

WildCat HealthCare



Team : S2Q - SYSADMINIS

Koduri, Hrithik Ashok
Lakka, Netranandini
Mishra, Pallavi
Myakala, Chaitanya Rajeev
Nair, Praveen Rajendran
Vemulapalli, Rajesh

Github Link : bit.ly/wildcatCodeRepo

Website Link : <https://bit.ly/WildcatHealthCare>

Indexes

Chapter 1	3
Requirement Analysis and ER description	3
Chapter 2	5
ER Diagram	5
Chapter 3	6
Relational Schema(4NF)	6
Data Dictionary	8
Chapter 4	16
DML	16
Indexes	24
DDL	24
Chapter 5	26
Triggers and Procedures	26
Triggers	26
ON_INSERT_EMPLOYEE	26
ON_INSERT_APPOINTMENT_TABLE	26
Trigger_Patinet_and_Employee_NewUser	28
Appointment_ID_Creation	29
Procedures	29
Update_MeetingLink	29
Update_Employee_Experience	30
Type_Employee	31
Functions	31
ENCRYPT_SSN	31
Chapter 6	32
User Interface	32
Chapter 7	46
Implementation Plan - Praveen and Chaitanya	46
Appendix A	46
Lessons Learnt - Praveen and Chaitanya	46
Create Table Statements	46

Chapter 1

Requirement Analysis and ER description

Wildcat Healthcare is a hospital based in Tucson, United States. This health system is the largest employer in Arizona and is one of the largest in the United States with over a thousand employees.

Given that Wildcat Healthcare is a sophisticated system with diagnosis labs, offers surgical procedures and treatments, and sells medications, they want to create a database to keep track of their employees, patients, appointments, medical procedures, and bills. The Wildcat Healthcare database must be able to meet all their operational requirements. These key data requirements are outlined.

The Employees working at Wildcat Healthcare have an employee ID, SSN, an employee name (first name, middle name, last name), date of joining, age, gender, designation, experience, email-ID, area, and type.

Employees are two types, can either be medical staff or non-medical staff. Non-medical staff will be classified depending on their work type, whereas the medical staff is classified depending on their specialization. Medical staff will get a meeting link if an appointment has been booked with the respective doctor. Employees [0:M] may be associated with a single department [1:1], and each department has its own Department ID and Department name. Each member in the medical staff may or may not take appointments, but every appointment [0:M] is associated with one medical staff member [1:1]. Each appointment taken stores data such as appointment ID, type and appointment details (appointment date and appointment time). Each employee [1:1] may receive pay slips [0:M] with Payment ID, amount and payment date. During each appointment [1:1], there could or could not be any diagnosis [0:M], and each diagnosis is recorded by its ID and name.

The hospital maintains records of its patients. A patient is identified by patient ID, patient name (first name, last name), gender, age, email, address (street name, area name, city, state and pin code). Each patient [1:1] can hold multiple medical records [0:M] which consist of record id, date, known disease and diagnosis. Each patient [1:1] may pay medical bills [0:M] and every medical bill is associated with single patient.

Every bill stores information like, patient bill ID, patient bill status, bill date, amount, transaction information.

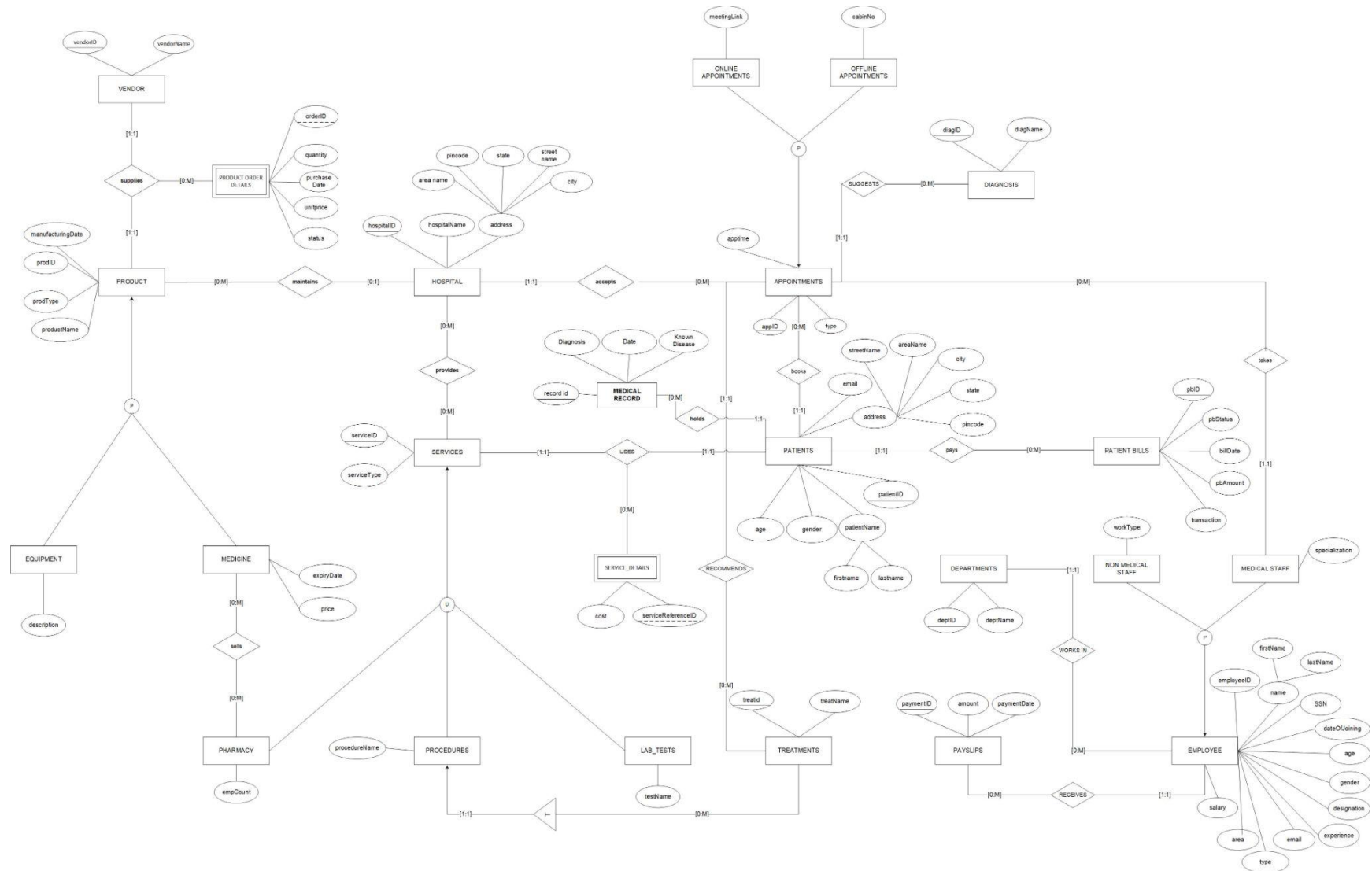
The hospital accepts appointments from patients. To be noted that each appointment [0:M] that has been booked is only associated with one patient [1:1]. After pandemic hit, Wildcat Healthcare started serving both offline and online pertaining to different services. So, a patient can book either an online appointment, who receives a meeting link or can book for an offline appointment, who gets informed about the cabin room number, before arriving at the hospital. Each registered appointment will have details such as application ID, type, appointment details (date and time).

The organization provides procedures, lab tests and a pharmacy as one of the parts of its service offerings. Hospital registers procedure name for every procedure, employee count for every pharmacy, and test name for every test done. Treatments [0:M] come under procedures [1:1] and are identified by a treatment ID and a name. Treatments are generally recommended during the appointment with the doctors. Depending on the patient condition, during each appointment [1:1], patient may not be recommended to undergo treatment [0:M] or sometimes multiple treatment procedures. Each patient [1:1] uses one service at a time [1:1]. Services have service ID and service type. Hospital also stores service details such as Service reference ID and cost.

Hospital has its own Hospital ID, name and address (area name, pin code, state, street name and city). This hospital maintains products which will be supplied by vendors. Every product [0:M] may or may not be associated with the hospital [0:1], and the hospital may or may not have those products. Every product is supplied by a specific vendor. And one vendor [1:1] can supply only that particular product [1:1]. Hospital also stores product order details, which includes Order ID, quantity, purchase date, unit price and status.

The products are of two types, equipment and medicines. All the equipment will have their own manufacturing date, but a medicine will have an expiry date, price along with the manufacturing date. Medicines are sold in pharmacy. Pharmacy might run out of medicines or may have full stock and the particular medicine [0:M] may or may not be sold in that pharmacy [0:M]

ER Diagram



Chapter 3

Relational Schema(4NF)

HOSPITALS(hospitalID, name, area, street, city, state, pincode)

PRODUCTS(prodID, name, type, hospitalID, manufacturingDate)

Foreign key hospitalID references HOSPITALS

EQUIPMENT(equipmentID, description)

Foreign key equipmentID references PRODUCTS

MEDICINES(medicineID, expiryDate, Price)

Foreign key medicineID references PRODUCTS

VENDORS(vendorID, name)

PRODUCT_ORDER_DETAILS(prodID, vendorID, orderID, quantity, purchaseDate, unitPrice, status)

Foreign key vendorID references VENDORS

Foreign key prodID references PRODUCTS

SERVICES(serviceID, type)

HOSPITAL_PROVIDES_SERVICES(hospitalID, serviceID)

PHARMACY(pharmaID, empCount)

Foreign key pharmaID references SERVICES

PHARMACY_SELLS_MEDICINES(pharmaID, medicineID)

Foreign key pharmaID references SERVICES

Foreign key medicineID references PRODUCTS

PROCEDURES(procedureID, name)

Foreign key procedureID references SERVICES

LAB_TESTS(labID, name)

Foreign key labID references SERVICES

TREATMENTS(treatmentID, name, procedureID, appID)

Foreign key procedureID references SERVICES

Foreign key appID references APPOINTMENTS

PATIENTS(patientID, firstName, LastName, age, gender, email, area, street, city, state, pincode)

PATIENT_BILLS(pbID, status, date, amount, transaction, patientID)

Foreign key patientID references PATIENTS

SERVICE_DETAILS(patientID, serviceID, serviceRefID, cost)

Foreign key patientID references PATIENTS

Foreign key serviceID references SERVICES

APPOINTMENTS(appID, type, appTime, patientID, hospitalID, medicalStaffID)

Foreign key patientID references PATIENTS

Foreign key hospitalID references HOSPITALS

Foreign key medicalStaffID references MEDICALSTAFF

ONLINE_APPOINTMENTS(appID, meetingLink)

OFFLINE_APPOINTMENTS(appID, cabinNo)

MEDICAL_RECORDS(recordID, diagnosis, date, knownDisease, patientID)

Foreign key patientID references PATIENTS

DIAGNOSIS(diagID, name, appID)

Foreign key appID references APPOINTMENTS

EMPLOYEES(empID, firstName, LastName, SSN, dateOfJoining, age, gender, designation, experience, email, type, area, departmentID, salary)

Foreign key departmentID references DEPARTMENTS

UNIQUE constraint(SSN)

MEDICAL_STAFF(msID, specialization)

NON_MEDICAL_STAFF(nmsID, workType)

PAYSLIPS(paymentID, amount, paymentDate, empID)

Foreign key empID references EMPLOYEES

DEPARTMENTS(deptID, name)

Data Dictionary

Schema Construct	Construct description	Other Information
APPOINTMENTS	Entity class to model appointments information	
appointmentDetails	Composite attribute consisting of appointment date and time	Attribute should not contain null values
appID	Identifying number of the appointment	Attribute should not contain null values
appTime	time of the appointment	Attribute should not contain null values
appDate	Date of the appointment	Attribute should not contain null values
type	Type of the appointment (Online/ Offline)	Attribute should not contain null values
DEPARTMENT	Entity class to model department information	
deptID	Identifying number of the department	Identifying attribute
deptName	Name of the department	Attribute should not contain null values
DIAGNOSIS	Entity class to model diagnosis information	
diagnosisID	Identifying number of diagnosis	Identifying attribute

diagnosisName	Name of the diagnosis	Attribute should not contain null values
EMPLOYEE	Entity class to model Employee information	
employeeid	Identifying number for employees	Identifying attribute
area	Area of specialization	
age	Age of the patient	Cannot be negative, typically range from 0-150
email	E mail ID of the employee	Attribute should not contain null values
experience	Years of working experience in the hospital	Cannot be negative, typically range from 0-100
designation	Designation of the employee	
gender	Gender of the employee	Gender can be male/female/other
salaryRange	Salary range of the employee	Cannot be negative
SSN	Social Security Number of the employee	Attribute should not contain null values
dateOfJoining	Date of joining of the employee	
type	Type (Medical / Non-Medical staff)	Attribute should not contain null values

name	Name of the employee	Cannot be null
EQUIPMENT	Entity class to model Equipment information	
manufacturingDate	Date of equipment manufacturing	
HOSPITAL	Entity Class to model Hospital information	
hospitalID	Identifying number of the hospital	Identifying attribute
name	Name of the hospital	Attribute should not contain null values
location	Composite attribute made up of pincode, state, streetname,city,area	Attribute should not contain null values
pincode	Pincode where the hospital is located.	Attribute should not contain null values
state	State where the hospital is located.	Attribute should not contain null values
streetName	Name of the street where the hospital is located.	Attribute should not contain null values
city	Name of the city where hospital is located.	Attribute should not contain null values
area	Name of area where hospital is located.	Attribute should not contain null values
LAB_TESTS	Entity class to model LABS information	
testName	Tests performed in the lab	Cannot be null

MEDICINE	Entity class to model Medicine information	
manufactureDate	Date the Drug has been manufactured	Attribute should not contain null values
price	The cost of drug	Attribute should not contain null values
expiryDate	Expiry date of the medicine	Attribute should not contain null values
MEDICAL_RECORDS	Entity class to model medical record information	
recordID	Identifying number of the record	Attribute should not contain null values
diagnosis	The details of the diagnosis	Attribute should not contain null values
date	Date of registration of the patient	Attribute can contain null values.
knownDiseases	List of known conditions for the patient	Attribute can contain null values.
MEDICAL_STAFF	Entity class to model medical staff information	
specialization	Specialization of medical staff	Attribute should not contain null values
meetingLink	Appointment meeting link that will be sent	Attribute should not contain null values
NON_MEDICAL_STAFF	Entity Class to Model non medical staff Information	

workType	Type of work non-medical staff doing	Can be null
OFFLINE_APPOINTMENTS	Entity Class to Model offline Appointment Information	
cabinNo	Cabin number of the appointment	Attribute should not contain null values
ONLINE_APPOINTMENTS	Entity Class to Model Online appointment Information	
meetingLink	Meeting link of the appointment	Attribute should not contain null values
PATIENTS	Entity Class to Model Patient Information	
patientID	Identifying number of the patient	Identifying attribute
gender	Gender of the patient	Gender can be male/female/other
firstName	First name of the patient	Attribute should not contain null values
lastName	Last name of the patient	Attribute should not contain null values
age	Age of the patient	
email	Email address of the patient	Attribute should not contain null values
street	Street address of the patient	Attribute should not contain null values

areaName	Area name the patient lives in	Attribute should not contain null values
city	City the patient lives in	Attribute should not contain null values
state	State the patient lives in	Attribute should not contain null values
pincode	Pin code of the address	Attribute should not contain null values
PATIENT_BILLS	Entity class to model PatientBills Information	
patinetBillID	Patient Bill ID to uniquely identify the bill	Identifying Attribute
patientBillAmount	Bill amount of the patient	Attribute should be zero or positive
patientBillStatus	Status of the bill payment	
billDate	Date on which the bill was made	Attribute should not contain null values
transaction	Transaction information	
PAY_SLIPS	Entity class to model PatientBills Information	
paymentID	Unique identifier for the payslips	Identifying Attribute
date	Date of the salary payment	Attribute should not contain null values

amount	Amount of salary paid	Attribute should not contain null values
PHARMACY	Entity class to model pharmacy information	
empCount	Number of employees in the pharmacy	Attribute should not contain null values
PROCEDURES	Entity Class to model Procedure Information	
procedureName	Name of the procedure.	Attribute should not contain null values
PRODUCT	Entity Class to model product Information	
productID	Unique identifier for the product	Identifying Attribute
productType	Type of the product	Attribute should not contain null values
productName	Name of the product	
PRODUCT_ORDER_DETAILS	Entity Class to model product order details Information	
orderId	Unique identifier for the product order details	Identifying Attribute
quantity	Quantity of the products ordered	Attribute should not contain null values
purchaseDate	Date of the products ordered	Attribute should not contain null values

unitPrice	Price of the ordered products	Attribute should not contain null values
status	Status of the order delivery	
SERVICE_DETAILS	Entity Class to model service details Information	
serviceReferenceID	Unique identifier for the service details	Identifying Attribute
cost	Cost of the service	
SERVICES	Entity class to model services information	
serviceID	Identifying number for the services offered by hospital	Identifying attribute
serviceType	Name of the services offered by the hospital	Attribute should not contain null values
TREATMENT	Entity class to model Treatment information	
treatmentID	Identifying number for the treatment	Identifying attribute
treatmentName	Description of the treatment done	Details of the treatment
VENDOR	Entity Class to Model vendor Information	
vendorID	Identifying number for vendor	Supplier attribute
vendorName	Name of the vendor	Attribute should not contain null values

Chapter 4

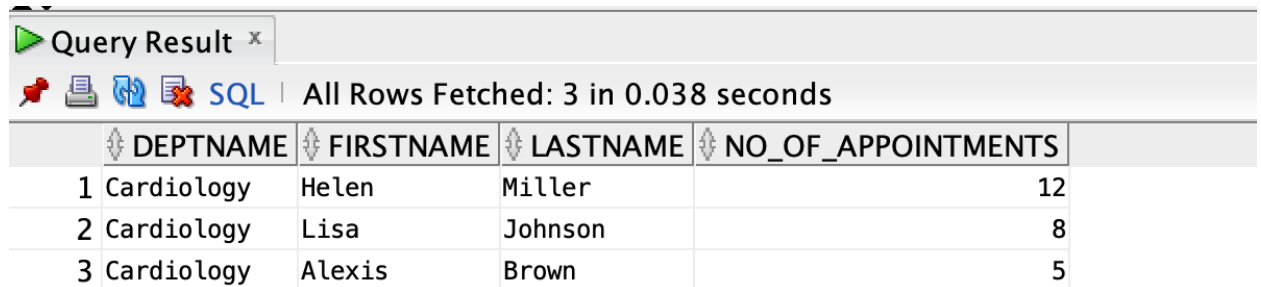
DML

1. **Query:** To get doctors with highest appointment in a specified department
Concepts: with table, dense rank

Code:

```
with dept_appointment as (  
    select deptname, e.firstname, e.lastname, count (*) as  
no_of_appointments  
    from appointment a join employee e  
    on a.employeeid = e.employeeid  
    join department d  
    on e.deptid = d.deptid  
    group by deptname, e.firstname, e.lastname  
) ,  
ranked_appointment as (  
    select deptname, firstname, lastname, no_of_appointments,  
    dense_rank() over (partition by deptname order by  
no_of_appointments) as drank  
    from dept_appointment  
    where deptname = 'Cardiology'  
)  
select deptname, firstname, lastname, no_of_appointments  
from ranked_appointment order by drank desc;
```

Output:



Query Result x				
SQL All Rows Fetched: 3 in 0.038 seconds				
	DEPTNAME	FIRSTNAME	LASTNAME	NO_OF_APPOINTMENTS
1	Cardiology	Helen	Miller	12
2	Cardiology	Lisa	Johnson	8
3	Cardiology	Alexis	Brown	5

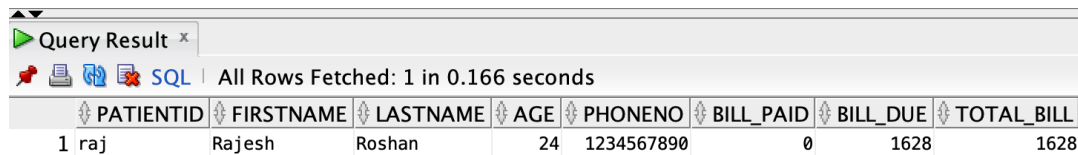
2. **Query:** To get pending bill, paid bill and total bill of patients
Concepts: with table, dpivot

Code:

```

with pivoted_paitent_bill as (
select * from (select * from patient_bill pb)
pivot (
sum(pbamount) for PBStatus in ('Paid' as Patient_Bill_Paid,
'Pending' as Patient_Bill_Pending)
)
),
patient_all_bills as (
select patientid, coalesce(sum(Patient_Bill_Paid), 0) as
Bill_paid,coalesce(sum(Patient_Bill_Pending), 0) as Bill_Due
from pivoted_paitent_bill
group by patientid
)
select p.PatientID, firstname, lastname, age, phoneno,
Bill_Paid, Bill_Due, (Bill_Paid + Bill_Due) as Total_Bill
from patient p
join patient_all_bills pab on p.patientid = pab.patientid
where p.patientID = 'raj';

```

Output:


The screenshot shows a 'Query Result' window with the following data:

PATIENTID	FIRSTNAME	LASTNAME	AGE	PHONENO	BILL_PAID	BILL_DUE	TOTAL_BILL
1 raj	Rajesh	Roshan	24	1234567890	0	1628	1628

3. **Query:** To get list of patients with pending bill greater than certain amount(here \$5000)

Concepts: with table, fetch, group by with having

Code:

```

with due_bill as (
select patientID, sum(pbamount) as Bill_Due from
Patient_bill
where PBstatus = 'Pending'
group by patientID
having sum(pbamount) > 5000
)

select p.PatientID, firstname, lastname, age, gender, phoneno,
Bill_Due
from patient p join due_bill d on p.patientId = d.patientID
order by Bill_Due desc
fetch first 5 rows only;

```

Output:

Query Result x						
All Rows Fetched: 5 in 0.1 seconds						
	PATIENTID	FIRSTNAME	LASTNAME	AGE	GENDER	BILL_DUE
1	P016	chaitanya	rajeev	29	Other	9876
2	P010	chaitanya	rajeev	46	Other	9252
3	P008	chaitanya	rajeev	37	Male	8836
4	P009	chaitanya	rajeev	49	Male	8635
5	P020	chaitanya	rajeev	61	Female	7655

4. **Query:** To get highest paid medical staff in a specific department (here Cardiology)

Concepts: row_number, partition by

Code:

```
select EmployeeID, e.deptID, DeptName as Department_Name
      ,firstname as "Employee First Name"
      , lastname as "Employee Last Name", Age, Gender,
Designation, salary
      , row_number() over (partition by e.deptID order by salary
desc) orderbysalary
from employee e join
department d on e.deptid = d.deptid
where deptName = 'Cardiology';
```

Output:

Query Result

SQL

All Rows Fetched: 7 in 0.041 seconds

	EMPLOYEEID	DEPTID	DEPARTMENT_NAME	Employee First Name	Employee Last Name	AGE	GENDER	DESIGNATION	SALARY	ORDERBYSALARY
1	1024	D102	Cardiology	Alexis	Brown	42	Female	Doctor	98390	1
2	1025	D102	Cardiology	Lisa	Johnson	44	Female	Doctor	82632	2
3	1014	D102	Cardiology	Dustin	Anderson	33	Male	Nurse	73000	3
4	1026	D102	Cardiology	Helen	Miller	45	Female	Doctor	71652	4
5	1015	D102	Cardiology	Mark	Thomas	50	Other	Nurse	53000	5
6	1043	D102	Cardiology	Alexis	Dora	42	Female	Doctor	23456	6
7	1023	D102	Cardiology	Alexis	Dora	42	Female	Doctor	23456	7

5. **Query:** To get Equipments and Medicines which are not delivered and order amount is greater than a certain value (here 2000)

Concepts: multiple joins

Code:

```
- EQUIPMENTS
select vendorname, equipname, manufacturingdate, quantity,
unitprice, (quantity*unitprice) as Pending_Order_Amount, status
```

```

from PRODUCT_ORDER_DETAILS d
join product p
on d.prodid = p.prodid
join equipment e
on p.prodid = e.prodid
join vendor v
on d.vendorid = v.vendorid
where status in ('Shipped', 'Packed')
and prodtype = 'EQUIPMENT'
and (quantity*unitprice) > 2000
order by Pending_Order_Amount desc;

```

Output:

Query Result x							
All Rows Fetched: 5 in 0.056 seconds							
VENDORNAME	EQUIPNAME	MANUFACTURINGDATE	QUANTITY	UNITPRICE	PENDING_ORDER_AMOUNT	STATUS	
1 Koninklijke Philips	OPD furniture	11-07-2001	10	579	5790	Shipped	
2 Medtronic PLC	Defibrillators	22-05-2022	8	400	3200	Shipped	
3 Abbott Laboratories	Defibrillators	22-05-2022	6	495	2970	Shipped	
4 Novartis AG	Defibrillators	22-05-2022	6	345	2070	Packed	
5 Centene Corp.	Beds	21-05-2013	4	504	2016	Packed	

Code:

```

- MEDICINE
select vendorname, prodname, manufacturedate, expdate,
quantity, unitprice, (quantity*unitprice) as
Pending_Order_Amount, status
from PRODUCT_ORDER_DETAILS d
join product p
on d.prodid = p.prodid
join medicine m
on p.prodid = m.prodid
join vendor v
on d.vendorid = v.vendorid
where status in ('Shipped', 'Packed')
and prodtype = 'Medicine'
and (quantity*unitprice) > 2000
order by Pending_Order_Amount desc;

```

Output:

Query Result x							
SQL All Rows Fetched: 5 in 0.053 seconds							
VENDORNAME	PRODNAME	MANUFACTUREDDATE	EXPDATE	QUANTITY	UNITPRICE	PENDING_ORDER_AMOUNT	STATUS
1 Medtronic PLC	Lisinopril	29-01-2019	24-06-2024	12	810	9720	Packed
2 Henry Schein	Metformin	27-07-2017	24-06-2024	10	862	8620	Shipped
3 Danaher Corp	Metformin	27-07-2017	24-06-2024	8	928	7424	Packed
4 Abbott Laboratories	Amoxicilin	10-04-2020	24-06-2025	8	823	6584	Packed
5 Henry Schein	Amoxicilin	10-04-2020	24-06-2025	7	778	5446	Packed

6. **Query:** To get vendors list for Equipments and Medicines which are not delivered and order amount pending is greater than a certain value (here 2000) along with total order pending from those vendors

Concepts: multiple joins, cube

Code:

```
select coalesce (vendorname, 'Total'), sum(quantity*unitprice)
as Pending_Order_Amount
from PRODUCT_ORDER_DETAILS d
join product p
on d.prodid = p.prodid
join medicine m
on p.prodid = m.prodid
join vendor v
on d.vendorid = v.vendorid
where status in ('Shipped', 'Packed')
and prodtype = 'Medicine'
and (quantity*unitprice) > 2000
group by cube(vendorname)
order by Pending_Order_Amount desc;
```

Output:

Query Result x	
SQL All Rows Fetched: 5 in 0.332 seconds	
COALESCE(VENDORNAME,'TOTAL')	PENDING_ORDER_AMOUNT
1 Total	37794
2 Henry Schein	14066
3 Medtronic PLC	9720
4 Danaher Corp	7424
5 Abbott Laboratories	6584

Code:

```
select coalesce (vendorname, 'Total'), sum(quantity*unitprice)
as Pending_Order_Amount
```

```

from PRODUCT_ORDER_DETAILS d
join product p
on d.prodid = p.prodid
join equipment e
on p.prodid = e.prodid
join vendor v
on d.vendorid = v.vendorid
where status in ('Shipped', 'Packed')
and prodtype = 'EQUIPMENT'
and (quantity*unitprice) > 2000
group by cube(vendorname)
order by Pending_Order_Amount desc;

```

7. **Query:** To get the top 5 appointments based on chronological order for each department
Concepts: with, rank, partition by

Code:

```

with appointment_department as (

    select deptname, e.firstname as employee_firstname,
    e.lastname as employee_lastname, a.appid as appointmentId,
    patientid, appointment_date, appointment_time,
    dense_rank() over(partition by deptname order by
    appointment_date, appointment_time) as drank
    from appointment a
    join employee e
    on a.employeeid = e.employeeid
    join department d
    on e.deptid = d.deptid
)

select deptname, employee_firstname, employee_lastname,
appointmentId, patientid, appointment_date, appointment_time
from appointment_department
where drank <=5;

```

Output:

Query Result x						
All Rows Fetched: 46 in 0.268 seconds						
DEPTNAME	EMPLOYEE_FIRSTNAME	EMPLOYEE_LASTNAME	APPOINTMENTID	PATIENTID	APPOINTMENT_DATE	APPOINTMENT_TIME
1 Cardiology	Lisa	Johnson	A381	P003	16-04-2009	08:30
2 Cardiology	Lisa	Johnson	A435	P003	16-04-2009	08:30
3 Cardiology	Lisa	Johnson	A408	P003	16-04-2009	08:30
4 Cardiology	Lisa	Johnson	A322	P003	16-04-2009	11
5 Cardiology	Lisa	Johnson	A323	P004	09-03-2012	13
6 Dermatology	Praveen	Nair	A260	raj	29-11-0022	13
7 Dermatology	Praveen	Nair	A360	himanshu	30-11-0022	11
8 Dermatology	Praveen	Nair	A341	raj	30-11-0022	12
9 Dermatology	Praveen	Nair	A281	raj	30-11-0022	13
10 Dermatology	Praveen	Nair	A501	raj	01-12-0022	11

8. **Query:** To get the online and offline appointment details for patients
Concepts: joins

Code:

--ONLINE APPOINTMENT

```
select a.appid, firstname, lastname, email, meetinglink,
appointmentdate, appointmenttime
from appointment a
join employee e
on a.employeeid = e.employeeid
join online_appointment oa
on a.appid = oa.appid
WHERE patientid = 'raj';--patient id
```

Output:

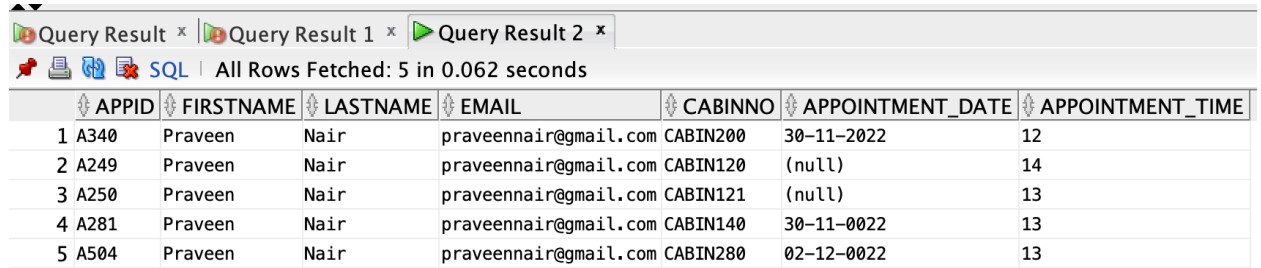
Query Result x Query Result 1 x Query Result 2 x						
All Rows Fetched: 22 in 0.039 seconds						
APPID	FIRSTNAME	LASTNAME	EMAIL	MEETINGLINK	APPOINTMENT_DATE	APPOINTMENT_TIME
1 A520	Praveen	Nair	praveennair@gmail.com	https://us05web.zoom...	06-12-0022	15
2 A561	Hrithik	Koduri	hrithik@gmail.com	https://us04web.zoom...	05-12-0022	12
3 A280	Praveen	Nair	praveennair@gmail.com	https://us05web.zoom...	01-12-0022	13
4 A341	Praveen	Nair	praveennair@gmail.com	https://us05web.zoom...	30-11-0022	12
5 A501	Praveen	Nair	praveennair@gmail.com	https://us05web.zoom...	01-12-0022	11

Code:

--OFFLINE APPOINTMENT

```
select a.appid, firstname, lastname, email, cabinno,
appointmentdate, appointmenttime
from appointment a
join employee e
on a.employeeid = e.employeeid
join offline_appointment oa
on a.appid = oa.appid
WHERE patientid = 'raj';--patient id
```

Output:



Query Result x Query Result 1 x Query Result 2 x

SQL | All Rows Fetched: 5 in 0.062 seconds

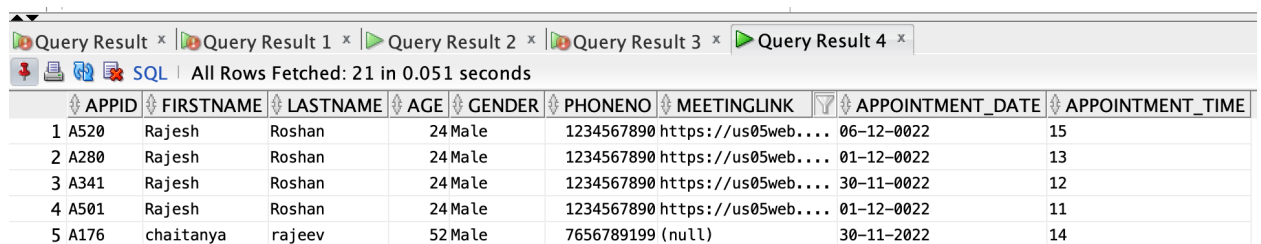
	APPID	FIRSTNAME	LASTNAME	EMAIL	CABINNO	APPOINTMENT_DATE	APPOINTMENT_TIME
1	A340	Praveen	Nair	praveennair@gmail.com	CABIN200	30-11-2022	12
2	A249	Praveen	Nair	praveennair@gmail.com	CABIN120	(null)	14
3	A250	Praveen	Nair	praveennair@gmail.com	CABIN121	(null)	13
4	A281	Praveen	Nair	praveennair@gmail.com	CABIN140	30-11-0022	13
5	A504	Praveen	Nair	praveennair@gmail.com	CABIN280	02-12-0022	13

9. **Query:** To get the online and offline appointment details for doctors
Concepts: joins

Code:

```
--ONLINE APPOINTMENT
select a.appid, firstname, lastname, age, gender, phoneno,
meetinglink, appointmentdate, appointmenttime
from appointment a
join patient p
on a.patientid = p.patientid
join online_appointment oa
on a.appid = oa.appid
WHERE employeeid = 'prav';--employee id for doctor
```

Output:



Query Result x Query Result 1 x Query Result 2 x Query Result 3 x Query Result 4 x

SQL | All Rows Fetched: 21 in 0.051 seconds

	APPID	FIRSTNAME	LASTNAME	AGE	GENDER	PHONENO	MEETINGLINK	APPOINTMENT_DATE	APPOINTMENT_TIME
1	A520	Rajesh	Roshan	24	Male	1234567890	https://us05web....	06-12-0022	15
2	A280	Rajesh	Roshan	24	Male	1234567890	https://us05web....	01-12-0022	13
3	A341	Rajesh	Roshan	24	Male	1234567890	https://us05web....	30-11-0022	12
4	A501	Rajesh	Roshan	24	Male	1234567890	https://us05web....	01-12-0022	11
5	A176	chaitanya	rajeev	52	Male	7656789199	(null)	30-11-2022	14

Code:

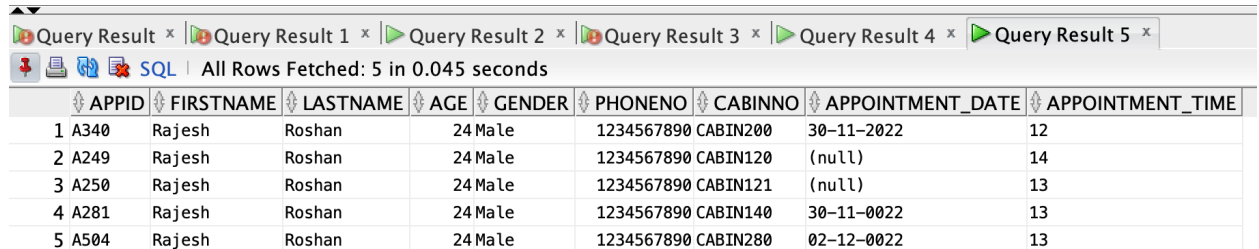
```
--OFFLINE APPOINTMENT
select a.appid, firstname, lastname, age, gender, phoneno,
cabinno, appointmentdate, appointmenttime
from appointment a
```

```

join patient p
on a.patientid = p.patientid
join offline_appointment oa
on a.appid = oa.appid
WHERE employeeid = 'prav';--employee id for doctor

```

Output:



Query Result 1 x Query Result 2 x Query Result 3 x Query Result 4 x Query Result 5 x

SQL | All Rows Fetched: 5 in 0.045 seconds

	APPID	FIRSTNAME	LASTNAME	AGE	GENDER	PHONENO	CABINNO	APPOINTMENT_DATE	APPOINTMENT_TIME
1	A340	Rajesh	Roshan	24	Male	1234567890	CABIN200	30-11-2022	12
2	A249	Rajesh	Roshan	24	Male	1234567890	CABIN120	(null)	14
3	A250	Rajesh	Roshan	24	Male	1234567890	CABIN121	(null)	13
4	A281	Rajesh	Roshan	24	Male	1234567890	CABIN140	30-11-0022	13
5	A504	Rajesh	Roshan	24	Male	1234567890	CABIN280	02-12-0022	13

Indexes

10. Query:

- Indexing on EmployeeID in Appointment table
- Hash Indexing on EmployeeID and PaymentDate

Concept: To reduce the retrieval time and faster execution. Our dataset is small but comes in very handy when there are hundreds of thousands of records.

Code:

```

- Indexing
-- Indexing EmployeeId in APPOINTMENT_WITH_ EMPLOYEE table
CREATE INDEX APPOINTMENT WITH_ EMPLOYEE ON APPOINTMENT
(employeeID);

```

```

-Hash Partitioning for Payslip table
CREATE INDEX Hash APPOINTMENT WITH EMPLOYEE ON Payslip
(employeeid, paymentdate)
GLOBAL PARTITION BY HASH (employeeID) PARTITIONS 2;

```

DDL

11. Query: Create patient bill table with relevant constraints

Concepts: Check, Constraint, foreign key referential integrity

Code:

```

Create table PATIENT_BILL_Dummy (
    pbId varchar2(4) constraint payment_pk PRIMARY KEY,
    pbStatus varchar2(4) constraint pbStatus_constraint check
(pbStatus in ('Paid','Pending',null)),
    billDate TIMESTAMP ,
    pbAmount number(10), constraint pbAmount_constraint check
(pbAmount >= 0),
    transactionType varchar2(10)constraint
transactionType_constraint check (transactionType in
('Card','Cash','Online','Check',null)),
    patientID varchar2(4),
    CONSTRAINT patientBill_fk FOREIGN KEY (patientID)
REFERENCES PATIENT (patientID)

);

```

12. Query: Create employee table with relevant constraints

Concepts: Check, Constraint, foreign key referential integrity

Code:

```

Create table employee (
    employeeID varchar2(4)
        constraint employee_PK PRIMARY KEY,
    firstName varchar2(20),
    lastName varchar2(20),
    age Number (2)
        constraint age_constraint
        check (age >= 0 or age is null),
    gender varchar2(6)
        constraint gender_constraint
        check (gender in ('Male', 'Female', 'Others', null, '
')),
    designation varchar2(10),
    experience number(2)
        constraint experience_constraint
        check (experience >=0 or experience is null),
    salary number(10)
        constraint salary_constraint
        check(salary >= 0 or salary is null),
    deptID varchar2(4)
        constraint emp_dept_FK REFERENCES department(deptID)
        ON DELETE SET NULL,
    email varchar2(40)
        constraint email_constraint

```

```

        CHECK ((email LIKE '%@%') or email is null),
type varchar2(40),
Area varchar2(40),
DateOfJoin date,
SSN number(8)
        constraint SSN_constraint
        check (SSN >= 10000000 or SSN in (null, 0))
);

```

Chapter 5

Triggers, Procedures and Functions

Triggers

ON_INSERT_EMPLOYEE

Purpose:

Employee is a superclass, with Medical_Staff and Non_Medical_Staff as subclasses (partition subclasses). As we have a new record in superclass, we need a record in subclasses as well.

Action:

When a new record is inserted into the Employee table, based on the type attribute in Employee table, a record with employeeID and specialization will be inserted into the Medical_Staff table or a record with employeeID and area will be inserted into the Non_Medical_Staff table.

Code:

```

create or replace TRIGGER on_insert_employee
    AFTER INSERT
    ON employee
    FOR EACH ROW

    BEGIN

        IF (:new.type = 'Medical Staff') THEN
            INSERT INTO medicalstaff (employeeid, specialization) VALUES
            (:new.employeeID , :new.area);

            dbms_output.put_line('Employee record inserted in employee
and medical staff table');
        ELSE

```

```

        INSERT INTO non_medical_staff VALUES (:new.employeeID ,
:new.area);
        dbms_output.put_line('Employee record inserted in employee
and non medical staff table');
    END IF;
END;

```

ON_INSERT_APPOINTMENT_TABLE

Purpose:

Compound trigger with operations before insert and after insert, on inserting a new record in the Appointment table, a patient can book an appointment with a doctor, available on date and time. When a patient tries to book an appointment, an insert statement on table Appointment runs.

Action:

Before Insert, this trigger will check if the doctor is available on that particular date and time, if available then the record will be inserted to the appointment table, if not raise an application error exception.

After inserting a new appointment in the appointment table, if the appointment type selected is offline then, a record with appId(appointment ID) and cabinID (cabin number for offline appointment) will be inserted into Offline_Appointment table. If the appointment type is online, then a record with appId and meetingLink(doctor's zoom link from medical_staff) will be inserted into the Online_Appointment table.

Code:

```

create or replace TRIGGER on_insert_appointment_table
FOR INSERT
ON APPOINTMENT
COMPOUND TRIGGER
v_check NUMBER;
BEFORE EACH ROW is
BEGIN
SELECT count(appid) into v_check from appointment
WHERE employeeID = :new.employeeID
    and appointment_Date = :new.appointment_Date
    and appointment_time = :new.appointment_time;

    IF (v_check>0) THEN
        dbms_output.put_line ('No appointments available for selected
date, time and doctor');
    
```

```

        raise_application_error(-20001,'Appointment is not available,
unable book appointment');
    END IF;
    END BEFORE EACH ROW;

    AFTER EACH ROW IS
        v_meetinglink online_appointment.meetinglink%type;
        v_cabinno offline_appointment.cabinno%type;

    BEGIN

        IF ( :new.type = 'Online') THEN
            SELECT meetinglink INTO v_meetinglink FROM medicalstaff
                WHERE employeeid = :new.employeeid;
            INSERT INTO online_appointment VALUES (:new.appid,
v_meetinglink);
        ELSE
            v_cabinno := concat('CABIN',cabin_no_sequence.NEXTVAL);
            INSERT INTO offline_appointment VALUES (:new.appid,
v_cabinno);
        END IF;
    END AFTER EACH ROW;
END on_insert_appointment_table;
/

```

Trigger_Patinet_and_Employee_NewUser

Purpose:

Application is built using Django, which creates a few default tables. When a new user registered from the web app, details are stored into the Auth_User table. With null values django cannot iterate on attributes.

Action:

Insert record into the Patient table or the Employee table based on an attribute v_is_staff in Auth_User(django tables). If v_is_staff is '0' insert record into patient and if v_is_staff is '1', insert record into employee table. After insertion, replace the null values with dummy values either 0 for number type attributes and a single space for varchar type attributes.

Code:

```

create or replace TRIGGER trigger_patient_and_employee_newuser
BEFORE INSERT OR UPDATE ON AUTH_USER
FOR EACH ROW
DECLARE
v_ID patient.patientID%type;
v_is_staff number(1);

```

```

BEGIN
    v_ID := :new.username;
    v_is_staff := :new.is_staff;

    if v_is_staff = 0 then
        INSERT into PATIENT (patientID) values (v_ID);
        Update patient set firstName = ' ', lastName = ' ', age
=0, gender=' ', phoneNo= 0, StreetName=' ', areaName=' ', city=' ',
state=' ', pincode=0, email = '@' where patientId = v_ID ;
        dbms_output.put_line ('New patient registered ''' || v_ID ||
''' updated details in patient table ');
    else
        INSERT into employee (employeeID) values (v_ID);
        Update employee set firstName = ' ', lastName = ' ', age
=0, gender=' ', designation = ' ',experience = 0, salary= 0, DeptID='
', email='@ ', type='Medical Staff', Area = ' ', dateOfJoin =
sysdate, SSN = 0 where employeeID = v_ID ;
        dbms_output.put_line ('New employee registered ''' || v_ID ||
''' updated details in employee table ');
    end if;

END;

```

Appointment_ID_Creation

Purpose:

A patient can book multiple appointments, each appointment ID should be unique

Action:

Upon insertion of a new record into the Appointment table, creating a new appointment ID using sequence and writing that value in the attribute appID in the Appointment table.

Code:

```

create or replace TRIGGER appointment_ID_creation
    BEFORE INSERT
    ON appointment
    FOR EACH ROW
    BEGIN
        :new.appid := concat('A',appointment_ID_sequence.NEXTVAL);
    END;

-- Sequence
CREATE SEQUENCE appointment_ID_sequence

```

```
START WITH 100
MAXVALUE 999
;
```

Procedures

Update_MeetingLink

Purpose:

Doctors can attend a patient either offline or online, for online a unique zoom link is created for each medical_staff

Action:

MedicalStaff table is updated with a meetingLink against each medical staff created from employeeid and zoom link url

Code:

```
create or replace procedure update_meetingLink  IS
  CURSOR employee_medicalStaff IS
    SELECT employeeid
      FROM medicalStaff ;
  BEGIN
    FOR m IN employee_medicalStaff LOOP
      update medicalStaff set meetinglink = concat
('https://zoom.us/join/',employeeID) where employeeid = m.employeeid;

    end loop;
  END;
```

Update_Employee_Experience

Purpose:

In employee table we store records of employee joining date for each employee and a derived attribute experience, which is calculated from joining_date attribute

Action:

From joining date attribute in the Employee table, calculating years of experience by taking the difference of joining date and sysdate , and updating the derived attribute experience attribute in years.

Code:

```

create or replace procedure Update_Employee_Experience  IS
  CURSOR Emp IS
    SELECT employeeID, dateOfJoin
      FROM Employee ;
  BEGIN
    FOR e IN Emp LOOP
      IF ( e.dateOfJoin is not null) THEN
        update employee set experience =
floor(months_between(sysdate , dateOfJoin)/12) where employeeID =
e.employeeID ;
        END IF;
      end LOOP;
    END;
  /

```

Type_Employee

Purpose:

If a record is inserted without a type(medical staff or non medical staff) attribute in the Employee table(superclass). We don't have records for the employee in the subclasses.

Action:

Based on the designation of the employee we are updating the type of employee in the Employee table as either Medical_Staff or Non_Medical_staff

Code:

```

create or replace procedure type_employee IS

  CURSOR employees IS
    select employeeID, designation
      from employee;

  BEGIN
    FOR emp IN employees LOOP
      IF (emp.designation = 'Doctor' OR emp.designation =
'Nurse' OR emp.designation ='Intern') THEN
        UPDATE employee SET type = 'Medical Staff' WHERE
employeeid = emp.employeeID;
      ELSE
        UPDATE employee SET type = 'Non Medical Staff' WHERE
employeeid = emp.employeeID;
      END IF;
    END LOOP;
  
```

END;

Functions

ENCRYPT_SSN

Purpose:

SSN of employee is stored in the Employee table, using dbms_crypt package from oracle DB we are encrypting SSN

Function: Encrypt_SSN encrypts the number passed to the function, input can be a number output type is raw. Encryption is done with AES 256 which is 32 bytes encryption with a 32 bytes key which is generated randomly.

Code:

```
CREATE or REPLACE Function ENCRYPT_SSN (ssn_input number)
RETURN RAW
IS
encrypted_raw raw(2000);
encryption_type pls_integer := DBMS_CRYPTO.ENCRYPT_AES256 +
DBMS_CRYPTO.CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5;

num_key_bytes      NUMBER := 256/8;
key_bytes_raw      RAW (32);
BEGIN
DBMS_OUTPUT.PUT_LINE ( 'Before encryption ' || ssn_input);
key_bytes_raw := DBMS_CRYPTO.RANDOMBYTES (num_key_bytes);
encrypted_raw := DBMS_CRYPTO.ENCRYPT
(
    src => UTL_I18N.STRING_TO_RAW (ssn_input, 'AL32UTF8'),
    typ => encryption_type,
    key => key_bytes_raw
);
RETURN encrypted_raw;
END;
/
```

Below, we are encrypting the number - 86878892 (8 digit random number), output shows the encrypted number.

	ENCRYPT_SSN(86878892)
1	BFC8AA8FC89DE4B9F7CED95201CA85EE

Chapter 6

User Interface

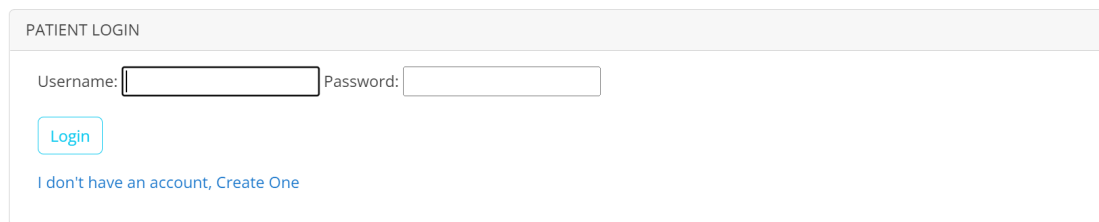
UI Walkthrough and test cases

Patient

1. Patient Login

This screen can be accessed at - <http://54.148.54.117/login/>

Wildcat Healthcare – Patient Login



PATIENT LOGIN

Username: Password:

[I don't have an account, Create One](#)

Fig.1 - Patient Login Screen

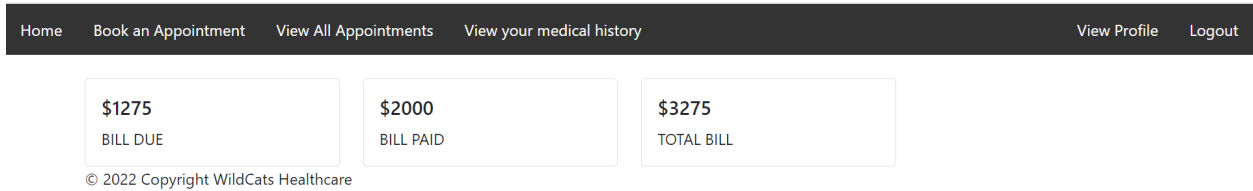
This is the first screen for a patient, from here he can either choose to create a new account or use existing login credentials. Each of these cases are covered in 1.1 (existing login credentials) and 1.2 (creating new account) respectively.

1.1 Existing credentials

Username : profcurrim

Password : Edmislove123

This should log you into the patient portal homepage shown below with the following bill due, bill paid and total bill values for prof. Currim as fetched from the database.



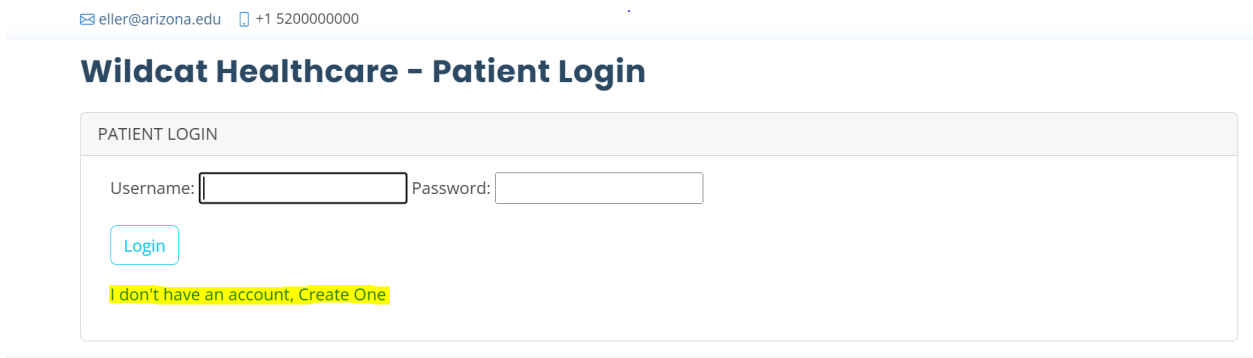
A screenshot of the Patient Portal Home page. At the top is a dark navigation bar with links: Home, Book an Appointment, View All Appointments, View your medical history, View Profile, and Logout. Below the navigation bar are three white boxes displaying bill information: \$1275 BILL DUE, \$2000 BILL PAID, and \$3275 TOTAL BILL. At the bottom left, there is a copyright notice: © 2022 Copyright WildCats Healthcare.

Fig.2 - Patient Portal Home

1.2 Creating a new account

This screen can be accessed at - <http://54.148.54.117/registerpatient/>

Alternative to using the existing login is to create a new account. This can be done by clicking the I don't have an account, Create one option on the patient login screen as shown below.



A screenshot of the Wildcat Healthcare - Patient Login page. At the top, there is a contact information bar with an email icon and 'eller@arizona.edu' and a phone icon and '+1 5200000000'. Below this is the title 'Wildcat Healthcare - Patient Login'. The main content area is titled 'PATIENT LOGIN' and contains a form with two input fields: 'Username:' and 'Password:'. Below the input fields is a blue 'Login' button. At the bottom of the form, there is a link that says 'I don't have an account, Create One'.

Fig.3 - Create a new account option

Clicking on this will take you to the registration page for patients as shown below.

Wildcat Healthcare – Patient Registration

NEW PATIENT REGISTRATION

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

SignUp

[Login, I already have an account](#)

Fig.4 - Patient Registration Page

Here you can pick your username and set your password. The username is restricted to 10 characters and the password follows all the rules seen on the registration page. To check if all the constraints are being checked let us manually try to violate a password check constraint. For this I will set the username and password exactly the same, which is in violation of one of the password rules, furthermore I will make sure it doesn't meet the minimum length requirement. Let us see what error the form throws. Below screenshot shows my entries, which are -

Username : kaka

Password : kaka

Confirm Password : kaka

Wildcat Healthcare – Patient Registration

NEW PATIENT REGISTRATION

Username*

kaka

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password*

....

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

....

Enter the same password as before, for verification.

SignUp

[Login, I already have an account](#)

Fig.4 - Intentionally violating the password requirements

The error yield will pinpoint to the portion where the user has made a mistake as shown below.

Wildcat Healthcare – Patient Registration

NEW PATIENT REGISTRATION

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

The password is too similar to the username.

This password is too short. It must contain at least 8 characters.

This password is too common.

Password confirmation*

Enter the same password as before, for verification.

Fig.5 - Real Time alert to the user after validation

2. Patient Home

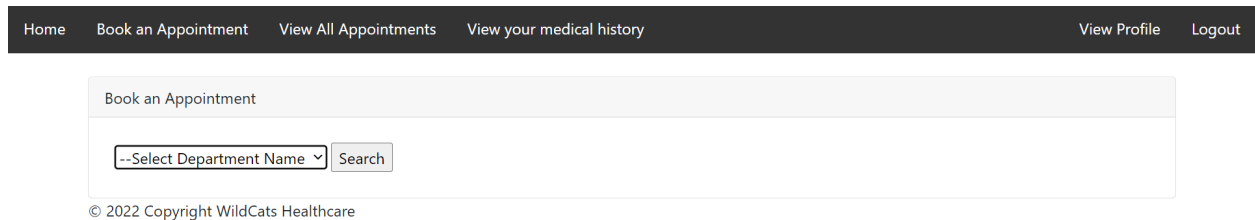
This screen requires the patient to be logged in first - <http://54.148.54.117/login/>

The patient home exists at - <http://54.148.54.117/>

Patient home comes with a variety of options including the ability to book new appointments, view all appointments, View medical history and View/Update his profile. Now I have logged in here for profcurrim user using the credentials discussed in section 1 (Patient Login), if you login with a newly registered user, he may not have any bills/appointments/history etc. But you can always create these - put in the profile info from the view profile tab, book an appointment etc.

2.1 Book a new appointment

Click on book a new appointment, this will take you to a page that shows you the list of departments with which a new appointment can be booked.



Home Book an Appointment View All Appointments View your medical history View Profile Logout

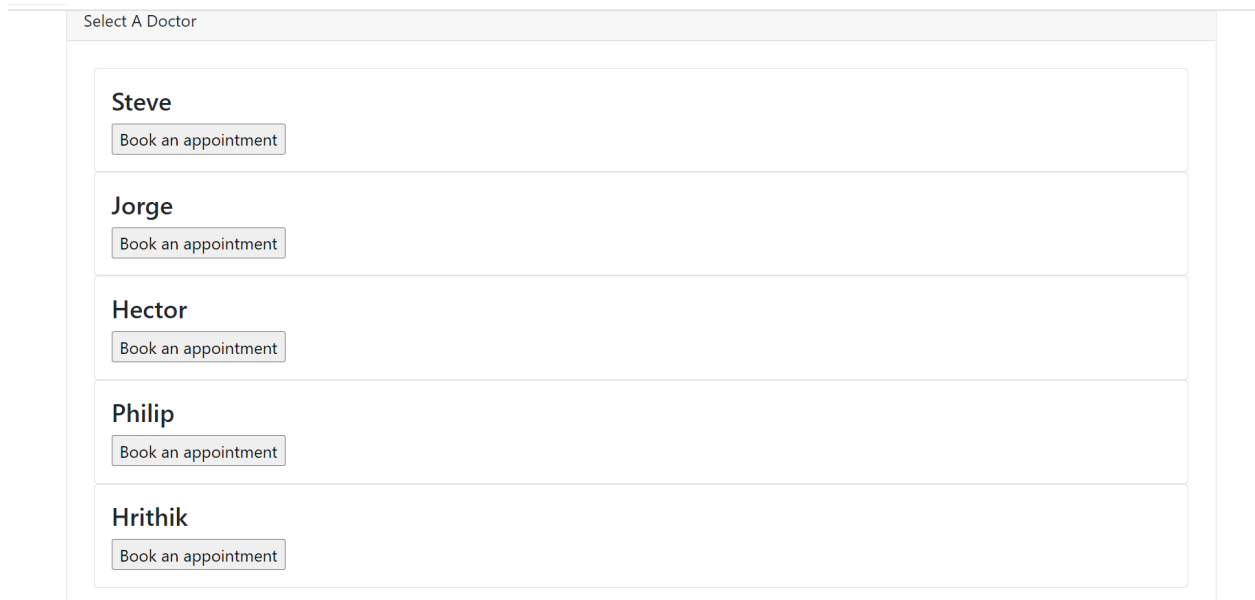
Book an Appointment

--Select Department Name Search

© 2022 Copyright WildCats Healthcare

Fig.6 - Select a department page

From here select a department of your choice, not all departments have doctors, hence we recommend the 'neurology department' since we are sure that there is a doctor against it. Furthermore we have this doctor's login on the doctor front created, so that he can view the appointments. Selecting the department will show you a list of available doctors in that department.



Select A Doctor

Steve
Book an appointment

Jorge
Book an appointment

Hector
Book an appointment

Philip
Book an appointment

Hrithik
Book an appointment

Fig.7 - Select a doctor

For the purposes of this walkthrough, I will select 'Hrithik', you are welcome to pick any one of your choice.

When I click on book an appointment, the next screen will allow me to pick a date and time.

Note: All dates are dynamically populated as sysdate + next 7 days, as we allow appointments to be booked at most 7 days in advance.

For the purposes of this demo, I will choose, 22/12/08 - 13:00 hrs - online appointment.
Note : Appointment for this date and time is booked and it will not allow another booking for this doctor on this date and time.

The screenshot shows a web application header with navigation links: Home, Book an Appointment, View All Appointments, View your medical history, View Profile, and Logout. Below the header is a form titled "Pick A Time". The form contains three dropdown menus for selecting the date (22/12/08), time (13), and mode (Online). A "Select" button is located below the dropdowns. At the bottom of the form, there is a copyright notice: "© 2022 Copyright WildCats Healthcare".

Fig.8 - Picking date, time and mode of appointment

Upon selecting the slot and submitting, this should redirect you to the appointment successful page.

The screenshot shows a web application header with navigation links: Home, Book an Appointment, View All Appointments, View your medical history, View Profile, and Logout. Below the header is a green confirmation message: "Successfully booked an appointment!". At the bottom of the page, there is a copyright notice: "© 2022 Copyright WildCats Healthcare".

Fig.9 - Successful appointment

This is followed by a mail to the patient's email id.

Now let us look for this appointment in the View all appointments tab. This shows all appointments including the one booked just now. One can use the zoom link from here to join the meeting.

Online Appointments

First Name	Last Name	Email	Meeting Link	Appointment Date	Appointment Time
Hrithik	Koduri	hrithik@gmail.com	Click here to join	02-DEC-0022	12:00 hrs
Hrithik	Koduri	hrithik@gmail.com	Click here to join	08-DEC-0022	13:00 hrs
Hrithik	Koduri	hrithik@gmail.com	Click here to join	02-DEC-0022	14:00 hrs
Hrithik	Koduri	hrithik@gmail.com	Click here to join	04-DEC-0022	11:00 hrs

Fig.10 - All appointments

Next is the medical history functionality. Right now Prof. Currim doesn't have any medical history against him. At a later point we will login as a patient who has some medical history against him. But for now the screen should look as shown below.

Home	Book an Appointment	View All Appointments	View your medical history	View Profile	Logout
------	---------------------	-----------------------	---------------------------	--------------	--------

Diagnosis	Known Disease
-----------	---------------

© 2022 Copyright WildCats Healthcare

Fig.11 - View medical history

Now the last thing here on the navbar is 'view profile' section, it should help you edit any of your profile related information.

Edit

Firstname:

Lastname:

Age:

Gender:

Phoneno:

Streetname:

Areaname:

City:

State:

Pincode:

Fig.12 - Profile

I will change the age here to 25 and click update. It should update and reload the form with updated details fetched from the database as shown below.

Edit

Firstname:

Lastname:

Age:

Gender:

Phoneno:

Streetname:

Areaname:

City:

State:

Pincode:

[Update](#)

© 2022 Copyright WildCats Healthcare

Fig.13 - Update Profile

3. Security features

3.1 User sessions

A user session ends as soon as you logout and you cannot go back to the previous session once you log out by pressing the back button. To test this feature let us first click logout, this should take you to the below screen.

eller@arizona.edu +1 5200000000

Wildcat Healthcare – Patient Login

PATIENT LOGIN

Successfully Logged Out!

[Click here to go back to the Login Screen](#)

Fig.14 - Logged Out Screen

Now if I press the back button you will notice that it takes me back to the login screen.

Note: Furthermore you will notice that the url shows additional information on where it should go once logged in '?next=/patientprofile/'. This is worth noting because this was the last screen you were on before you logged out and hence the screen you wanted to go back to when you pressed back. This information gets saved and passed on, so it doesn't go to the default home page.

3.2 URL protection

It can be often noticed that you can manually enter a URL that should require login and go there. For example patient profile or medical history or book appointment should not be accessible just by entering the path on url bar, these should only be accessible if the user is logged in otherwise they should be redirected to login screen, this feature has been implemented and can be tested out.

Doctor

2.Doctor Login

Heading over to this url : <http://54.148.54.117/employeelogin/>

Employee LOGIN

Login

Username: Password:

[Contact Admin for creation of new account](#)

Fig.15 - Employee Login

This is the first screen for an employee. He can login into his existing account, for creation of a new account the employee should contact admin.

2.1 Existing credentials

Username : hrithik

Password : Medicinproject@2022

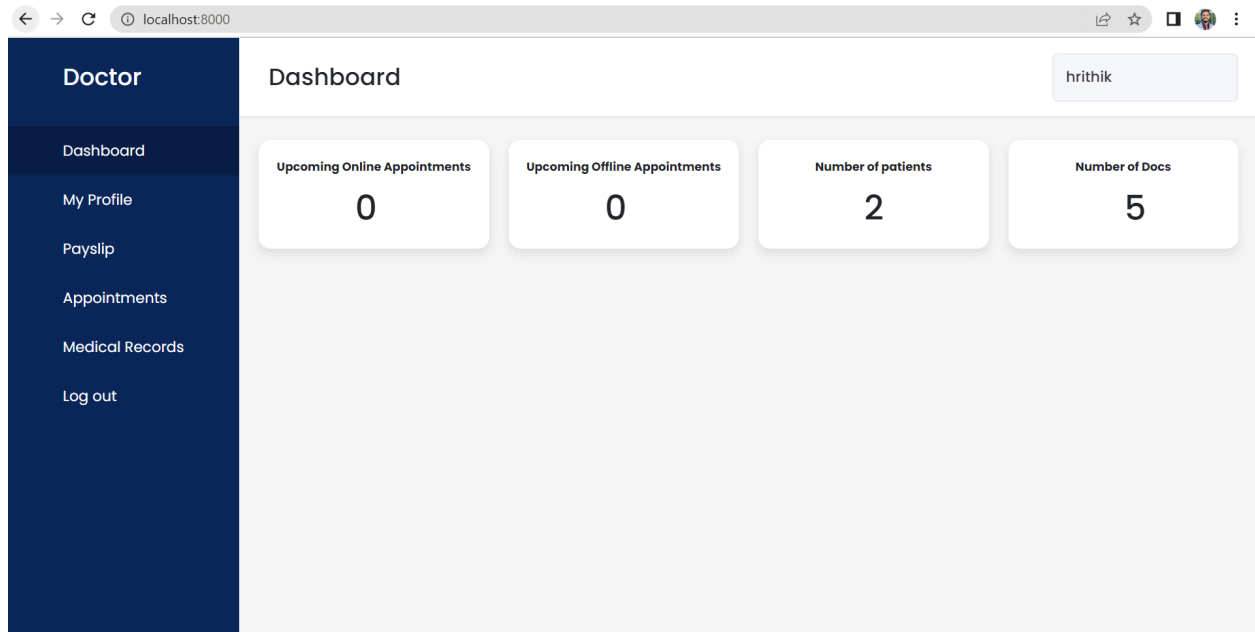


Fig.16 - Doctor Dashboard

This is the snapshot of the Doctor Login. He has functionalities to view his profile, see his payslip and appointment details and medical records of patients. In the dashboard it shows the Upcoming Online Appointments, Offline Appointments, Number of patients, Number of doctors in the same department.

2.1 Update Employee details

A screenshot of the "Employee Update Details" form. The sidebar is identical to the previous screenshot, but "My Profile" is now highlighted. The main content area has a header "Edit" and a form with three input fields: "Firstname" with the value "Hrithik", "Lastname" with the value "Koduri", and "Email" with the value "hrithik@gmail.com". Below the input fields is a blue "Update" button.

Fig.17 - Employee Update Details

Doctor can update his details like First name, Last name, Email. Here we are validating email with @. So if you type without @ it will throw an integrity error and it will not update in the database.

2.2 View his Payslip

Doctor	Dashboard	hrithik
Dashboard		
My Profile		
Payslip		
Appointments		
Medical Records		
Log out		

First Name	Last Name	Designation	Payment Date	Payment ID	Dept ID
Hrithik	Koduri	Doctor	10-MAY-2022	9000	D118
Hrithik	Koduri	Doctor	10-NOV-2022	10000	D118

Fig.18 - Employee Payslip

Can view his payslip over his employment duration in the organization.

2.3 View Upcoming Appointments

Doctor	Dashboard	hrithik
Dashboard		
My Profile		
Payslip		
Appointments		
Medical Records		
Log out		

Online Appointments					
First Name	Age	Gender	Zoom Link	Appointment Date	Appointment Time
Dr	16	Male	Zoom Link	02-DEC-0022	12
Dr	16	Male	Zoom Link	02-DEC-0022	14
Dr	16	Male	Zoom Link	04-DEC-0022	11
Rajesh	24	Male	Zoom Link	05-DEC-0022	12
Rajesh	24	Male	Zoom Link	01-DEC-0022	10
Rajesh	24	Male	Zoom Link	05-DEC-0022	13

offline Appointments					
First Name	Age	Gender	Cabin No	Appointment Date	Appointment Time

Fig.19 - Doctor Appointments

Doctors can view upcoming appointments. Different sections for different online and offline appointments. Online appointment details come with Zoom link and Offline appointments come with Cabin No.

2.4 Medical Records of Patient

The screenshot shows a web application interface for a doctor. On the left is a dark blue sidebar with the title 'Doctor' and a list of menu items: 'Dashboard', 'My Profile', 'Payslip', 'Appointments', 'Medical Records' (which is highlighted), and 'Log out'. The main content area has a header 'Dashboard' and a user profile 'hrithik'. Below the header is a section titled 'Search Patient Record' containing a search input field with the text '@ raj' and a 'Submit' button.

Fig.20 - Searching Patient Medical Records

Doctors can check the medical records of patients. Upon entering patient details. For example enter the username as “raj”

Upon checking raj patient records, the doctor gets a view like this :

The screenshot shows the same web application interface as Fig.20, but now displaying the medical records for the patient 'raj'. The sidebar and header are identical. The main content area is titled 'Medical Records of raj' and contains a table with two columns: 'DIAGNOSIS' and 'KNOWNDISEASE'. The table has 10 rows of data.

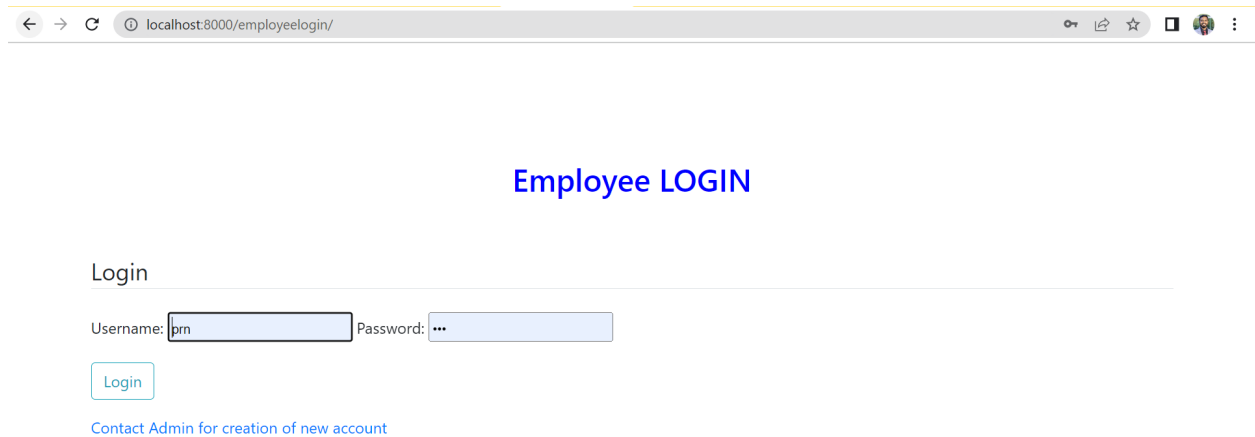
DIAGNOSIS	KNOWNDISEASE
Ultrasound_scan	Cirrhosis
check	check
nbv	bvn
check	check
xz	check
check	check
gdf	fg
check	check

Fig.21 - Patient Medical Records

3. Admin

3.1 Admin Login

Head over to this link : <http://54.148.54.117/employeelogin/>, After logging in as an employee, then head over to <http://54.148.54.117/adminstats/>.



The screenshot shows a web browser window with the address bar displaying 'localhost:8000/employeelogin/'. The page title is 'Employee LOGIN'. Below the title, there is a 'Login' section with a horizontal line. Under this line, there are two input fields: 'Username:' with the value 'prn' and 'Password:' with three dots. Below these fields is a 'Login' button. At the bottom of the login section, there is a link that says 'Contact Admin for creation of new account'.

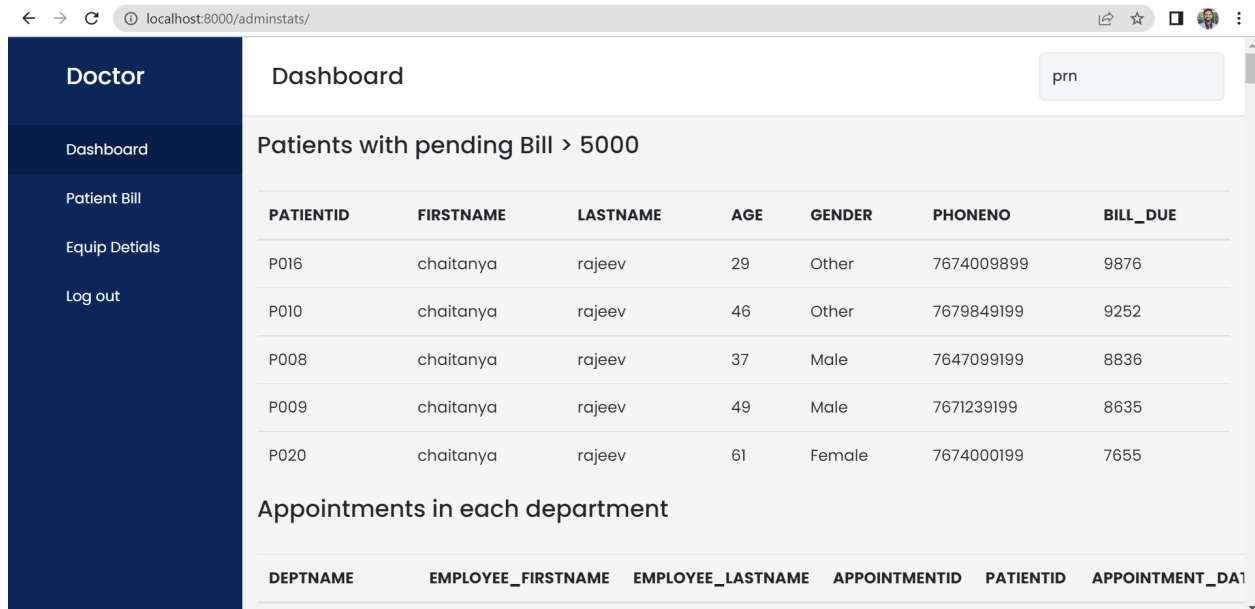
Fig.22 - Admin Login Screen

3.1 Existing credentials

Username : prn

Password : prn

After logging in, Head over to this link <http://54.148.54.117/adminstats/>



The screenshot shows a web application interface for an admin dashboard. On the left is a dark blue sidebar with the title 'Doctor' and a list of menu items: 'Dashboard', 'Patient Bill', 'Equip Details', and 'Log out'. The main content area has a header 'Dashboard' and a search bar containing 'prn'. Below the header, there is a section titled 'Patients with pending Bill > 5000' which contains a table with 7 columns: PATIENTID, FIRSTNAME, LASTNAME, AGE, GENDER, PHONENO, and BILL_DUE. The table lists 6 patients. Below this table is a section titled 'Appointments in each department' which shows the start of a table with columns: DEPTNAME, EMPLOYEE_FIRSTNAME, EMPLOYEE_LASTNAME, APPOINTMENTID, PATIENTID, and APPOINTMENT_DA1.

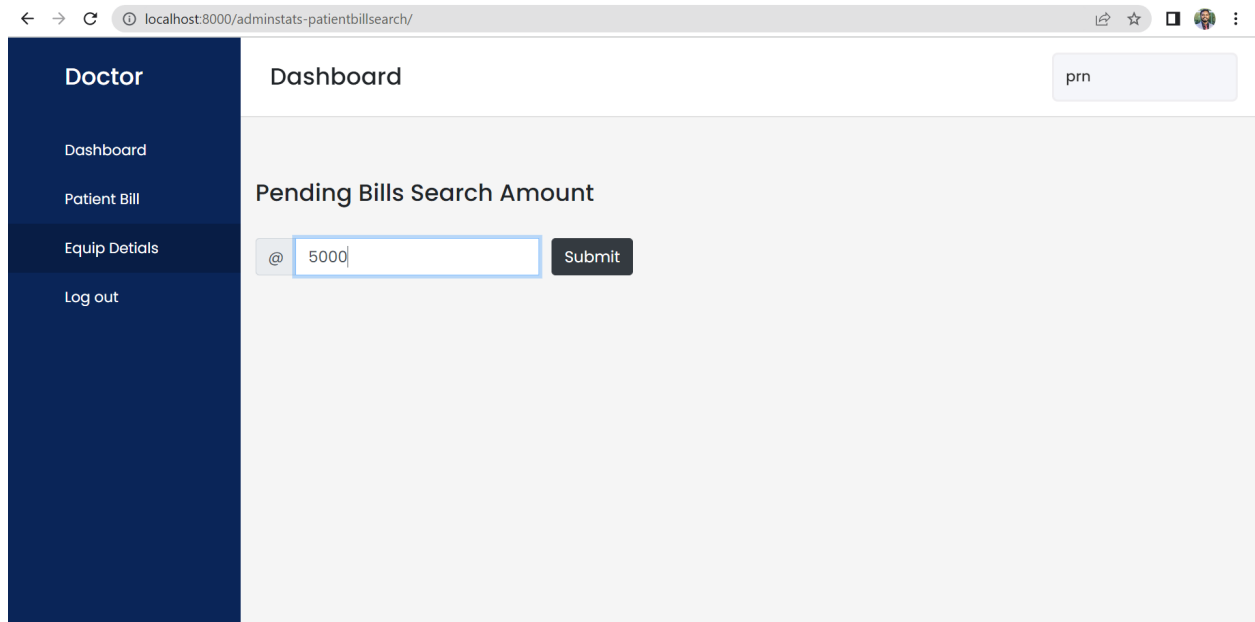
PATIENTID	FIRSTNAME	LASTNAME	AGE	GENDER	PHONENO	BILL_DUE
P016	chaitanya	rajeev	29	Other	7674009899	9876
P010	chaitanya	rajeev	46	Other	7679849199	9252
P008	chaitanya	rajeev	37	Male	7647099199	8836
P009	chaitanya	rajeev	49	Male	7671239199	8635
P020	chaitanya	rajeev	61	Female	7674000199	7655

DEPTNAME	EMPLOYEE_FIRSTNAME	EMPLOYEE_LASTNAME	APPOINTMENTID	PATIENTID	APPOINTMENT_DA1
----------	--------------------	-------------------	---------------	-----------	-----------------

Fig.23 - Admin Dashboard

This is the admin stats dashboard, He can view the entire data. The data covers Patient Bills, Equipment Details, Appointment in each department, Highest paid medical staff in each department.

3.2 Patient Bill Search



The screenshot shows a web application interface for a patient bill search. On the left is a dark blue sidebar with the title 'Doctor' and a list of menu items: 'Dashboard', 'Patient Bill', 'Equip Details', and 'Log out'. The main content area has a header 'Dashboard' and a search bar containing 'prn'. Below the header, there is a section titled 'Pending Bills Search Amount' which contains a search form with a text input field containing '5000' and a 'Submit' button.

Fig.24 - Patient Bill

Admin can view patient bills which are above any number that is given in the field. Any given number. Dynamically it will fetch a bill greater than 5000.

3.3 Equipment Details

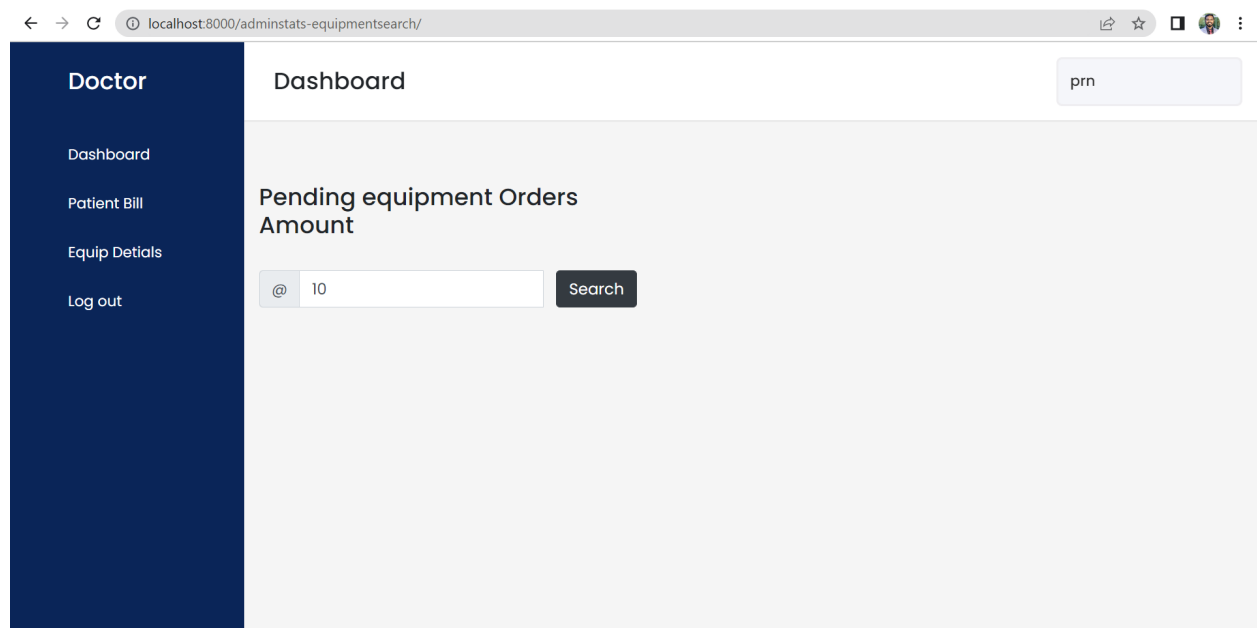


Fig.25 - Equipment Details

Admin can view equipment details which amount is greater than 10 or any given number. Dynamically it will fetch orders greater than 10.

Chapter 7

Implementation Plan

Steps :

1. Database creation with SQL & Data Insertion
2. Integration with current softwares
3. Front-End Creation + Integration
4. Testing (In addition to testing throughout process)
5. Connecting to the cloud
6. Training Employees
7. Maintenance

Purpose	Person Hours
Requirements Finalizing	7
Finalizing Stack & DB Connection	9
Database Creation	15
Backend SQL/PLSQL	24
Potential Delays / Hurdles	20
User Interface	15
Package Integration	17
Data Insertion	6
Training Peers	20
Connecting to Cloud	4
Maintenance	10
Testing	20
Total	167

Expenses	Amount	Notes
Cloud Provider (AWS)	\$4,975.00	
Storage (AWS)	\$960.00	\$80 per month * 12 months
Personnel	\$100,800	\$70 per hour for 160 hours total; 3 Developers + 3 testers + 1 database admin + 2 Project Manager
Processor License (Oracle)	\$17,500.00	
Software Update + License Support (Oracle)	\$3,850.00	
Additional Fees	\$20000	Food,space,recreation
Total \$	\$148,085	

Appendix A

Lessons Learnt

- We chose django as a framework for this project without learning about the framework in depth. As we started building we realized that most of django is centered around eliminating the use of SQL, and our purpose was to demonstrate knowledge in SQL, this meant we had to forego many advantages of the framework and its ORM features and write raw SQL queries. Had we known this beforehand we would have chosen XAMPP or another stack and we could have achieved similar results in half the time.
- Frontend takes time and requires a dedicated learning attempt.

- Oracle SQL Developer is easy to maintain Databases and for cloud deployments, but privileges as a user(student) are limited. Knowing the privileges upfront would definitely help, by not spending additional efforts to learn new packages, and other objects(Ex: Scheduler Objects, UTL packages)

Create Table Statements

DDL - Create EMPLOYEE table

```
Create table employee(
    employeeID varchar2(4)
    constraint employee_PK PRIMARY KEY,
    firstName varchar2(20),
    lastName varchar2(20),
    age Number (2)
    constraint age_constraint check (age >= 0 or age is null),
    gender varchar2(6)
    constraint gender_constraint check (gender in ('Male',
'Female', 'Others', null, ' ')),
    designation varchar2(10),
    experience number(2)
    constraint experience_constraint check (experience >=0 or
experience is null),
    salary number(10)
    constraint salary_constraint check(salary >= 0 or salary is
null),
    deptID varchar2(4)
    constraint emp_dept_FK REFERENCES department(deptID) ON DELETE
SET NULL,
    email varchar2(40)
    constraint email_constraint CHECK ((email LIKE '%@%') or email
is null),
    type varchar2(40),
    Area varchar2(40),
    DateOfJoin date,
    SSN number(8) constraint SSN_constraint check (SSN
>= 10000000 or SSN in (null, 0))
);
```

DDL - Create PATIENT table

```
Create table PATIENT(
```

```

    patientID varchar2(4) constraint patient_pk PRIMARY KEY,
    firstName varchar2(20),
    lastName varchar2(20),
    age Number (2),
    gender varchar2(6),
    email varchar2(50),
    streetName varchar2(10),
    areaName varchar2(10),
    city varchar2(10),
    state varchar2(10),
    pincode number(5),
    constraint email_constraint CHECK ((email LIKE '%@%') or email
is null),
);

```

DDL - Create HOSPITAL table

```

Create table HOSPITAL(
    hospitalID varchar(4) constraint app_pk PRIMARY KEY,
    hospitalName varchar2(10) constraint hname NOT NULL,
    streetName varchar2(10),
    areaName varchar2(10),
    city varchar2(10),
    state varchar2(10),
    pincode number(5)
);

```

DDL - Create APPOINTMENT table

```

Create table APPOINTMENT(
    appID varchar(4) constraint appid_pk PRIMARY KEY,
    appTime TIMESTAMP default current_timestamp
    appType varchar2 (20),
    hospitalID varchar(4),
    patientID varchar2(4),
    medicalstaffID varchar2(4),
    CONSTRAINT hospital_fk FOREIGN KEY (hospitalID) REFERENCES
HOSPITAL (hospitalID),
    CONSTRAINT patient_fk FOREIGN KEY (patientID) REFERENCES PATIENT
(patientID),
    CONSTRAINT patient_staff_fk FOREIGN KEY (medicalstaffID)
REFERENCES medicalstaff(employeeID)
);

```

DDL - Create ONLINE_APPOINTMENT table

```

Create table ONLINE_APPOINTMENT(
    appID varchar2(4) constraint online_appID_pk PRIMARY KEY,
    constraint online_app_fk FOREIGN KEY (appID) REFERENCES
    APPOINTMENT(appID),
    meetingLink varchar2(100)
);

```

DDL - Create PAYSLIP table

```

CREATE TABLE payslip(
    paymentID varchar2(4),
    amount number(20),
    paymentDate date,
    employeeID varchar2(4),
    constraint payslip_pk primary key (paymentID),
    constraint payslip_fk_employee foreign key (employeeID)
    references employee(employeeID)
);

```

DDL - Create PAIENT_BILL table

```

Create table PATIENT_BILL (
    pbId varchar2(4) constraint payment_pk PRIMARY KEY,
    pbStatus varchar2(4),
    billDate TIMESTAMP ,
    pbamount number(10)
    transactionType varchar2(10),
    patientID varchar2(4),
    CONSTRAINT patientBill_fk FOREIGN KEY (patientID) REFERENCES
    PATIENT (patientID)
);

```

DDL - Create DEPARTMENT table

```

Create table DEPARTMENT(
    deptID varchar2(4) constraint dept_pk PRIMARY KEY,
    deptName varchar2(10),
    branch varchar2(10)
);

```

DDL - Create MDEICAL STAFF table

```

Create table MEDICALSTAFF (
    employeeID varchar2(4) constraint ms_pk PRIMARY KEY,

```

```
        specialization varchar2(20),
        constraint ms_fk FOREIGN KEY (employeeid) REFERENCES
EMPLOYEE(employeeID)
);
```

DDL - Create NON MEDICAL STAFF table

```
CREATE TABLE non_medical_staff(
    employeeID varchar2(4),
    workType varchar2(50),
    constraint non_medical_staff_pk primary key (employeeID),
    constraint non_medical_staff_fk_employee foreign key (employeeID)
references employee(employeeID)
)
```

DDL - Create EQUIPMENT table

```
Create table EQUIPMENT (
    prodID varchar2(4) constraint prodID_pk PRIMARY KEY,
    manufacturingDate date,
    constraint prodID_fk FOREIGN KEY (prodID) REFERENCES
PRODUCT(prodID)
);
```

DDL - Create MEDICAL_RECORD table

```
Create table MEDICAL_RECORD (
    recordID varchar2(4) constraint recordID_pk PRIMARY KEY,
    recordDate date,
    diagnosis varchar2(100),
    knownDisease varchar2(100),
    patientID varchar2(4),
    constraint patientID_fk4 FOREIGN KEY (patientID) REFERENCES
PATIENT(patientID)
);
```

DDL - Create OFFLINE_APPOINTMENT table

```
Create table OFFLINE_APPOINTMENT (
    appID varchar(4) constraint oapp_pk PRIMARY KEY,
    cabinNo varchar2(30),
    constraint oapp_fk FOREIGN KEY (appid) REFERENCES
appointment(appID)
);
```

DDL - Create APPOINTMENT_WITH_MEDICALSTAFF table

```
Create table APPOINTMENTS_WITH_MEDICALSTAFF (  
    appID varchar(4),  
    employeeID varchar2(4),  
    constraint appid_medical FOREIGN KEY (appid) REFERENCES  
appointment(appID),  
    constraint emp_medical FOREIGN KEY (employeeID) REFERENCES  
employee(employeeID),  
    CONSTRAINT PK_Appointments_with_medicalstaff PRIMARY KEY  
(appID,employeeID)  
);
```

DDL - Create MEDICALSTAFF_ATTENDS_PATIENT table

```
Create table MEDICALSTAFF_ATTENDS_PATIENT (  
    patientID varchar2(4),  
    employeeID varchar2(4),  
    constraint patientID_medical FOREIGN KEY (patientID) REFERENCES  
patient(patientid),  
    constraint emp_medicalstaff FOREIGN KEY (employeeID) REFERENCES  
employee(employeeID),  
    CONSTRAINT PK_MEDICALSTAFF_ATTENDS_PATIENT PRIMARY KEY  
(patientID,employeeID)  
);
```

DDL - Create DIAGNOSIS table

```
CREATE TABLE diagnosis(  
    diagID varchar2(4) constraint diagnosis_pk PRIMARY KEY,  
    name varchar2(50),  
    appID varchar2(4),  
    constraint app_diagnosis FOREIGN KEY (appID) REFERENCES  
appointment(appID)  
);
```

DDL - Create PAITENT_MEDICINES_ORDERS table

```
CREATE TABLE patient_medicines_orders(  
    patientID varchar2(4),  
    serviceID varchar2(4),  
    orderID varchar2(4),  
    customerName varchar2(50),  
    billAmount number(20,2),  
    medicineName varchar2(50),
```

```

        quantity number(10),
        paymentType varchar2(50),
        constraint patient_medicines_orders_pk primary key (patientID),
        constraint patient_medicines_orders_fk_patient foreign key
(patientID) references patient(patientID),
        constraint patient_medicines_orders_fk_pharmacy foreign key
(serviceID) references pharmacy(serviceID)
    );

```

DDL - Create PHARMACY table

```

CREATE TABLE pharmacy(
    serviceID varchar2(4),
    hospitalID varchar2(4),
    constraint pharmacy_pk primary key (serviceID),
    constraint pharmacy_fk_services foreign key (serviceID)
references service(serviceID)
constraint pharmacy_fk_hospital foreign key (hospitalID) references
hospital(hospitalID)
);

```

DDL - Create SERVICE table

```

CREATE TABLE service(
    serviceID varchar2(4),
    serviceType varchar2(10),
    constraint service_pk primary key (serviceID)
);

```

DDL - Create PROCEDURE table

```

CREATE TABLE procedure(
    serviceID varchar2(4),
    name varchar2(25),
    cost number(20,2),
    constraint procedure_pk primary key (serviceID),
    constraint procedure_fk_service foreign key (serviceID)
references service(serviceID)
);

```

DDL - Create TREATMENT table

```

CREATE TABLE treatment(
    treatmentID varchar2(4),

```

```

        name varchar2(100),
        appID varchar2(4),
        serviceID varchar2(2),
        constraint treatment_pk primary key (treatmentID),
        constraint treatment_fk_service foreign key (serviceID)
references service(serviceID),
        constraint treatment_fk_appointment foreign key (appID)
references appointment(appID)
    );

```

DDL - Create LAB table

```

CREATE TABLE lab_test(
    labID varchar2(4),
    name varchar2(20),
    constraint lab_pk primary key (labID)
);

```

DDL - Create PURCHASE_ORDER table

```

CREATE TABLE purchase_order(
    purchaseID varchar2(4),
    purchaseDate date,
    brand varchar2(10),
    description varchar2(20),
    vendorID varchar2(4),
    serviceID varchar2(4),
    hospitalID varchar2(4),
    constraint purchase_order_pk primary key (purchaseID),
    constraint purchase_order_fk_vendor foreign key (vendorID)
references vendor(vendorID),
    constraint purchase_order_fk_service foreign key (serviceID)
references service(serviceID),
    constraint purchase_order_fk_hospital foreign key (hospitalID)
references hospital(hospitalID)
);

```

DDL - Create VENDOR table

```

CREATE TABLE vendor(
    vendorID varchar2(4),
    vendorName varchar2(10),
    constraint vendor_pk primary key (vendorID)
);

```


DDL - Create MEDICINE table

```
CREATE TABLE medicine(  
    prodID varchar2(4),  
    price number(5,2),  
    expDate date,  
    constraint medicine_pk primary key (prodID),  
    constraint medicine_fk_product foreign key (prodID) references  
product(prodID)  
);
```

DDL - Create PRODUCT table

```
CREATE TABLE product(  
    prodID varchar2(4),  
    prodName varchar(25),  
    prodtype varchar2(10),  
    hospitalID varchar2(4)  
    manufacturingDate date,  
    constraint product_pk primary key (prodID)  
    constraint hospital_fk foreign key(hospitalID)  
HOSPITAL(hospitalID)  
);
```

DDL - Create ORDER_DETAIL table

```
CREATE TABLE order_detail(  
    prodID varchar2(4),  
    purchaseID varchar2(4),  
    orderId varchar2(4),  
    unitPrice number(5),  
    units number(5),  
    constraint order_detail_pk primary key (prodID, purchaseID,  
orderId),  
    constraint order_detail_fk_purchase_order foreign key  
(purchaseID) references purchase_order(purchaseID),  
    constraint order_detail_fk_product foreign key (prodID)  
references product(prodID)  
);
```

DDL - Create APPOINTMENT_SUGGEST_DIAGNOSIS table

```
CREATE TABLE appointment_suggest_diagnosis(  

```

```

        appID varchar2(4),
        diagID varchar2(4),
        constraint appointment_suggest_diagnosis_pk primary key
(appID,diagID),
        constraint appointment_suggest_diagnosis_fk_diagnosis foreign key
(diagID) references diagnosis(diagID),
        constraint appointment_suggest_diagnosis_fk_appointment foreign
key (appID) references appointment(appID)
);

```

DDL - Create PAITENT_GETS_TEST table

```

CREATE TABLE patient_gets_test(
    patientID varchar2(4),
    testID varchar2(4),
    constraint patient_gets_test_pk primary key (patientID,testID),
    constraint patient_gets_test_fk_patient foreign key (patientID)
references patient(patientID),
    constraint patient_gets_test_fk_test foreign key (testID)
references test(testID)
);

```

DDL - Create VENDOR_SUPPLIESTO_HOSPITAL table

```

CREATE TABLE vendor_suppliesto_hospital(
    vendorID varchar2(4),
    hospitalID varchar2(4),
    constraint vendor_suppliesto_hospital_pk primary key
(vendorID,hospitalID),
    constraint vendor_suppliesto_hospital_fk_patient foreign key
(vendorID) references vendor(vendorID),
    constraint vendor_suppliesto_hospital_fk_test foreign key
(hospitalID) references hospital(hospitalID)
);

```

DDL - Create HOSPITAL_PROVIDES_SERVICES table

```

CREATE TABLE hospital_provides_services(
    hospitalID varchar2(4),
    serviceID varchar2(4),
    constraint hospital_provides_services_pk primary key (hospitalID,
serviceID),
    constraint hospital_provides_services_fk_hospital foreign key
(hospitalID) references hospital(hospitalID),

```

```
        constraint hospital_provides_services_fk_service foreign key
(serviceID) references service(serviceID)
);
```

DDL - Create PURCHASE_ORDER_DETAILS table

```
CREATE TABLE PRODUCT_ORDER_DETAILS(
    orderID varchar2(20),
    prodid varchar2(4),
    vendorid varchar2(4),
    quantity NUMBER(4),
    purchaseDate date,
    unitPrice Number(5,2),
    status varchar2(10)
    constraint prod_order_id PRIMARY KEY (orderID, prodid, vendorid)
    constraint prodid_fk foreign key(prodid) references
PRODUCT(prodid)
    constraint vendorid_fk foreign key(vendorid) references
VENDOR(vendorid)
);
```

DDL - Create EMERGENCY_SELLS_MEDICINE table

```
CREATE TABLE PHARMACY_SELLS_MEDICINE(
    pharmacyID varchar2(20),
    medicineID varchar2(4),
    constraint sells_pk PRIMARY KEY (pharmacyID, medicineID),
    constraint pharmacy_sells_fk foreign key(pharmacyID) references
PHARMACY(serviceid),
    constraint medicine_sells_fk foreign key(medicineID) references
MEDICINE(prodid)
);
```

DDL - Create SERVICE_DETAILS table

```
CREATE TABLE SERVICE_DETAILS(
    serviceID varchar2(4),
    patientID varchar2(4),
    serviceReferenceID varchar2(20),
    cost NUMBER(4,2),
    constraint serviceDetails_pk PRIMARY KEY (serviceID, patientID,
serviceReferenceID),
    constraint service_used_fk foreign key(serviceID) references
SERVICE(serviceid),
```

```
        constraint patient_usingService_fk foreign key(patientID)
PATIENT(patientid)
);
```

Peer Reviews (Presentation)

I especially liked

- Advanced technical aspects in the project, all Triggers and procedures explained well
- Compound Triggers usage was good and front-end login was great with sessions, Covered all aspects of the database. Implemented suggestions by Prof. Currim
- ER diagram explanation was clean and straightforward, Pallavi did a great job explaining it, she knows her target audience and has spoken to their tech-wavelength.
- Everyone dressed accordingly with business professionals, everyone was well prepared and spoke confidently.
- Queries were significantly complex that contained multiple WITH clauses, pivots, fetch, and more. The team also showed their indexes and explained the benefits of using them. Even though it wasn't working during the demo, sending the email with the zoom link was a very nice functionality and displays significant business value. The demo was a nice touch and I like how they showed the dashboard from both the patient and doctor end.
- Great use of indexes. Used compound triggers, great use! Team did a great job at explaining triggers, it was easy to follow. Good highlight of handling null values.
- The team efficiently explained the triggers and procedures. Loved how they worked on the future scope of their project as well. The team managed to do so well in such a short amount of time. Highly impressed.
- The website functionality was fulfilling to the project with all its features. Questions were answered well.

Areas of improvement

- Minor recommendation: add slide page number to your slides, so it's easier to reference during Q&A (e.g., "I have a question on slide 7" vs. "I have a question on your DDL slide")
- More structured presentation on the front end will be better since this part is good. Time management, too much unrelated details.
- Discussed about a few ER diagram instances but posted a big chunk of ER. Was not visible from the back