University
Mohammed VI
Polytechnic

College of Computing

BCNF - Decomposition - Queries

Data Management Course

UM6P College of Computing

Professor: Karima Echihabi **Program:** Computer Engineering
Session: Fall 2025

Team Information

Team Name	Sekel
Member 1	Anass Mahdad
Member 2	Rania Moujahed
Member 3	Haitam Laghmam
Member 4	Abdelmoughit Labrigui
Member 5	Youssef Ouazzani
Member 6	Mohammed Idriss Nana
Repository Link	`https://github.com/Data-Project-Hierarchy/Sekel`

1 Introduction

Before building a reliable database, we must ensure that our schema does not introduce problems such as redundancy, loss of data integrity, or anomalies. A well-designed schema forms the foundation for any database system, ensuring that information is organized, consistent, and easily retrievable. Proper database design also improves performance and simplifies future maintenance. After refining and obtaining a correct schema, we must also know how to alter and manipulate the database using SQL queries and effectively work with the data it holds. This allows us to interact with the system, extract meaningful insights, and ensure the database meets real-world needs.

2 Requirements

Normalization

- Validate each relation against BCNF.
- Verify lossless joins and dependency preservation after decomposition.

DDL: Schema Creation

- Create tables for MNHS entities (use exact names and keys from the logical schema).
- Define primary and foreign keys and constraints (NOT NULL, UNIQUE).
- Alter a table to add an attribute (e.g., add Email to Patient).

DML: Data Manipulation

- Insert at least 5 sample rows per table.
- Update a patient's phone number and a hospital's region.
- Delete a scheduled appointment that was cancelled.

Queries

The following queries should be executed on the MNHS schema created:

1. Select all patients ordered by last name.
2. List distinct insurance types.
3. Retrieve staff who work in hospitals located in Rabat.
4. Find all appointments that are scheduled within the next seven days.
5. Count the number of appointments per department.

6. Compute the average unit price of medications per hospital.
7. List hospitals with more than twenty emergency admissions.
8. Find medications in the therapeutic class Antibiotic where the unit price is below two hundred.
9. For each hospital list the top three most expensive medications.
10. For each department return counts of Scheduled, Completed, and Cancelled appointments in a single result.
11. List patients who have no scheduled appointments in the next thirty days.
12. For each staff member compute the total number of appointments and the percentage share of appointments in their hospital.
13. Show all drugs that are below ReorderLevel in at least one hospital and include the list of those hospitals.
14. Find hospitals that stock every antibiotic in the catalog.
15. For each hospital and drug class return the average unit price and flag whether it is above the citywide average for that class.
16. Return the next appointment date for each patient.
17. Among patients with at least two emergency visits list those whose latest emergency visit was within the last fourteen days.
18. For each city rank hospitals by the number of completed appointments in the last ninety days.
19. Within each city return medications whose hospital prices show a spread greater than thirty percent between minimum and maximum.
20. Data quality check on stock entries list rows with negative quantity or non positive unit price.

3 Implementation & Results

DataImplementation

```sql
-- DDL Statements to create MNHS Database and Tables
CREATE DATABASE MNHS;
USE MNHS;

CREATE TABLE Patient (
    IID INT PRIMARY KEY,
    CIN VARCHAR(10) UNIQUE NOT NULL,
    FullName 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)
);

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

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

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

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)
);

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)
);

CREATE TABLE Appointment (
    CAID INT PRIMARY KEY,
```

```
    Reason VARCHAR(100),
    Status ENUM('Scheduled','Completed','Cancelled') DEFAULT 'Scheduled',
    FOREIGN KEY(CAID) REFERENCES ClinicalActivity(CAID)
);

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)
);

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

CREATE TABLE Covers (
    InsID INT,
    IID INT,
    PRIMARY KEY(InsID, IID),
    FOREIGN KEY(InsID) REFERENCES Insurance(InsID),
    FOREIGN KEY(IID) REFERENCES Patient(IID)
);

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)
);

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

CREATE TABLE Stock (
    HID INT,
    MID INT,
    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)
);

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

CREATE TABLE Includes (
    PID INT,
    MID INT,
    Dosage VARCHAR(100),
    Duration VARCHAR(50),
    PRIMARY KEY(PID, MID),
    FOREIGN KEY(PID) REFERENCES Prescription(PID),
    FOREIGN KEY(MID) REFERENCES Medication(MID)
);

CREATE TABLE ContactLocation (
    CLID INT PRIMARY KEY,
    City VARCHAR(50),
    Province VARCHAR(50),
    Street VARCHAR(100),
    Number VARCHAR(10),
    PostalCode VARCHAR(10),
    Phone_Location VARCHAR(15)
);

CREATE TABLE Have (
    IID INT,
    CLID INT,
    PRIMARY KEY(IID, CLID),
    FOREIGN KEY(IID) REFERENCES Patient(IID),
    FOREIGN KEY(CLID) REFERENCES ContactLocation(CLID)
);

-- ALTER Statements to modify MNHS Tables

ALTER TABLE Patient ADD Email VARCHAR(100);
ALTER TABLE Hospital MODIFY Region VARCHAR(100);

-----------------------------------------------
-- DML Statements to insert sample data into MNHS Database
```

```
INSERT INTO Hospital VALUES
    (1, 'Benguerir Central Hospital', 'Benguerir', 'Marrakech-Safi'),
    (2, 'Casablanca University Hospital', 'Casablanca', 'Casablanca-Settat'),
    (3, 'Rabat General Hospital', 'Rabat', 'Rabat-Salé-Kénitra'),
    (4, 'Marrakech City Hospital', 'Marrakech', 'Marrakech-Safi'),
    (5, 'Tangier Regional Hospital', 'Tangier', 'Tanger-Tetouan-Al Hoceima');

INSERT INTO Department VALUES
    (10, 1, 'Cardiology', 'Heart Care'),
    (20, 2, 'Radiology', 'Imaging'),
    (30, 3, 'Neurology', 'Brain and Nerve Care'),
    (40, 4, 'Pediatrics', 'Child Health'),
    (50, 5, 'Orthopedics', 'Bone and Joint Care');

INSERT INTO Patient VALUES
    (1, 'CIN123', 'Sara El Amrani', '1999-04-10', 'F', 'A+', '0612345678',
    'Sara@um6p.ma'),
    (2, 'CIN456', 'Youssef Benali', '1988-09-22', 'M', 'O-', '0678912345',
    'Youssef@um6p.ma'),
    (3, 'CIN789', 'Laila Haddad', '2001-12-05', 'F', 'B+', '0654321987',
    'Laila@um6p.ma'),
    (4, 'CIN321', 'Rachid El Fassi', '1975-07-15', 'M', 'AB-', '0687654321',
    'Rachid@um6p.ma'),
    (5, 'CIN654', 'Nadia Choufani', '1995-03-30', 'F', 'O+', '0623456789',
    'Nadia@um6p.ma');

INSERT INTO ContactLocation VALUES
    (1, 'Benguerir', 'Marrakech-Safi', 'Main St', '12', '43100', '0612345678'),
    (2, 'Casablanca', 'Casablanca-Settat', 'Ocean Ave', '45', '20000', '0678912345'),
    (3, 'Rabat', 'Rabat-Salé-Kénitra', 'King St', '78', '10000', '0654321987'),
    (4, 'Marrakech', 'Marrakech-Safi', 'Palm Rd', '34', '40000', '0687654321'),
    (5, 'Tangier', 'Tanger-Tetouan-Al Hoceima', 'Harbor Blvd', '56', '90000',
    '0623456789');

INSERT INTO Have VALUES
    (1, 1),
    (2, 2),
    (3, 3),
    (4, 4),
    (5, 5);

INSERT INTO Staff VALUES
    (501, 'Dr. Amina Idrissi', 'Active'),
    (502, 'Technician Omar Lahlou', 'Active'),
    (503, 'Nurse Fatima Zahra', 'Active'),
    (504, 'Dr. Yassine El Khatib', 'Retired'),
    (505, 'Technician Salma Rachid', 'Active'),
    (506, 'Nurse Ahmed Benjelloun', 'Retired');
```

```sql
INSERT INTO ClinicalActivity VALUES
    (1001, 1, 501, 10, '2025-10-10', '10:00:00'),
    (1002, 2, 502, 20, '2025-10-12', '11:00:00'),
    (1003, 3, 503, 30, '2025-10-15', '09:30:00'),
    (1004, 4, 504, 40, '2025-10-18', '14:00:00'),
    (1005, 3, 504, 40, '2025-10-18', '15:00:00'),
    (1006, 2, 504, 40, '2025-10-18', '16:00:00'),
    (1007, 4, 502, 40, '2025-10-18', '17:00:00'),
    (1010, 5, 505, 50, '2025-10-20', '13:00:00');

INSERT INTO Appointment VALUES
    (1001, 'Routine check-up', 'Scheduled'),
    (1002, 'Follow-up imaging', 'Completed'),
    (1003, 'Neurological assessment', 'Cancelled'),
    (1004, 'Pediatric consultation', 'Scheduled'),
    (1005, 'Orthopedic evaluation', 'Completed');

INSERT INTO Emergency VALUES
    (1003, 2, 'Admitted'),
    (1004, 1, 'Discharged'),
    (1005, 3, 'Transferred');

INSERT INTO Work_in VALUES
    (501, 10),
    (502, 20),
    (503, 30),
    (504, 40),
    (505, 50),
    (506, 10);

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

INSERT INTO Covers VALUES
    (1, 1),
    (2, 2),
    (3, 3),
    (4, 4),
    (5, 5);

INSERT INTO Expense VALUES
    (2001, 1, 1001, 150.00),
    (2002, 2, 1002, 300.00),
    (2003, 3, 1003, 500.00),
    (2004, 4, 1004, 200.00),
```

```
    (2005, 5, 1005, 400.00);

INSERT INTO Medication VALUES
    (301, 'Paracetamol', 'Tablet', '500mg', 'Paracetamol', 'Analgesic',
    'PharmaCorp'),
    (302, 'Amoxicillin', 'Capsule', '250mg', 'Amoxicillin', 'Antibiotic',
    'BioMed'),
    (303, 'Ibuprofen', 'Tablet', '200mg', 'Ibuprofen', 'Anti-inflammatory',
    'MediHealth'),
    (304, 'Lisinopril', 'Tablet', '10mg', 'Lisinopril', 'Antihypertensive',
    'HealthPlus'),
    (305, 'Metformin', 'Tablet', '500mg', 'Metformin', 'Antidiabetic',
    'WellnessLabs');

INSERT INTO Stock VALUES
    (1, 301, '2025-10-01 09:00:00', 0.50, 100, 20),
    (1, 302, '2025-10-01 09:00:00', 1.00, 50, 10),
    (2, 303, '2025-10-02 10:00:00', 0.75, 200, 30),
    (2, 304, '2025-10-02 10:00:00', 1.50, 80, 15),
    (3, 305, '2025-10-03 11:00:00', 0.60, 150, 25);

INSERT INTO Prescription VALUES
    (4001, 1001, '2025-10-10'),
    (4002, 1002, '2025-10-12'),
    (4003, 1003, '2025-10-15'),
    (4004, 1004, '2025-10-18'),
    (4005, 1005, '2025-10-20');

INSERT INTO Includes VALUES
    (4001, 301, '500mg twice daily', '5 days'),
    (4002, 302, '250mg three times daily', '7 days'),
    (4003, 303, '200mg every 6 hours', '3 days'),
    (4004, 304, '10mg once daily', '30 days'),
    (4005, 305, '500mg twice daily', '60 days');
```

Before moving to the queries note that this report includes outputs where some values appearing were not initially present in the database implementation but were locally added by the persons responsible for the corresponding queries for testing purposes

Query 1

```
-- 1. Select all patients ordered by last name
SELECT *
FROM Patient
ORDER BY FullName;
```

IID	CIN	FullName	Birth	Sex	BloodGroup	Phone
3	CIN789	Laila Haddad	2001-12-05	F	B+	0654321987
5	CIN654	Nadia Choufani	1995-03-30	F	O+	0623456789
4	CIN321	Rachid El Fassi	1975-07-15	M	AB-	0687654321
1	CIN123	Sara El Amrani	1999-04-10	F	A+	0612345678
2	CIN456	Youssef Benali	1988-09-22	M	O-	0678912345

Figure 1: Query 12 Execution Results

Explanation: Retrieves all patient records sorted alphabetically by their full names (no attribute last name).

Query 2

```sql
-- 2. List distinct insurance types.
SELECT DISTINCT I.Type
FROM insurance I;
```

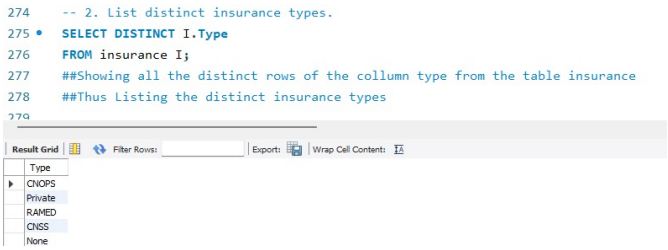


Figure 2: Query 12 Execution Results

Explanation: Shows all distinct values of the column `Type` from the `insurance` table.

Query 3

```sql
-- 3. Retrieve staff who work in hospitals located in Rabat
SELECT W.STAFF_ID
FROM Work_in W, Hospital H, Department D
WHERE D.DEP_ID = W.DEP_ID
  AND H.HID = D.HID
  AND H.City = 'Rabat';
```

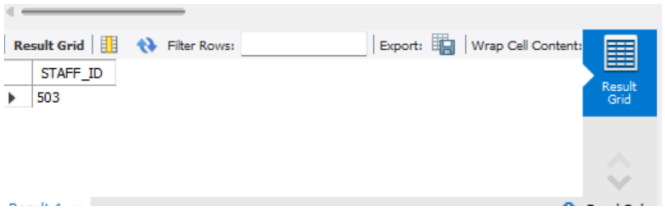


Figure 3: Output of Query 3

Explanation: Join `Work_in`, `Department`, and `Hospital`, then filter by city.

Query 4

```sql
-- 4. Find all appointments that are scheduled within the next seven days.
SELECT A.CAID, A.Reason, A.Status
FROM Appointment A
JOIN ClinicalActivity CA ON A.CAID = CA.CAID
WHERE CA.Date BETWEEN CURRENT_DATE()
    AND DATE_ADD(CURRENT_DATE(), INTERVAL 7 DAY);
```

	CAID	Reason	Status
▶	1003	Neurological assessment	Cancelled
	1004	Pediatric consultation	Scheduled

Figure 4: Query 4 Execution Results

Explanation: Filters appointments using a one-week date window.

Query 5

```sql
SELECT ca.DEP_ID, COUNT(ca.CAID) AS cnt
FROM ClinicalActivity ca
GROUP BY ca.DEP_ID;
```

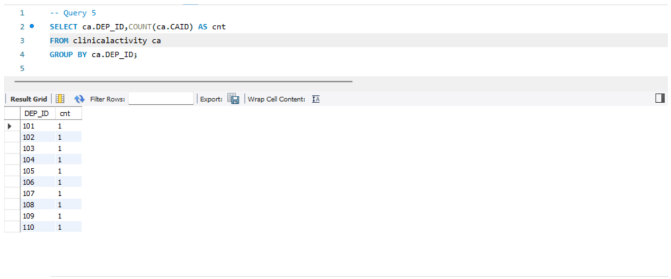


Figure 5: Output of Query 5

Explanation: Groups by department and counts appointments.

Query 6

```sql
SELECT  H.hid, H.name AS HospitalName, AVG(S.UnitPrice) AS Avg_UnitPrice
FROM Hospital AS H
LEFT JOIN Stock AS S ON H.hid = S.hid
LEFT JOIN Medication M ON M.mid = S.mid
GROUP BY H.hid, H.name;
```

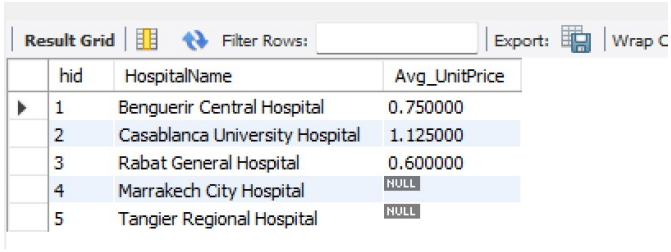


Figure 6: Output of Query 6

Explanation: Joins `Hospital`, `Stock`, and `Medication` to compute average prices.

Query 7

```sql
-- 7. List hospitals with more than twenty emergency admissions.
SELECT H.HID, H.Name, H.City, H.Region
FROM Hospital H
JOIN Department D ON H.HID = D.HID
JOIN ClinicalActivity CA ON CA.DEP_ID = D.DEP_ID
JOIN Emergency E ON CA.CAID = E.CAID
GROUP BY H.HID, H.Name, H.City, H.Region
HAVING COUNT(E.CAID) > 20;
```

	HID	Name	City	Region
▶	1	Benguerir Central Hospital	Benguerir	Marrakech-Safi

Figure 7: Query 7 Execution Results

Explanation: Counts emergency admissions and filters with `HAVING`.

Query 8

```sql
-- 8. Find medications in the therapeutic class Antibiotic where the unit price is l
SELECT m.M_NAME
FROM medication m, stock s
WHERE m.MID = s.MID
  AND m.TherapeuticClass='Antibiotic'
  AND s.UnitPrice < 200;
```

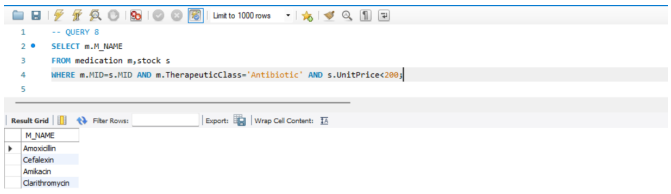


Figure 8: Output of Query 8

Explanation: Joins `medication` and `stock` and filters by class and price.

Query 9

```sql
-- 9. For each hospital list the top three most expensive medications
SELECT S.HID, S.MID, S.UnitPrice
FROM Stock S
WHERE (
    SELECT COUNT(DISTINCT S1.UnitPrice)
    FROM Stock S1
    WHERE S.HID = S1.HID AND S.UnitPrice < S1.UnitPrice
) < 3
ORDER BY S.HID, S.UnitPrice DESC;
```

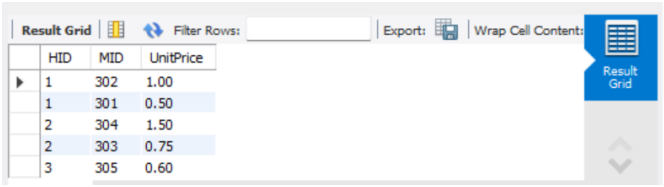
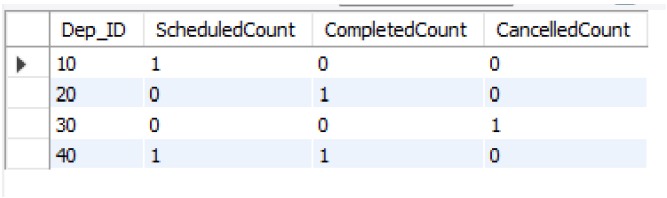


Figure 9: Output of Query 9

Explanation: Uses a correlated subquery to keep only the top 3 prices.

Query 10

```sql
-- 10. For each department return counts of scheduled completed and cancelled appoi
SELECT d.Dep_ID,
       SUM(a.status = 'Scheduled') AS ScheduledCount,
       SUM(a.status = 'Completed') AS CompletedCount,
       SUM(a.status = 'Cancelled') AS CancelledCount
FROM Department d, Appointment a, ClinicalActivity ca
WHERE d.Dep_ID = ca.DEP_ID
  AND ca.CAID = a.CAID
GROUP BY d.Dep_ID;
```



Dep_ID	ScheduledCount	CompletedCount	CancelledCount
10	1	0	0
20	0	1	0
30	0	0	1
40	1	1	0

Figure 10: Query 12 Execution Results

Explanation: Conditional aggregation counts each status per department using sum to ensure that the rows with a count = 0 are displayed.

Query 11

```sql
-- 11. List patients who have no scheduled appointments in the next thirty days.
SELECT DISTINCT P.IID, P.Fullname
FROM patient P
LEFT JOIN clinicalactivity C ON C.IID = P.IID
LEFT JOIN appointment A ON A.CAID = C.CAID
     AND A.Status = 'Scheduled'
```

```
    AND C.Date BETWEEN CURDATE()
        AND DATE_ADD(CURDATE(), INTERVAL 30 DAY)
WHERE A.CAID IS NULL;
```

	IID	Fullname
▶	1	Sara El Amrani
	2	Youssef Benali
	3	Laila Haddad
	4	Rachid El Fassi
	5	Nadia Choufani

Figure 11: Query 12 Execution Results

Explanation: Uses left join to keep patients lacking upcoming scheduled appointments.

Query 12

```
WITH HospitalAppointments AS (
    SELECT H.HID, COUNT(A.CAID) AS APP_CNT
    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
    GROUP BY H.HID
),
StaffAppointments AS (
    SELECT S.STAFF_ID, S.FullName, H.HID, COUNT(A.CAID) AS APP_CNT
    FROM Staff S
    JOIN ClinicalActivity CA ON CA.STAFF_ID = S.STAFF_ID
    JOIN Department D ON D.DEP_ID = CA.DEP_ID
    JOIN Hospital H ON H.HID = D.HID
    JOIN Appointment A ON A.CAID = CA.CAID
    GROUP BY S.STAFF_ID, S.FullName, H.HID
)
SELECT SA.STAFF_ID, SA.FullName, SA.HID, SA.APP_CNT AS total_appointments,
       ROUND(SA.APP_CNT * 100.0 / HA.APP_CNT, 2) AS percentage_share
FROM StaffAppointments SA
```

```
JOIN HospitalAppointments HA ON SA.HID = HA.HID
ORDER BY SA.HID, percentage_share DESC;
```

STAFF_ID	FullName	HID	Total_Appointments	Percentage_Share
501	Dr. Amina Idrissi	1	1	100.00
502	Technician Omar Lahlou	2	1	100.00
503	Nurse Fatima Zahra	3	1	100.00
504	Dr. Yassine El Khatib	4	2	100.00

Figure 12: Query 12 Execution Results

Explanation: Two CTEs compute totals per hospital and per staff.

Query 13

```
-- 13. Show all drugs that are below ReorderLevel in at least one hospital and incl
SELECT m.M_Name, h.Name
FROM stock s, medication m, hospital h
WHERE s.MID = m.MID
  AND s.HID = h.HID
  AND s.Qty < s.ReorderLevel;
```

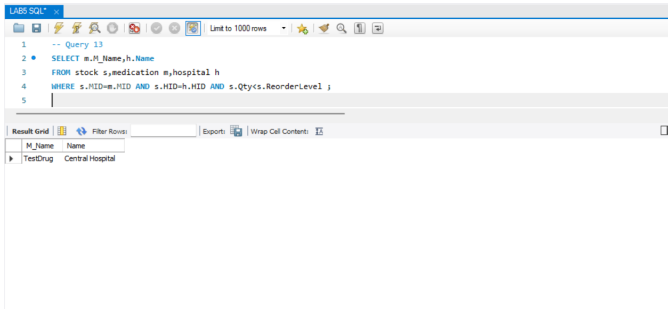


Figure 13: Output of Query 13

Explanation: Joins `stock`, `medication`, and `hospital` to filter by quantity.

Query 14

```
-- 14. Find hospitals that stock every antibiotic in the catalog.
SELECT H.HID, H.Name AS HospitalName
FROM Hospital H
WHERE NOT EXISTS (
```

```
    SELECT *
    FROM Medication M
    WHERE M.TherapeuticClass = 'Antibiotic'
      AND NOT EXISTS (
          SELECT *
          FROM Stock S
          WHERE S.HID = H.HID
            AND S.MID = M.MID
      )
);
```

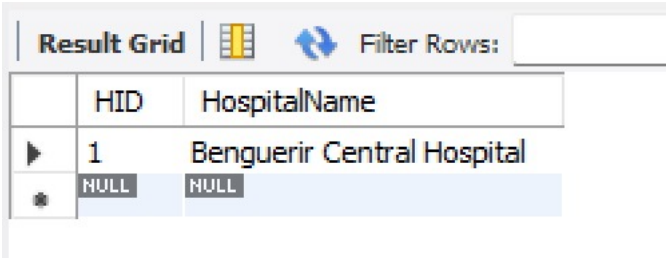


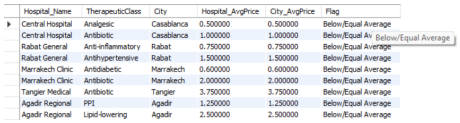
Figure 14: Output of Query 14

Explanation: Division using two nested NOT EXISTS.

Query 15

```
-- 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 Hospital_Name,
    m.TherapeuticClass,
    h.City,
    AVG(s.UnitPrice) AS Hospital_AvgPrice,
    city_avg.City_AvgPrice,
    CASE
        WHEN AVG(s.UnitPrice) > city_avg.City_AvgPrice THEN 'Above Average'
        ELSE 'Below/Equal Average'
    END AS Flag
FROM Stock s
JOIN Medication m ON s.MID = m.MID
JOIN Hospital h ON s.HID = h.HID
JOIN (
    SELECT h.City, m.TherapeuticClass, AVG(s.UnitPrice) AS City_AvgPrice
    FROM Stock s
    JOIN Medication m ON s.MID = m.MID
    JOIN Hospital h ON s.HID = h.HID
```

```
    GROUP BY h.City, m.TherapeuticClass
) AS city_avg
ON h.City = city_avg.City AND m.TherapeuticClass = city_avg.TherapeuticClass
GROUP BY h.HID, m.TherapeuticClass;
```



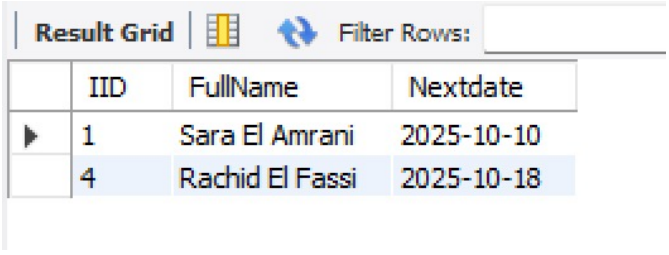
Hospital_Name	TherapeuticClass	City	Hospital_AvgPrice	City_AvgPrice	Flag
Central Hospital	Analgesic	Casablanca	0.500000	0.500000	Below/Equal Average
Central Hospital	Antibiotic	Casablanca	1.000000	1.000000	Below/Equal Average
Rabat General	Anti-inflammatory	Rabat	0.750000	0.750000	Below/Equal Average
Rabat General	Antihypertensive	Rabat	1.500000	1.500000	Below/Equal Average
Marrakech Clinic	Antidiabetic	Marrakech	0.600000	0.600000	Below/Equal Average
Marrakech Clinic	Antibiotic	Marrakech	2.000000	2.000000	Below/Equal Average
Tangier Medical	Antibiotic	Tangier	3.750000	3.750000	Below/Equal Average
Agadir Regional	PPI	Agadir	1.250000	1.250000	Below/Equal Average
Agadir Regional	Lipid-lowering	Agadir	2.500000	2.500000	Below/Equal Average

Figure 15: Output of Query 15

Explanation: Uses a subquery to compute city averages and compares.

Query 16

```
-- 16. Compute the average unit price of medications per hospital.
SELECT
    P.IID,
    P.FullName,
    MIN(CA.Date) AS Nextdate
FROM Patient P
JOIN ClinicalActivity CA ON P.IID = CA.IID
JOIN Appointment A ON CA.CAID = A.CAID
WHERE A.Status = 'Scheduled'
GROUP BY P.IID, P.FullName
ORDER BY P.IID;
```



Result Grid | Filter Rows:

IID	FullName	Nextdate
1	Sara El Amrani	2025-10-10
4	Rachid El Fassi	2025-10-18

Figure 16: Output of Query 16

Explanation: `MIN(Date)` yields each patient's earliest future appointment.

Query 17

```sql
-- 17. Among patients with at least two emergency visits list those
--     whose latest emergency visit was within the last fourteen days.
SELECT P.IID, P.FullName
FROM Patient P
JOIN ClinicalActivity CA ON CA.IID = P.IID
JOIN Emergency E ON E.CAID = CA.CAID
GROUP BY P.IID, P.FullName
HAVING COUNT(E.CAID) >= 2
   AND MAX(CA.Date) BETWEEN DATE_SUB(CURRENT_DATE(), INTERVAL 14 DAY)
   AND CURRENT_DATE();
```

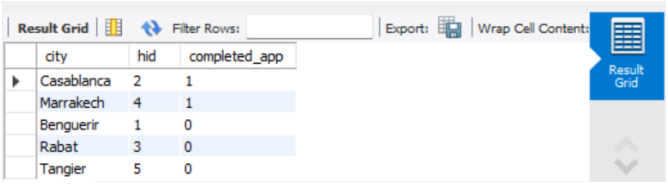
IID	FullName
3	Laila Haddad

Figure 17: Query 17 Execution Results

Explanation: Uses `HAVING` to filter by count and recentness.

Query 18

```sql
-- 18. For each city rank hospitals by completed appointments in last 90 days.
SELECT h.city, h.hid, COUNT(a.caid) AS completed_app
FROM Hospital h
LEFT JOIN Department d ON d.HID = h.HID
LEFT JOIN ClinicalActivity ca ON ca.DEP_ID = d.DEP_ID
LEFT JOIN Appointment a ON a.caid = ca.caid
  AND (ca.date >= (CURDATE() - INTERVAL 90 DAY) AND a.status='Completed')
GROUP BY h.HID, h.City
ORDER BY completed_app DESC, City;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city	hid	completed_app
Casablanca	2	1
Marrakech	4	1
Benguerir	1	0
Rabat	3	0
Tangier	5	0

Figure 18: Output of Query 18

Explanation: Counts completed appointments using left joins.

Query 19

```sql
-- 19. Medications whose prices vary by more than 30 percent within a city.
SELECT M.M_Name, H.city
FROM medication M
INNER JOIN stock S ON S.MID = M.MID
INNER JOIN hospital H ON H.HID = S.HID
GROUP BY H.city, M.M_Name
HAVING ((MAX(S.UnitPrice) - MIN(S.UnitPrice)) / MIN(S.UnitPrice)) > 0.3;
```

Figure 19: Output of Query 19

Explanation: Computes price spread using a HAVING condition.

Query 20

```sql
-- 20. Data quality check on stock entries list rows with negative quantity or non
SELECT *
FROM Stock S
WHERE S.Qty < 0 OR S.UnitPrice <= 0;
```

Figure 20: Output of Query 20

Explanation: Simple filter to identify invalid stock records.

4 Discussion

Challenges faced:
The more advanced queries (Q9, Q10, Q12, Q14, Q15, Q16, Q18, Q19) required using conditional aggregation, set logic, and subqueries. All of which demanded a clear

and structured way of thinking. Several questions involving dates (Q4, Q11, Q17) also required careful handling of time intervals.

Observations

The normalized schema made the logic easier to follow, especially when writing queries that relied on multiple relationships or grouped analysis. The DDL constraints helped keep the data consistent, which made the more complex operations more reliable.

Lessons Learned

Working on more advanced SQL requires thinking in terms of sets, relationships, and grouping. A clean schema and a clear logical approach make even the advanced queries much more manageable.

5 Conclusion

In this lab, we have verified each relation's BCNF status, lossless join and dependency preservation after decomposition. We also had valuable hands-on experience regarding SQL, be it the implementation of the database (creation of tables), the filling of said database and last, but certainly not the least, the implementation of a wide variety of queries using SQL.