

REQUIREMENTS:

---- Stored Procedure 1 ----

```
CREATE PROCEDURE CheckInsuranceCoverage
```

```
    @p_PatientID INT,
```

```
    @p_InsuranceCoverageType NVARCHAR(305) OUTPUT
```

```
AS
```

```
BEGIN
```

```
    SET @p_InsuranceCoverageType = NULL;
```

```
    SELECT @p_InsuranceCoverageType = Coverage
```

```
    FROM InsuranceCoverage
```

```
    WHERE PatientID = @p_PatientID;
```

```
END
```

```
DECLARE @InsuranceType NVARCHAR(255);
```

```
EXEC CheckInsuranceCoverage @p_PatientID = 101, @p_InsuranceCoverageType = @InsuranceType  
OUTPUT;
```

```
PRINT 'Insurance Coverage Type: ' + COALESCE(@InsuranceType, 'Not Available');
```

```
DECLARE @InsuranceType NVARCHAR(255);
```

```
EXEC CheckInsuranceCoverage @p_PatientID = 101, @p_InsuranceCoverageType = @InsuranceType  
OUTPUT;
```

```
PRINT 'Insurance Coverage Type: ' + COALESCE(@InsuranceType, 'Not Available');
```

```
DECLARE @InsuranceType1 NVARCHAR(255);
```

```
EXEC CheckInsuranceCoverage @p_PatientID = 102, @p_InsuranceCoverageType = @InsuranceType1  
OUTPUT;
```

```
PRINT 'Insurance Coverage Type: ' + COALESCE(@InsuranceType1, 'Not Available');
```

```
DECLARE @InsuranceType2 NVARCHAR(255);
```

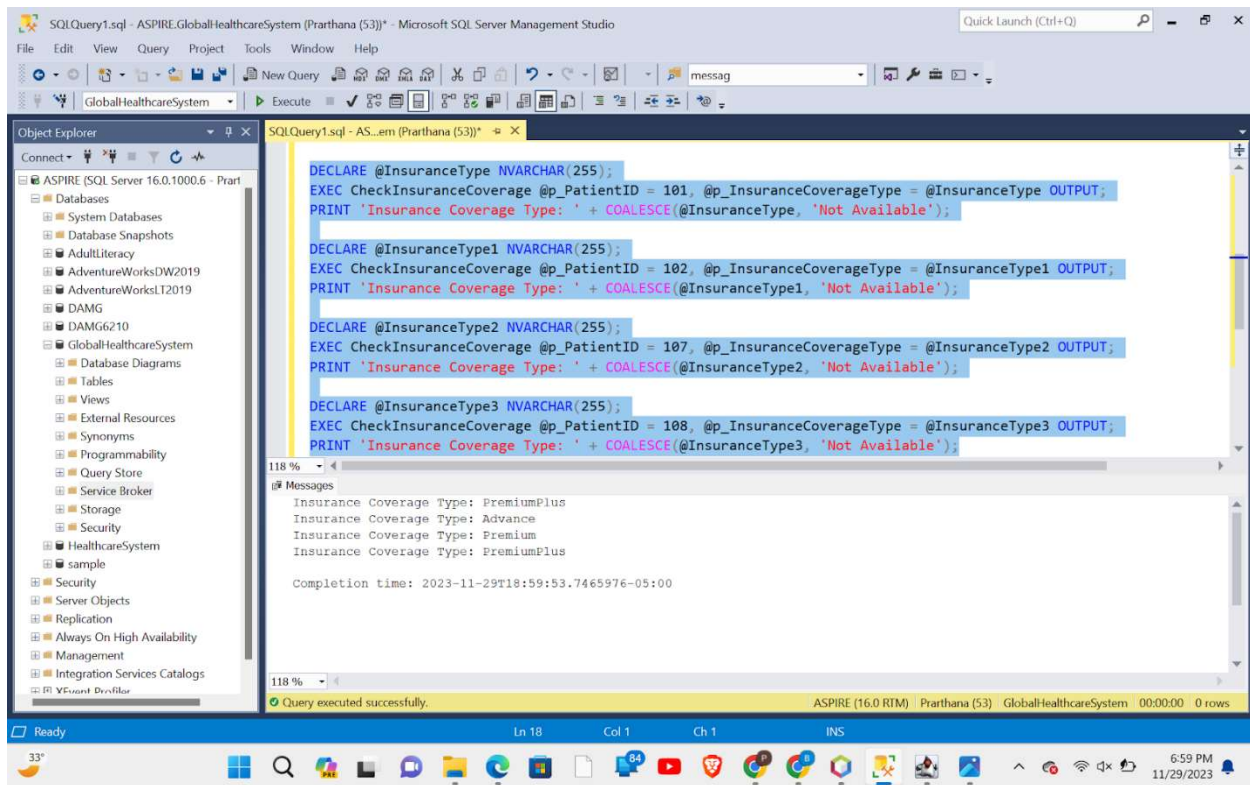
```
EXEC CheckInsuranceCoverage @p_PatientID = 107, @p_InsuranceCoverageType = @InsuranceType2  
OUTPUT;
```

```
PRINT 'Insurance Coverage Type: ' + COALESCE(@InsuranceType2, 'Not Available');
```

```
DECLARE @InsuranceType3 NVARCHAR(255);
```

```
EXEC CheckInsuranceCoverage @p_PatientID = 108, @p_InsuranceCoverageType = @InsuranceType3  
OUTPUT;
```

```
PRINT 'Insurance Coverage Type: ' + COALESCE(@InsuranceType3, 'Not Available');
```



---- Stored Procedure 2 ----

```
CREATE PROCEDURE CalculateInsuranceCoverage
```

```
@p_PatientID INT,
```

```

    @p_TotalBillAmount DECIMAL(10, 2),
    @p_PatientPayment DECIMAL(10, 2) OUTPUT,
    @p_InsuranceCoverage DECIMAL(10, 2) OUTPUT
AS
BEGIN
    DECLARE @InsuranceCoverageType VARCHAR(50);

    SELECT @InsuranceCoverageType = COVERAGE
    FROM InsuranceCoverage
    WHERE PatientID = @p_PatientID;

    SET @p_PatientPayment =
        CASE
            WHEN @InsuranceCoverageType = 'Basic' THEN @p_TotalBillAmount * 0.75
            WHEN @InsuranceCoverageType = 'Advance' THEN @p_TotalBillAmount * 0.5
            WHEN @InsuranceCoverageType = 'Premium' THEN @p_TotalBillAmount * 0.25
            WHEN @InsuranceCoverageType = 'PremiumPlus' THEN 0
            ELSE 100
        END;

    SET @p_InsuranceCoverage = @p_TotalBillAmount - @p_PatientPayment;

    SELECT
        @p_PatientPayment AS PatientPayment,
        @p_InsuranceCoverage AS InsuranceCoverage;
END

DECLARE @PatientPayment DECIMAL(10, 2);
DECLARE @InsuranceCoverage DECIMAL(10, 2);

```

EXEC CalculateInsuranceCoverage

@p_PatientID = 103,
@p_TotalBillAmount = 1000,
@p_PatientPayment = @PatientPayment OUTPUT,
@p_InsuranceCoverage = @InsuranceCoverage OUTPUT;

PRINT 'Patient Payment: ' + CAST(@PatientPayment AS VARCHAR(500));

PRINT 'Insurance Coverage: ' + CAST(@InsuranceCoverage AS VARCHAR(500));

EXEC CalculateInsuranceCoverage

@p_PatientID = 102,
@p_TotalBillAmount = 1000,
@p_PatientPayment = @PatientPayment OUTPUT,
@p_InsuranceCoverage = @InsuranceCoverage OUTPUT;

PRINT 'Patient Payment: ' + CAST(@PatientPayment AS VARCHAR(500));

PRINT 'Insurance Coverage: ' + CAST(@InsuranceCoverage AS VARCHAR(500));

EXEC CalculateInsuranceCoverage

@p_PatientID = 107,
@p_TotalBillAmount = 1000,
@p_PatientPayment = @PatientPayment OUTPUT,
@p_InsuranceCoverage = @InsuranceCoverage OUTPUT;

PRINT 'Patient Payment: ' + CAST(@PatientPayment AS VARCHAR(500));

PRINT 'Insurance Coverage: ' + CAST(@InsuranceCoverage AS VARCHAR(500));

EXEC CalculateInsuranceCoverage

@p_PatientID = 104,

@p_TotalBillAmount = 1000,

@p_PatientPayment = @PatientPayment OUTPUT,

@p_InsuranceCoverage = @InsuranceCoverage OUTPUT;

PRINT 'Patient Payment: ' + CAST(@PatientPayment AS VARCHAR(500));

PRINT 'Insurance Coverage: ' + CAST(@InsuranceCoverage AS VARCHAR(500));

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The left pane shows the 'Object Explorer' with the 'GlobalHealthcareSystem' database selected. The right pane shows the 'SQL Query1.sql' script with the following code:

```
CREATE PROCEDURE CalculateInsuranceCoverage
    @p_PatientID INT,
    @p_TotalBillAmount DECIMAL(10, 2),
    @p_PatientPayment DECIMAL(10, 2) OUTPUT,
    @p_InsuranceCoverage DECIMAL(10, 2) OUTPUT
AS
BEGIN
    DECLARE @InsuranceCoverageType VARCHAR(50);

    SELECT @InsuranceCoverageType = COVERAGE
    FROM InsuranceCoverage
    WHERE PatientID = @p_PatientID;

    SET @p_PatientPayment =
        CASE
            WHEN @InsuranceCoverageType = 'Basic' THEN @p_TotalBillAmount * 0.75
            WHEN @InsuranceCoverageType = 'Advance' THEN @p_TotalBillAmount * 0.5
            WHEN @InsuranceCoverageType = 'Premium' THEN @p_TotalBillAmount * 0.25
            WHEN @InsuranceCoverageType = 'PremiumPlus' THEN 0
            ELSE 100
        END;
END;
```

The 'Results' pane shows the output of the procedure execution:

PatientPayment	InsuranceCoverage
750.00	250.00

The status bar at the bottom indicates 'Query executed successfully.' and 'ASPIRE (16.0 RTM) Prarthana (53) GlobalHealthcareSystem 00:00:00 4 rows'.

SQLQuery1.sql - ASPIRE.GlobalHealthcareSystem (Prarthana (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

GlobalHealthcareSystem

Object Explorer

- Connect +
- ASPIRE (SQL Server 16.0.1000.6 - Prarthana)
 - Databases
 - System Databases
 - Database Snapshots
 - AdventureWorksDW2019
 - AdventureWorkst.I2019
 - DAMG
 - DAMG6210
 - GlobalHealthcareSystem
 - Database Diagrams
 - Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store
 - Service Broker
 - Storage
 - Security
 - HealthcareSystem
 - sample
 - Security
 - Server Objects
 - Replication
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - Event Viewer

SQLQuery1.sql - AS...em (Prarthana (53))

```

ELSE 100;
END;

SET @p_InsuranceCoverage = @p_TotalBillAmount - @p_PatientPayment;

SELECT
    @p_PatientPayment AS PatientPayment,
    @p_InsuranceCoverage AS InsuranceCoverage;
END

DECLARE @PatientPayment DECIMAL(10, 2);
DECLARE @InsuranceCoverage DECIMAL(10, 2);

EXEC CalculateInsuranceCoverage
    @p_PatientID = 103,
    @p_TotalBillAmount = 1000,
    @p_PatientPayment = @PatientPayment OUTPUT,
    @p_InsuranceCoverage = @InsuranceCoverage OUTPUT;

PRINT 'Patient Payment: ' + CAST(@PatientPayment AS VARCHAR(500));

```

89 %

Results Messages

	PatientPayment	InsuranceCoverage
1	750.00	250.00
1	500.00	500.00
1	250.00	750.00
1	0.00	1000.00

Query executed successfully. ASPIRE (16.0 RTM) Prarthana (53) GlobalHealthcareSystem 00:00:00 4 rows

Ready

33°

Windows taskbar icons and system clock: 7:05 PM 11/29/2023

SQLQuery1.sql - ASPIRE.GlobalHealthcareSystem (Prarthana (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

GlobalHealthcareSystem

Object Explorer

- Connect +
- ASPIRE (SQL Server 16.0.1000.6 - Prarthana)
 - Databases
 - System Databases
 - Database Snapshots
 - AdventureWorksDW2019
 - AdventureWorkst.I2019
 - DAMG
 - DAMG6210
 - GlobalHealthcareSystem
 - Database Diagrams
 - Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store
 - Service Broker
 - Storage
 - Security
 - HealthcareSystem
 - sample
 - Security
 - Server Objects
 - Replication
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - Event Viewer

SQLQuery1.sql - AS...em (Prarthana (53))

```

PRINT 'Patient Payment: ' + CAST(@PatientPayment AS VARCHAR(500));
PRINT 'Insurance Coverage: ' + CAST(@InsuranceCoverage AS VARCHAR(500));

EXEC CalculateInsuranceCoverage
    @p_PatientID = 102,
    @p_TotalBillAmount = 1000,
    @p_PatientPayment = @PatientPayment OUTPUT,
    @p_InsuranceCoverage = @InsuranceCoverage OUTPUT;

PRINT 'Patient Payment: ' + CAST(@PatientPayment AS VARCHAR(500));
PRINT 'Insurance Coverage: ' + CAST(@InsuranceCoverage AS VARCHAR(500));

EXEC CalculateInsuranceCoverage
    @p_PatientID = 107,
    @p_TotalBillAmount = 1000,
    @p_PatientPayment = @PatientPayment OUTPUT,
    @p_InsuranceCoverage = @InsuranceCoverage OUTPUT;

PRINT 'Patient Payment: ' + CAST(@PatientPayment AS VARCHAR(500));
PRINT 'Insurance Coverage: ' + CAST(@InsuranceCoverage AS VARCHAR(500));

```

89 %

Results Messages

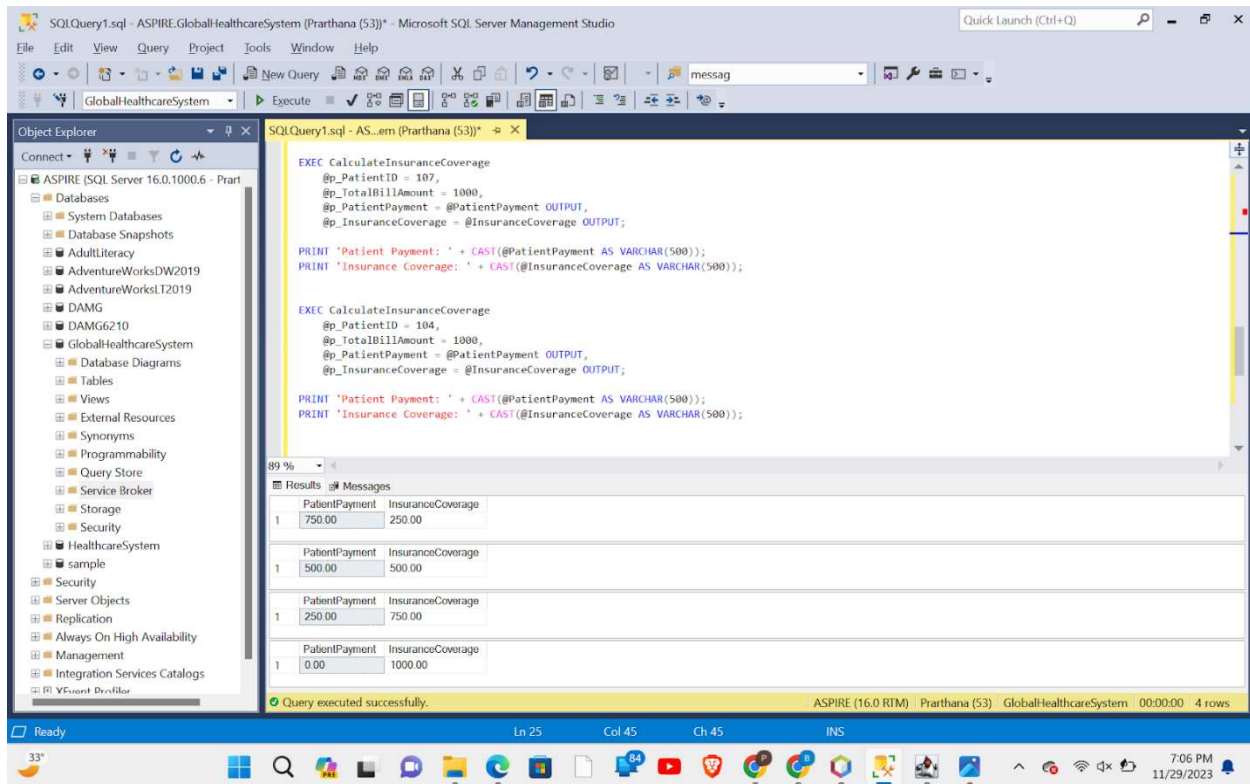
	PatientPayment	InsuranceCoverage
1	750.00	250.00
1	500.00	500.00
1	250.00	750.00
1	0.00	1000.00

Query executed successfully. ASPIRE (16.0 RTM) Prarthana (53) GlobalHealthcareSystem 00:00:00 4 rows

Ready

33°

Windows taskbar icons and system clock: 7:06 PM 11/29/2023



---- Stored Procedure 3 ----

CREATE PROCEDURE GetIllnessPrevalence

@Country VARCHAR(50)

AS

BEGIN

SELECT

T.[Diagnosed_Illness],

COUNT(P.PatientID) AS PatientCount

FROM

Treatment T

INNER JOIN

HealthcareInstitution H ON T.InstitutionID = H.InstitutionID

INNER JOIN

Patient P ON T.PatientID = P.PatientID

WHERE

P.Country = @Country

GROUP BY

T.[Diagnosed_Illness]

ORDER BY

PatientCount DESC;

END;

EXEC GetIllnessPrevalence @Country = 'USA';

EXEC GetIllnessPrevalence @Country = 'India';

The screenshot shows the Microsoft SQL Server Management Studio interface. The main window displays a SQL query in the 'SQLQuery12.sql' file. The query is as follows:

```
SELECT T.[Diagnosed_Illness],  
COUNT(P.PatientID) AS PatientCount  
FROM  
Treatment T  
INNER JOIN  
HealthcareInstitution H ON T.InstitutionID = H.InstitutionID  
INNER JOIN  
Patient P ON T.PatientID = P.PatientID  
WHERE  
P.Country = @Country  
GROUP BY  
T.[Diagnosed_Illness]  
ORDER BY  
PatientCount DESC;  
END;  
  
EXEC GetIllnessPrevalence @Country = 'USA';  
EXEC GetIllnessPrevalence @Country = 'India';
```

The 'Results' pane shows two tables of data. The first table is for the USA and the second is for India.

Diagnosed_Illness	PatientCount
COVID19	6
Diabetes	3
Flu	2
BP	2

Diagnosed_Illness	PatientCount
COVID19	6
BP	5
Flu	2
Diabetes	1

The status bar at the bottom indicates that the query was executed successfully, returning 8 rows.

---- VIEW 1 ----

CREATE VIEW PatientTreatmentHistoryView1 AS

SELECT

P.PatientID,
P.Patient_FirstName,
P.Patient_LastName,
T.TreatmentID,
T.[Diagnosed_Illness],
D.DoctorID,
D.Doctor_FirstName,
H.InstitutionID,
H.Institution_Name,
T.Date

FROM Patient P

JOIN Treatment T ON P.PatientID = T.PatientID

JOIN Doctor D ON T.DoctorID = D.DoctorID

JOIN HealthcareInstitution H ON T.InstitutionID = H.InstitutionID;

select * from PatientTreatmentHistoryView1

order by PatientID

SQLQuery_1.sql - localh...em (SA) 9+ finalQueries.sql - localh...em (SA) 9+ X db_sample.sql - localh...le (SA) 9+ localhost

Users > sohamshah > Documents > DMDD > Project > finalQueries.sql

Run Cancel Disconnect Change Database: GlobalHealthCareSyst... Estimated Plan Enable Actual Plan Parse Enable SQLCMD To Notebook

```

1416 ----- VIEW 1 -----
1417
1418 CREATE VIEW PatientTreatmentHistoryView1 AS
1419 SELECT
1420     P.PatientID,
1421     P.Patient_FirstName,
1422     P.Patient_LastName,
1423     T.TreatmentID,
1424     T.(Diagnosed_Illness),
1425     D.DoctorID,
1426     D.Doctor_FirstName,
1427     H.InstitutionID,
1428     H.Institution_Name,
1429     T.Date
1430 FROM Patient P
1431 JOIN Treatment T ON P.PatientID = T.PatientID
1432 JOIN Doctor D ON T.DoctorID = D.DoctorID
1433 JOIN HealthcareInstitution H ON T.InstitutionID = H.InstitutionID;
1434
1435 select * from PatientTreatmentHistoryView1
1436 order by PatientID
1437

```

Results Messages

	PatientID	Patient_FirstName	Patient_LastName	TreatmentID	Diagnosed_Illness	DoctorID	Doctor_FirstName	InstitutionID	Institution_Name
1	101	John	Doe	501	COVID19	201	Anna	1	City General Hospital
2	101	John	Doe	513	Flu	212	Pavel	2	Suburb Medical Center
3	102	Alice	Smith	514	COVID19	214	Yuri	4	Metro Health Center
4	102	Alice	Smith	502	BP	202	Vladimir	2	Suburb Medical Center
5	103	Charlie	Brown	503	COVID19	203	Natalia	3	Rural Clinic
6	103	Charlie	Brown	515	Flu	215	Yulia	5	Coastal Medical Clinic
7	104	Eva	Johnson	516	BP	216	Nikolai	6	Hilltop Wellness Center
8	104	Eva	Johnson	504	BP	204	Ivan	4	Metro Health Center
9	105	Grace	Williams	505	Diabetes	205	Elena	5	Coastal Medical Clinic
10	105	Grace	Williams	517	COVID19	217	Ekaterina	7	Valley Family Health
11	106	Ian	Miller	518	Flu	218	Artem	8	Downtown Urgent Care
12	106	Ian	Miller	506	COVID19	206	Sergei	6	Hilltop Wellness Center
13	107	Karen	Davis	507	Diabetes	207	Maria	7	Valley Family Health

Ln 72, Col 0 Spaces: 4 UTF-8 LF SQL 35 rows Choose SQL Language 00:00:00 localhost: GlobalHealthCareSystem

---- View 2 -----

CREATE VIEW MostEfficientRegulatoryDept AS

SELECT TOP 1

R.Regulatory_Dept_ID,

RD.Authorizer_Name,

COUNT(R.UniqueRecordID) AS ReportCount

FROM

RegulatoryDept RD

JOIN

Record R ON RD.Regulatory_Dept_ID = R.Regulatory_Dept_ID

GROUP BY

R.Regulatory_Dept_ID, RD.Authorizer_Name

ORDER BY

ReportCount DESC;

SELECT * FROM MostEfficientRegulatoryDept;

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SERVERS' tree is expanded to show the 'GlobalHealthCareSystem' database, with the 'Tables' folder selected. The 'dbo.RegulatoryDept' table is highlighted. The main window displays a SQL query in the 'SQLQuery_1.sql' file. The query is as follows:

```

1249 ----- View 2 -----
1250
1251
1252 CREATE VIEW MostEfficientRegulatoryDept AS
1253 SELECT TOP 1
1254     R.Regulatory_Dept_ID,
1255     RD.Authorizer_Name,
1256     COUNT(R.UniqueRecordID) AS ReportCount
1257 FROM
1258     RegulatoryDept RD
1259 JOIN
1260     Record R ON RD.Regulatory_Dept_ID = R.Regulatory_Dept_ID
1261 GROUP BY
1262     R.Regulatory_Dept_ID, RD.Authorizer_Name
1263 ORDER BY
1264     ReportCount DESC;
1265
1266 SELECT * FROM MostEfficientRegulatoryDept;
  
```

Below the query, the 'Results' tab shows the output of the query. The results are as follows:

Regulatory_Dept_ID	Authorizer_Name	ReportCount
1002	Jason Nash	19

---- View 3 ----

CREATE VIEW TreatmentPatientTurnover AS

SELECT

T.TreatmentID,

COUNT(P.PatientID) AS PatientCount,

COUNT(P.PatientID) * T.Cost AS TreatmentTurnover

FROM

Patient P

JOIN

TREATMENT T ON P.PatientID = T.PatientID

GROUP BY

T.TreatmentID, T.Cost;

(--Inserting a dummy value in treatment 501 to check

```
INSERT INTO Treatment (TreatmentID, PatientID, InstitutionID, DoctorID, Cost, Description, Date, Diagnosed_Illness)
```

```
VALUES (501, 102, 1, 201, 500, 'COVID-19 Diagnostic Check-up', '2023-05-25', 'COVID19');
```

```
SELECT * FROM Treatment)
```

```
SELECT * FROM TreatmentPatientTurnover
```

```
order by TreatmentTurnover DESC
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
INSERT INTO Treatment (TreatmentID, PatientID, InstitutionID, DoctorID, Cost, Description, Date, Diagnosed_Illness)
VALUES (501, 102, 1, 201, 500, 'COVID-19 Diagnostic Check-up', '2023-05-25', 'COVID19');

SELECT * FROM Treatment

CREATE VIEW TreatmentPatientTurnover AS
SELECT
    T.TreatmentID,
    COUNT(P.PatientID) AS PatientCount,
    COUNT(P.PatientID) * T.Cost AS TreatmentTurnover
FROM
    Patient P
JOIN
    TREATMENT T ON P.PatientID = T.PatientID
GROUP BY
    T.TreatmentID, T.Cost;

SELECT * FROM TreatmentPatientTurnover
order by TreatmentTurnover DESC
```

The Results pane shows the output of the query, displaying a table with the following data:

TreatmentID	PatientCount	TreatmentTurnover
511	1	5000
501	2	1000
506	1	900
509	1	900
504	1	800
510	1	800
503	1	700
520	1	650
519	1	550
514	1	500
517	1	450
502	1	400
507	1	400
476	1	400

The status bar at the bottom indicates: "Query executed successfully. ASPIRE (16.0 RTM) ASPIRE\prart (57) GlobalHealthcareSystem 00:00:00 20 rows".

---- Trigger 1 ----

```
CREATE TRIGGER tr_Patient_Audit
```

```
ON Patient
```

```
AFTER INSERT, UPDATE
```

```
AS
```

```
BEGIN
```

INSERT INTO AuditPatient (PatientID, ChangeType, ChangeDate)

SELECT

i.PatientID,

CASE

WHEN EXISTS (SELECT * FROM INSERTED i, DELETED d WHERE i.PatientID = d.PatientID) THEN
'UPDATE'

WHEN EXISTS (SELECT * FROM INSERTED) THEN 'INSERT'

WHEN EXISTS (SELECT * FROM DELETED) THEN 'DELETE'

END AS ChangeType,

GETDATE() AS ChangeDate

FROM INSERTED i

FULL JOIN DELETED d ON i.PatientID = d.PatientID;

DELETE from AuditPatient

where PatientID = NULL;

END;

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'SERVERS' tree shows the 'GlobalHealthCareSystem' database selected. The main pane shows a SQL script for creating a trigger named 'tr_Patient_Audit' on the 'Patient' table. The script includes logic for auditing insert, update, and delete operations by inserting into an 'AuditPatient' table. The 'Messages' pane at the bottom shows a successful execution message at 6:02:27 AM, indicating that the commands completed successfully and the total execution time was 00:00:00.009.

```
353 CREATE TRIGGER tr_Patient_Audit
354 ON Patient
355 AFTER INSERT, UPDATE
356 AS
357 BEGIN
358     -- Your auditing logic here, e.g., inserting into an audit table
359     INSERT INTO AuditPatient (PatientID, ChangeType, ChangeDate)
360     SELECT
361         i.PatientID,
362         CASE
363             WHEN EXISTS (SELECT * FROM INSERTED i, DELETED d WHERE i.PatientID = d.PatientID) THEN 'UPDATE'
364             WHEN EXISTS (SELECT * FROM INSERTED) THEN 'INSERT'
365             WHEN EXISTS (SELECT * FROM DELETED) THEN 'DELETE'
366         END AS ChangeType,
367         GETDATE() AS ChangeDate
368     FROM INSERTED i
369     FULL JOIN DELETED d ON i.PatientID = d.PatientID;
370     DELETE from PatientAudit
371     where PatientID = NULL;
372 END;
373
374 CREATE TRIGGER tr_Patient_Audit1
375 ON Patient
376 AFTER DELETE
377 AS
378 BEGIN
379     INSERT INTO DeletedPatient
380     SELECT *
381     FROM DELETED;
382 END;
383
```

Messages

6:02:27 AM Started executing query at line 374
Commands completed successfully.
Total execution time: 00:00:00.009

```
INSERT INTO Patient (PatientID, Patient_FirstName, Patient_MiddleName, Patient_LastName,
Patient_Phone_Num,
```

```
Patient_Date_of_Birth, Sex, Height, Weight, Blood_Group, Address, Country,
```

```
Next_of_Kin_Name, Emergency_Phone_Number, Patient_Password)
```

```
VALUES
```

```
(113, 'Georgia', 'M', 'Rodrigues', 857213455689, '1999-05-29', 'F', 1.50, 50, 'B+', 'Mumbai', 'India',
```

```
'Michael James', 8576196543210, EncryptByKey(Key_GUID('PatientPass_SM'), CONVERT(VARBINARY,
'Pass101')));
```

```
SELECT * from AuditPatient;
```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the server hierarchy for 'damg6210'. The central query window shows the execution of an INSERT statement into the 'Patient' table. The statement includes a VALUES clause with patient details. Below the query, a 'SELECT * from AuditPatient;' statement is also visible. The bottom pane, titled 'Results', shows the output of the SELECT statement, which is a single row from the AuditPatient table.

AuditID	PatientID	ChangeType	ChangeDate
1	113	INSERT	2023-11-24 11:11:19.790

```
UPDATE Patient
```

```
SET Weight = 52
```

```
WHERE PatientID = 113;
```

SELECT * from AuditPatient;

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Servers' tree is expanded to show the 'GlobalHealthCareSystem' database. The 'Tables' folder is expanded, and 'dbo.AuditPatient' is selected. The main window displays a query execution window with the following SQL code:

```
394 UPDATE Patient
395 SET Weight = 52
396 WHERE PatientID = 113;
397
398 SELECT * from AuditPatient;
399
```

Below the query window, the 'Results' tab is active, showing a table with 2 rows and 5 columns: AuditID, PatientID, ChangeType, and ChangeDate. The data is as follows:

AuditID	PatientID	ChangeType	ChangeDate
1	2	INSERT	2023-11-24 11:11:19.790
2	3	UPDATE	2023-11-24 11:17:34.697

The status bar at the bottom indicates the current position is Line 398, Column 4 (28 selected), with 4 spaces, UTF-8 encoding, LF line endings, 2 rows, MSSQL engine, and a duration of 00:00:00.

---- Trigger 2 ----

CREATE TRIGGER tr_Patient_Audit1

ON Patient

AFTER DELETE

AS

BEGIN

INSERT INTO DeletedPatient

SELECT *

FROM DELETED;

END;

DELETE from Patient WHERE PatientID = 113

Select * from DeletedPatient

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the server hierarchy for 'damg6210', with the 'GlobalHealthCareSystem' database selected. The right pane shows a SQL query window with the following text:

```
402 DELETE from Patient where PatientID = 113
403
404 Select * from DeletedPatient
405
406
407
408
```

Below the query window, the 'Results' tab is active, displaying a table with 9 columns: PatientID, Patient_FirstName, Patient_MiddleName, Patient_LastName, Patient_Phone_Num, Patient_Date_of_Birth, Sex, Height, and an unlabeled column. The table contains two rows of data:

PatientID	Patient_FirstName	Patient_MiddleName	Patient_LastName	Patient_Phone_Num	Patient_Date_of_Birth	Sex	Height	
1	Georgia	M	Rodrigues	857213455689	1999-05-29	F	1.5	
2	Georgia	M	Rodrigues	857213455689	1999-05-29	F	1.5	

The status bar at the bottom indicates 'Ln 404, Col 1 (28 selected) Spaces: 4 UTF-8 LF 2 rows MSSQL 00:00:00 localhost : GlobalHealthCareSystem'.

---- Trigger 3 ----

CREATE TRIGGER tr_Patient_ValidateEmergencyPhoneNumber

ON Patient

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM INSERTED

WHERE LEN(CONVERT(VARCHAR(20), Emergency_Phone_Number)) <> 10

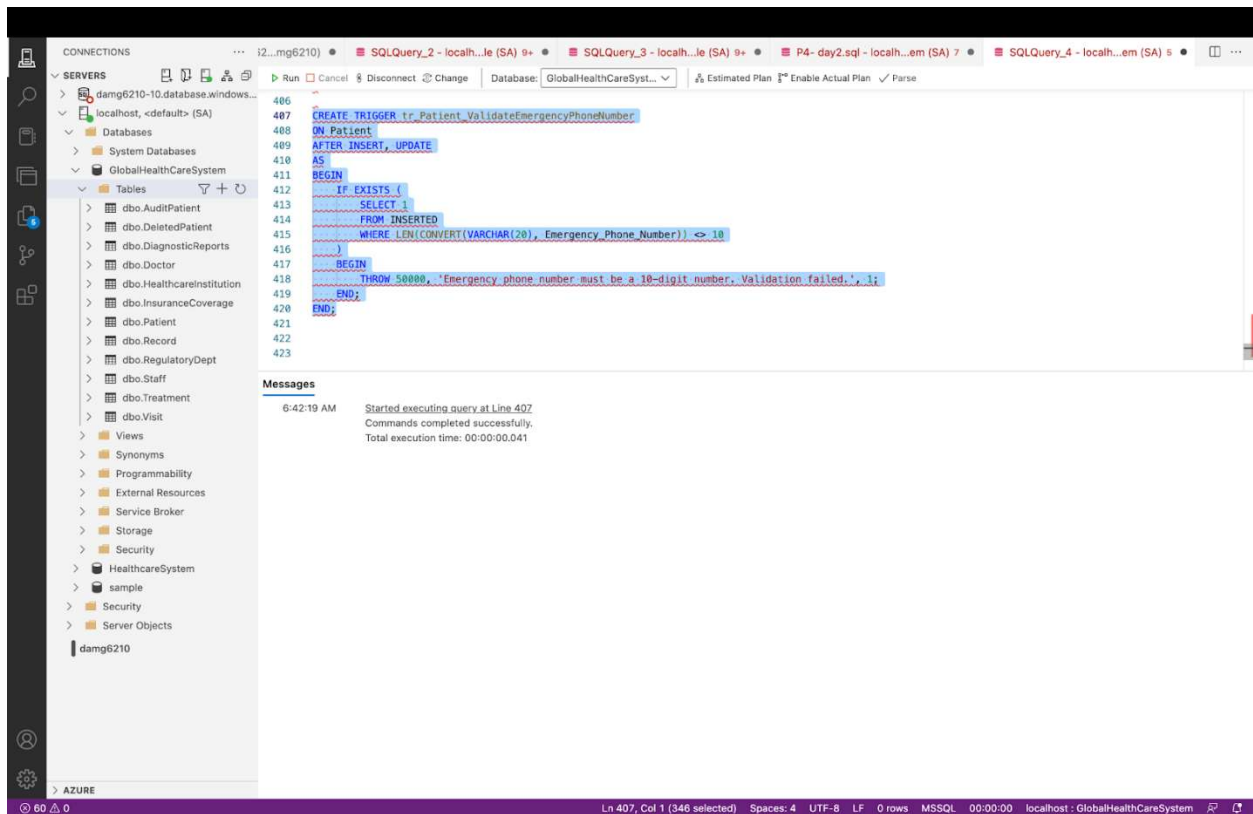
)

BEGIN

THROW 50000, 'Emergency phone number must be a 10-digit number. Validation failed.', 1;

END;

END;



UPDATE Patient

SET Emergency_Phone_Number = '876538926'

WHERE PatientID = 144;

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'SERVERS' tree shows a local instance of SQL Server with various databases and tables. The 'GlobalHealthCareSystem' database is expanded, showing tables like 'Patient', 'Prescription', and 'Record'. The main pane shows the definition of a trigger named 'tr_Patient_ValidateEmergencyPhoneNumber' on the 'Patient' table. The trigger is an AFTER INSERT, UPDATE trigger that checks if the emergency phone number is a 10-digit number. If it is not, it throws an error and updates the patient's emergency phone number to '876538926'.

```
1357 ----- Trigger 3 -----
1358
1359 CREATE TRIGGER tr_Patient_ValidateEmergencyPhoneNumber
1360 ON Patient
1361 AFTER INSERT, UPDATE
1362 AS
1363 BEGIN
1364     IF EXISTS (
1365         SELECT 1
1366         FROM INSERTED
1367         WHERE LEN(CONVERT(VARCHAR(20), Emergency_Phone_Number)) <> 10
1368     )
1369     BEGIN
1370         THROW 50000, 'Emergency phone number must be a 10-digit number. Validation failed.', 1;
1371     END;
1372 END;
1373
1374 UPDATE Patient
1375 SET Emergency_Phone_Number = '876538926'
1376 WHERE PatientID = 144;
1377
```

The 'Messages' pane at the bottom shows the execution results:

```
7:57:30 PM Started executing query at Line 1374
(1 row affected)
(0 rows affected)
Msg 50000, Level 16, State 1, Procedure tr_Patient_ValidateEmergencyPhoneNumber, Line 12
Emergency phone number must be a 10-digit number. Validation failed.
Total execution time: 00:00:00.029
```

select * from Patient
where PatientID = 144;

CONNECTIONS: SQLQuery_1.sql - localh...em (SA) 9+ finalQueries.sql - localh...em (SA) 9+ db_sample.sql - localh...le (SA) 9+ localhost

SERVERS: localhost, <default> (SA) GlobalHealthCareSystem

GlobalHealthCareSystem Tables:

- dbo.AuditPatient
- dbo.DeletedPatient
- dbo.DiagnosticReports
- dbo.Doctor
- dbo.HealthcareInstitution
- dbo.InsuranceCoverage
- dbo.Patient
- dbo.Prescription
- dbo.Record
- dbo.RegulatoryDept
- dbo.Staff
- dbo.Treatment
- dbo.Visit

Views: Synonyms, Programmability, External Resources, Service Broker, Storage, Security, HealthcareSystem, sample, Tables

AZURE: Loading ...

Query Editor: Users > sohamshah > Documents > DMDD > Project > finalQueries.sql

Database: GlobalHealthCareSyst...

Run Cancel Disconnect Change Database: GlobalHealthCareSyst... Estimated Plan Enable Actual Plan Parse Enable SQLCMD To Notebook

```

1360 ON Patient
1361 AFTER INSERT, UPDATE
1362 AS
1363 BEGIN
1364     IF EXISTS (
1365         SELECT 1
1366         FROM INSERTED
1367         WHERE LEN(CONVERT(VARCHAR(20), Emergency_Phone_Number)) < 10
1368     )
1369     BEGIN
1370         THROW 50000, 'Emergency phone number must be a 10-digit number. Validation failed.', 1;
1371     END;
1372 END;
1373
1374 UPDATE Patient
1375 SET Emergency_Phone_Number = '876538926'
1376 WHERE PatientID = 144;
1377
1378 select * from Patient
1379 where PatientID = 144;
1380

```

Results Messages

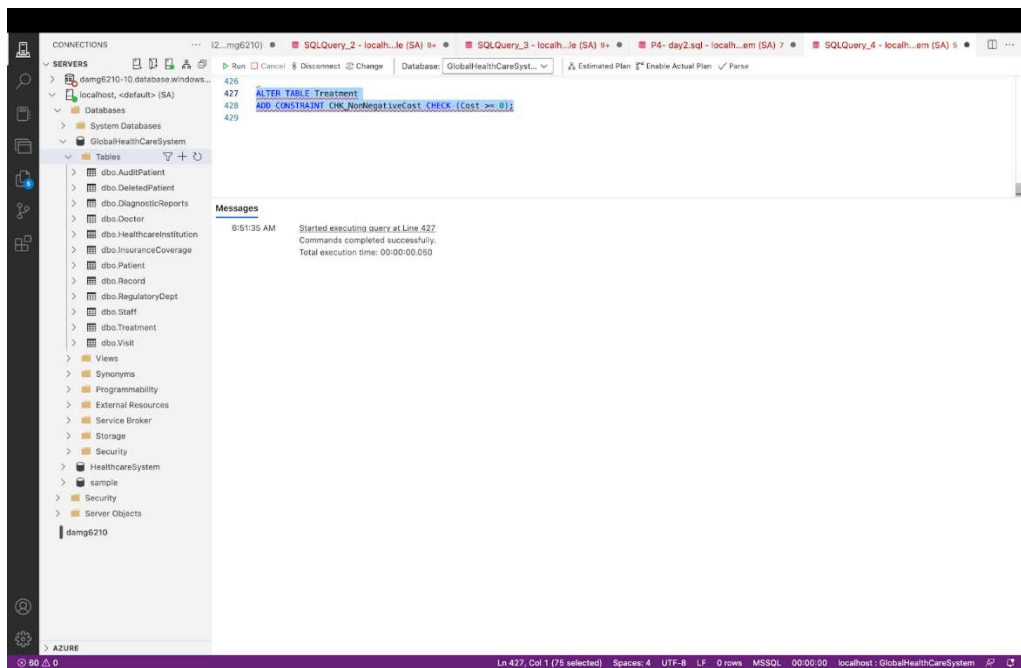
Height	Weight	Blood_Group	Address	Country	Next_of_Kin_Name	Emergency_Phone_Number	Patient_Password
1.6	55	AB-	15 Moscow Avenue	Russia	Igor Volkov	8576543210	0x003C5309CA4D1045828485f

Ln 1378, Col 1 (44 selected) Spaces: 4 UTF-8 LF SQL 1 rows Choose SQL Language 00:00:00 localhost: GlobalHealthCareSystem

---- Check Constraint on Cost using Table Treatment ----

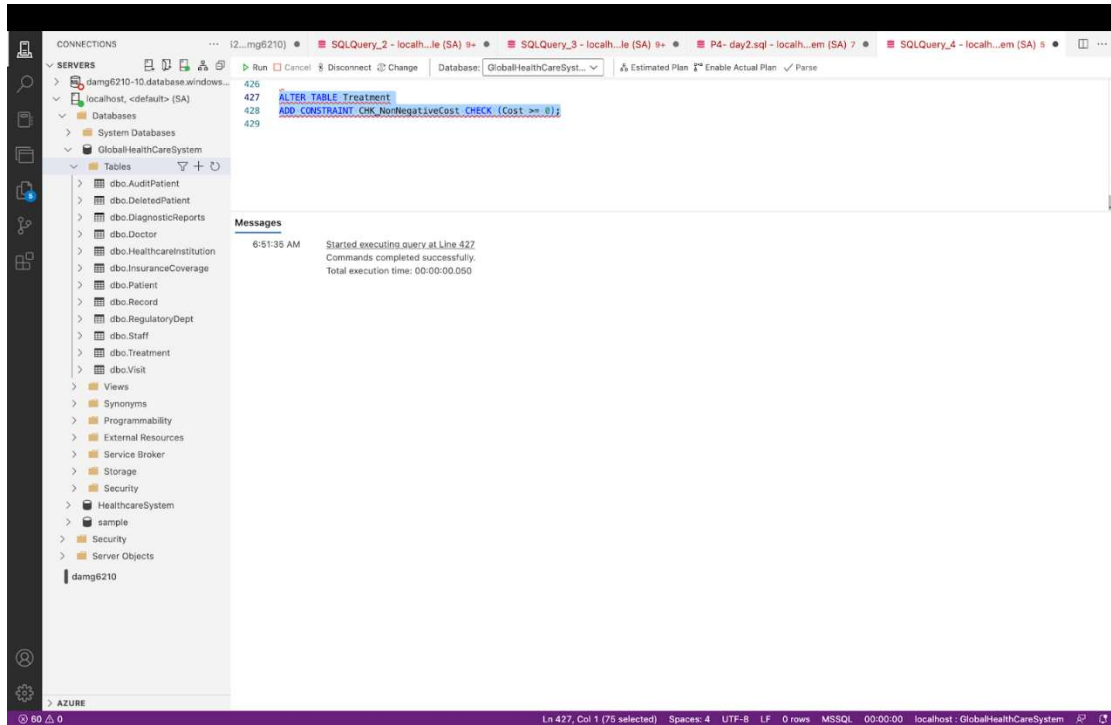
ALTER TABLE Treatment

ADD CONSTRAINT CHK_NonNegativeCost CHECK (Cost >= 0);

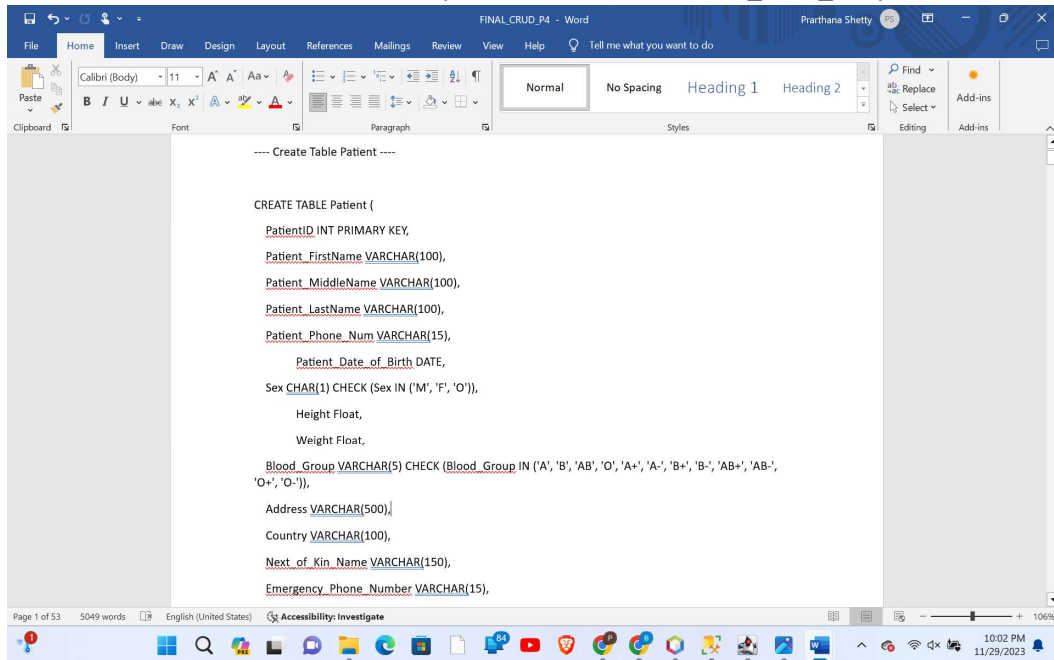


Update Treatment set Cost = -1000

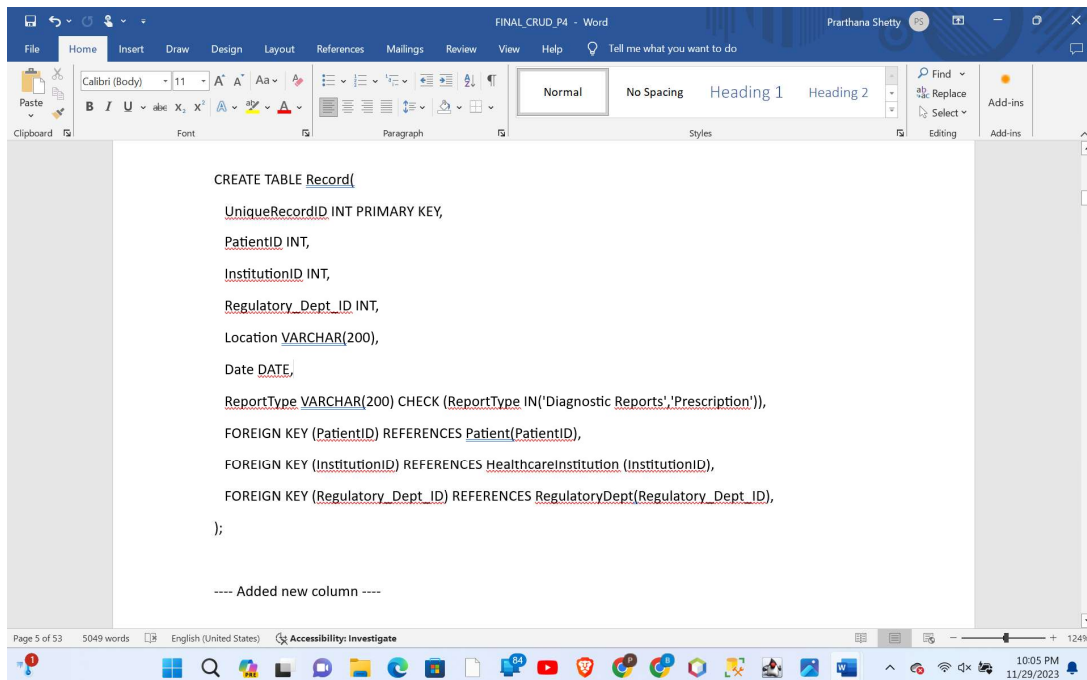
where TreatmentID = 520



//Patient sex and Patient Blood Group Constraints (Refer 'Final_Crud_P4' pdf file and 'SQL_CRUD' file)



//Record Constraint for 'ReportType' which acts as a Discriminator for 'Record' table Subtypes 'Diagnostic Reports' and 'Prescription' (Refer 'Final_Crud_P4' pdf file and 'SQL_CRUD' file)



---- UDF (User Defined Function)

CREATE FUNCTION CalculateBMI(@Weight FLOAT, @Height FLOAT)

RETURNS FLOAT

AS

BEGIN

DECLARE @BMI FLOAT;

SET @BMI = @Weight / POWER(@Height, 2);

RETURN @BMI;

END;

ALTER TABLE Patient

ADD BMI AS dbo.CalculateBMI(Weight, Height);

SELECT PatientID, Weight, Height, BMI

FROM Patient;

The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the server hierarchy for 'localhost' (sa), with the 'GlobalHealthCareSystem' database selected. The right pane shows the execution of a SQL script. The script includes the following statements:

```
CREATE FUNCTION CalculateBMI(@Weight FLOAT, @Height FLOAT)
RETURNS FLOAT
AS
BEGIN
    DECLARE @BMI FLOAT;
    SET @BMI = @Weight / POWER(@Height, 2);
    RETURN @BMI;
END;

ALTER TABLE Patient
ADD BMI AS dbo.CalculateBMI(Weight, Height);

SELECT PatientID, Weight, Height, BMI
FROM Patient;
```

The 'Results' pane shows the output of the SELECT statement, displaying 12 rows of patient data with their IDs, weights, heights, and calculated BMI values.

	PatientID	Weight	Height	BMI
1	101	70	1.7	24.221453287197235
2	102	55	1.6	21.484374999999996
3	103	80	1.75	26.122448979591837
4	104	50	1.55	20.811654526534856
5	105	60	1.6	23.437499999999996
6	106	85	1.8	26.234567981234566
7	107	70	1.65	25.71166207529844
8	108	75	1.75	24.489795918367346
9	109	55	1.6	21.484374999999996
10	110	80	1.7	27.68166089965398
11	111	70	1.7	24.221453287197235
12	112	55	1.6	21.484374999999996

---- Encryption ----

create MASTER KEY

ENCRYPTION BY PASSWORD = 'ShreeRam@765';

```
SELECT NAME KeyName,  
symmetric_key_id KeyID,  
key_length KeyLength,  
algorithm_desc KeyAlgorithm  
FROM sys.symmetric_keys;  
go
```

```
CREATE CERTIFICATE PatientPass  
WITH SUBJECT = 'Patient Password';  
GO
```

```
CREATE SYMMETRIC KEY PatientPass_SM  
WITH ALGORITHM = AES_256  
ENCRYPTION BY CERTIFICATE PatientPass;  
GO
```

```
OPEN SYMMETRIC KEY PatientPass_SM  
DECRYPTION BY CERTIFICATE PatientPass;
```

```
INSERT INTO Patient (PatientID, Patient_FirstName, Patient_MiddleName, Patient_LastName,  
Patient_Phone_Num,  
Patient_Date_of_Birth, Sex, Height, Weight, Blood_Group, Address, Country,
```

Next_of_Kin_Name, Emergency_Phone_Number, Patient_Password)

VALUES

(146, 'Shantanu', 'R', 'Mahakal', 8575769054, '1998-10-30', 'M', 1.68, 60, 'B+', '15 Pond St', 'USA',
'Prarthana Doe', 8579876510, EncryptByKey(Key_GUID('PatientPass_SM'), CONVERT(VARBINARY,
'Pass146')));

SELECT PatientID, Patient_FirstName, Patient_LastName, Patient_Password FROM Patient
where PatientID = 146;

UPDATE Patient set patientpass = 'Pass146' where PatientID= 146

SELECT * FROM Patient

The screenshot shows a SQL Server Enterprise Manager interface. The query window displays the following SQL code:

```
1427 CREATE SYMMETRIC KEY PatientPass_SM
1428 WITH ALGORITHM = AES_256
1429 ENCRYPTION BY CERTIFICATE PatientPass;
1430 GO
1431
1432 OPEN SYMMETRIC KEY PatientPass_SM
1433 DECRYPTION BY CERTIFICATE PatientPass;
1434
1435 INSERT INTO Patient (PatientID, Patient_FirstName, Patient_MiddleName, Patient_LastName, Patient_Phone_Num,
1436 Patient_Date_of_Birth, Sex, Height, Weight, Blood_Group, Address, Country,
1437 Next_of_Kin_Name, Emergency_Phone_Number, Patient_Password)
1438 VALUES
1439 (146, 'Shantanu', 'R', 'Mahakal', 8575769054, '1998-10-30', 'M', 1.68, 60, 'B+', '15 Pond St', 'USA',
1440 'Prarthana Doe', 8579876510, EncryptByKey(Key_GUID('PatientPass_SM'), CONVERT(VARBINARY, 'Pass146')));
1441
1442 SELECT PatientID, Patient_FirstName, Patient_LastName, Patient_Password FROM Patient
1443 where PatientID = 146;
1444
```

The Results pane shows a single row of data:

PatientID	Patient_FirstName	Patient_LastName	Patient_Password
146	Shantanu	Mahakal	0x003C5309CA4D1045828485632624828E020000008B572B266E...

--- to decrypt ---

OPEN SYMMETRIC KEY PatientPass_SM


```

DECRYPTION BY CERTIFICATE PatientPass;

SELECT *,

CONVERT(varchar, DecryptByKey([Patient_Password]))

AS 'Decrypted password'

FROM Patient;

GO

```

The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays a query window with the following script:

```

1448 SET Patient_Password = EncryptByKey(Key_GUID('PatientPass_SM'), CONVERT(VARBINARY, 'Passw1'))
1449 where PatientID=101;
1450
1451 SELECT * FROM Patient
1452
1453 --- to decrypt ---
1454 OPEN SYMMETRIC KEY PatientPass_SM
1455 DECRYPTION BY CERTIFICATE PatientPass;
1456 SELECT *,
1457 CONVERT(varchar, DecryptByKey([Patient_Password]))
1458 AS 'Decrypted password'
1459 FROM Patient;
1460 GO

```

The bottom pane shows the results of the query, displaying a table with the following columns: Country, Next_of_Kin_Name, Emergency_Phone_Number, Patient_Password, BMI, PatientPass, and Decrypted password. The table contains 16 rows of data, including patients from China, USA, India, Russia, and Prarthana Doe.

Country	Next_of_Kin_Name	Emergency_Phone_Number	Patient_Password	BMI	PatientPass	Decrypted password
China	Sophie Li	8576543210	0x003C5309CA4D1045828485632024828E02000000954E5E006...	22.857142857142858	Passw33	Pass133
China	William Yang	8575432109	0x003C5309CA4D1045828485632024828E020000002243118E5...	21.484374999999999	Passw34	Pass134
China	Sophia Chen	8574321098	0x003C5309CA4D1045828485632024828E02000000AF6ACD89BA...	24.61810377477591	Passw35	Pass135
USA	Matthew Lee	8573210987	0x003C5309CA4D1045828485632024828E0200000084407920788...	22.03856749311295	Passw36	Pass136
China	Emma Wang	8579876543	0x003C5309CA4D1045828485632024828E020000008BAF9C648...	25.95155709342561	Passw37	Pass137
India	Arjun Das	8578765432	0x003C5309CA4D1045828485632024828E0200000068D5757545...	26.122448979591837	Passw38	Pass138
India	Sophie Rao	8577654321	0x003C5309CA4D1045828485632024828E02000000745515495C...	24.092970521541954	Passw39	Pass139
China	David Jin	8576543210	0x003C5309CA4D1045828485632024828E02000000F51075E7DA...	26.234567901234566	Passw40	Pass140
Russia	Maria Nikolayeva	8579876543	0x003C5309CA4D1045828485632024828E02000000C9E2D02745...	24.489795918367346	Passw41	Pass141
Russia	Alexei Sokolov	8578765432	0x003C5309CA4D1045828485632024828E02000000EEDFDD999...	23.030045351473927	Passw42	Pass142
Russia	Svetlana Ivanova	8577654321	0x003C5309CA4D1045828485632024828E020000009A056E4FF0...	24.691358024691358	Passw43	Pass143
Russia	Igor Volkov	8576543210	0x003C5309CA4D1045828485632024828E020000005EF30667DC...	21.484374999999999	Passw44	Pass144
Russia	Natalia Kuznetsova	8575432109	0x003C5309CA4D1045828485632024828E020000007890806543...	24.221453287197235	Passw45	Pass145
USA	Prarthana Doe	8579876510	0x003C5309CA4D1045828485632024828E0200000085728266E...	21.258503401360546	Pass146	Pass146

----- NON CLUSTERED INDEX 1 -----

```

CREATE NONCLUSTERED INDEX IX_Treatment_Diagnosed_Illness_InstitutionID
ON Treatment (Diagnosed_Illness, InstitutionID);

```

```

SELECT *

FROM Treatment

WHERE Diagnosed_Illness = 'COVID19';

```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Servers' tree is expanded to show the 'GlobalHealthCareSystem' database. The 'Tables' folder is selected. In the center, a query window is open with the following SQL code:

```

1393 CREATE NONCLUSTERED INDEX IX_Treatment_Diagnosed_Illness_InstitutionID
1394 ON Treatment (Diagnosed_Illness, InstitutionID);
1395
1396
1397
1398 SELECT *
1399 FROM Treatment
1400 WHERE Diagnosed_Illness = 'COVID19';

```

Below the query window, the 'Results' tab is active, displaying a grid of 15 rows and 8 columns. The columns are: TreatmentID, PatientID, InstitutionID, DoctorID, Cost, Description, Date, and Diagnosed_Illness. The data shows various COVID-19 diagnostic check-ups performed at different institutions and by different doctors.

TreatmentID	PatientID	InstitutionID	DoctorID	Cost	Description	Date	Diagnosed_Illness
1	501	101	201	500	COVID-19 Diagnostic Check-up	2023-05-20	COVID19
2	503	103	203	700	COVID-19 Diagnostic Check-up	2023-03-22	COVID19
3	506	106	206	900	COVID-19 Diagnostic Check-up	2023-06-25	COVID19
4	509	109	209	900	COVID-19 Diagnostic Check-up	2022-12-28	COVID19
5	514	102	214	500	COVID-19 Diagnostic Check-up	2023-04-04	COVID19
6	517	105	217	450	COVID-19 Diagnostic Check-up	2023-12-07	COVID19
7	520	108	220	650	COVID-19 Diagnostic Check-up	2022-12-10	COVID19
8	521	104	216	350	COVID-19 Diagnostic Check-up	2023-12-06	COVID19
9	523	106	218	150	COVID-19 Diagnostic Check-up	2023-12-08	COVID19
10	524	109	210	550	COVID-19 Diagnostic Check-up	2022-12-09	COVID19
11	525	111	208	650	COVID-19 Diagnostic Check-up	2022-12-10	COVID19
12	526	111	201	500	COVID-19 Diagnostic Check-up	2023-05-20	COVID19
13	527	121	216	350	COVID-19 Diagnostic Check-up	2023-12-06	COVID19
14	532	123	216	350	COVID-19 Diagnostic Check-up	2023-12-02	COVID19
15	537	129	216	350	COVID-19 Diagnostic Check-up	2023-12-08	COVID19

---- NON CLUSTERED INDEX 2 ----

CREATE NONCLUSTERED INDEX IX_Treatment_Diagnosis_Date

ON Treatment (Diagnosed_Illness, Date);

SELECT *

FROM Treatment

WHERE Diagnosed_Illness = 'COVID19' and Date = '03/22/2023';

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SERVERS' tree is expanded to show the 'GlobalHealthCareSystem' database, with the 'Tables' folder selected. The main window displays a SQL query in the 'SQLQuery_1.sql' file. The query is as follows:

```
604  
605  
606  
607 CREATE NONCLUSTERED INDEX IX_Treatment_Diagnosis_Date  
608 ON Treatment (Diagnosed_Illness, Date);  
609  
610 SELECT *  
611 FROM Treatment  
612 WHERE Diagnosed_Illness = 'COVID19' and Date = '03/22/2023';  
613  
614  
615
```

Below the query, the 'Results' tab shows a single row of data:

	TreatmentID	PatientID	InstitutionID	DoctorID	Cost	Description	Date	Diagnosed_Illness
1	503	103	3	203	700	COVID-19 Diagnostic Check-up	2023-03-22	COVID19

The status bar at the bottom indicates the current position is Ln 632, Col 1, with 1 row returned.

---- NON CLUSTERED INDEX 3 ----

CREATE NONCLUSTERED INDEX IX_Visit_DoctorID

ON Visit (DoctorID);

SELECT *

FROM Visit

WHERE DoctorID = 203;

The screenshot shows a SQL Server Enterprise Manager interface. The query window displays the following SQL code:

```

1480 ----- NON CLUSTERED INDEX 3 -----
1481
1482 CREATE NONCLUSTERED INDEX IX_Visit_DoctorID
1483 ON Visit (DoctorID);
1484
1485 SELECT
1486 FROM Visit
1487 WHERE DoctorID = 203;
1488
1489

```

The results pane shows the following data:

	VisitID	DoctorID	PatientID	InstitutionID	Visit_Date	Visit_Time
1	403	203	103	3	2022-12-03	10:30:00
2	418	203	102	3	2022-12-19	06:53:32
3	424	203	101	7	2023-02-10	17:30:12

---- TO VIEW NUMBER OF NON CLUSTERED INDEX ----

SELECT

t.name AS TableName,

i.name AS IndexName,

i.type_desc AS IndexType,

col.name AS ColumnName

FROM

sys.indexes AS i

INNER JOIN

sys.index_columns AS ic ON i.object_id = ic.object_id AND i.index_id = ic.index_id

INNER JOIN

sys.columns AS col ON ic.object_id = col.object_id AND ic.column_id = col.column_id

INNER JOIN

sys.tables AS t ON i.object_id = t.object_id

WHERE

i.type_desc = 'NONCLUSTERED';

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SERVERS' tree is expanded to 'GlobalHealthCareSystem' > 'Tables'. The main pane displays a SQL query with line numbers 634 to 649. The query is as follows:

```
634
635 SELECT
636     t.name AS TableName,
637     i.name AS IndexName,
638     i.type_desc AS IndexType,
639     col.name AS ColumnName
640 FROM
641     sys.indexes AS i
642 INNER JOIN
643     sys.index_columns AS ic ON i.object_id = ic.object_id AND i.index_id = ic.index_id
644 INNER JOIN
645     sys.columns AS col ON ic.object_id = col.object_id AND ic.column_id = col.column_id
646 INNER JOIN
647     sys.tables AS t ON i.object_id = t.object_id
648 WHERE
649     i.type_desc = 'NONCLUSTERED';
```

Below the query, the 'Results' tab shows a table with 5 rows and 4 columns: TableName, IndexName, IndexType, and ColumnName.

	TableName	IndexName	IndexType	ColumnName
1	Visit	IX_Visit_DoctorID	NONCLUSTERED	DoctorID
2	Treatment	IX_Treatment_Diagnosed_Illness_InstitutionID	NONCLUSTERED	Diagnosed_Illness
3	Treatment	IX_Treatment_Diagnosed_Illness_InstitutionID	NONCLUSTERED	InstitutionID
4	Treatment	IX_Treatment_Diagnosis_Date	NONCLUSTERED	Diagnosed_Illness
5	Treatment	IX_Treatment_Diagnosis_Date	NONCLUSTERED	Date

The status bar at the bottom indicates 'Ln 652, Col 6', 'Spaces: 4', 'UTF-8', 'LF', 'SQL', '5 rows', 'Choose SQL Language', '00:00:00', and 'localhost: GlobalHealthCareSystem'.