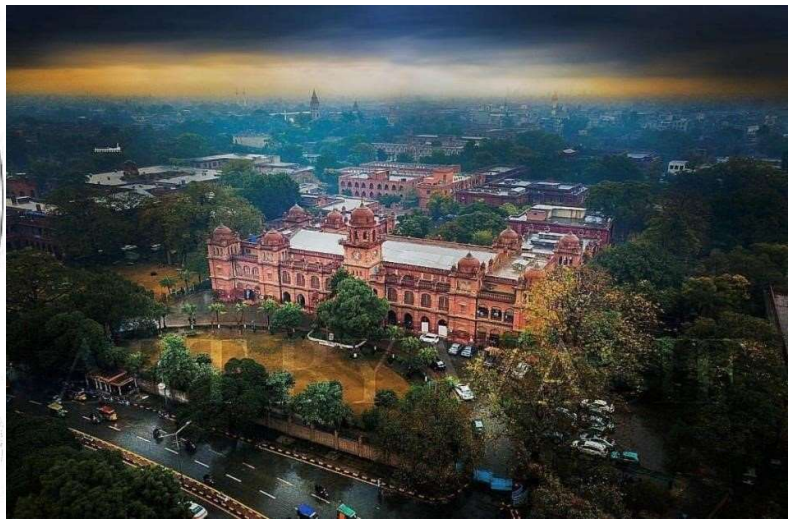

UNIVERSITY OF THE PUNJAB

DEPARTMENT OF IT

DATABASE SYSTEMS -COURSE PROJECT:

CLINICAL MANAGEMENT SYSTEM

SUBMITTED TO:

PROFESSOR DR. ASIF SOHAIL

SUBMITTED BY:

BITF21M009-AMINA BIBI

BITF21M019-EISHA MEHMOOD

BITF21M028-MARYAM KIRAN

QUESTION#01:

Introduction to the working of the system

The Clinical Management System is a comprehensive solution designed to streamline and optimize various aspects of a clinical facility's operations. This system encompasses a relational database structure that efficiently manages data related to clinical staff, sanitation teams, security personnel, doctors, patients, appointments, treatments, pharmaceuticals, and billing.

Key Components:

1. **Clinical Staff Management:**
    - The system maintains crucial information about clinical staff, including their identification details, contact information, designation, and address.
    - Staff members are uniquely identified by their StaffID, which serves as the primary key in the Clinical_Staff table.
2. **Sanitation Team and Security Personnel:**
    - Sanitation teams and security personnel are organized in separate tables, each linked to the Clinical_Staff table through foreign key relationships.
    - The Sanitation_Team table contains information about Care Certificate numbers, access levels, and salaries.
    - The Security_Personnel table includes details such as security license numbers, first aid and CPR certification, and salaries.
3. **Doctor Information:**
    - Doctors are managed in the Doctors table, linked to Clinical_Staff through foreign key relationships.
    - Information includes medical license numbers, years of experience, and availability hours.
4. **Patient Records:**
    - Patient data is stored in the Patient table, containing essential information like patient ID, name, date of birth, height, weight, and blood pressure.
    - PatientID serves as the primary key in the Patient table.
5. **Appointment Scheduling:**
    - Appointments between patients and doctors are facilitated through the Appointments table.
    - Each appointment has a unique AppointmentID, capturing details such as patient ID, staff ID (doctor), appointment date, time, and status.
6. **Treatment and Medication Management:**

- The system manages treatments in the Treatment table, detailing treatment names, durations, and a unique TreatmentID.
- Inventory_Item and Pharmacy tables store information about pharmaceutical items, including item names, purchase dates, expiry dates, and quantities.

7. **Billing and Payments:**
    - Billing information is stored in the Billing table, connecting patients, appointments, billing methods, payable amounts, and billing dates.

Objective: The primary objective of the Healthcare Management System is to enhance the efficiency of healthcare facility operations by providing a centralized platform for managing staff, appointments, patient records, treatments, medications, and billing. The system aims to streamline workflows, improve data accuracy, and ensure seamless communication between different components, ultimately contributing to the overall quality of patient care.

QUESTION#04:

Construction of the Relational Schema by using both bottom-up approach and top-down approach.

Bottom-UP Approach:

R1(PatientID,PName,DOB,Height,Weight,BP,StaffID,Name,Designation,ContactInfo,Address,SPStaffID,CareCertificateNumber,AccessLevel,Salary,STStaffID,SecurityLicenseNumber,FirstAidAndCPRCertification,SPSalary,DStaffID,MedicalLicenseNumber,YearsOfExperience,AvailabilityStartHour,AvailabilityEndHour,**IssueID**, **Current_Issue**,BillingID,BillingMethod,PayableAmount,BDate,AppointmentID,AppointmentDate, AppointmentTime,Status , TreatmentID, TName,Duration,Notes ,Dosage,UsageNote, ItemID, ItemName, PurchasedDate, ExpiryDate, StorageLocation, Quantity)

1NF: No Multivalue

R1(PatientID, **IssueID**,**Current_Issue**)

R2(PatientID,PName,DOB,Height,Weight,BP,StaffID,Name,Designation,ContactInfo,Address,SPStaffID,CareCertificateNumber,AccessLevel,Salary,STStaffID,SecurityLicenseNumber,FirstAidAndCPRCertification,SPSalary,DStaffID,MedicalLicenseNumber,YearsOfExperience,AvailabilityStartHour,AvailabilityEndHour, BillingID,BillingMethod,PayableAmount,BDate,AppointmentID,AppointmentDate, AppointmentTime,Status,TreatmentID,TName,Duration,Notes ,Dosage,UsageNote, ItemID, ItemName, PurchasedDate, ExpiryDate, StorageLocation, Quantity)

2NF: No PFD

R1(PatientID, **IssueID**,**Current_Issue**)

R2(PatientID,PName,DOB,Height,Weight,BP,StaffID,Name,Designation,ContactInfo,Address,STStaffID,CareCertificateNumber,AccessLevel,Salary,SPStaffID,SecurityLicenseNumber,FirstAidAndCPRCertification,SPSalary,DStaffID,MedicalLicenseNumber,YearsOfExperience,AvailabilityStartHour,AvailabilityEndHour, BillingID,BillingMethod,PayableAmount,BDate,AppointmentID,AppointmentDate, AppointmentTime,Status,TreatmentID,TName,Duration,Notes ,Dosage,UsageNote, ItemID, ItemName, PurchasedDate, ExpiryDate, StorageLocation, Quantity)

3NF: No NK->NK

R1(PatientID, **IssueID**,**Current_Issue**)

R2(PatientID,PName,DOB,Height,Weight,BP)

R3(StaffID,Name,Designation,ContactInfo,Address)

R4(STStaffID,CareCertificateNumber,AccessLevel,Salary)

R5(SPStaffID,SecurityLicenseNumber,FirstAidAndCPRCertification,SPSalary)

R6(DStaffID,MedicalLicenseNumber,YearsOfExperience,AvailabilityStartHour,AvailabilityEndHour)

R7(PatientID, AppointmentID,BillingID,BillingMethod,PayableAmount,BDate)

R8(AppointmentID,DStaffID,PatientID,AppointmentDate,AppointmentTime,

Status)

R9(TreatmentID, TName,Duration)

R10(ItemID, ItemName, PurchasedDate, ExpiryDate, StorageLocation, Quantity)

R11(TreatmentID, ItemID, UsageNote)

R12(AppointmentID, TreatmentID, Notes, Dosage)

Top-Down Approach:

These relationship are already in 1NF ,2NF and 3Nf.

R1(PatientID, **IssueID**,**Current_Issue**)

R2(PatientID,PName,DOB,Height,Weight,BP)

R3(StaffID,Name,Designation,ContactInfo,Address)

R4(STStaffID,CareCertificateNumber,AccessLevel,Salary)

R5(SPStaffID,SecurityLicenseNumber,FirstAidAndCPRCertification,SPSalary)

R6(DStaffID,MedicalLicenseNumber,YearsOfExperience,AvailabilityStartHour,AvailabilityEndHour)

R7(PatientID, AppointmentID,BillingID,BillingMethod,PayableAmount,BDate)

R8(AppointmentID,DStaffID,PatientID,AppointmentDate,AppointmentTime,

Status)

R9(TreatmentID, TName,Duration)

R10(ItemID, ItemName, PurchasedDate, ExpiryDate, StorageLocation, Quantity)

R11(TreatmentID, ItemID, UsageNote)

R12(AppointmentID, TreatmentID, Notes, Dosage)

QUESTION#05:

Description of the relations.

Table Name: **Clinical_Staff**

Attribute	Data Type	Size	Constraints
StaffID	NUMBER	4	PRIMARY KEY
Name	VARCHAR2	15	NOT NULL
Designation	VARCHAR2	25	NOT NULL
ContactInfo	NUMBER	15	NOT NULL
Address	VARCHAR2	35	NOT NULL

Table Name: **Sanitation_Team**

Attribute	Data Type	Size	Constraints
StaffID	NUMBER	4	PRIMARY KEY FOREIGN KEY references to Clinical_Staff
CareCertificateNumber	VARCHAR2	7	NOT NULL, UNIQUE
AccessLevel	VARCHAR2	25	
Salary	NUMBER	(8,2)	NOT NULL

Table Name: **Security_Personnel**

Attribute	Data Type	Size	Constraints

StaffID	NUMBER	4	PRIMARY KEY FOREIGN KEY references to Clinical_Staff
SecurityLicenseNumber	VARCHAR2	15	NOT NULL, UNIQUE
FirstAidAndCPRCertification	VARCHAR2	5	
Salary	NUMBER	(8,2)	NOT NULL

Table Name: **Doctors**

Attribute	Data Type	Size	Constraints
StaffID	NUMBER	4	PRIMARY KEY FOREIGN KEY references to Clinical_Staff
MedicalLicenseNumber	VARCHAR2	15	NOT NULL, UNIQUE
YearsOfExperience	NUMBER		DEFAULT 0, NOT NULL
AvailabilityStartHour	VARCHAR2	5	
AvailabilityEndHour	VARCHAR2	5	

Table Name: **Patient**

Attribute	Data Type	Size	Constraints
PatientID	NUMBER		PRIMARY KEY
Name	VARCHAR2	25	NOT NULL
DOB	DATE		DEFAULT SYSDATE , NOT NULL
Height	NUMBER		
Weight	NUMBER		
BP	VARCHAR2	7	

Table Name: **Patient_Issue**

Attribute	Data Type	Size	Constraints
PatientID	NUMBER		PRIMARY KEY, FOREIGN KEY references to Patient

IssueID	NUMBER		PRIMARY KEY
Current_Issue	VARCHAR2	255	NOT NULL

Table Name: **Appointments**

Attribute	Data Type	Size	Constraints
AppointmentID	NUMBER		PRIMARY KEY
PatientID	NUMBER		FOREIGN KEY references to Patient
StaffID	NUMBER		UNIQUE, FOREIGN KEY references to Doctors
AppointmentDate	DATE		DEFAULT SYSDATE, NOT NULL, UNIQUE
AppointmentTime	VARCHAR2	8	NOT NULL, CHECK (AppointmentTime BETWEEN '00.00.00' AND '23.59.59'), UNIQUE
Status	VARCHAR2	10	DEFAULT 'PENDING', NOT NULL

Table Name: **Treatment**

Attributes	Data Type	Size	Constraints
TreatmentID	NUMBER		PRIMARY KEY

Name	VARCHAR2	25	UNIQUE, NOT NULL
Duration	VARCHAR2	15	DEFAULT NULL

Table Name: **Appointment_Treatment**

Attribute	Data Type	Size	Constraints
TreatmentID	NUMBER		PRIMARY KEY, FOREIGN KEY references to Treatment
AppointmentID	NUMBER		PRIMARY KEY, FOREIGN KEY references to Appointments
Notes	VARCHAR2	255	DEFAULT 'NOT WORKING', NOT NULL
Dosage	VARCHAR2	25	NOT NULL

Table Name: **Pharmacy**

Attribute	Data Type	Size	Constraints
ItemID	NUMBER		PRIMARY KEY, NOT NULL
ItemName	VARCHAR2	25	
PurchasedDate	Date		DEFAULT SYSDATE, NOT NULL
ExpiryDate	DATE		DEFAULT SYSDATE + 7 months
StorageLocation	VARCHAR2	25	NOT NULL

Quantity	NUMBER		DEFAULT '0' NOT NULL CHECK (Quantity >= 0)

Table Name: **Inventory_Item**

Attribute	Data Type	Size	Constraints
TreatmentID	NUMBER		PRIMARY KEY, FOREIGN KEY references to Treatment
ItemID	NUMBER		PRIMARY KEY, FOREIGN KEY references to Pharmacy
UsageNotes	VARCHAR2	255	DEFAULT 'used only after consulting with your healthcare provide' NOT NULL

Table Name: **Billing**

Attribute	Data Type	Size	Constraints
BillingID	NUMBER		PRIMARY KEY
AppointmnetID	NUMBER		FOREIGN KEY references to Appointments
PatientID	NUMBER		FOREIGN KEY references to Patient, NOT NULL

BillingMethod	VARCHAR2	255	CHECK ('cash','debitcard', 'MobilePayment') , NOT NULL
PayableAmount	Decimal	10,2	CHECK greater than 0, NOT NULL
BDate	DATE		DEFAULT SYSDAYE, NOT NULL

QUESTION#06:

CREATE TABLE statements for all the relations of your system.

Clinical_Staff:

```
CREATE TABLE Clinical_Staff (
  StaffID NUMBER(4),
  Name VARCHAR2(15) NOT NULL,
  Designation VARCHAR2(25) NOT NULL,
  ContactInfo NUMBER(15) NOT NULL,
  Address  VARCHAR2(35) NOT NULL,
  CONSTRAINT ClinicalStaff_PK PRIMARY KEY (StaffID)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CLINICAL_STAFF	STAFFID	NUMBER	-	4	0	1	-	-	-
	NAME	VARCHAR2	15	-	-	-	-	-	-
	DESIGNATION	VARCHAR2	25	-	-	-	-	-	-
	CONTACTINFO	NUMBER	-	15	0	-	-	-	-
	ADDRESS	VARCHAR2	35	-	-	-	-	-	-
									1 - 5

Sanitation_Team:

```
CREATE TABLE Sanitation_Team (
  StaffID NUMBER(4),
  CareCertificateNumber VARCHAR2(7) NOT NULL UNIQUE,
  AccessLevel VARCHAR2(25),
```

```
  Salary NUMBER(5,2) NOT NULL,
  CONSTRAINT SanitationTeam_PK PRIMARY KEY (StaffID),
  CONSTRAINT SanitationTeam_FK1 FOREIGN KEY (StaffID) REFERENCES Clinical_Staff(StaffID)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SANITATION_TEAM	STAFFID	NUMBER	-	4	0	1	-	-	-
	CARECERTIFICATENUMBER	VARCHAR2	7	-	-	-	-	-	-
	ACCESSLEVEL	VARCHAR2	25	-	-	-	✓	-	-
	SALARY	NUMBER	-	8	2	-	-	-	-

1 - 4

Security_Personnel:

```
CREATE TABLE Security_Personnel (
  StaffID NUMBER(4),
  SecurityLicenseNumber VARCHAR2(15) NOT NULL UNIQUE,
  FirstAidAndCPRCertification VARCHAR2(5) Default 'No',
  Salary NUMBER(5,2) NOT NULL,
  CONSTRAINT SecurityPersonnel_PK PRIMARY KEY (StaffID),
  CONSTRAINT SecurityPersonnel_FK1 FOREIGN KEY (StaffID) REFERENCES Clinical_Staff(StaffID)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SECURITY_PERSONNEL	STAFFID	NUMBER	-	4	0	1	-	-	-
	SECURITYLICENSENUMBER	VARCHAR2	15	-	-	-	-	-	-
	FIRSTAIDANDCPRCERTIFICATION	VARCHAR2	5	-	-	-	✓	'No'	-
	SALARY	NUMBER	-	8	2	-	-	-	-

1 - 4

Doctors:

```
CREATE TABLE Doctors (
  StaffID NUMBER(4),
  MedicalLicenseNumber VARCHAR2(15) UNIQUE NOT NULL,
  YearsOfExperience NUMBER DEFAULT 0 NOT NULL,
  AvailabilityStartHour VARCHAR2(5),  -- Format: HH:MM
  AvailabilityEndHour VARCHAR2(5),    -- Format: HH:MM
  CONSTRAINT Doctors_PK PRIMARY KEY (StaffID),
  CONSTRAINT Doctors_FK1 FOREIGN KEY (StaffID) REFERENCES Clinical_Staff(StaffID)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DOCTORS	STAFFID	NUMBER	-	4	0	1	-	-	-
	MEDICALLICENSENUMBER	VARCHAR2	15	-	-	-	-	-	-
	YEARSOFEXPERIENCE	NUMBER	22	-	-	-	-	0	-
	AVAILABILITYSTARTHOUR	VARCHAR2	5	-	-	-	✓	-	-
	AVAILABILITYENDHOUR	VARCHAR2	5	-	-	-	✓	-	-

1 - 5

Patient:

```
CREATE TABLE Patient (
 PatientID NUMBER NOT NULL,
 Name VARCHAR2(25) NOT NULL,
 DOB DATE DEFAULT SYSDATE NOT NULL,
 Height NUMBER,
 WEIGHT NUMBER,
 BP VARCHAR2(7),
 CONSTRAINT Patient_PK PRIMARY KEY (PatientID)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PATIENT	PATIENTID	NUMBER	22	-	-	1	-	-	-
	NAME	VARCHAR2	25	-	-	-	-	-	-
	DOB	DATE	7	-	-	-	-	SYSDATE	-
	HEIGHT	NUMBER	22	-	-	-	✓	-	-
	WEIGHT	NUMBER	22	-	-	-	✓	-	-
	BP	VARCHAR2	7	-	-	-	✓	-	-

1 - 6

Patient_Issue:

```
CREATE TABLE Patient_Issue (
 PatientID NUMBER,
 IssueID NUMBER,
 Current_Issue VARCHAR(255) NOT NULL,
 CONSTRAINT PK_Patient_Issue PRIMARY KEY (PatientID, IssueID),
 CONSTRAINT FK_Patient_Issue_Patient FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PATIENT_ISSUE	PATIENTID	NUMBER	22	-	-	1	-	-	-
	ISSUEID	NUMBER	22	-	-	2	-	-	-
	CURRENT_ISSUE	VARCHAR2	255	-	-	-	-	-	-

1 - 3

Appointments:

```
CREATE TABLE Appointments (
  AppointmentID NUMBER,
  PatientID NUMBER NOT NULL,
  StaffID NUMBER NOT NULL,
  AppointmentDate DATE DEFAULT SYSDATE NOT NULL,
  AppointmentTime VARCHAR2(8) NOT NULL, -- Assuming format 'HH:MI:SS'
  Status VARCHAR2(10) DEFAULT 'Pending' NOT NULL,
  CONSTRAINT Appointments_PK PRIMARY KEY (AppointmentID),
  CONSTRAINT Appointments_FK1 FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
  CONSTRAINT Appointments_FK2 FOREIGN KEY (StaffID) REFERENCES Doctors(StaffID),
  CONSTRAINT Appt_Time_Check CHECK (AppointmentTime BETWEEN '00:00:00' AND '23:59:59'),
  CONSTRAINT Appt_DateTime_StaffID_UQ UNIQUE (AppointmentDate, AppointmentTime, StaffID)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
APPOINTMENTS	APPOINTMENTID	NUMBER	22	-	-	1	-	-	-
	PATIENTID	NUMBER	22	-	-	-	-	-	-
	STAFFID	NUMBER	22	-	-	-	-	-	-
	APPOINTMENTDATE	DATE	7	-	-	-	-	SYSDATE	-
	APPOINTMENTTIME	VARCHAR2	8	-	-	-	-	-	-
	STATUS	VARCHAR2	10	-	-	-	-	'Pending'	-

1 - 6

Treatment:

```
CREATE TABLE Treatment (
  TreatmentID NUMBER,
  Name VARCHAR2(25) NOT NULL UNIQUE,
  Duration VARCHAR2(15) DEFAULT NULL, --3 DAYS OR 2 MONTHS
  CONSTRAINT PK_Treatment PRIMARY KEY(TreatmentID)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TREATMENT	TREATMENTID	NUMBER	22	-	-	1	-	-	-
	NAME	VARCHAR2	25	-	-	-	-	-	-
	DURATION	VARCHAR2	15	-	-	-	✓	NULL	-

1 - 3

Appointment_Treatment:

```
CREATE TABLE Appointment_Treatment (
  TreatmentID NUMBER,
  AppointmentID NUMBER ,
  Notes VARCHAR2(255) DEFAULT 'WORKING' NOT NULL,
  Dosage VARCHAR2(25) NOT NULL,
  CONSTRAINT Appointment_Treatment_PK PRIMARY KEY (AppointmentID,TreatmentID ),
  CONSTRAINT Appointment_Treatment_FK1 FOREIGN KEY (AppointmentID) REFERENCES
Appointments(AppointmentID),
  CONSTRAINT Appointment_Treatment_FK2 FOREIGN KEY (TreatmentID) REFERENCES
Treatment(TreatmentID)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
APPOINTMENT_TREATMENT	TREATMENTID	NUMBER	22	-	-	2	-	-	-
	APPOINTMENTID	NUMBER	22	-	-	1	-	-	-
	NOTES	VARCHAR2	255	-	-	-	-	'WORKING'	-
	DOSAGE	VARCHAR2	25	-	-	-	-	-	-

1 - 4

Pharmacy :

```
CREATE TABLE Pharmacy (
  ItemID NUMBER NOT NULL,
  ItemName VARCHAR2(25),
  PurchasedDate DATE DEFAULT SYSDATE NOT NULL,
  ExpiryDate DATE DEFAULT ADD_MONTHS(SYSDATE, 7),
  StorageLocation VARCHAR2(25) NOT NULL,
  Quantity NUMBER DEFAULT 0 NOT NULL,
  CONSTRAINT PK_Pharmacy PRIMARY KEY (ItemID),
    -- Check constraint to ensure Quantity is either 0 or greater than 0
  CONSTRAINT Quantity_Check CHECK (Quantity >= 0)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PHARMACY	ITEMID	NUMBER	22	-	-	1	-	-	-
	ITEMNAME	VARCHAR2	25	-	-	-	✓	-	-
	PURCHASEDDATE	DATE	7	-	-	-	-	SYSDATE	-
	EXPIRYDATE	DATE	7	-	-	-	✓	ADD_MONTHS(SYSDATE, 7)	-
	STORAGELOCATION	VARCHAR2	25	-	-	-	-	-	-
	QUANTITY	NUMBER	22	-	-	-	-	0	-
									1 - 6

Inventory_Item:

```
CREATE TABLE Inventory_Item (

 TreatmentID NUMBER,

 ItemID NUMBER  ,

 UsageNotes VARCHAR2(255) DEFAULT 'Used only after consulting with your healthcare provide' NOT NULL,

 CONSTRAINT Inventory_Item_PK PRIMARY KEY (ItemID ,TreatmentID ),

 CONSTRAINT Inventory_Item_FK1 FOREIGN KEY (ItemID ) REFERENCES Pharmacy(ItemID),
CONSTRAINT Inventory_Item_FK2 FOREIGN KEY (TreatmentID) REFERENCES Treatment(TreatmentID)

);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
INVENTORY_ITEM	TREATMENTID	NUMBER	22	-	-	2	-	-	-
	ITEMID	NUMBER	22	-	-	1	-	-	-
	USAGENOTES	VARCHAR2	255	-	-	-	-	'Used only after consulting with your healthcare provide'	-
									1 - 3

Billing:

```
CREATE TABLE Billing (

 BillingID NUMBER,

 AppointmentID NUMBER,

 PatientID NUMBER NOT NULL,

 BillingMethod VARCHAR2(255) CHECK (BillingMethod IN ('cash', 'debitcard', 'MobilePayment')) NOT NULL,

 PayableAmount DECIMAL(10,2) DEFAULT 0 CHECK (PayableAmount >= 0) NOT NULL,

 BDate DATE DEFAULT SYSDATE NOT NULL,

 CONSTRAINT Billing_PK PRIMARY KEY (BillingID),

 CONSTRAINT Billing_FK1 FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),

 CONSTRAINT Billing_FK2 FOREIGN KEY (AppointmentID) REFERENCES Appointments(AppointmentID)

);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BILLING	BILLINGID	NUMBER	22	-	-	1	-	-	-
	APPOINTMENTID	NUMBER	22	-	-	-	✓	-	-
	PATIENTID	NUMBER	22	-	-	-	-	-	-
	BILLINGMETHOD	VARCHAR2	255	-	-	-	-	-	-
	PAYABLEAMOUNT	NUMBER	-	10	2	-	-	0	-
	BDATE	DATE	7	-	-	-	-	SYSDATE	-

1 - 6

--I create this sequence just to make my insertion efficient and consistent CREATE

SEQUENCE SEQ_BillingID START WITH 1 INCREMENT BY 1;

QUESTION#07:

Desing of at least two VIEWS, that you feel are the most important.

Let's insert some data first:

Insertion Queries:

-- Insertion queries for Clinical_Staff

INSERT INTO Clinical_Staff VALUES (1, 'Dr. John Smith', 'Doctor', 1234567890, '123 Main St');

INSERT INTO Clinical_Staff VALUES (2, 'Dr. Sarah Elif', 'Doctor', 9876543210, '456 Oak St');

INSERT INTO Clinical_Staff VALUES (3, 'Mark Davis', 'Sanitation Staff', 5551112222, '789 Pine St');

INSERT INTO Clinical_Staff VALUES (4, 'Emily White', 'Sanitation Staff', 3334445555, '101 Maple St');

INSERT INTO Clinical_Staff VALUES (5, 'Jack Thompson', 'Security Guard', 7778889999, '202 Elm St');

-- Insertion queries for Security_Personnel

INSERT INTO Security_Personnel VALUES (5, 'SLN11111', 'Yes', 32000.00);

-- Insertion queries for Sanitation_Team

INSERT INTO Sanitation_Team VALUES (3, 'CC67890', 'Medium', 20000.00);

INSERT INTO Sanitation_Team VALUES (4, 'CC11111', 'Low', 18000.00);

-- Insertion queries for Doctors

```
INSERT INTO Doctors VALUES (1, 'MLN12345', 5, '08:00', '17:00');

INSERT INTO Doctors VALUES (2, 'MLN67890', 10, '09:00', '18:00');

-- Insertion queries for Patient

INSERT INTO Patient VALUES (1, 'John Smith', TO_DATE('1990-05-15', 'YYYY-MM-DD'), 170, 70, '120/80');

INSERT INTO Patient VALUES (2, 'Alice Johnson', TO_DATE('1985-08-22', 'YYYY-MM-DD'), 160, 55, '110/70');

-- Insertion queries for Patient_Issue

INSERT INTO Patient_Issue VALUES (1, 1, 'Headache');

INSERT INTO Patient_Issue VALUES (1, 2, 'Fever');

INSERT INTO Patient_Issue VALUES (2, 3, 'Back Pain');



-- Insertion queries for Appointments

INSERT INTO Appointments VALUES (1, 1, 1, SYSDATE, '09:00:00', 'Pending');

INSERT INTO Appointments VALUES (2, 2, 2, SYSDATE, '10:30:00', 'Pending');

INSERT INTO Appointments VALUES (3, 1, 1, SYSDATE, '12:00:00', 'Pending');

INSERT INTO Appointments VALUES (4, 1, 2, SYSDATE, '11:00:00', 'Pending');

INSERT INTO Appointments VALUES (5, 1, 1, SYSDATE, '10:00:00', 'Pending');

-- Insertion queries for Treatment

INSERT INTO Treatment VALUES (1, 'Pain Relief', '3 DAYS');

INSERT INTO Treatment VALUES (2, 'Physical Therapy', '2 MONTHS');


-- Insertion queries for Appointment_Treatment

INSERT INTO Appointment_Treatment VALUES (1, 1, 'Prescribed for headache', 'As needed');

INSERT INTO Appointment_Treatment VALUES (2, 2, 'Physical therapy plan', 'Daily');

-- Insertion queries for Pharmacy

INSERT INTO Pharmacy VALUES (1, 'Painkiller', SYSDATE, ADD_MONTHS(SYSDATE, 12), 'Shelf A', 100);

INSERT INTO Pharmacy VALUES (2, 'Antibiotic', SYSDATE, ADD_MONTHS(SYSDATE, 9), 'Shelf B', 50);
```

```
INSERT INTO Pharmacy VALUES (3, 'Cough Syrup', SYSDATE,
ADD_MONTHS(SYSDATE, 6), 'Shelf C', 75);


-- Insertion queries for Inventory_Item

INSERT INTO Inventory_Item VALUES (1, 1, 'Usage note for Painkiller');

INSERT INTO Inventory_Item VALUES (1, 2, 'Usage note for Antibiotic');

INSERT INTO Inventory_Item VALUES (2, 3, 'Usage note for Cough Syrup');


-- Insertion queries for Billing
```

As billing only possible if appointment status is clear let’s update patient 1’s status

```
-- Update the status of Patient 1's appointment to 'Clear'

UPDATE Appointments

SET Status = 'Clear'

WHERE PatientID = 1;


INSERT INTO Billing VALUES (1, 1, 1, 'cash', 500.00, SYSDATE);
```

View-01:

```
CREATE VIEW PatientAppointments AS

SELECT

    A.AppointmentID,

    P.PatientID,

    P.Name AS PatientName,

    D.Name AS DoctorName,

    A.AppointmentDate,

    A.AppointmentTime,

    A.Status

FROM

    Appointments A

    JOIN Doctors D ON A.StaffID = D.StaffID

    JOIN Patient P ON A.PatientID = P.PatientID;
```

APPOINTMENTID	PATIENTID	PATIENTNAME	DOCTORNAME	APPOINTMENTDATE	APPOINTMENTTIME	STATUS
5	1	John Smith	Dr. John Smith	12/21/2023	10:00:00	Pending
3	1	John Smith	Dr. John Smith	12/17/2023	12:00:00	Pending
1	1	John Smith	Dr. John Smith	12/16/2023	09:00:00	Clear
2	2	Alice Johnson	Dr. Sarah Elif	12/16/2023	10:30:00	Clear
4	1	John Smith	Dr. Sarah Elif	12/17/2023	11:00:00	Pending

View-02:

```
CREATE VIEW BillingDetails AS
SELECT
    B.BillingID,
    P.Name AS PatientName,
    B.BillingMethod,
    B.PayableAmount,
    B.BDate
FROM
    Billing B
    JOIN Patient P ON B.PatientID = P.PatientID;
```

BILLINGID	PATIENTNAME	BILLINGMETHOD	PAYABLEAMOUNT	BDATE
1	John Smith	cash	500	12/16/2023
2	Alice Johnson	cash	50	12/16/2023

2 rows returned in 0.07 seconds

QUESTION#09:

SELECT statement for at least five common reports to be generated by the system.

1. List of Patients and Their Appointments:

```
SELECT
    P.PatientID,
    P.Name AS PatientName,
    A.AppointmentID,
    A.AppointmentDate,
    A.AppointmentTime,
    A.Status
FROM
    Patient P
JOIN
    Appointments A ON P.PatientID = A.PatientID
Order By AppointmentID ;
```

PATIENTID	PATIENTNAME	APPOINTMENTID	APPOINTMENTDATE	APPOINTMENTTIME	STATUS
1	John Smith	1	12/16/2023	09:00:00	Clear
2	Alice Johnson	2	12/16/2023	10:30:00	Clear
1	John Smith	3	12/17/2023	12:00:00	Pending
1	John Smith	4	12/17/2023	11:00:00	Pending
1	John Smith	5	12/21/2023	10:00:00	Pending

2. Count of Appointments by Doctor:

```
SELECT
    CS.Name AS DoctorName,
    COUNT(A.AppointmentID) AS AppointmentCount
FROM
    Clinical_Staff CS
JOIN
    Doctors D ON CS.StaffID = D.StaffID
JOIN
    Appointments A ON D.StaffID = A.StaffID
```

```
GROUP BY
    CS.Name;
```

DOCTORNAME	APPOINTMENTCOUNT
Dr. Sarah Elif	2
Dr. John Smith	3

3. List of Treatments and their Usage:

```
SELECT
    T.Name AS TreatmentName,
    IT.UsageNotes
FROM
    Treatment T
JOIN
    Inventory_Item IT ON T.TreatmentID = IT.TreatmentID;
```

TREATMENTNAME	USAGENOTES
Pain Relief	Usage note for Painkiller
Pain Relief	Usage note for Antibiotic
Physical Therapy	Usage note for Cough Syrup

4. Billing Information for a Patient:

```
SELECT
    B.BillingID,
    B.AppointmentID,
    P.Name AS PatientName,
    B.BillingMethod,
    B.PayableAmount,
    B.BDate
FROM
    Billing B
JOIN
    Patient P ON B.PatientID = P.PatientID;
```

BILLINGID	APPOINTMENTID	PATIENTNAME	BILLINGMETHOD	PAYABLEAMOUNT	BDATE
1	1	John Smith	cash	500	12/16/2023
2	2	Alice Johnson	cash	50	12/16/2023

2 rows returned in 0.02 seconds Download

5. Whole Staff Data:

```
SELECT
    CS.Name AS StaffName,
    CS.Address,
    CS.ContactInfo,
    S.StaffID,
    S.SecurityLicenseNumber,
    NULL AS CareCertificateNumber,
    S.Salary,
    NULL AS AccessLevel  -- Placeholder for Security Personnel (No Access Level)
FROM
    Clinical_Staff CS
JOIN
    Security_Personnel S ON CS.StaffID = S.StaffID

UNION ALL

SELECT
    CS.Name AS StaffName,
    CS.Address,
    CS.ContactInfo,
    ST.StaffID,
    NULL AS SecurityLicenseNumber,
    CareCertificateNumber,
    ST.Salary,
    ST.AccessLevel

FROM
```

```
    Clinical_Staff CS
JOIN
    Sanitation_Team ST ON CS.StaffID = ST.StaffID;
```

STAFFNAME	ADDRESS	CONTACTINFO	STAFFID	SECURITYLICENSENUMBER	CARECERTIFICATENUMBER	SALARY	ACCESSLEVEL
Jack Thompson	202 Elm St	7778889999	5	SLN11111	-	32000	-
Mark Davis	789 Pine St	5551112222	3	-	CC67890	20000	Medium
Emily White	101 Maple St	3334445555	4	-	CC11111	18000	Low

QUESTION#10:

Demonstration of at least two functions, two stored procedures, and two database triggers.

Triggers:

Trigger-01

```
CREATE OR REPLACE TRIGGER Check_Appointment_Availability
BEFORE INSERT OR UPDATE ON Appointments
FOR EACH ROW
DECLARE
   StartHour VARCHAR2(5);
   EndHour VARCHAR2(5);
BEGIN
   -- Fetch the availability hours for the specified doctor
   SELECT AvailabilityStartHour, AvailabilityEndHour
   INTO StartHour, EndHour
   FROM Doctors
   WHERE StaffID = :NEW.StaffID;

   -- Check if the appointment time is within the doctor's availability
   IF :NEW.AppointmentTime NOT BETWEEN StartHour AND EndHour THEN
      RAISE_APPLICATION_ERROR(-20001, 'Appointment time is outside of doctor''s availability.');
   END IF;
END;
```

Trigger-02

```
CREATE OR REPLACE TRIGGER CheckApptStatus
```

```
BEFORE INSERT ON Billing
FOR EACH ROW DECLARE
vAppointmentStatus VARCHAR2(10);
BEGIN
   -- Retrieve the appointment status for the specified AppointmentID
   SELECT Status INTO vAppointmentStatus
   FROM Appointments
   WHERE AppointmentID = :NEW.AppointmentID;

   -- Check if the appointment status is 'Clear'
   IF vAppointmentStatus != 'Clear' THEN
      RAISE_APPLICATION_ERROR(-20001, 'Billing can only be created for appointments with status ''Clear''.');
   END IF;
END;
```

Procedures:

Procedure-01

```
CREATE OR REPLACE PROCEDURE UpdatePatientWeight(
pPatientID NUMBER,     pNewWeight NUMBER
) AS
BEGIN
   UPDATE Patient SET Weight = pNewWeight WHERE PatientID = pPatientID;
END;
```

```
Patient weight updated successfully.

Statement processed.
```

PATIENTID	NAME	DOB	HEIGHT	WEIGHT	BP
1	John Smith	05/15/1990	170	75	120/80
2	Alice Johnson	08/22/1985	160	55	110/70

2 rows returned in 0.35 seconds Download

Procedure-02

```
CREATE OR REPLACE PROCEDURE GenerateBilling(
pAppointmentID NUMBER,     pBillingMethod
VARCHAR2,     pPayableAmount NUMBER
) AS     vPatientID
NUMBER;     vBillingID
NUMBER;
BEGIN
   SELECT PatientID INTO vPatientID FROM Appointments WHERE
AppointmentID = pAppointmentID;        vBillingID :=
SEQ_BillingID.NEXTVAL;


   INSERT INTO Billing (BillingID, AppointmentID, PatientID, BillingMethod,
PayableAmount, BDate)
   VALUES (vBillingID, pAppointmentID, vPatientID, pBillingMethod,
pPayableAmount, SYSDATE);
END;
--Let's update the status of patient with id 2 to check procedure:
UPDATE Appointments SET
Status = 'Clear'
WHERE PatientID = 2;
```

```
Billing entry generated successfully.

Statement processed.
```

BILLINGID	APPOINTMENTID	PATIENTID	BILLINGMETHOD	PAYABLEAMOUNT	BDATE
1	1	1	cash	500	12/16/2023
2	2	2	cash	50	12/16/2023

Functions:

Function-01

```
CREATE OR REPLACE FUNCTION CalculateAge(pPatientID NUMBER) RETURN
NUMBER IS     vDOB DATE;     vAge NUMBER;
BEGIN
   -- Retrieve the date of birth from the Patient table using the provided PatientID
   SELECT DOB INTO vDOB FROM Patient WHERE PatientID = pPatientID;

   -- Calculate the age using the retrieved date of birth     vAge
:= TRUNC(MONTHS_BETWEEN(SYSDATE, vDOB) / 12);

   RETURN vAge;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      -- Raise a custom exception to signal that no data was found
      RAISE_APPLICATION_ERROR(-20001, 'Patient not found');
END;
--Function Call
DECLARE
   vAge NUMBER; BEGIN     vAge :=
CalculateAge(1); -- Assuming PatientID 1
   DBMS_OUTPUT.PUT_LINE('Age: ' || vAge);
END;
```

```
Age: 33

Statement processed.
```

Function-02

```
CREATE OR REPLACE FUNCTION GetEarlierUpcomingAppointment(pPatientID
NUMBER)
RETURN VARCHAR2 IS
vNextAppointment VARCHAR2(50);
BEGIN
vNextAppointment:='Nothing';
   -- Select the earliest upcoming pending appointment for the specified patient
SELECT MIN(AppointmentDate) || ' ' || MIN(AppointmentTime)
   INTO vNextAppointment
   FROM Appointments
   WHERE PatientID = pPatientID AND
        AppointmentDate >= SYSDATE AND
        Status = 'Pending';

   -- If no data is found, handle the case where no pending appointment is found
   IF vNextAppointment like 'Nothing' THEN
      RETURN 'NoAppointment'; -- Use a sentinel value instead of NULL
   ELSE
      RETURN vNextAppointment;
 END IF;


EXCEPTION
   WHEN NO_DATA_FOUND THEN
      -- Raise a custom exception to signal that no data was found
      RAISE_APPLICATION_ERROR(-20001, 'Patient not found or no appointment
scheduled yet');
END;
```

--Function Call

--Updating a record to check function working

```
UPDATE Appointments
SET AppointmentDate = TO_DATE('21/12/2023', 'DD/MM/YYYY')
WHERE PatientID = 1 AND AppointmentID = 5;

DECLARE     vAppointmentInfo VARCHAR2(100); BEGIN
vAppointmentInfo := GetEarlierUpcomingAppointment(1);
   -- Check if the result is the sentinel value
   IF vAppointmentInfo = 'NoAppointment' THEN
      DBMS_OUTPUT.PUT_LINE('No pending upcoming appointments found.');
ELSE
      DBMS_OUTPUT.PUT_LINE('Next Appointment: ' ||VAppointmentInfo);
END IF;
END;
```

```
Next Appointment: 12/21/2023 10:00:00

Statement processed.
```

Function-03

```
CREATE OR REPLACE FUNCTION GetTotalPaidAmount(pPatientID NUMBER)
RETURN DECIMAL IS     vTotalPayable NUMBER(10,2) := 0;
BEGIN
   -- Calculate the total payable amount for the given patient
   SELECT SUM(PayableAmount)
   INTO vTotalPayable
   FROM Billing
   WHERE PatientID = pPatientID;

   -- If no data is found, return 0
   RETURN NVL(vTotalPayable, 0);

END;
```

--Function Call

```
DECLARE     vTotalPaid
NUMBER(10,2); BEGIN
   vTotalPaid := GetTotalPayableAmount(1); -- Replace 3 with the desired
PatientID
   IF vTotalPaid != 0 THEN
      DBMS_OUTPUT.PUT_LINE('Total Payable Amount: ' || vTotalPaid);
   ELSE
      DBMS_OUTPUT.PUT_LINE('Either the patient does not exist or has not paid
yet');
   END IF;
END;
```

```
Total Payable Amount: 500

Statement processed.
```

Function-04

```
CREATE OR REPLACE FUNCTION HasUpcomingAppointment(
pPatientID NUMBER ) RETURN NUMBER IS
appointmentCount NUMBER(1);
BEGIN
   -- Count pending appointments for the specified patient
   SELECT COUNT(*)
   INTO appointmentCount
   FROM Appointments
   WHERE PatientID = pPatientID AND
        AppointmentDate >= TRUNC(SYSDATE) AND -- Compare only date portion
        Status = 'Pending';

   -- Return 1 if at least one appointment exists, 0 otherwise
   RETURN CASE WHEN appointmentCount > 0 THEN 1 ELSE 0 END;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
```

```
        RETURN 0;
END;
--Function Call
DECLARE     vHasAppointment NUMBER(1); BEGIN
vHasAppointment := HasUpcomingAppointment(1);

    IF vHasAppointment = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Patient has an upcoming appointment.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Patient has no upcoming appointments.');
    END IF;
END;
```

```
Patient has an upcoming appointment.

Statement processed.
```

`