Databases
Seminary 6

Model for the practical exam

1hour 15 minutes

PART I

Let R be a table in a SQL Server database with schema R[FK1, FK2, C1, C2, C3, C4, C5]. The primary key is {FK1, FK2}. Answer questions 1-3 using the legal instance below (each question has at least one correct answer).

FK1	FK2	C1	C2	C3	C4	C5
1	1	Pisica pe acoperisul fierbinte	Tennessee Williams	100	20	AB
1	2	Conul Leonida fata cu reactiunea	Ion Luca Caragiale	50	50	CQ
1	3	Concert din muzica de Bach	Hortensia Papadat-Bengescu	50	10	QC
2	1	Fata babei si fata mosneagului	Ion Creanga	100	100	QM
2	2	Frumosii nebuni ai marilor orase	Fanus Neagu	10	10	BA
2	3	Frumoasa calatorie a ursilor panda povestita de un saxofonist care avea o iubita la Frankfurt	Matei Visniec	100	20	MQ
3	1	Mansarda la Paris cu vedere spre moarte	Matei Visniec	200	10	PQ
3	2	Richard al III-lea se interzice sau Scene din viata lui Meyerhold	Matei Visniec	100	50	PQ
3	3	Masinaria Cehov. Nina sau despre fragilitatea pescarusilor impaiati	Matei Visniec	100	100	AZ
4	1	Omul de zapada care voia sa intalneasca soarele	Matei Visniec	100	100	CP
4	2	Extraterestrul care isi dorea ca amintire o pijama	Matei Visniec	50	10	CQ
4	3	O femeie draguta cu o floare si ferestre spre nord	Edvard Radzinski	10	100	CP
4	4	Trenul din zori nu mai opreste aici	Tennessee Williams	200	200	MA

```
-- drop table R

create table R(
FK1 int not null,
FK2 int not null,
C1 varchar(100) not null,
C2 varchar(50) not null,
C3 int,
C4 int,
C5 varchar(20),
CONSTRAINT pk_R PRIMARY KEY(FK1, FK2))
```

```
insert into R(FK1, FK2, C1, C2, C3, C4, C5) values
(1, 1, 'Pisica pe acoperisul fierbinte', 'Tennessee Williams', 100, 20, 'AB'),
(1, 2, 'Conul Leonida fata cu reactiunea', 'Ion Luca Caragiale', 50, 50, 'CQ'),
(1, 3, 'Concert din muzica de Bach', 'Hortensia Papadat-Bengescu', 50, 10, 'QC'),
(2, 1, 'Fata babei si fata mosneagului', 'Ion Creanga', 100, 100, 'QM'),
(2, 2, 'Frumosii nebuni ai marilor orase', 'Fanus Neagu', 10, 10, 'BA'),
(2, 3, 'Frumoasa calatorie a ursilor panda povestita de un saxofonist care avea o iubita la Frankfurt', 'Matei Visniec', 100, 20, 'MQ'),
(3, 1, 'Mansarda la Paris cu vedere spre moarte', 'Matei Visniec', 200, 10, 'PQ'),
(3, 2, 'Richard al III-lea se interzice sau Scene din viata lui Meyerhold', 'Matei Visniec', 100, 50, 'PQ'),
(3, 3, 'Masinaria Cehov. Nina sau despre fragilitatea pescarusilor impaiati', 'Matei Visniec', 100, 100, 'AZ'),
```

(4, 1, 'Omul de zapada care voia sa intalneasca soarele', 'Matei Visniec', 100, 100, 'CP'),
(4, 2, 'Extraterestrul care isi dorea ca amintire o pijama', 'Matei Visniec', 50, 10, 'CQ'),
(4, 3, 'O femeie draguta cu o floare si ferestre spre nord', 'Edvard Radzinski', 10, 100, 'CP'),
(4, 4, 'Trenul din zori nu mai opreste aici', 'Tennessee Williams', 200, 200, 'MA')

1. Consider query Q below:

```
SELECT C2, SUM(C3) TotalC3, AVG(C3) AvgC3
FROM R
WHERE C3 >= 100 OR C1 LIKE '%Pisica%'
GROUP BY C2
HAVING SUM(C3) > 100
```

a. Q returns 3 records and value *Matei Visniec* is in its result set.
b. Q returns 3 records and value *Matei Visniec* is not in its result set.
c. Q returns 2 records and value *Ion Creanga* is not in its result set.
d. Q returns 2 records and value *Ion Creanga* is in its result set.
e. None of the above answers is correct.

```
SELECT C2, SUM(C3) TotalC3, AVG(C3) AvgC3
FROM R
WHERE C3 >= 100 OR C1 LIKE '%Pisica%'
GROUP BY C2
HAVING SUM(C3) > 100
```

Results | Messages

	C2	TotalC3	AvgC3
1	Matei Visniec	600	120
2	Tennessee Williams	300	150

Answer: c

2. How many records does the following query return?

```
SELECT *
FROM
(SELECT FK1, FK2, C3+C4 TotalC3C4 FROM R
WHERE FK1 = FK2) r1
INNER JOIN (SELECT FK1, FK2, C5
FROM R
WHERE C5 LIKE '%Q%') r2 ON r1.FK1 = r2.FK1 AND r1.FK2 = r2.FK2
```

a. 2
b. 8
c. 0
d. 1
e. None of the above answers is correct.

```sql
SELECT * FROM
(SELECT FK1, FK2, C3+C4 TotalC3C4 FROM R
WHERE FK1 = FK2) r1
INNER JOIN (SELECT FK1, FK2, C5
FROM R
WHERE C5 LIKE '%Q%') r2 ON r1.FK1 = r2.FK1 AND r1.FK2 = r2.FK2
```

Results | Messages

FK1	FK2	TotalC3C4	FK1	FK2	C5

Answer: c

3. Table R has a single trigger defined on it:

CREATE OR ALTER TRIGGER TrOnUpdate
ON R
FOR UPDATE AS
DECLARE @total INT = 0
SELECT @total = SUM(i.C3 - d.C3)
FROM deleted d INNER JOIN inserted i ON d.FK1 = i.FK1 AND d.FK2 = i.FK2 WHERE d.C3 < i.C3
PRINT @total

What's the value returned by the PRINT statement in the trigger when the UPDATE below is executed?

UPDATE R
SET C3 = 300
WHERE FK1 < FK2

a. 550
b. 700
c. 650
d. 600
e. None of the above answers is correct.

```sql
CREATE OR ALTER TRIGGER TrOnUpdate
ON R
FOR UPDATE AS
DECLARE @total INT = 0
SELECT @total = SUM(i.C3 - d.C3)
FROM deleted d INNER JOIN inserted i ON d.FK1 = i.FK1 AND d.FK2 = i.FK2 WHERE d.C3 < i.C3
PRINT @total

-- What's the value returned by the PRINT statement in the trigger when the UPDATE below :

UPDATE R
SET C3 = 300
WHERE FK1 < FK2
```

```
Messages
700

(3 rows affected)

Completion time: 2020-12-04T03:51:24.6786305-08:00
```

Answer: b

PART II

Create a database to manage train schedules. The database will store data about the routes of all the trains.

a) The entities of interest to the problem domain are: *Trains*, *Train Types*, *Stations* and *Routes*.
b) Each train has a name and belongs to a type. The train type has only a description.
c) Each station has a name.
d) Station names are unique.
e) Each route has a name, an associated train, and a list of stations with arrival and departure times in each station.
f) Route names are unique.
g) The arrival and departure times are represented as hour:minute pairs, e.g., train arrives at 5pm and leaves at 5:10pm.

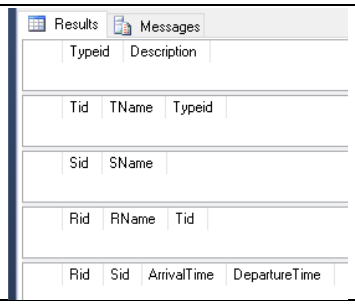
1) Write an SQL script that creates the corresponding relational data model. (4p)
2) Implement a stored procedure that receives a route, a station, arrival and departure times, and adds the station to the route. If the station is already on the route, the arrival and departure times are updated. (1p)
3) Create a view that shows the names of the routes that pass through all the stations. (2p)
4) Implement a function that lists the names of the stations with more than ***R*** routes, where ***R>=1*** is a function parameter. (2p)
(1p of)

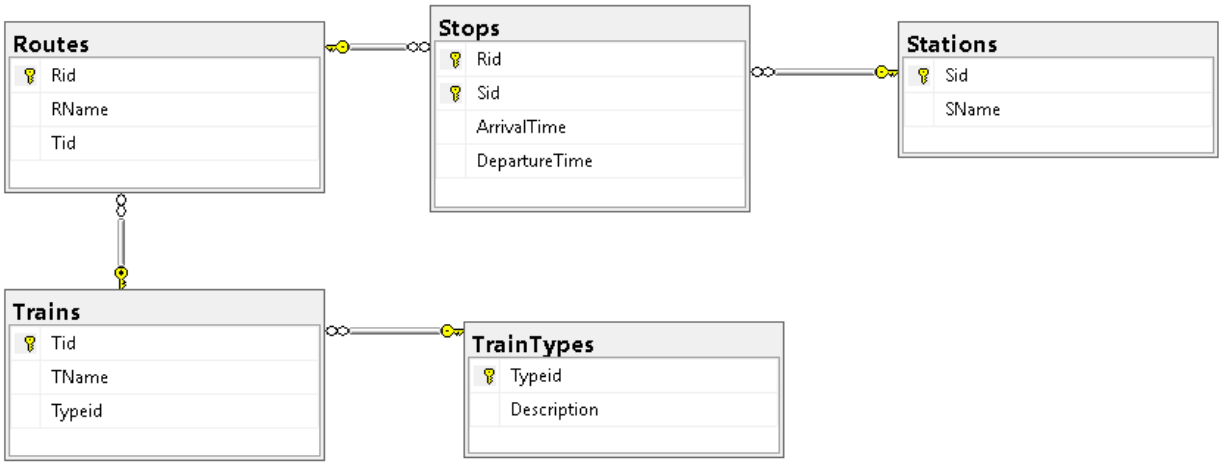
Solution:

```
-- 1) Write an SQL script that creates the corresponding relational data model.      (4p)

drop database PE_IE
go
create database PE_IE
go
use PE_IE
go

create table Stations(
       Sid int primary key identity(1,1),
       SName VARCHAR(50) unique
)

create table TrainTypes(
       Typeid int primary key identity(1,1),
       Description VARCHAR(50)
)

create table Trains
```

```
(
	Tid int primary key identity(1,1),
	TName varchar(50),
	Typeid int foreign key references TrainTypes(Typeid)
)

create table Routes(
	Rid int primary key identity(1,1),
	RName varchar(50) unique,
	Tid int foreign key references Trains(Tid)
)

create table Stops( -- RoutesStations
	Rid int foreign key references Routes(Rid),
	Sid int foreign key references Stations(Sid),
	ArrivalTime time,
	DepartureTime time
	CONSTRAINT pk_Stops PRIMARY KEY(Rid, Sid)
)

GO
```

```
select * from TrainTypes
select * from Trains
select * from Stations
select * from Routes
select * from Stops
```






```
INSERT INTO TrainTypes VALUES('description 1'), ('descrition 2')
INSERT INTO Trains values ('InterRegio', 1), ('Intercity', 1), ('Regio', 2)
INSERT INTO Stations values ('Cluj-Napoca'), ('Brasov'), ('Bucuresti')
Insert into Routes values ('Sighisoara', 1), ('Medias', 2)
INSERT Stops VALUES(1,1,'12:00:00', '18:00:00'), (1,2,'15:30:00', '22:42:00'),
(2,2,'08:05:00', '21:48:00')
GO
```

```
select * from TrainTypes
select * from Trains
select * from Stations
select * from Routes
select * from Stops
```

Results | Messages

	Typeid	Description
1	1	description 1
2	2	descrition 2

	Tid	TName	Typeid
1	1	InterRegio	1
2	2	Intercity	1
3	3	Regio	2

	Sid	SName
1	1	Cluj-Napoca
2	2	Brasov
3	3	Bucuresti

	Rid	RName	Tid
1	1	Sighisoara	1
2	2	Medias	2

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	2	2	08:05:00.0000000	21:48:00.0000000

```
-- 2) Implement a stored procedure that receives a route, a station, arrival and departure times and adds the station to the route. If the station is already on the route, the arrival and departure times are updated. (1p)

go
create proc Add_Stops @Rid int, @Sid int, @at time, @dt time
AS
	DECLARE @nr int;
	SET @nr = 0;
	SELECT @nr = COUNT(*) FROM Stops WHERE Rid=@Rid and Sid=@Sid

	IF(@nr<>0) BEGIN
		UPDATE Stops
		SET ArrivalTime=@at, DepartureTime=@dt
		WHERE Rid=@Rid and Sid=@Sid
	END
	ELSE BEGIN
		INSERT INTO Stops VALUES (@Rid, @Sid, @at, @dt)
	END
Go
```

```
-- insert
select * from Stops
EXEC Add_Stops 2,1, '5:00:00', '9:00:00'
select * from Stops
```

Results | Messages

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	2	2	08:05:00.0000000	21:48:00.0000000

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	2	1	05:00:00.0000000	09:00:00.0000000
4	2	2	08:05:00.0000000	21:48:00.0000000

```
-- already inserted
select * from Stops
EXEC Add_Stops 2,1, '15:00:00', '19:00:00'
select * from Stops
```

```
(4 row(s) affected)
```

Results | Messages

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	2	1	05:00:00.0000000	09:00:00.0000000
4	2	2	08:05:00.0000000	21:48:00.0000000

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	2	1	15:00:00.0000000	19:00:00.0000000
4	2	2	08:05:00.0000000	21:48:00.0000000

```sql
-- second version with names for Stations and Routes
create proc Add_RoutesStations_Names @RName varchar(50), @SName varchar(50), @at time, @dt time -
- Add_Stops
AS
	DECLARE @nr int;
	DECLARE @Rid int;
	DECLARE @Sid int;
	SET @nr = 0;

	SELECT @Rid=Rid FROM Routes WHERE RName=@RName
	SELECT @Sid=Sid FROM Stations WHERE SName=@SName

	--SELECT @nr = COUNT(*) FROM Stops WHERE Rid=@Rid and Sid=@Sid
	SELECT @nr = COUNT(*) FROM RoutesStations WHERE Rid=@Rid and Sid=@Sid


	IF(@nr<>0) BEGIN
		UPDATE RoutesStations --Stops
		SET ArrivalTime=@at, DepartureTime=@dt
		WHERE Rid=@Rid and Sid=@Sid
	END
	ELSE BEGIN
		-- INSERT INTO Stops VALUES (@Rid, @Sid, @at, @dt)
		INSERT INTO RoutesStations VALUES (@Rid, @Sid, @at, @dt)
	END
go
```

```sql
-- insert
-- select * from Stops
select * from RoutesStations
-- EXEC Add_Stops 2,3, '5:00:00', '9:00:00'
EXEC Add_RoutesStations_Names 'Medias','Bucuresti',
'5:00:00', '9:00:00'
-- select * from Stops
select * from RoutesStations

-- delete from RoutesStations where Rid=2 and Sid=3
```

Results | Messages

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	1	3	12:00:00.0000000	18:00:00.0000000
4	2	1	05:00:00.0000000	09:00:00.0000000
5	2	2	08:05:00.0000000	21:48:00.0000000

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	1	3	12:00:00.0000000	18:00:00.0000000
4	2	1	05:00:00.0000000	09:00:00.0000000
5	2	2	08:05:00.0000000	21:48:00.0000000
6	2	3	05:00:00.0000000	09:00:00.0000000

```sql
-- already inserted
-- select * from Stops
select * from RoutesStations
-- EXEC Add_Stops 2,3, '5:00:00', '9:00:00'
EXEC Add_RoutesStations_Names 'Medias','Bucuresti',
'15:00:00', '19:00:00'
-- select * from Stops
select * from RoutesStations
-- delete from RoutesStations where Rid=2 and Sid=3
```

Results | Messages

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	2	1	15:00:00.0000000	19:00:00.0000000
4	2	2	08:05:00.0000000	21:48:00.0000000
5	2	3	05:00:00.0000000	09:00:00.0000000

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	2	1	15:00:00.0000000	19:00:00.0000000
4	2	2	08:05:00.0000000	21:48:00.0000000
5	2	3	15:00:00.0000000	19:00:00.0000000

```sql
-- third version with all validations
alter procedure Add_RoutesStations_Names_Validation @RName varchar(50), @SName varchar(50), @at time,
@dt time -- Add_Stops
AS
	DECLARE @nr int;
	DECLARE @Rid int;
	DECLARE @Sid int;
	SET @nr = 0;
	SET @Rid = 0;
	SET @Sid = 0;

	SELECT @Rid=Rid FROM Routes WHERE RName=@RName
	SELECT @Sid=Sid FROM Stations WHERE SName=@SName

	--SELECT @nr = COUNT(*) FROM Stops WHERE Rid=@Rid and Sid=@Sid
	SELECT @nr = COUNT(*) FROM RoutesStations WHERE Rid=@Rid and Sid=@Sid

	IF (@Rid=0 and @Sid=0) BEGIN
		PRINT 'The Route and the Station does not exist'
		RETURN
	END
	ELSE BEGIN
	IF (@Rid=0) BEGIN
		PRINT 'The Route does not exist'
		RETURN
	END
	ELSE BEGIN
			IF (@Sid=0) BEGIN
				PRINT 'The Station does not exist'
				RETURN
			END
			ELSE BEGIN
				IF(@nr<>0 ) BEGIN
					UPDATE RoutesStations --Stops
					SET ArrivalTime=@at, DepartureTime=@dt
					WHERE Rid=@Rid and Sid=@Sid
				END
				ELSE BEGIN
					-- INSERT INTO Stops VALUES (@Rid, @Sid, @at, @dt)
					INSERT INTO RoutesStations VALUES (@Rid, @Sid, @at, @dt)
				END
```

```
                    END
        END
        END
go
```

```
-- insert
-- select * from Stops
select * from RoutesStations
-- EXEC Add_Stops 2,3, '5:00:00', '9:00:00'
EXEC Add_RoutesStations_Names_Validation
'Medias','Bucuresti', '5:00:00', '9:00:00'
-- select * from Stops
select * from RoutesStations
```

Results | Messages

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	1	3	12:00:00.0000000	18:00:00.0000000
4	2	1	05:00:00.0000000	09:00:00.0000000
5	2	2	08:05:00.0000000	21:48:00.0000000

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	1	3	12:00:00.0000000	18:00:00.0000000
4	2	1	05:00:00.0000000	09:00:00.0000000
5	2	2	08:05:00.0000000	21:48:00.0000000
6	2	3	05:00:00.0000000	09:00:00.0000000

```
-- already inserted
-- select * from Stops
select * from RoutesStations
-- EXEC Add_Stops 2,1, '15:00:00', '19:00:00'
EXEC Add_RoutesStations_Names_Validation
'Medias','Cluj-Napoca', '15:00:00', '19:00:00'
-- select * from Stops
select * from RoutesStations
```

Results | Messages

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	2	1	05:00:00.0000000	09:00:00.0000000
4	2	2	08:05:00.0000000	21:48:00.0000000

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	2	1	15:00:00.0000000	19:00:00.0000000
4	2	2	08:05:00.0000000	21:48:00.0000000

```
-- no Routes
-- select * from Stops
select * from RoutesStations
-- EXEC Add_Stops 2,1, '15:00:00', '19:00:00'
EXEC Add_RoutesStations_Names_Validation
'Pitesti','Cluj-Napoca', '15:00:00', '19:00:00'
-- select * from Stops
select * from RoutesStations
```

```
(4 row(s) affected)
The Route does not exist

(4 row(s) affected)
```

Results | Messages

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	2	1	15:00:00.0000000	19:00:00.0000000
4	2	2	08:05:00.0000000	21:48:00.0000000

	Rid	Sid	ArrivalTime	DepartureTime
1	1	1	12:00:00.0000000	18:00:00.0000000
2	1	2	15:30:00.0000000	22:42:00.0000000
3	2	1	15:00:00.0000000	19:00:00.0000000
4	2	2	08:05:00.0000000	21:48:00.0000000

```
-- no Stations
-- select * from Stops
select * from RoutesStations
-- EXEC Add_Stops 2,1, '15:00:00', '19:00:00'
EXEC Add_RoutesStations_Names_Validation
'Medias','Oradea', '15:00:00', '19:00:00'
-- select * from Stops
select * from RoutesStations
```

```
(4 row(s) affected)
The Station does not
exist

(4 row(s) affected)
```

```
-- no Routes and no Stations
-- select * from Stops
select * from RoutesStations
-- EXEC Add_Stops 2,1, '15:00:00', '19:00:00'
EXEC Add_RoutesStations_Names_Validation
'Pitesti','Oradea', '15:00:00', '19:00:00'
-- select * from Stops
select * from RoutesStations
```

```
(4 row(s) affected)
The Route and the
Station does not exist

(4 row(s) affected)
```

```
-- 3) Create a view that shows the names of the routes that contain all the stations.
(2p)
```

```
CREATE VIEW vRoutesStations
       AS
       SELECT RName
       FROM Routes r INNER JOIN Stops ss ON r.Rid = ss.Rid
       GROUP BY RName
       HAVING COUNT(*) = (SELECT COUNT(*) FROM Stations)
```

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

```
SELECT * FROM vRoutesStations
-- nothing is returned, because we have 3
Stations and no Routes has all the 3
stations
```

Results | Messages

RName

```
-- add the left stations for a route
INSERT RoutesStations
VALUES(1,3,'12:00:00', '18:00:00') -- for
route Sighisoara

SELECT * FROM vRoutesStations -- so, we
have route Sighisoara with all the 3
stations
```

Results | Messages

	RName
1	Sighisoara

```
-- 4) Create a function that lists the names of the stations with more than R routes,
where R>=1 is a function parameter. (2p)

CREATE FUNCTION uf_StationsRoutes(@r int)
RETURNS TABLE
AS
RETURN
       SELECT DISTINCT s.Sid, SName, count(SName) as NoOfRoutes
       FROM Stations s INNER JOIN Stops ss ON  ss.Sid=s.Sid
       -- INNER JOIN Routes r ON  r.Rid=ss.Rid
       group by s.Sid, SName
       having count(SName)>=@r
       -- no need of s.Sid, when SNane is UNIQUE
go
```

```
SELECT * FROM uf_StationsRoutes(1)
SELECT * FROM uf_StationsRoutes(2)
SELECT * FROM uf_StationsRoutes(3)
```

Results | Messages

	Sid	SName	NoOfRoutes
1	1	Cluj-Napoca	2
2	2	Brasov	2

	Sid	SName	NoOfRoutes
1	1	Cluj-Napoca	2
2	2	Brasov	2

Sid	SName	NoOfRoutes

```
-- or
SELECT DISTINCT SName, count(SName) as NoOfRoutes
FROM Stations s INNER JOIN Stops ss ON  ss.Sid=s.Sid
group by SName
having count(SName)>=2

-- or
SELECT SName, count(SName) as NoOfRoutes
FROM Stations s INNER JOIN Stops ss ON  ss.Sid=s.Sid -- Stops
group by SName
having count(SName)>=2
```