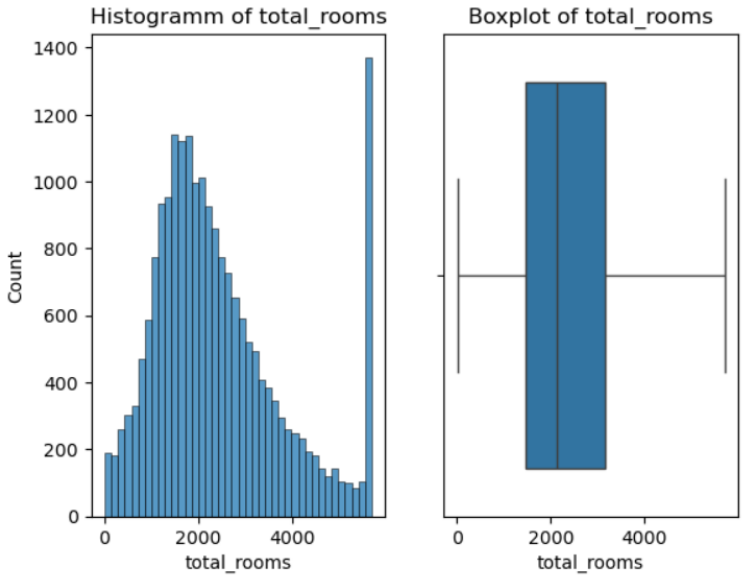


***Figure 2**: Histogram and boxplot of the column "total_rooms" after the winsorisation*

The boxplot shows no data point outside the box, there is no outliers left in the column total_rooms. The elevated count at the upper limit of the histogram is due to the presence of numerous values exceeding the upper bound, which were aggregated into the final bin.

This treatment of the outliers has been repeated for the columns total_bedrooms, population, households, median_income and median_house_value and the visual control after treament was done by plotting the boxplot again, see figure 3.

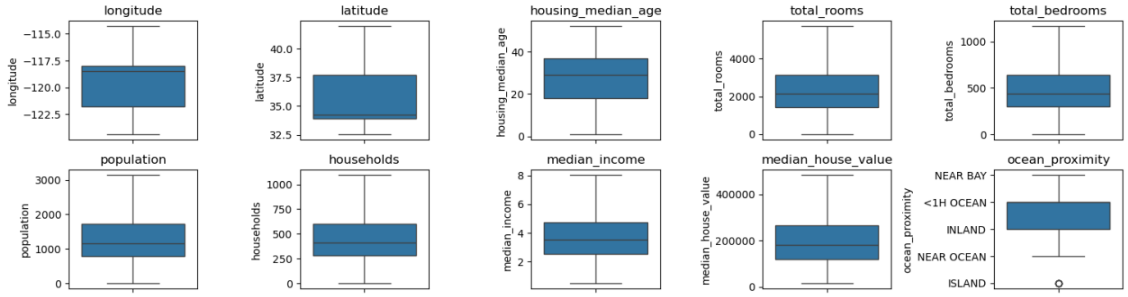


***Figure 3**: inspection of the data after outliers' treatment of all the columns with outliers*

There is no value outside the plot, that means the treatment was effective.

### 1.7. Save the cleaned data

The cleaned data df1 has been saved using the following line *df1.to_csv("modified_df.csv", index=False)* to a csv file. It will be used in the next step for the visualization.

## 2. Data visualization

### 2.1. Histogram of the data

Histograms illustrate the distribution and frequency of values, showing how often certain values occur. This allows characteristics of the distribution to be identified. However, it is important to ensure that the number of bars (bins) in the histogram is appropriately adjusted. It must correspond to the data, otherwise, important information may be lost.

The histograms of the dataset have been plotted with the following code:
*df1.hist(figsize=(15, 10), bins=30, edgecolor="black")*
*plt.subplots_adjust(hspace=0.4, wspace=0.2)*

They are all represented in the figure 4 above.

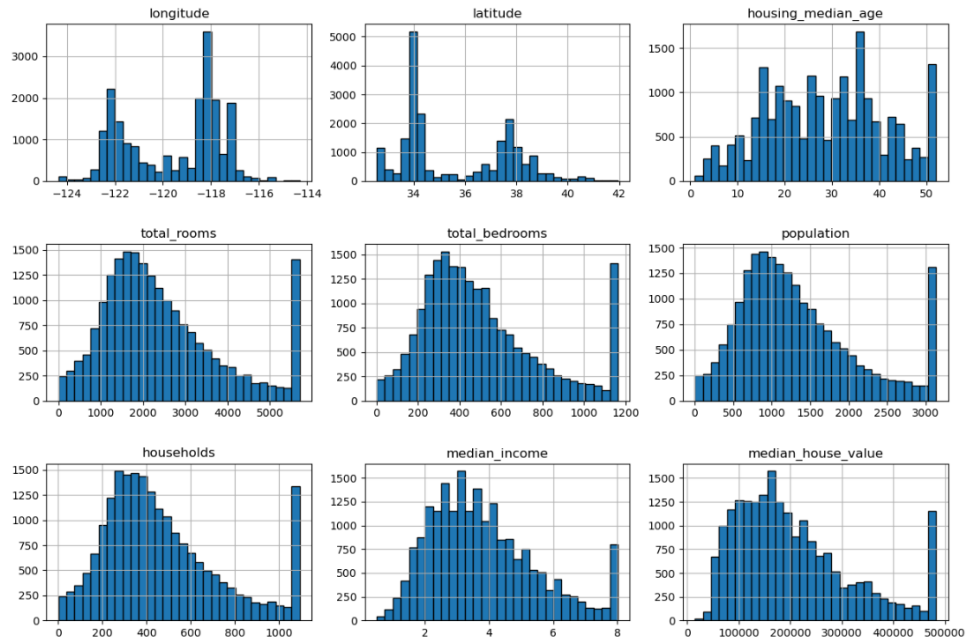


***Figure 4**: Histogram plots for the analysis of distribution*

The histograms of longitude and latitude are not symmetrical and exhibit two peaks (bimodal), while the other seven histograms are unimodal and slightly right-skewed, indicating that more data points are concentrated on the lower end of the range, with the mean greater than the median. The generally high peak at the upper limit of these histograms is due to the aggregation of outliers in the final bin.

## 2.2. Bar chart of the features grouped by ocean_proximity

A bar chart is a good choice when comparing data across different categories. The bars make it easy to visualize the differences at a quick glance. The data have been grouped by proximity to the ocean and the mean of the numeric features has been calculated using the code line: *proximitygroups=df.groupby(by="ocean_proximity").mean(numeric_only=True)*

The mean data are summarized in table 2 below:

***Table** 2: Mean of the housing features grouped by proximity to the ocean*

| ocean_proximity | longitude | latitude | housing_median_age | total_rooms | total_bedrooms | population | households | median_income | median_house_value |
|---|---|---|---|---|---|---|---|---|---|
| **<1H OCEAN** | -118.847766 | 34.560577 | 29.279225 | 2628.343586 | 546.539185 | 1520.290499 | 517.744965 | 4.230682 | 240084.285464 |
| **INLAND** | -119.732990 | 36.731829 | 24.271867 | 2717.742787 | 533.881619 | 1391.046252 | 477.447565 | 3.208996 | 124805.392001 |
| **ISLAND** | -118.354000 | 33.358000 | 42.400000 | 1574.600000 | 420.400000 | 668.000000 | 276.600000 | 2.744420 | 380440.000000 |
| **NEAR BAY** | -122.260694 | 37.801057 | 37.730131 | 2493.589520 | 514.182819 | 1230.317467 | 488.616157 | 4.172885 | 259212.311790 |
| **NEAR OCEAN** | -119.332555 | 34.738439 | 29.347254 | 2583.700903 | 538.615677 | 1354.008653 | 501.244545 | 4.005785 | 249433.977427 |

And are visualized in a bar chart with the following code:

*fig=px.bar(data_frame=proximitygroups,*

```
    y=["total_rooms","population","total_bedrooms","population","households"],
    barmode="group",height=600,title=" housing features by proximity to the ocean")
fig.show()
```

The figure 5 below shows the bar chart of total_rooms, population, total_bedrooms, population and households by the proximity to the ocean.

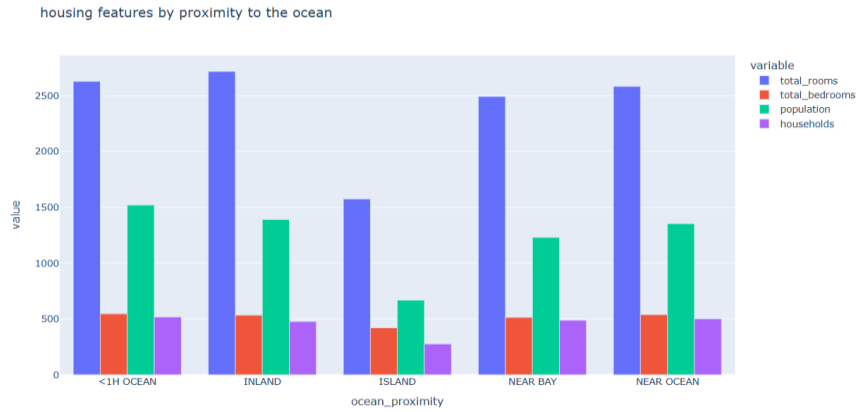


***Figure 5**: bar chart of the mean of housing features by ocean_proximity*

The chart shows that the Inland region has the highest values for total_rooms, total_bedrooms, population, and households, while the Island region has the lowest values for these features.

The bar charts for median_house_value and housing_median_age by proximity to the ocean were plotted separately due to differences in their units, which would make them difficult to visualize on the same graph as total_rooms, total_bedrooms, population, and households. These charts are shown in Figures 6 and 7 below.

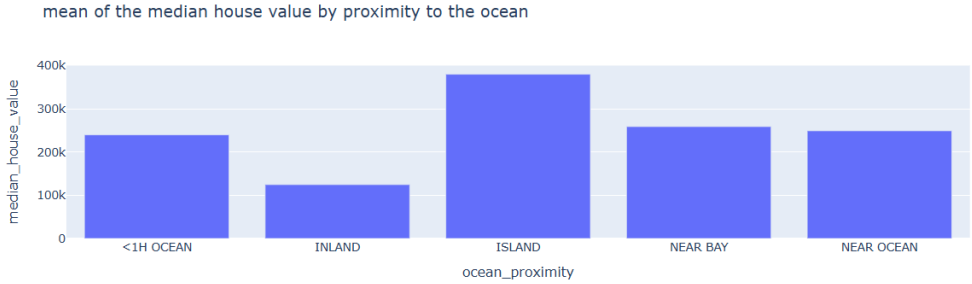


***Figure 6**: bar chart of the mean of median_house_value by ocean_proximity*

The mean of House values is highest in the Island region (380440 US Dollars) and lowest in the Inland region (124805 US Dollars).

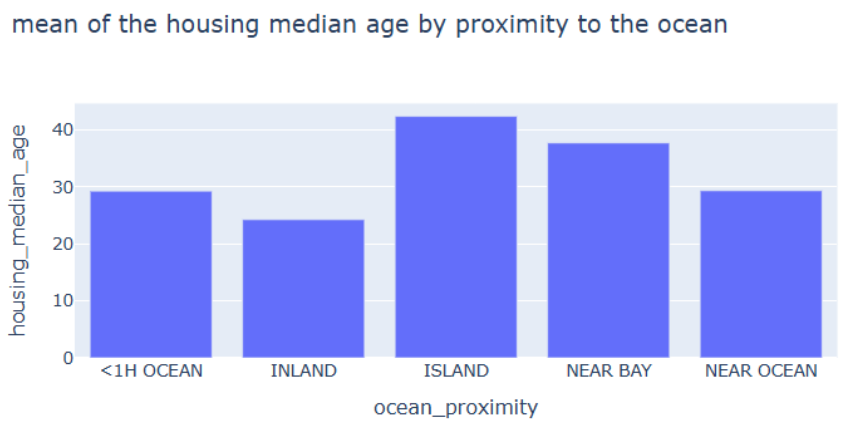


***Figure 7**: bar chart of the mean of housing_median_age by ocean_proximity*

Island region has older houses, with a mean housing_median_age of 42 years, while the Inland region has more new buildings, resulting in a lower mean of 24 years.

## 2.3. Heatmap to visualize the correlation between housing features

A heat map is a two-dimensional representation of data used to investigate the correlation between variables. The correlation coefficients, ranging from -1 to +1, are displayed in the squares and measure the linear relationship between two variables. For the visualizations of correlation between the features, I used seaborn heatmap. It has been plotted using the code:

```
selected_cols =
df1[["longitude","latitude","housing_median_age","total_rooms","total_bedrooms","population"
,"households","median_income","median_house_value"]]
# Calculate the correlation matrix
corr_matrix = selected_cols.corr()
# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f", square=True)
plt.title("Correlation Heatmap of Housing Data")
plt.show()
```

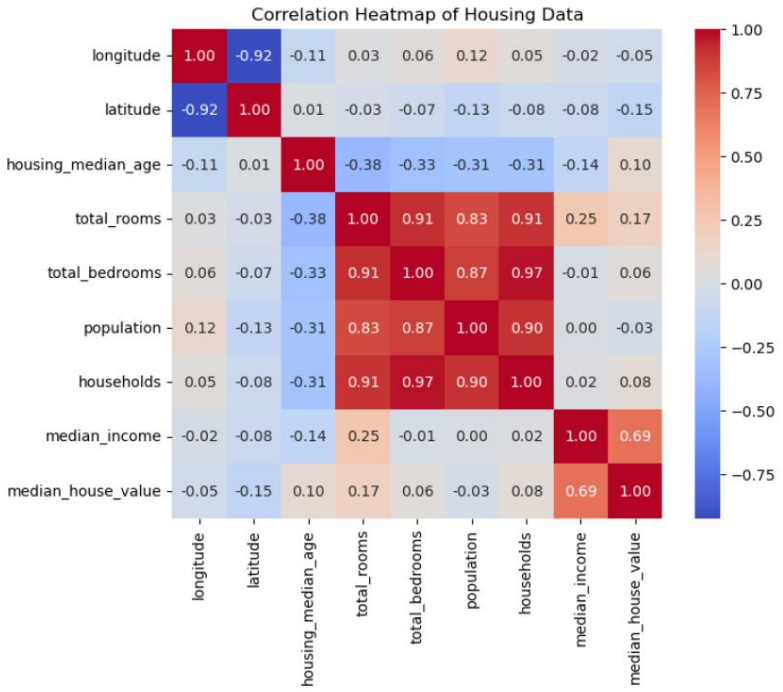


***Figure 8:*** *Heatmap of the housing features*

The heatmap in the figure 8 above shows a strong correlation between the features total_rooms, total_bedrooms, population, and households. This is logical, as more rooms generally imply more bedrooms, which in turn suggests that more people are likely to live in a housing unit. Consequently, this leads to a higher population within the block.
There is also a moderate correlation of 0.69 between median_income and median_house_value.
Additionally, the heatmap indicates that location (longitude and latitude), as well as the features housing_median_age, total_rooms, total_bedrooms, population, and households, do not show any correlation with each other.

## 2.4. Relational plot for further investigation of the correlation

The relational plot in figure 9 illustrates the relationship between the households, which is the total number of households, a group of people residing within a home unit, for a block and total_bedrooms within the block. I then included the contribution of the medie_house_value as an additional variable. The graph was generated using the following code:

```
sns.relplot(data=df1, x="total_bedrooms", y="households", hue="median_house_value")

plt.show()
```

The graph is represented in the figure 9 below:

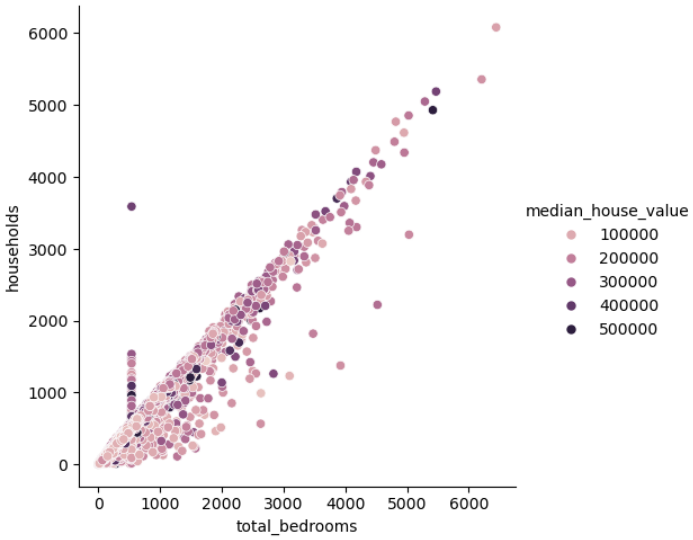


***Figure 9:*** *relational plot between households and total bedrooms*

It can clearly be seen that higher values of total bedrooms within a block are associated with higher numbers of households. The clustering of data points around the upward-sloping correlation line indicates a strong positive correlation of 0.97 between these variables. Additionally, we observe from the graph that median_house_value has no apparent impact on either households or total_bedrooms.

## 2.5. Mapping of the median_house_value with the location

Geospatial data visualization can be easily integrated into Plotly Express to display a dataset feature in relation to its geographic location using a scatter mapbox plot. In this case, I chose to map the median_house_value by location, as location typically has a significant impact on house prices.This has been done with the code:

```
fig = px.scatter_mapbox(df1, lat="latitude", lon="longitude", color="median_house_value", size_max=10,
    zoom=5, mapbox_style="open-street-map")
fig.show()
```

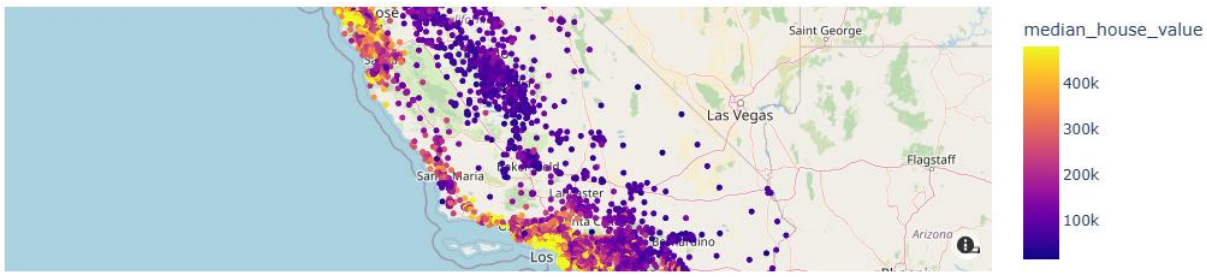


***Figure 10****: mapping of the median house value with the location (latitude and longitude)*

The map in figure 10 shows that high-value houses are predominantly located near the ocean, with property values decreasing as the distance from the coastline increases.

## Conclusion

In summary, this assignment provided an excellent opportunity to apply the concepts learned in Data Quality, Data Wrangling, and Data Visualization. By working with a real-world dataset on California Housing Prices, I practiced essential steps of exploratory data analysis, including data cleaning, and the creation of meaningful visual representations. These steps are crucial not only for understanding the data, but also for effectively communicating insights to a broader audience.

Through systematic data wrangling, including handling missing values, detecting and treating outliers via Winsorization, and removing duplicates, the dataset was prepared for meaningful analysis. Subsequent exploratory data analysis using Python's powerful visualization libraries like Matplotlib, Seaborn, and Plotly revealed valuable insights into the factors influencing housing prices in California.

Key findings include a strong positive correlation among housing features such as total rooms, total bedrooms, households, and population, as well as a moderate correlation between median income and median house value. Geographical analysis showed that house values tend to be higher in areas closer to the ocean and lower inland. Furthermore, the Island region was found to have older and more expensive homes compared to other regions.

Ultimately, the goal of this exercise was not only to practice data visualization techniques but also to deepen my understanding of how data can inform solutions to real-world challenges. This project strengthened my practical skills in exploratory data analysis and visualization using Python. The combination of data wrangling and visualization techniques not only improved the dataset's quality but also uncovered patterns that can support informed decision-making in fields such as real estate and urban planning. Overall, this project highlights the importance of structured, reproducible data analysis processes and showcases Python's power and versatility as a tool for modern data science tasks.

## References

https://www.kaggle.com/datasets/camnugent/california-housing-prices

https://github.com/toukoalvine/Tidying-up-the-dataset-California-Housing-Prices- (GitHub repository)

Ben Bolker. Exploratory data analysis and graphics. Princeton University Press, 2007

Francis John Anscombe. „Graphs in statistical analysis“. The American Statistician 27.1 (1973), S. 17–21

https://maucher.pages.mi.hdm-stuttgart.de/python4datascience/09MachineLearningInaNutshell.html

https://seaborn.pydata.org/generated/seaborn.heatmap.html

McKinney,W.(2012). *Python forDataAnalysis:DataWrangling with Pandas,NumPy,andIPython*. O'ReillyMedia.

Rainer Schnell (1994) Graphisch gestützte Datenanalyse Verlag Oldenbourg, München 1994 ISBN: 978-3-486-23118-2