**HOSPITAL MANAGEMENT APPLICATION**

**USING SQL & MongoDB**

By

Jayalakshmi Vaidyanathan

**HOSPITAL MANAGEMENT**

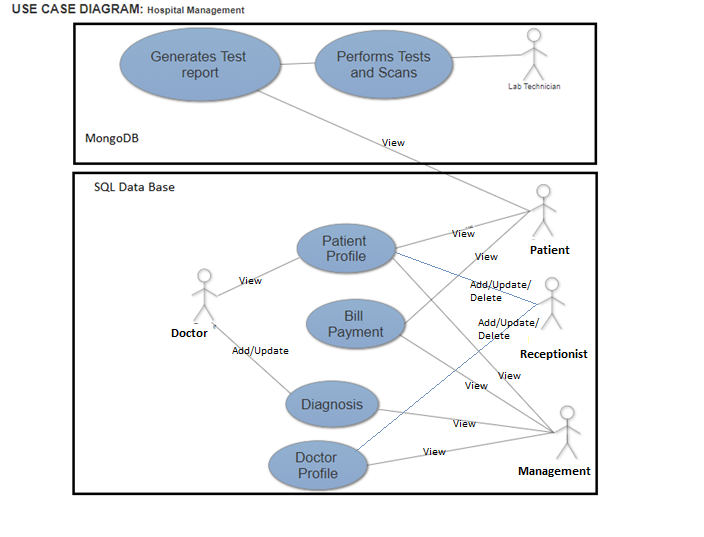
1. **INTRODUCTION:**

Data Management is one of the major challenges in Healthcare. Hospitals struggle to store and manage the patients records and images from various tests and surgeries carried out. Medical Reports are the most intensive and diverse in Hospital Management Information System. This application works on a hybrid model that combines both SQL Data base and MongoDB to solve the data storage and management challenges

* 1. **Scope:**

The hospital application provides solution for the following:

1. Patient profile management using SQL
2. Diagnosis management Using SQL
3. Billing Management Using SQL
4. Revenue and other Business Management Reports Using SQL
5. Repository for Lab reports,images and scans, using large files handling features of MongDB
6. **USERS:**



1. **Doctor:**

* Updates Diagnoses details about the patients
* Reads the patients history
* Reads patient’s Lab reports

1. **Patient:**

* Views his profile, Diagnosis and test report
* Makes bill payments

1. **Receptionist:**

* Adds, Updates, deletes Patient’s profile
* Adds, Updates, deletes Doctor’s profile

1. **Management:**

* View profiles of Doctors, patient and payments for business management purpose

1. **Lab Technician:**

* Conducts all lab tests and records the test results and scanned images

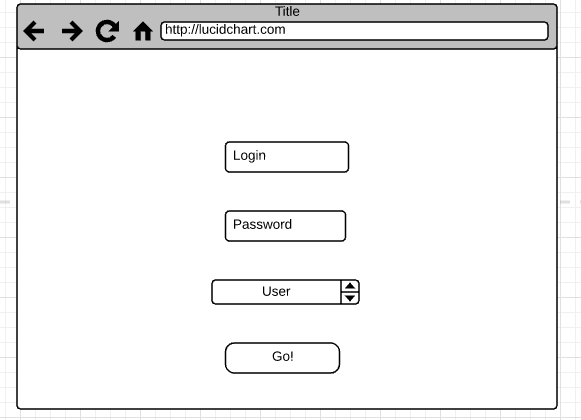
1. **USE CASE:**
2. **Patient Details Management**

* All the registered patients are recorded in tables.
* The Patients are provided with their profile login
* Doctors view the patient’s details
* The hospital Management can also view the information for business purpose

1. **Diagnosis Details Management**

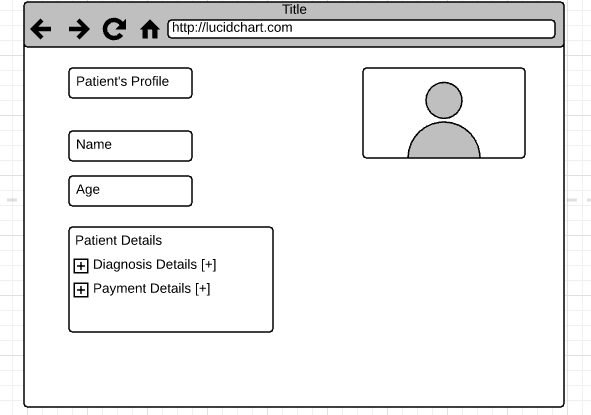
* The doctor updates every diagnosis of patients visiting for consultation
* The patients can view the updated data
* The hospital Management can also view the information for business purpose

1. **Billing Management**
   * 1. Payments made by patients are recorded
     2. The hospital Management can also view the information for business purpose
2. **Lab Data Management**
   * 1. Lab Technician updated the scan and other test reports
     2. Doctors can view their patient’s reports
     3. Patients can view their reports online
3. **USER INTERFACE:**
   * + 1. **Login**



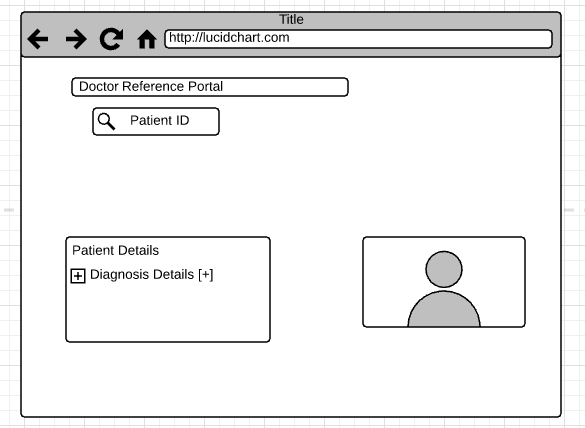
**Login**: Based on whether the user is a patient, doctor or admin, the login page will lead to corresponding user interface. The User can be chosen from the dropdown box. Once the credentials are validated, it guides to the corresponding user portals mentioned below.

* + - 1. **Patient’s Profile**



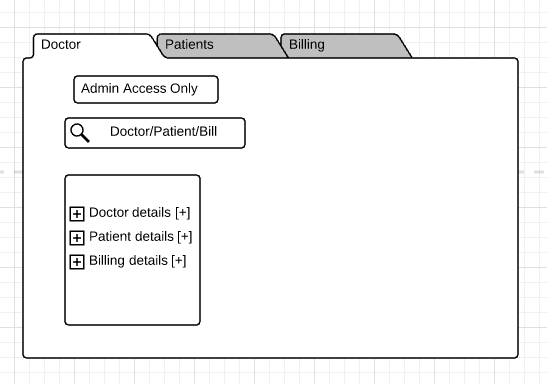
**Patient’s Profile:** The patients can access the above page to view their hospital details

* + - 1. **Diagnosis Details**



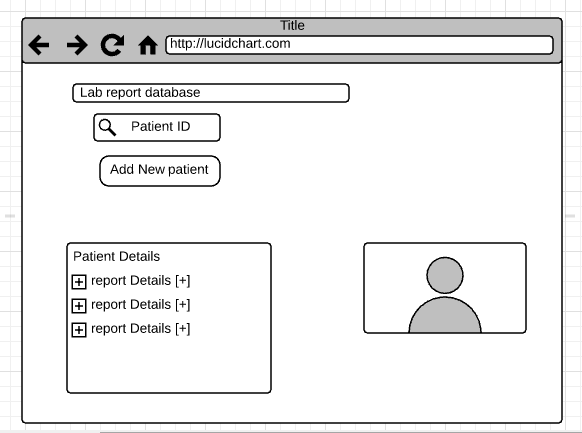
**Diagnosis Details:** Doctors can view the patient’s history from the above portal

* + - 1. **Management**



The management can view all the details about the doctors, patients and financial information from then above portal.

1. **Lab Reports**



**Lab Report:** The Lab technicians can update the test results, images and scan in then above portal

1. **SQL Vs NoSQL:**
2. **SQL:**

The below use cases are handled in SQL. Due to ACID property of SQL, they are used for transactional processes , where we require structured data handling systems and consistent data retrieval.

1. **Patient Record Detail**

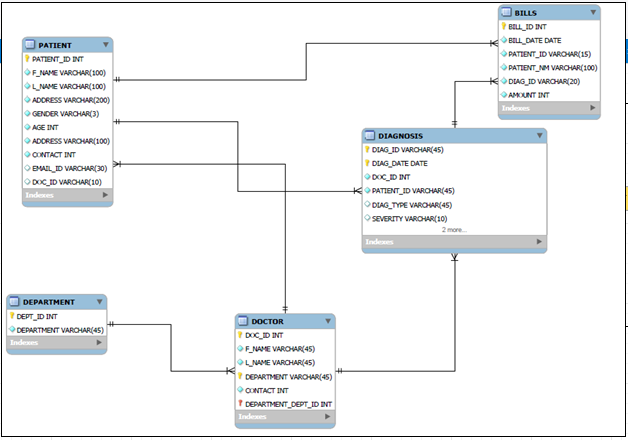
All the patient’s profile details have to be stored and retrieved whenever the patient or doctor wishes to view them. The data retrieval has to be consistent and precise. Also, the details of all patients follow structured pattern. In such a case SQL is the appropriate solution

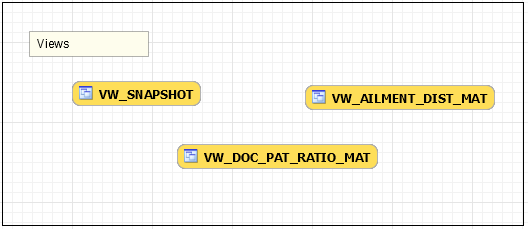
1. **Diagnosis Details Management**

When a patient visits a doctor, the doctor has to update patient’s diagnosis details. This has to be available to patients when he /she views his/her profile. The data has to be appropriate and instantly updated. SQL is better option as they provide structured and consistent data retrieval.

1. **Billing Management**

The payment of bills by the patients has to be secured and involves the same kind of transactions throughout. In such situation where it involves consistent, secured and homogenous transaction SQL is the best choice.

****

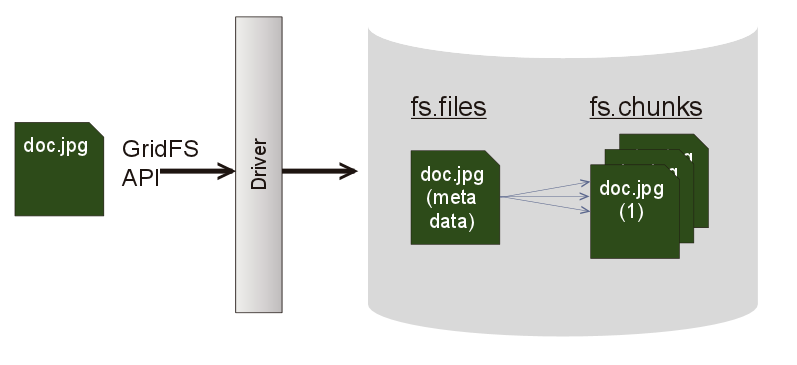


**SQL – UML Diagram**

1. **NoSQL:**

**Lab Data Management**

There are variety of tests that are conducted to patients which ranges from a simple blood tests to MRI scans. Each patient produces voluminous and varied data for every lab test. In such case, handling lab data in a structured environment will be very difficult. More flexible environment like NoSQL is needed. The BASE property of NoSQL helps in storing and managing different test results in an easy fashion and also data retrieval is faster in case of large data. The file system handling features like mongofiles and GridFS provides solutions for handling very large files greater than 16MB as well. Handling images and videos are also tedious process in SQL

****

1. **QUERY ENUMERATION:**
2. **Patient Details Management**

* **Patient Login:** Patient ID and password must match to access his profile
* **Receptionist:** Needs to add, update and delete the patient profile details if needed
* **Doctor** has to view the patient’s details
* **Management** can View patient’s details

1. **Diagnosis Details Management**

* **Doctor** needs to add or update for each consultation
* **Patient** can view the updated data
* **Management** can view for Business analysis reports

1. **Billing Management**

* **Patient** can view the payments
* **Receptionist** can add or update the billing details
* **Management**  can view for Business analysis reports

1. **Lab Data Management**

* **Lab technician can add or** update the test results and reports
* **Doctors** can view the reports
* **Patients** can view the reports

1. **BUSINESS METRICS:**
2. Overall Snapshot of the hospital management
3. Revenue Metrics
4. Doctor to Patient Ratio
5. Gender wise Ailment Frequency
6. Department wise revenue report
7. **Design Database Schema & MongoDB structure**
8. **SQL**
9. **TABLES**

* DOCTOR

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **COLUMN** | **DATATYPE** | **NULL/NOT NULL** | **AUTO INCREMENT** | **PRIMARY KEY?** | **FOREIGN KEY?** | **INDEX** |
| DOC\_ID | INT | NOT NULL | Y | Y | Y | Y |
| F\_NAME | CHAR(30) | NOT NULL |  |  |  |  |
| L\_NAME | CHAR(30) | NOT NULL |  |  |  |  |
| DEPARTMENT | VARCHAR(30) | NOT NULL |  |  |  |  |
| CONTACT | VARCHAR(20) | NOT NULL |  |  |  |  |
| EXPERIENCE | INTEGER(9) | NOT NULL |  |  |  |  |

**PRIVILIGES**

|  |  |
| --- | --- |
| **USER** | **PRIVILIGES** |
| Receptionist | INSERT,UPDATE |
| Management | SELECT |
| Patient | SELECT |

* PATIENT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **COLUMN** | **DATATYPE** | **NULL/NOT NULL** | **AUTO INCREMENT** | **PRIMARY KEY?** | **FOREIGN KEY** | **INDEX** |
| PATIENT\_ID | INT | NOT NULL | Y | Y |  | Y |
| F\_NAME | CHAR(30) | NOT NULL |  |  |  |  |
| L\_NAME | CHAR(30) | NOT NULL |  |  |  |  |
| GENDER | CHAR(12) | NOT NULL |  |  |  |  |
| ADDRESS | VARCHAR(256) | NOT NULL |  |  |  |  |
| CONTACT | VARCHAR(20) | NOT NULL |  |  |  |  |
| EMAIL\_ID | VARCHAR (100) | NULL |  |  |  |  |
| DOC\_ID | VARCHAR(10) | NOT NULL |  |  | Y |  |

**PRIVILIGES**

|  |  |
| --- | --- |
| **USER** | **PRIVILIGES** |
| Receptionist | INSERT,UPDATE |

* DIAGNOSIS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **COLUMN** | **DATATYPE** | **NULL/NOT NULL** | **AUTO INCREMENT** | **PRIMARY KEY?** | **FOREIGN KEY** |
| DIAG\_ID | INT | NOT NULL | Y | Y |  |
| DIAG\_DATE | DATE | NOT NULL |  |  |  |
| DOC\_ID | VARCHAR(20) | NOT NULL |  |  | Y |
| PATIENT\_ID | INT | NOT NULL |  |  | Y  ON DELETE CASCADE |
| DIAG\_TYPE | VARCHAR(45) | NULL |  |  |  |
| SEVERITY | VARCHAR(10) | NULL |  |  |  |
| TREATMENT | VARCHAR(45) | NULL |  |  |  |
| PRESC\_DRUGS | VARCHAR(45) | NULL |  |  |  |

PRIVILIGES

|  |  |
| --- | --- |
| **USER** | **PRIVILIGES** |
| DOCTOR | INSERT,UPDATE |
| PATIENT | SELECT |

* DEPARTMENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **COLUMN** | **DATATYPE** | **NULL/NOT NULL** | **AUTO INCREMENT** | **PRIMARY KEY?** |
| DEPT\_ID | INT | NOT NULL | N | Y |
| DEPT\_NAME | VARCHAR(45) | NOT NULL | N |  |

PRIVILIGES

|  |  |
| --- | --- |
| **USER** | **PRIVILIGES** |
| Managenment | SELECT |

* BILL

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **COLUMN** | **DATATYPE** | **NULL/NOT NULL** | **AUTO INCREMENT** | **PRIMARY KEY?** | **FOREIGN KEY** | **INDEX** |
| BILL\_ID | INT | NOT NULL | Y | Y |  |  |
| BILL\_DATE | CHAR(30) | NOT NULL |  |  |  | Y |
| PATIENT\_ID | CHAR(30) | NOT NULL |  |  | Y  ON DELETE CASCADE |  |
| PATIENT\_NM | CHAR(12) | NOT NULL |  |  |  |  |
| DIAG\_ID | VARCHAR(256) | NOT NULL |  |  | Y |  |
| AMOUNT | DECIMAL | NOT NULL |  |  |  |  |

**PRIVILIGES**

|  |  |
| --- | --- |
| **USER** | **PRIVILIGES** |
| PATIENTS | SELECT |
| Receptionist | INSERT,UPDATE |

1. **VIEWS**

* VW\_SNAPSHOT

CREATE VIEW VW\_SNAPSHOT AS

SELECT

D.DOC\_ID,

CONCAT(D.F\_NAME,' ',D.L\_NAME) AS DOCTOR,

P.PATIENT\_ID,

CONCAT(P.F\_NAME,' ',P.L\_NAME) AS PATIENT,

DG.DIAG\_DATE ,

DG.DIAG\_TYPE ,

DG.SEVERITY ,

DG.TREATMENT ,

DG.PRESC\_DRUGS ,

B.AMOUNT

FROM DOCTOR D

JOIN

PATIENT P

ON D.DOC\_ID = P.DOC\_ID

JOIN DIAGNOSIS DG

ON P.PATIENT\_ID = DG.PATIENT\_ID

AND D.DOC\_ID = DG.DOC\_ID

JOIN BILL B

ON P.PATIENT\_ID = B.PATIENT\_ID AND

DG.DIAG\_ID = B.DIAG\_ID

ORDER BY DG.DIAG\_DATE;

* VW\_DOC\_PAT\_RATIO\_MAT

CREATE VIEW VW\_DOC\_PAT\_RATIO\_MAT AS

SELECT

CONCAT(D.F\_NAME,' ',D.L\_NAME) DOCTOR\_NAME,

COUNT(P.PATIENT\_ID)

FROM DOCTOR D

JOIN

PATIENT P

ON D.DOC\_ID = P.DOC\_ID

JOIN DIAGNOSIS DG

ON P.PATIENT\_ID = DG.PATIENT\_ID

AND D.DOC\_ID = DG.DOC\_ID

GROUP BY D.DOC\_ID

ORDER BY DG.DIAG\_DATE;

* VW\_AILMENT\_DIST\_MAT

CREATE VIEW VW\_AILMENT\_DIST\_MAT AS

SELECT

P.GENDER ,

DG.DIAG\_TYPE,

COUNT(P.PATIENT\_ID) COUNT

FROM

PATIENT P

JOIN DIAGNOSIS DG

ON P.PATIENT\_ID = DG.PATIENT\_ID

GROUP BY P.GENDER,DG.DIAG\_TYPE

ORDER BY P.GENDER,DG.DIAG\_TYPE;

1. **STORED PROCEDURE**

* PROC\_REVENUE\_METRIX

CREATE PROCEDURE PROC\_REVENUE\_METRIX

AS

BEGIN

SELECT A.YEAR,

A.MONTH,

SUM(A.AMOUNT) AS REVENUE\_MAT

FROM (SELECT

EXTRACT(MONTH FROM BILL\_DATE) AS MONTH,

EXTRACT(YEAR FROM BILL\_DATE) AS YEAR,

AMOUNT

FROM BILL ) A

GROUP BY A.YEAR,A.MONTH

ORDER BY A.YEAR,A.MONTH

END;

1. **NoSQL**
2. **DATABASE**
   1. Gridfs
3. **COLLECTIONS:**
   1. Patient
   2. Db.ctFiles.files
   3. Db.ctFiles.chunks
      * 1. **Patients**

{"Patient\_id":11221,

"Patient\_name":"SIMON TAYLOR",

"gender":"MALE",

"Age":"50",

"Adderess":"LONDON",

"Contact":"877-609-2983",

"Email\_id":"taylor @email.com","Doc\_id":"1101" }, ……..

**B.fs.chunks**

{

"files\_id": ObjectId("574a65c19f54bfea8a2fa46b"),

"n": NumberInt(0),

"data": "Mongo Binary Data"

} …..

C.fs.files

{

"filename": "sample.jpeg",

"chunkSize": NumberInt(261120),

"uploadDate": ISODate("2017-11-28T11:32:33.557Z"),

"md5": "7b772939325e146578b07f72c62bba4f",

"length": NumberInt(645)

}

1. **DATABASE POPULATION**
2. **SQL:**

**DDL and Scripts:**

CREATE TABLE DOCTOR (

DOC\_ID INT NOT NULL UNIQUE ,

F\_NAME CHAR(30) NOT NULL,

L\_NAME CHAR(30) NOT NULL,

DEPARTMENT VARCHAR(30) NOT NULL,

CONTACT VARCHAR(20) NOT NULL,

EXPERIENCE INTEGER (9) NOT NULL,

CONSTRAINT doc\_pkey PRIMARY KEY (DOC\_ID)

);

CREATE INDEX idx\_doc\_doc\_id ON DOCTOR (DOC\_ID);

GRANT INSERT,UPDATE ON DOCTOR TO receptionist;

GRANT SELECT ON DOCTOR TO management;

CREATE TABLE PATIENT (

PATIENT\_ID INT NOT NULL UNIQUE ,

F\_NAME CHAR(30) NOT NULL,

L\_NAME CHAR(30) NOT NULL,

GENDER CHAR(12) NOT NULL,

AGE INT(3) NOT NULL,

ADDRESS VARCHAR(256) NOT NULL,

CONTACT VARCHAR(20) NOT NULL,

EMAIL\_ID VARCHAR (100),

DOC\_ID INT NOT NULL,

CONSTRAINT doc\_pkey PRIMARY KEY (PATIENT\_ID),

CONSTRAINT patient\_doctor\_fk FOREIGN KEY (DOC\_ID) REFERENCES DOCTOR (DOC\_ID));

GRANT INSERT,UPDATE ON PATIENT TO RECEPTIONIST;

CREATE INDEX idx\_pat\_pat\_id

ON PATIENT (PATIENT\_ID);

CREATE TABLE DIAGNOSIS (

DIAG\_ID INT NOT NULL UNIQUE,

DIAG\_DATE DATE NOT NULL,

DOC\_ID INT NOT NULL,

PATIENT\_ID INT NOT NULL,

DIAG\_TYPE VARCHAR(45) NULL,

SEVERITY VARCHAR(10) NULL,

TREATMENT VARCHAR(45) NULL,

PRESC\_DRUGS VARCHAR(45) NULL,

CONSTRAINT diag\_pkey PRIMARY KEY (DIAG\_ID),

CONSTRAINT diag\_doc\_fk FOREIGN KEY (DOC\_ID) REFERENCES DOCTOR (DOC\_ID),

CONSTRAINT diag\_pat\_fk FOREIGN KEY (PATIENT\_ID) REFERENCES PATIENT (PATIENT\_ID)

ON DELETE CASCADE

);

GRANT INSERT,UPDATE ON DIAGNOSIS TO DOCTOR\_LIST;

GRANT SELECT ON DIAGNOSIS TO PATIENT\_LIST;

CREATE TABLE DEPARTMENT (

DEPT\_ID INT NOT NULL AUTO\_INCREMENT,

DEPT\_NAME VARCHAR(45) NOT NULL,

CONSTRAINT dept\_pkey PRIMARY KEY (DEPT\_ID));

GRANT SELECT ON DEPARTMENT TO ADMIN2;

CREATE TABLE BILL(

BILL\_ID INT NOT NULL UNIQUE ,

BILL\_DATE DATE NOT NULL,

PATIENT\_ID INT NOT NULL,

PATIENT\_NM VARCHAR(200) NOT NULL,

DIAG\_ID INT NOT NULL ,

AMOUNT INT NOT NULL,

CONSTRAINT bill\_pkey PRIMARY KEY (BILL\_ID),

CONSTRAINT bill\_pt\_fk FOREIGN KEY (DIAG\_ID) REFERENCES DIAGNOSIS (DIAG\_ID),

CONSTRAINT bill\_diag\_fk FOREIGN KEY (PATIENT\_ID) REFERENCES PATIENT (PATIENT\_ID) ON DELETE CASCADE);

CREATE INDEX idx\_bill\_bill\_dt

ON BILL (BILL\_DATE);

GRANT INSERT,UPDATE ON BILL TO Receptionist;

GRANT SELECT ON BILL TO PATIENT\_LIST;

CREATE VIEW VW\_SNAPSHOT AS

SELECT

D.DOC\_ID,

CONCAT(D.F\_NAME,' ',D.L\_NAME) AS DOCTOR,

P.PATIENT\_ID,

CONCAT(P.F\_NAME,' ',P.L\_NAME) AS PATIENT,

DG.DIAG\_DATE ,

DG.DIAG\_TYPE ,

DG.SEVERITY ,

DG.TREATMENT ,

DG.PRESC\_DRUGS ,

B.AMOUNT

FROM DOCTOR D

JOIN

PATIENT P

ON D.DOC\_ID = P.DOC\_ID

JOIN DIAGNOSIS DG

ON P.PATIENT\_ID = DG.PATIENT\_ID

AND D.DOC\_ID = DG.DOC\_ID

JOIN BILL B

ON P.PATIENT\_ID = B.PATIENT\_ID AND

DG.DIAG\_ID = B.DIAG\_ID

ORDER BY DG.DIAG\_DATE;

CREATE VIEW VW\_DOC\_PAT\_RATIO\_MAT AS

SELECT

CONCAT(D.F\_NAME,' ',D.L\_NAME) DOCTOR\_NAME,

COUNT(P.PATIENT\_ID)

FROM DOCTOR D

JOIN

PATIENT P

ON D.DOC\_ID = P.DOC\_ID

JOIN DIAGNOSIS DG

ON P.PATIENT\_ID = DG.PATIENT\_ID

AND D.DOC\_ID = DG.DOC\_ID

GROUP BY D.DOC\_ID

ORDER BY DG.DIAG\_DATE;

CREATE VIEW VW\_AILMENT\_DIST\_MAT AS

SELECT

P.GENDER ,

DG.DIAG\_TYPE,

COUNT(P.PATIENT\_ID) COUNT

FROM

PATIENT P

JOIN DIAGNOSIS DG

ON P.PATIENT\_ID = DG.PATIENT\_ID

GROUP BY P.GENDER,DG.DIAG\_TYPE

ORDER BY P.GENDER,DG.DIAG\_TYPE;

CREATE PROCEDURE PROC\_REVENUE\_METRIX

AS

BEGIN

SELECT A.YEAR,

A.MONTH,

SUM(A.AMOUNT) AS REVENUE\_MAT

FROM (SELECT

EXTRACT(MONTH FROM BILL\_DATE) AS MONTH,

EXTRACT(YEAR FROM BILL\_DATE) AS YEAR,

AMOUNT

FROM BILL ) A

GROUP BY A.YEAR,A.MONTH

ORDER BY A.YEAR,A.MONTH

END;

**Data Population:**

Refer to DUMP.sql to find the entire dataset. Please find the samples below:

INSERT INTO DOCTOR

(DOC\_ID, F\_NAME, L\_NAME, DEPARTMENT, CONTACT, EXPERIENCE)

values ('1101', 'SAM', 'HENRY', 'CARDIOLOGY', '877-609-2233', 10),

('1100', 'MARK', 'TWAIN', 'CARDIOLOGY', '877-638-2234', 30),

('1102', 'CHARLES', 'DICKENS', 'OPTHAMOLOGY', '877-609-2245', 35),

('1104', 'ROWLING', 'JK', 'ONCOLOGY', '877-611-2223', 10),

('1106', 'HENRY', 'OO', 'CARDIOLOGY', '897-609-2443', 11),

('1107', 'WILLIAM', 'SHAKESPEARE', 'ONCOLOGY', '877-678-2203', 12),

('1199', 'SIDNEY', 'SHELDON', 'OPTHAMOLOGY', '857-719-2234', 25),

('1103', 'JEFFREY', 'ARCHER', 'CARDIOLOGY', '867-929-2783', 12),

('1112', 'WILLIAM', 'WORDSWORTH', 'ONCOLOGY', '887-611-2230', 21),

('1111', 'ROBERT', 'FROST', 'OPTHAMOLOGY', '887-699-2236', 11),

('1114', 'JOHN', 'MILTON', 'CARDIOLOGY', '877-689-2238', 19),

('1115', 'SHELLEY', 'PB', 'OPTHAMOLOGY', '867-679-2273', 10)

;

INSERT INTO PATIENT

(PATIENT\_ID,F\_NAME,L\_NAME,GENDER, AGE,ADDRESS,CONTACT,EMAIL\_ID,DOC\_ID)

VALUES

('11221','SIMON','TAYLOR','MALE','50','LONDON','877-609-2983','taylor @email.com','1101'),

('11222','ISAAC','NEWTON','MALE','30','CALIFORNIA','877-699-2984','newton @email.com','1106'),

('11223','ALBERT','EINSTIEN','MALE','60','TEXAS','887-609-2985','einstien@email.com','1101'),

('11224','MICHAEL','FARADEY','MALE','80','LONDON','877-689-2986','faradey@email.com','1106'),

('11225','MARIE','CURIE','FEMALE','30','CALIFORNIA','877-677-2987','curie@email.com','1112'),

('11226','ANNA','TAYLOR','FEMALE','35','TEXAS','977-659-2988','taylor@email.com','1101'),

('11227','EMILEY','MIKE','FEMALE','30','LONDON','847-609-2989','mike@email.com','1112'),

('11228','THOMAS','EDISON','MALE','40','CALIFORNIA','877-629-2990','edison@email.com','1101'),

('11229','CHARLES','DARWIN','MALE','44','TEXAS','807-640-2991','darwin@email.com','1111'),

('11230','STEPHEN','HAWKING','MALE','20','CALIFORNIA','877-909-2992','hawking@email.com','1101');

INSERT INTO DIAGNOSIS

(DIAG\_ID ,DIAG\_DATE, DOC\_ID, PATIENT\_ID, DIAG\_TYPE, SEVERITY, TREATMENT, PRESC\_DRUGS)

values ('2211', '2017/01/01', '1101', '11221','AGIOPLASTY', '1', 'CONSULTATION', 'ZZT YYYY'),

('2212', '2017/04/06', '1106', '11222','CATRACT', '3', 'SURGERY', 'XXXX YYYY'),

('2213', '2017/03/12', '1101', '11223','AGIOPLASTY', '1', 'CONSULTATION', 'ZZZ YYYY'),

('2214', '2016/06/04', '1106', '11224','EYE INFECTION', '1', 'CONSULTATION', 'XXXX QQQ , TUYHG SVCHN'),

('2215', '2017/03/09', '1112', '11225','STOMACH CANCER', '2', 'RADIOTHERAPHY', 'XXXX TTTT'),

('2216', '2017/12/11', '1101', '11226','AGIOPLASTY', '1', 'CONSULTATION', 'XXXX YYYY'),

('2217', '2016/03/10', '1112', '11227','BLOOD CANCER', '1', 'CONSULTATION', 'DDD YYYY'),

('2218', '2017/05/07', '1101', '11228','OPEN HEART', '4', 'SURGERY', 'XXXX YYYY'),

('2219', '2017/03/10', '1111', '11229','CATRACT', '3', 'SURGERY', 'XXXX EEEE'),

('2220', '2016/11/05', '1101', '11230','CHEST CONGESION', '2', 'CONSULTATION', 'XXXX YYYY');

INSERT INTO DEPARTMENT (

DEPT\_ID ,DEPT\_NAME)

values ('1', 'ONCOLOGY'),

('2', 'OPTHAMOLOGY'),

('3', 'CARDIOLOGY')

;

INSERT INTO BILL

(BILL\_ID , BILL\_DATE,PATIENT\_ID, PATIENT\_NM ,DIAG\_ID, AMOUNT)

values ('3311', '03/21/2017', '11221', 'SIMON TAYLOR','2211', '1000'),

('3312', '2017/04/06', '11222','ISAAC NEWTON','2212', '200'),

('3313', '2017/03/12', '11223','ALBERT EINSTIEN', '2213', '600'),

('3314', '2016/06/04', '11224', 'MICHAEL FARADEY', '2214', '800'),

('3315', '2017/03/09', '11225', 'MARIE CURIE', '2215', '550'),

('3316', '201712/11', '11226', 'ANNA TAYLOR', '2216', '120'),

('3317', '2016/03/10', '11227', 'EMILEY MIKE', '2217', '250'),

('3318', '2017/05/07', '11228', 'THOMAS EDISON', '2218', '100'),

('3319', '2017/03/10', '11229', 'CHARLES DARWIN', '2219', '390'),

('3320', '2016/11/05', '11230', 'STEPHEN HAWKING', '2220', '340');

1. **NoSQL:**
2. **Creating Database**

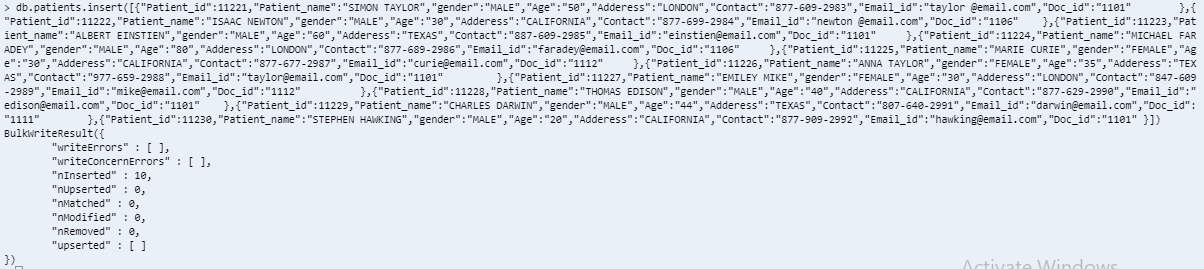
Use gridfs

1. **Creating Collection :**

* **Patient:**

db.createCollection("patient")

Inserting data into collection Patient as shown in below screenshot:



* **ctFiles.files**

db.createCollection("ctFiles.files ")

* **ctFiles.chunks**

db.createCollection("ctFiles.chunks ")

1. **File upload using Nodejs-gridfs-angular2 – Setup**

* git clone https://github.com/rahil471/fileupload-nodejs-gridfs-angular2.git
* cd fileupload-nodejs-gridfs-angular2/
* npm install
* In new terminal, started node server

node app.js

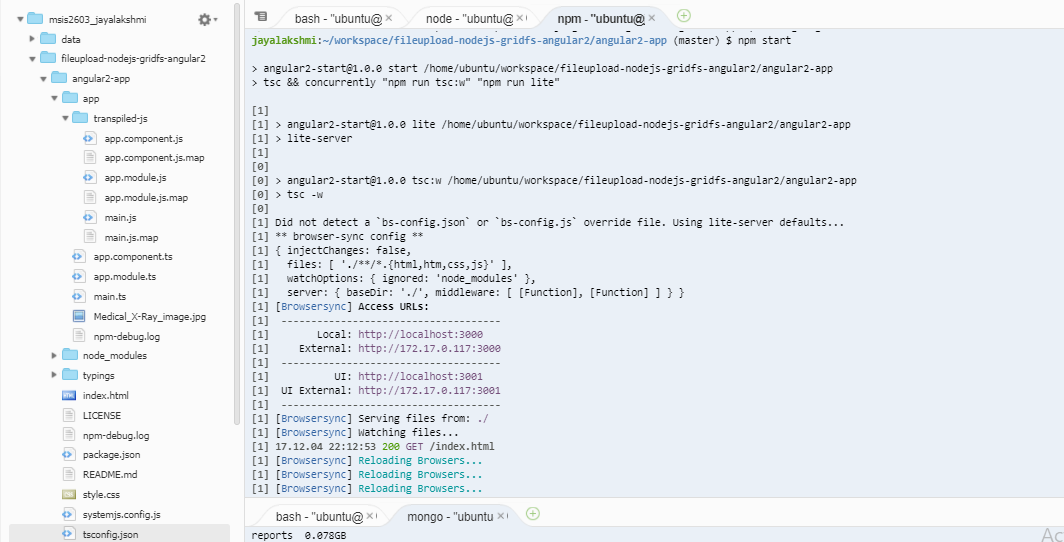
the node server started on port 3002



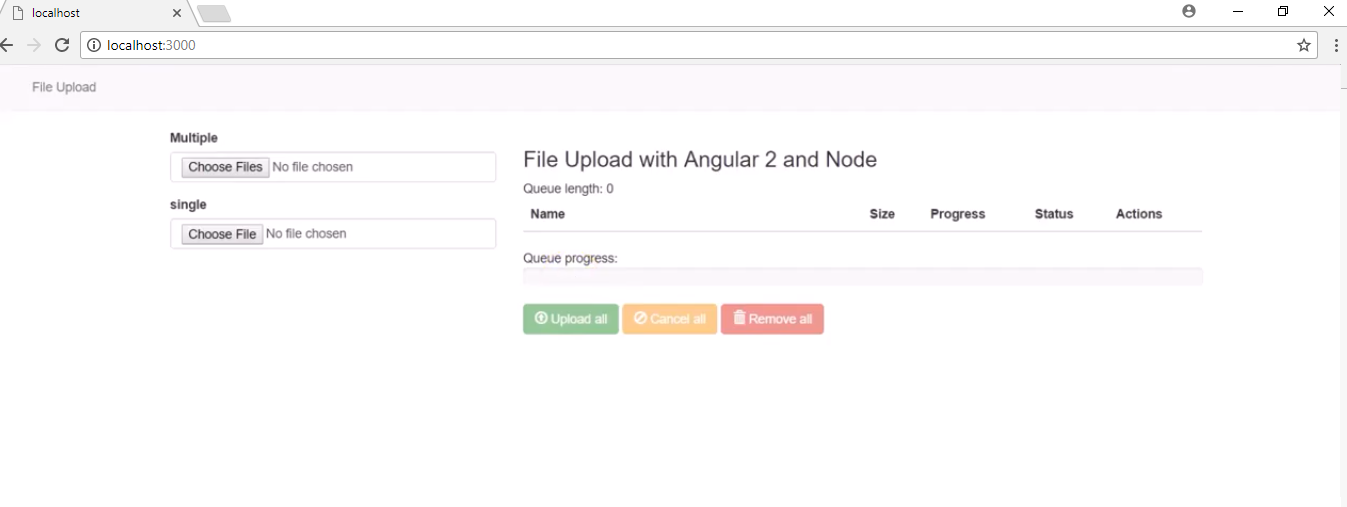
* Connection to the application:

cd angular2-app

npm start



* angular2 app will be running on port 3000 and will open in browser as shown below. The images to be stored is uploaded



1. **Queries to upload and retrieve files:**

// Query to upload images

mongofiles -d database put Medical\_X-Ray\_image.jpg

//Query to retrieve images

mongofiles -d database get Medical\_X-Ray\_image.jpg

//Query to delete images

mongofiles -d database delete Medical\_X-Ray\_image.jpg

//Query to upload videos

mongofiles.exe -d gridfs put sample.mp3

//The below query uploads the bson data

./mongofiles –port 28017 -d gridfs put people.bson

1. **QUERIES:**
   1. **USER: RECEPTIONIST**
2. **To add new patient’s profile**

INSERT INTO PATIENT

(PATIENT\_ID,F\_NAME,L\_NAME,GENDER, AGE,ADDRESS,CONTACT,EMAIL\_ID,DOC\_ID)

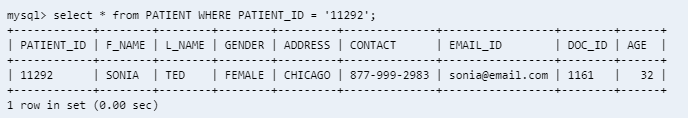
VALUES

('11292','SONIA','TED','FEMALE','32','CHICAGO','877-999-2983','sonia@email.com','1161');

**Before Insert:**



**After Insert:**

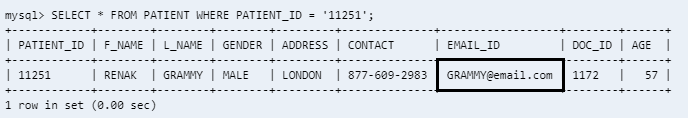


1. **When the patient’s email Id needs to be updated by the receptionist**

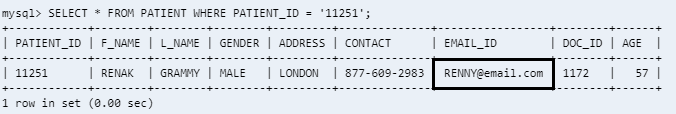
UPDATE PATIENT SET EMAIL\_ID = 'RENNY@email.com'

WHERE PATIENT\_ID = '11251';

**Before Update:**



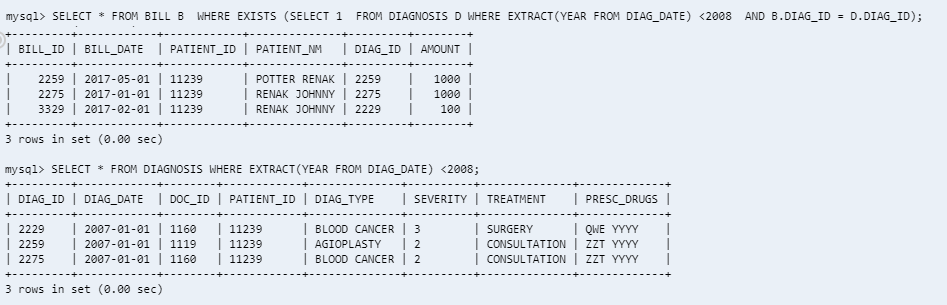
**After Update:**



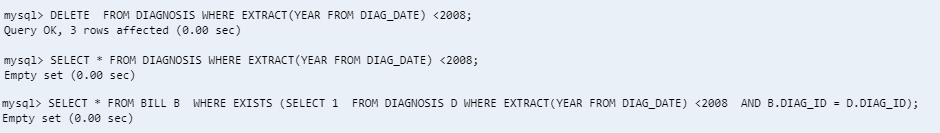
1. **To delete very old consultation details of patient**

DELETE FROM DIAGNOSIS WHERE EXTRACT(YEAR FROM DIAG\_DATE) < 2018 ;

**Before deletion:**



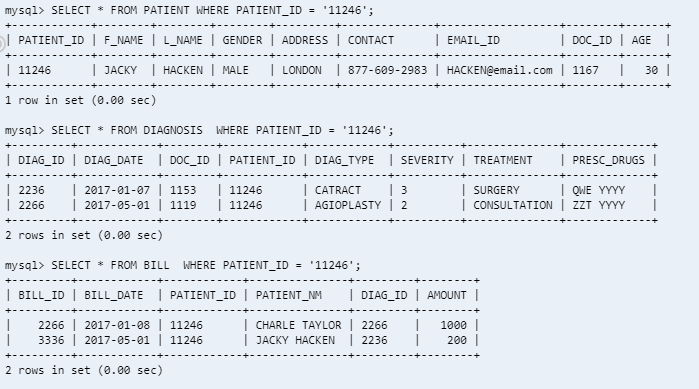
**After Deletion:**



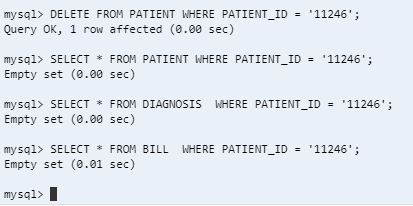
1. **To delete a particular patient’s details (may be upon death)**

DELETE FROM PATIENT WHERE PATIENT\_ID = '11246';

**Before Delete:**



**After Delete:**



1. **USER: DOCTOR**
2. **When Doctor needs to add new diagnosis data of his existing patient**

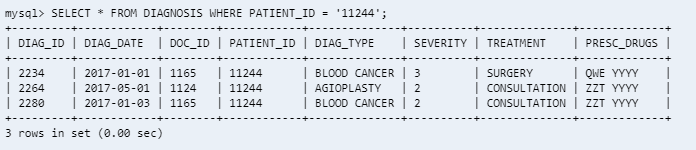
INSERT INTO DIAGNOSIS

(DIAG\_ID ,DIAG\_DATE, DOC\_ID, PATIENT\_ID, DIAG\_TYPE, SEVERITY, TREATMENT, PRESC\_DRUGS)

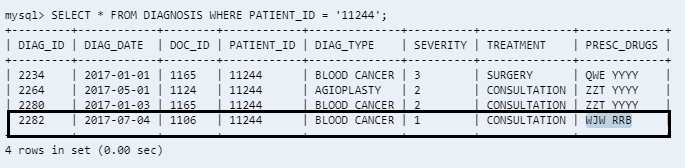
VALUES

('2282' , '2017-07-04','1106' ,'11244','BLOOD CANCER' ,'1' ,'CONSULTATION','WJW RRB' );

**Before Insert:**



**After Insert:**



1. **When doctor access the patient’s history data**

SELECT

P.PATIENT\_ID,

CONCAT(P.F\_NAME,' ',P.L\_NAME) AS PATIENT,

DG.DIAG\_DATE ,

DG.DIAG\_TYPE ,

DG.SEVERITY ,

DG.TREATMENT ,

DG.PRESC\_DRUGS

FROM PATIENT P

JOIN DIAGNOSIS DG

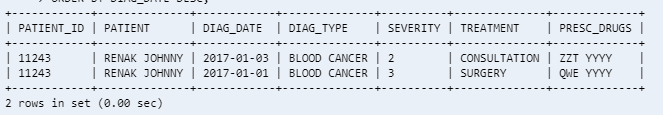
ON P.PATIENT\_ID = DG.PATIENT\_ID

AND P.DOC\_ID = DG.DOC\_ID

WHERE P.DOC\_ID = '1164'

AND P.PATIENT\_ID = '11243'

ORDER BY DIAG\_DATE DESC;



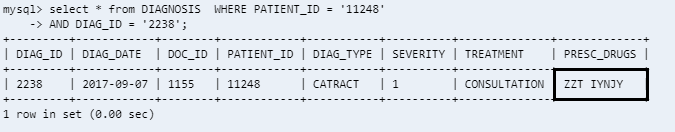
1. **When Doctor prescribes a change of drug for treatment**

UPDATE DIAGNOSIS SET PRESC\_DRUGS = 'RAFAGA KFNVB'

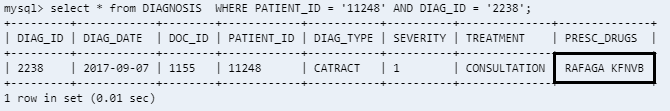
WHERE PATIENT\_ID = '11248'

AND DIAG\_ID = '2238';

**Before Update:**



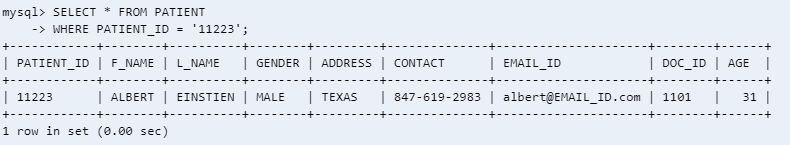
**After Update:**



1. **USER: PATIENT**
2. **When user wants to access his profile details:**

SELECT \* FROM PATIENT

WHERE PATIENT\_ID = '11223';

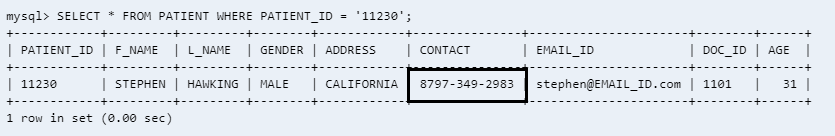


1. **When user needs to update any of his personal details:**

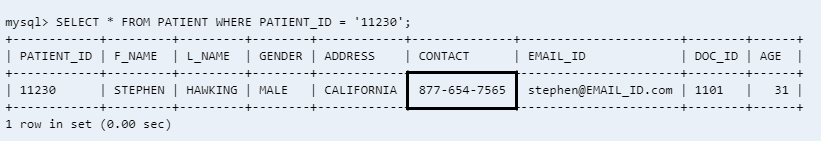
UPDATE PATIENT SET CONTACT = '877-654-7565'

WHERE PATIENT\_ID = '11230';

**Before Update:**



**After Update:**



1. **To find the latest consultation details, maximum of 5 consultations**

SELECT

DOCTOR,

PATIENT\_ID,

PATIENT,

DIAG\_DATE ,

DIAG\_TYPE ,

SEVERITY ,

TREATMENT ,

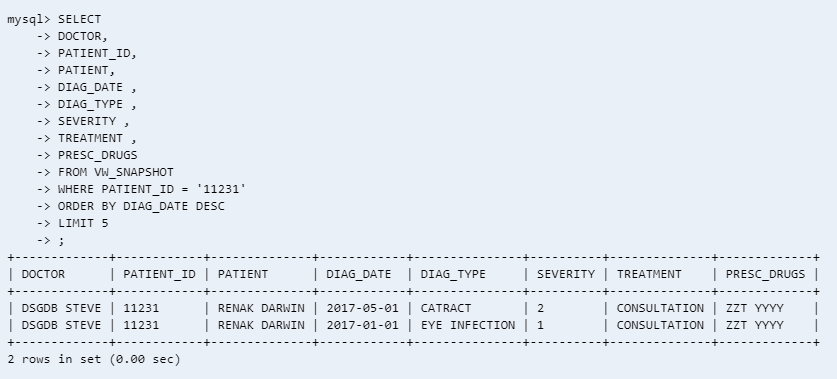
PRESC\_DRUGS

FROM VW\_SNAPSHOT

WHERE PATIENT\_ID = '11231'

ORDER BY DIAG\_DATE DESC

LIMIT 5;



1. **USER: HOSPITAL MANAGEMENT**

**To analyze department-wise performance**

SELECT DP.DEPT\_NAME AS DEPARTMENT,SUM(S.AMOUNT) AS REVENUE

FROM DEPARTMENT DP

JOIN DOCTOR D

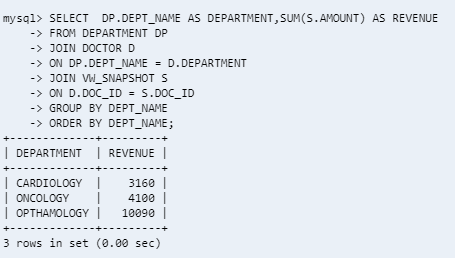
ON DP.DEPT\_NAME = D.DEPARTMENT

JOIN VW\_SNAPSHOT S

ON D.DOC\_ID = S.DOC\_ID

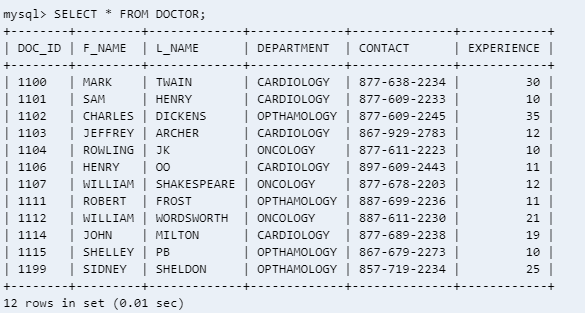
GROUP BY DEPT\_NAME

ORDER BY DEPT\_NAME;

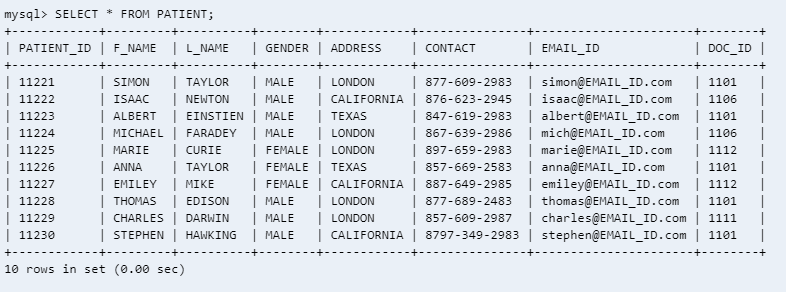


**TABLES**

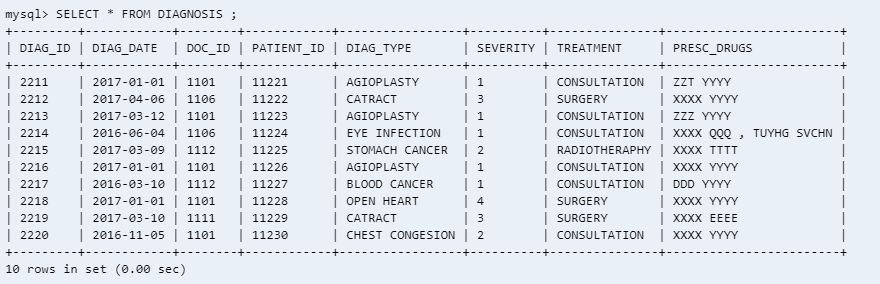
SELECT \* FROM DOCTOR LIMIT 12;



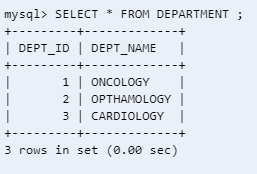
SELECT \* FROM PATIENT LIMIT 10;



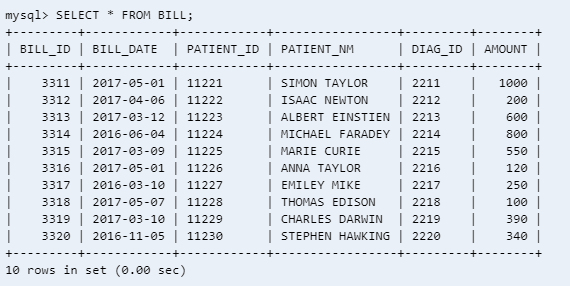
SELECT \* FROM DIAGNOSIS LIMIT 10;



SELECT \* FROM DEPARTMENT ;



SELECT \* FROM BILL LIMIT 10;



**VIEWS:**

CREATE VIEW VW\_SNAPSHOT AS

SELECT

D.DOC\_ID,

CONCAT(D.F\_NAME,' ',D.L\_NAME) AS DOCTOR,

P.PATIENT\_ID,

CONCAT(P.F\_NAME,' ',P.L\_NAME) AS PATIENT,

DG.DIAG\_DATE ,

DG.DIAG\_TYPE ,

DG.SEVERITY ,

DG.TREATMENT ,

DG.PRESC\_DRUGS ,

B.AMOUNT

FROM DOCTOR D

JOIN

PATIENT P

ON D.DOC\_ID = P.DOC\_ID

JOIN DIAGNOSIS DG

ON P.PATIENT\_ID = DG.PATIENT\_ID

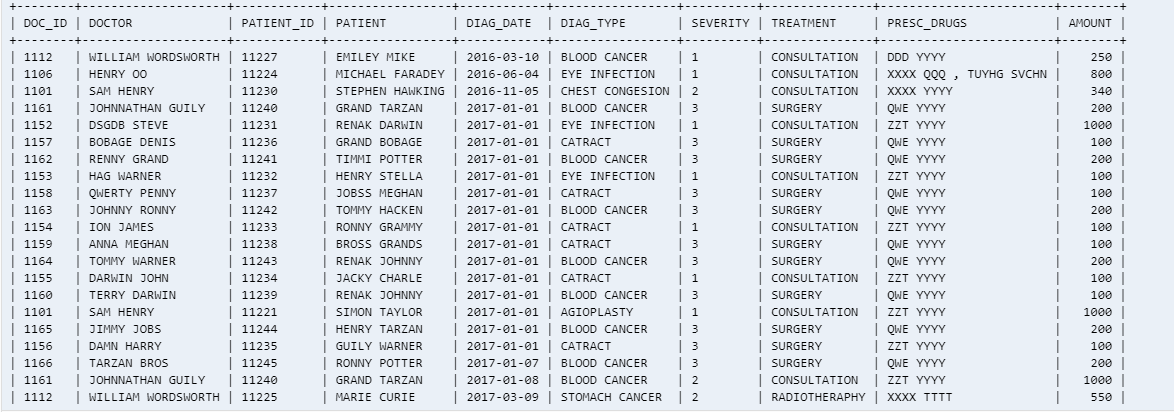
AND D.DOC\_ID = DG.DOC\_ID

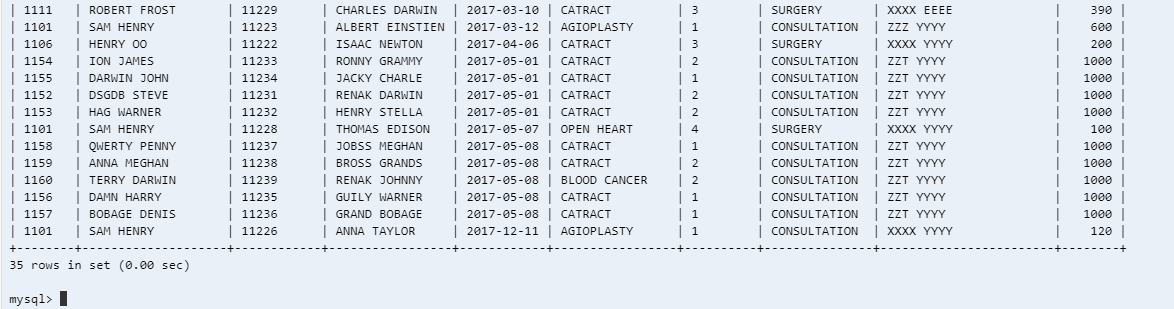
JOIN BILL B

ON P.PATIENT\_ID = B.PATIENT\_ID AND

DG.DIAG\_ID = B.DIAG\_ID

ORDER BY DG.DIAG\_DATE;





CREATE VIEW VW\_DOC\_PAT\_RATIO\_MAT AS

SELECT

CONCAT(D.F\_NAME,' ',D.L\_NAME) DOCTOR\_NAME,

COUNT(P.PATIENT\_ID)

FROM DOCTOR D

JOIN

PATIENT P

ON D.DOC\_ID = P.DOC\_ID

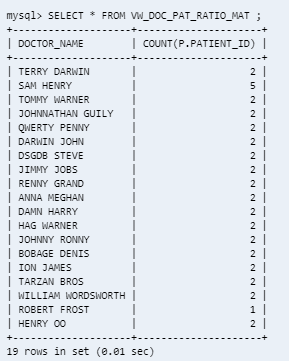
JOIN DIAGNOSIS DG

ON P.PATIENT\_ID = DG.PATIENT\_ID

AND D.DOC\_ID = DG.DOC\_ID

GROUP BY D.DOC\_ID

ORDER BY DG.DIAG\_DATE;



CREATE VIEW VW\_AILMENT\_DIST\_MAT AS

SELECT

P.GENDER ,

DG.DIAG\_TYPE,

COUNT(P.PATIENT\_ID) COUNT

FROM

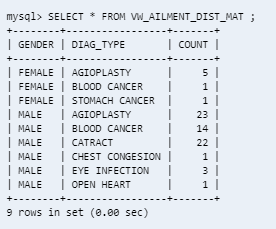
PATIENT P

JOIN DIAGNOSIS DG

ON P.PATIENT\_ID = DG.PATIENT\_ID

GROUP BY P.GENDER,DG.DIAG\_TYPE

ORDER BY P.GENDER,DG.DIAG\_TYPE;



**STORED PROCEDURE:**

CREATE PROCEDURE REVENUE\_METRIX

AS

BEGIN

SELECT A.YEAR,

A.MONTH,

SUM(A.AMOUNT) AS REVENUE\_MAT

FROM (SELECT

EXTRACT(MONTH FROM BILL\_DATE) AS MONTH,

EXTRACT(YEAR FROM BILL\_DATE) AS YEAR,

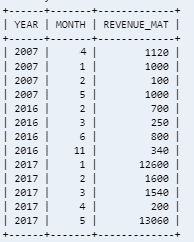
AMOUNT

FROM BILL ) A

GROUP BY A.YEAR,A.MONTH

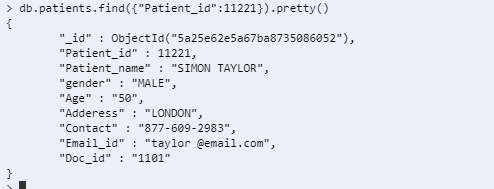
ORDER BY A.YEAR,A.MONTH

END;



1. **NoSQL:**
   1. **USER: LAB TECHNICIAN**
      1. **To retrieve patient details:**

Db.patients.find({“Patient\_id”:11221}).pretty()

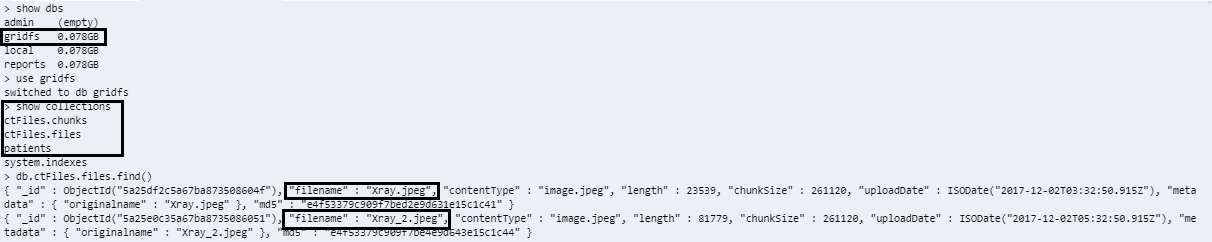


* + 1. **To retrieve and view the images in mongo:**
* **The metadata of the uploaded files are viewed as follows:**

Use gridfs

Db.ctFiles.files.find()

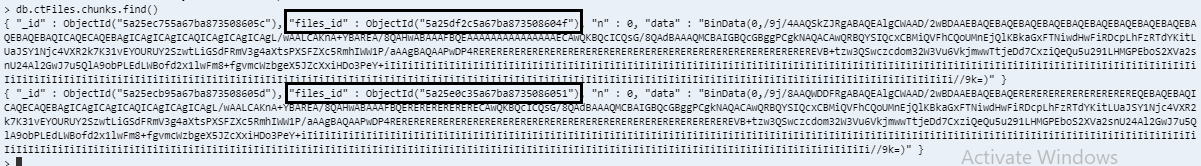
Meta data of uploaded files from collection ctFiles



* Chunks of the uploaded files are viewed as:

Db.ctFiles.chunks.find()

Information about the chunks and the file id:



* + 1. **To retrieve and view the images in application**

By Typing the below host address the image is retrieved from Mongo DB into the application:

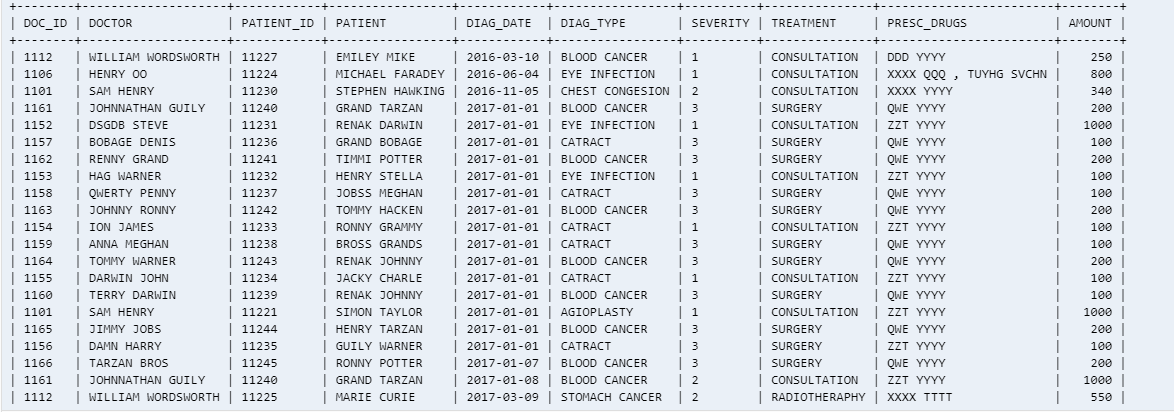
localhost3002/files/Xray.jpeg

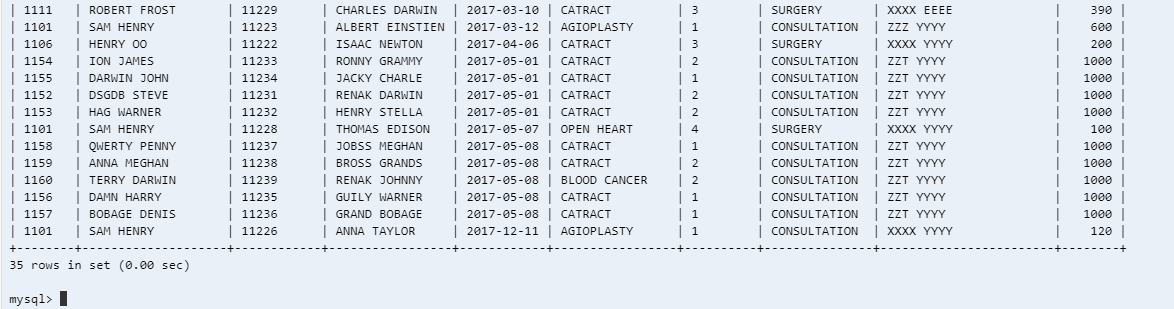
Screenshot of the retrieved image from mongo



1. **BUSINESS MATRICS**
2. **Overall Snapshot of the hospital management – View**

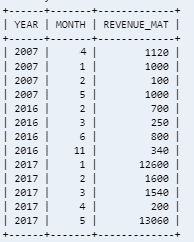
SELECT \* FROM VW\_SNAPSHOT ;





1. **Revenue Metrics – Stored procedure**

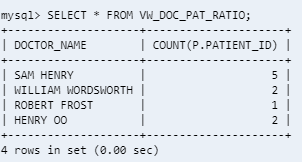
EXEC REVENUE\_METRIX;



Note: Despite the syntax for stored procedure being right, the procedure did not get executed in icloud9. Executed the select statement and derived the above result

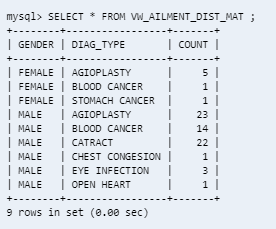
1. **Doctor to Patient Ratio - View**

SELECT \* FROM VIEW VW\_DOC\_PAT\_RATIO\_MAT LIMIT 4 ;



1. **Gender wise Ailment Frequency**

SELECT \* FROM VIEW VW\_AILMENT\_DIST\_MAT ;



1. **PERFORMANCE IMPROVEMENT:**

For performance improvement, indexing was used to tables **PATIENT, DOCTOR and BILL**.As table **DIAGNOSIS** requires frequent updating, indexing was not applied as it will slow down the update process. Refer below tabulation and screenshot for time taken to fetch data before and after indexing. The execution time has reduced after adding index to corresponding tables

Queries:

* CREATE INDEX idx\_pat\_pat\_id

ON PATIENT (PATIENT\_ID);

* CREATE INDEX idx\_doc\_doc\_id

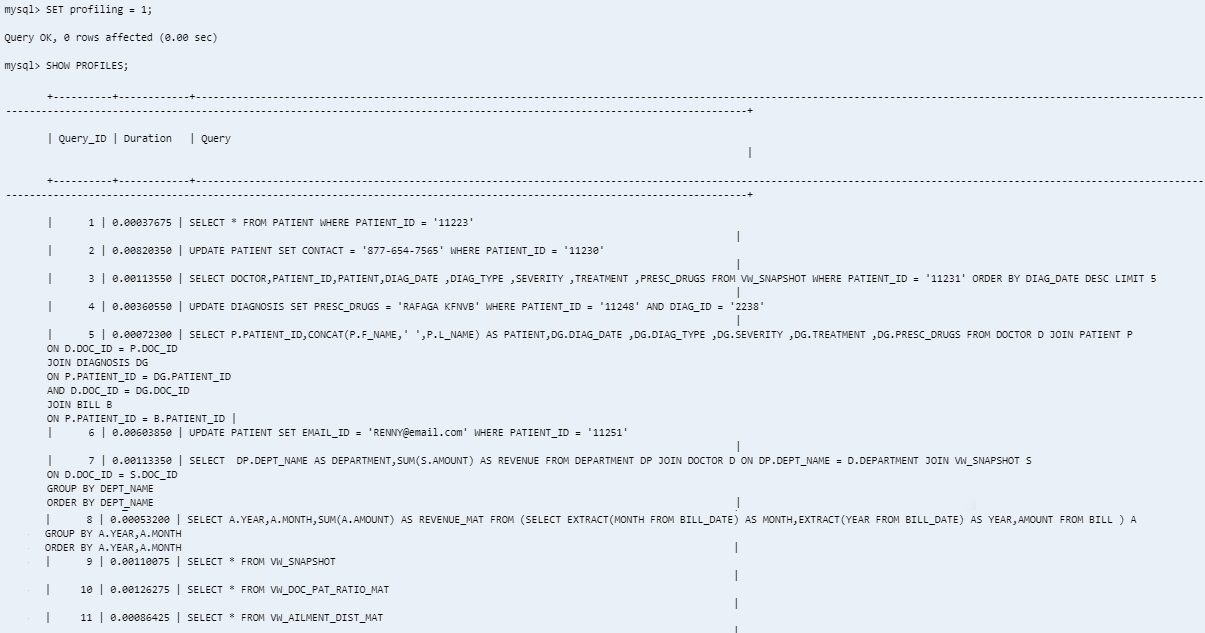
ON DOCTOR (DOC\_ID);

* CREATE INDEX idx\_bill\_bill\_dt

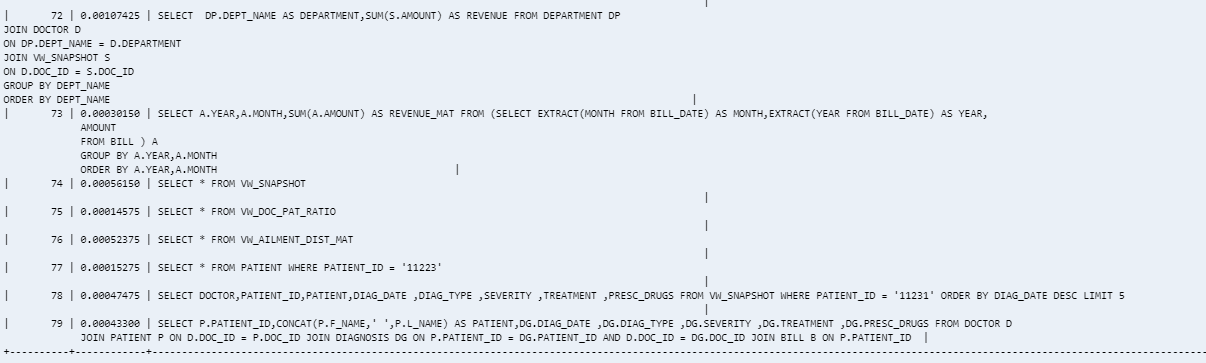
ON BILL (BILL\_DATE);

|  |  |  |  |
| --- | --- | --- | --- |
| **Query\_ID** | **QUERIES** | **DURATION WITHOUT INDEX** | **DURATION WITH INDEX** |
| 1 | SELECT A.YEAR,A.MONTH,SUM(A.AMOUNT) AS REVENUE\_MAT FROM (SELECT EXTRACT(MONTH FROM BILL\_DATE) AS MONTH,EXTRACT(YEAR FROM BILL\_DATE) AS YEAR,  AMOUNT  FROM BILL ) A  GROUP BY A.YEAR,A.MONTH  ORDER BY A.YEAR,A.MONTH ; | 0.000532 | 0.0003015 |
| 2 | SELECT \* FROM VW\_SNAPSHOT ; | 0.00110075 | 0.0005615 |
| 3 | SELECT \* FROM VW\_DOC\_PAT\_RATIO; | 0.00126275 | 0.00014575 |
| 4 | SELECT \* FROM VW\_AILMENT\_DIST\_MAT ; | 0.00086425 | 0.00052375 |
| 5 | SELECT \* FROM PATIENT WHERE PATIENT\_ID = '11223' ; | 0.00037675 | 0.00015275 |
| 6 | SELECT DOCTOR,PATIENT\_ID,PATIENT,DIAG\_DATE ,DIAG\_TYPE ,SEVERITY ,TREATMENT ,PRESC\_DRUGS FROM VW\_SNAPSHOT WHERE PATIENT\_ID = '11231' ORDER BY DIAG\_DATE DESC LIMIT 5 ; | 0.0011355 | 0.00047475 |
| 7 | SELECT P.PATIENT\_ID,CONCAT(P.F\_NAME,' ',P.L\_NAME) AS PATIENT,DG.DIAG\_DATE ,DG.DIAG\_TYPE ,DG.SEVERITY ,DG.TREATMENT ,DG.PRESC\_DRUGS FROM DOCTOR D   JOIN PATIENT P ON D.DOC\_ID = P.DOC\_ID JOIN DIAGNOSIS DG ON P.PATIENT\_ID = DG.PATIENT\_ID AND D.DOC\_ID = DG.DOC\_ID JOIN BILL B ON P.PATIENT\_ID = B.PATIENT\_ID  AND  DG.DIAG\_ID = B.DIAG\_ID   WHERE D.DOC\_ID = '1166'  AND P.PATIENT\_ID = '11245' ORDER BY DIAG\_DATE DESC ; | 0.000723 | 0.000433 |
| 8 | SELECT DP.DEPT\_NAME AS DEPARTMENT,SUM(S.AMOUNT) AS REVENUE FROM DEPARTMENT DP JOIN DOCTOR D ON DP.DEPT\_NAME = D.DEPARTMENT JOIN VW\_SNAPSHOT S ON D.DOC\_ID = S.DOC\_ID GROUP BY DEPT\_NAME ORDER BY DEPT\_NAME ; | 0.0011335 | 0.00107425 |

**Screenshot without indexing:**



**Screenshot After Indexing:**



**Sample:**

SELECT DP.DEPT\_NAME AS DEPARTMENT,SUM(S.AMOUNT) AS REVENUE

FROM DEPARTMENT DP

JOIN DOCTOR D

ON DP.DEPT\_NAME = D.DEPARTMENT

JOIN VW\_SNAPSHOT S

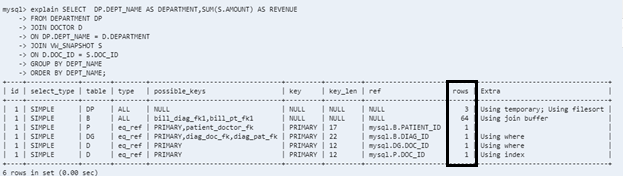
ON D.DOC\_ID = S.DOC\_ID

GROUP BY DEPT\_NAME

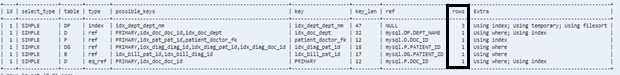
ORDER BY DEPT\_NAME;

From below screenshots its clear that the number of rows processed for the above query has reduced after indexing.

Before Indexing:



**After Indexing:**



1. **SUMMARY:**

The project gave me hands on experience on SQL and NoSQL. Gained deeper knowledge on handling Database - starting from design, implementation and validation of a database application. Also, explored the file handling features of mongo using Nodejs and GridFS starting from installation to file data retrieval. Got a better perspective of DBMS from a business point.

1. **hardest part of this project**

* Installation and implementing the GridFS and mongofiles features using Nodejs
* Performance improvement for SQL
* Populating source dataset for both structured and Unstructured DB
* Deciding between SQL Vs NoSQL

1. **problems faced this project**

* Designing the management system
  + Choosing the tables and their attributes were challenging
  + Framing the relationship between the tables
  + Designing views and stored procedure for Business Metrics
* Execution errors while deploying the code
* Performance improvement for SQL
* Choosing the appropriate attribute to perform indexing was challenging
* Preparing large dataset for testing the performance improvement code consumed more time
* Installation and implementing errors while using Nodejs feature to upload images and vidoes as mongofiles

1. **How was the problem solved?**

* **Designing the management system**
* The table attribute and constraints were planned and penned down before writing the scripts.
* The views and stored procedures were also designed based on the Business Metrics requirements
* **Execution errors while deploying the code**

Based on the error code returned, corresponding solution were referred online and the errors were fixed

* **Performance improvement for SQL**
* Major impacting columns were identified. Indexing was applied to those columns. The time taken to execute the queries before and after applying index were noted and compared.
* indexing was used to tables **PATIENT (PATIENT\_ID), DOCTOR (DOC\_ID) and BILL (BILL\_DATE)**.
* As table **DIAGNOSIS** requires frequent updating, indexing was not applied as it will slow down the update process.
* **Installation and implementing errors while Using Nodejs feature The file upload was done using Nodejs,gridFs features of mongo**
* Referenced SQL MogoDB manual
* The fix for the error codes were searched in various forums online
* Referenced <https://github.com/rahil471/fileupload-nodejs-gridfs-angular2>
* For the Nodejs setup

1. **If I were to do this project again**

* Include Trigger on deletion of records from tables. Since hospital data are very confidential and important it is necessary to record the changes done to data. Also,they can be referred incase of auditing or any future purpose
* Include sharding features in NoSQL for performance improvement
* Improvise on the design by analyzing the real hospital scenario
* Include trend analysis for each patient based on the Lab test results in NoSQL
* Expand the scope to handle Drug Department

1. **Reference:**

<https://github.com/rahil471/fileupload-nodejs-gridfs-angular2>

<https://www.w3schools.com/nodejs/nodejs_mongodb_createcollection.asp>