MIS 4330 Video Activity – Week 1

Course Introduction Video

1. **What is your major, and how do you anticipate SQL to help with your career?**

I'm a digital marketing major going into Data Analytics. I want to become a marketing analyst and SQL is the primary skill for that field.

2. Do you have a PC or Mac computer? Do you anticipate any difficulties loading SQL Server Management Studio on your computer? If yes, please explain.

I have a MAC. I haven't faced any problems yet.

Chapter 2 – Part 1

1. **The Planet table describes diameter in kilometres (km). Write an SQL statement that shows the diameter of all objects in miles instead of kilometres. Include the name of the object in the output. Use Aliases when appropriate. Order the output from largest to smallest. When you are done, do a Google search to check your answer.**

```
SELECT
      Name,
     (diameter * 0.62) AS diamater_in_miles
FROM Planet
ORDER BY diamater_in_miles;
```

100 %

Results | Messages

	Name	diamater_in_miles
1	Pluto	1469.40
2	Moon	2154.50
3	Mercury	3024.98
4	Mars	4214.76
5	Venus	7504.48
6	Earth	7908.72
7	Neptune	30707.36
8	Uranus	31693.16
9	Satum	74732.32
10	Jupiter	88650.08

2. **Write an SQL statement that groups the information by Inner or outer locations. Show the Maximum and Minimum density of the planets. Also, include the column that depicts inner and outer in the SELECT statement (Order by this as well). Don't include the moon or Pluto.**

```
SELECT *
FROM Planet;

SELECT
    Location,
    MAX(Density) AS Max_Density,
    MIN(Density) AS Min_Density
FROM Planet
WHERE Name <> 'moon' AND Name <> 'Pluto'
GROUP BY Location
ORDER BY Location;
```

100 %

Results | Messages

	Location	Max_Density	Min_Density
1	Inner	5514	3933
2	Outer	1638	687

3. **Write an SQL Query to return Interest and the average gravity of the planets. Don't include the moon. Group the data by Interest and only include groups with an average gravity of 10 or greater!**

```
SELECT
    Interest,
    AVG(Gravity) AS Avg_Gravity
FROM Planet
WHERE Name <> 'moon'
GROUP BY Interest
HAVING AVG(Gravity) >= 10;
```

100 %

Results | Messages

	Interest	Avg_Gravity
1	High	13.966666

Chapter 2 – Part 2

1. **How many minutes old are you? Do not hard wire the current date.**

Format with , (e.g., 1,000 instead of 1000).

```
SELECT FORMAT(DATEDIFF(minute, '19950827', GETDATE()), 'n0') AS 'Minutes_Old';
```

100 %

Results | Messages

	Minutes_Old
1	13,716,564

Chapter 2 – Ben-Gan Readings

1. **Describe the differences between Regular and Unicode character data types.**

Regular	**Unicode**
A char stores fixed-length, non-unicode characters.	A n-char stores fixed-length unicode characters.
One byte to store the data.	Two bytes to store data.
char stores upto 8000 bytes.	n-char stores upto 4000 bytes.
It is a widely preferred data type.	It is preferred less.
It uses Unicode standards for storing the data.	It uses ASCII standards to store the data.
Syntax : col_name char(n); *n is the number of bytes.	Syntax : col_name nchar(n); *n is the number of bytes.
The number of bytes is not mandatory to specify. It automatically takes the default value of 1 and converts it in ASCII format.	The letter n(national) in nchar has to be specified in order to convert the data in Unicode format.

2. **Read Page 74 Date and time data types. Based on the fact our University data set uses Dates, but doesn't consider times, what other data types besides DATETIME should we consider?**

- T-SQL supports six date and time data types: two legacy types called DATETIME and SMALLDATETIME, and four later additions called **DATE, TIME, DATETIME2, and DATETIMEOFFSET.**

Data type	Storage (bytes)	Date range	Accuracy	Recommended entry format and example
DATETIME	8	January 1, 1753, through December 31, 9999	3 1/3 milliseconds	'YYYYMMDD hh:mm:ss.nnn' '20160212 12:30:15.123'
SMALLDATETIME	4	January 1, 1900, through June 6, 2079	1 minute	''YYYYMMDD hh:mm' '20160212 12:30'
DATE	3	January 1, 0001, through December 31, 9999	1 day	'YYYY-MM-DD' '2016-02-12'
TIME	3 to 5	N/A	100 nanoseconds	'hh:mm:ss.nnnnnnn' '12:30:15.1234567'
DATETIME2	6 to 8	January 1, 0001, through December 31, 9999	100 nanoseconds	'YYYY-MM-DD hh:mm:ss.nnnnnnn' '2016-02-12 12:30:15.1234567'
DATETIMEOFFSET	8 to 10	January 1, 0001, through December 31, 9999	100 nanoseconds	'YYYY-MM-DD hh:mm:ss.nnnnnnn [+|-] hh:mm' '2016-02-12 12:30:15.1234567 +02:00'

TABLE 2-1 Date and time data types