Moroccan National Health Services (MNHS)

Data Management Course

Mohammed VI Polytechnic University (UM6P)
Professor: Karima Echihabi
Teaching Assistant: Amine Benjelloun
Program: Computer Engineering
Session: Fall 2025

Team Information

Team Name:	Team1 / Section 2
Member 1:	Soukaina EL KESSIRI
Member 2:	Soukaina EL JAOUHARI
Member 3:	Alaa Allah EL BAZZAZ
Member 4:	Mohamed Ziad EL ABBASSI
Member 5:	Youness EL HACHIMI
Member 6:	Oussama EL HILALI
Repository Link:	`https://github.com/hash-gdel`

Prepared as part of the Data Management Project.

1- Introduction

This lab advances the MNHS database implementation, focusing on normalization validation and comprehensive SQL query development. Our objectives were to verify that all relations satisfy BCNF for an optimal design, implement the refined schema using DDL, populate it with realistic data via DML, and develop a suite of 20 analytical SQL queries to demonstrate the system's functionality

2- Requirements

- Normalization: Validate all relations against BCNF and verify lossless joins and dependency preservation.
- DDL: Create the MNHS database schema with tables, keys, and constraints using SQL.
- DML: Populate the database with sample data and perform update/delete operations.
- 20 Queries: Implement and execute the specified set of 20 SQL queries on the MNHS database.

3: Methodology

In this lab, we advanced our MNHS database implementation by focusing on normalization validation and comprehensive SQL query development. Our approach involved systematically verifying that all database relations satisfy BCNF to ensure optimal design, followed by implementing a complete set of analytical queries addressing various healthcare scenarios. We employed a structured methodology that combined theoretical database principles with practical SQL implementation, ensuring our solution meets both academic rigor and real-world healthcare data management requirements.

Normalization Validation

- **BCNF Verification:** Systematically validated each relation against BCNF to ensure optimal database design
- **Candidate Key Analysis:** Identified all candidate keys and functional dependencies for each entity
- **Super Key Validation:** Verified that every determinant is a super key

- **Schema Preservation:** No decomposition necessary as all relations were already in BCNF initially
- **Integrity Maintenance:** Preserved both lossless joins and functional dependencies throughout the schema

Data Implementation Strategy

- **Realistic Sample Data:** Created comprehensive test data with at least 5 representative rows per table
- **Medical Relevance:** Populated tables with medically relevant information
- **Relationship Integrity:** Ensured foreign key relationships were properly maintained across all tables
- **Edge Case Testing:** Included negative quantities and prices to test data quality validation

Query Implementation Methodology

We prioritized code clarity and simplicity in our query implementations, employing the following approaches:

- **Query 1:** We extracted last names from full names using string manipulation functions (SUBSTRING, CHARINDEX, REVERSE) to properly order patients alphabetically by last name, accommodating the specific name format in our dataset.
- **Query 2:** We used simple projection to retrieve distinct insurance types, providing a clear overview of available coverage options in the MNHS system.
- **Query 3:** We implemented nested subqueries with multiple JOIN conditions across Staff, Work_in, Department, and Hospital tables to accurately filter staff members working in Rabat-based hospitals.
- **Query 4:** We used date arithmetic with CURRENT_DATE and INTERVAL to dynamically identify appointments scheduled within the next seven days, joining Appointment and ClinicalActivity tables for comprehensive scheduling information.
- **Query 5:** We applied basic aggregation with GROUP BY across Department, ClinicalActivity, and Appointment tables to count appointments per department, providing departmental workload metrics.

- **Query 6:** We utilized the AVG function with GROUP BY on the Stock table to compute average medication unit prices per hospital, enabling pricing analysis across healthcare facilities.

- **Query 7:** We implemented multi-table JOINs across Hospital, Department, ClinicalActivity, and Emergency tables with HAVING clause filtering to identify hospitals with more than twenty emergency admissions, focusing on high-volume emergency care facilities.

- **Query 8:** We combined therapeutic class filtering with price constraints through JOIN operations between Medication and Stock tables to identify affordable antibiotics (unit price below 200), supporting medication affordability analysis.

- **Query 9:** We used correlated subqueries with COUNT operations to implement a top-N per category pattern, identifying the three most expensive medications for each hospital through strategic JOINs between Stock, Medication, and Hospital tables.

- **Query 10:** We employed conditional aggregation with CASE statements to pivot appointment status counts (Scheduled, Completed, Cancelled) per department in a single result set, providing comprehensive appointment analytics.

- **Query 11:** We utilized NOT IN with date range subqueries to identify patients without scheduled appointments in the next thirty days, joining Patient, ClinicalActivity, and Appointment tables for accurate patient engagement tracking.

- **Query 12:** We implemented advanced analytics using Common Table Expressions (CTEs) to calculate staff appointment percentages relative to hospital totals, with proper CASE statement handling for division by zero scenarios, providing staff performance metrics.

- **Query 13:** We applied business logic filtering through JOIN operations between Medication, Stock, and Hospital tables to identify drugs below reorder levels across healthcare facilities, supporting inventory management.

- **Query 14:** We used relational division approach with multiple CTEs to find hospitals that stock every antibiotic in the catalog, implementing comprehensive medication availability analysis through strategic JOINs and COUNT operations.

- **Query 15:** We implemented comparative analytics with correlated subqueries to flag hospitals with above-average pricing per therapeutic class, using CASE state-

ments and JOIN operations between Stock, Medication, and Hospital tables for pricing intelligence.

- **Query 16:** We used MIN aggregation with date filtering to identify the next scheduled appointment for each patient, joining Patient, Appointment, and ClinicalActivity tables for comprehensive patient scheduling overview.

- **Query 17:** We combined emergency visit counting with date filtering using subqueries and GROUP BY operations to identify patients with at least two emergency visits whose latest visit was within the last fourteen days, supporting patient follow-up protocols.

- **Query 18:** We applied ranking logic with time-based filtering to compare hospital performance by completed appointments per city over the last ninety days, using JOIN operations across Hospital, Department, ClinicalActivity, and Appointment tables for geographic performance analysis.

- **Query 19:** We used percentage-based HAVING conditions with MIN and MAX aggregations to identify medications with significant price variations (greater than 30% spread) across cities, joining Stock, Hospital, and Medication tables for pricing disparity analysis.

- **Query 20:** We implemented data quality validation through simple selection with WHERE conditions to flag invalid stock entries containing negative quantities or non-positive unit prices, ensuring data integrity in inventory management.

5- Discussion

5.0.1 Challenges Faced

- Constantly changing data via INSERT (to visualize the queries output -each query needed some specific data to be visualized), requiring group members to re-run queries for a similar output.

- SQL errors in the data that needed to be corrected and took some time (spaces in ENUM), PRIMARY KEY conflicts (double inserting some primary keys).

- New commands like string manipulation in query one (to sort by last name) and to set intervals to find appointment for example in a certain range and function differences (LEN LENGTH, DATEADD INTERVAL).

5.0.2 Lessons learned

- Forced the team to agree on a final, complete and correct script for all data inserts
- Learned how constraints protect data quality and improved understanding error messages
- Learned to distinguish between standard SQL and SQL dialects while looking up functions to complete complex queries

6- Conclusion

In this lab, we successfully advanced the MNHS database by validating its BCNF compliance, implementing the refined schema with DDL, and populating it with realistic data using DML. The development and execution of 20 comprehensive SQL queries demonstrated the system's ability to support complex healthcare data management scenarios, resulting in a fully functional database system.