MSIS 2603 Project | Team Data Tribe

Care Intelligence System

A HOSPITAL DATABASE MANAGEMENT SYSTEM

TEAM DATA TRIBE :

Aashka Aradhya W1443155
Aditi Tupsakhare W1425744
Aishwarya Dingre W1409920
Dhiren Rikhra W1443153

Table of Contents

Business Application Description

1.1 Introduction:

Hospital management requires a lot of decision making which is highly difficult if there is no strong management system in place. Since you need precise and accurate implementation at every stage, the automation system in the hospital has to be self-sufficient. It is not possible to maintain paperwork for all the hospital activities and an automated database management system is necessary. A reliable, cost-effective, and efficient system becomes the backbone of the success of a medical center.

A well implemented hospital database helps to achieve good quality ratings and avoids errors by tracking all the necessary details. It also helps the facilities in better revenue management. Along with improved data security this system will also help the decision makers with all the necessary data required to take those decisions. In addition to these benefits, tracking details such as room occupancy, staff availability, operation information and several business metrics to measure performance of the hospital are readily available.

1.2 Objective of Care Intelligence System

A Unified Healthcare Data Platform: Data can be overwhelming, but it doesn't have to be this way. To drive healthcare efficiency, we designed a modular product for a smooth transition into a data-driven world within 4 weeks.

Finally, care that is coordinated and efficient: Forget complicated EHRs and Excel sheets. Switch to smarter and efficient care management that lets you focus on what you do best – provide seamless care

Consolidated Patient Information: Patient information like medical records, care gaps, and physician notes are present across different systems making it an operational nightmare. CIS will integrate these sources to bring data at a unified location whenever you need it

Holistic patient profiles for care teams and physicians: A complete view of the patient in one window. CIS provides a 360-degree view of the patient without switching between applications

Focus on your patients, CIS will handle the rest: CIS can streamline tasks by automating patient prioritization and assignment to care managers. Spare the long hours spent in searching patients in files and EMRs

Integrated workflows: Ensure consistency and productivity by binding the care teams together to a trusted source of information

Low Administrative Burden: Care Managers spend a lot of their time doing redundant work like writing letters, hunting for documents, and sending mails. CIS will eliminate all these daunting tasks by an integrated data lake

1.3 Scope

For the purpose of this project, we have limited our scope to empowering our users to access consolidated information across user functionalities. We have not yet integrated pharmacy inventory management and lab testing diagnostics in our system. We aim to start with basic functionalities and expand our scope based on user feedback and experience.

User Types or Entities

Users interacting with our Care Intelligence System:

1. **Admin**- Responsible for staff registration, shift schedule allocation, room availability approval and invoice generation and will overlook revenue management.

2. **Doctor**- Access scheduled, completed operations and appointments, view schedule, view patient history, enter and/or alter visit details (notes, prescription, next visits etc.) for patients

3. **Patient**- Register as new user or log-in as existing user, check doctor availability and book appointment, update medical history for CIS's records

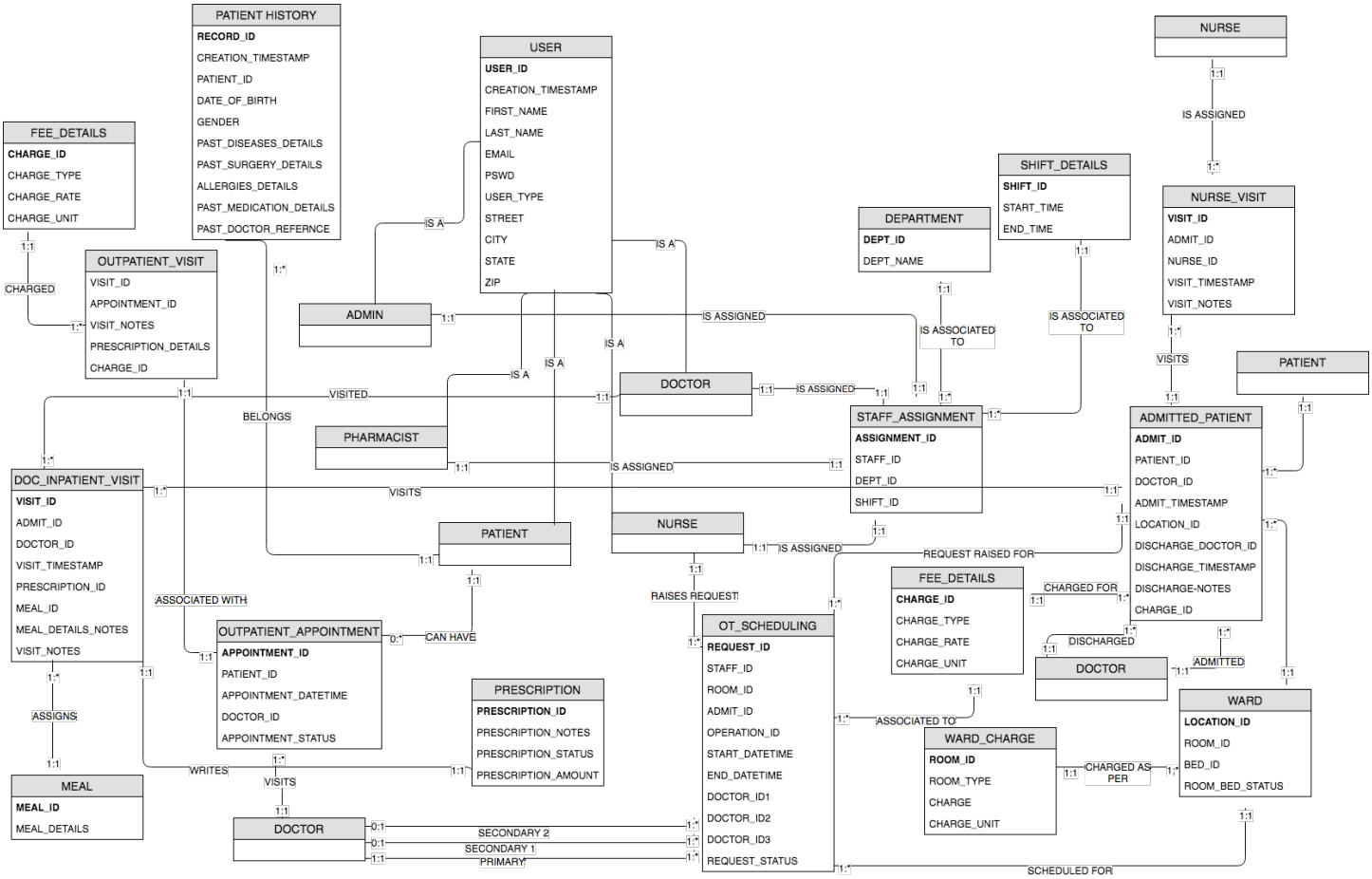
4. **Nurse**- View patient history, doctor visit notes, view and edit in-patient visits, monitor patient discharge process, raise requests for Operation Theatres (OTs) , view and update meal plans, record visit details

5. **Pharmacist**- View and approve all prescriptions raised for admitted patients and generate bills for prescriptions approved.

Tables

1. USER
2. PATIENT_HISTORY
3. DEPARTMENT
4. SHIFT_DETAILS
5. STAFF_ASSIGNMENT
6. WARD
7. WARD_CHARGE
8. ADMITTED_PATIENT
9. PRESCRIPTION
10. MEAL
11. DOC_INPATIENT_VISIT
12. OUTPATIENT_APPOINTMENT
13. FEE_DETAILS
14. OUTPATIENT_VISIT
15. OT_SCHEDULING
16. NURSE_VISIT

Logical Schema – UML Model



***Have attached a separate file containing the UML for a clear view**

Use Cases

1. **Admin:**

   1.1. Register doctors and nurses into the system and generate login credentials for them.
   1.2. Approval for operation theatre allocation requests raised by the staff.
   1.3. Bill generation for outpatients and inpatients.
   1.4. Schedule generation and shift allocation for the staff

2. **Doctor:**

   2.1. Doctors can enter his/her availability in the system and can choose the shift according to their convenience
   2.2. View the schedule for inpatient visits and outpatient appointments
   2.3. View patient history
   2.4. Enter the visit details like prescription, visit notes and next visit details if required
   2.5. Doctor can view his/her operation schedule

3. **Patient**

   3.1. Can register into the system by providing basic details and medical history.
   3.2. Can book an appointment with the doctor providing desired date and time
   3.3. Can view the previous prescriptions given by the doctor

4. **Nurse**

   4.1. View patient history
   4.2. View doctor visit notes
   4.3. Raise request for operation theater as specified by doctor
   4.4. Record the details for every visit
   4.5. View meal plans

5. **Pharmacist**

   5.1. View, access and approve all the prescriptions
   5.2. Generate pharmacy bills

Physical Schema – Database Dictionary

1. USER

USER			
Name	**Type**	**Constraint**	**Domain**
USER_ID	INT	PRIMARY KEY AUTO_INCREMENT	
CREATION_TIMESTAMP	TIMESTAMP	NOT NULL	
FIRST_NAME	VARCHAR(30)	NOT NULL	
LAST_NAME	VARCHAR(50)		
EMAIL	VARCHAR(30)	NOT NULL	
PSWD	VARCHAR(20)	NOT NULL	
PHONE	VARCHAR(12)		
USER_TYPE	VARCHAR(20)	NOT NULL	PATIENT,DOCTOR,ADMIN, NURSE, PHARMACIST
STREET	VARCHAR(50)		
CITY	VARCHAR(30)		
STATE	VARCHAR(30)		
ZIP	VARCHAR(10)		

2. PATIENT_HISTORY

PATIENT HISTORY			
Name	**Type**	**Constraint**	**Domain**
RECORD_ID	INT	PRIMARY KEY AUTO_INCREMENT	
CREATION_TIMESTAMP	TIMESTAMP	NOT NULL	
PATIENT_ID	INT	FOREIGN KEY USER(USER_ID) NOT NULL	
DATE_OF_BIRTH	DATE	NOT NULL	
GENDER	VARCHAR(10)	NOT NULL	
PAST_DISEASES_DETAILS	VARCHAR(200)	DEFAULT N/A	
PAST_SURGERY_DETAILS	VARCHAR(200)	DEFAULT N/A	
ALLERGIES_DETAILS	VARCHAR(200)	DEFAULT N/A	
PAST_MEDICATION_DETAILS	VARCHAR(200)	DEFAULT N/A	
PAST_DOCTOR_REFERENCE	VARCHAR(100)	DEFAULT N/A	

3. DEPARTMENT

DEPARTMENT			
Name	**Type**	**Constraint**	**Domain**
DEPT_ID	INT	PRIMARY KEY	
DEPT_NAME	VARCHAR(25)	NOT NULL	

4. **SHIFT_DETAILS**

SHIFT_DETAILS			
Name	**Type**	**Constraint**	**Domain**
SHIFT_ID	INT	PRIMARY KEY	
START_TIME	VARCHAR(20)	NOT NULL	
END_TIME	VARCHAR(20)	NOT NULL	

5. **STAFF_ASSIGNMENT**

STAFF_ASSIGNMENT			
Name	**Type**	**Constraint**	**Domain**
ASSIGNMENT_ID	INT	PRIMARY KEY	
STAFF_ID	INT	UNIQUE KEY, FOREIGN KEY USER(USER_ID) NOT NULL	
DEPT_ID	INT	FOREIGN KEY DEPT(DEPT_ID) NOT NULL	
SHIFT_ID	INT	FOREIGN KEY SHIFT(SHIFT_ID) NOT NULL	

6. **WARD**

WARD			
Name	**Type**	**Constraint**	**Domain**
LOCATION_ID	INT	PRIMARY KEY	
ROOM_ID	INT	UNIQUE KEY, FOREIGN KEY WARD_CHARGE(ROOM_ID) NOT NULL	
BED_ID	INT	UNIQUE KEY DEFAULT 1	
ROOM_BED_STATUS	VARCHAR(15)	NOT NULL DEFAULT 'AVAILABLE'	AVAILABLE, OCCUPIED

7. **WARD_CHARGE**

WARD_CHARGE			
Name	**Type**	**Constraint**	**Domain**
ROOM_ID	INT	PRIMARY KEY	
ROOM_TYPE	VARCHAR(20)	NOT NULL	OT, GENERAL, TWIN-SHARING, PRIVATE
CHARGE	DOUBLE	NOT NULL	
CHARGE_UNIT	VARCHAR(20)	NOT NULL	
CREATION_TIMESTAMP	TIMESTAMP		
LAST_UPDATED	TIMESTAMP		

8. ADMITTED_PATIENT

ADMITTED_PATIENT			
Name	**Type**	**Constraint**	**Domain**
ADMIT_ID	INT	PRIMARY KEY AUTO_INCREMENT	
PATIENT_ID	INT	FOREIGN KEY USER(USER_ID) NOT NULL	
DOCTOR_ID	INT	FOREIGN KEY USER(USER_ID) NOT NULL	
ADMIT_TIMESTAMP	TIMESTAMP	NOT NULL	
LOCATION_ID	INT	FOREIGN KEY WARD(LOCATION_ID) NOT NULL	
DISCHARGE_DOCTOR_ID	INT	FOREIGN KEY USER(USER_ID)	
DISCHARGE_TIMESTAMP	TIMESTAMP		
DISCHARGE_NOTES	VARCHAR(100)		
CHARGE_ID	INT	FOREIGN KEY FEE_DETAILS(CHARGE_ID)	

9. PRESCRIPTION

PRESCRIPTION			
Name	**Type**	**Constraint**	**Domain**
PRESCRIPTION_ID	INT	PRIMARY_KEY AUTO_INCREMENT	
PRESCRIPTION_NOTES	VARCHAR(200)	NOT NULL	
PRESCRIPTION_STATUS	VARCHAR(10)	NOT NULL DEFAULT 'PENDING'	PENDING, APPROVED
PRESCRIPTION_AMOUNT	DOUBLE		

10. MEAL

MEAL			
Name	**Type**	**Constraint**	**Domain**
MEAL_ID	INT	PRIMARY KEY	
MEAL_DETAILS	VARCHAR(100)	NOT NULL	

11. DOC_INPATIENT_VISIT

DOC_INPATIENT_VISIT			
Name	**Type**	**Constraint**	**Domain**
VISIT_ID	INT	PRIMARY_KEY AUTO_INCREMENT	
ADMIT_ID	INT	FOREIGN KEY ADMITTED_PATIENT(ADMIT_ID) NOT NULL	
DOCTOR_ID	INT	FOREIGN KEY USER(USER_ID) NOT NULL	
VISIT_TIMESTAMP	TIMESTAMP	NOT NULL	
PRESCRIPTION_ID	INT	FOREIGN KEY PRESCRIPTION(PRESCRIPTION_ID)	
MEAL_ID	INT	FOREIGN KEY MEAL(MEAL_ID) NOT NULL	
MEAL_DETAILS_NOTES	VARCHAR(100)		
VISIT_NOTES	VARCHAR(100)		

12. OUTPATIENT_APPOINTMENT

OUTPATIENT_APPOINTMENT			
Name	**Type**	**Constraint**	**Domain**
APPOINTMENT_ID	INT	PRIMARY_KEY AUTO_INCREMENT	
PATIENT_ID	INT	FOREIGN KEY USER(USER_ID) NOT NULL	
APPOINTMENT_DATETIME	DATETIME	NOT NULL	
DOCTOR_ID	INT	FOREIGN KEY USER(USER_ID) NOT NULL	
APPOINTMENT_STATUS	VARCHAR(20)	NOT NULL DEFAULT 'SCHEDULED'	SCHEDULED, COMPLETED, CANCELLED

13. FEE_DETAILS

FEE_DETAILS			
Name	**Type**	**Constraint**	**Domain**
CHARGE_ID	INT	PRIMARY_KEY AUTO_INCREMENT	
CHARGE_TYPE	VARCHAR(50)	NOT NULL	
CHARGE_RATE	DOUBLE	NOT NULL	
CHARGE_UNIT	VARCHAR(10)	NOT NULL	PER DAY, PER HOUR, PER OPERATION

14. OUTPATIENT_VISIT

OUTPATIENT_VISIT			
Name	**Type**	**Constraint**	**Domain**
VISIT_ID	INT	PRIMARY_KEY AUTO_INCREMENT	
APPOINTMENT_ID	INT	FOREIGN KEY OUTPATIENT_APPOINTMENT(APPOINTMENT_ID) NOT NULL	
VISIT_NOTES	VARCHAR(100)		
PRESCRIPTION_DETAILS	VARCHAR(200)		
CHARGE_ID	INT	FOREIGN KEY FEE_DETAILS(CHARGE_ID) NOT NULL	

15. OT_SCHEDULING

OT_SCHEDULING			
Name	**Type**	**Constraint**	**Domain**
REQUEST_ID	INT	PRIMARY_KEY AUTO_INCREMENT	
STAFF_ID	INT	FOREIGN KEY USER(USER_ID) NOT NULL	
ROOM_ID	INT	FOREIGN KEY WARD(ROOM_ID) NOT NULL	
ADMIT_ID	INT	FOREIGN KEY ADMITTED_PATIENT(ADMIT_ID) NOT NULL	
START_DATETIME	DATETIME	NOT NULL	
END_DATETIME	DATETIME	NOT NULL	
OPERATION_ID	INT	FOREIGN KEY FEE_DETAILS(CHARGE_ID) NOT NULL	
DOCTOR_ID1	INT	FOREIGN KEY USER(USER_ID) NOT NULL	
DOCTOR_ID2	INT	FOREIGN KEY USER(USER_ID)	
DOCTOR_ID3	INT	FOREIGN KEY USER(USER_ID)	
REQUEST_STATUS	VARCHAR(20)	NOT NULL DEFAULT 'PENDING'	PENDING, APPROVED, COMPLETED, CANCELLED, DENIED

16. NURSE_VISIT

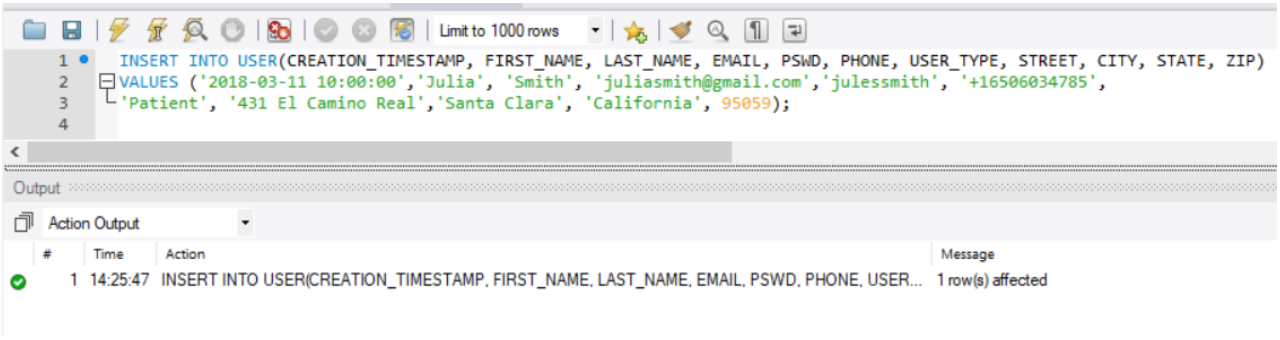
NURSE_VISIT			
Name	**Type**	**Constraint**	**Domain**
VISIT_ID	INT	PRIMARY_KEY AUTO_INCREMENT	
ADMIT_ID	INT	FOREIGN KEY ADMITTED_PATIENT(ADMIT_ID) NOT NULL	
NURSE_ID	INT	FOREIGN KEY USER(USER_ID) NOT NULL	
VISIT_TIMESTAMP	TIMESTAMP	NOT NULL	
VISIT_NOTES	VARCHAR(100)		

Queries

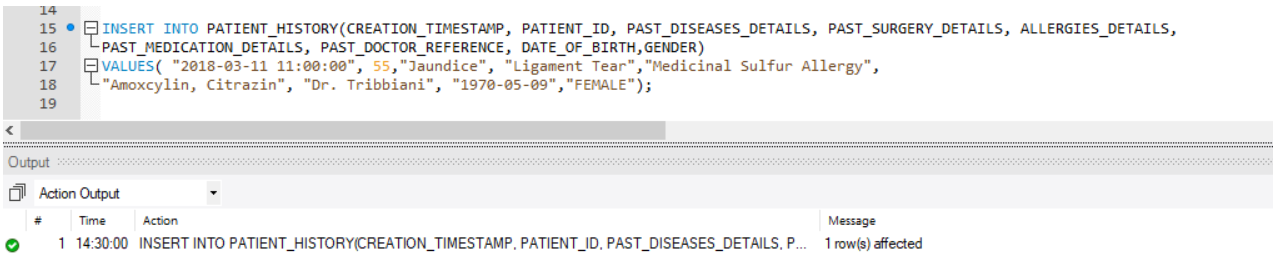
Before demonstrating queries associated with each user's functionality, we would like to present a patient flow beginning with a patient's first interaction with the system, getting admitted in the hospital, nurse visits and doctor visits associated with the admit, approval of prescription by the pharmacist, scheduling an OT for operation, discharge of the patient and finally generating a bill

For this purpose, we are considering a patient Julia Smith, a 48-year-old woman who has suffered from Jaundice and has undergone a surgery for Ligament tear in the past. She has an allergy of medicinal sulfur allergy and has been referred to our hospital by Dr. Tribbiani. She has been admitted in the hospital to undergo a Spine Surgery.

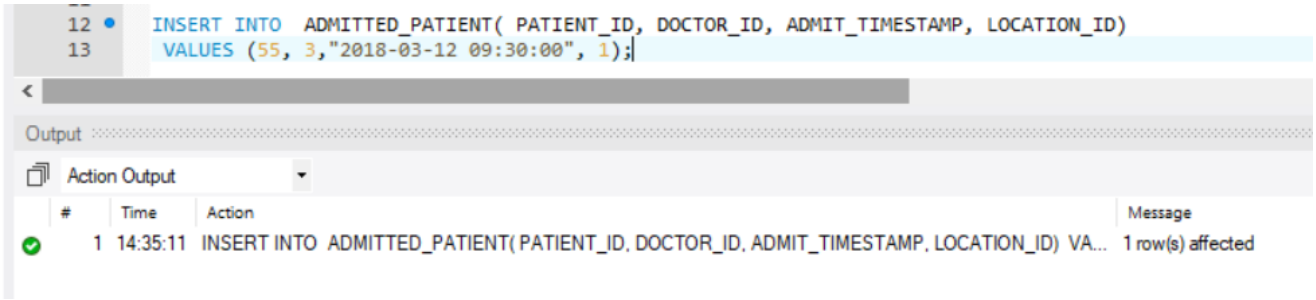
i) Julia's basic information is captured on first time registration with the system



ii) Julia enters her medical history



iii) Julia gets admitted in the hospital



iv) Record of Julia's nurse and doctor inpatient visits through the course of her admit duration

```
INSERT INTO  DOC_INPATIENT_VISIT( ADMIT_ID, DOCTOR_ID, VISIT_TIMESTAMP, PRESCRIPTION_ID, MEAL_ID, MEAL_DETAILS, VISIT_NOTES)
VALUES (11, 4,"2018-03-13 15:00:00", 3, 3, 'Regular-add omega 3','Stable' );
```

Output

Action Output

#	Time	Action	Message
1	14:50:08	INSERT INTO DOC_INPATIENT_VISIT(ADMIT_ID, DOCTOR_ID, VISIT_TIMESTAMP, PRESCRIPTION_ID, ...	1 row(s) affected

```
INSERT INTO  NURSE_VISIT( ADMIT_ID, NURSE_ID, VISIT_TIMESTAMP, VISIT_NOTES)
 VALUES (11, 9,"2018-03-13 10:30:00", 'Meal and Medicines given');

SELECT * FROM NURSE_VISIT WHERE ADMIT_ID=11;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

VISIT_ID	ADMIT_ID	NURSE_ID	VISIT_TIMESTAMP	VISIT_NOTES
122	11	9	2018-03-13 10:30:00	Meal and Medicines given
NULL	NULL	NULL	NULL	NULL

v) Approval of prescription by the pharmacist and updating pharmacy bill amount

PRESCRIPTION_ID	PRESCRIPTION_NOTES	PRESCRIPTION_STATUS	PRESCRIPTION_AMOUNT
2	Baclofen (Kemstro)-2. Econazole Nitrate (Spect...	PENDING	NULL
3	Edetate (Endrate)-1.Macugen (Pegaptanib Sodi...	PENDING	NULL
4	Paliperidone (Invega)-2	PENDING	NULL
5	Zanamivir (Relenza)-2. Pamidronate Disodium (...	PENDING	NULL
6	Bacteriostatic Saline (Bacteriostatic NaCl)-2	PENDING	NULL
7	Baver (Aspirin).Macugen (Pegaptanib Sodium)-2...	PENDING	NULL
8	Octreotide Acetate (Sandostatin)-2	PENDING	NULL
9	Baver (Aspirin)-2	PENDING	NULL
10	Pamidronate Disodium (Aredia)-2	PENDING	NULL
NULL	NULL	NULL	NULL

PRESCRIPTION 17

```
UPDATE PRESCRIPTION
SET PRESCRIPTION_AMOUNT=100, PRESCRIPTION_STATUS="APPROVED"
WHERE PRESCRIPTION_ID=3;

SELECT * FROM PRESCRIPTION;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

PRESCRIPTION_ID	PRESCRIPTION_NOTES	PRESCRIPTION_STATUS	PRESCRIPTION_AMOUNT
1	Zanamivir (Relenza)-2. H.P. Acthar Gel (Reposit...	PENDING	NULL
2	Baclofen (Kemstro)-2. Econazole Nitrate (Spect...	PENDING	NULL
3	Edetate (Endrate)-1.Macugen (Pegaptanib Sodi...	APPROVED	100
4	Paliperidone (Invega)-2	PENDING	NULL
5	Zanamivir (Relenza)-2. Pamidronate Disodium (...	PENDING	NULL
6	Bacteriostatic Saline (Bacteriostatic NaCl)-2	PENDING	NULL
7	Baver (Aspirin).Macugen (Pegaptanib Sodium)-2...	PENDING	NULL
8	Octreotide Acetate (Sandostatin)-2	PENDING	NULL
9	Baver (Aspirin)-2	PENDING	NULL
10	Pamidronate Disodium (Aredia)-2	PENDING	NULL

PRESCRIPTION 18

vi) Scheduling an OT for her operation and approval of operation schedule

```
14
15  INSERT INTO  OT_SCHEDULING( STAFF_ID, ROOM_ID, ADMIT_ID,
16  OPERATION_ID,START_DATETIME, END_DATETIME, DOCTOR_ID1,DOCTOR_ID2)
17  VALUES (4, 9, 1,9, "2018-03-13 13:00:00", "2018-03-13 15:00:00", 4,3);
```

Output

Action Output

#	Time	Action	Message
1	14:40:53	INSERT INTO OT_SCHEDULING(STAFF_ID, ROOM_ID, ADMIT_ID, OPERATION_ID,START_DATETIME...	1 row(s) affected

```
9  UPDATE OT_SCHEDULING
10 SET REQUEST_STATUS="Approved"
11 WHERE ADMIT_ID=11;
12
13 SELECT * FROM OT_SCHEDULING;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

REQUEST_ID	STAFF_ID	ROOM_ID	ADMIT_ID	OPERATION_ID	START_DATETIME	END_DATETIME	DOCTOR_ID1	DOCTOR_ID2	DOCTOR_ID3	REQUEST_STATUS
1	4	9	1	9	2018-03-10 13:00:00	2018-03-10 15:00:00	4	NULL	NULL	Approved
2	18	10	2	7	2018-03-11 14:30:00	2018-03-11 16:00:00	14	NULL	NULL	Approved
3	11	11	3	7	2018-03-15 15:00:00	2018-03-15 18:00:00	14	NULL	NULL	Pending
4	9	10	4	10	2018-02-27 10:00:00	2018-02-27 14:00:00	3	NULL	NULL	Completed
5	26	11	8	5	2018-01-03 13:00:00	2018-01-03 14:30:00	15	NULL	NULL	Completed
6	4	9	1	10	2018-03-13 13:00:00	2018-03-13 15:00:00	4	3	NULL	Pending
7	4	9	11	10	2018-03-13 13:00:00	2018-03-13 15:00:00	4	3	NULL	Approved
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

OT_SCHEDULING 14 ×

Apply

vii) Discharging Julia from the hospital

```
26 UPDATE ADMITTED_PATIENT
27 SET DISCHARGE_DOCTOR_ID=7, DISCHARGE_TIMESTAMP="2018-03-15 09:30:00",
28 DISCHARGE_NOTES='BP in normal range. Blood test report required on next visit'
29 WHERE PATIENT_ID =55;
```

Output

Action Output

#	Time	Action	Message
1	14:40:53	INSERT INTO OT_SCHEDULING(STAFF_ID, ROOM_ID, ADMIT_ID, OPERATION_ID,START_DATETIME...	1 row(s) affected
2	14:45:32	UPDATE ADMITTED_PATIENT SET DISCHARGE_DOCTOR_ID=7, DISCHARGE_TIMESTAMP="2018-03-1...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

viii) Generating the total bill for Julia on discharge which includes per day charge for ward, per operation charge, prescription charges and per hour charge for the operation theatre used

```
57
58  SELECT R1.ADMIT_ID AS 'ADMIT ID' ,SUM(R1.CHARGE)AS 'TOTAL BILL AMOUNT'
59  FROM(
60  (SELECT AP.ADMIT_ID, TIMESTAMPDIFF(DAY,AP.ADMIT_TIMESTAMP, AP.DISCHARGE_TIMESTAMP) * FD.CHARGE_RATE AS 'CHARGE'
61  FROM ADMITTED_PATIENT AP, FEE_DETAILS FD
62  WHERE AP.CHARGE_ID=FD.CHARGE_ID AND AP.DISCHARGE_DOCTOR_ID IS NOT NULL
63  AND AP.ADMIT_ID=11)
64  UNION
65  (SELECT DISTINCT DV.ADMIT_ID, P.PRESCRIPTION_AMOUNT AS 'CHARGE'
66  FROM DOC_INPATIENT_VISIT DV, PRESCRIPTION P, ADMITTED_PATIENT AP
67  WHERE AP.DISCHARGE_DOCTOR_ID IS NOT NULL AND
68        DV.ADMIT_ID=AP.ADMIT_ID AND
69        P.PRESCRIPTION_ID=DV.PRESCRIPTION_ID
70        AND DV.ADMIT_ID=11)
71  UNION
72  (SELECT O.ADMIT_ID, FD.CHARGE_RATE AS 'CHARGE'
73  FROM ADMITTED_PATIENT AP, OT_SCHEDULING O, FEE_DETAILS FD
74  WHERE AP.DISCHARGE_DOCTOR_ID IS NOT NULL
75        AND AP.ADMIT_ID=O.ADMIT_ID
76        AND O.OPERATION_ID= FD.CHARGE_ID AND
77        O.ADMIT_ID=11
78      )
79   UNION
80  (SELECT O.ADMIT_ID, TIMESTAMPDIFF(HOUR,O.START_DATETIME,O.END_DATETIME)*WC.CHARGE AS 'CHARGE'
81   FROM OT_SCHEDULING O, ADMITTED_PATIENT AP, WARD_CHARGE WC
82   WHERE AP.DISCHARGE_DOCTOR_ID IS NOT NULL
83         AND O.ADMIT_ID=AP.ADMIT_ID AND
84         O.ROOM_ID=WC.ROOM_ID AND
85         O.ADMIT_ID=11
86  )) AS R1;
87
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ADMIT ID	TOTAL BILL AMOUNT
11	20100

Doctor

i) Each Doctor can view his/her own schedule

```
#**
CREATE VIEW VIEW_STAFF_SCHEDULE AS
SELECT U.USER_ID,  U.FIRST_NAME, U.LAST_NAME, U.USER_TYPE, D.DEPT_NAME, SD.START_TIME, SD.END_TIME
FROM USER U, SHIFT_DETAILS SD, DEPARTMENT D, STAFF_ASSIGNMENT SA
WHERE U.USER_ID=SA.STAFF_ID
  AND SA.DEPT_ID=D.DEPT_ID
  AND SA.SHIFT_ID=SD.SHIFT_ID;

#DOCTOR VIEWS HIS OWN SCHEDULE
SELECT * FROM VIEW_STAFF_SCHEDULE
 WHERE USER_ID=3;
```

100% 32:14

Result Grid | Filter Rows: Search | Export:

USER_ID	FIRST_NAME	LAST_NAME	USER_TYPE	DEPT_NAME	START_TIME	END_TIME
3	Christina	Williams	Doctor	Orthopaedics	5:30	15:00

ii) Each Doctor can view his/her upcoming appointments

```
CREATE VIEW VIEW_OUTPATIENT_APPOINTMENTS AS
SELECT OA.APPOINTMENT_ID, OA.PATIENT_ID, OA.DOCTOR_ID , U1.FIRST_NAME, U1.LAST_NAME,
OA.APPOINTMENT_DATETIME, OA.APPOINTMENT_STATUS
FROM OUTPATIENT_APPOINTMENT OA, USER U, USER U1
WHERE OA.DOCTOR_ID = U.USER_ID AND OA.PATIENT_ID = U1.USER_ID;

#DOCTOR CAN VIEW OUTPATIENT APPOINMENTS: SCHEDULED, CANCELLED OR COMPLETED
SELECT PATIENT_ID AS 'Patient ID', CONCAT(FIRST_NAME ," ",LAST_NAME) AS 'Patient Name', APPOINTMENT_DATETIME, APPOINTMENT_STATUS
FROM VIEW_OUTPATIENT_APPOINTMENTS
WHERE Doctor_ID= 13 AND (APPOINTMENT_STATUS='Cancelled' OR APPOINTMENT_STATUS='Scheduled');
```

100% 75:26

Result Grid | Filter Rows: Search | Export:

Patient ID	Patient Name	APPOINTMENT_DATETIME	APPOINTMENT_STATUS
42	Cameron Jack	2018-03-20 11:00:00	Scheduled
36	Lip Gallagher	2018-03-10 10:00:00	Cancelled

iii) Each Doctor can view his/her upcoming operations and their status if approved or pending

```
#DOCTOR CAN CHECK UPCOMING OPERATIONS WHICH ARE APPROVED OR PENDING APPROVAL
SELECT CONCAT(U.FIRST_NAME, " ", U.LAST_NAME) AS 'Patient Name', FD.CHARGE_TYPE AS 'Operation Type',
OS.ROOM_ID AS 'OT Room', OS.START_DATETIME, OS.END_DATETIME, OS.REQUEST_STATUS
FROM OT_SCHEDULING AS OS, ADMITTED_PATIENT AP, USER U,FEE_DETAILS FD
WHERE OS.ADMIT_ID= AP.ADMIT_ID AND AP.PATIENT_ID= U.USER_ID AND FD.CHARGE_ID = OS.OPERATION_ID AND OS.DOCTOR_ID1= 14;
```

100% 2:38

Result Grid | Filter Rows: Search | Export:

Patient Name	Operation Type	OT Room	START_DATETIME	END_DATETIME	REQUEST_STATUS
Jeremy Allen	Lymphadenectomy	10	2018-03-11 14:30:00	2018-03-11 16:00:00	Approved
Vincent Gogh	Lymphadenectomy	11	2018-03-15 15:00:00	2018-03-15 18:00:00	Pending

iv) Doctors can update their schedule and view their updated schedule

```sql
#DOCTOR UPDATE HIS AVAILABILITY
UPDATE STAFF_ASSIGNMENT
SET SHIFT_ID=2 WHERE STAFF_ID=55;

 #VIEW UPDATED SHIFT DETAILS OF A DOCTOR
 SELECT STAFF_ID, CONCAT(U.FIRST_NAME," ",U.LAST_NAME)AS 'DOCTOR NAME', SD.START_TIME AS 'SHIFT START TIME',
 SD.END_TIME AS 'SHIFT END TIME'
 FROM STAFF_ASSIGNMENT SA, SHIFT_DETAILS SD, USER U
 WHERE SA.STAFF_ID=U.USER_ID
       AND SA.SHIFT_ID=SD.SHIFT_ID
       AND STAFF_ID=55;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

STAFF_ID	DOCTOR NAME	SHIFT START TIME	SHIFT END TIME
55	Jimmy Gallo	14:30	23:00

Admin

i) Admin can view schedule of all nurses

```sql
#
CREATE VIEW VIEW_STAFF_SCHEDULE AS
SELECT U.USER_ID,  U.FIRST_NAME, U.LAST_NAME, U.USER_TYPE, D.DEPT_NAME, SD.START_TIME, SD.END_TIME
FROM USER U, SHIFT_DETAILS SD, DEPARTMENT D, STAFF_ASSIGNMENT SA
WHERE U.USER_ID=SA.STAFF_ID
  AND SA.DEPT_ID=D.DEPT_ID
  AND SA.SHIFT_ID=SD.SHIFT_ID;

 #ADMIN VIEWS SCHEDULE OF ALL NURSES
SELECT * FROM VIEW_STAFF_SCHEDULE
 WHERE USER_TYPE='Nurse';
```

100% 37:14

Result Grid | Filter Rows: Search | Export:

USER_ID	FIRST_NAME	LAST_NAME	USER_TYPE	DEPT_NAME	START_TIME	END_TIME
9	Frank	Abel	Nurse	Orthopaedics	5:30	15:00
12	Monica	Geller	Nurse	Neurology	5:30	15:00
20	Adams	Douglas	Nurse	General surgery	5:30	15:00
23	Jessica	Pearson	Nurse	Oncology	5:30	15:00
26	Nigel	Frances	Nurse	Cardiology	5:30	15:00
10	John	Held	Nurse	Orthopaedics	14:30	23:00
18	Amelia	Jones	Nurse	Neurology	14:30	23:00
21	Richard	Castle	Nurse	General surgery	14:30	23:00
24	Heidi	Louis	Nurse	Oncology	14:30	23:00
27	Bret	Lee	Nurse	Cardiology	14:30	23:00
11	Jon	Snow	Nurse	Orthopaedics	22:30	06:00
19	Sonia	Craig	Nurse	Neurology	22:30	06:00
22	Vitoria	Jane	Nurse	General surgery	22:30	06:00
25	Edwin	Peterson	Nurse	Oncology	22:30	06:00
28	George	Fox	Nurse	Cardiology	22:30	06:00

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

ii) Admin can generate total bill of a particular patient

```
#**TO GENERATE TOTAL BILL AMOUNT FOR ALL ADMITTED PATIENTS ON DISCHARGE
CREATE VIEW VIEW_TOTAL_BILL AS
SELECT R1.ADMIT_ID ,SUM(R1.CHARGE)AS 'TOTAL_BILL_AMOUNT',R1.DISCHARGE_TIMESTAMP, R1.PATIENT_ID, R1.DOCTOR_ID
FROM(
(SELECT AP.ADMIT_ID, TIMESTAMPDIFF(DAY,AP.ADMIT_TIMESTAMP, AP.DISCHARGE_TIMESTAMP) * FD.CHARGE_RATE AS 'CHARGE',
AP.DISCHARGE_TIMESTAMP, AP.PATIENT_ID, AP.DOCTOR_ID
FROM ADMITTED_PATIENT AP, FEE_DETAILS FD
WHERE AP.CHARGE_ID=FD.CHARGE_ID AND AP.DISCHARGE_DOCTOR_ID IS NOT NULL)
UNION
(SELECT DISTINCT DV.ADMIT_ID, P.PRESCRIPTION_AMOUNT AS 'CHARGE', AP.DISCHARGE_TIMESTAMP, AP.PATIENT_ID, AP.DOCTOR_ID
FROM DOC_INPATIENT_VISIT DV, PRESCRIPTION P, ADMITTED_PATIENT AP
WHERE AP.DISCHARGE_DOCTOR_ID IS NOT NULL AND
      DV.ADMIT_ID=AP.ADMIT_ID AND P.PRESCRIPTION_ID=DV.PRESCRIPTION_ID)
UNION
(SELECT O.ADMIT_ID, FD.CHARGE_RATE AS 'CHARGE', AP.DISCHARGE_TIMESTAMP, AP.PATIENT_ID, AP.DOCTOR_ID
FROM ADMITTED_PATIENT AP, OT_SCHEDULING O, FEE_DETAILS FD
WHERE AP.DISCHARGE_DOCTOR_ID IS NOT NULL AND AP.ADMIT_ID=O.ADMIT_ID AND O.OPERATION_ID= FD.CHARGE_ID
    )
 UNION
 (SELECT O.ADMIT_ID, TIMESTAMPDIFF(HOUR,O.START_DATETIME,O.END_DATETIME)*WC.CHARGE AS 'CHARGE', AP.DISCHARGE_TIMESTAMP,
 AP.PATIENT_ID, AP.DOCTOR_ID
 FROM OT_SCHEDULING O, ADMITTED_PATIENT AP, WARD_CHARGE WC
 WHERE AP.DISCHARGE_DOCTOR_ID IS NOT NULL AND O.ADMIT_ID=AP.ADMIT_ID AND O.ROOM_ID=WC.ROOM_ID
 )) AS R1
 GROUP BY R1.ADMIT_ID;
```

```
#**VIEW TOTAL BILL OF AN ADMITTED PATIENT ON DISCHARGE
SELECT V.ADMIT_ID, CONCAT(U.FIRST_NAME," ", U.LAST_NAME) AS 'Patient Name' ,V.TOTAL_BILL_AMOUNT as 'Total Bill Amount(in $)'
FROM VIEW_TOTAL_BILL V, USER U
WHERE V.PATIENT_ID= U.USER_ID
    AND V.ADMIT_ID=8;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ADMIT_ID	Patient Name	Total Bill Amount(in $)
8	Noel Fisher	34640

iii) Admin can view current occupancy rate for all rooms and beds

```
#**TO VIEW CURRENT OCCUPANCY RATE
SELECT R2.NO_OCCUPIED_ROOMS/R1.NO_TOTAL_ROOMS *100 AS 'OCCUPANCY RATE (%)'
FROM(SELECT COUNT(W.LOCATION_ID) AS 'NO_TOTAL_ROOMS'
FROM WARD W, WARD_CHARGE WC
WHERE W.ROOM_ID=WC.ROOM_ID
      AND ROOM_TYPE<>"OT") R1,

(SELECT COUNT(W1.LOCATION_ID) AS 'NO_OCCUPIED_ROOMS'
FROM WARD W1
WHERE W1.ROOM_BED_STATUS="OCCUPIED"
) R2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

OCCUPANCY RATE (%)
23.5294

Nurse

i) Nurse can view the most recent doctor visit notes, meal details and prescription

```
#NURSE VIEWS DOCTOR NOTES, MEAL PLAN DETAILS AND PRESCRIPTION OF MOST RECENT DOCTOR VISIT
SELECT DV.ADMIT_ID, CONCAT(U.FIRST_NAME," ",U.LAST_NAME)AS 'PATIENT NAME', DV.VISIT_NOTES AS 'DOC VISIT NOTES',
M.MEAL_DETAILS AS 'MEAL PLAN', DV.MEAL_DETAILS 'MEAL NOTES', P.PRESCRIPTION_NOTES AS 'PRESCRIPTION NOTES'
FROM DOC_INPATIENT_VISIT DV, USER U, MEAL M, ADMITTED_PATIENT AP, PRESCRIPTION P
WHERE AP.PATIENT_ID=U.USER_ID AND
     DV.ADMIT_ID=AP.ADMIT_ID AND
     DV.MEAL_ID= M.MEAL_ID AND
     P.PRESCRIPTION_ID= DV.PRESCRIPTION_ID AND
    DV.ADMIT_ID=2 AND
    DV.VISIT_TIMESTAMP =(SELECT MAX(DV2.VISIT_TIMESTAMP)
                         FROM DOC_INPATIENT_VISIT DV2 WHERE DV2.ADMIT_ID=2)
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ADMIT_ID	PATIENT NAME	DOC VISIT NOTES	MEAL PLAN	MEAL NOTES	PRESCRIPTION NOTES
2	Jeremy Allen	Monitor closely	Vegan Diet	Regular-add omega 3	Bayer (Aspirin),Macugen (Pegaptanib Sodium)-2...

ii) Nurse can view his/her own schedule

```
#**
CREATE VIEW VIEW_STAFF_SCHEDULE AS
SELECT U.USER_ID, U.FIRST_NAME, U.LAST_NAME, U.USER_TYPE, D.DEPT_NAME, SD.START_TIME, SD.END_TIME
FROM USER U, SHIFT_DETAILS SD, DEPARTMENT D, STAFF_ASSIGNMENT SA
WHERE U.USER_ID=SA.STAFF_ID
  AND SA.DEPT_ID=D.DEPT_ID
  AND SA.SHIFT_ID=SD.SHIFT_ID;

#NURSE VIEWS HIS/HER OWN SCHEDULE
SELECT * FROM VIEW_STAFF_SCHEDULE
 WHERE USER_ID=9;
```

100% 1:19

Result Grid | Filter Rows: Search | Export:

USER_ID	FIRST_NAME	LAST_NAME	USER_TYPE	DEPT_NAME	START_TIME	END_TIME
9	Frank	Abel	Nurse	Orthopaedics	5:30	15:00

Result Grid | Form Editor

Pharmacist

i) Pharmacist can view all pending prescriptions

```
164  #PHARMACISTS VIEW PENDING PRESCRIPTION REQUEST
165  SELECT DISTINCT P.PRESCRIPTION_ID , CONCAT(U.FIRST_NAME," ",U.LAST_NAME)AS 'PATIENT NAME',
166  P.PRESCRIPTION_NOTES AS'PRESCRIPTION', P.PRESCRIPTION_STATUS AS 'STATUS'
167  FROM PRESCRIPTION P , DOC_INPATIENT_VISIT DV, ADMITTED_PATIENT AP, USER U
168  WHERE DV.PRESCRIPTION_ID=P.PRESCRIPTION_ID AND AP.ADMIT_ID=DV.ADMIT_ID
169        AND AP.PATIENT_ID=U.USER_ID
170        AND P.PRESCRIPTION_STATUS='PENDING';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PRESCRIPTION_ID	PATIENT NAME	PRESCRIPTION	STATUS
3	Michael Jordan	Edetate (Endrate)-1.Macugen (Pegaptanib Sodi...	PENDING
7	Jeremy Allen	Bayer (Aspirin).Macugen (Pegaptanib Sodium)-2...	PENDING
5	Dhiren RIkhra	Zanamivir (Relenza)-2. Pamidronate Disodium (...	PENDING
2	Noel Fisher	Baclofen (Kemstro)-2. Econazole Nitrate (Spect...	PENDING
5	Noel Fisher	Zanamivir (Relenza)-2. Pamidronate Disodium (...	PENDING
4	Johnie Walker	Paliperidone (Invega)-2	PENDING

ii) Pharmacist approves the prescription and inserts the prescription amount

```
172  #PHARMACISTS APPROVES THE PRESCRIPTION AND ENTERS AMOUNT FOR IT
173  UPDATE PRESCRIPTION
174  SET PRESCRIPTION_STATUS='APPROVED', PRESCRIPTION_AMOUNT=70.00
175  WHERE PRESCRIPTION_ID=3;
176
177
178  SELECT DISTINCT P.PRESCRIPTION_ID , CONCAT(U.FIRST_NAME," ",U.LAST_NAME)AS 'PATIENT NAME',
179  P.PRESCRIPTION_NOTES AS'PRESCRIPTION', P.PRESCRIPTION_STATUS AS 'STATUS', PRESCRIPTION_AMOUNT AS 'PRESCRIPTION BILL'
180  FROM PRESCRIPTION P , DOC_INPATIENT_VISIT DV, ADMITTED_PATIENT AP, USER U
181  WHERE DV.PRESCRIPTION_ID=P.PRESCRIPTION_ID AND AP.ADMIT_ID=DV.ADMIT_ID
182        AND AP.PATIENT_ID=U.USER_ID
183        AND P.PRESCRIPTION_STATUS='APPROVED' AND P.PRESCRIPTION_ID=3;
184
```

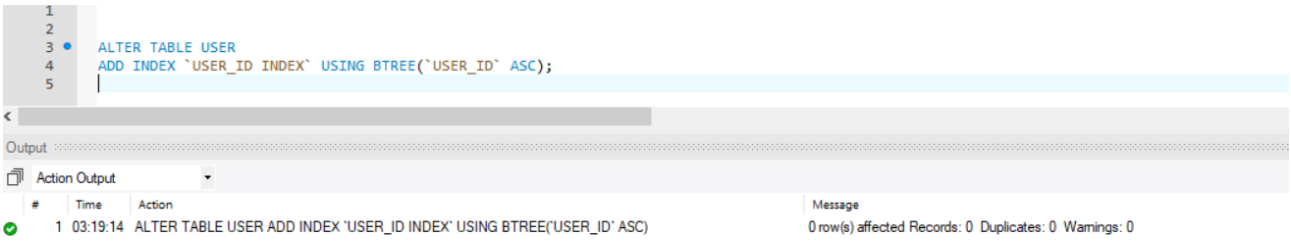
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PRESCRIPTION_ID	PATIENT NAME	PRESCRIPTION	STATUS	PRESCRIPTION BILL
3	Michael Jordan	Edetate (Endrate)-1.Macugen (Pegaptanib Sodi...	APPROVED	70

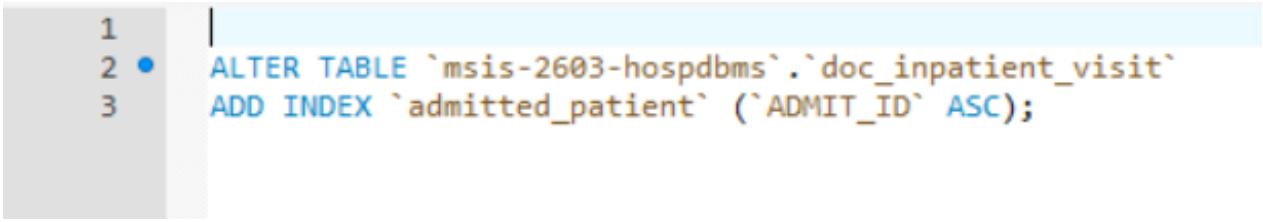
Index

1. **Index on the relation DOC_INPATIENT_VISIT on the attribute VISIT_ID**

2. **Index on the relation USER on the attribute USER_ID**



3. **Index on the relation DOC_INPATIENT_VISIT on the attribute ADMIT_ID**

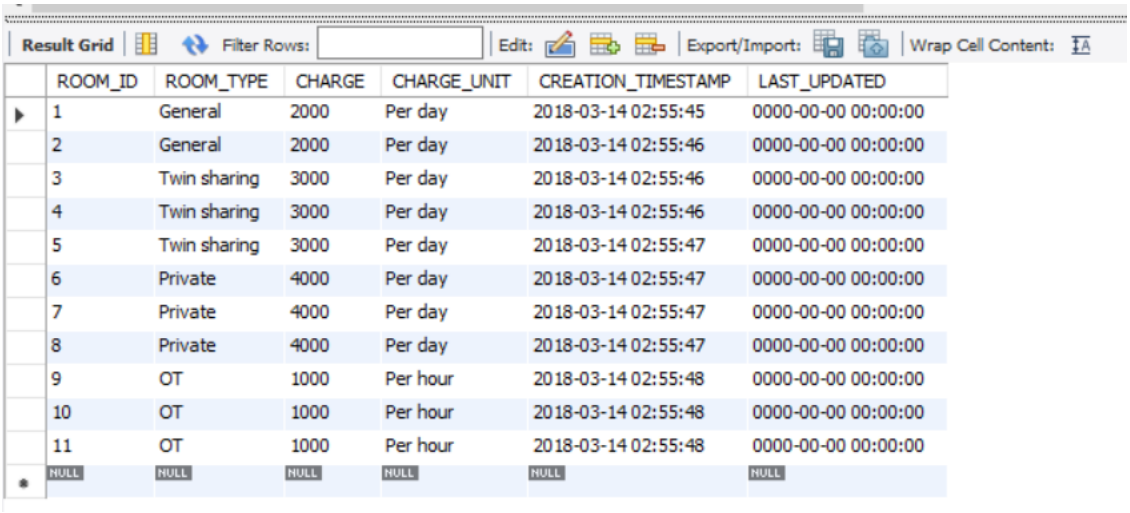


Triggers

1. **Capture the creation timestamp whenever a new ward charge is inserted**

```
DELIMITER $$
USE `msis-2603-project-data-tribe`$$
CREATE DEFINER = CURRENT_USER TRIGGER `msis-2603-project-data-tribe`.`ward_charge_BEFORE_INSERT`
BEFORE INSERT ON `ward_charge` FOR EACH ROW
BEGIN
SET NEW.CREATION_TIMESTAMP = NOW();
END$$
DELIMITER ;
```

On insertion, creation timestamp is captured:



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ROOM_ID	ROOM_TYPE	CHARGE	CHARGE_UNIT	CREATION_TIMESTAMP	LAST_UPDATED
1	General	2000	Per day	2018-03-14 02:55:45	0000-00-00 00:00:00
2	General	2000	Per day	2018-03-14 02:55:46	0000-00-00 00:00:00
3	Twin sharing	3000	Per day	2018-03-14 02:55:46	0000-00-00 00:00:00
4	Twin sharing	3000	Per day	2018-03-14 02:55:46	0000-00-00 00:00:00
5	Twin sharing	3000	Per day	2018-03-14 02:55:47	0000-00-00 00:00:00
6	Private	4000	Per day	2018-03-14 02:55:47	0000-00-00 00:00:00
7	Private	4000	Per day	2018-03-14 02:55:47	0000-00-00 00:00:00
8	Private	4000	Per day	2018-03-14 02:55:47	0000-00-00 00:00:00
9	OT	1000	Per hour	2018-03-14 02:55:48	0000-00-00 00:00:00
10	OT	1000	Per hour	2018-03-14 02:55:48	0000-00-00 00:00:00
11	OT	1000	Per hour	2018-03-14 02:55:48	0000-00-00 00:00:00
NULL	NULL	NULL	NULL	NULL	NULL

2. **Capture the last updated timestamp whenever a ward charge is updated**

```
DROP TRIGGER IF EXISTS `msis-2603-project-data-tribe`.`ward_charge_BEFORE_UPDATE`;

DELIMITER $$
USE `msis-2603-project-data-tribe`$$
CREATE DEFINER = CURRENT_USER TRIGGER `msis-2603-project-data-tribe`.`ward_charge_BEFORE_UPDATE`
BEFORE UPDATE ON `ward_charge` FOR EACH ROW
BEGIN
SET new.LAST_UPDATED = NOW();
END$$
DELIMITER ;
```

On updating, the last updated timestamp is captured:

```
UPDATE WARD_CHARGE
SET CHARGE=5000
WHERE ROOM_TYPE="Private";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ROOM_ID	ROOM_TYPE	CHARGE	CHARGE_UNIT	CREATION_TIMESTAMP	LAST_UPDATED
1	General	2000	Per day	2018-03-14 02:55:45	0000-00-00 00:00:00
2	General	2000	Per day	2018-03-14 02:55:46	0000-00-00 00:00:00
3	Twin sharing	3000	Per day	2018-03-14 02:55:46	0000-00-00 00:00:00
4	Twin sharing	3000	Per day	2018-03-14 02:55:46	0000-00-00 00:00:00
5	Twin sharing	3000	Per day	2018-03-14 02:55:47	0000-00-00 00:00:00
6	Private	5000	Per day	2018-03-14 03:05:18	2018-03-14 03:05:18
7	Private	5000	Per day	2018-03-14 03:05:18	2018-03-14 03:05:18
8	Private	5000	Per day	2018-03-14 03:05:18	2018-03-14 03:05:18
9	OT	1000	Per hour	2018-03-14 02:55:48	0000-00-00 00:00:00
10	OT	1000	Per hour	2018-03-14 02:55:48	0000-00-00 00:00:00
11	OT	1000	Per hour	2018-03-14 02:55:48	0000-00-00 00:00:00
NULL	NULL	NULL	NULL	NULL	NULL

```
UPDATE WARD_CHARGE
SET CHARGE=1500
WHERE ROOM_TYPE="OT";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

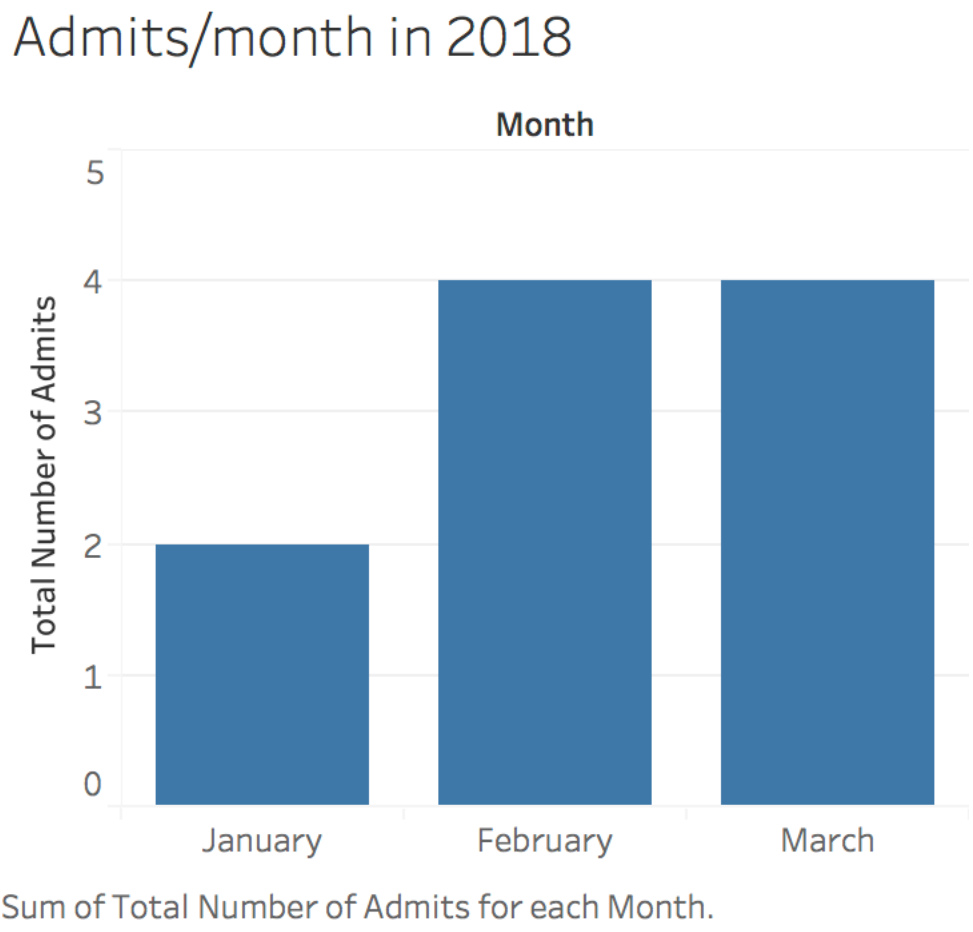
ROOM_ID	ROOM_TYPE	CHARGE	CHARGE_UNIT	CREATION_TIMESTAMP	LAST_UPDATED
1	General	2000	Per day	2018-03-14 02:55:45	0000-00-00 00:00:00
2	General	2000	Per day	2018-03-14 02:55:46	0000-00-00 00:00:00
3	Twin sharing	3000	Per day	2018-03-14 02:55:46	0000-00-00 00:00:00
4	Twin sharing	3000	Per day	2018-03-14 02:55:46	0000-00-00 00:00:00
5	Twin sharing	3000	Per day	2018-03-14 02:55:47	0000-00-00 00:00:00
6	Private	5000	Per day	2018-03-14 03:05:18	2018-03-14 03:05:18
7	Private	5000	Per day	2018-03-14 03:05:18	2018-03-14 03:05:18
8	Private	5000	Per day	2018-03-14 03:05:18	2018-03-14 03:05:18
9	OT	1500	Per hour	2018-03-14 03:07:07	2018-03-14 03:07:07
10	OT	1500	Per hour	2018-03-14 03:07:07	2018-03-14 03:07:07
11	OT	1500	Per hour	2018-03-14 03:07:07	2018-03-14 03:07:07
NULL	NULL	NULL	NULL	NULL	NULL

1. **Hospital can visualize the number of admits per month**

```
99    #** NO. OF PATIENTS ADMITTED MONTHLY
100   SELECT MONTHNAME(ADMIT_TIMESTAMP) AS 'MONTH' ,COUNT(ADMIT_ID) AS 'TOTAL NO. OF ADMITS'
101   FROM ADMITTED_PATIENT
102   GROUP BY MONTH(ADMIT_TIMESTAMP);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MONTH	TOTAL NO. OF ADMITS
Januarv	2
Februarv	4
March	4



Sum of Total Number of Admits for each Month.

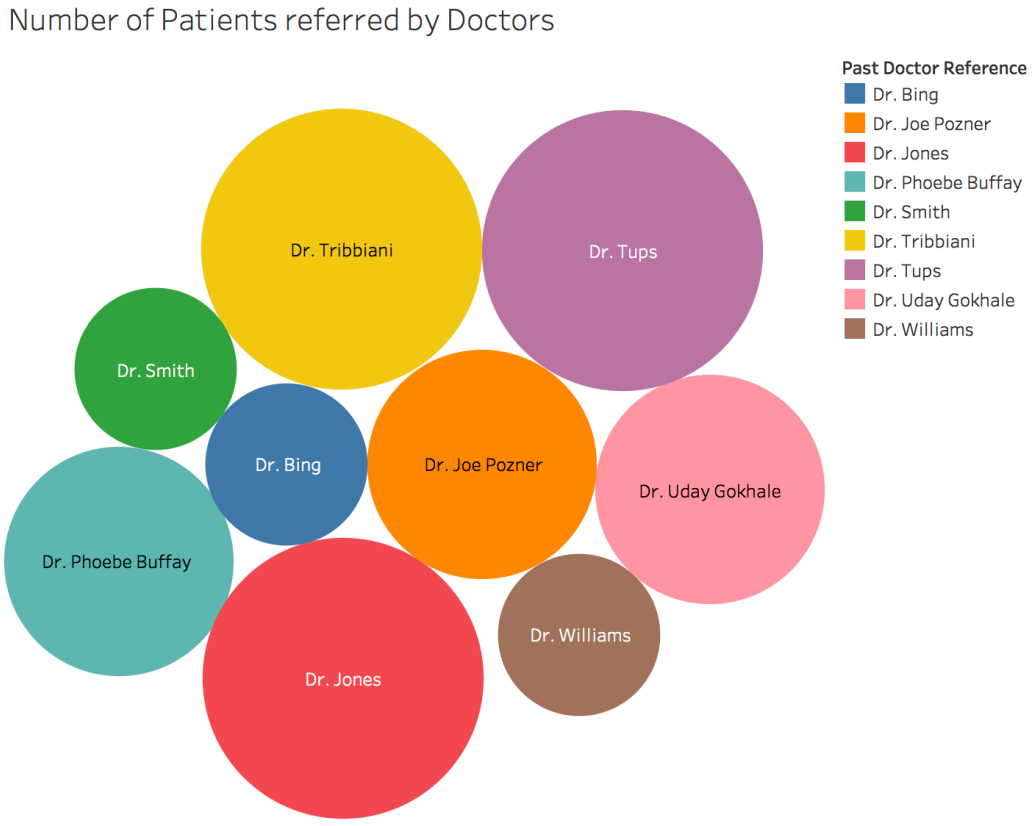
2. Hospital can visualize which doctors are referring patients to their hospital, patient's gender and age, month-wise segmentation of different revenue streams and geographic distribution of patients

```
43
44  SELECT CONCAT(U.FIRST_NAME, " ", U.LAST_NAME) AS 'PATIENT NAME', P.PAST_DOCTOR_REFERENCE,
45  YEAR(CURRENT_TIMESTAMP) - YEAR(P.DATE_OF_BIRTH) AS 'AGE',  P.GENDER, U.CITY, U.STATE
46  FROM USER U, PATIENT_HISTORY P
47  WHERE U.USER_ID = P.PATIENT_ID;
48
49
```

100% 1:49

Result Grid | Filter Rows: Search | Export:

PATIENT NAME	PAST_DOCTOR_REFERENCE	AGE	GENDER	CITY	STATE
Michael Jordan	Dr. Tups	38	MALE	San Francisco	California
Vincent Gogh	Dr. Uday Gokhale	48	MALE	San Francisco	California
Winnie Pooh	Dr. Smith	19	FEMALE	San Francisco	California
Peter Craig	Dr. Uday Gokhale	34	MALE	San Francisco	California
Dhiren Rlkhra	Dr. Jones	27	MALE	San Mateo	California
Jim Green	Dr. Tups	28	FEMALE	Santa Cruz	California
Frank Gallagher	Dr. Phoebe Buffay	53	MALE	Oakland	California
Lip Gallagher	Dr. Tups	58	MALE	San Jose	California
Fiona Chesney	Dr. Williams	25	FEMALE	Santa Clara	California
William Macy	Dr. Tribbiani	20	MALE	Fremont	California
Emma Kenney	Dr. Joe Pozner	23	FEMALE	Los Angeles	California
Noel Fisher	Dr. Bing	53	FEMALE	Oakland	California
Jeremy Allen	Dr. Tribbiani	18	MALE	Santa Clara	California
Cameron Jack	Dr. Tribbiani	10	FEMALE	Santa Clara	California
Mickey Milkovic	Dr. Phoebe Buffay	58	FEMALE	San Jose	California
Jack Daniels	Dr. Jones	58	MALE	Fremont	California
Jim Beam	Dr. Jones	48	MALE	Oakland	California
Johnie Walker	Dr. Joe Pozner	43	MALE	Santa Clara	California

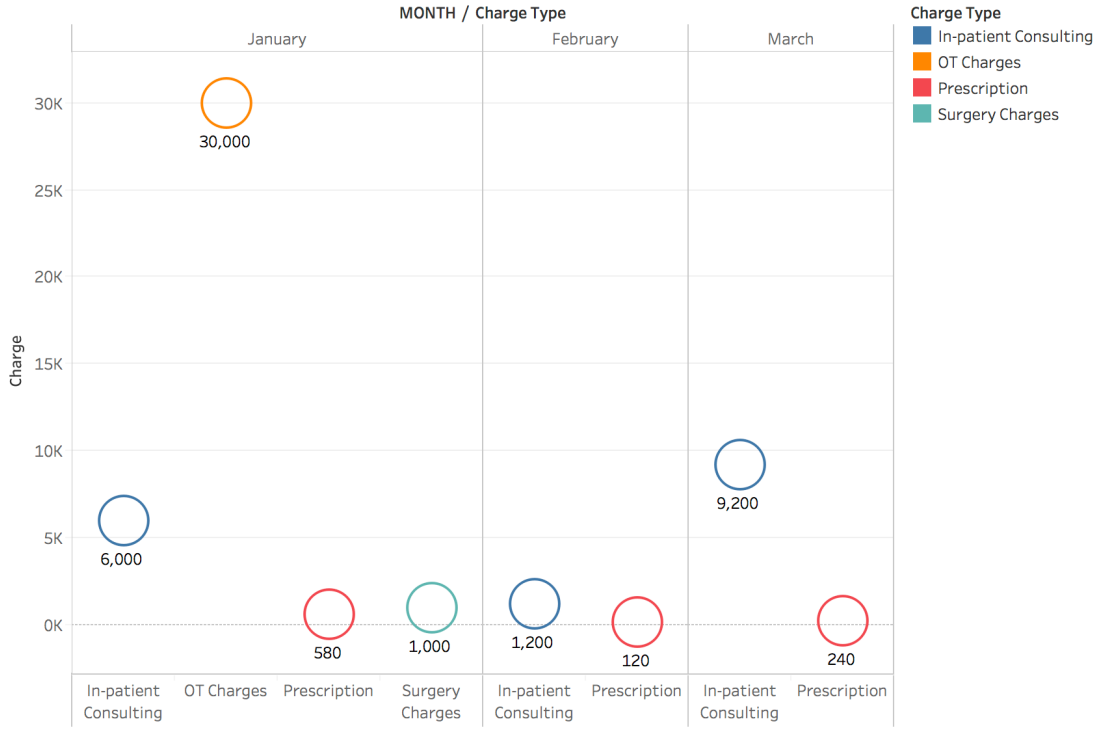


Past Doctor Reference. Color shows details about Past Doctor Reference. Size shows count of Patient Name. The marks are labeled by Past Doctor Reference.

Patient Gender and Age Visualization

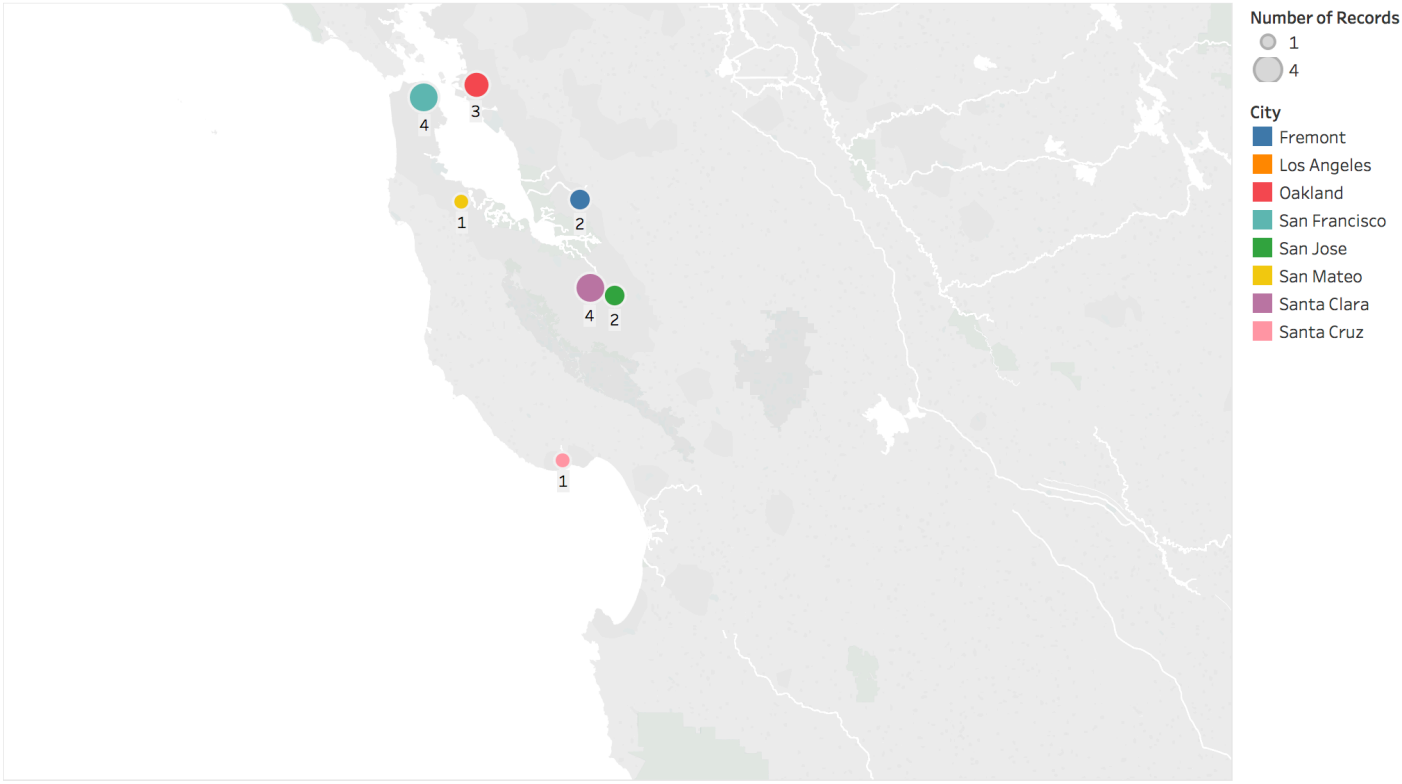
Sum of AGE for each Patient Name broken down by Gender. Color shows details about Patient Name. The marks are labeled by sum of AGE.

Month-wise visualization of revenue streams



Sum of Charge for each Charge Type broken down by MONTH. Color shows details about Charge Type. The marks are labeled by sum of Charge.

Geographic Distribution of Patients



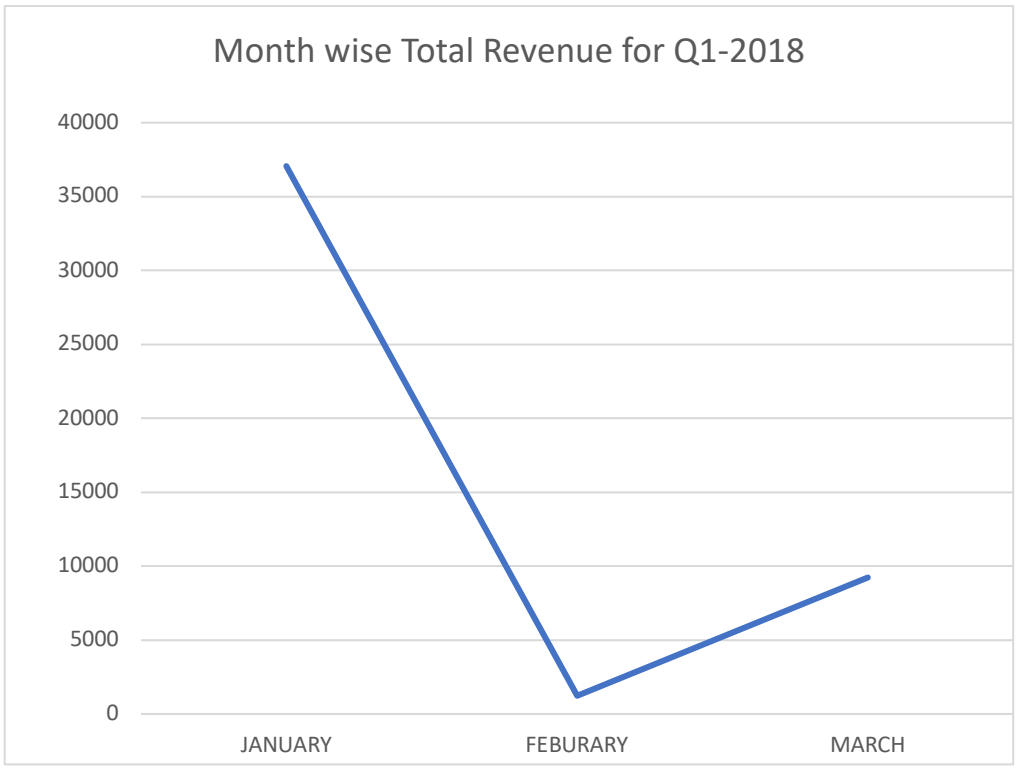
Map based on Longitude (generated) and Latitude (generated). Color shows details about City. Size shows sum of Number of Records. The marks are labeled by sum of Number of Records. Details are shown for State.

3. Hospital can analyze revenue based on different parameters such as period, departments etc.

```
59  #**VIEW MONTHLY REVENUE
60  SELECT monthname(DISCHARGE_TIMESTAMP) AS 'MONTH', SUM(TOTAL_BILL_AMOUNT) AS 'TOTAL REVENUE (in $)'
61  FROM VIEW_TOTAL_BILL
62  GROUP BY MONTH(DISCHARGE_TIMESTAMP);
63
64
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

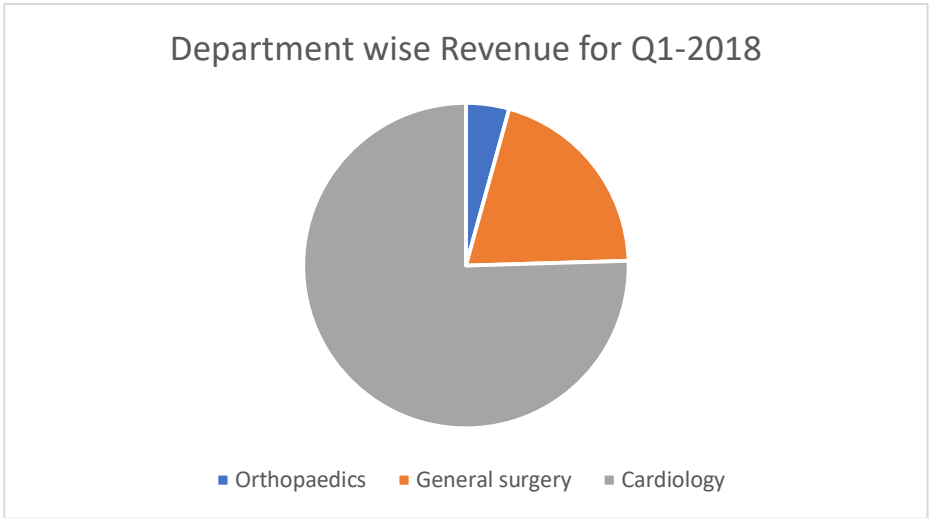
MONTH	TOTAL REVENUE (in $)
January	37060
February	1230
March	9240




```
66  #**VIEW REVENUE DEPATMENT WISE
67  SELECT D.DEPT_NAME AS 'DEPARTMENT NAME', SUM(TOTAL_BILL_AMOUNT) AS 'TOTAL REVENUE (in $)'
68  FROM VIEW_TOTAL_BILL V, STAFF_ASSIGNMENT SA, DEPARTMENT D
69  WHERE V.DOCTOR_ID=SA.STAFF_ID AND
70        SA.DEPT_ID= D.DEPT_ID
71  GROUP BY D.DEPT_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DEPARTMENT NAME	TOTAL REVENUE (in $)
Orthopaedics	2040
General surgery	9620
Cardiology	35870



4. Doctor can view average recovery time and also segment it across months and departments

```
#AVERAGERECOVERY TIME
SELECT AVG(TIMESTAMPDIFF(DAY,AP.ADMIT_TIMESTAMP, AP.DISCHARGE_TIMESTAMP)) AS 'Average Recovery Time'
FROM ADMITTED_PATIENT AP
WHERE AP.DISCHARGE_TIMESTAMP IS NOT NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Average Recovery Time
6.8333

```
#Segment average recovery time across months and departments
SELECT MONTHNAME(AP.ADMIT_TIMESTAMP) AS 'Month', D.DEPT_NAME,
AVG(TIMESTAMPDIFF(DAY,AP.ADMIT_TIMESTAMP,AP.DISCHARGE_TIMESTAMP)) AS 'Average Recovery Time'
FROM ADMITTED_PATIENT AP, STAFF_ASSIGNMENT SA, DEPARTMENT D
WHERE AP.DISCHARGE_DOCTOR_ID IS NOT NULL AND AP.DOCTOR_ID= SA.STAFF_ID AND SA.DEPT_ID=D.DEPT_ID
GROUP BY MONTHNAME(AP.ADMIT_TIMESTAMP),D.DEPT_NAME
ORDER BY MONTHNAME(AP.ADMIT_TIMESTAMP);
```

100% 40:58

Result Grid | Filter Rows: Search | Export:

Month	DEPT_NAME	Average Recovery Time
February	Cardiology	3.0000
February	General surgery	6.0000
January	Cardiology	9.0000
January	General surgery	6.0000
March	General surgery	12.0000
March	Orthopaedics	5.0000

Average Recovery Time per Department per Month

Sum of Average Recovery Time for each Dept Name broken down by Month. Color shows details about Dept Name. The marks are labeled by sum of Average Recovery Time. Details are shown for Month.

Project Summary

1. Experience with this exercise

Care Intelligence System has been designed to support various entities interacting with the hospital. Data can be inordinate, unorganized and fragmented which can act as a barrier to growth of an organization. CIS aims to overcome this barrier by managing data efficiently to catapult an organization to success. The primary goal of every medical facility is to keep patients safe while administering appropriate treatments and remedies for their conditions and illnesses. some studies now suggest that medical errors are prevalent enough to be the third leading cause of death[1] in the U.S. Medical personnel are human and errors can happen, especially in the realm of organization and CIS aims to plug this flaw in the medical system by ensuring that patient care is not compromised on account of mismanaged data.

The process of developing CIS application has been an enriching experience for us as a team, as we got an opportunity to practically apply concepts we learnt in class and understand the nuances and importance of good database design. Moreover, as a team we simulated a corporate environment as we delegated tasks and collaborated to implement a cohesive application. We effectively leveraged each individual team member's strengths to collectively enhance our own skill set as the exercise involved designing a database from scratch and progressing to run queries, implement triggers, indexes and finally visualize business metrics.

2. Hardest part of this project

The hardest part of the project was developing a normalized ER model for the following reasons:

i) Constantly ensuring coverage of all our use cases in our model
ii) While progressing through the project, we kept making incremental improvements to our schema which resulted in constant back and forth to capture these changes in our ER Model
iii) Translating the concepts, we had at inception of the project to the model was a challenge, as there is an inevitable gap in the way our thoughts are organized and in the way a DBMS organizes data

3. Problems we ran against in this project

i) Populating data was a challenge as we needed a rich database in order to effectively capture functionalities of our users and to generate interesting business metrics
ii) Capturing use cases in the schema as we had to think like each user and simulate their behaviors and requirements to develop a rich model

[1] https://www.npr.org/sections/health-shots/2016/05/03/476636183/death-certificates-undercount-toll-of-medical-errors

iii) Maintaining various versions of the schema and ensuring the whole team had access to the latest updated version
iv) Coordinating team meetings as all class projects were due around the same time
v) Delegation of tasks as each process is tightly integrated to various other processes and stages of the project

4. Solution to these problems

i) We rationalized the learnings from the process of populating and repopulating data as it exposed us to complex nuances of the DBMS, which made the process less painful
ii) We got inputs from friends in the medical care industry to get insights on the challenges they face and the systems they implement to overcome these challenges and also to understand the KPIs of the industry
iii) We adopted a naming convention that enforced version control and made it standard practice to communicate to the team each time we generate a new version
iv) We coordinated with team members to religiously schedule meetings and set agendas to track progress and instead of remote task delegation, executed every aspect of the project through in-person meetings

5. If we were to do this project again, the methodology we would follow

i) Our process of selecting the project domain was first selecting an industry that deals with a lot of data, and then populating data to align with our conceptual model. Now that we have a strong foundation and understand the nuances of populating data, we would approach it from the other way round, and first identify available data and model our project domain around that. This would enable us to simulate an experience closer to real-life where systems are designed to handle existing unstructured data
ii) We would like to explore functionalities and capabilities of NoSQL

6. Suggestions to refine this project for the next class:

Decrease focus on data population as we personally enjoyed the process of generating interesting insights through querying and visualizing data