[**BUAN 6320.504 - Database Foundations for Business Analytics - F23**](https://elearning.utdallas.edu/webapps/blackboard/execute/courseMain?course_id=_330166_1)

**Project - Output Report**

**Group #11**

**Hospital Database Management System**

Group Members:

1. Aashay Bhujbal -asb220014
2. Mani Saxena -mxs230066
3. Harsh Kumar Lal Das -hxl220024
4. Karthik Reddy Vuyyuru -kxv230006

**Data Definition Language**

1. Creating the Doctor Table

**A white background with black and white clouds

Description automatically generated**

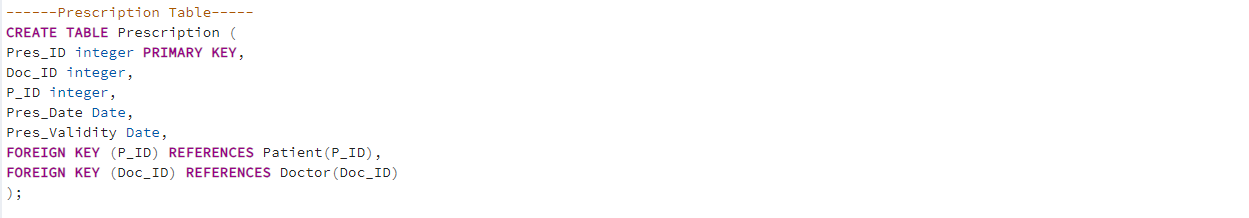
1. Creating the Patient Table



1. Creating the Appointment Table



1. Creating the Prescription Table



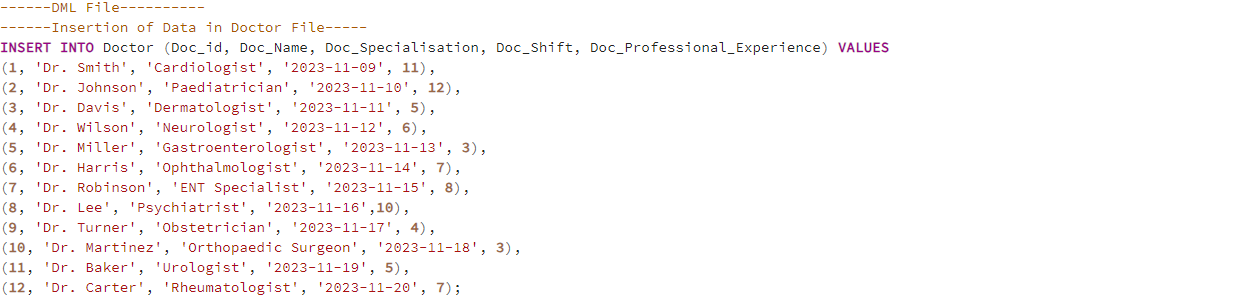
1. Creating the Bill Table



Data Manipulation Language

**DML File**

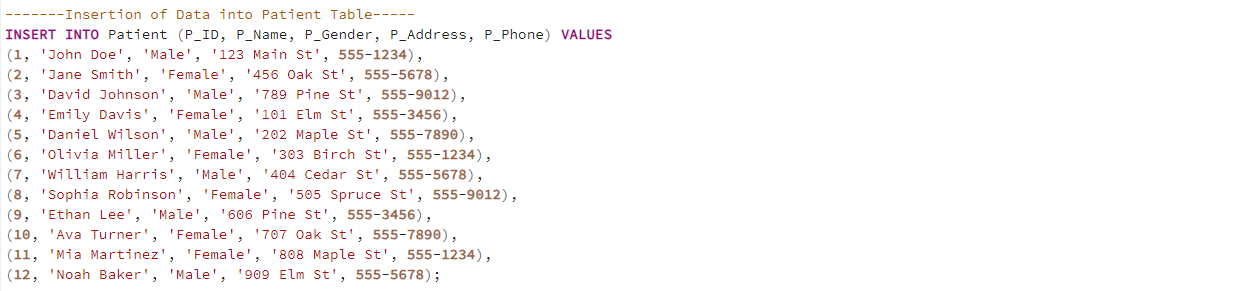
1. Inserting Data into Doctor Table

****

**Result**



1. Inserting Data into Patient Table



**Result**

A screenshot of a computer

Description automatically generated

1. Inserting Data into the Appointment Table

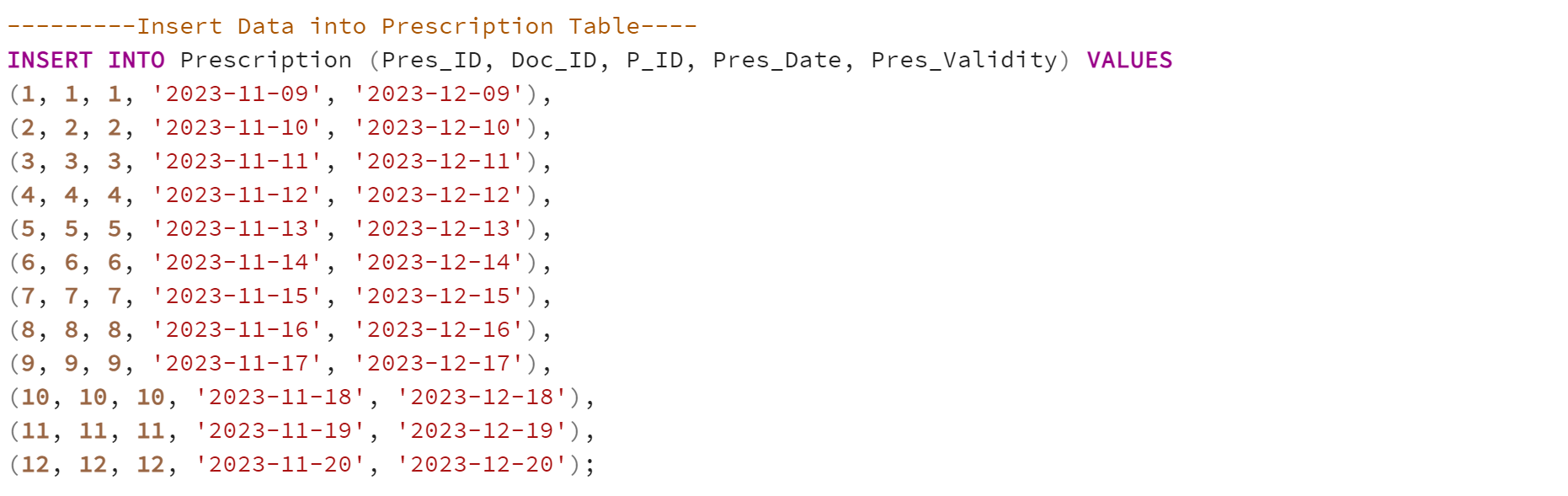
A screenshot of a computer code

Description automatically generated

**Result**

****

1. Inserting Data into the Prescription Table



**Result**

A screenshot of a computer

Description automatically generated

5. Inserting Data into the Bill Table

A screenshot of a computer

Description automatically generated

**Result**

**A screenshot of a table

Description automatically generated**

**SEQUENCES**

---------Sequence for Doctor id

CREATE SEQUENCE doc\_id\_seq

INCREMENT BY 1

START WITH 1

MINVALUE 1

NO MAXVALUE

CACHE 1;

---------Sequence for Appointment Id

CREATE SEQUENCE Apt\_ID\_seq

INCREMENT BY 1

START WITH 1

MINVALUE 1

NO MAXVALUE

CACHE 1;

------Sequence for Patient Id

CREATE SEQUENCE P\_ID\_seq

INCREMENT BY 1

START WITH 1

MINVALUE 1

NO MAXVALUE

CACHE 1;

--------Sequence for Bill Id

CREATE SEQUENCE Bill\_Id\_seq

INCREMENT BY 1

START WITH 1

MINVALUE 1

NO MAXVALUE

CACHE 1;

--------Sequence for Prescription Id

CREATE SEQUENCE Pres\_Id\_seq

INCREMENT BY 1

START WITH 1

MINVALUE 1

NO MAXVALUE

CACHE 1;

**Triggers**

**-- Create a trigger to automatically set doc\_id before insert**

**CREATE OR REPLACE FUNCTION set\_doc\_id()**

**RETURNS TRIGGER AS $$**

**BEGIN**

**- If doc\_id is not provided, set it to the next value from the sequence**

**IF NEW.doc\_id IS NULL THEN**

**NEW.doc\_id := nextval('doc\_id\_seq');**

**END IF;**

**RETURN NEW;**

**END;**

**-- Attach the trigger to the doctors table**

**CREATE TRIGGER set\_doc\_id\_trigger**

**BEFORE INSERT ON doctors**

**FOR EACH ROW**

**EXECUTE FUNCTION set\_doc\_id();**

**12 Basic Queries: 2 Advanced Queries**

**Query 1**

1. SELECT \* FROM Doctor;

**A screenshot of a computer

Description automatically generated**

1. SELECT \* FROM Patient;

**A screenshot of a computer

Description automatically generated**

1. SELECT \* FROM Appointment;

A screenshot of a computer

Description automatically generated  
iv) SELECT \* FROM Prescription.

A screenshot of a computer

Description automatically generated

1. SELECT \* FROM Bill;

A screenshot of a table

Description automatically generated

**Query 2**

SELECT

Doc\_id,

Doc\_Name,

Doc\_Specialisation,

Doc\_Shift,

Doc\_Professional\_Experience

FROM Doctor;

**A screenshot of a computer

Description automatically generated**

SELECT

p\_id,

p\_name,

p\_gender,

p\_address,

p\_phon

FROM Patient;

A screenshot of a computer

Description automatically generated

SELECT

Apt\_ID,

Doc\_id,

P\_ID,

Apt\_Date,

Apt\_Time,

Apt\_Type

FROM Appointment;

A screenshot of a computer

Description automatically generated

SELECT

Pres\_ID,

Doc\_ID,

P\_ID,

Pres\_Date,

Pres\_Validity

FROM Prescription;

A screenshot of a computer

Description automatically generated

v)

SELECT

Bill\_Id,

P\_ID,

Bill\_Date,

Bill\_Amt,

Bill\_Status

FROM Bill;

A screenshot of a computer

Description automatically generated

**Query 3**

CREATE VIEW HealthcareView AS

Select A.Apt\_ID,

A.Doc\_id as app\_Doc\_ID,

A.P\_ID as app\_P\_ID,

A.Apt\_Date,

A.Apt\_Time,

A.Apt\_Type,

A.Apt\_Room ,

D.Doc\_id,

D.Doc\_Name,

D.Doc\_Specialisation,

D.Doc\_Shift,

D.Doc\_Professional\_Experience,

P.P\_ID as patient\_ID,

P.P\_Name,

P.P\_Gender,

P.P\_Address,

P.P\_Phone ,

PR.Pres\_ID,

PR.Doc\_ID as Prescription\_DOC\_ID,

PR.P\_ID as prescription\_patient\_id,

PR.Pres\_Date,

PR.Pres\_Validity,

B.Bill\_Id,

B.P\_ID AS bill\_Patient\_id,

B.Bill\_Date,

B.Bill\_Amt,

B.Bill\_Status

From Appointment A

JOIN Doctor D on D.doc\_id = A.doc\_id

JOIN Patient P on P.P\_ID = A.P\_ID

JOIN Prescription PR on (PR.doc\_id = A.doc\_id and PR.p\_id = A.doc\_id and PR. Pres\_Date = A. Apt\_Date)

JOIN bill B on (B. P\_ID = A.P\_ID AND B.Bill\_Date = A. Apt\_Date);

SELECT \* FROM HealthcareView;

**Result**

**A screenshot of a test

Description automatically generated**

**Query 4**

SELECT \*

FROM Bill B

Join

Patient P

ON

P.P\_ID = B.P\_ID;

**Result**

**A screenshot of a computer

Description automatically generated**

**Query 5**

**SELECT**

**Doc\_id,**

**Doc\_Name,**

**Doc\_Specialisation,**

**Doc\_Shift,**

**Doc\_Professional\_Experience**

**FROM Doctor**

**ORDER BY Doc\_Name ASC;**

**Result**

**A screenshot of a computer

Description automatically generated**

**Query 6**

Select

A.Apt\_ID,

D.Doc\_id,

P.P\_ID,

A.Apt\_Date,

A.Apt\_Time

FROM Appointment A

Join doctor D on D.DOC\_id = A.doc\_id

Join Patient P on P.P\_ID = A.P\_ID

LIMIT 10;

A screenshot of a computer

Description automatically generated

**Query 7**

SELECT DISTINCT \*

FROM Prescription P

JOIN Doctor D ON D.Doc\_ID = P.Doc\_ID

Join Patient PA on PA.P\_ID = P.P\_ID

**Result**

**A screenshot of a computer

Description automatically generated**

**Query 8**

SELECT

D.Doc\_ID,

D.Doc\_Name,

Count(D.Doc\_ID)

FROM Doctor D

INNER JOIN Patient P ON P.P\_ID = D.Doc\_ID

GROUP BY D.Doc\_ID, D.Doc\_Name

HAVING D.Doc\_ID>0

ORDER by D.Doc\_ID;

A screenshot of a computer

Description automatically generated

**Query 9**

Select \*

From Doctor

Where Doc\_ID IN (1,3,5,7,9,11);

**Result**

**A screenshot of a computer

Description automatically generated**

**Query 10**

Select LENGTH(Doc\_ID)

From Doctor;

**Result**

A screenshot of a computer

Description automatically generated

**Query 11**

Select \* from bill ;

BEGIN TRANSACTION;

Delete from bill where bill\_ID = 1;

Rollback;

Select \* from bill;

------Initial Table

A screenshot of a table

Description automatically generated

----Table after deletion

A table with numbers and a pen

Description automatically generated

-----Table after rollback

A screenshot of a table

Description automatically generated

**Query 12**

SELECT \* FROM doctor;

-- Start a transaction

BEGIN TRANSACTION;

-- Update the row with Doc\_id = 1

UPDATE doctor

SET Doc\_Name = 'Dr. Avnish',

Doc\_Specialisation = 'Ophthalmologist',

Doc\_Shift = '2023-11-03', -- Assuming 'Doc\_Shift' is a date column

Doc\_Professional\_Experience = 3

WHERE Doc\_id = 1;

-- Display the state of the "doctor" table after the update

SELECT \* FROM doctor;

-- Rollback the changes made during the transaction

ROLLBACK;

**Result**

UPDATED TABLE

**A screenshot of a computer

Description automatically generated**

**ADVANCE QUERIES**

**Query 1**

**-----Defining a CTE by the name Patient\_List**

WITH Patient\_list as

( SELECT

DISTINCT P.P\_ID,

P.P\_Name,

P.P\_Gender,

P.P\_Address,

P.P\_Phone

FROM

appointment A

JOIN

patient P

ON

P.P\_ID = A.P\_ID

JOIN

doctor D

ON

D.Doc\_ID = A.Doc\_ID

WHERE Doc\_name LIKE 'Dr. M%')

---List of patient's having appointments with Doctor having first name starting with M

SELECT

DISTINCT

b.P\_ID,

PL.P\_Name,

PL.P\_Gender,

PL.P\_Address,

PL.P\_Phone,

SUM( CASE WHEN b.bill\_status = 'Paid'

THEN

b.Bill\_Amt else 0 end ) AS total\_paid\_amount,

SUM(CASE WHEN b.bill\_status = 'Unpaid' then b.Bill\_Amt else 0 end)

AS total\_unpaid\_amount

FROM bill b

JOIN Patient\_list PL ON PL.P\_ID = B.P\_ID

GROUP BY

b.P\_ID,

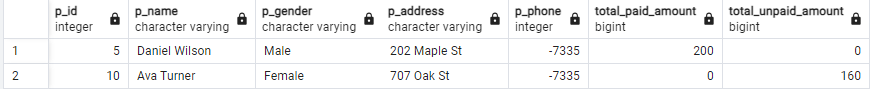
PL.P\_Name,

PL.P\_Gender,

PL.P\_Address,

PL.P\_Phone;

**Result:**



**----Query 2:** Details of doctors who have appointments with patients with maximum and minimum bills.

WITH patient\_total\_bill as

(SELECT

DISTINCT P\_ID,

SUM(bill\_amount) as total\_bill ,

FROM bill

Group by P\_ID ),

max\_min\_bill as

(SELECT

MAX(total\_bill) as max\_bill,

MIN(total\_bill) as min\_bill

FROM patient\_total\_bill),

Patient\_list AS

(SELECT

DISTINCT P\_ID

FROM patient\_total\_\_bill

WHERE (total\_bill = (select max\_bill from\_max\_min\_bill)

OR

(total\_bill = (select min\_bill from\_max\_min\_bill)))

SELECT

DISTINCT D.Doc\_id,

D.Doc\_Name,

D.Doc\_Specialisation,

D.Doc\_Shift,

D.Doc\_Professional\_Experience

FROM Appointment A

JOIN Doctor D on D.doc\_id = A.doc\_id

WHERE A.P\_ID in (select distinct P\_ID from patient\_list);

**A screenshot of a phone

Description automatically generated**