Healthcare Schema Project

By Anthony "Ant" Cassetta

Table of Contents

Business rules:

1. A Facility has many buildings; A Building must belong to one and only one Facility.

2. A Building may have many Physicians practicing from them; A Physician may only practice out of one Building.

3. A Facility may accept many Insurance plans; An Insurance plan may be accepted at many facilities.

4. An Insurance plan may have many Bills; A bill may go to only one Insurance Plan.

5. A Patient may have many visits with many Physicians; A Physician may have many visits with many Patients.

6. A physician must have one or more Specialties; A Specialty may apply to one or more Physician.

7. Every Appointment creates one and only one Bill; A Bill belongs to one and only one Appointment.

Business Constraints:

1. A Physician will have a documented maximum capacity for Appointments per-day, exceptions can be made to go over that limit.

2. If a Physician visit is canceled, the first Patient on that Physicians the waitlist will scheduled a visit, then that patient will be removed from the Waitlist.

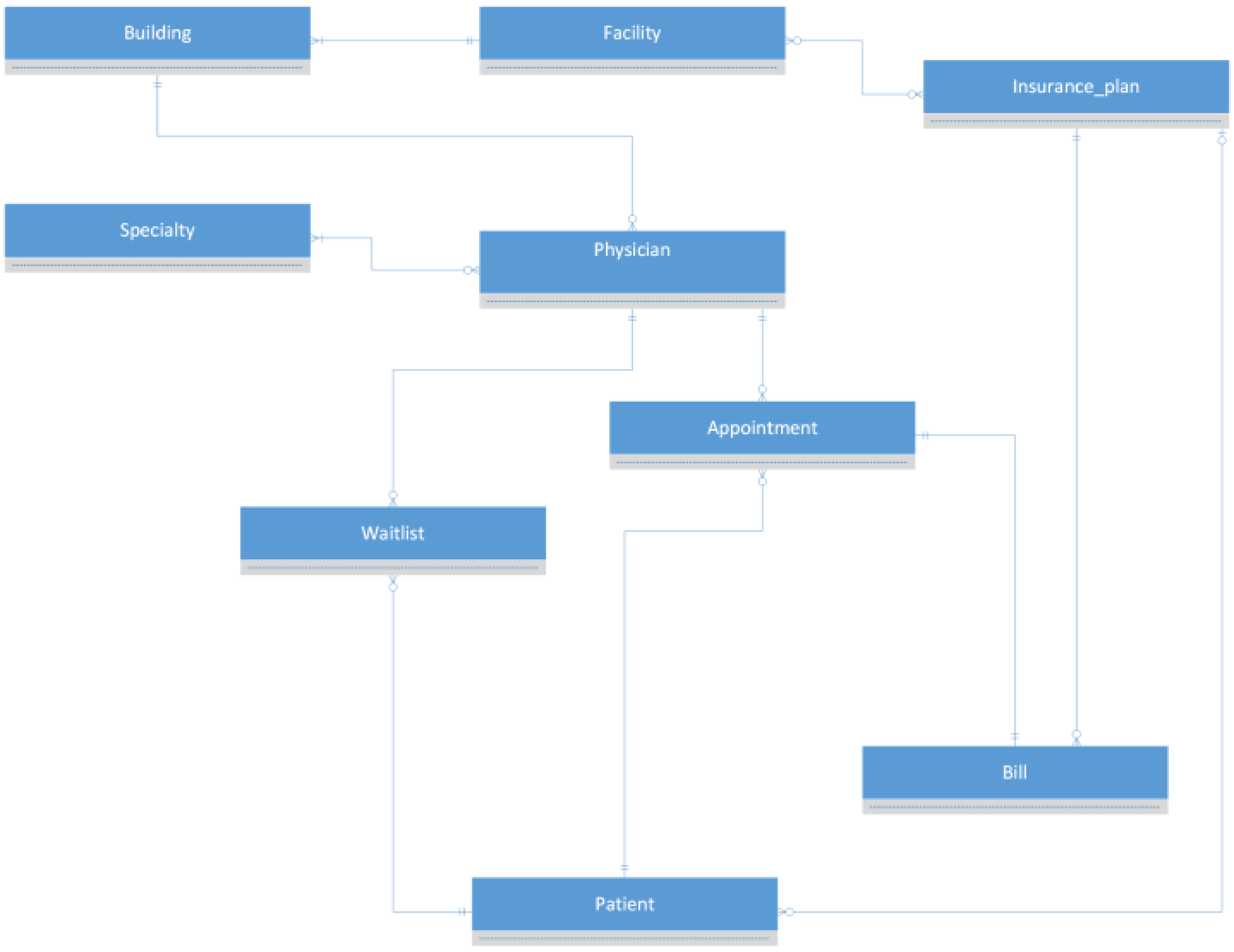
3. A Patient and Physician combination may appear on the Waitlist once; The Waitlist may have many Patient and physician combinations listed.

4. balance on Patient table will be a Trigger controlled value. When a Bill is created, the trigger will take copay_amt minus copay_paid. Then add this new value to the patient's current balance.

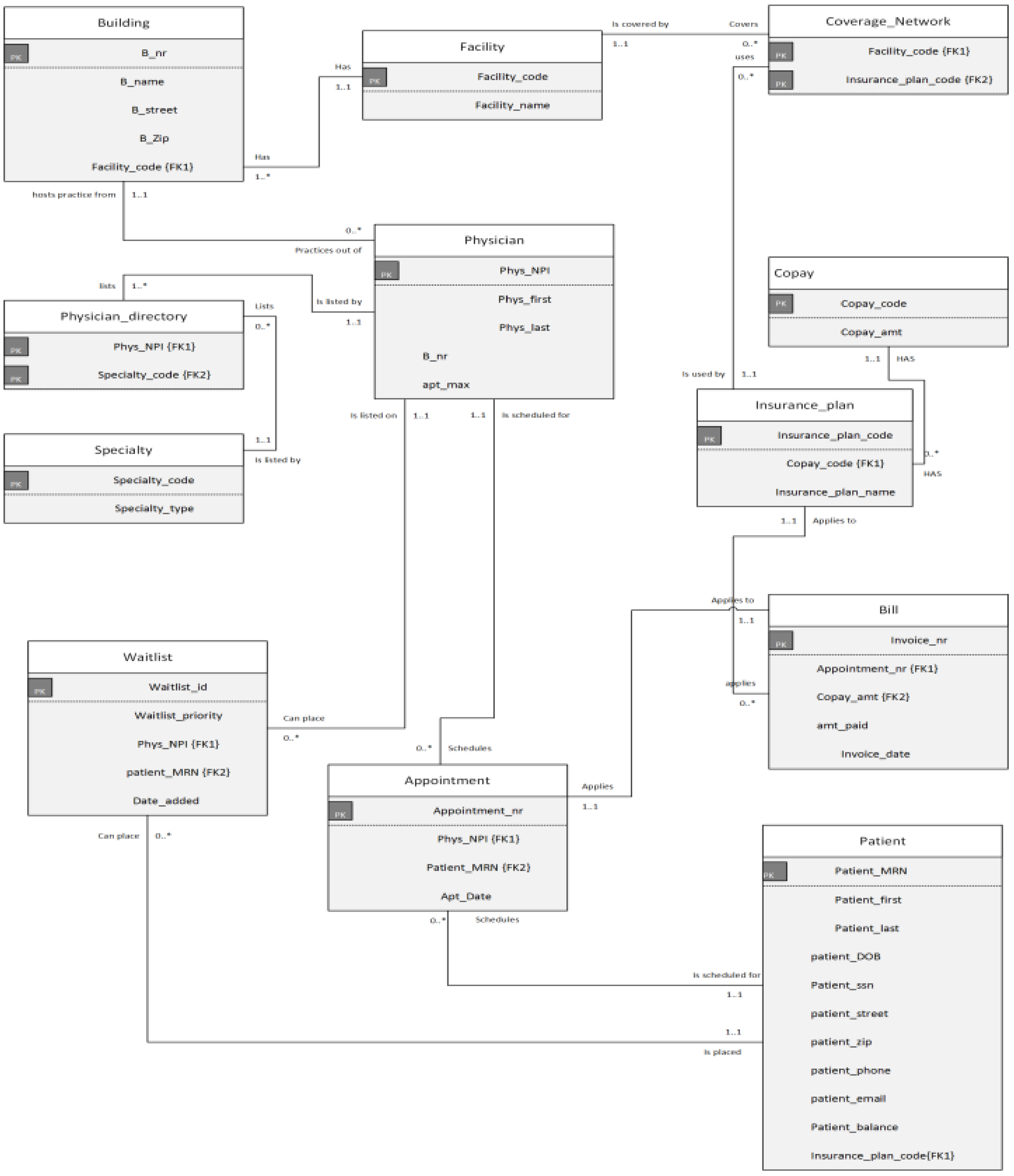
5. Combination on the Waitlist once an appointment is made the combination will then be removed from the Waitlist.

6. When an existing Appointment is canceled, it shall be replaced by the first Patient and Physician combination on the Waitlist once an appointment is made the combination will then be removed from the Waitlist.

Conceptual Entity Relationship Diagram:



Logical Entity Relationship Diagram:



Use Cases:

1. Health center management requests the first and last names of all physicians that work in the "Agnes" or "Palladius" buildings. Write a single query that retrieves this information for management.

2. Auditors request the names of all patients that currently have insurance, as well as the name of their current insurance plan. Write a single query that retrieves this information for the auditors.

3. A patient phones a physician's administrative assistant asking for an appointment, and the administrative assistant decides to add the patient to the waiting list so that the patient will be the next one to be scheduled for a canceled appointment. Develop a parameterized stored procedure that accomplishes this, then invoke the stored procedure for a patient of your choosing.

4. A receptionist requests the first and last names of all physicians that a specific patient has *never* visited. Write a single query that retrieves this information for the receptionist.

5. A patient visits a physician but fails to render copayment at the time of visit, necessitating that $30 be added to the balance of that patient. Develop a parameterized stored procedure that accomplishes this, then invoke the stored procedure for a patient of your choosing.

6. A patient cancels their insurance plan, and the hospital staff must update the system to reflect this cancelation. Develop a parameterized stored procedure that accomplishes this, then invoke the stored procedure for a patient of your choosing.

7. A receptionist needs to know the names of all physicians that are booked for the next two days. Being booked means the physicians have no available appointments for either day. Write a single query that retrieves this information for the receptionist.

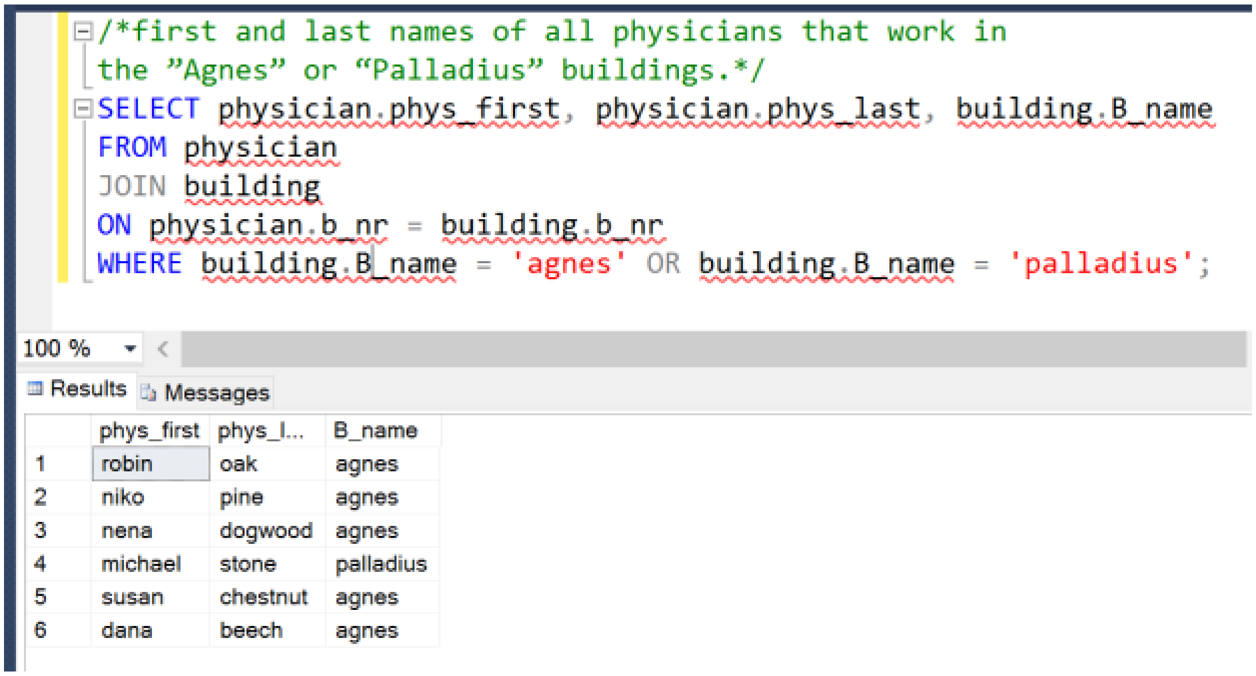
8. Health center management requests the insurance plan with the most patient enrollees, and for that plan, its name, required copayment amount, and the number of patient enrollees. Write a single query that retrieves this information for the management.

9. Health center management requests the names of all patients, and for each patient, the names of the physicians that they visited more than once, along with the number of visits to each of these physicians. If a patient has not visited any physicians, or did not visit any physicians more than once, management does not want to see them in the list. Write a single query that retrieves this information for the management.

10. Health center management requests the names of all physicians, and for each physician, the number of different patients that visited the physician. Management would like to this to be ordered from the highest number of different visitors to the lowest number. Multiple visits to the same physician by the same patient only count as one unique visit for purposes of this request. Management is interested in the number of different visitors, but not whether the

same patient visited the same physician multiple times. Write a single query that retrieves this information for the management.

MS SQL Query Executions Demonstrating Use Cases:



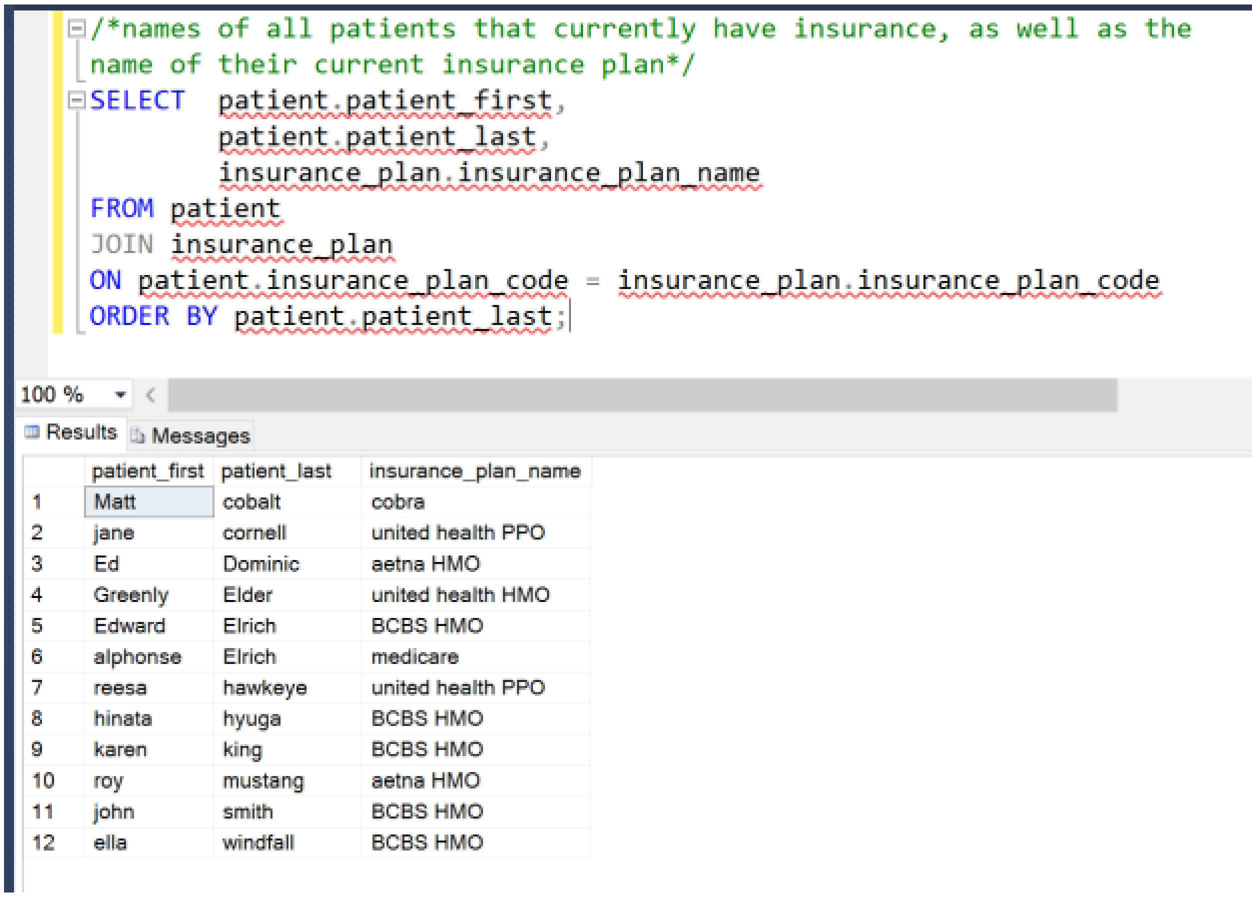
```
/*first and last names of all physicians that work in
the "Agnes" or "Palladius" buildings.*/
SELECT physician.phys_first, physician.phys_last, building.B_name
FROM physician
JOIN building
ON physician.b_nr = building.b_nr
WHERE building.B_name = 'agnes' OR building.B_name = 'palladius';
```

100 %

Results | Messages

	phys_first	phys_l...	B_name
1	robin	oak	agnes
2	niko	pine	agnes
3	nena	dogwood	agnes
4	michael	stone	palladius
5	susan	chestnut	agnes
6	dana	beech	agnes

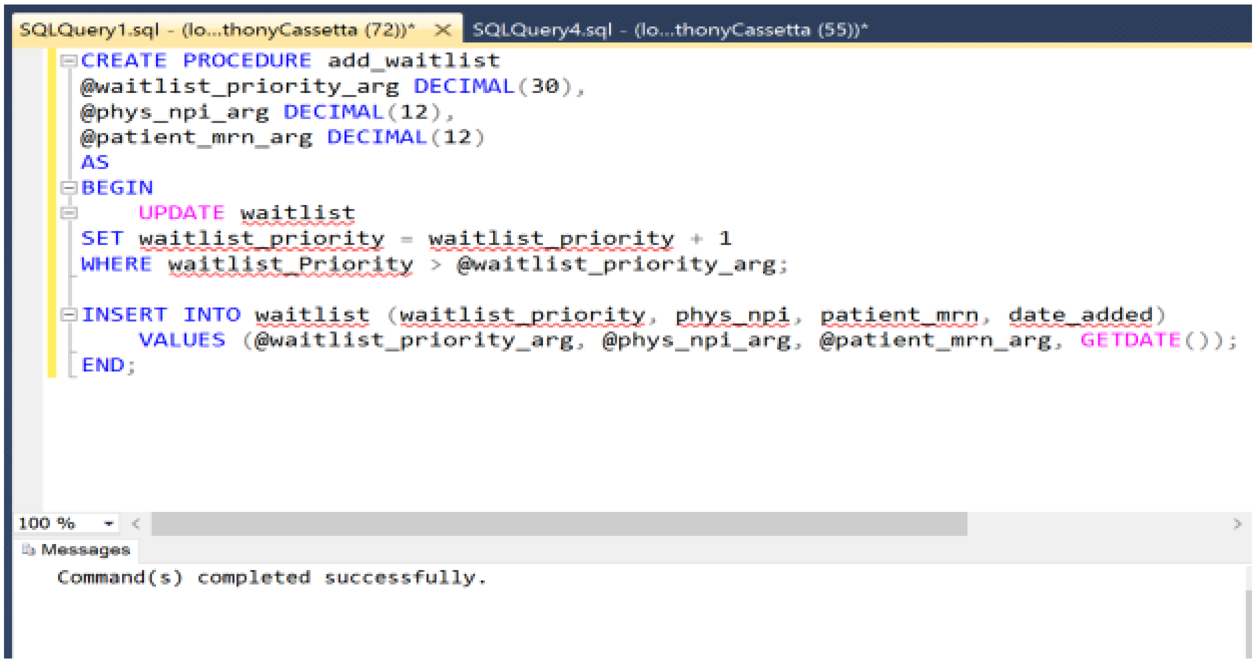
Figure 1: UC-1



```sql
/*names of all patients that currently have insurance, as well as the
name of their current insurance plan*/
SELECT  patient.patient_first,
        patient.patient_last,
        insurance_plan.insurance_plan_name
FROM patient
JOIN insurance_plan
ON patient.insurance_plan_code = insurance_plan.insurance_plan_code
ORDER BY patient.patient_last;
```

	patient_first	patient_last	insurance_plan_name
1	Matt	cobalt	cobra
2	jane	cornell	united health PPO
3	Ed	Dominic	aetna HMO
4	Greenly	Elder	united health HMO
5	Edward	Elrich	BCBS HMO
6	alphonse	Elrich	medicare
7	reesa	hawkeye	united health PPO
8	hinata	hyuga	BCBS HMO
9	karen	king	BCBS HMO
10	roy	mustang	aetna HMO
11	john	smith	BCBS HMO
12	ella	windfall	BCBS HMO

Figure 2: UC-2

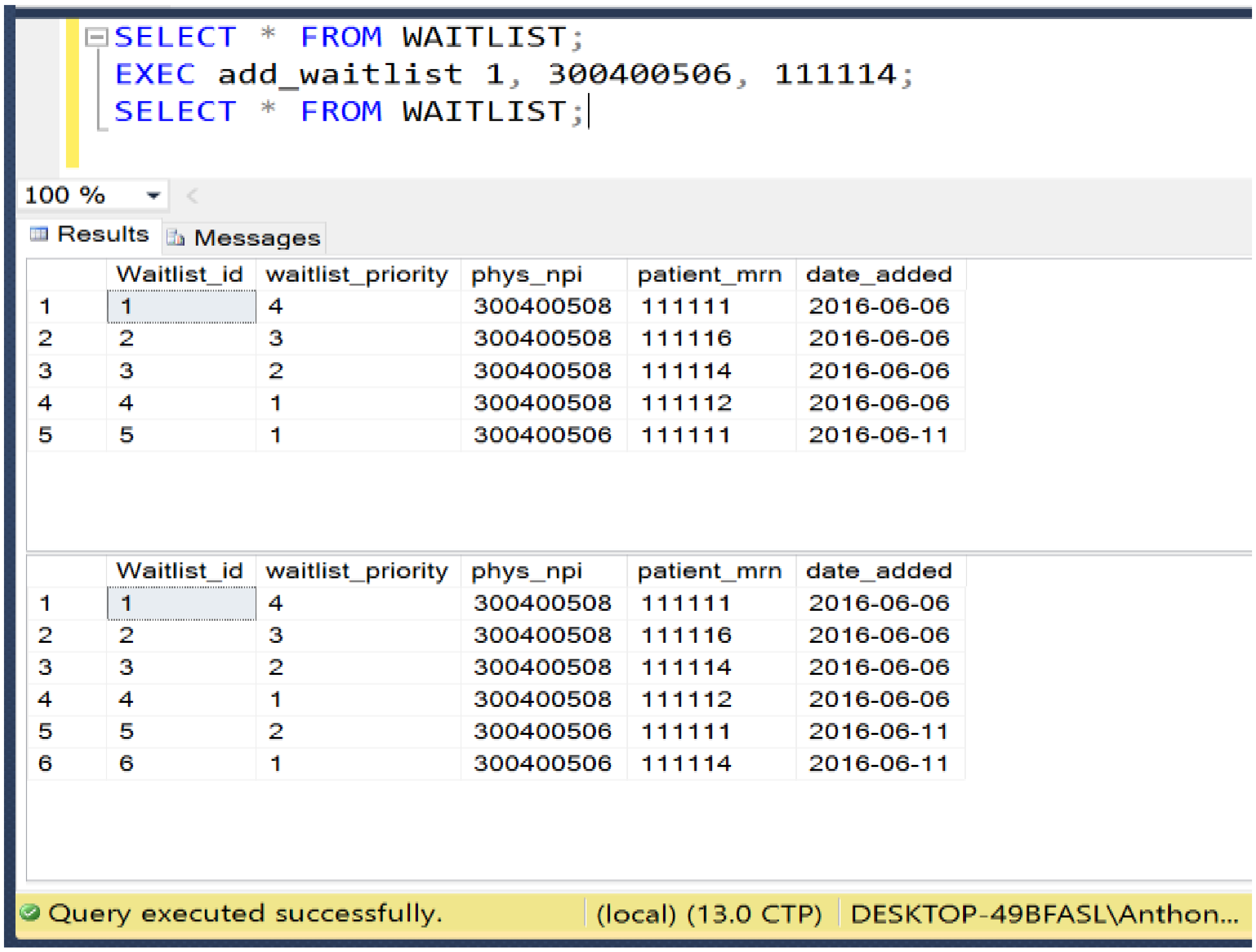


SQLQuery1.sql - (lo...thonyCassetta (72))*
SQLQuery4.sql - (lo...thonyCassetta (55))*

```sql
CREATE PROCEDURE add_waitlist
@waitlist_priority_arg DECIMAL(30),
@phys_npi_arg DECIMAL(12),
@patient_mrn_arg DECIMAL(12)
AS
BEGIN
    UPDATE waitlist
SET waitlist_priority = waitlist_priority + 1
WHERE waitlist_Priority > @waitlist_priority_arg;

INSERT INTO waitlist (waitlist_priority, phys_npi, patient_mrn, date_added)
    VALUES (@waitlist_priority_arg, @phys_npi_arg, @patient_mrn_arg, GETDATE());
END;
```

```
Command(s) completed successfully.
```

Figure 3: UC-3.1



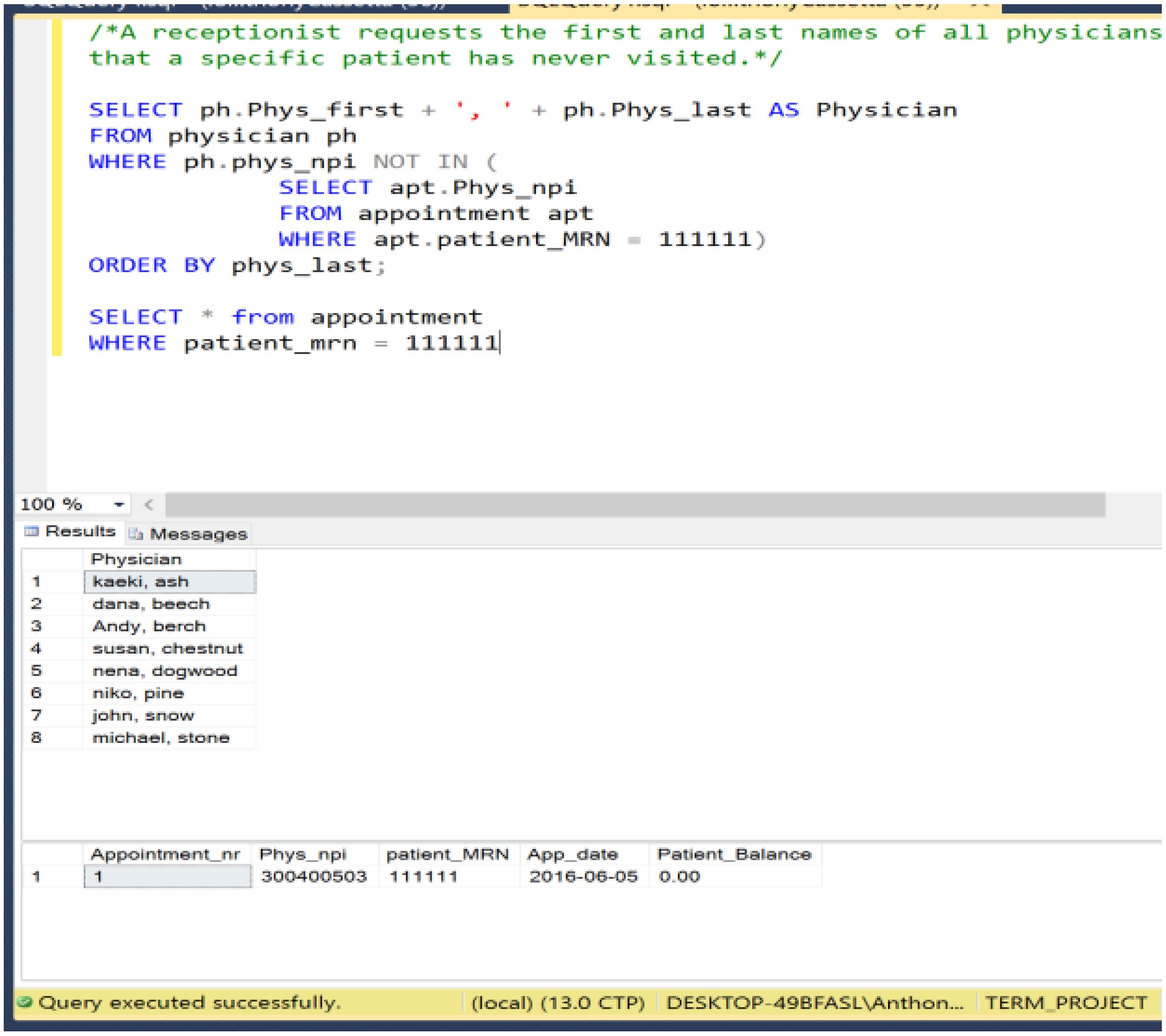
```
SELECT * FROM WAITLIST;
EXEC add_waitlist 1, 300400506, 111114;
SELECT * FROM WAITLIST;
```

	Waitlist_id	waitlist_priority	phys_npi	patient_mrn	date_added
1	1	4	300400508	111111	2016-06-06
2	2	3	300400508	111116	2016-06-06
3	3	2	300400508	111114	2016-06-06
4	4	1	300400508	111112	2016-06-06
5	5	1	300400506	111111	2016-06-11

	Waitlist_id	waitlist_priority	phys_npi	patient_mrn	date_added
1	1	4	300400508	111111	2016-06-06
2	2	3	300400508	111116	2016-06-06
3	3	2	300400508	111114	2016-06-06
4	4	1	300400508	111112	2016-06-06
5	5	2	300400506	111111	2016-06-11
6	6	1	300400506	111114	2016-06-11

```
Query executed successfully.    (local) (13.0 CTP)    DESKTOP-49BFASL\Anthon...
```

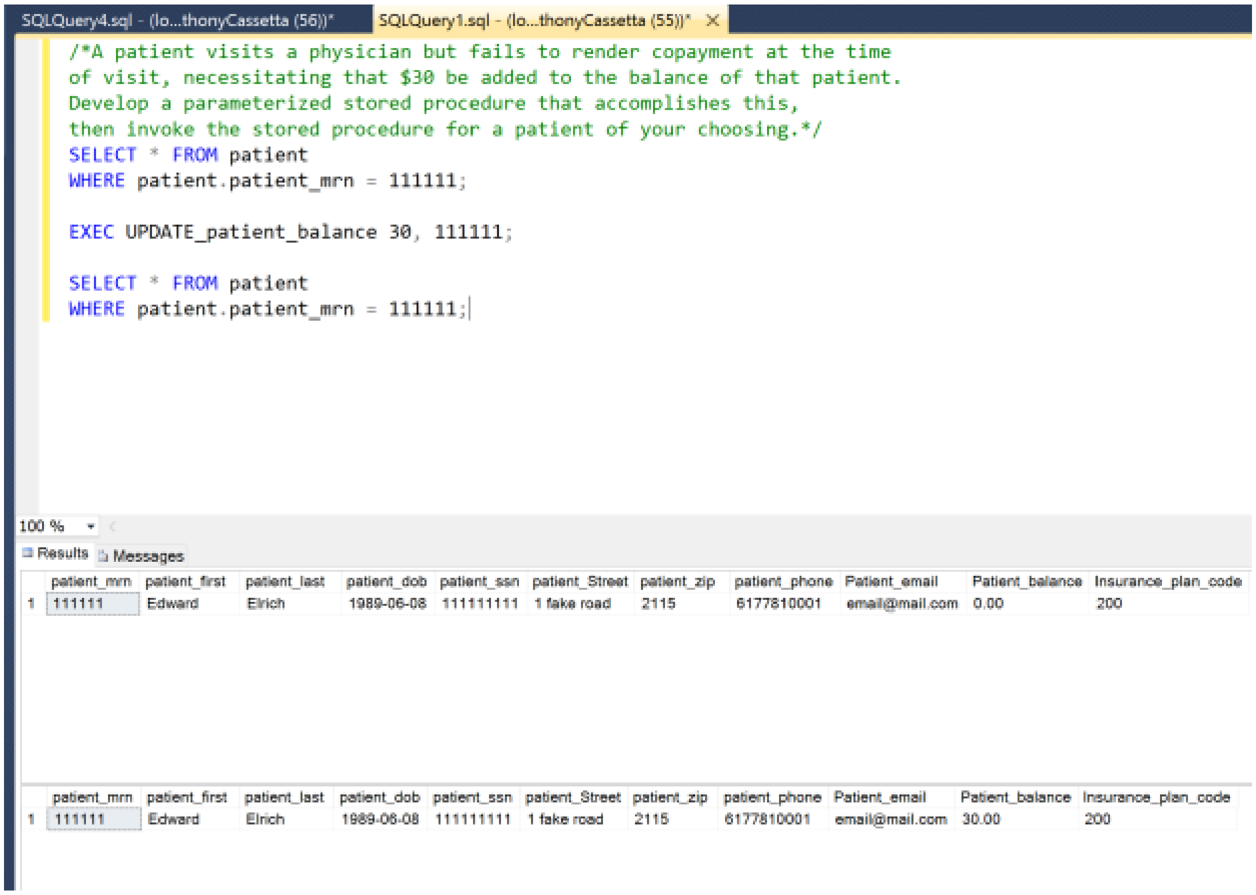
Figure 4: UC-3.2



```
/*A receptionist requests the first and last names of all physicians
that a specific patient has never visited.*/

SELECT ph.Phys_first + ', ' + ph.Phys_last AS Physician
FROM physician ph
WHERE ph.phys_npi NOT IN (
            SELECT apt.Phys_npi
            FROM appointment apt
            WHERE apt.patient_MRN = 111111)
ORDER BY phys_last;

SELECT * from appointment
WHERE patient_mrn = 111111
```

```
100 %
Results  Messages

     Physician
1    kaeki, ash
2    dana, beech
3    Andy, berch
4    susan, chestnut
5    nena, dogwood
6    niko, pine
7    john, snow
8    michael, stone

     Appointment_nr  Phys_npi   patient_MRN  App_date    Patient_Balance
1    1               300400503  111111       2016-06-05  0.00

Query executed successfully.    (local) (13.0 CTP)   DESKTOP-49BFASL\Anthon...   TERM_PROJECT
```

Figure 5: UC-4

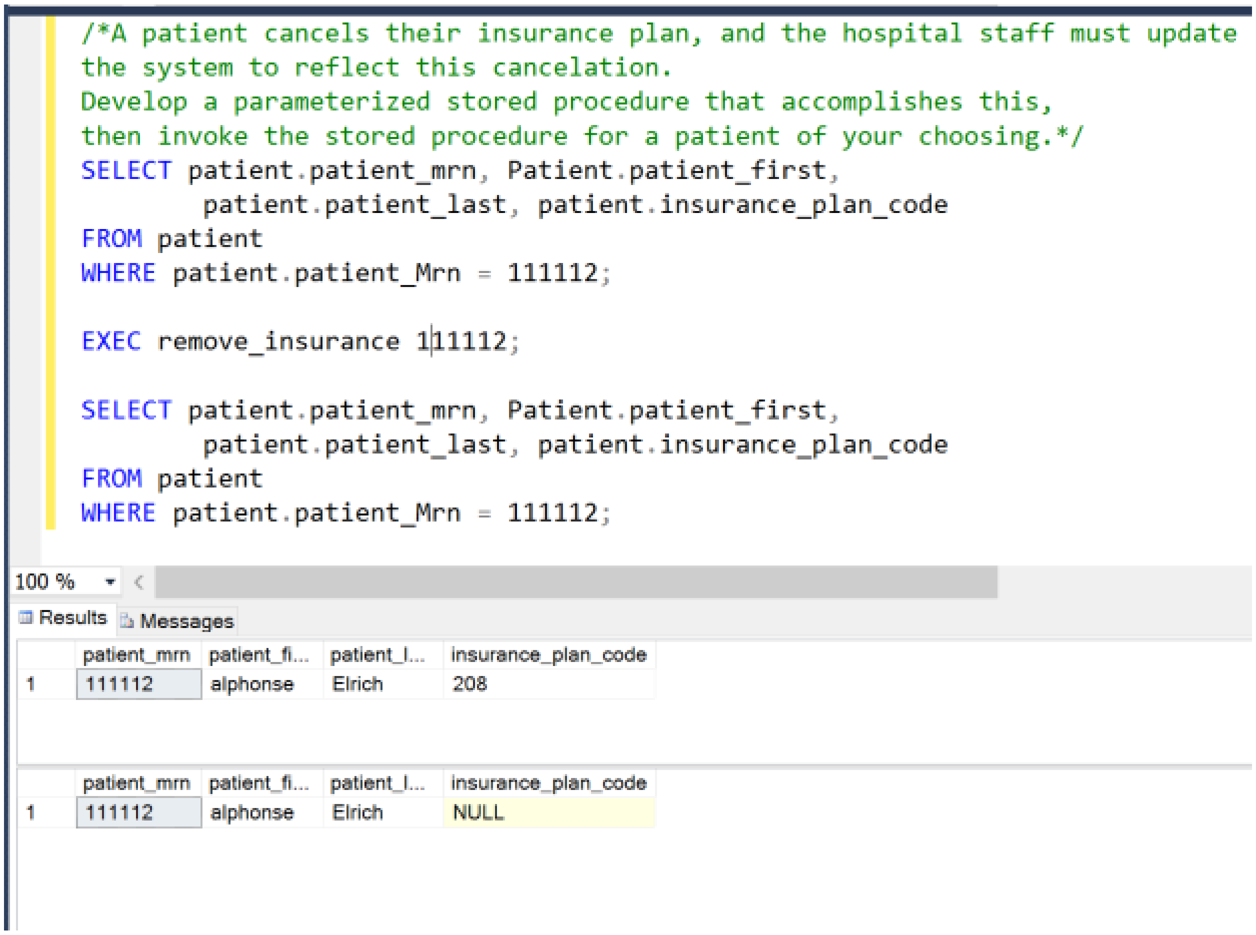

SQLQuery4.sql - (lo...thonyCassetta (56))*
SQLQuery1.sql - (lo...thonyCassetta (55))*
100 %
Results
Messages

```
/*A patient visits a physician but fails to render copayment at the time
of visit, necessitating that $30 be added to the balance of that patient.
Develop a parameterized stored procedure that accomplishes this,
then invoke the stored procedure for a patient of your choosing.*/
SELECT * FROM patient
WHERE patient.patient_mrn = 111111;

EXEC UPDATE_patient_balance 30, 111111;

SELECT * FROM patient
WHERE patient.patient_mrn = 111111;
```

	patient_mrn	patient_first	patient_last	patient_dob	patient_ssn	patient_Street	patient_zip	patient_phone	Patient_email	Patient_balance	Insurance_plan_code
1	111111	Edward	Elrich	1989-06-08	111111111	1 fake road	2115	6177810001	email@mail.com	0.00	200

	patient_mrn	patient_first	patient_last	patient_dob	patient_ssn	patient_Street	patient_zip	patient_phone	Patient_email	Patient_balance	Insurance_plan_code
1	111111	Edward	Elrich	1989-06-08	111111111	1 fake road	2115	6177810001	email@mail.com	30.00	200

Figure 6: UC-5

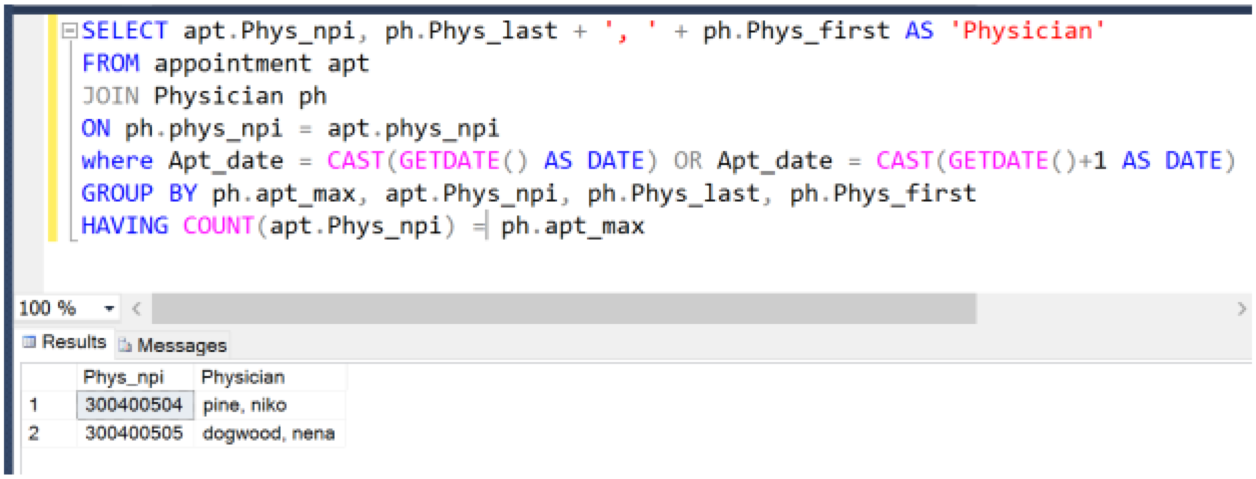


```
/*A patient cancels their insurance plan, and the hospital staff must update
the system to reflect this cancelation.
Develop a parameterized stored procedure that accomplishes this,
then invoke the stored procedure for a patient of your choosing.*/
SELECT patient.patient_mrn, Patient.patient_first,
        patient.patient_last, patient.insurance_plan_code
FROM patient
WHERE patient.patient_Mrn = 111112;

EXEC remove_insurance 111112;

SELECT patient.patient_mrn, Patient.patient_first,
        patient.patient_last, patient.insurance_plan_code
FROM patient
WHERE patient.patient_Mrn = 111112;
```

100 % Results Messages

```
   patient_mrn  patient_fi...  patient_l...  insurance_plan_code
1  111112       alphonse       Elrich        208

   patient_mrn  patient_fi...  patient_l...  insurance_plan_code
1  111112       alphonse       Elrich        NULL
```

Figure 7: UC-6

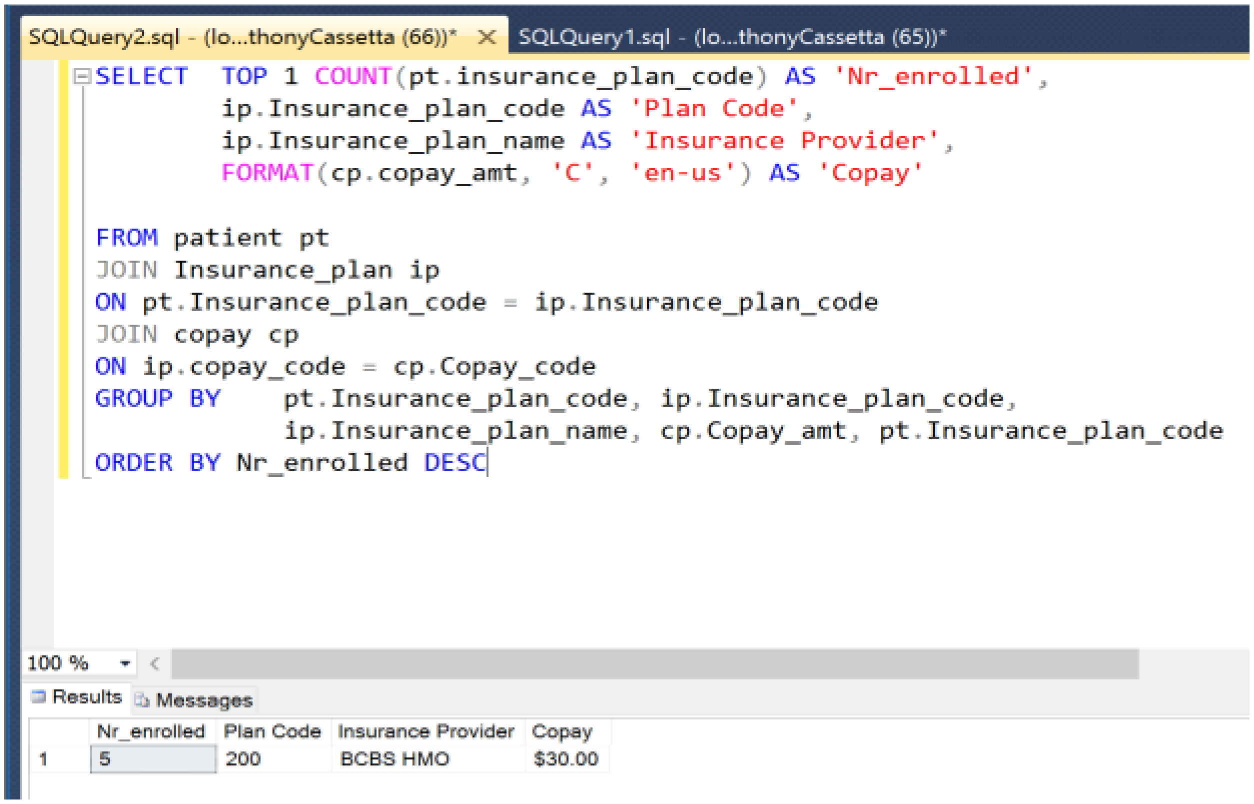


```
SELECT apt.Phys_npi, ph.Phys_last + ', ' + ph.Phys_first AS 'Physician'
FROM appointment apt
JOIN Physician ph
ON ph.phys_npi = apt.phys_npi
where Apt_date = CAST(GETDATE() AS DATE) OR Apt_date = CAST(GETDATE()+1 AS DATE)
GROUP BY ph.apt_max, apt.Phys_npi, ph.Phys_last, ph.Phys_first
HAVING COUNT(apt.Phys_npi) = ph.apt_max
```

100 % Results Messages

```
   Phys_npi   Physician
1  300400504  pine, niko
2  300400505  dogwood, nena
```

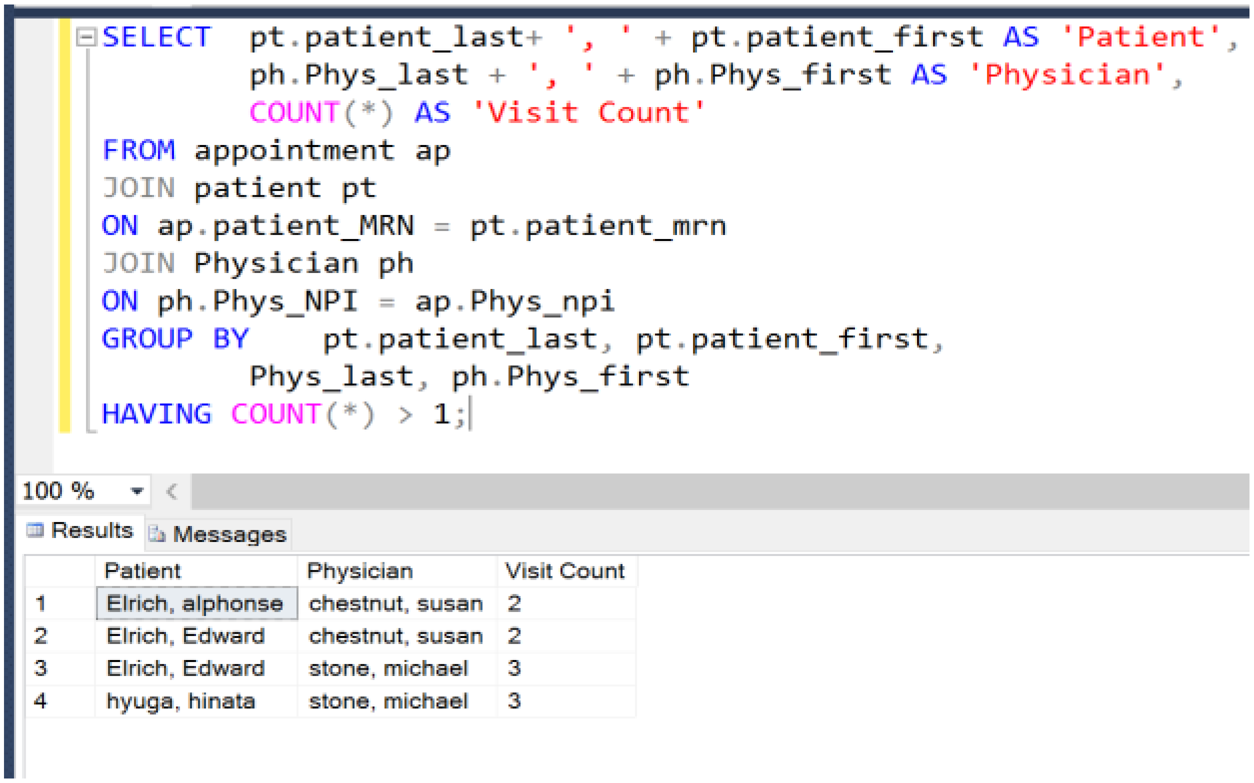
Figure 8: UC-7


SQLQuery2.sql - (lo...thonyCassetta (66))*
SQLQuery1.sql - (lo...thonyCassetta (65))*
100 %
Results
Messages

```sql
SELECT   TOP 1 COUNT(pt.insurance_plan_code) AS 'Nr_enrolled',
         ip.Insurance_plan_code AS 'Plan Code',
         ip.Insurance_plan_name AS 'Insurance Provider',
         FORMAT(cp.copay_amt, 'C', 'en-us') AS 'Copay'

FROM patient pt
JOIN Insurance_plan ip
ON pt.Insurance_plan_code = ip.Insurance_plan_code
JOIN copay cp
ON ip.copay_code = cp.Copay_code
GROUP BY     pt.Insurance_plan_code, ip.Insurance_plan_code,
             ip.Insurance_plan_name, cp.Copay_amt, pt.Insurance_plan_code
ORDER BY Nr_enrolled DESC
```

```
   Nr_enrolled  Plan Code  Insurance Provider  Copay
1  5            200        BCBS HMO            $30.00
```

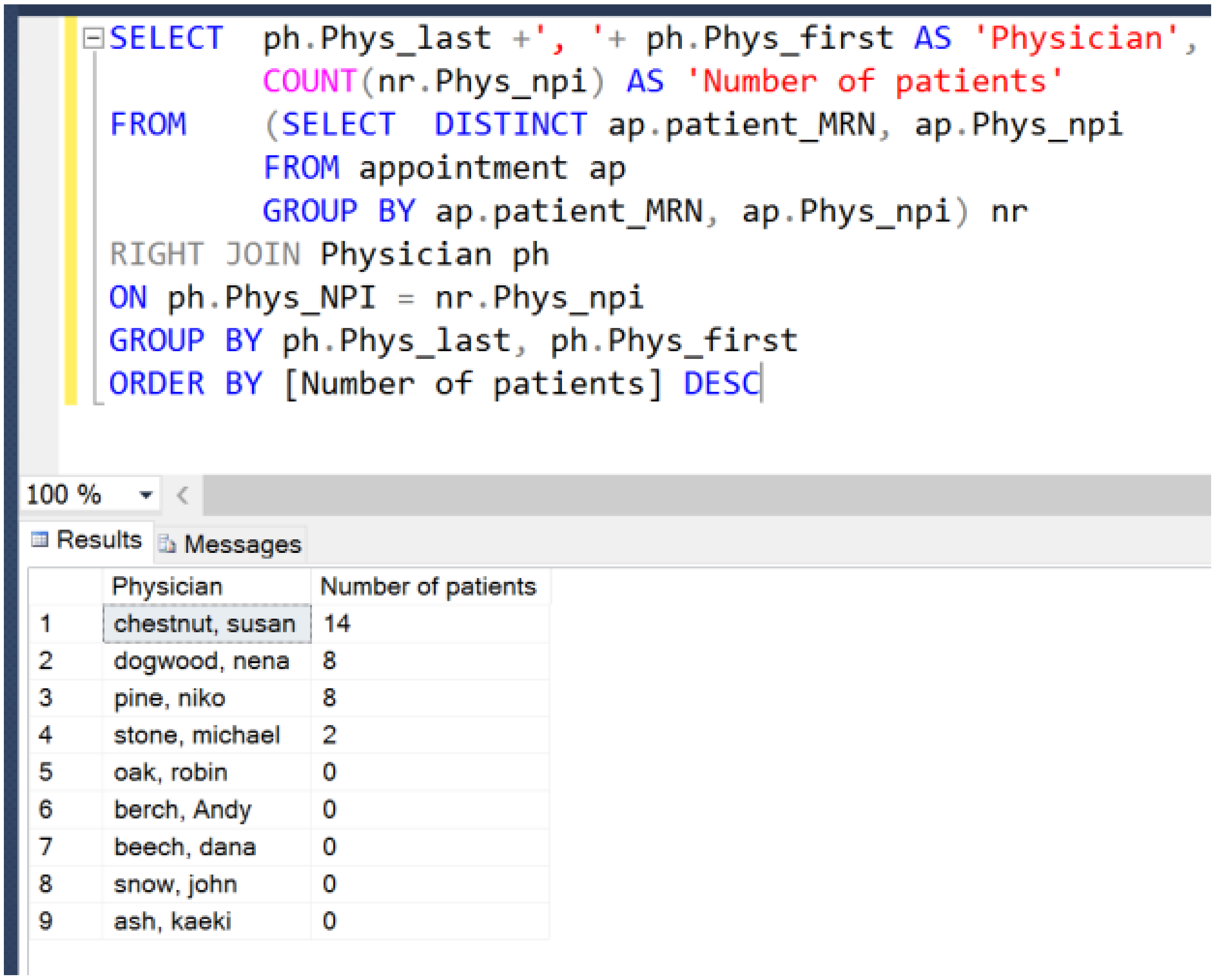
Figure 9: UC-8



```
SELECT  pt.patient_last+ ', ' + pt.patient_first AS 'Patient',
        ph.Phys_last + ', ' + ph.Phys_first AS 'Physician',
        COUNT(*) AS 'Visit Count'
FROM appointment ap
JOIN patient pt
ON ap.patient_MRN = pt.patient_mrn
JOIN Physician ph
ON ph.Phys_NPI = ap.Phys_npi
GROUP BY    pt.patient_last, pt.patient_first,
        Phys_last, ph.Phys_first
HAVING COUNT(*) > 1;
```

```
   Patient           Physician          Visit Count
1  Elrich, alphonse  chestnut, susan    2
2  Elrich, Edward    chestnut, susan    2
3  Elrich, Edward    stone, michael     3
4  hyuga, hinata     stone, michael     3
```

Figure 10: UC-9



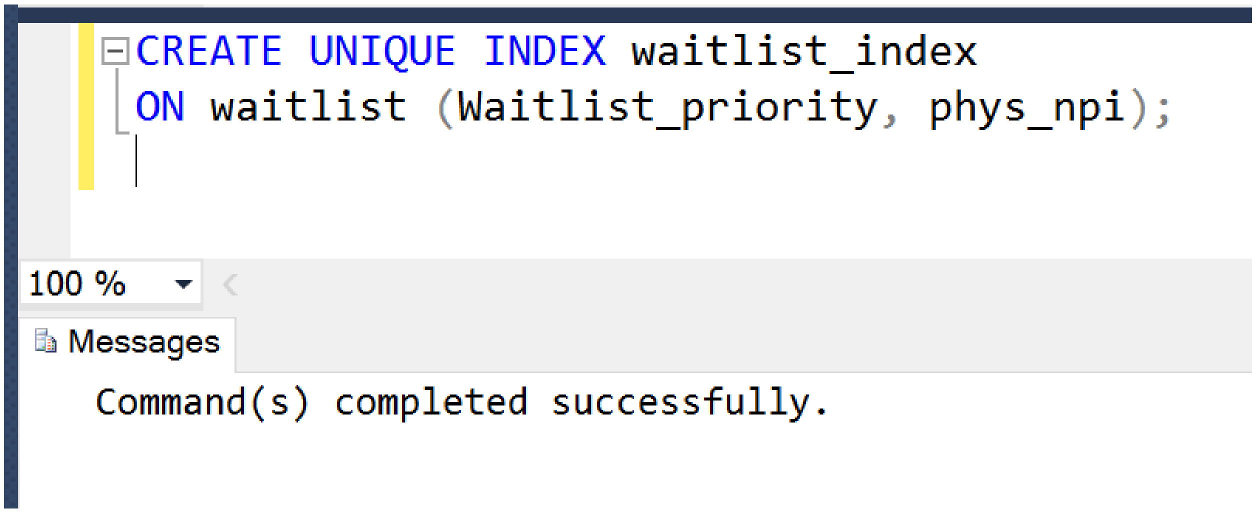
```
SELECT   ph.Phys_last +', '+ ph.Phys_first AS 'Physician',
         COUNT(nr.Phys_npi) AS 'Number of patients'
FROM     (SELECT  DISTINCT ap.patient_MRN, ap.Phys_npi
         FROM appointment ap
         GROUP BY ap.patient_MRN, ap.Phys_npi) nr
RIGHT JOIN Physician ph
ON ph.Phys_NPI = nr.Phys_npi
GROUP BY ph.Phys_last, ph.Phys_first
ORDER BY [Number of patients] DESC
```

	Physician	Number of patients
1	chestnut, susan	14
2	dogwood, nena	8
3	pine, niko	8
4	stone, michael	2
5	oak, robin	0
6	berch, Andy	0
7	beech, dana	0
8	snow, john	0
9	ash, kaeki	0

Figure 11: UC-10

Screenshots illustrating the creation of two indexes, along with explanations:

Index 1.



```
CREATE UNIQUE INDEX waitlist_index
ON waitlist (Waitlist_priority, phys_npi);
```

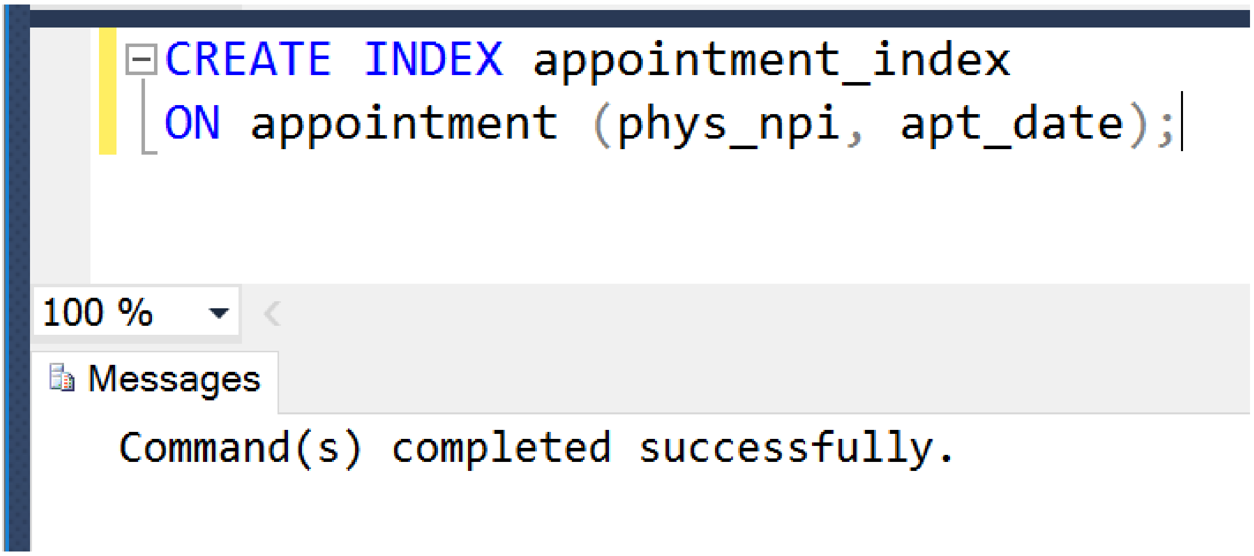
100 %

Messages

```
Command(s) completed successfully.
```

With the assumption that this schema will be used at a much larger scale than this project demonstrates, this index would serve to efficiently allow searching through the waitlist based on the Physician a requestor would need information from. This index also enforces a uniqueness constraint that will prevent any physician from having multiple patients with the same priority level. It is unnecessary to create an index on the waitlist table in regard to patients as the business constraints state that a patient should only be listed at most once, for each physician.

Index 2.



```
CREATE INDEX appointment_index
ON appointment (phys_npi, apt_date);
```

100 %

Messages

```
Command(s) completed successfully.
```

With the same assumptions about scale as above, an index on the appointment table would expedite searching for available visits, or searching for existing appointments by date and provider ignorer to update or delete that appointment.

A similar index for patient and appointment date would also benefit the table, as Searching by physician or patient are the most likely queries.

Notes:
Primary key naming convention “pk_<table name>”
Foreign key naming convention “<host table name>_fk”
Priority scale highest 1, priority lowers as number gets greater.
Were this a production instillation I would have built a trigger to create a visit invoice in the Bill table, which would automatically fire when an insert was performed on appointment.