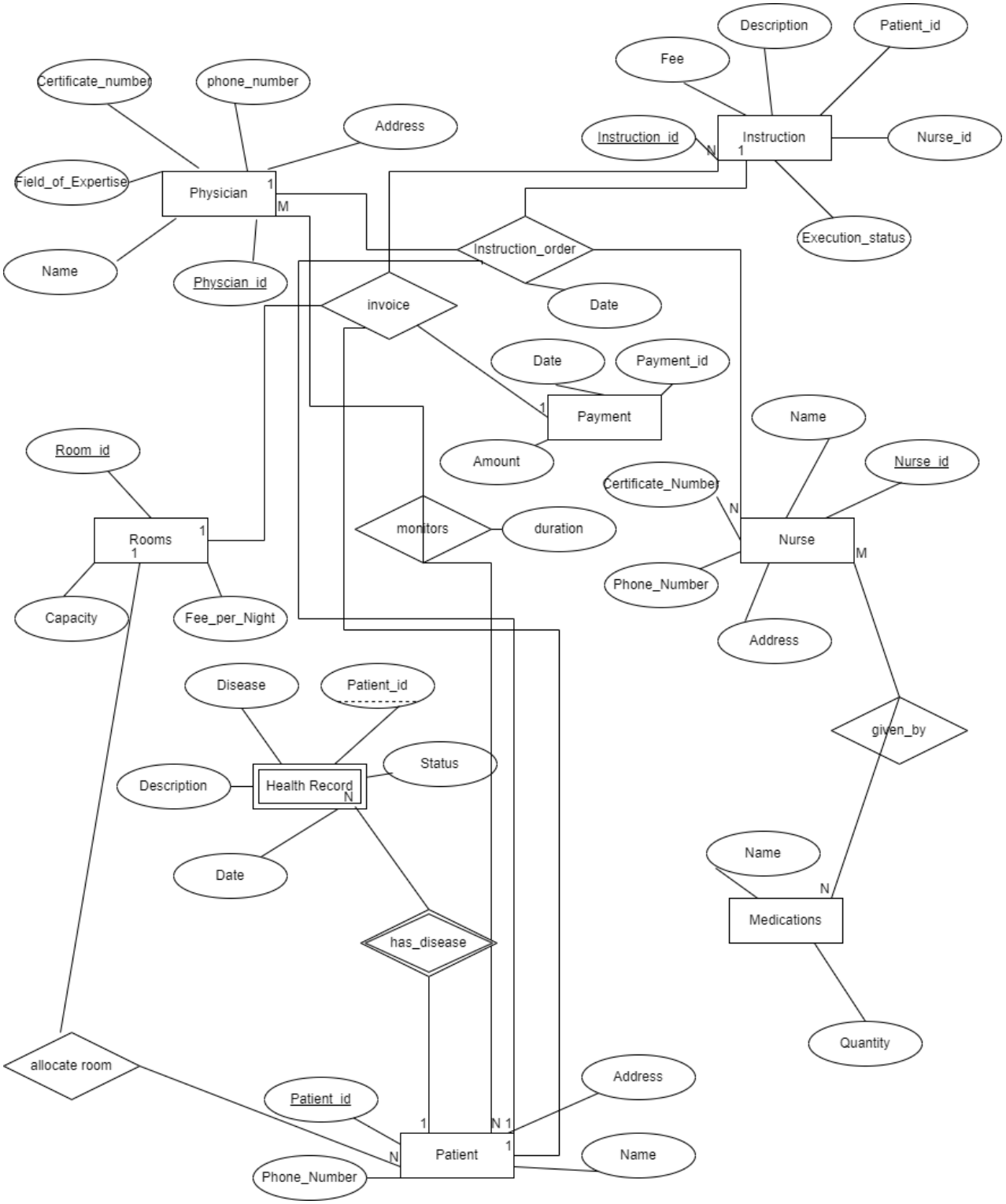
Project Report

Data Base Name: *Hospital*

Kalyan Venkat Madireddy

ER- Diagram



Some of the Assumptions made:

1) The assumption is that one room can be occupied by only one patient.

2) Many health records are assumed to be associated with each patient.

3) Physicians can specialize in multiple fields of expertise.

4) A weak entity is used to identify medication.

5) The foreign key for Health Record is the Patient ID.

Relations and Keys:

1. **Patients** (patient_ID, patient_name, address, phoneNumber)
   primary key: { patient_ID}
   foreign key: {}
2. **Physician** (physician_ID, physician_name, address, field_Exp, phoneNumber, certificate_Number)
   primary key: {physician_ID}
   foreign key: {}
3. **Nurse**(nurse_id, name, certification_num, address, phoneNumber, physician_ID)
   primary key: { nurse_id }
   foreign key: { physician_ID references physician(physician_ID)}
4. **Medication**(medication_id, name, amount, patient_id, nurse_id)
   primary key: { medication_id }
   foreign key: {patient_id references s patients(patient_id), nurse_id references nurse(nurse_id)}
5. **Room** (room_num, capacity, fee_per_night, patient_ID)
   primary key: { room_num }
   foreign key: { patient_ID references patients(patient_ID) }
6. **Health_Record**(:record_id, patient_ID, description, date, status, disease_history)
   primary key: {}
   foreign key: { patient_ID references patients(patient_ID) }
7. **Payment**(payment_id, patient_ID, room_id, amount, date)
   primary key: { payment_id}
   foreign key: { patient_ID references patients(patient_ID), room_id references rooms(room_num) }

Queries, descriptions, and results:

views and description:

1. **Patient Drug/Medication View**
   Description: The following attachment presents a screenshot of the results obtained from a View query that merges the columns from the Patient and Medications tables. This query showcases the drugs/medicines prescribed to patients along with their respective quantities.

```
USE hospital;

CREATE VIEW Patient_treatment_View AS
SELECT pat.patient_name AS patient_Name, med.Name AS treatment_medication, med.amount AS Drug_Capacity
FROM patients pat, medications med
WHERE pat.Patient_ID = med.Patient_id;

select * from Patient_treatment_View;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

patient_Name	treatment_medication	Drug_Capacity
Kevin	Paracetamal	111.54
Sachin	neflon	134.56
Sehwag	burnol	200.54
Pat	anti acid	234.76
Matt	Azitromycin	33.65

2. **Patient Disease Suffering View**

Description: The following attachment presents a screenshot of the results obtained from a View query that merges the columns from the Patient and health record tables. This query showcases the disease the patients are suffering from along with the reported date of the disease.

```
# view 2

CREATE VIEW Patient_Disease_View AS
SELECT pat.patient_name AS patient_Name, record.description AS Disease, record.date AS reported_date
FROM patients pat,healthRecord record
WHERE pat.Patient_ID = record.Patient_ID;

select * from Patient_Disease_View;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

patient_Name	Disease	reported_date
Kevin	Malaria	2022-09-01
Sachin	Diarrhea	2020-08-14
Sehwag	allergy	2021-11-21
Pat	stomach pain	2021-11-04
Matt	red eye	2019-07-08

Patient_Disease_View 2

3. **Patient Bill View**

Description: The following attachment presents a screenshot of the results obtained from a View query that merges the columns from the Patient and payment tables. This query shows the bill amount generated for the patients along with the room number in which they are kept in, during their stay at the hospital.

```sql
#view 3

CREATE VIEW Patient_bill_View AS
SELECT pat.patient_name AS patient_Name,bill.amount AS bill, bill.room_id AS patient_kept_room
FROM patients pat,payments bill
WHERE pat.Patient_ID = bill.Patient_ID;

select * from Patient_bill_View;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

patient_Name	bill	patient_kept_room
Kevin	168.98	201
Sachin	150.77	202
Sehwag	200.98	203
Pat	238.98	204
Matt	300.54	205

Triggers and Description:

1. **update_health_every_time**
   Description: This trigger is executed following the insertion in the patient relation. It runs every time a new patient is added, and its purpose is to insert a record into the health_record relation using the patient_id of the recently added patient. Here's the screen shot showing the syntax and the result.

```sql
# Trigger 1
# Checking room capacity before allocating the room to the patient
#example of before trigger

DELIMITER //
CREATE TRIGGER update_health_every_time
AFTER INSERT ON patients
FOR EACH ROW
BEGIN
INSERT INTO healthRecord (patient_ID, status, date, description)
VALUES (NEW.patient_ID, 'NEW', NULL, NULL);
END;
//
DELIMITER ;


# Trigger 2
delimiter //
```

Output

Action Output

#	Time	Action	Message
41	19:52:18	USE hospital	0 row(s) affected
42	19:52:18	USE hospital	0 row(s) affected
43	19:52:19	USE hospital	0 row(s) affected
44	19:52:19	USE hospital	0 row(s) affected
45	19:52:28	CREATE TRIGGER update_health_every_time AFTER INSERT ON patients FOR E...	0 row(s) affected

2. **check_availability**

Description: This trigger checks for the available capacity of the rooms, every time we try to allocate the room beforehand the room is allocated for the patient based on the current and max capacity allowed. Here's the screen shot showing the syntax and the result.

```
47    # Trigger 2
48    delimiter //
49    CREATE TRIGGER check_availability
50    BEFORE INSERT ON rooms
51    FOR EACH ROW
52    BEGIN
53        DECLARE allowed_capacity INT;
54        DECLARE current_capacity INT;
55        SET allowed_capacity = (
56        SELECT capacity
57        FROM rooms
58        WHERE room_num = NEW.room_num);
59        SET current_capacity = (
60        SELECT COUNT(*)
61        FROM rooms
62        WHERE room_num = NEW.room_num);
63        IF allowed_capacity <= current_capacity THEN
64            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The room selected not having enough space';
65        END IF;
66    END;
67    //
```

Output

Action Output

#	Time	Action	Message
53	20:01:50	CREATE TRIGGER check_availability BEFORE INSERT ON rooms FOR EACH RO...	0 row(s) affected

3. **generate_bill**

Description: Every time a room is allocated, it's entered into the payments tab using the patient id. Here's the screen shot showing the syntax and the result.

```
70    # Trigger 3
71    DELIMITER //
72    CREATE TRIGGER generate_bill
73    AFTER INSERT ON rooms
74    FOR EACH ROW
75    BEGIN
76    INSERT INTO payments (patient_ID, room_id)
77    VALUES (NEW.patient_ID, NEW.room_num);
78    END;
79    //
80    DELIMITER ;
81
82
83
```

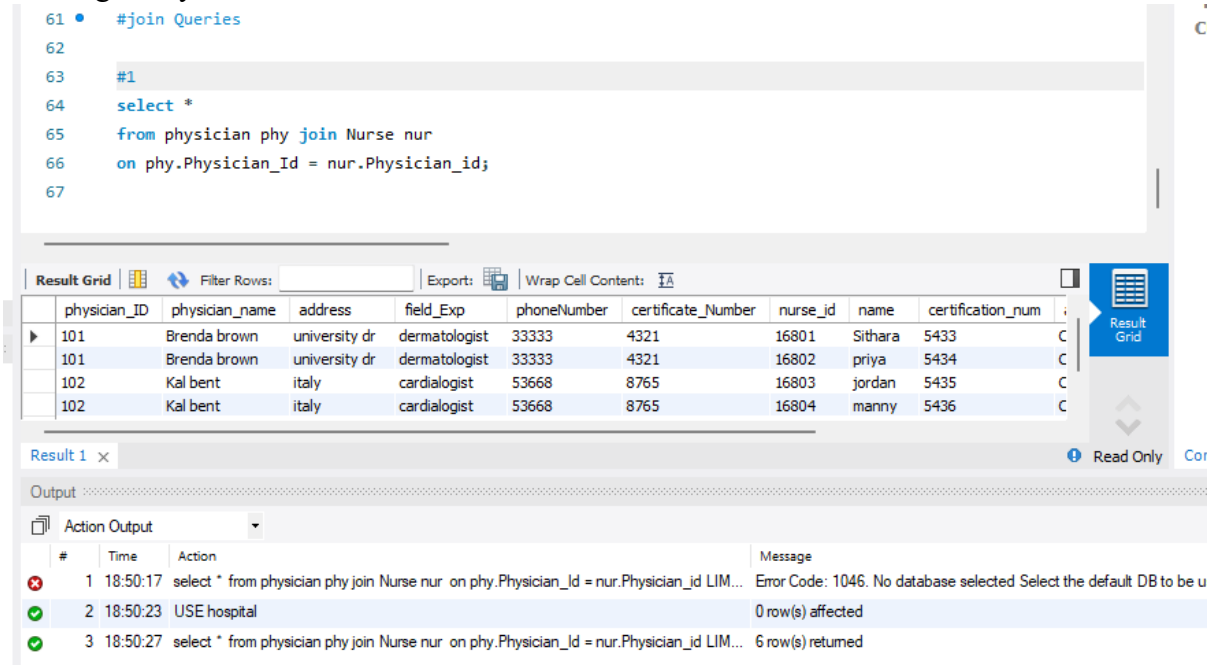
Output

Action Output

#	Time	Action	Message
54	20:02:51	CREATE TRIGGER check_availability BEFORE INSERT ON rooms FOR EACH RO...	Error Code: 1359. Trigger already exists
55	20:15:18	CREATE TRIGGER generate_bill AFTER INSERT ON rooms FOR EACH ROW BE...	0 row(s) affected

Join Queries:

1. **Joining physician and nurse relations**
   Description: Here we are joining the physician and the nurse relation tables when the physician id of the physician matches the nurse's physician id, it's a way for identifying which nurse is allocated to which physician. Here's the screen shot showing the syntax and the result.



```
61  #join Queries
62
63  #1
64  select *
65  from physician phy join Nurse nur
66  on phy.Physician_Id = nur.Physician_id;
67
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

physician_ID	physician_name	address	field_Exp	phoneNumber	certificate_Number	nurse_id	name	certification_num
101	Brenda brown	university dr	dermatologist	33333	4321	16801	Sithara	5433
101	Brenda brown	university dr	dermatologist	33333	4321	16802	priya	5434
102	Kal bent	italy	cardialogist	53668	8765	16803	jordan	5435
102	Kal bent	italy	cardialogist	53668	8765	16804	manny	5436

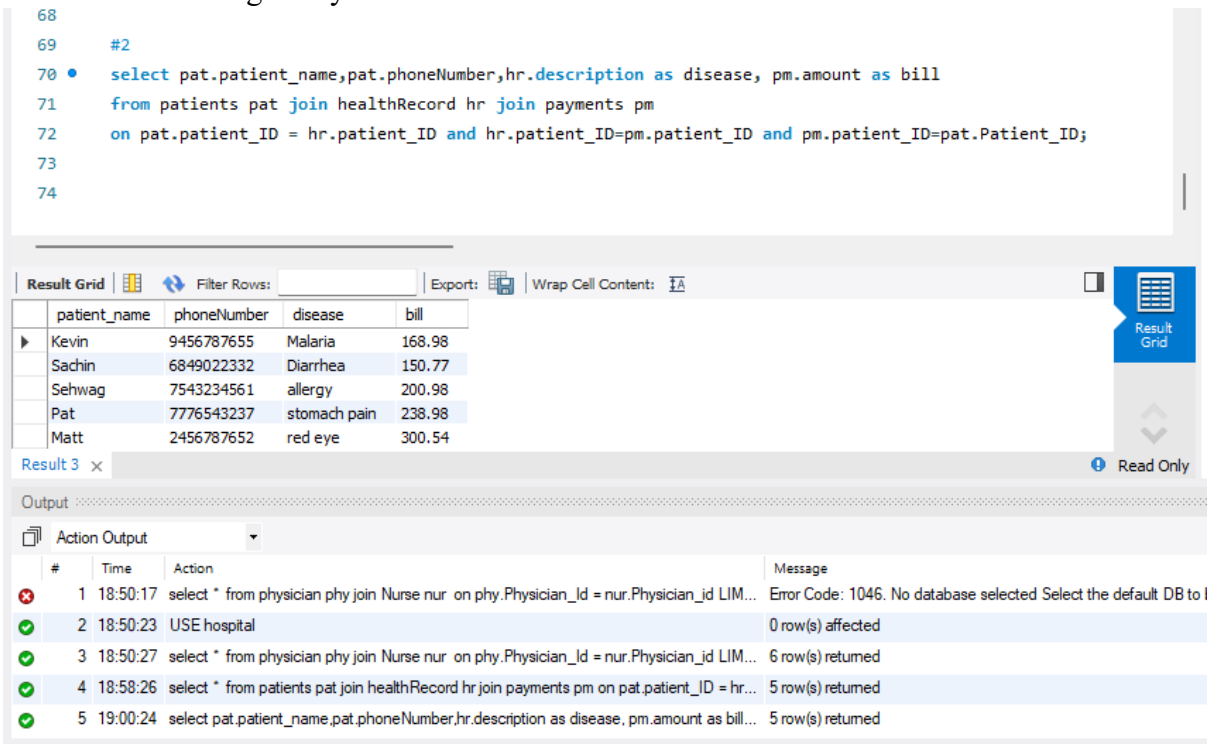
Result 1 | Read Only

Output

Action Output

#	Time	Action	Message
1	18:50:17	select * from physician phy join Nurse nur on phy.Physician_Id = nur.Physician_id LIM...	Error Code: 1046. No database selected Select the default DB to be u
2	18:50:23	USE hospital	0 row(s) affected
3	18:50:27	select * from physician phy join Nurse nur on phy.Physician_Id = nur.Physician_id LIM...	6 row(s) returned

2. **Joining patient, health record and payments relations**
   Description: This join query will join the 3 tables and shows the patient's name, his disease he is suffering from, the amount of bill generated for his treatment by joining all the three tables mentioned above using the patient id as the main key. Here's the screen shot showing the syntax and the result.



```
68
69  #2
70  select pat.patient_name,pat.phoneNumber,hr.description as disease, pm.amount as bill
71  from patients pat join healthRecord hr join payments pm
72  on pat.patient_ID = hr.patient_ID and hr.patient_ID=pm.patient_ID and pm.patient_ID=pat.Patient_ID;
73
74
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

patient_name	phoneNumber	disease	bill
Kevin	9456787655	Malaria	168.98
Sachin	6849022332	Diarrhea	150.77
Sehwag	7543234561	allergy	200.98
Pat	7776543237	stomach pain	238.98
Matt	2456787652	red eye	300.54

Result 3 | Read Only

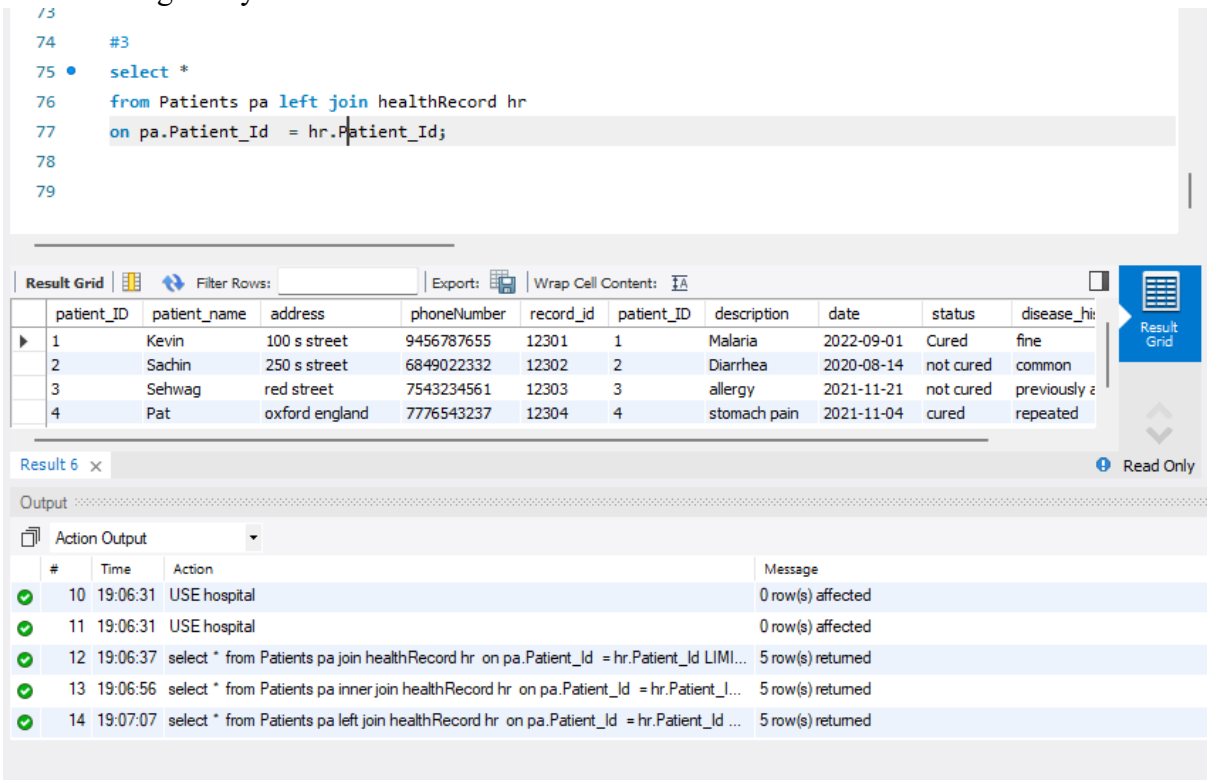
Output

Action Output

#	Time	Action	Message
1	18:50:17	select * from physician phy join Nurse nur on phy.Physician_Id = nur.Physician_id LIM...	Error Code: 1046. No database selected Select the default DB to l
2	18:50:23	USE hospital	0 row(s) affected
3	18:50:27	select * from physician phy join Nurse nur on phy.Physician_Id = nur.Physician_id LIM...	6 row(s) returned
4	18:58:26	select * from patients pat join healthRecord hr join payments pm on pat.patient_ID = hr...	5 row(s) returned
5	19:00:24	select pat.patient_name,pat.phoneNumber,hr.description as disease, pm.amount as bill...	5 row(s) returned

3. **Joining patients and health record using left join**

   Description: Here the two relationships are joined to one another using the left join relationship, the left join tends to include all those on the left table even if the match is not found on the right-hand side with the null values to fill it up. Here's the screen shot showing the syntax and the result.



```
#3
select *
from Patients pa left join healthRecord hr
on pa.Patient_Id  = hr.Patient_Id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

patient_ID	patient_name	address	phoneNumber	record_id	patient_ID	description	date	status	disease_hi
1	Kevin	100 s street	9456787655	12301	1	Malaria	2022-09-01	Cured	fine
2	Sachin	250 s street	6849022332	12302	2	Diarrhea	2020-08-14	not cured	common
3	Sehwag	red street	7543234561	12303	3	allergy	2021-11-21	not cured	previously a
4	Pat	oxford england	7776543237	12304	4	stomach pain	2021-11-04	cured	repeated

Result 6 | Read Only

Output

Action Output

#	Time	Action	Message
10	19:06:31	USE hospital	0 row(s) affected
11	19:06:31	USE hospital	0 row(s) affected
12	19:06:37	select * from Patients pa join healthRecord hr on pa.Patient_Id = hr.Patient_Id LIMI...	5 row(s) returned
13	19:06:56	select * from Patients pa inner join healthRecord hr on pa.Patient_Id = hr.Patient_I...	5 row(s) returned
14	19:07:07	select * from Patients pa left join healthRecord hr on pa.Patient_Id = hr.Patient_Id ...	5 row(s) returned

4. **Joining the patient and room using the right join**

   Description: Here the patient and the room relationships has been bought together using the right join. Same as the above, this right join tends to include the rows on the right table even though if the match is not found with the left side. Here's the screen shot showing the syntax and the result.

```
78
79      #4
80 •    select *
81      from Patients pa right join rooms ro
82      on pa.Patient_Id  = ro.Patient_Id;
83
84
85
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

patient_ID	patient_name	address	phoneNumber	room_num	capacity	fee_per_night	patient_ID
1	Kevin	100 s street	9456787655	201	30	100.00	1
2	Sachin	250 s street	6849022332	202	12	78.00	2
3	Sehwag	red street	7543234561	203	23	150.00	3
4	Pat	oxford england	7776543237	204	18	170.00	4
5	Matt	queens new york	2456787652	205	36	100.00	5

Result 7 | Read Only

Output

Action Output

#	Time	Action	Message
11	19:06:31	USE hospital	0 row(s) affected
12	19:06:37	select * from Patients pa join healthRecord hr on pa.Patient_Id = hr.Patient_Id LIMI...	5 row(s) returned
13	19:06:56	select * from Patients pa inner join healthRecord hr on pa.Patient_Id = hr.Patient_I...	5 row(s) returned
14	19:07:07	select * from Patients pa left join healthRecord hr on pa.Patient_Id = hr.Patient_Id ...	5 row(s) returned
15	19:12:14	select * from Patients pa right join rooms ro on pa.Patient_Id = ro.Patient_Id LIMIT ...	5 row(s) returned

5. **Joining the physician and nurse relations**

Description: Here, I'm joining the above-mentioned relations, to see which nurses are allocated to which physician. Here's the screen shot showing the syntax and the result.

```
108
109     #5
110
111 •   SELECT phy.physician_name AS physician_name, nur.name AS allocated_nurse
112     FROM physician phy
113     JOIN nurse nur ON phy.physician_ID = nur.physician_ID
114     GROUP BY phy.physician_name, nur.name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

physician_name	allocated_nurse
Brenda brown	Sithara
Brenda brown	priya
Kal bent	jordan
Kal bent	manny
Kal bent	penny
Kal bent	joe

Result 23 | Read Only

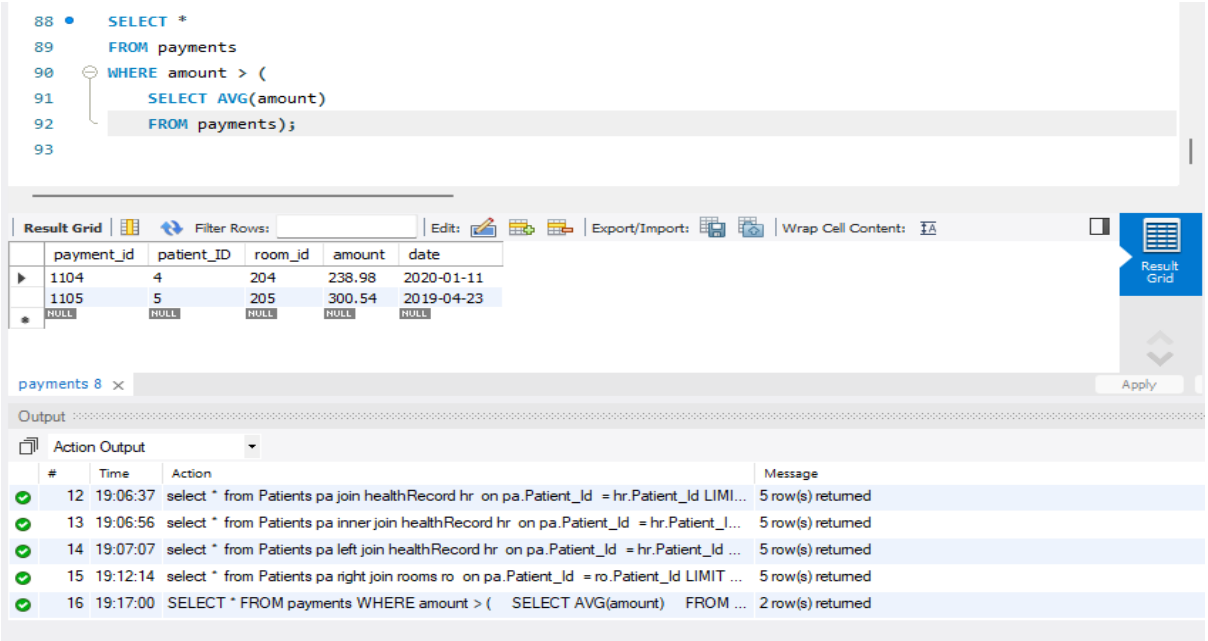
Output

Action Output

#	Time	Action	Message
69	20:33:40	SELECT phy.physician_name AS physician_name, nur.name AS allocated_nurse FR...	6 row(s) returned
70	20:33:41	SELECT phy.physician_name AS physician_name, nur.name AS allocated_nurse FR...	6 row(s) returned

Nested Queries:

1. Description: This query will only select the amount that is greater than the average amount generated by the patients and display only that row using the nested condition. Here's the screen shot showing the syntax and the result.



```
88  SELECT *
89  FROM payments
90  WHERE amount > (
91      SELECT AVG(amount)
92      FROM payments);
93
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

payment_id	patient_ID	room_id	amount	date
1104	4	204	238.98	2020-01-11
1105	5	205	300.54	2019-04-23
NULL	NULL	NULL	NULL	NULL

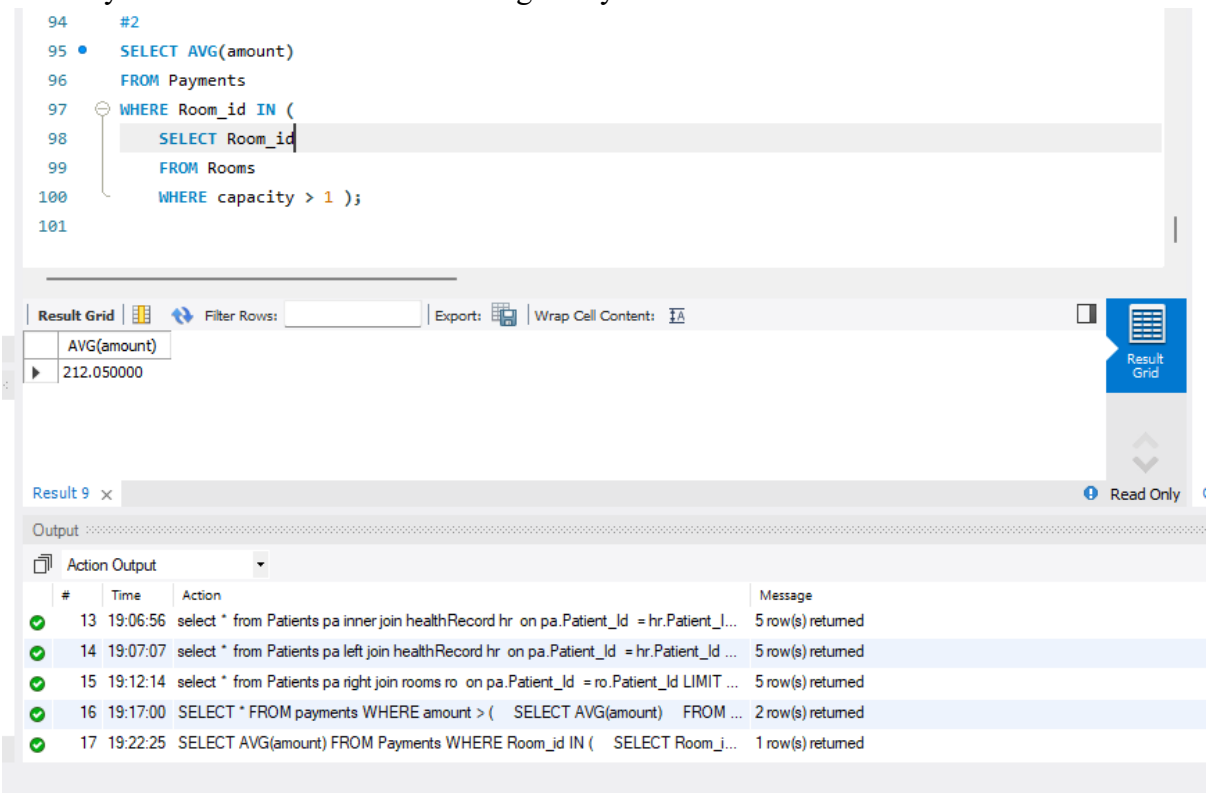
payments 8 | Apply

Output

Action Output

#	Time	Action	Message
12	19:06:37	select * from Patients pa join healthRecord hr on pa.Patient_Id = hr.Patient_Id LIMI...	5 row(s) returned
13	19:06:56	select * from Patients pa inner join healthRecord hr on pa.Patient_Id = hr.Patient_I...	5 row(s) returned
14	19:07:07	select * from Patients pa left join healthRecord hr on pa.Patient_Id = hr.Patient_Id ...	5 row(s) returned
15	19:12:14	select * from Patients pa right join rooms ro on pa.Patient_Id = ro.Patient_Id LIMIT ...	5 row(s) returned
16	19:17:00	SELECT * FROM payments WHERE amount > (SELECT AVG(amount) FROM ...	2 row(s) returned

2. Description: Here this query only shows the average amount generated for the patients who are placed in the rooms which have the capacity of more than one, in other words the average amount of bill for patients who shared their room during their stay. Here's the screen shot showing the syntax and the result.



```
94  #2
95  SELECT AVG(amount)
96  FROM Payments
97  WHERE Room_id IN (
98      SELECT Room_id
99      FROM Rooms
100     WHERE capacity > 1 );
101
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AVG(amount)
212.050000

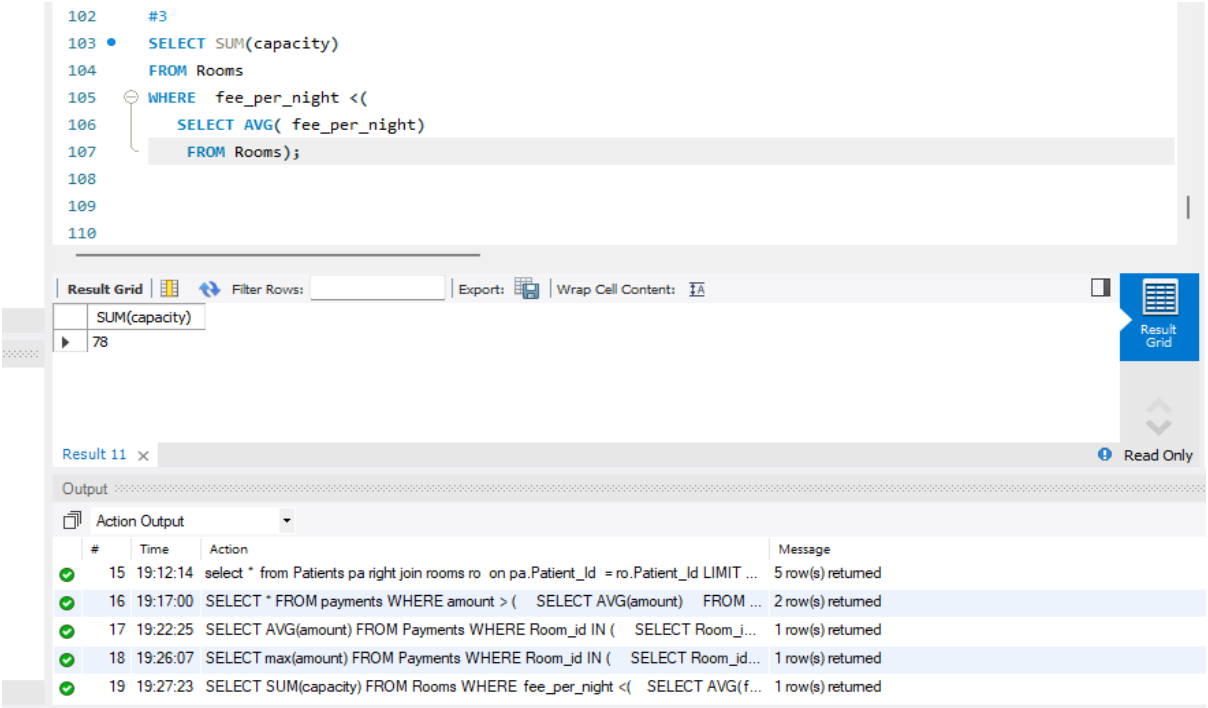
Result 9 | Read Only

Output

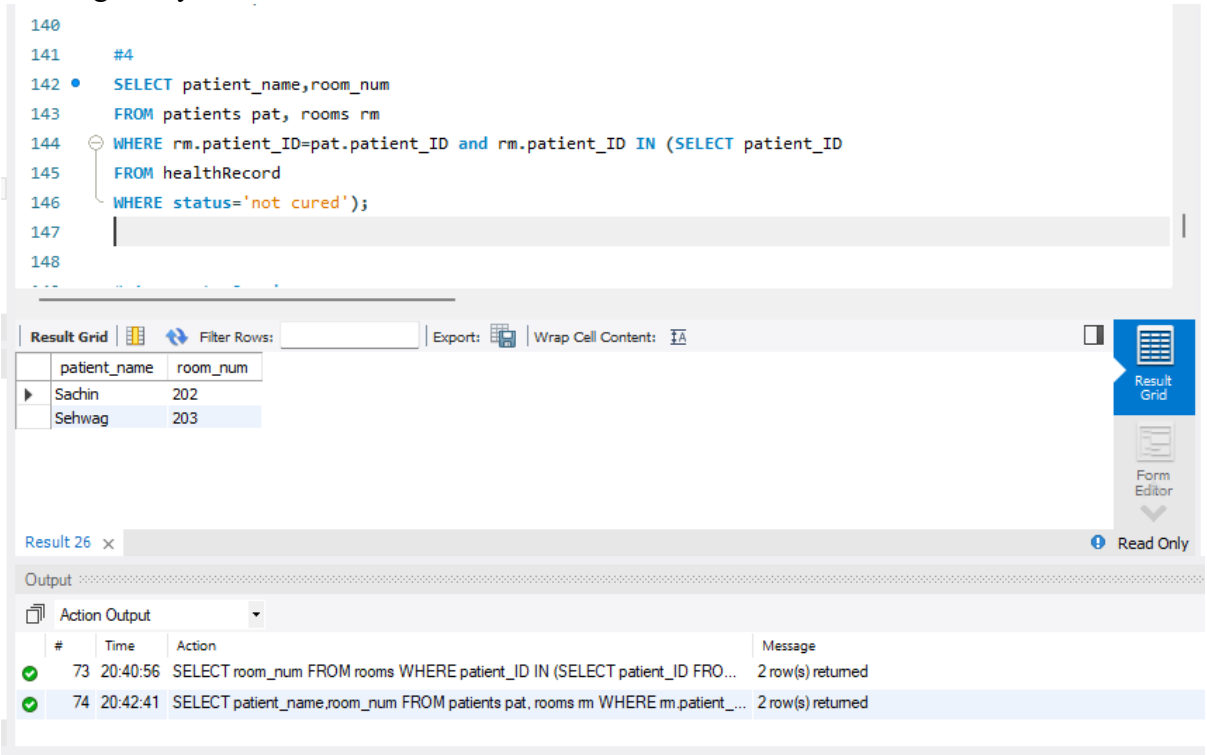
Action Output

#	Time	Action	Message
13	19:06:56	select * from Patients pa inner join healthRecord hr on pa.Patient_Id = hr.Patient_I...	5 row(s) returned
14	19:07:07	select * from Patients pa left join healthRecord hr on pa.Patient_Id = hr.Patient_Id ...	5 row(s) returned
15	19:12:14	select * from Patients pa right join rooms ro on pa.Patient_Id = ro.Patient_Id LIMIT ...	5 row(s) returned
16	19:17:00	SELECT * FROM payments WHERE amount > (SELECT AVG(amount) FROM ...	2 row(s) returned
17	19:22:25	SELECT AVG(amount) FROM Payments WHERE Room_id IN (SELECT Room_i...	1 row(s) returned

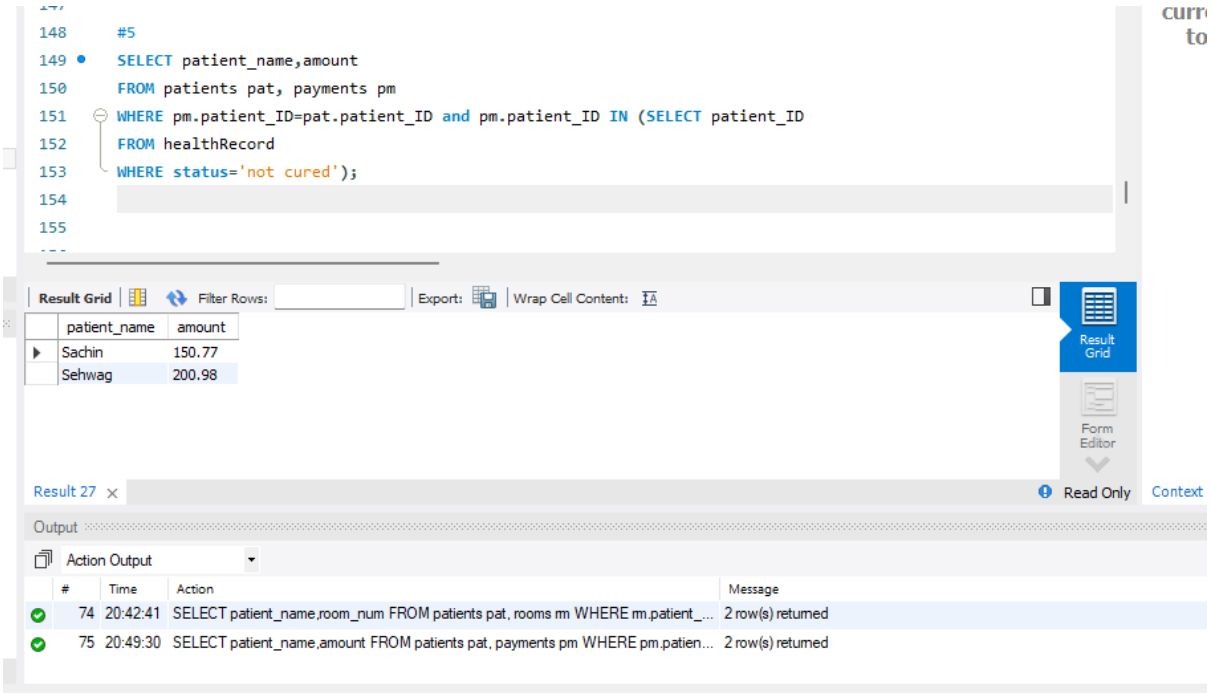
3. Description: In this query, we only select those rooms which have fees per night less than the average fees per night for all the rooms and sum them up and shows the result. Here's the screen shot showing the syntax and the result.



4. Description: This query will show the name of the patients whose disease is not yet cured, and which room they are still in the hospital. Here's the screen shot showing the syntax and the result.

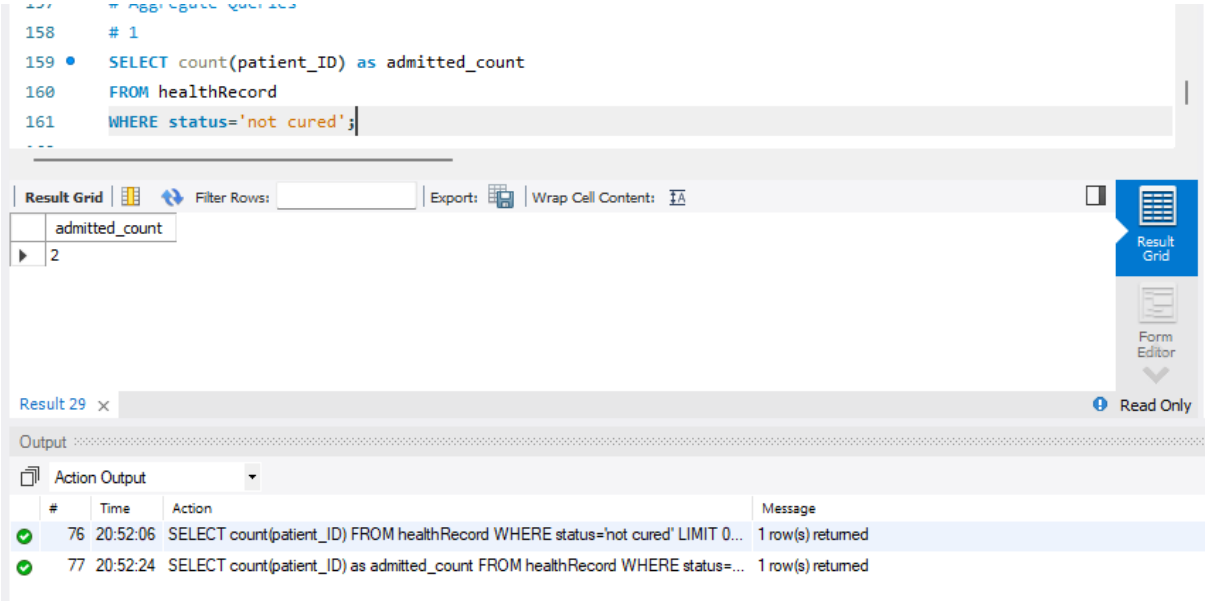


5. Description: This query will show the name of the patients whose disease is not yet cured, and the bill they have generated so far in the hospital. Here's the screen shot showing the syntax and the result.

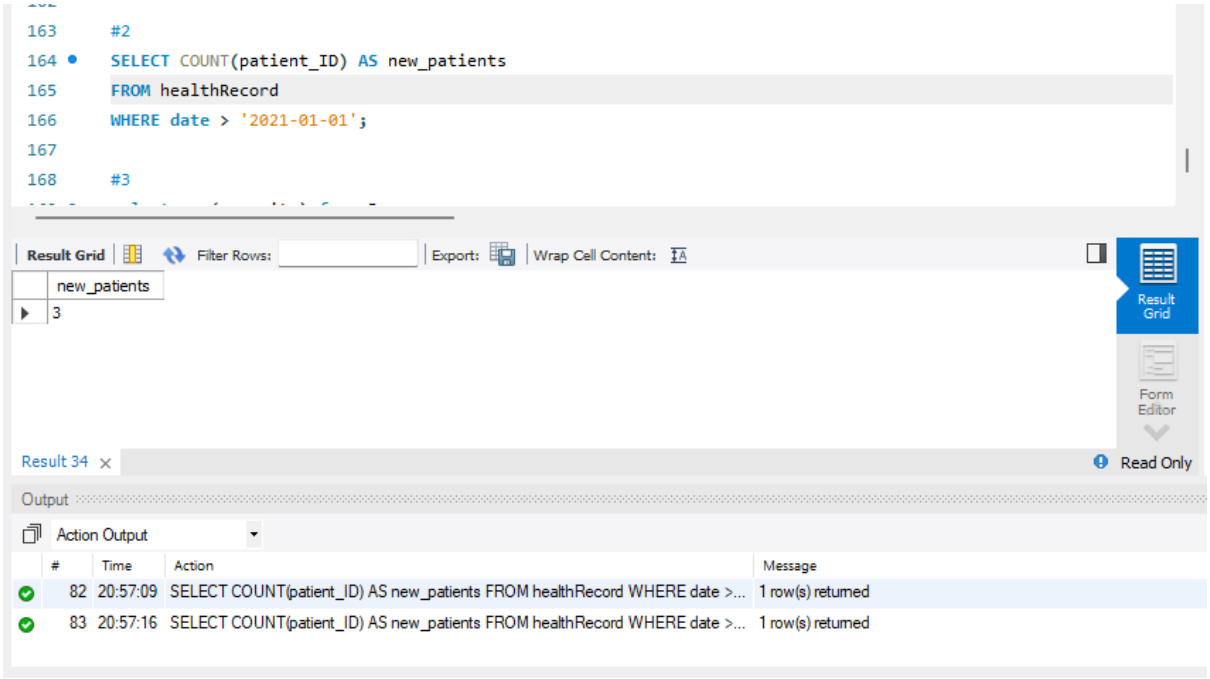


Aggregate Queries:

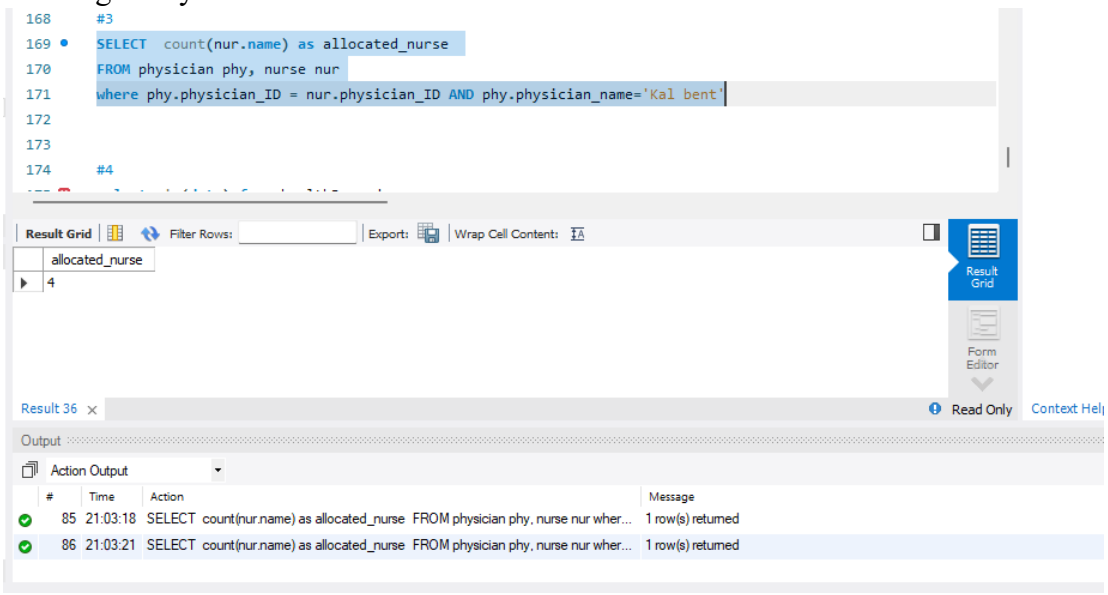
1. Description: This query calculates the count of total no. of patients who are not cured yet in the hospital that are admitted mostly. Here's the screen shot showing the syntax and the result.



2. Description: This query calculates the total no. of new patients in the hospital, that is patients joined after 1st January 2021. Here's the screen shot showing the syntax and the result.



3. Description: This query calculates the no. of nurses that are allocated to the physician name kal bent on total through various cases that is patients .Here's the screen shot showing the syntax and the result.



4. Description: This query finds the average amount paid by the patients who are staying in more than 2 member rooms and are paying less than 100 dollars per night as the bill amount. Here's the screen shot showing the syntax and the result.

```
174    #4
175 •  select avg(amount)
176    from payments, rooms
177    where rooms.patient_ID=payments.patient_ID AND capacity>2 and fee_per_night<=100;
178
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

avg(amount)
206.763333

Result 38 | Read Only

Output

Action Output

#	Time	Action	Message
88	21:06:55	select avg(amount) from payments, rooms where rooms.patient_ID=payments.patient...	1 row(s) returned
89	21:06:57	select avg(amount) from payments, rooms where rooms.patient_ID=payments.patient...	1 row(s) returned

5. Description: This query tries to age the hospital, by calculating the difference between the first and recent admitted patient approximately. Here's the screen shot showing the syntax and the result.

```
179    #5
180 •  SELECT
181      FLOOR(DATEDIFF(MAX(date), MIN(date)) / 365) AS years,
182      FLOOR((DATEDIFF(MAX(date), MIN(date)) % 365) / 30) AS months,
183      (DATEDIFF(MAX(date), MIN(date)) % 30) AS days
184    FROM healthRecord
185    WHERE patient_id IS NOT NULL;
186
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

years	months	days
3	1	11

Result 43 | Read Only

Output

Action Output

#	Time	Action	Message
93	21:10:53	SELECT FLOOR(DATEDIFF(MAX(date), MIN(date)) / 365) AS years, FLOOR((DA...	1 row(s) returned
94	21:11:05	SELECT FLOOR(DATEDIFF(MAX(date), MIN(date)) / 365) AS years, FLOOR((DA...	1 row(s) returned