
POLITECNICO DI TORINO

Master degree course in Digital Skills for Sustainable Societal Transitions

Master Degree Thesis

# Design of a Containerized Dashboard Platform and AI-Assisted Data Enrichment for Student Career Monitoring

**Relatori**
Prof. Gianvito Urgese
Dr. Giuseppe Fanuli

**Candidato**
Barati Mohammad

February 2026

# Abstract

The growing availability of data offers significant opportunities to improve data-driven decision-making in complex organizational contexts. In higher education, effectively integrating and analyzing heterogeneous data sources is essential for evaluating educational programs, supporting governance processes, and enabling evidence-based planning.

However, institutional and external data are often fragmented across multiple systems and formats and subject to access and governance constraints, making systematic analysis difficult without dedicated data engineering solutions. This thesis presents the design and implementation of a containerized, end-to-end data pipeline for data integration, enrichment, and visualization to support analytical activities in academic environments.

The proposed methodology is based on a modular architecture that combines data ingestion from heterogeneous sources, transformation and harmonization workflows, structured storage, and interactive dashboards. Structured institutional data are integrated with complementary external information using a semi-automatic acquisition strategy designed to operate under realistic operational constraints. The pipeline is organized into distinct stages including ingestion, transformation, integration, storage, and visualization.

Data harmonization is implemented through Python-based workflows to resolve inconsistencies in schemas and identifiers, while semi-structured external data is processed through an agent-based information extraction component built on a local large language model runtime, Ollama. All processed data are stored in a relational database implemented using PostgreSQL and made accessible through interactive dashboards developed with Grafana. The entire infrastructure is containerized using Docker to ensure portability and ease of deployment. The methodology is validated through a concrete use case focused on the analysis of academic career within a PhD program at the Politecnico di Torino.

# Acknowledgments

I express my sincere gratitude to my supervisor, Prof. Gianvito Urgese, for their invaluable guidance, constructive feedback, and continuous support throughout the development of this thesis. I am also grateful to Giuseppe Fanuli for his insightful discussions and technical advice, which greatly contributed to shaping the methodology and improving the outcomes of this work.

I would like to thank my family and friends for their encouragement, patience, and unwavering support during this demanding but rewarding journey.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Over the previous decade, the dramatic prevail of digital data has fundamentally impacted and reshaped how institutions, organizations, and decision-makers operate. Universities are no more exceptions. Academic institutions, continuously generate large volumes of data related to students, staff, research outputs, fundings, and career trajectories. Nonetheless, the availability of data does not automatically necessarily mean actionable knowledge. On the contrary, the heterogeneity, fragmentation, and dispersion of data sources normally makes it complex and difficult to extract meaningful insights in a systematic, reliable, and scalable way. This challenge, is particularly evident when data is distributed across different formats and levels of structure, ranging from institutional databases to semi-structured documents and external public sources.

Data intelligence, and analytics play a very critical role in addressing such challenge. Across multiple fields—including business intelligence, healthcare analytics, smart cities, finance, and education—modern data pipelines are being used to ingest, transform, integrate, and analyze heterogeneous data, in order to support informed decision-making [1][2] . In academic environments, data intelligence enable universities to monitor key performance indicators, evaluate educational outcomes, evaluate research productivity, and understand long-term career trajectories of graduates. These insights is becoming increasingly important not only for internal governance but also for accreditation processes, strategic planning, and accountability toward funding bodies and public institutions.

Within the broader context, the analysis of PhD students' careers represent a particularly complex and relevant use case. PhD programs are known as long-term investments, both for students and institutions, and their outcomes can not be evaluated solely based on completion rates or time-to-degree. Post-graduation career paths—whether in academia, industry, or any other sectors—could be key indicators of the effectiveness and societal impact of doctoral training [3]. Data related to PhD students are normally scattered across internal university portals,

administrative databases, manually curated files, and external sources such as professional networking platforms. As a result, performing even very basic analyses frequently might requires manual effort, ad hoc data cleaning, and non-reproducible workflows.

This thesis address this concrete and non-trivial problem by a focus on the analysis of PhD students' performance and career outcomes in the Department of Control and Computer Engineering at the Politecnico di Torino. The work does not aim to propose a novel theoretical model; Rather, it aims to present a practical, end-to-end data pipeline. It would demonstrate a modern and effective approach to harmonization and integration of heterogeneous data sources, and transforming them into accessible and meaningful intelligence dashboard. In doing so, the thesis aligns with the growing body of applied research and industrial practices that emphasize robust data engineering architectures as a prerequisite for advanced analytics and decision support [4].

The starting point of the project, is the recognition that the available data is heterogeneous. Core information about PhD students is stored in institutional databases, and provided in the form of comma-separated value (CSV) files which were extracted from the university portal. While these datasets are structured, they are not normally consistent across sources. Differences in identifiers, naming conventions, missing values, and schema evolution brings about significant challenges in attempting to integrate them into a single coherent dataset. In parallel, information about the professional status of PhD students, particularly after graduation, is not recorded within the university systems. This information is often available only through external sources, such as professional profiles on platforms like LinkedIn.

Retrieving and integrating such external data normally might introduce additional layers of complexity. External sources are usually semi-structured or unstructured, and may require manual access. They are also subject to variability in content and format. In this project, public PDF documents containing professional information were manually collected; this reflects a realistic constraint often encountered in institutional settings as automated scraping is normally restricted for legal or ethical reasons. The challenge, therefore, was not only to store these documents; but also to extract relevant information—such as employment status, sector, and role—in a systematic, reproducable, and scalable way.

In order to address this challenge, this thesis integrates a modern data pipeline architecture which combines a traditional data engineering technique with the recent advances in large language models (LLMs). In particular, agent-based system built on top of Ollama is used to extract structured employment information from unstructured PDF documents. This approach takes into account the natural language understanding capabilities of LLMs, to identify and classify relevant fields from textual data, which demonstrates how AI-based components could be integrated into data pipelines. Similar approaches have recently gained attention, in

both academia and industry, where LLMs are used for extracting information, understanding documents, and enriching the semantics of data.

Once the data collected and extracted, all data undergoes a set of transformation and integration steps implemented in Python. Python was chosen because of its extensive ecosystem of data processing libraries, ease of integrating with databases, and widespread adoption in data engineering workflows [5] (McKinney, 2017). During this phase, a major effort is devoted to data cleaning, normalization, and reconciliation. For instance, student records that are identified by matriculation numbers in one dataset must be matched with matriculation in another dataset representing the publications by conducting series of joins. Also, inconsistencies in missing fields need to be resolved to ensure data integrity. These steps are crucial, to minimize the chances of poorly integrated data which would be undermining the reliability of any subsequent analysis or visualization.

The curated data is then loaded into a relational database management system. Specifically in this case, PostgreSQL is chosen. The choice of the relational database is motivated by its robustness, supporting the complex queries, and suitability for the structured analytical workloads. The database schema is designed to reflect the integrated view of PhD students' academic and professional trajectories, enabling efficient querying and aggregation in the downstream usecases. This centralized storage represents a key milestone of the project indicating that data which was previously fragmented across multiple sources and formats is now harmonized within a single, well-defined platform.

To improve the transparency, reproducibility, and version control of data transformations, dbt is integrated into the project architecture. dbt structures the transformation layer into modular, explicitly defined steps, enabling observable and maintainable transformation logic. In addition, it facilitates the creation of analytics-ready data marts that are optimized for efficient downstream querying and analysis.

To make the data accessible and actionable for the downstream end users, the pipeline entails a visualization layer which was implemented by using Grafana. Grafana connects directly to the PostgreSQL database to provide customizable dashboards, based on various option such as student name, lecturers' name, and cycles in which the student are enrolled. This allows stakeholders to explore the data interactively by themselves. Through these dashboards, members of academic commissions can analyze distributions of students academic performance in term of cumulative soft/hard skills trainings, publications, or current working status of PhD students, adapting the visualizations to their specific analytical needs. The emphasis is not on predefined static reports, but on flexibility and transparency, which in the end enables users to understand the results.

An important architectural aspect of the project is its full containerization within the Docker ecosystem. By encapsulating each component of the pipeline—data

ingestion, transformation scripts, database, LLM-based agent querying, and visualization layer—within containers, the system achieve a high degree of reproducability, portability, and ease of deployment. Containerization reflects current best practices in data engineering and software development, where infrastructure-as-code and modular architectures are progresively adopted to reduce operational complexity and improve maintainability [6].

This thesis aims to demonstrate a state-of-the-art way of working with data in institutional context. By integrating structured university data, with semi-structured external information, leveraging LLM-based agents for reasoning and analysis, and finally, delivering results through interactive dashboards, this very project reflects a broader shift toward data-driven decision-making supported by flexible and scalable infrastructures. While the specific use case focuses on PhD students in a single department, the architectural principles and methodological choices can be generalized to other academic units and organizational settings.

The remainder of this manuscript is organized as follows:

**Chapter 2** offers an in-depth overview of the literature, covering data intelligence and the role of data pipelines, data integration from heterogeneous data sources, relational databases for analytical workloads, visualization and decision support systems, large language models for information extraction, dbt in modern data transformation architectures, python ecosystem for data engineering, and containerization and reproducability in data engineering pipeline development.

**Chapter 3** describes the methodology and the architecture of the data system in question, starting from questions to be addressed by the platform, and continuing to the architecture including different layers of data ingestion, data staging, data warehouse, data marts, data visualization, data storage, and containerization and deployment. It also presents an abstraction of the execution and workflow of the architecture.

**Chapter 4** evaluates the intermediate results, diving into the observed issues and harmonization results, as well as database level results engulfing the analytical views and query stability. It also presents the the different dashboards results curated throughout the project, including teaching and training hours distribution, course-level enrollment and exam outcomes, international mobility analysis, training hours and program requirements, and publication analysis.

**Chapter 5** summarizes the main outcomes and milestones, followed by the advantages and potential implications of the system in question. It also illustrates the limitations of the curated system and envisioning the future work implications, including automating employment status acquisition, integration of workflow orchestration tools, and integration of generative sql and llm-based querying.

In conclusion, this work presents a practical and comprehensive solution to the problem of analyzing PhD students' career outcomes using heterogeneous data sources. By design and implementation of a complete data pipeline—from ingestion to visualization—the thesis shows how cutting-edge technologies can be applied to

a real-world problem that is highly relevant for academic governance but often underserved by existing tools. The result is a unified platform that enables the academic commission to access, explore, and interpret data in a way that was previously not possible, laying the groundwork for more informed and data-driven evaluations of doctoral programs.

# Chapter 2

# Literature Review

## 2.1 Data Intelligence and the Role of Data Pipelines

Data intelligence has become a fundamental element in modern organizations. It enables the transformation of raw data all the way through actionable knowledge to supports strategic and operational decision-making. The concept engulfs, **data collection, integration, analysis, and visualization**, and is tightly copled with the evolution of data engineering architectures. Early approaches to data intelligence relied on monolithic data warehouses; where data from operational systems was periodically **extracted, transformed, and loaded (ETL)** into centralized repositories [2]. While it was effective for structured and relatively stable data sources, these architectures struggle to adapt to the volume, velocity, and variety of contemporary data.

More recent paradigms emphasize modular, scalable, and automated data pipelines which capable of handling heterogeneous data sources [1][7]. A data pipeline can be defined, as an end to end system, that ingests data from one or more sources, applies a series of transformations, and delivers the processed data to storage systems or analytical tools. Reis and Housley [4] highlight that modern data pipelines prioritize reproducibility, observability, and maintainability, often leveraging containerization and orchestration technologies.

In academic and institutional contexts, data pipeline play a very crucial role to enable evidence-based decision-making. Universities normally generate data across a wide spectrum of activities, including admissions, course delivery, research output, and alumni tracking. Nevertheless, these data are often siloed across different departments and information systems, which limits the analytical value [8]. The literature consistently identify data integration as one of the most primary challenges in educational data analytics, particularly when combining administrative records with external or semi-structured sources.

## 2.2 Heterogeneous Data Integration

Heterogeneous data integration refers to the process of combining data that might differs in the *format, structure, semantics,* or *origin*. According to Lenzerini, heterogeneity can be classified into syntactic heterogeneity (different data formats), structural heterogeneity (different schemas), and semantic heterogeneity (different meanings or interpretations of data fields) [9]. In practice, real-world datasets often exhibit all three forms in parallel to each other.

CSV files extracted from institutional databases are a common example of semi-structured data. While CSV is simple, and widely supported, it lack explicit schema definitions and metadata. This, increases the risk of inconsistencies when datasets are generated by different systems or at different times [10]. Studies in data engineering emphasize the importance of schema reconciliation, data cleaning, and normalization to address these issues [11].

In the context of higher education, several work highlight the difficulty of integrating student-related data, across administrative systems, learning management systems, and external platforms [12]. These challenges are normally exacerbated when unique identifiers are not consistently used, requiring record linkage based on names, dates, or even any other potentially ambiguous attributes. Techniques such as deterministic and probabilistic record linkage have been proposed in the literature [13], but their effectiveness depends heavily on data quality and availability of matching fields.

The project described in this thesis aligns with these challenges by addressing the integration of heterogeneous CSV files originating from a university portal, where differences in identifiers and field definitions must be resolved before meaningful analysis can be performed. The literature supports the notion that such integration is a prerequisite for any form of reliable analytics, particularly in institutional settings where data governance constraint limit the ability to redesign upstream systems.

## 2.3 Relational Databases for Analytical Workloads

Despite the growing popularity of **NoSQL** (Not Only SQL), relational database management systems (**RDBMSs**) still remain a cornerstone of analytical infrastructures; Especially, when dealing with structured and moderately sized datasets. Systems such as **PostgreSQL** are widely used in both academia and industry due to their robustness, support for **SQL** standards, and extensibility [14].

The literature on data warehousing, emphasizes the importance of schema design in enabling efficient analytical queries [1]. Star and snowflake schemas are commonly adopted to separate fact tables from dimension tables. This improves

query performance and interpretability. While this thesis does not aim to implement a full-scale enterprise data warehouse, the principles of relational modeling and normalization are directly relevant. A well-designed schema facilitates data consistency, reduces redundancy, and supports complex aggregations required for performance analysis.

In educational analytics, relational databases are frequently used as backend systems for dashboards and reporting tools [15]. Their compatibility, with visualization platforms, and ability to handle transactional updates, make them suitable for the iterative and exploratory analysis. The choice of PostgreSQL in this project also reflects best practices documented in the literature, particularly for applications that require both the reliability and flexibility.

## 2.4 Visualization and Decision Support Systems

Data visualization, is a critical component of data intelligence. It serve as the primary interface among complex datasets and human decision-makers. Due to Few (2009), effective visualizations could reduce cognitive load, and enable users to identify patterns, trends, and anomalies, the ones which would be difficult to detect through raw data, or tabular reports[16]. Within institutional settings, dashboards also are widely used to monitor key performance indicators (KPIs) and enhance and support strategic planning.

**Grafana**, as a modern visualization platforms, is designed to connect directly to databases and provide interactive, real-time dashboards[17]. Unlike traditional static reporting tools, such platform allows users to customize views, apply filters, and dive deep into underlying data. The literature highlights that such interactivity is particularly valuable in exploratory analytics, where users may not have predefined hypotheses [18].

In higher education, dashboards have been used to analyze student performance, retention rates, and learning behaviors [19]. However, many existing systems focus primarily on teaching and learning analytics, with less attention given to long-term career outcomes of doctoral students. This gap underscores the relevance of the present work, which extends visualization practices to a domain that is increasingly important for institutional evaluation but still underexplored in the literature.

## 2.5 External Data Sources and Career Tracking

The analysis of post-graduation career trajectories has so long been recognized as a critical, yet methodologically challenging aspect of higher education analytics. Traditional institutional data sources, typically, provide proper coverage of enrollment, progression, and completion. However, they fail to capture employment outcomes

once graduates leave the academic system [3]. As a result, universities lack systematic and longitudinal visibility into the professional destinations of PhD graduates. Despite the strategic relevance of such information for program evaluation, funding accountability, and policy-making.

Conventional approaches to career tracking are reliant on alumni surveys, administrative follow-ups, or national statistics. While valuable, these methods normally suffer from well documented constraints, including insufficient response rates, delayed availability, recall bias, and limited granularity [8]. In particular, survey-based methods are poorly suited for capturing such dynamic career trajectories and are costly to maintain over time. Consequently, the literature increasingly explores the use of online professional platforms as alternative data sources for graduate outcome analysis.

Among these platforms, *LinkedIn* has emerged as the dominant source of publicly accessible professional career data. Prior studies demonstrate its applicability for analyzing labor market trends, occupational mobility, and skill distributions across sectors. In the context of doctoral education, *LinkedIn* profiles often contain detailed chronological employment histories, sector transitions, and role descriptions that are otherwise unavailable through institutional channels.

However, the literature is explicit in acknowledging that *LinkedIn* data is not readily amenable to automated institutional use, as programmatic access is restricted, automated scraping violates platform policies, and API-based solutions are either unavailable or incompatible with academic research constraints [20]. These limitations are particularly relevant for universities, where legal compliance, ethical responsibility, and data governance requirements outweigh the benefits of large-scale automation.

As a result, multiple studies adopt manual or semi-manual data acquisition strategies, especially when the target population is limited and the objective is high-quality, interpretable data rather than large-scale statistical inference [15]. In such settings, manual access to publicly available profiles represents a pragmatic and defensible methodological choice rather than a technical shortcoming.

This thesis follows this line of work by employing manual collection of publicly visible LinkedIn profiles, which are stored as PDF documents for subsequent processing. The use of PDFs provides a stable and archivable representation of professional information at a given point in time, facilitating reproducibility and auditability. While this approach introduces human involvement at the data acquisition stage, it reflects the realistic constraints under which institutional analytics must operate and is consistent with established practices in educational data research.

## 2.6 Large Language Models for Information Extraction

Extracting structured information unstructured documents has been a problem in data management since long ago. Traditional information extraction techniques normally rely on rule-based systems; pattern matching, or supervised machine learning models trained on annotated datasets [11]. While effective in several special domains, these approaches exhibit limited robustness when applied to varied documents, characterized by high heterogenity in layout and narrative structure[21].

Professional documents derived from LinkedIn profiles, particularly when stored as PDFs, exemplify this challenge. Employment information may be presented through free-text descriptions, chronological timelines, or mixed-format sections without consistent semantic markers. Recent advances in **large language models (LLMs)**, however, has drastically altered the view on information processing and extraction. Transformer-based architectures and large-scale foundation models shows strong capabilities in contextual understanding, semantic inference, and zero-shot or few-shot task generalization [22]. Unlike traditional systems, LLMs can infer implicit structure and meaning from raw text, making them particularly suitable for extracting information from heterogeneous professional documents.

Nevertheless, the literature cautions against unconstrained use of **LLMs** in data pipelines. Issues such as hallucinated outputs and inconsistent formatting poses risks when LLM-generated data is directly consumed by downstream analytical systems [23] . To mitigate these risks, recent research advocates for agent-based architectures, where **LLMs** operate within explicitly defined tasks, prompts, and validation workflows [24] .

In this thesis, an agent-based extraction strategy is adopted, leveraging a local LLM runtime implemented using **Ollama**. The agent is tasked to extract a predefined set of employment-related attributes; such as current employment status, sector, and role from LinkedIn-derived PDF documents. This approach limits model behavior, enhances reproducibility, and allows for systematic post-processing and validation of outputs.

The choice of local LLM execution is consistent with emerging best practices for privacy-preserving and institutionally deployable AI systems. Running models locally, avoids external data transmission, ensures full control over inference parameters, and aligns with data governance requirements commonly encountered in academic environments [22] .

Crucially, the extraction process is designed to be semi-automatic. Human involvement is limited to the data acquisition, while interpretation and structuring are largely handled by the agent-based system. This division of responsibilities, ensures a methodological stance supported by the literature: human oversight ensures correctness and compliance at the data boundary, while automated systems

provides consistency, scalability, and repeatability in data transformation [23] .

In summary, the use of LLMs in this thesis is not positioned as a standalone innovation but as a controlled component within a broader data engineering pipeline. By embedding LLM-based agents into a structured workflow, this work demonstrates how recent advances in natural language processing can be operationalized for reliable information extraction in real-world institutional analytics.

## 2.7 dbt in Modern Data Transformation Architectures

Evolution of data engineering architectures is progressively shifting from traditional ETL pipelines toward ELT-based approaches; through which the data is initially loaded into centralized analytical databases, and subsequently transformed for analytical use cases [1][4] . Within this paradigm, **dbt** (data build tool) has lately emerged as a widely adopted framework for managing data transformations directly within the data warehouse. dbt formalizes transformations logic, as SQL-based models that are version-controlled and modular, introducing software engineering principles into analytical data workflows [25].

From a methodological standpoint, dbt promotes a declarative and transparent approach to data transformation. Rather than embedding business logic in opaque scripts, or orchestration layers, transformations are expressed as a composable models that can be materialized as views or tables in the target database. This design pattern, enhances reproducibility, auditability, and maintainability properties, that are very consistently emphasized in the literature, as critical for long-lived analytical systems [7]. Built-in mechanisms for testing, documentation generation, and lineage tracking further support data quality assurance and governance, reducing the risk of silent data errors in downstream analyses.

Organizations typically operate under constraints such as heterogeneous data sources, evolving reporting requirements, and the need for transparent and explainable data transformations. In such contexts, dbt's layered modeling approach,commonly separating raw, staging, and business-level models, mirrors classical data warehousing principles while offering greater flexibility and lower operational overhead. Several studies and practitioner oriented analyses indicate that this modularization reduces technical debt and facilitates collaboration between technical and non-technical stakeholders [4] [26] .

Within the broader landscape of data engineering tools, dbt also occupies a very distinct position as a transformation layer framework, that complements ingestion pipelines, relational storage systems, and visualization platforms. While this thesis does not adopt dbt as the primary transformation engine, the architectural principles advocated by dbt, namely modularity, explicit dependencies and version control, inform the design choices of the proposed data pipeline. As such, dbt serves

as a conceptual and methodological reference for structuring data transformations in analytical systems where reliability, traceability, and long-term maintainability are essential.

## 2.8 Python Ecosystem for Data Engineering

Python has emerged long ago as the dominant language for data engineering and analytics supported by a rich ecosystem of libraries for data manipulation, database connectivity, and file processing [5] [27]. Libraries for CSV handling, PDF parsing, HTTP requests, and database interaction enable rapid development of data pipelines without sacrificing expressiveness or maintainability.

The literature consistently highlights Python's suitability for glue code connecting several components of a data pipeline [28]. Python is also particularly attractive because of its readability and extensive documentation as well as its integration with relational databases through libraries, such as psycopg2, which further facilitates seamless data integration.

## 2.9 Containerization and Reproducibility

Reproducibility is a central and key concern within data engineering domain. Differences in software versions, dependencies, and system configurations can lead to inconsistent results and might hinder maintainability. Containerization technologies, most notably and in particular **Docker**, have risen to address this issue by encapsulating applications and their dependencies into isolated environments [6] [29].

The literature on DevOps and data engineering emphasizes containerization as a best practice for deploying complex pipelines [26]. By containerizing databases, transformation scripts, LLM runtimes, and visualization tools, it becomes possible to reproduce the entire system on different machines with minimal effort. This is particularly relevant for academic projects, where longevity and transferability are important but often neglected.

## 2.10 Research Gap

Despite extensive research in data engineering, educational analytics, and natural language processing, the literature reveals three critical gaps when addressing the concrete problem of institutional PhD career analysis:

- **Inadequate treatment of manually acquired and semi-structured external data.** Research on heterogeneous data integration typically assumes

automated ingestion or unrestricted programmatic access to data sources. As a result, there is limited methodological guidance on how to incorporate manually collected external data—such as PDF snapshots of professional profiles—into structured data pipelines. This gap is particularly relevant in academic contexts, where manual acquisition is often unavoidable and semi-structured documents represent a dominant data format.

- **Fragmented use of LLM-based extraction in end-to-end analytics pipelines.** Although recent work demonstrates the potential of large language models for extracting information from unstructured documents, these techniques are usually presented as standalone **Natural Language Processing (NLP)**, which are concerned with enabling computers to process, analyze, and generate human language in both written and spoken form [30]. The literature rarely addresses how LLM-based, agent-driven extraction can be systematically embedded within complete data engineering pipelines, including validation, normalization, relational storage, and visualization for decision support.

In summary, the literature does not yet provide an end-to-end, institutionally viable solution that combines heterogeneous institutional data, manually acquired external career information, controlled LLM-based extraction, and interactive analytics within a unified architecture. This thesis directly addresses these gaps by proposing and implementing a complete data pipeline tailored to departmental-level PhD performance and career analysis.

# Chapter 3

# Methodology

This section describes the methodology adopted within this thesis to design, implement, and validate an end-to-end, containerized data platform for academic analytics. The methodology is structured around solving a set of concrete questions, posed by the Academic Board regarding visualization, and analysis of doctoral students' activities — from teaching involvement and course participation to publication records and theirs international mobility. The approach combines traditional data engineering techniques, AI-assisted information extraction, data warehousing, and interactive visualization, implemented within a reproducible, modular architecture.

## 3.1 Questions Addressed by the Platform

The platform developed in this project seeks to answer several analytical questions of direct interest to academic governance:

1. Teaching Activity Monitoring
    - The Academic Board requires visualizations of teaching hours by PhD students per year, per cycle, and cumulative over years.
    - Users must be able to view the set of active cycles via interactive checkboxes.
    - Selecting one or multiple cycles should update the visualizations to include only the relevant students.
    - Upon clicking on a student's bar in a visualization, the system should reveal detailed breakdowns of courses and hours for that student.
2. Publication Analytics
    - Two key publication visualizations — "Publications per Student" and "Publications on Journals by Quartile" — must support drill-downs.
    - Clicking on a publication bar element should display detailed publication information for the student.

- Additionally, the Board requested a dimension to monitor publication performance of tutors, not only students.

3. Course Enrollment and Exam Tracking
    - A visualization is needed where each row represents a course offered in a given year, and bars within each row indicate the number of students enrolled and the number of exams passed, enabling examination of cohort engagement in course activity.
4. International Mobility Patterns
    - For internationalization strategy, the Board needs to examine mobility activities: total duration of mobility periods for each student, destinations (country and host institution), timings relative to the PhD period, and distribution of mobility durations (e.g., >30 days, >5 months).
    - Cumulative and per-year summaries should be available.
5. Professional Status and Employment Outcome
    - The Academic Board expressed the need to monitor the most recent professional status of PhD students, with particular attention to employment outcomes following graduation.

The methodology described in the following sections details how these requirements are operationalized through the platform's architecture and data processes.

## 3.2 High-Level Architecture

The platform is implemented as a containerized, modular pipeline that ingests raw data, transforms it into analytical forms, enriches semi-structured content via AI, stores the results in a structured database, and exposes interactive dashboards for exploration and analysis (Figure 3.1). The pipeline consists of the following layers:

- **Extraction Layer**: Responsible for ingesting raw data from structured and semi-structured sources.
- **Staging Layer**: Performs initial cleaning, normalization, schema alignment, and AI-assisted enrichment.
- **Data Warehouse Layer**: Implements core analytical schemas and business logic using dbt (data build tool).
- **Data Mart Layer**: Organizes analytics-ready tables optimized for dashboard querying.
- **Data Serving Layer**: Visualizes the data through interactive dashboards in Grafana.
- **Data Storage Layer**: Keeps track of the data in the PostgreSQL database persistently.

- **Containerization and Execution Model**: Ensures reproducible deployment and consistent environments via Docker.

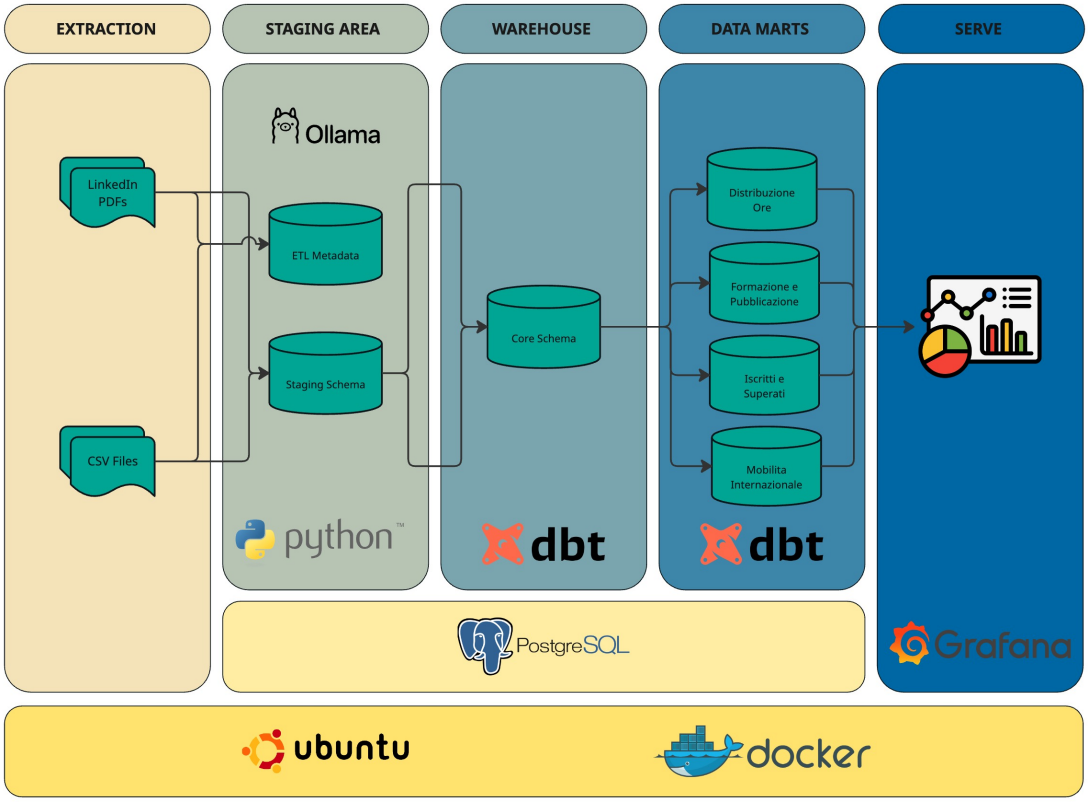


Figure 3.1: Overall architecture diagram of the proposed system

Each layer is designed to address specific transformation and integration challenges and to ensure that the platform responds to the analytical questions defined above. The following subsections describe each layer in detail.

## 3.3 Data Sources and Initial Ingestion

The platform ingests data from two broad categories of sources: structured institutional data provided in CSV format and semi-structured external data provided as PDF documents. These sources differ in terms of structure, accessibility, and processing requirements, and are therefore handled through distinct ingestion and validation strategies.

### 3.3.1 Structured Institutional Data (CSV)

Structured institutional data constitutes the core input of the platform and includes information related to PhD students' enrollment, academic activities, courses, publications, collaborations, and international mobility. These datasets are exported from institutional information systems and provided in CSV format as shown in the figure (Figure 3.2). Although structured, the files exhibit heterogeneity in terms of schemas, delimiters, and naming conventions. All institutional input files follow documented directory structures and naming patterns. In particular, student-related data is organized by PhD cycle under directories of the form `cicli/<ciclo>/`, while course-related data is organized under the `corsi/` directory. This organization enables cycle-based processing and facilitates incremental updates when new cohorts are added.

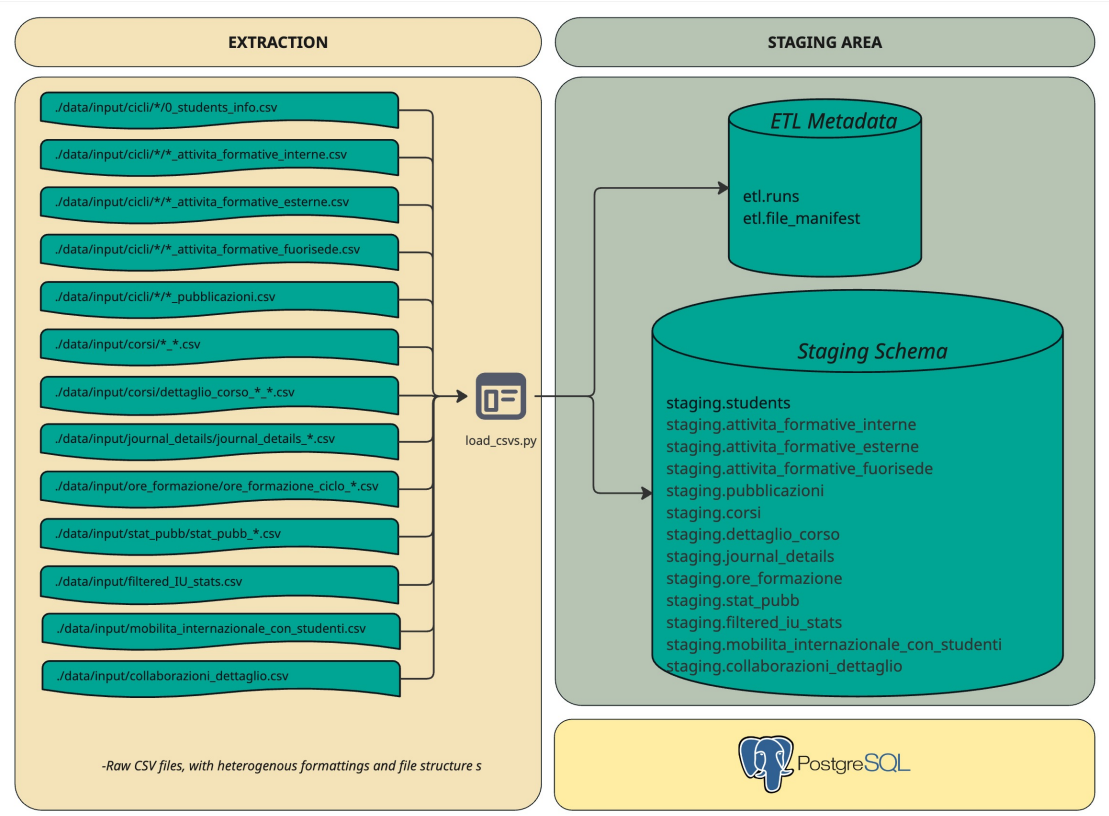


Figure 3.2: CSV lane from raw institutional files to the staging layer.

#### Student Registry and Academic Profile Data

The primary student registry file provides identifying and administrative information for all PhD students belonging to a given cycle as shown in the table 3.1. This

dataset represents the reference point for integrating all other student-related data and is expected to be located at `cicli/<ciclo>/0_students_info.csv`.

Table 3.1: Student Information Schema

| **Column Name** | **Description** | **Delimiter** |
|---|---|---|
| Matricola dottorando/a | Unique PhD student identifier | 11*; |
| Matricola da dipendente/docente | Employee identifier (if applicable) | |
| Email | Institutional email address | |
| Cognome | Last name | |
| Nome | First name | |
| Ciclo | PhD cycle | |
| Tutore | Main tutor | |
| Co-tutore | Co-tutor | |
| Status | Academic status | |
| Ore Soft Skills | Accumulated soft-skill hours | |
| Ore Hard Skills | Accumulated hard-skill hours | |

### Teaching and Training Activities

Teaching and training activities are provided at the individual student level through separate CSV files, each identified by the student *matricola* . Internal (table 3.2), external (table 3.3), and off-site activities (table 3.4) are distinguished to reflect different types of academic engagement and recognition rules.

Table 3.2: Internal Training Activities

| **Column Name** | **Description** | **Delimiter** |
|---|---|---|
| Cod Ins. | Course code | 9*, |
| Nome insegnamento | Course name | |
| Ore | Declared hours | |
| Ore riconosciute | Recognized hours | |
| Voto | Grade | |
| Coeff. voto | Grade coefficient | |
| Data esame | Exam date | |
| Tipo form. | Training type | |
| Liv. Esame | Exam level | |

Table 3.3: External Training Activities

| Column Name | Description | Delimiter |
|---|---|---|
| Denominazione | Activity name | 9*\t |
| Ore dichiarate | Declared hours | |
| Ore riconosciute | Recognized hours | |
| Ore calcolate | Calculated hours | |
| Coeff. voto | Grade coefficient | |
| Tipo form. | Training type | |
| Tipo Richiesta | Request type | |
| Data attività | Activity date | |
| Data convalida | Validation date | |

Table 3.4: Off-site Activities

| Column Name | Description | Delimiter |
|---|---|---|
| Denominazione | Name / description of the activity | 8*, |
| Luogo | Location where the activity took place | |
| Ente | Responsible or issuing organization | |
| Periodo | Reference period of the activity | |
| Data autorizzazione | Authorization date | |
| Data aut. pagamento | Payment authorization date | |
| Data attestaz. | Certification date | |
| Data convalida | Validation date | |

### Publications and Research Output

Publication data is provided per student and includes detailed metadata on research outputs, enabling both per-student and aggregated analyses (table 3.5). These files are located under `cicli/<ciclo>/` and named `<matricola>_pubblicazioni.csv`.

Table 3.5: Student Publications

| Column Name | Description | Delimiter |
|---|---|---|
| Anno | Publication year | 10*\t |
| Tipo | Publication type | |
| Titolo | Title | |
| Rivista | Journal name | |
| Autori | Authors | |
| Convegno | Conference | |
| Referee | Peer review status | |
| Grado proprietà dottorandi | Student ownership degree | |
| Punteggio | Evaluation score | |
| Indicatore R | Research indicator | |

### Courses, Exams, and Enrollments

Course-related datasets describe student enrollment and exam outcomes for each course offered in a given academic year as in the format of table 3.6. These files are expected under the `corsi/` directory with names of the form `<COD_INS>_<YYYY>.csv`.

Table 3.6: Course Enrollment File

| Column Name | Description | Delimiter |
|---|---|---|
| matricola | Student identifier | 7*\t |
| cognome | Last name | |
| nome | First name | |
| codInsegnamento | Course code | |
| codCorsoDottorato | PhD course code | |
| PeriodoDidattico | Teaching period | |
| stato | Enrollment status | |

### International Mobility Data

International mobility information describes periods spent abroad by PhD students and is essential for monitoring internationalization strategies and compliance with mobility expectations. This dataset is provided in a single CSV file (table 3.7), `mobilita_internazionale_con_studenti.csv`, at the root of the input directory.

Table 3.7: International Mobility Dataset

| Column Name | Description | Delimiter |
|---|---|---|
| matricola | Student identifier | 11*, |
| cognome | Last name | |
| nome | First name | |
| tutore | Tutor | |
| ciclo | PhD cycle | |
| tipo | Mobility type | |
| paese | Destination country | |
| ente | Hosting institution | |
| periodo | Mobility period | |
| durata_giorni | Duration (days) | |
| anno | Reference year | |

### 3.3.2 Semi-Structured External Data (PDF)

Some relevant information, particularly regarding the most recent professional status of students (e.g. current role, employer, and length of employment), is not available through institutional systems. This information is obtained from publicly available professional profiles, such as LinkedIn pages, which are manually accessed and stored as PDF documents under the external data directory. These documents contain semi-structured textual descriptions of employment history. Due to variability in layout and terminology, PDF documents cannot be processed through standard CSV ingestion. Instead, they are parsed using Python-based PDF extraction libraries and subsequently processed through an AI-assisted information extraction pipeline based on a local large language model runtime. This pipeline, as show in figure 3.3, applies deterministic prompts and schema-aware parsing to extract attributes such as most recent position, company, employment start date, and inferred employment length.

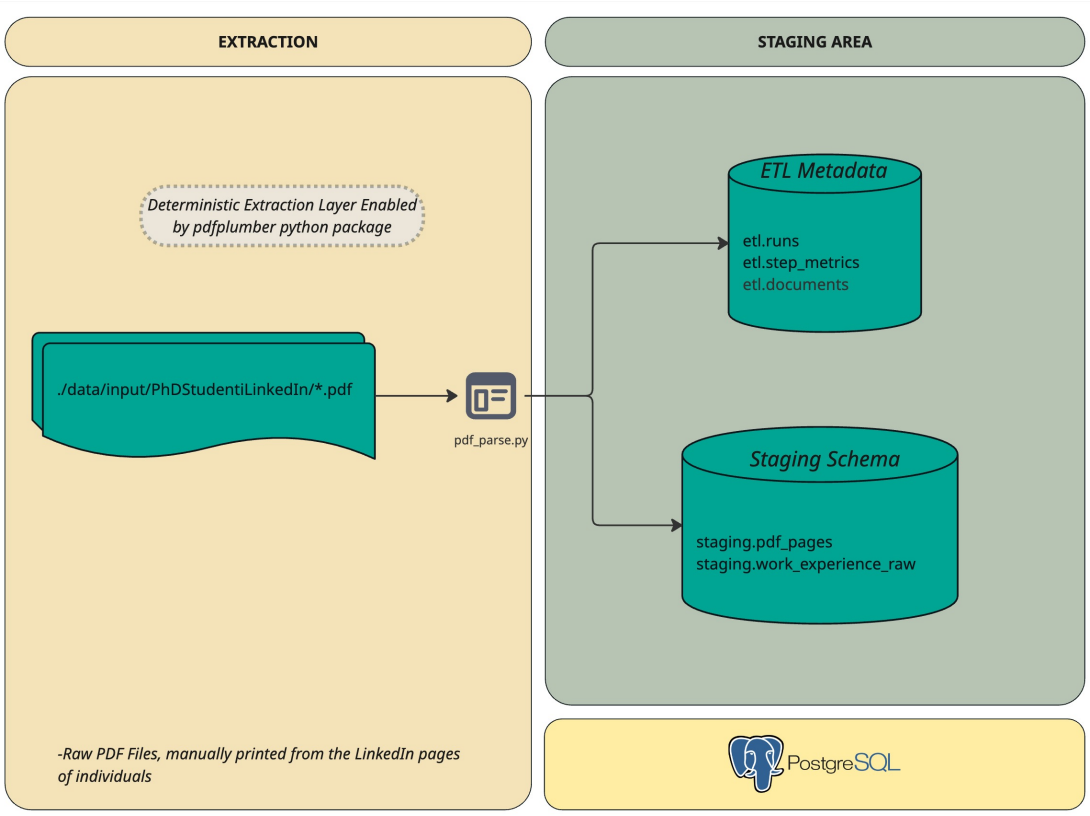


Figure 3.3: LinkedIn PDF deterministic extraction layer.

During the initial ingestion phase, all input files—both CSV and PDF—are validated through automated checks. These checks verify file presence, required columns, delimiter consistency, and basic data integrity constraints. Any missing or inconsistent formats are flagged early in the pipeline to prevent propagation of errors to downstream transformation and visualization stages.

## 3.4 Staging Layer: Cleaning, Normalization, and Enrichment

After raw files are ingested into the platform, they are processed in the staging layer, whose purpose is to convert heterogeneous inputs into consistent, quality-assured datasets that can be safely modeled in the warehouse. Concretely, the staging layer enforces a strict separation between *input idiosyncrasies* (delimiters, naming, date formats, missing fields, duplicated rows) and the *analytical semantics* required downstream.

### 3.4.1 Schema Alignment and Standardization

Source datasets often encode the same concept using different representations (e.g., student identifiers, dates, course codes, cycle labels). The staging transformations standardize these representations through:

- **Canonical column naming**: harmonization of field names across files to a unified naming convention.
- **Type normalization**: explicit casting of dates, numeric quantities (hours, scores), and categorical fields.
- **Key normalization**: normalization of primary identifiers (e.g., *matricola*) and generation of stable surrogate keys where necessary.

This step ensures that downstream dbt models can rely on stable types and keys, instead of embedding ad-hoc parsing logic in the warehouse layer.

### 3.4.2 Data Cleaning and Consistency Rules

Before data is promoted to warehouse modeling, the staging layer applies cleaning rules that prevent propagation of low-quality records:

- **Deduplication**: removal of duplicated rows and consolidation of duplicated entities produced by repeated exports.
- **Missing-value handling**: enforcement of mandatory fields, defaulting of optional fields, and explicit nullability rules.
- **Entity unification**: deterministic matching rules reconcile cases where a student appears under multiple identifiers, producing a single canonical student entity.
- **Integrity checks**: basic constraints (e.g., non-negative durations/hours, valid date ranges, referential integrity for course codes and cycles) are validated prior to load into the warehouse.

### 3.4.3 CSV Lane: From Staging to Warehouse

Once the staged CSV-derived tables are standardized and validated, they are exposed as the contract boundary for warehouse modeling. The warehouse layer (implemented through dbt) then applies business logic and dimensional modeling on top of staging outputs, producing conformed entities and analytics-ready facts and dimensions as shown in the figure 3.4.

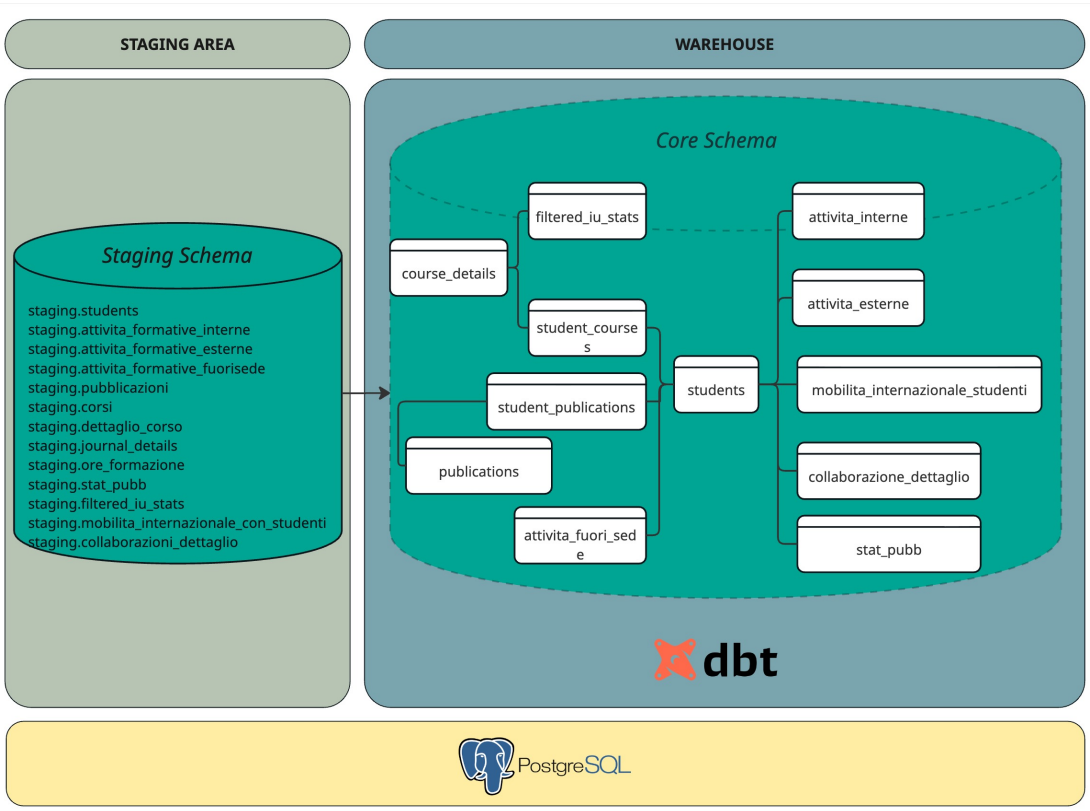


Figure 3.4: CSV Lane - Staging to Warehouse.

### 3.4.4 AI-Assisted Extraction for External PDFs

Semi-structured external PDFs (e.g., professional profiles) require a different strategy than CSV ingestion. The staging layer therefore supports an AI-assisted extraction pipeline composed of:

- **PDF parsing**: raw text is extracted using Python PDF utility libraries, preserving section order when possible and removing layout artifacts.
- **Local LLM inference**: a local large language model runtime (e.g., Ollama) executes deterministic, schema-aware prompts to extract structured attributes (e.g., most recent role, company, employment start date, and inferred employment length).
- **Privacy and reproducibility**: local inference avoids exposing personal data to external services and keeps the extraction process reproducible under containerized execution.

### 3.4.5 LinkedIn Probabilistic Extraction Layer: From Staging to Warehouse

In addition to deterministic extraction, the platform supports a *probabilistic extraction* pattern for employment information, as shown in the figure 3.5, where the model may output multiple candidates (e.g., company names, roles, dates) and their associated confidence signals. Instead of forcing premature decisions at extraction time, the staging layer stores:

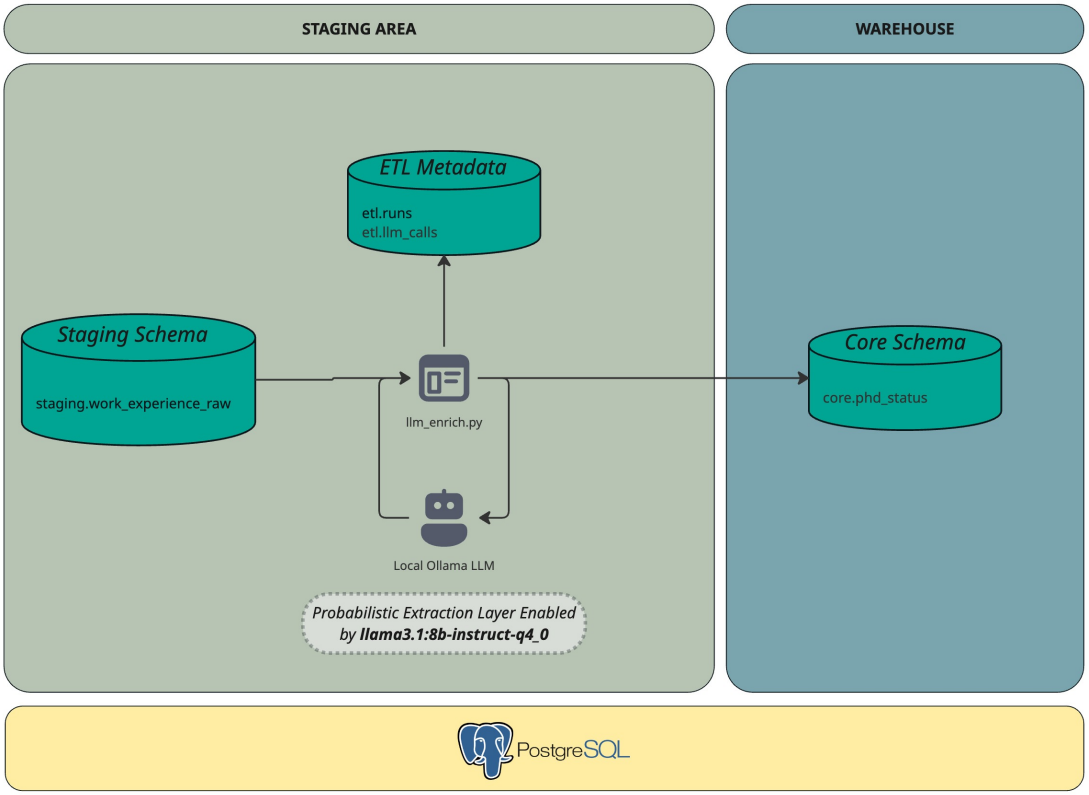


Figure 3.5: LinkedIn Probabilistic Extraction Layer – Staging to Warehouse.

- the **raw extracted spans** (verbatim text fragments),
- the **normalized candidates** (cleaned company/role/date fields),
- and **confidence/quality metadata** used to guide downstream selection.

Warehouse dbt models then consolidate these candidates into a single conformed employment snapshot per student, using transparent selection logic (e.g., highest-confidence candidate, tie-breaking by recency, and rule-based validation against allowed date ranges).

Overall, the staging layer acts as a quality firewall: it absorbs the variability of raw inputs (both structured and semi-structured), enforces standardized schemas

and integrity constraints, and exposes only validated, analytics-safe datasets for rigorous warehouse modeling.

## 3.5 Data Warehouse Layer with dbt

The data warehouse layer is implemented as a collection of dbt models materialized as tables. Each model is a deterministic SQL transformation of a staging source (defined via `source('staging', ...)`), with explicit cleaning rules, key logic, and deduplication strategy. This makes the warehouse auditable: for every dashboard metric, it is possible to identify (i) the exact model producing it, (ii) the staging table it reads from, (iii) the transformation rules applied, and (iv) the record-level grain at which the data is stored.

### 3.5.1 Canonical Student Entity

The model `students.sql` materializes the canonical student dimension. It reads from `source('staging','students')` and standardizes identifiers and profile fields by trimming whitespace, lowercasing email addresses, and converting hour/point measures into numeric values using comma-to-dot normalization. The primary key is `matricola_dottorando`. To ensure repeatable builds in the presence of multiple loads of the same student, the model applies a deduplication rule:

- rows are partitioned by `matricola_dottorando`,
- the most recent record is selected using `row_number()` ordered by `loaded_at desc`.

This establishes a stable, single-row-per-student representation used as a foreign key target by multiple fact-like models.

### 3.5.2 Internal Training Activities

The model `attivita_interne.sql` represents internal teaching/training activity events at record-level grain (one row per declared internal activity instance). It reads from `source('staging','attivita_formative_interne')` and performs the following transformations:

- **Normalization**: trims string fields and converts numeric measures (`ore`, `ore_riconosciute`, `coeff_voto`, `punti`) to `numeric` after replacing commas with decimal points; placeholder values such as `-` are coerced to null before casting.
- **Imputation policy**: numeric measures default to zero via `coalesce(...,0)` for hours, coefficient, and points, preventing null-propagation in downstream aggregations.

- **Date parsing**: `data_esame` is parsed only if it matches the pattern `DD/MM/YYYY`; otherwise it becomes null.
- **Derived field**: `anno` is computed as the extracted year from `data_esame`.

To ensure deterministic identity across runs, the model generates a surrogate key `id` using `dbt_utils.generate_surrogate_key` over the business-defining attributes (student, cycle, course, date, activity descriptors, and measures). Reload duplicates are resolved by retaining the latest record per `id` based on `loaded_at`.

### 3.5.3 External Training Activities

The model `attivita_esterne.sql` represents externally recognized activities, materialized at event grain. It reads from `source('staging','attivita_formative_esterne')` and applies strict cleaning:

- text fields are trimmed and empty strings are coerced to null;
- hour measures (`ore_dichiarate`, `ore_riconosciute`, `ore_calcolate`) and point/coefficient measures are cast to numeric with comma-to-dot normalization;
- `data_attivita` and `data_convalida` are parsed only if they match `DD/MM/YYYY` (including single-digit day/month).

An `id` surrogate key is built from the full set of defining attributes (including both dates and measures) to guarantee deterministic record identity. The model then deduplicates by selecting the most recent `loaded_at` per `id`. The resulting table is a clean fact-like dataset keyed by (`id`) with foreign key `matricola` intended to link to `students.matricola_dottorando`.

### 3.5.4 Off-site Activities

The model `attivita_fuori_sede.sql` captures off-site activities, which are structurally different from hour-based training events because they include geographic and administrative details (`luogo`, `ente`, `periodo`) and multiple administrative dates. It reads from `source('staging','attivita_formative_fuorisede')` and:

- normalizes descriptive fields by trimming and nulling empty strings;
- parses administrative dates via `to_date(...,'DD/MM/YYYY')` as best-effort (invalid formats become null);
- generates a deterministic surrogate key `id` across student/cycle/activity descriptors and the full set of dates.

Reload duplicates are resolved by retaining the latest record per `id` (highest `loaded_at`).

### 3.5.5 Collaboration Details

The model `collaborazioni_dettaglio.sql` stores collaboration activities at record-level grain (one row per collaboration activity instance). It reads from `source('staging','collabor` and standardizes:

- identifiers and labels (`matricola`, `ciclo`, `tutor`, `tipo_attivita`, `materia`, `docente`, `corso_di_laurea`) via trimming and null coercion;
- `ore` into `numeric(10,2)` with comma-to-dot normalization.

A surrogate key `id` is generated from the student identifier, cycle, tutor, activity descriptors, and hours. The table is deduplicated by keeping the most recent load per `id`. The `matricola` field is designed to reference `students.matricola_dottorando`.

### 3.5.6 Course Reference Model

The model `course_details.sql` is a reference/lookup table for courses, keyed by the natural compound key (`cod_ins`, `anno`). It reads from `source('staging','dettaglio_corso')` and enforces:

- year parsing only when `anno` matches a four-digit pattern;
- numeric casting of teaching-hour components (`h_les`, `h_ex`, `h_lab`, `h_tut`) with comma-to-dot normalization;
- integer parsing of `years_teaching` where valid.

If duplicate rows arrive for the same (`cod_ins`, `anno`), the model keeps the latest by `loaded_at`. This ensures a stable course reference layer for downstream joins.

### 3.5.7 Student–Course Participation

The model `student_courses.sql` represents enrollment-level participation, reading from `source('staging','corsi')`. It normalizes identifiers and descriptive attributes and parses `anno` only when it is a valid four-digit year. Unlike other models, it does not implement surrogate keys or deduplication; it therefore preserves the staging grain as-is. The output contains student identifiers (`matricola`) and course identifiers (`cod_ins`, `anno`), enabling course enrollment and exam-status analysis through the `stato` field.

### 3.5.8 Filtered IU Statistics for Dashboards

The model `filtered_iu_stats.sql` is a dashboard-oriented table designed to support multi-year course statistics in a wide format (separate columns per year). It reads from `source('staging','filtered_iu_stats')` and applies strict integer parsing for count metrics (e.g., `iscritti_2025`, `superati_2025`, and IIS variants).

An `id` surrogate key is generated from (`cod_ins`, `nome_insegnamento`, `docente`), and duplicates are resolved by selecting the latest `loaded_at` per `id`. The `cod_ins` field forms a loose link to `course_details.cod_ins`.

### 3.5.9 Publication Facts

The model `publications.sql` materializes publication facts at publication grain, using `titolo` as the primary key. It reads from `source('staging','pubblicazioni')` and:

- parses `anno` only if it matches a four-digit pattern;
- normalizes text fields via trimming and empty-to-null conversion;
- parses numeric indicators (`indicatore_r`, `errore_val`) using a numeric regex check and comma-to-dot casting;
- extracts a simplified first-author representation by splitting the `autori` string on `', '` into `autore_cognome` and `autore_nome`.

If duplicate publications with the same `titolo` are loaded, the model keeps the latest record by `loaded_at`. This yields a stable publication reference used by attribution models.

### 3.5.10 Publication Attribution

The model `student_publications.sql` maps students to publications at the grain (`matricola`, `titolo`). It reads from the same staging source `pubblicazioni` but extracts student-specific evaluation fields: `grado_proprieta_dottorandi`, `punteggio`, and `grado_proprieta`. Numeric fields tolerate commas and blanks and are cast to `numeric`. Duplicate pairs (`matricola`, `titolo`) are resolved by retaining the newest by `loaded_at`. The model is designed to function as a bridge table referencing `students.matricola_dottorando` and `publications.titolo`.

### 3.5.11 Journal Quartile Details

The model `journal_details.sql` normalizes journal-related metadata used for quartile analysis. It reads from `source('staging','journal_details')` and enforces:

- ISSN normalization by stripping non-alphanumeric characters (keeping digits and X) and reformatting 8-character ISSNs as `-`;
- year parsing (`anno`) as a four-digit integer;
- quartile parsing accepting both `Q1..Q4` and `1..4`.

A surrogate key `id` is computed from the tutor/author identity fields and journal attributes to provide a deterministic row identifier. Duplicates are resolved by latest

loaded_at. The model is used to compute quartile distributions in publication dashboards.

### 3.5.12 Publication Statistics

The model stat_pubb.sql stores aggregated publication counts per person (as provided by the staging dataset), including counts of journals, conferences, chapters, posters, abstracts, and patents. It reads from source('staging','stat_pubb') and casts each count to integer only when strictly numeric. A surrogate key id is generated over (matricola, ciclo) and the set of counts, and duplicates are resolved by selecting the latest loaded_at. This model supports tutor- and cycle-level performance summaries without recomputing counts from publication-level facts.

### 3.5.13 International Mobility

The model mobilita_internazionale_con_studenti.sql represents mobility events and includes a specific reconciliation strategy to handle incomplete identifiers. The staging source is known to provide matricola inconsistently; therefore the model:

- parses and normalizes descriptive fields (destination, period, type) and numeric duration/year fields;
- parses authorization/payment dates only when they match DD/MM/YYYY;
- constructs normalized name keys (cognome_norm, nome_norm) by lowercasing and collapsing whitespace;
- attempts to fill missing matricola by joining to students on normalized name *only when the name maps to exactly one student.*

The number of possible matches is stored in matched_students_count, making ambiguity explicit instead of silently assigning potentially wrong identifiers. A surrogate key id is generated across the full mobility descriptor set and used for deduplication by loaded_at. This design preserves auditability: when a mobility record lacks matricola, the reason and matching outcome remain visible.

### 3.5.14 Training Hours Summary

The model ore_formazione.sql stores per-student summary measures of training hours (ore_soft_skill, ore_hard_skill, ore_totali) as provided by the institutional system. It reads from source('staging','ore_formazione'), converts measures to numeric with comma-to-dot normalization, and uses a surrogate key id built from (matricola, ciclo, tutor) and the hour totals. Duplicates are resolved by latest loaded_at. This model supports fast dashboard queries where totals are required without recomputing from event-level activities.

### 3.5.15 Auditability Guarantees

Across the warehouse layer, auditability is implemented through three recurring patterns in the SQL models:

- **Deterministic cleaning rules**: every string normalization, cast, and date parsing rule is encoded in SQL, not in dashboards.
- **Deterministic identifiers**: event-level models generate surrogate keys from business-defining attributes, ensuring that the same logical record is assigned the same identifier across rebuilds.
- **Controlled deduplication**: when duplicate loads occur, the resolution policy is explicit (`keep latest by loaded_at`), preventing silent drift.

These design choices ensure that the warehouse can be rebuilt reproducibly and that all analytical results are traceable to the staged source tables and the exact transformation logic applied.

## 3.6 Data Mart Layer: Feature Engineering and Optimization

On top of the core warehouse schemas, dbt produces a set of *data marts*: domain-specific, analytics-ready tables engineered to match dashboard access patterns as shown in the figure 3.6. Unlike warehouse models—which preserve record-level detail and enforce canonical cleaning rules—marts intentionally restructure the data for fast filtering, grouping, and drill-downs, so that dashboards do not need to embed complex SQL logic.

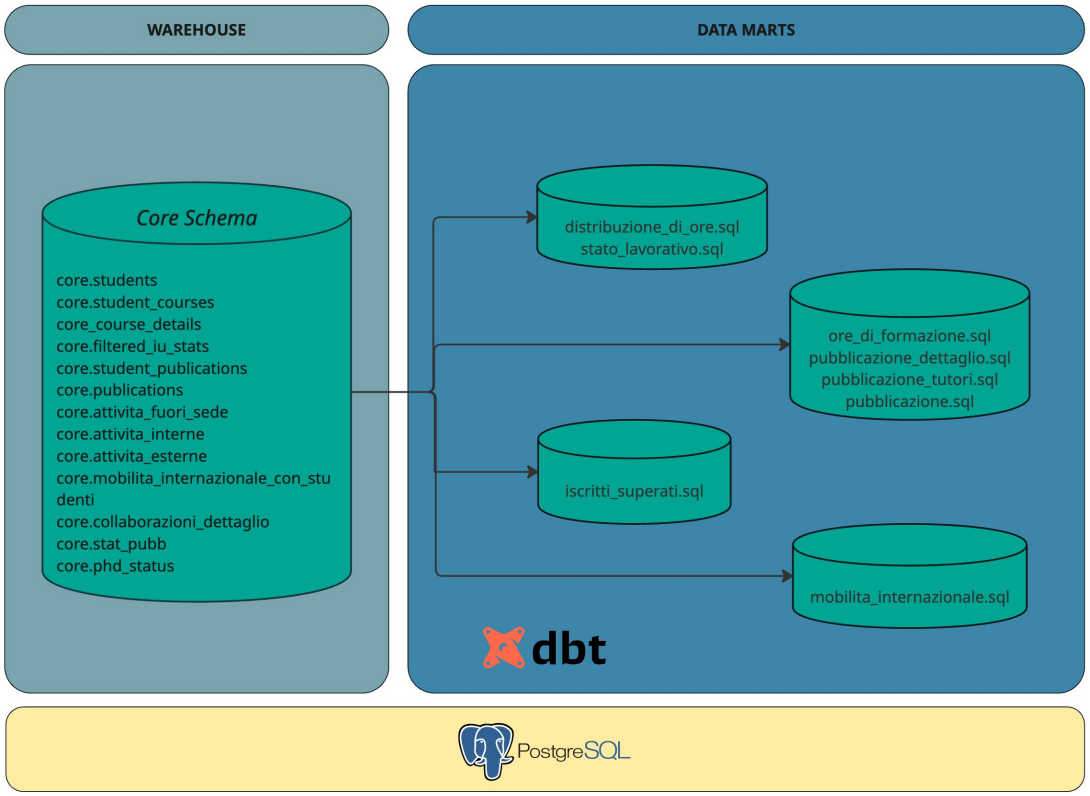


Figure 3.6: Warehouse to Data Marts.

The marts are designed explicitly around the analytical questions posed by the Academic Board and are organized by domain:

- **Teaching Activity Mart**: provides yearly and cumulative summaries of teaching/training hours per student and per cycle, with associated course-level details to enable drill-down from aggregated views to individual teaching records.
- **Publication Mart**: encodes publication counts and breakdowns by type and year, and supports grouping by student and tutor. Where available, it also includes journal-based stratification (e.g., quartile distributions) to monitor output quality and positioning.
- **Mobility Mart**: represents mobility activity both at event level (destinations and periods) and as engineered features (total duration, per-year duration, and cycle-level aggregates) to support distribution analyses and strategic monitoring.
- **Course and Exam Mart**: tracks engagement in the didactic offer by summarizing enrollments and exams passed per course per academic year, enabling cohort-level comparison of participation and outcomes.

By concentrating feature engineering in marts, the platform ensures that governance dashboards remain thin clients: they query stable, purpose-built tables rather than duplicating transformation logic in each visualization panel.

## 3.7 Data Serving Layer: Grafana Dashboards

The data serving layer exposes warehouse and mart outputs through Grafana dashboards connected directly to the PostgreSQL database. Grafana is used as the interactive analytical frontend, providing a visual environment to explore metrics through bar charts, tables, stacked visualizations, and drill-down panels. The dashboards are built to answer the Academic Board's questions with minimal user friction: users should be able to filter cohorts, inspect individual cases, and move from aggregate to detail without writing queries.

The key capabilities implemented in the dashboards include:

- **Yearly and cumulative teaching views**: interactive visualizations of hours with cycle selection implemented via checkboxes, enabling fast comparison across cohorts.
- **Drill-down exploration**: clicking on a student bar reveals the detailed breakdown of course participation and hours that generated the aggregate value.
- **Qualification tracking**: indicators and summaries highlight whether individual students satisfy program requirements (e.g., minimum hard-skill hours from program-offered courses), allowing rapid identification of critical cases.
- **Publication analytics**: panels display publications per student and journal quartile distributions, with drill-down to publication-level details and additional monitoring of tutor performance where required.
- **Course performance**: each row represents a course in a given year, with comparative bars for enrollment volume and exams passed, enabling immediate assessment of engagement and outcomes.
- **International mobility analysis**: dashboards summarize mobility durations, destinations, and duration distributions, supporting both compliance monitoring and strategic internationalization planning.

Grafana variables and interactive parameters allow users to dynamically select cycles, years, and subsets of students. This enables academic governance users to slice the data according to operational needs while keeping the transformation logic centralized in dbt models and marts, rather than duplicated across dashboard queries.

## 3.8 Containerization and Deployment

To ensure reproducibility, portability, and controlled execution, the entire platform is deployed as a containerized stack using Docker. All components of the system are defined and orchestrated through a single `docker-compose.yml` file, which specifies the services required to run the platform end to end. The stack includes the following containers:

- **PostgreSQL database**: persistent storage for staging schemas, warehouse models, and data marts.
- **Pipeline execution service**: responsible for data ingestion, validation, staging transformations, and AI-assisted enrichment.
- **dbt service**: executes warehouse and mart models and runs data quality tests.
- **Ollama LLM runtime**: provides a local large language model used for deterministic and probabilistic extraction from semi-structured PDF documents.
- **Grafana service**: hosts interactive dashboards connected directly to the PostgreSQL database.

Each service runs in an isolated container based on Ubuntu-derived images, with configuration managed through environment variables that define credentials, connection strings, and runtime parameters. This setup ensures that the full analytical stack can be started, stopped, updated, or destroyed consistently across environments, avoiding configuration drift between development, testing, and execution contexts.

## 3.9 Execution and Workflow

The operational workflow of the system follows a deterministic sequence of steps, executed either during initial setup or when new data becomes available:

1. **Environment setup**: Docker and Docker Compose are installed, required environment variables are configured, and directory structures for input CSV and PDF data are prepared.
2. **Database initialization**: the pipeline container executes bootstrap and migration scripts to create schemas and prepare the PostgreSQL database for ingestion and transformation.
3. **Data ingestion**: structured CSV files and semi-structured PDF documents are ingested, parsed, and loaded into staging schemas.
4. **LLM-based enrichment**: the Ollama service initializes the required language model, and the pipeline container runs enrichment routines to extract structured attributes from semi-structured PDF content.

5. **dbt execution**: dbt commands are executed to build warehouse models and data marts, with automated tests validating schema integrity and data quality.
6. **Dashboard provisioning**: Grafana dashboards are imported and configured to connect to the PostgreSQL data source.
7. **Exploration and analysis**: end users interact with dashboards to analyze teaching activities, course participation, publication outputs, and international mobility patterns.

The workflow is designed to be repeatable and idempotent, supporting periodic re-execution (e.g., annually or upon data refresh) without manual reconfiguration.

## 3.10 Summary

The proposed methodology integrates structured data engineering practices with targeted AI-assisted enrichment and interactive visualization to deliver a comprehensive platform for academic analytics. By organizing ingestion, transformation, and serving into clearly defined architectural layers and deploying the system as a fully containerized stack, the approach ensures reproducibility, data quality, and analytical flexibility. The platform directly addresses the analytical requirements articulated by the Academic Board—such as drill-down analysis of teaching activities, publication performance monitoring, and international mobility tracking—while providing a robust foundation that can be extended to additional analytical use cases beyond the initial validation scenario. “‘latex

# Chapter 4

# Results

This chapter presents the results obtained from the implementation of the proposed data platform. Since the objective of the work is not the development of new analytical models but the realization of a complete data integration and visualization infrastructure, results are evaluated in terms of: (i) the quality and consistency of the integrated data obtained after harmonization, (ii) the analytical views enabled by the database, and (iii) the dashboards that operationally support monitoring and decision-making activities. The chapter therefore focuses on observable outcomes produced by the system rather than on implementation details already discussed in the Methodology.

## 4.1 Intermediate Results: Data Integration and Harmonization Outcomes

### 4.1.1 Observed Issues in Raw Institutional Data

The first set of results emerges during the ingestion of raw institutional exports. Although the data is provided in structured CSV format, direct loading into a relational database reveals several inconsistencies that prevent immediate analytical use.

A primary issue concerns student identification. Some datasets identify students uniquely through a *matricola*, while others rely solely on name and surname fields. In addition, variations in capitalization, spacing, and diacritics are present across files. When loaded without preprocessing, this leads to duplicated student entities and broken joins across datasets.

A second issue relates to schema heterogeneity. The same semantic concept is represented using different column names across files (e.g., course codes, activity types), and some files contain placeholder or unused columns. This prevents straightforward joins and requires explicit schema alignment before integration.

A third issue concerns format inconsistency, particularly in delimiters and dates. Institutional exports use semicolons, commas, and tab delimiters interchangeably, while date fields appear in multiple formats or are embedded in textual descriptions. These inconsistencies directly affect the ability to perform year-based aggregations.

Finally, there is a granularity mismatch across datasets. Some files provide one row per student, others one row per event (e.g., course activity, mobility period), and others only aggregated statistics. Without normalization, these differences lead to ambiguity when computing totals or comparisons.

These issues constitute the "raw state" of the data and represent the baseline against which subsequent improvements can be evaluated.

### 4.1.2 Results of Harmonization and Integration

After passing through the staging and transformation layers, the data exhibits a substantially different structure. The most relevant observable outcomes are:

- A single canonical student dimension, uniquely identified by *matricola* and consistently referenced by all fact tables.
- Normalized representations of courses, publications, mobility events, and training activities, each enabling stable joins across datasets.
- Explicit year attributes derived from heterogeneous date representations, enabling consistent yearly and cumulative analysis.
- Removal of duplicated or orphan records that were present in the raw exports.

The contrast between the initial and final database states is significant. In the initial state, analytical queries require ad-hoc parsing and string manipulation, often producing ambiguous results. In the final state, queries rely on explicit keys and precomputed attributes, reducing complexity and increasing reliability.

This transformation is a result in itself: the platform converts a collection of loosely related files into an analytics-ready relational model. Importantly, this outcome is observable and verifiable through database inspection and query behavior, not inferred indirectly.

## 4.2 Database-Level Results: Analytical Views and Query Stability

Once harmonized, the integrated PostgreSQL database supports two classes of queries: validation queries and analytical queries.

Validation queries confirm the internal consistency of the dataset, such as verifying that each student belongs to exactly one cycle or that all mobility events reference a valid student. The absence of integrity violations after transformation confirms that harmonization steps were effective.

More importantly for this chapter, the database supports stable analytical queries. Queries for yearly totals, cumulative metrics, and filtered aggregations can be expressed concisely and without embedding data-cleaning logic. This stability is a key indicator of result quality: the complexity of handling heterogeneous inputs has been absorbed upstream, leaving dashboards to focus exclusively on analysis.

## 4.3 Dashboard Results

The final and most visible results of the platform are the Grafana dashboards. These dashboards are not evaluated aesthetically, but functionally: each one corresponds to a specific monitoring requirement and demonstrates that the underlying data supports the intended analytical questions. For clarity, each dashboard category is discussed separately. Screenshot placeholders are included to facilitate integration in the final document, having preserved the privacy of the students by covering their student number, name and surname to prevent them from being identifiable.

### 4.3.1 Teaching and Training Hours Distribution

**Dashboard:** `Distribuzine delle ore e Stato Lavorativo`

This dashboard, presented in figure 4.1 provides yearly, cycle-based, and cumulative views of teaching and training hours performed by PhD students. Its primary result is enabling visibility into workload distribution over time. Once the data is integrated, the dashboard allows users to:

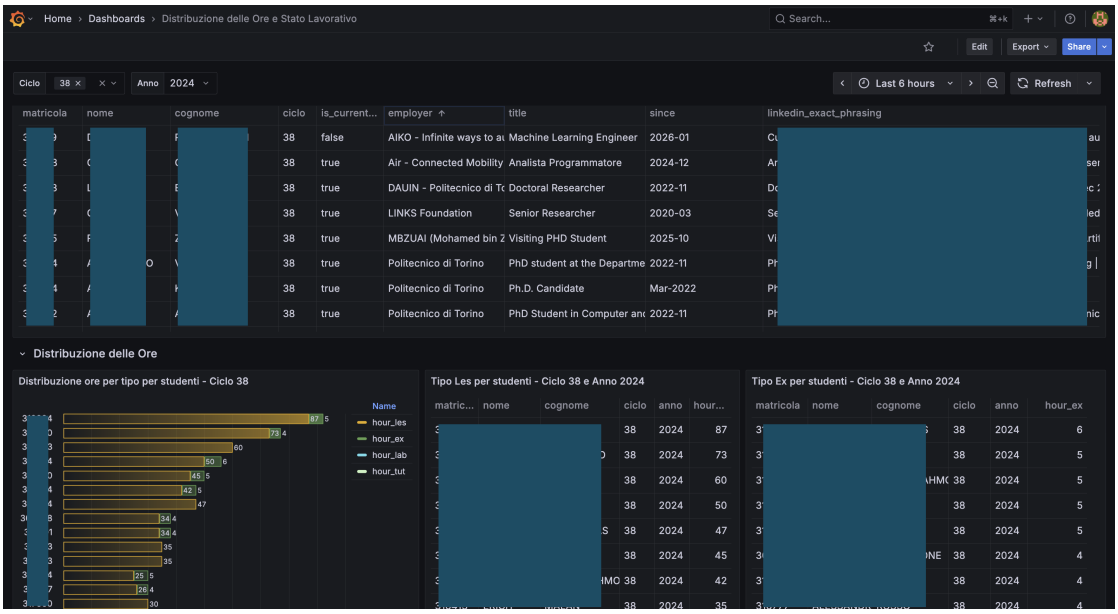


Figure 4.1: Grafana dashboard `Distribuzine delle ore e Stato Lavorativo`

- filter students by active cycles using dashboard variables,
- switch between yearly and cumulative views,
- identify students with low or high engagement,
- drill down from aggregated bars to detailed course-level contributions.

The key result is that the dashboard operates on integrated, harmonized tables: filters and drill-downs remain consistent across cycles and years without requiring manual preprocessing of raw exports.

### 4.3.2 Course-Level Enrollment and Exam Outcomes

**Dashboard:** `Iscritti e Superati`

Figure 4.2 represents a snapshot of the dashboard *iscritti e superati* that visualizes courses offered in a given year, with bars indicating the number of enrolled students and the number of exams passed.

The result is the ability to compare engagement and outcomes across courses using a single, consistent view. The dashboard also can be customized by the courses to be included, as well as the years, having a clear comparison among the courses between the different years. This highlights patterns that were not easily

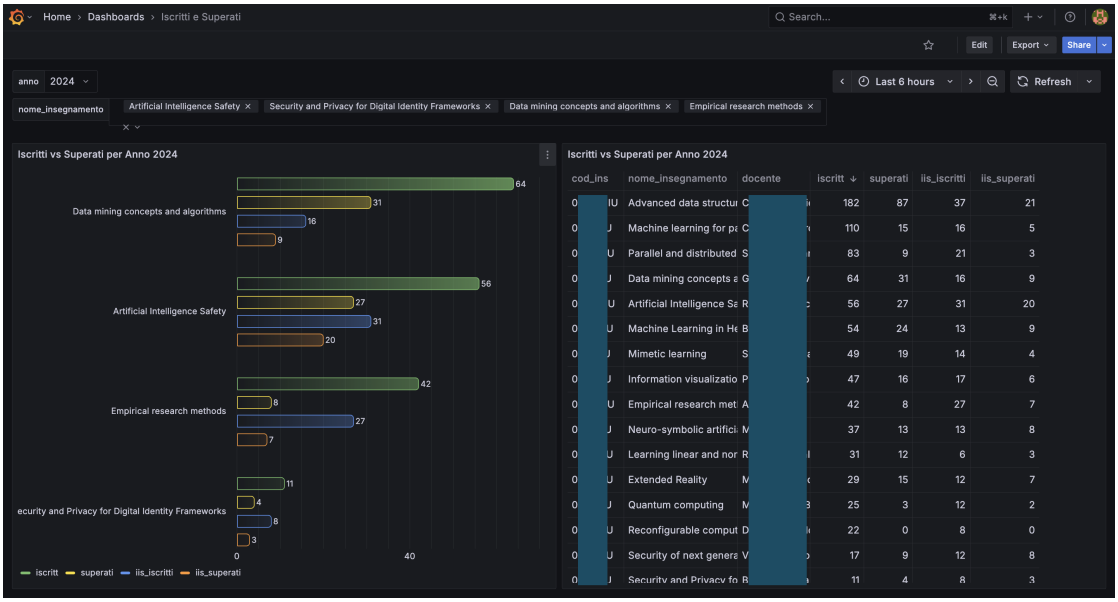


Figure 4.2: Grafana dashboard `Iscritti e Superati`

observable in raw exports, such as courses with high enrollment but low completion rates, supporting data-driven evaluation of course structure and scheduling.

### 4.3.3 International Mobility Analysis

**Dashboard:** `Mobilita Internazionale`

International mobility is analyzed through dashboards, as presented in the figure 4.3, showing total duration abroad per student, destination distributions, and duration-based filters (figure 4.3).

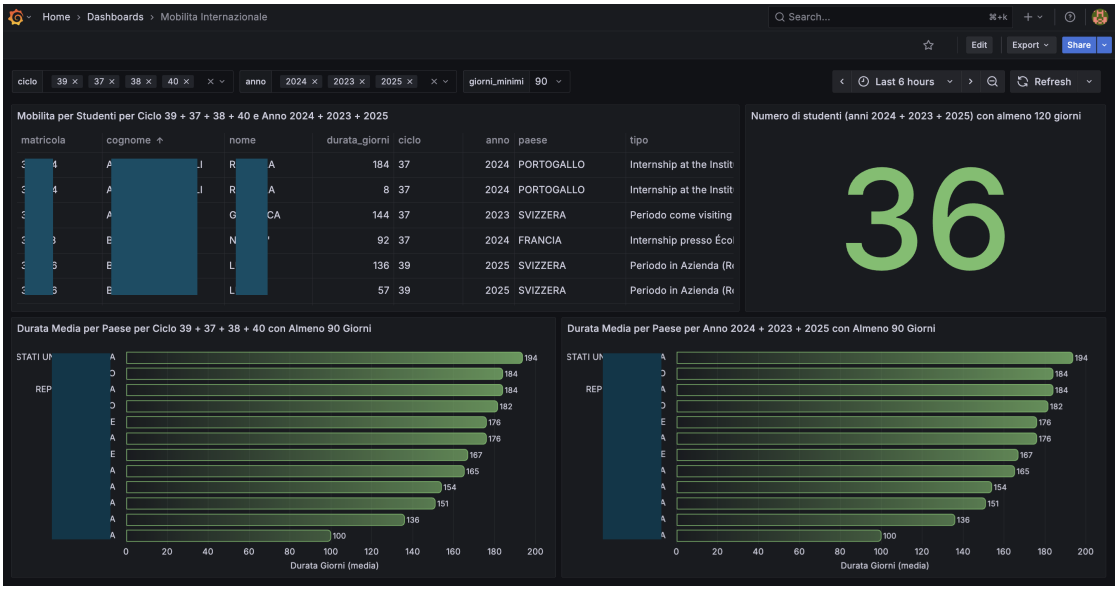


Figure 4.3: Grafana dashboard `Mobilita Internazionale`

The result is not merely visualization but operational support for strategic reasoning: identifying common destinations, typical durations, and disparities across cohorts.

### 4.3.4 Training Hours and Program Requirements

**Dashboard:** `Ore Formazione`

This dashboard consolidates soft skill and hard skill hours and supports verification of program requirements, such as minimum hard skill hours obtained from program-offered courses (figure 4.4). This transforms a typically manual verification task into a reproducible analytical control, where compliance indicators can be derived consistently from the integrated database.

### 4.3.5 Publications Analysis

**Dashboard:** `Pubblicazioni`

Publication dashboards, as shown in figure 4.5, support both student-level and tutor-level monitoring, including breakdowns by journal quartile and drill-downs


Figure 4.4: Grafana dashboard `Ore Formazione`

to publication metadata. The result is the ability to assess publication output from multiple perspectives using a unified dataset, which was not feasible with the original fragmented exports.

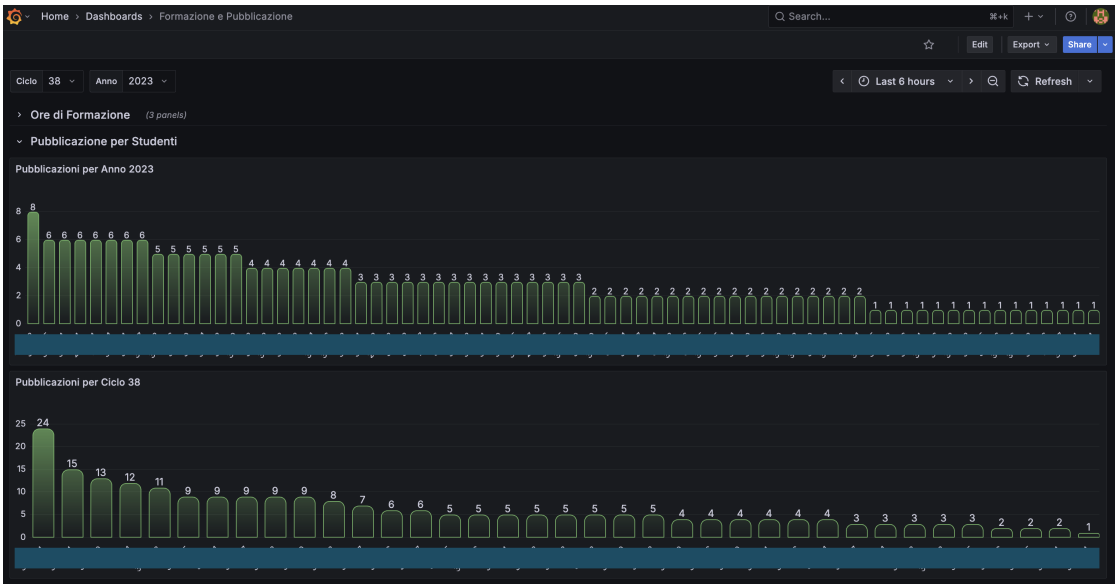


Figure 4.5: Grafana dashboard `Pubblicazioni`

## 4.4 Summary of Results

The results demonstrate that the platform successfully transforms heterogeneous institutional and external data into a coherent analytical system. The main outcomes are:

- resolution of identifier, schema, and format mismatches present in raw data;
- creation of a stable relational model suitable for analytical querying;
- implementation of dashboards that directly support monitoring activities without manual data manipulation.

Rather than optimizing numerical performance metrics, the results show that the platform enables forms of analysis that were previously impractical or unreliable, fulfilling its role as a decision-support infrastructure.

# Chapter 5

# Conclusion

This thesis presented the design and implementation of a containerized analytics platform that transforms heterogeneous academic data into a coherent decision-support environment. The work addressed a practical institutional need: enabling consistent monitoring of doctoral students' activities and outcomes through integrated data storage and interactive dashboards. The contribution is both architectural and operational: a reproducible data pipeline that ingests structured institutional exports, harmonizes inconsistencies, enriches semi-structured data from PDF documents using LLM-assisted extraction, and exposes results through a dashboard layer supporting governance and evaluation workflows.

## 5.1 Major Outcomes and Milestones

A first milestone of the project was the formalization of the end-to-end data flow—from input files to dashboards—into a modular architecture. In many academic environments, data analysis is often performed through one-off spreadsheets and manual reconciliations. This approach is fragile: results are hard to reproduce, data cleaning is implicit and undocumented, and changes in input schemas can break downstream analyses. By adopting a pipeline-based approach and anchoring integration into a relational database, this project established a stable foundation in which transformation logic is explicit and repeatable. This alone is a meaningful result: the platform creates continuity across years and cohorts, which is critical when monitoring PhD programs whose evaluation spans multiple cycles.

A second milestone was the harmonization of heterogeneous institutional sources into a consistent analytical model. The work demonstrated that *structured data* does not necessarily mean *integrable data*. Differences in identifiers (matricola versus name-based records), inconsistent delimiters, schema mismatches, and varying levels of granularity make direct analysis unreliable and, in practice, infeasible. The platform resolves these issues upstream by producing clean join keys, standardized

attributes (year, cycle, course codes), and stable relational representations. This significantly reduces the cognitive and operational burden on academic committees: dashboard interpretation becomes the primary activity, rather than manual troubleshooting of raw exports.

A third milestone was the deployment of interactive dashboards that implement concrete governance questions. The dashboards were not designed as generic reports, but as analytical instruments that support drill-down analysis (e.g., selecting an individual student to inspect course-level teaching hours) and multi-level aggregation (per year, per cycle, and cumulative). This design improves transparency: committees are not only presented with aggregate totals, but can directly inspect how those totals are composed. This is essential for trust and for actionable decision-making, particularly in discussions involving workload balance, compliance with training requirements, or mobility engagement.

A fourth milestone was the introduction of AI-assisted data enrichment into the pipeline. The platform integrates a local LLM runtime, `llama3.1:8b-instruct-q4_0`, to extract structured information from semi-structured documents. The key contribution is not the use of LLMs per se, but their integration into a controlled data engineering workflow. LLM output is treated as intermediate data that must be validated, normalized, and stored with traceability, rather than as a final authoritative result. This establishes a pragmatic pattern for institutional analytics: LLMs can extend analytical coverage where structured sources are incomplete, without displacing established data management principles.

## 5.2 Advantages and Potential Implications

The primary advantage of this pipeline-first, dashboard-first approach is that it converts a collection of disconnected datasets into a maintained institutional asset. Once established, the database schema and dashboard layer allow analyses to evolve without redoing the integration effort each time new data arrives. This is particularly relevant for long-running programs such as PhD courses, where decisions rely on multi-year patterns rather than isolated snapshots.

A second advantage is improved reproducibility and accountability. When a committee decision relies on a dashboard indicator, it becomes possible to trace that indicator back to a specific query and a specific set of transformed tables. This represents a clear departure from spreadsheet-driven reporting, where transformations are often hidden, versioning is informal, and results may not be reconstructible after the fact.

A third advantage lies in extensibility. The platform is not intrinsically tied to a single metric or analytical question. Once ingestion, storage, and visualization layers are in place, new indicators can be introduced by adding transformation logic and dashboard panels. This is the practical meaning of generality in this thesis:

the system is generalizable at the methodological level, as it supports repeated analytical patterns—ingest, reconcile, enrich, model, visualize—that recur across institutional analytics problems.

Finally, integrating LLM-based enrichment into the pipeline enables institutional analytics to incorporate semi-structured evidence that would otherwise remain unused. In many governance contexts, relevant information exists primarily in documents rather than databases. The ability to interpret and structure this information—under privacy-preserving and controlled execution conditions—opens the door to richer analytics, provided that reliability and validation are handled rigorously.

## 5.3 Limitations

The platform is constrained by the nature of its source data. Institutional exports reflect what upstream systems record; missing fields cannot be reconstructed without additional sources. Furthermore, LLM-assisted extraction from PDF documents inherits limitations typical of semi-structured inputs, including variability in formatting, incomplete timestamps, and ambiguity in interpreting employment histories. While local model execution improves privacy and operational control, it does not guarantee correctness. For these reasons, enriched fields should be treated as complementary signals rather than authoritative ground truth, particularly when derived from public or self-reported profiles.

Another limitation is operational. Although the pipeline is containerized and reproducible, execution remains largely batch-oriented. In a production environment, a more robust orchestration and monitoring layer would be required to support scheduling, retries, incremental updates, and automated alerting.

## 5.4 Future Work

Several extensions would substantially strengthen the platform and move it toward a more mature analytics service.

### 5.4.1 Automating Employment Status Acquisition

A clear direction for future work is improving the acquisition of up-to-date professional status information. At present, external career data are collected through manual PDF snapshots of professional profiles and processed via an LLM-based extraction component. A more scalable approach would involve partial automation of data acquisition, provided that it remains compliant with legal, ethical, and governance constraints. Possible directions include:

- building a controlled pipeline based on user-consented data, where graduates opt in to share updated employment information;
- integrating alternative open datasets or research-oriented labor market sources, where available;
- leveraging institutional alumni services and structured feedback loops to maintain employment records over time.

If profile-based acquisition remains relevant, an important methodological step would be to formalize extraction confidence and provenance, allowing dashboards to distinguish between verified, inferred, and missing values.

### 5.4.2 Integrating Workflow Orchestration

The current pipeline demonstrates the full analytical flow but would benefit from integration with orchestration frameworks such as Apache Airflow or Dagster. Such integration would enable scheduled ingestion cycles, incremental processing, explicit dependency management, retry logic, failure isolation, and automated notifications when validation checks fail. Orchestration would also clarify operational ownership by treating pipelines as managed assets with observable execution histories rather than ad hoc scripts.

### 5.4.3 Generative SQL and LLM-Based Querying

A promising extension involves enabling LLM-assisted querying of the integrated database through generative SQL. In this scenario, academic committee members could express analytical questions in natural language (e.g., "identify students with fewer than a given number of teaching hours in a specific year"), and the system would generate SQL queries against controlled views. The value of this approach lies not in replacing dashboards, but in supporting ad hoc exploration without requiring SQL expertise. This direction, however, requires strict safeguards, including constrained schemas, validation of generated queries, logging and auditing, and mechanisms to prevent ambiguous interpretations from producing misleading results.

### 5.4.4 Comparing Agent Models and Establishing Evaluation Protocols

The LLM-based enrichment component could be significantly strengthened through systematic evaluation. Future work could compare different agent strategies—prompting approaches, model sizes, structured output formats, and tool-augmented agents—using measurable criteria such as extraction accuracy for specific fields, robustness across

document layouts, consistency across repeated runs, latency, computational cost, and sensitivity to incomplete or ambiguous inputs. A rigorous evaluation framework would require labeled ground truth for a subset of documents and standardized metrics such as precision, recall, and error categorization. This would shift the enrichment component from being merely plausible and useful to being quantitatively validated.

## 5.5 Closing Remarks

This thesis demonstrates that effective institutional analytics depends less on advanced modeling techniques and more on reliable infrastructure. By integrating heterogeneous institutional exports into a coherent database, enriching selected dimensions through controlled LLM-assisted extraction, and exposing results through interactive dashboards, the proposed platform enables analyses that would otherwise require extensive manual and non-reproducible effort. The primary contribution is a practical framework for transforming dispersed academic data into accessible intelligence, supporting governance decisions with transparency and repeatability. The future directions outlined—automation of external status acquisition, workflow orchestration, generative SQL interfaces, and systematic agent evaluation—represent realistic and impactful pathways for evolving the platform into a robust decision-support system for academic program monitoring.

# Bibliography


[1] R. Kimball and M. Ross. *The Data Warehouse Toolkit*. United States: Wiley Computer Publishing, 2013. ISBN: 978-1118530801.

[2] William H. Inmon. *Building the Data Warehouse Fourth Edition*. United States: Wiley Computer Publishing, 2005. ISBN: 978-0764599446.

[3] OECD. *Education at a Glance*. Tech. rep. Organisation for Economic Co-operation and Development, 2019.

[4] Matt Housley Joe Reis. *Fundamentals of Data Engineering: Plan and Build Robust Data Systems*. United States: O'Reilly Media, 2022. ISBN: 978-1098108304.

[5] Wes McKinney. *Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter*. United States: O'Reilly Media, 2022. ISBN: 978-1098104030.

[6] Claus Pahl. «Containerization and the PaaS Cloud». In: *IEEE Cloud Computing* (2015).

[7] Martin Kleppmann. *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. United States: O'Reilly Media, 2017. ISBN: 978-1449373320.

[8] Ben K. Daniel. «Big Data and analytics in higher education: Opportunities and challenges». In: *British Journal of Educational Technology* (2014).

[9] Maurizio Lenzerini. «Data Integration: A Theoretical Perspective». In: *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. 2002.

[10] Yakov Shafranovich. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. Internet Engineering Task Force (IETF). 2020. URL: `https://datatracker.ietf.org/doc/rfc4180/`.

[11] Hong Hai Do Erhard Rahm. «Data Cleaning: Problems and Current Approaches». In: *DBLP* (2000).

[12] Phil Long George Siemens. «Penetrating the Fog: Analytics in Learning and Education». In: *Tecnologie Didattiche* (2011).

[13] Alan B. Sunter Ivan P. Fellegi. «A Theory for Record Linkage». In: *American Statistical Association* (2012).

[14] Greg Kemnitz Michael Stonebraker. «The POSTGRES next generation database management system». In: *Communications of the ACM* (1991).

[15] Sebastian Ventura Cristobal Romero. «Educational data mining and learning analytics: An updated survey». In: *WIREs Data Mining and Knowledge Discovery* (2020).

[16] S. Few. «Information Dashboard Design—Effective Visual Communication of Data». In: *O'Reilly Media, Newton* (2009).

[17] Grafana Labs. *Grafana*. `https://grafana.com/`. Accessed: 2026-02-02. 2025.

[18] Vadim Ogievetsky Jeffrey Heer Michael Bostock. «A tour through the visualization zoo». In: *Communications of the ACM* (2010).

[19] Katrien Verbert et al. «Learning dashboards: An overview and future research opportunities». In: *Personal and Ubiquitous Computing* (2013).

[20] Alexandra Olteanu et al. «Social Data: Biases, Methodological Pitfalls, and Ethical Boundaries». In: *Front Big Data* (2019).

[21] Ollama, Inc. *Ollama: Run Large Language Models Locally*. `https://ollama.com/`. 2024.

[22] Rishi Bommasani et al. «On the Opportunities and Risks of Foundation Models». In: *Machine Learning* (2022).

[23] Wayne Xin Zhao et al. «A Survey of Large Language Models». In: *Computation and Language* (2023).

[24] Shunyu Yao et al. «Tree of Thoughts: Deliberate Problem Solving with Large Language Models». In: *Computation and Language* (2023).

[25] dbt Labs. *dbt: Analytics Engineering for Transforming Data*. `https://www.getdbt.com/`. 2024.

[26] Dirk Merkel. «Docker: lightweight Linux containers for consistent development and deployment». In: *Linux Journal* (2014).

[27] Python Software Foundation. *Python Programming Language — Official Website*. `https://www.python.org/`. 2024.

[28] Fred L Drake Jr Guido Van Rossum. *An Introduction to Python*. United States: Network Theory Limited, 2017. ISBN: 978-1906966133.

[29] Docker, Inc. *Docker: Accelerated Container Application Development*. `https://www.docker.com/`. 2024.

[30] Dan Jurafsky and James H. Martin. *Speech and Language Processing*. 3rd ed. Draft edition. Prentice Hall, 2023. URL: `https://web.stanford.edu/~jurafsky/slp3/`.