Journal of Health Informatics

J Health Inform. 2026;18:e1885
DOI: https://doi.org/10.59681/2175-4411.v18.iEspecial.2026.1885

ARTIGO COMPLETO
CBIS 2026

MIMIC-IV Extractor Tool: improving clinical data access through a Python abstraction layer

MIMIC-IV Extractor Tool: melhoria do acesso a dados clínicos por meio de uma camada de abstração em Python

Willian de Vargas[1], Viviane Rodrigues Botelho[1], Ana Trindade Winck[1]

ORCID: 0000-0002-6703-4657 · 0000-0003-1864-5926 · 0000-0002-3320-112X

[1] Universidade Federal de Ciências da Saúde de Porto Alegre (UFCSPA), Porto Alegre (RS), Brasil.

ARTICLE INFORMATION

TYPE
Artigo completo · CBIS 2026

IDENTIFIER
e1885

DOI
10.59681/2175-4411.v18.iEspecial.2026.1885

RECEIVED
17 May 2026

ACCEPTED
20 July 2026

REVIEW PROCESS
Double Blind Review

HANDLING EDITOR
Juliano de Souza Gaspar

ACCESS
Open, CC BY 4.0

CORRESPONDING AUTHOR
Willian de Vargas
willianv@ufcspa.edu.br

LICENSE

This is an open access article published under the Creative Commons CC BY 4.0 license, which permits use, distribution, and reproduction in any medium, provided the original work is properly cited.

CC BY 4.0

ABSTRACT

Objective: To simplify the process of accessing and consolidating MIMIC-IV data, reducing the overhead of data engineering tasks, allowing researchers to direct their efforts toward clinical analysis. **Method:** A Python-based abstraction layer, the "MIMIC-IV Extractor Tool," was developed to encapsulate complex SQL queries into high-level methods that return DataFrames. The approach was validated through a practical use case. **Results:** The abstraction strategy effectively reduced the complexity of the data retrieval workflow, enabling the extraction of clinical information (such as age, gender, and diagnoses) without the need to directly manipulate SQL logic. **Conclusion:** This proof of concept validates the feasibility of using abstraction layers to facilitate data acquisition, allowing researchers to focus on clinical analysis and promoting advancements in biomedical data science.

Keywords: Data Science; Information Storage and Retrieval; Electronic Health Records.

RESUMO

Objetivo: Simplificar o processo de acesso e consolidação de dados da base MIMIC-IV, reduzindo o esforço dedicado às tarefas de engenharia de dados, permitindo que pesquisadores direcionem sua atenção para a análise clínica. **Método:** Desenvolveu-se uma camada de abstração em Python, implementada como a biblioteca de código aberto "MIMIC-IV Extractor Tool", que encapsula consultas SQL complexas em métodos de alto nível que retornam DataFrames. Validou-se a abordagem através de um caso de uso prático. **Resultados:** A estratégia de abstração reduziu efetivamente a complexidade do fluxo de trabalho de recuperação de dados, permitindo a extração de informações clínicas (como idade, gênero e diagnósticos) sem a necessidade de manipular lógica SQL diretamente. **Conclusão:** Esta prova de conceito valida a viabilidade de usar camadas de abstração para facilitar a aquisição de dados, permitindo que pesquisadores foquem na análise clínica e promovendo o avanço da ciência de dados biomédicos.

Descritores: Ciência de Dados; Armazenamento e Recuperação da Informação; Registros Eletrônicos de Saúde.

CBIS 2026 XXI CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE

Paper presented at the 21st Brazilian Congress of Health Informatics (CBIS 2026), held from September 23 to 25, 2026, in Brasília (DF), Brazil, and published in the CBIS 2026 section of the Journal of Health Informatics.

HOW TO CITE THIS ARTICLE: de Vargas W, Botelho VR, Winck AT. MIMIC-IV Extractor Tool: improving clinical data access through a Python abstraction layer. J Health Inform 2026;18:e1885. doi: https://doi.org/10.59681/2175-4411.v18.iEspecial.2026.1885

1 INTRODUCTION

Currently, a large amount of data, highly relevant to health systems, is generated through biomedical research. All this data requires proper management for its analysis to be viable [1]. In this context, the large amount and variability of data make it possible for researchers from the most diverse areas of study to use this information to conduct their research.

Computing at the service of medicine is a multidisciplinary field, capable of promoting computational models that make it possible to make inferences related to the health status of patients through the use of scientific computing methods in the analysis of biomedical data [2]. This field, despite being very promising, brings researchers challenges related to data access and manipulation, stemming from information security requirements and the inherent complexity of heterogeneous biomedical data structures.

Biomedical data have a high level of sensitivity, when we think about information security, implying the work of Ethics Committees together with researchers, making the research process more bureaucratic. On the other hand, some regulatory frameworks allow the use of publicly accessible and properly de-identified databases without requiring formal evaluation by an Ethics Committee. In Brazil, for example, Resolution 510/16, of the National Health Council, makes it feasible to use publicly accessible databases so that research can be conducted without the need for evaluation by an Ethics Committee [3].

Among the publicly available biomedical databases, the Medical Information Mart for Intensive Care IV (MIMIC-IV) stands out as one of the most widely used resources in clinical research. MIMIC-IV is a public biomedical database containing de-identified information from patients admitted to the Beth Israel Deaconess Medical Center in Boston, Massachusetts, between 2008 and 2019 [4]. The database follows a modular organization composed of hospital-wide clinical data, intensive care unit data, and clinical notes, containing information such as laboratory tests, medications, diagnoses, vital signs, and free-text medical notes [4].

Although MIMIC-IV provides a rich source of clinical information, its modular structure and large data volume create substantial technical barriers for researchers without advanced knowledge of Structured Query Language (SQL) and data engineering. Accessing relevant information often requires complex joins between multiple tables, application of business rules, and familiarity with database structures.

In heterogeneous environments, extract, transform, load (ETL) processes are commonly used to integrate data from different sources while maintaining consistency and quality [5]. In parallel, cloud computing technologies have enabled scalable storage and processing infrastructures, reducing local infrastructure costs and facilitating large-scale biomedical analysis [6-8]. Despite these advances, researchers still face difficulties related to technical implementation and data manipulation workflows.

In this context, Python has become one of the most widely adopted programming languages in scientific computing due to its syntactic simplicity, open-source ecosystem, and extensive support for data analysis libraries [9,10]. The popularity of Python in the scientific community creates an opportunity for the development of tools capable of abstracting technical database operations and simplifying access to biomedical information.

Several recent studies have explored the use of MIMIC-IV for clinical prediction and retrospective analyses. Previous works investigated mortality prediction in patients with bone tumors [11], early prediction of sepsis-associated encephalopathy using machine learning models [12], and the impact of beta-blockers on mortality in critically ill oncology patients [13]. These studies reinforce the scientific relevance of MIMIC-IV while also highlighting the technical complexity involved in extracting and consolidating its data.

Therefore, there is a demand for tools capable of reducing the engineering overhead involved in accessing and consolidating MIMIC-IV data, allowing multidisciplinary research teams to operate more efficiently regardless of their primary area of expertise. In this context, this work proposes the development of a Python library as a proof of concept (PoC) aimed at demonstrating the feasibility of simplifying access to MIMIC-IV through high-level abstraction methods that encapsulate SQL logic and return data structures directly usable in scientific workflows. The time and effort demanded by data engineering tasks represent a real cost to scientific productivity, given that such resources could otherwise be directed toward clinical analysis and discovery. Abstraction layers address this imbalance directly, serving as a strategy of efficiency for multidisciplinary research teams.

2 METHODS

This section describes the methodology adopted for the development of this work.

2.1 Accessing MIMIC-IV data

Access to the data contained in MIMIC-IV is conditioned on the completion of a training course in research with human participants, typically provided by the Collaborative Institutional Training Initiative (CITI). This training addresses the protection of human research participants and includes the requirements of the Health Insurance Portability and Accountability

Act (HIPAA), which establishes the standards for information protection that health organizations in the United States must follow.

After completing the course and obtaining the completion certificate, the researcher must submit the documentation on the PhysioNet platform and sign a Data Use Agreement (DUA). The DUA is a mandatory legal contract that requires users to adequately safeguard the dataset, strictly prohibits any attempts to re-identify individuals, and forbids the sharing of the data.

2.2 Preparation of the development environment

The main objective of this work is to develop a PoC tool to optimize the data consolidation workflow in MIMIC-IV, reducing the time and effort spent on data engineering tasks. Rather than delivering a production-ready solution, this work focuses on demonstrating the feasibility of reducing the complexity of data access through abstraction mechanisms. Taking advantage of the widespread use of the Python programming language in the scientific field, due to its syntactic simplicity and extensive library ecosystem, the proposed approach was implemented as a Python library.

For this purpose, a dedicated development environment was set up. The strategies for local database management and the technical procedures for data import followed a practical guide focused on the access and retrieval of MIMIC-IV biomedical data [14].

The development of the library was conducted using the open-source Integrated Development Environment (IDE) Visual Studio Code, making use of specific plugins to support Python development. Additionally, DBeaver was adopted as a local SQL client to facilitate testing and data exploration during the development process.

For development, libraries were incorporated into the work to establish the connection between the Python library and the locally hosted MIMIC-IV database. In this context, the SQLAlchemy library was used to manage database connections and execute queries within a local environment.

2.3 Development and publication of the library

It is important to highlight that the implementation of the library was conducted as a PoC, with the primary objective of validating the proposed approach. The focus was not on delivering a fully optimized or production-ready system, but rather on demonstrating that the abstraction of complex SQL queries into Python methods is a viable strategy to simplify data access in MIMIC-IV.

The library is composed of SQL queries that are encapsulated by Python methods. The queries are responsible for retrieving and consolidating the MIMIC-IV data, and the methods play the role of an interface between the library user and SQL queries. In essence, users will only operate using Python methods, assigning them parameters to customize SQL queries, enabling researchers to retrieve complex information efficiently, without the need to divert attention from their primary research objectives to low-level query logic.

The first step towards defining the queries in SQL and the main data of interest to be extracted from MIMIC-IV was the analysis of its data model. After the analysis, it was possible to identify that very relevant data about the patients, such as sociodemographic information (age, ethnicity, gender, etc.), are present in the database. This information, however, has a certain complexity of access, in technical terms, due to the need to join tables to add this information to the study dataset. Based on this finding, the process of creating specific queries to return each piece of information was started, based on the unique identifier of each patient contained in the MIMIC-IV to carry out the necessary data aggregation processes and return this information so that the user can easily add it to their dataset using the Python programming language.

The methods in Python, used in the construction of this library to encapsulate the SQL queries, were designed to bring as a final result an object that could be used in the most dynamic ways possible by users with an intermediate knowledge of data manipulation in this programming language. For this reason, all methods return objects of type DataFrame. DataFrames are data structures implemented by the Pandas library, which is aimed at data analysis, and are capable of storing two-dimensional tabular data, changeable in size and potentially heterogeneous. In addition, DataFrames contain labeled axes (columns and rows), being very similar to tables [15].

Since the main objective of this work is to support the scientific community in making better use of the data available in MIMIC-IV, it is convenient that this library is made available to everyone who wants to use it and contribute to its development.

2.4 Ethical Aspects

All stages of the project were conducted using public data. Therefore, as this is a study that does not involve research with human subjects, it is in accordance with Resolution No. 510, of April 7, 2016, of the Brazilian National Health Council, and does not require submission to a Research Ethics Committee.

3 RESULTS

As a result of this work, we developed a PoC implementation in the form of an open-source Python library called “MIMIC-IV Extractor Tool.” Its main objective is to demonstrate how the access and consolidation of biomedical data from MIMIC-IV can be

simplified for researchers who work primarily with Python and seek to minimize the time spent on data engineering tasks. This library is available in a public repository on GitHub, accessible at: *https://github.com/willian2610/MIMIC-IV-Extractor-Tool*.

The library acts as an interactive interface between the user (researchers) and the MIMIC-IV database, which is hosted in a local environment (PostgreSQL), as shown in Figure 1. Its main purpose is to abstract the complexity of SQL queries, allowing users to retrieve data of interest without directly manipulating query logic.

The selection of methods implemented in the library was guided by the relevance of the information for the construction of datasets for biomedical studies, prioritizing core patient attributes such as age, race, and gender. In addition, the library includes more specific methods capable of retrieving information such as intensive care unit (ICU) admissions based

Table 1. Available methods on MIMIC-IV Extractor Tool Library

Method	Description
get_data_from_mimic	This method runs a given query on MIMIC-IV.
get_patients_age	This method calculates the true age of patients by using a Hospital Admission ID.
get_patients_race	This method retrieves patients' race information from hospital admissions, supporting strategies like the most recent race per patient.
get_patients_gender	This method returns patients' gender using their subject IDs.
get_icu_stays_by_diagnosis	This method uses a diagnosis term to return all ICU stays that can relate to that diagnosis.

Source: Prepared by the authors (2026).

Figure 1. MIMIC-IV Extractor Tool serving as an interface between Users and MIMIC-IV.



Source: Prepared by the authors (2026).

Figure 2. Instantiating the library class.

```
from mimic_extractor import MimicExtractor
import pandas as pd

ME = MimicExtractor(user="postgres", password="admin", database="mimic",
                    host="localhost", port=5432)
```

Source: Prepared by the authors (2026).

Figure 3. Getting ICU stays under sepsis diagnosis.

```
""" Get all ICU Stays under 'Sepsis' diagnosis """
icu_stays = ME.get_icu_stays_by_diagnosis(diagnosis_term="sepsis")
```

Source: Prepared by the authors (2026).

Figure 4. Getting patients' ages.

```
""" Using Hospital Admission Id (hadm_id) to get patients age """
p_ages = ME.get_patients_age(hadm_ids=icu_stays['hadm_id'].tolist())
```

Source: Prepared by the authors (2026).

Figure 5. Merging icu_stays and p_ages DataFrames.

```
""" Adding the patients ages to the icu_stays DataFrame """
icu_stays_final = pd.merge(icu_stays, p_ages, on=['hadm_id', 'subject_id'])
```

Source: Prepared by the authors (2026).

Figure 6. Filtering DataFrame information.

```
""" Removing all results where patients are outside the age window from 30 to 49 years old """
dataset_age = icu_stays_final.drop(icu_stays_final[(icu_stays_final.age < 30) |
                                                   (icu_stays_final.age > 49)].index)
""" Removing all results where the patients gender is not Male """
final_dataset = dataset_age.drop(dataset_age[dataset_age.gender!="M"].index)
```

Source: Prepared by the authors (2026).

on diagnosis.

The methods made available by the library act as facilitators for consolidating the information contained in MIMIC-IV. The researcher can make use of more generic methods that capture patients' information to create their own datasets based on MIMIC-IV data and customize them according to the needs of their research. These methods can be used individually or in combination, allowing queries that are quite complex in terms of data manipulation in SQL and business rules to be easily resolved with just the use of parameters chosen by the researcher. Table 1 presents the available methods along with their descriptions.

To better illustrate the use of library methods, we are going to present a use case:

Use case: Suppose a researcher needs to obtain a dataset of male patients aged between 30 and 49 years who were admitted to the ICU under the diagnosis of sepsis.

If this query were created directly in SQL, the researcher would first need to map which of the available tables could contain sufficiently useful information for the composition of the dataset to be feasible. In this case, a total of 5 tables would need to be crossed using different identifiers, in addition to applying business rules to correctly filter the results.

When using the library, the researcher would need to use two methods in combination, applying the appropriate manipulations to the resulting DataFrames, such as merging and removing unwanted rows.

The first step is to instantiate the library class. For this, we need to import the class MimicExtractor and then instantiate it by passing the parameters responsible for connecting to the database (Figure 2).

Now, we need to get all the ICU stays under the desired diagnosis (Figure 3):

Executing the method shown in Figure 3 returns a DataFrame containing 19 columns with a total of 25,298 rows, considering the full MIMIC-IV dataset.

The next step is to retrieve the age of all patients of interest. For this, all of the Hospital Admission IDs (hadm_id) retrieved in the previous step should be used in order to generate a new DataFrame that will relate the patient with their respective age at the moment of admission to the ICU (Figure 4).

With a DataFrame containing the ICU stays under sepsis diagnosis and a DataFrame containing the ages of the patients, it is possible to merge both objects since the columns hadm_id and subject_id are present in both DataFrames (Figure 5).

Now we have a DataFrame with 20 columns and 25,298 records, containing all the information of the ICU stays under the diagnosis of sepsis in addition to the age of the patient. The last step is to filter the information we need from this DataFrame (male patients with ages from 30 to 49 years old), as shown in Figure 6:

The final dataset for this use case is composed of a DataFrame with 15 columns and 1,508 rows. With the combination of the library methods, we were able to retrieve data from MIMIC-IV in a simple way, without the need to manipulate SQL code.

This use case serves as a practical validation of the proposed PoC, illustrating how the library can effectively reduce the complexity involved in querying MIMIC-IV. By comparing the traditional SQL-based approach with the proposed abstraction, it becomes evident that the PoC successfully simplifies the data retrieval workflow.

4 DISCUSSION

The MIMIC-IV Extractor Tool addresses a critical bottleneck in biomedical research by replacing complex SQL logic with a high-level Python interface. The practical use case demonstrated that operations involving the joining of multiple distinct tables can be reduced to just two methods, substantially reducing the engineering effort required and the risk of extraction errors introduced by manual query construction.

When compared to existing literature, a clear difference can be observed: while researchers usually focus on high-level clinical outcomes, their work presupposes an advanced level of data engineering to handle the modular complexity of MIMIC-IV. This work, on the other hand, focuses on the infrastructure used to access the data, providing tools that make these clinical insights more accessible to multidisciplinary researchers.

In addition, by returning results as Pandas DataFrames, the library integrates easily into the scientific tools already familiar to the community. These findings confirm that abstraction layers are a valid and necessary strategy for accelerating data-driven healthcare research, enabling multidisciplinary collaboration while preserving analytical depth.

5 CONCLUSION

Data engineering overhead is an avoidable tax on research productivity. By abstracting the complexity of data retrieval, the developed tool allows researchers to redirect their effort toward clinical analysis and value generation.

This work presented a PoC demonstrating that a Python-based abstraction layer effectively simplifies the process of accessing and consolidating data from the MIMIC-IV database. By abstracting the complexity of data retrieval, the developed tool allows researchers to redirect their effort from infrastructure concerns to clinical analysis and value generation.

The initiative contributes to the advancement of science by enabling multidisciplinary teams to conduct research that previously required complex, specialized engineering processes. By lowering the entry point for MIMIC-IV data access, the tool broadens the range of researchers who can engage with one of the most comprehensive freely available clinical databases, potentially accelerating the pace of biomedical discoveries.

As an open-source project, the tool is designed to evolve alongside the needs of the scientific community. Future development directions include the integration with cloud environments such as Google Cloud Platform (GCP) and Amazon Web Services (AWS), which would enable scalable and infrastructure-independent data access, particularly relevant for institutions with limited local computing resources. Additionally, the expansion of available retrieval methods, guided by the clinical demands expressed by the research community, represents a natural path for consolidating the tool as a reliable resource for biomedical data science.

Links

The library is available on GitHub, accessible at: https://github.com/willian2610/MIMIC-IV-Extractor-Tool.

Author contributions. The authors contributed equally to all stages of the article: conception and design of the study; data collection, analysis, and interpretation; drafting and critical revision of the manuscript; and approval of the final version.

Acknowledgments. Not applicable.

Funding. This research received no specific funding.

Data and materials availability. The source code of the MIMIC-IV Extractor Tool is openly available on GitHub at https://github.com/willian2610/MIMIC-IV-Extractor-Tool. The MIMIC-IV data used in this study are available through PhysioNet (https://physionet.org/content/mimiciv/) to credentialed users who complete the required training and sign the Data Use Agreement. Under that agreement, the authors cannot redistribute the data.

Conflicts of interest. The authors declare no conflicts of interest.

Use of artificial intelligence. During the preparation of this work, artificial intelligence (AI) tools were used for the purposes of grammar verification and to assist in the verification of the Portuguese-to-English translation of the manuscript. Following the use of these tools, we reviewed, edited, and validated the resulting text to ensure its accuracy and academic integrity. We take full responsibility for the final content and the original ideas presented in this article.

REFERENCES

1. Dash S, Shakyawar SK, Sharma M, Kaushik S. Big data in healthcare: management, analysis and future prospects. J Big Data. 2019;6(1):54. doi: https://doi.org/10.1186/s40537-019-0217-0
2. Fang R, Pouyanfar S, Yang Y, Chen SC, Iyengar SS. Computational Health Informatics in the Big Data Age. ACM Comput. Surv. 2017;49(1):1-36. doi: https://doi.org/10.1145/2932707
3. Brasil. Conselho Nacional de Saúde. Resolução nº 510, de 7 de abril de 2016 [Internet]. Brasília: Conselho Nacional de Saúde; 2016 [cited 2026 Oct 18]. Available from: http://conselho.saude.gov.br/resolucoes/2016/Reso510.pdf
4. Johnson AEW, Bulgarelli L, Shen L, Gayles A, Shammout A, Horng S, et al. MIMIC-IV, a freely accessible electronic health record dataset. Sci Data. 2023;10(1):1. doi: https://doi.org/10.1038/s41597-022-01899-x
5. Kimball R, Caserta J. The Data Warehouse ETL Toolkit. Indianapolis: Wiley Publishing; 2004.
6. Borges HP, Souza JND, Schulze B, Mury AR. Computação em nuvem. 2011.
7. Haas, Padilha RJ, Monteiro TB, Beck MB. Problema do caixeiro viajante utilizando algoritmo genético no ambiente em nuvem. 2015.
8. Wang H, Zheng Z, Wu L, Li P. New directly revocable attribute-based encryption scheme and its application in cloud storage environment. Cluster Comput. 2017;20(3):2385-92. doi: https://doi.org/10.1007/s10586-016-0701-7
9. Oliphant T. Python for Scientific Computing. Comput. Sci. Eng. 2007;9(3):10-20. doi: https://doi.org/10.1109/mcse.2007.58
10. McKinney W. Python para análise de dados: tratamento de dados com pandas, NumPy e IPython. São Paulo: Novatec Editora; 2018.
11. Gao J, Su SD, Wang Y. Development and validation of a mortality risk prediction model for patients with bone tumors using the MIMIC-IV database: a retrospective cohort study. Research Square [Preprint]. 2025. doi: https://doi.org/10.21203/rs.3.rs-5789788/v1
12. Han Y, Xie X, Qiu J, Tang Y, Song Z, Li W, et al. Early prediction of sepsis associated encephalopathy in elderly ICU patients using machine learning models: a retrospective study based on the MIMIC-IV database. Front Cell Infect Microbiol. 2025;15:1545979. doi: https://doi.org/10.3389/fcimb.2025.1545979
13. Chen BA, Lu S, Yang QH, Chen HB, Yuan Y. Beta-Blockers and mortality in patients with malignancy and cardiovascular disease: an analysis of the MIMIC-IV database. BMC Cancer. 2025;25(1):1351. doi: https://doi.org/10.1186/s12885-025-14745-1

14. Vargas W, Jardim AG, Botelho VR, Pianoschi TA, Winck AT. Acesso e Recuperação de Dados Biomédicos no MIMIC-IV. In: Minicursos do XXV Simpósio Brasileiro de Computação Aplicada à Saúde. [place unknown]: SBC; 2025. p. 45-88. doi: https://doi.org/10.5753/sbc.16731.5.2

15. Pandas. API Reference [Internet]. 2022. Available at: https://pandas.pydata.org/docs/reference/index.html

EDITORIAL NOTES

The Journal of Health Informatics is a scientific journal maintained by the Brazilian Society of Health Informatics (Sociedade Brasileira de Informática em Saúde), a non-profit scientific association dedicated to the development of Health Informatics and Digital Health in Brazil. The editorial process is conducted with scientific independence, free from institutional, commercial, or sponsor interference, and follows international best practices in publication ethics.

This article was double-blind peer reviewed by specialists in the field. The editorial team used artificial intelligence tools, under human supervision, in the initial screening, in checking the text, references and numbers and, when needed, in redrawing figures, charts and tables, which the authors approved in the proof. The tools and safeguards are described in the journal's Policy on the use of artificial intelligence. Editorial decisions were made by people.

