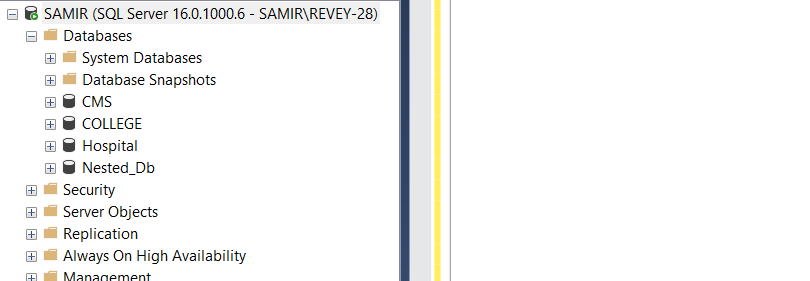
**Lab-2:Use of Multiple Table**

1. Create Database CMS.

Query: CREATE DATABASE CMS ;



1. Create Multiple Tables : Department, Student, Staff, Subjects and Marks.

Query:CREATE TABLE Department(Did INT NOT NULL identity (1,1) PRIMARY KEY ,

Dname VARCHAR(20),

Db\_no INT );

CREATE TABLE Student(Sid INT NOT NULL identity (20,1) PRIMARY KEY ,

Sname VARCHAR(20),

Address VARCHAR(20),

Dob DATE ,

Did INT,

FOREIGN KEY (Did) references Department(Did) );

CREATE TABLE Staff(Staff\_id INT NOT NULL identity (40,1) PRIMARY KEY ,

Staff\_name VARCHAR(20),

Did INT,

FOREIGN KEY (Did) references Department(Did) );

CREATE TABLE Subjects ( Sub\_id VARCHAR(10) PRIMARY KEY ,

Sub\_name VARCHAR(20),

Credit\_hr INT,

Staff\_id INT

FOREIGN KEY (Staff\_id) references Staff(Staff\_id) );

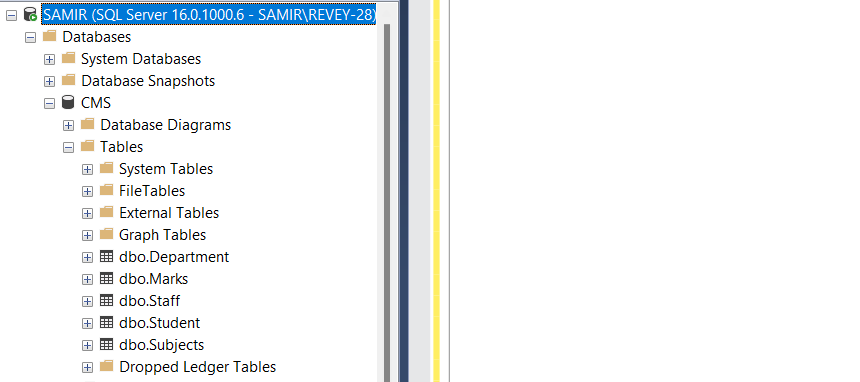
CREATE TABLE Marks(Obatained\_marks INT,

Sub\_id VARCHAR(10),

Sid INT,

FOREIGN KEY (Sub\_id) references Subjects(Sub\_id),

FOREIGN KEY (Sid) references Student(Sid) );



1. Insert any 10/10 records to each of the tables through GUI and display.

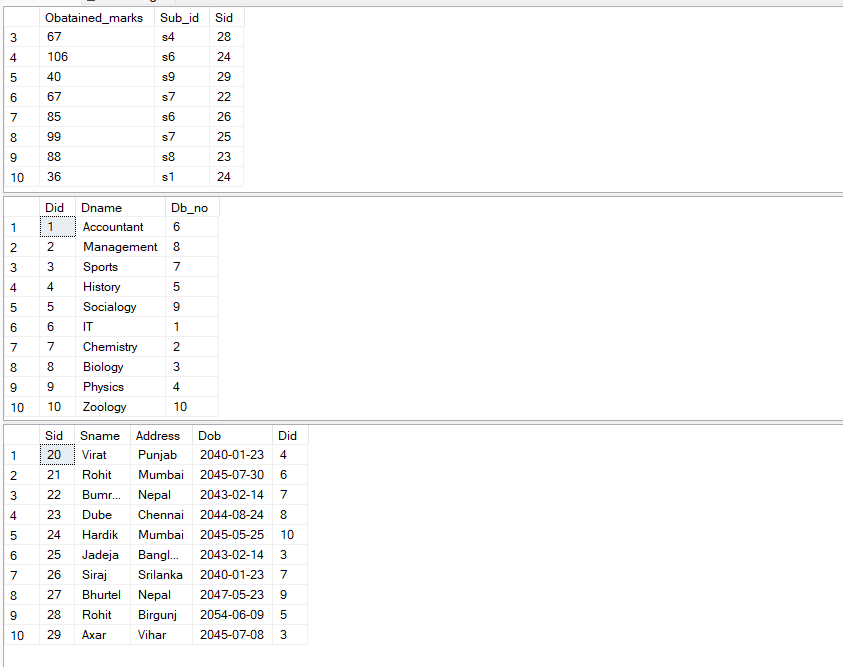
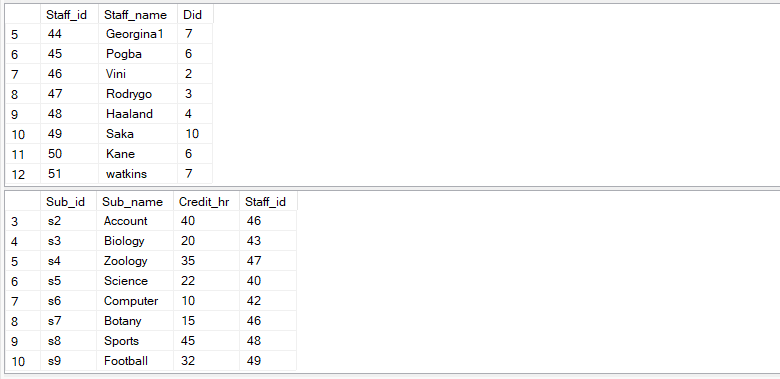
Query : SELECT \* FROM Marks

SELECT \* FROM Department

SELECT \* FROM Student

SELECT \* FROM Staff

SELECT \* FROM Subjects



4.Display records of those student who get maximum marks.

Query: SELECT \*

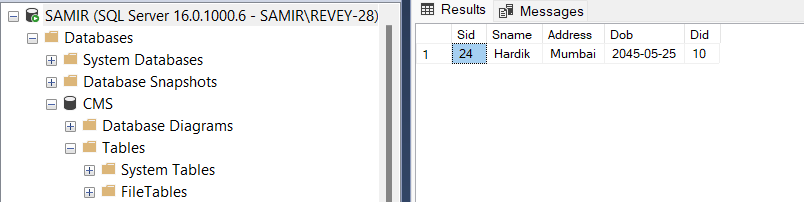
FROM Student AS s

WHERE s.sid IN (SELECT m.sid

FROM Marks AS m

WHERE Obatained\_marks In (SELECT MAX(Obatained\_marks)

FROM Marks ) );



5.Find name and marks of all students who get minimum marks.

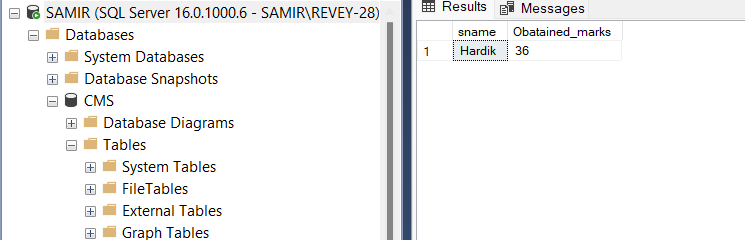
Query: SELECT s.sname , m.Obatained\_marks

FROM Student AS s INNER JOIN Marks AS m

ON s.sid=m.sid

WHERE Obatained\_marks IN (SELECT MIN(Obatained\_marks)

FROM Marks);



6.Display list of subjects learned by student of dob less than 2045-10-10.

Query: SELECT sub.Sub\_id,sub.Sub\_name ,sub.credit\_hr ,sub.Staff\_id , s.sname

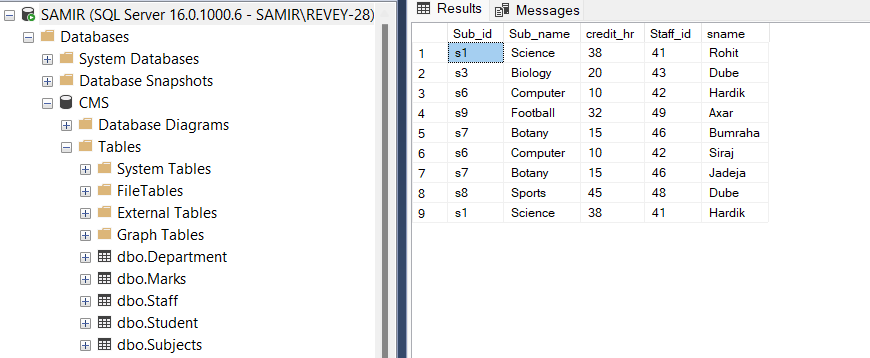
FROM Subjects AS sub INNER JOIN Marks AS m

ON sub.sub\_id=m.sub\_id

INNER JOIN Student AS s

ON s.sid =m.sid

WHERE s.Dob < '2045-10-10' ;



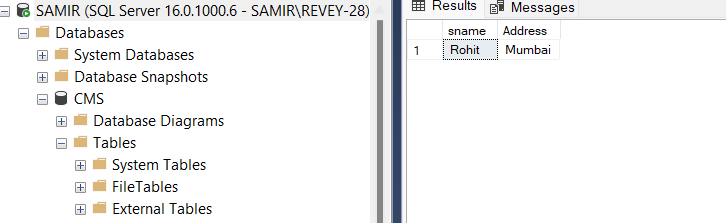
7.Display name of all students of department 'IT' or of address start with 'k'

Query: SELECT s.sname , s.Address

FROM Department AS d INNER JOIN Student AS s

ON d.Did=s.Did

WHERE s.Address LIKE 'K%' OR d.Dname = 'IT';



8.Increase marks of all students of address 'Mumbai'by 20%.

Query : SELECT s.Sname, s.Address ,m.Obatained\_marks

FROM Student s INNER JOIN Marks AS m

ON s.sid =m.sid

WHERE s.Address ='Mumbai' ;

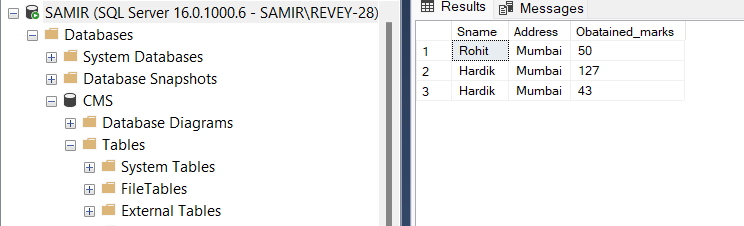
UPDATE Marks

SET Obatained\_marks =Obatained\_marks + 0.2 \* Obatained\_marks

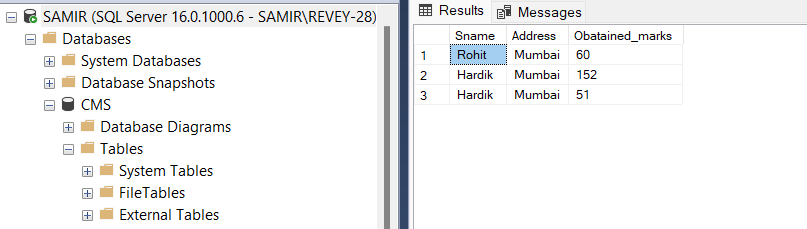
WHERE Sid IN (SELECT sid

FROM student

WHERE Address ='Mumbai');



Before



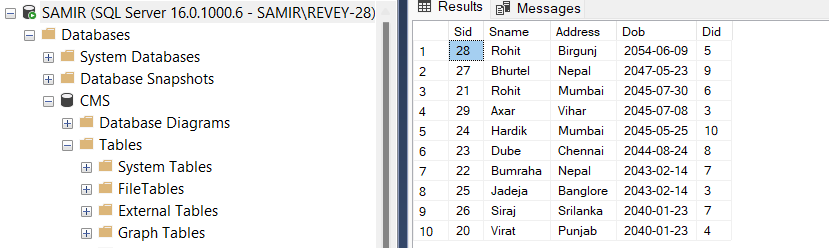
After

9.Display record of all student in descending order of their dob.

Query: SELECT \*

FROM Student s

ORDER by s.Dob DESC;

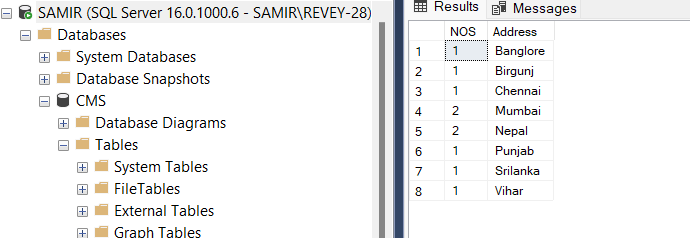


10.Display total no.of student and their address in every address level.

Query: SELECT COUNT (sid) AS NOS ,Address

FROM Student

GROUP BY (Address);



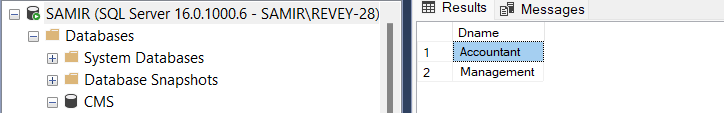
11.Display all department with no students.

Query: SELECT Dname

FROM Student s FULL OUTER JOIN Department d

ON s.did=d.did

WHERE s.did is NULL;



12.Display records of all student of address start with 'B' and get greater or equal than average marks.

Query: SELECT \*

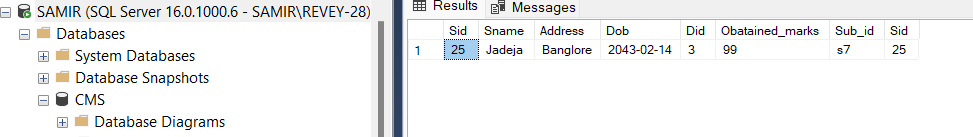
FROM Student s INNER JOIN Marks m

ON s.sid=m.sid

WHERE s.Address LIKE 'B%' AND

Obatained\_marks >= (SELECT AVG(Obatained\_marks)

FROM Marks);

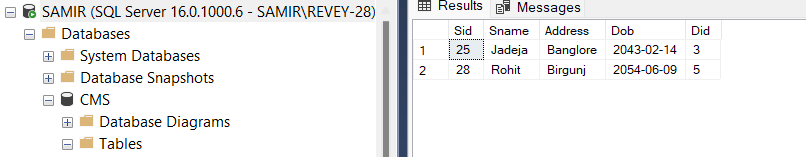


13.Display 5 oldest student of address start with 'B'

Query: SELECT TOP 5\*

FROM student s

WHERE s.ADDRESS LIKE 'B%'

ORDER BY Dob ASC ;

14.Increase the credit hr of all subjects of name contains letter 's' and study by student of address 'Mumbai' by 2 hr.

Query: UPDATE Subjects

SET Credit\_hr = Credit\_hr +2

FROM Student s INNER JOIN Marks AS m

ON s.sid =m.sid

INNER JOIN Subjects AS sub

ON sub.Sub\_id =m.Sub\_id

WHERE sub.Sub\_name LIKE '%S%' AND s.Address ='Mumbai' ;

SELECT s.Sname, s.Address,sub.Credit\_hr

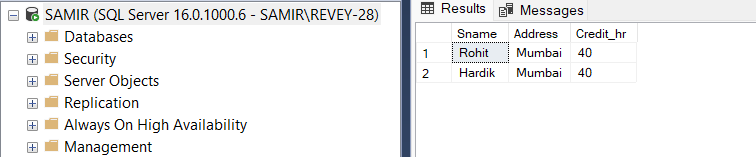
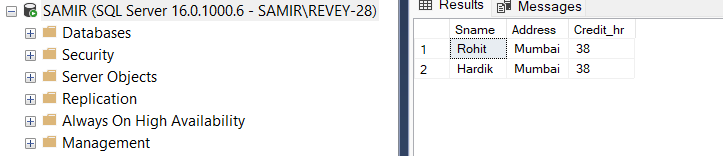
FROM Student s INNER JOIN Marks AS m

ON s.sid =m.sid

INNER JOIN Subjects AS sub

ON sub.Sub\_id =m.Sub\_id

WHERE sub.Sub\_name LIKE '%S%' AND s.Address ='Mumbai' ;



After

Before

15.Display the record all staffs who not help to any students.

Query: SELECT st.Staff\_name

FROM Student s RIGHT JOIN Marks AS m

ON s.sid =m.sid

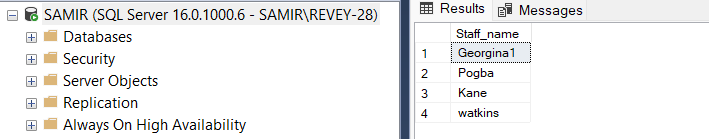
RIGHT JOIN Subjects AS sub

ON sub.Sub\_id =m.Sub\_id

Right JOIN Staff AS st

ON st.Staff\_id =sub.Staff\_id

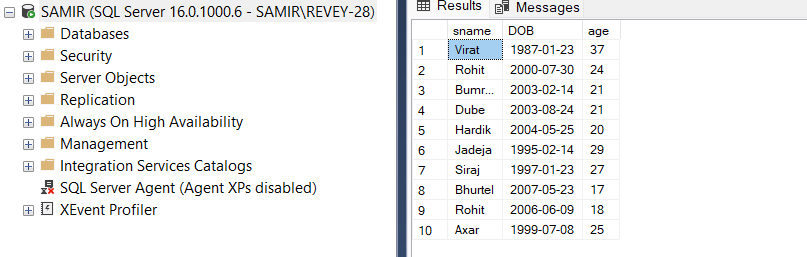
WHERE sub.Staff\_id is NULL ;



16.Find out current age from dob of all students.

Query :SELECT sname, DOB, DATEDIFF (year,DOB,GETDATE()) AS age

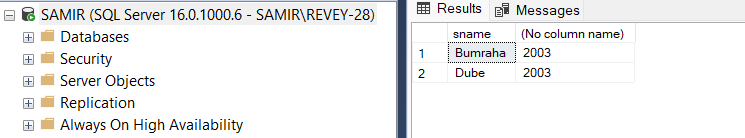
FROM Student ;



17.Display only those student whose dob contain 2003 yr.

Query : SELECT sname, year(DOB)

FROM Student

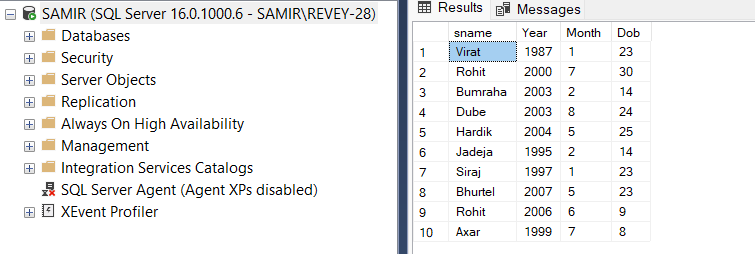
WHERE year(DOB) = '2003'

18.Display year,month and day of all students and their names.

Query:SELECT sname, year(DOB) AS Year,MONTH(DOB) AS Month ,

DAY(DOB) AS Dob

FROM Student ;



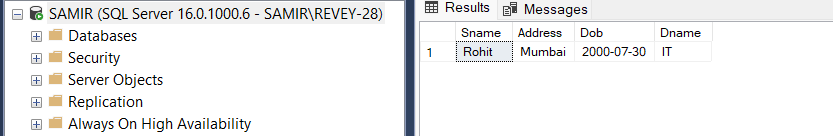
19.Display all student who associated with 'IT' department.

Query : SELECT Sname,Address,Dob,Dname

FROM Department AS d INNER JOIN Student AS s

ON d.Did=s.Did

WHERE d.Dname = 'IT';



20.Find join of above 5-tables.

Query: SELECT \*

FROM Student s INNER JOIN Marks AS m

ON s.sid =m.sid

INNER JOIN Subjects AS sub

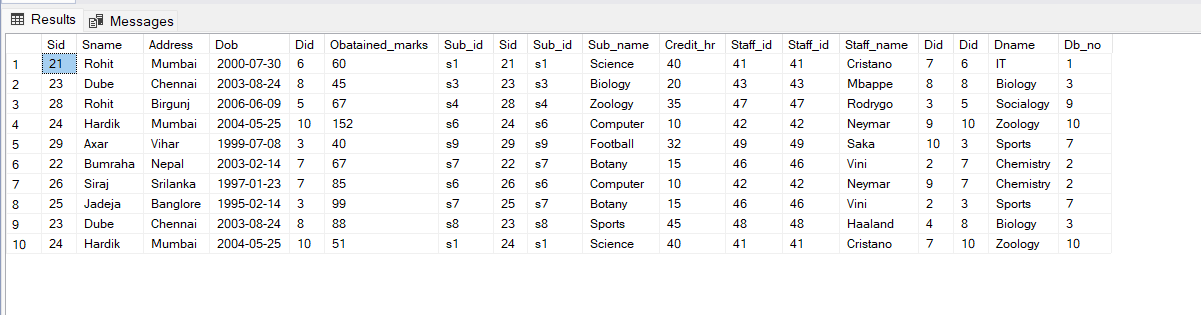
ON sub.Sub\_id =m.Sub\_id

INNER JOIN Staff AS st

ON st.Staff\_id =sub.Staff\_id

INNER JOIN Department AS d

ON d.Did =s.Did ;



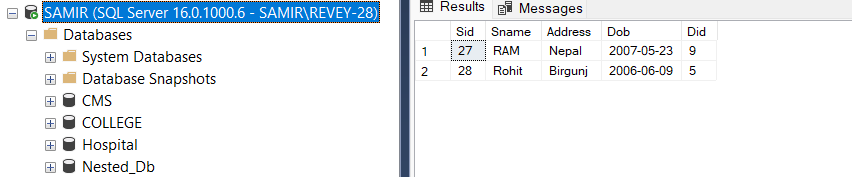
**Lab-3:Use of View**

1.Create a view ‘student\_view’ that display all student of age less than 20.

Query : SELECT sname ,dob,DATEDIFF (year,DOB,GETDATE()) AS age

FROM Student

WHERE DATEDIFF (year,DOB,GETDATE()) < 20 ;



2.Create a view 'Student\_subjects' that display all student who takes 'BOTANY' subject.

Query : CREATE VIEW student\_subjects AS

SELECT sub.Sub\_id,sub.Sub\_name, s.Sname, s.Sid

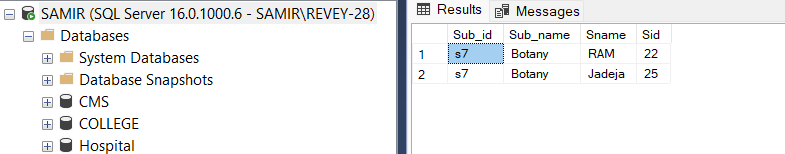
FROM Student s INNER JOIN Marks AS m

ON s.sid =m.sid

INNER JOIN Subjects AS sub

ON sub.Sub\_id =m.Sub\_id

WHERE Sub\_name ='BOTANY' ;



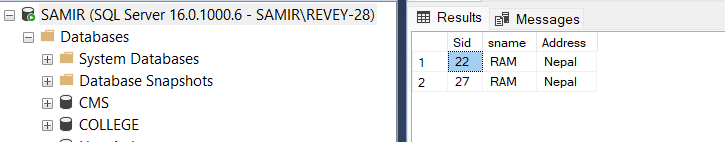
3.Create a view 'student details' that contain sid,sname and address of those student of address NEPAL.

Query : CREATE VIEW student\_Details AS

SELECT Sid, sname,Address

FROM Student

WHERE address ='Nepal'



4.Insert any 3 additional records to student\_details view.

Query:INSERT into student\_Details VALUES('samir', 'kathmandu' ),

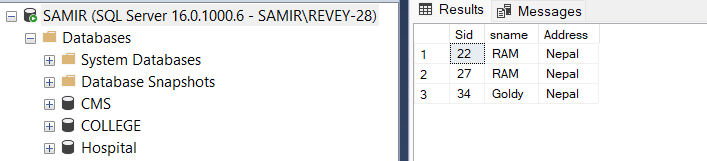
('Reven', 'Lalitpur'),

('Goldy' , 'Nepal') ;

5.Display the view ‘ student\_details’

Query: SELECT \*

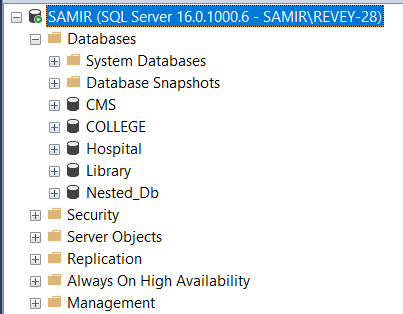
FROM student\_Details ;



**Lab-4:Use of Constraints**

1.Create database name ‘ Library ’.

Query: CREATE DATABASE Library ;



1. Create table Books,Student and Author with proper constraints.

Query:CREATE TABLE Book (ISBN VARCHAR(10) PRIMARY KEY,

bname VARCHAR (20),

price INT NOT NULL,

noP INT UNIQUE ,

sid INT,

FOREIGN KEY(sid) references Student (sid));

CREATE TABLE Student (sid INT identity (11,1) PRIMARY KEY ,

sname VARCHAR(20),

age INT CHECK(age > 0 and age < 110) ,

address VARCHAR(20) DEFAULT 'Baneshwor')

CREATE TABLE Author(aid INT identity (21,1) PRIMARY KEY ,

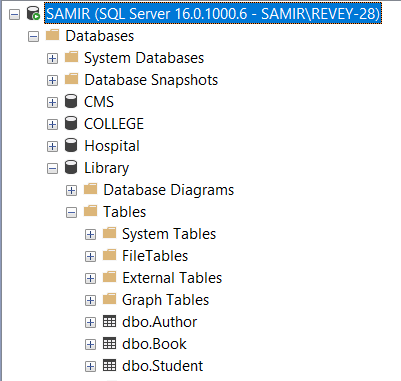
aname VARCHAR(20) NOT NULL,

Phone\_no VARCHAR(10) UNIQUE ,

address VARCHAR(20),

ISBN VARCHAR(10),

FOREIGN KEY (ISBN) references Book(ISBN));



1. Insert any 5 data into table Books,Student and Author.

Query:INSERT into Student (sname,age) VALUES('samir',20),

('saurab',18),

('sangeet',18),

('subash',19),

('santosh',21) ;

INSERT into Author VALUES('Bhupendra Singh Saud','9818345672','kathmandu','1'),

('Tej Bahadur Shahi','9876543720','Bhaktapur','3'),

('Pearson','9875436271','Lalitpur','4'),

('Indra Chaudhary','9876543768','Illam','2'),

('Dipak Bhatta','9876583456','Pokhara','1');

INSERT into Book VALUES ( '1','DBMS',500,200,11),

( '2','Operating System',450,170,12),

( '3','T0C',600,180,11),

( '4','Computer Network',400,110,14),

( '5','AI',550,199,15);

1. Display table Books,student and Author.

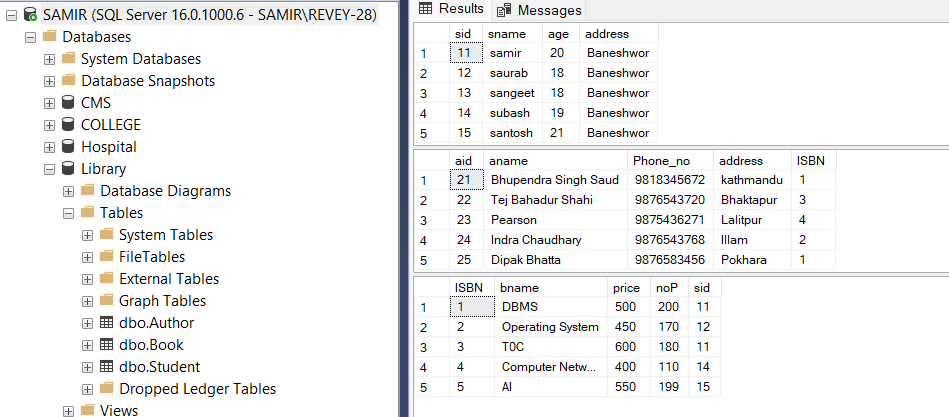
Query: SELECT \*

FROM student

SELECT \*

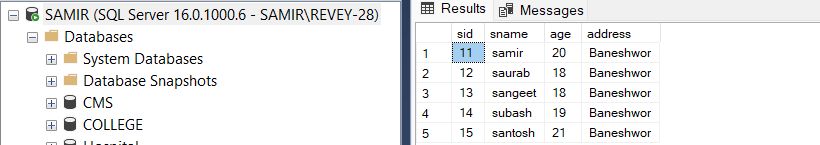
FROM Author

SELECT \*

FROM Book

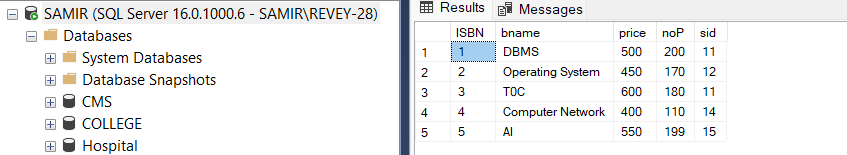
1. Test the ‘default’ constraints.

Query: address VARCHAR(20) DEFAULT 'Baneshwor' (In table : Student)

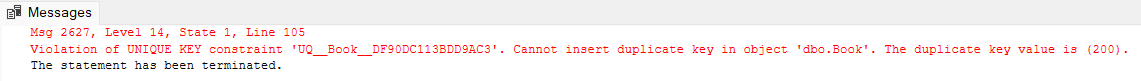


1. Test for ‘Unique’ constraint.

Query: noP INT UNIQUE (In table: Book)

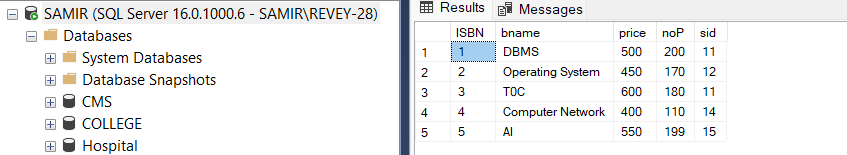


Query: INSERT into Book VALUES ( '6','Dsa',700,200,15) ;

 Here, we are unable to insert data as ‘noP’ should be unique i.e 200 is already in table.

1. Test for ‘Not Null ’ constraint.

Query: price INT NOT NULL, (In table: Book)

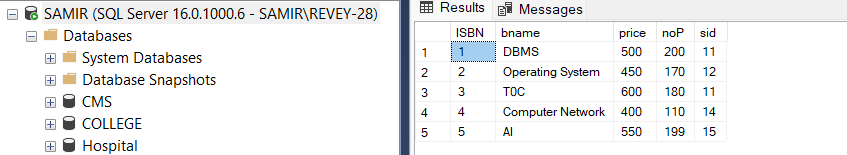


Query: INSERT into Book VALUES ( '6','Dsa',NULL,230,15);

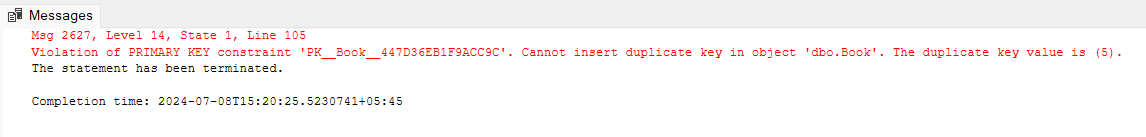
Here ,we are unable to insert data as ‘price’ should not be NUll.

1. Test for ‘Primary Key’ constraint.

Query:ISBN VARCHAR(10) PRIMARY KEy



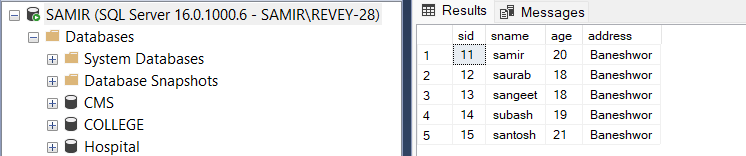
Query: INSERT into Book VALUES ( '5','Dsa',240,230,15)

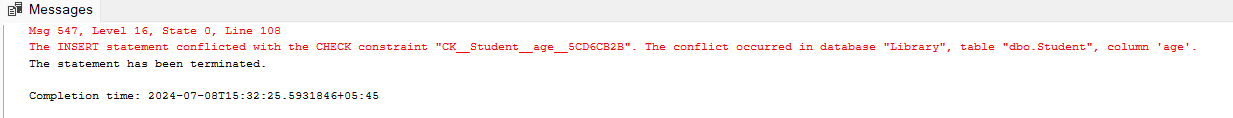


Here , we are unable to insert data as Primary Key cant be same i.e 5 is already in table.

1. Test for ‘Check’ constraint.

Query: age INT CHECK(age > 0 and age < 110) (In table: Student)



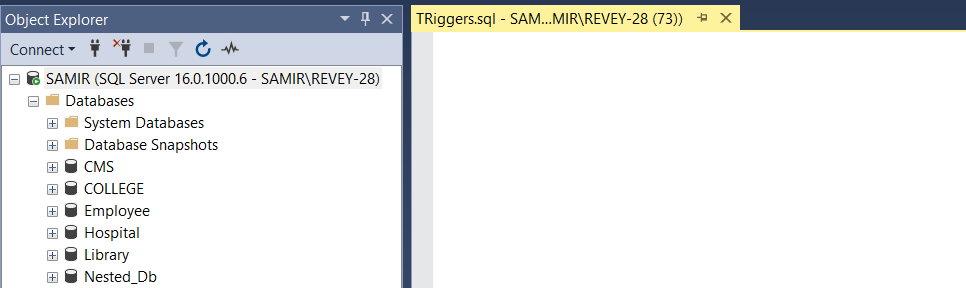
Query:INSERT into Student(sname,age) VALUES ('Ram',-20);

Here, we are unable to insert data with age -20 as age must be 0 < age < 110 .

**Lab-5:Use of Triggers**

1.Create database name ‘Employee’

Query: CREATE DATABASE Employee;



2.Create table Employee,Employee\_log and Total\_salary with proper constraints.

Query:CREATE TABLE Employee

(eid INT NOT NULL PRIMARY KEY ,

ename VARCHAR(20),

salary FLOAT,

Address VARCHAR(20) );

CREATE TABLE Employee\_log

(eid INT,

ename VARCHAR(20),

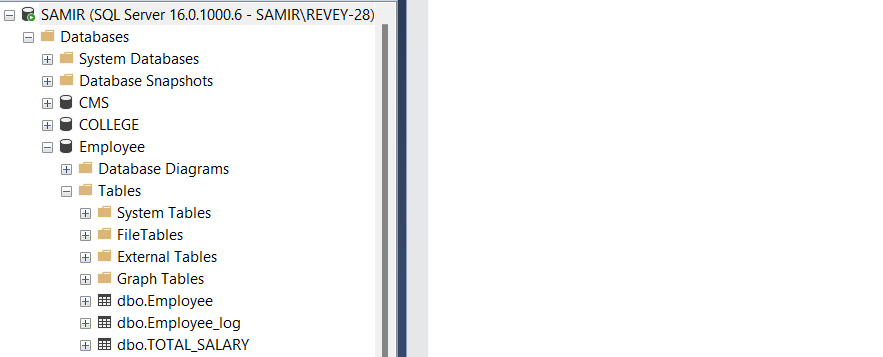
old\_salary FLOAT,

new\_salary FLOAT,

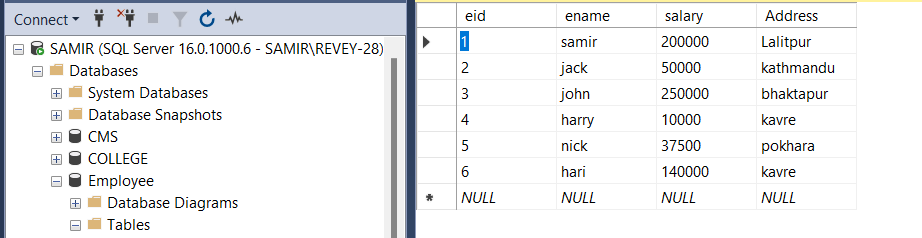
date\_time DATETIME );

CREATE TABLE TOTAL\_SALARY

( salary\_sum FLOAT );



3.Insert any 5 data into Employee table through GUI.



4.Create a trigger to find total sum of salary and store to total\_salary table.

Query:CREATE trigger total\_salary\_update

ON Employee

AFTER INSERT,UPDATE,DELETE

AS

BEGIN

declare @total float

SELECT @total=SUM(salary)

FROM employee ;

UPDATE total\_salary

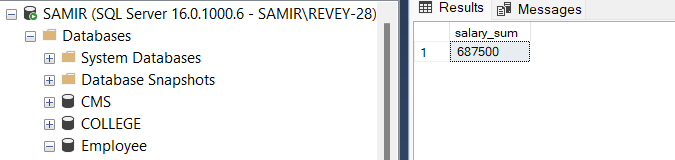
SET salary\_sum=@total;

END

5.Display total\_salary table after activation of trigger total\_salary\_update.

Query:SELECT\*

FROM TOTAL\_SALARY



6.create trigger Employee\_log\_update .

Query:CREATE trigger Employee\_Log\_update

ON Employee

AFTER UPDATE

AS

BEGIN

Insert into Employee\_log(eid,ename,old\_Salary,new\_salary,date\_time)

SELECT deleted.eid,deleted.ename,deleted.salary AS old\_salary,inserted.salary AS new\_salary,GETDATE()

FROM inserted

JOIN deleted

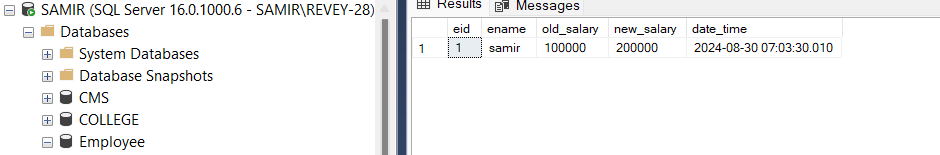
on inserted.eid=deleted.eid

END ;

7.Display Employee\_log table after activation of trigger Employee\_Log\_Update.

Query: SELECT\*

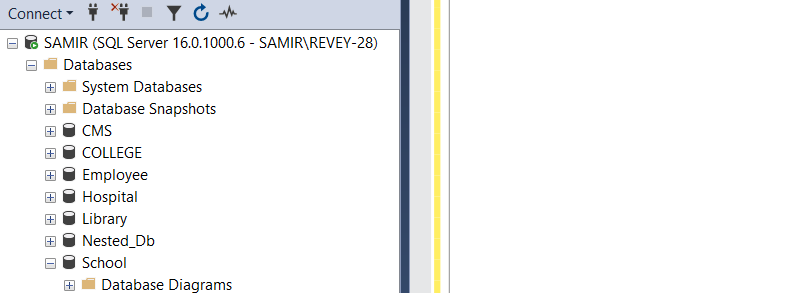
FROM Employee\_log



**Lab-6:Use of Store Procedure**

1.Create database name ‘School’

Query: CREATE DATABASE School;



2.Create table Teacher and Student with proper constraints.

Query:CREATE TABLE Teacher

(tid INT NOT NULL PRIMARY KEY ,

tname VARCHAR(20),

salary FLOAT,

Address VARCHAR(20));

CREATE TABLE Student

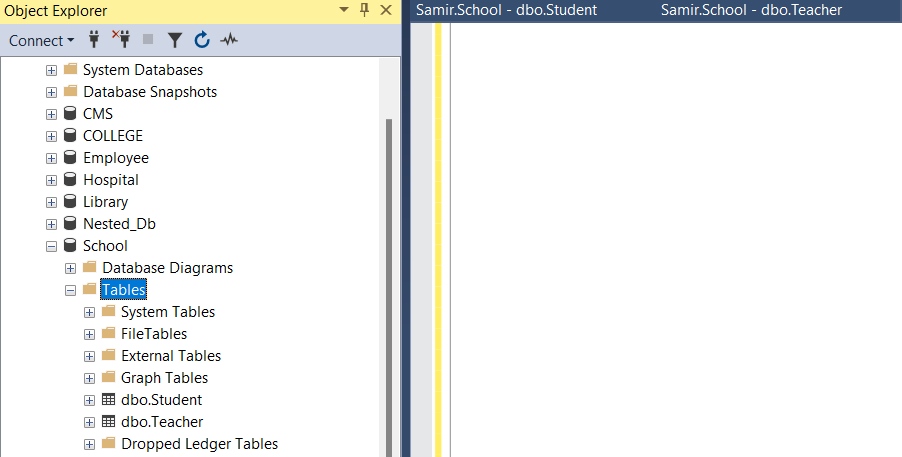
(sid INT,

sname VARCHAR(20),

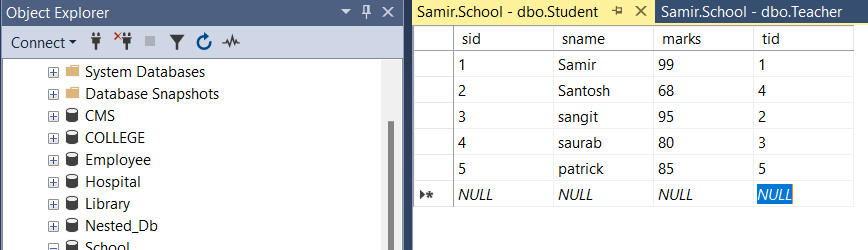
marks FLOAT,

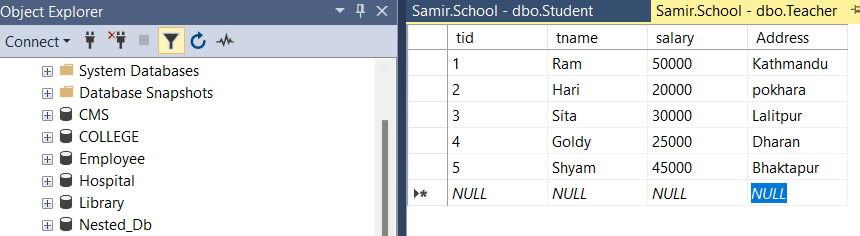
tid INT,

FOREIGN KEY (tid) references Teacher(tid) );



3.Insert any 5 data into Teacher and Student table through GUI.





4.Create a Stored Procedure teacher\_student that retrieves data by joining the Teacher and Student tables based on their tid (teacher ID).

Query: CREATE procedure teacher\_student

AS

BEGIN

SELECT tname,address,sname,marks

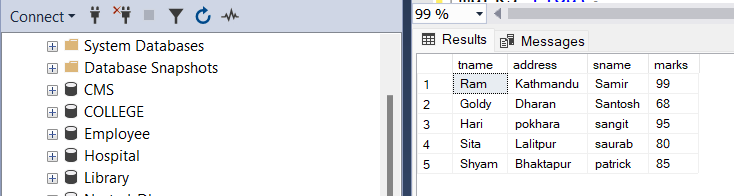
FROM Teacher t inner join Student s

on t.tid = s.tid

END

5.Display procedure teacher\_student.

Query:EXEC teacher\_student;



E