Reading:

- chapter 8: “Stored Procedures and User-Defined Functions” (Petkovic)
- chapter 7: “Advanced SQL” Routines (Hoffer, Ramesh, & Topi)

Homework:

FIGURE 7-15 **Adult literacy program (for Problems and Exercises 6–14)**

TUTOR (TutorID, CertDate, Status)

TutorID	CertDate	Status
100	1/05/2008	Active
101	1/05/2008	Temp Stop
102	1/05/2008	Dropped
103	5/22/2008	Active
104	5/22/2008	Active
105	5/22/2008	Temp Stop
106	5/22/2008	Active

STUDENT (StudentID, Read)

StudentID	Read
3000	2.3
3001	5.6
3002	1.3
3003	3.3
3004	2.7
3005	4.8
3006	7.8
3007	1.5

MATCH HISTORY (MatchID, TutorID, StudentID, StartDate, EndDate)

MatchID	TutorID	StudentID	StartDate	EndDate
1	100	3000	1/10/2008	
2	101	3001	1/15/2008	5/15/2008
3	102	3002	2/10/2008	3/01/2008
4	106	3003	5/28/2008	
5	103	3004	6/01/2008	6/15/2008
6	104	3005	6/01/2008	6/28/2008
7	104	3006	6/01/2008	

TUTOR REPORT (MatchID, Month, Hours, Lessons)

MatchID	Month	Hours	Lessons
1	6/08	8	4
4	6/08	8	6
5	6/08	4	4
4	7/08	10	5
1	7/08	4	2

1) Create a database named ““AdultLiteracy” on your RDBMS environment. Using Figure 7-5 above, write DDL commands to create table structures for each entity above. Name your tables the following names: Tutor, Student, MatchHistory, TutorReport

```
create database AdultLiteracy;
use AdultLiteracy;

create table Tutor
(
	TutorID int not null,
	CertDate date,
	[Status] varchar(10),
	constraint tutor_pk primary key (TutorID)
);
```

```
create table Student
(
	StudentID int not null,
	[Read] decimal(2,1),
	constraint student_pk primary key (StudentID)
);

create table MatchHistory
(
	MatchID int not null,
	TutorID int,
	StudentID int,
	StartDate date,
	EndDate date,
	constraint matchhistory_pk primary key (MatchID),
	constraint matchhistory_fk1 foreign key (TutorID) references
Tutor(TutorID),
	constraint matchhistory_fk2 foreign key (StudentID) references
Student(StudentID)
);

create table TutorReport
(
	MatchID int not null,
	[Month] char(4),
	Hours int,
	Lessons int,
	constraint tutorreport_pk1 primary key (MatchID, [Month])
	);
```

2) Write SQL scripts to insert sample data from Fig 7-5 into the database.

```
insert into Tutor values(100, '1/05/2008', 'Active'), (101,'1/05/2008', 'Temp
Stop'), (102,'1/05/2008', 'Dropped'),
(103, '5/22/2008', 'Active'), (104, '5/22/2008', 'Active'), (105, '5/22/2008',
'Temp Stop'), (106, '5/22/2008', 'Active');

insert into Student values(3000,2.3),(3001,5.6), (3002, 1.3), (3003,3.3),
(3004,2.7), (3005, 4.8), (3006,7.8), (3007,1.5);

insert into MatchHistory values(1, 100, 3000, '1/10/2008', null), (2, 101, 3001,
'1/15/2008', '5/15/2008'),
(3, 102, 3002, '2/10/2008', '3/01/2008'), (4, 106, 3003, '5/28/2008', null),
(5, 103, 3004, '6/01/2008', '6/15/2008'), (6, 104, 3005, '6/01/2008',
'6/28/2008'),
(7, 104, 3006, '6/01/2008', null);

insert into TutorReport values(1, '6/08', 8, 4), (4, '6/08', 8, 6), (5, '6/08', 4,
4), (4, '7/08', 10, 5), (1, '7/08', 4, 2);
```

3) 7. Write the SQL command to add MATH SCORE to the STUDENT table.

```
ALTER table Student add MATHSCORE decimal(3,2);
```

4) 8. Write the SQL command to add SUBJECT to TUTOR. The only values allowed for SUBJECT will be Reading, Math, and ESL.

```
alter table Tutor add SUBJECT varchar(10);

alter table Tutor add constraint subject_ck check (SUBJECT in ('Reading', 'Math',
'ESL'));
```

5) 9. What do you need to do if a tutor signs up and wants to tutor in both reading and math? (Don't need to write SQL).

   Add a table named subject with subject_no and subject_name in this table, and change subject in tutor table into subject_no.

6) 10. Write the SQL command to find any tutors who have not submitted a report for July.

```
select m.TutorID

from MatchHistory m

where m.MatchID in

(

select MatchID

from TutorReport

where [Month] not like '7/%');
```

7) Where do you think student and tutor information such as name, address, phone, and e-mail should be kept? Write the necessary SQL commands to capture this information. Make up sample data to populate your newly created table.

   Create a table named person, including name, address, phone, and e-mail.

```
create table person
(
personID int not null,
name varchar(20),
address varchar(50),
phone char(10),
email varchar(20),
tutor_student_id int,
constraint person_pk primary key (personID)
);

insert into person values(1, 'lily', '100 washington st', '1111111111',
'lily@gmail.com', 3000);
insert into person values(2, 'lucy', '200 washington st', '1111111112',
'lucy@gmail.com', 3001);
insert into person values(3, 'ben', '300 washington st', '1111111113',
'ben@gmail.com', 3002);
```

```
insert into person values(4, 'da', '400 washington st', '1111111114',
'da@gmail.com', 3003);
insert into person values(5, 'ge', '500 washington st', '1111111115',
'ge@gmail.com', 3004);
insert into person values(6, 'bai', '600 washington st', '1111111116',
'bai@gmail.com', 3005);
insert into person values(7, 'jia', '700 washington st', '1111111117',
'jia@gmail.com', 3006);
insert into person values(8, 'jane', '800 washington st', '1111111118',
'jane@gmail.com', 3007);

insert into person values(9, 'peter', '100 pleasant st', '1111111122',
'peter@gmail.com', 100);
insert into person values(10, 'dan', '200 pleasant st', '1111111123',
'dan@gmail.com', 101);
insert into person values(11, 'mohs', '201 pleasant st', '1111111124',
'mohs@gmail.com', 102);
insert into person values(12, 'liu', '202 pleasant st', '1111111125',
'liu@gmail.com', 103);
insert into person values(13, 'dan', '203 pleasant st', '1111111126',
'dan@gmail.com', 104);
insert into person values(14, 'han', '204 pleasant st', '1111111127',
'han@gmail.com', 105);
insert into person values(15, 'many', '205 pleasant st', '1111111128',
'many@gmail.com', 106);
```

8) List all active students in June by name. (the names you made up above). Include the number of hours students received tutoring and how many lessons they completed. Write the SQL command.

```
select p.name, r.Hours, r.Lessons
from person p
join MatchHistory m on m.StudentID = p.tutor_student_id
join Tutor t on t.TutorID = m.TutorID
join TutorReport r on r.MatchID = m.MatchID
       where t.Status = 'Active';
```

9) Which tutors, by name, are available to tutor? Write the SQL command.

```
select p.name
from person p
join Tutor t on t.TutorID = p.tutor_student_id
       where t.Status = 'Active';
```

10) Which tutor needs to be reminded to turn in reports? Write the SQL command.

```
select p.name
from person p
join MatchHistory m on m.TutorID = p.tutor_student_id
where m.MatchID not in
(select MatchID
       from TutorReport);
```

11) Create a stored procedure that returns a result set of all tutors that are available to tutor. No input parameters required

```
CREATE PROCEDURE available_tutor
AS
select p.name
from person p
join Tutor t on t.TutorID = p.tutor_student_id
       where t.Status = 'Active';
```

12) Create a User Define Function which determine if a specific tutor is available to tutor. The function takes a TutorID as an input value and returns a scaler value of ‘Y’ or ‘N’ corresponding to the tutor availability

```
CREATE FUNCTION is_available_tutor (tid)
RETURNS char(1) AS
BEGIN
    tid in
       (select t.TutorID
       from Tutor t
       where t.Status = 'Active'
       )
    RETURN 'Y'
    tid not in
       (select t.TutorID
       from Tutor t
       where t.Status = 'Active'
       )
    RETURN 'N'
END
```