Universitat Politècnica de Catalunya

Facultad de Informática de Barcelona

Algorithms, Data Structures and Databases

Project Report

Group

Mohammed Faiz Sayyed
Alexandros Tremopoulos

Professors

Prof. Anna Queralt
Prof. Oscar Romero
Prof. Josefina Sierra

Barcelona, October 27th 2023

Access to Development and Operations Platforms

For your convenience and to ensure transparency, we have made our project development and operations platforms accessible to you. Kindly find the links below to access the respective platforms:

1. **Development Platform (Google Drive):**

Access our project documents, code files on Google Drive. Click the following link to view and download project-related materials:Link

2. **Version Control (GitHub Repository):**

Explore our project repository on GitHub to review the codebase, scripts, and version history. Link

1 Introduction

In the realm of data analysis, this project offers a unique opportunity to conduct a multifaceted exploration of Barcelona's real estate landscape. By leveraging the provided data-sets, in-depth quantitative analyses can be performed to uncover intricate patterns and relationships between rental prices, district-wise economic disparities. Utilizing statistical techniques, such as regression analysis, clustering, and data visualization tools we can analyze rental pricing trends. Through rigorous data analysis, the project aims to reveal correlations between rental costs and income levels. Such insights are invaluable for stakeholders, enabling them to make informed decisions regarding urban development strategies, targeted investments, and social welfare programs, fostering a more equitable and economically sustainable living environment in Barcelona.

1.1 Datasets

Note: the datasets are a sample of Barcelona's population and they do not contain every household and every person living in Barcelona.

- **Rent price in Barcelona**

  This dataset contains information about rent prices in Barcelona from 2018 to 2022. It includes variables such as year, trimester district, neighborhood, and monthly rent price in euros. The data is collected over several years and provides insights into the rental market trends in Barcelona

  The columns in our dataset represent crucial attributes that provide context and insights into the housing market in Barcelona. Here's an overview of each column:

  1. **Year:**
     - Description: The "Year" column signifies the specific year in which the data was recorded.
  2. **Trimester:**
     - Description:The "Trimester" column indicates the quarter of the year (1, 2, 3, or 4) during which the data was collected.
  3. **District:**
     - Description:The "District" column categorizes different administrative areas within Barcelona, each with its unique characteristics and demographics.
  4. **Neighbourhood:**
     - Description:The "Neighbourhood" column specifies the specific neighborhoods within each district, offering a more granular view of the city's rental market.
  5. **Price:**
     - Description: This column represent the average rent prices in euros per month or per m2.
  6. **Average_rent:**
     - Description: This column(string) indicates if the row is related with average rent per month or per m2

- **Average gross taxable income per person (€/Year) for the city of Barcelona**

  This dataset is sourced from the Ajuntament de Barcelona (City Council of Barcelona) and provides data on gross income per person in different districts of Barcelona. It includes variables like district, neighborhood, average gross income per person. The dataset offers insights into the economic disparities and income distribution across various districts and neighborhoods in Barcelona

1. **Any:**
   - Description:
     The "Any" column signifies the specific year in which the data was recorded.
2. **Codi_Districte:**
   - Description: This column contains numerical codes representing different districts within Barcelona. Each district is assigned a unique code for identification purposes.
3. **Nom_Districte:**
   - Description: The "Nom_Districte" column contains the names of the districts in Barcelona, corresponding to the numerical codes in the "Codi_Districte" column.
4. **Codi_Barri:**
   - Description: Similar to the district codes, the "Codi_Barri" column contains numerical codes representing different neighborhoods (barrios) within specific districts.
5. **Nom_Barri:**
   - Description: The "Nom_Barri" column contains the names of the neighborhoods in Barcelona, corresponding to the numerical codes in the "Codi_Barri" column.
6. **Seccio_Censal:**
   - Description: This column represents the numerical codes for census sections within specific neighborhoods. Census sections are smaller geographic units within neighborhoods.
7. **Import_Renda_Bruta_€:**
   - Description: The "Import_Renda_Bruta_€" column contains numerical values representing gross income in euros (€) for the specified census sections.

2 Landing Zone

2.1 Landing Zone Implementation

In the initial phase of our project, we meticulously designed and implemented the Landing Zone, a fundamental component of our data management strategy.

We adhered to the project's recommendation and we created a dedicated folder named "landing," which serves as the repository for all files related to this zone. This centralized approach streamlined the organization and accessibility of our data files, promoting a systematic workflow throughout the project.

2.1.1 Temporal Landing:

In the Temporal Landing area, a straightforward approach was adopted. Raw data, in its original **CSV format**, was ingested through a copy operation into the designated folder located at '/landing/temporal/'. This initial step allowed for swift access to the raw data, providing a foundation for subsequent processing and analysis stages.

2.1.2 Persistent Landing:

To ensure efficient data management and facilitate future exploratory analysis, we meticulously designed the Persistent Landing area located at '/landing/persistent/'. Raw data, in CSV format, was processed using the Python **Pandas library**. Leveraging Pandas, we separated the data into versions based on relevant timestamps, creating structured datasets ready for further exploration. Each version was intelligently organized into subfolders, utilizing a naming convention consisting of the dataSourceName and timestamp at the time of ingestion. This methodical approach not only automated

the exploration process but also provided essential context, indicating the origin and timeline of each dataset.

3 Formatted Zone

In our implementation of the Formatter Zone, we employed a systematic approach to structure and format our data within a relational database environment. The chosen strategy aligned seamlessly with the project's guidelines and best practices, ensuring a robust foundation for subsequent data analysis tasks.

The use of the **os library** allowed us to effortlessly navigate the landing zone, identifying and loading all relevant CSV files into individual Pandas DataFrames. This step was crucial, as it laid the groundwork for the subsequent stages of our data processing pipeline. Each DataFrame, representing a distinct data source version, was meticulously prepared for insertion into the database.

Our implementation showcased flexibility and adaptability by dynamically generating database tables for each DataFrame. Using the **DuckDB** library, we established a seamless connection to the relational database. The ability to create tables on-the-fly ensured that our database schema mirrored the original data, accommodating potential schema variations without manual intervention.

Data integrity and accuracy were paramount during the data insertion process. Each DataFrame was meticulously inserted into its corresponding DuckDB table, maintaining the relational integrity of the database. Additionally, we conducted a thorough verification process. Retrieving and displaying metadata, including table names and column details, provided essential insights into the created schema. Verifying the first few rows of each table offered valuable assurance regarding the successful integration of data.

4 Trusted Zone

Next, comes the trusted zone, where we conduct data quality processes. The outcome is a database with a single table per dataset, by integrating separate versions. To do this, we use the formatted.db database. Data preprocessing takes place to take insights of data, with plots, descriptive statistics etc., along with searching and handling missing values, duplicates, outliers etc. We try to continue the steps automated, that can apply for every incoming dataset, but after some steps, that is not feasible.

We have implemented data processing and profiling system using two distinct notebook files tailored to handle specific tasks efficiently. The first file, leveraging SQL queries, is adept at tasks such as sorting, searching for missing values, identifying duplicates, and exploring the uniqueness of values of each feature. The second file utilizes python's **pandas** data frames, to conduct in-depth data profiling, statistical analyses, visualization and outlier handling. For the visualization **matplotlib** and **seaborn** libraries are used.

In the formatted.db different versions of 'income' and 'rents' table exist and each related filename starts with these, e.x.'income_2018_2023_10_06_15_18_19'. Here we have written a code that reads all tables, and union these that start with the same word, all 'income' tables in one, all 'rent' tables in one, and it can apply to any other possible dataset. In addition, we give data types in the columns of the table or replacing not used characters like the euro sign in column names, by iterating through the merged data. We also check for duplicates (SQL), and there were none. Then we sort the tables by year(SQL) and the two created tables (5 first rows) are:

- Table: income (Any, Codi Districte, Nom Districte, Codi Barri, Nom Barri, Seccio Censal, Import Renda Bruta, id (2018, 1, 'Ciutat Vella', 1, 'el Raval', 1, 12391, 1) (2018, 1, 'Ciutat Vella', 1, 'el Raval', 2, 9577, 2) (2018, 1, 'Ciutat Vella', 1, 'el Raval', 3, 9833, 3) (2018, 1, 'Ciutat Vella', 1, 'el Raval', 4, 12977, 4) (2018, 1, 'Ciutat Vella', 1, 'el Raval', 5, 10142, 5)

- Table: rents (Year, Trimester, District, Neighbourhood, Average Rent, Price, id) (2018, 1, 'Ciutat Vella', 'el Raval', 'average rent (euro/month)', Decimal('792.740'), 1) (2018, 1, 'Ciutat Vella',

'Gothic Quarter', 'average rent (euro/month)', Decimal('998.400'), 2) (2018, 1, 'Ciutat Vella', 'la Barceloneta', 'average rent (euro/month)', Decimal('870.840'), 3) (2018, 1, 'Ciutat Vella', 'Sant Pere, Santa Caterina i la Ribera', 'average rent (euro/month)', Decimal('923.440'), 4) (2018, 1, 'Eixample', 'Fort Pienc', 'average rent (euro/month)', Decimal('910.530'), 5)

We continue by checking if there are missing values(SQL) and we found that neither dataset had. Afterwards we check all unique values that exist in each column table. The exploration data analysis continues in python. We convert the tables to dataframes so we can explore them with pandas and python. First we are exploring the income table. Printing number of unique values per column and statistics for the income column:

- Any 3, Codi_Districte 10, Nom_Districte 11, Codi_Barri 73, Nom_Barri 80, Seccio_Censal 181, Import_Renda_Bruta 2866, id 3204
- count 3204 , mean 21236.11 , std 7670.81 , min 7328 , 25% 16418.25 , 50% 19781 , 75% 23811.25, max 47029 , Name: Import_Renda_Bruta

At this point we do some plots to understand our data better. We plot the average income of the year 4.1, from which we can observe the distributions of income among districts, with Sarria-Sant Gervasi having the biggest average while Nou Bartis the lowest. We also plot income per year 4.2, where they are almost the same. There is only one slight difference in the year 2018 with income being just a bit lower. Lastly, we plot the same for the dozens of neighbourhoods 4.3, where a big variety of income is displayed.

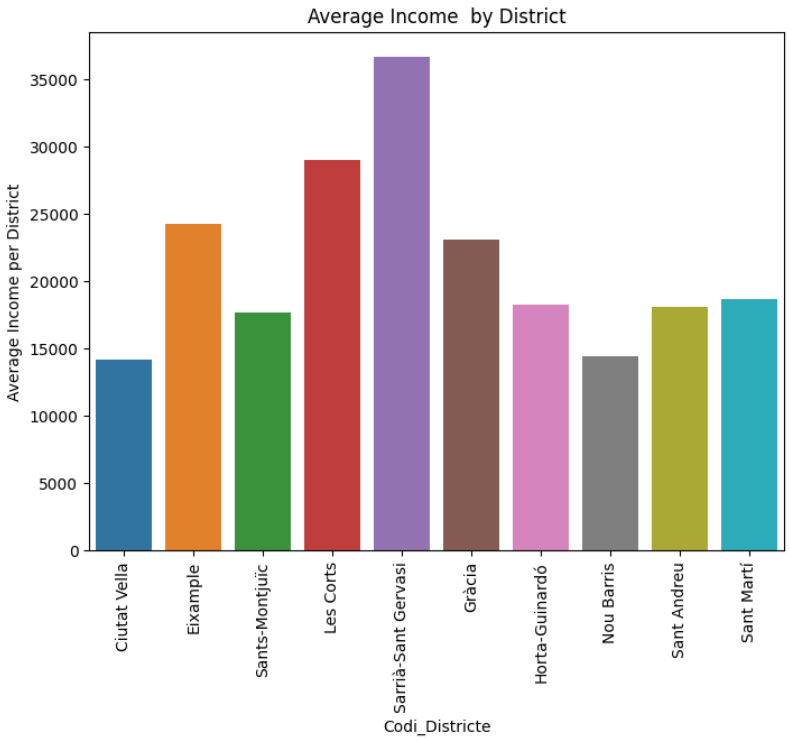

Figura 4.1: Average income per district

Similar preprocessing and exploratory data analysis takes place for the 'rents' table. Unique values in columns and statistics for rent price:

- Year 3, Trimester 4, District 10, Neighbourhood 72, Average_rent 2, Price 1084, id 1616
- count 1616, mean 458.78, std 477.11, min 6.2, 25% 13.1, 50% 185.7, 75% 845.37, max 2023.4, Name: Import_Renda_Bruta

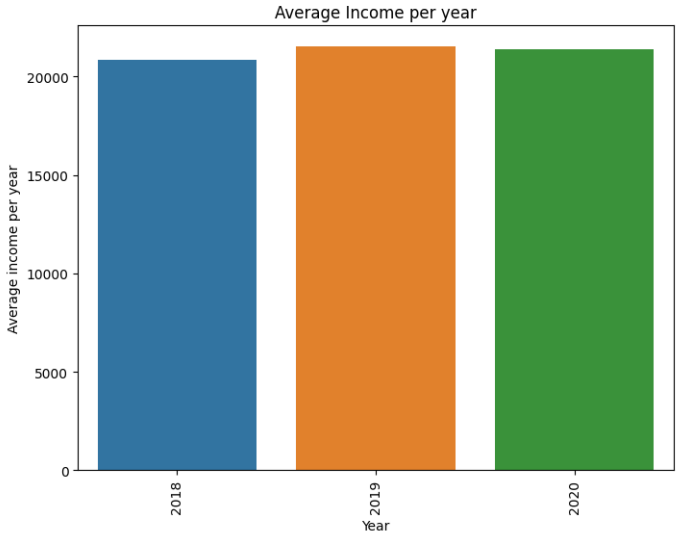

Figura 4.2: Average income per year

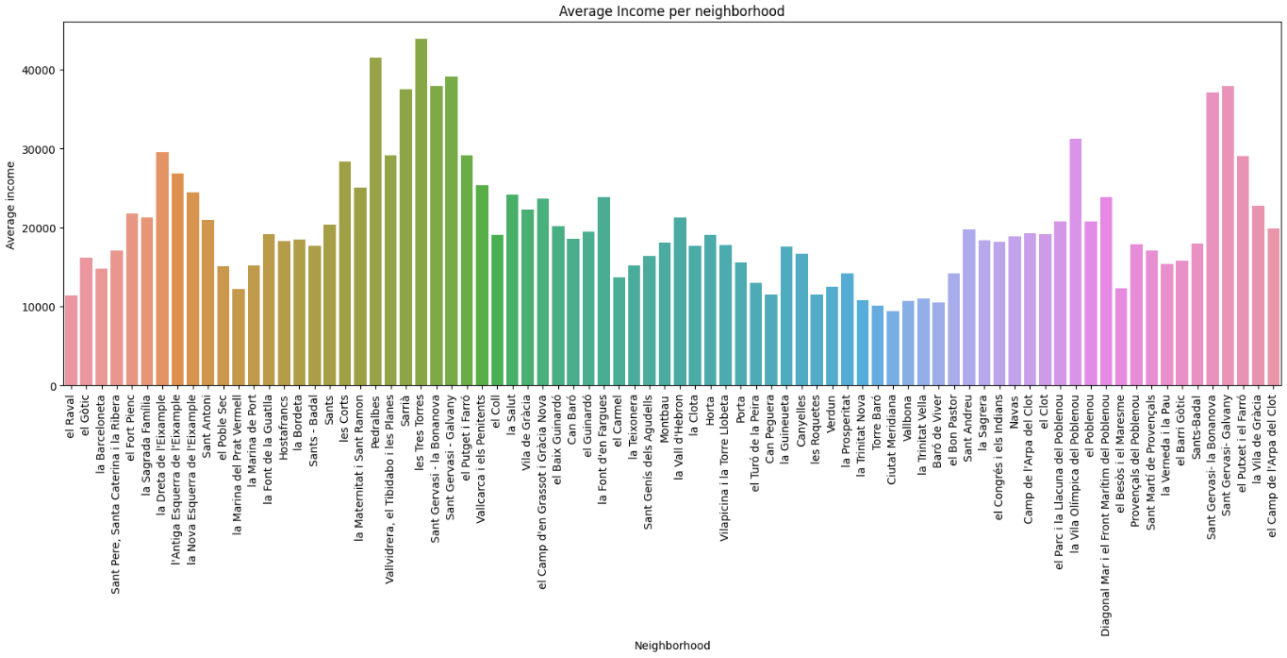

Figura 4.3: Average income per neighbourhood

And then the same logic for plots, average rent per district 4.5, per year 4.4 and per neighbourhood 4.6. Once again, Sarria-Sant Gervasi with the highest avg rent and Nou Barris with the lowest. Les Corts in second place. In addition, the plot per year is similar with the one of income. The year 2018 is slightly lower than 2019 and 2020. Lastly, the rent per neighbourhood for more detailed presentation of the rent distribution.

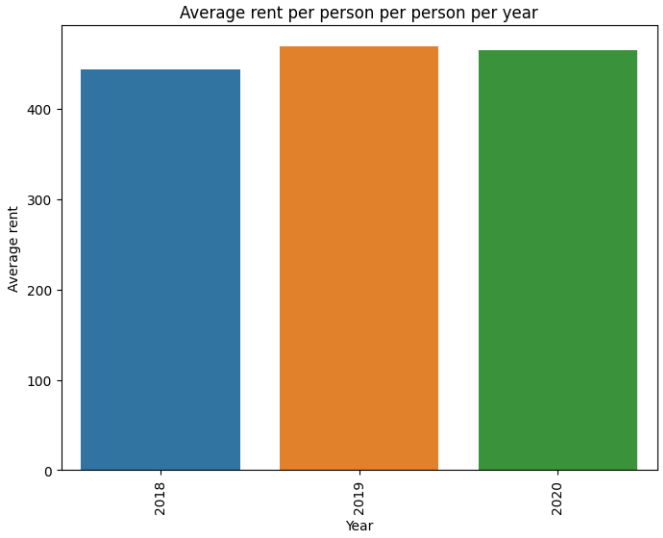

Figura 4.4: Average rent per year

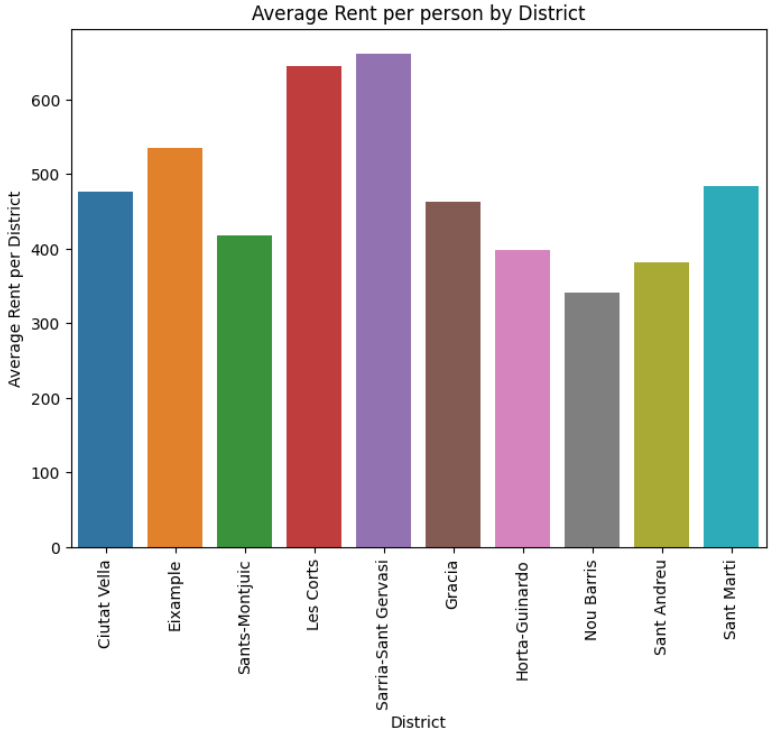

Figura 4.5: Average rent per district

The last phase of the trusted zone, is to search for outliers. This takes place for 'Import_Renda_Bruta' (i.e. income) for the income table and for 'Price' (avg rent) for the 'rents' table. The other columns are strings or code indicators show searching of outliers is not useful. A boxplot for the 'Price' column 4.7 gives no outlier, while for 'Import_Renda_Bruta' 4.8 there are many that need to be investigated. The areas with a lot of income are displayed as outliers(above upper bound). Thus, we only keep

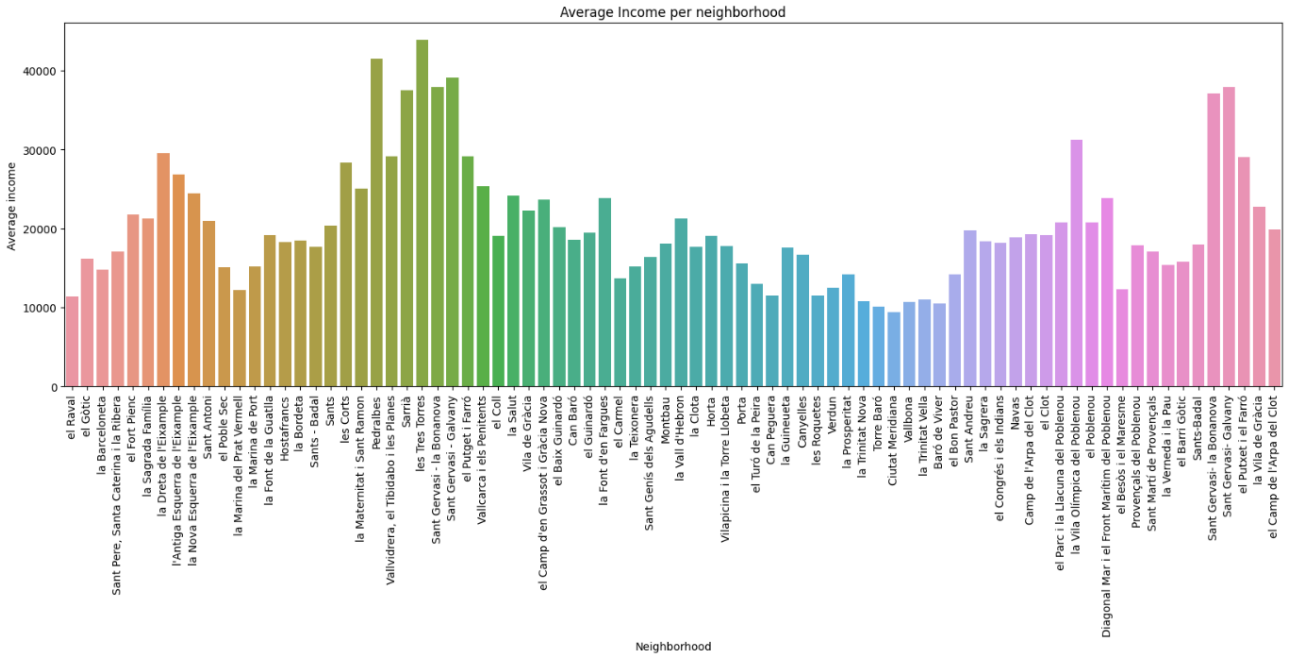

Figura 4.6: Average income per neighbourhood

extreme outliers to explore with upper bound being 46003.5 euros. Here we have 69 cases and we are exploring them further. We group by those cases by district and we see that one is in 'Eixample', 11 in 'Les Corts' and 57 in "Sarria-Sant Gervasi'. The two latter are those with the highest income above all districts and we do not think them as outliers because it makes a lot of sense. For the one case in 'Eixample', we think there is not enough evidence and that is possible that someone leaves in 'Eixample' while having a high income.

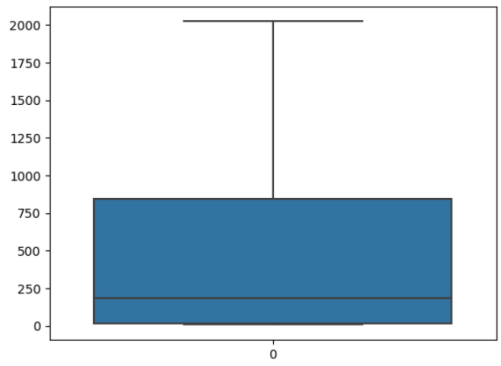

Figura 4.7: Rent boxplot

5 Exploitation Zone

The exploitation zone contains different entities with greater usage and importance by separating concepts of interest. Also, data integration and homogenization of certain values take place. Here, we've crafted two dedicated notebook files, each tailored to specific tasks, ensuring a strategic approach to extracting valuable information from our refined dataset.

In the first file , values of columns are explored, to check which need homogenization in order to be correct integrated. Both datasets have as columns the 'district' and the 'neighbourhood'. We observe that some values, even though the refer to the same place, they are written different. For example 'la

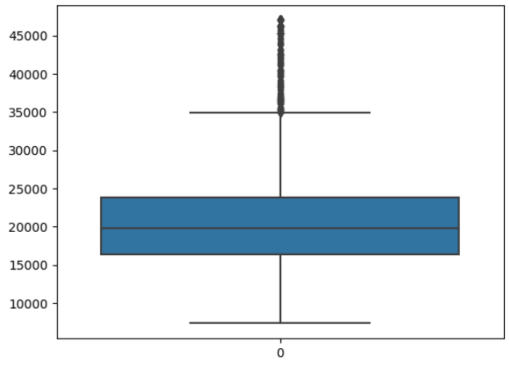

Figura 4.8: Income boxplot

Sagrada Família ' and 'la Sagrada Familia' where the one has a spanish accent while the other has not. To solve this, we apply some functions in the 'district' and 'neighbourhood' columns of both datasets. Those delete whitespaces before and after the value, convert all letters to lowercase, and decodes values so they do not contain accents and other similar characters. Now we can correctly compare the values between datasets. Then first for 'neighbourhoods' we compare the values of the two datasets and check how many of them are in both datasets and how many are only in the one of them. So, after some checking and corrections (e.x. one had 'fort pienc' and other 'el fort pienc' and they were all converted to 'el fort pienc') they have 70 common neighbourhoods, one had 9 extra and the other had 1 extra neighbourhood. Now, for districts, they had the same 10 districts, but one had both 'eixample and "l'eixample", so all values were converted to the latter one.

Second file handles the integration and the creation of entity tables in the exploitation database. It includes the dataframes that contain the distinct combinations of district and neighbourhood for both datasets. The rents datasets has 71 different combinations while the income one has 79. We merge them, and now we have 80 different combinations. We will use them for our entities.

Before presenting the result we will explain why we decided to create such entities. As mentioned many times, our datasets are related to the average rent of each neighbourhood,district per year and the income per person (sampled) that leaves in some neighbourhood/district per year. These two datasets have a few columns, besides those 3 mentioned (i.e. neighbourhood/district/year), but they are not important for our entities because they mainly are some IDs that cannot connect the two datasets, except the 'seccio censal' in income dataset. They have 3 common features and we can integrate them by these 3. There could be a few analysis in the features that can compare rents and income per area etc. But the purpose in this zone is to have general entities aside the future analysis. We thought that one important entity is the integration of distinct neighbourhoods and districts (named area) while other could be the time(year). From these two tables we can obtain the important features that both datasets use by using foreign keys in another two tables, one of them being 'rents' and the other being 'income'. 'Rents' will keep the average rent and use the area and time tables with the foreign keys. Then 'income' will use the same foreign keys and keeping 'income' and 'seccio censal' features as well.

We are creating our new database and the first table is 'area', which table contains id, district, neighbourhood fields inserting all values from the merged dataframe. Then another table called 'time' is created and it contains id and year values (only 2018,2019,2020). Two more tables are needed. One is the rents table, that should include the average rent per area and per year. We want to avoid redundancy, thus we do not write again areas and years but we use them as foreign keys of the two previous tables. Consequently, the fields are: 'ID', 'Avg rent', 'AreaID' and 'YearID'. Similar for the 'income' table, which has 'ID', 'Seccio censal', 'Income', 'AreaID' and 'YearID'. The way to create the foreign keys was possible after some preprocessing and extra columns in rent and income dataframes, so they would easily be inserted in the tables. We have four entity tables and if we print the first five rows of them, we get this:

- Table area: (District, Neighbourhood, ID) ('sants-montjuic', 'sants-badal', 1) ('sarria-sant ger-

vasi', 'sant gervasi- la bonanova', 2) ('sarria-sant gervasi', 'sant gervasi- galvany', 3) ('sarria-sant gervasi', 'el putxet i el farro', 4) ('gracia', 'la vila de gracia', 5)

- Table time: (ID, Year) (1, '2018') (2, '2019') (3, '2020')
- Table income: (Seccio Censal, Income, AreaID, YearID, ID) (1, 12391, 7, 1, 1) (2, 9577, 7, 1, 2) (3, 9833, 7, 1, 3) (4, 12977, 7, 1, 4) (5, 10142, 7, 1, 5)
- Table rent: (Avg rent, AreaID, YearID, ID) (Decimal('792.740'), 7, 1, 1) (Decimal('998.400'), 8, 1, 2) (Decimal('870.840'), 9, 1, 3) (Decimal('923.440'), 10, 1, 4) (Decimal('910.530'), 11, 1, 5)

6 Operation workflow

Our operations are organized into separate files for each zone, ensuring clarity and modularity. Each zone file contains functions for specific tasks, requiring the database path as a parameter. These functions are designed to be customizable, accommodating project-specific needs. Additionally, we've added orchestrating code to seamlessly integrate different zone components. Certain functions are optional, allowing users to adapt the system as per their project requirements, ensuring flexibility and user-friendliness in our data management backbone. Theres a Separate preprocessing folder which can be adapted for the future use as our project grows.

7 Limitations and Future Scope

There was a misunderstanding regarding the "Average Rent" column, which was later discovered to have two distinct values: "average rent (euro/month)" and "average rent per surface (euro/m2)." This realization occurred relatively late in the development process. However, these issues are being addressed in the ongoing development cycle. Additionally, it was noticed that the "Price" column had different values corresponding to each "Trimester." This variability could imply rent price fluctuations based on specific time intervals.

By openly acknowledging these limitations and outlining plans for improvement, we are prioritizing data quality assurance. This ensures that future analyses will be conducted with a more precise and comprehensive understanding of the dataset.

Our operational environment, unfortunately, does not fully embrace automation due to project-specific modifications and challenges in understanding the process of complete automation. Despite our efforts to incorporate automation aspects, the specific nuances of automating the entire workflow, especially in the context of our unique dataset and project requirements, posed challenges. This lack of complete automation highlighted the need for further exploration and understanding of automated processes, ensuring a more streamlined and efficient workflow in future projects.

We also believe that we made correct choices of using Google Drive, Github, Python (and all libaries) and DuckDB. We do not consider of replacing them in the future.