Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN SISTEMAS DE INFORMACIÓN

Design, implementation and exploitation of a data warehouse about basketball statistics from the main European leagues.

Student: Ángel Meijón de la Fuente

Direction: Susana Ladra González

Hilda Romero Velo

A Coruña, June 2024.

To my parents and my sister.

Acknowledgements

Thanks to my family for being with my during every moment of this journey. Also, thanks to everyone that I´ve met during all this years, you made it easier and more fun.

Abstract

Nowadays, data analysis and manipulation is a critical aspect in every working sector of our society. These include many subjects such as medicine, finances, education, industry... The reason behind this enormous growth in interest for data is that it allows companies to make decisions that are backed by statistics that suggest that they are correct. It also allows to check the results from previous decisions.

As I mentioned before, data analysis is present in every sector, and sports are no exception. This project will consist on showing how a good use of data could help basketball teams and players to be more successful. In order to do it, we will create a Data Warehouse based on the statistics collected from the main European basketball leagues during the last 5 seasons.

Firstly, the tools required for this project will be explained, as well as the strategy that will be used during its development. Then, we will show the entire development of creating the Data Warehouse starting from scratch, doing the ETL process that will allow us to manipulate it and finally exploiting it in order to obtain useful conclusions.

Resumo

Hoxe en día, a análise e manipulación de datos é un aspecto crítico en todos os sectores laborais da nosa sociedade. Estes inclúen moitos sectores como a medicina, as finanzas, a educación, a industria... A razón detrás deste enorme crecemento do interese polos datos é que permite ás empresas tomar decisións apoiadas por estatísticas que suxiren que son correctas. Tamén permite comprobar os resultados das anteriores decisións.

Como comentei antes, a análise de datos está presente en todos os sectores, e os deportes non son unha excepción. Este proxecto consistirá en mostrar como un bo uso dos datos pode axudar a equipos e xogadores de baloncesto a ser máis exitosos. Para facelo, crearemos un Data Warehouse a partir das estatísticas recollidas das principais ligas europeas de baloncesto durante as últimas 5 tempadas.

En primeiro lugar, explicaranse as ferramentas necesarias para este proxecto, así como a estratexia que será utilizada durante o seu desenvolvemento. Despois, mostraremos todo o traballo de creación do Data Warehouse comezando de cero, facendo o proceso ETL que nos permitirá manipulalo e finalmente explotándoo para obter conclusións útiles.

Keywords:

- Data Warehouse
- Web scraping
- ETL proccess
- Data exploitation

Palabras chave:

- Almacén de Datos
- Raspado web
- Proceso ETL
- Explotación de datos

Contents

List of Figures

List of Tables

Chapter 1
Introduction

First of all, we will start by introducing our data warehouse project. In this section, some key aspects of the project will be explained, such as its motivation, its goals and the structure of steps that will be followed during this process.

1.1 Project motivation

Nowadays, data control and analysis is a fundamental piece for success in every single aspect of our society. Thanks to the current place where technologies are, data is very accessible and its correct use usually gives companies competitive advantages in comparison to their competitors [3].

Obviously, the world of sports is no stranger to this situation. It is very frequent to see professional teams and athletes take advantage of data analysis to improve their performances, no matter what sport we are talking about.

This is the reason why we figured that the design, implementation and exploitation of a data warehouse about basketball statistics would be an interesting approach for this project. In order to do this from scratch, we need to build our own database, perform the entire Extraction, Transformation and Loading (ETL) process and finally exploit the resultant data warehouse.

Once the project is finished, we will be able to obtain good visual representations of the stored statistics, similar to what professional teams have to help them making their decisions. Also, we can use the obtained results to see if we can find things that may be different from what would be expected from an outside point of view.

To sum up, the idea behind this project is to put ourselves in the place of a professional basketball team, and use data to analyze patterns and interesting aspects of the game that may be difficult to see without data mining techniques. Also, we have the added complexity of designing and building our own database from scratch, so that will be the first thing that

we will need to take care of in the project.

1.2 Project goals

As it was previously mentioned, data warehouses are very useful for companies as they are frequently used to help their decision making in different aspects. Bringing this situation to the world of sports, and specifically, basketball, we can see this project being useful because it can give an extensive perspective on many players and teams performances during a considerable amount of time.

The main goal of this project would be to build an analytical environment that could be useful for decision making in relation to the game of basketball. This would also allow us to obtain visual representations of the statistics about all the players and teams that figure in the data warehouse.

To do this, we will design and create from scratch our own database. Once it is properly created, following all the restrictions we set at the designing phase, we will perform the ETL process to make sure it follows the required business logic. Finally, we will be able to exploit the resultant data warehouse to analyze and study the data with the help of visual elements.

1.3 Project structure

Here is a list of the different chapters that we will go through during this project:

- Chapter 1, Introduction: brief first look at the project. It includes its motivation, goals and structure.
- Chapter 2, Theoretical concepts: description of some key concepts that will be important for the project, including the operational and analytical environments, web scraping, ETL process and bussines intelligence.
- Chapter 3, Technologies and Tools: detailed view of the different technologies and tools that will be utilized, with special focus on their relevance in the project.
- Chapter 4, Methodology and planning: explanation of the methodology that will be followed during the project. It focuses on the reasoning behind choosing that methodology, and why does is fit our project. Also, we can our detailed project planning.
- Chapter 5, Analysis: presentation and explanation of our domain. We take a look at our starting point, how we want our domain to be and also what questions from this domain will be answered after our project is completed.

- Chapter 6, Project design: conceptual, logical and physical designs built for our project. These have a huge importance because they will serve as a base for future steps.
- Chapter 7, ETL process: realization of the entire ETL process. It is typically the part of data warehouse projects that consumes more time and resources.
- Chapter 8, Data Warehouse exploitation: in depth review of the obtained graphics and visual representations of the data stored after being properly exploited. This is the section where we can see the interesting aspects and patterns that we were looking for.
- Chapter 9, Conclusions: reflection of the work done, focusing both on how it was done and how it could be used for future purposes.

Chapter 2

Theoretical concepts

In this second chapter, some key concepts that are part of the project will be explained.

2.1 Operational environment

The operational environment is a fundamental piece for a data warehouse project because it provides the necessary data to fill the the system. We can define it as the set of systems and processes that are in charge of transactional data management and the business operations. This environment must be correctly managed and optimized in order to ensure that all the data is precise, consistent and available when required for future steps of the project like the ETL process. We often find some challenges when working with raw data:

- Data absence.
- Wrong data.
- Inconsistent data that raises contradictions.
- Data mistakes.

With this project, we want to transform this raw data so it is easier to manipulate and exploit it. The objective would be to start with this raw data and end up with organized knowledge that could help in decision making.

2.2 Analytical environment

The analytical environment main purpose is allowing us to analyze our data and make informed decisions based on it. It provides the tools and technologies for us to properly analyze the data, generate reports and visual graphics and ultimately making decisions based on this data.

Analytical databases have some advantages such as:

- Data veracity.
- Data consistency and coherence.
- Data cleaning and purification.

Data warehouses are a good example of analytical databases. They manage big quantities of data and are more often built with long term decision making purposes [4]. That is the reason why a data warehouse is a good fit for what we want to achieve in this project.

2.3 Web scraping

Web scraping is a very common technique that consists on extracting data from websites. It is typically used to gather data that ends up being copied into a database or a spreadsheet [5]. Scraping a web page involves fetching it and collect data from it. Fetching a proper website is a key component of this process. It is very important that you find a website containing everything you need from it, and you also need to make sure that it is allowed to scrape on it, since there are websites that do not permit this for privacy or security reasons. Therefore, web crawling (automatically accessing a website and obtaining data via a software program) is a main component of web scraping. Once fetched, extraction can take place.

There are many ways to apply this technique, even there are some applications specifically built for this purpose (Octoparse or Apify, for example). In our project, the web scraping will be done implementing our own script. We will use a series of Python codes that scrape the given websites by providing the corresponding links and specifying what we need from the web structure. This code will save all the information scraped into Comma-Separated Values (CSV) files, that will be used as a starting point for other steps of the project.

2.4 ETL process

This process consists on obtaining the data from the original sources and loading it into the data warehouse that will conform the project. It is named after its 3 main steps: Extraction, Transformation and Load [6]. This is a critical part of the project, and it is often the process that takes most of the time and resources in real life data warehouse projects. It is a periodical process where data is loaded with a predefined periodicity. Below, we will see in detail each step of this process.

2.4.1 Extraction

In this phase, we obtain the raw data directly from our sources, and we temporally store it as a starting point for the following steps. These sources we obtain data from can be many different things: a public website, a company database...

Once all the required data is extracted, we can either delete this raw data or keep it as backup since we may end up needing it in case of a mistake that causes a restart of the project, which is probably the more recommended option.

2.4.2 Transformation

This second phase will focus on making sure that the data we have follows the requirements that were set before the start of the ETL process. There must be a series of rules applied to the extracted data in order to prepare it for the Load phase. Some of the most common errors solved in this phase are:

- Non correspondent type values (double where it should be integer, for example)
- Not allowing repeated primary keys
- Data uniformity (values that mean the same but are not exactly equally written)

2.4.3 Load

This final step of the process consists on loading the data into the end target, which can be any data store. In most cases, like it happens in this project, the transformed data will be loaded into a data warehouse. Once this point is reached, the data should be ready to be analyzed and exploited. It is important to make sure that it follows rules set at the design that was built during the early stages of the project.

2.5 Business intelligence

Bussines intelligence is a fundamental aspect for any company that wants to put their data into good use and utilize them to make informed decisions. It consists on a series of strategies, technologies and processes used to collect, transform, analyze and exploit data [7]. Some of its main characteristics are:

- Correct and reliable data.
- Data distribution, so all the people that need to use the data to make decisions can easily access it.
- Data analysis.

Chapter 3

Technologies and tools

Now that the theory of the project is explained, it is time to get to know what technologies and tools will be used to carry out the entire process. We will see specific methods and tools for each step of the project.

3.1 Pentaho Data Integration

Pentaho [8] is a free set of tools dedicated to the correct development of the ETL process. Its first version was created in 2004 and currently it has two different editions: Community Edition and Enterprise Edition. For our project, Community Edition will be enough.

Pentaho has an extensive number of tools that can be useful while working on an ETL process, some of them are:

- Running web scraping scripts, which allows users to get their raw data directly from the same application where they will process it later.
- Loading raw data from different sources: a CSV file, an existing database, a JavaScript Object Notation (JSON) script...
- Allowing to manage Slowly Changing Dimension (SCD), with the proper tools for each possible scenario (overwriting elements, creating new versions of elements...).
- Allowing, once the process is finished, to load the processed data into an existing database.

These are some of the reasons why Pentaho is the tool chosen to do this part of the project. Also, there are other interesting aspects about it. For example, the fact that it allows you to create separate transformations for each dimension in your project, so if a mistake occurs in one of them it should not affect the rest of your work.

3.2 PostgreSQL

PostgreSQL [9], also known as Postgres, is a free and open-source relational database management system. It was initially released in 1996. It features, among many other things, transactions with Atomicity, Consistency, Isolation, Durability (ACID properties), automatically updatable views, triggers, foreign keys, and stored procedures. It is supported on all major operating systems, including Linux, macOS, and Windows, and handles a range of workloads from single machines to data warehouses or even public web services with many concurrent users.

Postgres will be a key element of this project because it will be used to store the data once it is properly processed. Essentially, we will have our data warehouse in Postgres. It is a solid tool for this purpose since it is quite accessible to the rest of the tools of the project (it is easy to store data in it using Pentaho and it is also easy to obtain data from it using Power BI), and also it has a user friendly interface that allows us to have a clear perspective of our data warehouse.

3.3 Power BI

Power BI [10] is an interactive data visualization software product developed by Microsoft in 2011. It has completely free and easily accessible version which will be enough to carry out this project. It is capable of translating the data from different sources, usually a data warehouse, into graphs, maps, tables... It could also be used for the ETL process, but in this project that will not be the case since it is more desirable to use a tool such as Pentaho that is specialized in that aspect.

One of the main reasons behind the selection of Power BI for this final part of the project is the fact that it is a very interactive and visual software, which allows you to see in real time the results of your work and provides a lot of services and alternatives for portraying data. This tool has a very user friendly interface, shown in figure 3.1. As we can see in the mentioned image, it has three panels at the right of the screen:

- The first one allows users to pick any attribute they want from their tables, which need to be previously loaded from the database.
- The second one allows users to select the graphic they prefer among plenty of options: columns graphic, area graphic, map graphic...
- The third one allows users to filter their information in case they do not want everything in their database to be shown.

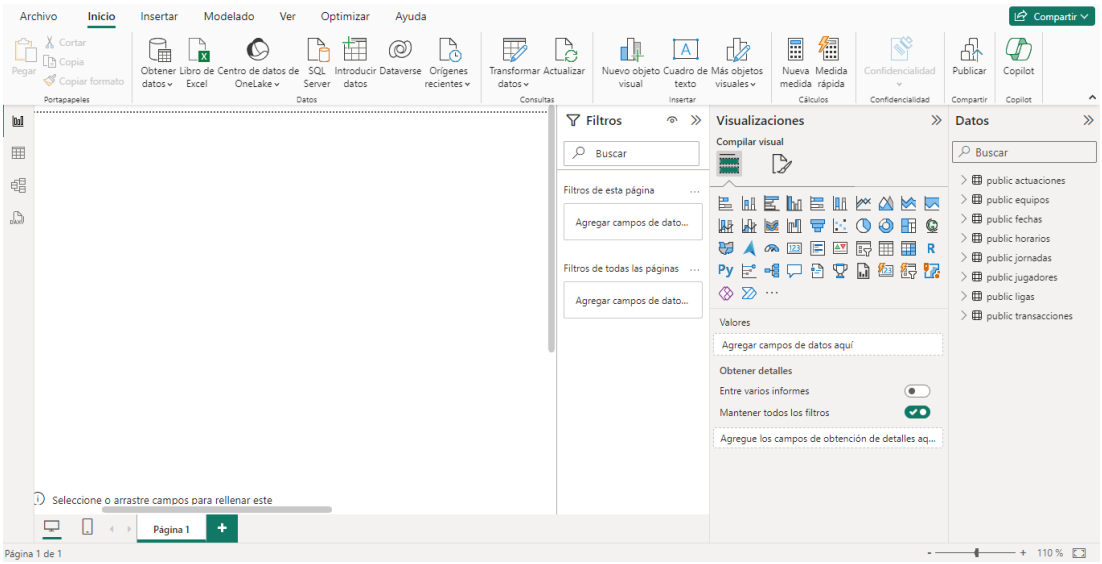


Figure 3.1: Power BI interface

3.4 BeautifulSoup

Beautiful Soup is a Python package for parsing HyperText Markup Language (HTML) and eXtensible Markup Languag (XML) documents. It creates a parse tree for documents that can be used to extract data from HTML, which is really useful for web scraping. It was first developed in 2004 and it is in constant renewal, witch its last version being released in January 2024.

Beautiful Soup receives the required website link or list of links from the user, represents the parsed data as a tree and iterates over it using ordinary Python loops.

3.5 Diagrams.net

Previously known as draw.io, diagrams.net is a cross-platform graph drawing software firstly developed in 2010. Its interface can be used to create diagrams such as flowcharts, wireframes, Unified Modeling Language (UML) diagrams, organizational charts network diagrams...

It is available as an online web app, and as an offline desktop application for Linux, macOS, and Windows. In this project, the web app was used to develop some conceptual graphs used at the beginning of the project.

Chapter 4

Methodology and planning

In this chapter, we will see in full detail the methodology followed to carry out the project, from its beginning to its final steps. We will also analyze the plan that was designed to do it, and compare it to how the job was actually done during this period of time.

4.1 Methodology

As it often happens with this kind of projects involving Data Warehouses, the Kimball methodology was the one chosen for the project.This methodology was first designed by the American author Ralph Kimball more than 30 years ago, and it has been the most popular method to carry out data warehouse projects for a long time. One of the reasons behind its huge success is that this methodology is very focused on the final users.

The Kimball Lifecycle methodology covers a sequence of high level tasks for the effective design, development and deployment of a data warehouse, or any other business intelligence system. Some of its recommendations are:

- Taking a "bottom-up" approach to data warehousing, contrary to the older "top-down" approach that used to be followed in these projects.
- Using flexible techniques and designs that allow us to adapt to any given change in any given moment.
- Giving a big importance to the business requirements. It is important to have good communication with business users in order to understand what they expect from us.
- Frequent and agile updates on the work done.
- Focusing on the project maintenance overtime, to make sure it can last over the years.

We can see a detailed view of the Kimball Lifecycle methodology and its different phases in image 4.1. As we can observe, after the project planning and business requirements definition, there are three differentiated phases in this methodology.

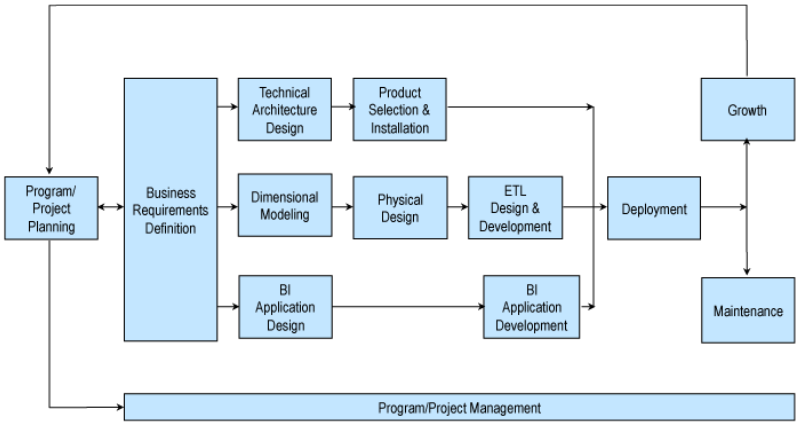


Figure 4.1: Kimball Lifecycle methodology overview [1]

Firstly, we have the technical architecture design, followed by the product selection and installation. This phase consists on making sure that all the data is properly integrated and that all the technologies and tools are compatible so we will not find problems in future steps.

Secondly, we have the entire process of designing our database and performing the ETL process on it. It will be the nuclear phase of our project.

Thirdly, we find a third phase that involves everything related with Business Intelligence (BI) at the application level.

Finally, once these steps are completed, we find the deployment of the elaborated system, which includes the previously mentioned maintenance and the growth that it may experience overtime.

Following all these phases and recommendations, the project was carried out in an agile way, trying to have as much communication as possible with the tutors to make sure that the growth was constant and controlled. This allowed us to easily manage every change or inconvenience that was found during the project, and it also helped us obtaining the desired results.

4.2 Planning

The planning part of a project is fundamental for its proper development [11]. It involves every part of the project, and it is usually its the starting point.

4.2.1 Estimation

The start of the project consisted on planning its different sections. Project planning is a key factor of every project because it helps improving many areas, such as:

- Time management: knowing what you are going to do and how you want to do it from the beginning will allow us to manage the time we have in a proper way.
- Quality assurance: if the project is correctly planned, we would ideally have enough time and resources to carry out the project in a way that ensures the final product to be a quality product.
- Project efficiency and productivity: if we carefully plan the steps of the project, we will know what to do all the time, so no time would be wasted.
- Realistic goals setting: knowing what we will do from the beginning will allow us to set goals that we will be capable of reaching.
- Budget control: for projects that require any amount of money, process planning is fundamental in order to adjust the estimated expenses to the provided funds. Although this is not the case in this project, as no financing has been allocated for it.
- Client satisfaction: if the project is correctly planned, both the work itself and the final result will be better, which would increase the client satisfaction overall.

4.2.2 Follow-up

Once the planning process was finished, it was time to follow the progress to check if it was adequate considering what was originally planned. This part of the project consists on using the project planning as a baseline to compare the estimated progress to the actual progress that you were able to obtain.

In order to see and follow this process, we have two tables (tables 4.1 and 4.2). The first one is a table where we plan our project step by step, trying to estimate how long each part of the project will take before it is finished. The second table is obtained once the project is completed, and it gives us the information of how much time each step of the project took to be carried out. Both tables also includes the dates when each step was estimated or realized.

The comparison between these two tables allows us to check how precise our estimations were compared to what ended up happening. This way, you can easily see how you performed in the development of the project.

Task	Estimated time in days	Estimated dates
Project planning	12	Jan. 19th - Jan. 31st
Project designing	20	Feb. 1st - Feb. 20th
Data gathering	8	Feb. 21st - Feb. 28th
Data scraping	27	March 1st - March 27th
ETL process	45	March 28th - May 11th
Data exploitation	16	May 12th - May 28th
Memory writing	121	Jan. 31st - June 15th

Table 4.1: Full project estimated times and dates, separated by tasks.

Task	Actual time in days	Actual dates
Project planning	10	Jan. 19th - Jan. 29th
Project designing	24	Jan. 30th - Feb. 22nd
Data gathering	7	Feb. 22nd - Feb. 29th
Data scraping	31	March 1st - March 31st
ETL process	41	April 1st - May 11th
Data exploitation	12	May 12th - May 24th
Memory writing	119	Jan. 31st - June 13th

Table 4.2: Full project actual times and dates, separated by tasks.

4.2.3 Costs estimation

According to many sources [12], the salary of a data analyst in Spain rounds 12€ per hour. If we adapt this salary to the companies sector, it would have a slightly bigger cost than that because of all the extra expenses a company has. With this in mind, a total of 15€ per hour looks right for this kind of project.

We will assume that each worked day present in table 4.2 implied 4 hours of work. Multiplying the cost per hour by the number of hours that this project required (barring the memory writing), the total expected amount would be around 7500€.

Chapter 5
Analysis

This chapter will consist on explaining the domain of the project. In order to do it, it is important to start by answering the following question: "Where do I start from?". Once the starting point is set, we will be ready to fully analyze the domain, as well as the analytical necessities that the project will cover once it´s completed.

5.1 Starting point

One of the biggest challenges that this project presents is that we will not be working with an existing data warehouse provided by a company, as it usually happens in this kind of projects. We will be starting from scratch, so the design and implementation of the data will be among the first things we need to take care of.

Since we know that our future database will allow us to analyze players and teams performances, it is very important that our domain focuses on how the players and teams performed over a considerable period of time. It is also interesting to find what circumstances affect them the most, for example the time at which they play their games, the opponent they are facing, the number of minutes they have played during a specific amount of time...

Now that we know what we want to study, we can start building our domain.

5.2 Domain description

As it was previously mentioned, our project will consist on analyzing the performances of basketball players and teams. In order to do it, we will build a database with information about the last 5 seasons (from the 2019-2020 season to the 2023-2024 season) of the 10 main European basketball leagues.

The list of covered leagues is the following:

- Spanish ACB (Spain, Andorra)

- Turkish BSL (Turkey)
- Italian Lega Basket Serie A (Italy)
- French Jeep Elite (France, Monaco)
- German BBL (Germany)
- Adriatic League Liga ABA (Bosnia, Croatia, Serbia, Montenegro, Slovenia, North Macedonia)
- Greek HEBA A1 (Greece)
- Israeli BSL (Israel)
- Lithuanian LKL (Lithuania)
- Polish OBL (Poland)

From each of these 10 leagues, we will extract data about their teams and players, which will allow us to analyze their differences and similarities, and we will also be able to compare them in many different aspects.

Another important aspect of the domain is that, apart from the players and teams statistics, we will also obtain data about the player transactions between this 10 leagues. This will allow us to see what leagues bring more players from the rest, what leagues export more players to the rest and many other interesting aspects.

5.3 Analytical necessities

Now that we comprehend how the domain will be, we can figure out how we are going to use it. An efficient way to find this out is to elaborate a series of queries that, ideally, we should be able to answer once the project is completed. Here are some of these questions:

- Which position play the players that average the most points in each league? Is is similar in every league or does it vary?
- How does the age of a player affect their performance? At what age do most players reach their best statistics?
- How does the height and weight of a player influence their impact of the game? Do taller players get better defensive stats than the shorter ones?
- How does the importance of a game affect players? Do players tend to improve or worsen their performances during playoff games compared to the regular season?

- Do players perform better if they play in their home country league, or do overseas player perform better?
- What players perform well even when their teams do not win?
- How does the time at which a game is played affect the players performances? Do players tend to play better in the mornings, afternoons or nights?
- How does the opponent affect the players performances? Are there players that perform better against quality teams compared to below average teams?
- How does the week day at which a game is played affect the players performances? Do players tend perform better during week days or during the weekends?
- Which leagues bring more players from the other ones? Which ones bring the lesser amount?
- Which leagues bring more players directly from another team? Which ones bring more free agent players?
- Which leagues complete more transactions? Is it related to the strength of the league?

Chapter 6

Design

This next chapter will consist on showing the designs that were made at the beginning of the project. This series of designs will end up being converted into a physical model, which will be the content of the data warehouse that we will explode in the final step of the project.

Before beginning our designing phase, it is interesting to point out a series of advantages and disadvantages of the practices that will be followed. Some of the main advantages from our method are:

- The modeling is fast to construct as no normalization is involved, which means swift execution of the initial phase of the data warehousing design process.
- The use of star schemes allows most data operators to easily comprehend it because of its visual structure, which simplifies querying and analysis.
- It enables fast data retrieval from the data warehouse, as data is segregated into fact tables and dimensions.
- Conformed dimensional structure for data quality framework. The Kimball approach to data warehouse life-cycle is also referred to as the business dimensional lifestyle approach because it allows bussines intelligence tools to deeper across several star schemes and generates reliable insights.

However, we also find a few disadvantages that we will need to deal with:

- Data is not entirely integrated before reporting, so the idea of a 'single source of truth' is lost.
- Performance issues may occur due to the addition of columns in the fact table, as these tables are quite in-depth. The addition of new columns can expand the fact table dimensions, affecting its performance. Also, the dimensional data warehouse model becomes difficult to alter with any change in the business needs.

- As the Kimball model is business process-oriented, instead of focusing on the enterprise as a whole, it cannot handle all the Business Intelligence (BI) reporting requirements.

With all these aspects in mind, we can start desigining our system.

6.1 Conceptual design

For this design, we will use a kind of diagram called Dimensional Fact Model (DFM). These diagrams, as indicated by their name, show the different dimensions and facts of the project [13]. They are very common in data warehouse project since they are simple to understand and show plenty of information.

6.1.1 Theoretical introduction

Before showing the DFM diagrams, we should start with a brief theoretical introduction to fully understand what we are designing at this early stage of the project.

First of all, we must understand the differences between what a dimension is and what a fact is.

We define a **fact** as a key factor for decision making that usually reflects an event that occurs in our domain. These facts often have numeric attributes that allow us to quantify certain analytical values. These values are called **metrics**. Metrics can be categorized in 3 types:

- additives: they can be added in every dimension.
- semi-additives: they can be added in just some specific dimensions.
- non-additives: they can not be added in any dimension.

Finally, we have **dimensions**. Dimensions are companions to facts, and they help us describing the objects in a fact table. A fact tends to have multiple dimensions, which define its minimal granularity.

Star schema is the fundamental element of the dimensional data warehouse model. The combination of a fact table with several dimensional tables is frequently called the star schema because of its shape. Kimball dimensional modeling allows users to construct several star schemes to fulfill various reporting needs. One big advantage of star schema is that small dimensional-table queries run instantaneously. We can see a visual representation of how a star schema works in figure 6.1.

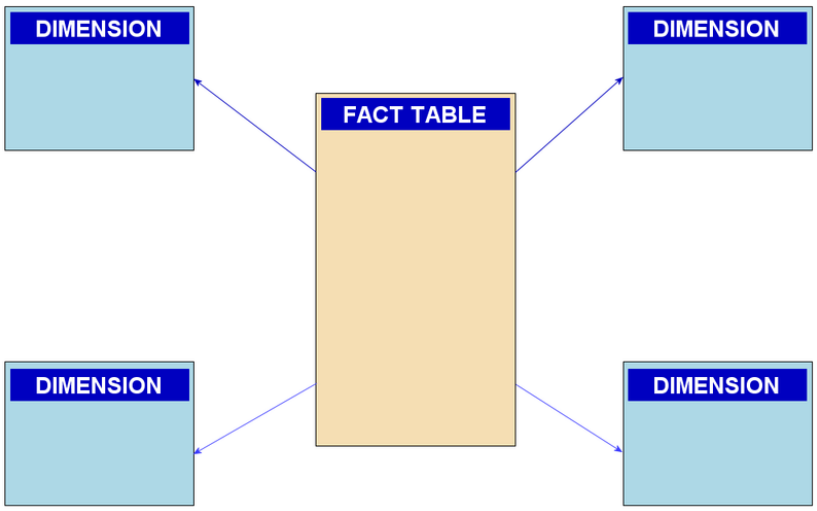


Figure 6.1: Basic overview of a star schema [2]

6.1.2 Project facts

In this project, we will be working with two facts: **performances and transactions**.

About performances, we will store information about the player and the team that he plays for and that he is facing. We will also collect some data about every specific game that is played (hour, shift, type of game...). Regarding the performance in the game itself, the following metrics will be considered:

- Number of minutes played.
- Number of points scored.
- Number of assists given.
- Number of defensive rebounds taken.
- Number of offensive rebounds taken.
- Number of steals carried out.
- Number of blocks carried out.
- Percentage of two-point baskets scored.
- Percentage of three-point baskets scored.
- Percentage of free throws scored.
- Number of turnovers committed.

- Number of personal fouls committed.
- Result of the game (Win or Loss).

The transactions fact will be different compared to the performances one. The main differential factor is that it does not have any metric, since some possible numerical attributes that could be interesting to store, such as the prize of the transaction, are in most cases confidential and cannot be extracted from the internet. This fact will focus on storing information about the players that arrived or left this 10 leagues, taking into account factors as what league they come from or what team they used to play for.

6.1.3 Project dimensions

In order to deploy all the required information, we ended up with the following dimensions:

- Player: it stores all the relevant information about every player of the database. It has the following attributes: name, age, height, weight, position, nationality, birth city.
- Team: it stores the information about the teams of the database. Its attributes are: name, country, city, foundation year, capacity of their arena and league position (for each season, we store the position of the previous one).
- League: it stores the information about the 10 leagues mentioned before that will make up our data warehouse. It has these attributes: name, country and ranking (the corresponding number for each league if we order them by level).
- Date: this dimension is critical in every data warehouse project. It allows us to store information about the dates on which the games of our warehouse were played. It has the following attributes: full date, week day, month day, fortnight, month number, quarter and year.
- Timetable: it allows us to store information about the specific hour that each game was played at. It has these attributes: hour, shift (morning, afternoon or night).
- Game: it stores information about the type of game that is being played. It has this attributes: game number (the first game of a team in the league will be number one, the second one number two and so on), type (regular season or playoffs).

It is important to take into account that we will have many **shared dimensions** in our project. Shared dimensions are two or more dimensions that have the exact same attributes, so they belong as one in our diagrams, even if later in the project we will differentiate them. This will be the case in many examples, such as the origin league and the destination league in a transaction, or the own team and opponent team of a player in a game.

Taking all this information into consideration, the resultant DFMs are shown in figures 6.2 and 6.3.

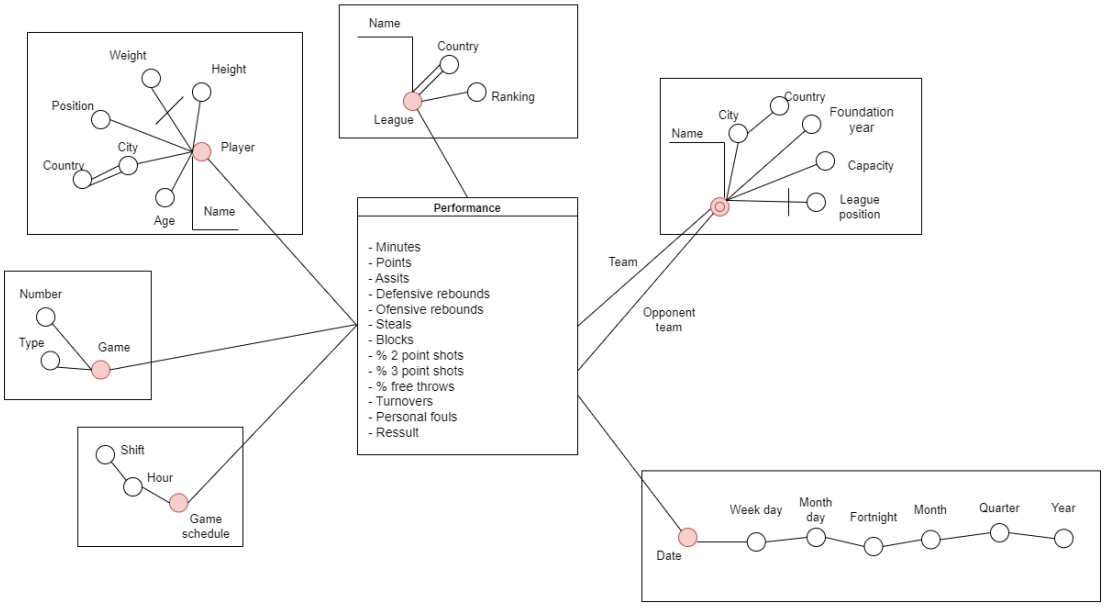


Figure 6.2: Performance DFM

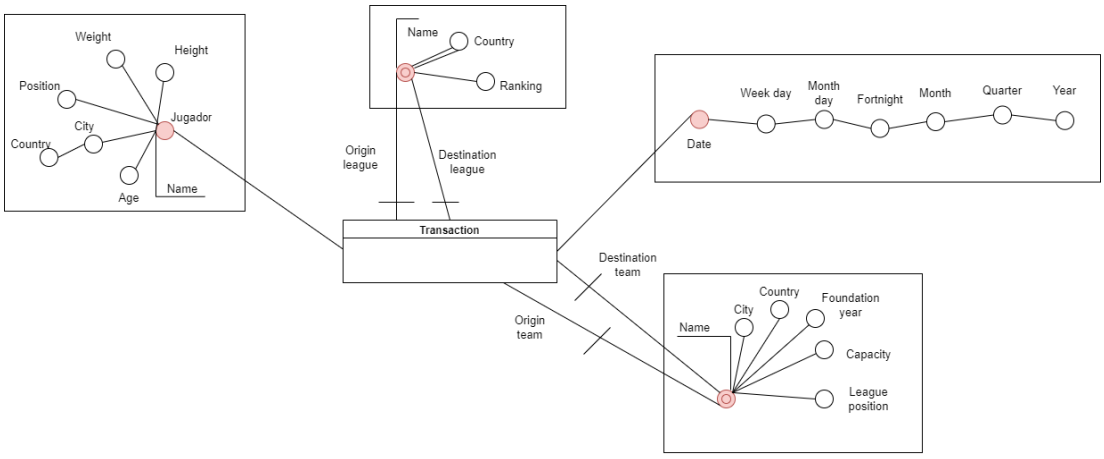


Figure 6.3: Transaction DFM

6.1.4 Change management

Now that we have introduced every dimension and fact that will conform our project, it is important to see how we are going to handle those dimensions that have attributes that vary over time. Here are the situations of our data warehouse that will be suffering changes:

- Players age: the age of every player will increase by one on a yearly basis.
- Players height and weight: sometimes players can grow from one season to another. They can also gain or lose weight.
- Players position: players can change their position from one season to another.
- Teams league position: even if it could happen, most teams do not finish in the same league position for two years on a row, so this attribute will be adapted to be changed on a yearly basis.
- Teams arena capacity: since this data warehouse includes the years where leagues where affected by COVID restrictions, some teams may have played in different arenas during some seasons, which means that the arena capacity could change from one year to another.
- The dimensions league, date, timetable and game will not suffer changes, so it will not be necessary to adapt them for change management.

All of these changes will be managed using Slowly Changing Dimension (SCD). Ralph Kimball divided SCD in three different types [14]:

- SCD type 1: it consists on overwriting the old values with the new ones. This option works well if you do not need to keep registers of old values, nor study the evolution of your attributes.
- SCD type 2: it consists on creating a new version of the attributes every time they change, also keeping the older ones. This option fits projects that want to follow the evolution of the attributes over time.
- SCD type 3: it consist on adding new columns to the attribute table every time it changes. This option is often used for specific situations like redefining the category of a product, for example.

Since SCD 1 does not follow the evolution of the dimensions and SCD 3 would require many extra columns in situations where more than one attribute can vary, we will be working with SCD 2. It is the one that fits our project the most because it allows us to have properly registered every version of every team and player during the five different seasons that we are analyzing.

SCD 2 requires using **surrogate keys**. A surrogate key is an attribute that acts as a primary key for a dimension, which allows us to have different versions of a same element. We will also store a starting date and an end date for each version of every player and team that has more than one entry in the table.

Taking this into account, the following attributes were created in those dimensions that change with SCD 2:

- start date: indicates the date at which the correspondent version started to exist.
- end date: indicates the date at which the correspondent version stopped being the current one.
- version: indicates the number of version that the correspondent one is. If it it the first version of a player or team, it will have value 1, if it is the second version it will have value 2, and so on.

Also, it is important to clarify that we will use the name of a player as its surrogate key, so we will have no need to create a specific one.

6.2 Logical design

Once the DFMs are completed, it is time to use them to start building our warehouse. This process starts by creating **relational models**, which will be the guide that will be followed in order to correctly implement everything we need in our database. A relational model represents how data is stored in relational databases. A relational database consists of a collection of tables, each of which is assigned a unique name. In this project we will have two relational models, one for each fact that we will be working with.

As we can see in figures 6.4 and 6.5 the dimensions that change over time have the attributes correspondent to SCD 2, previously explained in subsection 6.1.4.

6.3 Physical design

Lastly, we have to build our physical design. It consists on our tables constructed in our specific database manager. As it was introduced in section 3.2, we will be working with PostgreSQL for this part of the project. The interface used to operate it on our device is called pgAdmin 4. It is a very user-friendly interface that will allow us to create our database and all its tables, with the specifications we require.

For this project, we will create a database called "TFG", which will include 8 tables: six for our dimensions and two for our facts. For each table, we will create its attributes (specifying their type) and we will also be able to handle their foreign keys if they have any (this will be specially helpful for the fact tables).

Here is a list of the tables, including their specific attributes:

- Teams: includes all the teams with their attributes. We can see all its attributes in detail in figure 6.6.

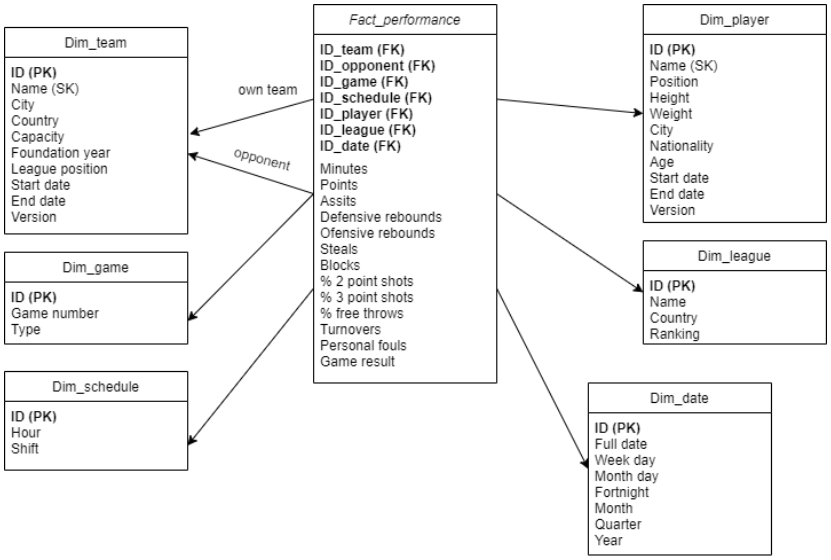


Figure 6.4: Relational model for the performance fact

- Players: includes all the players with their attributes. We can see all its attributes in detail in figure 6.7.
- Leagues: includes all the leagues with their attributes. We can see all its attributes in detail in figure 6.8.
- Games: includes all the games with their attributes. We can see all its attributes in detail in figure 6.9.
- Schedules: includes all the schedules with their attributes. We can see all its attributes in detail in figure 6.10.
- Dates: includes all the dates with their attributes. We can see all its attributes in detail in figure 6.11.
- Performances: includes all the performances with their attributes. We can see all its attributes in detail in figures 6.12 and 6.13, and its foreign keys specifications in figure 6.14.
- Transactions: includes all the transactions with their attributes. We can see all its attributes in detail in figure 6.15, and its foreign keys specifications in figure 6.16.

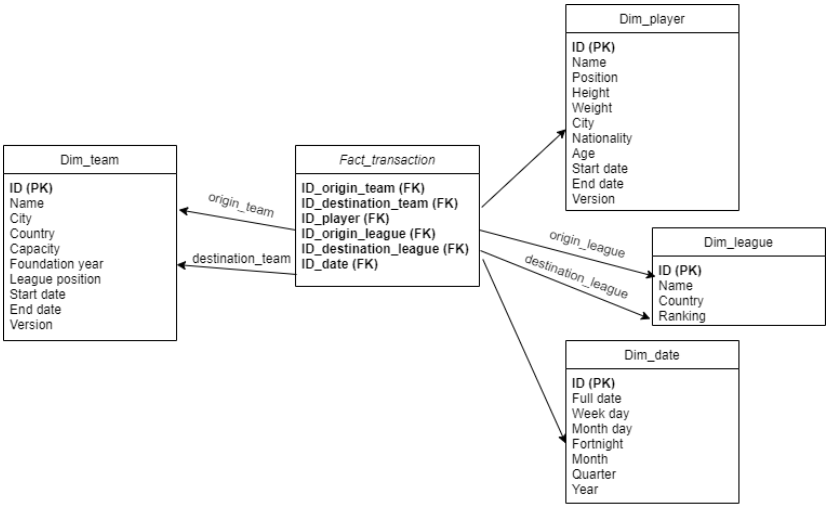


Figure 6.5: Relational model for the transaction fact

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
✎ 🗑	id_equipo	integer			●	●
✎ 🗑	nombre	text			○	○
✎ 🗑	pais	text			○	○
✎ 🗑	ciudad	text			○	○
✎ 🗑	año de fundacion	integer			○	○
✎ 🗑	capacidad	integer			○	○
✎ 🗑	clasificacion	integer			○	○
✎ 🗑	fecha_inicio	timestamp with time zone			●	○
✎ 🗑	fecha_fin	timestamp with time zone			●	○
✎ 🗑	version	integer			●	○

Figure 6.6: Teams table attributes

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
✎ 🗑	id_jugador	integer			●	●
✎ 🗑	nombre	text			○	○
✎ 🗑	posicion	text			○	○
✎ 🗑	altura	double precision			○	○
✎ 🗑	peso	integer			○	○
✎ 🗑	ciudad	text			○	○
✎ 🗑	nacionalidad	text			○	○
✎ 🗑	edad	integer			○	○
✎ 🗑	fecha_inicio	timestamp with time zone			●	○
✎ 🗑	fecha_fin	timestamp with time zone			●	○
✎ 🗑	version	integer			●	○

Figure 6.7: Players table attributes

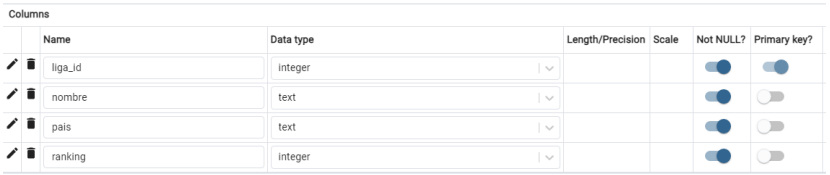


Figure 6.8: Leagues table attributes

Figure 6.9: Games table attributes

Figure 6.10: Schedules table attributes

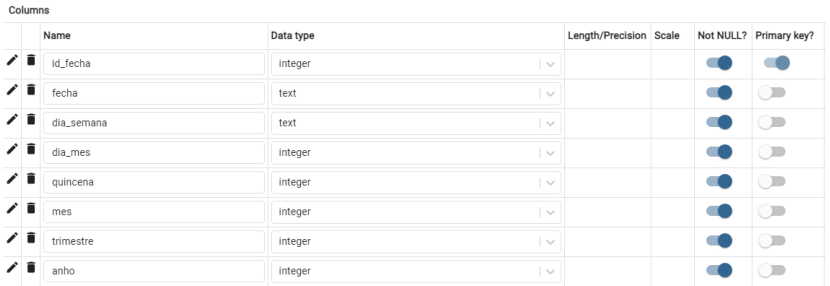


Figure 6.11: Dates table attributes

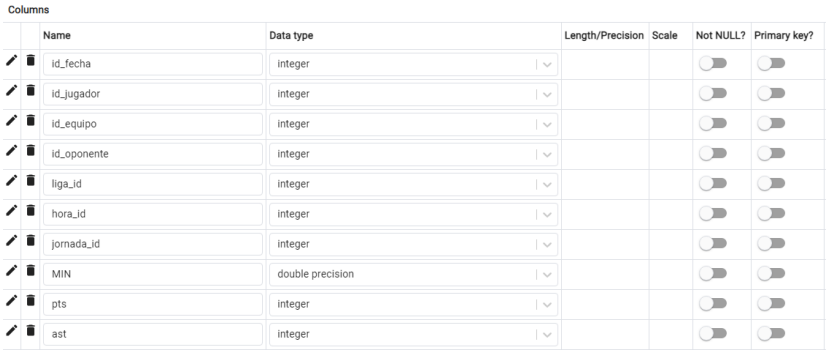


Figure 6.12: Performances table attributes, part 1

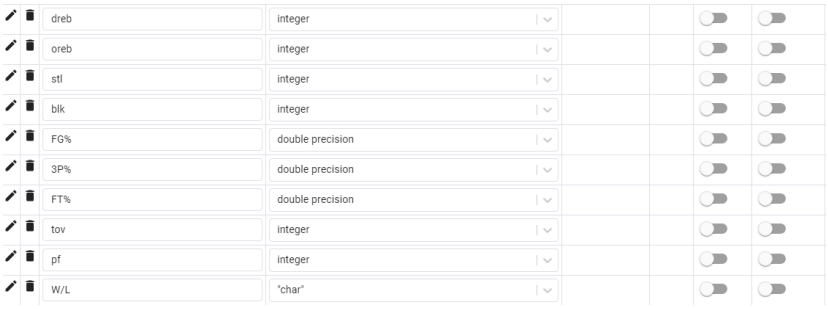


Figure 6.13: Performances table attributes, part 2

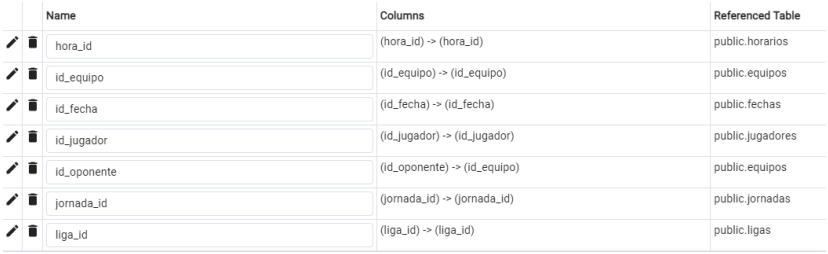


Figure 6.14: Performances table foreign keys

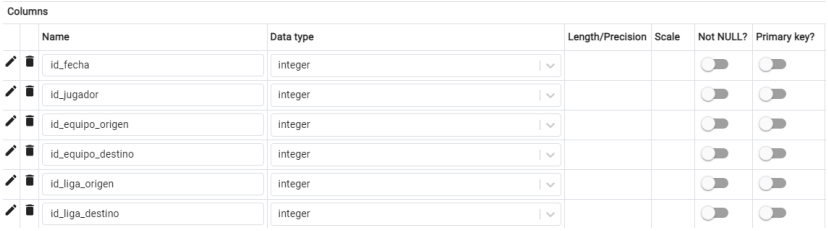


Figure 6.15: Transactions table attributes

Figure 6.16: Transactions table foreign keys

Chapter 7

ETL process

At this stage of the project, it is time to start the ETL process. This will be the part of the project that will consume more time and resources. It will be done using Pentaho Data Integration, explained at section 3.1.

As it was previously explained, the ETL process is divided in three phases: Extraction, Transformation and Load. Now, we will explain each one of them in detail.

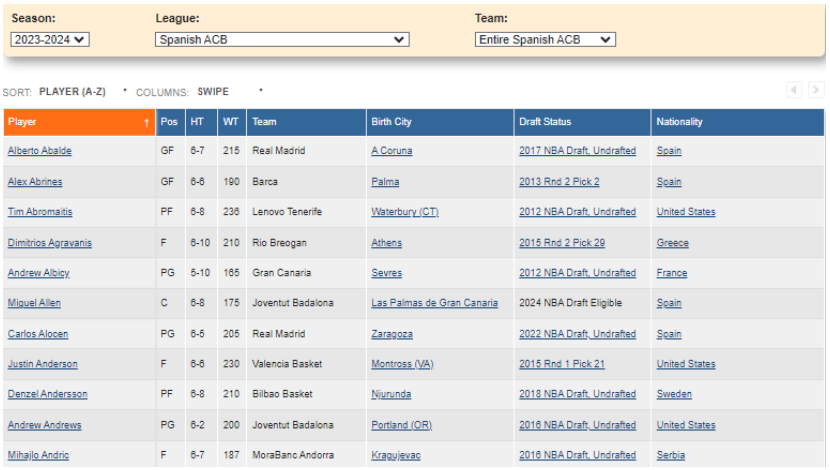
7.1 Extraction

For this first phase of the ETL process, we will extract data directly from the internet using web scraping, explained in section 2.3. The first step consisted on finding websites that gave us all the information we required. The lower the number of websites we needed to get all we wanted from, the easier it would be for us to extract the data later. With this purpose, we started a profound search of basketball web pages, until we found one that had almost everything we needed. This website is https://basketball.realgm.com/international/.

It is a very complete website that has a lot of information about all the basketball leagues. It is interesting for us because it also has registers of previous years, which means that this website could allow us to find everything we needed from many of its sections. It also facilitated filtering your search by season and league, making it easier for you to find specific information. The images 7.1 , 7.2 and 7.3 show us a few examples of how the data appears in this website. Even if this website was very complete and gave us most of what we needed to build our database, we had to complete this information with some data extracted from Wikipedia [1]. For example, the league position of every team during every season.

Now that we have found the website we will obtain most of our data from, it is time to start scraping it. One of the most popular techniques to do it is using python codes, specifically the BeautifulSoup library, previously explained in section 3.4. This will be the procedure for

[1] https://wikipedia.com/

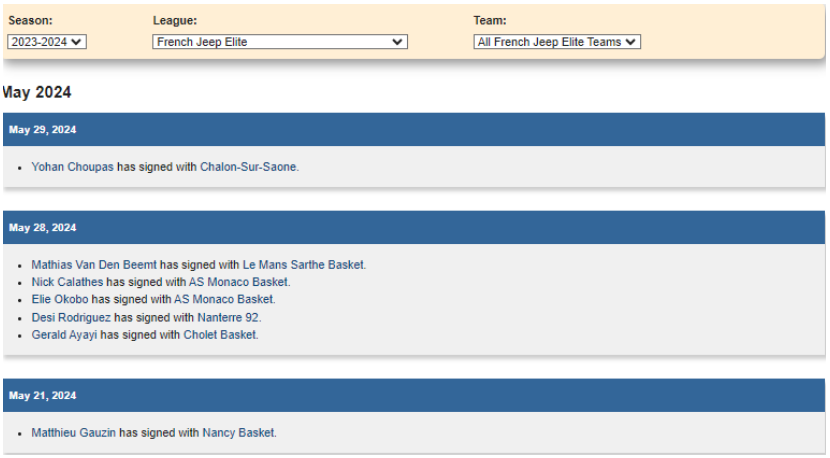


Season: 2023-2024 League: Spanish ACB Team: Entire Spanish ACB

SORT: PLAYER (A-Z) COLUMNS: SWIPE

Player	Pos	HT	WT	Team	Birth City	Draft Status	Nationality
Alberto Abalde	GF	6-7	215	Real Madrid	A Coruna	2017 NBA Draft, Undrafted	Spain
Alex Abrines	GF	6-6	190	Barca	Palma	2013 Rnd 2 Pick 2	Spain
Tim Abromaitis	PF	6-8	238	Lenovo Tenerife	Waterbury (CT)	2012 NBA Draft, Undrafted	United States
Dimitrios Agravanis	F	6-10	210	Rio Breogan	Athens	2015 Rnd 2 Pick 29	Greece
Andrew Albicy	PG	5-10	165	Gran Canaria	Sevres	2012 NBA Draft, Undrafted	France
Miquel Alleń	C	6-8	175	Joventut Badalona	Las Palmas de Gran Canaria	2024 NBA Draft Eligible	Spain
Carlos Alocen	PG	6-5	205	Real Madrid	Zaragoza	2022 NBA Draft, Undrafted	Spain
Justin Anderson	F	6-6	230	Valencia Basket	Montross (VA)	2015 Rnd 1 Pick 21	United States
Denzel Andersson	PF	6-8	210	Bilbao Basket	Njurunda	2018 NBA Draft, Undrafted	Sweden
Andrew Andrews	PG	6-2	200	Joventut Badalona	Portland (OR)	2016 NBA Draft, Undrafted	United States
Mihajlo Andric	F	6-7	187	MoraBanc Andorra	Kragujevac	2016 NBA Draft, Undrafted	Serbia

Figure 7.1: 2023-2024 Spanish ACB players list from the website



Season: 2023-2024 League: French Jeep Elite Team: All French Jeep Elite Teams

May 2024

May 29, 2024

- Yohan Choupas has signed with Chalon-Sur-Saone.

May 28, 2024

- Mathias Van Den Beemt has signed with Le Mans Sarthe Basket.
- Nick Calathes has signed with AS Monaco Basket.
- Elie Okobo has signed with AS Monaco Basket.
- Desi Rodriguez has signed with Nanterre 92.
- Gerald Ayayi has signed with Cholet Basket.

May 21, 2024

- Matthieu Gauzin has signed with Nancy Basket.

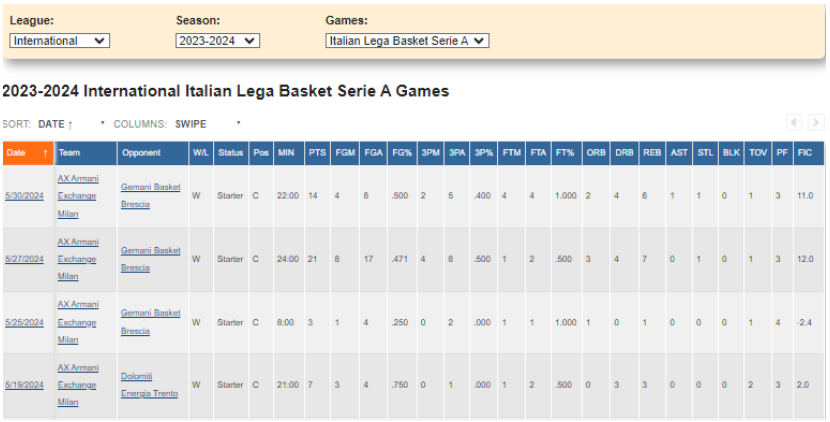
Figure 7.2: 2023-2024 French Jeep Elite transactions list from the website

this project.

For each fact and dimension, a python code was developed. Each code took the links with the raw data from the website and turned them into a CSV file (for this, we use the pandas library) that would later be used to insert the gathered data in the database. We have an example of how these codes look like in section A.1 of appendix A.

It is also interesting to point out that maybe the website table has some rows that are not of our interest. That is the case with the 'Draft Status' row from the players table (image 7.1). If we have a column in the original table that we do not need in our CSV output, we can just skip it at the beginning of our code. This way, it will be ignored and we will not have to deal with it later. Once the code is fully developed and correctly executed, image 7.4 would be the resultant CSV file.

We need to do this same process of developing a Python code that receives the website



League: International | Season: 2023-2024 | Games: Italian Lega Basket Serie A

2023-2024 International Italian Lega Basket Serie A Games

SORT: DATE ↑ COLUMNS: SWIPE

Date ↑	Team	Opponent	W/L	Status	Pos	MIN	PTS	FGM	FGA	FG%	3PM	3PA	3P%	FTM	FTA	FT%	ORB	DRB	REB	AST	STL	BLK	TOV	PF	FIC
5/30/2024	AX Armani Exchange Milan	Germani Basket Brescia	W	Starter	C	22:00	14	4	8	.500	2	5	.400	4	4	1.000	2	4	6	1	1	0	1	3	11.0
5/27/2024	AX Armani Exchange Milan	Germani Basket Brescia	W	Starter	C	24:00	21	8	17	.471	4	8	.500	1	2	.500	3	4	7	0	1	0	1	3	12.0
5/25/2024	AX Armani Exchange Milan	Germani Basket Brescia	W	Starter	C	8:00	3	1	4	.250	0	2	.000	1	1	1.000	1	0	1	0	0	0	1	4	-2.4
5/19/2024	AX Armani Exchange Milan	Dolomiti Energia Trento	W	Starter	C	21:00	7	3	4	.750	0	1	.000	1	2	.500	0	3	3	0	0	0	2	3	2.0

Figure 7.3: 2023-2024 performances list of an Italian Lega Basket Serie A player from the website

	A	B	C	D	E	F	G
1	nombre	posicion	altura	peso	ciudad	nacionalidad	edad
2	Bashir Ahmed	SF	6.7	210	New York (NY)	United States	26
3	Byron Allen	C	6.7	275	Leland (MS)	United States	38
4	Braian Angola-Rodas	F	6.6	195	Villanueva	Colombia	26
5	Dominic Artis	G	6.3	190	Oakland (CA)	United StatesKosovo	26
6	Luka Asceric	SG	6.7	196	Sankt Polten	AustriaSerbia	23
7	Edin Atic	GF	6.7	190	Bugojno	Bosnia and Herzegovina	23
8	Roko Badzim	SG	6.6	190	Sibenik	Croatia	22
9	Jure Balazic	SF	6.8	225	Novo Mesto	Slovenia	39
10	Scott Bamforth	SG	6.2	190	Albuquerque (NM)	United StatesKosovo	30
11	Nejc Baric	PG	6.0		Trbovlje	Slovenia	23
12	Billy Baron	SG	6.2	195	East Greenwich (RI)	United States	29
13	Mirza Begic	C	7.3	265	Bijeljina	Bosnia and HerzegovinaSlovenia	34
14	James Bell	SF	6.6	220	Union (NJ)	United StatesItaly	28
15	Josip Bilinovac	SF	6.4	198	Mostar	Bosnia and HerzegovinaCroatia	29
16	Stefan Bircevic	PF	6.11	210	Lazarevac	Serbia	30
17	Milko Bjelica	PF	6.7	225	Belgrade	SerbiaMontenegro	35
18	Jaka Blazic	SF	6.5	212	Jesenice	Slovenia	29
19	Ryan Boatright	PG	5.11	175	Aurora (IL)	United StatesArmenia	27
20	Matej Bosnjak	C	6.9			Croatia	18
21	Luka Bozic	PG	6.5	195	Bjelovar	Croatia	24
22	Duje Brala	G	6.5	180	Zadar	Croatia	17

Figure 7.4: CSV file obtained after executing the python code scraping the players links

links and gives us a CSV file with the correspondent attributes for each one of our dimensions and facts.

However, for some dimensions such as the date or the games, we will not need to extract data from any website. In this cases we can just develop a code that gives us what we need with no necessity of website links. We can see an example of this situation in section A.2 of appendix A. All our codes will be executed by Pentaho Data Integration. This way, we can open Pentaho with an empty database and leave it with a full one, with no need of using any other tool. We will have a job that runs this code and then uses it as an starting point for each dimension or fact transformation process. Image 7.5 shows us how this is done.

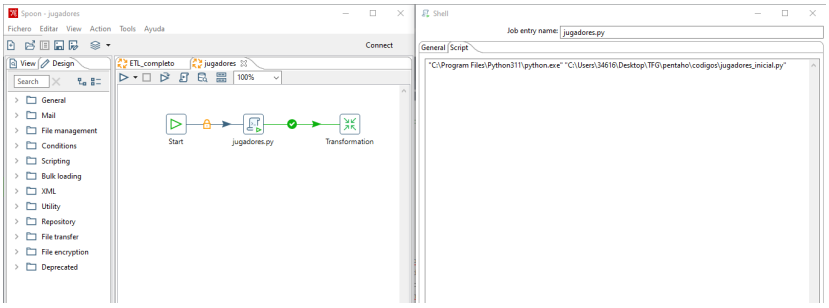


Figure 7.5: Python code run with Pentaho

7.2 Transformation

The transformation phase consists on ensuring that your data follows certain business rules and restrictions to be in your data warehouse. In order to do this, raw data needs to go through a conversion process. This process is done with Pentaho, building a transformation for each table we will be storing in our system. We can divide our transformations in three categories.

7.2.1 Dimensions that do not change overtime

This is the most simple scenario. It includes the following dimensions:

- League.
- Date.
- Timetable.
- Game.

All of these dimensions are here because they do not change overtime, so we will not need to apply SCD 2 in these cases. Instead, we will have simple transformations that do not require any change management.

Now we will see the entire process followed by these dimensions, using the leagues table as example.

- Firstly, we create a job that runs the required python code. Once it is finished and the CSV file is generated, we run a transformation from the same job. This is shown in image 7.6.
- Secondly, we perform the transformation. Image 7.7 shows it. It starts by picking up the raw data with the "CSV input file ligas" step, detailed in image 7.8. Then we add the primary key to each league. This is done by step "Añadir secuencia", that just adds

Figure 7.6: Pentaho job for the leagues table

a numeric id for each row. It is shown in image 7.9. Finally, we load our data with the added primary keys into our system. This is done with he "Salida Tabla ligas" step, that simply ensures that every column of our raw data corresponds to the appropriate column in our data warehouse. Image 7.10 shows this.

Figure 7.7: Pentaho transformation for the leagues table

By following this exact same process for each of the dimensions mentioned earlier, we successfully store them in our system according to all the existing rules and restrictions.

7.2.2 Dimensions that change overtime

This second scenario involves those dimensions where we will need to apply SCD 2. These are the following:

- Player.
- Team.

These two dimensions will require a more complex transformation than the previous ones, since we need to make sure that we are properly managing all the different versions of teams and players generated over the years. Once again, we must start with a Pentaho job that runs the Python script to obtain the raw data, and then performs the corresponding transformation. This is shown in image 7.11, using the players table as example. The transformation performed in these dimensions is shown in image 7.12. Here is an explanation of this transformation step by step:

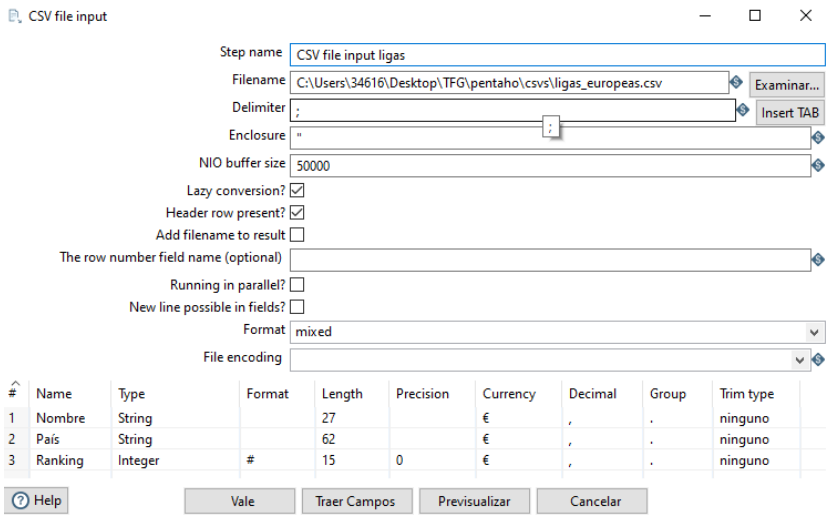


Figure 7.8: CSV input file ligas

- We load in parallel two CSV files, an initial one with players of the first season (“CSV file input jugadores1”) and a second one with players of the other four seasons (“CSV file input jugadores2”). Images 7.13 and 7.14 show this process.
- We assign a primary key to each row of our tables. This is done with steps “Añadir secuencia” and “Añadir secuencia 2”, which add a numeric primary key to each element. It has the exact same behaviour as the one present in the league transformation, shown in image 7.9.
- The “Selecciona/Renombra valores” and “Selecciona/Renombra valores 2” are steps that simply reorder our tables. We place the id first since this is often recommended in these kind of projects. Image 7.15 shows this step, being both of them exactly the same.
- At this point we can bind our two tables. We do this with the ’Append streams’ step, that simply puts one after the other in a common table. It just requires the two input tables, as shown in image 7.16.
- Now that we have one table with all the players we need, we can perform the SCD 2 technique. In Pentaho, this is done with a “Búsqueda/actualización en dimensión” step. As we see in image 7.17 this step lets you pick your primary key (id) and surrogate key (name), and creates the version, starting date and end date fields. It also lets you assign the rest of the fields of your table to its corresponding one in the database, as shown in image 7.18.

Following this process for the players and teams tables, we will have them inserted in our system while properly handling the changes they suffer overtime.

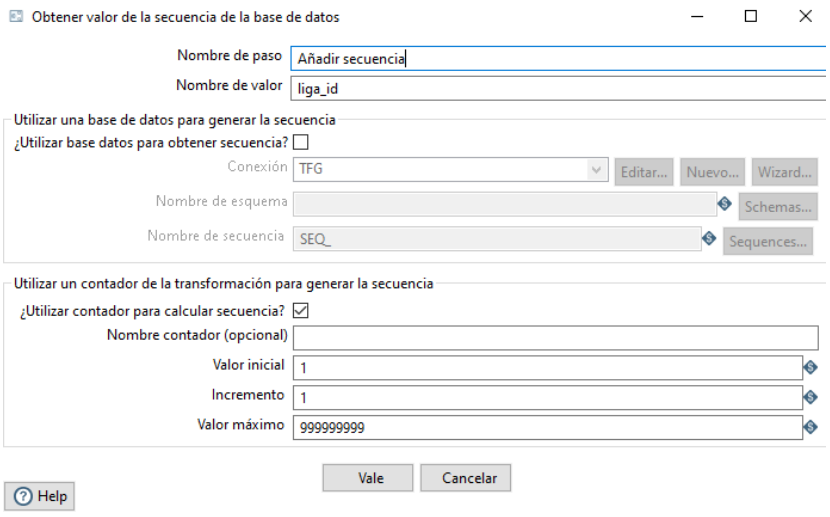


Figure 7.9: Add sequence

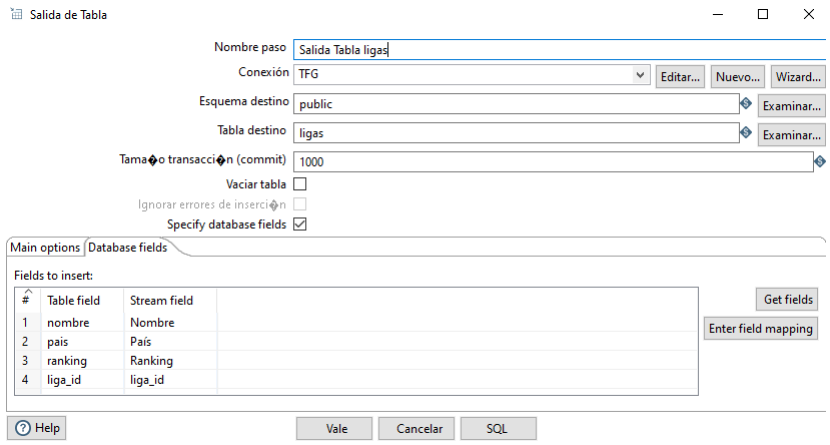


Figure 7.10: Table output for the league table

7.2.3 Facts

Now that every dimension has been inserted into our data warehouse, we can proceed to add the facts. This had to be the last step of the transformation process because these fact tables need to obtain their foreign keys from the dimension ones, so the dimensions needed to be first. With this being said, we will explain the transformations for the two facts that conform this project:

- Performance.
- Transaction.

We will be using images from the performance fact to show how this was done. Following the same structure as the dimensions, we must start by running the Python code to obtain

Figure 7.11: Pentaho job for the players table

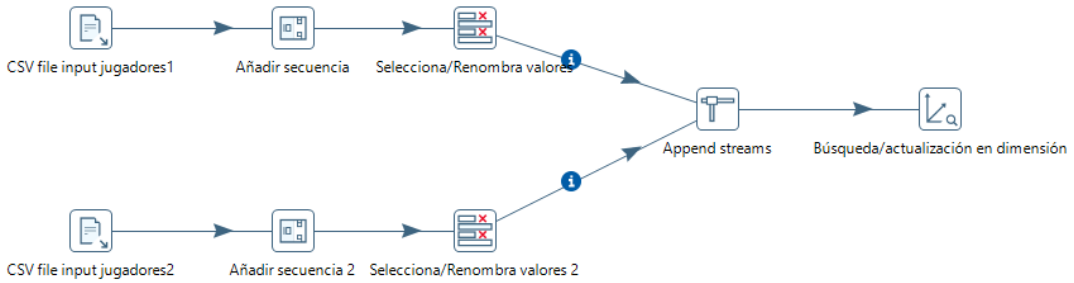


Figure 7.12: Pentaho transformation for the players table

the CSV files with the raw data. In the Pentaho job of image 7.19 we can see how we run the code, and we connect it to another job afterwards.

This second job will be an intermediate job where we will just join the five different transformations we have, one for each season of our domain. Image 7.20 shows it. Now, we will see how these transformations work. Image 7.21 shows us how they look like.

We will explain it step by step:

- We start by picking up the raw data from the CSV file. This is done by step 'CSV file input', shown in detail in image 7.22.
- Now we need to make sure that the foreign keys are correctly assigned. We will start by switching the players and teams names for their corresponding id, assigned in section 7.2.2. To do this, we have two "Entrada tabla" steps and three "Búsqueda en flujo de datos" steps. The first two will be for correctly selecting the corresponding players from the database[2] as shown in images 7.23 and 7.24. The three remaining will be used to switch the name of players, teams and opponents by their corresponding id. Image 7.25 shows these three steps.
- Once this is done, we find four different "Búsqueda en base de datos" steps. This will have a similar behaviour to the previous ones, since they will switch the dates, timetables, leagues and games by their corresponding ids. As we can see in image 7.26, where

[2] An auxiliary column "temporada" was created to facilitate the search of teams and players by season.

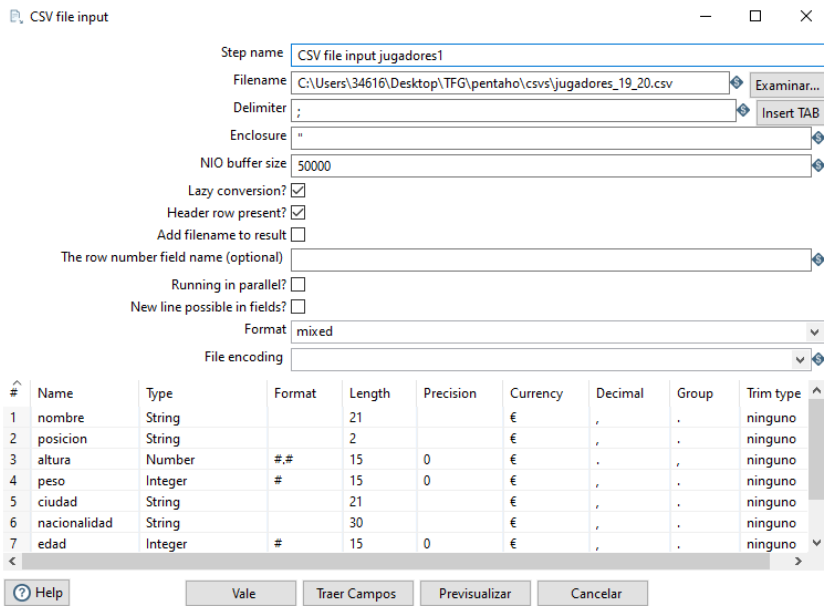


Figure 7.13: CSV input file for 19-20 season players

we show the example of the dates table, this steps finds the selected field, assigns it to its matching database field and returns its id. This same process is exactly the same for each remaining table in each remaining step.

- Finally, we find a "Selecciona/Renombra valores" step that is just used to reorder our columns in the desired order (image 7.27) and a "Salida Tabla" step, where we load our table in the data warehouse. As we see in image , we must assign each column to its corresponding one.

These steps will introduce the 19-20 performances with all the foreign keys correctly assigned. It must be repeated for each remaining season. Once it is done, we create the already mentioned job in image 7.20, which will be the one giving us the entire performances table. As expected, the procedure for the transactions table would be exactly the same.

7.3 Load

This will be the last step of our ETL process. It consists on loading the output of the previous steps into our data warehouse. If everything was done correctly, all the data types and keys restrictions set at the physical design in section 6.3 should not be violated, and the transformed data will fit our database without any problem.

To load all the data in our system, we created a 'Master Job', which is a job that reunites all the jobs of every dimension and fact into an only one. This way, once we execute this job,

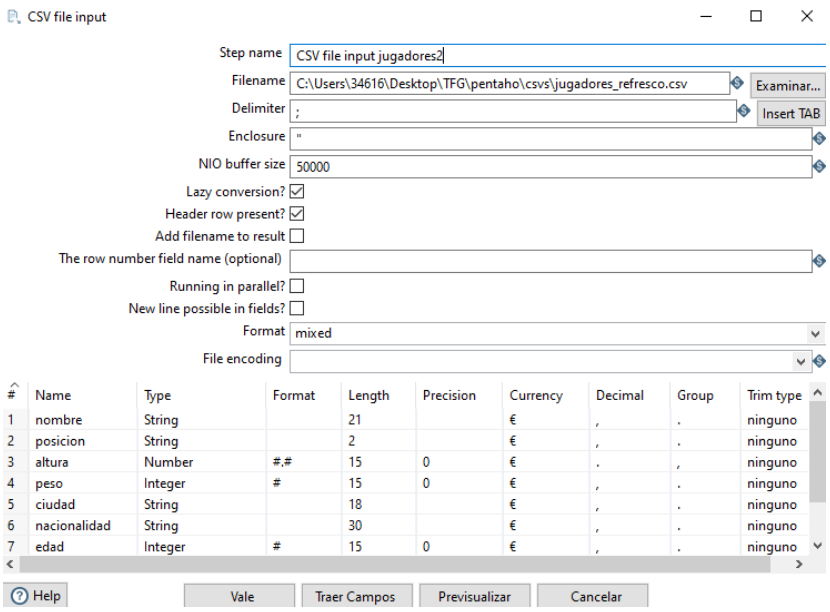


Figure 7.14: CSV input file for last four seasons players

the entire data warehouse will be filled with the data that each specific job sets for each table. Image 7.29 shows the master job.

Now that the data warehouse is filled, it is time to check if everything went as planned. In order to do this, we can go back to our Postgres (3.2) interface, pgAdmin 4. Once there, we will check the results table by table:

- Dates table in image 7.30.
- Timetables table in image 7.31.
- Games table in image 7.32.
- Leagues table in image 7.33.
- Players table in images 7.34 and 7.35. In this case, we show an specific player with many versions to see how the SCD 2 technique was applied[3].
- Teams table in image 7.36. Again, we show a team with its different versions to see the SCD 2 technique result.
- Performances table in images 7.37 and 7.38.
- Transactions table in image 7.39.

[3] The start date and end date are automatically set by Pentaho when the transformations are run, in a real case there would be the start date and end date of each season

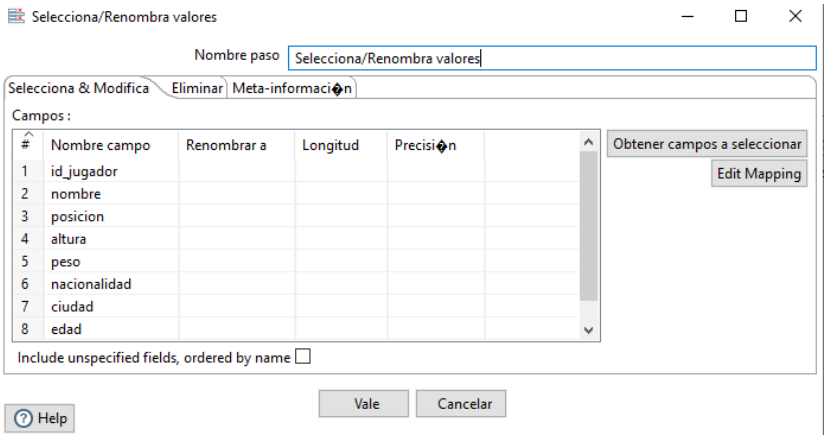


Figure 7.15: Step to reorder our columns to put the id first

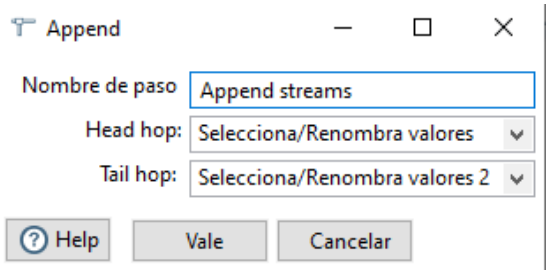


Figure 7.16: Append streams step

If there was any problem with a data type or a primary or foreign key, pgAdmin 4 would have warned us about it by raising an error and we would not be allowed to insert the data. Since this did not happen, everything worked correctly.

This sets the end of the ETL process, meaning that we have our data warehouse built and ready to be exploited.

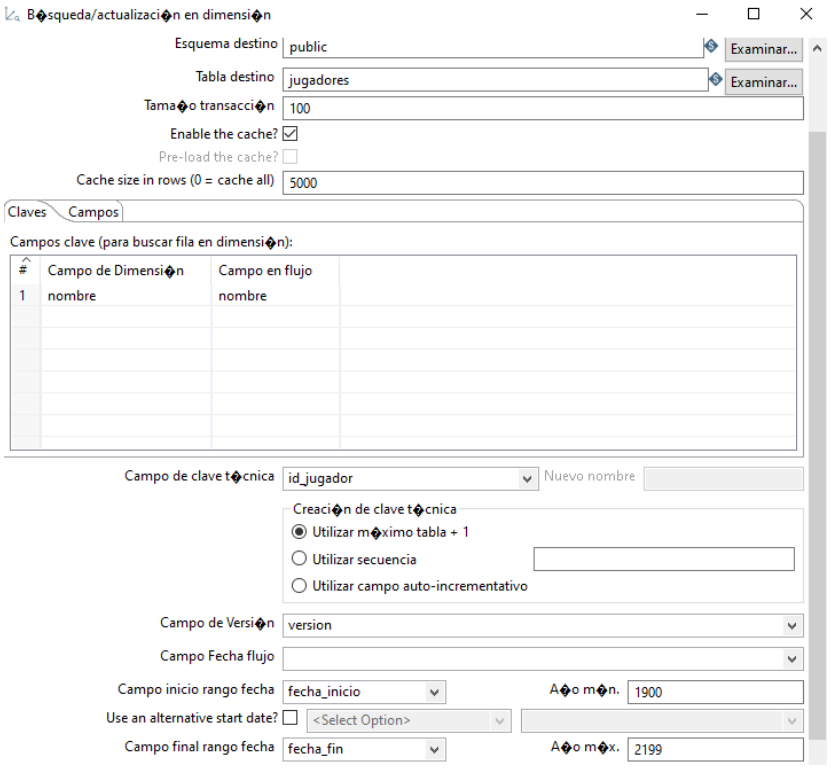


Figure 7.17: Dimension lookup/update part 1

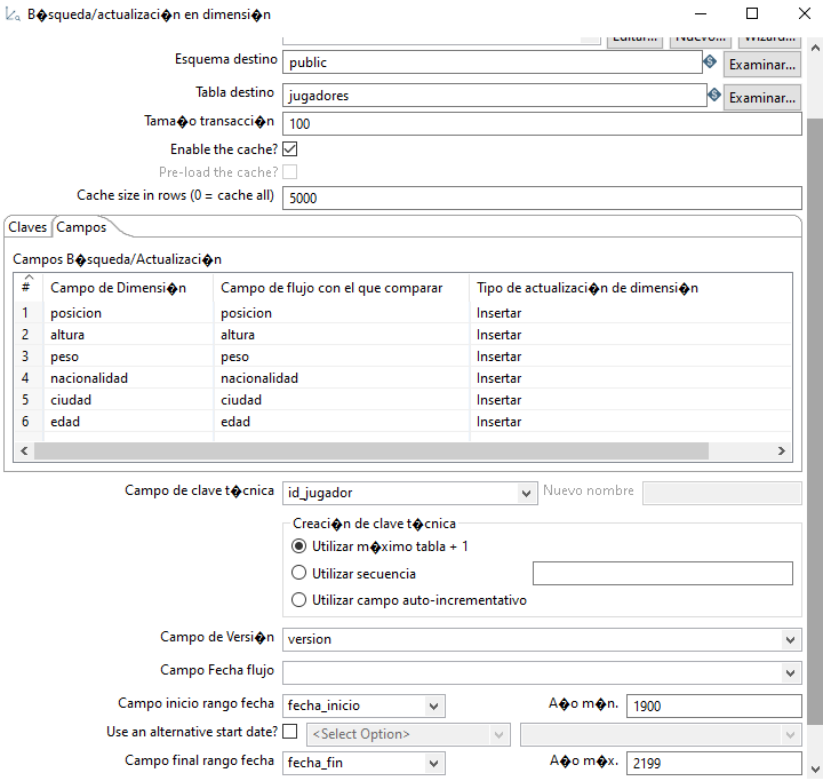


Figure 7.18: Dimension lookup/update part 2

Figure 7.19: Pentaho job for the performances table

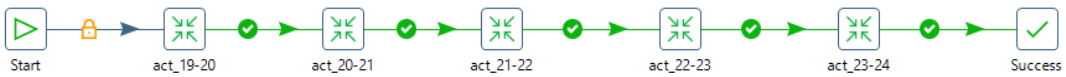


Figure 7.20: Pentaho job to join the performance transformations of all the seasons

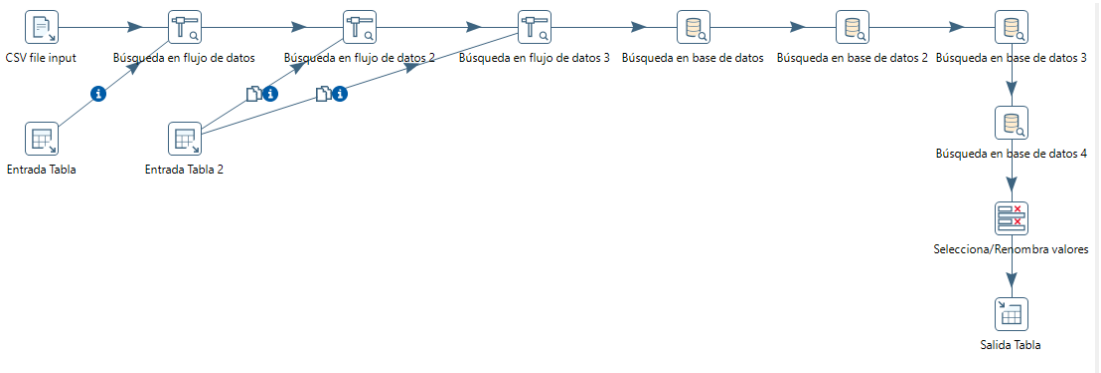
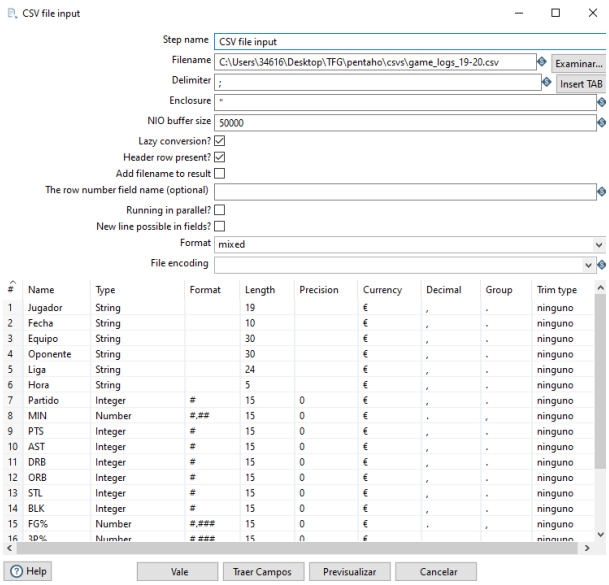


Figure 7.21: Pentaho transformation for a season of players performances



#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	Jugador	String		19		€	,	.	ninguno
2	Fecha	String		10		€	,	.	ninguno
3	Equipo	String		30		€	,	.	ninguno
4	Oponente	String		30		€	,	.	ninguno
5	Liga	String		24		€	,	.	ninguno
6	Hora	String		5		€	,	.	ninguno
7	Partido	Integer	#	15	0	€	,	.	ninguno
8	MIN	Number	#,##	15	0	€	.	,	ninguno
9	PTS	Integer	#	15	0	€	,	.	ninguno
10	AST	Integer	#	15	0	€	,	.	ninguno
11	DRB	Integer	#	15	0	€	,	.	ninguno
12	ORB	Integer	#	15	0	€	,	.	ninguno
13	STL	Integer	#	15	0	€	,	.	ninguno
14	BLK	Integer	#	15	0	€	,	.	ninguno
15	FG%	Number	#.###	15	0	€	.	,	ninguno
16	3P%	Number	#.###	15	0	€			ninguno

Figure 7.22: CSV input file step for the 19-20 player performances

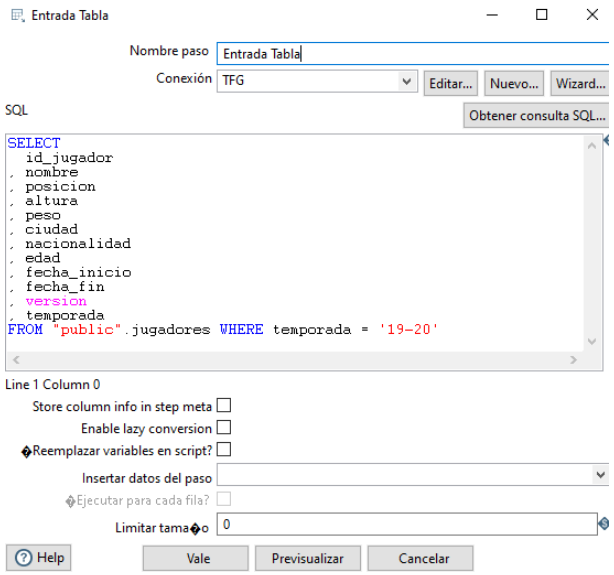


Figure 7.23: Table input for 19-20 players

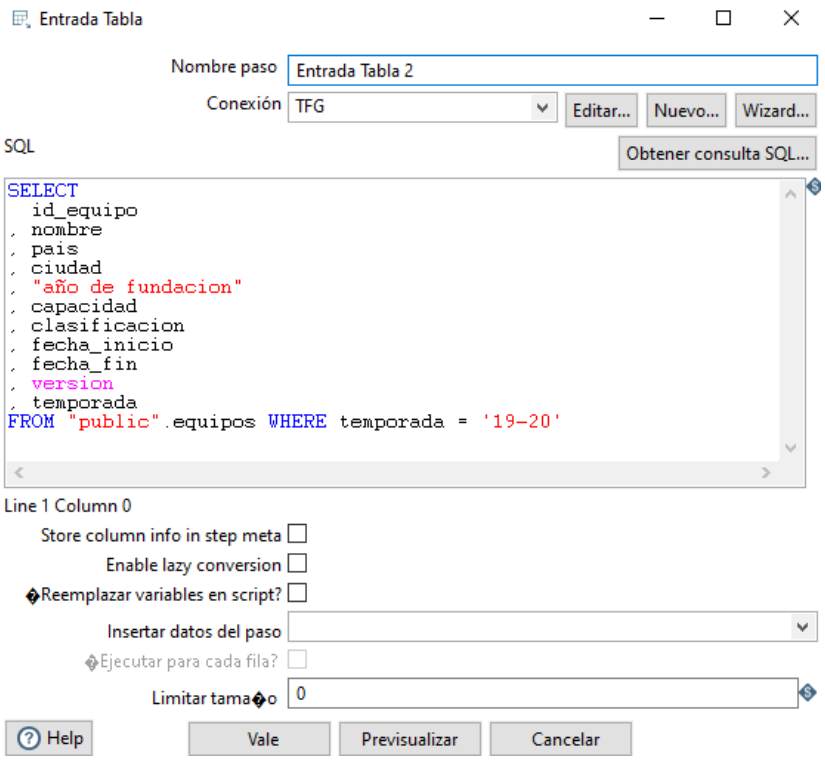


Figure 7.24: Table input for 19-20 teams

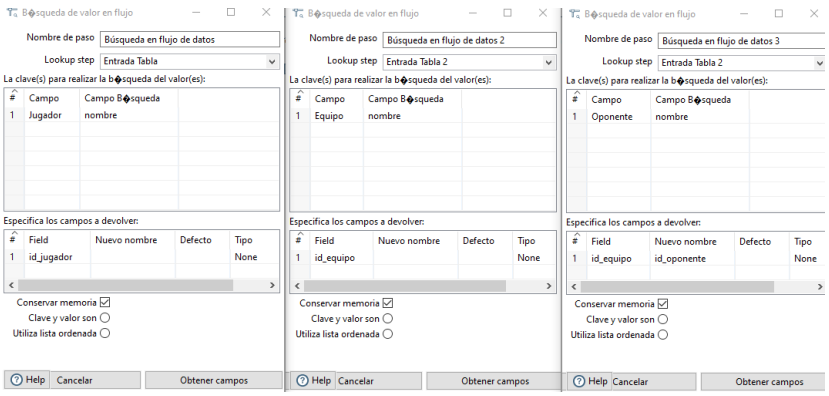


Figure 7.25: Steps to switch players, teams and opponent by their corresponding id

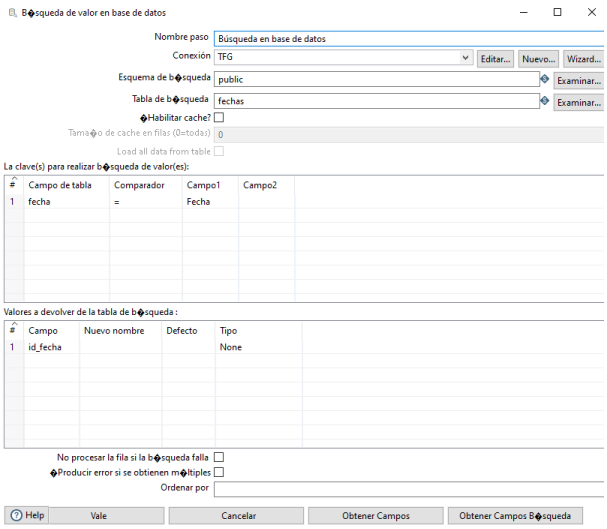


Figure 7.26: Step that switches the date by its id

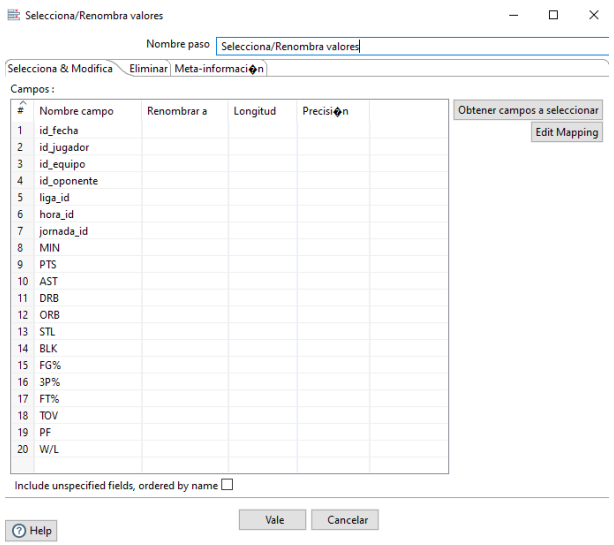


Figure 7.27: Step to reorder the columns in the desired order

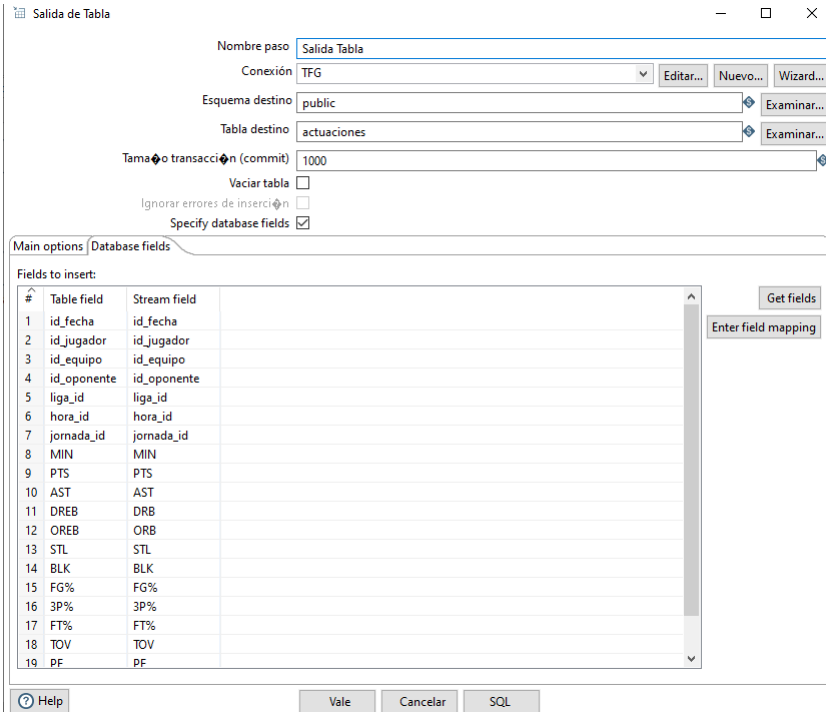


Figure 7.28: Table output for 19-20 performances

Figure 7.29: Master job that fills the entire data warehouse

```
1  SELECT * FROM public.fechas
2  ORDER BY id_fecha ASC
```

Data output | Messages | Notifications

	id_fecha [PK] integer	fecha text	dia_semana text	dia_mes integer	quincena integer	mes integer	trimestre integer	anho integer
1	1	01/01/2019	Tuesday	1	1	1	1	2019
2	2	01/02/2019	Wednesday	2	1	1	1	2019
3	3	01/03/2019	Thursday	3	1	1	1	2019
4	4	01/04/2019	Friday	4	1	1	1	2019
5	5	01/05/2019	Saturday	5	1	1	1	2019
6	6	01/06/2019	Sunday	6	1	1	1	2019
7	7	01/07/2019	Monday	7	1	1	1	2019
8	8	01/08/2019	Tuesday	8	1	1	1	2019
9	9	01/09/2019	Wednesday	9	1	1	1	2019
10	10	01/10/2019	Thursday	10	1	1	1	2019

Figure 7.30: Resultant dates table

```
1  SELECT * FROM public.horarios
2  ORDER BY hora_id ASC
```

Data output | Messages | Notifications

	hora_id [PK] integer	hora text	turno text
1	1	00:00	Noche
2	2	00:15	Noche
3	3	00:30	Noche
4	4	00:45	Noche
5	5	01:00	Noche
6	6	01:15	Noche
7	7	01:30	Noche
8	8	01:45	Noche
9	9	02:00	Noche
10	10	02:15	Noche

Figure 7.31: Resultant timetables table

```
1  SELECT * FROM public.jornadas
2  ORDER BY jornada_id ASC
```

Data output | Messages | Notifications

	jornada_id [PK] integer	jornada integer	tipo text
1	1	1	RS
2	2	2	RS
3	3	3	RS
4	4	4	RS
5	5	5	RS
6	6	6	RS
7	7	7	RS
8	8	8	RS
9	9	9	RS
10	10	10	RS

Figure 7.32: Resultant games table

```
1  SELECT * FROM public.ligas
2  ORDER BY liga_id ASC
```

Data output | Messages | Notifications

	liga_id [PK] integer	nombre text	pais text	ranking integer
1	1	Spanish ACB	Spain, Andorra	1
2	2	Turkish BSL	Turkey	2
3	3	Italian Lega Basket Serie A	Italy	3
4	4	French Jeep Elite	France, Monaco	4
5	5	German BBL	Germany	5
6	6	Adriatic League Liga ABA	Bosnia, Croatia, Serbia, Montenegro, Slovenia, North Macedonia	6
7	7	Greek HEBA A1	Greece	7
8	8	Israeli BSL	Israel	8
9	9	Lithuanian LKL	Lithuania	9
10	10	Polish OBL	Poland	10

Figure 7.33: Resultant leagues table

```
1  SELECT * FROM jugadores WHERE nombre = 'Nikola Mirotic';
```

Data output | Messages | Notifications

	id_jugador [PK] integer	nombre text	posicion text	altura double precision	peso integer	ciudad text
1	12540	Nikola Mi...	F	6.1	250	Podgorica
2	745	Nikola Mi...	F	6.1	250	Podgorica
3	3514	Nikola Mi...	F	6.1	250	Podgorica
4	6199	Nikola Mi...	F	6.1	250	Podgorica
5	8843	Nikola Mi...	F	6.1	250	Podgorica

Figure 7.34: Resultant players table part 1

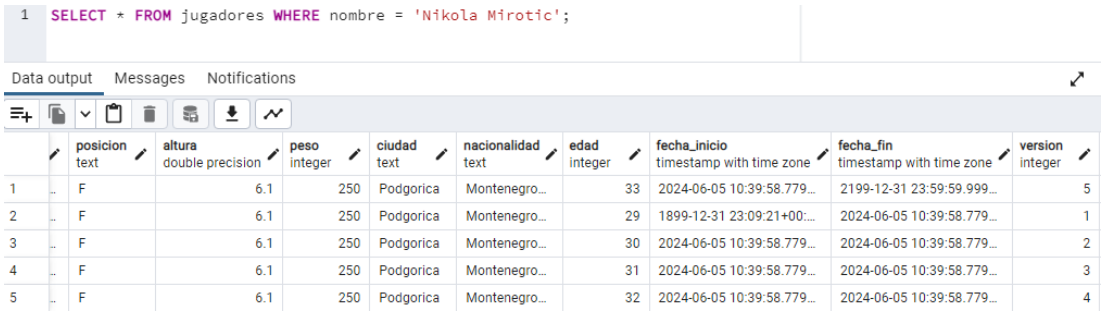


Figure 7.35: Resultant players table part 2

Figure 7.36: Resultant teams table

```
1 SELECT * FROM public.actuaciones
2
```

Data output | Messages | Notifications

	id_fecha integer	id_jugador integer	id_equipo integer	id_oponente integer	liga_id integer	hora_id integer	jornada_id integer	MIN double precision	pts integer	ast integer	dreb integer
1	433	1	30	25	6	79	41	33	21	3	4
2	425	1	30	23	6	79	19	14	8	0	0
3	404	1	30	20	6	49	48	26	11	1	4
4	396	1	30	28	6	75	28	15	5	0	2
5	390	1	30	29	6	69	1	27	5	1	3
6	384	1	30	24	6	81	32	23	6	2	2
7	370	1	30	27	6	81	15	13	4	0	2
8	363	1	30	26	6	69	4	11	1	0	2
9	356	1	30	19	6	81	25	22	17	0	1
10	425	2	30	23	6	79	19	36	8	8	4

Figure 7.37: Resultant performances table part 1

```
1 SELECT * FROM public.actuaciones
2
```

Data output | Messages | Notifications

	dreb integer	oreb integer	stl integer	blk integer	FG% double precision	3P% double precision	FT% double precision	tov integer	pf integer	W/L "char" (1)
1	3	4	0	2	0	1	1	1	0	1	W
2	0	0	2	0	0	0	0	1	1	3	W
3	1	4	2	0	0	1	0	0	1	4	L
4	0	2	1	0	0	0	0	0	1	2	L
5	1	3	2	1	1	0	0	1	0	3	L
6	2	2	2	0	0	0	0	0	1	5	L
7	0	2	2	0	0	0	0	1	1	3	W
8	0	2	1	0	0	0	0	0	1	3	L
9	0	1	0	1	0	1	1	0	2	1	W
10	8	4	2	1	0	0	0	0	1	0	W

Figure 7.38: Resultant performances table part 2

```
1 SELECT * FROM public.transacciones
2
```

Data output | Messages | Notifications

	id_fecha integer	id_jugador integer	id_equipo_origen integer	id_equipo_destino integer	id_liga_origen integer	id_liga_destino integer
1	515	2104	[null]	137	[null]	2
2	509	2546	[null]	146	[null]	2
3	508	1896	[null]	137	[null]	2
4	507	2697	[null]	150	[null]	2
5	504	2594	[null]	148	[null]	2
6	493	2583	141	137	2	2
7	487	2444	135	137	2	2
8	477	2591	136	136	2	2
9	458	2470	136	136	2	2
10	425	2572	12	150	1	2

Figure 7.39: Resultant transactions table

Chapter 8

Data Warehouse exploitation

As we have mentioned plenty of times during the project, data warehouses are not just useful because they store big quantities of information. Their main advantage is the fact that they can be exploited in order to use the information they store to obtain interesting details that can be helpful in different scenarios.

With this purpose, this last part of the project will consist on exploiting the data warehouse we have previously built. This will allow us to answer questions like the ones we raised in section 5.3 and similar ones.

This will be done using Power BI, a tool built by Microsoft explained in section 5.3. Essentially, this chapter of the project will consist on building and explaining a series of graphics and maps that could be helpful to show in an organized way some of the information stored in our data warehouse. Ideally, this could be useful for teams and players in their decision making since basketball is a sport very influenced by statistics nowadays. The obtained results are shown below.

8.1 Brief data warehouse overview

It is interesting to start by showing how large the data warehouse we are working with is. Power BI allows us to easily check this. The cards shown in image 8.1 are a good way to measure this. We can see that we are working with nearly 6,000 players and exactly 200 teams. However, since we are using the SCD 2 technique (detailed in section 6.1.4) for teams and players, we will end up with more than 13,000 versions of players and more that 750 versions of teams. It is also worth mentioning the huge amount of player performances we have analyzed, being more than 230,000 games, and the more than 3,000 transactions stored. To sum up, this first sections comes to show how big of a data warehouse we are working with.

Figure 8.1: Overview of the dimensions of our data warehouse

8.2 Points scored by league

The first question that may come to mind is a very simple one: in which league do players score the most points? The graph shown in image 8.2 shows us the solution.

The first thing that draws our attention from this graph is that there is not a big difference between any of the leagues in terms of points scored. All of the values oscillate between 7 and 9 points. However, there are a few interesting things we can mention:

- The league where players average the most points is the Israeli BSL. This is interesting to mention since it is not a league that the general public would consider a particularly strong one in terms of scoring.
- On the other hand, it is also interesting to see how the two leagues with the lowest points per game averaged by their players are the Greek HEBA A1 and the Spanish ACB. This is something that we would not expect before doing this project, since teams from this leagues tend to perform really well in European competitions, so it would seem logical to think that there would be a lot of points scored in them.
- We also have to mention how 6 of the 10 analyzed league oscillate between 8 and 8.5 points. This comes to show how European leagues tend to have a similar playing style, with no big differences in terms of scoring between them.

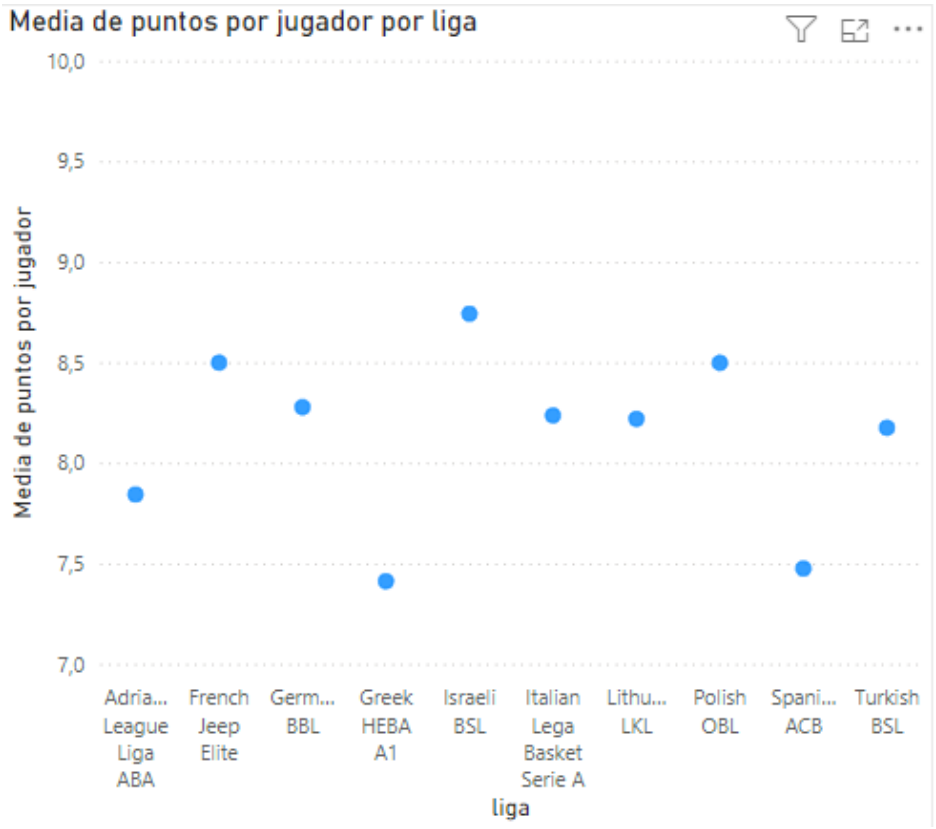


Figure 8.2: Average points per player in each league

8.3 Assists given by position

Another interesting point to analyze is how does the position[1] of a player affect his stats. A good example of this is shown in figure 8.3, where we see how many assist per game are given by each position, ordered from higher to lower. In this case, the results came exactly as expected. This are some things we can observe:

- We can see how the 4 most predominant positions in assists given are PG, G, SG and GF. This is something natural since guards commonly initiate the offense and distribute more passes to the rest of the players.
- After them we find SF, F and PF. This is also something we could expect since forwards do not initiate the offense as much as guards, but they can do it in certain situations.
- Finally, we have FC and C. This makes sense since centers do not initiate the offense, they often focus on finishing attacks and defending, so they do not give a lot of assists.

[1] PG: Point guard, SG: Shooting guard, G: PG/SG, SF: Small Forward, PF: Power forward, F: SF/PG, GF: G/F, C: Center, FC: F/C

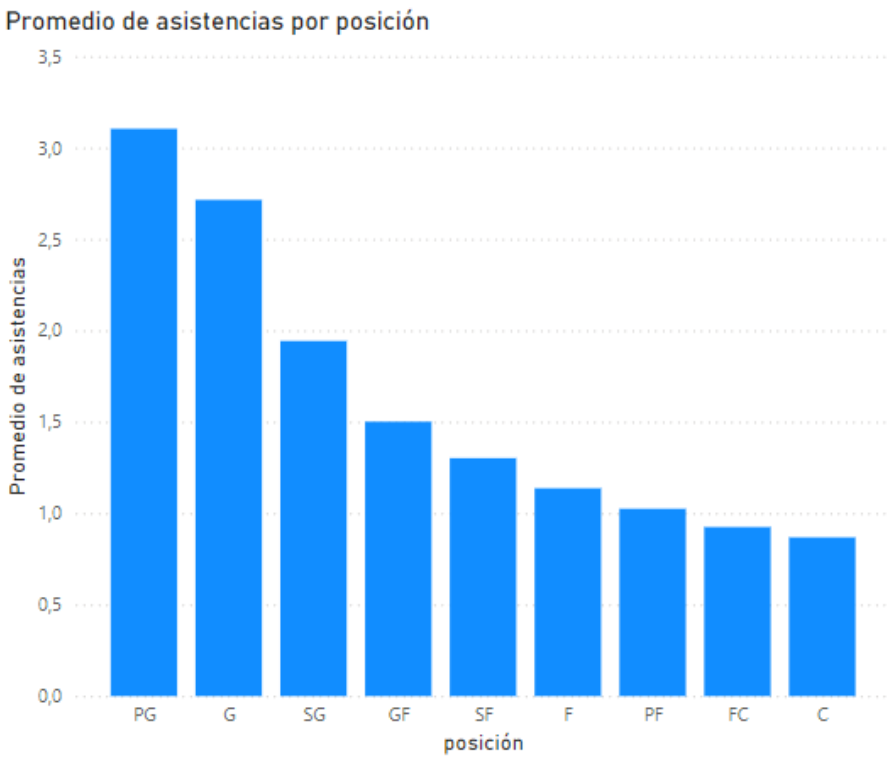


Figure 8.3: Average assist per player in each position

8.4 Blocks made by position

This third section will have a similar structure to the previous one, since we are analyzing the players stats separated by positions once again.

In this case, we will see how many blocks were put by each position. Instead of using averages, this time we will do the addition, so we can see the proportion of the total amount of blocks that each position represents. We will use the ring graph shown in figure 8.4.

Similarly to the previous one, we have the results we expected:

- Centers and forwards dominate this category, since they are more often than not players specialized in the defensive side of the game.
- Guards do not have a lot of influence in this statistic, which also makes sense because they tend to be shorter players that do not focus on defense as much as they do on offense.
- One surprising thing is finding the FC position lower than expected. It is safe to assume that this happens because there are not a lot of players assigned to that position in our system, and since we are doing the addition this greatly affects its position in the graph.

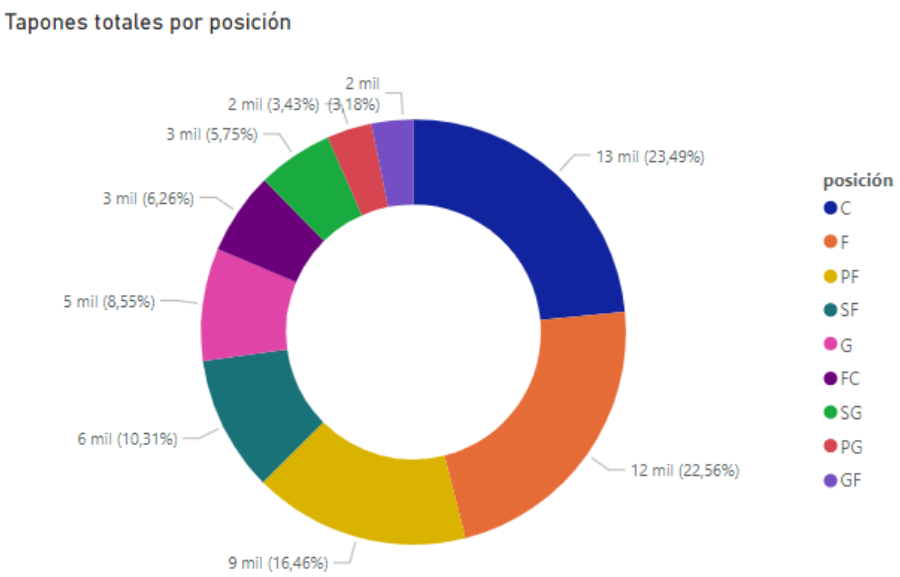


Figure 8.4: Blocks for position from the total amount

8.5 Fouls commited by age

Something we can also take a look at is how does a players age affect his behaviour on the court. An interesting approach to do this is using the number of personal fouls averaged by every age recorded in our data warehouse. Figure 8.5 is a treemap that shows this.

Here are a few interesting things we can comment about it:

- There is not a big predominance of any specific age, which is expected since there are a lot of different values.
- However, as we can appreciate in the left side of the graph, ages in the high 20s and low 30s have higher values than the rest. This may be striking since it seems logical to think that this statistic would be dominated by players of lower ages because they are more inexperienced. That is not the case in this situation.
- Following the previous point, it is interesting to point out how ages from 18 to 21 occupy the bottom right corner of the graph, which is the one that represents the smallest area. Something that must be said is that these players tend to play less minutes in the games, so this stat may be affected because the lower amount of minutes you play means the lower amount of personal fouls you will commit.

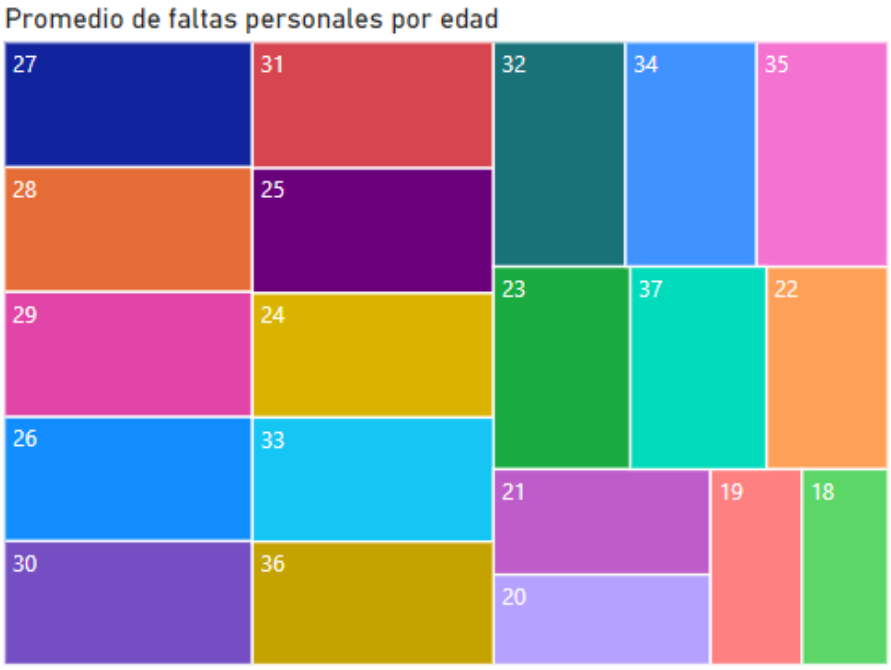


Figure 8.5: Personal fouls committed by age

8.6 Points scored by shift

Another aspect worth considering is whether players' performances are affected by the time at which they play. Since it would be complicated to analyze hour by hour, we have the following shifts:

- Noon: includes games that start between 12 p.m. and 4 p.m. (not included).
- Afternoon: includes games that start between 4 pm. and 8 pm. (not included).
- Night: includes the games that start after 8 pm.

Taking this into account, we will see how many points do players average in each one of these shifts, as shown in figure 8.6. This graph shows us that there is practically no difference in terms of points averaged in each shift. This is quite surprising because we could always expect that the time at which a game is played could affect how players performed. However, that is not the case. It seems that basketball players are not greatly affected by the time at which they play. A reason behind this is that they play their games in closed arenas, so the weather conditions do not affect them in the same way as it does in other sports that are played outside.

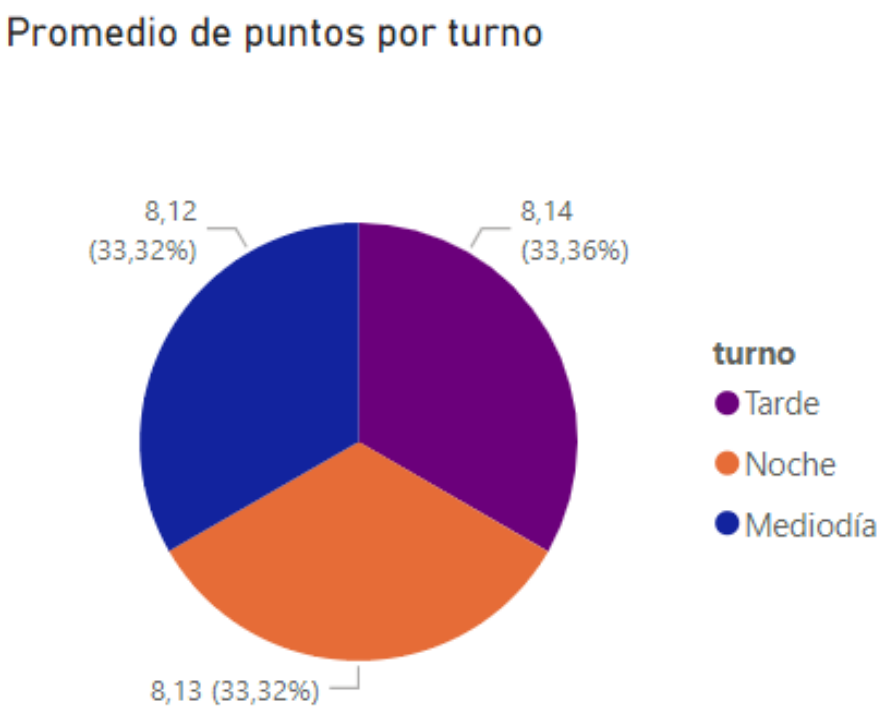


Figure 8.6: Points averaged per shift

8.7 Points scored by nationality

In this section we will see a different approach for these graphs. Instead of focusing on the players age or position, now we will take into account their nationality. The best way to show this is probably using a map, which Power BI allows us to do. With this premise, we have the graph shown in figure 8.7. This map tells us the following:

- There is a huge predominance from two places: Europe and United States. This could be expected since United States is the country where basketball is more popular in the entire world, and they produce a huge number of high level players. It also makes sense to find a big European influence in this map since we are talking about the 10 main European leagues, so we will find local players playing in them.
- Africa and South America have some relevance, but they are very far from the impact from the other two places mentioned before.
- Finally, we can see how Asia and Oceania have little to no impact in this graph. This also makes sense because basketball is not a big sport in Asia, and Oceania has its own leagues which are strong enough to keep their local players there.

Figure 8.7: Map of players that score more points

8.8 Points scored by opponent league position

One more interesting thing to analyze is if players score more points against teams that are presumably weaker. One way to see this is by looking up if they score more points against teams that have low positions in the league. This is what the graph in image 8.8 shows us. Since including all the 10 leagues would make the graph difficult to understand, we just took the Spanish ACB and the Turkish BSL, which are the leagues ranked 1st and 2nd in our system. It is also worth mentioning that the Spanish ACB hast more league positions than the Turkish BSL because COVID restrictions during some of the seasons that we are analyzing caused that more teams participated in it.

Here are some things that we can see in the graph:

- Coinciding with what we saw in section 8.2, we can see that clearly the Turkish BSL averages more points than the Spanish ACB.
- However, contrary to what we may have thought, we do not see a big difference in terms of points averaged between the different league positions. It may be logical to think that players would have scored more against lower ranked teams, but that does not seem to be the case.

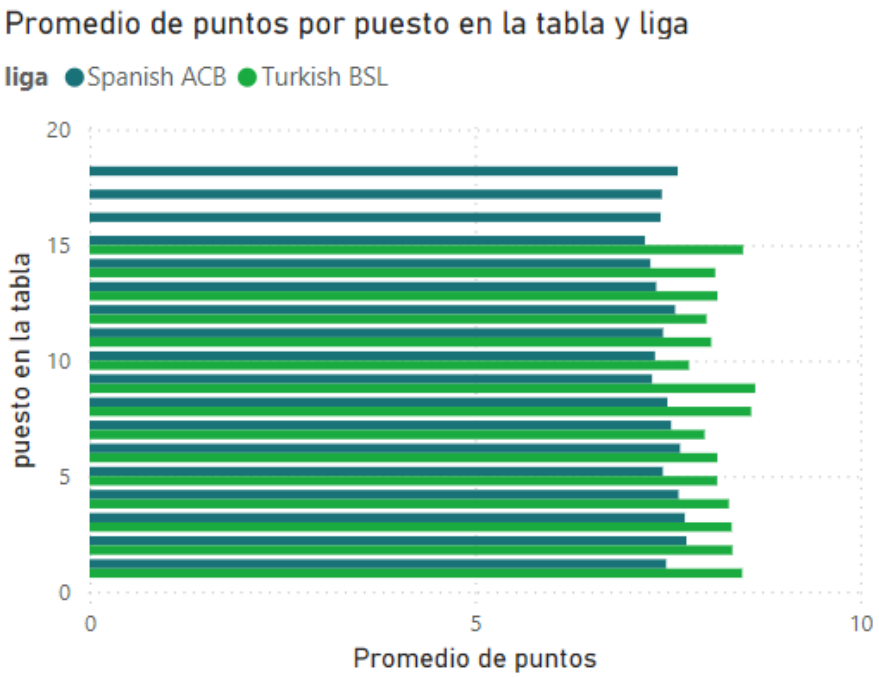


Figure 8.8: Points averaged against each league position in Spanish ACB and Turkish BSL

8.9 Minutes played by age

As it was briefly discussed in section 8.5, the age of a player usually affects the number of minutes they play in the games. This is something that we can appreciate in the graph from image 8.9. Taking a look at it we can say the following:

- It is quite clear that coaches tend to have more trust on experienced players rather than younger ones. This is the reason why the number of minutes increases drastically when players reach around 25 years of age.
- Something striking is that players of around 35 years of age play a similar amount of minutes that players around 25 years of age. This may not be something that the general public would think, since it is common to think that players of 33 or more years are usually veteran players that do not play a big role on the court. However, that is not the case. A good reason behind this may be that the sports medicine has evolved to a point that allows players to have longer careers than they used to in previous eras.

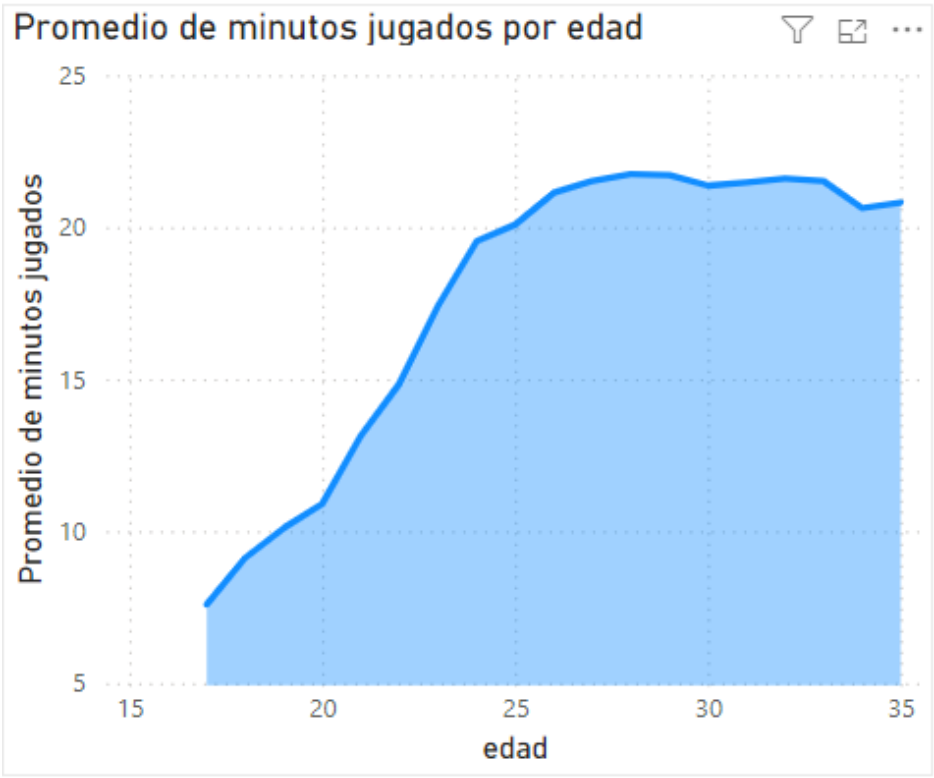


Figure 8.9: Minutes averaged by age

8.10 Points scored by game nature

As we know, we distinguish between two types of games: regular season and playoff. Playoff games are usually more important than the regular season ones, since the league winner is decided in the playoffs. It is interesting to see if players improve their numbers in playoff games. The graph in image 8.10 shows this. We can say some things about it:

- We can appreciate how in every analyzed league, the points averaged per player does not vary a lot from regular season games to playoff ones. There are a few leagues where it increases a bit in the regular season, a few where it increases in the playoffs and even a couple leagues that have the exact same number.
- This is something interesting to see because the general public could think that players would score more points in the playoffs since they are trying to win their league so they play harder than usual. However, that is not the case. Some of the reasons behind this can be that defenses also get tougher during playoff games, countering the offensive improvement, and also the fact that playoff games are played at the final stages of the season, when players usually tend to have more accumulated fatigue.

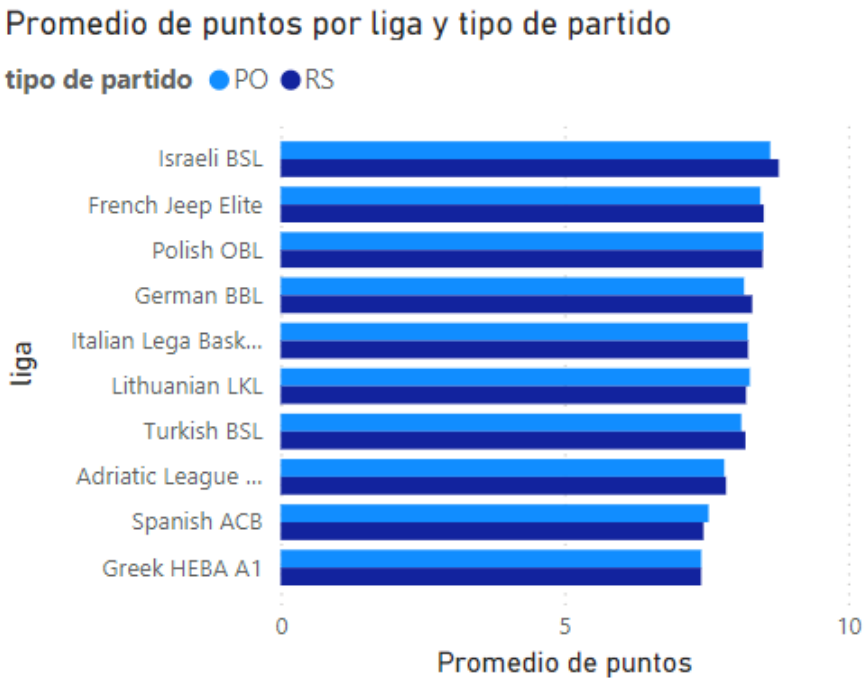


Figure 8.10: Points averaged by game nature

8.11 Transactions made by origin league

After analyzing the performance fact, it is time to work with the transaction one. We will start this by showing where do teams sign a bigger number of players from. Before showing and explaining it we must say that, for spacing purposes, these graphs will not show the league names in them. Instead, we will have their correspondent id. Here is a list of what number is assigned to each league, so we have full context to analyze it properly:

- 1: Spanish ACB.
- 2: Turkish BSL.
- 3: Italian Lega Basket Serie A.
- 4: French Jeep Elite.
- 5: German BBL.
- 6: Adriatic League Liga ABA.
- 7: Greek HEBA A1.
- 8: Israeli BSL.

- 9: Lithuanian LKL.
- 10: Polish OBL.

With this being explained, we can now see the graph in image 8.11. There are some interesting things to comment about it:

- The first thing that catches our attention is clearly the immense predominance of transactions that do not have an origin league. A bit over 80% of the stored transactions have this in common. This happens because most teams prefer to sign free agent[2] players because they do not have to pay any money to another team, since no team has his rights at that moment. Also, players often prefer to sign for a new team as free agents rather than switching directly from a team to another. The reason behind this is that free agency allows players to weigh their options and mull different offers, which often ends up in bigger salaries for them.
- Focusing on the non free agent transactions, we can see how the Spanish ACB and the French Jeep Elite are the leagues that export the greater percentage of players compared to the rest. Even if there is not a huge difference with the other, this makes sense because both Spain and France are traditionally two of the European countries with more basketball pedigree, so it is logical to think that they would produce more professional players than the rest.

8.12 Transactions by origin league breakdown

This section will be similar to the previous one, but now we will separate the transactions by leagues. Again, the names will not appear in the graphic and we will use the same numbers as in section 8.11.

The graph in image 8.12 shows us in detail from what league do the players signed in each league come from. It is important to mentioned that we skipped the free agent transactions in this graph, so we could just focus on those transactions that actually have an origin league and a destination league. Some things worth mentioning about the graph are:

- It is interesting to see how every single league has itself as the most common origin league. This shows how teams prefer sending their players to their local leagues rather than abroad. This may be surprising since we could think that teams would not want to let their players go to a rival of their same league and benefit them. However, that does not seem to be the case.

[2] Free agent: player that currently has not a contract with any team.

Figure 8.11: Number of transactions by origin league

- Another aspect that strikes is how big is the difference regarding the number of transactions between leagues. We can see how some leagues, such as the Spanish ACB or the French Jeep Elite make a considerably bigger number of transactions than others such as the Lithuanian LKL. This comes to show how some leagues have more economic power than others, which allows them to carry out more player transactions.

Figure 8.12: Number of transactions separated by leagues

Chapter 9

Conclusions

This project consisted on designing, implementing and exploiting a data warehouse about basketball statistics of the 10 main European leagues during their last 5 seasons. This included all its designing phases, as well as the ETL process necessary to implement it.

Firstly, some key theoretical concepts about data mining were introduced. These were needed in order to understand the purpose and importance of this type of projects. One of the main ideas of this project is demonstrating how helpful data mining can be to obtain relevant information from any specific area, and how it can be accessed and analyzed to facilitate decision making.

Another important aspect of the project was the technologies and tools used during its development. Every project needs the proper tools to be carried out in the most effective way, and this one is no exception. Tools such as Pentaho or Power BI have been proved as good decisions to perform the different parts of this project.

Another aspect worth mentioning is the designing section of the project. Both the conceptual and the logical design were focused on the realization of diagrams that explained the different dimensions, facts and relations between them that we have in our project. These two designs served as a fundamental base for the physical design, when we started building our system with PostgreSQL interface pgAdmin 4.

At this point of the project, the ETL process was carried out. This process is often the most important one during data warehouse projects, because it is responsible for ensuring that our data warehouse will be properly built. It must guarantee many key aspects such as data integrity, consistency and quality.

Lastly, we finished the project with the exploitation of our data warehouse once it was properly built. Power BI allowed us to load our data warehouse and develop a series of visual designs and graphs that allow us to obtain interesting information about our domain. The idea is that this information could be helpful if required for people involved in decision making in the basketball world.

To summarize, we can say that this project proves the huge relevance of data control and data mining in current times, specifically in the sports sector. It opens up a whole new world of possibilities for teams and players that they did not have before, and this trend will keep going for many years to come. It is safe to assume that we are just scratching the surface of data analysis in the sports world, and that it is something that will continue to be a key factor during many years.

About future lines of work for this project, there are a few worth mentioning:

- Real time analysis: it would be of big interest if there could be a way to efficiently store data in real time. This means, at the same time that a game is being played, all its statistics of interest would be recorded and properly stored, and the correspondent graphics updated.

- Clustering: something very common in data warehouse projects is finding groups of elements that have a similar behaviour in the domain. This technique allows us to find patterns that would be helpful for prediction purposes, for example. In our project, we could use this to group players based on their playing style, which would be given to us based on their in game statistics.

- Video analysis: this technique is very common in the sports world. It consists on obtaining our data directly from recordings of the games. It would be of special interest in certain situations, for instance, when you want to analyze how a team plays while recording their stats at the same time. If the statistics part is automatized, you can just focus on the game itself.

All of these techniques would be solid ways to follow this project, since they would probably allow us to keep digging in our data warehouse and would open up new possibilities for us to keep exploiting it.

Appendices

Appendix A

Material adicional

A.1 Code to scrape the list of players

Here is the example of a python code to scrape the players from all the leagues during all the seasons:

```
from bs4 import BeautifulSoup
from datetime import datetime
import requests
import pandas as pd
import time
import re

def get_player_info(player_row):
    cells = player_row.find_all("td")
    player_info = {
        "nombre": cells[0].text.strip(),
        "posicion": cells[1].text.strip(),
        "altura": cells[2].text.strip(),
        "peso": cells[3].text.strip(),
        "equipo": cells[4].text.strip(),
        "ciudad": cells[5].text.strip(),
        "nacionalidad": cells[6].text.strip(),
    }
    return player_info

def get_age_from_player_page(player_url, age_adjustment):
    max_retries = 3
    for attempt in range(max_retries):
        try:
            response = requests.get(player_url)
            if response.status_code == 200:
                soup = BeautifulSoup(response.content,
```

```
    "html.parser")
                birth_info = soup.find("strong", string="Born:")
                if birth_info:
                    birth_date_str =
    birth_info.find_next("a").text.strip()
                    birth_date = datetime.strptime(birth_date_str,
    "%b %d, %Y")
                    today = datetime.now()
                    age = today.year - birth_date.year -
    ((today.month, today.day) < (birth_date.month, birth_date.day))
                    return age - age_adjustment
            break
        except requests.exceptions.RequestException as e:
            if attempt < max_retries - 1:
                time.sleep(2)
            else:
                return None
    return None

def get_players_data(url, age_adjustment):
    max_retries = 3
    for attempt in range(max_retries):
        try:
            response = requests.get(url)
            if response.status_code == 200:
                soup = BeautifulSoup(response.content,
    "html.parser")
                table = soup.find("table", {"class": "tablesaw"})
                if table:
                    player_rows = table.find_all("tr")[1:]
                    players_data = []
                    total_players = len(player_rows)
                    for i, row in enumerate(player_rows, start=1):
                        player_info = get_player_info(row)
                        base_url = "https://basketball.realgm.com"
                        player_url = base_url +
    row.find("a")["href"]
                        player_info["edad"] =
    get_age_from_player_page(player_url, age_adjustment)
                        players_data.append(player_info)
                    return players_data
            break
        except requests.exceptions.RequestException as e:
            if attempt < max_retries - 1:
                time.sleep(2)
            else:
```

```
                return []
    return []

urls_age_minus_4 = [
    #Full list of 2019-2020 season players urls
]

urls_age_minus_3 = [
    #Full list of 2020-2021 season players urls
]

urls_age_minus_2 = [
    #Full list of 2021-2022 season players urls
]

urls_age_minus_1 = [
    #Full list of 2022-2023 season players urls
]

urls_no_adjustment = [
    #Full list of 2023-2024 season players urls
]

all_players_data = []

# Procesar URLs
for url in urls_age_minus_4:
    players_data = get_players_data(url, age_adjustment=3)
    all_players_data.extend(players_data)

for url in urls_age_minus_3:
    players_data = get_players_data(url, age_adjustment=3)
    all_players_data.extend(players_data)

for url in urls_age_minus_2:
    players_data = get_players_data(url, age_adjustment=2)
    all_players_data.extend(players_data)

for url in urls_age_minus_1:
    players_data = get_players_data(url, age_adjustment=1)
    all_players_data.extend(players_data)

for url in urls_no_adjustment:
    players_data = get_players_data(url, age_adjustment=0)
    all_players_data.extend(players_data)

```

```
df = pd.DataFrame(all_players_data)

#Establish our ruta_archivo

df.to_csv(ruta_archivo, sep=';', index=False)
```

As we can see, the biggest challenge was to obtain the age of a player since it is not directly given to us by the website. We had to obtain it from every players individual preview, and adapt it for each one of the 5 seasons if necessary.

A.2 Code to obtain all our dates

We can obtain all the dates we require for this project (from 2019 to 2024) with this code:

```
import pandas as pd

def generar_csv_horas_con_turno(ruta_archivo):
    fechas = pd.date_range(start='2019-01-01', end='2024-12-31')

    df = pd.DataFrame({'fecha': fechas})

    df['dia_semana'] = df['fecha'].apply(lambda x: x.strftime('%A'))
    df['dia_mes'] = df['fecha'].dt.day
    df['quincena'] = df['fecha'].dt.day.apply(lambda x: 1 if x <=
    15 else 2)
    df['mes'] = df['fecha'].dt.month
    df['trimestre'] = df['fecha'].dt.quarter
    df['año'] = df['fecha'].dt.year
    df['fecha'] = df['fecha'].dt.strftime('%m/%d/%Y')
    df.to_csv(ruta_archivo, index=False)

# Establish ruta_archivo
generar_csv_horas_con_turno(ruta_archivo)
```

List of Acronyms

ACID properties Atomicity, Consistency, Isolation, Durability. 8

BI Business Intelligence. 11, 18

CSV Comma-Separated Values. iv, 5, 7, 29–31, 33, 35

DFM Dimensional Fact Model. 18, 21, 23

ETL Extraction, Transformation and Loading. 1–4, 6–8, 11, 28, 36, 38, 63

HTML HyperText Markup Language. 9

JSON JavaScript Object Notation. 7

SCD Slowly Changing Dimension. 7, 22, 23, 31–33, 37, 49

UML Unified Modeling Language. 9

XML eXtensible Markup Languag. 9

Glossary

analytical environment Set of tools, technologies and practices that companies use to analyze data and use it for decision making. 4

BeautifulSoup Python package for parsing HTML and XML documents. It creates a parse tree for documents that can be used to extract data from HTML, which is useful for web scraping. 28

bussines intelligence Set of tools, techniques and processes used to get access to data, present it and analyzing it with the purpose of transforming it into useful knowledge. 2, 6, 17

data warehouse Database designed with the objective of storing large amounts of data from different sources, which will end up being used to help in decision making. v, 1–6, 8, 10, 14, 17, 18, 21, 31, 32, 34, 36–38, 44, 49, 50, 53, 63, 64

dimension Entity that describes the objects present in your data warehouse table. v, 7, 18, 20–22, 29, 30, 34, 36, 39, 40

fact Measurable events related to the functional area covered by a data warehouse. iv, 17, 18, 20, 21, 23–25, 29, 30, 34, 36, 59

foreign key Column (or combination of columns) in a table whose values must match values of a column in some other table. 38

operational environment Set of systems and processes used to gather, store and manipulate data. 4

pandas Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. 29

primary key Column or columns that contain values that uniquely identify each row in a table. 22, 31, 33

project planning Systematic and strategic process of defining, organising, and outlining all the essential components and activities required to achieve a specific goal or objective within a predetermined time frame and budget. 2, 12

surrogate key Unique identifier derived from the data itself. 22, 23, 33

treemap Diagram representing hierarchical data in the form of nested rectangles, where the area of each one corresponds to its numerical value in proportion to the total amount. 53

Bibliography

[1] "Kimball dw/bi lifecycle methodology," 2008, last accessed 24/05/2024. [Online]. Available: https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dw-bi-lifecycle-method/

[2] "Star schema," 2004, last accessed 26/05/2024. [Online]. Available: https://en.wikipedia.org/wiki/Star_schema

[3] "How does data analysis influence business decision making?" 2024, last accessed 09/05/2024. [Online]. Available: https://www.sage.com/en-us/blog/how-does-data-analysis-influence-business-decision-making/

[4] "Data warehousing: Best assistance for business decision-makers," 2023, last accessed 13/05/2024. [Online]. Available: https://www.existbi.com/blog/data-warehousing-assistance-decision-makers/

[5] "What is web scraping and what is it used for?" 2023, last accessed 19/05/2024. [Online]. Available: https://www.parsehub.com/blog/what-is-web-scraping/

[6] "¿qué son los procesos etl?" 2017, last accessed 15/05/2024. [Online]. Available: https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/qu-son-los-procesos-etl

[7] "What is business intelligence?" 2023, last accessed 20/05/2024. [Online]. Available: https://www.techtarget.com/searchbusinessanalytics/definition/business-intelligence-BI

[8] "¿qué es pentaho data integration?" 2022, last accessed 22/05/2024. [Online]. Available: https://keepcoding.io/blog/que-es-pentaho-data-integration/

[9] "What is postgresql?" 2021, last accessed 22/05/2024. [Online]. Available: https://www.ibm.com/topics/postgresql

[10] "¿qué es power bi y cuáles son sus características?" 2020, last accessed 23/05/2024. [Online]. Available: https://www.xmslatam.com/que-es-power-bi-y-cuales-son-caracteristicas/

[11] "Importance of project planning: An overview." 2023, last accessed 24/05/2024. [Online]. Available: https://www.theknowledgeacademy.com/blog/importance-of-project-planning/

[12] "Salario medio para analista de datos en españa, 2024," 2024, last accessed 25/05/2024. [Online]. Available: https://es.talent.com/salary?job=analista+de+datos

[13] "Fact table vs. dimension table: What´s the difference?" 2024, last accessed 26/05/2024. [Online]. Available: https://builtin.com/articles/fact-table-vs-dimension-table

[14] R. Kimball, "Slowly changing dimensions," 2004, last accessed 24/05/2024. [Online]. Available: https://www.kimballgroup.com/2008/08/slowly-changing-dimensions/