ADVANCED FEATURES OF AN EMR DATABASE SYSTEM

Prepared By:
Shweta Rajaram Patil(800989198)
Meetu Uthra(800985777)
Tushar Arvind(800823103)

Course: Applied Databases
University of North Carolina at Charlotte

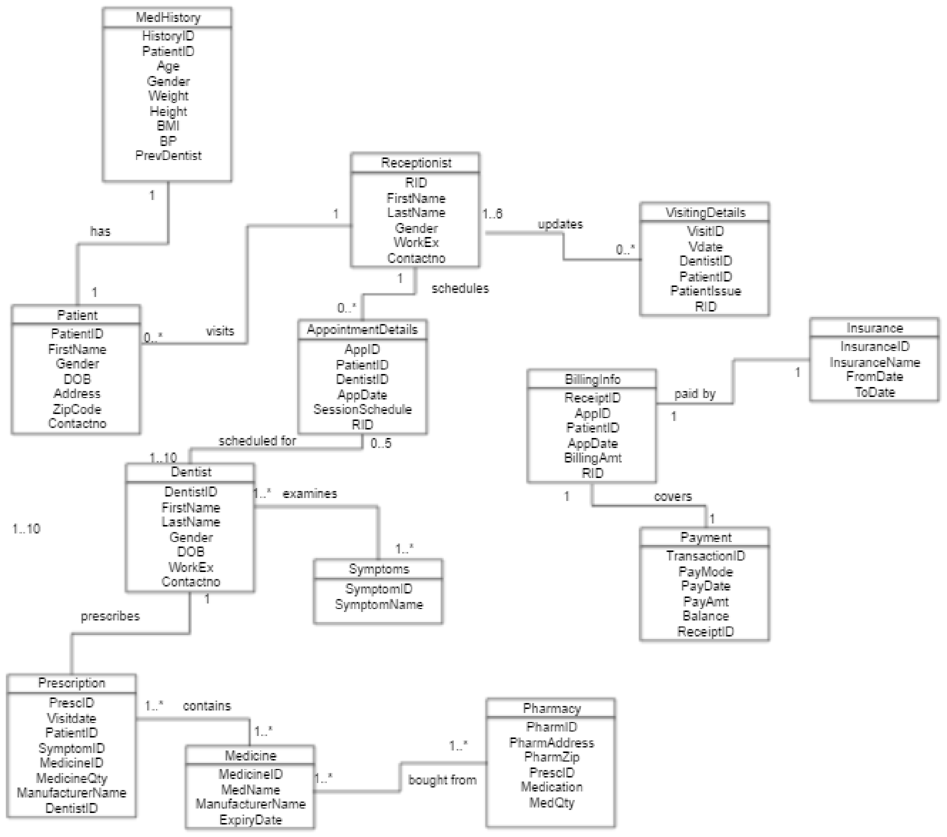
Project Scope:

This Electronic Medical Records system captures information of Patients dealing with Dental disease. The key focus is to be able to retrieve Patient's Med History, Visiting details, Insurance Info, Appointment details, Billing Info and the Dentist associated with the Patient. The database will support the retrieval of information of all patients that have been admitted in a hospital over the past one month.

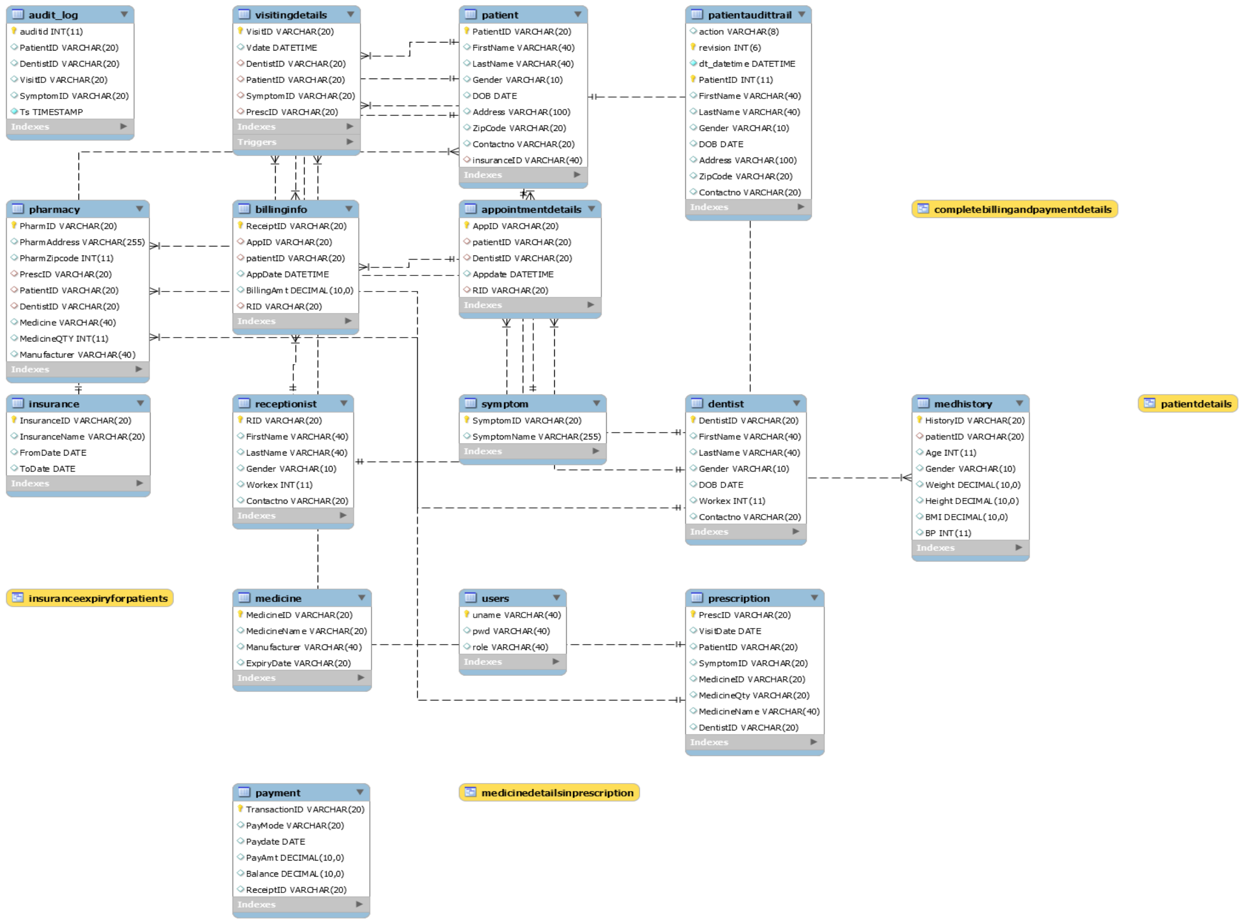
Project Constraints

- A patient can take treatment or be associated with only one Dentist at a time.
- Max 10 appointments will be handled by the clinic in a day.
- A patient can book only one appointment in a day.
- For every appointment booked by a patient he will be provided with one Visit ID which will be valid for one day using which he can make multiple visits in a day.

UML Structure:



ER Diagram for the EMR Database: BiteIn Dental Clinic:



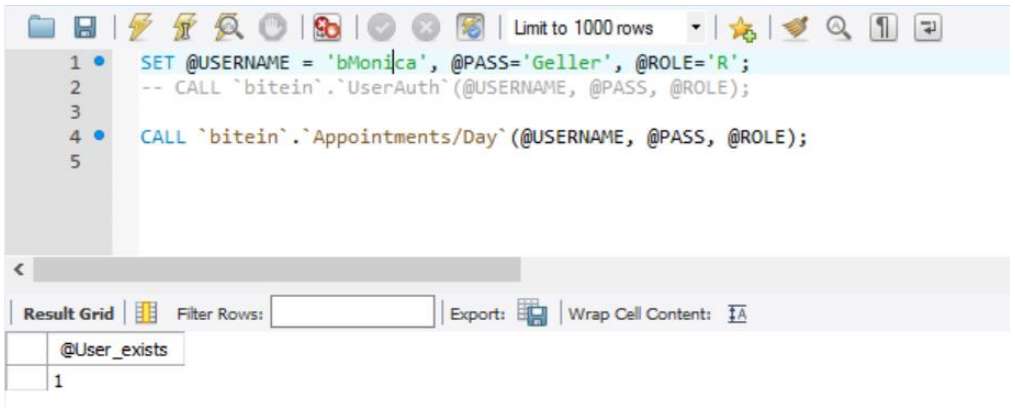
User Authentication:

Authentication is achieved using username, password and role. Following Stored Procedure is used to implement authentication

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `UserAuth`(IN `USERNAME` VARCHAR(64), IN
`PASS` VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
   SELECT @present := COUNT(uname) FROM users WHERE concat(uname,pwd,role) =
concat(@USERNAME,@PASS,@ROLE);
   IF @present > 0 THEN  SET @User_exists = 1;
   ELSE SET @User_exists = 0;
   END IF;
   SELECT @User_exists;
END$$
DELIMITER ;
```

CORRECT USERNAME & PASSWORD:

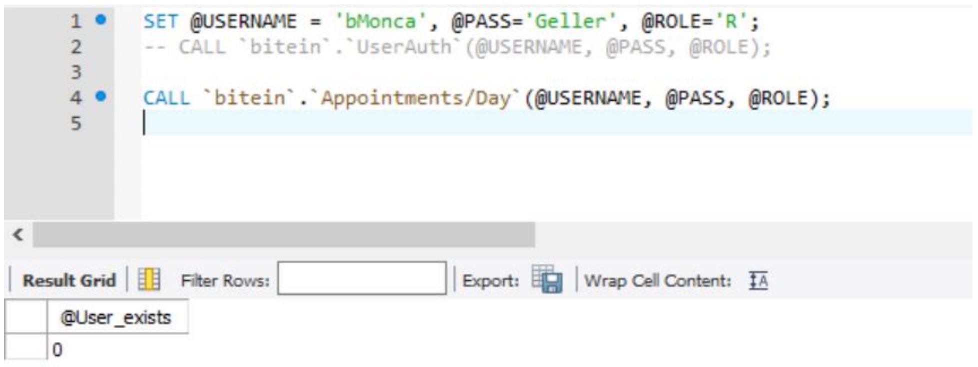
```
SET @USERNAME = 'bRoss', @PASS='Geller', @ROLE='D';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);
CALL `bitein`.`Appointments/Day`(@USERNAME, @PASS, @ROLE);
```



```
SET @USERNAME = 'bMonica', @PASS='Geller', @ROLE='R';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`Appointments/Day`(@USERNAME, @PASS, @ROLE);
```

@User_exists
1

INCORRECT USERNAME & PASSWORD:



```
SET @USERNAME = 'bMonca', @PASS='Geller', @ROLE='R';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`Appointments/Day`(@USERNAME, @PASS, @ROLE);
```

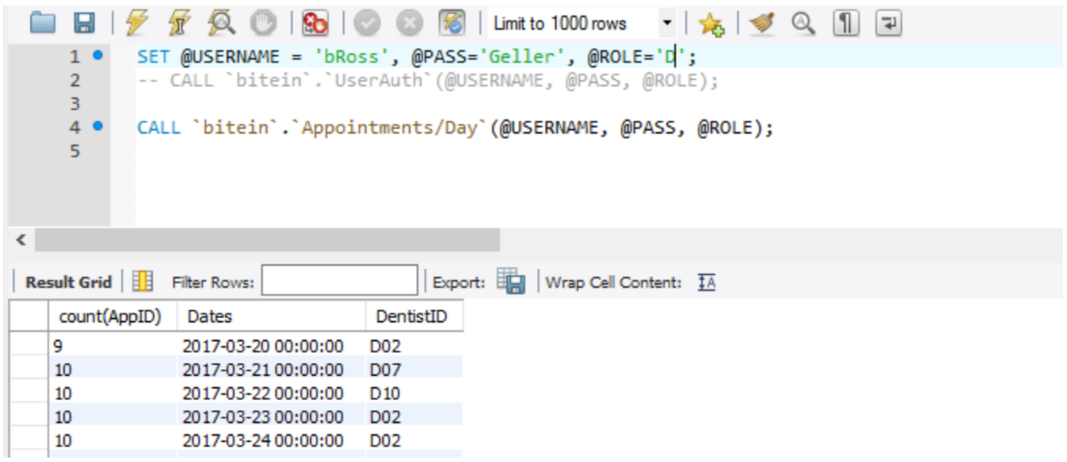
@User_exists
0

User Authorization

After user is authenticated, for accessing any of the tables, user have an authorized access to that table. This requirement is achieved using Role Based Authorization. We have 5 user roles defined:

a. Dentist
b. Receptionist
c. Patient
d. Pharmacist
e. Admin

<u>USER EXISTS AND AUTHORIZED:</u>

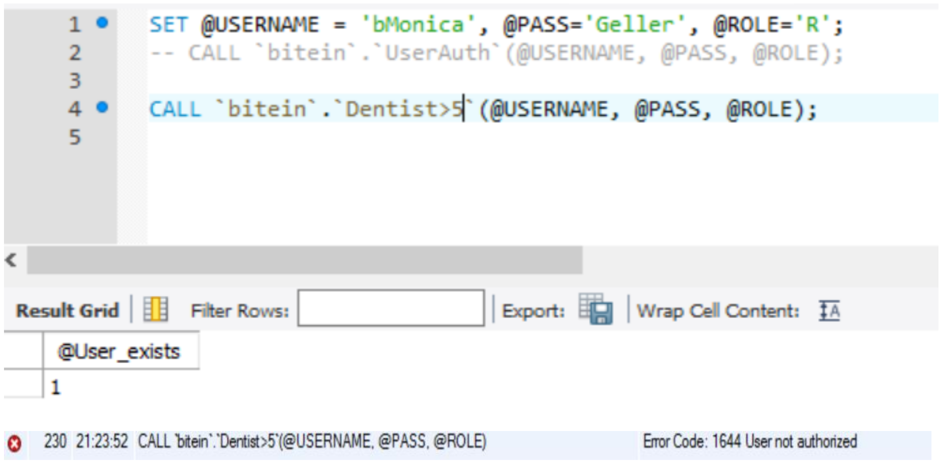


```
SET @USERNAME = 'bRoss', @PASS='Geller', @ROLE='D';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`Appointments/Day`(@USERNAME, @PASS, @ROLE);
```

count(AppID)	Dates	DentistID
9	2017-03-20 00:00:00	D02
10	2017-03-21 00:00:00	D07
10	2017-03-22 00:00:00	D10
10	2017-03-23 00:00:00	D02
10	2017-03-24 00:00:00	D02

<u>USER EXITS AND NOT AUTHORIZED:</u>

We authenticate the user that is trying to access a table. If user is not authorized, Error message is thrown: “User not authorized”.



```
SET @USERNAME = 'bMonica', @PASS='Geller', @ROLE='R';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`Dentist>5`(@USERNAME, @PASS, @ROLE);
```

@User_exists
1

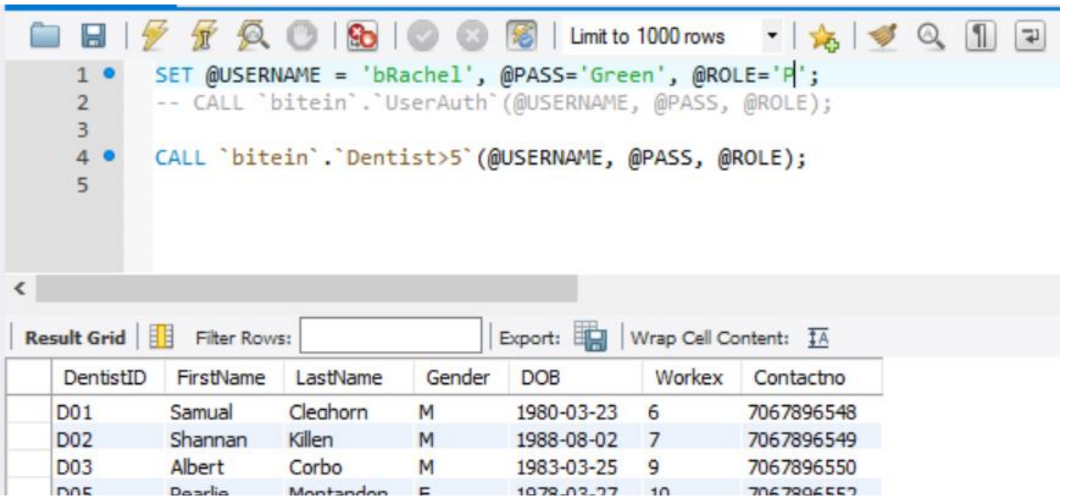
```
230  21:23:52  CALL `bitein`.`Dentist>5`(@USERNAME, @PASS, @ROLE)    Error Code: 1644 User not authorized
```

Stored Procedures:

Every stored procedure is a capability that can be assigned as a function of a user role. Below stored procedures have limitations according to the roles specified.

- **Only patient and admin should be able to see dentist work experience**

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `Dentist>5`(IN `USERNAME` VARCHAR(64), IN `PASS`
VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);
IF @User_exists > 0 THEN
IF @ROLE='P' OR @ROLE='Adm'  THEN
        SELECT *FROM BITEIN.DENTIST
        WHERE WORKEX>5;
ELSE
        SET @message_text = ('User not authorized');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
END IF;
END IF;
END$$
DELIMITER ;
```

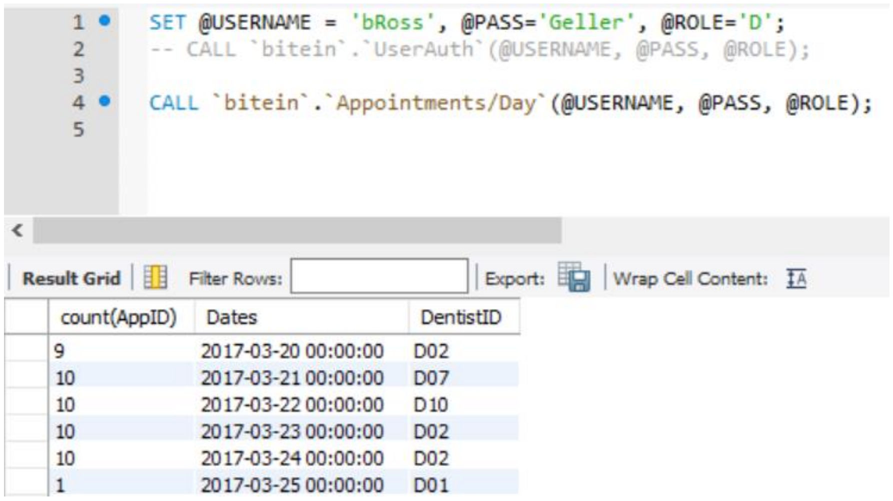


```
SET @USERNAME = 'bRachel', @PASS='Green', @ROLE='P';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`Dentist>5`(@USERNAME, @PASS, @ROLE);
```

DentistID	FirstName	LastName	Gender	DOB	Workex	Contactno
D01	Samual	Cleghorn	M	1980-03-23	6	7067896548
D02	Shannan	Killen	M	1988-08-02	7	7067896549
D03	Albert	Corbo	M	1983-03-25	9	7067896550
D05	Pearlie	Montandon	F	1978-03-27	10	7067896552

- **Only Dentist and Admin should be able to see his appointments**

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `Appointments/Day`(IN `USERNAME` VARCHAR(64), IN
`PASS` VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);
IF @User_exists > 0 THEN
IF @ROLE='D' OR @ROLE='Adm'  THEN
        select count(AppID), Appdate as Dates, DentistID from appointmentdetails
        group by(appdate) order by Appdate ASC;
ELSE
        SET @message_text = ('User not authorized');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
END IF;
END IF;
END$$
DELIMITER ;
```

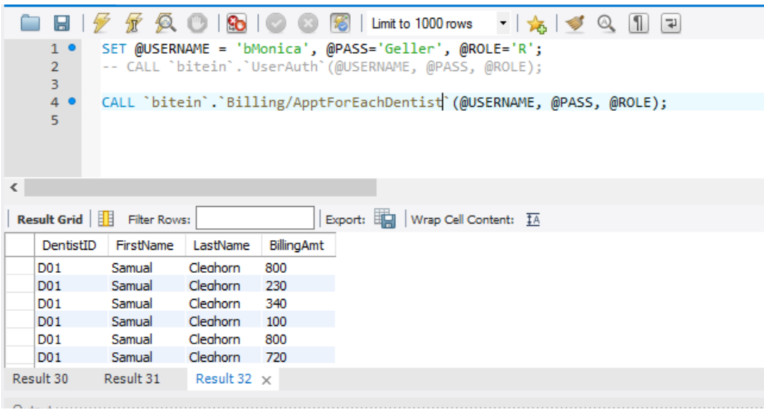


```
SET @USERNAME = 'bRoss', @PASS='Geller', @ROLE='D';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`Appointments/Day`(@USERNAME, @PASS, @ROLE);
```

count(AppID)	Dates	DentistID
9	2017-03-20 00:00:00	D02
10	2017-03-21 00:00:00	D07
10	2017-03-22 00:00:00	D10
10	2017-03-23 00:00:00	D02
10	2017-03-24 00:00:00	D02
1	2017-03-25 00:00:00	D01

Only Receptionist and Admin should be able to see billing details

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `Billing/ApptForEachDentist`(IN `USERNAME` VARCHAR(64), IN `PASS` VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);
IF @User_exists > 0 THEN
IF @ROLE='R' OR @ROLE='Adm'  THEN
        select d.DentistID, d.FirstName, d.LastName, BillingAmt from billinginfo as b join appointmentdetails as a on b.AppID=a.AppID join dentist as d on d.DentistID = a.DentistID;
ELSE
        SET @message_text = ('User not authorized');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
END IF;
END IF;
END$$
DELIMITER ;
```

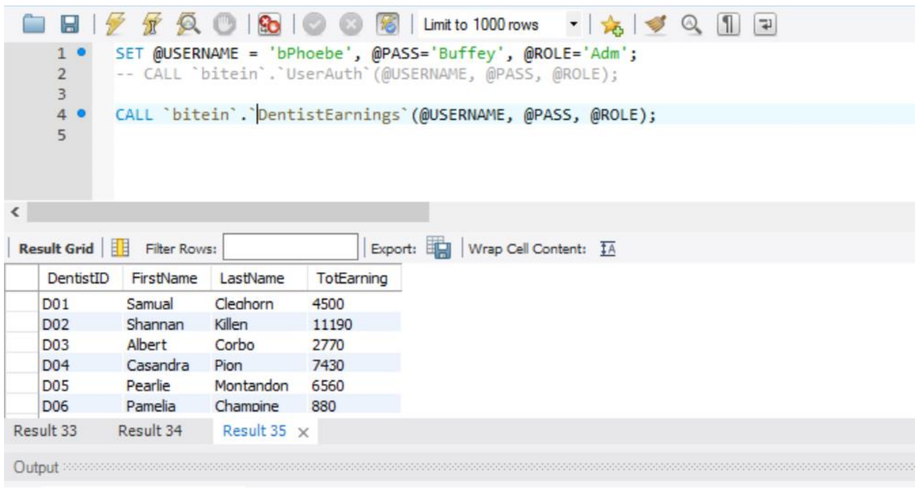


```
SET @USERNAME = 'bMonica', @PASS='Geller', @ROLE='R';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`Billing/ApptForEachDentist`(@USERNAME, @PASS, @ROLE);
```

DentistID	FirstName	LastName	BillingAmt
D01	Samual	Cleghorn	800
D01	Samual	Cleghorn	230
D01	Samual	Cleghorn	340
D01	Samual	Cleghorn	100
D01	Samual	Cleghorn	800
D01	Samual	Cleghorn	720

Only Admin should be able to see Dentist earnings for the clinic

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `DentistEarnings`(IN `USERNAME` VARCHAR(64), IN
`PASS` VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);
IF @User_exists > 0 THEN
IF @ROLE='Adm' THEN
        select d.DentistID, d.FirstName, d.LastName, sum(BillingAmt) as TotEarning from billinginfo as b join
appointmentdetails as a
        on b.AppID=a.AppID join dentist as d on d.DentistID = a.DentistID
        group by(DentistID);
ELSE
        SET @message_text = ('User not authorized');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
END IF;
END IF;
END$$
DELIMITER ;
```

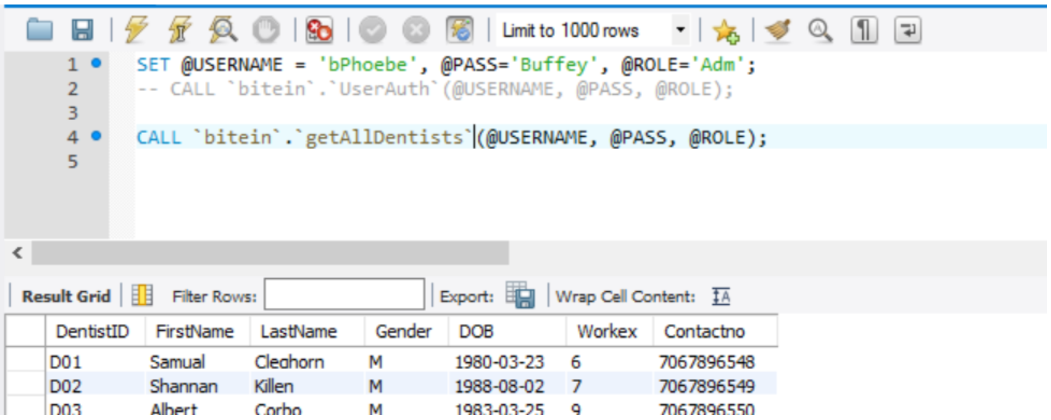


```
SET @USERNAME = 'bPhoebe', @PASS='Buffey', @ROLE='Adm';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`DentistEarnings`(@USERNAME, @PASS, @ROLE);
```

DentistID	FirstName	LastName	TotEarning
D01	Samual	Cleghorn	4500
D02	Shannan	Killen	11190
D03	Albert	Corbo	2770
D04	Casandra	Pion	7430
D05	Pearlie	Montandon	6560
D06	Pamelia	Champine	880

All authorized users can see all dentists

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `getAllDentists`(IN `USERNAME` VARCHAR(64), IN
`PASS` VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);
IF @User_exists > 0 THEN
        SELECT *FROM BITEIN.DENTIST;
ELSE
        SET @message_text = ('User not authorized');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
END IF;
END$$
DELIMITER ;
```

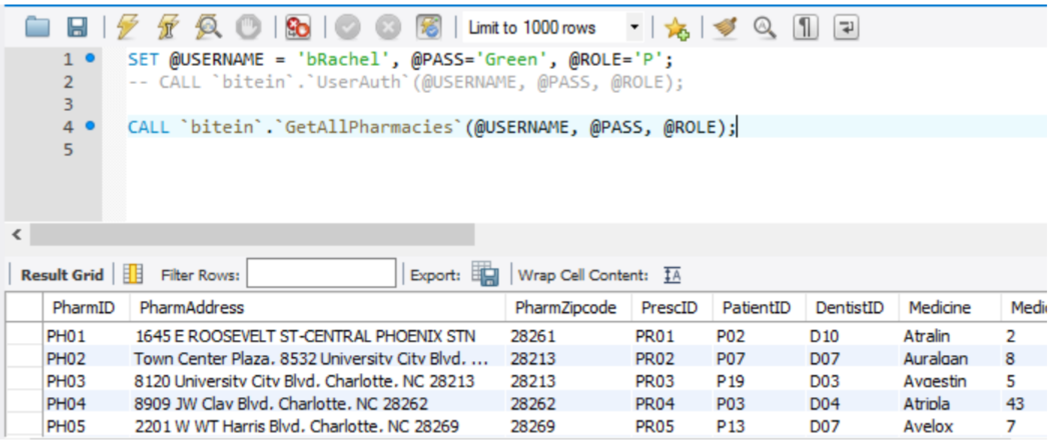


```
SET @USERNAME = 'bPhoebe', @PASS='Buffey', @ROLE='Adm';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`getAllDentists`(@USERNAME, @PASS, @ROLE);
```

DentistID	FirstName	LastName	Gender	DOB	Workex	Contactno
D01	Samual	Cleghorn	M	1980-03-23	6	7067896548
D02	Shannan	Killen	M	1988-08-02	7	7067896549
D03	Albert	Corbo	M	1983-03-25	9	7067896550

Patient, Receptionist and admin can see all pharmacies

DELIMITER $
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetAllPharmacies`(IN `USERNAME` VARCHAR(64), IN `PASS` VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);
IF @User_exists > 0 THEN
IF @ROLE='R' OR 'P' OR @ROLE='Adm' THEN
        *SELECT *FROM BITEIN.PHARMACY;*
ELSE
        SET @message_text = ('User not authorized');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
END IF;
END IF;
END$$
DELIMITER ;



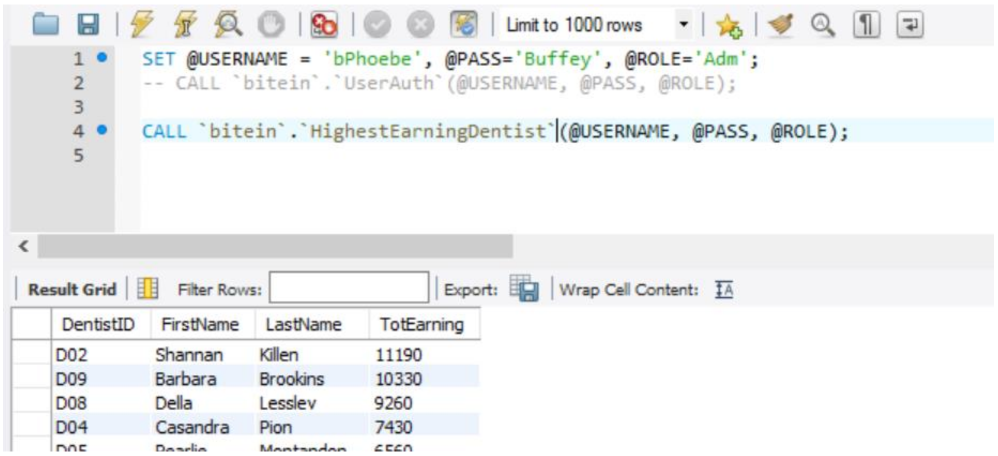
```
SET @USERNAME = 'bRachel', @PASS='Green', @ROLE='P';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`GetAllPharmacies`(@USERNAME, @PASS, @ROLE);
```

PharmID	PharmAddress	PharmZipcode	PrescID	PatientID	DentistID	Medicine	Medi
PH01	1645 E ROOSEVELT ST-CENTRAL PHOENIX STN	28261	PR01	P02	D10	Atralin	2
PH02	Town Center Plaza, 8532 University City Blvd, ...	28213	PR02	P07	D07	Auralgan	8
PH03	8120 University City Blvd, Charlotte, NC 28213	28213	PR03	P19	D03	Avgestin	5
PH04	8909 JW Clay Blvd, Charlotte, NC 28262	28262	PR04	P03	D04	Atripla	43
PH05	2201 W WT Harris Blvd, Charlotte, NC 28269	28269	PR05	P13	D07	Avelox	7

Admin can see Highest earning dentist

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `HighestEarningDentist`(IN `USERNAME` VARCHAR(64), IN `PASS` VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

```
IF @User_exists > 0 THEN
IF @ROLE='Adm' THEN
        select d.DentistID, d.FirstName, d.LastName, sum(BillingAmt) as TotEarning from billinginfo as b join
appointmentdetails as a
        on b.AppID=a.AppID join dentist as d on d.DentistID = a.DentistID
        group by(DentistID) order by TotEarning DESC;
ELSE
        SET @message_text = ('User not authorized');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
END IF;
END IF;
END$$
DELIMITER ;
```

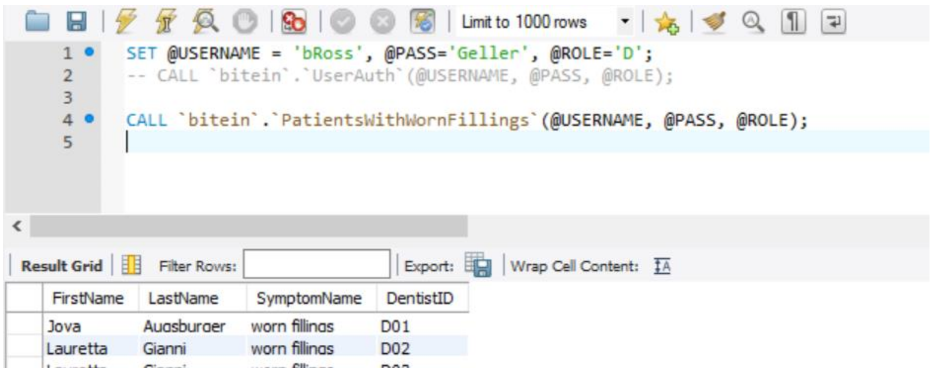


```
SET @USERNAME = 'bPhoebe', @PASS='Buffey', @ROLE='Adm';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`HighestEarningDentist`(@USERNAME, @PASS, @ROLE);
```

DentistID	FirstName	LastName	TotEarning
D02	Shannan	Killen	11190
D09	Barbara	Brookins	10330
D08	Della	Lesslev	9260
D04	Casandra	Pion	7430
D05	Pearlie	Montandon	6560

Only Dentist and Admin can see patient issue details

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `PatientsWithWornFillings`(IN `USERNAME`
VARCHAR(64), IN `PASS` VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);
IF @User_exists > 0 THEN
IF @ROLE='D' OR @ROLE='Adm'  THEN
        select distinct FirstName, LastName, SymptomName, DentistID from patient as p join prescription as pr
        on p.PatientID=pr.PatientID join symptom s on s.SymptomID=pr.SymptomID
        where SymptomName='worn fillings'
        group by(dentistID);
ELSE
        SET @message_text = ('User not authorized');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
END IF;
END IF;
END$$
DELIMITER ;
```

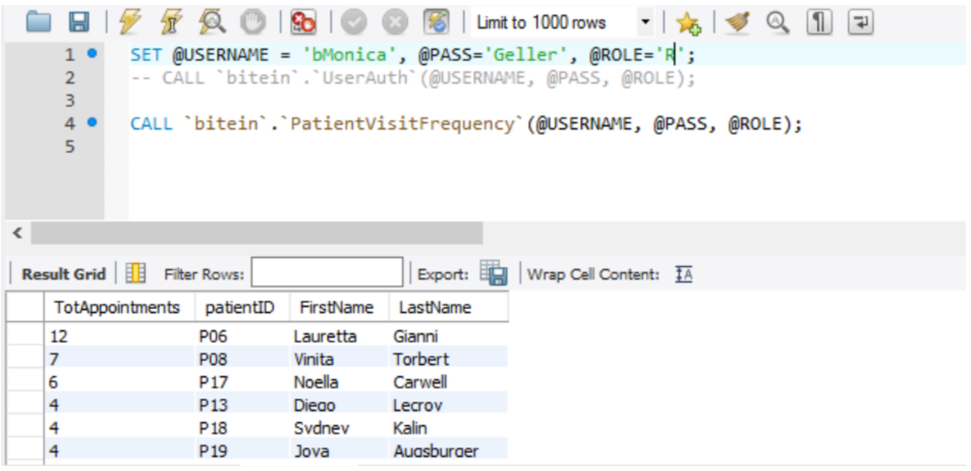


```
SET @USERNAME = 'bRoss', @PASS='Geller', @ROLE='D';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`PatientsWithWornFillings`(@USERNAME, @PASS, @ROLE);
```

FirstName	LastName	SymptomName	DentistID
Jova	Augsburger	worn fillings	D01
Lauretta	Gianni	worn fillings	D02

Admin and Receptionist can see patient visits and frequency of their visits

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `PatientVisitFrequency`(IN `USERNAME` VARCHAR(64),
IN `PASS` VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);
IF @User_exists > 0 THEN
IF @ROLE='R' OR @ROLE='Adm' THEN
        select count(AppID) as TotAppointments, p.patientID, FirstName, LastName
        from appointmentdetails a join patient p on a.patientID=p.PatientID
        group by(patientID) order by TotAppointments DESC;
ELSE
        SET @message_text = ('User not authorized');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
END IF;
END IF;
END$$
DELIMITER ;
```



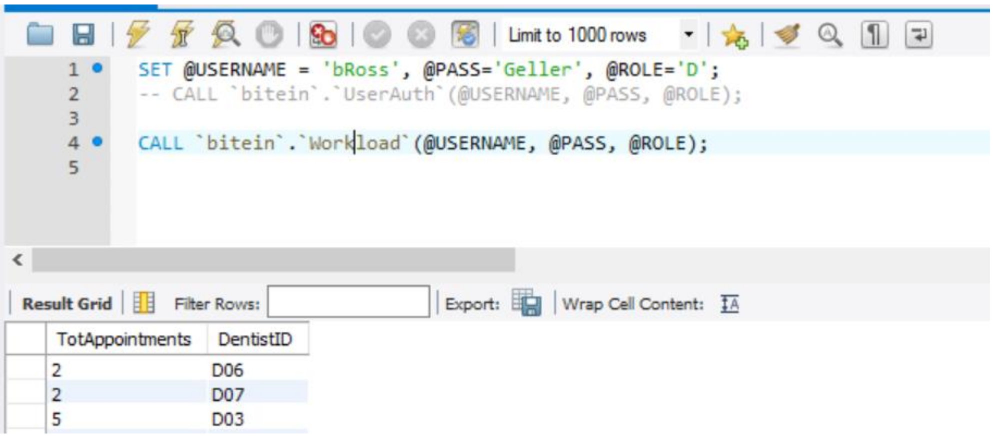
```
SET @USERNAME = 'bMonica', @PASS='Geller', @ROLE='R';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`PatientVisitFrequency`(@USERNAME, @PASS, @ROLE);
```

TotAppointments	patientID	FirstName	LastName
12	P06	Lauretta	Gianni
7	P08	Vinita	Torbert
6	P17	Noella	Carwell
4	P13	Diego	Lecroy
4	P18	Sydney	Kalin
4	P19	Jova	Augsburger

Only Dentist and Admin can see dentist workload

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `WorkLoad`(IN `USERNAME` VARCHAR(64), IN `PASS`
VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);
IF @User_exists > 0 THEN
IF @ROLE='D' OR @ROLE='Adm'  THEN
```

```
        select count(AppID) as TotAppointments, DentistID from appointmentdetails
        group by(DentistID) order by TotAppointments ASC;
ELSE
        SET @message_text = ('User not authorized');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
END IF;
END IF;
END$$
DELIMITER ;
```

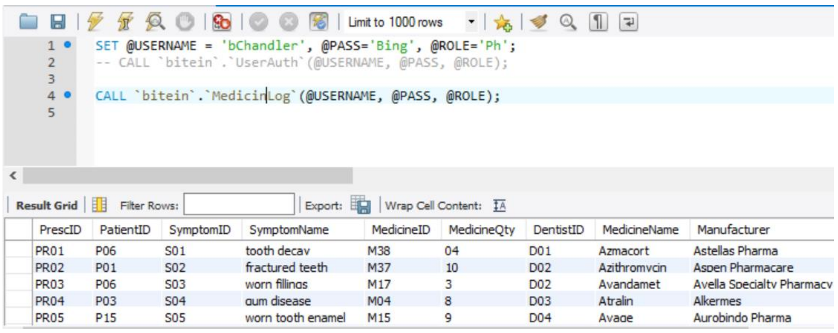


```
SET @USERNAME = 'bRoss', @PASS='Geller', @ROLE='D';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`Workload`(@USERNAME, @PASS, @ROLE);
```

TotAppointments	DentistID
2	D06
2	D07
5	D03

Only pharmacist and admin can see the Medicine details

```
 DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `MedicinLog`(IN `USERNAME` VARCHAR(64), IN `PASS`
VARCHAR(20), IN `ROLE` VARCHAR(5))
BEGIN
CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);
IF @User_exists > 0 THEN
IF @ROLE='Ph' THEN
        SELECT PrescID, PatientID, p.MedicineID, MedicineQty, m.MedicineName, Manufacturer, ExpiryDate
        FROM
        Medicine m JOIN prescription p
        ON m.medicineID=p.medicineID
ELSE
        SET @message_text = ('User not authorized');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
END IF;
END IF;
END$$
DELIMITER ;
```



```
SET @USERNAME = 'bChandler', @PASS='Bing', @ROLE='Ph';
-- CALL `bitein`.`UserAuth`(@USERNAME, @PASS, @ROLE);

CALL `bitein`.`MedicinLog`(@USERNAME, @PASS, @ROLE);
```

PrescID	PatientID	SymptomID	SymptomName	MedicineID	MedicineQty	DentistID	MedicineName	Manufacturer
PR01	P06	S01	tooth decay	M38	04	D01	Azmacort	Astellas Pharma
PR02	P01	S02	fractured teeth	M37	10	D02	Azithromycin	Aspen Pharmacare
PR03	P06	S03	worn fillings	M17	3	D02	Avandamet	Avella Specialty Pharmacy
PR04	P03	S04	gum disease	M04	8	D03	Atralin	Alkermes
PR05	P15	S05	worn tooth enamel	M15	9	D04	Avage	Aurobindo Pharma

Database Tables:

- Dentist(DentistID, FirstName, LastName, Gender, DOB, WorkEx, Contactno)
- Insurance(InsuranceID, Name, FromDate, ToDate)
- Patient(PatientID, FirstName, LastName, Gender, DOB, Address, ZipCode,Contactno)
- AppointmentDetails(AppID, PatientID, DentistID, AppDate, SessionSchedule, RID)
- BillingInfo(ReceiptID, AppID, PatientID, AppDate, BillingAmt, RID)
- MedHistory(HistoryID, PatientID, Age, Gender, Weight, Height, BMI, BP)
- Medicine(MedicineID, MedicineName, Manufacturer, ExpiryDate)
- Payment(TransactionID, PayMode, PayDate, PayAmt, Balance, ReceiptID)
- Symptom(SymptomID, SymptomName)
- Precription(PrescID VisitDate, PatientID, SymptomID, MedicineID, MedicineQty, MedicineName, DentistID)
- VisitingDetails(VisitID, Vdate, DentistID, PatientID, SymptomID, PrescriptionID)
- Pharmacy(PharmID, PharmAddress, PharmZipcode, PrescID, PatientID, DentistID, Medicine, MedicineQTY, Manufacturer)
- User(uname, pwd, role)

DB Script:

Attached separately with the report.

User Roles:

User Roles are defined in a user table which also contains usernames and passwords of the users.

D=Dentist
P=Patient
Ph=Pharmacist
R=Receptionist
Adm=DB Admin

```
CREATE TABLE users
(uname varchar(40) primary key,
pwd varchar(40),
role varchar(40));
INSERT INTO `bitein`.`users` (`uname`, `pwd`, `role`) VALUES ('bRoss', 'Geller', 'D');
INSERT INTO `bitein`.`users` (`uname`, `pwd`, `role`) VALUES ('bMonica', 'Geller', 'R');
INSERT INTO `bitein`.`users` (`uname`, `pwd`, `role`) VALUES ('bRachel', 'Green', 'P');
INSERT INTO `bitein`.`users` (`uname`, `pwd`, `role`) VALUES ('bChandler', 'Bing', 'Ph');
INSERT INTO `bitein`.`users` (`uname`, `pwd`, `role`) VALUES ('bPhoebe', 'Buffey', 'Adm');
```

Views:

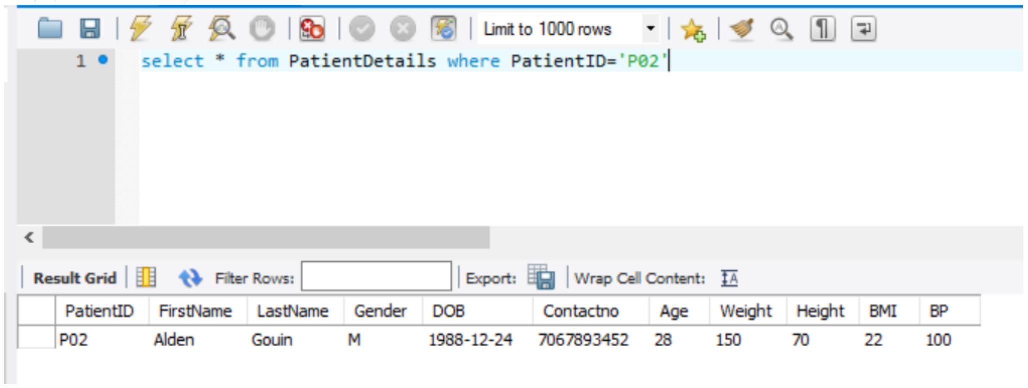
Views are used for ease of access for different user roles. Below are few views designed for the database.

- Medicine Details in Prescription – For the dentist

CREATE VIEW `MedicineDetailsInPrescription` AS
SELECT PrescID, PatientID, p.SymptomID, SymptomName, p.MedicineID, MedicineQty, DentistID, m.MedicineName, Manufacturer, ExpiryDate
FROM
Medicine m JOIN prescription p
ON m.medicineID=p.medicineID
JOIN Symptom s
ON s.SymptomID=p.SymptomID;

- Patient Details visiting the clinic – For the dentist

CREATE VIEW `PatientDetails` AS
SELECT P.PatientID, FirstName, LastName, p.Gender, DOB, Contactno,
Age, Weight, Height, BMI, BP
FROM patient p JOIN medhistory h
ON p.paatientID=h.patientID;;
USE `bitein`;
CREATE OR REPLACE VIEW `PatientDetails` AS
SELECT P.PatientID, FirstName, LastName, Gender, DOB, Contactno,
Age, Gender, Weight, Height, BMI, BP
FROM patient p JOIN medhistory h
ON p.patientID=h.patientID;

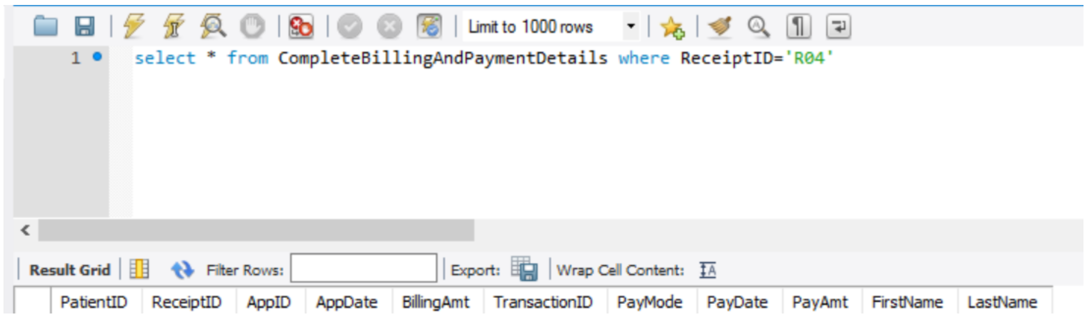


```
select * from PatientDetails where PatientID='P02'
```

PatientID	FirstName	LastName	Gender	DOB	Contactno	Age	Weight	Height	BMI	BP
P02	Alden	Gouin	M	1988-12-24	7067893452	28	150	70	22	100

- Patient Billing and Payment Details – For the receptionist

CREATE VIEW `CompleteBillingAndPaymentDetails` AS
SELECT b.PatientID, b.ReceiptID, AppID, AppDate, BillingAmt,
TransactionID, PayMode, PayDate, PayAmt, Balance
FirstName, LastName FROM
BillingInfo b JOIN Payment p
ON b.ReceiptID = p.ReceiptID
JOIN Patient PT
ON pt.PatientID = b.patientID;

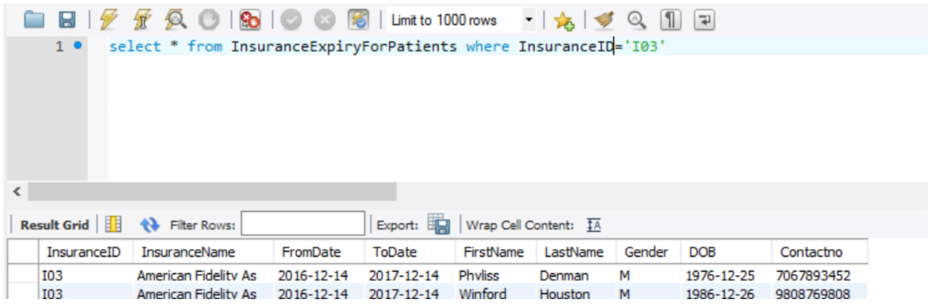


```
select * from CompleteBillingAndPaymentDetails where ReceiptID='R04'
```

PatientID	ReceiptID	AppID	AppDate	BillingAmt	TransactionID	PayMode	PayDate	PayAmt	FirstName	LastName

- Insurance Expiry Details for Patients

  CREATE VIEW `InsuranceExpiryForPatients` AS
  select i.InsuranceID, InsuranceName, FromDate, ToDate, FirstName, LastName, Gender, DOB, Contactno
  from Insurance i join Patient p
  on i.InsuranceID = p.InsuranceID;



```
select * from InsuranceExpiryForPatients where InsuranceID='I03'
```

InsuranceID	InsuranceName	FromDate	ToDate	FirstName	LastName	Gender	DOB	Contactno
I03	American Fidelitv As	2016-12-14	2017-12-14	Phvliss	Denman	M	1976-12-25	7067893452
I03	American Fidelitv As	2016-12-14	2017-12-14	Winford	Houston	M	1986-12-26	9808769808

Indexes:

All foreign keys and primary keys are indexed to help speed up the retrieval of data from tables.

Audit Trail using Triggers:

Audit Trail is a way of tracking nay changes into the database. Frequently edited items like

- **VISITING DETAILS:**

  Creating a audit table for VisitingDetails Table:
  create table Visitaudit_log
  (auditid int NOT NULL Auto_Increment primary key ,
  PatientID varchar(20), DentistID varchar(20),
  VisitID varchar(20) , SymptomID varchar(20),
  modifiedby varchar(40),
  Ts DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP);

<u>This trigger will insert values into audit table after delete operation</u>

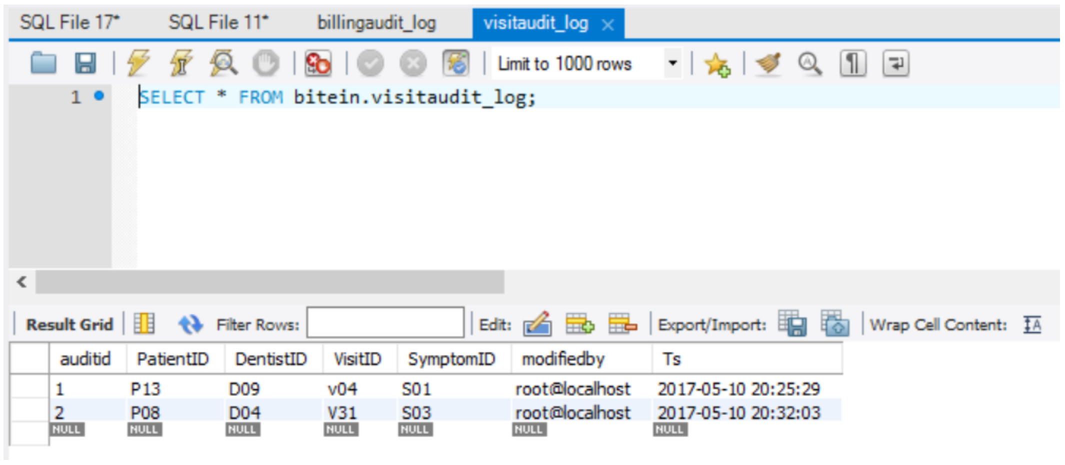
Delimiter $
create trigger Visit_Delete
After Delete on visitingdetails
for each row
begin
insert into Visitaudit_log
Values (auditid,Old.PatientID,Old.DentistID,Old.VisitID,Old.SymptomID, current_user(), current_timestamp());
end;
$

<u>This trigger will insert values into audit table after update operation</u>

Delimiter $
create trigger Visit_Update
After update on visitingdetails
for each row
begin
insert into Visitaudit_log
Values (auditid,Old.PatientID,Old.DentistID,Old.VisitID,Old.SymptomID, current_user(), current_timestamp());
end;
$

<u>This trigger will insert values into audit table before insert operation</u>

Delimiter $
create trigger Visit_Insert
After insert on visitingdetails
for each row
begin
insert into Visitaudit_log
Values (auditid,new.PatientID,new.DentistID,new.VisitID,new.SymptomID, current_user(), current_timestamp());
end;
$



```
SELECT * FROM bitein.visitaudit_log;
```

auditid	PatientID	DentistID	VisitID	SymptomID	modifiedby	Ts
1	P13	D09	v04	S01	root@localhost	2017-05-10 20:25:29
2	P08	D04	V31	S03	root@localhost	2017-05-10 20:32:03
NULL	NULL	NULL	NULL	NULL	NULL	NULL

BILLINGINFO:

<u>Creating a audit table</u>

(auditid int NOT NULL Auto_Increment primary key ,
ReceiptID varchar(20),
AppID varchar(20) ,
patientID varchar(20) ,

```
AppDate datetime ,
BillingAmt decimal(10,0) ,
RID varchar(20),
modifiedby varchar(40),
Ts DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP);
```

This trigger will insert values into audit table after delete operation

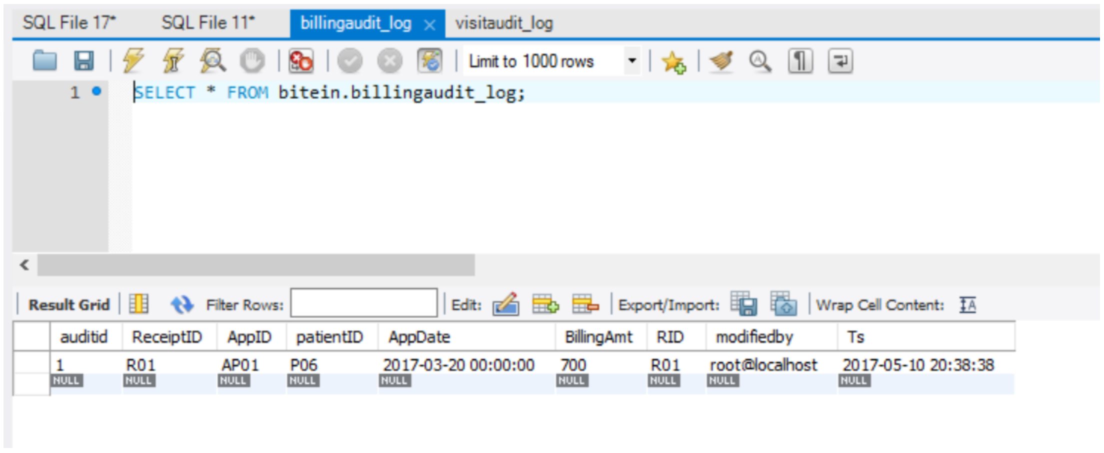
```
Delimiter $
create trigger Billing_Delete
After Delete on billinginfo
for each row
begin
insert into billingaudit_log
Values (auditid,Old.ReceiptID,Old.AppID,Old.PatientID,Old.AppDate,Old.BillingAmt,Old.RID, current_user(),
current_timestamp());
end;
$
```

This trigger will insert values into audit table after update operation

```
Delimiter $
create trigger Billing_Update
After update on billinginfo
for each row
begin
insert into billingaudit_log
Values (auditid,Old.ReceiptID,Old.AppID,Old.PatientID,Old.AppDate,Old.BillingAmt,Old.RID, current_user(),
current_timestamp());
end;
$
```

This trigger will insert values into audit table before insert operation

```
Delimiter $
create trigger Billing_Insert
After insert on billinginfo
for each row
begin
insert into billingaudit_log
Values (auditid,new.ReceiptID,new.AppID,new.PatientID,new.AppDate,new.BillingAmt,new.RID, current_user(),
current_timestamp());
end;
$
```


SQL File 17* SQL File 11* billingaudit_log visitaudit_log

```
SELECT * FROM bitein.billingaudit_log;
```

auditid	ReceiptID	AppID	patientID	AppDate	BillingAmt	RID	modifiedby	Ts
1	R01	AP01	P06	2017-03-20 00:00:00	700	R01	root@localhost	2017-05-10 20:38:38
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

PRESCRIPTION:

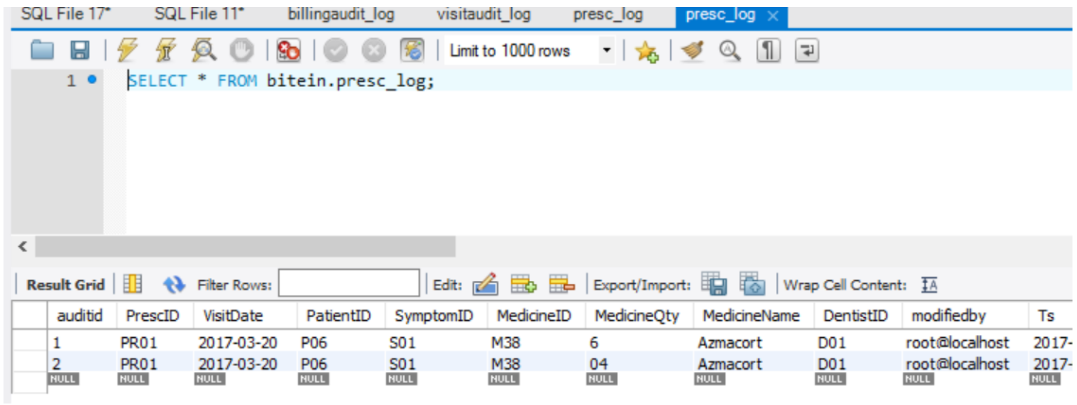
Creating audit table
create table Presc_log
(auditid int NOT NULL Auto_Increment primary key ,
PrescID varchar(20),
VisitDate date ,
PatientID varchar(20) ,
SymptomID varchar(20) ,
MedicineID varchar(20) ,MedicineQty varchar(20) ,
MedicineName varchar(40) ,
DentistID varchar(20),
modifiedby varchar(40),
Ts DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP);

This trigger will insert values into audit table after delete operation
Delimiter $
create trigger Presc_Delete
After Delete on prescription
for each row
begin
insert into presc_log
Values (auditid,Old.PrescID,Old.VisitDate,Old.PatientID,Old.SymptomID,Old.MedicineID,Old.MedicineQty, Old.MedicineName, Old.DentistID, current_user(), current_timestamp());
end;
$

This trigger will insert values into audit table after update operation
Delimiter $
create trigger Presc_Update
After update on prescription
for each row
begin
insert into presc_log
Values (auditid,Old.PrescID,Old.VisitDate,Old.PatientID,Old.SymptomID,Old.MedicineID,Old.MedicineQty, Old.MedicineName, Old.DentistID, current_user(), current_timestamp());
end;
$

<u>This trigger will insert values into audit table before insert operation</u>

```
Delimiter $
create trigger Presc_Insert
After insert on prescription
for each row
begin
insert into presc_log
Values
(auditid,new.PrescID,new.VisitDate,new.PatientID,new.SymptomID,new.MedicineID,new.MedicineQty,new.Medici
neName, new.DentistID, current_user(), current_timestamp());
end;
$
```



```
SELECT * FROM bitein.presc_log;
```

	auditid	PrescID	VisitDate	PatientID	SymptomID	MedicineID	MedicineQty	MedicineName	DentistID	modifiedby	Ts
	1	PR01	2017-03-20	P06	S01	M38	6	Azmacort	D01	root@localhost	2017-
	2	PR01	2017-03-20	P06	S01	M38	04	Azmacort	D01	root@localhost	2017-
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

APPOINTMENTDETAILS:

<u>Creating Audit table:</u>

```
create table Appt_log
(auditid int NOT NULL Auto_Increment  primary key ,
AppID varchar(20),
patientID varchar(20) ,
DentistID varchar(20) ,
Appdate datetime ,
RID varchar(20),
Modifiedby varchar(40),
Ts DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP);
```

<u>This trigger will insert values into audit table after delete operation</u>

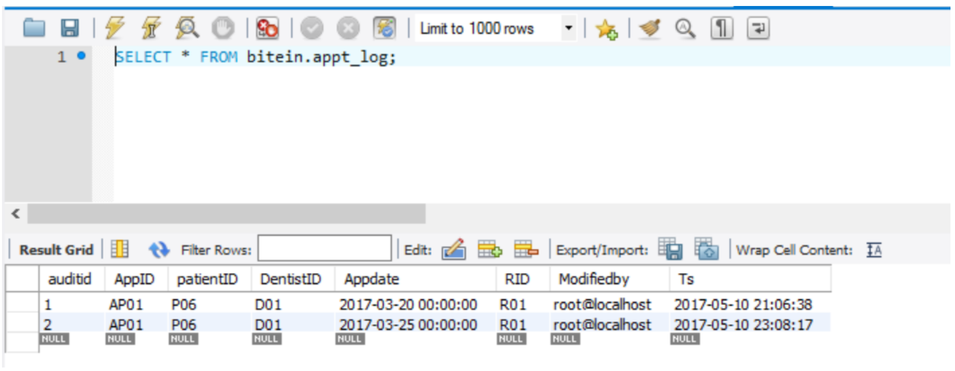
```
Delimiter $
create trigger Appt_Delete
After Delete on appointmentdetails
for each row
begin
insert into Appt_log
Values (auditid,Old.AppID,Old.PatientID,Old.DentistID, Old.AppDate,Old.RID, current_user(),
current_timestamp());
end;
$
```

__This trigger will insert values into audit table after update operation__

```
Delimiter $
create trigger Appt_Update
After update on appointmentdetails
for each row
begin
insert into appt_log
Values (auditid,Old.AppID,Old.PatientID,Old.DentistID, Old.AppDate,Old.RID, current_user(),
current_timestamp());
end;
$
```

__This trigger will insert values into audit table before insert operation__

```
Delimiter $
create trigger Appt_Insert
After insert on appointmentdetails
for each row
begin
insert into appt_log
Values (auditid,new.AppID,new.PatientID,new.DentistID, new.AppDate,new.RID, current_user(),
current_timestamp());
end;
$
```



```
SELECT * FROM bitein.appt_log;
```

auditid	AppID	patientID	DentistID	Appdate	RID	Modifiedby	Ts
1	AP01	P06	D01	2017-03-20 00:00:00	R01	root@localhost	2017-05-10 21:06:38
2	AP01	P06	D01	2017-03-25 00:00:00	R01	root@localhost	2017-05-10 23:08:17
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

PAYMENT:

Creating audit table:

```
create table Pay_log
(auditid int NOT NULL Auto_Increment  primary key ,
TransactionID varchar(20),
PayMode varchar(20) ,
Paydate date ,
PayAmt decimal(10,0),
Balance decimal(10,0),
ReceiptID varchar(20),
modifiedby varchar(40),
Ts DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP);
```

This trigger will insert values into audit table after delete operation

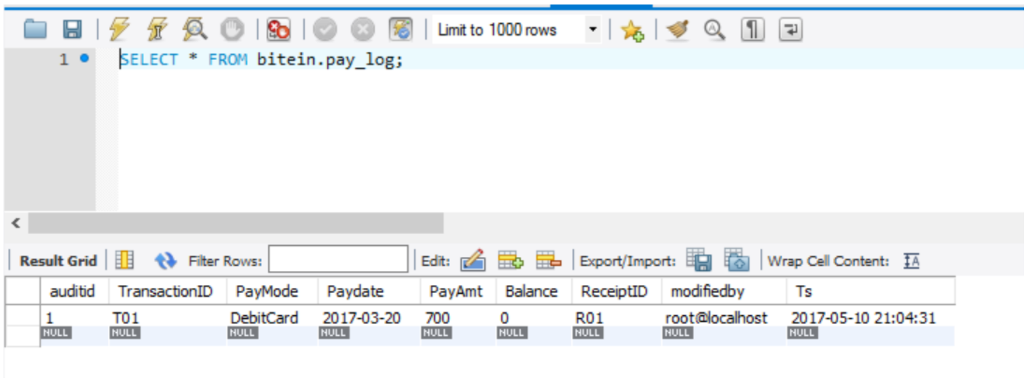
```
Delimiter $
create trigger Pay_Delete
After Delete on payment
for each row
begin
insert into pay_log
Values (auditid,Old.TransactionID,Old.PayMode,Old.Paydate, Old.PayAmt,Old.Balance, Old.ReceiptID,
current_user(), current_timestamp());
end;
$
```

This trigger will insert values into audit table after update operation

```
Delimiter $
create trigger Pay_Update
After update on payment
for each row
begin
insert into pay_log
Values (auditid,Old.TransactionID,Old.PayMode,Old.Paydate, Old.PayAmt,Old.Balance, Old.ReceiptID,
current_user(), current_timestamp());
end;
$
```

This trigger will insert values into audit table before insert operation

```
Delimiter $
create trigger Pay_Insert
After insert on payment
for each row
begin
insert into pay_log
Values (auditid,new.TransactionID,new.PayMode,new.Paydate, new.PayAmt,new.Balance, new.ReceiptID,
current_user(), current_timestamp());
end;
$
```



```
SELECT * FROM bitein.pay_log;
```

auditid	TransactionID	PayMode	Paydate	PayAmt	Balance	ReceiptID	modifiedby	Ts
1	T01	DebitCard	2017-03-20	700	0	R01	root@localhost	2017-05-10 21:04:31
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Conclusion:

Additional features were developed in the database system - stored procedures, triggers, indexes, views and user authentication and role based authentication - were implemented for the database.
This facilitated more robust and efficient application development and to support security, privacy, audit trail and other requirements of the database created for BiteIn Clinic.