

- Business Challenge: Create a simulated Top Dog Game with 5 dice, just like the Antique Top Dog Machine.
 - The Top Dog machine has a mechanical system that spins the round base containing the dice, providing a randomized result of 5 different dice. Point assignments are provided on the front of the machine.

- Coding Elements to Solve Lab 5
 - CREATE TABLE/INSERT INTO
 - SELECT, FROM, ORDER BY
 - ADDING COLUMNS
 - ORDER BY including TOP Function
 - CASE EXPRESSION (Type 2 - SEARCHED) [see game image at bottom of assignment for winning possibilities]
 - Random Number Generator NEWID()
 - CROSS JOIN
 - INSERT INTO SELECT STATEMENT
 - Using Alias to improve COLUMN Readability

- Tables you will create (Milestone 1) include (DICE1, DICE2, DICE3, DICE4, DICE5, RunningTable)
- For this Lab/Problem, copy code into Notepad whenever you see the (*****Demonstrate Code*****) indicator. For this problem, you will build upon prior code to ultimately create a final coding solution.

Milestone 1 – Table Creation (DDL) and INSERT Statements (DML)

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.

DICE1

INSERT INTO DICE1 (d1)VALUES (1);

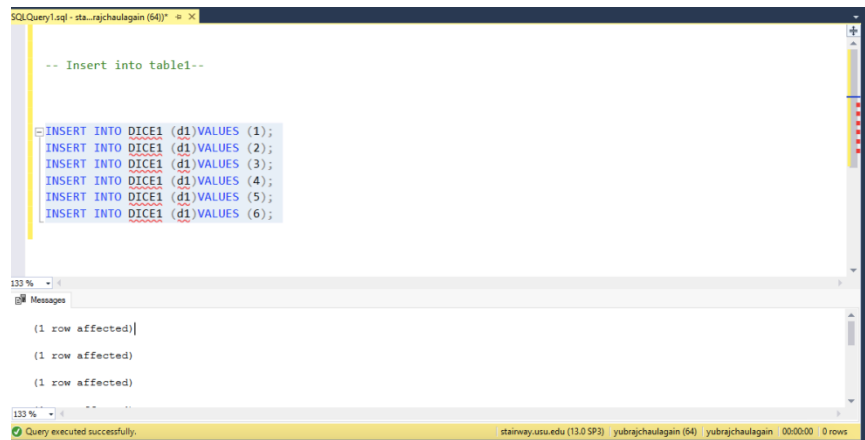
INSERT INTO DICE1 (d1)VALUES (2);

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 table1--  
  
INSERT INTO DICE1 (d1)VALUES (1);  
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);
```

Messages

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

Query executed successfully. | stairway.uns.edu (13.0 SP3) | yubrajchaulagain (64) | yubrajchaulagain | 00:00:00 | 0 rows

DICE2

INSERT INTO DICE2 (d2)VALUES (1);

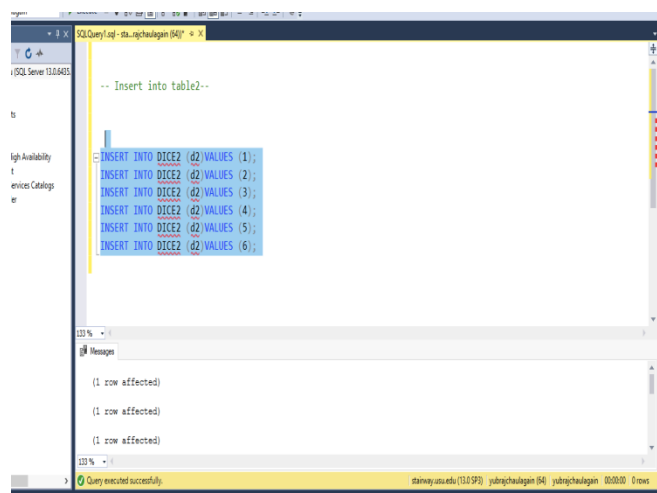
INSERT INTO DICE2 (d2)VALUES (2);

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 table2--  
  
INSERT INTO DICE2 (d2)VALUES (1);  
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);
```

Messages

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

Query executed successfully. | stairway.uns.edu (13.0 SP3) | yubrajchaulagain (64) | yubrajchaulagain | 00:00:00 | 0 rows

Dice3

```
INSERT INTO DICE3 (d3)VALUES (1);
```

```
INSERT INTO DICE3 (d3)VALUES (2);
```

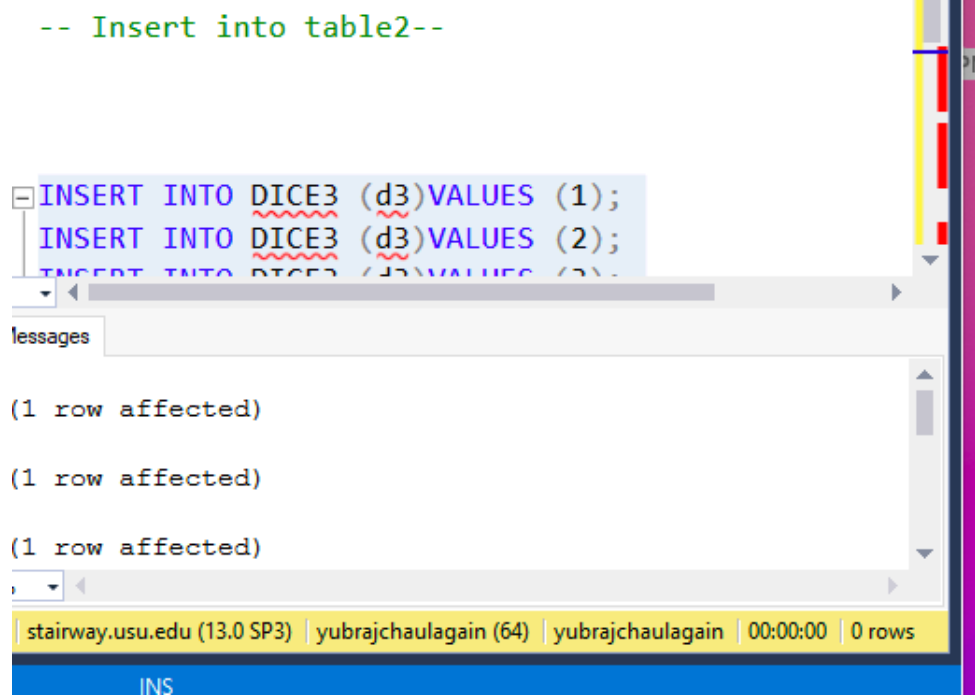
```
INSERT INTO DICE3 (d3)VALUES (3);
```

```
INSERT INTO DICE3 (d3)VALUES (4);
```

```
INSERT INTO DICE3 (d3)VALUES (5);
```

```
INSERT INTO DICE3 (d3)VALUES (6);
```

```
-- Insert into table2--
```



```
INSERT INTO DICE3 (d3)VALUES (1);  
INSERT INTO DICE3 (d3)VALUES (2);  
INSERT INTO DICE3 (d3)VALUES (3);
```

messages

(1 row affected)

(1 row affected)

(1 row affected)

stairway.usu.edu (13.0 SP3) | yubrajchaulagain (64) | yubrajchaulagain | 00:00:00 | 0 rows

Dice 4

```
INSERT INTO DICE4 (d4)VALUES (1);
```

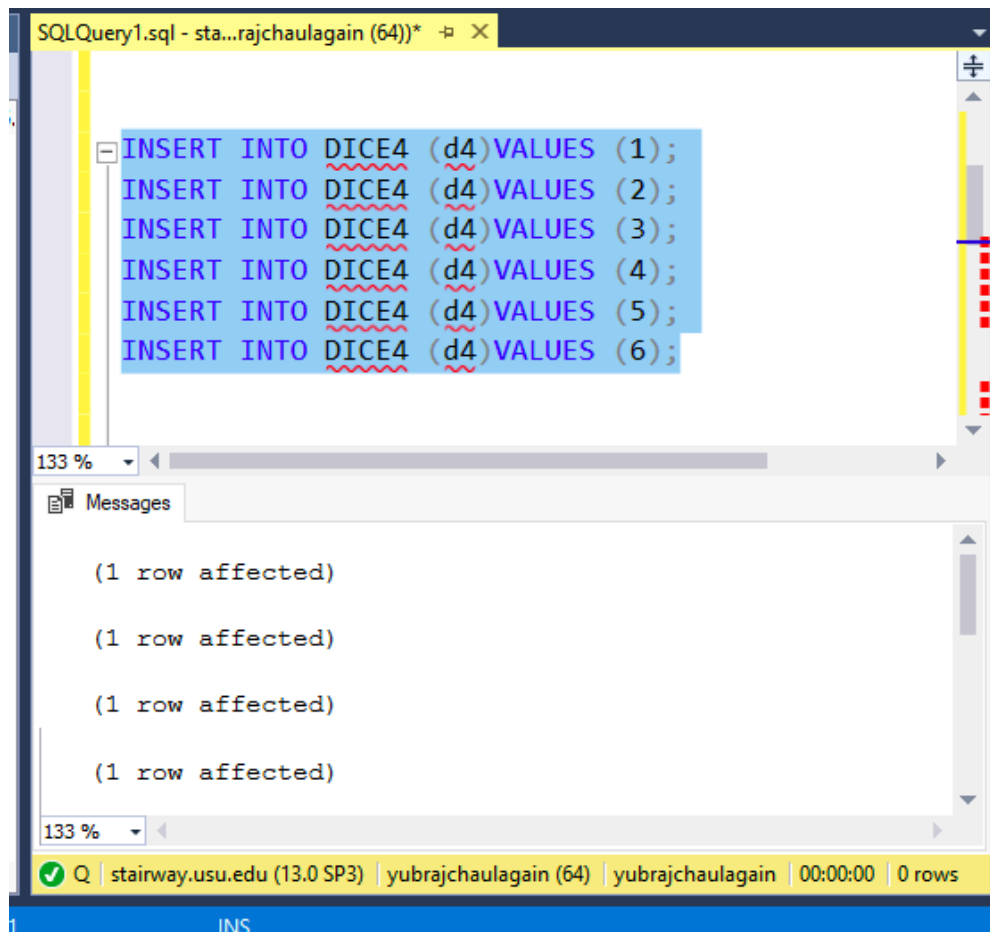
INSERT INTO DICE4 (d4)VALUES (2);

INSERT INTO DICE4 (d4)VALUES (3);

INSERT INTO DICE4 (d4)VALUES (4);

INSERT INTO DICE4 (d4)VALUES (5);

INSERT INTO DICE4 (d4)VALUES (6);



Dice 5

INSERT INTO DICE5 (d5)VALUES (1);

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;**

SQLQuery1.sql - sta...rajchaulagain (64) * - X

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

133 %

Results Messages

	d1	d2	d3	d4	d5
1	1	1	1	1	1
2	1	2	1	1	1
3	1	3	1	1	1
4	1	4	1	1	1
5	1	5	1	1	1
6	1	6	1	1	1
7	1	1	2	1	1
8	1	2	2	1	1
9	1	3	2	1	1
10	1	4	2	1	1
11	1	5	2	1	1
12	1	6	2	1	1
13	1	1	3	1	1
14	1	2	3	1	1
15	1	3	3	1	1
16	1	4	3	1	1
17	1	5	3	1	1
18	1	6	3	1	1

Query executed successfully.

stairway.usu.edu (13.0 SP3) | yubrajchaulagain (64) | yubrajchaulagain | 00:00:00 | 7,776 rows

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

7776 ROWS

stairway.usu.edu (13.0 SP3) | yubrajchaulagain (64) | yubrajchaulagain | 00:00:00 | 7,776 rows

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();
```

133 %

Results Messages

	d1	d2	d3	d4	d5
1	1	4	6	5	2
2	2	4	3	3	1
3	5	1	5	2	4
4	3	1	4	5	6
5	6	1	3	6	6
6	3	3	6	4	4
7	5	2	4	6	5
8	5	3	1	4	3
9	2	2	6	5	5
10	6	5	3	3	3
11	2	3	3	2	4
12	6	6	3	3	5
13	5	3	4	5	3
14	4	5	6	2	6
15	5	6	3	1	6
16	1	2	2	1	5
17	5	3	3	3	3

Query executed successfully.

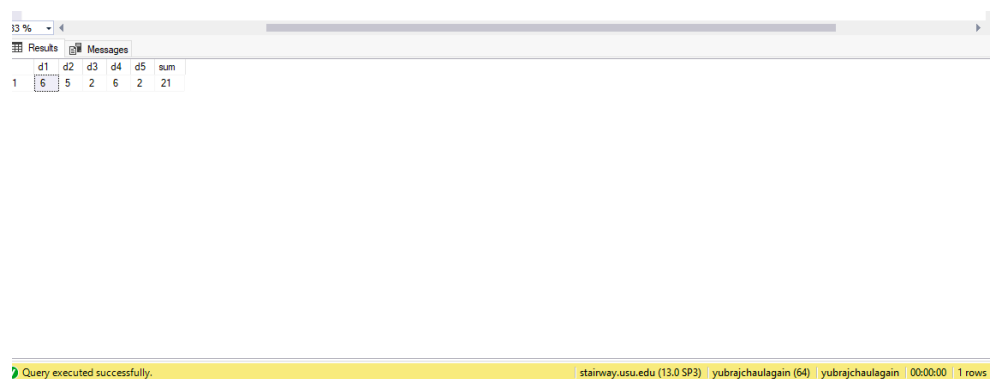
stairway.usu.edu (13.0 SP3) | yubrajchaulagain (64) | yubrajchaulagain | 00:00:00 | 7,776 rows

Provide Answers/Code for all three items

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'.

```
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();
```



The screenshot shows a SQL query results window with a table containing one row. The columns are labeled d1, d2, d3, d4, d5, and sum. The values in the row are 6, 5, 2, 6, 2, and 21 respectively. The status bar at the bottom indicates the query was executed successfully.

	d1	d2	d3	d4	d5	sum
1	6	5	2	6	2	21

Query executed successfully. | stairway.usu.edu (13.0 SP3) | yubrajchaulagain (64) | yubrajchaulagain | 00:00:00 | 1 rows

2. Write a CASE EXPRESSION to create a 7th 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'. ***The Case Expression possibilities (win combinations) is provided in the game image at the bottom of this assignment.

```
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

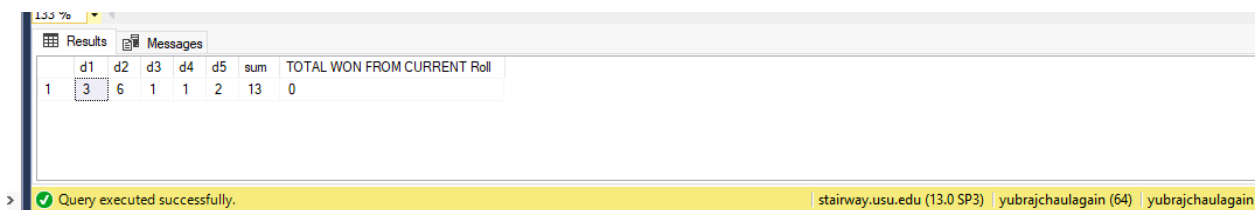
WHEN d1+d2+d3+d4+d5 = 29 THEN 10

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();



d1	d2	d3	d4	d5	sum	TOTAL WON FROM CURRENT Roll
1	3	6	1	1	2	0

Provide Code after finishing these two items

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
```

```
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
```

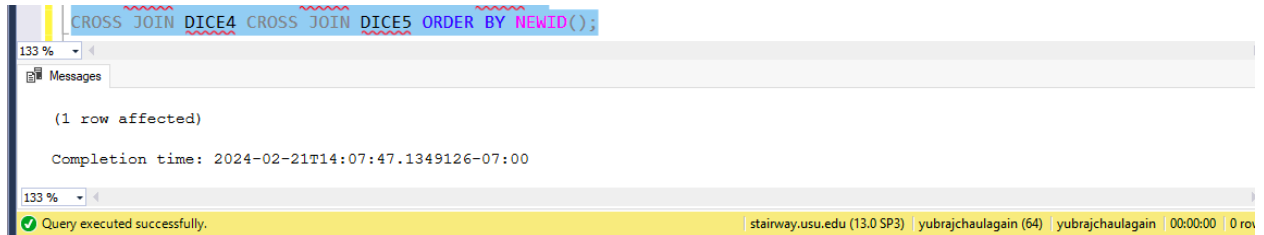
```
WHEN d1+d2+d3+d4+d5 = 29 THEN 10
```

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();



2. Congratulations! Open a separate 'New Query' Window and:

1. Write SELECT *

FROM RunningTable;

133 %

Results Messages

	RollNum	d1	d2	d3	d4	d5	SUM	TOTAL
1	1	3	2	2	1	1	9	3

Q | stairway.usu.edu (13.0 SP3) | yubrajchaulagain (94) | yubrajchaulagain | 00:00:00 | 1 rows

- This should provide results similar to below:

Milestone 5 – WINDOW FUNCTION

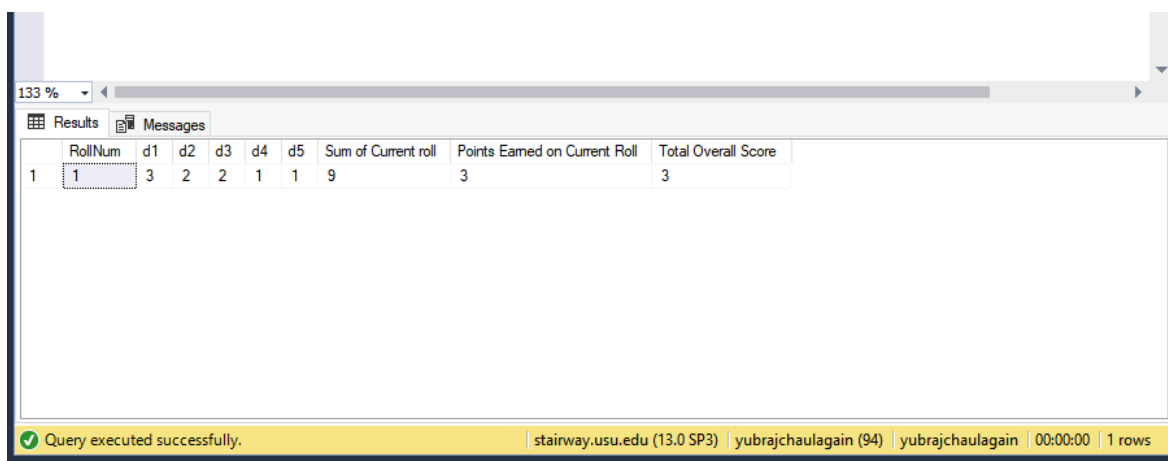
NOTE: You have created some amazing code that replicates the Top Dog mechanical game. Your current code creates random dice results, the sum of the 5 dice, and points

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'  
from runningtable  
ORDER BY RollNUM DESC;
```

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



The screenshot shows a SQL query results window with a zoom level of 133%. The window has tabs for 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with the following data:

RollNum	d1	d2	d3	d4	d5	Sum of Current roll	Points Earned on Current Roll	Total Overall Score
1	3	2	2	1	1	9	3	3

At the bottom of the window, a status bar indicates: "Query executed successfully. | stairway.usu.edu (13.0 SP3) | yubrajchaulagain (94) | yubrajchaulagain | 00:00:00 | 1 rows"

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 10
```

```
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;

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

Results		Messages								
	RollNum	d1	d2	d3	d4	d5	Sum of Current roll	Points Earned on Current Roll	Total Overall Score	
1	35	1	6	2	1	4	14	2	12	

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