University
Mohammed VI
Polytechnic

College of Computing

Deliverable 3: Relational Algebra, SQL and Functional Dependencies

Data Management Course

UM6P College of Computing

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

Team Information

Team Name	alfari9 alkhari9
Member 1	Ilyas Rahmouni
Member 2	Malak Koulat
Member 3	Aymane Raiss
Member 4	Zakaria Harira
Member 5	Youness Latif
Member 6	Rayane Khaldi
Member 7	Younes Lougnidi
Repository Link	`https://github.com/...`

---

1 Part 1: Relational Algebra and SQL

1. Find the names of patients who have had at least one clinical activity handled by active staff.

Relational Algebra:

$$\pi_{\text{FullName}}\Big(\sigma_{\text{Status='Active'}}(\text{Staff}) \bowtie_{\text{Staff.STAFF\_ID=ClinicalActivity.STAFF\_ID}} \text{ClinicalActivity} \bowtie_{\text{ClinicalActivity.IID=Patient.IID}} \text{Patient}\Big)$$

SQL:

```
SELECT 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';
```

Listing 1: Query for patients treated by active staff.

2. Find Staff IDs of staff who are either 'Active' or have issued at least one prescription.

Relational Algebra:

$$(\pi_{\text{STAFF\_ID}}(\sigma_{\text{Status='Active'}}(\text{Staff}))) \cup (\pi_{\text{STAFF\_ID}}(\text{ClinicalActivity} \bowtie_{\text{CAID}} \text{Prescription}))$$

SQL:

```
SELECT STAFF_ID FROM Staff WHERE Status = 'Active'
UNION
SELECT CA.STAFF_ID
FROM ClinicalActivity CA
JOIN Prescription P ON CA.CAID = P.CAID;
```

Listing 2: Query for active staff or staff who issued prescriptions.

3. Find Hospital IDs of hospitals located in 'Benguerir' or having at least one department with the specialty 'Cardiology'.

Relational Algebra:

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

SQL:

```sql
SELECT HID FROM Hospital WHERE City = 'Benguerir'
UNION
SELECT HID FROM Department WHERE Specialty = 'Cardiology';
```

Listing 3: Query for hospitals in Benguerir or with a Cardiology department.

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

Relational Algebra:

$$(\pi_{\text{HID}}(\sigma_{\text{Specialty='Cardiology'}}(\text{Department}))) \cap (\pi_{\text{HID}}(\sigma_{\text{Specialty='Pediatrics'}}(\text{Department})))$$

SQL:

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

Listing 4: Query for hospitals with both Cardiology and Pediatrics.

5. Find staff members who have worked in every department of the hospital with HID = 1.

Relational Algebra:

$$(\pi_{\text{STAFF\_ID, Dep\_ID}}(\text{Work\_in})) \,/\, (\pi_{\text{DEP\_ID}}(\sigma_{\text{HID}=1}(\text{Department})))$$

SQL:

```sql
SELECT W.STAFF_ID
FROM Work_in W
WHERE W.Dep_ID IN (SELECT DEP_ID FROM Department WHERE HID = 1)
GROUP BY W.STAFF_ID
HAVING COUNT(DISTINCT W.Dep_ID) = (
    SELECT COUNT(*) FROM Department WHERE HID = 1
);
```

Listing 5: Query for staff who worked in all departments of Hospital 1.

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

Relational Algebra:

$$(\pi_{\text{STAFF\_ID, CAID}}(\text{ClinicalActivity})) \,/\, (\pi_{\text{CAID}}(\sigma_{\text{DEP\_ID}=2}(\text{ClinicalActivity})))$$

SQL:

```sql
SELECT STAFF_ID
FROM ClinicalActivity
WHERE DEP_ID = 2
GROUP BY STAFF_ID
HAVING COUNT(DISTINCT CAID) = (
    SELECT COUNT(*) FROM ClinicalActivity WHERE DEP_ID = 2
);
```

Listing 6: Query for staff who participated in all activities of Department 2.

7. Find pairs of staff members (s1, s2) such that s1 has handled more clinical activities than s2.

Relational Algebra:

Let StaffCounts $\leftarrow \gamma_{\text{STAFF\_ID; COUNT(CAID)}\rightarrow\text{ActivityCount}}(\text{ClinicalActivity})$

$$\pi_{\text{s1.STAFF\_ID, s2.STAFF\_ID}}\left((\rho_{\text{s1}}(\text{StaffCounts})) \bowtie_{\text{s1.ActivityCount}>\text{s2.ActivityCount}} (\rho_{\text{s2}}(\text{StaffCounts}))\right)$$

SQL:

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

Listing 7: Query for pairs of staff based on activity count.

8. Find Patient IDs of patients who had clinical activities with at least two different staff members.

Relational Algebra:

$$\pi_{\text{IID}}\left(\sigma_{\text{StaffCount}\geq 2}\left(\gamma_{\text{IID; COUNT(DISTINCT STAFF\_ID)}\rightarrow\text{StaffCount}}(\text{ClinicalActivity})\right)\right)$$

SQL:

```sql
SELECT IID
FROM ClinicalActivity
GROUP BY IID
HAVING COUNT(DISTINCT STAFF_ID) >= 2;
```

Listing 8: Query for patients seen by two or more different staff.

9. Find CAIDs of clinical activities performed in September 2025 at hospitals located in "Benguerir".

Relational Algebra:

$$\pi_{\text{CAID}}\Big(\sigma_{\substack{\text{City='Benguerir'}\\ \wedge\text{Date}\geq\text{'2025-09-01'}\\ \wedge\text{Date}\leq\text{'2025-09-30'}}}\Big(\text{ClinicalActivity} \bowtie_{\text{ClinicalActivity.DEP\_ID=Department.DEP\_ID}} \text{Department} \bowtie_{\text{Department.HID=Hospital.HID}} \text{Hospital}\Big)\Big)$$

SQL:

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

Listing 9: Query for activities in Benguerir during September 2025.

10. Find Staff IDs of staff who have issued more than one prescription.

Relational Algebra:

$$\pi_{\text{STAFF\_ID}}\Big(\sigma_{\text{PrescriptionCount}>1}\big(\gamma_{\text{STAFF\_ID; COUNT(DISTINCT PID)}\rightarrow\text{PrescriptionCount}}(\text{ClinicalActivity} \bowtie_{\text{ClinicalActivity.CAID=Prescription.CAID}} \text{Prescription})\big)\Big)$$

SQL:

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

Listing 10: Query for staff who have issued more than one prescription.

11. List IIDs of patients who have scheduled appointments in more than one department.

Relational Algebra:

Let Scheduled $\leftarrow \sigma_{\text{Status='Scheduled'}}(\text{Appointment})$

$$\pi_{\text{IID}}\Big(\sigma_{\text{DeptCount}>1}\big(\gamma_{\text{IID; COUNT(DISTINCT DEP\_ID)}\rightarrow\text{DeptCount}}(\text{Scheduled} \bowtie_{\text{Scheduled.CAID=ClinicalActivity.CAID}} \text{ClinicalActivity})\big)\Big)$$

SQL:

```
SELECT CA.IID
FROM ClinicalActivity CA
JOIN Appointment A ON CA.CAID = A.CAID
WHERE A.Status = 'Scheduled'
GROUP BY CA.IID
HAVING COUNT(DISTINCT CA.DEP_ID) > 1;
```

Listing 11: Query for patients with appointments in multiple departments.

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

Relational Algebra:

$$\Big(\pi_{\text{STAFF\_ID}}(\text{Staff})\Big) - \Big(\pi_{\text{STAFF\_ID}}\big(\sigma_{\substack{\text{Status='Scheduled'} \\ \wedge\text{EXTRACT(MONTH FROM Date)=11} \\ \wedge\text{EXTRACT(DAY FROM Date)=6}}}(\text{ClinicalActivity} \bowtie_{\text{ClinicalActivity.CAID=Appointment.CAID}} \text{Appointment})\big)\Big)$$

SQL:

```
SELECT STAFF_ID FROM Staff
EXCEPT
SELECT CA.STAFF_ID
FROM ClinicalActivity CA
JOIN Appointment A ON CA.CAID = A.CAID
WHERE A.Status = 'Scheduled'
  AND EXTRACT(MONTH FROM CA.Date) = 11
  AND EXTRACT(DAY FROM CA.Date) = 6;
```

Listing 12: Query for staff with no appointments on November 6.

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

Relational Algebra:

Let DeptCounts $\leftarrow \gamma_{\text{DEP\_ID; COUNT(CAID)}\rightarrow\text{NumActivities}}(\text{ClinicalActivity})$

Let GlobalAvg $\leftarrow \gamma_{\text{AVG(NumActivities)} \rightarrow \text{AvgVal}}(\text{DeptCounts})$

$$\pi_{\text{DEP\_ID}}(\text{DeptCounts} \bowtie_{\text{DeptCounts.NumActivities} < \text{GlobalAvg.AvgVal}} \text{GlobalAvg})$$

SQL:

```
SELECT DEP_ID
FROM ClinicalActivity
GROUP BY DEP_ID
HAVING COUNT(CAID) < (
    SELECT AVG(ActivityCount)
    FROM (
        SELECT COUNT(CAID) AS ActivityCount
        FROM ClinicalActivity
        GROUP BY DEP_ID
    ) AS DepartmentActivityCounts
);
```

Listing 13: Query for departments with below-average activity count.

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

Relational Algebra:

.....

SQL:

```
SELECT spc.STAFF_ID, spc.IID
FROM (
    SELECT CA.STAFF_ID, CA.IID, COUNT(*) AS ApptCount
    FROM ClinicalActivity CA
    JOIN Appointment A ON CA.CAID = A.CAID
    WHERE A.Status = 'Completed'
    GROUP BY CA.STAFF_ID, CA.IID
) AS spc
WHERE spc.ApptCount = (
    SELECT MAX(ApptCount)
    FROM (
        SELECT CA.STAFF_ID, CA.IID, COUNT(*) AS ApptCount
        FROM ClinicalActivity CA
        JOIN Appointment A ON CA.CAID = A.CAID
        WHERE A.Status = 'Completed'
        GROUP BY CA.STAFF_ID, CA.IID
    ) AS inner_spc
    WHERE inner_spc.STAFF_ID = spc.STAFF_ID
);
```

Listing 14: Query to find the top patient for each staff member.

15. List patients who had at least 3 emergency admissions during the year 2024.

Relational Algebra:

$$\pi_{\text{IID}}\Big(\sigma_{\text{EmergencyCount}\geq 3}\big(\gamma_{\text{IID; COUNT(CAID)}\rightarrow\text{EmergencyCount}}\big(\sigma_{\text{EXTRACT(YEAR FROM Date)='2024'}}(\text{ClinicalActivity} \bowtie_{\text{ClinicalActivity.CAID=Emergency.CAID}} \text{Emergency})\big)\big)\Big)$$

SQL:

```sql
SELECT CA.IID
FROM ClinicalActivity CA
JOIN Emergency E ON CA.CAID = E.CAID
WHERE EXTRACT(YEAR FROM CA.Date) = 2024
GROUP BY CA.IID
HAVING COUNT(E.CAID) >= 3;
```

Listing 15: Query for patients with 3+ emergency admissions in 2024.

2 Part 2: Refinement (Functional Dependencies)

Summary of Functional Dependencies by Relation

1. **Patient**
   - IID → {CIN, FullName, Birth, Sex, BloodGroup, Phone}
   - CIN → {IID, FullName, Birth, Sex, BloodGroup, Phone}
   - *Justification:* IID is the primary key; CIN is a candidate key.

2. **Hospital**
   - HID → {Name, City, Region}
   - *Justification:* HID is the primary key.

3. **Department**
   - DEP_ID → {HID, Name, Specialty}
   - {HID, Name} → {DEP_ID, Specialty}
   - *Justification:* Unique department ID globally, and unique department name within a hospital.

4. **Staff**
   - STAFF_ID → {FullName, Status}
   - *Justification:* STAFF_ID is the primary key.

5. **Work_in**
   - {STAFF_ID, Dep_ID} → {}
   - *Justification:* Pure many-to-many relationship; no non-key attributes.

6. **ClinicalActivity**
   - CAID → {IID, STAFF_ID, DEP_ID, Date, Time}
   - *Justification:* CAID is the primary key.

7. **Appointment**
   - CAID → {Reason, Status}
   - *Justification:* CAID is both the primary and foreign key to ClinicalActivity.

8. **Emergency**
   - CAID → {TriageLevel, Outcome}
   - *Justification:* CAID is both the primary and foreign key to ClinicalActivity.

9. **Insurance**
   - InsID → {Type}
   - *Justification:* InsID is the primary key.

10. **Expense**
    - ExpID → {InsID, CAID, Total}
    - CAID → {ExpID, InsID, Total}
    - *Justification:* ExpID is the primary key; CAID is unique, establishing a 1-to-1 relationship with ClinicalActivity.

11. **Medication**
    - MID → {Name, Form, Strength, ActiveIngredient, TherapeuticClass, Manufacturer}
    - *Justification:* MID is the primary key.

12. **Stock**
    - {HID, MID, StockTimestamp} → {UnitPrice, Qty, ReorderLevel}
    - *Justification:* Composite key (HID, MID, StockTimestamp) represents stock history per hospital-medication pair.

13. **Prescription**
    - PID → {CAID, DateIssued}
    - CAID → {PID, DateIssued}
    - *Justification:* PID is the primary key; CAID is unique.

14. **Includes**
    - {PID, MID} → {Dosage, Duration}
    - *Justification:* Many-to-many relationship between Prescription and Medication.

15. **ContactLocation**
    - CLID → {City, Province, Street, Number, PostalCode, Phone_Location}
    - Additional Transitive Dependencies:
      - {City, Province, Street} → PostalCode
      - PostalCode → {City, Province}
      - {Number, Street, City, Province} → Phone_Location
    - *Justification:* CLID is the primary key; transitive dependencies reveal normalization issues.

16. **Have**

- {IID, CLID} → {}
- *Justification:* Many-to-many relationship between Patient and ContactLocation.

17. **Covers**

- {InsID, IID} → {}
- *Justification:* This is a pure many-to-many relationship between Insurance and Patient. The combination (InsID, IID) is the primary key with no non-key attributes.