Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technologies



Bachelor Thesis

Comparative analysis of tools and technologies for data flow and data architecture design

Barkin Serhan Koca

© 2026 CZU Prague

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Barkin Koca

Informatics

Thesis title

Comparative analysis of tools and technologies for data flow and data architecture design

---

Objectives of thesis

The thesis focuses on the topic of web scrapping, data warehousing and processing, data analysis and visualization. The main objective of the thesis is to propose and implement a software aided data flow for scraping, storage and processing of real estate data.
Partial goals of the thesis are:
- study and analysis of available information sources
- specification of testing scenario, its requirements and goals
- outline the overall data flow and select appropriate tools for each partial step in the development life cycle
- implement and test the solution with actual data from chosen real estate agency for a given region and time frame

Methodology

The theoretical part of the thesis is based on study and analysis of available literature and online information resources with focus on topics such as web scrapping, data analysis, data processing and visualization and usage of ETL tools. In the practical part, a process for scrapping, storage and analysis of real estate data will be outlined and for each step a comparative analysis will be conducted to select the most appropriate tools / software. The proposed solution will be implemented and tested using real data. The level of automation, optimization and integration between various steps will be analyzed and measured. Based on the results of both thesis parts, conclusions will be formulated.

The proposed extent of the thesis

40-50

Keywords

SDLC, Data Processing, Data Analysis, Python, Technology Analysis, ETL, web scrapping

Recommended information sources

JEFFRIES, R. The Nature of Software Development, 2015. ISBN 1941222374
KIMBALL, R. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data 1st Edition, 2004. ISBN 8126505540
MITCHELL, R. Web Scraping with Python, 2018. ISBN 1491985577
REEVE, A. Managing Data in Motion: Data Integration Best Practice Techniques and Technologies, 2013. ISBN 0123971675
STEPHENS-DAVIDOWITZ, S. Don't Trust Your Gut: Using Data to Get What You Really Want in Life, 2022. ISBN 1526605112

Expected date of thesis defence

2024/25 SS – PEF

Thesis supervisor

Ing. Jan Pavlík, Ph.D.

Supervising department

Department of Information Technologies

Electronic approval: 09. 07. 2024

doc. Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 04. 11. 2024

doc. Ing. Tomáš Šubrt, Ph.D.

Dean

Prague on 10. 03. 2026

Declaration

I declare that I have worked on my bachelor thesis titled "Comparative analysis of tools and technologies for data flow and data architecture design" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on March 15, 2026 _____

Acknowledgement

I would like to thank Ing. Jan Pavlík, Ph.D., for his valuable advice, support and continious supervision during my work on this thesis

Comparative analysis of tools and technologies for data flow and data architecture design

Abstract

My thesis focuses on comparative analysis of tools and technologies for dataflow and data architecture design in an end-to-end pipeline.

Functionality and use case of this pipeline is as follows, data gets extracted from a real estate company website specifically filtered for rental apartments in Prague. Subsequently, extracted data goes through cleaning and preparation steps as a next step which is performed to have use case ready data handy.

Furthermore, research gets conducted in efforts to explore options for storage of this data. Finally, stored data is being pulled to visualize basic analysis derived from data obtained from web and visualized.

To summarize, the aim is primarily for conducting research comparative analysis on the tools and technologies for practical use case of Prague rental apartments analysis to provide an overview.

Keywords: SDLC, data analysis, data extraction, data preparation, data cleaning, data storage, data visualization, web scraping, technology analysis, ETL

Srovnávací analýza nástrojů a technologií pro návrh datových toků a datové architektury

Abstrakt

Moje práce se zaměřuje na srovnávací analýzu nástrojů a technologií pro datový tok a návrh datové architektury v komplexním pipeline.

Funkce a případ použití tohoto pipeline je následující: data jsou extrahována z webových stránek realitní kanceláře, speciálně filtrována pro pronajímané byty v Praze. Následně extrahovaná data procházejí kroky čištění a přípravy, což je další krok, který se provádí za účelem zajištění dostupnosti dat připravených pro případ použití.

Dále je proveden výzkum s cílem prozkoumat možnosti ukládání těchto dat. Nakonec jsou uložená data extrahována pro vizualizaci základní analýzy odvozené z dat získaných z webu a vizualizována.

Stručně řečeno, cílem je primárně provést srovnávací analýzu nástrojů a technologií pro praktické využití analýzy pronajímaných bytů v Praze, aby se poskytl přehled.

Klíčová slova: SDLC, data analysis, data extraction, data preparation, data cleaning, data storage, data visualization, web scraping, technology analysis, ETL

Table of content

1 Introduction 9

2 Objectives and Methodology 10

2.1 Objective 10

2.2 Methodology 10

3 Literature Review 11

3.1 Data collection 11

3.2 Web scraping 11

3.3 Tools, methods and libraries 15

3.3.1 Manual data collection 15

3.3.2 Pattern recognition (regular expressions) 15

3.3.3 HTTP requests and web communication 16

3.3.4 HTML parsing (extracting from page structure) 16

3.3.5 DOM interaction for Dynamic Websites 16

3.3.6 Vertical Aggregation (Gathering data from many similar sites) 17

3.3.7 Using semantic markup (Structured data extraction) 17

3.3.8 Visual scraping using computer vision 17

3.4 Data preparation and cleaning 18

3.5 Data storage 22

3.6 Exploratory data analysis 24

3.7 Data visualization 27

4 Practical Part 30

4.1 Introduction 30

4.2 Selection of tools 31

4.2.1 Comparative analysis of data extraction stage 31

4.2.2 Comparative analysis of data cleaning and preparation 34

4.2.3 Comparative analysis of data storage and visualization 35

4.3 Implementation 37

5 Results and Conclusion 42

6 References 46

7 List of pictures, tables, graphs and abbreviations 48

7.1 List of Figures 48

7.2 List of Tables 48

1 Introduction

Primary focus of my thesis is to make comparative analysis of tools and technologies for a data architecture design.

This architecture is focused on extraction of real estate data in Czech Republic, Prague. It is to create end to end data flow including scraping of data from Sreality.cz, specifically for rental apartment prices located in Prague.

It searches through advertisement pages collects individual property details and collects data such as price, energy efficiency, area and dates. In order to make process more accurate and reliable way firstly attempts to pull data from site's API and if that does not return desired data implemented fallback option is triggered which is where the HTML scraping happens.

Moreover, concurrent processing is used for enabling the processing of multiple listings in parallel which puts it to even more efficient position.

After the extraction as next step in this pipeline this data goes through cleaning and preparation steps. Where data standardization, validation of the dataset is performed. To increase data completeness and accuracy, API obtain values are prioritized where HTML based fallback is supplementing missing fields. On top of that, filtration happens to remove empty entries.

Afterwards, properly structure data is being investigated for storage options for further usage and in the final step stored data is being used for visualizing the rental prices obtained from this pipeline to show some statistics.

2 Objectives and Methodology

2.1 Objective

-Study and analysis of available information sources for tools and technologies
-Specifying testing scenario, what is required and final statistical analysis results
-Outlining the overall data flow and selecting appropriate tools for each partial step in the pipeline
-Implementation and testing the actual solution with the data extracted from real estate agency website for given region and time.

2.2 Methodology

Theoretical part of this thesis is based on the study and analysis of literature and information sources online which primarily focuses on topics like web scrapping, data analysis, data processing, data cleaning, data storage, visualization, ETL tools.

Whereas in the practical part, end to end process for scraping, storing, analysis, visualization of data obtained through extraction from real estate website is outlined and moreover for each step in this pipeline comparative analysis is conducted to select most appropriate tools and technologies for the chosen testing scenario which is implemented and tested using real data, the level of automation, optimum approach and integration with overall solution will be also analysed and measured. Based on this results conclusion is formulated.

The theoretical part of the thesis is based on study and analysis of available literature and online information resources with focus on topics such as web scrapping, data analysis, data processing and visualization and usage of ETL tools. In the practical part, a process for scrapping, storage and analysis of real estate data will be outlined and for each step a comparative analysis will be conducted to select the most appropriate tools / software. The proposed solution will be implemented and tested using real data. The level of automation, optimization and integration between various steps will be analyzed and measured. Based on the results of both thesis parts, conclusions will be formulated.

3 Literature Review

3.1 Data collection

Have you have ever wondered how to extract valuable information from websites quickly and efficiently, lets dive into it together. Web scraping is the ultimate technique that empowers organizations to gather large amounts of data from various online sources, enabling them to make informed decisions and gain a competitive edge.

When we say Web scraping, we refer to the automated process of extracting data from websites or other online platforms. It involves utilizing web crawlers or bots to navigate through websites and collect relevant information, such as text, images, links, or contact details. By leveraging this technique, organizations can save time, improve efficiency, and gain access to a wealth of valuable data that can fuel their data needs. (Jeffries, R, 2015)



Figure 1 - Webscraping process, source, (https://dev.to/barbaraulowee/how-to-make-money-web-scraping-2p93, 2021)

3.2 Web scraping

When we want to retrieve information web scraping is the way to go. In simple terms, it works in a way that you go to the website, load its content and take out the information that makes sense for your use case. It is the same process as opening a webpage and read what's on that page. The scraper you developed will send a message to website and it will get a response back which could be data and even pictures or videos.

The data residing in the website's comes in various forms and scraping is a good way to collect all these data. While it is possible to do manual scraping it is usually done with bots or programs. Which can handle quite complex tasks from simpler more generic less volume

data extractions to more complex data joins and high volumes and some can even handle webpage better. (Kimball, R., 2004)

Web scraping usually consists of two parts. Which is first for program to get to webpage and then to extract the information. For the first part program/bot sends message to website with content of requesting some for a particular page on website or some specific information. Once the message reaches to the website it sends back the response in form it was asked for which can be anything from text, pictures, videos or the webpage itself.

Moreover, after the bot gets the information it scans it filters for the information you flagged as necessary. Such as the title of the page, prices of things, address, phone number, email address etc. It then tidies up this information so that it then can be stored and used which is also the process what is turning the information on the website something that we can use.

For these two parts – one to get to the webpage and second extracting the chosen information – there are also tools that can click, fill out forms and auto perform the extraction with tool UI specified parameters.

Furthermore, these tools heavily depend on the users skillset and size of their project. To elaborate, some tools require coding knowledge which has capabilities like automated scheduling while other tools are for mostly non-coders where you can still get the and can help you do the scraping without writing code. These tools designed to help you get the information you need quickly. (Mitchell, R., 2018)

To summarize, web scraping is a way to get the information you want by making a request to the webpage for the specific parts of data you want and saving in a shape and form you find it makes sense.

Regardless of tools, bots, methodologies, programs the goal is one and the same one which is to get the information and transform it into something useful for your use case which can vary from searching of that information sort of like storing it into something query able or database, using it for developing models or applying different analysis and visualizations.

This is the core reason what makes web scraping so important as it is tend to be taken as source of information. (Reeve, A, 2013)

The idea of web scraping has been around since the early days of the internet when bac in then people used to call it screen scraping, web harvesting, data extraction. Nowadays, it goes mostly as web scraping and bots or crawlers stand for the programs that moves through many pages.

Web scraping can also be broken down to parts as, program usage to read a web page, loading the content and then get the data fitting to your filters. For instance, scraper programs are designed to send a request to a website and then receive data in whichever type the website delivers then analyzes and scans it to find the specifications of data you configured as necessary which could be product names, advertisement IDs, countries, regions, links, contact information. Subsequently, scraper program will gather the results and produce outout in a data file such as JSON, CSV, Excel, SQLite or even a database.

From broader perspective web scraper processing is much more complex then to 'getting response from website' because processes heavily depend on the website you are scraping and the specifications of extraction you configured. While some scrapers use simple HTTP requests, others can emulate a web browser to interact with websites which contains a lot of JavaScript, navigating between websites or icons. (Stephens-Davidowitz, 2022)

Furthermore, sometimes text analysis, pattern matching and data cleaning to extract everything is necessary. Thus, the way how web scraping works can be summarized as, a program pulling content of website, finds your data specifications and extracts them into a file.

If your understanding of internet is all about browser, this would be covering only a small fraction of all other things of it has to offer. Without a doubt web browsers are great at running code, arranging objects in human readable way and displaying images, web scrapers on the other hand offers the ability to quickly and efficiently run through large quantity of data. (James Densmore, 2021)

Moreover, web scrapers allows you to build a functionality to view database that spans thousands even millions of pages in seconds instead of manual approach of viewing each page individually.

Another functionality is in some occasions where standard search engines cannot reach to certaion data points web scrapers are on top of that too. Search for “Flights from prague” on a standard search engine and you will see bunch of advertisements of popular flight selling sites, most search engines lack detailed metrics that a good web scraper can show the changing cost of flights from Prague across multiple webpages. (Andreas Kretz, 2018)

From a different perspective this brings the question of their difference from the APIs. What Application Programming Interface or API is essentially, small-scale program that provides available stream of formatted data from one to another computer. It can be found for various kinds of information such as X feeds, wikipedie articles or Facebook and depending case to case it can be a way better option then for a bot/crawler to do the same work. (Piethein Strengholt, 2020)

On the other hand, it can not always be relaible option in such cases like, collecting small, finite from multiple websites which is missing API although information you filtered is fairly small sources may not have the infrastructure or technical ability for an API.

It could also be that, information you are trying to extract is valuable and intended for limited usage. Even an API can exist, provide the number of requests, have the limitations on rate but then the data available may not be in the format you wanted.

This brings us to yet another scenario where web scraping comes in, anything you can see in your web browser you can pull through a Python script. Anything you can pull through a script you can store in a database. And anything you can store in a database you can do almost anything with it. (Jan Kunigk, Ian Buss, Paul Wilkinson)

An extension of ability to access limitless data is practical applications. When we have access to limitless data market forecasting accuracy increases as well as ML translation.

Regardless of the area we work in, web scraping almost always increases the efficiency of business operations.

3.3 Tools, methods and libraries

Web scraping can be approached in many ways from a very manual, one-off copy and paste method to far more sophisticated software systems. Choosing an approach will depend on the type of site, the output format wanted and the amount of data needed. Some of tools and approaches which are used when processing data from the web and how they are applied in practice are. (Ramez Elmasri, Shamkant B. Navathe, 2015)

3.3.1 Manual data collection

The oldest method of extracting data from a website is still the most basic. An individual, depending on the size of the extraction, browses to a page, highlights the necessary information and pastes it into a spreadsheet, database, word process document or other file.

This approach does not require any programming knowledge and is suitable for extremely small or one-off data collections, or ones which are exceptionally tricky in their layout or nature and require human interaction.

It is very slow though, prone to human error and is invariably impossible to create a long-term maintenance process around since it quickly becomes infeasible to review periodically or process large quantities of data.

3.3.2 Pattern recognition (regular expressions)

One of the simplest ways of processing data from a webpage is using regular expression or other pattern matching techniques to locate and capture pieces of data which have a reliably defined layout and appear consistently throughout a webpage (such as phone numbers, email addresses, dates, etc).

Regular expressions are usually very fast and reliable for simple extractions, but can be very cumbersome and prone to failure when the layout of a webpage changes or a data field appears in slightly different formats.

3.3.3 HTTP requests and web communication

Fundamental method used by many crawlers involves HTTP programming which entails sending requests to a web server and grabbing the page automatically. With HTTP programming, a Linux command line tool (wget) or a programming library (Python requests) might be used to load pages, send forms, attach headers, persist sessions and use cookies.

This method is very effective for websites that output static HTML or list data through predefined endpoints returning JSON or XML. Since most HTTP requests consist of "getting the page", a good understanding of HTTP programming forms the base of most crawlers.

3.3.4 HTML parsing (extracting from page structure)

once the HTML code is fetched, the crawler must parse through it to identify and extract specific objects. HTML parsing can be achieved using libraries such as beautifulsoup (Python). These tools convert HTML documents into a hierarchical object structure that you can search for tags, class names, IDs, links, tables and nested objects. HTML parsing is one of the most common crawling techniques because they are very flexible and can handle static websites that contain all elements in the source code.

3.3.5 DOM interaction for Dynamic Websites

Many modern websites generate the page content dynamically through JavaScript which cannot be seen in the raw source code. For these types of websites, the crawler may use DOM parsing and browser automation techniques. This involves using tools such as Selenium to automate a real web browser (e.g. Chrome).

Web page interactions such as clicking buttons, scrolling and waiting for content to load can be achieved and the complete rendered web page can be extracted. This method is fast and

suitable for interactive sites, infinite scrolling websites and those that request user interactions.

3.3.6 Vertical Aggregation (Gathering data from many similar sites)

Another scraping method is vertical aggregation where the goal is to gather data from many similar sites in the same industry or category, creating a single combined dataset. For example, a price comparison aggregator will scrape many online retailers to get product prices, descriptions, and stock.
This kind of scraping involves collecting data from multiple websites, then cleaning and formatting it so the final output is consistent. Vertical aggregation can be applied to price comparison, market research, competitor monitoring, and directory building.

3.3.7 Using semantic markup (Structured data extraction)

Many modern websites provide semantic metadata to make it easier for search engines to identify the content. This data, often embedded in the page as JSONLD, RDFa, or microdata, can be read directly by your scraper, providing structured data fields such as product name, capacity, price, ratings, publish date, etc. Moreover, recognizing and utilizing semantic markup can improve data quality and reduce parsing requirements, provided the site offers it.

3.3.8 Visual scraping using computer vision

In cases where you can't simply extract data via parsing text or HTML content such as the data being embedded in images, dashboards, animations, PDF documents, or visually complex interfaces computer vision techniques can be employed.

Using OCR, you can detect text from images, and using scene or layout analysis, you can identify tables or graphical areas. While it is more challenging, this method allows extraction of data from otherwise inaccessible sources, but it may demand more effort to process and verify.

3.4 Data preparation and cleaning

This is the stage where extracted raw data goes through transformation pipeline in order to have ready to analyze dataset. First step is to understand the usage of the extracted data, how it wants to be used, what is trying to be achieved with it, what is the requiered form of data, what analyzes will be performed. Once we answer these questions they will become the driver of all subsequent processes. Mistake on those is a quick way to generate false reports. (Steve Hoberman, 2009)

With cleary answered questions, implementation can start which includes hands on preperation of cataloging the collected data, refresh frequency of it and table relations. Combination of multiple sources where each has different updates and granularity levels is crucial.

Moreover, most of the preperations involve extracting the data and placeni into staging zone that is separate from source inputs and hence auditable and reproducible and ready to be used in the analyzes workflow.

Afterward there follows shaping the structure of the data, making sure the transformation of types so that date, timestamp, number and boolean types are properely presented, make changes so that similar categories have similar schemas, associating currencies and measurements points and anchoring time series to a common standard so you're not translating between timezones during analysis, and standardizing the handling of timestamps and indexes so we do not shoot ourselves in the foot by mixing resolutions, datatypes or similar. (Len Silverston, 2011)

Furthermore, selection of a representation for complex objects such as nested JSON's or many to one repeated attribute, normalized into separate tables, denormalized, exploded into multiple rows or flattened into multiple columns. (Lawrence Corr, Jim Stagnitto, 2011)

After we manage to shape data into consistent format, next stage of transformation is to make data both actianable and relevant to the use case.

This may involve removing irrelevant data, selecting relevant ones, specifiying the use case, finding appropriate granularity, constructing variables that will present all different aspects of business logic.

Often then not in analytics, preperation phase brings us to one tidy, easy to use and visualize dataset. Bringing attention back to the importance of the use case, in ML the preperation steps may also include feature engineeringand preparing a training set and efforts to avoid data leakage, as data should not be present at prediction then later on influencse data used to train a model. (Christopher Adamson, 2010)

Another activity on preparatory step is enrichment with joining reference tables and sometimes additional external context when it helps interpreting the data. Following part in preparation step is validation to detect what is incorrect. Those could be actual uniqueness of unique key segments etc.

Good preperation also means good documentation, a data dictionary describing what each field holds, what your formula columns mean, how filters and exclusions were used, the end date of your data so you can reproduce your work tomorrow and in six months from now.
It is some way to make sure that any user reading the data understands what it represents, that the numbers are actual, and above all else, that it corresponds to something we can find in the real world.

Data cleaning is a more specific subset of data preparation, specific to data quality. Data cleaning is done to improve the accuracy of data, where preparation is more focused on transforming data to the required format.

Cleaning begins with an understanding of "quality", since most data sets are designed for some specific purpose, and what is "quality" depends on the use case.

In a design of a dataset about marketing campaign a single invalid address might not be tolerable or it could also be a field in a forecast model which has hundreds of fields so it would not have major impact. More straightforward issue is about preventing the invalid values or inconsistent records from skewing analysis, but it is also important not to overclean

so that you don’t lose any valuable information in a way that it causes bias or changes the story through analysis. (Pramod J. Sadalage, Martin Fowler, 2012)

Simultaneously, missing data is yet another issue and it is very important to understand that what is that value supposed to be and treat it carefully, considering that it is not necessarily random. Among the reasons of why those values could be missing are, system outage, changes and differences between user groups and removing rows randomly can produce bias in results.

Cleaning method includes more sophisticated approaches as well, such as, deleting row which has no to low impact and adding N/A categories to value that are definable. In model development putting flag to highlight missing values is a way to maintain that data. The way it will be handled is decided upon the reason why these values are missing and how it causes vulnerability to architectural flow. (Wes McKinney, 2022)

Duplicates can also cause various issues thus they are also part of what data cleaning aims to do, which could be complex to solve time to time. We need to define what constitutes a duplicate for our domain repeated customer profile, same order ID twice, or simply identical log entries. (Jacqueline Kazil, Katharine Jarmul, 2016)

Therefore, duplicate cleaning can indicate the need of rules implementation for filtering records when it comes to saving them. To illustrate, profile with most accurate and full data, latest refreshed one or profile with the most correlations.

When we are dealing with it we need to, resolve the entities, figure out when the match happened between two records whether it was based on set of matching rules, they are actually the same person or use another probabilistic technique.(Hadley Wickham, 2014)

Furthermore, there is also inconsistent categories and bad formatting. There usually exists a lot of variants of the same category. Text field could contain hidden or special characters. Format and time zone of date columns also tend to differ.

Numeric field could contain non-numeric symbols, it could be being measured in different units. The normalized canonical value with cleaning rules and mapping tables fixes these. Having explicit mapping for these will help support keeping the data clean, transparent and repeatable. (Lee Baker, 2019)

Therefore, the approach we design for data cleaning needs to be aware of the domain with cross variable checks to compare unit prices to totals, when discount applied should not be more then total cost, comparing the start and end date, making sure the links used are valid and validation about order number association with real customer are couple examples to that. While deleting, best practice is to flag and explore them and only fix them when you conclude that they do need be based on your findings. (Dorian Pyle, 1999)

In analytics, it's optional whether to keep outliers but best practice is when you do their impact must be controlled and lowered. Whereas, in modelling it is possible to minimize values or give them low weight scores to limit rare outlier effects.

To summarize, well performed cleaning data should be reproducible. Using spreadsheets to edit them might first look like a good idea but this will come with difficulties like hard auditing and easily forgettable, even more so if the data is getting refreshed.

Thus, it should be implemented in a way that it can be repeated over new data as scripts, queries or pipeline tasks and those must contain good and clear logging of the changes, effected records and applied rules.

Moreover, data quality checks should run on top of them ideally also automated way, with settled logs to give alerts of errors or break such as, spike in missing values or new categories popping up. In standard operating procedure of the workflow of cleaning, issues should be made visible and manageable.

Results should produce trustworthy and consistent data and documented approaches what reduces the error and increases the confidence when we want to make conclusions from the dataset.

3.5 Data storage

Data storage is the output after completing the extraction, cleaning and preparation steps in data flow pipeline which allows data to be safe, relevant and performant, which is not only where the data ends up. Additionally, it is the architecture of data and its storage is relevant to the format of the design of flow, its security, access and retrieval, maintenance and scalability. Well stored data ensures effectiveness of reporting, dashboard and things like machine learning while avoiding the data loss or leaks that could occur without elastic architecture. (Edward R. Tufte, 1986)

First thing to consider is the type of data you have, tables, files, logs, video, audio, images etc. and their use case. Analytics requires different infrastructure than transactional systems, for instance.

One of the primary point in data storage is choice of method when storing data and for it to match the workload of the relational database which requires consistency and clear defined relationships, read of heavy queries on terabytes of historical data happening in a data warehouse for analytics workloads, raw and semi structured data being put into a data loke or object store for deciding on analyzes in next steps. (Cole Nussbaumer Knaflic, 2015)

Deployment of storage architecture and the layers of it is as important as the way data structed within a storage architecture. A good design should be unpollutable and easy to find and use. Which is nothing but establishing standards of names and directories or database schema layouts, along with rules for partitioning and indexing.

Partitioning by such as date range, geographic area or industry can increase performance and reduce costs by minimizing the amount of the data that needs to be queried.

Indexing on the other hand can make accesing to data quicker process in database management system which will come with impact on storage requierements and update time. Architecture includes things like definition of keys and relationships which helps proper join of records.

Security is another core aspect when it comes to data storage, especially when it's the case of sensitive data. Access controls, encryption at rest transit, separation of environments, policies for legal and compliance reports, tokenization of sensitive datasets, minimizing risk while being able to perform analysis. These all will minimize the radius of loss using least privilege and traceability.

Another one of the objectives is reliability. Data can be lost due to reasons of incorrect human intervention and errors in data pipeline. Best practice in data storage has therefore always included several safeguards such as backups, version, replication between zones or regions and a document recovery plan.

Overwriting of existing data and processing steps from subsequent data is common in analytical applications but prevention is possible with creating new versions. Moreover, it helps enable repeatability of the analysis and enables you to extract audit trail, when you are asking questions about the data it is possible to determine the answer find out how it was generated and from which inputs. (Stephen Few, 2006)

As data volumes increase storage design needs to evolve, balancing the need for productivity and cost. Without structuring this can lead to high costs, slow responses and high maintenance overheads.

Cost is controlled by selecting appropriate big data file formats and compression, archiving older data to cheaper storage classes, deleting duplicate and unused copies and removing data that is no longer required.

Performance is managed by examining query response times, storage growth and access proportions and optimizing partitions, indexes, clustering and caching accordingly.

Crucially this should be driven by real needs, the most elegant design is not necessarily the one with the most features, but the one that best meets the current and aspirational requirements.

3.6 Exploratory data analysis

Exploratory data analysis is performed to understand what data we actually have, its behavior and stories it is suggesting.

Part of EDA is developing an instinct for the basic infrastructure, what is the shape of the data in terms of numbers of columns and rows, the nature of each column, the nature of each column's values and how well that fits the unit of analysis you thought you would be addressing. (Claus O. Wilke, 2019)

Even cursory inspections here could catch huge mistakes (for example, a researcher using repetition to increase sample size, assuming that this is acceptable as independent data points, time zone misalignments, mixing populations rather than comparing them). Perfection isn't required; just enough familiarity to avoid mistakes or get reoriented quickly. (Kieran Healy, 2018)

A large focus of EDA is quantifying and illustrating variables in a way that illuminates their shape, structure and quality. For numerical fields, that means looking beyond the mean and median for normality and into the distribution, the varying degrees of spread and skew and multimodality.

Categorical fields that mean inspecting the distribution of values via frequency counts, common and uncommon categories, labeling inconsistencies or whether columns change behavior by group or by condition. Special care of missing data is necessary and amount that it exists. Throughout this process we look for outlandish values and check patterns that may include instrumentation problems.

Exploratory data analysis comes into play especially when you begin looking at relationships. When we look at how variables are related to each other, how slices of the data compare to each other, whether there’s evidence that groups deviate from expectations, we are looking at relationships.

We check correlations, compare distributions, test for nonlinear of threshold effects with averages or other margin estimates that are not as simple as mean. When we work with time series data, EDA usually refers to identification of trend, seasonality, spike and regime changes since interpretation of any model can be indiced by time and entirely change.

For this case specifically graphs are great option because clusters and breaks can be a lot easier to see in the chart then the numbers. At this stage you are not trying to establish anything, only effort is on trying reduce your scope of reasonable explanations.

When we think about exploratory data analysis it is also important to consider if the workflow is feasible when it comes to the data. We need to ask ourselves if our claims are descriptive if the variables we use are comparable across segments and if my definition of the measurement is aligned with the concept. We also need to check if the outcome is too imbalanced if there is an agent or driver missing from the data and if the measurement definition has changed as we moved from the pilot to the project. Exploratory data analysis is crucial here. (Foster Provost, Tom Fawcett, 2013)

There are two ways to learn from data: quantitative and qualitative methods. The best way to do this is to combine exploratory data analysis and these two methods. Quantitative analysis is about measuring things looking for relationships and documenting the results. It tells us how much how often how long and how strong something is. It also checks if the conditions are likely to be significant or just due to chance.

On the other hand, quantitative analysis focuses on settled definitions and statistical frameworks to understand what the data means. This type of analysis is mostly based on definitions and structured data. It is also concerned with reducing behavior or phenomena to a set of numbers. Quantitative analysis is ultimately concerned, with understanding the data in a way and exploratory data analysis helps us to do that.

Qualitative analysis are mostly focusing on what a measure means in people's opinion and afterwards in order to discover and explore the language, story lines and documents people write or produce, it is based on interviews, surveys, collected responses, and observations, artifacts, behaviors and situations.

In quantitative work, to avoid false conclusions which are derived by fuzzy definitions, we start with well-defined variables and measures. Moreover, after having measures and definition appropriate, we can start with comparing groups, following change over time, test relationships between variables, and quantify the levels of uncertainty. Testing of hypothesis, significant differences, multiple variable controls and time series models that explore trends.

Furthermore, a way to good quantitative analysis needs to follow logics standards to find sources of bias which will undermine results confounding and affecting your data, its long-term usability and error measurement. Automatically meaningful is not significantly emphasizes quantitative work. However, data quality and casual pathways are while the effect of it still subject to problems.
The culmination is a set of numeric results that conform to standards so that others will be able to validate or build.

From the view of qualitative analysis, it’s more for depth, nuance and complexity. It might begin with a data transcript, focus groups field notes, support tickets, user recordings or written feedback then finally to analyze that data through coding, thematic analysis, narrative analysis and theory.

Qualitative analysis looks for embedded patterns, model, inconsistencies and contextual factors which effects the behavior instead of looking averages. Unknown variables are identified this way to measure and explain what words with spelling mean and reveal the underlying mechanism behind a quantitative pattern.

Moreover, development of thematic analysis and controlling for alternative explanations is possible when performed a well documentation. The ultimate practice resembles a human elucidation of context and measurement causalities.

The strength of qualitative and quantitative approaches together is that they can compensate for each other, with data from one reminding the other of what to look for or becoming a basis to test the other's theories.

Quantitative can provide the scale to know if we are seeing a genuine phenomenon or an isolated anomaly, while qualitative can set the scope for what phenomena we should measure. The combination "mixed methods" approach is especially recommended when decisions matter a lot, because it ensures you are not optimizing for a metric which does not accurately reflect people's lived experience.

Typical is a process of doing qualitative initial research to develop a model, then using quantitative data to identify priorities, and finally back to qualitative methods again to examine people's lived experience in focused areas to refine intervention ideas. In practice, the most rigorous results come from triangulation using multiple indicators converging on a coherent story and transparent reporting of methods, assumptions, and constraints so your fellow researchers can judge your conclusions.

3.7 Data visualization

Data visualization is the graphical representation of data using charts, maps, tables, dashboards, diagrams to help viewers see patterns, relationships and trends more easily and quickly than they can from the underlying numbers.

Good visualization is never merely decoration, it is a thinking tool and a communication tool, a way for you to find out what the data is revealing as you analyze it, and a way for you to communicate what is important to others as you help make difficult decisions. Since our visualizations influence our viewers' understanding, the best ones are always honest and clear before they are beautiful.

A good visualization depends on its purpose and its audience. The ideal chart varies depending on the action you want the viewer to take, compare one value to another, see a trend, get a picture of composition, spot an outlier, or investigate a relationship.
The chart also varies depending on the audience business managers looking for a quick "business story", analysts wanting detail. If the choosing of the wrong chart kind, then true stories are hidden, or in the worst case, a false story emerges. (Cathy O’Neil, Rachel Schutt, 2013)

For instance, trends over time are generally best with lines, category comparisons are generally best with bars, and associations between two continuous variables are generally best with scatterplots. The problem is to find a form that reveals a message rather than hiding one.

Good visualization requires clean structure and consistent definitions (see below) because, unfortunately, much visual confusion just reflects a deeper data confusion. Before you construct a visual, you need to verify comparability of measures (e.g., same units, same time window, same definition of groups), verify aggregation suitability (e.g., averages do not equate totals, and ratios do not equal raw data).

Common sources of false clouds of data are comparing apples to oranges, ignoring seasonality, reporting totals instead of rates. A visual should allow accurate reasoning a drop of a stamp on the axes is usually all it takes to avoid unintentional inference.

Design decisions impact recognition. Visually communicate what the question is, so that the viewer does not have to guess. Select scales to help promote accurate comparison. Use axes to not over/under emphasize differences. Avoid clutter such as too many lines, categories or annotations which diminish the art and difficulty of the chart, allowing the main point to be missed.

Flat simplification can also be a weakness when unnecessary information complicates the signal. A strong visual will highlight what is relevant without stripping away the ability of the viewed to cross check their assumptions if the perception derived from charting guidance’s decision making.

Visualization is particularly helpful for detecting patterns that would be difficult to discern in a table. The shape of distributions can sometimes reveal skew, heavy tails and multimodal behavior that a simple mean will hide. Visualizations of time series data can make cycles, breaks and slow trends apparent (and adjust your forecast or analysis accordingly).

Plots of relationships may reveal nonlinear effects, clusters and outliers, which may imply that different segments behave differently. During exploratory work, visualization is an

iterative technique you 'plot, see what you didn't expect, adjust the view and plot again'. If you are responsible in this process, this is not 'fishing', it is how you develop an intuition for the data and learn which questions can be tested statistically.

4 Practical Part

4.1 Introduction

I have used a script that implements a web data extraction workflow from Sreality.cz website that is optimized for speed, repeatability and robustness under real world variability.

Instead of relying on a single technique (such as, just browser automation or just HTML parsing) my script is using a layered approach

First off, search result discovery is performed via direct HTTP request and followed by HTML parsing which has detailed page extraction feature with fall back option to HTML scraping when necessary.

Other targeted fallbacks along with normalization and alias mapping is applied to keep consistent and accurate for use case data set result.

On top of that, there is thread-based parallelism used to scale throughput so that it remains practical when collecting many listings from the real estate webpage.

Lastly, there is also connection pooling and shared session used to reduce the overhead support high parallel request volume.

Subsequently, data set extracted goes through the cleaning and preparation steps where special characters, whitespaces and hidden characters handled in the same script.Moreover, inconsistent data formats, data compatibility issues, duplications, naming inconsistencies are being handled.

After these steps, data is being saved in CSV format which is then being converted to excel. From there on visualization is being performed in the same excel file to better present the data for audience.

4.2 Selection of tools

In this chapter details of comparative analysis of each chapter will be performed and described.

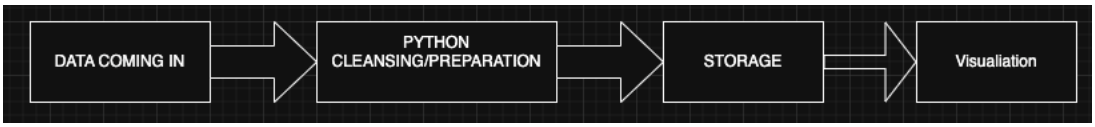


Figure 2 - Stage diagram, source: (Own drawing,2026)

For each step in this end-to-end flow, I performed comparative analysis of available tools and technologies for my specific test case scenario. Options were explored for data extraction, available libraries in python, low-no code tools and their functionalities in practise for determining beast approach. Afterwards, options for cleaning and preparation were explored in the same fashion and most efficient way was chosen.
Furthermore, different format for storage of the result set data were researched and most straightforward and logical option was used for visualization as well.

4.2.1 Comparative analysis of data extraction stage

In my test case, I needed to have one time snapshot of listings on Sreality.com website where I filtered for rental apartments that are in Prague, which would extract all available advertisements in the moment of extraction instead of script running perpetually which would monitor and extract site in real time. Thus, fast, controllable, lightweight approach of HTTP requests (Python requests) and HTML parsing (Python Beatifulsoup) not only fitted the constraints but additionally it mapped well to common pattern of interaction used by most listing sites, predictable URL, filter, server response, parse the relevant chunks and forward the results.

When I were to compare that approach to alternatives, outstanding difference was that most tools are designed to avoid common constraints so they add cost or don’t bring a clear benefit.

One know old but gold method is the manual copy and paste, while it technically works it fails straight away on practically. It is very slow, since its all about human-manual task it is error-prone and almost impossible to scale for all the data especially when extraction from multiple webpages. On top of that, it can cause differences on the same data collected by different people due to missed pages or different sorting of pages.

Furthermore, when I looked at regex-based method directly on raw HTML I found out that, it can be quick for simple patterns but it can be inaccurate for complex structured websites because HTML changes often and nested elements thus break regex rules.

Moreover, when I experienced where content block included multiple nested tags and attributes, Beatifulsoup was much more maintainable due to its features of using tree structure.

Full crawling framework such as Scrapy(Python) or Nutch(Java) was another option, which are quite good for cases requiring good concurrency, auto throttling, retries and distributed crawling and link discovery. Particularly when working on large crawls on regular basis which scales into a lot of pages.

Which was exact need of my case, since I target only single domain, one filter and it will be one time snapshot. Whereas, scrapy is a framework that requires more project scaffolding, additional considerations that I did not needed.

Method which is more common in today's technology world is browser automation (such as Selenium, Playwright), where real web site is actually visited to scrape. Browser automation replicates a user using the browser, loads pages and scrolls through them, clicks and looks for elements and waits for JavaScript to run. Automation of it is where content load in dynamic fashion and content generated after user driven actions or when DOM is accurately predicted.

But for my test scenario it was not correct approach because rendering takes a lot of time, in the meantime it consumes a lot of resource. In addition, it is very brittle for example, if you change title or label the script changes, if you change ID of an element in DOM script breaks

if website shift layout even few pixels script again breaks. Moreover, it increases the maintenance cost, for my one-off snapshot browser automation was unnecessary escalation of effort.

More technical way is to scrape behind the page, in other words capture the website fetch requests and pick well formatted JSON data which can be good method when website already offers internal APIs this can be good approach when it delivers that. Along with that, for my test scenario this was still not the best approach due to fact that it involved private endpoints, cookies and tokens handling.

With my goal of one-time snapshot extraction which the needed field were being extracted by HTML parser searching for APIs costed time and possibly less robust solution considering that endpoints can be rate limited or protected where page fetching is not.

One other approach would be to use those public APIs but since most real estate websites do not have official public APIs or even if there is they do not give access/private and sometimes limit the fields or requests. Sometimes, they do exist but don't come with particular filters or data. Thus, it was not feasible solution either.

Finally, no-code and low-code scraping platforms (import.io, brow
ser extensions and point and click scrapers). First of all, they are great at closing the gap of skill when for some when it comes to programming and quite ad hoc.

However, this also was not ideal method in my specific case as they tend to be less transparent, controllable, costly and occasionally less reproducible even when website has changing structure. In addition, they have the issue of integration for example when the solution is desired to be integrated into a data pipeline it is very difficult to have consistent result with right formats and cleaning rules.

To summarize, requests and beatifulsoup was just the right fit. It matched the requirements of my practical case where I needed, single-site, clear filter, one-time snapshot and efficiency.

While all the other alternative approaches were strong candidates they were design more for different use cases such as large-scale crawling which are optimized for scale, browser automation for dynamic rendering which costs speed and stability, APIs which offered somewhat structured data but added extra fragility and no-low code tools were good in terms of accessibility but they decrease control and auditability. Therefore, for my use case this approach was the most balanced and rational engineering decision.

4.2.2 Comparative analysis of data cleaning and preparation

In this stage, I found it quite practical to use Pandas and regex, it was a good match for the dataset I had, where my data was already in mostly usable shape but of course still needed cleaning, standardization and normalization.

Pandas is essentially the “workbench” for tabular data in python. It allows loading the scraped data output, makes sure of schema consistent, helps remove duplicates, handles missing values, redefines columns and efficiently creates clean results while keeping it reproducible and audit friendly.

Regex, on the other hand, complements this in a great way because scraped fields contain often mixed-up text and regex gives full control to isolate components without needing heavy natural language processing.

Alternative approach to those is SQL based cleaning, which is not very common but excellent for filtering, aggregation, joins but messy for scraped string SQL becomes verbose and less convenient than Pandas since also you would need to setup a database which does not add much value when cleaning is mostly column wise transformation.

Another alternative is to use data-wrangling libraries and tools such as OpenRefine or Trifacta platforms. Essentially, they offer options to explore messy datasets and interactive transformations. Additionally, cleaning rules are direct and use fully scripted, repeatable ones which integrates directly with scraping results.

Moreover, in the case of more complex frameworks such as Apache Spark(PySpark) or Dask which can process big datasets in parallel but it is not really necessary since it is a one-time snapshot from a real estate website about rental listings. Since they also come with cost of substantial amount of setup complexity, dependency and cognitive load it is only good to use when we actually have too big dataset or there is a need of distributed computing.

Furthermore, in this topic one more method is schema validation and modelling tools like Pydantic and Pandera. They are great tools for rule enforcement and filtering this way and also for anomalies detection. On the other hand, since my goal more for cleaning and preparing data for analysis and visualization there is no real use case for building a data product.

NLP libraries (spaCy, NLTK) stand yet as another method for parsing. When the case includes entity extraction, sentiment or topic modelling but it adds additional runtime and complexity without outstanding benefit. String method could be replaced with regex rules but it is usually more robust especially for patterns varying across listings (unit symbols etc) so it decreases failures on edge cases and thus explicit transformations happen.

All together, my approach of using Pandas is best for this test scenario because it provides fast, clear and tabular manipulation for a dataset which fits into dataframes. Subsequently, regex messiness of web text efficiently which is a common case of scraped fields. Look at other alternatives, they are either manual, heavy to operate or database centric. To conclude, usage of pandas and regex delivered the best and efficient dataset which was ready for analysis.

4.2.3 Comparative analysis of data storage and visualization

Final stage of the flow is data storage and then visualization, where I used CSV for storage and Excel for visualization which was a very practical and coherent choice for my testing scenario to produce one time snapshot of Prague rental apartment listings from Sreality.cz, after cleaning and putting into structured tables it was the time for visualization of results directly in Excel.

Moreover, in this testing scenario Excel is the “delivery package” itself which enables pivot tables, charts and sanity checks without need of any additional software. It preserves data types consistently, allows multiple sheets in one file and keeps everything basically in one place for both exploration and presentation of data via visuals.

In comparison, databases are great for pipelines, querying, historical datasets, restricted schemas, however my scenario avoids these overheads due to the nature of it. Where in the case of databases there would be a need to set up schemas, manage connections and loading of data. SQLite for example although its lighter still requires most of steps like in databases and it’s not very presentable for audience.

Another alternative is JSON, which is ideal for nested structures and APIs but needs flattening before it becomes chart friendly.

Parquet are perfectly suited for large scale analysis in python or Spark, because they are compressed and quick to read but they are not native Excel first formats and would require conversion back into Excel anyway. In an Excel visualization workflow, this is an unnecessary additional step that does not add value.

Python native formats like Pickle do not pass the interoperability test. In addition to being not readable by Excel, not great for sharing, and not a good presentation artifact, they are in no way better than Excel for presenting in this context. Similarly, no-code BI tools like PowerBI or Tableau would provide more dynamic dashboards and visuals; but requiring other tools and deployment effort is unnecessary here.

Goal is for me to produce clear reporting and exploratory charts, Excel is the quickest route from a cleaned raw dataset to a beautiful visual, especially given the audience's familiarity with the tool. Even cloud based data warehouses and lake house solutions (BigQuery, Snowflake, S3) are way too much in this situation. They are perfectly deployed when volume, refresh frequency, governance, and collaboration time outnumber the task complexity; but I am not trying to optimize scale here.

4.3 Implementation

Primary purpose of my script is to create structured dataset containing set of variables specifically price, energy efficiency, area and date the listing was posted. In practice, it is working in two stages. First stage is the scraping and extracting data from the website, whereas second stage is cleaning and preparation of data.

On the data retrieval instead of relying on one source my implementation uses combination of both HTML parsing and API-based retrieval. Which is dependent on data that is being retrieved.

To manage these requests efficiently, I use reusable sessions provided by the requests python library which prevents opening new connection for every single request. The session is configured via headers such as User-Agent which helps scraper interact with website like its normal browser. Moreover, the connection pool is extended with help of HTTP adapter, so the script can process large number of parallel requests in the flow.

Extraction starts with filtered results of apartment rentals in Prague which is paginated. Every page's html content gets downloaded and parsed with Beatifulsoup. Afterwards, anchor element search is performed which containing hyperlinks forwarding to property detail pages. Script converts them into URLs. Also repeated processing of same listings is avoided by removing duplicate links.

Subsequently, once detail page links are collected, advertisement by advertisement extraction starts by function written which I called scrape_detail. Primary feature of it is first try to get property data through site API (I have mentioned it in comparative part, this stands as additional option.) and extracts the ID of the ad from the URL and via that creates the API request.

This is a better approach then scraping values directly from text becuase via API data is in structured JSON format and from the response of that script captures target fields. Though, it does not fully depend on the APIs when some fieldds are missing or incomplete it aims for HTML of the ad parses it.It looks for <dt> and <dd> pairs which are in most cases for

labels and values. If this also does not work then it goes for another alternative that seaches divider-based section in page.

Furthermore, when the collection completes, filteration happens for the final dataset. Additionally, name mapping functionality was added because website tend to use differentt labes for same concept thus it produced the desired results. To illustrate, area appears as usable area, area, total area similary date appears as posted, update, displayed so this way column names are standardized.

```
Def scrape_detail(url: str) -> dict:
    # Prefer JSON API detail when possible for reliable fields
    m = re.search(r"/(\d+)(?:\?|$)", url)
    api_row = {}
    if m:
        estate_id = m.group(1)
        api_url = f"https://www.sreality.cz/api/en/v2/estates/{estate_id}"
        api_resp = SESSION.get(api_url, timeout=30)
        if api_resp.ok:
            data = api_resp.json()
            items = data.get(„items“, [])

            def find_item(name=None, item_type=None):
                for it in items:
                    if name and it.get(„name“) == name:
                        return it
                    if item_type and it.get(„type“) == item_type:
                        return it
                return None

            price_obj = data.get(„price_czk“) or {}
            price = price_obj.get(„value“) or price_obj.get(„value_raw“) or „“
            if isinstance(price, (int, float)):
                price = f"{price}".replace(„,“, „ „)

            energy_item = find_item(item_type="energy_efficiency_rating")
            area_item = find_item(item_type="area")
            posted_item = find_item(name="Update") or find_item(name="Posted")

            api_row = {
                „price“: str(price),
                „energy_efficiency“: energy_item.get(„value“, „“) if energy_item else „“,
                „area“: f"{area_item.get(‚value‘, ‘‘)} m²“ if area_item and area_item.get(„value“) else „“,
                „posted“: posted_item.get(„value“, „“) if posted_item else „“,
            }

    # Fallback to HTML scrape if API missing or incomplete
```

```
    resp = SESSION.get(url, timeout=30)
    resp.raise_for_status()
    html = resp.text
    soup = BeautifulSoup(html, „html.parser“)
    all_kv = extract_kv_dtdd(soup)
    row = pick_fields(all_kv)

    # Fallback for price if missing in dt/dd
    if not row[„price“]:
        price_el = soup.select_one(„[class*=price]“)
        if price_el:
            row[„price“] = price_el.get_text(„ „, strip=True).replace(„/month“, „“).strip()

    # Fallback for area if missing
    if not row[„area“]:
        h1 = soup.find(„h1“)
        if h1:
            text = h1.get_text(„ „, strip=True)
            m = re.search(r“(\d+\s*m²)“, text)
            if m:
                row[„area“] = m.group(1)

    # Fallback for posted/update if present as meta
    if not row[„posted“]:
        meta = soup.find(„meta“, {„property“: „og:updated_time“})
        if meta and meta.get(„content“):
            row[„posted“] = meta[„content“]

    # Prefer API values when present, otherwise HTML fallbacks
    merged = {}
    for key in [„price“, „energy_efficiency“, „area“, „posted“]:
        merged[key] = api_row.get(key) or row.get(key, „“)

    return merged
```

Subsequently, the script also does cleaning and formatting of the data before saving the results into the CSV file. While these steps are not very complicated in themselves, without some preparation, the raw web data would not be very consistent in terms of formatting or structure, which makes the extracted values much more difficult to analyze or categorize later.

A simple first cleaning task used in the script is to unify the naming of extracted fields. On real estate pages the same attribute may be named slightly differently depending on the page as afore mentioned. To get around the unification of the texts used as keys in the dictionary,

the text is normalized by deleting gratuitous space, converting to lowercase and removing trailing colons.

The values are cleaned also before writing the output file. Valuable extracted from HTML may have some formatting characters in the text like nonbreaking spaces or added white space around the value. The script removes those characters and reduce the text so that output file looks better. It may not be a great change, but it avoids small formatting issues that cause trouble later when the dataset is loaded into another application like Excel.

Picking only the most relevant variables is another way to prepare. The scraper would go on to find many other pieces of information on each listing page, but the script would only take four fields of info (price, energy efficiency, area and posted date). This reduces the amount of information stored, and makes analyzing data easier later on.

Another correction made by the script is a slight cleaning of the price field. Sometimes the text "/month" is added after the actual price after it has been extracted. This additional text might not be strictly necessary depending on the rental classification for all listings on the page, so it's removed here so that only the correct price is saved without requiring further processing of the file later.

Yet another step in the preparation process is to keep the format of each row consistent. This is useful because the code starts by creating a blank row that already contains all the columns needed for output before any data is parsed to the variable. If a key value cannot be found for a listing, that will be left blank (empty string) rather than missing altogether in the output file. Doing it this way results in a more reliable dataset in the end because it becomes table like. Each output value has the same meaning and is located in the same position in each row.

A final correction simply prevents duplicate extraction because a specific listing URL may be used more than once in a search. In practice, the href value from an anchor tag is stored in a set after first being discovered. Any subsequent listings with the same link in their address are ignored, so the script never writes that detail to the CSV file repeatedly. Duplicates can distort measures of central tendency, for example, if they are present among

the data to be summarized later on. Another way of fighting duplication is filtering out blank rows. During this session of preparing the data for the database, once the second step of field extraction has completed, the code checks to see if the row (list of column values) contains at least one example that is not blank. If everything is missing, the record isn't even written to the output file.

```
def extract_kv_dtdd(soup: BeautifulSoup) -> dict:
    out = {}
    dts = soup.find_all("dt")
    if not dts:
        dts = soup.select("dt[class]")
    for dt in dts:
        dd = dt.find_next_sibling("dd")
        if not dd:
            continue
        key = norm_key(dt.get_text(" ", strip=True))
        val = dd.get_text(" ", strip=True).replace("\xa0", " ").strip()
        if key and val:
            out[key] = val

    # Fallback: some pages render in divider rows
    if not out:
        rows = soup.select("div.MuiDivider-root, div[class*='Divider']")
        for row in rows:
            cells = row.select("dt, dd, div")
            if len(cells) >= 2:
                key = norm_key(cells[0].get_text(" ", strip=True))
                val = cells[1].get_text(" ", strip=True).replace("\xa0", " ").strip()
                if key and val:
                    out[key] = val

    return out


def pick_fields(all_kv: dict) -> dict:
    row = {k: "" for k in WANTED.keys()}
    for field, aliases in WANTED.items():
        for alias in aliases:
            if alias in all_kv:
                row[field] = all_kv[alias]
                break

    # price cleanup
    if row["price"]:
        row["price"] = row["price"].replace("/month", "").strip()

    return row
```

5 Results and Conclusion

At the end of the pipeline attributes are being collected and data is being saved which contains in total 4 rows and 3061 columns. Using this data visual presentation is performed for informative purposes of the testing scenario.

Table 1- Final data, source: (Own pipeline result,2026)

price	energy_effici	area	posted
24000	Class A	46 m2	25.12.2025
50000	Class G	109 m2	25.12.2025
33000	Class B	83 m2	25.12.2025
17600	Class G	34 m2	25.12.2025
24950	Class C	83 m2	25.12.2025
20200	Class G	58 m2	25.12.2025
16500	Class G	24 m2	25.12.2025
22500	Class C	54 m2	25.12.2025
21900	Class D	53 m2	25.12.2025
36000	Class A	78 m2	24.12.2025
40000	Class G	99 m2	24.12.2025
23000	Class B	55 m2	23.12.2025
89133	Class B	137 m2	23.12.2025
49443	Class B	160 m2	23.12.2025
28000	Class B	59 m2	22.12.2025
99000	Class G	165 m2	22.12.2025
26000	Class G	77 m2	22.12.2025
49000	Class G	92 m2	22.12.2025
36480	Class G	72 m2	22.12.2025
25000	Class B	55 m2	22.12.2025
14990	Class G	24 m2	22.12.2025
20800	Class G	37 m2	21.12.2025
26000	Class G	80 m2	21.12.2025

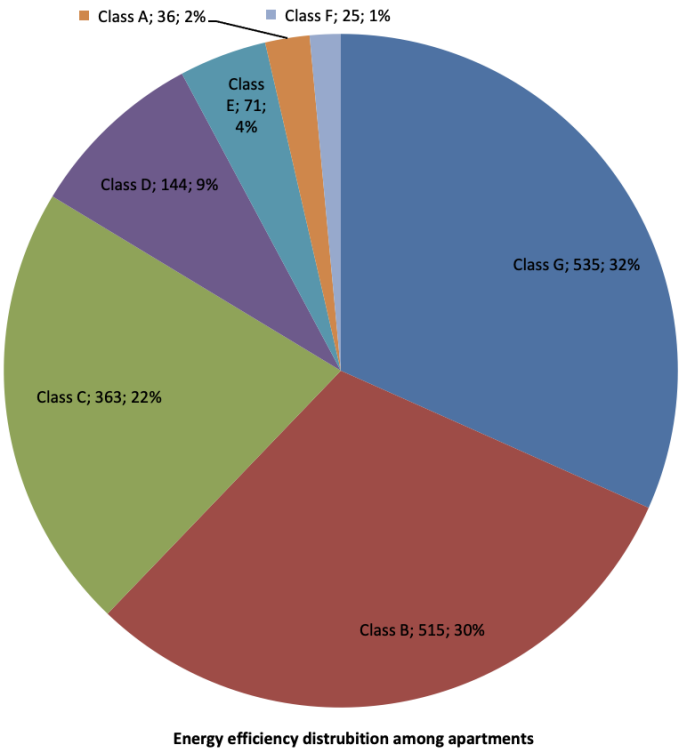


Figure 3 - Pie chart, source :(Own visualization 1, 2026)

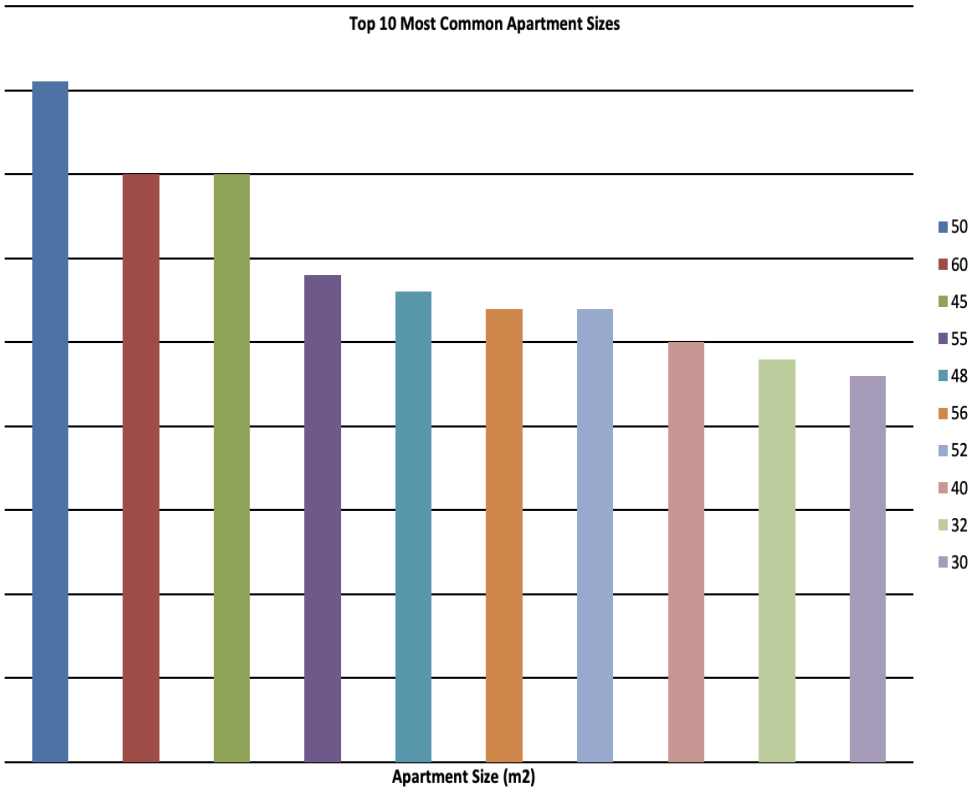


Figure 4 - Visual, source: (Own visualization 2, 2026)

After completing all testing scenario and its analysis, comparative analysis of tools, technologies and methods for each stage of the pipeline the study offers a detailed comparison between those tools, methods and technologies for web scraping, cleaning, preperation and storage of real estate date.

One of the findings is that Beautiful Soup is much faster and more efficient for scraping static content than Selenium. Because Selenium renders the all web page in a browser, whereas Beautiful Soup directly works with the HTML source code. Which allows it to extract the information without processing a full web page.

The web scraping scripts successfully extracted real estate data from Sreality.cz collected necessary information. Moreover, data was stored in Excel and CSV formats, which made it easier for further analysis.

The research compared, programming language options for web scraping, no-low code available tools and libraries which functions for the extraction. As a result of the study I have decided to use requests and Beatiful Soup library which best suited for my test scenario.

As a next step I continiued with data cleaning and preparation where I leveregared Pandas and regex for handling special charaters, hidden charaters, whitespaces, removed duplicates, handled missing values, redefined columns.

Finally, using this result dataset I have created visual to make it even more informative and presentable.

6 References

1. Jeffries, R. The Nature of Software Development, 2015. ISBN 1941222374
2. Kimball, R. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data 1st Edition, 2004. ISBN 8126505540
3. Mitchell, R. Web Scraping with Python, 2018. ISBN 1491985577
4. Reeve, A. Managing Data in Motion: Data Integration Best Practice Techniques and Technologies, 2013. ISBN 0123971675
5. Stephens-Davidowitz, S. Don't Trust Your Gut: Using Data to Get What You Really Want in Life, 2022. ISBN 1526605112
6. James Densmore, 2021, Data Pipelines Pocket Reference. ISBN 1492087831
7. Andreas Kretz, 2018, The Data Engineering Cookbook. ISBN 1835880357
8. Piethein Strengholt, 2020, Data Management at Scale.
9. Jan Kunigk, Ian Buss, Paul Wilkinson, 2022, Architecting the Modern Data Platform. ISBN 149196927X
10. Ramez Elmasri, Shamkant B. Navathe, 2015, Fundamentals of Database Systems. ISBN 0133970779
11. Steve Hoberman, 2009, Data Modeling Made Simple. ISBN 0977140067
12. Len Silverston, 2011, The Data Model Resource Book. ISBN 0470178450
13. Lawrence Corr, Jim Stagnitto, 2011, Agile Data Warehouse Design. ISBN 0956817203
14. Christopher Adamson, Star Schema: The Complete Reference
15. Pramod J. Sadalage, Martin Fowler, 2012, NoSQL Distilled. ISBN 0321826620
16. Wes McKinney, 2022, Python for Data Analysis. ISBN 109810403X
17. Jacqueline Kazil, Katharine Jarmul, Data Wrangling with Python. ISBN 1491948817
18. Hadley Wickham, 2014, Tidy Data
19. Lee Baker, 2019, Practical Data Cleaning. ISBN 1795483458
20. Dorian Pyle,1999, Data Preparation for Data Mining
21. Edward R. Tufte, 1986, The Visual Display of Quantitative Information
22. Cole Nussbaumer Knaflic, 2015, Storytelling with Data
23. Stephen Few, 2006, Information Dashboard Design. ISBN 1938377001
24. Claus O. Wilke,2019, Fundamentals of Data Visualization. ISBN 1492031089
25. Kieran Healy, 2018, Data Visualization: A Practical Introduction. ISBN 0691181616

26. Foster Provost, Tom Fawcett, 2013, Data Science for Business. ISBN 1449361323
27. Cathy O'Neil, Rachel Schutt, 2013, Doing Data Science. ISBN 1449358659
28. David Spiegelhalter, 2019, The Art of Statistics.ISBN 1541618513
29. Peter Bruce, Andrew Bruce, Peter Gedeck, 2020, Practical Statistics for Data Scientists. ISBN 149207294X
30. Andriy Burkov, 2020, Machine Learning Engineering. ISBN 1999579577

7 List of pictures, tables, graphs and abbreviations

7.1 List of Figures

Figure 1 - Webscraping process, source, (https://dev.to/barbaraulowee/how-to-make-money-web-scraping-2p93, 2021) 11
Figure 2 - Stage diagram, source: (Own drawing,2026) 31
Figure 3 - Pie chart, source :(Own visualization 1, 2026) 43
Figure 4 - Visual, source: (Own visualization 2, 2026) 44

7.2 List of Tables

Table 1- Final data, source: (Own pipeline result,2026) 42