Department of Informatics

Master Degree in
Data Science

Analysis of Rental Data for Out-of-Town Students in the Metropolitan City of Milan

Iarocci Luca 894066, Mondella Nicholas 859673, Prati Davide 845926

Contents

Roles and Responsibilities

Iarocci Luca:	Public transport data collection, cleaning and integration; report writing (introduction and sections 1 to 3).
Mondella Nicholas:	Rental listings and demographic data collection, cleaning and integration; core database modeling.
Prati Davide:	Data quality and exploration; report overseer and report writing (introduction, sections 4 and 5, and conclusions and further developments).

Common efforts: source and data selection.

Introduction

In the first few months of 2023 one of the main topics of discussion in Italy, especially among off-site university students, has been the cost of renting a house, in particular in the case of big Italian university cities such as Milan, Bologna or Padua.
In response to the increase in rents, students all over the country started organizing protests and demonstrations, asking for a regularization of the students' rent market. The most symbolic act consisted in setting up camp in the near proximity of universities and institutional sites; the first demonstrator leading such protests was Ilaria Lamera, an environmental engineering student from Politecnico di Milano. Her initiative sparked others of the same sort, leading to a heated public debate on the unsustainable hike in housing prices [**1**].

It's no coincidence that the first demonstrator's tent was erected in Milan: the metropolitan city ranks first in Italy for number of university students (around 210 000), 60% of which resides outside its borders [**1**]. Furthermore, the already high rents of the city - compared to national average - experienced a 20% increase over the last two years and the average price of a single room in Milan exceeded an exorbitant 600 €/month.

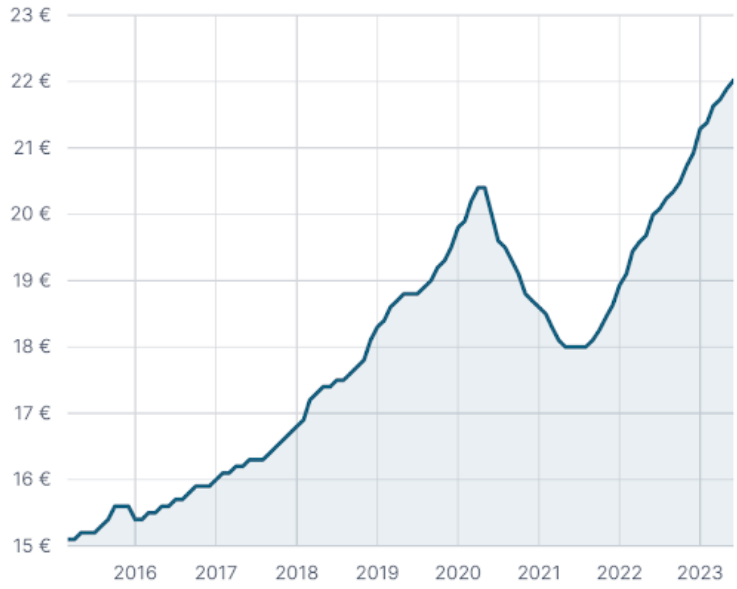

Figure 1: IMMOBILIARE.IT*; rent over time in Milan* [€/m^2]

Objective

The goal of this project is to create a tool that may help off-site students find the best housing solution in Milan, taking into consideration the characteristics of the house, its monthly rent, and the travel time and costs involved in reaching the university from the selected house daily. To do so we decided to take into account just the possibility of renting a house, and not to buy it, which is objectively unrealistic for a 20-year-old student, even more so considering the rising prices. Similarly, it's also relatively unrealistic for said 20-year-old to own a car. That is why only public transport (i.e. train and subway, or a combination of these) was considered when computing travel times. In addition to this, we have decided not to neglect the aspect of entertainment and social life, and also took into consideration a set of plausible interesting locations related to this point and gathering demographic data about the area in which the house is located, such as population density and average age. We are fully aware of the fundamental role that mental health and leisure play in an off-site student's quality of life; and besides, why not benefit from the money saved from the rent to take care of it?

The solution we developed ultimately consists in a database of rent offers of the Milan metropolitan area, collecting data about each specific listing and integrating it with information about the house's public transport connections to the city's main points of interest. Such a repository of data, other than being useful in searching the right house based on a multitude of factors and characteristics, also opens up the possibility to analyze Milan's rent housing market as a whole through statistical and machine learning techniques.

1. Data Acquisition

To obtain information regarding rents in Milan and its neighboring areas, we decided to scrape data from what is by far the leading site for buying and renting properties in Italy, immobiliare.it. As previously stated, our focus lies on the rental property market in the Metropolitan City of Milan (former Province of Milan). An overview is easily accessible at the following link.

Figure 1.1: *Map of the metropolitan city of Milan*

The metropolitan city of Milan is composed by 133 municipalities for a total of around 3.2 million inhabitants.
As noticeable from the map above, the exclave of San Colombano al Lambro is also included; for historical reasons this municipality belongs to the metropolitan city of Milan, even if it is located between the provinces of Lodi and Pavia, more than 20 kilometers from the nearest Milanese municipality. On the map San Colombano appears in a different position compared to its actual location due to cartographic displacement. For obvious logistic reasons regarding distance and travel time to Milan, we will exclude that municipality from our analysis, even if it should be formally counted in our study of the territory.

Regarding public transport, information was gathered from OpenStreetMap and Google Maps using API and automated web scraping techniques respectively.

OpenStreetMap, which unlike Google Maps gives open access to its geographical data, was used to easily collect information about train and subway stations, namely their locations and names. On the other hand, Google Maps was used to compute travel times from each station to the respective destination, benefiting from the fact that public transport schedules are incorporated in its system, a feature currently not supported by OpenStreetMap when it comes to Milan.

Finally, Urbistat was used to obtain demographic data about the municipality where the rented house is located. In particular, to understand the appeal of the environment from a young student's point of view we concentrated on specific metrics, for example the percentage of young people residing in the area.

In the following sections, the selection logic behind the gathered information from each source and the techniques and strategies implemented to acquire it (schematized in figure 1.2) will be presented in detail.

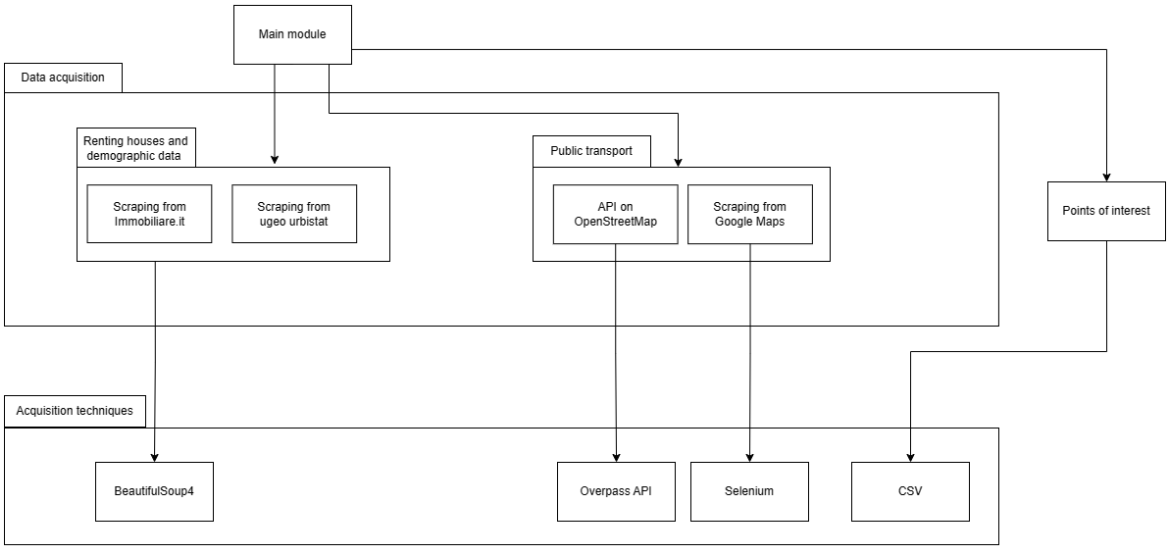

Figure 1.2: *Draw.io schema for data acquisition*

1.1. Renting houses scraping from Immobiliare.it

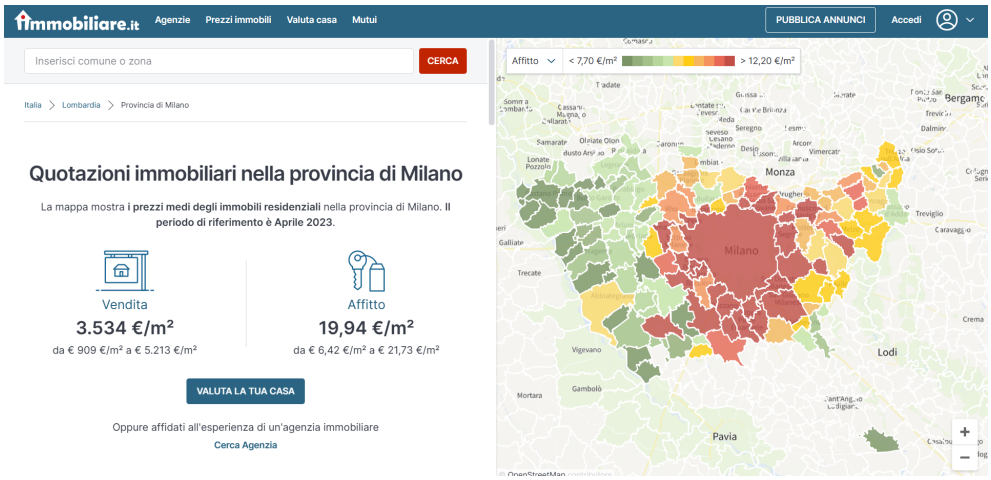

Figure 1.3: *Immobiliare front page for Milan province*

As previously stated, the choice of immobiliare.it as the main source for gathering information about the rental market is motivated by the platform's relevance in Italy. Even if the data completeness of this source can not be precisely quantified due to analogical old-fashioned advertising, it is important to notice that the platform is used both by citizens and a large number of estate agents alike to promote house sales and rents, implying an adequate coverage of the market.
At the time of writing, the number of rental listings on the site regarding the metropolitan mity of Milan amounted to about 7.500, the city of Milan accounting for about 6.600 or 90% of the total alone. Obviously these figures are constantly subject to changes, due to the fact that some houses actually find a tenant and are consequently removed from the site, meanwhile other rental applications are uploaded.
For analysis purposes, considering Milan as a single geographical entity would lead both to a class imbalance problem and to a non-representative picture of the diverse realities of the city's rental market: it is quite obvious that the average rent decreases as one moves away from the center, so it would be a stretch to unify all of Milan's listings under a single tag. Being aware of that, a further segmentation of the territory was taken into consideration. Luckily immobiliare.it gives a more granular division in geographical sub-zones of the city of Milan and other major cities, thus providing a quick to use solution to said problems. For example, as far as Milan is concerned, immobiliare.it presents a division into 30 concentric radial and sectoral areas around the city center (as illustrated in image 1.4 shown below). This hierarchical-like structure, composed of municipalities and sub-zones, has been replicated, when present, in the data collected from the site.

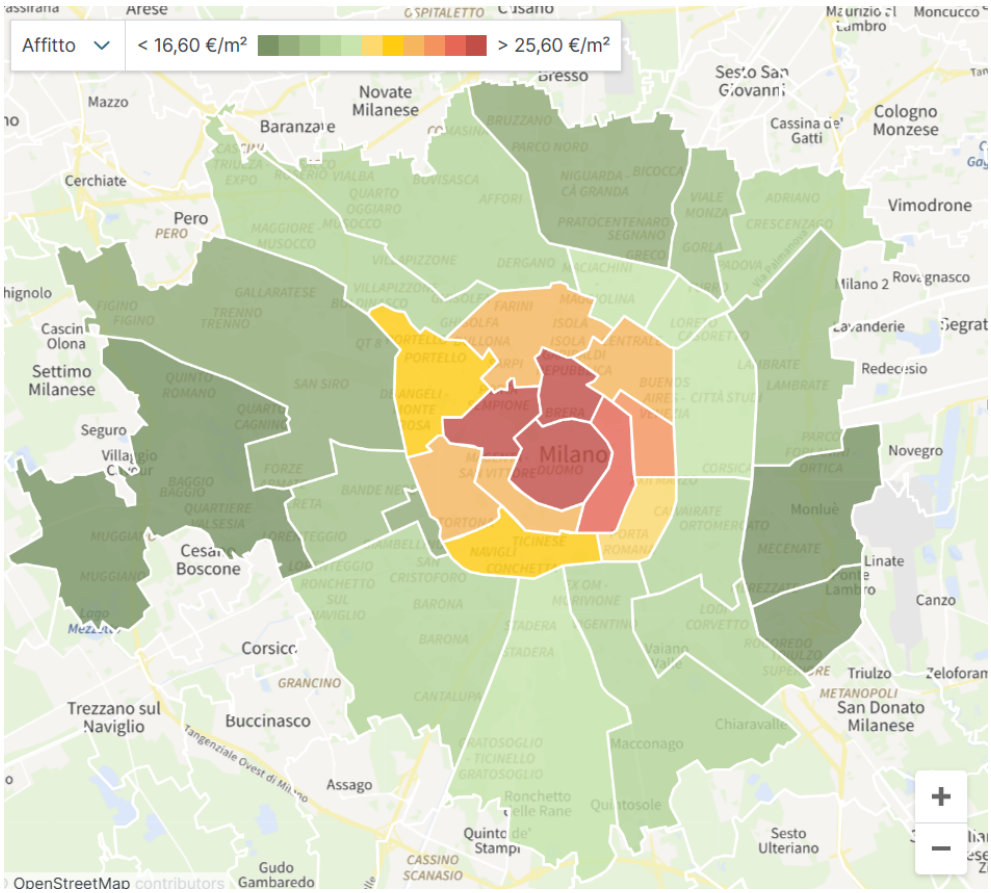

Figure 1.4: *Milan split in sub areas according to* IMMOBILIARE.IT

The data collection process works by exploring the site's listings following an approach similar to that of a depth-first search. This is done by using the site's structure and manipulating the page URL accordingly to obtain the desired information. The root URL of the search algorithm is the previously linked summary page for the metropolitan city of Milan's rental market. The page presents a list of the municipalities composing the territory, hyperlinking each municipality name to the respective summary page. The municipality summary page, other than providing interesting information about the city and the city's real estate market, also directly links to the city's rental listings and, when present, to its sub-zones, each one having its own dedicated listing page. To further illustrate the URL structure, an example of the URL of the summary page is provided in figure 1.5, this page then links to the listing page with a URL like the one shown in the figure 1.6.

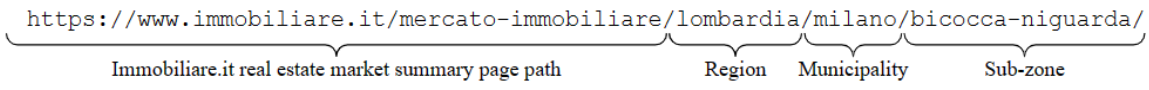

Figure 1.5: *Summary page's URL*

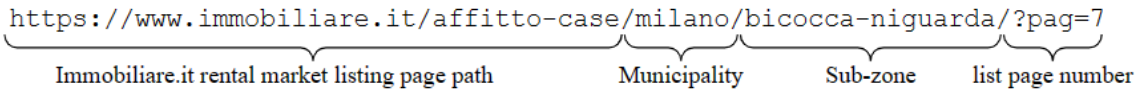

Figure 1.6: *URL to one of the listing pages for research (the 7^{th})*

Data scraping is implemented in Python and uses the BeautifulSoup library [**2**] to navigate the HTML source code of the web pages described and to retrieve the following information:

From the listing page for each advert:

- URL
- name/title
- type (whole property/room, flat/house)
- number of rooms and rooms' description
- surface (m^2)
- floor number
- monthly rent and contract type
- presence of a lift, balcony/ies, and garage/s
- garden surface (m^2)
- loan surface
- if animals or smokers are accepted or not

From the market summary page of each municipality:

- name
- sub-zones
- minimum, maximum, and average rent and sale price

Overall, the listings' information is mostly standardized, with some important details (like surface, number of rooms, rent, type, etc.) being displayed in a tabular-like format while other less important ones are reported as tags under "other features". The information selection criteria comes down to the perceived relevance of said details and their completeness over the multiple listings.
Regarding the information retrievable from the market summary page, some difficulties arose in extracting the basic demographic data already present on the site due to the structure of the web page. The problem was addressed, as anticipated, by using another source altogether (Urbistat) for gathering even more detailed data on the matter.

1.2. Public transport data

To better understand the appeal of a rental offer from an off-site student's point of view, an evaluation of the connections between the property and the city's possible locations of interest is proposed. This analysis enriches the information about each listing and may help make a better choice; for example: a house farther away from the city center but well connected to the desired locations may prove both less expensive and easier to find in the first place.
Data generation and collection related to this topic is the result of a two-step process:

- *step 1*: locating the nearest train or metro station from a given property, while also estimating the distance and time needed to reach it on foot.
- *step 2*: computing the time required to reach a desired destination and its distance from said public transport station.

The set of desired locations was manually generated, saved as a CSV file and designed to be modifiable by future users. For this operation to work, the following information had to be specified for each location: name, address, geographic coordinates, arrival time and date. This project used only a ten-locations set, covering major universities (Politecnico di Milano Campus Leonardo, Politecnico di Milano Campus Bovisa, Università degli Studi di Milano, Università Milano Bicocca) and the city's cultural, shopping and nightlife districts (Navigli, Piazza Duomo, Pinacoteca di Brera, Arco della Pace, Alcatraz nightclub, San Siro Stadium).

Both steps needed a list of metro and train stations of the metropolitan area and their location. OpenStreetMap, an open geographical database, was the selected source for retrieving this data; it is possible to interact with the site and its database through the use of APIs. For the goal of this project we relied on OverpassAPI [**3**], a read-only API optimized for data retrieval. The API's requests were formulated using its own query language (Overpass QL) and ultimately let us retrieve the desired information as JSON files. Milan's number of metro and train stations turned out to be 177.

A distance function from the GeoPy library [**4**] was used to determine the linear distance between the nearest station and a given property (*step 1*). Considering all the possible geographical coordinates' pairs (specific house + different stations), an iterative process compared the distances and selected the nearest one. The same function was used in *step 2* for computing the distance between each station + destination match. The time needed to reach the nearest station on foot was simply estimated based on the average walking speed of $5km/h$, i.e multiplying the distance in kilometers by a constant of 12, to obtain the distance in minutes.
However, this approach implied a certain degree of approximation due to the fact that a straight path is rarely walkable; it nonetheless provided sufficiently precise results during tests, considering the walkability of Milan and its surrounding areas. On the other hand, computing travel times when using public transport needed a more so-

phisticated approach.

To collect the information described in *step 2*, a scraping from Google Maps was implemented. It relied on Selenium, a browser automation tool [**5**]. Despite the site's very complex URL structure, it has a rather relatively simple and straightforward visual interface, thus making a browser automation solution easier. Although Google Maps offers a paid access plan to its data through API, it was ultimately decided not to buy it for the sake of keeping this project no-budget. The solution developed interacts with the page, as illustrated in figure 1.7, starting by selecting the "arrive by" mode (1) and setting the desired date (2) and time of arrival (3). Then, after inserting the starting point (4) and destination (5) addresses, the program gets an estimated travel time via public transport (6) from the HTML source code. This procedure is then repeated for each station + point of interest pair, resulting in a many to many (177 by 10 to be exact) time table describing the public transport connections in terms of travel time.
The joint information obtained in *steps 1* and *2* ultimately let us estimate the total time needed to reach one of our selected destinations starting from a given property.

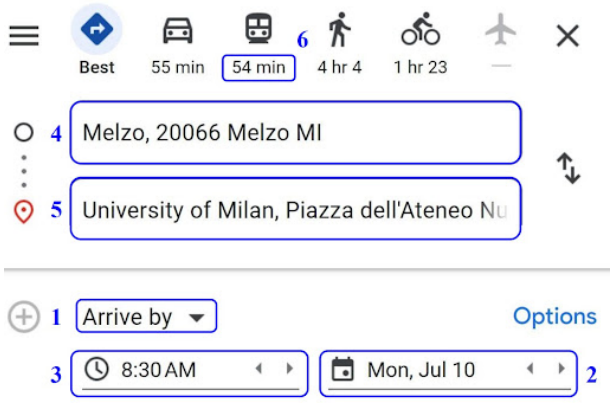

Figure 1.7: *Example of searching for a route from point A to point B on Google Maps.*

1.3. Demographic data

The source of the demographic data chosen for this project is Urbistat. The company collects information primarily from ISTAT, reorganizing, visualizing and giving access to it through the AdminStat portal. The site's structure closely resembles that of immobiliare.it, presenting a summary page for the metropolitan city of Milan, hyperlinking each municipality to its dedicated page, and suggesting a data scraping strategy similar to the one used for immobiliare.it.
In an iterative process implemented in python that uses the BeautifulSoup library, each municipality's demographic information page was accessed through said hyperlinks and the following information was extracted:

- population
- population density
- surface (km^2)
- number of households
- average age
- percentage of males and females
- percentage of foreigners

Unfortunately the site does not provide demographic data for the municipality's sub-zones.
Moreover, to further complicate the data collection process, Milan's sub-zones provided by immobilare.it follow a market logic and don't match Milan's census sections (called NIL), whose demographic data is actually available through the city's open data portal **[6]**. In conclusion, a satisfactory solution to the lack of sub-zones' demographic information was not found, and the goal of a more granular demographic data integration of the dataset remains open to further developments.

2. Data Storage

Data modeling, and consequently the design and choice of a proper database model, was the next step in pursuing the goal of this project. Said data storage solution should ideally both efficiently represent the multiple relations between the gathered information from multiple sources, and support a simple but powerful query language. When observing the collected data, it is clear that it mostly follows a rigid schema, where each source's records present a relatively well defined structure and share a large set of somewhat standardized features. For example, one is likely to expect features like price, location, surface and rooms for every serious listing and they are presented by the site itself in a tabular fashion; in general, only the "other feature" tags change from ad to ad, usually being of less importance anyway. The coherence to a rigid schema is even more noticeable in the case of demographic and public transport station data, the latter being provided by Overpass API in JSON format but, for what concerns the selected information, easily convertible to tabular format, given that name and location are a logical starting point for a station to be saved in a geographical database. In conclusion, when it comes to the collected data structure from single sources, its mostly rigid schema and the fact that the schema itself is known in advance suggests the use of a relational database, neither needing nor benefitting from the schema free approach of No-SQL solutions.

Another point to consider when making the right database architecture choice is the relation between data. It's easy to interpret the connection between the differently-collected information as a series of relations of the many-to-one type (ex. the nearest station and municipality associated to each property) and many-to-many type (ex. a travel time is associated to each station + point of interest pair), further supporting the feasibility and logic of a relational database solution. Although the evaluation of spatial relationships between objects may suggest an alternative graph database solution, it is important to note that the spatial analysis performed consisted merely in measuring the distance (both in terms of time and space) between pairs of objects and computing the total distance of a-three-step journey (property-station-point of interest) at best. Thus, is reasonable to conclude that said analysis neither requires the relations' modeling power of a graph database nor would benefit from its superior capabilities when it comes to handling and computing paths; in fact, the complex and demanding computational task of finding the shortest path between a property and the desired destination is performed almost entirely by Google Maps.
Finally, the data volume and the scope of the project do not require a distributed approach to data storing, hence a No-SQL solution would not benefit from one of its main strengths.
In conclusion, in light of the above observations, it was decided to opt for a relational database. The proposed solution is based on SQLite and was developed using the database tool DBeaver [**7**].

2.1. Database modeling

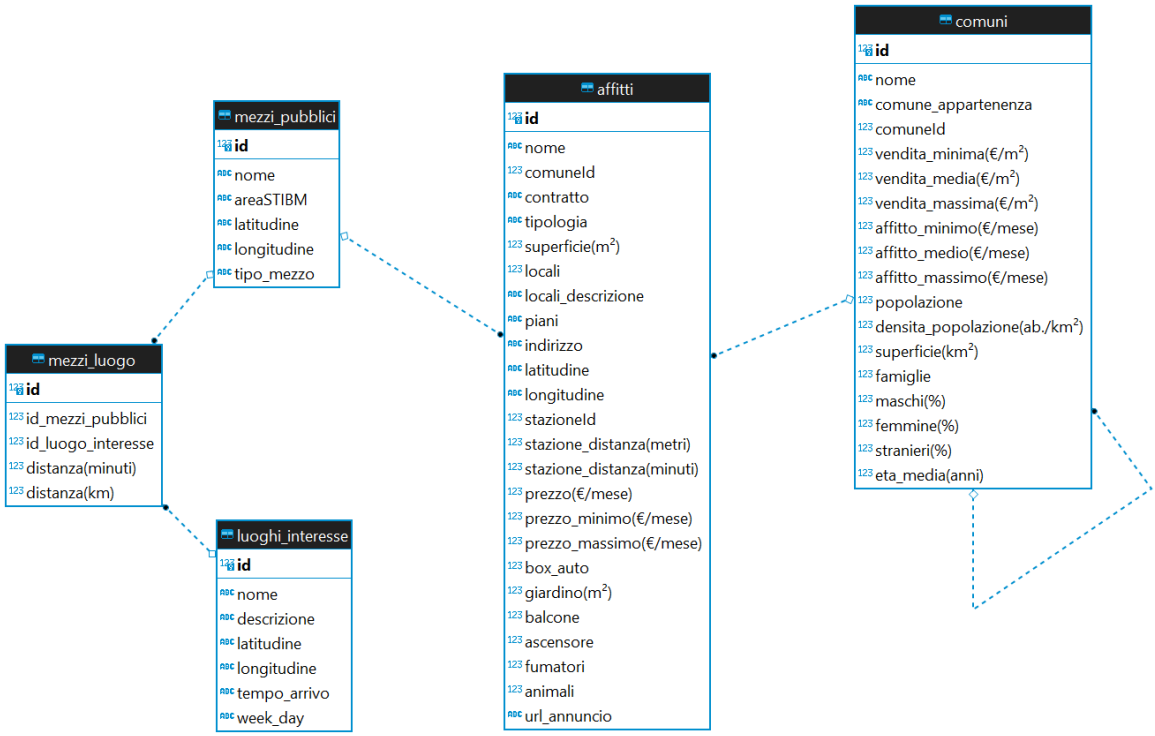

Figure 2.1: *Entity relationship schema*

The database structure, illustrated in figure 2.1, comprises the following five tables and their respective features:

- COMUNI is a 17-feature-table containing the following information about the 228 zones provided by immobiliare.it, comprising municipalities and subzones for Milan's metropolitan area. This table puts together the information related to the rental market summary page and the demographic of each zone.
  - *nome*: name
  - *comune_appartenenza*: municipality's name to which a zone belongs
  - *comuneId*: key linking to the table itself, specifying the zone's municipality id
  - *vendita_minima(€/m²)*: minimum selling price
  - *vendita_media(€/m²)*: average selling price
  - *vendita_massima(€/m²)*: maximum selling price
  - *affitto_minimo(€/mese)*: minimum monthly rent

  - *affitto_medio(€/mese)*: average monthly rent
  - *affitto_massimo(€/mese)*: maximum monthly rent
  - *popolazione*: population
  - *densita_popolazione(ab./km^2)*: population density, people per km^2
  - *superficie(km^2)*: surface in km^2
  - *famiglie*: number of households
  - *maschi(%)*: percentage of men in a given zone
  - *femmine(%)*: percentage of women in a given zone
  - *stranieri(%)*: percentage of foreigners in a given zone
  - *eta_media(anni)*: average age of the population

- AFFITTI is a 24-feature-table containing the following listings' information about 7500 properties:
  - *nome*: name/title
  - *comuneId*: foreign key linking to comuni
  - *contratto*: contract type
  - *tipologia*: property type (whole property/room, flat/house)
  - *superficie(m^2)*: surface in m^2
  - *locali*: number of rooms
  - *locali_descrizione*: rooms' description
  - *piani*: floor number
  - *indirizzo*: address
  - *latitudine*: latitude
  - *longitudine*: longitude
  - *stazioneId*: foreign key linking to mezzi_pubblici
  - *stazione_distanza(metri)*: distance from the nearest station
  - *stazione_distanza(minuti)*: time needed to reach the nearest station
  - *prezzo(€/mese)*: monthly rent
  - *prezzo_minimo(€/mese)*: minimum monthly rent over time
  - *prezzo_massimo(€/mese)*: maximum monthly rent over time

  - *box_auto*: garage, it indicates the amount
  - *giardino(m^2)*: garden surface in m^2
  - *balcone*: balcony (assumes the value 1 if present, 0 otherwise)
  - *ascensore*: lift (assumes the value 1 if present, 0 otherwise)
  - *fumatori*: smokers accepted (assumes the value 1 if yes, 0 if no)
  - *animali*: animals accepted (assumes the value 1 if yes, 0 if no)
  - *url_annuncio*: listing's URL

- MEZZI_PUBBLICI is a 5-feature-table containing the following information about the 177 public transport stations of Milan's metropolitan area:
  - *nome*: name
  - *areaSTIBM*: fare zone according to Milan's integrated fare zone system (STIBM)
  - *latitudine*: latitude
  - *longitudine*: longitude
  - *tipo_mezzo*: type of transportation, either subway or train

- LUOGHI_INTERESSE is a 6-feature-table containing the following information about the 10 city location taken into consideration:
  - *nome*: specific name and address
  - *descrizione*: common name
  - *latitudine*: latitude
  - *longitudine*: longitude
  - *tempo_arrivo*: arrival time desired
  - *week_day*: arrival date desired, either monday or saturday (TRUE if monday, FALSE if saturday)

- MEZZI_LUOGO is a 4-feature-table containing the following information about public transport connections for all 1770 stations + city's locations pairs:
  - *id_mezzi_pubblici*: foreign key linking to mezzi_pubblici
  - *id_luogo_interesse*: foreign key linking to luoghi_interesse
  - *distanza(minuti)*: time needed in minutes
  - *distanza(km)*: distance in *km*

The tables described are linked to one another as follows:

- AFFITTI links to COMUNI and MEZZI_PUBBLICI through many-to-one relationships
- MEZZI_LUOGO is a bridging table describing a many-to-many relationship between MEZZI_PUBBLICI and LUOGHI_INTERESSE
- COMUNI presents a self-linking relationship connecting subzones to the municipality they belong to

3. Data ETL, Transformation, Cleaning and Integration

3.1. Data ETL

The multiple data sources used in this project not only implied the need of different data extraction techniques but also of distinct approaches when handling the data ETL process of each source as a whole. Said process was implemented mostly in two different ways, either by following a more "horizontal" logic or a rather "vertical" one. The former called for a more traditional approach, according to which all the source's data was extracted (and consequently saved in a temporary file or format), transformed and loaded as a whole, whereas the latter used a record-based approach where each single record was singularly extracted, transformed and loaded into a preexisting table.
The "vertical" approach was used when dealing with immobiliare.it and Urbistat; in those cases each listing's (municipality's) data was progressively subjected to the whole ETL process one record at a time, ultimately composing the AFFITTI and COMUNI tables. The "horizontal" approach was instead preferred when dealing with OpenStreetMap. This was mostly due to the volume and way the API provides data (only two JSON files, one for train stations and the other one for subways). Lastly, an hybrid batch-based approach was used when generating and gathering data about public transport connections, computing and subjecting the stations + locations pairs' data to the ETL process one location at the time, resulting in a 10-batch process.

3.2. Data Transformation and Cleaning

The extracted data, due to it inherently following a rigid schema (hence its mostly being standardized), resulted in the need for a relatively limited cleaning process. The following list summarizes the procedures implemented prior to the loading phase, organized by source:

- immobiliare.it
  - on listings: missing numerical data were replaced with zeros. Similarly, the same thing was done with missing tags, coherently to their being modeled as boolean. Unspecified rent prices labeled as 'prezzo su richiesta' (rent on demand) were assigned a flag value of zero. The points separating three-figure groups were removed from rent prices, originally extracted as strings and then casted to integers.
  - on municipalities: missing numerical data were replaced with zeros.

- Urbistat

  missing numerical data were replaced with zeros.

- OpenStreetMaps

  the data collected in the form of two JSON files was subjected to a basic feature selection process removing all excess information. Stations with missing geographical coordinates were excluded.

- Google Maps

  travel times, extracted as an "h ora mm min"-type string, were converted to minutes and casted to integers.

Finally, after data loading, a deduplication step for AFFITTI was implemented directly on the database by querying the saved data. This step was needed due to the fact that a few properties are located in close proximity to zone borders and/or don't have a precise enough location, hence their possible presence in more than one list of advertisements. The identification of said duplicates was based on a url match criterion. Lacking a meaningful criteria for assigning the correct zone to these properties, a "keep first" approach was used, that is to say the first instance of a duplicate was kept and the others removed from the dataset.

3.3. Data Integration

From a logical modeling point of view, this data integration process can be mostly interpreted as an enrichment of the listings, these being the starting point to which to add information.
Starting from municipalities and zones, each listing is linked to a geographic entity – usually shared by more than one listing – by site's design. The integration process worked by linking the respective zone to each property using a zone identifier. Zones' information was collected and stored in a dedicated table where data about demographics and real estate markets was merged by a url match criterion. The merging followed an outer join logic, thus preserving all the information gathered – also the one of zones having no listings.
On the other hand, public transport data integration was the result of either an elaboration of the collected data (ex. the locating of the nearest station) or a data generation-like process (ex. the computing of travel times using Google Maps). The first type of integration mentioned above linked the nearest station to every listing via a unique identifier; this was done, as previously explained, using the extracted coordinates of listings and stations for computing and comparing the linear distance of every pair. Public transport stations data was stored, according to the minimization principle, in a separate table and referenced to in a many-to-one relationship by the listings information table AFFITTI. The second type of integration – related to public connections and travel times – was the result of a systematic querying of Google Maps through browser automation means. Every station + location pair's data was

saved in a bridging table referencing both stations and locations throughout unique identifiers. The resulting table comprised all possible combinations of starting points + destinations and their respective travel times, even though not necessarily all pairs were used due to the lack of houses in some areas.

4. Data Quality

Regarding the measurement of the data quality, we deem it appropriate to consider and evaluate four different dimensions: accuracy, completeness, currency and timeliness. We will devote a specific section of this chapter to each of them.
Clearly, the data quality will be evaluated only on the COMUNI, AFFITTI, and MEZZI_PUBBLICI datasets. Considering the other two, LUOGHI_INTERESSE was obtained manually, while the MEZZI_LUOGO dataset is the result of the join between MEZZI_PUBBLICI and LUOGHI_INTERESSE, so it does not require further analysis.

4.1. Completeness

Completeness of a dataset is simply the coverage with which the observed phenomenon is represented in the dataset.
There are various measures of completeness, among which we will consider two types: attribute completeness, which refers to the presence of null values in the column corresponding to an attribute, measurable as the ratio of the number of null values to the total column size; and table completeness, which is measured as the ratio of the total number of null values within the dataset and, as denominator, the size of the table.
To evaluate completeness, we will use the pandas profiling library [**8**] in Python, which allows us, among other things, to obtain a detailed report on the characteristics of the reference dataset, including the number and percentage of null values within each attribute.

As first dataset, we study the COMUNI one. The results on the number of elements with a value set equal to zero, as obtained from the pandas profiling report, are as follows:

Figure 4.1: *Amount and percentage of zero entries for each attribute that exhibit this phenomenon in the* COMUNI *table*

As can be easily noticed, the columns of this analysis can be naturally divided into two distinct groups: one composed of the attributes with 63 null values and the other composed of those with 95 null values.
Studying the dataset, we observe that the columns in the second group - *popolazione, densita_popolazione, superficie, famiglie, maschi (%), femmine (%), stranieri (%), eta_media* - have zero values in correspondence of all tuples representing neighborhoods and zones, rather than actual municipalities. On the other hand, the columns in the first group - *vendita_minima, vendita_massima, affitto_minimo, affitto_massimo* - have zero values only for neighborhoods and zones of municipalities other than Milan. The reason for this is that immobiliare.it does not provide minimum and maximum sales and rentals data for subzones of municipalities other than Milan, while it still presents average data. On the other hand, regarding the demographic columns, as they were obtained from Urbistat, they simply represent the 133 municipalities in the metropolitan area without any division into zones or neighborhoods. Furthermore, these entries have a value of zero because during the data cleaning phase, among other things, we also set everything that was represented in NaN format as equal to zero.
So, as for attribute completeness, we have, exactly as indicated in figure 4.1, a value of 27.6% for the attributes of maximum and minimum sales and rentals, and a much higher value of 41.7% for all the demographic columns obtained from Urbistat. To calculate table completeness, considering our dataset has 18 attributes for 228 tuples, we perform the following calculation:

$$\text{Table completeness} = \frac{63 \times 4 + 95 \times 9}{228 \times 18} = 27\%,$$

where in the numerator, we have the number of elements set to zero: 63 elements for 4 columns and 95 elements for the other 9 demographic columns; while in the denominator, we simply have the size of the dataset.

Shifting our focus to the AFFITTI table, in this case the pandas profiling report provides us with the following results:

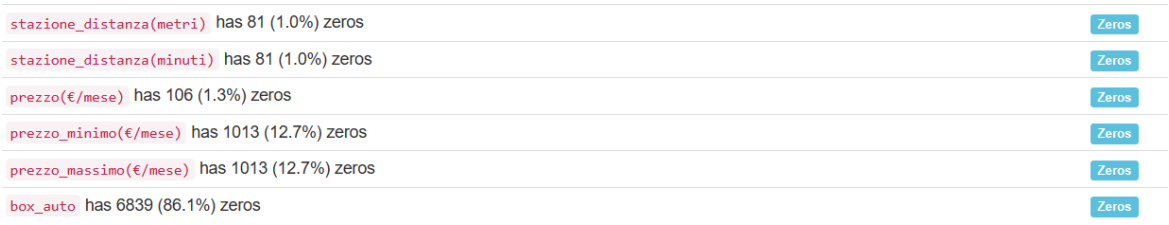

Figure 4.2: *Amount and percentage of zero entries for each attribute that exhibit this phenomenon in the* AFFITTI *table*

Analyzing these attributes one by one, we immediately notice that regarding both the *box_auto* column and the first two columns, related to the distance of the apartment from the nearest station in meters and minutes, the presence of values equal to zero is

not due to dataset incompleteness. In fact, *box_ auto* is set to zero whenever there is no car parking space available, and this is a situation that actually applies analogously to other columns as well, such as *balcony* or *smokers*. As for the distance in meters, on the other hand, it presents a value of zero when the distance of the apartment to the nearest station falls within the range $(0m, 90m)$, considering that the function we defined to calculate it approximates downwards with a sensitivity of 0.1 km. Furthermore, since the distance in minutes is calculated as the product of the distance in kilometers and the constant 12, which assumes that 1 km is covered in an average time of 12 minutes, clearly by multiplying the value of 0 by a constant it will still result in 0.
Therefore, the columns that actually affect the completeness of the AFFITTI dataset are the remaining 3. The tuples that have a value of zero for both minimum and maximum price are the ones that did not provide a historical record on immobiliare.it. On the other hand, the tuples that have the current price equal to zero are the ones that do not contain numerical values for the sales price but rather strings such as 'upon request'. These kinds of string were all converted to zero during the cleaning phase, as already mentioned above.
Thus, in this case, the attribute completeness is equal to 1.3% for *prezzo* column and to 12.7% for *prezzo_ minimo* and *prezzo_ massimo*, while the other two columns indicated in the image actually have full completeness, just like all the other attributes in the dataset.
In addition, from the pandas profiling report, it is also noted that there are three missing values for both the latitude and longitude attributes, referring to the same records. As for these two fields, the attribute completeness is clearly the same and relatively irrelevant, as it is equal to $3/7947 = 0.04\%$.
On the other hand, as for table completeness, similarly to before, we calculate it using the formula

$$\text{Table completeness} = \frac{106 + 1013 \times 2 + 3 \times 2}{7947 \times 25} = 1.08\%,$$

where 7947 and 25 are the dataset dimensions, and 106, 1013 ($\times 2$) and 3 ($\times 2$) are the number of zero values.

Finally, we focus on the MEZZI_PUBBLICI dataset. In this case, the report returns a single result:

Figure 4.3: *Amount and percentage of null entries for each attribute that exhibit this phenomenon in the* MEZZI_PUBBLICI *table*

thus the attribute *areaSTIBM* is the only one that presents missing data, which evidently are not retrieved from the API in those cases. Additionally, analyzing the dataset and specifically the *areaSTIBM* attribute, as seen in the overview from pandas profiling, it is noticed that there is a string value associated with the Repetti metro stop, which clearly does not align with the division into STIBM areas. However, we

will address this issue during the accuracy evaluation.
Anyway, in this case, the completeness of the *areaStibm* attribute is equal to 7.3%, as indicated, while for calculating the table completeness, we use the same formula as before:

$$\text{Table completeness} = \frac{13}{177 \times 6} = 1.2\%,$$

where 177 and 6 are respectively the number of rows and columns of the dataset.

4.2. Accuracy

In a data quality context, the accuracy of a value is defined as the closeness between the given value and another value, which is nothing more than the actual correct representation of the phenomenon that the first value aims to represent.
However, since very often, and also in our case, it happens that we are not able to know what this actual correct representation mentioned above is, or at least it would be very costly to do so, the accuracy as presented with this definition is referred to as semantic accuracy, while another dimension is introduced and will be the one that we will actually evaluate as well.
This other type of accuracy is called syntactic accuracy. It can be considered when we know the reference domain of our data, and it can be defined as the degree to which the values we have actually belong to this reference domain.

But even when evaluating syntactic accuracy, we must consider that the accuracy of our datasets has been significantly improved during the data cleaning phase. As discussed in Chapter 3, values such as price in the AFFITTI dataset that would have originally contained phrases like 'price upon request' were set to 0.
It can be said that in the tradeoff between accuracy and completeness, which was discussed in the previous section, we have chosen to prioritize accuracy at the expense of completeness.
In any case, regarding many numerical data, we can verify that they belong to the reference domain, meaning they essentially take positive values. We are referring to data such as price and surface area of an apartment, or demographic data. Even in this case, we obtain the results from the pandas profiling report, and we do not encounter any issues.

A different discussion can be had regarding the latitude and longitude values in both the AFFITTI and MEZZI_PUBBLICI datasets. Being within the metropolitan city of Milan, their reference domain will not deviate much from the coordinates of the center of Milan, which, assuming Piazza Duomo as the reference point, has a position given by (45.46; 9.19), as obtained from the LUOGHI_INTERESSE dataset.
By analyzing the maximum and minimum values of latitudes and longitudes in the dataset, we find that they all fall within an acceptable range around the coordinates chosen as the center of Milan, except for a single isolated case, which coordinates are (42.76290, 11.11280) and actually refer to a location in the province of Grosseto, in

Tuscany. Clearly, it does not belong to the reference domain and is a case where syntactic accuracy is not met. However, upon checking the corresponding record, it is indeed referring to an apartment in Milan. Therefore, it is only the coordinate data that is incorrect.

Another singular case is the one involving Repetti metro stop in MEZZI_PUBBLICI dataset, which we have already mentioned in the course of the previous section. In fact, at row 96 of the MEZZI_PUBBLICI dataset, the Repetti stop does not have an actual area or range of areas as the value of the *areaSTIBM* column, as is the case for other stations, but presents a long string that, in summary, explains how this stop has not yet been inaugurated and how it will have to belong to the new M4 line [**9**]. So clearly this is not an inaccurate figure per se, but one that clearly needs to be updated. We will continue this discussion then in the next section, when we will deal with the temporal dimensions of data quality.

4.3. Currency and Timeliness

Currency measures how quickly the data is updated, compared to the corresponding phenomenon in the real world. Timeliness, on the other hand, is a measure more closely related to the degree of data obsolescence and is based on the fact that a data point only holds value if it is fully usable at a given moment for a specific process.
For example, in our case, searching for a specific apartment or finding its current rental price should be done with the most up-to-date data available. This is because the apartment might have been rented out in the meantime or its price might have changed. Similarly, an analysis on the least expensive apartments in a certain area could yield incorrect results due, in addition to the reasons seen above, also to the lack of updates with newly listed apartments that were not previously included in the dataset.
In short, it is the CRUD operations (Create, Retrieve, Update, Delete) that we need to consider, and we cannot neglect them. Thus the data from immobiliare.it should be collected at least on a daily basis to assume at least a partial adaptation to these operations.

Finally, as for Repetti's case again, we can conclude that we are indeed faced with a problem of non-currency, but not on our part, as on the part of OpenSteetMap, from which we obtained the information. in support of this, again referring to [**9**], the other stations on the line are divided between those that are not indicated at all (such as California), or are indicated but without the corresponding STIBM area being reported (such as the case of Argonne). Presumably, by updating the data on perhaps a weekly basis, sooner or later they will turn out to be up-to-date and complete.

5. Data Exploration

At this point, we have actually obtained a database capable of satisfying numerous queries, which constitute the actual doubts and typical objectives of those who approach a search with the intention of finding the best housing in a determinate area for their specific needs.
In this section, we have gathered some of these potential requests, which in our opinion can be representative of the average user and can also fully demonstrate the potential of our dataset. For each query, we will initially state the purpose, then present the query text, and finally its results. In the case of result tables with a large number of rows, we will only show the first 10, while in some cases we will abbreviate column names for space-saving purposes.

1. Goal: to find, for each municipality, the number of listings, sorted in descending order.

```
SELECT    c.comune_appartenenza AS comune,
          COUNT(*) AS numero_annunci
FROM      affitti a
          JOIN comuni c
               ON c.id = a.comuneId
          JOIN mezzi_pubblici mp
               ON mp.id = a.stazioneId
WHERE     a."stazione_distanza(metri)" < 800
GROUP BY  c.comuneId
ORDER BY  numero_annunci DESC;
```

comune	numero_annunci
Milano	5093
Sesto San Giovanni	49
Bollate	22
San Donato Milanese	20
Rho	15
Cernusco sul Naviglio	13
Cologno Monzese	11
Abbiategrasso	10
Vimodrone	9
San Giuliano Milanese	9

2. Goal: to find apartments that are less than a quarter-hour away from Bicocca University by metro.

```
SELECT    a.nome,
          a.indirizzo,
          a."prezzo(€/m^2)",
          a.url_annuncio
FROM      affitti a
          JOIN mezzi_pubblici mp
               ON a.stazioneId=mp.id
          JOIN mezzi_luogo ml
               ON mp.id=ml.id_mezzi_pubblici
          JOIN luoghi_interesse li
               ON ml.id_luogo_interesse=li.id
WHERE     li.descrizione="Bicocca"
          AND (a."stazione_distanza(minuti)" + ml."distanza(minuti)")
          <15
          AND mp.tipo_mezzo="metro"
          AND a."prezzo(€/mese)" > 0
ORDER BY  a."prezzo(€/m^2)" ASC;
```

nome	indirizzo	prezzo	url_annuncio
Bilocale via Gorizia 51, Sesto Marelli, Sesto San Giovanni	via Gorizia 51, Sesto Marelli, Sesto San Giovanni	630	https://www.immobiliare.it/annunci/104422077/
Bilocale via Oslavia 18, Sesto Marelli, Sesto San Giovanni	via Oslavia 18, Sesto Marelli, Sesto San Giovanni	700	https://www.immobiliare.it/annunci/104058587/
Bilocale via Sagrado 15, Sesto Marelli, Sesto San Giovanni	via Sagrado 15, Sesto Marelli, Sesto San Giovanni	700	https://www.immobiliare.it/annunci/104230711/
Monolocale via Esiodo 12/14, Precotto, Milano	via Esiodo 12/14, Precotto, Milano	700	https://www.immobiliare.it/annunci/103236734/
Bilocale viale Giovanni Suzzani, Bicocca, Milano	viale Giovanni Suzzani, Bicocca, Milano	700	https://www.immobiliare.it/annunci/104445943/
Bilocale via Luigi Pulci, Bicocca, Milano	via Luigi Pulci, Bicocca, Milano	750	https://www.immobiliare.it/annunci/104054543/
Bilocale via Gorizia 15, Sesto Marelli, Sesto San Giovanni	via Gorizia 15, Sesto Marelli, Sesto San Giovanni	770	https://www.immobiliare.it/annunci/103753800/

Monolocale via BUOZZI 97, Centro, Sesto San Giovanni	via BUOZZI 97, Centro, Sesto San Giovanni	800	https://www.immobiliare.it/annunci/104330173/
Bilocale via Oslavia 74, Sesto Marelli, Sesto San Giovanni	via Oslavia 74, Sesto Marelli, Sesto San Giovanni	830	https://www.immobiliare.it/annunci/104098817/
Bilocale viale Rodi 91, Bicocca, Milano	viale Rodi 91, Bicocca, Milano	900	https://www.immobiliare.it/annunci/104531715/

3. Goal: to find apartments that are less than a quarter-hour away from Bicocca University, and less than half a hour away from both Duomo and San Siro, all three via metro, to potentially perform map triangulation to visualize the reference area.

```
SELECT      a.id,
            a."prezzo(€/mese)",
            (m1."distanza(minuti)" + a."stazione_distanza(minuti)")
            AS minuti_arrivo_duomo,
            (m2."distanza(minuti)" + a."stazione_distanza(minuti)")
            AS minuti_arrivo_sansiro,
            (m3."distanza(minuti)" + a."stazione_distanza(minuti)")
            AS minuti_arrivo_bicocca,
            a.url_annuncio
FROM        affitti a
            JOIN mezzi_pubblici mp
                ON mp.id = a.stazioneId
            JOIN mezzi_luogo m1
                ON m1.id_mezzi_pubblici = mp.id
            JOIN mezzi_luogo m2
                ON m2.id_mezzi_pubblici = mp.id
            JOIN mezzi_luogo m3
                ON m3.id_mezzi_pubblici = mp.id
            JOIN luoghi_interesse li1
                ON li1.id = m1.id_luogo_interesse
            JOIN luoghi_interesse li2
                ON li2.id = m2.id_luogo_interesse
            JOIN luoghi_interesse li3
                ON li3.id = m3.id_luogo_interesse
WHERE       (a."stazione_distanza(minuti)" + m1."distanza(minuti)")
            < 30
            AND (a."stazione_distanza(minuti)" + m2."distanza(minuti)")
```

```
                < 30
                AND (a."stazione_distanza(minuti)" + m3."distanza(minuti)")
                < 15
                AND li1.descrizione LIKE 'Piazza Duomo'
                AND li2.descrizione LIKE 'Stadio San Siro'
                AND li3.descrizione LIKE 'Bicocca'
                AND a."prezzo(€/mese)" > 0
ORDER BY        a."prezzo(€/mese)" ASC;
```

id	prezzo	duomo	sansiro	bicocca	url_annuncio
4896	700	23	29	13	https://www.immobiliare.it/annunci/104445943/
4918	750	22	28	12	https://www.immobiliare.it/annunci/104054543/
4879	900	21	27	11	https://www.immobiliare.it/annunci/104531715/
4907	950	21	27	11	https://www.immobiliare.it/annunci/104358165/
4910	950	21	27	11	https://www.immobiliare.it/annunci/104358389/
4908	950	21	27	11	https://www.immobiliare.it/annunci/104358953/
4905	950	21	27	11	https://www.immobiliare.it/annunci/104360797/
4906	950	21	27	11	https://www.immobiliare.it/annunci/104361801/
4925	1200	20	26	10	https://www.immobiliare.it/annunci/103894803/
4883	1370	23	29	13	https://www.immobiliare.it/annunci/104552161/

4. Goal: once an apartment is chosen from the previous query results (we chose the first one, id=4896), find the distance of that apartment from all the recreational points of interest and from the attended university (Bicocca).

```
SELECT      li.descrizione
            AS nome_luogo_interesse,
            ml."distanza(km)"+a."stazione_distanza(metri)"/1000
            AS distanza_luogo_km
FROM        affitti a
            JOIN mezzi_pubblici mp
```

```
        ON a.stazioneId=mp.id
    JOIN mezzi_luogo ml
        ON mp.id=ml.id_mezzi_pubblici
    JOIN luoghi_interesse li
        ON ml.id_luogo_interesse=li.id
WHERE   a.id=4896
    AND li.descrizione NOT IN ('Polimi Piola','Polimi Bovisa',
    'Statale sede principale')
ORDER BY distanza_luogo_km ASC;
```

nome_luogo_interesse	distanza_luogo_km
Bicocca	1,01
Alcatraz	3,34
Pinacoteca di Brera	5,09
Arco della pace	5,62
Piazza Duomo	5,82
Navigli	7,47
Stadio San Siro	9,86

5. Goal: to extract the apartment with the minimum price within a 3-kilometer radius from each point of interest.

```
SELECT  li.descrizione AS luogo_interesse,
        a.nome,
        a."prezzo(€/mese)",
        a.url_annuncio
FROM    affitti a
    JOIN mezzi_luogo ml
        ON ml.id_mezzi_pubblici = a.stazioneId
    JOIN luoghi_interesse li
        ON li.id = ml.id_luogo_interesse
WHERE   (a."stazione_distanza(metri)"/1000 + ml."distanza(km)")
    < 3
    a."prezzo(€/mese)" > 0
    AND a."prezzo(€/mese)" =
            (SELECT MIN(a1."prezzo(€/mese)")
             FROM   affitti a1
                    JOIN mezzi_luogo ml2
                    ON ml2.id_mezzi_pubblici = a1.stazioneId
                    JOIN luoghi_interesse li2
                    ON li2.id = ml2.id_luogo_interesse
```

```
        WHERE  A1."PREZZO(€/MESE)" > 0
               AND LI2.ID = LI.ID
               AND (A1."STAZIONE_DISTANZA(METRI)"/1000
               + ML2."DISTANZA(KM)") < 3)
GROUP BY  LI.ID;
```

luogo_interesse	nome	prezzo	url_annuncio
Bicocca	Monolocale via demonte 4, Prato Centenaro, Milano	575	https://www.immobiliare.it/annunci/100038438/
Polimi Piola	Monolocale via Francesco Cavezzali 11, Turro, Milano	500	https://www.immobiliare.it/annunci/86266562/
Polimi Bovisa	Monolocale piazza Prealpi 4, Certosa, Milano	375	https://www.immobiliare.it/annunci/104298685/
Statale sede principale	Monolocale via Ennio, Martini - Insubria, Milano	500	https://www.immobiliare.it/annunci/104059767/
Arco della pace	Monolocale via Luigi Manfredini, Paolo Sarpi, Milano	550	https://www.immobiliare.it/annunci/103448764/
Piazza Duomo	Monolocale via Salasco, Bocconi, Milano	550	https://www.immobiliare.it/annunci/104363959/
Stadio San Siro	Bilocale via Privata Carlo Antonio Carlone 8, Bande Nere, Milano	550	https://www.immobiliare.it/annunci/46934207/
Pinacoteca di Brera	Monolocale via Luigi Manfredini, Paolo Sarpi, Milano	550	https://www.immobiliare.it/annunci/103448764/
Navigli	Monolocale via Salasco, Bocconi, Milano	550	https://www.immobiliare.it/annunci/104363959/
Alcatraz	Monolocale via delle Querce 7, Bovisa, Milano	490	https://www.immobiliare.it/annunci/104480257/

6. Goal: to extract the three-bedroom and four-bedroom apartments (or even more) outside of Milan that have a rental price lower than the average rental price of two-bedroom apartments, sorted by price.

```
SELECT    A.NOME,
          A."PREZZO(€/MESE)",
          A.URL_ANNUNCIO,
```

```
        a.locali AS numero_locali
FROM    affitti a
        JOIN comuni c
            ON a.comuneId=c.Id
WHERE   c.comune_appartenenza <> "Milano"
        AND a.locali>=3
        AND a."prezzo(€/mese)" > 0
        AND a."prezzo(€/mese)" <
            (SELECT     AVG(a1."prezzo(€/mese)")
             FROM       affitti a1
             WHERE      a1.locali=2)
ORDER BY a."prezzo(€/mese)" ASC;
```

nome	prezzo	url_annuncio	numero_locali
Appartamento via Conte Suardi 84, Segrate Centro, Segrate	400	https://www.immobiliare.it/annunci/104442785/	5
Trilocale via Mazzini, 2, Centro - Piazza Gramsci, Cinisello Balsamo	500	https://www.immobiliare.it/annunci/103315710/	3
Trilocale via Guerciotti 33, Piscina, Legnano	500	https://www.immobiliare.it/annunci/103387542/	3
Trilocale via Fermi 20/A, Turbigo	500	https://www.immobiliare.it/annunci/100700927/	3
Trilocale via fagnani, Centro, Sedriano	560	https://www.immobiliare.it/annunci/104276869/	3
Trilocale via Giuseppe Verdi 3, Centro, Opera	580	https://www.immobiliare.it/annunci/97298074/	3
Trilocale via Papa Giovanni XXIII 34, Buscate	600	https://www.immobiliare.it/annunci/103586084/	3
Trilocale via Mazzini, 2, Centro - Piazza Gramsci, Cinisello Balsamo	600	https://www.immobiliare.it/annunci/103425798/	3
Trilocale via Alessandro Manzoni 3, Liscate	600	https://www.immobiliare.it/annunci/69782766/	3

Trilocale via Pacinotti, Centro, Magenta	600	https://www.immobiliare.it/annunci/103967555/	3

Conclusions and future developments

In conclusion, we have created a relational dataset capable of providing valuable information to those who are searching for a rental apartment in the metropolitan city of Milan, whether it be for study purposes or other reasons. This is a problem that is particularly close to our hearts as we fully experience the university life and understand its relevance, which can easily affect our colleagues or potentially our future versions in other cities.

We are satisfied with the results, despite certain process-related challenges encountered, especially during the scraping phase, particularly in the standardization and normalization of data, such as handling missing or peculiar data. However, we believe that our dataset holds immense potential, only partially highlighted by the chosen queries.

Perhaps the primary potential future development stemming from this relational model and dataset could be the creation of a website or user-friendly application that allows users to visualize all the rental listings in the metropolitan city of Milan through an interactive map. Users would be able to apply various filters based on their specific needs, as partially demonstrated in the data exploration section. For instance, in the third query of the exploration phase, we had developed the idea of delineating a zone around three (or potentially more) specific points of interest based on the minutes it takes to reach them. Obviously such a filter could be implemented also taking into account both temporal and spatial/geographic distance. Undoubtedly, a graphical visualization of such a result would greatly enhance its value and user experience.

The main challenge to effectively utilize a project of this nature lies in managing the currency of the dataset, particularly for a dynamic city like Milan and a fast-paced real estate market. It will be essential to develop an effective data cleaning phase to remove outdated listings and subsequently establish a process for updating or adding new information. Additionally, as exemplified by the observation regarding the M4 subway line, which was inaugurated during the conception of this project, public transportation data will also undergo changes and need to be appropriately managed.

References

[1] Ygnazia Cigna. “Milano, cresce la protesta degli studenti contro il caro affitti ma l’intervento del Comune (e del Mur) si fa attendere - L’inchiesta” Open, 07-05-2023.

[2] Beautiful soup
https://pypi.org/project/beautifulsoup4/

[3] https://overpass-turbo.eu/

[4] GeoPy
https://pypi.org/project/geopy/

[5] Selenium
https://pypi.org/project/selenium/

[6] https://dati.comune.milano.it/

[7] DBeaver
https://dbeaver.io/

[8] Pandas Profiling
https://pypi.org/project/pandas-profiling/

[9] https://www.metro4milano.it/cantieri/repetti/