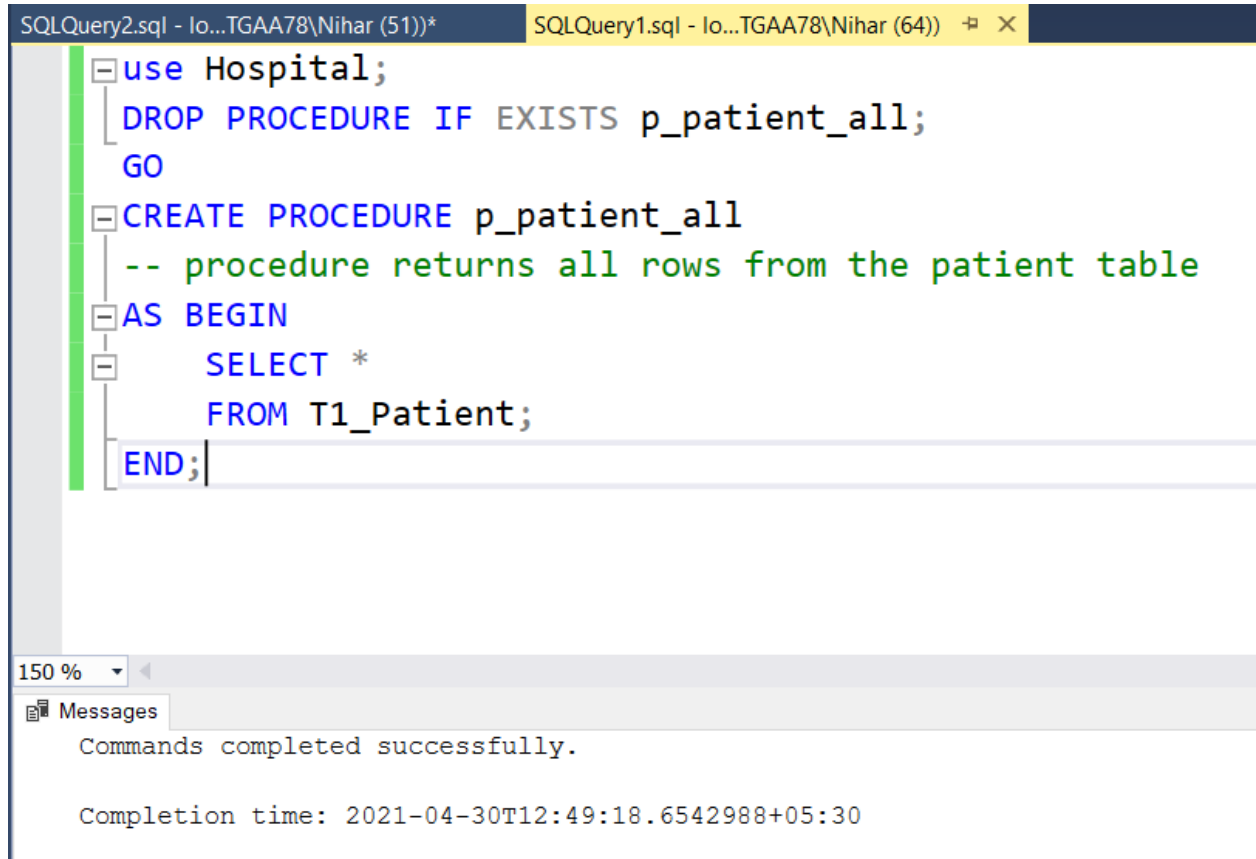


DBMS LAB 6

1. Write two stored Procedures relevant to your database.



```
SQLQuery2.sql - lo...TGAA78\Nihar (51))*  SQLQuery1.sql - lo...TGAA78\Nihar (64))  +  X

use Hospital;
DROP PROCEDURE IF EXISTS p_patient_all;
GO
CREATE PROCEDURE p_patient_all
-- procedure returns all rows from the patient table
AS BEGIN
    SELECT *
    FROM T1_Patient;
END;
```

150 %

Messages

Commands completed successfully.

Completion time: 2021-04-30T12:49:18.6542988+05:30

SQLQuery2.sql - lo...TGAA78\Nihar (51))*

SQLQuery1.sql - lo...TGAA78\Nihar (64))

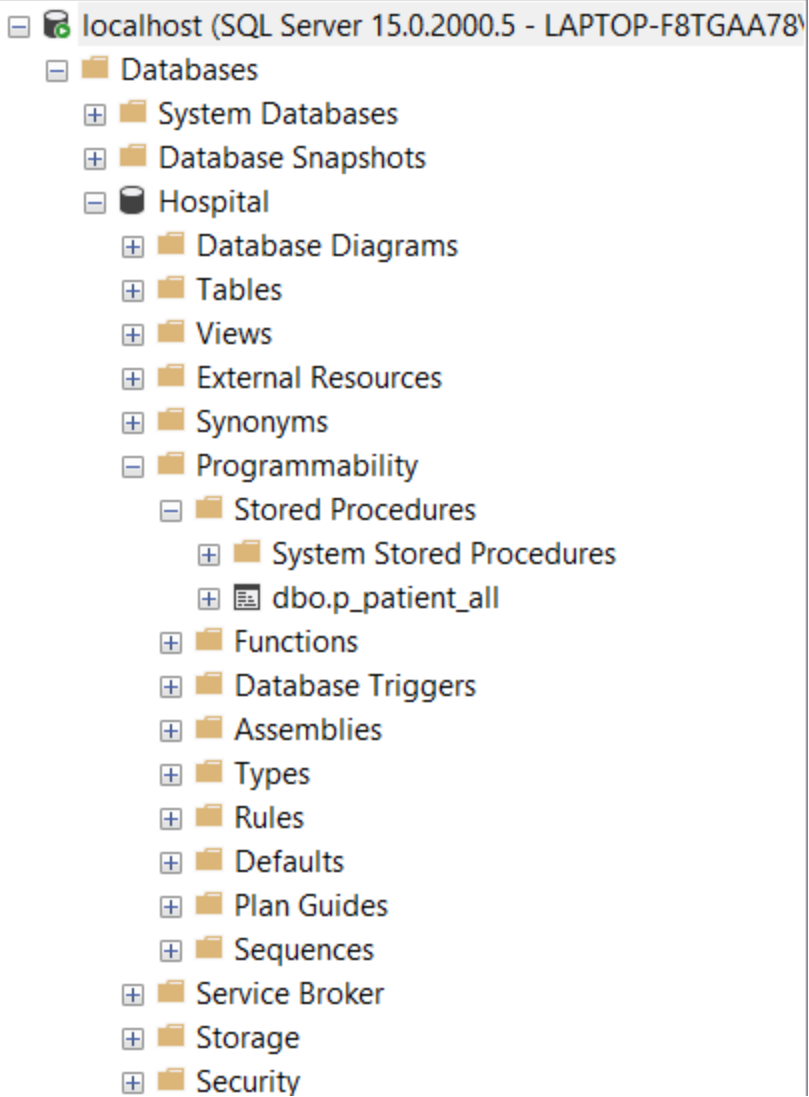
```
use Hospital
```






```
EXEC p_patient_all;
```

150 %

Results Messages

	patient_id	name	phone	address	disease	doctor_id	room_id
1	1	Mohan	127524789	UP	Nose Surgery	3	1
2	2	Sanchit	122424159	Bihar	Appendix	2	2
3	3	Gourab	121765529	MH	Sinus	1	3
4	4	Aparna	454222589	Karnataka	Liver Failure	4	4
5	5	Jaynit	1257531	UP	Leg Fracture	5	7



Connect ▾     

localhost (SQL Server 15.0.2000.5 - LAPTOP-F8TGAA78)

- Databases
 - System Databases
 - Database Snapshots
 - Hospital
 - Database Diagrams
 - Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Stored Procedures
 - System Stored Procedures
 - dbo.p_doctor
 - dbo.p_patient_all
 - Functions
 - Database Triggers
 - Assemblies
 - Types
 - Rules
 - Defaults
 - Plan Guides
 - Sequences
 - Service Broker
 - Storage
 - Security



```
use Hospital;  
DROP PROCEDURE IF EXISTS p_doctor;  
GO  
CREATE PROCEDURE p_doctor (@id INT)  
-- procedure returns all rows from the patient table  
AS BEGIN  
    SELECT *  
    FROM T1_Doctor  
    WHERE doctor_id = @id;  
END;
```

150 % ▾

Messages

Commands completed successfully.

Completion time: 2021-04-30T12:56:56.2450613+05:30

SQLQuery5.sql - Io...TGAA78\Nihar (56))*   SQLQuery3.sql - Io...TGAA78\Nihar (53)

```
EXEC p_doctor 1;
```

150 % ▾

Results Messages

	doctor_id	doctor_name	doctor_phone	doctor_address	doctor_specialization
1	1	ARJUN	123456789	UP	ENT

2. Write a transaction to illustrate atomicity (related to your database)

```
SQLQuery10.sql - I...TGAA78\Nihar (52))*  SQLQuery9.sql - lo...TGAA78\Nihar (51))* X
--atomicity--
use Hospital
BEGIN TRAN Transaction_update
UPDATE T1_Doctor SET doctor_address = 'Telangana' where doctor_id = 1
UPDATE T1_Patient SET phone = '8097283441' where patient_id = 5
COMMIT
```

150 %

Messages

(1 row affected)

(1 row affected)

Completion time: 2021-04-30T15:36:36.3279718+05:30

```
SQLQuery10.sql - I...TGAA78\Nihar (52))* X  SQLQuery9.sql - lo...TGAA78\Nihar (51))*
use Hospital
SELECT * from T1_Patient where patient_id = 5
SELECT * from T1_Doctor where doctor_id = 1
```

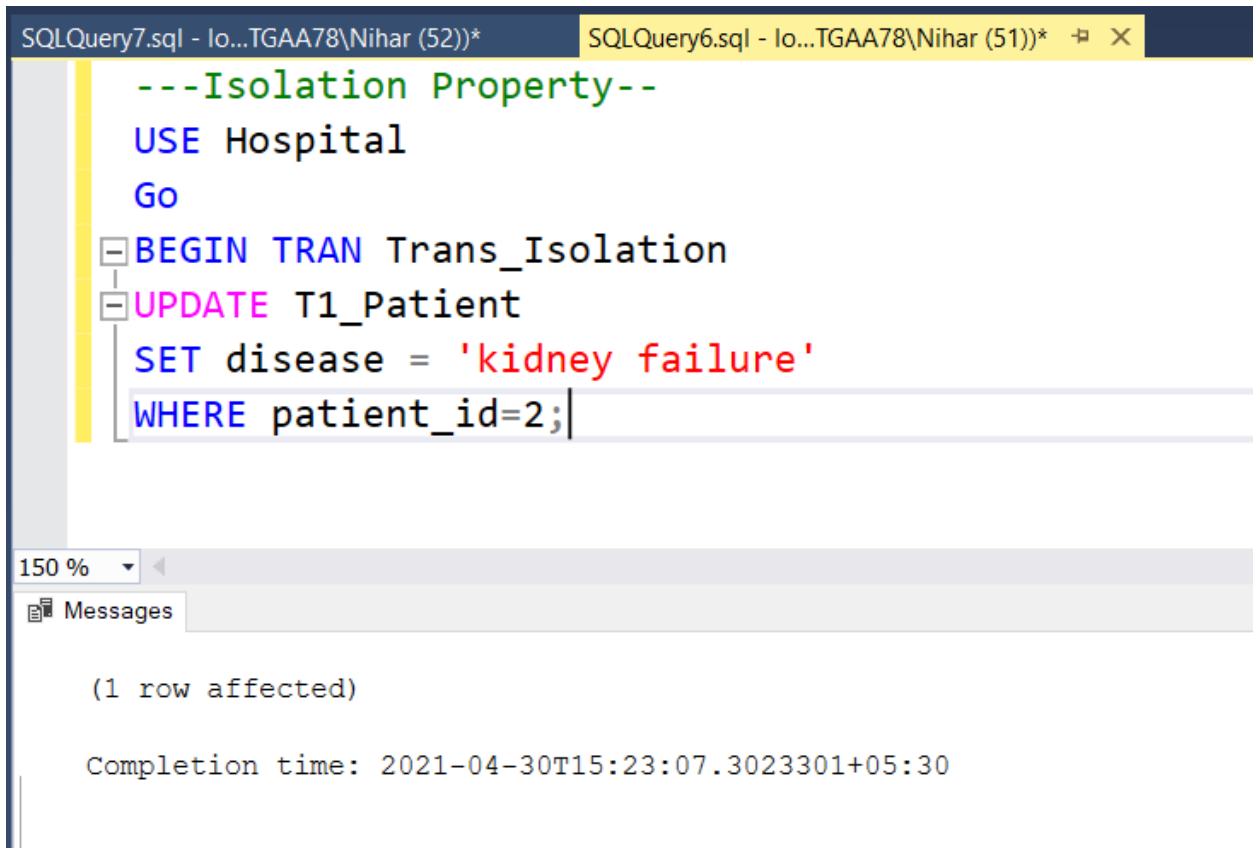
150 %

Results Messages

	patient_id	name	phone	address	disease	doctor_id	room_id
1	5	Jaynit	8097283441	UP	Leg Fracture	5	7

	doctor_id	doctor_name	doctor_phone	doctor_address	doctor_specialization
1	1	ARJUN	123456789	Telangana	ENT

3. Write a transaction to illustrate isolation level. It can be on commit or uncommitted read (related to your database)



The screenshot shows a SQL query editor with two tabs: 'SQLQuery7.sql - lo...TGAA78\Nihar (52))*' and 'SQLQuery6.sql - lo...TGAA78\Nihar (51))*'. The active tab contains the following SQL code:

```
---Isolation Property--  
USE Hospital  
Go  
BEGIN TRAN Trans_Isolation  
UPDATE T1_Patient  
SET disease = 'kidney failure'  
WHERE patient_id=2;
```

Below the code editor, there is a 'Messages' pane showing the execution results:

```
(1 row affected)  
  
Completion time: 2021-04-30T15:23:07.3023301+05:30
```

```
USE Hospital
```

```
Go
```

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
```

```
GO
```

```
[-] BEGIN TRAN Trans_Isolation1
```

```
[-] Select * from T1_Patient
```

```
WHERE patient_id = 2;
```

150 %

Results Messages

	patient_id	name	phone	address	disease	doctor_id	room_id
1	2	Sanchit	122424159	Bihar	kidney failure	2	2