**General Clinic Database Management (GCDM)**

**Final report:**

I. **Abstract**: The General Clinic Database Management (GCDM) propose is to automate the current manual system. The current system requires tremendous paper forms, with data stored in spreadsheet throughout the clinic management structure. The GCDM will consist of many staffs in the clinic. Stuff members can only be receptionists, nurses, or doctors. The GCDM can also have 50 patients at time, with 50 rooms dedicated for any patient, and those rooms are superintended by nurses. A doctor can attend many patients and the receptionist maintains the record of every patient. Invoice are created base on treatments and medications assigned to the patient.

II. **Mission** **statement**: The goal of the General Clinic Database Management (GCDM) is to maintain patient’s information, by facilitating the distributions of information between doctor, staff’s member’s, including doctor’s appointments, patient’s prescription and invoice system.

III. **Mission** **objectives**:

The objectives of this database are:

To maintain (enter, update, and delete) data on Patients

To maintain (enter, update, and delete) data on Staffs

To maintain (enter, update, and delete) data on Doctors

To maintain (enter, update, and delete) data on Rooms

To maintain (enter, update, and delete) data on Medications

To maintain (enter, update, and delete) data on Records

To perform searches on Patients

To perform searches on Staffs

To perform searches on Doctors

To perform searches on Rooms

To perform searches on Medications

To perform searches on Records

To track the status of any Patients

To track the status of any Rooms

To track the status of any Medications

To report on Patients

To report on Staffs

To report on Doctors

To report on Rooms

To report on Medications

To report on Record

IV. **Major user views:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data | Access Type | Doctor | Nurse | Receptionists | Managers |
|  |  |  |  |  |  |
|  | Maintain |  |  | X | X |
| Patients | Query | x | X | X | X |
|  | Report | x | X | X | X |
|  |  |  |  |  |  |
|  | Maintain |  |  |  | X |
| Stuffs | Query |  |  |  | X |
|  | Report |  |  |  | X |
|  |  |  |  |  |  |
|  | Maintain |  |  |  | X |
| Doctors | Query |  | X |  | X |
|  | Report |  | X |  | X |
|  |  |  |  |  |  |
|  | Maintain |  | X |  |  |
| Rooms | Query |  | X |  |  |
|  | Report |  | X | X |  |
|  |  |  |  |  |  |
|  | Maintain |  |  |  |  |
| Records | Query |  |  |  | X |
|  | Report | X | X | X | X |
|  |  |  |  |  |  |
|  | Maintain |  |  |  | X |
| Nurse | Query |  |  |  | X |
|  | Report |  |  | X | X |
|  |  |  |  |  |  |
|  | Maintain |  |  |  | X |
| Receptionist | Query |  |  |  | X |
|  | Report |  |  |  | X |
|  |  |  |  |  |  |
|  | Maintain |  | X |  |  |
| Medications | Query | X |  |  |  |
|  | Report | X | X |  |  |

V- **E/R Diagram**

* **Patient table:** The Keys attribute of the patient table is the ***patient\_id*** representing the primary keys of the patient table. The patient has a 1 to 1 relationship with the check-in table, where ***patient\_id*** is also located inside the check-in table as the foreign key. As for constraint, the patient table can only support 50 patients at time. The patient table also have a relationship with the prescription table via the check-in table with a 1 to many relationship, where ***patient\_id*** is added inside the prescription table as the primary keys
* **Stuff table:** The keys attribute of the stuff table is the primary keys ***stuff\_id*** and unique key ***Stuff\_tiltle***. The stuff has a 1 to many relationships with the check-in table where one doctor (define by the title) can manage multiple patient and ***stuff\_id*** defined as the foreign key inside checking table. The ***stuff\_id*** also have a 1 to many relationships with the room table, where one nurse (define by the title) can manage multiple patient. the ***stuff\_id*** is defined as foreign keys inside the room table.
* **Room table:** The keys attribute of the room table is the ***room\_id*** representing the primary key of the room table. The maximum number of room at the time is 50. The room have a 1 to 1 relationship with the checking table and also a many room to one nurse.
* **Check-in table:** The keys attribute of the check-in table is the ***C\_id*** representing the primary key of the check-in table, with ***patient\_id***, ***stuff\_id***, ***room\_id*** as the foreign keys representing each patient table, stuff table, and room table. The checking table have a one to many relationships with the prescription table with ***C\_id***as the foreign keys inside the prescription table. The checking table have a one to many relationships with the patient table, stuff table, and room table.
* **Medication table:** The keys attribute of the medication table is the ***med\_id*** representing the primary key of the medication table. The medication table have a many to many relationships with the prescription table, where the ***med\_id***is the foreign key inside the prescription table
* **Prescription table:** The keys attribute of the prescription table is the ***Prescription\_id*** representing the primary key of the prescription table. Where we have the ***stuff\_id*** as the foreign key of the staff table and ***med\_id*** as the foreign keys of the med id

**Summary:**

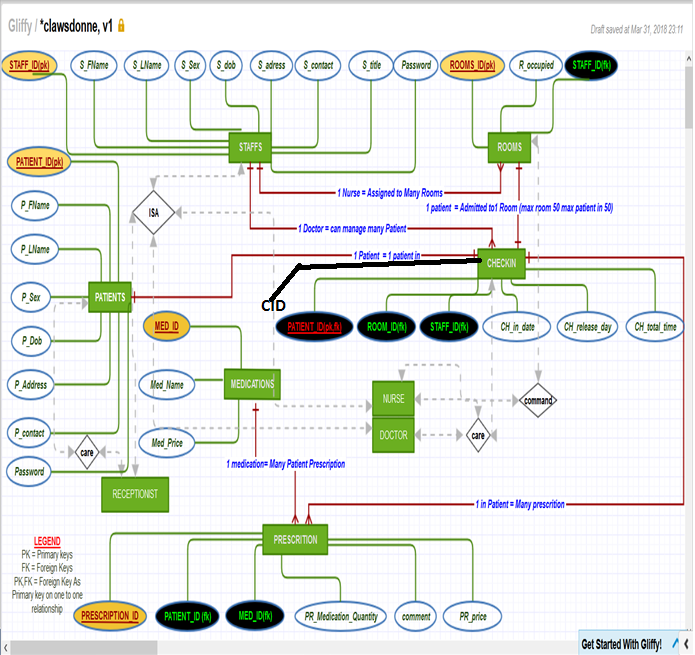
1.Each patient have to paid invoice “1 patient to 1 invoice”

2. Each patient is added to a room “1 room to 1 patient” max room 50, max patient 50

3. A room can be assign to many nurse “1 room for many nurse”

4. A doctor can attend many patient “1 doctor for many patient”

5. a receptionist take care of many record “1 receptionist for many record”

––

**Room Trigger to Limit Max number of rooms after INSERT:**

ALTER trigger [dbo].[Limitroomto50]

on [dbo].[Room]

after insert

as

declare @roomCount int

select @roomCount = Count(\*)

from room

if @roomCount > 50

begin

rollback

end

**Check-in Trigger to Limit Max number of checking patient after UPDATE:**

ALTER trigger [dbo].[Limitcheckinpatiento50]

on [dbo].[checkin]

after update

as

declare @patientCount int

select @patientCount = Count(\*)

from checkin

if @patientCount > 50

begin

rollback

end

VI**- Relational Model** BCNF verification

**Patient Table:**

patient\_id P\_FName P\_LName P\_Sex P\_Dob P\_Address P\_contact

PN001 Tyshawn Benson M 1986-12-12 298 Roehampton St. Port Chester, NY 10573 (251) 546-9442

PN002 Elise Cardenas F 1987-11-01 8995 North Pennsylvania St. (949) 569-4371

**1NF:**

* **The primary key of the patient table is PATIENT\_ID.**
* **No repetition of the key inside the patient table**
* **No duplicated rows in the patient table (Table has a Primary key “PAYIENT\_ID”)**
* **Each record is also unique**
* Each attribute of a Patient table only has an atomic value

**2NF:**

* **All non-primary key (“**P\_FName P\_LName P\_Sex P\_Dob P\_Address P\_contact”) **attribute is fully functional depend on the primary key (“Patient\_ID”)**
* **And all** non-prime attribute is dependent on the proper subsets of any candidate’s key of Patient Table

**3NF:**

* **All non-primary key (“**P\_FName P\_LName P\_Sex P\_Dob P\_Address P\_contact”)attribute is transitively dependent on the primary key **(“Patient\_ID”)**

**BCNF:**

* **Every** determinant **(“**P\_FName P\_LName P\_Sex P\_Dob P\_Address P\_contact”)is a candidate key **base on the primary key (“Patient\_ID”)**

**Stuff Table:**

([staff\_id], [S\_FName], [S\_LName], [S\_sex], [S\_Dob], [S\_Address], [S\_Contact], [S\_Title], [S\_salary]) S100 Trinity Yang M 1979-11-01 8387 West Blackburn St. Montclair, NJ 07042 (630) 446-8851 Receptionist 8769.00

S101 Dax Chung M 1986-12-12 298 Roehampton St. Port Chester, NY 10573 (251) 546-9442 Nurse 10623.00

S102 Dorian Andersen M 1987-11-01 8995 North Pennsylvania St. (949) 569-4371 Doctor 21000.00

**1NF:**

* **The primary key of the stuff table is STAFF\_ID.**
* **No repetition of the key inside the patient table**
* **No duplicated rows in the stuff table (Table has a Primary key “STAFF\_ID”)**
* **Each record is also unique**
* Each attribute of a **stuff** table only has an atomic value

**2NF:**

* **All non-primary key (**[S\_FName], [S\_LName], [S\_sex], [S\_Dob], [S\_Address], [S\_Contact], [S\_Title], [S\_salary]) **attribute is fully functional depend on the primary key (“STAFF \_ID”)**
* **And all** non-prime attribute is dependent on the proper subsets of any candidate’s key of Staff Table

**3NF:**

* **All non-primary key (**[S\_FName], [S\_LName], [S\_sex], [S\_Dob], [S\_Address], [S\_Contact], [S\_Title], [S\_salary]) attribute is transitively dependent on the primary key **(“STAFF \_ID”)**

**BCNF:**

* **Every** determinant ([staff\_id], [S\_FName], [S\_LName], [S\_sex], [S\_Dob], [S\_Address], [S\_Contact], [S\_Title], [S\_salary])is a candidate key **base on the patient (“STAFF\_ID”)**

**Room Table:**

room\_id , R\_occupied\_by , staff\_id

R201 P101 S103

R202 NULL S106

R203 NULL NULL

**1NF:**

* **The primary key of the** room **table is** ROOM **\_ID.**
* **No repetition of the key inside the patient table**
* **No duplicated rows in the** room **table (Table has a Primary key “**room **\_ID”)**
* **Each record is also unique**
* Each attribute of a roomtable only has an atomic value
* **Each record is also unique**

**2NF:**

* **All non-primary key (**R\_occupied\_by , staff\_id) **attribute is fully functional depend on the primary key (“**ROOM **\_ID”)**
* **And all** non-prime attribute is dependent on the proper subsets of any candidate’s key of Room Table

**3NF:**

* **All non-primary key (**R\_occupied\_by , staff\_id) attribute is transitively dependent on the primary key **(“**ROOM **\_ID”)**

**BCNF:**

* **Every** determinant **(**R\_occupied\_by , staff\_id) **is** a candidate key **base on the patient (“**ROOM **\_ID”)**
* Staff: only determinant is Stuf\_ID

**Checkin Table:**

(“checkin\_id”, “patient\_id”, “room\_id”, “stuff\_id”,“ch\_in\_date”)

12 PN001 R200 S102 2018-10-01

13 PN002 R201 S104 2018-10-01

14 PN003 R202 S102 2018-10-01

**1NF:**

* **The primary key of the** checkin **table is** checkin **\_ID.**
* **No repetition of the key inside the** checkin **table**
* **No duplicated rows in the** checkin **table (Table has a Primary key “**checkin **\_ID”)**
* **Each record is also unique**
* Each attribute of a checkintable only has an atomic value
* **Each record is also unique**

**2NF:**

* **All non-primary key** (“checkin\_id”, “patient\_id”, “room\_id”, “stuff\_id”,“ch\_in\_date”)
* **Attribute is fully functional depend on the primary key (“**checkin **\_ID”)**
* **And all** non-prime attribute is dependent on the proper subsets of any candidate’s key of Room Table

**3NF:**

**All non-primary key** (“checkin\_id”, “patient\_id”, “room\_id”, “stuff\_id”,“ch\_in\_date”)

* attribute is transitively dependent on the primary key **(“**checkin **\_ID”)**

**BCNF:**

**Every** determinant (“checkin\_id”, “patient\_id”, “room\_id”, “stuff\_id”,“ch\_in\_date”)

* **is** a candidate key **base on the patient (“**checkin **\_ID”)**
* Staff: only determinant is (Stuf\_ID , Stuff\_tiltle)
* Patient: only determinant is (Patient\_ID)
* Room: only determinant is (Room\_ID)

**Prescription Table:**

(“prescription\_id”, “patient\_id”, “med\_id”, “checkin\_id”, “comment”, “PR\_med\_quantity”, “PR\_price”)

20 PN001 16 12 An abdominal aortic aneurysm (AAA) 2 16.00

21 PN001 17 12 Roseola 3 8.70

**1NF:**

* **The primary key of the** prescription **table is** prescription **\_ID.**
* **No repetition of the key inside the patient table**
* **No duplicated rows in the** prescription **table (Table has a Primary key “**prescription **\_ID”)**
* **Each record is also unique**
* Each attribute of a prescription table only has an atomic value
* **Each record is also unique**

**2NF:**

* **All non-primary key** (“patient\_id”, “med\_id”, “checkin\_id”, “comment”, “PR\_med\_quantity”, “PR\_price”)
* **attribute is fully functional depend on the primary key (“**prescription **\_ID”)**
* **And all** non-prime attribute is dependent on the proper subsets of any candidate’s key of Room Table

**3NF:**

* **All non-primary key** ( “patient\_id”, “med\_id”, “checkin\_id”, “comment”, “PR\_med\_quantity”, “PR\_price”)
* attribute is transitively dependent on the primary key **(“**prescription **\_ID”)**

**BCNF:**

* **Every** determinant (“patient\_id”, “med\_id”, “checkin\_id”, “comment”, “PR\_med\_quantity”, “PR\_price”)
* **is** a candidate key **base on the patient (“**prescription **\_ID”)**
* medication: only determinant is (med\_ID)
* Patient: only determinant is (Patient\_ID)
* checkin: only determinant is (checkin\_ID)

**Mediction Table:**

(“med\_id”, “med\_name”, “med\_price”)

2017 Diclofenac Sodium 50 mg Tab 2.90

2018 Calamine Lotion 11.00

2019 Chlorhe Wash 19.52

2020 Chlorhexidine Gluconate 0.2% Mouth Wash 19.00

2021 Fusidic Acid 2 % w/v Cream 12.24

**1NF:**

* **The primary key of the medication table is** med **\_ID.**
* **No repetition of the key inside the patient table**
* **No duplicated rows in the medication table (Table has a Primary key “**med **\_ID”)**
* **Each record is also unique**
* Each attribute of a Patient table only has an atomic value

**2NF:**

* **All non-primary key** (“med\_name”, “med\_price”) **attribute is fully functional depend on the primary key (“**med **\_ID”)**
* **And all** non-prime attribute is dependent on the proper subsets of any candidate’s key of Patient Table

**3NF:**

* **All non-primary key (**“med\_name”, “med\_price”) attribute is transitively dependent on the primary key **(“**med **\_ID”)**

**BCNF:**

* **Every (**“med\_name”, “med\_price”) is a candidate key **base on the primary key (“Patient\_ID”)**

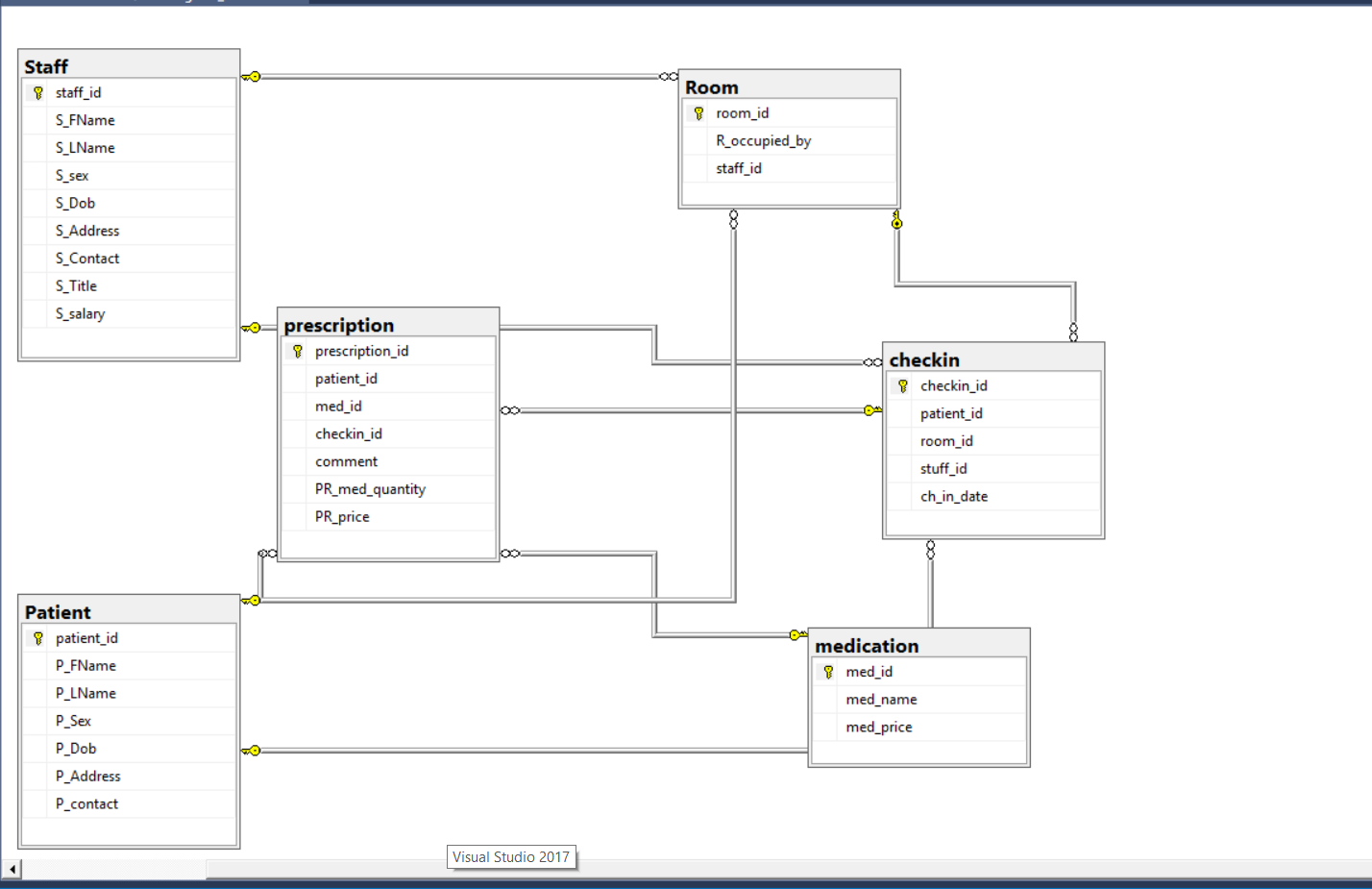


TABLE PRIMARY KEY FOREIGN KEY

STAFF staff\_id none

PATIENT patient\_id none

ROOM room\_id staff\_id

MEDICATION med\_id none

PRESCRIPTION prescription\_id patient\_id, med\_id

CHECKIN patient\_id patient\_id, room\_id, stuff\_id

VII **Complete Database Functionalities:**

A). Actor List:

* Receptionist
* Nurse
* Doctor
* Manager
* Patient

B). A list of uses cases:

I- RECEPTIONIST:

Case 1: Insert operation (enter patient to the patient table)

Use case name: Enter a new patient

Actor: Receptionist

Steps:

1. Actor clicks on “Enter New Patient” button;

2. A new patient form tab open with predefined patient (auto increment) id is shown;

3. Prompt to enter patient name, sex, DOB, address, contact number;

4. All information is displayed; ask for confirmation

5. User clicks on “Confirm” button

Case 2: Insert operation (Assign nurse a room)

Use case name: Assign nurse to a room

Actor: Receptionist

Steps:

1. Actor clicks on “assign Rooms” button;

2. A new patient form tab open with stuff id box and predefined room id;

3. Prompt to enter Stuff Id;

4. All information is displayed; ask for confirmation

5. User clicks on “Confirm” button

Case 2: Join operation (join the stuff table and room table to see which nurse are assigned to which room)

Use case name: Room View Report (list of room and stuff who govern the room)

Actor: Receptionist

Steps:

1. Actor clicks on “View room Report” button;

2. A room form open with search Box;

3. Prompt to enter room Id or stuff id;

4. All information is displayed; ask for confirmation

5. User clicks on “Confirm” button

Case 3: Join, group and aggregate operation (aggregate and calculate the patient bill, by joining the prescription table to the patient table then group then using his id then calculate the grand total for each patient)

Use case name: Patient Unpaid Prescription Report

Actor: Receptionist

Steps:

1. Actor clicks on “Unpaid Patient Prescription” button;

2. A new patient form tab open with search Box;

3. Prompt to enter Patient Id;

4. All information is displayed; ask for confirmation

5. User clicks on “Confirm” button

Case 4: Delete/Update operation (remove or update patient from the patient table)

Use case name: Update/Remove Patient

Actor: Receptionist

Steps:

1. Actor clicks on “Update/Remove Patient” button;

2. A new patient form tab open with search Box;

3. Prompt to enter Patient Id;

4. All information is displayed; ask for update or remove

5. user input new info or select remove box

6. User clicks on “Confirm” button

II- NURSE:

Case 1: Update operation (Update patient and doctor to a room via the check-in table)

Use case name: assign patient and doctor to a room

Actor: Nurse

Steps:

1. Actor clicks on “Assign Patient / Doctor to a Room” button;

2. A new form open with Patient Id box, doctor Id box, check in date and room combo box (loaded with only empty room);

3. Prompt to enter Patient Id box, doctor Id box and select a room from the list;

4. All information is displayed; ask for confirmation; Ask for confirmation

5. A new form open with Patient Id, doctor Id, the room selected;

6. Prompt to enter patient admission date, and set the room as occupied;

7. All information is displayed; ask for confirmation

8. User clicks on “Confirm” button

Case 2: update Operation (change the current doctor from the check-in table to another room)

Use case name: Update the Doctor who currently seeing the patient to a new one

Actor: Nurse

Steps:

1. Actor clicks on “Update check in doctor” button;

2. Prompt to enter Patient Id, doctor Id box and id;

3. Do the Id’s match; Ask for confirmation

4. The new form open for that specific with the request;

5. Prompt to enter new Doctor id;

6. All information is displayed; ask for confirmation

7. User clicks on “Confirm” button

Case 3: Join and aggregate (view the list of admitted patient and total number of admitted patient, by combining the check-in patient with the patient table then count the total number of admitted patient)

Use case name: Check for the list and number of admitted patient

Actor: Nurse

Steps:

1. Actor clicks on “: Check / Number of admitted patient” button;

2. New form open with the Total number of available and occupied room;

3. Prompt to click on the number of room to check the list of occupied or available room;

4. A Room form open base on the Group selected;

5. User clicks on “Exit” button

Case 4: Delete operation (Remove a patient from the check-in table by using the delete operation and set to not cascade)

Use case name: Remove patient from a room

Actor: Nurse

Steps:

1. Actor clicks on “Remove Admitted Patient” button;

2. Prompt to enter Patient Id, and room id;

3. Ask for confirmation

4. New admitted patient form open;

5. Prompt to click on delete to remove the patient from admitted list;

6. Are you want to discharge the patient; ask for confirmation

7. User clicks on “Confirm” button

III- DOCTOR:

Case 1: insert operation (the doctor creates a prescription by using the insert operation )

Use case name: create Prescription

Actor: Doctor

Steps:

1. Actor clicks on “Create Prescription” button;

2. A new prescription form open with auto increment prescription number;

3. Prompt to enter patient id, Medication id, medication quantity number and comment;

4. All information is displayed; ask for confirmation

5. User clicks on “Confirm” button

Case 2: Join and update operation (to update the prescription table the doctor need to join the prescription table with the medication and project the require field then insert the desire value)

Use case name: Update Prescription

Actor: Doctor

Steps:

1. Actor clicks on “Update Prescription” button;

2. Prompt to enter prescription id;

3. All information is displayed; ask for confirmation

4. User clicks on “Confirm” button

5. A prescription form with patient id, Medication id, medication quantity number, comment and total cost;

6. Ask the user to select the field to change and input the new value into it;

7. All information is displayed; ask for confirmation

8. User clicks on “Confirm” button

Case 3: join group and aggregate operation (to view the prescription record list, the doctor need to join the patient table with prescription then retrieve the patient need needed them group by the patient name, then calculate the total price of all prescription assign to that customer)

Use case name: view Prescription report

Actor: Doctor

Steps:

1. Actor clicks on “Prescription report” button;

2. Prompt to enter patient id;

3. All information is displayed; ask for confirmation

4. User clicks on “Confirm” button

5. A prescription form open with all prescription assign to the patient id and the total number;

7. All information is displayed; ask for confirmation

8. User clicks on “Confirm” button

Case 4: Delete operation (to delete prescription the diction just need to use the command to delete)

Use case name: Delete Prescription

Actor: Doctor

Steps:

1. Actor clicks on “Delete Prescription” button;

2. Prompt to enter prescription id to delete;

3. All information is displayed; ask for confirmation

4. User clicks on “Confirm” button

IV- MANAGER:

Case 1: Insert operation (use the insert command to insert a new stuff to the stuff table)

Use case name: Enter new stuff

Actor: Manager

Steps:

1. Actor clicks on “Enter new stuff” button;

2. A new stuff form tab open with predefined stuff (auto increment) id is shown;

3. Prompt to enter stuff name, sex, DOB, address, contact number, date joined, salary, job profile;

4. All information is displayed; ask for confirmation

5. User clicks on “Confirm” button

Case 2: update operation (Use the update command to update a stuff from the stuff table)

Use case name: Update stuff member

Actor: Manager

Steps:

1. Actor clicks on “Update stuff member” button;

2. Prompt to enter stuff id;

3. All information is displayed; ask for confirmation

4. User clicks on “Confirm” button

5. A stuff form with the specific stuff details open;

6. Ask the user to select the field to change and input the new value into it;

7. All information is displayed; ask for confirmation

8. User clicks on “Confirm” button

Case 3: Delete operation (Use the delete command to delete a stuff member from the stuff table)

Use case name: Delete stuff member

Actor: Manager

Steps:

1. Actor clicks on “Delete stuff member” button;

2. Prompt to enter stuff id to delete;

3. All information is displayed; ask for confirmation

4. User clicks on “Confirm” button

Case 4: Join, aggregate (To view the stuff report , we need to join the stuff table with the check-table then count the number of patient that the stuff overview and produce the report for that stuff number)

Use case name: view stuff report

Actor: Manager

Steps:

1. Actor clicks on “view report” button;

2. Prompt to select the user enter user title or id;

3. All information is displayed; ask for confirmation

4. User clicks on “Confirm” button

5. A stuff form open with all stuff assign to the tittle id or a specific stuff based on the stuff id;

7. All information is displayed; ask for confirmation

8. User clicks on “Confirm” button

V- PATIENT:

Case 1: join operation and aggregate (To view the prescription for the patient we need to need to connect the prescription table with the patient table then join that with the medication table to show the complete prescription with all medication name)

Use case name: view Prescription report

Actor: Patient

Steps:

1. Actor clicks on “Prescription report” button;

2. Prompt to enter patient id;

3. All information is displayed; ask for confirmation

4. User clicks on “Confirm” button

5. A prescription form open with all prescription assign to the patient id and the total number;

7. All information is displayed; ask for confirmation

8. User clicks on “Confirm” button

Case 2: Create View and join aggregate and group by operation (To view the total bill for the patient we need to need to connect the prescription table with the patient table then join that with the medication and group by patient id and add the total cost for all prescription to a total price)

Use case name: Billing

Actor: Patient

Steps:

1. Actor clicks on “Patient Billing” button;

2. Prompt to enter patient id;

3. Do the Id match; Ask for confirmation

4. The patient form open for that specific id;

3. All information is displayed; ask for confirmation

4. prompt to click ok/print invoice

**VIII Use Case Realization (GLOBAL CASE FOR EACH SAME TYPE OF INSERT UPDATE AND DELETE JOIN AGGREGATE**

I- RECEPTIONIST:

Case 1: Insert operation (enter patient to the patient table)

Use case name: Enter a new patient

Actor: Receptionist

Steps:

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN001', 'Tyshawn', 'Benson', 'M', '1986-12-12', '298 Roehampton St.

Port Chester, NY 10573','(251) 546-9442');

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN002', 'Elise', 'Cardenas', 'F', '1987-11-1', ' 8995 North Pennsylvania St.','(949) 569-4371');

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN003', 'Aryan', 'Beard', 'M', '1988-09-07', ' 179 Poor House Avenue','(125) 546-4478');

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN004', 'Talon', 'Friedman', 'M', '1999-02-07', ' Myrtle Beach, SC 29577',' (630) 446-8851');

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN005', 'Angela', 'Cross', 'F', '2000-01-11', ' Baldwinsville, NY 13027','(226) 906-2721');

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN006', 'VaughnA', 'ConradS', 'M', '2001-12-01', '8387 West Blackburn St.','(671) 925-1352');

Case 2: Insert operation (Assign nurse a room)

Use case name: Assign nurse to a room

INSERT INTO Room (room\_id, staff\_id)

VALUES ('R200', 'S101');

INSERT INTO Room (room\_id, staff\_id)

VALUES ('R201', 'S103');

INSERT INTO Room (room\_id, staff\_id)

VALUES ('R202', 'S106');

INSERT INTO Room (room\_id)

VALUES ('R203');

INSERT INTO Room (room\_id)

VALUES ('R204');

INSERT INTO Room (room\_id)

VALUES ('R205');

Case 2: Join operation (join the stuff table and room table to see which nurse are assigned to which room)

Use case name: Room View Report (list of room and stuff who govern the room)

--dispaly all room

SELECT \* FROM ROOM

--JOIN ROOM AND STAFF TO Dispplay all room assgin to a Nurse with name

SELECT Room.room\_id, Staff.S\_FName,Staff.S\_LName, Staff.S\_Title

FROM Room

INNER JOIN Staff

ON Room.staff\_id=Staff.staff\_id;

--JOIN Rroom id and stuff id

SELECT Room.room\_id, Staff.staff\_id, Staff.S\_FName,Staff.S\_LName, Staff.S\_Title

FROM Room

INNER JOIN Staff

ON Room.staff\_id=Staff.staff\_id;

Case 3: Join, group and aggregate operation (aggregate and calculate the patient bill, by joining the prescription table to the patient table then group then using his id then calculate the grand total for each patient)

Use case name: Patient Unpaid Prescription Report

-- CREATE VIEW WITH TRIPE JOIN OPERTION TO SHOW THE PRECRITION OF THE PATIENT THEN SAVE TO TEMP TO ADD TOTAL PRICE

CREATE VIEW [THE PATIENT TATOL] AS

SELECT prescription\_id, Patient.P\_FName, Patient.P\_LName, medication.med\_name, comment, PR\_med\_quantity, PR\_price

FROM prescription

JOIN medication

ON medication.med\_id = prescription.med\_id

JOIN Patient

ON Patient.patient\_id = prescription.patient\_id

WHERE prescription.patient\_id = 'PN002' – JUST ADD ID NUMBER

--DISPLAY THE VIEW AND ADD THE TOTAL PRICE OF ALL MEDICATION

SELECT \* FROM [THE PATIENT TATOL];

SELECT SUM(PR\_price) AS TOTAL\_PRICE FROM [THE PATIENT TATOL]

Case 4: Delete/Update operation (remove or update patient from the patient table)

Use case name: Update/Remove Patient

Actor: Receptionist

--DELETE A PATIENT FROM PATIENT TABLE

DELETE FROM Patient

WHERE patient\_id ='PN001';

--UPDATE PATIENT ADRESS TO NEW ONE

UPDATE Patient   
SET P\_adress = ‘newadress’

WHERE patient\_id ='PN001';

II- NURSE:

Case 1: update operation (Upddate patient and doctor to a room via the check-in table)

Use case name: assign patient and doctor to a room

Actor: Nurse

Steps:

---ADD patient , stuff , room and check in dat to checkin table

UPDATE checkin

SET room\_id = 'R200', stuff\_id = 'S102' ,ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN001';

UPDATE checkin

SET room\_id = 'R201', stuff\_id = 'S104' ,ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN002';

UPDATE checkin

SET room\_id = 'R202', stuff\_id = 'S102' ,ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN003';

UPDATE checkin

SET ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN004';

UPDATE checkin

SET ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN005';

Case 2: update Operation (change the current doctor from the check-in table to another room)

Use case name: Update the Doctor who currently seeing the patient to a new one

Actor: Nurse

UPDATE checkin

SET room\_id = 'R200', stuff\_id = 'S103' ,ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN001';

Case 3: Join and aggregate (view the list of admitted patient and total number of admitted patient, by combining the check-in patient with the patient table then count the total number of admitted patient)

Use case name: Check for the list and number of admitted patient

Actor: Nurse

Steps:

--DISPLAY CHECKIN TABLE

SELECT \* FROM checkin

---TRIPLE JOIN TO DISPLAY LEST OF THE PATIENT NAME DOC WHO TAKE CARE OF THEM

SELECT checkin.patient\_id, Patient.P\_FName, Patient.P\_LName, CHECKIN.room\_id,CHECKIN.stuff\_id, Staff.S\_FName, Staff.S\_LName, Staff.S\_Title

FROM checkin

JOIN Patient

ON Patient.patient\_id = checkin.patient\_id

JOIN Staff

ON CHECKIN.stuff\_id = Staff.staff\_id

--total number of patient in check in

SELECT COUNT( patient\_id ) AS TotalNumbPatientInCHECKin FROM checkin

--total number of patient who have benn assing to doc

SELECT COUNT( patient\_id ) AS TotalNumbPatientInCHECKin FROM checkin where stuff\_id IS NOT NULL

Case 4: Delete operation (Remove a patient from the check-in table by using the delete operation and set to not cascade)

Use case name: Remove patient from a room

Actor: Nurse

Steps:

--DELETE A PATIENT FROM CHECKIN 1

DELETE FROM checkin

WHERE patient\_id ='PN001';

--DELETE A PATIENT FROM CHECKIN 2

DELETE FROM checkin

WHERE patient\_id ='PN002';--2,3

--DELETE A PATIENT FROM CHECKIN 3

DELETE FROM checkin

WHERE patient\_id ='PN003';--2,3

III- DOCTOR:

Case 1: insert operation (the doctor creates a prescription by using insert oeration)

Use case name: create Prescription

Actor: Doctor

Steps:

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN001', 3, 6, 'An abdominal aortic aneurysm (AAA)', 2);

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN001', 7, 6, 'Roseola', 3);

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN002', 3, 6, 'Strep Throat', 4);

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN003', 7, 6, 'Breathing difficulty, Strep Throat', 8);

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN002', 3, 6, 'Hand-Foot-Mouth Disease', 7);

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN003', 7, 6, 'THE COLD, Hand-Foot-Mouth Disease', 1);

Case 2: Join and update operation (to update the prescription table the doctor need to join the prescription table with the medication and project the require field then insert the desire value)

Use case name: Update Prescription

Actor: Doctor

Steps:

-- USING UPDATE AND JOIN ON PRESCRITION PRICE BY JOIN THE MEDICATION TO RETRIVE PRICE AND MULT THAT PRICE BY THE CORRESPONDING MED QUATITY

UPDATE prescription

SET PR\_price = PR\_med\_quantity \* medication.med\_price

FROM prescription

JOIN medication

ON medication.med\_id = prescription.med\_id

Case 3: join group and aggregate operation (to view the prescription record list, the doctor need to join the patient table with prescription then retrieve the patient need needed them group by the patient name, then calculate the total price of all prescription assign to that customer)

Use case name: view Prescription report

Actor: Doctor

Steps:

-- TRIPE JOIN OPERTION TO SHOW THE PRECRITION AND PATIENT , MED NAME AND ALL DETAILS

SELECT prescription\_id, Patient.P\_FName, Patient.P\_LName, medication.med\_name, comment, PR\_med\_quantity, PR\_price

FROM prescription

JOIN Patient

ON Patient.patient\_id = prescription.patient\_id

JOIN medication

ON medication.med\_id = prescription.med\_id

Case 4: Delete operation (to delete prescription the diction just need to use the command to delete)

Use case name: Delete Prescription

Actor: Doctor

Steps:

--DELETE A PATIENT FROM PRESCRITION 1

DELETE FROM prescription

WHERE patient\_id ='PN001';

--DELETE A PATIENT FROM PRESCRITION 2

DELETE FROM prescription

WHERE patient\_id ='PN002';--2,3

--DELETE A PATIENT FROM PRESCRITION 3

DELETE FROM prescription

WHERE patient\_id ='PN003';--2,3

IV- MANAGER:

Case 1: Insert operation (use the insert command to insert a new stuff to the stuff table)

Use case name: Enter new stuff

Actor: Manager

Steps:

---ADD STUFF TO STUFF TABLE

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title)

VALUES ('S100', 'Trinity', 'Yang', 'M', '1979-11-01', '8387 West Blackburn St.

Montclair, NJ 07042','(630) 446-8851','Recpetionist', 8769);

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title)

VALUES ('S101', 'Dax', 'Chung', 'M', '1986-12-12', '298 Roehampton St.

Port Chester, NY 10573','(251) 546-9442','Nurse', 10623);

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title)

VALUES ('S102', 'Dorian', 'Andersen', 'M', '1987-11-1', ' 8995 North Pennsylvania St.','(949) 569-4371','Doctor',21000);

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title)

VALUES ('S103', 'Katelynn', 'Ross', 'F', '1988-09-07', ' 179 Poor House Avenue','(125) 546-4478','Nurse', 25000);

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title)

VALUES ('S104', 'Riya', 'Duran', 'M', '1999-02-07', ' Myrtle Beach, SC 29577',' (630) 446-8851','Doctor', 11232);

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title)

VALUES ('S105', 'Josephine', 'Mooney', 'F', '2000-01-11', ' Baldwinsville, NY 13027','(226) 906-2721','Manager', 12321);

Case 2: update operation (Use the update command to update a stuff from the stuff table)

Use case name: Update stuff member

Actor: Manager

Steps:

UPDATE Staff

SET (S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title)

VALUE ('S106', 'Brock', 'Dodson', 'M', '2001-12-01', '8387 West Blackburn St.','(671) 925-1352','Nurse', 13654);

WHERE Staff\_id = 'S104';

Case 3: Delete operation (Use the delete command to delete a stuff member from the stuff table)

Use case name: Delete stuff member

Actor: Manager

Steps:

--DELETE A PATIENT FROM PATIENT TABLE

DELETE FROM Staff

WHERE Staff\_id = 'S104';

Case 4: Join , aggregate ( To view the stuff report , we need to join the stuff table with the check-table then count the number of patient that the stuff overview and produce the report for that stuff number)

Use case name: view stuff report

Actor: Manager

Steps:

--dispaly all stuff and room

SELECT \* FROM ROOM

--JOIN ROOM AND STAFF TO Dispplay all room assgin to a Nurse with name

SELECT Room.room\_id, Staff.S\_FName,Staff.S\_LName, Staff.S\_Title

FROM Room

INNER JOIN Staff

ON Room.staff\_id=Staff.staff\_id;

--JOIN Rroom id and stuff id

SELECT Room.room\_id, Staff.staff\_id, Staff.S\_FName,Staff.S\_LName, Staff.S\_Title

FROM Room

INNER JOIN Staff

ON Room.staff\_id=Staff.staff\_id;

--display all stuff

SELECT staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title FROM Staff

--COUNT THE NUMBER OF STUFF

SELECT COUNT( staff\_id ) AS TotalNumberofStuff FROM Staff

-- TOTAL NUMBER BY TITLE nurse

SELECT COUNT(staff\_id) AS total\_nurse

FROM Staff

WHERE S\_Title = 'Nurse';

-- TOTAL NUMBER BY TITLE nurse

SELECT COUNT(staff\_id) AS total\_doc

FROM Staff

WHERE S\_Title = 'Doctor';

-- GROUP BY TITLE

SELECT S\_Title, COUNT(S\_Title) AS les\_Trite, SUM(S\_salary) AS les\_salary

FROM Staff

GROUP BY S\_Title

ORDER BY S\_Title;8. User clicks on “Confirm” button

V- PATIENT:

Case 1: join operation and aggregate (To view the prescription for the patient we need to need to connect the prescription table with the patient table then join that with the medication table to show the complete prescription with all medication name)

Use case name: view Prescription report

Actor: Patient

Steps:

-- TRIPE JOIN OPERTION TO SHOW THE PRECRITION OF THE PATIENT

SELECT prescription\_id, Patient.P\_FName, Patient.P\_LName, medication.med\_name, comment, PR\_med\_quantity, PR\_price

FROM prescription

JOIN medication

ON medication.med\_id = prescription.med\_id

JOIN Patient

ON Patient.patient\_id = prescription.patient\_id

WHERE prescription.patient\_id = 'PN002' //any patient number

Case 2: create view join aggregate and group by operation (To view the total bill for the patient we need to need to connect the prescription table with the patient table then join that with the medication and group by patient id and add the total cost for all prescription to a total price)

Use case name: Billing

Actor: Patient

Steps:

-- CREATE VIEW WITH TRIPE JOIN OPERTION TO SHOW THE PRECRITION OF THE PATIENT THEN SAVE TO TEMP TO ADD TOTAL PRICE

CREATE VIEW [THE PATIENT TATOL] AS

SELECT prescription\_id, Patient.P\_FName, Patient.P\_LName, medication.med\_name, comment, PR\_med\_quantity, PR\_price

FROM prescription

JOIN medication

ON medication.med\_id = prescription.med\_id

JOIN Patient

ON Patient.patient\_id = prescription.patient\_id

WHERE prescription.patient\_id = 'PN002' -- 1 ,2 ,3

--DISPLAY THE VIEW AND ADD THE TOTAL PRICE OF ALL MEDICATION

SELECT \* FROM [THE PATIENT TATOL];

SELECT SUM(PR\_price) AS TOTAL\_PRICE FROM [THE PATIENT TATOL]

**VIIII TEST PLAN AND RECCORD**

I- RECEPTIONIST:

Case 1: Insert operation (enter patient to the patient table)

Use case name: Enter a new patient

Actor: Receptionist

**INPUT:**

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN001', 'Tyshawn', 'Benson', 'M', '1986-12-12', '298 Roehampton St.

Port Chester, NY 10573','(251) 546-9442');

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN002', 'Elise', 'Cardenas', 'F', '1987-11-1', ' 8995 North Pennsylvania St.','(949) 569-4371');

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN003', 'Aryan', 'Beard', 'M', '1988-09-07', ' 179 Poor House Avenue','(125) 546-4478');

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN004', 'Talon', 'Friedman', 'M', '1999-02-07', ' Myrtle Beach, SC 29577',' (630) 446-8851');

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN005', 'Angela', 'Cross', 'F', '2000-01-11', ' Baldwinsville, NY 13027','(226) 906-2721');

INSERT INTO Patient (patient\_id, P\_FName, P\_LName, P\_sex, P\_Dob, P\_Address, P\_Contact)

VALUES ('PN006', 'VaughnA', 'ConradS', 'M', '2001-12-01', '8387 West Blackburn St.','(671) 925-1352');

**OUTPUT:**

PN001 Tyshawn Benson M 1986-12-12 298 Roehampton St. Port Chester, NY 10573 (251) 546-9442

PN002 Elise Cardenas F 1987-11-01 8995 North Pennsylvania St. (949) 569-4371

PN003 Aryan Beard M 1988-09-07 179 Poor House Avenue (125) 546-4478

PN004 Talon Friedman M 1999-02-07 Myrtle Beach, SC 29577 (630) 446-8851

PN005 Angela Cross F 2000-01-11 Baldwinsville, NY 13027 (226) 906-2721

PN006 VaughnA ConradS M 2001-12-01 8387 West Blackburn St. (671) 925-1352

Case 2: Insert operation (Assign nurse a room)

Use case name: Assign nurse to a room

**INPUT:**

INSERT INTO Room (room\_id, staff\_id)

VALUES ('R200', 'S101');

INSERT INTO Room (room\_id, staff\_id)

VALUES ('R201', 'S103');

INSERT INTO Room (room\_id, staff\_id)

VALUES ('R202', 'S106');

INSERT INTO Room (room\_id)

VALUES ('R203');

INSERT INTO Room (room\_id)

VALUES ('R204');

INSERT INTO Room (room\_id)

VALUES ('R205');

**OUTPUT:**

R200 NULL S101

R201 NULL S103

R202 NULL S106

R203 NULL NULL

R204 NULL NULL

R205 NULL NULL

Case 2: Join operation (join the stuff table and room table to see which nurse are assigned to which room)

Use case name: Room View Report (list of room and stuff who govern the room)

**INPUT:**

--dispLAY all room

SELECT \* FROM ROOM

--JOIN ROOM AND STAFF TO Dispplay all room assgin to a Nurse with name

SELECT Room.room\_id, Staff.S\_FName,Staff.S\_LName, Staff.S\_Title

FROM Room

INNER JOIN Staff

ON Room.staff\_id=Staff.staff\_id;

--JOIN Rroom id and stuff id

SELECT Room.room\_id, Staff.staff\_id, Staff.S\_FName,Staff.S\_LName, Staff.S\_Title

FROM Room

INNER JOIN Staff

ON Room.staff\_id=Staff.staff\_id;

**OUTPUT:**

R200 Dax Chung Nurse

R201 Katelynn Ross Nurse

R202 Brock Dodson Nurse

R200 S101 Dax Chung Nurse

R201 S103 Katelynn Ross Nurse

R202 S106 Brock Dodson Nurse

Case 3: Join, group and aggregate operation (aggregate and calculate the patient bill, by joining the prescription table to the patient table then group then using his id then calculate the grand total for each patient)

Use case name: Patient Unpaid Prescription Report

**INPUT:**

-- CREATE VIEW WITH TRIPE JOIN OPERTION TO SHOW THE PRECRITION OF THE PATIENT THEN SAVE TO TEMP TO ADD TOTAL PRICE

CREATE VIEW [THE PATIENT TATOL] AS

SELECT prescription\_id, Patient.P\_FName, Patient.P\_LName, medication.med\_name, comment, PR\_med\_quantity, PR\_price

FROM prescription

JOIN medication

ON medication.med\_id = prescription.med\_id

JOIN Patient

ON Patient.patient\_id = prescription.patient\_id

WHERE prescription.patient\_id = 'PN001' – JUST ADD ID NUMBER

--DISPLAY THE VIEW AND ADD THE TOTAL PRICE OF ALL MEDICATION

SELECT \* FROM [THE PATIENT TATOL];

SELECT SUM(PR\_price) AS TOTAL\_PRICE FROM [THE PATIENT TATOL]

**OUTPUT:**

22 Elise Cardenas Calamine Lotion Strep Throat 4 44.00

24 Elise Cardenas Aceclofenac 100 mg Tab Hand-Foot-Mouth Disease 7 56.00

TOTAL\_PRICE = 100.00

Case 4: Delete/Update operation (remove or update patient from the patient table)

Use case name: Update/Remove Patient

Actor: Receptionist

**INPUT:**

--DELETE A PATIENT FROM PATIENT TABLE

DELETE FROM Patient

WHERE patient\_id ='PN001';

--UPDATE PATIENT ADRESS TO NEW ONE

UPDATE Patient   
SET P\_adress = ‘newadress’

WHERE patient\_id ='PN006';

**OUTPUT:**

PN002 Elise Cardenas F 1987-11-01 8995 North Pennsylvania St. (949) 569-4371

PN003 Aryan Beard M 1988-09-07 179 Poor House Avenue (125) 546-4478

PN004 Talon Friedman M 1999-02-07 Myrtle Beach, SC 29577 (630) 446-8851

PN005 Angela Cross F 2000-01-11 Baldwinsville, NY 13027 (226) 906-2721

PN006 VaughnA ConradS M 2001-12-01 newadress (671) 925-1352

II- NURSE:

Case 1: update operation (Update patient and doctor to a room via the check-in table)

Use case name: assign patient and doctor to a room

Actor: Nurse

**INPUT:**

---ADD patient , stuff , room and check in dat to checkin table

UPDATE checkin

SET room\_id = 'R200', stuff\_id = 'S102' ,ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN001';

UPDATE checkin

SET room\_id = 'R201', stuff\_id = 'S104' ,ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN002';

UPDATE checkin

SET room\_id = 'R202', stuff\_id = 'S102' ,ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN003';

UPDATE checkin

SET ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN004';

UPDATE checkin

SET ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN005';

**OUTPUT:**

12 PN001 R200 S102 2018-10-01

13 PN002 R201 S104 2018-10-01

14 PN003 R202 S102 2018-10-01

15 PN004 NULL NULL 2018-10-01

16 PN005 NULL NULL 2018-10-01

17 PN006 NULL NULL NULL

Case 2: update Operation (change the current doctor from the check-in table to another room)

Use case name: Update the Doctor who currently seeing the patient to a new one

Actor: Nurse

**INPUT:**

UPDATE checkin

SET room\_id = 'R200', stuff\_id = 'S103' ,ch\_in\_date = '2018-10-01'

WHERE patient\_id = 'PN001';

**OUTPUT:**

12 PN001 R200 S103 2018-10-01

Case 3: Join and aggregate (view the list of admitted patient and total number of admitted patient, by combining the check-in patient with the patient table then count the total number of admitted patient)

Use case name: Check for the list and number of admitted patient

Actor: Nurse

**INPUT:**

--DISPLAY CHECKIN TABLE

SELECT \* FROM checkin

---TRIPLE JOIN TO DISPLAY LEST OF THE PATIENT NAME DOC WHO TAKE CARE OF THEM

SELECT checkin.patient\_id, Patient.P\_FName, Patient.P\_LName, CHECKIN.room\_id,CHECKIN.stuff\_id, Staff.S\_FName, Staff.S\_LName, Staff.S\_Title

FROM checkin

JOIN Patient

ON Patient.patient\_id = checkin.patient\_id

JOIN Staff

ON CHECKIN.stuff\_id = Staff.staff\_id

--total number of patient in check in

SELECT COUNT( patient\_id ) AS TotalNumbPatientInCHECKin FROM checkin

--total number of patient who have benn assing to doc

SELECT COUNT( patient\_id ) AS TotalNumbPatientInCHECKin FROM checkin where stuff\_id IS NOT NULL

**OUTPUT:**

12 PN001 R200 S102 2018-10-01

13 PN002 R201 S104 2018-10-01

14 PN003 R202 S102 2018-10-01

15 PN004 NULL NULL 2018-10-01

16 PN005 NULL NULL 2018-10-01

17 PN006 NULL NULL NULL

PN001 Tyshawn Benson R200 S102 Dorian Andersen Doctor

PN002 Elise Cardenas R201 S104 Riya Duran Doctor

PN003 Aryan Beard R202 S102 Dorian Andersen Doctor

TotalNumbPatientInCHECKin = 6

TotalNumbPatientIndoc = 3

Case 4: Delete operation (Remove a patient from the check-in table by using the delete operation and set to not cascade)

Use case name: Remove patient from a room

Actor: Nurse

**INPUT:**

--DELETE A PATIENT FROM CHECKIN 1

DELETE FROM checkin

WHERE patient\_id ='PN001';

--DELETE A PATIENT FROM CHECKIN 2

DELETE FROM checkin

WHERE patient\_id ='PN002';--2,3

--DELETE A PATIENT FROM CHECKIN 3

DELETE FROM checkin

WHERE patient\_id ='PN003';--2,3

**OUTPUT:**

15 PN004 NULL NULL 2018-10-01

16 PN005 NULL NULL 2018-10-01

17 PN006 NULL NULL NULL

III- DOCTOR:

Case 1: insert operation (the doctor creates a prescription by using insert operation)

Use case name: create Prescription

Actor: Doctor

**INPUT:**

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN001', 16, 12, 'An abdominal aortic aneurysm (AAA)', 2);

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN001', 17, 12, 'Roseola', 3);

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN002', 18,13, 'Strep Throat', 4);

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN003', 19, 14, 'Breathing difficulty, Strep Throat', 8);

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN002', 16, 13, 'Hand-Foot-Mouth Disease', 7);

INSERT INTO prescription(patient\_id, med\_id, checkin\_id, comment,PR\_med\_quantity)

VALUES ('PN003', 21, 14, 'THE COLD, Hand-Foot-Mouth Disease', 1);

**OUTPUT:**

20 PN001 16 12 An abdominal aortic aneurysm (AAA) 2 NULL

21 PN001 17 12 Roseola 3 NULL

22 PN002 18 13 Strep Throat 4 NULL

23 PN003 19 14 Breathing difficulty, Strep Throat 8 NULL

24 PN002 16 13 Hand-Foot-Mouth Disease 7 NULL

25 PN003 21 14 THE COLD, Hand-Foot-Mouth Disease 1 NULL

Case 2: Join and update operation (to update the prescription table the doctor need to join the prescription table with the medication and project the require field then insert the desire value)

Use case name: Update Prescription

Actor: Doctor

**INPUT:**

-- USING UPDATE AND JOIN ON PRESCRITION PRICE BY JOIN THE MEDICATION TO RETRIVE PRICE AND MULT THAT PRICE BY THE CORRESPONDING MED QUATITY

UPDATE prescription

SET PR\_price = PR\_med\_quantity \* medication.med\_price

FROM prescription

JOIN medication

ON medication.med\_id = prescription.med\_id

**OUTPUT:**

20 PN001 16 12 An abdominal aortic aneurysm (AAA) 2 16.00

21 PN001 17 12 Roseola 3 8.70

22 PN002 18 13 Strep Throat 4 44.00

23 PN003 19 14 Breathing difficulty, Strep Throat 8 156.16

24 PN002 16 13 Hand-Foot-Mouth Disease 7 56.00

25 PN003 21 14 THE COLD, Hand-Foot-Mouth Disease 1 12.24

Case 3: join group and aggregate operation (to view the prescription record list, the doctor need to join the patient table with prescription then retrieve the patient need needed them group by the patient name, then calculate the total price of all prescription assign to that customer)

Use case name: view Prescription report

Actor: Doctor

**INPUT:**

-- TRIPE JOIN OPERTION TO SHOW THE PRECRITION AND PATIENT , MED NAME AND ALL DETAILS

SELECT prescription\_id, Patient.P\_FName, Patient.P\_LName, medication.med\_name, comment, PR\_med\_quantity, PR\_price

FROM prescription

JOIN Patient

ON Patient.patient\_id = prescription.patient\_id

JOIN medication

ON medication.med\_id = prescription.med\_id

**OUTPUT:**

20 Tyshawn Benson Aceclofenac 100 mg Tab An abdominal aortic aneurysm (AAA) 2 16.00

21 Tyshawn Benson Diclofenac Sodium 50 mg Tab Roseola 3 8.70

22 Elise Cardenas Calamine Lotion Strep Throat 4 44.00

23 Aryan Beard Chlorhe Wash Breathing difficulty, Strep Throat 8 156.16

24 Elise Cardenas Aceclofenac 100 mg Tab Hand-Foot-Mouth Disease 7 56.00

25 Aryan Beard Fusidic Acid 2 % w/v Cream THE COLD, Hand-Foot-Mouth Disease 1 12.24

Case 4: Delete operation (to delete prescription the diction just need to use the command to delete)

Use case name: Delete Prescription

Actor: Doctor

**INPUT:**

--DELETE A PATIENT FROM PRESCRITION 1

DELETE FROM prescription

WHERE patient\_id ='PN001';

--DELETE A PATIENT FROM PRESCRITION 2

DELETE FROM prescription

WHERE patient\_id ='PN002';--2,3

**OUTPUT:**

23 PN003 19 14 Breathing difficulty, Strep Throat 8 156.16

25 PN003 21 14 THE COLD, Hand-Foot-Mouth Disease 1 12.24

IV- MANAGER:

Case 1: Insert operation (use the insert command to insert a new stuff to the stuff table)

Use case name: Enter new stuff

Actor: Manager

**INPUT:**

---ADD STUFF TO STUFF TABLE

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title, S\_salary)

VALUES ('S100', 'Trinity', 'Yang', 'M', '1979-11-01', '8387 West Blackburn St.

Montclair, NJ 07042','(630) 446-8851','Recpetionist', 8769);

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title, S\_salary)

VALUES ('S101', 'Dax', 'Chung', 'M', '1986-12-12', '298 Roehampton St.

Port Chester, NY 10573','(251) 546-9442','Nurse', 10623);

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title, S\_salary)

VALUES ('S102', 'Dorian', 'Andersen', 'M', '1987-11-1', ' 8995 North Pennsylvania St.','(949) 569-4371','Doctor',21000);

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title, S\_salary)

VALUES ('S103', 'Katelynn', 'Ross', 'F', '1988-09-07', ' 179 Poor House Avenue','(125) 546-4478','Nurse', 25000);

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title, S\_salary)

VALUES ('S104', 'Riya', 'Duran', 'M', '1999-02-07', ' Myrtle Beach, SC 29577',' (630) 446-8851','Doctor', 11232);

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title, S\_salary)

VALUES ('S105', 'Josephine', 'Mooney', 'F', '2000-01-11', ' Baldwinsville, NY 13027','(226) 906-2721','Manager', 12321);

INSERT INTO Staff (staff\_id, S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title, S\_salary)

VALUES ('S106', 'Brock', 'Dodson', 'M', '2001-12-01', '8387 West Blackburn St.','(671) 925-1352','Nurse', 13654);

**OUTPUT:**

S100 Trinity Yang M 1979-11-01 8387 West Blackburn St. Montclair, NJ 07042 (630) 446-8851 Receptionist 8769.00

S101 Dax Chung M 1986-12-12 298 Roehampton St. Port Chester, NY 10573 (251) 546-9442 Nurse 10623.00

S102 Dorian Andersen M 1987-11-01 8995 North Pennsylvania St. (949) 569-4371 Doctor 21000.00

S103 Katelynn Ross F 1988-09-07 179 Poor House Avenue (125) 546-4478 Nurse 25000.00

S104 Riya Duran M 1999-02-07 Myrtle Beach, SC 29577 (630) 446-8851 Doctor 11232.00

S105 Josephine Mooney F 2000-01-11 Baldwinsville, NY 13027 (226) 906-2721 Manager 12321.00

S106 Brock Dodson M 2001-12-01 8387 West Blackburn St. (671) 925-1352 Nurse 13654.00

Case 2: update operation (Use the update command to update a stuff from the stuff table)

Use case name: Update stuff member

Actor: Manager

**INPUT:**

UPDATE Staff

SET (S\_FName, S\_LName, S\_sex, S\_Dob, S\_Address, S\_Contact, S\_Title)

VALUE ('Brock', 'Dodson', 'M', '2001-12-01', new adreess.','(671) 925-1352','Nurse', 13654);

WHERE Staff\_id = 'S104';

**OUTPUT:**

S104 Riya Duran M 1999-02-07 new adress (630) 446-8851 Doctor 11232.00

Case 3: Delete operation (Use the delete command to delete a stuff member from the stuff table)

Use case name: Delete stuff member

Actor: Manager

**INPUT:**

--DELETE A PATIENT FROM PATIENT TABLE

DELETE FROM Staff

WHERE Staff\_id = 'S104';

**OUTPUT:**

S100 Trinity Yang M 1979-11-01 8387 West Blackburn St. Montclair, NJ 07042 (630) 446-8851 Receptionist 8769.00

S101 Dax Chung M 1986-12-12 298 Roehampton St. Port Chester, NY 10573 (251) 546-9442 Nurse 10623.00

S102 Dorian Andersen M 1987-11-01 8995 North Pennsylvania St. (949) 569-4371 Doctor 21000.00

S103 Katelynn Ross F 1988-09-07 179 Poor House Avenue (125) 546-4478 Nurse 25000.00

S105 Josephine Mooney F 2000-01-11 Baldwinsville, NY 13027 (226) 906-2721 Manager 12321.00

S106 Brock Dodson M 2001-12-01 8387 West Blackburn St. (671) 925-1352 Nurse 13654.00

Case 4: Join, aggregate (To view the stuff report, we need to join the stuff table with the check-table then count the number of patient that the stuff overview and produce the report for that stuff number)

Use case name: view stuff report

Actor: Manager

**INPUT:**

--dispaly all stuff and room

SELECT \* FROM ROOM

--JOIN ROOM AND STAFF TO Dispplay all room assgin to a Nurse with name

SELECT Room.room\_id, Staff.S\_FName,Staff.S\_LName, Staff.S\_Title

FROM Room

INNER JOIN Staff

ON Room.staff\_id=Staff.staff\_id;

--JOIN Rroom id and stuff id

SELECT Room.room\_id, Staff.staff\_id, Staff.S\_FName,Staff.S\_LName, Staff.S\_Title

FROM Room

INNER JOIN Staff

ON Room.staff\_id=Staff.staff\_id;

--COUNT THE NUMBER OF STUFF

SELECT COUNT( staff\_id ) AS TotalNumberofStuff FROM Staff

-- TOTAL NUMBER BY TITLE nurse

SELECT COUNT(staff\_id) AS total\_nurse

FROM Staff

WHERE S\_Title = 'Nurse';

-- TOTAL NUMBER BY TITLE nurse

SELECT COUNT(staff\_id) AS total\_doc

FROM Staff

WHERE S\_Title = 'Doctor';

-- GROUP BY TITLE

SELECT S\_Title, COUNT(S\_Title) AS les\_Trite, SUM(S\_salary) AS les\_salary

FROM Staff

GROUP BY S\_Title

ORDER BY S\_Title;

**OUTPUT:**

R200 PN001 S101

R201 PN002 S103

R202 PN003 S106

R203 NULL NULL

R204 NULL NULL

R205 NULL NULL

R200 S101 Dax Chung Nurse

R201 S103 Katelynn Ross Nurse

R202 S106 Brock Dodson Nurse

TotalNumberofStuff = 7

total\_nurse = 3

total\_doc = 2

V- PATIENT:

Case 1: join operation and aggregate (To view the prescription for the patient we need to need to connect the prescription table with the patient table then join that with the medication table to show the complete prescription with all medication name)

Use case name: view Prescription report

Actor: Patient

**INPUT:**

-- TRIPE JOIN OPERTION TO SHOW THE PRECRITION OF THE PATIENT

SELECT prescription\_id, Patient.P\_FName, Patient.P\_LName, medication.med\_name, comment, PR\_med\_quantity, PR\_price

FROM prescription

JOIN medication

ON medication.med\_id = prescription.med\_id

JOIN Patient

ON Patient.patient\_id = prescription.patient\_id

WHERE prescription.patient\_id = 'PN002' //any patient number

**OUTPUT:**

22 Elise Cardenas Calamine Lotion Strep Throat 4 44.00

24 Elise Cardenas Aceclofenac 100 mg Tab Hand-Foot-Mouth Disease 7 56.00

Case 2: create view join aggregate and group by operation (To view the total bill for the patient we need to need to connect the prescription table with the patient table then join that with the medication and group by patient id and add the total cost for all prescription to a total price)

Use case name: Billing

Actor: Patient

**INPUT:**

-- CREATE VIEW WITH TRIPE JOIN OPERTION TO SHOW THE PRECRITION OF THE PATIENT THEN SAVE TO TEMP TO ADD TOTAL PRICE

CREATE VIEW [THE PATIENT TATOL] AS

SELECT prescription\_id, Patient.P\_FName, Patient.P\_LName, medication.med\_name, comment, PR\_med\_quantity, PR\_price

FROM prescription

JOIN medication

ON medication.med\_id = prescription.med\_id

JOIN Patient

ON Patient.patient\_id = prescription.patient\_id

WHERE prescription.patient\_id = 'PN002' -- 1 ,2 ,3

--DISPLAY THE VIEW AND ADD THE TOTAL PRICE OF ALL MEDICATION

SELECT \* FROM [THE PATIENT TATOL];

SELECT SUM(PR\_price) AS TOTAL\_PRICE FROM [THE PATIENT TATOL]

**OUTPUT:**

22 Elise Cardenas Calamine Lotion Strep Throat 4 44.00

24 Elise Cardenas Aceclofenac 100 mg Tab Hand-Foot-Mouth Disease 7 56.00

TOTAL\_PRICE = 100.00

**X- Conclusion:**

In conclusion, after doing this project I can say that sql is very important during the design of an application. The advantage of designed your data storage base on the need of the application is very useful. For that I can say the general clinic management database was a good project but would be more successfully if I had more time.

**XI-Reference:**

<https://docs.microsoft.com/en-us/sql/index?view=sql-server-2017#pivot=main&panel=databases>

-Database Systems—A Practical Approach to Design, Implementation, and Management” by Thomas Connolly and Carolyn Begg, Addison Wesley

<https://www.w3schools.com/sql/>