Instructions

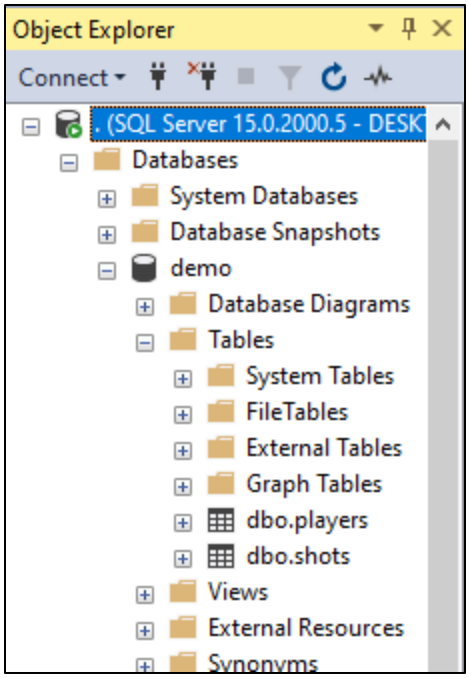
For each answer, please include your answer as text, and any screenshot(s) which demonstrate your answer was executed. Most importantly, make sure to include evidence your answer is correct. This will most likely be a screenshot. If you had issues, problems, or had to make assumptions include them in your answer.

Your Answers:

1. In the demo database, create two tables:

    a. The first table **players** should have columns player id (int pk), player name (varchar), shots attempted (int) shots made (int)
    b. The second table **shots** should have columns shot id (int pk), player id (int fk to players), clock time (datetime) shot made (bit)
    c. Add two players to the players table. Mary and Sue initialize the players with 0 shots attempted and made.

```
-- create the demo database
use master
drop database if exists demo
create database demo
use demo

-- drop tables if they exist
drop table if exists shots
drop table if exists players

-- create table 1: players
create table players (
	player_id int identity,
	player_name varchar(100),
	shots_attempted int,
	shots_made int,
	constraint PK_player_id primary key (player_id)
)

-- create table 2: shots
create table shots (
	shot_id int identity,
	player_id int,
	clock_time datetime,
	shot_made bit,
	constraint PK_shot_id primary key (shot_id),
	constraint FK_player_id foreign key (player_id) references players(player_id)
)

-- add two players to the players table
insert into players (player_name, shots_attempted, shots_made)
values ('Mary', 0, 0),
	('Sue', 0, 0)

-- check the results
select * from players
```



	player_id	player_name	shots_attempted	shots_made
1	1	Mary	0	0
2	2	Sue	0	0

2. Write transaction safe code as a stored procedure which when given a player id, clock time, and whether the shot was made (bit value) will add the record to the **shots** table and update the player record in the **players** table. For example, If Mary takes a shot and makes it, then misses the next one, there would be two records in the **shots** table and her row in the **players** table should have 2 attempt and 1 shot made. Execute the stored procedure to demonstrate the transaction is ACID compliant.

```
-- drop procedure if exists
drop procedure if exists p_record_shot
go

-- create the procedure
create procedure p_record_shot (
	@player_id int,
	@clock_time datetime,
	@shot_made bit
) as

begin try
begin transaction

	-- insert data into the shots table
	insert into shots (player_id, clock_time, shot_made)
	values (@player_id, @clock_time, @shot_made)

	-- update the player tablea accordingly
	update players set shots_attempted = shots_attempted + 1 where player_id =
@player_id
```

```
        if @shot_made = 1 update players set shots_made = shots_made + 1 where player_id =
@player_id

        -- print if it was committed or not
        print 'Committing'

        -- commit if there were no errors
        commit

end try
begin catch

        -- throw an error if it did not commit
        select error_number() as error, error_message() as message

        -- print if it is rolling back
        print 'Rolling back'

        -- roll back if there were errors
        rollback

end catch
go

-- call the stored procedure
select * from shots; select * from players -- look at the current state of the tables
exec p_record_shot @player_id = 1, @clock_time = '2021-05-01 18:30', @shot_made = 1 --
execute the stored procedure
select * from shots; select * from players -- look at the new state of the tables

select * from shots; select * from players -- look at the current state of the tables
exec p_record_shot @player_id = 1, @clock_time = '2021-05-01 18:30:00', @shot_made = 0 --
execute the stored procedure
select * from shots; select * from players -- look at the new state of the tables
```

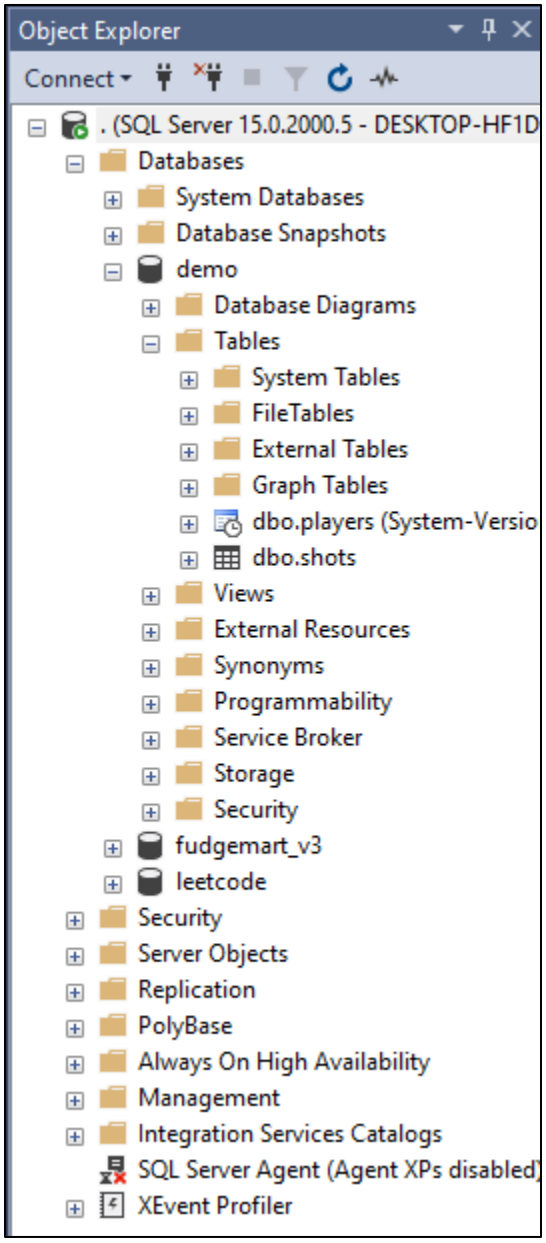
	shot_id	player_id	clock_time	shot_made
1	1	1	2021-05-01 18:30:00.000	1
2	2	1	2021-05-01 18:35:00.000	0

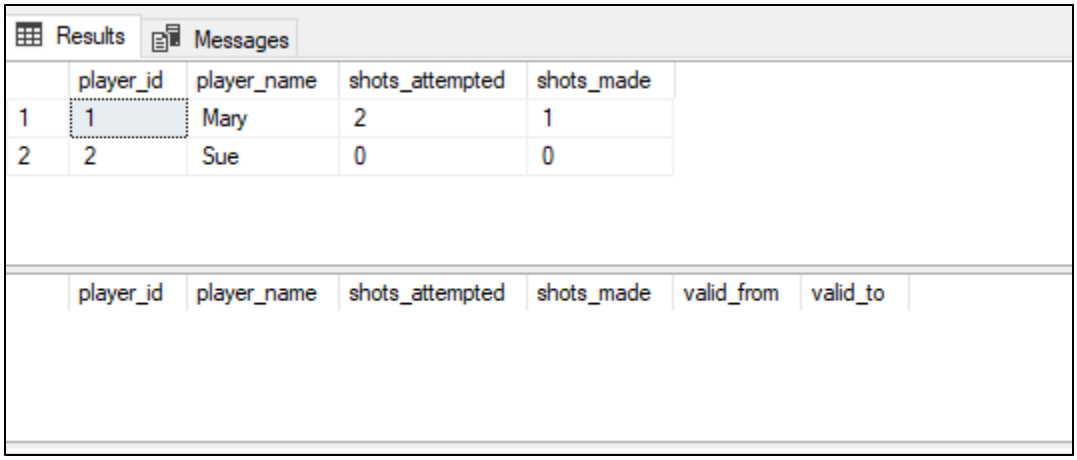
	player_id	player_name	shots_attempted	shots_made
1	1	Mary	2	1
2	2	Sue	0	0

3. Alter the **players** table to be a system-versioned temporal table.

```
alter table players
add
        valid_from datetime2 (2) generated always as row start hidden
                constraint df_valid_from default dateadd(second, -1, sysutcdatetime())
        ,valid_to datetime2 (2) generated always as row end hidden
```

```
            constraint df_valid_to default convert(datetime2, '9999.12.31
23:59:59:999')
       , period for system_time (valid_from, valid_to);
go

alter table players
       set (system_versioning = on (history_table = dbo.players_history));
go

select * from players
select * from players_history
```





Results | Messages

	player_id	player_name	shots_attempted	shots_made
1	1	Mary	2	1
2	2	Sue	0	0

player_id	player_name	shots_attempted	shots_made	valid_from	valid_to

4. Execute your stored procedure from part 2 to create at least 15 shot records over a 5-minute period. Make sure there are records in the first ½ of the 5-minute period and at few in the last minute of the 5-minute period.

```
-- initialize the shot count at 0
declare @shot_count int
set @shot_count = 0

-- initialize the game clock at some time
declare @game_clock datetime
set @game_clock = '2021-07-26 20:25:00'

-- iterate through 25 shots or 5 minutes whichever comes first
while (@shot_count <= 25 and @game_clock <= '2021-07-26 20:30:00')
begin

	-- at random either Mary or Sue takes a shot
	declare @player_id_generator int
	set @player_id_generator = (select cast(round(rand()*(2-1)+1, 0) as int))

	-- at random 10 to 15 seconds between each shot
	declare @time_elapsed int
	set @time_elapsed = (select cast(round(rand()*(15-10)+10, 0) as int))

	-- at random did they make or miss the shot
	declare @was_shot_made bit
	set @was_shot_made = (select cast(round(rand(), 0) as int))

	-- add to the parameters after each iteration
	set @shot_count = @shot_count + 1
	set @game_clock = DATEADD(second, @time_elapsed, @game_clock)

	-- execute the stored procedure
	exec p_record_shot @player_id = @player_id_generator,
	                   @clock_time = @game_clock,
	                   @shot_made = @was_shot_made

end
```

```sql
-- check that the data was recorded properly
select * from shots
```

Results | Messages

	shot_id	player_id	clock_time	shot_made
1	1	1	2021-05-01 18:30:00.0000000	1
2	2	1	2021-05-01 18:35:00.0000000	0
3	3	2	2021-07-26 20:25:13.0000000	0
4	4	2	2021-07-26 20:25:28.0000000	1
5	5	1	2021-07-26 20:25:42.0000000	0
6	6	2	2021-07-26 20:25:56.0000000	0
7	7	2	2021-07-26 20:26:08.0000000	0
8	8	1	2021-07-26 20:26:22.0000000	1
9	9	1	2021-07-26 20:26:33.0000000	1
10	10	1	2021-07-26 20:26:44.0000000	1
11	11	1	2021-07-26 20:26:59.0000000	1
12	12	2	2021-07-26 20:27:09.0000000	0
13	13	1	2021-07-26 20:27:21.0000000	0
14	14	1	2021-07-26 20:27:34.0000000	1
15	15	2	2021-07-26 20:27:48.0000000	0
16	16	2	2021-07-26 20:28:01.0000000	1
17	17	1	2021-07-26 20:28:14.0000000	1
18	18	2	2021-07-26 20:28:24.0000000	0
19	19	1	2021-07-26 20:28:39.0000000	0
20	20	2	2021-07-26 20:28:52.0000000	1
21	21	2	2021-07-26 20:29:05.0000000	0
22	22	2	2021-07-26 20:29:18.0000000	0
23	23	2	2021-07-26 20:29:30.0000000	1
24	24	1	2021-07-26 20:29:43.0000000	0
25	25	1	2021-07-26 20:29:55.0000000	0
26	26	1	2021-07-26 20:30:09.0000000	1

1. 5. Write SQL queries to show:
    a. The player statistics at the end of the 5-minute period (current statistics).
    b. The player statistics exactly 2 minutes and 30 seconds into the period.
    c. The player statistics in the last minute of the period.

```sql
-- a. show the stats for each player
select
	player_name,
	shots_attempted,
	shots_made,
	cast(cast(shots_made as float) / cast(shots_attempted as float) as decimal(10, 2))
as shots_made_percent
from
	players
```

	player_name	shots_attempted	shots_made	shots_made_percent
1	Mary	14	8	0.57
2	Sue	12	4	0.33

```sql
-- b. show the player stats at 2.5 minute mark

select
	player_name,
	sum(row_count) as shots_attempted,
	sum(cast(shot_made as int)) as shots_made,
	cast(cast(sum(cast(shot_made as int)) as float) / cast(sum(row_count) as float) as
decimal(10, 2)) as shots_made_percent
from (

		select
			player_name,
			1 as row_count,
			shot_made
		from
			shots
		join
			players on players.player_id = shots.player_id
		where
			clock_time between '2021-07-26 20:25:00' and '2021-07-26 20:27:30'

) sub
group by
	player_name
```

	player_name	shots_attempted	shots_made	shots_made_percent
1	Mary	6	4	0.67
2	Sue	5	1	0.20

```sql
-- c. show the player stats in the last minute of the period

select
	player_name,
	sum(row_count) as shots_attempted,
	sum(cast(shot_made as int)) as shots_made,
	cast(cast(sum(cast(shot_made as int)) as float) / cast(sum(row_count) as float) as
decimal(10, 2)) as shots_made_percent
from (

		select
			player_name,
			1 as row_count,
			shot_made
		from
			shots
		join
			players on players.player_id = shots.player_id
		where
```

```
                    clock_time between '2021-07-26 20:29:00' and '2021-07-26 20:30:00'

) sub
group by
       player_name
```

	player_name	shots_attempted	shots_made	shots_made_percent
1	Mary	2	0	0.00
2	Sue	3	1	0.33