Lesson 3: Using Attribute and Spatial Queries for Data Exploration

## INTRODUCTION

In this lesson you will learn about using attribute and spatial queries for data exploration. Information presented includes set algebra and Boolean algebra. In this lesson you will also gain an understanding the value of data dictionaries and deciphering coded values. You will also learn about buffering and dissolving as well as the importance of each as they relate to spatial queries and data exploration.

## LESSON OBJECTIVES

By the end of this lesson, you will be able to:

1. Perform advanced query to prepare data for use in analysis.

2. Use a data dictionary to decipher coded data.

3. Determine how to use queries to address a question.

## LEARNING SEQUENCE

|  |  |
| --- | --- |
|  | |
| Required Reading | Read the following:  Using Attribute and Spatial Queries for Data Exploration   * Set Algebra * Boolean Algebra * Data Dictionaries * Spatial Queries * Buffering * Dissolving |
| Assignments | Complete the following:   * Lab: Using Attribute and Spatial Queries for Data Exploration * Quiz: Using Advanced Attribute and Spatial Queries for Data Exploration |

## INSTRUCTION

**Using Attribute and Spatial Queries for Data Exploration**

## Attribute Queries

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. An example of an attribute query would be 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 Operations

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.

|  |
| --- |
| *< Less than*  *> Greater than*  *= Equal to*  *< > Not equal to*  *> and < may not be applied to nominal attributes* |

* 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 is represented by both the left angle bracket and right angle bracket used together.

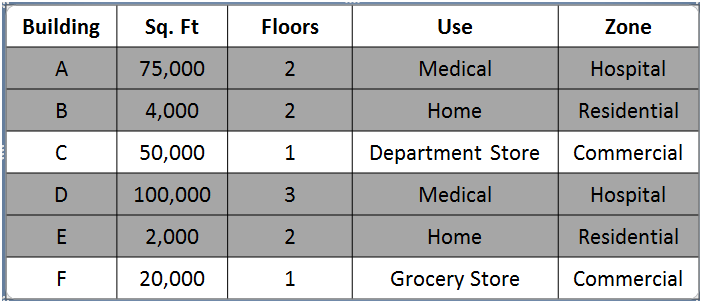
## Applied Alone or in Combination

The three operations: greater than, less than, and equal to 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 results 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? 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. Let’s look at some examples of table queries using simple set algebra operations.

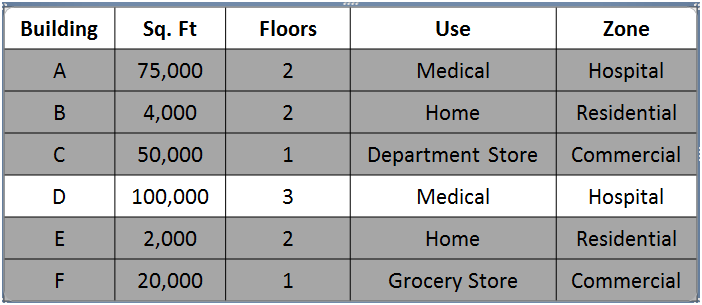
**Floors > 1**

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 greater than the number 1.



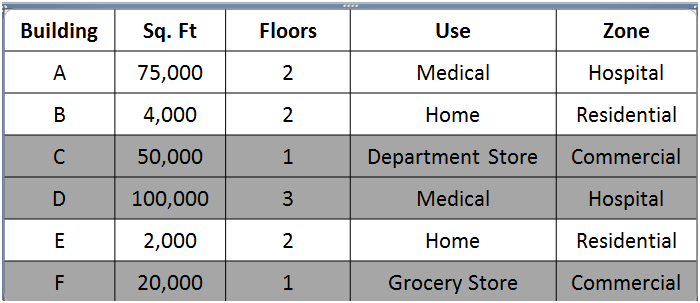
**Floors <= 2**

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 D 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**

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



**Boolean Algebra**

## Multiple Conditional Operations

Boolean algebra has multiple conditional operators. The three most common Boolean operators are:”OR”, ”AND”, and”NOT”. The Boolean operators evaluate values on the left and right side of the operator, and then assign 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.

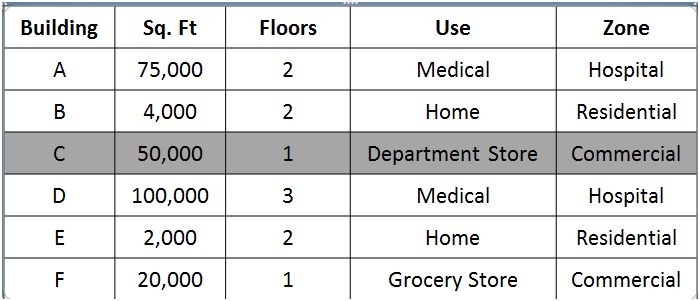
|  |
| --- |
| **Key Facts**  **Multiple Conditional Operations**   * are *OR, AND,* and *NOT* * are evaluated by assigning a true or false outcome to each condition * do not follow a specific order of operations * are not distributable |

## 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 two, or both criteria, does 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.

Consider an example using the buildings attribute table provided. Note that our statement has two 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 of these two queries is 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.

**Zone = Commercial AND Sq. Ft > 40,000**

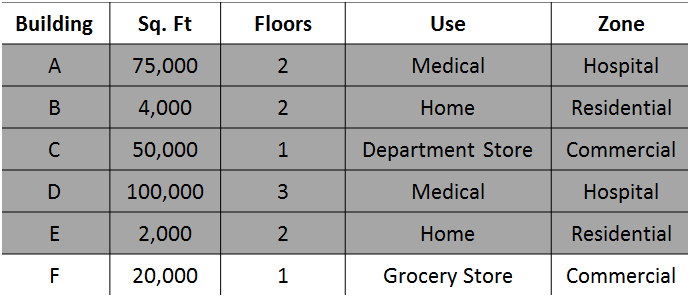


**OR Boolean Operator**

The “OR” Boolean operator again considers the results of two 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 two attribute queries and it or Boolean operator, is if both attribute queries return false.

Consider the statement that reads floors > 1 OR use = 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 two queries on each side of the “OR” Boolean operator.

**Floors > 1 OR Use = Department Store**

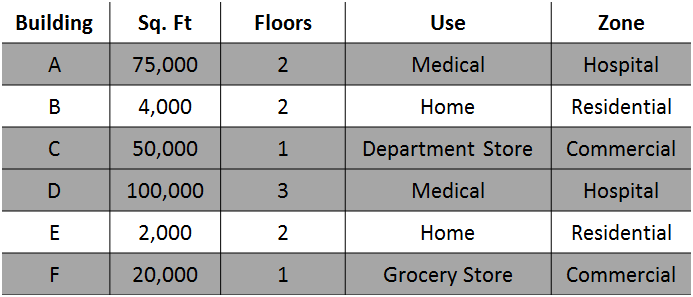


**NOT Boolean Operator**

The “NOT” Boolean operator selects attributes that do not meet the attribute query following the “NOT” Boolean operator. In other words, 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 to false and vice versa.

Consider the query “NOT” zone equals residential. Parentheses placed around zone equals residential help clarify the Boolean operator and the attribute query. For the selection, each record is evaluated on the zone attribute to see if it is equal to residential. If the zone equals residential, it will return a value of true. When we apply the “NOT” Boolean operator to this query, we would select all records not returned for the initial query or those records that had values not equal to residential Therefore, you could read the statement as select all buildings 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.

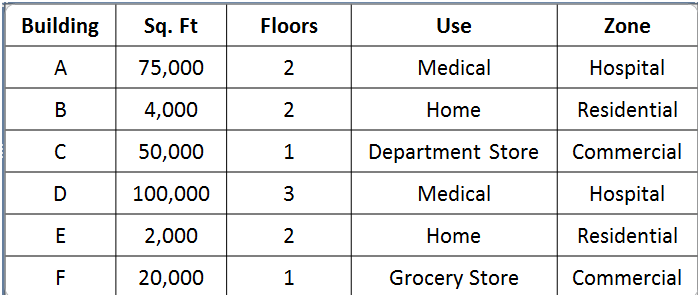
**Zone NOT Residential**



## Set and Boolean Algebra Practice

Now it is time to practice. Complete Situations 1 and 2.

**Situation 1:** Take a few moments to determine what an appropriate attribute query would be to look for a home with more than 2,000 square feet using the Attribute Table: Buildings A – F.

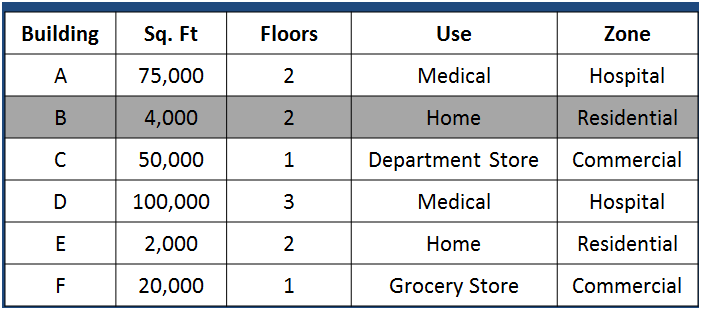
Attribute Table: Buildings A – F 

**Situation 1 Answer**

Situation 1: Looking for a home with more than 2,000 square feet.

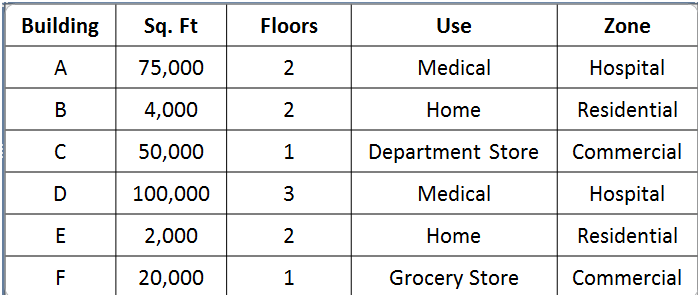
Based on the situation where we want to find a home with more than 2000 square feet and appropriate attribute query we query on two different fields, use, and square feet. As we want both attributes to be true in order to select the record, we use the “AND” Boolean operator. Therefore, our attribute query reads use equals home “AND” square feet are greater than 2000. This selects building B as it is a home and has more than 2000 feet. It does 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

**Use = Home AND Sq. Ft > 2,000**



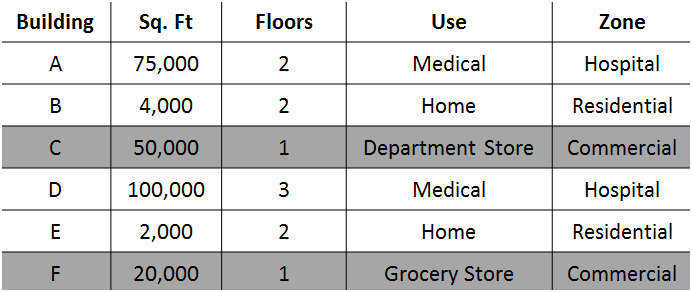
**Situation 2:** Take a few moments to determine what an appropriate attribute query would be to look for a commercially zoned building that has more than 10,000 square feet and more than one floor. Use the provided Attribute Table: Buildings A – F.

**Attribute Table: Buildings A – F**



**Situation 2: Answer**

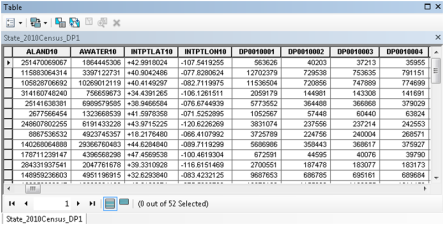
For this attribute query, we are using 3 attribute queries on 3 different attribute fields, and two Boolean operators. The query used here to satisfy the situation is Zone = Commercial AND Square Footage > 10,000 OR Floors > 1. To evaluate this attribute query we must keep in mind the order of operations for Boolean operators. “NOT” is evaluated before “AND” is evaluated before “OR”. Once true or false values have been evaluated for these queries, these two values should be evaluated according to the rules associated with the “AND” Boolean operator. Once this true or false value has been determined, it should be placed to the left of “OR” floors greater than 1. Buildings C and F are the only two building selected as a both meet the requirements of the zone equaling commercial and square footage greater than 10,000. Note that building C and F 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 C and F buildings still meet the criteria set forth in the statement.



**Data Dictionaries**

## Importance of Data Dictionaries

Often times an 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 its meaning. The purpose of the data dictionary is to provide the descriptive field names for attribute tables. The data dictionaries are often provided in a text file format (.txt) or Microsoft Excel format (.xls).

As an example, here is 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.

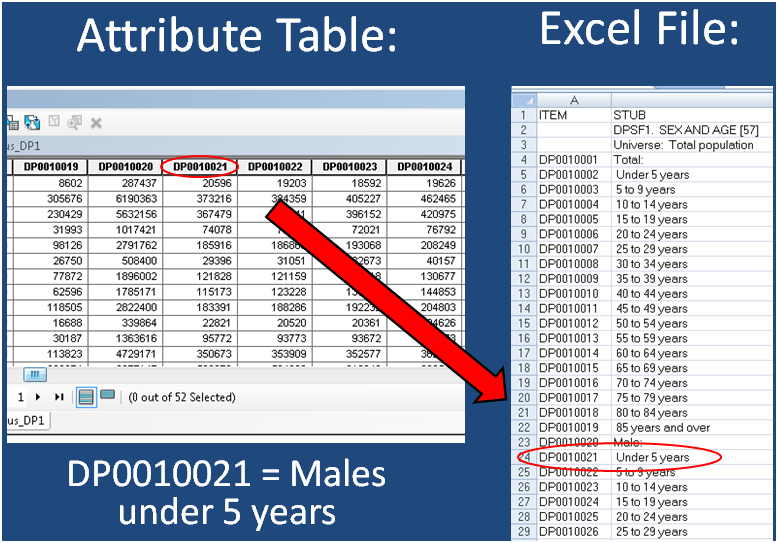
So the natural question is why do they do this?

The reasons why they use these sometimes hard to decipher field names are:

* Sometimes titles of fields can be too long to fit into the allotted space for that particular data. Additionally, a proper explanation can be made in the space allotted.
* 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 which will explain the field in more detail. This also makes it easier for querying. For example, if you are searching for the total population of females, it is easier to specify DP0020003 in your query.

## Deciphering Coded Values

To decipher the coded values, you should cross-reference the attribute table’s field name with the data dictionary to find its definition. For example, in this image, the field name DP0010021 represents males under the age of five as found in Excel file serving as the data dictionary.



**Spatial Queries**

## What are Spatial Queries?

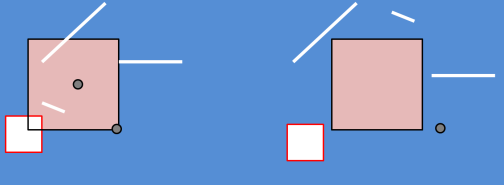
The spatial query is where features are selected based on their location relative to other features. For example, let’s say there was a power outage and you digitized the area with no power. You could use that digitized polygon to select all the facilities within this digitized area to perhaps dispatch 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.

|  |
| --- |
| **Key Facts**  **Spatial Queries**   * Select features based on their location relative to other features * Combining multiple queries allows for more complex searches |

## “intersect”

The “intersect” spatial query selects any features that geometrically share a common part with the source feature. For the examples, the light red squares serves as the source, selecting feature, and the points, white lines, and white fill polygon are the target features available for selection.

In the image on the left, all three white lines, both points, and the white polygon are selected. All six of these features touch in some way within the light red polygon. In the image on the right, the point and lines to the polygon do not touch the source light red polygon; none of them are selected as no intersections exist.

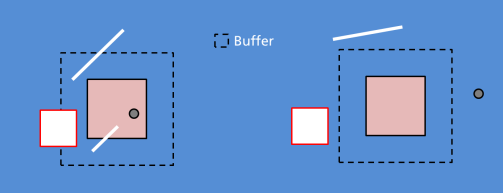


Examples of intersections Examples of non-intersection

## “are within a distance”

The next spatial query is “are within a distance”. For this particular spatial query, it is necessary to specify a distance from the source feature in order to create a buffer. The query then selects all features intersecting the buffer.

For example in the image on the left, the dashed line represents the buffer at a set distance from the light red source polygon. The two lines, point, and white fill polygon are all selected as they all touch the buffer in some way. In the image on the right, we see examples where the point, line and polygon are not selected as they do not intersect with the buffer.



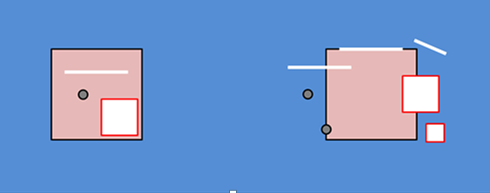
Examples of selection within a distance Examples of selection not within a distance

## “completely contain”

For the “completely contain” spatial selection, each point in the geometry of the target features must fall inside the source feature, including its boundaries. This is also considered to be when the source or selecting feature completely contains another feature.

The image on the left is an example where the light red source feature completely contains the point, white line, and white fill box. In the image 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.



Example of completely contained section Example of not completely contained section

## “are completely within”

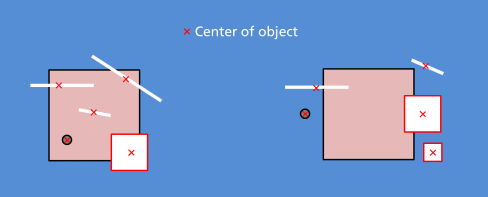
The next spatial selection is “are completely within”. For this spatial selection, target features must fall within the geometry of the source feature. This is the reverse operator of completely contain. For example on the left, the point, line, and white fill polygon all are completely contained within 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.

## 

Example of completely within selection Example of a not completely within selection

## “have their center in”

“Have their center in” selects all features whether their centers 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 three 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.

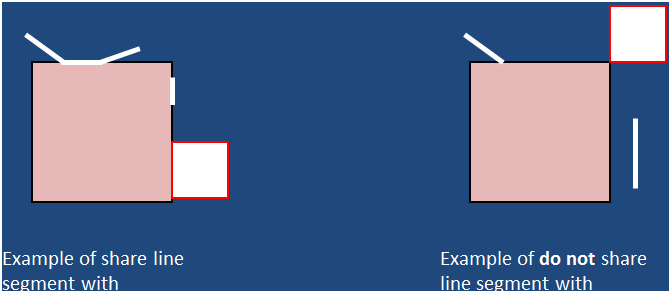


Example of have center in selection Example of not having center in selection

“share a line segment with”

The next spatial selection operator is “share a line segment with”. For this spatial selection operator, the geometries of the source and target features must have at least two contiguous vertices in common in order for a selection to take place.

The example on the left depicts a scenario where all features would be selected because the two lines and the 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 do have one vertex falling on the boundary of the source polygon, remember that they must share least two contiguous vertices, and simply touching at one point does not cause a selection.

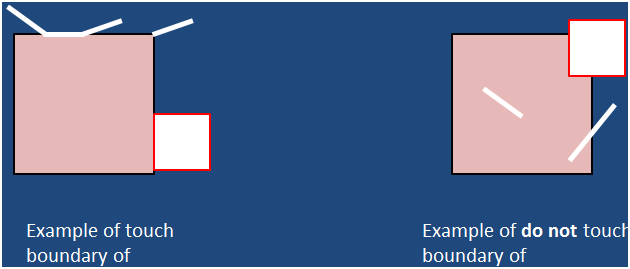


## “touch the boundary of”

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 they will not be selected. For example, on the left, the two lines, and white fill box touch the boundary of the source polygon, but neither of the lines or the box extend inside of the source polygon.

Therefore, both lines and the box are selected. For example on the right, since the two lines and white box extruded into the geometry of the source polygon, they are not selected even though they touch the boundary of the source feature.

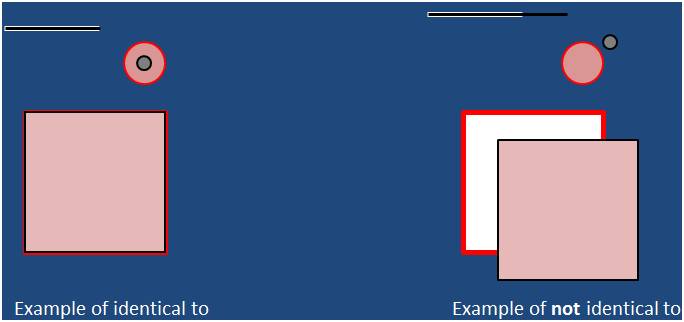


## “are identical to”

The “are identical to” spatial selection operator selects only geometries of features that are strictly equal. For example, on the left, the thin black line within the figure of the white line both have exactly the same vertices, and are considered equal.

The fact that they are symbolized differently is not relevant for the selection, as it is only the location of the nodes and vertices that determine whether the features are identical. This applies to the two points; the light red circle and the gray circle have the same center point, and are considered to be identical.

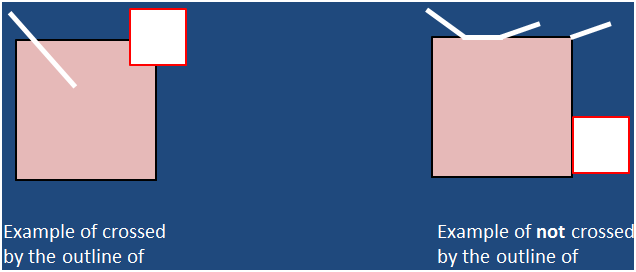
Lastly, the light red box with the black outline shares the exact same vertices as the white fill the box with red outline. The figure on the right provides examples where the geometry between these points, lines, and polygons are not identical, therefore no selections occur.



## “are 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 the example on the left, the white line and white fill box both cross the outline of the light red source polygon and do not share any line segments. In the example on the right, the line on the top of the source polygon sharing a line segment is not selected, the line in the of top right is not selected because it does not cross the boundary of the source polygon, and the white fill box is not selected because it is only sharing a line segment along the boundary.

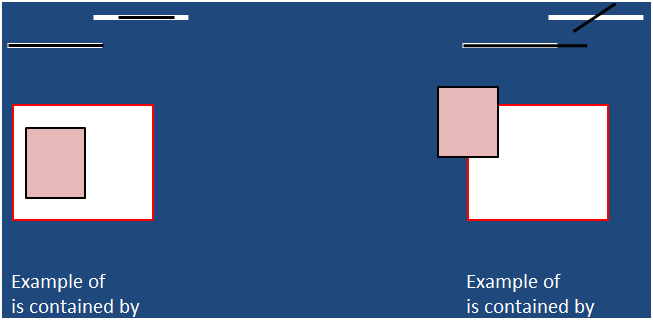


## “is contained by”

“Is contained by” selects target features where the geometry of the source feature falls inside the geometry of the target feature including its boundaries.

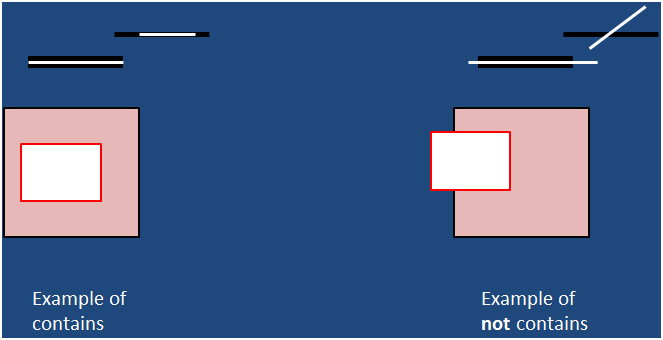
In the image on the left, the thin black line is considered the source feature and is completely contained within the longer white line; therefore the white line is selected. The light red source box is completely contained within the white fill box; therefore the white fill box is selected.

In the image on the right, as the black source lines extend past the extents of the white lines, 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”

The “contains” spatial selection operator selects geometry of the target feature that falls within the geometry of the source feature, including its boundaries. The image on the left serves as an example where all target features are selected because the source features (represented by the black lines and the light red polygon) encapsulate the target features (represented by the white lines and the white fill box). In the image on the right, we see examples where the target features are not contained, and therefore, are not selected.

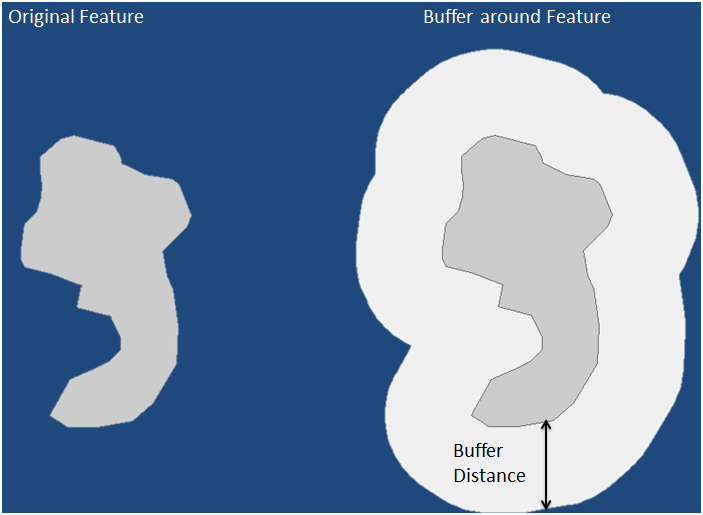


**Buffering**

## Buffers

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 points, lines, polygons, and 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 could be to determine how many homes are within one mile of the coastline. In this case, the coastline is the input and one mile is the threshold distance, which results in a polygon that extends one mile out from the coastline. We can then utilize spatial selection to select all houses that are inside the buffer.

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

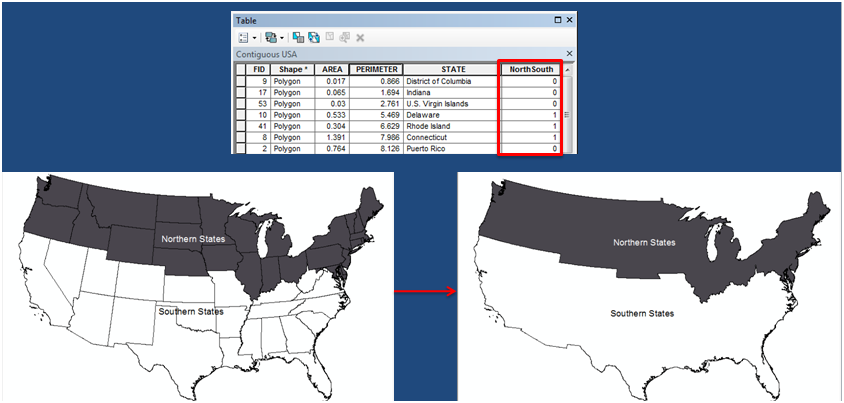


**Dissolving**

## Dissolve

The dissolve operation combines similar features within a data layer based on a shared attribute. For example, the data set used to represent the map on the left has the attribute of north-south. If the state is a northern state it has the value of one, and if the state is a southern state it has a value of zero.

If we use this data set, and dissolve on the north-south attribute field, the result would be demonstrated by the map on the right. 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.



## SUMMARY

This lesson focused on using attributes and spatial queries for data exploration. You learned about attribute queries and how to create them as well as the role algebra has in this process. The lesson also provided information on data dictionaries, deciphering coded values, and examples of spatial queries to understand how to apply this knowledge.

## ASSIGNMENTS

1. Lab: Using Attribute and Spatial Queries for Data Exploration

2. Quiz: Using Advanced Attribute and Spatial Queries for Data Exploration