College of Saint Benedict & Saint John's University
Computer Science Department

CSCI 331 Project Phase II

Group 3

Team Members: Matt DeRosa, Max O'Brien, Ellie Smith, Mason Meyer, and Evan Quinn

April 4, 2024

Table of Contents

(0) An updated table showing the proposed system functionalities..……...……...……3-7

(1) Conceptual database design–High-level data modeling via ER/EER diagram.....8

(2) Logical database design–Relational Schema Diagram and description.....9-11

(3) Physical database design–Oracle mapping.....12-19

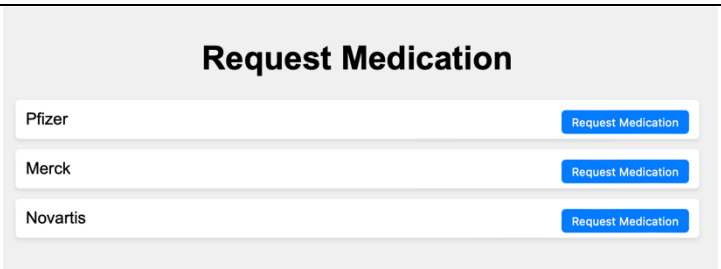
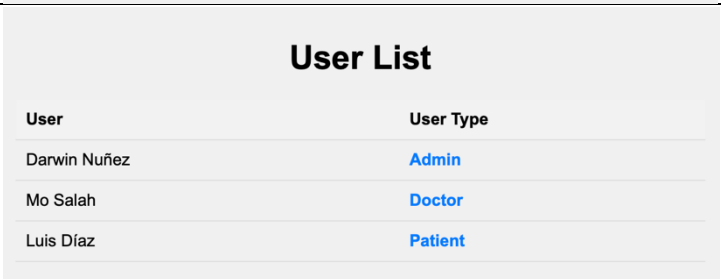
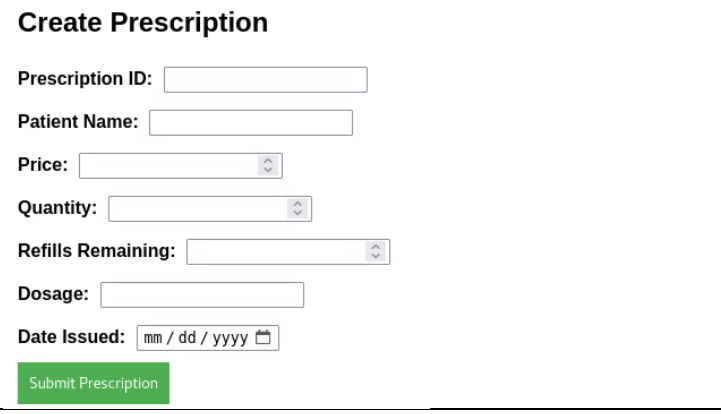
(4) SQL Routines & (5) Data Processing.....20-34

(0) An updated table showing the proposed system functionalities:

Proposed Functionality	Member Responsible	Brief Description	Sample User Interface with data included
PATIENT: Create Account	Matt	Allows new user to create an account on the management database	**Create New User** Date of Birth: mm/dd/yyyy Last Name: First Name: Email: Phone Number: Street: City: State: Zip Code: Sex: Female Insurance ID: Create Patient
PATIENT: Login	Matt	Allows users to login to the software and will direct them to a page depending on what type of user they are	**Login** PatientId or Email: Password: Login
PATIENT: View/edit profile	Matt	Allow patients to view and edit personal information including insurance information and their primary doctor. Patient Id, DOB, Last, and First and not able to be edited.	**Edit Patient Information** Patient ID: p08148394 Date of Birth: 12/25/1999 Last Name: Doe First Name: Jane Email: testemail@gmail.com Phone Number: 123-123-1234 Street: 12345 Junegrass Ln City: Cityville State: ST Zip Code: 12345 Sex: Female Insurance ID: 1234567890 Save Changes

<table>
<tr><td>PATIENT: View Appointment Info</td><td>Matt</td><td>Allows the patient to see the doctor ID number, patient ID number, date of consultation, doctor’s notes (in patient description)</td><td>View Appointments
<table><tr><th>Doctor's Name</th><th>Date</th><th>Notes</th></tr><tr><td>Dr. Smith</td><td>2024-04-15</td><td>Follow-up appointment. Patient is responding well to medication.</td></tr><tr><td>Dr. Johnson</td><td>2024-04-20</td><td>Annual check-up scheduled.</td></tr></table></td></tr>
<tr><td>PATIENT: View Diagnoses</td><td>Max</td><td>Patients can view diagnoses that doctors have added to their profile.</td><td>Diagnoses
Common cold
Acute sinusitis

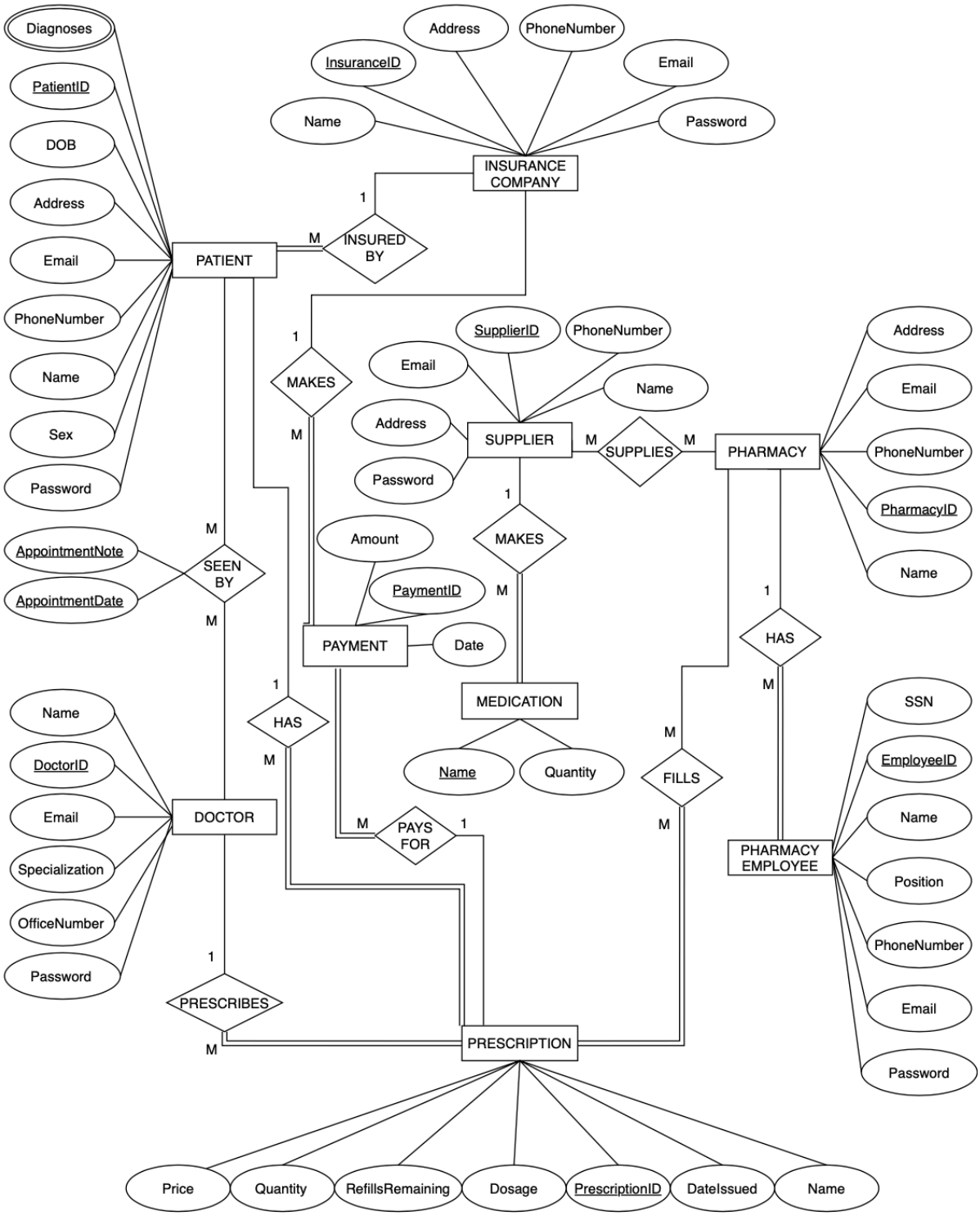
**patients will see this from a button on their profile</td></tr>
<tr><td>PATIENT: See prescription history</td><td>Max</td><td>Allow patients to see their prescriptions history and payment status for all of their prescriptions.</td><td>Prescription History
<table><tr><th>Prescription ID</th><th>Date Issued</th><th>Name</th><th>Dosage</th><th>Refills Remaining</th><th>Price</th><th>Quantity</th></tr><tr><td>P123456</td><td>2024-03-25</td><td>Medicine A</td><td>10mg</td><td>2</td><td>$20.00</td><td>30</td></tr><tr><td>P789012</td><td>2024-03-30</td><td>Medicine B</td><td>20mg</td><td>1</td><td>$30.00</td><td>20</td></tr></table></td></tr>
<tr><td>SUPPLIER: View/edit profile</td><td>Max</td><td>Allows suppliers to update their profiles—including the drugs they offer.</td><td>View/Edit Profile
Name: Pfizer
ID: 123456789
Email: john.doe@example.com
Phone Number: 123-456-7890
Drugs: Medication A, Medication B
Address: 123 Main St, City, State, 12345
Edit Profile</td></tr>
<tr><td>PHARMACY EMPLOYEE: View Inventory</td><td>Evan</td><td>View all drugs currently available in the pharmacy.</td><td>Medications Available
Allegra
Claritan
Xyzal</td></tr>
<tr><td>PHARMACY EMPLOYEE: Bill Patient/Insurance</td><td>Evan</td><td>Allows pharmacy employees to request a payment from insurance.</td><td>Request Payment
Evan Request Payment
Ellie Request Payment
Matt Request Payment</td></tr>
</table>

<table>
<tr><td>PHARMACY EMPLOYEE: Request medication from supplier</td><td>Evan</td><td>When a pharmacy is low, they have the ability to request more medication from their supplier</td><td> 
</td></tr>
<tr><td>PHARMACY EMPLOYEE: View medicine info</td><td>Evan</td><td>Allow pharmacy employees to view prescription information.</td><td> 
</td></tr>
<tr><td>HOSPITAL ADMIN: View & manage users</td><td>Max</td><td>See a list of all users and their types of users in the database and add users.</td><td> 
</td></tr>
<tr><td>DOCTOR: Submit prescription request</td><td>Mason</td><td>Allows doctors to create prescriptions for their patients.</td><td> 
</td></tr>
</table>

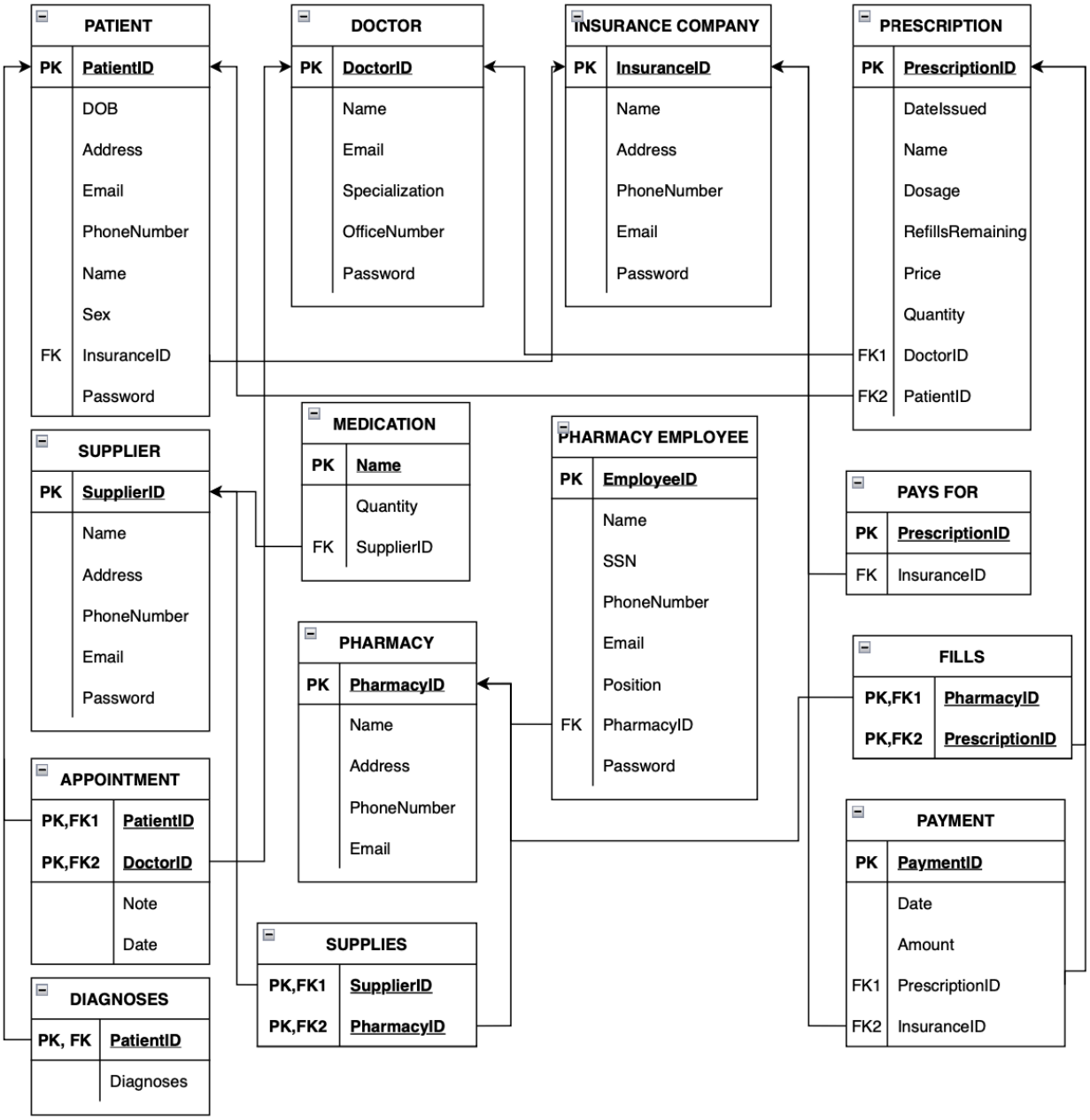
DOCTOR: View patient info	Mason	Allow doctor users to view certain information from their patients' profiles and view their diagnoses.	**Patient Information** **Patient ID:** 12345 **Name:** John Doe **Sex:** Male **DOB:** 01/01/1980 **Address:** 123 Main St, Anytown, USA **Email:** johndoe@example.com **Phone Number:** 555-1234 **Diagnoses:** Condition A, Condition B
DOCTOR: Add appointment note	Mason	Allows doctors to leave appointment note and date after seeing a patient.	**Appointment Notes** **Date:** mm / dd / yyyy **Notes:** Save Notes
DOCTOR: Edit patient diagnoses.	Mason	Allows doctors to edit diagnosis for a patient.	**Add Diagnosis** **Patient ID:** **Patient Name:** **Diagnosis:** **Date of Diagnosis:** mm / dd / yyyy Submit Diagnosis
DOCTOR: View/edit profile	Ellie	Allows doctors to view their entire profile and update everything except their ID.	**Doctor Profile** Name: Dr. John Smith Email: john.smith@example.com Specialization: Cardiologist Office Number: Room 101 Update

<table>
<tr><td>INSURANCE COMPANY: View patients they cover and how much each owes</td><td>Ellie</td><td>Insurance company can view a list of patients they cover along with their unpaid prescription balance.</td><td>Patients with Unpaid Prescription Balances
<table><tr><th>Name</th><th>Patient ID</th><th>Unpaid Prescription Balance</th></tr><tr><td>John Doe</td><td>P123456</td><td>$50.00</td></tr><tr><td>Jane Smith</td><td>P789012</td><td>$30.00</td></tr></table></td></tr>
<tr><td>INSURANCE COMPANY: Pay balance</td><td>Ellie</td><td>Insurance company can pay bills for patients. They have the ability to enter patient ID and prescription ID to see balance and can then make a payment.</td><td>Insurance Payment
Patient ID:
Prescription ID:
Check Unpaid Balance
Enter Payment Amount:
Make Payment</td></tr>
<tr><td>INSURANCE COMPANY: View/edit profile</td><td>Ellie</td><td>Allows insurance company to view their entire profile and update everything except their ID.</td><td>Insurance Company Profile
Insurance ID:
INS123456
Name:
Example Insurance Company
Address:
123 Main Street
Phone Number:
123-456-7890
Email:
info@exampleinsurance.co
Update</td></tr>
</table>

(1) Conceptual database design–High-level data modeling via ER/EER diagram



(2) Logical database design–Relational Schema Diagram and description



In our relational ER mapping, we chose to create twelve different tables (bolded below) to incapsulate our database system. A description of the database connections are below.

PATIENT entity:

- We decided to use the PatientID as its primary key because it is an unique attribute for each patient.
- For simplicity, we chose to limit our database to insured patients only. Although this does not represent the true hospital pharmacy database, we were unsure how to handle payments if not all patients were insured.
    - Thus, the relationship has total participation from the patient side resulting in the FK approach for the relational map. The insurance company's PK is then added to the **PATIENT** table as a FK.
- Since many patients can be seen by many doctors, we chose to map this relationship by using another lookup relation approach to reduce redundancy. This relationship between doctor and patient creates a new table "**APPOINTMENT**" where the combination of the FKs and PKs from Patient and Doctor are used as the primary keys. This relationship also has attributes date and notes from the appointment.
- The last relationship stemming from the patient entity is its relationship to the **PRESCRIPTION** entity. Patients may have many prescriptions. Thus, the patient ID was added as a FK the **PRESCRIPTION** table.
- A patient can have multiple diagnoses. This multivalued attribute was

DOCTOR entity:

- We decided to use the DoctorID as its primary key because it is a unique attribute for each doctor.
- As explained in our patient entity mapping, many doctors can see many patients. This results in a lookup relation approach to reduce redundancy. This relationship between doctor and patient creates the APPOINTMENT table.
- A doctor can prescribes many prescriptions. In order to avoid nulls in this relationship we chose to map this using the FK approach. Thus, the doctor ID was added as a FK the **PRESCRIPTION** table.

PHARMACY entity:

- We decided to use the PharmacyID as its primary key because it is a unique attribute for each pharmacy.
- A pharmacy can have many employees. This relationship has total participation from the many employee side resulting in the FK approach. The pharmacy's PK is then added to the **PHARMACY EMPLOYEE** table as a FK.
- Many pharmacies can be supplied by many medication **SUPPLIER**s. This resulted in a new look up relation table called **SUPPLIES** which includes the combination of FKs from **PHARMACY** and **SUPPLIER** as the PK.
- Many pharmacies fill many prescriptions. This resulted in a new look up relation table called **FILLS** which includes the combination of FKs from **PHARMACY** and **PRESCRIPTIONS** as the PK.

SUPPLIER entity:

- We decided to use the SupplierID as its primary key because it is a unique attribute for each supplier.
- Suppliers can make many **MEDICATIONS**. This relationship has total participation on the MEDICATION side which results in a FK of SupplierID in the **MEDICATION** table.

INSURANCE COMPANY entity:

- We decided to use the InsuranceID as its primary key because it is a unique attribute for each company.

PRESCRIPTION entity:

- We decided to use the PrescriptionID as its primary key because it is a unique attribute for each prescription.
- Prescriptions are paid for by the patient's **INSURANCE COMPANY** because every patient in the database is insured. This is done by the **INSURANCE COMPANY** making a **PAYMENT** which must pay for a single prescription. A prescription can be paid for with multiple payments.

PAYMENT:

- We decided to use the PaymentID as its primary key because it is a unique attribute for each Payment
- Because **PAYMENTS** relationships with **INSURANCE COMPANY** and **PRESCRIPTION** were total participation on the **PAYMENT** (many) side, a FK approach was used for this mapping. FKs InsuranceID and PrescriptionID were placed in the PAYMENT table.

PHARMACY EMPLOYEE entity:

- We decided to use the EmployeeID as its primary key because it is a unique attribute for each Employee.

(3) Physical database design–Oracle mapping

Below is our DDL Create Table statements, including domain constraints, integrity constraints, primary keys, foreign keys (with on delete actions), check constraints, etc. Te insert statements for our sample records are also below.

```
DROP TABLE HealthCareManagement_PATIENT CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_PATIENT(
PATIENT_ID CHAR(10) PRIMARY KEY,
DOB DATE,
STREET VARCHAR(30),
CITY VARCHAR(15),
STATE CHAR(2),
ZIP_CODE CHAR(5),
EMAIL VARCHAR(50),
PHONE_NUMBER VARCHAR(20),
LAST VARCHAR(10),
FIRST VARCHAR(10),
SEX VARCHAR(10),
INSURANCE_ID CHAR(10),
PASSWORD char(30),
FOREIGN KEY (INSURANCE_ID) REFERENCES
HealthCareManagement_INSURANCECOMPANY(INSURANCE_ID)
ON DELETE SET NULL);

DROP TABLE HealthCareManagement_DOCTOR CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_DOCTOR(
DOCTOR_ID CHAR(10) PRIMARY KEY,
LAST VARCHAR(10),
FIRST VARCHAR(10),
EMAIL VARCHAR(50),
PASSWORD char(30),
SPECIALIZATION VARCHAR(20),
OFFICE_NUMBER CHAR(3));

DROP TABLE HealthCareManagement_INSURANCECOMPANY CASCADE
CONSTRAINTS;
CREATE TABLE HealthCareManagement_INSURANCECOMPANY(
INSURANCE_ID CHAR(10) PRIMARY KEY,
INSURANCE_NAME VARCHAR(20),
STREET VARCHAR(30),
CITY VARCHAR(15),
STATE CHAR(2),
ZIP_CODE CHAR(5),
PHONE_NUMBER VARCHAR(20),
```

```
EMAIL VARCHAR(50),
PASSWORD char(30));


DROP TABLE HealthCareManagement_PRESCRIPTION CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_PRESCRIPTION(
PRESCRIPTION_ID CHAR(10) PRIMARY KEY,
DATE_ISSUED Date,
PRESCRIPTION_NAME VARCHAR(20),
DOSAGE VARCHAR(5),
REFILLS_REMAINING CHAR(3),
PRICE DECIMAL(6,2),
QUANTITY CHAR(5),
DOCTOR_ID CHAR(10),
PATIENT_ID CHAR(10),
FOREIGN KEY (DOCTOR_ID) REFERENCES
HealthCareManagement_DOCTOR(DOCTOR_ID)
ON DELETE CASCADE,
FOREIGN KEY (PATIENT_ID) REFERENCES
HealthCareManagement_PATIENT(PATIENT_ID)
ON DELETE CASCADE);

DROP TABLE HealthCareManagement_SUPPLIER CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_SUPPLIER(
SUPPLIER_ID CHAR(10) PRIMARY KEY,
SUPPLIER_NAME VARCHAR(20),
STREET VARCHAR(30),
CITY VARCHAR(15),
STATE CHAR(2),
ZIP_CODE CHAR(5),
PHONE_NUMBER VARCHAR(20),
PASSWORD char(30),
EMAIL VARCHAR(50));

DROP TABLE HealthCareManagement_MEDICATION CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_MEDICATION(
NAME VARCHAR(30) PRIMARY KEY,
QUANTITY CHAR(5),
SUPPLIER_ID CHAR(10),
FOREIGN KEY (SUPPLIER_ID) REFERENCES
HealthCareManagement_SUPPLIER(SUPPLIER_ID)
ON DELETE SET NULL);

DROP TABLE HealthCareManagement_PHARMACYEMPLOYEE CASCADE
CONSTRAINTS;
```

```
CREATE TABLE HealthCareManagement_PHARMACYEMPLOYEE(
EMPLOYEE_ID CHAR(10) PRIMARY KEY,
LAST VARCHAR(10),
FIRST VARCHAR(10),
SSN VARCHAR(9),
PHONE_NUMBER VARCHAR(20),
EMAIL VARCHAR(50),
POSITION VARCHAR(10),
PHARMACY_ID CHAR(10),
PASSWORD char(30),
FOREIGN KEY (PHARMACY_ID) REFERENCES
HealthCareManagement_PHARMACY(PHARMACY_ID)
ON DELETE SET NULL);

DROP TABLE HealthCareManagement_PAYSFOR CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_PAYSFOR(
PRESCRIPTION_ID CHAR(10) PRIMARY KEY,
INSURANCE_ID CHAR(10),
FOREIGN KEY (INSURANCE_ID) REFERENCES
HealthCareManagement_INSURANCECOMPANY(INSURANCE_ID)
ON DELETE SET NULL);

DROP TABLE HealthCareManagement_APPOINTMENT CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_APPOINTMENT(
PRIMARY KEY(PATIENT_ID, DOCTOR_ID),
NOTE VARCHAR(30),
APPOINTMENT_DATE Date,
PATIENT_ID CHAR(10),
DOCTOR_ID CHAR(10),
FOREIGN KEY (PATIENT_ID) REFERENCES
HealthCareManagement_PATIENT(PATIENT_ID)
ON DELETE CASCADE,
FOREIGN KEY (DOCTOR_ID) REFERENCES
HealthCareManagement_DOCTOR(DOCTOR_ID)
ON DELETE CASCADE);

DROP TABLE HealthCareManagement_PHARMACY CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_PHARMACY(
PHARMACY_ID CHAR(10) PRIMARY KEY,
PHARMACY_NAME VARCHAR(20),
STREET VARCHAR(30),
CITY VARCHAR(15),
STATE CHAR(2),
ZIP_CODE CHAR(5),
PHONE_NUMBER VARCHAR(20),
```

```
PASSWORD char(30),
EMAIL VARCHAR(50));

DROP TABLE HealthCareManagement_FILLS CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_FILLS(
PHARMACY_ID CHAR(10),
PRESCRIPTION_ID CHAR(10),
PRIMARY KEY(PHARMACY_ID, PRESCRIPTION_ID),
FOREIGN KEY (PHARMACY_ID) REFERENCES
HealthCareManagement_PHARMACY(PHARMACY_ID)
ON DELETE CASCADE,
FOREIGN KEY (PRESCRIPTION_ID) REFERENCES
HealthCareManagement_PRESCRIPTION(PRESCRIPTION_ID)
ON DELETE CASCADE);

DROP TABLE HealthCareManagement_DIAGNOSES CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_DIAGNOSES(
PATIENT_ID CHAR(10) PRIMARY KEY,
DIAGNOSES VARCHAR(30),
FOREIGN KEY (PATIENT_ID) REFERENCES
HealthCareManagement_PATIENT(PATIENT_ID)
ON DELETE SET NULL);

DROP TABLE HealthCareManagement_SUPPLIES CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_SUPPLIES(
SUPPLIER_ID CHAR(10),
PHARMACY_ID CHAR(10),
PRIMARY KEY(SUPPLIER_ID, PHARMACY_ID),
FOREIGN KEY (SUPPLIER_ID) REFERENCES
HealthCareManagement_SUPPLIER(SUPPLIER_ID)
ON DELETE CASCADE,
FOREIGN KEY (PHARMACY_ID) REFERENCES
HealthCareManagement_PHARMACY(PHARMACY_ID)
ON DELETE CASCADE);

DROP TABLE HealthCareManagement_PAYMENT CASCADE CONSTRAINTS;
CREATE TABLE HealthCareManagement_PAYMENT(
PAYMENT_ID CHAR(10) PRIMARY KEY,
PAYMENT_DATE DATE,
AMOUNT DECIMAL(10,2),
INSURANCE_ID CHAR(10),
PRESCRIPTION_ID CHAR(10),
FOREIGN KEY (INSURANCE_ID) REFERENCES
HealthCareManagement_INSURANCECOMPANY(INSURANCE_ID)
ON DELETE CASCADE,
```

```
FOREIGN KEY (PRESCRIPTION_ID) REFERENCES
HealthCareManagement_PRESCRIPTION(PRESCRIPTION_ID)
ON DELETE CASCADE);


INSERT INTO HealthCareManagement_INSURANCECOMPANY VALUES ('INS001',
'HealthPlus', '1234 Main St', 'Anytown', 'NY', '12345', '123-456-7890', 'info@healthplus.com',
'ths8673incd58n');
INSERT INTO HealthCareManagement_INSURANCECOMPANY VALUES ('INS002',
'MediCare', '5678 Elm St', 'Springfield', 'IL', '23456', '234-567-8901', 'support@medicare.com',
'thsbth6793niincd58n');
INSERT INTO HealthCareManagement_INSURANCECOMPANY VALUES ('INS003',
'WellFare', '9101 Oak St', 'Liberty', 'TX', '34567', '345-678-9012', 'contact@wellfare.com',
'password');
INSERT INTO HealthCareManagement_INSURANCECOMPANY VALUES ('INS004',
'SureHealth', '1213 Pine St', 'Centerville', 'CA', '45678', '456-789-0123', 'help@surehealth.com',
'password1');
INSERT INTO HealthCareManagement_INSURANCECOMPANY VALUES ('INS005',
'LifeSecure', '1415 Maple St', 'New Hope', 'FL', '56789', '567-890-1234', 'info@lifesecure.com',
'password2');

INSERT INTO HealthCareManagement_SUPPLIER VALUES ('SUP001', 'PharmaCo', '1234
Drug St', 'Medicity', 'CA', '67890', '678-901-2345', 'password00', 'supply@pharmaco.com');
INSERT INTO HealthCareManagement_SUPPLIER VALUES ('SUP002', 'MedSupplies', '5678
Pharma St', 'Careville', 'TX', '78901', '789-012-3456', 'password194', 'order@medsupplies.com');
INSERT INTO HealthCareManagement_SUPPLIER VALUES ('SUP003', 'DrugNest', '9101
Health St', 'Pilltown', 'FL', '89012', '890-123-4567', 'password90732', 'contact@drugnest.com');
INSERT INTO HealthCareManagement_SUPPLIER VALUES ('SUP004', 'VitaPharm', '1213
Vitamin St', 'Supplecity', 'NY', '90123', '901-234-5678', 'password320', 'info@vitapharm.com');
INSERT INTO HealthCareManagement_SUPPLIER VALUES ('SUP005', 'CareGoods', '1415
Remedy St', 'Aidville', 'IL', '01234', '012-345-6789', 'password-4392', 'service@caregoods.com');

INSERT INTO HealthCareManagement_DOCTOR VALUES ('DOC001', 'Smith', 'John',
'john.smith@hospital.com', 'thsbaibniincd68n', 'Cardiology', '101');
INSERT INTO HealthCareManagement_DOCTOR VALUES ('DOC002', 'Johnson', 'Emily',
'emily.johnson@clinic.com', 'thsbaibniincd08n', 'Dermatology', '102');
INSERT INTO HealthCareManagement_DOCTOR VALUES ('DOC003', 'Williams', 'David',
'david.williams@medcenter.com', 'thsbaibniincd98n', 'Neurology', '103');
INSERT INTO HealthCareManagement_DOCTOR VALUES ('DOC004', 'Brown', 'Sophia',
'sophia.brown@healthcare.com', 'thsbaib', 'Pediatrics', '104');
INSERT INTO HealthCareManagement_DOCTOR VALUES ('DOC005', 'Davis', 'Michael',
'michael.davis@generalhospital.com', 'ibnincd98n', 'General', '105');
```

INSERT INTO HealthCareManagement_PATIENT VALUES ('PAT001', TO_DATE('1990-01-01','YYYY-MM-DD'), '1234 Life St', 'Anytown', 'NY', '12345', 'patient1@email.com', '123-456-7890', 'Doe', 'Jane', 'Female', 'INS001', 'thsbaibniincd58n');
INSERT INTO HealthCareManagement_PATIENT VALUES ('PAT002', TO_DATE('1985-02-02','YYYY-MM-DD'), '5678 Health Rd', 'Wellville', 'TX', '23456', 'patient2@email.com', '234-567-8901', 'Brown', 'John', 'Male', 'INS002', 'thsbaibniincd59n');
INSERT INTO HealthCareManagement_PATIENT VALUES ('PAT003', TO_DATE('1975-03-03','YYYY-MM-DD'), '9101 Care Ave', 'Curecity', 'CA', '34567', 'patient3@email.com', '345-678-9012', 'Smith', 'Emily', 'Female', 'INS003', 'thsbaibniincd60n');
INSERT INTO HealthCareManagement_PATIENT VALUES ('PAT004', TO_DATE('2000-04-04','YYYY-MM-DD'), '1213 Remedy Blvd', 'Aidtown', 'FL', '45678', 'patient4@email.com', '456-789-0123', 'Johnson', 'Michael', 'Male', 'INS004', 'thsbaibniincd61n');
INSERT INTO HealthCareManagement_PATIENT VALUES ('PAT005', TO_DATE('1995-05-05','YYYY-MM-DD'), '1415 Wellness Ln', 'Hopetown', 'IL', '56789', 'patient5@email.com', '567-890-1234', 'Williams', 'Sophia', 'Female', 'INS005', 'thsbaibniincd62n');

INSERT INTO HealthCareManagement_PRESCRIPTION VALUES ('PRSC001', TO_DATE('2023-01-01','YYYY-MM-DD'), 'Amoxicillin', '500mg', '05', 25.00, '30', 'DOC001', 'PAT001');
INSERT INTO HealthCareManagement_PRESCRIPTION VALUES ('PRSC002', TO_DATE('2023-02-01','YYYY-MM-DD'), 'Ibuprofen', '200mg', '03', 15.00, '20', 'DOC002', 'PAT002');
INSERT INTO HealthCareManagement_PRESCRIPTION VALUES ('PRSC003', TO_DATE('2023-03-01','YYYY-MM-DD'), 'Metformin', '850mg', '02', 30.00, '60', 'DOC003', 'PAT003');
INSERT INTO HealthCareManagement_PRESCRIPTION VALUES ('PRSC004', TO_DATE('2023-04-01','YYYY-MM-DD'), 'Lisinopril', '10mg', '04', 22.00, '90', 'DOC004', 'PAT004');
INSERT INTO HealthCareManagement_PRESCRIPTION VALUES ('PRSC005', TO_DATE('2023-05-01','YYYY-MM-DD'), 'Atorvastatin', '20mg', '01', 45.00, '10', 'DOC005', 'PAT005');

INSERT INTO HealthCareManagement_MEDICATION VALUES ('Amoxicillin', '100', 'SUP001');
INSERT INTO HealthCareManagement_MEDICATION VALUES ('Ibuprofen', '200', 'SUP002');
INSERT INTO HealthCareManagement_MEDICATION VALUES ('Metformin', '150', 'SUP003');
INSERT INTO HealthCareManagement_MEDICATION VALUES ('Lisinopril', '120', 'SUP004');
INSERT INTO HealthCareManagement_MEDICATION VALUES ('Atorvastatin', '80', 'SUP005');

INSERT INTO HealthCareManagement_PHARMACY VALUES ('PHRM001', 'City Pharmacy', '123 Cure St', 'Healtown', 'NY', '12345', '123-456-1111', 'password9', 'pharmacy@citypharm.com');

```
INSERT INTO HealthCareManagement_PHARMACY VALUES ('PHRM002', 'MediPharm',
'456 Pill Rd', 'Medville', 'TX', '23456', '234-567-2222', 'password10', 'info@medipharm.com');
INSERT INTO HealthCareManagement_PHARMACY VALUES ('PHRM003', 'CarePlus
Pharmacy', '789 Health Ave', 'Carecity', 'CA', '34567', '345-678-3333', 'password11',
'support@careluspharm.com');
INSERT INTO HealthCareManagement_PHARMACY VALUES ('PHRM004', 'Wellness
Pharmacy', '1012 Remedy Blvd', 'Welltown', 'FL', '45678', '456-789-4444', 'password12',
'contact@wellnesspharm.com');
INSERT INTO HealthCareManagement_PHARMACY VALUES ('PHRM005', 'Hope
Pharmacy', '1314 Wellness Ln', 'Hopetown', 'IL', '56789', '567-890-5555', 'password13',
'service@hopepharm.com');

INSERT INTO HealthCareManagement_PHARMACYEMPLOYEE VALUES ('EMP001',
'Miller', 'Alice', '123456789', '123-456-6666', 'alice.miller@pharmacy.com', 'Pharmacist',
'PHRM001', 'password123');
INSERT INTO HealthCareManagement_PHARMACYEMPLOYEE VALUES ('EMP002',
'Wilson', 'Bob', '987654321', '234-567-7777', 'bob.wilson@pharmacy.com', 'Assistant',
'PHRM002', 'password1234');
INSERT INTO HealthCareManagement_PHARMACYEMPLOYEE VALUES ('EMP003',
'Moore', 'Clara', '456789123', '345-678-8888', 'clara.moore@pharmacy.com', 'Manager',
'PHRM003', 'password93483');
INSERT INTO HealthCareManagement_PHARMACYEMPLOYEE VALUES ('EMP004',
'Taylor', 'Dan', '654321987', '456-789-9999', 'dan.taylor@pharmacy.com', 'Technician',
'PHRM004', 'password0383');
INSERT INTO HealthCareManagement_PHARMACYEMPLOYEE VALUES ('EMP005',
'Anderson', 'Eva', '321654987', '567-890-0000', 'eva.anderson@pharmacy.com', 'Clerk',
'PHRM005', 'password1849');

INSERT INTO HealthCareManagement_PAYSFOR VALUES ('PRSC001', 'INS001');
INSERT INTO HealthCareManagement_PAYSFOR VALUES ('PRSC002', 'INS002');
INSERT INTO HealthCareManagement_PAYSFOR VALUES ('PRSC003', 'INS003');
INSERT INTO HealthCareManagement_PAYSFOR VALUES ('PRSC004', 'INS004');
INSERT INTO HealthCareManagement_PAYSFOR VALUES ('PRSC005', 'INS005');

INSERT INTO HealthCareManagement_APPOINTMENT (PATIENT_ID, DOCTOR_ID,
NOTE, APPOINTMENT_DATE) VALUES ('PAT001', 'DOC001', 'Follow-up Check',
TO_DATE('2023-06-01', 'YYYY-MM-DD'));
INSERT INTO HealthCareManagement_APPOINTMENT (PATIENT_ID, DOCTOR_ID,
NOTE, APPOINTMENT_DATE) VALUES ('PAT002', 'DOC002', 'Routine Checkup',
TO_DATE('2023-07-01', 'YYYY-MM-DD'));
INSERT INTO HealthCareManagement_APPOINTMENT (PATIENT_ID, DOCTOR_ID,
NOTE, APPOINTMENT_DATE) VALUES ('PAT003', 'DOC003', 'Consultation',
TO_DATE('2023-08-01', 'YYYY-MM-DD'));
```

```
INSERT INTO HealthCareManagement_APPOINTMENT (PATIENT_ID, DOCTOR_ID,
NOTE, APPOINTMENT_DATE) VALUES ('PAT004', 'DOC004', 'Annual Physical',
TO_DATE('2023-09-01', 'YYYY-MM-DD'));
INSERT INTO HealthCareManagement_APPOINTMENT (PATIENT_ID, DOCTOR_ID,
NOTE, APPOINTMENT_DATE) VALUES ('PAT005', 'DOC005', 'Emergency Visit',
TO_DATE('2023-10-01', 'YYYY-MM-DD'));

INSERT INTO HealthCareManagement_FILLS VALUES ('PHRM001', 'PRSC001');
INSERT INTO HealthCareManagement_FILLS VALUES ('PHRM002', 'PRSC002');
INSERT INTO HealthCareManagement_FILLS VALUES ('PHRM003', 'PRSC003');
INSERT INTO HealthCareManagement_FILLS VALUES ('PHRM004', 'PRSC004');
INSERT INTO HealthCareManagement_FILLS VALUES ('PHRM005', 'PRSC005');

INSERT INTO HealthCareManagement_DIAGNOSES VALUES ('PAT001', 'Cough');
INSERT INTO HealthCareManagement_DIAGNOSES VALUES ('PAT002', 'Flu');
INSERT INTO HealthCareManagement_DIAGNOSES VALUES ('PAT003', 'Asthma');
INSERT INTO HealthCareManagement_DIAGNOSES VALUES ('PAT004', 'Diabetes');
INSERT INTO HealthCareManagement_DIAGNOSES VALUES ('PAT005', 'Hypertension');

INSERT INTO HealthCareManagement_SUPPLIES VALUES ('SUP001', 'PHRM001');
INSERT INTO HealthCareManagement_SUPPLIES VALUES ('SUP002', 'PHRM002');
INSERT INTO HealthCareManagement_SUPPLIES VALUES ('SUP003', 'PHRM003');
```

(4) SQL Routines & (5) Data Processing

PATIENT: Create Account (Matt)

- **Description:** Allows new Patients to create a new account. Trigger before insertion on table patient. The trigger additionally uses a function to randomly generate a patient's ID
- **SQL Queries:** (file: MattFunctionalitiesCode)

```oracle
--Trigger to update the patient table when a new patient is created
--uses the function Generategenerate_random_patient_id to create an id for a patient
--Matt DeRosa
CREATE OR REPLACE TRIGGER create_Account
BEFORE INSERT ON HealthCareManagement_Patient
FOR EACH ROW
BEGIN
    :NEW.PATIENT_ID := Generate_Random_Patient_ID();
    :NEW.DOB := :NEW.DOB; -- DOB
    :NEW.STREET := :NEW.STREET; -- STREET
    :NEW.CITY := :NEW.CITY; -- CITY
    :NEW.STATE := :NEW.STATE; -- STATE
    :NEW.ZIP_CODE := :NEW.ZIP_CODE; -- ZIP_CODE
    :NEW.EMAIL := :NEW.EMAIL; -- EMAIL
    :NEW.PHONE_NUMBER := :NEW.PHONE_NUMBER; -- PHONE_NUMBER
    :NEW.LAST := :NEW.LAST; -- LAST
    :NEW.FIRST := :NEW.FIRST; -- FIRST
    :NEW.SEX := :NEW.SEX; -- SEX
    :NEW.INSURANCE_ID := :NEW.INSURANCE_ID; -- INSURANCE_ID
    :NEW.PASSWORD := :NEW.PASSWORD; -- PASSWORD
END;
/
```

```oracle
--Function for Creating a new patient Id when they create an account
--Matt DeRosa
CREATE OR REPLACE FUNCTION Generate_Random_Patient_ID
RETURN CHAR IS
    l_prefix CHAR(3) := 'PAT';
    l_suffix CHAR(7);
BEGIN
    -- Generate a random number between 1000000 and 9999999
    l_suffix := TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(1000000, 9999999)));

    -- Concatenate prefix and suffix to form the patient ID
    RETURN l_prefix || l_suffix;
END;
/
```

```oracle
--pretest output
-- Selecting all rows from the healthcaremanagement_patient table
--Matt DeRosa
SELECT * FROM HealthCareManagement_Patient;

-- Inserting a new patient without specifying the patient ID
INSERT INTO HealthCareManagement_Patient (DOB, STREET, CITY, STATE, ZIP_CODE, EMAIL, PHONE_NUMBER, LAST, FIRST, SEX, INSURANCE_ID, PASSWORD)
VALUES (TO_DATE('1990-01-01','YYYY-MM-DD'), '1234 Life St', 'Atlanta', 'NY', '12345', 'test@email.com',
'123-480-4387', 'Doe', 'John', 'Male', 'INS001', 'password123');

-- Selecting all rows from the healthcaremanagement_patient table to test the output
SELECT * FROM HealthCareManagement_Patient;
```

- **Output:** The last row shows that there is a random id number generated. Proving the function and trigger work properly.

```
Trigger CREATE_ACCOUNT compiled

PATIENT_ID DOB       STREET                         CITY            ST ZIP_C EMAIL                                          PHONE_NUMBER
LAST       FIRST      SEX        INSURANCE_ PASSWORD
---------- --------- ----------------------------- --------------- -- ----- ------------------------------------------------ --------------------
---------- ---------- ---------- ---------- ------------------------------
PAT001     01-JAN-90 789 Updated St                 Updated City    NY 54321 updated_email@example.com                       555-555-5555
Doe        Jane       Female     INS-UPDATE thsbaibniincd58n
PAT002     02-FEB-85 5678 Health Rd                 Wellville       TX 23456 patient2@email.com                              234-567-8901
Brown      John       Male       INS002     thsbaibniincd59n
PAT003     03-MAR-75 9101 Care Ave                  Curecity        CA 34567 patient3@email.com                              345-678-9012
Smith      Emily      Female     INS003     thsbaibniincd60n
PAT004     04-APR-00 1213 Remedy Blvd               Aidtown         FL 45678 patient4@email.com                              456-789-0123
Johnson    Michael    Male       INS004     thsbaibniincd61n
PAT005     05-MAY-95 1415 Wellness Ln               Hopetown        IL 56789 patient5@email.com                              567-890-1234
Williams   Sophia     Female     INS005     thsbaibniincd62n


1 row inserted.

PATIENT_ID DOB       STREET                         CITY            ST ZIP_C EMAIL                                          PHONE_NUMBER
LAST       FIRST      SEX        INSURANCE_ PASSWORD
---------- --------- ----------------------------- --------------- -- ----- ------------------------------------------------ --------------------
---------- ---------- ---------- ---------- ------------------------------
PAT001     01-JAN-90 789 Updated St                 Updated City    NY 54321 updated_email@example.com                       555-555-5555
Doe        Jane       Female     INS-UPDATE thsbaibniincd58n
PAT002     02-FEB-85 5678 Health Rd                 Wellville       TX 23456 patient2@email.com                              234-567-8901
Brown      John       Male       INS002     thsbaibniincd59n
PAT003     03-MAR-75 9101 Care Ave                  Curecity        CA 34567 patient3@email.com                              345-678-9012
Smith      Emily      Female     INS003     thsbaibniincd60n
PAT004     04-APR-00 1213 Remedy Blvd               Aidtown         FL 45678 patient4@email.com                              456-789-0123
Johnson    Michael    Male       INS004     thsbaibniincd61n
PAT005     05-MAY-95 1415 Wellness Ln               Hopetown        IL 56789 patient5@email.com                              567-890-1234
Williams   Sophia     Female     INS005     thsbaibniincd62n
PAT6110021 01-JAN-90 1234 Life St                   Atlanta         NY 12345 test@email.com                                  123-480-4387
Doe        John       Male       INS001     password123
```

PATIENT: Login (Matt)

- **Description:** Creates a view with only the necessary fields for logging in to validate a patient log in.
- **SQL Queries:** (file: MattFunctionalitiesCode

```sql
--View: Login
--This view will validate the login credentials provided by the user.
--Matt DeRosa
CREATE OR REPLACE VIEW User_Login AS
SELECT patient_ID, email, PASSWORD
FROM healthcaremanagement_patient;

--test view for log in validation
SELECT * FROM User_Login;
```

- **Output:**

```
View USER_LOGIN created.


PATIENT_ID EMAIL                                              PASSWORD
---------- ------------------------------------------------- ------------------------------
PAT001     updated_email@example.com                         thsbaibniincd58n
PAT002     patient2@email.com                                thsbaibniincd59n
PAT003     patient3@email.com                                thsbaibniincd60n
PAT004     patient4@email.com                                thsbaibniincd61n
PAT005     patient5@email.com                                thsbaibniincd62n
```

PATIENT: View/edit profile (Matt)

- **Description:** View profile is just the entire Patient table, so nothing was implemented to see a patient. For a patient to edit their own profile a procedure was created that allows a patient to edit only their phone number, email, address lines, insurance, and sex.
- **SQL Queries:** (file: MattFunctionalitiesCode

```oracle
CREATE OR REPLACE PROCEDURE Edit_Patient_Info(
    p_patient_id IN CHAR,
    p_phone_number IN VARCHAR,
    p_email IN VARCHAR,
    p_street IN VARCHAR,
    p_city IN VARCHAR,
    p_state IN CHAR,
    p_zip_code IN CHAR,
    p_insurance_id IN CHAR,
    p_sex IN VARCHAR
)
AS
BEGIN
    -- Update the specified columns for the patient
    UPDATE HealthCareManagement_PATIENT
    SET
        PHONE_NUMBER = p_phone_number,
        EMAIL = p_email,
        STREET = p_street,
        CITY = p_city,
        STATE = p_state,
        ZIP_CODE = p_zip_code,
        INSURANCE_ID = p_insurance_id,
        SEX = p_sex
    WHERE PATIENT_ID = p_patient_id;

    -- Commit the transaction
    COMMIT;

    -- Output success message
    DBMS_OUTPUT.PUT_LINE('Patient information updated successfully.');
EXCEPTION
    WHEN OTHERS THEN
        -- Output error message if an exception occurs
        DBMS_OUTPUT.PUT_LINE('Error updating patient information: ' || SQLERRM);
END;
/
```

```oracle
--Test procedure for editing a patient profile
SELECT * FROM HealthCareManagement_PATIENT WHERE PATIENT_ID = 'PAT001';

-- Call the procedure to update the patient's information
BEGIN
    Edit_Patient_Info(
        p_patient_id => 'PAT001',
        p_phone_number => '555-555-5555',
        p_email => 'updated_email@example.com',
        p_street => '789 Updated St',
        p_city => 'Updated City',
        p_state => 'NY',
        p_zip_code => '54321',
        p_insurance_id => 'INS-UPDATE',
        p_sex => 'Female'
    );
END;
/

-- After running the procedure, select the patient's information again to verify the changes
SELECT * FROM HealthCareManagement_PATIENT WHERE PATIENT_ID = 'PAT001';
```

- **Output:** The new updatable data for a specific patient is shown in the bottom output.

```
Procedure EDIT_PATIENT_INFO compiled


PATIENT_ID DOB       STREET                         CITY            ST ZIP_C EMAIL                                          PHONE_NUMBER
LAST       FIRST      SEX        INSURANCE_ PASSWORD
---------- --------- ------------------------------ --------------- -- ----- -------------------------------------------------- --------------------
---------- ---------- ---------- ------------------------------
PAT001     01-JAN-90 1234 Life St                   Anytown         NY 12345 patient1@email.com                                 123-456-7890
Doe        Jane       Female     INS001     thsbaibniincd58n


PL/SQL procedure successfully completed.


PATIENT_ID DOB       STREET                         CITY            ST ZIP_C EMAIL                                          PHONE_NUMBER
LAST       FIRST      SEX        INSURANCE_ PASSWORD
---------- --------- ------------------------------ --------------- -- ----- -------------------------------------------------- --------------------
---------- ---------- ---------- ------------------------------
PAT001     01-JAN-90 789 Updated St                 Updated City    NY 54321 updated_email@example.com                          555-555-5555
Doe        Jane       Female     INS-UPDATE thsbaibniincd58n
```

PATIENT: View Appointment Info (Matt)

- **Description:** Created a view for only the necessary info a patient would need to see about an appointment. Doctors name, date of appointment, and the notes left on the appointment.
- **SQL Queries:** (file: MattFunctionalitiesCode

```sql
-- Create a view to provide patients with information about their appointments,
-- including the name of the doctor, appointment date, and notes.
--Matt DeRosa
CREATE OR REPLACE VIEW Patient_Appointment_Info AS
SELECT D.LAST || ', ' || D.FIRST AS DOCTOR_NAME,
       A.APPOINTMENT_DATE,
       A.NOTE
FROM HealthCareManagement_APPOINTMENT A
JOIN HealthCareManagement_DOCTOR D ON A.DOCTOR_ID = D.DOCTOR_ID;

--Test Patient_Appointment_Info
Select * From Patient_Appointment_Info
```

- **Output:** This allows a patient to only access certain information about their appointment.

```
View PATIENT_APPOINTMENT_INFO created.


DOCTOR_NAME            APPOINTME NOTE
---------------------- --------- ------------------------------
Smith, John            01-JUN-23 Follow-up Check
Johnson, Emily         01-JUL-23 Routine Checkup
Williams, David        01-AUG-23 Consultation
Brown, Sophia          01-SEP-23 Annual Physical
Davis, Michael         01-OCT-23 Emergency Visit
```

PATEINT: View Diagnoses (Max)

- **Description:** To view patient a diagnosis, a view was made to see results.
- **SQL Queries:**

```sql
CREATE VIEW HealthCareManagement_SEEDIAGNOSIS AS
SELECT
    p.PATIENT_ID,
    p.FIRST || ' ' || p.LAST AS Patient_Name,
    p.DOB,
    d.DIAGNOSES
FROM
    HealthCareManagement_PATIENT p
```

- **Output:**

```
PATIENT_ID PATIENT_NAME          DOB       DIAGNOSES
---------- --------------------- --------- ----------------------
PAT001     Jane Doe              01-JAN-90 Cough
PAT002     John Brown            02-FEB-85 Flu
PAT003     Emily Smith           03-MAR-75 Asthma
PAT004     Michael Johnson       04-APR-00 Diabetes
PAT005     Sophia Williams       05-MAY-95 Hypertension
```

SUPPLIER: View/edit profile (Max)

- **Description:** For a supplier to edit their own profile a procedure was created that allows them to update information.
- **SQL Queries:**

```
CREATE OR REPLACE PROCEDURE Edit_Supplier_Info(
    s_supplier_id IN CHAR,
    s_supplier_name IN VARCHAR,
    s_street IN VARCHAR,
    s_city IN VARCHAR,
    s_state IN CHAR,
    s_zip_code IN CHAR,
    s_phone_number IN VARCHAR,
    s_email IN VARCHAR)
AS
BEGIN
    UPDATE HealthCareManagement_SUPPLIER
    SET
        SUPPLIER_NAME = s_supplier_name,
        STREET = s_street,
        CITY = s_city,
        STATE = s_state,
        ZIP_CODE = s_zip_code,
        PHONE_NUMBER = s_phone_number,
        EMAIL = s_email
    WHERE SUPPLIER_ID = s_supplier_id;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Supplier information updated successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error updating supplier information: ' || SQLERRM);
END;
/
```

- **Output:**

```
Procedure EDIT_SUPPLIER_INFO compiled

SUPPLIER_I SUPPLIER_NAME        STREET                       CITY           ST ZIP_C PHONE_NUMBER        PASSWORD                       EMAIL
---------- -------------------- ---------------------------- -------------- -- ----- -------------------- ------------------------------ ------------------------
SUP001     PharmaCo             1234 Drug St                 Medicity       CA 67890 678-901-2345        password00                     supply@pharmaco.com

PL/SQL procedure successfully completed.

SUPPLIER_I SUPPLIER_NAME        STREET                       CITY           ST ZIP_C PHONE_NUMBER        PASSWORD                       EMAIL
---------- -------------------- ---------------------------- -------------- -- ----- -------------------- ------------------------------ ------------------------
SUP001     UpdatedName          1111 Update St.              Update City    UT 11111 763-123-1233        password00                     updatedEmail@gmail.com
```

PHARMACY EMPLOYEE: View Inventory (Evan)

- **Description:** Allows employees to view all medication currently in the pharmacy.
- **SQL Queries:**

```
----View for Prescription Information
create or replace view Pharmacy_inventory as
    select *
    from HealthCareManagement_MEDICATION;
```

- **Output:**

```
NAME                           QUANT SUPPLIER_I
------------------------------ ----- ----------
Amoxicillin                    100   SUP001
Ibuprofen                      200   SUP002
Metformin                      150   SUP003
Lisinopril                     120   SUP004
Atorvastatin                   80    SUP005
```

PHARMACY EMPLOYEE: Bill Patient/Insurance (Evan)

- **Description:** Returns amount due for insurance company to pay for.
- **SQL Queries:**

```
--Bill
create or replace function Bill_Insurance(insure_id varchar, pay_id varchar) return int as
due int := 0;
begin
select p.amount into due
from healthcaremanagement_payment P
where insure_id = P.insurance_id and pay_id = p.payment_id;


Return due;
end;


select Bill_Insurance('INS001', 'PAT001') from dual;
```

- **Output:** no output for now as there is currently no payments due

```
Function BILL_INSURANCE compiled
```

PHARMACY EMPLOYEE: Request medication from supplier (Evan)

- **Description:** Refills the quantity of medication from a specific supplier(will need to be edited when part 3 is started due to the actual trigger being a button pressed)
- **SQL Queries:**

```
--These are place holders for the actual trigger which will be the press of a button in the UI
select quantity
from healthcaremanagement_medication;

CREATE OR REPLACE PROCEDURE UpdateSupplierQuantity(supplier_id varchar) as
begin
    UPDATE healthcaremanagement_medication M
    SET quantity = 100
    WHERE supplier_id = M.supplier_id;


END;
/

CREATE OR REPLACE TRIGGER CheckQuantityTrigger
BEFORE INSERT OR UPDATE ON healthcaremanagement_medication
FOR EACH ROW
WHEN (NEW.quantity < 10)

BEGIN

UpdateSupplierQuantity(:NEW.supplier_id);
END;
/

--Tests

BEGIN
   UpdateSupplierQuantity('SUP001');
    END;
```

- **Output:**

```
QUANT
-----
100
200
150
120
80


Procedure UPDATESUPPLIERQUANTITY compiled


Trigger CHECKQUANTITYTRIGGER compiled
```

PHARMACY EMPLOYEE: View medicine info (Evan)

- **Description: Allows for viewing of medication/prescription information**
- **SQL Queries:**

```
--View for Prescription information
create or replace view Medication_Info as
    select *
    from HealthCareManagement_Prescription;
```

- **Output:**

```
PRESCRIPTI DATE_ISSU PRESCRIPTION_NAME    DOSAG REF      PRICE QUANT DOCTOR_ID  PATIENT_ID
---------- --------- -------------------- ----- --- ---------- ----- ---------- ----------
PRSC001    01-JAN-23 Amoxicillin          500mg 05           25 30    DOC001     PAT001
PRSC002    01-FEB-23 Ibuprofen            200mg 03           15 20    DOC002     PAT002
PRSC003    01-MAR-23 Metformin            850mg 02           30 60    DOC003     PAT003
PRSC004    01-APR-23 Lisinopril           10mg  04           22 90    DOC004     PAT004
PRSC005    01-MAY-23 Atorvastatin         20mg  01           45 10    DOC005     PAT005
```

HOSPITAL ADMIN: View and manage users (Max)

- **Description:** See a list of all users and their types of users in the database and add users. Also view info about how many users there are of each type.
- **SQL Queries: View for Hospital Admin to see all users**

```
CREATE OR REPLACE VIEW AllUsersOverview AS
SELECT USER_ID, NAME, USER_TYPE, EMAIL, PHONE_NUMBER
FROM AllUsersOverview
ORDER BY User_Type, NAME;
```

Function to see the count of each user type

```oracle
CREATE OR REPLACE FUNCTION CountUsersByType (userType VARCHAR2)
RETURN INT IS
    userCount INT;
BEGIN
    SELECT COUNT(*)
    INTO userCount
    FROM AllUsersOverview
    WHERE User_Type = userType;

    RETURN userCount;
END;
/

SELECT CountUsersByType('Patient') AS Patient_Count FROM dual;
SELECT CountUsersByType('Doctor') AS Doctor_Count FROM dual;
SELECT CountUsersByType('Pharmacy Employee') AS PharmacyEmployee_Count FROM dual;
```

- **Output:**

```
USER_ID    NAME                 USER_TYPE         EMAIL                                              PHONE_NUMBER
---------- -------------------- ----------------- -------------------------------------------------- ------------------
DOC001     John Smith           Doctor            john.smith@hospital.com
DOC002     Emily Johnson        Doctor            emily.johnson@clinic.com
DOC003     David Williams       Doctor            david.williams@medcenter.com
DOC004     Sophia Brown         Doctor            sophia.brown@healthcare.com
DOC005     Michael Davis        Doctor            michael.davis@generalhospital.com
EMP001     Alice Miller         Pharmacy Employee alice.miller@pharmacy.com                          123-456-6666
EMP002     Bob Wilson           Pharmacy Employee bob.wilson@pharmacy.com                            234-567-7777
EMP003     Clara Moore          Pharmacy Employee clara.moore@pharmacy.com                           345-678-8888
EMP004     Dan Taylor           Pharmacy Employee dan.taylor@pharmacy.com                            456-789-9999
EMP005     Eva Anderson         Pharmacy Employee eva.anderson@pharmacy.com                          567-890-0000
PAT001     Jane Doe             Patient           updated_email@example.com                          555-555-5555

USER_ID    NAME                 USER_TYPE         EMAIL                                              PHONE_NUMBER
---------- -------------------- ----------------- -------------------------------------------------- ------------------
PAT002     John Brown           Patient           patient2@email.com                                 234-567-8901
PAT003     Emily Smith          Patient           patient3@email.com                                 345-678-9012
PAT004     Michael Johnson      Patient           patient4@email.com                                 456-789-0123
PAT005     Sophia Williams      Patient           patient5@email.com                                 567-890-1234
PAT6110021 John Doe             Patient           test@email.com                                     123-480-4387
```

```
PATIENT_COUNT
-------------
            6


DOCTOR_COUNT
------------
           5


PHARMACYEMPLOYEE_COUNT
----------------------
                     5
```

DOCTOR: Submit prescription request – Trigger (Mason)

- **Description:** Allows doctors to create prescriptions for their patients.

- **SQL Queries:**

```oracle_plsql
-- Create the stored procedure for inserting new prescriptions
CREATE OR REPLACE PROCEDURE Insert_Prescription (
    p_prescription_id IN HealthCareManagement_PRESCRIPTION.PRESCRIPTION_ID%TYPE,
    p_date_issued IN HealthCareManagement_PRESCRIPTION.DATE_ISSUED%TYPE DEFAULT SYSDATE,
    p_prescription_name IN HealthCareManagement_PRESCRIPTION.PRESCRIPTION_NAME%TYPE,
    p_dosage IN HealthCareManagement_PRESCRIPTION.DOSAGE%TYPE,
    p_refills_remaining IN HealthCareManagement_PRESCRIPTION.REFILLS_REMAINING%TYPE,
    p_price IN HealthCareManagement_PRESCRIPTION.PRICE%TYPE,
    p_quantity IN HealthCareManagement_PRESCRIPTION.QUANTITY%TYPE,
    p_doctor_id IN HealthCareManagement_PRESCRIPTION.DOCTOR_ID%TYPE,
    p_patient_id IN HealthCareManagement_PRESCRIPTION.PATIENT_ID%TYPE
) AS
BEGIN
    INSERT INTO HealthCareManagement_PRESCRIPTION (
        PRESCRIPTION_ID, DATE_ISSUED, PRESCRIPTION_NAME, DOSAGE, REFILLS_REMAINING, PRICE, QUANTITY, DOCTOR_ID, PATIENT_ID
    ) VALUES (
        p_prescription_id, COALESCE(p_date_issued, SYSDATE), p_prescription_name, p_dosage,
        p_refills_remaining, p_price, p_quantity, p_doctor_id, p_patient_id
    );
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/

-- Create a trigger that ensures DATE_ISSUED is set to SYSDATE if not provided
CREATE OR REPLACE TRIGGER Ensure_Date_Issued
BEFORE INSERT ON HealthCareManagement_PRESCRIPTION
FOR EACH ROW
WHEN (NEW.DATE_ISSUED IS NULL)
BEGIN
    :NEW.DATE_ISSUED := SYSDATE;
END;
```

- **Output:** Make new prescription

```oracle_plsql
--TEST
SELECT PRESCRIPTION_ID,
       TO_CHAR(DATE_ISSUED, 'YYYY-MM-DD') AS DATE_ISSUED,
       PRESCRIPTION_NAME,
       DOSAGE,
       REFILLS_REMAINING,
       PRICE,
       QUANTITY,
       DOCTOR_ID,
       PATIENT_ID
FROM HealthCareManagement_PRESCRIPTION
WHERE PRESCRIPTION_ID = 'RX202340';
```

DOCTOR: View patient info – View (Mason)

- **Description:** Allow doctor users to view certain information from their patients' profiles and view their diagnoses.
- **SQL Queries:**

```oracle_plsql
CREATE OR REPLACE VIEW Doctor_Patient_Diagnoses AS
SELECT
    p.PATIENT_ID,
    p.FIRST || ' ' || p.LAST AS Patient_Name,
    p.DOB,
    p.STREET,
    p.CITY,
    p.STATE,
    p.ZIP_CODE,
    p.EMAIL,
    p.PHONE_NUMBER,
    p.SEX,
    d.DIAGNOSES
FROM
    HealthCareManagement_PATIENT p
JOIN
    HealthCareManagement_DIAGNOSES d ON p.PATIENT_ID = d.PATIENT_ID;
```

- **Output:**

PATIENT_ID	PATIENT_NAME	DOB	STREET	CITY	ST	ZIP_C	EMAIL	PHONE_NUMBER	SEX	DIAGNOSES
PAT001	Jane Doe	01-JAN-90	1234 Life St	Anytown	NY	12345	patient1@email.com	123-456-7890	Female	Cough
PAT002	John Brown	02-FEB-85	5678 Health Rd	Wellville	TX	23456	patient2@email.com	234-567-8901	Male	Flu
PAT003	Emily Smith	03-MAR-75	9101 Care Ave	Curecity	CA	34567	patient3@email.com	345-678-9012	Female	Asthma
PAT004	Michael Johnson	04-APR-00	1213 Remedy Blvd	Aidtown	FL	45678	patient4@email.com	456-789-0123	Male	Diabetes
PAT005	Sophia Williams	05-MAY-95	1415 Wellness Ln	Hopetown	IL	56789	patient5@email.com	567-890-1234	Female	Hypertension

DOCTOR: Edit patient info (Diagnosis) - Procedure (Mason)

- **Description:** The Update_Patient_Diagnosis procedure updates a patient's diagnosis in a database. It takes a patient ID and a new diagnosis as inputs, applies the update, and includes error handling to revert changes if an error occurs, ensuring data integrity.
- **SQL Queries**

```oracle
-- Procedure to update patient's diagnosis
CREATE OR REPLACE PROCEDURE Update_Patient_Diagnosis(
    p_patient_id IN HealthCareManagement_DIAGNOSES.PATIENT_ID%TYPE,
    p_new_diagnosis IN HealthCareManagement_DIAGNOSES.DIAGNOSES%TYPE)
IS
BEGIN
    UPDATE HealthCareManagement_DIAGNOSES
    SET DIAGNOSES = p_new_diagnosis
    WHERE PATIENT_ID = p_patient_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No such patient exists.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error updating diagnosis: ' || SQLERRM);
        ROLLBACK;--execption handling to undo any changes made to the database
END;
/

-- Test the procedure with sample data
BEGIN
    Update_Patient_Diagnosis(p_patient_id => 'PAT001', p_new_diagnosis => 'Updated Diagnosis Example');
END;
/

-- Commit the transaction
COMMIT;

-- Verify the update
SELECT * FROM HealthCareManagement_DIAGNOSES WHERE PATIENT_ID = 'PAT001';
```

- **Output:**

```
Script Output ×
Task completed in 2.756 seconds

PATIENT_ID DIAGNOSES
---------- ------------------------------
PAT001     Updated Diagnosis Example
```

DOCTOR: Edit appointment notes – Function (Mason)

- **Description:** Allows doctors to leave appointment note and date after seeing a patient.
- **SQL Queries:**

```oracle_plsql
-- Function to add or update an appointment note
CREATE OR REPLACE FUNCTION AddOrUpdate_Appointment_Note
(
    p_patient_id IN HealthCareManagement_APPOINTMENT.PATIENT_ID%TYPE,
    p_doctor_id IN HealthCareManagement_APPOINTMENT.DOCTOR_ID%TYPE,
    p_note IN HealthCareManagement_APPOINTMENT.NOTE%TYPE,
    p_appointment_date IN HealthCareManagement_APPOINTMENT.APPOINTMENT_DATE%TYPE
)
RETURN VARCHAR2
IS
    v_count NUMBER;
BEGIN
    -- Check if an appointment already exists
    SELECT COUNT(*)
    INTO v_count
    FROM HealthCareManagement_APPOINTMENT
    WHERE PATIENT_ID = p_patient_id
      AND DOCTOR_ID = p_doctor_id
      AND APPOINTMENT_DATE = p_appointment_date;

    IF v_count = 0 THEN
        -- Insert new appointment note if it does not exist
        INSERT INTO HealthCareManagement_APPOINTMENT (PATIENT_ID, DOCTOR_ID, NOTE, APPOINTMENT_DATE)
        VALUES (p_patient_id, p_doctor_id, p_note, p_appointment_date);
    ELSE
        -- Update existing appointment note
        UPDATE HealthCareManagement_APPOINTMENT
        SET NOTE = p_note
        WHERE PATIENT_ID = p_patient_id
          AND DOCTOR_ID = p_doctor_id
          AND APPOINTMENT_DATE = p_appointment_date;
    END IF;

    COMMIT;

    RETURN 'Success'; --function updates NOTE but doesn't print success
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RETURN 'Error: ' || SQLERRM;
END;
/
```

- **Output:** Updated PAT001

PATIENT_ID	PATIENT_FI	PATIENT_LA	DOCTOR_ID	DOCTOR_FIR	DOCTOR_LAS	NOTE	APPOINTMEN
PAT005	Sophia	Williams	DOC005	Michael	Davis	Emergency Visit	2023-10-01
PAT004	Michael	Johnson	DOC004	Sophia	Brown	Annual Physical	2023-09-01
PAT003	Emily	Smith	DOC003	David	Williams	Consultation	2023-08-01
PAT002	John	Brown	DOC002	Emily	Johnson	Routine Checkup	2023-07-01
PAT001	Jane	Doe	DOC001	John	Smith	Updated follow-up note for demonstration	2023-06-01

DOCTOR: View/edit profile (Ellie)

- **Description:** A simple query filtered on the doctor's ID allows the doctor to view their profile. An SQL procedure takes in doctor's DOCTOR_ID along with updated LAST name, FIRST name, EMAIL, PASSWORD, SPECIALIZATION, and/or OFFICE_NUMBER and updates the doctor's profile. The DOCTOR_ID cannot be changed.
- **SQL Queries:** (file: Doctor_Edit_Profile)

```oracle_plsql
SELECT * FROM HealthCareManagement_DOCTOR WHERE DOCTOR_ID = 'DOC001';
```

```oracle
CREATE OR REPLACE PROCEDURE Edit_Doctor_Info(d_doctor_id IN CHAR,
                                             d_last IN VARCHAR,
                                             d_first IN VARCHAR,
                                             d_email IN VARCHAR,
                                             d_specialization IN VARCHAR,
                                             d_office_number IN CHAR,
                                             d_password IN CHAR)
AS
BEGIN
    -- Update the specified columns for the patient
    UPDATE HealthCareManagement_DOCTOR
    SET
        LAST = d_last,
        FIRST = d_first,
        EMAIL = d_email,
        PASSWORD = d_password,
        SPECIALIZATION = d_specialization,
        OFFICE_NUMBER = d_office_number
    WHERE DOCTOR_ID = d_doctor_id;

    COMMIT;

    -- Output success message
    DBMS_OUTPUT.PUT_LINE('Patient information updated successfully.');
EXCEPTION
    WHEN OTHERS THEN
        -- Output error message if an exception occurs
        DBMS_OUTPUT.PUT_LINE('Error updating patient information: ' || SQLERRM);
END;
```

- **Output:**

DOCTOR_ID	LAST	FIRST	EMAIL	PASSWORD	SPECIALIZATION	OFFICE_NUMBER
DOC001	Smith	John	john.smith@hospital.com	thsbaibniincd68n	Cardiology	101

```oracle
--TEST STATEMENTS:

SELECT * FROM HealthCareManagement_DOCTOR WHERE DOCTOR_ID = 'DOC001';
-- OUTPUT:
--DOCTOR_ID     LAST    FIRST   EMAIL                   PASSWORD            SPECIALIZATION  OFFICE_NUMBER
--DOC001        Smith   John    john.smith@hospital.com thsbaibniincd68n    Cardiology      101

-- Call the procedure to update the doctor's information
EXEC Edit_Doctor_Info('DOC001', 'Smith', 'Ellie', 'updated_email@example.com', 'Physicians Assistant', '102', 'newPassword');


SELECT * FROM HealthCareManagement_DOCTOR WHERE DOCTOR_ID = 'DOC001';
-- NEW OUTPUT WITH UPDATED CHANGES:
--DOCTOR_ID     LAST    FIRST   EMAIL                       PASSWORD        SPECIALIZATION          OFFICE_NUMBER
--DOC001        Smith   Ellie   updated_email@example.com   newPassword     Physicians Assistant     102
```

INSURANCE COMPANY: View patients they cover and how much each owes (Ellie)

- **Description:** A view was created to retrieve information about patients covered by the insurance company and their outstanding payments. Then, a procedure allows the insurance company to view the information from the view.
- **SQL Queries:** (file: Insurance_View_Patient_Balances)

```oracle
-- Create a view to retrieve information about patients covered by the insurance company and their outstanding payments
CREATE OR REPLACE VIEW Insurance_Company_Covered_Patients AS
SELECT      P.PATIENT_ID,
            P.LAST || ', ' || P.FIRST AS PATIENT_NAME,
            PC.INSURANCE_ID,
            SUM(PRICE) AS AMOUNT_OWED
FROM        HealthCareManagement_PATIENT P
            JOIN    HealthCareManagement_PAYSFOR PC ON P.INSURANCE_ID = PC.INSURANCE_ID
            JOIN    HealthCareManagement_PRESCRIPTION PR ON P.PATIENT_ID = PR.PATIENT_ID
GROUP BY    P.PATIENT_ID, P.LAST, P.FIRST, PC.INSURANCE_ID;

-- Create a procedure that allows the insurance company to view the information from the view
CREATE OR REPLACE PROCEDURE View_Covered_Patients_Information (i_insurance_id IN CHAR)
AS
BEGIN
    -- Retrieve information about patients covered by the insurance company and their outstanding payments
    FOR patient_record IN (SELECT * FROM Insurance_Company_Covered_Patients WHERE INSURANCE_ID = i_insurance_id) LOOP
        DBMS_OUTPUT.PUT_LINE('Patient ID: ' || patient_record.PATIENT_ID);
        DBMS_OUTPUT.PUT_LINE('Patient Name: ' || patient_record.PATIENT_NAME);
        DBMS_OUTPUT.PUT_LINE('Amount Owed: $' || patient_record.AMOUNT_OWED);
        DBMS_OUTPUT.PUT_LINE('------------------------');
    END LOOP;
END;
```

- **Output:**

```oracle
--TEST STATEMENTS:

SET SERVEROUTPUT ON;
EXEC View_Covered_Patients_Information('INS001');
-- OUTPUT:
--Patient ID: PAT001
--Patient Name: Doe, Jane
--Amount Owed: $25
--------------------------
```

INSURANCE COMPANY: Pay balance (Ellie)

- **Description:** An SQL function allows insurance companies to see unpaid balance on certain prescriptions when they input patient ID and prescription ID. When the company finds one and makes a payment, an SQL trigger updates the prescription price.
- **SQL Queries:** (file: Insurance_Make_Payment)

```oracle
-- THIS FUNCTION TAKES IN A PATIENT ID AND PRESCRIPTION ID AND RETURNS UNPAID BALANCE TO INSURANCE COMPANY
CREATE OR REPLACE FUNCTION GetPrescriptionPrice(p_patient_id IN HealthCareManagement_PRESCRIPTION.PATIENT_ID%TYPE,
                                                p_prescription_id IN HealthCareManagement_PRESCRIPTION.PRESCRIPTION_ID%TYPE) RETURN DECIMAL
AS
    v_prescription_price DECIMAL := 0;

BEGIN
    -- Retrieve the prescription price for the given patient ID and prescription ID
    SELECT PRICE
    INTO v_prescription_price
    FROM HealthCareManagement_PRESCRIPTION
    WHERE PATIENT_ID = p_patient_id AND PRESCRIPTION_ID = p_prescription_id;

    -- Return the prescription price
    RETURN v_prescription_price;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
```

```oracle
--THIS TRIGGER CHANGES THE PRESCRIPTION PRICE AFTER A PAYMENT HAS BEEN MADE
CREATE or REPLACE TRIGGER ChangePrescriptionPriceAfterPayment
    AFTER INSERT ON HealthCareManagement_PAYMENT
    For Each Row
BEGIN
    UPDATE  HealthCareManagement_PRESCRIPTION
    SET     PRICE=PRICE-:NEW.AMOUNT
    WHERE   PRESCRIPTION_ID=:NEW.PRESCRIPTION_ID;
END;
```

- **Output:**

```
-- FUNCTION TEST STATEMENTS:
SELECT * FROM HealthCareManagement_PRESCRIPTION WHERE PATIENT_ID='PAT001' AND PRESCRIPTION_ID='PRSC001';
--PRESCRIPTION_ID   DATE_ISSUED     PRESCRIPTION_NAME   DOSAGE  REFILLS_REMAINING   PRICE   QUANTITY    DOCTOR_ID   PATIENT_ID
-- PRSC001          01-JAN-23       Amoxicillin         500mg   05                  25      30          DOC001      PAT001
SELECT GetPrescriptionPrice('PAT001','PRSC001') from DUAL;
-- 25

--TRIGGER TEST STATEMENTS:
SELECT * FROM HealthCareManagement_PRESCRIPTION;
INSERT INTO HealthCareManagement_PAYMENT (PAYMENT_ID, PAYMENT_DATE, AMOUNT, INSURANCE_ID, PRESCRIPTION_ID)
        VALUES ('PAY001', TO_DATE('2023-06-15', 'YYYY-MM-DD'), 15.00, 'INS002', 'PRSC002');
SELECT * FROM HealthCareManagement_PRESCRIPTION;

--FIRST SELECT:
--PRESCRIPTION_ID   DATE_ISSUED     PRESCRIPTION_NAME   DOSAGE  REFILLS_REMAINING   PRICE   QUANTITY    DOCTOR_ID   PATIENT_ID
--PRSC001           01-JAN-23       Amoxicillin         500mg   05                  25      30          DOC001      PAT001
--PRSC002           01-FEB-23       Ibuprofen           200mg   03                  15      20          DOC002      PAT002
--PRSC003           01-MAR-23       Metformin           850mg   02                  30      60          DOC003      PAT003
--PRSC004           01-APR-23       Lisinopril          10mg    04                  22      90          DOC004      PAT004
--PRSC005           01-MAY-23       Atorvastatin        20mg    01                  45      10          DOC005      PAT005

--AFTER INSERT:
--PRESCRIPTION_ID   DATE_ISSUED     PRESCRIPTION_NAME   DOSAGE  REFILLS_REMAINING   PRICE   QUANTITY    DOCTOR_ID   PATIENT_ID
--PRSC001           01-JAN-23       Amoxicillin         500mg   05                  25      30          DOC001      PAT001
--PRSC002           01-FEB-23       Ibuprofen           200mg   03                  0       20          DOC002      PAT002
--PRSC003           01-MAR-23       Metformin           850mg   02                  30      60          DOC003      PAT003
--PRSC004           01-APR-23       Lisinopril          10mg    04                  22      90          DOC004      PAT004
--PRSC005           01-MAY-23       Atorvastatin        20mg    01                  45      10          DOC005      PAT005
```

INSURANCE COMPANY: View/edit profile (Ellie)

- **Description:** A simple query filtered on the insurance company's ID allows the insurance company to view their profile. An SQL procedure takes in insurance company's INSURANCE_ID along with updated INSURANCE_NAME, STREET, CITY, STATE, ZIP_CODE, PHONE_NUMBER, EMAIL, and/or PASSWORD and updates the insurance company's profile. The INSURANCE_ID cannot be changed.
- **SQL Queries:** (file: Insurance_Edit_Profile)

```
SELECT * FROM healthcaremanagement_insurancecompany WHERE INSURANCE_ID = 'INS001';
```

```oracle
CREATE OR REPLACE PROCEDURE Edit_Insurance_Info(i_insurance_id IN CHAR,
                                                i_insurance_name IN VARCHAR,
                                                i_street IN VARCHAR,
                                                i_city IN VARCHAR,
                                                i_state IN CHAR,
                                                i_zip_code IN CHAR,
                                                i_phone_number IN VARCHAR,
                                                i_email IN VARCHAR,
                                                i_password IN CHAR)
AS
BEGIN
    -- Update the specified columns for the insurance company
    UPDATE HealthCareManagement_INSURANCECOMPANY
    SET
        INSURANCE_NAME = i_insurance_name,
        STREET = i_street,
        CITY = i_city,
        STATE = i_state,
        ZIP_CODE = i_zip_code,
        PHONE_NUMBER = i_phone_number,
        EMAIL = i_email,
        PASSWORD = i_password
    WHERE INSURANCE_ID = i_insurance_id;

    COMMIT;

    -- Output success message
    DBMS_OUTPUT.PUT_LINE('Insurance company information updated successfully.');
EXCEPTION
    WHEN OTHERS THEN
        -- Output error message if an exception occurs
        DBMS_OUTPUT.PUT_LINE('Error updating insurance company information: ' || SQLERRM);
END;
```

- **Output:**

INSURANCE_ID	INSURANCE_NAME	STREET	CITY	STATE	ZIP_CODE	PHONE_NUMBER	EMAIL	PASSWORD
INS001	New Insurance	123 New St	New City	NY	54321	555-555-5555	info@newinsurance.com	newpassword

```oracle
--TEST STATEMENTS:

SELECT * FROM HealthCareManagement_INSURANCECOMPANY WHERE INSURANCE_ID = 'INS001';
-- OUTPUT:
--INSURANCE_ID  INSURANCE       STREET          CITY        ZIP_CODE    PHONE_NUMBER    EMAIL               PASSWORD
--INS001        HealthPlus      1234 Main St    Anytown NY  12345       123-456-7890    info@healthplus.com ths8673incd58n

-- Call the procedure to update the insurance company's information
EXEC Edit_Insurance_Info('INS001', 'New Insurance', '123 New St', 'New City', 'NY', '54321', '555-555-5555', 'info@newinsurance.com', 'newpassword');


SELECT * FROM HealthCareManagement_INSURANCECOMPANY WHERE INSURANCE_ID = 'INS001';
-- NEW OUTPUT WITH UPDATED CHANGES:
--INSURANCE_ID  INSURANCE       STREET          CITY        ZIP_CODE    PHONE_NUMBER    EMAIL               PASSWORD
--INS001        New Insurance   123 New St      New City    NY  54321   555-555-5555    info@newinsurance.com   newpassword
```