# **Real estates is Czech republic**

## **About this project**

My initial idea was to try to scrape any website and clean the acquired data. For this project I chose **sreality.cz**.

I scraped 100 pages and gained basic information about each ad - whether the property is for rent or sale, what kind of property it is, what is the area of the property as well as the land, layout (if the property is a flat), address and price.

## **Methodology**

### Scraping

The problem that I encountered during scraping was that two different values, that I wanted to scrape, had the same class name. I resolved it using *css :nth-of-type* selector.

Here is part of the scraping code *(the rest can be seen here:* [*https://github.com/ZuzanL/data\_projects/blob/main/web\_scraping/scrape\_realestate.py*](https://github.com/ZuzanL/data_projects/blob/main/web_scraping/scrape_realestate.py) *).* In the next step I created a dictionary which I saved as a json file.

| *# Find all properties and their information on the page*  properties = soup.find\_all('div', class\_='css-173t8lh')   *#Break the loop if no more products are found*  **if** **not** properties:  **break**   **for** product **in** properties:  name = product.find('p', class\_='css-d7upve')  **if** name:  p\_name = name.get\_text()  **else**:  p\_name = "N/A"    address = product.select\_one('p.css-d7upve:nth-of-type(2)') *# Selects the second p with class css-d7upve*  **if** address:  p\_address = address.get\_text(strip=**True**)  **else**:  p\_address = "N/A"   price = product.find('p', class\_='css-ca9wwd')  **if** price:  p\_price = price.get\_text()  **else**:  p\_price = "N/A" |
| --- |

### Cleaning

In this picture is shown how my dataframe loaded from the json file initially looked like.



Biggest challenge, during cleaning this dataset, was extracting data from the first column (**‘name’**). My aim was to create new columns:

* Whether the property is on sale or for rent
* What kind of property it is (house, flat, land…)
* What is the area of the property
* What is the layout of the flat (only flats had this information about them)
* What is the area of the land (for every kind of property except the flat)

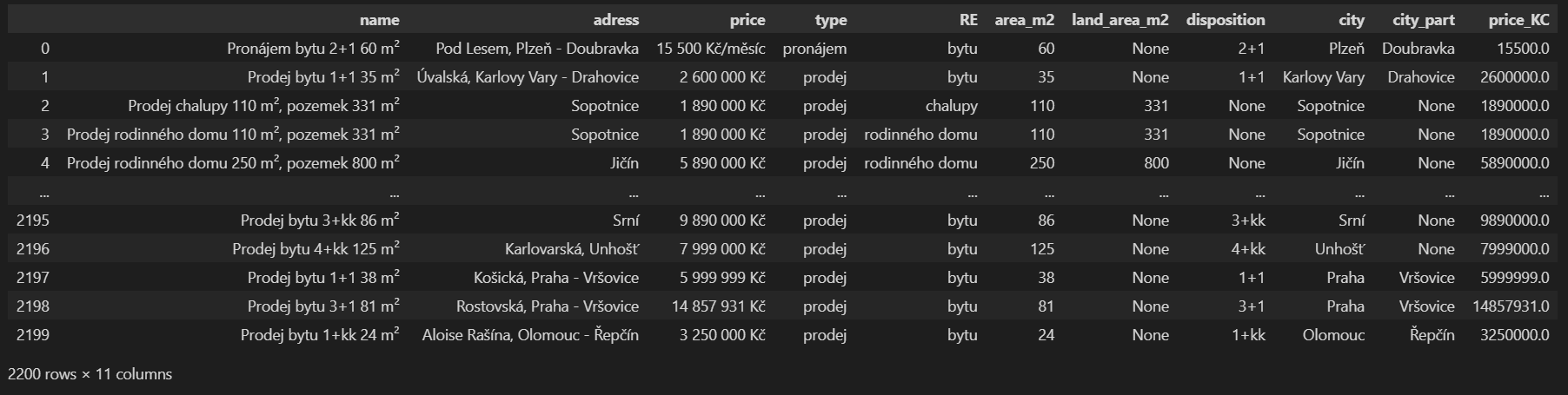
From column **‘address’** I needed to extract the city. Properties in smaller cities or in villages did not have any information about street or city-part, so those were simple (extract just the one word that represented the city). Other properties had information about which street are they on, followed by the city and sometimes also a city part. To include all of the conditions was a bit challenging for me, but this is how I’ve done it:

| **def** **extract\_city**(adress):  adress\_line = adress.split()  **if** len(adress\_line) == 1:  **return** adress, **None** *# Single word: village, no city part*  **else**:  city\_match = re.search(r"([^-,\n]+)(?:\s\*-\s\*([^,\n]+))?$", adress)  **if** city\_match:  city = city\_match.group(1).strip() *#remove spaces*  city\_part = city\_match.group(2).strip() **if** city\_match.group(2) **else** **None** *#remove spaces*  **return** city, city\_part  **else**:  **return** **None**, **None** *# No match, return None for both*  df\_copy[['city', 'city\_part']] = df\_copy['adress'].apply(**lambda** x: pd.Series(extract\_city(x))) |
| --- |

From the last column **‘price’** I extracted just the numbers.

The whole code and all the regular expressions I used can be seen here: <https://github.com/ZuzanL/data_projects/blob/main/web_scraping/cleanup_RE_scraped_data%20%E2%80%93%20k%C3%B3pia.ipynb>

This is the final data frame that I saved as a csv file.



### SQL database

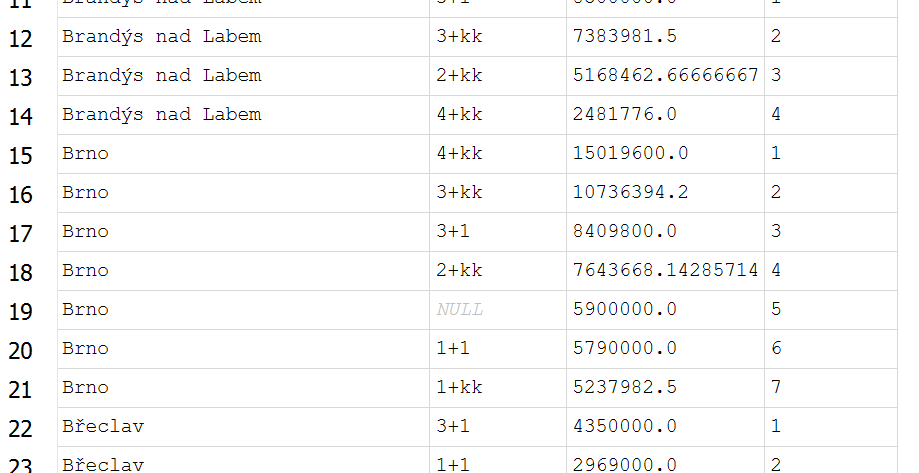
My next step was to create an sql database only from the new columns.

| **import** pandas **as** pd **from** sqlalchemy **import** create\_engine   df = pd.read\_csv('real\_estate\_clean.csv')  *# Selecting columns to include (exclude the first three)* columns\_to\_include = df.columns[3:] df\_to\_sql = df[columns\_to\_include]  engine = create\_engine('sqlite:///real\_estate.db')  *# Writing the DataFrame to the SQL database* df\_to\_sql.to\_sql('czech\_RE', engine, if\_exists='replace', index=**False**)  connection = engine.raw\_connection()  read\_df = pd.read\_sql('SELECT \* FROM czech\_RE', connection)   connection.close()  print(read\_df) |
| --- |

## **SQL queries**

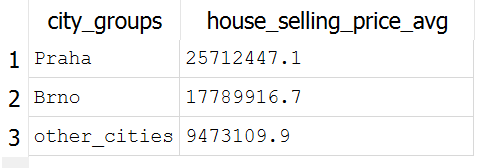
#### **For each city, calculate the average price\_KC and rank the properties within each city based on their price\_KC in descending order. Display the property type, city, price\_KC, average city price, and the rank.**

| WITH city\_avg\_prices as(  **SELECT** city, disposition, **avg**(price\_KC) **as** average\_price  **FROM** czech\_RE  **WHERE** RE = 'bytu' **and** **type** = 'prodej'  **GROUP** **BY** city, disposition ) **SELECT**   \*,  **RANK**() **OVER**(**PARTITION** **BY** city **ORDER** **BY** average\_price **DESC**) **as** rank\_price **FROM** city\_avg\_prices |
| --- |



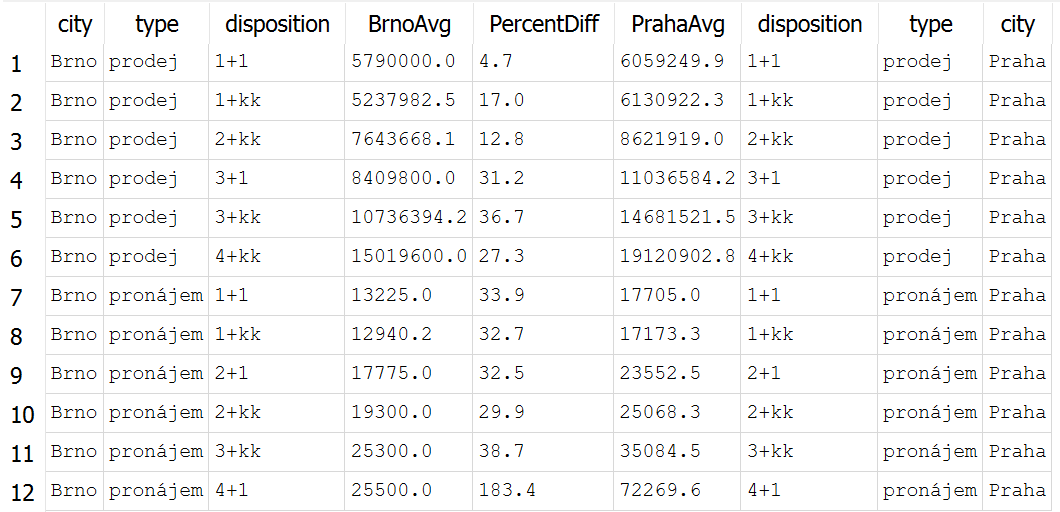
#### **Calculate the average price for houses in Praha, Brno and other cities.**

| **SELECT**   **CASE**   **WHEN** city = 'Praha' **THEN** 'Praha'  **WHEN** city = 'Brno' **THEN** 'Brno'  **ELSE** 'other\_cities'  **END** **AS** 'city\_groups',  **round**(**avg**(price\_KC),1) **as** house\_selling\_price\_avg **FROM** czech\_RE **WHERE** RE = 'rodinného domu' **AND** **type** = 'prodej' **GROUP** **BY**  **CASE**  **WHEN** city = 'Praha' **THEN** 'Praha'  **WHEN** city = 'Brno' **THEN** 'Brno'  **ELSE** 'other\_cities'  **END** **ORDER** **BY** house\_selling\_price\_avg **DESC** |
| --- |



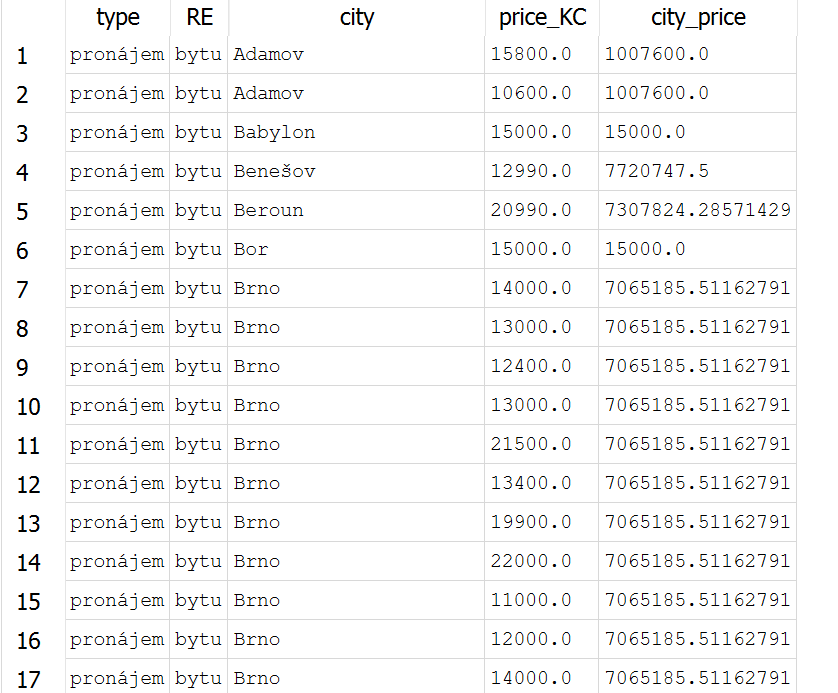
#### **Compare average prices for apartments in Brno and Prague according to their layout and calculate percentage difference**

| WITH brno\_prices AS (  **SELECT** city, **type**, disposition, **round**(**avg**(price\_KC),1) **as** BrnoAvg  **FROM** czech\_RE  **WHERE** city = 'Brno' **and** RE = 'bytu'  **GROUP** **BY** city, **type**, disposition ), praha\_prices **AS** (  **SELECT** city, **type**, disposition, **round**(**avg**(price\_KC),1) **as** PrahaAvg  **FROM** czech\_RE  **WHERE** city = 'Praha' **AND** RE = 'bytu'  **GROUP** **BY** city, **type**, disposition ) **SELECT**   b.city,   b.type,   b.disposition,   BrnoAvg,   **round**((100 / (BrnoAvg/PrahaAvg)),1) - 100 **as** PercentDiff,  PrahaAvg,   p.disposition,   p.type,   p.city **FROM** brno\_prices b **JOIN** praha\_prices p **ON** b.disposition = p.disposition **and** b.type = p.type |
| --- |



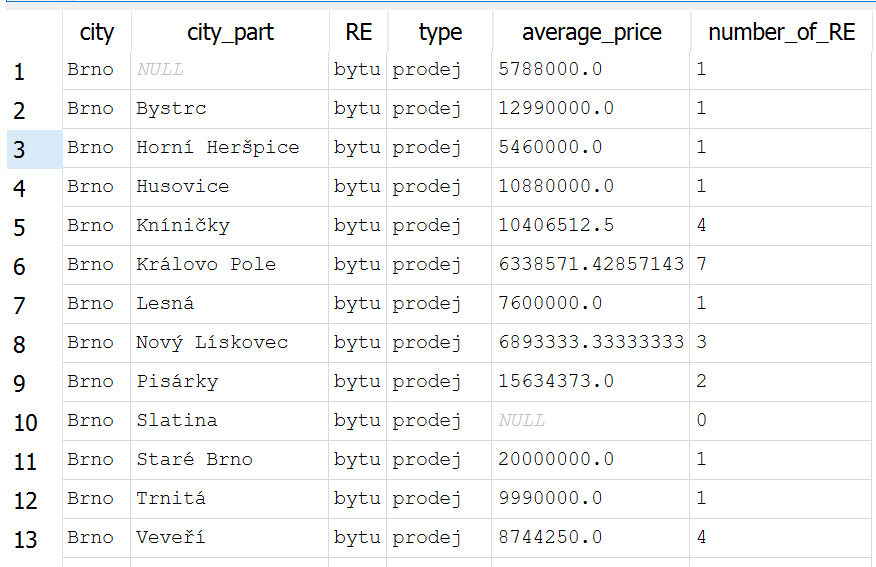
#### **For each city calculate average price and display only rents of apartments**

| WITH avg\_city\_prices AS (  **SELECT** **type**, RE, city, price\_KC,  **avg**(price\_KC) **OVER**(**PARTITION** **BY** city) **AS** city\_price  **FROM** czech\_RE ) **SELECT** \* **FROM** avg\_city\_prices **WHERE** RE = 'bytu' **AND** **type** = 'pronájem' |
| --- |



#### **For Brno and Prague and its city parts show an average price for apartment (for rent or for sale)**

| **SELECT** city, city\_part, RE, **type**, **avg**(price\_KC) **as** average\_price, **count**(price\_KC) **as** number\_of\_RE **FROM** czech\_RE **WHERE** (city = 'Brno' **OR** city = 'Praha') **and** RE = 'bytu'  **GROUP** **BY** city, city\_part, RE, **type**  **ORDER** **BY** **type** |
| --- |



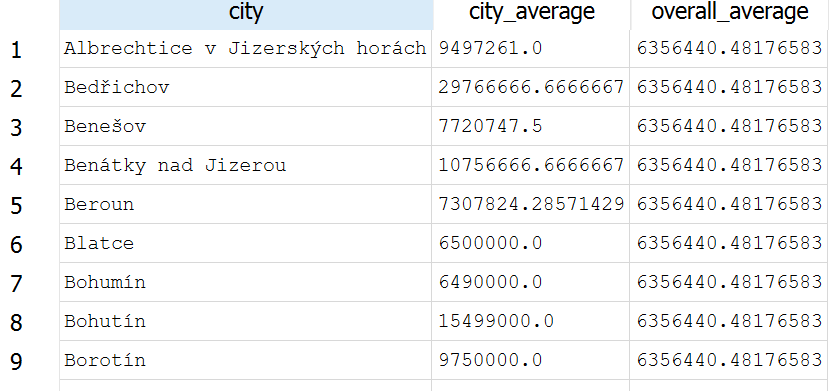
#### **Find the average RE\_area\_m2 for each property type and then list all properties that have an RE\_area\_m2 greater than the average for their type.**

| WITH avg\_RE\_area\_REtypes AS (  **SELECT** **type**, RE, city, area\_m2,  **avg**(area\_m2) **OVER** (**PARTITION** **BY** RE) **AS** avg\_area\_RE   **FROM** czech\_RE ) **SELECT** \* **FROM** avg\_RE\_area\_REtypes **WHERE** area\_m2 > avg\_area\_RE |
| --- |



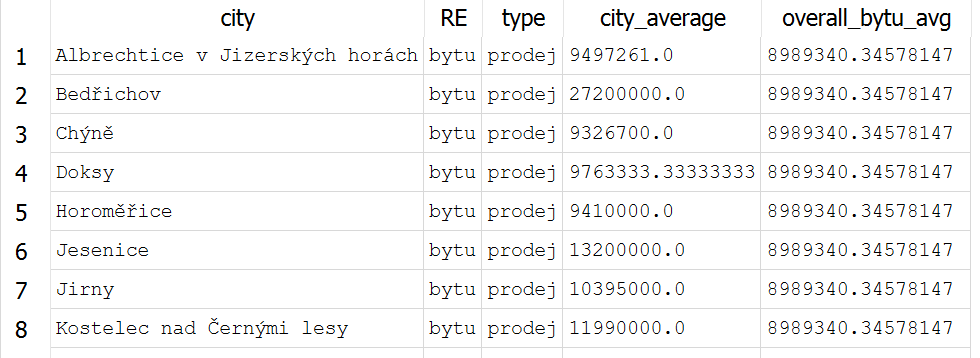
#### **Find the cities where the average price\_KC is greater than the overall average price\_KC for all properties.**

| **SELECT** city, **avg**(price\_KC) **as** city\_average, (**SELECT** **avg**(price\_KC) **FROM** czech\_RE) **as** overall\_average **FROM** czech\_RE **GROUP** **BY** city **HAVING** **avg**(price\_KC) > (**SELECT** **avg**(price\_KC) **FROM** czech\_RE) |
| --- |



#### **Find the cities where the average price\_KC for selling apartments is greater than the overall average price\_KC for all selling apartments.**

| **SELECT**   city,   RE,   **type**,   **avg**(price\_KC) **as** city\_average,  (**SELECT** **avg**(price\_KC) **FROM** czech\_RE **WHERE** **type** = 'prodej' **and** RE = 'bytu') **as** overall\_bytu\_avg **FROM** czech\_RE **WHERE** **type** = 'prodej' **and** RE = 'bytu' **GROUP** **BY** city, RE, **type** **HAVING** **avg**(price\_KC) > (**SELECT** **avg**(price\_KC) **FROM** czech\_RE **WHERE** **type** = 'prodej' **and** RE = 'bytu') |
| --- |



#### **Which Brno city parts have higher average apt. rents than average apt. Rent in Brno?**

| **SELECT**   city\_part,   **avg**(price\_KC) **as** city\_part\_avg,   **round**((**SELECT** **avg**(price\_KC) **FROM** czech\_RE **WHERE** **type** = 'pronájem' **and** RE = 'bytu' **and** city = 'Brno'),2) **as** brno\_avg\_rent **FROM** czech\_RE **WHERE** **type** = 'pronájem' **and** RE = 'bytu' **and** city = 'Brno' **GROUP** **BY** city\_part **HAVING** **avg**(price\_KC) > (  **SELECT** **avg**(price\_KC) **FROM** czech\_RE **WHERE** **type** = 'pronájem' **and** RE = 'bytu' **and** city = 'Brno'  ) |
| --- |

