Milestone 1

Create a simulated Top Dog Game with 5 dice, just like the Antique Top Dog Machine.

1. Write the code necessary to create the 6 TABLES illustrated above (DDL). Sample Code is provided for you below. These Tables do not need a Primary/Foreign Key relationship.

```
CREATE TABLE RunningTable(

RollNum INT IDENTITY(1,1),
d1 INT NULL,
d2 INT NULL,
d3 INT NULL,
d4 INT NULL,
d5 INT NULL,
SUM INT NULL,
TOTAL INT);

CREATE TABLE DICE1(
d1 INT NOT NULL
);

CREATE TABLE DICE2(
d2 INT NOT NULL
);

CREATE TABLE DICE3(
d3 INT NOT NULL
);

CREATE TABLE DICE4(
d4 INT NOT NULL
);

CREATE TABLE DICE5(
d5 INT NOT NULL
);
```

2. Populate the 6 TABLES above using appropriate INSERT INTO Statements (DML) (i.e., 1, 2, 3, 4, 5, 6). The RunningTable does not need any INSERT STATEMENTS at this time

-- Insert into table1 INSERT INTO DICE1 (d1) VALUES (1);	-- Insert into table2 INSERT INTO DICE2 (d2) VALUES (1);	-- Insert into table3 INSERT INTO DICE3 (d3) VALUES (1);	-- Insert into table4 INSERT INTO DICE4 (d4) VALUES (1);	-- Insert into table5 INSERT INTO DICE5 (d5) VALUES (1);	-- Running Table

INSERT INTO DICE1 (d1) VALUES (2);
 	
INSERT INTO DICE1 (d1) VALUES (3);
 	
INSERT INTO DICE1 (d1) VALUES (4);
 	
INSERT INTO DICE1 (d1) VALUES (5);
 	
INSERT INTO DICE1 (d1) VALUES (6);
	INSERT INTO DICE2 (d2) VALUES (2);
	
	INSERT INTO DICE2 (d2) VALUES (3);
	
	INSERT INTO DICE2 (d2) VALUES (4);
	
	INSERT INTO DICE2 (d2) VALUES (5);
	
	INSERT INTO DICE2 (d2) VALUES (6);
		INSERT INTO DICE3 (d3) VALUES (2);
		
		INSERT INTO DICE3 (d3) VALUES (3);
		
		INSERT INTO DICE3 (d3) VALUES (4);
		
		INSERT INTO DICE3 (d3) VALUES (5);
		
		INSERT INTO DICE3 (d3) VALUES (6);
			INSERT INTO DICE4 (d4) VALUES (2);
			
			INSERT INTO DICE4 (d4) VALUES (3);
			
			INSERT INTO DICE4 (d4) VALUES (4);
			
			INSERT INTO DICE4 (d4) VALUES (5);
			
			INSERT INTO DICE4 (d4) VALUES (6);
				INSERT INTO DICE5 (d5) VALUES (2);
				
				INSERT INTO DICE5 (d5) VALUES (3);
				
				INSERT INTO DICE5 (d5) VALUES (4);
				
				INSERT INTO DICE5 (d5) VALUES (5);
				
				INSERT INTO DICE5 (d5) VALUES (6);	

Milestone 2

CROSS JOIN, Random Number, and TOP FUNCTION

1. Write the code necessary to CROSS JOIN the 5 Dice TABLES (DO NOT use SELECT *).

```
SELECT d1, d2, d3, d4, d5
FROM DICE1 CROSS JOIN DICE2 CROSS JOIN DICE3 CROSS JOIN DICE4 CROSS JOIN DICE5;
```

2. How many rows are created with the CROSS JOIN?

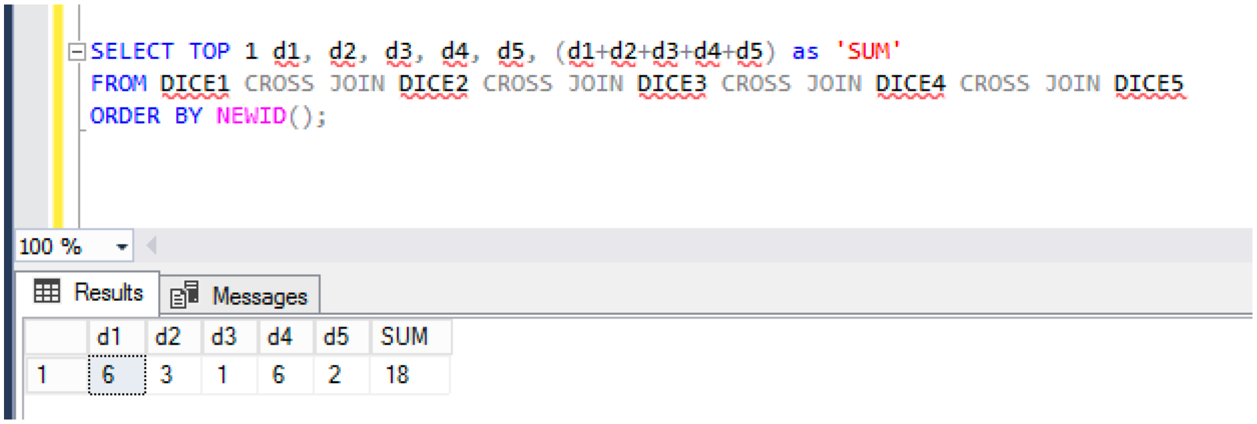
7776.

3. Create a Random Number using NEWID() in the ORDER BY Clause (ascending order).

```
SELECT d1, d2, d3, d4, d5
FROM DICE1 CROSS JOIN DICE2 CROSS JOIN DICE3 CROSS JOIN DICE4 CROSS JOIN DICE5
ORDER BY NEWID();
```

Milestone 3 – Calculated Column & CASE EXPRESSION

1. Write code to create a 6th Column that adds all 5 dice. Use an appropriate alias. Do NOT use a SUM Function for this requirement. Call this new column 'SUM'.



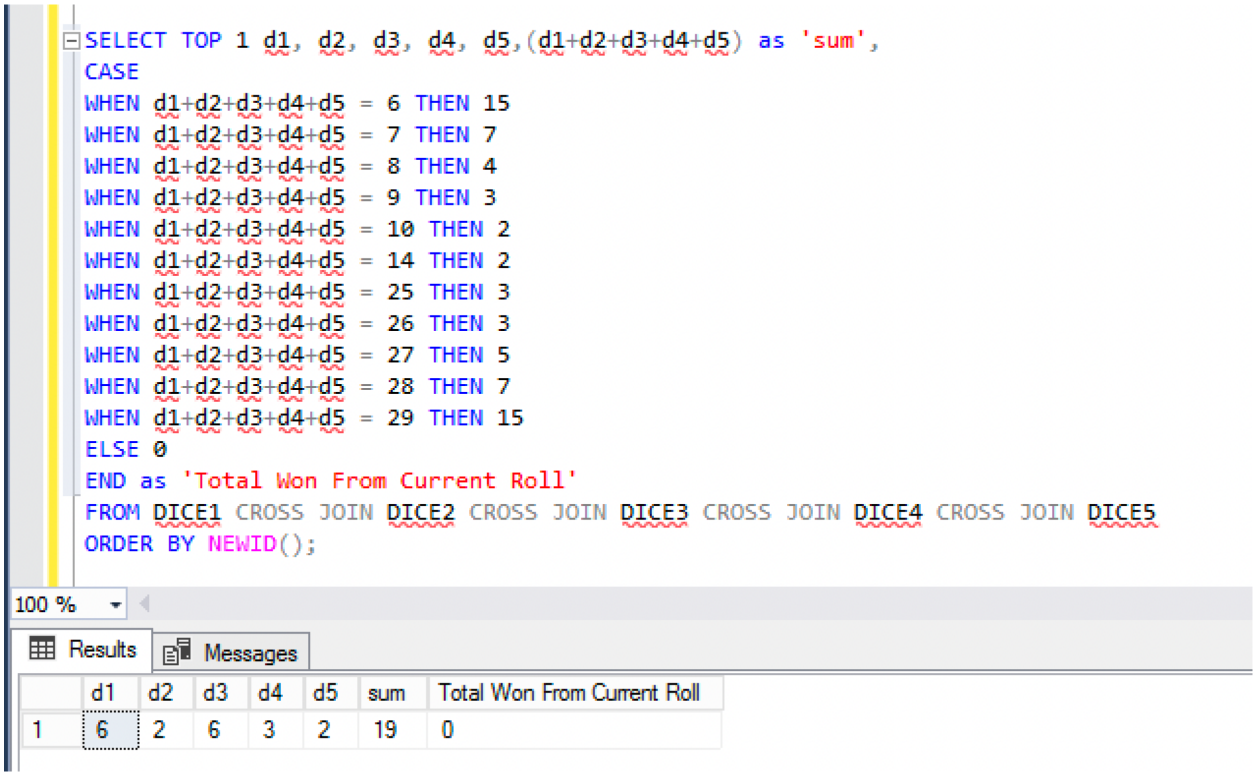
```sql
SELECT TOP 1 d1, d2, d3, d4, d5, (d1+d2+d3+d4+d5) as 'SUM'
FROM DICE1 CROSS JOIN DICE2 CROSS JOIN DICE3 CROSS JOIN DICE4 CROSS JOIN DICE5
ORDER BY NEWID();
```

100 %

Results | Messages

	d1	d2	d3	d4	d5	SUM
1	6	3	1	6	2	18

2. Write a CASE EXPRESSION to create a 7 th Column that adds all 5 dice, and then indicate how many points were won. Include an ELSE statement with 0 if nothing was won. Name the Case 'Total Won from Current Roll'.



```sql
SELECT TOP 1 d1, d2, d3, d4, d5,(d1+d2+d3+d4+d5) as 'sum',
CASE
WHEN d1+d2+d3+d4+d5 = 6 THEN 15
WHEN d1+d2+d3+d4+d5 = 7 THEN 7
WHEN d1+d2+d3+d4+d5 = 8 THEN 4
WHEN d1+d2+d3+d4+d5 = 9 THEN 3
WHEN d1+d2+d3+d4+d5 = 10 THEN 2
WHEN d1+d2+d3+d4+d5 = 14 THEN 2
WHEN d1+d2+d3+d4+d5 = 25 THEN 3
WHEN d1+d2+d3+d4+d5 = 26 THEN 3
WHEN d1+d2+d3+d4+d5 = 27 THEN 5
WHEN d1+d2+d3+d4+d5 = 28 THEN 7
WHEN d1+d2+d3+d4+d5 = 29 THEN 15
ELSE 0
END as 'Total Won From Current Roll'
FROM DICE1 CROSS JOIN DICE2 CROSS JOIN DICE3 CROSS JOIN DICE4 CROSS JOIN DICE5
ORDER BY NEWID();
```

100 %

Results | Messages

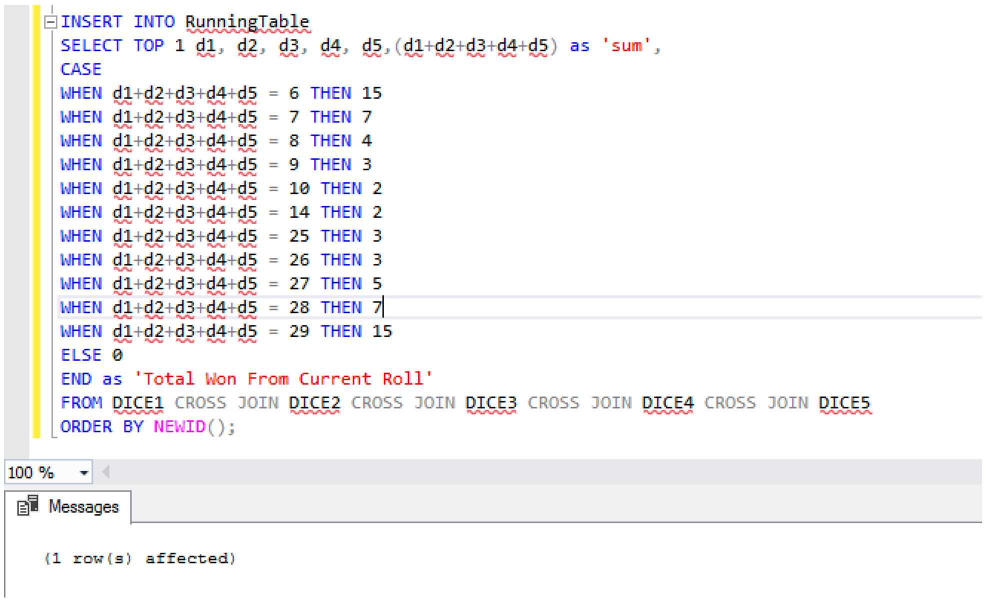
	d1	d2	d3	d4	d5	sum	Total Won From Current Roll
1	6	2	6	3	2	19	0

Milestone 4 –

INSERT INTO SELECT STATEMENT

NOTE: As an introduction to this step, the CREATE STATEMENT for the RunningTable TABLE includes 8 columns. However, the RollNum column is designed as a Surrogate Key and is ignored when inserting data into a table. As a result, the RunningTable contains 7 columns to insert data – exactly the same number of columns your current SELECT statement contains after finishing. Milestone 4.

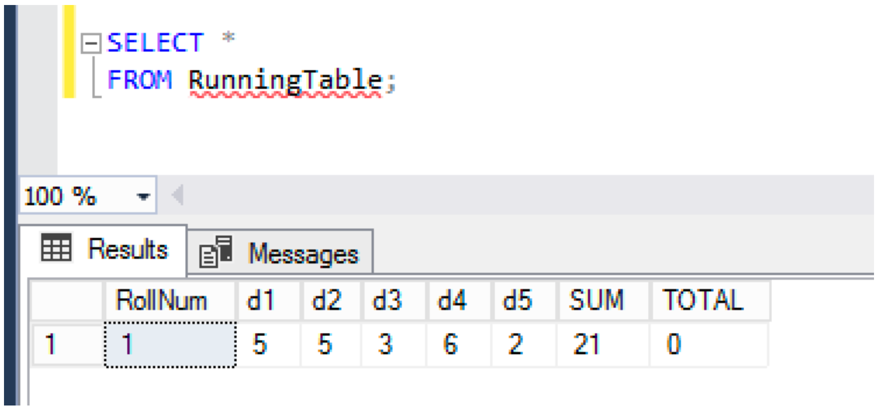
1. Just above your SELECT Statement from Milestone 3, write the following code:
INSERT INTO RunningTable



```
INSERT INTO RunningTable
SELECT TOP 1 d1, d2, d3, d4, d5,(d1+d2+d3+d4+d5) as 'sum',
CASE
WHEN d1+d2+d3+d4+d5 = 6 THEN 15
WHEN d1+d2+d3+d4+d5 = 7 THEN 7
WHEN d1+d2+d3+d4+d5 = 8 THEN 4
WHEN d1+d2+d3+d4+d5 = 9 THEN 3
WHEN d1+d2+d3+d4+d5 = 10 THEN 2
WHEN d1+d2+d3+d4+d5 = 14 THEN 2
WHEN d1+d2+d3+d4+d5 = 25 THEN 3
WHEN d1+d2+d3+d4+d5 = 26 THEN 3
WHEN d1+d2+d3+d4+d5 = 27 THEN 5
WHEN d1+d2+d3+d4+d5 = 28 THEN 7
WHEN d1+d2+d3+d4+d5 = 29 THEN 15
ELSE 0
END as 'Total Won From Current Roll'
FROM DICE1 CROSS JOIN DICE2 CROSS JOIN DICE3 CROSS JOIN DICE4 CROSS JOIN DICE5
ORDER BY NEWID();
```

```
(1 row(s) affected)
```

2. If you see the result: 1 row(s) affected, Congratulations! Open a separate 'New Query' Window and write the following:



```
SELECT *
FROM RunningTable;
```

	RollNum	d1	d2	d3	d4	d5	SUM	TOTAL
1	1	5	5	3	6	2	21	0

Milestone 5 – WINDOW FUNCTION

You have created some amazing code that replicates the Top Dog mechanical game. Your current code creates random dice results, the sum of the 5 dice, and points won, if any (Total). The following code will allow your total points won to continue to add up until you quit the game – a feature beyond the original game.

1. Return to the coding window that has all of the code through Milestone 4 (#1).
In other words, delete the "SELECT * FROM RUNNINGTABLE;" code you produced in Milestone 4 (#2)

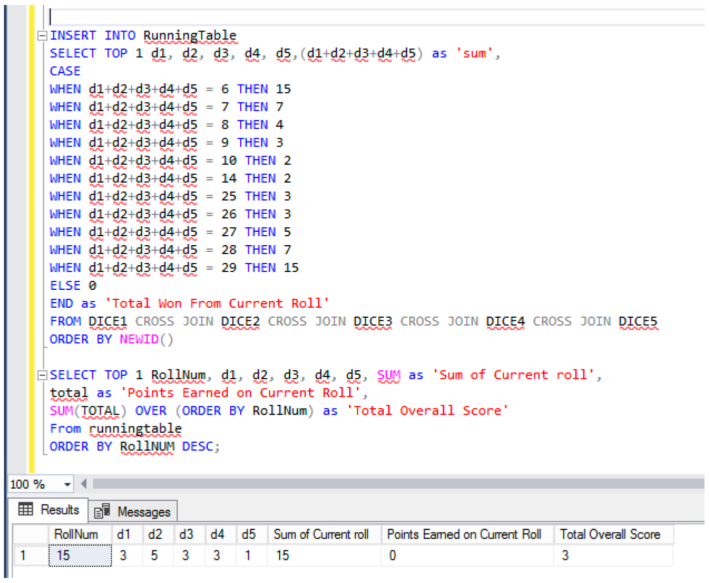
2. Below this code (Your main code with INSERT INTO RunningTable), write the following:
SELECT TOP 1 RollNum, d1, d2, d3, d4, d5, SUM as 'Sum of Current roll',
total as 'Points Earned on Current Roll',
SUM(TOTAL) OVER (ORDER BY RollNum) as 'Total Overall Score' f
rom runningtable
ORDER BY RollNUM DESC;

```
INSERT INTO RunningTable
SELECT TOP 1 d1, d2, d3, d4, d5,(d1+d2+d3+d4+d5) as 'sum',
CASE
WHEN d1+d2+d3+d4+d5 = 6 THEN 15
WHEN d1+d2+d3+d4+d5 = 7 THEN 7
WHEN d1+d2+d3+d4+d5 = 8 THEN 4
WHEN d1+d2+d3+d4+d5 = 9 THEN 3
WHEN d1+d2+d3+d4+d5 = 10 THEN 2
WHEN d1+d2+d3+d4+d5 = 14 THEN 2
WHEN d1+d2+d3+d4+d5 = 25 THEN 3
WHEN d1+d2+d3+d4+d5 = 26 THEN 3
WHEN d1+d2+d3+d4+d5 = 27 THEN 5
WHEN d1+d2+d3+d4+d5 = 28 THEN 7
WHEN d1+d2+d3+d4+d5 = 29 THEN 15
ELSE 0
END as 'Total Won From Current Roll'
FROM DICE1 CROSS JOIN DICE2 CROSS JOIN DICE3 CROSS JOIN DICE4 CROSS JOIN DICE5
ORDER BY NEWID()

SELECT TOP 1 RollNum, d1, d2, d3, d4, d5, SUM as 'Sum of Current roll',
total as 'Points Earned on Current Roll',
SUM(TOTAL) OVER (ORDER BY RollNum) as 'Total Overall Score'
From runningtable
ORDER BY RollNUM DESC;
```

3. Highlight and run all of your code (everything through Milestone 4 (#1) and the code above.

4. Run the code over and over….as you win, your 'Total Overall Score' should reflect a cumulative score increasing with each win.



```
INSERT INTO RunningTable
SELECT TOP 1 d1, d2, d3, d4, d5,(d1+d2+d3+d4+d5) as 'sum',
CASE
WHEN d1+d2+d3+d4+d5 = 6 THEN 15
WHEN d1+d2+d3+d4+d5 = 7 THEN 7
WHEN d1+d2+d3+d4+d5 = 8 THEN 4
WHEN d1+d2+d3+d4+d5 = 9 THEN 3
WHEN d1+d2+d3+d4+d5 = 10 THEN 2
WHEN d1+d2+d3+d4+d5 = 14 THEN 2
WHEN d1+d2+d3+d4+d5 = 25 THEN 3
WHEN d1+d2+d3+d4+d5 = 26 THEN 3
WHEN d1+d2+d3+d4+d5 = 27 THEN 5
WHEN d1+d2+d3+d4+d5 = 28 THEN 7
WHEN d1+d2+d3+d4+d5 = 29 THEN 15
ELSE 0
END as 'Total Won From Current Roll'
FROM DICE1 CROSS JOIN DICE2 CROSS JOIN DICE3 CROSS JOIN DICE4 CROSS JOIN DICE5
ORDER BY NEWID()

SELECT TOP 1 RollNum, d1, d2, d3, d4, d5, SUM as 'Sum of Current roll',
total as 'Points Earned on Current Roll',
SUM(TOTAL) OVER (ORDER BY RollNum) as 'Total Overall Score'
From runningtable
ORDER BY RollNUM DESC;
```

	RollNum	d1	d2	d3	d4	d5	Sum of Current roll	Points Earned on Current Roll	Total Overall Score
1	15	3	5	3	3	1	15	0	3

5. To start a game over, you would just TRUNCATE the RunningTable:

TRUNCATE TABLE RunningTable