**mHealh Database**

Corey Bright

Elijah Chong Tay

Harpreet Bains

Faizan Rafieuddin

Adesh Rai

Hannah Applebaum

Group 2

**Table of Contents**

Introduction 2

Project Specification 2

Statement of Objectives 2

Requirements 2

Design 3

*Schema* 3

*Data Dictionary* 4

*Constraints* 6

*Queries* 9

*Requirements Matrix* 9

Implementation 10

*Queries for Table creation* 10

*Constraint Test Data 13*

*Testing Procedure for Constraints 15*

*Queries for View Creation* 16

*Queries for Trigger Creation 18*

*Trigger Test data* 21

*Testing Procedure for Triggers* 27

**Introduction**

This document attempts to summarize the requirements, analysis, and design of a database system designed to support the needs of Hazen health center workers and patients.

**Project Specification**

This database will contain information related to patients i.e., college students and teachers. It will have tables that store appointment times and dates. It will also have tables that have all the patient information relating to medicines, allergies, personal information, disabilities, emergency contacts, vaccinations, etc. This database will also have a table that stores all relevant information for the Doctors present at the Hazen Health Center for e.g., their availability, specialization etc. Additionally, it will allow for communication via messages between the employees of the Hazen Health Center and the patients.

**Statement of Objectives**

The Database system we will be working on is related to the Hazen Health Portal. It would facilitate the operational team at Hazen to store health-related information allowing more functionality to the simple frontend that has already been designed. This database will allow for better health care by improving all aspects of patient care, including effectiveness, patient-centeredness, communication, efficiency, and equity. ‘

**Requirements**

The requirements list is as follows:

1. The database must have functionalities that enable a patient to schedule appointments with no overlaps at the same time between different patients and the same patient.
2. The database must allow patients to cancel and reschedule appointments based on the situation.
3. The database must provide functionalities to store the employee information and enable the staff at the Hazen Health Center to retrieve and inspect information regarding the patients, appointments, when required and according to their respective roles.
4. The database must contain the information related to doctors currently working at the Hazen Health center along with their specialization and availability throughout the day.
5. The database must contain immunization, insurance, and medication information for the patients.
6. The database must allow multiple medications, immunizations, and insurances for a patient as there is a possibility that these fields have multiple entries.
7. The database must allow one-way communication between the patients and the employees of the Hazen Health Center via messaging.

**Design**

**Schema**

**person** (person\_id: INT, first\_name: STRING, middle\_intial: STRING, last\_name: STRING, date\_of\_birth: DATE) Keys: {person\_id}, {first\_name, last\_name, date\_of\_birth, address}

**phone\_number** (person\_id: INT references person, phone\_number: INT) Keys: {person\_id, phone\_number}

**patient** (patient\_id: INT reference person(person\_id), password\_hash: BINARY(64), school\_email: varchar(50)) Keys: {patient\_id}

**doctor** (doctor\_id: INT reference person(person\_id), primary\_id: VARCHAR(50), secondary\_id: VARCHAR(50)) Keys: {doctor\_id}

**specialty** (doctor\_id: INT references doctor, specialty: STRING) Keys: {doctor\_id, specialty}

**doctor\_availability** (availability\_id: INT, doctor\_id: INT references doctor, availability\_date: DATE, start\_time: TIME, end\_time: TIME) Keys: {doctor\_id, specialty}

**employee** (employee\_id: INT reference person(person\_id), job\_title: STRING, start\_date: DATE, end\_date: DATE, employee\_email: VARCHAR(50), secondary\_email: VARCHAR(50)) Keys: {employee\_id}

**appointment** (appointment\_id: INT, patient\_id: INT references patient, doctor\_id: INT references doctor, date: DATE, start\_time: TIME, end\_time: TIME, location: STRING) Keys: {appointment\_id}, {patient\_id, location, date, time}

**insurance** (insurance\_id: INT, patient\_id: INT references patient, name: STRING, group\_number: INT, policy\_number: INT). Keys: {insurance\_id}

**medication** (medication:\_id, INT, patient\_id: INT references patient,  medication\_name: STRING, start\_date: DATE, end\_date: DATE) Keys: {medication\_id)

**immunization** (immunization\_id: ID, patient\_id: INT references patient, vaccine\_name: STRING, immunization\_date: DATE) Keys: {immunization\_id}

**message** (message\_id: INT,  sender\_id: INT references employee(employee\_id), receiver\_id: INT references patient(patient\_id), title: STRING body: TEXT) Keys: {message\_id}

**Data Dictionary**

PERSON

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| person\_id | INT | Key for a person | 1 |
| first\_name | VARCHAR(20) | Person’s first name | Henry |
| middle\_initial | VARCHAR(1) | Person’s middle initial (optional, defaults to blank) | W |
| last\_name | VARCHAR(30) | Person’s last name | Ale |
| date\_of\_birh | DATE | Person’s birth day (Format: YYYY-MM-DD) | 2001-11-09 |

TELEPHONE

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| telephone | STRING(10) | Person’s 10 digit telephone number | 5854385531 |
| person\_id | INT | Key for a person | 1 |

EMPLOYEE

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| employee\_id | INT | Key for an employee | 12 |
| start\_date | DATE | Date employee started (Format: YYYY-MM-DD) | 2015-02-03 |
| end\_date | DATE | Date employee ended  (Format: YYYY-MM-DD)  (optional, defaults to blank) | 2016-03-24 |
| job\_title | VARCHAR(30) | Employee’s job position | Secretary |
| employee\_email | VARCHAR(50) | employees email address | abc123@hazen.com |
| secondary\_email | VARCHAR(50) | secondary email address | Ab1123@gmail.com |

PATIENT

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| patient\_id | INT | Key for a patient | 9 |
| password\_hash | BINARY(64) | Hashed password for a patient | 1f6e2a9cdc81f93fc9bfefe8 |
| school\_email | VARCHAR(50) | employees email address | abc1@brockport.edu |

MESSAGE

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| message\_id | INT | Key for message | 57 |
| sender\_id | INT | Key for sender | 1 |
| receiver\_id | INT | Key for receiver | 8 |
| title | VARCHAR(50) | Title of message | Reminder |
| body | TEXT | Message being sent | This is your reminder that you have an appointment on 02/14/2024 |

INSURANCE

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| insurance\_id | INT | Key for insurance | 25 |
| patient\_id | INT | Key for a patient | 1 |
| name | VARCHAR(80) | Insurance Name | Blue Cross Blue Shield Association |
| policy\_number | VARCHAR(20) | Insurance policy number | BC331FFS31S2D |
| group\_number | VARCHAR(20) | Insurance group number | 234BX3FGH12 |

IMMUNIZATION

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| immunization\_id | INT | Key for immunization | 8 |
| patient\_id | INT | Key for a patient | 1 |
| vaccine\_name | VARCHAR(50) | Name of vaccine | Chickenpox Vaccination |
| Immunization\_date | DATE | Date vaccine was given  (Format: YYYY-MM-DD) | 2015-08-13 |

MEDICATION

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| medication\_id | INT | Key for medication | 30 |
| patient\_id | INT | Key for a patient | 1 |
| Medication\_name | VARCHAR(50) | Name of medication | Adderall |
| start\_date | DATE | Date medication started  (Format: yyyy-mm-dd) | 2013-10-09 |
| end\_date | DATE | Date medication ended  (Format: YYYY-MM-DD)  (optional, defaults to blank) | 2013-11-09 |

DOCTOR

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| doctor\_id | INT | Key for doctor | 5 |
| primary\_email | VARCHAR(50) | Primary email for doctor | Doctor1@gmail.com |
| secondary\_email | VARCHAR(50) | secondary email for doctor (optional, defaults to blank) | Doctor2@gmail.com |

SPECIALTY

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| doctor\_id | INT | Key for doctor | 5 |
| specialty | VARCHAR(30) | Describes what the doctor specializes | Neurology |

DOCTOR\_AVAILABILITY

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| availability\_id | INT | Key for availability | 3 |
| doctor\_id | INT | Key for a doctor | 1 |
| availability\_date | DATE | Date of availability (Format: yyyy-mm-dd) | 2023-01-01 |
| start\_time | TIME | HH:MI:SS | 12:00:00 |
| end\_time | TIME | HH:MI:SS | 12:30:00 |

APPOINTMENT

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| appointment\_id | INT | Key for appointment | 30 |
| patient\_id | INT | Key for a patient | 1 |
| doctor\_id | INT | Key for a doctor | 8 |
| appointment\_date | DATE | Date of the appointment  (Format: yyyy-mm-dd) | 2023-01-01 |
| start\_time | TIME | time for start of appointment.  (Format: YYYY-MM-DD HH:MI:SS) | 2023-02-14 12:30:00 |
| endtart\_time | TIME | Date and time for end of appointment  (Format: YYYY-MM-DD HH:MI:SS) | 2023-02-14 13:00:00 |

**Constraints**

**The following custom domains are part of the solution:**

**location:** This attribute defines the location where the appointment would take place, this can be either one of the two options i.e., “Online” or “In Person”.

**time:** This attribute defines the time when the appointment will take place. The domain of times from which a value can be chosen for this attribute must be in between 8:00 AM to 6:00 PM.

**start date:** This attribute defines the start date of a medicine, and it should be between the dates from January 1950 to December 2100 (just in case).

**end date:** This attribute defines the end date of a medicine, and it should be between the dates from January 1950 to December 2100 (just in case).

**date:** This attribute defines the date for when an immunization was taken. The value of this should be in between January 1950 to the Current Date i.e., not in the future.

**availability:** This attribute defines the times when a doctor is available throughout the day. The domain of times from which a value can be chosen for this attribute must be in between 8:00 AM to 6:00 PM.

**d.o.b:** This attribute defines the Date of Birth of a person. This should be in between January 1900 and January 2014.

**apptDate:** This attribute defines the Date when the appointment will take place. This value should be in between the current date (included) and 30 days after the current date (included).

**Key constraints are as follows:**

* The start date of a medication cannot be after the end date.
* A patient cannot be scheduled for multiple appointments at the same time on the same day.
* An appointment cannot be scheduled with a doctor who is not available on the given date and time.
* A doctor's availability cannot overlap with their scheduled appointments.
* A doctor’s specialty can be changed but not when they have any upcoming appointments already scheduled.
* A message must always have a body as a message without any content would be irrelevant.
* The patient can have multiple insurance entries as a person can have primary and secondary insurance carriers where the name of the insurance carriers can be the same.
* The policy number and group number of an insurance cannot be the same.
* A person must have a unique email, and phone number.
* An insurance must have a unique policy number.
* Information like email, first name, last name, and phone number should be set to Not Null as this information is critical for the health records.
* For the Doctor, the specialty should be set to Not Null as a Doctor must have some qualification.
* An appointment must have a date and a time and thus these values should also be set to Not Null.
* An Employee must contain a job title and thus this value should be set to Not Null.
* All entities must have a unique primary key i.e., their Id’s for e.g., doctor id, patient id, message id, etc.

**Queries**

The following queries will satisfy the project requirements:

1.)  Create an appointment using attributes as per the APPOINTMENTS schema.

2.) Add a Doctor ID and Patient ID to a record in the Appointment table, thereby adding a Doctor to an Appointment.

3.) Return names and contact information about all Persons assigned to an Appointment.  
4.) Find records of Immunizations, Medication, Insurance for the Patient of an Appointment.

5.) List names and contact information for Doctors of a given Specialty, Availability, and Time.

6.) Create a message using the attributes as per the MESSAGE schema.

7.) Modify an appointment record.

8.)  Add a Patient ID to a record in the Immunization table, thereby adding an Immunization to a patient.

9.) Add a Patient ID to a record in the Insurance table, thereby adding an Insurance to a patient.

10.) Add a Patient ID to a record in the Medication table, thereby adding a Medication to a patient.

11.) Return time, date, and location using the appointment\_id.

***Requirements Matrix***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Query | Req 1 | Req 2 | Req 3 | Req 4 | Req 5 | Req 6 | Req 7 | Req 8 |
| 1 | X |  |  |  |  |  |  |  |
| 2 |  |  | X |  |  |  |  |  |
| 3 | X |  | X |  |  |  |  |  |
| 4 |  |  | X |  | X |  |  |  |
| 5 |  |  |  | X |  |  |  |  |
| 6 |  |  |  |  |  |  |  | X |
| 7 |  | X |  |  |  |  |  | X |
| 8 |  |  |  |  |  | X |  |  |
| 9 |  |  |  |  |  | X | X |  |
| 10 |  |  |  |  |  | X | X |  |
| 11 |  | X | X |  |  |  |  |  |

***Entity-Relationship Diagram***

****

\*Note: Attributes that have two names (ex: start/end date) are two separate attributes, we just didn’t have enough space so we combined them in the er diagram. Additionally, in the availability table (top right), the attribute doctor\_id is cut off by the watermark.

**Implementation**

**Queries in SQL for Table Creation**

The person appointment table contains a constraint to make sure that the location is either in-person or online.

CREATE TABLE appointment (

appointment\_id INT(11) NOT NULL PRIMARY KEY AUTO INCREMENT,

patient\_id INT(11) NOT NULL,

doctor\_id INT(11) NOT NULL,

first\_name VARCHAR(50) NOT NULL,

appointment\_date DATE, NOT NULL,

start\_time TIME NOT NULL,

end\_date TIME NOT NULL,

location VARCHAR(10),

ADD CONSTRAINT location\_domain CHECK (location IN ("online", "in-person", "Online", "In-person"))

);

CREATE TABLE doctor (

doctor\_id INT(11) NOT NULL PRIMARY KEY,

primary\_email VARCHAR(50) NOT NULL,

secondary\_email VARCHAR(50) DEFAULT NULL,

FOREIGN KEY (doctor\_id) reference person(person\_id)

);

The doctor\_availability table contains a constraint to make sure that the end of their available is not before the start of their availability.

CREATE TABLE doctor\_availability (

availaiblity\_id INT(11) NOT NULL PRIMARY KEY AUTO INCREMENT,

doctor\_id DATE NOT NULL,

start\_time DATE NOT NULL,

end\_time DATE NOT NULL,

FOREGIN KEY (doctor\_id) REFERENCES doctor,

CONSTRAINT time\_check CHECK (start\_time < end\_time)

);

The employee table contains a constraint to make sure that the end of their start date is not after their end date.

CREATE TABLE employee (

employee\_id INT(11) NOT NULL PRIMARY KEY,

start\_date DATE NOT NULL,

end\_date DATE DEFAULT NULL,

job\_title VARCHAR(50) NOT NULL,

employee\_email VARCHAR(50) NOT NULL,

secondary\_email VARCHAR(50) DEFAULT NULL,

FOREIGN KEY (employee\_id) REFERENCES person(person\_id),

CONSTRAINT date\_check CHECK (start\_date < end\_date)

);

CREATE TABLE immunization (

immunization\_id INT(11) NOT NULL PRIMARY KEY AUTO INCREMENT,

patient\_id INT(11) NOT NULL,

vaccine\_name VARCHAR(50) NOT NULL,

immunization\_date DATE NOT NULL,

FOREIGN KEY (patient\_id) REFERENCES patient,

);

The insurance table contains a constraint to make sure the group number and policy number aren’t the same.

CREATE TABLE insurance (

insurance\_id INT(11) NOT NULL PRIMARY KEY AUTO INCREMENT,

patient\_id INT(11) NOT NULL,

insurance\_name VARCHAR(50) NOT NULL,

policy\_number VARCHAR(20) NOT NULL,

group\_number VARCHAR(20) NOT NULL,

FOREIGN KEY (patient\_id) REFERENCES patient,

ADD CONSTRAINT policy\_group\_chk CHECK (policy\_number != group\_number);

);

The medication table contains a constraint to make sure the end date isn’t before the start date.

CREATE TABLE medication (

medication\_id INT(11) NOT NULL PRIMARY KEY AUTO INCREMENT,

patient\_id INT(11) NOT NULL,

medication\_name VARCHAR(50) NOT NULL,

start\_time DATE NOT NULL,

end\_time DATE NOT NULL,

FOREIGN KEY (patient\_id) REFERENCES patient,

ADD CONSTRAINT end\_date\_chk CHECK (start\_date < end\_date)

);

CREATE TABLE message (

message\_id INT(11) NOT NULL PRIMARY KEY AUTO INCREMENT,

sender\_id INT(11) NOT NULL,

reciever\_id INT(11) NOT NULL,

title VARCHAR(50) DEFAULT NULL,

body TEXT NOT NULL

FOREIGN KEY (sender\_id)REFERENCES employee(employee\_id),

FOREIGN KEY (reciever\_id)REFERENCES patient(patient\_id)

);

CREATE TABLE patient (

patient\_id INT(11) NOT NULL PRIMARY KEY,

password\_hash binary(64) NOT NULL,

school\_email VARCHAR(50) NOT NULL,

FOREIGN KEY (patient\_id)REFERENCES person(person\_id)

);

The person table contains a constraint to check that the birthday is between 1900 and 2014.

CREATE TABLE person (

person\_id INT(11) NOT NULL AUTO INCREMENT,

first\_name VARCHAR(50) NOT NULL,

middle\_initial CHAR(1) DEFAULT NULL,

last\_name VARCHAR(50) NOT NULL,

birth\_date DATE NOT NULL,

CONSTRAINT birthday\_check CHECK (birth\_date BETWEEN "1900-01-01" AND "2014-01-31")

);

CREATE TABLE specialty (

doctor\_id INT(11) NOT NULL,

specialty VARCHAR(50) NOT NULL,

FOREIGN KEY (doctor\_id) references doctor,

PRIMARY KEY(doctor\_id, specialty)

);

CREATE TABLE telephone (

person\_id INT(11) NOT NULL,

telephone VARCHAR(15) NOT NULL,

FOREIGN KEY (person\_id) references person

PRIMARY KEY(person\_id, telephone)

)

**Constraint Test Data**

Appointment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **appointment\_id** | **patient\_id** | **doctor\_id** | **appointment\_date** | **start\_time** | **end\_time** | **location** |
| 11 | 21 | 1 | 2023-10-02 | 12:00:00 | 13:00:00 | Online |
| 12 | 21 | 1 | 2023-10-02 | 12:00:00 | 13:00:00 | online |
| 13 | 21 | 2 | 2023-10-02 | 14:00:00 | 15:00:00 | In-person |
| 14 | 23 | 2 | 2023-10-04 | 14:00:00 | 15:00:00 | in-person |

Medication

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **medication\_id** | **patient\_id** | **medication\_name** | **start\_time** | **end\_time** |
| 11 | 21 | Larin | 2023-06-01 | 2023-03-01 |
| 12 | 21 | Aspirin |  | 2023-06-01 |

Doctor\_Availabity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **availability\_id** | **doctor\_id** | **appointment\_date** | **start\_time** | **end\_time** |
| 31 | 2 | 2023-05-01 | 7:00:00 | 12:00:00 |
| 32 | 2 | 2023-05-01 | 5:00:00 | 13:00:00 |
| 33 | 2 | 2023-05-01 | 17:00:00 | 19:00:00 |
| 34 | 2 | 2023-05-01 | 18:00:00 | 17:00:00 |

Insurance

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **insurance\_id** | **patient\_id** | **name** | **policy\_number** | **group\_number** |
| 11 | 27 | NationWide | ABC123456 | ABC123456 |
| 12 | 28 | Geico | 123456789 | 123456789 |
| 13 | 28 | BlueCross Blue Shield | asdfghjkl | asdfghjkl |
| 14 | 30 | Progressive | ASDFGHJKL | ASDFGHJKL |

Person

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **person\_id** | **first\_name** | **middle\_name** | **last\_name** | **birth\_date** |
| 31 | Ram | Kumar | Rai | 1899-12-31 |
| 32 | Hari | Lal | Chettri | 2015-12-31 |

Employee

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **employee\_id** | **start\_date** | **end\_date** | **job\_title** | **employee\_email** | **secondary\_email** |
| 10 | 2023-06-01 | 2023-05-01 | Job | [employee@gmail.com](mailto:employee@gmail.com) | Null |
| 9 | 2023-06-01 | 2023-05-31 | Job | [employee@gmail.com](mailto:employee@gmail.com) | Null |

Telephone

|  |  |
| --- | --- |
| **person\_id** | **telephone** |
| 1 | 123456789 |
| 2 | 12345678 |

**Testing Procedure for Constraints**

|  |  |  |  |
| --- | --- | --- | --- |
| **Individual Steps Performed** | **Expected Result** | **Actual Results** | Req. # |
| Enter this sql statement to check for appointment location:    INSERT INTO `appointment` (`appointment\_id`, `patient\_id`, `doctor\_id`, `appointment\_date`, `start\_time`, `end\_time`, `location`) VALUES (NULL, '1234', '9876', '2023-04-30', '12:00:00', '13:00:00', 'On-line'); | Error due to wrong input for location | #4025 - CONSTRAINT `location\_domain` failed for `mhealth` . `appointment` | 1 |
| Checking doctor\_availability start\_constraint    INSERT INTO `doctor\_availability` (`doctor\_id`, `availability\_date`, `start\_time`, `end\_time`) VALUES (1, '2023-04-30', '06:00:00', '12:00:00') | Error due to availability  out-off range | #4025 - CONSTRAINT `start\_time\_domain` failed for `mhealth`.`doctor\_availability` | 6 |
| Checking doctor\_availability end\_constraint  INSERT INTO `appointment` (`patient\_id`, `doctor\_id`, `appointment\_date`, `start\_time`, `end\_time`, `location`) VALUES (1234, 9876, '2023-04-30', '12:00:00', '13:00:00', 'Online'); | Error due to availability  out-off range | #1644 - Doctor is not available at the given date and time. | 6 |
| Check person date of birth:  INSERT INTO `person` (`person\_id`, `first\_name`, `middle\_initial`, `last\_name`, `birth\_date`) VALUES (5, 'Sita', NULL, 'Rai', '1899-12-31') | Error due to date-of-birth being before 1900-01-01 | #4025 - CONSTRAINT `birth\_date\_domain` failed for `mhealth`.`person` | 7 |
| Checking if Policy Number and Group Number are equal:    INSERT INTO `insurance` (`patient\_id`, `name`, `policy\_number`, `group\_number`) VALUES (21, 'Ram', 1234567, 1234567) | Error due to same policy and group number | #4025 - CONSTRAINT `policy\_group\_chk` failed for `mhealth`.`insurance` | 5 |
| Check in medication:    INSERT INTO `medication` (`patient\_id`, `medication\_name`, `start\_date`, `end\_date`) VALUES (21, 'Advil', '2023-04-29', '2023-04-28') | Error due to start date is greater than end date | #4025 - CONSTRAINT `end\_date\_chk` failed for `mhealth`.`medication` | 6 |

**Queries in SQL for View Creation**

Two “helper view” were created to be called on from other views. This prevented repetitive code. The full\_name view concats a person first, middle (if applicable) and last name together and the telephone view formats a telephone number as (xxx) xxx-xxxx.

A total of 10 views were created. The code for 3 of these views is shown below.

This view shows current employee information, the same structure is also used for a view called prior employee info expect with a different WHERE clause. There is also a doctor\_info and patient\_info that has similar structure.

CREATE VIEW current\_employee\_info AS

SELECT

(SELECT `f`.`FullName`

FROM `mhealth`.`full\_name` `f`

WHERE `f`.`person\_id` = `mhealth`.`employee`.`employee\_id`

) AS `FullName`,

`mhealth`.`employee`.`job\_title` AS `JobTitle`,

CONCAT(

TIMESTAMPDIFF(

YEAR,

`mhealth`.`employee`.`start\_date`,

CURDATE()),

' year(s) and ',

TIMESTAMPDIFF(

MONTH,

`mhealth`.`employee`.`start\_date`,

CURDATE()) MOD 12,

' months(s)'

) AS `YearsWorked`,

`mhealth`.`employee`.`employee\_email` AS `EmployeeEmail`,

`mhealth`.`employee`.`secondary\_email` AS `SecondaryEmail`,

(SELECT `ft`.`formatted\_telephone`

FROM `mhealth`.`formatted\_telephone` `ft`

WHERE `ft`.`person\_id` = `mhealth`.`employee`.`employee\_id`

) AS `PhoneNumbers`

FROM

(`mhealth`.`employee`

JOIN `mhealth`.`person` `p`

ON (`p`.`person\_id` = `mhealth`.`employee`.`employee\_id`)

)

WHERE

`mhealth`.`employee`.`end\_date` IS NULL OR `mhealth`.`employee`.`end\_date` > CURDATE()

ORDER BY

`p`.`last\_name`, `mhealth`.`employee`.`start\_date`

This view shows if a person is a minor or not.

CREATE VIEW minor\_stats AS

SELECT

( SELECT `f`.`FullName`

FROM `mhealth`.`full\_name` `F`

WHERE f`.`person\_id` = `mhealth`.`person`.`person\_id`

) AS `FullName`,

`mhealth`.`person`.`birth\_date` AS `BirthDate`,

CASE WHEN TO\_DAYS(CURDATE()) - TO\_DAYS(`mhealth`.`person`.`birth\_date`) < 6570 THEN 'Minor' ELSE 'Adult'

END AS `Status`

FROM

`mhealth`.`person`

ORDER BY

`mhealth`.`person`.`last\_name`,

`mhealth`.`person`.`birth\_date`

This view displays messages

CREATE VIEW messages AS

SELECT

`mhealth`.`message`.`message\_id` AS `MessageID`,

(SELECT `fn`.`FullName`

FROM `mhealth`.`full\_name` `FN`

WHERE `fn`.`person\_id` = `mhealth`.`message`.`sender\_id`

) AS `SenderName`,

`E`.`job\_title` AS `SenderJobTitle`,

`mhealth`.`message`.`title` AS `Title`,

`mhealth`.`message`.`body` AS `Body`

FROM

(

`mhealth`.`message`

JOIN `mhealth`.`employee` `E`

ON (`E`.`employee\_id` = `mhealth`.`message`.`sender\_id`)

)

ORDER BY

`mhealth`.`message`.`message\_id`

**Queries in SQL for Trigger Creation**

A total of 6 unique triggers were created.

One issue that occurred while making constraints was that the CURDATE() function could not be used in check statements, so any constraints that needed to use CURDATE() was created as a trigger.

This trigger structure was used for the employee, immunization, and medication table.

CREATE TRIGGER {table name}\_date\_trigger

BEFORE INSERT OR UPDATE

ON {table name}

BEGIN

IF (NEW.start\_date NOT BETWEEN 1950-01-01 AND CURDATE())

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Invalid start date. Only dates

between January 1950 and the current date are allowed.';

END IF;

END

This trigger checks if a doctor is available when an appointment is made.

CREATE TRIGGER doctor\_availability\_trigger

BEFORE INSERT OR UPDATE

ON appointment

BEGIN

IF NOT EXISTS (

SELECT \*

FROM doctor\_availability

WHERE doctor\_id = NEW.doctor\_id

AND `availability\_date` = NEW.appointment\_date

AND ((NEW.start\_time BETWEEN start\_time AND end\_time)

AND (NEW.end\_time BETWEEN start\_time AND end\_time))

) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Doctor is not available at the given date and time.';

END IF;

END

This trigger prevents a doctor from making two availabilities at the same time frame.

CREATE TRIGGER prevent\_availability\_overlap\_trigger

BEFORE INSERT OR UPDATE

ON doctor\_availability

BEGIN

DECLARE overlap\_count INT;

SELECT COUNT(\*) INTO overlap\_count

FROM doctor\_availability

WHERE doctor\_id = NEW.doctor\_id

AND availability\_date = NEW.availability\_date

AND start\_time < NEW.end\_time

AND end\_time > NEW.start\_time;

IF overlap\_count > 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'The availability time overlaps with an existing availability.';

END IF;

END

This trigger prevents a patient from booking two appointments at the same time.

CREATE TRIGGER prevent\_double\_booking \_trigger

BEFORE INSERT OR UPDATE

ON appointment

BEGIN

DECLARE num\_appointments INTEGER;

SELECT COUNT(\*) INTO num\_appointments

FROM appointment

WHERE patient\_id = NEW.patient\_id

AND `appointment\_date` = NEW.`appointment\_date`

AND start\_time <= NEW.end\_time

AND end\_time >= NEW.start\_time;

IF num\_appointments > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot make two appointments during the same time frame and date.';

END IF;

END

**Triggers Test Data**

Appointment – Time Out of Bounds

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **appointment\_id** | **patient\_id** | **doctor\_id** | **appointment\_date** | **start\_time** | **end\_time** | **location** |
| 1 | 32 | 1 | 2023-10-08 | 6:00:00 | 10:00:00 | in-person |
| 2 | 54 | 4 | 2023-10-07 | 6:00:00 | 10:00:00 | online |
| 3 | 11 | 6 | 2023-10-09 | 10:00:00 | 19:00:00 | in-person |
| 4 | 42 | 8 | 2023-10-06 | 14:00:00 | 20:00:00 | online |
| 5 | 77 | 10 | 2023-10-04 | 5:00:00 | 10:00:00 | in-person |

Appointment – Double Booking

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **appointment\_id** | **patient\_id** | **doctor\_id** | **appointment\_date** | **start\_time** | **end\_time** | **location** |
| 1 | 10 | 2 | 2023-10-05 | 9:00:00 | 10:00:00 | in-person |
| 2 | 3 | 5 | 2023-10-05 | 9:00:00 | 10:00:00 | in-person |
| 3 | 23 | 9 | 2023-10-16 | 12:00:00 | 13:00:00 | in-person |
| 4 | 66 | 3 | 2023-10-16 | 12:00:00 | 13:00:00 | in-person |
| 5 | 19 | 11 | 2023-10-16 | 12:00:00 | 13:00:00 | in-person |

Employee – Date Out Bounds

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **employee\_id** | **start\_date** | **end\_date** | **job\_title** | **employee\_email** | **secondary\_email** |
| 28 | 1935-11-22 | 1936-04-28 | Medical Records Clerk | [xavierturner20@email.com](mailto:xavierturner20@email.com) | [xavierturner202@email.com](mailto:xavierturner202@email.com) |
| 24 | 1927-07-03 | 1941-10-16 | Medical Billing Specialist | [annabellesanchez19@email.com](mailto:annabellesanchez19@email.com) | [annabellesanchez192@email.com](mailto:annabellesanchez192@email.com) |
| 18 | 1943-05-11 | 1946-01-07 | Medical Assistant | [jonathanperez42@email.com](mailto:jonathanperez42@email.com) | [jonathanperez422@email.com](mailto:jonathanperez422@email.com) |
| 20 | 1948-09-12 | 1949-12-05 | Receptionist | [mackenziebarnes67@email.com](mailto:mackenziebarnes67@email.com) | [mackenziebarnes672@email.com](mailto:mackenziebarnes672@email.com) |
| 25 | 1932-02-19 | 1942-11-14 | Medical Coder | [johncampbell54@email.com](mailto:johncampbell54@email.com) | [johncampbell542@email.com](mailto:johncampbell542@email.com) |
| 12 | 1945-06-24 | 1949-03-25 | Medical Transcriptionist | [veronicabrooks23@email.com](mailto:veronicabrooks23@email.com) | [veronicabrooks232@email.com](mailto:veronicabrooks232@email.com) |
| 17 | 1949-01-08 | 1949-02-18 | Medical Assistant | [kevinscott89@email.com](mailto:kevinscott89@email.com) | [kevinscott892@email.com](mailto:kevinscott892@email.com) |
| 15 | 1939-12-31 | 1940-09-02 | Receptionist | [victoriaroberts76@email.com](mailto:victoriaroberts76@email.com) | [victoriaroberts762@email.com](mailto:victoriaroberts762@email.com) |
| 22 | 1947-03-17 | 1948-08-01 | Medical Coder | [christopherrodriguez28@email.com](mailto:christopherrodriguez28@email.com) | [christopherrodriguez282@email.com](mailto:christopherrodriguez282@email.com) |
| 14 | 1930-08-09 | 1938-12-19 | Medical Transcriptionist | [madisonmiller94@email.com](mailto:madisonmiller94@email.com) | [madisonmiller942@email.com](mailto:madisonmiller942@email.com) |

Immunization – Date Out of Bounds

|  |  |  |  |
| --- | --- | --- | --- |
| **immunization\_id** | **patient\_id** | **vaccine\_name** | **immunization\_date** |
| 1 | 34 | Tdap | 1949-11-14 |
| 2 | 43 | HPV | 1941-10-16 |
| 3 | 45 | MMR | 1946-01-07 |
| 4 | 22 | Hepatitis A | 1949-11-05 |
| 5 | 41 | Hepatitis B | 1942-11-14 |
| 6 | 28 | Varicella | 1931-03-25 |
| 7 | 31 | Influenza | 1940-02-18 |
| 8 | 47 | Pneumococcal | 1937-09-02 |
| 9 | 21 | Meningococcal | 1948-08-01 |
| 10 | 37 | Shingles | 1938-12-19 |

Doctor\_Availability – Time Out of Bounds

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **availability\_id** | **doctor\_id** | **availability\_date** | **start\_time** | **end\_time** |
| 1 | 1 | 2022-10-03 | 09:00:00 | 13:00:00 |
| 2 | 1 | 2022-10-03 | 09:00:00 | 13:00:00 |
| 3 | 1 | 2022-10-07 | 08:00:00 | 12:00:00 |
| 4 | 1 | 2022-10-07 | 08:00:00 | 12:00:00 |
| 5 | 3 | 2022-10-05 | 14:00:00 | 18:00:00 |
| 6 | 2 | 2022-10-07 | 11:00:00 | 15:00:00 |
| 7 | 3 | 2022-10-03 | 11:00:00 | 15:00:00 |
| 8 | 3 | 2022-10-05 | 14:00:00 | 18:00:00 |
| 9 | 4 | 2022-10-03 | 09:00:00 | 13:00:00 |
| 10 | 4 | 2022-10-03 | 09:00:00 | 13:00:00 |

Medication – Date Out of Bounds

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **medication\_id** | **patient\_id** | **medication\_name** | **start\_date** | **end\_date** |
| 1 | 23 | Aspirin | 1935-11-22 | NULL |
| 2 | 27 | Tylenol | 1927-07-03 | NULL |
| 3 | 18 | Advil | 1943-05-11 | 2023-07-13 |
| 4 | 25 | Lipitor | 1948-09-12 | 2023-04-10 |
| 5 | 16 | Zoloft | 1932-02-19 | NULL |
| 6 | 12 | Prozac | 1945-06-24 | 2023-05-12 |
| 7 | 29 | Celebrex | 1949-01-08 | 2023-01-11 |
| 8 | 19 | Nexium | 1939-12-31 | 2023-05-17 |
| 9 | 11 | Zyrtec | 1947-03-17 | 2023-04-09 |
| 10 | 21 | Claritin | 1930-08-09 | 2023-04-21 |

**Testing Procedure for Triggers**

|  |  |  |  |
| --- | --- | --- | --- |
| Individual Steps Performed | Expected Result | Actual Results | Requirements |
| Inserting into the immunization table with incorrect start date  INSERT INTO `immunization` (`immunization\_id`, `patient\_id`, `vaccine\_name`, `immunization\_date `) VALUES (1, '34', '9876', ‘Tdap’, '1949-11-14') | Will cause an error as the date taken for the immunization is before 1950 which should be detected in the insert trigger | #1644 – Invalid immunization date. Only dates between January 1950 and the current data are allowed | 5 |
| Updating one of the records in the immunizations table with incorrect start date  UPDATE immunization  SET immunization\_date = `1949-11-14` WHERE immunization\_id = 1; | Will cause an error as the date taken for the immunization is before 1950 which should be detected in the update trigger | #1644 – Invalid immunization date. Only dates between January 1950 and the current data are allowed | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Individual Steps Performed** | **Expected Result** | **Actual Results** | **Requirements** |
| Inserting into the medication table with incorrect dates for the medication  INSERT INTO medication (patient\_id, medication\_name, start\_date, end\_date)  VALUES (23, "Aspirin", "1935-11-22", NULL); | Will cause an error as the start date for the medication is before 1950 which should be detected in the insert trigger | #1644 – Invalid start date. Only dates between January 1950 and the current date are allowed | 5 |
| Updating a record in the medication table with incorrect start and end dates.  UPDATE medication SET start\_date = `1935-11-22`  WHERE medication\_id = 1; | Will cause an error as the start date for the medication is before 1950 which should be detected in the update trigger | #1644 – Invalid start date. Only dates between January 1950 and the current date are allowed | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Individual Steps Performed** | **Expected Result** | **Actual Results** | **Requirements** |
| Inserting into the appointment table with date that is out of bounds  INSERT INTO `appointment` (`appointment\_id`, `patient\_id`, `doctor\_id`, `appointment\_date`, `start\_time`, `end\_time`, `location`) VALUES (1, '32', '1', '2023-10-08', '6:00:00', '10:00:00', 'in-person'); | Will cause an error as the start time of the appointment is before 08:00:00 AM which should be detected in the insert trigger | #1644 – Doctor is not available at the given date and time | 1 |
| Updating a record in the appointment table with date that is out of bounds  UPDATE appointment  SET start\_time = 6:00:00  WHERE appointment\_id = 1; | Will cause an error as the start time of the appointment is before 08:00:00 AM which should be detected in the update trigger | #1644 – Doctor is not available at the given date and time | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Individual Steps Performed** | **Expected Result** | **Actual Results** | **Requirements** |
| Inserting into the appointment table with date, start, and end times similar to one of the records already present in the table    INSERT INTO `appointment` (`appointment\_id`, `patient\_id`, `doctor\_id`, `appointment\_date`, `start\_time`, `end\_time`, `location`) VALUES (4, '66', '3', '2023-10-16', '12:00:00', '13:00:00', 'in-person'); | Should cause an error as this insert trigger would detect double bookings i.e., more than one booking about to take place on the same date, start, and end times | Does not cause an error and the query gets accepted which is unusual behavior. This was caused by a missing string in the trigger which was fixed | 1 |
| Updating a record in the appointments table with date, start, and end times similar to one of the records already present in the table    UPDATE appointment  SET appointment\_date = `2023-10-16`, start\_time = `12:00:00`, end\_time = `13:00:00`  WHERE appointment\_id = 1; | Should cause an error as this update trigger would detect double bookings i.e., more than one booking about to take place on the same date, start, and end times | Does not cause an error and the query gets accepted which is unusual behavior. This was caused by a missing string in the trigger which was fixed. | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Individual Steps Performed** | **Expected Result** | **Actual Results** | **Requirements** |
| Inserting into the employee table with incorrect start and end dates.  INSERT INTO `employee` (`employee\_id`, `start\_date`, `end\_date`, `job\_title`, `primary\_email`, `secondary\_email`) VALUES,  (28, '1935-11-22', '1936-04-28', Medical Records Clerk, 'xavierturner20@email.com', ' xavierturner202@email.com '); | Will cause an error as the start date for the employee is before 1950 which should be detected in the insert trigger | #1644 – Invalid start date. Only dates between January 1950 and the current date are allowed | 3 |
| Updating a record in the employee table with incorrect start and end dates.  UPDATE employee  SET start\_time = `1927-07-03`, end\_time = `1941-10-16`  WHERE employee\_id = 24; | Will cause an error as the start date for the employee is before 1950 which should be detected in the update trigger | #1644 – Invalid start date. Only dates between January 1950 and the current date are allowed | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Individual Steps Performed** | **Expected Result** | **Actual Results** | **Requirements** |
| Inserting into the doctor\_availability table with date, start, and end times similar to one of the records already present in the table.  INSERT INTO `doctor\_availability` (`availability\_id`, `doctor\_id`, `availability\_date`, `start\_time`, `end\_time`) VALUES,  (2, 1, '2022-10-03', '09:00:00', '13:00:00') | Will cause an error as this would cause a doctor to be available at two times which is not possible. | #1644 - The availability time overlaps with an existing availability | 4 |
| Updating a record in the doctor\_availability table with date, start, and end times similar to one of the records already present in the table.  UPDATE doctor\_availability  SET availability\_date = `2022-10-07`, start\_time = `08:00:00`, end\_time = `12:00:00`  WHERE availability\_id = 3; | Will cause an error as this would cause a doctor to be available at two times which is not possible. | #1644 - The availability time overlaps with an existing availability | 4 |