CS 480 - Term Project
Hospital Database

Names: Harsh Dasika, Kathleen Melonashi

Part 1 - Assumptions:

Cardinality Constraints:

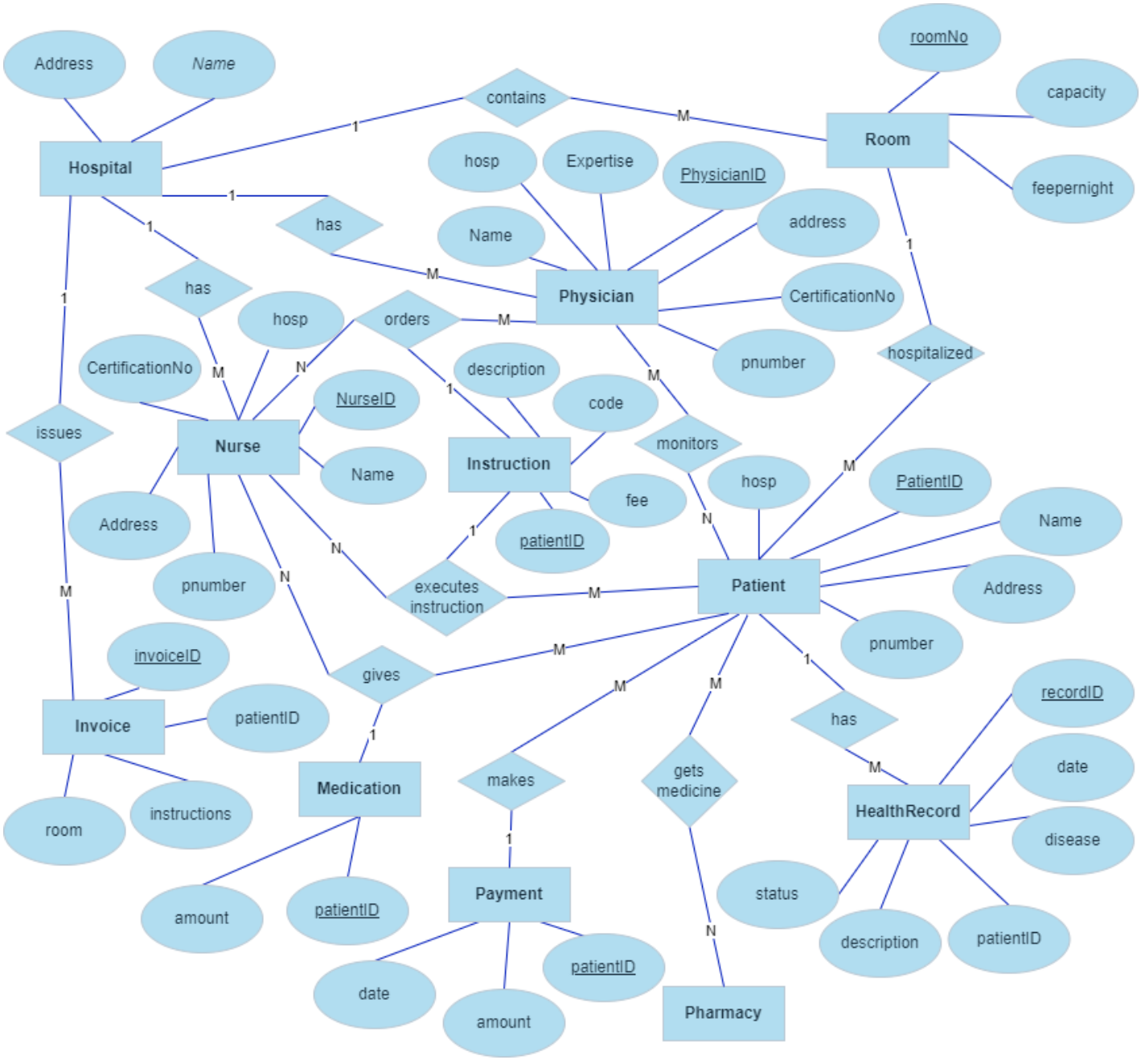
- Hospital and physician (1-to-Many)
- Hospital and nurse (1-to-Many)
- Hospital and room (1-to-Many)
- Hospital and invoice (1-to-Many)
- Physician and nurse (Many-to-Many)
- Physician and patient (Many-to-Many)
- Physician and instruction (1-to-Many)
- Patient and room (1-to-Many)
- Patient and health record (1-to-Many)
- Patient and pharmacy (Many-to-Many)
- Patient and payment (1-to-Many)
- Patient and nurse (Many-to-Many)
- Patient and Medication (1-to-Many)
- Nurse and Instruction (1-to-Many)

Assumption 1: We are assuming that the invoice, instruction, medication, and payment tables will contain the patientID so each of them can be linked to a specific patient.

Assumption 2: We created a table containing different hospital names meaning we added an additional attribute hosp for each physician, nurse and patient, which associates each member with the appropriate hospital.

Assumption 3: We included pharmacy in the EER design, however since we did not give it any attributes, therefore we did not include the table pharmacy in the hospital-scheme database.

Part 2 - (E)ER Design:



Part 3 - Relations and keys:

1. **Hospital**(name, address)
    a. Primary key: {name}

2. **Physician**(phys_id, name, cert_num, expertise, address, p_number, hosp)
    a. Primary key: {phys_id}
    b. Foreign key: {hosp references Hospital(name)}

3. **Nurse**(nurse_id, name, cert_num, address, p_number, hosp)
    a. Primary key: {nurse_id}
    b. Foreign key: {hosp references Hospital(name)}

4. **Patient**(id, name, address, p_number, hosp)
    a. Primary key: {id}
    b. Foreign key: {hosp references Hospital(name)}

5. **Room**(room_no, capacity, fee)
    a. Primary key: {room_no}

6. **HealthRecord**(record_id, patient_id, disease, date, status, description)
    a. Primary key: {record_id}
    b. Foreign key: {patient_id references Patient(id)}

7. **Instruction**(code, fee, patient_id, description)
    a. Primary key: {code}
    b. Foreign key: {patient_id references Patient(id)}

8. **Invoice**(invoice_id, patient_id, room, instructions)
    a. Primary key: {invoice_id}
    b. Foreign key: {patient_id references Patient(id), room references Room(room_no), instructions references Instruction(code)}

9. **Payments**(patient_id, date, amount)
    a. Foreign key: {patient_id references Patient(id)}

10. **Medication**(patient_id)
    a. Foreign key: {patient_id references Patient(id)}

Part 4 - Views and Descriptions

View 1: Display patient information along with their corresponding physician and nurse:

```
CREATE VIEW patient_info AS
SELECT p.id AS patient_id, p.name AS patient_name, p.address AS
patient_address, p.p_number AS patient_phone,
       h.name AS hospital_name,
       ph.name AS physician_name, ph.expertise AS physician_expertise,
       n.name AS nurse_name
FROM patient p
JOIN hospital h ON p.hosp = h.name
LEFT JOIN physician ph ON p.hosp = ph.hosp
LEFT JOIN nurse n ON p.hosp = n.hosp;
```

patient_id	patient_name	patient_address	patient_phone	hospital_name	physician_name	physician_expertise	nurse_name
1	Calum Calderon	28 Pacific Ave, Chicago	985-918-0465	Northwestern Memorial	Harvey Specter	Neurology	Joyce Lopez
1	Calum Calderon	28 Pacific Ave, Chicago	985-918-0465	Northwestern Memorial	Harvey Specter	Neurology	Elaine Harris
1	Calum Calderon	28 Pacific Ave, Chicago	985-918-0465	Northwestern Memorial	Mike Ross	Internal Medicine	Joyce Lopez
1	Calum Calderon	28 Pacific Ave, Chicago	985-918-0465	Northwestern Memorial	Mike Ross	Internal Medicine	Elaine Harris
2	Edwin Stone	224 E 65th St, Chicago	410-428-8203	UI Health	Jessica Pearson	Cardiology	John Keen
3	Rosalie Pacheco	18 Goodrich Ln, Chicago	312-608-5334	UChicago Medicine	Louis Litt	Pediatrics	Dolores Tatum
4	Leonard Klein	834 S Halsted St, Chicago	606-785-0259	Northwestern Memorial	Harvey Specter	Neurology	Joyce Lopez
4	Leonard Klein	834 S Halsted St, Chicago	606-785-0259	Northwestern Memorial	Harvey Specter	Neurology	Elaine Harris
4	Leonard Klein	834 S Halsted St, Chicago	606-785-0259	Northwestern Memorial	Mike Ross	Internal Medicine	Joyce Lopez
4	Leonard Klein	834 S Halsted St, Chicago	606-785-0259	Northwestern Memorial	Mike Ross	Internal Medicine	Elaine Harris
5	Anton Mosley	27 S Aberdeen St, Chicago	206-585-2999	Rush Medical Center	Rachel Zane	Dermatology	NULL

- Why is this view useful?

The view patient_info is important for the database because it provides easy access to patient-physician-nurse relationships. It helps with simplifying queries: Instead of writing complex queries using the entire database, patient_info shows all we need to find out about relationships between medics and patients.

View 2: Display patients names along with their invoices and total payment:

```
CREATE VIEW invoice_info AS
SELECT i.invoice_id, p.name AS patient_name, r.room_no, instr.code AS
instruction_code, i.instructions,
       (r.fee + instr.fee) AS total_amount
FROM invoice i
JOIN patient p ON i.patient_id = p.id
JOIN room r ON i.room = r.room_no
JOIN instruction instr ON i.instructions = instr.code;
```

invoice_id	patient_name	room_no	instruction_code	instructions	total_amount
304	Calum Calderon	123	423	423	2500
305	Calum Calderon	123	349	349	4500
592	Edwin Stone	124	126	126	15500
593	Edwin Stone	124	942	942	6000
324	Rosalie Pacheco	125	349	349	3500
876	Leonard Klein	126	568	568	10200
210	Anton Mosley	127	942	942	5100

- Why is this view useful?

This view holds invoice details in one place. This view contains multiple information from tables invoice, patient, room and instruction into one single set. We can use this view to retrieve information instead of using multiple JOIN statements. Users can use invoide_info to quickly understand the invoice data without needing to understand the underlying table relationships. This view also allows users to use attributes such as total amount by combining fees and instructions table.

View 3: View to display patient's name along with their corresponding diseases:

```
CREATE VIEW healthrecord_info AS
SELECT h.record_id, p.name AS patient_name, h.disease, h.date,
h.status, h.description
FROM healthrecord h
JOIN patient p ON h.patient_id = p.id;
```

record_id	patient_name	disease	date	status	description
1	Calum Calderon	Covid-19	2023-02-26	Recovering	Contracted Covid-19, currently recovering
2	Edwin Stone	Coronary Artery Disease	2022-12-02	Critical	Plaque buildup in arteries, condition critical
3	Rosalie Pacheco	Pneumonia	2023-04-14	Stable	Cold turned into pneumonia, now stable
6	Rosalie Pacheco	Covid-19	2023-05-24	Stable	Exposed to covid
4	Leonard Klein	Epilepsy	2021-01-17	Critical	Medication not stopping seizures anymore
7	Leonard Klein	Emotional changes	2022-02-14	Critical	Experiencing intense emotions of fear, anxiety
5	Anton Mosley	Psoriasis	2023-06-13	Remission	Symptoms vanishing, needs monitoring

- Why is this view useful?

This view joins together the patient and healthrecord table. By using this view, users can avoid once again using the JOIN statement, however this view improves readability. There is no risk of displaying health records for the wrong patients.

View 4: View to display all the hospitals in the database, the number of physicians, nurses and patients:

```
CREATE VIEW hospital_summary AS
SELECT h.name AS hospital_name,
       COUNT(DISTINCT ph.phys_id) AS total_physicians,
       COUNT(DISTINCT n.nurse_id) AS total_nurses,
       COUNT(DISTINCT p.id) AS total_patients
FROM hospital h
LEFT JOIN physician ph ON h.name = ph.hosp
LEFT JOIN nurse n ON h.name = n.hosp
LEFT JOIN patient p ON h.name = p.hosp
GROUP BY h.name;
```

hospital_name	total_physicians	total_nurses	total_patients
Ascension Saint Francis	0	1	0
Northwestern Memorial	2	2	2
Rush Medical Center	1	0	1
UChicago Medicine	1	1	1
UI Health	1	1	1

- Why is this view useful?

Although this is a small database and there is no room for confusion, when it comes to big organizations such as hospitals the databases are huge and the error rate when it comes to computing queries is higher. When given such as view as hospital_summary it is easy to see the numbers we are trying to find.

View 5: View to display a concise summary of the patients' current health status, including latest disease, status and description:

```
CREATE VIEW patient_health_status AS
SELECT p.id AS patient_id, p.name AS patient_name, p.hosp AS
hospital_name,
       hr.disease AS latest_disease, hr.status AS latest_status,
hr.description AS latest_description
FROM patient p
LEFT JOIN healthrecord hr ON p.id = hr.patient_id
```

```
WHERE (hr.patient_id, hr.date) IN
    (SELECT patient_id, MAX(date) FROM healthrecord GROUP BY
patient_id);
```

patient_id	patient_name	hospital_name	latest_disease	latest_status	latest_description
1	Calum Calderon	Northwestern Memorial	Covid-19	Recovering	Contracted Covid-19, currently recovering
2	Edwin Stone	UI Health	Coronary Artery Disease	Critical	Plaque buildup in arteries, condition critical
3	Rosalie Pacheco	UChicago Medicine	Covid-19	Stable	Exposed to covid
4	Leonard Klein	Northwestern Memorial	Emotional changes	Critical	Experiencing intense emotions of fear, anxiety
5	Anton Mosley	Rush Medical Center	Psoriasis	Remission	Symptoms vanishing, needs monitoring

- Why is this view useful?

The patient_health_status view is useful for the database because it offers a quick and easy-to-access overview of each patient's current health status, enabling medical professionals to make informed decisions, monitor patient conditions. This view consolidates essential health information, streamlines data retrieval, and facilitates timely patient assessments, contributing to improved patient care and medical decision-making.

Part 5 - Triggers and Descriptions

Trigger 1: This trigger automatically creates a new invoice everytime a new health record is inserted for a patient.

```
DELIMITER //
CREATE TRIGGER new_invoice AFTER INSERT ON healthrecord
FOR EACH ROW
BEGIN
    INSERT INTO invoice (patient_id, room, instructions)
    VALUES (NEW.patient_id, (SELECT MAX(room_no) FROM room), (SELECT
MAX(code) FROM instruction WHERE patient_id = NEW.patient_id));
END;
//
DELIMITER ;
```

- Why is this trigger useful?

This trigger is useful because it automatically creates an invoice for the patient as soon as their new record is added to Healthrecord. This cuts down on the manual effort needed to create an invoice otherwise.

Trigger 2: This trigger checks if a patient has already been added to the patients table and stops the insert if the patient is already there.

```
DELIMITER //
CREATE TRIGGER check_dup_patient
BEFORE INSERT ON patient
FOR EACH ROW
BEGIN
    IF EXISTS (SELECT 1 FROM patient WHERE id = NEW.id) THEN
        SIGNAL SQLSTATE '45000'
    END IF;
END;
//
DELIMITER ;
```

- Why is this trigger useful?

This trigger is useful because it checks if a patient has already been added before inserting them to the patients table. This is necessary because we want each patient to have a unique patientID, and having duplicates of a patient will violate that.

Trigger 3: This trigger sets a new patient's health status to "Pending" before they can be "checked" by a physician.

```
DELIMITER //
CREATE TRIGGER set_pending_status
AFTER INSERT ON patient
FOR EACH ROW
BEGIN
    INSERT INTO healthrecord (record_id, patient_id, disease, date, status, description)
    VALUES (NULL, NEW.id, 'Not confirmed', DATE(NOW()), 'Pending', 'Diagnosis not available
yet');
END;
//
DELIMITER ;
```

- Why is this trigger useful?

This trigger is useful because new patients that have no diagnosis automatically can obviously have no health status in the system, so they must be identified as such. It can also help when going back and seeing which patients have not been diagnosed yet.

Part 6 - Queries, Descriptions, and Results

NOTE: We created some queries using the views we created above. In order to run those queries the view needs to be created first.

1. List all of the patients names and their corresponding disease from the healthrecord table: (Join query 1)

```
SELECT p.name AS patient_name, h.disease
FROM patient p
JOIN healthrecord h ON p.id = h.patient_id;
```

	patient_name	disease
▶	Calum Calderon	Covid-19
	Edwin Stone	Coronary Artery Disease
	Rosalie Pacheco	Pneumonia
	Leonard Klein	Epilepsy
	Anton Mosley	Psoriasis

2. Find the patient who has the highest total payment amount: (Join query 2)

```
SELECT p.name AS patient_name, SUM(amount) AS total_payment
FROM patient p
JOIN payments pm ON p.id = pm.patient_id
GROUP BY p.name
ORDER BY total_payment DESC
LIMIT 1;
```

	patient_name	total_payment
▶	Edwin Stone	14500

3. List all the patients who were hospitalized in a room whose fee was more than 300 but less than 1300: (Join query 3)

```
SELECT DISTINCT p.name AS patient_name, r.room_no, r.fee
FROM patient p
JOIN invoice i ON p.id = i.patient_id
JOIN room r ON i.room = r.room_no
WHERE r.fee > 300 AND r.fee < 1300;
```

	patient_name	room_no	fee
▶	Edwin Stone	124	1000
	Rosalie Pacheco	125	500

4. Calculate the average of all rooms in the hospital: (Aggregation query 1)

```
SELECT AVG(fee) AS avg_fee
FROM room;
```

	avg_fee
▶	660.0000

5. Find the hospital name and number of physicians working in each hospital who have expertise in Cardiology: (Aggregation query 2)

```
SELECT h.name AS hospital_name, COUNT(*) AS num_cardiologists
FROM hospital h
JOIN physician p ON h.name = p.hosp
WHERE p.expertise = 'Cardiology'
GROUP BY h.name;
```

	hospital_name	num_cardiologists
▶	UI Health	1

6. Find the maximum and minimum fees paid by patients. (Aggregation Query 3)

```
SELECT MAX(amount) AS MaxAmountPaid, MIN(amount) AS MinAmountPaid
FROM payments;
```

	MaxAmountPaid	MinAmountPaid
▶	14500	1000

7. Find the names of the patients who have a health record status as critical: (Nested Query 1)

```
SELECT name AS patient_name
FROM patient
WHERE id IN (
    SELECT patient_id
    FROM healthrecord
    WHERE status = 'Critical'
);
```

	patient_name
▶	Edwin Stone
	Leonard Klein

8. Calculate the total and average amount paid by all patients at Northwestern Memorial. (Nested Query 2)

```
SELECT SUM(amount) AS Total, AVG(amount) AS Average
FROM Hospital.payments
WHERE Payments.patient_id IN (
    SELECT Patient.id
    FROM Hospital.patient
    WHERE Patient.hosp = 'Northwestern Memorial'
);
```

	Total	Average
▶	11000	5500.0000

9. Calculate the average payment amount for patients who have more than one health record: (Nested Query 3)

```
SELECT AVG(avg_payment) AS average_payment_multiple_records
FROM (
    SELECT patient_id, AVG(amount) AS avg_payment
    FROM payments
    WHERE patient_id IN (
        SELECT patient_id
        FROM healthrecord
        GROUP BY patient_id
        HAVING COUNT(*) > 1
    )
    GROUP BY patient_id
) AS multiple_records_payments;
```

	average_payment_multiple_records
▶	6500.00000000

10. Find the hospital with the highest total fees from all invoices in the database:

```
SELECT hosp.name AS hospital_name, SUM(inv.instructions) AS total_fees
FROM hospital hosp
LEFT JOIN patient p ON hosp.name = p.hosp
LEFT JOIN (
    SELECT patient_id, SUM(fee) AS instructions
    FROM instruction
    GROUP BY patient_id
) inv ON p.id = inv.patient_id
GROUP BY hosp.name
ORDER BY total_fees DESC
LIMIT 1;
```

	hospital_name	total_fees
▶	UI Health	14500

11. Find the patients who have been treated for the same disease by multiple physicians:

```
SELECT DISTINCT p.name AS patient_name, hr.disease AS treated_disease
FROM patient p
JOIN healthrecord hr ON p.id = hr.patient_id
JOIN physician ph ON p.hosp = ph.hosp
GROUP BY p.id, hr.disease
HAVING COUNT(DISTINCT ph.phys_id) > 1;
```

patient_name	treated_disease
Calum Calderon	Covid-19
Leonard Klein	Emotional changes
Leonard Klein	Epilepsy

12. Find the patients who have been admitted to the same room more than once and display the details on their stay:

```
SELECT p.name AS patient_name, r.room_no, COUNT(inv.invoice_id) AS stays_count
FROM patient p
JOIN invoice inv ON p.id = inv.patient_id
JOIN room r ON inv.room = r.room_no
GROUP BY p.name, r.room_no
HAVING COUNT(inv.invoice_id) > 1;
```

patient_name	room_no	stays_count
Calum Calderon	123	2
Edwin Stone	124	2

13. Find the total amount paid by patients whose status was marked “Critical.” (Q7 related)

```
SELECT SUM(amount) AS TotalFromCrit
FROM (
    SELECT Healthrecord.patient_id, Payments.amount
    FROM Hospital.healthrecord
    JOIN Payments on Healthrecord.patient_id = Payments.patient_id
    WHERE Healthrecord.status = 'Critical'
) as sumCritical;
```

	TotalFromCrit
▶	34500

14. Find total amount for each patient using the invoice_info view:

```
SELECT patient_name, SUM(total_amount) AS total_amount_for_patient
FROM invoice_info
GROUP BY patient_name;
```

	patient_name	total_amount_for_patient
▶	Calum Calderon	7000
	Edwin Stone	21500
	Rosalie Pacheco	3500
	Leonard Klein	10200
	Anton Mosley	5100

15. Find all patients who have Harvey Specter as their doctor using patient_info view:

```
SELECT patient_name, patient_address, patient_phone, hospital_name,
nurse_name
FROM patient_info
WHERE physician_name = 'Harvey Specter';
```

	patient_name	patient_address	patient_phone	hospital_name	nurse_name
▶	Calum Calderon	28 Pacific Ave, Chicago	985-918-0465	Northwestern Memorial	Joyce Lopez
	Calum Calderon	28 Pacific Ave, Chicago	985-918-0465	Northwestern Memorial	Elaine Harris
	Leonard Klein	834 S Halsted St, Chicago	606-785-0259	Northwestern Memorial	Joyce Lopez
	Leonard Klein	834 S Halsted St, Chicago	606-785-0259	Northwestern Memorial	Elaine Harris

16. Find the total amount patients have to pay if 20% of the payment is covered by the hospital using invoice_info view:

```
SELECT patient_name, SUM(total_amount) * 0.8 AS patient_amount_to_pay
FROM invoice_info
GROUP BY patient_name;
```

patient_name	patient_amount_to_pay
Calum Calderon	5600.0
Edwin Stone	17200.0
Rosalie Pacheco	2800.0
Leonard Klein	8160.0
Anton Mosley	4080.0

17. Find the number of patients who are in a critical condition and set the total amount to pay to 0 joining 2 created views. (invoice_info and patient_health_status):

```
SELECT p.patient_id, p.patient_name, p.hospital_name, p.latest_disease,
p.latest_status,
       p.latest_description,
       CASE WHEN p.latest_status = 'Critical' THEN 0 ELSE i.total_amount
END AS total_amount_to_pay
FROM patient_health_status p
LEFT JOIN invoice_info i ON p.patient_name = i.patient_name;
```

patient_id	patient_name	hospital_name	latest_disease	latest_status	latest_description	total_amount_to_pay
1	Calum Calderon	Northwestern Memorial	Covid-19	Recovering	Contracted Covid-19, currently recovering	2500
1	Calum Calderon	Northwestern Memorial	Covid-19	Recovering	Contracted Covid-19, currently recovering	4500
2	Edwin Stone	UI Health	Coronary Artery Disease	Critical	Plaque buildup in arteries, condition critical	0
2	Edwin Stone	UI Health	Coronary Artery Disease	Critical	Plaque buildup in arteries, condition critical	0
3	Rosalie Pacheco	UChicago Medicine	Covid-19	Stable	Exposed to covid	3500
4	Leonard Klein	Northwestern Memorial	Emotional changes	Critical	Experiencing intense emotions of fear, anxiety	0
5	Anton Mosley	Rush Medical Center	Psoriasis	Remission	Symptoms vanishing, needs monitoring	5100

Part 7 (Transactions and Descriptions) on next page →

Part 7 - Transactions and Descriptions

Transaction 1: Create a transaction to insert a new patient and their health record into the database:

```
START TRANSACTION;

INSERT INTO patient (id, name, address, p_number, hosp)
VALUES (6, 'Isabella Garcia', '123 Main St, Chicago', '312-555-7890',
'UChicago Medicine');

INSERT INTO healthrecord (record_id, patient_id, disease, date,
status, description)
VALUES (8, 6, 'Asthma', '2023-07-28', 'Stable', 'Diagnosed with
asthma');

COMMIT;
```

id	name	address	p_number	hosp
1	Calum Calderon	28 Pacific Ave, Chicago	985-918-0465	Northwestern Memorial
2	Edwin Stone	224 E 65th St, Chicago	410-428-8203	UI Health
3	Rosalie Pacheco	18 Goodrich Ln, Chicago	312-608-5334	UChicago Medicine
4	Leonard Klein	834 S Halsted St, Chicago	606-785-0259	Northwestern Memorial
5	Anton Mosley	27 S Aberdeen St, Chicago	206-585-2999	Rush Medical Center
6	Isabella Garcia	123 Main St, Chicago	312-555-7890	UChicago Medicine

record_id	patient_id	disease	date	status	description
3	3	Pneumonia	2023-04-14	Stable	Cold turned into pneumonia, now stable
4	4	Epilepsy	2021-01-17	Critical	Medication not stopping seizures anymore
5	5	Psoriasis	2023-06-13	Remission	Symptoms vanishing, needs monitoring
6	3	Covid-19	2023-05-24	Stable	Exposed to covid
7	4	Emotional changes	2022-02-14	Critical	Experiencing intense emotions of fear, anxiety
8	6	Asthma	2023-07-28	Stable	Diagnosed with asthma

- Why is this transaction useful?

The transaction provided for the database is useful because it ensures data consistency and integrity by adhering to the ACID properties. By treating the transaction as a single unit of work (atomicity), it guarantees that either all statements succeed or none do. The transaction enforces integrity constraints, keeping the database in a consistent state (consistency), while isolating concurrent operations to avoid data corruption (isolation). Once committed, the transaction's changes become permanent and survive system failures (durability). This reliability is crucial for

accurate and reliable hospital management systems, ensuring patient care and medical records are handled with precision and accuracy.

Transaction 2: Create a transaction to update a patient's contact number and create a new record for the invoice:

```
START TRANSACTION;

UPDATE patient
SET p_number = '312-555-1234'
WHERE id = 3;

INSERT INTO payments (patient_id, date, amount)
VALUES (3, '2023-07-28', 1500);

COMMIT;
```

id	name	address	p_number	hosp
1	Calum Calderon	28 Pacific Ave, Chicago	985-918-0465	Northwestern Memorial
2	Edwin Stone	224 E 65th St, Chicago	410-428-8203	UI Health
3	Rosalie Pacheco	18 Goodrich Ln, Chicago	312-555-1234	UChicago Medicine
4	Leonard Klein	834 S Halsted St, Chicago	606-785-0259	Northwestern Memorial
5	Anton Mosley	27 S Aberdeen St, Chicago	206-585-2999	Rush Medical Center
6	Isabella Garcia	123 Main St, Chicago	312-555-7890	UChicago Medicine

patient_id	date	amount
1	2023-02-27	1000
2	2022-12-10	14500
3	2023-04-17	3000
4	2021-01-23	10000
5	2023-06-13	5000
3	2023-07-28	1500

- Why is this transaction useful?

This transaction is useful because it ensures data integrity and consistency while handling important patient information and financial records in a hospital management system. The use of a transaction ensures that both the patient's contact update and payment record insertion are treated as a single unit, either succeeding entirely or not at all.