HSLU Hochschule Luzern

Database Management for Data Scientist

Clinical Notes Database

Team AC:
Carlos Arzaga
carlos.arzaga@stud.hslu.ch
Andrea Valle
andrea.valle@stud.hslu.ch

Professors:
Luis Teran

March 12, 2024

Contents

1 Introduction & Initial Situation

The road to complete digitization in the Swiss healthcare system is a long one. Hospitals are the main stakeholders that hold the keys to completing the process. With the increasing adoption of artificial intelligence (AI) across the healthcare industry, natural language processing (NLP) models could be a catalyst for addressing this problem. NLP is a branch of AI, which can extract complex clinical data by translating free-form text into a standard form.

In this project, we are evalutating the feasability of integrating the capabilities of NLP with relational databases to create a system capable of processing and storing clinical notes. Specifically, we will use two specialized NLP subfields and demonstrate their integration into a database framework.

Firstly, we use Named Entity Recognition (NER), employing a ClinicalBERT transformer (Alsentzer et al., 2019) to identify and extract various entities such as problems, medications, and conditions from clinical notes. Secondly, we utilize a T5 (Text-to-Text Transfer Transformer), as mentioned in (Raffel et al., 2020), for summarizing the content of these notes. This approach allows us to efficiently handle and organize complex clinical data.

2 Objective

We will be working on the modeling, loading, transforming, querying, and optimization of a medical clinical notes database that will be integrated with the currently existing electronic health record (EHR) system for the University of Zurich Hospital (USZ). This is a proof of concept, that if successful could potentially expand to a continued partnership to build up the digitization of other areas within the hospital.

Through this, we will gain experience using SQL database technologies and directly applying them to a practical use case. The project outcome will assist in making informed decisions through data visualizations using a Business Intelligence (BI) tool. We will be utilizing MySQL as our SQL database and Metabase as our visualization tool, though in specific situations, we may explore alternative products within the same technology category for our project. The entire project will be implemented using database technologies. In the end, our goal is to provide a dashboard that would allow clinicians to see accurate and up-to-date clinical information on their patients.

3 Project Idea and Use Case

3.1 Motivation

Driven by the ongoing digital revolution, particularly in Switzerland's healthcare sector where programs are being designed to enhance digital transformation (DigitalSwitzerland, 2023). Furthermore, a study by MDPortals in 2021 highlighted that a staggering 80% of healthcare data remains unstructured (MDPortals, 2021). This project seeks to address and streamline unstructured clinical notes, offering an innovative solution to harness the potential for improving healthcare outcomes.

3.2 Decision Support

With clinical notes there are a few pain points for clinicians that generally take up the time, on average, physicians spend an additional 1.84 hours per day on electronic health record (EHR) documentation outside of regular office hours. This underscores the excessive time commitment required for EHR documentation, which impacts patient care (Gaffney et al., 2022). Some of this time is spent summarizing clinical notes. The second pain point is identifying key features from clinical notes such as the most relevant medical condition, medication, family history, etc. for each patient.

3.3 Data

The data set used for our project was obtained from Kaggle.com (Kaggle, 2022). The de-identified data was originally provided by the National Board of Medical Examiners (NBME) who sponsored a Kaggle competition called "NBME - Score Clinical Patient Notes". The competition aimed to develop automated solutions for scoring clinical text, such as patient notes, using NLP techniques. Specifically, the provided text data originates from the USMLE® Step 2 Clinical Skills examination, a test for medical licensure. This examination evaluates a medical trainee's skill in identifying important clinical information during interactions with standardized patients. The data consists of patient notes containing patient histories with various, features, and a set of training and testing files to be used to train the NLP models. We chose this data as it was open source and is of a similar structure in that clinical notes are taken by the physicians at USZ. Furthermore, we generated synthetic data, including fictitious patient identifiers and their related demographic information..

The clinical notes/summaries data support decisions related to patient diagnosis and treatment planning. By extracting key information from these notes, our database aids in faster and more accurate clinical decision-making. The results, in the form of structured summaries or extracted features from the notes, are presented in a user-friendly metabase dashboard. This includes visualizations of patient histories, critical medical conditions, and current medication lists. Users, primarily healthcare professionals, can interact with the data through the intuitive dashboard. Which involves them searching for specific patient notes, filtering data based on certain criteria, and perhaps modifying or annotating the summarized notes for further clarification or correction.

Our use case necessitates clinical notes and patient records, which include symptoms, treatment, medication history, and various patient demographics. This data would typically come from the hospital's clinical database and primarily the EHR system. It is used to improve patient care, support clinical decision-making, and facilitate healthcare research. The primary administrators of the data include healthcare providers, hospital administrators, IT professionals, and potentially, patients. Access is strictly role-based. Healthcare providers have primary access to patient-specific data which includes in-house researchers, while IT staff have broader access to maintenance and development integration with the database we develop. In the future patient access might be limited to their records, however, that would be left and is defined by the data regulations implemented at USZ.

Compliance with regulations will comply with the standards that are currently being developed as part of the Swiss Personalized Health Network (SPHN). Data security is ensured through encryption, secure user authentication, and regular audits. Data integrity is maintained through

backups and redundancy systems. Once the integration of our database is configured to USZ's EHR system, the end goal would be to integrate with the SPHN Connector. This Connector is a dockerized solution facilitating the building of pipelines for converting data from sources into graph data based on an RDF schema, aligning with the SPHN Framework (SPHN Semantic Framework, 2023). This envisioned integration architecture is depicted in Figure 1, while the core elements of the SPHN Connector are illustrated in Figure 2. However, these details of this architecture are beyond the scope of our project and are included for conceptual understanding

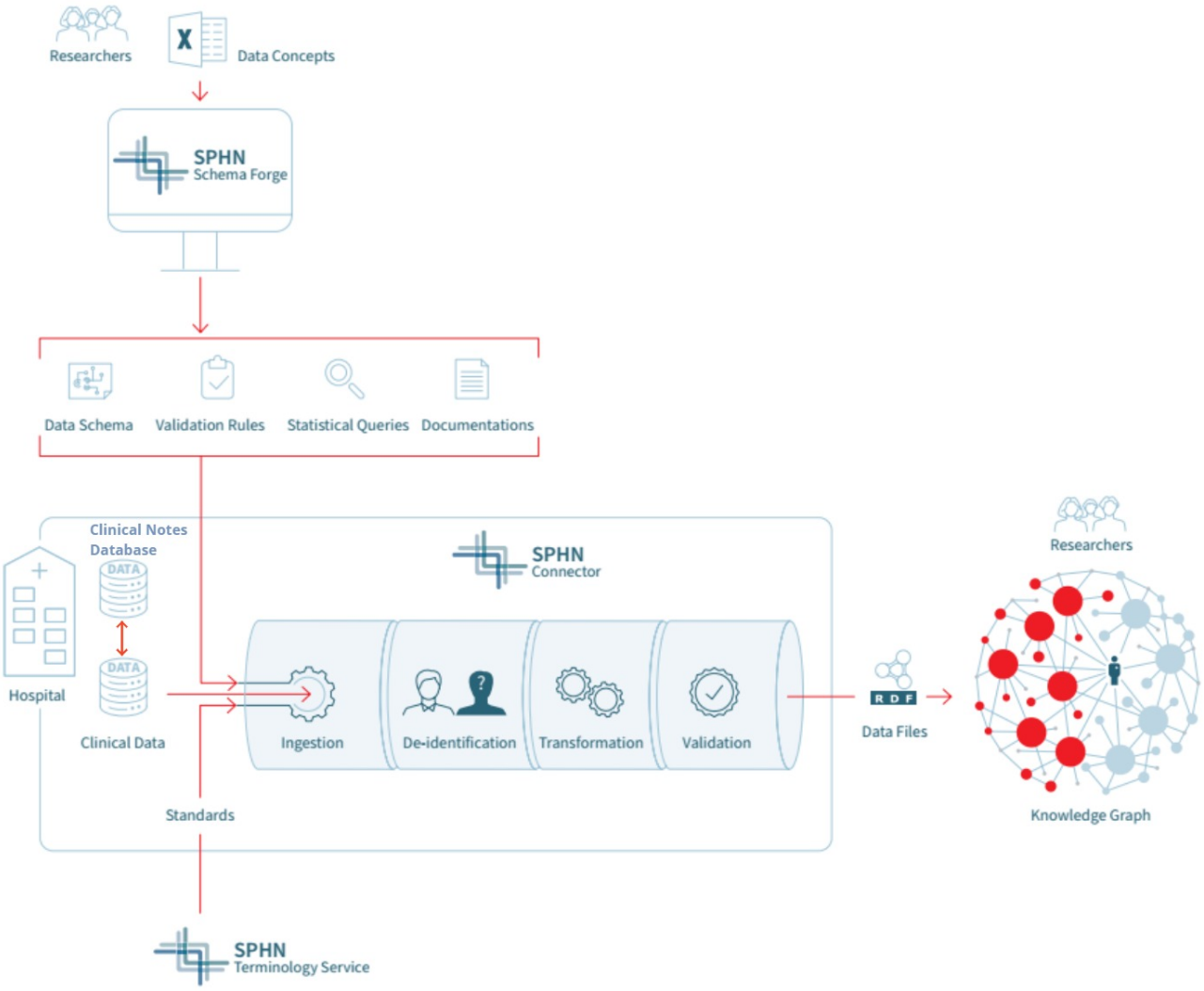


Figure 1: The proposed end-goal integration architecture showcasing the flow from our Clinical Notes Database at the hospital, through the SPHN Connector, to the generation of RDF data files for a Knowledge Graph accessible by researchers. Adapted from ("SPHN Fact Sheet", 2023).

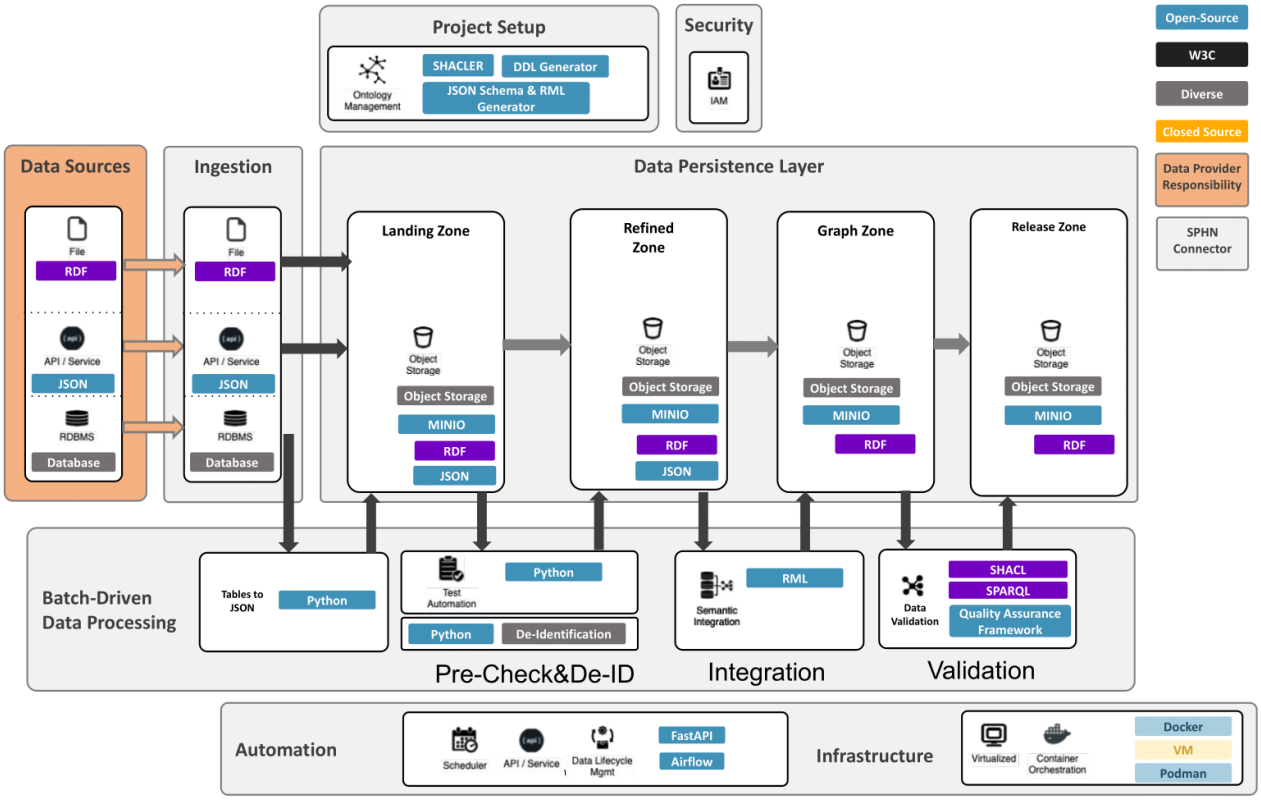


Figure 2: Detailed architecture of the SPHN Connector, illustrating the data flow from various data sources (SPHN Semantic Framework, 2023).

3.4 Database Technology

• What purpose does the database serve?

The database serves to store, organize, and manage clinical notes and patient data efficiently. It utilizes powerful NLP transformers to obtain key information from clinical notes and facilitates integration with and analysis of medical information for improved patient care and clinical decision-making.

• What database software do they use, for what reasons?

MySQL is used for its robustness, scalability, and strong community support. It's well-suited for handling structured data in SQL format, which is common in healthcare applications.

• What are the elements, and data flows of the database system?

The system includes data input interfaces, storage mechanisms, processing modules for queries, and output interfaces for reporting. Data flow involves inputting clinical data, processing it for storage, and retrieving it for analysis and reporting.

• Which SQL or NoSQL database software is used and for what reasons?

SQL database (MySQL) is chosen for its strong consistency, structured query language, and established reliability in handling structured data, crucial for clinical databases.

• How is the database system implemented and integrated?

The database is implemented as part of a larger healthcare information system and integrated with existing EHR systems for seamless data exchange and consistency.

• How is the data entered or migrated into the database?

Data is entered through automated data extraction tools and manual entry interfaces. Migration from existing systems is done through ETL (Extract, Transform, Load) processes.

• How is the data queried, manipulated, and transformed?

SQL queries are used for data retrieval. Data manipulation and transformation are handled through built-in functions and custom scripts for specific analytical purposes.

• How can the database system and queries be optimized in terms of volume and speed?

Optimization of Database System and Queries: Optimization is achieved by indexing critical fields, optimizing query structures, and using caching mechanisms. Regular monitoring and tuning are conducted to manage volume and enhance speed.

4 Model and Schema

4.1 ER Model

Our following ER model reflects a structure where patient information is central, with various data tables like clinical notes, medical conditions, and appointments linked to individual patients. Additionally, clinical notes are connected to analyses, both in raw and summarized forms.

4.1.1 Entities

- **Patients**: Identified by `patient_id`.
- **Clinical_Notes**: Identified by `pn_num`, related to **Patients**.
- **Clinical_Analysis_Table**: Contains `analysis_id`, related to **Clinical_Notes**.
- **Medical_Conditions_Table**: Linked to **Patients**.
- **Medications_Table**: Linked to **Patients**.
- **Appointments_Table**: Linked to **Patients**.
- **Notes_Summarization_Table**: Linked to **Clinical_Notes**.

Below are the SQL scripts that were crafted for the creation of tables.

```sql
CREATE TABLE IF NOT EXISTS patients (
    patient_id INT,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    age INT,
    gender VARCHAR(10),
    contact_information  VARCHAR(255),
    emergency_contact VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS clinical_notes (
    pn_num INT AUTO_INCREMENT PRIMARY KEY,
    patient_id INT,
    case_num INT,
    pn_history TEXT
);

CREATE TABLE IF NOT EXISTS clinical_analysis_table (
    analysis_id INT AUTO_INCREMENT PRIMARY KEY,
    note_id INT,
    entity VARCHAR(255),
    text VARCHAR(255),
    date_analysis datetime
);

CREATE TABLE IF NOT EXISTS medical_conditions_table (
    condition_id INT AUTO_INCREMENT PRIMARY KEY,
    patient_id INT,
    condition_name VARCHAR(255),
    status bool
);

CREATE TABLE IF NOT EXISTS medications_table (
    medication_id INT AUTO_INCREMENT PRIMARY KEY,
    patient_id INT,
    medication_name VARCHAR(255),
    dosage VARCHAR(10),
    purpose text
);

CREATE TABLE IF NOT EXISTS appointments_table (
    appointment_id INT AUTO_INCREMENT PRIMARY KEY,
    patient_id INT,
    date_time datetime,
    purpose VARCHAR(255),
    notes TEXT
);
```

```
CREATE TABLE IF NOT EXISTS notes_summarization_table (
    summary_id INT AUTO_INCREMENT PRIMARY KEY,
    note_id INT,
    summary TEXT,
    chr_text INT,
    chr_sum INT
```

4.1.2 Relationships

- **Clinical_Notes** has a foreign key to **Patients** (`patient_id`).
- **Clinical_Analysis_Table** has a foreign key to **Clinical_Notes** (`note_id`).
- **Medical_Conditions_Table**, **Medications_Table**, and **Appointments_Table** each have a foreign key to **Patients** (`patient_id`).
- **Notes_Summarization_Table** is linked to **Clinical_Notes** (`note_id`).

Below are the SQL scripts that were crafted for the creation of relationships.

```
ALTER TABLE clinical_notes
ADD FOREIGN KEY (patient_id) REFERENCES patients(patient_id);

ALTER TABLE clinical_analysis_table
ADD FOREIGN KEY (note_id) REFERENCES clinical_notes(pn_num);

ALTER TABLE medical_conditions_table
ADD FOREIGN KEY (patient_id) REFERENCES patients(patient_id);

ALTER TABLE medications_table
ADD FOREIGN KEY (patient_id) REFERENCES patients(patient_id);

ALTER TABLE appointments_table
ADD FOREIGN KEY (patient_id) REFERENCES patients(patient_id);

ALTER TABLE analysis_results_table
ADD FOREIGN KEY (analysis_id) REFERENCES
    clinical_analysis_table(analysis_id);

ALTER TABLE notes_summarization_table
ADD FOREIGN KEY (note_id) REFERENCES clinical_notes(pn_num);
```

4.2 DDL Schema

The next DDL schema outlines the structure of our database, ensuring that it is in line with the specific needs of clinical data processing and storage.

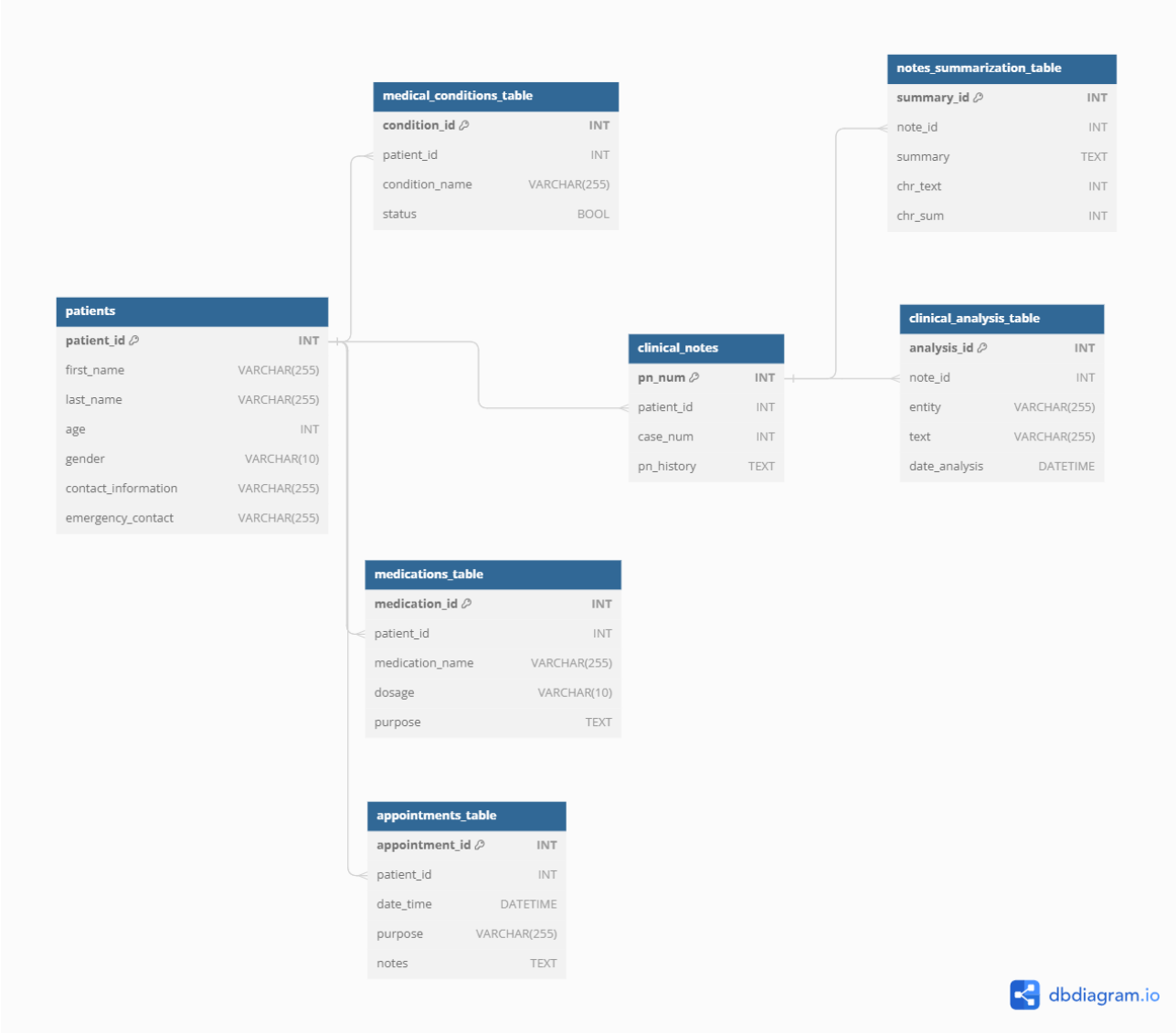


Figure 3: Clinical Notes Database schema diagram illustrating the structure and relationships of the tables within the clinical database, including ‘patients‘, ‘clinical notes‘, and associated data tables.

4.3 Normalization

The database design adheres to several normalization principles to ensure efficiency and data integrity. Each table is structured so that each field contains only atomic values, that is, each field contains the smallest unit of data, ensuring that there is no unnecessary duplication and promoting data consistency. In addition, each table is defined with a primary key, a unique identifier for each record, which is essential for quickly and accurately accessing specific rows within a table.

In addition, the use of foreign keys is an important feature of our database schema. These foreign keys create relationships between tables, enabling efficient organization and retrieval of related data across tables. This relational structure is particularly important in clinical settings, where data are often interconnected and need to be cross-referenced, for example, to link

patient records to their respective medical conditions or treatments.

5 Loading and Transforming Data

• What data sources do you have?

The data sources are comma-separated value (CSV) files containing patient notes, patient demographics, and medication data.

• What do the raw files look like?

The raw files are CSV formatted, with different fields for each table such as pn num, case num, and pn history for patient notes; patient id, last name, age, gender, contact information, and emergency contact for patients; and patient id, medication name, dosage, and purpose for medications.

• What target database structure do you need for your analysis?

The target database structure includes tables for patient notes, patients, and medications, with fields corresponding to the columns in the CSV files.

• Do you choose ETL or ELT? For what reasons?

Our chosen approach is ELT (Extract, Load, Transform). In which the raw data is loaded directly into the database and then transformations are applied afterward, as seen with the update statement on the clinical notes table.

The reason for choosing ELT was due to the utilization of secure NLP transformers and additionally utilizing MySQL database's own transformation capabilities, allowing for direct manipulation of data after it is loaded.

5.1 System Architecture

Our database system is structured to ensure a continuous and secure flow of data between the server and the client, facilitated by a network-based virtual machine (VM) configuration. This architecture is designed to efficiently manage the storage and processing of clinical notes.

The database server is hosted online, on a virtual machine. This server is the central repository where the MySQL database, named "clinical notes", resides.

Clients, such as researchers or health care providers, access the database remotely. In this configuration, the client can be a laptop or workstation from which scripts or queries are executed. The client interacts with the database server via the Internet.

The connection between the client and the server is established through the standard MySQL communication protocol, as the system adopts Python's mysql.connector library.

The credentials to access the database are the following:

```
host = 86.119.40.9
user = admin
password = Xvn239vn$mACo92!
database = clinical_notes
```

5.2 Data Loading

• How do you load the data into the database?

Data is loaded into the database using the LOAD DATA LOCAL INFILE command in SQL. This command takes a specified CSV file from a local directory and loads the data into the corresponding tables in the database. Fields are specified to be terminated by commas, and lines are terminated by newline characters. The first line, which typically contains column headers, is ignored during the load process. After loading, additional SQL commands are executed to update certain fields in the database, aligning with the required data structure for analysis.

Below are the SQL scripts that were crafted for the loading.

```
LOAD DATA LOCAL INFILE
    'C:\\Users\\andreavalle\\Desktop\\clinicalnotes\\datasets\\patient_notes.csv'
INTO TABLE patient_notes
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(pn_num, case_num, pn_history);

LOAD DATA LOCAL INFILE
    'C:\\Users\\andreavalle\\Desktop\\clinical-notes\\datasets\\patients.csv'
INTO TABLE patients
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(patient_id,last_name,age,gender,contact_information,emergency_contact);

LOAD DATA LOCAL INFILE
    'C:\\Users\\andreavalle\\Desktop\\clinical-notes\\datasets\\medications.csv'
INTO TABLE medications_table
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(patient_id,medication_name,dosage,purpose);

SET SQL_SAFE_UPDATES = 0;
UPDATE clinical_notes
SET case_num = patient_id;
SET SQL_SAFE_UPDATES = 1;
```

• How do you transform the data into the target structure?
The transformation of data into the target structure is performed through a series of scripts that leverage NLP models for named entity recognition and summarization.

```python
import torch
from transformers import pipeline
from database  import *

nlp_ner = pipeline("ner", model="samrawal/bert-base-uncased_clinical-ner")

query = "SELECT * FROM clinical_notes"
cursor.execute(query)
rows = cursor.fetchall()

for pn_num, patient_id, case_num, clinical_text in rows:
    ner_results = nlp_ner(clinical_text)

    current_entity = None

    current_entities = []
    current_entity_words = []
    for entity in ner_results:

        if entity['entity'].startswith('B-'):


            if current_entity:
                current_entities.append({'entity': current_entity, 'text': ' '.join(current_entity_words)})
            current_entity_words = []
            # Start a new entity
            current_entity = entity['entity'][2:]  # Remove the 'B-'
            current_entity_words = [entity['word']]









        elif entity['entity'].startswith('I-'):
            # If there's a continuing entity, add it to the current entity's words
            if current_entity and current_entity == entity['entity'][2:]:
                current_entity_words.append(entity['word'])

        # If there was an ongoing entity, process and store it


    current_entities_clean=[]
    for entity_data in current_entities:
        entity_data['text'] = entity_data['text'].replace(' ##', '')
        current_entities_clean.append(entity_data)
    # Store the last entity (if any)

    # Insert entities into the clinical_analysis_table
    for entity_data in current_entities_clean:
        query_insert = "INSERT INTO clinical_analysis_table (note_id, entity, text, date_analysis) VALUES (%s, %s, %s, NOW())"
        cursor.execute(query_insert, (pn_num, str(entity_data['entity']), str(entity_data['text'])))
        conn.commit()
```

Named entity recognition: The script `nlp_processing.py` uses an NLP pipeline to process clinical text from the clinical notes table, extract entities, and then store the results in the clinical analysis table. It identifies entities marked as beginning (B-) and inside (I-) an entity and groups them accordingly.

```python
from transformers import pipeline
from database  import *

# SQL query to find problems and their corresponding patient_id
query = """
SELECT cn.patient_id, cat.text
FROM clinical_analysis_table cat
JOIN clinical_notes cn ON cat.note_id = cn.pn_num
WHERE cat.entity = 'problem'
"""

# Execute the query
cursor.execute(query)
rows = cursor.fetchall()

# Insert data into medical_conditions_table
insert_query = """
INSERT INTO medical_conditions_table (patient_id, condition_name, status)
VALUES (%s, %s, %s)
"""

# Iterate over the results and insert each into the medical_conditions_table
for patient_id, condition_text in rows:
    status = True
    cursor.execute(insert_query, (patient_id, condition_text, status))

# Commit the changes to the database
conn.commit()

# Close the cursor and connection
cursor.close()
conn.close()
```

Condition Extraction: The script `process_ner.py` extracts specific entities categorized as 'problem' from the clinical analysis table, linking them to patient IDs, and inserts this data into the medical conditions table.

```python
import pandas as pd
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
from tqdm import tqdm
from database import *

# Initialize tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("Falconsai/medical_summarization")
model = AutoModelForSeq2SeqLM.from_pretrained("Falconsai/medical_summarization")

# Fetch data from clinical_notes
query = "SELECT * FROM clinical_notes"
cursor.execute(query)
rows = cursor.fetchall()

# Function to apply summarizer model to individual texts
def apply_summarizer(text):
    input_ids = tokenizer.encode(text, truncation=True, padding=True, return_tensors="pt")
    summaries = model.generate(input_ids, max_length=230, min_length=30, do_sample=False)
    return tokenizer.decode(summaries[0], skip_special_tokens=True)

# Process each row
for pn_num, patient_id, case_num, clinical_text in rows:
    summary = apply_summarizer(clinical_text)

    # Insert the summary without character counts
    query_insert = "INSERT INTO notes_summarization_table (note_id, summary) VALUES (%s, %s)"
    cursor.execute(query_insert, (pn_num, summary))
    conn.commit()

    # Update the record with character counts
    query_update = """
    UPDATE notes_summarization_table
    SET chr_text = LENGTH(%s),
        chr_sum = LENGTH(%s)
    WHERE note_id = %s
    """
    cursor.execute(query_update, (clinical_text, summary, pn_num))
    conn.commit()
```

Summarization: In `summarizer.py`, a summarization model is applied to the clinical text from the clinical notes table. The resulting summaries are then stored in the notes summarization table, along with character counts for both the original text and the summary.

These transformations prepare the raw clinical text data for analysis by converting it into structured data within the database, thus fitting the target structure required for further processing or insights extraction.

5.3 Data Transformation

To maintain the quality of the data stored in the clinical analysis table, it's crucial to remove entries that do not contain meaningful information. One common scenario is the existence of rows with empty text fields, which can occur due to various reasons like data entry errors or incomplete data extraction.

```
DELETE FROM clinical_analysis_table WHERE text = '';
```

In the notes summarization table, it's beneficial to track the length of both the original clinical notes and their summarized versions for further analysis, such as understanding the efficiency of the summarization process.

```
UPDATE notes_summarization_table
SET chr_text = LENGTH(%s),
    chr_sum = LENGTH(%s)
WHERE note_id = %s
```

6 Queries and Analyses

6.1 Database Queries

In this section of the report, we present a series of SQL queries made to extract vital information from our clinical notes database. These queries are designed to delve into various aspects of patient data, from the most recent clinical notes to detailed analyses of medical entities such as examinations, problems, and treatments. Each query has a specific purpose, providing essential insights for effective patient care and medical research.

```
SELECT
    p.patient_id,
    CONCAT(p.first_name, ' ', p.last_name) AS patient_name,
    cn.pn_num AS latest_note_id,
    cn.pn_history AS latest_note_content
FROM
    patients p
LEFT JOIN
    clinical_notes cn ON p.patient_id = cn.patient_id
WHERE
    cn.pn_num = (SELECT MAX(pn_num) FROM clinical_notes
    WHERE patient_id = p.patient_id);
```

	patient_id	patient_name	latest_note_id	latest_note_content
▶	0	Corissa Romero	2451	CC: heart pounding HPI Mr. Dillon Cleveland is a...
	1	Clarisse Clohisey	10991	20 YO F COMPLAINS OF ABDOMINAL PAIN FOR...
	2	Dru Inmett	22141	Ms Montgomery is a 44yo G2P2 who presents w...
	3	Louise Delhay	39935	35 yo man with a 2 month history of epigastric ...
	4	Trace Conahy	45588	Karin Moore is a 45yo female with no PMH who ...
	5	Odo Codi	57092	26yo F presents for follow-up from ER visit for ...
	6	Lavinia Place	61780	HPI: Mr. Smith is a 17 yo man p/w c/o CP x1day...
	7	Madella Cariss	74283	CC: irregular menses HPI: 35F G0 with irregular ...
	8	Francene Biesinger	84379	67 yo f c/o sleeping problem since 3 weeks ago,...
	9	Suzanne Dumberell	95334	patient is a 20 yo F who presents with a headac...

This query retrieves the most recent clinical note for each patient. It's essential for understanding the current medical status or the latest observations made by healthcare providers.

```
SELECT
    cn.patient_id,
    SUM(CASE WHEN cat.entity = 'test' THEN 1 ELSE 0 END) AS test_count,
    SUM(CASE WHEN cat.entity = 'problem' THEN 1 ELSE 0 END) AS
        problem_count,
    SUM(CASE WHEN cat.entity = 'treatment' THEN 1 ELSE 0 END) AS
        treatment_count
FROM
    clinical_analysis_table cat
JOIN
    clinical_notes cn ON cat.note_id = cn.pn_num
GROUP BY
    cn.patient_id;
```

	patient_id	test_count	problem_count	treatment_count
▶	0	50	1089	141

This query focuses on quantifying the occurrence of key medical entities - tests, problems, and treatments - for each patient. It provides a numerical perspective on the frequency of these entities in clinical documentation.

```
SELECT
    p.patient_id,
    CONCAT(p.first_name, ' ', p.last_name) AS patient_name,
    GROUP_CONCAT(DISTINCT CASE WHEN ca.entity = 'problem'
    THEN ca.text END SEPARATOR ', ') AS problems,
```

```
    GROUP_CONCAT(DISTINCT CASE WHEN ca.entity = 'treatment'
    THEN ca.text END SEPARATOR ', ') AS treatments
FROM
    patients p
JOIN
    clinical_analysis_table ca ON p.patient_id = ca.note_id
GROUP BY
    p.patient_id;
```

	patient_id	patient_name	problems
▶	1	Clarisse Clohisey	chest pain, chills, dispnea, dyaphoresis, fever, heart pounding, mi, nausea, nk, non - ...

This query compiles a list of all unique problems and treatments associated with each patient. It offers a detailed view of the patient's medical history and the care they have received, encapsulating crucial aspects of their health records.

7 Performance Optimization

7.1 Optimization Strategies

The clinical notes database, with its rich and complex datasets, requires efficient querying capabilities to ensure that healthcare providers can access and analyze patient information promptly and accurately. A key aspect of this optimization involves the strategic use of indexes.

To improve the performance of our clinical notes database, we implemented an index on the `patient_id` column of the patients table, as shown in the following SQL command:

```
CREATE INDEX idx_patient_id ON patients(patient_id);
```

In the field of health data management, the ability to quickly and accurately analyze patient data is critical. To facilitate this task, we designed a sophisticated SQL query encapsulated in a view called patient condition analysis. This view is designed to provide a comprehensive overview of patients' medical conditions along with their clinical notes. The purpose of this query is to simplify the process of retrieving and analyzing critical patient information, which is critical for healthcare providers in making informed decisions.

```
CREATE VIEW patient_condition_analysis AS
SELECT
    p.patient_id,
    CONCAT(p.first_name, ' ', p.last_name) AS full_name,
    p.age,
    p.gender,
    COUNT(mc.condition_id) AS number_of_conditions,
    GROUP_CONCAT(DISTINCT mc.condition_name SEPARATOR ', ')
    AS list_of_conditions,
    MAX(cn.pn_num) AS latest_note_id,
```

```
    MAX(cn.pn_history) AS latest_note_content
FROM
    patients p
LEFT JOIN
    medical_conditions_table mc ON p.patient_id = mc.patient_id
LEFT JOIN
    clinical_notes cn ON p.patient_id = cn.patient_id
GROUP BY
    p.patient_id, p.first_name, p.last_name, p.age, p.gender;
```

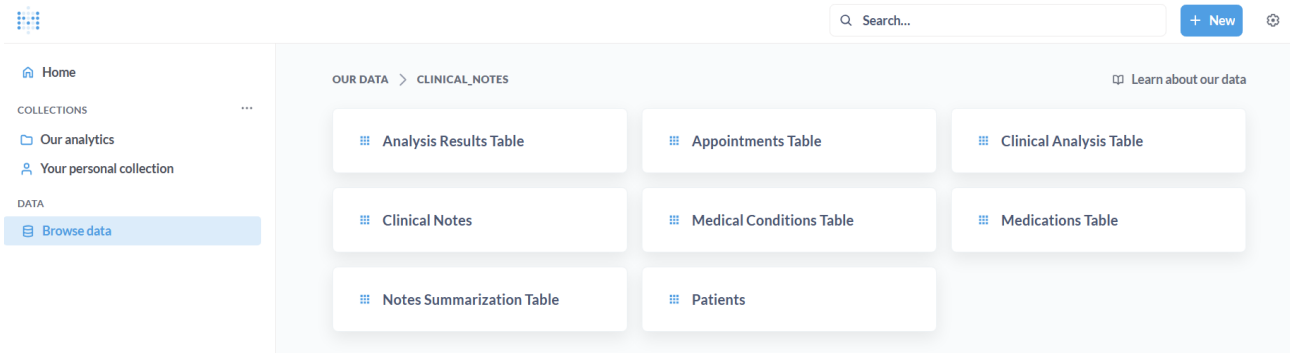
Another view, patient medication appointment analysis, is a valuable tool for healthcare providers and pharmacists. It allows them to quickly assess the frequency of patient visits, medications prescribed, and whether there are patterns or trends in medication prescriptions in relation to patient demographics and frequency of visits. This can help assess treatment effectiveness, understand patient adherence to appointments, and monitor medication management. Thus, vision plays a crucial role in ensuring effective and informed health care.

```
CREATE VIEW patient_medication_appointment_analysis AS
SELECT
    p.patient_id,
    CONCAT(p.first_name, ' ', p.last_name) AS full_name,
    p.age,
    p.gender,
    COUNT(DISTINCT a.appointment_id) AS total_appointments,
    MAX(a.date_time) AS last_appointment_date,
    COUNT(DISTINCT m.medication_id) AS total_medications,
    GROUP_CONCAT(DISTINCT m.medication_name SEPARATOR ', ')
    AS list_of_medications
FROM
    patients p
LEFT JOIN
    appointments_table a ON p.patient_id = a.patient_id
LEFT JOIN
    medications_table m ON p.patient_id = m.patient_id
GROUP BY
    p.patient_id, p.first_name, p.last_name, p.age, p.gender;
```

8 Visualization and Interpretation

8.1 BI Tool

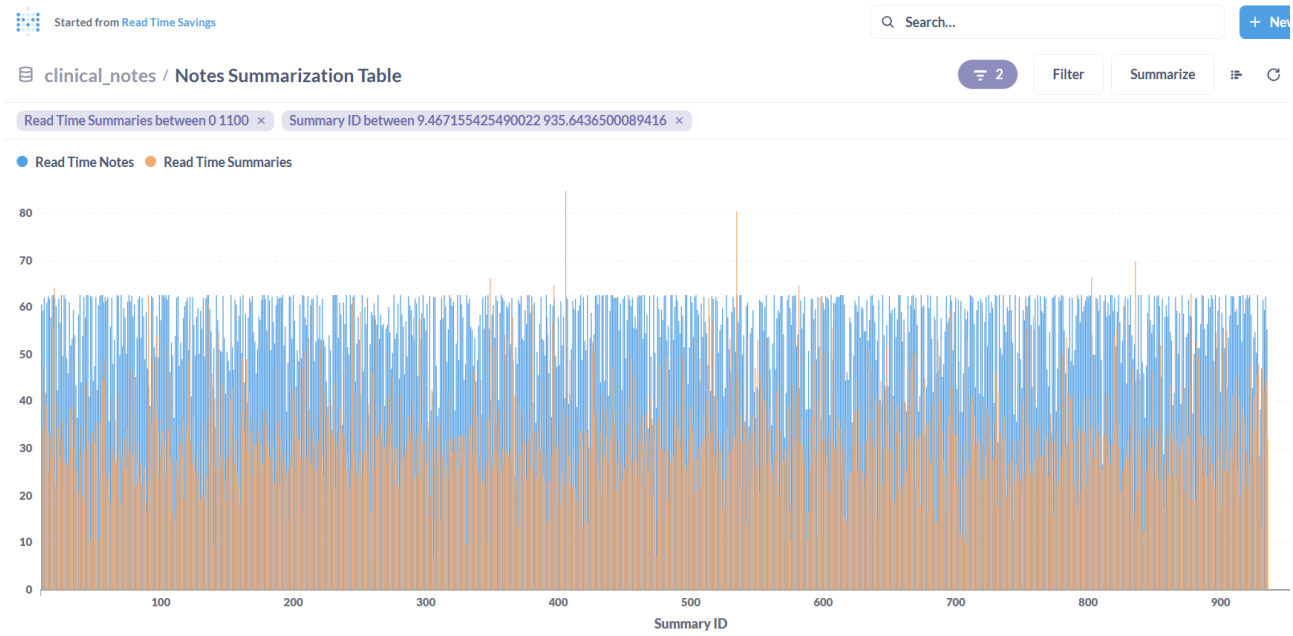
Utilizing Metabase made the integration with MySQL very intuitive as many of the expressions one can use to perform further analysis on data from specific tables or views one must build queries. The interface is very user-friendly and we were able to master the BI tool well enough to then train or give clear explanations to our stakeholders.



This screen shot above is a view of our database homepage.

8.2 Parameterized Visualizations

We created a parameterized visualization with the metabase tool with data from our clinical note database to visualize the differences between the amount of time per clinical note a clinician would take to read.



8.3 Concrete Decisions

In our practical example, we conducted an analysis to convert the character counts of the complete clinical notes and their summaries into word counts. This analysis then helped estimate the time saved in reviewing or reading these notes by translating word counts into time savings. First, convert the average number of characters in each column to the average number of words. Assuming an average word length (including spaces) is about 5 characters, you can divide the number of characters by 5.

For example, if the average characters in clinical notes are C and in summaries are S, then:

$$\text{Average words in clinical notes} = \frac{C}{5}$$
$$\text{Average words in summaries} = \frac{S}{5}$$

Calculate Reading Time: Next, calculate the reading time using the average time to read a word (0.33 seconds). Multiply the average number of words by 0.33 to get the reading time in seconds.

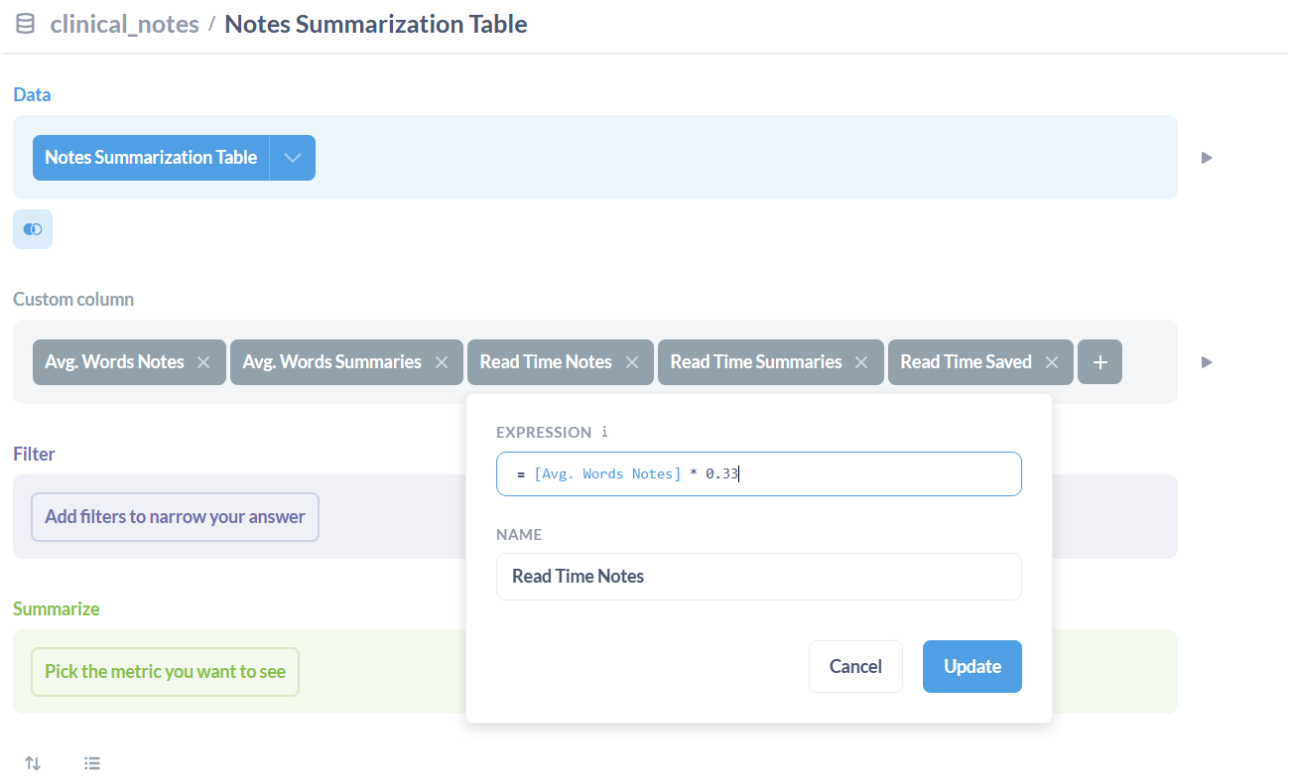
$$\text{Reading time for clinical notes} = \frac{C}{5} \times 0.33$$
$$\text{Reading time for summaries} = \frac{S}{5} \times 0.33$$

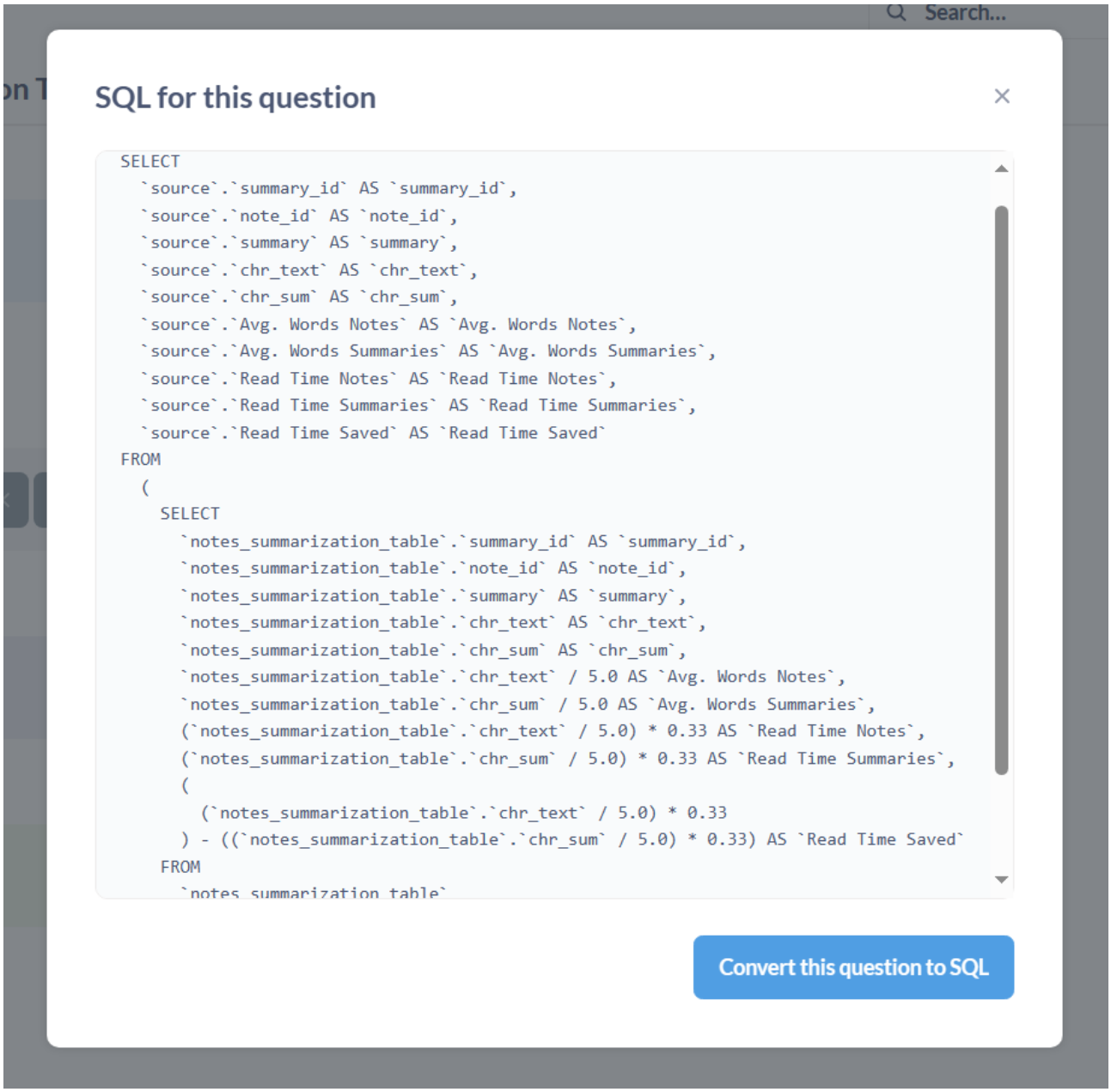
Find the Difference: Finally, subtract the reading time of the summaries from the reading time of the clinical notes to find the time saved.

$$\text{Time difference} = \left(\frac{C}{5} \times 0.33\right) - \left(\frac{S}{5} \times 0.33\right)$$

8.4 Personas

Our stakeholders, equipped with diverse backgrounds and varying levels of technical expertise, can effectively perform following necessary calculations in Metabase. This tool's intuitive interface simplifies complex data analysis, enabling stakeholders to seamlessly estimate time savings from reading clinical note summaries. They can use Metabase's query builder to calculate average word counts and reading times, leveraging the data visualizations to derive actionable insights. This approach ensures that even those without deep technical knowledge can make informed decisions based on the analyzed clinical data.





9 Lessons Learned

This project, while challenging, provided numerous valuable insights and lessons. Firstly, the importance of data structure and organization was underscored, emphasizing that efficient data management is crucial for effective analysis. We learned the intricacies of integrating NLP models into database systems, which required a deep understanding of both the linguistic aspects and the technical infrastructure. Handling datasets efficiently, especially keeping healthcare restrictions, taught us the significance of data privacy and ethical considerations. The project also highlighted the power of visualization tools in interpreting complex data, making it accessible to non-technical stakeholders. Finally, this journey reinforced the idea that continuous learning and adaptation are key in the ever-evolving field of data science and healthcare technology.

10 Conclusion

In concluding the project and documentation of "DBM - Clinical Notes," it is evident that the endeavor was a comprehensive exploration into the field of database management and clinical data analysis. The project successfully achieved its goal of developing a robust and efficient database system utilizing MySQL for managing clinical notes. The integration of advanced NLP techniques for entity recognition and summarization, coupled with sophisticated data transformation and loading strategies, has significantly enhanced the accessibility and utility of clinical data. The use of Metabase as a BI tool for visualizing and interpreting the data further underscores the project's commitment to making complex data comprehensible and actionable. The successful implementation of this project not only showcases our team's technical prowess but also sets a precedent for future endeavors in healthcare data management and analysis. The detailed documentation and the public repository serve as valuable resources for ongoing and future research in this vital field.

11 Appendix

[] The complete code to our project can be found on our public repository on github.
https://github.com/valleandreaa/Clinical-NLP-Feature-Extraction