Sets

A **set** is a collection of distinct objects:
A set may be described with words or can use notation to list each element or pattern in curly braces. Here are two ways to show the same set.
Set A is the set of colors in the USA Flag:
A = {red, white, blue}

Each distinct object in the set is called an **element**.

To show set notation using a pattern, use the **ellipses** to show that the pattern should continue. Effectively, they mean "and so on".
A = {1, 2, 3….}

A **subset** is a set of elements contained within a specified set:
If A = {1, 2, 3, 4, 5, 6} & B = {3, 4, 5} then "B" is a subset of "A".

The **universal set** is collection of all elements under consideration and is designated with a **U** symbol (not to be confused with the set operator Union's symbol).

The **null set** is an empty set is a set that has no elements. It is also called and is symbolized by the **ø** symbol or null symbol.

The **cardinality** of set is simply the number of elements it has:
If Set A represents the days of a week, then the cardinality of A is 7. The mathematical way to express this is |A| = 7.

Venn Diagrams are used to display the relationships among sets.

Logical Operators

The logical **AND** is represented by the symbol ∩ or intersection. An intersection of two sets means that the element is in BOTH sets. For example, if Set A = {pine, oak, elm} and Set B = {maple, pine, walnut, cherry}, then the A ∩ B = {pine}. Pine is the only element in both sets.

The logical **OR** is represented by the symbol ∪ or union. An intersection of two sets means that the element is in one, the other, or both sets, in other words AT LEAST ONE of the sets. For example, if Set A = {pine, oak, elm} and Set B = {maple, pine, walnut, cherry}, then the A ∪ B = {pine, oak, elm, maple, walnut, cherry}.

The logical **NOT** is represented by **´** symbol. NOT refers to an element that is NOT in a set. For example, if A is the set of folks who drink coffee, then NOT **A'** is the set of folks who do NOT drink coffee. NOT means effectively "the opposite of". NOT is a unary operator and acts only on one input. It is the complement of a set.

The logical **XOR** is represented by $A \veebar B$ symbol. The logical XOR is known as the "exclusive or" logical operators. It yields true if exactly one (but not both) of two conditions is true.

Truth Tables

Truth tables are really just a new way to view sets - kind of like the list view. A truth table is a way of viewing the possible combinations of answers to a series of yes or no questions and then determining the outcome for each scenario. The number of rows (after the header row) is always equal to 2^n where n is the number of questions (also called conditions, criteria, or propositions).

If set A was the set of folks who drink tea and B was the set of folks who drink coffee. What are the possible scenarios for responses? You could have a person who drinks neither, only tea, only coffee, or both.

The truth table would then look like if we represented No as ”0” and Yes as “1”.

TRUTH TABLE

Set A (Tea)	Set B (Coffee)
0	0
0	1
1	0
1	1

Do the first two columns look familiar? You have just counted in binary from 0 to 3. The first two columns for two questions will always be the same: 00, 01, 10, 11 (no-no, no-yes, yes-no, yes-yes). You could think of this like the flip of two coins. You could have HH, HT, TH, or TT. See two coins, two choices each toss. 2 to the 2nd power = 4. 4 different scenarios.

Let’s start with a **truth table for AND**. We want our outcome to be true only when BOTH A AND B are true (drink both beverages).

A (Tea) B (Coffee) A AND B (Both) 0 0 0 0 1 0 1 0 0 1 1 1

TRUTH TABLE - AND

Set A (Tea)	Set B (Coffee)	A AND B (Both)
0	0	0
0	1	0
1	0	0
1	1	1

An **OR truth table** with two conditions would again have the same first two columns. In this example we would be looking for those folks who drink at least one of the beverages. We want to give them some coupons, for instance.

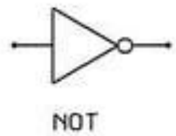
TRUTH TABLE - OR

Set A (Tea)	Set B (Coffee)	A OR B (Both)
0	0	0
0	1	1
1	0	1
1	1	1

The logical **"NOT" operator** means that the opposite is true. It is the complement of a SET. NOT is a unary operator and acts only on one input.

TRUTH TABLE - NOT

Set A (Tea)	NOT 'A
0	1
1	0



The logical "**XOR" operator** yields true if exactly one (but not both) of two conditions is true. Those folks who drink only one type of beverage.

TRUTH TABLE - XOR

Set A (Tea)	Set B (Coffee)	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

3 Bit Binary Number (2^3 or $2\times2\times2$ = 8) Thus (0 → 1 through 7) Rows

Place "3" or 2^2 or "**4**" **COLUMN "C"**	Place "2" or 2^1or "**2**" **COLUMN "B"**	Place "1" or 2^0 or "**1**" **COLUMN "A"**	Decimal	Because
0	0	0	0	0=0
0	0	1	1	1=1
0	1	0	2	2=2
0	1	1	3	3=2+1
1	0	0	4	4=4
1	0	1	5	5=4+1
1	1	0	6	6=4+1
1	1	1	7	7=4+2+1

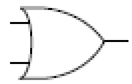
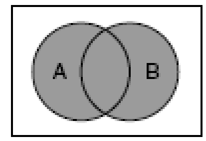
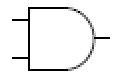
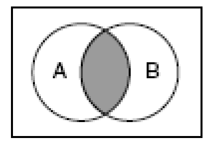
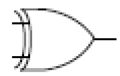
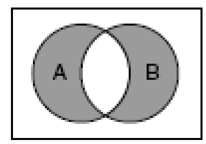
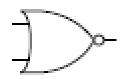
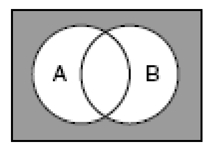
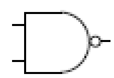
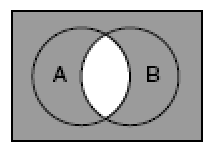
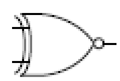
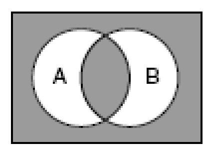
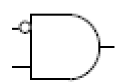
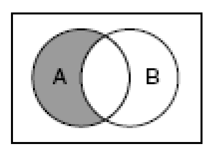
COLUMN "A" NOTES:
When building the truth tables the sequence is 0 1 0 1 0 1 0 (alternating at every row).
COLUMN "B" NOTES:
When building the truth tables the sequence is 00 11 00 11 00 (alternating at every second row).
COLUMN "C" NOTES:
When building the truth tables the sequence is 0000 1111 0000 1111 0000 (alternating at every fourth row).

Now, let's look at truth tables if there are three conditions (also called **criteria, questions, or propositions**). In this new example let's say that a person needs to take either ENG 101 **OR** ENG 155 **AND** SPC 205 in order to enroll in ENG 199. In this case ENG 101 is the first condition (A), ENG 155 is the second (B), and SPC 205 is the last(C). The outcome is whether or not he/she gets into ENG 199. So, we would create a truth table for the following relationship: (A OR B) AND C.

In building the different scenarios, we would again look to binary. In this case, we would have 3 bits (2^3 or $2\times2\times2$ = 8), so there would be 8 rows. If we didn't want to think too hard about the combinations of responses, we could simply count in binary from 0 to 7 (we always start with 0).
Our columns would be 000, 001, 010, 011, 100, 101, 110, and 111. See: 0, 1, 2, 3, 4, 5, 6, and 7.

TRUTH TABLE (A or B) and C

SET A (ENG 101)	SET B (ENG 155)	SET C (SPC 205)	A ∪ B	(A ∪ B) ∩ C
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

Name	Input A	Input B	Output	Symbol	Venn diagram
OR	0 1 0 1	0 0 1 1	0 1 1 1		
AND	0 1 0 1	0 0 1 1	0 0 0 1		
XOR	0 1 0 1	0 0 1 1	0 1 1 0		
NOR	0 1 0 1	0 0 1 1	1 0 0 0		
NAND	0 1 0 1	0 0 1 1	1 1 1 0		
XNOR	0 1 0 1	0 0 1 1	1 0 0 1		
INH	0 1 0 1	0 0 1 1	0 1 0 0		
IMP	0 1 0 1	0 0 1 1	1 0 1 1