HOSPITAL DATABASE

CS 480 Spring 2023 Final Project

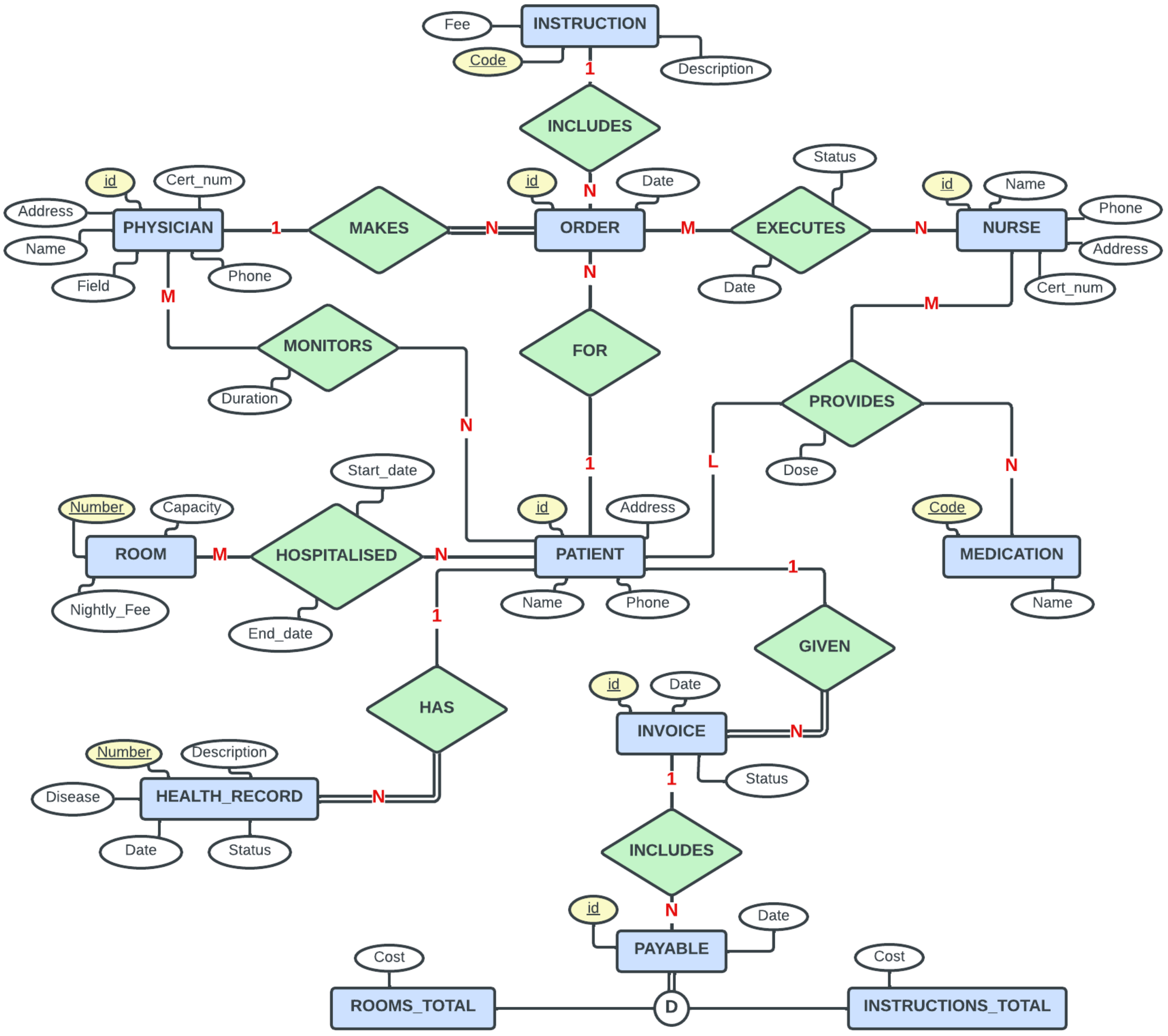
Raka Primardika (hybrid)
Russel Tjahjadi (in-person)

Assumptions.

1. A physician can order many instructions to many patients.

ER Diagram.

Here's our ER Diagram:



Relational Mapping.

```
Patient(id, name, address, phone_num)
Primary key: {id}
Foreign key: N/A
```

```
Physician(id, name, cert_num, address, field, phone_num)
Primary key: {id}
Foreign key: N/A
```

```
Nurse(id, name, cert_num, address, phone_num)
Primary key: {id}
Foreign key: N/A
```

```
Medication(code, name)
Primary key: {code}
Foreign key: N/A
```

```
Room(number, capacity, fee_nightly)
Primary key: {number}
Foreign key: N/A
```

```
Instruction(code, description, fee)
Primary key: {code}
Foreign key: N/A
```

```
Health_Record(id, pat_id, status, description, disease, date)
Primary key: {id}
Foreign key: {
  pat_id references Patient(id)
}
```

```
Monitors(phy_id, pat_id, duration)
Primary key: {phy_id, pat_id}
Foreign key: {
  phy_id references Physician(id),
  pat_id references Patient(id)
}
```

```
Hospitalised(pat_id, room_num, start_date, end_date)
Primary key: {pat_id, room_num}
Foreign key: {
  pat_id references Patient(id),
  Room_num references Room(number)
}
```

```
Provide(nur_id, pat_id, med_code, dose)
Primary key: {nur_id, pat_id, med_code}
Foreign key: {
  pat_id references Patient(id),
  nur_id references Nurse(id),
  med_code references Medication(code)
}
```

```
Invoice(id, pat_id, status, date)
Primary key: {id}
Foreign key: {
  pat_id references Patient(id)
}
```

```
Doctor_order(id, pat_id, phy_id, ins_code, date)
Primary key: {id}
Foreign key: {
  pat_id references Patient(id),
  phy_id references Physician(id),
  ins_code references Instruction(code)
}
```

```
Executes(ord_id, nur_id, status, date)
Primary key: {ord_id, nur_id}
Foreign key: {
  ord_id references Order(id),
  nur_id references Nurse(id),
}
```

```
Payable(id, inv_id, date, type, amount)
Primary key: {id}
Foreign key: {
  inv_id references Invoice(id)
}
```

Note: The `Order` entity is reflected as `Doctor_order` in our schema because it clashes with an SQL keyword.

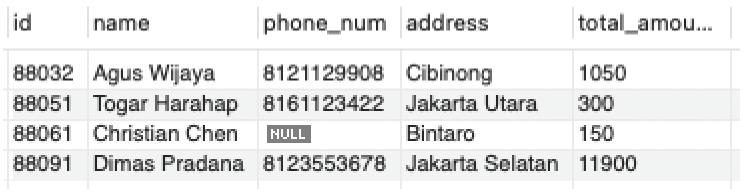
Views and Descriptions.

For the sake of organisation, each of our 3 views will be put on a separate page.

1. Unpaid invoices

This view is useful for finding out which patients haven't paid their invoices in full. So, this view lists all unpaid invoices, along with information about the patient and the aggregate sum of the cost. The hope is that it helps provide contact information to ask the patient to pay promptly.

```sql
CREATE VIEW Unpaid_invoices AS
SELECT I.id, name, phone_num, address, SUM(amount) AS total_amount
FROM Invoice I
    JOIN Payable P ON I.id = P.inv_id
    JOIN Patient T ON I.pat_id = T.id
WHERE invoice_status <> "Paid"
GROUP BY I.id;
```

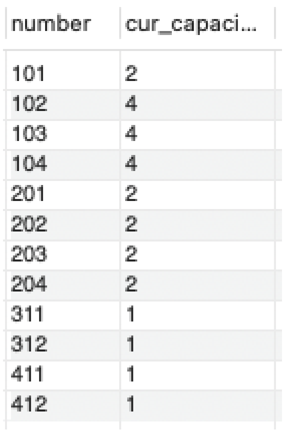


id	name	phone_num	address	total_amou...
88032	Agus Wijaya	8121129908	Cibinong	1050
88051	Togar Harahap	8161123422	Jakarta Utara	300
88061	Christian Chen	NULL	Bintaro	150
88091	Dimas Pradana	8123553678	Jakarta Selatan	11900

2. Available Rooms

This view lists all rooms with at least 1 available bed at the moment (hence the `NOW()`). Alongside that, it provides information on the number of unoccupied beds for each room. This is calculated by counting the number of patients currently in the room and subtracting that from the room capacity. The idea is that hospital employees can quickly check available rooms to match a patient's needs.

```sql
CREATE VIEW Available_rooms AS
SELECT number, capacity - COALESCE(num_occupants, 0) AS cur_capacity
-- Ref: https://www.w3schools.com/sql/func_sqlserver_coalesce.asp
FROM Room R LEFT JOIN
    (SELECT room_num, COUNT(pat_id) AS num_occupants
    FROM Hospitalised
    WHERE NOW() >= start_date AND NOW() <= end_date
    GROUP BY room_num) AS CR ON R.number = CR.room_num
WHERE capacity - COALESCE(num_occupants, 0) > 0;
```



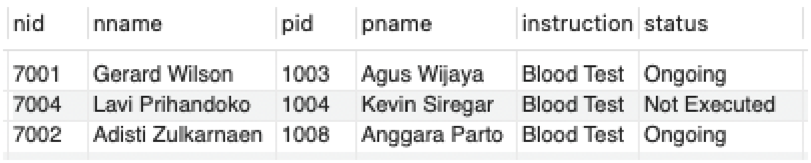
number	cur_capaci...
101	2
102	4
103	4
104	4
201	2
202	2
203	2
204	2
311	1
312	1
411	1
412	1

Note: For the example output, we had to manually change the `NOW()` to `2023-04-04` so that some data is shown. None of our data has patients hospitalised on the date we did this project.

3. Pending orders

This view provides concise information on the orders that have not been fully completed. Imagine a scenario in the emergency department where there are multiple orders and patients. This view can keep track of what hasn't been done, and provides the names of the nurses who are assigned to work on them as well.

```sql
CREATE VIEW Pending_orders AS
SELECT  N.id AS nid, N.name AS nname,
        P.id AS pid, P.name AS pname,
        I.description AS instruction,
        executes_status AS status
FROM Doctor_order O
    JOIN Executes E ON O.id = E.ord_id
    JOIN Instruction I ON O.ins_code = I.code
    JOIN Nurse N ON N.id = E.nur_id
    JOIN Patient P ON P.id = O.pat_id
WHERE executes_status <> "Completed"
```



nid	nname	pid	pname	instruction	status
7001	Gerard Wilson	1003	Agus Wijaya	Blood Test	Ongoing
7004	Lavi Prihandoko	1004	Kevin Siregar	Blood Test	Not Executed
7002	Adisti Zulkarnaen	1008	Anggara Parto	Blood Test	Ongoing

Triggers and Descriptions.

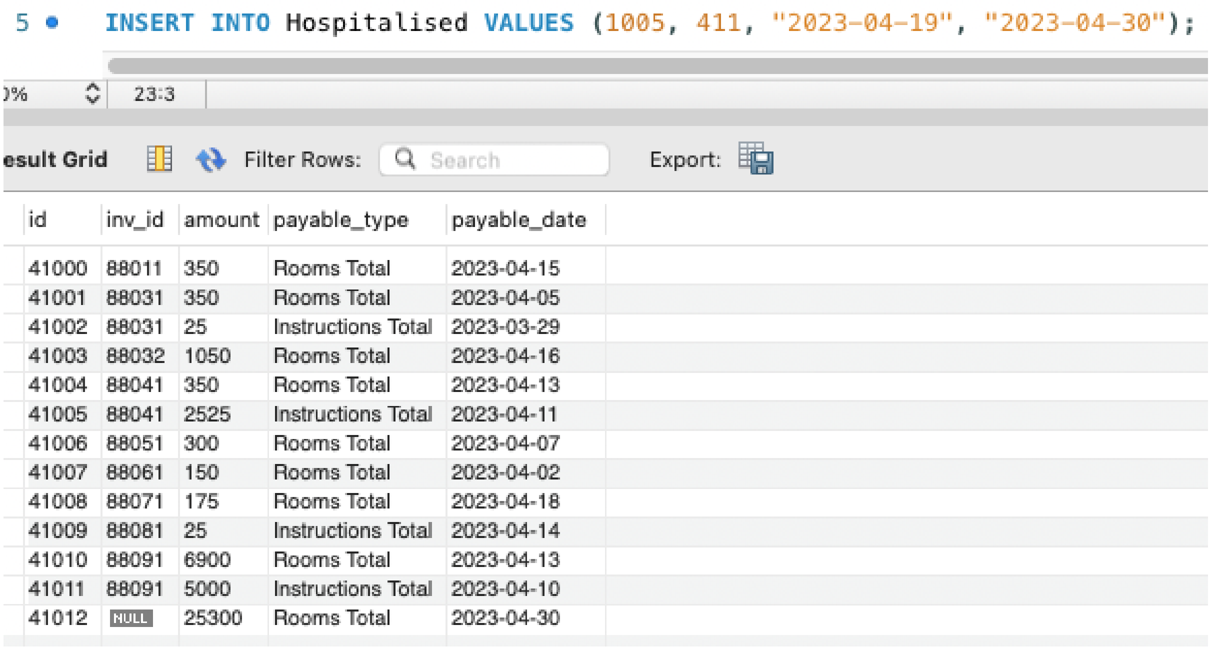
For the sake of organisation, each of our 3 triggers will be put on a separate page.

1. Create payable for each room booking

This view makes it convenient for calculating the cost of rooms. What this trigger does is automatically create a payable item for the room booking. It grabs the nightly fee for the room and multiplies it by the number of nights booked.

```
CREATE TRIGGER create_room_payable
AFTER INSERT ON Hospitalised
FOR EACH ROW
INSERT INTO Payable(inv_id, payable_type, amount, payable_date)
VALUES (
    NULL,
    "Rooms Total",
    (DATEDIFF(NEW.end_date, NEW.start_date))
    *
    (SELECT fee_nightly FROM Room WHERE number = NEW.room_num),
    CAST(NOW() AS DATE)
);
```

Example run and result:



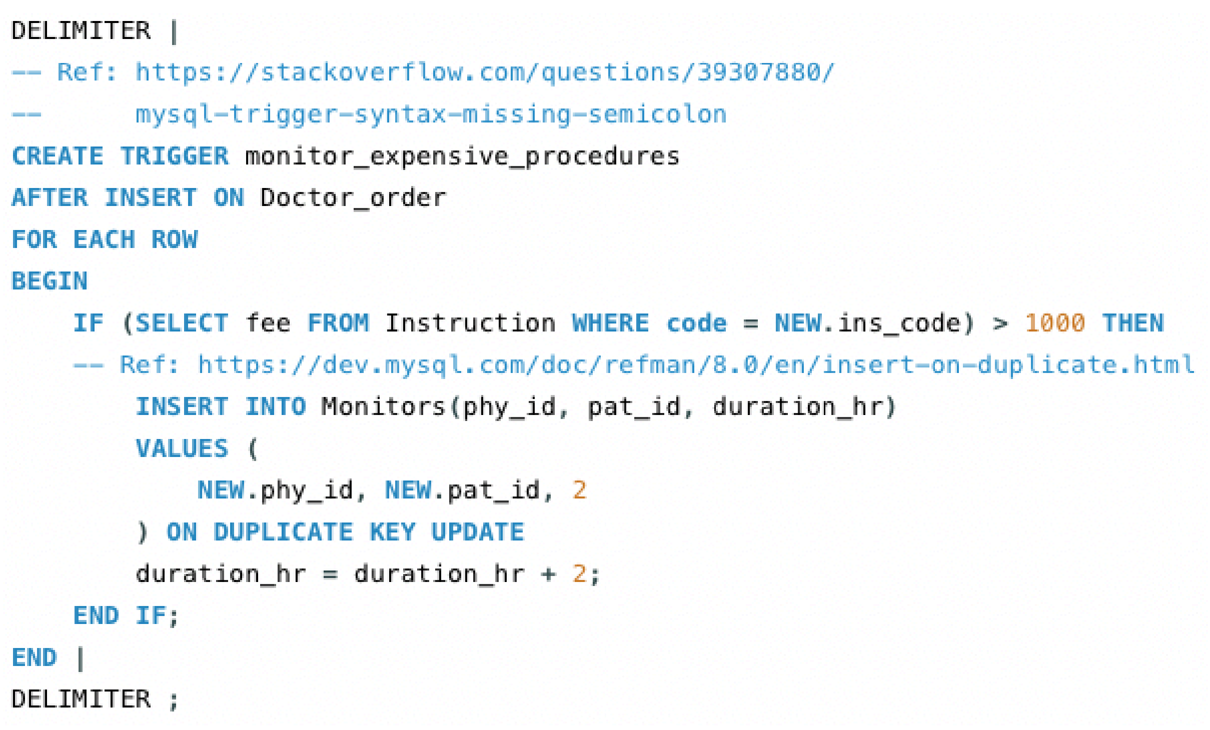
```
INSERT INTO Hospitalised VALUES (1005, 411, "2023-04-19", "2023-04-30");
```

id	inv_id	amount	payable_type	payable_date
41000	88011	350	Rooms Total	2023-04-15
41001	88031	350	Rooms Total	2023-04-05
41002	88031	25	Instructions Total	2023-03-29
41003	88032	1050	Rooms Total	2023-04-16
41004	88041	350	Rooms Total	2023-04-13
41005	88041	2525	Instructions Total	2023-04-11
41006	88051	300	Rooms Total	2023-04-07
41007	88061	150	Rooms Total	2023-04-02
41008	88071	175	Rooms Total	2023-04-18
41009	88081	25	Instructions Total	2023-04-14
41010	88091	6900	Rooms Total	2023-04-13
41011	88091	5000	Instructions Total	2023-04-10
41012	NULL	25300	Rooms Total	2023-04-30

The last entry (41012) was inserted to the Payables table because a new booking was inserted into the Hospitalised table.

2. Monitor expensive procedures

Some instructions are expensive. The hospital policy mandates that physicians who order procedures that cost more than 1000 have to monitor the patient for at least two hours. This trigger is activated upon an order made by a physician for a patient. If the cost of the order is more than 1000, automatically add two more hours to the Monitors table for the physician-patient.



```
DELIMITER |
-- Ref: https://stackoverflow.com/questions/39307880/
--      mysql-trigger-syntax-missing-semicolon
CREATE TRIGGER monitor_expensive_procedures
AFTER INSERT ON Doctor_order
FOR EACH ROW
BEGIN
    IF (SELECT fee FROM Instruction WHERE code = NEW.ins_code) > 1000 THEN
    -- Ref: https://dev.mysql.com/doc/refman/8.0/en/insert-on-duplicate.html
        INSERT INTO Monitors(phy_id, pat_id, duration_hr)
        VALUES (
            NEW.phy_id, NEW.pat_id, 2
        ) ON DUPLICATE KEY UPDATE
        duration_hr = duration_hr + 2;
    END IF;
END |
DELIMITER ;
```

Note: There was a weird error where our SQL workspaces didn't register the semicolons properly. That's why the "delimiter" clauses exist, as a workaround.

3. Unassign monitoring physician upon paid invoice

While this situation does not make sense in the real world, let's say our hospital would like to automatically free up all physicians monitoring a patient once the patient has paid their invoice. Upon an update to an invoice saying that it's now "Paid", all entries in the monitoring table with the patient listed in the invoice will be removed.

```
DELIMITER |
CREATE TRIGGER Unassign_monitoring
AFTER UPDATE ON Invoice
FOR EACH ROW
BEGIN
    IF (NEW.invoice_status = "Paid"
    AND OLD.invoice_status <> "Paid") THEN
        DELETE FROM Monitors
        WHERE pat_id = NEW.pat_id;
    END IF;
END |
DELIMITER ;
```

Queries, Results and Descriptions.

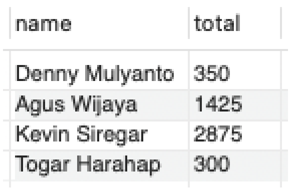
For the sake of organisation, each of our 15 queries will be put on a separate page.

Num	Query	Join	Aggregate	Nested
1	People covered by insurance	Yes	Yes	Yes
2	Most common administered med	Yes	Yes	-
3	Unused rooms in April 2023	-	-	Yes
4	Proper instruction orders?	Yes	-	-
5	Instruction with the most expense	Yes	Yes	-
6	Patients with unpaid invoices	Yes	-	-
7	Instructions assigned physicians	Yes	-	-
8	Omitting cheap instructions	Yes	Yes	-
9	Number of staff	-	Yes	Yes
10	Patients with multiple orders	Yes	Yes	-
11	Patients with records monitored?	Yes	-	Yes
12	Total revenue from rooms	-	Yes	-
13	Contact missing nurses	Yes	-	-
14	Detecting favouritism of nurses	Yes	Yes	-
15	Total revenue already paid	Yes	Yes	-

1. People covered by insurance

An insurance company is willing to cover some patients that didn't pay for VIP treatment. Find the names and invoice amounts of patients that are billed at least a total of USD 300 but didn't book rooms on the VIP 4th floor.

```sql
SELECT name, total
FROM Patient T, (SELECT pat_id, SUM(amount) AS total
        FROM Invoice I
            JOIN Payable P ON I.id = P.inv_id
        WHERE pat_id NOT IN (SELECT pat_id
                            FROM Hospitalised
                            WHERE room_num > 400)
        GROUP BY pat_id
        HAVING total >= 300) AS S
WHERE pat_id = T.id;
```

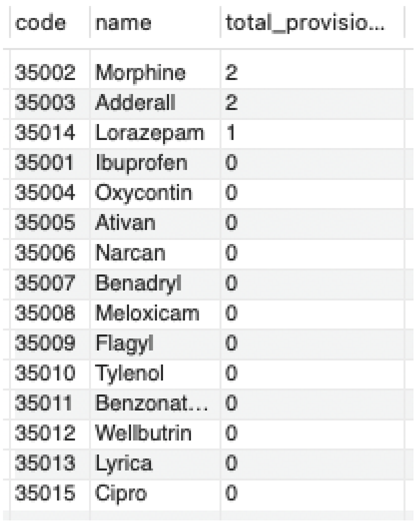


name	total
Denny Mulyanto	350
Agus Wijaya	1425
Kevin Siregar	2875
Togar Harahap	300

2. Most common administered medicine

In order for the hospital to make the logistics more efficient, we would like to identify the most common medications given. Find the codes, names, and total provisions of the medications, ordered by the number of times they have been provided.

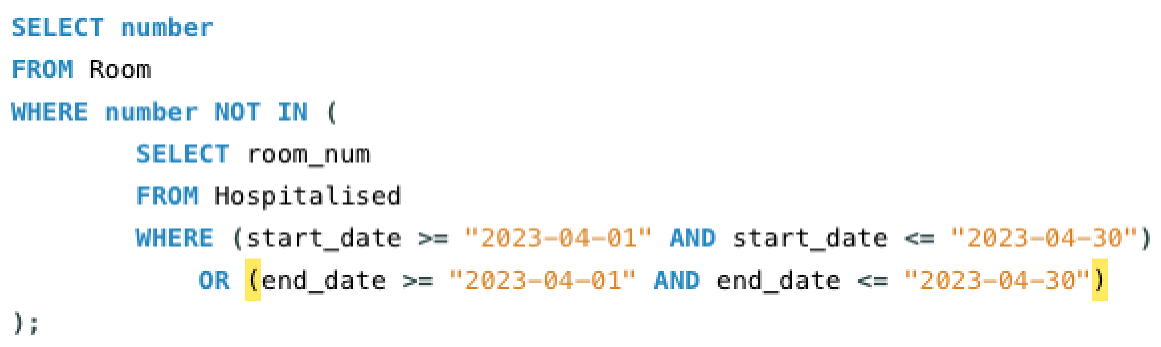
```
SELECT code, name, COUNT(dose) AS total_provisions
FROM Provide P
    RIGHT JOIN Medication M ON P.med_code = M.code
GROUP BY code
ORDER BY total_provisions DESC;
```



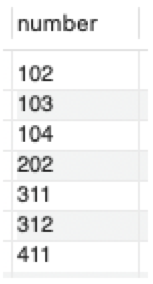
code	name	total_provisio...
35002	Morphine	2
35003	Adderall	2
35014	Lorazepam	1
35001	Ibuprofen	0
35004	Oxycontin	0
35005	Ativan	0
35006	Narcan	0
35007	Benadryl	0
35008	Meloxicam	0
35009	Flagyl	0
35010	Tylenol	0
35011	Benzonat...	0
35012	Wellbutrin	0
35013	Lyrica	0
35015	Cipro	0

3. Unused rooms in April 2023

The hospital is looking to repurpose some rooms. To gain insights, list out the room numbers that were not used by any patient in April 2023.



```
SELECT number
FROM Room
WHERE number NOT IN (
        SELECT room_num
        FROM Hospitalised
        WHERE (start_date >= "2023-04-01" AND start_date <= "2023-04-30")
            OR (end_date >= "2023-04-01" AND end_date <= "2023-04-30")
);
```

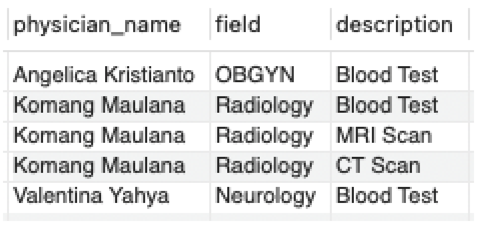


number
102
103
104
202
311
312
411

4. Proper instruction orders?

Some physicians might order an instruction that might not be appropriate to their field of expertise. To crack down on this, the hospital wants to find out what kind of orders different physicians have made. List out the physician names, fields, and instructions made for patients.

```
SELECT physician_name, field, description
FROM Doctor_order O
    JOIN Physician D ON D.id = O.phy_id
    JOIN Instruction I ON I.code = O.ins_code;
```

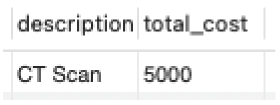


physician_name	field	description
Angelica Kristianto	OBGYN	Blood Test
Komang Maulana	Radiology	Blood Test
Komang Maulana	Radiology	MRI Scan
Komang Maulana	Radiology	CT Scan
Valentina Yahya	Neurology	Blood Test

5. Instruction with the most expense

To analyse the hospital’s spending on instructions, it would be helpful to identify the instruction that is costing the most. Find the name, number of orders and the total cost of the instruction that costs the most after orders are made. Cost is the fee of the instruction multiplied by the amount of times it has been ordered.

```
SELECT description, SUM(fee) AS total_cost
FROM Doctor_Order O
    JOIN Instruction I ON I.code = O.ins_code
GROUP BY code
ORDER BY total_cost DESC
LIMIT 1;
```

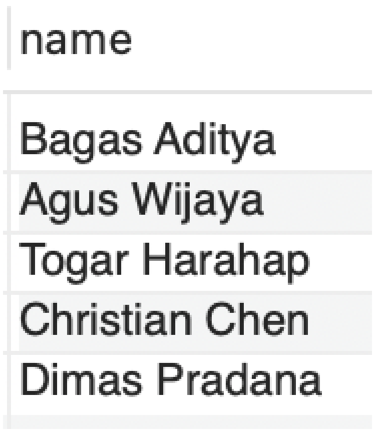


description	total_cost
CT Scan	5000

6. Patients with unpaid invoices

As certain emergencies arise in certain situations when a patient arrives at the hospital, like a patient that could've been shot with a gun, the hospitals need to operate on the patient first and then worry about the payment, that is why we need to keep track of who has not paid their invoices yet. This query will find the patient names who have not paid their bills issued by the hospital.

```
SELECT Patient.name
FROM Patient
    JOIN Invoice on Invoice.pat_id = Patient.id
WHERE Invoice.invoice_status = "Not Paid";
```

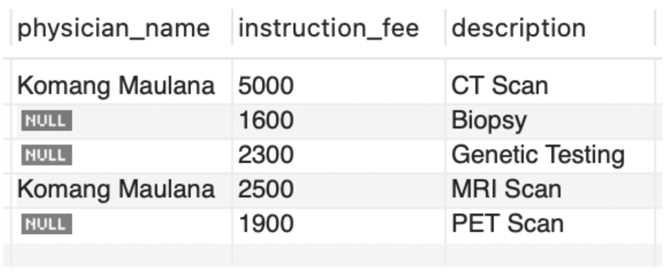


name
Bagas Aditya
Agus Wijaya
Togar Harahap
Christian Chen
Dimas Pradana

7. Instructions assigned physicians

Sometimes, certain instructions are not assigned to a particular physician. The hospital must keep track of that as well and make sure that each of the instructions is assigned to one physician because as instructions are costly, we need to make sure that someone is processing it and not just leaving it out.
The query should find out which physician is in charge of a certain instruction (which is more than $300) , the cost of that instruction and its description.

```
SELECT physician_name, I.fee AS instruction_fee, description
FROM Instruction I
    LEFT JOIN Doctor_order O ON I.code = O.ins_code
    LEFT JOIN Physician D ON D.id = O.phy_id
WHERE I.fee > 300;
```

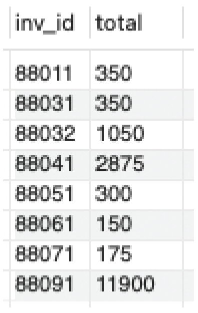


physician_name	instruction_fee	description
Komang Maulana	5000	CT Scan
NULL	1600	Biopsy
NULL	2300	Genetic Testing
Komang Maulana	2500	MRI Scan
NULL	1900	PET Scan

8. Omitting cheap instructions

Insurance companies don't want to pay for instructions under 100 dollars. List out all invoices along with their total price, but exclude instruction costs under 100 dollars.

```
SELECT inv_id, SUM(amount) AS total
FROM Invoice I
    JOIN Payable P ON P.inv_id = I.id
WHERE NOT (payable_type = "Instructions Total"
        AND amount < 100)
GROUP BY I.id;
```

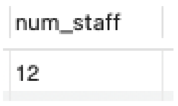


inv_id	total
88011	350
88031	350
88032	1050
88041	2875
88051	300
88061	150
88071	175
88091	11900

9. Number of staff

The hospital obviously doesn’t want to be understaffed. To make sure, count the total number of physicians and nurses.

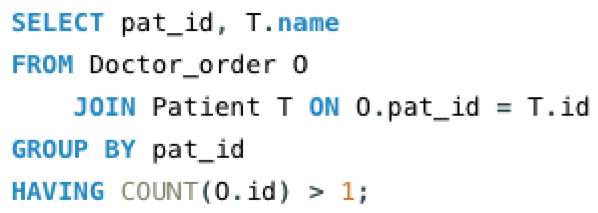
```
SELECT SUM(total) AS num_staff FROM
(SELECT COUNT(*) AS total FROM Physician
-- Ref: https://www.w3schools.com/sql/sql_union.asp
UNION ALL
SELECT COUNT(*) AS total FROM Nurse) AS T;
```



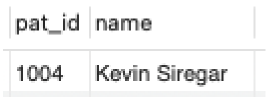
num_staff
12

10. Patients with multiple orders

Some patients might be at a higher priority than others and that we can assign more nurses to them. The hospital prioritises patients who have more than 1 order made to them by physicians. List out the IDs and names of patients with more than one order.



```sql
SELECT pat_id, T.name
FROM Doctor_order O
    JOIN Patient T ON O.pat_id = T.id
GROUP BY pat_id
HAVING COUNT(O.id) > 1;
```

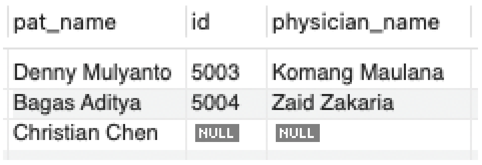


pat_id	name
1004	Kevin Siregar

11. Patients with health records monitored?

The hospital wants to make sure that physicians are aware of patients' prior medical records. That's why they want to ensure that there's at least one physician monitoring a patient with health records. Find the patient name and the monitoring physician's name and id. If no physician is monitoring, include null.

```sql
SELECT T.name AS pat_name, D.id, D.physician_name
FROM Patient T
    LEFT JOIN Monitors M ON T.id = M.pat_id
    LEFT JOIN Physician D ON M.phy_id = D.id
WHERE T.id IN (
    SELECT pat_id FROM Health_record
);
```

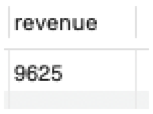


pat_name	id	physician_name
Denny Mulyanto	5003	Komang Maulana
Bagas Aditya	5004	Zaid Zakaria
Christian Chen	NULL	NULL

12. Total revenue from rooms

For administration analysis, the hospital would like to know the total revenue that will be collected just from rooms in 2023.

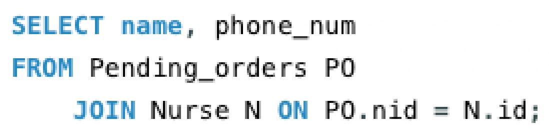
```sql
SELECT SUM(amount) AS revenue
FROM Payable
WHERE payable_type = "Rooms Total"
    AND payable_date >= "2023-01-01"
    AND payable_date <= "2023-12-31";
```



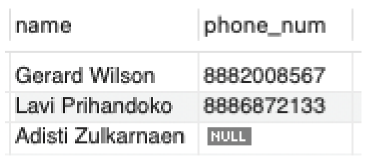
revenue
9625

13. Contact missing nurses

Hypothetical situation: an order is pending to be done on a patient, but the nurses can’t be found. For all nurses assigned to orders that haven’t been executed, find their name and contact information (phone number). This query takes advantage of one of the database views, so it’s heavily simplified.



```
SELECT name, phone_num
FROM Pending_orders PO
    JOIN Nurse N ON PO.nid = N.id;
```

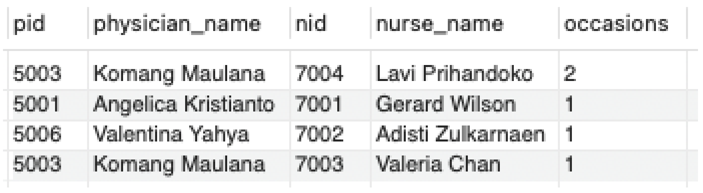


name	phone_num
Gerard Wilson	8882008567
Lavi Prihandoko	8886872133
Adisti Zulkarnaen	NULL

14. Detecting favouritism of nurses

The idea for orders is that the physician who made the order shouldn’t be able to hand pick certain nurses who can execute them. This avoids favouritism and encourages fair treatment between nurses. The hospital would like to find out if there are any physicians violating this practice. For every order with an assigned nurse, count the number of every physician-nurse pair. An even distribution of the numbers would mean a lack of favouritism.

```sql
SELECT D.id AS pid, D.physician_name,
    N.id AS nid, N.name AS nurse_name,
    COUNT(ord_id) AS occasions
FROM Executes E
    JOIN Doctor_order O ON E.ord_id = O.id
    JOIN Nurse N ON E.nur_id = N.id
    JOIN Physician D ON O.phy_id = D.id
GROUP BY D.id, N.id
ORDER BY occasions DESC;
```



pid	physician_name	nid	nurse_name	occasions
5003	Komang Maulana	7004	Lavi Prihandoko	2
5001	Angelica Kristianto	7001	Gerard Wilson	1
5006	Valentina Yahya	7002	Adisti Zulkarnaen	1
5003	Komang Maulana	7003	Valeria Chan	1

15. Total revenue already paid

Count the total revenue already paid. That is, sum up all the invoices that have been paid in full.

```sql
SELECT SUM(amount) AS revenue
FROM Invoice I
    JOIN Payable P ON I.id = P.inv_id
WHERE invoice_status = "Paid";
```

revenue
3800

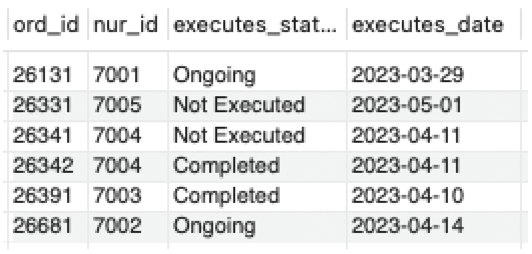
Transactions and Descriptions.

For the sake of organisation, each of our 2 transactions will be put on a separate page.

1. Making a new order and assigning a nurse

Suppose a physician wants to make an order for a patient and for the sake of time, would like to assign a nurse immediately. This transaction would prevent maybe some other program that assigns a nurse automatically at a different time.

```sql
SET TRANSACTION READ WRITE;
-- a. Doctor makes a new order
INSERT INTO Doctor_order
(id, phy_id, pat_id, ins_code, order_date)
VALUES
(26331, 5003, 1003,
(SELECT code FROM Instruction
WHERE description = "Adderall"),
CAST(NOW() AS DATE));
-- b. Assign a nurse to handle it
INSERT INTO Executes
(ord_id, nur_id, executes_status, executes_date)
VALUES
(26331, 7005, "Not Executed",
CAST(NOW() AS DATE));
-- c. Verify changes
SELECT *
FROM Executes;
-- Finish
COMMIT;
```

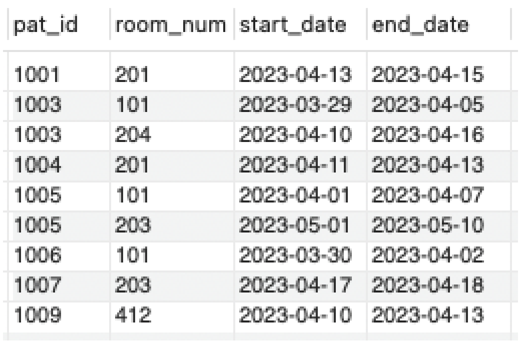


ord_id	nur_id	executes_stat...	executes_date
26131	7001	Ongoing	2023-03-29
26331	7005	Not Executed	2023-05-01
26341	7004	Not Executed	2023-04-11
26342	7004	Completed	2023-04-11
26391	7003	Completed	2023-04-10
26681	7002	Ongoing	2023-04-14

2. Booking a new room while checking for empty ones

When booking for a new room, there must exist at least one vacant bed in a room. This transaction aims to prevent the case where 2 bookings are made at the same time when there is only one vacant spot.

```
SET TRANSACTION READ WRITE;
-- a. Check for empty rooms
SELECT * FROM Available_rooms;
-- b. Book a new room for a patient
INSERT INTO Hospitalised
(pat_id, room_num, start_date, end_date)
VALUES
(1005, 203, CAST(NOW() AS DATE), "2023-05-10");
-- c. Verify booking exists
SELECT * FROM Hospitalised;
-- d. Verify payable exists
SELECT * FROM Payable;
-- Finish
COMMIT;
```



pat_id	room_num	start_date	end_date
1001	201	2023-04-13	2023-04-15
1003	101	2023-03-29	2023-04-05
1003	204	2023-04-10	2023-04-16
1004	201	2023-04-11	2023-04-13
1005	101	2023-04-01	2023-04-07
1005	203	2023-05-01	2023-05-10
1006	101	2023-03-30	2023-04-02
1007	203	2023-04-17	2023-04-18
1009	412	2023-04-10	2023-04-13

References

Missing semicolon error:
https://stackoverflow.com/questions/39307880/mysql-trigger-syntax-missing-semicolon

SQL Coalesce to treat null values as 0:
https://www.w3schools.com/sql/func_sqlserver_coalesce.asp

SQL Insert On Duplicate Key Update:
https://dev.mysql.com/doc/refman/8.0/en/insert-on-duplicate.html

Union include duplicates:
https://www.w3schools.com/sql/sql_union.asp