Hospital Database Report

Samuel Baynes

ITCS 3160-054

May 2, 2023

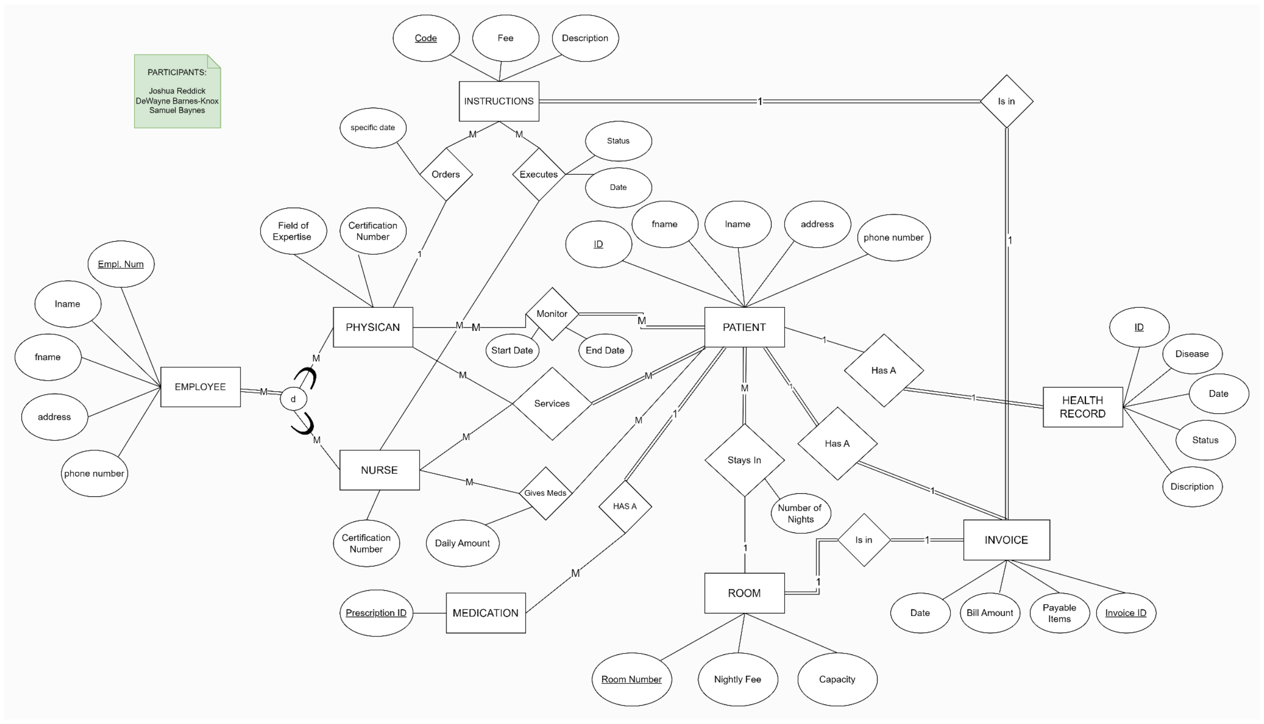
The following report details the design of a hospital database we created, based on an initial set of requirements and assumptions. The report details additional assumptions we made, provides an (E)ERD, relations and keys, views and descriptions, triggers, queries and results, and describes the planning going into each part of the process.

1- **Additional Assumptions:**

- Each patient receives one invoice for their visit consisting of their room and instructions.
- Patients stay in one room for the entirety of their stay, sometimes multiple patients share a room.
- Each medication has an unique ID
- While patients receive their medication from a pharmacy, the pharmacy is not part of the hospital.
- Each invoice has a unique id.
- Every Nurse has many patients and Patients have multiple Nurses.
- Hospital gives services (could be multiple services) to patients.
- Not every physician is monitoring a patient at all times, but all patients have a physician monitoring them.
- Nurses may complete multiple instructions for a physician because the physician is capable of giving multiple instructions at a time.
- Instructions are categorized as a single set that appears on the invoice. Every set of instructions appears on an invoice and every invoice has a patient's instructions included.

2- **(E)ERD**

The hospital (E)ERD was updated and improved based on the feedback from Project 1 in a number of ways. Initially (on Project 1), we included the pharmacy entity and did not include relationships between Instructions and Invoice, and Rooms and Invoice. The updated (E)ERD, found below, does not include the pharmacy entity because the patient has medication, but the medication is provided from outside the scope of the hospital. Also, the relationships to the Invoice were added. Additionally, we added a few minor assumptions (found in the previous section) to assert the totality of the Rooms and Instructions relationship to the Invoice and the 1 to 1 relationships found on the updated (E)ERD. These relations were applied to the schema for Project 2 and are also applied in our schema and tables for Project 3.



3- **Relations and keys**

The following relationships, entities, and their respective primary and foreign keys reflect the schema of the hospital database.

RELATIONSHIPS:
monitors(physician_num, patient_id, startDate, endDate)
primary key {physician_num, patient_id}
foreign key{physician_num references physician(physician_num), patient_id references patient(patient_id)}

give_meds(patient_id, nurse_id, daily_amount)
primary key {patient_id, nurse_id}
foreign key {patient_id references patient(patient_id), nurse_num references (nurse_num)}

ENTITIES
physician(physician_num, field_of_expertise, certification_number, fname , lname, address, phone_number)
primary key: {physician_num}

nurse(nurse_num, certification_number, fname, lname, address, phone_number)
primary key: {nurse_num}

medication(prescription_id, patient_id)
primary key: {prescription_id}
foreign key: {patient_id references patient(patient_id)}

patient(patient_id, fname, lname, address, phone_number, room_num, num_nights)

primary key: {patient_id}
foreign key: {room_num references room(room_num)}

room(room_num, nightly_fee, capacity)
primary key: {room_num}

invoice(invoice_id, invoice_date, bill_amount, payable_items, patient_id, room_num, code_id)
primary key: {invoice_id}
foreign key: {patient_id references patient(patient_id), room_num references room(room_num), code_id references instructions(code_id)}

healthRecord(patient_id, disease, record_date, record_status, record_description)
primary key: {patient_id}
foreign key: {patient_id references patient(patient_id)}

instructions(code_id, fee, description, status, execution_date, order_date, physician_num, nurse_num)
primary key: {code_id}
foreign key: {nurse_num references (nurse_num), physician_num references (physician_num)}

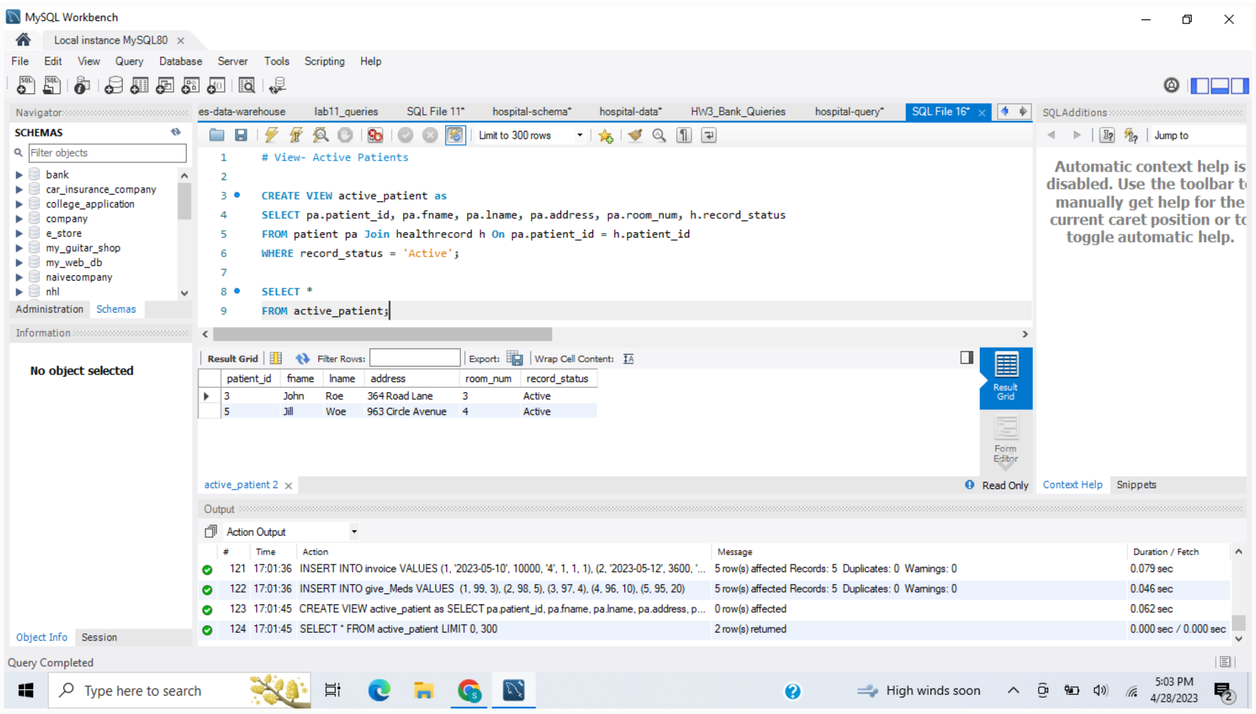
4- **Views and description**

View 1- active_patient : Active patients, their address, and room number

The following query created a view called “active_patient” that is used to display active patients at the hospital’s patient ID, first and last name, address, room number, and their status at the hospital. A table like this may be useful for administrative or reception purposes because it hides private medical information, while displaying who is currently staying at the hospital and where they may be found in the hospital.

```
CREATE VIEW active_patient as
SELECT pa.patient_id, pa.fname, pa.lname, pa.address, pa.room_num, h.record_status
FROM patient pa Join healthrecord h On pa.patient_id = h.patient_id
WHERE record_status = 'Active';
```

The next screenshot displays the *active_patient* view.



```
# View- Active Patients

CREATE VIEW active_patient as
SELECT pa.patient_id, pa.fname, pa.lname, pa.address, pa.room_num, h.record_status
FROM patient pa Join healthrecord h On pa.patient_id = h.patient_id
WHERE record_status = 'Active';

SELECT *
FROM active_patient;
```

patient_id	fname	lname	address	room_num	record_status
3	John	Roe	364 Road Lane	3	Active
5	Jill	Woe	963 Circle Avenue	4	Active

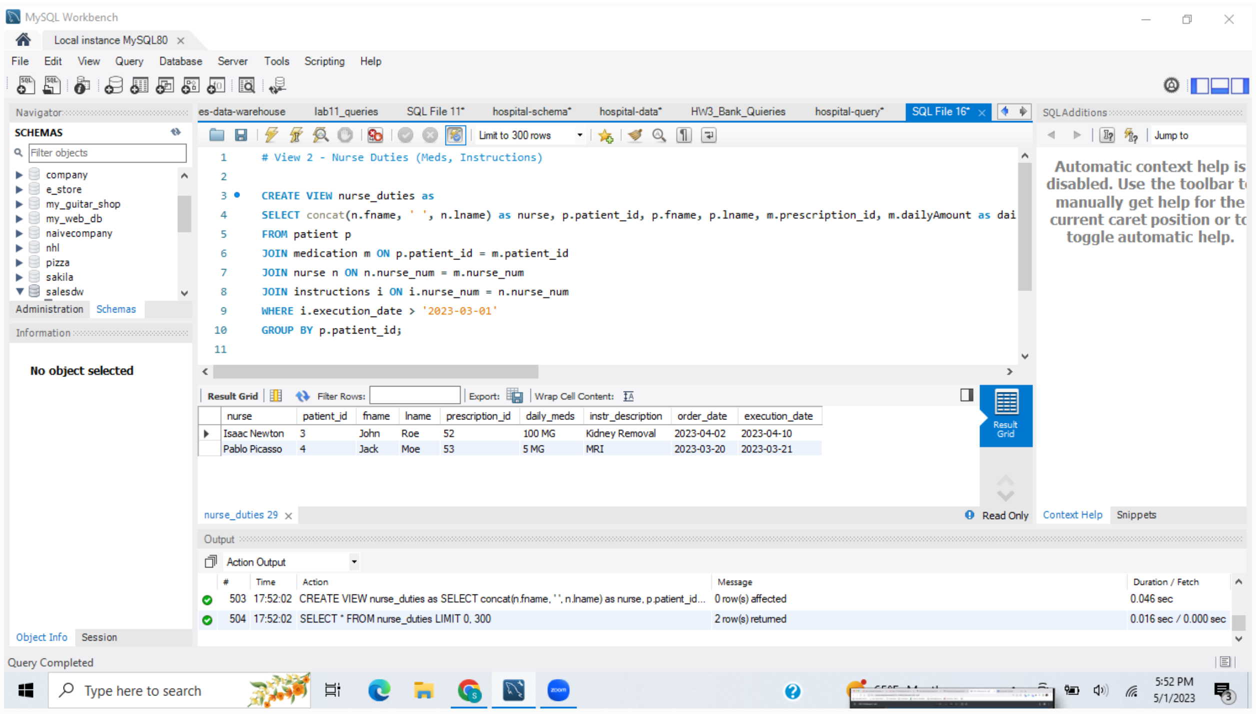
```
#    Time      Action                                                                          Message                                                        Duration / Fetch
121  17:01:36  INSERT INTO invoice VALUES (1, '2023-05-10', 10000, '4', 1, 1, 1), (2, '2023-05-12', 3600, '...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.079 sec
122  17:01:36  INSERT INTO give_Meds VALUES  (1, 99, 3), (2, 98, 5), (3, 97, 4), (4, 96, 10), (5, 95, 20)   5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.046 sec
123  17:01:45  CREATE VIEW active_patient as SELECT pa.patient_id, pa.fname, pa.lname, pa.address, p...      0 row(s) affected                                              0.062 sec
124  17:01:45  SELECT * FROM active_patient LIMIT 0, 300                                                       2 row(s) returned                                              0.000 sec / 0.000 sec
```

View 2 - nurse_duties : Diplays the nurse assigned to each patient and instructions for the nurse for dates after March 1, 2023.

The "nurse_duties" view is used to display the nurse assigned to each patient and the duties required of the nurse including the medication needing to be administered and an instruction description for instructions given after March 1, 2023. This would be a useful table for nurses to use throughout their daily assignments. It joins the patient, medication, nurse, and instructions tables, and groups by patients, but narrows the information to only necessary requirements of a nurse's job in relation to taking care of a patient. Limiting results to only dates after March 1st, using a "Where" statement allows the nurse to see current patients (assuming it's March), although the date may be modified in a real-life scenario. The view is written as follows:

```
CREATE VIEW nurse_duties as
SELECT concat(n.fname, ' ', n.lname) as nurse, p.patient_id, p.fname, p.lname,
m.prescription_id, m.dailyAmount as daily_meds, i.instr_description, i.order_date,
i.execution_date
FROM patient p
JOIN medication m ON p.patient_id = m.patient_id
JOIN nurse n ON n.nurse_num = m.nurse_num
JOIN instructions i ON i.nurse_num = n.nurse_num
WHERE i.execution_date > '2023-03-01'
GROUP BY p.patient_id;
```

The screenshot found below shows the results of the *nurse_duties* view run on our hospital database.



```
# View 2 - Nurse Duties (Meds, Instructions)

CREATE VIEW nurse_duties as
SELECT concat(n.fname, ' ', n.lname) as nurse, p.patient_id, p.fname, p.lname, m.prescription_id, m.dailyAmount as dai
FROM patient p
JOIN medication m ON p.patient_id = m.patient_id
JOIN nurse n ON n.nurse_num = m.nurse_num
JOIN instructions i ON i.nurse_num = n.nurse_num
WHERE i.execution_date > '2023-03-01'
GROUP BY p.patient_id;
```

nurse	patient_id	fname	lname	prescription_id	daily_meds	instr_description	order_date	execution_date
Isaac Newton	3	John	Roe	52	100 MG	Kidney Removal	2023-04-02	2023-04-10
Pablo Picasso	4	Jack	Moe	53	5 MG	MRI	2023-03-20	2023-03-21

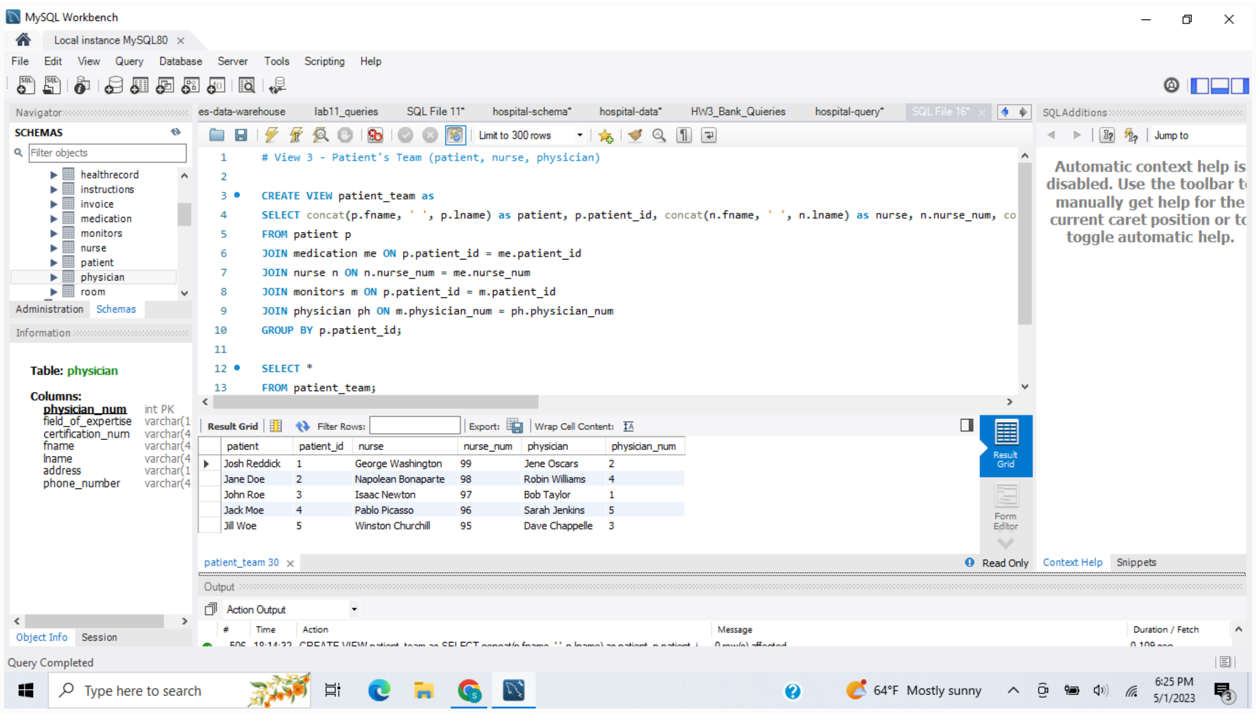
```
#    Time      Action                                                                       Message            Duration / Fetch
503  17:52:02  CREATE VIEW nurse_duties as SELECT concat(n.fname, ' ', n.lname) as nurse, p.patient_id...  0 row(s) affected  0.046 sec
504  17:52:02  SELECT * FROM nurse_duties LIMIT 0, 300                                       2 row(s) returned  0.016 sec / 0.000 sec
```

View 3 - patient_team : Displays the patient and their team (comprised of their Physician and Nurse)

The *patient_team* view is used to display the staff members responsible for taking care of each patient and their ID numbers. This table would be useful for keeping track of who's responsible for each patient. A "Concat" aggregation on the names helps identify and differentiate the names and their roles on the team. Also, this table would be useful performing additional queries associated with the relations because it eliminates an assortment of Join functions. The query is as follows:

```
CREATE VIEW patient_team as
SELECT concat(p.fname, ' ', p.lname) as patient, p.patient_id, concat(n.fname, ' ',
n.lname) as nurse, n.nurse_num, concat(ph.fname, ' ', ph.lname) as physician,
ph.physician_num
FROM patient p
JOIN medication me ON p.patient_id = me.patient_id
JOIN nurse n ON n.nurse_num = me.nurse_num
JOIN monitors m ON p.patient_id = m.patient_id
JOIN physician ph ON m.physician_num = ph.physician_num
GROUP BY p.patient_id;
```

The next screenshot shows the *patient_team* view when run on our database.



```sql
# View 3 - Patient's Team (patient, nurse, physician)

CREATE VIEW patient_team as
SELECT concat(p.fname, ' ', p.lname) as patient, p.patient_id, concat(n.fname, ' ', n.lname) as nurse, n.nurse_num, co
FROM patient p
JOIN medication me ON p.patient_id = me.patient_id
JOIN nurse n ON n.nurse_num = me.nurse_num
JOIN monitors m ON p.patient_id = m.patient_id
JOIN physician ph ON m.physician_num = ph.physician_num
GROUP BY p.patient_id;

SELECT *
FROM patient_team;
```

patient	patient_id	nurse	nurse_num	physician	physician_num
Josh Reddick	1	George Washington	99	Jene Oscars	2
Jane Doe	2	Napolean Bonaparte	98	Robin Williams	4
John Roe	3	Isaac Newton	97	Bob Taylor	1
Jack Moe	4	Pablo Picasso	96	Sarah Jenkins	5
Jill Woe	5	Winston Churchill	95	Dave Chappelle	3

Table: physician

Columns:

physician_num	int PK
field_of_expertise	varchar(1
certification_num	varchar(4
fname	varchar(4
lname	varchar(4
address	varchar(1
phone_number	varchar(4

5- <u>Triggers and description</u>

The query below is a trigger designed to limit every nurses Certification Number to six digits, when a new Certification Number is entered and it exceeds six digits, it sets the new Certification Number to 0 (‘000000’). This is a useful trigger because every nurse’s certification number should be of an unique value, but also of one uniform length.

```
mysql> delimiter //
CREATE TRIGGER certificationNumberLimit
BEFORE INSERT
ON nurse FOR EACH ROW
BEGIN
IF length(NEW.certification_num) > 6 or length(NEW.certification_num) < 6 THEN
UPDATE nurse set NEW.certification_num = 6
END IF;
END
mysql> delimiter ;
```

This next query is designed to limit the price of a room per night to 100 dollars. If the entered price of a room exceeds 100 dollars the price per night of a room is set to 100 dollars by default. The price of a room per night can be below 100 dollars. This trigger is useful to our database because it prevents payment discrepancies at the end of a patients visit and reduces the potential for error.

```
mysql> delimiter //
CREATE TRIGGER nightlyFeeMax
```

```
BEFORE INSERT
ON room FOR EACH ROW
BEGIN
IF NEW.nightly_fee > 100 THEN UPDATE room SET NEW.nightly_fee = 100
END IF;
END
mysql> delimiter ;
```

The last query is a trigger that limits the amount of medication a patient can take per day to no more than 999 miligrams. If the value entered for the daily amount cell exceeds 1000 MG (or has a length of 7 characters) then by default the value will be 250 MG. This trigger is useful to a hospital database because it can help keep track of and conserve the amount of medication being allocated to patients on a daily basis.

```
mysql> delimiter //
CREATE TRIGGER maximum_dosage
BEFORE INSERT
ON medication FOR EACH ROW
BEGIN
IF length(NEW.dailyAmount) > 7 THEN UPDATE medication SET NEW.dailyAmount =
'250 MG'
END IF;
END
mysql> delimiter ;
```

6- <u>**Queries, descriptions, and results.**</u>

The following 15 queries were designed to exemplify the operational capabilities of our hospital database. The queries are categorized into Join (*Query 1-3*), Aggregation (*Query 4-6*), Nested (*Query 7-9*), and Other various queries (*Query 10-15*). Each query includes a short description, the query, and a screenshot of the query being run on MYSQL. The queries found in the “Other Queries” section include many join, aggregation, and nested elements, but these elements are described in greater detail in the first three sections below. The tables in our hospital database only include five tuples per table, so the results of the queries are often limited in the number of results, but they’re scalable.

a. JOIN QUERIES

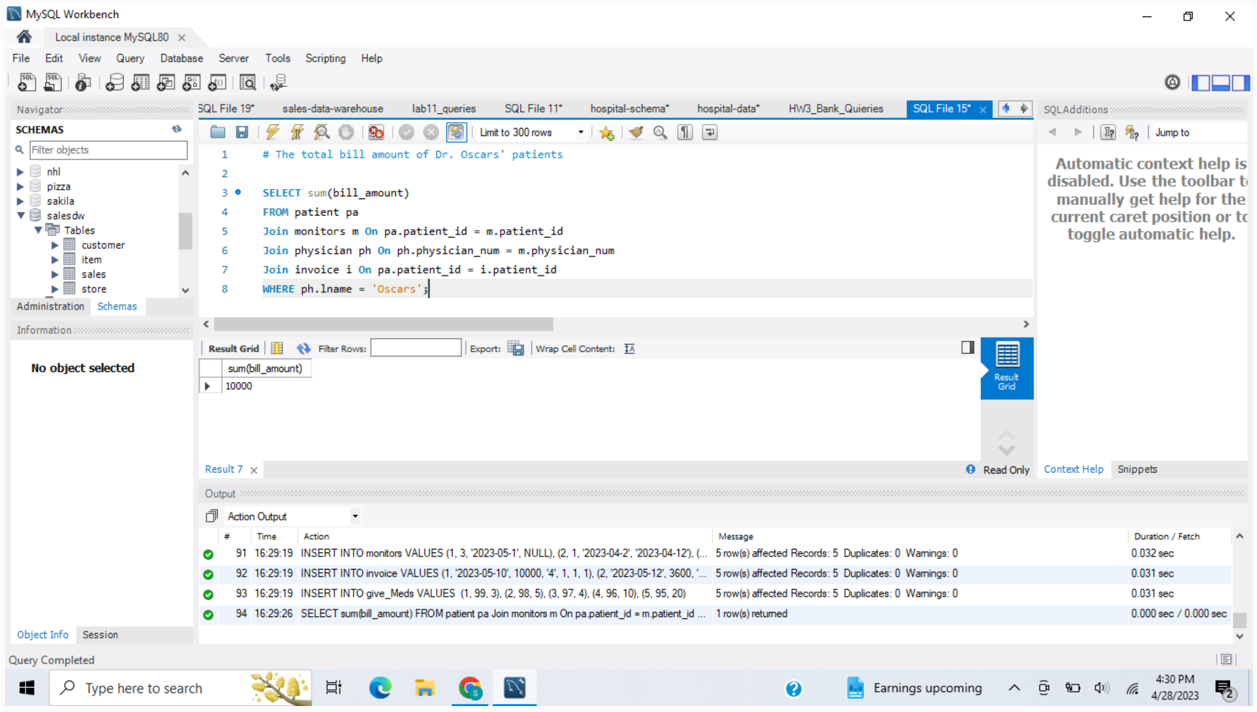
Query 1 - Total Bill amount of Dr. Oscar’s patients

The following query calculates and returns the total bill amount of patients monitored by Dr. Oscar. This query is completed by joining the patient, physician, and invoice entity tables, along with the monitors relationship tables based on the physician and patient id’s, then finding where the physician’s last name is Oscars.

```
SELECT sum(bill_amount)
```

FROM patient pa
Join monitors m On pa.patient_id = m.patient_id
Join physician ph On ph.physician_num = m.physician_num
Join invoice i On pa.patient_id = i.patient_id
WHERE ph.lname = 'Oscars';

The following screenshot shows the results of the above query:



```
# The total bill amount of Dr. Oscars' patients

SELECT sum(bill_amount)
FROM patient pa
Join monitors m On pa.patient_id = m.patient_id
Join physician ph On ph.physician_num = m.physician_num
Join invoice i On pa.patient_id = i.patient_id
WHERE ph.lname = 'Oscars';
```

```
sum(bill_amount)
10000
```

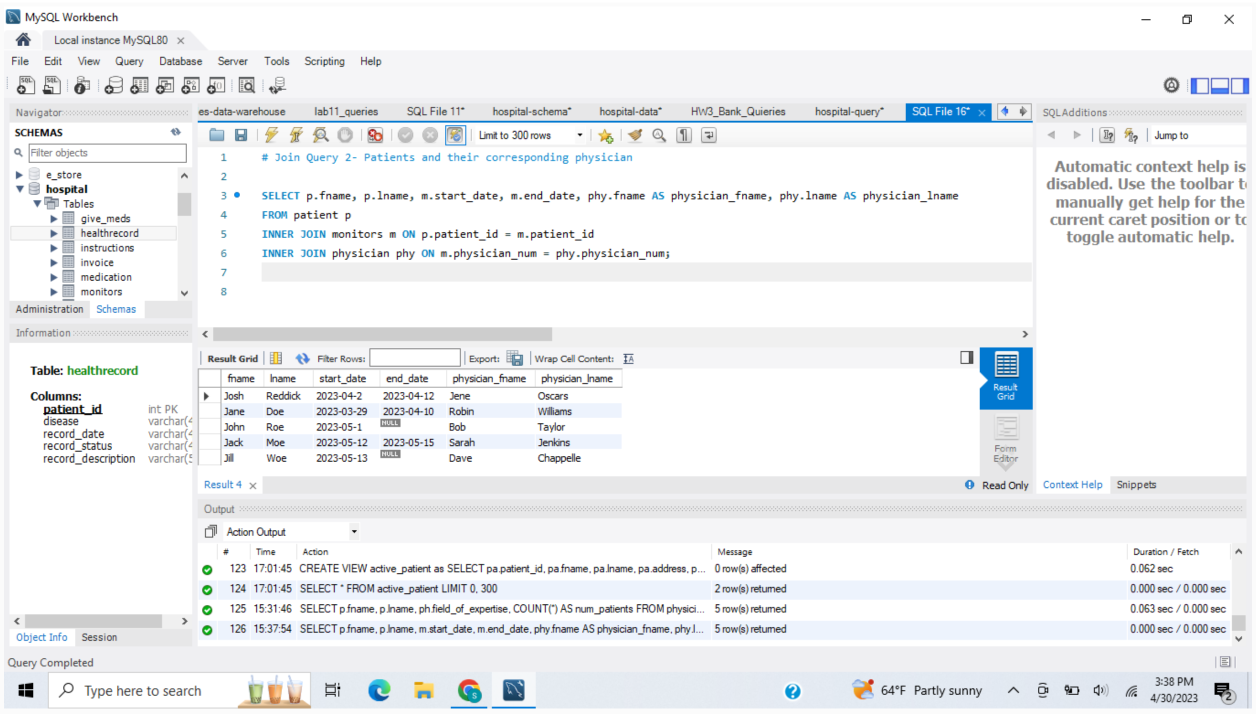
```
#   Time      Action                                                                              Message                                                      Duration / Fetch
91  16:29:19  INSERT INTO monitors VALUES (1, 3, '2023-05-1', NULL), (2, 1, '2023-04-2', '2023-04-12'), (...   5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0   0.032 sec
92  16:29:19  INSERT INTO invoice VALUES (1, '2023-05-10', 10000, '4', 1, 1, 1), (2, '2023-05-12', 3600, '...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0   0.031 sec
93  16:29:19  INSERT INTO give_Meds VALUES  (1, 99, 3), (2, 98, 5), (3, 97, 4), (4, 96, 10), (5, 95, 20)       5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0   0.031 sec
94  16:29:26  SELECT sum(bill_amount) FROM patient pa Join monitors m On pa.patient_id = m.patient_id ...     1 row(s) returned                                            0.000 sec / 0.000 sec
```

Query 2- Patients and their corresponding physician

This query shows the patient's first name, last name, start date, end date, and first name/last name of the physician who monitors the patients.

SELECT p.fname, p.lname, m.start_date, m.end_date, phy.fname AS physician_fname, phy.lname AS physician_lname
FROM patient p
INNER JOIN monitors m ON p.patient_id = m.patient_id
INNER JOIN physician phy ON m.physician_num = phy.physician_num;

The following screenshot displays the results of the above query:



```
# Join Query 2- Patients and their corresponding physician

SELECT p.fname, p.lname, m.start_date, m.end_date, phy.fname AS physician_fname, phy.lname AS physician_lname
FROM patient p
INNER JOIN monitors m ON p.patient_id = m.patient_id
INNER JOIN physician phy ON m.physician_num = phy.physician_num;
```

fname	lname	start_date	end_date	physician_fname	physician_lname
Josh	Reddick	2023-04-2	2023-04-12	Jene	Oscars
Jane	Doe	2023-03-29	2023-04-10	Robin	Williams
John	Roe	2023-05-1	NULL	Bob	Taylor
Jack	Moe	2023-05-12	2023-05-15	Sarah	Jenkins
Jill	Woe	2023-05-13	NULL	Dave	Chappelle

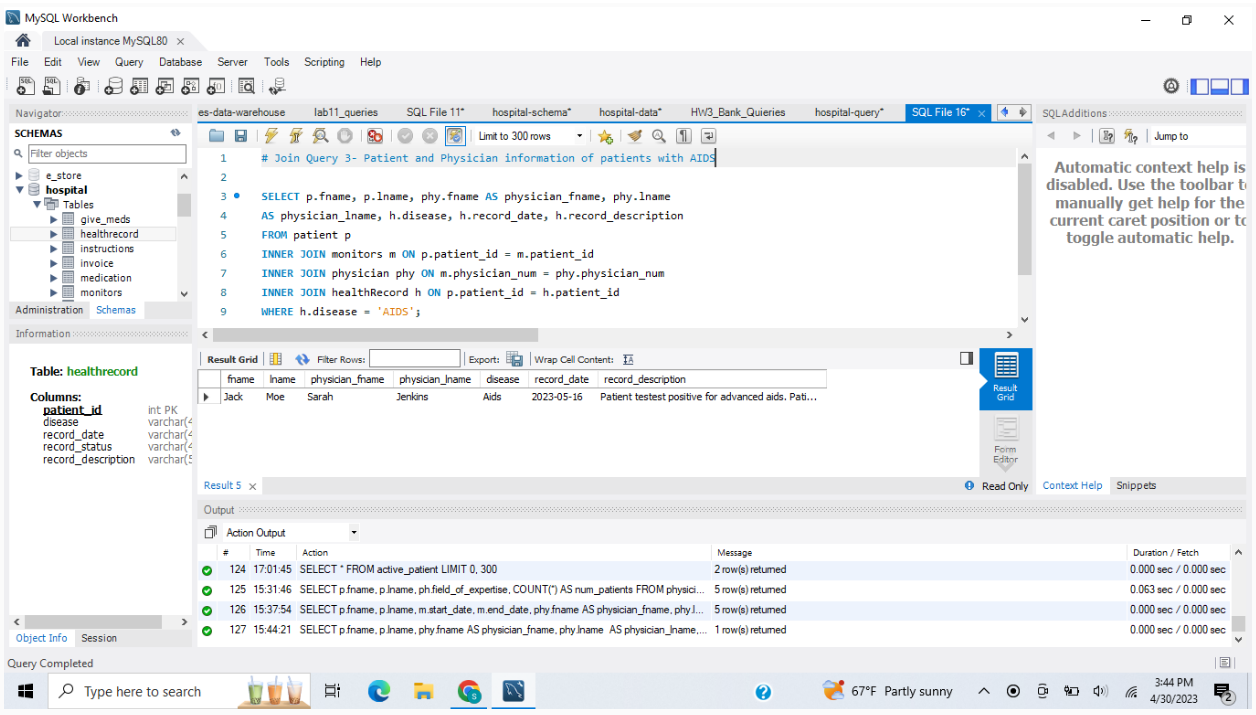
```
#    Time      Action                                                                              Message             Duration / Fetch
123  17:01:45  CREATE VIEW active_patient as SELECT pa.patient_id, pa.fname, pa.lname, pa.address, p...   0 row(s) affected   0.062 sec
124  17:01:45  SELECT * FROM active_patient LIMIT 0, 300                                           2 row(s) returned   0.000 sec / 0.000 sec
125  15:31:46  SELECT p.fname, p.lname, ph.field_of_expertise, COUNT(*) AS num_patients FROM physici...   5 row(s) returned   0.063 sec / 0.000 sec
126  15:37:54  SELECT p.fname, p.lname, m.start_date, m.end_date, phy.fname AS physician_fname, phy.l...   5 row(s) returned   0.000 sec / 0.000 sec
```

Query 3 - Patient and Physician information of patients with AIDS

This query shows the first name, last name, physician first name/last name, disease, record date, and record description of patients who have a health record with the disease "AIDS" and the physician responsible for monitoring them.

```
SELECT p.fname, p.lname, phy.fname AS physician_fname, phy.lname
AS physician_lname, h.disease, h.record_date, h.record_description
FROM patient p
INNER JOIN monitors m ON p.patient_id = m.patient_id
INNER JOIN physician phy ON m.physician_num = phy.physician_num
INNER JOIN healthRecord h ON p.patient_id = h.patient_id
WHERE h.disease = 'AIDS';
```

The next screenshot displays the results of *Query 3.*



```
# Join Query 3- Patient and Physician information of patients with AIDS

SELECT p.fname, p.lname, phy.fname AS physician_fname, phy.lname
AS physician_lname, h.disease, h.record_date, h.record_description
FROM patient p
INNER JOIN monitors m ON p.patient_id = m.patient_id
INNER JOIN physician phy ON m.physician_num = phy.physician_num
INNER JOIN healthRecord h ON p.patient_id = h.patient_id
WHERE h.disease = 'AIDS';
```

fname	lname	physician_fname	physician_lname	disease	record_date	record_description
Jack	Moe	Sarah	Jenkins	Aids	2023-05-16	Patient testest positive for advanced aids. Pati...

```
#    Time      Action                                                                              Message              Duration / Fetch
124  17:01:45  SELECT * FROM active_patient LIMIT 0, 300                                           2 row(s) returned    0.000 sec / 0.000 sec
125  15:31:46  SELECT p.fname, p.lname, ph.field_of_expertise, COUNT(*) AS num_patients FROM physici...   5 row(s) returned    0.063 sec / 0.000 sec
126  15:37:54  SELECT p.fname, p.lname, m.start_date, m.end_date, phy.fname AS physician_fname, phy.l...   5 row(s) returned    0.000 sec / 0.000 sec
127  15:44:21  SELECT p.fname, p.lname, phy.fname AS physician_fname, phy.lname  AS physician_lname,...    1 row(s) returned    0.000 sec / 0.000 sec
```

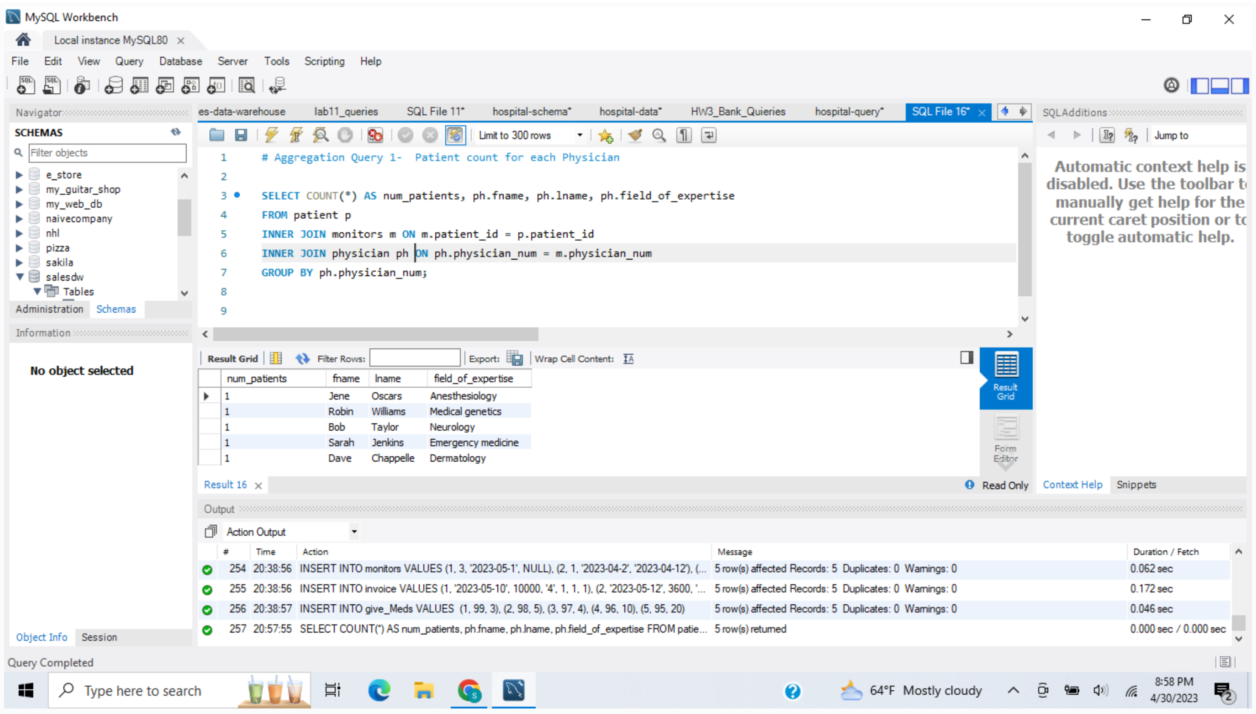
b. AGGREGATION QUERIES

Query 4 - Patient count for each Physician

This query shows the number of patients monitored by physicians, the physician's first name, last name, and field of expertise by joining the patient, physician, and monitors tables, grouping the physician_num and performing an aggregation function by counting the number of patients for each physician.

```
SELECT COUNT(*) AS num_patients, ph.fname, ph.lname, ph.field_of_expertise
FROM patient p
INNER JOIN monitors m ON m.patient_id = p.patient_id
INNER JOIN physician ph ON ph.physician_num = m.physician_num
GROUP BY ph.physician_num;
```

The following screenshot shows *Query 4* with our current values in the database. Each doctor in our database has only one patient assigned to them. So, a second screenshot is provided where we assigned two current patients to Dr. Taylor to confirm the functionality of *Query 4.*



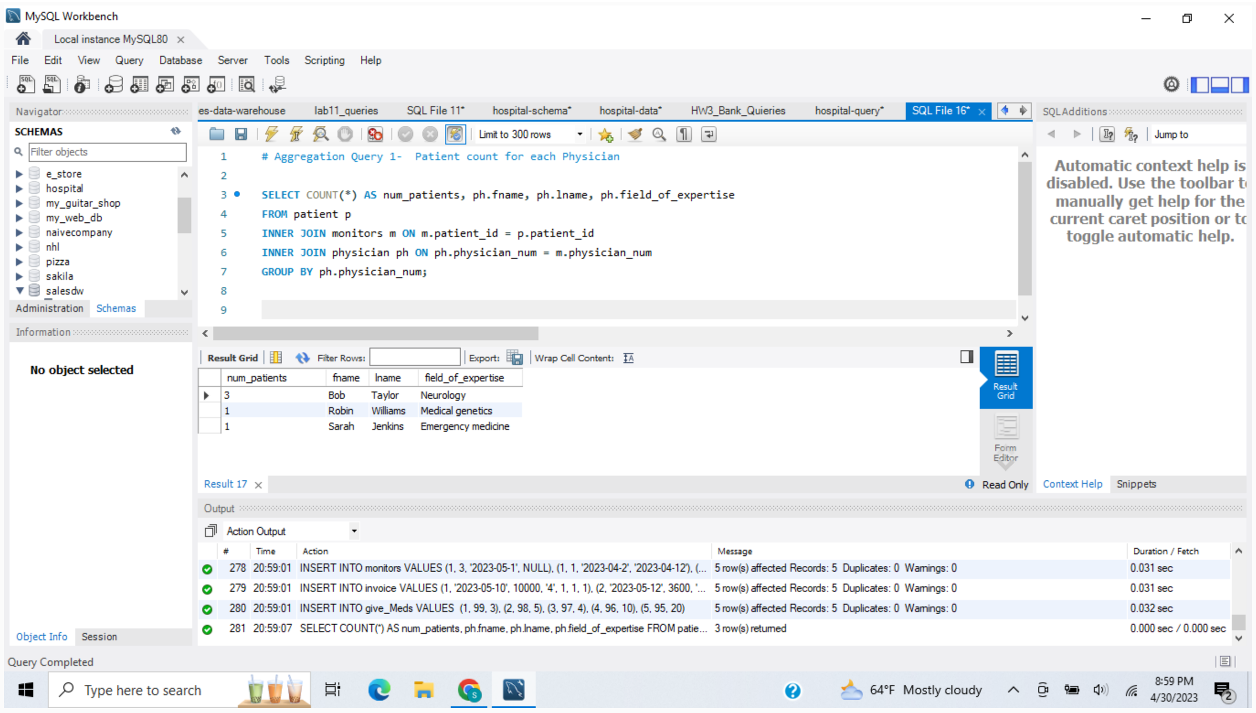
```
# Aggregation Query 1-  Patient count for each Physician

SELECT COUNT(*) AS num_patients, ph.fname, ph.lname, ph.field_of_expertise
FROM patient p
INNER JOIN monitors m ON m.patient_id = p.patient_id
INNER JOIN physician ph ON ph.physician_num = m.physician_num
GROUP BY ph.physician_num;
```

num_patients	fname	lname	field_of_expertise
1	Jene	Oscars	Anesthesiology
1	Robin	Williams	Medical genetics
1	Bob	Taylor	Neurology
1	Sarah	Jenkins	Emergency medicine
1	Dave	Chappelle	Dermatology

```
#    Time      Action                                                                          Message                                                         Duration / Fetch
254  20:38:56  INSERT INTO monitors VALUES (1, 3, '2023-05-1', NULL), (2, 1, '2023-04-2', '2023-04-12'), (...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.062 sec
255  20:38:56  INSERT INTO invoice VALUES (1, '2023-05-10', 10000, '4', 1, 1, 1), (2, '2023-05-12', 3600, '...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.172 sec
256  20:38:57  INSERT INTO give_Meds VALUES  (1, 99, 3), (2, 98, 5), (3, 97, 4), (4, 96, 10), (5, 95, 20)     5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.046 sec
257  20:57:55  SELECT COUNT(*) AS num_patients, ph.fname, ph.lname, ph.field_of_expertise FROM patie...     5 row(s) returned                                         0.000 sec / 0.000 sec
```

The screenshot below shows where we temporarily assigned two additional patients to Dr. Taylor in order to confirm *Query 4* does indeed work. As you can see, the query now counts and displays three patients are assigned to Dr. Taylor.



```
# Aggregation Query 1-  Patient count for each Physician

SELECT COUNT(*) AS num_patients, ph.fname, ph.lname, ph.field_of_expertise
FROM patient p
INNER JOIN monitors m ON m.patient_id = p.patient_id
INNER JOIN physician ph ON ph.physician_num = m.physician_num
GROUP BY ph.physician_num;
```

num_patients	fname	lname	field_of_expertise
3	Bob	Taylor	Neurology
1	Robin	Williams	Medical genetics
1	Sarah	Jenkins	Emergency medicine

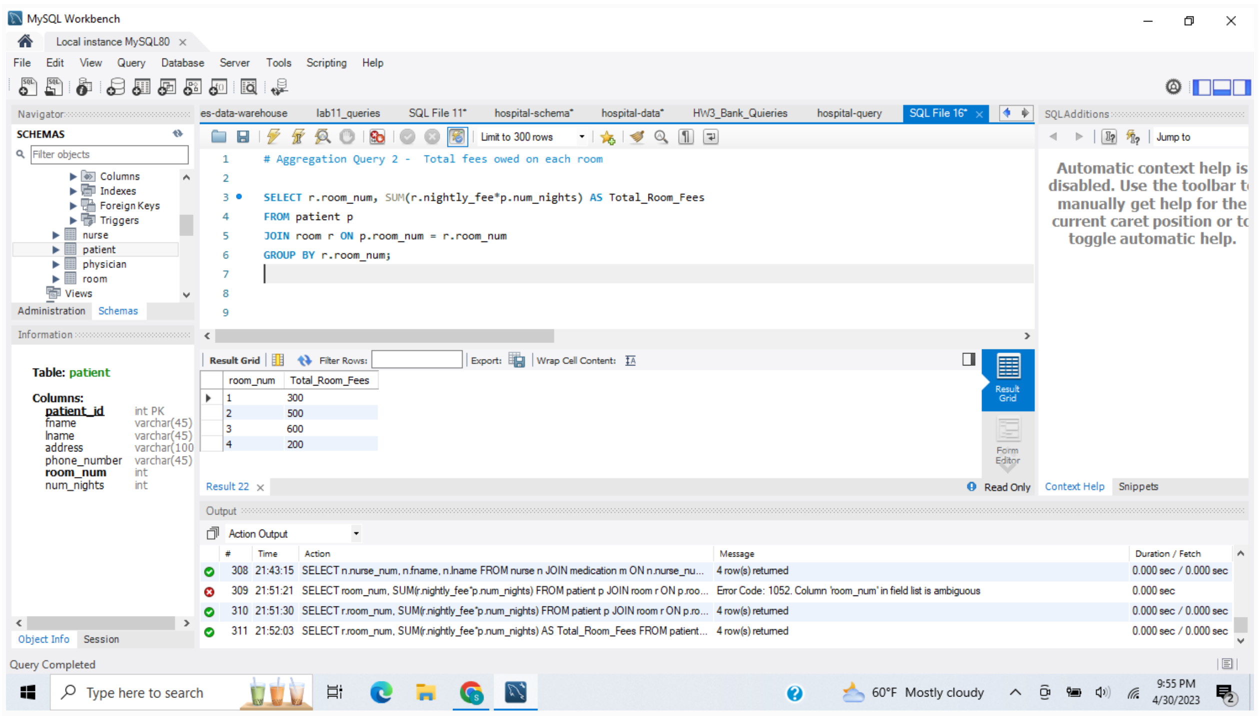
```
#    Time      Action                                                                          Message                                                         Duration / Fetch
278  20:59:01  INSERT INTO monitors VALUES (1, 3, '2023-05-1', NULL), (1, 1, '2023-04-2', '2023-04-12'), (...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.031 sec
279  20:59:01  INSERT INTO invoice VALUES (1, '2023-05-10', 10000, '4', 1, 1, 1), (2, '2023-05-12', 3600, '...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.031 sec
280  20:59:01  INSERT INTO give_Meds VALUES  (1, 99, 3), (2, 98, 5), (3, 97, 4), (4, 96, 10), (5, 95, 20)     5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.032 sec
281  20:59:07  SELECT COUNT(*) AS num_patients, ph.fname, ph.lname, ph.field_of_expertise FROM patie...     3 row(s) returned                                         0.000 sec / 0.000 sec
```

Query 5- Total fees owed on each room

The following query calculates the Total Room Fees for each room by joining the patient and rooms tables, grouping by the room number, then multiplying the nightly fee by the number of nights patients stay in a room. When we “Group by” the room number and perform a “Sum” aggregation, the total of all patients room costs will result in the Total Room Fee.

```
SELECT r.room_num, SUM(r.nightly_fee*p.num_nights) AS Total_Room_Fees
FROM patient p
JOIN room r ON p.room_num = r.room_num
GROUP BY r.room_num;
```

The following screenshot shows the results of *Query 5.* Although there are five rooms in the hospital database, only four were used because two patients shared room number 4 and no one stayed in room number 5.



```
# Aggregation Query 2 -  Total fees owed on each room

SELECT r.room_num, SUM(r.nightly_fee*p.num_nights) AS Total_Room_Fees
FROM patient p
JOIN room r ON p.room_num = r.room_num
GROUP BY r.room_num;
```

room_num	Total_Room_Fees
1	300
2	500
3	600
4	200

Table: patient

Columns:

patient_id	int PK
fname	varchar(45)
lname	varchar(45)
address	varchar(100
phone_number	varchar(45)
room_num	int
num_nights	int

```
#    Time      Action                                                                        Message                                                         Duration / Fetch
308  21:43:15  SELECT n.nurse_num, n.fname, n.lname FROM nurse n JOIN medication m ON n.nurse_nu...   4 row(s) returned                                          0.000 sec / 0.000 sec
309  21:51:21  SELECT room_num, SUM(r.nightly_fee*p.num_nights) FROM patient p JOIN room r ON p.roo...  Error Code: 1052. Column 'room_num' in field list is ambiguous   0.000 sec
310  21:51:30  SELECT r.room_num, SUM(r.nightly_fee*p.num_nights) FROM patient p JOIN room r ON p.ro...  4 row(s) returned                                          0.000 sec / 0.000 sec
311  21:52:03  SELECT r.room_num, SUM(r.nightly_fee*p.num_nights) AS Total_Room_Fees FROM patient...   4 row(s) returned                                          0.000 sec / 0.000 sec
```

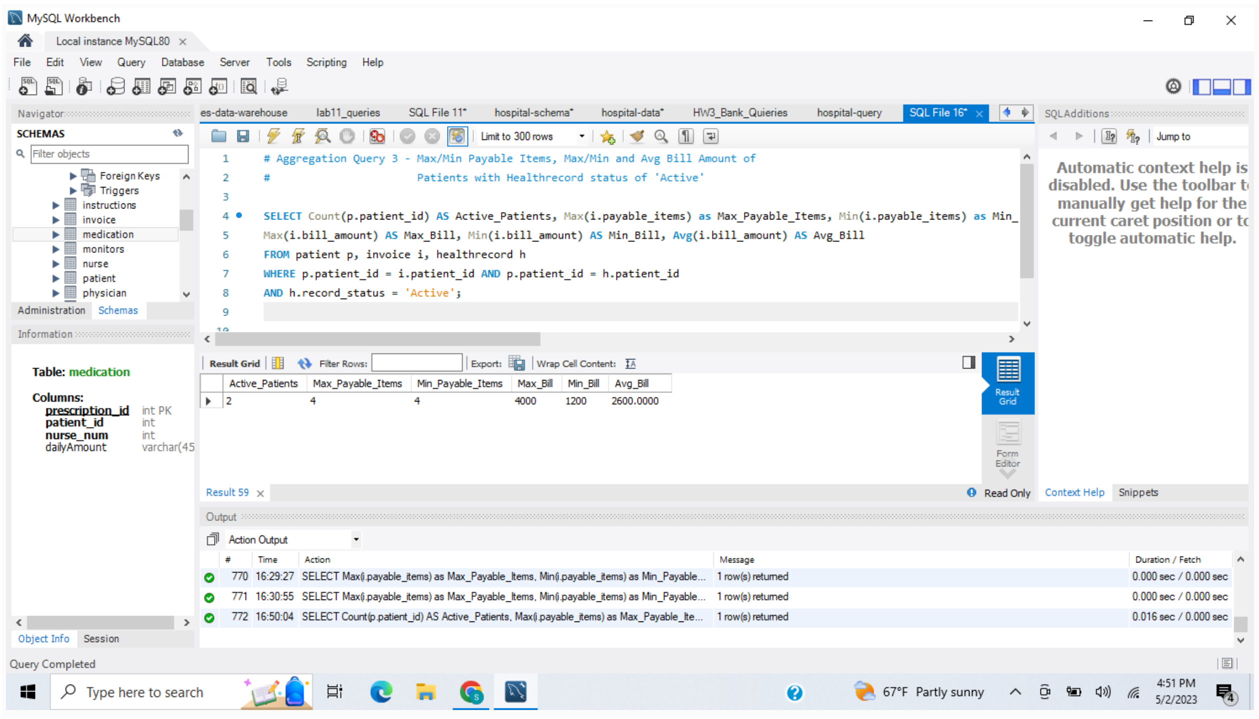
Query Completed

Query 6- Max/Min Payable Items, Max/Min and Avg Bill Amount of Patients with Healthrecord status of 'Active', also Count of Active Patients

This query serves as an accounting tool by providing a numeratic summary of funds and items owed by active patients in the system. The query joins the healthrecord, patient, and invoice tables on the patients’ IDs and narrows the search to patients with an ‘Active’ healthrecord status. Next the query selects and returns the number of patients, the maximum and minimum number of payable items, and the maximum, minimum and average bill amount of all of those patients by using the Count, Max, Min, and Avg aggregation operators provided by MYSQL. The query reads as such:

```
SELECT Count(p.patient_id) AS Active_Patients, Max(i.payable_items) as
Max_Payable_Items, Min(i.payable_items) as Min_Payable_Items,
Max(i.bill_amount) AS Max_Bill, Min(i.bill_amount) AS Min_Bill, Avg(i.bill_amount) AS
Avg_Bill
FROM patient p, invoice i, healthrecord h
WHERE p.patient_id = i.patient_id AND p.patient_id = h.patient_id
AND h.record_status = 'Active';
```

The screenshot below shows the results of *Query 6* in our database.



```
# Aggregation Query 3 - Max/Min Payable Items, Max/Min and Avg Bill Amount of
#                       Patients with Healthrecord status of 'Active'

SELECT Count(p.patient_id) AS Active_Patients, Max(i.payable_items) as Max_Payable_Items, Min(i.payable_items) as Min_
Max(i.bill_amount) AS Max_Bill, Min(i.bill_amount) AS Min_Bill, Avg(i.bill_amount) AS Avg_Bill
FROM patient p, invoice i, healthrecord h
WHERE p.patient_id = i.patient_id AND p.patient_id = h.patient_id
AND h.record_status = 'Active';
```

Active_Patients	Max_Payable_Items	Min_Payable_Items	Max_Bill	Min_Bill	Avg_Bill
2	4	4	4000	1200	2600.0000

Result 59

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or tc toggle automatic help.

Table: medication

Columns:

prescription_id	int PK
patient_id	int
nurse_num	int
dailyAmount	varchar(45

```
Action Output
#    Time      Action                                                                     Message            Duration / Fetch
770  16:29:27  SELECT Max(i.payable_items) as Max_Payable_Items, Min(i.payable_items) as Min_Payable...  1 row(s) returned  0.000 sec / 0.000 sec
771  16:30:55  SELECT Max(i.payable_items) as Max_Payable_Items, Min(i.payable_items) as Min_Payable...  1 row(s) returned  0.000 sec / 0.000 sec
772  16:50:04  SELECT Count(p.patient_id) AS Active_Patients, Max(i.payable_items) as Max_Payable_Ite...  1 row(s) returned  0.016 sec / 0.000 sec
```

c. NESTED QUERIES

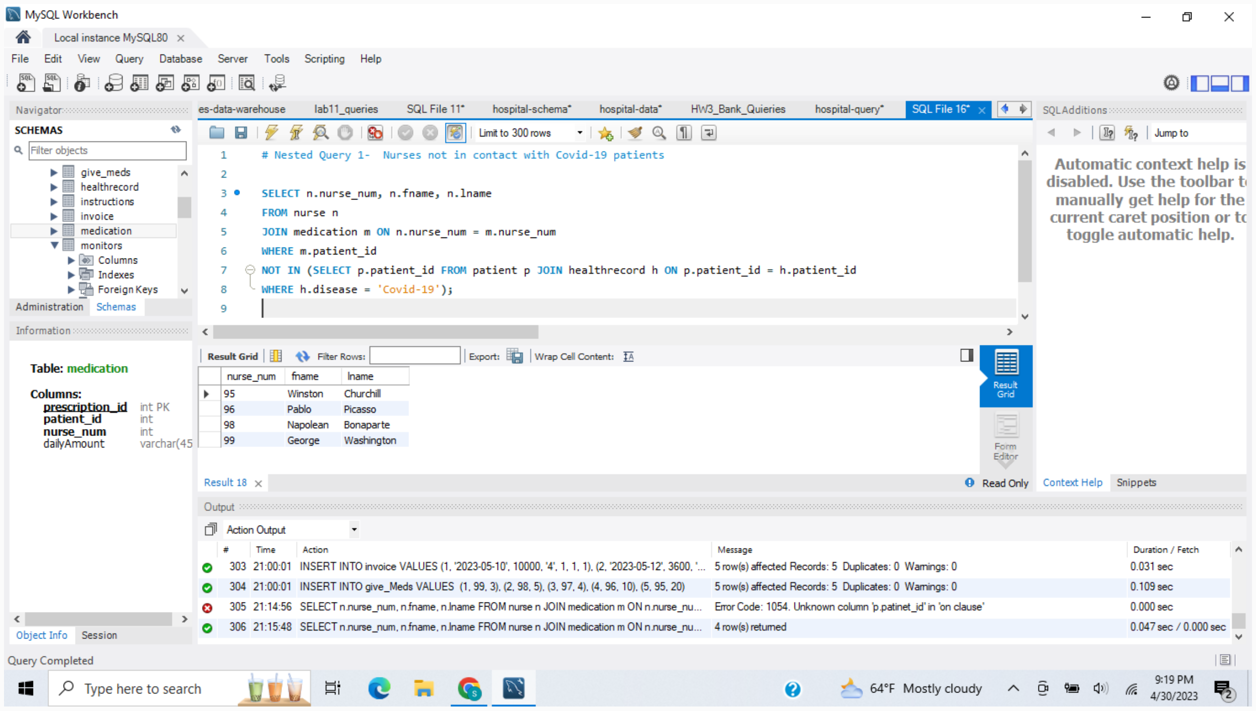
Query 7- Nurses not in contact with Covid-19 patients

This nested query determines and shows nurses that have not administered medications to patients with Covid-19. This would be an especially useful query for a hospital to use during the pandemic to determine which nurses are less at risk for getting Covid-19. In order to find these nurses from the database we performed an inner query that determined patient IDs' belonging to patients that have Covid-19 (by joining the Patient and Healthrecord tables and where statement), then a "NOT IN" set comparison operator was used to eliminate these patients' IDs from the Nurse and Medication select statement.

```
SELECT n.nurse_num, n.fname, n.lname
FROM nurse n
JOIN medication m ON n.nurse_num = m.nurse_num
WHERE m.patient_id
```

NOT IN (SELECT p.patient_id FROM patient p JOIN healthrecord h ON p.patient_id = h.patient_id
WHERE h.disease = 'Covid-19');

The following screenshot shows the results of *Query 7.* Note, the nurse with nurse_num= 97 administers medication to the Covid-19 patient and she is not shown in the resulting table.



```
# Nested Query 1-  Nurses not in contact with Covid-19 patients

SELECT n.nurse_num, n.fname, n.lname
FROM nurse n
JOIN medication m ON n.nurse_num = m.nurse_num
WHERE m.patient_id
NOT IN (SELECT p.patient_id FROM patient p JOIN healthrecord h ON p.patient_id = h.patient_id
WHERE h.disease = 'Covid-19');
```

nurse_num	fname	lname
95	Winston	Churchill
96	Pablo	Picasso
98	Napolean	Bonaparte
99	George	Washington

Table: medication

Columns:
prescription_id int PK
patient_id int
nurse_num int
dailyAmount varchar(45

```
#   Time      Action                                                                              Message                                                      Duration / Fetch
303 21:00:01  INSERT INTO invoice VALUES (1, '2023-05-10', 10000, '4', 1, 1, 1), (2, '2023-05-12', 3600, '...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0   0.031 sec
304 21:00:01  INSERT INTO give_Meds VALUES  (1, 99, 3), (2, 98, 5), (3, 97, 4), (4, 96, 10), (5, 95, 20)  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0   0.109 sec
305 21:14:56  SELECT n.nurse_num, n.fname, n.lname FROM nurse n JOIN medication m ON n.nurse_nu...     Error Code: 1054. Unknown column 'p.patinet_id' in 'on clause'  0.000 sec
306 21:15:48  SELECT n.nurse_num, n.fname, n.lname FROM nurse n JOIN medication m ON n.nurse_nu...     4 row(s) returned                                            0.047 sec / 0.000 sec
```

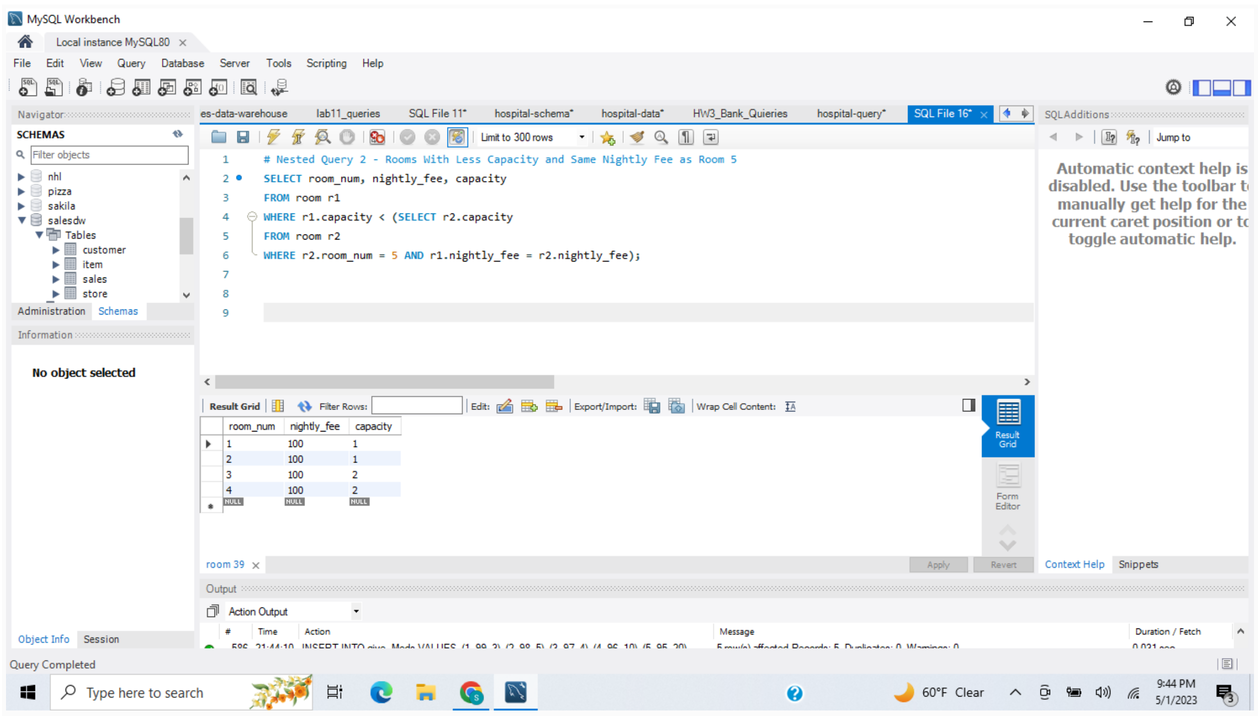
Query Completed

Query 8 - Rooms With Less Capacity and Same Nightly Fee as Room 5

The following query finds rooms with a smaller capacity than Room 5 with the same nightly fee as Room 5 by performing a nested query. This query would be useful for patients looking to change rooms because they require more personal space, but need to stay within their budget. The inner query determines Room 5's capacity and finds a room (r2) with the same fee of the room in the outer query (r1), then the "Where" statement in the outer query uses a less than (<) set comparison operator to return only rooms with a smaller capacity than the results of the inner query. Here's the nested query described:

SELECT room_num, nightly_fee, capacity
FROM room r1
WHERE r1.capacity < (SELECT r2.capacity
FROM room r2
WHERE r2.room_num = 5 AND r1.nightly_fee = r2.nightly_fee);

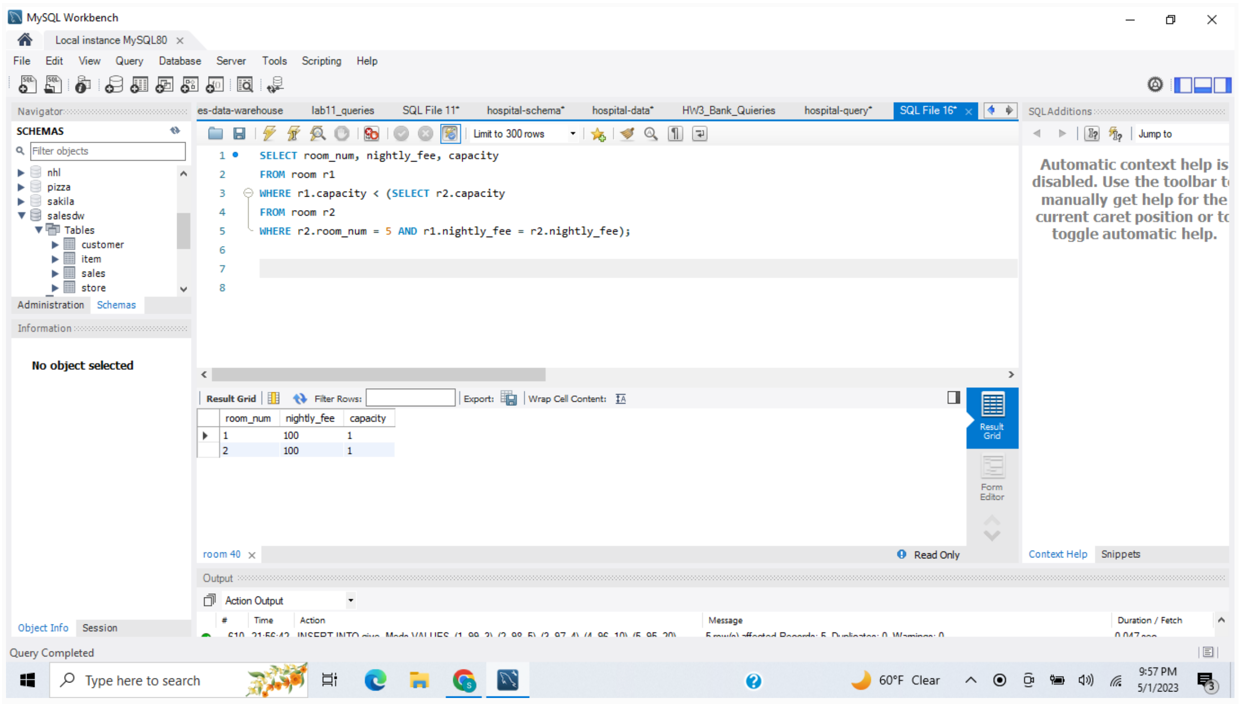
The following screenshot depicts this nested query when run on our hospital database. In our current database, Room 5 has a capacity of five and a nightly fee of $100. So the results show Rooms 1 through 4 all have the same nightly fee of $100 and a smaller room capacity than Room 5.



```sql
# Nested Query 2 - Rooms With Less Capacity and Same Nightly Fee as Room 5
SELECT room_num, nightly_fee, capacity
FROM room r1
WHERE r1.capacity < (SELECT r2.capacity
FROM room r2
WHERE r2.room_num = 5 AND r1.nightly_fee = r2.nightly_fee);
```

room_num	nightly_fee	capacity
1	100	1
2	100	1
3	100	2
4	100	2
NULL	NULL	NULL

Since our rooms' values are all similar to each other, and in order to further confirm the query is working properly, I temporarily adjusted Room 3 to have a nightly fee of $200 and Room 4 to have a capacity of six. The following screenshot depicts the same nested query, with the temporary values, and shows only Rooms 1 and 2 have the same nightly fee of Room 5 and a smaller capacity.



```sql
SELECT room_num, nightly_fee, capacity
FROM room r1
WHERE r1.capacity < (SELECT r2.capacity
FROM room r2
WHERE r2.room_num = 5 AND r1.nightly_fee = r2.nightly_fee);
```

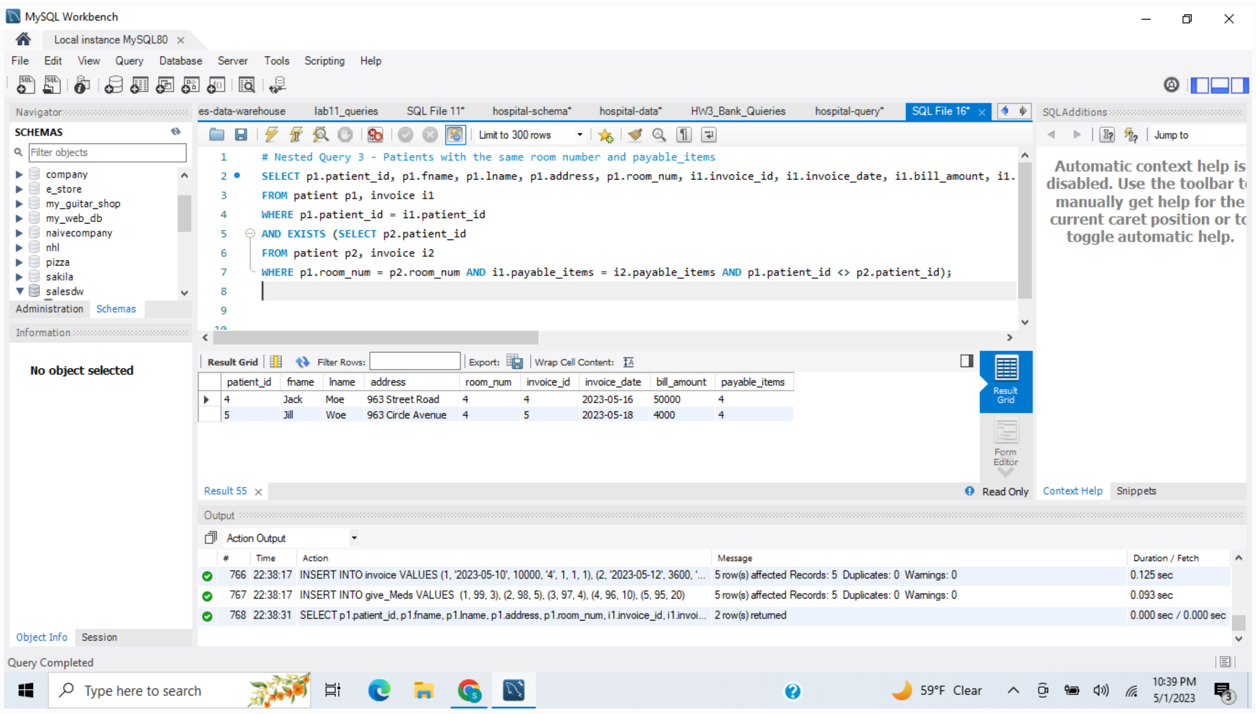
room_num	nightly_fee	capacity
1	100	1
2	100	1

Query 9 - Patients sharing the same room and with the same payable items

The next nested query returns patients' information when two patients share a room and have the same amount of payable items. The inner query finds if a second patient exists that shares a room and has the same amount of payable items as the patient in the outer query. The inner query also determines if the patient ID's are different (this part is very important). Both the patient and invoice tables are required in both the outer and inner queries. Finally, both patients' information from the patient and invoice table are returned from the outer query's select statement. This type of query may be useful for differentiating patients that have very similar records. The query is as follows:

```
SELECT p1.patient_id, p1.fname, p1.lname, p1.address, p1.room_num, i1.invoice_id,
i1.invoice_date, i1.bill_amount, i1.payable_items
FROM patient p1, invoice i1
WHERE p1.patient_id = i1.patient_id
AND EXISTS (SELECT p2.patient_id
FROM patient p2, invoice i2
WHERE p1.room_num = p2.room_num AND i1.payable_items = i2.payable_items AND
p1.patient_id <> p2.patient_id);
```

The following screenshot shows the results of the query in our database. It found "Jack Moe" and "Jill Woe" share the same room number (4) and have the same amount of payable items (4).



```
# Nested Query 3 - Patients with the same room number and payable_items
SELECT p1.patient_id, p1.fname, p1.lname, p1.address, p1.room_num, i1.invoice_id, i1.invoice_date, i1.bill_amount, i1.
FROM patient p1, invoice i1
WHERE p1.patient_id = i1.patient_id
AND EXISTS (SELECT p2.patient_id
FROM patient p2, invoice i2
WHERE p1.room_num = p2.room_num AND i1.payable_items = i2.payable_items AND p1.patient_id <> p2.patient_id);
```

patient_id	fname	lname	address	room_num	invoice_id	invoice_date	bill_amount	payable_items
4	Jack	Moe	963 Street Road	4	4	2023-05-16	50000	4
5	Jill	Woe	963 Circle Avenue	4	5	2023-05-18	4000	4

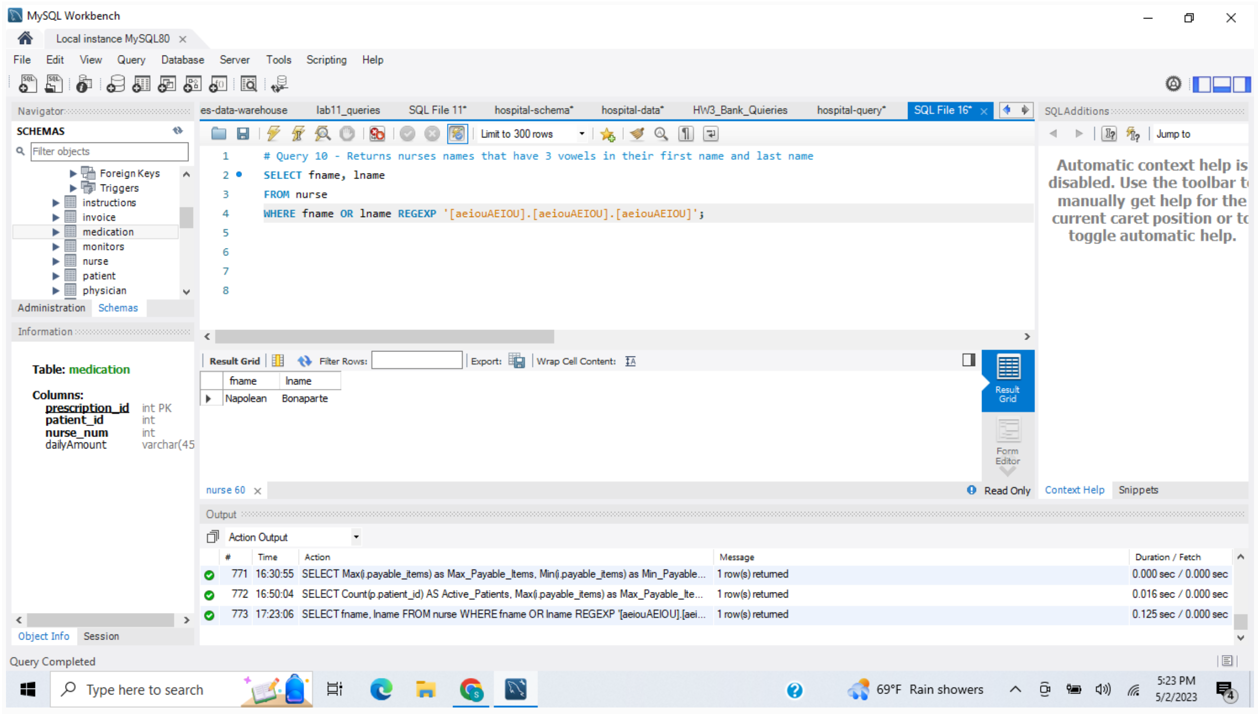
```
#    Time      Action                                                                        Message                                                          Duration / Fetch
766  22:38:17  INSERT INTO invoice VALUES (1, '2023-05-10', 10000, '4', 1, 1, 1), (2, '2023-05-12', 3600, '...   5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0   0.125 sec
767  22:38:17  INSERT INTO give_Meds VALUES (1, 99, 3), (2, 98, 5), (3, 97, 4), (4, 96, 10), (5, 95, 20)       5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0   0.093 sec
768  22:38:31  SELECT p1.patient_id, p1.fname, p1.lname, p1.address, p1.room_num, i1.invoice_id, i1.invoi...   2 row(s) returned                                          0.000 sec / 0.000 sec
```

d. OTHER QUERIES

Query 10 - Returns nurses' names that have 3 vowels in their first name and last name.

SELECT fname, lname
FROM nurse
WHERE fname OR lname REGEXP '[aeiouAEIOU].[aeiouAEIOU].[aeiouAEIOU]';

Query 10 Screenshot:



```
# Query 10 - Returns nurses names that have 3 vowels in their first name and last name
SELECT fname, lname
FROM nurse
WHERE fname OR lname REGEXP '[aeiouAEIOU].[aeiouAEIOU].[aeiouAEIOU]';
```

fname	lname
Napolean	Bonaparte

Table: medication

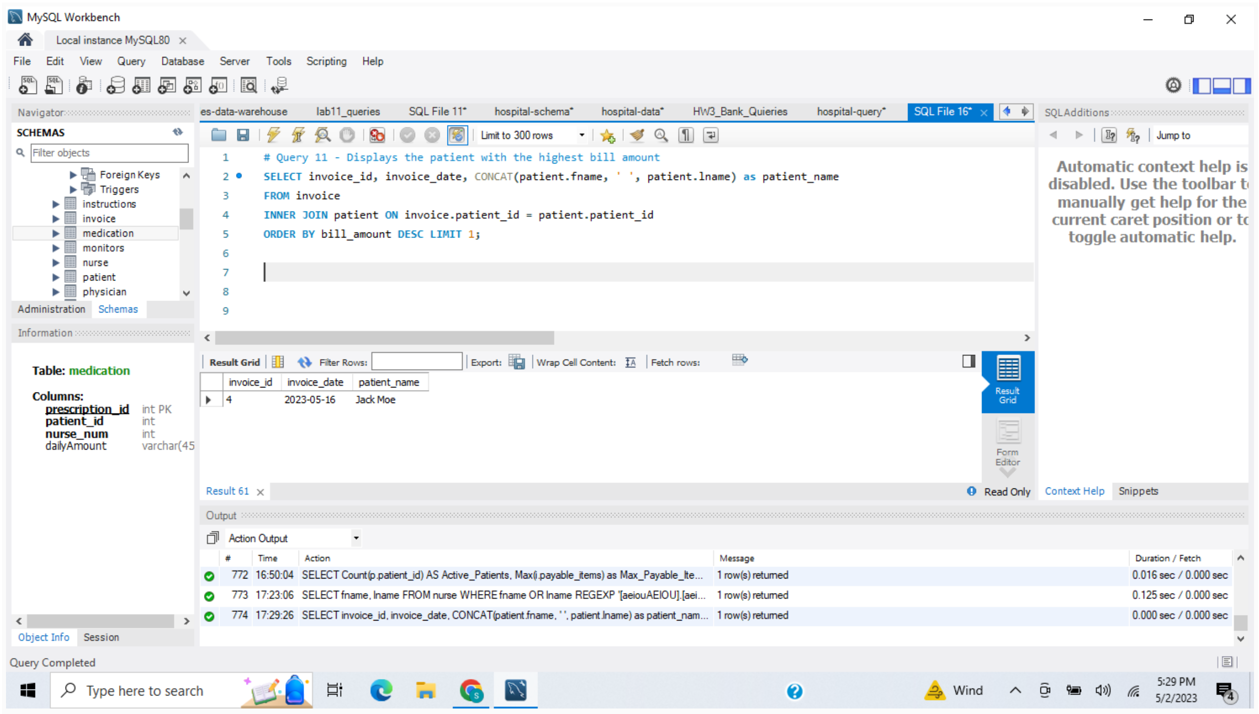
Columns:
- prescription_id int PK
- patient_id int
- nurse_num int
- dailyAmount varchar(45

```
#    Time      Action                                                                       Message             Duration / Fetch
771  16:30:55  SELECT Max(i.payable_items) as Max_Payable_Items, Min(i.payable_items) as Min_Payable...   1 row(s) returned   0.000 sec / 0.000 sec
772  16:50:04  SELECT Count(p.patient_id) AS Active_Patients, Max(i.payable_items) as Max_Payable_Ite...  1 row(s) returned   0.016 sec / 0.000 sec
773  17:23:06  SELECT fname, lname FROM nurse WHERE fname OR lname REGEXP '[aeiouAEIOU].[aei...           1 row(s) returned   0.125 sec / 0.000 sec
```

Query 11 - Displays the patient with the highest bill amount

SELECT invoice_id, invoice_date, CONCAT(patient.fname, ' ', patient.lname) as patient_name
FROM invoice
INNER JOIN patient ON invoice.patient_id = patient.patient_id
ORDER BY bill_amount DESC LIMIT 1;

Query 11 Screenshot (next page)*:*



```
# Query 11 - Displays the patient with the highest bill amount
SELECT invoice_id, invoice_date, CONCAT(patient.fname, ' ', patient.lname) as patient_name
FROM invoice
INNER JOIN patient ON invoice.patient_id = patient.patient_id
ORDER BY bill_amount DESC LIMIT 1;
```

invoice_id	invoice_date	patient_name
4	2023-05-16	Jack Moe

Table: medication

Columns:

prescription_id	int PK
patient_id	int
nurse_num	int
dailyAmount	varchar(45

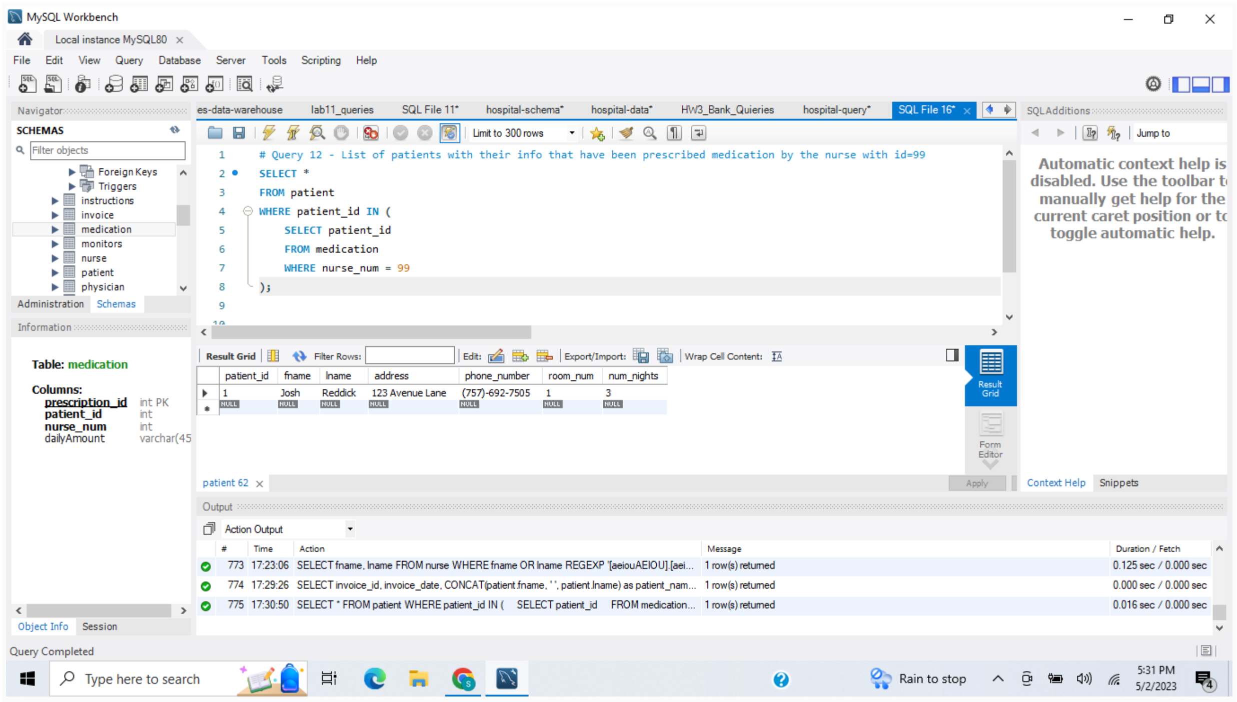
```
#    Time      Action                                                                          Message             Duration / Fetch
772  16:50:04  SELECT Count(p.patient_id) AS Active_Patients, Max(i.payable_items) as Max_Payable_Ite...  1 row(s) returned   0.016 sec / 0.000 sec
773  17:23:06  SELECT fname, lname FROM nurse WHERE fname OR lname REGEXP '[aeiouAEIOU].[aei...  1 row(s) returned   0.125 sec / 0.000 sec
774  17:29:26  SELECT invoice_id, invoice_date, CONCAT(patient.fname, ' ', patient.lname) as patient_nam...  1 row(s) returned   0.000 sec / 0.000 sec
```

Query Completed

Query 12 - List of patients with their info that have been prescribed medication by the nurse with id=99

```
SELECT *
FROM patient
WHERE patient_id IN (
   SELECT patient_id
   FROM medication
   WHERE nurse_num = 99
);
```

Query 12 Screenshot (next page):



```
# Query 12 - List of patients with their info that have been prescribed medication by the nurse with id=99
SELECT *
FROM patient
WHERE patient_id IN (
    SELECT patient_id
    FROM medication
    WHERE nurse_num = 99
);
```

patient_id	fname	lname	address	phone_number	room_num	num_nights
1	Josh	Reddick	123 Avenue Lane	(757)-692-7505	1	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
#    Time      Action                                                                        Message            Duration / Fetch
773  17:23:06  SELECT fname, lname FROM nurse WHERE fname OR lname REGEXP '[aeiouAEIOU].[aei...  1 row(s) returned  0.125 sec / 0.000 sec
774  17:29:26  SELECT invoice_id, invoice_date, CONCAT(patient.fname, ' ', patient.lname) as patient_nam...  1 row(s) returned  0.000 sec / 0.000 sec
775  17:30:50  SELECT * FROM patient WHERE patient_id IN (    SELECT patient_id    FROM medication...  1 row(s) returned  0.016 sec / 0.000 sec
```

Table: medication

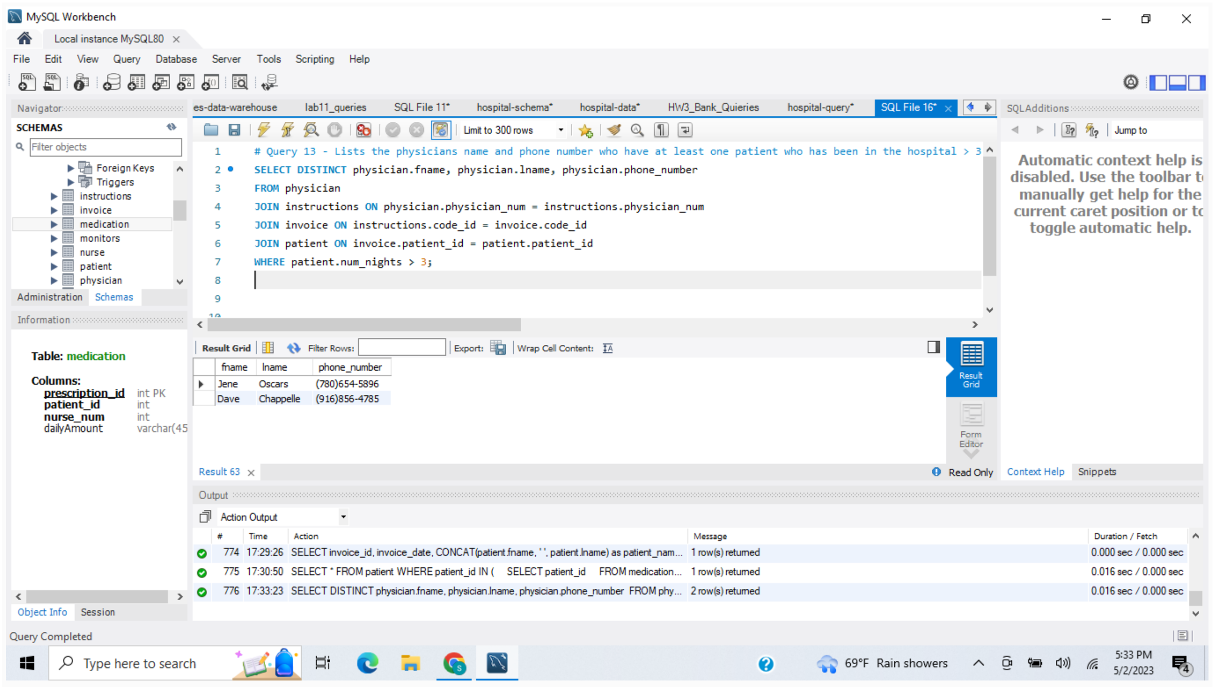
Columns:

prescription_id	int PK
patient_id	int
nurse_num	int
dailyAmount	varchar(45

Query 13 - Lists the physicians name and phone number who have at least one patient who has been in the hospital > 3 days

```
SELECT DISTINCT physician.fname, physician.lname, physician.phone_number
FROM physician
JOIN instructions ON physician.physician_num = instructions.physician_num
JOIN invoice ON instructions.code_id = invoice.code_id
JOIN patient ON invoice.patient_id = patient.patient_id
WHERE patient.num_nights > 3;
```

Query 13 Screenshot (next page):



```
# Query 13 - Lists the physicians name and phone number who have at least one patient who has been in the hospital > 3
SELECT DISTINCT physician.fname, physician.lname, physician.phone_number
FROM physician
JOIN instructions ON physician.physician_num = instructions.physician_num
JOIN invoice ON instructions.code_id = invoice.code_id
JOIN patient ON invoice.patient_id = patient.patient_id
WHERE patient.num_nights > 3;
```

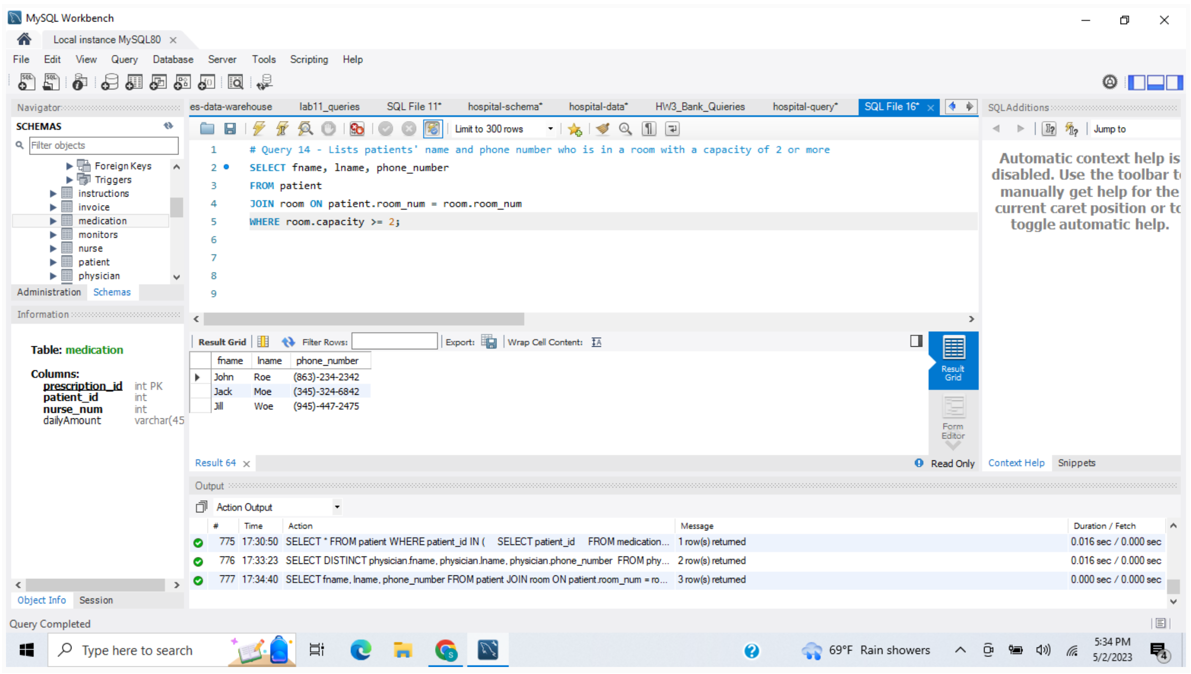
fname	lname	phone_number
Jene	Oscars	(780)654-5896
Dave	Chappelle	(916)856-4785

```
#    Time      Action                                                                          Message            Duration / Fetch
774  17:29:26  SELECT invoice_id, invoice_date, CONCAT(patient.fname, ' ', patient.lname) as patient_nam...  1 row(s) returned  0.000 sec / 0.000 sec
775  17:30:50  SELECT * FROM patient WHERE patient_id IN (    SELECT patient_id    FROM medication...         1 row(s) returned  0.016 sec / 0.000 sec
776  17:33:23  SELECT DISTINCT physician.fname, physician.lname, physician.phone_number  FROM phy...          2 row(s) returned  0.016 sec / 0.000 sec
```

Query 14 - Lists patients' name and phone number who is in a room with a capacity of 2 or more

SELECT fname, lname, phone_number
FROM patient
JOIN room ON patient.room_num = room.room_num
WHERE room.capacity >= 2;

Query 14 Screenshot:



```
# Query 14 - Lists patients' name and phone number who is in a room with a capacity of 2 or more
SELECT fname, lname, phone_number
FROM patient
JOIN room ON patient.room_num = room.room_num
WHERE room.capacity >= 2;
```

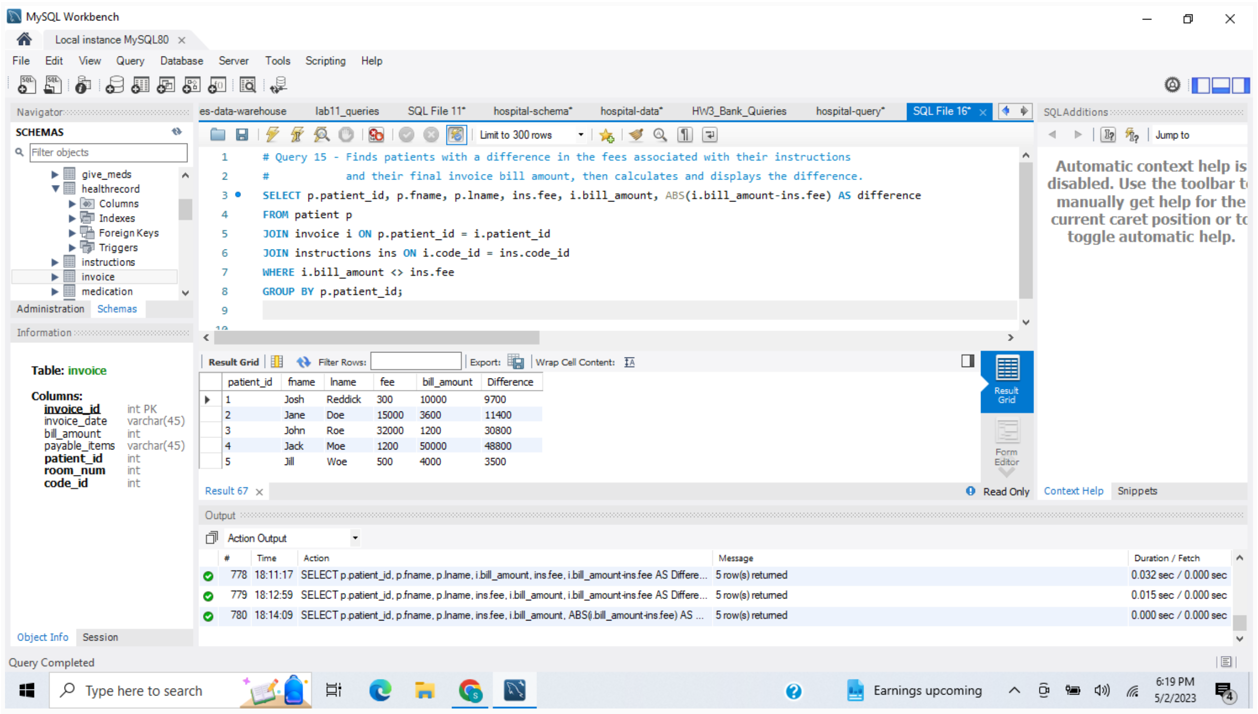
fname	lname	phone_number
John	Roe	(863)-234-2342
Jack	Moe	(345)-324-6842
Jill	Woe	(945)-447-2475

```
#    Time      Action                                                                                   Message            Duration / Fetch
775  17:30:50  SELECT * FROM patient WHERE patient_id IN (    SELECT patient_id    FROM medication...                  1 row(s) returned  0.016 sec / 0.000 sec
776  17:33:23  SELECT DISTINCT physician.fname, physician.lname, physician.phone_number  FROM phy...                   2 row(s) returned  0.016 sec / 0.000 sec
777  17:34:40  SELECT fname, lname, phone_number FROM patient JOIN room ON patient.room_num = ro...                    3 row(s) returned  0.000 sec / 0.000 sec
```

Query 15 - Finds patients with a difference in the fees associated with their instructions and their final invoice bill amount, then calculates and displays the difference.

```
SELECT p.patient_id, p.fname, p.lname, ins.fee, i.bill_amount,
ABS(i.bill_amount-ins.fee) AS difference
FROM patient p
JOIN invoice i ON p.patient_id = i.patient_id
JOIN instructions ins ON i.code_id = ins.code_id
WHERE i.bill_amount <> ins.fee
GROUP BY p.patient_id;
```

Query 15 Screenshot:



```
# Query 15 - Finds patients with a difference in the fees associated with their instructions
#           and their final invoice bill amount, then calculates and displays the difference.
SELECT p.patient_id, p.fname, p.lname, ins.fee, i.bill_amount, ABS(i.bill_amount-ins.fee) AS difference
FROM patient p
JOIN invoice i ON p.patient_id = i.patient_id
JOIN instructions ins ON i.code_id = ins.code_id
WHERE i.bill_amount <> ins.fee
GROUP BY p.patient_id;
```

patient_id	fname	lname	fee	bill_amount	Difference
1	Josh	Reddick	300	10000	9700
2	Jane	Doe	15000	3600	11400
3	John	Roe	32000	1200	30800
4	Jack	Moe	1200	50000	48800
5	Jill	Woe	500	4000	3500

Table: invoice

Columns:

invoice_id	int PK
invoice_date	varchar(45)
bill_amount	int
payable_items	varchar(45)
patient_id	int
room_num	int
code_id	int

```
#    Time      Action                                                                                   Message            Duration / Fetch
778  18:11:17  SELECT p.patient_id, p.fname, p.lname, i.bill_amount, ins.fee, i.bill_amount-ins.fee AS Differe...  5 row(s) returned  0.032 sec / 0.000 sec
779  18:12:59  SELECT p.patient_id, p.fname, p.lname, ins.fee, i.bill_amount, i.bill_amount-ins.fee AS Differe...  5 row(s) returned  0.015 sec / 0.000 sec
780  18:14:09  SELECT p.patient_id, p.fname, p.lname, ins.fee, i.bill_amount, ABS(i.bill_amount-ins.fee) AS ...    5 row(s) returned  0.000 sec / 0.000 sec
```

Query Completed