



OPEN ACCESS

EDITED BY

Emmanouil Spanakis,
Foundation for Research and
Technology Hellas (FORTH), Greece

REVIEWED BY

Michaela Th. Mayrhofer,
Papillon Pathways e.U., Austria
Yashpal Ramakrishnaiah,
Monash University, Australia

*CORRESPONDENCE

Benedikt Schmid
✉ schmid_b@ukw.de

RECEIVED 17 November 2025

REVISED 28 January 2026

ACCEPTED 17 February 2026

PUBLISHED 19 March 2026

CITATION

Schrader NB, Meißner B, Fischer P,
Röder D, Ertl M, Meybohm P and
Schmid B (2026) Extraction and
processing of intensive care chart data
from a patient data management system.
Front. Digit. Health 8:1747227.
doi: 10.3389/fgth.2026.1747227

COPYRIGHT

© 2026 Schrader, Meißner, Fischer,
Röder, Ertl, Meybohm and Schmid. This
is an open-access article distributed
under the terms of the [Creative
Commons Attribution License \(CC BY\)](#).
The use, distribution or reproduction in
other forums is permitted, provided the
original author(s) and the copyright
owner(s) are credited and that the
original publication in this journal is
cited, in accordance with accepted
academic practice. No use, distribution
or reproduction is permitted which does
not comply with these terms.

Extraction and processing of intensive care chart data from a patient data management system

Nikolas B. Schrader¹, Burkhard Meißner², Paul Fischer¹,
Daniel Röder¹, Maximilian Ertl², Patrick Meybohm¹ and
Benedikt Schmid^{1*}

¹Department of Anaesthesiology, Intensive Care, Emergency and Pain Medicine, University Hospital Würzburg, Würzburg, Germany, ²Service Center Medical Informatics, University Hospital Würzburg, Würzburg, Germany

Background: Routine clinical data captured in Patient Data Management Systems (PDMS) in intensive care and perioperative settings are an invaluable resource for clinical research. However, the proprietary, fragmented, and transaction-oriented architecture of many systems severely limits secondary data use and requires extensive Extract, Transform, and Load (ETL) processing.

Methods: We developed a modular, Python-based ETL framework that enables flexible, domain-specific extraction of high-frequency, multimodal PDMS data. The system provides reusable components for data retrieval, preprocessing, harmonization, and de-identification, allowing extraction methods to be adapted or extended without modifying the core architecture. Each clinical domain is represented through dedicated Pydantic models enforcing consistent output schemas, type constraints, and automated plausibility checks. SQLAlchemy abstracts database access, while structured preprocessing logic resolves common documentation inconsistencies and transforms heterogeneous PDMS entries into standardized representations.

Results: The framework produces reproducible, analysis-ready datasets through a transparent, auditable workflow. An integrated audit logger records extraction parameters, transformations, and derived fields, providing full traceability. Salted, irreversible pseudonymization is embedded directly into the pipeline, supporting compliance with the European General Data Protection Regulation (GDPR; German: Datenschutz-Grundverordnung, DSGVO) and Art. 27 of the Bayerisches Krankenhausgesetz (BayKrG). By encapsulating extraction logic in modular processing units with consistent validation and automated de-identification, the system replaces complex *ad hoc* queries with standardized, maintainable, and research-ready processes.

Conclusion: The presented framework overcomes substantial technical and regulatory barriers to the secondary use of PDMS data by operationalizing a governance-first extraction pipeline. Its modular architecture encapsulates site-specific PDMS queries in a bounded adapter layer, while keeping validation, pseudonymization, and audit logging portable and reusable across domains and installations. By embedding domain-level validation models, irreversible pseudonymization, and structured auditing, the framework enables reproducible, governance-compliant access to high-frequency intensive care data. Rather than requiring immediate alignment to a common data model, it provides a pragmatic foundation on which semantic and syntactic interoperability can be added incrementally as requirements and resources evolve.

KEYWORDS

anaesthesia, extract transform and load (ETL), intensive care medicine, patient data management system (PDMS), Python (programming language), SQL (structured query language)

1 Introduction

The secondary use of routine clinical data from electronic health records (EHR) is an increasingly valuable resource for clinical research (1, 2), quality improvement (3), and the development of decision-support systems (4). Intensive care medicine (ICM) and perioperative settings, in particular, generate high-frequency, multimodal data captured within Patient Data Management Systems (PDMS). While these systems facilitate accurate, detailed and flexible clinical documentation, they are not designed to provide direct access for research purposes. Consequently, Extract, Transform and Load (ETL) processes for secondary data use face significant methodological and technical challenges (5). Different individual approaches to address these challenges have been published before (6, 7).

For clarity, this paper uses the following terminology consistently: *Electronic Health Record (EHR)* refers broadly to hospital-wide digital patient record systems; *Patient Data Management System (PDMS)* refers to specialised intensive care/perioperative documentation systems that generate high-frequency bedside data; and *clinical data warehouse (CDW)* refers to an analytics-oriented repository populated via ETL from operational systems.

Access to and utilization of EHR data is subject to a plethora of laws and regulations. In Germany, the secondary use of hospital routine data is governed by strict legal frameworks, most notably the European General Data Protection Regulation (GDPR; Datenschutz-Grundverordnung, DSGVO) and the Bayerisches Krankenhausgesetz (BayKrG). Together, these regulations establish the legal basis for the pseudonymized secondary use of routine hospital data for scientific purposes, provided that appropriate technical and organizational safeguards are implemented.

The Patient Data Management System (PDMS) Copra (Copra v6, COPRA System GmbH, Berlin, Germany) is a modular platform that has been widely implemented in intensive care and perioperative settings (8, 9). Its primary function is the continuous, real-time documentation of clinical information, supporting direct patient care and ensuring medico-legal traceability. The system acquires high-frequency physiological signals (e.g., heart rate, blood pressure, ventilatory parameters), discrete clinical events (e.g., medication administrations,

laboratory results, procedures), as well as structured and unstructured data such as clinical scores and narrative notes.

These data points are stored within a relational database optimized for transactional integrity and operational performance, ensuring reliability and consistency of bedside documentation. However, this optimization is achieved at the expense of secondary data use. The underlying database schema is highly denormalized and fragmented, often featuring data redundancy and reliance on proprietary system-specific coding instead of standardized terminologies (e.g., LOINC, ATC). This leads to incomplete or inconsistently implemented data capturing logic. While these design choices safeguard clinical performance and medico-legal requirements, they present considerable barriers for analytical queries, research applications, and data interoperability.

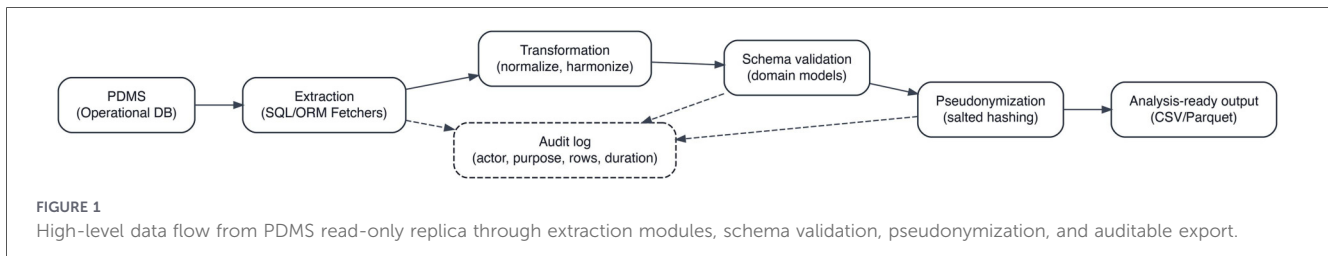
However, the database remains technically accessible, allowing custom extractions via direct queries and tailored transformation pipelines. This accessibility provides a foundation for research-oriented workflows, even though substantial preprocessing and harmonization are required to access the data suitable for secondary use.

This paper presents a structured ETL methodology for data from our institutional PDMS. We discuss the inherent data pipeline hurdles and propose replicable solutions to make high-quality, analysis-ready data accessible to the wider research community.

2 Methods

2.1 High-level architecture

Data is retrieved from a read-only PDMS database replica using SQL- or ORM-based fetchers, transformed into standardized intermediate representations, validated against predefined schemas, and pseudonymized before export as analysis-ready CSV or Parquet files. In parallel, audit-log records are generated for key processing steps to document actor, purpose, row counts, and runtime, thereby supporting traceable and governance-compliant secondary use of PDMS data (Figure 1).



```

-- Params
DECLARE @fallnr NVARCHAR(MAX) = :fallnr; -- e.g. '123'

SELECT
    fall.FALLNR AS case_number,
    patient.ID AS patient_id,
    patient.GESCHLECHT AS patient_sex,
    TRY_CONVERT(date, patient.GEB) AS patient_dob,
    CASE
        WHEN patient.GEB IS NULL THEN NULL
        ELSE CAST(DATEPART(year, patient.GEB) AS char(4))
    END AS patient_dob_yyyy,
    CASE
        WHEN fall.AUFN < patient.GEB THEN NULL
        ELSE DATEDIFF(YEAR, patient.GEB, fall.AUFN)
        - CASE
            WHEN DATEADD(YEAR, DATEDIFF(YEAR, patient.GEB, fall.AUFN),
                patient.GEB) > fall.AUFN
            THEN 1 ELSE 0
        END
    END AS
    patient_age_at_admission
FROM CO6_Medic_Data_Fall AS fall
JOIN CO6_Medic_Data_Patient AS patient
    ON fall.Patient_ID = patient.ID
    AND patient.deleted = 0
WHERE fall.FALLNR = @fallnr
    AND fall.deleted = 0
ORDER BY fall.FALLNR;
  
```

FIGURE 2
Straightforward query design for demographic data retrieval (SQL).

2.2 Data extraction

2.2.1 Technical access

To minimize the risk of affecting clinical system performance, data access was established via a secured ODBC connection to a redundant, read-only copy of the operational database. Initial extractions utilized direct Structured Query Language (SQL) (10). A basic application is shown in Figure 2.

Other, more relational data structures required a more complex extraction algorithm to ensure accurate outputs. Here, we describe the extraction of point-of-care blood gas analysis results (Figure 3).

As research demands expanded, a dedicated Python-based extraction framework was developed, using SQLAlchemy as an

abstraction layer (11). This framework enabled reusable workflows across data domains (vital signs, therapies, medications), automated preprocessing steps such as time stamp conversion (UTC to CET/CEST), and implemented clinically necessary logic to address inconsistencies in documentation.

For example, documentation of device usage or therapy intervals often lacked explicit termination markers (end times). Furthermore, reverification of ongoing therapies always produced additional entries for the same treatment, resulting in multiple records for a single continuous intervention. Identifying truly ongoing interventions vs. documentation gaps was therefore critical. To address this, an interval-based logic was introduced that heuristically cross-referenced therapy records with patient bed occupancy data to infer missing end times and to rejoin fragmented therapy intervals into continuous episodes.

This modular pipeline facilitated reproducibility, improved transparency, and reduced errors compared with *ad hoc* SQL queries. To comply with legal requirements, the framework included an auditing mechanism for user-specific request logging and an integrated hashing for sensitive parameters.

2.3 Data transformation and harmonization

Extracted PDMS data required systematic data cleaning and restructuring. A modular transformation framework was

implemented in Python, wherein each clinical domain (vital signs, medications, laboratory values, devices, demographics) was processed by a dedicated service module. Pydantic models defined the output, enforcing type safety, plausibility checks, and schema consistency.

2.3.1 Temporal processing

All data streams were normalized to Central European Time (CET/CEST). For event-based records such as laboratory results or medication administrations, only the time stamps were

```
-- Params
DECLARE @fallnr          nvarchar(50) = CONVERT(nvarchar(50), :fallnr);
-- e.g. '123'
DECLARE @from_ts        datetime2(0) = TRY_CONVERT(datetime2(0), :start_date);
-- optional
DECLARE @to_ts          datetime2(0) = TRY_CONVERT(datetime2(0), :end_date);
-- optional (exclusive)

IF OBJECT_ID('tempdb..#base_bga') IS NOT NULL DROP TABLE #base_bga;

-- Base for all non-SWISSLAB systems (e.g., ABL/POCT)
SELECT
    f.FALLNR,
    lab.val,
    lab.Unit,
    lab.DateTimeTo,
    labvar.ID          AS LabvarID,
    labvar.COPRAName,
    labvar.LaborSystem,
    CASE
        WHEN labvar.LaborSystem LIKE 'ABL%' OR labvar.LaborSystem LIKE 'POCT%'
    THEN 'POCT'
        ELSE labvar.LaborSystem
    END AS SystemGroup
INTO #base_bga
FROM CO6_Medic_Data_Labor AS lab
JOIN CO6_Medic_Config_LaborVariables AS labvar
    ON lab.LaborVariable = labvar.ID
JOIN (
    SELECT Patient_ID, FALLNR
    FROM CO6_Medic_Data_Fall
    WHERE FALLNR = @fallnr
) AS f
    ON lab.Parent_ID = f.Patient_ID
WHERE lab.deleted = 0
    AND lab.FlagCurrent = 1
    AND labvar.LaborSystem <> 'SWISSLAB'
    AND (@from_ts IS NULL OR lab.DateTimeTo >= @from_ts)
    AND (@to_ts IS NULL OR lab.DateTimeTo < @to_ts);

-- Dynamic column list (exluding Bemerkung)
DECLARE @cols_bga nvarchar(max);
SELECT @cols_bga = STRING_AGG(QUOTENAME(COPRAName), ',') WITHIN GROUP (ORDER
    BY COPRAName)
FROM (SELECT DISTINCT COPRAName FROM #base_bga WHERE COPRAName <>
```

FIGURE 3
(Continued)

```

'Bemerkung') d;

IF @cols_bga IS NULL
BEGIN
    SELECT CAST(NULL AS nvarchar(50)) AS FALLNR,
           CAST(NULL AS nvarchar(50)) AS SystemGroup,
           CAST(NULL AS datetime2(0)) AS DateTimeTo,
           CAST(NULL AS nvarchar(200)) AS [Type];
    RETURN;
END;

-- Pivot
DECLARE @sql_bga nvarchar(max) = N'
SELECT FALLNR, SystemGroup, DateTimeTo, [Type], ' + @cols_bga + N'
FROM (
    SELECT
        b.FALLNR,
        b.SystemGroup,
        b.DateTimeTo,
        (
            SELECT STRING_AGG(v.val, ', ')
            FROM (
                SELECT DISTINCT bx.val
                FROM #base_bga AS bx
                WHERE bx.SystemGroup = b.SystemGroup
                     AND bx.DateTimeTo = b.DateTimeTo
                     AND bx.COPRAName = ''Bemerkung''
            ) AS v
        ) AS [Type],
        b.COPRAName,
        CONCAT(b.val, COALESCE(' ' + b.Unit, '')) AS val
    FROM #base_bga AS b
    WHERE b.COPRAName <> ''Bemerkung''
) AS s
PIVOT (
    MAX(val) FOR COPRAName IN (' + @cols_bga + N')
) AS p
ORDER BY FALLNR, DateTimeTo, SystemGroup, [Type]';

EXEC sys.sp_executesql @sql_bga;

```

FIGURE 3
Complex query design for relational data structures and content-aware filtering, e.g. in point-of-care diagnostic applications (SQL).

converted. No additional aggregation or alignment was applied at this stage.

2.3.2 Vital signs

Most numeric values (e.g., blood pressure, heart rate, ventilator parameters) were consistently stored with predefined

units in the PDMS. Consequently minimal harmonization was required beyond plausibility filtering and time standardization.

2.3.3 Medications

Medication data required the most extensive processing. Documentation occurred in various formats (volume, dose, or rate), and drug concentrations were often not explicitly

available. Concentrations were therefore reconstructed from ingredient definitions using internal reference catalogs and classification algorithms. This step was essential to derive standardized application rates (e.g., mg/h, U/h) across projects.

2.3.4 Devices and therapies

Support devices such as extracorporeal membrane oxygenation (ECMO), dialysis, microaxial pumps (e.g., Impella systems), or intra-aortic balloon pump (IABP) were documented as intervals, but end times were frequently missing. To resolve this, therapy records were cross-referenced with patient bed occupancy and related device tables, allowing inference of active vs. terminated therapies.

2.3.5 Internal catalogs and variable definitions

Since international terminologies (e.g., LOINC, ATC) were unavailable in the source system, harmonization relied on internally maintained catalogs. These provided stable, project-wide definitions for drugs, laboratory parameters, and device attributes, serving as the central reference for variable identifiers to ensure consistency across service modules.

2.3.6 Output and validation

Each service produced a validated, patient-centered output model. Primary validation via pydantic enforced schema conformity, rejected clinically implausible values, and ensured consistency. Final outputs were stored in structured, analysis-ready formats (CSV or Parquet) with standardized variable naming. Secondary validation involved a sample-based auditing process, comparing extracted data against values displayed in the clinical PDMS front end to confirm consistency.

2.3.7 Usability and security

To make the extraction framework accessible beyond technically trained users, a lightweight front-end interface was implemented. The front-end interface incorporated fine-grained, user-based access control and audit logging, fulfilling institutional governance requirements as well as the traceability and accountability obligations defined under GDPR (Art. 5, 32) and Art. 27 BayKrG.

Predefined extraction templates for common research domains (e.g., epidemiology, vital parameters) provide standardized, validated data pipelines and minimize the risk of retrieving direct identifiers (e.g., patient name, date of birth). Extractions were tied to case numbers, enabling clinicians and researchers to obtain datasets relevant to specific trials or patient groups without requiring direct SQL interaction. Pseudonymization of sensitive identifiers was achieved by salted, irreversible hashing.

By abstracting low-level query complexity from end-user access, we reduced the risk of unintended database interference and promoted data integrity. In combination, the flexible back-end workflows and secure front-end interface created a scalable, reproducible, and governance-compliant process for secondary use of PDMS data.

Together, the secure front-end and modular back-end workflows provide auditable, role-restricted, and pseudonymized access to PDMS extracts in compliance with GDPR and Art. 27 BayKrG.

2.4 Data and code availability

A simplified version of the extraction framework demonstrating the demographic-data workflow is publicly available on GitHub (12). The full implementation in our hospital environment cannot be shared in its operational form, as extraction and transformation logic must be tailored to the institution-specific PDMS schema and is therefore neither directly portable nor readily interpretable outside our setting.

3 Results

3.1 Performance characteristics

To characterize operational performance, we measured throughput at two levels: * the framework's *processing layer* (schema validation, plausibility checks, and optional salted hashing), isolated from database access; and * *end-to-end extraction*, including database queries, domain-specific transformations, and validation. Unless noted otherwise, results summarize 5 repeated runs per cohort size, preceded by 1 warmup run. Only the repeated "run" phase is aggregated. To reduce dependence on a single fixed cohort and better reflect operational variability, case identifiers were **resampled per repeat** from a larger ID pool (seeded for reproducibility).

3.1.1 Processing-layer throughput

Table 1 reports processing time for schema validation and optional hashing, excluding database query execution and network latency. These measurements isolate framework overhead from installation-specific factors.

Throughput remained stable across record counts (~180,000 rows/s without hashing, ~156,000 rows/s with hashing), indicating near-linear scaling. Salted hashing adds approximately 13%–15% overhead—a predictable cost for irreversible pseudonymization.

3.1.2 End-to-end extraction throughput

Table 2 reports end-to-end extraction performance including database query execution, domain-specific transformations, and validation. Demographics represents a simple extraction (one row per case); drugs represents a transformation-heavy extraction with medication parsing, concentration reconstruction, and interval processing.

For demographics, throughput was in the low-thousands of records per second and increased for larger cohorts after warmup, reflecting reduced one-time overheads (e.g., caching and initialization) in the aggregated runs. For the more complex drugs resource—requiring transformation-heavy event processing—throughput remained stable at approximately 3,100–

TABLE 1 Processing-layer throughput for schema validation and optional salted hashing (single process).

| Records | Runtime (s), no hashing | Throughput (rows/s), no hashing | Runtime (s), with hashing | Throughput (rows/s), with hashing |
|---------|-------------------------|---------------------------------|---------------------------|-----------------------------------|
| 5,000 | 0.028 | 180,310 | 0.032 | 157,466 |
| 20,000 | 0.112 | 178,503 | 0.129 | 155,074 |
| 80,000 | 0.444 | 180,139 | 0.512 | 156,102 |

TABLE 2 End-to-end extraction throughput for demographics (simple) and drugs (transformation-heavy) resources.

| Resource | Cases | Records (mean) | Runtime (s) | Throughput (records/s) | Records/Case | Validation failures |
|--------------|-------|----------------|-------------|------------------------|--------------|---------------------|
| Demographics | 100 | 100.2 | 0.04 | 2,595 | 1.00 | 0% |
| Demographics | 500 | 501.4 | 0.20 | 2,833 | 1.00 | 0% |
| Demographics | 1,000 | 1,002.0 | 0.33 | 4,450 | 1.00 | 0% |
| Drugs | 100 | 26,648.6 | 6.72 | 3,973 | 266.49 | 0% |
| Drugs | 500 | 141,019.2 | 44.87 | 3,161 | 282.04 | 0% |
| Drugs | 1,000 | 283,905.6 | 91.90 | 3,115 | 283.91 | 0% |

4,000 records/s despite processing ~266–284 medication events per case. Validation failure rates were 0% across all runs, indicating that the extracted records conformed to the defined schema contracts and plausibility checks in this sample.

3.1.3 Multi-domain pipeline throughput

Table 3 reports performance for the complete multi-domain extraction pipeline **perio_pstudy** (demographics, drugs, operating room timestamps, and aggregated drug windows).

The near-linear scaling across case counts and the stable throughput (~3,800 records/s at 1,000 cases) demonstrate that the framework handles complex, multi-domain extractions predictably. For larger cohorts, the extraction layer automatically chunks identifier lists into bounded batches to avoid database parameter limits and maintain stable query execution.

3.1.4 Data quality observations

Benchmark runs also revealed upstream data quality characteristics in the sampled benchmark cohort: of 1,838 unique case numbers sampled, only 1 (0.05%) mapped to multiple patient identifiers, indicating rare but present identifier inconsistencies in the source system. No overlapping case admission windows were detected among 1,694 cases with complete timestamps, confirming temporal consistency in the source data for this sample.

3.2 Example usage of the pipeline for demographic data extraction, audit and de-identification

To illustrate end-to-end behavior, Figure 4 shows an exemplary demographic extraction for two cases, including de-identification and audit logging.

TABLE 3 Multi-domain pipeline throughput for perio_pstudy extraction (demographics + drugs + operating room timestamps + aggregations).

| Cases | Total records (mean) | Runtime (s) | Throughput (records/s) |
|-------|----------------------|-------------|------------------------|
| 100 | 46,519.0 | 14.83 | 3,138 |
| 500 | 190,057.8 | 57.04 | 3,349 |
| 1,000 | 367,049.8 | 96.01 | 3,824 |

The corresponding audit log (Figure 5) and CSV output (Figure 6) document all parameters, derived fields, and pseudonymized identifiers, demonstrating transparent and traceable access to analysis-ready PDMS data.

4 Discussion

4.1 Technical and methodological considerations

The extraction of routine clinical data from PDMS presents fundamental technical and methodological hurdles. While optimized for flexible clinical documentation and transactional performance, these systems inherently lack robust tools for analytical data extraction. The highly fragmented, denormalized, and proprietary data structures necessitate highly complex, advanced queries. As a result, extensive schema knowledge will be necessary during the initial steps of data extraction. During these initial steps, the SQL-based approach is highly effective for targeted retrieval of domains (e.g., demographics, ventilator parameters, laboratory results). However, direct SQL data extraction lacked a robust mechanism for role-based access control beyond user-specific database credentials, and logging user-specific extraction requests was necessary to comply with national regulations.

```

from pipeline.audit_logger import AuditLogger
from pipeline.extraction_pipeline import run_demographics

audit = AuditLogger(
    path="logs/audit.jsonl",
    include_id_samples=True, # Show sample IDs in internal audit logs
    id_hash_salt=None # Or set a salt to hash IDs in audit logs
)

df = run_demographics(
    by="cases",
    ids=["123", "234"],
    fields=[
        "case_number",
        "patient_sex",
        "patient_age_today",
    ],
    hash_salt="custom_demographics_salt", # Salt for hashing IDs in output
    out="out/demographics.csv", # Output CSV path or None to
disable export
    audit=audit, # AuditLogger instance or None to
disable auditing
    actor="n.schrader", # Identifier for the actor
triggering extraction
)

audit.close()

```

FIGURE 4

Application of the ETL pipeline to extract basic demographics, including audit logging and de-identification (Python).

logs/audit.jsonl

```

{
  "ts": "2025-11-12T10:39:57.193437+01:00",
  "actor": "n.schrader",
  "action": "fetch",
  "resource": "demographics",
  "by": "cases",
  "ids": {"count": 2, "ids": ["123", "234"]},
  "fields_requested": ["case_number", "patient_sex", "patient_age_today"],
  "fetch_fields": ["case_number", "patient_sex", "patient_date_of_birth"],
  "include_fields": ["case_number", "patient_sex", "patient_age_today"],
  "derived_added": ["patient_date_of_birth"],
  "hashed": true,
  "out": "out/demographics.csv",
  "out_format": null,
  "rows": 2,
  "duration_ms": 334.2735409969464
}

```

FIGURE 5

Output of the created audit log in JSON format.

out/demographics.csv

```
case_number,patient_sex,patient_age_today
a6e808094eb7,W,67
29a9c06d57ff,M,39
```

FIGURE 6

Output of the data extraction in comma-separated values format.

Although de-identification of sensitive parameters via hashing in SQL would be technically feasible, and query-based parameter minimization is obvious, practical implementation and configuration for each individual use case proved highly complex. Furthermore, the use of custom units, varied time formats, and system-specific coding schemes requires extensive data harmonization. Consequently, without dedicated ETL pipelines, these rich datasets remain inaccessible to the majority of clinical researchers. All these hurdles have been described before (5, 13–15).

4.2 Component portability and reusability

Although query logic and mappings are necessarily PDMS- and site-specific, the framework was designed to separate reusable infrastructure from institution-specific adapters. Table 4 summarizes the portability of key components as reflected by the project structure.

In practice, porting to a different PDMS installation requires updating the adapter layer (ORM models, domain-specific queries, and local registries), while leaving the core execution, validation, audit, and pseudonymization components unchanged.

4.3 Comparison with existing ETL approaches

The proposed framework targets governed extraction and transformation of high-frequency PDMS data into analysis-ready tables with explicit schema contracts and built-in pseudonymization and auditability. This focus differs from (i) pipelines built around public research datasets and (ii) common-data-model harmonisation projects. Table 5 summarizes these distinctions.

MIMIC-Extract provides a well-known extraction and preprocessing pipeline for the MIMIC-III dataset, emphasising reproducible feature generation on a fixed, public ICU schema (16). In contrast, PDMS installations in routine care use proprietary schemas and local variable registries. Therefore, our approach emphasises a clear adapter layer for site-specific extraction combined with a portable core for validation, traceability, and governance.

Common-data-model ecosystems such as OHDSI/OMOP prioritise semantic interoperability and cross-site comparability, typically requiring extensive mapping to standard vocabularies and a CDM schema (17). Full semantic interoperability requires continuous vocabulary management, extensive mapping effort, and ongoing maintenance as PDMS configurations evolve. For many ICU research and quality-

TABLE 4 Portability characteristics of framework components. High portability indicates reuse without code changes; medium portability requires configuration or mapping; low portability is institution- or vendor-specific.

| Component (repository layer) | Portability | Typical adaptation when porting |
|---|-------------|--|
| Database access abstraction (connection/) | High | Connection string/credentials/read-only replica |
| Time and hashing utilities (helpers/datetime_helpers.py , helpers/hashing.py) | High | Time zone policy, salt management, governance rules |
| Execution layer (registry, batching, export endpoints) (pipeline/) | High | Reuse as-is; register site-specific resources/adapters |
| Audit logging (pipeline/audit_logger.py) | High | Log sink, retention/rotation, ID sampling & hashing policy |
| Output contracts and plausibility checks (schemas/) | Medium | Extend/adjust schemas and plausibility rules per site |
| Local registries and catalogs (constants/) | Low | Update variable IDs, internal catalogs, mixture tables |
| PDMS schema models (models/) | Low | Replace with vendor/site-specific ORM models |
| Domain extractors (methods/) | Low | Rewrite queries and joins for local PDMS schema |

improvement use cases, the marginal benefit of immediate CDM conversion is outweighed by the operational cost and risk of introducing mapping errors. Our framework can serve as a preceding step by producing quality-controlled, pseudonymised exports that can then be mapped to a CDM when needed.

Finally, interoperability standards such as SMART on FHIR facilitate app-level integrations where FHIR interfaces are available (18). However, for many PDMS-backends, direct relational extraction remains necessary for high-frequency bedside signals and vendor-specific documentation structures.

4.4 Limitations and trade-offs

The framework does not aim to produce a fully standardized interoperability output (e.g., OMOP CDM or FHIR resources) directly from the PDMS. This is not an omission but a deliberate separation of concerns: PDMS databases are often proprietary and locally customized, and forcing full semantic harmonization at the point of extraction increases implementation complexity, reduces transparency, and slows iteration. Our approach therefore prioritizes traceability (audit logs), reproducibility (schema contracts), and usability (rapid integration by domain experts), while

TABLE 5 Comparison of the proposed framework with established clinical ETL approaches.

| Framework | Primary target | Standardization | Audit integration | Pseudonymization | Source access |
|--------------------|------------------------|-------------------|-------------------|------------------|----------------------|
| OMOP CDM ETL tools | Multi-site research | OMOP vocabularies | Limited | External | Full schema required |
| FHIR exporters | Interoperability | FHIR resources | Variable | External | API-based |
| MIMIC-Extract | Research (public data) | Custom | None | Pre-applied | Public dataset |
| i2b2 ETL | Research networks | i2b2 ontology | Basic | External | Star schema |
| Proposed framework | Institutional | Internal catalogs | Integrated | Embedded | Direct database |

OMOP, observational medical outcomes partnership; FHIR, fast healthcare interoperability resources; i2b2, informatics for integrating biology and the bedside.

treating interoperability as an optional downstream transformation step.

In secondary-use contexts governed by GDPR, the ability to demonstrate who accessed which data, for what purpose, and how records were transformed is operationally critical. By embedding audit logging and pseudonymization directly into the extraction pipeline, the framework provides governance guarantees that many general-purpose ETL stacks or ad-hoc scripts do not. Standardization can still be applied later, but governance must be established at the point of data access.

Although the output is not a universal interoperability standard, each domain export follows a defined explicit schema contract with validation and plausibility checks. This provides a strong form of syntactic consistency across extractions and enables incremental harmonization (e.g., mapping selected domains to OMOP or FHIR profiles) without destabilizing the extraction layer.

A practical strength of the architecture is its low barrier to adoption. The modular design, explicit schema definitions, and auditable execution allow integration and maintenance by a single clinically trained developer with limited institutional overhead. This is often the only realistic pathway for initiating secondary-use pipelines in resource-constrained environments.

Future work will focus on optional mapping modules to standardize representations (e.g., OMOP/FHIR) for selected domains, leveraging the existing schema contracts and audit trail to keep transformations traceable.

4.5 Ethical and legal considerations

All data handling complied with the European General Data Protection Regulation (GDPR; German: Datenschutz-Grundverordnung, DSGVO). The processing of special categories of personal data for scientific research is permitted under Art. 9(2)(j) GDPR in conjunction with appropriate safeguards, while the general conditions for lawful processing are governed by Art. 6(1) GDPR. In Bavaria, the specific legal basis enabling hospital-internal research with pseudonymized routine data is Art. 27 Bayerisches Krankenhausgesetz (BayKrG), which permits the secondary use of hospital data for scientific purposes without individual patient consent, provided that suitable technical and organizational measures are implemented. These provisions are complemented by the Bundesdatenschutzgesetz (BDSG) and, in a more general manner, the Bayerisches Datenschutzgesetz (BayDSG), which

establish additional requirements for processing personal data in public institutions. At the federal level, the Gesundheitsdatennutzungsgesetz (GDNG) introduces further provisions for the use of health data in research and quality-improvement contexts, thereby reinforcing the legal framework established by the DSGVO and BayKrG.

In line with Art. 32 GDPR and the security requirements defined in these laws, state-of-the-art technical and organizational measures were applied to ensure confidentiality, integrity, and availability of the data. Patient identifiers were removed at the time of extraction and replaced with salted, irreversible pseudonyms. All data linkage across domains was conducted exclusively within this pseudonymized environment.

Access to raw datasets was limited to authorized staff bound by professional confidentiality and institutional governance rules. All extraction and transformation steps were performed within a secure computing environment physically located at the university hospital, with no transfer of personal data to external servers or cloud infrastructure. Audit logs documented every data access and pipeline execution, ensuring full traceability.

5 Conclusion

In conclusion, routine data stored in PDMS contain unique opportunities for clinical and translational research, and the development of automation and decision-support systems. However, secondary use requires overcoming substantial technical and methodological barriers. The approach presented here offers a reproducible methodology for extracting, transforming, harmonizing, and validating PDMS data, making them accessible to a wider research community while maintaining compliance with ethical and legal requirements. By prioritizing governance, auditability, and schema contracts over premature standardization, the framework provides a practical foundation upon which interoperability efforts can be built incrementally.

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: https://github.com/DataScienceUKW/pdms_data_processing_pipeline.

Author contributions

NS: Software, Writing – original draft, Project administration, Methodology, Conceptualization, Writing – review & editing. BM: Writing – review & editing, Software. PF: Writing – review & editing. DR: Writing – review & editing, Conceptualization. ME: Software, Writing – review & editing. PM: Writing – review & editing. BS: Conceptualization, Project administration, Supervision, Writing – original draft, Writing – review & editing.

Funding

The author(s) declared that financial support was not received for this work and/or its publication.

Conflict of interest

The author(s) declared that this work was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

- Moor M, Bennett N, Plečko D, Horn M, Rieck B, Meinshausen N, et al. Predicting sepsis using deep learning across international sites: a retrospective development and validation study. *eClinicalMedicine*. (2023) 62:102124. doi: 10.1016/j.eclinm.2023.102124
- Schneeweiss S, Patorno E. Conducting real-world evidence studies on the clinical outcomes of diabetes treatments. *Endocr Rev*. (2021) 42(5):658–90. doi: 10.1210/endo/bnab007
- Hoque DME, Kumari V, Hoque M, Ruseckaite R, Romero L, Evans SM. Impact of clinical registries on quality of patient care and clinical outcomes: a systematic review. *PLoS One*. (2017) 12(9):e0183667. doi: 10.1371/journal.pone.0183667
- Solomon J, Dauber-Decker K, Richardson S, Levy S, Khan S, Coleman B, et al. Integrating clinical decision support into electronic health record systems using a novel platform (EvidencePoint): developmental study. *JMIR Form Res*. (2023) 7:e44065. doi: 10.2196/44065
- Holmes JH, Beinlich J, Boland MR, Bowles KH, Chen Y, Cook TS, et al. Why is the electronic health record so challenging for research and clinical care? *Methods Inf Med*. (2021) 60(01/02):032–48. doi: 10.1055/s-0041-1731784
- Maletzky A, Böck C, Tschoellitsch T, Roland T, Ludwig H, Thumfart S, et al. Lifting hospital electronic health record data treasures: challenges and opportunities. *JMIR Med Inform*. (2022) 10(10):e38557. doi: 10.2196/38557
- Daniel Boie S, Meyer-Eschenbach F, Schreiber F, Giesa N, Barrenetxea J, Guinemer C, et al. A scalable approach for critical care data extraction and analysis in an academic medical center. *Int J Med Inf*. (2024) 192:105611. doi: 10.1016/j.ijmedinf.2024.105611
- Zuber A, Kumpf O, Spies C, Höft M, Deffland M, Ahlborn R, et al. Does adherence to a quality indicator regarding early weaning from invasive ventilation improve economic outcome? A single-centre retrospective study. *BMJ Open*. (2022) 12(1):e045327. doi: 10.1136/bmjopen-2020-045327
- Klopfenstein SAI, Flint AR, Heeren P, Prendke M, Chaoui A, Ocker T, et al. Developing a scalable annotation method for large datasets that enhances alarms with actionability data to increase informativeness: mixed methods approach. *J Med Internet Res*. (2025) 27:e65961. doi: 10.2196/65961
- ISO/IEC. Information technology — database languages — sQL — part 1: framework (SQL/framework). In: *ISO/IEC 9075-1:2023(E)*. 6th ed. Geneva, Switzerland: International Organization for Standardization (2023).
- Bayer M. SQLAlchemy, python SQL toolkit and object relational mapper. (2023). Available online at: <https://www.sqlalchemy.org/> (Accessed November 01, 2025).
- Schrader NB. Pdms_data_processing_pipeline. (2025). Available from: Available online at: https://github.com/DataScienceUKW/pdms_data_processing_pipeline (Accessed November 01, 2025).
- Lamer A, Moussa MD, Marcilly R, Logier R, Vallet B, Tavernier B. Development and usage of an anesthesia data warehouse: lessons learnt from a 10-year project. *J Clin Monit Comput*. 2023;37(2):461–72. doi: 10.1007/s10877-022-00898-y
- Priou S, Kempf E, Flicoteaux R, Jankovic M, Chatellier G, Tournigand C, et al. “Where have my patients gone?”: a simulation study on real-world data processing in clinical data warehouses. *Health Policy Technol*. (2024) 13(3):100893. doi: 10.1016/j.hlpt.2024.100893
- Albu E, Gao S, Stijnen P, Rademakers FE, Van Bussel BC, Collyer T, et al. Challenges and recommendations for electronic health records data extraction and preparation for dynamic prediction modeling in hospitalized patients: practical guide and tutorial. *J Med Internet Res*. (2025) 27:e73987. doi: 10.2196/73987
- Wang S, McDermott MB, Chauhan G, Ghassemi M, Hughes MC, Naumann T. MIMIC-extract: a data extraction, preprocessing, and representation pipeline for MIMIC-III. *Proceedings of the ACM Conference on Health, Inference, and Learning* (2020). p. 222–35
- OHDSI Community. *The Book of OHDSI: Observational Health Data Sciences and Informatics*. OHDSI (2019). Available online at: <https://ohdsi.github.io/TheBookOfOhdsi/> (Accessed November 01, 2025).
- Mandel JC, Kreda DA, Mandl KD, Kohane IS, Ramoni RB. SMART on FHIR: a standards-based, interoperable apps platform for electronic health records. *J Am Med Inform Assoc*. (2016) 23(5):899–908. doi: 10.1093/jamia/ocv189

Generative AI statement

The author(s) declared that generative AI was not used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher’s note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.