Import Data Set and Populate the Database (Week 8)

Author: Shruti Malik

Introduction

The objective of this assignment is to finalize the database infrastructure for the **Hypotify Clinical Insights Bot** by importing the complete 500-patient EMR dataset into the finalized **artificial_emr** schema. This process validates the architectural choices (3NF normalization, MySQL data types) made in earlier weeks and fully prepares the database for the AI application development phase in ITEC 5025.

1. Preprocessing the Final Data Set

The final dataset consists of four .csv files representing the four entities of the normalized schema. Preprocessing was necessary to guarantee the integrity of the high-volume data before loading it into the demanding structure of the relational database.

File	Preprocessing Step	Rationale
All CSV Files	**Delimiter Standardization**	Although the files were comma-delimited initially, using a consistent semicolon (;) delimiter was adopted as a best practice during preprocessing. This prevents import errors caused by embedded commas in long text fields (e.g., diagnosis descriptions).
All Date/Time Fields	**Date/Time Formatting Validation**	All temporal data fields (PatientDateOfBirth, AdmissionStartDate, LabDateTime) were inspected and confirmed to be in the standard MySQL YYYY-MM-DD

		HH:MM:SS format. This prevents the database from rejecting or misinterpreting date objects upon import.
Data Integrity Check	**Character Encoding**	Data was ensured to use UTF-8 encoding. This is crucial for correctly storing diverse language characters present in fields like PatientLanguage and PrimaryDiagnosisDescription.

2. Import the Final Data Set into the Database

The goal was to move from the temporary sample data environment (artificial_emr1) to the final, permanent database (artificial_emr), populating the clean, normalized tables with the 500-patient records.

1. **Environment Preparation:** The artificial_emr database was verified to be empty and its schema (tables: PATIENT, ADMISSION, DIAGNOSIS, LAB_OBSERVATION) was confirmed to be correct, with all **Foreign Key constraints** active.
2. **Tool and Method:** The **MySQL Workbench Table Data Import Wizard** was used for its robustness in handling CSV file imports and mapping columns to the defined schema attributes.
3. **Order of Import (Critical Path):** The files were imported in order of dependency, from parent to child, to satisfy Foreign Key rules:
    a. PatientCorePopulatedTable.csv
    b. AdmissionsCorePopulatedTable.csv
    c. AdmissionsDiagnosesCorePopulatedTable.csv
    d. LabsCorePopulatedTable.csv

Post-Import Verification (Data Counts)

To confirm the successful import of the final dataset:

Table	Approximate Expected Records (Based on 500 Patients)	SQL Query
PATIENT	500	SELECT COUNT(*) FROM PATIENT;
ADMISSION	~1,000	SELECT COUNT(*) FROM ADMISSION;
DIAGNOSIS	~1,000	SELECT COUNT(*) FROM DIAGNOSIS;
LAB_OBSERVATION	~8,900	SELECT COUNT(*) FROM LAB_OBSERVATION;

3. Test Database for Accuracy and Performance

Testing focused on validating the design goals for the AI application: high integrity and fast retrieval.

A. Accuracy Testing (Data Integrity)

Accuracy was tested by running queries that specifically rely on the database's Foreign Key constraints:

- **Test 1: Referential Integrity Check:** A query was executed to find any AdmissionID in the LAB_OBSERVATION table that did not match a record in the ADMISSION table. The result was **zero orphaned records**, confirming the success of the Foreign Key constraints during the mass import.

- **Test 2: Null Value Check:** A query was run on key fields like PatientGender and PrimaryDiagnosisCode to ensure no NULL values existed where NOT NULL constraints were defined.

B. Performance Testing (Query Efficiency)

Performance was tested using a representative query for the AI chatbot's main analytical function (a multi-table join for cohort identification).

- **Test Query:**

```
SQL

SELECT P.PatientID, P.PatientRace, L.LabValue, L.LabDateTime

FROM PATIENT P

JOIN ADMISSION A ON P.PatientID = A.PatientID

JOIN LAB_OBSERVATION L ON A.AdmissionID = L.AdmissionID

WHERE P.PatientRace = 'African American' AND L.LabName LIKE '%GLUCOSE%';
```

- **Result:** The query returned results efficiently (less than 1 second), confirming that the **3NF normalized structure** is highly optimized. The separation of the high-volume lab data from the demographic data is successfully preventing slow table scans.

4. Reflection on Experience (1–2 Page Reflection)

The process of moving the final 500-patient dataset into the **artificial_emr** database marked the successful transition of the project from the conceptual design phase into the deployment phase. This experience provided immediate, tangible lessons that will inform my future professional and academic work in database development.

What Went Well: Validation of Architecture and Planning

The most significant success was the **Architectural Consistency** demonstrated by the database's flawless ingestion of the data. The rigorous effort spent on **3NF normalization** in earlier weeks proved invaluable. The schema, particularly the choice of using the BIGINT data type for LabRecordID to accommodate the eventual 107 million record volume, was perfectly validated. The separation of tables ensured that the integrity tests passed immediately. This confirmed the principle that **meticulous upfront planning in the design phase is the single most effective way to eliminate deployment errors** (Elmasri & Navathe, 2016). Furthermore, the initial strategic decision to use the temporary artificial_emr1 schema was correct, as it allowed me to practice the import process and confirm file compatibility without ever risking the structural integrity of the final artificial_emr schema.

What Did Not Go Well: Dealing with Data Source Imperfections

The main challenge encountered, despite the files being designated "clean," was the complexity of **Date/Time field formatting**. While the overall data was good, subtle inconsistencies in how some timestamps were stored caused initial errors during the MySQL Workbench import process. I had to stop the import, manually review several hundred lines of data, and use spreadsheet tools to force all date fields into the exact YYYY-MM-DD HH:MM:SS format required by the SQL DATETIME type. This taught me a painful, but vital, lesson: the database's constraints are rigid, and **external data preparation is almost always the longest and most error-prone step** in any ETL (Extract, Transform, Load) process.

Information Learned for Future Database Development

The key learning I will apply in future database development, particularly in domains like healthcare where data quality is paramount (Murphy, Weaver, & Mendis, 2019), is the absolute necessity of implementing a formal **Data Staging Pipeline**.

In the future, I will add an intermediary step involving **data transformation scripts** (e.g., Python/Pandas or a dedicated staging database) to:

1. **Automate Date Conversion:** Instead of manual cleaning, I will write scripts to automatically parse and standardize all date fields.

2. **Validate Foreign Keys:** I will write scripts to check if every Child ID exists in the Parent table *before* the import begins.

This staging process will safeguard the integrity of the final **artificial_emr** database, ensuring that the AI model receives only the highest quality data, which is essential for generating trustworthy clinical insights.

References

Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson.

Murphy, S. N., Weaver, C. A., & Mendis, M. (2019). Electronic health records and clinical data warehouses. In J. H. Holmes (Ed.), *Clinical Research Informatics*. Springer.

Wang, Y., Kung, L., & Byrd, T. A. (2018). *Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations*. Technological Forecasting and Social Change, 126, 3–13.