Alamein International University
Computer Science & Engineering

Real Estate Data Analytics: Empowering Informed Investment Decisions

By

Rawan Badr Saleh Awad (20100239)

Supervised by Mentors

Ayman Akl | ayman.akl@phoenix-mea.com

Abdullah Elashry | abdullah.elashry@phoenix-mea.com

Duration & Location

July 2023: October 2023

Company name: Phoenix Consulting EGY

Location: El Jazeera Tower 1, 4th floor , besides El Sadaat Academy-Maadi–Cairo

Table of Contents

Abstract

The Real Estate Data Analytics project is a complex and multifaceted project that aims to provide investors and developers, among other stakeholders in the real estate industry, with invaluable insights and data-driven decision support. The project covers the entire data lifecycle, starting with data acquisition and thorough cleaning and ending with a thorough and in-depth analysis, with the results presented through user-friendly dashboards. By strategically utilizing the enormous potential inherent in data, the main objective of this initiative is to provide users with the tools and knowledge they need to make informed investment decisions within the complex and ever-evolving real estate market.

The real estate business has not been excluded from this approach to change in an era where data-driven decision-making has emerged as an essential component of success in many industries. The initiative recognizes the vital need for well-informed, evidence-based decisions in the real estate industry, which is marked by big investments, sizeable risks, and a dynamic environment. The Real Estate Data Analytics project aims to close this gap by offering an extensive and well-organized framework that enables stakeholders to effectively manage risks and make well-informed decisions.

The careful procedure of gathering and cleaning data is the foundation of this project. In order to ensure a large and diversified dataset, the acquisition phase entails obtaining data from a variety of sources, such as property databases, market trends, and socioeconomic indicators. After being obtained, the data is subjected to extensive cleaning and quality control procedures to get rid of errors, inconsistencies, and missing data, ensuring the accuracy of the studies that come after.

The project's emphasis on in-depth analysis sets it apart as a valuable resource for stakeholders in the real estate domain. Using data analytics and statistical techniques, the data is subjected to comprehensive examination, uncovering meaningful patterns, trends. These insights go beyond surface-level observations, delving deep into market dynamics, property performance, and investment opportunities. To make the wealth of information accessible and actionable, the project incorporates user-friendly and intuitive dashboards. These dashboards offer a visual representation of the data, providing stakeholders with an efficient means of interpreting the complex information. Users can customize and manipulate the visuals to suit their specific needs, allowing for quick comparisons and informed decision-making.

To sum up, the Real Estate Data Analytics project offers developers, investors, and other industry stakeholders an integrated approach that equips them with the knowledge and skills needed to succeed in the ever-changing and complex real estate market. This effort allows the transition of data into educated decisions, thereby improving the sustainability and success of real estate investments, by precisely managing the whole data lifecycle, from acquisition to visualization.

Introduction

The Real Estate Data Analytics project is a significant and game-changing initiative in the property industry, which is infamous for its complexity and fundamental volatility. It originated from the realization that stakeholders face significant obstacles in the real estate industry when attempting to make well-informed investment decisions. This industry's complexity necessitates an organized, data-driven approach, and this initiative is an effective means designed to meet these problems.

At its core, the project is rooted in the concept of harnessing the capabilities of data analytics and visualization tools to empower participants within the real estate industry. It adopts a systematic and comprehensive approach that spans the entire data lifecycle, commencing with data acquisition and culminating in insightful visualization. This initiative aims to provide invaluable insights and reliable information to investors, developers, and other industry players, ultimately leading to a data-driven transformation in real estate investment decisions.

Due to its complexity and innate volatility, the real estate industry presents difficulties for developers, investors, and other stakeholders. The Real Estate Data Analytics project was developed in this environment. It acknowledges the complexity of real estate markets and the need for organized, data-driven insights to enable stakeholders.

It is impossible to overestimate the importance of data-driven decision-making in the real estate sector, where each decision has significant implications for finances. The Real Estate Data Analytics project is expected to have a major impact because it is utilizing cutting-edge data analytics and visualization tools. It creates a more open and effective investing environment by giving stakeholders the tools they need to confidently and precisely navigate the complex real estate market.

This introduction sets the stage for a comprehensive exploration of the Real Estate Data Analytics project. In the subsequent sections, we will delve into the details of the real estate industry, the pivotal role of data analytics, the specific objectives of this project, and the methodologies employed to achieve its goals. Additionally, we will discuss the project's implications for the industry and the transformative potential it holds for stakeholders.

1. The Complex Landscape of Real Estate Investment

Within the complex field of real estate investment, the industry holds a special and pivotal place in the global economy, making a significant contribution to wealth creation, job creation, and economic growth. Investors looking for opportunities for portfolio diversification and long-term returns have historically been drawn to it. But what sets the real estate market apart from other investment categories are its complexities and difficulties.

The real estate market comprises a wide range of industries, such as retail, commercial, industrial, and residential real estate. Every industry has unique dynamics, consumer preferences, and motivators. Whether local or global, urban or rural, real estate investments require sizable financial commitments and an extended investment horizon. Thus, a thorough comprehension of both the present and the future state of the market is essential.

Numerous factors, such as governmental policies, infrastructure development, societal trends, interest rates, and economic conditions, affect real estate investment decisions. The complicated connection between these variables and the immobility of real estate assets makes making investments a more complex decision. Real estate assets also need to be constantly monitored in order to adjust to the dynamically changing market conditions.

2. The Role of Data in Real Estate Decision-Making

This is an innovation that has affected many industries, including the real estate sector, where data is now the primary factor considered when making decisions. Data has a critical role in influencing investment strategies and decision-making processes, as acknowledged by the Real Estate Data Analytics project. Not only is data a set of figures and statistics, but it's also an invaluable resource that can give stakeholders the knowledge and understanding they need to successfully negotiate the complicated dynamics of the real estate industry.

Data-driven decision-making in real estate involves the systematic collection, analysis, and interpretation of information relevant to the industry. This encompasses a diverse range of data sources, from property databases, market trends, and socio-economic indicators to geographic information systems (GIS) data, property transaction records, and more. The project underscores the importance of compiling a comprehensive dataset that offers a holistic view of the market, capturing the intricate interplay of variables that influence property values and investment opportunities.

Fragmentation, inconsistency, and a lack of structure are common traits of raw data. Before data can be used for decision-making, it needs to go through a thorough cleaning and quality control process. These actions are crucial for guaranteeing the data's consistency and dependability by removing errors, inconsistencies, and missing information. The Real Estate

Data Analytics project emphasizes data integrity as a fundamental concept in the decision-making process and takes an exhaustive approach to data collection and cleaning.

3. Introducing the Real Estate Data Analytics Project

The Real Estate Data Analytics project is conceived as a comprehensive and transformative solution to the multifaceted challenges faced by real estate investors, developers, and other stakeholders. It recognizes that addressing the complexity and volatility of the real estate sector requires a structured framework and access to meaningful data insights. This initiative aims to fulfill this need by facilitating smarter, data-driven decision-making through the application of data analytics and visualization tools.

The core objectives of the Real Estate Data Analytics project can be summarized as follows:

3.1. Data Acquisition and Cleaning

The project starts with the careful process of gathering data, which comes from numerous trustworthy sources. This comprises, among other things, regulations regarding zoning, economic indicators, market trends, demographic data, and databases of past transactions. Making an extensive dataset that faithfully captures the complexities of the real estate market is the goal.

Following the acquisition, the data is subjected to strict cleaning and quality control processes. This entails locating and fixing errors, omissions, and inconsistent data. To guarantee that the data is trustworthy and consistent and provides a solid basis for further analysis, data cleaning is crucial.

3.2. In-Depth Analysis

Data, on its own, has limited value unless it can be transformed into meaningful insights. The Real Estate Data Analytics project utilizes data analytics and statistical techniques to perform comprehensive in-depth analysis of the collected and cleaned data.

The in-depth analysis goes beyond superficial observations and offers stakeholders a deep understanding of market dynamics, property performance, investment opportunities, and emerging trends. These insights are indispensable for decision-makers looking to seize opportunities and mitigate risks within the real estate sector.

3.3. Intuitive Visualization

Understanding complex data is often a daunting task, even for industry professionals. To make the wealth of information accessible and actionable, the Real Estate Data Analytics project incorporates user-friendly and intuitive visualization tools. These tools encompass customizable dashboards, heat maps, geographic information systems (GIS), and 3D visualizations.

The visualization tools enable users to interact with the data, filter information, and explore specific aspects of the real estate market tailored to their needs. This level of interactivity enhances the efficiency of decision-making by providing a clear and holistic view of the real estate landscape.

Real Estate Data Analytics project represents a pivotal step towards a more structured, informed, and efficient approach to real estate investment. It acknowledges and addresses the complexities and challenges inherent to the real estate industry by harnessing the power of data. By offering a systematic framework for data acquisition, rigorous data cleaning processes, in-depth analysis, and intuitive visualization, this project equips stakeholders with the tools and knowledge to thrive in the dynamic and competitive real estate market.

As we delve deeper into this document, we will further explore the project's methodology, significance, and implications for the real estate industry.

4. Methodology of the Real Estate Data Analytics Project

The Real Estate Data Analytics project uses a solid methodology that includes several steps, from data collection to visualization, in order to accomplish its goals. Comprehending this methodology is essential to recognizing the breadth and dependability of the insights this initiative offers.

4.1. Data Collection

The project's first phase involves the methodical gathering of data from numerous trustworthy sources. Public records, property databases, government organizations, real estate listings, market reports, and more are some examples of these sources. The project's ability to fully represent the complexity of the real estate market—which can differ greatly amongst markets, property types, and geographic areas—is ensured by the variety of data sources used.

4.2. Data Cleaning and Preprocessing

Once the data is collected, it undergoes a rigorous cleaning and preprocessing stage. This is essential to eliminate inconsistencies, inaccuracies, and missing data, which can significantly impact the quality of subsequent analyses. Data cleaning involves techniques such as outlier detection, data imputation, and standardization to ensure the data is accurate, reliable, and consistent.

4.3. Exploratory Data Analysis (EDA)

After data preprocessing, the project conducts exploratory data analysis to gain initial insights into the dataset. This involves the use of descriptive statistics, data visualization, and data profiling to identify trends, patterns, and anomalies within the data. EDA helps in understanding the distribution of variables, relationships between variables, and initial observations about the real estate market.

4.4. Interactive Visualization

The insights derived from advanced analytics are presented to users through interactive visualization tools. These tools include customizable dashboards, heat maps, geographic information systems (GIS), and 3D visualizations. Stakeholders can interact with the data, filter information, and explore specific aspects of the real estate market that are relevant to their needs. Visualization tools transform complex data into easily understandable and actionable information.

5. Significance of the Real Estate Data Analytics Project

The significance of the Real Estate Data Analytics project is profound in the context of the real estate industry. It offers several critical benefits that address the pressing needs of investors, developers, and other stakeholders:

5.1. Informed Decision-Making

The project empowers stakeholders with accurate and data-driven insights, allowing them to make informed investment decisions. This is especially crucial in a sector where substantial financial commitments and long-term planning are the norm. Informed decisions help mitigate risks and enhance the success rate of real estate investments.

5.2. Risk Mitigation

Investments in real estate are inherently risky because of changing market conditions and unforeseen circumstances. The analytics of the project offer instruments for evaluating and controlling these risks successfully. Stakeholders can create strategies to protect themselves from price swings and make wise investment decisions by using data-driven risk analysis.

5.3. Market Trend Analysis

Understanding market trends is pivotal for long-term real estate investment. The Real Estate Data Analytics project enables users to analyze trends in property values, demand, and supply. This knowledge allows for strategic positioning in the market and helps investors capitalize on emerging opportunities.

In conclusion, the Real Estate Data Analytics project offers a comprehensive and structured solution for real estate investors, developers, and other industry participants. It acknowledges and addresses the complexities and challenges inherent to the real estate industry by harnessing the power of data. By offering a systematic framework for data acquisition, rigorous data cleaning processes, in-depth analysis, and intuitive visualization, this project equips stakeholders with the tools and knowledge to thrive in the dynamic and competitive real estate market. As this initiative evolves and adapts, it has the potential to revolutionize the way real estate investments are executed, fostering greater transparency, efficiency, and success in this vital economic sector.

Problem Statement

As discussed before in our introduction the real estate sector plays a pivotal role in the global economy, contributing substantially to economic growth, job creation, and wealth generation. In the real estate sector, stakeholders face numerous challenges, the existing challenges can be summarized as follows:

1. Lack of Data-Driven Insights

Traditional real estate investment decisions often rely on limited data and intuition. Investors and developers frequently face the daunting task of navigating a complex and dynamic market landscape without the benefit of comprehensive, data-driven insights. As a result, decisions may be suboptimal, leading to missed opportunities and unnecessary risks.

2. Market Volatility

The real estate market is known for its natural risk, which may vary quickly due to a range of outside factors such as the state of the economy, natural disasters, and world events. It is difficult for stakeholders to recognize profitable investment opportunities and efficiently reduce risks as a result of these sudden changes in the market. It is clear that timely and accurate insights are required to navigate these changes.

3. Information Overload

Data from a wide range of sources, such as property listings, market reports, government regulations, and economic indicators, is flooded into the real estate sector. Even though there is a lot of data available, stakeholders are frequently overwhelmed, which makes it challenging to obtain useful and applicable insights. It becomes a difficult task to combine this enormous amount of data into a logical framework for making decisions.

In our case here our supposed the problem at hand is to perform comprehensive data analysis on web-scraped data from various real estate websites on real estate properties of Dubai. This analysis aims to provide insights into the current state of the real estate market, including property prices, location trends, demand, and supply dynamics, to aid stakeholders in making informed decisions. The specific challenges and components of this problem can be broken down in our proposed solution below.

Proposed Solution

To address these challenges and empower stakeholders with the tools necessary for informed decision-making, the proposed Real Estate Data Analytics project leverages several key processes:

1. **Data Collection**

- Source Selection: This step involves identifying and selecting the real estate websites from which data will be scrapped. It's important to choose reputable and well-maintained sources.
- Scraping Process: Develop web scraping scripts or tools to extract data from these websites. This may involve scraping information like property listings, prices, location details, property types, and other relevant data.
- Data Freshness: Ensuring the collected data is up-to-date is crucial. Frequent scraping or data source APIs may be used to maintain data freshness.

2. **Data Cleaning**

- Handling Missing Data: Identify and deal with missing data points, either by imputing values or deciding on a strategy to handle records with missing information.
- Data Validation: Verify the integrity and accuracy of the scraped data. Data validation checks can help in identifying erroneous entries.
- Standardization: Standardize data formats and units. For instance, ensure that property sizes are consistently represented in square meters or square feet.

3. **Data Integration**

- **Merging Data:** Combine data from various sources into a single dataset. This may involve resolving differences in data formats or schema.
- **Schema Consistency:** Create a consistent schema for the integrated dataset, ensuring that data fields are labeled uniformly.

4. **Exploratory Data Analysis (EDA)**

    - **Descriptive Statistics:** Calculate key statistics such as mean, median, and standard deviation for property prices and other relevant attributes.
    - **Data Visualization**: Create charts and graphs to visualize the distribution of property prices, property types, and other variables. Histograms, box plots, and scatter plots are commonly used.
    - **Outlier Detection:** Identify and deal with outliers, as they can skew the analysis. Outliers might represent erroneous data or unique properties.

5. **Market Segmentation**

- **Property Types:** Categorize properties into different types, such as residential, commercial, or luxury. Analyze each category separately.
- **Geographic Segmentation:** Analyze trends in different neighborhoods or regions of Dubai. This can include identifying areas with high demand, growth potential, or unique characteristics.

6. **Reporting and Visualization**

    - **Data Presentation:** Create clear and informative reports summarizing the findings. Interactive dashboards can be valuable for visualizing data trends and sharing insights.
    - **Actionable Recommendations:** Provide practical recommendations based on the analysis, such as where to invest, areas with potential for property development, or market entry points.

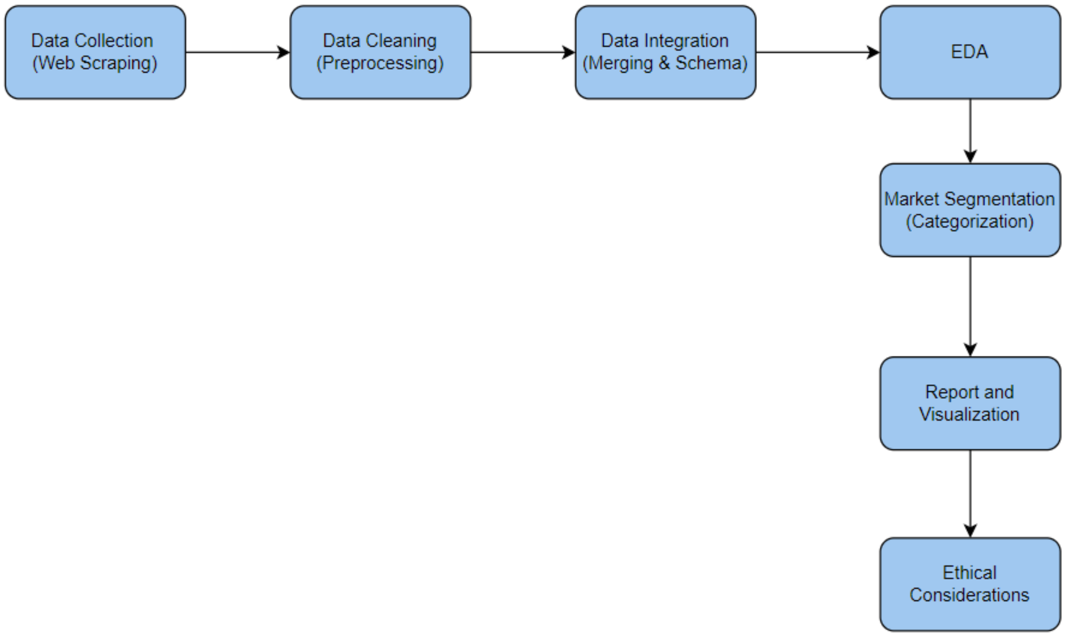
7. **Ethical Considerations**

- **Legal Compliance:** Ensure that web scraping activities comply with relevant laws, such as data protection regulations and terms of service of the scraped websites.
- **Privacy Protection:** Safeguard any personal or sensitive data that might be inadvertently collected during scraping, and be mindful of individuals' privacy rights.

A solid data analysis solution for Dubai's real estate market will be created by carefully and methodically addressing these issues. This will provide stakeholders in this dynamic industry important insights and enable them to make well-informed decisions.

Solution

Before discussing the solution in detail, I will first present the entire architecture's pipelines, outlining all the steps involved. Then, I will delve into each step individually, providing comprehensive details for a clear understanding.


Data Collection (Web Scraping)
Data Cleaning (Preprocessing)
Data Integration (Merging & Schema)
EDA
Market Segmentation (Categorization)
Report and Visualization
Ethical Considerations

Architecture Pipeline for Real State Analytics Model

In this flowchart:

- Arrows represent the flow of data and processes.
- Each box represents a step or component in the pipeline.
- The flow starts from "Data Collection" on the left and progresses through each step, ultimately ending with "Ethical Considerations" on the down right.

This flowchart provides a visual overview of the data analysis pipeline, from data collection to ethical considerations, demonstrating how each step contributes to the overall process.

In this paper, we will provide an overview of both the high-level design and the low-level design. Subsequently, we will offer a concise explanation in the following sections.

1. **High-Level Design:**

The Real Estate Data Analytics project addresses these challenges by implementing a comprehensive solution:

- **Data Gathering and Integration:** It collects data from diverse sources, including property records, market trends, economic indicators, and more.
- **Data Cleaning and Transformation:** Raw data is cleaned, standardized, and transformed into a structured format suitable for analysis.
- **Data Analysis:** Advanced analytics techniques are applied to identify patterns, trends, and correlations within the real estate market.
- **Data Visualization:** Intuitive dashboards and interactive visualizations are created to present insights in a user-friendly manner.

2. **Low-Level Design:**

At a more granular level, the project involves:

- **Data Collection:** Automated data collection scripts and APIs are employed to gather real-time data.
- **Data Preprocessing:** Data is cleaned using algorithms for outlier detection, missing value imputation, and data transformation.
- **Analytics Models:** Statistical models, machine learning algorithms, and predictive analytics are used for in-depth analysis.
- **Dashboard Development:** Tools like Tableau or Power BI are used to design interactive dashboards.

Data Gathering

This crucial step is the foundation of the entire data analysis process. It involves collecting data from a wide range of diverse sources relevant to the Dubai real estate market. These sources may include property records, market trends, economic indicators, and various other data points. The process is often multifaceted and can encompass the following aspects:

1. **Web Scraping:** Real estate data is collected from a multitude of online sources, such as real estate websites, property listings, and auction sites. Web scraping scripts are employed to extract structured data, including property details, prices, locations, and images.

2. **API Integration:** Some sources may offer Application Programming Interfaces (APIs) that allow for more streamlined and structured data retrieval. These APIs can provide real-time information, such as market trends, transaction history, or property listings, ensuring data freshness.

In our project we gathered our data from various real estate /property websites which were : https://www.propertyfinder.ae/ , https://www.bayut.com/ , and https://www.dubizzle.com/ , which are one of the most popular and reliable trusted real estate website sources to get data from.

Once we defined our resources to get data from we now proceed in our first process which is data gathering and extract data using web scraping, but first what is web scraping to begin with?

Web Scraping:

Web scraping, also known as web harvesting or web data extraction, is a data collection technique used to extract information from websites. It involves automated processes that navigate web pages, retrieve content, and save it in a structured format for further analysis or use. Web scraping plays a pivotal role in the acquisition of data from the vast expanse of the World Wide Web, offering valuable insights and information to individuals and organizations across various domains.

Key Elements of Web Scraping

1. Data Sources: Web scraping is used to collect data from websites, online databases, and other online sources. The data can range from text, images, and links to more complex content like product listings, real estate details, and financial information.

2. Automated Processes: Web scraping utilizes automated tools or scripts to navigate websites, simulate human interaction, and extract data. These tools are designed to crawl websites systematically, following links, and scraping relevant data along the way.

3. Data Extraction: The core of web scraping is data extraction. This process involves identifying specific elements on a web page, such as HTML elements, text, tables, or images, and extracting them into a structured format, often in a data file like CSV, JSON, or a database.

4. **Data Transformation:** Once data is extracted, it may undergo transformation to clean, format, and organize it into a consistent structure suitable for analysis, reporting, or integration with other datasets.

Challenges and Considerations

Web scraping is a powerful tool, but it comes with challenges and ethical considerations, including:

1. Legal Compliance: Web scraping activities must adhere to laws and regulations, including copyright and data protection laws. Some websites also explicitly prohibit web scraping in their terms of service.

2. Ethical Usage: Web scrapers should respect the terms and conditions of websites they scrape and avoid overloading servers with requests, as it can lead to server strain and blockage.

3. Data Quality: The quality and accuracy of scraped data can vary. Data cleaning and validation are often required to address issues like missing data, duplicate entries, and inconsistencies.

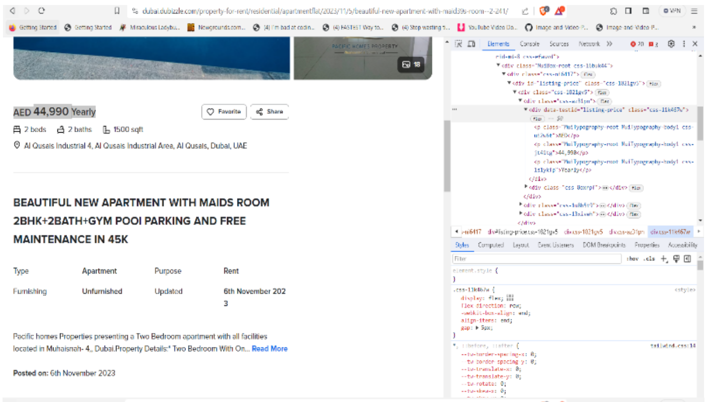
In summary, web scraping is a valuable technique for data collection and analysis, offering a wealth of information from the internet. When used responsibly and ethically, it can provide critical insights for businesses, researchers, and individuals across a wide range of fields.

In our project we go to our targeted websites and ensure that web scraping is allowed by reviewing the website's terms of service and robots.txt file, then we understand our website’s structure by identifying HTML and CSS elements of data to be scrapped.

We then use web scraping libraries the one we used in our project was BeautifulSoup (Python), and to send requests to the website we also used HTTP libraries like requests (Python), we then parse our data to scrape and navigate through it.

We also Handled Pagination and Navigation as the data spans multiple pages, we needed to implement mechanisms to navigate through the website's pagination system or sitemaps.

Finally, in the process of web scraping is to store the extracted data in a structured format such CSV, JSON, a database, or other suitable data storage options.



dubai.dubizzle.com/property-for-rent/residential/apartmentflat/2023/11/5/beautiful-new-apartment-with-maid39s-room-2-241/

AED 44,990 Yearly

2 beds · 2 baths · 1500 sqft

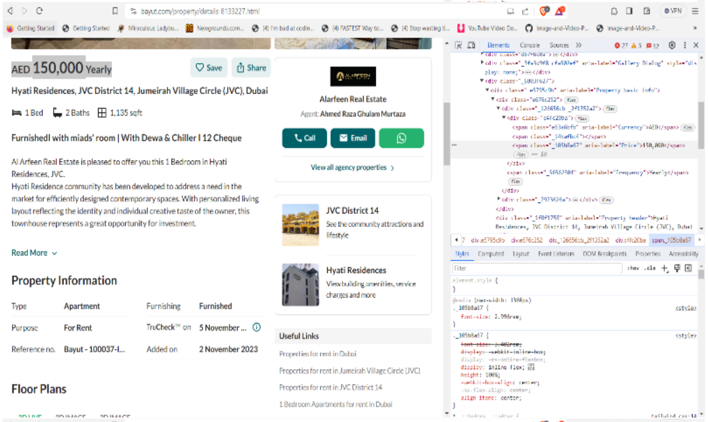
Al Qusais Industrial 4, Al Qusais Industrial Area, Al Qusais, Dubai, UAE

BEAUTIFUL NEW APARTMENT WITH MAIDS ROOM 2BHK+2BATH+GYM POOI PARKING AND FREE MAINTENANCE IN 45K

Type	Apartment	Purpose	Rent
Furnishing	Unfurnished	Updated	6th November 2023

Pacific homes Properties presenting a Two Bedroom apartment with all facilities located in Muhaisnah- 4,, Dubai.Property Details:* Two Bedroom With On... Read More

Posted on: 6th November 2023



bayut.com/property/details-8133227.html

AED 150,000 Yearly

Hyati Residences, JVC District 14, Jumeirah Village Circle (JVC), Dubai

1 Bed · 2 Baths · 1,135 sqft

Furnishedl with miads' room | With Dewa & Chiller I 12 Cheque

Al Arfeen Real Estate is pleased to offer you this 1 Bedroom in Hyati Residences, JVC.
Hyati Residence community has been developed to address a need in the market for efficiently designed contemporary spaces. With personalized living layout reflecting the identity and individual creative taste of the owner, this townhouse represents a great opportunity for investment.

Read More

Property Information

Type	Apartment	Furnishing	Furnished
Purpose	For Rent	TruCheck™ on	5 November ...
Reference no.	Bayut - 100037-l...	Added on	2 November 2023

Floor Plans

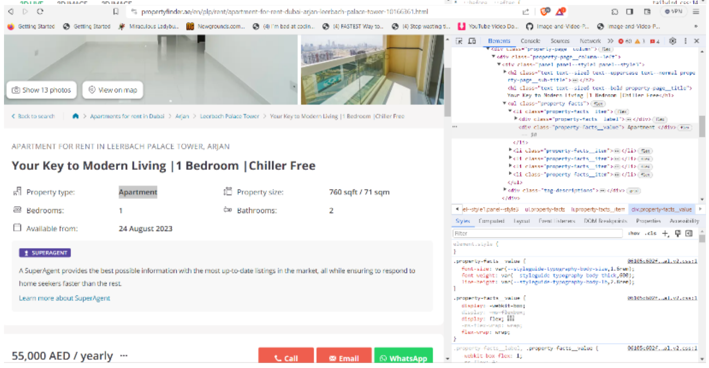
Alarfeen Real Estate
Agent: Ahmed Raza Ghulam Murtaza

View all agency properties

JVC District 14
See the community attractions and lifestyle

Hyati Residences
View building amenities, service charges and more

Useful Links
Properties for rent in Dubai
Properties for rent in Jumeirah Village Circle (JVC)
Properties for rent in JVC District 14
1 Bedroom Apartments for rent in Dubai



propertyfinder.ae/en/plp/rent/apartment-for-rent-dubai-arjan-leerbach-palace-tower-10166361.html

Back to search > Apartments for rent in Dubai > Arjan > Leerbach Palace Tower > Your Key to Modern Living |1 Bedroom |Chiller Free

APARTMENT FOR RENT IN LEERBACH PALACE TOWER, ARJAN

Your Key to Modern Living |1 Bedroom |Chiller Free

Property type:	Apartment	Property size:	760 sqft / 71 sqm
Bedrooms:	1	Bathrooms:	2
Available from:	24 August 2023		

SUPERAGENT

A SuperAgent provides the best possible information with the most up-to-date listings in the market, all while ensuring to respond to home seekers faster than the rest.

Learn more about SuperAgent

55,000 AED / yearly

The above screenshots for websites where we gather and understand the page's data to start extractions.

Data Integration

Data integration is the process of combining data from different sources into a single, unified view or dataset, making it easier to analyze, manage, and extract meaningful insights. Data integration is a critical component of many data-related tasks, such as business intelligence, data warehousing, and data analysis. Here's an in-depth explanation of data integration:

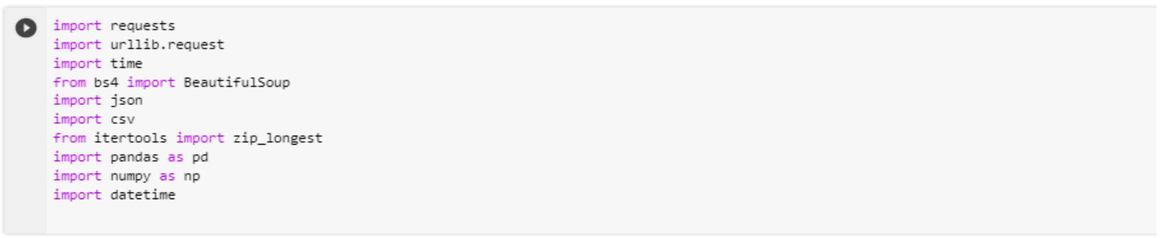
Key Concepts in Data Integration:

1. **Data Sources:** Data integration involves collecting data from various sources, which can include databases, spreadsheets, web services, APIs, flat files, and more. These sources may be structured, semi-structured, or unstructured data.

2. **Data Transformation:** Often, data from different sources come in different formats, structures, and levels of quality. Data transformation is the process of converting and standardizing data so that it can be effectively combined and analyzed. This may involve cleaning, reformatting, and enriching the data.

3. **Data Mapping and Matching:** Data integration typically involves mapping data elements from different sources to corresponding fields in the target dataset. Data matching ensures that similar data points are correctly linked, even when the data source uses variations or discrepancies.

Explanation of what went on in web scraping:



```
import requests
import urllib.request
import time
from bs4 import BeautifulSoup
import json
import csv
from itertools import zip_longest
import pandas as pd
import numpy as np
import datetime
```

1. Library Imports:

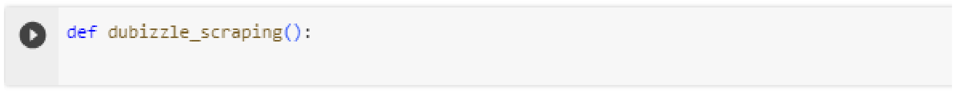
This section of code imports various Python libraries and modules that will be used in the script:

- requests: Used for making HTTP requests to fetch web pages.
- urllib.request: Imported but not used in the code.

- time: Used for timing the scraping process.
- BeautifulSoup from the bs4 library: Used for parsing the HTML content of web pages.
- json: Imported but not used in the code.
- csv: Used for reading and writing CSV files.
- itertools.zip_longest: Used for zipping lists of different lengths together.
- pandas as pd: Used for creating and manipulating DataFrames.
- numpy as np: Imported but not used in the code.
- datetime: Used for tracking the start and end times of the scraping process.

2. Function Definition:

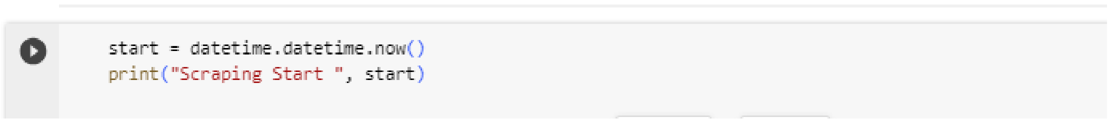
Now, let's move on to the next part of the code:



```
def dubizzle_scraping():
```

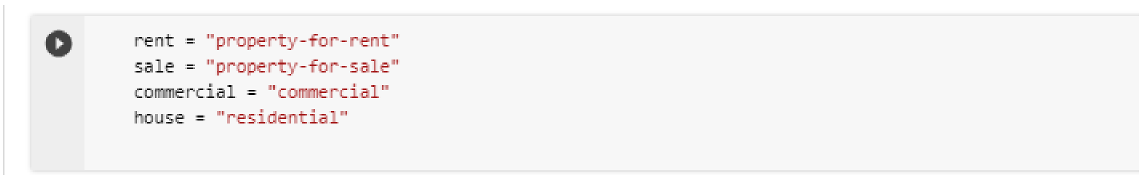
This line defines a Python function named dubizzle_scraping, which will contain the main logic for scraping Dubizzle property data.

3. Start Time and Constants:



```
    start = datetime.datetime.now()
    print("Scraping Start ", start)
```

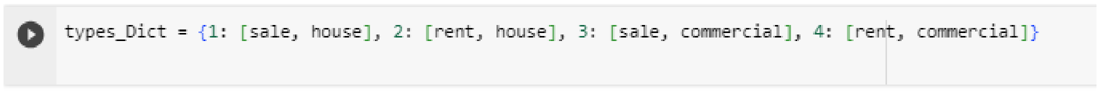
Here, a timestamp is created using datetime.datetime.now() to record the start time of the scraping process. This start time is printed to the console.



```
    rent = "property-for-rent"
    sale = "property-for-sale"
    commercial = "commercial"
    house = "residential"
```

4. **Property Type and Category Mapping**:

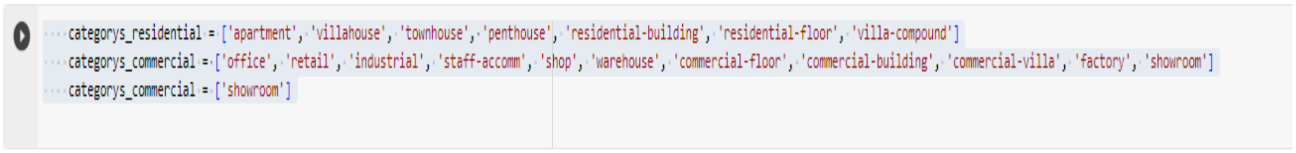
These lines define string variables for different categories of property listings on Dubizzle.



```
types_Dict = {1: [sale, house], 2: [rent, house], 3: [sale, commercial], 4: [rent, commercial]}
```

This dictionary, types_Dict, maps numeric identifiers (1, 2, 3, 4) to pairs of listing types and categories. For example, type 1 corresponds to property for sale in residential areas, and type 3 corresponds to property for sale in commercial areas.

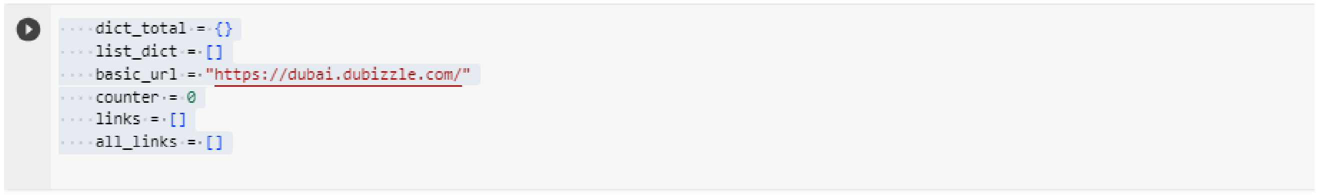
5. **Property Categories Lists:**



```
    categorys_residential = ['apartment', 'villahouse', 'townhouse', 'penthouse', 'residential-building', 'residential-floor', 'villa-compound']
    categorys_commercial = ['office', 'retail', 'industrial', 'staff-accomm', 'shop', 'warehouse', 'commercial-floor', 'commercial-building', 'commercial-villa', 'factory', 'showroom']
    categorys_commercial = ['showroom']
```

These lists define various property categories for both residential and commercial properties. The last assignment of categorys_commercial seems to be redundant, as it overwrites the previous definition.

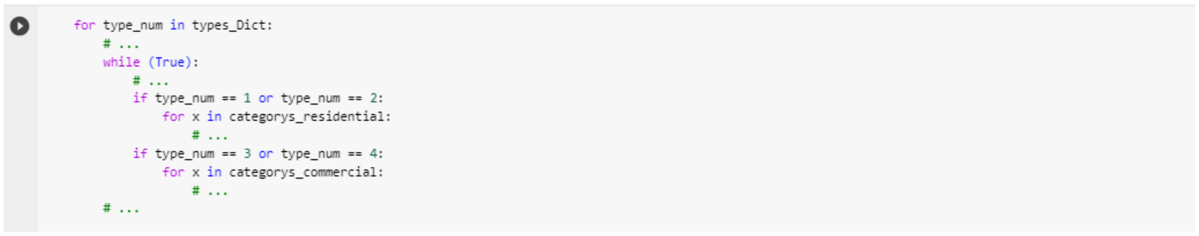
6. **Variables Initialization:**



```
    dict_total = {}
    list_dict = []
    basic_url = "https://dubai.dubizzle.com/"
    counter = 0
    links = []
    all_links = []
```

This group of code initializes several variables and data structures for storing scraped property data, URLs, and other information. For instance, dict_total will store information about individual property listings, and list_dict will collect dictionaries of property data. basic_url is set to the base URL of the Dubizzle website, and counter is used to keep track of the number of properties scraped. links is a list for storing the URLs of property listings.

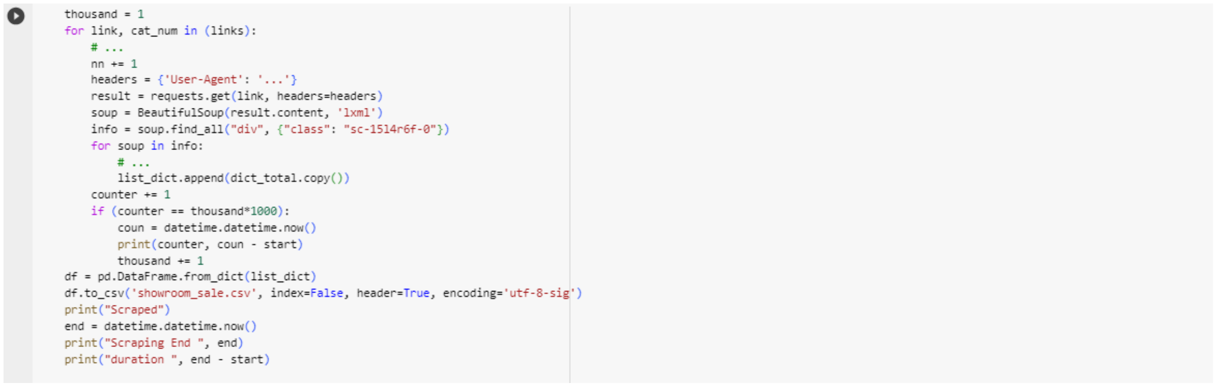
7. **Loop for Scraping Property Listings:**



```
for type_num in types_Dict:
    # ...
    while (True):
        # ...
        if type_num == 1 or type_num == 2:
            for x in categorys_residential:
                # ...
        if type_num == 3 or type_num == 4:
            for x in categorys_commercial:
                # ...
    # ...
```

In this block of code, the script iterates over different combinations of property types and categories and makes HTTP requests to Dubizzle to fetch property listings for those combinations. It collects URLs for each listing and continues to the next page until there are no more pages to scrape.

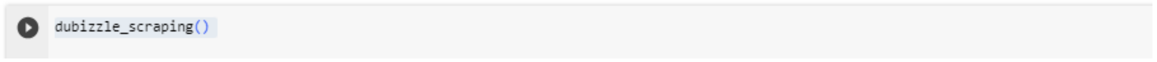
8. Loop for Scraping Individual Property Listings:



```
thousand = 1
for link, cat_num in (links):
    # ...
    nn += 1
    headers = {'User-Agent': '...'}
    result = requests.get(link, headers=headers)
    soup = BeautifulSoup(result.content, 'lxml')
    info = soup.find_all("div", {"class": "sc-1514r6f-0"})
    for soup in info:
        # ...
        list_dict.append(dict_total.copy())
    counter += 1
    if (counter == thousand*1000):
        coun = datetime.datetime.now()
        print(counter, coun - start)
        thousand += 1
df = pd.DataFrame.from_dict(list_dict)
df.to_csv('showroom_sale.csv', index=False, header=True, encoding='utf-8-sig')
print("Scraped")
end = datetime.datetime.now()
print("Scraping End ", end)
print("duration ", end - start)
```

In this part, the script loops through the collected URLs of property listings and extracts detailed information about each listing. This includes data such as URL, unit, price, city, location, address, bedrooms, bathrooms, and size. The extracted data is appended to the list_dict list. The script also tracks the progress and prints information every 1000 properties scraped. Finally, it converts the collected data into a Pandas DataFrame, saves it to a CSV file, and prints a message indicating the end of the scraping process.

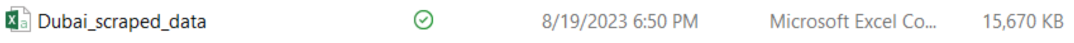
9. Function Invocation:



```
dubizzle_scraping()
```

This line calls the dubizzle_scraping() function, which initiates the entire scraping process when the script is run.

After data extraction and integration concepts has been used on our data we get a CSV we name it Dubai_scraped _data, from our websites merged together in one file containing information which might be informative /useful to our users, stack-holders, or whomever this data concerns.



```
Dubai_scraped_data    ✓    8/19/2023 6:50 PM    Microsoft Excel Co...    15,670 KB
```

Having scraping, extracting and saving our data, we notice when we look through our CSV file that our raw data is in need of some manipulation / fixation to be able for use, which leads us to our next step of this project which is data preprocessing/cleaning and transformation.

But we need to first briefly understand what is data preprocessing/cleaning? why is it important? And how does it affect or its role our project?

Data Cleaning and transformation

Data Cleaning and Transformation is a critical step in the data preparation process. It involves the systematic cleaning, standardization, and restructuring of raw data into a more organized and usable format, which is essential for accurate and meaningful analysis. Here's a detailed discussion of this crucial process:

Data Cleaning and Transformation Process

1. Data Inspection: The process begins with a thorough examination of the raw data. This step involves identifying and understanding the data's structure, content, and potential issues, such as missing values, duplicates, and inconsistencies.

2. Handling Missing Data: Missing data points can significantly impact the quality of analysis. Data cleaning includes strategies for dealing with missing data, such as imputation (filling in missing values using statistical methods) or removing records with missing values.

3. Duplicate Data: Duplicate records can skew results and lead to inaccuracies. Data cleaning involves identifying and removing duplicate entries to ensure that each data point is represented only once

4. Data Standardization: Raw data often comes in various formats, units, and notations. Data cleaning includes standardizing data to ensure uniformity. This may involve converting currencies, dates, and measurements to consistent units.

5. Handling Outliers: Outliers are data points that deviate significantly from the norm. Data cleaning identifies and addresses outliers, which may involve removing them, transforming them, or understanding their significance in the analysis.

6. Data Validation: Data validation checks are implemented to verify the accuracy and integrity of the data. This step aims to identify and rectify inconsistencies and discrepancies that could affect the data's reliability.

7. Data Transformation: Data transformation encompasses processes like aggregating, pivoting, and filtering data to make it more suitable for analysis. It may also involve creating new variables or features that provide additional insights.

8. Normalization and Scaling: In some cases, data cleaning and transformation involve normalizing or scaling variables to ensure that they are on the same scale, which can be particularly important for machine learning algorithms.

9. Data Encoding: Categorical data may need to be encoded into numerical values for analysis. Techniques like one-hot encoding or label encoding are commonly used for this purpose.

Importance of Data Preprocessing/Cleaning and Transformation:

1. Accuracy: Cleaned and transformed data is more accurate and less prone to errors. This accuracy is crucial for making informed decisions and deriving meaningful insights.

2. Consistency: Standardized and well-structured data promotes consistency and facilitates comparisons across different data sources and time periods.

3. Quality Assurance: Data cleaning ensures that data meets quality standards and fulfills requirements, reducing the risk of making decisions based on poor-quality information.

4. Ease of Analysis: Data that is cleaned and transformed is easier to work with, which makes subsequent data analysis and modeling more efficient.

5**. Data Visualization:** Cleaned and structured data is essential for creating accurate and meaningful data visualizations, which aid in better understanding and communication of insights.

6. Ethical and Legal Compliance: Data cleaning plays a role in ensuring that sensitive and private information is appropriately managed, protecting privacy and complying with data protection regulations.

In summary, Data Cleaning and Transformation is a foundational step in the data preparation process. It helps in removing noise, errors, and inconsistencies from raw data, ensuring that the data is accurate, reliable, and in a suitable format for analysis. Clean and well-structured data serves as the basis for informed decision-making and meaningful insights in data-driven projects.

Now knowing what data cleaning is and its importance, in the following sections we'll understand how our Dubai scraped is data is cleaned and transformed.

The code provided is a Python script for data cleaning and transformation of a dataset stored in a CSV file called "Dubai_scraped_data.csv." This script uses the pandas library to load, clean, and transform the data. Here's a breakdown of what the code does:

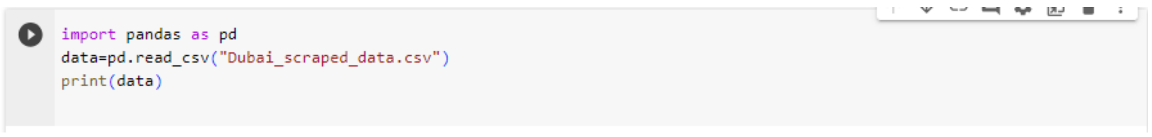
The code you provided is a Python script for data cleaning and transformation of a dataset stored in a CSV file called "Dubai_scraped_data.csv." This script uses the pandas library to load, clean, and transform the data. Here's a breakdown of what the code does:

1. Importing Libraries:

- The script begins by importing the pandas library, which is a powerful library for data manipulation and analysis.

2. Loading the Data:

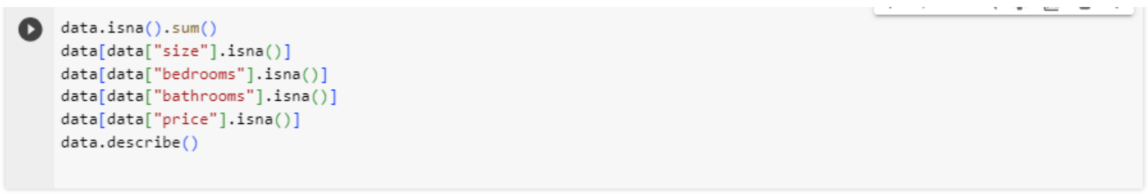
- The code reads the data from the "Dubai_scraped_data.csv" file and stores it in a pandas DataFrame called `data`.
- It then prints the loaded data to inspect its contents.



```python
import pandas as pd
data=pd.read_csv("Dubai_scraped_data.csv")
print(data)
```

3. Missing Value Check:

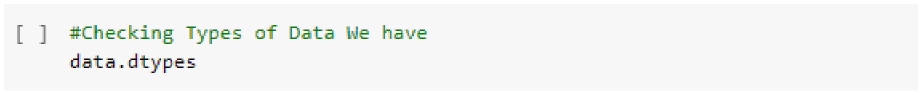
- The code performs a missing value check to identify and count missing values (NaN) in the dataset.
- It then prints subsets of the data where specific columns have missing values (e.g., "size," "bedrooms," "bathrooms," "price").
- The `describe()` method provides summary statistics for numeric columns.



```
data.isna().sum()
data[data["size"].isna()]
data[data["bedrooms"].isna()]
data[data["bathrooms"].isna()]
data[data["price"].isna()]
data.describe()
```

4. Data Type Check:

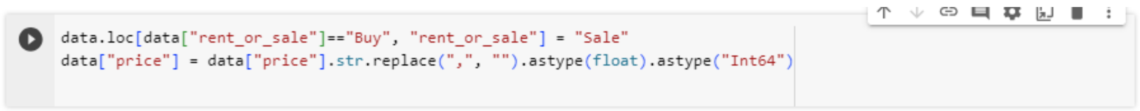
- The code checks the data types of columns in the DataFrame using the `dtypes` attribute.



```
[ ] #Checking Types of Data We have
    data.dtypes
```

5. Data Transformation - Cleaning "rent_or_sale" Column:

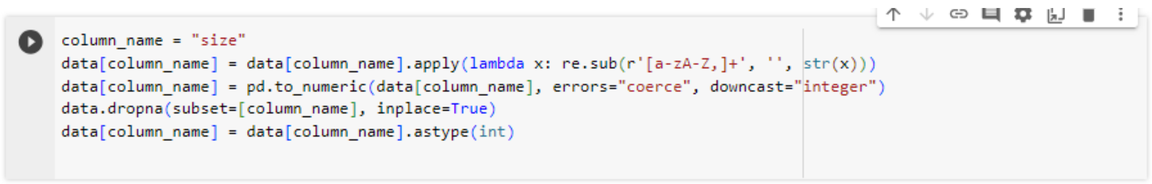
- The script changes values from "Buy" to "Sale" in the "rent_or_sale" column.
- It also removes commas from the "price" column and converts it to an integer data type.



```
data.loc[data["rent_or_sale"]=="Buy", "rent_or_sale"] = "Sale"
data["price"] = data["price"].str.replace(",", "").astype(float).astype("Int64")
```

6. Data Transformation - Cleaning "size" Column:

- The code uses regular expressions to remove non-numeric characters and convert the "size" column to numeric values.
- It then drops rows with missing values in the "size" column.

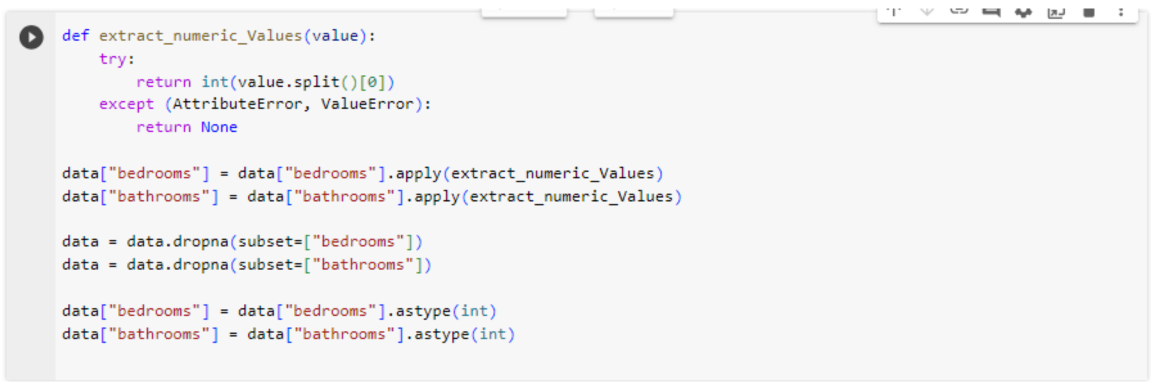


```
column_name = "size"
data[column_name] = data[column_name].apply(lambda x: re.sub(r'[a-zA-Z,]+', '', str(x)))
data[column_name] = pd.to_numeric(data[column_name], errors="coerce", downcast="integer")
data.dropna(subset=[column_name], inplace=True)
data[column_name] = data[column_name].astype(int)
```

7. Data Transformation - Cleaning "bedrooms" and "bathrooms" Columns:

- The script defines a function, `extract_numeric_Values`, to extract numeric values from string representations of bedroom and bathroom counts.

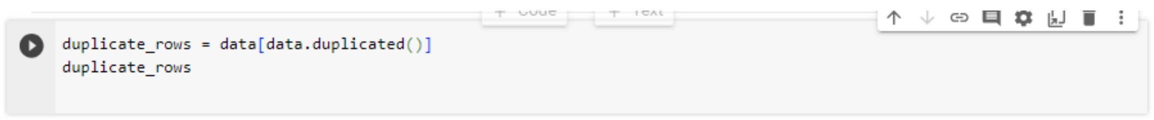
- It applies this function to the "bedrooms" and "bathrooms" columns, converting the values to integers.
- Rows with missing values in these columns are dropped.



```
def extract_numeric_Values(value):
    try:
        return int(value.split()[0])
    except (AttributeError, ValueError):
        return None

data["bedrooms"] = data["bedrooms"].apply(extract_numeric_Values)
data["bathrooms"] = data["bathrooms"].apply(extract_numeric_Values)

data = data.dropna(subset=["bedrooms"])
data = data.dropna(subset=["bathrooms"])

data["bedrooms"] = data["bedrooms"].astype(int)
data["bathrooms"] = data["bathrooms"].astype(int)
```

8. Checking for Duplicates:

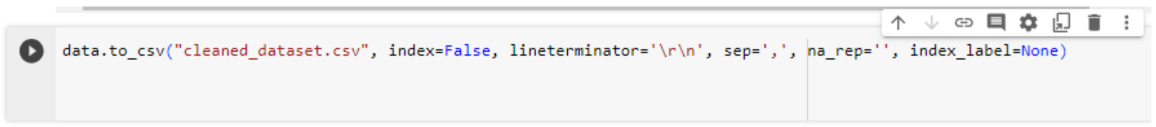
The script checks for and prints duplicate rows in the dataset.



```
duplicate_rows = data[data.duplicated()]
duplicate_rows
```

9. Saving the Cleaned Dataset:

- Finally, the script saves the cleaned dataset to a new CSV file called "cleaned_dataset.csv" with specific formatting options, including using a Windows-style line terminator.

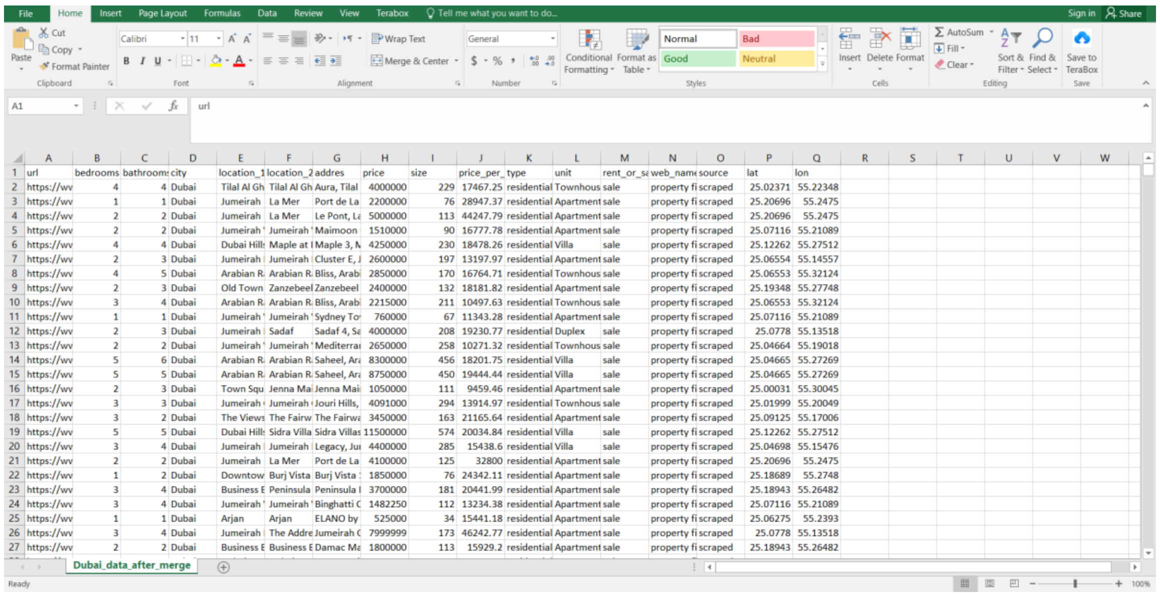


```
data.to_csv("cleaned_dataset.csv", index=False, lineterminator='\r\n', sep=',', na_rep='', index_label=None)
```

The code essentially demonstrates a comprehensive data cleaning and transformation process to improve the quality and structure of the initial dataset before further analysis or modeling.

After this process the new cleaned data set is saved in a file with all the cleaned data ready for further analysis.

cleaned_dataset 8/21/2023 5:58 PM Microsoft Excel Co...



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	url	bedrooms	bathrooms	city	location_1	location_2	addres	price	size	price_per_	type	unit	rent_or_sa	web_name	source	lat	lon
2	https://wv	4	4	Dubai	Tilal Al Gh	Tilal Al Gh	Aura, Tilal	4000000	229	17467.25	residential	Townhous	sale	property fi	scraped	25.02371	55.22348
3	https://wv	1	1	Dubai	Jumeirah	La Mer	Port de La	2200000	76	28947.37	residential	Apartment	sale	property fi	scraped	25.20696	55.2475
4	https://wv	2	2	Dubai	Jumeirah	La Mer	Le Pont, La	5000000	113	44247.79	residential	Apartment	sale	property fi	scraped	25.20696	55.2475
5	https://wv	2	2	Dubai	Jumeirah	Jumeirah	Maimoon	1510000	90	16777.78	residential	Apartment	sale	property fi	scraped	25.07116	55.21089
6	https://wv	4	4	Dubai	Dubai Hills	Maple at I	Maple 3, M	4250000	230	18478.26	residential	Villa	sale	property fi	scraped	25.12262	55.27512
7	https://wv	2	3	Dubai	Jumeirah	Jumeirah	Cluster E, J	2600000	197	13197.97	residential	Apartment	sale	property fi	scraped	25.06554	55.14557
8	https://wv	4	5	Dubai	Arabian R	Arabian R	Bliss, Arabi	2850000	170	16764.71	residential	Townhous	sale	property fi	scraped	25.06553	55.32124
9	https://wv	2	3	Dubai	Old Town	Zanzebeel	Zanzebeel	2400000	132	18181.82	residential	Apartment	sale	property fi	scraped	25.19348	55.27748
10	https://wv	3	4	Dubai	Arabian R	Arabian R	Bliss, Arabi	2215000	211	10497.63	residential	Townhous	sale	property fi	scraped	25.06553	55.32124
11	https://wv	1	1	Dubai	Jumeirah	Jumeirah	Sydney To	760000	67	11343.28	residential	Apartment	sale	property fi	scraped	25.07116	55.21089
12	https://wv	2	3	Dubai	Jumeirah	Sadaf	Sadaf 4, Sa	4000000	208	19230.77	residential	Duplex	sale	property fi	scraped	25.0778	55.13518
13	https://wv	2	2	Dubai	Jumeirah	Jumeirah	Mediterra	2650000	258	10271.32	residential	Townhous	sale	property fi	scraped	25.04664	55.19018
14	https://wv	5	6	Dubai	Arabian R	Arabian R	Saheel, Ara	8300000	456	18201.75	residential	Villa	sale	property fi	scraped	25.04665	55.27269
15	https://wv	5	5	Dubai	Arabian R	Arabian R	Saheel, Ara	8750000	450	19444.44	residential	Villa	sale	property fi	scraped	25.04665	55.27269
16	https://wv	2	3	Dubai	Town Squ	Jenna Mai	Jenna Mai	1050000	111	9459.46	residential	Apartment	sale	property fi	scraped	25.00031	55.30045
17	https://wv	3	3	Dubai	Jumeirah	Jumeirah	Jouri Hills,	4091000	294	13914.97	residential	Townhous	sale	property fi	scraped	25.01999	55.20049
18	https://wv	3	2	Dubai	The Views	The Fairw	The Fairwa	3450000	163	21165.64	residential	Apartment	sale	property fi	scraped	25.09125	55.17006
19	https://wv	5	5	Dubai	Dubai Hills	Sidra Villa	Sidra Villas	11500000	574	20034.84	residential	Villa	sale	property fi	scraped	25.12262	55.27512
20	https://wv	3	4	Dubai	Jumeirah	Jumeirah	Legacy, Jui	4400000	285	15438.6	residential	Villa	sale	property fi	scraped	25.04698	55.15476
21	https://wv	2	2	Dubai	Jumeirah	La Mer	Port de La	4100000	125	32800	residential	Apartment	sale	property fi	scraped	25.20696	55.2475
22	https://wv	1	2	Dubai	Downtow	Burj Vista	Burj Vista	1850000	76	24342.11	residential	Apartment	sale	property fi	scraped	25.18689	55.2748
23	https://wv	3	4	Dubai	Business E	Peninsula	Peninsula I	3700000	181	20441.99	residential	Apartment	sale	property fi	scraped	25.18943	55.26482
24	https://wv	3	4	Dubai	Jumeirah	Jumeirah	Binghatti C	1482250	112	13234.38	residential	Apartment	sale	property fi	scraped	25.07116	55.21089
25	https://wv	1	1	Dubai	Arjan	Arjan	ELANO by	525000	34	15441.18	residential	Apartment	sale	property fi	scraped	25.06275	55.2393
26	https://wv	3	4	Dubai	Jumeirah	The Addre	Jumeirah C	7999999	173	46242.77	residential	Apartment	sale	property fi	scraped	25.0778	55.13518
27	https://wv	2	2	Dubai	Business E	Business E	Damac Ma	1800000	113	15929.2	residential	Apartment	sale	property fi	scraped	25.18943	55.26482

We observe from our data in the file the following:

1. Missing Data Analysis:

- We can identify which columns have missing data and the number of missing values in each column. This helps you understand the completeness of the dataset.

2. Data Types:

- We can see that the data types of each column are appropriately represented (e.g., integers, floats, strings).
-

3. Data Cleaning and Standardization:

- The code removes duplicates from the dataset, ensuring that each data point is unique.
- It standardizes the "rent_or_sale" column by replacing "Buy" with "Sale."
- The "price" column is cleaned by removing commas and converting it to an integer data type, making it more suitable for analysis.
- The "size" column is cleaned and converted to numeric values by removing non-numeric characters.

- The "bedrooms" and "bathrooms" columns are cleaned and converted to integers, removing non-numeric characters.

4. Data Distribution and Outliers:

- After the data cleaning and transformation, we can observe the distribution of data in the "price," "size," "bedrooms," and "bathrooms" columns. Analyzing this distribution can help you identify outliers and understand the central tendencies of these attributes.

5. Duplicate Data:

- We can see that duplicate rows in the dataset is removed.

6. Data Integrity:

- By applying various data cleaning and transformation operations, we ensure that the data maintains its integrity and is in a more consistent and accurate state.

7. Data Storage:

- The cleaned data is saved to a new CSV file ("cleaned_dataset.csv"), which can be used for further analysis or shared with others.

Overall, the code helps you assess the quality and reliability of your dataset by addressing missing values, standardizing data, removing duplicates, and transforming data into more suitable formats for analysis. These actions improve the data's integrity and make it ready for more in-depth data analysis, modeling, or reporting.

The next step in our project which is data analysis.This process can help stakeholders, such as real estate professionals, investors, and policymakers, make informed decisions.

We'll now explain this process effect on our project.

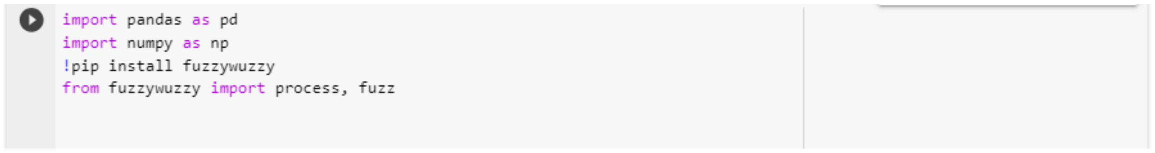
Firstly, we renamed our cleaned data set from the previous step and named it Dubai_data_after_merge to clarify that this is the final cleaned and merged data from resources targeted that we'll be working on.

Dubai_data_after_merge 8/22/202

The provided snippets of code perform several data cleaning and transformation operations on a dataset. Here's a step-by-step explanation of what each part of the code does:

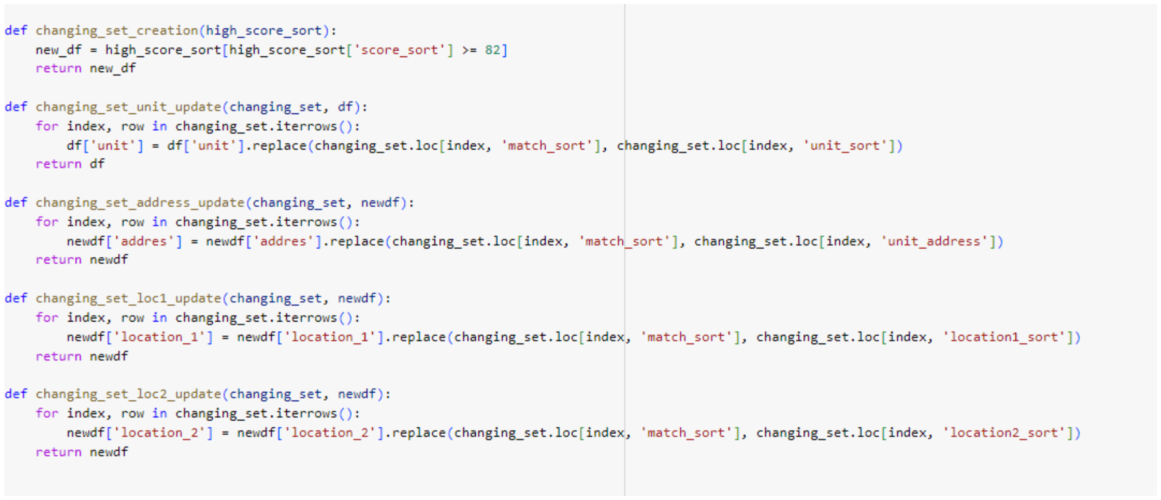
1. Importing Libraries:

- The code starts by importing the necessary Python libraries: pandas, numpy, and fuzzywuzzy. The fuzzywuzzy library is used for string matching and similarity scoring.


```
import pandas as pd
import numpy as np
!pip install fuzzywuzzy
from fuzzywuzzy import process, fuzz
```

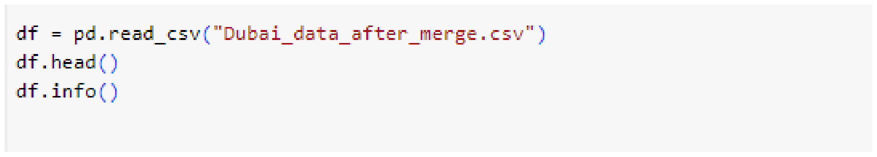
2. Functions for Data Transformation:

- The code defines several functions for updating and cleaning specific columns in the dataset. These functions use the fuzzywuzzy library to perform string matching and replace values based on similarity scores. There are functions for updating the "unit,""address","location_1," "location_2," and "unit" columns.


```
def changing_set_creation(high_score_sort):
    new_df = high_score_sort[high_score_sort['score_sort'] >= 82]
    return new_df

def changing_set_unit_update(changing_set, df):
    for index, row in changing_set.iterrows():
        df['unit'] = df['unit'].replace(changing_set.loc[index, 'match_sort'], changing_set.loc[index, 'unit_sort'])
    return df

def changing_set_address_update(changing_set, newdf):
    for index, row in changing_set.iterrows():
        newdf['addres'] = newdf['addres'].replace(changing_set.loc[index, 'match_sort'], changing_set.loc[index, 'unit_address'])
    return newdf

def changing_set_loc1_update(changing_set, newdf):
    for index, row in changing_set.iterrows():
        newdf['location_1'] = newdf['location_1'].replace(changing_set.loc[index, 'match_sort'], changing_set.loc[index, 'location1_sort'])
    return newdf

def changing_set_loc2_update(changing_set, newdf):
    for index, row in changing_set.iterrows():
        newdf['location_2'] = newdf['location_2'].replace(changing_set.loc[index, 'match_sort'], changing_set.loc[index, 'location2_sort'])
    return newdf
```

3. Loading Data:

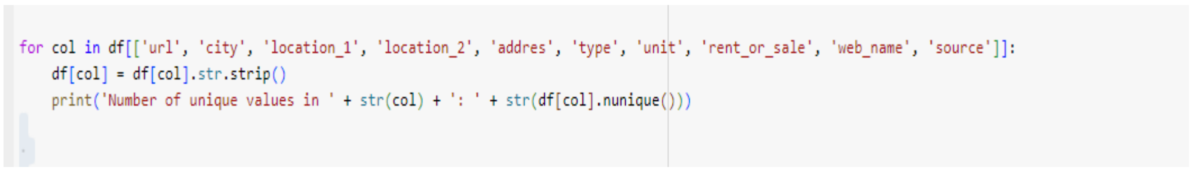
- The code reads a CSV file named "Dubai_data_after_merge.csv" into a pandas DataFrame called `df`.
- It prints the first few rows of the dataset using `df.head()` and provides information about the DataFrame's columns and data types using `df.info()`.



```
df = pd.read_csv("Dubai_data_after_merge.csv")
df.head()
df.info()
```

4. Cleaning Leading and Trailing Spaces:

- A loop iterates through selected columns and removes leading and trailing spaces from the values in these columns. It also counts and prints the number of unique values in each of these columns.

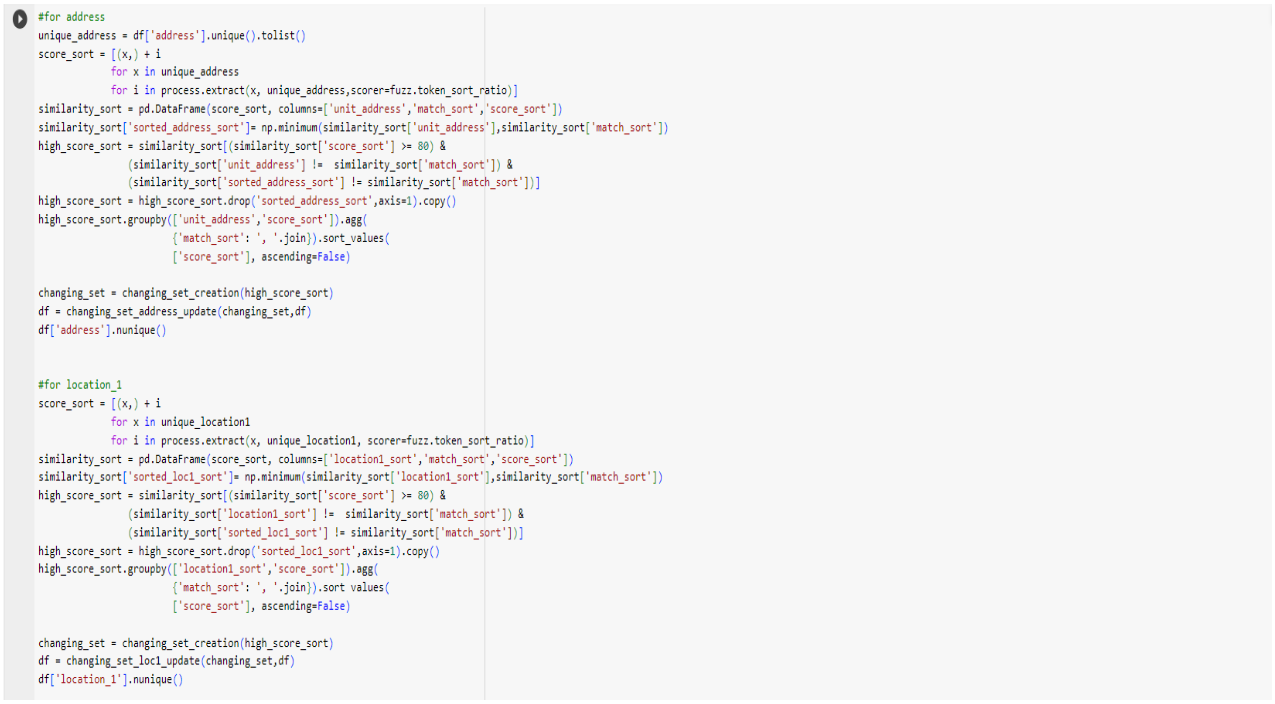


```
for col in df[['url', 'city', 'location_1', 'location_2', 'addres', 'type', 'unit', 'rent_or_sale', 'web_name', 'source']]:
    df[col] = df[col].str.strip()
    print('Number of unique values in ' + str(col) + ': ' + str(df[col].nunique()))
```

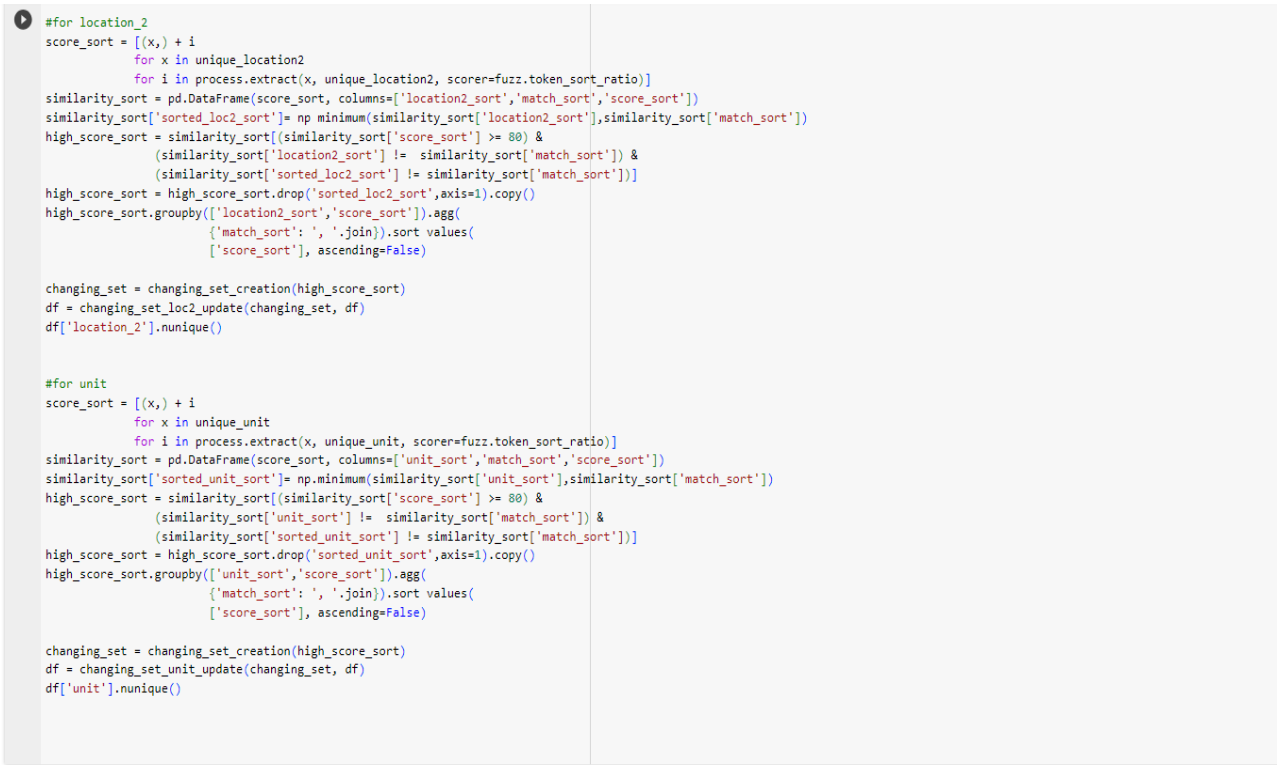
5. String Matching and Updating Columns:

- The code performs a series of operations for different columns by calculating similarity scores between values in the column and then replacing them based on certain conditions. This includes the following steps:
- For "address", "location_1," "location_2," and "unit" columns:
- Similarity scores are calculated between values in the column and stored in a DataFrame called `similarity_sort`.
- High-scoring matches are selected, and the code generates a "changing_set" for each column.
- The functions for updating each column (e.g., `changing_set_unit_update`, `changing_set_address_update`) are called, and the DataFrame `df` is updated based on the "changing_set" for each column.
- The number of unique values in the updated column is printed.

- The goal of these operations is to clean and standardize the values in these columns based on their similarity to other values in the same column.



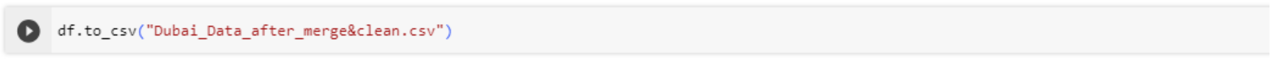
```python
#for address
unique_address = df['address'].unique().tolist()
score_sort = [(x,) + i
             for x in unique_address
             for i in process.extract(x, unique_address,scorer=fuzz.token_sort_ratio)]
similarity_sort = pd.DataFrame(score_sort, columns=['unit_address','match_sort','score_sort'])
similarity_sort['sorted_address_sort']= np.minimum(similarity_sort['unit_address'],similarity_sort['match_sort'])
high_score_sort = similarity_sort[(similarity_sort['score_sort'] >= 80) &
                (similarity_sort['unit_address'] !=  similarity_sort['match_sort']) &
                (similarity_sort['sorted_address_sort'] != similarity_sort['match_sort'])]
high_score_sort = high_score_sort.drop('sorted_address_sort',axis=1).copy()
high_score_sort.groupby(['unit_address','score_sort']).agg(
                        {'match_sort': ', '.join}).sort_values(
                        ['score_sort'], ascending=False)

changing_set = changing_set_creation(high_score_sort)
df = changing_set_address_update(changing_set,df)
df['address'].nunique()


#for location_1
score_sort = [(x,) + i
             for x in unique_location1
             for i in process.extract(x, unique_location1, scorer=fuzz.token_sort_ratio)]
similarity_sort = pd.DataFrame(score_sort, columns=['location1_sort','match_sort','score_sort'])
similarity_sort['sorted_loc1_sort']= np.minimum(similarity_sort['location1_sort'],similarity_sort['match_sort'])
high_score_sort = similarity_sort[(similarity_sort['score_sort'] >= 80) &
                (similarity_sort['location1_sort'] !=  similarity_sort['match_sort']) &
                (similarity_sort['sorted_loc1_sort'] != similarity_sort['match_sort'])]
high_score_sort = high_score_sort.drop('sorted_loc1_sort',axis=1).copy()
high_score_sort.groupby(['location1_sort','score_sort']).agg(
                        {'match_sort': ', '.join}).sort values(
                        ['score_sort'], ascending=False)

changing_set = changing_set_creation(high_score_sort)
df = changing_set_loc1_update(changing_set,df)
df['location_1'].nunique()
```



```
#for location_2
score_sort = [(x,) + i
             for x in unique_location2
             for i in process.extract(x, unique_location2, scorer=fuzz.token_sort_ratio)]
similarity_sort = pd.DataFrame(score_sort, columns=['location2_sort','match_sort','score_sort'])
similarity_sort['sorted_loc2_sort']= np minimum(similarity_sort['location2_sort'],similarity_sort['match_sort'])
high_score_sort = similarity_sort[(similarity_sort['score_sort'] >= 80) &
                (similarity_sort['location2_sort'] !=  similarity_sort['match_sort']) &
                (similarity_sort['sorted_loc2_sort'] != similarity_sort['match_sort'])]
high_score_sort = high_score_sort.drop('sorted_loc2_sort',axis=1).copy()
high_score_sort.groupby(['location2_sort','score_sort']).agg(
                        {'match_sort': ', '.join}).sort values(
                        ['score_sort'], ascending=False)

changing_set = changing_set_creation(high_score_sort)
df = changing_set_loc2_update(changing_set, df)
df['location_2'].nunique()


#for unit
score_sort = [(x,) + i
             for x in unique_unit
             for i in process.extract(x, unique_unit, scorer=fuzz.token_sort_ratio)]
similarity_sort = pd.DataFrame(score_sort, columns=['unit_sort','match_sort','score_sort'])
similarity_sort['sorted_unit_sort']= np.minimum(similarity_sort['unit_sort'],similarity_sort['match_sort'])
high_score_sort = similarity_sort[(similarity_sort['score_sort'] >= 80) &
                (similarity_sort['unit_sort'] !=  similarity_sort['match_sort']) &
                (similarity_sort['sorted_unit_sort'] != similarity_sort['match_sort'])]
high_score_sort = high_score_sort.drop('sorted_unit_sort',axis=1).copy()
high_score_sort.groupby(['unit_sort','score_sort']).agg(
                        {'match_sort': ', '.join}).sort values(
                        ['score_sort'], ascending=False)

changing_set = changing_set_creation(high_score_sort)
df = changing_set_unit_update(changing_set, df)
df['unit'].nunique()
```

6. Saving the Cleaned Data:

- After all the cleaning and transformation operations, the code saves the cleaned dataset to a new CSV file named "Dubai_Data_after_merge&clean.csv."



```
df.to_csv("Dubai_Data_after_merge&clean.csv")
```

The code focuses on improving data quality and standardizing values in specific columns based on string similarity. It uses the fuzzywuzzy library for matching and scoring, helping to ensure consistency and accuracy in the dataset.

From the above data analysis, we notice some changes in our datasets which are as follows:

1. **Standardized Data:**

By using fuzzy string matching techniques, we've standardized the values in columns like 'address,' 'location_1,' 'location_2,' and 'unit.' This ensures that similar entries are represented consistently, which can be essential for accurate analysis and reporting.

2. **Reduced Data Redundancy**:

The code identified and reduced redundancy in the data. Similar entries with slight variations are replaced with a common representation, which can lead to a cleaner and more efficient dataset.

3. **Enhanced Data Quality:**

The cleaning and transformation steps, such as removing leading and trailing spaces, improve data quality. Clean data is less prone to errors and is easier to work with in subsequent data analysis tasks.

4. **Unique Value Count:**

The code provides the number of unique values in each column after the cleaning and transformation operations. This information is useful for understanding the diversity and uniqueness of data in each column.

5. **String Matching and Score Filtering:**

The code calculates similarity scores and identifies high-scoring matches for certain columns. By applying a minimum score threshold (e.g., 80), we've filtered out less confident matches and focused on high-quality matches.

6. **Creation of 'Changing Set'**:

A "changing set" is created for each column based on high-scoring matches. This set contains pairs of original values and standardized values, which can be valuable for auditing and reference.

7.Data Integrity: These operations enhance data integrity by reducing discrepancies and inconsistencies in the dataset. This can lead to more reliable and accurate analysis results.

8. **Simplified Data Analysis:**

The cleaned and standardized data is now more suitable for various data analysis tasks, such as aggregation, reporting, and modeling, as it reduces noise and enhances the signal in the dataset.

9. **Reduced Data Entry Errors:**

   The standardization process reduces the potential for data entry errors by providing a consistent format for values in the specified columns.

In summary, the code we've executed on our data has significantly improved its quality, consistency, and usability. These improvements can lead to more reliable and accurate insights when conducting data analysis and other tasks with the cleaned dataset.

After our previous data analysis, we'll continue our data analysis on the last saved dataset, for this step we are going to do EDA (Exploratory Data Analysis) but first we need to understand what is EDA?

EDA stands for Exploratory Data Analysis. It is an essential and often the first step in the data analysis process. EDA is the process of summarizing the main characteristics of a dataset, often with the help of statistical graphics and plots. Its primary purpose is to gain a better understanding of the data, identify patterns, detect outliers, and test assumptions.

Here are some key aspects of Exploratory Data Analysis:

1. **Data Summarization:**

EDA involves summarizing the data through various statistical measures, such as mean, median, variance, and standard deviation. These measures provide an overview of the dataset's central tendencies and dispersion.

2. **Data Visualization:**

EDA uses data visualization techniques to represent the data graphically. This includes histograms, box plots, scatter plots, bar charts, and more. Visualizations help reveal patterns, trends, and outliers that may not be apparent from the raw data.

3. **Uncovering Patterns:**

EDA aims to uncover patterns within the data. For instance, it can reveal seasonality in time series data, clusters in data points, or correlations between variables.

4. **Identifying Outliers:**

Outliers are data points that deviate significantly from the majority of the data. EDA helps in detecting these outliers, which may be errors or points of interest.

5. **Handling Missing Data**:

EDA involves dealing with missing data points by either imputing missing values or excluding data with missing values, depending on the nature and extent of the missing data.

6. **Testing Assumptions:**

EDA tests assumptions made about the data, such as the normality of distributions or the independence of variables. Violation of these assumptions can impact subsequent data analysis.

7. **Feature Selection:**

In the context of machine learning and predictive modeling, EDA helps in selecting relevant features (variables) and understanding their relationships with the target variable.

8. **Data Cleaning:**

EDA often reveals data quality issues such as duplicate records, inconsistent data formats, and data entry errors. Addressing these issues is a crucial part of the EDA process.

9. **Hypothesis Generation:**
   EDA may lead to the generation of hypotheses or research questions for more in-depth analysis. It helps in guiding further investigations.

10. **Communication:**

 EDA is a valuable tool for communicating data insights to stakeholders. Visualizations and summaries can convey complex information in a more accessible way.

11. **Iterative Process:**

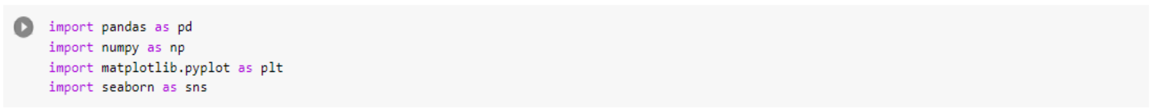
EDA is often an iterative process. As we uncover new insights and ask more questions, you may need to revisit and refine your EDA techniques.

EDA is not a one-size-fits-all process; it varies depending on the dataset, the research goals, and the tools and techniques used. It is a fundamental step in data analysis, providing the foundation for more advanced statistical and machine learning methods.

The provided snippets of code perform several EDA operations on a dataset. Here's a step-by-step explanation of what each part of the code does:

1. **Importing Libraries:**

- Here, the code imports necessary Python libraries:
- pandas for data manipulation and analysis.
- numpy for numerical operations.
- matplotlib for creating data visualizations.
- seaborn for enhancing the quality of visualizations.

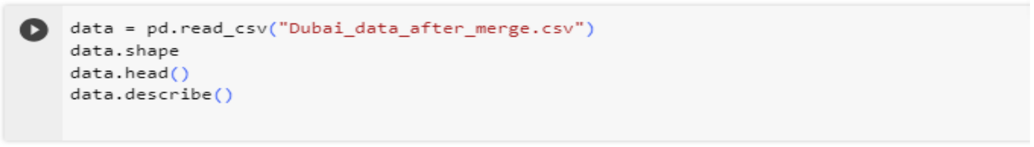


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. **Loading and Initial Exploration of Data:**

- The code loads a dataset from a CSV file called "Dubai_data_after_merge.csv" using pandas and displays:
- The shape (number of rows and columns) of the dataset.
- The first few rows of the dataset using data.head().

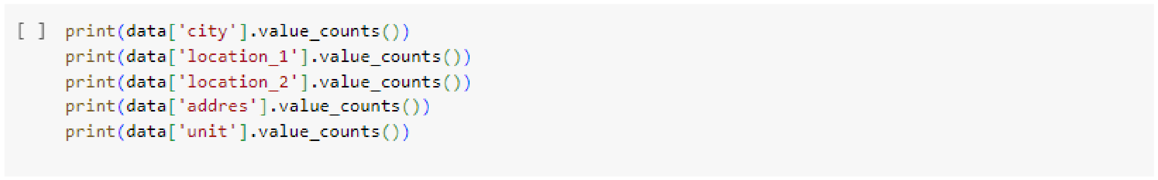
- Descriptive statistics, such as count, mean, and quartiles, using data.describe().



```
data = pd.read_csv("Dubai_data_after_merge.csv")
data.shape
data.head()
data.describe()
```

3. **Data Summary**:

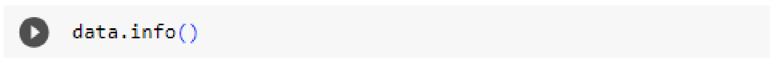
- The code prints the count of unique values for several categorical columns ('city,' 'location_1,' 'location_2,' 'addres,' and 'unit') using the `value_counts()` function. This helps in understanding the distribution of data in these columns.



```
[ ] print(data['city'].value_counts())
    print(data['location_1'].value_counts())
    print(data['location_2'].value_counts())
    print(data['addres'].value_counts())
    print(data['unit'].value_counts())
```

4. **Data Information:**

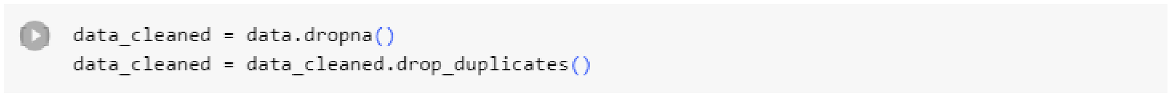
- The `data.info()` function provides information about the dataset, including the data types of columns and the presence of missing values.



```
data.info()
```

5. **Data Cleaning:**

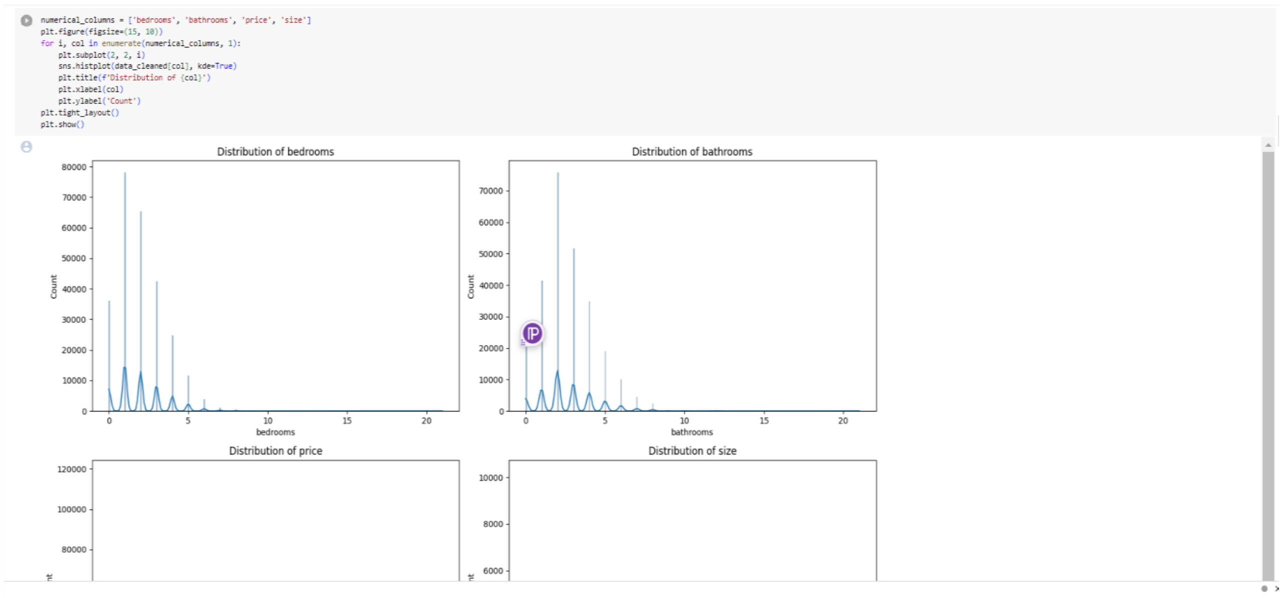
- The code removes rows with missing values using `dropna()`.
- It also removes duplicate records using `drop_duplicates()`, creating a cleaned version of the dataset named `data_cleaned`.



```
data_cleaned = data.dropna()
data_cleaned = data_cleaned.drop_duplicates()
```

6. **Data Visualization - Numerical Columns:**

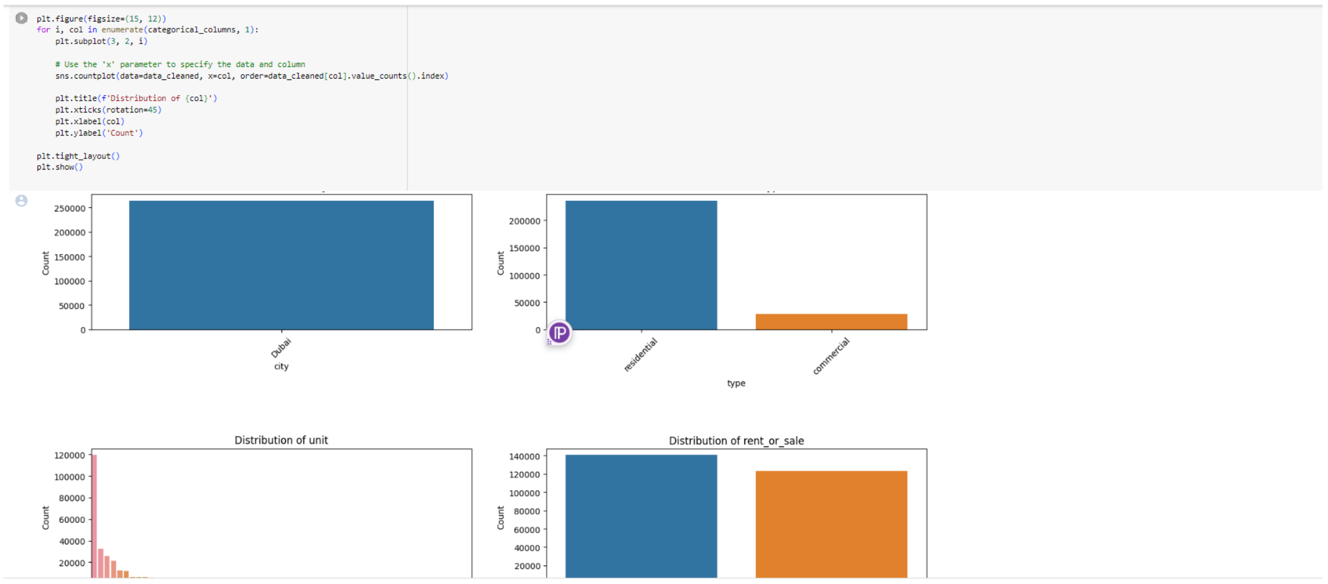
- This part of the code creates histograms with Kernel Density Estimation (KDE) for numerical columns ('bedrooms,' 'bathrooms,' 'price,' and 'size'). The histograms show the distribution of these numerical attributes.



```
numerical_columns = ['bedrooms', 'bathrooms', 'price', 'size']
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_columns, 1):
    plt.subplot(2, 2, i)
    sns.histplot(data_cleaned[col], kde=True)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

7. Data Visualization - Categorical Columns:

- This section generates count plots for categorical columns ('city,' 'type,' 'unit,' 'rent_or_sale,' and 'source'). These plots display the distribution of values in each category.

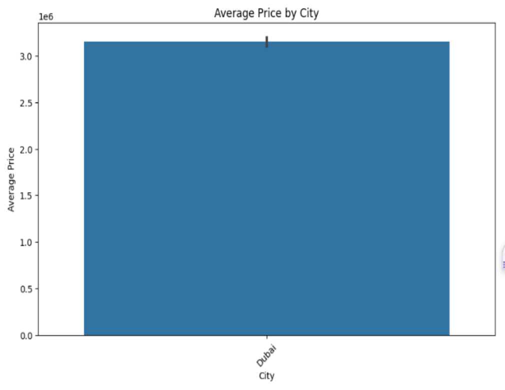


```
plt.figure(figsize=(15, 12))
for i, col in enumerate(categorical_columns, 1):
    plt.subplot(3, 2, i)

    # Use the 'x' parameter to specify the data and column
    sns.countplot(data=data_cleaned, x=col, order=data_cleaned[col].value_counts().index)

    plt.title(f'Distribution of {col}')
    plt.xticks(rotation=45)
    plt.xlabel(col)
    plt.ylabel('Count')

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

8. Price Analysis:

- This code segment creates a bar plot that displays the average property prices by city, helping visualize price variations across different cities.

```
plt.figure(figsize=(10, 6))
sns.barplot(x='city', y='price', data=data_cleaned)
plt.title('Average Price by City')
plt.xticks(rotation=45)
plt.xlabel('City')
plt.ylabel('Average Price')
plt.show()
```

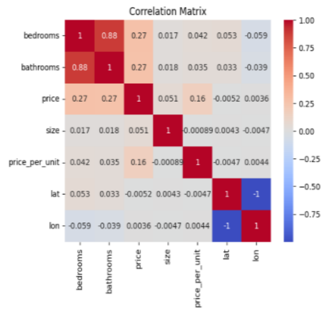


9. Correlation Analysis:

- The code calculates and visualizes a correlation matrix for numerical attributes. The heatmap shows the correlations between attributes, helping identify relationships between them.

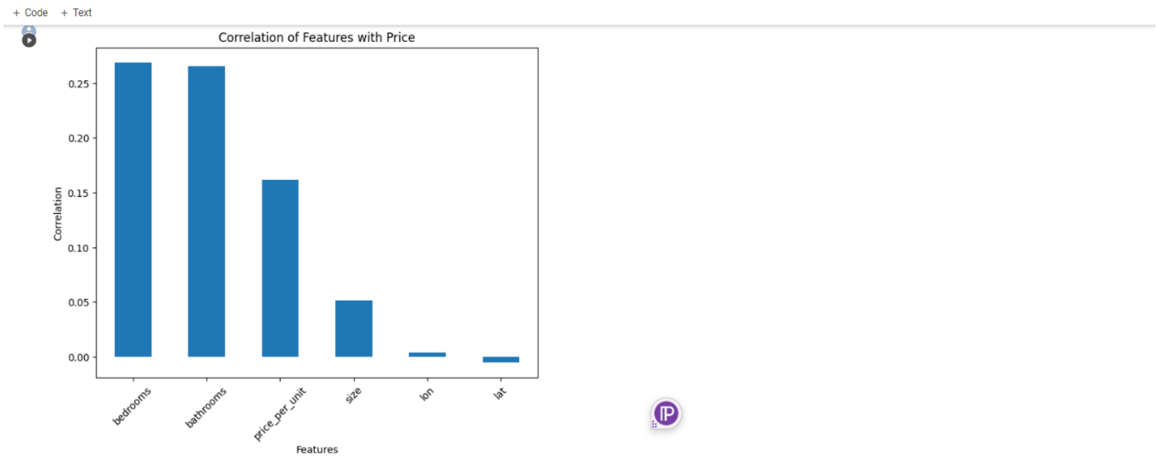
```
correlation_matrix = data_cleaned.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

```
<ipython-input-15-4cfbf105a1c0>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = data_cleaned.corr()
```



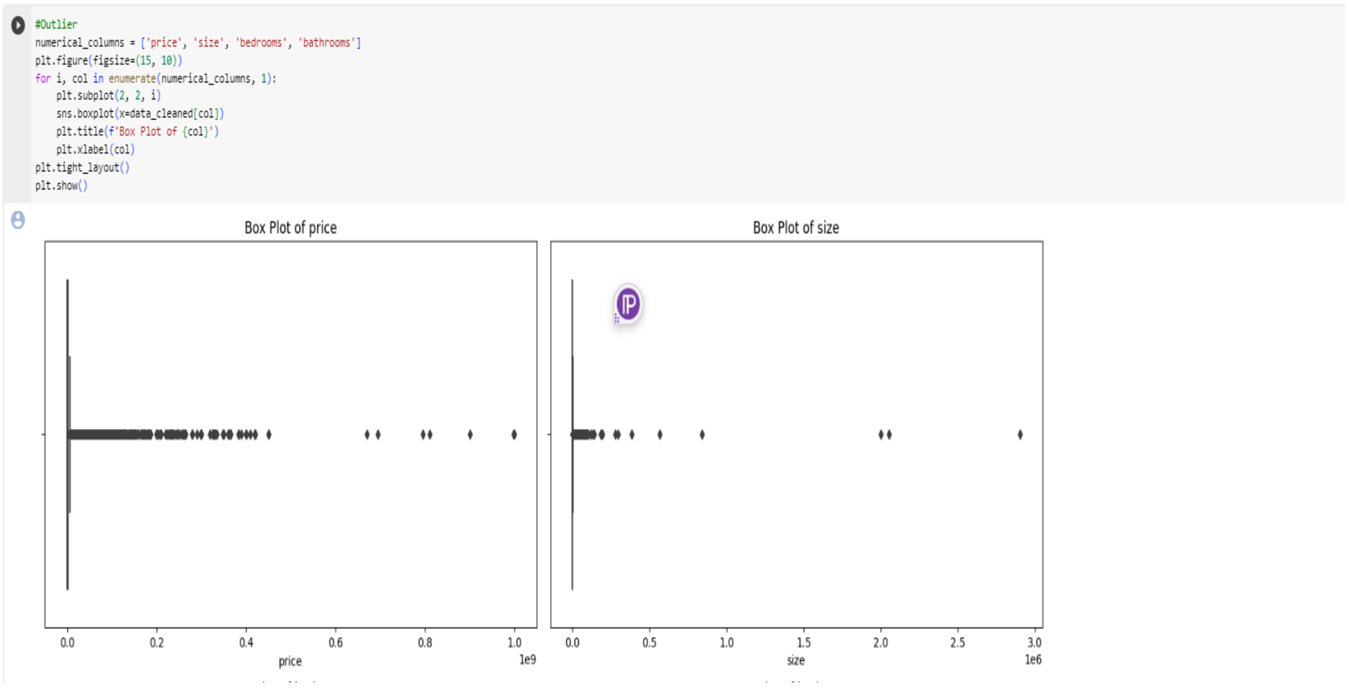
	bedrooms	bathrooms	price	size	price_per_unit	lat	lon
bedrooms	1	0.88	0.27	0.017	0.042	0.053	-0.059
bathrooms	0.88	1	0.27	0.018	0.035	0.033	-0.039
price	0.27	0.27	1	0.051	0.16	-0.0052	0.0036
size	0.017	0.018	0.051	1	-0.00089	0.0043	-0.0047
price_per_unit	0.042	0.035	0.16	-0.00089	1	-0.0047	0.0044
lat	0.053	0.033	-0.0052	0.0043	-0.0047	1	-1
lon	-0.059	-0.039	0.0036	-0.0047	0.0044	-1	1

```
correlation_with_price = correlation_matrix['price'].sort_values(ascending=False)

plt.figure(figsize=(8, 6))
correlation_with_price.drop('price').plot(kind='bar')
plt.title('Correlation of Features with Price')
plt.xlabel('Features')
plt.ylabel('Correlation')
plt.xticks(rotation=45)
plt.show()
```

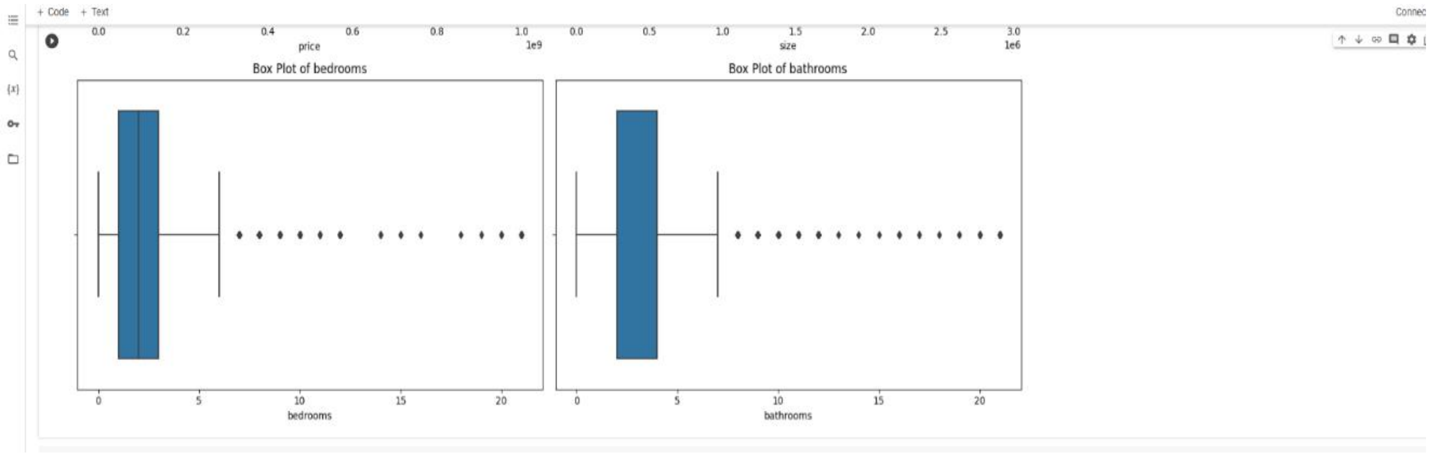


10. Outlier Detection and Removal:

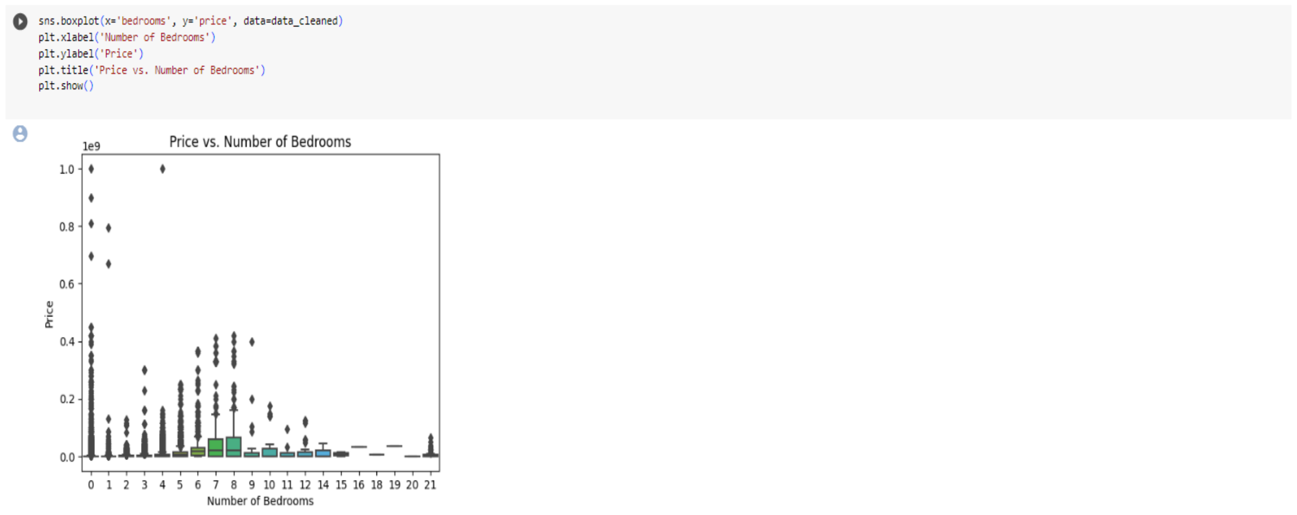
- This part of the code creates box plots for numerical columns to detect potential outliers.
- The code also removes outliers using the Interquartile Range
- (IQR) method and displays the cleaned dataset.



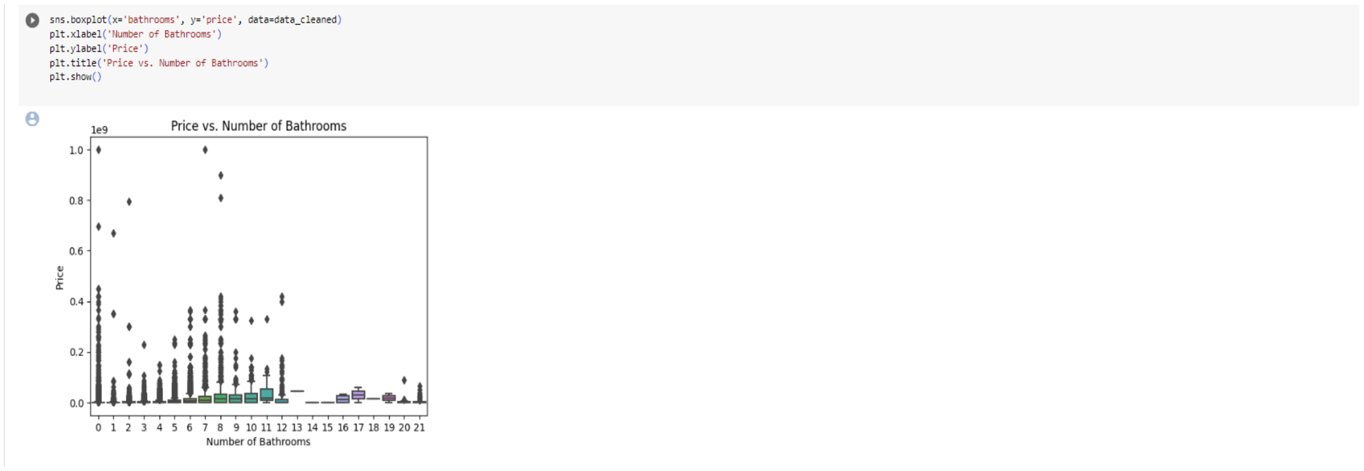
```
#Outlier
numerical_columns = ['price', 'size', 'bedrooms', 'bathrooms']
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_columns, 1):
    plt.subplot(2, 2, i)
    sns.boxplot(x=data_cleaned[col])
    plt.title(f'Box Plot of {col}')
    plt.xlabel(col)
plt.tight_layout()
plt.show()
```



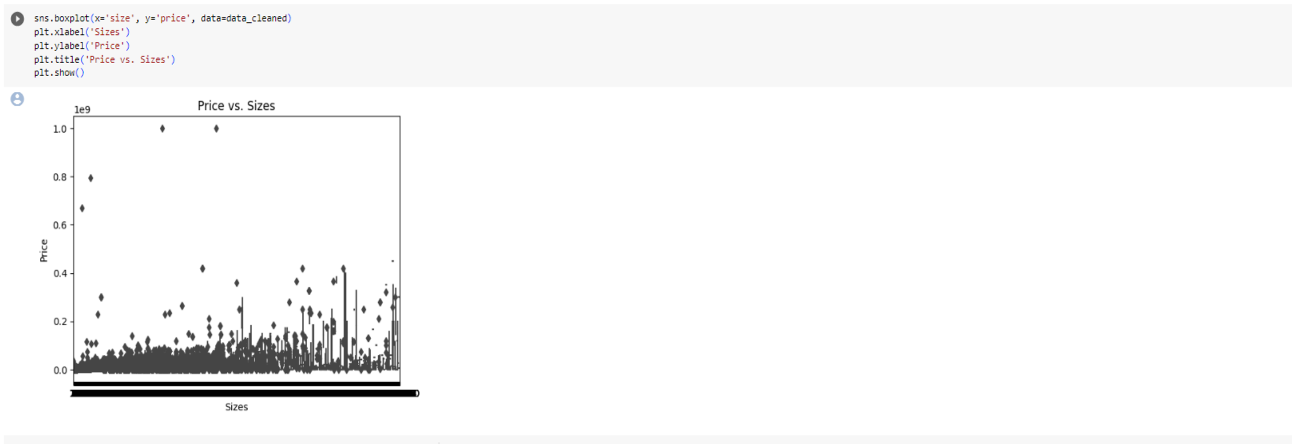
- This part of the code creates additional box plots to visualize the relationships between 'bedrooms,' 'bathrooms,' 'size,' and 'price.' These plots help identify potential outliers.



```
sns.boxplot(x='bedrooms', y='price', data=data_cleaned)
plt.xlabel('Number of Bedrooms')
plt.ylabel('Price')
plt.title('Price vs. Number of Bedrooms')
plt.show()
```



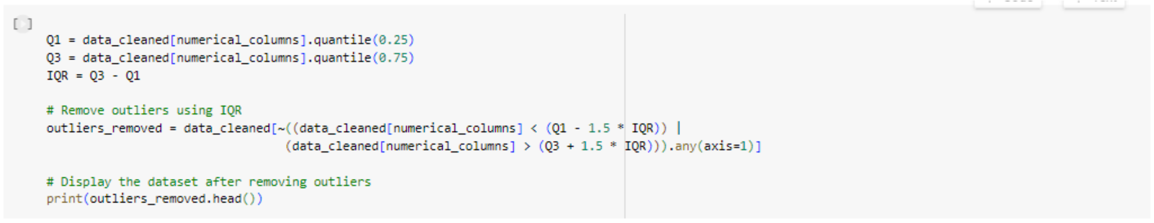
```
sns.boxplot(x='bathrooms', y='price', data=data_cleaned)
plt.xlabel('Number of Bathrooms')
plt.ylabel('Price')
plt.title('Price vs. Number of Bathrooms')
plt.show()
```



```
sns.boxplot(x='size', y='price', data=data_cleaned)
plt.xlabel('Sizes')
plt.ylabel('Price')
plt.title('Price vs. Sizes')
plt.show()
```

11. Outlier Removal Using IQR:

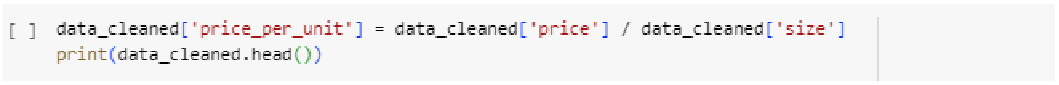
- This section calculates the Interquartile Range (IQR) for the numerical columns ('price,' 'size,' 'bedrooms,' 'bathrooms') and uses it to identify and remove outliers. The dataset without outliers is printed for reference.



```
[ ]
Q1 = data_cleaned[numerical_columns].quantile(0.25)
Q3 = data_cleaned[numerical_columns].quantile(0.75)
IQR = Q3 - Q1

# Remove outliers using IQR
outliers_removed = data_cleaned[~((data_cleaned[numerical_columns] < (Q1 - 1.5 * IQR)) |
                                 (data_cleaned[numerical_columns] > (Q3 + 1.5 * IQR))).any(axis=1)]

# Display the dataset after removing outliers
print(outliers_removed.head())
```

12. Price Per Unit Calculation:

- A new column, 'price_per_unit,' is created by dividing 'price' by 'size.' This represents the price per unit area or size.

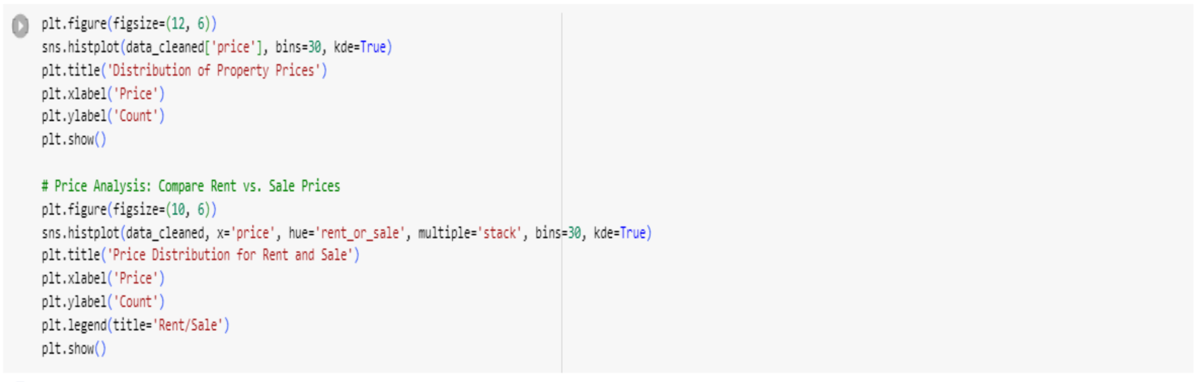


```
[ ] data_cleaned['price_per_unit'] = data_cleaned['price'] / data_cleaned['size']
    print(data_cleaned.head())
```

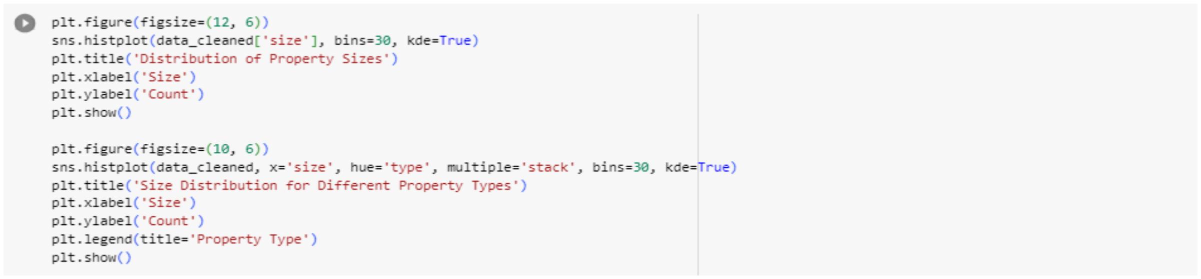
13. Price and Size Distribution Analysis:

- These code sections provide various visualizations of price and size distributions:
- Histogram of property prices with KDE.
- Price distribution for rental and sale properties.

- Histogram of property sizes with KDE.
- Size distribution for different property types.



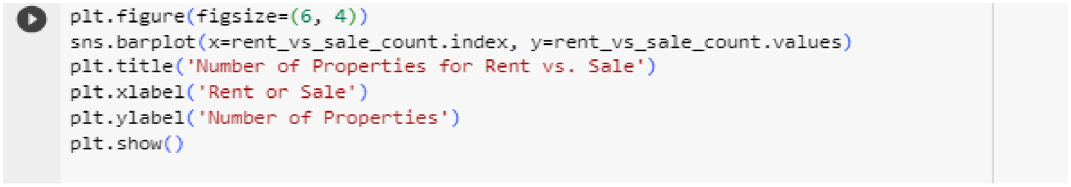
```
plt.figure(figsize=(12, 6))
sns.histplot(data_cleaned['price'], bins=30, kde=True)
plt.title('Distribution of Property Prices')
plt.xlabel('Price')
plt.ylabel('Count')
plt.show()

# Price Analysis: Compare Rent vs. Sale Prices
plt.figure(figsize=(10, 6))
sns.histplot(data_cleaned, x='price', hue='rent_or_sale', multiple='stack', bins=30, kde=True)
plt.title('Price Distribution for Rent and Sale')
plt.xlabel('Price')
plt.ylabel('Count')
plt.legend(title='Rent/Sale')
plt.show()
```



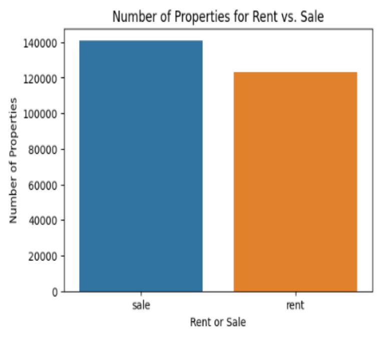
```
plt.figure(figsize=(12, 6))
sns.histplot(data_cleaned['size'], bins=30, kde=True)
plt.title('Distribution of Property Sizes')
plt.xlabel('Size')
plt.ylabel('Count')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(data_cleaned, x='size', hue='type', multiple='stack', bins=30, kde=True)
plt.title('Size Distribution for Different Property Types')
plt.xlabel('Size')
plt.ylabel('Count')
plt.legend(title='Property Type')
plt.show()
```

14. Number of Properties for Rent vs. Sale:

- This code segment generates a bar plot to compare the number of properties available for rent and sale.

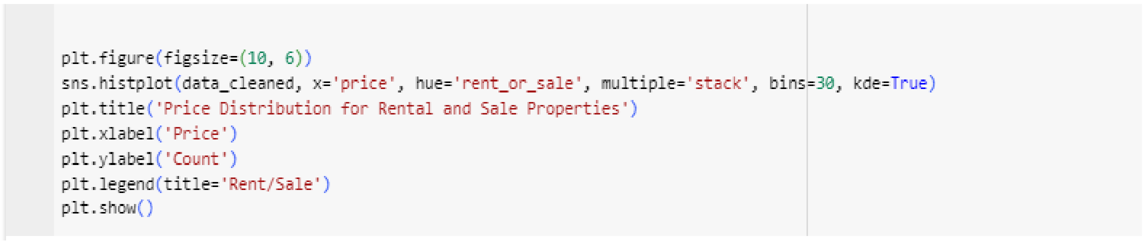


```
plt.figure(figsize=(6, 4))
sns.barplot(x=rent_vs_sale_count.index, y=rent_vs_sale_count.values)
plt.title('Number of Properties for Rent vs. Sale')
plt.xlabel('Rent or Sale')
plt.ylabel('Number of Properties')
plt.show()
```



15. Price Distribution for Rental and Sale Properties:

- This final section visualizes the price distribution for both rental and sale properties, comparing their distributions.



```
plt.figure(figsize=(10, 6))
sns.histplot(data_cleaned, x='price', hue='rent_or_sale', multiple='stack', bins=30, kde=True)
plt.title('Price Distribution for Rental and Sale Properties')
plt.xlabel('Price')
plt.ylabel('Count')
plt.legend(title='Rent/Sale')
plt.show()
```

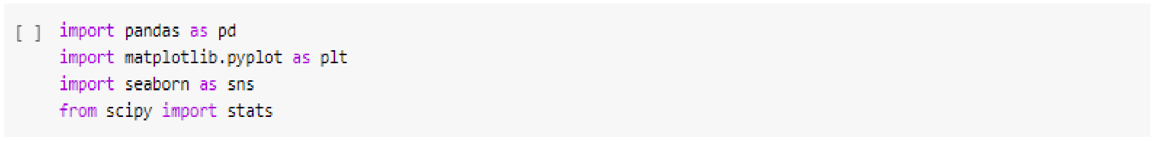
The code conducts a comprehensive exploratory data analysis (EDA) on the dataset, including data cleaning, visualization of numerical and categorical data, outlier detection, and statistical analysis. These analyses help in understanding the dataset's characteristics, relationships between attributes, and potential data issues.

Finally, the last step in data analysis and manipulation is cleaning our data from its outliers, let’s see how this effect in our data.

This code performs outlier detection and removal on a dataset. Let's go through each part and explain the code and its output:

1. **Library Imports**:

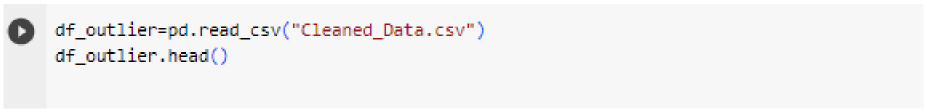
- This section imports necessary libraries, including pandas for data manipulation, matplotlib for plotting, seaborn for enhanced visualization, and scipy for statistical functions.



```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from scipy import stats
```

2. **Loading the Dataset:**

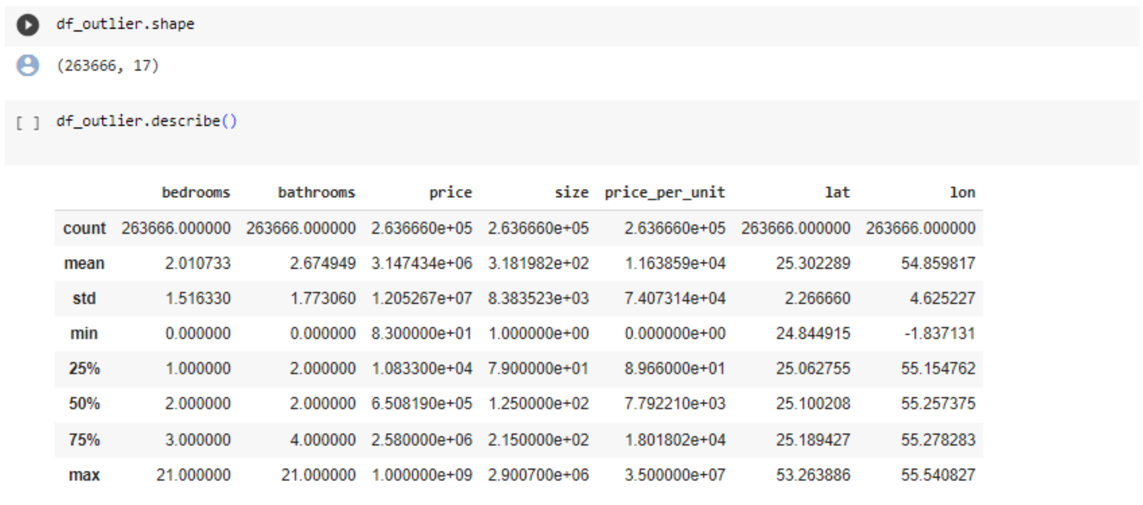
- The code loads a dataset from a CSV file called "Cleaned_Data.csv" into a DataFrame named `df_outlier`.



```
df_outlier=pd.read_csv("Cleaned_Data.csv")
df_outlier.head()
```

3. Exploring the Dataset:

- These lines provide information about the dataset, including its shape (number of rows and columns) and summary statistics (mean, min, max, etc.) for numerical columns.



```
df_outlier.shape
```

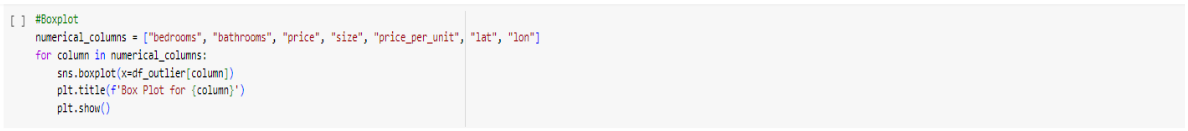
```
(263666, 17)
```

```
[ ] df_outlier.describe()
```

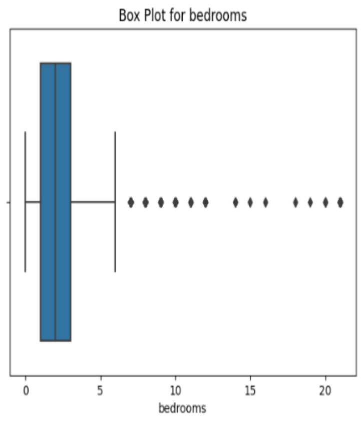
	bedrooms	bathrooms	price	size	price_per_unit	lat	lon
count	263666.000000	263666.000000	2.636660e+05	2.636660e+05	2.636660e+05	263666.000000	263666.000000
mean	2.010733	2.674949	3.147434e+06	3.181982e+02	1.163859e+04	25.302289	54.859817
std	1.516330	1.773060	1.205267e+07	8.383523e+03	7.407314e+04	2.266660	4.625227
min	0.000000	0.000000	8.300000e+01	1.000000e+00	0.000000e+00	24.844915	-1.837131
25%	1.000000	2.000000	1.083300e+04	7.900000e+01	8.966000e+01	25.062755	55.154762
50%	2.000000	2.000000	6.508190e+05	1.250000e+02	7.792210e+03	25.100208	55.257375
75%	3.000000	4.000000	2.580000e+06	2.150000e+02	1.801802e+04	25.189427	55.278283
max	21.000000	21.000000	1.000000e+09	2.900700e+06	3.500000e+07	53.263886	55.540827

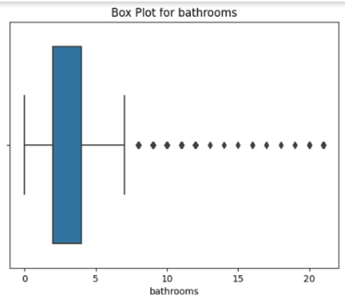
4. Boxplots for Numerical Columns:

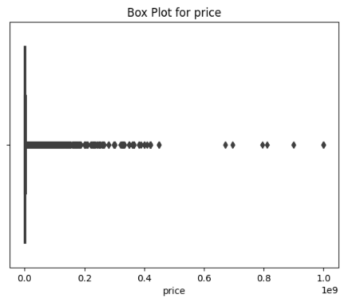
- This loop creates box plots for each numerical column to visually identify potential outliers. Each box plot shows the distribution of values for that column.



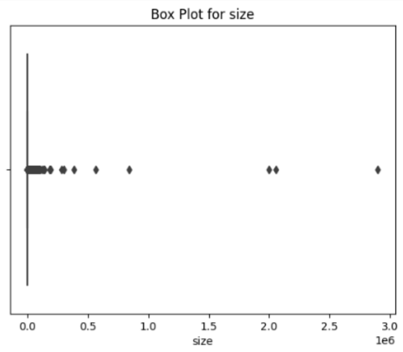
```
[ ] #Boxplot
    numerical_columns = ["bedrooms", "bathrooms", "price", "size", "price_per_unit", "lat", "lon"]
    for column in numerical_columns:
        sns.boxplot(x=df_outlier[column])
        plt.title(f'Box Plot for {column}')
        plt.show()
```

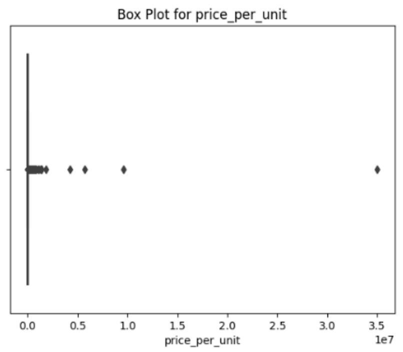



Box Plot for bathrooms
0
5
10
15
20
bathrooms


Box Plot for price
0.0
0.2
0.4
0.6
0.8
1.0
1e9
price

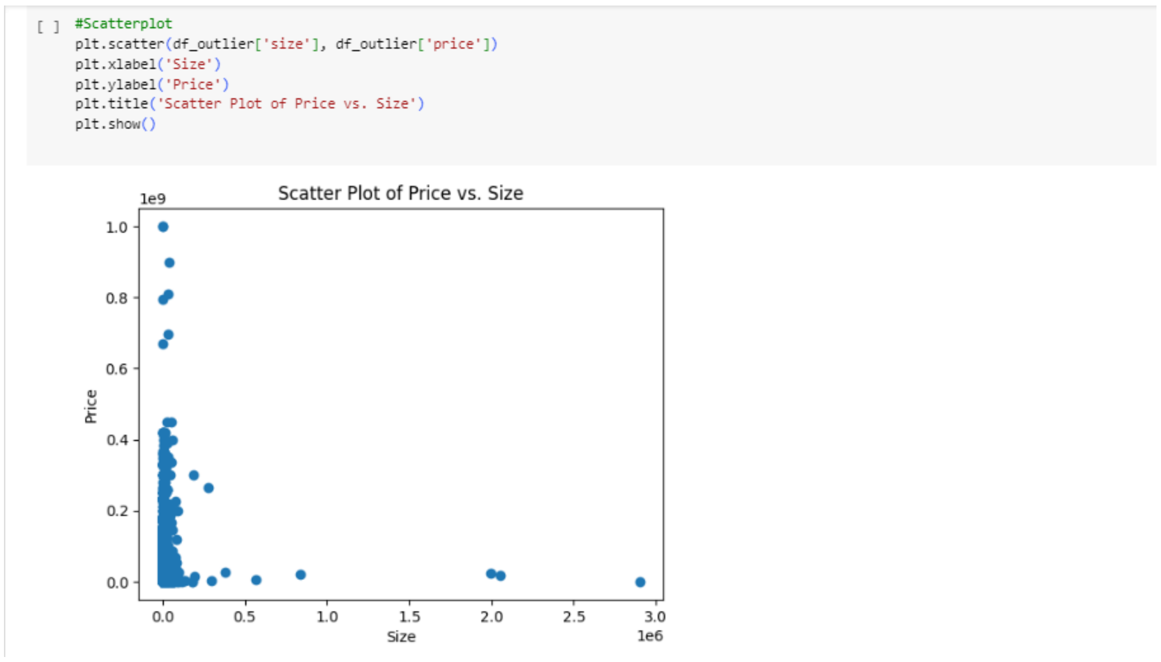
+ Code + Text


Box Plot for size
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e6
size


Box Plot for price_per_unit
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1e7
price_per_unit

5. **Scatter Plot:**

- This code generates a scatter plot to visualize the relationship between 'size' and 'price' in the dataset.



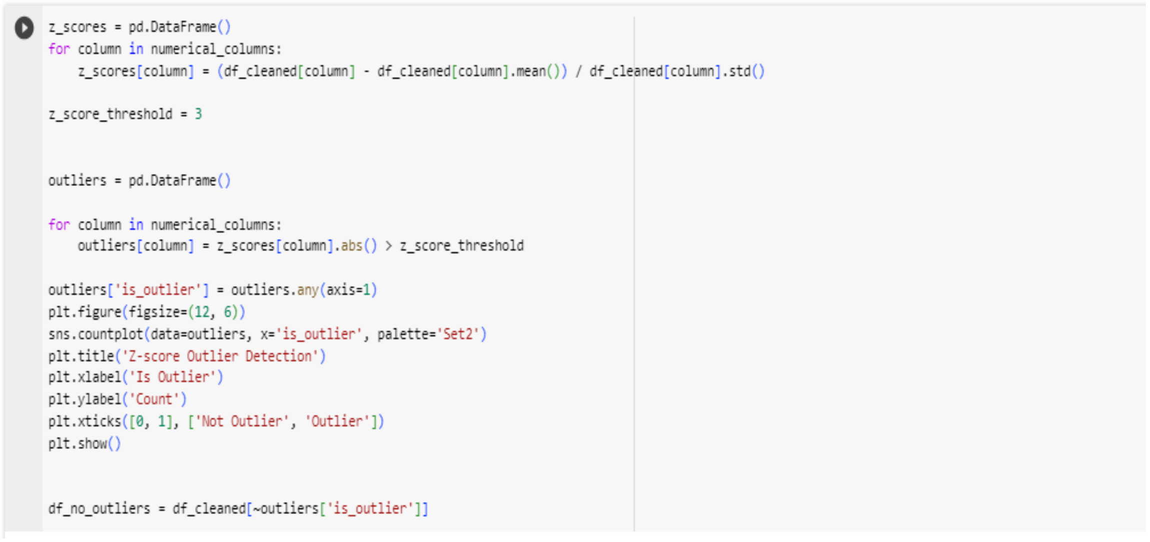
```
[ ] #Scatterplot
    plt.scatter(df_outlier['size'], df_outlier['price'])
    plt.xlabel('Size')
    plt.ylabel('Price')
    plt.title('Scatter Plot of Price vs. Size')
    plt.show()
```

6. **Z-Score Outlier Detection:**

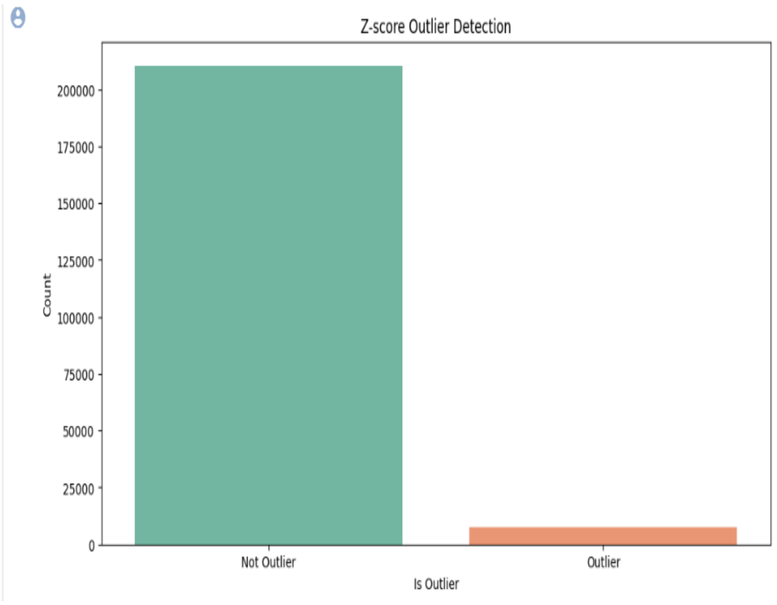
- This section calculates z-scores for each numerical column in the dataset and compares them against a threshold (z_score_threshold). Rows with z-scores exceeding the threshold are marked as outliers.

7. **Count of Outliers:**

- This code creates a count plot to visualize the number of outliers detected based on z-scores for each numerical column. It distinguishes between outliers and non-outliers.

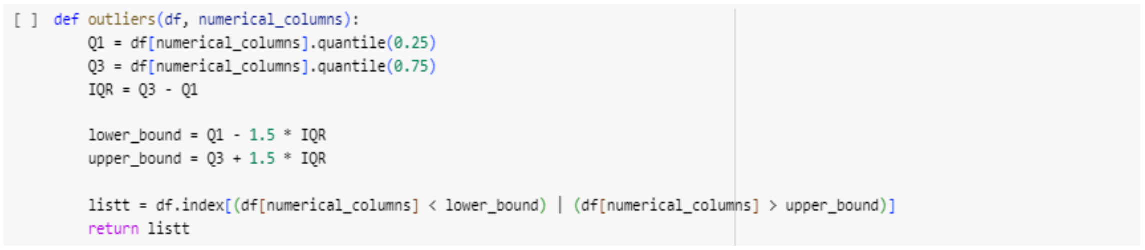


```
z_scores = pd.DataFrame()
for column in numerical_columns:
    z_scores[column] = (df_cleaned[column] - df_cleaned[column].mean()) / df_cleaned[column].std()

z_score_threshold = 3


outliers = pd.DataFrame()

for column in numerical_columns:
    outliers[column] = z_scores[column].abs() > z_score_threshold

outliers['is_outlier'] = outliers.any(axis=1)
plt.figure(figsize=(12, 6))
sns.countplot(data=outliers, x='is_outlier', palette='Set2')
plt.title('Z-score Outlier Detection')
plt.xlabel('Is Outlier')
plt.ylabel('Count')
plt.xticks([0, 1], ['Not Outlier', 'Outlier'])
plt.show()


df_no_outliers = df_cleaned[~outliers['is_outlier']]
```



8. **Outlier Removal Functions**:

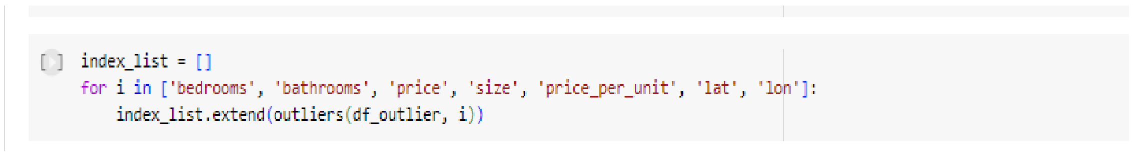
- These functions, `outliers` and `remove_outliers`, are defined to detect and remove outliers based on the Interquartile Range (IQR) method.



```
def outliers(df, numerical_columns):
    Q1 = df[numerical_columns].quantile(0.25)
    Q3 = df[numerical_columns].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    listt = df.index[(df[numerical_columns] < lower_bound) | (df[numerical_columns] > upper_bound)]
    return listt
```

9. **List of Outliers:**

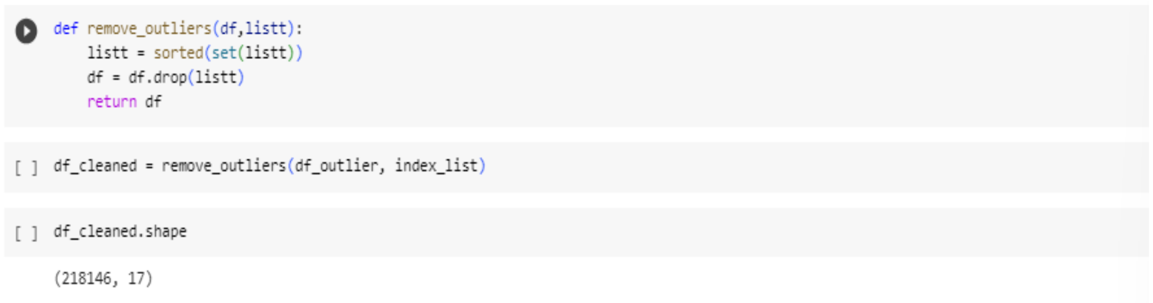
- This code creates a list of indices that correspond to the rows containing outliers for each numerical column.



```
index_list = []
for i in ['bedrooms', 'bathrooms', 'price', 'size', 'price_per_unit', 'lat', 'lon']:
    index_list.extend(outliers(df_outlier, i))
```

10. **Outlier Removal:**

- Outliers are removed from the `df_outlier` dataset using the `remove_outliers` function, resulting in the cleaned DataFrame `df_cleaned`.



```
def remove_outliers(df,listt):
    listt = sorted(set(listt))
    df = df.drop(listt)
    return df
```

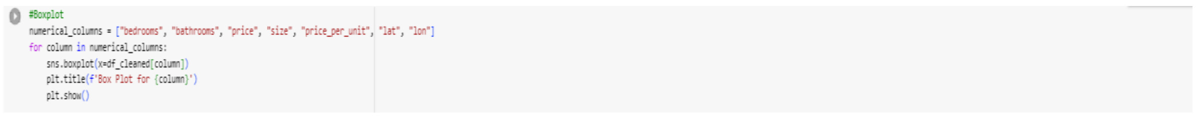
```
df_cleaned = remove_outliers(df_outlier, index_list)
```

```
df_cleaned.shape
```

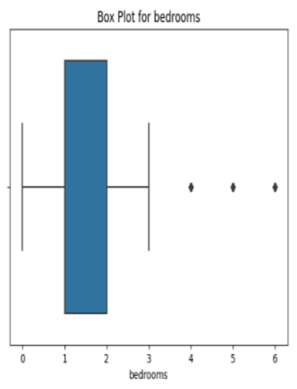
```
(218146, 17)
```

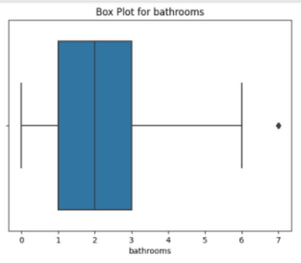
11. Boxplots After Outlier Removal:

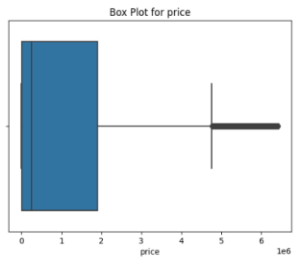
- Similar to step 4, this code generates box plots for numerical columns, but after outliers have been removed. This helps visualize the cleaned data.

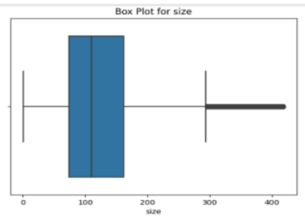


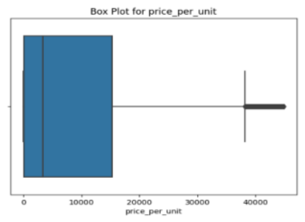
```
#Boxplot
numerical_columns = ["bedrooms", "bathrooms", "price", "size", "price_per_unit", "lat", "lon"]
for column in numerical_columns:
    sns.boxplot(x=df_cleaned[column])
    plt.title(f'Box Plot for {column}')
    plt.show()
```

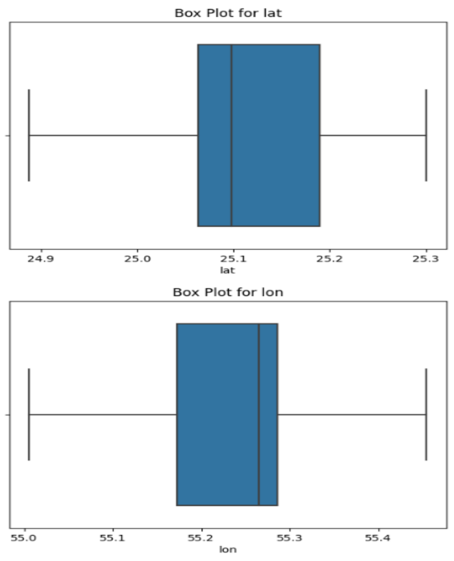






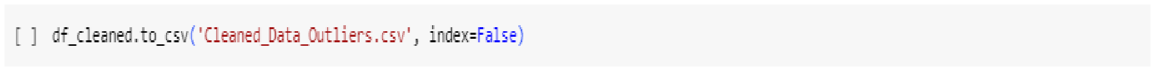






12. Saving the Cleaned Data:

- The code saves the cleaned dataset (without outliers) to a new CSV file named "Cleaned_Data_Outliers.csv."



```
[ ] df_cleaned.to_csv('Cleaned_Data_Outliers.csv', index=False)
```

The output of this code includes visualizations of box plots for numerical columns, scatter plots, a count plot showing the number of outliers, and box plots for the cleaned dataset after outlier removal. The cleaned data is saved as a new CSV file. The code primarily focuses on outlier detection and removal to improve data quality.

Last step is to visualize our data with intuitive dashboards and interactive visualizations that are created to present insights in a user-friendly manner.

Power BI is an all-in-one business intelligence solution that allows users to connect to data from multiple sources, convert and model it, produce interactive and visually attractive reports and dashboards, and share insights with others. It is a very useful tool for making data-driven choices and generating actionable insights within enterprises.

1. Data Sources and Connectivity:

Power BI supports a broad range of data sources, including databases (SQL Server, Oracle, MySQL), cloud services (Azure, AWS, Google Cloud), on-premises data sources, Excel spreadsheets, web services (REST APIs), and others. This adaptability enables you to collect data from many platforms. We used previously collected data from the python scripts given above, which we imported into the Power BI application.

2. Data Modeling:

In the Power BI Desktop application, you can create data models by defining relationships between tables. These relationships allow you to combine data from various sources, providing a comprehensive view for analysis. Power BI employs a tabular data model with features such as hierarchies and calculated columns.

3. Report Building:

You can begin creating reports once your data has been loaded and modeled. The Power BI Desktop has an easy-to-use interface for creating reports. Visuals such as bar charts, pie charts, line charts, tables, maps, and more can be added. The formatting, colors, and interaction behaviors of these visuals can all be changed.

4. Dashboards:

In Power BI, dashboards resemble high-level summary pages. Key graphics from your reports can be pinned to a dashboard to give a brief summary of the most crucial metrics. These dashboards allow users to navigate directly to comprehensive reports.

5. Sharing and Collaboration:

Another cloud-based platform where you may publish your dashboards and reports is Power BI Service. Colleagues and other interested parties can view or update them if you share them with them. Reports can be accessed while on the go with a mobile app and can be embedded into webpages or other applications.

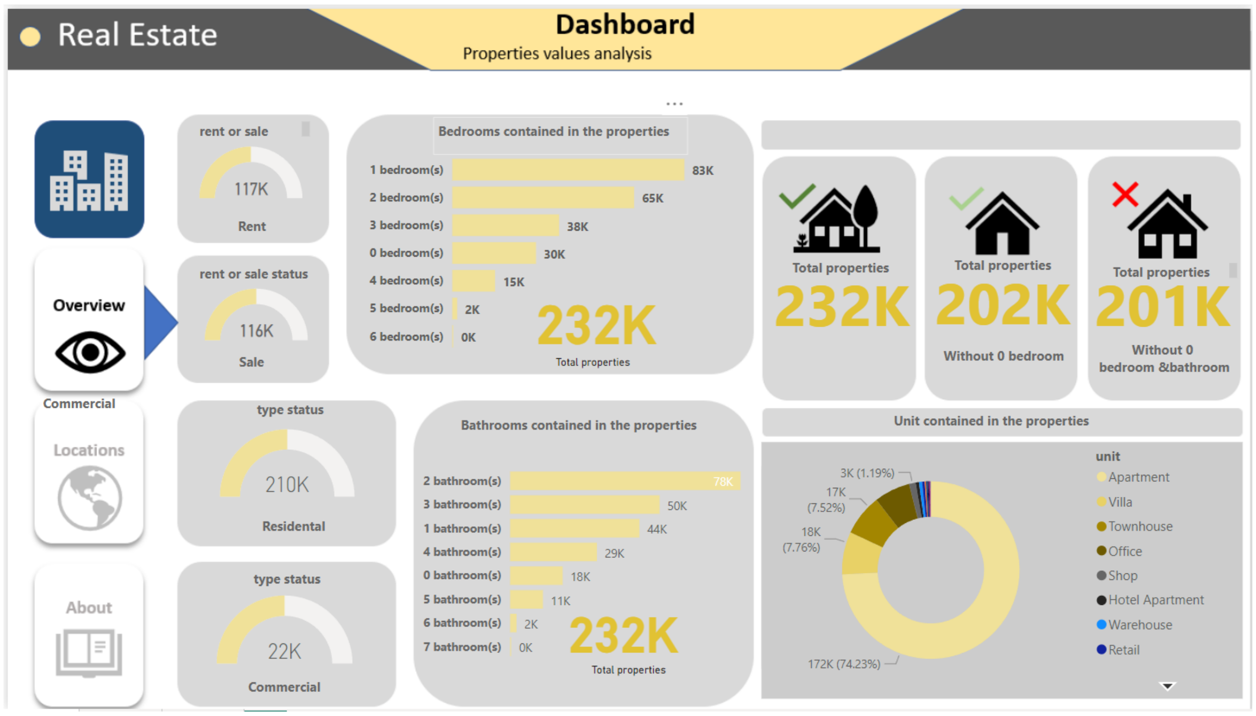
6. Data Refresh and Scheduled Updates:

It is essential that your reports are kept current. You can plan data refreshes with Power BI, which means that at predetermined intervals, your reports will automatically be updated with the most recent information from your sources.

7. Security and Permissions:

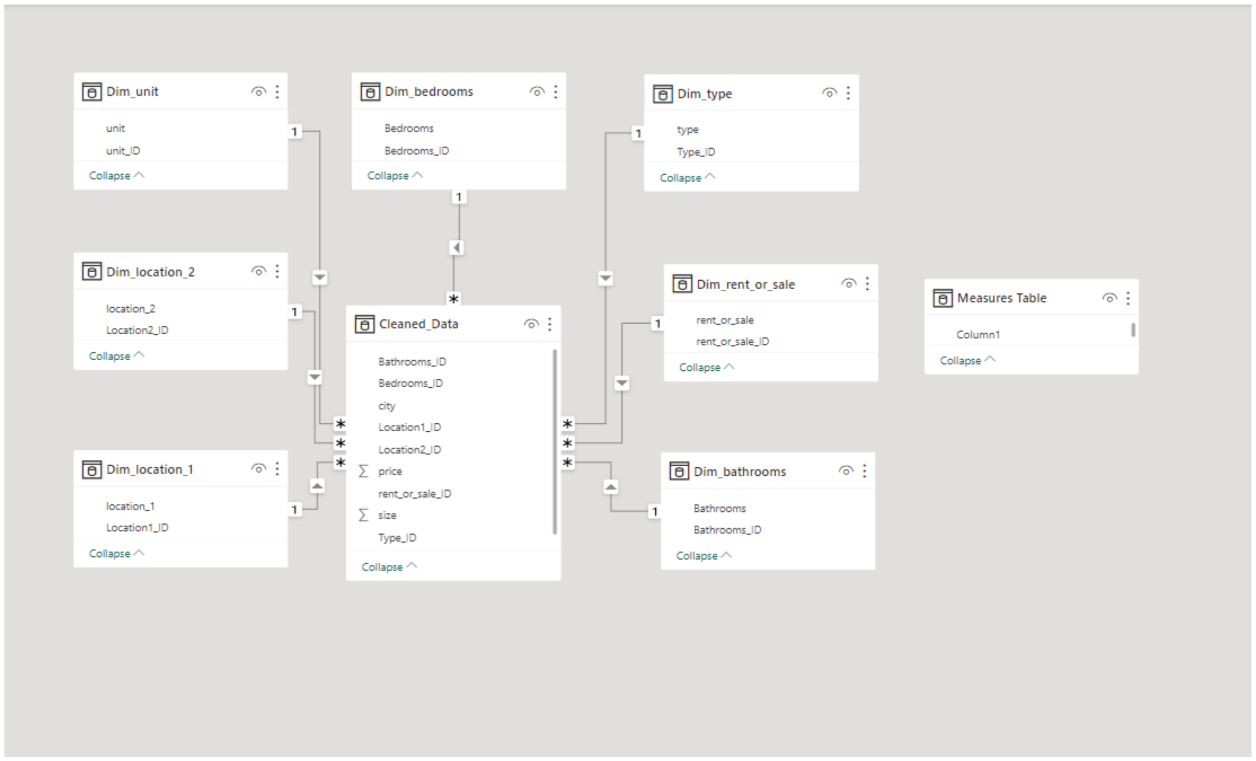
Power BI offers strong security capabilities. You are in charge of determining who may access your dashboards and reports, as well as to what extent. Row-level security, role-based permissions, and user authentication interaction with Azure Active Directory are all included in this.

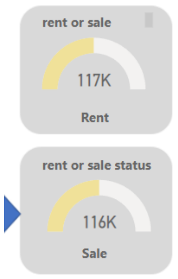
Dashboard PowerBI for our project:



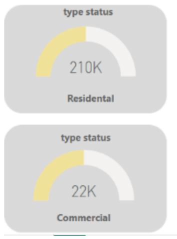
This is a dashboard built by Power BI to show the generalization of all the data that was collected during the scraping cleaning, and pre-processing of all the information we got from Dubbizle, Bayut and propertyfinder, this dashboard shows a lot of information in us in a small form factor which would take days to deduce from either the websites or the scraped data.

This the overview of the entire dashboard. Below are the dataclasses as input to PowerBI

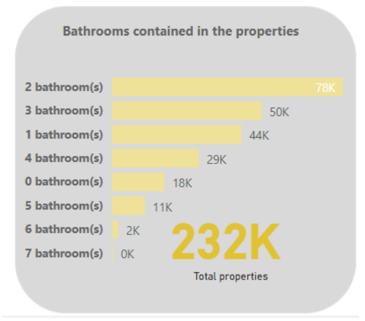




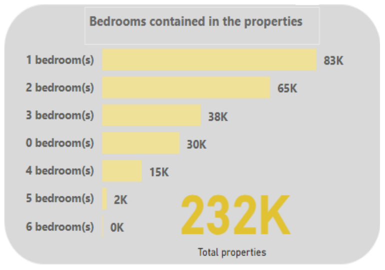
This is the distribution between the residential units where which the units are either for rent or sale, the unit distribution is somewhat equal, which shows that there is split supply of apartments/units that are either for rent or sale.



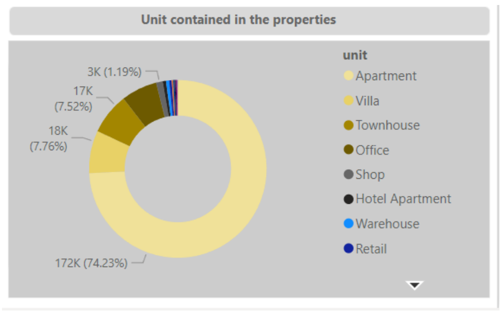
This is the distrbution of the amount of units that are available that are either residential or commercial. This shows a complete shift in the amount of supply from the commercial to the residential side, which means that logically, the commercial units should be priced higher than the residential units.



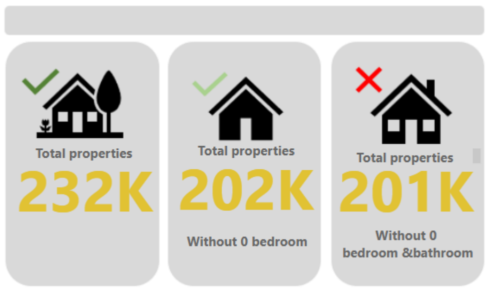
This shows the distribution between the amount of apartments with a certain amount of bathrooms, as is clear, there is an inverse effect against the amount of bathrooms starting from 5 to 7, with 0 being almost as uncommon as 5, while the most common in any unit is 2 bathrooms which is standard.



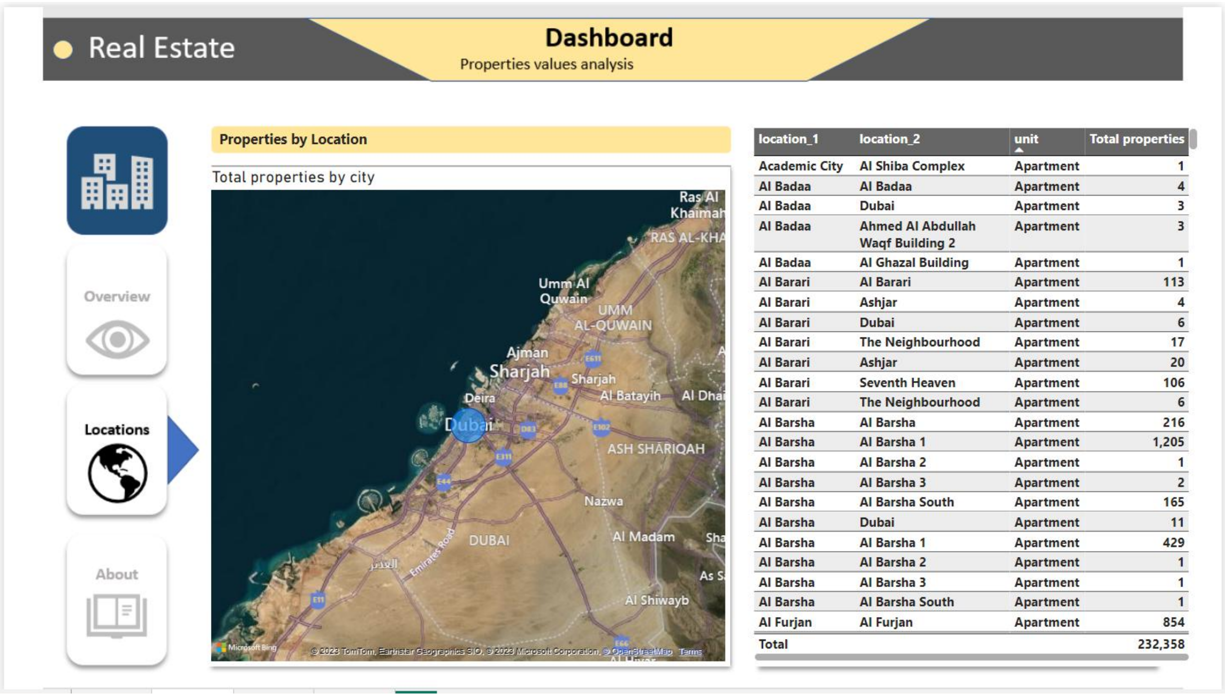
This shows the distribution between the amount of apartments with a certain amount of bedroom, this again shows the same sort of distribution we saw in the bathroom distrubtion but among the entire list with the exception of apartments with 0 bedrooms as it is not standard for an apartment to have 0 bedrooms



This is a Pi-Chart that shows the unit type distribution in the data we saw, here we can see that most of the data collected is apartments, where it is set at almost 75% of the data, while the rest of the unit types are split amongst the last 25%, with the lowest being retail, and warehouses.



Finally, This shows the amount of total properties collected during this project, with 232k datapoints collected, 202k of which atleast have 1 bedroom, and 201k of which atleast have a bedroom and a bathroom



Real Estate

Dashboard

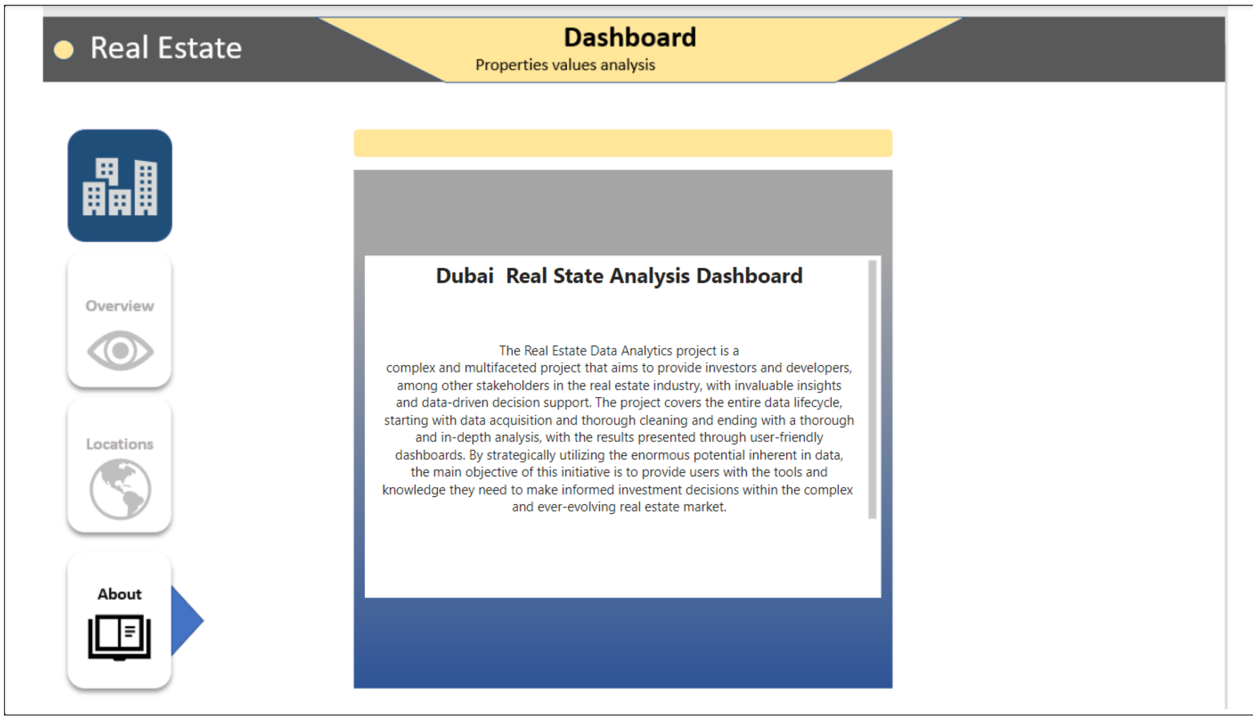
Properties values analysis

Properties by Location

Total properties by city

location_1	location_2	unit	Total properties
Academic City	Al Shiba Complex	Apartment	1
Al Badaa	Al Badaa	Apartment	4
Al Badaa	Dubai	Apartment	3
Al Badaa	Ahmed Al Abdullah Waqf Building 2	Apartment	3
Al Badaa	Al Ghazal Building	Apartment	1
Al Barari	Al Barari	Apartment	113
Al Barari	Ashjar	Apartment	4
Al Barari	Dubai	Apartment	6
Al Barari	The Neighbourhood	Apartment	17
Al Barari	Ashjar	Apartment	20
Al Barari	Seventh Heaven	Apartment	106
Al Barari	The Neighbourhood	Apartment	6
Al Barsha	Al Barsha	Apartment	216
Al Barsha	Al Barsha 1	Apartment	1,205
Al Barsha	Al Barsha 2	Apartment	1
Al Barsha	Al Barsha 3	Apartment	2
Al Barsha	Al Barsha South	Apartment	165
Al Barsha	Dubai	Apartment	11
Al Barsha	Al Barsha 1	Apartment	429
Al Barsha	Al Barsha 2	Apartment	1
Al Barsha	Al Barsha 3	Apartment	1
Al Barsha	Al Barsha South	Apartment	1
Al Furjan	Al Furjan	Apartment	854
Total			232,358

This is a heatmap of all the locations that have renting or sellable areas within the collected datapoints, this heatmap shows the highest density of units within the Dubai area, while all areas around it, while there is some units, it Is not as dense as Dubai.



Real Estate

Dashboard

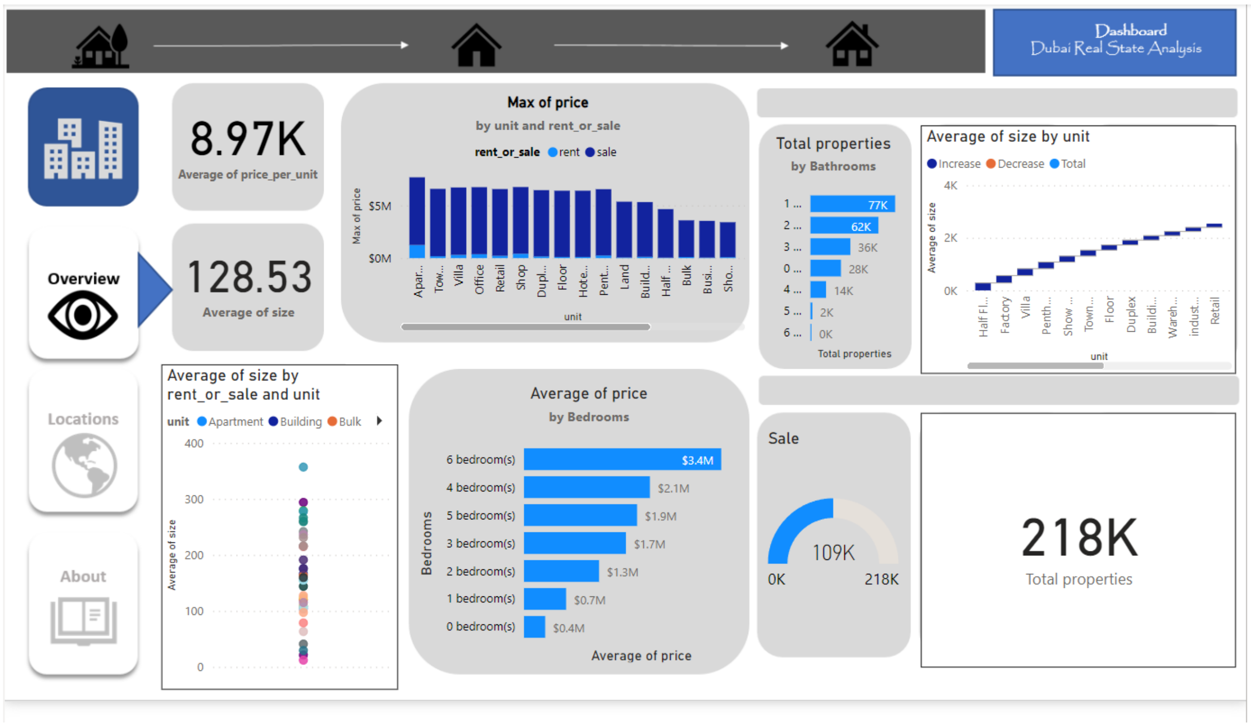
Properties values analysis

Dubai Real State Analysis Dashboard

The Real Estate Data Analytics project is a complex and multifaceted project that aims to provide investors and developers, among other stakeholders in the real estate industry, with invaluable insights and data-driven decision support. The project covers the entire data lifecycle, starting with data acquisition and thorough cleaning and ending with a thorough and in-depth analysis, with the results presented through user-friendly dashboards. By strategically utilizing the enormous potential inherent in data, the main objective of this initiative is to provide users with the tools and knowledge they need to make informed investment decisions within the complex and ever-evolving real estate market.

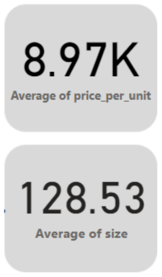
This about page explains the project thoroughly.

Another interactive Power BI with slightly changed visualizations and interactions:

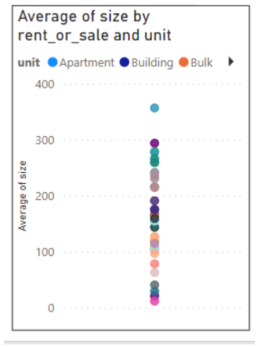


To show the generalization of all the data that was collected during the scraping cleaning, and

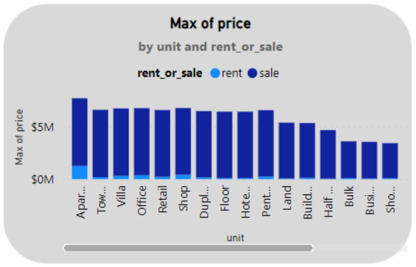
pre-processing of all the information we got from Dubbizle, Bayut and propertyfinder, this dashboard shows a lot of information in us in a small form factor which would take days to deduce from either the websites or the scraped data. This the overview of the entire dashboard. Below are the dataclasses as input to PowerBI



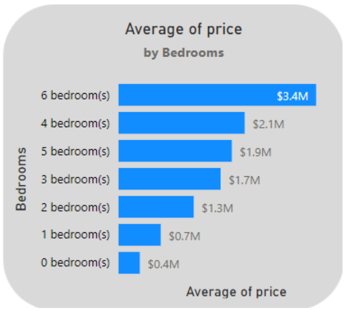
This shows the Average price per unit and the Average size in SQM



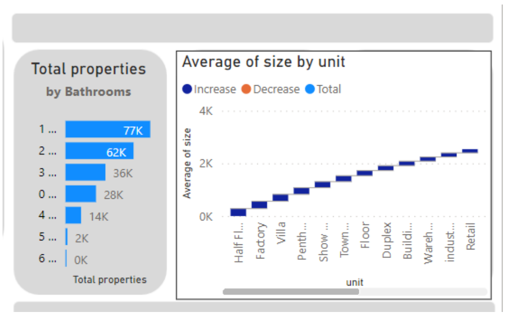
The Average Size split by the renting rate of the apartment



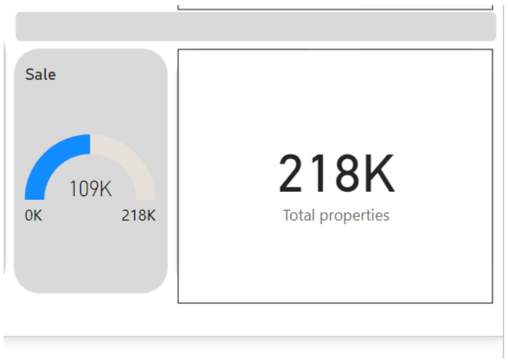
The Maximum price for each of the categories



The Average price of an apartment given the amount of bedrooms

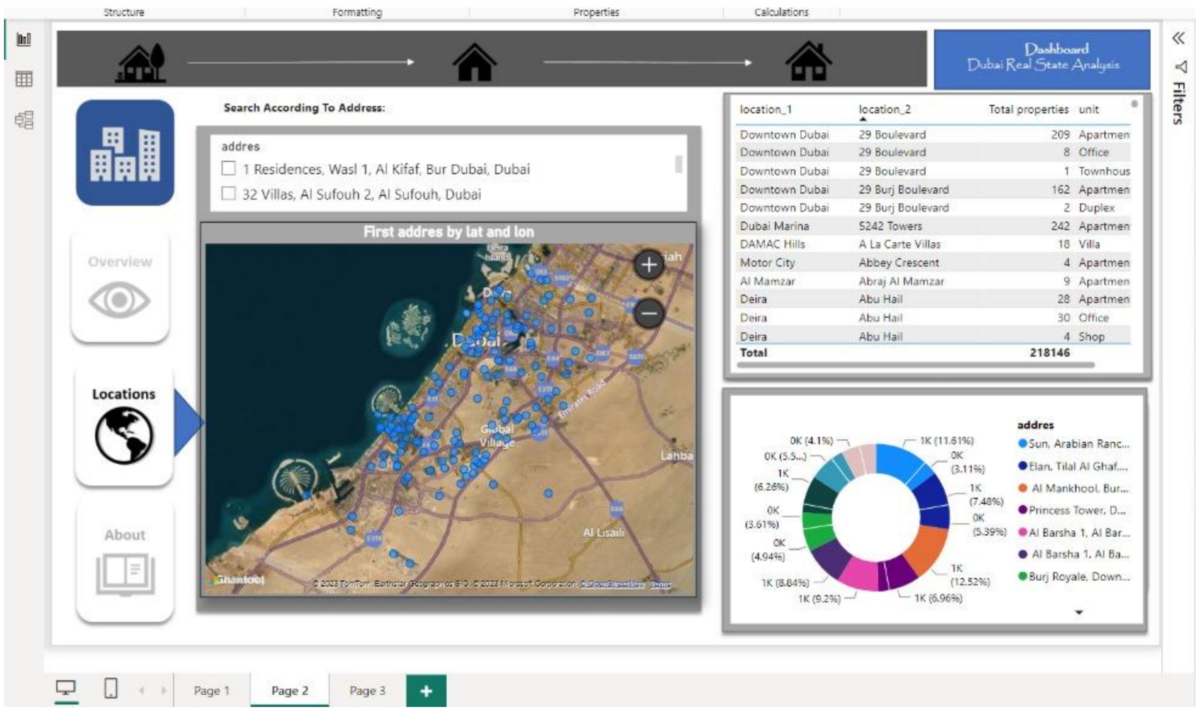


The amount of Apartments in each split in bathroom count and the average size of all the units comparatively



The Amount of units on sale and the total properties in the dataset

This is a heatmap of all the locations that have renting or sellable areas within the collected datapoints, this heatmap shows the highest density of units within the Dubai area, while all areas around it, while there is some units, it Is not as dense as Dubai. It also includes a pichart which shows the address density, with the biggest being 12%



location_1	location_2	Total properties	unit
Downtown Dubai	29 Boulevard	209	Apartmen
Downtown Dubai	29 Boulevard	8	Office
Downtown Dubai	29 Boulevard	1	Townhous
Downtown Dubai	29 Burj Boulevard	162	Apartmen
Downtown Dubai	29 Burj Boulevard	2	Duplex
Dubai Marina	5242 Towers	242	Apartmen
DAMAC Hills	A La Carte Villas	18	Villa
Motor City	Abbey Crescent	4	Apartmen
Al Mamzar	Abraj Al Mamzar	9	Apartmen
Deira	Abu Hail	28	Apartmen
Deira	Abu Hail	30	Office
Deira	Abu Hail	4	Shop
Total		**218146**	

Task Allocation

Data Scrapping	Data Cleaning	Data Integration	Exploratory Data Analysis (EDA)	Market Segmentation	Reporting and Visualization	Ethical Considerations	PowerBI & Report	Presentation
Abdelsalam Ahmed & Rowan Badr	Abdelsalam Ahmed	Rowan Badr	Rowan Badr	Abdelsalam Ahmed	Rowan Badr	Abdelsalam Ahmed	Abdelsalam Ahmed & Rowan Badr	Abdelsalam Ahmed & Rowan Badr

Conclusion

The Real Estate Data Analytics project is poised to revolutionize the way stakeholders approach real estate investments. By harnessing the power of data, it addresses the industry's challenges, offering data-driven insights, informed decision-making, and improved risk management.

In conclusion, this project represents a significant step toward maximizing opportunities and minimizing risks in the dynamic real estate market, ultimately benefiting investors, developers, and all stakeholders involved.