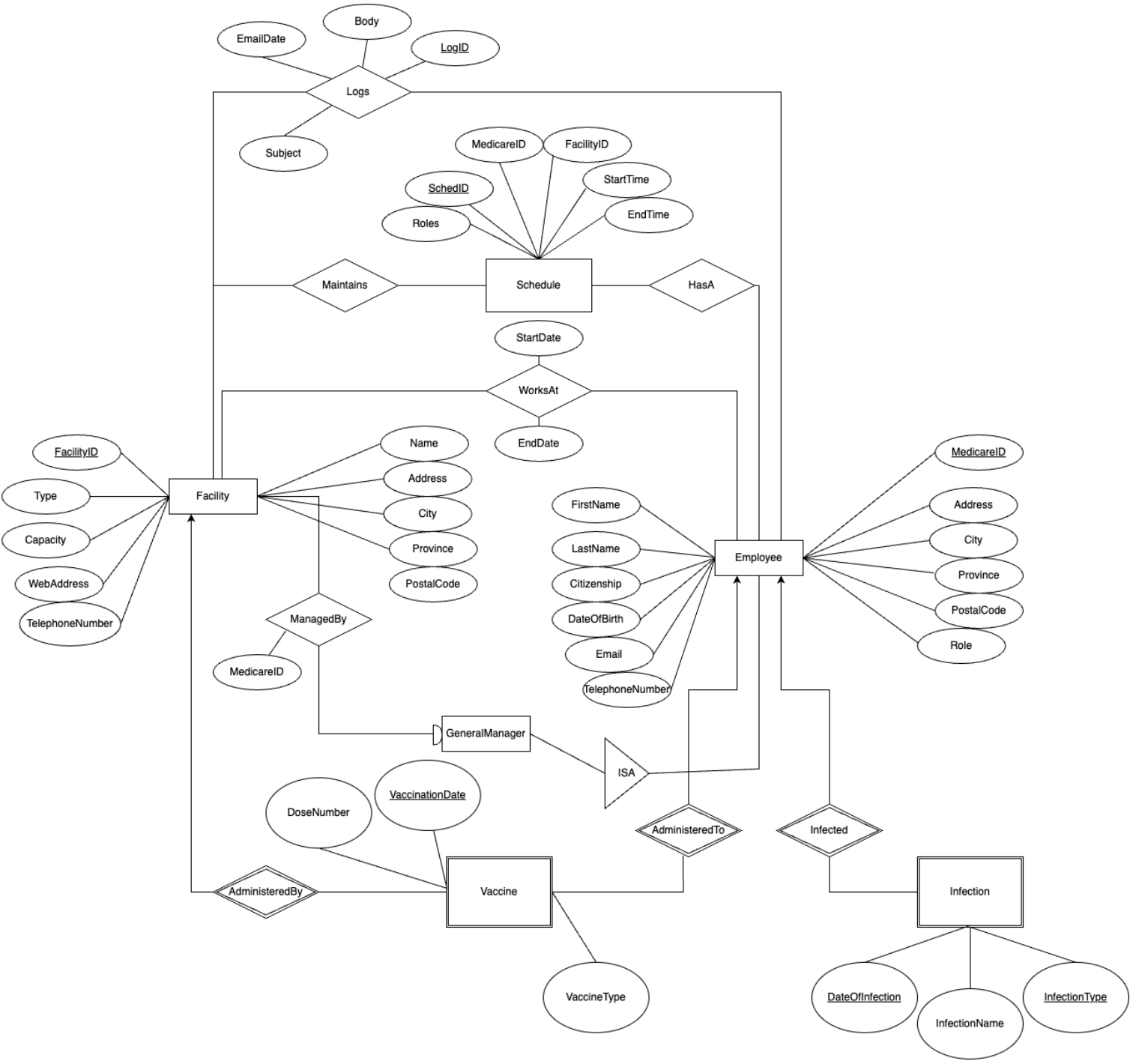
Date: 2023/ 04/ 11

Part 1: E/R Diagram

1.1. E/R Diagram:



1.2. E/R Diagram Details and Database Schema

Strong Entities:

- Facility(FacilityID, Name, Type, Capacity, Address, City, Province, PostalCode, WebAddress, TelephoneNumber)
    - Primary key: FacilityID
    - No Foreign key.
    - FD = {
        - FacilityID → Name, Type, Capacity, Address, City, Province, PostalCode, WebAddress, TelephoneNumber; Name, Type, Address, City, Province
        - PostalCode → FacilityID, Address, City, Province, WebAddress, TelephoneNumber }

- GeneralManager(MedicareID)
    - Primary key: MedicareID
    - Foreign key: MedicareID references Employee.MedicareID
    - Referential constraints: when a new MedicareID is added/ updated in Employee for a manager, the change must be cascaded to GeneralManager; when a MedicareID is deleted in Employee, the corresponding GeneralManage tuple (if it exists) will be deleted.

- Employee(MedicareID, FirstName, LastName, Role, Citizenship, DateOfBirth, Email, Address, City, Province, PostalCode, TelephoneNumber)
    - Primary key: MedicareID
    - No Foreign key.
    - FD = {
        - MedicareID → FirstName, LastName, Role, Citizenship, DateOfBirth, Email, Address, City, Province, PostalCode, TelephoneNumber; FirstName, LastName}

- Schedule(SchedID, MedicareID, FacilityID, StartTime, EndTime, Roles)
    - Primary key: SchedID
    - Foreign keys: MedicareID references Employee.MedicareID, FacilityID references Facility.FacilityID
    - Referential constraints: when an Employee's MedicareID is updated in Employee or when a Facility's FacilityID is updated in Facility, the change must be cascaded to Schedule; when a MedicareID is deleted in Employee or when a FacilityID is deleted in Facility, the corresponding MedicareID/ FacilityID will be deleted as well in Schedule.

    - FD = {

    - SchedID → Role, StartDate, EndDate, StartTime, EndTime, MedicareID, FacilityID
    - MedicareID, FacilityID, StartDate, StartTime → EndDate, EndTime, Role, SchedID
      }

All the above FDs are in 3NF because the LHS are superkeys.

- For Facility, FacilityID determines all the other attributes plus itself (trivial). This is because the FacilityID is the unique identification number of each facility.
  Name, Type, Address, City, Province and PostalCode together can determine the other attributes as well, plus themselves. This is because 2 facilities can have the same address or the same name but the type will distinguish them. However, the Type and Name or Type and Address (including City, Province and PostalCode) alone cannot determine the others.There can exist 2 different facilities with the same name and type; or the same type and at the same address (shared building).
- For GeneralManager, the only FD is { MedicareID → MedicareID }, which is trivial and is not mentioned.
- For Employee, MedicareID determines all the other attributes plus itself (trivial). This is because the MedicareID is the unique identification number of each employee.
  FirstName, LastName, DateOfBirth, Citizenship together can determine the other attributes as well, plus themselves. Two employees can have the same first name and last name but their email will differ because an email address is a unique identifier that is used to send and receive electronic messages. However, Email alone might not be sufficient to determine the other attributes in case the employee does not have an email address.
- For Schedule, SchedID determines all the other attributes plus itself (trivial). This is because the SchedID is the unique identification number of each schedule.
  MedicareID, Date, StartTime can determine the other attributes. This is because an employee can be scheduled on the same date, at the same facility but cannot be at the same time due to our constraints. Additionally, the second FD (MedicareID, FacilityID, StartDate, StartTime) are also a super key and determine the entire schema plus themselves.

Strong Relationships:

- Maintains(FacilityID, SchedID)
- HasA(MedicareID, SchedID)
- ManagedBy(FacilityID, MedicareID)
- WorksAt(MedicareID, FacilityID, StartDate, EndDate)
- Logs(LogID, EmailDate, Subject, Body, MedicareID, FacilityID)

Weak Entities:

- Vaccine(FacilityID, MedicareID, VaccinationDate, VaccineType, DoseNumber)
    - Primary keys: FacilityID, MedicareID, VaccineType, VaccinationDate
    - Foreign keys: FacilityID, MedicareID
    - Referential constraints: when a MedicareID is updated in Employee or when a FacilityID is updated in Facility, the change must be cascaded to Vaccine; when a MedicareID is deleted along with its tuple in Employee or when a FacilityID is deleted along with its tuple in Facility, the corresponding tuple(s) in Vaccine will also be deleted.
    - FD = { FacilityID, MedicareID, VaccineType, VaccinationDate → DoseNumber }
- Infection(MedicareID, DateOfInfection, InfectionType, InfectionName)
    - Primary keys: MedicareID, DateOfInfection
    - Foreign key: MedicareID
    - Referential constraint: when a new MedicareID is updated in Employee, the change must be cascaded to Infection; when a MedicareID along with its tuple(s) is deleted in Employee, the corresponding tuple in Infection will also be deleted.
    - FD = { MedicareID, DateOfInfection, and InfectionType → InfectionName }

All the above FDs are in 3NF because the LHS are superkeys.

- For Vaccine, FacilityID, MedicareID, VaccinationDate together determine DoseNumber, VaccinationType and themselves (trivial). This is because the FacilityID and MedicareID alone cannot determine all the other attributes as an employee can be vaccinated multiple times at the same facility. The VaccinationDate can help MedicareID and FacilityID specify each unique vaccination session with the type and dose number.
- For Infection, MedicareID, DateOfInfection, and InfectionType determine the InfectionName and themselves (trivial). They will always result in unique values for the InfectionType because an InfectionType is a subset of an InfectionName (for example, “HCov229E” InfectionType can only refer to “Covid19” InfectionName and nothing else.)

1.3. BCNF Requirement:

All of the relations in our database are in BCNF. First of all, since all of them are in 3NF, we only need to check for any non-trivial dependencies:

- For Employee, only 1 primary key exists (MedicareID) and it is trivial.
- For Facility, only 1 primary key exists (FacilityID) and it is trivial.
- For Schedule, only 1 primary key exists (SchedID) and it is trivial.
- For Vaccine, we have multiple keys (FacilityID, MedicareID, VaccinationDate, VaccinationType). The FD {FacilityID, MedicareID, VaccineType, VaccinationDate

→ DoseNumber} has no non-trivial dependency because the value for the RHS (DoseNumber) can be uniquely determined by the LHS.

- For Infection, all 3 primary keys together determine the RHS (InfectionName), no non-trivial dependency.

1.4. 3NF Requirement:

Since all of the relations and FDs are in BCNF, they are all already in 3NF.

1.5.

1. Trigger to check schedule conflict:

```
Delimiter $$

CREATE TRIGGER NoConflictTimeInsertTrigger

BEFORE INSERT ON Schedule

FOR EACH ROW

BEGIN

    IF (

        EXISTS (

            SELECT *

            FROM Schedule s1

            WHERE s1.MedicareID = NEW.MedicareID

            AND (

                (s1.StartTime <= NEW.StartTime AND s1.EndTime >
NEW.StartTime)

                OR (s1.StartTime < NEW.EndTime AND s1.EndTime >= NEW.EndTime)

                OR (s1.StartTime >= NEW.StartTime AND s1.EndTime <=
NEW.EndTime)

            )

            AND s1.SchedID <> NEW.SchedID

        )

    ) THEN
```

```
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'NoConflictTime constraint
violated';

    END IF;

END; $$

Delimiter;



Delimiter $$

CREATE TRIGGER NoConflictTimeUpdateTrigger

BEFORE UPDATE ON Schedule

FOR EACH ROW

BEGIN

      IF (

      EXISTS (

            SELECT *

             FROM Schedule s1

            WHERE s1.MedicareID = NEW.MedicareID

            AND (

            (s1.StartTime <= NEW.StartTime AND s1.EndTime > NEW.StartTime)

            OR (s1.StartTime < NEW.EndTime AND s1.EndTime >= NEW.EndTime)

              OR (s1.StartTime >= NEW.StartTime AND s1.EndTime <=
NEW.EndTime)

            )

            AND s1.SchedID <> NEW.SchedID

      )

      ) THEN

      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'NoConflictTime constraint
violated';

      END IF;

END; $$
```

```
Delimiter;
```

2. Trigger to check schedule duration

```
DELIMITER $$

CREATE TRIGGER TimeDurationTrigger

BEFORE INSERT ON Schedule

FOR EACH ROW

BEGIN

    IF EXISTS (

            SELECT *

            FROM Schedule s2

            WHERE s2.MedicareID = NEW.MedicareID

            AND (

                (NEW.FacilityID = s2.FacilityID AND ABS(TIMESTAMPDIFF(HOUR,
s2.EndTime, NEW.StartTime)) < 1)

                OR (NEW.FacilityID <> s2.FacilityID AND
ABS(TIMESTAMPDIFF(HOUR, s2.EndTime, NEW.StartTime)) < 24)

            )

            AND s2.SchedID <> NEW.SchedID

        )

    THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Time duration conflict detected';

    END IF;

END;

$$

DELIMITER;
```

3. Trigger to check for infected employees

```
DELIMITER $$
```

```
CREATE TRIGGER InfectedEmployeeTrigger

BEFORE INSERT ON Schedule

FOR EACH ROW

BEGIN

    DECLARE num_rows INT;

    SELECT COUNT(*) INTO num_rows

    FROM Schedule s3

    JOIN Employee e ON s3.MedicareID = e.MedicareID

    JOIN Infection i ON s3.MedicareID = i.MedicareID

    WHERE s3.StartTime < DATE_SUB(i.DateOfInfection, INTERVAL 14 DAY)

    AND s3.MedicareID = NEW.MedicareID;


    IF num_rows > 0 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Infected employee cannot schedule shifts';

    END IF;

END; $$

DELIMITER ;
```

4. Trigger to check for vaccinated employees

```
DELIMITER $$

CREATE TRIGGER VaccinatedEmployeeTrigger

BEFORE INSERT ON Schedule

FOR EACH ROW

BEGIN

    DECLARE vaccinated INT DEFAULT 0;

    SELECT COUNT(*) INTO vaccinated

    FROM Schedule s4
```

```
    JOIN Employee e ON s4.MedicareID = e.MedicareID

    JOIN Vaccine v ON s4.MedicareID = v.MedicareID

    WHERE VaccinationDate > DATE_SUB(EndTime, INTERVAL 6 MONTH)

    AND s4.MedicareID = NEW.MedicareID;

    IF vaccinated = 0 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Employee must be vaccinated within 6 months to be
scheduled';

    END IF;

END; $$

DELIMITER ;
```

Part 2: SQL Commands

2.1: Database Creation Statements:

Creating “Schedule” Table:

```
CREATE TABLE Schedule (

    SchedID INT PRIMARY KEY,
    MedicareID INT,
    FacilityID INT,
    Roles VARCHAR(255),
    Date DATE,
    StartTime DATETIME,
    EndTime DATETIME,
    FOREIGN KEY (MedicareID) REFERENCES Employee(MedicareID),
    FOREIGN KEY (FacilityID) REFERENCES Facility(FacilityID)
);
```

Creating “Employee” table

```
CREATE TABLE Employee (

MedicareID INTEGER PRIMARY KEY,
FirstName VARCHAR(30),
LastName VARCHAR(30),
Citizenship VARCHAR(30),
DateOfBirth DATE,
Email VARCHAR(50),
TelephoneNumber CHAR(10),
Address VARCHAR(255),
City VARCHAR(30),
Province VARCHAR(30),
PostalCode CHAR(6),
Role VARCHAR(30));
```

Creating “Facility” table

```
CREATE TABLE Facility (

FacilityID INTEGER PRIMARY KEY,
Type varchar(30),
Capacity INTEGER,
WebAddress VARCHAR(30),
TelephoneNumber CHAR(10),
Name VARCHAR(30),
Address VARCHAR(255),
```

City VARCHAR(30),
Province VARCHAR(30),
PostalCode CHAR(6),
*ManagerID INTEGER)***;**

Creating “Vaccine” table

CREATE TABLE Vaccine (

DoseNumber INTEGER,
VaccinationDate DATE,
FacilityID INTEGER,
VaccineType VARCHAR(30),
MedicareID INTEGER,
PRIMARY KEY (DoseNumber, MedicareID)
FOREIGN KEY (MedicareID) REFERENCES Employee(MedicareID),
FOREIGN KEY (FacilityID) REFERENCES Facility(FacilityID));

Creating “Infection” table

CREATE TABLE Infection (

DateOfInfection DATE NOT NULL,
InfectionType VARCHAR(30) COLLATE utf8mb3_unicode_ci NOT NULL ,
InfectionName VARCHAR(255) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
MedicareID INTEGER DEFAULT NULL,
PRIMARY KEY (DateOfInfection, InfectionType),
FOREIGN KEY (MedicareID) REFERENCES Employee (MedicareID) ON DELETE SET DEFAULT ON UPDATE CASCADE);

Creating “GeneralManager” table

CREATE TABLE GeneralManager (

MedicareID INTEGER PRIMARY KEY
FOREIGN KEY (MedicareID) REFERENCES Employee (MedicareID) ON DELETE SET DEFAULT ON UPDATE CASCADE);

Creating “Maintains” table

CREATE TABLE Maintains (

FacilityID INTEGER,
SchedID INTEGER,
PRIMARY KEY (FacilityID, SchedID));

Creating “HasA” table

CREATE TABLE HasA (

MedicareID INTEGER,
SchedID INTEGER,
PRIMARY KEY (MedicareID, SchedID));

Creating “ManagedBy” table

CREATE TABLE ManagedBy (

FacilityID INTEGER PRIMARY KEY,
MedicareID INTEGER);

Creating “WorksAt” table

CREATE TABLE WorksAt (

MedicareID INTEGER,

FacilityID INTEGER,

StartDate DATETIME,

EndDate DATETIME,

PRIMARY KEY (MedicareID, FacilityID)

FOREIGN KEY (MedicareID) REFERENCES Employee (MedicareID) ON DELETE SET DEFAULT ON UPDATE CASCADE);
FOREIGN KEY (FacilityID) REFERENCES Facility(FacilityID) ON DELETE SET DEFAULT ON UPDATE CASCADE));

2.2: Database Manipulation Statements:

1. **Create/Delete/Edit/Display a Facility.**

SQL Query::

Result:

2. **Create/Delete/Edit/Display a Employee.**
3. **Create/Delete/Edit/Display a Vaccination.**
4. **Create/Delete/Edit/Display an Infection.**
5. **Assign/Delete/Edit schedule for an Employee. (Attempt to schedule a conflicting assignment for an employee)**

Done. Screenshots will be provided.

6. **Get details of all the facilities in the system. Details include facility's name, address, city, province, postal-code, phone number, web address, type, capacity, general manager's name and number of employees currently working for the facility. Results should be displayed sorted in ascending order by province, then by city, then by type, then by number of employees currently working for the facility.**

SQL Query:

```
WITH EmployeeCount AS (
  SELECT
    FacilityID,
    COUNT(MedicareID) AS NumberOfEmployees
  FROM
    Schedule
  GROUP BY
    FacilityID
)

SELECT
  f.Name AS FacilityName,
  f.Address,
  f.City,
  f.Province,
  f.PostalCode,
  f.TelephoneNumber,
  f.WebAddress,
  f.Type,
```

```
 f.Capacity,
 CONCAT(e.FirstName, ' ', e.LastName) AS GeneralManagerName,
 ec.NumberOfEmployees
FROM
 Facility f
JOIN
 ManagedBy mb ON f.FacilityID = mb.FacilityID
JOIN
 Employee e ON mb.MedicareID = e.MedicareID
JOIN
 EmployeeCount ec ON f.FacilityID = ec.FacilityID
ORDER BY
 f.Province ASC,
 f.City ASC,
 f.Type ASC,
 ec.NumberOfEmployees ASC;
```

Result:

```
mysql> WITH EmployeeCount AS (
    ->   SELECT
    ->     FacilityID,
    ->     COUNT(MedicareID) AS NumberOfEmployees
    ->   FROM
    ->     Schedule
    ->   GROUP BY
    ->     FacilityID
    -> )
    ->
    -> SELECT
    ->   f.Name AS FacilityName,
    ->   f.Address,
    ->   f.City,
    ->   f.Province,
    ->   f.PostalCode,
    ->   f.TelephoneNumber,
    ->   f.WebAddress,
    ->   f.Type,
    ->   f.Capacity,
    ->   CONCAT(e.FirstName, ' ', e.LastName) AS GeneralManagerName,
    ->   ec.NumberOfEmployees
    -> FROM
    ->   Facility f
    -> JOIN
    ->   ManagedBy mb ON f.FacilityID = mb.FacilityID
    -> JOIN
    ->   Employee e ON mb.MedicareID = e.MedicareID
    -> JOIN
    ->   EmployeeCount ec ON f.FacilityID = ec.FacilityID
    -> ORDER BY
    ->   f.Province ASC,
    ->   f.City ASC,
    ->   f.Type ASC,
    ->   ec.NumberOfEmployees ASC;
+--------------------------+--------------------+----------+----------+------------+-----------------+--------------------+----------+----------+--------------------
-+------------------+
| FacilityName             | Address            | City     | Province | PostalCode | TelephoneNumber | WebAddress         | Type     | Capacity | GeneralManagerName
 | NumberOfEmployees |
+--------------------------+--------------------+----------+----------+------------+-----------------+--------------------+----------+----------+--------------------
-+------------------+
| Pharmaprox               | 99 Cowboy Boulevard | Calgary  | Alberta  | P5M2G8     | 5143849728      | www.pharmaprox.com | Pharmacy |        4 | George Smith
 |                1 |
| Winnipeg Clinic          | 33 Empty Street    | Winnipeg | Manitoba | H8L6M2     | 5149878900      | www.winnclinic.com | Clinic   |       11 | Joe Smith
 |                1 |
| Clinic Vanier            | 344 Yellow Street  | Montreal | Quebec   | H8J6D0     | 5145433333      | www.vanierclinic.com | Clinic   |       24 | Xaavian Ali
 |                2 |
| Hospital Maisonneuve Rosemont | 34 Rue Clark       | Montreal | Quebec   | H7A8L8     | 5144535678      | www.hrosemont.com  | Hospital |       12 | Xaavian Ali
 |                6 |
+--------------------------+--------------------+----------+----------+------------+-----------------+--------------------+----------+----------+--------------------
-+------------------+
4 rows in set (0.01 sec)
```

7. **Get details of all the employees currently working in a specific facility. Details include employee's first-name, last-name, start date of work, date of birth, Medicare card number, telephone-number, address, city, province, postal-code, citizenship, and email address. Results should be displayed sorted in ascending order by role, then by first name, then by last name.**

SQL Query:

*Note: For this query, we are assuming that “currently working in a specific facility” means that an Employee is employed at a specific facility, and not necessarily scheduled for the day. Therefore as per requirements, we will check if the endtime of an Employee working at a facility is NULL or not to filter them out.

```
SELECT
  e.FirstName,
  e.LastName,
  wa.StartDate,
  e.DateOfBirth,
  e.MedicareID,
  e.TelephoneNumber,
  e.Address,
  e.City,
  e.Province,
  e.PostalCode,
  e.Citizenship,
  e.Email
FROM
  WorksAt wa
JOIN
  Employee e ON wa.MedicareID = e.MedicareID
WHERE
  wa.FacilityID = ****INSERT DESIRED FACILITY ID***
  AND wa.EndDate IS NULL
ORDER BY
  e.Role ASC,
  e.FirstName ASC,
  e.LastName ASC;
```

Result:

For Facility ID 100:

```
mysql> SELECT
    ->   e.FirstName,
    ->   e.LastName,
    ->   wa.StartDate,
    ->   e.DateOfBirth,
    ->   e.MedicareID,
    ->   e.TelephoneNumber,
    ->   e.Address,
    ->   e.City,
    ->   e.Province,
    ->   e.PostalCode,
    ->   e.Citizenship,
    ->   e.Email
    -> FROM
    ->   WorksAt wa
    -> JOIN
    ->   Employee e ON wa.MedicareID = e.MedicareID
    -> WHERE
    ->   wa.FacilityID = 100
    ->   AND wa.EndDate IS NULL
    -> ORDER BY
    ->   e.Role ASC,
    ->   e.FirstName ASC,
    ->   e.LastName ASC;
+-----------+----------+------------+-------------+--------------+-----------------+------------+---------+----------+------------+-------------+-----------------+
| FirstName | LastName | StartDate  | DateOfBirth | MedicareID | TelephoneNumber | Address    | City    | Province | PostalCode | Citizenship | Email           |
+-----------+----------+------------+-------------+--------------+-----------------+------------+---------+----------+------------+-------------+-----------------+
| Lucie     | Smith    | 2019-02-24 | 1990-01-23  |   11111111 | 5148372899      | 34 Pine St | Montreal | Quebec   | F4S7Q2     | Russian     | lucie@gmail.com |
+-----------+----------+------------+-------------+--------------+-----------------+------------+---------+----------+------------+-------------+-----------------+
1 row in set (0.00 sec)
```

For FacilityID 101:

```
mysql>
mysql> SELECT
    ->   e.FirstName,
    ->   e.LastName,
    ->   wa.StartDate,
    ->   e.DateOfBirth,
    ->   e.MedicareID,
    ->   e.TelephoneNumber,
    ->   e.Address,
    ->   e.City,
    ->   e.Province,
    ->   e.PostalCode,
    ->   e.Citizenship,
    ->   e.Email
    -> FROM
    ->   WorksAt wa
    -> JOIN
    ->   Employee e ON wa.MedicareID = e.MedicareID
    -> WHERE
    ->   wa.FacilityID = 101
    ->   AND wa.EndDate IS NULL
    -> ORDER BY
    ->   e.Role ASC,
    ->   e.FirstName ASC,
    ->   e.LastName ASC;
+-----------+----------+------------+-------------+------------+-----------------+-------------+---------+----------+------------+-------------+-----------------+
| FirstName | LastName | StartDate  | DateOfBirth | MedicareID | TelephoneNumber | Address     | City    | Province | PostalCode | Citizenship | Email           |
+-----------+----------+------------+-------------+------------+-----------------+-------------+---------+----------+------------+-------------+-----------------+
| Jane      | Smith    | 2020-03-25 | 1991-01-14  |   22222222 | 5143928744      | 35 Oak St   | Montreal | Quebec   | L8V4X7     | Russian     | jane@gmail.com  |
| John      | Smith    | 2020-05-27 | 1993-01-02  |   44444444 | 5141029387      | 46 Big St   | Montreal | Quebec   | P9P4N7     | Italian     | john@gmail.com  |
| Mike      | Smith    | 2019-06-28 | 1994-01-23  |   55555555 | 5147774637      | 353 Palm St | Montreal | Quebec   | D4G2A8     | Canadian    | mike@gmail.com  |
| Sally     | Smith    | 2019-08-30 | 1998-01-30  |   77777777 | 5142009877      | 63 Funny St | Montreal | Quebec   | H1H1H9     | Canadian    | sally@gmail.com |
| Sam       | Smith    | 2020-07-29 | 1988-01-13  |   66666666 | 5149283744      | 85 Maple St | Montreal | Quebec   | K8K4F7     | Canadian    | sam@gmail.com   |
+-----------+----------+------------+-------------+------------+-----------------+-------------+---------+----------+------------+-------------+-----------------+
5 rows in set (0.00 sec)
```

8. **For a given employee, get the details of all the schedules she/he has been scheduled during a specific period of time. Details include facility name, day of the year, start time and end time. Results should be displayed sorted in ascending order by facility name, then by day of the year, the by start time.**

SQL Query:

```
SELECT

  f.Name AS FacilityName,
  DATE(s.StartTime) AS DayOfYear,
  TIME(s.StartTime) AS StartTime,
  TIME(s.EndTime) AS EndTime
FROM
  Schedule s
JOIN
  Facility f ON s.FacilityID = f.FacilityID
WHERE
 s.MedicareID = ****desired employee id****
  AND s.StartTime >= ****desired start date****
  AND s.EndTime <= ****desired end date****
ORDER BY
  FacilityName ASC,
  DayOfYear ASC,
  StartTime ASC;
```

Result:

```
mysql> SELECT
    ->   f.Name AS FacilityName,
    ->   DATE(s.StartTime) AS DayOfYear,
    ->   TIME(s.StartTime) AS StartTime,
    ->   TIME(s.EndTime) AS EndTime
    -> FROM
    ->   Schedule s
    -> JOIN
    ->   Facility f ON s.FacilityID = f.FacilityID
    -> WHERE
    ->   s.MedicareID = 44444444
    ->   AND s.StartTime >= '2023-01-01'
    ->
    -> ORDER BY
    ->   FacilityName ASC,
    ->   DayOfYear ASC,
    ->   StartTime ASC;
+-----------------------------+------------+-----------+----------+
| FacilityName                | DayOfYear  | StartTime | EndTime  |
+-----------------------------+------------+-----------+----------+
| Hospital Maisonneuve Rosemont | 2023-04-10 | 19:30:00  | 07:30:00 |
| Hospital Maisonneuve Rosemont | 2023-04-11 | 19:30:00  | 07:30:00 |
+-----------------------------+------------+-----------+----------+
2 rows in set (0.00 sec)
```

9. **Get details of all the doctors who have been infected by COVID-19 in the past two weeks. Details include doctor's first-name, last-name, date of infection, and the name of the facility that the doctor is currently working for. Results should be displayed sorted in ascending order by the facility name, then by the first-name of the doctor.**

SELECT e.FirstName, e.LastName, i.DateOfInfection, f.Name AS FacilityName

FROM Employee e

JOIN WorksAt w ON e.MedicareID = w.MedicareID

JOIN Facility f ON w.FacilityID= f.FacilityID

JOIN Infection i ON e.MedicareID = i.MedicareID AND DateOfInfection >= DATE_SUB(NOW(), INTERVAL 2 WEEK)

WHERE e.Role = 'Doctor'AND i.InfectionType=’Covid-19’

ORDER BY FacilityName ASC, e.FirstName ASC;

```
mysql> Select e.FirstName, e.LastName, i.DateOfInfection,f.Name AS FacilityName From Employee e Join WorksAt w on e.MedicareID = w.MedicareID Join Facility f On w.FacilityID= f.FacilityID Join Infection i On e.MedicareID= i.MedicareId and DateOfInfection >= DATE_SUB(NOW(
),INTERVAL 2 WEEK) Where e.Role='Doctor' Order By FacilityName ASC, e.FirstName ASC;
Empty set (0.00 sec)

mysql>
```

10. **List the emails generated by a given facility. The results should be displayed in ascending order by the date of the emails.**

```
SELECT

   l.Subject,

   l.Body,

   l.EmailDate

   FROM

       Logs l

   WHERE

       l.FacilityID = ".$facilityID."

   ORDER BY

       l.EmailDate ASC;
```

```
mysql> select l.EmailDate,e.Email
    -> From Logs l
    -> Inner Join Employee e On l.MedicareID = e.MedicareID
    -> Inner join Schedule s On e.MedicareID=s.MedicareID
    -> Inner Join Facility f On s.FacilityID= f.FacilityId
    -> Where f.Name = 'Mega Hospital'
    -> Order by l.EmailDate ASC;
+------------+--------------------+
| EmailDate  | Email              |
+------------+--------------------+
| 2023-04-12 | ash@outlook.com    |
| 2023-04-16 | lucie@gmail.com    |
| 2023-04-16 | lucie@gmail.com    |
| 2023-04-16 | lucie@gmail.com    |
| 2023-04-16 | xaavian@hotmail.com |
| 2023-04-16 | xaavian@hotmail.com |
| 2023-04-23 | lucie@gmail.com    |
| 2023-04-23 | lucie@gmail.com    |
| 2023-04-23 | lucie@gmail.com    |
| 2023-04-23 | xaavian@hotmail.com |
| 2023-04-23 | xaavian@hotmail.com |
+------------+--------------------+
11 rows in set (0.00 sec)

mysql>
```

11. **For a given facility, generate a list of all the doctors and nurses who have been on schedule to work in the last two weeks. The list should include first-name, last-name, and role. Results should be displayed in ascending order by role, then by first name.**

```
SELECT DISTINCT

      e.FirstName,

      e.LastName,

      e.Role

  FROM

      Employee e

  JOIN

      Schedule s ON e.MedicareID = s.MedicareID

  WHERE

      s.FacilityID = ".$facilityID." AND

      s.StartTime >= DATE_SUB(NOW(), INTERVAL 2 WEEK) AND
```

```
      s.StartTime <= NOW() AND

      e.Role IN ('Nurse', 'Doctor')

  ORDER BY

      e.Role ASC, e.FirstName ASC;
```

```
mysql> SELECT s.Roles, SUM(TIMESTAMPDIFF(SECOND
59" Group by s.Roles Order by s.Roles ASC;
+--------+------------+
| Roles  | TotalHours |
+--------+------------+
| Doctor |     8.0000 |
+--------+------------+
1 row in set (0.00 sec)

mysql>
```

12. For a given facility, give the total hours scheduled for every role during a specific period. Results should be displayed in ascending order by role.

```
SELECT s.Roles, SUM(TIMESTAMPDIFF(SECOND ,s.StartTime,
s.EndTime)/3600) AS TotalHours From Schedule s

Where s.FacilityID = ".$facilityID." AND s.StartTime >=
'".$startTime.
```

13. For every facility, provide the province where the facility is located, the facility name, the capacity of the facility, and the total number of employees in the facility who have been infected by COVID-19 in the past two weeks. The results should be displayed in ascending order by province, then by the total number of employees infected.

```
SELECT f.Province as FacilityProvince, f.Name as FacilityName,
f.Capacity as Capacity, COUNT(DISTINCT i.MedicareID) as
EmployeesInfected

FROM Facility f

JOIN WorksAt w ON f.FacilityID = w.FacilityID

JOIN Employee e on w.MedicareID = e.MedicareID

LEFT JOIN Infection i ON e.MedicareID = i.MedicareID AND
i.DateOfInfection >= DATE_SUB(NOW(), INTERVAL 2 WEEK)

GROUP BY f.FacilityID

ORDER BY f.Province ASC, EmployeesInfected ASC;
```

FacilityProvince	FacilityName	Capacity	EmployeesInfected
Alberta	Pharmaprox	4	0
Manitoba	Winnipeg Clinic	11	0
Quebec	Mega Hospital	20	0
Quebec	Hospital Maisonneuve Rosemont	12	0
Quebec	Clinic Vanier	24	0

14. **For every doctor who is currently working in the province of "Québec", provide the doctor's first-name, last-name, the city of residence of the doctor, and the total number of facilities the doctor is currently working for. Results should be displayed in ascending order by city, then in descending order by total number of facilities.**

```
SELECT e.FirstName as FirstName, e.LastName as LastName, e.City as
City, COUNT(DISTINCT w.FacilityID) as TotalFacilities

FROM Employee e

JOIN WorksAt w ON e.MedicareID = w.MedicareID

WHERE w.EndDate IS NULL AND e.Province = 'Quebec' AND e.Role =
'Doctor'

GROUP BY e.MedicareID

ORDER BY e.city ASC, TotalFacilities DESC;
```

FirstName	LastName	City	TotalFacilities
Lucie	Smith	Montreal	1
Xaavian	Ali	Montreal	1
Jane	Smith	Montreal	1

15. **Get details of the nurse(s) who is/are currently working and has the highest number of hours scheduled in the system since they started working as a nurse. Details include first-name, last-name, first day of work as a nurse, date of birth, email address, and total number of hours scheduled.**

```
SELECT e.FirstName, e.LastName, e.DateOfBirth, e.Email,
MIN(w.StartDate) as FirstDayAsNurse, SUM(TIMESTAMPDIFF(HOUR,
s.StartTime, s.EndTime)) as TotalHoursScheduled

FROM Employee e

INNER JOIN WorksAt w ON e.MedicareID = w.MedicareID

INNER JOIN Schedule s ON w.MedicareID = s.MedicareID AND w.FacilityID
= s.FacilityID

WHERE e.role = 'Nurse' AND w.EndDate IS NULL

GROUP BY e.MedicareID

ORDER BY TotalHoursScheduled DESC

LIMIT 1;
```

FirstName	LastName	DateOfBirth	Email	FirstDayAsNurse	TotalHoursScheduled
John	Smith	1993-01-02	john@gmail.com	2020-05-27	24

16. **Get details of the nurse(s) or the doctor(s) who are currently working and has been infected by COVID-19 at least three times. Details include first-name, last-name, first day of work as a nurse or as a doctor, role (nurse/doctor), date of birth, email address, and total number of hours scheduled. Results should be displayed sorted in ascending order by role, then by first name, then by last name.**

```
SELECT e.FirstName, e.LastName,
```

```
e.Role,

e.DateOfBirth,

e.Email,

SUM(TIMESTAMPDIFF(HOUR, s.StartTime, s.EndTime)) AS
TotalHoursScheduled

FROM Employee e

JOIN WorksAt w ON e.MedicareID = w.MedicareID AND EndDate IS NULL

JOIN Schedule s ON w.MedicareID = s.MedicareID

LEFT JOIN Infection i ON e.MedicareID = i.MedicareID

WHERE e.Role IN ('Nurse', 'Doctor')

GROUP BY e.MedicareID

HAVING COUNT(i.MedicareID) >= 3

ORDER BY e.Role ASC, e.FirstName ASC, e.LastName ASC;
```

FirstName	LastName	Role	DateOfBirth	Email	TotalHoursScheduled
Sam	Smith	Nurse	1988-01-13	sam@gmail.com	24

17. **Get details of the nurse(s) or doctor(s) who are currently working and has never been infected by COVID-19. Details include first-name, last-name, first day of work as a nurse or as a doctor, role (nurse/doctor), date of birth, email address, and total number of hours scheduled. Results should be displayed sorted in ascending order by role, then by first name, then by last name**

```
WITH WorkHours AS (

   SELECT

       e.MedicareID,

       SUM(TIMESTAMPDIFF(SECOND, s.StartTime, s.EndTime) / 3600) AS
TotalHours

   FROM

       Employee e
```

```
JOIN
    Schedule s ON e.MedicareID = s.MedicareID
WHERE
    s.StartTime <= NOW() AND s.EndTime >= NOW()
GROUP BY
    e.MedicareID
),
FilteredEmployee AS (
    SELECT
        e.*
    FROM
        Employee e
    LEFT JOIN
        Infection i ON e.MedicareID = i.MedicareID
    WHERE
        i.InfectionName != 'COVID-19' OR i.InfectionName IS NULL
)
SELECT
    fe.FirstName,
    fe.LastName,
    fe.Role,
    fe.DateOfBirth,
    fe.Email,
    wh.TotalHours as TotalHoursScheduled
FROM
    FilteredEmployee fe
JOIN
    WorkHours wh ON fe.MedicareID = wh.MedicareID
WHERE
    fe.Role IN ('Nurse', 'Doctor')
ORDER BY
```

```
      fe.Role ASC, fe.FirstName ASC, fe.LastName ASC;
```

```
+-----------+----------+-------+-------------+-----------------+---------------------+
| FirstName | LastName | Role  | DateOfBirth | Email           | TotalHoursScheduled |
+-----------+----------+-------+-------------+-----------------+---------------------+
| John      | Smith    | Nurse | 1993-01-02  | john@gmail.com  |             12.0000 |
| Sally     | Smith    | Nurse | 1998-01-30  | sally@gmail.com |             12.0000 |
+-----------+----------+-------+-------------+-----------------+---------------------+
2 rows in set (0.02 sec)
```

18. You should show the trigger(s) used by your system. Explain the trigger(s) used and their benefits.

The following triggers were added to the Schedule entity (as per part 1.6) and their usages are as follows:

a. NoConflictTimeInsertTrigger
   i. This trigger is used to make sure that there is no time conflict whenever a new schedule is inserted into the database, which is that: An employee cannot be scheduled at two different conflicting times neither at the same facility nor at different facilities.

b. NoConflictTimeUpdateTrigger
   i. This trigger acts the same as NoConflictTimeInsertTrigger but upon updating the schedule.

c. TimeDurationTrigger
   i. This trigger is used to ensure that if an employee is scheduled for two different periods on the same day either at the same facility or at different facilities, then at least one hour should be the duration between the first schedule and the second one.

d. InfectedEmployeeTrigger
   i. This trigger is used to ensure that if a nurse or a doctor is infected by COVID-19, then he/she cannot be scheduled to work for at least two weeks from the date of infection.

e. VaccinatedEmployeeTrigger
   i. This trigger is used to ensure that an employee cannot be scheduled if she/he is not vaccinated, at least one vaccine for COVID-19 in the past six months prior to the date of the new schedule.

19. You need to demonstrate the integrity of all the requirements provided in the description. Example, the system should not allow a user to schedule an employee on two different conflicting time.

i. Error when trying to schedule an employee at the same time at 2 different facilities:

```
mysql> select * from Schedule;
+---------+------------+------------+---------------------+---------------------+------------+
| SchedID | FacilityID | MedicareID | StartTime           | EndTime             | Roles      |
+---------+------------+------------+---------------------+---------------------+------------+
|       1 |        102 |   12345678 | 2022-12-01 09:00:00 | 2022-12-01 17:00:00 | Doctor     |
|       2 |        101 |   44444444 | 2022-12-01 09:00:00 | 2022-12-01 17:00:00 | Nurse      |
|       3 |        101 |   44444444 | 2022-12-02 09:00:00 | 2022-12-02 17:00:00 | Nurse      |
|       4 |        101 |   55555555 | 2022-12-02 09:00:00 | 2022-12-02 17:00:00 | Nurse      |
|       5 |        101 |   55555555 | 2022-12-03 09:00:00 | 2022-12-03 17:00:00 | Nurse      |
|       6 |        101 |   66666666 | 2022-12-03 09:00:00 | 2022-12-03 17:00:00 | Nurse      |
|       7 |        101 |   44444444 | 2022-12-03 09:00:00 | 2022-12-03 17:00:00 | Nurse      |
|       8 |        104 |   88888888 | 2022-12-03 09:00:00 | 2022-12-03 17:00:00 | Nurse      |
|       9 |        102 |   12345678 | 2022-12-03 09:00:00 | 2022-12-03 17:00:00 | Doctor     |
|      10 |        103 |   33333333 | 2022-12-03 09:00:00 | 2022-12-03 17:00:00 | Doctor     |
|     909 |        104 |   99999999 | 2023-04-30 13:00:00 | 2023-04-23 17:00:00 | Pharmacist |
+---------+------------+------------+---------------------+---------------------+------------+
11 rows in set (0.00 sec)

mysql> INSERT INTO Schedule (SchedID, MedicareID, FacilityID, Roles, StartTime, EndTime) VALUES (910, 99999999, 102, 'Pharmacist', '2023-04-30 13:00:00', '2023-04-23 17:00:00');
ERROR 1644 (45000): NoConflictTime constraint violated
```

ii. Error when trying to schedule an employee who got infected within 2 weeks from the scheduled date:

```
mysql> select * from Infection;
+----------------+---------------+------------+---------------+
| DateOfInfection | InfectionType | MedicareID | InfectionName |
+----------------+---------------+------------+---------------+
| 2022-01-01     | Covid-19      |   66666666 | Flu           |
| 2022-01-02     | Covid-19      |   66666666 | Common Cold   |
| 2022-01-03     | Covid-19      |   66666666 | Covid-19      |
| 2022-01-04     | SARS-Cov-2    |   12345678 | Covid-19      |
| 2022-01-05     | SARS-Cov-2    |   22222222 | Pneumonia     |
| 2022-01-06     | SARS-Cov-2    |   99999999 | Flu           |
| 2023-04-06     | Covid-19      |   55555555 | Covid-19      |
+----------------+---------------+------------+---------------+
7 rows in set (0.00 sec)

mysql> INSERT INTO Schedule (SchedID, MedicareID, FacilityID, Roles, StartTime, EndTime) VALUES (899, 55555555, 101, 'Nurse', '2023-04-12 13:00:00', '2023-04-12 17:00:00');
ERROR 1644 (45000): Infected employee cannot schedule shifts
```

20. You need to demonstrate the generation of emails and the logs of the emails produced by the system.

Here's the schedule for employee 2, before being infected.

Schedules for Employee: 2

Add Schedule

SchedID	MedicareID	FacilityID	Roles	StartTime	EndTime	Actions
21	2	100	Nurse	2023-05-19 19:20:00	2023-05-19 08:30:00	Edit Delete
22	2	100	Nurse	2023-04-19 16:53:00	2023-04-20 22:53:00	Edit Delete

Go Back

This is after adding the infection. As we can see, all shifts for the next 2 weeks have been cancelled.

Schedules for Employee: 2

Add Schedule

SchedID	MedicareID	FacilityID	Roles	StartTime	EndTime	Actions
21	2	100	Nurse	2023-05-19 19:20:00	2023-05-19 08:30:00	Edit Delete

Go Back

As we can see the infection has been added correctly.

Infections

Add Infection

DateOfInfection	InfectionType	InfectionName	MedicareID	Actions
2022-01-01	Covid-19		66666666	Edit Delete
2022-01-02	Covid-19		66666666	Edit Delete
2022-01-03	Covid-19		66666666	Edit Delete
2022-01-04	Influenza		12345678	Edit Delete
2022-01-05	Influenza		22222222	Edit Delete
2022-01-06	Influenza		99999999	Edit Delete
2023-04-06	Covid-19	Covid-19	55555555	Edit Delete
2023-04-11	Covid-19	Covid-19	2	Edit Delete

Go Back

The email to a Doctor that has had overlapping schedules in the past 2 weeks with the infected employee has also been generated correctly (today is april 11th):

Query 10

EmailDate	Subject	Body
2023-04-11	Warning	One of your colleagues that you have worked with in the past two weeks have been
2023-04-12	Infected	You have been infected
2023-04-16	Warning	One of your colleagues that you have worked with in the past two weeks have been
2023-04-16	Warning	One of your colleagues that you have worked with in the past two weeks have been
2023-04-16	Warning	One of your colleagues that you have worked with in the past two weeks have been
2023-04-16	Warning	One of your colleagues that you have worked with in the past two weeks have been
2023-04-23	Mega Hospital Schedule for Monday 24-Apr-2023 to Sunday 30-Apr-2023	Mega Hospital, 32 Rue du Hopital, Montreal, Quebec, H0H0H0, Xaavian Ali, xaav
2023-04-23	Mega Hospital Schedule for Monday 24-Apr-2023 to Sunday 30-Apr-2023	Mega Hospital, 32 Rue du Hopital, Montreal, Quebec, H0H0H0, Mike Smith, mike@gma

Go Back

Here are the 2 queries I've used.

1) Cancelling shifts for the next 2 weeks:

```
DELETE FROM
            Schedule
        WHERE
            MedicareID = :MedicareID
        AND StartTime BETWEEN :DateOfInfection AND
DATE_ADD(:DateOfInfection, INTERVAL 2 WEEK);
```

2) Generating email

```
INSERT INTO Logs (EmailDate, Subject, Body, MedicareID, FacilityID)
        SELECT
            CURDATE(),
            'Warning',
            'One of your colleagues that you have worked with in the
past two weeks have been',
            e.MedicareID,
            s1.FacilityID
        FROM
            Employee e
        JOIN
            Schedule s1 ON e.MedicareID = s1.MedicareID
```

```
        JOIN
            Schedule s2 ON s1.FacilityID = s2.FacilityID AND s1.SchedID
<> s2.SchedID
        WHERE
            s2.MedicareID = :MedicareID
        AND
            s1.StartTime <= s2.EndTime
        AND
            s1.EndTime >= s2.StartTime
        AND s1.StartTime >= DATE_SUB(:DateOfInfection, INTERVAL 2
WEEK);
```