University
Mohammed VI
Polytechnic

College of Computing

Deliverable #4: Normalization and SQL Implementation

Data Management Course

UM6P College of Computing

Professor: Karima Echihabi **Program:** Computer Engineering

Session: Fall 2025

Team Information

Team Name	Groupe2
Member 1	Abir Fakhreddine
Member 2	Malak El Assali
Member 3	Nada El Farissi
Member 4	Amine Chrif
Member 5	Anass Fertat
Member 6	Yasser Hallou
Repository Link	`https://github.com/beaNoBeebea`

---

1 Introduction

After having refined the conceptual schema for the MNHS database in the previous labs, our next step is to validate the schema looking at the deeper theoretical and practical level. In this lab, our mission is to ensure the relational model is well structured and actually usable

In the first part, we focus on normalization, more precisely on validating that each relation is in BCNF. We do this to eliminate redundancy and prevent any anomalies when modification occurs. If a relation is not in BCNF, we decompose it and verify that the decomposition is lossless and dependency preserving.

Once the schema is theoretically validated, we can implement it in SQL. We first use Data Definition Language (DDL) to translate the schema into actual database tables. We then use Data Manipulation Language (DML) to fill the database with sample data and perform updates and deletions that mimic real life scenarios in a healthcare setting. And finally, we move to the practical use of the MNHS database through a range of SQL queries. .

2 Requirements

In this deliverable, we must theoretically validate the relational schema and implement some queries that retrieve essential information in a real-world scenario.

Our first task is to normalize the schema, we must validate each relation against BCNF and verify losslessness and dependency preservation for eventual decompositions.

Next, we should implement the refined MNHS schema using DDL in SQL, including primary keys, foreign keys and participation constraints. We must also apply some schema alteration.

For the data manipulation part, we must insert at least five sample rows to each table, then perform a few updates on the tables such as modifying or deleting data.

Finally, our last task is to write and execute 20 SQL queries, covering:

- Selection, projection and ordering
- Joins across multiple entities
- Grouping and ranking
- Data quality checks
- Set-based logic (for example hospitals stocking every antibiotic)

3 Methodology

3.1 Normalization in Healthcare Databases

Normalization is a process to organize data in order to avoid redundancy and to improve overall performance. This is absolutely crucial for a well-structured database especially in healthcare systems like MNHS because information on patients, prescriptions or appointments must be accurate and up to date.

Without normalization, some data might be stored in different places multiple times. For example, if we modify a patient's address, updating in one location and not the other could lead to inconsistencies in the database. On the other hand, if the database is normalized, modifying

the data in one table will not pose any problem, we only need to change the data in the master table for the changes to be applied everywhere else.

4 Implementation & Results

4.1 BCNF Validation

4.1.1 Patient

Relation: Patient(IID, CIN, FirstName, LastName, Birth, Sex, BloodGroup, Phone)
Functional dependencies:

- IID → CIN, FirstName, LastName, Birth, Sex, BloodGroup, Phone
- CIN → IID, FirstName, LastName, Birth, Sex, BloodGroup, Phone

Is it a BCNF?
Yes, it's a BCNF because both 'IID' and 'CIN' are candidate keys and therefore superkeys.

4.1.2 Contact_Location

Relation: Contact_Location(CLID, Address, City, Phone)
Functional dependencies:

- CLID → Address, City, Phone

Is it a BCNF?
Yes, it's a BCNF because 'CLID' is the primary key and therefore a superkey.

4.1.3 Hospital

Relation: Hospital(HID, Name, City, Region)
Functional dependencies:

- HID → Name, City, Region

Is it a BCNF?
Yes, it's a BCNF because 'HID' is the primary key and therefore a superkey.

4.1.4 Department

Relation: Department(DEP_ID, Name, Specialty, HID)
Functional dependencies:

- DEP_ID → Name, Specialty, HID

Is it a BCNF?
Yes, it's a BCNF because 'DEP_ID' is the primary key and therefore a superkey.

4.1.5 Staff

Relation: Staff(STAFF_ID, FullName, Status)
Functional dependencies:

- STAFF_ID → FullName, Status

Is it a BCNF?
Yes, it's a BCNF because 'STAFF_ID' is the primary key and therefore a superkey.

4.1.6 Practitioner

Relation: Practitioner(STAFF_ID, License_Number, Specialty)
Functional dependencies:

- STAFF_ID → License_Number, Specialty

Is it a BCNF?
Yes, it's a BCNF because 'STAFF_ID' is the primary key and therefore a superkey.

4.1.7 Caregiving

Relation: Caregiving(STAFF_ID, Grade, Ward)
Functional dependencies:

- STAFF_ID → Grade, Ward

Is it a BCNF?
Yes, it's a BCNF because 'STAFF_ID' is the primary key and therefore a superkey.
No decomposition needed.

4.1.8 Technical

Relation: Technical(STAFF_ID, Modality, Certifications)
Functional dependencies:

- STAFF_ID → Modality, Certifications

Is it a BCNF?
Yes, it's a BCNF because 'STAFF_ID' is the primary key and therefore a superkey.
No decomposition needed.

4.1.9 Insurance

Relation: Insurance(InsID, Type)
Functional dependencies:

- InsID → Type

Is it a BCNF?
Yes, it's a BCNF because 'InsID' is the primary key and therefore a superkey.
No decomposition needed.

4.1.10 Clinical Activity

Relation: ClinicalActivity(CAID, Date, Time, DEP_ID, STAFF_ID, IID, ExpID)

Functional dependencies:

- CAID → Title, Time, Date, IID, STAFF_ID, DEP_ID, ExpID

Is it a BCNF?

The **ClinicalActivity** table is a BCNF because 'CAID' is the primary key of the table and therefore the dependencies in this table are those of the primary key which is a superkey.

4.1.11 Appointment

Relation: Appointment(CAID, Reason, Status)

Functional dependencies:

- CAID → Reason, Status, Title, Time, Date, IID, STAFF_ID, DEP_ID

Is it a BCNF?

The **Appointment** table is a BCNF because 'CAID' is the primary key and therefore is a superkey.

4.1.12 Emergency

Relation: Emergency(CAID, TriageLevel, Outcome)

Functional dependencies:

- CAID → TriageLevel, Outcome, Title, Time, Date, IID, STAFF_ID, DEP_ID

Is it a BCNF?

The **Emergency** table is a BCNF because 'CAID' is the primary key and therefore a superkey.

4.1.13 Expense

Relation: Expense(ExpID, Total, InsID, CAID)

Functional dependencies:

- ExpID → Total, InsID, CAID
- CAID → ExpID, Total, InsID

Is it a BCNF?

Yes, it's a BCNF because both 'ExpID' and 'CAID' are candidate keys and therefore superkeys.

4.1.14 Prescription

Relation: Prescription(PID, DateIssued, CAID)

Functional dependencies:

- PID → DateIssued, CAID
- CAID → PID, DateIssued

Is it a BCNF?

Yes, it's a BCNF because both 'PID' and 'CAID' are candidate keys and therefore superkeys.

4.1.15 Medication

Relation: Medication(MID, Name, Form, Strength, ActiveIngredient, TherapeuticClass, Manufacturer)

Functional dependencies:

- MID → Name, Form, Strength, ActiveIngredient, TherapeuticClass, Manufacturer

Is it a BCNF?
Yes, it's a BCNF because 'MID' is the primary key and therefore a superkey.

4.1.16 Has_Contact_Location

Relation: Has_Contact_Location(IID, CLID)

Functional dependencies:

- {IID, CLID} → attributes

Is it a BCNF?
Yes, it's a BCNF because {IID, CLID} is the primary key and therefore a superkey.

4.1.17 Insurance_Covers

Relation: Insurance_Covers(InsID, IID)

Functional dependencies:

- {InsID, IID} → attributes

Is it a BCNF?
Yes, it's a BCNF because {InsID, IID} is the primary key and therefore a superkey.

4.1.18 Include_Medication

Relation: Include_Medication(PID, MID)

Functional dependencies:

- {PID, MID} → attributes

Is it a BCNF?
Yes, it's a BCNF because {PID, MID} is the primary key and therefore a superkey.
No decomposition needed.

4.1.19 Stock

Relation: Stock(HID, MID, StockTimestamp, UnitPrice, Qty, ReorderLevel)

Functional dependencies:

- {HID, MID, StockTimestamp} → UnitPrice, Qty, ReorderLevel

Is it a BCNF?
The **Stock** table is a BCNF because {HID, MID, StockTimestamp} is the primary key of the table and therefore a superkey.

4.1.20 Work_in

Relation: Work_in(STAFF_ID, DEP_ID)

Functional dependencies:

- {STAFF_ID, DEP_ID} → attributes

Is it a BCNF?

Yes, it's a BCNF because {STAFF_ID, DEP_ID} is the composite primary key and therefore the only superkey.

No decomposition needed.

4.2 DDL: Schema Creation

4.2.1 Database Creation

```
CREATE DATABASE MNHS;
USE MNHS;
```

4.2.2 Table Creation

```
-- Patient
CREATE TABLE Patient (
    IID INT PRIMARY KEY,
    CIN VARCHAR(10) UNIQUE NOT NULL,
    FirstName VARCHAR(100) NOT NULL,
    LastName VARCHAR(100) NOT NULL,
    Birth DATE,
    Sex ENUM ('M', 'F') NOT NULL,
    BloodGroup ENUM ('A+', 'A-', 'B+', 'B-', 'O+', 'O-', 'AB+', '
       AB-'),
    Phone VARCHAR(15)
);

-- Hospital
CREATE TABLE Hospital (
    HID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    City VARCHAR(50) NOT NULL,
    Region VARCHAR(50)
);

-- Department
CREATE TABLE Department (
    DEP_ID INT PRIMARY KEY,
    HID INT NOT NULL,
    Name VARCHAR(100) NOT NULL,
    Specialty VARCHAR(100),
    FOREIGN KEY (HID) REFERENCES Hospital(HID)
);

```

```sql
-- Staff
CREATE TABLE Staff (
    STAFF_ID INT PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    Status ENUM ('Active','Retired') DEFAULT 'Active'
);

-- Practitioner / Caregiving / Technical
CREATE TABLE Practitioner (
    STAFF_ID INT PRIMARY KEY,
    License_Number VARCHAR(30),
    Specialty VARCHAR(50),
    FOREIGN KEY (STAFF_ID) REFERENCES Staff(STAFF_ID)
        ON DELETE CASCADE
);

CREATE TABLE Caregiving (
    STAFF_ID INT PRIMARY KEY,
    Grade VARCHAR(50),
    Ward VARCHAR(50),
    FOREIGN KEY (STAFF_ID) REFERENCES Staff(STAFF_ID)
        ON DELETE CASCADE
);

CREATE TABLE Technical (
    STAFF_ID INT PRIMARY KEY,
    Modality VARCHAR(100),
    Certifications VARCHAR(50),
    FOREIGN KEY (STAFF_ID) REFERENCES Staff(STAFF_ID)
        ON DELETE CASCADE
);

-- Clinical Activity
CREATE TABLE ClinicalActivity (
    CAID INT PRIMARY KEY,
    IID INT NOT NULL,
    STAFF_ID INT NOT NULL,
    DEP_ID INT NOT NULL,
    Date DATE NOT NULL,
    Time TIME,
    FOREIGN KEY (IID) REFERENCES Patient(IID),
    FOREIGN KEY (STAFF_ID) REFERENCES Staff(STAFF_ID),
    FOREIGN KEY (DEP_ID) REFERENCES Department(DEP_ID)
);

-- Appointment
CREATE TABLE Appointment (
    CAID INT PRIMARY KEY,
    Reason VARCHAR(100),
    Status ENUM ('Scheduled','Completed','Cancelled')
        DEFAULT 'Scheduled',
```

```
    FOREIGN KEY (CAID) REFERENCES ClinicalActivity(CAID)
);

-- Emergency
CREATE TABLE Emergency (
    CAID INT PRIMARY KEY,
    TriageLevel INT CHECK (TriageLevel BETWEEN 1 AND 5),
    Outcome ENUM('Discharged','Admitted','Transferred','Deceased')
        ,
    FOREIGN KEY (CAID) REFERENCES ClinicalActivity(CAID)
);

-- Insurance
CREATE TABLE Insurance (
    InsID INT PRIMARY KEY,
    Type ENUM('CNOPS','CNSS','RAMED','Private','None') NOT NULL
);

CREATE TABLE Insurance_Covers(
    InsID CHAR(11),
    IID INT,
    PRIMARY KEY (IID,InsID),
    FOREIGN KEY (IID) REFERENCES Patient(IID),
    FOREIGN KEY (InsID) REFERENCES Insurance(InsID)
);

-- Expense
CREATE TABLE Expense (
    ExpID INT PRIMARY KEY,
    InsID INT,
    CAID INT UNIQUE NOT NULL,
    Total DECIMAL(10,2) NOT NULL CHECK (Total >= 0),
    FOREIGN KEY (InsID) REFERENCES Insurance(InsID),
    FOREIGN KEY (CAID) REFERENCES ClinicalActivity(CAID)
);

-- Medication
CREATE TABLE Medication (
    MID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Form VARCHAR(50),
    Strength VARCHAR(50),
    ActiveIngredient VARCHAR(100),
    TherapeuticClass VARCHAR(100),
    Manufacturer VARCHAR(100)
);

-- Stock
CREATE TABLE Stock (
    HID INT,
    MID INT,
```

```sql
    StockTimestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
    UnitPrice DECIMAL(10,2) CHECK (UnitPrice >= 0),
    Qty INT DEFAULT 0 CHECK (Qty >= 0),
    ReorderLevel INT DEFAULT 10 CHECK (ReorderLevel >= 0),
    PRIMARY KEY (HID, MID, StockTimestamp),
    FOREIGN KEY (HID) REFERENCES Hospital(HID),
    FOREIGN KEY (MID) REFERENCES Medication(MID)
);

-- Prescription
CREATE TABLE Prescription (
    PID INT PRIMARY KEY,
    CAID INT UNIQUE NOT NULL,
    DateIssued DATE NOT NULL,
    FOREIGN KEY (CAID) REFERENCES ClinicalActivity(CAID)
);

-- Include_Medication (Prescription <-> Medication)
CREATE TABLE Include_Medication (
    PID INT,
    MID INT,
    PRIMARY KEY (PID, MID),
    FOREIGN KEY (PID) REFERENCES Prescription(PID),
    FOREIGN KEY (MID) REFERENCES Medication(MID)
);

-- Contact Location + Has_Contact_Location
CREATE TABLE Contact_Location (
    CLID INT PRIMARY KEY,
    Address VARCHAR(200),
    City VARCHAR(50),
    Phone VARCHAR(20)
);

CREATE TABLE Has_Contact_Location (
    IID INT,
    CLID INT,
    PRIMARY KEY(IID, CLID),
    FOREIGN KEY (IID) REFERENCES Patient(IID),
    FOREIGN KEY (CLID) REFERENCES Contact_Location(CLID)
);

-- Work_In (Staff <-> Department)
CREATE TABLE Work_In (
    STAFF_ID INT,
    DEP_ID INT,
    PRIMARY KEY(STAFF_ID, DEP_ID),
    FOREIGN KEY (STAFF_ID) REFERENCES Staff(STAFF_ID),
    FOREIGN KEY (DEP_ID) REFERENCES Department(DEP_ID)
);
```

4.3 DML: Data Manipulation

4.3.1 Sample Data Insertion

```
-- Patient Data
INSERT INTO Patient VALUES
(1,'AA123456','Sara','El Mansouri','1995-04-12','F','A+','
   0611223344'),
(2,'BB987654','Youssef','Haddad','1988-11-03','M','O-','0677889900
   '),
(3,'CC556677','Mona','Bennani','2001-07-21','F','B+','0655332211')
   ,
(4,'DD112233','Omar','Chakir','1979-02-17','M','AB+','0611557799')
   ,
(5,'EE998877','Imane','Fassi','1990-12-30','F','O+','0622446688');

-- Hospital Data
INSERT INTO Hospital VALUES
(1,'CHU Rabat','Rabat','Rabat-Sale'),
(2,'CHU Marrakech','Marrakech','Marrakech-Safi'),
(3,'CHU Casablanca','Casablanca','Casablanca-Settat'),
(4,'Avicenne','Rabat','Rabat-Sale'),
(5,'Ibn Sina','Fes','Fes-Meknes');

-- Department Data
INSERT INTO Department VALUES
(10,1,'Cardiology','Heart'),
(11,1,'Emergency','Urgent Care'),
(12,2,'Pediatrics','Children'),
(13,3,'Oncology','Cancer'),
(14,4,'Radiology','Imaging');

-- Staff Data
INSERT INTO Staff VALUES
(100,'Dr. Ahmed Idrissi','Active'),
(101,'Dr. Salma Zahra','Active'),
(102,'Nurse Laila Amrani','Active'),
(103,'Tech Reda El Fassi','Active'),
(104,'Dr. Yassine Toumi','Retired');

-- Practitioner Data
INSERT INTO Practitioner VALUES
(100,'LIC123','Cardiology'),
(101,'LIC456','Pediatrics'),
(104,'LIC789','Internal Medicine');

-- Caregiving Data
INSERT INTO Caregiving VALUES
(102,'Senior Nurse','Ward A'),
(105,'Nurse','Ward B'),
(106,'Assistant Nurse','Ward C'),
(107,'Head Nurse','Ward D'),
```

```
(108,'Nurse','Ward E');

-- Technical Data
INSERT INTO Technical VALUES
(103,'Radiology','CT Certified'),
(109,'Lab','Blood Analysis'),
(110,'Radiology','MRI Certified'),
(111,'Surgery','Sterilization'),
(112,'Maintenance','BioMed Equipment');

-- Clinical Activity Data
INSERT INTO ClinicalActivity VALUES
(500,1,100,10,'2025-02-10','09:00:00'),
(501,2,101,12,'2025-02-11','10:30:00'),
(502,3,102,11,'2025-02-12','14:00:00'),
(504,5,104,13,'2025-02-14','11:15:00'),
(600,1,100,10,'2025-02-10','09:00:00'),
(601,2,101,12,'2025-02-11','10:30:00'),
(602,3,102,11,'2025-02-12','14:00:00'),
(603,4,103,14,'2025-02-13','08:45:00'),
(604,5,104,13,'2025-02-14','11:15:00');

-- Appointment Data
INSERT INTO Appointment VALUES
(500,'Routine Checkup','Scheduled'),
(501,'Follow-up','Completed'),
(502,'Vaccination','Scheduled'),
(504,'Consultation','Scheduled');

-- Emergency Data
INSERT INTO Emergency VALUES
(600,3,'Admitted'),
(601,5,'Transferred'),
(602,1,'Discharged'),
(603,4,'Admitted'),
(604,2,'Deceased');

-- Insurance Data
INSERT INTO Insurance VALUES
(1,'CNOPS'),
(2,'CNSS'),
(3,'RAMED'),
(4,'Private'),
(5,'None');

-- Insurance_Covers Data
INSERT INTO Insurance_Covers VALUES
('1',1),
('2',2),
('3',3),
('4',4),
```

```
('2',5);

-- Expense Data
INSERT INTO Expense VALUES
(900,1,500,250.00),
(901,2,501,120.00),
(902,3,502,80.00),
(904,5,504,0.00);

-- Medication Data
INSERT INTO Medication VALUES
(200,'Amoxicillin','Capsule','500mg','Amoxicillin',
     'Antibiotic','Pfizer'),
(201,'Paracetamol','Tablet','1g','Acetaminophen',
     'Analgesic','Sanofi'),
(202,'Ibuprofen','Tablet','400mg','Ibuprofen',
     'Anti-inflammatory','Bayer'),
(203,'Ceftriaxone','Injection','1g','Ceftriaxone',
     'Antibiotic','Roche'),
(204,'Azithromycin','Tablet','500mg','Azithromycin',
     'Antibiotic','Pfizer');

-- Stock Data
INSERT INTO Stock (HID, MID, UnitPrice, Qty) VALUES
(1,200,50,100),
(1,201,20,200),
(2,202,35,300),
(3,203,120,50),
(4,204,90,80);

-- Prescription Data
INSERT INTO Prescription VALUES
(300,500,'2025-02-10'),
(301,501,'2025-02-11'),
(302,502,'2025-02-12'),
(304,504,'2025-02-14');

-- Include_Medication Data
INSERT INTO Include_Medication VALUES
(300,200),
(301,201),
(302,202),
(304,204);

-- Contact_Location Data
INSERT INTO Contact_Location VALUES
(1,'123 Rue Hassan II','Rabat','0611223344'),
(2,'45 Bd Zerktouni','Casablanca','0622334455'),
(3,'Hay Illiot','Fes','0633445566'),
(4,'Centre Ville','Marrakech','0644556677'),
(5,'Oued Fes','Fes','0655667788');
```

```sql

-- Has_Contact_Location Data
INSERT INTO Has_Contact_Location VALUES
(1,1),
(2,2),
(3,3),
(4,4),
(5,5);

-- Work_In Data
INSERT INTO Work_In VALUES
(100,10),
(101,12),
(102,11),
(103,14),
(104,13);
```

4.3.2 Update Operations

```sql
-- Update a patient's phone number
UPDATE Patient
SET Phone = '0611223344'
WHERE IID = 1;

-- Update a hospital's region
UPDATE Hospital
SET Region = 'Rabat-Sale'
WHERE HID = 1;
```

4.3.3 Delete Operations

```sql
-- Delete a scheduled appointment that was cancelled
DELETE FROM Appointment
WHERE CAID = 503 AND Status = 'Cancelled';

-- Note: Due to foreign key constraints, we must also delete
-- the corresponding clinical activity
DELETE FROM ClinicalActivity
WHERE CAID = 503;
```

4.4 SQL Queries

1. **Select all patients ordered by last name.**

```sql
SELECT *
FROM Patient
ORDER BY LastName;
```

Query Result:

IID	CIN	FirstName	LastName	Birth	Sex	BloodGroup	Phone
3	CC556677	Mona	Bennani	2001-07-21	F	B+	0655332211
4	DD112233	Omar	Chakir	1979-02-17	M	AB+	0611557799
1	AA123456	Sara	El Mansouri	1995-04-12	F	A+	0611223344
5	EE998877	Imane	Fassi	1990-12-30	F	O+	0622446688
2	BB987654	Youssef	Haddad	1988-11-03	M	O-	0677889900

2. **List distinct insurance types.**

```
SELECT DISTINCT Type
FROM Insurance;
```

Query Result:

Type
CNOPS
CNSS
RAMED
Private
None

3. **Retrieve staff who work in hospitals located in Rabat.**

```
SELECT DISTINCT S.STAFF_ID, S.FullName, S.Status
FROM Staff S
JOIN Work_in W ON W.STAFF_ID = S.STAFF_ID
JOIN Department D ON D.DEP_ID = W.DEP_ID
JOIN Hospital H ON H.HID = D.HID
WHERE H.City = 'Rabat';
```

Query Result:

STAFF_ID	FullName	Status
100	Dr. Ahmed Idrissi	Active
102	Nurse Laila Amrani	Active
103	Tech Reda El Fassi	Active

4. **Find all appointments that are scheduled within the next seven days.**

```
SELECT CA.CAID, CA.IID, CA.STAFF_ID, CA.DEP_ID, CA.Date, CA.
   Time
FROM ClinicalActivity CA
JOIN Appointment A ON CA.CAID = A.CAID
WHERE A.Status = 'Scheduled'
    AND CA.Date BETWEEN CURRENT_DATE
        AND CURRENT_DATE + INTERVAL 7 DAY;
```

Query Result:

CAID	IID	STAFF_ID	DEP_ID	Date	Time
No results (depends on current date)					

5. **Count the number of appointments per department.**

```
SELECT D.DEP_ID, D.Name, COUNT(A.CAID) AS AppointmentCount
FROM Department D
LEFT JOIN ClinicalActivity CA ON D.DEP_ID = CA.DEP_ID
LEFT JOIN Appointment A ON CA.CAID = A.CAID
GROUP BY D.DEP_ID, D.Name;
```

Query Result:

DEP_ID	Name	AppointmentCount
10	Cardiology	1
11	Emergency	1
12	Pediatrics	1
13	Oncology	1
14	Radiology	0

6. **Compute the average unit price of medications per hospital.**

```
SELECT HID, AVG(UnitPrice) AS AvgUnitPrice
FROM Stock
GROUP BY HID;
```

Query Result:

HID	AvgUnitPrice
1	35.000000
2	35.000000
3	120.000000
4	90.000000

7. **List hospitals with more than twenty emergency admissions.**

```
SELECT
    h.HID,
    h.Name AS HospitalName,
    h.City,
    h.Region,
    COUNT(*) AS EmergencyAdmissions
FROM Hospital h
INNER JOIN Department d ON h.HID = d.HID
INNER JOIN ClinicalActivity ca ON d.DEP_ID = ca.DEP_ID
INNER JOIN Emergency e ON e.CAID = ca.CAID
WHERE e.Outcome = 'Admitted'
GROUP BY h.HID, h.Name, h.City, h.Region
HAVING COUNT(*) > 20;
```

Query Result:

HID	Name	City	Region
No results (no hospital has > 20 admissions)			

8. **Find medications in the therapeutic class Antibiotic where the unit price is below two hundred.**

```sql
SELECT DISTINCT M.Name
FROM Medication M
JOIN Stock S ON S.MID = M.MID
WHERE S.UnitPrice < 200
    AND M.TherapeuticClass = 'Antibiotic';
```

Query Result:

Name
Amoxicillin
Ceftriaxone
Azithromycin

9. **For each hospital list the top three most expensive medications.**

```sql
SELECT S1.HID, S1.MID, S1.UnitPrice
FROM Stock S1
WHERE (
    SELECT Count(*)
    FROM Stock S2
    WHERE S1.HID = S2.HID
        AND S1.UnitPrice < S2.UnitPrice
) < 3
ORDER BY S1.HID, S1.UnitPrice DESC;
```

Query Result:

HID	MID	UnitPrice
1	200	50.00
1	201	20.00
2	202	35.00
3	203	120.00
4	204	90.00

10. **For each department return counts of Scheduled Completed and Cancelled appointments in a single result.**

```sql
SELECT
    D.DEP_ID,
    D.Name AS DepartmentName,
    SUM(CASE WHEN A.Status = 'Scheduled' THEN 1 ELSE 0 END)
        AS totalScheduled,
    SUM(CASE WHEN A.Status = 'Cancelled' THEN 1 ELSE 0 END)
        AS totalCancelled,
    SUM(CASE WHEN A.Status = 'Completed' THEN 1 ELSE 0 END)
        AS totalCompleted
FROM Department D
LEFT JOIN ClinicalActivity CA ON D.DEP_ID = CA.DEP_ID
LEFT JOIN Appointment A ON CA.CAID = A.CAID
GROUP BY D.DEP_ID, D.Name;
```

Query Result:

DEP_ID	Name	Scheduled	Cancelled	Completed
10	Cardiology	1	0	0
11	Emergency	1	0	0
12	Pediatrics	0	0	1
13	Oncology	1	0	0
14	Radiology	0	0	0

11. **List patients who have no scheduled appointments in the next thirty days.**

```sql
SELECT DISTINCT P.*
FROM Patient P
LEFT JOIN ClinicalActivity CA
    ON CA.IID = P.IID
LEFT JOIN Appointment A
    ON CA.CAID = A.CAID
    AND A.Status = 'Scheduled'
    AND CA.Date BETWEEN CURRENT_DATE
        AND (CURRENT_DATE + INTERVAL 30 DAY)
WHERE A.CAID IS NULL;
```

Query Result:

IID	CIN	FirstName	LastName	Birth	Sex	Blood	Phone
1	AA123456	Sara	El Mansouri	1995-04-12	F	A+	0611223344
2	BB987654	Youssef	Haddad	1988-11-03	M	O-	0677889900
3	CC556677	Mona	Bennani	2001-07-21	F	B+	0655332211
4	DD112233	Omar	Chakir	1979-02-17	M	AB+	0611557799
5	EE998877	Imane	Fassi	1990-12-30	F	O+	0622446688

12. **For each staff member compute the total number of appointments and the percentage share of appointments in their hospital.**

```sql
SELECT
    s.STAFF_ID,
    s.FullName,
    h.HID AS HospitalID,
    COUNT(a.CAID) AS AppointmentCount,
    COUNT(a.CAID) * 100.0 /
    NULLIF (
        (
            SELECT COUNT(*)
            FROM Appointment a2
            JOIN ClinicalActivity ca2 ON a2.CAID = ca2.CAID
            JOIN Department d2 ON d2.DEP_ID = ca2.DEP_ID
            JOIN Hospital h2 ON h2.HID = d2.HID
            WHERE h2.HID = h.HID
        ),
        0
```

```
    ) AS PercentageShare
FROM Staff s
LEFT JOIN ClinicalActivity ca
    ON s.STAFF_ID = ca.STAFF_ID
LEFT JOIN Appointment a
    ON a.CAID = ca.CAID
LEFT JOIN Department d
    ON d.DEP_ID = ca.DEP_ID
LEFT JOIN Hospital h
    ON h.HID = d.HID
GROUP BY
    s.STAFF_ID,
    s.FullName,
    h.HID;
```

Query Result:

STAFF_ID	FullName	HospitalID	AppointmentCount	PercentageShare
100	Ahmed Idrissi	1	1	50.00000
101	Salma Zahra	2	1	100.00000
102	Laila Amrani	1	1	50.00000
103	Reda El Fassi	4	1	100.00000
104	Yassine Toumi	3	1	100.00000
105	Amina Rahali	*NULL*	0	*NULL*
106	Yassin El Baroudi	*NULL*	0	*NULL*
107	Hajar Benhima	*NULL*	0	*NULL*
108	Othmane Karimi	*NULL*	0	*NULL*

13. **Show all drugs that are below ReorderLevel in at least one hospital and include the list of those hospitals.**

```
SELECT DISTINCT M.Name, H.Name AS HospitalName
FROM Medication M
JOIN Stock S ON M.MID = S.MID
JOIN Hospital H ON H.HID = S.HID
WHERE S.Qty < S.ReorderLevel;
```

Query Result:

Medication Name	Hospital Name
No results (all stocks above reorder level)	

14. **Find hospitals that stock every antibiotic in the catalog.**

```
SELECT H.HID, H.Name
FROM Hospital H
WHERE (
    SELECT Count(*)
    FROM Stock S
    JOIN Medication M ON M.MID = S.MID
```

```
    WHERE M.TherapeuticClass = 'Antibiotic'
        AND H.HID = S.HID
) = (
    SELECT Count(*)
    FROM Medication
    WHERE TherapeuticClass = 'Antibiotic'
);
```

Query Result:

HID	Name	City	Region
No results (no hospital stocks all antibiotics)			

15. **For each hospital and drug class return the average unit price and flag whether it is above the citywide average for that class.**

```
SELECT
    H.Name AS HospitalName,
    M.TherapeuticClass,
    AVG(S.UnitPrice) AS local_average_prc,
    (
        SELECT AVG(S2.UnitPrice)
        FROM Stock S2
        JOIN Hospital H2 ON S2.HID = H2.HID
        JOIN Medication M2 ON S2.MID = M2.MID
        WHERE H2.City = H.City
            AND M2.TherapeuticClass = M.TherapeuticClass
    ) AS city_average_prc,
    CASE
        WHEN AVG(S.UnitPrice) > (
            SELECT AVG(S2.UnitPrice)
            FROM Stock S2
            JOIN Hospital H2 ON S2.HID = H2.HID
            JOIN Medication M2 ON S2.MID = M2.MID
            WHERE H2.City = H.City
                AND M2.TherapeuticClass = M.TherapeuticClass
        ) THEN 'Above City Average'
        WHEN AVG(S.UnitPrice) < (
            SELECT AVG(S2.UnitPrice)
            FROM Stock S2
            JOIN Hospital H2 ON S2.HID = H2.HID
            JOIN Medication M2 ON S2.MID = M2.MID
            WHERE H2.City = H.City
                AND M2.TherapeuticClass = M.TherapeuticClass
        ) THEN 'Below City Average'
        ELSE 'At City Average'
    END AS Flag
FROM Hospital H
JOIN Stock S ON H.HID = S.HID
JOIN Medication M ON S.MID = M.MID
GROUP BY H.HID, H.Name, M.TherapeuticClass, H.City;
```

Query Result:

Hospital	Class	Local Avg	City Avg	Flag
CHU Rabat	Analgesic	20.00	20.00	At City Average
CHU Rabat	Antibiotic	50.00	70.00	Below City Average
CHU Marrakech	Anti-inflammatory	35.00	35.00	At City Average
CHU Casablanca	Antibiotic	120.00	120.00	At City Average
Avicenne	Antibiotic	90.00	70.00	Above City Average

16. **Return the next appointment date for each patient.**

```sql
SELECT CA.IID, P.FirstName, P.LastName,
       MIN(CA.Date) AS NextAppointmentDate
FROM ClinicalActivity CA
JOIN Appointment A
    ON CA.CAID = A.CAID
JOIN Patient P
    ON CA.IID = P.IID
WHERE A.Status = 'Scheduled'
    AND CA.Date >= CURRENT_DATE
GROUP BY CA.IID, P.FirstName, P.LastName;
```

Query Result:

CAID	IID	STAFF_ID	DEP_ID	Date	Time
500	1	100	10	2025-02-10	09:00:00
501	2	101	12	2025-02-11	10:30:00
502	3	102	11	2025-02-12	14:00:00
503	4	103	14	2025-02-13	08:45:00
504	5	104	13	2025-02-14	11:15:00
600	1	100	10	2025-02-10	09:00:00
601	2	101	12	2025-02-11	10:30:00
602	3	102	11	2025-02-12	14:00:00
603	4	103	14	2025-02-13	08:45:00
604	5	104	13	2025-02-14	11:15:00

17. **Among patients with at least two emergency visits list those whose latest emergency visit was within the last fourteen days.**

```sql
SELECT
    p.IID,
    p.FirstName,
    p.LastName,
    MAX(ca.Date) AS LatestEmergencyDate,
    COUNT(e.CAID) AS EmergencyVisitCount
FROM Patient p
INNER JOIN ClinicalActivity ca
    ON p.IID = ca.IID
INNER JOIN Emergency e
    ON e.CAID = ca.CAID
```

```
GROUP BY
    p.IID,
    p.FirstName,
    p.LastName
HAVING
    COUNT(e.CAID) >= 2
    AND MAX(ca.Date) >= CURRENT_DATE - INTERVAL 14 DAY;
```

Query Result:

IID	Name	Latest Date	Visit Count
No results (no patients meet criteria)			

18. **For each city rank hospitals by the number of completed appointments in the last ninety days.**

```
SELECT
    H.City,
    H.Name AS HospitalName,
    COUNT(A.CAID) AS CompletedAppointments
FROM Hospital H
JOIN Department D
    ON D.HID = H.HID
JOIN ClinicalActivity CA
    ON CA.DEP_ID = D.DEP_ID
JOIN Appointment A
    ON A.CAID = CA.CAID
WHERE A.Status = 'Completed'
    AND CA.Date >= CURRENT_DATE - INTERVAL 90 DAY
GROUP BY H.City, H.HID, H.Name
ORDER BY H.City, CompletedAppointments DESC;
```

Query Result:

City	Hospital Name	Completed
No results (depends on date range)		

19. **Within each city return medications whose hospital prices show a spread greater than thirty percent between minimum and maximum.**

```
-- Create views for max and min prices
CREATE VIEW CitiesMax AS
SELECT City, MID, MAX(UnitPrice) AS max
FROM Hospital H
JOIN Stock S ON H.HID = S.HID
GROUP BY City, MID;

CREATE VIEW CitiesMin AS
SELECT City, MID, MIN(UnitPrice) AS min
FROM Hospital H
JOIN Stock S ON H.HID = S.HID
```

```sql
GROUP BY City, MID;

-- Query using the views
SELECT DISTINCT H.City, S.MID, M.Name
FROM Hospital H
JOIN Stock S ON H.HID = S.HID
JOIN Medication M ON S.MID = M.MID
WHERE (
    (SELECT max
     FROM CitiesMax Ma
     WHERE H.City = Ma.City AND S.MID = Ma.MID)
    > 1.30 *
    (SELECT min
     FROM CitiesMin Mi
     WHERE H.City = Mi.City AND S.MID = Mi.MID)
);
```

Query Result:

City	MID
No results (no spread > 30%)	

20. **Data quality check on stock entries list rows with negative quantity or non positive unit price.**

```sql
SELECT HID, MID, StockTimestamp, UnitPrice, Qty, ReorderLevel
FROM Stock
WHERE Qty < 0 OR UnitPrice <= 0;
```

Query Result:

HID	MID	Timestamp	Price	Qty	Reorder
No results (all data valid)					

5 Discussion

In this lab, we worked on some new SQL concepts, we discovered new and more advanced syntax, we dealt with some date manipulations and queries involving multiple joins. We also adjusted some parts of the schema to simplify data retrieval, for example in the patient table, we switched full name out for first and last name.

Overall, we found the SQL queries significantly more difficult than the previous labs, they sometimes required nested subqueries which made it very challenging. To overcome this challenge, we did some extra research and we used AI tools for verification, as many of these queries used new concepts and advanced syntax.

6 Conclusion

This lab brought the theory of normalization to life, we implemented an actual functional MNHS database on MySQL. We realized the importance of a clean normalized structure when we started writing queries and applying modifications on the tables.

We did find some difficulty writing the queries, as many of them included new concepts and syntax, but the challenges we faced helped us acquire new information and get more comfortable with SQL. Overall, this lab demonstrated how a well-designed database made data easier to manage and to work with in real life scenarios.