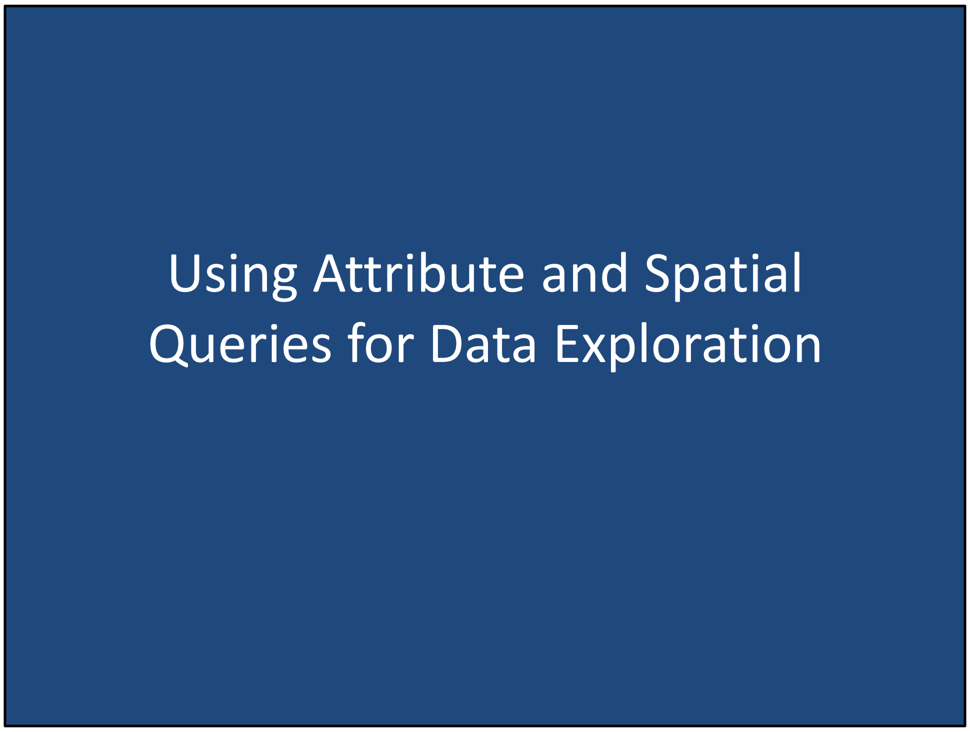
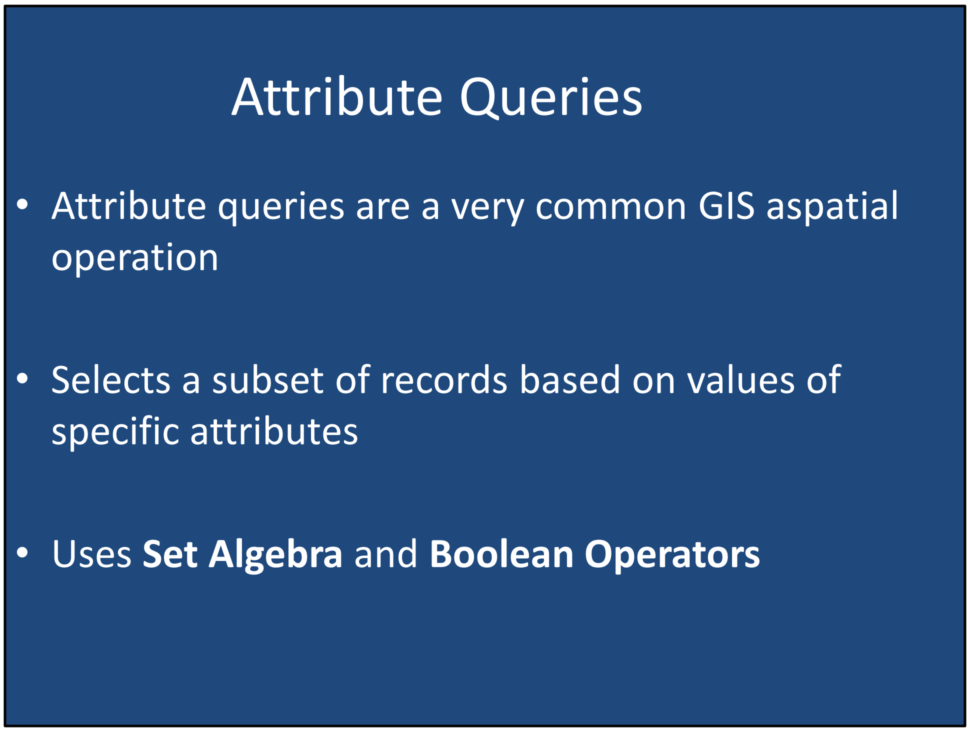


Using Attribute and Spatial Queries for Data Exploration

Using attribute and spatial queries for data exploration.

Attribute Queries

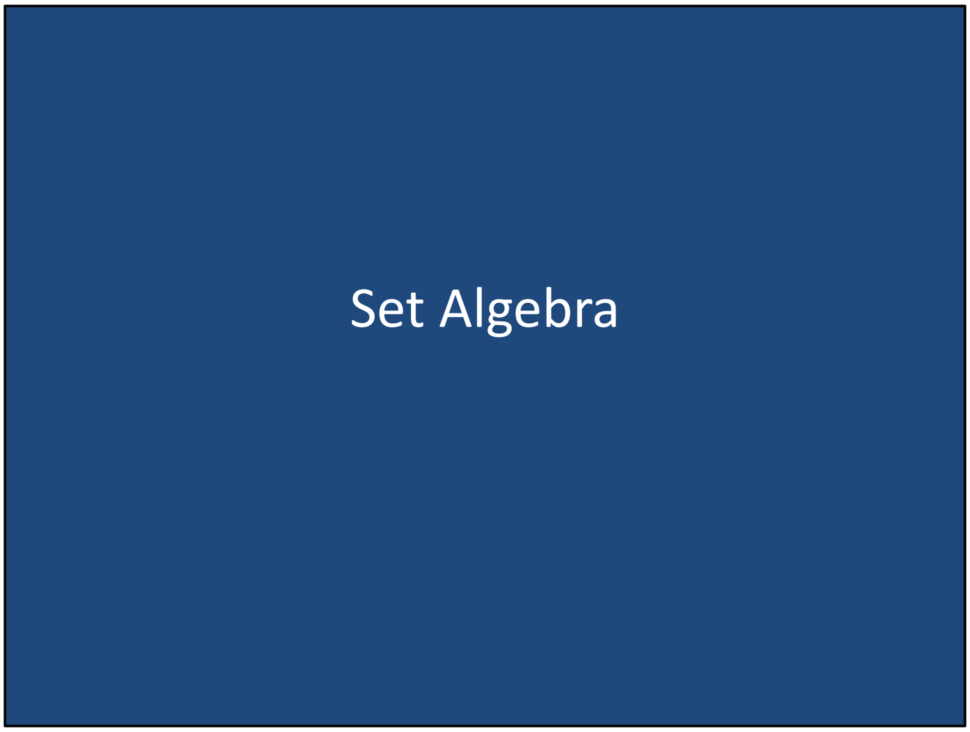
Attribute Queries



Attribute Queries

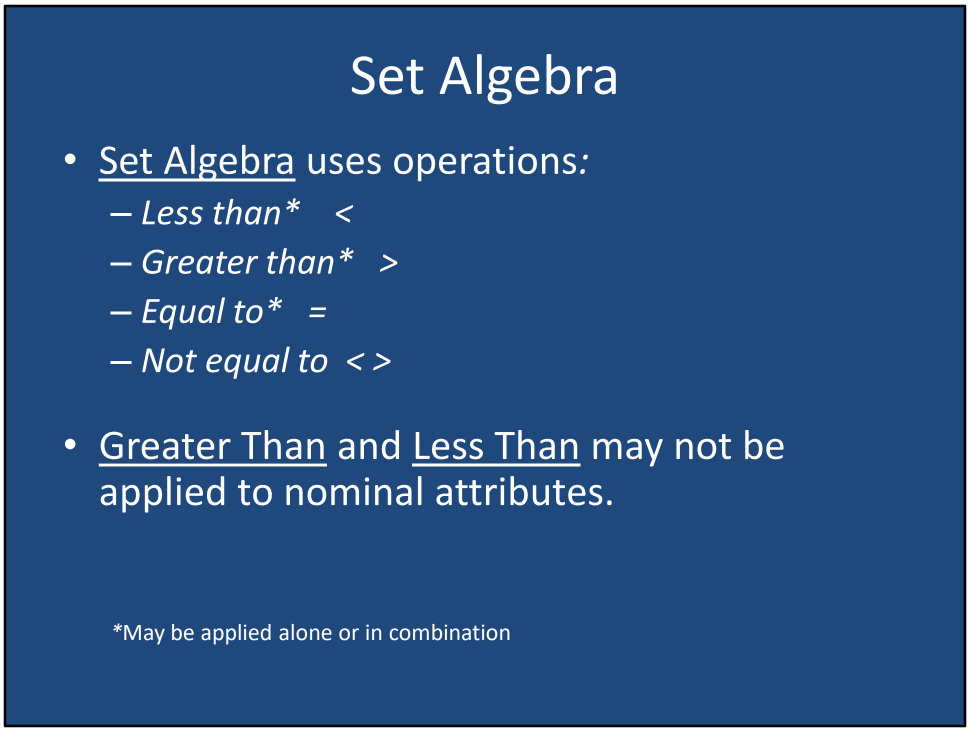
- Attribute queries are a very common GIS aspatial operation
- Selects a subset of records based on values of specific attributes
- Uses **Set Algebra** and **Boolean Operators**

Attribute queries are an extremely common GIS aspatial operation. Attribute queries select a subset of records based on values of specific attributes. Each attribute query must specify three things: an attribute field, a set algebra operator, and an attribute value. For example of an attribute query, image if we had a data set of land parcels for sale. If we were interested in selecting parcels that are at least six hundred acres in size, our attribute query would be: Acres greater than six hundred where 'Acres' is the attribute field, 'greater than' is the set algebra operator, and 'six hundred' is the attribute value we wish to evaluate. Attribute queries can also select records based on multiple attributes combined together using Boolean operators, such as 'and', 'or', and 'not'.



Set Algebra

Set Algebra

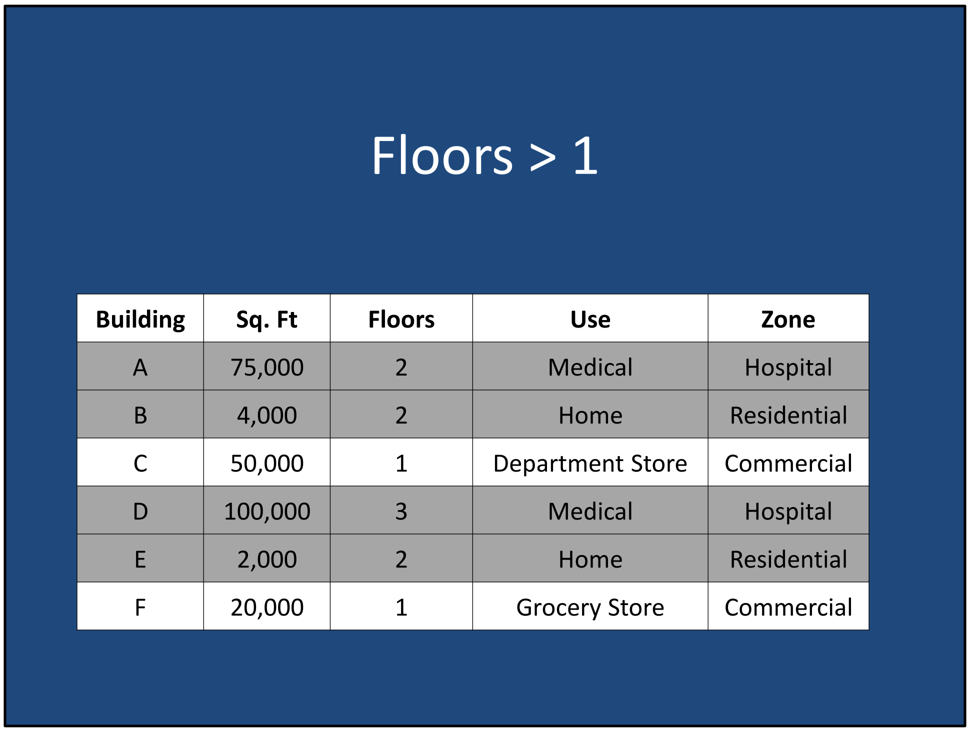


Set Algebra

- Set Algebra uses operations*:*
  - *Less than** <
  - *Greater than** >
  - *Equal to** =
  - *Not equal to* < >
- Greater Than and Less Than may not be applied to nominal attributes.

*May be applied alone or in combination

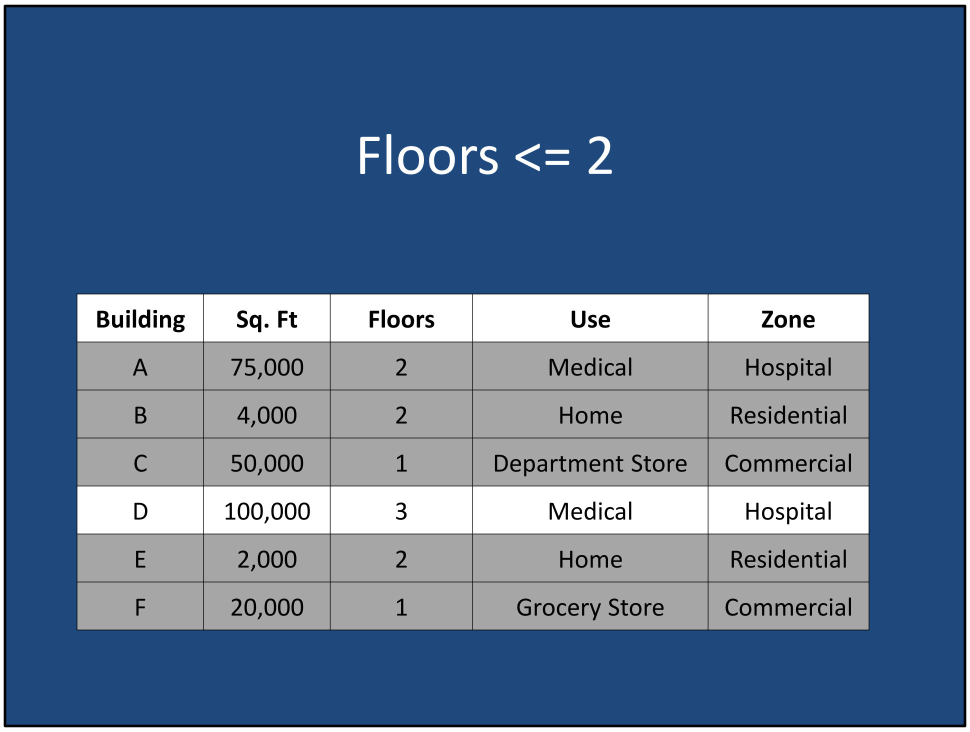
Set algebra uses operations to determine whether two values are equivalent or not. The four basic set algebra operations are less than, greater than, equal to, and not equal to. The less than operation checks to see whether the value on the left is less than the value on the right. The less than operation is represented by the left angle bracket symbol. The greater than operation checks to see where the value on the left is greater than the value on the right. The greater than operation Is represented by the right angle bracket symbol. The equal to operation checks to see whether the values on both sides are equal to each other. The equal to operation Uses the =. The not equal to operation checks to see if the values on both sides are different from each other, and is equivalent to the combination of less than and greater than. The not equal to operation And is represented by both the left angle bracket and right angle bracket used together. The three operations annotated with the astrik means that they can be applied alone, or in combination. So for instance, you can perform the test of whether the value in the left side is less than or equal to the value of the right side. The symbol that would represent this operation, would be both the left angle bracket followed by the =. The result of all of these operations are either the value of “true”, or “false”. As a quick challenge question, I am making the claim that greater than and less than may not be applied to nominal attributes. Why do you think this is the case? <<pause>> the answer is because nominal attributes are only descriptors, and it is illogical to compare them with respect to magnitude or rank. It is, however, logical to compare them using the equal to, or not equal to operators.



Floors > 1

Building	Sq. Ft	Floors	Use	Zone
A	75,000	2	Medical	Hospital
B	4,000	2	Home	Residential
C	50,000	1	Department Store	Commercial
D	100,000	3	Medical	Hospital
E	2,000	2	Home	Residential
F	20,000	1	Grocery Store	Commercial

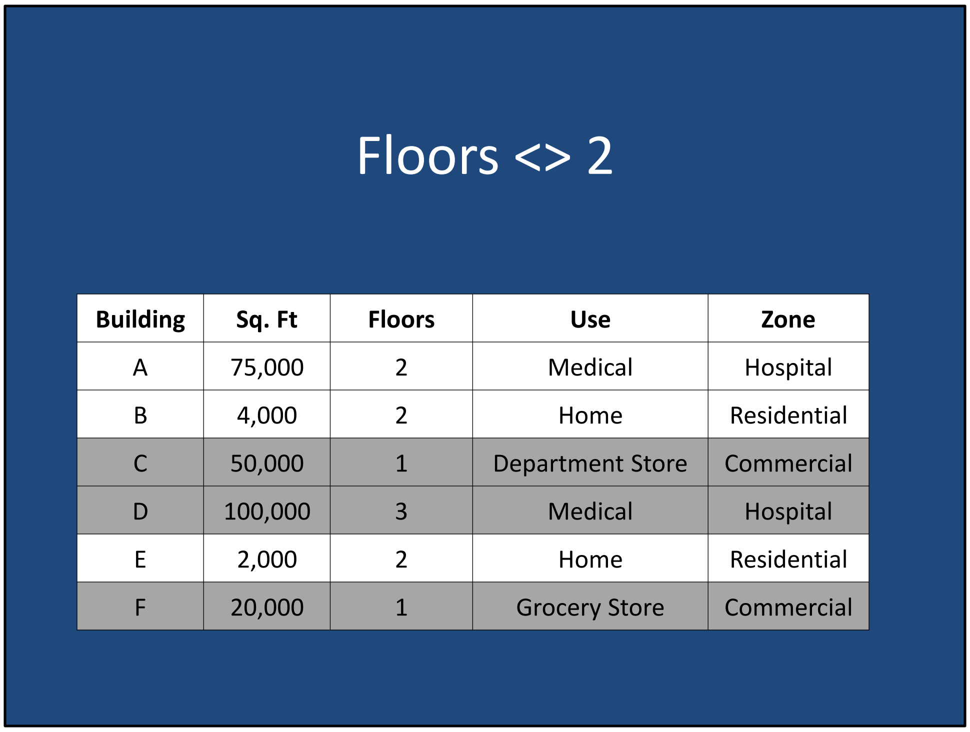
Let’s look at some examples of table queries using simple set algebra operations. If we consider this attribute table representing different buildings, their square footage, number of floors, use, and zone, we can perform some simple attribute queries to select a subset of these buildings. For example, if we use the attribute query floors greater than 1, buildings A, B, D, and E would be selected as their floors attribute are all equal than the number 1.



Floors <= 2

Building	Sq. Ft	Floors	Use	Zone
A	75,000	2	Medical	Hospital
B	4,000	2	Home	Residential
C	50,000	1	Department Store	Commercial
D	100,000	3	Medical	Hospital
E	2,000	2	Home	Residential
F	20,000	1	Grocery Store	Commercial

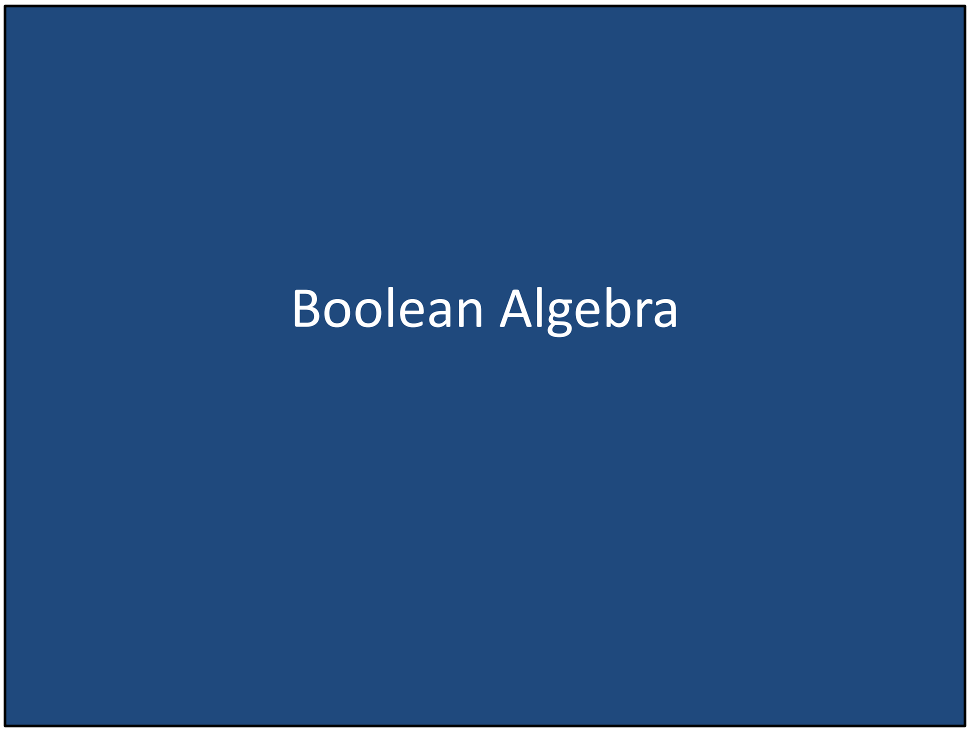
If we change our attribute query to now read floors less than or equal to 2, then buildings A, B, C, E, and F are all selected, leaving building the as the only building not selected. This is because all the selected buildings have an attribute value of 2 or less for the floors attribute.



Floors <> 2

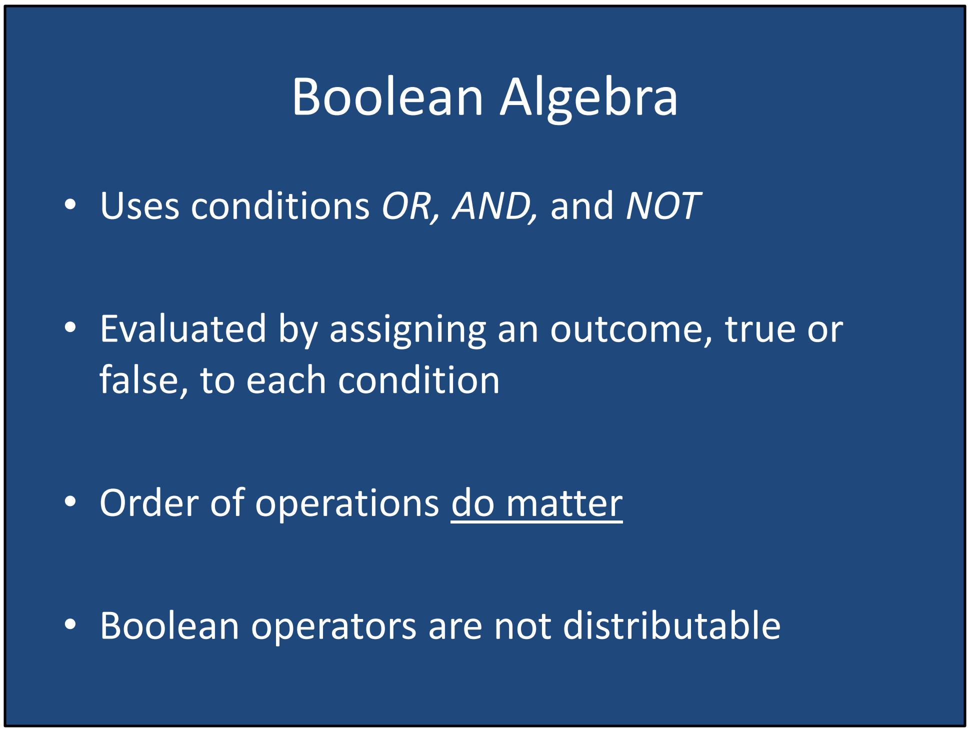
Building	Sq. Ft	Floors	Use	Zone
A	75,000	2	Medical	Hospital
B	4,000	2	Home	Residential
C	50,000	1	Department Store	Commercial
D	100,000	3	Medical	Hospital
E	2,000	2	Home	Residential
F	20,000	1	Grocery Store	Commercial

As a final example, if our attribute query reads floors not equal to 2, then only buildings see, the, and F are selected, as those buildings have a floor attribute that is anything but the number 2.



Boolean Algebra

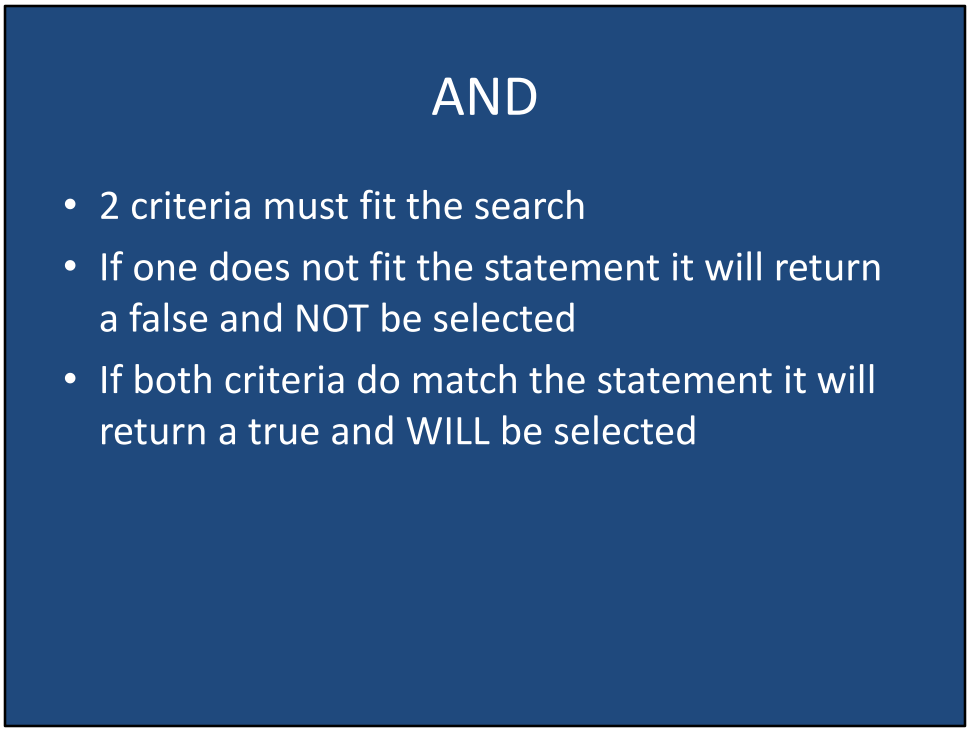
Boolean algebra



Boolean Algebra

- Uses conditions *OR, AND,* and *NOT*
- Evaluated by assigning an outcome, true or false, to each condition
- Order of operations <u>do matter</u>
- Boolean operators are not distributable

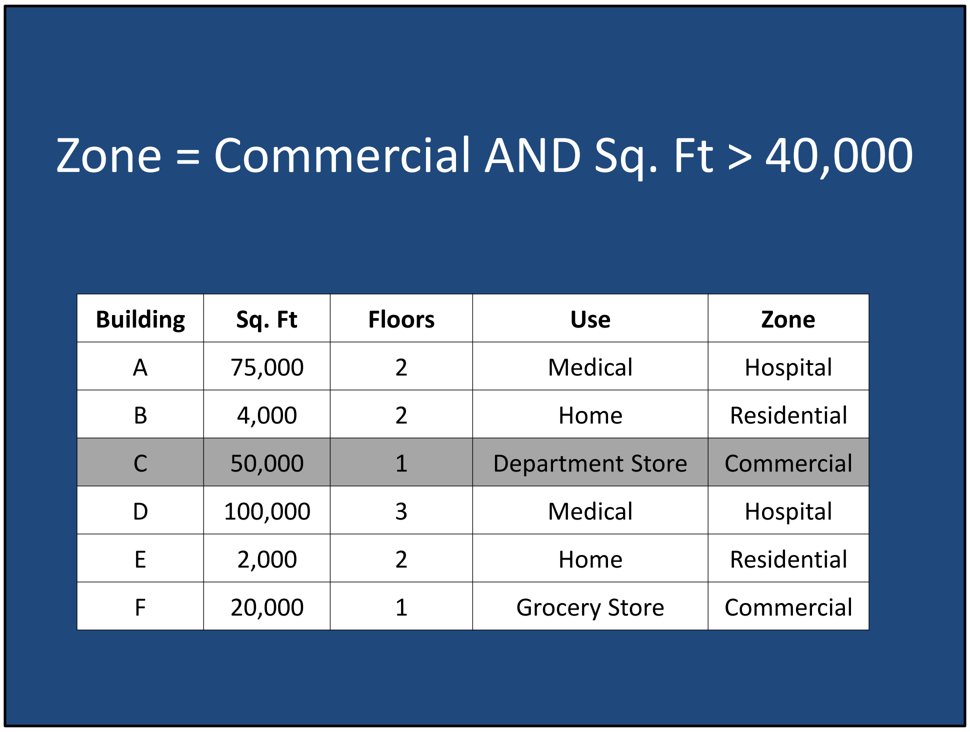
Boolean algebra has multiple conditional operators. The three operators we are going to focus on are the most common Boolean operators. They are: or, and, and not. The Boolean operators evaluate values on the left and right side of the operator, and then assigns an outcome. The outcome is a Boolean, or binary result such as true/false, 0/1, on/off, or any other dichotomy. In Boolean algebra, order of operations do matter, therefore it is not uncommon to use many sets of parentheses to force a particular order of operations. It is also important to note that Boolean operators are not distributable inside of parentheses.



AND

- 2 criteria must fit the search
- If one does not fit the statement it will return a false and NOT be selected
- If both criteria do match the statement it will return a true and WILL be selected

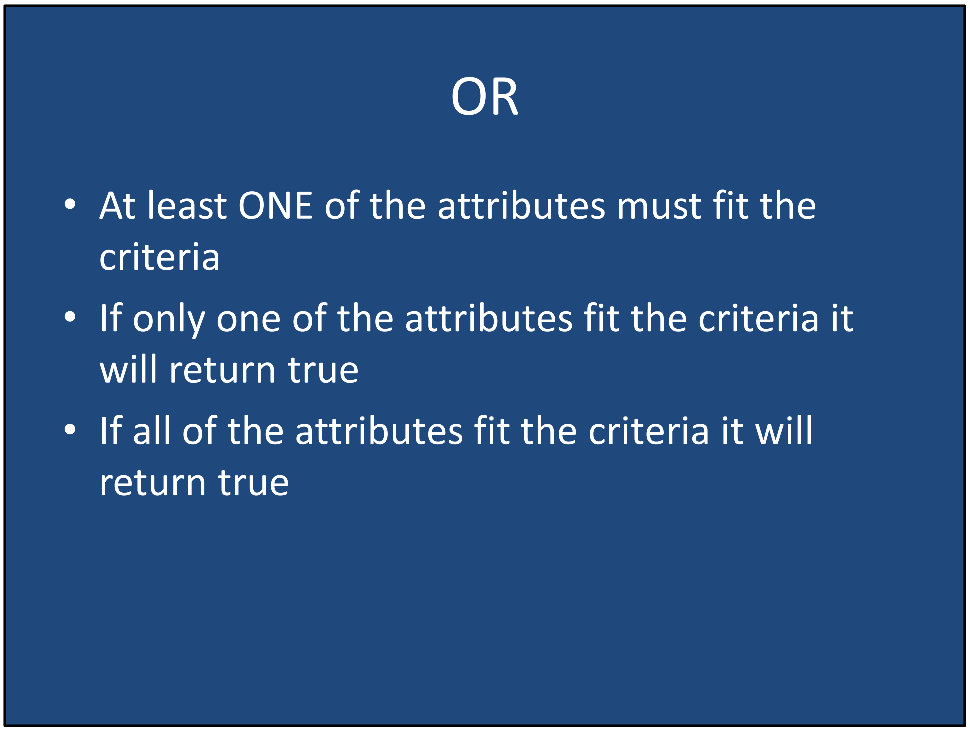
Let's begin with the and Boolean operator. For the and Boolean operator, the queries to the left and to the right of the and Boolean operator must both evaluate to true for the entire statement to be considered true. If one of the 2, or both criteria, do not evaluate to true, the entire state will return false, and therefore, the record will not be selected. The only way the record will be selected using the and Boolean operator is if both queries evaluate to true.



Zone = Commercial AND Sq. Ft > 40,000

Building	Sq. Ft	Floors	Use	Zone
A	75,000	2	Medical	Hospital
B	4,000	2	Home	Residential
C	50,000	1	Department Store	Commercial
D	100,000	3	Medical	Hospital
E	2,000	2	Home	Residential
F	20,000	1	Grocery Store	Commercial

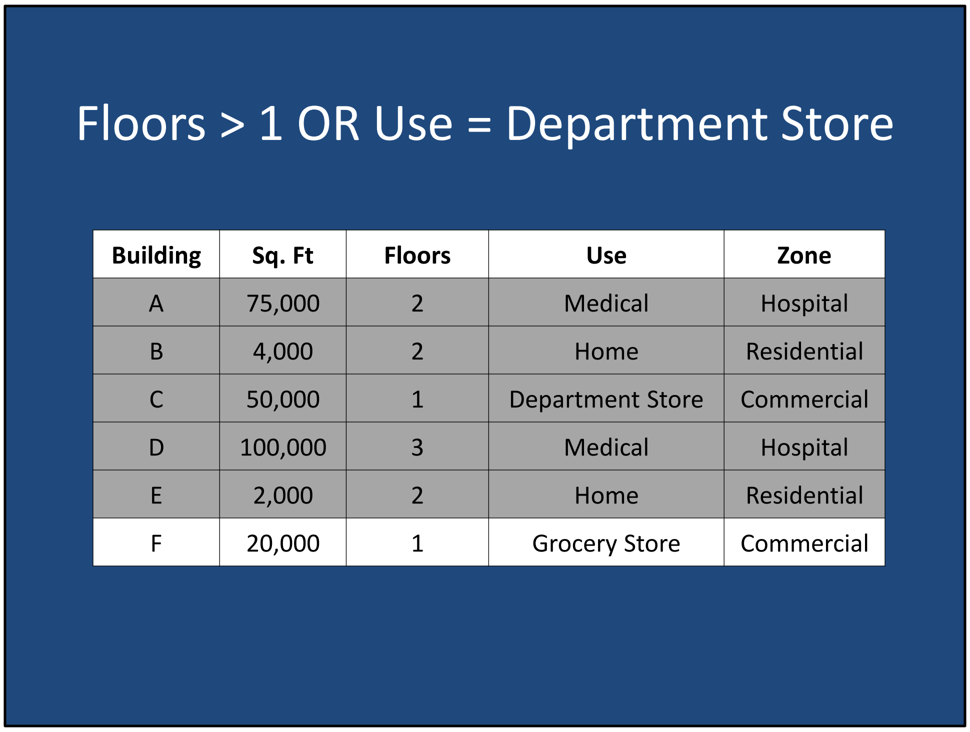
Let's consider an example using the buildings attribute table shown here. Note that our statement has 2 queries combined with an and Boolean operator. The left query selects all buildings where the zone attribute is equal to commercial. The right query selects all buildings where the square feet attribute is greater than 40,000. The and Boolean operator will only select a record if a building is owned to commercial and has a square footage greater than 40,000. If either one of these 2 queries are not met, then the record is not selected. Therefore, based on this query, building C is the only building that is both zoned commercial and has a square footage greater than 40,000 feet.



OR

- At least ONE of the attributes must fit the criteria
- If only one of the attributes fit the criteria it will return true
- If all of the attributes fit the criteria it will return true

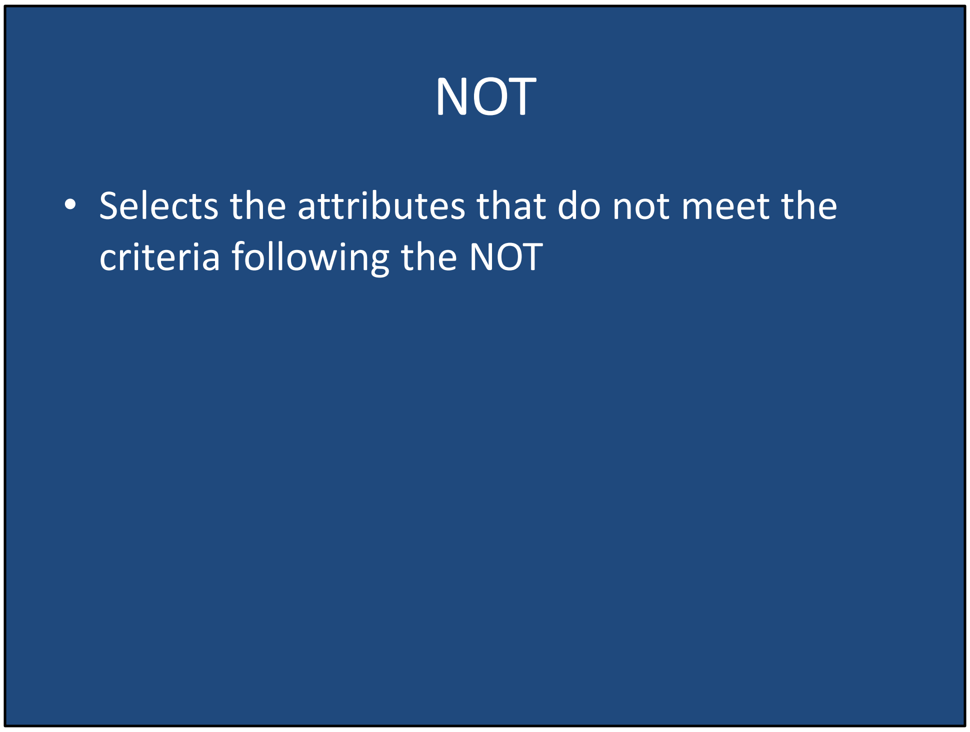
The or Boolean operator again considers the results of 2 queries, one on each side. In order for the entire statement to be considered true, at least one of the attribute queries must return true. If only one of the attribute queries returns true, the record will still be selected as the entire statement is considered true. If all of the attribute queries return true, then the record will still be selected as the entire statement is considered true. However, if neither of the attribute queries return true, in other words they both return false, then the record will not be selected. Therefore, the only way a statement is considered false using to attribute queries and it or Boolean operator, is if both attribute queries return false.



Floors > 1 OR Use = Department Store

Building	Sq. Ft	Floors	Use	Zone
A	75,000	2	Medical	Hospital
B	4,000	2	Home	Residential
C	50,000	1	Department Store	Commercial
D	100,000	3	Medical	Hospital
E	2,000	2	Home	Residential
F	20,000	1	Grocery Store	Commercial

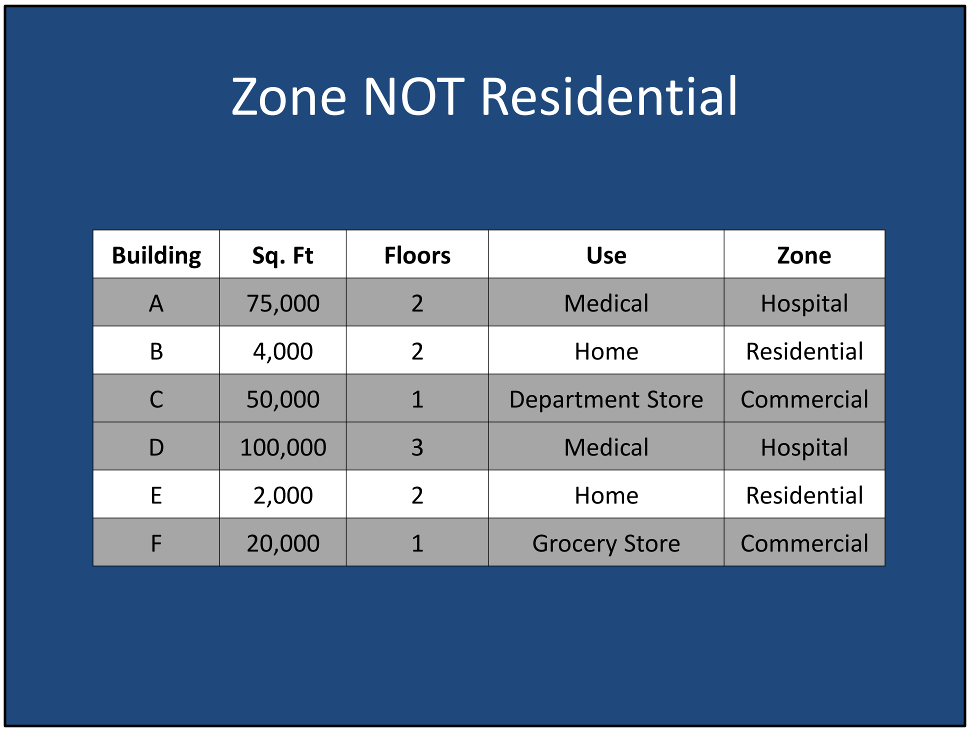
As an example of the statement using the or Boolean operator, let's consider the statement that reads floors greater than one or use equals department store. This statement will select all records where the building has more than one floor, or the building is used as a department store. Therefore, building F is the only record not selected because it has one floor and is used as a grocery store And does not meet either of the 2 queries on each side of the or Boolean operator.



NOT

- Selects the attributes that do not meet the criteria following the NOT

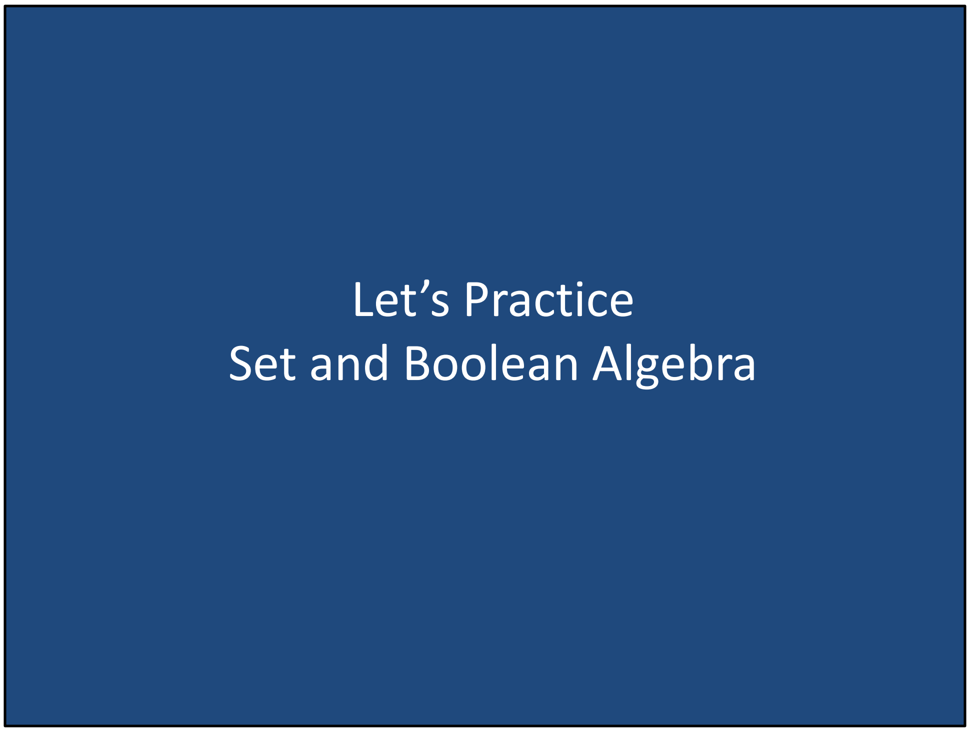
The final Boolean operator we will consider is the not Boolean operator. The not Boolean operator selects attributes that do not meet the attribute query following the not Boolean operator. In other words, in the attribute query following the not Boolean operator will have its evaluated value switched. This means that attribute queries that evaluate to true, will be switched the false and vice versa.



Zone NOT Residential

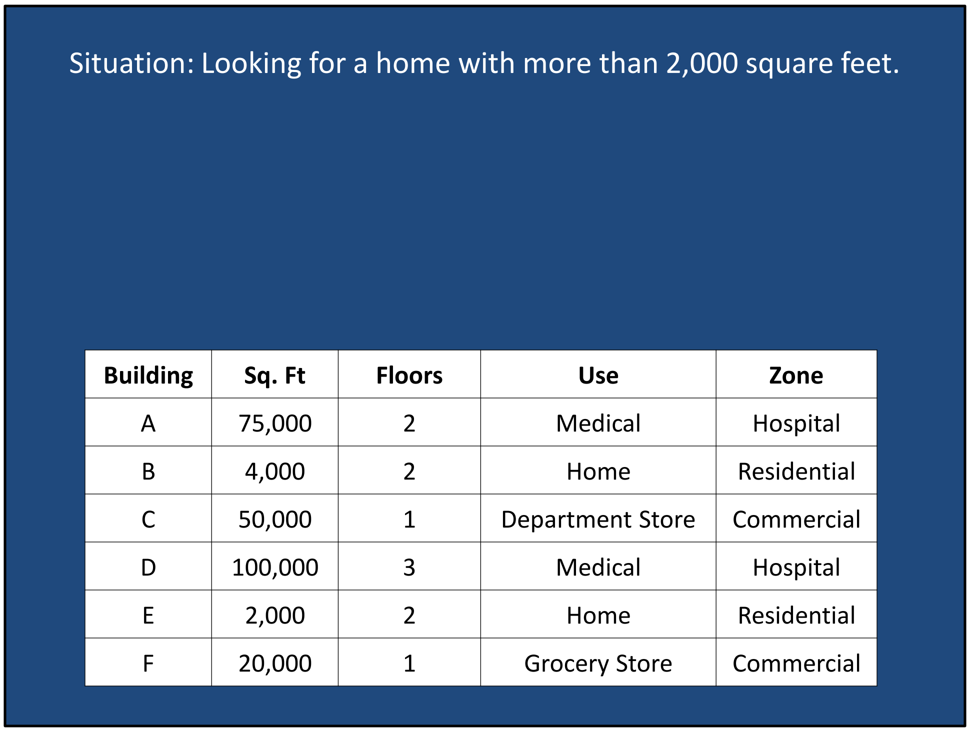
Building	Sq. Ft	Floors	Use	Zone
A	75,000	2	Medical	Hospital
B	4,000	2	Home	Residential
C	50,000	1	Department Store	Commercial
D	100,000	3	Medical	Hospital
E	2,000	2	Home	Residential
F	20,000	1	Grocery Store	Commercial

Let's consider the query not zone equals residential. Parentheses of them placed around zone equals residential to help clarify what is the Boolean operator and what is the attribute query. For the selection, 1st, each record is evaluated on the zone attribute to see if it is equal to residential. If the zone is equal residential, therefore returning a value of true, next the not Boolean operators evaluated turning the true value to false. Therefore, you could read the statement as select all building where zone is not equal to residential. Based on this query, buildings A, C, D, and F are selected as none of them are zoned residential.



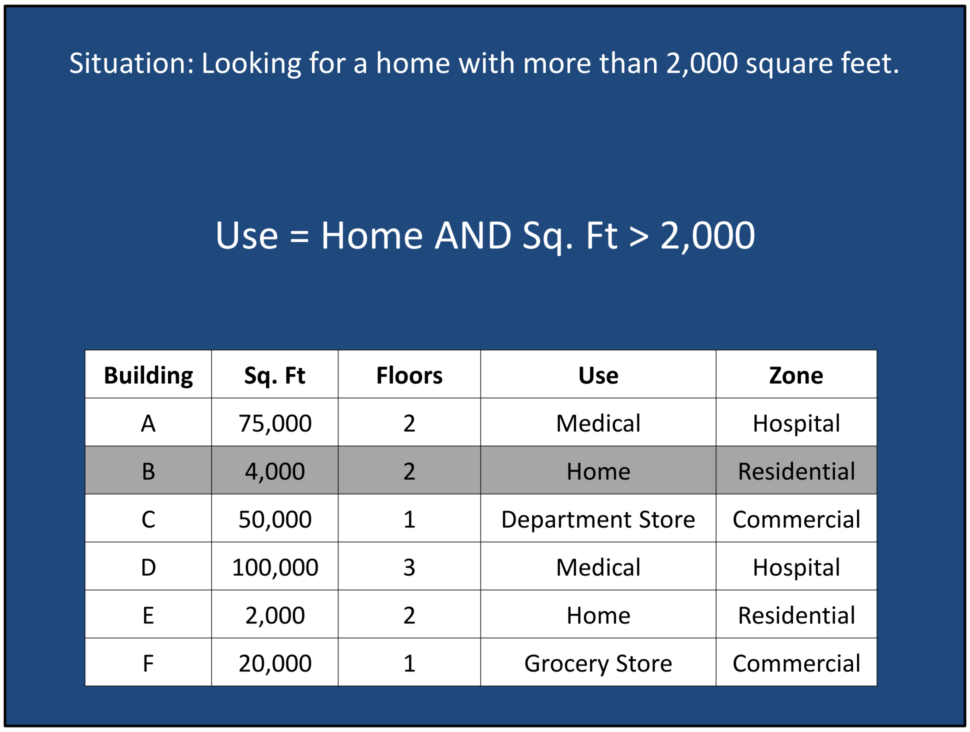
Let’s Practice
Set and Boolean Algebra

Let’s practice set and Boolean algebra a little more.


Situation: Looking for a home with more than 2,000 square feet.

Building	Sq. Ft	Floors	Use	Zone
A	75,000	2	Medical	Hospital
B	4,000	2	Home	Residential
C	50,000	1	Department Store	Commercial
D	100,000	3	Medical	Hospital
E	2,000	2	Home	Residential
F	20,000	1	Grocery Store	Commercial

Let's consider the same attribute table of buildings a through F. It's also consider the situation where we are looking for a home with more than 2000 ft.^{2}. Take a few moments to construct an attribute query that will select all homes with more than 2000 ft.^{2}.

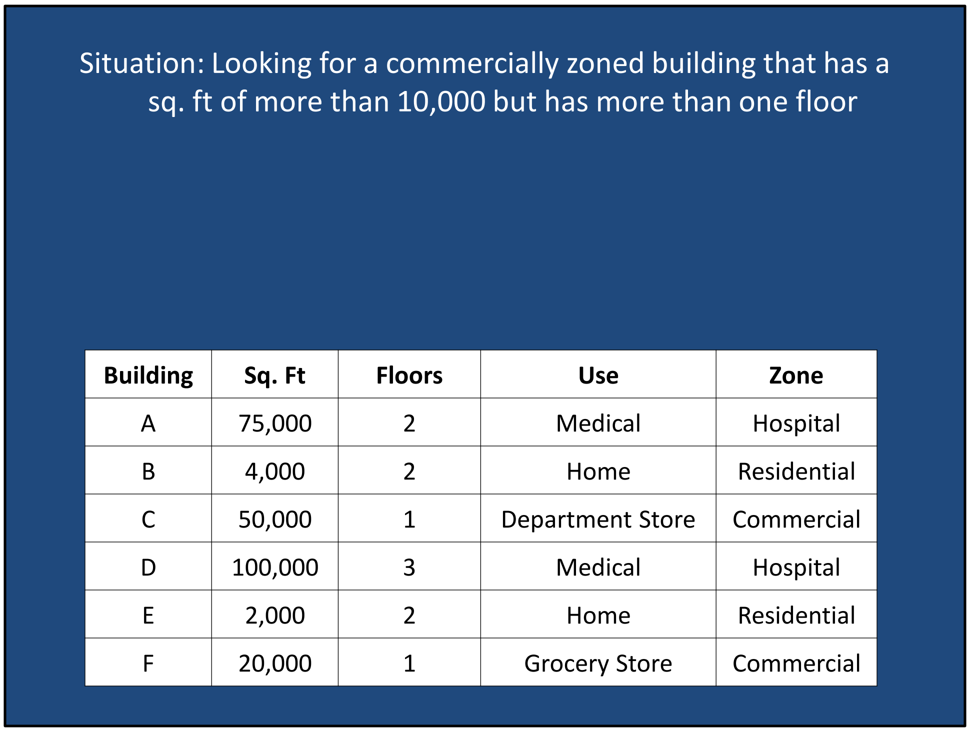


Situation: Looking for a home with more than 2,000 square feet.

Use = Home AND Sq. Ft > 2,000

Building	Sq. Ft	Floors	Use	Zone
A	75,000	2	Medical	Hospital
B	4,000	2	Home	Residential
C	50,000	1	Department Store	Commercial
D	100,000	3	Medical	Hospital
E	2,000	2	Home	Residential
F	20,000	1	Grocery Store	Commercial

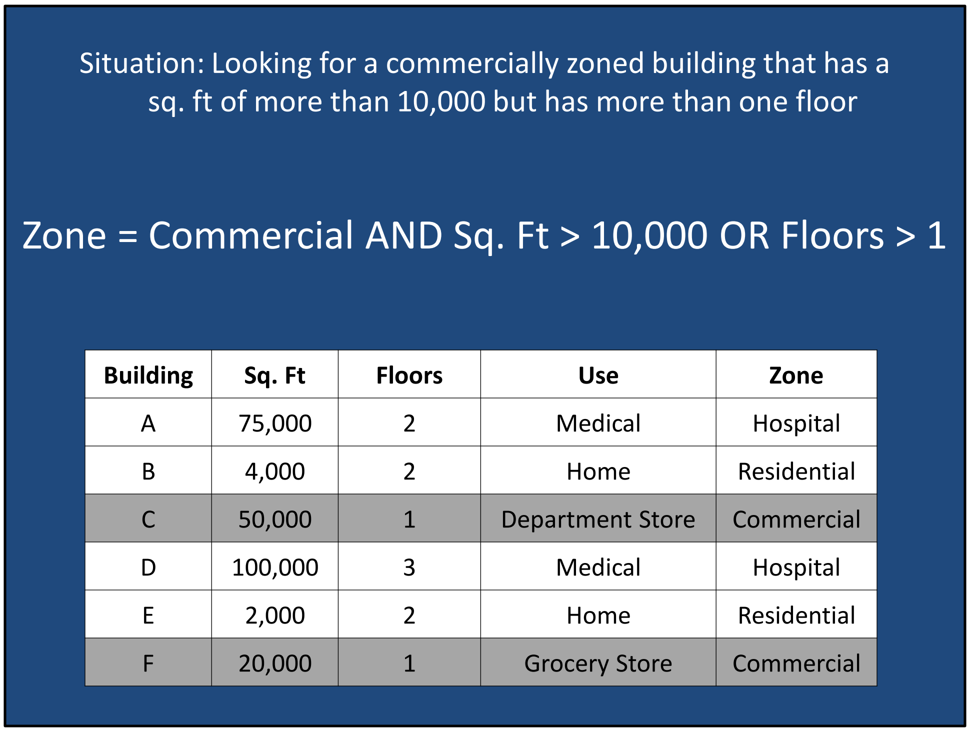
Based on the situation where we want to find a home with more than 2000 ft.^{2}, and appropriate attribute query we query on 2 different fields, the fields use, and square feet. As we want both attributes to be true in order to select the record, we use the and Boolean operator. Therefore, are attribute query reads use equals home and square feet is greater than 2000. This selects building be as it is a home and has more than 2000 feet. It is not select record E, because even though the use is home, the square footage is exactly 2000 and is not greater than the requested 2000 ft.^{2}.



Situation: Looking for a commercially zoned building that has a sq. ft of more than 10,000 but has more than one floor

Building	Sq. Ft	Floors	Use	Zone
A	75,000	2	Medical	Hospital
B	4,000	2	Home	Residential
C	50,000	1	Department Store	Commercial
D	100,000	3	Medical	Hospital
E	2,000	2	Home	Residential
F	20,000	1	Grocery Store	Commercial

Now let's consider a different situation. Here, we want to select all records where the building is commercially zoned, has a square footage of more than 10,000, but has more than one floor. Take a few moments to determine what an appropriate attribute query would be.



Situation: Looking for a commercially zoned building that has a sq. ft of more than 10,000 but has more than one floor

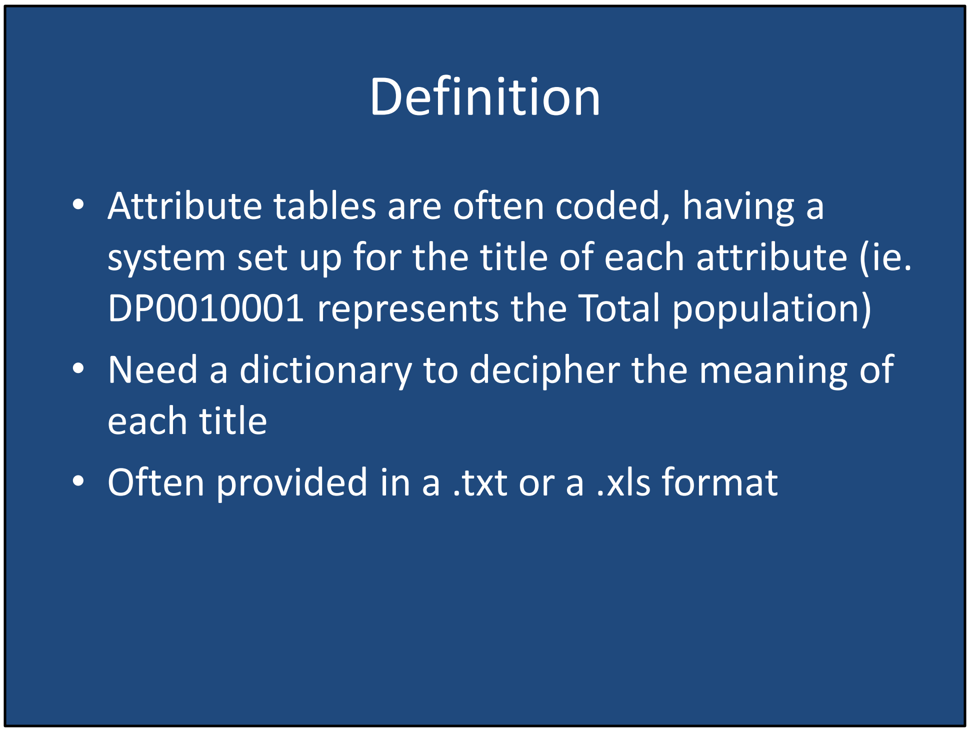
Zone = Commercial AND Sq. Ft > 10,000 OR Floors > 1

Building	Sq. Ft	Floors	Use	Zone
A	75,000	2	Medical	Hospital
B	4,000	2	Home	Residential
C	50,000	1	Department Store	Commercial
D	100,000	3	Medical	Hospital
E	2,000	2	Home	Residential
F	20,000	1	Grocery Store	Commercial

For this attribute query, where using 3 attribute queries on 3 different attribute fields, and to Boolean operators. The query used here to satisfy the situation is zone equals commercial and square feet is greater than 10,000 or floors is greater than one. To evaluate this attribute query we must keep in mind the order of operations for Boolean operators. Not is evaluated before and and and is evaluated before or. Therefore, zone equals commercial and square feet is greater than 10,000 should be evaluated to true or false and then that's true or false value should be placed to the left of or floors greater than 1. Buildings see, and F are the only 2 building selected as a both meet the requirements of the zone equaling commercial and square footage greater than 10,000. Note that building CNF only have one floor, but since the attribute queries surrounding the and Boolean operator evaluated to true, even if floors evaluates to false, the record is still selected. In other words, no building meets every single attribute query in this statement, the building C and F still meet the criteria set forth in the statement.

Data Dictionaries

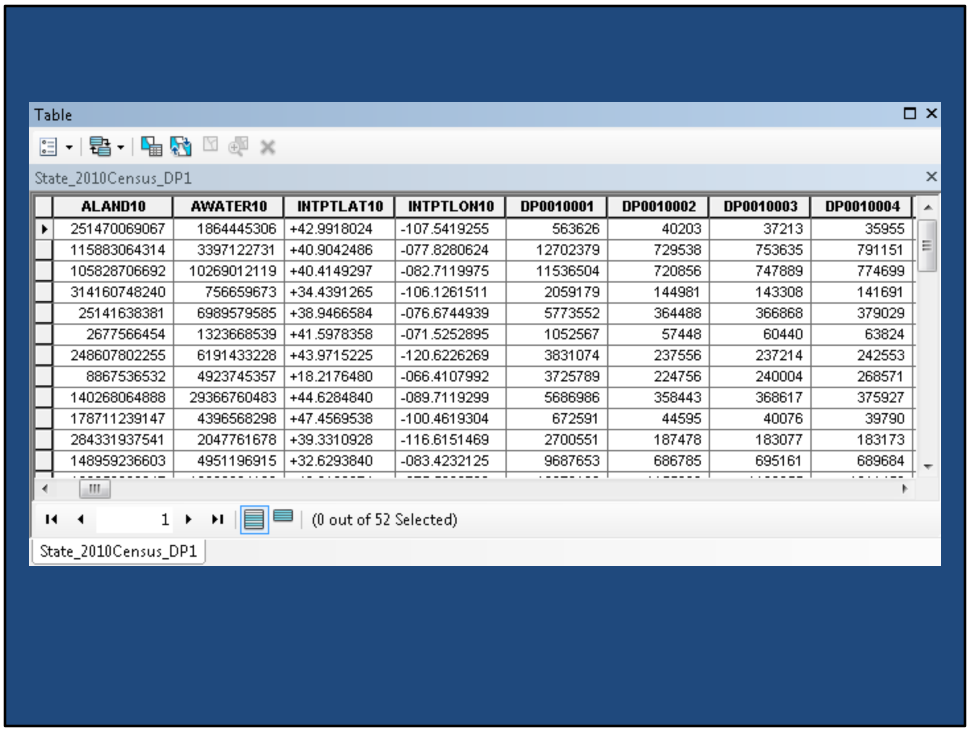
Data dictionaries



Definition

- Attribute tables are often coded, having a system set up for the title of each attribute (ie. DP0010001 represents the Total population)
- Need a dictionary to decipher the meaning of each title
- Often provided in a .txt or a .xls format

Often times in attribute table's field names are coded, having a system set up for the title of each attribute. For example, in a census data set, the field name DP0010001 represents total population. Naturally, DP0010001 is not a self evident field name, therefore, we needed dictionary to decipher the meaning. The purpose of the data dictionary is to provide the descriptive field name for attribute tables. The data dictionaries are often provided in a text file format or Microsoft Excel format.



Table

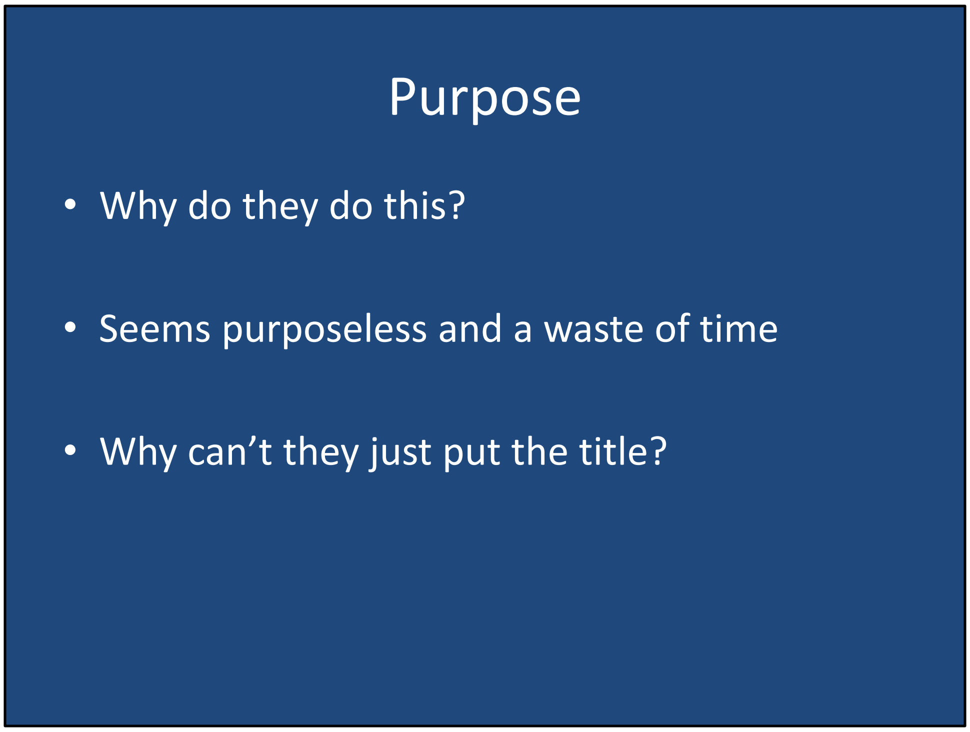
State_2010Census_DP1

ALAND10	AWATER10	INTPTLAT10	INTPTLON10	DP0010001	DP0010002	DP0010003	DP0010004
251470069067	1864445306	+42.9918024	-107.5419255	563626	40203	37213	35955
115883064314	3397122731	+40.9042486	-077.8280624	12702379	729538	753635	791151
105828706692	10269012119	+40.4149297	-082.7119975	11536504	720856	747889	774699
314160748240	756659673	+34.4391265	-106.1261511	2059179	144981	143308	141691
25141638381	6989579585	+38.9466584	-076.6744939	5773552	364488	366868	379029
2677566454	1323668539	+41.5978358	-071.5252895	1052567	57448	60440	63824
248607802255	6191433228	+43.9715225	-120.6226269	3831074	237556	237214	242553
8867536532	4923745357	+18.2176480	-066.4107992	3725789	224756	240004	268571
140268064888	29366760483	+44.6284840	-089.7119299	5686986	358443	368617	375927
178711239147	4396568298	+47.4569538	-100.4619304	672591	44595	40076	39790
284331937541	2047761678	+39.3310928	-116.6151469	2700551	187478	183077	183173
148959236603	4951196915	+32.6293840	-083.4232125	9687653	686785	695161	689684

1 (0 out of 52 Selected)

State_2010Census_DP1

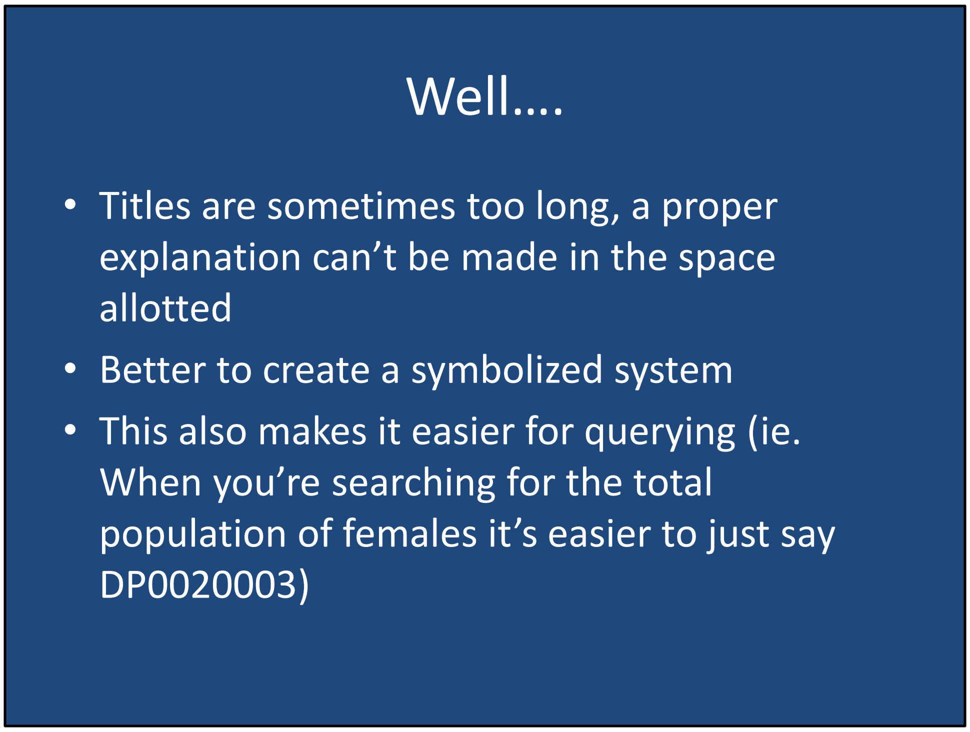
As an example, here's the state census data for the 2010 United States Census. Looking at the attributes field names, they are all coded and do not provide easily identifiable information.



Purpose

- Why do they do this?
- Seems purposeless and a waste of time
- Why can't they just put the title?

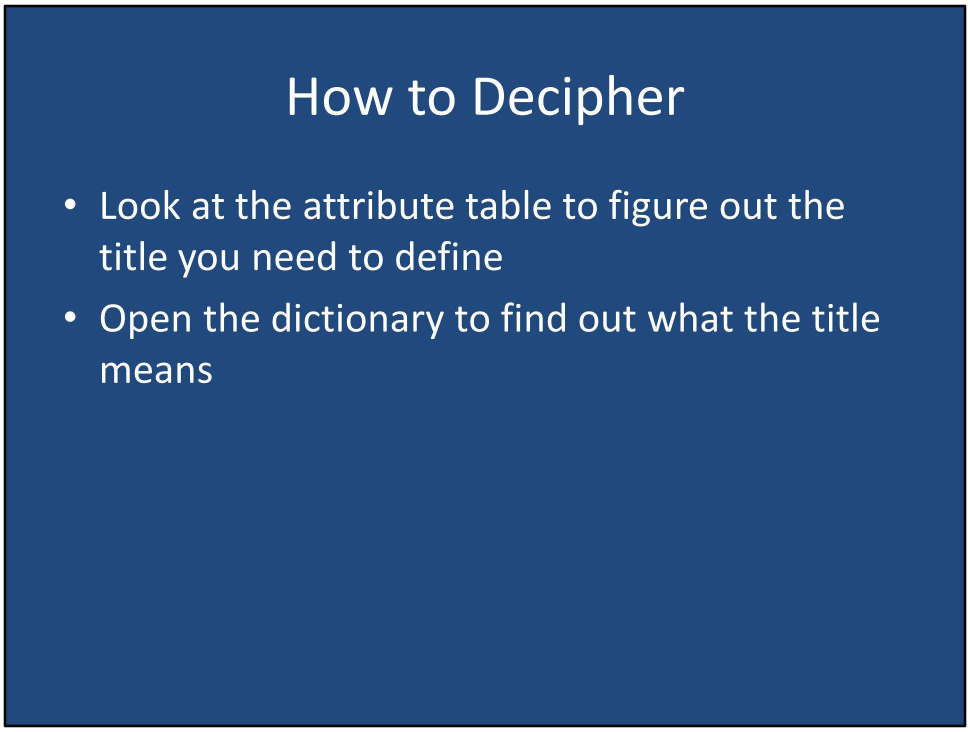
So natural question is, why do they do this? Does seem purposeless and a waste of time initially, as it would make more sense to just put the title for each field in the attribute table.



Well….

- Titles are sometimes too long, a proper explanation can’t be made in the space allotted
- Better to create a symbolized system
- This also makes it easier for querying (ie. When you’re searching for the total population of females it’s easier to just say DP0020003)

The reason why they use these sometimes hard to decipher field names, are that titles of fields can sometimes be too long to fit into the allotted space for that particular data. Additionally, a proper explanation can be made in the space allotted. Therefore, it is sometimes better to create a symbolized system which puts the coded value for the field name, and be cross-referenced to a data dictionary Winchell explained the field in more detail. This also makes it easier for querying. For example, if you’re searching for the total population of females, is easier to specify DP0020003 in your query.

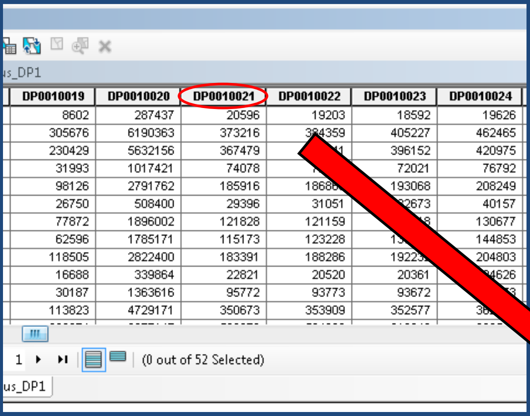


How to Decipher

- Look at the attribute table to figure out the title you need to define
- Open the dictionary to find out what the title means

To decipher the coded values, you should cross-reference the attribute tables field name to the data dictionary to find the definition. For example here, the field name DP0010021 represents males under 5 years of age as found in Excel file serving as the data dictionary.

Attribute Table:



us_DP1

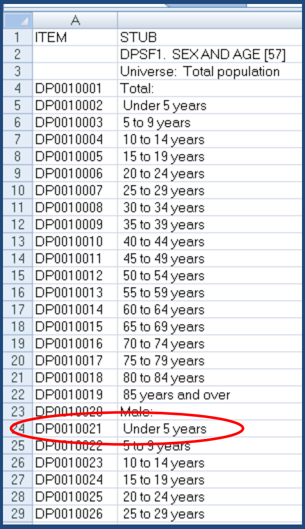
DP0010019	DP0010020	DP0010021	DP0010022	DP0010023	DP0010024
8602	287437	20596	19203	18592	19626
305676	6190363	373216	384359	405227	462465
230429	5632156	367479	[illegible]	396152	420975
31993	1017421	74078	[illegible]	72021	76792
98126	2791762	185916	[illegible]	193068	208249
26750	508400	29396	31051	[illegible]	40157
77872	1896002	121828	121159	[illegible]	130677
62596	1785171	115173	123228	[illegible]	144853
118505	2822400	183391	188286	[illegible]	204803
16688	339864	22821	20520	20361	[illegible]
30187	1363616	95772	93773	93672	[illegible]
113823	4729171	350673	353909	352577	[illegible]

1 (0 out of 52 Selected)

us_DP1

DP0010021 = Males under 5 years

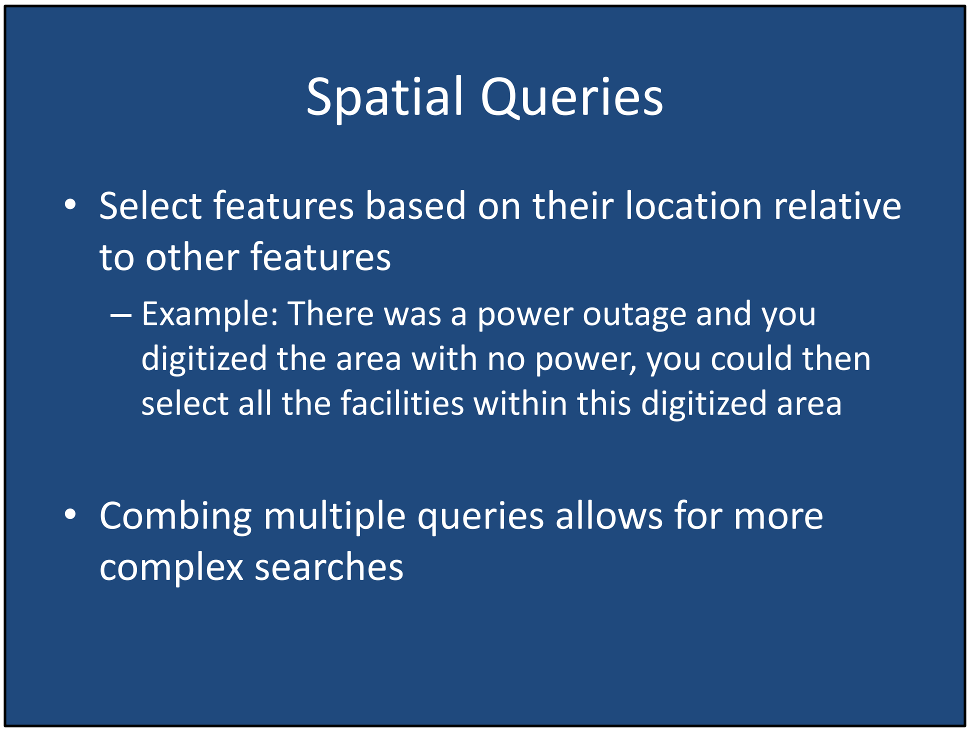
Excel File:



	A	
1	ITEM	STUB
2		DPSF1. SEX AND AGE [57]
3		Universe: Total population
4	DP0010001	Total:
5	DP0010002	Under 5 years
6	DP0010003	5 to 9 years
7	DP0010004	10 to 14 years
8	DP0010005	15 to 19 years
9	DP0010006	20 to 24 years
10	DP0010007	25 to 29 years
11	DP0010008	30 to 34 years
12	DP0010009	35 to 39 years
13	DP0010010	40 to 44 years
14	DP0010011	45 to 49 years
15	DP0010012	50 to 54 years
16	DP0010013	55 to 59 years
17	DP0010014	60 to 64 years
18	DP0010015	65 to 69 years
19	DP0010016	70 to 74 years
20	DP0010017	75 to 79 years
21	DP0010018	80 to 84 years
22	DP0010019	85 years and over
23	DP0010020	Male:
24	DP0010021	Under 5 years
25	DP0010022	5 to 9 years
26	DP0010023	10 to 14 years
27	DP0010024	15 to 19 years
28	DP0010025	20 to 24 years
29	DP0010026	25 to 29 years

Spatial Queries

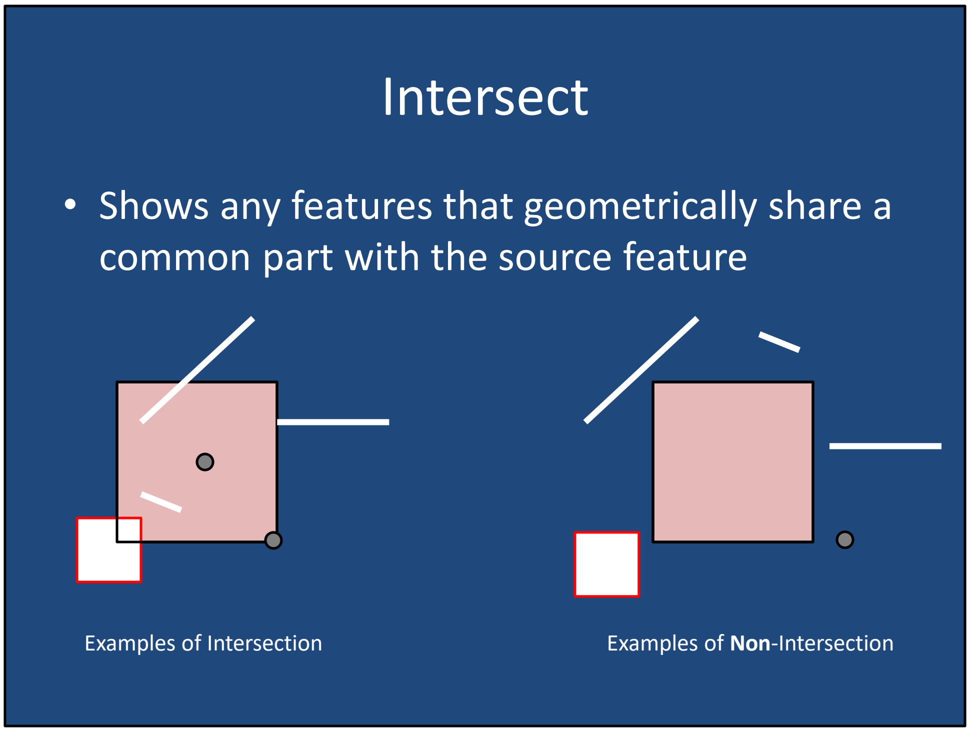
Spatial queries



Spatial Queries

- Select features based on their location relative to other features
  - Example: There was a power outage and you digitized the area with no power, you could then select all the facilities within this digitized area
- Combing multiple queries allows for more complex searches

The spatial query is where features are selected based on their location relative to other features. For example, let's say there was a power outage in you digitize the area with no power. You could use that digitize polygon to select all the facilities within this digitized area to perhaps dispatch utility crews. Much like attribute queries, you can combine multiple spatial queries to construct more complex queries. For the remainder of this section, specific types of spatial queries will be discussed.



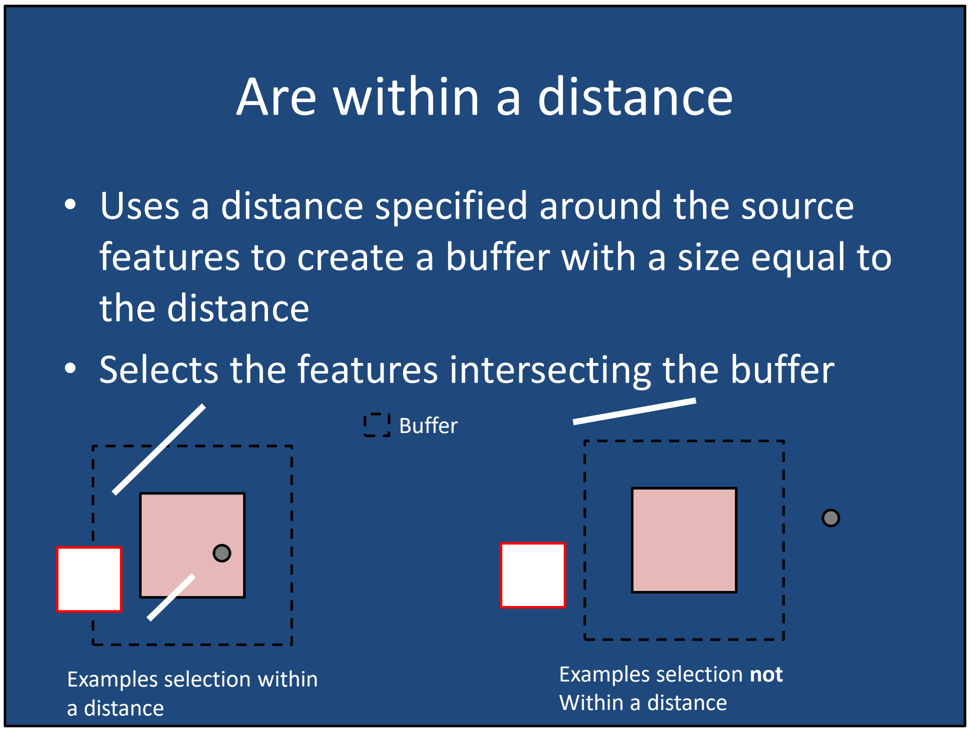
Intersect

- Shows any features that geometrically share a common part with the source feature

Examples of Intersection

Examples of **Non**-Intersection

The intersect spatial query selects any features that geometrically share a common part with the source feature. For the examples, the light red square serves as the source, selecting feature, and the points, white lines, and white fill polygon are the features available for selection. For the example of the intersection of the left, all 3 white lines, both points, and white polygon are selected as all 6 of these features touch in some way the light red polygon. On the right, as of the points line to polygons touch the source light red polygon, none of them are selected as no intersections exist.

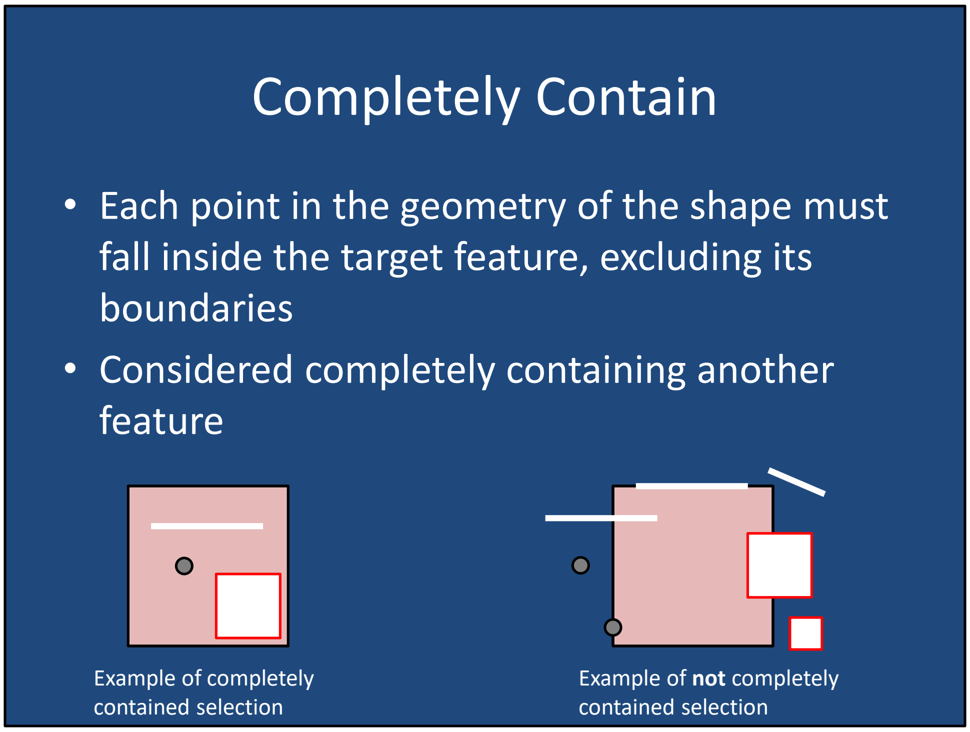


Are within a distance

- Uses a distance specified around the source features to create a buffer with a size equal to the distance
- Selects the features intersecting the buffer

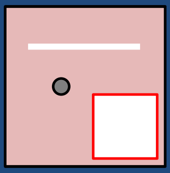
Buffer
Examples selection within a distance
Examples selection **not** Within a distance

The neck spatial query is "are within a distance". For the spatial query, it uses a distance specified around the source features to create a buffer with the size equal to the distance. The query then selects all features intersecting the buffer. For example on the left, the – line represents the buffer at a set distance from the light red source polygon. The 2 lines, point, and white fill polygon are all selected as they all touch in some way the buffer. On the right, we see examples where the point line and polygon are not selected as they do not intersect with the buffer.

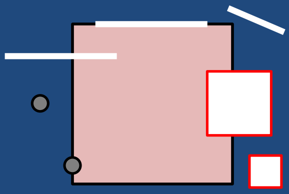


Completely Contain

- Each point in the geometry of the shape must fall inside the target feature, excluding its boundaries
- Considered completely containing another feature

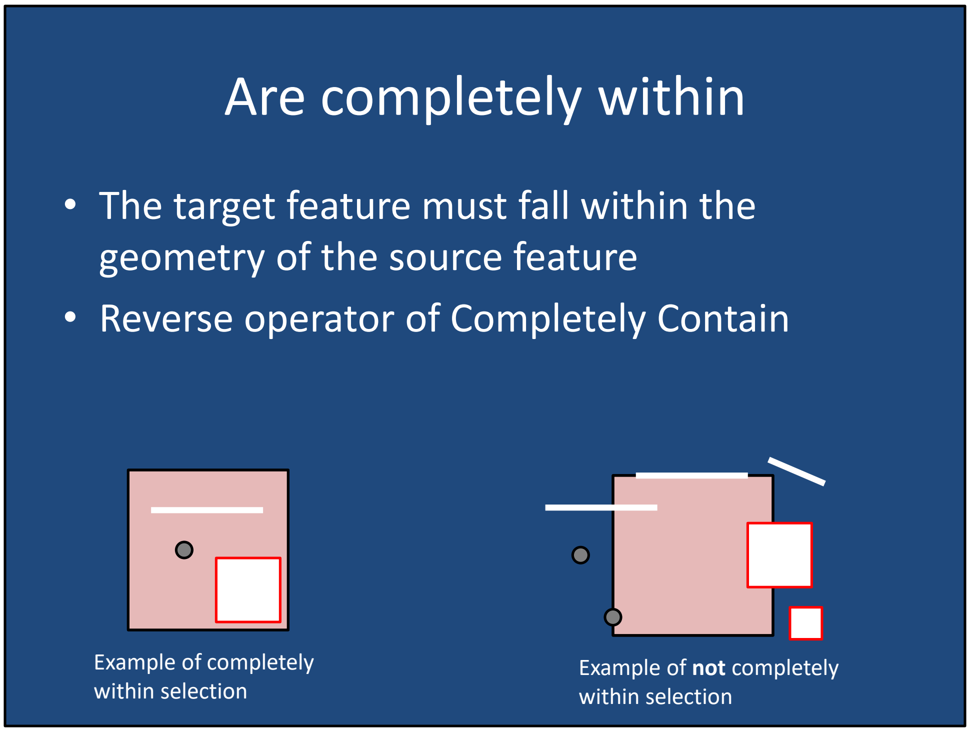


Example of completely contained selection



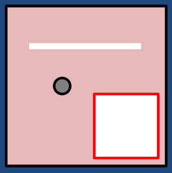
Example of **not** completely contained selection

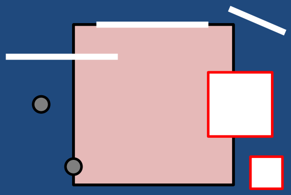
For the completely contain spatial selection, each point in the geometry of the shape must fall inside the target feature, including its boundaries. This is also considered to be when the source, selecting feature completely contains another feature. On the left, is an example where the light red source feature completely contains the point, white line, and white fill box. On the right, none of the points, lines, or polygons are completely contained within the source polygon, therefore none of them are selected. It is important to note the white line running across the top of the border of the source polygon, and the point on the bottom left edge of the source polygon. As both of these features exist on the boundary of the source polygon, they are not considered to be contained within, and are not selected.



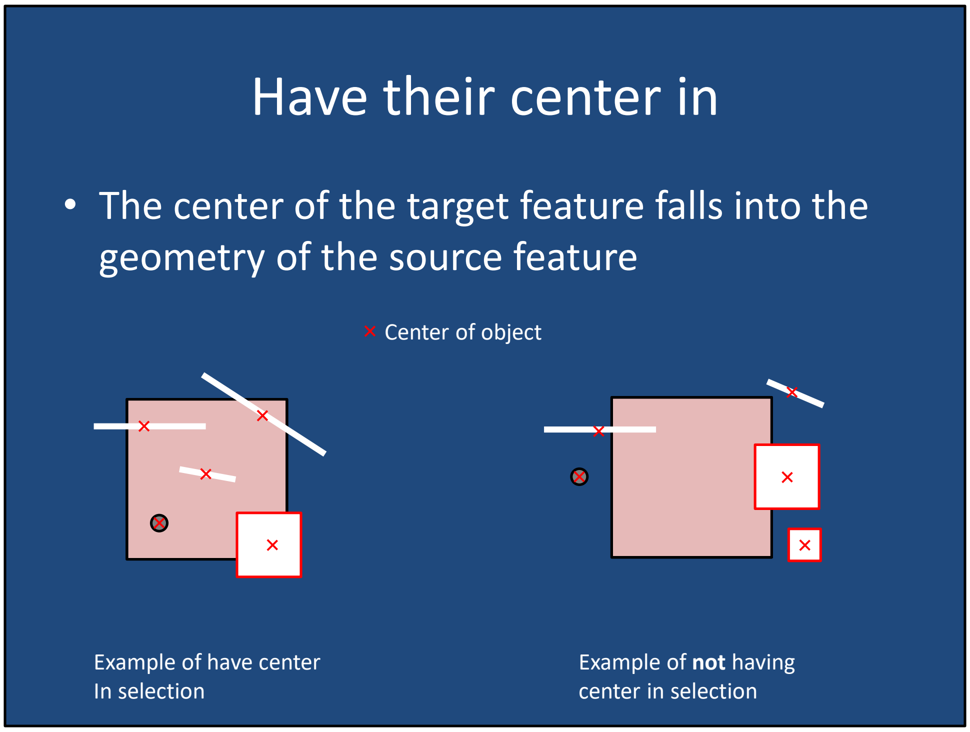
Are completely within

- The target feature must fall within the geometry of the source feature
- Reverse operator of Completely Contain


Example of completely within selection


Example of **not** completely within selection

The next spatial selection is "are completely within". For this spatial selection, target features must fall within the geometry of the source feature. This is the reverse operator of completely contain. For example on the left, the point, line, and white fill polygon all are completely contained within the light red source polygon, and therefore they are selected. On the right, none of the target features are completely within the light red source polygon, therefore none of them are selected.

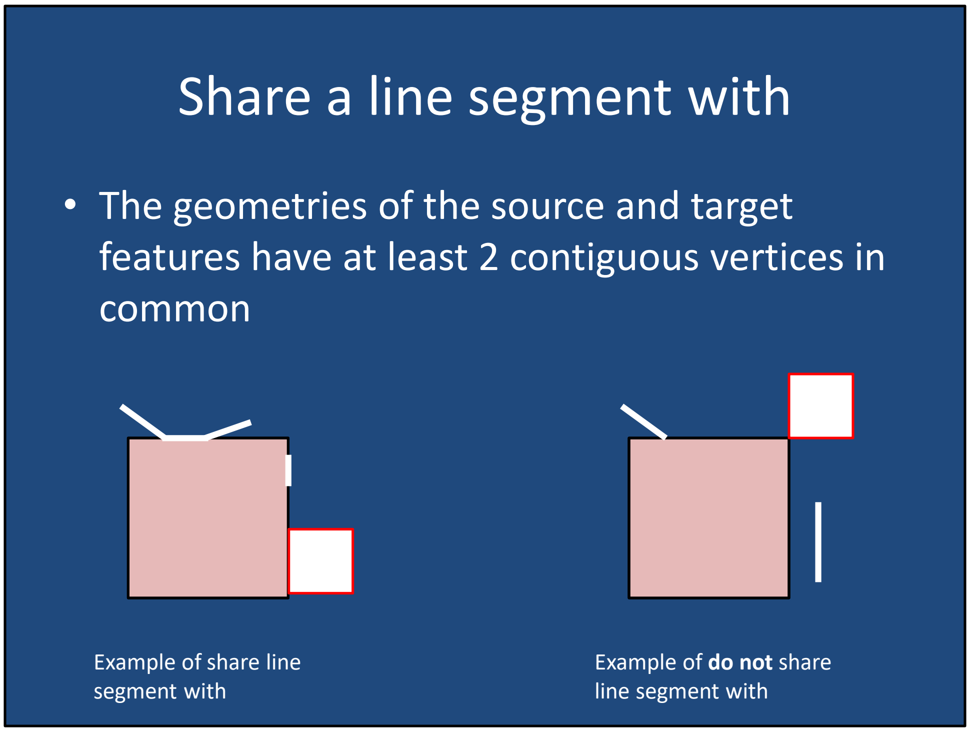


Have their center in

- The center of the target feature falls into the geometry of the source feature

× Center of object
Example of have center
In selection
Example of **not** having
center in selection

"Have their center in" selects all features weather center fall inside the geometry of the source feature. In our examples, the center of each feature is denoted by a red X. In our example on the left, all 3 lines, point, and white fill polygon, have their centers falling inside of the light red source polygon, therefore they were all selected. On the right, none of the target features have their center inside the source polygon, therefore none of them are selected.



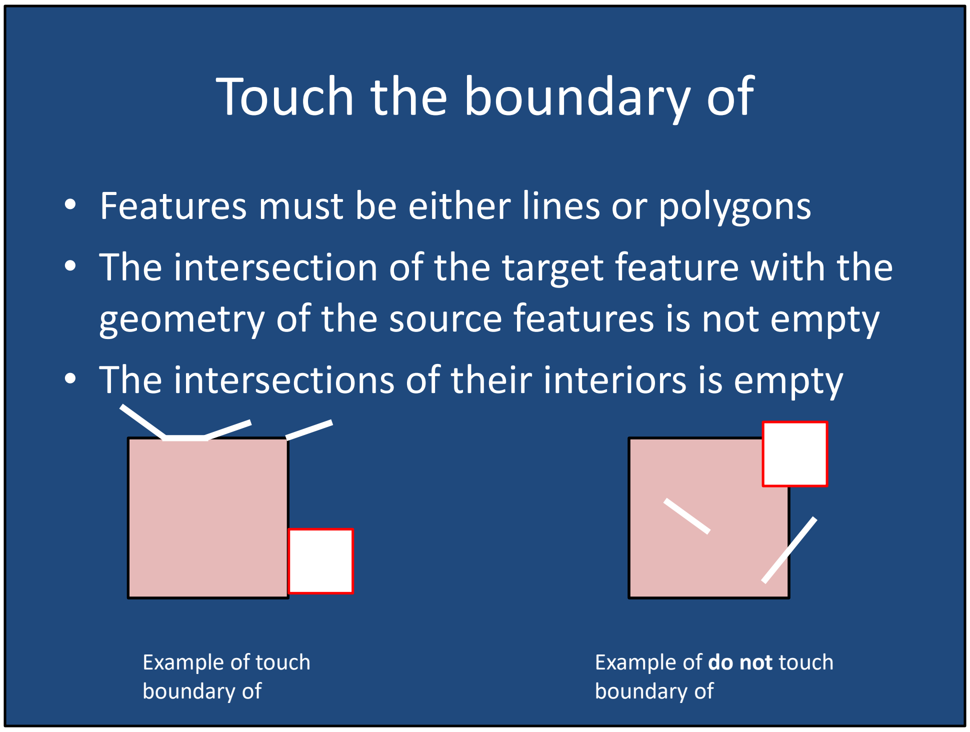
Share a line segment with

- The geometries of the source and target features have at least 2 contiguous vertices in common

Example of share line segment with

Example of **do not** share line segment with

The next spatial selection operator is “share a line segment with”. For the spatial selection operator, the geometries of the source and target features must have at least 2 contiguous vertices in common in order for the selection to take place. For example on the left, this is an example where the 2 lines and white fill box share line segments with the light red source polygon. In our example on the right, none of the features are selected, even though the features to have one vertex falling on the boundary of the source polygon, remember that they must share least 2 contiguous vertices, and simply touching at one point does not cause a selection.



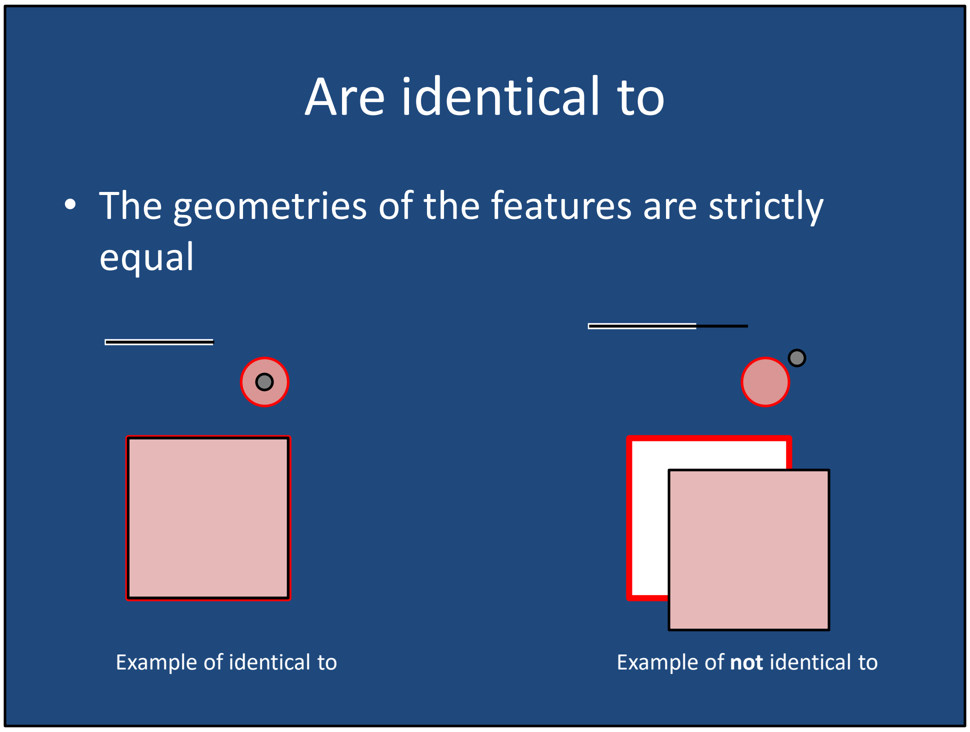
Touch the boundary of

- Features must be either lines or polygons
- The intersection of the target feature with the geometry of the source features is not empty
- The intersections of their interiors is empty

Example of touch boundary of

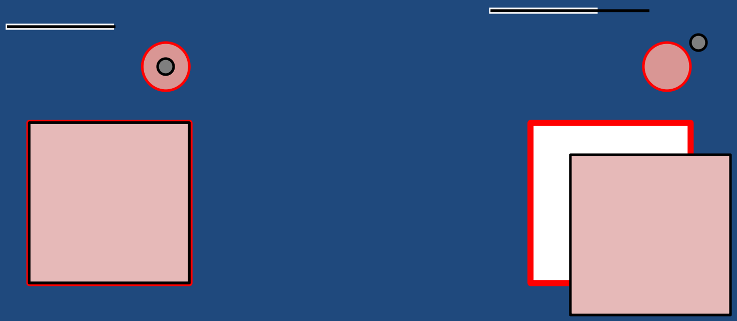
Example of **do not** touch boundary of

The “touch the boundary of” spatial selection operator selects polygon or line features where the intersection of the target feature with the geometry of the source feature is not empty. The intersections of their interiors must be empty otherwise it will not be selected. For example, on the left, the 2 lines, and white box touch the boundary of the source polygon, but neither of the lines where the box extend inside of the source polygon. Therefore, both lines and the box are selected. For example on the right, since the 2 lines and white box extruded into the geometry of the source polygon, they are not selected even though they touch the boundary.



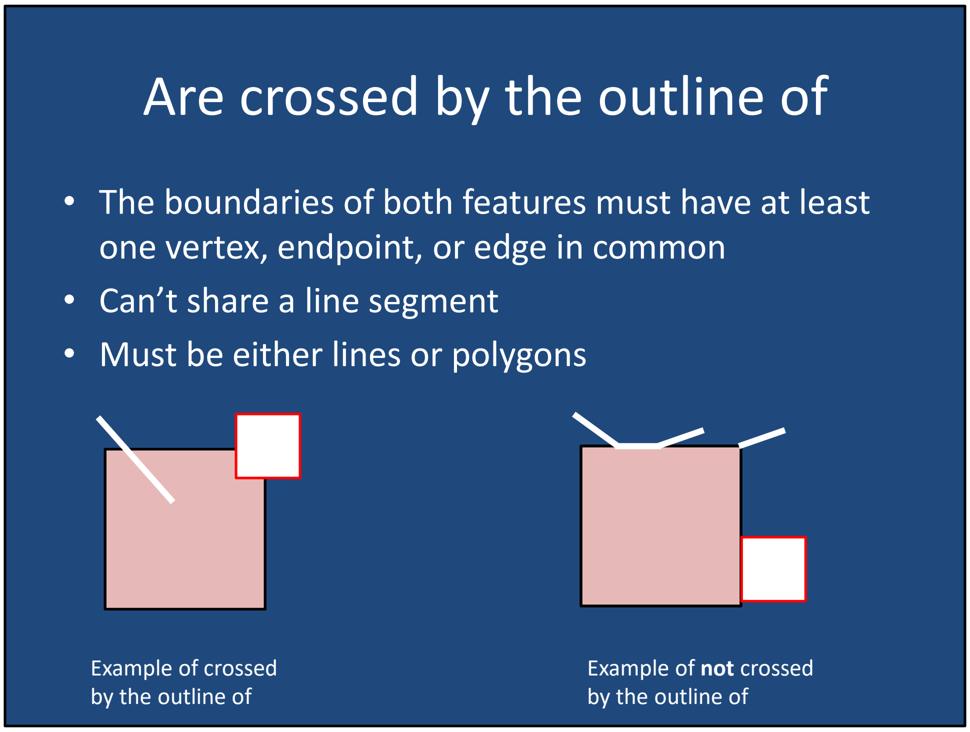
Are identical to

- The geometries of the features are strictly equal



Example of identical to   Example of **not** identical to

The "are identical to" spatial selection operation selects only geometries of features that are strictly equal. For example, on the left, the thin black line in figure white line both have exactly the same vertices, and are considered equal. The fact that they are symbolized differently is the relevant for the selection, as it is the location of the nodes and vertices that determine whether the features are identical. This applies to the 2 points, the light red circle and the gray circle have the same center point, and are considered to be identical. Last, the light red box with the black outline shares the exact same vertices as the white fill the box with red outline. On the right, are examples where the geometry between these points line to polygons are not identical, therefore no selections occur.



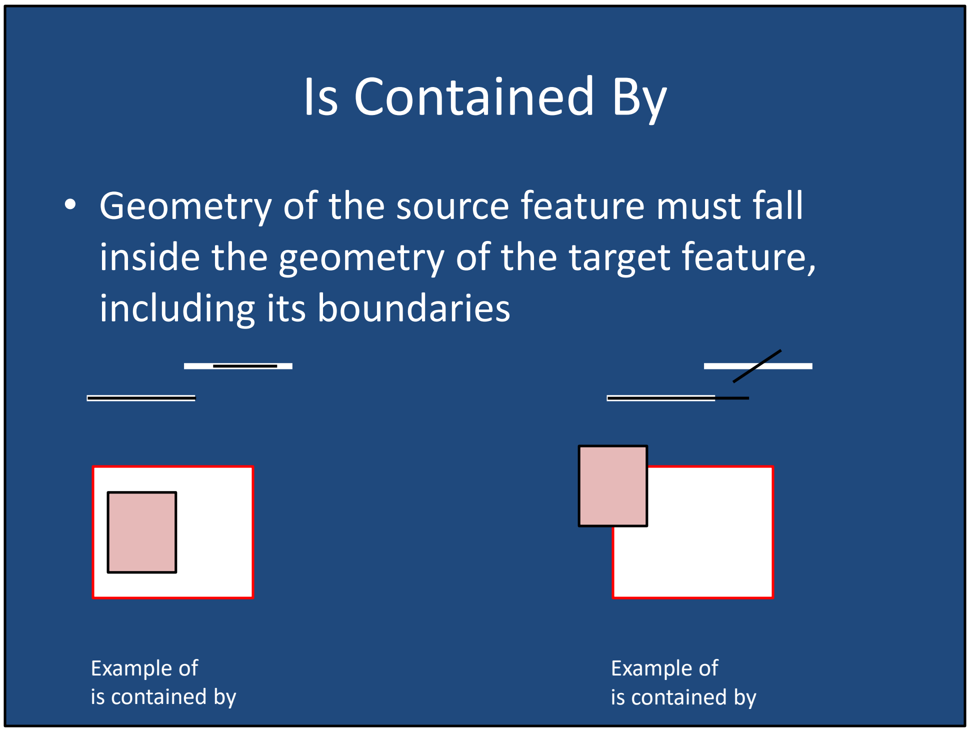
Are crossed by the outline of

- The boundaries of both features must have at least one vertex, endpoint, or edge in common
- Can't share a line segment
- Must be either lines or polygons

Example of crossed by the outline of

Example of **not** crossed by the outline of

For the "are crossed by the outline of" spatial selection operator, the boundaries of both features must have at least one vertex, and point, or edging common for a selection to take place. It is important to note that the target and source features cannot share a line segment, and must be either lines or polygons. In our example of a successful selection, the white line and white fill box both cross the outline of the light red source polygon and do not share any line segments. On the right, the line on the top of the source polygon sharing a line segment is not selected, the line of top right is not selected because it does not cross the boundary of the source polygon, and the white box is not selected because it is only sharing a line segment along the boundary.



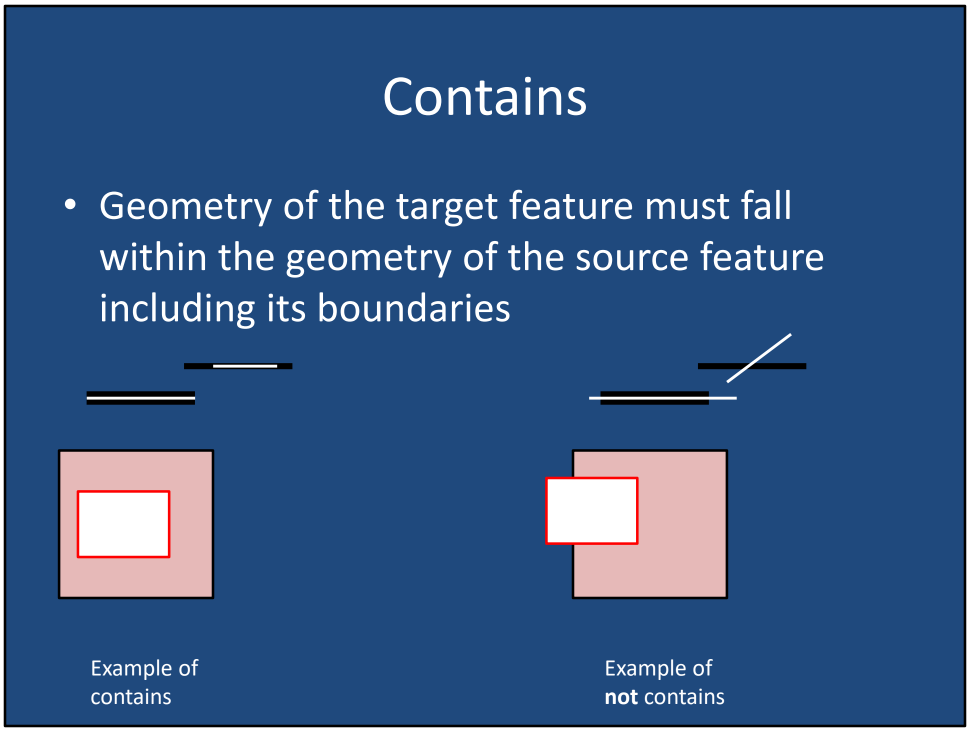
Is Contained By

- Geometry of the source feature must fall inside the geometry of the target feature, including its boundaries

Example of
is contained by

Example of
is contained by

“Is contained by” selects target features where the geometry of the source feature falls inside the geometry of the target feature including its boundaries. On the left, the thin black line is considered the source feature and is completely contained within the longer white line, therefore the white line is selected. The light red source box is completely contained within the white fill box, therefore the white fill box is selected. On the right, as the black source lines extend past the extents of the white line, no selections occur. Similarly, as the light red source polygon extends across the boundaries of the target white fill box feature, no selection occurs there either.



Contains

- Geometry of the target feature must fall within the geometry of the source feature including its boundaries

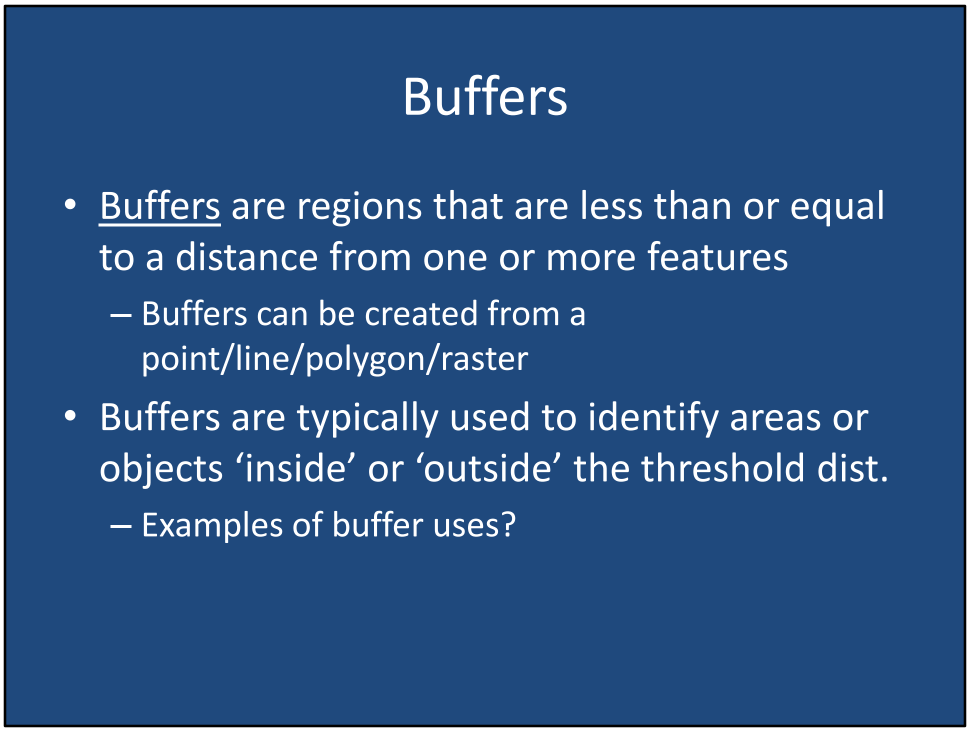
Example of contains

Example of **not** contains

Finally, the "contains" spatial selection operator selects geometry of the target feature that falls within the geometry of the source feature including its boundaries. Our left examples of containment all results in selections because the target features are within the source features represented by the black lines, and light red polygon. On the right, we see examples where the target features are not contained, and therefore, are not selected.

Buffering

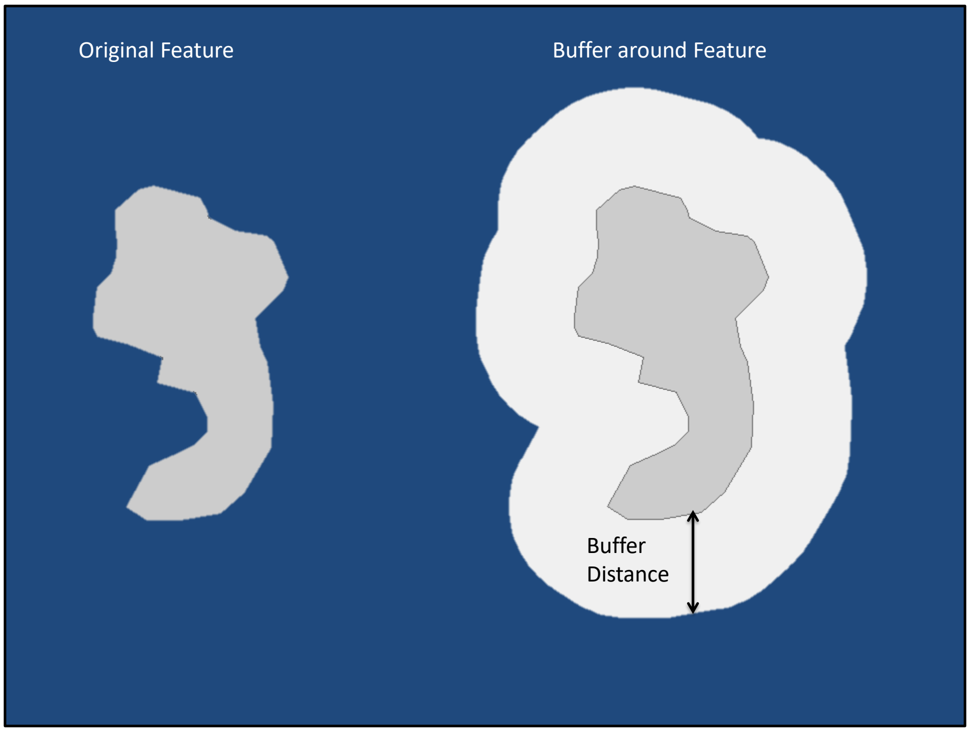
Buffering



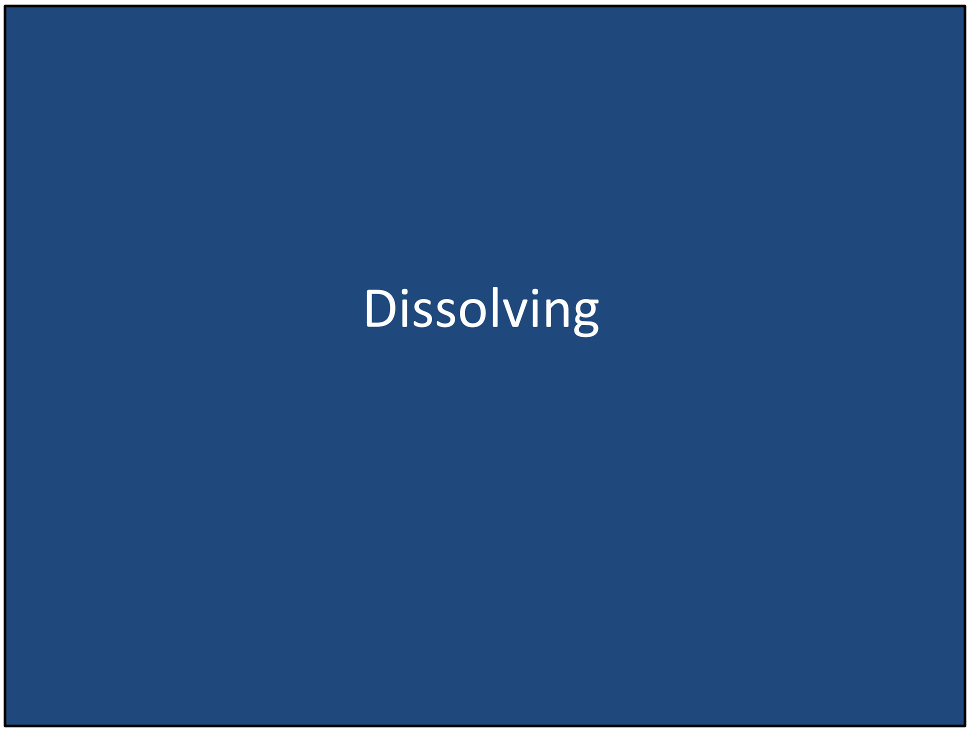
Buffers

- Buffers are regions that are less than or equal to a distance from one or more features
  - Buffers can be created from a point/line/polygon/raster
- Buffers are typically used to identify areas or objects 'inside' or 'outside' the threshold dist.
  - Examples of buffer uses?

A buffer is a region that is less than or equal to a distance from one or more input features. Buffers can be created from point/line/polygon/raster geospatial data sets. Buffers are typically used to identify areas or objects "inside", or "outside" the threshold distance. Can you think of any examples where you might use a buffer? One example of a use of a buffer could be to determine how many homes are within 1 mile of the coastline. In this case, the coastline is the input, 1 mile is the threshold distance, which results, in a polygon that extends 1 mile out from the coastline. We can then performing spatial selection by selecting all houses that are inside the buffer.



To illustrate a buffer, the polygon to the left is the original input features. After specifying a buffer distance, and running the buffer tool, a new polygon is created at the set buffer distance and surrounds and follows the outline of the original input feature.

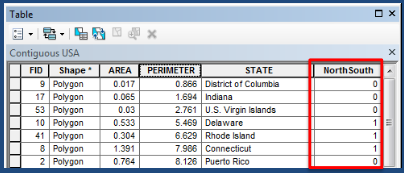


Dissolving

Dissolving

Dissolve

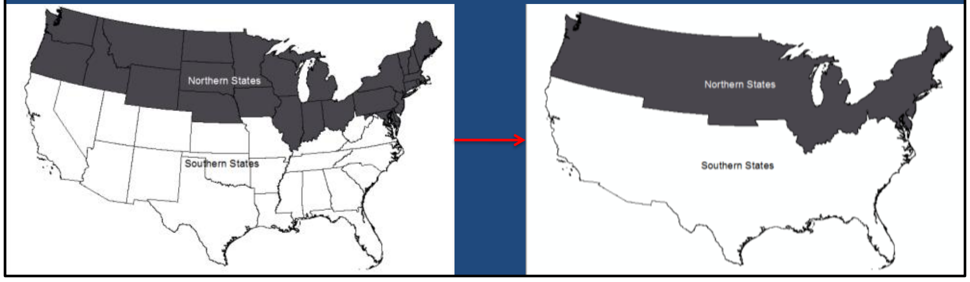
Dissolve combines similar features within a data layer based on an attribute



Table

Contiguous USA

FID	Shape *	AREA	PERIMETER	STATE	NorthSouth
9	Polygon	0.017	0.866	District of Columbia	0
17	Polygon	0.065	1.694	Indiana	0
53	Polygon	0.03	2.761	U.S. Virgin Islands	0
10	Polygon	0.533	5.469	Delaware	1
41	Polygon	0.304	6.629	Rhode Island	1
8	Polygon	1.391	7.986	Connecticut	1
2	Polygon	0.764	8.126	Puerto Rico	0



The dissolve operation combines similar features within a data layer based on a shared attribute. For example, the data set on the left has the attribute of north-south. If the state is a northern state, and has the value of one, and if the state is a southern state, it is a value of zero. If we use this data set, and dissolve on this field, all states that have the value of one, are dissolved into a single polygon. All states that have the value of zero for the north-south attribute, are dissolved into a second single polygon. The dissolve field can have as many different values as you choose, however, only records with identical values for that attribute will be combined into a single feature.