Capítulo 2

Acesso e Recuperação de Dados Biomédicos no MIMIC-IV

Willian de Vargas, André Gonçalves Jardim, Viviane Rodrigues Botelho, Thatiane Alves Pianoschi, Ana Trindade Winck

Abstract

This chapter provides a practical and accessible guide for accessing and retrieving clinical data from the MIMIC-IV database, one of the leading open-source biomedical information sources. It covers the database's modular structure, local access strategies via PostgreSQL and SQLite, and the use of SQL and Python for exploratory analyses. The text includes comprehensive tutorials for data import, building derived views, and extracting relevant clinical subsets. The goal is to empower healthcare professionals and researchers to use real clinical data in reproducible studies, promoting technical autonomy and interdisciplinarity in data science applied to healthcare.

Resumo

Este capítulo apresenta um guia prático e acessível para acesso e recuperação de dados clínicos a partir do banco de dados MIMIC-IV, uma das principais fontes abertas de informações biomédicas. São abordadas a estrutura modular do banco, estratégias de acesso local via PostgreSQL e SQLite, e o uso de SQL e Python para análises exploratórias. O texto inclui tutoriais completos para importação dos dados, construção de visões derivadas e extração de subconjuntos clínicos relevantes. O objetivo é capacitar profissionais da saúde e pesquisadores a utilizarem dados clínicos reais em estudos reprodutíveis, promovendo a autonomia técnica e a interdisciplinaridade em ciência de dados aplicada à saúde.

2.1. Introdução

Bancos de dados clínicos abertos têm desempenhado um papel fundamental no avanço da pesquisa em saúde, oferecendo acesso a informações para investigação científica e desenvolvimento de tecnologias para a área da saúde. Dentre os repositórios mais conhecidos, destaca-se o *PhysioNet*, mantido pelo Laboratório de Fisiologia Computacional do *Massachusetts Institute of Technology* (MIT). Essa iniciativa disponibiliza dados biomédicos anonimizados de atendimentos hospitalares reais, com o objetivo de fomentar o desenvolvimento de pesquisas científicas e soluções tecnológicas aplicadas à medicina.

Um dos conjuntos de dados mais completos e utilizados da plataforma *PhysioNet* é o MIMIC (*Medical Information Mart for Intensive Care*), atualmente em sua quarta versão, o MIMIC-IV. Publicada em 2024, essa versão reúne dados clínicos anonimizados de cerca de 315 mil pacientes atendidos no hospital *Beth Israel Deaconess Medical Center*, incluindo internações hospitalares e atendimentos em unidades de terapia intensiva entre 2008 e 2019. O banco está dividido em módulos, oferecendo acesso a informações detalhadas como sinais vitais, exames laboratoriais, prescrições, procedimentos e notas clínicas, além de permitir análises temporais e multivariadas.

Apesar de seu grande potencial, a complexidade estrutural do MIMIC-IV pode representar uma barreira significativa para profissionais da saúde e pesquisadores com pouca familiaridade em ciência de dados e bancos de dados relacionais. Este capítulo propõe um olhar prático e acessível sobre o processo de acesso, configuração e exploração dos dados do MIMIC-IV, utilizando ferramentas como PostgreSQL, SQLite, SQL e Python. São apresentados tutoriais para importar os dados, realizar análises exploratórias e extrair subconjuntos relevantes para estudos clínicos, com o objetivo de contribuir para a democratização do uso do MIMIC-IV na comunidade científica brasileira, promovendo a interdisciplinaridade e a autonomia técnica dos profissionais interessados em aplicar ciência de dados à saúde.

2.2. O Banco de Dados MIMIC

O *Medical Information Mart for Intensive Care* (MIMIC) é um banco de dados relacional disponibilizado pela plataforma *PhysioNet*. Desenvolvido a partir de uma colaboração entre o hospital *Beth Israel Deaconess Medical Center* e o laboratório de Fisiologia Computacional do MIT (MIT-LCP), o MIMIC tem como objetivo fornecer acesso gratuito e anonimizado a dados clínicos reais de pacientes que passaram por atendimento em unidades de terapia intensiva (UTIs) deste hospital [Johnson et al. 2024].

A iniciativa do MIMIC surgiu da necessidade de disponibilizar conjuntos de dados clínicos detalhados para pesquisadores, promovendo o avanço da medicina baseada em evidências e o desenvolvimento de ferramentas computacionais para suporte à decisão médica. Desde sua primeira versão, lançada em 2003, o projeto evoluiu até chegar à versão mais recente, o MIMIC-IV, que oferece uma estrutura de dados mais moderna e dividida em módulos [Johnson et al. 2024].

Entre os módulos disponibilizados pelo MIMIC-IV, destacam-se dois conjuntos principais de tabelas: o módulo HOSP e o módulo ICU. Esses módulos estruturam as informações de maneira complementar, permitindo analises tanto em nível geral de inter-

nação quanto em situações de cuidados intensivos.

O módulo HOSP abrange os dados hospitalares gerais, incluindo informações administrativas e clínicas ao longo da internação do paciente. Esse módulo contém tabelas com resultados de exames laboratoriais, prescrições médicas, procedimentos realizados, diagnósticos, notas clínicas, registros demográficos e administrativos, entre outros.

O módulo ICU é focado nos dados coletados especificamente durante a permanência dos pacientes em unidades de terapia intensiva (UTI). Ele inclui medições frequentes de sinais vitais, fluidos, intervenções realizadas, uso de dispositivos médicos, entre outros dados críticos que refletem o estado clínico dos pacientes em tempo quase real.

Essa divisão entre os módulos permite análises direcionadas e mais eficientes. Pesquisadores podem optar por explorar aspectos mais amplos da internação hospitalar utilizando os dados do módulo HOSP ou concentrar-se em episódios de maior gravidade, com base nos registros detalhados das UTIs presentes no módulo ICU. Essa flexibilidade é fundamental para o desenvolvimento de estudos clínicos com diferentes escopos.

Para facilitar o uso e promover a padronização das análises realizadas com o MIMIC-IV, foi desenvolvido o repositório MIMIC-IV *Concepts*, que disponibiliza uma coleção de consultas SQL pré-definidas e reutilizáveis, denominadas *concepts*. Essas consultas implementam definições clínicas comumente utilizadas em pesquisas, como ventilação mecânica, uso de vasopressores, comorbidades e critérios diagnósticos, permitindo que diferentes estudos adotem critérios uniformes. Um exemplo prático é a *view vitalSign*, que consolida medições frequentes de sinais vitais dos pacientes, como frequência cardíaca, pressão arterial, temperatura e saturação de oxigênio. Essa *view* organiza os dados de forma estruturada a partir de múltiplas tabelas do módulo ICU, simplificando o acesso a informações essenciais para a avaliação do estado clínico dos pacientes ao longo do tempo.

2.2.1. Histórico e Evolução do MIMIC

O banco de dados MIMIC teve sua primeira versão disponibilizada em 2003, inicialmente como uma iniciativa modesta, composta por dados clínicos de pacientes internados em UTIs do *Beth Israel Deaconess Medical Center* (BIDMC). Desde então, o projeto passou por diversas fases de expansão e refinamento, resultando nas versões MIMIC-II, MIMIC-III e, mais recentemente, no MIMIC-IV, lançado oficialmente em 2024 [Johnson et al. 2024].

O MIMIC-II trouxe melhorias significativas em relação à padronização dos dados e ao suporte para estudos retrospectivos com foco em desfechos clínicos. Já o MIMIC-III, amplamente adotado pela comunidade científica, consolidou o banco como uma das principais fontes públicas de dados clínicos [Johnson et al. 2016], abrangendo mais de 60.000 admissões em UTI entre os anos de 2001 e 2012. Ele introduziu melhorias na documentação, maior detalhamento nas tabelas e a inclusão de notas clínicas desidentificadas, ampliando o potencial de uso em pesquisas com Processamento de Linguagem Natural (PLN).

A versão mais atual do *MIMIC-IV* abrange hospitalizações e atendimentos no departamento de emergência do *Beth Israel Deaconess Medical Center* entre 2008 e 2019,

reunindo dados de aproximadamente 315 mil pacientes, totalizando mais de 524 mil admissões hospitalares. O *MIMIC-IV* é composto por registros eletrônicos de saúde (*Electronic Health Record* - EHRs) e sistemas de gestão hospitalar, como o *MetaVision*, e inclui informações demográficas, sinais vitais, resultados laboratoriais, administração de medicamentos, diagnósticos, procedimentos, observações clínicas e notas desidentificadas. Sua organização modular e a incorporação de dados do departamento de emergência ampliam o escopo da base para além das UTIs, permitindo estudos mais robustos e com maior representatividade, inclusive em análises de trajetórias clínicas desde a entrada no hospital até o desfecho final [Johnson et al. 2024].

Todos os registros são anonimizados, e os identificadores dos pacientes são substituídos por códigos, de modo a proteger a privacidade e atender às exigências éticas e legais para o uso dos dados em pesquisas. A utilização do MIMIC exige que os pesquisadores concluam um curso de proteção de dados e apresentem uma proposta de pesquisa aprovada [Johnson et al. 2024].

Uma das grandes vantagens do MIMIC é a riqueza e diversidade dos dados. Além dos dados estruturados, como exames laboratoriais e medicações de sinais vitais, ele também inclui notas clínicas em linguagem natural, o que permite pesquisas avançadas utilizando técnicas de Processamento de Linguagem Natural. Essa variedade viabiliza uma ampla gama de estudos, desde análises epidemiológicas e avaliações de desempenho hospitalar até aplicações de aprendizado de máquina para predição de desfechos clínicos.

A relevância do MIMIC para a pesquisa biomédica pode ser observada na quantidade crescente de estudos que o utilizam como base. Uma busca nas tendências de publicação do PubMed [National Library of Medicine (US) 2025] pelos termos "MIMIC-I", “MIMIC-II”, “MIMIC-III” ou “MIMIC-IV” revela um aumento consistente nas publicações ao longo da última década, refletindo o crescente interesse da comunidade científica nessa base de dados, conforme apresentado na Figura 2.1.

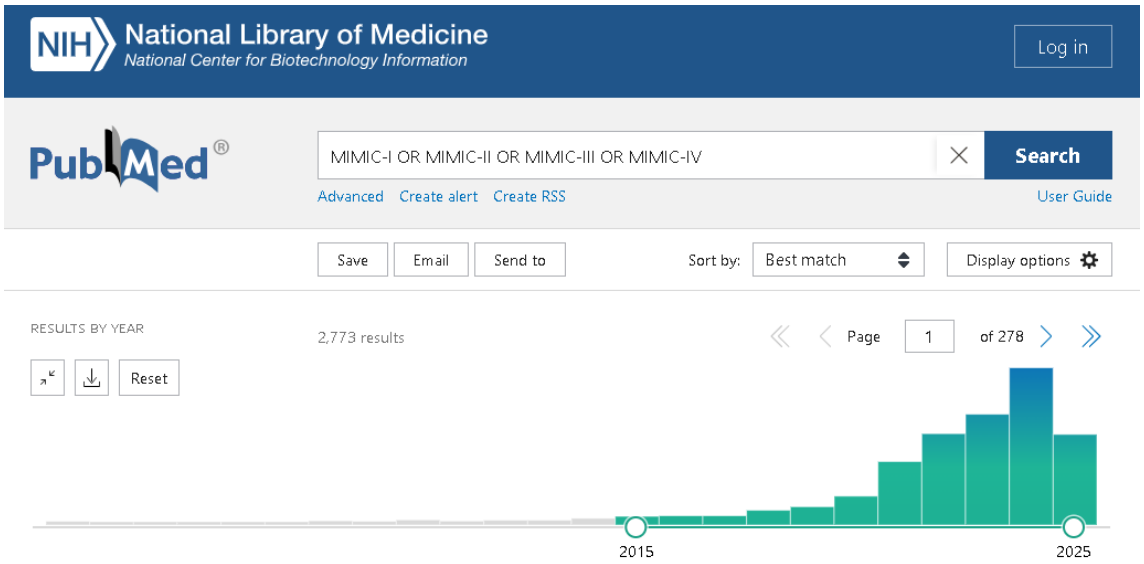

Figura 2.1. Tendência de publicações no PubMed contendo os termos *MIMIC-I*, *MIMIC-II*, *MIMIC-III* ou *MIMIC-IV*, no período de 2015 a 2025. [National Library of Medicine (US) 2025]

2.2.2. Exemplos da utilização do MIMIC em estudos científicos

A versatilidade do MIMIC é evidenciada pelo crescente número de estudos científicos que utilizam seus dados para desenvolver modelos preditivos, avaliar riscos clínicos e propor intervenções baseadas em evidências. A seguir, são apresentados exemplos de aplicações práticas recentes com base na literatura.

Em [Jung et al. 2024] foram explorados fatores preditivos para a progressão da insuficiência cardíaca em pacientes hipertensos, utilizando exclusivamente dados de diagnósticos prévios à hipertensão presentes no banco MIMIC-IV. O objetivo foi possibilitar a antecipação do risco de insuficiência cardíaca no momento do diagnóstico de hipertensão. Para isso, os autores aplicaram testes qui-quadrado e modelos baseados em *XGBoost* para analisar fatores preditivos específicos por faixa etária. Os resultados revelaram um conjunto de condições associadas ao agravamento da insuficiência cardíaca, incluindo fibrilação atrial, insuficiência renal, doença pulmonar obstrutiva crônica, anemia e uso de anticoagulantes. A abordagem adotada contribui para o monitoramento personalizado e a estratificação de risco desses pacientes.

No campo das doenças infecciosas, em [Pérez-Tome et al. 2024] foi desenvolvido um modelo preditivo de mortalidade por sepse utilizando técnicas de aprendizado de máquina, com foco específico no algoritmo *Random Forest*. O estudo foi conduzido com dados de pacientes internados em unidades de terapia intensiva (UTIs) de três hospitais espanhóis, além de pacientes provenientes da base de dados MIMIC-III, totalizando 4.739 pacientes (180 locais e 4.559 do MIMIC-III). O objetivo foi construir um classificador capaz de prever o risco de óbito em pacientes com sepse, a partir de variáveis fisiológicas e laboratoriais coletadas durante a internação. Os resultados demonstraram elevado desempenho do modelo, com acurácia de 96,77% e área sob a curva ROC (AUC) de 95% no conjunto de dados local, e ainda melhor desempenho com os dados do MIMIC-III: acurácia de 98,28% e AUC de 97,3%. Entre as variáveis mais relevantes para a predição da mortalidade por sepse, destacaram-se os níveis de lactato, o débito urinário e os parâmetros relacionados ao equilíbrio ácido-base. Curiosamente, os níveis de potássio foram mais determinantes na base MIMIC-III do que nos dados dos hospitais espanhóis, o que pode refletir particularidades clínicas ou laboratoriais específicas do contexto norte-americano. Esses achados demonstram não apenas a viabilidade de modelos de aprendizado de máquina na estratificação de risco em pacientes sépticos, como também reforçam a utilidade do MIMIC como um repositório confiável e robusto para o desenvolvimento e validação de ferramentas de apoio à decisão clínica em ambientes de terapia intensiva.

Em um estudo voltado para complicações renais em pacientes críticos, em [Lin et al. 2024] foram desenvolvidos e validados modelos preditivos de lesão renal aguda (LRA) em pacientes com pancreatite aguda (PA) utilizando dados do banco MIMIC-IV. Foram analisados 1.235 pacientes internados com PA, dos quais 54% desenvolveram LRA durante a hospitalização. Sete algoritmos de aprendizado de máquina foram aplicados para a construção dos modelos: *Random Forest*, *Support Vector Machine* (SVM), *K-Nearest Neighbors*(KNN), *Naive Bayes* (NB), *Neural Network* (NNET), *Generalized Linear Model* (GLM) e *Gradient Boosting Machine* (GBM). O modelo baseado em GBM apresentou o melhor desempenho, com AUC de 0.867 no conjunto de teste, indicando alta capacidade discriminativa para prever a ocorrência de LRA. Os autores destacam o potencial

desses modelos para apoiar decisões clínicas ao identificar precocemente pacientes em risco, promovendo intervenções oportunas e potencialmente reduzindo a mortalidade em unidades de terapia intensiva.

Ainda no contexto da sepse, em [Sun et al. 2024] foi desenvolvido e validado um nomograma preditivo (ferramenta gráfica que estima a probabilidade de um desfecho clínico com base em múltiplas variáveis) para estimar a mortalidade em 30 dias de pacientes com sepse associada a sangramento gastrointestinal (GIB), utilizando dados do banco MIMIC-IV. O estudo retrospectivo incluiu 1.435 pacientes, divididos aleatoriamente em coortes de treinamento e validação. Para a construção do modelo, os autores aplicaram regressão LASSO para seleção de variáveis e regressão logística multivariada para estimativa dos riscos. O desempenho do modelo foi avaliado por meio do índice de concordância (C-index), curvas ROC e análise de decisão (DCA), apresentando boa capacidade discriminativa tanto no conjunto de treinamento (C-index: 0.746) quanto de validação (C-index: 0.716). O nomograma incorporou variáveis como idade, tabagismo, glicose, ureia (BUN), lactato, escore SOFA, ventilação mecânica $\geq 48h$, nutrição parenteral e DPOC como fatores preditivos independentes. Os autores destacam o potencial clínico do modelo como uma ferramenta de apoio à tomada de decisões individualizadas no tratamento de pacientes críticos com sepse e GIB.

Em [Fan et al. 2025] foi realizado um estudo de coorte retrospectivo utilizando dados de 5.110 pacientes com diferentes tipos de acidente vascular cerebral (AVC), extraídos do banco de dados MIMIC-IV, abrangendo o período de 2010 a 2020. O objetivo do estudo foi identificar os principais fatores associados à mortalidade em curto, médio e longo prazos — especificamente em 30 dias, 90 dias, 1 ano e 3 anos após o evento. As análises estatísticas incluíram modelos de regressão de Cox, *Random Forest* e *Gradient Boosting*, que revelaram a importância de variáveis como tipo de AVC, índice de comorbidades de Charlson, escore SOFA, níveis de hemoglobina, idade avançada, tempo de internação e disfunções orgânicas. O estudo mostrou, por exemplo, que um aumento no escore SOFA está relacionado a maior risco de mortalidade em todos os períodos analisados. Esses achados oferecem informações valiosas para o aprimoramento da estratificação de risco, planejamento terapêutico e definição de estratégias de acompanhamento individualizado em pacientes com AVC.

Esses estudos ilustram não apenas a variedade de aplicações do MIMIC em diferentes domínios clínicos, mas também sua contribuição efetiva para o desenvolvimento de modelos de apoio à decisão médica baseados em dados reais. A capacidade de acessar dados clínicos estruturados e não estruturados, como exames laboratoriais, sinais vitais e notas clínicas, permite a construção de soluções que atendem a necessidades específicas em contextos de cuidados intensivos. Além disso, a ampla janela temporal e a riqueza demográfica da base possibilitam a análise de tendências, estratificação de risco e identificação precoce de eventos adversos, tornando o MIMIC uma ferramenta estratégica para a medicina de precisão.

Graças à sua qualidade, abrangência e acessibilidade, o MIMIC tornou-se um recurso bastante relevante para pesquisadores de diversas áreas, incluindo medicina, enfermagem, engenharia biomédica, ciência da computação e estatística. O acesso aberto e gratuito à base, aliado ao seu contínuo aprimoramento técnico e documental, facilita a re-

alização de estudos reprodutíveis, estimula colaborações interdisciplinares e democratiza o uso de dados clínicos em projetos acadêmicos e de inovação tecnológica. Essa abertura tem incentivado a formação de comunidades científicas em torno do uso do MIMIC, promovendo a troca de conhecimento e boas práticas no desenvolvimento de soluções orientadas por dados.

Portanto, a compreensão do funcionamento do MIMIC, suas origens, estrutura e potenciais de aplicação pode ser bastante útil para profissionais e pesquisadores interessados em explorar a interface entre dados clínicos e de saúde. O domínio dessa base de dados permite não apenas a realização de estudos retrospectivos com alto valor científico, mas também a criação de ferramentas com impacto direto na prática clínica. Com isso, o MIMIC se consolida como um elemento central na formação de uma nova geração de soluções em saúde, baseadas em ciência de dados e evidências clínicas reais.

2.3. Acesso ao MIMIC-IV

O acesso ao banco de dados MIMIC-IV pode ser realizado por diferentes meios, a depender das necessidades do usuário e dos recursos disponíveis. Esta seção apresenta as opções de acesso, os pré-requisitos para obtenção dos dados e um tutorial para configurar um ambiente local e importar os dados utilizando o PostgreSQL.

2.3.1. Formas de Acesso

Atualmente, existem duas formas principais de acessar os dados do MIMIC-IV:

- **Acesso Local**: envolve o download dos arquivos no formato CSV e posterior importação para um sistema gerenciador de banco de dados (SGBD), como PostgreSQL ou SQLite.
- **Acesso em Nuvem**: permite o uso de instâncias públicas hospedadas na Amazon Web Services (AWS) ou Google Cloud Platform (GCP), geralmente através de ferramentas como BigQuery.

Embora o acesso em nuvem seja conveniente para aplicações de produção ou análise de grandes volumes de dados, este tutorial terá como foco o acesso local, pois ele proporciona maior controle de custos, portabilidade e independência de infraestrutura externa.

2.3.2. Pré-requisitos para Acesso aos Dados

Para acessar a versão completa do MIMIC-IV, é necessário:

- Criar uma conta no portal PhysioNet (`https://physionet.org/`).
- Realizar um curso sobre ética em pesquisa com seres humanos, como o CITI Program (`https://physionet.org/about/citi-course/`).
- Submeter o certificado de conclusão e aceitar formalmente o termo de uso de dados (Data Use Agreement – DUA), disponível no próprio portal.

Se você está apenas explorando a estrutura do banco ou deseja seguir os exemplos deste tutorial, também é possível utilizar a versão Demo do MIMIC-IV, que é totalmente aberta e não exige cadastro ou certificação.

2.3.2.1. Download dos Dados

Na página de download os dados são distribuídos no formato `.csv.gz`, organizados por domínios como `hosp` e `icu`. Existem duas opções de download, uma que é a versão completa dos dados (Figura 2.2) e outra que é a versão de demonstração dos dados (Figura 2.3):

Versão Completa do MIMIC-IV

1. Acesse `https://physionet.org/content/mimiciv/`
2. Clique em "Files"
3. Baixe o arquivo ZIP correspondente

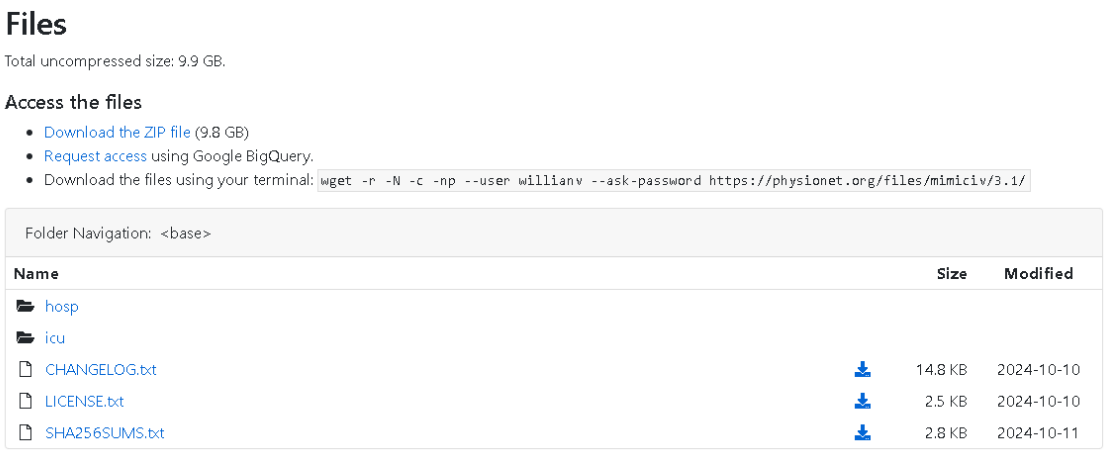

Figura 2.2. Página de download da versão completa do MIMIC-IV.

Versão Demo do MIMIC-IV

1. Acesse `https://physionet.org/content/mimic-iv-demo/`
2. Clique em "Files"
3. Baixe o arquivo ZIP correspondente

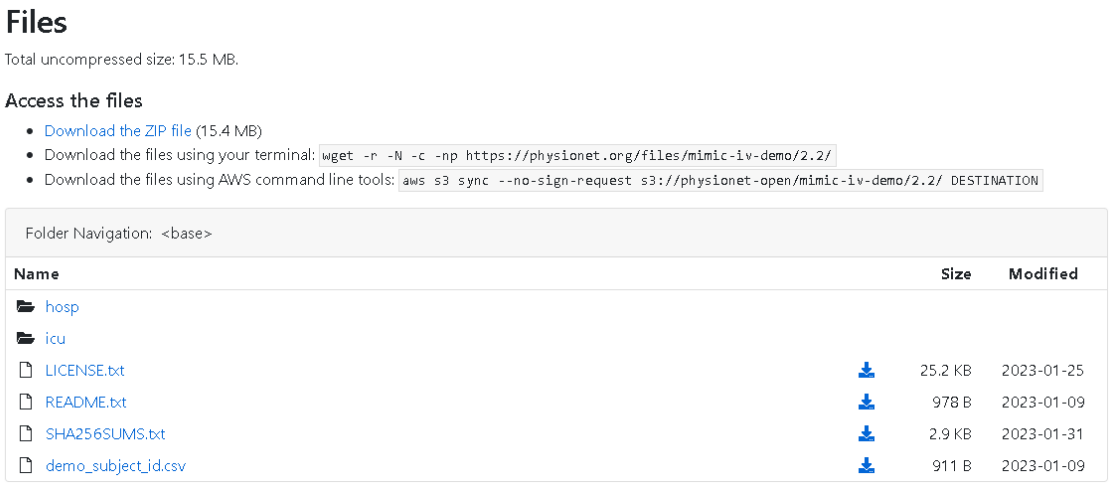

Figura 2.3. Página de download da versão demo do MIMIC-IV.

2.3.3. Configuração do Ambiente Local usando PostgreSQL *database*

A seguir, apresentamos o processo completo para configurar um ambiente local e importar os dados do MIMIC-IV utilizando o PostgreSQL. Os passos estão organizados para usuários de diferentes sistemas operacionais.

Passo 1. Instalar o PostgreSQL

Linux (Ubuntu):

```
sudo apt update
sudo apt install postgresql postgresql-contrib
```

Após a instalação, o serviço do PostgreSQL será iniciado automaticamente. Você pode verificar o status com:

```
sudo systemctl status postgresql
```

Windows/macOS:

Baixe o instalador em: `https://www.postgresql.org/download/`

Após instalar, verifique se tudo está instalado corretamente executando o seguinte comando no seu terminal:

```
psql --version
```

Passo 2. Criar um Banco de Dados

Antes de importar os dados do MIMIC-IV, precisamos criar um banco de dados vazio no PostgreSQL. Isso pode ser feito de duas formas:

- Via terminal (linha de comando)
- Via interface gráfica, como o pgAdmin (opcional, para quem preferir evitar o terminal)

Vamos primeiro explicar o método via terminal, que é o mais direto e compatível com os scripts de importação.

A. Via Terminal:

Primeiramente, abra seu terminal:

- Linux (Ubuntu): Pressione `Ctrl + Alt + T` ou procure por "Terminal" no menu de aplicativos
- Windows: Pressione `Win + R`, digite cmd ou powershell e pressione Enter
- macOS: Abra o Terminal pela busca (`Command + Espaço` → digite "Terminal")

Digite o seguinte comando para criar o banco de dados

```
createdb mimiciv
```

Esse comando cria um banco de dados chamado mimiciv, onde os dados do MIMIC-IV serão importados.

Se você receber um erro como:

```
createdb: command not found
```

ou

```
could not connect to database postgres: FATAL: role "seu_usuario
    " does not exist
```

isso pode indicar que o PostgreSQL não está configurado para o seu usuário do sistema.

Alternativa: usando o usuário postgres

O PostgreSQL, por padrão, cria um usuário chamado postgres. Você pode usar esse usuário para criar o banco com o comando abaixo (Linux/Mac):

```
sudo -u postgres createdb mimiciv
```

Esse comando pede a senha do seu sistema (não a do banco de dados).

Se estiver no Windows e quiser usar o usuário postgres, você pode: Abrir o terminal do psql (o cliente interativo do PostgreSQL):

```
psql -U postgres
```

Criar o banco dentro do terminal do PostgreSQL:


```
CREATE DATABASE mimiciv;
\q  -- para sair
```

B. Via pgAdmin:

Se você prefere evitar o terminal, também pode criar o banco pelo pgAdmin, a interface web oficial do PostgreSQL.

Quando você instala o PostgreSQL no seu computador, o pgAdmin geralmente é instalado automaticamente. Para abri-lo:

- Windows: Procure por pgAdmin no menu iniciar e clique para abrir.
- macOS/Linux: Você pode buscar pelo pgAdmin 4 na lista de aplicativos ou executá-lo a partir do terminal com o comando pgadmin4 (caso esteja instalado via terminal).

Conectar ao Servidor PostgreSQL:

Ao abrir o pgAdmin, a tela inicial será exibida. Caso você não tenha conectado a nenhum servidor, será solicitado para adicionar uma nova conexão.

Clique em **"Add New Server"** (Adicionar Novo Servidor).

Na janela que abrir, insira os seguintes dados:

- **General (Figura 2.4):**

  Name: Escolha um nome para a conexão, como PostgreSQL Local ou algo de sua escolha.

- **Connection (Figura 2.5):**

  Host: localhost (se estiver utilizando a instalação local).

  Port: 5432 (porta padrão do PostgreSQL).

  Username: postgres (ou o nome de usuário configurado no PostgreSQL).

  Password: A senha do seu usuário PostgreSQL.

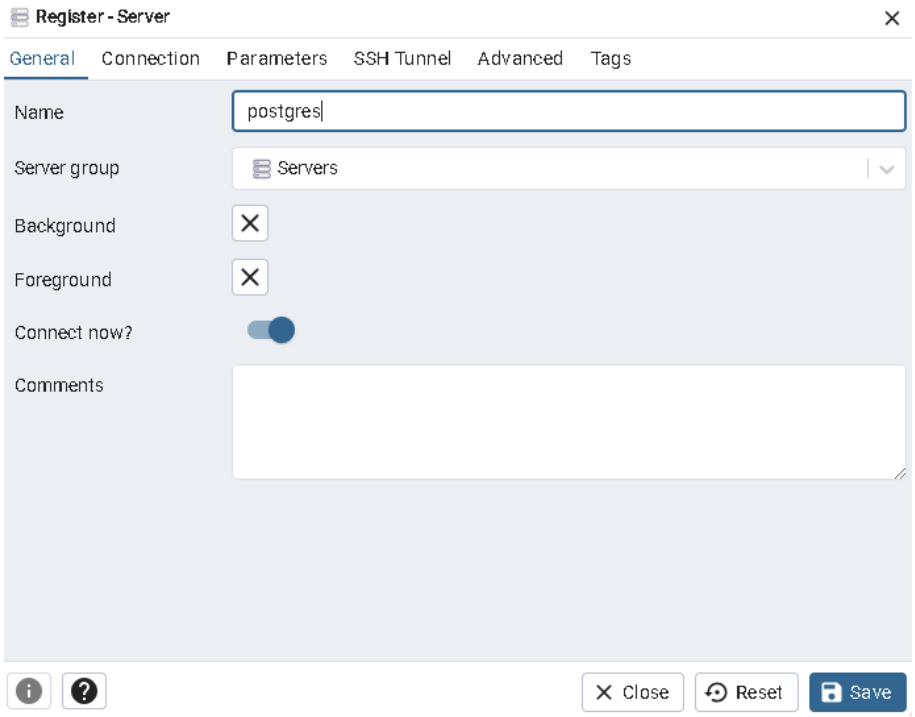

Figura 2.4. Captura de tela da aba "General"da tela de criação de um novo Server.

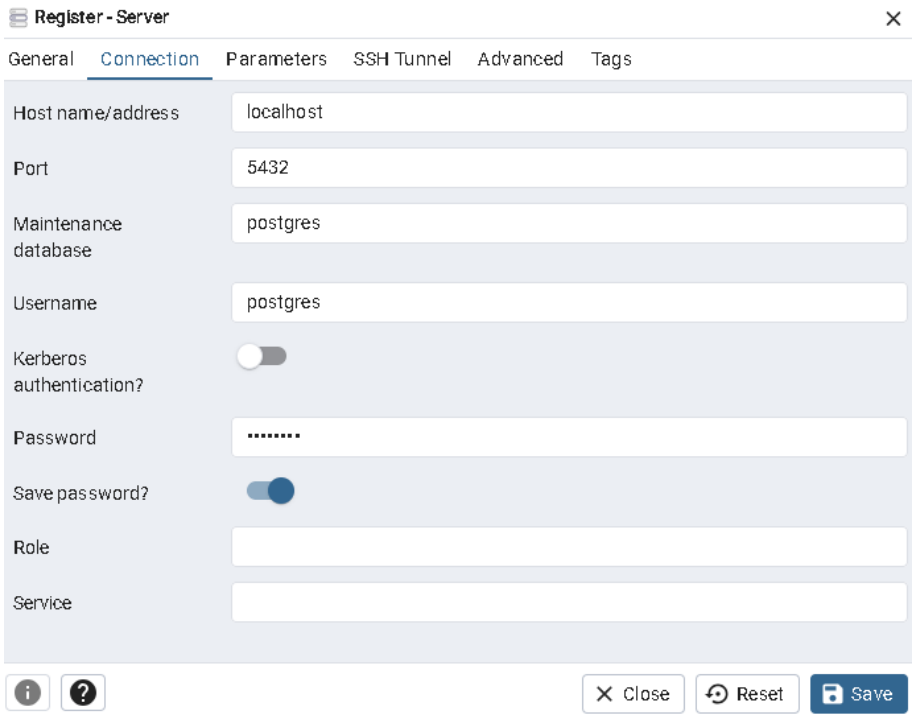

Figura 2.5. Captura de tela da aba "Connection"da tela de criação de um novo Server.

Após preencher os campos, clique em Save para estabelecer a conexão com o servidor.

Criar um Banco de Dados:

Após conectar-se ao servidor, o pgAdmin mostrará o painel de navegação à esquerda, com o nome do seu servidor na árvore.

Expanda o servidor recém-adicionado clicando sobre ele e, em seguida, sobre a pasta Databases.

Clique com o botão direito em Databases e selecione Create > Database..., como demonstrado na Figura 2.6

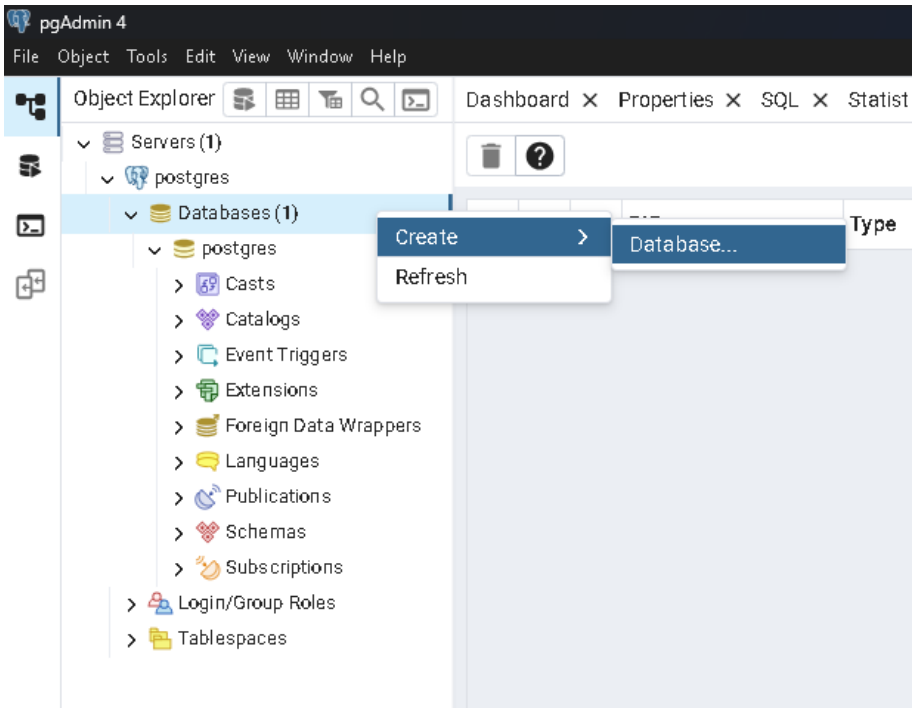

Figura 2.6. Captura de tela mostrando a opção de criação de uma nova base de dados no pgAdmin.

Na janela Create - Database, insira as seguintes informações:

- Database: Nomeie o banco de dados como mimiciv.
- Deixe as outras configurações no padrão.

Após preencher o nome do banco, clique em Save para criar o banco de dados.

Verificar o Banco Criado:

Após criar o banco, ele aparecerá na lista de bancos de dados dentro do servidor (Figura 2.7).

Clique sobre o banco de dados mimiciv para expandir sua estrutura e começar a trabalhar com ele.

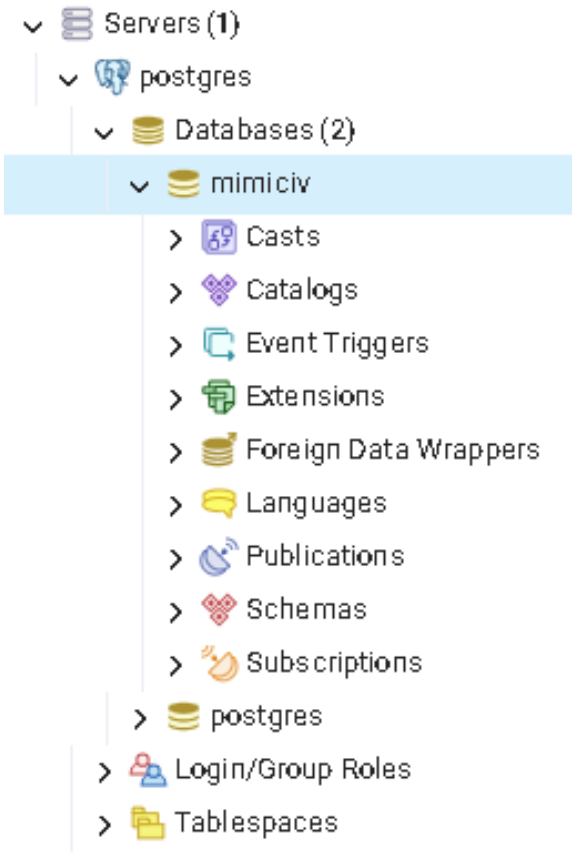

Figura 2.7. Captura de tela mostrando o novo banco de dados criado.

Passo 3. Baixar os Scripts Oficiais

Para importar os dados corretamente no PostgreSQL, você precisa dos scripts SQL que criam as tabelas e realizam a carga dos arquivos CSV. Esses scripts são fornecidos oficialmente pelo grupo MIT-LCP no repositório GitHub:

Repositório: `https://github.com/MIT-LCP/mimic-code`

Você tem duas opções para obter os arquivos:

Opção 1: Baixar ZIP

- Acesse o link acima.
- Clique no botão verde "Code".
- Escolha a opção "Download ZIP".
- Após o download, extraia o conteúdo do arquivo ZIP para uma pasta no seu computador. Por exemplo:

  `C:/mimic-code` no Windows

  `~/Documentos/mimic-code` no Linux/macOS

Opção 2: Clonar com Git

Se você tem o Git instalado em seu computador, pode clonar o repositório diretamente com o comando:

```
git clone https://github.com/MIT-LCP/mimic-code.git
```

Esse comando criará uma pasta chamada mimic-code com todos os arquivos dentro. A vantagem dessa abordagem é que, caso os scripts sejam atualizados no futuro, você poderá baixar as atualizações simplesmente rodando:

```
git pull
```

Organização dos Arquivos

Depois de baixar ou clonar, a estrutura da pasta será como a seguir:

```
mimic-iv-3.1/
|-- mimic-iii/
|   \-- ...
|-- mimic-iv/
|   |-- buildmimic/
|      |-- postgres/
|         |-- constraint.sql
|         |-- create.sql
|         |-- index.sql
|         \-- load.sql
|      |-- sqlite/
|      \-- ...
|   |-- concepts/
|   \-- ...
\-- ...
```

Esses são os principais arquivos que usaremos para criar as tabelas e importar os dados para o banco de dados mimiciv.

Passo 4. Descompactar os Arquivos CSV

Após o download do arquivo ZIP (realizado previamente), seja na versão completa ou na Demo, extraia o arquivo com um descompactador de arquivos de sua preferência.

Estrutura dos Dados Locais

Após o desempacotamento do arquivo ZIP, o diretório da base terá a seguinte estrutura:

```
mimic-iv-3.1/
|-- hosp/
|   |-- admissions.csv.gz
|   |-- labevents.csv.gz
|   |-- patients.csv.gz
|   |-- transfers.csv.gz
|   \-- ...
|-- icu/
|   |-- chartevents.csv.gz
|   |-- icustays.csv.gz
|   |-- procedureevents.csv.gz
|   \-- ...
```

Cada subpasta representa um "módulo"do banco de dados (core, hosp, icu), com arquivos CSV comprimidos. A importação converte esses arquivos em tabelas relacionais dentro do SGBD de sua escolha.

Passo 5. Criar as Tabelas no PostgreSQL

O script create.sql do repositório oficial cria todos os esquemas e tabelas necessários no banco mimiciv. Ele está localizado no seguinte caminho dentro do repositório: mimic-code/mimic-iv/buildmimic/postgres/create.sql

Para executar o script, use o seguinte comando no seu terminal:

```
psql -U [seu_usuario] -d mimiciv -f [caminho_local]/mimic-code/
   mimic-iv/buildmimic/postgres/create.sql
```

Substitua:

- seu_usuario: seu nome de usuário do PostgreSQL (por padrão, pode ser postgres).
- caminho_local: caminho absoluto para o arquivo no seu sistema.

Passo 6. Importar os Arquivos CSV

Os scripts load_gz.sql (caso os arquivos a serem importados sejam .gz) ou load.sql (caso os arquivos a serem importados sejam .csv) importam os arquivos diretamente no banco, por meio de comandos COPY. Estes scripts estão localizados no seguinte caminho dentro do repositório: mimic-code/mimic-iv/buildmimic/postgres/load.sql e mimic-code/mimic-iv/buildmimic/postgres/load_gz.sql

Antes de executar, edite esses scripts em um editor de texto e ajuste todos os caminhos dos arquivos (.csv.gz ou .csv) para apontar corretamente para o local onde você extraiu os arquivos em seu computador.

Exemplo de linha antes de ajustar:

```
\COPY mimiciv_hosp.admissions FROM admissions.csv DELIMITER ','
   CSV HEADER NULL '';
```

Exemplo após ajuste:

```
\COPY mimiciv_hosp.admissions FROM 'C:/Users/usuario/Downloads/
   mimic-iv-3.1/hosp/admissions.csv.gz' DELIMITER ',' CSV HEADER
    NULL '';
```

Para executar o script, use um dos seguintes comandos no seu terminal:

```
psql -U [seu_usuario] -d mimiciv -f [caminho_local]/mimic-code/
   mimic-iv/buildmimic/postgres/load.sql
```

```
psql -U [seu_usuario] -d mimiciv -f [caminho_local]/mimic-code/
   mimic-iv/buildmimic/postgres/load_gz.sql
```

No caso da importação da versão completa do MIMIC-IV, este processo pode levar muitas horas. Isso é completamente normal e esperado, considerando o alto volume de dados a serem importados. É importante manter com computador ligado durante todo o processo.

Passo 7. Verificar a Importação

Após importar os dados com sucesso, é importante validar que:

- Os dados estão presentes em cada tabela.
- É possível fazer consultas simples com o SQL.

Para isso, utilizaremos o cliente de linha de comando do PostgreSQL, chamado psql.

No terminal, execute:

```
psql -U seu_usuario -d mimiciv
```

Substitua seu_usuario pelo seu usuário PostgreSQL, geralmente postgres.

Se tudo estiver funcionando corretamente, você verá algo assim:

```
mimiciv=#
```

Dentro do terminal interativo do psql, execute o seguinte comando para verificar se as tabelas realmente existem:


```
\dt mimiciv.*
```

Esse comando lista todas as tabelas criadas nos esquemas do MIMIC-IV (como hosp e icu).

Você deverá ver uma saída como esta (resumo):

```
 Schema |       Name        | Type  |  Owner
--------+-------------------+-------+----------
 hosp   | admissions        | table | postgres
 hosp   | diagnoses_icd     | table | postgres
 ...
 icu    | caregiver       | table | postgres
 icu    | chartevents       | table | postgres
 ...
```

Agora vamos executar consultas simples para contar os registros e verificar se as tabelas têm dados:

```
SELECT COUNT(*) FROM mimiciv.hosp.patients;
SELECT COUNT(*) FROM mimiciv.hosp.admissions;
SELECT COUNT(*) FROM mimiciv.hosp.diagnoses_icd;
SELECT COUNT(*) FROM mimiciv.icu.chartevents;
```

Os resultados variam entre a versão completa e a Demo, mas devem sempre ser maiores que zero.

2.3.4. Configuração do Ambiente Local usando SQLite

Além do uso de bancos relacionais completos como o PostgreSQL, o MIMIC-IV também pode ser importado para um banco de dados SQLite, o que pode ser útil para fins de exploração leve dos dados, testes rápidos ou situações em que uma instalação de servidor de banco de dados não é viável.

A equipe do PhysioNet fornece dois scripts para facilitar essa importação: import.sh, escrito em shell script POSIX, e import.py, escrito em Python. Ambos permitem gerar um arquivo SQLite contendo todas as tabelas do MIMIC-IV a partir dos arquivos CSV ou CSV.GZ disponibilizados.

Requisitos

Para utilizar o script import.sh, são necessários:

- Shell POSIX compatível (ex: `bash`, `zsh`, `dash`)
- SQLite[1]

[1] `https://sqlite.org/index.html`

- gzip (geralmente já instalado por padrão em sistemas Linux, BSD e macOS)

Para utilizar o script `import.py`, é necessário:

- Python 3 instalado
- Biblioteca pandas[2]

Estrutura de diretórios

Os scripts devem estar localizados na mesma pasta onde estão os arquivos CSV do MIMIC-IV. A estrutura recomendada é:

```
path/to/mimic-iv/
|-- import.sh
|-- import.py
|-- hosp/
|   |- admissions.csv.gz
|   |- ...
|   \- transfers.csv.gz
\-- icu/
    |- chartevents.csv.gz
    |- ...
    \_ procedureevents.csv.gz
```

Execução dos scripts

Para gerar o banco de dados SQLite, basta executar um dos scripts desejados diretamente no terminal:

```
$ ./import.sh
```

ou

```
$ python import.py
```

A execução completa pode demorar, especialmente durante o carregamento da tabela `chartevents`, que é volumosa.

Ao final do processo, será gerado um arquivo `mimic4.db`, que pode ser aberto com qualquer ferramenta compatível com SQLite, como `DB Browser for SQLite`[3] ou diretamente via linha de comando usando o utilitário `sqlite3`:

```
$ sqlite3 mimic4.db
```

[2] https://pandas.pydata.org/
[3] https://sqlitebrowser.org/

Ajuste necessário no script `import.py`

Durante os testes com o script `import.py`, foi identificado um erro relacionado à conversão de colunas de data/hora para o tipo `datetime`. O trecho original do código utilizava o parâmetro `format='ISO8601'` na função `pandas.to_datetime`, conforme segue:

```
df[c] = pd.to_datetime(df[c], format='ISO8601')
```

Contudo, o argumento `format='ISO8601'` não é reconhecido diretamente pelo `pandas`, o que resultava no seguinte erro de execução:

```
ValueError: time data does not match format ISO8601
```

Para contornar esse problema e tornar a conversão mais robusta, a linha foi substituída por:

```
df[c] = pd.to_datetime(df[c], errors='coerce')
```

Essa alteração permite que o `pandas` identifique automaticamente os formatos de data/hora e, quando não for possível realizar a conversão, o valor inválido é substituído por `NaT` (*Not a Time*). Essa abordagem garante maior estabilidade durante o processamento dos arquivos CSV, especialmente em colunas como `charttime`, `admittime` e `dob`, que podem conter registros incompletos ou fora de padrão.

Considerações

O script `import.sh` define todos os campos como texto, o que pode limitar certos tipos de consultas mais complexas. Já o script em Python pode oferecer maior controle sobre os tipos de dados, mas consome mais memória e tempo durante a importação.

2.4. Estrutura do MIMIC-IV

A estrutura do MIMIC-IV foi projetada para refletir a complexidade do ambiente hospitalar, mantendo ao mesmo tempo a integridade relacional e a escalabilidade do banco de dados. A organização dos dados tem como objetivo facilitar a análise clínica e a pesquisa biomédica por meio da separação lógica das fontes de dados e pela utilização de identificadores anonimizados consistentes ao longo das tabelas.

2.4.1. Visão Geral da Arquitetura de Dados

O MIMIC-IV está organizado em dois grandes módulos principais: *hosp* e *icu*, representando respectivamente os dados do prontuário eletrônico hospitalar (EHR – *Electronic Health Record*) e do sistema de monitoramento clínico da UTI (*MetaVision*) [Johnson et al. 2023]. Essa separação modular reflete diretamente as origens dos dados e permite ao pesquisador focar em contextos específicos (internações gerais versus cuidados críticos), com maior controle e clareza.

O banco é composto por 31 tabelas no total, distribuídas entre os dois módulos, além de tabelas auxiliares de dicionário e mapeamento de conceitos clínicos [Johnson et al. 2023].

2.4.2. Módulo hosp

O módulo hosp agrega as informações mais amplas sobre os pacientes, coletadas ao longo de toda a jornada hospitalar. Esses dados provêm diretamente do EHR, e incluem desde informações administrativas até dados clínicos mais detalhados, como demonstrado nas Tabelas 2.1, 2.2 e Figura 2.8 [Johnson et al. 2023].

Tabela 2.1. Tabelas do módulo Hosp do MIMIC-IV e suas descrições (Colunas "MIMIC-IV v3.1 e MIMIC-IV Demo v2.2"representam o número de linhas preenchidas na tabela em cada base de dados) (Parte 1) [MIT Laboratory for Computational Physiology 2023]

Nome da Tabela	Descrição da Tabela	MIMIC-IV v3.1	MIMIC-IV Demo v2.2
admissions	Informações detalhadas sobre internações hospitalares.	546.028	275
diagnoses_icd	Diagnósticos codificados (ICD-9/ICD-10) faturados durante as hospitalizações.	6.364.488	4.506
drgcodes	Códigos de diagnóstico relacionados a grupos (DRG) faturados durante a internação.	761.856	454
d_hcpcs	Tabela de dimensão para *hcpcsevents*; descreve os códigos CPT (*Current Procedural Terminology*) de procedimentos médicos.	89.208	89.200
d_icd_diagnoses	Tabela de dimensão para *diagnoses_icd*; fornece a descrição de diagnósticos ICD-9/ICD-10.	112.107	109.775
d_icd_procedures	Tabela de dimensão para *procedures_icd*; fornece a descrição de procedimentos ICD-9/ICD-10.	86.423	85.257
d_labitems	Tabela de dimensão para *labevents*; contém a descrição dos exames laboratoriais.	1.650	1.622
emar	Registro eletrônico de administração de medicamentos (eMAR).	42.808.593	35.835
emar_detail	Informações suplementares das administrações registradas em *emar*.	87.371.064	72.018
hcpcsevents	Eventos faturados durante a hospitalização; inclui códigos CPT.	186.074	61
labevents	Resultados de exames laboratoriais a partir de amostras dos pacientes.	158.374.764	107.727

Tabela 2.2. Tabelas do módulo Hosp do MIMIC-IV e suas descrições (Colunas "MIMIC-IV v3.1 e MIMIC-IV Demo v2.2"representam o número de linhas preenchidas na tabela em cada base de dados) (Parte 2) [MIT Laboratory for Computational Physiology 2023]

Nome da Tabela	Descrição da Tabela	MIMIC-IV v3.1	MIMIC-IV Demo v2.2
microbiologyevents	Culturas microbiológicas.	3.988.224	2.899
omr	Contém informações diversas do prontuário eletrônico.	7.753.027	2.964
patients	Contém sexo, idade e data de óbito (se disponível) dos pacientes.	364.627	100
pharmacy	Informações sobre medicamentos prescritos: dosagem, fórmula, entre outros.	17.847.567	15.306
poe	Ordens médicas realizadas pelos profissionais de saúde.	52.212.109	45.154
poe_detail	Detalhes suplementares das ordens médicas registradas em *poe*.	8.504.982	3.795
prescriptions	Medicamentos prescritos.	20.292.611	18.087
procedures_icd	Procedimentos codificados realizados durante a hospitalização.	859.655	722
provider	Lista os identificadores de profissionais de saúde (deidentificados).	42.244	40.508
services	Serviço(s) hospitalar(es) responsáveis pelo cuidado do paciente.	593.071	319
transfers	Informações detalhadas sobre as transferências de unidade dos pacientes.	2.413.581	1.190

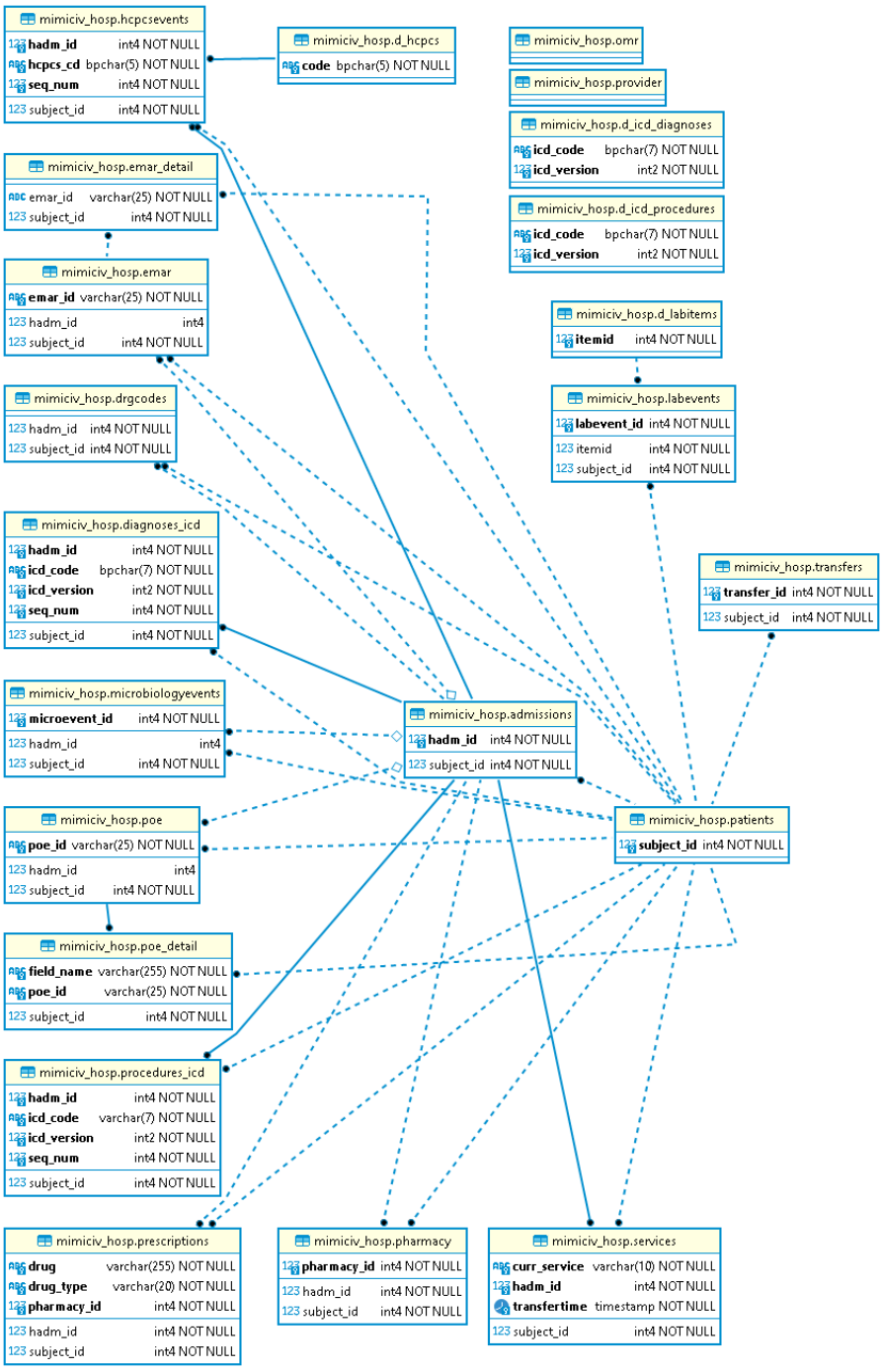

Figura 2.8. Diagrama ER do módulo Hosp do MIMIC-IV.

A granularidade dos dados varia: enquanto tabelas como *admissions* têm um registro por internação, outras como *labevents* podem conter centenas de entradas para um único paciente durante um único episódio hospitalar, refletindo medições realizadas várias vezes ao dia.

2.4.3. Módulo icu

Os dados contidos no módulo icu provêm do sistema de monitoramento contínuo da UTI (MetaVision), responsável por coletar medições detalhadas do estado clínico dos pacientes em cuidados intensivos, como demonstrado na Tabela 2.3 e Figura 2.9 [Johnson et al. 2023].

Tabela 2.3. Tabelas do módulo ICU do MIMIC-IV e suas descrições (Colunas "MIMIC-IV v3.1 e MIMIC-IV Demo v2.2"representam o número de linhas preenchidas na tabela em cada base de dados) [MIT Laboratory for Computational Physiology 2023]

Nome da Tabela	Descrição da Tabela	MIMIC-IV v3.1	MIMIC-IV Demo v2.2
caregiver	Lista os identificadores de profissionais de saúde (deidentificados) utilizados no módulo de UTI.	17.984	15.468
chartevents	Itens registrados durante a permanência na UTI; contém a maior parte das informações clínicas da UTI.	432.997.491	668.862
datetimeevents	Informações registradas com formato de data (ex: data da última diálise).	9.979.761	15.280
d_items	Tabela de dimensão que descreve os *itemid*; define os conceitos registrados nas tabelas de eventos da UTI.	4.095	4.014
icu_stays	Informações sobre a permanência na UTI, incluindo horários de admissão e alta.	94.458	140
ingredientevents	Ingredientes de administrações contínuas ou intermitentes, incluindo conteúdo nutricional e hídrico.	14.253.480	25.728
inputevents	Informações sobre infusões contínuas ou administrações intermitentes documentadas.	10.953.713	20.404
outputevents	Informações sobre saídas do paciente, como urina, drenagem, entre outras.	5.359.395	9.362
procedureevents	Procedimentos documentados durante a internação na UTI (ex: ventilação), mesmo que realizados fora da UTI (ex: exames de imagem).	808.706	1.468

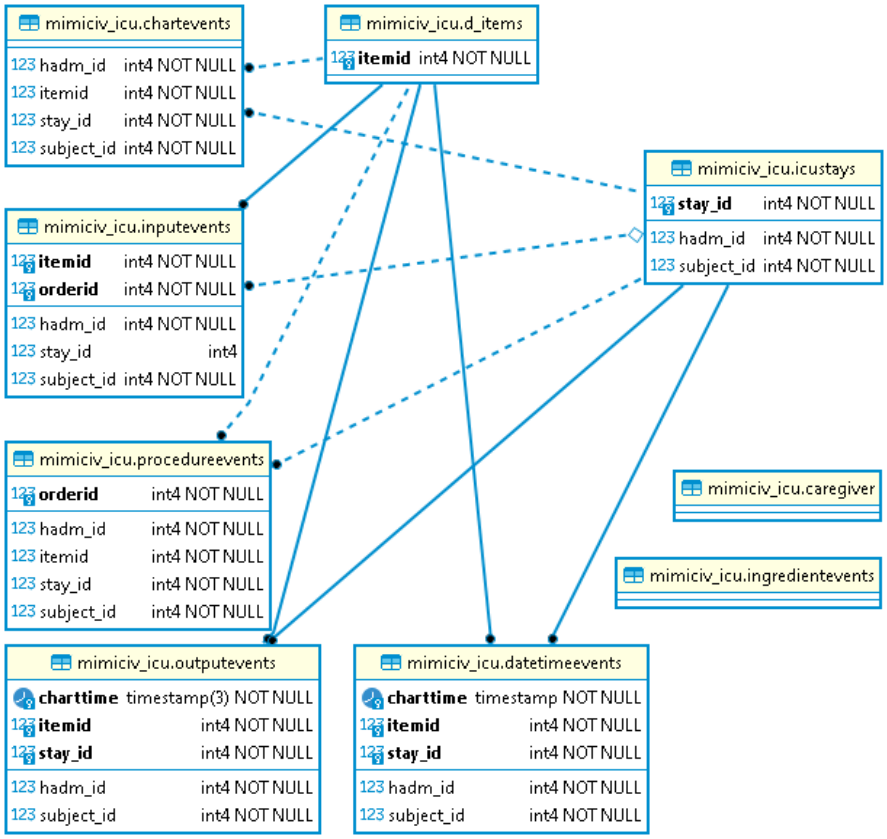

Figura 2.9. Diagrama ER do módulo ICU do MIMIC-IV.

É importante compreender que o módulo icu apresenta um volume de dados consideravelmente maior, pois representa medições contínuas – como frequência cardíaca, pressão arterial e saturação de oxigênio – coletadas a cada poucos minutos.

2.4.4. Relacionamentos e Identificadores

Uma característica essencial do MIMIC-IV é o uso de identificadores unificados que possibilitam a junção segura de dados entre as tabelas e os módulos. Os principais identificadores são:

- **subject_id**: identifica um paciente individual e é compartilhado entre todos os módulos.
- **hadm_id**: identifica uma internação hospitalar.
- **stay_id**: identifica uma internação na UTI.

Esses identificadores epseciais permitem, por exemplo, vincular exames laboratoriais (*labevents*), com eventos de internação (*admissions*), ou ainda cruzar dados de prescrição com registros de administração de medicamentos.

Além disso, o banco utiliza tabelas de apoio, como *d_items* e *d_lab_items*, que contém descrições dos itens presentes nas tabelas de eventos (*chartevents*, *labevents*, etc.), possibilitando a interpretação dos códigos numéricos.

2.4.5. Visões e Modelagem Relacional

A estrutura do MIMIC-IV é baseada nos princípios da modelagem relacional, seguindo normas clássicas de organização de dados em tabelas normalizadas. Esse modelo permite eliminar redundâncias e manter a consistência e integridade dos dados. No entanto, essa mesma estrutura exige que o usuário tenha familiaridade com conceitos fundamentais de bancos de dados relacionais para realizar consultas com precisão e eficiência. Neste ponto, é importante revisar alguns elementos centrais da modelagem relacional:

- **Tabelas**: Estruturas organizadas por colunas e registros (linhas), onde cada tabela representa uma entidade do modelo. Por exemplo, *patients* representa indivíduos, *admissions* representa episódios de internação, e *chartevents* contém observações clínicas contínuas.
- **Chaves primárias**: Um identificador único por registro, como *subject_id* em *patients*, que permite distinguir inequivocamente cada linha.
- **Chaves estrangeiras**: Atributos que criam uma ligação entre tabelas diferentes. Por exemplo, *subject_id* aparece em várias tabelas, estabelecendo relacionamentos entre elas.
- **Relacionamentos**: A modelagem relacional favorece uma estrutura onde as entidades estão interligadas por meio de identificadores desidentificados. Com isso, é possível navegar pelas informações do paciente entre diversas tabelas — da internação hospitalar até o nível dos sinais vitais.
- **Visões (Views)**: Embora o MIMIC-IV não forneça visões SQL pré-definidas, o uso de visões personalizadas é altamente recomendado durante as análises. Visões são consultas salvas que simulam tabelas e podem facilitar o reuso de lógicas complexas, principalmente em projetos com alto volume de *queries* recorrentes.

Esse modelo relacional torna o banco escalável, robusto e adequado para análises avançadas, mas impõe um desafio para iniciantes — especialmente aqueles com pouca familiaridade com SQL e com a lógica de bancos normalizados.

2.4.6. Granularidade, Temporalidade e Qualidade dos Dados

Um dos aspectos mais relevantes da estrutura do MIMIC-IV é sua granularidade. Ela varia consideravelmente entre as tabelas, impactando diretamente o tipo de análise que pode ser conduzida:

- **admissions**: Cada linha representa um episódio de internação hospitalar. A granularidade é uma entrada por internação.

- **chartevents**: Uma das tabelas mais volumosas. Cada linha representa uma observação clínica (sinais vitais, dados de monitoramento), com centenas ou milhares de entradas por paciente/dia.
- **noteevents**: Armazena notas clínicas em formato de texto livre. A granularidade depende do profissional e do tipo de nota registrada.

Essa granularidade exige atenção redobrada ao realizar junções (*joins*) entre tabelas, pois um relacionamento 1:n entre admissions e chartevents, por exemplo, pode gerar resultados inflados se a agregação não for realizada adequadamente.

Outro aspecto essencial é a temporalidade dos dados. A maioria das tabelas apresenta campos com marcações de tempo:

- **charttime**: representa o horário do evento registrado.
- **storetime**: horário em que o dado foi efetivamente armazenado no sistema.
- **starttime** e **endtime**: comuns em tabelas de prescrição ou procedimentos.

Esses campos permitem reconstruir a trajetória clínica do paciente ao longo do tempo, possibilitando análises longitudinais e temporais. No entanto, para preservar a privacidade dos indivíduos, todas as datas no MIMIC-IV foram alteradas com um deslocamento aleatório, diferente para cada paciente, mantendo a ordem e os intervalos entre os eventos [Johnson et al. 2023]. Esse deslocamento aleatório mantém a coerência temporal interna dos registros, mas inviabiliza análises agregadas por datas reais, como sazonalidade ou padrões semanais.

Além disso, há campos com valores ausentes ou registros generalizados (como códigos genéricos de procedimentos) devido a inconsistências no sistema original ou à política de privacidade.

2.4.7. Considerações sobre a estrutura do MIMIC-IV

A estrutura de dados do MIMIC-IV é, ao mesmo tempo, uma de suas maiores forças e um de seus maiores desafios. A modularidade entre os dados hospitalares (hosp) e os dados da UTI (icu) permite uma divisão lógica que reflete a origem dos registros, facilitando análises específicas por tipo de atendimento. Ao mesmo tempo, essa separação exige cuidado no planejamento das consultas, já que parte dos dados relevantes pode estar distribuída entre ambos os módulos.

A modelagem relacional com identificadores desidentificados aumenta a segurança e a privacidade dos dados, enquanto oferece uma estrutura robusta e flexível para o desenvolvimento de pesquisas. A granularidade variável e os aspectos temporais tornam o banco ideal para análises clínicas detalhadas, previsão de complicações, modelagem da permanência hospitalar, entre outros.

No entanto, o uso efetivo do MIMIC-IV requer um bom entendimento da arquitetura dos dados, da qualidade dos registros e das limitações impostas pela desidentificação.

Na próxima sessão, abordaremos como extrair e manipular esses dados de forma eficiente, utilizando comandos SQL, scripts em Python e práticas de modelagem de dados aplicadas à pesquisa clínica.

2.5. Recuperação de dados e análises exploratórias

2.5.1. MIMIC-IV Concepts

O MIMIC-IV é um banco de dados clínico complexo, composto por dezenas de tabelas brutas com estruturas distintas e, muitas vezes, difíceis de manipular diretamente. Para tornar a análise desses dados mais acessível e eficiente, a equipe mantenedora do projeto, em colaboração com a comunidade científica, desenvolveu o que se convencionou chamar de *Concepts* — consultas SQL que geram visões semânticas derivadas das tabelas originais.

As chamadas *Concept Tables* são visões organizadas que reúnem dados clínicos de interesse comum, como sinais vitais, exames laboratoriais, medicamentos administrados, entre outros. Esses conjuntos derivados são amplamente utilizados em pesquisas, pois oferecem padronização e facilitam a reprodutibilidade dos estudos.

Para auxiliar na criação dessas tabelas, o próprio repositório utilizado para importar o MIMIC-IV disponibiliza a pasta ./concepts-postgres, compatível tanto com PostgreSQL quanto com SQLite. Essa pasta contém scripts SQL prontos para gerar as *Concept Tables*, simplificando a manipulação dos dados e evitando erros frequentes associados à extração manual.

- O repositório oficial no GitHub pode ser acessado em: `https://github.com/MIT-LCP/mimic-code/tree/main/mimic-iv/concepts_postgres`
- As consultas estão organizadas por domínio clínico (ex: `demographics`, `firstday`, `measurement`, etc).
- O uso das *Concept Tables* é recomendado como ponto de partida para qualquer análise exploratória no MIMIC-IV.

2.5.2. Exemplo de utilização do *Concept*: recuperação de sinais vitais

Nesta subseção, mostraremos como construir uma sequência temporal com os principais sinais vitais registrados durante uma internação hospitalar no MIMIC-IV. Para isso, utilizaremos como exemplo uma internação selecionada aleatoriamente, da qual extrairemos frequência cardíaca, frequência respiratória, pressão arterial (sistólica, diastólica e média), temperatura corporal e saturação periférica de oxigênio (*SpO2*).

Etapa 1. Criação da *Concept Tables* vitalsign

Dentre os scripts de criação de *Concept Tables* disponibilizadas no repositório mencionado na seção anterior, a `Concept Table vitalsign` é a que consolida os principais sinais vitais medidos durante a internação dos pacientes. Essa tabela derivada facilita a

extração de informações como frequência cardíaca, pressão arterial, temperatura corporal, frequência respiratória e saturação de oxigênio (*SpO2*).

Para criar a `vitalsign`, basta executar o seguinte script SQL, adaptado do repositório oficial para criar uma nova tabela com essas informações:

```
CREATE TABLE vitalsign AS
SELECT
    ce.subject_id, ce.stay_id, ce.charttime
    , AVG(CASE WHEN itemid IN (220045) AND valuenum > 0 AND
       valuenum < 300 THEN valuenum END) AS heart_rate
    , AVG(CASE WHEN itemid IN (220179, 220050, 225309) AND
       valuenum > 0 AND valuenum < 400 THEN valuenum END) AS sbp
    , AVG(CASE WHEN itemid IN (220180, 220051, 225310) AND
       valuenum > 0 AND valuenum < 300 THEN valuenum END) AS dbp
    , AVG(CASE WHEN itemid IN (220052, 220181, 225312) AND
       valuenum > 0 AND valuenum < 300 THEN valuenum END) AS mbp
    , AVG(CASE WHEN itemid = 220179 AND valuenum > 0 AND
       valuenum < 400 THEN valuenum END) AS sbp_ni
    , AVG(CASE WHEN itemid = 220180 AND valuenum > 0 AND
       valuenum < 300 THEN valuenum END) AS dbp_ni
    , AVG(CASE WHEN itemid = 220181 AND valuenum > 0 AND
       valuenum < 300 THEN valuenum END) AS mbp_ni
    , AVG(CASE WHEN itemid IN (220210, 224690) AND valuenum > 0
       AND valuenum < 70 THEN valuenum END) AS resp_rate
    , ROUND(CAST(AVG(CASE
        WHEN itemid IN (223761) AND valuenum > 70 AND valuenum <
            120 THEN (valuenum - 32) / 1.8 -- converted to degC
           in valuenum call
        WHEN itemid IN (223762) AND valuenum > 10 AND valuenum <
            50 THEN valuenum END) -- already in degC, no
           conversion necessary
            AS NUMERIC), 2) AS temperature
    , MAX(CASE WHEN itemid = 224642 THEN value END) AS
       temperature_site
    , AVG(CASE WHEN itemid IN (220277) AND valuenum > 0 AND
       valuenum <= 100 THEN valuenum END) AS spo2
    , AVG(CASE WHEN itemid IN (225664, 220621, 226537) AND
       valuenum > 0 THEN valuenum END) AS glucose
FROM chartevents ce
WHERE ce.stay_id IS NOT NULL AND ce.itemid IN (
    220045 -- Heart Rate
    , 225309 -- ART BP Systolic
    , 225310 -- ART BP Diastolic
    , 225312 -- ART BP Mean
    , 220050 -- Arterial Blood Pressure systolic
    , 220051 -- Arterial Blood Pressure diastolic
    , 220052 -- Arterial Blood Pressure mean
    , 220179 -- Non Invasive Blood Pressure systolic
    , 220180 -- Non Invasive Blood Pressure diastolic
```



```
    , 220181 -- Non Invasive Blood Pressure mean
    , 220210 -- Respiratory Rate
    , 224690 -- Respiratory Rate (Total)
    , 220277 -- SPO2, peripheral
    -- GLUCOSE, both lab and fingerstick
    , 225664 -- Glucose finger stick
    , 220621 -- Glucose (serum)
    , 226537 -- Glucose (whole blood)
    -- TEMPERATURE
    -- 226329 -- Blood Temperature CCO (C)
    , 223762 -- "Temperature Celsius"
    , 223761  -- "Temperature Fahrenheit"
    , 224642 -- Temperature Site
    )
GROUP BY ce.subject_id, ce.stay_id, ce.charttime
```

Etapa 2. Extração dos sinais vitais

Com a tabela `vitalsign` criada, podemos agora recuperar os sinais vitais registrados ao longo da internação de um paciente específico. O trecho de código abaixo exemplifica como obter uma sequência temporal dos dados fisiológicos a partir da data de admissão hospitalar, calculando o tempo decorrido (`offset`, em minutos) entre o momento da medição e o início da internação.

A variável `hadm_id` deve ser substituída pelo identificador da internação hospitalar desejada. Como resultado, teremos um conjunto de vetores dos sinais vitais definidos na consulta a seguir.

```
SELECT
    a.hadm_id
    , (julianday(v.charttime) - julianday(a.admittime)) * 24 *
       60 AS offset
    , v.heart_rate, v.sbp, v.dbp
    , v.sbp_ni, v.dbp_ni
    , v.resp_rate
    , v.temperature
    , v.spo2
    , v.glucose
FROM admissions a
INNER JOIN icustays i
    ON a.hadm_id = i.hadm_id
LEFT JOIN vitalsign v
    ON i.stay_id = v.stay_id
WHERE a.hadm_id = {hadm_id}
```

Com essa consulta, foi obtido um exemplo dos vetores de sinais vitais extraídos de uma única internação hospitalar (`hadm-id`). Como pode ser observado na Tabela 2.4, cada linha representa um registro temporal associado a um *offset* em minutos desde a

admissão, e cada coluna corresponde a um dos sinais vitais disponíveis. Os valores ausentes (NaN) indicam que o dado correspondente não estava disponível no momento da coleta. Esses vetores serão posteriormente organizados e interpolados para composição de janelas temporais com resolução horária padronizada.

Tabela 2.4. Exemplo de amostra dos vetores de sinais vitais extraídos de uma internação hospitalar

hadm_id	offset	heart_rate	sbp	dbp	sbp_ni	dbp_ni	resp_rate	temperature	SpO_2	glucose
x	14310	80	115	66	NaN	NaN	15	NaN	100	NaN
x	14311	80	118	67	NaN	NaN	15	36.5	100	NaN
x	14321	80	104	60	NaN	NaN	17.50	NaN	100	NaN
x	14331	81	123	69	NaN	NaN	15	NaN	100	NaN
x	14336	70	113	62	NaN	NaN	9	NaN	100	NaN
x	14341	70	107	60	NaN	NaN	15	NaN	100	NaN
x	...	...	...	...	...	...	...	...	...	...

Etapa 3. Visualização dos sinais vitais

Com os dados de sinais vitais extraídos, o próximo passo é visualizá-los ao longo do tempo para facilitar a interpretação clínica e a análise exploratória. O código Python a seguir mostra como plotar essa série temporal utilizando a biblioteca `matplotlib`. A estratégia adotada permite representar a evolução dos parâmetros fisiológicos durante a internação hospitalar, com destaque para os principais sinais vitais.

O trecho de código é organizado em etapas:

- **Definição da Função `plot_numeric_data`**: Essa função recebe um eixo do matplotlib, um DataFrame com os dados de entrada, um dicionário de informações de plotagem e um tempo nulo opcional (usado para interromper linhas contínuas quando o paciente sai da UTI). Ela filtra dados ausentes, organiza por ordem cronológica (`offset`) e plota os dados com os parâmetros visuais definidos.

```
def plot_numeric_data(ax, df_input, plot_info, null_time=
    None):
     for i, (column, plot_args) in enumerate(plot_info.items
        ()):
         df = df_input[['offset', column]].copy().dropna()

         # insert a null when the patient leaves the ICU to
            break the line in the plot
         if null_time is not None:
             dff = pd.DataFrame([[null_time, None]], columns
                =['offset', column])
             df = pd.concat([df, dff], ignore_index=True)
         df.sort_values('offset', inplace=True)
         ax.plot(df['offset']/TIME_COEF, df[column], **
            plot_args)
```


- **Configurações Iniciais**: Define constantes como o `TIME_COEF`, usado para converter minutos em dias (`offset/1440`), facilitando a leitura dos eixos temporais, e configurações visuais da figura (tamanho, fontes, títulos etc.).

```
# === Initial Configurations ===
TIME_COEF = 24 * 60.0  # conversion from minutes to days
TIME_UNIT = 'days'
null_time = 8 * 24 * 60  # time used to separate distinct
    admissions

plt.rcParams.update({'font.size': 22})
fig = plt.figure(figsize=(20, 10))
gs = fig.add_gridspec(1, hspace=0.1)
ax0 = gs.subplots(sharex=True, sharey=False)

fig.suptitle(f'Vital Signs and Cardiac Markers - Patient ID:
    XXXXXXXX', fontsize=24, y=0.98)
ax0.set_xlim([0, 13])
```

- **Plotagem dos sinais vitais**:
  - **`Heart Rate`** e **`Respiratory Rate`** são plotados como linhas com marcadores circulares, utilizando a função `plot_numeric_data`.
  - **Pressão arterial (sistólica e diastólica)** é representada por uma área sombreada entre as curvas, indicando a faixa de variação da pressão.
  - **Temperatura corporal (°C)** é exibida com marcador em formato de losango, em duas fases (antes e depois do tempo nulo), permitindo distinguir diferentes internações, se aplicável.
  - **Saturação periférica de oxigênio (SpO_2)** é plotada com linha tracejada e marcadores quadrados.

```
# === Vital Signs ===
plot_info_vitals = OrderedDict([
    ['heart_rate', {'label': 'Heart Rate (bpm)', 'color':
       colors[0], 'marker': 'o', 'lw': 2.5, 'markersize':
       6}],
    ['resp_rate', {'label': 'Respiratory Rate (ipm)', 'color
       ': colors[1], 'marker': 'o', 'lw': 2.5, 'markersize':
        6}],
])
plot_numeric_data(ax0, vitals, plot_info_vitals, null_time=
    null_time)

# ===> Blood Pressure (shaded area)
bp = vitals[['offset', 'dbp', 'sbp']].dropna().copy()
plot_args = {'color': colors[3], 'alpha': 0.3}
bp_early = bp['offset'] < null_time
```

```
ax0.fill_between(bp.loc[bp_early, 'offset'] / TIME_COEF, bp.
   loc[bp_early, 'dbp'], bp.loc[bp_early, 'sbp'], label='
   Blood Pressure (mmHg)', **plot_args)
ax0.fill_between(bp.loc[~bp_early, 'offset'] / TIME_COEF, bp
   .loc[~bp_early, 'dbp'], bp.loc[~bp_early, 'sbp'], **
   plot_args)

# ===> Temperature
temp_early = temp['offset'] < null_time
ax0.plot(temp.loc[temp\_early, 'offset'] / TIME\_COEF, temp.
   loc[temp\_early, 'valuenum'], label='Temperature (C)',
   color=colors[2], marker='d', markersize=10, lw=2)
ax0.plot(temp.loc[~temp_early, 'offset'] / TIME_COEF, temp.
   loc[~temp_early, 'valuenum'], color=colors[2], marker='d'
   , markersize=10, lw=2)

# ===> SpO2
spo2_data = vitals[['offset', 'spo2']].dropna()
spo2_early = spo2_data['offset'] < null_time
ax0.plot(spo2\_data.loc[spo2_early, 'offset'] / TIME\_COEF,
   spo2_data.loc[spo2\_early, 'spo2'], label='SpO2 (%)',
         color=colors[7], marker='s', markersize=8, lw=2,
            linestyle='--')
ax0.plot(spo2_data.loc[~spo2_early, 'offset'] / TIME_COEF,
   spo2_data.loc[~spo2_early, 'spo2'],
         color=colors[7], marker='s', markersize=8, lw=2,
            linestyle='--')
```

- **Finalização**: Os elementos visuais do gráfico são refinados com legenda, rótulos, grades e ajustes de layout para melhor legibilidade.

```
ax0.legend(loc='upper right', bbox_to_anchor=(1.0, 1.0),
   fontsize=18)
ax0.set_ylabel('Values', fontsize=20)
ax0.set_xlabel('Days', fontsize=20)
ax0.grid(True, linestyle='--', alpha=0.7)

# === Final Layout ===
plt.tight_layout(rect=[0, 0, 1, 0.96])
```

A execução do código acima gera um gráfico consolidado que ilustra a evolução temporal dos sinais vitais do paciente ao longo do período de internação (Figura 2.10). Cada curva representa um parâmetro fisiológico distinto, com codificação de cores e marcadores que facilitam a identificação visual. A faixa sombreada referente à pressão arterial permite visualizar as variações entre os valores sistólico e diastólico. A divisão entre internações (ou fases distintas do atendimento) é indicada por grandes interrupções nas

linhas, controladas pelo parâmetro `null_time`. Essa visualização proporciona uma análise rápida do estado clínico do paciente ao longo do tempo e pode ser utilizada tanto para fins exploratórios quanto como base para métodos de modelagem preditiva.

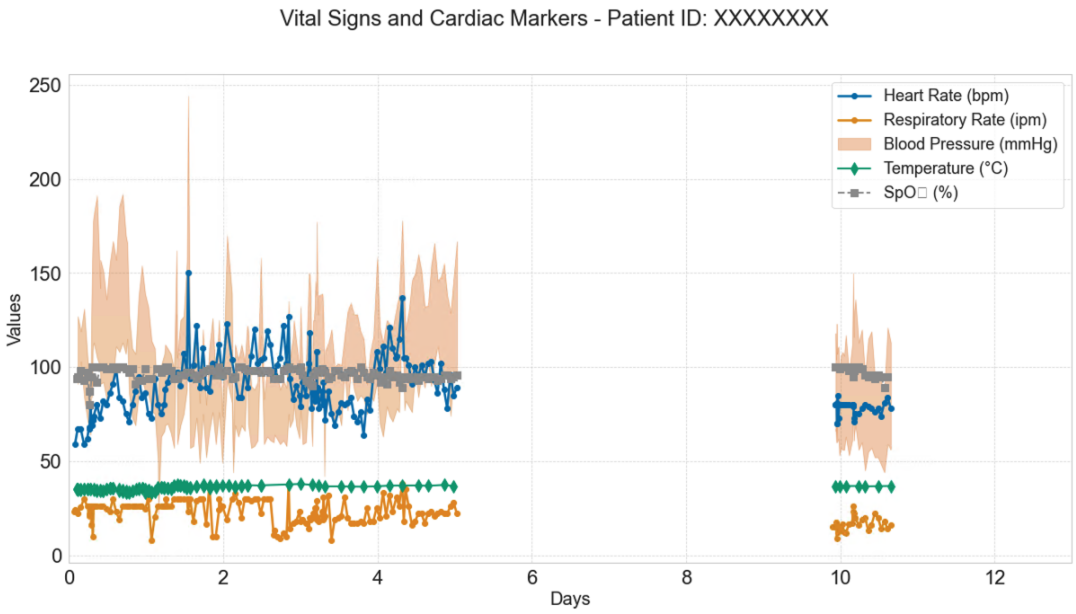

Figura 2.10. Visualização temporal dos sinais vitais ao longo da internação hospitalar para um paciente selecionado.

2.6. Aplicação prática com Python e SQL

Nesta sessão, você aprenderá a extrair um subconjunto específico de pacientes a partir do MIMIC-IV, utilizando Python com SQL. Este processo é útil para análises clínicas direcionadas, estudos estatísticos e desenvolvimento de modelos preditivos.

Objetivo

Selecionar admissões de pacientes com os seguintes critérios:

- Sexo: Masculino
- Idade: entre 20 e 35 anos no momento da admissão
- Diagnóstico principal: Pneumonia

Pré-requisitos

- Ter uma instância local do MIMIC-IV em um banco PostgreSQL ou arquivo local com o banco de dados em criado via SQLite
- Ter acesso autorizado ao MIMIC (via PhysioNet)
- Python instalado com as bibliotecas `pandas`, `sqlalchemy` ou `sqlite3`

Etapa 1: Importação das bibliotecas

Vamos começar importando as bibliotecas necessárias:

- `pandas`: Para manipulação de dados em DataFrames
- `sqlalchemy`: Para conexão e execução de comandos SQL em bancos de dados relacionais, necessário para utilização do PostgreSQL
- `sqlite3`: Para conexão e execução de comandos SQL em bancos de dados relacionais, necessário para utilização do SQLite

Caso ainda não tenha as bibliotecas instaladas, use o seguinte comando:

```
pip install pandas sqlalchemy
```

```
import pandas as pd
import sqlalchemy
import sqlite3
```

Etapa 2: Conexão com o Banco de Dados

Configure a conexão com o banco local que contém o MIMIC-IV.

PostgreSQL

```
def connect_postgres(user, password, host, port, db):
    url = f"postgresql://{user}:{password}@{host}:{port}/{db}"
    engine = sqlalchemy.create_engine(url)
    return engine
```

SQLite

```
engine = sqlite3.connect('mimic4.db') # Passando o caminho do
   arquivo do banco de dados como parâmetro
```

Etapa 3: Consulta SQL - Extração de pacientes

Nesta etapa, construímos uma consulta SQL para buscar pacientes do sexo masculino.

```
query_pacientes = """
SELECT
    subject_id,
    gender,
    anchor_age,
    anchor_year,
    anchor_year_group,
    dod
FROM patients
WHERE gender = 'M'
"""
```



```
df_pacientes = pd.read_sql(query_pacientes, engine)
```

Explicação das colunas utilizadas:

- **subject_id** é o identificador único atribuído a cada paciente.
- **gender** indica o sexo biológico do paciente, sendo `M` para masculino e `F` para feminino.
- **anchor_age** representa a idade aproximada do paciente no ano de referência definido em **anchor_year**.
- **anchor_year** é o ano utilizado como ponto de ancoragem temporal para os dados clínicos do paciente, permitindo anonimização.
- **anchor_year_group** agrupa os pacientes com base no período de seu ano de referência (por exemplo, 2008–2010).
- **dod** (date of death) indica a data de óbito do paciente, quando disponível.

Agora, vamos transformar todos os subject_ids retornados em uma string para que possa ser utilizada posteriormente durante a busca de informações referentes a admissão desses pacientes e também dos diagnósticos deles.

O subject_id é o identificador único atribuído à cada paciente. Com ele, podemos recuperar as informações de cada um deles.

```
# Transforma a lista de IDs em uma string formatada para o SQL

subject_ids = df_pacientes['subject_id'].unique().tolist()
subject_ids_str = '(' + ', '.join(str(sid) for sid in
   subject_ids) + ')'
```

Etapa 4: Consulta SQL - Extração de admissões

Agora, com auxílio dos subject_ids recuperados no passo anterior, vamos recuperar todas as admissões referente a estes pacientes.

```
query_admissoes = f"""
SELECT
    subject_id,
    hadm_id,
    admittime
FROM admissions
WHERE subject_id IN {subject_ids_str}
"""
df_admissoes = pd.read_sql(query_admissoes, engine)
```

Explicação das colunas utilizadas:

- **subject_id** é o identificador do paciente, como anteriormente.
- **hadm_id** é o identificador único de cada admissão hospitalar, permitindo a vinculação com outros eventos clínicos ocorridos durante a internação.
- **admittime** representa a data e hora da admissão do paciente no hospital.

Agora, vamos transformar todos os hadm_ids retornados em uma string para que possa ser utilizada posteriormente durante a busca de informações referentes aos diagnósticos.

O hadm_id é o identificador único atribuído à cada admissão. Com ele, podemos recuperar as informações de cada uma delas.

```
# Transforma a lista de IDs em uma string formatada para o SQL

hadm_ids = df_admissoes['hadm_id'].unique().tolist()
hadm_ids_str = '(' + ', '.join(str(hid) for hid in hadm_ids) + '
   )'
```

Etapa 5: Consulta SQL - Extração dos Diagnósticos dos Pacientes

Agora, com o auxílio dos subject_ids e hadm_ids obtidos anteriormente, vamos buscar todos os diagnósticos realizados nas admissões destes pacientes.

```
query_diagnosticos = f"""
SELECT
    d.subject_id,
    d.hadm_id,
    d.seq_num,
    d.icd_code,
    d.icd_version,
    dx.long_title as diagnosis
FROM diagnoses_icd d
JOIN d_icd_diagnoses dx ON dx.icd_code = d.icd_code AND dx.
   icd_version = d.icd_version
WHERE d.subject_id in {subject_ids_str}
AND d.hadm_id in {hadm_ids_str}
"""

df_diag = pd.read_sql(query_diagnosticos, engine)
```

Explicação das colunas utilizadas:

- **subject_id** e **hadm_id** identificam o paciente e sua internação, respectivamente.
- **seq_num** indica a ordem do diagnóstico dentro da admissão, sendo 1 geralmente o diagnóstico principal.
- **icd_code** é o código da Classificação Internacional de Doenças (CID), que identifica a condição diagnosticada.

- **icd_version** informa a versão da classificação utilizada (por exemplo, ICD-9 ou ICD-10).
- **long_title** (renomeado como **diagnosis**) fornece a descrição textual completa da condição médica codificada.

Etapa 6: Consulta SQL - Juntar Pacientes + Admissões + Diagnósticos

Agora que temos 3 dataframes contendo as informações sobre Pacientes, Admissões e Diagnósticos, precisamos unir tudo isso em um Dataframe único. Para isso, usaremos as colunas subject_id e hadm_id para realizarmos a união.

```
# Merge 1: pacientes + admissões
df_merged = pd.merge(df_pacientes, df_admissoes, on="subject_id"
    , how="inner")

# Merge 2: adicionar diagnósticos
df_merged = pd.merge(df_merged, df_diag, on=["subject_id", "
    hadm_id"], how="inner")
```

Etapa 7: Aplicar Filtros Específicos

Agora que temos todos os dados em mãos, é hora de limpar o nosso subconjunto para que fiquem apenas os casos de interesse:

- Sexo: Masculino (já filtrado durante a obtenção de dados dos pacientes)
- Idade: Entre 40 e 50 anos (no momento da admissão)
- Diagnóstico: Pneumonia

Você deve ter percebido que não existe uma coluna de idade do paciente. Para calcularmos a idade de cada paciente, precisamos utilizar uma regra de negócio aplicada durante o processo de anonimização dos dados.

A idade é calculada desta forma:

- birth_year = anchor_year - anchor_age
- age_at_admission = admit_time - birth_year

```
# Para SQLite, descomente a linha a seguir para transformar a
    coluna admittime para o formato datetime
# df_merged['admittime'] = pd.to_datetime(df_merged['admittime
    '])

# Calcular a idade real na admissão
df_merged['age_at_admission'] = df_merged['admittime'].dt.year -
    (df_merged['anchor_year'] - df_merged['anchor_age'])
```



```
# Ajustar para verificar se o paciente já fez aniversário no ano
    da admissão
df_merged['age_at_admission'] -= ((df_merged['admittime'].dt.
   month < 1) |
                                  ((df_merged['admittime'].dt.
                                     month == 1) &
                                   (df_merged['admittime'].dt.
                                      day < 1))).astype(int)
```

Agora que anexamos a coluna com a idade no momento da admissão ao nosso conjunto de dados, podemos aplicar os filtros:

- Primeiro, manteremos apenas as linhas em que os pacientes com a idade dentro do intervalo de interesse: entre 40 e 50 anos.
- Por último, manteremos apenas as linhas em que o Diagnóstico contenha o termo "Pneumonia".

```
# Filtro por idade
df_filtrado = df_merged[(df_merged["age_at_admission"] >= 40) &
   (df_merged["age_at_admission"] <= 50)]

# Filtro por diagnóstico
df_final = df_filtrado[
    df_filtrado["diagnosis"].str.contains("pneumonia", case=
       False, na=False)
]
```

Etapa 8: Exportar o Conjunto Final

Por fim, exportaremos o subconjunto obtido para um arquivo no formato .csv. Este arquivo pode ser usado para análises ou para auxiliar a extrair outras informações relevantes sobre os pacientes ou admissões nele contidos.

```
df_final.to_csv("subconjunto_mimiciv.csv", index=False)
print("Arquivo salvo como 'subconjunto_mimiciv.csv'")
```

Revisando: O que aprendemos?

- Como se conectar ao MIMIC-IV com Python
- Como buscar e combinar diferentes tabelas (pacientes, admissões, diagnósticos)
- Como aplicar filtros específicos para extrair um subconjunto de dados
- Como exportar resultados para CSV

Com essa implementação, extraímos um subconjunto específico e relevante de dados, pronto para ser utilizado em análises clínicas, estudos estatísticos ou como base para investigações mais aprofundadas dentro do próprio MIMIC-IV, utilizando os identificadores de pacientes (subject_id) e de admissões hospitalares (hadm_id). Embora o exemplo apresentado tenha sido intencionalmente simples, ele demonstra, de forma prática, como o MIMIC-IV pode ser explorado com o uso combinado de Python e SQL.

O MIMIC-IV é um banco de dados robusto e abrangente, capaz de sustentar uma ampla gama de aplicações — desde pesquisas epidemiológicas e desenvolvimento de modelos preditivos até estudos longitudinais e análises de desfechos clínicos. A flexibilidade e a riqueza das informações disponíveis tornam este recurso extremamente valioso para pesquisadores e profissionais da saúde interessados em ciência de dados aplicada à área médica.

2.7. Aspectos éticos na utilização do MIMIC

O uso de dados clinicos reais, como os presentes no MIMIC, envolve uma série de responsabilidades éticas fundamentais. Embora o MIMIC tenha sido cuidadosamente projetado para proteger a privacidade dos indivíduos, o acesso ao banco completo está condicionado ao cumprimento de normas de ética em pesquisa, conforme exigido por regulamentos internacionais como o Health Insurance Portability and Accountability Act (HIPAA), além das diretrizes de proteção de dados sensíveis adotadas pelo MIT e pela plataforma PhysioNet.

2.7.1. Requisitos para Acesso ao MIMIC-IV

Para obter acesso aos dados do MIMIC-IV, o pesquisador deve cumprir os requisitos descritos na sessão 2.3.2. Esse processo garante que o usuário compreende os princípios éticos envolvidos, como o respeito à privacidade dos pacientes, a confidencialidade dos dados e a proibição explícita de qualquer tentativa de reidentificação dos indivíduos presentes no banco.

2.7.2. Anonimização dos Dados e Conformidade com o HIPAA

O MIMIC é uma base de dados desidentificada, em conformidade com a HIPAA. A anonimização é realizada por meio de diversas estratégias [Johnson et al. 2024]:

- Remoção de identificadores diretos, como nomes, números de documentos, endereços e contatos.
- Substituição de datas reais por datas deslocadas aleatoriamente, mantendo a coerência temporal interna para análises longitudinais, mas inviabilizando qualquer mapeamento com calendários reais.
- Codificação de identificadores, como subject_id, hadm_id e stay_id, que são únicos, porém artificiais.
- Generalização de valores sensíveis, como datas de nascimento e faixas etárias extremas (pacientes com mais de 89 anos, por exemplo, são representados como tendo "91 anos").

Além disso, os dados são revisados periodicamente pela equipe do MIT-LCP para garantir que continuam dentro dos padrões de desidentificação seguros mesmo após atualizações da base.

2.7.3. Termo de Uso de Dados e Responsabilidade dos Pesquisadores

Ao aceitar o termo de uso (DUA) [PhysioNet 2025], o pesquisador assume legalmente o compromisso de:

- Não tentar identificar qualquer indivíduo ou instituição referenciada nos dados restritos do PhysioNet;
- Exercer todo o cuidado razoável e prudente para evitar a divulgação da identidade de qualquer indivíduo ou instituição referenciada nos dados restritos do PhysioNet em qualquer publicação ou outra comunicação;
- Não compartilhar o acesso aos dados restritos do PhysioNet com ninguém;
- Exercer todo o cuidado razoável e prudente para manter a segurança física e eletrônica dos dados restritos do PhysioNet;
- Se encontrar informações dentro dos dados restritos do PhysioNet que possam permitir a identificação de qualquer indivíduo ou instituição, reportar prontamente a localização dessa informação por e-mail para PHI-report@physionet.org, citando a localização específica da informação em questão;
- Solicitar o acesso aos dados restritos do PhysioNet com o único propósito de uso legítimo em pesquisa científica, utilizando o privilégio de acesso, se concedido, exclusivamente para esse propósito;
- Completar um programa de treinamento em proteções de sujeitos de pesquisa humana e regulamentações HIPAA, fornecendo a comprovação de que o fez;
- Indicar o propósito geral para o qual pretende usar o banco de dados na solicitação de acesso;
- Se os resultados forem divulgados publicamente, contribuir também com o código utilizado para produzir esses resultados para um repositório aberto à comunidade de pesquisa;
- Reconhecer que o acordo pode ser rescindido por qualquer uma das partes a qualquer momento, mas que as obrigações com relação aos dados do PhysioNet continuarão após a rescisão.

O não cumprimento desses requisitos pode resultar em penalidades institucionais e legais, além de comprometer a integridade de toda a comunidade científica usuária do MIMIC.

2.7.4. Utilização Ética

Neste documento, algumas demonstrações e exemplos práticos foram elaborados com base na versão completa do MIMIC-IV, obtida mediante autorização formal, conforme os requisitos do PhysioNet. No entanto, todas as atividades apresentadas são inteiramente reproduzíveis utilizando a versão Demo do MIMIC-IV, que é amplamente divulgada pela plataforma PhysioNet e livre de restrições de acesso ou obrigações éticas adicionais.

A versão Demo contém um subconjunto representativo dos dados reais, suficientemente anonimizados e reduzidos para fins de ensino, testes e exploração de ferramentas. Isso garante que qualquer participante do minicurso poderá reproduzir os exemplos propostos sem infringir regulamentos éticos ou a política de uso da base de dados.

Reforçamos, também, que nenhuma parte deste material propõe ou orienta o uso da versão completa do MIMIC-IV sem a devida autorização. Ao adotar a versão Demo como referência para reprodução dos tutoriais, este documento mantém seu compromisso com as boas práticas em pesquisa e com os princípios éticos fundamentais na manipulação de dados clínicos.

2.8. Conclusões

O MIMIC-IV é amplamente reconhecido como uma importante fonte de dados clínicos públicos, reunindo informações detalhadas sobre pacientes em contextos hospitalares e de terapia intensiva. Sua estrutura relacional, modular e com granularidade temporal permite análises em múltiplas escalas, desde investigações populacionais até o monitoramento clínico individualizado.

Neste capítulo, foram abordadas as etapas essenciais para acessar e utilizar o MIMIC-IV, incluindo a configuração de ambientes locais com PostgreSQL e SQLite. Por meio de exemplos práticos com SQL e Python, mostramos como realizar análises exploratórias e extrair subconjuntos clinicamente relevantes, fornecendo uma base sólida para pesquisas mais acessíveis, reproduzíveis e alinhadas aos objetivos de diversos estudos.

Embora o MIMIC-IV seja tecnicamente complexo, essa complexidade pode ser superada com o uso de ferramentas adequadas e a disseminação de boas práticas. Ao reduzir a barreira técnica de entrada, ampliamos o acesso ao banco de dados, permitindo que mais pesquisadores e profissionais possam utilizá-lo, impulsionando o avanço da ciência de dados em saúde. Essa democratização do acesso fortalece uma comunidade multidisciplinar comprometida com a medicina baseada em evidências e com a evolução contínua da prática clínica por meio de análises informadas e inovadoras.

Referências

[Fan et al. 2025] Fan, X., Xu, J., Ye, R., Zhang, Q., and Wang, Y. (2025). Retrospective cohort study based on the mimic-iv database: analysis of factors influencing all-cause mortality at 30 days, 90 days, 1 year, and 3 years in patients with different types of stroke. *Frontiers in Neurology*, 15.

[Johnson et al. 2024] Johnson, A., Bulgarelli, L., Pollard, T., Gow, B., Moody, B., Horng, S., Celi, L. A., and Mark, R. (2024). Mimic-iv.

[Johnson et al. 2023] Johnson, A. E. W., Bulgarelli, L., Shen, L., Gayles, A., Shammout, A., Horng, S., Pollard, T. J., Hao, S., Moody, B., Gow, B., Lehman, L.-W. H., Celi, L. A., and Mark, R. G. (2023). MIMIC-IV, a freely accessible electronic health record dataset. *Sci. Data*, 10(1):1.

[Johnson et al. 2016] Johnson, A. E. W., Pollard, T. J., Shen, L., Lehman, L.-W. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. *Sci. Data*, 3(1):160035.

[Jung et al. 2024] Jung, J., Kim, D., and Hwang, I. (2024). Exploring predictive factors for heart failure progression in hypertensive patients based on medical diagnosis data from the mimic-iv database. *Bioengineering*, 11(6):531.

[Lin et al. 2024] Lin, S., Lu, W., Wang, T., Wang, Y., Leng, X., Chi, L., Jin, P., and Bian, J. (2024). Predictive model of acute kidney injury in critically ill patients with acute pancreatitis: a machine learning approach using the mimic-iv database. *Renal Failure*, 46(1).

[MIT Laboratory for Computational Physiology 2023] MIT Laboratory for Computational Physiology (2023). MIMIC-IV Documentation. `https://mimic.mit.edu/docs/iv`. Acesso em: Abril de 2025.

[National Library of Medicine (US) 2025] National Library of Medicine (US) (2025). PubMed: Trending articles. `https://pubmed.ncbi.nlm.nih.gov/trending/`. Accessed: 2025-05-07.

[PhysioNet 2025] PhysioNet (2025). Physionet credentialed health data use agreement 1.5.0. `https://physionet.org/content/mimiciv/view-dua/3.1/`. Acesso em: 22 abr. 2025.

[Pérez-Tome et al. 2024] Pérez-Tome, J. C., Parrón-Carreño, T., Castaño-Fernández, A. B., Nievas-Soriano, B. J., and Castro-Luna, G. (2024). Sepsis mortality prediction with machine learning tecniques. *Medicina Intensiva (English Edition)*, 48(10):584–593.

[Sun et al. 2024] Sun, B., Man, Y.-l., Zhou, Q.-y., Wang, J.-d., Chen, Y.-m., Fu, Y., and Chen, Z.-h. (2024). Development of a nomogram to predict 30-day mortality of sepsis patients with gastrointestinal bleeding: An analysis of the mimic-iv database. *Heliyon*, 10(4):e26185.

