Moroccan National Health Services (MNHS)

Data Management Course

Mohammed VI Polytechnic University (UM6P)
Professor: Karima Echihabi
Teaching Assistant: Hasnae Zerouaoui
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.

Introduction

This lab advances our MNHS database design by implementing complex queries using both Relational Algebra and SQL. We tackle 15 diverse healthcare scenarios ranging from basic patient-staff interactions to advanced analytical queries involving division and comparative analysis. The exercises strengthen our understanding of query optimization, functional dependencies, and the translation between theoretical algebra and practical SQL implementation. This comprehensive approach bridges database theory with real-world healthcare data management challenges.

Requirement

Part 1: Query Implementation

- Write database queries using both Relational Algebra and SQL code for each query.

Part 2:Refinement

- Identify and list functional dependencies (FDs) in the MNHS schema.

Query 1:

Relational Algebra

$$\pi_{\text{FullName}}\left(\textbf{Patient} \bowtie \textbf{ClinicalActivity} \bowtie \left(\sigma_{\text{Status='Active'}}(\textbf{Staff})\right)\right)$$

SQL Implementation

```
SELECT DISTINCT
    P.FullName
FROM
    Patient P
JOIN
    ClinicalActivity CA ON P.IID = CA.IID
JOIN
    Staff S ON CA.STAFF_ID = S.STAFF_ID
WHERE
    S.Status = 'Active ';
```

Query 2:

Relational Algebra

$$\pi_{\text{StaffID}}(\sigma_{\text{Status}='Active'}(\text{Staff})) \cup \pi_{\text{StaffID}}(\text{Staff} \bowtie \text{ClinicalActivity} \bowtie \text{Prescription\_generate})$$

SQL Implementation

```
SELECT StaffID
FROM Staff
WHERE Status = 'Active'
UNION
SELECT DISTINCT S.StaffID
FROM Staff S
JOIN ClinicalActivity CA ON S.StaffID = CA.StaffID
JOIN Prescription_generate P ON CA.CAID = P.CAID;
```

Query 3:Hospitals in Benguerir or Cardiology.

$$\text{Result} = \pi_{\text{HID}}\Big(\sigma_{\text{City}='Benguerir'}(\text{Hospital}) \cup \sigma_{\text{Specialty}='Cardiology'}(\text{Department})\Big)$$

SQL Implementation

```
SELECT HID FROM (
    SELECT HID FROM Hospital WHERE City = 'Benguerir'
    UNION
    SELECT HID FROM Department WHERE Specialty = 'Cardiology'
) AS Result;
```

Query 4

Find Hospital IDs of hospitals that have both 'Cardiology' and 'Pediatrics' departments.

Relational Algebra

$$HID(Specialty = Cardiology(Department)) \cap HID(Specialty = Pediatrics(Department))$$

SQL Implementation

Listing 1: We find the set of hospitals with Cardiology departments and intersect it with the set of hospitals with Pediatrics departments, and then we only keep hospitals present in both sets.

```
SELECT d.HID
FROM Department d
WHERE d.Specialty = 'Cardiology'
INTERSECT
SELECT d.HID
FROM Department d
WHERE d.Specialty = 'Pediatrics';
```

Query 5: Staff Members Who Worked in Every Department

Relational Algebra

$$\text{AllDepartments} \leftarrow \pi_{\text{DEP}} \left(\sigma \text{HID} = 1(\text{Department})\right)$$
$$\text{StaffDepartments} \leftarrow \pi_{\text{STAFF\_ID, DEP\_ID}}(\text{Work\_in})$$
$$\text{Result} \leftarrow \text{StaffDepartments}/\text{AllDepartments}$$

SQL Implementation

```
SELECT W.STAFF_ID
FROM Work_in W
WHERE NOT EXISTS (
    SELECT D.DEP_ID
    FROM Department D
    WHERE D.HID = 1
    AND D.DEP_ID NOT IN (
        SELECT W2.DEP_ID
        FROM Work_in W2
        WHERE W2.STAFF_ID = W.STAFF_ID
    )
);
```

Query 6

Find staff members who participated in every clinical activity of the department with `DEP_ID = 2`.

Relational Algebra

$$STAFF\_ID, CAID(DEP\_ID = 2(ClinicalActivity)) \ / \ CAID(DEP\_ID = 2(ClinicalActivity))$$

SQL Implementation

Listing 2: We count the total number of activities in department 2, then we find the staff members who have the same number of activities as the total count of activities in that department 2, which means that those staff members participated in all clinical activities.

```
SELECT s.STAFF_ID, s.FullName
FROM Staff s
WHERE s.STAFF_ID IN (
    SELECT c.STAFF_ID
    FROM ClinicalActivity c
    WHERE c.DEP_ID = 2
    GROUP BY c.STAFF_ID
```

```
    HAVING COUNT(*) = (
        SELECT COUNT(*)
        FROM ClinicalActivity
        WHERE DEP_ID = 2
    )
);
```

Query 7: Staff Member Pairs by Activity Count

Relational Algebra

$$\begin{aligned}
\text{StaffCounts} &\leftarrow \pi_{\text{STAFF_ID, COUNT(CAID)}\to\text{ActivityCount}}(\text{ClinicalActivity}) \\
S_1 &\leftarrow \rho_{\text{STAFF_ID}\to s_1,\text{ActivityCount}\to\text{count1}}(\text{StaffCounts}) \\
S_2 &\leftarrow \rho_{\text{STAFF_ID}\to s_2,\text{ActivityCount}\to\text{count2}}(\text{StaffCounts}) \\
\text{Result} &\leftarrow \pi_{s_1,s_2}\left(\sigma_{\text{count1}>\text{count2}}(S_1 \times S_2)\right)
\end{aligned}$$

SQL Implementation

```
SELECT S1.STAFF_ID AS s1, S2.STAFF_ID AS s2
FROM (SELECT STAFF_ID, COUNT(CAID) AS ActivityCount
      FROM ClinicalActivity
      GROUP BY STAFF_ID) S1,
     (SELECT STAFF_ID, COUNT(CAID) AS ActivityCount
      FROM ClinicalActivity
      GROUP BY STAFF_ID) S2
WHERE S1.ActivityCount > S2.ActivityCount;
```

Query 8: Patients with Multiple Staff Members

Relational Algebra

$$\begin{aligned}
\text{PatientStaffCounts} &\leftarrow \pi_{\text{IID, COUNT(DISTINCT STAFF_ID)}\to\text{UniqueStaffCount}}(\text{ClinicalActivity}) \\
\text{FilteredPatients} &\leftarrow \sigma_{\text{UniqueStaffCount}\geq 2}(\text{PatientStaffCounts}) \\
\text{Result} &\leftarrow \pi_{\text{IID}}(\text{FilteredPatients})
\end{aligned}$$

SQL Implementation

```
SELECT IID
FROM ClinicalActivity
GROUP BY IID
```

```sql
HAVING COUNT(DISTINCT STAFF_ID) >= 2;
```

Query 9:

Relational Algebra

$\pi_{\text{CAID}}(\sigma_{\text{Date}>='2025-09-01' \wedge \text{Date}<='2025-09-30' \wedge \text{City}='Benguerir'}(\text{ClinicalActivity} \bowtie \text{Department} \bowtie \text{Hospital}))$

SQL Implementation

```sql
SELECT CA.CAID
FROM ClinicalActivity CA
JOIN Department D ON CA.DeptID = D.DeptID
JOIN Hospital H ON D.HospitalID = H.HospitalID
WHERE CA.Date BETWEEN '2025-09-01' AND '2025-09-30'
  AND H.City = 'Benguerir';
```

Query 10:

Relational Algebra

$$\pi_{S.\texttt{STAFF\_ID}}\Big(\sigma_{P_1.\texttt{PID} \neq P_2.\texttt{PID}}\Big((\sigma_{S.\texttt{STAFF\_ID}=CA.\texttt{STAFF\_ID}}((\sigma_{CA.\texttt{CAID}=P_1.\texttt{CAID}}(\texttt{Prescription\_P1} \times \texttt{ClinicalActivity\_CA})) \times \texttt{Staff\_S})) \times (\sigma_{CA.\texttt{CAID}=P_2.\texttt{CAID}}(\texttt{Prescription\_P2}))\Big)\Big)$$

SQL Implementation

```sql
SELECT CA.StaffID
FROM ClinicalActivity CA
JOIN Prescription_generate P ON CA.CAID = P.CAID
GROUP BY CA.StaffID
HAVING COUNT(P.PID) > 1;
```

Query 11: Patients with multiple appointments.

$\text{R}_1 = \text{Appointment} \bowtie_{\text{Appointment.CAID} = \text{ClinicalActivity.CAID}} (\text{ClinicalActivity})$

$\text{R}_2 = \sigma_{\text{Status}='Scheduled'}(R_1)$

$\text{R}_3 = \pi_{\text{IID,DEPID}}(R_2)$

$\text{R}_4 = \rho_{(\text{IID\_C,DEPID\_C})}(R_3)$

$\text{R}_5 = R_3 \bowtie_{R_3.\text{IID}=R_4.\text{IID\_C} \wedge R_3.\text{DEPID} \neq R_4.\text{DEPID\_C}} (R_4)$

$\text{Result} = \pi_{\text{IID}}(R_5)$

SQL Implementation

```
SELECT DISTINCT R5.IID  -- Keep only unique patient IDs
FROM (
    -- join R3 with itself R4 to find patients with appointments in different
        departments
    SELECT R3.IID, R3.DEPID, R4.IID_C, R4.DEPID_C
    FROM (
        -- Subquery R3: scheduled appointments with patient ID and department ID
        SELECT Appointment.IID, ClinicalActivity.DEPID
        FROM Appointment
        JOIN ClinicalActivity
            ON Appointment.CAID = ClinicalActivity.CAID  -- Join to get department
                of each appointment
        WHERE Appointment.Status = 'Scheduled'  -- Keep only scheduled
            appointments
    ) AS R3
    JOIN (
        -- Subquery R4: create a copy of R3
        SELECT Appointment.IID AS IID_C, ClinicalActivity.DEPID AS DEPID_C
        FROM Appointment
        JOIN ClinicalActivity
            ON Appointment.CAID = ClinicalActivity.CAID
        WHERE Appointment.Status = 'Scheduled'
    ) AS R4
    ON R3.IID = R4.IID_C  -- Same patient
    AND R3.DEPID <> R4.DEPID_C  -- Different departments
) AS R5;
```

Query 12: Staff IDs who have no scheduled appointments on the day of the Green March holiday (November 6).

Relational Algebra

$$\pi_{\text{STAFF\_ID}}(\textbf{Staff}) - \pi_{\text{STAFF\_ID}}(\sigma_{\text{Date='2025-11-06'} \wedge \text{Status=' Scheduled '}}(\textbf{ClinicalActivity} \bowtie \textbf{Appointment}))$$

SQL Implementation

Listing 3: SQL for Finding Staff with No Scheduled Appointments

```
SELECT
    S.STAFF_ID
FROM
    Staff S
LEFT JOIN
```

```
    (
        SELECT CA.STAFF_ID
        FROM ClinicalActivity CA
        JOIN Appointment A ON CA.CAID = A.CAID
        WHERE CA.Date = '2025-11-06'
        AND A.Status = ' Scheduled '
    ) AS ScheduledStaff ON S.STAFF_ID = ScheduledStaff.STAFF_ID
WHERE
    ScheduledStaff.STAFF_ID IS NULL;
```

Query 13

Find departments whose average number of clinical activities is below the global departmental average.

Relational Algebra

$$\text{DeptCount} \leftarrow \gamma_{\text{DEP\_ID}};\text{value} \leftarrow \text{COUNT(CAID)(ClinicalActivity)}$$

$$\text{TotalActivities} \leftarrow \gamma;\text{total} \leftarrow \text{COUNT(CAID)(ClinicalActivity)}$$

$$\text{Departments} \leftarrow \gamma;\text{number} \leftarrow \text{COUNT(DISTINCT DEP\_ID)(ClinicalActivity)}$$

$$DEP\_ID(\text{value} < (\text{total/number})(\text{DeptCount} \times \text{TotalActivities} \times \text{Departments}))$$

SQL Implementation

Listing 4: We calculate the global average by dividing the total number of clinical activities by the number of departments that contains clinical activities, then we compare each department's activity count against this global average, if it's below it, we take it, otherwise no.

```
SELECT d.DEP_ID, d.Name
FROM Department d
WHERE (
    SELECT COUNT(*)
    FROM ClinicalActivity ca
    WHERE ca.DEP_ID = d.DEP_ID
) < (
    (SELECT COUNT(*) FROM ClinicalActivity) /
    (SELECT COUNT(DISTINCT DEP_ID) FROM ClinicalActivity)
);
```

Query 14: For each staff member, return the patient who has the greatest number of completed appointments with that staff member.

Relational Algebra

$$R_1 = Appointment \bowtie_{Appointment.CAID=ClinicalActivity.CAID} ClinicalActivity$$

$$R_2 = \sigma_{\text{Status}='Completed'}(R_1)$$

$$R_3 = \gamma_{STAFF\_ID,\, IID}\big(COUNT(CAID) \rightarrow n\big)(R_2)$$

$$R_4 = \gamma_{STAFF\_ID}\big(MAX(n) \rightarrow maxN\big)(R_3)$$

$$Result = \pi_{STAFF,\, IID}\Big(R_3 \bowtie R_3.STAFF\_ID = R_4.STAFF\_ID \land R_3.n = R_4.maxN\ R_4\Big)$$

SQL Implementation

Listing 5: SQL for Finding Patient with Max Completed Appointments per Staff Member

```
WITH RankedAppointments AS (

SELECT
CA.STAFF_ID,
CA.IID,
COUNT(CA.CAID) AS ApptCount,

ROW_NUMBER() OVER (
PARTITION BY CA.STAFF_ID
ORDER BY COUNT(CA.CAID) DESC
) AS rn
FROM
ClinicalActivity CA
JOIN
Appointment A ON CA.CAID = A.CAID
WHERE
A.Status = ' Completed '
GROUP BY
CA.STAFF_ID,
CA.IID
)
SELECT
S.FullName AS StaffName,
P.FullName AS PatientName,
```

```
RA.ApptCount
FROM
RankedAppointments RA
JOIN
Staff S ON RA.STAFF_ID = S.STAFF_ID
JOIN
Patient P ON RA.IID = P.IID
WHERE

RA.rn = 1
ORDER BY
StaffName;
```

Query 15: Patients with 3 emergencies in 2024

$$R_1 = Emergency \bowtie_{Emergency.CAID=ClinicalActivity.CAID} ClinicalActivity$$

$$R_2 = \sigma_{\text{Outcome}='Admitted'}(R_1)$$

$$R_3 = \sigma_{\text{Date}\geq'2024-01-01' \wedge \text{Date}\leq'2024-12-31'}(R_2)$$

$$R_4 = \pi_{IID,\ CAID}(R_3)$$

$$R_5 = \gamma_{IID}(COUNT(CAID) \rightarrow n)(R_4)$$

$$Result = \sigma_{n\geq 3}(R_5)$$

SQL Implementation

```
-- Select patients (IID) with 3 or more admitted emergency appointments in 2024
SELECT IID
FROM (
    SELECT R4.IID, COUNT(R4.CAID) AS n
    FROM (

        SELECT E.IID, E.CAID
        FROM Emergency E
        JOIN ClinicalActivity C -- Join to get details of each emergency
            ON E.CAID = C.CAID
        WHERE C.Outcome = 'Admitted'
          AND C.Date >= '2024-01-01'
          AND C.Date <= '2024-12-31'
    ) AS R4
    GROUP BY R4.IID   -- Count appointments per patient
) AS R5
WHERE n >= 3;    -- Keep only patients with 3 or more appointments
```

Methodology

When we designed our solution for this lab, we tried to explain or to write our queries in relational algebra, and then translate it into a correct and comprehensible SQL code. We treated each algebra operation as a precise instruction, and we made sure to write a clean code without errors while, of course, respecting what was in our algebra equations. This often meant breaking complex problems into smaller, manageable steps, and trying to build the solution step by step. For the design choices, we tried to make our code clear and simple to read and understand, that is the reason why we opted for the following methods:

Query 4 and 6 : We created JOIN patterns across patient, clinical activity, and medication tables, using WHERE conditions to filter by specific medication types and implement set operations through UNION for combined result sets to find the departments fulfilling our condition.

Query 5 : We used relational division implemented via NOT EXISTS to find staff members who worked in all departments of a specific hospital, where we created a nested SELECT to verify no department was missing from their work history.

Query 1: We performed a natural join between Patient, ClinicalActivity, and filtered Staff entities. This ensured we only retrieved patients handled by currently active medical staff.

Query 7: We created two instances of the same query (S1 and S2) that calculates activity counts per staff member. Then, We applied a filter condition: $S1.ActivityCount > S2.ActivityCount$ to select pairs where the first staff member handled more activities than the second one.

Query 8: We counted distinct staff members per patient, filtering for patients with at least two different staff members through clinical activities. Query 2: We performed a selection on ClinicalActivity for dates within September 2025, then joined with Department and Hospital entities, and the we followed it by a selection for the city of Benguerir.

Query 9:We created a join chain from Patient through ClinicalActivity and Prescription to Medication, with selection for specific medication types and projection to patient identifiers.

Query 10: We performed a semi-join between Staff and Emergency entities through ClinicalActivity, selecting for staff involved in emergency cases.

Query 11: We employed grouping and aggregation to calculate activity counts per department, then we used comparative selection against the global departmental average.

Query 12:We employed set difference between all staff members and those with scheduled appointments on a specific date.

Query 15:We performed joins between Medication, Stock, and Hospital entities with aggregation to calculate inventory metrics.

Query 13:We created a main query that selects department names and IDs from the Department table. And then, after that, We applied a WHERE condition comparing department activity count against the average (total activities / number of departments)

Query 3 : We applied a WHERE clause filter on ContactLocation.City to target specific geographic locations and used DISTINCT to ensure each patient appears only once even if they have multiple addresses in the same city

Query 14: We used GROUP BY on insurance type to categorize coverage patterns and then applied COUNT() on patient IDs to calculate the number of patients per insurance type, and finally, we added HAVING clause to filter for insurance types.

In the resolution of this lab, we did make some assumptions that helped us go through this project, for example: Staff-Department Relationship: We assumed a many-to-many relationship between staff and departments via the $Work_intable$.

Clinical Activity Structure: We assumed all clinical activities link to specific patients and staff.

Patient-Staff Interactions: We assumed patients can have multiple distinct staff members through clinical activities.

Hospital: We assumed departments belong to specific hospitals.

Refinement

PATIENT

- **FD1**: IID → CIN, FullName, Birth, Sex, BloodGroup, Phone
  Justification: The primary key IID determines all the attributes of the entity.
- **FD2**: CIN → IID, FullName, Birth, Sex, BloodGroup, Phone
  Justification: The uniqueness of the attribute CIN determines all the other fields.

Hospital

- **FD3**: HID → Name, City, Region
  Justification: HID is a primary key.
- **FD4**: City → Region
  Justification: Knowing the city determines the region.

Department

- **FD5**: DEP_ID → HID, Name, Specialty
  Justification: Primary key.
- **FD6**: DEP_ID → Hospital.Name, Hospital.City, Hospital.Region
  Justification: The DEP_ID uniquely determines the name, city, and region of the hospital linked using the foreign key.

STAFF

- **FD7**: STAFF_ID → FullName, Status
  Justification: Primary key.

Caregiving

This is an entity in the ISA Hierarchy; it inherits the functional dependencies of the superclass Staff. Additionally:

- **FD8**: Staff_ID → Staff_Name, Staff_Status
  Justification: Primary key.

Technical

This is an entity in the ISA Hierarchy; it inherits the functional dependencies of Staff. Additionally:

- **FD9**: Staff_ID → Modality, Certifications
  Justification: Primary key.

Practitioner

This is an entity in the ISA Hierarchy; it inherits the functional dependencies of Staff. Additionally:

- **FD10**: Staff_ID → licenseNumber, Specialty
  Justification: Primary key.

Work_In

This table does not have proper functional dependencies, but it inherits dependencies from the linked entities via foreign keys.

ClinicalActivity

- **FD11**: CAID → IID, STAFF_ID, DEP_ID, Date, Time
  Justification: Primary key.
- **FD12**: CAID → Patient.CIN, Patient.FullName, Patient.Birth, Patient.Sex, Patient.BloodGroup, Patient.Phone
  Justification: The primary key determines the fields of the Patient entity referenced by IID.
- **FD13**: CAID → Staff.FullName, Staff.Status
  Justification: The primary key determines the fields of Staff referenced by STAFF_ID.
- **FD14**: CAID → Department.HID, Department.Name, Department.Specialty
  Justification: The primary key determines the fields of Department referenced by DEP_ID.
- **FD15**: CAID → Hospital.Name, Hospital.City, Hospital.Region
  Justification: The primary key determines the fields of Hospital referenced indirectly via DEP_ID.

Appointment

This is an ISA entity of ClinicalActivity; it inherits the superclass dependencies. Additionally:

- **FD16**: CAID → Reason, Status
  Justification: Primary key.

Emergency

This is an ISA entity of ClinicalActivity; it inherits the superclass dependencies. Additionally:

- **FD17**: CAID → TriageLevel, Outcome
  Justification: Primary key.

Insurance

- **FD18**: InsID → Type
  Justification: Primary key.

Expense

- **FD19**: ExpId → InsID, CAID, Total
  Justification: Primary key.
  Note: All dependents of CAID are also dependent on ExpId; the same applies to dependents of InsID.

Medication

- **FD20**: MID → Name, Form, Strength, ActiveIngredient, TherapeuticClass, Manufacturer
  Justification: Primary key.

Stock

- **FD21**: (HID, MID, StockTimestamp) → UnitPrice, Qty
  Justification: Primary key.
  Note: All dependents of HID and MID are also dependent on (HID, MID, StockTimestamp).

Prescription

- **FD22**: PID → CAID, DateIssued
  Justification: Primary key.
  Note: All dependents of CAID are also dependents of PID.

INCLUDES

- **FD23**: (MID, PID) → Dosage, Duration
  Justification: Primary key.
  Note: All dependents of MID and PID are also dependent on (MID, PID).

ContactLocation

- **FD24**: CLID → City, Province, Street, Number, PostalCode, Phone_Location
  Justification: Primary key.
- **FD25**: City → Province
  Justification: City uniquely determines Province.
- **FD26**: PostalCode → City
  Justification: PostalCode uniquely determines City.

Have

This table does not have proper functional dependencies, but inherits dependencies from linked entities via foreign keys.

Discussion

Problems faced :

- Translating queries from sentences to relational algebra was challenging.
- Agree on a solution as team members .

Lessons learned :

- This time, we divided the work in a better way than the last lab, we were more efficient
- Asking help from other members is quite helpful when you are really stuck
- Don't leave the work until last minute even if you have tons of exams

Conclusion

This lab advances our **MNHS database design** by implementing complex queries using both Relational Algebra and **SQL**. We successfully tackled diverse healthcare scenarios, focusing on the rigorous application of query primitives, and gaining practical experience with concepts like query optimization and functional dependencies.